SIEMENS

SIMOTION

SIMOTION Manual Collection

Application Manual

07/2021

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ **DANGER**
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ **WARNING**
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ **CAUTION**
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ **WARNING**
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

Ⓟ 06/2021 Subject to change

Table of contents

SIMOTION Documentation 1

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

2

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Engineering System Handling 3

3.1 CamTool

Preface

Contents of manual

This manual describes the SIMOTION CamTool optional package.

This document is part of the SIMOTION Engineering System documentation package.

Scope

This manual applies to SIMOTION SCOUT in conjunction with the SIMOTION CamTool optional package.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Information:

The following is a description of the purpose and use of the manual:

- "Description" section
  This section provides a brief overview of the basic functions of SIMOTION CamTool and its integration into SIMOTION SCOUT.
- "Installation" section
  This section explains how SIMOTION CamTool is installed and the requirements which need to be met before it can be used.
- "Planning/Configuring" section
  This chapter describes the basic functions of SIMOTION CamTool, and contains information on how to use the CamTool when editing cams.
- "Functions" section
  This section explains how to create a cam.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

3.1.1 Fundamental safety instructions

3.1.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

3.1.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

3.1.1.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

3.1.2 Description

Introduction

The graphical user interface in SIMOTION CamTool allows you to create, edit and optimize cams.

SIMOTION CamTool is fully integrated in SIMOTION SCOUT. This allows you to also use in SIMOTION CamTool information configured in SIMOTION SCOUT (e.g. axis settings).

Basic functions

SIMOTION CamTool provides the following basic functions:

- Insert and edit cams.
  Cams can be added to a SCOUT project using SIMOTION CamTool. In addition, you can edit with CamTool a cam created with CamEdit: Cams can also be imported from a text file or uploaded from a SIMOTION device.
- Modifying the representation of the cam in CamTool
  In SIMOTION CamTool, you can show and hide diagrams, change the representation parameters of the axes and diagrams, and adapt the lines and fonts. You can also represent auxiliary lines in the diagram.
- Converting cams from SIMOTION CamTool to SIMOTION CamEdit.
  To edit with SIMOTION CamEdit a cam edited in SIMOTION CamTool, the cam must be converted.
- Exporting cams into a text file.
- Downloading cams to a SIMOTION device
- Printing a cam.

3.1.3 Installing the Software

3.1.3.1 Installing SIMOTION CamTool

Introduction

SIMOTION CamTool is an optional package for SIMOTION SCOUT.

SIMOTION SCOUT must already be installed on the system on which you want to install SIMOTION CamTool. For further information, refer to the system requirements.

Note

You require administrator rights for the installation.

Following installation, every user (including those without administrator rights) can work with SIMOTION CamTool.

System requirements

The system requirements for SIMOTION CamTool match the system requirements of SIMOTION SCOUT.

Installing SIMOTION CamTool

To install SIMOTION CamTool:

1. Insert the CD with the installation software into the CD-ROM drive.
2. Start Windows Explorer and select the CD-ROM drive.
3. Open the **CamTool\Disk1** directory.
4. Double-click **setup.exe**. The installation program starts.
5. Run through the installation program. SIMOTION CamTool is now installed.

3.1.3.2 Deinstalling SIMOTION CamTool

Uninstalling SIMOTION CamTool

To uninstall SIMOTION CamTool:

1. Click **Start > Settings > Control Panel** in the Windows taskbar. The **Control Panel** window appears.
2. Double-click **Software**. The **Software Properties** window appears.
3. Select **SIMOTION SCOUT CamTool** on the **Install/Uninstall** tab.
4. Click **Add/Remove**. The uninstall program starts.
5. Run through the uninstall program. SIMOTION CamTool is now uninstalled.

Note

You require administrator rights for the uninstall process.

3.1.4 Configuring

3.1.4.1 Content

Overview

This chapter describes how you work with SIMOTION CamTool. You learn how

- to customize the display of the working area
- to edit a cam with CamTool
- to save a cam

- to customize the display of the cam
- to download a cam to a SIMOTION device

Note

The following operating instructions primarily describe the operation of SIMOTION CamTool using the functions in the menu bar.

You can also execute the functions from the context menus. In this case, right-click the element that you want to edit.

You can also execute the most import functions using the icons in the SIMOTION CamTool toolbar. Pay attention to the tooltip which is displayed when you place the mouse pointer on an icon in the toolbar.

3.1.4.2 Customizing the Working Area Display

Changing the representation using the toolbar

Introduction

You can use the icons in the function bar to show or hide the scaled cam profile, the V diagram (velocity diagram), the A diagram (acceleration diagram), and the J diagram (jerk diagram). Use the zoom tool to enlarge/reduce the display or move it around with the hand tool.

Using the function bar to change the display

To change how diagrams are displayed using the function bar:

1. A tool tip appears when you position the cursor over an icon in the function bar.
2. Click the icon for the function you wish to use.

Function bar icons

The following functions can be used to change the screen display:

Table 3-1 Icons in the SIMOTION CamTool function bar

Icon	Meaning/Operation
	Use this icon (selection tool) to select the cam segments for editing.
	Use this icon to activate/deactivate the display for the scaled cam profile. If you have not made any scaling specifications, the icon will be grayed out. **Note:** When the scaled cam profile is displayed, you will not be able to edit the original curves in the diagram displays.
	Use this icon to show or hide the V diagram (velocity diagram).
	Use this icon to show or hide the A diagram (acceleration diagram).

Icon	Meaning/Operation
	Use this icon to show or hide the J diagram (jerk diagram).
	Use this icon to activate/deactivate the zoom tool. You can also use the ESC key to deactivate the zoom tool. When activated, the zoom tool enables you to perform a variety of functions, depending on whether you position the cursor over the diagram area or a coordinate axis, and whether you press the SHIFT key while doing this: **Zoom tool over diagram area (zoom effective in all directions simultaneously)** Left-click to double the size of the entire display. Right-click to halve the size of the entire display. **Zoom tool over coordinate axis (zoom effective in direction of coordinate axis)** Left-click to double the size of the display in the direction of the coordinate axis over which the cursor is positioned. Right-click to halve the size of the display in the direction of the coordinate axis over which the cursor is positioned. **Zoom tool with SHIFT key pressed** The cursor assumes the functions of the hand tool for as long as the SHIFT key is pressed and held down. When the hand tool is activated, you can move the diagram area using drag-and-drop. Any diagrams which are currently visible can be moved in this way.
	Use this icon to activate/deactivate the zoom function. You can also use the ESC key to deactivate the zoom function. **Zoom function on the diagram area** When the zoom function is activated, you can press the mouse button down and outline a section of the diagram area you wish to enlarge. **Zoom function on the coordinate axis** When the zoom function is activated, you can press the mouse button down and outline the section of a coordinate axis you wish to enlarge. The enlargement takes effect in the direction you worked in when outlining the section on the coordinate axis.
	Use this icon to reset the previous zoom setting.
	Use this icon to activate/deactivate the hand tool. You can also use the ESC key to deactivate the hand tool. When the hand tool is activated, you can move the diagram area using drag-and-drop. Any diagrams which are currently visible can be moved in this way.
	Use this icon to restore the entire display to the normal view.

Maximizing working area

Maximizing working area

To set the largest possible working area for SIMOTION CamTool:

1. Click **Maximized working area** in the **View** menu. The detail view and project navigator will close.
2. Maximize the window containing the cam diagrams.

or

1. Select **View > Project navigator** to close the project navigator.
2. Select **View > Detail view** to close the detail view.
3. Maximize the window containing the cam diagrams.

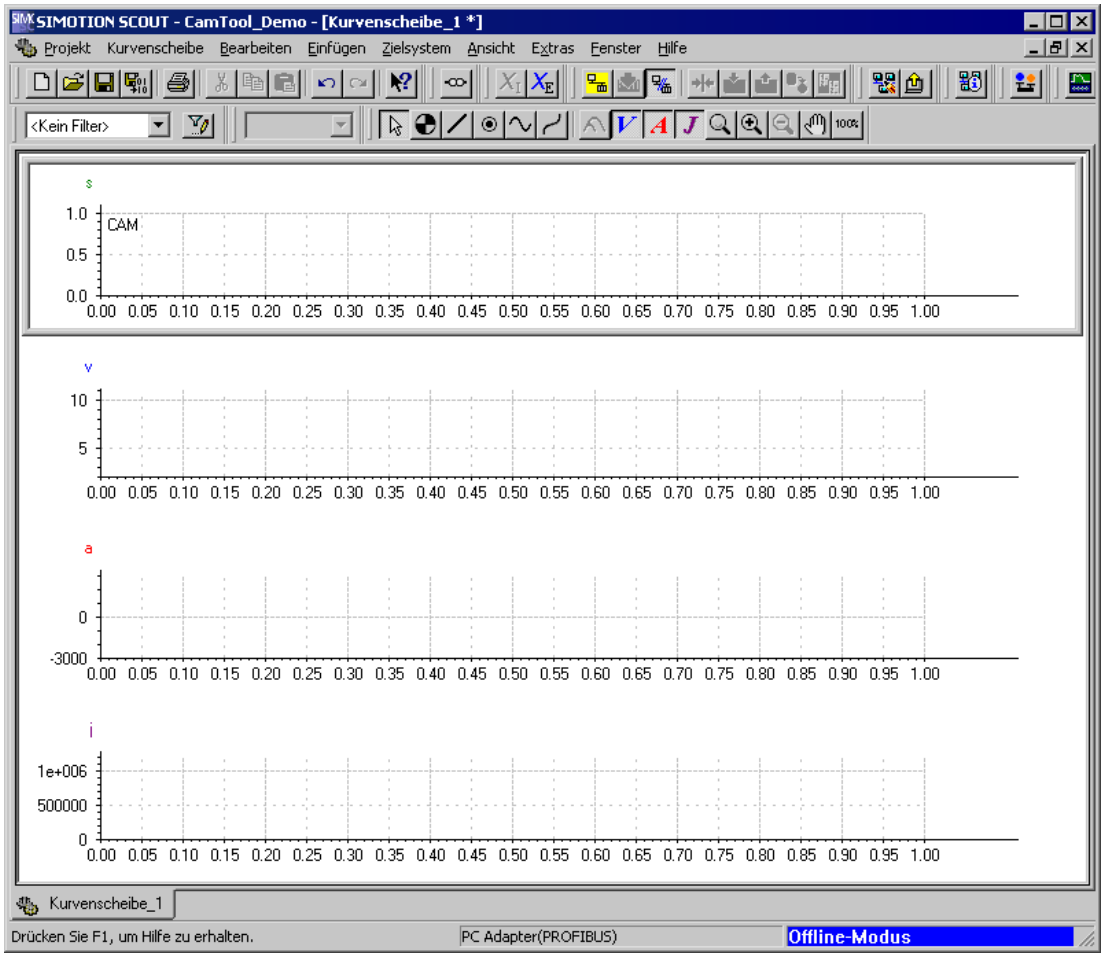


Figure 3-1 Cam diagrams maximized in the SIMOTION SCOUT working area

See also

Changing the representation using the toolbar (Page 59)

3.1.4.3 Editing a Cam with CamTool

Content

Overview

With SIMOTION CamTool, you can edit a cam that is inserted in a SCOUT project.

You can use the following methods to insert and edit a cam. You can

- add a cam to a SCOUT project using SIMOTION CamTool.
- edit with CamTool a cam created with CamEdit.
- import and edit a cam from a text file.
- upload and edit a cam from a SIMOTION device.

Adding a cam to a SCOUT project using CamTool

Requirement

SIMOTION CamTool must be installed as an optional package for SIMOTION SCOUT.

The project in which you wish to insert the cam must be open in SIMOTION SCOUT. There must be at least one SIMOTION device configured in this project.

Inserting a cam in a SCOUT project

To insert a cam in a SCOUT project:

1. In the project navigator, find the SIMOTION device under which you wish to insert a new cam.
2. Open the **Cams** folder and double-click **Insert cam with CamTool.**

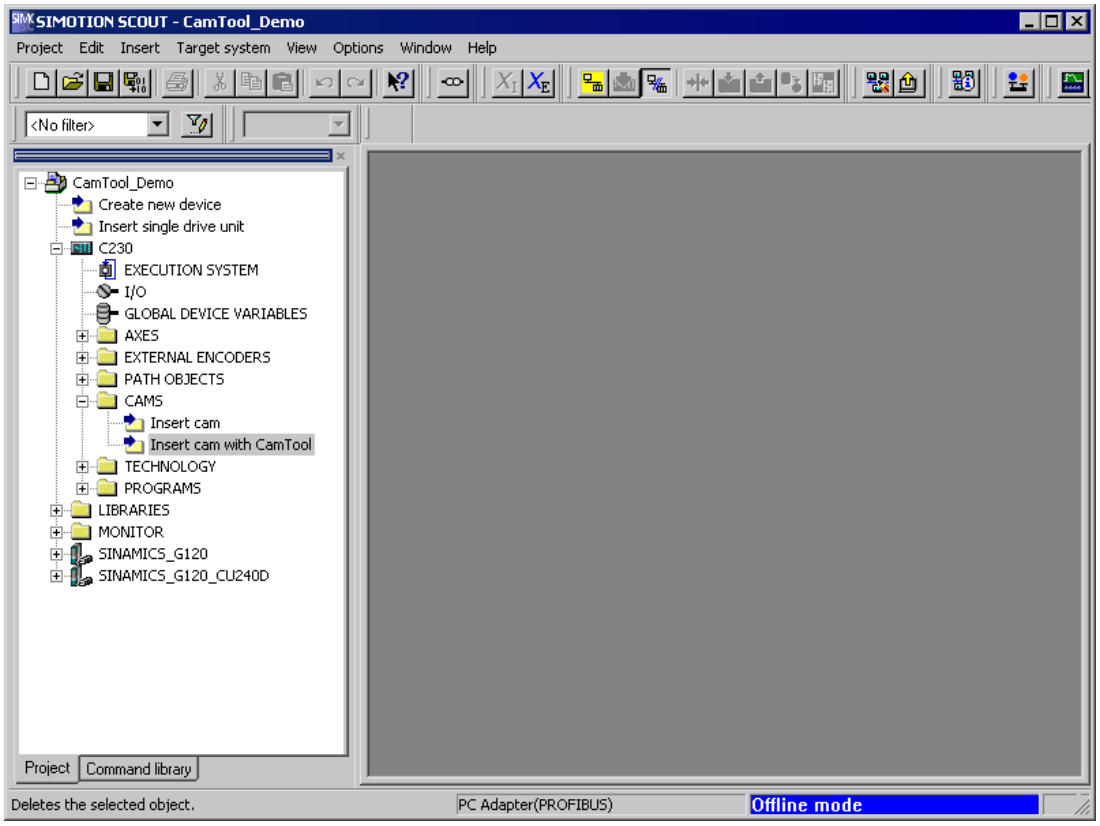


Figure 3-2 Inserting a new cam with CamTool

3. The **Insert cam** window appears. Enter a unique designation for the cam under **Name** (all the cams within the project are listed under **Existing cams**).

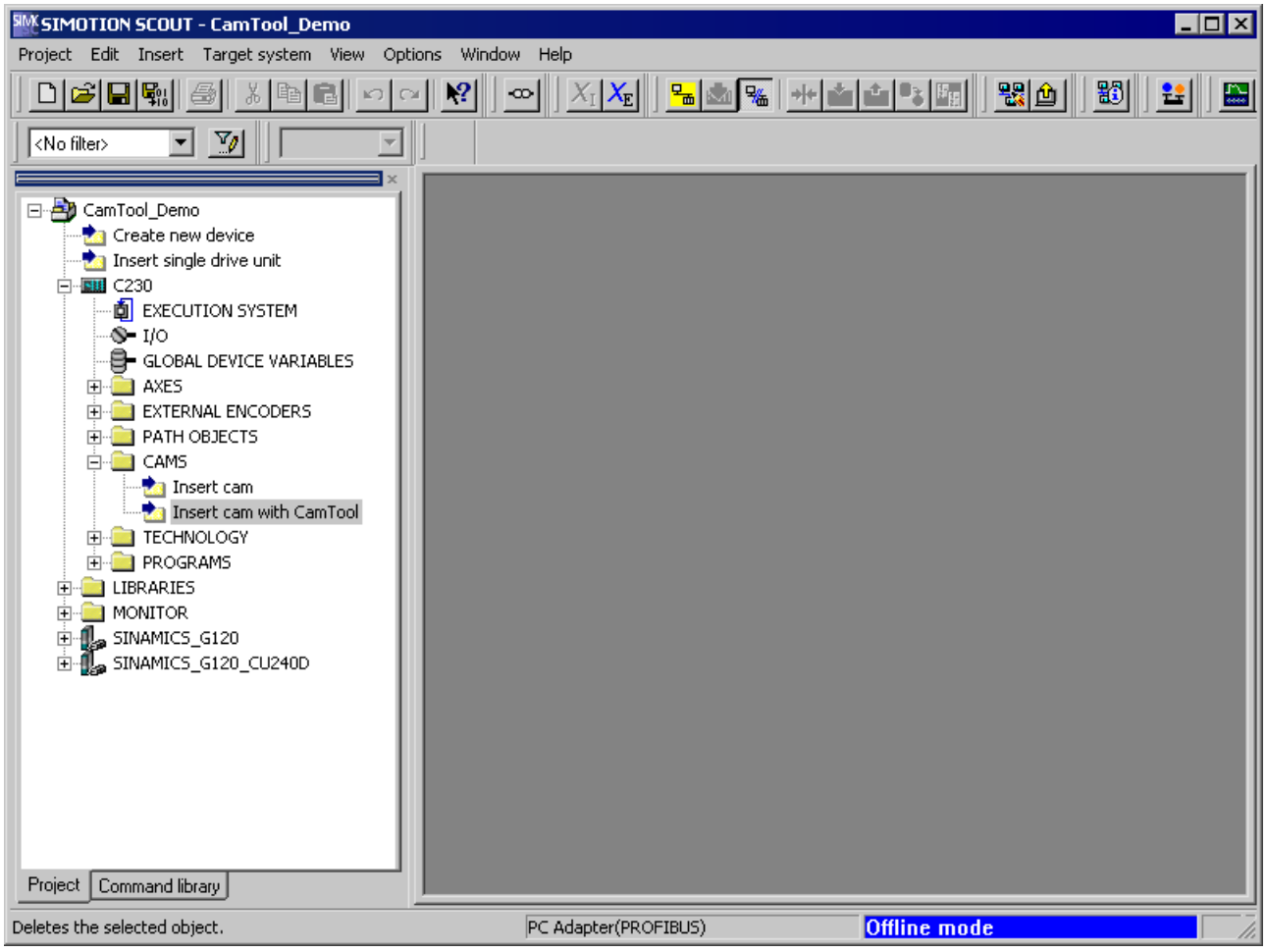


Figure 3-3 Window for inserting cams

4. Click **OK**. A window appears in the SIMOTION SCOUT working area. Depending on the settings in the **Cam > Diagrams** menu item, the cam diagrams will appear in this window.

Note

The cam diagrams show:

- The master axis in the horizontal direction (X-axis) and
- The slave axis in the vertical direction (Y-axis)

See also

Changing the representation using the toolbar (Page 59)

Edit cam created with CamEdit with CamTool

Editing a cam with CamTool

To edit a cam created with CamEdit using CamTool:

1. Close the cam in SIMOTION CamEdit.
2. In the project navigator, find the cam you wish to edit with SIMOTION CamTool.
3. Right-click the cam and select **Convert to CamTool** in the context menu that appears. The cam is opened with SIMOTION CamTool the next time the cam is opened.
4. Double-click the cam. The cam is opened and displayed with SIMOTION CamTool. Segment boundaries between individual cam segments are marked in the S diagram (path diagram).

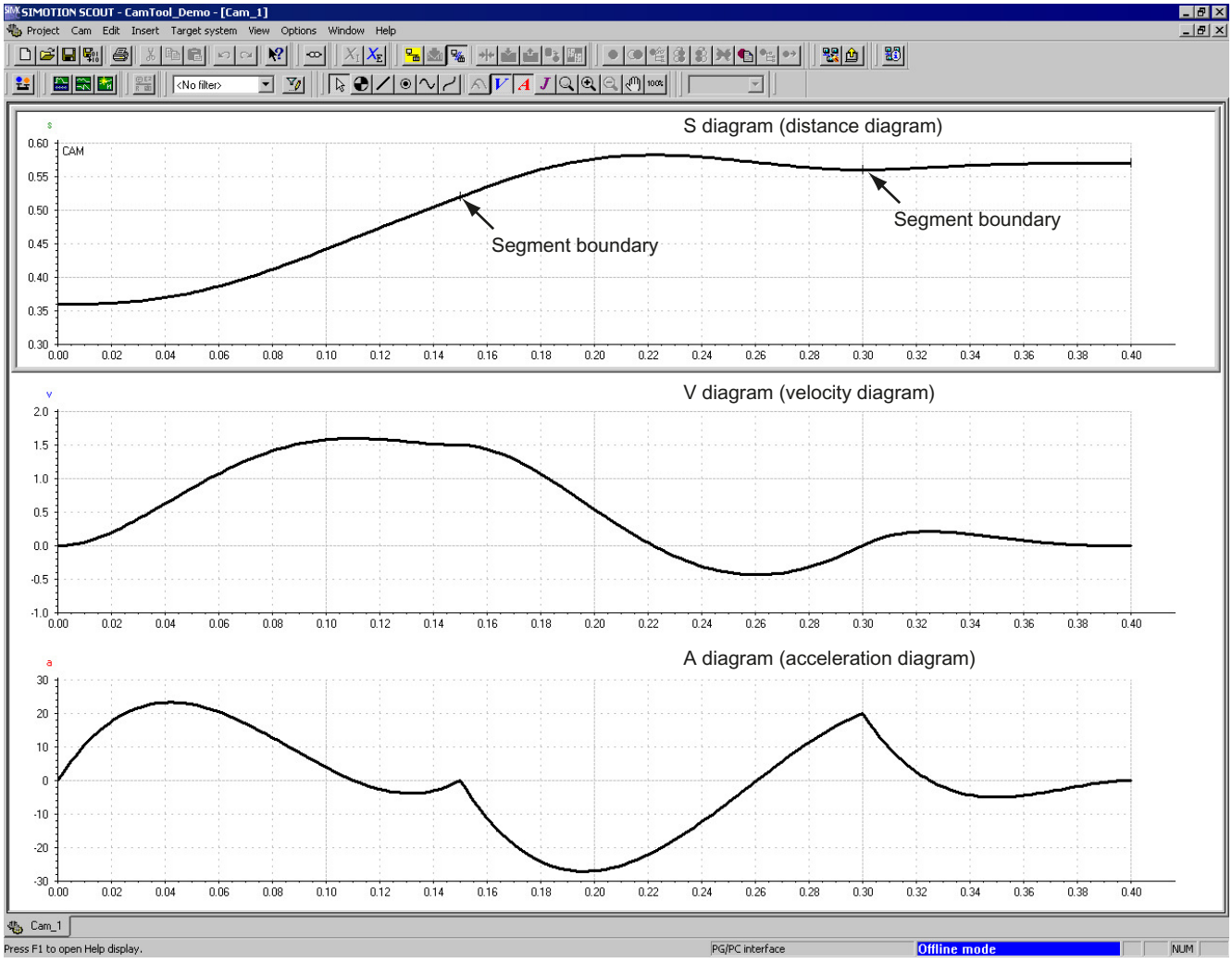


Figure 3-4 Cam created with CamEdit opened with CamTool

5. Select the cam segment you wish to edit. Take the displayed segment boundaries into account.

6. Press the DEL key to delete the cam segment. SIMOTION CamTool replaces the cam segment with an interpolation curve (transition). Fixed points are inserted in the cam profile where necessary to retain the corner points of the original curve.

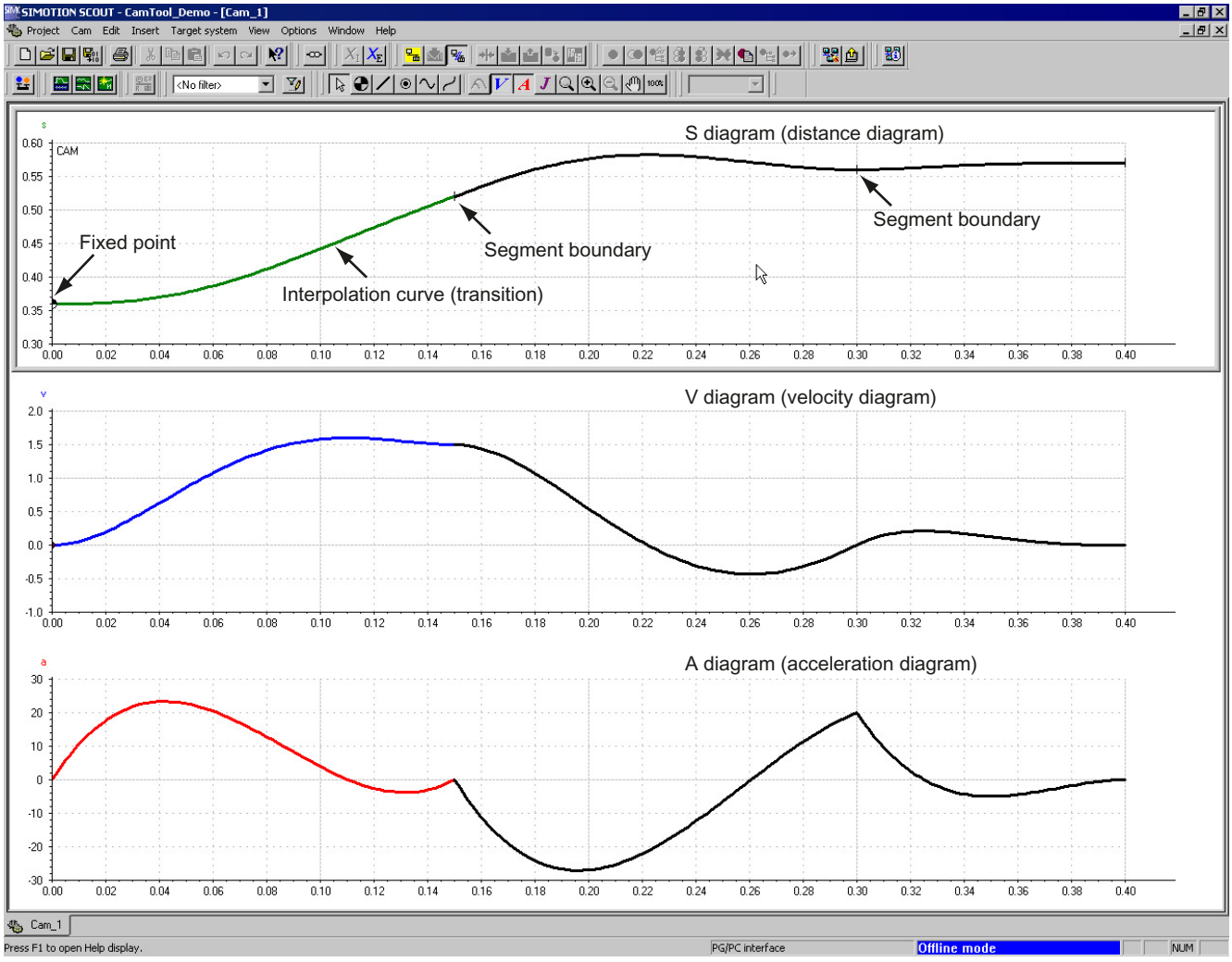


Figure 3-5 Cam segment replaced with fixed point and transition (interpolation polynomial) by SIMOTION CamTool

7. You can edit (e.g. change the position) the cam segment (e.g. fixed point) inserted by SIMOTION CamTool.
8. You can optimize the interpolation curve (transition) inserted by SIMOTION CamTool (e.g. velocity).

See also

Changing the representation using the toolbar (Page 59)

Importing the geometry

Introduction

You can reimport a cam exported from SIMOTION CamTool as a text file into CamTool (e.g. in order to reimport a cam edited in an external program).

Note

The way the cam is represented in the text file must be compatible with the Microsoft Excel CSV format.

Importing a cam

There are two variants for the import of external cams:

- Import with creation of a new cam
- Import with overwriting of an existing cam

Import with creation of a new cam

You can import a new cam via the context menu **Export/import -> Import external cam -> CamTool...**

Import with overwriting of an existing cam

To import a cam from a text file, proceed as follows:

1. Open the cam in which you wish to insert the cam from the text file with SIMOTION CamTool.
   or
   Insert a new cam in which you wish to import the cam from the text file.

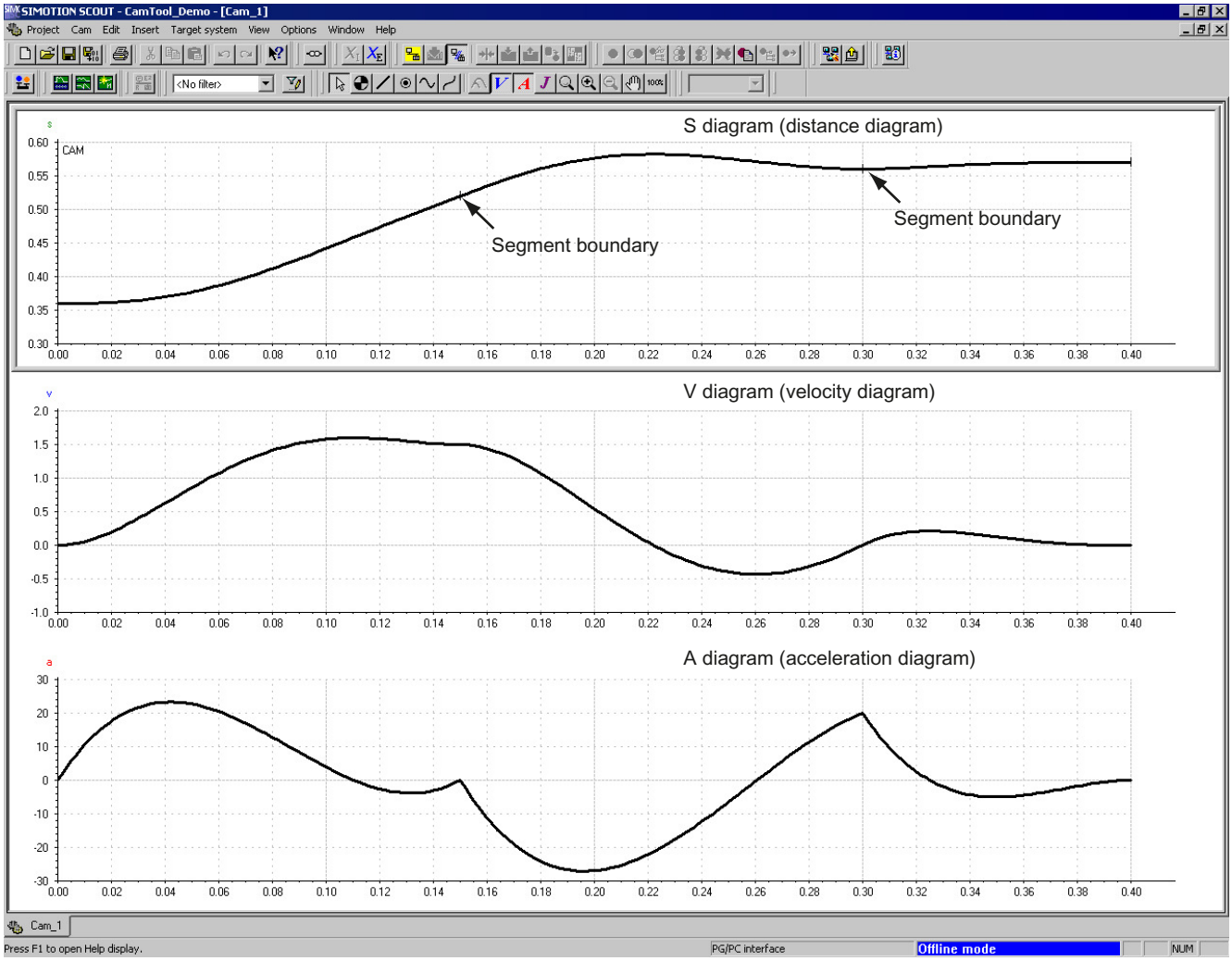


Figure 3-6 Cam imported from a text file

2. Click a diagram to show the **Cam** menu.

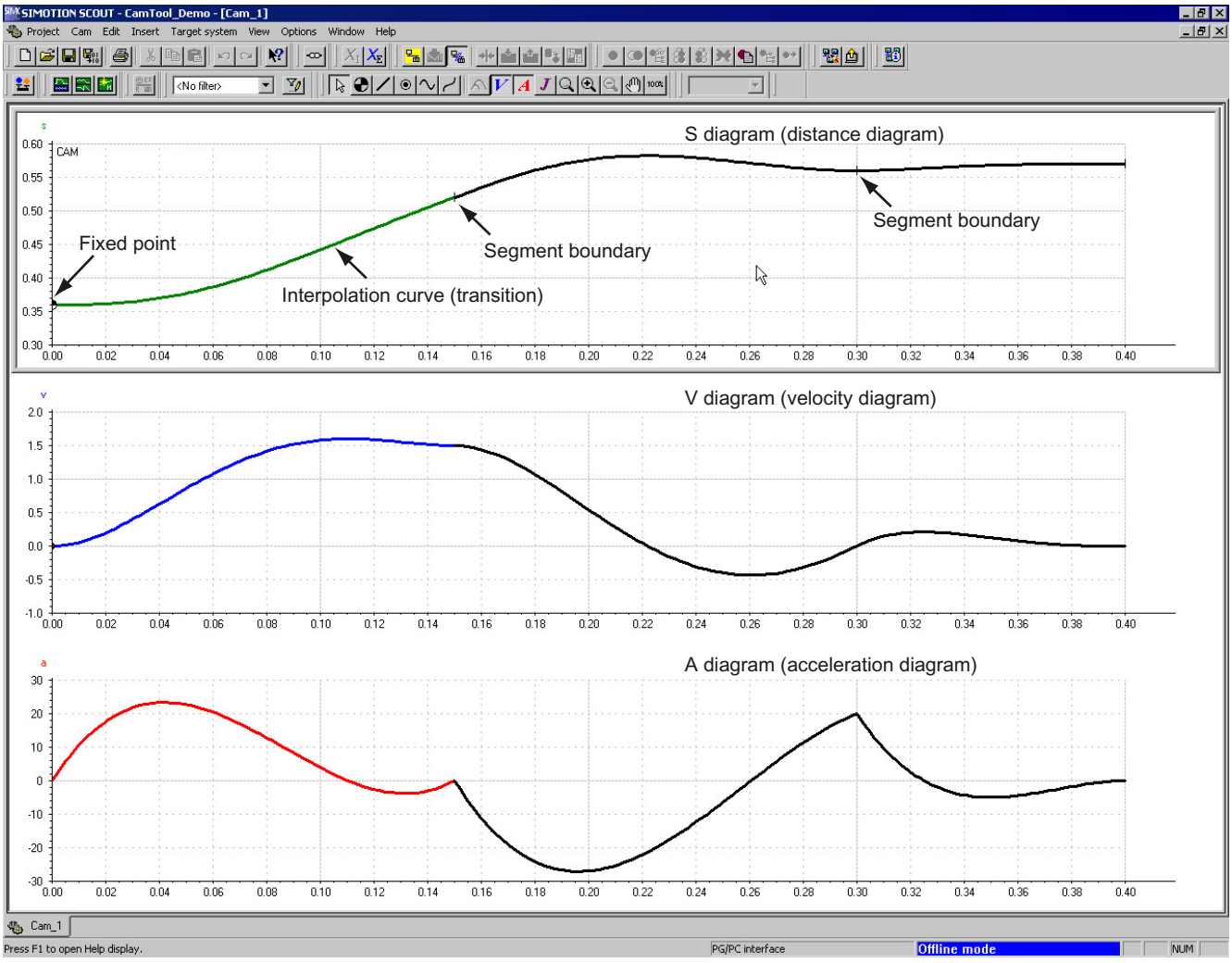


Figure 3-7 Cam segment replaced with fixed point and transition (interpolation polynomial) by SIMOTION CamTool

3. Click the **Cam > Import geometry** menu item. The file selection window appears. At **Search in**, navigate to the text file that contains the cam and select the text file. The designation for the text file is entered at **Name**.
4. Click **OK** in the file selection window. The cam is imported.

Note

If you are importing a cam from a text file and have previously changed the cam displayed, a window appears. This window enables you to accept the changed cam into the project. The cam is then imported from the text file.

Editing an imported cam

To edit a cam imported into SIMOTION CamTool, proceed as follows:

1. The cam imported from a text file is displayed in SIMOTION CamTool. Segment boundaries between individual cam segments are marked in the S diagram (path diagram).
   Select the cam segment you wish to edit. Take account of the displayed segment boundaries.
2. Press the DEL key to delete the cam segment. SIMOTION CamTool replaces the cam segment with an interpolation curve (transition). Fixed points are inserted in the cam profile where necessary to retain the corner points of the original curve.
3. You can edit (e.g. change the position) the cam segment (e.g. fixed point) inserted by SIMOTION CamTool.
4. You can optimize the interpolation curve (transition) inserted by SIMOTION CamTool (e.g. velocity).

See also

Changing the representation using the toolbar (Page 59)

Exporting the geometry

Introduction

You can export a cam created with SIMOTION CamTool as a text file (in order to edit the cam in an external program, for example).

Note

The cam is represented in the text file compatible with the Microsoft Excel CSV format.

The following options are available when exporting:

- As polynomials (polynomial coefficients of the selected cam)
- As raw data (segment properties of the selected cam)

Exporting the cam as polynomials

To export a cam as polynomials, proceed as follows:

1. Open the cam with SIMOTION CamTool.
2. Click a diagram to show the **Cam** menu.
3. Click the **Cam > Export geometry > As polynomials...** menu item. The file selection window appears.
4. At **Save in**, select the target for the text file and enter a designation for the text file at **Name**.
5. Click **OK** in the file selection window. The cam is saved as a text file with the entered designation to the selected target.

Exporting the cam as raw data

To export a cam as raw data, proceed as follows:

1. Open the cam with SIMOTION CamTool.
2. Click a diagram to show the **Cam** menu.
3. Click the **Cam > Export geometry > As raw data...** menu item. The file selection window appears.
4. At **Save in**, select the target for the text file and enter a designation for the text file at **Name**.
5. Click **OK** in the file selection window. The cam is saved as a text file with the entered designation to the selected target.

Note

No export with separated transition segments

Note that the cam cannot be exported with separated transition segments.

Separated transition segments can result, for example, through the deletion of individual segments.

See also

Changing the representation using the toolbar (Page 59)

Upload cam from SIMOTION device

Requirement

The cam must be opened with SIMOTION CamTool.

Note

If you are creating a new cam which has never been downloaded to the SIMOTION device, you first need to download the entire configuration **including** the new cam to the SIMOTION device.

Only then will you be able to upload the cam currently opened with SIMOTION CamTool from the SIMOTION device. Uploading is only possible ONLINE.

Uploading a cam

To upload a cam from a SIMOTION device: :

1. Click the **Project > Connect to target system** menu item. The system switches to ONLINE mode.

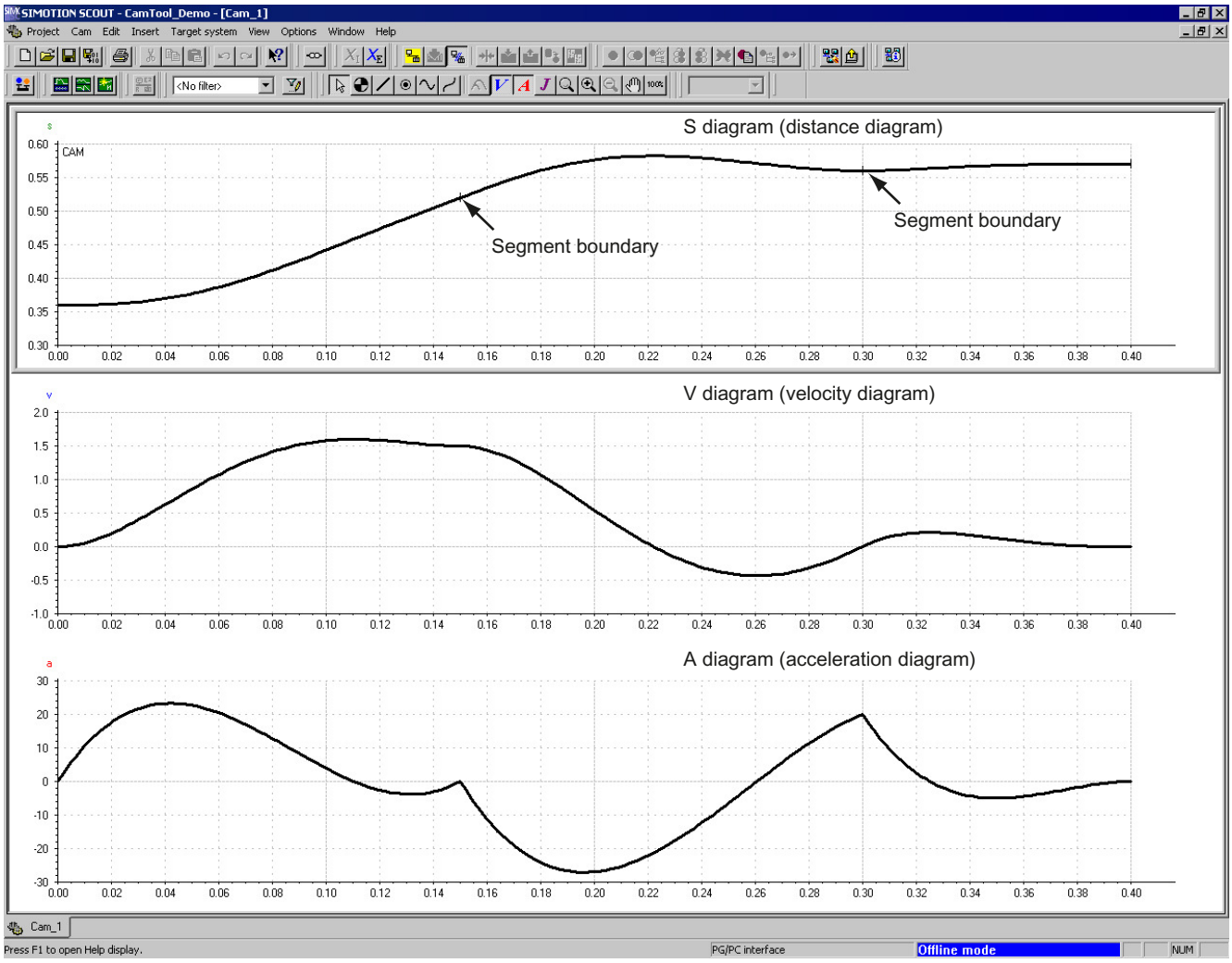


Figure 3-8 Cam uploaded from a SIMOTION device

2. Click a diagram to show the **Cam** menu.

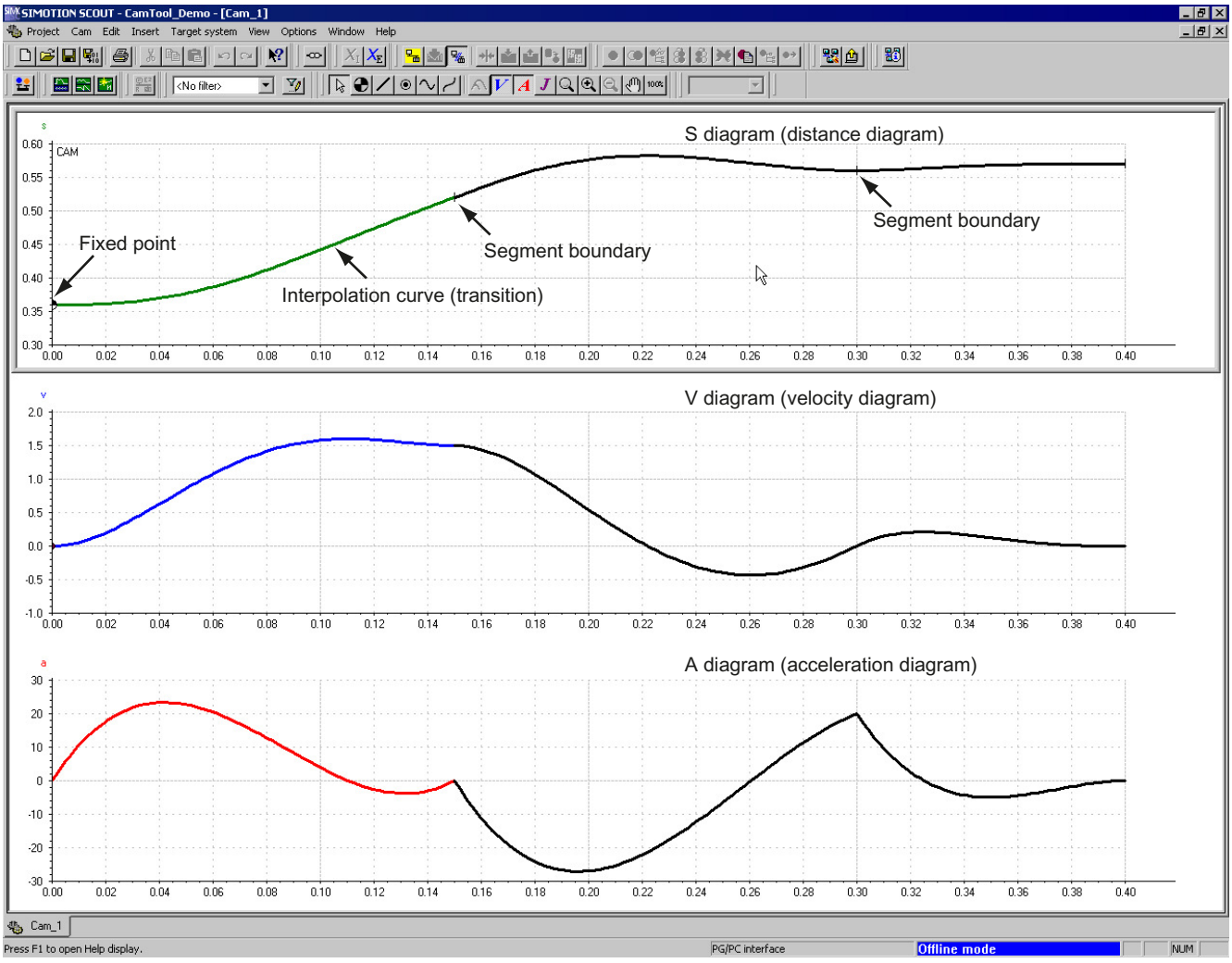


Figure 3-9 Cam segment replaced with fixed point and transition (interpolation polynomial) by SIMOTION CamTool

3. Click the **Cam > Upload cam** menu item. The cam is uploaded from the SIMOTION device.
4. You can switch back to OFFLINE mode once uploading is complete by clicking the **Project > Disconnect from target system** menu item.

Note

If you are uploading a cam from the SIMOTION device and have already changed the cam displayed, a window will appear. This window enables you to accept the changed cam into the project. The cam is then uploaded from the SIMOTION device.

Editing an uploaded cam

To edit a cam that has been uploaded from a SIMOTION device:

1. The uploaded cam is displayed in SIMOTION CamTool. Segment boundaries between individual cam segments are marked in the S diagram (path diagram).
   Select the cam segment you wish to edit. Please take account of the segment boundaries displayed.
2. Press the DEL key to delete the cam segment. SIMOTION CamTool replaces the cam segment with an interpolation curve (transition). Fixed points are inserted in the cam profile where necessary to retain the vertices from the original curve.
3. You can edit (e.g. change the position of) the cam segment (e.g. fixed point) inserted by SIMOTION CamTool.
4. You can optimize the interpolation curve (transition) inserted by SIMOTION CamTool (e.g. its velocity).

See also

Changing the representation using the toolbar (Page 59)

3.1.4.4 Save cam

Requirement

The cam must be opened with SIMOTION CamTool.

Note

If you are exiting SIMOTION CamTool and have already changed the cam, a window will appear. This window enables you to accept the changed cam into the project. SIMOTION CamTool then closes.

If you are uploading a cam from a SIMOTION device and have already changed the cam displayed, a window will appear. This window enables you to accept the changed cam into the project. The cam is then uploaded from the SIMOTION device.

If you are importing a cam from a text file and have already changed the cam displayed, a window will appear. This window enables you to accept the changed cam into the project. The cam is then imported from the text file.

Saving a cam

To save a cam:

1. Click the **Project > Save** menu item. The cam is accepted into the SCOUT project.

See also

Changing the representation using the toolbar (Page 59)

3.1.4.5 Customize the display of the cam

Display the cam

Diagram display

The S diagram (distance diagram) of the cam is always displayed in the SIMOTION SCOUT working area. You can show or hide the V diagram (velocity diagram), the A diagram (acceleration diagram) and the J diagram (jerk diagram) via icons in the toolbar.

Representation parameters

You can customize the display parameters (e.g. representation range) for the master axis, the slave axis and the individual diagrams. This also includes the fonts and lines used for the display.

Show/hide diagram

Requirement

The cam must be opened with SIMOTION CamTool.

Showing/Hiding a diagram

To show/hide a diagram:

1. Click a diagram to show the **Cam** menu.
2. In the **Cam > Diagrams** menu item, click the diagram you wish to show/hide. The diagram is shown/hidden.

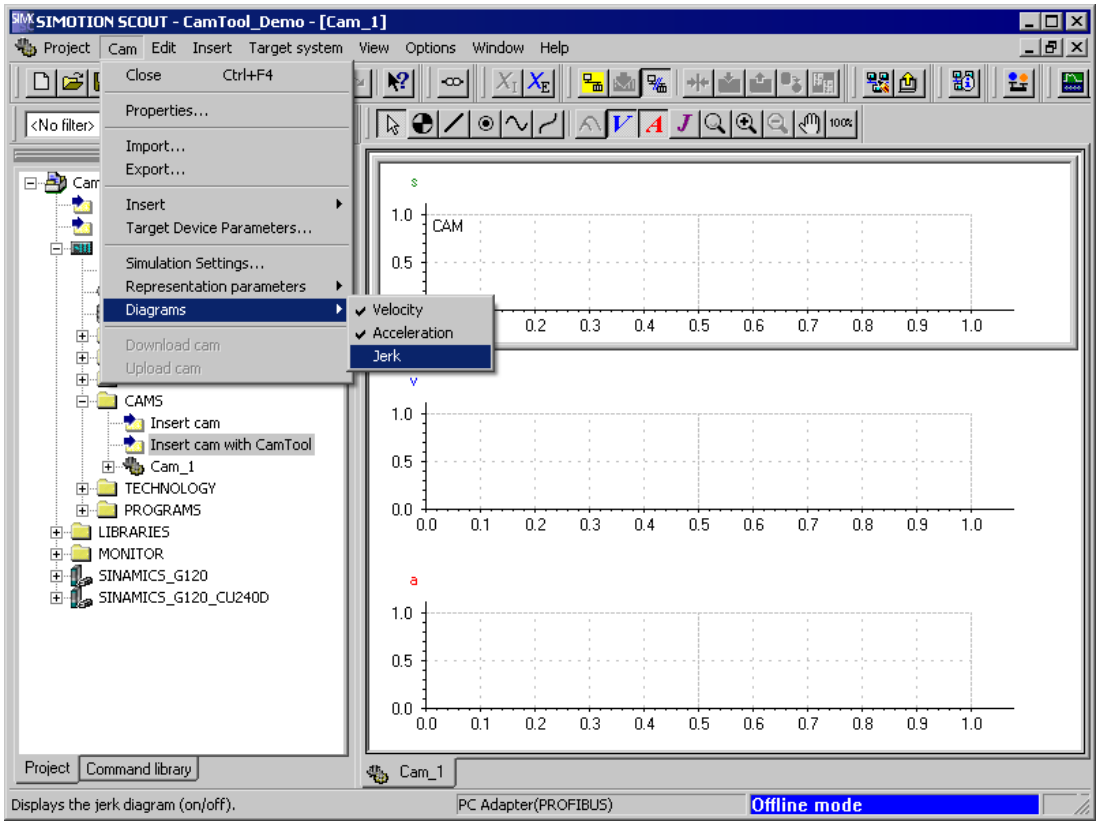


Figure 3-10 Showing/hiding diagrams

See also

Changing the representation using the toolbar (Page 59)

Change axis representation parameters

Requirement

The cam must be opened with SIMOTION CamTool.

Changing representation parameters for axes

To change the representation parameters for the master axis or slave axis:

1. Click a diagram to show the **Cam** menu.
2. In the **Cam > Representation parameters** menu item, click the axis whose representation parameters you wish to change.
3. The **Master properties** window (for the master axis) or the **S diagram (slave) properties** window (for the slave axis) appears.

Change the representation parameters for the master axis in **Master properties.**

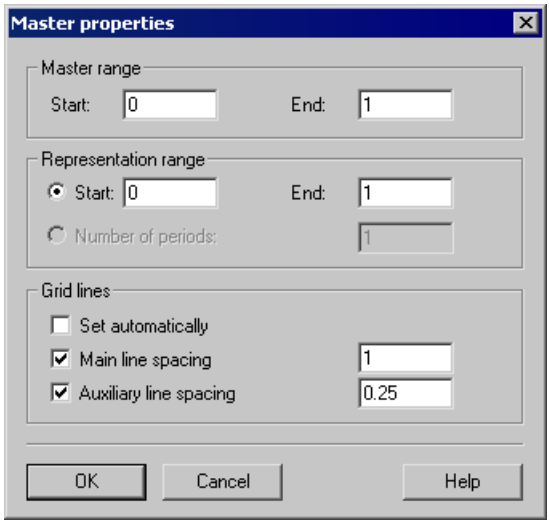


Figure 3-11 Master properties window

Change the representation parameters for the slave axis in **S diagram (slave) properties.**

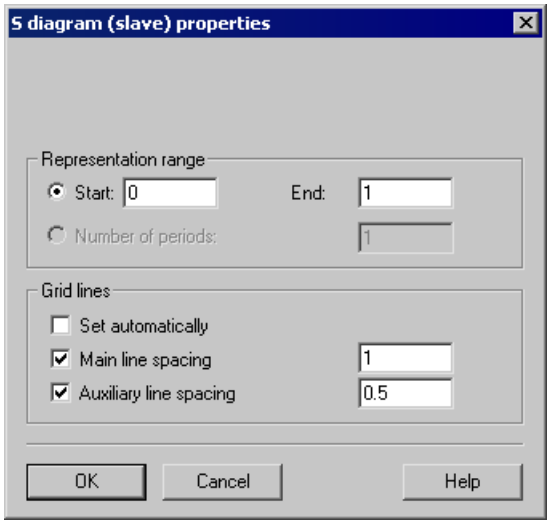


Figure 3-12 Slave properties window

Table 3-2 Parameters in the window for the master or S diagram (slave) properties

Field/Button		Meaning/Operation
Master range (Master properties)		Specify the **master range** (definition range) for the curve by setting its start and end points. Cam segments must be contained within the master range (see also **Master range** under **Target device properties** on the **Coordinates** tab).
	Start	Here, you enter the **start** point of the curve or the start of the curve's master range.
	End	Here, you enter the **end** point of the curve or the end of the curve's master range.
Representation range		
	Start (Number of periods deactivated)	Here, you specify the **start** of the representation range. If the **Number of periods** option is activated, the system will specify the start.
	End (Number of periods deactivated)	Here, you enter the **end** of the representation range. If the **Number of periods** option is activated, the system will specify the end.
	Number of periods (Cyclic absolute, Cyclic relative)	If you have specified the **Cyclic absolute** or **Cyclic relative execution type** under **Target device parameters** on the **Coordinates** tab, you will be able to select the **Number of periods** option. Enter the number of periods you want to display here.
Grid lines		
	Set automatically	If you activate **Set automatically**, the optimum distance between the grid lines will be set automatically. If you deactivate **Set automatically**, you will be able to specify the distance between the grid lines under **Main line spacing** or **Auxiliary line spacing**.
	Main line spacing	This is where you activate **Main line spacing** for the grid lines. If you deactivate the **Set automatically** option, you will be able to set the distance between the grid lines for main line spacing. **Note**: The distance between the grid lines defined in main line spacing must be a multiple of the distance between the specifications for grid lines with auxiliary line spacing.
	Auxiliary line spacing (Main line spacing activated)	If the **Main line spacing** option is activated, you will also be able to activate the grid line display for **Auxiliary line spacing**. If you deactivate the **Set automatically** option, you will be able to set the distance between the grid lines for auxiliary line spacing. **Note**: The distance between the grid lines defined in main line spacing must be a multiple of the distance between the specifications for grid lines with auxiliary line spacing.

See also

Changing the representation using the toolbar (Page 59)

Change diagram representation parameters

Requirement

The cam must be opened with SIMOTION CamTool.

Changing representation parameters for a diagram

To change the representation parameters for the V diagram (velocity diagram), A diagram (acceleration diagram), or the J diagram (jerk diagram):

1. Click a diagram to show the **Cam** menu.
2. In the **Cam > Representation parameters** menu item, click the diagram whose representation parameters you wish to change.
3. The **V diagram properties** window (for the velocity diagram), the **A diagram properties** window (for the acceleration diagram), or the **J diagram properties** window (for the jerk diagram) appears.

For the V diagram (velocity diagram), change the representation parameters in **V diagram properties.**

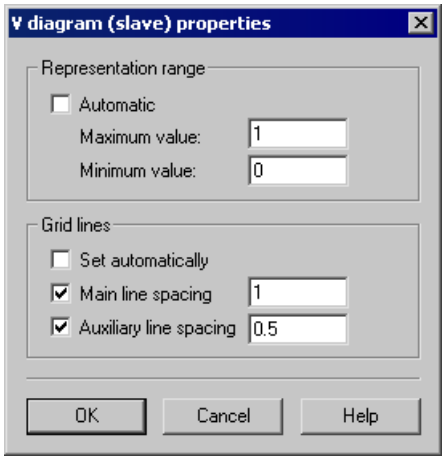


Figure 3-13 V diagram properties window

For the A diagram (acceleration diagram), change the representation parameters in **A diagram properties.**

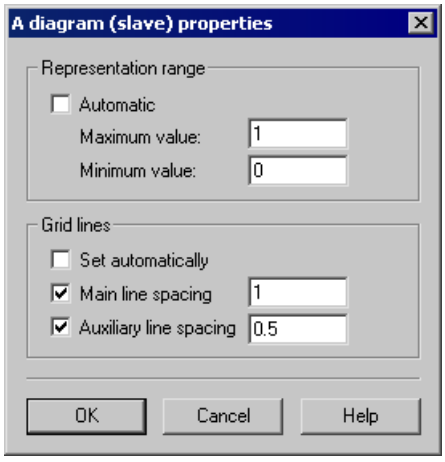


Figure 3-14 A diagram properties window

For the J diagram (jerk diagram), change the representation parameters in **J diagram properties.**

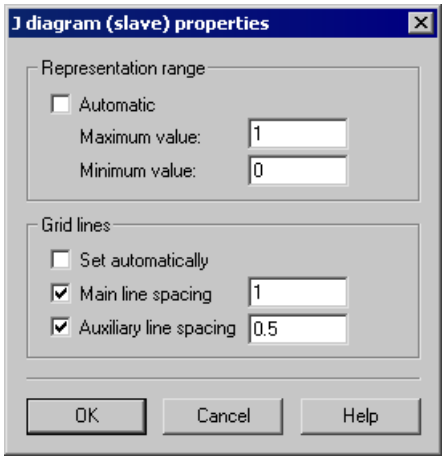


Figure 3-15 J diagram properties window

Table 3-3 Parameters in the V/A/J diagram properties window

Field/Button		Meaning/Operation
Representation range		
	Automatic	If you activate **Automatic**, the representation range will be automatically adapted in the Y direction to fit the value range for the curve shown. If you deactivate **Automatic**, you will be able to specify the representation range in the Y-direction using **Maximum value** and **Minimum value**.
	Maximum value (Automatic deactivated)	Here, you enter the **maximum value** of the representation range. If the **Automatic** option is activated, the system will specify the maximum value.

Field/Button		Meaning/Operation
	Minimum value (Automatic deactivated)	Here, you enter the **minimum value** of the representation range. If the **Automatic** option is activated, the system will specify the minimum value.
Grid lines		
	Set automatically	If you activate **Set automatically**, the optimum distance between the grid lines will be set automatically. If you deactivate Set automatically, you will be able to specify the distance between the grid lines under **Main line spacing** or **Auxiliary line spacing**.
	Main line spacing	This is where you activate **Main line spacing** for the grid lines. If you deactivate the **Set automatically** option, you will be able to set the distance between the grid lines for main line spacing. **Note:** The distance between the grid lines defined in main line spacing must be a multiple of the distance between the specifications for grid lines with auxiliary line spacing.
	Auxiliary line spacing (Main line spacing activated)	If the **Main line spacing** option is activated, you will also be able to activate the grid line display for **Auxiliary line spacing**. If you deactivate the **Set automatically** option, you will be able to set the distance between the grid lines for auxiliary line spacing. **Note:** The distance between the grid lines defined in main line spacing must be a multiple of the distance between the specifications for grid lines with auxiliary line spacing.

See also

Changing the representation using the toolbar (Page 59)

Change lines and fonts representation parameters

Default parameters for lines and fonts are used when displaying diagrams. You can change these default settings.

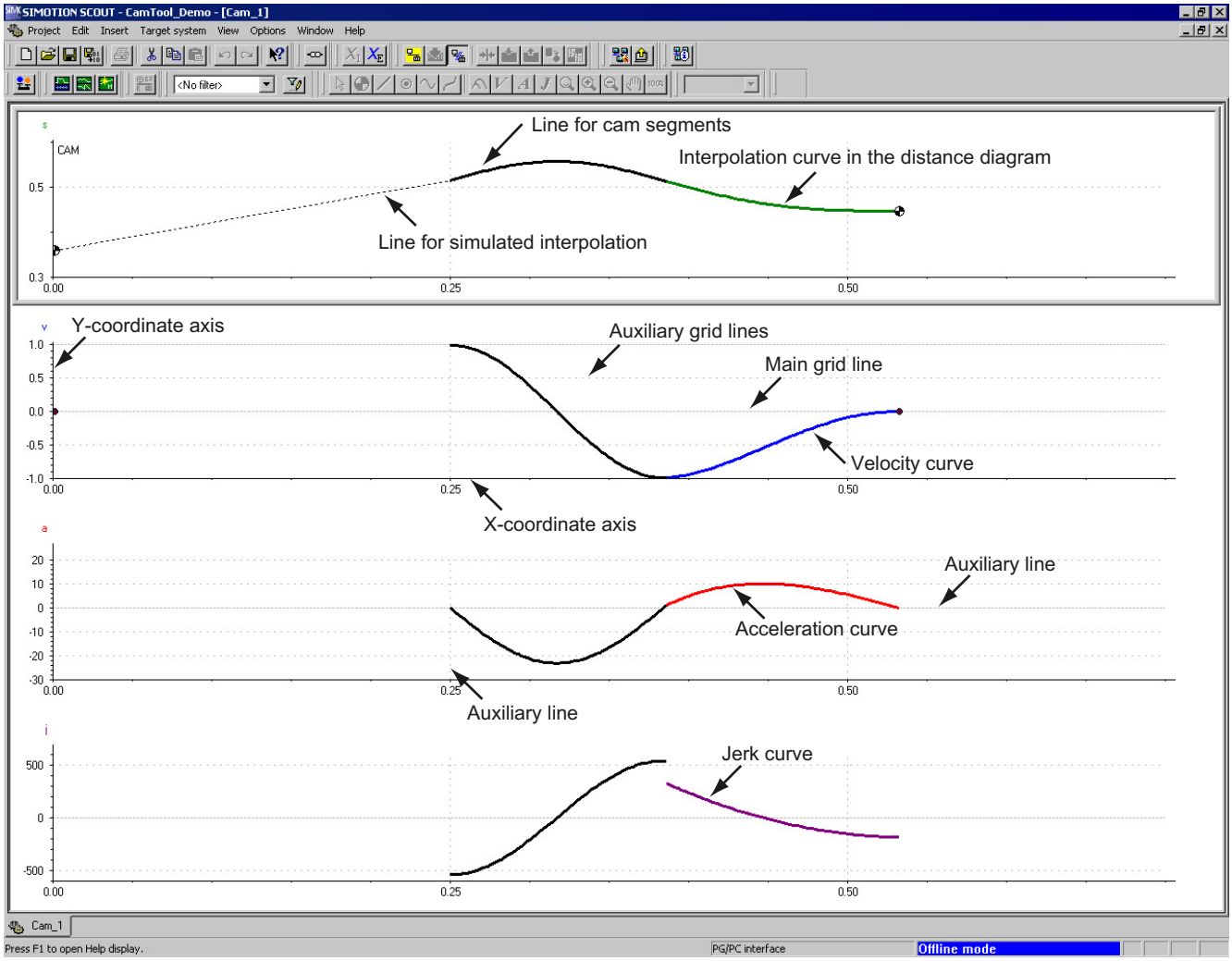


Figure 3-16 Representation parameters in diagrams (lines for scaling and physical limit value are not shown)

Table 3-4 Representation parameters in diagrams

Representation parameters	Note
Main grid lines	You can change the type and color of the line.
Auxiliary grid lines	You can change the type and color of the line.
Auxiliary line	You can change the type, color, and width of the line.
Line at physical limit	You can change the type, color, and width of the line.
X-axis	You can change the color and width of the line.

Representation parameters	Note
Y-axis	You can change the color and width of the line.
Line for simulated interpolation	You can change the color and width of the line. **Line for simulated interpolation** is used to display a transition interpolated by the target device.
Interpolation curve in the path diagram	You can change the color and width of the line.
Velocity curve	You can change the color and width of the line.
Acceleration curve	You can change the color and width of the line.
Jerk curve	You can change the color and width of the line.
Line for cam segments	You can change the color and width of the line.
Line for scaling	You can change the color and width of the line. The **line for scaling** is used to display a cam profile which has been scaled and moved.

Requirement

The cam must be opened with SIMOTION CamTool.

Changing representation parameters for lines and fonts

To change representation parameters for lines and fonts: :

1. Click a diagram to show the **Cam** menu.
2. Click **Lines and fonts** in the **Cam > Representation parameters** menu item.
3. The **Lines and fonts** window appears.

You can change the representation parameters for the lines on the **Line** tab.

Table 3-5 Parameters on the Line tab

Field/Button		Meaning/Operation
Type of line		Here, you select the **type of line** you wish to specify. The associated parameters and the type of line are displayed under **Settings**.
Settings		**Note:** Grayed-out parameters cannot be changed.
	Type	Here, you select the **type** of line. You can use **Preview** to display the type of line.
	Color	Here, you select the **color** of the line. You can use **Preview** to display the line.
	Width	Here, you select the **width** of the line. You can use **Preview** to display the type of line.
	Preview	Select **Preview** to display the **type**, **color**, and **width** specified for the line.

You can change the representation parameters for fonts on the **Fonts** tab.

Table 3-6 Parameters on the Fonts tab

Field/Button		Meaning/Operation
Settings		
	Font	Here, you select the **font**. You can use **Preview** to display the font.
	Font scheme	Here, you select the **font scheme**. You can use **Preview** to display the font.
	Font size	Here, you select the **font size**. You can use **Preview** to display the font.
	Preview	Select **Preview** to display the font specifications made in **font**, **font scheme**, and **font size**.

See also

Changing the representation using the toolbar (Page 59)

Displaying auxiliary lines in the diagram

You can display horizontal and vertical auxiliary lines in individual diagrams. Auxiliary lines can be used, for example, to determine the position of individual points on the cam profile.

Requirement

The cam must be opened with SIMOTION CamTool.

Displaying a horizontal auxiliary line in a diagram

To display a horizontal auxiliary line (in the direction of the X-axis) in a diagram:

1. In the diagram where you want to display a horizontal auxiliary line, click the X-axis and hold the mouse button down. The current position of the auxiliary line is displayed in the positioning window.
2. Drag the auxiliary line to the position where you wish to display it and release the mouse button. The auxiliary line and the positioning window showing its current position appear in the diagram.

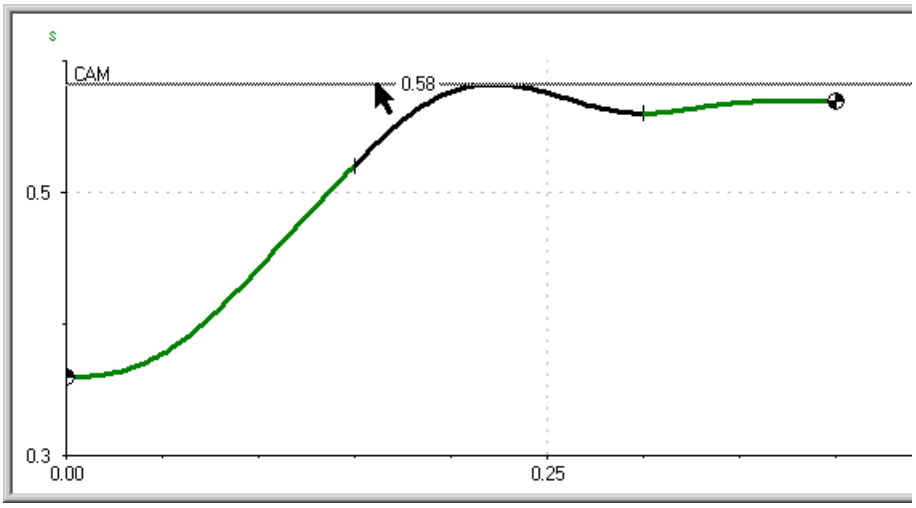


Figure 3-17 Displaying a horizontal auxiliary line

Displaying a vertical auxiliary line in all diagrams on show

To display a vertical auxiliary line (in the direction of the Y-axis) in all diagrams on show:

1. Click the Y-axis in a diagram and hold the mouse button down. The current position of the auxiliary line is displayed in the positioning window.
2. Drag the auxiliary line to the position where you wish to display it and release the mouse button. The auxiliary line and the positioning window showing its current position appear in all the diagrams that are on show.

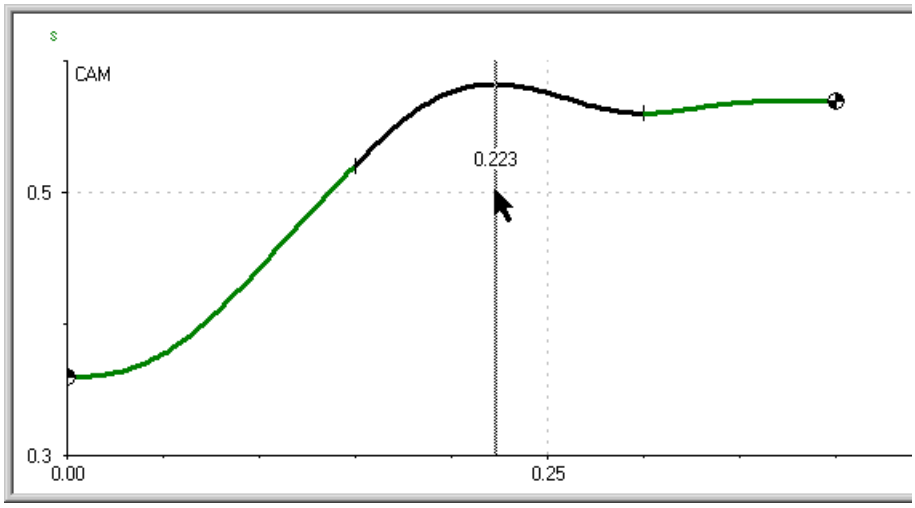


Figure 3-18 Displaying a vertical auxiliary line

Moving a positioning window along an auxiliary line

To move a positioning window showing the current position of the auxiliary line along the horizontal or vertical auxiliary line:

1. Place the cursor on the positioning window. The cursor indicates the direction in which you can move the window. Drag and drop the window showing the current position to the new position.

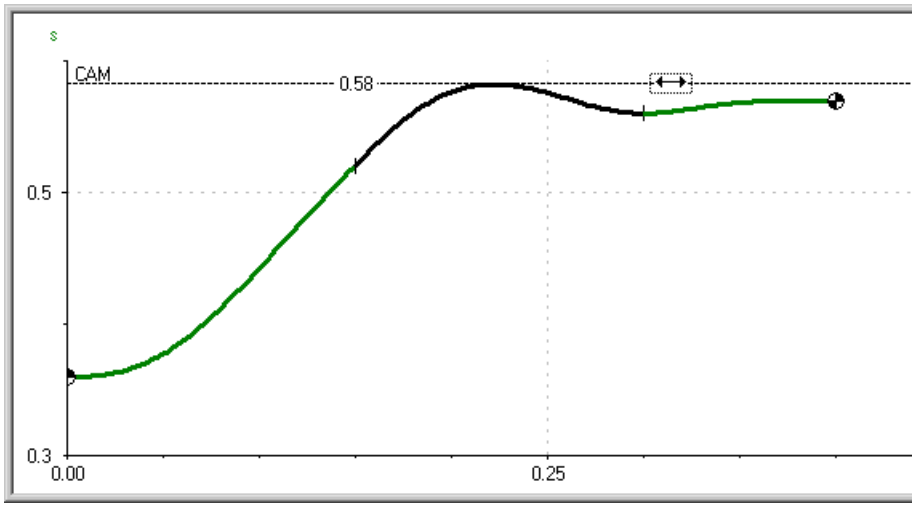


Figure 3-19 Moving a positioning window horizontally

Deleting an auxiliary line in a diagram

To delete a horizontal or vertical auxiliary line in a diagram:

1. Select the auxiliary line you wish to delete.
2. Press the DEL key. The auxiliary line is deleted.

See also

Changing the representation using the toolbar (Page 59)

3.1.4.6 Download cam to SIMOTION device

Requirements

The cam is opened with SIMOTION CamTool.

Note

If you create a cam and the cam has never been previously downloaded to the SIMOTION device, you must first download the entire configuration **with** the new cam to the SIMOTION device.

Only then can you download the cam independent of the other technology objects in the SIMOTION device. The download is only possible in ONLINE status.

Download cam to SIMOTION device

To download a cam to a SIMOTION device:

1. Click the menu Project > Connect to target system. The system changes to ONLINE status.
2. Click a diagram to show the Cam menu.
3. Click the menu Cam > Download cam. The cam is downloaded to the SIMOTION device.
4. After the download, you can change back to OFFLINE status. Click the menu Project > Disconnect from target system.

See also

Changing the representation using the toolbar (Page 59)

3.1.5 Functions

3.1.5.1 Content

Overview

In this chapter you learn how you can use SIMOTION CamTool to create and optimize a cam, and to make the simulation settings. In addition, you learn how you can use CamTool to edit a cam created with CamEdit and how you can export a cam as a text file.

Note

The following operating instructions primarily describe the operation of SIMOTION CamTool using the functions in the menu bar.

You can also execute the functions from the context menus. In this case, right-click the element that you want to edit.

You can also execute the most import functions using the icons in the SIMOTION CamTool toolbar. Pay attention to the Tooltip which is displayed when you place the mouse pointer on an icon in the toolbar.

3.1.5.2 Structure of a cam

Introduction

Create the cam in the S diagram (path diagram). The cam profile reflects the path-related interdependence between the master axis (X-axis in the diagram) and the slave axis (Y-axis in the diagram).

Cam structure

The cam consists of individual cam segments.

With SIMOTION CamTool, you can use fixed points, straight lines, sine curves, arc sine curves, and interpolation points as cam segments.

SIMOTION CamTool calculates interpolation curves between individual cam segments and displays the V diagram (velocity diagram), A diagram (acceleration diagram), and J diagram (jerk diagram).

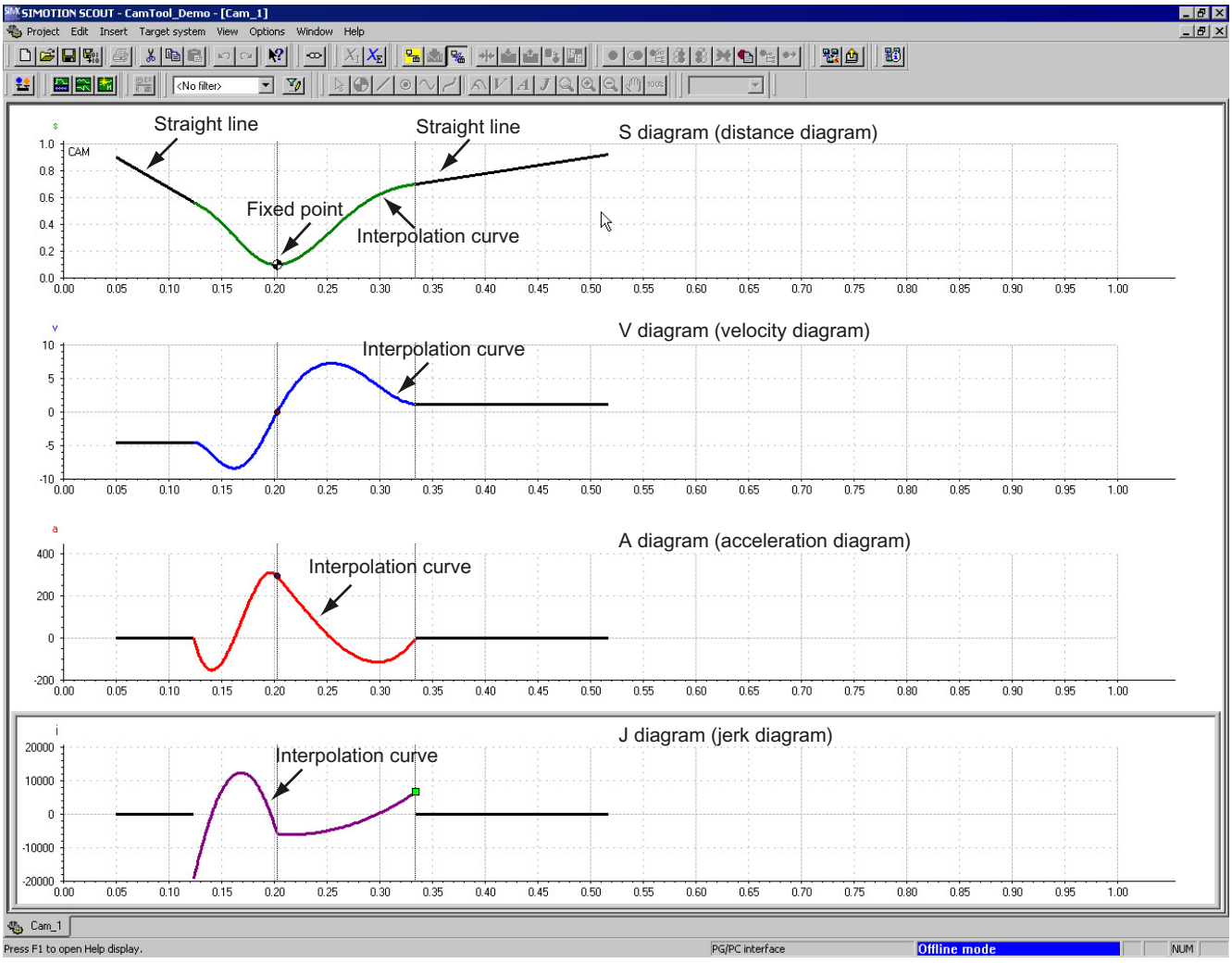


Figure 3-20 Example of a cam with interpolation curves

3.1.5.3 Fixed point

Fixed Point Definition

Definition

The cam consists of individual cam segments. With SIMOTION CamTool, you can use fixed points, lines, sine curves, arc sine curves, and interpolation points as cam segments.

A fixed point is a predefined slave axis position for a given master axis position. You can specify the velocity and acceleration at the position of the fixed point.

Note

Use a fixed point to specify a single, fixed position. CamTool calculates optimum transitions between adjacent cam segments.

Interpolation points (Page 111) must be used for any profile you wish to define. CamTool links the individual interpolation points with a cubic spline in order to create the profile specified by the interpolation points as accurately as possible.

Insert fixed point

Requirement

The cam must be opened with SIMOTION CamTool.

Inserting a fixed point

To insert a fixed point:

1. Click the S diagram (path diagram) to activate it.
2. Click Cam > Insert at fixed point. The cursor changes appearance.
3. Place the modified cursor at the position in the S diagram (path diagram) where you wish to insert the fixed point and click.
   The fixed point appears in the diagram. The position of the fixed point is shown in the tool tip.

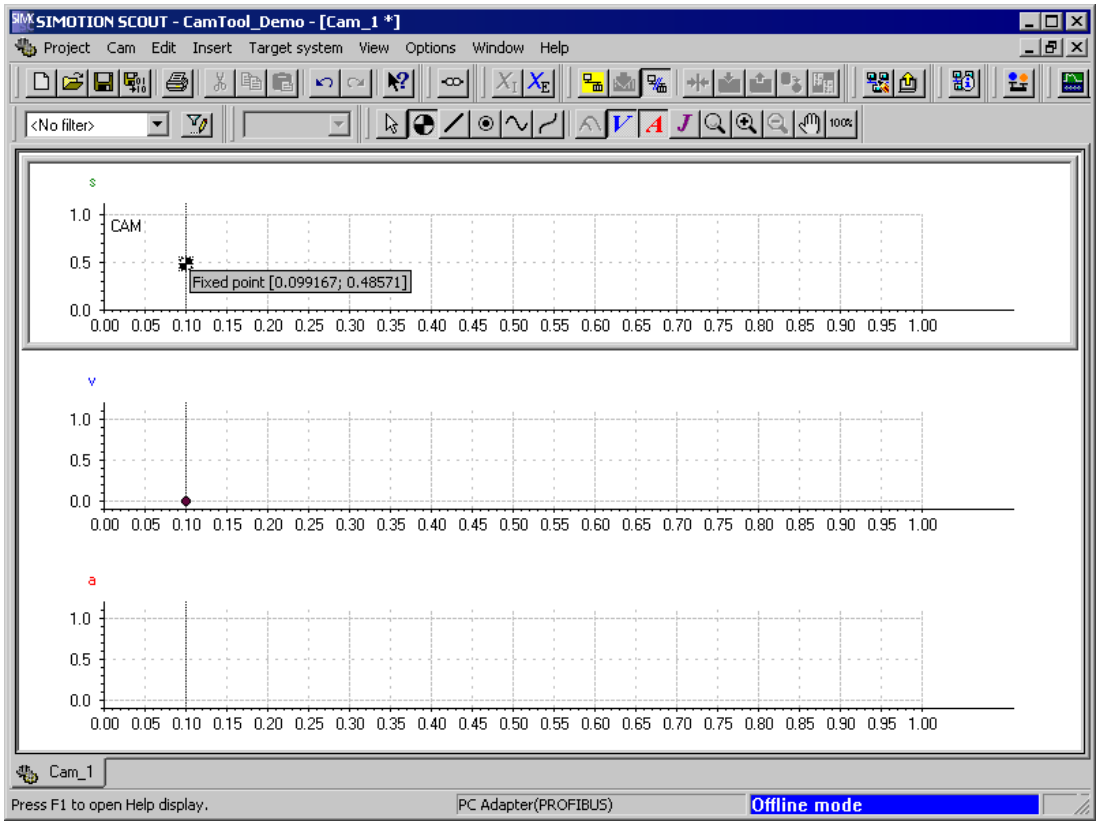


Figure 3-21 Inserting a fixed point

Note

Once you have activated the **Cam > Insert > Fixed point** function, you can continue inserting fixed points in the S diagram (path diagram) until:

- You press the ESC key
- You right-click (to activate the selection tool in the toolbar), or
- You activate another cam segment (straight line, sine curve, arc sine curve, interpolation point) for insertion.

See also

Changing the representation using the toolbar (Page 59)

Change fixed point position

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the position of a fixed point

To change the position of a fixed point:

1. Select the fixed point in the S diagram (path diagram). The position of the fixed point is shown in the tool tip.
2. Drag and drop the fixed point to the new position.

or

1. Double-click the fixed point. The **Fixed point properties** window appears.
2. Enter the fixed point's new x- or y position on the Position tab (Page 94) and click **OK**. The diagram is updated to show the fixed point's new position.

See also

Changing the representation using the toolbar (Page 59)

Changing the velocity at the position of a fixed point:

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the velocity of a fixed point

To change the velocity at the position of a fixed point:

1. The velocity at the position of a fixed point is shown in the form of a handle in the V diagram (velocity diagram).
   Select the handle. The cursor indicates the direction in which you can move the handle.
2. Drag and drop the handle to the new position.

or

1. Double-click the fixed point. The **Fixed point properties** window appears.
2. Enter the new velocity on the Dynamics tab (Page 95) in accordance with the table below and click **OK**. The handle in the V diagram (velocity diagram) appears at the new position.

Field/Button	Meaning/Operation
v =	**v** = indicates the current velocity at the position of the fixed point (see note below).
a = (manual input activated)	**a** = indicates the current acceleration at the position of the fixed point. If **Manual input** is deactivated, the system will calculate the acceleration value. The jerk diagram (J diagram) will then be constant at the fixed point. If **Manual input** is activated, you will be able to enter an acceleration value. To change the acceleration, enter the new acceleration value under **a** = and click **Accept** or **OK**. The diagrams update their display of the fixed point.
Manual input	If you activate **Manual input**, you will be able to enter an acceleration value under **a** =. If you deactivate **Manual input**, the system will calculate the acceleration value. The jerk diagram (J diagram) will then be constant at the fixed point.

Note

If you want to specify an absolute slave velocity, you must select an absolute **master velocity for calculations** in the **Simulation settings** window.

See also

Changing the representation using the toolbar (Page 59)

Changing the acceleration at a fixed point position:

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the acceleration of a fixed point

To change the acceleration at the position of a fixed point:

1. Right-click the fixed point and select **Direct Entry for Acceleration** in the context menu which appears.
2. The acceleration at the position of a fixed point is shown in the form of a handle in the A diagram (acceleration diagram).
   Select the handle. The cursor indicates the direction in which you can move the handle.
3. Drag and drop the handle to the new position.

or

1. Double-click the fixed point. The **Fixed point properties** window appears.
2. Activate the **Manual input** option on the Dynamics tab (Page 95) and enter the new acceleration value.
3. Click **OK**. The handle in the A diagram (acceleration diagram) appears at the new position.

Note

If you want to specify an absolute slave acceleration, you must select an **absolute master velocity for calculations** in the **Simulation settings** window.

See also

Changing the representation using the toolbar (Page 59)

Delete fixed point

Requirement

The cam must be opened with SIMOTION CamTool.

Deleting a fixed point

To delete a fixed point:

1. Select the fixed point you wish to delete.
2. Press the DEL key. The fixed point is deleted and all the diagrams amend their cam displays.

See also

Changing the representation using the toolbar (Page 59)

Position, fixed point properties

Position tab

Here, you specify the **position** of the fixed point in the path diagram.

You can set the following parameters:

Table 3-7 Parameters on the Position tab (Fixed point properties window)

Field/Button		Meaning/Operation
Position		
	x =	**x** = indicates the current x position of the fixed point. To change the position, enter the new position under **x** = and click **Accept** or **OK**. The diagram updates its display of the fixed point.
	y =	**y** = indicates the current y position of the fixed point. To change the position, enter the new position under **y** = and click **Accept** or **OK**. The diagram updates its display of the fixed point.

Dynamic response, fixed point properties

Dynamics tab

Here, you specify the **dynamic response** of the fixed point.

You can set the following parameters:

Table 3-8 Parameters on the Dynamics tab (Fixed point properties window)

Field/Button		Meaning/Operation
Dynamics		
	v =	**v** = indicates the current velocity at the position of the fixed point To change the velocity, enter the new velocity value under **v** = and click **Accept** or **OK**. The diagrams update their display of the fixed point. **Note:** If you want to specify an absolute slave velocity, you must select an absolute **master velocity for calculations** in the **Simulation settings** window.
	a = (manual input activated)	**a** = indicates the current acceleration at the position of the fixed point. If **Manual input** is deactivated, the system will calculate the acceleration value. The jerk diagram (J diagram) will then be constant at the fixed point. If **Manual input** is activated, you will be able to enter an acceleration value. To change the acceleration, enter the new acceleration value under **a** = and click **Accept** or **OK**. The diagrams update their display of the fixed point.
	Manual input	If you activate **Manual input**, you will be able to enter an acceleration value under **a** =. If you deactivate **Manual input**, the system will calculate the acceleration value. The jerk diagram (J diagram) will then be constant at the fixed point.

3.1.5.4 Straight line

Definition straight line

Definition

The cam consists of individual cam segments. With SIMOTION CamTool, you can use fixed points, lines, sine curves, arc sine curves, and interpolation points as cam segments.

A straight line defines a synchronous section in the cam. You can specify the velocity along the straight line.

Insert straight line

Requirement

The cam must be opened with SIMOTION CamTool.

Inserting a straight line

To insert a straight line:

1. Click the S diagram (path diagram) to activate it.
2. Click **Straight line** in the **Cam > Insert** menu item. The cursor changes appearance. Place the modified cursor at the position in the S diagram (path diagram) where you wish to insert the straight line and click. The straight line (whose size is predetermined) is inserted. The current position of the straight line is displayed in the tool tip.

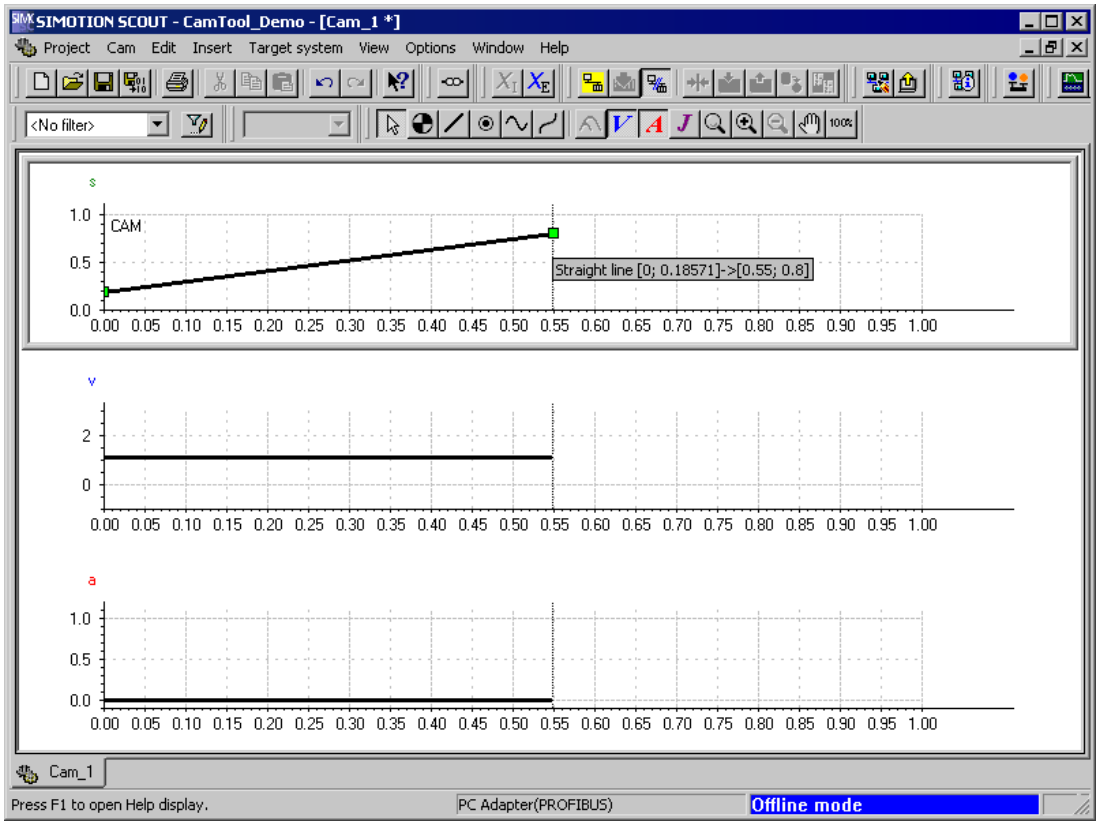


Figure 3-22 Inserting a straight line

Note

Once you have activated the **Cam > Insert > Straight line** function, you can continue inserting straight lines in the S diagram (path diagram) until:

- You press the ESC key
- You right-click (to activate the selection tool in the toolbar), or
- You activate another cam segment (fixed point, sine curve, arc sine curve, interpolation point) for insertion.

See also

Changing the representation using the toolbar (Page 59)

Change straight line position

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the position of a straight line

To change the position of a straight line:

1. Select the straight line in the S diagram (path diagram). The position of the straight line is displayed in the tool tip.
2. The straight line is displayed with handles. Drag and drop the handles for the straight line to the new positions.

or

1. Select the straight line in the S diagram (path diagram). The position of the straight line is displayed in the tool tip.
2. Drag and drop the entire straight line to the new position.

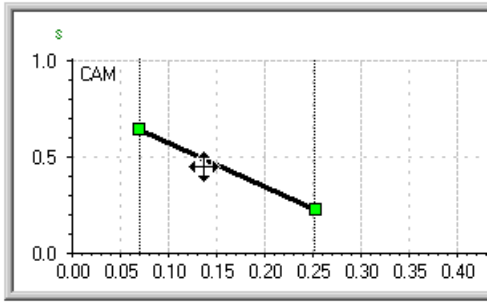


Figure 3-23 Changing the position of an entire straight line

or

1. Double-click the straight line. The **Straight line properties** window appears.
2. Enter the new positions on the Position tab (Page 100) (left: x1 and y1, right: x2 and y2) and click **OK**. The diagram is updated to show the new positions of the straight lines.

See also

Changing the representation using the toolbar (Page 59)

Changing the velocity along a straight line

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the velocity for a straight line

To change the velocity along a straight line:

1. The velocity along a straight line is represented by a line in the V diagram (velocity diagram). Select the line. The cursor indicates the direction in which you can move the line.
2. Drag and drop the entire line to the new position. The S diagram (path diagram) is adapted automatically.

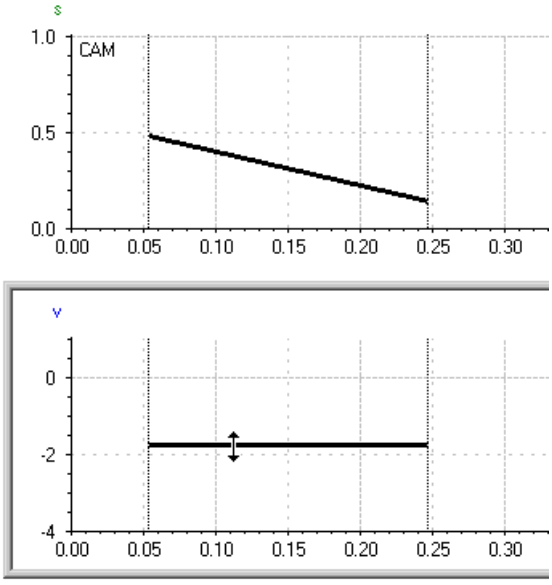


Figure 3-24 Changing the velocity along a straight line

or

1. Double-click the straight line. The **Straight line properties** window appears.
2. Enter the new velocity on the Dynamics tab (Page 100) and click **OK**. The line in the V diagram (velocity diagram) appears at the new position. The S diagram (path diagram) is adapted automatically.

Note

If you want to specify an absolute slave velocity, you must select an **absolute master velocity for calculations** in the **Simulation settings** window.

See also

Changing the representation using the toolbar (Page 59)

Delete straight line

Requirement

The cam must be opened with SIMOTION CamTool.

Deleting a straight line

To delete a straight line:

1. Select the straight line you wish to delete.
2. Press the DEL key. The straight line is deleted and all the diagrams amend their cam displays.

See also

Changing the representation using the toolbar (Page 59)

Position, straight line properties

Position tab

Here, you specify the **position** of the straight line in the path diagram.

You can set the following parameters:

Table 3-9 Parameters on the Position tab (Straight line properties window)

Field/Button			Meaning/Operation
Position			
	Left		
		x1 =	**x1** = indicates the current x1 position of the straight line. To change the position, enter the new position under **x1** = and click **Accept** or **OK**. The diagram updates its display of the straight line.
		y1 =	**y1** = indicates the current y1 position of the straight line. To change the position of the straight line, enter the new position under **y1** = and click **Accept** or **OK**. The diagram updates its display of the straight line.
	Right		
		x2 =	**x2** = indicates the current x2 position of the straight line. To change the position, enter the new position under **x2** = and click **Accept** or **OK**. The diagram updates its display of the straight line.
		y2 =	**y2** = indicates the current y2 position of the straight line. To change the position of the straight line, enter the new position under **y2** = and click **Accept** or **OK**. The diagram updates its display of the straight line.

Dynamic response, straight line properties

Dynamics tab

Here, you specify the **dynamic response** of the straight line.

You can set the following parameters:

Table 3-10 Parameters on the Dynamics tab (Straight line properties window)

Field/Button		Meaning/Operation
Dynamics		
	v =	**v** = indicates the current velocity for the straight line. To change the velocity, enter the new velocity value under **v** = and click **Accept** or **OK**. The diagrams update their display of the straight line. **Note:** If you want to specify an absolute slave velocity, you must select an absolute **master velocity for calculations** in the **Simulation settings** window.

3.1.5.5 Sine curve

Insert sine curve

Requirement

The cam must be opened with SIMOTION CamTool.

Inserting a sine curve

To insert a sine curve:

1. Click the S diagram (path diagram) to activate it.
2. Click **Sin** in the **Cam > Insert > Functions** menu item. The cursor changes appearance.
3. Place the modified cursor at the position in the S diagram (path diagram) where you wish to insert the sine curve and click. The sine curve (whose size is predetermined) is inserted. The current position of the sine curve is displayed in the tool tip.

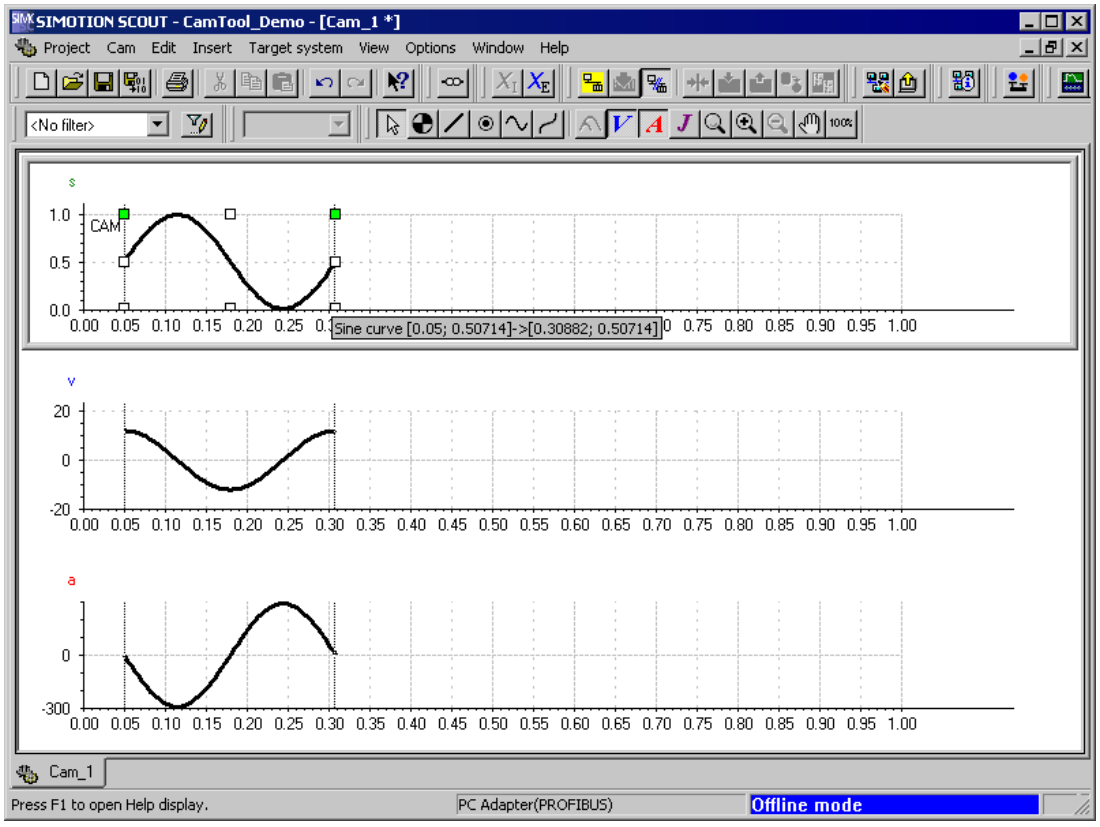


Figure 3-25 Inserting a sine curve

Note

Once you have activated the **Cam > Insert > Sin** function, you can continue inserting sine curves in the S diagram (path diagram) until:

- You press the ESC key
- You right-click (to activate the selection tool in the toolbar), or
- You activate another cam segment (fixed point, straight line, arc sine curve, interpolation point) for insertion.

See also

Changing the representation using the toolbar (Page 59)

Change sine curve position

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the position of a sine curve

To change the position of a sine curve:

1. Select the sine curve in the S diagram (path diagram).
   The current position of the sine curve is displayed in the tool tip.
2. The sine curve is displayed with handles. Place the cursor over a handle. The cursor indicates the direction in which you can move the handle.
3. Drag and drop the handles to the new positions.

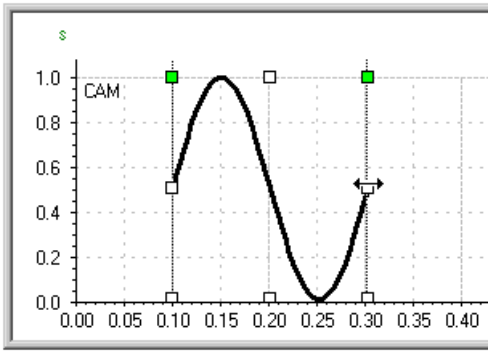


Figure 3-26 Changing the position of a sine curve

or

1. Select the sine curve in the S diagram (path diagram).
   The current position of the sine curve is displayed in the tool tip.
2. Drag and drop the entire sine curve to the new position.

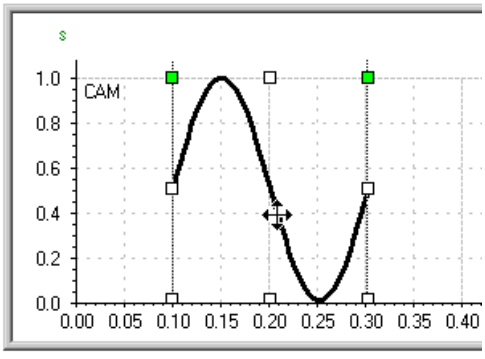


Figure 3-27 Changing the position of an entire sine curve

or

1. Double-click the sine curve. The **Fx sine properties** window appears.
2. Enter the new positions on the Position tab (Page 106) and click **OK**. The sine curve is displayed at the new positions.

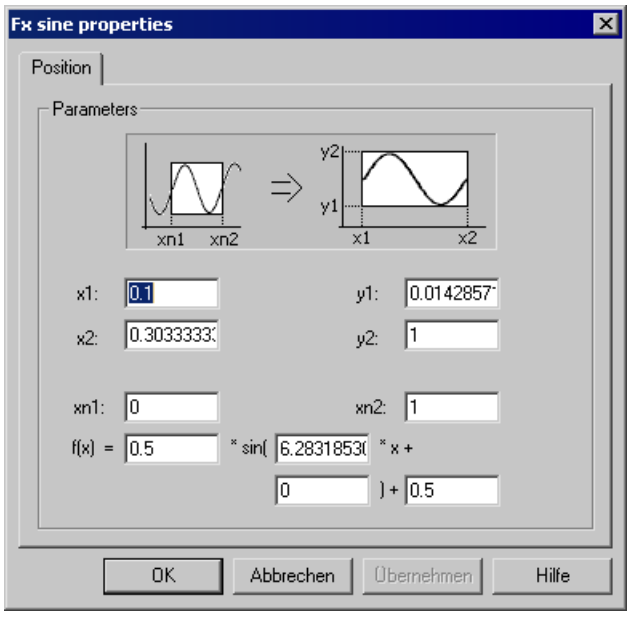


Figure 3-28 Position tab (Fx sine properties window)

See also

Changing the representation using the toolbar (Page 59)

Change sine curve definition range

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the definition range of a sine curve

To change the definition range of a sine curve:

1. Select the sine curve in the S diagram (path diagram).
   The current position of the sine curve is displayed in the tool tip.
2. The sine curve is displayed with handles. The handles for the definition range are marked in green. Place the cursor over a handle marked in green. The cursor indicates the direction in which you can move the handle.
3. Drag and drop the green handles to the new positions. While the handles are being moved, the current value for the relevant definition range limit is displayed in the tool tip.

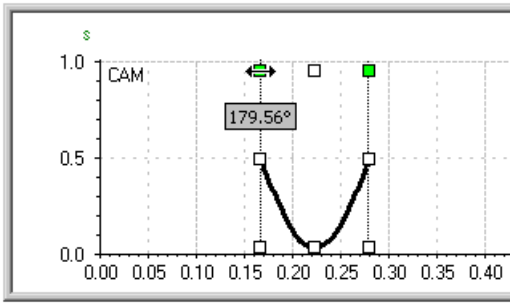


Figure 3-29 Changing the definition range of a sine curve

See also

Changing the representation using the toolbar (Page 59)

Delete sine curve

Requirement

The cam must be opened with SIMOTION CamTool.

Deleting a sine curve

To delete a sine curve:

1. Select the sine curve you wish to delete.
2. Press the DEL key. The sine curve is deleted and all the diagrams amend their cam displays.

See also

Changing the representation using the toolbar (Page 59)

Position, Fx sine properties

Position tab

Here, you specify the **position** of the sine curve in the path diagram.

You can set the following parameters:

Table 3-11 Parameters on the Position tab (Fx sine properties window)

Field/Button	Meaning/Operation
Parameters	The current parameters for the sine curve are shown in the input fields. To change the position, enter the new position in the input fields and click **Accept** or **OK**. The diagram updates its display of the sine curve.

3.1.5.6 Arc Sine Curve

Insert arc sine curve

Requirement

The cam must be opened with SIMOTION CamTool.

Note

The arc sine curve is calculated as an interpolation point table. You can specify the number of interpolation points in the **Fx arc sine properties** window. Double-click the arc sine curve to show the window.

Inserting an arc sine curve

To insert an arc sine curve:

1. Click the S diagram (path diagram) to activate it.
2. Click **Arcsin** in the **Cam > Insert > Functions** menu item. The cursor changes appearance.
3. Place the modified cursor at the position in the S diagram (path diagram) where you wish to insert the arc sine curve and click. The arc sine curve (whose size is predetermined) is inserted.
   The current position of the arc sine curve is displayed in the tool tip.

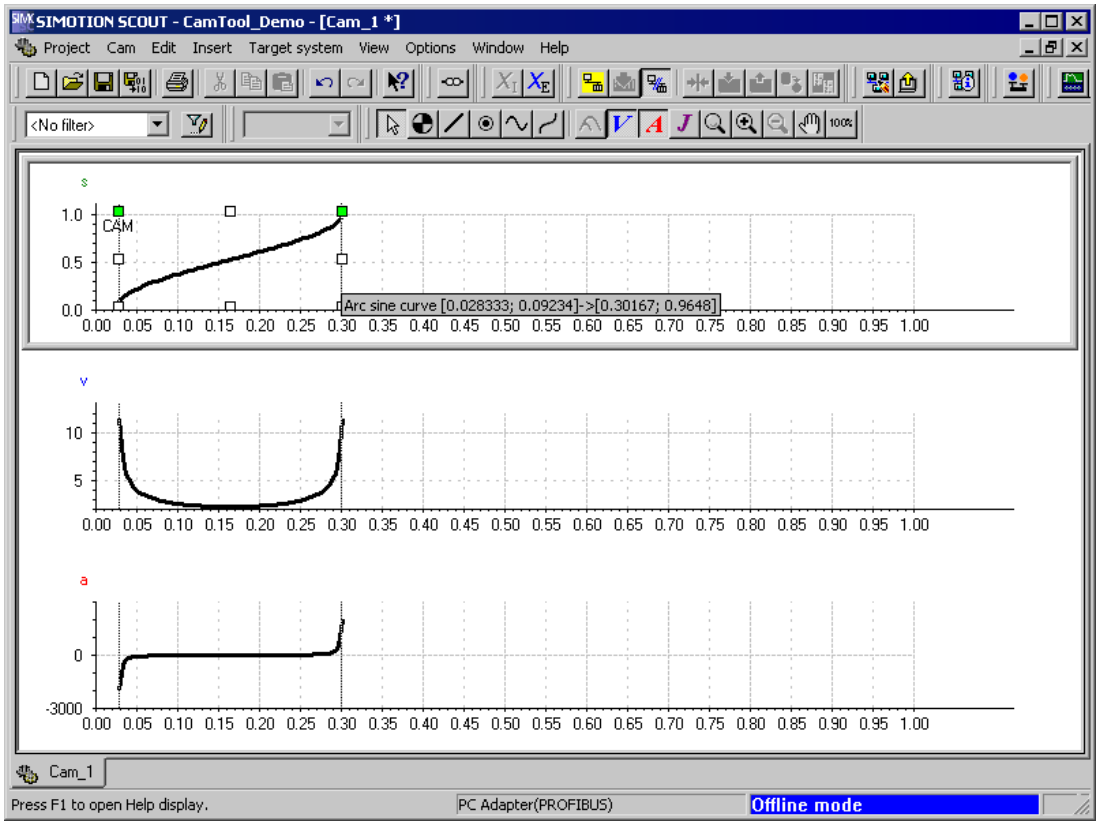


Figure 3-30 Inserting an arc sine curve

Note

Once you have activated the **Cam > Insert > Arcsin** function, you can continue inserting arc sine curves in the S diagram (path diagram) until:

- You press the ESC key
- You right-click (to activate the selection tool in the toolbar), or
- You activate another cam segment (fixed point, straight line, sine curve, interpolation point) for insertion.

See also

Changing the representation using the toolbar (Page 59)

Change arc sine curve position

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the position of an arc sine curve

To change the position of an arc sine curve:

1. Select the arc sine curve in the S diagram (path diagram).
   The current position of the arc sine curve is displayed in the tool tip.
2. The arc sine curve is displayed with handles. Place the cursor over a handle. The cursor indicates the direction in which you can move the handle.
3. Drag and drop the handles to the new positions.

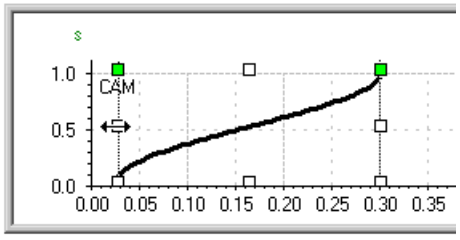


Figure 3-31 Changing the position of an arc sine curve

or

1. Select the arc sine curve in the S diagram (path diagram).
   The current position of the arc sine curve is displayed in the tool tip.
2. Drag and drop the entire arc sine curve to the new position.

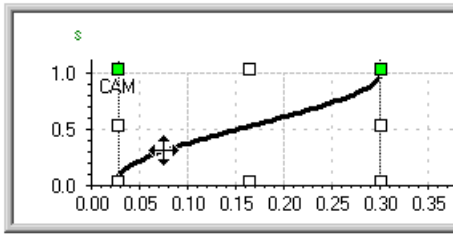


Figure 3-32 Changing the position of an entire arc sine curve

or

1. Double-click the arc sine curve.
   The **Fx arc sine properties** window appears.
2. Enter the new positions and the number of interpolation points on the Position tab (Page 111) and click OK. The arc sine curve is displayed at the new positions. The system saves the arc sine curve as an interpolation point table.

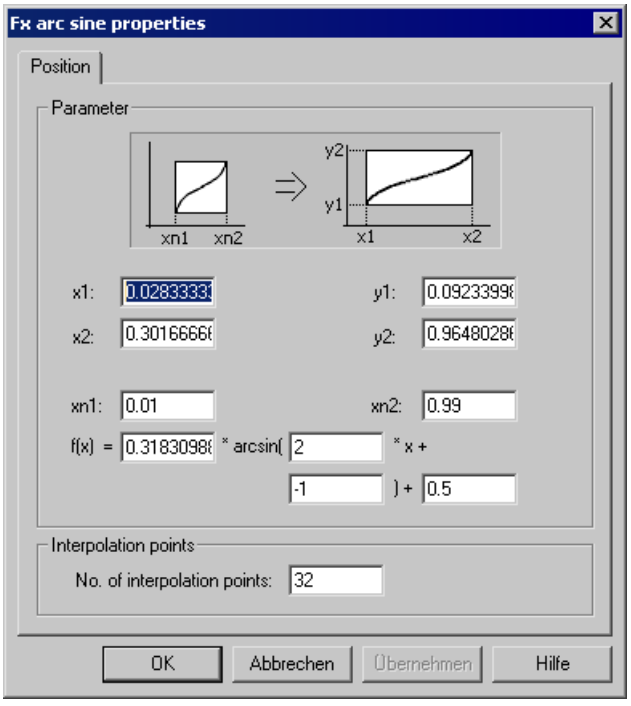


Figure 3-33 Position tab

See also

Changing the representation using the toolbar (Page 59)

Change arc sine curve definition range

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the definition range of an arc sine curve

To change the definition range of an arc sine curve:

1. Select the arc sine curve in the S diagram (path diagram).
   The current position of the arc sine curve is displayed in the tool tip.
2. The arc sine curve is displayed with handles. The handles for the definition range are marked in green. Place the cursor over a handle marked in green. The cursor indicates the direction in which you can move the handle.
3. Drag and drop the green handles to the new positions. While the handles are being moved, the current value for the relevant definition range limit is displayed in the tool tip.

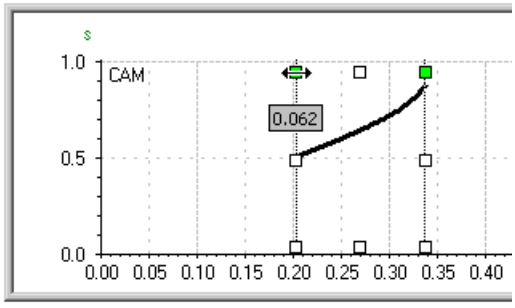


Figure 3-34 Changing the definition range of an arc sine curve

See also

Changing the representation using the toolbar (Page 59)

Delete arc sine curve

Requirement

The cam must be opened with SIMOTION CamTool.

Deleting an arc sine curve

To delete an arc sine curve:

1. Select the arc sine curve you wish to delete.
2. Press the DEL key. The arc sine curve is deleted and all the diagrams amend their cam displays.

See also

Changing the representation using the toolbar (Page 59)

Position, Fx arc sine properties

Position tab

Here, you specify the **position** of the arc sine curve in the path diagram.

You can set the following parameters:

Table 3-12 Parameters on the Position tab (Fx arc sine properties window)

Field/Button		Meaning/Operation
Parameters		The current parameters for the arc sine curve are shown in the input fields. To change the position, enter the new position in the input fields and click **Accept** or **OK**. The diagram updates its display of the arc sine curve.
Interpolation points		
	No. of interpolation points	The system saves the arc sine curve as an interpolation point table. Here, you enter the **no. of interpolation points** for the arc sine curve.

3.1.5.7 Interpolation point

Interpolation Point Definition

Definition

The cam consists of individual cam segments. Under SIMOTION CamTool, you can use fixed points, straight lines, sine curves, arc sine curves and interpolation points as cam segments.

An interpolation point is a specified position in the S diagram (distance diagram).

Insert interpolation point

Requirement

The cam must be opened with SIMOTION CamTool.

Note

Use a fixed point (Page 90) to specify a single, fixed position. CamTool calculates optimum transitions between adjacent cam segments.

Interpolation points must be used for any profile you wish to define. CamTool links the interpolation points with a cubic spline in order to create the profile specified by the interpolation points as accurately as possible.

Inserting an interpolation point

To insert an interpolation point:

1. Click the S diagram (path diagram) to activate it.
2. Click **Interpolation point** in the **Cam > Insert** menu item. The cursor changes appearance.
3. Place the modified cursor at the position in the S diagram (path diagram) where you wish to insert the interpolation point and click.
   The interpolation point appears in the diagram. The position of the interpolation point is shown in the tool tip.

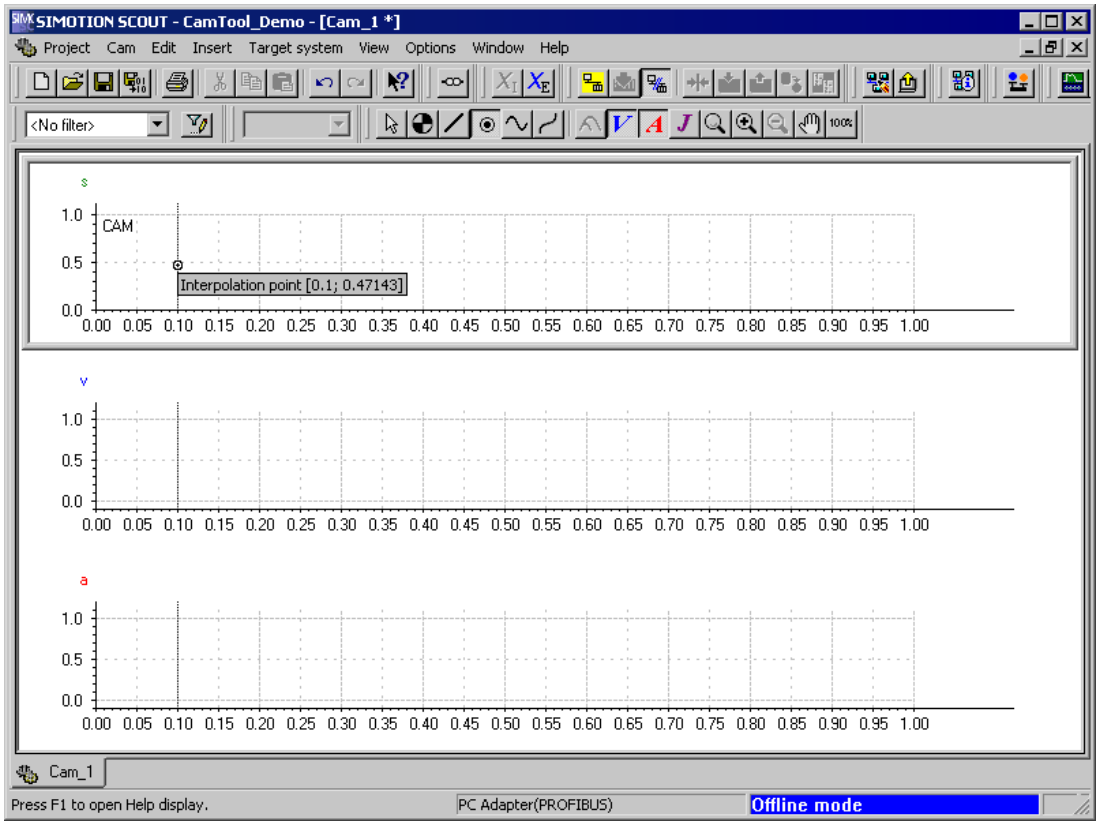


Figure 3-35 Inserting an interpolation point

Note

Once you have activated the **Cam > Insert > Interpolation point** function, you can continue inserting interpolation points in the S diagram (path diagram) until:

- You press the ESC key
- You right-click (to activate the selection tool in the toolbar), or
- You activate another cam segment (fixed point, straight line, sine curve, arc sine curve) for insertion.

See also

Changing the representation using the toolbar (Page 59)

Changing the interpolation point position

Requirement

The cam must be opened with SIMOTION CamTool.

Changing the position of an interpolation point

To change the position of an interpolation point:

1. Select the interpolation point in the S diagram (path diagram). The position of the interpolation point is shown in the tool tip.
2. Drag and drop the interpolation point to the new position.

or

1. Double-click the interpolation point. The Interpolation point properties window appears.
2. Enter the new position on the Position tab (Page 114) and click **OK**. The interpolation point appears at the new position.

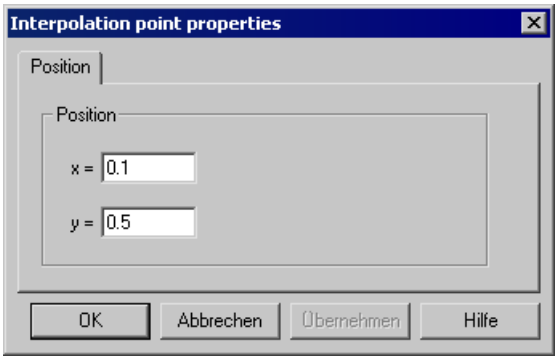


Figure 3-36 Position tab

See also

Changing the representation using the toolbar (Page 59)

Delete interpolation point

Requirement

The cam must be opened with SIMOTION CamTool.

Deleting an interpolation point

To delete an interpolation point:

1. Select the interpolation point you wish to delete.
2. Press the DEL key. The interpolation point is deleted and all the diagrams amend their cam displays.

See also

Changing the representation using the toolbar (Page 59)

Position, interpolation point properties

Position tab

Here, you specify the **position** of the interpolation point in the path diagram.

You can set the following parameters:

Table 3-13 Parameters on the Position tab (Interpolation point properties window)

Field/Button		Meaning/Operation
Position		
	x =	**x** = indicates the current x position of the interpolation point. To change the position, enter the new position under **x** = and click **Accept** or **OK**. The diagram updates its display of the interpolation point.
	y =	**y** = indicates the current y position of the interpolation point. To change the position, enter the new position under **y** = and click **Accept** or **OK**. The diagram updates its display of the interpolation point.

3.1.5.8 Optimizing a Cam

Optimizing a Cam

Introduction

Create the cam in the S diagram (path diagram). The cam profile reflects the path-related interdependence between the master axis (X-axis in the diagram) and the slave axis (Y-axis in the diagram).

Cam structure

The cam consists of individual cam segments.

In SIMOTION CamTool, you can use fixed points, straight lines, sine curves, arc sine curves and interpolation points as cam segments.

SIMOTION CamTool calculates interpolation curves between individual cam segments and displays the V diagram (velocity diagram), A diagram (acceleration diagram), and J diagram (jerk diagram).

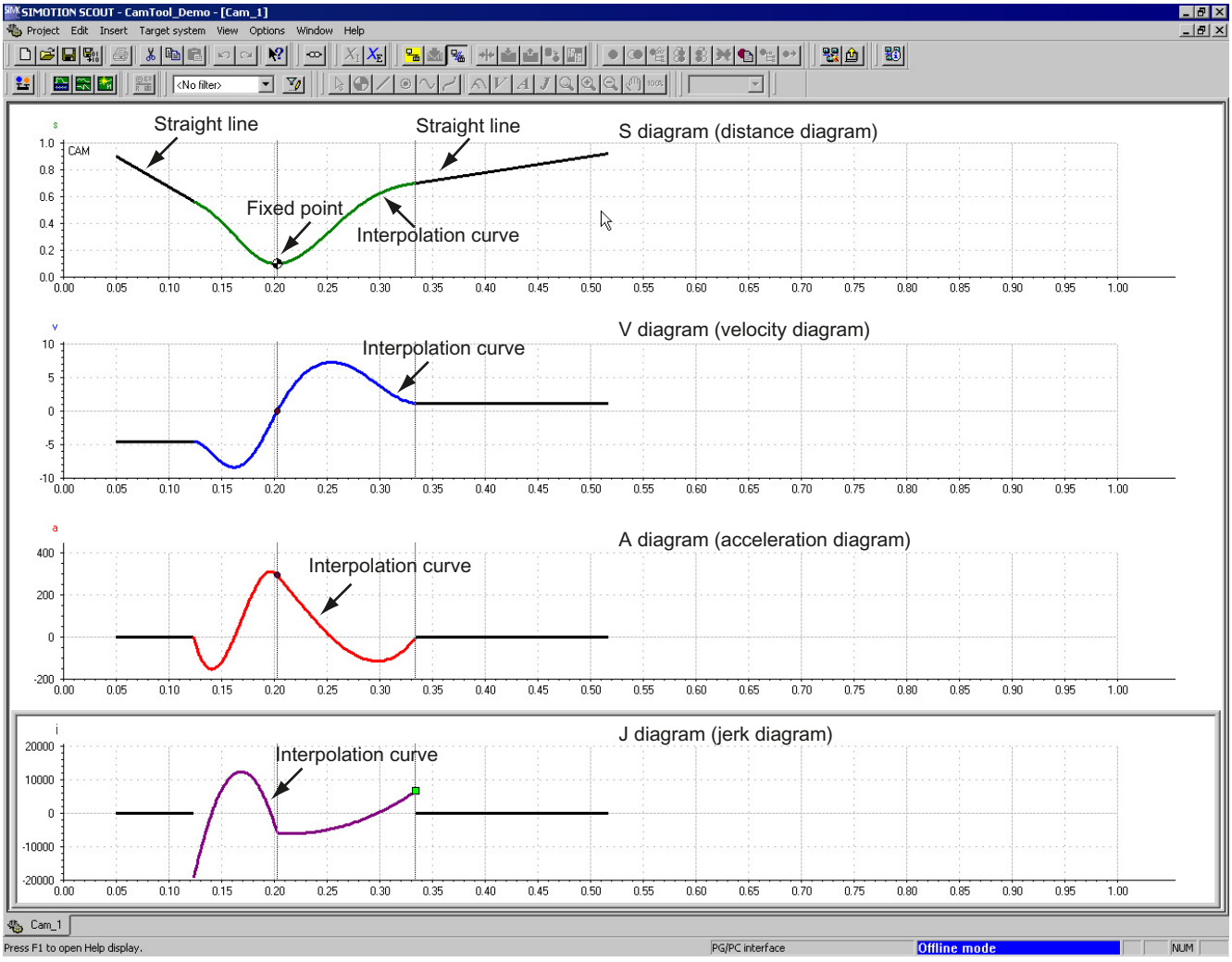


Figure 3-37 Example of a cam with interpolation curves

Interpolating several cam sections

You can interpolate all interpolation curves either with CamTool or by the target device. The default setting is CamTool.

Example:

You can convert a cam that was created with CamEdit as an interpolation point table to a CamTool cam. In this case, all cam segments are interpolated by the target device.

See also

Cam menu - CamTool (Page 134)

Optimize transition

Requirement

The cam must be opened with SIMOTION CamTool.

Optimizing a transition

To optimize a transition:

1. Right-click the transition you wish to optimize and select **Properties** in the context menu which appears.

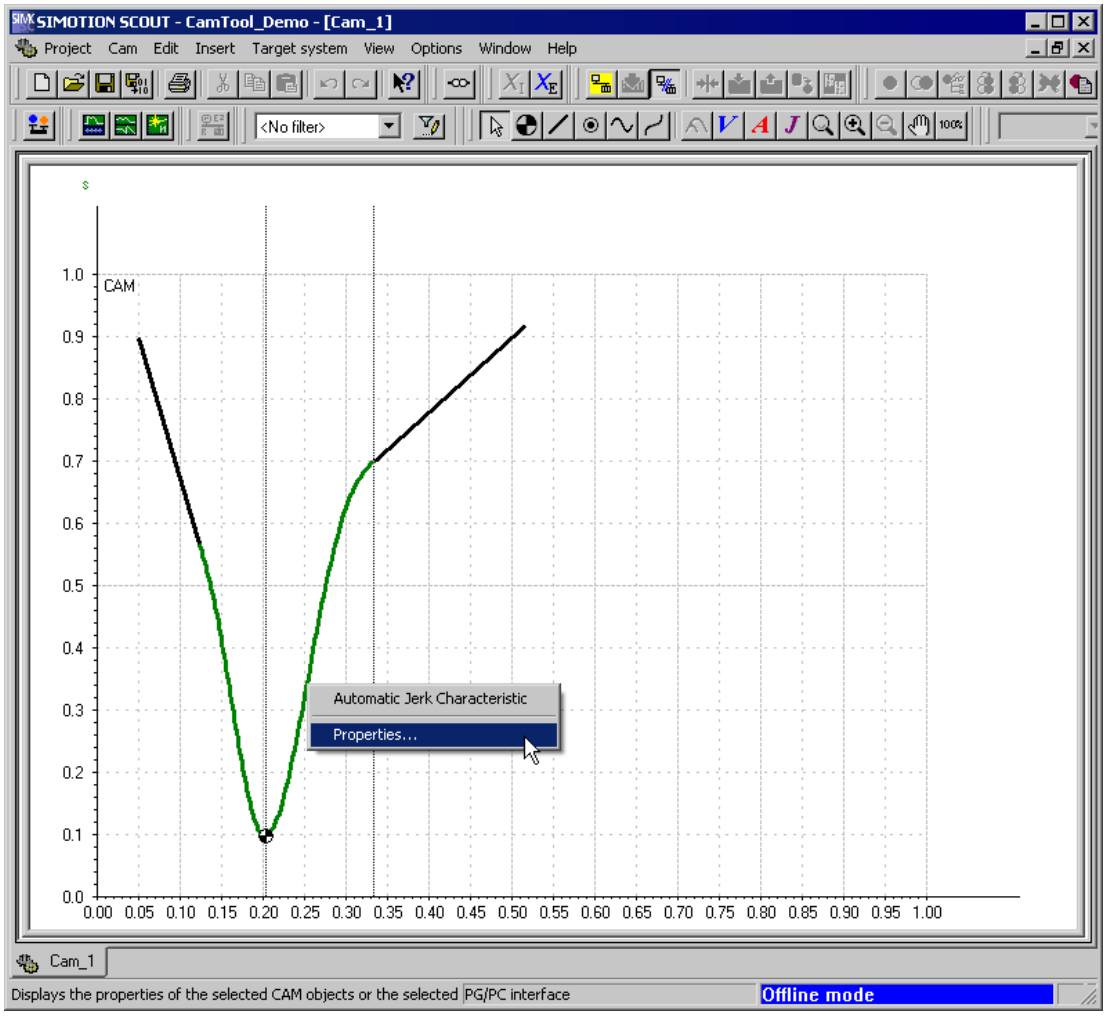


Figure 3-38 Selecting a transition for optimization

2. The **Interpolation curve properties** window appears. Specify the parameters on the Type tab (Page 118).

3. Click the Parameters tab (Page 118) and specify the parameters.
4. Click the Limit values tab (Page 119) to display the minimum and maximum values for the interpolation curve.

Note

Where the slave axis is concerned, you can accept the settings of an existing axis in the project via **Cam > Simulation Settings** and specify the axis limits. These simulation settings for the slave axis are used to calculate the percentage values displayed on the **Limit values** tab.

or

1. For the purpose of optimizing the cam, use the handles which, depending on the settings in the **Interpolation curve properties** window, are shown in the individual diagrams.
2. When positioned over a handle, the cursor indicates the direction in which you can move the handle.
3. Drag and drop the handle to the new position. All the diagrams amend their cam displays.

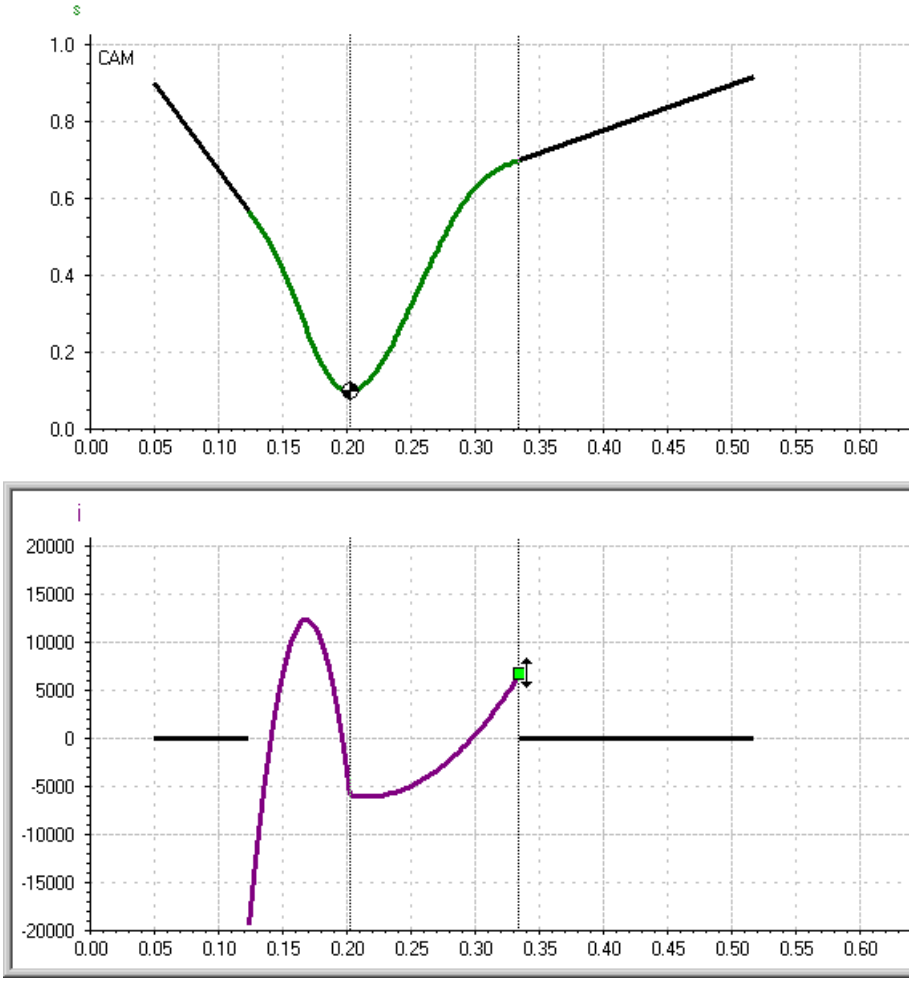


Figure 3-39 Optimizing a cam by dragging and dropping a handle

See also

Changing the representation using the toolbar (Page 59)

Cam menu - CamTool (Page 134)

Interpolation Curve properties

Here, you specify the parameters for the interpolation curve transition.

Type - Interpolation curve properties

Type

Here, you specify the **type** of interpolation curve for the transition.

You can set the following parameters:

Table 3-14 Parameters on the Type tab (Interpolation curve properties window)

Field/Button		Meaning/Operation
Transition		The type of transition is displayed here.
Allow interpolation by target device		If you activate this option, the system does not generate any interpolation curves. The transition is interpolated by the target device. If you click Accept or OK, the diagrams will update their display of the transition.
Optimize for (Allow interpolation by target device deactivated)		
	Do not optimize	If you select this option, all interpolation curves which are suitable for the transition are displayed under Applicable motion rules.
	Velocity	If you select this option, all interpolation curves which are suitable for optimizing velocity for the transition are displayed under Applicable motion rules.
	Acceleration	If you select this option, all interpolation curves which are suitable for optimizing acceleration for the transition are displayed under Applicable motion rules.
	Jerk	If you select this option, all interpolation curves which are suitable for optimizing jerk for the transition are displayed under Applicable motion rules.
Applicable motion rules (Allow interpolation by target device deactivated)		Applicable motion rules are displayed here. Select the motion rule you wish to use for the transition. If you click Accept or OK, the diagrams will update their display of the transition. Depending on the motion rule used, you can specify additional parameters on the Parameters tab.

Parameters - Interpolation curve properties

Parameters

Here, you specify the interpolation curve **parameters** for the transition.

You can set the following parameters:

Table 3-15 Parameters on the Parameters tab (Interpolation curve properties window)

Field/Button		Meaning/Operation
Jerk at boundary point (Allow interpolation by target device deactivated) (For a 6th order polynomial)		
	Automatic	If you activate **Automatic**, the system calculates the jerk at the boundary points. A 5th order polynomial is used for the calculation. **Note:** You can also activate this function via the context menu in the diagram display: Right-click the transition and select **Automatic Jerk Characteristic** in the context menu.
	Left (Automatic deactivated)	If you activate **Left**, you will be able to specify the jerk at the interpolation curve's left boundary point. Enter the jerk value in the input field.
	Right (Automatic deactivated)	If you activate **Right**, you will be able to specify the jerk at the interpolation curve's right boundary point. Enter the jerk value in the input field.
	(Jerk value) (Automatic deactivated)	If you activate the **Left** or **Right** option, you will be able to enter the jerk value here. **Note**: If you want to specify an absolute jerk value, you must select an absolute **master velocity for calculations** in the **Simulation settings** window.
Turning point of curve (Allow interpolation by target device deactivated) (For sloping sine curve) (For modified sine curve) (For modified acceleration trapezoid) (For harmonic combination)		
	Valid values are between 0 and 1	The turning point of the curve can be entered here.
	Optimum Value	If you click this button, the **optimum value** for the turning point of the curve is calculated by the system and entered under **Valid values are between 0 and 1**. **Note**: You can also activate this function via the context menu in the diagram display: Right-click the transition and select **Optimum Value** in the context menu.

Limit Values - Interpolation curve properties

Limit Values

The interpolation curve's **limit values** for the transition are displayed here.

You can set the following parameters:

Table 3-16 Parameters on the Limit Values tab (Interpolation curve properties window)

Field/Button		Meaning/Operation
Position		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.
Velocity		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.
Acceleration (v > 0)		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.
Acceleration (v < 0)		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.
Jerk (v > 0)		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.

Field/Button		Meaning/Operation
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.
Jerk (v < 0)		
	Min.	The minimum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is undershot, the minimum value will be displayed against a red background.
	Max.	The maximum value is displayed as an absolute value and as a percentage value here, depending on the physical limit value. If the physical limit value is overshot, the maximum value will be displayed against a red background.

Note

Where the slave axis is concerned, you can accept the settings of an existing axis in the project via Cam > Simulation Settings and specify the axis limits. These simulation settings for the slave axis are used to calculate the percentage values displayed on the Limit Values tab.

3.1.5.9 Specify parameters for target device

Introduction

You can use the parameters for the target device to adapt the cam to the SIMOTION device (e.g. by scaling the cam).

Requirement

The cam must be opened with SIMOTION CamTool.

Specifying parameters for the target device

To specify parameters for the target device:

1. Click a diagram to show the **Cam** menu.
2. Click the **Cam >Target Device Parameters** menu item.
   The **Target device parameters** window appears.

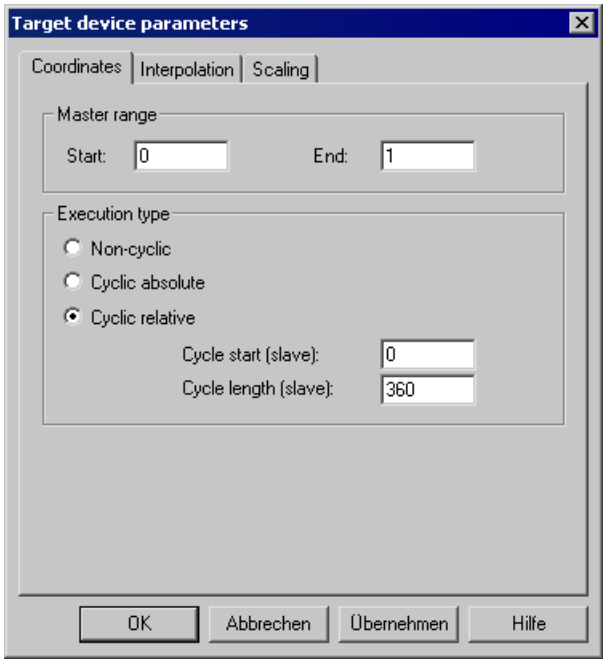


Figure 3-40 Target device parameters window

3. Specify the parameters on the Coordinates tab (Page 123).
4. Click the Interpolation tab (Page 124) and specify the parameters.
5. Click the Scaling tab (Page 125) and specify the parameters.

Note

If you specify scaling details, you will be able to show or hide the scaled cam profile via **Cam > Representation parameters > Display Scaling**.

When the scaled cam profile is displayed, you will not be able to edit the original curve in the diagram displays.

See also

Changing the representation using the toolbar (Page 59)

Target Device Parameters

Here, you specify parameters for the target device.

Coordinates - Target device parameters

Coordinates

Here, you specify the **coordinates** for the target device.

You can set the following parameters:

Table 3-17 Parameters on the Coordinates tab (position window)

Field/Button			Meaning/Operation
Master range			Specify the **master range** (definition range) for the curve by setting its start and end points. Cam segments must be contained within this master range (see also **Master range** under **S diagram (master) properties**.
	Start		Here, you enter the **start** point of the curve or the start of the curve's master range.
	End		Here, you enter the **end** point of the curve or the end of the curve's master range.
Execution type			The **execution type** specifies the cam's interpolation via the target device at the master range's boundary points.
	Non-cyclic		If you select **Non-cyclic** as the execution type, the cam's profile will end when the end value of the curve is reached. In general, the function values of the cam (slave position) are different at the start and end points of the master range. **Note:** If you activate **Non-cyclic**, the master range will be displayed once in the horizontal direction.
	Cyclic absolute		If you select the **Cyclic absolute** execution type, the position will revert from the end value to the initial value of the slave axis and the cam will be repeated. The cam's function values (slave position) at the start and end points of the master range are the same. If necessary, the target device will force these values to be the same (mean) value. **Note**: If you activate **Cyclic absolute**, the master range can be displayed in the horizontal direction again.
	Cyclic relative		If you select the **Cyclic relative** execution type, the cam will be repeated (starting from the end value of the slave axis). The cam's 1st derivative (velocity) at the start and end points of the master range are the same. If necessary, the target device will force these values to be the same (mean) value. **Note:** If you activate **Cyclic relative**, the master range can be displayed in the horizontal and vertical direction again. You can use the **Cycle start (slave)** and **Cycle length (slave)** parameters to optimize the display.
		Cycle start (slave) (Cyclic relative activated)	Enter the starting point for the display in the vertical direction under **Cycle start (slave)**.
		Cycle length (slave) (Cyclic relative activated)	Enter the length of the range displayed in the vertical direction under **Cycle length (slave)**.

Interpolation - position

Interpolation

Here, you specify the **interpolation** for the target device.

You can set the following parameters:

Table 3-18 Parameters on the Interpolation tab (position window)

Field/Button			Meaning/Operation
Interpolation type			The curve in the master range can consist of individual segments. Gaps may occur between the curve segments. If the gaps are wider than the maximum specified size, the target device will perform interpolation between the end points of the curve segments. Specify the type of interpolation performed by the target device between the gaps under **Interpolation type**.
	Linear		If you select **Linear** as the interpolation type, the gap will be closed smoothly by the insertion of straight-line sections between the interpolation points.
	Cubic splines		If you select **Cubic splines** as the interpolation type, a smooth method capable of differentiation will be used to close the gap. The curve runs through the specified interpolation points.
	Bezier splines		If you select **Bezier splines** as the interpolation type, the approximation curve will follow the specified interpolation points.
With gaps less than...			
	Master		
		Maintain gaps	Gaps may occur between the curve segments of the master axis. Here, you specify the maximum size of gaps to be retained.
		Merge points	Gaps may occur between the curve segments of the master axis. Here, you specify the maximum size of gaps to be closed by merging the end points.
	Slave		
		Maintain gaps	Gaps may occur between the curve segments of the slave axis. Here, you specify the maximum size of gaps to be retained.
		Merge points	Gaps may occur between the curve segments of the slave axis. Here, you specify the maximum size of gaps to be closed by merging the end points.

Scaling - position

Scaling

Here, you specify the **scaling** for the target device.

Note

Select **Cam > Representation parameters > Display Scaling** to display the scaled cam profile.

When the scaled cam profile is displayed, you will not be able to edit the original curve. To re-enable editing of the original curve,
select **Cam > Representation parameters > Display Scaling** again.

If a cam is scaled by selecting **Cam > Target Device Parameters** and the **Basic scaling** input field on the **Scaling** tab, or a cam which has already been scaled in SIMOTION CamEdit is opened in SIMOTION CamTool, the scaled cam profile will only be displayed for the S diagram (path diagram). The V, A, and J diagrams will only show the original (unscaled) cam profile.

You can set the following parameters:

Table 3-19 Parameters on the Scaling tab (Target device parameters window)

Field/Button			Meaning/Operation
Master axis			
	Basic scaling		Here, you enter the **basic scaling** for the master axis. The coordinate origin always forms the center of scaling. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
	Scalings		Here, you can specify the **scalings** for two master axis ranges. The center of scaling forms the start point of the scaling range. **Note**: Make sure there is no overlap when specifying two ranges.
		(Range1)	Enter the range you wish to scale under **From** and **To**. Enter the scaling factor under **Factor**. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
		(Range2)	Enter the range you wish to scale under **From** and **To**. Enter the scaling factor under **Factor**. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
	Offset		Here, you can enter a factor for the master axis **offset**. If you have specified a scaling, the offset will relate to the scaled curve. If you have not specified a scaling, the offset will relate to the unscaled curve.
Slave axis			

Field/Button			Meaning/Operation
	Basic scaling		Here, you enter the **basic scaling** for the slave axis. The coordinate origin always forms the center of scaling. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
	Scalings		Here, you can specify the **scalings** for two slave axis ranges. The center of scaling forms the start point of the scaling range. **Note**: Make sure there is no overlap when specifying two ranges.
		(Range1)	Enter the range you wish to scale under **From** and **To**. Enter the scaling factor under **Factor**. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
		(Range2)	Enter the range you wish to scale under **From** and **To**. Enter the scaling factor under **Factor**. **Note**: Do not use any scaling values greater than 5. Greater values cause a roughness in the cam profile
	Offset		Here, you can enter a factor for the slave axis **offset**. If you have specified a scaling, the offset will relate to the scaled curve. If you have not specified a scaling, the offset will relate to the unscaled curve.

3.1.5.10 Specify simulation settings

Introduction

You can either take the axis settings for the master and slave axes from axes configured in the SCOUT project or specify your own.

Requirement

The cam must be opened with SIMOTION CamTool.

Specifying simulation settings

To specify the simulation settings:

1. Click a diagram to show the **Cam** menu.
2. Click the **Cam >Simulation Settings** menu item. The Simulation settings window opens.

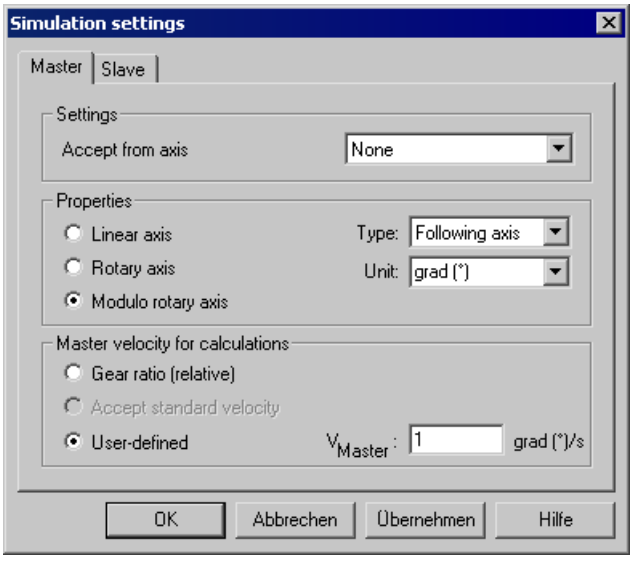


Figure 3-41 Simulation settings window

3. Specify the parameters on the Master tab (Page 127).
4. Click the Slave tab (Page 129) and specify the parameters.

See also

Changing the representation using the toolbar (Page 59)

Simulation Settings

Here, you specify the simulation settings for the **master** and **slave** axes.

Master - Simulation settings

Master tab

Here, you specify the simulation settings for the **master** axis.

You can set the following parameters:

Table 3-20 Parameters on the Master tab (Simulation settings window)

Field/Button		Meaning/Operation
Setting		
	Accept from axis	• **(Axis)** If you are creating the cam for an existing axis in the project, you can select the axis here. The configured values are accepted and displayed. You can change the **master velocity for calculations**. • **(External encoder)** If you are creating the cam for an existing external encoder in the project, you can select the external encoder here. The configured values are accepted and displayed. • **None** If you select **None**, you can specify all the parameters. • **Time axis** If you select **Time axis**, you can to specify the **dimension unit** under **Properties**.
Properties (None setting) (Time axis setting)		
	Linear axis (None setting)	If you select this option, you will simulate the master axis as a **linear axis**.
	Rotary axis (None setting)	If you select this option, you will simulate the master axis as a **rotary axis**.
	Modulo rotary axis (None setting)	If you select this option, you will simulate the master axis as a **modulo rotary axis**.
	Type (None setting)	Here, you select the **type** of axis.
	Dimension unit (None setting) (Time axis setting)	Here, you specify the dimension unit for the axis. Select a dimension unit or enter the dimension unit. If you click **Accept** or **OK**, the diagrams will update their display of the dimension unit. **Note**: The dimension unit is only used for display purposes. If the dimension unit is changed, the values displayed will not be converted.
Master velocity for calculations (Setting (axis)) (None setting)		
	Gear ratio (relative)	If you activate this option, all calculations will be based on relative values (values relative to the master axis).
	Accept standard velocity (Setting (axis))	If you activate this option, the configured default velocity for the axis which you defined under **Accept from axis** will be accepted. All calculations will be based on absolute values.

Field/Button		Meaning/Operation
	User-defined	If you activate this option, you must enter a velocity under **V Master**. All calculations will be based on absolute values.
	V Master (User-defined activated)	If you activate User-defined, you must enter the master velocity under **V Master**.

Slave - Simulation settings

Slave tab

Here, you specify the simulation settings for the **slave** axis.

You can set the following parameters:

Table 3-21 Parameters on the Slave tab (Simulation settings window)

Field/Button		Meaning/Operation
Setting		
	Accept from axis	• **(Axis)** If you are creating the cam for an existing axis in the project, you can select the axis here. The configured values are accepted and displayed. • **None** If you select **None**, you can specify all the parameters.
Properties (None setting)		
	Linear axis	If you select this option, you will simulate the slave axis as a **linear axis**.
	Rotary axis	If you select this option, you will simulate the slave axis as a **rotary axis**.
	Modulo rotary axis	If you select this option, you will simulate the slave axis as a **modulo rotary axis**.
	Type	Here, you select the **type** of axis.
	Dimension unit	Here, you specify the dimension unit for the axis. Select a dimension unit or enter the dimension unit. If you click **Accept** or **OK**, the diagrams will update their display of the dimension unit. **Note**: The dimension unit is only used for display purposes. If the dimension unit is changed, the values displayed will not be converted.
Limit (None setting)		
	Minimum	Here, you enter the **minimum** limit for the slave axis.
	Maximum	Here, you enter the **maximum** limit for the slave axis.

Field/Button		Meaning/Operation
Modulo values (None setting) (Simulation slave-axis activated as a modulo rotary axis)		
	Basis	If Modulo rotary axis is activated for slave axis simulation, you must enter the modulo **basis** here.
	Length	If Modulo rotary axis is activated for slave axis simulation, you must enter the modulo **length** here.

3.1.5.11 Edit cam created with CamTool with CamEdit

Converting a cam created with CamTool to CamEdit

To edit a cam created with CamTool using CamEdit:

1. Close the cam in SIMOTION CamTool.
2. In the project navigator, find the cam you wish to edit with SIMOTION CamEdit.
3. Right-click the cam and select **Convert to CamEdit** in the context menu that appears. The cam will be opened the next time SIMOTION CamEdit is opened.

Note

You can use SIMOTION CamEdit to edit any cam containing either polynomials only or interpolation points only. In general, such cams are those created with SIMOTION CamTool whose cam segments do not contain arc sine curves.

See also

Changing the representation using the toolbar (Page 59)

3.1.5.12 Print cam

Introduction

You can use SIMOTION SCOUT to print a cam opened with SIMOTION CamTool. The printout contains the cam's individual parameters and diagrams.

Note

The cam's printed diagrams reflect what can be seen on the screen view.

Printing a cam

To print a cam:

1. Open the cam with SIMOTION CamTool.
2. Click a diagram to show the **Cam** menu.
3. Use the **Cam > Diagrams** menu item to activate the diagrams you wish to display. The diagrams are displayed.
4. Click the **Project > Print** menu item. The printer selection window appears.
5. Specify the printer, printing range, and number of copies.
6. Click **OK**. The cam is printed out.

See also

Changing the representation using the toolbar (Page 59)

3.1.6 Menus

3.1.6.1 Arcsin, Sin, Straight line, Interpolation point context menu - CamTool

You can select the following functions:

Function	Meaning/information
Delete	Use this function to delete the cam object (arcsin, sin, straight line, or interpolation point) from the S diagram (path diagram).
Properties	Use this function to specify the parameters for the cam segment (arcsin, sin, straight line, or interpolation point).

3.1.6.2 Diagram area context menu - CamTool

You can select the following functions:

Function			Meaning/information
Insert			
	Functions		
		Arcsin	Use this function to insert an arc sine curve into the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note:** You can change the position of the arc sine curve after inserting it, by either dragging and dropping it or pulling the handles at the ends of the curve. You can also do this via the **Properties** context menu.

Function	Meaning/information
Sin	Use this function to insert a sine curve into the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note:** You can change the position of the sine curve after inserting it, by either dragging and dropping it or pulling the handles at the ends of the curve. You can also do this via the **Properties** context menu.
Fixed point	Use this function to insert a fixed point into the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note:** You can change the position of the fixed point after inserting it, by either dragging and dropping it or using the **Properties** context menu.
Straight line	Use this function to insert a straight line into the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note:** You can change the position of the straight line after inserting it, by either dragging and dropping it or pulling the handles at the ends of the line. You can also do this via the **Properties** context menu.
Interpolation point	Use this function to insert an interpolation point into the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note:** You can change the position of the interpolation point after inserting it, by either dragging and dropping it or using the **Properties** context menu.
Target Device Parameters	Use this function to edit the target device parameters.
Representation parameters	
Display Scaling	Use this function to show or hide the scaled cam profile. Specify the scaling via **Cam > Target Device Parameters**. **Note:** While the scaled cam profile is displayed, you will not be able to edit the original curve in the S diagram (path diagram).
Master	Use this function to specify the representation parameters for the master axis.
Slave	Use this function to specify the representation parameters for the slave axis.
V diagram	Use this function to specify the representation parameters for the V diagram (velocity diagram).
A diagram	Use this function to specify the representation parameters for the A diagram (acceleration diagram).
J diagram	Use this function to specify the representation parameters for the J diagram (jerk diagram).
Lines and fonts	Use this function to specify the representation parameters for the lines and fonts in the diagrams.
Simulation Settings	Use this function to specify the simulation settings for the master axis and the slave axis.
Diagrams	
Velocity	Use this function to show or hide the V diagram (velocity diagram).

Function		Meaning/information
	Acceleration	Use this function to show or hide the A diagram (acceleration diagram).
	Jerk	Use this function to show or hide the J diagram (jerk diagram).

3.1.6.3 Fixed point context menu - CamTool

You can select the following functions:

Function	Meaning/information
Delete	Use this function to delete the fixed point from the S diagram (path diagram).
Direct Entry for Acceleration	If you activate this function, you will be able to enter an acceleration value under **a** = on the **Dynamics** tab in the **Fixed point properties** window. If you deactivate **Direct Entry for Acceleration**, the system will calculate the acceleration value. The jerk diagram (J diagram) will then be constant at the fixed point.
Properties	You can use this function to specify the fixed point parameters.

3.1.6.4 Interpolation curve context menu - CamTool

You can select the following functions:

Function	Meaning/information
Optimum Value (for sloping sine curve) (for modifiied sine curve) (for modified acceleration trapezoid) (for harmonic combination)	If you activate this function, the system will calculate the **optimum value** for the interpolation curve's turning point.
Automatic Jerk Characteristic (for a 6th order polynomial)	If you activate this function, the system will calculate the jerk value at the interpolation curve's boundary points.
Properties	You can use this function to specify the interpolation curve parameters.

3.1.6.5 X-axis context menu - CamTool

You can select the following functions:

Function	Meaning/information
Diagram Properties	You can use this function to specify the representation parameters for the diagram.

3.1.6.6 Y-axis context menu - CamTool

You can select the following functions:

Function	Meaning/information
Diagram Properties	You can use this function to specify the representation parameters for the diagram.

3.1.6.7 Cam menu - CamTool

You can select the following functions:

Function			Meaning/note
Close			Use this function to close the active cam or the CamTool cam editor.
Properties			Use this function to display the properties of the active cam.
Import			Use this function to import a cam. The cam must be in the form of a text file.
Export			Use this function to export the active cam to a text file.
Insert			
	Functions		
		Arcsin	Use this function to insert an arc sine curve in the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note**: You can change the position of the arc sine curve after inserting it, by either dragging and dropping it or pulling the handles at the ends of the curve. You can also do this via the **Properties** context menu.
		Sin	Use this function to insert a sine curve in the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note**: You can change the position of the sine curve after inserting it, by either dragging and dropping it or pulling the handles at the ends of the curve. You can also do this via the **Properties** context menu.
	Fixed point		Use this function to insert a fixed point in the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note**: You can change the position of the fixed point after inserting it, by either dragging and dropping it or using the **Properties** context menu.
	Straight line		Use this function to insert a straight line in the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note**: You can change the position of the straight line after inserting it, by either dragging and dropping it or pulling the handles at the ends of the line. You can also do this via the **Properties** context menu.
	Interpolation point		Use this function to insert an interpolation point in the S diagram (path diagram). Place the modified cursor at the position where you want to insert the object and click. **Note**: You can change the position of the interpolation point after inserting it, by either dragging and dropping it or using the **Properties** context menu.
Target device parameters			Use this function to edit the target device parameters.
Representation parameters			

Function		Meaning/note
	Display scaling	Use this function to show or hide the scaled cam profile. Specify the scaling via the **Cam > Target device parameters** menu item. **Note**: While the scaled cam profile is displayed, you cannot edit the original curve in the S diagram (path diagram).
	Master	Use this function to specify the representation parameters for the master axis.
	Slave	Use this function to specify the representation parameters for the slave axis.
	V diagram	Use this function to specify the representation parameters for the V diagram (velocity diagram).
	A diagram	Use this function to specify the representation parameters for the A diagram (acceleration diagram).
	J diagram	Use this function to specify the representation parameters for the J diagram (jerk diagram).
	Lines and fonts	Use this function to specify the representation parameters for the lines and fonts in the diagrams.
Simulation settings		Use this function to specify the simulation settings for the master axis and the slave axis.
Diagrams		
	Velocity	Use this function to show or hide the V diagram (velocity diagram).
	Acceleration	Use this function to show or hide the A diagram (acceleration diagram).
	Jerk	Use this function to show or hide the J diagram (jerk diagram).
Allow interpolation of segments...		Use this function to interpolate all interpolation curves either with CamTool or by the target device. Further information can be found at Optimizing a Cam (Page 114).
Download cam		Use this function to download all the active cam's data (geometry data, scaling and offset values, etc.) to the target device.
Upload cam		Use this function to upload all the active cam's data (geometry data, scaling and offset values, etc.) from the target device to CamTool.

3.2 Getting Started SIMOTION SCOUT - sample project SIMOTION D435-2

Preface

Scope and standards

This **document** is part of the **Engineering System Handling documentation package**.

Scope of validity

This manual is valid for SIMOTION SCOUT, product version V5.1.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product level V5.1:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

3.2.1 Safety notes

3.2.1.1 Fundamental safety instructions

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity..

⚠ WARNING

Danger to life as a result of unsafe operating states resulting from software manipulation

Manipulation of the software, e.g. viruses, trojans, malware or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
- Make sure that you include all installed products in the integrated industrial security concept.
- Protect files on removable storage media against malware through appropriate protective measures, e.g. virus scanners.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

3.2.1.2 Specific safety information

Make sure your training system is fully disconnected from productive operation.

Observe the safety notes in the documentation of the devices used.

3.2.2 Getting Started with SIMOTION SCOUT

3.2.2.1 Aim of Getting Started

Getting Started introduces you to working with the SIMOTION SCOUT engineering system. You will create a simple sample project and in so doing, you will work through the typical steps involved in configuring devices, drives, and axes. You will become familiar with the most important tools that SIMOTION SCOUT provides for configuring, programming, and diagnostics.

3.2.2.2 Sample project

Getting Started provides you with instructions for creating a simple sample project.

Configuring steps

Prepare the configuration

- You reset the SIMOTION device to the factory settings.
- You configure the interface for network communication between the PG/PC and the SIMOTION device.

Create project, configure SIMOTION device and network communication with the PG/PC

- You create a project.
- You create a SIMOTION device and set up the network communication between the PG/PC and the SIMOTION device.

Configure the drive

- You commission the drive.

Configure the infeed

- You interconnect the infeed with the drive.

Configure and test the axis

- You set up an axis.
- You interconnect the axis with the drive.
- You test the axis with the axis control panel.

Configure inputs/outputs

- You configure I/Os for use in the sample program.

Program, set up and monitor SIMOTION

- You write a simple SIMOTION user program that controls the configured axis.
  - You create the variables required by the program.
  - You create the program and additional auxiliary programs with the graphical editors.
- You assign the finished programs to the tasks of the execution system.
- You start program execution in the SIMOTION runtime system.
- You monitor the program-controlled axis movement.
  - You monitor program execution.
  - You monitor the values in the symbol browser.
  - You compile values in a watch table.
  - You record the course of the axis motion with the trace.

Scope of the sample project

The SIMOTION documentation contains two versions of Getting Started with different sample project scopes:

- **Print version of Getting Started:** The present print version deals with configuring a SIMOTION D435-2 device. Auto configuration is used for configuring the drive, and only this is described.
- **Online help version of Getting Started:** The Getting Started you will find in the SIMOTION SCOUT online help is more general in nature. The sample project presented there takes account of the three platforms SIMOTION C, SIMOTION D, and SIMOTION P. The online help version also describes:
  - Configuring the drive with the drive wizard
  - Testing the drive with the drive control panel
  - Configuring a virtual axis. The virtual axis does not require a drive

  You can find the general version of Getting Started in the online help under **Getting Started SIMOTION SCOUT**.

Completed sample project

The completed sample project is included in the SIMOTION Utilities & Applications. You can find it there under **Examples > Getting Started**.

You can find information on the Utilities & Applications in the following section Utilities & applications (Page 145).

Following Getting Started

We recommend that, following Getting Started, you continue to familiarize yourself with SIMOTION SCOUT using the sample projects of the Utilities & Applications.

You can find information on the Utilities & Applications in the following section Utilities & applications (Page 145).

3.2.2.3 Preconditions

Training system

To create the sample project, you require a training system with a few components:

- SIMOTION D435-2 device with the latest firmware V4.5
- PG/PC with free Ethernet interface
  A USB Ethernet adapter is also suitable for the Ethernet connection.
- SINAMICS drive with infeed, power unit, and motor for operating an axis
- Full DRIVE-CLiQ wiring of the components; motor with DRIVE-CLiQ interface and thus with automatic encoder identification (SMI Sensor Module Integrated)
- SIMOTION SCOUT Engineering System

A SIMOTION D435-2 DP/PN device is used in the sample project. However, you can use any D4x5-2 device.

The PROFINET interface of the SIMOTION device is not used in the sample project. In the project, reference is made to PROFINET only when creating the SIMOTION device for the first time, in order to fully represent the work sequence.

Note

Getting Started deals with automatic drive configuration and not configuration using the drive wizard. To be able to carry out automatic drive configuration fully, full DRIVE-CLiQ wiring of the components involved is a necessary requirement.

Preparation of the training system

Your training system has been prepared for configuring with SIMOTION SCOUT:

- The hardware is ready installed and wired.
- The CF card with the latest firmware V4.5 is plugged in.
- The PG/PC and SIMOTION D435-2 are connected direct via Ethernet cable. The Ethernet X127 interface on the SIMOTION D435-2 is used for the connection.
- SIMOTION SCOUT has been installed on the PG/PC and correctly licensed.
- You have started SIMOTION SCOUT. The SIMOTION SCOUT Workbench is visible on the screen of the PG/PC.
- No project is open in SIMOTION SCOUT.

3.2.2.4 General information

SIMOTION SCOUT online help

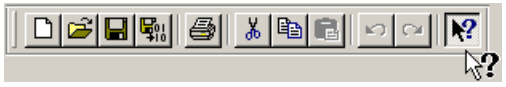
SIMOTION SCOUT has a comprehensive online help. This is divided into general help and context-sensitive help.

You open the general help as follows

- Select **Help > Help topics** in the menu, or
- press the **F1** key.

You open the context-sensitive help as follows

- Click the Help button of the dialog or window, or
- press the key combination **Shift+F1**, or
- click in the toolbar on the **Help** button.



With the mouse pointer (changed to a question mark), click the parameter or the window for which you require help.

You can find detailed information on using the context-sensitive help in the online help under **SIMOTION SCOUT > General > Use Help**.

Full text search in the online help

You can find important information on full text searching in the online help under **Basic > Use Online Help and Function Block Diagrams > Use Online Help > Full Text Search**.

Note

Full text search in the online help

Enclose the search term between asterisks * to include search results that contain the search term as part of a longer character string, e.g.

- **variable** finds only the whole words "variable" or "Variable",
- ***variable*** also finds "system variables", "variable assignment", etc.

Available documentation

Electronic documentation

The SIMOTION documentation is included in electronic form in the scope of delivery of SIMOTION SCOUT (SIMOTION SCOUT DVD Documentation, Utilities & Applications). You can search for all PDF documents in the electronic documentation with an index (SIMOTION.pdx). You can find an overview of the structure and content of the SIMOTION PDF documentation in the separate document *Overview of the SIMOTION Documentation*.

The content of the documents is also available in the SIMOTION SCOUT online help, with a few exceptions.

In the online help, you can find the *Overview of the SIMOTION Documentation* under **Overview of the SIMOTION Documentation > ... > Overview of the SIMOTION Documentation**.

Overview of SIMOTION

You can find a brief system overview of SIMOTION in the online help under **Basic > Basic Functions > System Overview**.

The manuals contain comprehensive information especially for configuring and commissioning a SIMOTION D, see the *SIMOTION D4x5-2 Manual* as well as the *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual*. You can also find the SIMOTION D manuals in the online help under **SIMOTION Devices > SIMOTION D**.

Project generator

With SIMOTION easyProject, basic functions required in just about every SIMOTION application can be quickly and conveniently integrated into an existing project or a new project. The desired components and functions are selected and configured here via dialogs. There is thus no need for time-consuming programming, and a uniform and standardized project configuration is guaranteed while simultaneously reducing potential errors. As of SIMOTION SCOUT V4.4, the project generator can be started direct from SIMOTION SCOUT.

The project generator is not used in Getting Started.

Utilities & applications

The free SIMOTION Utilities & Applications provide you with a wealth of important background information on all aspects of SIMOTION, tools, special functions, blocks, SIMOTION sample projects, as well as off-the-shelf standard applications for illustration or for use in your projects. There you can also find detailed information on scripting and a host of sample scripts that facilitate working with SIMOTION.

1. **FAQs**
   Interesting FAQs such as control of hydraulic axes or communication issues.
2. **Scripts**
   A host of scripts, helpful tips, but also extensive solutions that make recurring tasks easier, for example.
3. **Tools and documentation**
   You are provided with easy-to-use tools and in-depth documentation for many tasks.
4. **Examples**
   Sample projects for first-time-users, e.g. "Getting Started", as well as examples of special topics.

5. **Applications**
   For SIMOTION, there is a host of applications available to you that provide you with a sound basic framework. With the aid of the supplied documentation, you can use the applications as a basis for your own application, and adapt and expand them.
   Furthermore, functions are available to you under "Cross-Sector applications" that are of general help when creating your own applications, e.g. a LDPV1 library for drive communication via DPV1 services (read and write SINAMICS parameters, read errors and warnings from SINAMICS, deactivate objects, ramp-up coordination and much more), or the LCom library with functions for communication with TCP/IP for SIMOTION and SIMATIC.
   The new SIMOTION easyProject project generator significantly speeds up the creation of a standardized project basis for machine applications, and therefore reduces costs.
6. **SIMOTION IT**
   Under SIMOTION IT, you can find innovative examples and tools for bringing control and drive solutions a little closer to the IT world. These include a trace function that is executed via an Internet browser, as well as examples for user-defined Web pages for the SIMOTION IT Web server or for using OPC XML DA.

3.2.3 Prepare the configuration

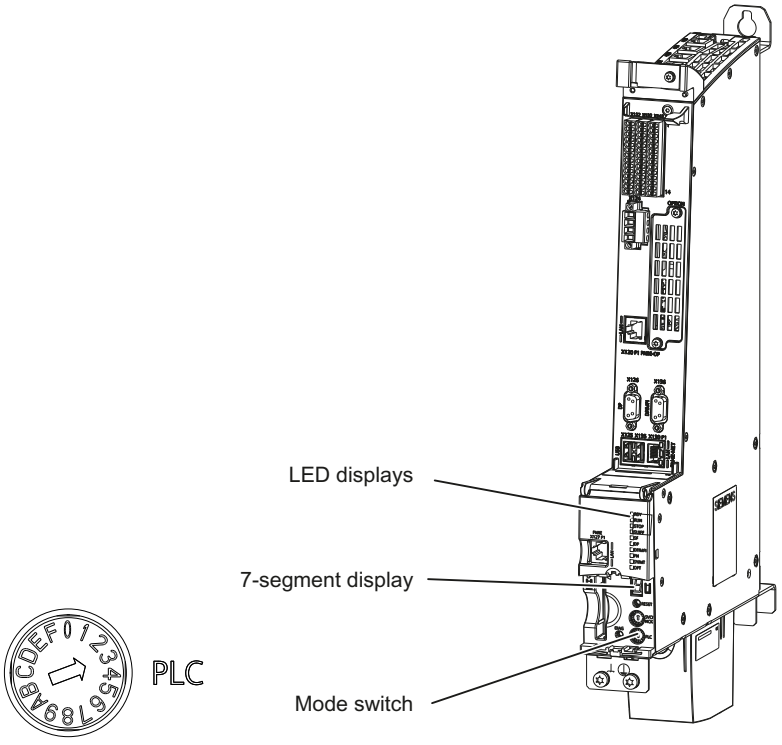
3.2.3.1 Restore factory settings

For the sample project, it is useful to reset the SIMOTION device to the factory settings.

In this way, you restore the default communications parameters and you delete user data installed on the device and the CF card by a previous configuration. The runtime licenses are retained.

You restore the factory settings on the SIMOTION D4x5-2 device as follows

1. Switch off the power supply of the SIMOTION device.
   The SIMOTION device is in a de-energized state when all the LEDs on the front of the device are off.
2. Set the mode selector switch **PLC** of the SIMOTION device to position **3**.



NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Observe the ESD regulations.

3. Switch on the power supply of the SIMOTION device.
   The default settings are loaded. The SIMOTION device switches to STOP mode.
   Wait for the procedure to finish. The elements on the front of the module indicate completion:
   - The 7-segment display shows status digit **6**: SIMOTION D4x5-2 has started up.
   - LEDs:
     LED **RDY** flashes green (0.5 Hz): the drive is ready for commissioning.
     The **STOP** LED shows a yellow light: the SIMOTION device is in STOP mode.
     All other LEDs are off.

   Startup is complete.
4. Set the mode selector switch to position **0**.
   - The **RUN** LED shows a green light: the SIMOTION device is in RUN mode.

Result

The SIMOTION device has been restored to its factory settings and the drive is ready for commissioning.

3.2.3.2 Set up interface for online communication

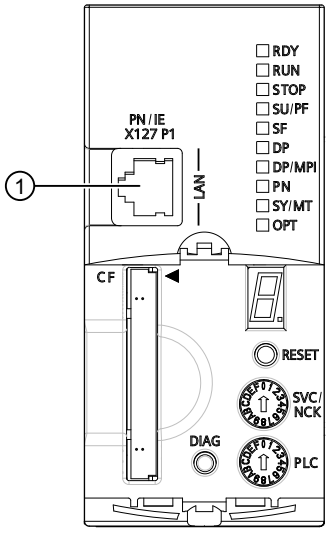
Online communication of the PG/PC with the SIMOTION device can be set up via PROFIBUS, PROFINET, or Industrial Ethernet. The sample project is restricted to the most frequent application case: communication via Industrial Ethernet.

Configuration of the Ethernet interface encompasses two steps:

- You insert the Ethernet address of the PG/PC into the default subnet of the SIMOTION device.
- You assign the Ethernet address to an access point in SIMOTION SCOUT.

Preconditions

- The PG/PC and the SIMOTION device are connected via an Ethernet cable.
- The X127 PN/IE interface on the SIMOTION device is used for the Ethernet connection.



① Ethernet interface X127 PN/IE

Figure 3-42 SIMOTION D435-2 module front

- The Ethernet interface X127 PN/IE has the default address:
  IP address: **169.254.11.22** - Subnet: **255.255.0.0**

Prepare the Ethernet interface of the PG/PC

The PG/PC must be located in the same subnet as the SIMOTION device. The subnet is specified by the factory settings of the SIMOTION device.

You insert the PG/PC into the subnet of the SIMOTION device as follows

1. In the Windows Control Panel of the PG/PC, open the Properties window of the network connection used, and then open the dialog **Properties of Internet protocol Version 4 (TCP/IPv4)**.

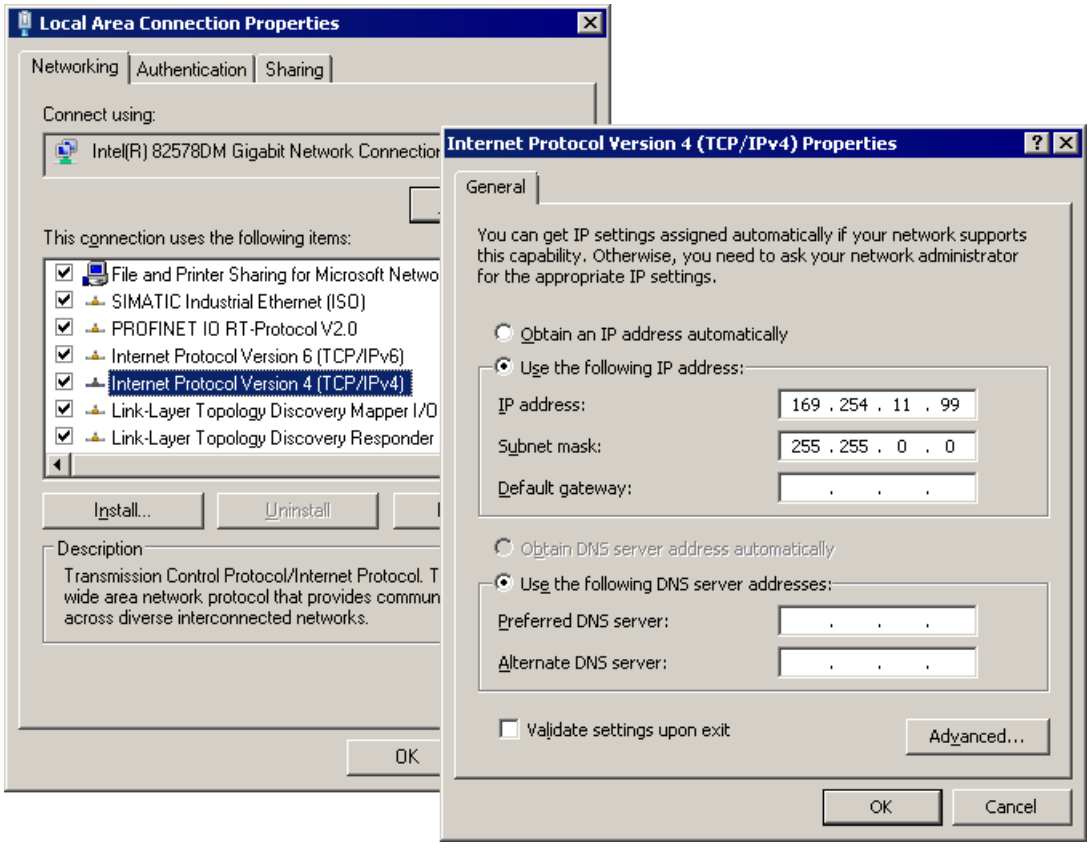


Figure 3-43 IP address of the PG/PC in subnet 169.254

2. Assign the following address:
   IP address: **169.254.11.99** - Subnet: **255.255.0.0**
3. Confirm with **OK**. Close the dialog.

Define the access point of the SIMOTION device

SIMOTION SCOUT has two access points for communication with controllers and individual drives:

- S7ONLINE
- DEVICE

In the sample project, the S7ONLINE access point is used for communication with the PG/PC.

You define the access point as follows

1. Go to SIMOTION SCOUT.
2. Select **Options > Set PG/PC interface** in the menu bar.
   The **Set PG/PC interface** dialog opens.
3. In the field **Access point of the application**, select the access point **S7ONLINE (STEP7)**.
   The interface currently still assigned to the access point is shown after the arrow pointing to the right.
4. Select the prepared Ethernet interface in the list **Interface programming used**.

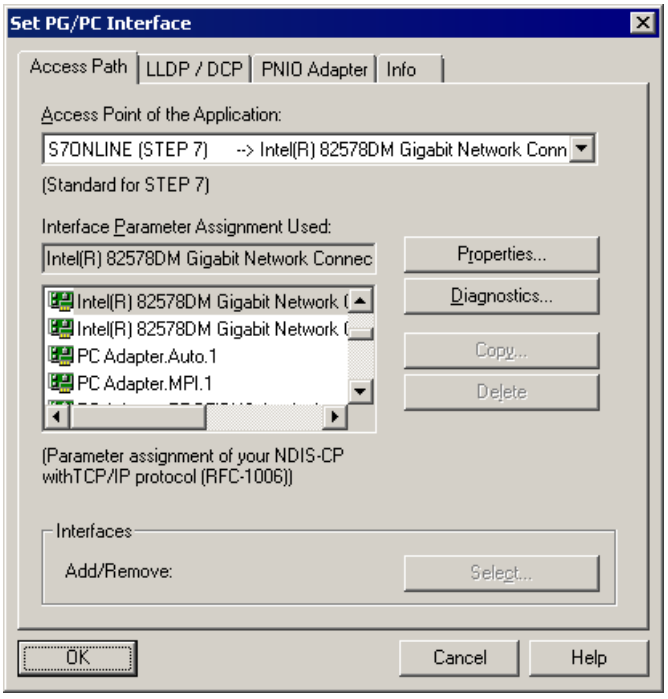


Figure 3-44 Set PG/PC interface

5. Confirm with **OK**.
   The S7ONLINE access point is activated.
   The Ethernet interface is assigned to the S7ONLINE access point.

3.2.3.3 Result in the sample project

The factory settings of the SIMOTION device have been restored.

The Ethernet interface has been prepared and can be used for configuring online communication between the PG/PC and the SIMOTION device.

3.2.4 Create a project

3.2.4.1 Overview

Aim of Getting Started

In this part of Getting Started, you create the sample project Sample_1. All of the subsequent configuring steps refer to this sample project.

Project

A project contains all the information that describes a machine and its function: configuration data, programs, motion profiles, drive data.

A project can contain several SIMOTION devices.

3.2.4.2 Create new project

You create a new project as follows

You start a new configuration in SIMOTION SCOUT by creating a new project.

1. Select **Project > New** from the menu bar. The **New Project** dialog is displayed.
2. Enter the project name under **Name**, e.g. **Sample_1**.
3. Under **Storage location (path)**, enter the path where you wish to store the project. The default path is already set.
4. Acknowledge with **OK**.
   The dialog closes.

Additional information about creating a new project

Default path of the project

The default path depends on the operating system:

- Windows XP
  C:\Program Files\Siemens\Step7\s7proj
- Windows 7
  C:\Program Files (x86)\Siemens\Step7\s7proj

Note

Displaying the project path

When you place the cursor over the project in the Project Navigator, the storage location of your project is displayed.

Project name and project directory name

The name of a SIMOTION SCOUT project can contain not more than 24 characters. The project appears under the full name in the dialogs.

When initially saving the project, SIMOTION SCOUT creates the directory name from the first 8 characters of the project name. SIMOTION SCOUT uses a numerical counter "_1", "_2", ... to resolve conflicting names resulting from the abbreviation of the 8 characters. The counter replaces the last characters of the directory name.

Note

It is useful to select project names in such a way that they can be uniquely distinguished by their first 8 characters. The project name and the directory name derived from it thus uniquely identify the same project.

3.2.4.3 Result in the sample project

The sample project of Getting Started has been created in SIMOTION SCOUT.

The project folder Sample_1 is visible in the project navigator.

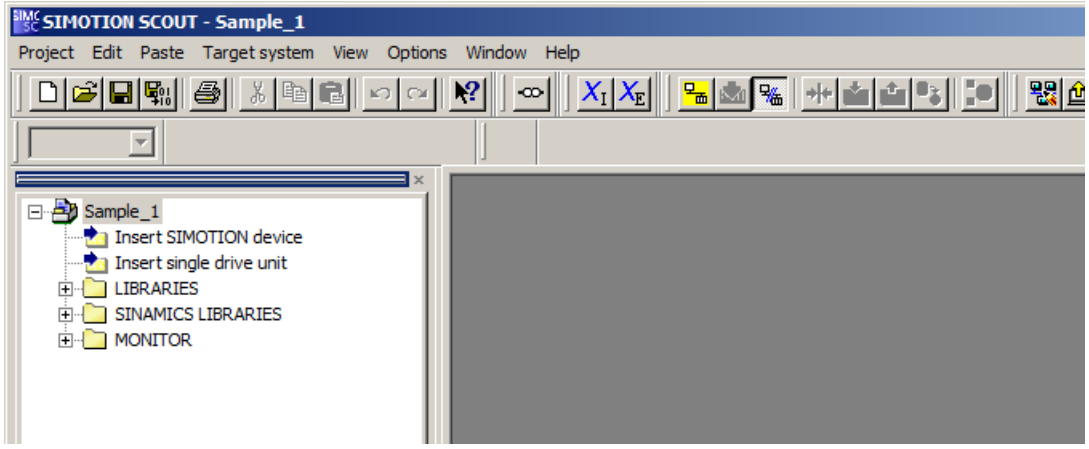


Figure 3-45 SIMOTION SCOUT Workbench, newly created sample project Sample_1

3.2.5 Create SIMOTION device and configure online communication

3.2.5.1 Overview

Aim of Getting Started

This part of Getting Started shows you how to create a SIMOTION device in the project and how to set up communication between the PG/PC and the SIMOTION device.

Steps

Creating a new SIMOTION device in the project involves three steps. SIMOTION SCOUT combines the steps into one coherent process:

1. Create a SIMOTION device.
2. Configure the PROFINET interface of the device.
   This step is omitted in the sample project. PROFINET is not used. However, SIMOTION SCOUT opens the configuration dialog if the PROFINET-supporting device version D4x5-2 DP/PN is installed in your training system.
3. Set up communication between the PG/PC and the SIMOTION device.

The newly created SIMOTION device appears in the project tree.

SIMOTION D platform

SIMOTION D is the drive-based version of the SIMOTION motion control system, based on the SINAMICS S120 family of drives. With SIMOTION D, the SIMOTION motion control functionalities and the SINAMICS drive software run on a SINAMICS-type closed-loop control hardware device. SIMOTION D devices have the following characteristic features:

- Motion control functionality and control functionality integrated directly in the drive
- Compact and with especially fast response
- Maximum scalability and flexibility as single-axis or multi-axis system in different performance versions for diverse applications
- Especially suitable for modular machine concepts with fast isochronous coupling, in the case of machines with a large number of axes, for example

The SINAMICS functionality of the closed-loop-control module of the SINAMICS S120 multi-axis drive system is integrated in SIMOTION D (SINAMICS Integrated).

You will find a brief introduction to SIMOTION D in the online help using the example of the D435 Control Unit. Click **Help > Tutorials > SIMOTION Drive-Based** in the SIMOTION SCOUT menu bar.

3.2.5.2 Create SIMOTION device

You create a SIMOTION D435-2 device in the project as follows

1. In the project navigator, double-click **Insert SIMOTION device**.
   The **Insert SIMOTION device** dialog appears.

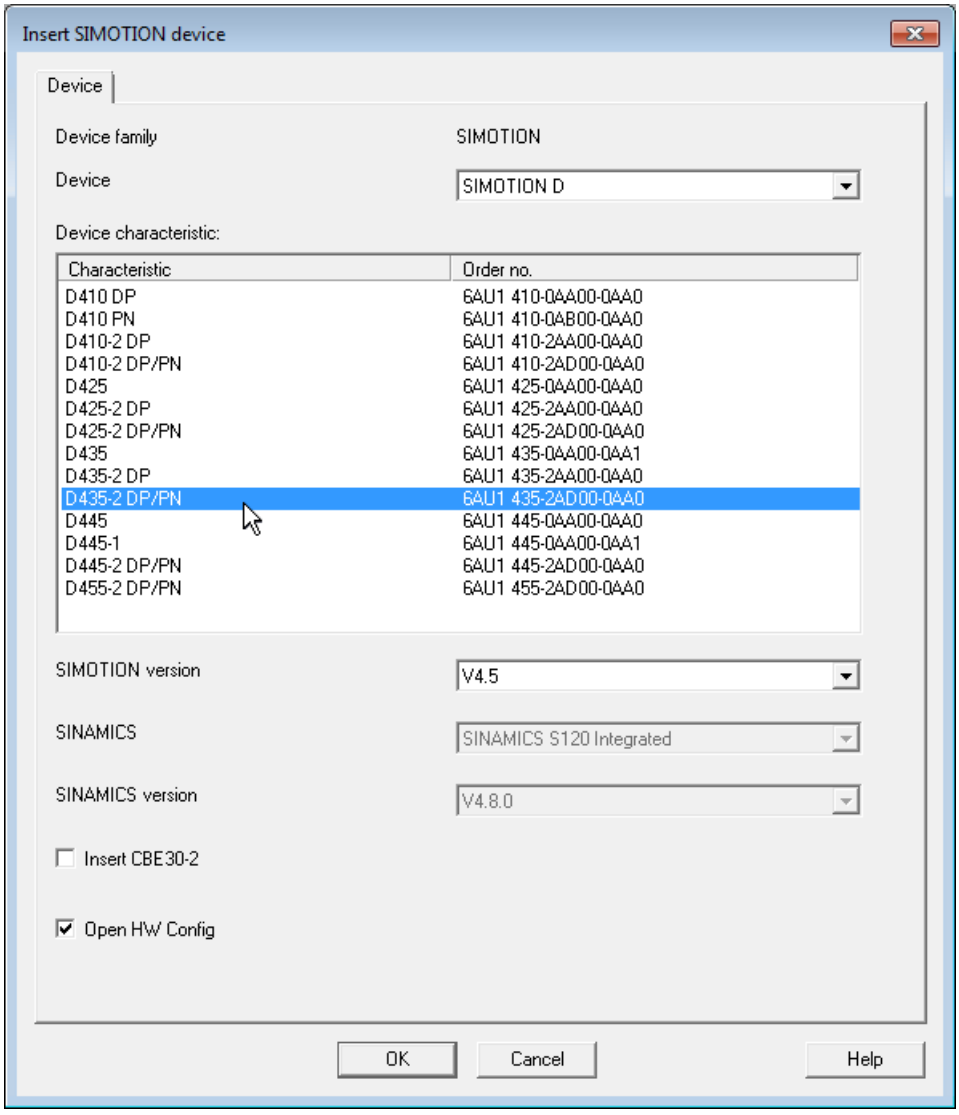


Figure 3-46 Select SIMOTION device

2. In the **Device** list field, the **SIMOTION D** platform is already pre-selected.
3. In the **Device version** list, select the SIMOTION device **D435-2 DP/PN**, and under **SIMOTION version**, select the firmware version of the device used.

Note

The selected firmware version must match the firmware version on the memory card of the SIMOTION device. Otherwise, you will receive an error message when going online with the device.

4. With the checkbox **Open HW Config**, you define whether the window for hardware configuration **HW Config** is to be opened after the new device has been created. Leave the checkbox activated.
5. Acknowledge with **OK**.
   The **Insert SIMOTION device** dialog is closed. SIMOTION SCOUT takes you to the next step Configure the PROFINET interface (Page 156).

3.2.5.3 Configure the PROFINET interface

If the SIMOTION device has a PROFINET IO interface, the dialog **Properties – Ethernet interface PNxIO** appears. The dialog enables integration of the SIMOTION device into an existing PROFINET IO subnet. If no subnet is known, you can create it here.

PROFINET is not used in the sample project. Click **Cancel**. The dialog closes. SIMOTION SCOUT takes you to the next step Set up PG/PC communication (Page 157).

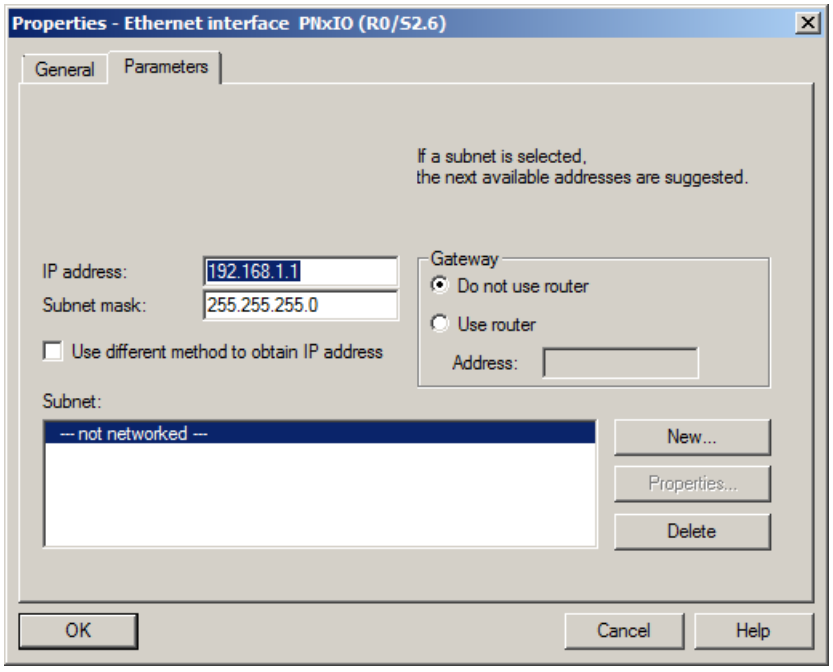


Figure 3-47 Dialog for selecting an Ethernet interface

3.2.5.4 Set up PG/PC communication

Interface Selection dialog

SIMOTION SCOUT opens the **Interface Selection** dialog. You configure the network communication between the PG/PC and the SIMOTION device in this dialog.

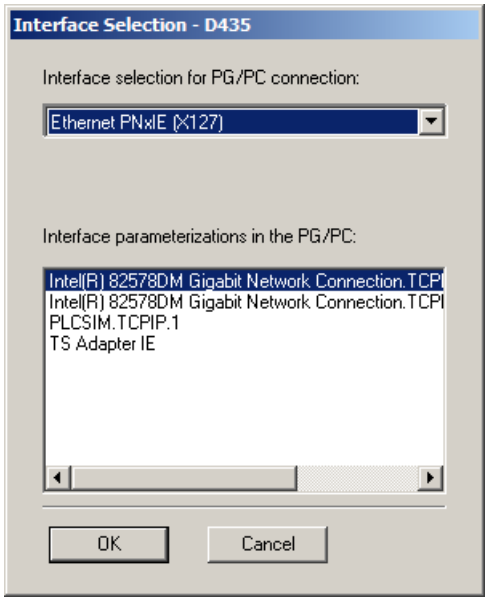


Figure 3-48 Set up PG/PC communication

Note

If a PG/PC is already available in the project and it is connected with a subnet, SIMOTION SCOUT automatically establishes the connection between the PG/PC and the SIMOTION device. In this case, the **Interface Selection** dialog does not appear.

The list field **Interface selection for PG/PC connection** in the upper half of the dialog provides the list of SIMOTION device interfaces for selection.

The lower field **Interface parameterizations in the PG/PC** lists the interfaces of the PG/PC. The upper field functions as a filter so that only the interfaces with suitable transmission protocol are available for selection.

You set up data communication between the SIMOTION device and the PG/PC as follows

1. Select the Ethernet interface **Ethernet PNxIE (X127)** for the SIMOTION D435-2 device.
2. Then select the prepared Ethernet interface of the PG/PC.
3. Confirm the configuration with **OK**.
   The **Interface Selection** dialog is closed.

3.2.5.5 Result in the sample project

Newly created SIMOTION device

Configuring of the SIMOTION device and network communication between the PG/PC and the SIMOTION device is complete.

- The newly created SIMOTION D435-2 device is shown in the project tree of the workbench.
- The PG/PC is connected with the SIMOTION device via Ethernet.
- SIMOTION SCOUT automatically opens the **HW Config** window and thus leads you to hardware configuration.

HW Config

HW Config shows the newly created SIMOTION device, the integrated SINAMICS drive unit (SINAMICS_Integrated), and the integrated PROFIBUS (DP-Integrated).

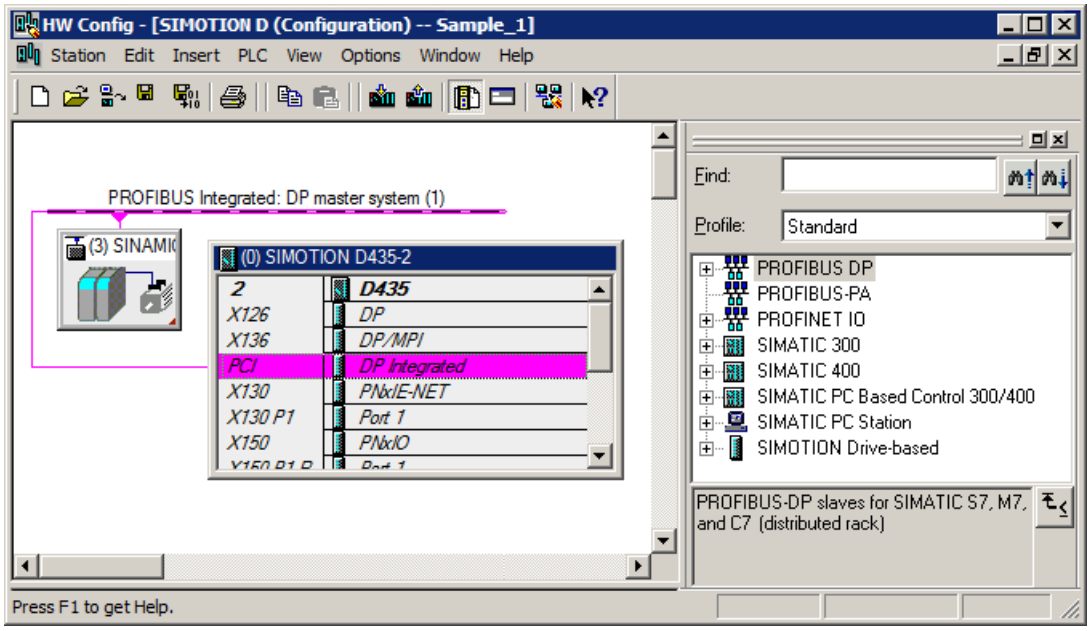


Figure 3-49 HW Config, SIMOTION D device with SINAMICS_Integrated

No other hardware configuration is required for the sample project. You can close HW Config. In the menu bar of **HW Config**, select the command **Station > Exit**.

3.2.6 Download the project to the target system

3.2.6.1 Overview

Aim of Getting Started

In this configuring step, you create the prerequisites for configuring the drive.

- You back up the created sample project to the hard disk.
- You compile the project into executable code.
- You establish online communication with the SIMOTION device.
- You download the project from the PG/PC to the SIMOTION device.

3.2.6.2 Save and compile the project

To be able to download a project created in SIMOTION SCOUT to the target system, the project must be saved in executable code.

The command **Save project and compile changes** combines both steps. The project is backed up to the hard disk. SIMOTION SCOUT searches the entire project for changes and compiles only the changes.

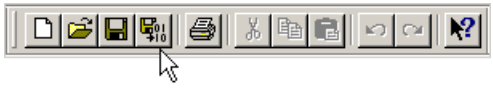
Note

Use the command **Save project and compile changes** for preference for day-to-day work.

Several variations of the "Save" command are available for selection under the menu title **Project**. You can find information on this in the online help under **Save and compile**.

You save and compile a project as follows

Select the menu command **Project > Save and compile changes** or click the relevant button in the toolbar.



SIMOTION SCOUT shows progress indicators for symbolic assignment and for compilation.

The compilation run is logged in the detail view of the workbench. Information, warning and compilation errors are shown there in plain text.

Switch on detail view

The detail view might be switched off. Click the menu item **View > Detail view** to activate the view.

3.2.6.3 Connect to selected target devices – Go online

To download project data from SIMOTION SCOUT to the hardware, or to transfer machine data in the other direction from the hardware to the SIMOTION SCOUT project, communication between the PG/PC and the SIMOTION device must be activated.

The status of the network communication is displayed in the footer of the workbench:

Online mode	Network communication is switched on. Data can be exchanged.
Offline mode	The network connection is switched off.

You go online as follows

1. Select the menu command **Project > Connect to selected target devices** or click the relevant button in the toolbar.

When the command is first called, SCOUT opens the **Target Device Selection** dialog. The dialog enables individual selection of the devices to which SIMOTION SCOUT is to connect.

2. In the dialog, select the configured SIMOTION device **D435** and the integrated drive **SINAMICS_Integrated**.

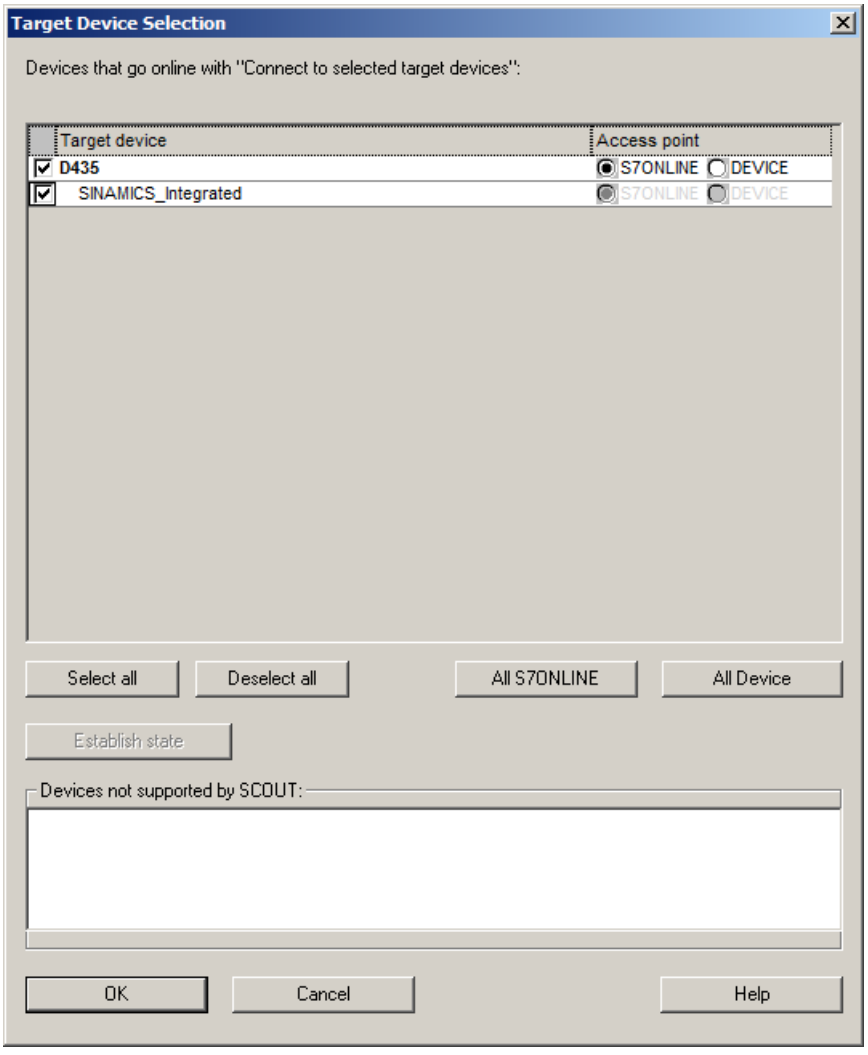
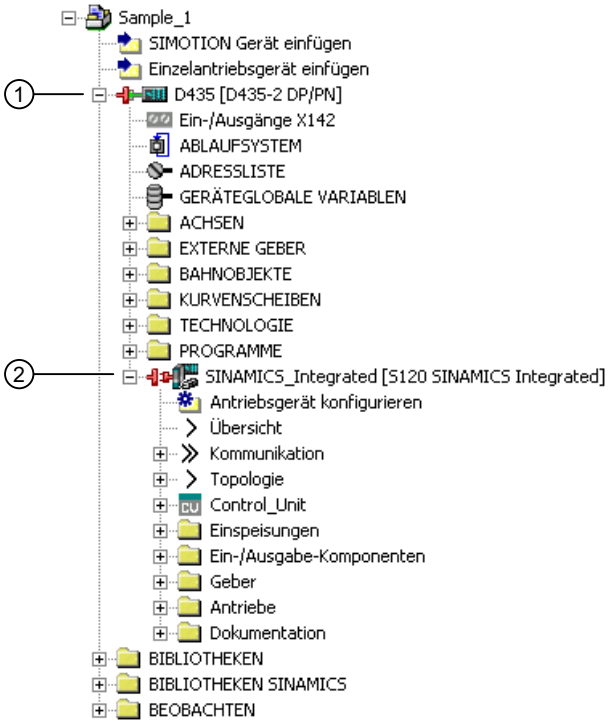


Figure 3-50 Target Device Selection dialog

3. Click **OK**.
   SIMOTION SCOUT establishes the online connection.
4. When first going online, SIMOTION SCOUT reports that access to the drive is not possible. Close the message. Track the consistency check to completion on the **Target system output** tab in the detail area.
   In the project tree, the connector symbols on the SIMOTION device and on the integrated SINAMICS drive change. Only the connector symbol of the SIMOTION device is partially colored in green. Green indicates the existing online connection.



① Device is in ONLINE mode. There is an inconsistency between the project data of the device in the PG/PC and the current values of the device in the target system.

② Element is not in ONLINE mode.

Figure 3-51 Project tree after first going online

No connection to the drive when first going online

To allow SIMOTION SCOUT to connect with the drive, the project must be downloaded to the SIMOTION device. A project download has not yet been carried out in the sample project. You therefore receive the corresponding error messages.

3.2.6.4 Download the project to the target system

A complete project download is only possible in STOP mode. If required, SIMOTION SCOUT offers a change of operating mode during the download procedure.

You download the project as follows

1. Select **Project > Download to target system** in the menu, or click the **Download project to target system** button in the toolbar.

   The dialog **Download to target system** appears.

2. Activate the checkbox **After loading, copy RAM to ROM**. This saves the RAM of the SIMOTION device to the memory card (ROM) of the SIMOTION device. In this way, the configuration is retained after the power supply has been switched off and on again.

3. Start the download operation with **Yes**.
   The project data and the data of the hardware configuration are downloaded to the RAM of the target system.
   If you are asked whether the CPU is to be switched to STOP, confirm with **Yes**.
   SIMOTION SCOUT carries out numerous checks that are logged on the **Target system output** tab in the detail area. You will find there the concluding entry **Download to target system completed successfully**.

Note

The first download to the target system also downloads the data of the technology package. This operation can take several minutes.

Note

Depending on the firmware version on the CF card and on the SINAMICS components (DRIVE-CLiQ components such as Line Module, Motor Modules, Terminal Modules, etc.), the firmware of the components is automatically upgraded or downgraded.

The update can take several minutes and its progress is tracked by corresponding messages appearing in the output window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is signaled by red-green flashing of the RDY LED:

- FW update in progress: RDY LED flashes slowly (0.5 Hz)
- FW update finished: RDY LED flashes quickly (2 Hz), POWER ON required

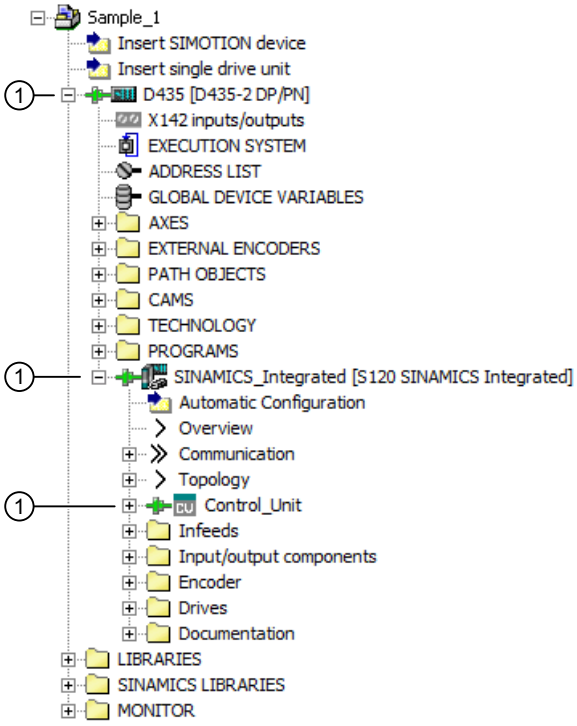
These flashing patterns are displayed additionally via a yellow RDY LED on the SIMOTION D, indicating that components connected to SIMOTION D are carrying out a firmware update or that all components have completed their firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SIMOTION SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

Automatically established connection to the drive

SIMOTION SCOUT automatically establishes the online connection to the drive immediately following the project download.

In the project tree, the connector symbol changes on the SIMOTION device, the integrated SINAMICS drive, and the SINAMICS Control Unit. The connector symbol is entirely green, indicating that the project data in the PG/PC is identical with the project data in the target system.



① Green connector symbol: Element is in online mode. The project data in the PG/PC is identical with the project data saved in the target system.

Figure 3-52 Project tree

From this point, you can access the drive online with the PG/PC.

3.2.7 Configure the drive

3.2.7.1 Overview

Aim of Getting Started

In this part of Getting Started, you configure the integrated drive of the SIMOTION D435-2 device. You use auto-configuration for this purpose.

Preconditions

- You have downloaded the sample project to the target system, refer to the section Download the project to the target system (Page 159).
- SIMOTION SCOUT is online with the SIMOTION device and the integrated drive (green connector symbols).
- The SINAMICS drive components Infeed and Power unit, as well as Motor and Encoder, are connected to the SIMOTION Control Unit via DRIVE-CLiQ. This requirement for auto-configuration was referred to at the start of Getting Started, see the section Preconditions (Page 143).

Drive

The speed and current control functions for controlling the motor are implemented in the drive.

Automatic drive configuration

SIMOTION SCOUT can read out the electronic type plates of the SINAMICS drive components via the DRIVE-CLiQ interface and can use this data to configure the drive automatically. The data does not therefore need to be entered manually. The sample configuration of a SIMOTION D435-2 device shown here requires full DRIVE-CLiQ wiring.

As an alternative to automatic drive configuration, you can also configure a D4x5-2 device offline. Offline configuring is demonstrated in the online help version of Getting Started for SIMOTION C and SIMOTION P.

3.2.7.2 Automatic configuration of the drive

You open automatic configuration as follows

Double-click on **Automatic Configuration** in the project navigator under the drive SINAMICS_Integrated.

The **Automatic Configuration** dialog is displayed.

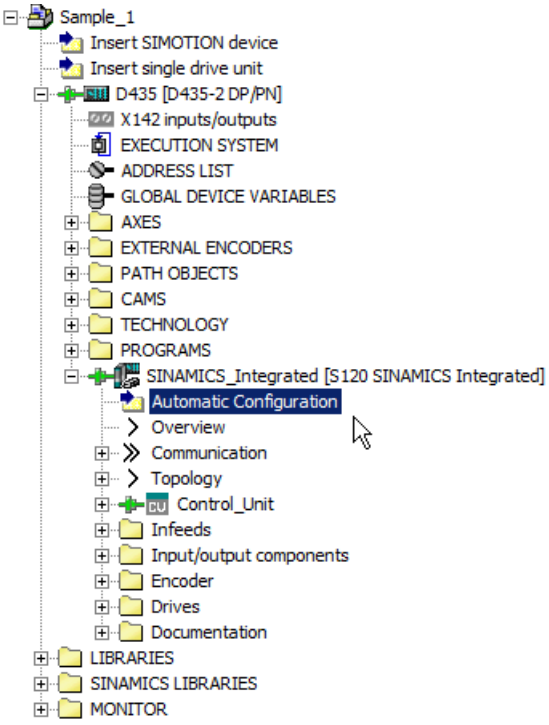


Figure 3-53 Call Project tree, Automatic Configuration

Note

In the project tree, the element **Automatic Configuration** is only visible if you have downloaded the project to the target system, and SIMOTION SCOUT is online with the SIMOTION device. The project must be consistent; see the section Download the project to the target system (Page 159).

You cause the drive to be configured automatically as follows

1. Click the **Configure** button in the **Automatic Configuration** dialog.
2. Confirm the prompt regarding restoring the factory settings with **Yes**.
   The confirmation prompt appears if the drive unit is not in the "First commissioning" state.

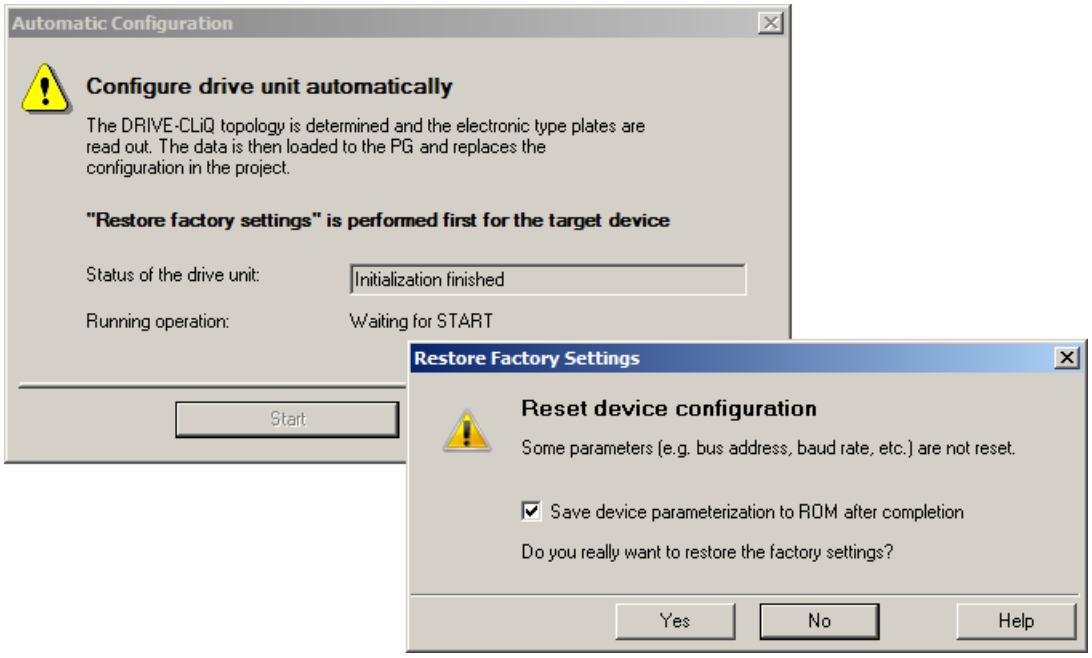


Figure 3-54 Start automatic drive configuration and restore factory settings

3. In the **Automatic Commissioning** dialog, you can specify whether you are using a drive object of the type servo or vector.
   Select **Servo**.
4. Click the **Create** button in the **Automatic Commissioning** dialog.
   Automatic configuring is started.

Note

Firmware update

If the firmware version on the DRIVE-CLiQ components is different to that on the CF card, a firmware update is performed automatically at this position.

In this case, proceed as follows:

- Wait for the procedure to finish. This can take several minutes.
- Go offline.
- Switch the power supply to the SIMOTION device off and then on again.

---

5. If a firmware update has not been performed, remain online to be able to see the changes in the project tree.

6. Open the **Infeeds** system folder in the project tree under the integrated drive.
   - If you are using an infeed with DRIVE-CLiQ interface, the auto configuration has created the infeed there.
   - If you are using an infeed without DRIVE-CLiQ interface, the folder is empty. The infeed is not known in the project.
     The other configuring requirements resulting in this way are dealt with in the next section Configure the infeed (Page 169).
7. Open the **Drives** system folder in the project tree under the integrated drive. The folder contains the drive detected by the auto configuration.

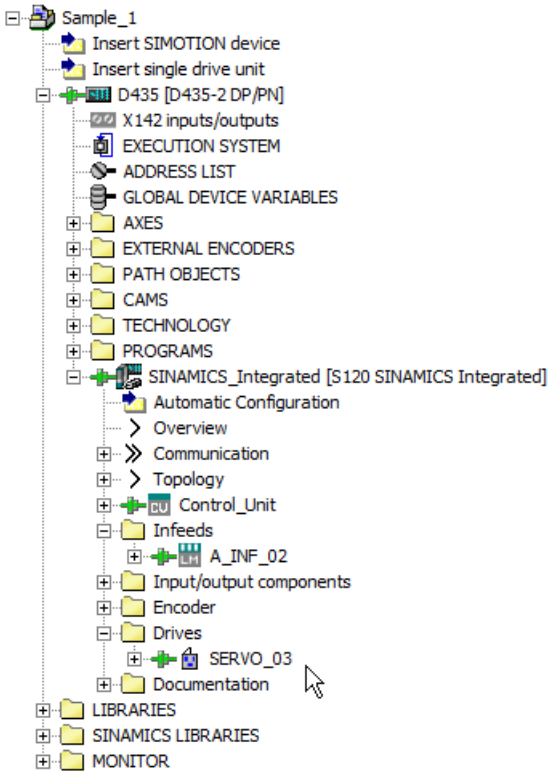


Figure 3-55 Project navigator, configured infeed and drives

8. Go offline.
   Click in the toolbar on the **Disconnect from target system** button.

In the footer of the workbench, **Offline mode** is indicated. Network communication between the PG/PC and the SIMOTION device has been cleared down.

Change the name of the drive

The automatic configuration also assigns the object name of the drive; in this example, SERVO_03.

You can change the name later. Changing is only possible in offline mode.

Proceed as follows:

1. Open the context menu of the drive by right-clicking. Select the **Rename** command there.
2. Assign the new name. Then confirm with **OK**.

As with all changes carried out in the project offline, the name change makes the project inconsistent. To restore the consistency between the "Project on the PG/PC" and the "Project on the SIMOTION device", another project download is necessary. The procedure is described in the section Download the project to the target system (Page 159).

3.2.7.3 Result in the sample project

If you are using an infeed with DRIVE-CLiQ interface, the drive is ready for operation. It can be interconnected with an axis.

3.2.8 Configure the infeed

3.2.8.1 Overview

Aim of Getting Started

In this part of Getting Started, you configure the infeed.

The integrated SINAMICS Control Unit only starts the drive when the infeed is ready. The project must therefore know the interface via which the drive receives the ready signal of the infeed.

Two cases must be distinguished here:

- Infeed with DRIVE-CLiQ interface
  If an infeed with DRIVE-CLiQ connection has already been created, the ready signal of the infeed (r0863.0) is automatically interconnected with "Infeed operation, p0864" of the drive when drives are inserted (only applies to drives that are attached to the same drive unit as the infeed).
  You can continue immediately with the next configuring step, see the section Configure the axis (Page 171).
- Infeed without DRIVE-CLiQ interface
  If you are using an infeed without a DRIVE-CLiQ interface, e.g. a Smart Line Module, you must wire the ready signal of the infeed via terminals. The following section describes the procedure.

Preconditions

- You have configured the integrated drive of the SIMOTION D435-2 device, see the section Configure the drive (Page 164).
- SIMOTION SCOUT is in offline mode.

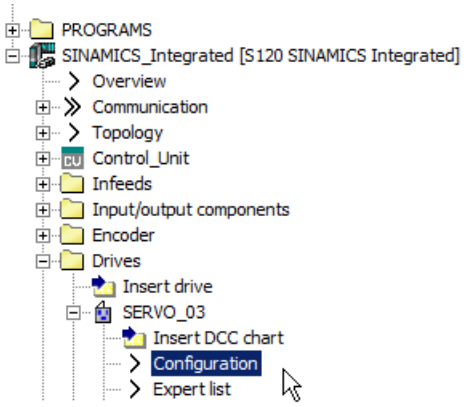
3.2.8.2 Configuring an infeed without DRIVE-CLiQ interface

An infeed without DRIVE-CLiQ interface provides the ready signal (p0863.0) via an output terminal. In the project, you specify the input (r0722) of the integrated SINAMICS Control Unit at which the signal is active. The drive supplied by the infeed uses the signal as a ready signal (p0864).

You interconnect the ready signal of the infeed as follows

In the sample project, the ready signal of the infeed (terminal "DO: Ready" of the infeed) is wired to the DI 0 of the D435-2.

1. Open the configuration dialog of the drive. To do so, double-click **Configuration** in the project tree below the drive.



2. Click in the working area on the **Configure DDS** button with the yellow background.

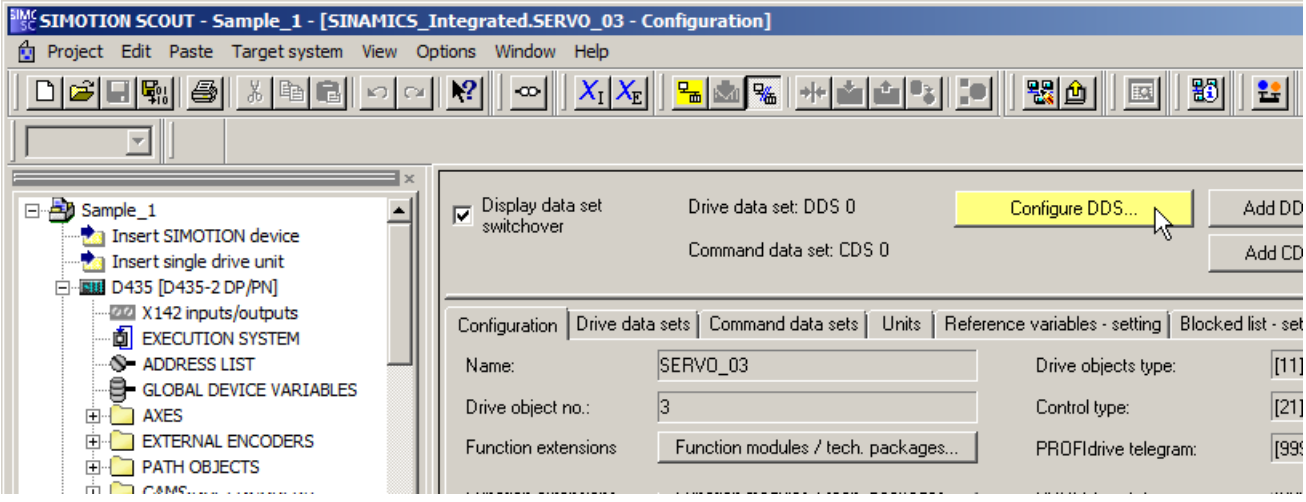


Figure 3-56 Configure DDS

The drive wizard opens.

3. Click **Next** in the drive wizard until you reach the dialog **Configuration - SINAMICS_Integrated - Power Unit BICO Technology**.

4. In the **Power unit BICO** dialog, input field **p0864**, select the digital input to which the ready signal of the infeed is wired (e.g. DI 0).

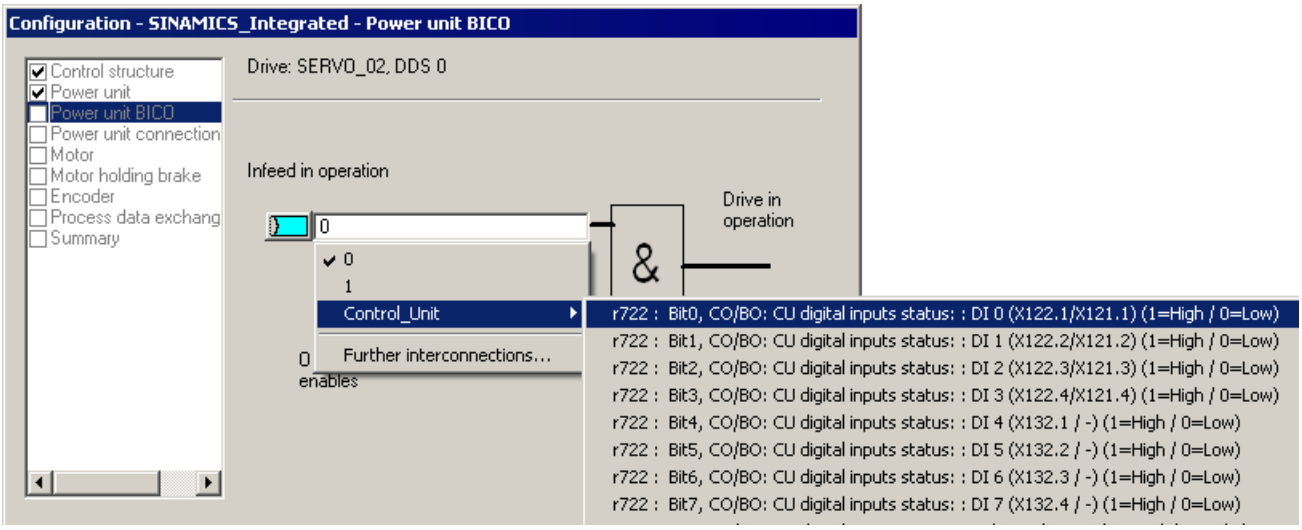


Figure 3-57 BICO interconnection in the drive wizard

5. Click **Next**. Run through all the other dialogs without change until the final **Summary** dialog.
6. Click **Finish**.
   The configuration is thus completed.
7. The configuration dialog of the drive is no longer required. Click **Close** at the bottom right of the dialog box.

3.2.9 Configure the axis

3.2.9.1 Overview

Aim of Getting Started

This part of Getting Started shows you how to create and configure an axis in the project, and how to interconnect it with the integrated drive of the SIMOTION D device. SIMOTION SCOUT provides the axis wizard for this purpose.

Preconditions

You have configured the integrated drive of the SIMOTION D device, see the section Configure the drive (Page 164).

Axis technology object

Technology objects represent the respective real objects (e.g. a position axis) in the controller.

The Axis technology object offers users a technological view of the drive and the encoder (actuator and sensor), provides technological functions for these components, and conceals the actual hardware interface.

The Axis technology object contains extensive functionality, e.g. communication with the drive, actual value processing, position control, and positioning functionality. It executes control and motion commands and indicates states and actual values.

Technological limitations and mechanical values of the axis and encoder (e.g. leadscrew pitch and gears) are set on the axis. You can then work exclusively with technological variables.

Axis wizard

SIMOTION SCOUT provides the axis wizard for creating a new axis. The wizard scans the basic settings and interconnects the TO axis with a drive.

The wizard can only be run through once. Later changes to the configuration can be entered in the corresponding dialogs of the TO.

3.2.9.2 Creating an axis

For the sample project, create the axis **Axis_2**. Assign the axis to the drive you have configured in the section Configure the drive (Page 164).

Run through the wizard and confirm all standard settings with **Next**.

Note

Axis_2

In the online help version of Getting Started, 2 axes are created, a virtual axis Axis_1 and a real axis Axis_2. In agreement with the designation there for the real axis, the designation Axis_2 is used here.

You can find Getting Started in the online help under **Getting Started SIMOTION SCOUT**.

You create a real axis in the project as follows

1. Open the **AXES** folder in the project navigator.
2. Double-click **Insert axis**. The **Insert axis** dialog appears.
3. Name the axis in the **Name** field. Use the designation **Axis_2** for the axis of the sample project.

4. Leave the preset axis technology at the default **Positioning**.

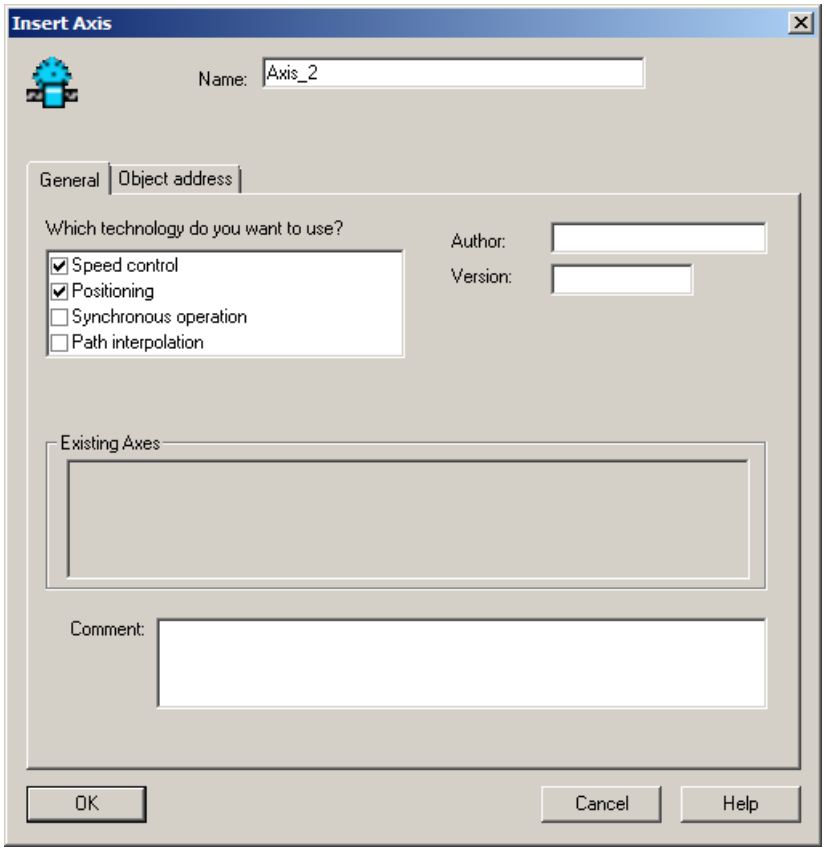


Figure 3-58 Axis wizard, create real axis

5. Click **OK**.
The axis configuration wizard appears.

6. Define the axis type.
   For the sample project, select preferably a linear axis with an interconnection to an electrical drive.
   Activate the fields **linear** and **electrical**.

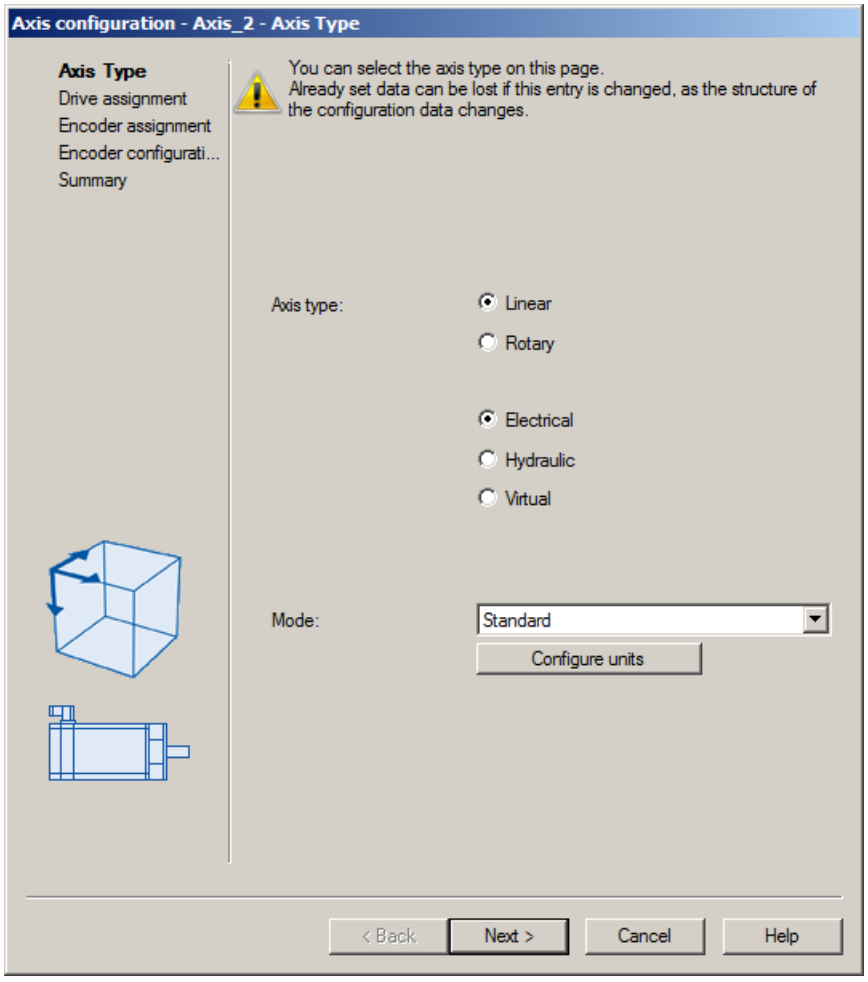


Figure 3-59 Axis wizard, determine axis type

7. Click **Next**.
   The **Drive assignment** dialog appears.

8. Assign the axis to the drive you have configured in the section Configure the drive (Page 164). To do so, click the drive unit in the **Assignment partner** column. Click in the tree under the drive object on the drive object you want to interconnect. **Assign** appears in the **Assignment** column. The axis/drive assignment is thus defined.

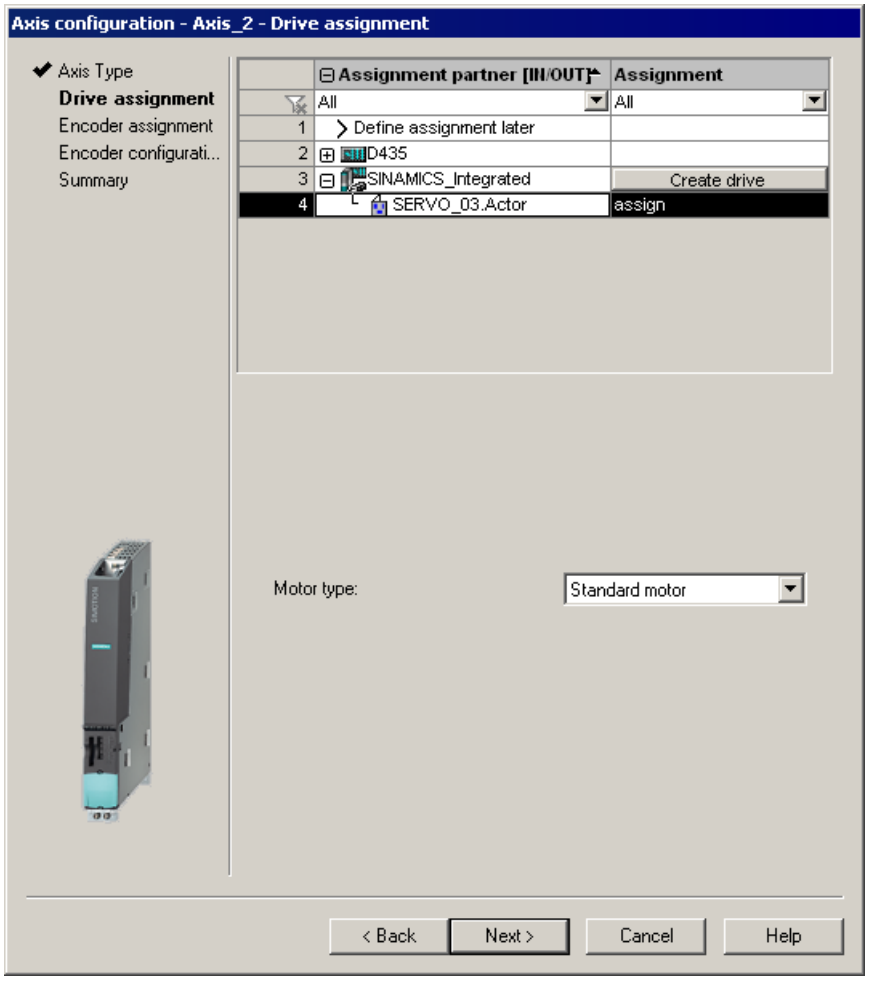


Figure 3-60 Axis wizard, assign drive

Click **Next**.

Note

As an alternative to assigning the axis to an already configured drive, the axis wizard offers two further selection options:

- Define the axis/drive assignment later:
  The axis is to be created and not assigned to a drive until later. Programming and simulation of the axis are also possible here.
- Create drive:
  The drive wizard can can be called up from the axis wizard (offline configuring). The axis can thus be created in one step along with the drive, and assigned to the drive.

The alternative procedures are not considered further in Getting Started.

9. Assign the motor encoder.
   In the **Encoder assignment** window, you use the motor encoder connected to the drive as the standard setting.
   Click **Next**.
10. The axis wizard then assembles the configuration data in an overview. You thus have an opportunity to check and correct the data before accepting it into the project.
    Close the dialog with **Finish**.
    The configured axis is displayed in the project navigator.

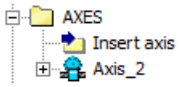


Figure 3-61 Created axis in the project navigator

Automatic settings of the engineering system

The engineering system automatically defines the PROFIdrive axis telegrams required for communication, as well as the addresses used.

In the same way, telegrams are extended and interconnections automatically created in the drive, depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

Drive and encoder data, as well as reference variables, maximum variables, torque limits, and granularity in torque reduction of the SINAMICS S120 are accepted automatically for the configuration of the SIMOTION technology objects "TO axis" and "TO external encoder". This data no longer has to be entered in SIMOTION.

Further axis configuration

Further axis configuration is possible via dialogs that can be accessed in the project navigator under the axis. No other configuration is required for the sample project.

Download the axis configuration to the target system

- Save and compile the project.
- Download the sample project with the axis configuration to the target system to be able to test the functioning of the axis in the next configuration step.

See also

Download the project to the target system (Page 159)

3.2.10 Test the axis with the axis control panel

3.2.10.1 Overview

Aim of Getting Started

In this part of Getting Started, you test the configured axis. SIMOTION SCOUT provides you with the axis control panel for this purpose.

Preconditions

- You have configured the infeed, see the section Configure the infeed (Page 169).
- You have created and fully configured an axis in the sample project, see the section Configure the axis (Page 171).
- The project with the axis configuration has been downloaded to the target system, see the section Download the project to the target system (Page 159).
- SIMOTION SCOUT is in online mode.

Axis control panel

The axis control panel is used for controlling and monitoring individual axes. You can use it to traverse axes together with the drive.

You can use the control panel to perform the following tasks, for example:

- Test each part of the system individually before program-driven axis motions are initiated.
- In the event of an error, test whether the individual axes and drives can be traversed from the control panel.
- Move the axes for optimization purposes (controller optimization).
- Carry out homing.
- Set and remove an axis enable.

3.2.10.2 Working with the axis control panel

In the sample project, you traverse the set-up axis in jog mode to test the correct functioning of the axis.

You open the axis control panel as follows

Open the **AXES** folder in the project navigator. Double-click below the axis on **Control panel**.

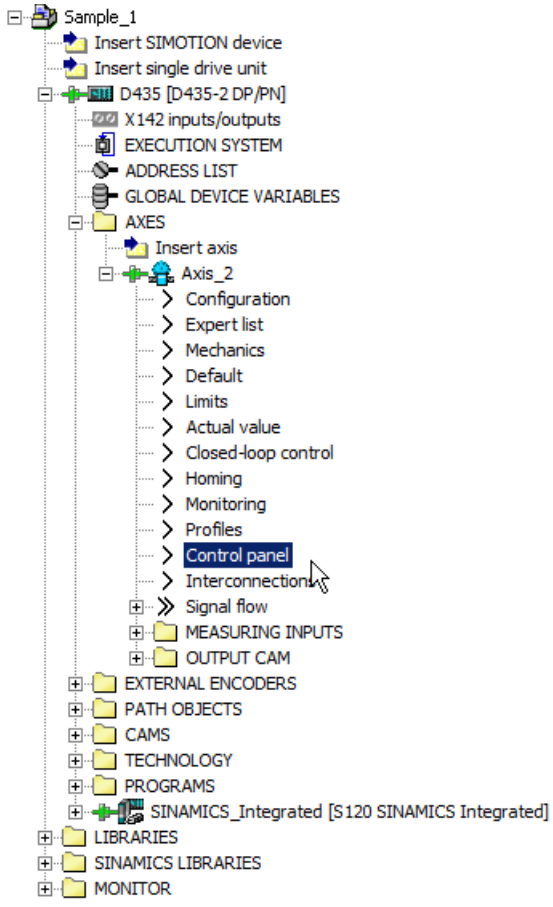
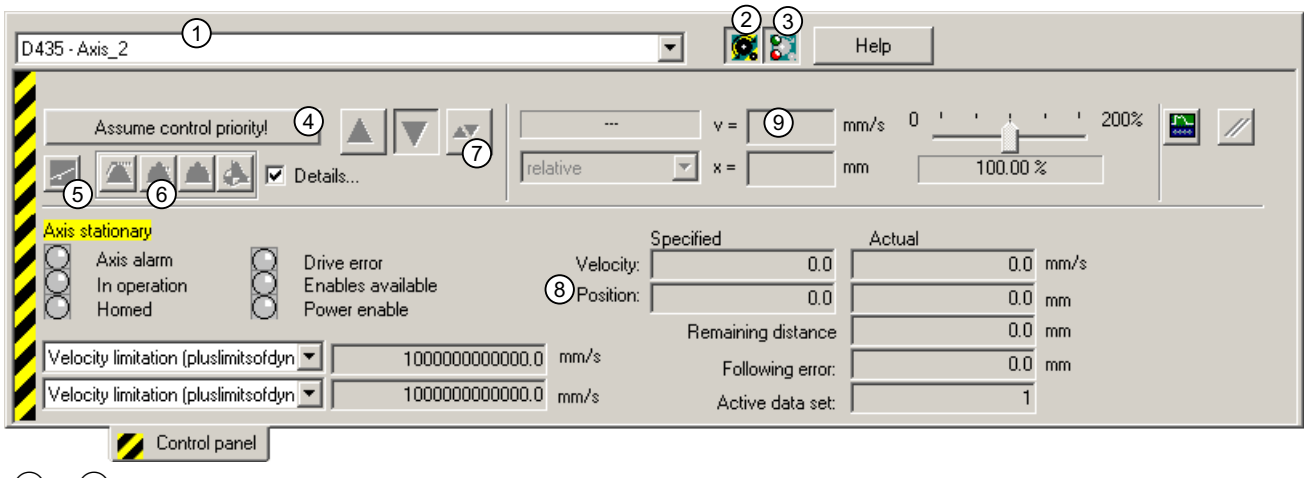


Figure 3-62 Calling the axis control panel

The axis control panel is opened in the detail view.



① ... ⑨ Reference is made to the circled digits in the text below.

Figure 3-63 Axis control panel

You traverse an axis with the axis control panel as follows

1. The field at top left of the control panel ① shows the currently selected axis.
2. Showing areas in the control panel:
   The control panel is divided into a control area and a diagnostics area. The areas may be hidden.
   Click the **Show/hide control area** button ② and/or the **Show/hide diagnostics area** button ③ to show the respective area.
3. Assume control priority – Observe safety regulations:
   To be able to traverse the axis from the PG/PC, the PG/PC must obtain control priority.
   Click the **Assume Control Priority** button ④.
   The **Assume Control Priority** dialog opens.

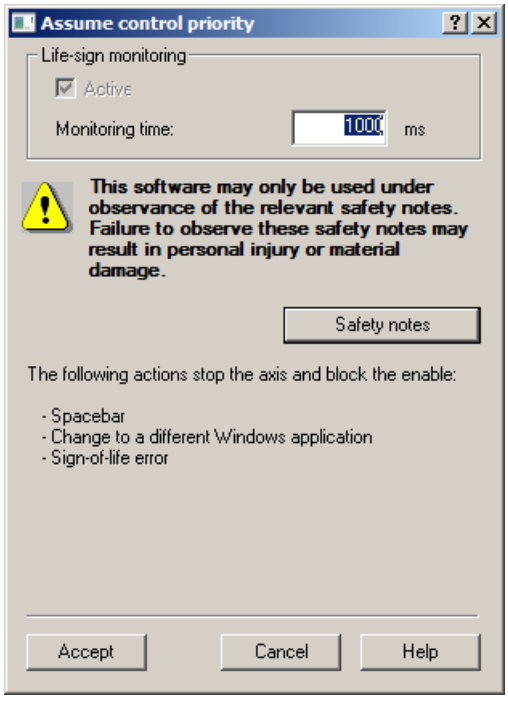


Figure 3-64 Axis control panel, assume control priority

Observe the safety regulations and confirm with **Accept**.
The PG/PC now has control priority.

Note

You can stop the axis at any time by pressing the spacebar.

4. Enable the axis:
   Click the **Set/remove enables** button ⑤ to enable the axis.

Confirm the **Switch axis enable** dialog that appears with **OK**.
The switches **Start motion**, **Stop motion** and **Jog** ⑦ are enabled.

5. Homing the axis:
   Click the **Home axis** button.
   In the dialog box, select the **Homing type Set home position**, and enter the value **0** under **Home position coordinates**.
   Confirm the dialog box with **OK** and start homing the axis.
6. Click the **Position-controlled traversing of the axis** button ⑥.

   The **Start axis position-controlled** dialog appears.
   Specify the velocity of the axis. Click **OK** to close the dialog.
   The velocity appears in the field **v=**⑨.
7. Start the axis motion:
   Click **Jog**⑦.

   The axis is traversed while you press the switch. In the fields **Velocity** and **Position**⑧, you can monitor the traverse motion.
   The axis test is thus executed.
8. Deactivate axis enable:
   Click **Set/remove enables**⑤.
   Confirm the **Disable axis** dialog with **OK**.

   Note

   If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

9. Return control priority:
   Click **Give up control priority**④ to deactivate control of the axes from the PG/PC. In this operating state, the axes can no longer be controlled from the PG/PC.
10. Go offline:
    Select **Project > Disconnect from target system** in the menu. Or click in the toolbar on the **Disconnect from target system** button.

3.2.10.3 Result in the sample project

You have traversed the axis of the sample project with the axis control panel and thus ensured its correct functioning. Configuring the axis is thus completed.

3.2.11 Configure digital outputs

Aim of Getting Started

In this part of Getting Started, you configure two of the digital inputs/outputs available on the SIMOTION D device as outputs.

You require the outputs for the program you will write in the subsequent configuring step, see the section Programming the SIMOTION application (Page 182).

Preconditions

SIMOTION SCOUT is in offline mode.

I/O channels of the terminal X142

You use the I/O channels of the terminal X142 for the sample project.

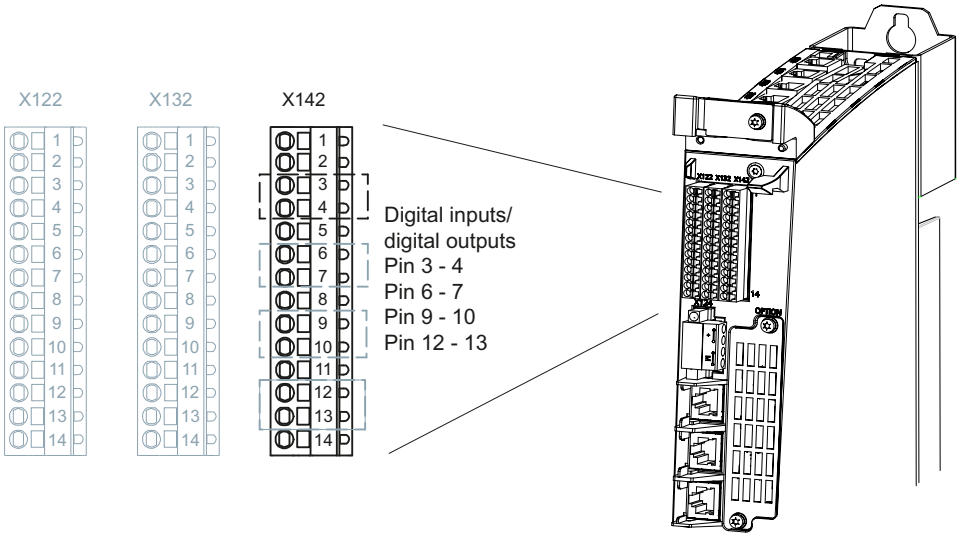


Figure 3-65 SIMOTION D445-2 DP/PN, position of the digital interface X142

The I/Os of the terminal X142 are permanently assigned to SIMOTION D4x5-2. The terminal is thus visible in the project tree under the SIMOTION device.

You define the digital outputs as follows

1. Double-click the element **Inputs/outupts X142** in the project tree below the SIMOTION device.
   The **I/O properties** dialog opens.
2. Go to the **Channels 0-7** tab.
   The terminal **X142** is represented on the tab, specifying the pin numbers and the current use of the channels.

3. Channels 0 and 1 on pins **3** and **4** are preset at the factory as digital inputs DI 0 and DI 1. Define the two channels as digital outputs DO 0 and DO 1. Select the value **DO** in each case in the **Function** list field.

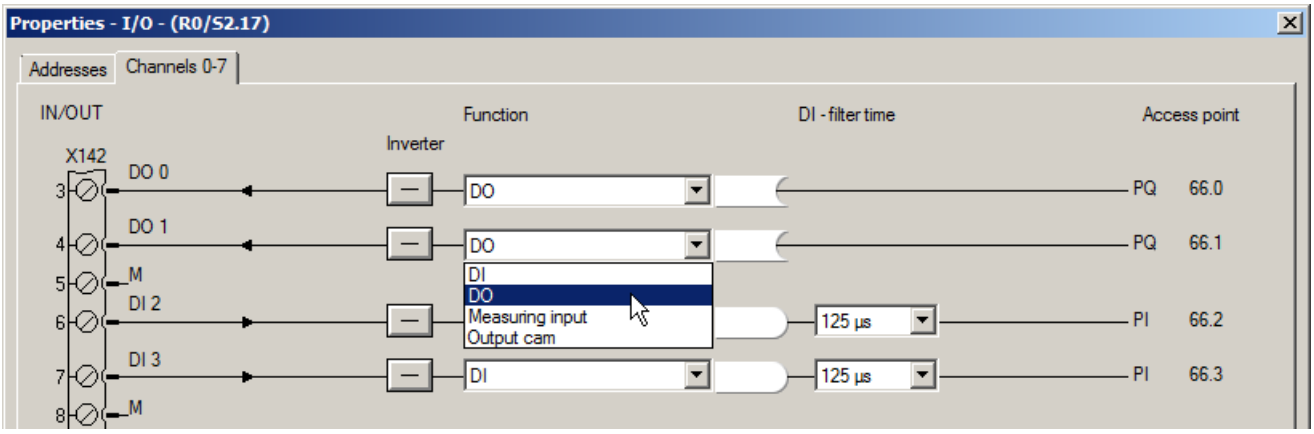


Figure 3-66 I/O properties

4. Confirm with **OK**.
Configuring the digital outputs is thus completed.

3.2.12 Programming the SIMOTION application

3.2.12.1 Overview

Aim of Getting Started

In this section, you write a simple SIMOTION user program for the sample project.

- You create the variables required by the program.
- You create the program and additional auxiliary programs with the graphical editors supplied by SIMOTION SCOUT.

Purpose of the program

The program controls the axis set up in the sample project.

- The infeed is switched on via a SINAMICS system call.
- The axis is traversed to a specific position and returned to the original position.
- The program states "Program started" and "Program end reached" are applied to digital outputs to evaluate them in the I/O.

Preconditions

- You have configured the infeed, see the section Configure the infeed (Page 169).
- You have configured an axis and tested its function, see the sections Configure the axis (Page 171) and Test the axis with the axis control panel (Page 177).
- You have prepared two digital outputs for use in the program, see the section Configure digital outputs (Page 181).
- SIMOTION SCOUT is in offline mode.

3.2.12.2 Variables

Variable types

Variable types

Several variable types are distinguished in SIMOTION:

- **System variables**
  Each SIMOTION device and each technology object has specific system variables. You can access system variables within the SIMOTION device from all programs.
- **I/O variables**
  An I/O variable is a symbolic variable name that is assigned to an I/O address of the SIMOTION device or to the I/O. Direct access to the I/O is thus possible.
  I/O variables are valid across all devices. All programs of the SIMOTION device have access to them.
- **Global device variables, unit variables, and local variables** are user-defined variables with a limited scope of validity:
  All programs of a SIMOTION device can access global device variables.
  Unit variables can be accessed by all programs, function blocks and functions defined within the same unit, e.g. ST unit, MCC unit, LAD/FBD unit.
  A unit (source) is a logic unit that you can create in your project and that can contain programs, functions and function blocks.
  Local variables can only be accessed within the program, the function or the function block in which they are defined.

You can find further information on variables in the online help under **Variable model**.

Variables of the sample project

You require the following variables for the sample project of Getting Started:

2 global device variables

- g_bo_start
- g_bo_ready

2 I/O variables

- q_bo_output0
- q_bo_output1

Additionally when using an infeed with DRIVE-CLiQ interface:

2 I/O variables

- LineModule_STW
- LineModule_ZSW

Instance

- myFB_LineControl

Use of the variables in the sample project

The variables **g_bo_start** and **g_bo_ready** redundantly describe the program states "Program started" and "Program end reached".

The I/O variables **q_bo_output0** and **q_bo_output1** apply the program state to the configured digital outputs of the SIMOTION device, e.g. for a display.

The instance **myFB_LineControl** and the I/O variables **LineModule_STW** and **LineModule_ZSW** control the infeed.

Creating global device variables

You create global device variables in the **Symbol browser** tab of the detail view.

You create global device variables as follows

Create the global device variables **g_bo_start** and **g_bo_ready** for the sample project as follows.

The variable definition encompasses:

- Variable name
- BOOL data type

Note

You can only create global device variables in offline mode.

1. Click in the project navigator under the SIMOTION device on the element **GLOBAL DEVICE VARIABLES**.
   The **Global device variables** table is displayed in the symbol browser.
2. Click in the **Name** column on the first free cell and enter the variable name **g_bo_start**. Press RETURN or TAB. The input focus jumps to the **Data type** field. Alternatively, you can click in the field to move the input focus there.
3. Enter the data type **BOOL** in the **Data type** field.

4. Press RETURN to confirm.
   The variable is created and is available in the project. A new empty line is opened for input in the symbol browser.
5. Create the global device variable **g_bo_ready** accordingly.

The figure below shows the full definition.

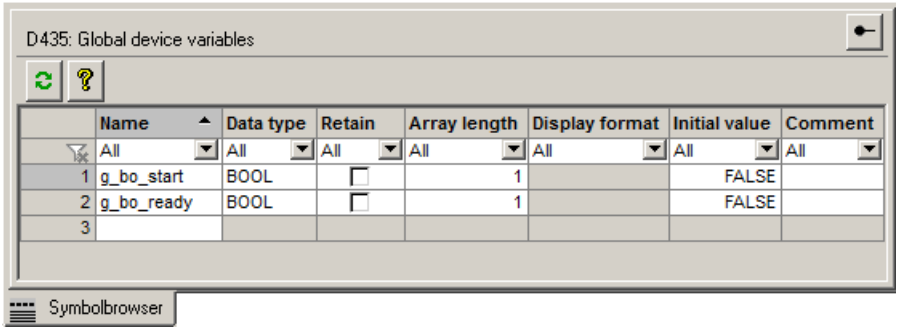


Figure 3-67 Global device variables of the sample project

Creating I/O variables

When configuring I/Os, SIMOTION SCOUT (as of Version 4.2) supports the symbolic assignment of inputs and outputs located on SIMOTION/SINAMICS components. To be able to access the drive-level I/O with an I/O variable, it is only necessary to assign the I/O variable to the I/O channel. SIMOTION SCOUT automatically sets up telegrams, BICO interconnections and addresses.

You create the I/O variables for output to the configured digital outputs as follows

You create the I/O variables **q_bo_output0** and **q_bo_output1** for the sample project as follows.

The variable declaration encompasses:

- Variable name
- Definition as output variable OUT
- BOOL data type
- Assignment to a digital output

Note

I/O variables can only be created in offline mode.

1. Double-click the element **ADDRESS LIST** in the project navigator below the SIMOTION device. The **Address list** tab opens in the detail view.
2. Click the first free cell in the **Name** column. Enter the variable name **q_bo_output0**. Press RETURN or TAB. The input focus jumps to the **I/O address** field. Alternatively, you can click in the field to move the input focus there.
3. Enter the keyword **OUT** in the **I/O address** field.

4. Enter the data type **BOOL** in the **Data type** field, or leave the field empty.
5. Assign the configured digital output to the variable:
   - Click the button ... in the **Assignment** cell. The Assignment dialog opens.
   - Open the **D435** element in the Assignment dialog.
   - Click the digital output **DO_0** below the **D435** element. **Assign** appears in the **Assignment** column.

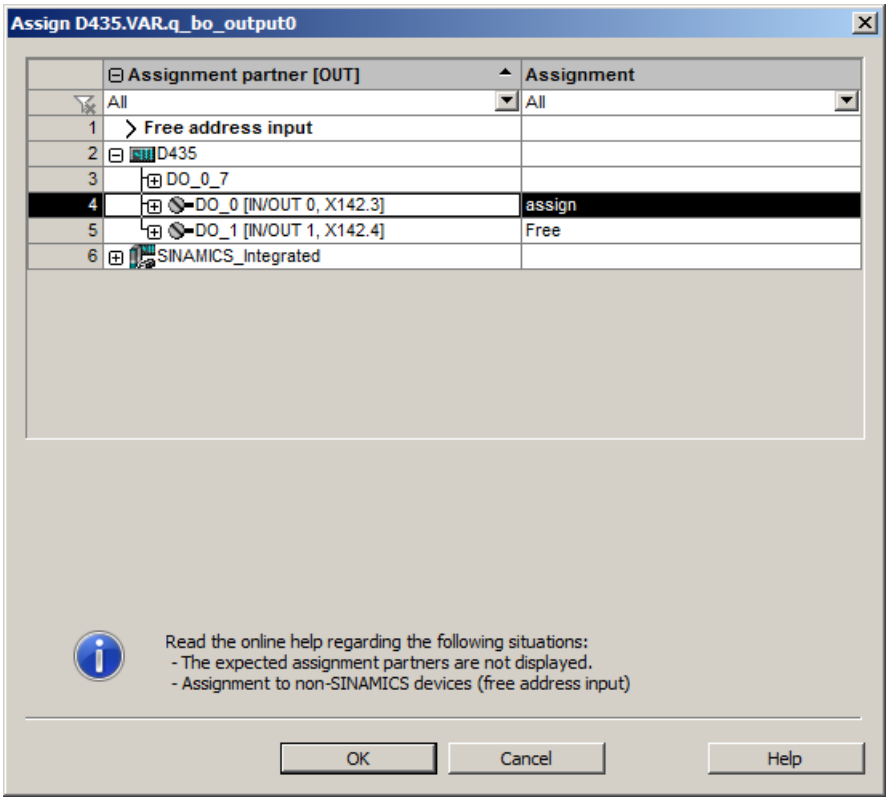


Figure 3-68 Assigning I/O variable and I/O channel

6. Confirm with **OK**.
   The assignment of I/O variable and I/O channel has been completed. The I/O variable is created and is available in the project. A new empty line is opened for input in the **Address list** table.

7. Create the I/O variable **q_bo_output1** in the same way. Assign the variable to the configured digital output DO 1.
   The figure below shows the full declaration:

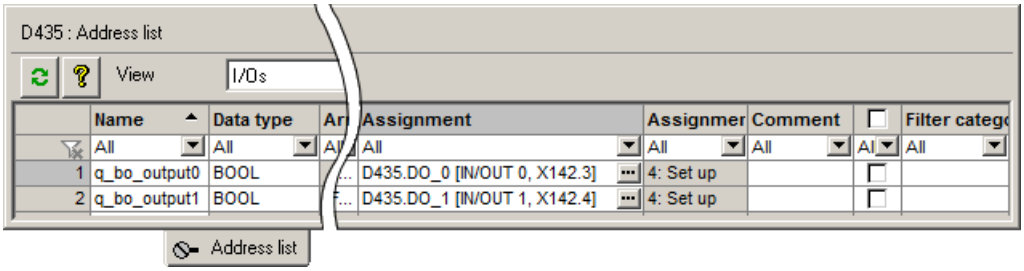


Figure 3-69 I/O variables of the sample project

8. Leave the **Address list** tab open for declaring further I/O variables.

You create the I/O variables for controlling the infeed as follows

If you are using an infeed with DRIVE-CLiQ interface, create also the I/O variables **LineModule_STW** and **LineModule_ZSW**.

Note

I/O variables can only be created in offline mode.

Proceed as for the definition of the I/O variables **q_bo_output0** and **q_bo_output1**.

1. Open the **Address list** tab in the detail area if not already visible.
2. Create the I/O variable **LineModule_STW**.
   - **I/O address**: OUT
   - **Data type**: WORD
   - **Assignment**: Control word E_STW1 of the SINAMICS infeed

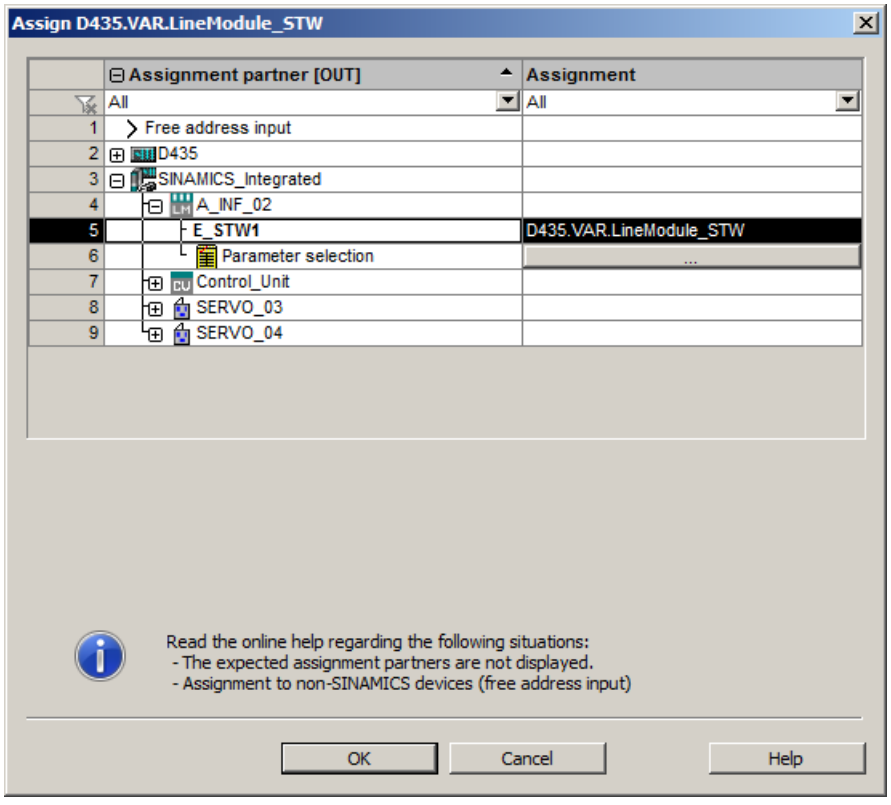


Figure 3-70 Assigning I/O variable and control word of the infeed

3. Create the I/O variable **LineModule_ZSW**.
   - **I/O address**: IN
   - **Data type**: WORD
   - **Assignment**: Status word E_ZSW1 of the SINAMICS infeed

The figure below shows the full declaration:

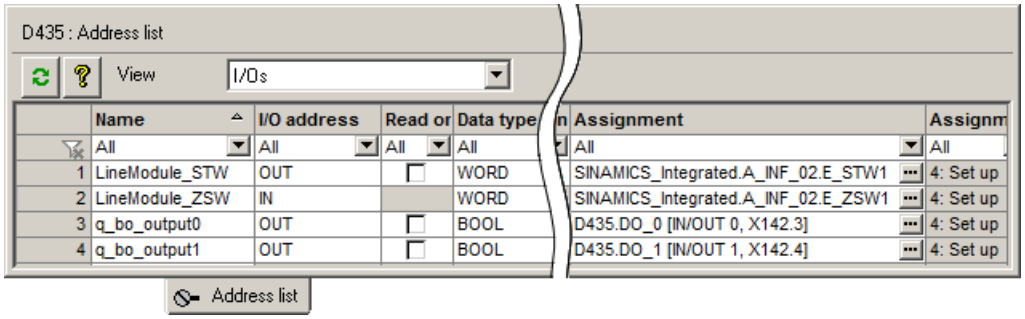


Figure 3-71 I/O variables of the sample project

Back up the configuration

Back up the variables created in the sample project.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.

3.2.12.3 Programming

Programming languages in the sample project

Within the scope of the sample project, you create a simple user program with the programming languages MCC and LAD/FBD.

For this, you use the variables you created in the previous section Variables (Page 183).

MCC Motion Control Chart

The programming language MCC

MCC (Motion Control Chart) is a graphical programming language.

The programmed motion sequences (machine sequences) are shown in the MCC as flowcharts (MCC charts). The structure of the flowcharts is oriented around the actual operating sequence of the machine. A sequential motion sequence is programmed.

System in SIMOTION SCOUT

The **PROGRAMS** folder under the SIMOTION device contains the MCC units created in the project.

An MCC unit contains the MCC programs that are to run on the SIMOTION device.

"MCC program" is a collective term for different program organization units (POU).

A POU can be a program, a function, or a function block. The type of the POU is indicated in the project navigator by an icon:

- Program
- Function
- Function block

An MCC chart is the graphical representation of a program organization unit POU.

An MCC unit can contain several MCC charts.

Program creation steps

Creation of an MCC program encompasses the following steps:

1. Creating the MCC unit.
2. Creating the MCC charts in the MCC unit.
3. Inserting MCC commands in the MCC chart and parameterizing the commands.

Creating the MCC unit

You create the MCC unit **motion** for the sample project as follows.

You create an MCC unit in the project as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert MCC unit**. The **Insert MCC unit** window appears.
2. Enter the name **motion**.
3. Go to the **Compiler** tab. For diagnostics purposes, activate the options **Permit program status** and **Permit single step**. In this way, you can monitor program execution later in online mode.
4. Confirm with **OK**.
   The MCC unit is created.
   - The unit appears in the project navigator under the PROGRAMS branch.
   - In the working area of the workbench, the declaration table of the unit opens. The variables declared there apply within the MCC unit and can be linked in other units.

   No other variable declaration is required for the sample project. You have already created the necessary variables as global device variables in the symbol browser.

Creating the MCC chart

You create the MCC chart **pos_axis** for the sample project as follows. The MCC chart **pos_axis** is a POU of the type program.

You create an MCC chart as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator.
2. Open the MCC unit **motion** in the **PROGRAMS** folder.
3. Double-click **Insert MCC chart**.
   The **Insert MCC Chart** window appears.
4. Enter the name **pos_axis**.
5. Select the creation type **Program**.
6. Confirm with **OK**.
   The MCC chart is created in the project.
   - The created MCC chart **pos_axis** appears in the **PROGRAMS** folder under the **motion** unit.
   - The MCC editor is opened in the working area of the workbench. The start and end nodes are already pre-defined. You can start MCC programming.

Inserting command blocks into an MCC chart

Every newly created MCC chart already contains a start and end node.

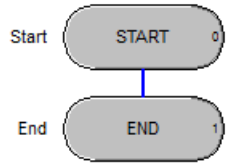


Figure 3-72 MCC chart, start and end block

You insert the MCC command blocks between these. The commands are processed in the direction from the start to the end node.

The MCC commands are available to you via:

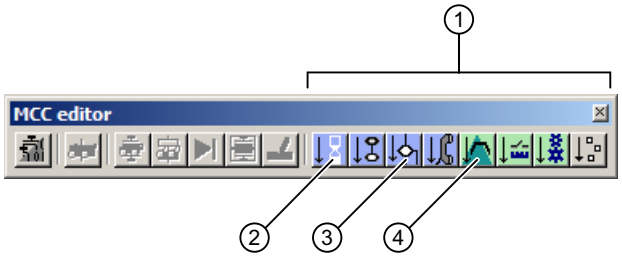
- **MCC editor** toolbar
- **MCC Chart > Paste** menu command
- Context menu of the command block

You work with the MCC editor toolbar as follows

Open the toolbar

The **MCC editor** toolbar becomes visible in the workbench as soon as you open an MCC chart.

The commands are arranged into command groups. The sample project uses commands from the command groups **Basic commands**, **Program structures**, and **Single axis commands**.



① Command groups
② Basic commands
③ Program structures
④ Single axis commands

Figure 3-73 MCC editor toolbar

Note

If you do not see the toolbar, check that the display is switched on: Open the menu **View > Toolbars**. Activate the checkbox for **MCC editor** in the **Toolbars** window.

Open the command group

Move the mouse over the colored buttons of the toolbar to show the command groups.

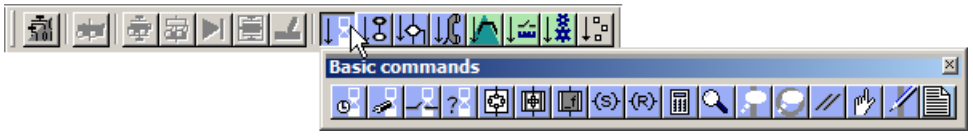


Figure 3-74 Open the command group

Keeping command groups open or closed continuously

Click the window title of a command group to keep the command group open continuously.

Select **Hide** in the context menu of a command group to close the command group.

Placing the command group as required

Drag the toolbar or the command groups of the toolbar with the mouse to any location on the workbench.

Docking the toolbar

Drag the toolbar or the command groups of the toolbar with the mouse to the edge areas of the workbench to dock them there.

Showing a tooltip for the command

Hold the mouse pointer briefly over a command button. The designation of the command is shown.

You insert commands into an MCC chart with the help of the MCC editor toolbar as follows

1. Click in the active MCC chart on the connecting line between two commands, or click the command after which the new command is to be inserted.
   The connecting line or the border of the command button is marked in blue. The marking flashes.
2. Select the command group in the **MCC editor** toolbar. Click the desired command in the command group.
   The command is inserted into the chart.

The newly inserted command is empty. It must then be parameterized; e.g. for a variable assignment, the name of the variable and the assigned value must be specified.

The following section Creating an MCC sample program: basic framework (Page 193) describes the procedure in detail.

3.2.12.4 Creating an MCC sample program: basic framework

Overview

In the sample project, you create the MCC program **pos_axis**. This program traverses an axis to a target position and back to the starting position.

Handling the infeed

If you are using an infeed without a DRIVE-CLIQ interface, the basic framework for program-controlled operation of the axis dealt with in this section is sufficient. The ready signal of the infeed is wired to the digital input DI 0 of the D435-2, see section Configure the infeed (Page 169).

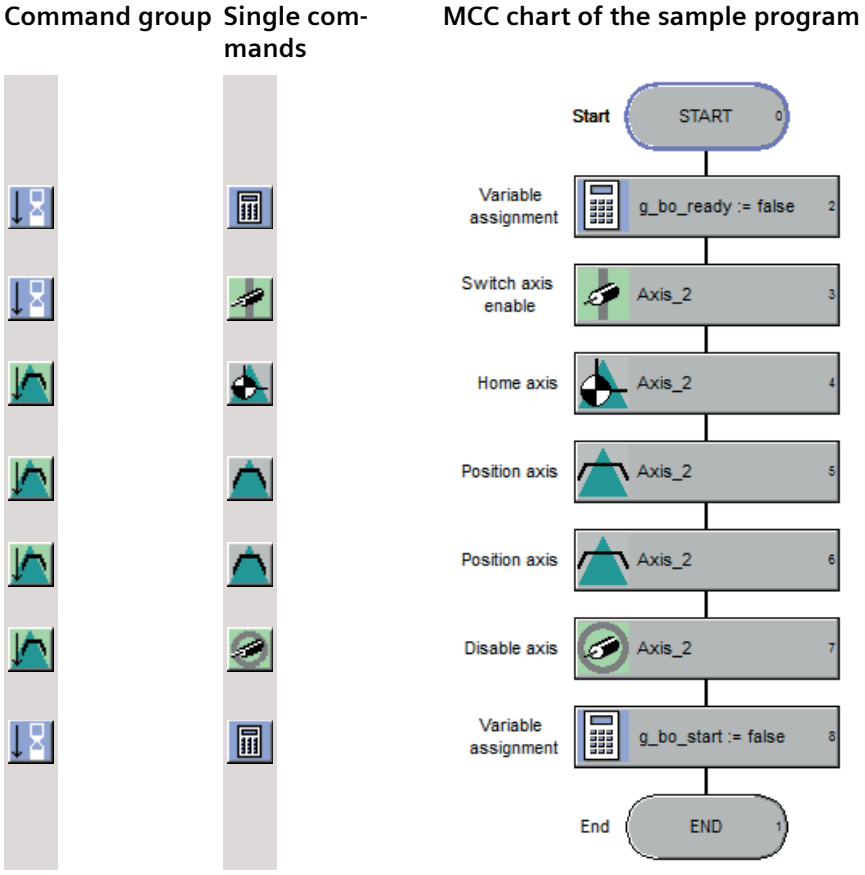
If you are using an infeed with a DRIVE-CLiQ interface, the program must switch the infeed on. Getting Started deals with this additional program sequence separately in the section Expanding the MCC sample program: control of the infeed (Page 205).

Program flow

The finished MCC chart

The figure below shows the finished MCC chart of the sample program.

The first two command blocks initialize the variables **g_bo_ready** and **g_bo_start**. The axis is then enabled, referenced, and traversed to the target position. When the position has been reached, the axis returns to the starting position. The enable is revoked. Finally, the variable **g_bo_start** is reset and the variable **g_bo_ready** is set.



Command groups and individual commands

In the figure above, the icons of the command group and of the individual command that you must click in sequence to insert the command into the MCC chart are shown in front of every MCC command.

The sections below describe the procedure in detail.

Variable assignment g_bo_ready:=false / g_bo_start:=true

The variable **g_bo_ready** is set to FALSE to set the variable to a defined starting point.

The variable **g_bo_start** is set to TRUE to indicate the start of the axis.

To assign the variables in the MCC editor, proceed as follows:

1. Mark the line between the start and end nodes.

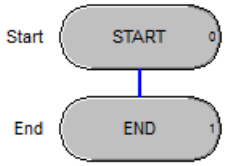


Figure 3-75 Empty MCC chart

2. Open the **Basic commands** command group in the **MCC editor** toolbar. Click the **Variable assignment** command in the command group.

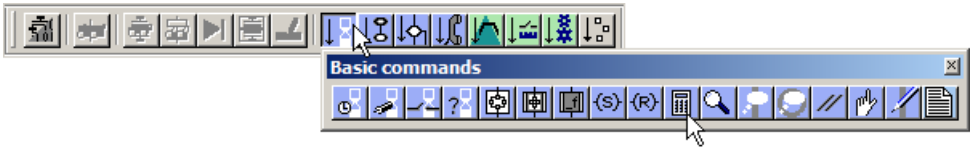


Figure 3-76 MCC editor toolbar, variable assignment command

The **Variable assignment** command block is inserted into the MCC chart after the start node.

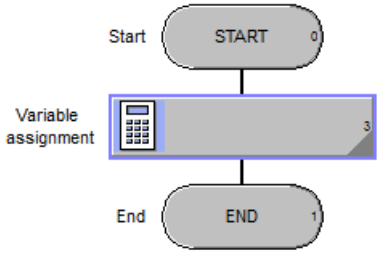


Figure 3-77 MCC chart with inserted variable assignment command block

3. Assign the value FALSE to the variable **g_bo_ready**. You transfer the variable name from the symbol browser to the command block:
   - Double-click the **Variable assignment** command block.
     The **Variable assignment** window appears.
   - Select the value **Formula** in the drop-down list.
     A table with the columns **Variables** and **Expression** appears.
   - Open the symbol browser. To do so, click **GLOBAL DEVICE VARIABLES** under the SIMOTION device in the project navigator.
   - Select the variable **g_bo_ready** in the symbol browser. To do so, click in the first column of the symbol browser on the preceding line number. The line is shown on a black background.

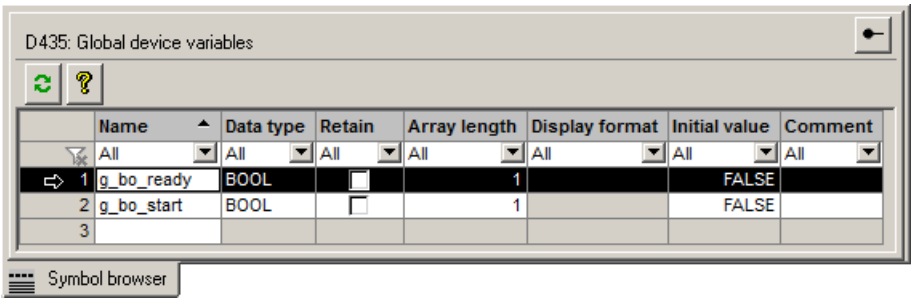


Figure 3-78 Symbol browser

   - Drag and drop the marked line to the **Variable** column of the **Variable assignment** window. The variable is inserted and its name displayed.
   - In the **Expression** column, enter the value **false** and confirm with RETURN.

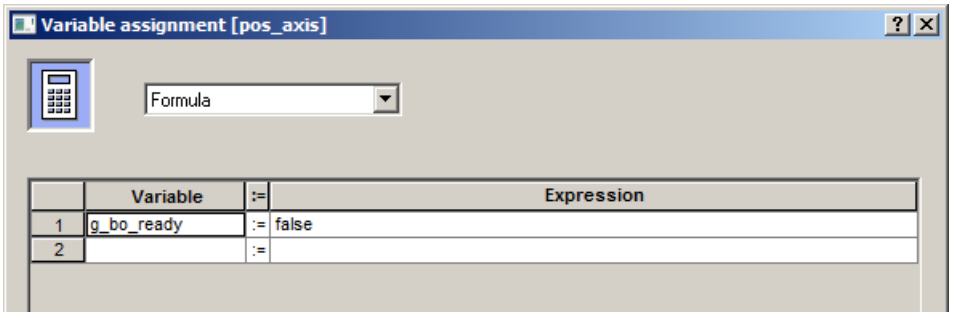


Figure 3-79 Variable assignment

4. Assign the value **true** to the variable **g_bo_ready** in the same way.

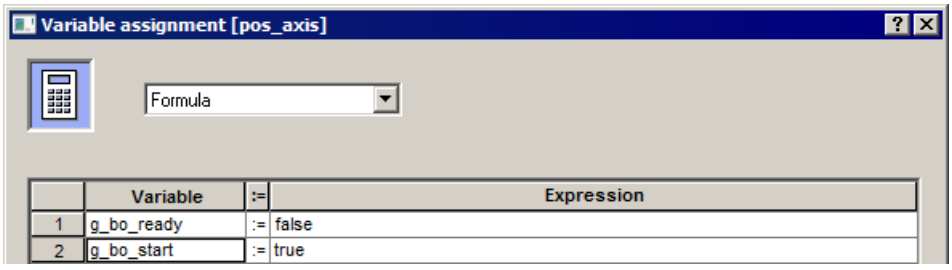


Figure 3-80 Variables assigned

5. Click **OK** to close the window.
You have set the global device variables to the values **false** and **true**.

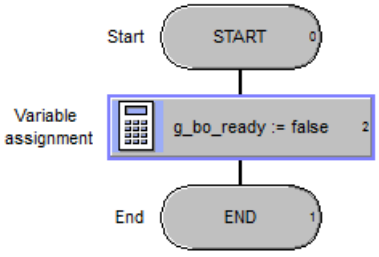


Figure 3-81 MCC chart, executed variable assignment

Switch axis enable

You switch the axis enable as follows

1. Insert the statement block **Switch axis enable** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Switch axis enable** command in the command group.

Figure 3-82 MCC editor toolbar, Switch axis enable command

The **Axis enable** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the axis enable:
   - Double-click the inserted **Switch axis enable** command block in the MCC chart. The **Switch axis enable** window is displayed.
   - In the **Axis** field, select the axis that is controlled by the program.

– Make sure that the **Delay program execution** option is selected for the sample project.
The command is only finished when the axis is enabled.
Leave the other parameters at the default values.

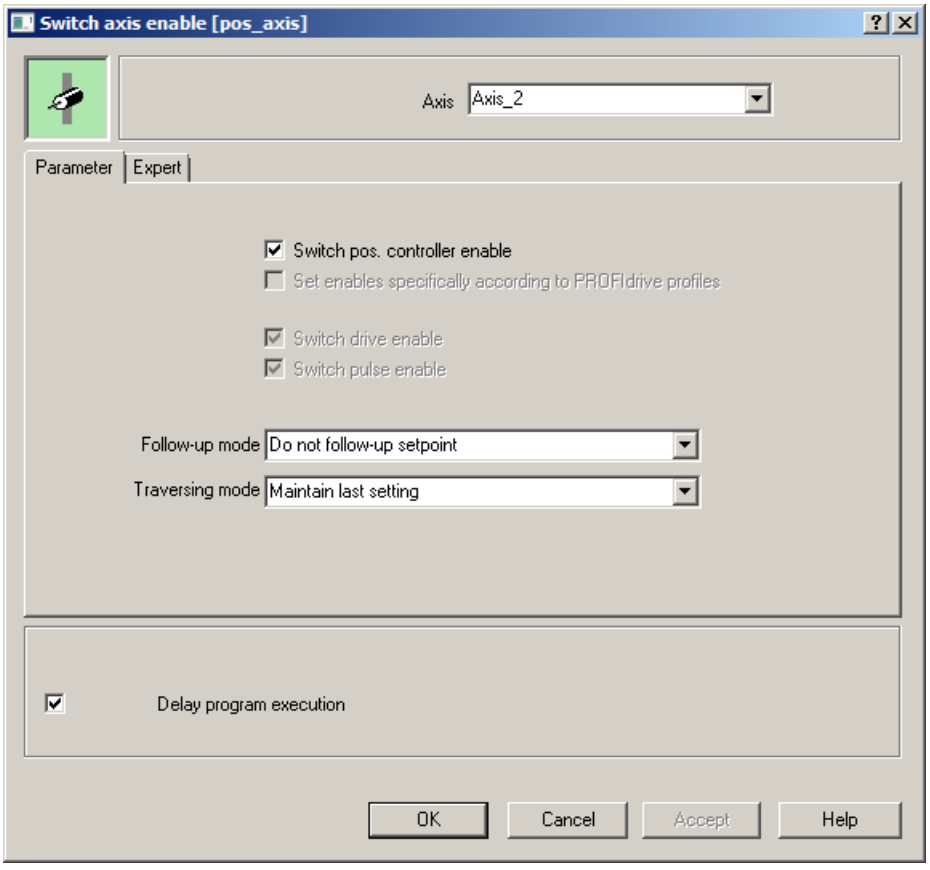


Figure 3-83 MCC chart, Switch axis enable

3. Click **OK** to close the window.
The command is parameterized.

Home axis

The position values of an axis are always in relation to the axis coordinate system. The homing procedure defines where the zero point of the coordinate system is.

Homing of the axis is implemented as follows

1. Insert the statement block **Home axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the command **Home axis** in the command group.

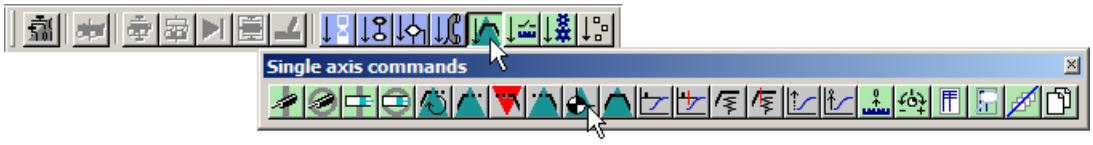


Figure 3-84 MCC editor toolbar, Home axis command

   The **Home axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize axis homing:
   - Double-click the inserted **Home axis** command block in the MCC chart. The **Home axis** window appears.
   - In the **Axis** field, select the axis that is controlled by the program.
   - In the **Parameters** tab, select the homing type **Set home position**. Enter the value **0** in the **Home position coordinates** field.
     With this homing type, the value of the home position coordinates are assigned to the current position of the axis (actual value). There is no active traversing motion.

– Leave the other parameters at the default values.

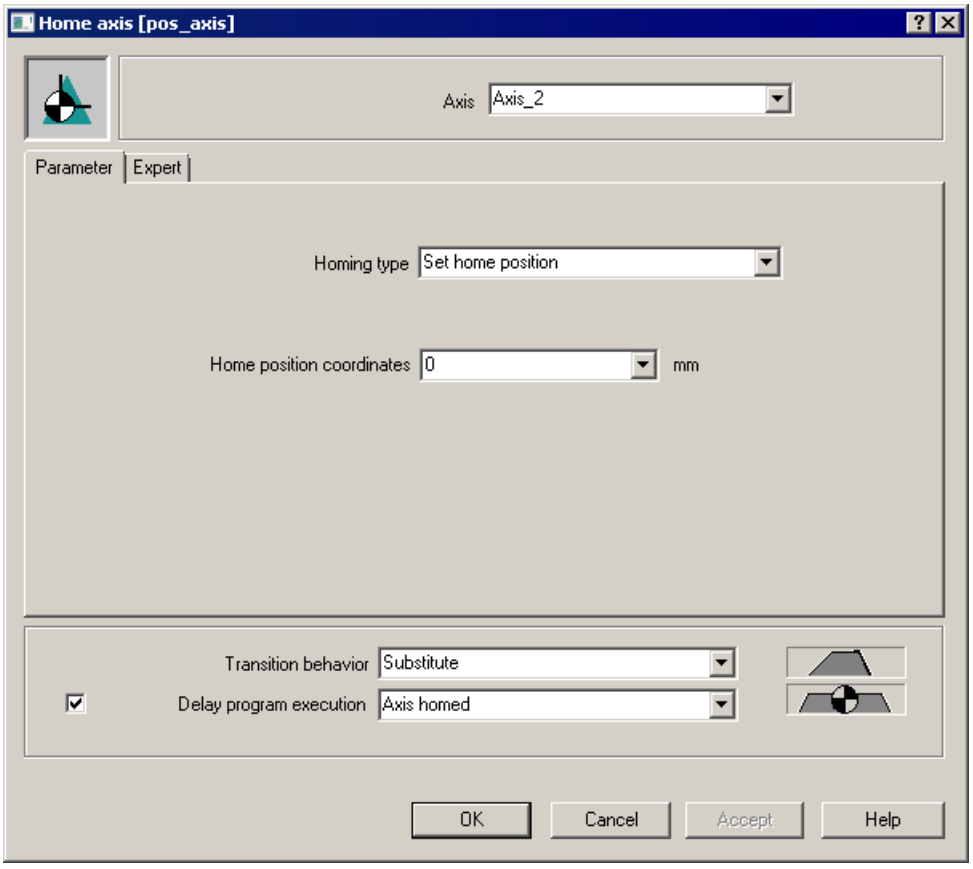


Figure 3-85 MCC chart, Home axis

3. Click **OK** to close the window.
   The command is parameterized.

Position the axis to the target position

The program is to traverse the axis at a velocity of 500 mm/s to the position 2000 mm.

You traverse the axis to the target position as follows

1. Insert the statement block **Position axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Position axis** command in the command group.

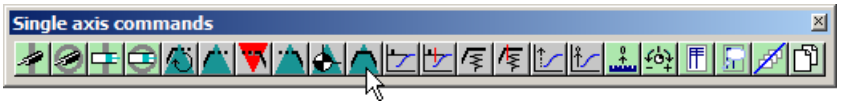


Figure 3-86 MCC editor toolbar, Position axis command

The **Position axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the axis motion:
   - Double-click the inserted **Position axis** command block in the MCC chart. The **Position axis** window appears.
   - In the **Axis** field, select the axis that is controlled by the program.
   - Enter the position 2000 in the **Parameters** tab. Set the velocity to 500 mm/s.
   - Leave the other parameters at the default values.

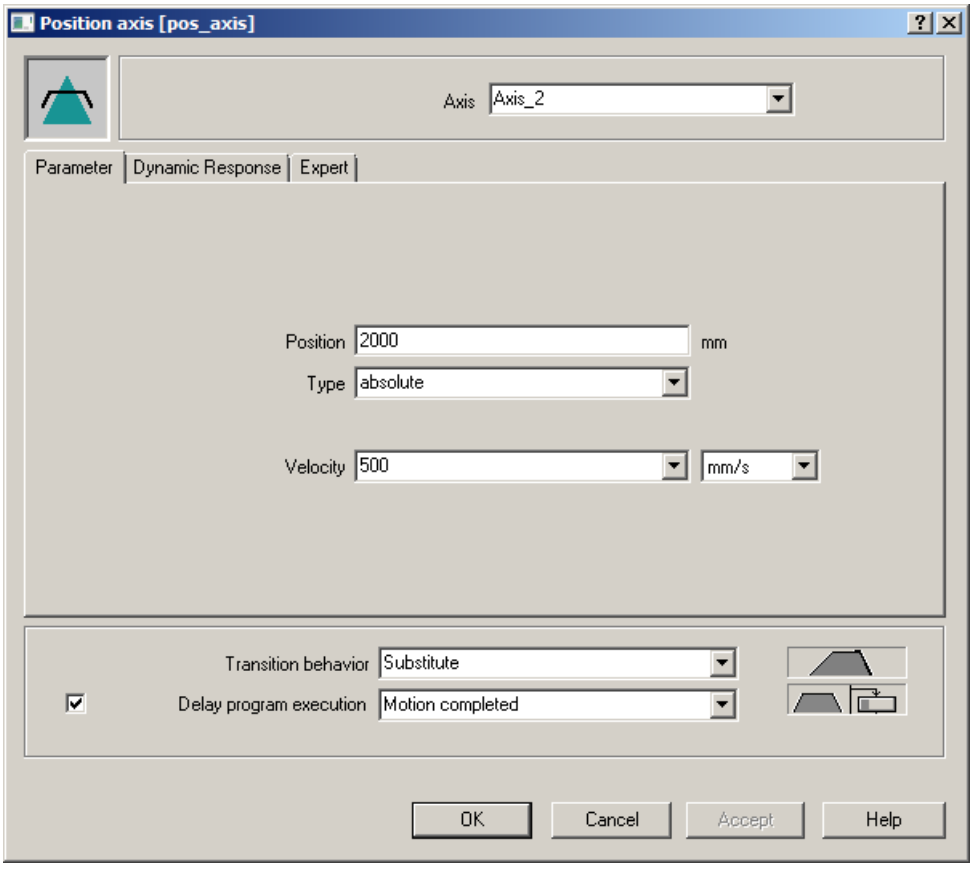


Figure 3-87 MCC chart, position the axis to the target position

3. Click **OK** to close the window.

The command is parameterized.

Position the axis to the starting position

The program is to return the axis to the starting position 0 mm at a velocity of 500 mm/s.

You return the axis to the starting position as follows

Proceed as for the step Position the axis to the target position (Page 201).

1. Insert the statement block **Position axis** into the MCC chart in front of the end node.
2. Parameterize the axis motion:
   - Enter the position 0 in the **Parameters** tab. Set the velocity to 500 mm/s.
   - Leave the other parameters at the default values.

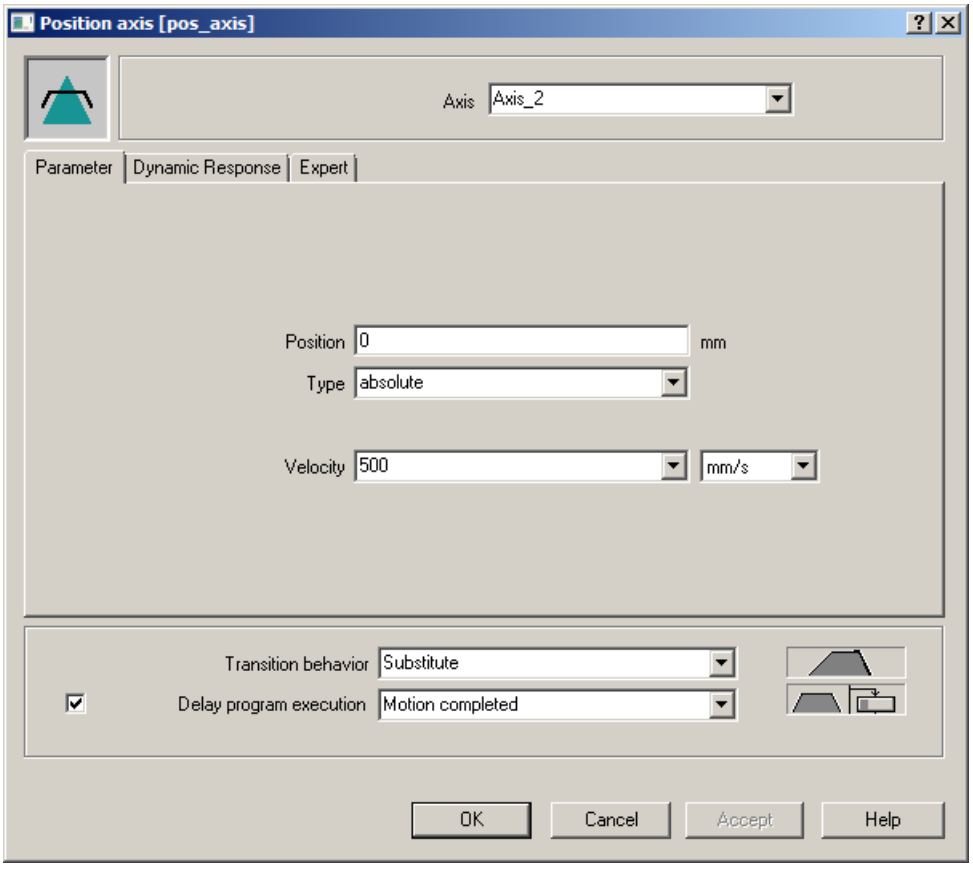


Figure 3-88 MCC chart, position the axis to the starting position

3. Click **OK** to close the window.
   The command is parameterized.

Disable axis

You disable the axis as follows

1. Insert the statement block **Disable axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Disable axis** command in the command group.

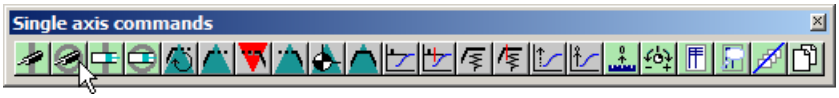


Figure 3-89 MCC editor toolbar, Disable axis command

The **Disable axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the command:
   - Double-click the inserted **Disable axis** command block in the MCC chart. The **Disable axis** window is displayed.

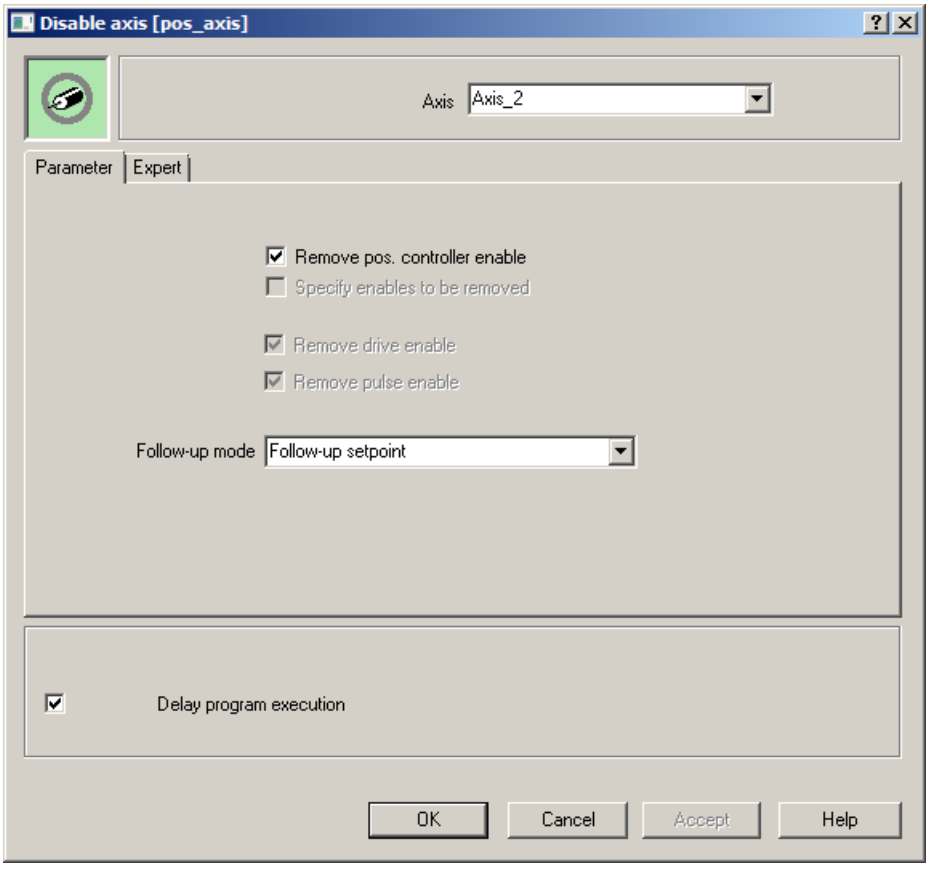


Figure 3-90 MCC chart, Disable axis

   - In the **Axis** field, select the axis that is controlled by the program.

– Leave the other parameters at the default values.

3. Click **OK** to close the window.
   The command is parameterized.

Variable assignment g_bo_start:=false / g_bo_ready:=true

The axis motion is finished. The variable **g_bo_start** is set to FALSE. The variable **g_bo_ready** is set to TRUE.

To assign the variables in the MCC editor, proceed as follows:

Proceed as for the step Variable assignment g_bo_ready:=false / g_bo_start:=true (Page 194)

1. Insert the statement block **Variable assignment** into the MCC chart in front of the end node.
2. Assign the value **false** to the variable **g_bo_start** and the value **true** to the variable **g_bo_ready**.
3. Click **OK** to close the window.
   The command is parameterized.

3.2.12.5 Expanding the MCC sample program: control of the infeed

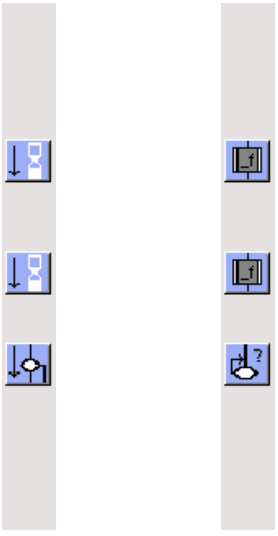
Program flow

If you are using an infeed with DRIVE-CLiQ interface, the programmed controller must switch on the infeed before the axis commands can then be issued. The program sequence shown below handles this task. The **pos_axis** program created until now must be expanded accordingly.

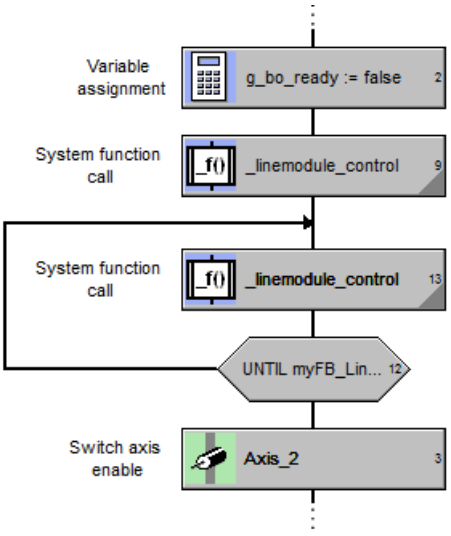
The finished MCC chart

The two times two blocks **_linemodule_control** system function call and **LineModule_STW** variable assignment are located in the MCC chart **pos_axis** before the first command for axis control **Switch axis enable**.

Command group **Single commands**



MCC chart of the sample program, control of the infeed



The first two blocks before the **UNTIL** loop reset the operating parameters of the infeed to the start values. Without this initialization, it may not be possible to switch the infeed on. In addition, this also acknowledges alarm messages that prevent enabling of the infeed.

The blocks within the **UNTIL** loop switch the infeed on. The loop repeats the switching command until the infeed is ready for operation.

Note

To avoid excess load on the performance of the round robin execution level (background task, motion tasks) during endless loops, it makes sense to use a _waitTime (T#0ms) in the loop while actively waiting for an event.

System function call _LineModule_control[FB]

You create the system function call _LineModule_control[FB] as follows

1. Insert the statement block **System function call** into the MCC chart:
   - Click the command block **g_bo_start := true**.
     The node is marked in blue.
   - Select the **Basic commands > System function call** command in the **MCC editor** toolbar.

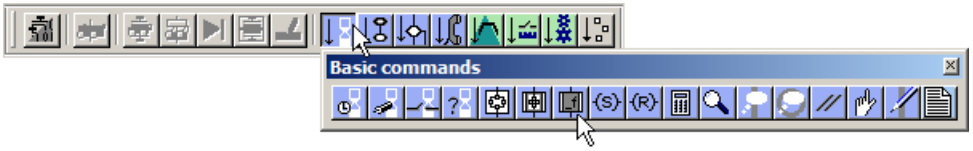


Figure 3-91 MCC editor toolbar, System function call command

The command block is inserted into the MCC chart.

2. Parameterize the function call:
   - Double-click the empty **System function call** command block in the MCC chart.
     The **System function call [pos_axis]** dialog appears.
   - Open the command library. You can find the command library in the screen area of the project navigator as a separate tab.

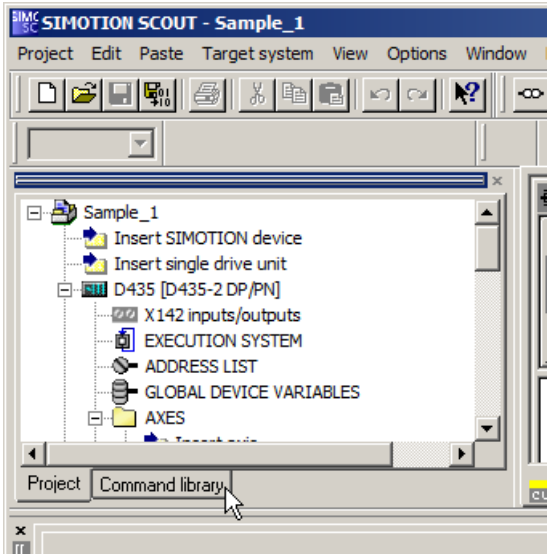


Figure 3-92 Open the command library

   - Go to the **Drives > SINAMICS** branch of the command library.
   - Drag & drop the **_LineModule_control[FB]** function call from the command library to the **System function** field of the **System function call[pos_axis]** window.

3. Define an instance for the system function call:
   – Enter the instance name **myFB_LineControl** in the **Instance** field of the **System function call [pos_axis]** window.
   As soon as you exit the field, e.g. by clicking on another element or with the TAB or RETURN keys, the **Variables declaration** dialog appears. The instance variable **myFB_LineControl** is assigned to the variable type **VAR** as default. The variable is thus valid locally within the program **pos_axis**.

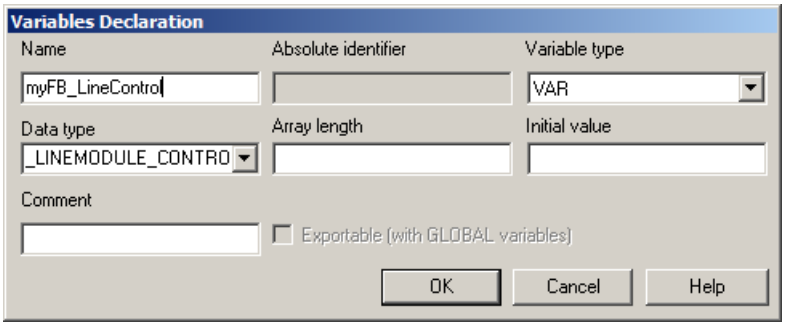


Figure 3-93 Variable declaration of the instance variable myFB_LineControl

Accept the default. Click **OK** to confirm the dialog.

– Set the input variables VAR_INPUT and the output variable VAR_OUTPUT of the instance to the following values.
Make use of the convenience of the autocomplete operator function **Ctrl+spacebar**:

Variable name	Value	Meaning
reset	true	Reset the infeed
periIn	LineModule_ZSW	Status word of the infeed
typeLM	ACTIVE_LINE_MODULE SMART_LINE_MODULE BASIC_LINE_MODULE	Type of the module to be controlled depending on the infeed used
periOut	LineModule_STW	Control word, sequence control infeed

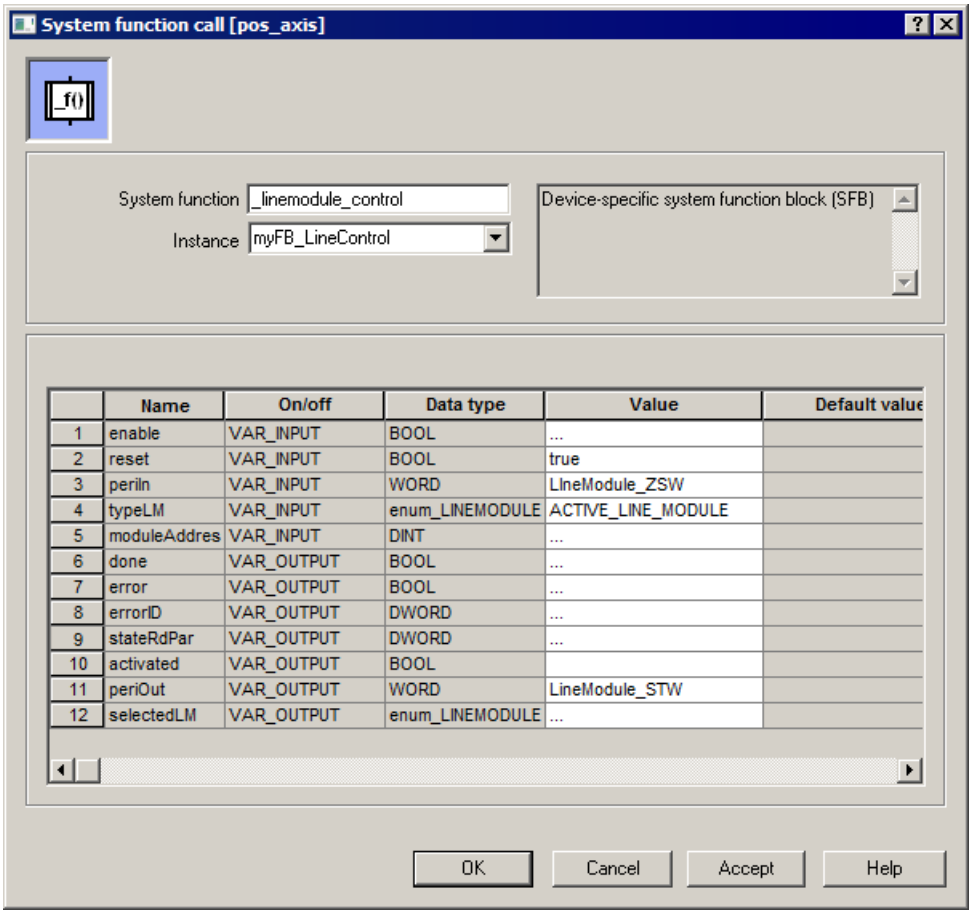


Figure 3-94 MCC chart, system function call _linemodule_control

4. Confirm with **OK**. The command is parameterized.

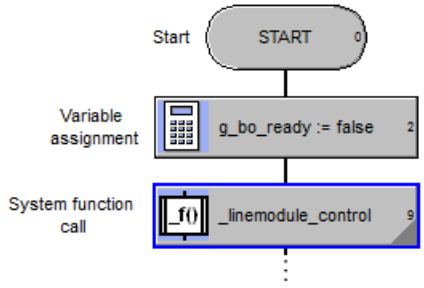


Figure 3-95 MCC chart, executed variable assignment _linemodule_control

Create UNTIL loop

You create the UNTIL loop as follows

1. Click the connecting line between the blocks **System function call** and the subsequent axis command **Switch axis enable**.
   The reference point is marked in blue.
2. In the **MCC editor** toolbar, select the command **Program structures > UNTIL: Loop with condition at end**

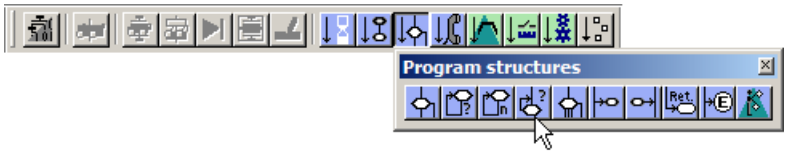


Figure 3-96 MCC editor toolbar, UNTIL loop

The **UNTIL** block with loop is inserted.

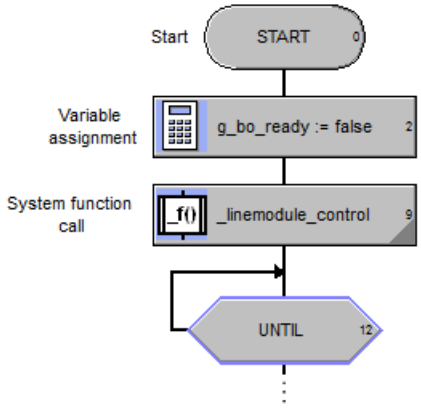


Figure 3-97 MCC editor, UNTIL loop

3. Parameterize the loop condition:
   The loop is to be run through until the infeed is ready for operation.
   - Double-click the UNTIL block. The window **UNTIL: Loop with condition at end[]pos_axis]** appears.
   - Select the value **Formula** in the field at top left.
   - Enter the loop condition in the **Until** field: **myFB_LineControl.activated=TRUE**.
     You can use the autocomplete function **Ctrl+spacebar** in this field too.
4. Confirm with **OK**.
   The loop is created and parameterized in the program.

Copy blocks

You copy the statement blocks in the MCC chart as follows

Copy the statement blocks **System function call _LineModule_control[FB]** and **Variable assignment LineModule_STW:=myFB_LineControl.periOut** into the **UNTIL** loop. Use the operator functions **Copy** and **Paste** for this purpose.

1. Mark the blocks:
   - Hold the **Shift** key down.
   - Click the blocks one after the other. Both blocks are edged in blue.
2. Copy the selected blocks to the clipboard:
   - Open the context menu with the right mouse key.
   - Select the **Copy** command in the context menu.

   Or use the **Ctrl+C** key function.
3. Select the target of the copy operation in the MCC chart:
   - Click the connecting line within the **UNTIL** loop. The connecting piece is marked in blue.

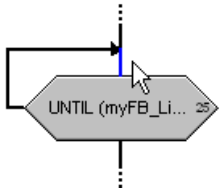


Figure 3-98 MCC chart, mark the UNTIL loop internal area

4. Insert the copied blocks at the target position:
   - Open the context menu again with the right mouse key.
   - Select the **Paste** command in the context menu.

   Or use the **Ctrl+V** key function.
   The blocks are inserted within the loop.

Adapt the system function call _linemodule_control

You change the system function call _linemodule_control as follows

The system function call **_linemodule_control** within the **UNTIL** loop switches the infeed on.

1. Open the system function block by double-clicking.
2. Adapt the parameter variables of the **myFB_LineControl** instance as shown below:

Variable name	Value	Meaning
enable	true	Switches the infeed on
reset		Empty field. The variable is thus set to **false**
periIn	LineModule_ZSW	Status word of the infeed
typeLM	ACTIVE_LINE_MODULE SMART_LINE_MODULE BASIC_LINE_MODULE	Type of the module to be controlled depending on the infeed used

3. Confirm with **OK**.

3.2.12.6 Create additional MCC programs for the sample project

Handling system events

The execution system that you will set up in the later section Configure execution system (Page 222) requires two further MCC programs:

- technology_fault
- peripheral_fault

These programs are used for handling system events. Handling is generally necessary. If a system event occurs that is not handled by the user program, the CPU goes to STOP mode.

Within the scope of the sample project, no specific handling of system events is necessary. The programs technology_fault and peripheral_fault therefore remain empty.

You create the MCC programs technology_fault and peripheral_fault as follows

1. Create the MCC unit **fault**.
   Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert MCC unit**. Assign the name **fault** to the MCC unit.
2. Create the MCC chart **technology_fault**.
   Open the MCC unit **fault** in the **PROGRAMS** folder. Double-click **Insert MCC chart**. Assign the name **technology_fault** to the MCC chart. Select the creation type **Program**.
   The program remains empty.

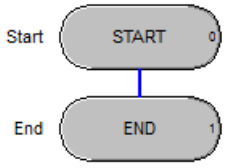


Figure 3-99 Empty program technology_fault

3. Create the MCC chart **peripheral_fault** within the MCC unit **fault**.
   This program also remains empty.

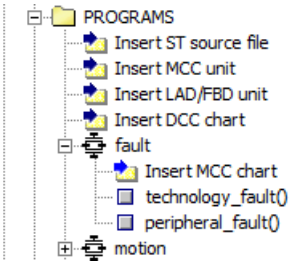
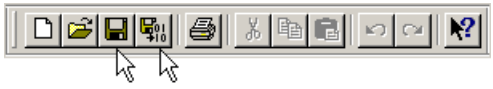


Figure 3-100 MCC programs technology_fault and peripheral_fault in the MCC unit fault

3.2.12.7 Back up MCC sample programs

Back up the created MCC programs.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.



Additional information

As an alternative to the command **Save project and compile changes**, you can find the command **Accept and compile** in the **MCC editor** toolbar.

This command compiles the currently selected program as well as all other programs of the same unit.

However, the command does not save the changes.

You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

3.2.12.8 LAD/FBD ladder logic/function block diagram

The LAD and FBD programming languages

LAD (ladder logic)

LAD stands for ladder logic. LAD is a graphical programming language. The syntax for the statements is similar to a circuit diagram. LAD enables simple tracking of the signal flow between conductor bars via inputs, outputs and operations. The LAD statements comprise elements and blocks that are connected graphically to form networks. LAD operations follow the rules of Boolean logic.

FBD (function block diagram)

FBD stands for function block diagram. FBD is a graphical programming language. To represent the logic relationships, it uses the logic boxes familiar from Boolean algebra. In addition, complex functions (e.g. mathematical functions) can be represented directly in conjunction with the logic boxes.

System in SIMOTION SCOUT

The **PROGRAMS** folder under the SIMOTION device contains the LAD/FBD units created in the project.

A LAD/FBD unit contains the LAD/FBD programs that are to run on the SIMOTION device. A LAD/FBD unit can contain several LAD/FBD programs.

"LAD/FBD program" is a collective term for different program organization units (POU).

A POU can be a program, a function, or a function block. The type of the POU is indicated in the project navigator by an icon:

- Program
- Function
- Function block

Program creation steps

Creation of a LAD/FBD program encompasses the following steps:

1. Creating a LAD/FBD unit.
2. Creating a LAD/FBD program in the LAD/FBD unit.
3. Inserting LAD/FBD commands in the LAD/FBD program and parameterizing the commands.
4. Saving and compiling the LAD/FBD program.

Switching the programming language

SIMOTION SCOUT allows simple switching between ladder logic and function block diagram. The LAD/FBD editor contains the command **LAD/FBD program > Switch to FBD** or **Switch to LAD**.

Create LAD/FBD unit

You create the LAD/FBD unit **bg_out** for the sample project as follows.

You create a LAD/FBD unit in the project as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert LAD/FBD unit**. The **Insert LAD/FBD unit** window appears.
2. Enter the name **bg_out** for the unit.
3. Go to the **Compiler** tab. For diagnostics purposes, activate the option **Permit program status**. In this way, you can monitor program execution later in online mode.
4. Confirm with **OK**.
   The LAD/FBD unit is created.
   - The LAD/FBD unit **bg_out** appears in the **PROGRAMS** folder.

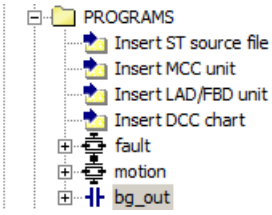


Figure 3-101 MCC and LAD/FBD units in the PROGRAMS folder

   - In the working area of the workbench, the declaration table of the unit opens. The variables declared there apply within the LAD/FBD unit and can be linked in other units. No other variable declaration is required for the sample project. You have already created the necessary variables as global device variables in the symbol browser.

Create LAD/FBD program

You create the LAD/FBD program **LAD_1** within the LAD/FBD unit **bg_out** as follows.

You create a LAD/FBD program in a LAD/FBD unit as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator.
2. Open the LAD/FBD unit **bg_out** in the **PROGRAMS** folder. Double-click **Insert LAD/FBD program**. The **Insert LAD/FBD program** window appears.
3. Enter the name **LAD_1**. The name must be unique throughout the project.

4. Select the creation type **Program**.

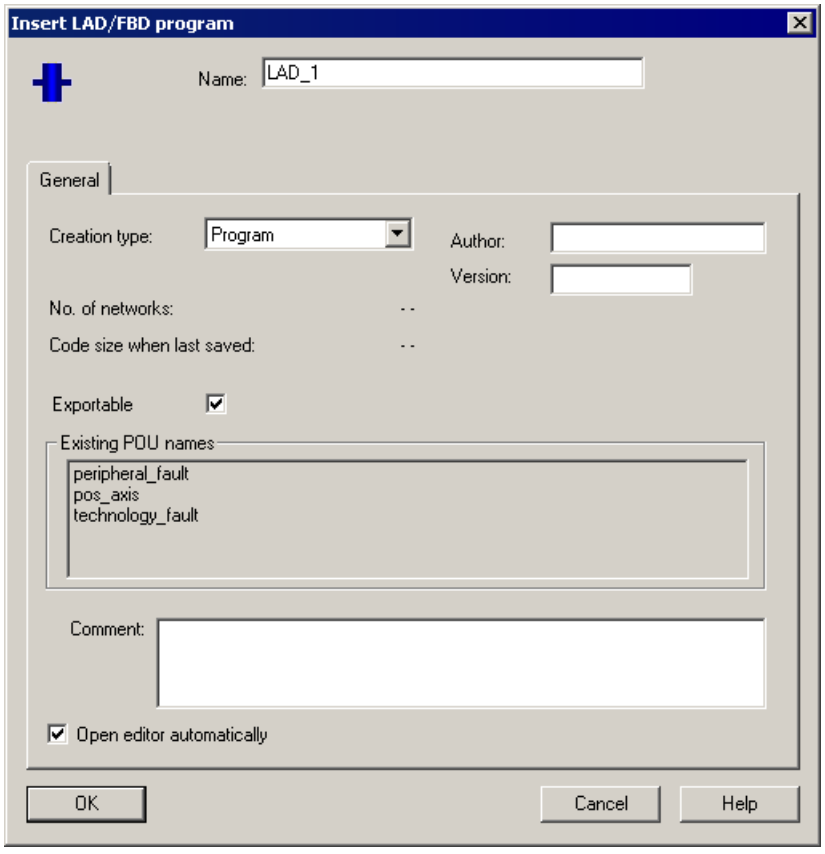


Figure 3-102 Insert LAD/FBD program

5. Confirm with **OK**.
   The LAD/FBD program **LAD_1** is created in the project.
   – The LAD/FBD program appears in the **PROGRAMS** folder.

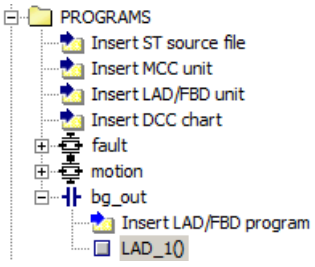


Figure 3-103 LAD program LAD_1 in the PROGRAMS folder

   – The LAD/FBD editor is opened in the working area of the workbench. You can start programming.

Creating a LAD sample program

You write the LAD program **LAD_1** for the sample project.

In this program, the global device variables **g_bo_start** and **g_bo_ready** are read cyclically, and the I/O variables **q_bo_output0** and **q_bo_output1** are set accordingly.

Transfer of the program status to the digital outputs DO 0 and DO 1 could also be programmed direct in the MCC chart **pos_axis**. However, Getting Started implements the assignment in an autonomous LAD program to introduce you to LAD programming.

You create the LAD program LAD_1 for the sample project as follows

1. Open the LAD/FBD program if it is not already visible in the working area of the workbench. For this purpose, double-click the LAD/FBD program under the LAD/FBD unit in the **PROGRAMS** folder of the project navigator.
   In the sample program, the empty LAD/FBD program **LAD_1** is already open.

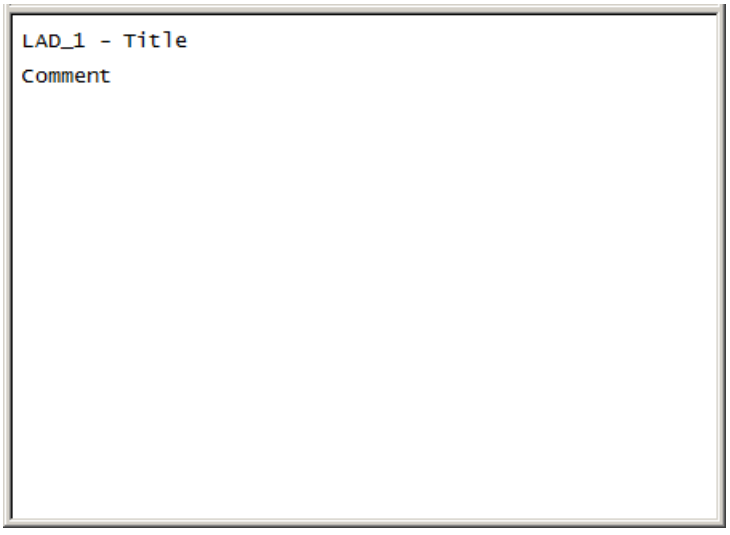


Figure 3-104 Empty LAD program following first-time creation in the LAD/FBD unit

2. Insert a network:
   – Click **Insert network** in the LAD/FBD toolbar.

   The **001** block with a coil is inserted.

3. Insert an NO contact in front of the coil:
   – Select the coil.

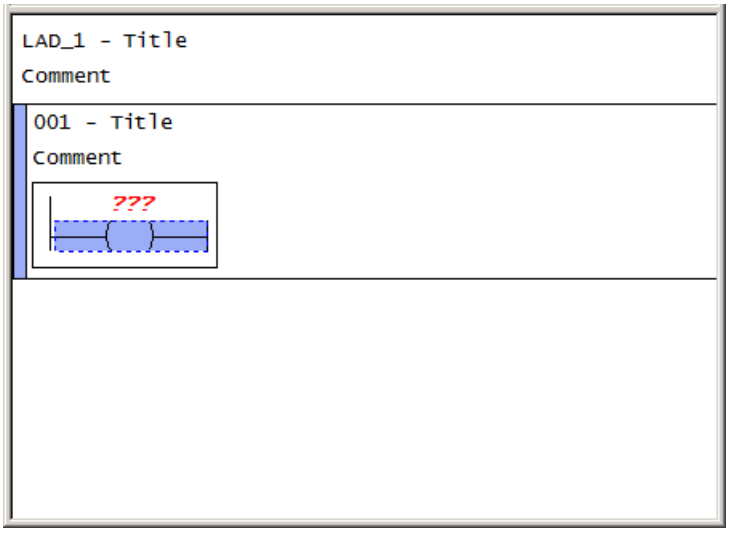


Figure 3-105 LAD program, selected coil

   – Click **NO Contact** in the toolbar.

   The NO contact is inserted in front of the coil.

4. Assign the **g_bo_start** variable to the NO contact and the **q_bo_output0** variable to the coil:
   - Click above the symbols for NO contact and coil on **???**.

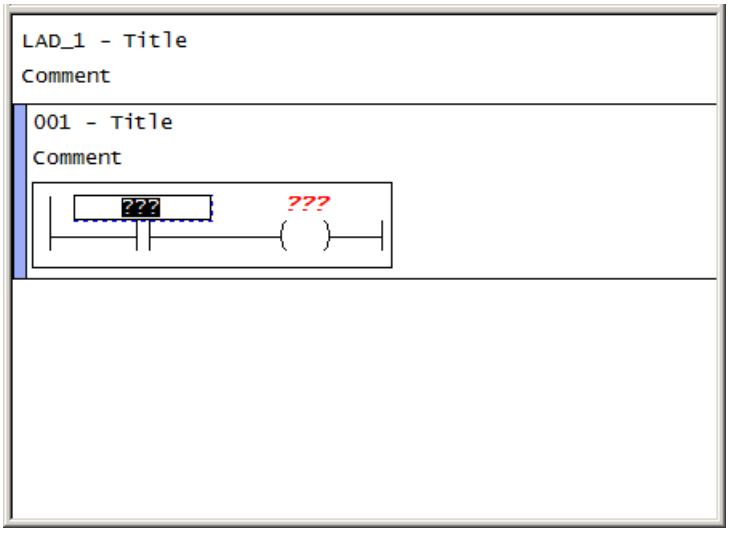


Figure 3-106 LAD program, opened field for entering the variable name

   - Transfer the variable names from the symbol browser or the ADDRESS LIST by dragging and dropping. Or use the Autocomplete function. Use RETURN to confirm each entry.

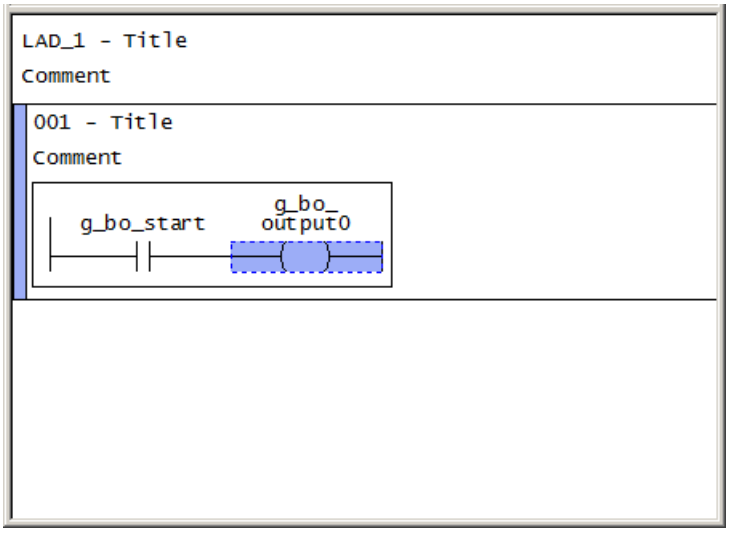


Figure 3-107 LAD program with inserted variables

5. Insert a network **002**:
   - Click the block **001** to select the block.
     A selected block has a blue background on the left edge.
   - Click **Insert network** in the LAD/FBD toolbar.
     The block **002** is inserted after the block **001**.

6. Insert an NO contact in front of the coil in network **002**.
7. Assign the **g_bo_ready** variable to the NO contact and the **q_bo_output1** variable to the coil. The LAD program is finished.

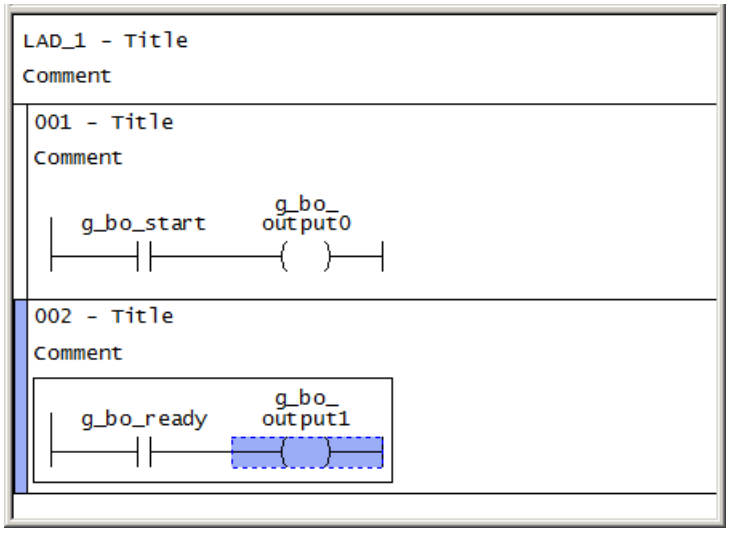


Figure 3-108 Finished LAD program LAD_1

Back up LAD/FBD sample program

Back up the created LAD/FBD sample program.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.

Additional information

As an alternative to the command **Save project and compile changes**, you can find the command **Accept and compile** in the **LAD/FBD toolbar**.

This command compiles the currently selected program as well as all other programs of the same unit.

However, the command does not save the changes.

You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

3.2.12.9 Other programming languages

SIMOTION SCOUT provides other programming languages in addition to MCC and LAD/FBD.

- **ST (Structured Text)**
  ST is a text-based, PASCAL-based programming language. The language is based on the international standard IEC 61131-3. This standard harmonizes the programming languages for programmable logic controllers (PLC). ST is based on the Structured Text part of this standard. In addition to the standardized programming language in accordance with IEC 61131-3, SIMOTION ST features technological commands.
  An easy-to-use text editor is provided for creating programs. The ST compiler compiles the edited program into executable code and indicates every syntax error, specifying the program line and the cause of the error.
  In the Command library tab of the project navigator, the commands and functions required for programming are shown in a tree. You can use these in all programming languages, e.g. when programming.

  Note

  In SIMOTION SCOUT V4.5 and higher, ST can also be used as a language for object-oriented programming. Further information can be found in the SIMOTION SCOUT Online Help and in the SIMOTION ST Structured Text Programming and Operating Manual.

- **DCC (Drive Control Chart)**
  Drive Control Chart (DCC) means graphic configuration and expansion of the device functionality by means of freely available control, calculation and logic blocks.
  DCC comprises the DCC editor and the DCB library (block library with standard DCC blocks). The user-friendly DCC Editor enables easy graphical configuration and a clear representation of control loop structures as well as a high degree of reusability of existing diagrams.
  The open-loop and closed-loop control functionality is defined by using multi-instance-capable blocks (Drive Control Blocks (DCBs)) from a predefined library (DCB library) that are selected and graphically linked by dragging and dropping

- **Libraries**
  Libraries allow modular software development and provide you with user-defined data types, functions and function blocks that you can use from all SIMOTION devices.
  Libraries can be written in all programming languages and used in all program sources (e.g. ST units, MCC units).

All programming languages enable you to structure the application using programs, functions and function blocks so that the application is manageable and re-usable.

To test the programs, there are comprehensive test functions available to you with program status and breakpoints in all languages. You can visualize and test your programs online.

For further information on programming languages, please refer to the respective manuals.

3.2.12.10 Result in the sample project

You have created the programs required for program-controlled traversing of the axis in the sample project of Getting Started.

3.2.13 Configure execution system

3.2.13.1 Overview

Aim of Getting Started

In this part of Getting Started, you get to know the SIMOTION execution system.

You assign the sample project programs to the tasks of the execution system. You thus transfer the programs from the SIMOTION SCOUT engineering system to the SIMOTION runtime system. The programs control the system as soon as the operating mode of the SIMOTION device is switched to RUN.

Preconditions

You have created the programs pos_axis, technology_fault, peripheral_fault, and LAD_1 for the sample project, see the section Programming the SIMOTION application (Page 182).

Execution levels and tasks

Execution levels define the chronological sequence of tasks in the execution system. A level can contain several tasks. The tasks provide the framework for program execution. A task may comprise several programs. By assigning the created programs to the tasks, you can, for example, define the priority, the time frame or the order in which the programs are to be executed.

3.2.13.2 Assign programs to tasks

Below you will assign the sample project programs to the tasks of the execution system.

Table 3-22 Assignment of programs to tasks

Programs of the sample project	Tasks
pos_axis	MotionTask_1
lad_1	BackgroundTask
technology_fault	TechnologicalFaultTask
peripheral_fault	PeripheralFaultTask

You assign programs to tasks in the execution system as follows

1. Double-click in the project navigator under the SIMOTION device on **EXECUTION SYSTEM**.
   The **EXECUTION SYSTEM** window appears in the working area.
2. Assign the MCC program **motion.pos_axis** to the task **MotionTask_1**:
   – In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > MotionTasks > MotionTask_1**.

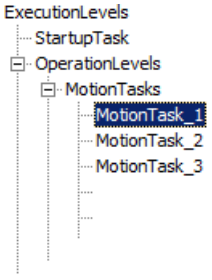


Figure 3-109 Execution system, tree view of the execution levels and tasks

The **MotionTasks** window appears on the right of the working area. The programs **pos_axis** and **LAD_1** as well as auxiliary programs of the **fault** unit are visible on the **Program assignment** tab under **Programs**.

– Select the MCC program **motion.pos_axis** and click the **>>** button. The program is displayed under **Programs used**. It is thus assigned to the task **MotionTask_1**.

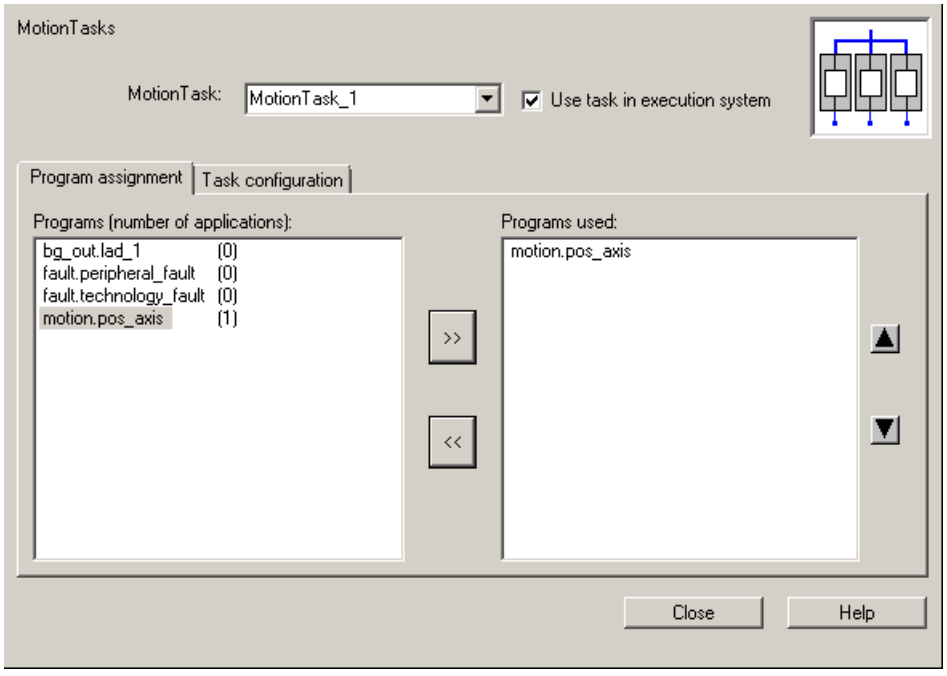


Figure 3-110 Execution system, MotionTasks window

The assignment is visible in the tree of the execution system. The program **motion.pos_axis** appears below the branch **MotionTask_1**.

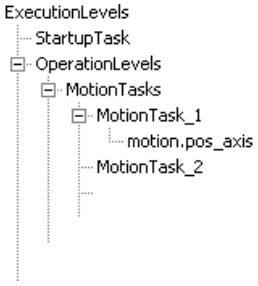


Figure 3-111 Execution system, MotionTask_1 with assigned program pos_axis

– Activate the checkbox **Activation after StartupTask** in the **Task configuration** tab. This executes the MCC program immediately after the SIMOTION device is started. If this checkbox is not activated, the program must be started explicitly by the call from another program that is assigned to the StartupTask or another active task.

3. Assign the LAD program **bg_out.kop_1** to the task **BackgroundTask**:

   – In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > BackgroundTask**. Assign the LAD program **bg_out.kop_1** to this task.

4. Assign the error handling routines:
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > TechnologicalFaultTask**. Assign the MCC program **fault.technology_fault** to this task.
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > PeripheralFaultTask**. Assign the MCC program **fault.peripheral_fault** to this task.
5. Click **Close**. Confirm with **Yes** if you are prompted to save.
   The execution system is configured.

Loading the configured execution system to the target system

- Save and compile the project.
- Go online.
- Download the sample project with the configured execution system to the target system.

See also

Download the project to the target system (Page 159)

3.2.13.3 Result in the sample project

Configuring the axis control is thus completed:

- You have set up an axis in the sample project.
- You have created a program for traversing an axis, as well as other programs that are necessary for operation.
- You have assigned these programs to the tasks of the SIMOTION runtime system.

In the following configuring steps, you will start the axis control on the SIMOTION device and monitor the program-controlled axis motion.

3.2.14 Starting and stopping the system

3.2.14.1 Overview

Aim of Getting Started

You switch the SIMOTION CPU of the sample project to RUN mode to start execution of the **pos_axis** program. You can see on the hardware that the axis rotates twice for approximately 4 seconds. After execution of the program, switch back to the STOP state.

You switch the operating mode in the dialog **Control Operating State**.

Preconditions

- You have created the programs pos_axis, technology_fault, peripheral_fault and LAD_1 for the sample project, and you have assigned them to the tasks of the SIMOTION execution system, see the sections Programming the SIMOTION application (Page 182) and Configure execution system (Page 222).
- The project has been compiled and downloaded to the target system, see the section Download the project to the target system (Page 159).
- SIMOTION SCOUT is in online mode.

3.2.14.2 RUN and STOP operating states

Operating states

In the SIMOTION SCOUT dialog **Control Operating State**, you can switch a SIMOTION CPU to RUN or STOP mode.

RUN mode

SIMOTION executes the user program and the associated system services:

- Read process image input
- Execution of the user programs assigned to the execution system
- Write process image output

The technology packages are active in this state. They can execute commands from the user program.

STOP mode

SIMOTION does not process any user program.

- It is possible to load a complete user program.
- All system services (communication, etc.) are active.
- The I/O modules are in a secure state. (This means, for example, the digital outputs are at "LOW level" and the analog outputs are de-energized)
- The technology packages are inactive, that is, all enables are deleted. No axis motions can be executed.

You can find the full description of the operating states in the online help under **SIMOTION device: Operating state**, or in the *SIMOTION SCOUT* Configuration Manual.

3.2.14.3 Mode selector switch on the software side and the hardware side

Control Operating State dialog

The menu command **Target system > Control operating state** or the button **Control operating state** in the toolbar of the workbench open the dialog **Control Operating State**.

The dialog lists all configured CPUs. The **State** column shows the current operating mode. The assigned switches in the **Control** column switch the CPU to the RUN or STOP mode. The switches are deactivated when the CPU is in offline mode.

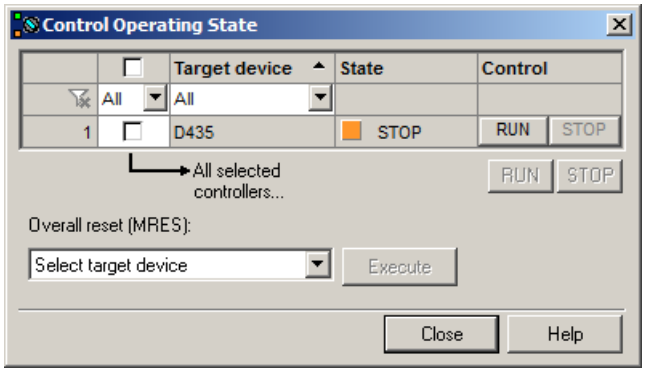


Figure 3-112 Control Operating State dialog

Note

As of SIMOTION V4.4, the **Control Operating State** dialog represents all configured SIMOTION CPUs. The previous dialog had to be opened individually for each CPU.

You can find a detailed description of the dialog in the *SIMOTION SCOUT* Configuration Manual.

Priority of the mode selector switch on the SIMOTION device

The setting of the mode selector switch on the SIMOTION device has priority. SIMOTION SCOUT can only switch a SIMOTION device to RUN mode if the mode selector switch on the device is set to 0 (RUN).

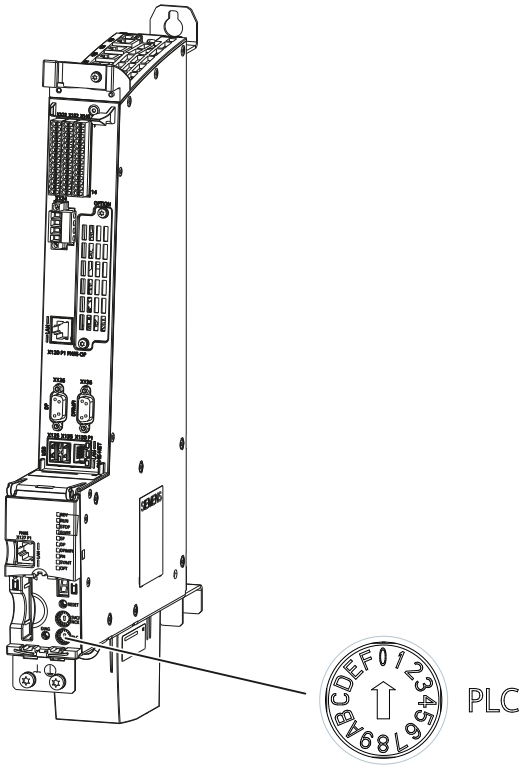


Figure 3-113 Mode selector switch of the SIMOTION D435-2, switch position 0 (RUN)

You can find detailed information on the mode selector switch of the SIMOTION D4x5-2 devices in the Manuals and Commissioning Manuals, as well as in the online help.

3.2.14.4 Start program control of the sample project

You start the axis control of the sample project on the SIMOTION device as follows

⚠ **WARNING**
Danger to life through unexpected machine movement
Make sure this presents no hazard to personnel or property.

1. Open the **Control Operating Mode** dialog. To do so, click **Control Operating Mode** in the toolbar.

2. Switch the SIMOTION CPU of the sample project to the RUN mode. To do so, click the assigned **RUN** switch in the **Control** column of the **Control Operating Mode** dialog.
   The axis starts to rotate immediately. The axis returns to a standstill after approximately 8 seconds.
   The SIMOTION device remains in the RUN mode after execution of the **pos_axis** program.
3. Switch the SIMOTION device to STOP mode. To do so, click the assigned **STOP** switch in the **Control** column.

Note

If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

3.2.15 Monitor the application

3.2.15.1 Overview

Aim of Getting Started

In this part of Getting Started, you monitor the program-controlled axis motion.

- You monitor program execution.
- You monitor the values in the symbol browser.
- You compile certain values in a watch table.
- You record the course of the axis motion with the trace.

Preconditions

- You have created the programs pos_axis, technology_fault, peripheral_fault and LAD_1 for the sample project, and you have assigned them to the tasks of the SIMOTION execution system, see the section Configure execution system (Page 222).
- The project has been compiled and downloaded to the target system, see the section Download the project to the target system (Page 159).

- SIMOTION SCOUT is in online mode.
- Operating mode of the SIMOTION device:
  - The SIMOTION device of the sample project has been switched to STOP mode on the software side. The SIMOTION SCOUT mode selector switch, **Control Operating State** dialog, shows the STOP mode.
  - The SIMOTION device of the sample project has been switched to RUN mode on the hardware side. The mode selector switch on the SIMOTION device is at position **0** (RUN).

3.2.15.2 Monitoring program execution

SIMOTION SCOUT provides several functions for monitoring program execution.

In the sample project, you use the function **Monitor**. With this function, you can track execution of the command blocks in an MCC chart. The currently executed command block is shown against a yellow background in the chart. The marking runs at the same speed as the actual program execution. The marking is therefore only visible with the "slow" commands that wait for the machine to implement the command.

You monitor the execution of the commands in the MCC chart pos_axis as follows

1. Open the **Control Operating State** dialog.

   The SIMOTION device must be in **STOP** mode.
2. Open the MCC chart **pos_axis**.
3. Switch the monitoring function on:
   - To do so, select the command **MCC chart > Monitor** in the menu bar of the workbench. The function is switched on if a checkmark is visible next to the menu item.
   - Or click in the **MCC editor** toolbar on **Monitor**.

4. Start program control: Switch the SIMOTION device to **RUN** mode in the **Control Operating State** dialog.
   The **pos_axis** program is run through once. The currently executed command block is shown against a yellow background.

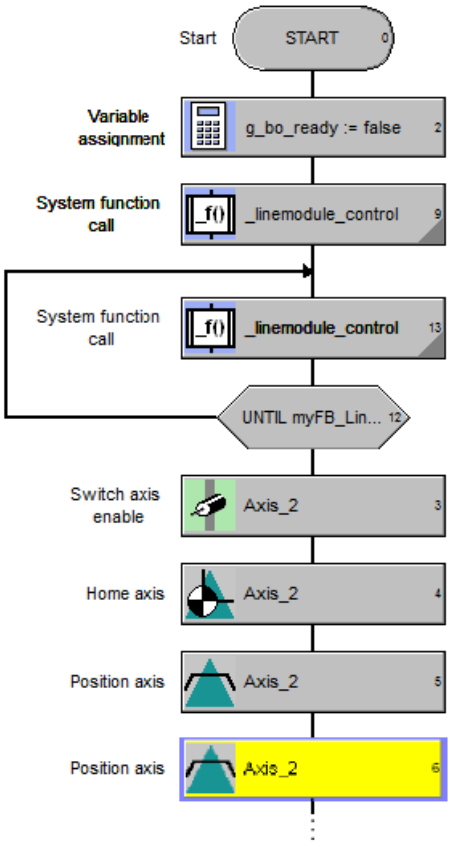


Figure 3-114 Monitor the program execution of the MCC chart pos_axis

5. Switch the SIMOTION device to **STOP** mode in the **Control Operating State** dialog.

Further information

You can find further information on program monitoring in the online help under **Monitoring program execution**.

3.2.15.3 Monitoring variables

In the symbol browser, you can monitor variables (read out status value) or assign values to them (assign control values).

For the sample project, you monitor the actual position of the axis during the program run.

Monitor variables in the symbol browser

1. In the **AXES** folder of the project navigator, select the axis created in the sample project. The system variables and configuration data of the axis are displayed on the **Symbol browser** tab of the detail view.
2. Open the system variable **positioningstate.actualposition** (actual position of the axis) in the symbol browser.
   You find the variable as follows:
   - Filter the list: You can specify a filter criterion in the filter line of the symbol browser. The last 5 criteria are saved and can be selected for re-use.
     Enter a suitable filter term, e.g.**positioningstate**, in the filter line. Press **RETURN** to confirm.
   - Search for variable: As an alternative to the filter function, you can search for the variable. Select the menu command **Edit > Find**. Enter a suitable search term in the **Find** dialog, e.g. **positioningstate**. Click **Find next**.

   The actual position of the axis is displayed in the **Status value** column of the symbol browser.
3. Start the program control via the dialog box **Control Operating State**.
   The program runs once. The changing values of the actual axis position are displayed in the symbol browser.
   The SIMOTION device is in the RUN state after the program has run.
4. Switch the SIMOTION device to STOP mode in the **Control Operating State** dialog.

Monitoring variables in watch table

Different variables, e.g. of several devices, can be combined into a table to be monitored.

To include a variable in a watch table, you must first create a watch table.

Proceed as follows:

1. Double-click in the project navigator under **MONITOR** on the entry **Insert watch table** and confirm your configuration.
2. Right-click in the symbol browser on the variable that you want to add to the watch table. Choose the **Add to watch table** command from the shortcut menu.
   The selected variable will be displayed in the table. In this way, you can add further variables to the watch table and monitor them.

You open a watch table as follows

You will find all created watch tables in the **MONITOR** folder of the project navigator. Double-click a watch table to open it.

You can find detailed information on this topic in the online help under **Working with the SCOUT Workbench > Working with lists > Watch table**.

3.2.15.4 Recording signals with the trace

Trace

You can use the trace to record and save signal characteristics and variable values. The recorded data can be used, for example, for diagnostics purposes in machine motion sequences, and for troubleshooting in user programs.

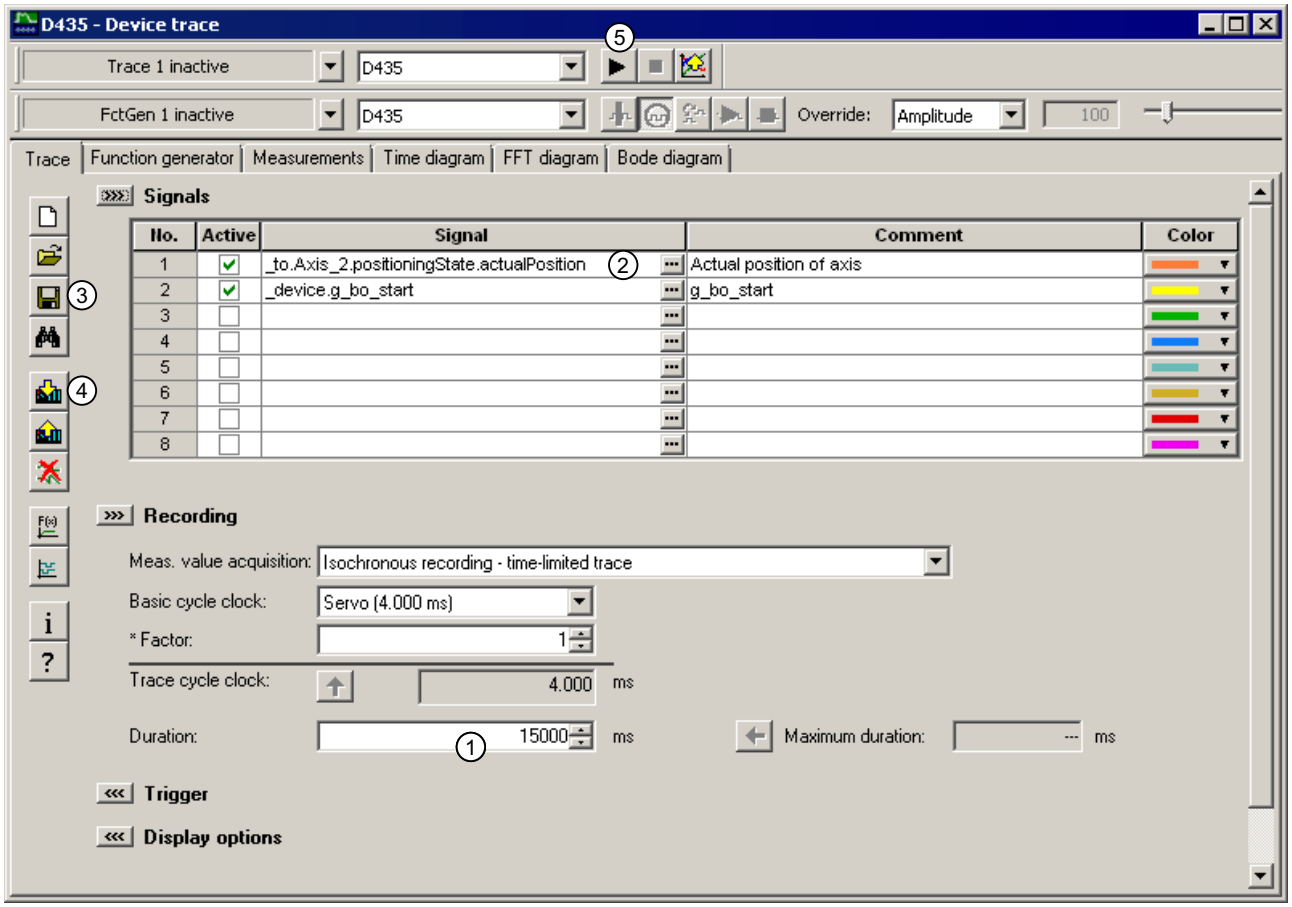
For the sample program, you record the actual position of the axis over time and represent it in the diagram.

Working with the trace

You call the trace as follows

1. Select the SIMOTION device in the project navigator.
2. Select the menu command **Target system > Device trace** or click the **Device trace function generator** button in the **Trace toolbar**.

The **Device trace** window appears in the working area.



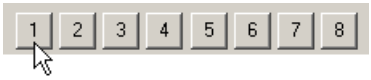
① ... ⑤ Reference is made to the circled digits in the text below.

Figure 3-115 Device trace window

You parameterize the trace for recording as follows

1. In the **Duration** field ① of the **Trace** tab, enter the recording duration 15000 ms.
2. Click the button ... ② in line **No 1** of the **Signals** table.
   The **Trace Signal Selection** window appears.

3. In the tree, select the branch **Sample_1 > (**SIMOTION device, e.g. **D435) > TO > Axis_2 > positioningstate**.
4. Select the system variable **actualPosition** in the variable table.
5. Click the button **1** to accept the system variable for channel 1.



The variable is displayed under **Signal name**.

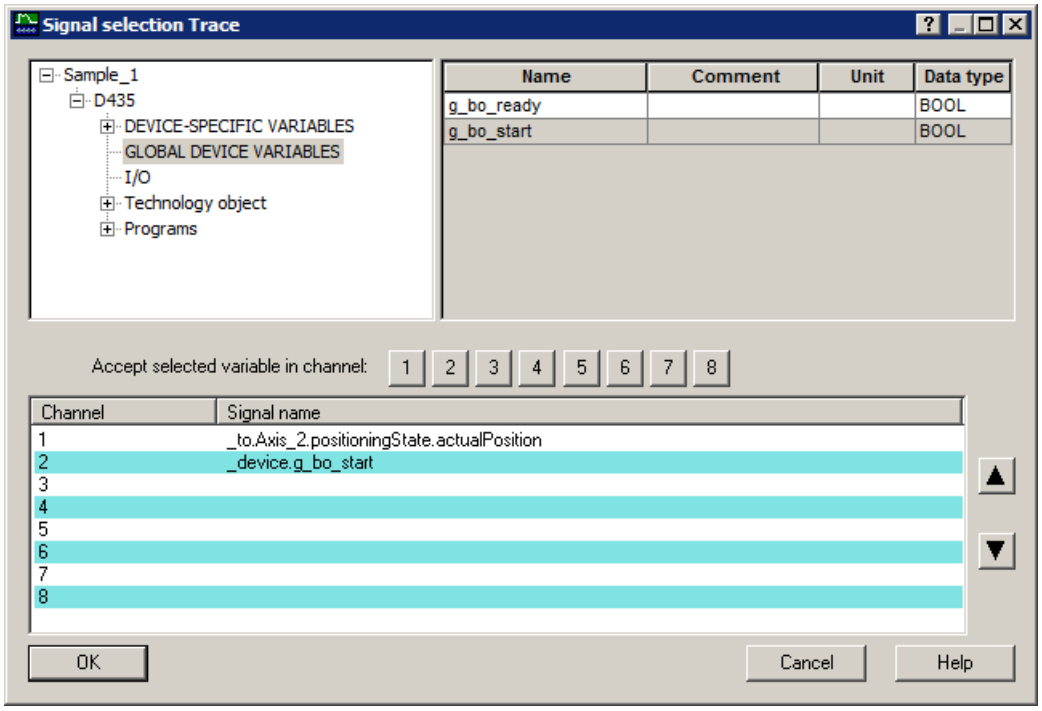


Figure 3-116 Trace signal selection

6. Repeat the channel assignment for the variable **g_bo_start**:
   Select the branch **(Project) > (SIMOTION device) >GLOBAL DEVICE VARIABLES**. Select the system variable **g_bo_start** in the variable table. Click the button **2**.

Note

You can also drag & drop the variables from the symbol browser or the watch table to the signal field of the Trace dialog. Procedure for dragging & dropping, see the section Variable assignment g_bo_ready:=false / g_bo_start:=true (Page 194).

7. Confirm with **OK**.
   The **Trace Signal Selection** window is closed.
   The trace is now parameterized for recording.

You save the parameterization of the trace as follows

The trace parameterization is not saved in the project data. When you close the project, the trace parameterization is deleted.

To save the parameterization, click the **Accept in catalog** button ③ on the **Trace** tab. In the **Catalog entry** field, enter the name under which the setting/parameterization is to be saved in the catalog of the trace.

You record with the trace as follows

1. Go online.
2. Download the parameterization of the trace to the target system:
   – Click the **Download parameterization** button ④ on the **Trace** tab.
   
   – Click **OK** to confirm the dialog that appears following successful downloading.
3. Open the **Control Operating State** dialog.
   
   The SIMOTION device must be in **STOP** mode.
4. Open the **Time diagram** tab in the **Device trace** window.

5. Start recording of the trace:
   - Click **Trace start**⑤ in the **Device trace** window.

   - Then immediately switch the SIMOTION device to RUN mode in the **Control Operating State** dialog.

   The program is started. The actual position of the axis is recorded and represented in the time diagram. After expiry of the recording duration, the signal profile of the actual position is displayed.

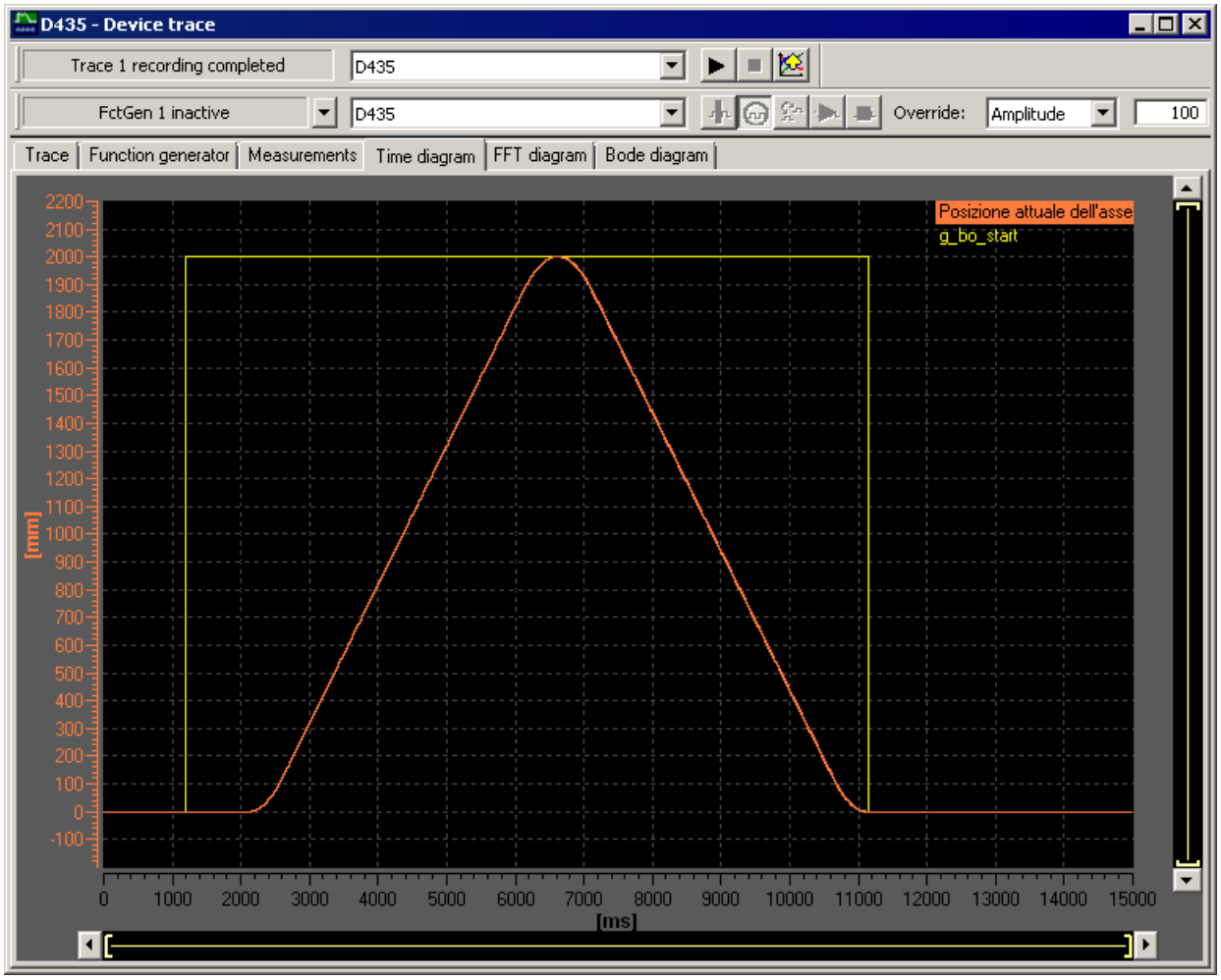


Figure 3-117 Time diagram of the axis

6. If the recorded curve is only partially displayed, select the menu item **Auto-scaling** in the context menu of the time diagram.
7. Switch the SIMOTION device to STOP mode in the **Control Operating State** dialog box.

Carrying out several measurements

You can carry out several measurements. They are shown on the **Measurements** tab and can be displayed in the diagram.

You can save and re-open recorded measurements for documentation purposes.

Further diagnostic functions

You can find a detailed overview of the extensive service and diagnostics options in the online help under **Diagnostics**, as well as in the product information *SIMOTION SCOUT Overview of service and diagnostics options*.

3.2.15.5 Result in the sample project

Creation of the trace time diagram concludes Getting Started.

We recommend that you continue to familiarize yourself with SIMOTION SCOUT using the sample projects of the Utilities & Applications.

You can find information on the Utilities & Applications in the section Utilities & applications (Page 145).

3.2.16 ESD directives

3.2.16.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

3.2.16.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

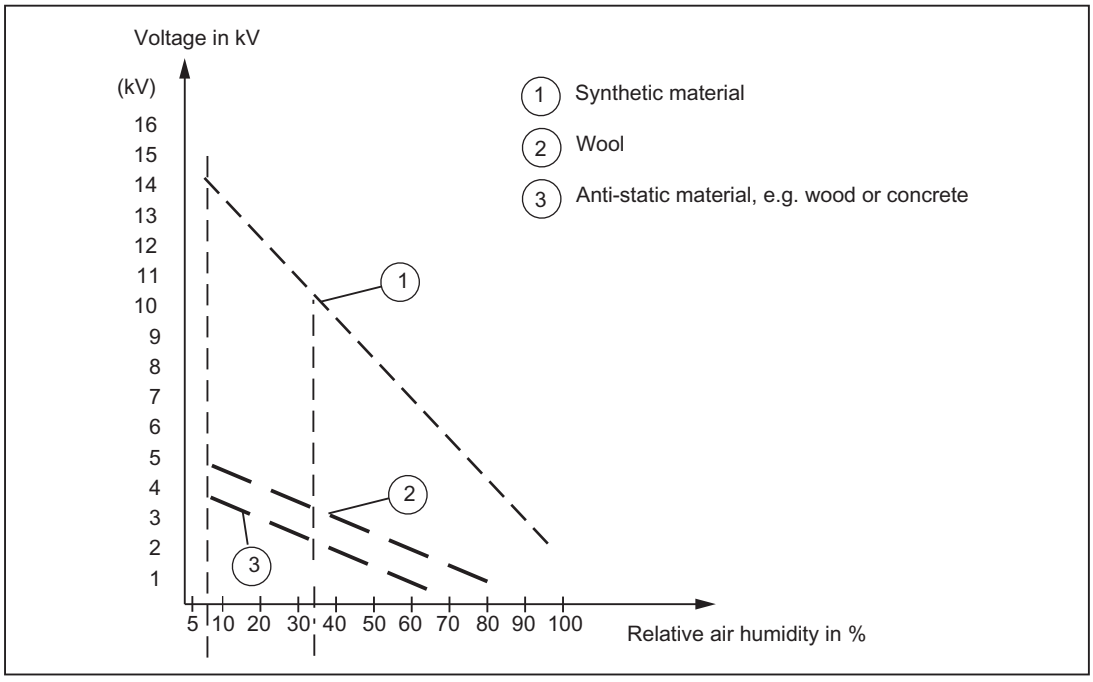


Figure 3-118 Electrostatic voltage that can accumulate on operating personnel

3.2.16.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

3.3 SIMOTION SCOUT

3.3.1 Preface

3.3.1.1 Validity

This **document** is part of the **Engineering System Handling documentation package**.

Scope of validity

This manual is valid for SIMOTION SCOUT in conjunction with the SIMOTION CamTool option package for product version V5.4.

3.3.1.2 Sections in this manual

The following is a list of sections included in this manual along with a description of the information presented in each section.

- Introduction
  This chapter contains an overview of the SIMOTION SCOUT Engineering System.
- Installation
  This chapter contains the system requirements for SIMOTION SCOUT, describes the procedure for installing and uninstalling it, and provides important information on the communications link to the SIMOTION device.
- User interface
  This chapter contains an overview of the SIMOTION SCOUT Workbench. It also provides notes concerning the language setting and the use of online help.
- Configuring/parameterizing
  This chapter describes the basic steps for operating SIMOTION SCOUT.
- Target system
  This chapter contains information on controlling the target system. You also learn how you can control the operating state with SIMOTION SCOUT and how you can upload data to the target system.
- Diagnostics
  This chapter contains information about which diagnostic functions are available and how these are operated.
- Service with SIMOTION SCOUT
  This chapter contains information about the service.
- Siemens SIMOTION Diagnostics
  This chapter contains information on how you can use the **Siemens Simotion Diagnostics** diagnostic tool for troubleshooting.
- Configuring a further connection (such as HMI)
  This chapter contains information on how to configure the HMI connection among other things.

- Product combinations
  This chapter describes topics such as compatibility and storage media as well as STEP 7, NetPro, Drive ES, HMI, and other interfaces.
- Technical specifications
  This chapter contains information about the quantity structure and the storage capacity requirement.
- Appendix
  The appendix contains the scripts for SIOMTION and a sample program for an axis configuration in SIOMOTION SCOUT.
- Index

3.3.1.3 SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

3.3.1.4 Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

3.3.2 Fundamental safety instructions

3.3.2.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

3.3.2.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

3.3.2.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

3.3.2.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

3.3.3 Introduction

3.3.3.1 Content of the configuring manual

The SIMOTION SCOUT Configuration Manual is a general description of the software. Not all available software functions are described in this document. All detailed, subject-specific information can be found in the context-sensitive online help and the corresponding documentation.

Important notes and information on the SIMOTION Motion Control system are contained in the following catalog:

- SIMOTION, SINAMICS S120 and Motors for Production Machines, PM 21 Catalog

3.3.3.2 SIMOTION SCOUT engineering system

Introduction

While the Motion Control system SIMOTION provides a wide variety of preprogrammed functions, it is also parameterizable and programmable for individual requirements. High-performance tools, which provide optimum support and ease of use for the necessary engineering steps, are required for this.

The SIMOTION SCOUT Engineering System is the environment for the uniform automation of production machines with SIMOTION and integrates into the SIMATIC environment in accordance with TIA (Totally Integrated Automation).

SIMOTION SCOUT provides a uniform, function-oriented view for your automation task, and at the same time it is very easy to use.

The possible SIMOTION applications range from simple, parameterizable, speed-controlled single axes through to complex, mechatronically-coupled and programmable multi-axis machines. Therefore, SIMOTION SCOUT provides views adapted to the task and can be expanded with additional tools (e.g. tool for the graphic creation of cams).

SIMOTION SCOUT is the engineering system for SIMOTION. It is integrated into STEP 7 and provides all the required tools for the following functionalities:

- Configuration
- Parameter assignment
- Programming
- Test
- Diagnostics

The following tasks are graphically supported with operator guidance:

- Creation of the hardware and network configuration
- Creation, configuration and parameter assignment of technology objects such as axes, output cams and cams.

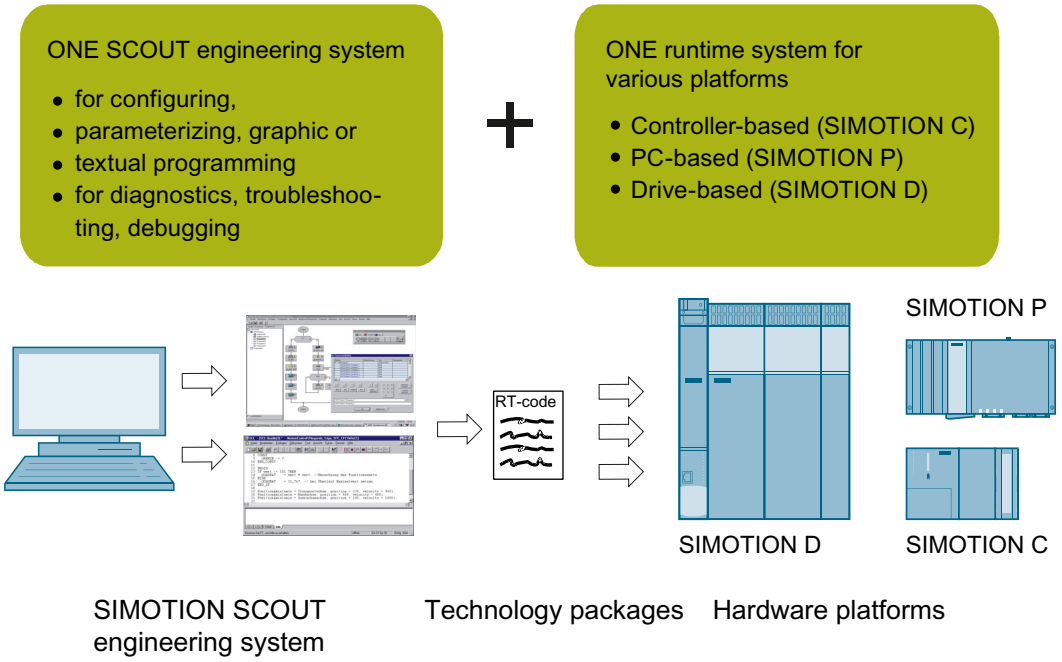


Figure 3-119 SIMOTION system overview

The automation topology is defined in the first engineering steps. The hardware and network configuration is created by parameterizing the required components and networks.

3.3.3.3 SIMOTION hardware platforms

To meet the complex requirements of machine construction, SIMOTION offers three hardware variants with different performance, packaging formats and expandability options. The basic system characteristics, like the engineering, are identical.

SIMOTION D (Drive-based)

SIMOTION D is a compact, drive-based version of SIMOTION based on the SINAMICS S120 drives family. For SIMOTION D, the SIMOTION runtime environment and the SINAMICS drive software run concurrently on the controller hardware in the SINAMICS S120 packaging format.

SIMOTION P (PC-based)

SIMOTION P is a PC-based, open motion control system from SIMOTION. Control, motion control, and HMI functions are executed together with standard PC applications on the SIMOTION P hardware platform.

SIMOTION P combines the openness of the Windows operating system with the real-time capability of SIMOTION P Runtime.

SIMOTION C (Controller-based)

SIMOTION C is the modular controller variant in the tried and trusted packaging system of the SIMATIC S7-300 with its very varied expandability options on the I/O bus. SIMOTION C240 high-performance motion controllers are available for control functions and motion control tasks. The integrated interface for four analog coupled drives makes the SIMOTION C particularly suitable for compact applications with the control of analog electrical drives and the operation of hydraulic axes. SIMOTION C also supports operation of four stepper motors at these interfaces. SIMOTION C240 PN offers a PROFINET interface instead of the encoder and drive interfaces.

3.3.3.4 Programming languages

Programming languages in SIMOTION SCOUT

SIMOTION provides different programming languages for the solution of Motion Control tasks, control logic, arithmetic calculations, etc. During runtime, the selected programming language has no effect – except in the different displays when debugging. You can create user applications in different programming languages and use them jointly in a project.

The following programming languages are available in SIMOTION SCOUT:

- **Motion Control Chart (MCC)**
  Graphical programming as a flow chart.
  In particular, for sequential tasks with a high level of motion control functionality.
- **Ladder Logic / Function Block Diagram (LAD/ FBD)**
  Graphical Programming as Ladder Logic / Function Block Diagram, supplemented by Motion Control Functions via PLCopen Function Blocks.
  In particular, for cyclic tasks with a high logic proportion.
- **Structured Text (ST)**
  textual programming in a high-level language.
  As the base language of the SIMOTION system, ST supports all system features and functions of the technology packages and is thus suitable for all tasks.
- **Drive Control Chart (DCC)**
  In many applications, the control of the drive system requires a linking logic which combines multiple statuses (e.g. entry control, system status) into one control signal (e.g. ON command).
  As well as logical links, drive systems are increasingly calling for arithmetic operations and/or saving elements. This kind of functionality is available on drive objects of the SINAMICS drive system and the SIMOTION control system in the form of a Drive Control Chart (DCC).
  With the Drive Control Chart Editor (DCC Editor) based on CFC, SIMOTION controllers and SINAMICS drives can be configured graphically.
  For further information, see Section DCC programming system (Page 443).

Sources (units)

Programs are created in program containers, the so-called sources (units). They are compiled in the engineering system. Any errors or warnings that occur during compilation are output in the diagnostics window. Sources compiled without error can then be loaded into the associated controller.

A source contains any number of programs, functions, function blocks and classes. Each executable part of a source (program, function, function block, class) is called a POU (Program Organization Unit).

A source is divided into an interface and an implementation section.

Interface section

All parts exported by the source are defined in the interface section. Other sources and external components (e.g. HMI systems) can access these parts. These include user-defined data types, data (variables and constants) as well as names of programs, functions, and function blocks.

Implementation section

The data types and data defined in the implementation section are global throughout the source and can be used by all POUs. The implementation section also contains the program code of the POUs. Any POUs not specified in the interface section can only be used within the source.

Note

Source concept

The source concept with encapsulation of code and data allows you to structure applications. For example, the functionality of an entire machine module with a defined external interface can be implemented in a single source.

Tasks

Programs are processed in tasks. A task is a job which is executed in a certain chronological sequence. The advantage of the task system (execution system) is that processes appended to the appropriate task levels can run simultaneously.

The SIMOTION Motion Control system uses high-performance CPUs on which a real-time operating system - suitable for fast control processes - is implemented. Each task is allocated a slice of the computing time. The organization of the task executions is performed by the operating system. A differentiation is made between user and system tasks that are independent of one another.

Additional references

For detailed information on the programming languages and the execution system, refer to the SIMOTION SCOUT online help and the appropriate programming and operating manuals.

Motion Control Chart (MCC)

MCC (Motion Control Chart) is a "flow diagram language" that graphically formulates the process sequences in production machines in a simple manner. The result is one or more flow diagrams, comprising MCC blocks that describe the chronological sequence of the individual machine actions. Due to its special means of expression, an MCC is ideally suited to programming sequential processes.

Motion Control Chart supports the simple description of the motion sequences of machines using powerful motion control commands, such as reference axis, position axis, synchronize or desynchronize cam, and many more.

To control the machine sequence, commands are available for awaiting conditions and for formulating computations, as well as for programming various control structures, such as polling (IF), case determination (CASE) and loops (FOR, WHILE, UNTIL). Several MCC programs may be created to describe different process situations. For example, you can create one MCC program to bring the machine to a defined initial state when it is switched on, a second MCC program for the normal production sequence, and a third MCC program to specify what the machine has to do in the event of a fault.

All MCC blocks – a selection of the most important SIMOTION functions – are available in tool bars. They are grouped according to function and are automatically inserted in the flow diagram at the marked point by means of a click. A click on the individual elements opens specific dialogs for parameterization. Obviously, you can also add your own comments for the further documentation of the process sequence. Functions from the SIMOTION command library that are not individually offered as MCC blocks can be used in an MCC program by means of a special command.

Performance features:

- Easy-to-use due to graphical illustration in the form of flow diagrams
- Hierarchical command library for motion control, PLC, and technology functions
- Control structures (IF, WHILE, CASE, etc.)
- Zooming for LAD, FBD and ST
- Subroutine calls (FB/FC, programs, methods (OOP))
- Structuring based on command module generation, i.e. combination of command sequences to form a module command
- User-friendly debug functions for online test and diagnosis, for example, single-step, program status or breakpoints for easier troubleshooting (debugging)
- Monitor, trace

Note

Implicit conversion to ST

When being compiled, programs written in MCC are implicitly converted to ST programs and then compiled.

You can export the intermediate result as an ST and use it as a basis for your own ST programs.

Additional references

Detailed information can be found in the SIMOTION SCOUT online help and in the SIMOTION MCC (Motion Control Chart) Programming and Operating Manual.

Ladder Logic / Function Block Diagram (LAD/FBD)

LAD/FBD is a graphical programming language and is available for Ladder Logic / Function Block Diagram. The statement syntax corresponds to a circuit diagram (LAD) or a function block diagram (FBD). LAD/FBD enable simple tracking of the signal flow between power rails via inputs, outputs, and operations. LAD and FBD programs are usually suitable for application in cyclic tasks (in particular, BackgroundTask).

LAD/FBD programs consist of elements and boxes that are graphically connected to networks. Their operations work mostly according to the rules of Boolean logic or simple arithmetic expressions and equations. Therefore, they are suitable only for control-relevant programs or also for motion control tasks by using the PLCopen blocks.

Functions, function blocks and programs can be programmed in LAD/FBD. A source can contain several LAD and FBD blocks. Only one POU can be implemented in an LAD or FBD block.

LAD/FBD also include commands for SIMOTION system control using standard logic functions. These commands are added from the command library. Motion Control tasks are preferably programmed with PLCopen blocks. Blocks which have been programmed in other SIMOTION languages can be called. User-friendly functions such as "on the fly" variable declarations or automatic syntax checks are available when programming in LAD or FBD. It is possible to switch over between LAD and FBD in the editor at any time. A program can therefore be viewed and processed in either LAD or FBD.

The following user-friendly debug functions are available for online testing and diagnostics:

- Program status
- Breakpoints

Note

Direct editing of motion commands is not recommended. Instead, it is better to use the PLCopen blocks. These blocks are designed for integration in logic-oriented programs.

Additional references

Detailed information can be found in the SIMOTION SCOUT online help and in the SIMOTION LAD/FBD Programming and Operating Manual.

Structured Text (ST)

Structured Text

ST is a high-level, PASCAL-based programming language. ST is based on the IEC 61131-3 standard. This standard harmonizes programming languages for programmable logic controllers (PLC).

The basic command scope is sufficient for the implementation of everything related to data management, arithmetic functions, control structures and I/O access. The addition of technology packages for Motion Control expands the scope of commands by other comprehensive, extremely flexible Motion Control commands.

In addition, applications can be subdivided into any number of sections. Such a section might be a program allocated to a runtime level, an instantiatable function block with its own memory, or a function without its own memory. In this case, the function blocks and functions are not allocated to a runtime level, but are instead called in programs.

Note

As of SIMOTION SCOUT V4.5, ST also supports object-oriented programming. Further information can be found in the SIMOTION SCOUT online help and in the SIMOTION ST (Structured Text) Programming and Operating Manual.

Performance features:

- Motion Control, PLC and technology functions in a single language
- Well-structured programs with comment capability
- Powerful editor functions, such as:
  - Syntax coloring
  - Automatic indenting
  - Automatic completion
  - Bookmarks
  - Fold (show and hide blocks)
  - Displaying sets of parentheses
  - Select text, e.g. by column
  - Using the command library
- Convenient debug functions for online testing and diagnostics, e.g. display of up-to-date variable content of the code sequence selected in the editor (program status) and breakpoints

Additional references

Detailed information can be found in the SIMOTION SCOUT online help and in the SIMOTION ST (Structured Text) Programming and Operating Manual.

3.3.3.5 CamEdit cam editor

CamEdit can be used to describe curves by means of either interpolation points or segments. A combination is not possible. If the curve is to be created from segments using polynomials, SIMOTION SCOUT provides the VDI wizard to assist in creation of the curve. Cam geometries are created in offline mode.

Information on the graphical creation of cams can be found in the CamTool section.

3.3.3.6 CamTool options package

SIMOTION CamTool is a powerful, graphical editor for creating and optimizing cams.

SIMOTION CamTool can be used as an expansion package for SIMOTION SCOUT and is completely integrated in the SIMOTION SCOUT user interface.

Basic functions

SIMOTION CamTool provides the following basic functions:

- Insert and edit cams.
  Cams can be inserted in a SIMOTION SCOUT project using the SIMOTION CamTool. You can also edit a cam created with CamEdit using CamTool: Cams can also be imported from a text file or read from a SIMOTION device.
- Customize display of the cam in CamTool.
  In SIMOTION CamTool, you can show and hide diagrams, change display parameters of the axes and diagrams and adjust the lines and fonts. You can also display auxiliary lines in the diagram.
- Convert cams from SIMOTION CamTool to SIMOTION CamEdit.
  To edit a cam that is edited in SIMOTION CamTool using SIMOTION CamEdit, the cam needs to be converted.
- Export cams to a text file.
- Load cams into a SIMOTION device.

Additional references

Detailed information can be found in the SIMOTION SCOUT online help and in the SIMOTION CamTool Configuration Manual.

See also

CamEdit cam editor (Page 252)

3.3.3.7 Technology packages and technology objects

Technology packages in SIMOTION SCOUT

Technology packages combine software functions which are required for automation in mechanical engineering in various sectors. They are loaded into the controller during configuration and expand the basic functionality through additional system functions. The functions of the technology packages can be accessed easily during engineering in the SIMOTION SCOUT command library. Access to the technology package functions is provided by additional language commands and system variables. Programming of motional sequences is therefore simple and integrated.

The following standard technology packages are available for SIMOTION SCOUT:

CAM technology package

The SIMOTION CAM technology package contains the basic Motion Control technologies, such as drive axis, position axis, following axis, synchronous object, cam, output cam, cam track, and measuring input.

PATH technology package

The SIMOTION PATH technology package contains additionally the path interpolation technology.
Path interpolation generates the traversing profile for the path, calculates the path interpolation points in the interpolation cycle, and uses the kinematic transformation to derive the axis setpoints for the interpolation cycle points.

CAM_EXT technology package

The CAM_EXT technology package also contains objects for preparing technological data at the system level, e.g. addition object, formula object.

TControl technology package

The SIMOTION technology package for temperature control (TControl) provides temperature channels with extensive functions. These functions are also accessed via additional language commands and system variables.

DCBlib

The SIMOTION DCBlib technology package contains interconnectable DCC (Drive Control Chart) blocks for drive-related control functions.

More sector-specific technology packages are also available as separate products.

The loadable technology packages support the creation of technology objects (e.g. positioning and synchronous axis, cam tracks, external encoders) which can be accessed over system functions and system variables for use in every SIMOTION programming language.

Technology objects in SIMOTION SCOUT

SIMOTION SCOUT uses technology objects (TOs) to represent the functionality of axes, cams, output cams, etc. After creating a technology object (e.g. axis) and configuring it (e.g. as positioning axis), you can access it when programming with system functions.

Note

Missing licenses

All TOs outside the basic functionality (Motion Control Basic) must be licensed. Missing licenses are indicated by a flashing group error LED. The number and type of missing licenses is stated by the online diagnostics. They are also displayed during downloading.

Additional references

Detailed information is provided in the SIMOTION SCOUT online help and in the following documents:

- Function Manual: SIMOTION Basic Functions
- Function Manual: SIMOTION Motion Control, TO Axis, Electric/Hydraulic, TO External Encoder
- Function Manual: SIMOTION Motion Control, Synchronous Operation TO, TO Cam
- Function Manual: SIMOTION Motion Control, Supplementary Technology Objects
- Function Manual: SIMOTION Motion Control Output Cams and Measuring Inputs
- Function Manual: SIMOTION Motion Control, TO Path Object
- Function Manual: SIMOTION Motion Control, Basic Functions for Modular Machines

See also

Licensing of the runtime components (Page 361)

3.3.3.8 CLib Studio option package

SIMOTION CLib Studio

If required, further functions and function blocks can be created in user libraries (SIMOTION CLib Studio) by means of C/C++ programming in the Windows environment.

They can be used in all SIMOTION languages (MCC, LAD, FBD, ST).

That allows the creation of application-specific and high-performance function extensions as well as adaptations including know-how protection.

3.3.4 Installation

3.3.4.1 SCOUT and SCOUT Standalone system requirements

Minimum requirements of the system

The readme file on the SIMOTION SCOUT DVD contains information on SIMOTION SCOUT system requirements; alternatively, you can access this information after installation at **Start** -> **All Programs** -> **Siemens Automation** -> **Documentation** -> **Readmes -> [English]**.

Note

Simultaneous operation of SIMOTION SCOUT, Starter and SIMATIC S7-Technology on one PC is not intended and is not possible.

SIMATIC S7-Technology is integrated as of SIMOTION SCOUT V4.0.

3.3.4.2 Installing SIMOTION SCOUT

Installing SIMOTION SCOUT

SIMOTION SCOUT is available as software packages with full license and upgrade license.

Requirements:

- SIMATIC STEP 7 is installed.
- You are logged on to the operating system with administrator rights.

Note

Read the readme file and the important information on the SIMOTION SCOUT Add-Ons DVD contained in the SIMOTION SCOUT software packages.

To install SIMOTION SCOUT

1. Insert the DVD 1 with SIMOTION SCOUT into the CD-ROM drive.
2. Start Windows Explorer and select the CD-ROM drive.
3. Open the root directory on the DVD.
4. Double-click **Setup.exe**.
5. Now follow the instructions in the installation program.
   The installation program prompts you to insert or connect the supplied data medium that contains the authorization. You can install the authorization during this setup. Or install the authorization with the Automation License Manager after installing SIMOTION SCOUT.
   For information, see the section AUTOHOTSPOT.

6. If a restart of the PC is required during the installation, carry this out.
   After restart of the operating system, log on at least as main user.
7. After the installation:
   Restart the PC and log on at least as main user.

All users who are logged on as main user are now able to start and operate SIMOTION SCOUT.

Installing SIMOTION SCOUT Standalone

Prerequisite

- No SIMATIC STEP 7 may be installed or
- No previous version of SIMOTION SCOUT Standalone must be installed.
- You are logged on to the operating system with administrator rights.

Note

Read the readme file and important information on the Add-On CD supplied with SCOUT.

To install SIMOTION SCOUT Standalone

1. Insert DVD 1 with SIMOTION SCOUT Standalone into the CD-ROM drive.
2. Start Windows Explorer and select the CD-ROM drive.
3. Open the root directory on the DVD.
4. Double-click **Setup.exe**.
5. Now follow the instructions in the installation program.
   The installation program prompts you to insert or connect the supplied data medium that contains the authorization. You can install the authorization during this setup. Or install the authorization with the Automation License Manager after installing SIMOTION SCOUT. Information on this can be found in the section titled "To install the authorization".
6. A restart of the PC is required during the installation, carry this out. You are requested to insert the CD"/".
   After the restart of the operating system, log on at least as main user.
7. After the installation:
   Restart the PC and log on at least as main user.

All users who are logged on as main user are now able to start and operate SIMOTION SCOUT.

SINAMICS Support Package (SSP)

You can use a SINAMICS Support Package (SSP) to upgrade the version of the drive units on a STARTER integrated into SIMOTION SCOUT.

This permits the use of new functions which only become available with new drive unit FW versions.

SSPs are available when installing SCOUT or can be installed at a later time.

In this regard, the following SSPs are relevant for SIMOTION SCOUT:

- "SINAMICS" SSP for upgrading single drive units (e.g. CU3xx)
- "SIMOTION SINAMICS Integrated" SSP for upgrading the SINAMICS drives integrated into SIMOTION D.

The readme files for the relevant SSP contain detailed information regarding installation.

3.3.4.3 Uninstalling SIMOTION SCOUT

Requirements:

You are logged on to the operating system with administrator rights.

Note

Note that the SIMATIC STEP 7 software must be uninstalled separately.

To uninstall SIMOTION SCOUT from the hard disk

1. Open the Windows **Control Panel**. Go to the **Programs and Functions** page (Windows 7).
2. Select **SIMOTION SCOUT Vx.x.x.x** and click **Uninstall/change** (Windows 7). Follow the instructions.
3. After the uninstall has completed, restart the PC.

Uninstalling SIMOTION SCOUT Standalone

Uninstall SIMOTION SCOUT Standalone as described in the "How to uninstall SIMOTION SCOUT from the hard disk" section.

3.3.4.4 Licenses

To install the authorization

Installing the authorization for SIMOTION SCOUT

A data medium containing the authorization is supplied along with the product CD so that you can use SIMOTION SCOUT. This contains the license key for the SIMOTION SCOUT Engineering System.

The authorization for SIMOTION SCOUT and SIMOTION SCOUT Standalone can be installed as follows:

1. Connect or insert the data medium containing the authorization and the license key.
2. Start the Automation License Manager:
   - In the start menu, **Start > All programs > Siemens Automation > Automation License Manager** or
   - Double-click the **Automation License Manager** icon
3. In the navigation area (left-hand window), select the drive where the data medium containing the authorization is located. The license key is displayed in the right-hand window.
4. Mark the license key and drag this with drag-and-drop to the target drive.
5. Exit the Automation License Manager.
6. Remove the data medium containing the authorization.

Note

Information on operating the Automation License Manager and transferring the license key can be found in the online help for the Automation License Manager.

The authorization for SIMOTION SCOUT and SIMOTION SCOUT Standalone can be upgraded as follows:

As of Version 4.0, the authorization for SIMOTION SCOUT has been upgraded from authorization (*single license*) to the licensing procedure involving a *floating license*. The Automation License Manager program manages the licenses.

1. Connect or insert the data medium containing the authorization and the upgrade license key.
2. Start the Automation License Manager:
   - In the start menu, **Start > All programs > Siemens Automation > Automation License Manager** or
   - Double-click the **Automation License Manager** icon. A new window opens.
3. In the navigation area (left-hand window), select the drive where the authorization for the old version of SIMOTION SCOUT is located. Generally, the authorization will be installed on a hard disk drive on the PC.
4. Transfer the authorization to the data medium containing the upgrade license key. To do this, select the license in the right-hand window and select **License key > Transfer...** via the menu.
5. In the **Transfer license key** dialog, select the connected data medium. Start the transfer by clicking **OK**.
   The authorization and upgrade license key will then be located on the data medium.

6. Select **License key > Upgrade...** in the menu.
   The previous authorization will be deleted and a new *floating license* will be available once the upgrade is complete.

   Note

   Do not interrupt the upgrade while it is in progress. Interrupting this process can lead to the license key being lost.

7. Now transfer the new floating license onto the hard disk drive. To do this, follow the same procedure as before: Via **License key > Transfer...** in the menu
8. After the transfer has completed successfully, remove the automation data medium.
9. Exit the Automation License Manager. SIMOTION SCOUT can now be operated without restrictions.

Saving and moving the license key

Saving and moving the license key

You can transfer the license key to a removable disk. This is useful when you want to save the license key when reinstalling a PC or when you want to use it on another PC. A copy of the license key is not created during this operation, it is moved.

Note

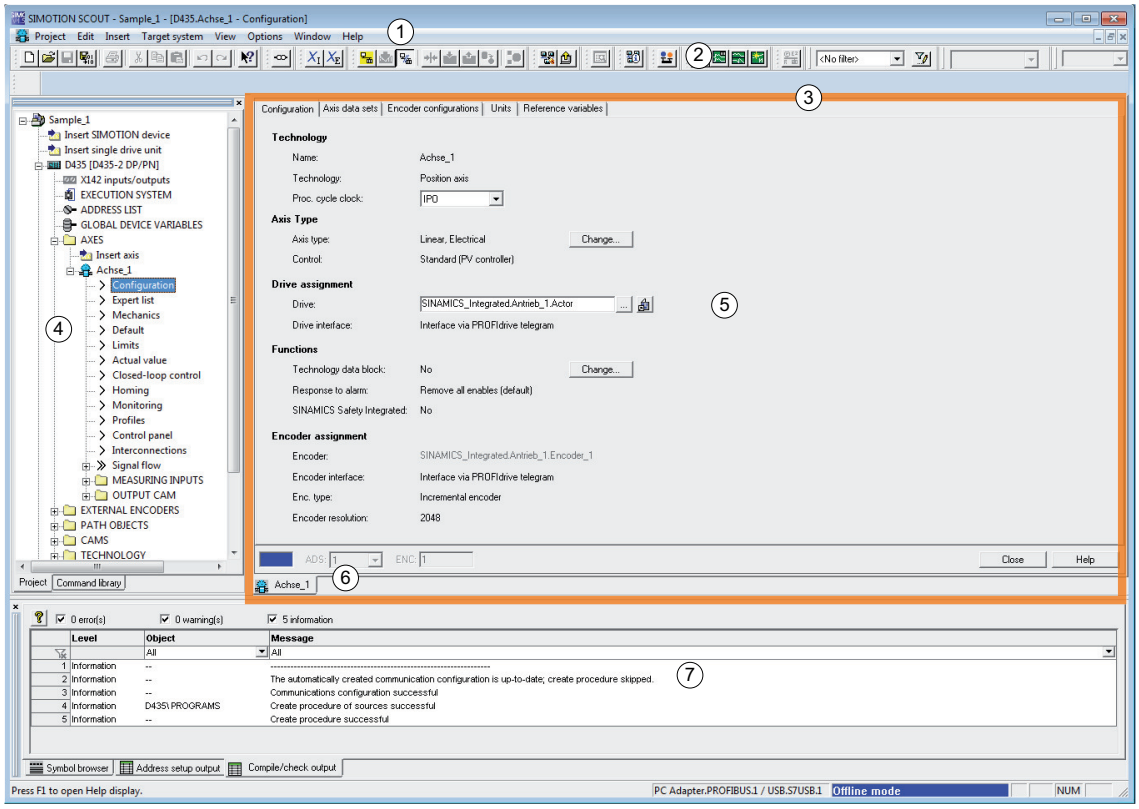
More detailed information on the license keys can be found in the online help for the Automation License Manager.

3.3.5 User interface

3.3.5.1 SIMOTION SCOUT - workbench

The SIMOTION SCOUT Workbench is the common framework for all other tools of the engineering system. The workbench is the navigation center of the individual engineering steps and offers a uniform and integrated view of all data and programs.

The following figure shows an example of the Workbench components:



① Menu bar
② Tool bars
③ Working area
④ Project navigator
⑤ Snap-in
⑥ Tab
⑦ Detail view

Workbench components

The workbench comprises the following components:

- **Menu bar**:
  You call the functions of SIMOTION SCOUT via the menus in the menu bar.
- **Tool bars:**
  Frequently used menu commands are also available in tool bars, which can be activated or deactivated as required. These provide quick access to the functions. The tool bars can be undocked from the header and relocated to a different position (e.g. at right, left, lower border) or as a window.
- **Working area**:
  The task-specific windows are displayed in the working area. In these windows, you can perform the configuration with wizards for the axis configuration and drive configuration. You also create programs in the working area. Further information about the active window in the working area is provided in the detail view.

- **Project navigator**:
  The project navigator provides an overview of the entire project. All defined elements, such as devices, drives, axes, etc., are displayed here in a tree structure.
- **Snap-in / tab:**
  A Snap-in is a program integrated automatically in the working area of the SIMOTION SCOUT Workbench. Snap-ins provide functions for processing SIMOTION SCOUT projects and are displayed as a work window in the working area of the workbench. Several Snap-ins can be opened. Open Snap-ins are displayed in the working area as tabs. The active Snap-in is visible in the foreground.
  The following Snap-ins are available:
  - Program editors
  - Wizards for the configuration of technology objects
  - Device diagnostics
  - Drive navigator
- **Detail display:**
  In the detail display, you can show more detailed information about the element selected in the project navigator and the active window in the working area, for example, variables of a program, system variables of a technology object, protocols of compiled program sources.

Additional references

Please refer to the following documents on these subjects

- Function Manual: *SIMOTION SCOUT Runtime Basic Functions*
- Function Manual: *SIMOTION Motion Control, Basic Functions for Modular Machines*
- Function Manual: *SIMOTION Motion Control, TO Axis, Electric/Hydraulic, TO External Encoder*
- Function Manual: *SIMOTION Motion Control, Synchronous Operation TO, TO Cam*
- Function Manual: *SIMOTION Motion Control Output Cams and Measuring Inputs*
- Function Manual: *SIMOTION Motion Control, TO Path Object*
- Function Manual: *SIMOTION SCOUT Supplementary Technology Objects*

and the online help.

Note

We recommend that you work through the sample configuration *Getting Started SIMOTION SCOUT*. This is a series of guided steps which teaches you how to work with SIMOTION SCOUT. For example, how to create, compile, and save a project, insert a SIMOTION device, insert and assign parameters for a technology object, and create a program.

When you have worked through all these steps, you will be able to create more complex projects.

Getting Started is available in two versions with different scopes, as a print document and in the online help.

Note

It is not recommended that you open **one** SIMOTION SCOUT project twice. This application case can result in malfunctions and is not supported by SIMOTION SCOUT. If you want to open two different projects, you must open SIMOTION SCOUT twice.

3.3.5.2 SIMOTION SCOUT - working area

The workbench displays all of the Snap-in work windows in the working area. Each Snap-in provides its own work window. You can open multiple instances of these windows. You can, for example, open several programs at the same time for editing. For further information, refer to the online help.

The following table provides an overview of the active functions in the working area of the workbench depending on the project mode (offline/online):

Function in project navigator	**Function in working area offline mode (Example)**	**Function in working area online mode (Example)**
Programs (MCC, ST, LAD/FBD) For further information, please go to: • SIMOTION MCC Motion Control Chart • SIMOTION ST • SIMOTION LAD/FBD	Create program	Monitor program
Execution system	Assign program	-
Technology objects (TO configuration)	Create and configure TO element	Change configuration data during RUN
Cam editor CamEdit/CamTool	Create cam	Change cam
Trace	Load and evaluate old recordings	Create current recordings
Axis control panel	-	Traverse axes during commissioning
Drive control panel	-	Traverse drives during commissioning
Path control panel	-	Commission kinematics

Window in the working area

You can change the size of the windows in the working area:
Click the edge of the window, hold down the left mouse button, and drag the window to the required size.

To maximize or restore a window, press <Ctrl+F11>.

Each window opened in the working area can be accessed via a tab at the bottom edge of the working area.

The following options are available to bring a window into the foreground:

- Click the relevant tab.
- Select the appropriate entry in the **Window** menu.

To close a window, proceed as follows:

- Configuration dialogs:
  Click **Close**.
- Editors for MCC, ST and LAD/FBD:
  Click the **X** button in the top right-hand corner.

Configuration dialogs and editors can also be closed with <Ctrl+F4>.

3.3.5.3 SIMOTION SCOUT project navigator

Using the project navigator

The project navigator in the SIMOTION SCOUT workbench

The project navigator has two tabs as standard, the **Project** and **Command library** tabs.

The **Project** tab displays the entire project structure and is used for managing elements within the projects.

The commands and functions required for programming are displayed in a tree on the **Command library** tab. You can search in the command library or set filters. You can use commands and functions in the ST, LAD/FBD and MCC programming languages, e.g. for the creation of conditions. In the MCC programming language, the functions are used, e.g. via the ST zoom or the system function call command.

Creating elements

Elements in SIMOTION SCOUT

There are different ways to insert elements in the project navigator tree:

Inserting hardware

Integrate hardware using:

- SIMOTION device element: **Insert SIMOTION device**

Alternatively, you can also call up the **Insert > SIMOTION device** menu.

- Drive element: **Insert single drive unit**

Note

You can insert a standalone drive (e.g. SINAMICS S120) with the **Insert single drive unit** element in the project navigator. It is commissioned using wizards in the working area of the workbench that contains the Starter functionality.

Elements within a SIMOTION device

Create the following elements within a SIMOTION device directly in the project navigator:

- Technology objects, e.g.:
  Axes, external encoders, cams, measuring inputs, output cams, synchronous operation, temperature channels, path objects
- Programs:
  - Insert ST program
  - Insert MCC source
  - Insert DCC charts
  - Insert LAD/FBD source

Other elements are created automatically, e.g.

- When creating a project:
  - Folder **LIBRARIES** (Insert library / insert DCC library)
  - Folder **LIBRARIES SINAMICS** (Insert DCC library)
  - Folder **MONITOR** (Insert watch table)
- When inserting a SIMOTION device:
  - **AXES**, **EXTERNAL ENCODERS**, **PATH OBJECTS**, **CAMS**, **TECHNOLOGY, PROGRAMS** folders
  - EXECUTION SYSTEM, ADDRESS LIST, GLOBAL DEVICE VARIABLES elements
- When creating an axis:
  - **MEASURING INPUTS**, **OUTPUT CAM, CAM TRACK** folders
  - Access to the configuration views
- Additionally when creating a following axis:
  - Synchronous operation element

Insert single drive unit

To insert a single drive unit element:

In the project navigator, double-click **Insert single drive unit**.
This enables you to insert a standalone drive (e.g. SINAMICS S120). The STARTER functionality in SCOUT is responsible for commissioning the drive concerned.

Inserting elements within a SIMOTION device

To insert technology objects, source files, or watch tables:

1. Open the SIMOTION device under which the element is to be created.
2. Select the relevant folder (e.g. **AXES**, **PROGRAMS**).
3. Select the desired function, e.g.:
   - **Insert > Technology object > ...** menu or
   - **Insert > Program > ...** menu or
   - **Insert > Watch table** menu

Displaying the station level of a SIMOTION device

This function provides a better transparency in large SIMOTION SCOUT projects. A project with several SIMOTION devices and a large number of drives can be displayed in a structure. The assignment in station levels, as known from SIMATIC STEP 7, is used.

If the function is activated, all drive units belonging to a STEP 7 station are displayed below this station.
If the function is deactivated, all drive units are displayed below the project.

Note

Optionally, the associated SIMATIC station of the SIMOTION device in the project navigator can be displayed. In this way, all drive units that are assigned to a SIMATIC station are also displayed in this structure.

Proceed as follows to activate or deactivate this function:

1. In the SIMOTION SCOUT, click the **Options > Settings** function.
2. Click the **Workbench** tab.
3. Activate/deactivate the **Display station level** function.
4. Click **OK** to confirm the change.

To become effective, the project has to be re-opened.

Opening the HW Config of a SIMOTION device

To call up the hardware configuration of a previously inserted SIMOTION device:

- In the project navigator, double-click the SIMOTION device whose hardware configuration you want to open or
- Select the desired SIMOTION device in the project navigator, open the context menu and click **Open HW configuration**.

Inserting a folder

Inserting a folder in the project navigator

For better structuring of objects in large projects, you can insert subfolders in the project navigator. Using this function, you can sort objects according to topic.

You cannot insert subfolders in DCC libraries and below slaves/devices.

To insert a subfolder, proceed as follows:

1. Select an object and perform **Insert > Folder** in the menu.
   or
2. Right-click the object and perform **Insert folder**.
   An empty folder is inserted.

Changing properties of the elements

You change the properties of an element as follows:

1. In the project navigator, select the element to be edited, e.g. a specific axis or a specific program.
2. Select the menu: **Edit > Object properties**.

Note

You define the hardware properties in the hardware configuration.

To rename an element:

1. Select the element you want to rename in the project navigator.
2. Open the context menu and select **Rename**.
3. Confirm the warning message.
4. Enter a new name and confirm the input by pressing the **Enter** key.

NOTICE
Damage resulting from errors in references to the axis Changing a name can have far-reaching consequences. References to the axis, particularly in programs, are lost. This can cause runtime errors in the program. • For this reason, make sure that you change all name references.

Wizards for configuration support

Wizards are provided to assist you in configuring axes and external encoders. The wizards guide you step-by-step through the configuration. Parameter interdependencies are taken into account.

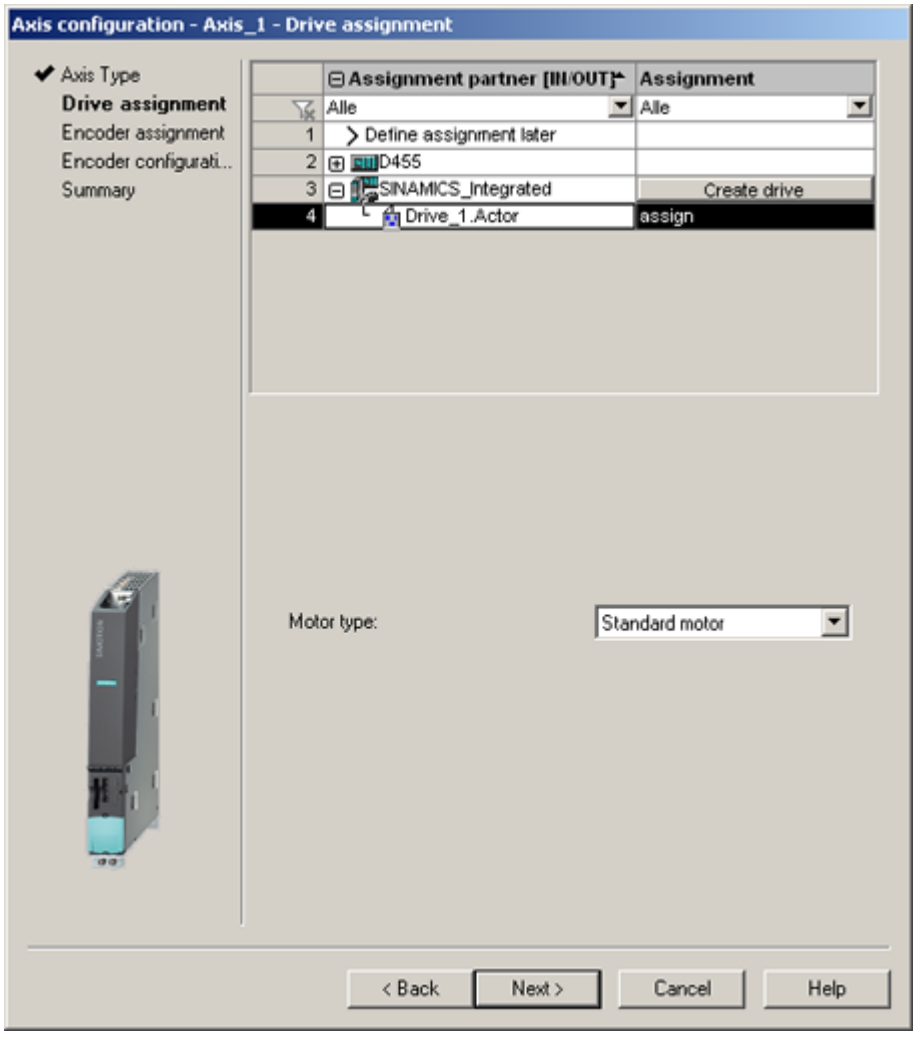


Figure 3-120 Axis configuration wizard - window for drive assignment

3.3.5.4 SIMOTION SCOUT - menus

SIMOTION SCOUT menus are subdivided into static and dynamic menus.

Static menu

Static menus are primarily used to control the workbench or a project and are permanently displayed.

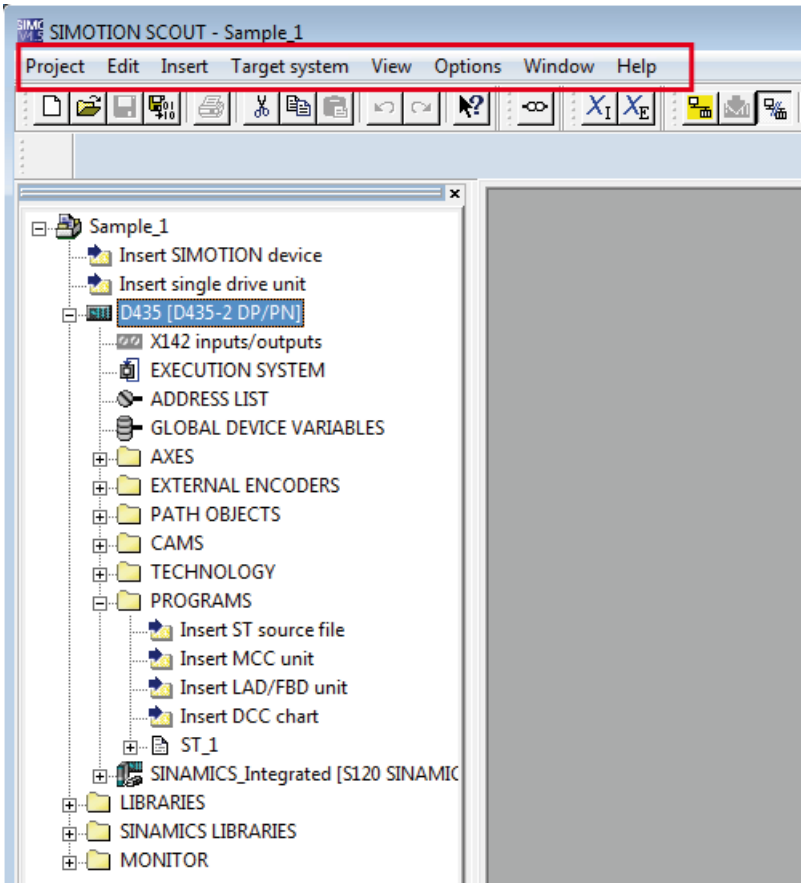


Figure 3-121 Static menu

Dynamic menu

Dynamic menus are provided by Snap-ins and are added to the static menu. The menu active in the working area is always displayed.

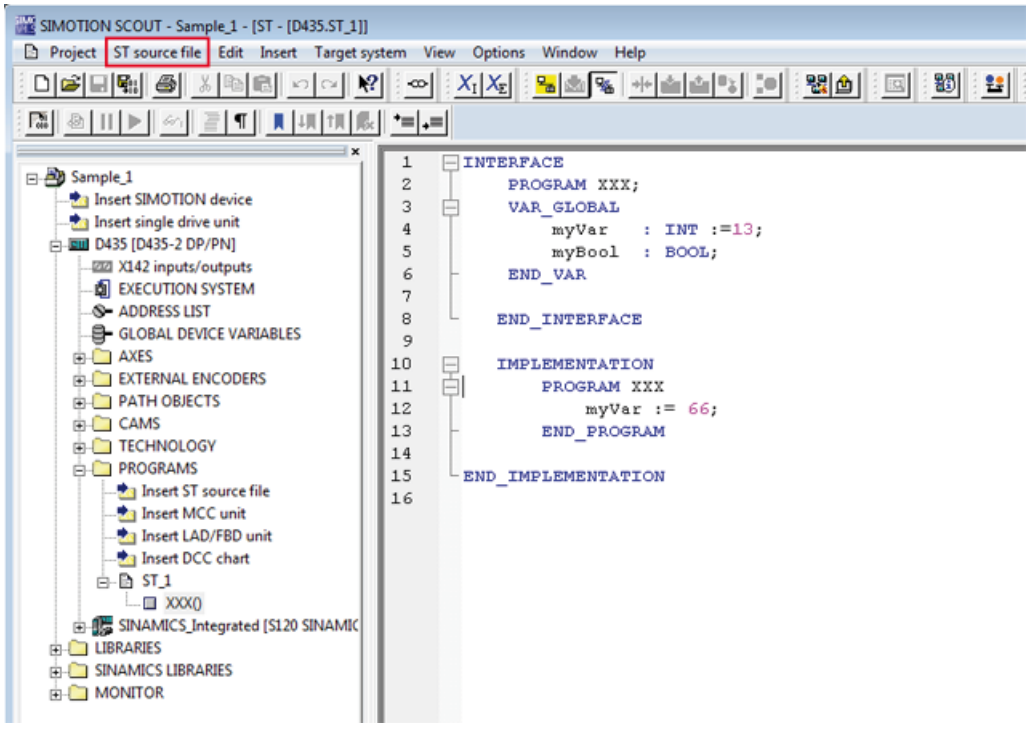


Figure 3-122 Dynamic menu

Table 3-23 Structure of the menu bar

Menu	Comment
Project	Static menu, always visible
(Dynamic menu)	See Dynamic menus table
Edit	Static menu, visible only when project has been loaded
Insert	Static menu, visible only when project has been loaded
Target system	Static menu, always visible
View	Static menu, always visible
Options	Static menu, always visible
Window	Static menu, always visible
Help	Static menu, always visible

Table 3-24 Dynamic menus

Dynamic menu	Comment
Axis	Visible only when the project is open and the associated Snap-in is active in the working area. The dynamic menus appear in the menu bar between the static menus **Project** and **Edit**.
Output cam	
Measuring input	
Synchronous operation	
Cam	
External encoder	
Sensor	
Controller object	
Drive	
Fixed gear	
Formula object	
Addition object	
Drive control panel	
Trace	
ST source file	
MCC source	Visible only when the project is open and the associated Snap-in is active in the working area. Dynamic menus appear in the second position on the menu bar, between Project and Edit.
MCC chart	
LAD/FBD source	
LAD/FBD program	

3.3.5.5 SIMOTION SCOUT - menu items

You can use the shortcuts listed in the table to call the menu items available in SIMOTION SCOUT.

Table 3-25 Menu items in SIMOTION SCOUT

Shortcuts	Menu item	Reaction
Project...		
Ctrl+N	New	Creates a new project
Ctrl+O	Open	Opens a project
Ctrl+S	Save	Saves a project
Ctrl+Alt+K	Check consistency	Checks project consistency
Ctrl+Alt+B	Save and compile changes	Saves the project and compiles the changes made in the project since the last compilation
Ctrl+P	Print	Prints the selected window
Alt+F4	Exit	Exits the SIMOTION SCOUT program

Shortcuts	Menu item	Reaction
Edit...		
Ctrl+Z	Undo	Undoes the last action
Ctrl+Y	Redo	Redoes the last action

Shortcuts	Menu item	Reaction
Edit...		
Ctrl+X	Cut	Cuts the selection
Ctrl+C	Copy	Copies the selection
Ctrl+V	Insert	Inserts the clipboard contents
Del	Delete contents	Deletes the selection
F2	Rename	Renames the selected tree object
Alt+Enter	Object properties	Displays the properties of an object in the project tree
Ctrl+Alt+O	Open object	Opens a new object of the selected tree object
Ctrl+A	Select all	Selects the entire contents in the ST and MCC Snap-ins

Ctrl+F	Find	Opens the Find window
F3	Find next	Continues the search from the current position
Ctrl+Shift+F	Search in the project	Opens the Find window
Ctrl+Shift+G	Replace in the project	Opens the Find and Replace window

CTRL+H	Replace	Opens the Replace window
CTRL+J	Display next position	

Shortcuts	Menu item	Reaction
Target system...		
Ctrl+L	Download / target device	Downloads to individual target device
Ctrl+D Is possible only online	Device diagnostics	Opens device diagnostics
Ctrl+I Is possible only online	Control operating state	Opens the dialog for control of the operating state

Shortcuts	Menu item	Reaction
View...		
Ctrl+F11	Maximize working area	Maximizes and minimizes the view of the working area
Ctrl+F12	Maximize detail view	Maximizes and minimizes the view of the detail view
Ctrl+Num+ (Plus key on the numeric keypad)	Zoom in	Magnifies the diagram in MCC
Ctrl+Num+ (Minus key on the numeric keypad)	Zoom out	Reduces the diagram in MCC
F5	Refresh	Refreshes the view

Shortcuts	Menu item	Reaction
Options...		
Ctrl+Alt+E	Settings	Opens the Properties window

Shortcuts	Menu item	Reaction
Window...		
Ctrl+Shift+F5	Arrange cascading	Arranges the opened windows in the working area
CTRL+SHIFT+F3	Arrange vertically	

Shortcuts	Menu item	Reaction
Snap-in menu...		
Ctrl+F4	Close	Closes the selected window
Ctrl+B	Accept and compile	Compiles the active object
Ctrl+E	Expert list	Opens the expert list for the current technology object

Shortcuts	Menu item	Reaction
ST source file		
Ctrl+space		Automatic completion
Ctrl+F2		Sets and deletes bookmarks
Ctrl+F4	ST source file > Close	Closes the ST source file
Ctrl+F7	ST source file > Program status on/off	Switches the program status function on or off
Ctrl+Shift+F2		Deletes all bookmarks in the ST source file
Ctrl+Shift+F3		Arranges windows, tile horizontally
Ctrl+Shift+F5		Arranges windows, tile vertically
Ctrl+Shift+F8		Formats the selected area
Ctrl+Shift+F9		Moves the cursor to the start of the current or higher-level block
Ctrl+Shift+F10		Moves the cursor to the end of the current block
Ctrl+Shift+F11		Moves the cursor to the start of the higher-level block, 1st level
Ctrl+Shift+F12		Moves the cursor to the start of the higher-level block, 2nd level
Ctrl+D		Duplicates the selected row
Ctrl+Alt+B		Displays bracket pairs in the current ST source file
Ctrl+Alt+C		Folding: Hide all blocks of the current ST source file
Ctrl+Alt+D		Folding: Display all blocks of the current ST source file
Ctrl+Alt+F		Folding: Display or hide folding information in the current ST source file

Shortcuts	Menu item	Reaction
ST source file		
Ctrl+Alt+I		Displays indentation level in the current ST source file
Ctrl+Alt+L		Displays or hides line numbers in the current ST source file
Ctrl+Alt+R		Folding: Display all subordinate blocks
Ctrl+Alt+T		Folding: Display/hide block.
Ctrl+Alt+V		Folding: Hide all subordinate blocks
Ctrl+Alt+W		Displays or hides spaces and tabs in the current ST source file
Ctrl+ADD (numeric keypad)		Increases the font size in the current ST source file
Ctrl+MINUS (numeric keypad)		Decreases the font size in the current ST source file
Ctrl+DIV (numeric keypad)		Resets the font size in the current ST source file to 100%
Alt+Shift+arrow key		Selects text by column
Alt+Shift+L		Changes selected text to upper case
Alt+Shift+U		Changes selected text to lower case

Shortcuts	Menu item	Reaction
MCC		
Ctrl+F4	Close	Closes the MCC source
Alt+Enter	Properties	Opens the MCC Source Properties window
Ctrl+R	Insert MCC chart	Inserts a new MCC chart
Ctrl+F7	Program status	Switches monitoring on and off
Ctrl+F8	Monitor	
Ctrl+F9	Single-step	
Ctrl+F10	Next step	
Return	–	Opens the MCC configuration dialog
Arrow keys	–	Changes within the tab

Shortcuts	Menu item	Reaction
LAD/FBD (source or program)		
Ctrl+F4	Close	Closes the MCC source
Alt+Enter	Properties	Opens the MCC Source Properties window
Ctrl+R	LAD/FBD source: Insert LAD/FBD program LAD/FBD program: Insert network	Inserts a new LAD/FBD program
Ctrl+L	Jump label On/Off	Switches jump labels on and off
Ctrl+Shift+K	Display/comment On/Off	Switches comments on and off
Ctrl+Shift+B	All box parameters	Displays all box parameters

Shortcuts	Menu item	Reaction
LAD/FBD (source or program)		
Ctrl+T	Symbol check and type update	Performs symbol check and type update
Ctrl+F7	Program status	Switches monitoring on and off
Shift+F2 for LAD Shift+F2 for FBD	Insert element / Insert make contact element / AND box	Inserts a make contact Inserts an AND box
Shift+F3 for LAD Shift+F3 for FBD	Insert element / Insert break contact element / OR box	Inserts a break contact Inserts an OR box
Shift+F7 for LAD Shift+F7 for FBD	Add element / Add coil Add element / assign-ment or jump	Adds a coil Adds an assignment or a jump
Shift+F8 for LAD Shift+F8 for FBD	Insert element / Insert Open branch element / Insert binary in-put	Inserts an Open branch Inserts a binary input
Shift+F9 for LAD Shift+F9 for FBD	Insert element / Insert Close branch element / Invert binary in-put	Inserts a Close branch Inverts the binary input
Shift+Alt+F9	Insert element / empty box	Inserts an empty box
Ctrl+3 Ctrl+1	Switch to FBD Switch to LAD	Switches the display

Shortcuts	Menu item	Reaction
Help...		
F1	Help topics	Opens the entire help available for SIMOTION SCOUT
Shift+F1	Context-sensitive help	Opens the context-sensitive help function for the selected object, parameter, etc.

Shortcuts	Menu item	Reaction
Hardware configuration...		
Ctrl+U (in HW Config)	Address overview	Opens the Address Overview window
F4	Optimize layout	

Note

List of all shortcuts

A complete list of all the shortcuts in SIMOTION SCOUT is available in the online help, and in the appropriate programming and operating manuals.

3.3.5.6 SIMOTION SCOUT - keyboard operation and shortcuts

Note

There are various keyboard assignments and shortcuts for the menu items to facilitate your work in SIMOTION SCOUT.

The following table provides an overview of the keyboard assignments and shortcuts that you can use for SIMOTION SCOUT.

Table 3-26 Keyboard operation

Keyboard operation / shortcuts	**Meaning**
Workbench: Change window...	
Alt+0	Project navigator
Alt+1	Working area
Alt+2	Detail view
Ctrl+F6	Next window in the working area
Ctrl+F11	Minimize/maximize working area in relation to the whole desktop
Ctrl+F12	Minimize/maximize detail view in relation to the whole desktop

Project navigator	
Click with left mouse button	Selects the tree object at the cursor position; the detail view displays the associated details
Double-click with left mouse button	Selects the tree object at the cursor position; the detail view displays the associated details; the associated Snap-in opens
Click with right mouse button	Selects the tree object at the cursor position; the detail view displays the associated details; the context menu opens
"Up/Down" arrow keys	Selects the tree object at the cursor position
"Return"	Snap-in for the selected tree object opens
"Context menu key"	Context menu for the selected tree object opens

Program editors	
Alt + "Left/Right" arrow keys	Go to the use location (starting at the selection)

3.3.5.7 SIMOTION SCOUT - using the context menus

The tree elements of the project navigator have context menus. These provide quick access to all major functions enabled for this tree element.

To call the desired function of the tree element via context menus, proceed as follows:

1. Select the appropriate element in the project navigator.
2. Open the context menu with a right-click.
3. Left-click the appropriate command to select it.

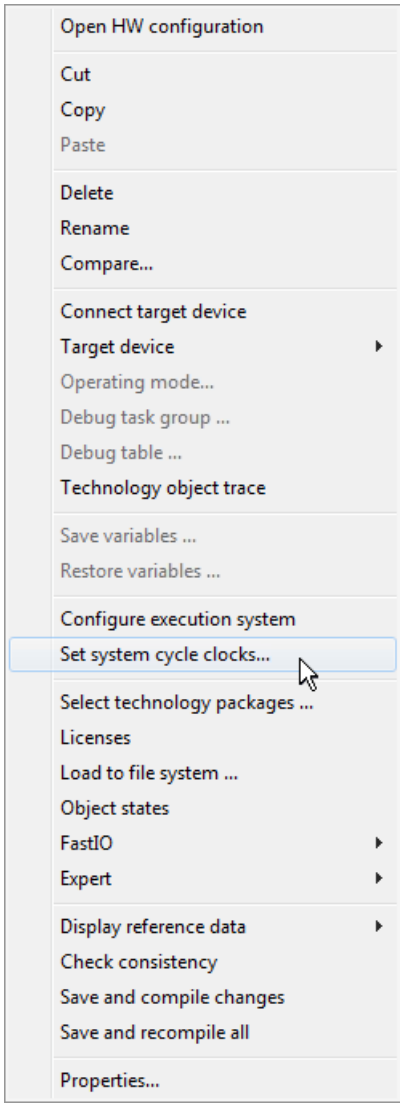


Figure 3-123 Context menu for SIMOTION device

3.3.5.8 SIMOTION SCOUT - detailed view

Using the detail view

When you select an element in the project navigator, the associated detail view will appear in the lower area of the workbench. A detailed view is not available for every element.

Depending on the selected element, different tabs are offered and information about the element is displayed therein.

To display the address list and the watch table, you must open them with a double-click. The detailed view then displays in the associated tabs the appropriate content for the selected element.

The tabs available are determined by the project mode (offline/online) and the active Snap-ins.

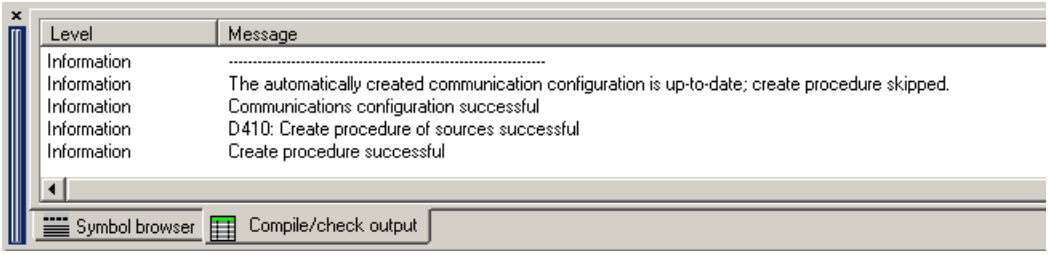


Figure 3-124 Example of detail view

Each tab is opened just once, i.e.:

- The active tab shows the details of the selected element.
- The contents of the tab change when you select a different element.

You can maximize or minimize each tab with <Ctrl+F12>.

Using the symbol browser

The symbol browser is a tab in the detail view. The symbol browser displays status values of the variables for the element selected in the project navigator.

Display symbol browser

To display the symbol browser, proceed as follows:

1. Select an element in the project navigator.
2. In the detail view, select the **Symbol browser** tab.

Displaying the symbol browser content continuously

To continuously display the content of the symbol browser for a selection, e.g. for the execution system, proceed as follows:

1. Click the symbol at the top right in the symbol browser view.

The active element remains displayed until this function is deactivated. To deactivate this function, click the pin again.

Address list

The address list is where you create I/O variables, which you then assign to a particular item of hardware. You can monitor I/O variables in the address list, and, when necessary, modify them. The I/O variables are assigned either manually or via an assignment wizard.

Opening the address list

To open the address list, proceed as follows:

1. Browse to the folder for the device in the project navigator.
2. Double-click the **Address list** element below the device.
   The address list opens in the details area.

Additional references

Further information about the address list is available in the SIMOTION SCOUT online help.

Watch table

With the symbol browser you can view the variables belonging to one object in your project; with the program status you can view the variables belonging to a selected monitoring area in the program. With watch tables, in contrast, you can monitor selected variables from different sources as a group (e.g. program sources, technology objects, SINAMICS drives - even on different devices). You can sort the variables in the watch table in any way you want, add comments regarding them, and combine them into groups. You can hide individual variables to make the watch table more manageable and also control variables via the watch table directly.

Creating a watch table

To create a watch table and assign your variables, proceed as follows:

1. In the project navigator, open the **Monitor** folder.
2. Double-click the **Insert watch table** entry to create a watch table.
3. Enter the name of the watch table.
   A watch table is created with this name in the **Monitor** folder.
4. In the project navigator, click the object from which you want to move variables to the watch table.
5. In the symbol browser, in the address list or in the expert list, select the corresponding variable line by clicking its number in the left column.
6. From the context menu, select the **Add to watch table** command and the appropriate watch table, e.g. **Watch_table_1**.
7. If you click the watch table, you will see from the **Watch table** tab of the detail view that the selected variable is now in the watch table.
8. Alternately, you can copy-and-paste variables to the watch table or enter them from the keyboard.
9. To monitor the variables of various objects, repeat steps 3 to 6.

You can also create a watch table directly by selecting a variable followed by **Add to watch table > New watch table** in the context menu.

The new watch table, containing the selected variable, is created automatically.

If you are connected to the target system, you can monitor the variable contents.

Using several watch tables (Watchtabelle_temp)

If you want to monitor several watch tables simultaneously (watch tables of different objects), create a temporary watch table.

The temporary watch table has the following properties:

- It is always called Watchtabelle_temp
- It does not save the contents when it is closed
- It can be filled using drag-and-drop
- It does not support control records
- With dropped watch tables, it does not display any control values (control checkmarks are not set).
- Transfers the control values and control checkmarks; Create copy transfers them to the copy of the temporary watch table.

Using a temporary watch table

1. Click **Watchtabelle_temp** below **Monitoring** in the project navigator.
   The temporary watch table is displayed in the detail view.
2. Drag the required watch tables to the temporary watch table using drag-and-drop.
   The watch tables are arranged one below the other.

Additional references

Detailed information on this topic is available in the SIMOTION SCOUT online help.

Working with lists

You have various options to make working with lists (e.g. watch table, address list, symbol browser or expert list) more transparent and easier to configure, and also to be able to display larger data volumes in a structure. You can hide unimportant or redundant information and focus on important information. You can also transfer and group contents quickly and easily.

Additional references

Detailed information on this topic is available in the SIMOTION SCOUT online help.

3.3.5.9 SIMOTION SCOUT - language settings

Language setting of the SIMOTION SCOUT

SIMOTION SCOUT uses the language default setting in the SIMATIC Manager. You can make changes to this setting in the SIMATIC Manager via the **Options > Settings... > Language** menu.

German, English, French and Italian languages are currently available in SIMOTION SCOUT. If a different language is set in the SIMATIC Manager, SIMOTION SCOUT uses the English setting.

Note

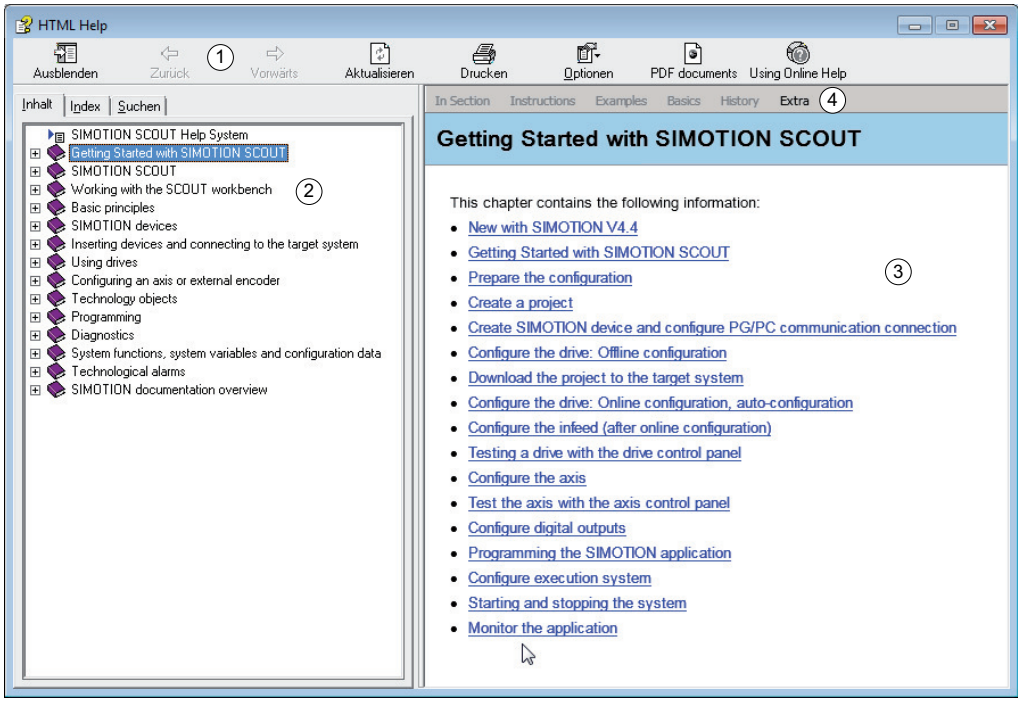
Only those languages installed in STEP 7 are available.

3.3.5.10 SIMOTION SCOUT - using help

SIMOTION SCOUT Online Help

SIMOTION SCOUT has a comprehensive context-sensitive help. The following is a description of how to work with the help system using different examples.

Structure of the online help window



1 Menu bar

You can use buttons in the menu bar to configure the help or display general information.

- Click "PDF documents" to display the available function diagrams.
- Click "Use online help" to display the help page for the full text search on the "Search" tab.

2 Navigation area

In the navigation area you can navigate through the help, open the index or search through the entire help.

- Click the "Contents", "Index", or "Search" tab to navigate through the help.
- Click the "+" in front of a book to open the table of contents.
- Click a book or help page to display its contents in the contents area.

3 Contents area

The contents for the help pages selected in the navigation area are displayed in the contents area.

Within the contents area, you can use symbols to display hidden information:

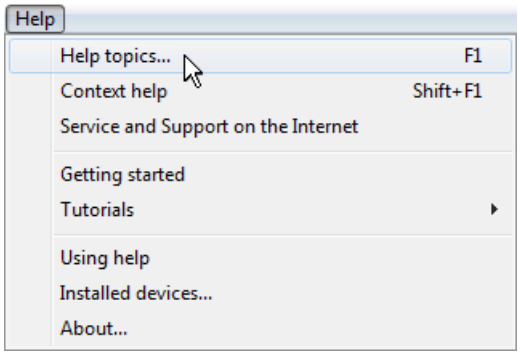
- Click the arrow to display or hide additional information.
- Click the minimized view to display a hidden figure.
- Click the copy symbol. The following program code will be saved automatically in the clipboard. You can then insert this in a program editor.

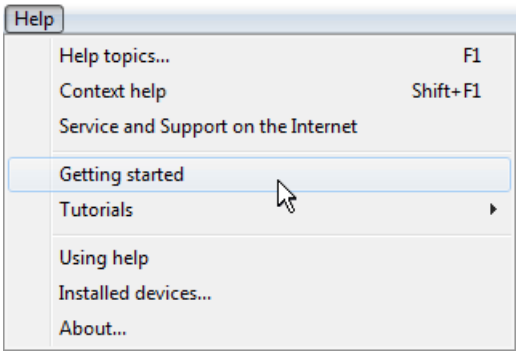
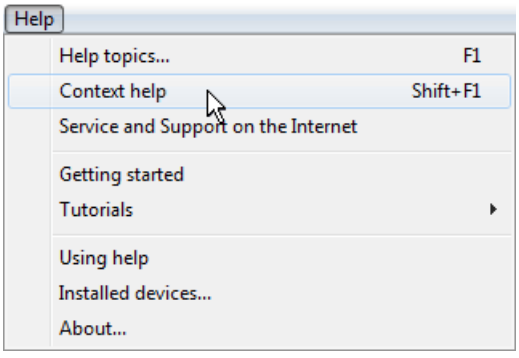
4 Link bar

The header area of many help pages has a link bar. This link bar displays further information for the selected help page. The link bar contains the following entries:

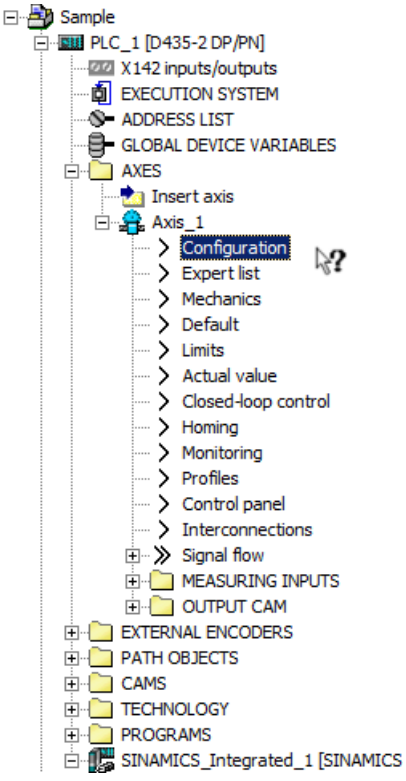
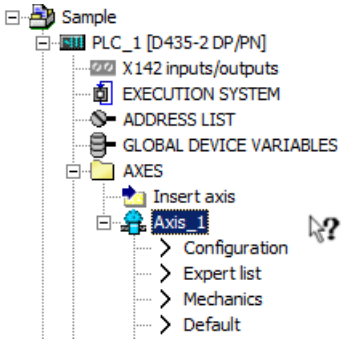
- In Section
  Links to headers of the displayed topic.
- Manuals
  Links to handling instructions.
- Examples
  Links to examples.
- Fundamentals
  Links to background information, such as definitions or details.
- History
  List of the most recently opened help pages.
- Options
  Link to the start page.
  Forwards and backwards in the previously opened help pages.

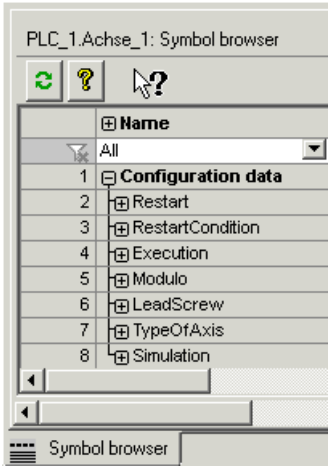
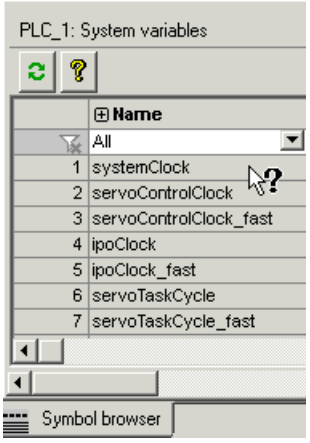
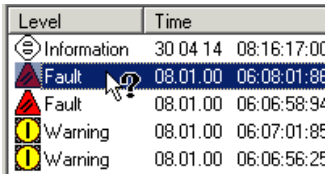
Features of online help

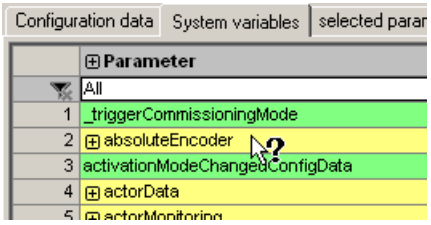
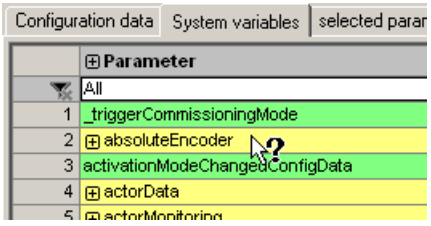
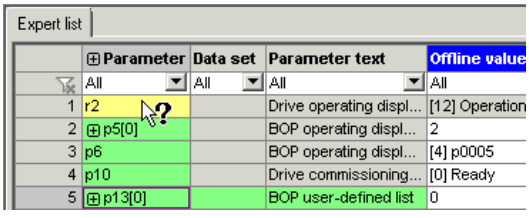
Features of online help	**To open the help:**	
Open entire help	Key <F1>	Press <F1>.
	or	
		Select "Help > Help topics" from the menu.
	The entire help opens.	

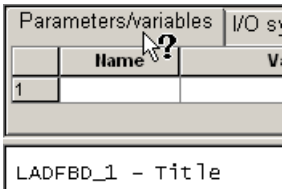
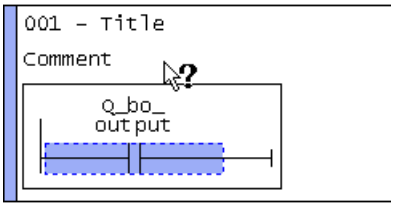
Features of online help	To open the help:	
Open Getting Started		Select "Help > Getting Started" from the menu.
	Getting Started opens.	
Help button	 or 	Click the Help button of the associated dialog or window.
	The context-sensitive help for the dialog opens.	
Context-sensitive help	<Shift+F1>	• Press <Shift+F1>.
	• or	
		• Click the button in the menu bar.
	• or	
		• In the menu, select "Help > Context help".
	The mouse pointer changes to a question mark.	
		• Then click with the changed mouse cursor on the dialog, the parameter, the input field or the menu item.
	The context-sensitive help for the selected entry opens.	

Examples of the context-sensitive help

Help in the project navigator		
General help		Press <Shift+F1>. Then click with the question mark on the project navigator. The general help for the project navigator opens.
Help for the technology object (only SIMOTION)		Press <Shift+F1>. Then click with the question mark on a technology object in the project navigator. The help for this technology object opens.

Help for the detail view		
General help for the symbol browser		Press <Shift+F1>. Then click with the question mark in the margin of the symbol browser. Alternatively, you can also open the help for the symbol browser by clicking . The general help for the symbol browser opens.
System variables in the symbol browser		In the project navigator, select the SIMOTION device or the technology object. The system variables of the element are displayed in the symbol browser. Press <Shift+F1>. Then click with the question mark on the system variable in the symbol browser. The associated help opens.
Help for alarms		Press <Shift+F1>. Then click with the question mark on the alarm shown on the "Alarms" tab. The help for the drive alarm or technological alarm opens.

Help for the expert list		
Configuration data		Press <Shift+F1>. Then click with the question mark on the configuration data item in the expert list. The help opens.
System variables		Press <Shift+F1>. Then click with the question mark on the system variable in the expert list. The help opens.
Parameters of the drives		Press <Shift+F1>. Then click with the question mark on the parameter in the expert list. The help opens.

Help for the LAD/FBD editor (only SIMOTION)		
Declaration area		Press <Shift+F1>. Then click with the question mark on a tab in the declaration area of the LAD/FBD editor. The help opens.
Editor area		Press <Shift+F1>. Then click with the question mark on an element in the editor area of the LAD/FBD editor. The help opens.

Searching in the online help

You can carry out a full-text search throughout the entire Help in the **Search** tab.

With a full-text search, you have to consider certain points to ensure that the search is successful. A simple search according to subject consists of the word or expression you want to find.

To refine your search, you can use wildcard expressions, nested expressions, Boolean operators, similar word hits, the previous result list or subject titles.

To carry out a full-text search:

1. In the menu bar, select **Help > Help topics** or press key **F1**. The entire help is displayed.
2. Click the **Search** tab and enter the search term. You can add Boolean operators by clicking the arrow next to the input field.

   Note

   If you enter the term *System*, this term is sought throughout the entire Help. Only the Help pages that contain the word *System* are found. Compound words, e.g. *Systemclock* are not found. Therefore it is usually more appropriate to search with wildcards if you are not searching for a definite term, e.g. **system** returns all the results that contain *system*.

3. Click **List subjects** to start the search. The search returns a maximum of the first 500 hits. If you want to sort the subject list, click **Title**, **Position** or **Order**. With many search results, it is better to sort them by position so hits are grouped together by subject matter.
4. Double-click the desired subject to display the Help page. The found terms are highlighted on the page.

   Note

   If the terms are not highlighted, of if you want to turn the highlighting off, click the **Options** button. select Activate/deactivate search term highlighting in the menu displayed.

You can also search just within the last result list, include similar word hits or only search the subject titles in the table of contents.

In the case of multiple matches, a clearer overview can be obtained by sorting them with **Position** , as the search results will then be arranged according to the associated topic.

Additional references

For more information regarding this topic, see the online help.

Getting Started with SIMOTION SCOUT

Calling Getting Started with SIMOTION SCOUT

Getting Started is displayed by default when SIMOTION SCOUT is opened.

Getting Started provides an overview of how you can work with SIMOTION SCOUT. An example shows you the different steps required to create a project and introduces the configuration, programming, and diagnostic tools.

In addition, the "New with SIMOTION V4.5" section provides an overview of which new functionalities are available in the latest version of SIMOTION.

Note

We recommend that you work through Getting Started with SIMOTION SCOUT. You will then be able to start work on more complex projects as well.

Note

Getting Started is available in two versions as of Version 4.4. The print version deals with the sample configuration based on a SIMOTION D435-2. The online help is more comprehensive. It takes account of all three SIMOTION platforms, as well as: Drive configuring with the drive wizard, testing of the drive with the drive control panel, configuring of a virtual axis.

Deactivating the Getting Started with SIMOTION SCOUT online help

To deactivate the default setting that automatically opens the online help when you launch SIMOTION SCOUT:

1. In the menu bar, select **Options > Settings...**
2. Click the **Workbench** tab.
3. Deactivate the **Display "Getting started" at start** checkbox.
4. Click **OK**.

Error remedy

In the online help, you can find detailed information about error messages appearing in the detail view of the SIMOTION SCOUT workbench. This is possible for technology object alarms and alarms for drive units and I/O devices.

To do this, click the **Help for event** button.

To call the context-sensitive help for the associated compiler error messages in the detailed view, click the button and then the question mark for the associated message.

Additional information about the error text can be found using the full-text search.

3.3.5.11 Adding add-ons to the workbench

Add-ons in SIMOTION SCOUT

SIMOTION provides you with the option of enhancing the functionality or accommodating customer-specific requirements. You can integrate add-ons. The CamTool add-on is currently available.

Note

More detailed information about the CamTool add-on is contained in the *SIMOTION SCOUT CamTool* configuration manual or the associated online help.

Add-ons are added to the workbench and are displayed as fully integrated:

- Menus and toolbars appear at the appropriate position in the workbench. After installation of add-ons, the menus are visible and the toolbars are active.
- The working windows appear in the working area of the workbench and have tabs.
- The detail view of the workbench shows details about the currently active add-on. If the associated details are distributed over several tabs, you can select the relevant tab and place it in the foreground.

3.3.6 Configuring/parameterizing

3.3.6.1 Configuration overview

In order that the machine or plant can perform the desired tasks, you must carry out the following steps with SIMOTION SCOUT:

- Create a project:
  The project is the generic term under which you store all of the relevant files.
- Configure hardware:
  Provide the SIMOTION system with information about the hardware.
- Configure technology objects (TO):
  Provide the SIMOTION system with information about the technologies used.
  SIMOTION SCOUT provides wizards for comprehensive assistance.
- Create and test open-loop and closed-loop control programs.
- Commission and optimize drives and axes.
- Test the system and utilize the diagnostic options.

SIMOTION SCOUT provides the workbench to enable you to perform these steps easily and efficiently.

3.3.6.2 SIMOTION SCOUT - basic settings

In SIMOTION SCOUT, you have the possibility of predefining various basic settings via the **Options > Settings** menu. Detailed information on the individual tabs and settings is provided in the SIMOTION SCOUT online help.

1. Open SIMOTION SCOUT.
2. Click the **Options > Settings...** menu.
   The **Settings** window opens.
3. Click the desired tab.
4. Enter the settings.
5. Click **OK**.

Note

Select **Scripts > Additional scripts > Scripts for SIMOTION SCOUT Engineering** in *Utilities&Applications* to access scripts for making useful standard settings.

3.3.6.3 Managing projects

SIMOTION SCOUT project

The project is the highest level in the data management hierarchy. SIMOTION SCOUT saves all data which belongs, for example, to a production machine, in the project directory. The project is therefore the sum of all data, programs, settings, etc.

The project encompasses all of the SIMOTION devices, drives, etc., that belong to a machine. The axes, external encoders, path objects, cams, technology, and programs are assigned hierarchically to the respective devices. This hierarchical structure is displayed in the project navigator.

Creating a new project

To create a new project

1. Open SIMOTION SCOUT from the Start menu on the Windows desktop **Start > All Programs > Siemens Automation > SIMATIC > STEP 7 > SIMOTION SCOUT** (Windows 7)**.**
   Or double-click the **SIMOTION SCOUT** icon on your PC desktop.
   SIMOTION SCOUT starts the workbench and the online help **Getting Started SIMOTION SCOUT**.
2. In the workbench menu bar, select **Project > New**.
   The **New Project** window opens.

Note

SIMOTION SCOUT creates a directory with the selected name. All files relevant to the project are stored in this directory.

Directory names and file names are shortened to 8 characters. If you assign longer names, they are displayed in SIMOTION SCOUT. However, these names are only managed with 8 characters in the Explorer.

Special characters may not be used.

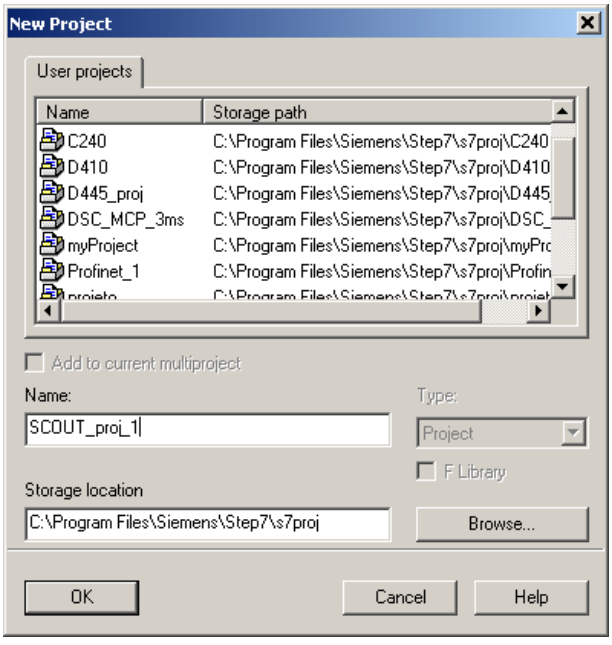


Figure 3-125 Creating a new project

3. Enter the following data in the dialog window:
   - **Name**:
     Select a name for the project.
   - **Storage location (path)**:
     The default storage location (path), where the new project is to be stored, is displayed here.
4. Click **OK** to confirm.

The new project appears in the project navigator with an icon and its full name. The associated tree is expanded.

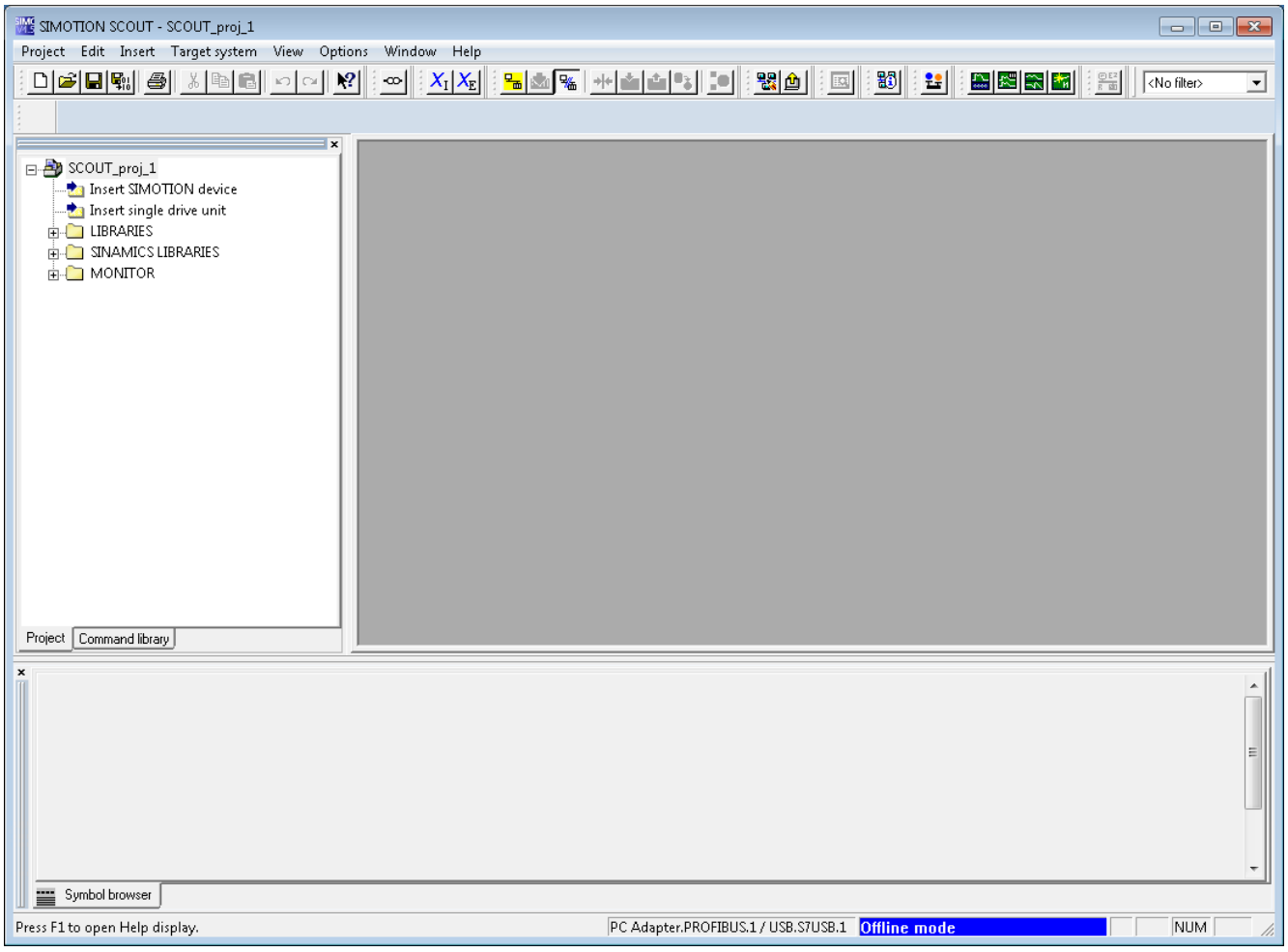


Figure 3-126 Workbench with new project

The following entries are located beneath the project name in the project navigator:

- **Insert SIMOTION device** element. Double-click this icon to select the required SIMOTION device, create and configure a new subnet and insert the new device in the HW Config program.

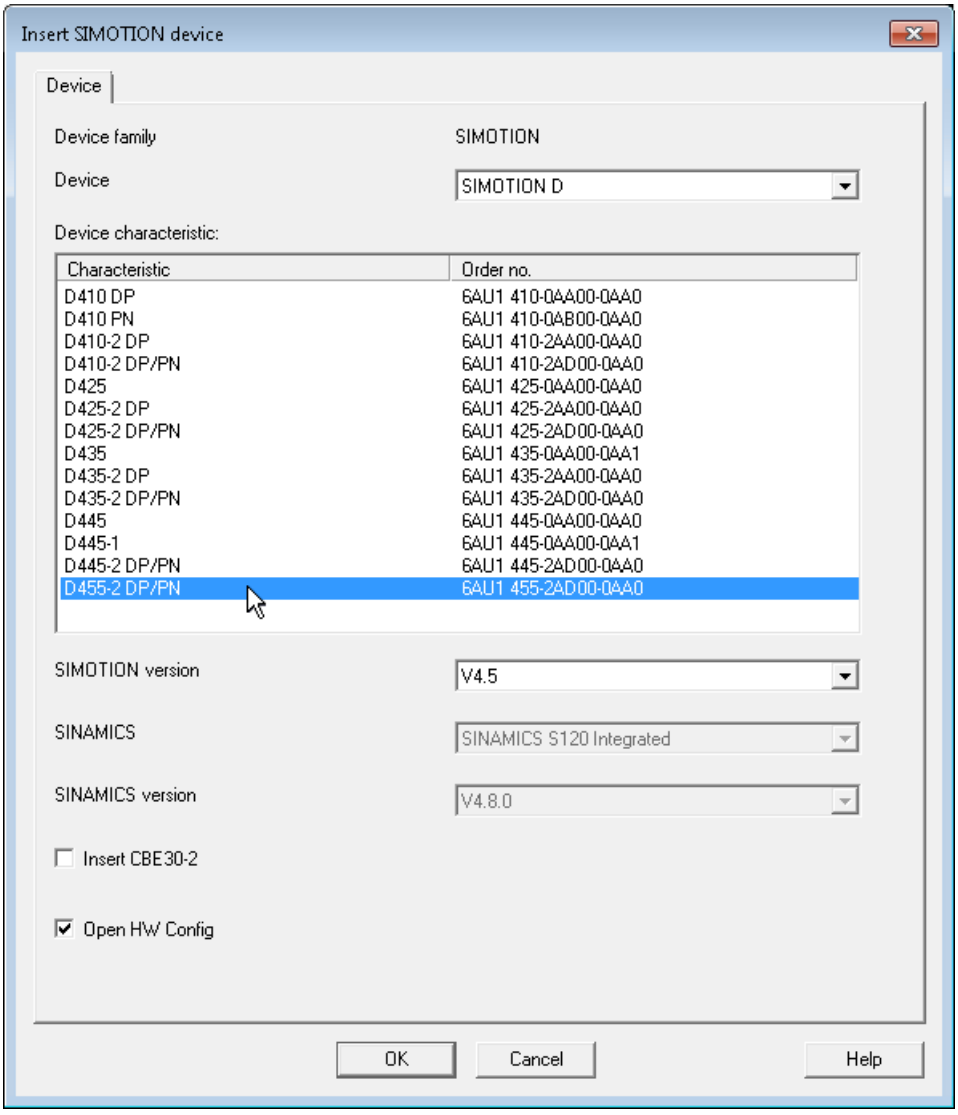


Figure 3-127 Insert SIMOTION device

- **Insert single drive unit** element. Double-click this icon to insert and commission a standalone drive (e.g. MM4 Basic). Although this drive cannot be linked to or configured in the master system of the project using a PROFIBUS DP connection, it appears in the project navigator within the project.
- Folder labeled **LIBRARIES**
- Folder labeled **MONITORING**

Open project

To open a project saved on the hard disk of the PC:

You can open projects that SIMOTION SCOUT has stored locally (e.g. on the hard disk).

1. Select the **Project > Open** menu.
2. In the **User projects** tab, select the desired project.
   If the project you want is not stored according to the default path: Click **Browse...** and follow the instructions on the screen.
3. Click **OK** to confirm.

If you want to open a project that was created in an older version, a message appears asking whether the project should be converted to the newer version. This function is available since SIMOTION SCOUT Version V3.1.

Note

The **Save as...** function saves all previously made changes with a new name (as of V4.4).

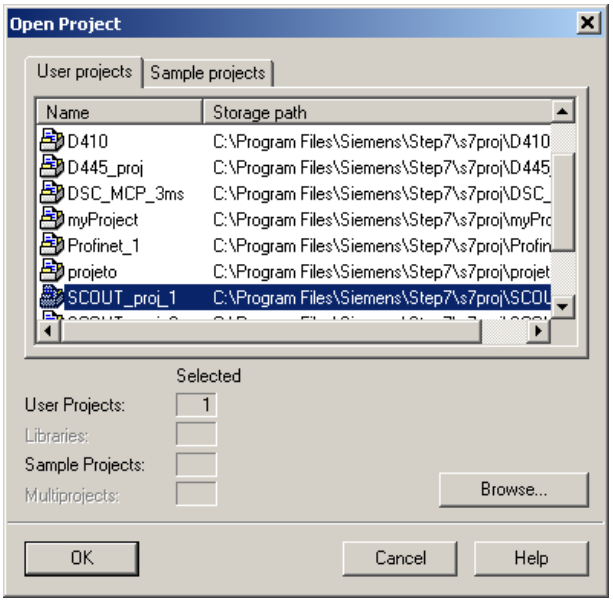


Figure 3-128 Opening a project stored on the hard disk

Note

The workbench displays one project at a time.

If you want to display two projects at the same time, e.g. in order to copy/paste parts of a project, start SIMOTION SCOUT a second time.

Saving and compiling

SIMOTION SCOUT distinguishes three commands in the main project menu that act differently when saving or saving and compiling:

- Save
- Save and compile changes
- Save and recompile all

Note

The project must be saved and compiled first before the download. For the default setting of SIMOTION SCOUT, these steps are performed automatically during the download.

Save

The project is saved to the hard disk. The changes are accepted into the project. No further processes (such as compilation or consistency checking) are triggered for the project.

Save and compile changes

On this command, the whole project is searched for changes. If a source is found which has been changed or has no compilation results, this and any linked sources are compiled and saved (e.g. during an FB call). Therefore only the changes are compiled. Use this command when performing your day-to-day tasks within a SCOUT version.

Save and recompile all

All sources of the entire project are recompiled with this command.

The **Save and recompile all** command is suitable if you are entirely sure that all the old data from older SCOUT versions has been removed and should be replaced with new compilation results. The command incorporates the following steps:

- Project-wide deletion of all compilation results
- Recompilation of all objects

Use this command if you specifically want to convert a project from an earlier SCOUT version to a later one. This involves accepting all the error correction and optimization elements in the new SCOUT. However in this case, the compilation results in the SIMOTION SCOUT and SIMOTION RT become inconsistent. The project navigator and object comparison feature then display the objects as "inconsistent" in ONLINE mode. You can only debug the project if you have loaded the entire project to the target system.

Note

The compilation results remain consistent provided there were no error messages and the compiler has not been changed.

Note

Libraries are compiled only for the device versions in which they are used.

Performing a consistency check

Checking the project for consistency and compiling the project

Before you download the project to the target system, the program sources must be compiled without error and the consistency ensured. In this process, I/O addresses or TO configurations are checked, for example.

1. Select the **Project > Save and compile all** menu.
2. SIMOTION SCOUT compiles all the changed sources.
3. Select the **Project > Check consistency** menu.

SIMOTION SCOUT checks, for example, whether all technology objects have been configured and that the source files have been compiled without errors.

To compile all programs automatically before a download and to check the consistency, the **Compile all programs before loading** and **Check consistency before loading** checkboxes must be activated via **Options > Settings** on the **Download** tab.

For further information, please refer to the online help for SIMOTION SCOUT.

Archiving and backing up projects on memory cards

Procedure

In SIMOTION SCOUT, you can back up your project to the memory card as a *.zip file.

Proceed as follows to archive the SIMOTION project on the memory card:

1. Open SIMOTION SCOUT and select the **Project -> Archive -> Normal...** menu command.
2. In the **Archive** dialog, select the SIMOTION project and save it to your drive (PG/PC).
3. Open the project.
4. Go online with SIMOTION.

5. In the project navigator, select the SIMOTION device and execute the **Target system -> Load -> Save archived project on card ... -> Normal ...** or **Reduced ...** menu command (see below).
6. In the dialog that is displayed, select the project and click **Open**. This saves the project to the CompactFlash Card as Project.zip in the directory USER\SIMOTION\HMI\PRJLOG.

Note

If you want to load the current project from the card, select the **Target system > Copy archived project from card to PG/PC...** menu command.

Prerequisite is that you have backed up the project with **Save archive project on card...** each time a change was made.

Reducing the project archive

If, for example, you want to send a project as an e-mail, the normal archived and zipped archives are too large.

You can influence the archiving of projects with the **Project > Archive > Reduced ...** function.

- Remove data from the archive to reduce the file size.
- Divide the archive into packets, so that you can create several ZIP files that are combined again when dearchiving.

For further information, please refer to the online help for SIMOTION SCOUT.

Additional references

Detailed information on loading data to the target device can be found in the *SIMOTION SCOUT Basic Functions* Function Manual.

Exporting and importing a project in XML format

A log of the export or import is displayed in the **XML export/import status display** tab in the detail view. The XML export log also contains a link to the exported file. Double-click this link to view the exported project data in the Internet browser.

Note

Projects are exported and imported version-neutral. However, if a project has version-specific properties, related errors may occur following import into another SCOUT version, e.g. when compiling ST programs.

Exporting project/objects in XML format

To export the entire project data in XML format:

1. Select **Project > Save and export**. The **Export Project** window with the **XML export/import status display** tab opens.
2. Select a target directory by clicking the **Browse** button. If you do not change the path, the export files are saved in the current project folder as standard. See also "Export of projects".
3. Click OK to confirm.
   The files in the target directory are deleted and the export of the project data is started. On completion of export, the successfully completed export and the target path are displayed in the XML export/import status display tab of the detail view.
4. Click the link silhouetted in blue to obtain more precise information about the exported project.

Importing an XML file

Select the file of the technology object / device in XML format to be imported to a new project. As import file, always select the XML file which has the name of the technology object / device. Normally the exported file is displayed in this field so that you only have to perform the following steps when you want to import another file.

To select the import file:

1. Click **Browse** to the right of the text field. A window opens.
2. Select the XML file that is to be imported.
   Only a technology object that is completely available can be imported. Always select the XML file which has the name of the previously exported technology object as the import file. If it is not possible to import the selected file, the import is aborted.
3. Click **Open** to confirm. The window closes and the source path is displayed.

Additional references

Please refer to the online help on this subject

Searching in the project

In the open project, you can search all the project data for variables or any text.

If you select a variable search, you can also search within the ST sources. All global and local variables at the declaration and use points are recognized.

Open the dialog using **Edit > Search in project...** on the menu bar or by using the shortcut **Ctrl + Shift + F**.

The results are displayed in the Search results tab of the detail view.

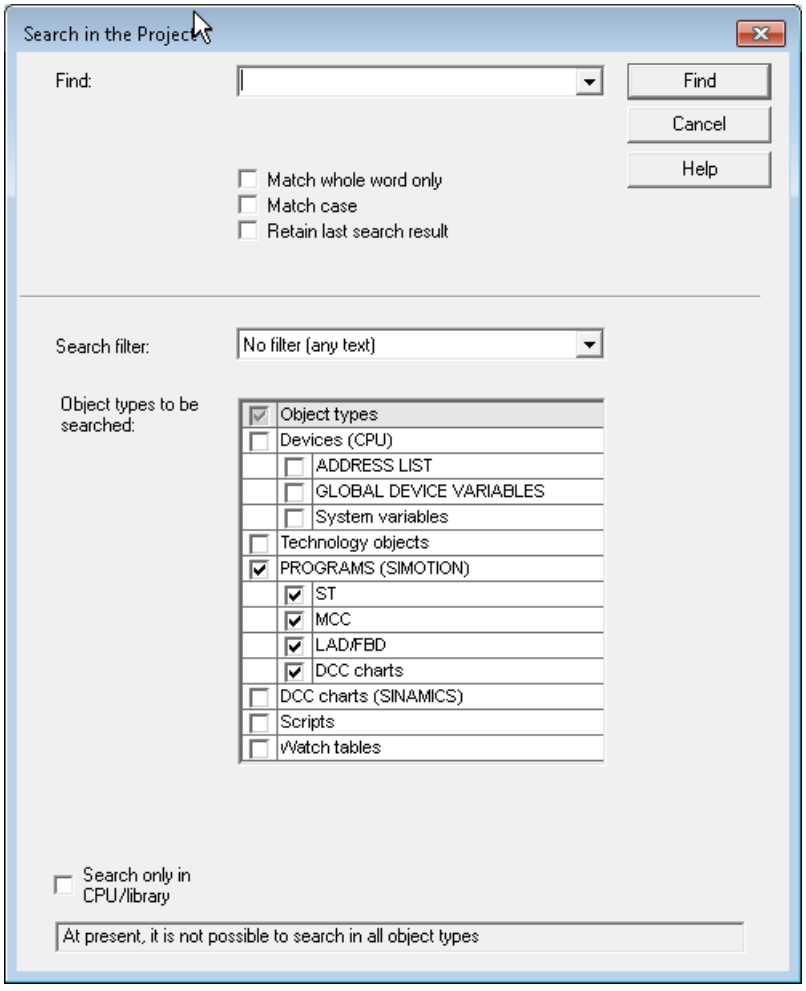


Figure 3-129 Searching in the project

Searching in CPU/Libraries only

If the "Search only in CPU/Library" checkbox is activated, a selection list of all the CPUs and user-defined libraries in the project appears. You can select one to search in this CPU/Library only.

For more information regarding this topic, see the online help.

Local search

You can search locally for a specific text in the various editors and the declaration tables. To start searching locally, open the search function with the **Ctrl+F** shortcut key while the current focus is on an editor (e.g. MCC) or the editor window for global and unit-global declarations and connections.

The *SIMOTION MCC Motion Control Chart Programming and Operating Manual* contains further information on local searches.

Replacing in the project

The **Replace in project** function is based on the **Search in project** function.

Open the dialog using **Edit > Replace in project...** on the menu bar or with the shortcut **Ctrl + Shift + G**.

When you carry out a replacement, both the results found and the replacement term are displayed in the "Search result" tab of the "detail view". The text can be edited again here.

Use the **Replace** button to replace all search results selected using the checkbox.

Boundary conditions:

- Only text can be entered.
- Replaced texts cannot be undone.

More information on this topic is available in:

- Online help.

Local replace

You can search locally for a specific text in the various editors and the declaration tables, and replace this text with freely selected text. To start the local replacement process, select the **Ctrl +H** shortcut key while the current focus is on an editor (e.g. the MCC editor) or the editor window for global and unit-global declarations and connections.

Printing projects

You can print various parts of a project or view them in the print preview in order to, for example, document settings, data, etc. The part to be printed depends on the element selected in the project navigator or the window active in the working area.

- An element is selected in the project navigator or a window is active in the working area. The data of the associated technology object, MCC chart, etc. is printed.
- The detail view is active. The data of the active tab of the detail view is printed.
- The project is selected in the project navigator. All data of the project (e.g. programs, execution system, technology objects) is printed.

Note

If you select **Print** or **Print preview** in the **Project** menu, the print dialog or the print preview is displayed immediately. If you select the context menu for **Print** or **Print preview** of the selected element in the project navigator, a window is displayed in which you can activate the components to be printed. If, for example, you only want to print certain parts of a technology object, select **Print** in the context menu.

For further information, please refer to the online help for SIMOTION SCOUT.

3.3.6.4 Configuring devices

Adding a SIMOTION device

You have created a project in SIMOTION SCOUT. The project navigator on the left of the SIMOTION SCOUT workbench is open. To insert a device, double-click the **Insert SIMOTION device** entry below the name of your project in the project navigator. The following dialog opens:

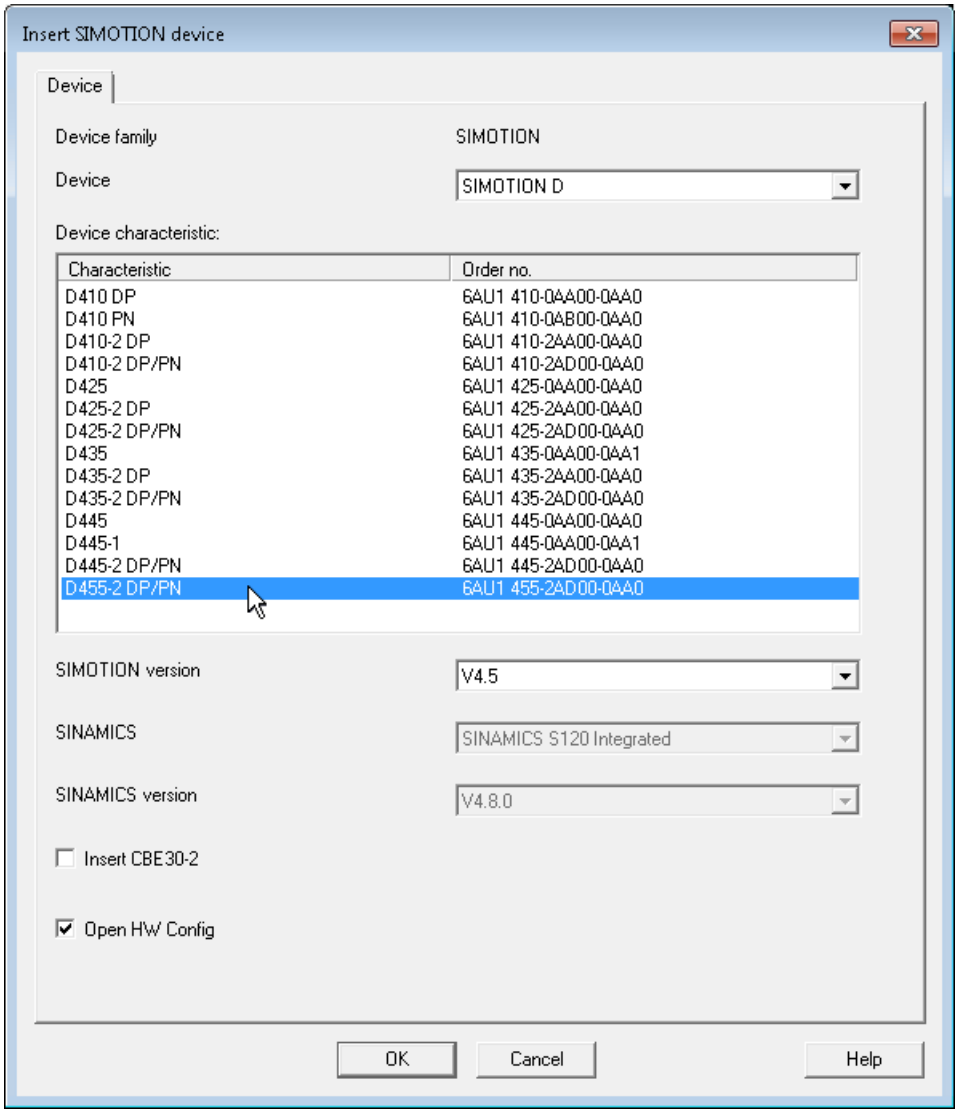


Figure 3-130 Insert SIMOTION device

Select the device platform you are working with at **Device**. The characteristics of the relevant platform then appear under **Device characteristic**.

At **SIMOTION Version**, select the firmware version of the device used.

Note

The selected firmware version must match the firmware version on the memory card of the SIMOTION device. Otherwise, you will receive an error message when going online with the device.

If the **Open HW Config** option is selected, the hardware configuration (Page 303) opens after the device has been inserted. Click **OK** to confirm.

Once you have created a SIMOTION device in the project via the **Insert SIMOTION device** element, the hardware configuration (**HW Config**) opens automatically.
In HW Config, you inform the SIMOTION system which hardware is present; for example:

- Type of the SIMOTION device
- Type of the I/O modules
- Type of the drives

In addition, you also specify which parameters the SIMOTION system should use, e.g.:

- Configuration of the SIMOTION device
- Configuration of the PROFIBUS/PROFINET
- Assignment of the hardware to PROFIBUS/PROFINET

The following sections explain how to use the HW Config tool:

- How to insert objects from the **hardware catalog**.
- How to edit objects.
- How to replace or delete objects.

Starting HW Config

HW Config can be started, e.g. to add a drive to the PROFIBUS.

To start HW Config:

- Double-click the appropriate SIMOTION device in the project navigator or
- Select the appropriate device in the project navigator and confirm your selection in the **Open HW configuration** context menu or
- Highlight the appropriate device in the project navigator and click **Edit > Open object** in the menu bar.

The HW Config program

The HW Config program is displayed as follows:

- A **hardware catalog** window
- A working window that is split into two:
  - In the upper half, you can see the rack or station frame with the CPU already inserted automatically in slot 2 (example for C240).
    Insert the objects from the hardware catalog and edit them here.
  - In the lower half, you can see detailed information about the selected objects.

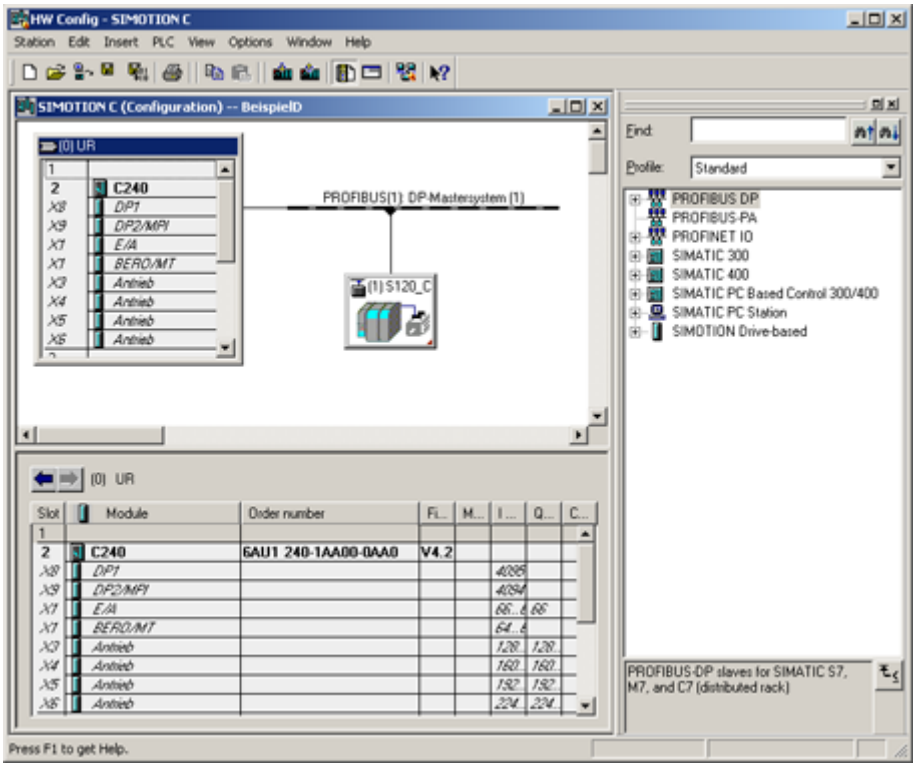


Figure 3-131 HW Config with opened hardware catalog and the CPU previously inserted in the module

HW Config: Opening the hardware catalog

You can open the hardware catalog in the following ways:

- Select the **Insert > Insert object** menu.
- Select the **View > Catalog** menu.

For further information, see Section Inserting a drive (Page 311).

SIMOTION devices in the hardware catalog

You can find the SIMOTION devices in the corresponding directories of the hardware catalog in HW Config.
With SIMOTION D, the SINAMICS version is selected in addition to the SIMOTION version. The SIMOTION version determines whether one or more SINAMICS versions are available.

Adding interfaces

After you have inserted the SIMOTION device, the dialog for configuring the interfaces opens

Note

With SIMOTION D, the PROFIBUS subnet **PROFIBUS Integrated DP master system** is created and configured automatically. The user may neither configure nor select it!

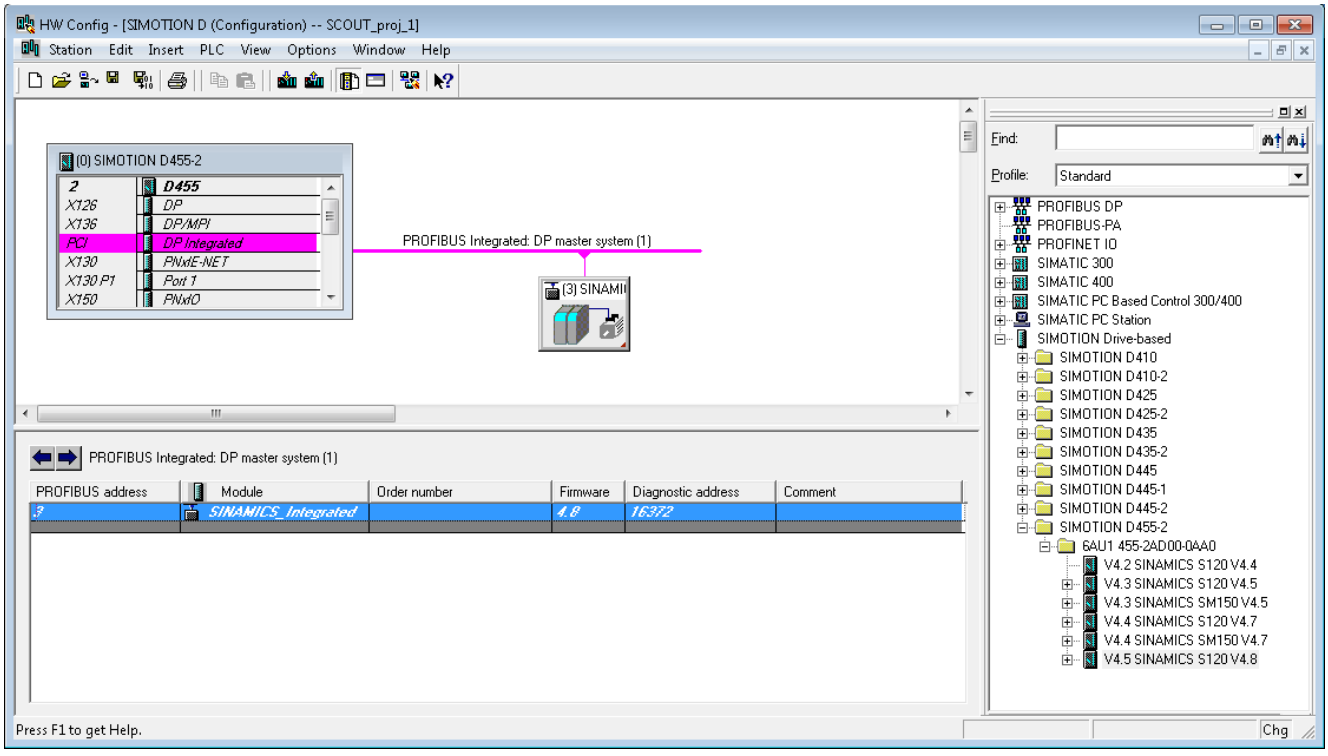


Figure 3-132 Hardware configuration of a D4x5

Detailed information on the configuration of the SIMOTION device is contained in the corresponding documentation for the SIMOTION devices.

Selecting technology packages

The technology packages (e.g. TP CAM, TP PATH, DCBlib) are available in various versions for installation purposes.

You can only use the functions of the technology objects selected if the technology objects are available in the target system. You can select the technology packages and their product versions (as well as the service packs and hotfix versions) for each SIMOTION device.

In the **Select Technology Packages** window, you can adopt a more selective approach when choosing the technology packages you want to use. To open this window, right-click a SIMOTION device in the project navigator. Then select the **Select Technology Packages** command in the context menu.

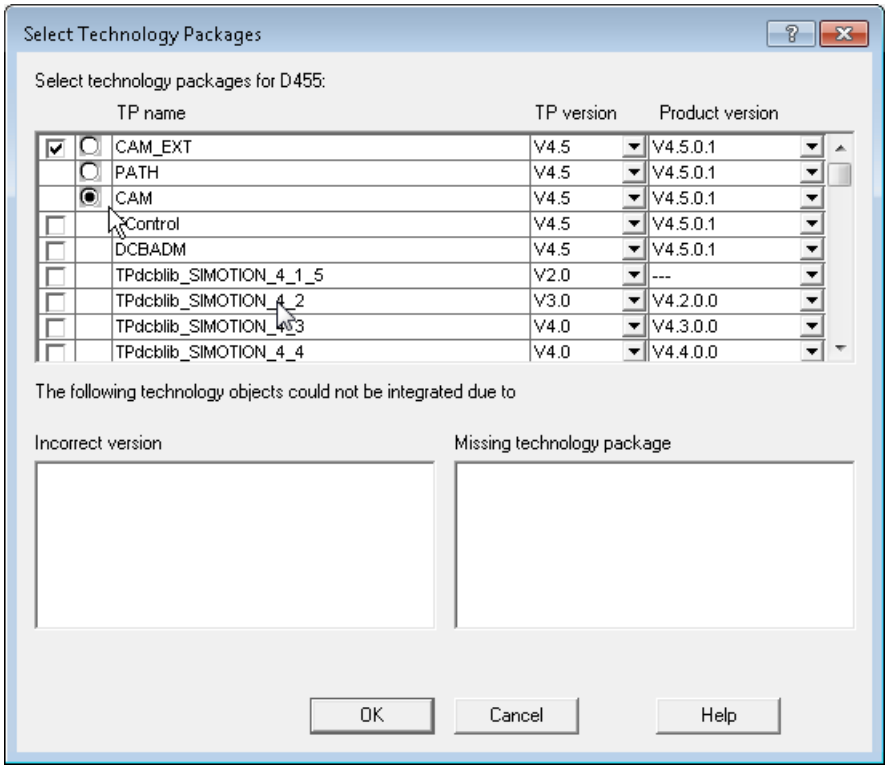


Figure 3-133 Selecting technology packages

Note

Device diagnostics can provide information on which technology package product version has been loaded to a CPU.

Loading technology packages to the target device

Technology packages are only loaded to the target device if no technology package has been loaded so far. If a technology package version changes, the technology package must be explicitly reloaded to the target device. Proceed as follows:

1. Select **Project > Download to target system** in SIMOTION SCOUT.
2. Click **Additional CPU options** in the **Download to target system** dialog.

3. Select the option **Replace product versions of the technology packages**.
4. Click **OK** to confirm.

For further information, please refer to the online help for SIMOTION SCOUT.

Connecting to the target system

Installing the interface card

Installing the interface card in the PG/PC

Communication between a SIMOTION device and a PG/PC requires an interface card for PROFIBUS or an Ethernet interface. PGs already have both, but PCs may need an interface card to be installed.
The PG/PC is used for configuration, parameter assignment, programming, and testing.

- Install the interface card in the PG/PC according to the manufacturer's installation instructions.
- Install the appropriate drivers on the PG/PC.
- Connect the interface card to the interface of the SIMOTION device indicated below using a suitable cable.

Table 3-27 Interface assignment for SIMOTION devices

SIMOTION device	Interface	As supplied
SIMOTION C230-2 / C240	X8 DP1	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X9 DP2 / MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X7 (Ethernet)	IP address: 169.254.11.22 Subnet: 255.255.0.0
SIMOTION C240 PN	X8 DP1	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X9 DP2 / MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X7 (Ethernet)	IP address: 169.254.11.22 Subnet: 255.255.0.0
	X11 (1 PROFINET interface with 3 ports)	Supplied without IP address and subnet
SIMOTION P350	X101 DP1	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X102 DP2 / MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	Ethernet	IP address: 169.254.11.22
SIMOTION P320-4	X1 Onboard Ethernet 1	Access to/from Windows: IP address: 169.254.11.21 Subnet: 255.255.0.0 Access to/from SIMOTION P Runtime: IP address: 169.254.11.22 Subnet: 255.255.0.0
	X3 PROFINET onboard interface with 3 ports	Supplied without IP address and subnet

SIMOTION device	Interface	As supplied
SIMOTION D4x5	X126 DP1	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X136 DP2 / MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X120 IE1/ OP (Ethernet)	IP address: 192.168.214.1 Subnet: 255.255.255.0
	X130 IE2/ NET (Ethernet)	IP address: 169.254.11.22 Subnet: 255.255.0.0
	With optionally inserted CBE30: X1400 (one PROFINET interface with 4 ports)	Supplied without IP address and subnet
SIMOTION D4x5-2 DP	X126 DP	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X136 DP/MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X120 P1 PN/IE-OP (Ethernet)	Supplied without IP address and subnet
	X127 P1 PN/IE (Ethernet)	IP address: 169.254.11.22 Subnet mask: 255.255.0.0
	X130 P1 PN/IE-NET (Ethernet)	Supplied without IP address and subnet
SIMOTION D4x5-2 DP/PN	X126 DP	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X136 DP/MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X127 P1 PN/IE (Ethernet)	IP address: 169.254.11.22 Subnet mask: 255.255.0.0
	X130 P1 PN/IE-NET (Ethernet)	Supplied without IP address and subnet
	X150 P1-P3 PN (1 PROFINET interface with 3 ports)	Supplied without IP address and subnet
	With the optional inserted CBE30-2: X1400 (1 PROFINET interface with 4 ports)	Supplied without IP address and subnet
SIMOTION D410 DP	X21 DP	PROFIBUS address 2, baud rate 1.5 Mbit/s
SIMOTION D410 PN	X200 and X201 (1 PROFINET interface with 2 ports)	Supplied without IP address and subnet
SIMOTION D410-2 DP	X21 DP/MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X24 DP	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X127 P1 PN/IE (Ethernet)	IP address: 169.254.11.22 Subnet mask: 255.255.0.0
SIMOTION D410-2 DP/PN	X21 DP/MPI	PROFIBUS address 2, baud rate 1.5 Mbit/s
	X127 P1 PN/IE (Ethernet)	IP address: 169.254.11.22 Subnet mask: 255.255.0.0
	X150 P1, P2 (1 PROFINET interface with 2 ports)	Supplied without IP address and subnet

Set PG/PC interface

Setting interfaces

Proceed as follows:

1. Start SIMOTION SCOUT.
2. Select the **Options > Set PG/PC interface** menu.
   The **Set PG/PC Interface** window opens.

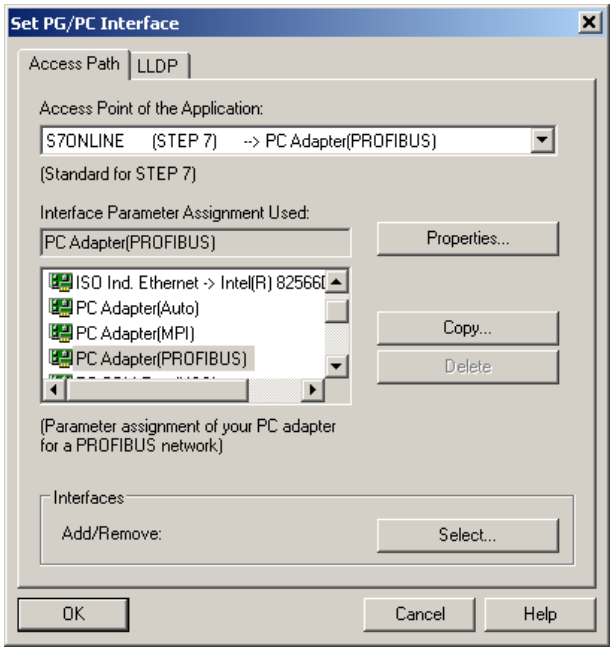


Figure 3-134 Set PG/PC interface (example)

3. Select the access point of the application.
4. Click **Select**.
   The **Install/Remove Interfaces** window opens.

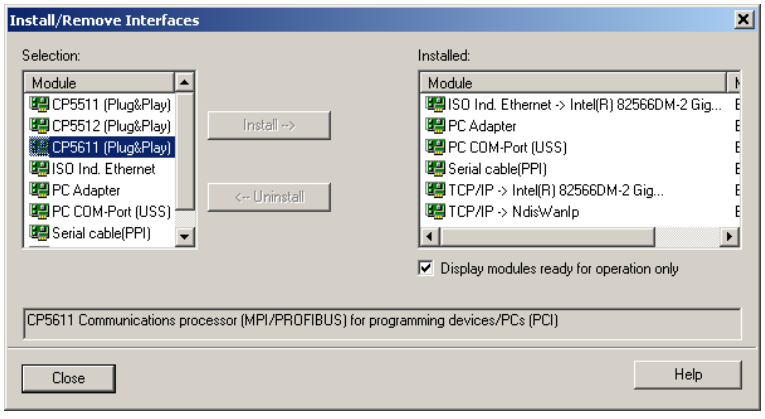


Figure 3-135 Installing/uninstalling interfaces

5. Select the module to be installed.
6. Click **Install**.

7. Click **Close** to exit the dialog.
8. Confirm the settings by clicking **OK**.

Installing/uninstalling other interfaces

Proceed as follows to install or uninstall additional interfaces:

1. In the **Set PG/PC Interface** window, select **S7ONLINE** as the access point of the application.
2. Click **Select** at **Add/remove interfaces**.
3. In the left-hand field, select the interface that you want to install, and click **Install**.
   Or select the interface that you want to uninstall in the right-hand field and click **Uninstall**.
4. Click **Close**.
5. Select the interface to be used in the **Used interface parameterization** field.
   The list of the used interface parameterization will be taken from SIMATIC STEP 7.
   With the PC Adapter (Auto) setting, the interface to which the PG/PC is connected is examined. This function automatically determines the current parameterization of the respective interface.
   Detailed information on the individual interfaces of the SIMOTION devices can be found in the relevant commissioning manuals.
6. Click **Properties** to parameterize the interface.
7. Click **OK** to confirm.

Specifying the access point

Selecting the access point

The addresses for a device to go online depend on the access point **S7ONLINE** or the alternative access point **DEVICE**.

Depending on the selected access point, you can go online via different interfaces.

Proceed as follows to select the access point:

1. Open the project.
2. Select the SIMOTION device in the project navigator.
3. Right-click and select **Properties...** in the context menu.
   The **Properties - [CPU]** window opens.
4. Go to the **Device / access point** tab. You can select the access point and specify the addresses.
5. Click **OK** to close the dialog.

Communication via PROFIBUS DP

PROFIBUS is a powerful, open, and robust bus system, which guarantees trouble-free communication. The system is fully standardized, which enables trouble-free connection of standardized components from a variety of manufacturers. Configuration, commissioning, and troubleshooting can be carried out from any side. This results in user-defined communication relationships that are very versatile, simple to implement, and easy to change.

The high-speed, cyclic exchange of data with field devices (distributed I/O) is carried out via the PROFIBUS DP protocol. Expansion of the protocol to include isochronous mode also integrates the drive components into communication via PROFIBUS DP. The individual components can be connected via the integrated interfaces, connections, interface modules or communications processors.
Important information on communication with PROFIBUS is contained in the following catalog: SIMOTION, SINAMICS S120 and Motors for Production Machines, PM 21 Catalog

The following PROFIBUS DP subnet applications are possible:

- PROFIBUS DP isochronous
  Compatible extension to the standard PROFIBUS DP:
  - Equidistance (constant synchronous bus cycle clock) enables synchronized time slices for master/slave applications.
  - Data exchange broadcast enables slave-to-slave communication.

  Digital drives (e.g. SINAMICS, MICROMASTER 4xx, ...) can be connected via PROFIBUS DP. In order to be able to operate the SIMOTION device with two isochronous PROFIBUS DP subnets, isochronous mode and the same bus cycle must be set. The cycle clock of the subnet of the second PROFIBUS interface is synchronized to the cycle clock of the subnet of the first PROFIBUS interface.
  If the bus cycle does not agree, the second PROFIBUS interface of the SIMOTION device will be operated as a PROFIBUS DP subnet.
- PROFIBUS DP (MPI)
  Standard PROFIBUS DP with DP-V0 and optional DP-V1 or DP-V2 functionality.
  - For the connection of distributed I/Os
  - For the connection to a higher-level automation system
  - For the connection of HMI devices
  - For the connection to the SCOUT engineering system

Ethernet communication

SIMOTION devices can also be connected to a PG/PC via Industrial Ethernet. Industrial Ethernet is a communication network with a transmission rate of 1 Gbit/s.

For additional information, refer to the Manual "SIMATIC NET Industrial Twisted Pair and Fiber Optic Networks." See the *SIMOTION Documentation Overview* for the Article No.

Examples for the use of Industrial Ethernet:

- For the connection of HMI devices
- For connection to the SCOUT engineering system
- For communication via UDP (User Datagram Protocol)
- TCP/IP
- IT DIAG

Communication via PROFINET

SIMOTION devices can be connected to a PROFINET subnet via interface modules or the onboard PROFINET interface. With PROFINET IO IRT/RT, IT services can be performed in parallel to realtime communication via an Ethernet cable. PROFINET IO supports parallel operation of:

- IRT (isochronous realtime Ethernet)
- RT (realtime Ethernet)
- TCP/IP, UDP, http ... (standard Ethernet services)
- For mixed operation of IRT and RT it has to be observed that the IRT-compatible devices must form an IRT domain, i.e. there must not be any non-IRT devices on the data transmission link between the IRT devices.

Additional references

Please refer to the following documents on this subject

- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- SIMOTION D410 Commissioning Manual
- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual
- SIMOTION C Operating Instructions
- SIMOTION Communication System Manual

and the SIMOTION SCOUT online help.

Inserting a drive

Drives with SIMOTION

Note the following:

Drives must be differentiated between:

- Drives that are connected to PROFIBUS/PROFINET
- Drives that are connected directly to the SIMOTION device

The following applies to drives connected to PROFIBUS/PROFINET:

- Only drives that meet the requirements of PROFIdrive profile V3.0 can be connected:
  - SINAMICS
  - MICROMASTER (PROFIBUS only)
  - COMBIMASTER 411 (PROFIBUS only)
  - MASTERDRIVES (MC, VC) (PROFIBUS only)
  - SIMODRIVE 611U (PROFIBUS only)
  - SIMODRIVE POSMO (CA, CD, SI) (PROFIBUS only)
  - ADI4 (analog drive interface for 4 axes)
  - IM 174 (interface module for 4 axes)
- These drives are taken into the hardware configuration.
- The drives of the SINAMICS and MICROMASTER drive families can be configured, assigned parameters, and commissioned with SIMOTION SCOUT.

All analog drives can be connected via one of the four analog interfaces on the SIMOTION C230-2/C240 or via the ADI4 and IM174 PROFIBUS module.

- These drives are inserted into the project via the hardware configuration.
- The SIMOTION device requires only the actual values from the encoder at the input and issues the setpoints at the analog output.
- The drives must be configured, assigned parameters, and commissioned directly on the corresponding device.

With SIMOTION D, drives on SINAMICS Integrated are not inserted via the HW Config. For more detailed information, see the *D Commissioning and Hardware Installation Manual*.

Inserting a SINAMICS drive on PROFIBUS DP

Initial situation

- The SIMOTION device is created in SIMOTION SCOUT
- A PROFIBUS subnet has been configured, for example, as **PROFIBUS DP isochronous** (by selecting **Edit > Isochronous operation**)
- The subnet interface is the master on this subnet of the SIMOTION device

Proceed as follows:

1. Open the **PROFIBUS DP** folder in the hardware catalog.
2. Open the **SINAMICS** subfolder.

3. Use drag-and-drop to drag an entry to the isochronous PROFIBUS subnet of the SIMOTION device.
4. Double-click the SINAMICS drive **Properties** to open them and set a cycle clock.

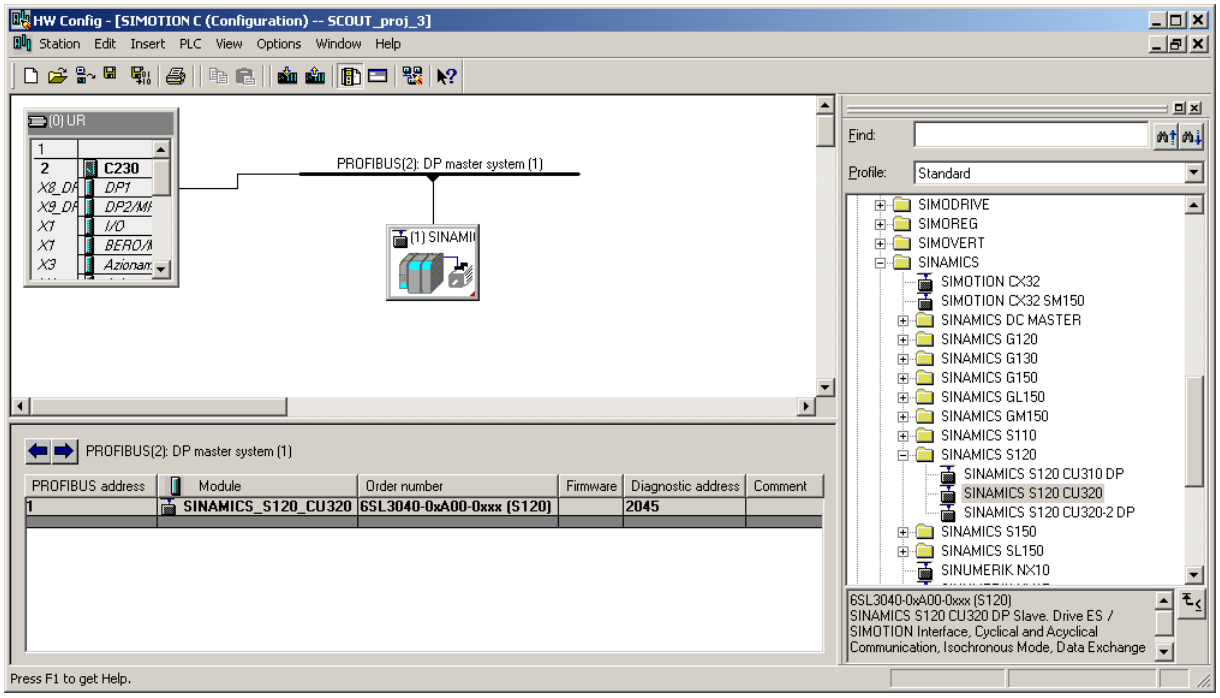


Figure 3-136 Inserting a SINAMICS drive

Inserting a SINAMICS drive on PROFINET IO

Initial situation

- The SIMOTION device is created in SIMOTION SCOUT
- A PROFINET IO system and an Ethernet subnet are configured

Proceed as follows:

1. Open the HW Config of the SIMOTION device.
2. Open the **PROFINET IO / Drives / SINAMICS** folder in the hardware catalog.

3. Select the SINAMICS S120 CBE20 drive.
4. Use drag-and-drop to drag the drive to the PROFINET IO subnet of the SIMOTION device.

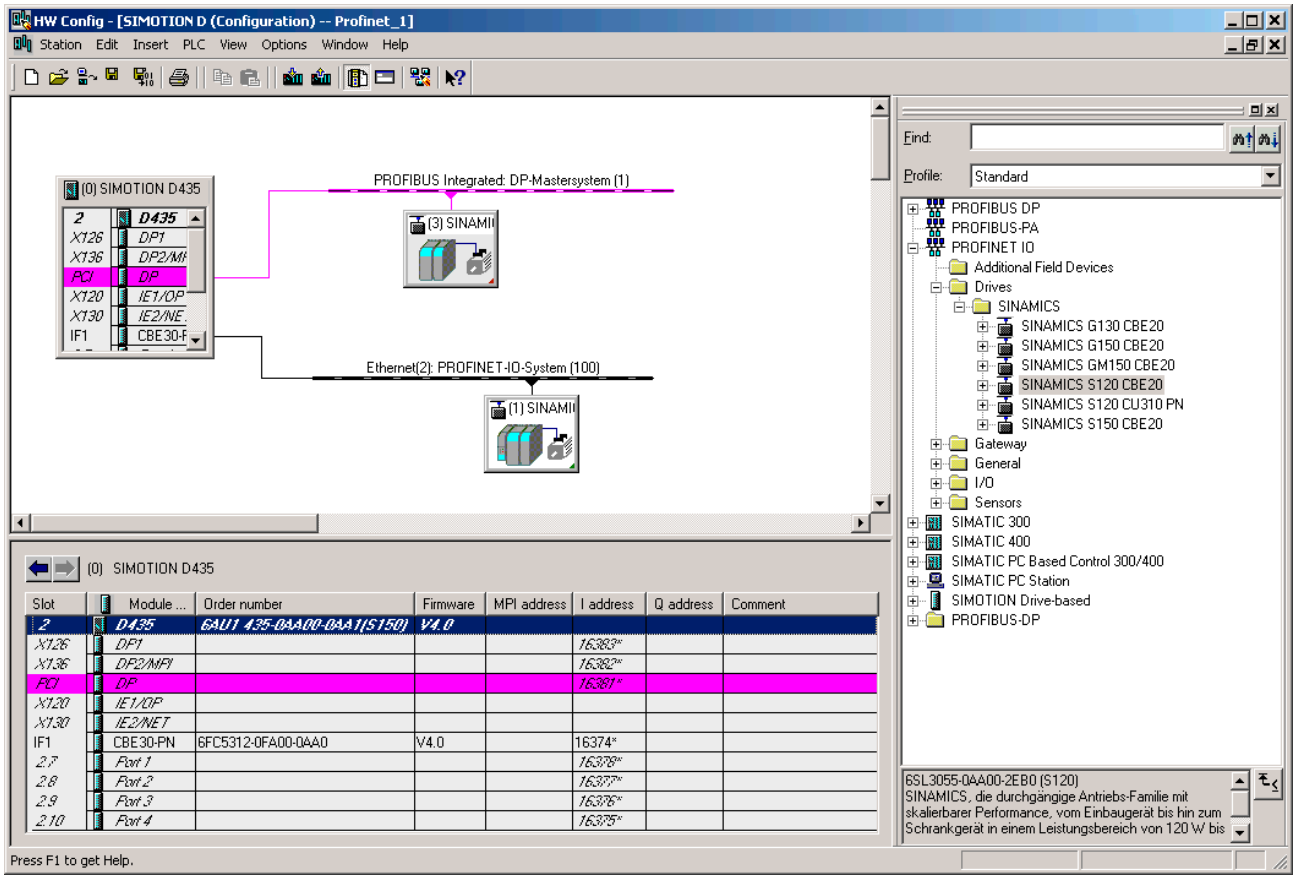


Figure 3-137 Inserting a SINAMICS drive with PROFINET

Configuration for PROFINET IRT

Information for the configuration of PROFINET IRT is contained in the Communications System Manual, Inserting and Configuring SINAMICS S120 section, as well as at the FAQ - SIMOTION & SINAMICS: Distributed synchronous operation and isochronous mode via PROFINET IRT (https://support.industry.siemens.com/cs/de/en/view/38486079)

Commissioning the drives

Drives that can be commissioned with SIMOTION SCOUT

You can use SIMOTION SCOUT to assign parameters to the following drives:

- SINAMICS
- MICROMASTER
- COMBIMASTER 411

For all other drives or for third-party drives, the appropriate commissioning program for the specific drive must be used. If required, you must commission these drives directly on the drive.

Table 3-28 Commissioning of various drives

Drive	Connection	Commissioning
SINAMICS S110, S120, S150, G110 CPM100, G120, G130, G150, GM150	PROFIBUS DP or analog interface	With SIMOTION SCOUT or STARTER
SINAMICS S120	PROFINET	With SIMOTION SCOUT or STARTER
MICROMASTER 410	analog interface	With SIMOTION SCOUT or DRIVE ES Basic or STARTER
MICROMASTER 420/430/440	PROFIBUS DP or analog interface	With SIMOTION SCOUT or DRIVE ES Basic or STARTER
COMBIMASTER 411	PROFIBUS DP or analog interface	With SIMOTION SCOUT or DRIVE ES Basic or STARTER
SIMODRIVE 611U	PROFIBUS DP or analog interface	With SIMOCOM U or DRIVE ES Basic
MASTERDRIVES MC, VC	PROFIBUS DP or analog interface	With SIMOVIS or DRIVE ES Basic
SIMODRIVE POSMO CA; CD, SI	PROFIBUS DP	With SIMOCOM U or DRIVE ES Basic
Third-party drives	Analog interface or PROFIBUS DP according to PROFIdrive profile V3.02	With commissioning tool supplied by the manufacturer

Note

If the Drive-ES Basic commissioning program is installed, it can be called by SIMOTION SCOUT in the project navigator via **Commissioning**.

SINAMICS S120 on SIMOTION

STARTER functionality in SIMOTION SCOUT

It is possible to parameterize drives directly with SIMOTION SCOUT. The following example describes how to insert a SINAMICS drive in SIMOTION SCOUT.

Requirements:

- SIMOTION SCOUT has been installed on the PG/PC
- A SIMOTION SCOUT has been created

Note

You can create the SINAMICS drive with PROFIBUS or PROFINET.

You must execute the following steps:

- Insert SIMOTION device
- Configure SINAMICS

Commissioning in SIMOTION SCOUT

1. Open the SIMOTION project in SIMOTION SCOUT.
2. Double-click **Insert SIMOTION device**.
3. Select a SIMOTION device.
4. Select the variant.
5. Click **OK** to confirm.
6. Select the PROFIBUS interface or Ethernet.
   If you have selected Ethernet, select the interface parameterization of the PG/PC.
7. Click **OK** to confirm.
8. With SIMOTION C and P only: Insert a SINAMICS device (see Inserting a drive (Page 311)).
9. Select **Station > Save and compile** in the HW Config and then close HW Config.
10. Open the structure tree of the SIMOTION device in the project navigator.
11. Open the structure tree of the drive (SINAMICS or SINAMICS_Integrated) in the project navigator.
12. Double-click **Configure drive unit**.
    The Configuration - Option Module window opens.
13. Run through the drive wizard.

The STARTER functionality now starts in SIMOTION SCOUT.

The configuring of the SINAMICS Integrated drive for SIMOTION D is described in the SIMOTION D Commissioning and Hardware Installation Manuals.

Note

You can insert a standalone drive (e.g. SINAMICS S120) with the "Insert single drive unit" element in the project navigator. It is commissioned using wizards in the working area of the workbench that contains the STARTER functionality.

Configuring the infeed

Overview

The integrated SINAMICS control unit only starts the drive when the infeed is ready. The project must therefore know the interface via which the drive receives the ready signal of the infeed.

Two cases must be distinguished here:

- Infeed with DRIVE-CLiQ interface
  If an infeed with DRIVE-CLiQ connection has already been created, the ready signal of the infeed (r0863.0) is automatically interconnected with "Infeed operation, p0864" of the drive when drives are inserted (only applies to drives that are attached to the same drive unit as the infeed).
  Further information about the infeed with the DRIVE-CLiQ interface is contained in the SIMOTION D Commissioning and Hardware Installation Manuals.
- Infeed without DRIVE-CLiQ interface
  If you are using an infeed without a DRIVE-CLiQ interface, e.g. a Smart Line Module, you must wire the ready signal of the infeed via terminals. The following section describes the procedure.

Configure the infeed without a DRIVE-CLiQ interface

Interconnecting the ready signal of the infeed

An infeed without DRIVE-CLiQ interface provides the ready signal (p0863.0) via an output terminal. In the project, you specify on which input (r0722) of SINAMICS Integrated the signal is active. The drive supplied by the infeed uses the signal as a ready signal (p0864).

Note

A drive on SINAMICS Integrated of a SIMOTION D can only be moved if the infeed is ready.

Requirements

- You have configured a drive.
- SIMOTION SCOUT is in online mode.

Procedure

In the next step, in order to wire the ready signal of the infeed (terminal "DO: Ready" of the infeed) with the DI 0 of the D4x5-2, proceed as follows in case you have not already established the interconnection during processing using the drive wizard:

1. Navigate in the project navigator to the **Drives** folder and double-click **Configuration** below the drive.

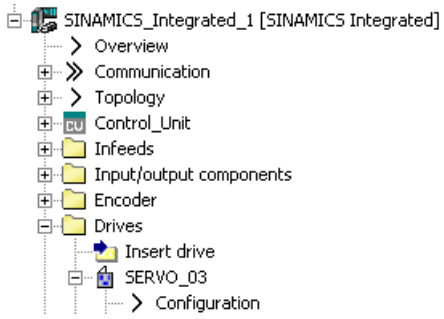


Figure 3-138 Configuring a drive

2. Click in the working area on the **Configure DDS** button with the yellow background.

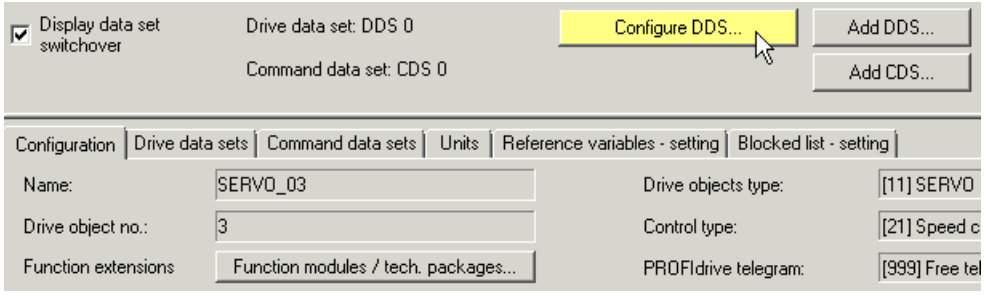


Figure 3-139 Configuring DDS

The drive wizard opens.

3. Click **Next** in the drive wizard until you reach the dialog **Configuration - SINAMICS_Integrated - BICO power unit**.

4. In the input field **p0864**, select the digital input (e.g. DI 0) to which the ready signal of the infeed is wired.

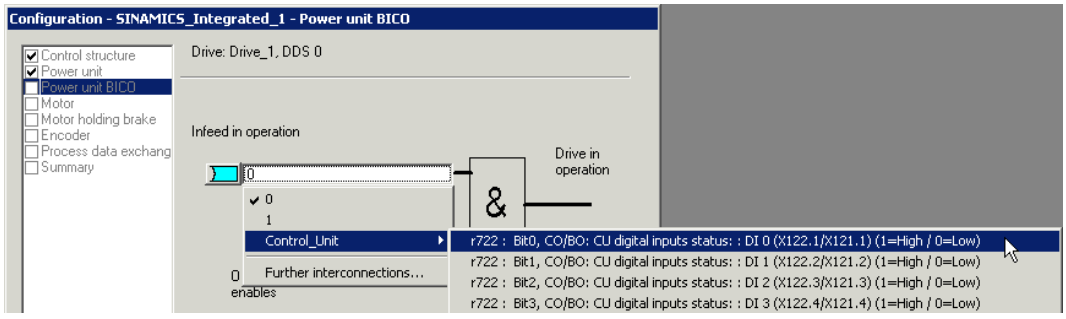


Figure 3-140 BICO interconnection

5. Click **Next**. Run through all the other dialogs without change until the final **Summary** dialog.

6. Click **Finish**.
Thus, the configuration is finished.

Testing the drive with the drive control panel

Drive control panel

The drive control panel is used to control and monitor individual drives. With the drive control panel, you can traverse drives and can:

- Test each part of the system individually before the drives are traversed in a coordinated manner by means of a program.
- Test in a fault situation whether the individual drives can be traversed by the drive control panel at all, or whether there are already problems here.

⚠ WARNING

This software may be used only when the associated safety instructions are observed! The non-observance can cause injury to persons or damage to material!

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns from the higher-level control have no effect.

The **Stop with preconfigured braking ramp** function is not guaranteed in all operating states. Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Requirements

- You have created and fully configured a drive in the project.
- You have stored and compiled the project. The project is consistent.
- SIMOTION SCOUT is in online mode.
- The project with the drive parameterization has been loaded to the target system.

Opening the drive control panel

To open the drive control panel, proceed as follows:

1. In the project navigator, navigate to the drive unit and open the **Drives** folder.
2. Open the **Commissioning** entry and double-click **Control panel**.
   The drive control panel opens in the detail view.

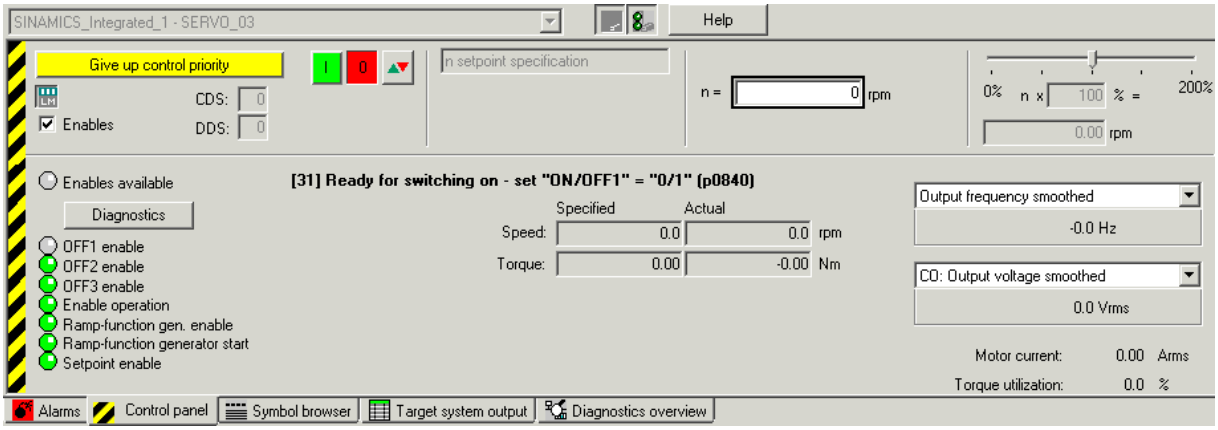


Figure 3-141 Drive control panel

Using the drive control panel

Assuming control priority

To assume control priority, proceed as follows:

1. Click the  button to show the control area in order to test the motor speed specified at **n=**.
2. Click the  button to show the diagnostics area in order to display and evaluate the defined parameters.
   You can determine the status of the required enables by clicking the **Diagnostics** button.
3. In the selection list, select the drive that you want to control or monitor using the PG/PC, **SINAMICS_Integrated_1 - SERVO_03** in the example.
4. Specify the parameters that you want to display in the selection lists on the bottom right-hand side of the diagnostics area. The values are displayed below.

5. Select **Assume control priority** to establish the connection to the PG/PC. The **Assume control priority** dialog is displayed.

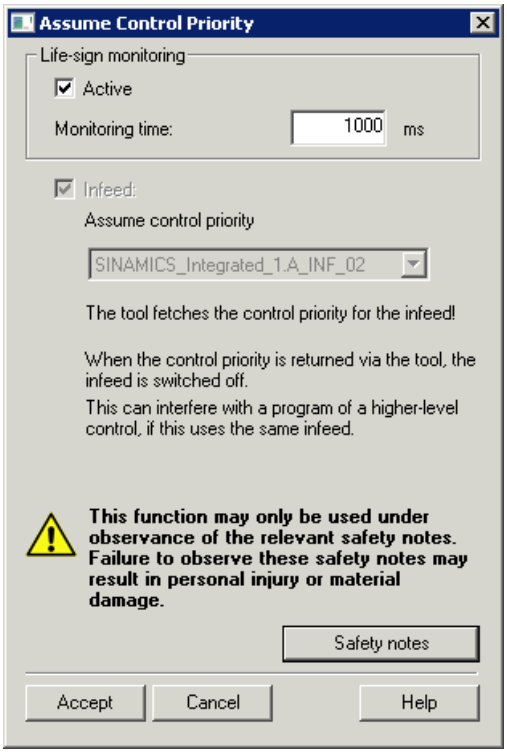


Figure 3-142 Assuming control priority

6. Consider the following infeed options for the type of connection you are using.
   - If you are using an infeed with a DRIVE-CLiQ interface, select the infeed for which the control priority is to be assumed at **Infeed** in the **Assume control priority** dialog. Activate the **Infeed** option if the infeed control priority is to be assumed and activated.
   - If you are using an infeed without a DRIVE-CLiQ interface, you must interconnect the **Infeed operation** signal (drive parameter p0864) yourself. In this case, the dialog does not contain any fields for selecting infeed.
   - If the signal of the **Closed-loop control operation** infeed is already BICO interconnected with the drive, the infeed is specified permanently. No selection is possible in the dialog, the fields are grayed out.
7. Observe the safety regulations and confirm with **Accept**.

The PG/PC has the control priority; the button caption has changed to **Return control priority**.

Note

You can also stop the drive at any time by pressing the SPACEBAR.

Activating the infeed

To switch on the infeed, proceed as follows:

1. Click the button for more information.

Activating enables

To activate the enables, proceed as follows:

1. Activate the **Enables** option to enable the drive.
   The ON and OFF buttons and are activated and the enable states are indicated by the LEDs.
   Click the "Diagnostics" button to show any missing enables.

Testing the drive

To test the drive, proceed as follows:

1. Specify a speed setpoint in the **n=** field.
2. Use the slider to set the current speed **setpoint** in relation to the maximum speed **setpoint (100%)**.
3. Click the green switch to start the drive.
   Or click the switch to control the drive in jog mode. The motion continues as long as you keep the switch pressed.
   In the diagnostics area you can monitor the set speed/torque as well as the actual speed/ torque.

Exiting the drive control panel

To exit the drive control panel, proceed as follows:

1. Click the red button to stop the drive.
2. Deactivate the **Enables** option.
   The LEDs are grayed out.
3. Deactivate the infeed. To do this, click **Infeed on/off**.
4. Click the **Return control priority** button to return the control priority of the control panel.
5. On the menu bar, select **Project > Disconnect from target system** to switch to offline mode.

Result

The drive can be operated. The drive configuration is complete.

3.3.6.5 Creating and testing an axis

TO axis technology object

Note

You create the technology object (TO) axis in SIMOTION SCOUT TIA.

You can create the axis both in offline and online mode.

Technology objects represent the respective real objects, e.g. a position axis, in the controller.

The technology object axis offers the user a technological view of the drive and the encoder (actuator and sensor) and provides technological functions, such as communication with the drive, actual value processing, position control and positioning functionality. It executes control and motion commands and indicates states and actual values.

Requirements

If you want to insert a real axis, you require a functional drive.

If you want to insert a virtual axis, you do not require a functional drive.

Note

Virtual axis

All axis types can also be virtual axes, i.e. they do not control real drives but are used as auxiliary axes, e.g. as a leading axis for several following axes (line shaft).

Axis technologies

You can use the following axis technologies:

- Speed-controlled axis
  Motion control is performed using a speed specification without position control.
- Positioning axis
  Motions are position-controlled.
- Synchronous axis
  The synchronous axis creates a grouping of the following axis and synchronous object.
- Path axis
  The path axis type can be interconnected with a path object.
  The path object can be used to calculate and traverse a linear, circular or polynomial path in the 2D/3D coordinate system for at least two path axes and up to three path axes.

Axis wizard

SIMOTION SCOUT provides the axis wizard for creating a new axis. Using the wizard, you interconnect the TO axis with a drive.

Note

Axis wizard - restriction

The wizard can only be run through once.

Subsequent changes to the axis configuration are possible in the corresponding dialogs of the TO axis.

Configuring Axes

Procedure

To insert an axis with the axis wizard and to interconnect with drive and encoder, proceed as follows:

1. Navigate to the **AXES** folder in the project navigator and open this folder with a double-click.
2. Select **Insert axis**.
   The **Insert axis** dialog opens.

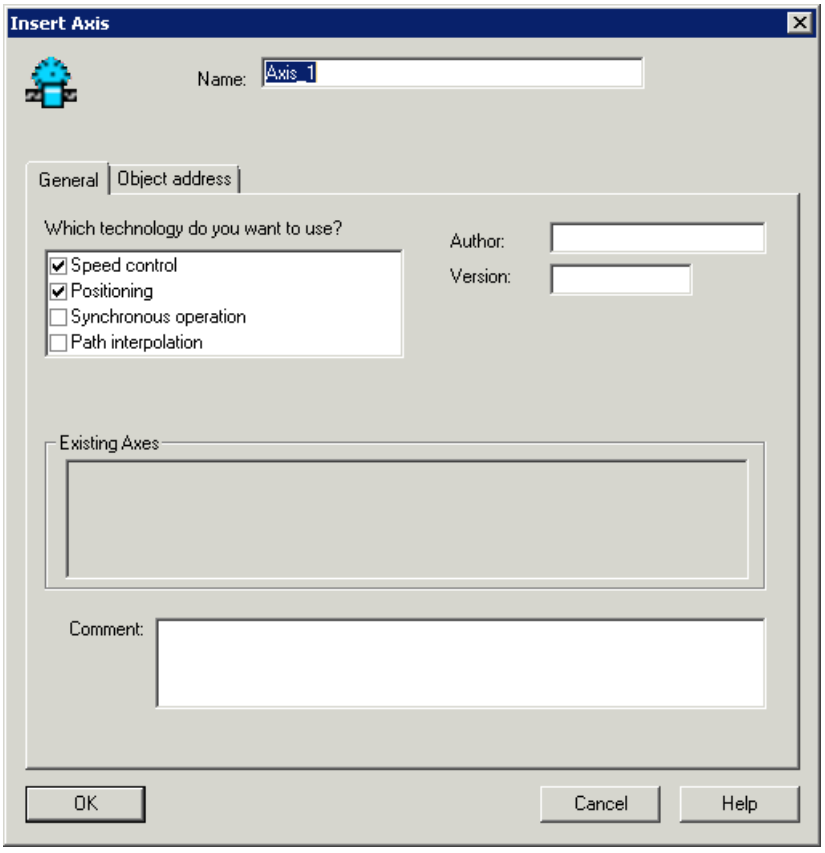


Figure 3-143 Creating an axis

3. Name the axis and select the axis technology that you want to use.
   Note here that the **Positioning** technology always requires the **Speed control** technology. Thus, this technology cannot be deactivated.

4. Click **OK** to exit the dialog.
   The **Axis configuration - Axis type** dialog opens.

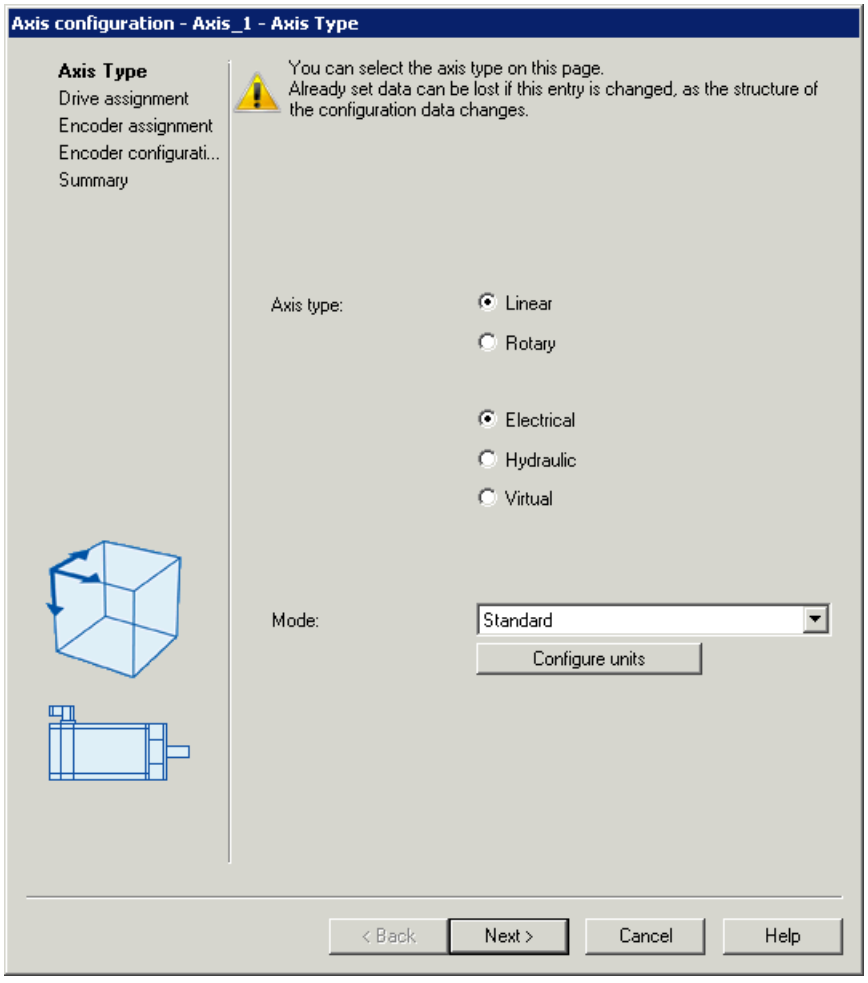


Figure 3-144 Axis configuration - defining the axis type

5. Select the axis type, and specify it. Select the mode for electric axes or hydraulic axes and confirm with **Next>**.
   The **Axis configuration - Drive assignment** dialog opens.

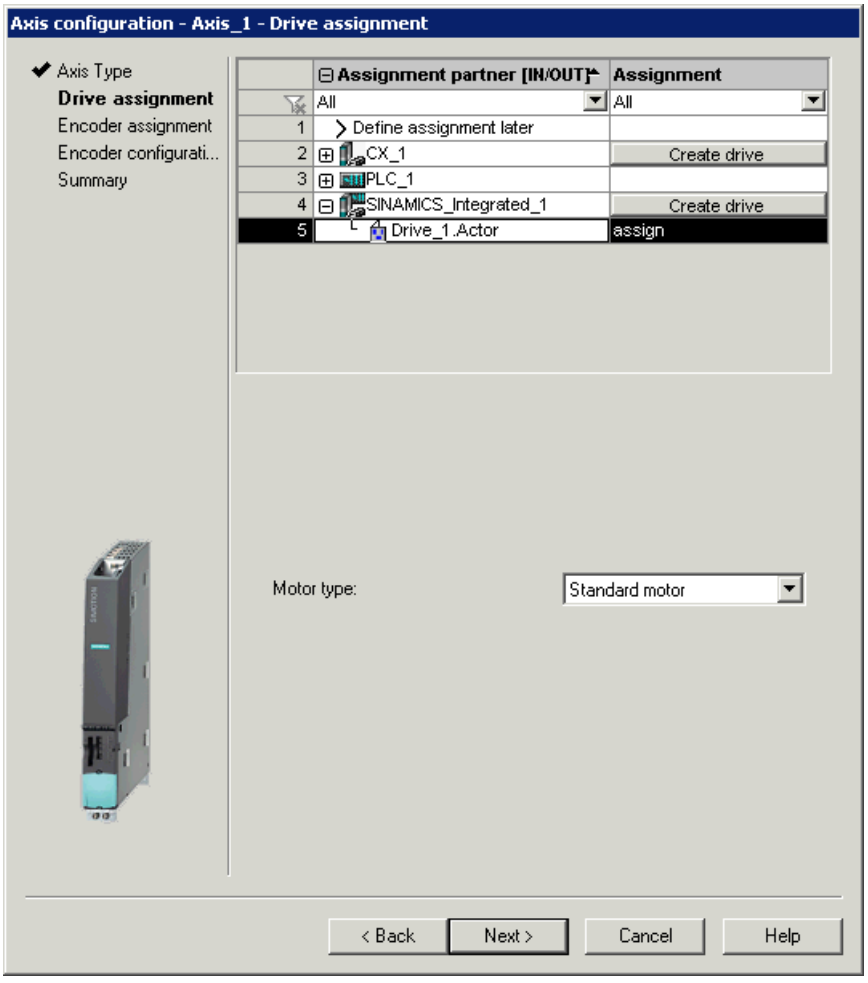


Figure 3-145 Axis configuration - Drive assignment

6. Depending on the selected axis type, with an electric axis you assign the drive which you have configured beforehand in the **Drive assignment** step.
   In the above example, (Axis configuration - Drive assignment figure), several drive units are configured and displayed:
   - SINAMICS Integrated (for the SIMOTION D)
   - DriveUnit_1 (e.g. CU320-2)
   - CX_1 (Controller Extension CX32-2)

   Note

   As an alternative to assigning the axis to an already configured drive, the axis wizard offers two further selection options:
   - Define the axis/drive assignment later:
     The axis is to be created and not assigned to a drive until later. Programming and simulation of the axis are also possible here.
   - Create drive:
     The drive wizard can be called up from the axis wizard (offline configuring). The axis can thus be created in one step along with the drive, and assigned to the drive.

   The alternative approaches are not being considered at this point.

   With an hydraulic axis, you configure the output for a Q-valve in the **Configuration of Q-output** step; the Q-valve represents the actuator for a volume flow.
   Confirm with **Next>**.
7. Specify the encoder to be used by the axis or the active axis data set in the **Encoder assignment** step.
   With an hydraulic axis, you configure the output for a Q-valve in the step **Configuration of Q-output**; the Q-valve represents the actuator for a volume flow. Confirm with **Next>**.
8. Specify the parameters for the used encoder in the **encoder configuration**. Only the encoder type has to be specified when using the symbolic assignment.
9. Once you have worked through all the steps of the configuration of a TO axis, the configured configuration is displayed once again for you to confirm.
   Close the dialog with **Finish**.

Result

The axis is displayed in the **AXES** folder of the project navigator.

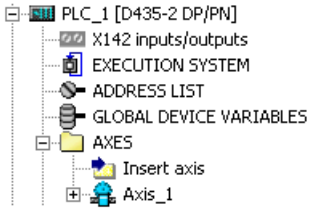


Figure 3-146 Axis created

Automatic settings of the engineering system

The engineering system automatically defines the PROFIdrive axis telegrams required for communication with the drive, as well as the addresses used.

In the same way, telegrams are extended and interconnections automatically created in the drive, depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

Drive and encoder data, as well as reference variables, maximum variables, torque limits, and granularity in torque reduction of the SINAMICS S120 are accepted automatically for the configuration of the SIMOTION technology objects "TO axis" and "TO externalEncoder". This data no longer has to be entered in SIMOTION.

Loading the axis configuration to the target system

Save and compile the changes and download the axis configuration to the target system in order to be able to test the function of the axis in the next configuration step.

Additional references

For detailed information, refer to the "Configuring the axis and external encoder" section in the SIMOTION SCOUT online help.

Testing the axis with the axis control panel

Using the axis control panel you can control and monitor individual axes.

You can perform the following tasks with the axis control panel:

- Test axes before you traverse them via a program.
- Testing as to whether you can move the axis using the axis control panel if a fault is detected.
- Set/remove axis enable.

Requirements

- You have created, configured and connected an axis to the drive.
- SIMOTION SCOUT is in online mode.
- You have loaded the project to the target system.

Opening the axis control panel

To open the axis control panel, proceed as follows:

1. Open the **AXES** folder in the project navigator.
2. Double-click the **Control panel** entry below the axis to be tested.

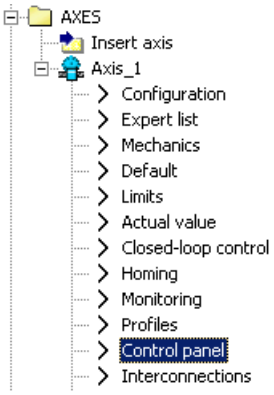


Figure 3-147 Select axis control panel

The axis control panel opens in the detail view.

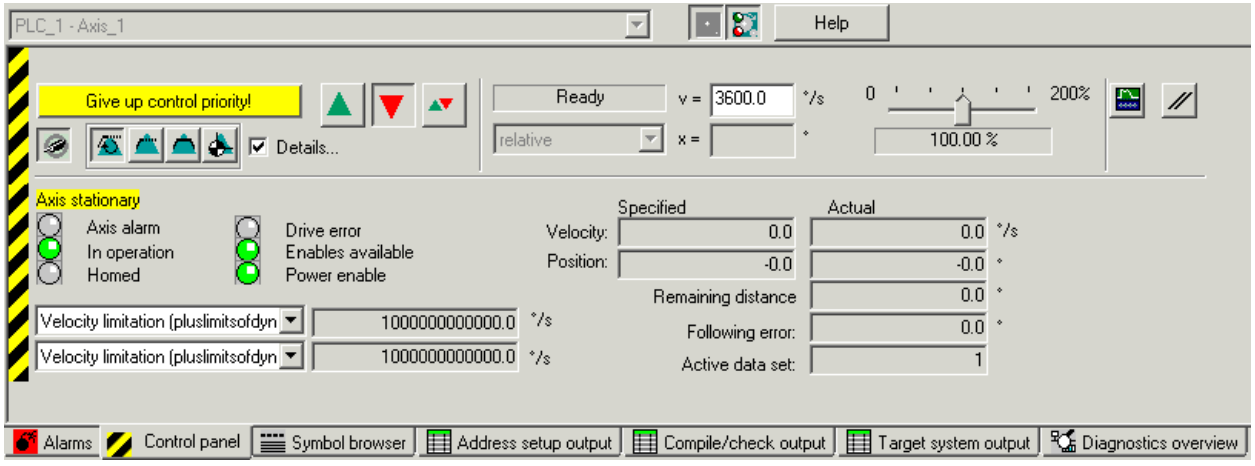


Figure 3-148 Axis control panel

Using the axis control panel

⚠ WARNING

Activating sign-of-life monitoring

Use the axis control panel in control mode only with activated sign-of-life monitoring and a suitably short monitoring time! Otherwise, if problems occur in the communication link between the control PC and the device, the axis may start moving in an uncontrollable manner.

Assume control priority

To assume control priority, proceed as follows:

1. Show the control area by clicking the button in order to test, for example, programmed traversing motions in this mode.
2. Show the diagnostic area by clicking the button in order to test traversing motions that are caused by the motion commands sent to the axis in this mode.
3. Select the axis that you want to traverse and monitor from the selection list, **PLC_1 - Axis_1** in the example.
4. Specify the parameters you want to display in the lower left side of the Diagnostics area using the selection lists. The corresponding values are shown next to them.
5. Select **Assume control priority** to establish the connection to the PG/PC.
   The **Assume control priority** dialog opens.

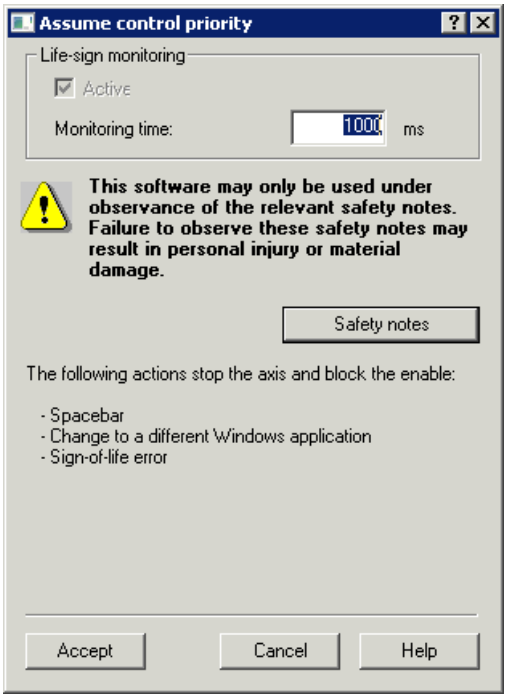


Figure 3-149 Assume control priority

6. Observe the safety regulations and confirm with **Accept**.

The PG/PC has the control priority; the button caption has changed to **Return control priority**.

Note

By pressing the SPACEBAR, you can stop the axis at any time.

Activating enables

To activate the enables, proceed as follows:

1. Click the **Set/remove enables** button to enable the axis.
   The LEDs indicate the status of the enables.

Specifying motion commands for axis

1. Specify motion commands for the axis.
   The following functions are available:

Button	Meaning/function
	Speed default This command specifies that the axis is started speed-controlled. Specify a speed setpoint to which the axis can be ramped up via a velocity profile. The command is possible for all axis types; position and following axes are operated speed-controlled.
	Start axis position-controlled This command specifies that the axis is started position-controlled. Specify a speed setpoint to which the axis can be ramped up via a velocity profile. The command is possible for position and following axes.
	Position axis With this command, you specify positioning of the axis (positioning or synchronous axis) at a specific position. The position value entered can be absolute or relative. Modulo axes can also be positioned via the shortest path. The programmed position must be within the software limit switches.
	Home axis With an absolute measuring system, homing is only required once during commissioning. When this is completed, the position value will be known when the machine is switched on. In the case of an incremental measuring system, the machine must be homed every time it is switched on.

Start axis motion

To start the axis motion, proceed as follows:

1. Click the switch to start the motion command parameterized last.
   Or click the switch to start the **Speed specification** or **Start axis position-controlled** motion commands in jog mode.
   The motion continues as long as the right-hand mouse button is pressed.

Exit axis control panel

To exit the axis control panel, proceed as follows:

1. To stop the system, click the switch.
2. Deactivate the **Set/remove enables** button. Confirm the **Disable axis** dialog with **OK**.

Note

If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

3. Click the **Return control priority** button to return the control priority of the control panel.
4. On the menu bar, select **Project > Disconnect from target system** to switch to offline mode.

Result

The axis can be traversed. Axis configuration is complete.

Potential problems

The axis cannot be traversed.

If the axis cannot be traversed using the axis control panel, test to see if you can traverse the drive using the drive control panel.

- If you cannot traverse the drive using the drive control panel, there is a problem with the drive.
- If you can traverse the drive with the drive control panel, there is a communication problem. Check the drive.

3.3.6.6 Programming the SIMOTION application

Using tags

Variable types

Variables are used to structure programs. They are wild cards in a program and can assume values.

In SIMOTION, different types of variables are distinguished:

- **System variables**
  Each SIMOTION device and technology object has specific system variables. You can access system variables within the SIMOTION device from all programs.
- **I/O variables**
  An I/O variable is a symbolic variable name that is assigned to an I/O address of the SIMOTION device or to the I/O. Thus, direct access to the peripherals is possible.
  I/O variables are valid for all devices. All programs of the SIMOTION device have access to them.
- **Global device variables, unit variables and local variables** are user-defined variables with limited scope:
  These global device variables can be accessed from all parts of the user program. They can also be accessed from HMI devices.
  Unit variables can be accessed by all programs, function blocks and functions defined within the same source, e.g. ST source, MCC source, LAD/FBD source.
  A source is a logic unit that you can create in your project and that can contain programs, functions and function blocks.
  Local variables can only be accessed within the program, the function or function block in which they are defined.
  If variables have been defined in the interface section of a source, they can be accessed by other sources and external components, e.g. HMI systems.

Creating global device variables

Note

Global device variables can only be created in offline mode.

To create global device variables, proceed as follows:

1. Click the **GLOBAL DEVICE VARIABLES** element in the project navigator under the SIMOTION device.
   In the symbol browser, the **Unit-global variables** table is displayed.
2. In the **Name** column, click the first empty cell and enter the variable name, **g_bo_start** in the example.
   Press the **<RETURN>** or **<TAB>** key. The input focus moves to the **Data type** field.
   Alternatively, you can click the field to move the input focus to it.
3. Enter the data type in the **Data type** field, **BOOL** in the example.
4. Press **<RETURN>** to confirm.
   The variable is created and available in the project. In the symbol browser, a new line opens for input.

5. Create other global device variables in the same way, **g_bo_ready** in the example.

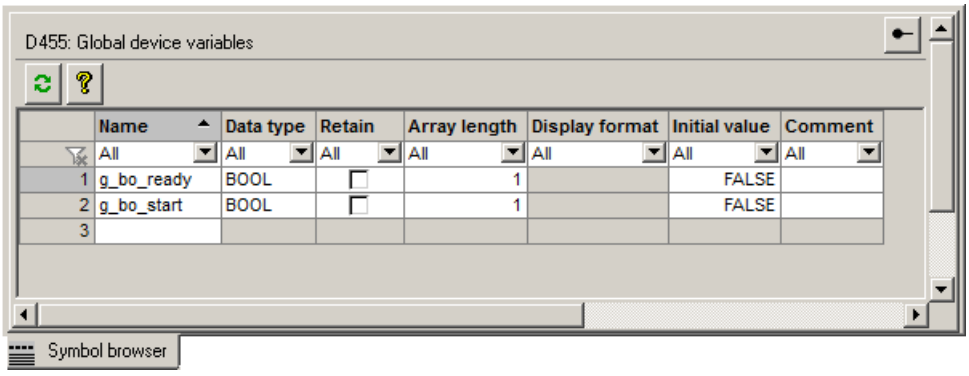


Figure 3-150 Creating global device variables

6. Select **Project > Save and compile changes** in the menu to accept the expansions in the project.

Creating I/O variables

Note

I/O variables can only be created in offline mode.

To create I/O variables, proceed as follows:

1. Double-click the **ADDRESS LIST** element below the SIMOTION device in the project navigator. In the detail view, the **Address list** tab opens.
2. Click the first free cell in the **Name** column. Enter a name for the variable.
3. Press the **<RETURN>** or **<TAB>** key. The cursor moves to the **I/O address** field.
4. Select the **IN** entry for input variable or **OUT** for variable in the **I/O address** column. [1)]
5. Click the three dots in the **Assign** column to open the Assignment dialog.[1)]
6. In the Assignment dialog, select the required variable and select the **Assign** entry in the **Assignment** column.[1)]
7. Confirm the dialog with **OK**.[1)]
8. Select **Project > Save and compile changes** in the menu to accept the expansions in the project.

1) Applies only to devices that support symbolic assignment (e.g. CU320-2).

Additional references

You will find detailed information on the creation of variables in the SIMOTION SCOUT online help and in the relevant Programming and Operating Manuals.

Using MCC

Overview

Program creation steps

Creation of an MCC program encompasses the following steps:

1. Create the MCC source.
2. Create the MCC chart in the MCC source.
3. Insert MCC commands in the MCC chart.

Creating the MCC unit

Procedure

To insert an MCC source, proceed as follows:

1. Open the **PROGRAMS** folder below the SIMOTION device in the project navigator.
2. Double-click **Insert MCC source**.
   The **Insert MCC source** dialog opens.

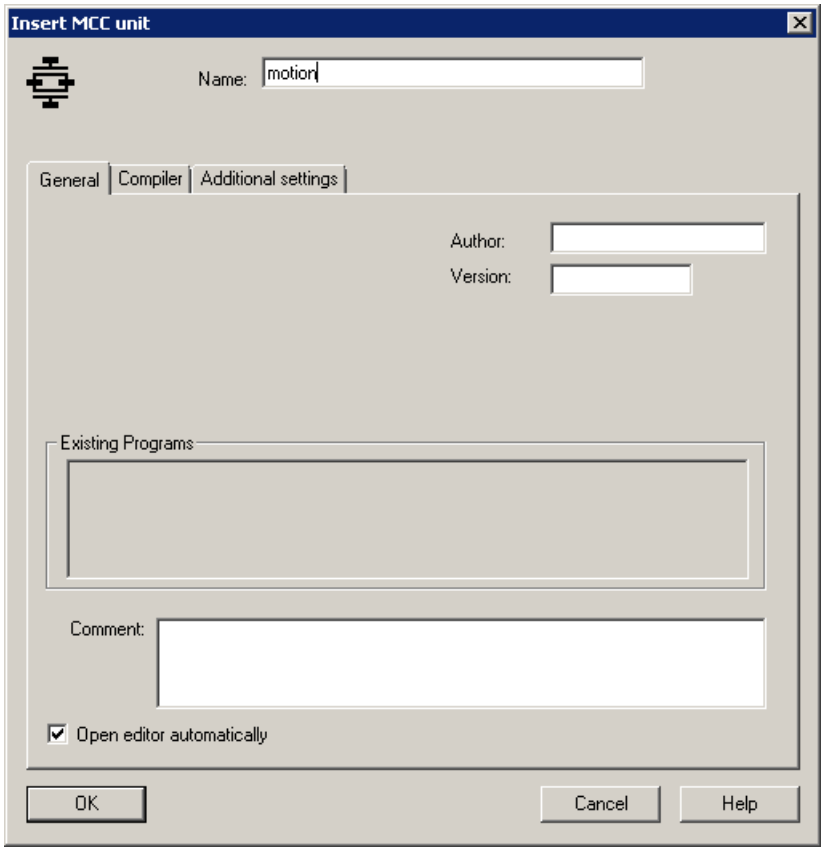


Figure 3-151 Inserting an MCC source

3. Assign a name, **motion** in the example.
4. Go to the **Compiler** tab. For diagnostics purposes, activate the options **Permit program status** and **Permit single step**. In this way, you can monitor program execution later in online mode.
5. Click **OK** to confirm.

Result

The MCC source is created.

- The source appears in the project navigator below the **PROGRAMS** branch.
- The declaration table of the source opens in the working area of the workbench. The variables declared there apply within the MCC source and can be linked in other sources.

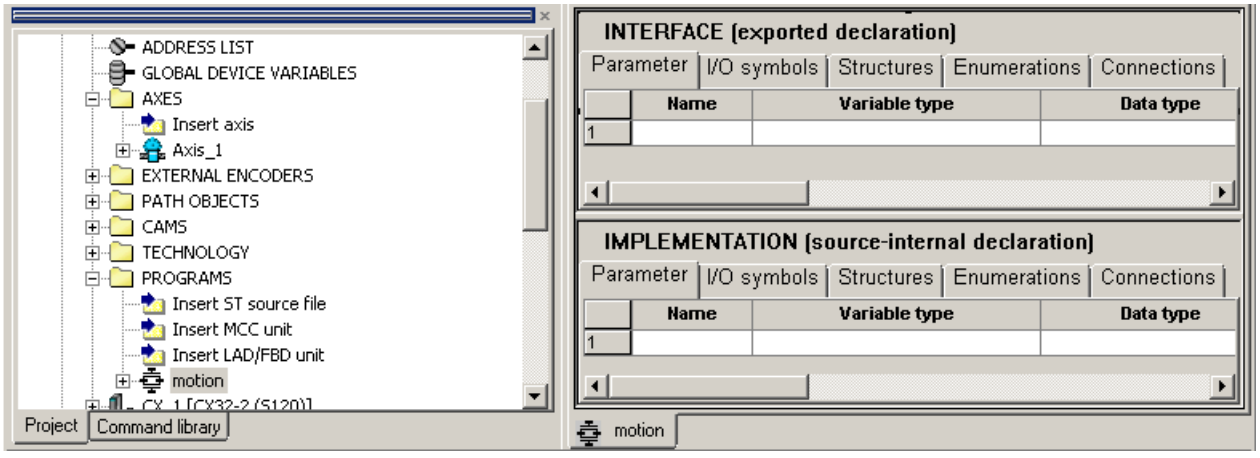


Figure 3-152 MCC source "motion" inserted

Creating an MCC chart

Procedure

To insert an MCC chart, proceed as follows:

1. Open the **PROGRAMS** folder below the SIMOTION device in the project navigator.
2. In the **PROGRAMS** folder, open the MCC source, **motion** in the example.

3. Double-click **Insert MCC chart**.
   The **Insert MCC chart** dialog opens.

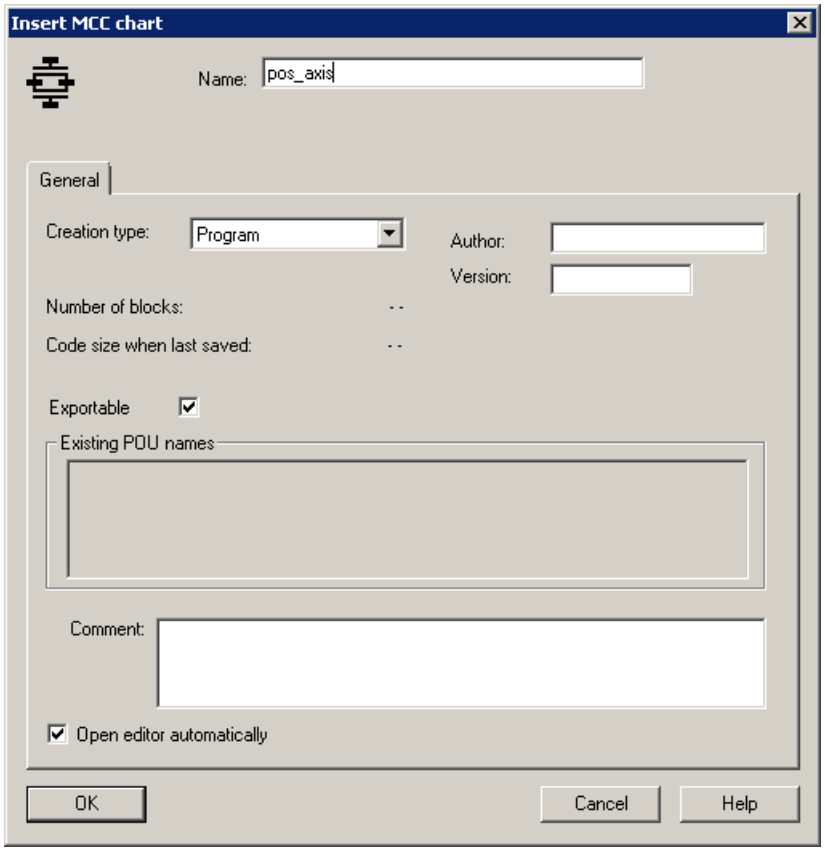


Figure 3-153 Inserting an MCC chart

4. Assign a name, **pos_axis** in the example.
5. Select the creation type **Program**.
6. Click **OK** to confirm.

Result

The MCC chart is created in the project.

- The created MCC chart **pos_axis** appears in the **PROGRAMS** folder below the **motion** source.
- The MCC editor is opened in the working area of the workbench. The start and end nodes are already pre-defined. You can start MCC programming.

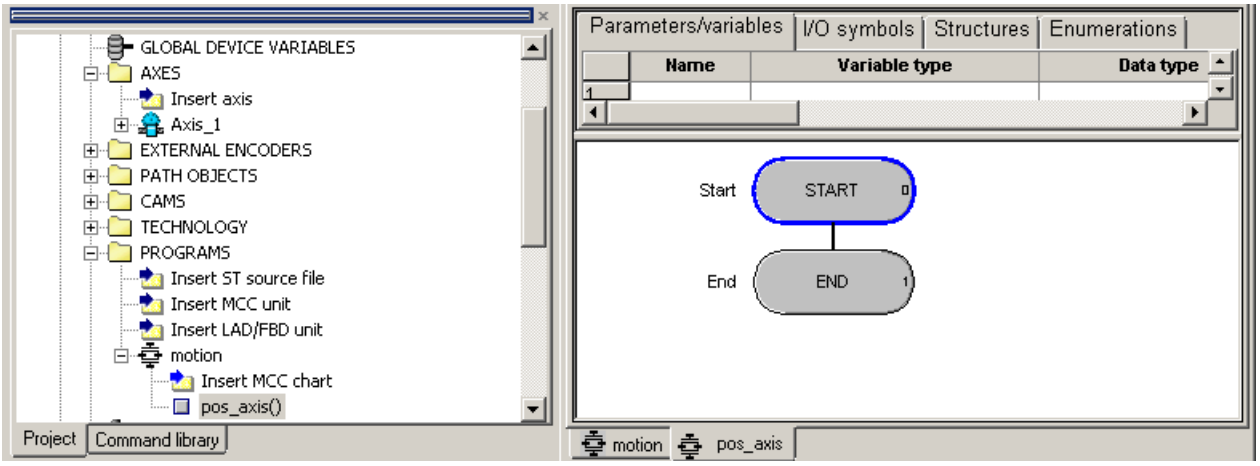


Figure 3-154 MCC chart "pos_axis" inserted

Using MCC command blocks

Every newly created MCC chart already contains a start and an end node.

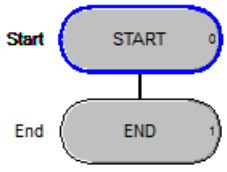


Figure 3-155 MCC chart, start/end nodes

You insert the MCC command blocks between these. The commands are processed in the direction from the start to the end node.

The MCC commands are available to you via:

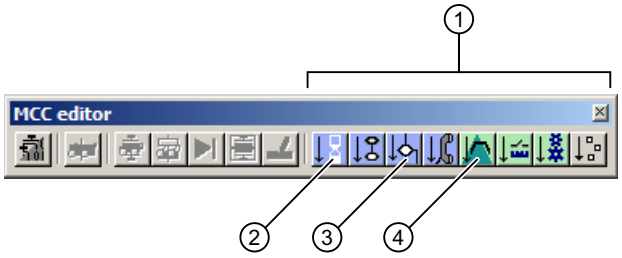
- **MCC editor** toolbar
- **MCC chart > Paste** menu command
- Context menu of the command block

Using the MCC editor toolbar

Opening the toolbar

The **MCC editor** toolbar is displayed in the workbench as soon as you open an MCC chart.

The commands are arranged as command groups.



① Command groups
② Basic commands
③ Program structures
④ Single axis commands

Note

If you do not see the toolbar, check whether the display is switched on: Open the menu **View > Toolbars**. Activate the checkbox for **MCC editor** in the **Toolbars** window.

Opening the command group

Move the cursor across the colored buttons of the toolbar to display the command groups.

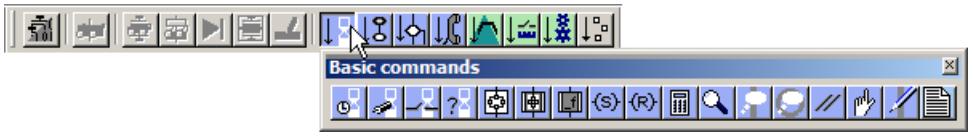


Figure 3-156 Opening the command group

Keeping command groups open or closed continuously

Click the window title of a command group to keep the command group open continuously.

Select "Hide" in the context menu of a command group to close the command group.

Placing the command group as required

Drag the toolbar or the command groups of the toolbar with the mouse to any location on the workbench.

Docking the toolbar

Drag the toolbar or the command groups of the toolbar with the mouse to the edge areas of the workbench to dock them there.

Showing a tooltip for the command

Hold the mouse pointer briefly over a command button. The designation of the command is shown.

Inserting MCC editor commands in the MCC chart

In order to insert an MCC editor command in the created MCC chart, proceed as follows:

1. In the active MCC chart, click the connecting line between two commands, or click the command after which the new command is to be inserted.
   The connecting line or the border line of the command button is marked blue. The marking flashes.
2. Select the command group in the **MCC editor** toolbar.
3. Click the desired command in the command group.

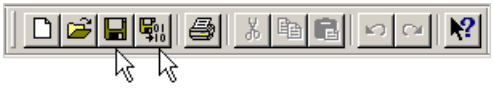
Result

The command was inserted into the chart and can now be parameterized.

Backing up the MCC program

To back up the MCC programs, proceed as follows:

1. To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.



Alternatively, you can click the **Accept and compile** command in the **MCC editor** toolbar.

This command compiles the currently selected program as well as all other programs of the same source.
However, the command does not save the changes.
You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

Additional references

For further information, refer to the SIMOTION MCC Motion Control Chart Programming and Operating Manual.

The "Getting Started section of the SIMOTION SCOUT" online help contains a detailed description of a sample programming in SIMOTION SCOUT.

Using LAD/FBD

Overview

Program creation steps

The creation of an LAD/FBD program encompasses the following steps:

1. Create an LAD/FBD source.
2. Create an LAD/FBD program in the LAD/FBD source.
3. Insert LAD/FBD commands in the LAD/FBD program.
4. Save and compile the LAD/FBD program.

Create LAD/FBD unit

Procedure

To create an LAD/FBD source, proceed as follows:

1. Open the **PROGRAMS** folder below the SIMOTION device in the project navigator.
2. Double-click **Insert LAD/FBD source**.
   The **Insert LAD/FBD source** dialog opens.

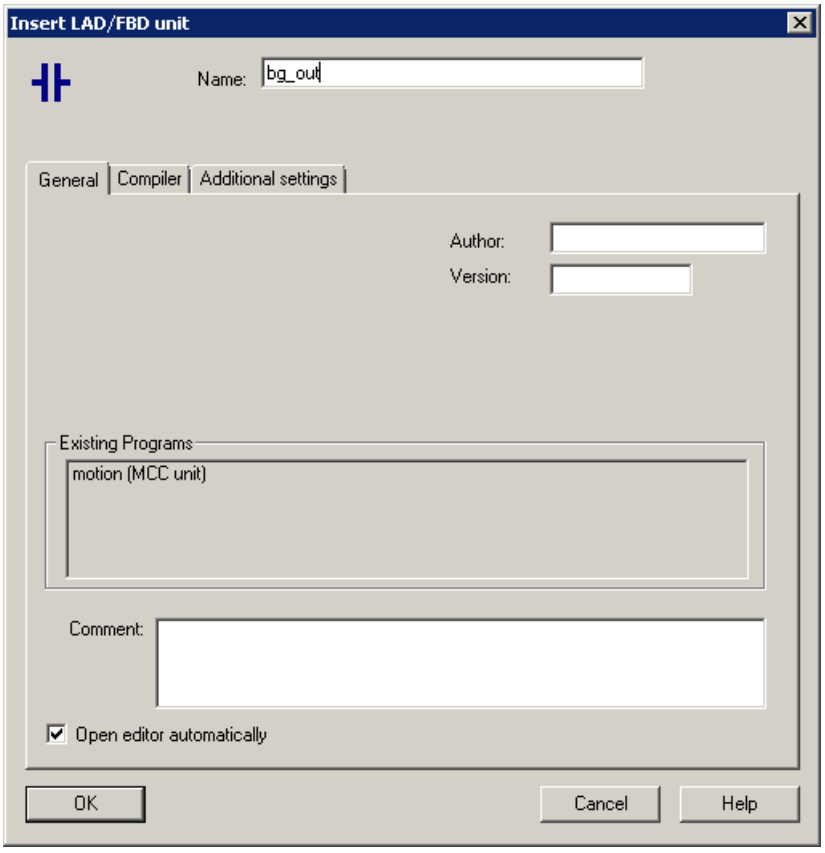


Figure 3-157 Inserting the LAD/FBD source

3. Assign a name, **bg_out** in the example.
4. Switch to the Compiler tab.
5. For diagnostics purposes, activate the option **Permit program status**. In this way, you can monitor program execution later in online mode.
6. Click **OK** to confirm.

Result

The LAD/FBD source is created.

- The LAD/FBD source **bg_out** appears in the **PROGRAMS** folder.
- The declaration table of the source opens in the working area of the workbench. The variables declared there apply within the LAD/FBD source and can be linked in other sources.

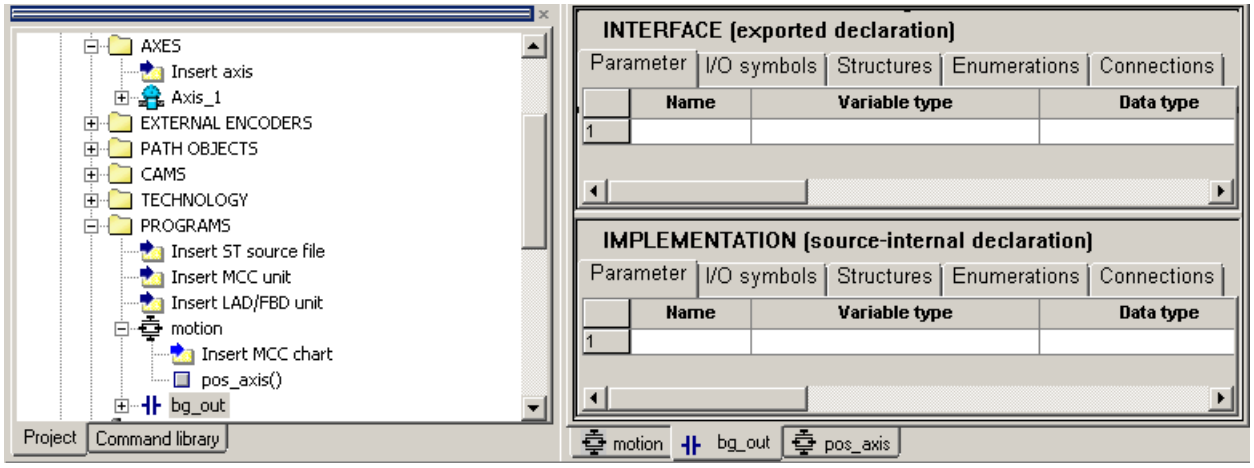


Figure 3-158 LAD/FBD source "bg_out" inserted

Create LAD/FBD program

Procedure

To insert an LAD/FBD program, proceed as follows:

1. Open the **PROGRAMS** folder below the SIMOTION device in the project navigator.
2. Open the LAD/FBD source **bg_out** in the **PROGRAMS** folder.

3. Double-click **Insert LAD/FBD program**.
   The **Insert LAD/FBD program** dialog opens.

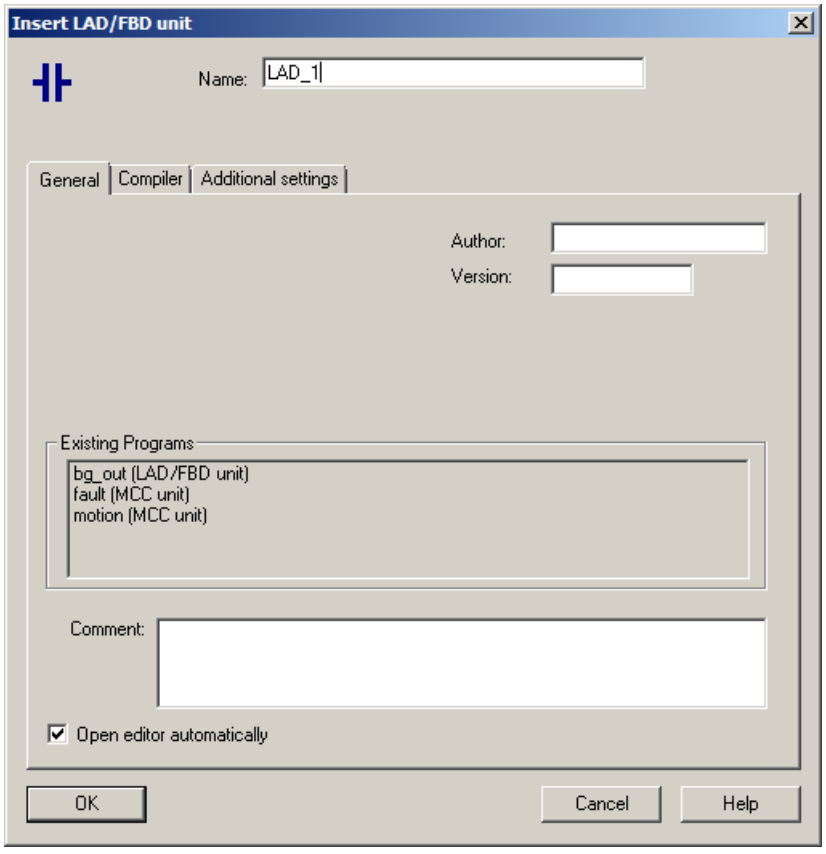


Figure 3-159 Inserting the LAD/FBD program

4. Assign a name, **LAD_1** in the example. The name must be unique throughout the project.
5. Select the creation type **Program**.
6. Click **OK** to confirm.

Result

The LAD/FBD program **LAD_1** is created in the project.

- The LAD/FBD program appears in the **PROGRAMS** folder.
- The LAD/FBD editor is opened in the working area of the workbench. You can start programming.

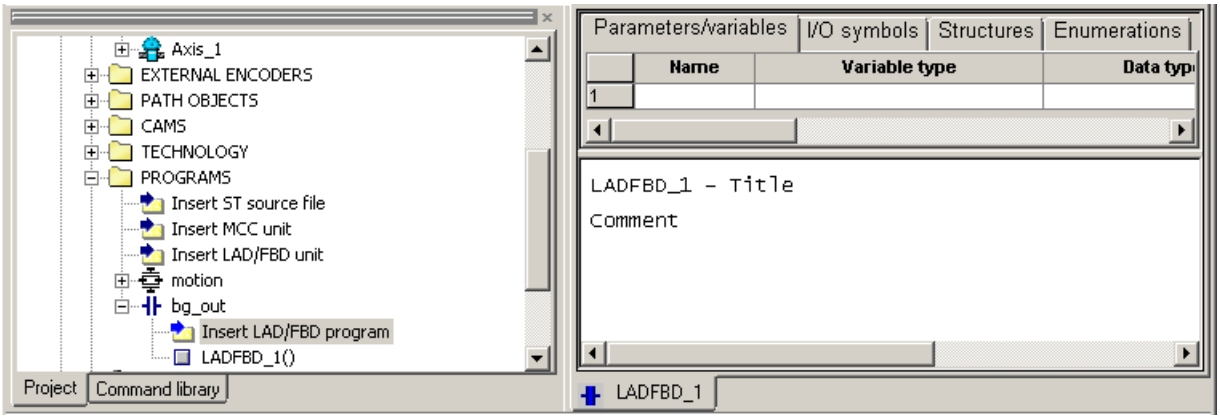


Figure 3-160 LAD/FBD program "LAD_1" inserted

Using the LAD/FBD toolbar

Opening the LAD/FBD toolbar

The LAD/FBD toolbar becomes visible in the workbench as soon as you open an LAD/FBD program.

Move the cursor across the active buttons of the toolbar to display the functions.

Note

If you do not see the toolbar, check whether the display is switched on: Open the menu **View > Toolbars**. In the **Toolbars** window, activate the checkbox for the **LAD/FBD toolbar**.

Switching the programming language

SIMOTION SCOUT allows simple switching between ladder logic and function block diagram. The LAD/FBD editor contains the command **LAD/FBD program > Switch to FBD** or **Switch to LAD**.

Backing up the LAD/FBD program

Procedure

In order to back up an LAD/FBD program, proceed as follows:

1. Click in the toolbar on the **Save project** or **Save project and compile changes** button.

Alternatively, you can click the **Accept and compile** command in the **LAD/FBD toolbar**.

This command compiles the currently selected program as well as all other programs of the same source.
However, the command does not save the changes.
You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

Additional references

For further information, refer to the Online Help and SIMOTION LAD/FBD Programming and Operating Manual.

The **Getting Started section of the SIMOTION SCOUT** online help contains a detailed description of a sample programming in SIMOTION SCOUT.

Using ST

Overview

Program creation steps

The creation of an ST program encompasses the following steps:

1. Create an ST source.
2. Save and compile the ST program.
3. Execute the ST program.

Creating an ST source

Procedure

1. In the project navigator, open the tree for your SIMOTION device (programs are assigned to the SIMOTION device on which they are to run).
2. Mark the Programs folder and select the **Insert > Program > ST source** menu command. The **Insert ST source** dialog opens.

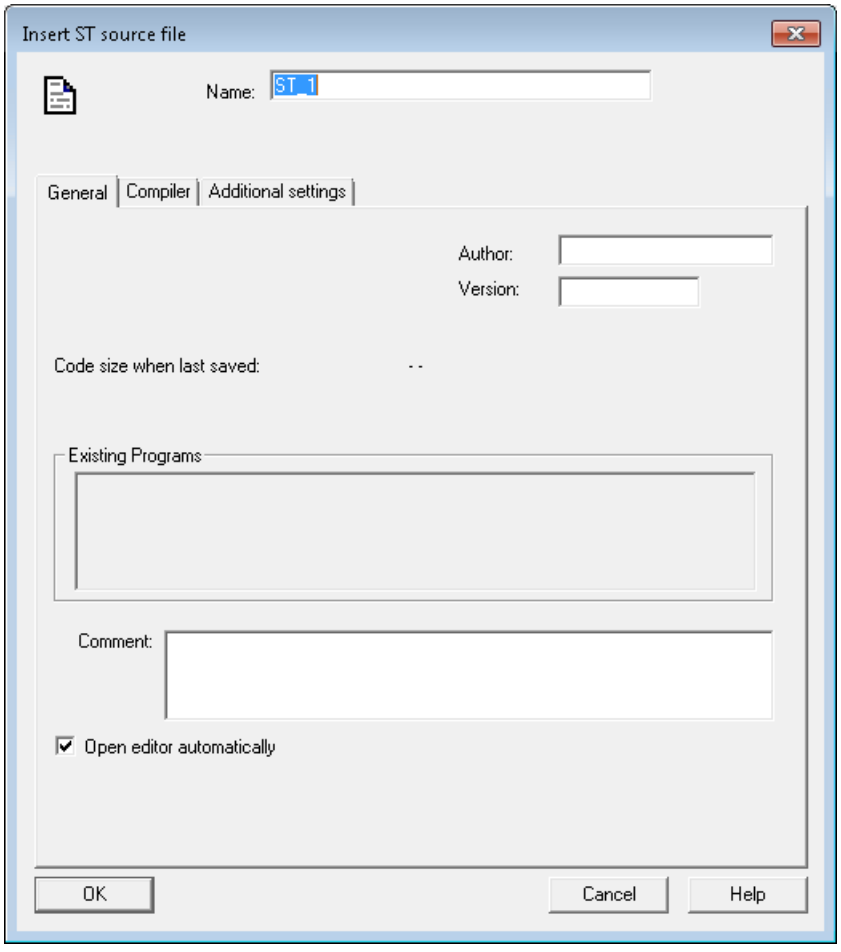


Figure 3-161 Inserting an ST source

3. Enter a name for the ST source consisting of up to 128 characters (see figure), e.g. ST_1, and click **OK** to confirm the entries.
   The ST editor appears in the working area. The ST source ST_1 is inserted in the navigator.

Result

The ST_1 ST program is created in the project.

- The ST program appears in the PROGRAMS folder.
- The ST editor opens in the working area of the workbench. You can start programming.

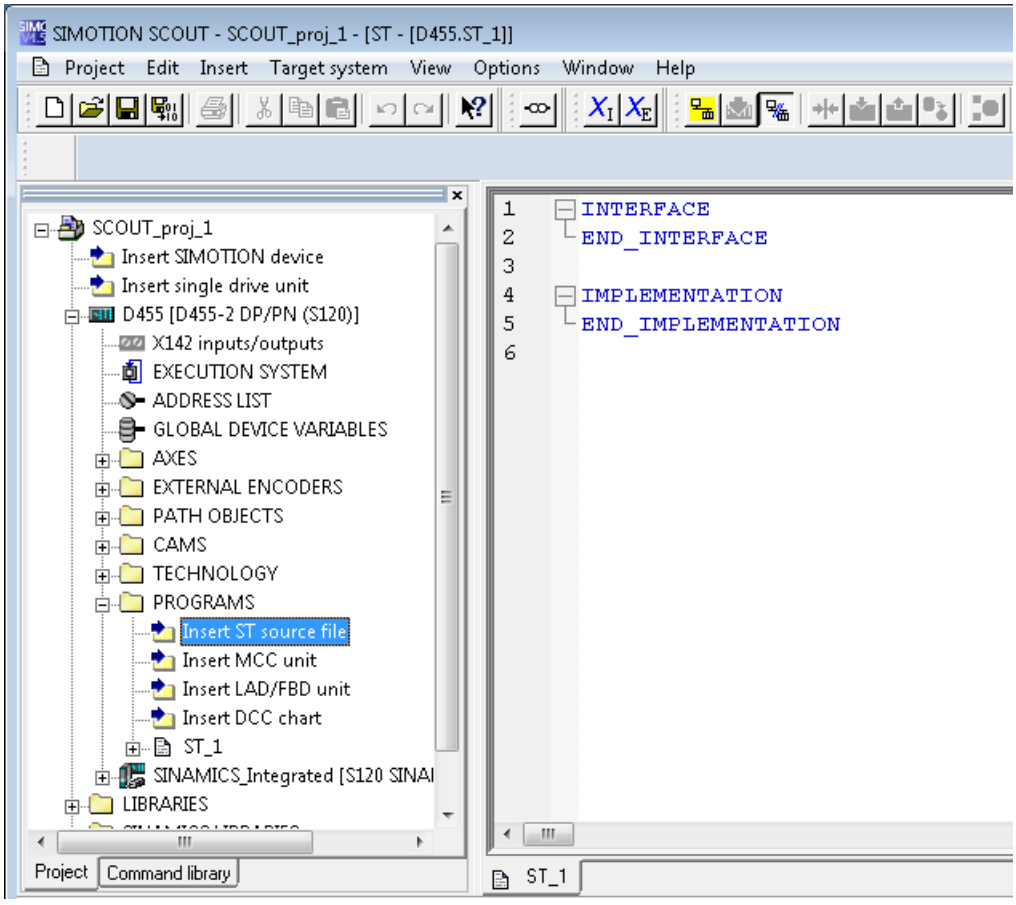


Figure 3-162 ST source "ST_1" inserted

Backing up the ST program

Procedure

To back up an ST program, proceed as follows:

1. Save the complete project with **Project > Save**.

Before you can run or test your program, you must compile it into executable machine code. The ST compiler performs this task.

Start the compiler as follows:

1. Click in the window with the ST editor to display the **ST source** menu. This menu is a dynamic menu and is only displayed if the window of the ST editor is active.
2. Start the compiler by selecting the **ST source > Accept and compile** menu command.

Additional references

Further information on ST programming can be found in the SIMOTION SCOUT online help and in the SIMOTION ST Structured Text Programming and Operating Manual.

Executing the ST program.

Before you can execute the sample program, you must assign it to an execution level or a task. When you have done this, you can establish the connection to the target system, download the program to the target system, and then start it.

Additional references

Further information on ST programming can be found in the SIMOTION SCOUT online help and in the SIMOTION ST Structured Text Programming and Operating Manual.

3.3.6.7 Configure execution system

Execution levels define the chronological sequence of tasks in the execution system. A level can contain several tasks. The tasks provide the framework for program execution. A task may comprise several programs. By assigning the created programs to the tasks, you can, for example, define the priority, the time frame or order in which the programs are to be executed.

Assigning programs to tasks

To assign programs to tasks in the execution system, proceed as follows:

1. Double-click in the project navigator below the SIMOTION device on **EXECUTION SYSTEM**.
   The **Execution System** window opens in the working area.
2. Assign the required program to the preferred task. In the sample MCC program, **motion.pos_axis** and **MotionTask_1** task.
   – In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > MotionTasks > MotionTask_1**.

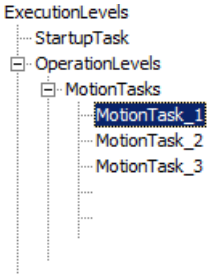


Figure 3-163 Execution system, tree view of the execution levels and tasks

The **MotionTasks** window opens on the right of the working area. The programs **pos_axis** and **LAD_1** as well as programs of the **fault** source are visible on the **Program assignment** tab at **Programs**.

– Select the MCC program **motion.pos_axis** and click the **>>** button. The program is displayed at **Programs used**. It is thus assigned to the task **MotionTask_1**.

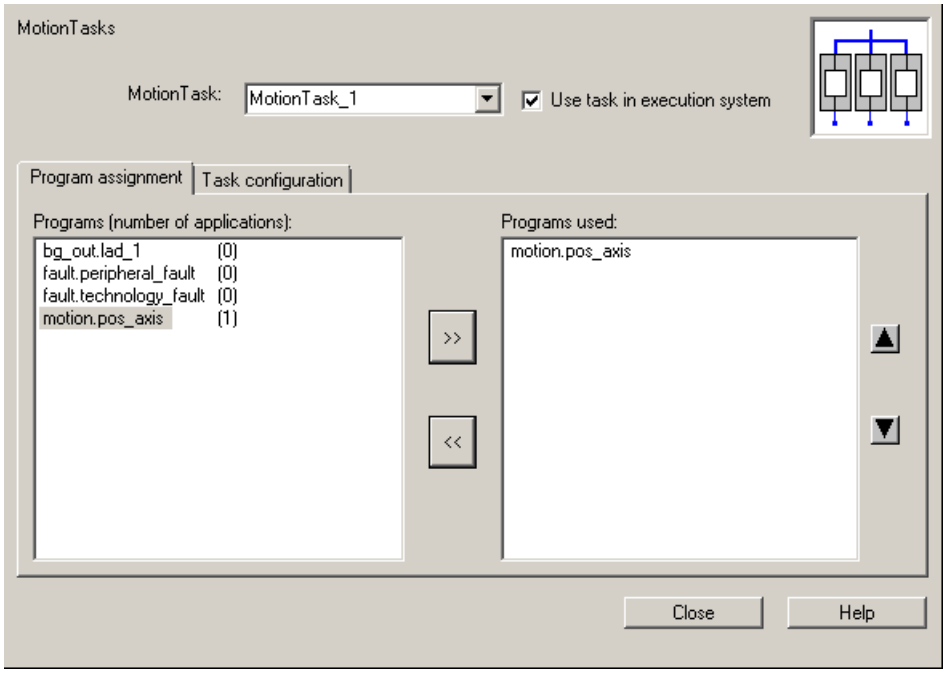


Figure 3-164 Execution system, MotionTasks window

The assignment is visible in the tree of the execution system. The program **motion.pos_axis** appears below the branch **MotionTask_1**.

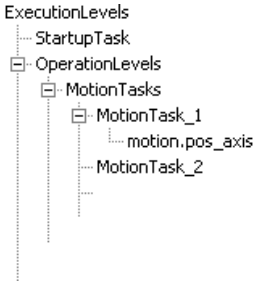


Figure 3-165 Execution system, MotionTask_1 with assigned pos_axis program

– In the **Task configuration** tab, activate the **Activation after StartupTask** option. This executes the MCC program immediately after the SIMOTION device is started. If this option is not activated, the program must be started explicitly by the call from another program that is assigned to the StartupTask or another active task.

3. Assign the LAD program **bg_out.kop_1** to the task **BackgroundTask**:

– In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > BackgroundTask**. Assign the LAD program **bg_out.kop_1** to this task.

4. Assign the fault handling routines:
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > TechnologicalFaultTask**. Assign the MCC program **fault.technology_fault** to this task.
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > PeripheralFaultTask**. Assign the MCC program **fault.peripheral_fault** to this task.
5. Click **Close**. Confirm with **Yes** if you are prompted to save.

Result

The execution system is configured.

Loading the configured execution system to the target system

Save and compile the changes, go online and load the configured execution system into the target system.

3.3.6.8 Project generator

Overview

With the project generator, you can quickly and easily create a new SIMOTION project or update an existing project on the basis of templates.

The templates include preconfigured SIMOTION devices and modules, all of which are supplied with the ProjectGenerator. These can be automatically integrated in a project with the ProjectGenerator.

Overview of the ProjectGenerator functions

You can execute the following functions with the ProjectGenerator:

- Create a new project or open and edit an existing project
- Create a new device or open and edit an existing device
- Automatic configuration of the modules on the basis of the selected hardware
- Copy files/directories in the file system
- XML interface for user-specific expansions
  - Generation of the user-interface elements
  - Generic function calls
- Assign programs to the execution system
- FTP transport of files to the SIMOTION device (e.g. SIMOTION IT pages)
- Import technology objects (TO)
- Interconnect technology objects (synchronous operation interconnection)

- Write configuration data and system variables to technology objects
- Import libraries:
  - Change code
  - Version check and version update
  - Setting and restoring constants
  - Creating and restoring user areas
- Import units
  - Change code
  - Version check and version update
  - Set and restore constants
  - Create and restore user areas

Integrating the ProjectGenerator

Integration in SCOUT

As of version V1.3, the ProjectGenerator can be integrated in SCOUT. The ProjectGenerator does not require an installation. You can integrate the ProjectGenerator via the menu **Options > Settings...**, **ProjectGenerator** tab directly from a DVD or the local hard disk. Select the path to the **ProjectGenerator.exe** file.

You then use the ProjectGenerator via the menu item **Project > ProjectGenerator**.

Note

The path must always be valid. If the DVD is removed from the drive or the local folder is deleted, an error message is issued.

Additional references

You can find additional information on this topic in the SIMATIC/SIMOTION Project Generator Application Manual.

3.3.6.9 Configuring multilingual messages

Multilingual messages and comments can be configured with the "Language-dependent texts" function. This is necessary, for example, if you want to output multilingual user-defined Alarm_S messages or diagnostic buffer entries.

To configure multilingual messages:

1. Select **Project > Language-dependent texts** in the menu. The **Select Language** window opens.
2. Set the language in which you want to configure the messages as the **current language**.
3. Click **OK**. The window closes.
4. Select **Project -> Messages -> Configure** in the menu and check the set language at **Configuration language** in the displayed window.
   The message texts and remarks are displayed in the language in which they were originally configured. These texts must then be compiled in the new configuration language.
5. In the **Message Configuration** window, double-click the messages for which you want to enter texts in the new configuration language or click the **Edit** button. The **Edit Message** window is displayed.
6. Enter the new message text and remarks in the changed language.
   The symbol name must not be changed in foreign language messages, as this symbol name is used for programming. Only the message text and remarks are language-dependent.
7. Click **OK** to confirm. The modified message text is entered in the Message configuration table.
8. Repeat steps 6 to 8 if you want to enter more message texts in the new configuration language.

For further information, please refer to the online help for SIMOTION SCOUT -> Configure multilingual texts/comments.

3.3.6.10 Know-how Protection

The know-how protection in SIMOTION SCOUT provides read- and write protection for program sources (ST sources, MCC charts, LAD/FBD programs and DCC charts) and libraries in your project. This protection prevents unauthorized viewing of the programs. You can specify different protection levels for programs and libraries. The specification applies to all protected programs and libraries.

If the know-how protection is set for a program, for example, the know-how protection is retained, even when this program is copied or exported.

Specifying the protection level

To specify the the know-how protection level, proceed as follows:

1. Select **Project > Programs know-how protection > Configure ...** in the menu bar.
2. Specify the protection level for programs and libraries.
3. Confirm your selection with **OK**.

Creating the standard login

To create the standard login for the know-how protection, proceed as follows:

1. Select **Project > Programs know-how protection > Edit standard login...** in the menu bar.
2. Enter the login and password, and confirm the entered password.
3. Confirm the input with **OK**.

Set know-how protection

To set the know-how protection, proceed as follows:

1. In the project navigator, mark the source for which you want to set the know-how protection.
2. Open the dialog for setting the know-how protection from the **Know-how protection > Set** context menu.
3. Enter the login and password, and confirm the entered password.
   Provided no standard login has been set and this login should be used as standard login, activate the **Use this login as standard login** checkbox.
4. Confirm the input with **OK**.

The lock symbol at the associated source identifies protected programs in the project navigator, e.g.  .

Additional references

For additional information on this topic, refer to:

- SIMOTION MCC Motion Control Chart Programming and Operating Manual
- SIMOTION LAD/FBD Programming and Operating Manual
- SIMOTION ST Structured Text Programming and Operating Manual
- DCC Programming and Operating Manual
- SIMOTION SCOUT Online Help

3.3.6.11 Saving and restoring variables from the device

Saving and restoring data with SCOUT

Using the SIMOTION SCOUT functions **Save variables** and **Restore variables** it is possible to save and then restore data (which has been changed during operation and is only saved in the SIMOTION device or on the memory card) to the hard disk. This can be used if a SIMOTION platform is changed or a version updated, for example.

The following types of data can be backed up:

- Data variables that are in the SRAM/NVRAM of the controller:
  - Retentive global device variables
  - Retentive global unit variables
  - TO retain data (as of V4.1)
- Data sets which have been backed up from the user program via _saveUnitDataSet or _exportUnitDataSet and are located on the memory card, including
  - Global unit variables of interface and implementation sections from program sources (ST, MCC or LAD/FBD sources):
    Retentive variables (VAR_GLOBAL RETAIN)
    Non-retentive variables (VAR_GLOBAL)

The **_exportUnitDataSet** function backs up unit variables in a **version-independent** format (XML).

By contrast, data is backed up in a **version-dependent** format (binary) via the **_saveUnitDataSet** function and can, therefore, only be read back in on a device/unit of the same version (e.g. V3.2) via the **_loadUnitDataSet** function.

Data sets saved via _saveUnitDataSet are automatically converted into XML format using the **Save variables** function. The data sets are available in binary format for the new version once the **Restore variables** function has been executed.

Data backed up using **Save variables** are saved as version-independent XML files in a user-defined folder on the PC's hard disk.

In addition to the data backup functions provided by SIMOTION SCOUT, backup functions are also available in the runtime system.

Note

When performing an update with a kernel version that is ≥ V4.1, the above SIMOTION SCOUT functions are only required for backing up and restoring unit data sets that have been created using _saveUnitDataSet.

Reason:

- Retain data remains valid after an update.
- Unit data backed up using _exportUnitDataSet also remains valid after an update.
- Retain data can also be backed up to a memory card **without the use of SIMOTION SCOUT:**
  - The _savePersistentMemoryData function saves **all** retain data from the SRAM/NVRAM; this can be retentive global unit variables in the interface or implementation sections of a source (VAR_GLOBAL_RETAIN) or retentive global device variables (RETAIN) and TO retain data
    (For further information, see List Manual: Device System Functions/Variables)
  - The **Back up diagnostic data** function (triggered via a service switch/diagnostic button or IT DIAG) saves **all** retain data (as above, see _savePersistentMemoryData). Subsequently executing the **Restore non-volatile data** function reactivates this backed up data.
    (For further information, see the SIMOTION D Commissioning and Installation Manuals)

A directory structure is created in the selected path using the **Save variables** function. This directory is assigned the name of the selected SIMOTION device or the selected unit, depending on whether the variables should be saved or restored for the entire device or just one unit.

The retentive global device variables are backed up in the unitdata.xml file. The number of subdirectories corresponds to the number of sources in the SIMOTION project. The names of the subdirectories correspond to the names of the sources. Each subdirectory contains the unitdata.xml file in which the retentive unit variables are stored. If necessary, a further file that contains the non-retentive global variables of the source is saved. This DS*.xml file is assigned the data set number as its name, which is transferred by specifying the parameter **ID** for the **_saveUnitDataSet** or **_exportUnitDataSet** function.

Example of a folder structure when backing up on the device:

The folder structure is as follows:

<Folder device>, e.g. D435
unitdataset.xml
DS000001.xml
<Folder device>, e.g. ST_UNIT1
Unitdataset.xml
DS000002.xml

Restore variables

If the **Restore** function is called in the context menu of a unit, select the directory with the name of this unit in the directory selection dialog.

If the **Restore** function is called in the context menu of a SIMOTION device, select the directory with the name of this device in the directory selection dialog.

Example: If the variables of the SIMOTION device have been stored in the directory D435, then this is the directory that must be selected for the restore operation.

Additional references

Detailed information on this topic can be found in:

- SIMOTION Runtime Basic Functions Function Manual, Section Data backup from user programrogram

3.3.6.12 Online multiuser mode

Overview

Thanks to the new Project comparison, Device upload and Download with supplementary data functions, several users can exchange project data that has been modified via the target device, online. This allows them to synchronize the status of their own SIMOTION project with that of another one, thereby updating it.

In online multiuser mode, it is possible for two users to work online on the same SIMOTION device. The following functions can be performed:

- Read and control system variables and configuration data
- Execute measuring functions and axis control panel
- Upload configuration data
- Upload programs or technology objects and other settings (e.g. execution system)
- Edit and download programs (following prior alignment, if applicable)
- Download individual program sources (possible even when other sources are inconsistent)

The following graphic shows a schematic representation of online multiuser mode:

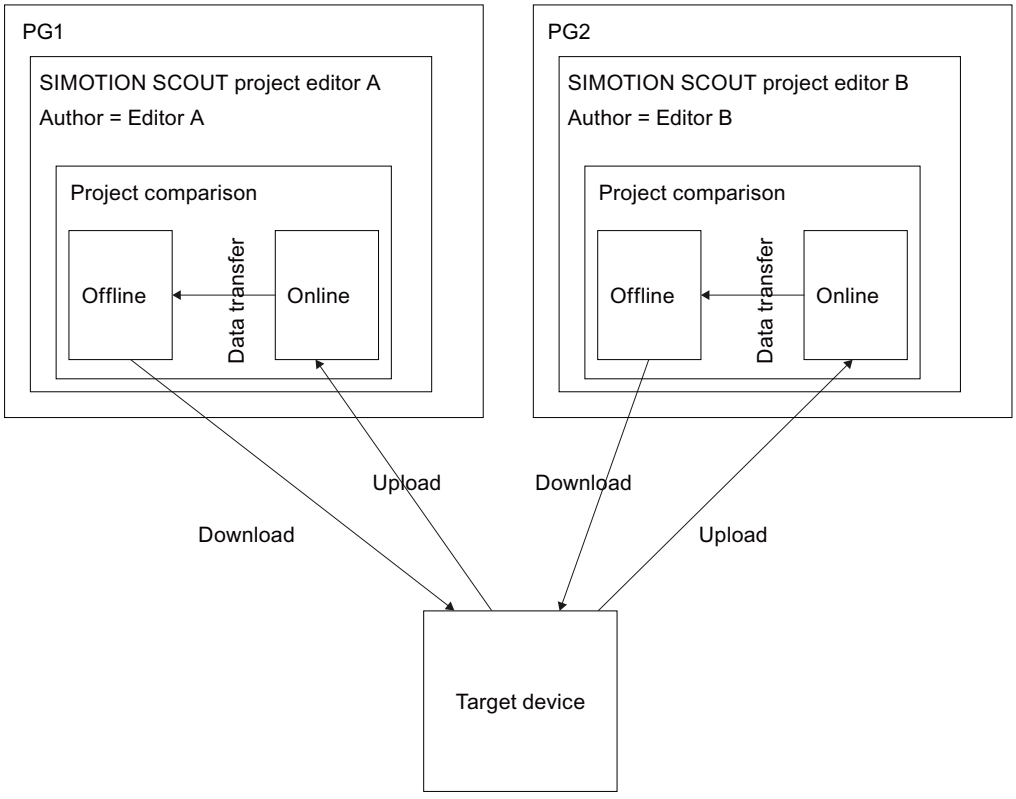


Figure 3-166 Schematic representation of online multiuser mode

Editors A and **B** are working on one SIMOTION project at the same time. Changes made by both editors are loaded to the target device. The project must be synchronized beforehand, otherwise parts of it could be found to be inconsistent and subsequently overwritten when the **Load project to target device** command is executed.

This means, for example, that **Editor A** must synchronize his project with **Editor B**'s project before loading it to the target device. This can be done via the synchronization mechanism in the project comparison facility.

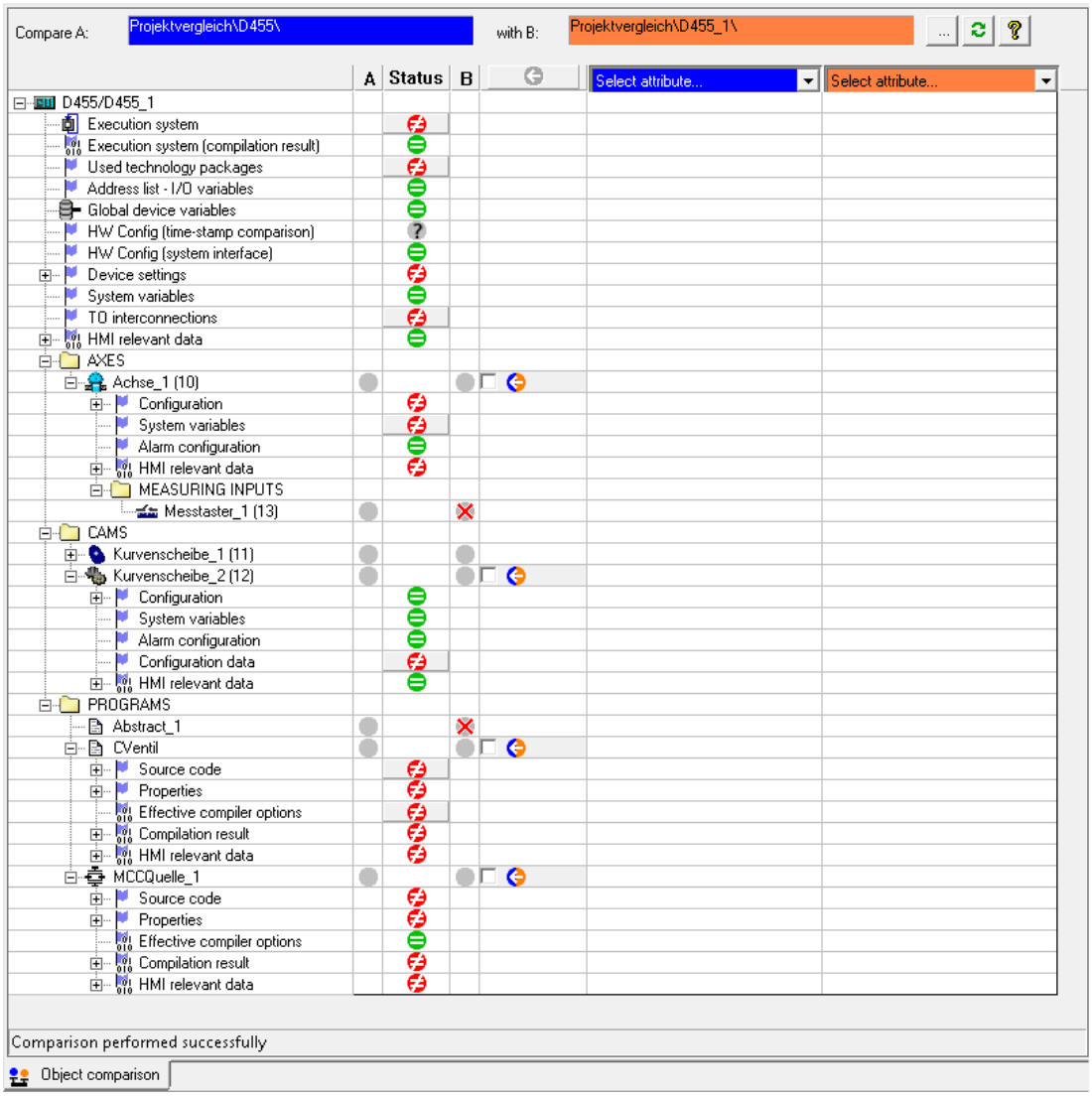


Figure 3-167 Synchronizing project data via the project comparison function

Working in online multiuser mode

Online consistency

As of V4.4, the online consistency flags are updated event-driven. As soon as a change takes place, SCOUT/STARTER is informed and the flags are updated. You can then no longer debug changed sources. The following requirements apply:

- SIMOTION controllers, as of V4.2
- SINAMICS drives, as of V2.6
- Only structural changes (genuine inconsistency) are taken into account in the case of SINAMICS drives.

Online multiuser mode

When several users work simultaneously, this can result in the following overlaps, for example:

- A user creates a program that is not known on the second computer.
- A user changes the execution system. The change is not known on the second computer.

To accept the changes of other users in your own project, various additional functions have been implemented in SIMOTION SCOUT:

- Download a number of programs: This makes it possible to load only those programs to a controller that are currently being edited, e.g. by a specific user.
- You can see changes to the execution system in the project comparison. You can then upload the changes via the detailed comparison.

Upload the source of another user

If another user has loaded a source to the controller that is part of your project, you cannot load this to the local SCOUT via the project comparison. To upload the source, proceed as follows:

1. Create an empty source with the same name in your project.
2. Perform an online/offline project comparison. An inconsistency is displayed in the comparison for the empty source.
3. Then change to the detailed comparison. The inconsistent source is displayed there.
4. Load the source to your project. By doing this, you make sure that you have a consistent project.

Avoid inconsistencies by loading via HW Config

The matching of the data by uploading to PG is supported only for SCOUT data. Furthermore, changes can be made to the hardware configuration, e.g. changes to addresses of the X142 interface. These changes must be replicated explicitly in the hardware configuration.

3.3.6.13 Licensing runtime

Licensing of the runtime components

Overview for the licensing

Functions can be licensed using the following software options:

- Motion control technology functions
  The licensing is performed axis-specifically for:
  - POS - position; use of the technology functions for position axis
  - GEAR; use of the technology functions for following axis
  - CAM; use of the technology functions for cam axis

The GEAR technology function contains the POS technology function, while the CAM technology function contains the POS and GEAR technology functions.

The MultiAxes package permits a simple licensing of the Motion control technology functions. It contains the license for the unlimited use of the CAM technology function on a SIMOTION device, e.g. a C240, a D4x5-2 or a P320-4.

- TControl technology function
  The use of the TControl technology package functions is licensed on a channel-specific basis in packages of eight temperature channels.
- IT functions, IT DIAG, and SIMOTION IT
  The use of these functions is licensed for each SIMOTION device.

Note

Another option is to order SIMOTION memory cards (MMC and CF) and SIMOTION P320-4 with pre-installed runtime licenses.

Additional references

Further important information for the licensing of the runtime software and the ordering data can be found in:

- Catalog: SIMOTION, SINAMICS S120 and Motors for Production Machines
- PM 21 catalog, section titled SIMOTION runtime software
- Configurator for SIMOTION Runtime Licenses in the Industry Mall (http://mall.automation.siemens.com)

Licenses and license key

Depending on the type and number of runtime components used in the project, licenses must be acquired as part of the licensing procedure for SIMOTION. The licenses required for a SIMOTION device are assigned to a license key. This license key is stored on the storage medium of the SIMOTION device during the licensing procedure.

There are two ways of ordering the licenses:

- Preinstalled licenses
  The license key is already stored on the card.
- Ordered licenses (Certificate of License)
  These licenses must be assigned to the storage medium using the Web-License-Manager. The determined license key is transferred to the hardware using SCOUT.

You require the following information to obtain the license key:

- The serial number of the SIMOTION device storage medium
  You can obtain the serial number from the storage medium or have it displayed online in the SIMOTION SCOUT (licensing wizard).
- The serial number of the CoL (Certificate of License)
  You have this number on paper.

Table 3-29 The serial number on the SIMOTION hardware assigned to the SIMOTION device

SIMOTION device	Hardware serial number of the module
SIMOTION C2xx	SIMOTION Micro Memory Card
SIMOTION P	The serial number is located on the rating plate of the SIMOTION P, starting with SVP...
SIMOTION D	SIMOTION CompactFlash Card

License keys can be generated separately from the licensing.

Note

When the SIMOTION Memory Card is deleted or formatted, the licensing data is also deleted. Archive the licensing data in order to be able to transfer it again to the storage medium in such a case. If the data is not backed up, you have to perform the licensing again. You can display the entered license key in the Web License Manager.

You will find additional information in the section:
License key is protected from deletion (as of kernel V4.1) (Page 366)

See also

FAQ - Dealing with licenses (http://support.automation.siemens.com/WW/view/en/36947932)

Determining licensing requirements

Note

Determine your license requirement only after you have completed the configuration! A license that has been assigned a license key cannot be withdrawn.

Once you have completed your project configuration with SIMOTION SCOUT and before you download it to the target device, you can determine the licenses required for the project. You have three options for determining license requirements. Before you begin, the project must have been saved and compiled. If you have not yet acquired any licenses, underlicensing is displayed.

Options for determining license requirements are as follows:

- **Offline mode** with open project
  The required licenses are displayed.
- **Online mode** with open project
  A comparison of the required and actual licenses is displayed.
- **Online mode** without project
  The actual licenses of the selected SIMOTION device are displayed.

Proceed as follows:

1. Select the **SIMOTION device** in the project navigator.
2. Select **Licenses** in the context menu.
   The required licenses for the project or a comparison of required and actual licenses are displayed.
3. You can close the window with **X** or continue with **Perform licensing...**.

The license check, i.e. the inspection of the license keys, is carried out in the target system. Possible responses in the case of underlicensing are described in the Underlicensing section.

Note

Memory cards can be purchased with integrated runtime licenses, which do not require separate licensing.

Displaying existing licenses of the SIMOTION device

Displaying via accessible nodes

You can use the list of **Accessible nodes** to determine the specific licenses that have already been assigned to the SIMOTION device. You can access the data of the SIMOTION device directly.

Note

This step is not necessary if the required and actual licenses are displayed within a project.

Requirements:

- SIMOTION SCOUT is running
- SIMOTION SCOUT is in online mode
- **No** projects are open

Proceed as follows:

1. Select the **Project > Accessible nodes** menu.
   The list of accessible nodes is displayed in the working area.
2. Select the relevant SIMOTION device.
3. Select **Licenses** in the shortcut menu.
   The Licenses dialog box appears, showing the actual licenses for the selected SIMOTION device.
4. You can close the window with **X** or continue with **Perform licensing...**.

Performing the licensing

To perform the licensing:

If there are no pre-installed licenses, you can acquire the licenses you need and then generate the license key required.

Requirements:

- Configuration has been completed
- The project has been saved and compiled
- The required licenses have been determined
- The license key has been determined or the serial numbers of the memory medium and the CoL are available
- SIMOTION SCOUT is in online mode

Proceed as follows:

1. Select the relevant SIMOTION device in SIMOTION SCOUT.
2. Open the context menu and click Licensing.
3. In the Licenses dialog box, click **Perform licensing...**
   If the Use wizard checkbox is activated, a wizard guides you through the licensing procedure.
   If the checkbox is not activated, the window for the expert licensing is opened. You can enter the license key there without running through the wizard.
   If you have not yet generated the license key, the wizard gives you the option to switch to the Web tool to generate one. Then switch back to the wizard.
4. If you have an online connection, continue with item 5.
   Otherwise, you can establish an online connection with **Online** in the **Step 2 of 3** window.
5. Enter the license key in the **Step 3 of 3** window.
6. Click **Finish**.
   The wizard is closed. Licensing is complete.

Note

The license key is written to the retentive memory when the project data is transferred to the target system.

Changing the license key

The license key is influenced by changes to the project, such as expanding it to include an additional axis. This is why underlicensing is displayed when the project is downloaded and the SF LED flashes at 0.5 Hz.

After you have determined the actual requirement and purchased the necessary licenses, generate the license key again. Now replace the license key already entered with the newly generated one.

License key is protected from being deleted (as from Kernel V4.1)

The license key is stored in the "KEYS" directory on the SIMOTION Memory Card.

When the controller starts up for the first time, the license key will be saved in the boot sector of the card and from this time is protected from being lost.

Operator actions cannot delete the license key in the boot sector. Also not by formatting the card or with the "Write boot sector..." function.

If the license key is no longer present on the card, it will be written again during the startup from the boot sector into the "KEYS" directory. This means that the system will repair any deletion on the "Key" file.

The license key can be changed at any time, for example, by relicensing. At the next startup, the license key will be saved again in the boot sector.

Licensing during hardware replacement

For the replacement of licensed SIMOTION components (MMC, CF, IsoPROFIBUS board or PN board), the associated license key must be assigned to the new SIMOTION component. In this case, contact the Customer Support for assistance.

Underlicensing

If SIMOTION SCOUT detects the presence of underlicensing during license verification, an entry is made in the diagnostics buffer. The verification is repeated every hour, and an entry is made in the diagnostics buffer each time underlicensing is detected.

The following information can be read from the diagnostics buffer entry:

- Number of required licenses
- Number of actual licenses
- Operating mode

As an additional warning signal, the SF LED flashes at 0.5 Hz as long as underlicensing is present on the system. Underlicensing will only be displayed if no acknowledgeable technological event is pending, as the same SF LED is used to indicate this as well.

3.3.6.14 Writing the boot sector

There are various reasons why it might prove necessary to write a boot loader, such as when new firmware is used on older hardware. Use the **Options > Write boot sector** menu in SIMOTION SCOUT to enable the boot sector on a memory card to be rewritten.

Additional references

For additional information on this topic, refer to:

- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- SIMOTION D410 Commissioning Manual

- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- SIMOTION SCOUT Online Help

3.3.7 Target system

3.3.7.1 Overview

In online mode, you can control the SIMOTION device with SIMOTION SCOUT, e.g.:

- Change and compile program sources
- Download
- Control operating mode
- Set the internal clock of the SIMOTION device
- Change Configuration Data in RUN Mode
- Control variables in RUN
- Copy current data to RAM
- Copy RAM to ROM
- Delete the RAM of the SIMOTION device (overall reset)
- Archive project data

3.3.7.2 Going online/offline with SIMOTION SCOUT

Online access points

You have two access points for communication between SIMOTION SCOUT and controllers and single drive units:

- **S7ONLINE**

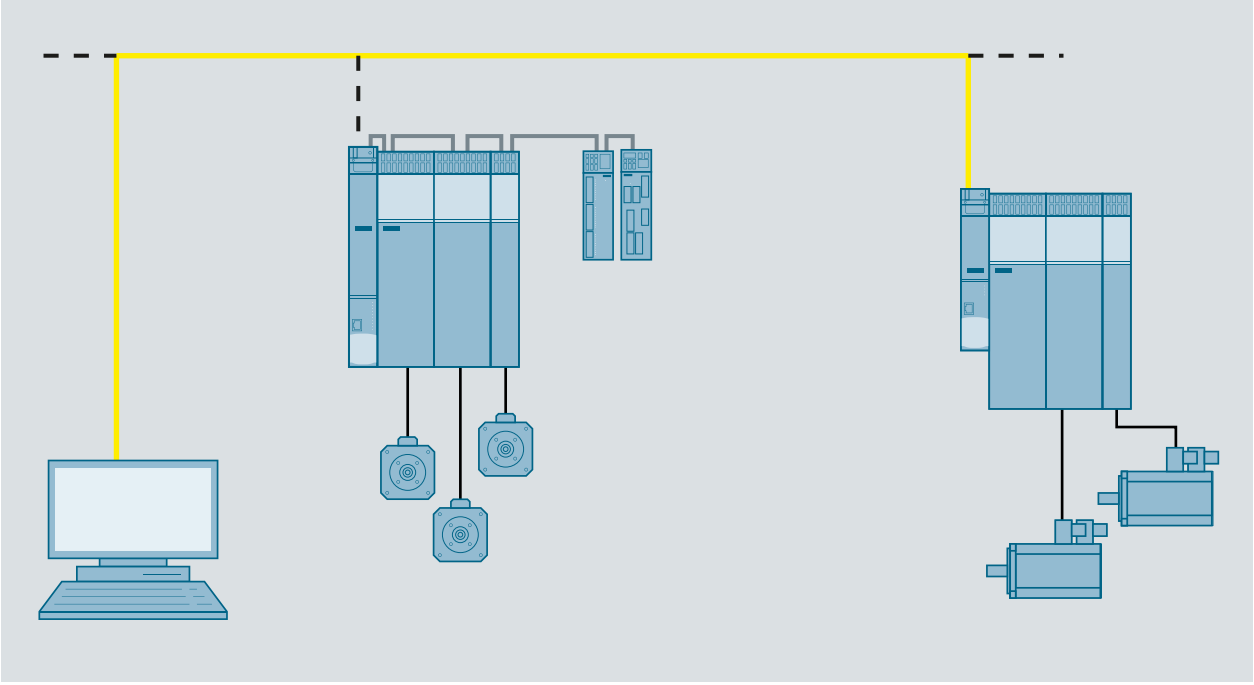


Figure 3-168 S7ONLINE

- **DEVICE**

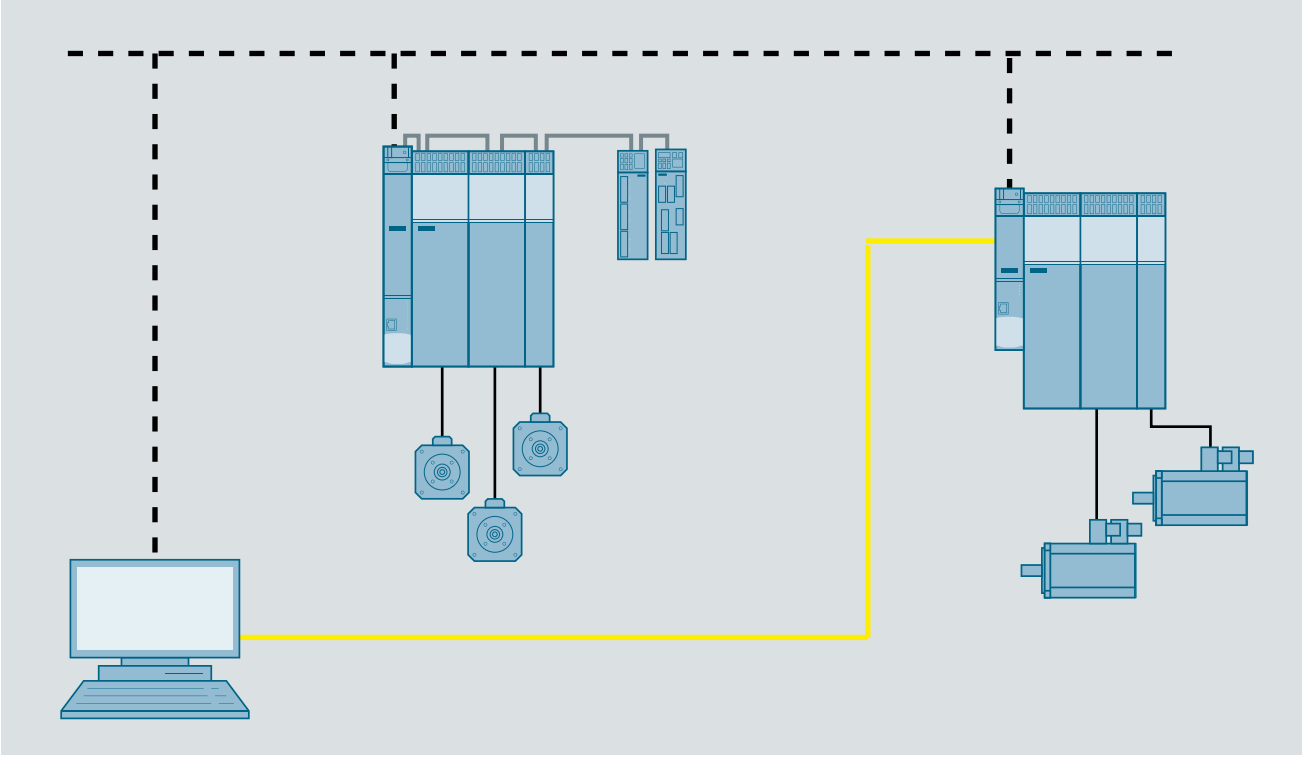


Figure 3-169 DEVICE

It is therefore possible to configure both access points for a device and to go online via these points.

Available nodes

The "Accessible nodes" function in SIMOTION SCOUT enables you to identify devices in a network, apply them to a project, and go online with them.

Note

The search for accessible nodes assumes that no online connection with an opened project exists.

Displaying accessible nodes

Proceed as follows to identify accessible nodes:

1. Select **Project > Accessible nodes** in the menu.
   All the devices found in the network are listed in the working area below the **Accessible Nodes** folder. Type information is also displayed for the devices.

Assigning interface parameterization to an access point

To create a connection between the access point, the interface parameterization, and the interface, proceed as follows:

1. Click the **PG/PC...** button to open the **Set PG/PC interface** dialog.
2. At **Access point of the application**, select the DEVICE access point to which you assign the interface parameterization.
3. Select one of the interface parameterizations to assign it to the access point, modify its properties, copy it, or delete it.
4. Click **OK** to accept the settings.
5. In the working area, click **Update** to restart the search for accessible nodes.

Searching for accessible nodes via the IP address

With TCP/IP interface parameterization you can search for nodes via the IP address. Enter the IP address in the **IP address of the sought node:** field.

For the automatic detection of accessible nodes via a PG/PC interface with TCP/IP, the nodes must be connected to the same physical Ethernet subnet as the PG/PC. Once a device is behind an IP router and so located in another physical Ethernet subnet, this device will no longer be automatically detected in the **accessible nodes**.

You can also enter an IP address and check whether it is possible to access a node downstream of an IP router with the configuration currently set on the Ethernet adapter located in the PG/PC (IP address / subnet mask).

Displaying additional information

To display additional information about identified devices, proceed as follows:

1. Right-click on a device or drive unit and select, for example, one of the following options from the context menu:
   - Device diagnostics
   - Operating state
   - Licenses

Going online with accessible nodes

Via **accessible nodes**, you can go online on devices for which no access has been configured in the project. Device type and address are determined automatically. If the device is in another subnet, you must enter the address manually in the interface properties of the device for the DEVICE access point.

Note

DEVICE must be set as access point for this function.

Assigning devices

To go online on a device that has not yet been assigned to the project, you must add the device to the project.

To add the device to the project, proceed as follows:

1. Select a device in the **Non-assigned devices - My project** or **Non-assigned devices - Accessible nodes** list.
2. Click the **Assign button to move the device to the Assigned projects** list.
3. Click **Connect to assigned devices** to go online with the devices.

Cancelling an assignment

To undo the assignment of a device to the project, proceed as follows:

1. In the **Assigned devices** list, select the desired device.
2. Click the **Cancel assignment** button.

Note

The SINAMICS Integrated or an assigned CX32-2 is always also selected for a SIMOTION D.

Setting the access point on the PG/PC

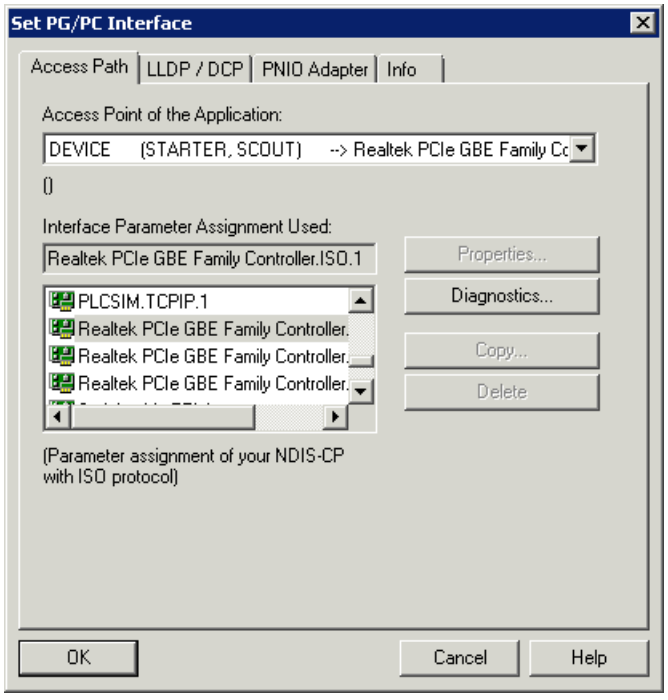
Note

The data for the PG/PC interface can only be changed when the interface is not online. If several instances of SIMOTION SCOUT have been started, an online connection may be established from only one of the instances. An attempt to go online simultaneously from several SIMOTION SCOUT applications to different CPUs results in errors.

Setting the access point

Proceed as follows to set the access point:

1. Select **Options > Set PG/PC interface** in the menu.
   The **Set PG/PC interface** dialog opens.
2. Select the desired access point at **Access point of the application**:



3. At **Interface parameterization in use**, select the interface with which you want to go online via the selected access point.
4. Confirm your selection with **OK**.

Select target devices

With the target device selection, you specify whether this device is to be used to go online when you perform "Connect to selected target devices".

Procedure

To select a target device, proceed as follows:

1. Select **Target system > Select target devices...** in the menu.
   The **Target device selection** dialog opens.

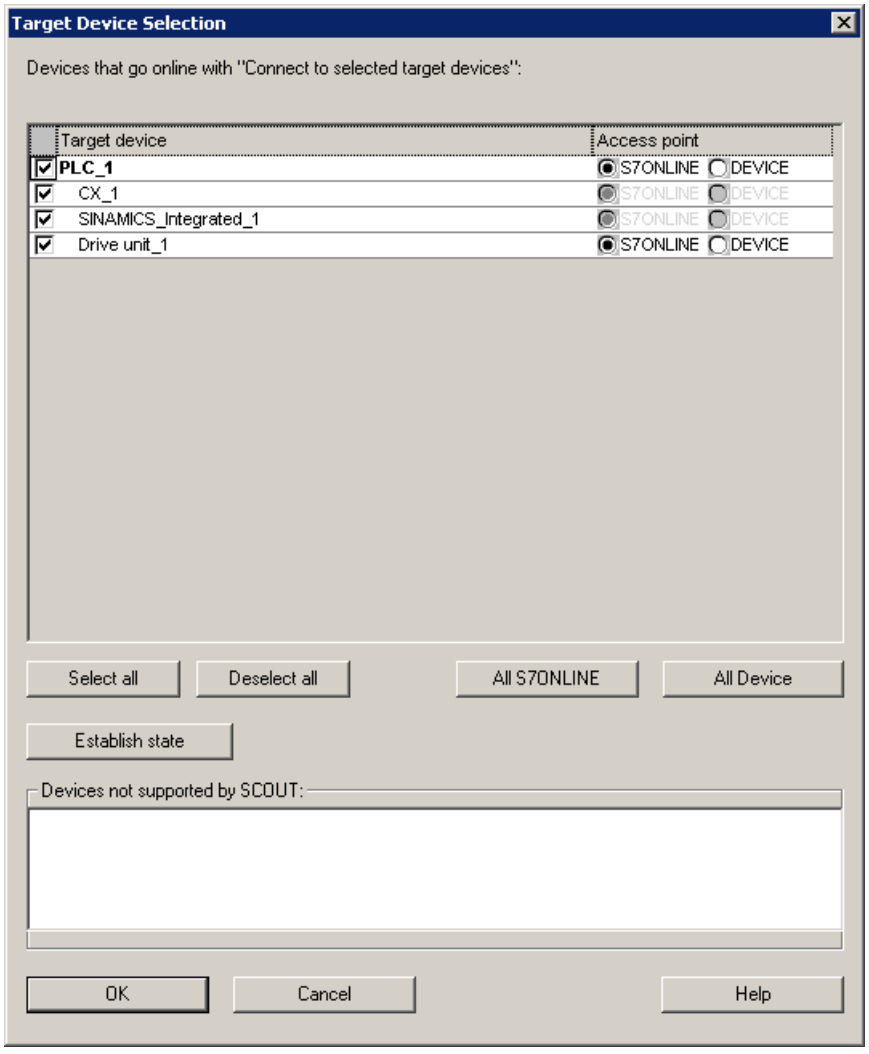


Figure 3-170 Selecting target devices and access points

2. Activate the checkbox for the respective device.
3. Click the **Establish state** button to:

- Establish a connection to a device that was deselected when going online and that has been subsequently selected.
- Separate a connection to a device that was selected when going online and that has been subsequently deselected.

3.3.7.3 Controlling the operating mode with SIMOTION SCOUT

> **⚠ WARNING**
>
> **Danger to life from unexpected machine movement**
>
> If the operating state is not switched under controlled conditions, this may endanger the safety of personnel and the machine.
>
> - Observe the safety regulations before you control a SIMOTION device via the mode selector switch in SIMOTION SCOUT.

Switching SIMOTION devices to RUN or STOP operating state

Proceed as follows:

1. Open the **Control operating state** dialog:
   Select the **Target system > Control operating state** menu. Or click the **Control Operating State** icon in the toolbar.

   The call is possible if at least one CPU of the project is in online mode.
2. In the **Control operating state** dialog, select the desired operating state **RUN** or **STOP** for the displayed devices. To do so, click the assigned button.
3. The **State** field reports whether it was possible to change the operating state.

Control operating state dialog

The dialog shows the operating state of all configured CPUs. The display can be filtered. The change of operating state is possible per device, or for several devices simultaneously.

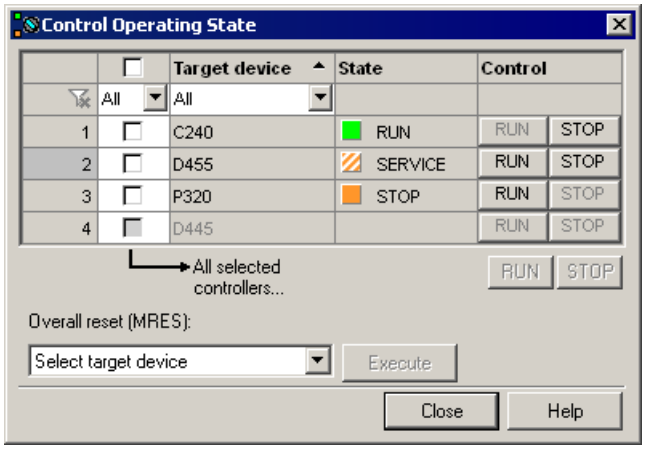


Figure 3-171 Mode selector switch as of SIMOTION V4.4

Note

As of Version 4.4, the **Control operating state** dialog represents all configured SIMOTION devices. The previous dialog had to be opened individually for each CPU.

Overall reset is possible in the dialog as before. You will find additional information about overall reset in the section Overall reset (Page 380).

The SIMOTION P State tool is retained unchanged.

Listed devices

The **Target device** column shows the names of the CPUs available in the project. The order corresponds to the order of the CPUs in the project navigator.

Offline/online distinction

If a CPU is in offline mode, the name of the CPU and all other assigned elements are grayed out in the table line.

The checkbox is always open for input.

Filtering

The list of the CPUs can be filtered using the following criteria:

- Selection by checkbox
- Name, or part of the name, of the CPU
- CPU in online mode

Filtering the display using checkboxes

The display can be restricted to CPUs that are activated in the checkbox columns. The field in the header of the checkbox column provides a choice of the following filter values.

Table 3-30 Filter values in the checkbox column

Filter value	Filter result
All	All CPUs are displayed, regardless of whether or not a checkbox is activated.
Set	Only CPUs with activated checkbox are displayed.
Not Set	Only CPUs without activated checkbox are displayed.

If the dialog is called from the context menu of a CPU, this CPU is automatically preselected (**Context menu > Target device > Operating state**).

All CPUs can be selected or deselected with the checkbox in the header.

Filtering the display by operating state or CPU name

The selection field in the header of the **Target device** column offers the following filter values:

Table 3-31 **Filter values in the Target device column**

Filter value	Filter result
All	All CPUs are displayed, regardless of whether they are in online or offline mode
Online	Only CPUs that are in online mode are displayed.
User-definable	The field can be edited, thus enabling filtering according to parts of names. Up to five user-defined filters remain for selection.

Operating states

State column: Shows the state of the CPUs in text form and via static LEDs.

Table 3-32 Operating states of a SIMOTION device

Operating state		Description
Text	**LED**	
STOP	(orange)	• Technology objects **inactive** (enables deleted, no axis motion) • User program is not executed • Loading a user program is possible • All system services are active (communication, etc.) • All analog and digital outputs set to 0 • The I/O modules (signal modules) are in the safe state (SIMOTION D)
STOP U	(orange/white)	• Technology objects are **active** • Technology objects can execute jobs for testing and commissioning functions. • Otherwise identical to STOP operating state • STOP U means stop user program • User program is not executed
RUN	(green)	• Technology objects are **active** • Execution of the user programs assigned to the execution system • Loading a user program is possible • Process image of the inputs and outputs is read or written
STOP	(orange/white)	• All tasks shut down, operating system stopped, real-time clock continues to run.
STARTUP	(orange/white)	• The display only appears if the state persists longer than 1 second, or in the event of a fault.

Operating state		Description
Text	**LED**	
SERVICE	(orange/white)	• Display, e.g. if master control has been fetched via the axis control panel.
SHUTDOWN	(orange/white)	• Display appears only if the state persists longer 1 second.
(empty)	(empty)	• CPU is in offline mode

Control operating state

A CPU can be switched to the designated operating state with the RUN and STOP switches. The switching options are dependent on the position of the mode selector switch on the SIMOTION device. The setting on the device takes priority.

Table 3-33 Switching options of the software switch dependent on the position of the mode selector switch on the SIMOTION device.

Mode selector switch position of the SIMOTION device	Switching option in the SCOUT "Control operating state" dialog
STOP	STOP
STOP U	STOP
RUN	RUN, STOP
MRES	STOP

Note

As of V4.4, manual switching to the **STOP U** operating state is no longer possible in the SCOUT **Control operating state** dialog.

If a CPU is in offline mode, both the RUN and STOP buttons are deactivated.

Simultaneous control of several CPUs

CPUs that are in online mode and that are selected in the filter column can be switched simultaneously to the RUN or STOP operating state. To do so, click the RUN or STOP switch below the list.

Observe that the switches only switch the CPUs that are visible in the dialog. For this reason, check the effect of the name filter in the **Target device** column.

Error messages

A CPU that can no longer assume the required state is indicated in color. The **State** cell changes to red or yellow/orange.

- Red: Error
- Yellow/orange: Note

A single click on the corresponding cell causes an error text or information text to appear in a roll-out tip.

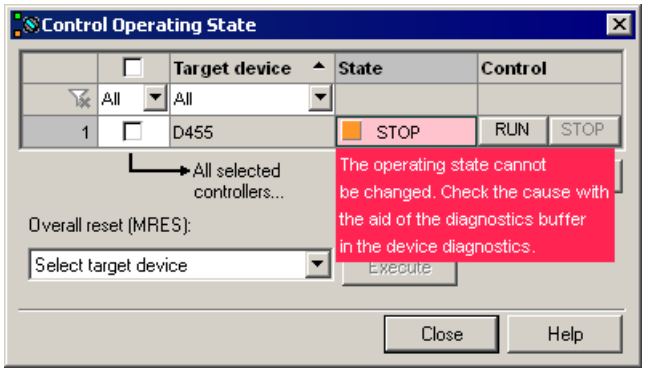


Figure 3-172 Roll-out tip with error description

Priority of the mode selector switch on the SIMOTION device

The setting of the mode selector switch on the SIMOTION device has priority. SIMOTION SCOUT can only switch a SIMOTION device to RUN mode if the mode selector switch on the device is set to **0** or **RUN**.

- **SIMOTION D**
  You can find the mode selector switch of the D410-2 and D4x5-2 in the lower area of the front behind the blanking cover.

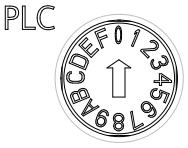


Figure 3-173 D410-2 and D4x5-2, mode selector switch, switch position 0 (RUN)

- **SIMOTION C**

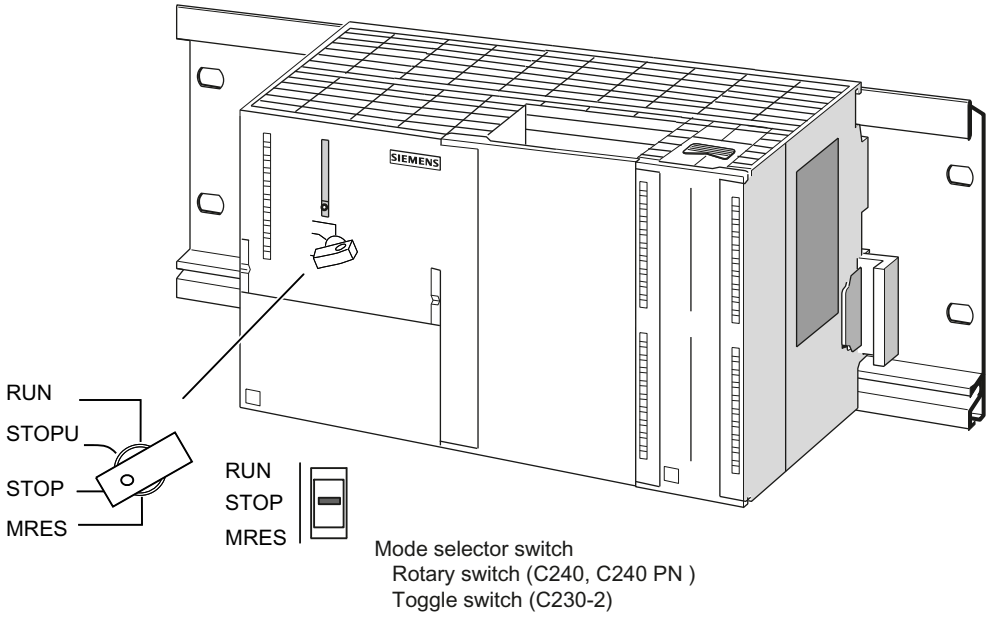


Figure 3-174 SIMOTION C240 module front

- **SIMOTION P**
  With SIMOTION P, the mode selector is displayed via the SIMOTION P Startup application. You can call this via **Start > Programs > SIMOTION P Startup**.
  For more information, see the SIMOTION P Manual.

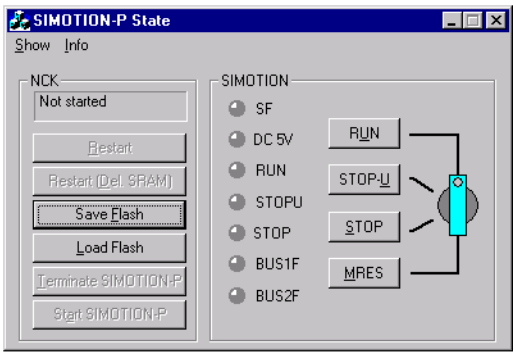


Figure 3-175 SIMOTION P state application

Additional references

Please refer to the following documents on this subject

-
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-4 E / P320-4 S* Commissioning and Hardware Installation Manual
- *SIMOTION C* Operating Instructions

and the SIMOTION SCOUT online help.

3.3.7.4 Overall reset

The SIMOTION device for which overall reset is to be carried out must be online.

Proceed as follows:

1. Open the **Control Operating State** dialog. Select the **Target system > Control operating state** menu command. Or click on **Control Operating State** in the toolbar.

   The call is possible if at least one CPU of the project is in online mode.

2. Switch the SIMOTION device for which overall reset is to be carried out to **STOP** in the **Control Operating State** dialog.
3. Select the SIMOTION device under **Overall reset (MRES)**. Click the **Execute** switch. Click **Yes** to confirm the safety prompt.
   The overall reset will now be performed.

3.3.7.5 Setting the time of day

Proceed as follows:

1. Select the SIMOTION device in the project navigator.
2. Select the **Target system > Set time of day** menu.
   The current date and the time of day of the PG/PC and the SIMOTION device are displayed.
3. Change the date and time of day of the SIMOTION device, if necessary:
   If you want to accept the values from the PG/PC:
   – Activate the **Accept from PG/PC** checkbox.

   If you want to enter values:
   – Deactivate the **Accept from PG/PC** checkbox.
   – Enter the values in the corresponding fields (**Module**).

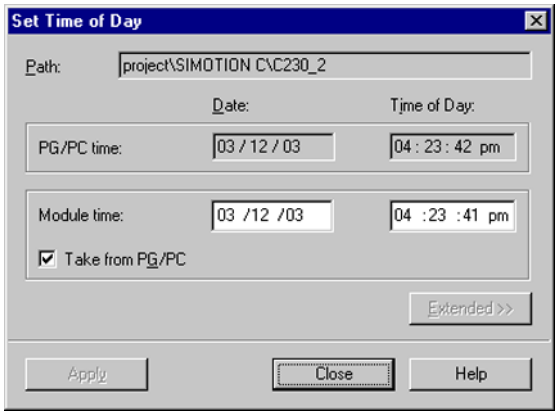


Figure 3-176 Setting the time of day

3.3.7.6 Loading data to the target system

You have to download the project data that you have created with SIMOTION SCOUT to the target system. The target system can consist of several CPUs (SIMOTION controllers).

The project data contains the programs (ST, MCC, LAD/FBD, and DCC) that you have created and compiled, the hardware configuration, and the technology objects that you have created and to which you have assigned parameters.

Additional references

The exact download procedure is described in:

- SIMOTION Basic Functions Function Manual

3.3.7.7 Archive project data to memory card

The **Target system > Load > Archive project on card...** function archives the complete SCOUT project on the MMC/CompactFlash card or the internal CFast or SSD of the P320-4 .
The **Target system > Copy archived project from card to the PG/PC...** function loads the archived SCOUT project from the card or the hard disk.

A zip file is archived. This function is only possible in the online mode. In addition to the zip file, an info file is saved. The **Target system > Load > Copy archived project from card to PG/PC** function can be used for this purpose. The following contents are written to the info file:

- The project name
- The size of the zipped project file
- The storage date

For further information, see also Section Service with SIMOTON SCOUT (Page 414).

3.3.7.8 Loading to the file system

As for a download to the target device, select **Edit > Target system > Download to file system** to save data of a device to a Memory Card / CF card directly via a card adapter on the PC. Alternatively, the device data can also be saved locally on a hard disk. The data can then be copied later to a Memory Card / CF card.

Note

A project is always created for exactly one kernel version and one device type and will only run on the relevant SIMOTION device if this exact kernel version and this device type are available. If this is not the case, the SIMOTION device remains in STOP operating state.

For further information, please refer to the online help for SIMOTION SCOUT.

3.3.8 Upgrading and project updates

3.3.8.1 General information

Prerequisite

You want to upgrade the firmware or change the platform within a station type.

Swapping modules involving SIMOTION families

Module replacement is only possible within the same device family:

- Within SIMOTION C
- Within SIMOTION D410
- Within SIMOTION D410-2
- Within SIMOTION D4x5
- Within SIMOTION D4x5-2
- Within SIMOTION P

The change involving the SIMOTION families (SIMOTION C, SIMOTION P, SIMOTION D410/ D410-2 and SIMOTION D4x5/D4x5-2) requires an XML export.

Procedure

The procedure in the Engineering System depends on what is involved:

- Upgrading the firmware version of a device within a platform
- Platform change: The platform is changed within the same SIMOTION version. For example, a SIMOTION C240 is replaced by a SIMOTION P320-4.

The sections Upgrading and changing platforms for a SIMOTION device (Page 383) and Upgrading devices and updating projects with the device update tool (Page 387) contain further information.

3.3.8.2 Upgrading and changing platforms for a SIMOTION device

Changing a SIMOTION device and subsequent TP upgrade (within a platform)

An upgrade is necessary when you want to replace the type or version of the SIMOTION device in your existing project. Perform this replacement in HW Config.

To replace a SIMOTION device:

1. Double-click the SIMOTION device to be replaced in the project navigator in SIMOTION SCOUT. HW Config opens.
2. Open the appropriate folder structure in the hardware catalog:
3. Select the module of the new version in the hardware catalog and drag this above the old module with the mouse (to the header of the rack shown in the case of SIMOTION D, to slot 2 in the case of SIMOTION C and P).

   Note

   Ensure that the module/device (SIMOTION D) to be replaced **does not get deleted.** When you change to the new module/device using drag-and-drop, the old module will be updated.

4. Confirm the displayed dialog box with **Yes** if you want to replace the SIMOTION device.
5. Alternatively, select the SIMOTION device and call the context menu by right-clicking in the header area of the device. Then select the **Replace object ...** command followed by the version required.

6. Accept the changes made to the hardware configuration with **Station > Save and compile**.
7. Close the HW Config.

Note

The data for the SIMOTION device is immediately accepted in the SIMOTION SCOUT project and the **entire project** is saved. In this way, the project also accepts all changes in the project (e.g. axis configuration).

If you are using technology packages in your project, these are updated automatically.

If the device replacement is made in the HW Config without open SIMOTION SCOUT, the update is made in SIMOTION SCOUT only after the project has been reopened.

The device replacement process differs from the platform replacement process in that it is really easy to accept project data during device replacement. Device replacement uses HW Config, whereas platform replacement (e.g. replacement involving SIMOTION C and D) requires an XML export/import.

Examples of device replacement:

- Replacement involving different power classes (e.g. D445-2 DP/PN <-> D455-2 DP/PN)
- Replacement involving generations (D4x5 -> D4x5-2)
- Replacement involving variants (D410-2 DP <-> D410-2 DP/PN; C240 <-> C240 PN)
- Replacement involving the main version (C240 V4.1 <-> C240 V4.2)

Upgrade of libraries

When the version of a SIMOTION device and the technology packages changes, the libraries created in SIMOTION SCOUT must also be adapted.

1. Select a library in the project navigator.
2. Right-click and select **Properties...** in the context menu.
   The Object Properties window opens.
3. Click in the **Technology packages** tab.
4. Activate the **Device-specific** option if the device library should be used.
5. Select in the associated tables the appropriate device and the technology packages.
6. Click **OK** to confirm the changes.
7. Select the same library.
8. Right-click and select **Save and compile** in the context menu. The upgrade is completed.
9. Repeat the operations for all created libraries.

Changing the SIMOTION platform

Platform changes involving, for example, the following SIMOTION devices:

- SIMOTION C
- SIMOTION P
- SIMOTION D

Proceed as follows:

1. Export the SIMOTION CPU data to be replaced via the **Expert > Save project and export object** context menu.
2. Create a new SIMOTION device involving a different platform. This will be automatically inserted in the project navigator.
3. Insert a new SIMOTION device or several master systems and configure them. The HW Config opens. The new station and the created master systems are already inserted.
4. Also open the old station in the HW Config. Switch to the old station.
5. Copy the DP slaves.
6. Switch to the new station.
7. Insert the DP slaves in the new station.
8. Check the configuration of the elements of the new station.
9. Close the HW Config.
10. Delete the old device in the project navigator.
11. Perform an object import on the new device with the previously exported data.

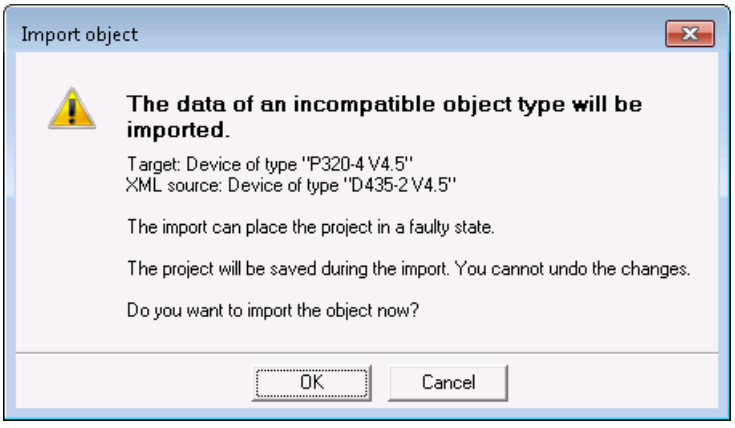


Figure 3-177 Example of importing an object with a station change

Connecting drives/slaves after platform changes

It may no longer be possible to assign drives/slaves after a platform change. To correct this, repeat the process with the original project (prior to platform replacement).

Upgrading technology packages

Overview

The SIMOTION TP technology packages (e.g. TP CAM, TP PATH, DCBlib) are available in various versions.

You can only use the functions of the technology objects selected if the technology objects are available in the target system. You can select the technology packages and their version for each SIMOTION device. Each version of SIMOTION SCOUT has a kernel (FW version) for the SIMOTION CPU and an associated technology package with the same version.

TPs during upgrades

Device replacement (in HW Config), platform replacement (XML export/import), or even upgrades may cause versions of SIMOTION technology packages (TPs), which are assigned to individual technology objects (TOs), to change.

- The TP version may change if the main version changes
  The TP version depends on the relevant main version in all cases; it may, however, remain unchanged through a number of main versions.
- If service packs and hotfixes are installed, there may even be a selection of TP product versions available for the same TP version

The TP version is automatically updated during device replacement (in HW Config). With a platform replacement (XML export/import), however, the required technology package along with the TP version and, if necessary, the product version have to be selected manually after the import.

It is also possible to define a specific TP product version (by selecting V4.1.5.3, for example).

If the product version

- "Vx.x.x.x" is displayed, the version cannot be determined (as with TP DCBlib, for example).
- "Select" is displayed, this means the TP product version has yet to be selected; if the TP is loaded to the CPU without any prior selection having been made, the latest available technology package is loaded automatically.

Selecting the TP product version

Selecting **Target device > Select Technology Packages ...** in SIMOTION SCOUT enables you to adopt a more targeted approach when choosing the technology packages you want to use.

Note

Device diagnostics can provide information on which technology package product version has been loaded to a CPU.

Loading technology packages to the target device

Technology packages are only loaded to the target device if no technology package has been loaded so far or if **Load to File System** is executed.

If a technology package version changes, the technology package must be explicitly reloaded to the target device.

Proceed as follows:

1. Select **Download project to target system** in SIMOTION SCOUT.
2. Select the **Replace product versions of the technology packages** option and confirm with **OK**.

For further information, please refer to the online help for SIMOTION SCOUT.

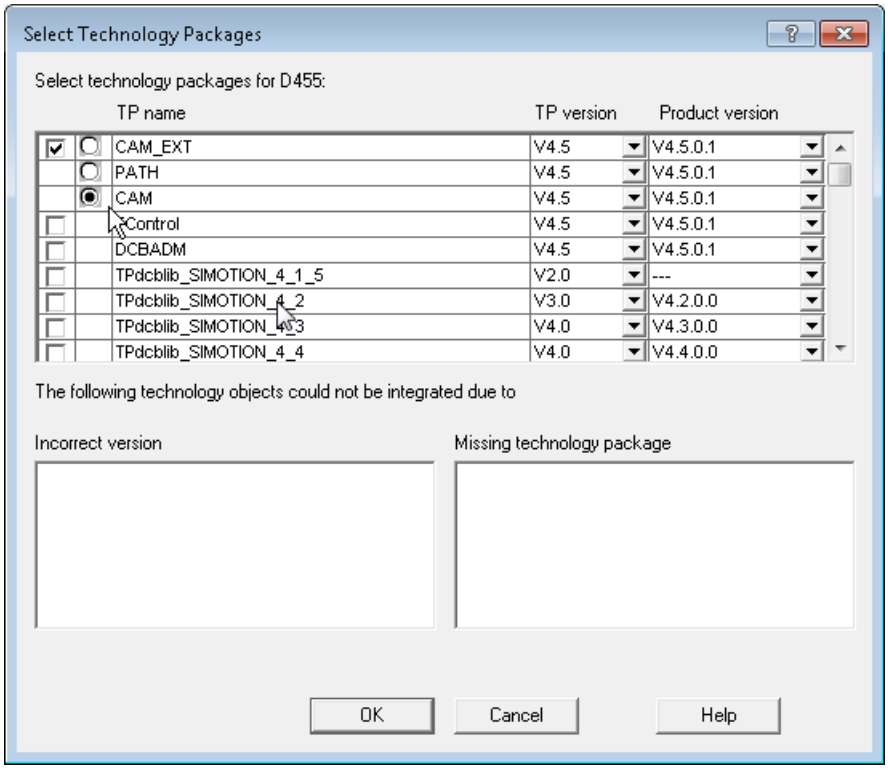


Figure 3-178 Select Technology Packages

3.3.8.3 Upgrading devices and project updates using the device update tool

Overview

SIMOTION offers a convenient solution for machine manufacturers and machine operators to update SIMOTION devices or SIMOTION projects.

Updating does not simply refer to an update to a higher version of firmware; rather, in general terms it refers to switching to a defined configuration, e.g. a project update. It is also possible to return (restore) to a previous configuration. Update or restore procedures can easily be

performed on SIMOTION devices locally or remotely. The data can be imported to a SIMOTION device via a convenient and handy storage medium or a communication connection.

Note

For SIMOTION P, there are currently only plans to update the project data, technology packages and user data. Firmware cannot be upgraded given the dependency on other Windows components.

Update data and update media

The update data is created by SIMOTION SCOUT based on one or more SIMOTION projects. All information required for the update is contained in the update data. This includes:

- SIMOTION project
- Technology packages
- User data
- Firmware

After the update data has been created, it can be handled flexibly depending on the SIMOTION device in question (SIMOTION C, D, or P). The storage or update medium can take the form of:

- CF/MMC card
- USB memory stick or
- SIMOTION IT DIAG file

Update wizard and Device Update tool

Call up the update wizard via SIMOTION SCOUT. This will give you a step-by-step guide to creating the requested update data, which is then either saved in an update archive or imported directly to an update medium.

1. Select the **Project > Start Device Update tool** menu, either directly in SIMOTION SCOUT or in an open SIMOTION project.

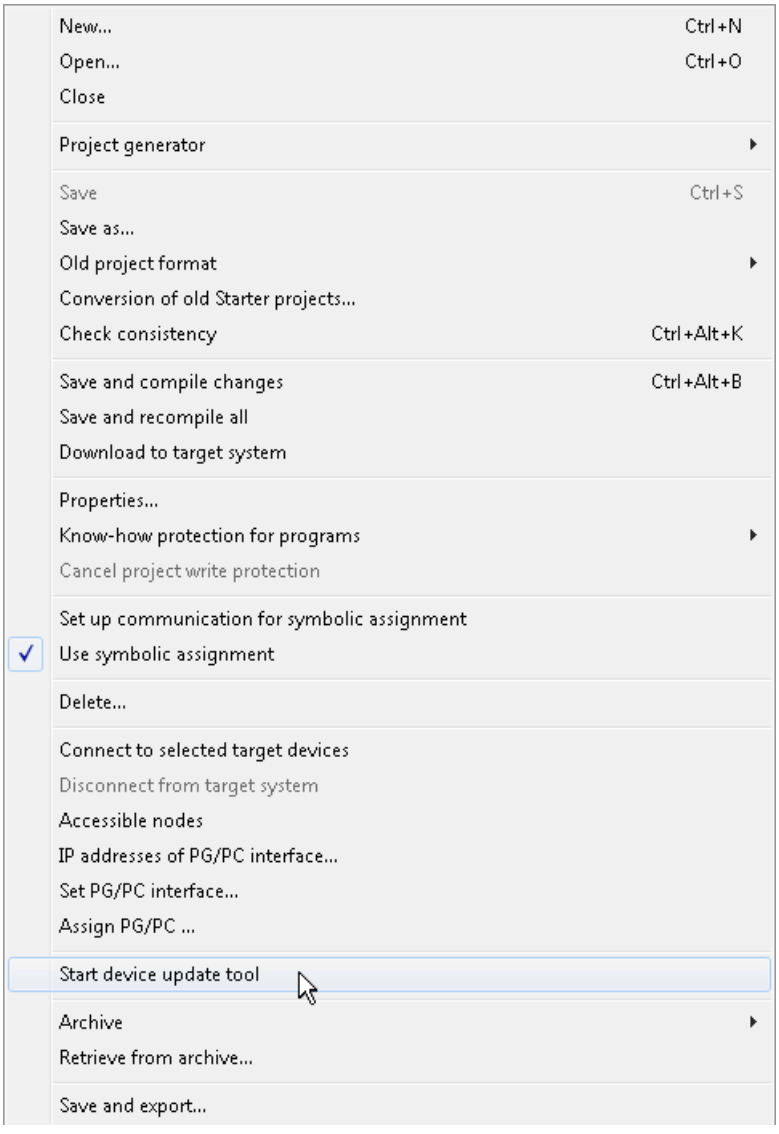


Figure 3-179 SIMOTION SCOUT - Starting the SIMOTION Device Update tool

2. The update wizard opens with the start screen. Now you can select the data for updating from the SIMOTION project to the SIMOTION device on a step-by-step basis, and specify whether you want to update subsets or the entire project.

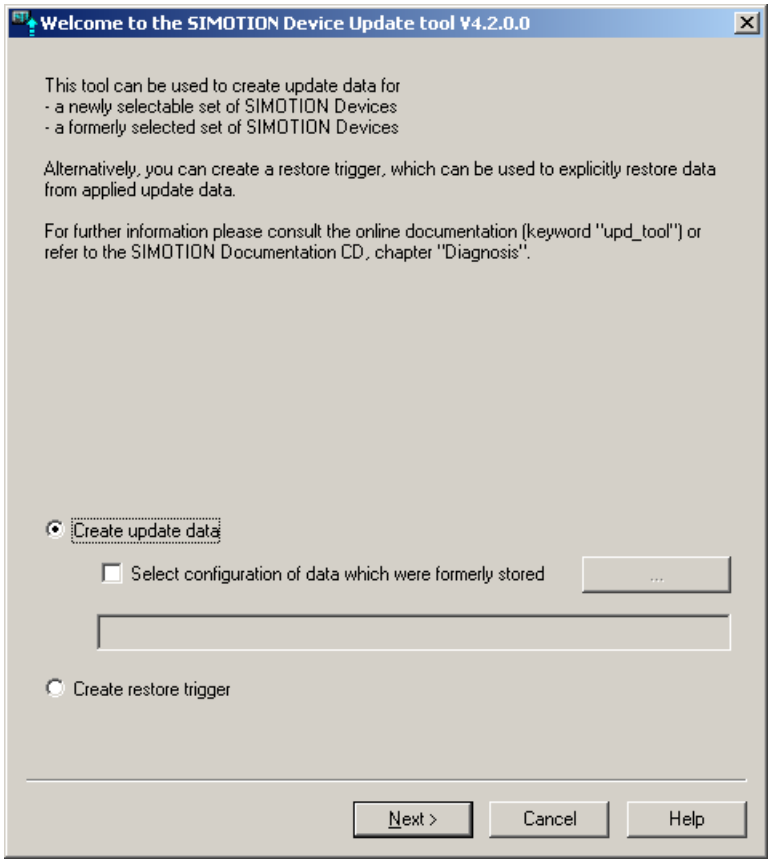


Figure 3-180 Wizard start page: Device Update tool

Additional references

A detailed overview of the update process can be found in:

- Operating Manual: Upgrading SIMOTION Devices
- SIMOTION SCOUT Online Help

3.3.9 Diagnostics

3.3.9.1 Overview of the possible diagnostic functions

A wide range of diagnostic functions can be used for the operation of SIMOTION devices in the online mode. These diagnostic options are summarized in the diagnostics overview:

- The diagnostics overview is a tab in the detail view and is available by default in online mode. You can call up detail displays from here.
- An Alarms tab is also available in the detail view. This provides a tabular overview of
  - Technological alarms (from technology objects)
  - Alarm_S messages (from user programs)
    The alarms can be acknowledged either individually or all together.
- The Address list tab in the detail area offers extended functions in terms of I/O diagnostics and hardware availability.
- Comprehensive diagnostic information (e.g. diagnostics buffer, system utilization, and task status, etc.) is accessible via Device Diagnostics as a window in the working area.
- You can record signal charts with the trace tool. The values of system variables can be recorded during runtime for diagnostic purposes.
- Program testing and debugging, e.g. variable control, program status, breakpoints

Additional diagnostic functions are available with SIMOTION V4.1 SP2 and higher. On the basis of simple operations (e.g. by setting the switch position) and without the need for the SIMOTION SCOUT engineering system, you can:

- Back up diagnostic data including non-volatile data (retain data) to CompactFlash Card (for SIMOTION D), MMC (for SIMOTION C) or hard disk (for SIMOTION P)
- Back up websites, including the most up-to-date content for diagnostic purposes, to the CompactFlash Card, MMC, or hard disk
- Restore backed up non-volatile data (retain data)

Further information can be found in the FAQs on the Utilities & Applications CD under:
FAQs > Engineering > Backing up diagnostic data and non-volatile data

SIMOTION IT DIAG also offers comprehensive diagnostic options that can be easily accessed via an Internet browser.

References

For more information, please refer to

- SIMOTION ST Structured Text Programming and Operating Manual
- SIMOTION MCC Motion Control Chart Programming and Operating Manual
- SIMOTION LAD/FBD Programming and Operating Manual
- Diagnostics Manual: SIMOTION IT Ethernet-based HMI and Diagnostic Function
- SIMOTION SCOUT Online Help
- *Overview of Service and Diagnostics Options*, Product Information

3.3.9.2 Using the diagnostics overview

The diagnostics overview is available as a tab in the detail view when the project is in online mode.

- In the detail view, select the **Diagnostics overview** tab.

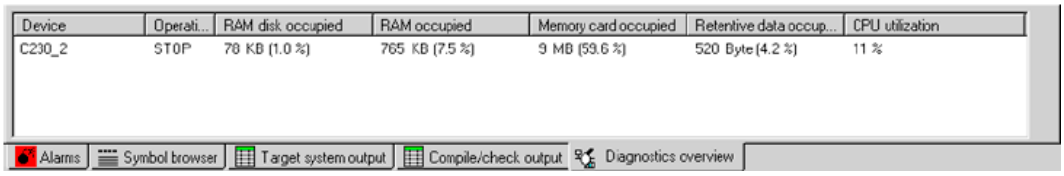


Figure 3-181 Diagnostics overview in the detail view (online mode)

The following are displayed for each accessible SIMOTION device:

- Operating mode
- Memory used (absolute and percentage display)
  RAM disk, RAM, memory card, retentive data
- CPU utilization (percentage display)

The drive devices specified in the hardware configuration are also displayed. To obtain a detailed display of the individual devices, open the device diagnostics.

3.3.9.3 Device diagnostics

Device diagnostics

In online mode, the device diagnostics function enables you to obtain a comprehensive display of diagnostics results of the individual SIMOTION devices.

Proceed as follows:

1. Select the desired SIMOTION device in the project navigator.
2. Select the **Target system > Device diagnostics...** menu.

or

1. Double-click the SIMOTION device in the Diagnostics overview tab in the detail view.

Note

You may open the device diagnostics for several SIMOTION devices simultaneously. This allows you to compare different devices.

You can also access these device diagnostics via the **Accessible nodes** function.

The **Device Diagnostics** window appears in the working area of the workbench. This window provides you with the following information:

- General information
- Diagnostics buffer
- Task runtimes
- Memory utilization

- System utilization
- User log file
- Syslog file
- Version overview
- Alarms

You have the following options:

- Print:
  Select the **Project > Print** menu.
- Save as text file:
  Click **Save as...**.
- Refresh:
  Click **Refresh** or press the **<F5>** function key.

You can also monitor and change the operating state:

- Click **Control operating state**.

Device diagnostics: General

This provides general information on the SIMOTION device:

- Select the **General** tab in the Device Diagnostics window.

The following information will be displayed:

- Name and system ID of the SIMOTION device
- Operating mode of the SIMOTION device
- MAC addresses
- IP addresses
- Subnet mask

- Standard gateway
- Article Nos. and names of the components used,
  e.g. SIMOTION device, Motion Control technology package.

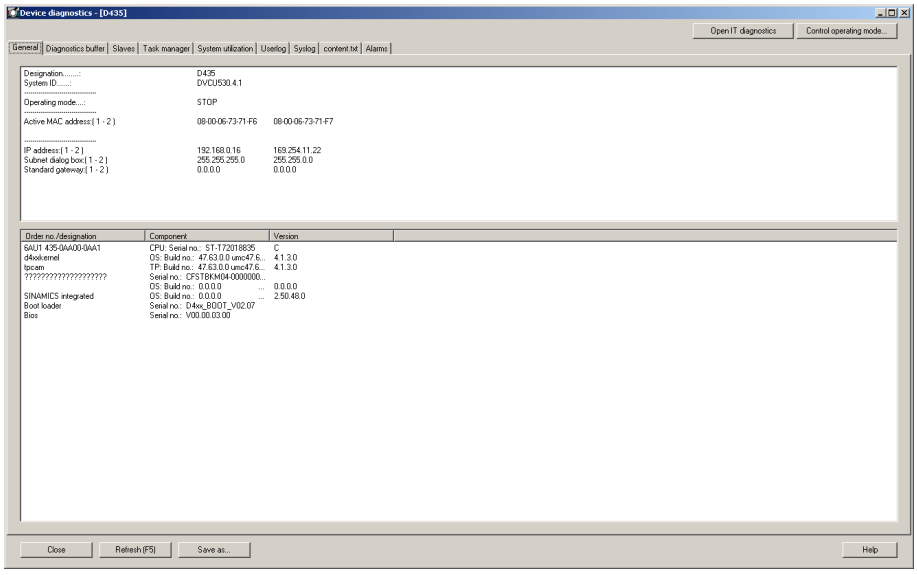


Figure 3-182 Display of general information in the device diagnostics

Device diagnostics: Diagnostics buffer

The diagnostics buffer is part of the system status list. It is possible to jump to the error position from the diagnostics buffer. It logs important events (e.g. changes in module state) in the order in which they occur. These include the following:

- Faults in a module
- Faults in the process wiring
- System faults in the CPU
- CPU operating state transitions
- Drive alarms
- Errors caused by the technology objects of SIMOTION
- Errors in the user program that caused an operating state transition (a double-click on the error message causes the cursor to jump to the error location in the program)
- User-defined entries with the _writeAndSendMessage function
- PMC error messages (SIMOTION D)
- Compatibility errors, e.g. the drive software with SIMOTION (SIMOTION D)

The SINAMICS Integrated diagnostics buffer is also displayed for SIMOTION D as of SIMOTION Version V4.1 SP2.

To work with the diagnostics buffer:

1. In the **Device Diagnostics** window, select the **diagnostics buffer** tab.
   The saved events are displayed in tabular form.
2. Select the event for which you want to obtain more information.
   Detailed information for the selected event is displayed in the lower pane of the window.

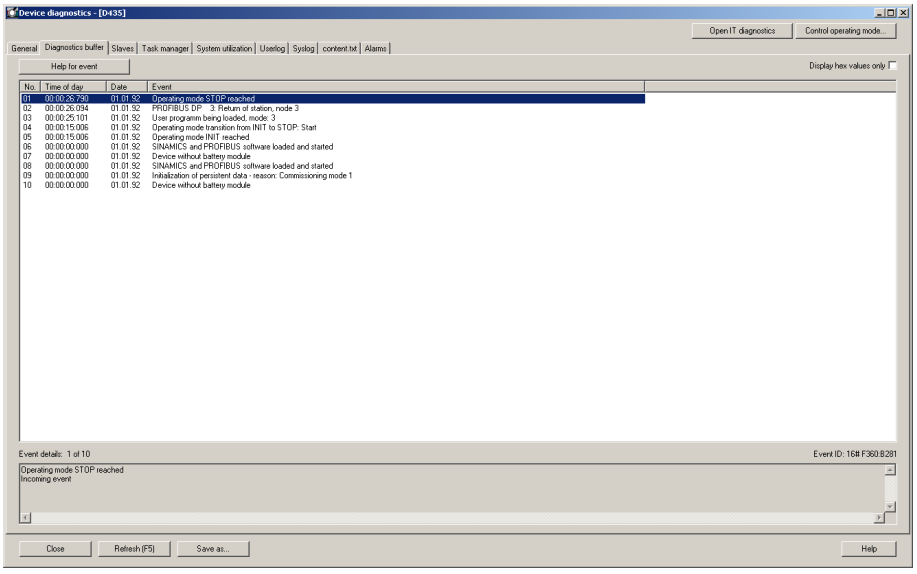


Figure 3-183 Example of the diagnostics buffer display in the device diagnostics

Device diagnostics: Task Manager

You can display the task runtimes and the status of the tasks set up in the project if you are connected online with the device. The resolution of the displayed task runtimes is performed in the servo cycle clock.

Note

The task runtimes are calculated to the µs and indicate the effective level runtime of the respective task (including the interrupt times). These thus correspond to the values of the **effectiveTaskruntime** device variables.

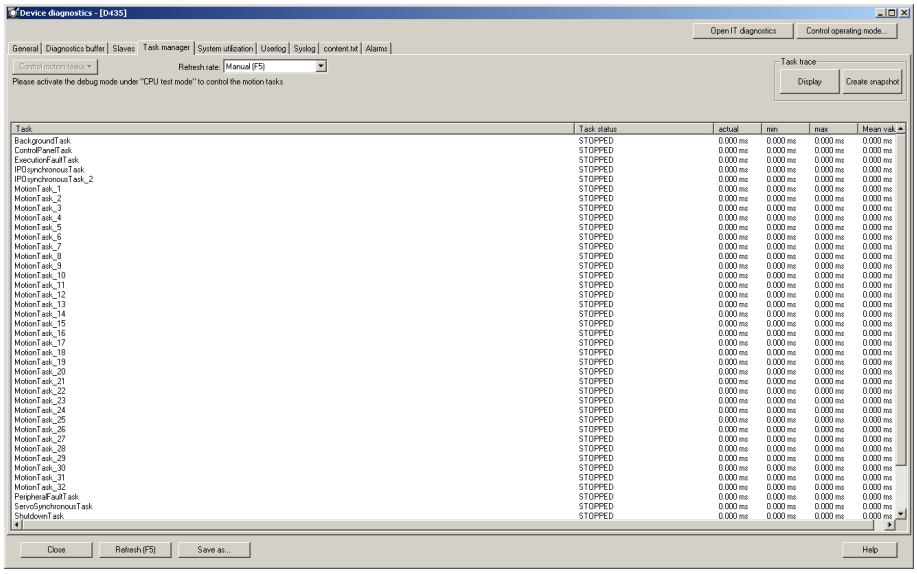


Figure 3-184 Example of the task runtimes display in the device diagnostics

The display is refreshed according to the refresh rate selected. The status and the following values are then displayed:

- Current runtime (current):
  Value of last polling
- Minimum runtime (min.):
  Minimum value since last transition from STOP to RUN
- Maximum runtime (max.):
  Maximum value since last transition from STOP to RUN
- Mean runtime (mean value):
  Value averaged from the last 10 cycles

The runtimes measured include the interruptions by higher-priority tasks.

Meaning of the various status displays:

- **RUNNING** (TASK_STATE_RUNNING)
  Task running, e.g.:
  - Via the **_startTask** function
  - As an active cyclic task
- **RUNNING_SCHEDULED** (TASK_STATE_RUNNING_SCHEDULED) (as of V4.1)
  Task interrupted by system.
  If the task status RUNNING_SCHEDULED remains pending for a long time, it identifies a long-runner in the user task, e.g. a programmed continuous loop.
- **STOP_PENDING** (TASK_STATE_STOP_PENDING)
  Task has received signal to stop; it is in a state between RUNNING and STOPPED.
  Actions may be performed until the task has stopped.
- **STOPPED** (TASK_STATE_STOPPED)
  Task stopped (e.g. via the **_resetTask** function), completed or not yet started.

- **SUSPENDED** (TASK_STATE_SUSPENDED)
  Task suspended by function **_suspendTask**.
  Use **_resumeTask***(name*) to cancel this command. The task then resumes from the point at which it was interrupted.
- **WAITING** (TASK_STATE_WAITING)
  Task is waiting due to function **_waitTime** or **WAITFORCONDITION**.
- **WAITING_FOR_NEXT_CYCLE** (TASK_STATE_WAIT_NEXT_CYCLE)
  TimerInterruptTask waiting for start trigger.
- **WAITING_FOR_NEXT_INTERRUPT** (TASK_STATE_WAIT_NEXT_INTERRUPT)
  SystemInterruptTask or UserInterruptTask is waiting for the triggering event to occur. When an interrupt occurs, the SystemInterruptTasks are started and executed once. Up to eight incoming interrupts can be stored in the buffer. If another interrupt occurs, the buffer overflows and the CPU goes into STOP operating state.
- **LOCKED** (TASK_STATE_LOCKED)
  Task locked by function **_disableScheduler**.
  This status prevents the activation of all user tasks (except the IPOSynchronousTask and IPOSynchronousTask_2) until command **_enableScheduler** is called. It does not, however, affect system tasks. The time watchdog for cyclic tasks is **not** suspended.

  Note

  This also prevents the activation of the SystemInterruptTasks and UserInterruptTasks.

Controlling MotionTasks

It is possible to control MotionTasks via SIMOTION SCOUT without a user program that has been created by the user. Consequently, you can test programs and influence MotionTask sequences in a very specific way.
Selected MotionTasks can be stopped, and locked or restarted for the sequence.

This means that programs in MotionTasks can also be downloaded in RUN mode. If you have made changes to sources and want to reload them in RUN mode, an active MotionTask can prevent this. You can then terminate specific MotionTasks with SIMOTION SCOUT and then carry out the download in RUN mode.

Additional references

Further information on downloading in RUN mode can be found in:

- *SIMOTION Basic Functions* Function Manual
- *SIMOTION SCOUT Task Trace* Function Manual

Device diagnostics: Checking memory utilization

To display the memory utilization for devices:

- Select the **Memory utilization** tab in the device diagnostics.

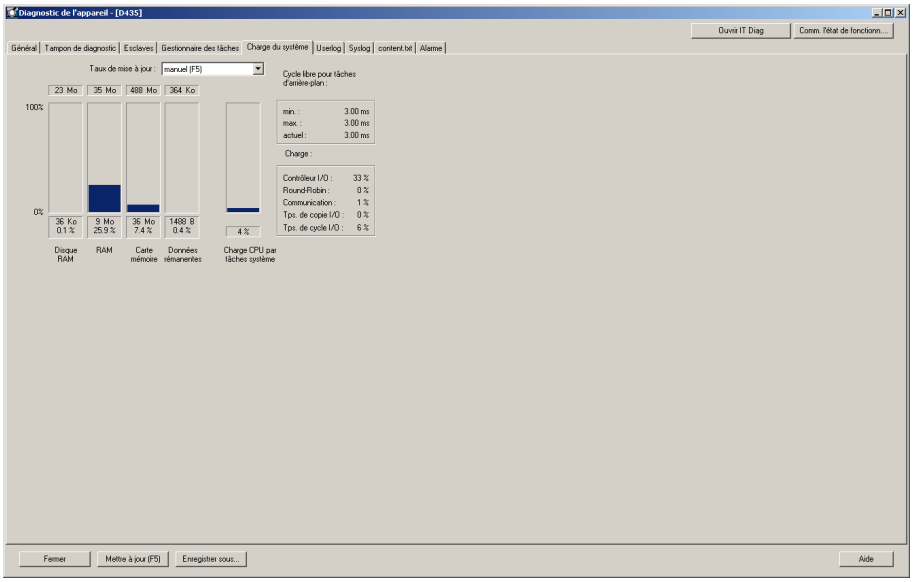


Figure 3-185 Example of the memory utilization display in the device diagnostics

Additional references

Further information on this topic can be found in:

- SIMOTION Basic Functions Function Manual, Section Overview of the memory in the target device
- SIMOTION SCOUT Online Help

Device diagnostics: Checking the system utilization

To display the system utilization:

- Select the **System Utilization** tab in the device diagnostics.

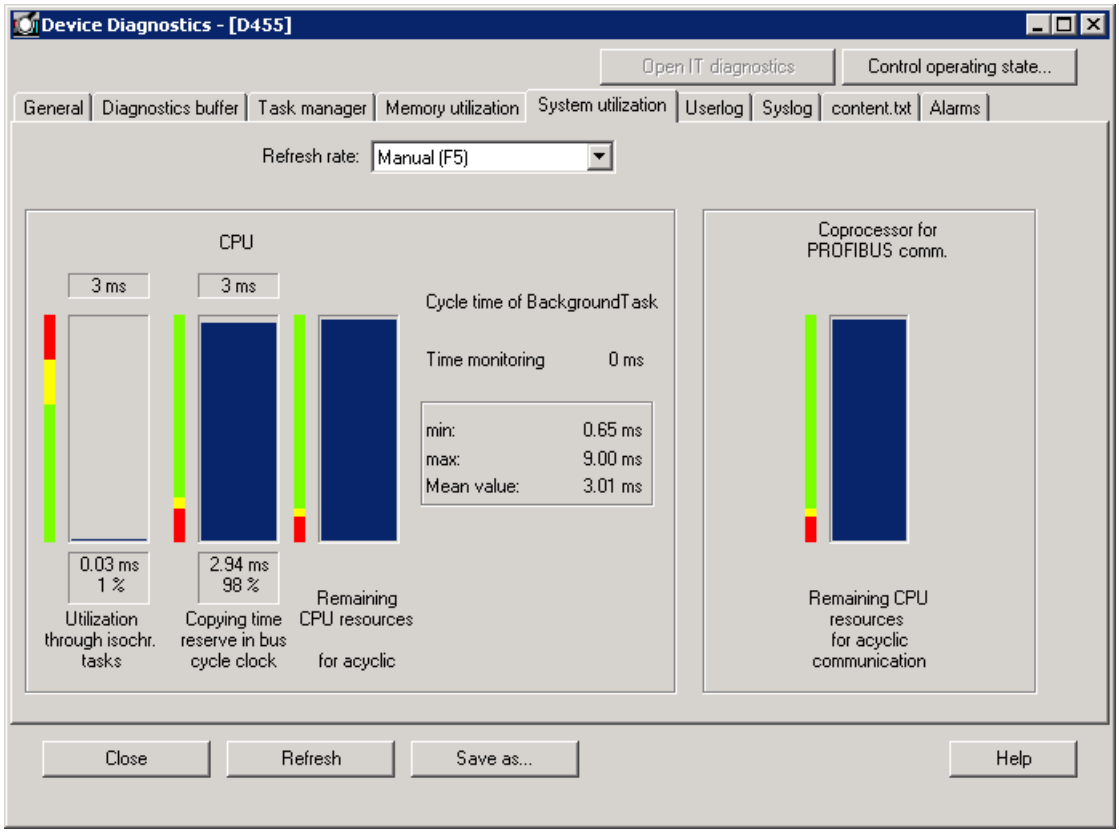


Figure 3-186 Example of the system utilization display in the device diagnostics

Additional references

Further information on this topic can be found in:

- The Function Manual "SIMOTION Basic Functions", in the chapter "Overview of Memory in Target Device"
- SIMOTION SCOUT Online Help

Device diagnostics: User log file

With the Userlog file, you can store your own text strings in the RT system. This is necessary, for example, when changes, which are to be documented, are made in the SIMOTION system on a plant which has already been commissioned.
Changes can be written in the SIMOTION SCOUT. These are loaded to the ROM of the target device. When required, the text strings can be read out again. The text editor for the Userlog file is integrated as a tab in the device diagnostics snap-in. This function is only available in online mode.

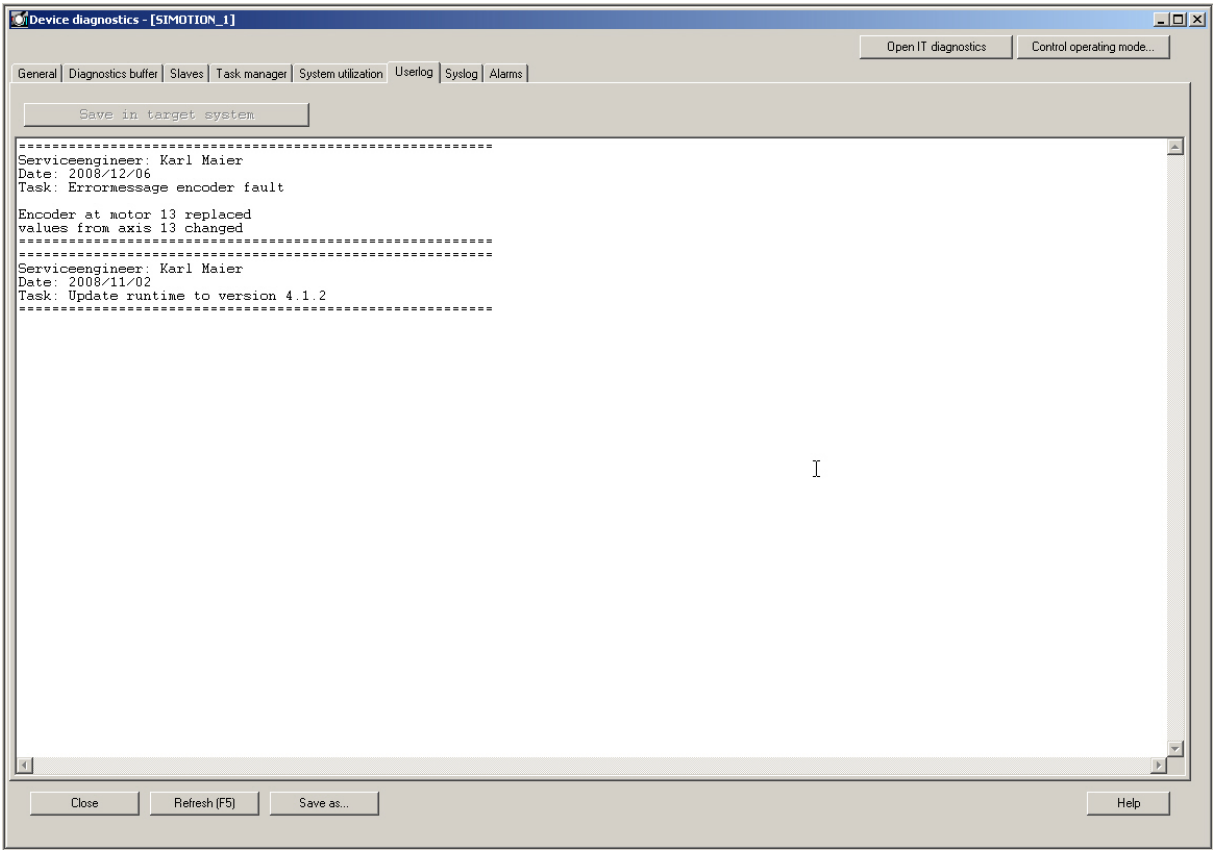


Figure 3-187 Example of the Userlog file display in the device diagnostics

How to work in the Userlog file

- Select the **Userlog** tab in the device diagnostics.
  The editor is in editing mode, i.e., you can immediately type or delete.
  The system adds no further system contents, such as date/time.
  You enter all of the necessary information.
- To save, click **Save as...** The Userlog file is stored as .txt.
  All text entries can be changed or deleted at any time.
  Access protection is not available.
- The Userlog file can also be read without a project.
  The online mode is required for this.
- The Userlog file remains after **user data are deleted**.

Device diagnostics: Syslog file

In addition to the user-defined Userlog file, the SIMOTION device also has a Syslog file. The ROM actions entered therein facilitate a subsequent diagnosis. This function is only available in online mode. The information of the Syslog file can also be read without a project.

The Syslog file logs the following actions:

- RAM2ROM
- Overall reset
- Formatting of the card from SIMOTION SCOUT

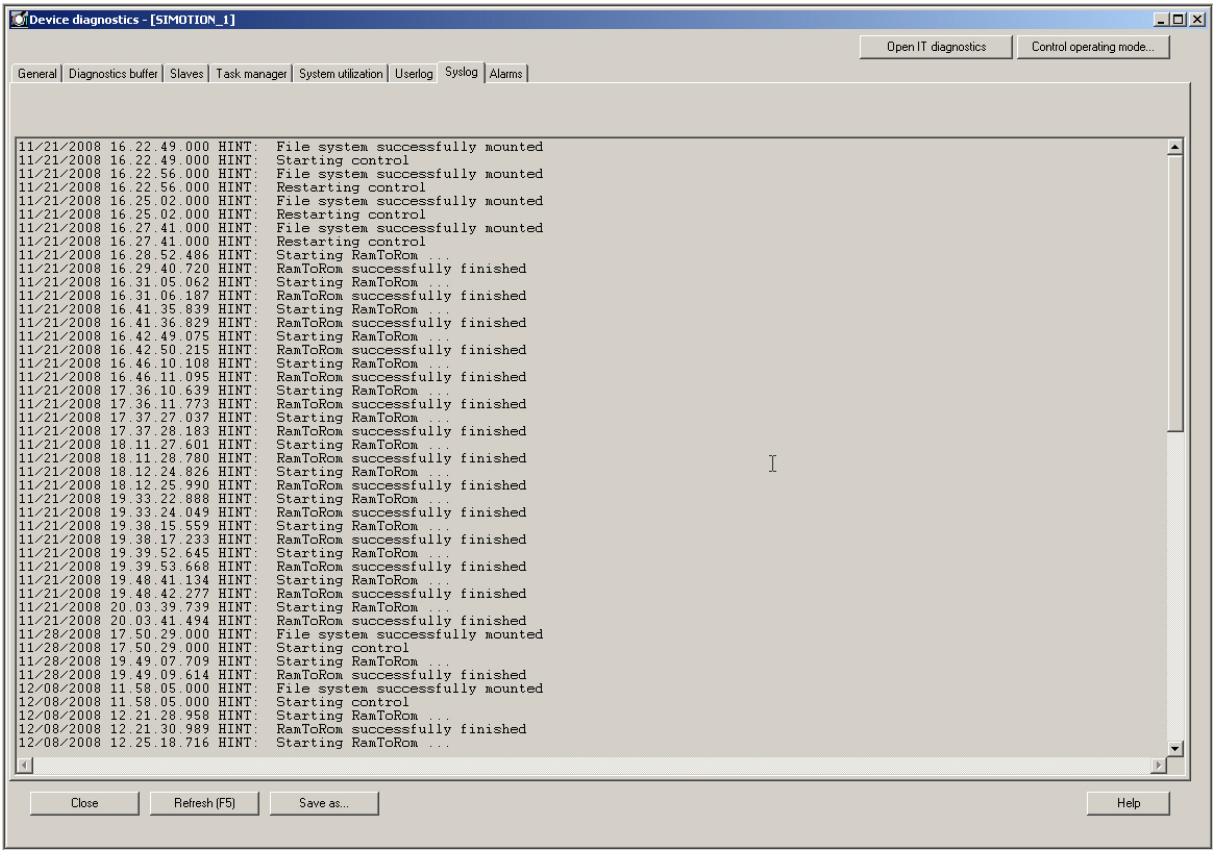


Figure 3-188 Example of the Syslog file display in the device diagnostics

Device diagnostics: Version overview

The "content.txt" tab displays the SIMOTION version and the SIMOTION device data stored in the CompactFlash card.

The following data is displayed:

- SIMOTION version
- BIOS version

- Components
  Versions of the SINAMICS components
- Internal version/stamp
  Internal components

This information is relevant for any questions to the hotline.

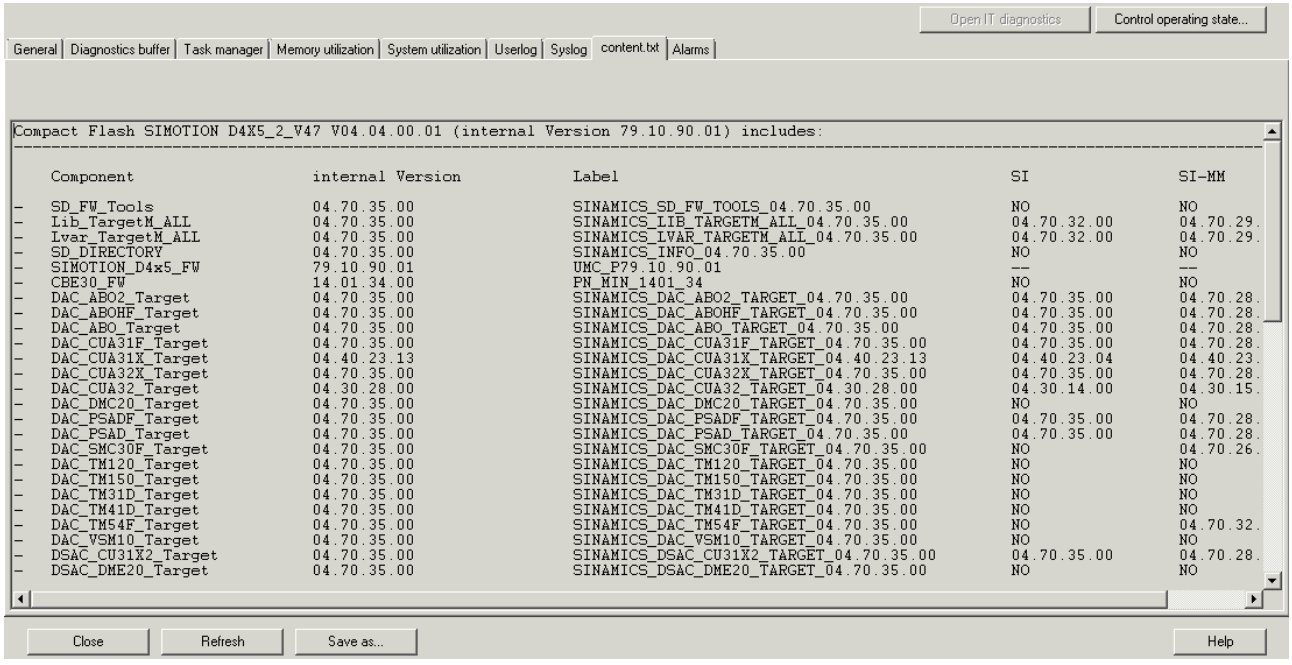


Figure 3-189 Example of the version overview display in the device diagnostics

Device diagnostics: Alarms

In the device diagnostics Alarms tab, pending alarms and configured messages are displayed in the same way as in the **Alarms** tab in the detail view.

Detailed information can be found in the SIMOTION online help, in the Alarms output window.

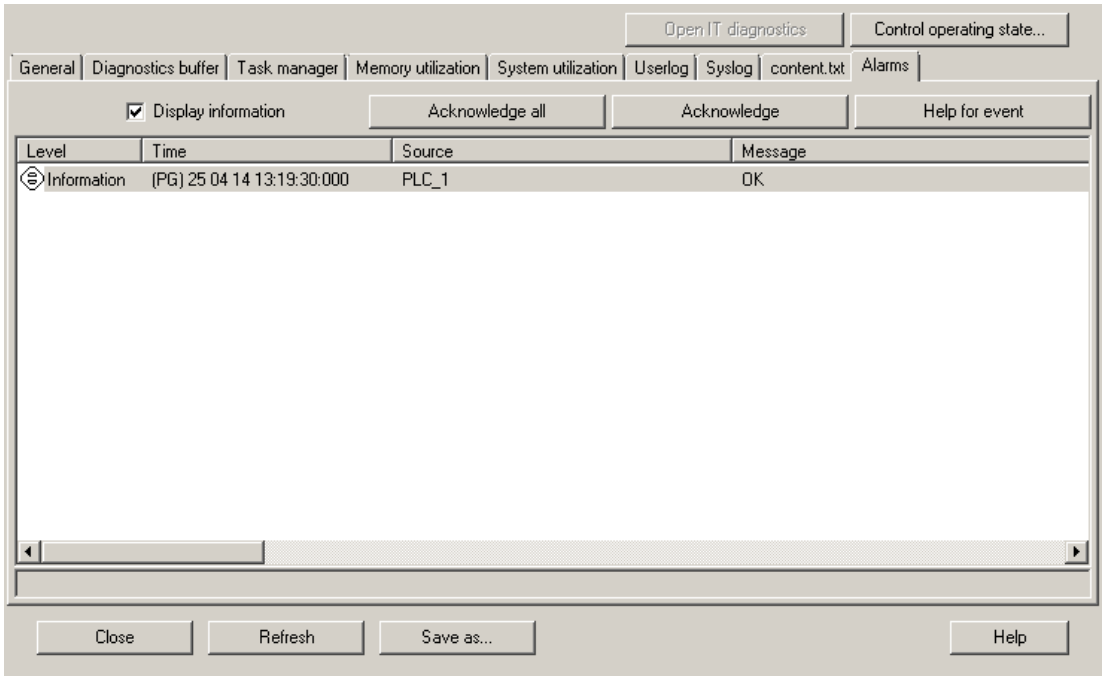


Figure 3-190 Device diagnostics - Alarms

3.3.9.4 Diagnostic functions in the address list

As of SIMOTION V4.2, the address list offers extended functions for I/O diagnostics. In online mode, the information is displayed in the **Availability** column. You can access more detailed information via a tool tip if you move the cursor over the relevant cell.

To open the address list:

1. Browse to the folder for your device in the project navigator.
2. Double-click the **ADDRESS LIST** entry.
   The address list opens in the detail area.

The summary contains the following diagnostic information:

- I/O stations which have failed completely
- Modules have been removed (e.g. for ET200S).
- Deactivated I/O stations
- I/O variables working with replacement values
- I/O stations whose set topology differs from the actual topology
- I/O stations configured to be isochronous are not isochronous
  - Distributed synchronous operation
  - Drive units
  - Isochronous I/O

- Partner device has stopped (e.g. I-device, I-slave)
- For PROFINET devices: Provider state/consumer state is showing an error
  - Controller
  - I/O device
  - Module
  - Submodule

See also the description of the **_quality()** system function in the section titled *Detailed status of I/O variables (as of kernel V4.2)* in the *SIMOTION ST Structured Text* Programming and Operating Manual.

For further information on diagnostics involving the address list, please refer to the online help.

3.3.9.5 Interconnection overview

The interconnection overview allows you to display all motion input and output interconnections of technology objects within the project. This overview is displayed in the SIMOTION SCOUT working area in the form of an interconnection tree.

The tree display enables the synchronous operation interconnections to be displayed in cascades. In the interconnection table below, you can see all the TOs interconnected on the input and output sides for the technology object selected in the interconnection tree.

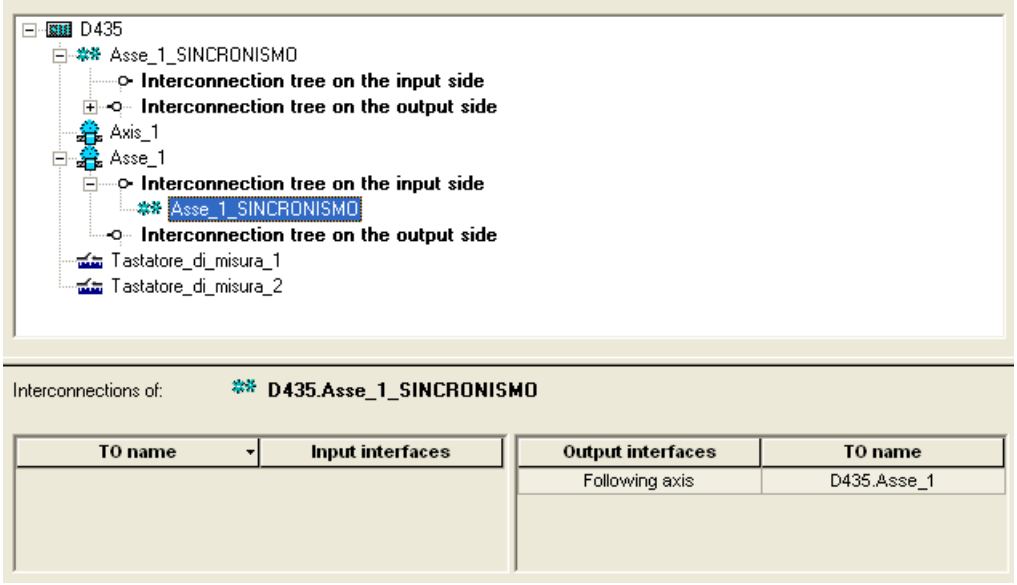


Figure 3-191 Example of an interconnection overview

Additional references

Further information on this topic can be found in:

- Function Manual: SIMOTION Basic Functions
- SIMOTION SCOUT Online Help

3.3.9.6 Service Overview

In online mode, the service overview shows a tabular complete overview of all configured axes in the project. The current state (including values from system variables) is displayed along with fault conditions.

The service overview is called up via the **Target system > Service overview** menu.

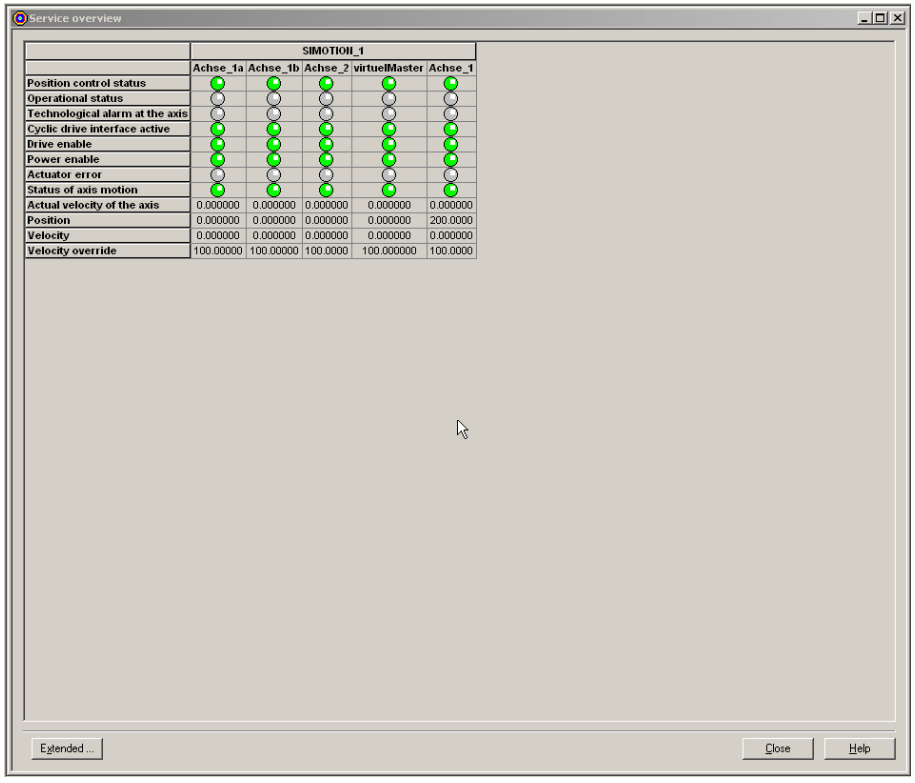


Figure 3-192 Service Overview

3.3.9.7 Trace and measuring functions

Trace, measuring function, and automatic controller setting

Overview

SIMOTION SCOUT provides support for the commissioning and optimization of technology objects and user programs: The following functions are available:

- Device trace / function generator
- System trace
- Measuring function
- Automatic controller setting

Calling trace and measuring functions

To invoke the trace and measuring functions in SIMOTION SCOUT, proceed as follows:

1. Select the SIMOTION device in the project navigator.
2. Select the **Target system** command in the menu and select the desired function.
   Or click the desired function in the toolbar.

① ② ③ ④

① Device trace / function generator
② System trace
③ Measuring function
④ Automatic controller setting

Device trace / function generator

You can use the device trace to record and evaluate parameters as well as system and program variables.

The device trace for system variables is mainly used to analyze time-synchronous sequences in the real-time system.

An automatic multiple trace is available as of SINAMICS V4.6. It can be used to trigger a recording automatically according to the trigger conditions. The measurement is stored as a YDB file (SIMOTION) or ACX file (SINAMICS) in a ring buffer on the memory card. At least five measurements can be stored in the ring buffer.
You can parameterize the multiple trace on the Trace tab at **Save to device (memory card)**.

Program variables can be traced in order to find logical errors in the execution system or in user programs. For this purpose, rather than a time-triggered measuring task, an event-triggered measuring task in the RT system can be used. The event that causes the measurement recording is the execution of a specific code position in the user program by the RT processor. To start recording, a trigger event based on the variable can be selected on the Device trace tab (on a positive edge, a tolerance range or a bit map, for example).

The function generator can be used for test purposes to dynamically generate setpoints with defined shapes (e.g. rectangle, sine) for various system variables. With the aid of the trace, for example, the system response can be recorded in order to optimize the controllers.

System trace

You can use the system trace to record and evaluate parameters, system variables, and program variables from multiple CPUs at the same time. It is essential that the CPUs communicate via PROFINET. An isochronous connection must exist between the CPUs, and the PROFINET Sync Master must be a SIMOTION device.

Function generator, mathematical processing and bit tracks are not available for recording with the system trace.

Measuring function

The measuring function is used for controller optimization.

The SIMOTION measuring functions are used to commission the axis controller without requiring a user program.

With the SINAMICS measuring function, you can directly inhibit the influence of higher-level control loops by means of simple parameterization, and analyze the dynamic response of individual drives. The free measuring function measures and averages several measurement series with parameterizable noise sources without master control. This measuring function is suitable for controller settings and for avoiding whirling resonance in magnetic suspension bearing applications with rotor systems.

Automatic controller setting

The automatic controller setting can be used to configure the speed controller in the drive. For Dynamic Servo Control (DSC), this is also possible for the position controller.

Additional references

Detailed information can be found in the SIMOTION SCOUT online help at Diagnostics.

Task Trace

Application area

The SIMOTION Task Trace supports you when troubleshooting in the SIMOTION multitasking environment. The SIMOTION Task Trace records the sequence of individual tasks, identifies user events that you can generate via a program command, and displays these graphically.

Structure of the Task Trace

The SIMOTION Task Trace includes two main components:

- The SIMOTION **Task Tracer**, which writes the task change and events to a buffer on the target device, and
- The SIMOTION **Task Profiler**, an application for displaying the recorded data

Additional references

Further information on this topic can be found in:

- Function Manual: Task Trace
- Diagnostics Manual: SIMOTION IT Ethernet-based HMI and Diagnostic Function
- SIMOTION SCOUT Online Help

Technology object trace

You can monitor commands at a technology object and track access to the system variables and configuration data.

In SIMOTION, technology objects can be influenced by configuration data, system variables, and commands during runtime. In order to represent the influences on a technology object during runtime through the user program, a TO trace / object trace, which records the last n actions at the technology object and arranges these in chronological order, is available as of V4.2.

Note

Simultaneous use of the **Trigger through program call** trigger condition for the TO trace and the device trace is not permitted.

If the TO trace and the device trace are active simultaneously, different trigger conditions must be used.

The recording can be read out from the technology object on request and represented in the ES.

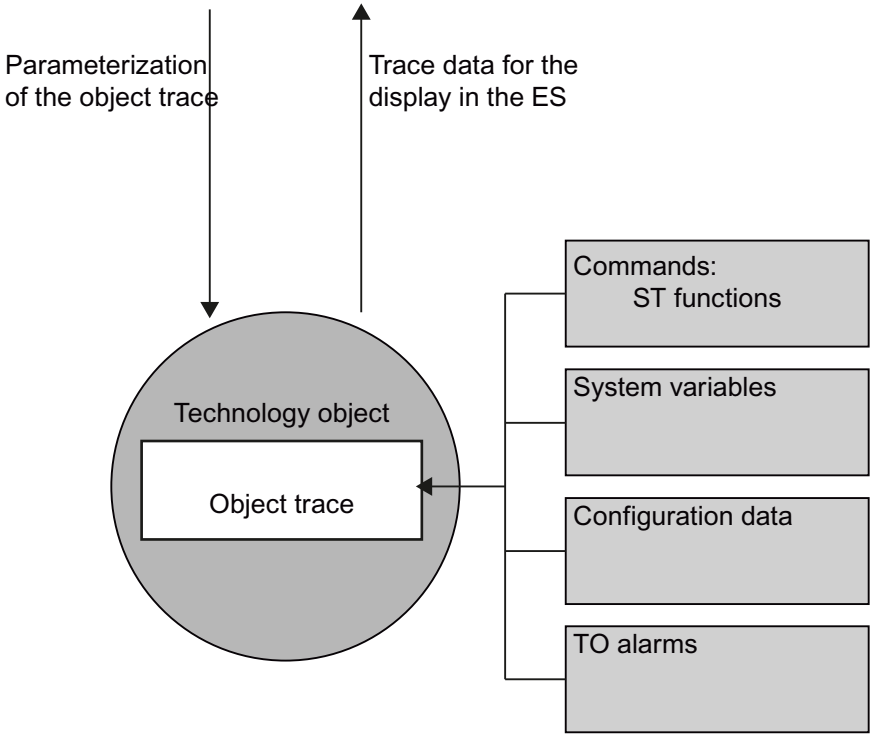


Figure 3-193 Principle of the object trace

The TO trace is used for the diagnostics of technology objects and displays the information and the events that influence a technology object during runtime in tabular form. The TO trace can be configured separately for each technology object.

It records commands and access to configuration data and system variables with their command and error statuses when the TO trace is read out. As of SIMOTION V4.4, alarms of the individual technology objects can also be recorded and an automatic trace is available. It can be used to trigger a recording automatically according to the trigger conditions immediately after a restart.

You parameterize the automatic trace in the **Settings tab** at **Save in the device (memory card)**.

The TO trace is represented in a table/list with a time stamp.

The TO trace is available for the following technology objects:

- Drive axis
- Position axis
- Following axis
- Path interpolation
- Following object
- External encoder
- Measuring input
- Output cam
- Cam track
- Addition object
- Formula object
- Sensor
- Temperature channel
- Fixed gear
- Controller object

Additional references

Additional information on this topic is contained in the SIMOTION Runtime Basic Functions Function Manual.

3.3.9.8 Accessible nodes

A list of the devices switched on and connected to the PG/PC is displayed via the **Accessible nodes** function in the SIMOTION SCOUT working area. The display is related to the PG/PC interface which is configured as the application access point in "Set PG/PC interface".

A SIMOTION project must **not** be open in order to use this function.

Proceed as follows:

1. Select the **Project > Accessible nodes** menu.
2. Select the corresponding node.
3. Right-click and select one of the following functions in the context menu. The available functions depend on the configuration.
   - Operating state ...
   - Device diagnostics ...
   - Copy from RAM to ROM ...
   - Licenses
   - Log files
   - Archived project
   - SINAMICS Upload
   - Edit Ethernet nodes...

Additional references

For additional information on this topic, refer to:

- SIMOTION SCOUT Online Help

3.3.9.9 Program testing and debugging

Comprehensive program testing and debugging functions are available in SIMOTION SCOUT. You can execute the following functions:

- Control variable
- Program status
- Breakpoints

Additional references

Please refer to the following documents on this subject:

- SIMOTION ST Structured Text Programming and Operating Manual
- SIMOTION MCC Motion Control Chart Programming and Operating Manual
- SIMOTION LAD/FBD Programming and Operating Manual
- SINAMICS/SIMOTION DCC Editor Description, Programming and Operating Manual

and the SIMOTION SCOUT online help.

3.3.9.10 Project comparison

You can use the SIMOTION SCOUT / Starter project comparison function to compare objects within the same project and/or objects from different projects (online or offline) with one another.

Objects are devices, programs, technology objects (TOs) or drive objects (DOs), and libraries.

The project comparison is available with SIMOTION SCOUT and Starter. Comparing projects is useful if you need to carry out service work on the system.

1. Start the project comparison by clicking the **Start object comparison** button.
2. The **Select comparison partners** dialog is displayed. Select the projects to be compared.

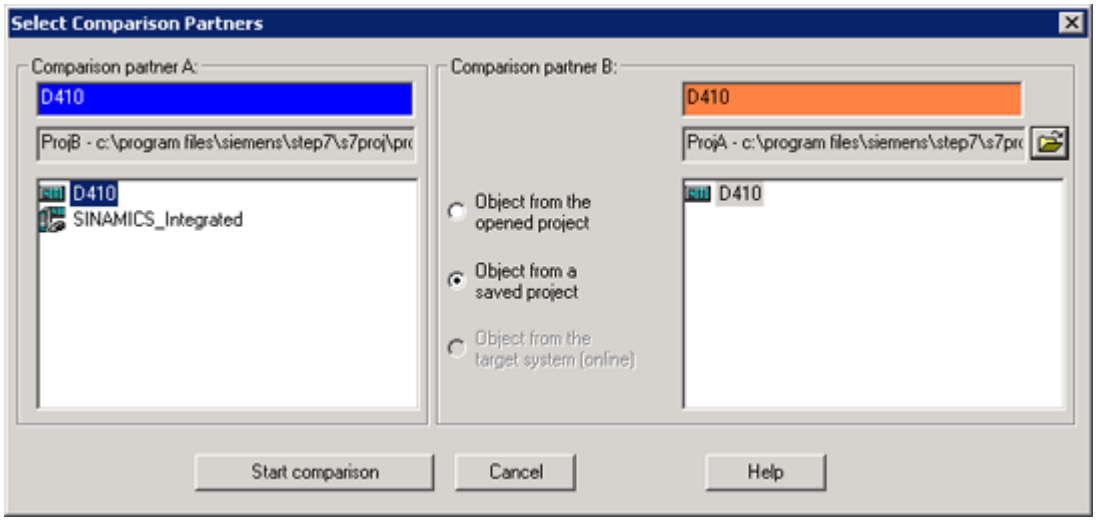


Figure 3-194 Select comparison partner dialog

Figure 3-195 Synchronizing project data via the project comparison function

Note

A detail comparison is possible only if supplementary data (e.g. program sources) has also been downloaded to the target device (**Options > Settings > Download**).

Additional references

For detailed information, refer to:

- Function Manual: SIMOTION Project Comparison
- SIMOTION SCOUT Online Help

3.3.9.11 Project overview

In Utilities&Applications, select **Scripts > Report scripts** to access scripts for generating reports on all

- TOs
- Programs, function blocks, functions
- Units
- Tasks and assigned programs

in the current SIMOTION project.

A script lists the relevant specific objects of the SIMOTION project in an HTML document, arranged in a tabular format. Once generated, the HTML document opens and is displayed automatically in the Internet Explorer. The information involved can also be saved from there in HTML or CSV format or as an Excel file.

3.3.9.12 Services and diagnostics without an Engineering System

SIMOTION devices have an integrated web server. The web server supports the display of diagnostic and system data in standard Internet browsers, even in the absence of an Engineering System, and the carrying out of project/firmware updates.

For more detailed information, see the *SIMOTION IT Ethernet-based*

HMI and Diagnostic Function Diagnostics Manual.

3.3.10 Service with SIMOTON SCOUT

3.3.10.1 Selecting the right project with SCOUT

The observation of the following recommendations for the configuring simplifies maintenance work.

Working with the right project

In order to ensure that you are always working with the right project, save the revised and archived project as a zip file on the **CF card**.

For details on how to do this, refer to the section titled Archiving and backing up projects on memory cards.
You should only use this zip file in future.

The project data can be archived on a memory card in SCOUT.

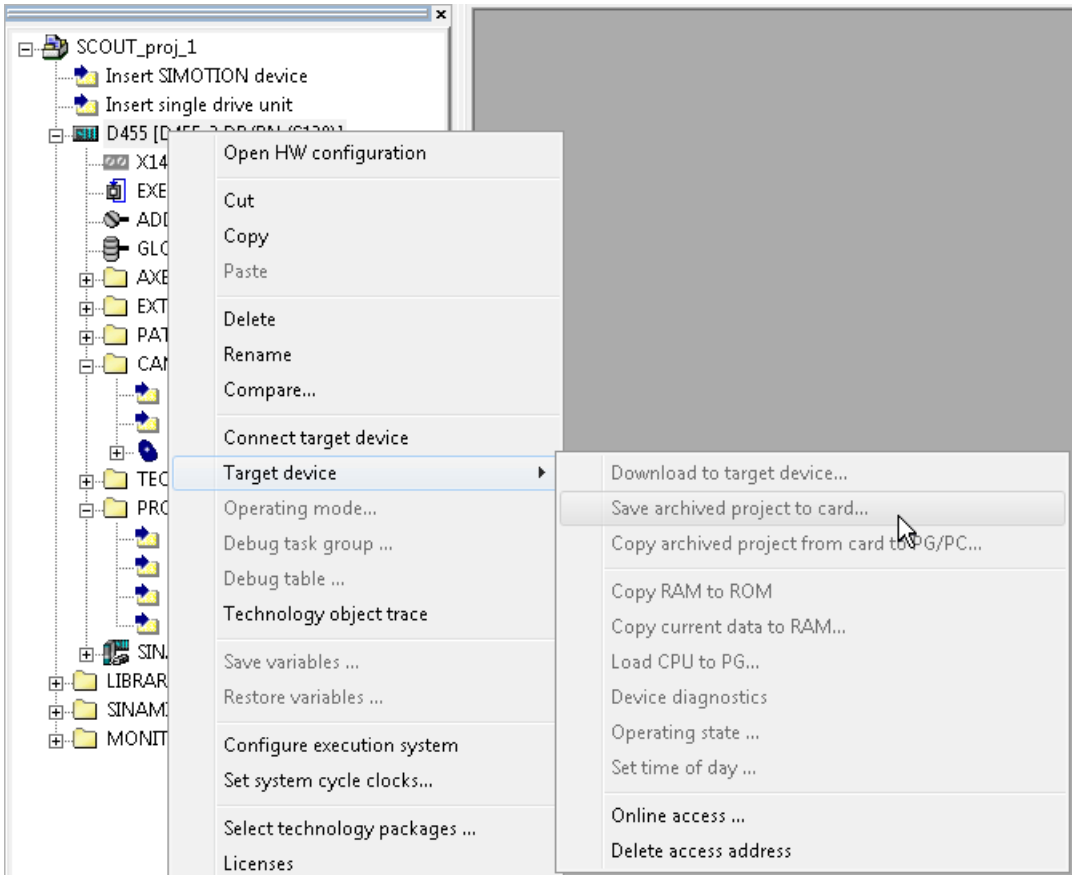


Figure 3-196 Archive SIMOTION project to card

The **accessible nodes** function can be used to determine on which CPU a Project.zip is stored.

Differences between the SCOUT project and the project data loaded to the device

Once you are online, any inconsistencies between the engineering project and the data in the target system are highlighted in the project navigator with "red" symbols.

The object comparison feature enables you to show these differences in detail and perform an alignment process.

For additional information, refer to the section titled Project comparison.

Load the project from the memory card.

The project data archived on the memory card can be transferred to "PROJEKT.ZIP" on the hard disk of the PG/PC in the **Accessible Nodes** view via context menu (right mouse button). Alternatively, the SCOUT function is also available in the **Load project from card...** context menu of the device. Next, the archive project transferred to the hard disk must be dearchived.

Use routing

The following must be considered in order to be able to use routing (e.g. for access to SINAMICS_Integrated with SIMOTION D4x5):

- If the project should be generated on another computer, a **PG/PC assignment** is necessary. The PG/PC has the unique "computer name" of the creation system and this must be adapted to the current system.
- This change can be made directly using the **Assign PG/PC** toolbar button.

The log files as additional help

The **Syslog file** and the **Userlog file** are also available to help verify the project (as of Runtime V4.0). These files are stored on the memory card and can be read out, for example, without a project on the **PG/PC** using the SCOUT function **Accessible Nodes**.

The log files on the memory card are directly accessible from the dialog (via context menu).

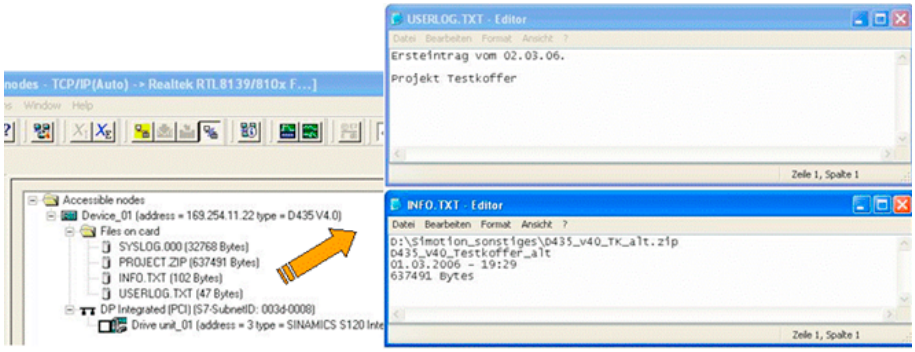


Figure 3-197 Log files via the context menu

- user\simotion\hmi\syslog\syslog.000
  System logbook: The system makes an entry here when a RAM TO ROM, an overall reset or a reformatting of the memory card is carried out.
- user\simotion\hmi\prjlog\userlog.txt
  Machine logbook: It is possible here to document the tasks carried out and the SCOUT version used for the next service worker. Call for editing from the SCOUT dialog **Device diagnostics**.

Below is other data, which is stored on the memory card and can be accessed from this dialog:

- USER/SIMOTION/HMI/PRJLOG/info.txt
  is created with the archived ES project and contains the name, size, save date.
- USER/SIMOTION/HMI/PRJLOG/project.zip
  archived ES project. Maximum one archived project can be archived on the memory card using SCOUT.

The above-named files visible in the dialog also remain on the memory card after **Delete user data**.

For a detailed description, please refer to:

- SCOUT Online Help -> Device diagnosis: Syslog / Userlog
- Device Diagnosis section: User log file
- Device Diagnosis section: Syslog file

3.3.10.2 SCOUT V4.5 and 5.1 project format

The project format of SIMOTION SCOUT V4.5 and SIMOTION SCOUT 5.1 is identical. This allows projects created with SCOUT V4.5 also to be processed with SCOUT V5.1. This also applies in the reverse case.

Nevertheless, note that SIMOTION SCOUT V5.1 includes new firmware versions for the CPUs, e.g. C240 PN V5.1. If in SIMOTION SCOUT V5.1, you add a CPU with new firmware to your project, this CPU is not known in a SIMOTION SCOUT V4.5 installation. The CPU is not displayed in the project navigator and a message appears in the detail view that the project contains unknown objects. In this case, you must continue to work with SIMOTION SCOUT V5.1.

3.3.10.3 Project was created in Version V4.1 / V4.2 / V4.3 / V4.4

If the project was created in Version V4.1 / V4.2 / V4.3 / V4.4, consider the following points:

- If SCOUT V4.5 is used to open a project created with an earlier version of SCOUT, a request is automatically made to convert the project to the current internal data format of SCOUT. In such cases, **only** the data management of the SCOUT project is converted to V4.5; the SIMOTION device version is not converted.
- After the project conversion, the system prompts whether the project should be opened write-protected.
  If the project needs to be changed, the write protection can also be revoked later.
  - To do this, select **Project > Revoke write protection** from the menu bar.

3.3.10.4 Project V4.1 / V4.2 / V4.3 / V4.4 has been edited with SCOUT V4.5

If the project of the versions V4.1 / V4.2 / V4.3 / V4.4 was edited with SCOUT V4.5, consider the following points:

- The project can be saved back in the original project version when closing the projects, exiting SCOUT or by selecting the SCOUT function **Save in old project format...**. This makes it possible to edit the project later using the corresponding SCOUT version again.

  Note

  During a reconversion, the internal compilation results of the **changed** sources may be deleted.

  This means that after the project has been opened again and the online connection made, the changed sources are inconsistent (shown in red).
  Only after the compilation and download are completed are the sources consistent again (shown in green).
  This is due to the editing with different SCOUT versions, which contain different compiler versions.

- If the project was opened write-protected for diagnostic purposes or if no changes were made in non-write protected mode, then it need not be saved.

Recommendation

After editing an earlier project version with SCOUT V4.5, the engineering project should remain in version V4.5. This eliminates the need to back-saving in the old project format so that the

project remains consistent at all times.
The engineering project is then **no longer** available in the old project version.

After the project changes are completed, they should be documented in the Userlog file.
The Userlog file is edited in the device diagnostics dialog.

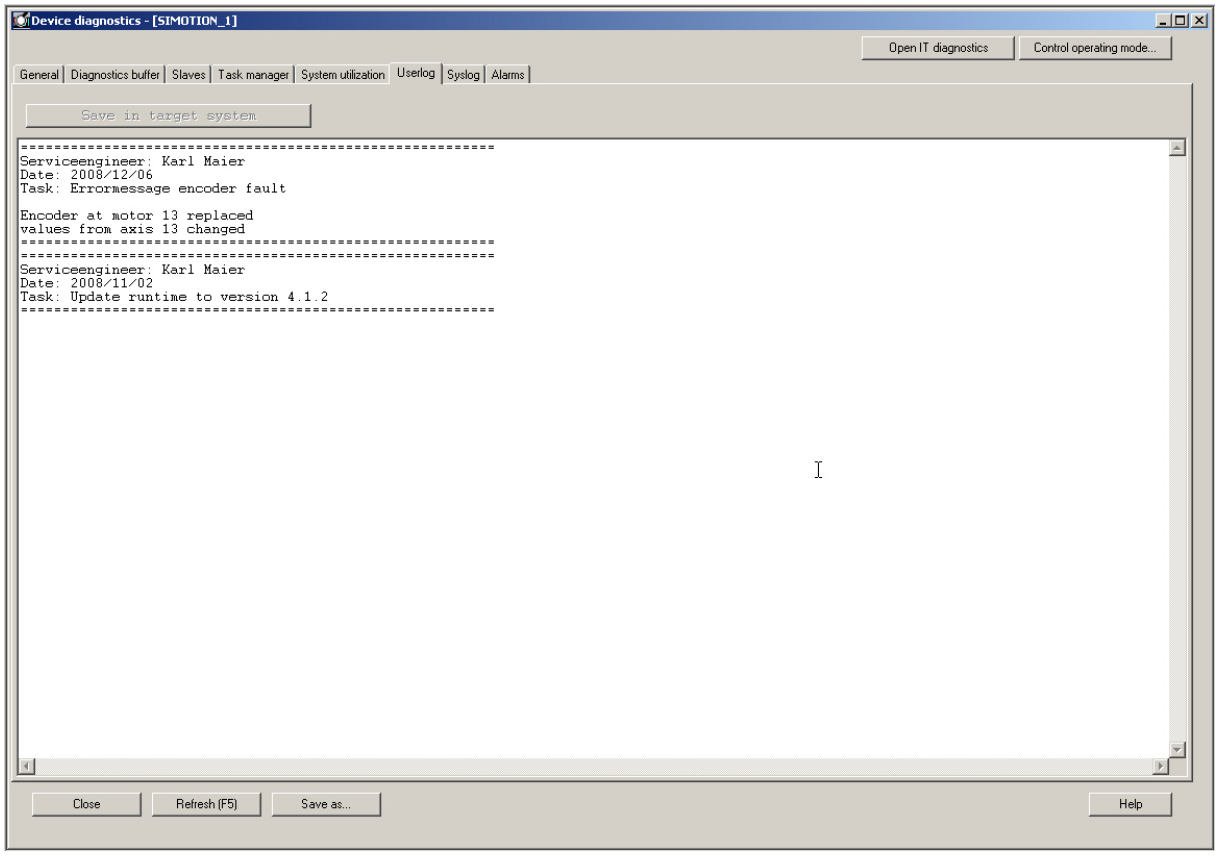


Figure 3-198 Example of the Userlog file display

3.3.10.5 Introduction of versioning with standard library and software components

It is possible to introduce versioning for any standard library and any software component during configuration, i.e., each component then has its own version identifier.

- For every unit and every library the version, author and a brief description in the comments field (offline version identifier) in the Properties dialog box in SIMOTION SCOUT.
- Each library and each software component can disclose its version identifier during runtime (online version identifier identical to offline version identifier).
  - Version identifier as constant(s) or
  - Version identifier as function return value, e.g. FCGetVersion.

- The version identifiers can be displayed separately on the available HMI systems, i.e., self-identification of all involved system components.
- A simple variant for applying versioning is in a date identifier as a constant in the format yyyy-mm-dd. This can be assigned to a variable, which is then, for example, passed on to HMI systems for the version display.

  Example, date identifier:

```
VAR_GLOBAL CONSTANT
    APP_VERSION : UDINT := 20060612; // 12.06.2006
END_VAR

VAR_GLOBAL
    G_uAppVersion : UDINT := APP_VERSION;
END_VAR
```

3.3.11 Siemens SIMOTION Diagnostics

Diagnostics dialogs in the event of a SCOUT crash

SCOUT and STARTER contain the diagnostics tool "Siemens Simotion Diagnostics". With this tool, you can help us constantly improve our product. If an internal non-recoverable fault occurs, safe continued operation is not guaranteed and consequently the application is closed. To support fast analysis and elimination of a such a fault, only data relevant to fault analysis is gathered and saved in one or more ZIP files prior to closing the application.

Fault profile:

In the event of a non-recoverable internal fault, the "SIEMENS SIMOTION Diagnostics" dialog appears.

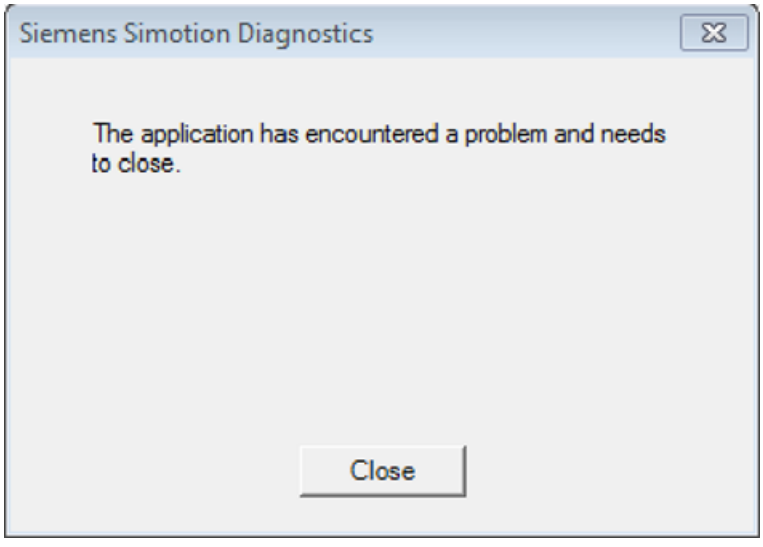


Figure 3-199 Siemens SIMOTION Diagnostics

Immediate measures:

1. Collect information about the fault (see below for instructions)
2. Contact the Siemens Hotline, http://support.automation.siemens.com

Steps for reporting the fault and support for further troubleshooting:

- Automatically created fault report
  The diagnostics tool automatically creates a fault report with technical information about the crash, such as the process image. This fault report is stored in the following directory (in compressed form as one or more ZIP files):
  <ProgramFiles>\Siemens\Step7\u7umc\data\AdvancedDiagnostics\
- Creating an error message
  The error message must include a brief description of the operator action prior to the occurrence of the problem, the software version used, and, if necessary, the configuration (installed Siemens products, hardware installation, etc.).
  After restarting SCOUT, you can see the installed versions in the "Help" menu at "Info" or by using the "System info ..." button, and you can copy and paste them into the document for fault description purposes, or to a separate text file.
- The ZIP files with the fault report must now be sent as an attachment to the Siemens Hotline http://support.automation.siemens.com together with the error message.

3.3.12 SIMOTION Runtime Simulation

3.3.12.1 Overview

Note

SIMOTION Runtime Simulation only for SIMOTION devices D4x5-2 as of V5.1

SIMOTION Runtime Simulation (SIMOSIM) is supported for SIMOTION devices D4x5-2 as of V5.1.

With SIMOTION Runtime Simulation, you can test and evaluate the general workflow of a SIMOTION runtime without connected hardware.

The simulation is so ideal for training purposes or, for example, to test and debug programs before they go online on a real module.

The projects are loaded into SIMOTION SCOUT and can be switched retentive with RamToRom. SINAMICS Integrated and external I/Os are not simulated.

Note

Simulation of SIMOTION devices

Note that not more than one device can be simulated concurrently within a project.

Restrictions

The following functions are not supported:

- CLib
- OA packages
- TPDCBADM, TPTControl, TPdcblib technology packages

Note

Technology package product versions

Note that only those technology packages integrated in the simulation (TPCAM, TPPATH, TPCAM_EXT) are supported.

3.3.12.2 Using SIMOSIM

Requirements

- You have installed SIMOTION SCOUT. SIMOSIM is part of the installation.
- You have activated the "Siemens SIMOSIM Virtual Ethernet Adapter" in the network settings of your PC.
- You have created a project in SIMOTION SCOUT and use SIMOTION D4x5-2 devices as of V5.1 in your project.
- SIMOTION devices and Siemens SIMOSIM Virtual Ethernet Adapter are contained in the same IP subnet.
- No active online connection to a real SIMOTION device exists.

Starting SIMOSIM in SIMOTION SCOUT

To start SIMOSIM and simulation mode in SIMOTION SCOUT, proceed as follows:

1. Open the project in SIMOTION SCOUT.
2. In the project navigator, select the device you want to simulate.
3. Click  "Start SIMOSIM" in the menu bar.
   The "Start Simulation Mode and SIMOSIM" dialog opens.

4. Click "PG/PC...".
   The "Set PG/PC Interface" dialog opens.

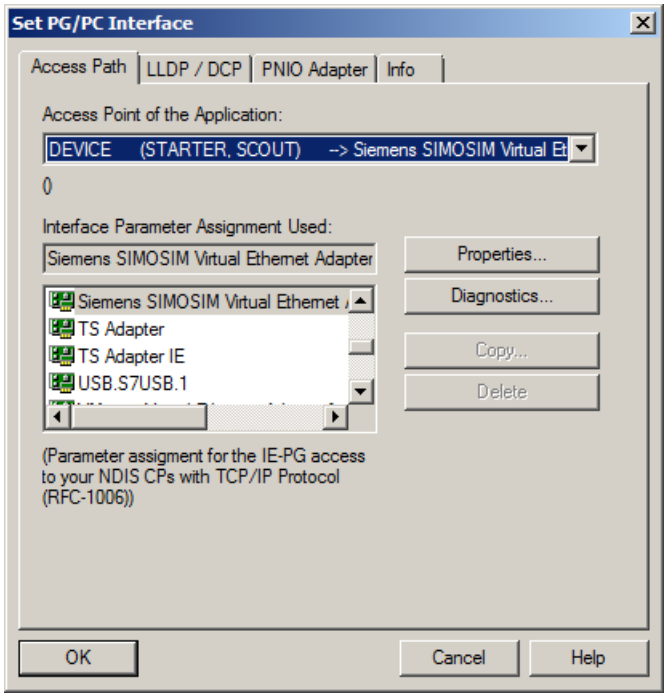


Figure 3-200 Setting the SIMOSIM PG/PC interface

5. Select "DEVICE ->Siemens SIMOSIM Virtual Ethernet Adapter.TCPIP.1" as access point of the application.
6. Confirm your selection with OK.

Note

Accepting Siemens SIMOSIM Virtual Ethernet Adapter automatically as a setting

To accept Siemens SIMOSIM Virtual Ethernet Adapter automatically as a setting, click "Accept settings" in the "Start Simulation Mode and SIMOSIM" dialog.

7. Click "Access point" in the "Start Simulation Mode and SIMOSIM" dialog.
   The "Properties > Device / Access point" dialog opens.

Figure 3-201 Access point used by SIMOSIM

8. Activate "DEVICE" as used access point.
9. Click "Set DEVICE addresses".
   The "DEVICE addresses" tab appears.

10. Activate X127 as virtual interface.
    You can use online access to the simulation only via interface X127.

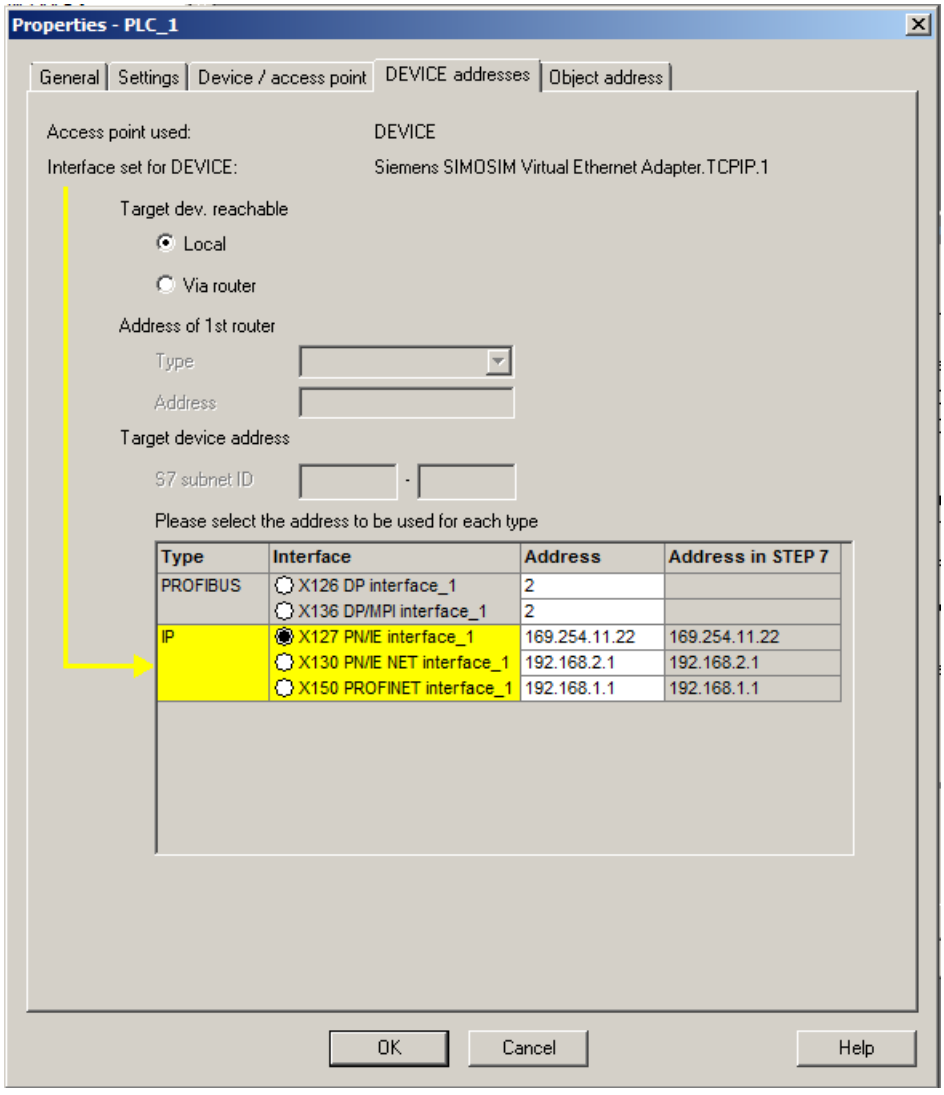


Figure 3-202 Setting the SIMOSIM DEVICE address

11. Click "OK" to confirm the selection.

12. Check your settings for the online interface.

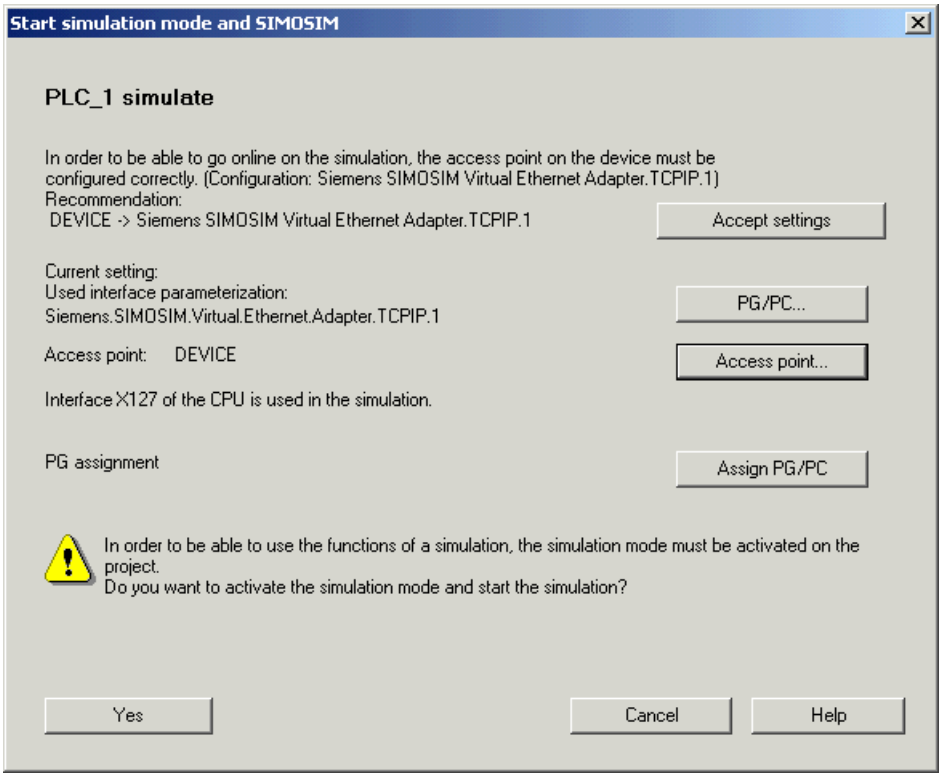


Figure 3-203 Starting simulation mode and SIMOSIM

13. To start simulation mode, click "Yes". Simulation mode is activated, SIMOTION simulation is started and a virtual controller is shown.

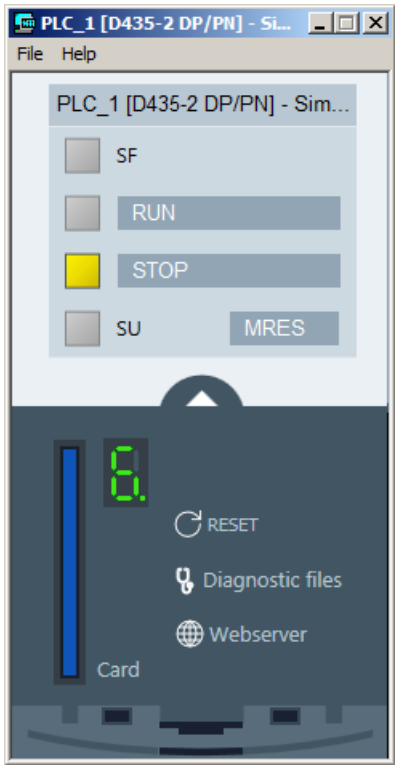


Figure 3-204 Virtual controller

The graphic user interface allows you to:

- Control the CPU operating state
- Open the Web server in the browser window
- Access the diagnostic files
- Reset the controller
- Access data on the memory card

Starting SIMOSIM in simulation mode

To start the simulation in activated simulation mode, proceed as follows:

1. Click "Start SIMOSIM" in the menu bar.
2. The "Start SIMOSIM" dialog opens.

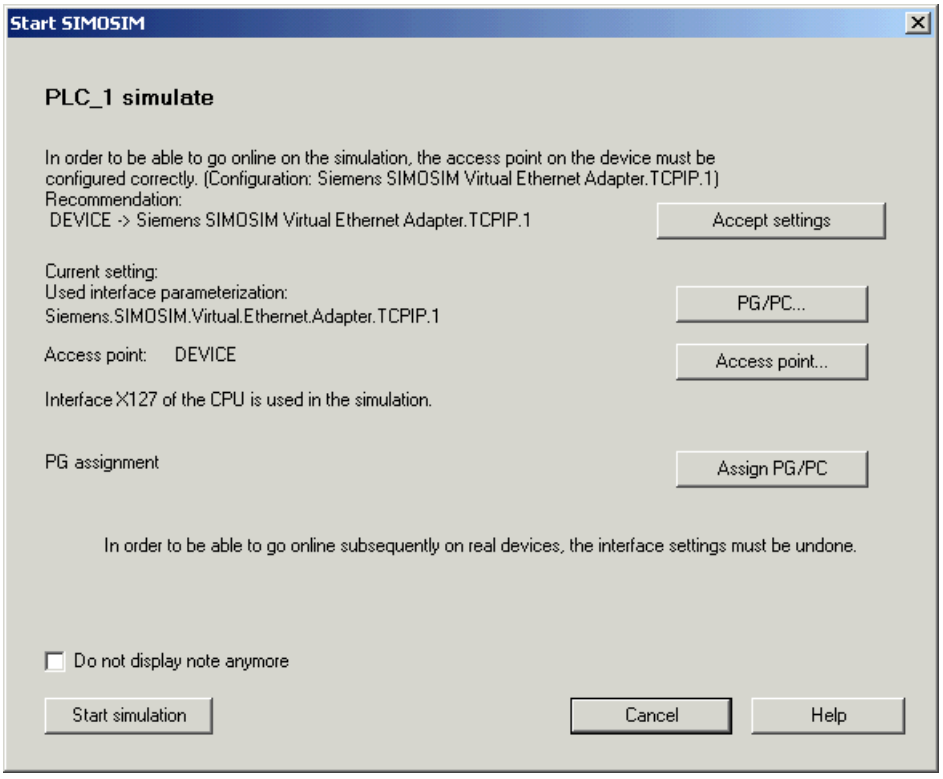


Figure 3-205 Restarting SIMOSIM

3. Click "Start simulation".

Result

SIMOSIM and simulation mode have been started.

You can now go online with a target device and download your project data into the virtual target system.

You can then simulate programs, axes, etc. You so allow SIMOSIM to write directly to the inputs.

Note

Going online in simulation mode

Note that only one device can go online in simulation mode.

Note

Simulation mode active/inactive

When simulation mode is active, this is color-highlighted in the SIMOTION SCOUT status bar.

To exit simulation mode, close the controller window by clicking the "X" in the upper area of the window. Deactivate the "Use simulation mode" option in the "Project" menu.
The flag in the status bar is reset. Simulation mode is exited.

Note

Emergency stop deactivated in debug mode

Note that in debug mode, no emergency stop is made with the space bar or by switching the application.

Note

Access to SIMOTION IT via Web browser using HTTPS

Note that the certificates must be created at the first start via HTTPS. This may take some time and corresponding messages may be displayed.

3.3.13 Configuring a further connection (such as HMI)

3.3.13.1 Rules for arranging modules in the HW Config

Slot rules

Modules must be inserted in the rack without gaps.

Exception:
Slot 3 remains empty in the configuration table. This slot is reserved.

Note

The actual arrangement has no gaps, as otherwise the bus on the backplane would be interrupted.

Table 3-34 Slot rule for SIMOTION rack 0 (SIMOTION C only)

Slot 1:	Power supply only, PS 307 xA
Slot 2:	CPU only
Slot 3:	Empty or interface module
Slots 4 to 11:	I/O modules or empty

SIMOTION SCOUT provides the following support when you configure a station:

- A message will appear if, for example, a module cannot be inserted into the desired slot.
- Address areas are also checked to prevent dual allocation of addresses.
- The status bar or messages displayed provide feedback, which you should pay attention to. Important information can also be found in the help.
- Temporary rules, that is, rules that apply only to a specific version, are not considered (such as restriction of the useable slots due to a functional restriction for individual modules).

Consult the documentation or the current product information for the modules.

Note

Modules that are installed but not configured are repeatedly addressed via the PROFIBUS. This requires additional computing time.

3.3.13.2 Routing

Routing designates the cross-network transfer of information from network x to another network y.
With SCOUT and STEP 7, it is possible to access SIMOTION, drives and S7 stations online via a PG/PC beyond the subnet boundaries, in order to, for example, load user programs or a hardware configuration or to perform test and diagnostics functions. You can connect this PG/PC to any position in the network and establish an online connection to all stations that can be reached via gateways.

Additional references

For further information, please go to:

- SIMOTION D410 Commissioning Manual
- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- SIMOTION Communication System Manual

3.3.13.3 HMI (Human Machine Interface) connection

SIMOTION allows the end user to communicate with operating devices (Human Machine Interface systems) such as operator panels.

The following configuration is possible, for example, with the SIMOTION C240:

- HMI device is connected to the non-isochronous PROFIBUS of SIMOTION device 1.
- Four other SIMOTION devices are connected to the PROFIBUS DP (isochronous) of SIMOTION device 1.

SIMOTION automatically also establishes connections between the HMI device and SIMOTION devices 2 to 5 (routing). Therefore, the HMI device can also display variables, messages, and alarms for these devices.

Note

The number of routed connections depends on the device. Between 4 and 10 routed connections are possible.
A maximum of two routers is possible.
For more information, please contact your local Siemens representative.

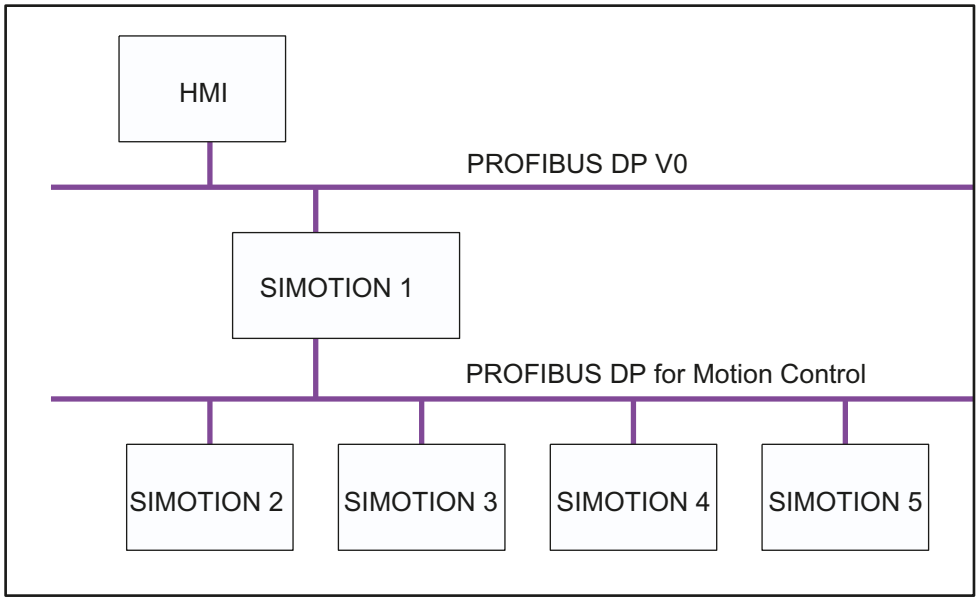


Figure 3-206 Example for the connection of an HMI device to a group of SIMOTION devices

Note

Routing by a controller (SIMOTION C, P or D) is possible only when it has been configured as active I-slave. However, please note that with an active I-slave, the PROFIBUS connection on the HMI can no longer be operated isochronously.

3.3.13.4 Higher-level automation systems

A SIMOTION device can be connected to a higher-level automation system. Communication takes place via the MPI interface of the SIMATIC STEP 7 device or the non-isochronous PROFIBUS interface of the SIMOTION device.

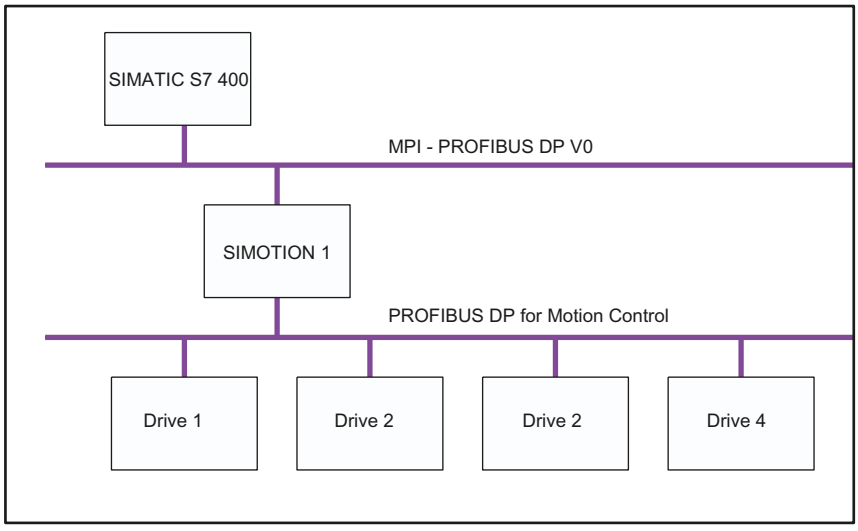


Figure 3-207 Higher-level automation system with SIMOTION device

Additional references

For additional information, refer to:

- *SIMOTION Communication* System Manual

3.3.13.5 TCP ports for access to SIMOTION/SINAMICS

TCP ports for access to SIMOTION/SINAMICS

Which TCP ports must be enabled in a router (or firewall) to permit access to a SIMOTION controller or SINAMICS drive via Ethernet?

NOTICE
Security information The functions and solutions described here are limited primarily to the implementation of the automation task. Please also note that in case of networking your plant with other parts of the plant, the company network or the Internet, appropriate protective measures within the framework of industrial security must be adopted. See also Internet (www.siemens.com/industrialsecurity).

Description

The following TCP ports must be enabled (open) for all routers and firewalls in the transmission link to permit the use of the following services: [1)]

Access to a SIMOTION controller / SINAMICS drive:

- ... via SCOUT (or STARTER) engineering software:
  - Port 102 (ISO-TSAP (Transport Service Access Point)
  - Port 5188 (only for SIMOTION)
  - Ping Request (ICMP) [2)]
- ... via web browser to the integrated web server:
  - Port 80 or 8080 (HTTP or HTTP alternate)
  - Port 443 (HTTPS)
- ... via FTP (only for SIMOTION):
  - Port 20, 21 (data communication, check)
- ... via TCP or UDP to the user program (only for SIMOTION):
  - Possible configurable port numbers are 1024 to 65535
- ... via OPC XML-DA (only for SIMOTION):
  - Port 80 or 8080 (HTTP or HTTP alternate)
  - Port 443 (HTTPS)
- ... via OPC UA (only for SIMOTION)
  - Port 4840

[1)] An open port can be verified with the following test, for example:

Under Windows, open the DOS window and enter the command "telnet <SIMOTION_IP_address> <Port>" (e.g. "telnet 192.168.214.1 102"). The space between <SIMOTION_IP_address> and the port number to be queried is important. If the screen in the DOS window turns black or if another window opens, your query has received a response via the port, i.e. the port is open. If this does not happen or if the query is canceled after some time with an error message, then SIMOTION does not respond via this port. A SIMOTION controller responds via port 5188 only if you were online via SCOUT (port 102) previously.

[2)] As of SIMOTION SCOUT / STARTER V4.3, ping-based connection monitoring can be deactivated. Therefore, the Ping Request (ICMP) may not have to be enabled. For this purpose, open the settings in SIMOTION SCOUT / STARTER at the "Tools" menu command and remove the checkmark at "Use S7-TCP connection monitoring" in the "CPU Download" tab.

3.3.14 Product combinations

3.3.14.1 Compatibility

General compatibility

Different hardware and software combinations as well as different Kernel and SIMOTION SCOUT version combinations are possible when using SIMOTION.

These possible combinations can be found in the compatibility list on the SIMOTION SCOUT add-on CD (at 1_Important) as well as on the Internet at:

Internet (https://support.industry.siemens.com/cs/ww/en/view/18857317)

Software compatibility

Technology packages with Kernel

The SIMOTION Kernel and the technology packages must always have the same version.

The current compatibility list is available online (http://support.automation.siemens.com/WW/view/en/18857317).

Project on memory card with SIMOTION Kernel

A project contains SIMOTION devices with a particular configured version. This exact SIMOTION Kernel version must be available on the relevant SIMOTION device.

SIMOTION SCOUT with project version

A project with a version earlier than the installed version of SIMOTION SCOUT will be converted to the current version when it is opened. Only the data content of the SIMOTION project will be converted to the current version, but not the SIMOTION device version.

See also

General compatibility (Page 433)

3.3.14.2 Memory media of the SIMOTION devices

Storing on storage media

The online project consisting of the technology package and the user data is saved to a non-volatile storage medium. The storage medium varies depending on the SIMOTION platform.

- SIMOTION C: Micro memory card
- SIMOTION P: Virtual memory card
  Handling and functionality correspond to the micro memory card of the SIMOTION C.
- SIMOTION P: CompactFlash card
- SIMOTION D: CompactFlash card

Note

The memory card must not be inserted or removed during operation. Removing the card in operating state triggers an overall reset, which results in the loss of stored data.

Additional references

Please refer to the following documents on this subject

- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- SIMOTION D410 Commissioning Manual
- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual
- SIMOTION C Operating Instructions

and the SIMOTION SCOUT online help.

Retentive data

Retentive data (non-volatile data) is saved to a non-volatile memory in the SIMOTION device. Depending on the SIMOTION device involved, this non-volatile memory buffers the data permanently (e.g. NVRAM for D410-2, D4x5-2).

You can also back up the non-volatile data to a memory card (e.g. CF or MMC) to protect data in the event of a spare part failing.

Use the _savePersistentMemoryData system function to back up the non-volatile data to the memory card.

In the event of a buffer failure or module replacement, the data is automatically restored at the time of ramping up.

Additional references

More information on this topic is available in

- Function Manual: Basic Functions

3.3.14.3 STEP 7

SIMATIC Manager

Open a SIMOTION project

You can also open a project in the SIMATIC Manager and edit it with the tools provided. However, you do not have any direct access there to specific SIMOTION components, such as technology object and programs.
You can open the SIMOTION SCOUT from the CPU.

Note

You can use **File > Manage...** to show and hide the projects you have created in SIMOTION SCOUT in the SIMATIC Manager.

SIMATIC Logon

Overview

As of SIMOTION SCOUT V4.1, you have the option of assigning a project password to provide access protection for projects. This functionality requires the previous installation of the SIMATIC Logon STEP 7 option.
This function makes it possible to restrict project creation to authorized personnel only and to track changes in versions created with SIMATIC Logon. It also therefore provides support for the validation process of a machine or system [according to FDA 21 CFR Part 11].

Installation / Prerequisites

The STEP 7 options SIMATIC Logon and Version Trail must be installed and licensed in addition to SIMOTION SCOUT.

Characteristics

When SIMATIC Logon is installed, password protection can be activated for any project. A SCOUT project saved with SIMATIC Logon can only be opened and edited with password protection.

- When a project that has been password protected with **SIMATIC Logon** is opened in SIMOTION SCOUT, the access dialog in which the project password must be entered appears. The project so can be opened only with the correct password.
  If an incorrect password is entered, an error message appears and the project remains closed.
- Passwords are assigned in a separate administrator screen using a process configured according to the guidelines according to FDA 21 CFR Part 11.
- Every file open and every change is automatically recorded in a revision log.
- SCOUT projects created with SIMATIC Logon can be archived, managed and their changes tracked in a clearly organized fashion using the SIMATIC Version Trail option.

Activate SIMATIC logon

When SIMATIC Logon is installed, password protection can be activated for any project.

To do this, select **Options > Access Protection > Activate** in SIMATIC Manager.

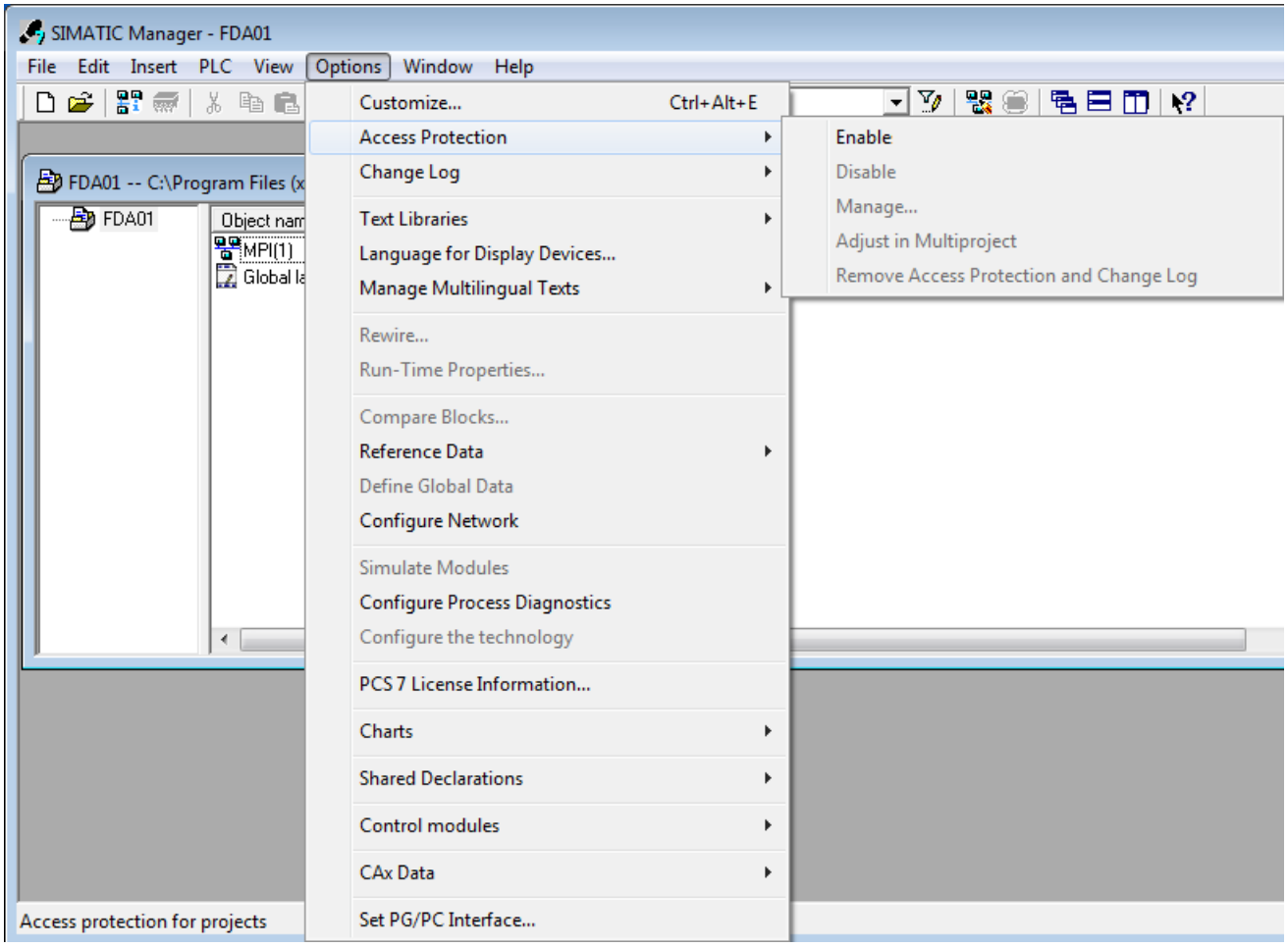


Figure 3-208 Activate SIMATIC Manager access protection

When access protection is activated, the SIMATIC Logon Service window opens. The administrator logs in here to assign a project password for this project.

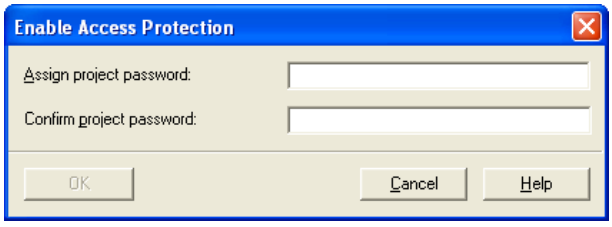


Figure 3-209 Activate access protection

Then the users are entered in the user administration for this project using drag-and-drop.

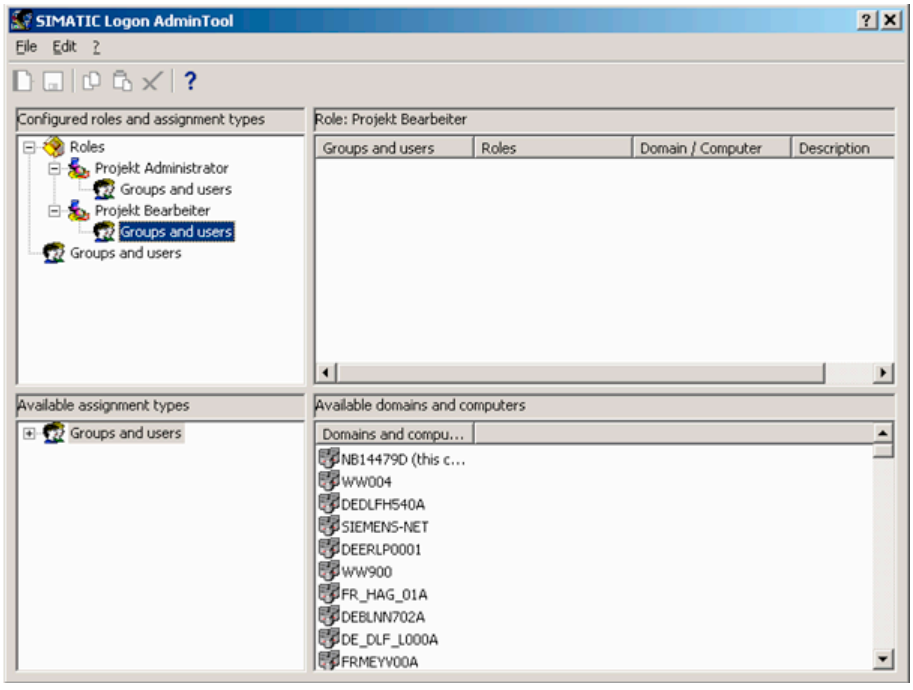


Figure 3-210 SIMATIC Logon AdminTool

Open protected project

When a protected project is opened, the following dialog appears if SIMATIC Logon is installed. Enter your user name and password here.

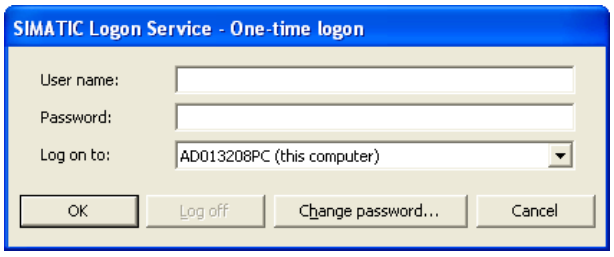


Figure 3-211 SIMATIC Logon Service

If SIMATIC Logon is not installed, the project can also be set to open only with the project password using STEP 7.

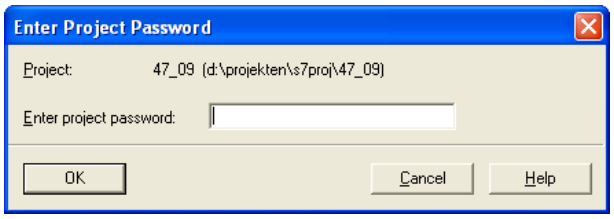


Figure 3-212 Enter the project password.

For more details on this topic, see the following document:

- SIMATIC Logon; SIMATIC Electronic Signature, Manual

SIMATIC Version Trail

Overview

SIMATIC Version Trail is a software option for SIMOTION engineering which, together with the SIMATIC Logon central user administration, can assign a version history to libraries and projects.

Installation / Prerequisites

SIMATIC Version Trail is a STEP7 option that must be installed and licensed in addition to SIMOTION SCOUT. SIMATIC Version Trail can only be used and licensed in a package together with SIMATIC Logon.

Function

When archiving, SIMATIC Version Trail creates a version history with the following information in association with SIMATIC Logon:

- Version
- Version name
- Date and time
- User
- Comment

This version history can be displayed and printed. Individual versions can be retrieved from the version history, and used further. SIMATIC Logon organizes the access protection.

View of a SIMATIC Version Trail window, showing the project name, comments and versioning data. The versions created with SIMATIC Logon can be documented and managed in a clearly organized fashion after changes.

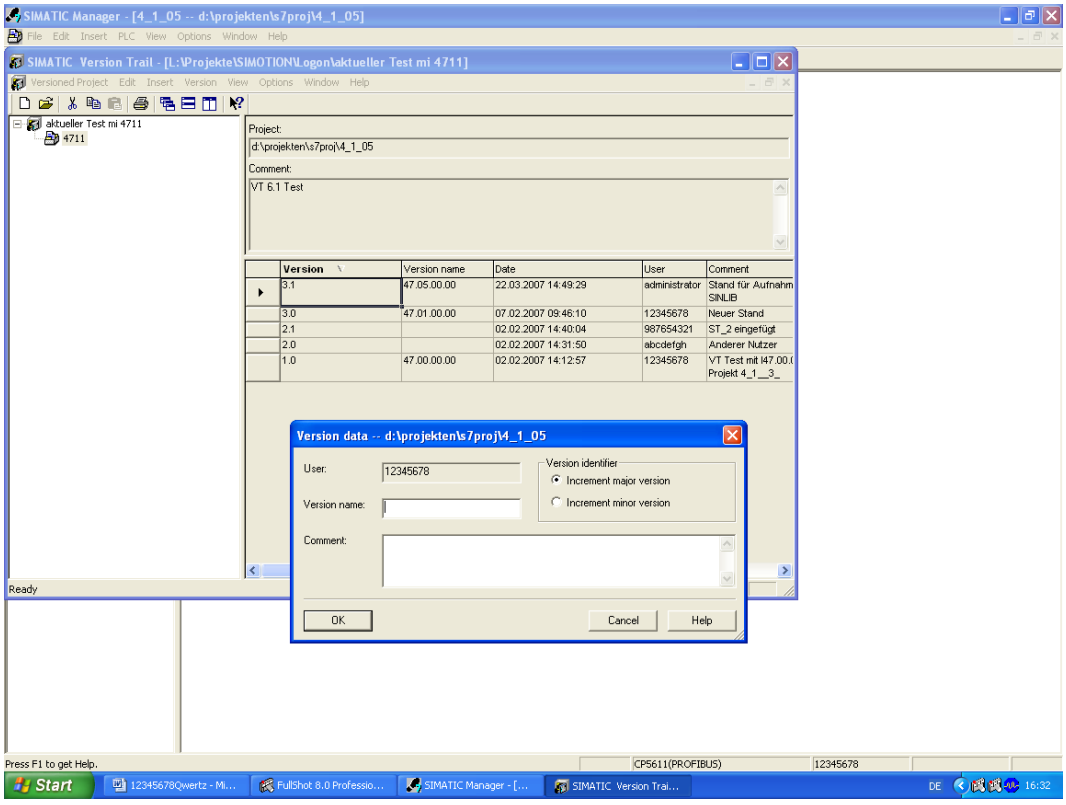


Figure 3-213 SIMATIC Version Trail - version data

Note

Please note that SIMATIC Version Trail cannot be used separately, but only in combination with SIMATIC Logon.

3.3.14.4 NetPro

NetPro in STEP 7

NetPro is integrated in the STEP 7 basic package. NetPro is a tool that supports network configuration in STEP 7.

Application for SIMOTION

So that network nodes can communicate, configure networks and/or subnets in NetPro. In NetPro, you create a graphic view of the networks and subnets and specify their properties or parameters. You can also define the node properties.

You can open NetPro in SIMOTION SCOUT using the button, whereby the NetPro application opens as a separate window. Application cases for the SIMOTION project are, for example, the connection of the engineering system via network nodes and the creation of routing connections.

Additional references

Detailed information can be found in:

- Online help, *SIMOTION SCOUT Getting Started*
- *SIMOTION SCOUT Communication* System Manual

3.3.14.5 HMI

Overview

The Human Machine Interface (HMI) connects the world of automation with the individual demands of the operator. As part of TIA, SIMATIC HMI supports uniform engineering under Windows, accesses common data and communicates uniformly.

For efficient machine operation and monitoring in various performance classes, you can use the following panels recommended for SIMOTION as well as PC-based panel systems with membrane keyboards or touch screen operation:

- Panels
  For efficient machine operation and monitoring in various performance classes, either text-based or graphics-capable with membrane keyboards or touch screen operation.
- Multi-panels
  These can be used for operator control and monitoring in the same way as the panels, with operation by means of touch screens or membrane keyboards. Multi-panels (MP) also permit the installation of additional applications and therefore provide typical PC flexibility.
- Panel PCs
  Are designed for demanding environmental conditions and provide the same degree of robustness and suitability for industry.

The WinCC flexible engineering software is available for the display of the user interface.

Additional references

Detailed information can be found in:

- Catalog: SIMOTION, SINAMICS S120 and Motors for Production Machines
- PM 21 Catalog.

SIMATIC WinCC flexible

SIMATIC WinCC flexible is the innovative HMI software under Windows 7 for all applications in the machine-related area. The engineering software enables the uniform configuration of all Windows-based SIMATIC HMI operator panels.
WinCC flexible Runtime provides basic HMI functionality on PCs, including messaging and logging systems and can be expanded when required via options. The runtime functions available on SIMATIC HMI operator panels depend on the device class.
The WinCC flexible engineering software can be integrated in the central programming software of the SIMATIC world, SIMATIC STEP 7, and used for the configuration of all operator panels. The engineering software of SIMATIC HMI also accesses variable lists and message lists of the controller and uses their communication parameters.

OPC / OPC XML-DA / OPC UA

Open communications standard for automation components. The goal of the standard is to ensure a problem-free and standardized data exchange between controllers, operator control and monitoring systems, field devices and office applications from different manufacturers.

The abbreviation OPC previously meant "OLE for Process Control", because the implementation was based on Microsoft's COM/DCOM technology.
Nowadays, one uses the term "Openness, Productivity and Collaboration". Today, a number of standards defined by the OPC Foundation have evolved. The OPC Foundation is a grouping of many manufacturers from the automation technology area.

SIMOTION supports the OPC DA (Data Access) and OPC AE (Alarm & Event) standards with the SIMATIC NET Softnet package. This is based on the Windows COM technology.

SIMOTION also supports OPC XML-DA (Data Access based on XML). The OPC server is located in the SIMOTION device and is accessed from a partner station no longer using COM mechanisms but with Web services and their XML-coded function calls. This makes the partner stations independent of hardware and operating systems.

The client application on the partner station works with the symbolic SIMOTION variable names. There is no dependency on the SIMOTION SCOUT database, which means that no consistency problems arise, even when there is a switch in versions.

OPC UA (Unified Architecture) is a standardized, industrial communication protocol for access to control data, e.g. by control systems. OPC UA Data Access permits the reading and writing of SIMOTION variables.

3.3.14.6 Drive ES

Drive ES engineering system

Drive ES is the engineering system used to integrate Siemens drive technology into the SIMATIC automation world easily, efficiently and cost-effectively in terms of communication, configuration and data management. The STEP 7 Manager user interface provides the basis.

Drive ES Basic is the basic software for the parameterization of all drives, online and offline. With the Drive ES Basic software, the automation and the drives are edited on the user interface of the SIMATIC Manager. Drive ES Basic is the starting point for common data archiving from complete projects and for extending the use of the routing and SIMATIC teleservice to drives. Drive ES Basic provides the engineering tools for the new motion control functions - data exchange broadcast, equidistance and isochronous operation with PROFIBUS DP.

The following commissioning tools are contained in Drive ES:

- STARTER Standalone for SINAMICS is not required and cannot be used in conjunction with SIMOTION SCOUT because STARTER is integrated in SIMOTION SCOUT.
- SIMOCOM U for SIMODRIVE
- Drive Monitor for MASTERDRIVES
  Note:
  Drive ES Basic is within the scope of delivery for SCOUT and SCOUT Standalone.

3.3.14.7 Commissioning drives (STARTER)

STARTER supports simple and rapid commissioning, optimization, and diagnostics for all new-generation Siemens drives with only one tool.

The STARTER drive/commissioning tool supports the following drives:

- SINAMICS
- MICROMASTER 420/430/440
- MICROMASTER 411 / COMBIMASTER 411
- COMBIMASTER
- ET200pro FC
- ET200S FC ICU24

The following variants are available:

- STARTER Standalone:
  STARTER as commissioning tool for applications without SIMOTION, but with integration of the new drives in SIMATIC S7. STARTER Standalone is not required in connection with SIMOTION SCOUT and cannot be used.
- STARTER integrated in SIMOTION SCOUT:
  For SIMOTION applications, SIMOTION SCOUT contains the entire functionality of STARTER.

Performance characteristics:

- Wizards support new users by providing solution-oriented dialog guidance, whereby the uniform graphics-based display facilitates understanding.
- However, experts are also able to access the individual parameters quickly.

3.3.14.8 CamTool

Graphic creation of cams

Simple text-based editors are already integrated in the basic SIMOTION SCOUT package for the creation of cams. The CamTool option package extends SIMOTION SCOUT with a powerful tool for the full graphical creation and optimization of cams. CamTool fully integrates into the SIMOTION SCOUT user interface. In SIMOTION CamTool, you can create, edit and optimize cams with the aid of a graphical user interface.

Additional references

Please refer to the following document on this subject

- Configuration Manual: SIMOTION CamTool

3.3.14.9 DCC programming system

Drive Control Chart

DCC enables SIMOTION and SINAMICS users to also implement and graphically configure drive-related tasks employing continuous closed-loop and open-loop control.

A set of Drive Control Blocks (DCB) is available in a library for this purpose. These function blocks can be graphically interconnected and configured in what are known as "charts" via a configuration tool (DCC editor).

A large number of Drive Control Blocks (DCB) are available for both DCC-SIMOTION and DCC-SINAMICS with identical functionality.

- Module library with administration, calculation, control, logic and complex modules.
- Graphical switching editor with various editing, macro, help, search, comparison and print functions.
- Sequence environment for SIMOTION with selectable, mixable scanning times and consistent data transfer between scanning periods.
- Sequence environment for SINAMICS with embedding of technology option in SINAMICS using the BICO technique, with the applications set using configured parameters
- Diagnostics environment with signal display, diagnostics and trace functions.
- Scalability with different performance features and project data volumes for DCC-SIMOTION and DCC-SINAMICS.

Note

Editing and deleting DCC charts

To edit DCC charts, a CFC editor must be installed and you require a valid license for DCC SINAMICS or DCC SIMOTION.

Additional references

More information on this topic is available in

- Programming and Operating Manual SINAMICS/SIMOTION Editor description DCC
- SINAMICS/SIMOTION DCC Module Description Function Manual
- SIMOTION SCOUT Online Help

3.3.15 Technical specifications

3.3.15.1 Quantity framework

Technical specifications

Note

See also the function overview in Chapter 9 of the PM 21 Catalog as well as the Industry Mall (http://www.siemens.com/automation/mall).

For quantity structures of SIMOTION devices, see:

- SIMOTION C Operating Instructions
- SIMOTION D4x5 Manual
- SIMOTION D4x5-2 Manual
- SIMOTION D410 Manual
- SIMOTION D410-2 Manual
- SIMOTION P320-3 and Panels Manual
- SIMOTION P350-3 and Panels Manual
- SIMOTION P320-4 E / P320-4 S Manual

3.3.15.2 Memory requirement

Memory requirement

Table 3-35 Memory requirement for each instance of a technology object

Technology object	Memory requirement / KB
Drive axis	180
Position axis	200
Following axis with following object	350
External encoder	150
Output cam	100
Measuring input	100
Cam In addition for each interpolation point pair	100 1
TController: Heating controller	85
TController: Cooling controller	70
TController: Heating/cooling controller	100
Position axis path interpolation	250

Technology object	Memory requirement / KB
Path object (for 4 axes, 1 cam)	200
Cam track	450
Addition object	50
Formula object	150
Fixed gear	100
Controller	100
Sensor	100

Table 3-36 Memory requirement of technology packages

Technology package	Memory requirement / KB
TP TControl	1.700
TP CAM	7.700
TP PATH	8.500
TP CAM_EXT	9.500

3.3.16 Appendix

3.3.16.1 Scripts for SIMOTION

The scripting functionality enables you to automate the configuration of devices, SIMOTION technology objects (TOs), and SINAMICS drive objects (DOs) with the help of the easy-to-learn script language VBScript. Standard scripts can be adapted to special situations occurring during runtime with interactive queries. This facilitates and speeds up commissioning. Other application possibilities include the documentation of the settings that have been made and the repetition of complex settings without error.

To access these scripts and the related documentation, select **Scripts** in **Utilities&Applications**.

The selection of documents and scripts you will find there should help you quickly get to grips with scripting in SIMOTION. As well as demonstration-only scripts for studying the code, there are also scripts and script libraries you can actually use yourself.

For further information, please also see *Scripts for sequence automation* in the online help.

3.3.16.2 Creating an example program for axis positioning in SIMOTION SCOUT

Program overview

In this section, a program for positioning the connected axis is created in the **MCC** (Motion Control Chart) editor.

The Getting Started section of the SIMOTION SCOUT online help contains a detailed description of an example configuration.

Requirements

- You need to have created a SIMOTION device
- You need to have created an *Axis 1* technology object (e.g. as a virtual axis).

Inserting the programs

Create four programs with MCC.

- motion_1
- backgr
- perfault
- tecfault

Creating a program with MCC

1. Open the **Programs** folder for the created SIMOTION device in the project navigator.
2. Double-click **Insert MCC unit**.
3. Enter the name **motion_1** for the MCC unit.
4. Click the **Compiler** tab.
5. Activate the required settings.
6. Click **OK**. An MCC unit is created in the project navigator.
7. Double-click the MCC unit motion_1 in the project navigator.
8. Double-click **Insert MCC chart**.
9. Enter the name **motion_1** for the MCC.
10. Click **OK**. The MCC is opened.
11. Click the **Single-axis commands > Switch axis enable** icon.
    The command is inserted.
12. Click the **Single-axis commands > Position axis** icon.
    The command is inserted.

13. Click the **Single-axis commands > Remove axis enable** icon. The command is inserted.

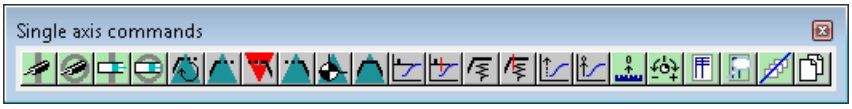


Figure 3-214 Menu bar of the single-axis commands

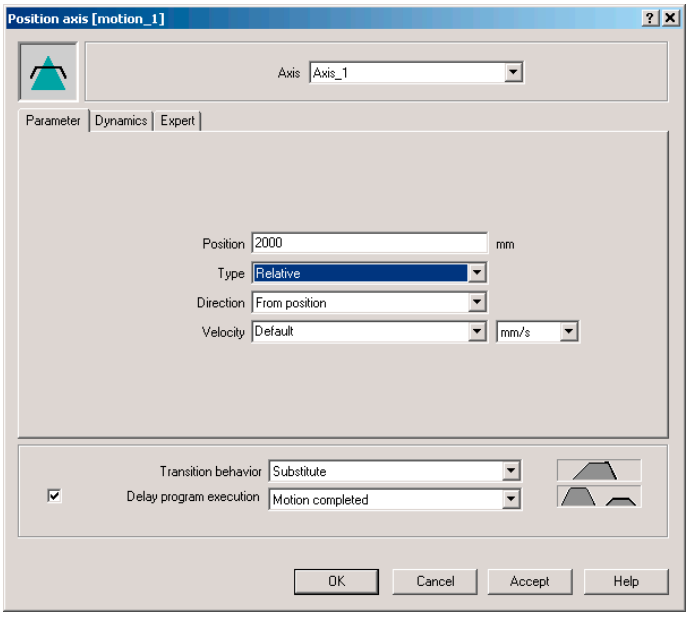


Figure 3-215 Opened command: Position axis

14. Double-click the **Switch axis enable** command.
15. Click **OK**.
16. Double-click the **Position axis** command.
17. Enter the value 2000 at Position.

18. Select **Relative** in the Type field.

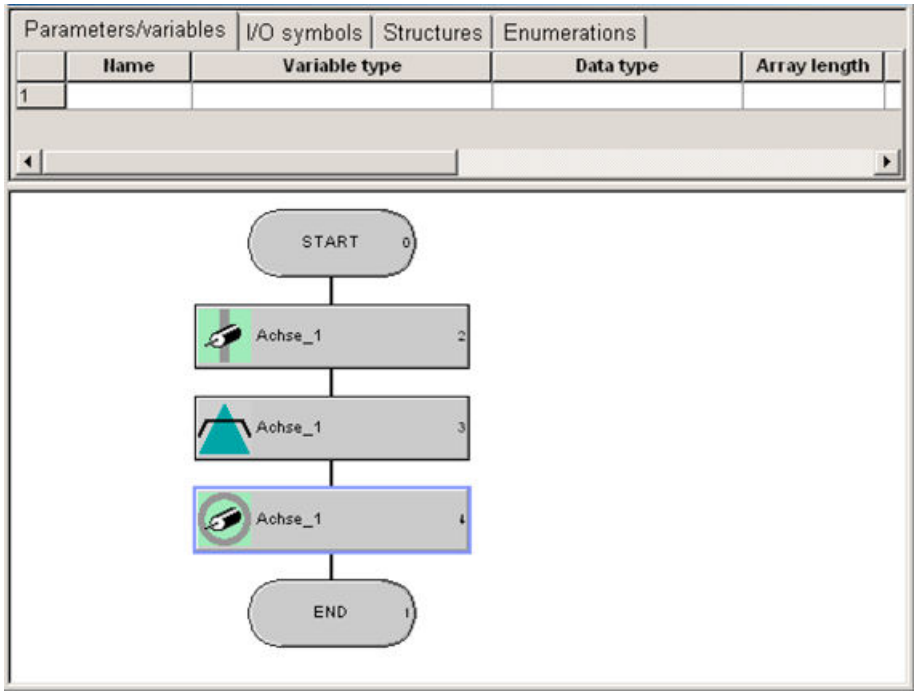


Figure 3-216 Arrangement of the single-axis commands in MCC motion_1

19. Click **OK**.
20. Double-click the **Remove axis enable** command.
21. Click **OK**.
22. Compile using the menu command **MCC chart > Apply and compile**.

Creating the backgr program

1. Double-click **Insert MCC unit**.
2. Enter the name **backgr** for the MCC unit.
3. Click the **Compiler** tab.
4. Activate the required settings.
5. Click **OK**. An MCC unit is created in the project navigator.
6. Double-click the MCC unit **backgr** in the project navigator.
7. Double-click **Insert MCC chart**.
8. Enter the name **backgr** for the MCC.
9. Click **OK**. The MCC is opened.

10. Enter the following variable data in the MCC unit **backgr**.
    - In the Name field: Run
    - As the variable type, select: VAR_GLOBAL
    - As the data type, select: BOOL
    - In the Initial value field: False

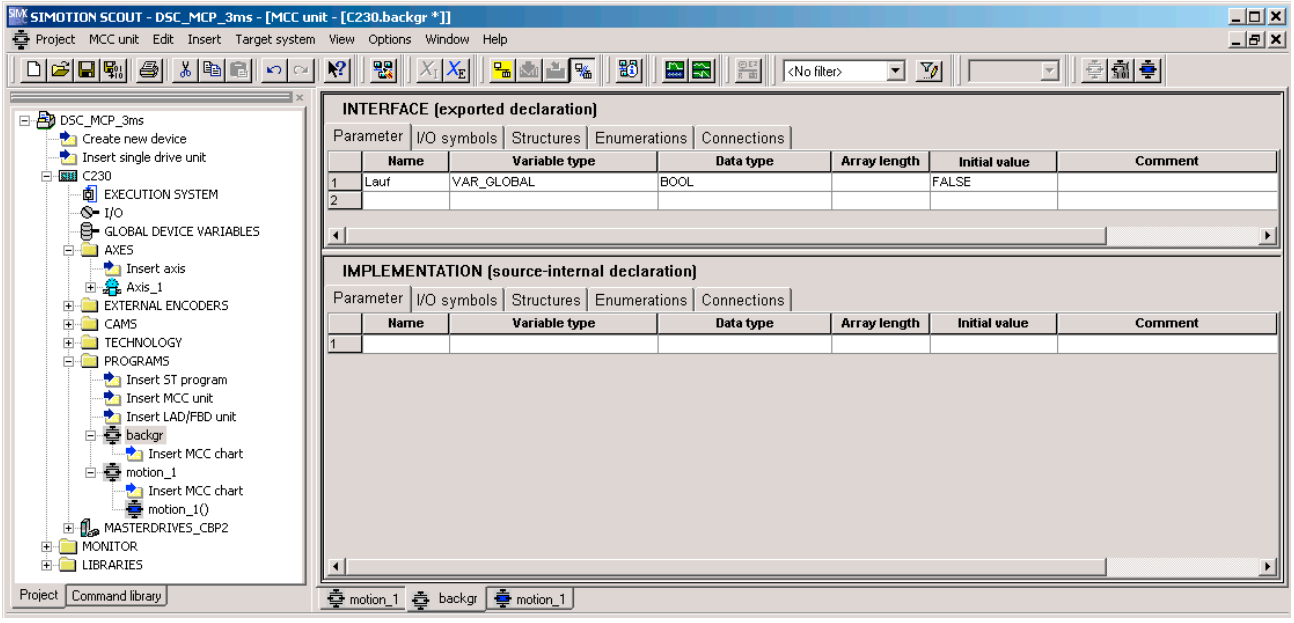


Figure 3-217 Entering a global variable

11. Switch to the MCC Chart **backgr** tab.

12. In the **Program structures** command bar, click on the **IF program branch** icon.

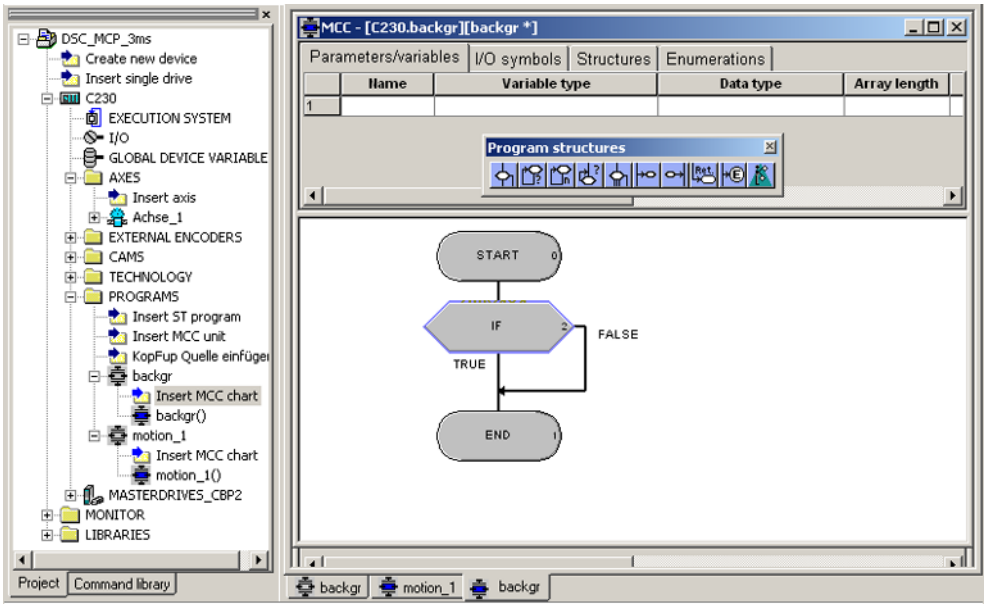


Figure 3-218 IF program branch at basic commands

13. Double-click the **IF program branch** command.
14. Select **Formula** and enter the condition **Run=true**.

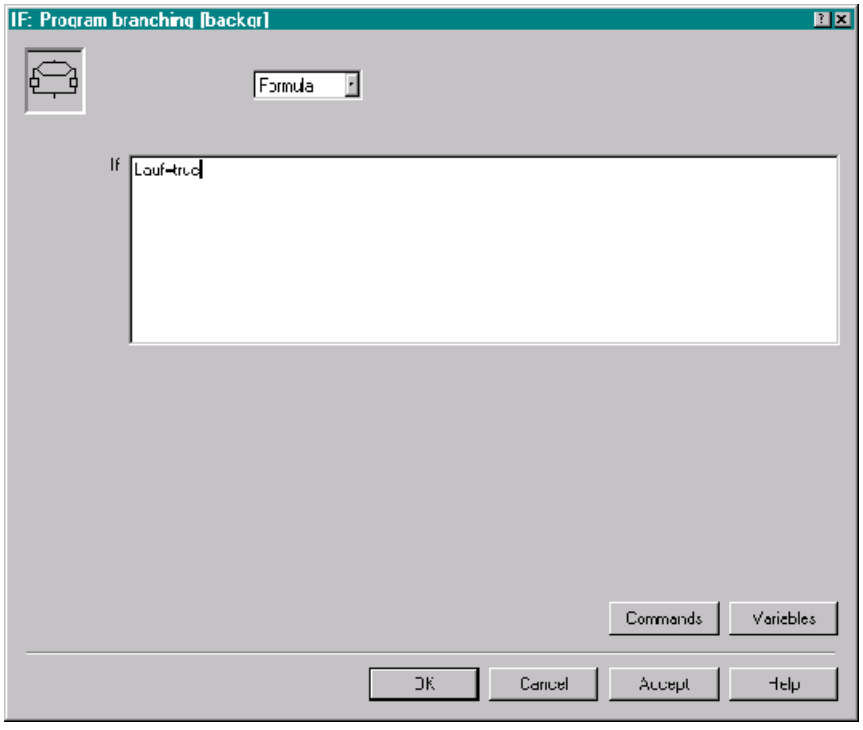


Figure 3-219 Configuration of the IF program branch

15. Confirm the input with **OK**.
16. In the **Task commands** command bar, select the command **Start task**.
    This inserts the command.
17. In the **Important commands** command bar, select the command **Variable assignment**.
    This inserts the command.
18. Double-click the **Start task** command.
19. Click **OK** to confirm.
20. Double-click the **Variable assignment** command.

21. Enter the statement **Run:=false**.

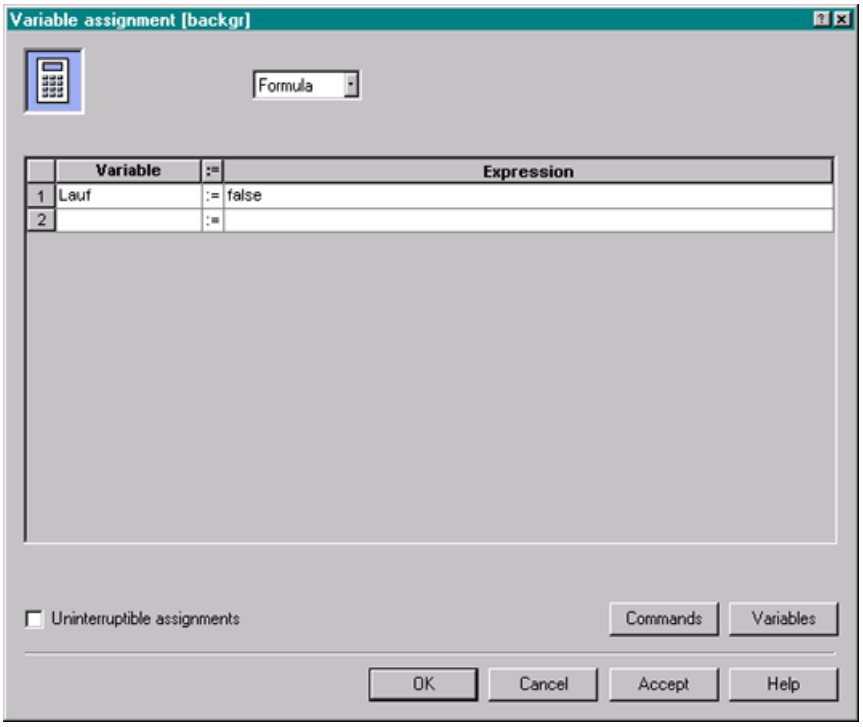


Figure 3-220 Configuration of the variable assignment

22. Click **OK** to confirm.

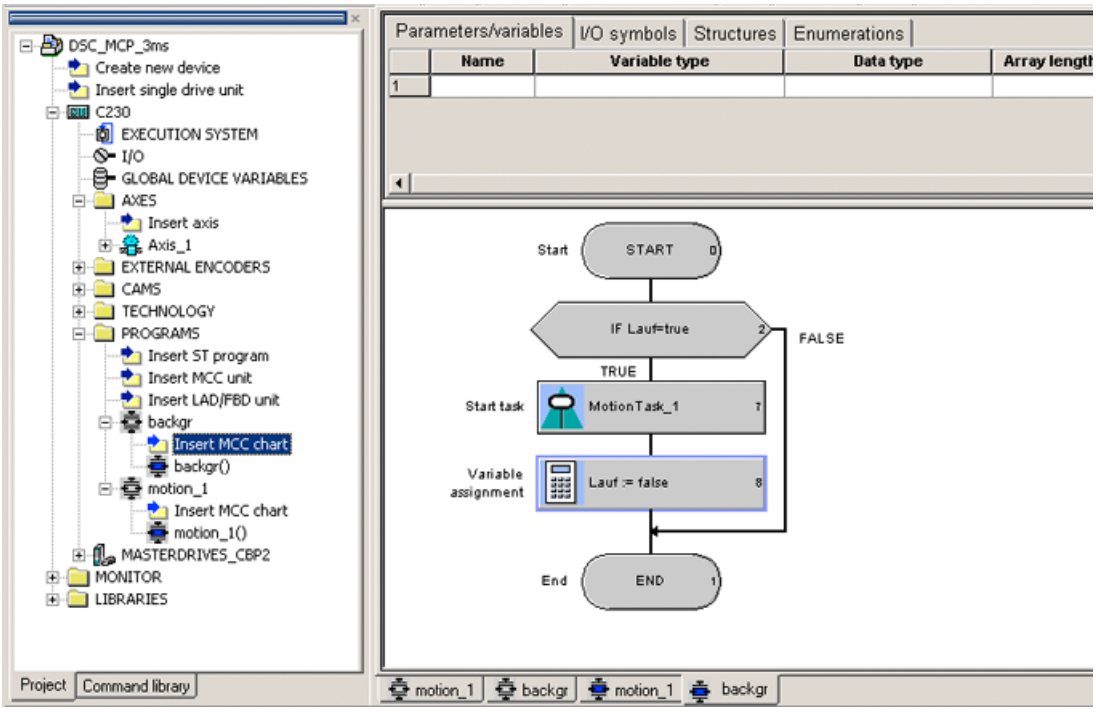


Figure 3-221 Completed program of the BackgroundTask

23. Select **MCC Chart > Accept and compile** from the menu bar.
    The detail display in the Compile/Check output tab indicates how compilation is progressing and when it is complete.

Creating the perfault program

1. Double-click **Insert MCC unit**.
2. Enter the name **perfault** for the MCC unit.
3. Click the **Compiler** tab.
4. Activate the required settings.
5. Click **OK**. An MCC unit is created in the project navigator.
6. Double-click the MCC unit perfault in the project navigator.
7. Double-click **Insert MCC chart**.
8. Enter the name **perfault** for the MCC.
9. Click **OK**. The MCC is opened.
10. Select from the menu bar **MCC chart > Apply and compile**.

Creating the tecfault program

The procedure for creating the **tecfault** program is exactly the same as for the perfault program.

In the MCC unit and MCC chart, enter **tecfault** as the name.

Assigning programs to the execution system

1. In the project navigator, double-click **Execution system** for the SIMOTION device. The execution system is opened in the working area.

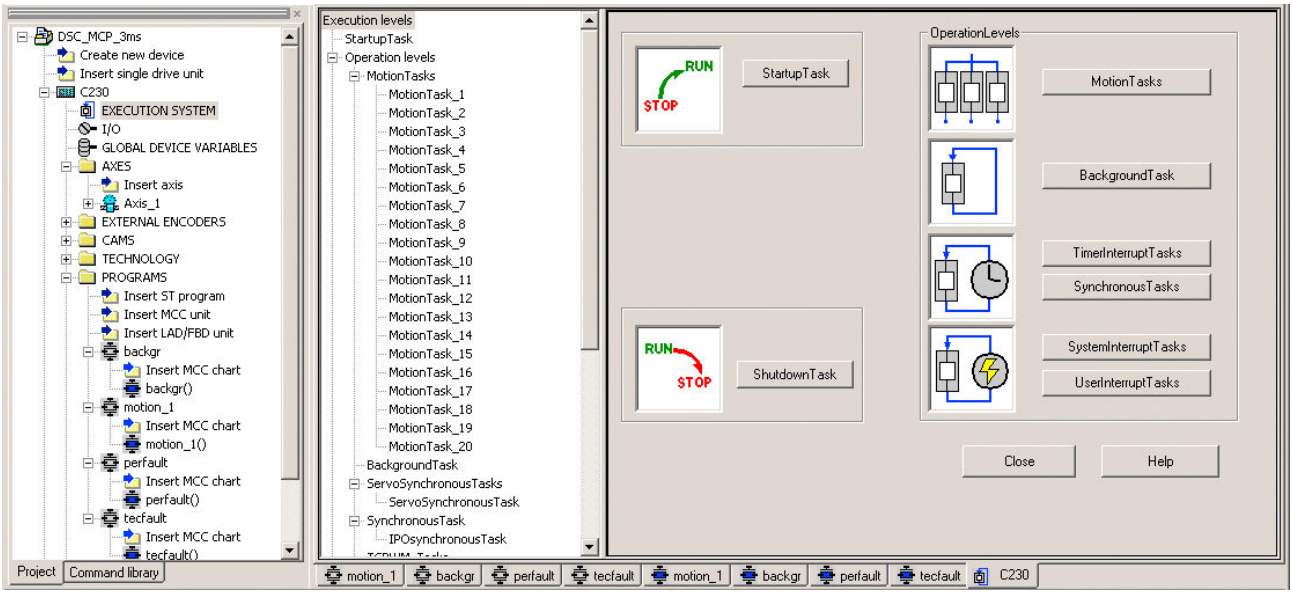


Figure 3-222 Execution system of the SIMOTION device

2. Click the **MotionTasks** button.
3. Assign the **motion_1** program to MotionTask_1.
4. Assign the **backgr** program to the BackgroundTask.
5. Assign the **perfault** program to the PeripheralFaultTask.

6. Assign the **tecfault** program to the TechnologicalFaultTask.

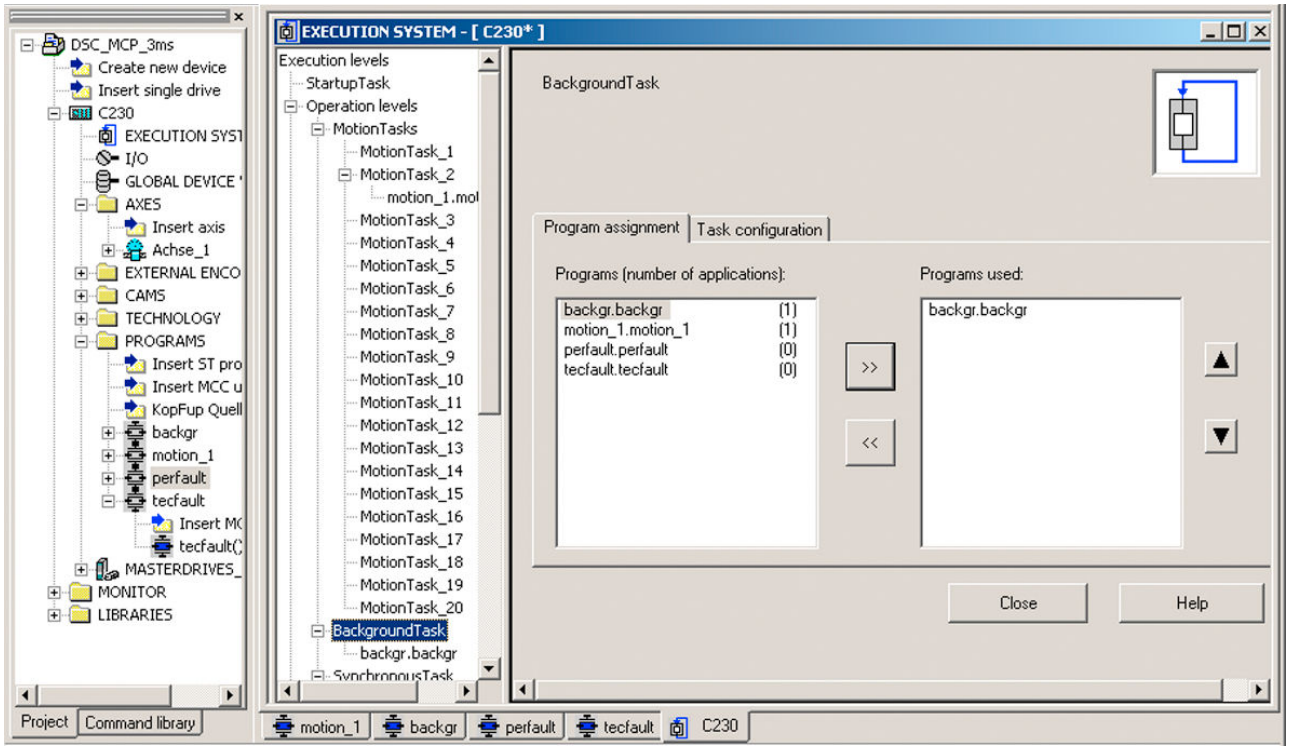


Figure 3-223 Assignment of the tasks in the task system

7. Click the **Save and compile changes** button.

Loading the programs to the SIMOTION device and switching to RUN

The toggle switch of the SIMOTION device is still set to **STOP**.

1. Establish an online connection. To do this, click the **Connect to selected target devices** button.
2. Click the **Download project to target system** button.
   This can take several minutes.
3. Switch the SIMOTION device to **RUN** via the **switch** after the project has been downloaded to the target system. Two LEDs, the green 5 VDC and the RUN LED, light up.

Monitoring and controlling in SIMOTION SCOUT

1. Click the **Connect to selected target devices** button.
2. Select the **backgr** program.
3. Activate the **Monitor** function in the MCC toolbar.
4. Select the **Symbol browser** tab.
5. As **control value**, enter the variable **Run** "**TRUE**".
6. Activate the setting in Status value.

7. Click **Start status**. The value **FALSE** is displayed.

8. Activate the setting in Control value.
9. Click **Control**. The active command is highlighted in yellow in the program.

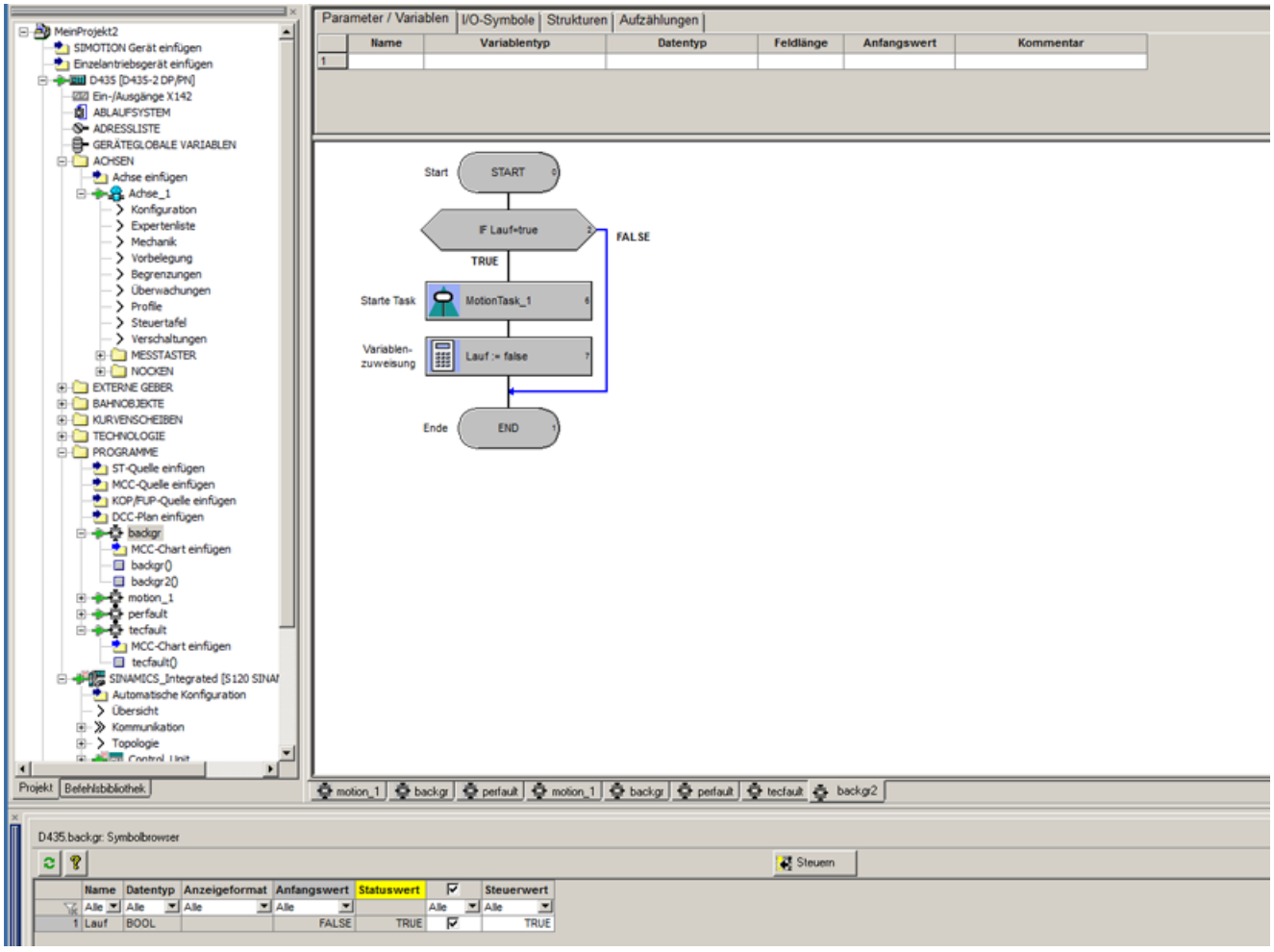


Figure 3-224 Monitoring and controlling the variable in the BackgroundTask

The motor now rotates at 6000 rpm and approaches the position 2000 mm. You can check this by selecting **Axis_1** at **Axes**, opening **basicMotion** in the symbol browser, and observing the **position**. The values are refreshed automatically when an online connection exists.

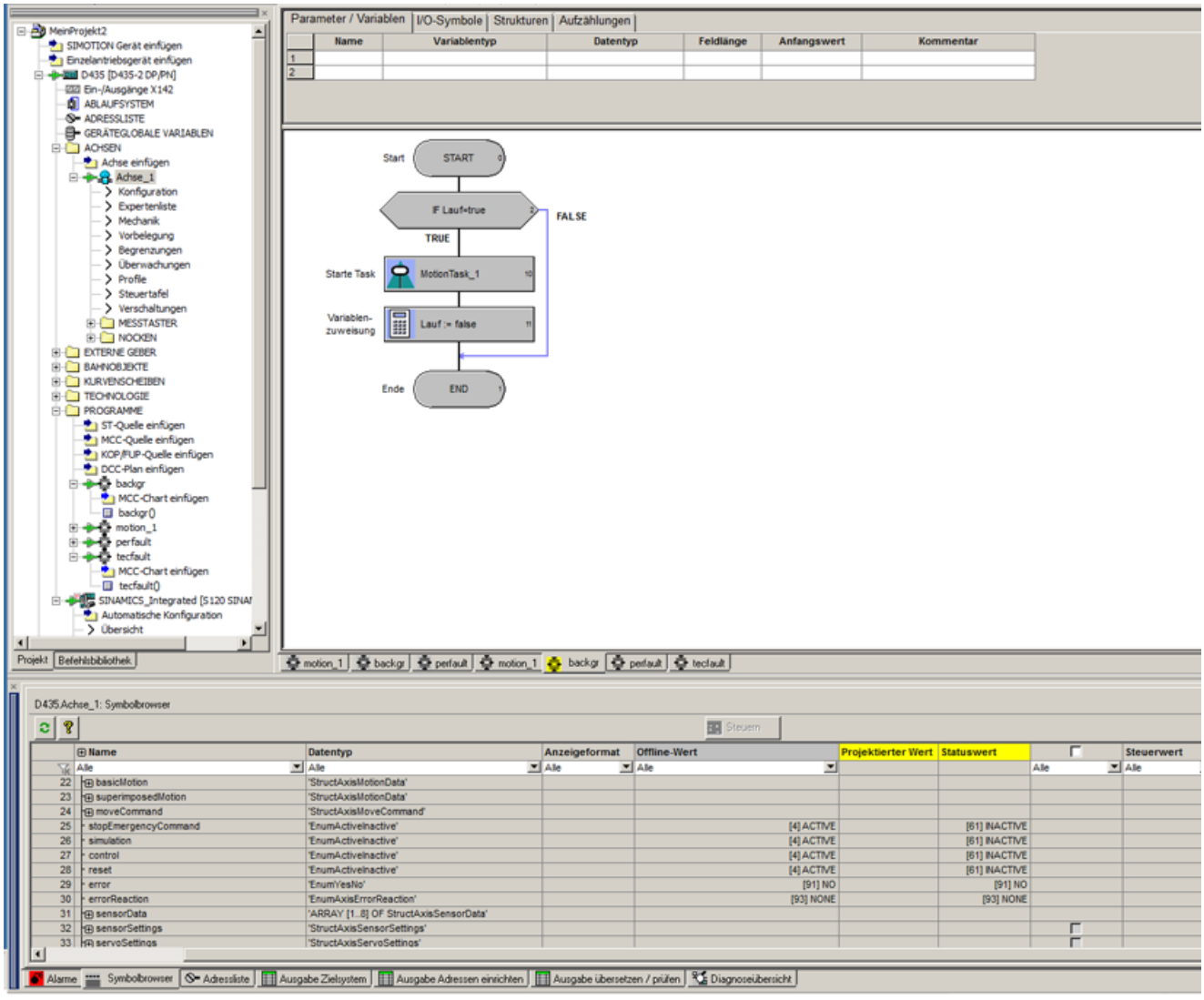


Figure 3-225 Checking the positioning motion in the symbol browser of Axis_1

When the axis has finished positioning, a value of 2000 mm appears here. If the positioning is restarted, the actual position value increases by 2000 mm for each motion.

3.4 Getting Started with SIMOTION SCOUT TIA

Preface

Scope and standards

This **document** is part of the **Engineering System Handling documentation package**.

Scope of validity

This manual is valid for SIMOTION SCOUT, product version V5.1.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT TIA and it comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.1:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/de/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT TIA, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

3.4.1 Fundamental safety instructions

3.4.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

3.4.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity..

⚠ WARNING

Danger to life as a result of unsafe operating states resulting from software manipulation

Manipulation of the software, e.g. viruses, trojans, malware or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
- Make sure that you include all installed products in the integrated industrial security concept.
- Protect files on removable storage media against malware through appropriate protective measures, e.g. virus scanners.

3.4.1.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

3.4.2 Getting Started with SIMOTION SCOUT TIA

3.4.2.1 Aim of Getting Started

Getting Started introduces you to working with the SIMOTION SCOUT TIA engineering system. You will create a simple sample project and in so doing, you will work through the typical steps involved in configuring devices, drives, and axes. You will become familiar with the most important tools that SIMOTION SCOUT TIA provides for configuring, programming, and diagnostics.

3.4.2.2 Sample project

Getting Started provides you with instructions for creating a simple sample project.

Configuring steps

Prepare the configuration

- You reset the SIMOTION device to the factory settings.
- You configure the interface for network communication between the PG/PC and the SIMOTION device.

Create project, configure SIMOTION device and network communication with the PG/PC

- You create a project.
- You create a SIMOTION device and set up the network communication between the PG/PC and the SIMOTION device .

Configure the drive

- You commission the drive.

Configure the infeed

- You interconnect the infeed with the drive.

Configure and test the axis

- You set up an axis.
- You interconnect the axis with the drive.
- You test the axis with the axis control panel.

Configure inputs/outputs

- You configure I/Os for use in the sample program.

Program, set up and monitor SIMOTION

- You write a simple SIMOTION user program that controls the configured axis.
  - You create the variables required by the program.
  - You create the program and additional auxiliary programs with the graphical editors.
- You assign the finished programs to the tasks of the execution system.
- You start program execution in the SIMOTION runtime system.
- You monitor the program-controlled axis movement.
  - You monitor program execution.
  - You monitor the values in the symbol browser.
  - You compile values in a watch table.
  - You record the course of the axis motion with the trace.

3.4.2.3 Requirements

Getting Started provides you with instructions for creating a simple sample project.

Trainings system

To create the sample project, you require a training system with a few components:

- SIMOTION D4x5-2 device with the latest firmware
- PG/PC with free Ethernet interface
  A USB Ethernet adapter is also suitable for the Ethernet connection.

  Note

  A USB PROFIBUS adapter, e.g. PC Adapter USB A2 (article number: 6GK1571-0BA00-0AA0) can be used for the PROFIBUS connection. Performance problems may occur if a previous version is used.

- Full DRIVE-CLiQ wiring of the components; motor with DRIVE-CLiQ interface and thus with automatic encoder identification (SMI Sensor Module Integrated)
- TIA Portal and SIMOTION SCOUT TIA engineering system

A SIMOTION D435-2 DP/PN device is used in the sample project.

The PROFINET interface of the SIMOTION device is not used in the sample project.

Note

Getting Started deals with automatic drive configuration and not configuration using the drive wizard. To be able to carry out automatic drive configuration fully, full DRIVE-CLiQ wiring of the components involved is a necessary requirement.

Preparation of the training system

Your training system has been prepared for configuring with TIA Portal and SIMOTION SCOUT TIA:

- The hardware is ready installed and wired.
- The CF card with the latest firmware is inserted.
- The PG/PC and SIMOTION D435-2 are connected direct via Ethernet cable. The Ethernet X127 interface on the SIMOTION D435-2 is used for the connection.
- TIA Portal (as of V14) is installed on the PG/PC.
- SIMOTION SCOUT TIA (as of V4.5) is installed on the PG/PC and licensed correctly.
- You have started the TIA Portal. The portal view is visible on the screen of the PG/PC.

Overview of configuring steps

The table below provides an overview of the individual configuring steps and where you implement them.

Configuring step	Where	Chapter
① Create project	TIA Portal	see Creating project (Page 471)
② Create the SIMOTION device and configure online communication (incl. compilation of the hardware configuration and downloading to the SIMOTION device)	TIA Portal	see Creating the SIMOTION device and configuring online communication (Page 472)
③ Start SIMOTION SCOUT TIA	TIA Portal	see Starting SIMOTION SCOUT TIA (Page 486)
④ Download the project to the target system (incl. compilation)	SIMOTION SCOUT TIA	see Downloading the project to the target system (Page 487)
⑤ Configure the drive	SIMOTION SCOUT TIA	see Configuring the drive (Page 494)
⑥ Configure the infeed	SIMOTION SCOUT TIA	see Configuring the infeed (Page 499)
⑦ Configuration of technology objects (in the TO Axis example)	SIMOTION SCOUT TIA	see Configuring axis (Page 501) / Testing the axis using the axis control panel (Page 507)
⑧ Configure digital outputs	SIMOTION SCOUT TIA	see Configuring digital outputs (Page 511)
⑨ Creation of user programs	SIMOTION SCOUT TIA	see Programming the SIMOTION application (Page 512)
⑩ Configuring the execution system	SIMOTION SCOUT TIA	see Configuring the execution system (Page 552)
⑪ Starting and stopping the system	SIMOTION SCOUT TIA	see Starting and stopping the system (Page 557)
⑫ Monitor the application	SIMOTION SCOUT TIA	see Monitoring the application (Page 560)

3.4.2.4 General information

TIA Portal online help

The TIA Portal has an extensive online help system.

How to open the online help

- Select **Help > Show help** in the menu, or
- press the **F1** key.

After calling the online help, you will be in the information system. What help is available to you and how you can operate the help in the TIA Portal is explained in **Further support > Help on the information system**.

SIMOTION SCOUT TIA Online Help

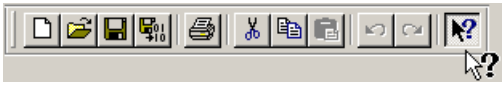
SIMOTION SCOUT TIA has a comprehensive online help. You can optionally search for specific topics in the online help or call the help from the context of your work.

How to open the online help

- Select **Help > Help topics** in the menu, or
- press the **F1** key.

How to open the context-sensitive online help

- Click the Help button of the dialog or window, or
- press the key combination **Shift+F1**, or
- click in the toolbar on the **Help** button.



With the mouse pointer (changed to a question mark), click the parameter or the window for which you require help.

You can find detailed information on using the context-sensitive help in the online help under **SIMOTION SCOUT TIA > General > Use Help**.

Full text search in the online help

You can find important information on full text searching in the online help under **Basic > Use Online Help and Function Block Diagrams > Use Online Help > Full Text Search**.

Note

Full text search in the online help

Enclose the search term between asterisks * to include search results that contain the search term as part of a longer character string, e.g.

- **variable** finds only the whole words "variable" or "Variable",
- ***variable*** also finds "system variables", "variable assignment", etc.

Available documentation

Electronic documentation

The documentation is included in electronic form in the scope of delivery of SIMOTION SCOUT TIA (SIMOTION SCOUT DVD Documentation, Utilities & Applications). You can search for all PDF documents in the electronic documentation with an index (SIMOTION.pdx). You can find an overview of the structure and content of the SIMOTION PDF documentation in the separate document *Overview of the SIMOTION Documentation*.

The content of the documents is also available in the SIMOTION SCOUT TIA online help, with a few exceptions.

In the online help, you can find the *Overview of the SIMOTION Documentation* under **Overview of the SIMOTION Documentation > ... > Overview of the SIMOTION Documentation**.

Note

You will find the *SIMOTION SCOUT TIA Configuration Manual* as a PDF document on the SIMOTION SCOUT DVD.

Overview of SIMOTION

You can find a brief system overview of SIMOTION in the online help under **Basic > Basic Functions > System Overview**.

The manuals contain comprehensive information especially for configuring and commissioning a SIMOTION D, see the *SIMOTION D4x5-2 Manual* as well as the *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual*. You can also find the SIMOTION D manuals in the online help under **SIMOTION Devices > SIMOTION D**.

Additional information

You will find additional information and documents on SIMOTION in the Industry Online Support. Here, you are provided with an overview of the most important technical information and solutions for SIMOTION.

http://support.automation.siemens.com/WW/view/de/10805436/130000

You can also access the Online Support via the hardware catalog of the TIA Portal (select the relevant modules and open the shortcut menu).

Utilities & applications

The free SIMOTION Utilities & Applications provide you with a wealth of important background information on all aspects of SIMOTION, tools, special functions, blocks, SIMOTION sample projects, as well as off-the-shelf standard applications for illustration or for use in your projects. There you can also find detailed information on scripting and a host of sample scripts that facilitate working with SIMOTION.

1. **FAQs**
   Interesting FAQs such as control of hydraulic axes or communication issues.
2. **Scripts**
   Many scripts, helpful tips, but also comprehensive solutions that make recurring tasks easier, for example.

   Note

   As part of SIMOTION SCOUT TIA (TIA Portal), only scripts can be used that influence or use pure SIMOTION SCOUT/SIMOTION SCOUT TIA data and functionalities.

   Scripting of the data/functionalities of the HWCN and the project handling of the TIA Portals (framework) is not possible.

   The following is explicitly impossible by scripting:
   - Creating, deleting and renaming of objects that have a representation in the TIA Portal (such as all SIMOTION devices).
   - Creating, deleting and renaming of projects.
   - File system accesses to TIA Portal projects.
   - Use functionality provided by the TIA Portal (framework), e.g.
   - Archiving/dearchiving.
   - All handling of TIA Portal data.

   The SIMOTION easyProject project generator cannot be used for projects with SIMOTION SCOUT TIA, because the scripting functionality is not yet available in the TIA Portal.

   Note also that external scripting of SIMOTION SCOUT TIA is not supported.
3. **Tools and documentation**
   You are provided with easy-to-use tools and in-depth documentation for many tasks.
4. **Examples**
   Sample projects for first-time-users, e.g. "Getting Started", as well as examples of special topics.

5. **Applications**
   For SIMOTION, there is a host of applications available to you that provide you with a sound basic framework. With the aid of the supplied documentation, you can use the applications as a basis for your own application, and adapt and expand them.
   Furthermore, functions are available to you under "Cross-Sector applications" that are of general help when creating your own applications, e.g. a LDPV1 library for drive communication via DPV1 services (read and write SINAMICS parameters, read errors and warnings from SINAMICS, deactivate objects, ramp-up coordination and much more), or the LCom library with functions for communication with TCP/IP for SIMOTION and SIMATIC.
6. **SIMOTION IT**
   Under SIMOTION IT, you can find innovative examples and tools for bringing control and drive solutions a little closer to the IT world. These include a trace function that is executed via an Internet browser, as well as examples for user-defined Web pages for the SIMOTION IT Web server or for using OPC XML DA.

Specific safety information

Make sure your training system is fully disconnected from productive operation.

Observe the safety notes in the documentation of the devices used.

3.4.3 Prepare the configuration

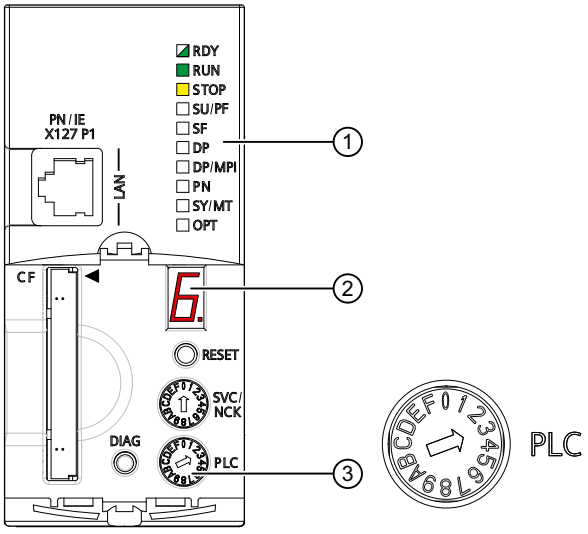
3.4.3.1 Restore factory settings

For the sample project, it is useful to reset the SIMOTION device to the factory settings.

In this way, you restore the default communications parameters and you delete user data installed on the device and the CF card by a previous configuration. The runtime licenses are retained.

You restore the factory settings on the SIMOTION device as follows

1. Switch off the power supply of the SIMOTION device.
2. Set the mode selector switch ③ of the SIMOTION device to position **3**.



① LED display
② 7-segment display
③ Mode selector switch in position 3

Figure 3-226 SIMOTION D435-2 module front

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Observe the ESD regulations.

3. Switch on the power supply of the SIMOTION device.
   The default settings are loaded. SIMOTION D435-2 switches to STOP mode.
   Wait for the procedure to finish. The elements on the front of the module indicate completion:
   - The 7-segment display ② shows status digit **6**: SIMOTION D435-2 has started up.
   - LEDs ①:
     LED **RDY** flashes green (0.5 Hz): the drive is ready for commissioning.
     LED **STOP** shows a yellow light: the SIMOTION device is in STOP mode.
     All other LEDs are off.
4. Turn the mode selector switch ③ to position **0**.
   - The LED **RUN** shows a green light: the SIMOTION device is in RUN mode.

Result

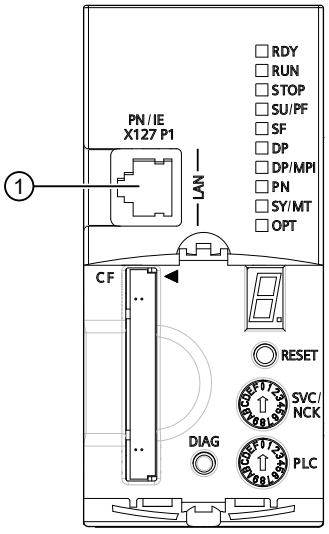
The SIMOTION device has been restored to its factory settings and is ready for commissioning.

3.4.3.2 Requirements for online communication

Online communication of the PG/PC with the SIMOTION device can be set up via PROFIBUS, PROFINET, or Industrial Ethernet. The sample project is restricted to the most frequent application case: communication via Industrial Ethernet.

Requirements

- The PG/PC and the SIMOTION device are connected via an Ethernet cable.
- The X127 PN/IE interface on the SIMOTION device is used for the Ethernet connection.



① Ethernet interface X127 PN/IE

Figure 3-227 SIMOTION D435-2 module front

- The Ethernet interface X127 PN/IE has the default address:
  IP address: **169.254.11.22** - Subnet: **255.255.0.0**

See also

Configure Ethernet interface (2) (Page 475)

3.4.3.3 Result in the sample project

The factory settings of the SIMOTION device have been restored. The requirements for online communication of the PG/PC with the SIMOTION device are fulfilled.

3.4.4 Create a project

3.4.4.1 Overview

Aim of Getting Started

In this part of Getting Started, you create the sample project Sample_1 in the TIA Portal. All of the subsequent configuring steps refer to this sample project.

Project

A project contains all the information that describes a machine and its function: configuration data, programs, motion profiles, drive data.

A project can contain several SIMOTION devices.

3.4.4.2 Create new project

After opening the TIA Portal, you are in the portal view. The portal view offers a task-oriented view of the tools, and provides the basic functions for the individual task areas.

You create a new project as follows

You start a new configuration by creating a new project in the TIA Portal.

1. Select **Start > Create new project** in the navigation of the portal view. The **Create new project** dialog appears.

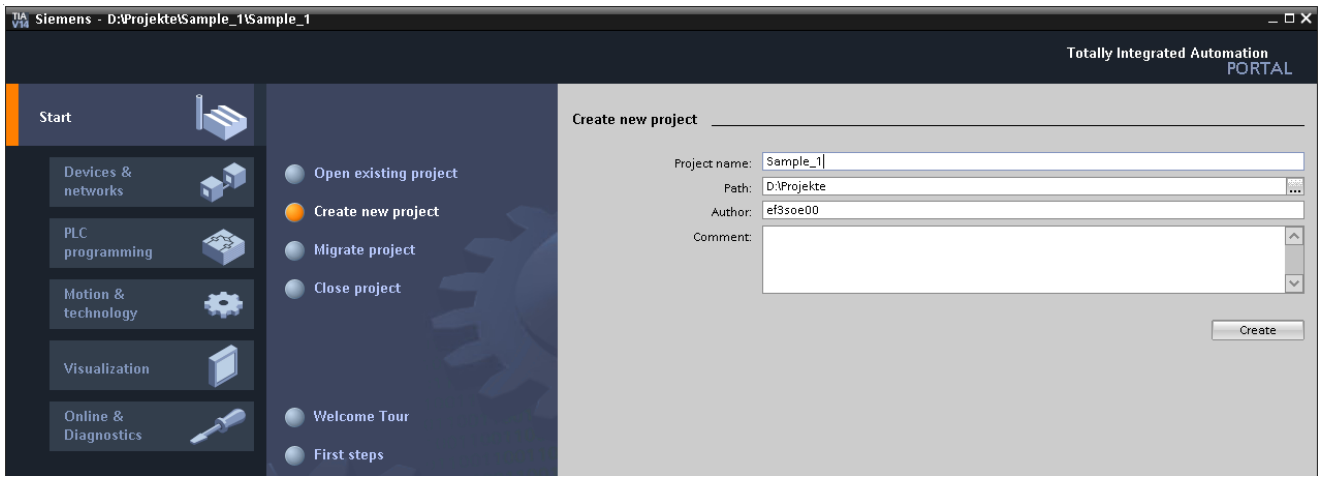


Figure 3-228 Create a project

2. Enter the project name under **Project name**, e.g. **Sample_1**.
3. Under **Path**, enter the path in which the project is to be stored. The default path is already set.
4. Confirm with **Create**.

3.4.4.3 Result in the sample project

The sample project of Getting Started has been created in the TIA Portal.

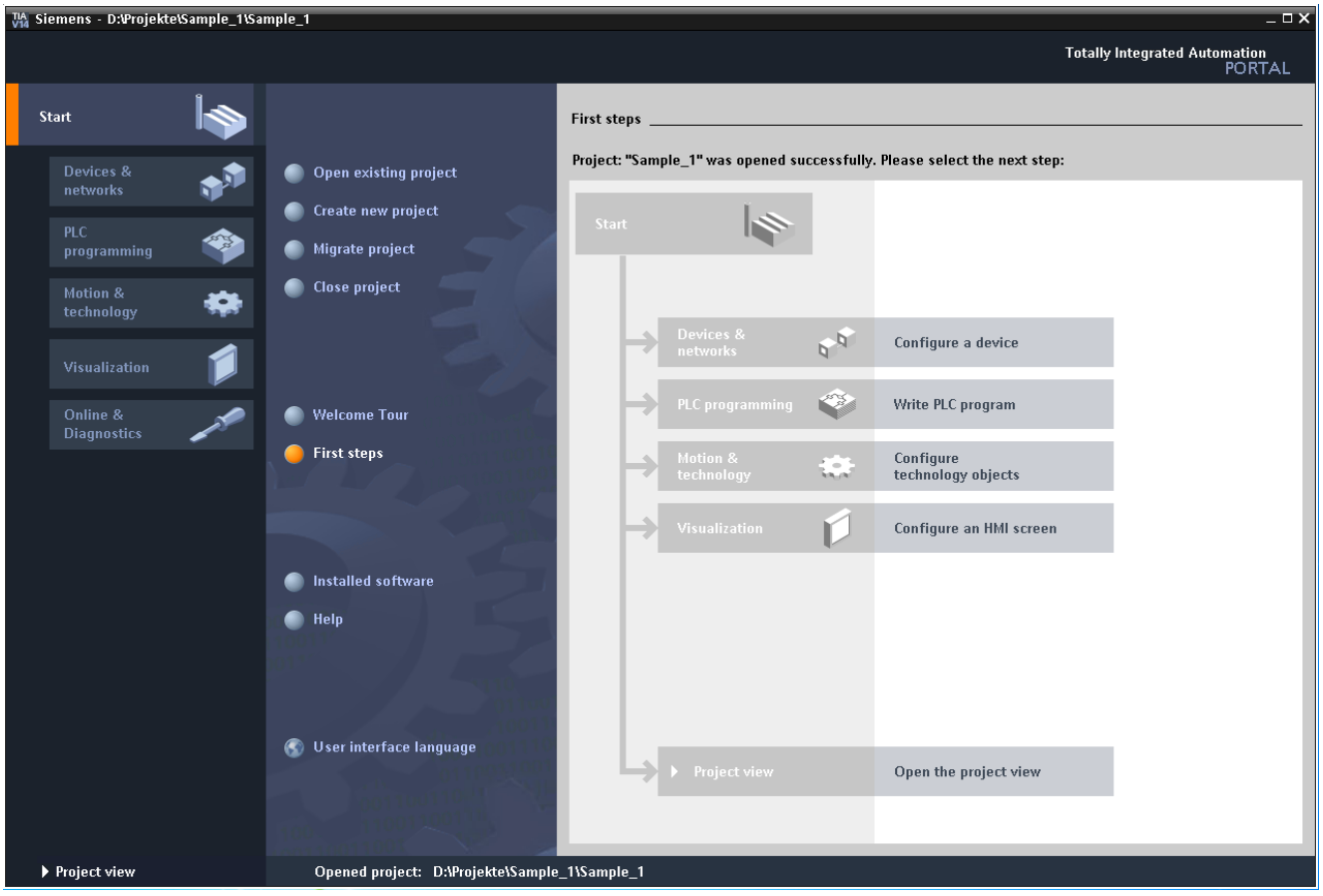


Figure 3-229 Project created

3.4.5 Create SIMOTION device and configure online communication

3.4.5.1 Overview

Aim of Getting Started

This part of Getting Started shows you how to create a SIMOTION device in the project in the TIA Portal and how to set up communication between the PG/PC and the SIMOTION device.

Steps

The following steps are executed in order to create a SIMOTION device in the project:

1. Create a SIMOTION device.
2. Configure the Ethernet interface.
3. Set up communication between PG/PC and SIMOTION device, compile the project, and download to the device.

SIMOTION D platform

SIMOTION D is the drive-based version of the SIMOTION motion control system, based on the SINAMICS S120 family of drives. With SIMOTION D, the SIMOTION motion control functionalities and the SINAMICS drive software run on a SINAMICS-type closed-loop control hardware device. SIMOTION D devices have the following characteristic features:

- Motion control functionality and control functionality integrated directly in the drive
- Compact and with especially fast response
- Maximum scalability and flexibility as single-axis or multi-axis system in different performance versions for diverse applications
- Especially suitable for modular machine concepts with fast isochronous coupling, in the case of machines with a large number of axes, for example

The SINAMICS functionality of the closed-loop-control module of the SINAMICS S120 multi-axis drive system is integrated in SIMOTION D (SINAMICS Integrated).

You will find a brief introduction to SIMOTION D in the online help using the example of the D435 Control Unit. Click **Help > Tutorials > SIMOTION Drive-Based** in the SIMOTION SCOUT menu bar.

3.4.5.2 Create SIMOTION device (1)

You create a SIMOTION controller in the project as follows

1. Select **Start > Devices & Networks > Add new device** in the navigation of the portal view. The dialog **Add new device** appears.
2. Under **Controller**, open the desired device variant (e. g.: SIMOTION D - Drivebased).
3. Select the desired SIMOTION device **D435-2 DP/PN**.
4. Click the Article No. to display its device-specific information.

5. Under **Version**, select the firmware version of the installed SIMOTION device.

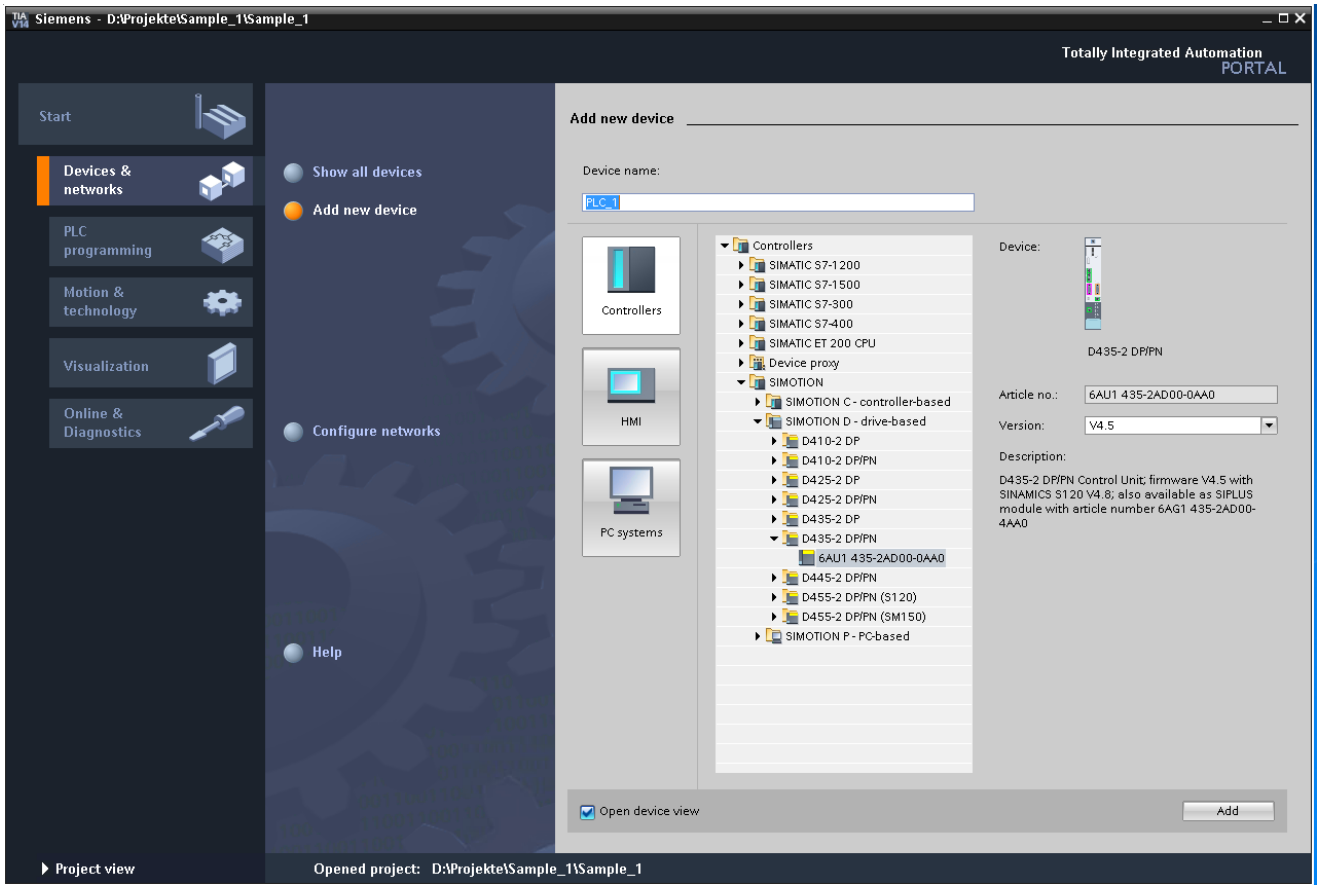


Figure 3-230 Select SIMOTION D435-2 DP/PN

6. Click **OK** to apply the settings and create the SIMOTION device.

Result

The SIMOTION device is created in the project. After you have created the device, you will be in the project view on the Device View tab.

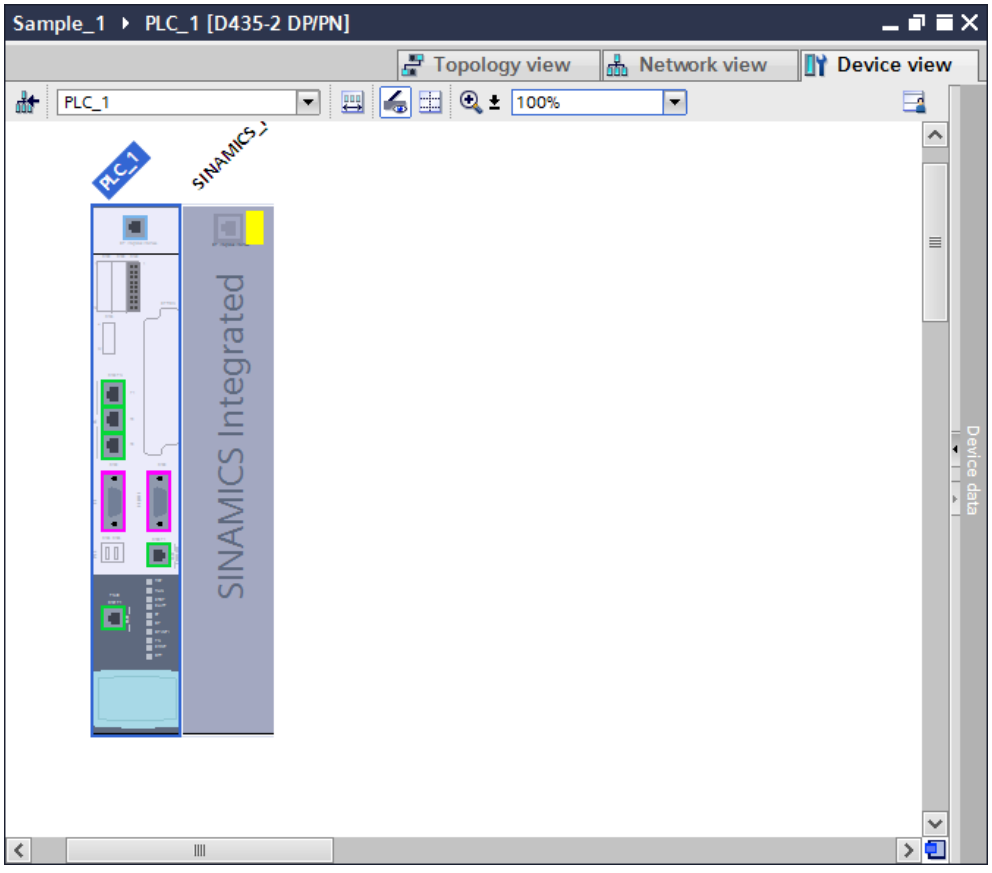


Figure 3-231 SIMOTION **D435-2 DP/PN** in the device view

Note

The selected firmware version must match the firmware version on the memory card of the device. Otherwise an error message will appear as soon as you switch to online mode.

3.4.5.3 Configure Ethernet interface (2)

You can set up online communication of the PG/PC with the SIMOTION device via PROFIBUS, PROFINET, or Industrial Ethernet.

The most common use case is communication via Industrial Ethernet (communication via PROFINET protocol).

It is necessary to create a subnet if you are configuring IO devices. The following example shows how a subnet is created.

Procedure

Proceed as follows to create a subnet:

1. To do so, first switch to the network view in the project view.
2. Click with the mouse on the Ethernet interface **X127** of the SIMOTION device on which you want to
   create a subnet.
3. Right-click to select **Add Subnet** from the shortcut menu.

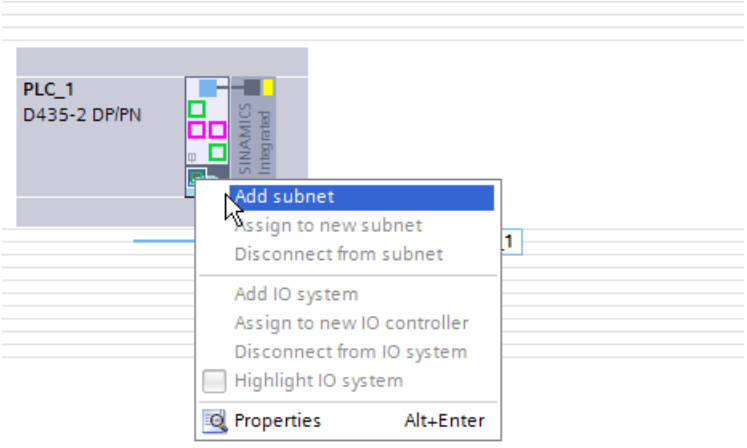


Figure 3-232 Adding a subnet

Result

The Ethernet subnet is shown in the network view.

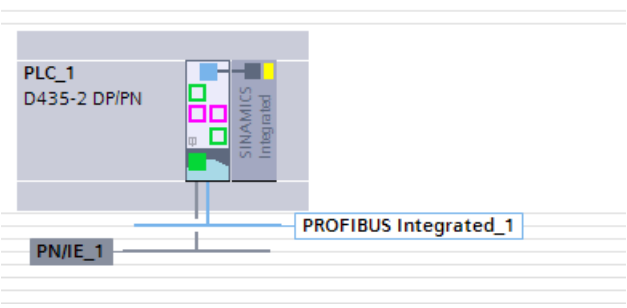


Figure 3-233 Ethernet subnet in the network view

3.4.5.4 Set up PG/PC interface (3)

Note

You assign the PG/PC interface in the TIA Portal.

Communication between a SIMOTION device and a PG/PC requires a conditioner card (for PROFIBUS) or an Ethernet interface. You configure, parameterize, program and test using the PG/PC.

You have the following functions to assign the PG/PC interface:

- **Connect online** function
  The function is available until the PG/PC interface has been successfully set up.
- **Online & diagnostics** function
- **Online access** function

Assign PG/PC interface

The following procedure describes the process for the Ethernet interface type by using the "Online access" function.

To assign the interfaces, proceed as follows:

1. Navigate to the relevant interface in the project navigation under **Online access**.
2. In the shortcut menu, click **Properties**.

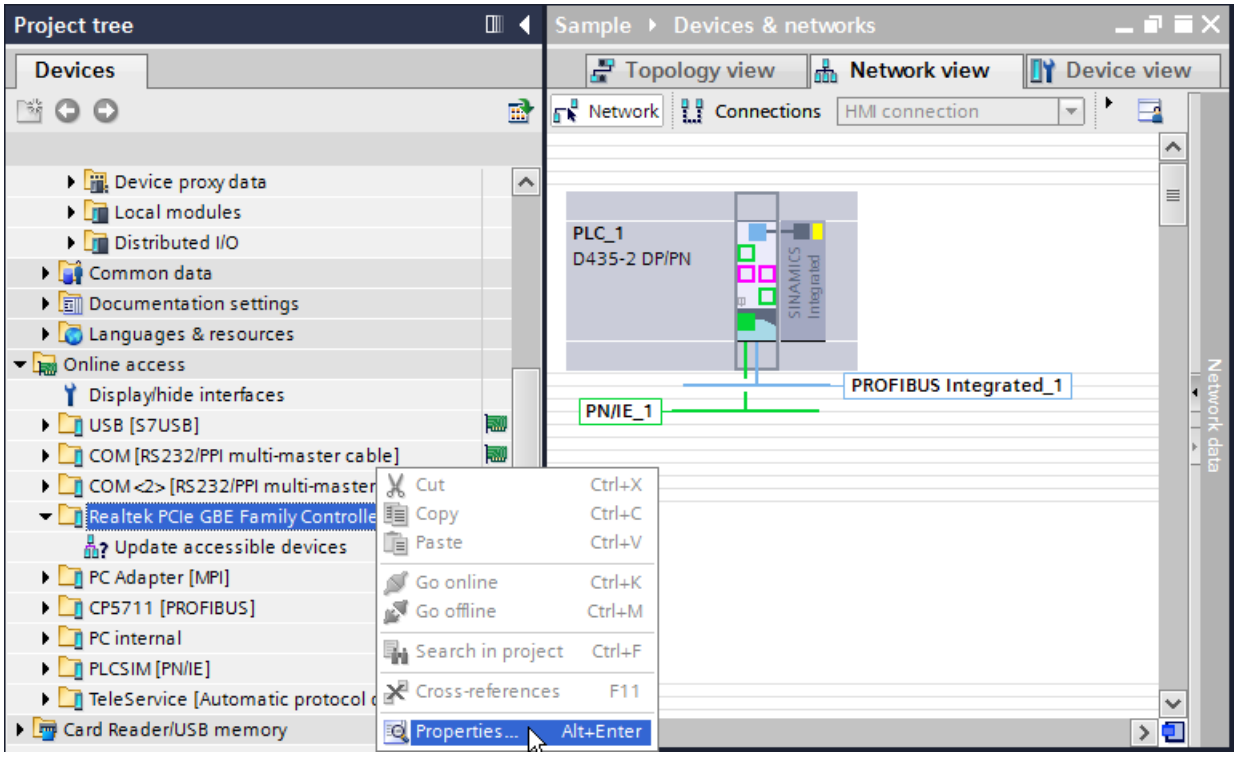


Figure 3-234 Online access properties

3. In the next step, select the subnet and apply the setting with **OK**.

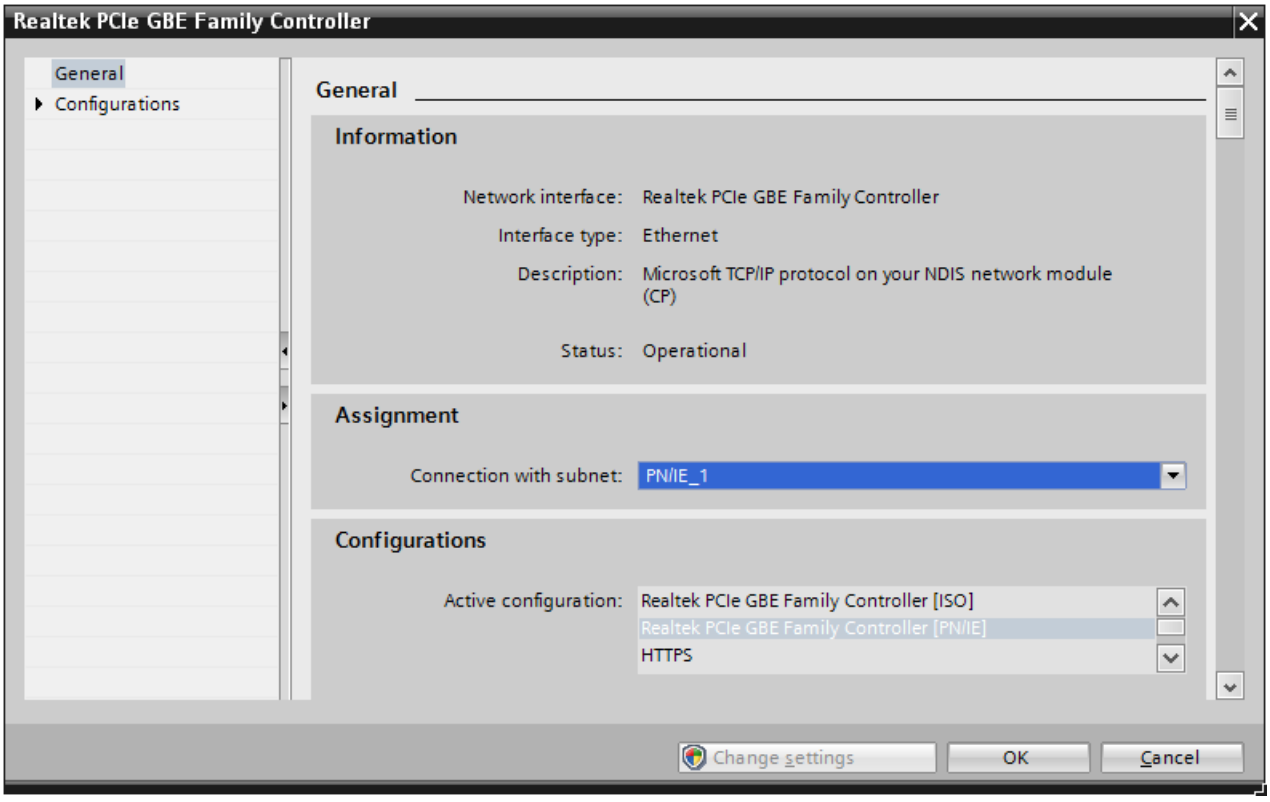


Figure 3-235 Assigning a subnet

Adding an IP address in the subnet

If the SIMOTION device and the PG/PC are not in the same subnet, add a suitable IP address from the subnet of the device to the PG/PC.

To automatically add an IP address in the subnet, proceed as follows:

1. Click Go online in the toolbar.
   The **Connect online** dialog opens.

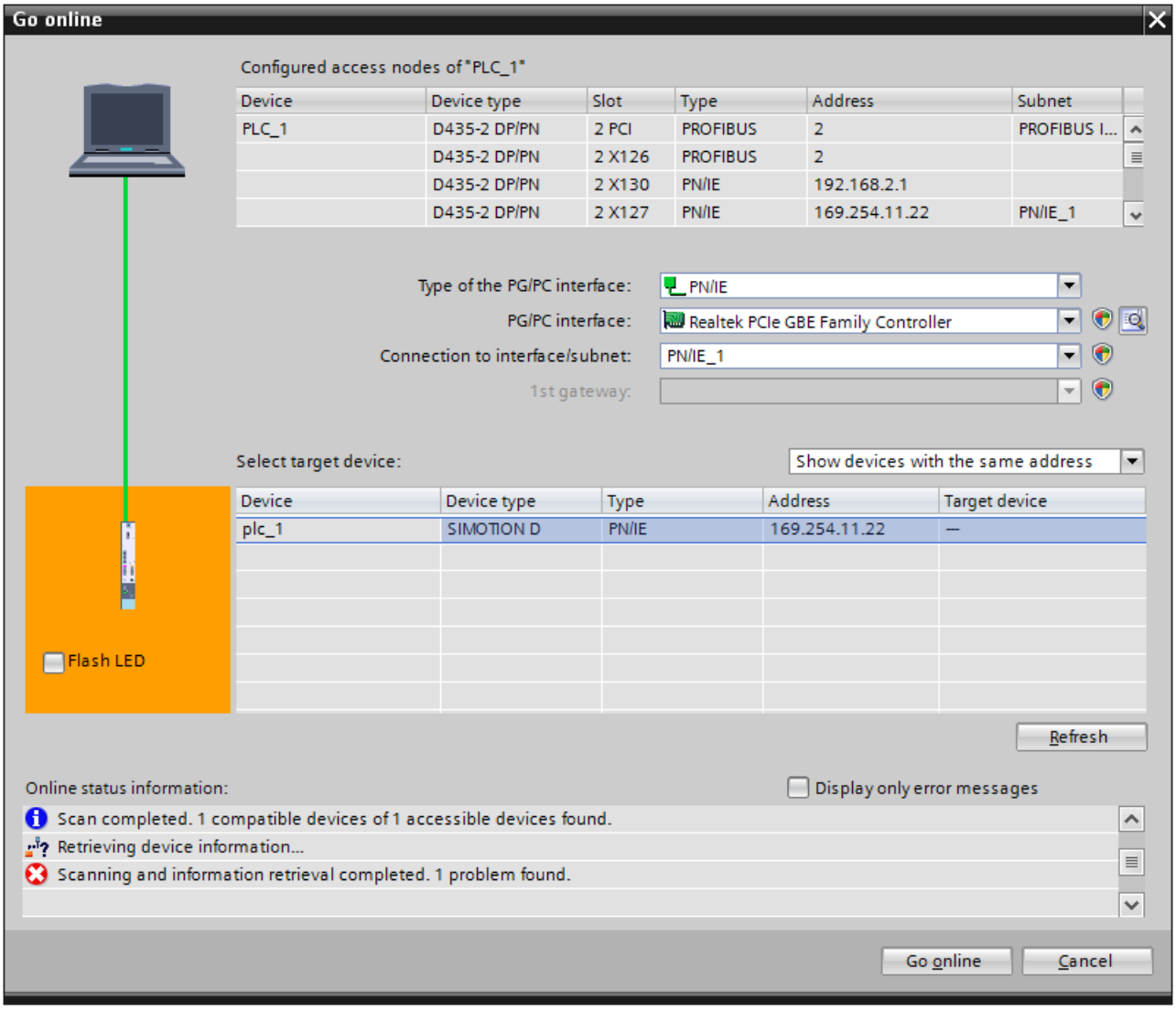


Figure 3-236 Selecting a device for online connection

2. Click **Connect** to establish a connection to the device located by the search.

3. If the SIMOTION device and the PG/PC are not in the same subnet, a message is displayed offering you the option of temporarily assigning a suitable IP address from the subnet of the device.

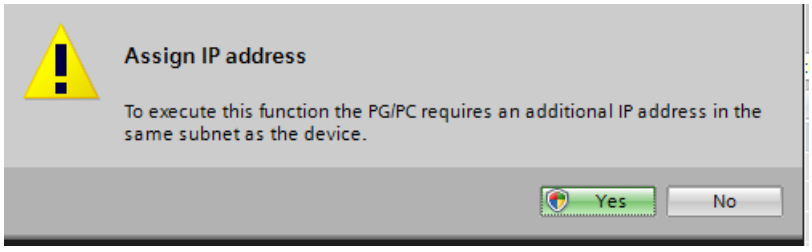


Figure 3-237 Assigning an IP address

4. Click **Yes** to confirm.

Figure 3-238 IP address assigned

Result

- You have assigned the PG/PC interface.
- The TIA Portal has assigned an IP address internally within the project.
- The online connection has been established.
  The title bar of the project navigation is now orange.

Displaying and deleting temporary IP addresses

You can have all temporarily assigned addresses displayed and delete them again.

1. Navigate to the relevant interface in the project navigator under Online access.
2. In the shortcut menu, click **Properties**.
3. Under **configuration**, select the item **IE-PG access**.

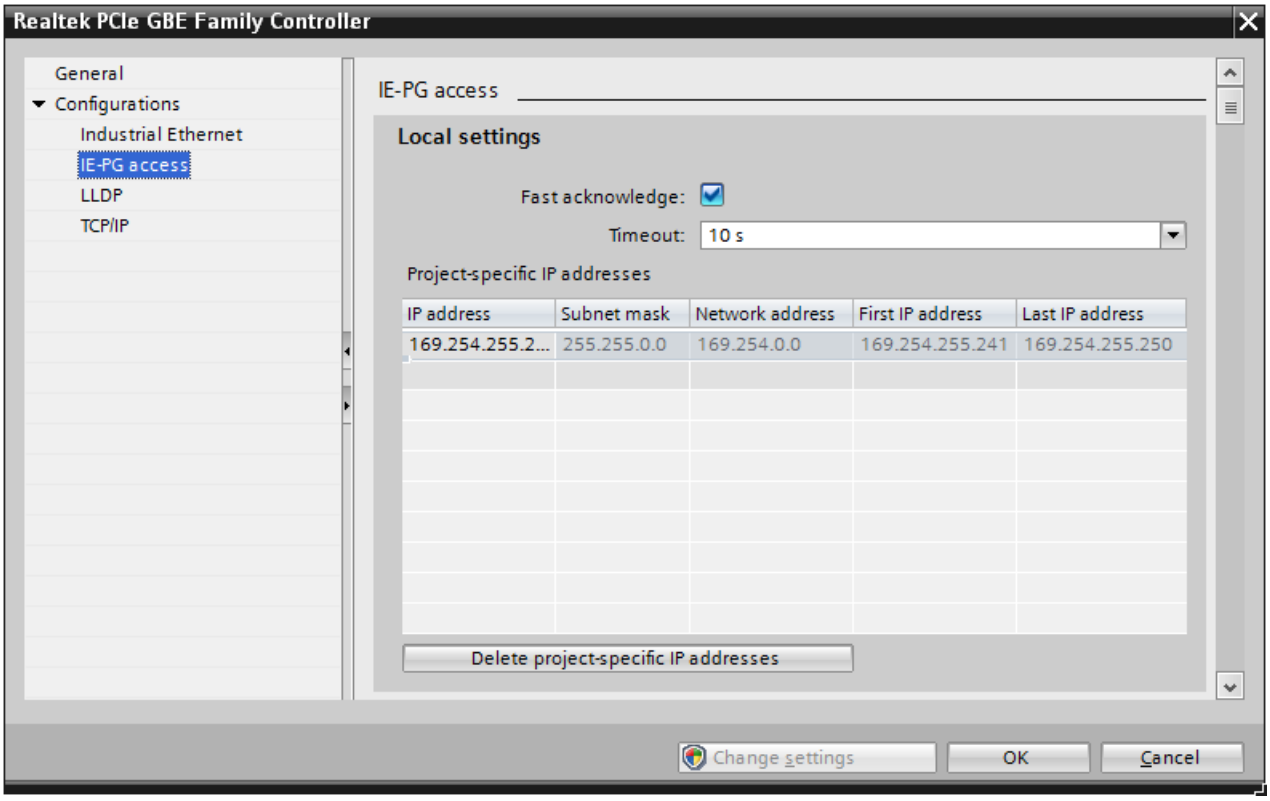


Figure 3-239 Displaying and deleting temporary IP addresses

Compiling the project and downloading the hardware configuration to the SIMOTION device

Note

Before you continue configuration in the SIMOTION SCOUT TIA, you should download the hardware configuration in the TIA Portal to the SIMOTION device.

Downloading to the SIMOTION device from the TIA Portal is necessary, for example, if you want to change the device names of the IP address. Moreover, downloading to the TIA Portal causes the routing information to be loaded.

Requirement

The online connection with the SIMOTION device is disconnected.

Procedure

To compile the hardware configuration and download to the SIMOTION device, proceed as follows:

1. Select the SIMOTION device and select via the shortcut menu **Compile > Hardware (changes only)**.

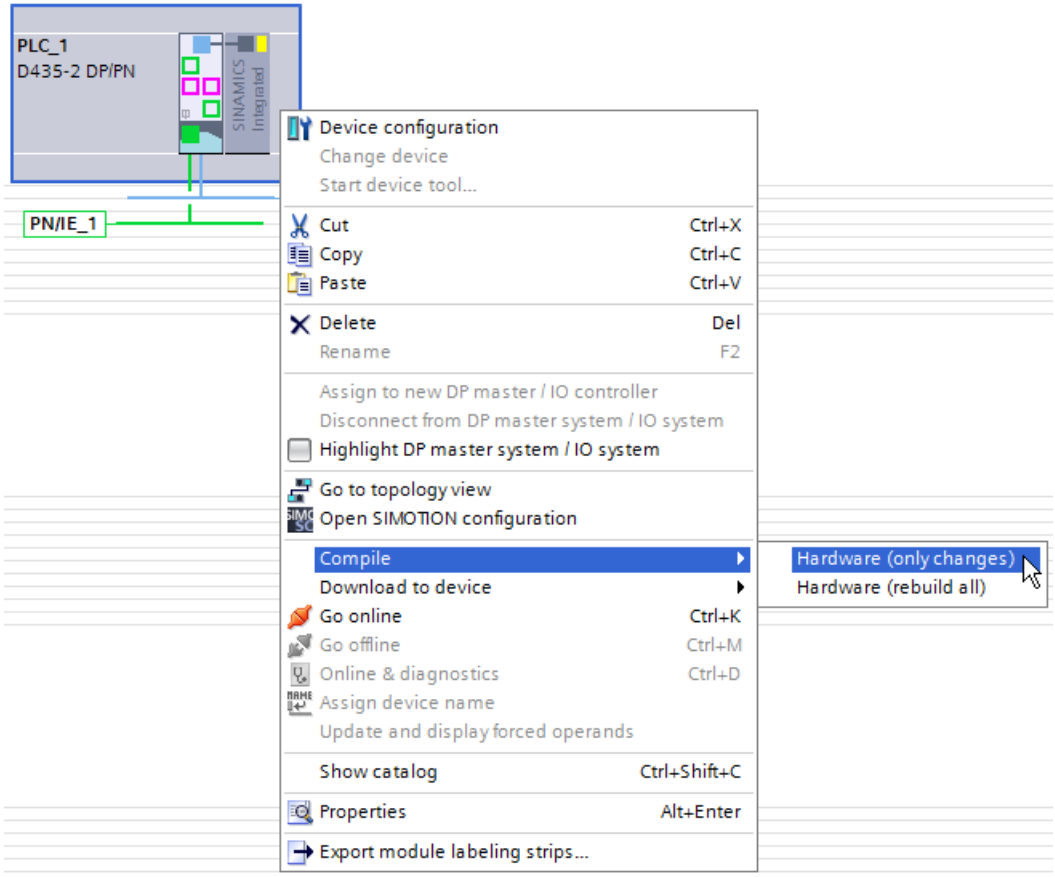


Figure 3-240 Compiling hardware

2. Select the SIMOTION device and select via the shortcut menu **Download to device > Hardware configuration**.

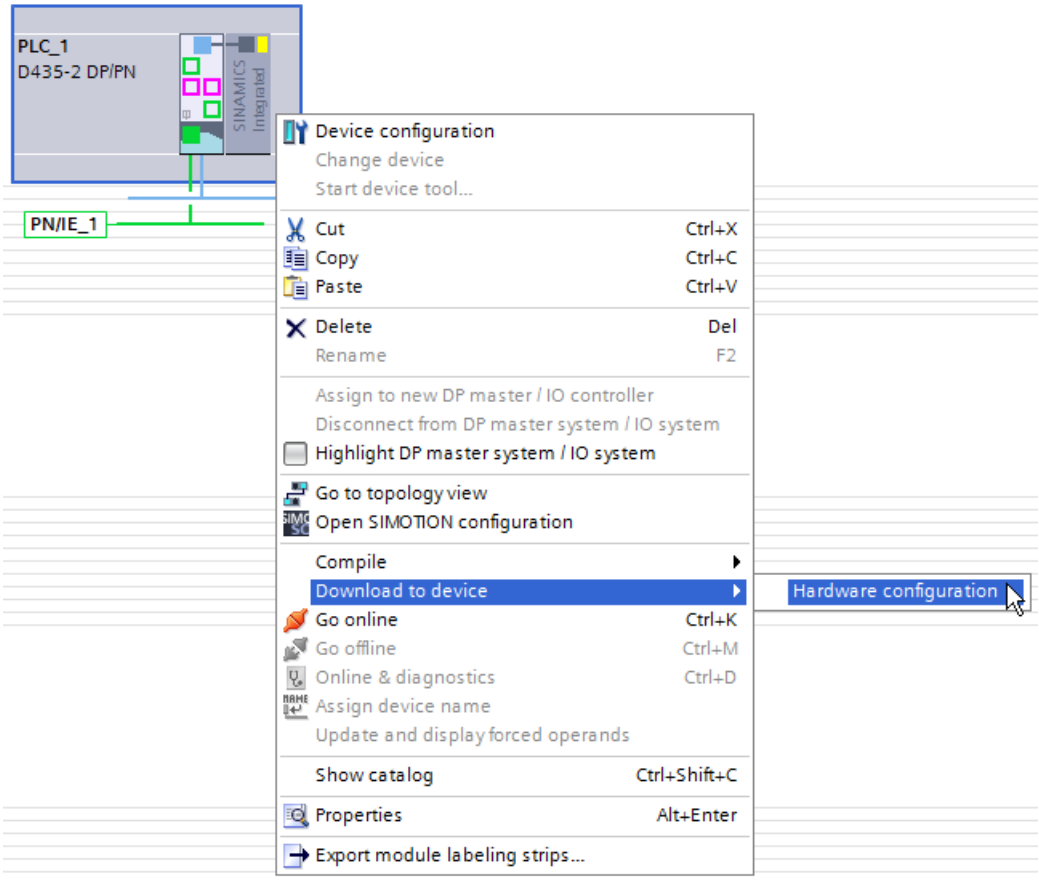


Figure 3-241 Loading hardware

Result

You have compiled the current hardware configuration and loaded it to the target device.

Additional references

For detailed information about the configuration of the online access, refer to the information system of the TIA Portal "Configuring online access".

3.4.5.5 Result in the sample project

Newly created SIMOTION device

Configuring of the SIMOTION device and network communication between the PG/PC and the SIMOTION device is complete.

- The newly created SIMOTION D435-2 device is shown in the project tree of the Project view.
- The PG/PC is connected with the SIMOTION device via Ethernet.
- The project is compiled and the hardware configuration is loaded into the SIMOTION device.

No other hardware configuration is required for the sample project.

3.4.6 Start SIMOTION SCOUT TIA

Note

You can only start SIMOTION SCOUT TIA from the TIA Portal.

After you have created the project in the TIA Portal and you have configured the hardware and communication, start SIMOTION SCOUT TIA to configure the technology.

Procedure

You can start SIMOTION SCOUT TIA in the following ways:

In the portal view

1. Click the "Motion & Technology" function.
2. Click the "Open SIMOTION configuration" entry in the secondary navigation.

In the project view

1. Start SIMOTION SCOUT TIA via the "Project > Open SIMOTION configuration" menu.
   Or double-click the "SIMOTION configuration" menu entry below the SIMOTION device in the project tree.

Note

Disconnecting the online connection to the TIA Portal

You should not go online simultaneously in the TIA Portal and SIMOTION SCOUT TIA.

Always disconnect the online connection in the TIA Portal before going online with SIMOTION SCOUT TIA. You need to do this so that programs can be compiled in SIMOTION SCOUT TIA.

Making basic settings

In SIMOTION SCOUT TIA, you can make various basic settings via the "Options > Settings..." menu.

To make the settings, proceed as follows:

1. In the "Options" menu, select the "Settings..." command.
   The "Settings" dialog opens.
2. Switch to the desired tab.
3. Make the settings.
4. Confirm with "OK".

Additional references

You will find detailed information on the individual tabs and settings in the SIMOTION SCOUT TIA Online Help.

3.4.7 Download the project to the target system

3.4.7.1 Overview

Aim of Getting Started

In this configuring step, you create the prerequisites for configuring the drive.

- You back up the created sample project to the hard disk.
- You compile the project into executable code.
- You establish online communication with the SIMOTION device.
- You download the project from the PG/PC to the SIMOTION device.

3.4.7.2 Save and compile the project

To be able to download a project created in SIMOTION SCOUT TIA to the target system, the project must be saved in executable code.

The command **Save project and compile changes** combines both steps. The project is backed up to the hard disk. SIMOTION SCOUT TIA searches the entire project for changes and compiles only the changes.

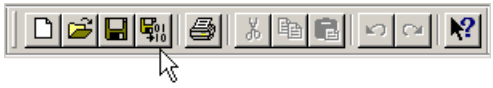
Note

Use the command **Save project and compile changes** for preference for day-to-day work.

Several variations of the "Save" command are available for selection under the menu title **Project**. You can find information on this in the online help under **Save and compile**.

You save and compile a project as follows

Select the menu command **Project > Save and compile changes** or click the relevant button in the toolbar.



The compilation run is logged in the detail view of the workbench. Information, warning and compilation errors are shown there in plain text.

Switch on detail view

The detail view might be switched off. Click the menu item **View > Detail view** to activate the view.

3.4.7.3 Connect to selected target devices – Go online

To download project data from SIMOTION SCOUT TIA to the hardware, or to transfer machine data in the other direction from the hardware to the SIMOTION SCOUT TIA project, communication between the PG/PC and the SIMOTION device must be activated.

The status of the network communication is displayed in the footer of the workbench:

Network communication is switched on. Data can be exchanged.

The network connection is switched off.

You go online as follows

1. Select the menu command **Project > Connect with selected target devices** or click the relevant button in the toolbar.

When the command is first called, SIMOTION SCOUT TIA opens the **Target Device Selection** dialog box. The dialog enables individual selection of the devices to which SIMOTION SCOUT TIA is to connect.

Note

For fast working with SCOUT TIA, we recommend only selecting devices that are currently needed (e.g. deselection of the SINAMICS drives if drive commissioning has been completed).

You will find the setting in the menu under **Target system** > **Select target devices...**. Selection and deselection of the device online can also be

performed via the shortcut menu **Connect to target device** on the device.

2. In the dialog box, select the configured SIMOTION device **PLC_1** and the integrated drive **SINAMICS_Integrated_1**.

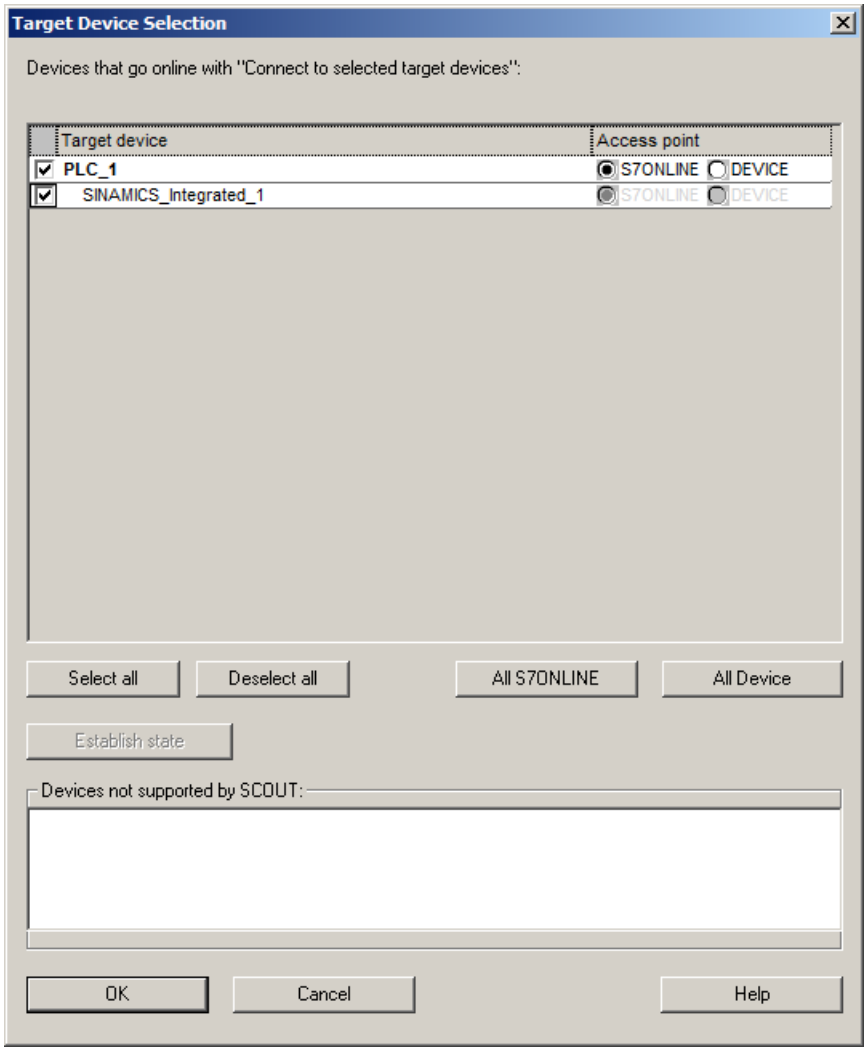


Figure 3-242 Target Device Selection dialog

Note

The properties of the access point S7ONLINE are defined in the TIA Portal. With this access point, you go online via the interface that you have configured in the TIA Portal.

The DEVICE (STARTER) access point provides the option of connecting SIMOTION SCOUT TIA either in parallel or alternatively to S7ONLINE, directly to a device, e.g. via the Ethernet interface. The properties configured in SIMOTION SCOUT TIA are only applied for the DEVICE access point.

3. Click **OK**.
   SIMOTION SCOUT TIA establishes the online connection.
4. Track the consistency check to completion on the **Target system output** tab in the detail area.

Note

If the hardware configuration has not yet been loaded, SIMOTION SCOUT TIA will indicate on the first attempt to go online that access to the SINAMICS Integrated is not possible. Access to the SINAMICS Integrated is only possible when the hardware configuration has been loaded (see Chapter AUTOHOTSPOT, Compiling the project and downloading the hardware configuration to the SIMOTION device). Close the message.

In the project tree, the connector symbols on the SIMOTION device and on the integrated SINAMICS drive change. Only the connector symbol of the SIMOTION device is partially colored green. Green indicates the existing online connection.

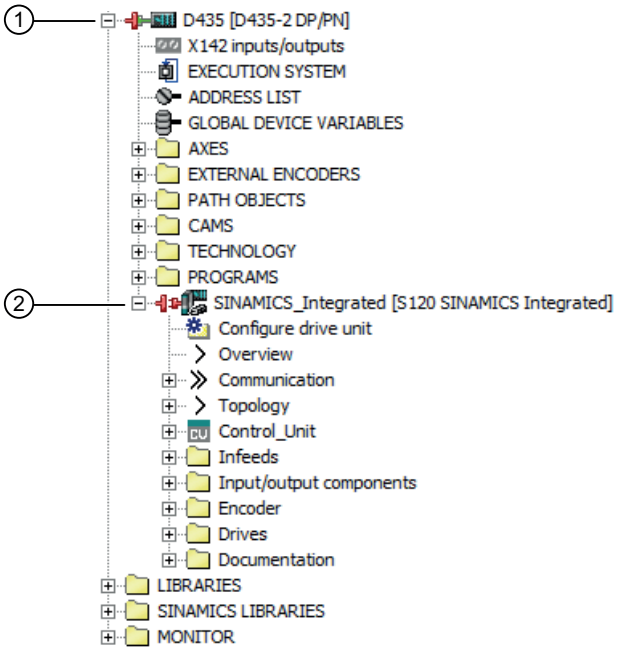


Figure 3-243 Project tree after first going online

Additional references

You will find detailed information on configuration of the DEVICE access point *SIMOTION SCOUT TIA Configuration Manual*.

3.4.7.4 Download the project to the target system

A complete project download is only possible in STOP mode. If required, SIMOTION SCOUT TIA offers a change of operating mode during the download procedure.

Note

A project download is necessary when you have made changes to the configuration and programming in SIMOTION SCOUT TIA.

You download the project as follows

1. Select **Project > Download to target system** in the menu, or click the **Download project to target system** button in the toolbar.

   The dialog **Download to target system** appears.

2. Activate the checkbox **After loading, copy RAM to ROM**. This saves the RAM of the SIMOTION device to the memory card (ROM) of the SIMOTION device. In this way, the configuration is retained after the power supply has been switched off and on again.

3. Start the download operation with **Yes**.
   The project data and the data of the hardware configuration are downloaded to the RAM of the target system.
   If you are asked whether the CPU is to be switched to STOP, confirm with **Yes**.
   SIMOTION SCOUT TIA performs numerous checks that are logged on the **Target system output** tab in the detail area. You will find there the concluding entry **Download to target system completed successfully**.

Note

The first download to the target system also downloads the data of the technology package. This operation can take several minutes.

Note

Depending on the firmware version on the CF card and on the SINAMICS components (DRIVE-CLiQ components such as Line Module, Motor Modules, Terminal Modules, etc.), the firmware of the components is automatically upgraded or downgraded.

Updating can take several minutes and is indicated by the relevant messages in the output window of the SIMOTION SCOUT TIA.

A firmware update on DRIVE-CLiQ components is signaled by red-green flashing of the RDY LED:

- Firmware update in progress: RDY LED flashes slowly (0.5 Hz)
- Firmware update ended: RDY LED flashes quickly (2 Hz), POWER ON required

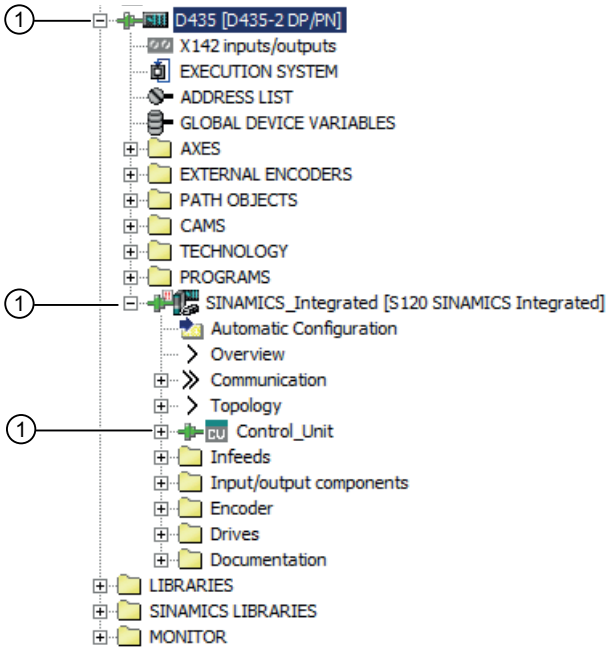
These flashing patterns are displayed additionally via a yellow RDY LED on the SIMOTION D, indicating that components connected to SIMOTION D are performing a firmware update or that all components have completed their firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SIMOTION SCOUT TIA and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

Automatically established connection to the drive

SIMOTION SCOUT TIA automatically establishes the online connection to the drive immediately following the project download.

In the project tree, the connector symbol changes on the SIMOTION device, the integrated SINAMICS drive, and the SINAMICS control unit. The connector symbol is entirely green, indicating that the project data in the PG/PC is identical with the project data in the target system.



① Green connector symbol: Element is in online mode. The project data in the PG/PC is identical with the project data saved in the target system.

Figure 3-244 Project tree

From this point, you can access the drive online with the PG/PC.

3.4.8 Configure the drive

3.4.8.1 Overview

Aim of Getting Started

In this part of Getting Started, you configure the integrated drive of the SIMOTION D435-2 device. You use auto-configuration for this purpose.

Requirements

- You have downloaded the sample project to the target system, refer to the section Download the project to the target system (Page 487).
- SIMOTION SCOUT TIA is online with the SIMOTION device and the integrated drive (green connector symbols).
- The SINAMICS drive components Infeed and Power unit, as well as Motor and Encoder, are connected to the SIMOTION control unit via DRIVE-CLiQ. This requirement for auto-configuration was referred to at the start of the Getting Started, see the section AUTOHOTSPOT.

Drive

The speed and current control functions for controlling the motor are implemented in the drive.

Automatic drive configuration

SIMOTION SCOUT can read out the electronic type plates of the SINAMICS drive components via the DRIVE-CLiQ interface and can use this data to configure the drive automatically. The data does not therefore need to be entered manually. The sample configuration of a SIMOTION D435-2 device shown here requires full DRIVE-CLiQ wiring.

As an alternative to automatic drive configuration, you can also configure a D4x5-2 device offline. Offline configuring is demonstrated in the online help version of Getting Started for SIMOTION C and SIMOTION P.

3.4.8.2 Automatic configuration of the drive

You open the Automatic Configuration as follows

Double-click on **Automatic Configuration** in the project navigator under the drive SINAMICS_Integrated.

The **Automatic Configuration** dialog is displayed.

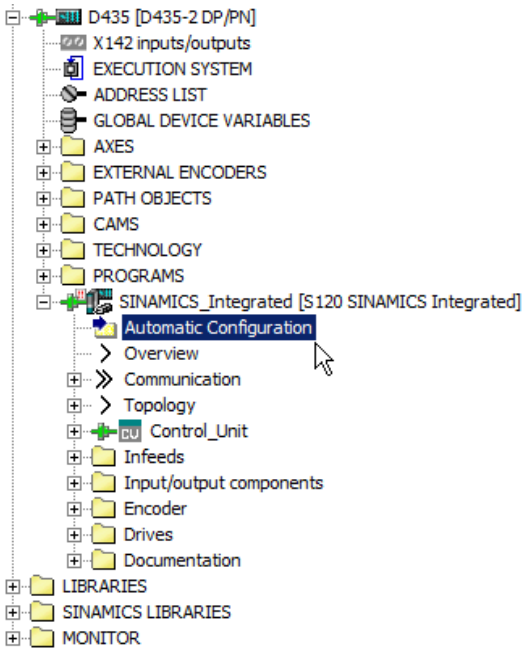


Figure 3-245 Call up Project tree, Automatic Configuration

Note

In the project tree, the element **Automatic Configuration** is only visible if you have downloaded the project to the target system and SIMOTION SCOUT TIA is online with the SIMOTION device. The project must be consistent; see the section Download the project to the target system (Page 487).

You cause the drive to be configured automatically as follows

1. Click the **Configure** button in the **Automatic Configuration** dialog.
2. Confirm the prompt regarding restoring the factory settings with **Yes**.
   The confirmation prompt appears if the drive unit is not in the "First commissioning" state.

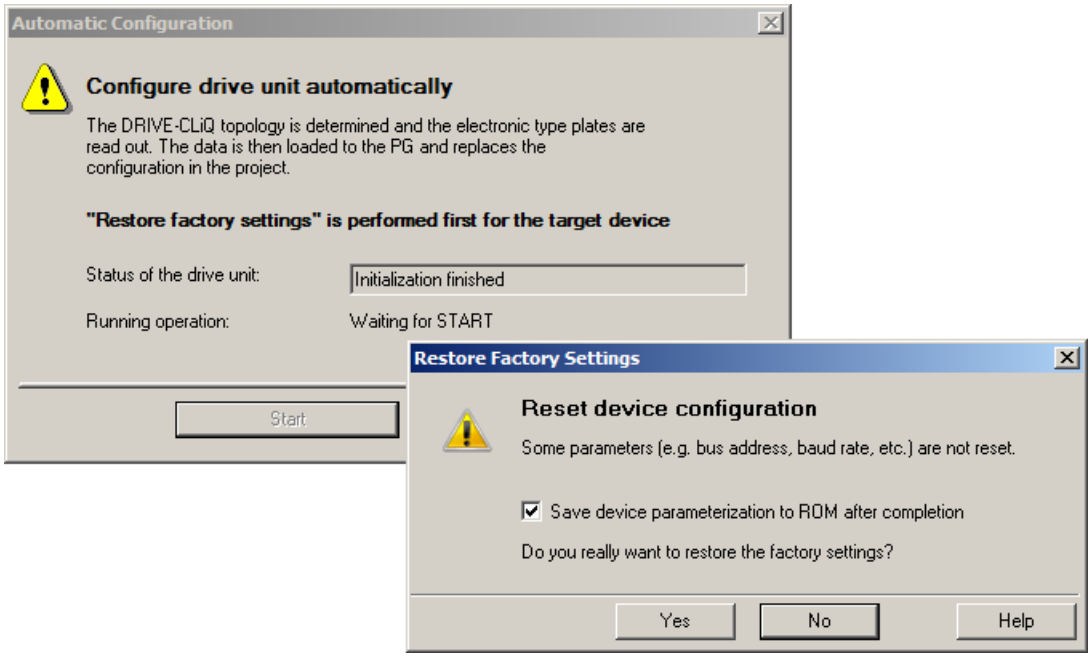


Figure 3-246 Start automatic drive configuration and restore factory settings

3. In the **Automatic Commissioning** dialog, you can specify whether you are using a drive object of the type servo or vector.
   Select **Servo**.
4. Click the **Create** button in the **Automatic Commissioning** dialog.
   Automatic configuring is started.

Note

Firmware update

If the firmware version on the DRIVE-CLiQ components is different to that on the CF card, a firmware update is now performed automatically.

In this case, proceed as follows:

- Wait for the procedure to finish. This can take several minutes.
- Go offline.
- Switch the power supply to the SIMOTION device off and then on again.

5. If a firmware update has not been performed, remain online to be able to see the changes in the project tree.

6. Open the **Infeeds** system folder in the project tree under the integrated drive.
   - If you are using an infeed with DRIVE-CLiQ interface, the auto configuration has created the infeed there.
   - If you are using an infeed without DRIVE-CLiQ interface, the folder is empty. The infeed is not known in the project.
     The other configuring requirements resulting in this way are dealt with in the next section Configure the infeed (Page 499).
7. Open the **Drives** system folder in the project tree under the integrated drive. The folder contains the drive detected by the auto configuration.

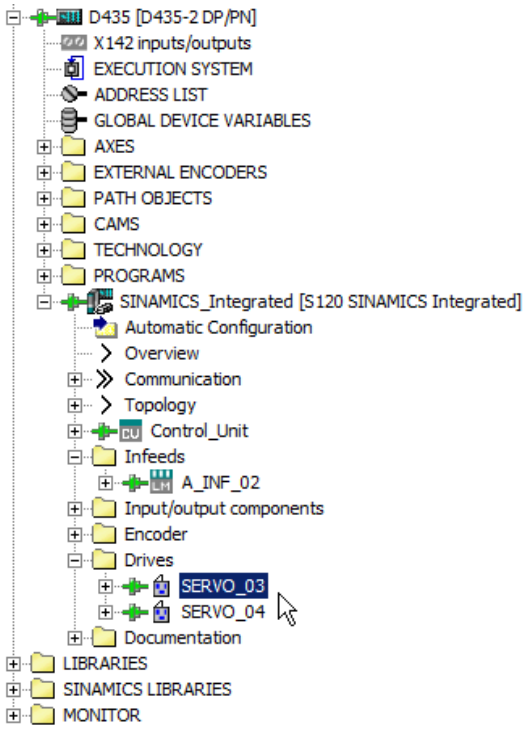


Figure 3-247 Project navigator, configured infeed and drives

8. Go offline.
   Click in the toolbar on the **Disconnect from target system** button.

In the footer of the workbench, **Offline mode** is indicated. Network communication between the PG/PC and the SIMOTION device has been cleared down.

Change the name of the drive

The automatic configuration also assigns the object name of the drive; in this example, SERVO_03.

You can change the name later. Changing is only possible in offline mode.

Proceed as follows:

1. Open the context menu of the drive by right-clicking. Select the **Rename** command there.
2. Assign the new name. Then confirm with **OK**.

As with all changes carried out in the project offline, the name change makes the project inconsistent. To restore the consistency between the "Project on the PG/PC" and the "Project on the SIMOTION device", another project download is necessary. The procedure is described in the section Download the project to the target system (Page 487).

3.4.8.3 Result in the sample project

If you are using an infeed with DRIVE-CLiQ interface, the drive is ready for operation. It can be interconnected with an axis.

3.4.9 Configure the infeed

3.4.9.1 Overview

Aim of Getting Started

In this part of Getting Started, you configure the infeed.

The integrated SINAMICS control unit only starts the drive when the infeed is ready. The project must therefore know the interface via which the drive receives the ready signal of the infeed.

Two cases must be distinguished here:

- Infeed with DRIVE-CLiQ interface
  If an infeed with DRIVE-CLiQ connection has already been created, the infeed ready signal (r0863.0) is automatically interconnected to "Infeed operation, p864" of the drive when drives are inserted (only applies to drives that are attached to the same drive unit as the infeed).
  You can continue immediately with the next configuring step, see the section Configure the axis (Page 501).
- Infeed without DRIVE-CLiQ interface
  If you are using an infeed without a DRIVE-CLiQ interface, e.g. a Smart Line Module, you must interconnect the ready signal of the infeed via terminals. The following section describes the procedure.

Requirements

- You have configured the integrated drive of the SIMOTION D435-2 device, see the section Configure the drive (Page 494).
- SIMOTION SCOUT TIA is in offline mode.

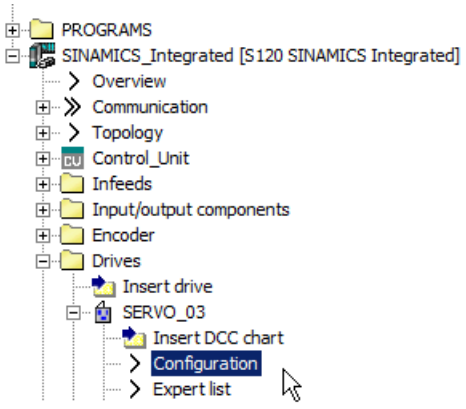
3.4.9.2 Configuring an infeed without DRIVE-CLiQ interface

An infeed without DRIVE-CLiQ interface provides the ready signal (p0863.0) via an output terminal. In the project, you specify the input (r0722) of the integrated SINAMICS Control Unit at which the signal is active. The drive supplied by the infeed uses the signal as a ready signal (p0864).

You interconnect the ready signal of the infeed as follows

In the sample project, the ready signal of the infeed (terminal "DO: Ready" of the infeed) is wired to the DI 0 of the D435-2.

1. Open the configuration dialog of the drive. To do so, double-click **Configuration** in the project tree below the drive.



2. Click in the working area on the **Configure DDS** button with the yellow background.

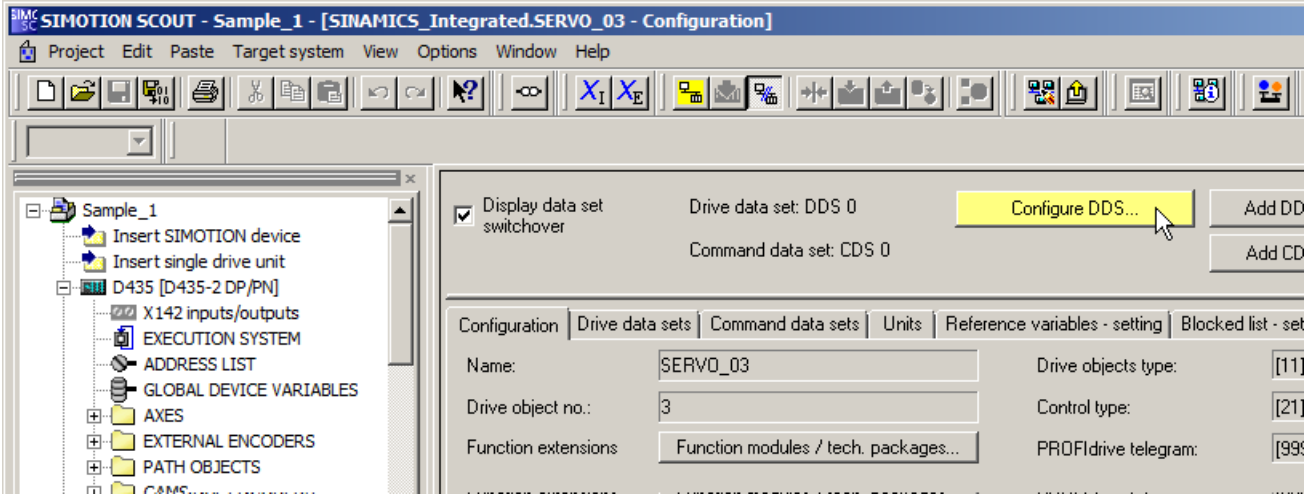


Figure 3-248 Configure DDS

The drive wizard opens.

3. Click **Next** in the drive wizard until you reach the dialog **Configuration - SINAMICS_Integrated - Power Unit BICO Technology**.

4. In the **Power unit BICO** dialog, input field **p0864**, select the digital input to which the ready signal of the infeed is wired (e.g. DI 0).

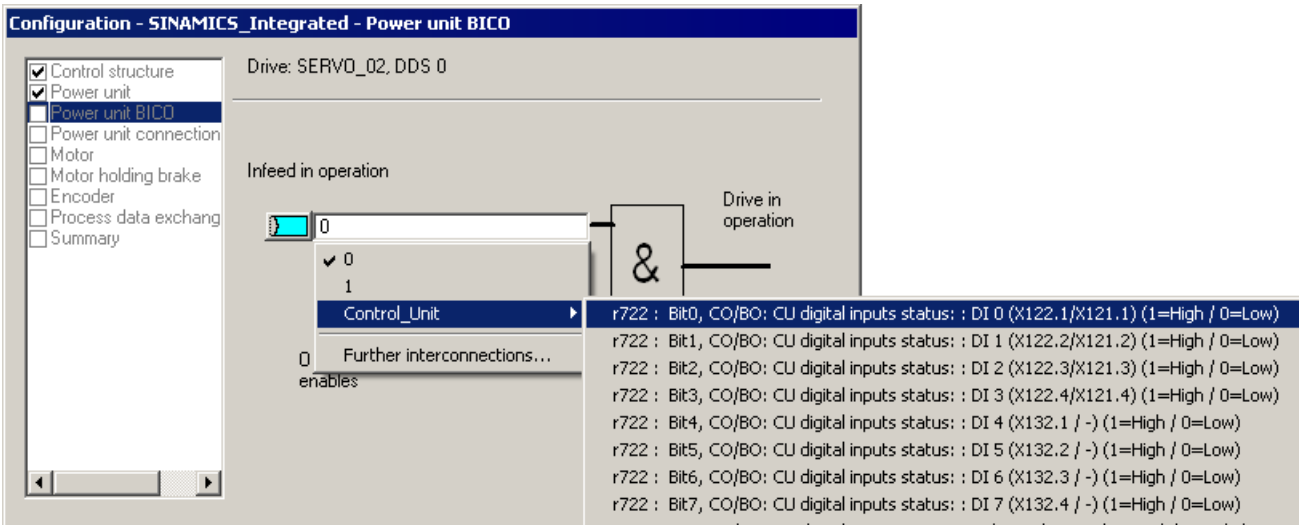


Figure 3-249 BICO interconnection in the drive wizard

5. Click **Next**. Run through all the other dialogs without change until the final **Summary** dialog.
6. Click **Finish**.
   The configuration is thus completed.
7. The configuration dialog of the drive is no longer required. Click **Close** at the bottom right of the dialog box.

3.4.10 Configure the axis

3.4.10.1 Overview

Aim of Getting Started

This part of Getting Started shows you how to create and configure an axis in the project, and how to interconnect it with the integrated drive of the SIMOTION D device. SIMOTION SCOUT TIA provides the axis wizard for this purpose.

Requirements

You have configured the integrated drive of the SIMOTION D device, see the section Configure the drive (Page 494).

Axis technology object

Technology objects represent the respective real objects (e.g. a position axis) in the controller.

The Axis technology object offers users a technological view of the drive and the encoder (actuator and sensor), provides technological functions for these components, and conceals the actual hardware interface.

The Axis technology object contains extensive functionality, e.g. communication with the drive, actual value processing, position control, and positioning functionality. It executes control and motion commands and indicates states and actual values.

Technological limitations and mechanical values of the axis and encoder (e.g. leadscrew pitch and gears) are set on the axis. You can then work exclusively with technological variables.

Axis wizard

SIMOTION SCOUT provides the axis wizard for creating a new axis. The wizard scans the basic settings and interconnects the TO axis with a drive.

The wizard can only be run through once. Later changes to the configuration can be entered in the corresponding dialogs of the TO.

3.4.10.2 Creating an axis

For the sample project, create the axis **Axis_1**. Assign the axis to the drive you have configured in the section Configure the drive (Page 494).

Run through the wizard and confirm all standard settings with **Next**.

You create a real axis in the project as follows

1. Open the **AXES** folder in the project navigator.
2. Double-click **Insert axis**. The **Insert axis** dialog appears.
3. Name the axis in the **Name** field. Use the designation **Axis_1** for the axis of the sample project.

4. Leave the preset axis technology at the default **Positioning**.

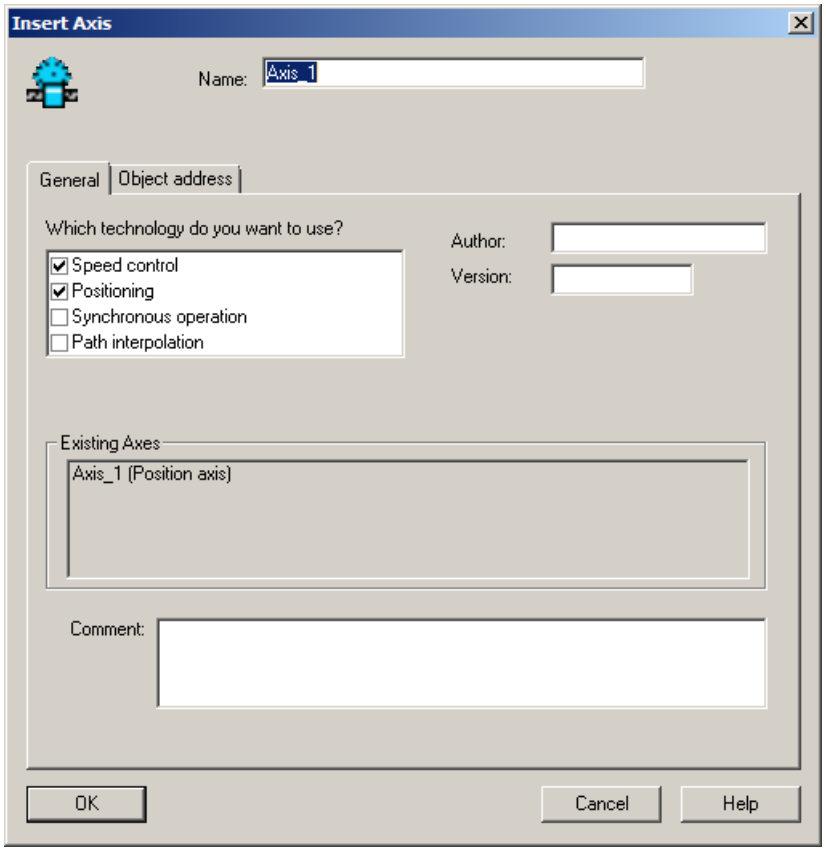


Figure 3-250 Axis wizard, creating a real axis

5. Click **OK**.
The axis configuration wizard appears.

6. Define the axis type.
   For the sample project, select preferably a linear axis with an interconnection to an electrical drive.
   Activate the fields **linear** and **electrical**.

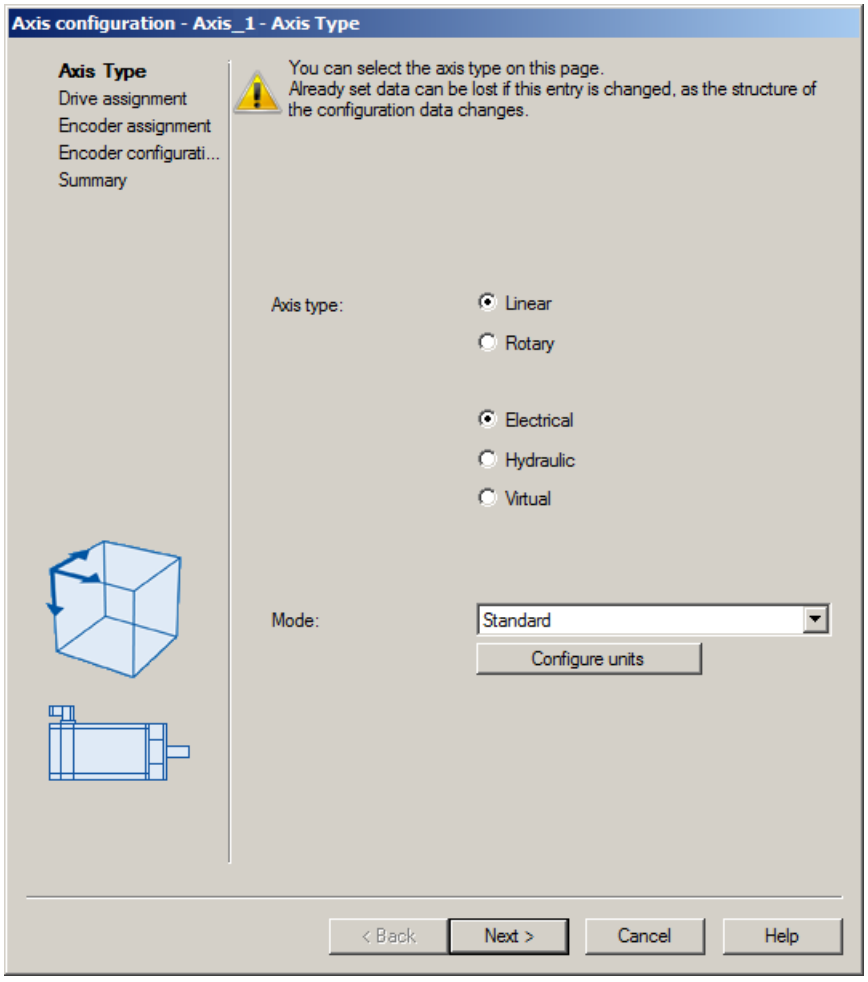


Figure 3-251 Axis wizard, determining the axis type

7. Click **Next**.
   The **Drive assignment** dialog appears.

8. Assign the axis to the drive you have configured in the section Configure the drive (Page 494). To do so, click the drive unit in the **Assignment partner** column. Click in the tree under the drive object on the drive object you want to interconnect. **Assign** appears in the **Assignment** column. The axis/drive assignment is thus defined.
   Click **Next**.

   Note

   As an alternative to assigning the axis to an already configured drive, the axis wizard offers two further selection options:

   - Define the axis/drive assignment later:
     The axis is to be created and not assigned to a drive until later. Programming and simulation of the axis are also possible here.
   - Create drive:
     The drive wizard can can be called up from the axis wizard (offline configuring). The axis can thus be created in one step along with the drive, and assigned to the drive.

   The alternative procedures are not considered further in Getting Started.

9. Assign the motor encoder.
   In the **Encoder assignment** window, you use the motor encoder connected to the drive as the standard setting.
   Click **Next**.
10. The axis wizard then assembles the configuration data in an overview. You thus have an opportunity to check and correct the data before accepting it into the project.
    Close the dialog with **Finish**.
    The configured axis is displayed in the project navigator.

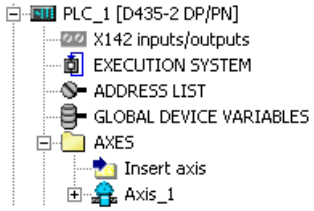


Figure 3-252 Axis created

Automatic settings of the engineering system

The engineering system automatically defines the PROFIdrive axis telegrams required for communication, as well as the addresses used.

In the same way, telegrams are extended and interconnections automatically created in the drive, depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

Drive and encoder data, as well as reference variables, maximum variables, torque limits, and granularity in torque reduction of the SINAMICS S120 are accepted automatically for the configuration of the SIMOTION technology objects "TO axis" and "TO external encoder". This data no longer has to be entered in SIMOTION.

Further axis configuration

Further axis configuration is possible via dialogs that can be accessed in the project navigator under the axis. No other configuration is required for the sample project.

Downloading the axis configuration to the target system

- Save and compile the project.
- Download the sample project with the axis configuration to the target system to be able to test the functioning of the axis in the next configuration step.

3.4.10.3 Download the axis configuration to the target system

Download the sample project with the axis configuration to the target system to be able to test the functioning of the axis in the next configuration step.

Steps

1. Save project and compile changes.
2. Establish online connection if offline.
3. Download the project to the target system.

Save and compile changes

Click in the toolbar on the **Save project and compile changes** button.

Establish the online connection

Click in the toolbar on the **Connect to selected target devices** button.

Download to target system

Click in the toolbar on the **Download project to target system** button.

3.4.11 Test the axis with the axis control panel

3.4.11.1 Overview

Aim of Getting Started

In this part of Getting Started, you test the configured axis. SIMOTION SCOUT TIA provides you with the axis control panel for this purpose.

Requirements

- You have configured the infeed, see the section Configure the infeed (Page 499).
- You have created and fully configured an axis in the sample project, see the section Configure the axis (Page 501).
- The project with the axis configuration has been downloaded to the target system, see the section Download the axis configuration to the target system (Page 506).
- SIMOTION SCOUT TIA is in online mode.

Control and monitoring of individual axes

The axis control panel is used for controlling and monitoring individual axes. You can use it to traverse axes together with the drive.

You can use the control panel to perform the following tasks, for example:

- Test axes before you traverse them via a program.
- Test whether the axes can be traversed using the control panel when a fault occurs.
- Set and remove an axis enable.

3.4.11.2 Working with the axis control panel

In the sample project, you traverse the set-up axis in jog mode to test the correct functioning of the axis.

You open the axis control panel as follows

Open the **AXES** folder in the project navigator. Double-click below the axis on **Control panel**.

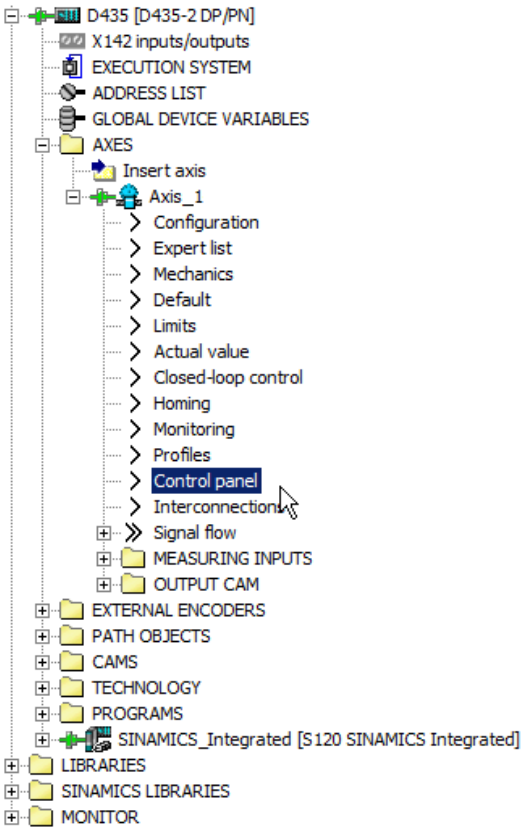
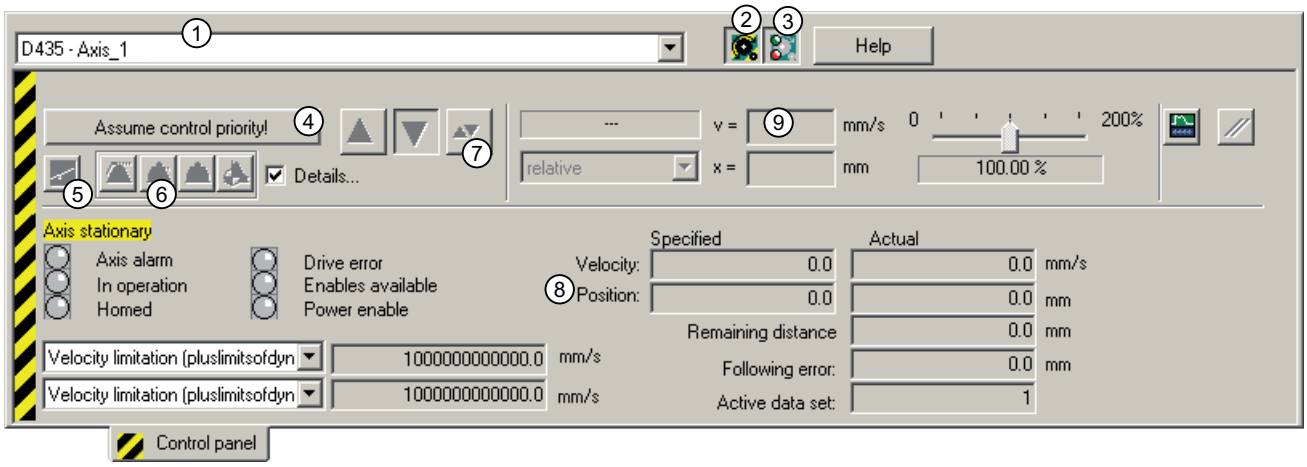


Figure 3-253 Calling the axis control panel

The axis control panel is opened in the detail view.



① ... ⑨ Reference is made to the circled digits in the text below.

Figure 3-254 Axis control panel

You traverse an axis with the axis control panel as follows

1. The field at top left of the control panel ① shows the currently selected axis.
2. Showing areas in the control panel:
   The control panel is divided into a control area and a diagnostics area. The areas may be hidden.
   Click the **Show/hide control area** button ② and/or the **Show/hide diagnostics area** button ③ to show the respective area.
3. Assume control priority – Observe safety regulations:
   To be able to traverse the axis from the PG/PC, the PG/PC must obtain control priority.
   Click the **Assume Control Priority** button ④.
   The **Assume Control Priority** dialog opens.

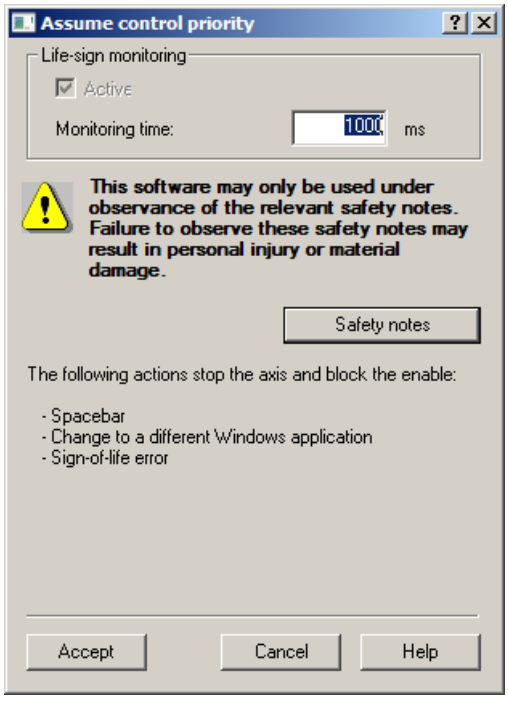


Figure 3-255 Axis control panel, assume control priority

Observe the safety regulations and confirm with **Accept**.
The PG/PC now has control priority.

Note

You can stop the axis at any time by pressing the space bar.

4. Enable the axis:
   Click the **Set/remove enables** button ⑤ to enable the axis.

Confirm the **Switch axis enable** dialog that appears with **OK**.
The switches **Start motion**, **Stop motion** and **Jog** ⑦ are enabled.

5. Homing the axis:
   Click the **Home axis** button.
   In the dialog box, select **Homing type Set home position**, and enter the value **0** under **Home position coordinates**.
   Confirm the dialog box with **OK** and start homing the axis.
6. Click the **Position-controlled traversing of the axis** button ⑥.

   The dialog **Start axis position-controlled** dialog appears.
   Specify the velocity of the axis. Click **OK** to close the dialog.
   The velocity appears in the field **v=**⑨.
7. Start the axis motion:
   Click **Jog**⑦.

   The axis is traversed while you press the switch. In the fields **Velocity** and **Position**⑧, you can monitor the traverse motion.
   The axis test is thus executed.
8. Deactivate axis enable:
   Click **Set/remove enables**⑤.
   Confirm the **Disable axis** dialog with **OK**.

   Note

   If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

9. Return control priority:
   Click **Give up control priority**④ to deactivate control of the axes from the PG/PC. In this operating state, the axes can no longer be controlled from the PG/PC.
10. Go offline:
    Select **Project > Disconnect from target system** in the menu. Or click in the toolbar on the **Disconnect from target system** button.

3.4.11.3 Result in the sample project

You have traversed the axis of the sample project with the axis control panel and thus ensured its correct functioning. Configuring the axis is thus completed.

3.4.12 Configure digital outputs

Aim of Getting Started

In this part of Getting Started, you configure two of the digital inputs/outputs available on the SIMOTION D device as outputs.

You require the outputs for the program you will write in the subsequent configuring step, see the section AUTOHOTSPOT.

Requirements

SIMOTION SCOUT TIA is in offline mode.

I/O channels of the terminal X142

You use the I/O channels of the terminal X142 for the sample project.

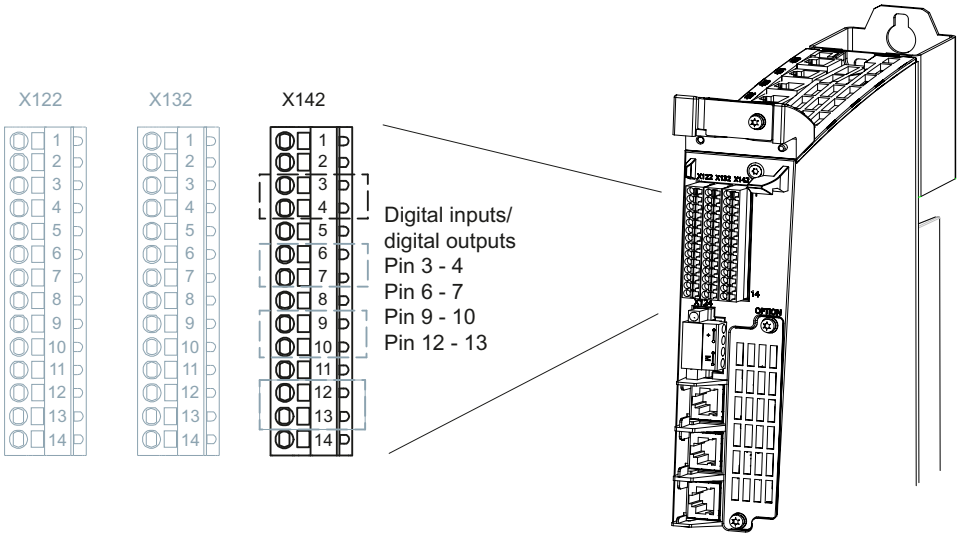


Figure 3-256 SIMOTION D445-2 DP/PN, position of the digital interface X142

The I/Os of the terminal X142 are permanently assigned to SIMOTION D4x5-2. The terminal is thus visible in the project tree under the SIMOTION device.

You define the digital outputs as follows

1. Double-click the element **Inputs/outupts X142** in the project tree below the SIMOTION device.
   The **I/O properties** dialog opens.
2. Go to the **Channels 0-7** tab.
   The terminal **X142** is represented on the tab, specifying the pin numbers and the current use of the channels.

3. Channels 0 and 1 on pins **3** and **4** are preset at the factory as digital inputs DI 0 and DI 1. Define the two channels as digital outputs DO 0 and DO 1. Select the value **DO** in each case in the **Function** list field.

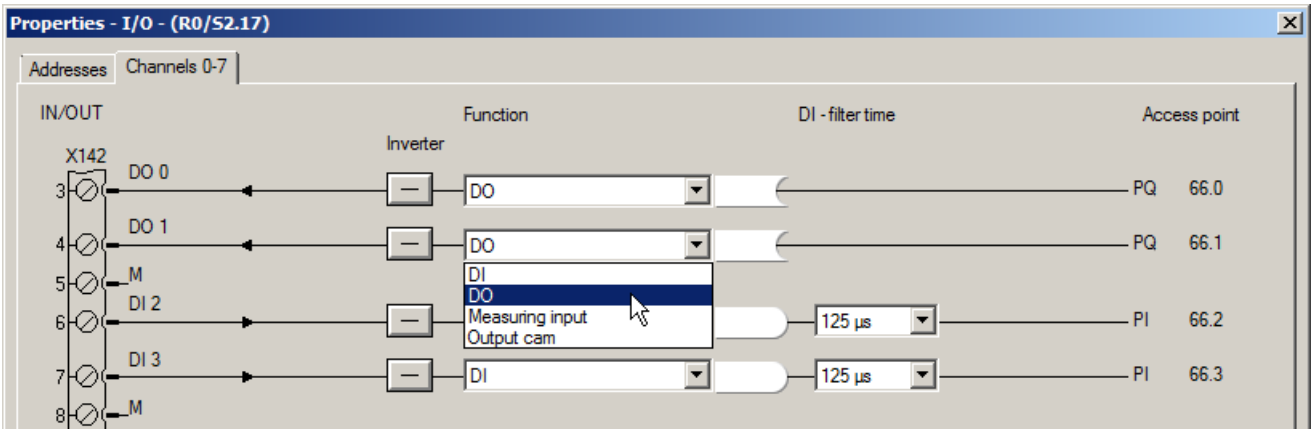


Figure 3-257 I/O properties

4. Confirm with **OK**.
Configuring the digital outputs is thus completed.

3.4.13 Programming the SIMOTION application

3.4.13.1 Overview

Aim of Getting Started

In this section, you write a simple SIMOTION user program for the sample project.

- You create the variables required by the program.
- You create the program and additional auxiliary programs with the graphical editors supplied by SIMOTION SCOUT TIA.

Purpose of the program

The program controls the axis set up in the sample project.

- The infeed is switched on via a SINAMICS system call.
- The axis is traversed to a specific position and returned to the original position.
- The program states "Program started" and "Program end reached" are applied to digital outputs to evaluate them in the I/O.

Requirements

- You have configured the infeed, see the section AUTOHOTSPOT.
- You have configured an axis and tested its function, see the sections AUTOHOTSPOT and AUTOHOTSPOT.
- You have prepared two digital outputs for use in the program, see the section AUTOHOTSPOT.
- SIMOTION SCOUT TIA is in offline mode.

3.4.13.2 Variables

Variable types

Variable types

Several variable types are distinguished in SIMOTION:

- **System variables**
  Each SIMOTION device and technology object has specific system tags. You can access system tags within the SIMOTION device from all programs.
- **I/O variables**
  An I/O tag is a symbolic tag name that is assigned to an I/O address of the SIMOTION device or to the I/O. Direct access to the I/O is thus possible.
  I/O variables are valid across all devices. All programs of the SIMOTION device have access to them.
- **Global device variables, unit variables, and local variables** are user-defined variables with a limited scope of validity:
  All programs of a SIMOTION device can access global device variables.
  All programs, function blocks, and functions that are defined within the same unit, e.g. ST source, MCC source, LAD/FBD source, can access unit variables. It is also possible to access variables of other units via the keyword USES.
  A unit (source) is a logical unit that you can create in your project and that can contain programs, functions, and function blocks.
  Local variables can only be accessed within the program, the function or the function block in which they are defined.

You can find further information on variables in the online help under **Variable model**.

Variables of the sample project

You require the following variables for the sample project of Getting Started:

2 global device variables

- g_bo_start
- g_bo_ready

2 I/O variables

- q_bo_output0
- q_bo_output1

Additionally when using an infeed with DRIVE-CLiQ interface:

2 I/O variables

- LineModule_STW
- LineModule_ZSW

Instance

- myFB_LineControl

Use of the variables in the sample project

The variables **g_bo_start** and **g_bo_ready** redundantly describe the program states "Program started" and "Program end reached".

The I/O variables **q_bo_output0** and **q_bo_output1** apply the program state to the configured digital outputs of the SIMOTION device, e.g. for a display.

The instance **myFB_LineControl** and the I/O variables **LineModule_STW** and **LineModule_ZSW** control the infeed.

Creating global device variables

You create global device variables in the **Symbol browser** tab of the detail view.

You create global device variables as follows

Create the global device variables **g_bo_start** and **g_bo_ready** for the sample project as follows.

The variable definition encompasses:

- Variable name
- BOOL data type

Note

You can only create global device variables in offline mode.

1. Click the **GLOBAL DEVICE VARIABLES** element in the project navigator under the SIMOTION device.
   The **Global device variables** table is displayed in the symbol browser.
2. Click in the **Name** column on the first free cell and enter the variable name **g_bo_start**. Press RETURN or TAB. The input focus jumps to the **Data type** field. Alternatively, you can click in the field to move the input focus there.
3. Enter the data type **BOOL** in the **Data type** field.

4. Press RETURN to confirm.
   The variable is created and is available in the project. A new empty line is opened for input in the symbol browser.
5. Create the global device variable **g_bo_ready** accordingly.

The figure below shows the full definition.

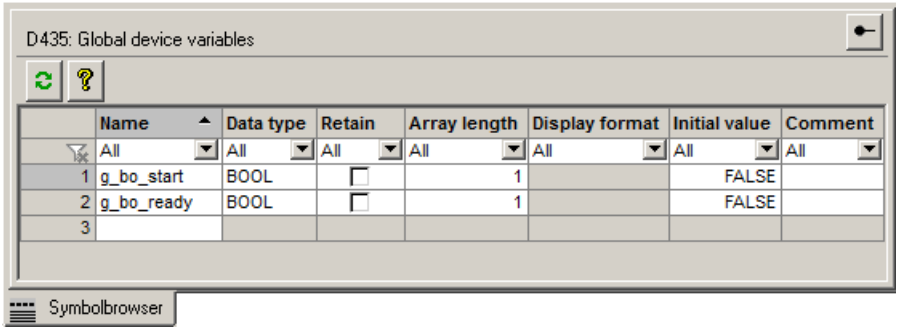


Figure 3-258 Global device variables of the sample project

Creating I/O variables

When configuring I/Os, SIMOTION SCOUT TIA supports the symbolic assignment of inputs and outputs located on SIMOTION/SINAMICS components. To be able to access the drive-level I/O with an I/O variable, it is only necessary to assign the I/O variable to the I/O channel. SIMOTION SCOUT TIA automatically sets up telegrams, BICO interconnections and addresses.

You create the I/O variables for output at the configured digital outputs as follows

You create the I/O variables **q_bo_output0** and **q_bo_output1** for the sample project as follows.

The variable declaration encompasses:

- Variable name
- Definition as output variable OUT
- BOOL data type
- Assignment to a digital output

Note

I/O variables can only be created in offline mode.

1. Double-click the element **ADDRESS LIST** in the project navigator below the SIMOTION device. The **Address list** tab opens in the detail view.
2. Click the first free cell in the **Name** column. Enter the variable name **q_bo_output0**. Press RETURN or TAB. The input focus jumps to the **I/O address** field. Alternatively, you can click in the field to move the input focus there.
3. Enter the keyword **OUT** in the **I/O address** field.
4. Enter the data type **BOOL** in the **Data type** field, or leave the field empty.

5. Assign the configured digital output to the variable:
   - Click the button ... in the **Assignment** cell. The Assignment dialog opens.
   - Open the **D435** element in the Assignment dialog.
   - Click the digital output **DO_0** below the **D435** element. **Assign** appears in the **Assignment** column.

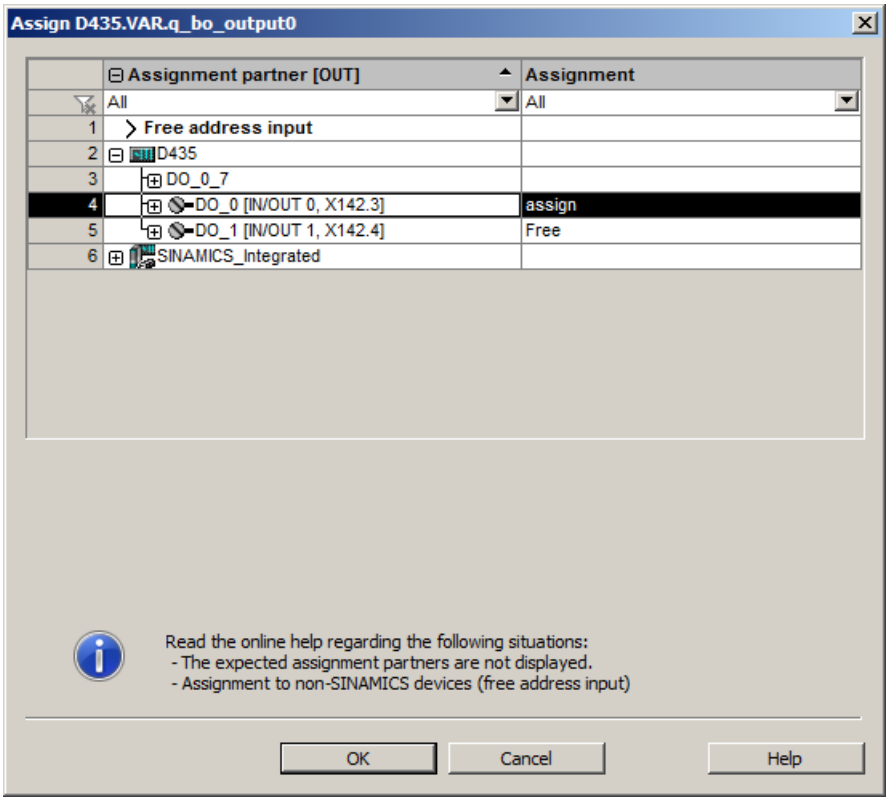


Figure 3-259 Assigning I/O variable and I/O channel

6. Confirm with **OK**.
   The assignment of I/O variable and I/O channel has been completed. The I/O variable is created and is available in the project. A new empty line is opened for input in the **Address list** table.

7. Create the I/O variable **q_bo_output1** in the same way. Assign the variable to the configured digital output DO 1.
   The figure below shows the full declaration:

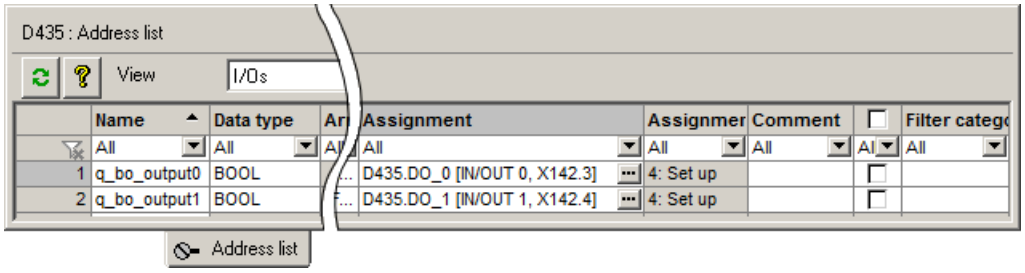


Figure 3-260 I/O variables of the sample project

8. Leave the **Address list** tab open for declaring further I/O variables.

You create the I/O variables for controlling the infeed as follows

If you are using an infeed with DRIVE-CLiQ interface, create also the I/O variables **LineModule_STW** and **LineModule_ZSW**.

Note

I/O variables can only be created in offline mode.

Proceed as for the definition of the I/O variables **q_bo_output0** and **q_bo_output1**.

1. Open the **Address list** tab in the detail area if not already visible.
2. Create the I/O variable **LineModule_STW**.
   - **I/O address**: OUT
   - **Data type**: WORD
   - **Assignment**: Control word E_STW1 of the SINAMICS infeed

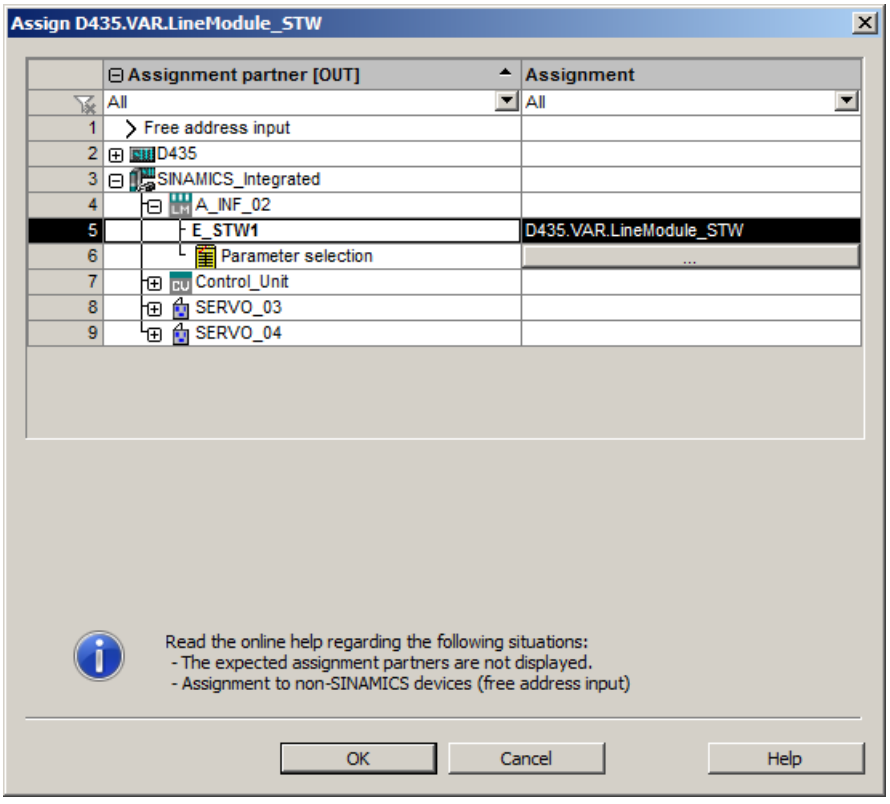


Figure 3-261 Assigning I/O variable and control word of the infeed

3. Create the I/O variable **LineModule_ZSW**.
   - **I/O address**: IN
   - **Data type**: WORD
   - **Assignment**: Status word E_ZSW1 of the SINAMICS infeed

The figure below shows the full declaration:

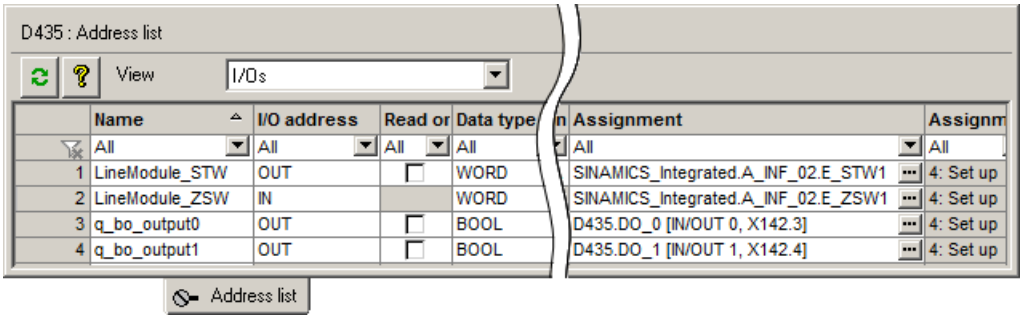
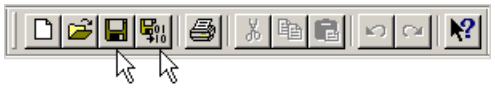


Figure 3-262 I/O variables of the sample project

Back up the configuration

Back up the variables created in the sample project.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.



3.4.13.3 Programming

Programming languages in the sample project

Within the scope of the sample project, you create a simple user program with the programming languages MCC and LAD/FBD.

For this, you use the variables you created in the previous section Variables (Page 513).

MCC Motion Control Chart

The programming language MCC

MCC (Motion Control Chart) is a graphical programming language.

The programmed motion sequences (machine sequences) are shown in the MCC as flowcharts (MCC charts). The structure of the flowcharts is oriented around the actual operating sequence of the machine. A sequential motion sequence is programmed.

System in the SIMOTION SCOUT TIA

The **PROGRAMS** folder under the SIMOTION device contains the MCC units created in the project.

An MCC unit contains the MCC programs that are to run on the SIMOTION device.

"MCC program" is a collective term for different program organization units (POE).

A POE can be a program, a function, or a function block. The type of the POE is indicated in the project navigator by an icon:

- Program
- Function
- Function block

An MCC chart is the graphical representation of a program organization unit POE.

An MCC unit can contain several MCC charts.

Program creation steps

Creation of an MCC program encompasses the following steps:

1. Creating the MCC unit.
2. Creating the MCC charts in the MCC unit.
3. Inserting MCC commands in the MCC chart and parameterizing the commands.

Creating the MCC unit

You create the MCC unit **motion** for the sample project as follows.

You create an MCC unit in the project as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert MCC unit**. The **Insert MCC unit** window appears.
2. Enter the name **motion**.
3. Go to the **Compiler** tab. For diagnostics purposes, activate the options **Permit program status** and **Permit single step**. In this way, you can monitor program execution later in online mode.
4. Confirm with **OK**.
   The MCC unit is created.
   - The unit appears in the project navigator under the PROGRAMS branch.
   - In the working area of the workbench, the declaration table of the unit opens. The variables declared there apply within the MCC unit and can be linked in other units.

   No other variable declaration is required for the sample project. You have already created the necessary variables as global device variables in the symbol browser.

Creating the MCC chart

You create the MCC chart **pos_axis** for the sample project as follows. The MCC chart **pos_axis** is a POE of the type program.

You create an MCC chart as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator.
2. Open the MCC unit **motion** in the **PROGRAMS** folder.
3. Double-click **Insert MCC chart**.
   The **Insert MCC Chart** window appears.
4. Enter the name **pos_axis**.
5. Select the creation type **Program**.
6. Confirm with **OK**.
   The MCC chart is created in the project.
   - The created MCC chart **pos_axis** appears in the **PROGRAMS** folder under the **motion** unit.
   - The MCC editor is opened in the working area of the workbench. The start and end nodes are already pre-defined. You can start MCC programming.

Inserting command blocks into an MCC chart

Every newly created MCC chart already contains a start and end node.

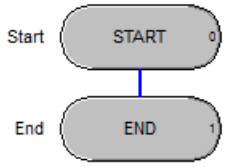


Figure 3-263 MCC chart, start and end block

You insert the MCC command blocks between these. The commands are processed in the direction from the start to the end node.

The MCC commands are available to you via:

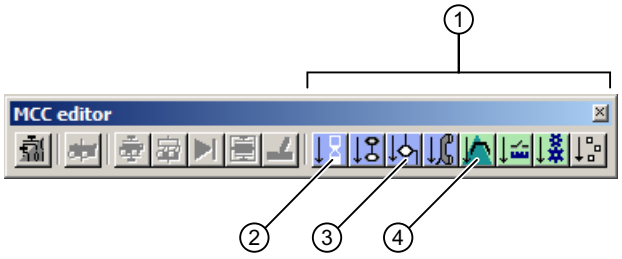
- The **MCC editor** toolbar
- **MCC Chart > Paste** menu command
- Context menu of the command block

You work with the MCC editor toolbar as follows

Open the toolbar

The **MCC editor** toolbar becomes visible in the workbench as soon as you open an MCC chart.

The commands are arranged into command groups. The sample project uses commands from the command groups **Basic commands**, **Program structures**, and **Single axis commands**.



① Command groups
② Basic commands
③ Program structures
④ Single axis commands

Figure 3-264 MCC editor toolbar

Note

If you do not see the toolbar, check that the display is switched on: Open the menu **View > Toolbars**. Activate the checkbox for **MCC editor** in the **Toolbars** window.

Open the command group

Move the mouse over the colored buttons of the toolbar to show the command groups.

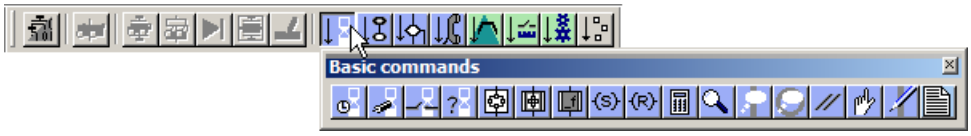


Figure 3-265 Open the command group

Keeping command groups open or closed continuously

Click the window title of a command group to keep the command group open continuously.

Select **Hide** in the context menu of a command group to close the command group.

Placing the command group as required

Drag the toolbar or the command groups of the toolbar with the mouse to any location on the workbench.

Docking the toolbar

Drag the toolbar or the command groups of the toolbar with the mouse to the edge areas of the workbench to dock them there.

Showing a tooltip for the command

Hold the mouse pointer briefly over a command button. The designation of the command is shown.

You insert commands into an MCC chart with the help of the MCC editor toolbar as follows

1. Click in the active MCC chart on the connecting line between two commands, or click the command after which the new command is to be inserted.
   The connecting line or the border of the command button is marked in blue. The marking flashes.
2. Select the command group in the **MCC editor** toolbar. Click the desired command in the command group.
   The command is inserted into the chart.

The newly inserted command is empty. It must then be parameterized; e.g. for a variable assignment, the name of the variable and the assigned value must be specified.

The following section Creating an MCC sample program: basic framework (Page 522) describes the procedure in detail.

3.4.13.4 Creating an MCC sample program: basic framework

Overview

In the sample project, you create the MCC program **pos_axis**. This program traverses an axis to a target position and back to the starting position.

Handling the infeed

If you are using an infeed without a DRIVE-CLIQ interface, the basic framework for program-controlled operation of the axis dealt with in this section is sufficient. The drive has a continuous ready signal of the infeed because the parameter is set to p0864=1, see the section AUTOHOTSPOT.

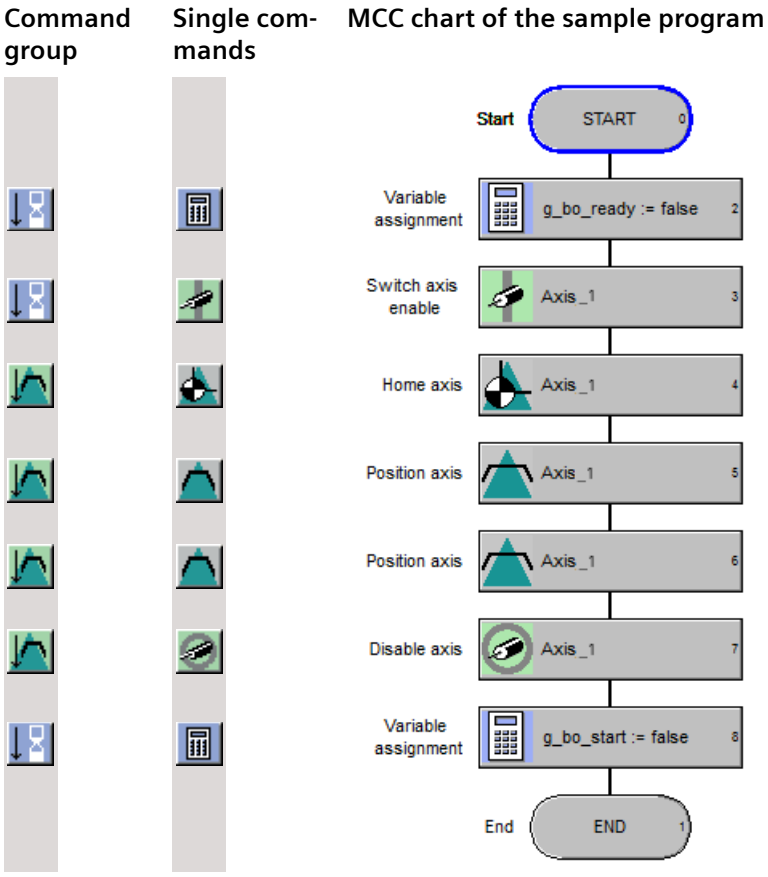
If you are using an infeed with a DRIVE-CLiQ interface, the program must switch the infeed on. Getting Started deals with this additional program sequence separately in the section Expanding the MCC sample program: control of the infeed (Page 535).

Program flow

The finished MCC chart

The figure below shows the finished MCC chart of the sample program.

The first two command blocks initialize the variables **g_bo_ready** and **g_bo_start**. The axis is then enabled, referenced, and traversed to the target position. When the position has been reached, the axis returns to the starting position. The enable is revoked. Finally, the variable **g_bo_start** is reset and the variable **g_bo_ready** is set.



Command groups and individual commands

In the figure above, the icons of the command group and of the individual command that you must click in sequence to insert the command into the MCC chart are shown in front of every MCC command.

The sections below describe the procedure in detail.

Variable assignment g_bo_ready:=false / g_bo_start:=true

The variable **g_bo_ready** is set to FALSE to set the variable to a defined starting point.

The variable **g_bo_start** is set to TRUE to indicate the start of the axis.

To assign the variables in the MCC editor, proceed as follows:

1. Mark the line between the start and end nodes.

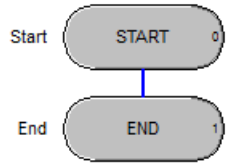


Figure 3-266 Empty MCC chart

2. Open the **Basic commands** command group in the **MCC editor** toolbar. Click the **Variable assignment** command in the command group.

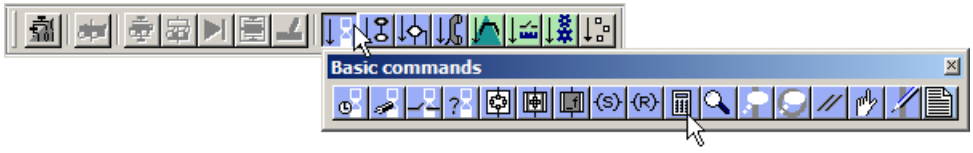


Figure 3-267 MCC editor toolbar, variable assignment command

The **Variable assignment** command block is inserted into the MCC chart after the start node.

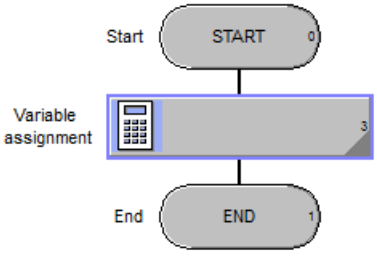


Figure 3-268 MCC chart with inserted variable assignment command block

3. Assign the value FALSE to the variable **g_bo_ready**. You transfer the variable name from the symbol browser to the command block:
   - Double-click the **Variable assignment** command block.
     The **Variable assignment** window appears.
   - Select the value **Formula** in the drop-down list.
     A table with the columns **Variables** and **Expression** appears.
   - Open the symbol browser. To do so, click **GLOBAL DEVICE VARIABLES** under the SIMOTION device in the project navigator.
   - Select the variable **g_bo_ready** in the symbol browser. To do so, click in the first column of the symbol browser on the preceding line number. The line is shown on a black background.

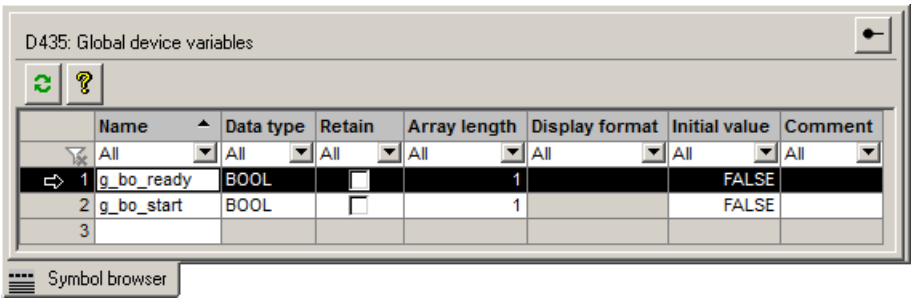


Figure 3-269 Symbol browser

   - Drag and drop the marked line to the **Variable** column of the **Variable assignment** window. The variable is inserted and its name displayed.
   - In the **Expression** column, enter the value **false** and confirm with RETURN.

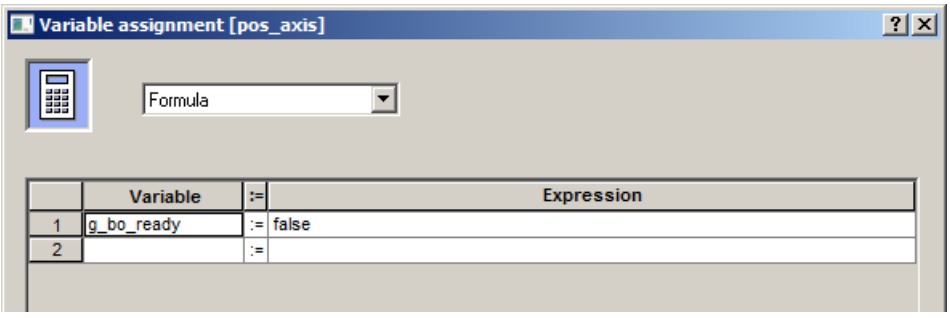


Figure 3-270 Variable assignment

4. Assign the value **true** to the variable **g_bo_ready** in the same way.

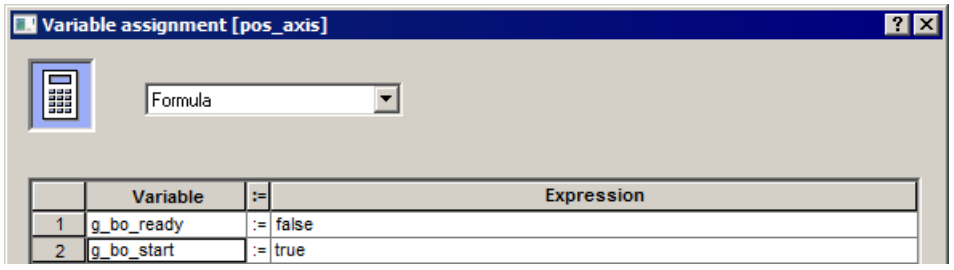


Figure 3-271 Variables assigned

5. Click **OK** to close the window.
   You have set the global device variables to the values **false** and **true**.

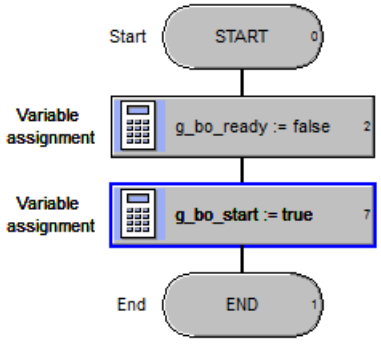


Figure 3-272 MCC chart, executed variable assignment

Variable assignment g_bo_start:=true

The variable **g_bo_start** is set to TRUE to indicate the start of the axis.

You assign the value TRUE to the variable g_bo_start as follows

1. Insert a second **Variable assignment** block into the MCC chart in front of the end node:
   - Click the command block to be followed by the new block.
     The node is marked in blue.
   - Select the **Basic commands > Variable assignment** command in the **MCC editor** toolbar.
     The **Variable assignment** command block is inserted into the MCC chart.
2. Assign the value TRUE to the variable **g_bo_start**. Use the Autocomplete function to select the variable:
   - Double-click on the empty **Variable assignment** command block in the MCC chart.
     The **Variable assignment** window appears.
   - Select the value **Formula** in the drop-down list.
   - Enter the initial characters of the variable name **g_bo_start** in the **Variable** column. Press the keys **Ctrl+space bar** for autocomplete. Select the variable from the list.

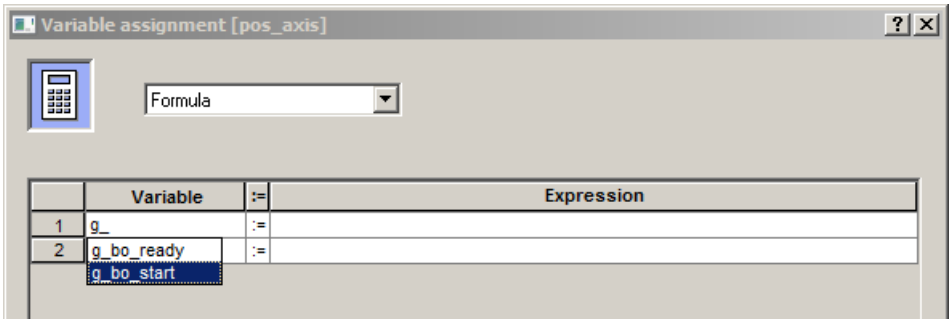


Figure 3-273 MCC chart, selection of the variable using the Autocomplete function

   - In the **Expression** column, enter the value **true** and confirm with RETURN.
3. Click **OK** to close the window.
   The command is parameterized.

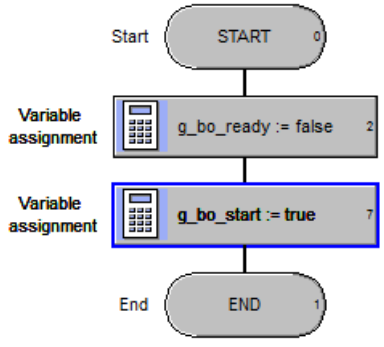


Figure 3-274 MCC chart, executed variable assignment

Switch axis enable

You switch the axis enable as follows

1. Insert the statement block **Switch axis enable** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Switch axis enable** command in the command group.

Figure 3-275 MCC editor toolbar, Switch axis enable command

The **Axis enable** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the axis enable:
   - Double-click the inserted **Switch axis enable** command block in the MCC chart. The **Switch axis enable** window is displayed.
   - In the **Axis** field, select the axis that is controlled by the program.

– Make sure that the **Delay program execution** option is selected for the sample project.
The command is only finished when the axis is enabled.
Leave the other parameters at the default values.

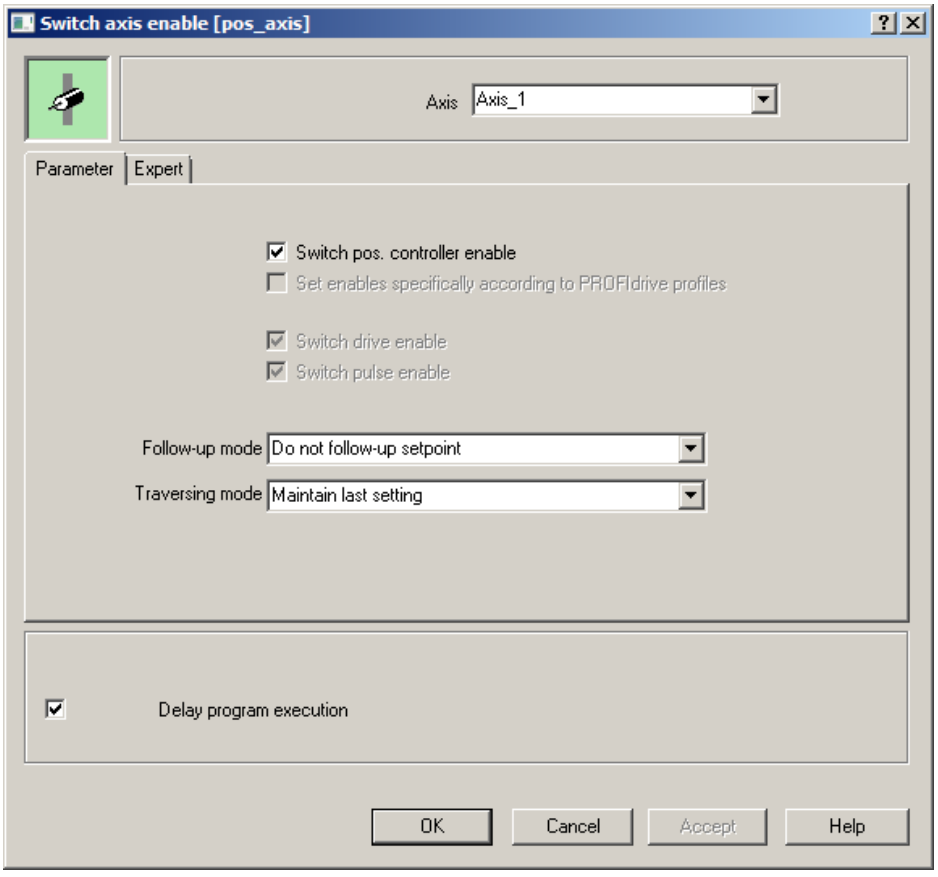


Figure 3-276 MCC chart, Switch axis enable

3. Click **OK** to close the window.
The command is parameterized.

Home axis

The position values of an axis are always in relation to the axis coordinate system.

The homing procedure defines where the zero point of the coordinate system is.

Homing of the axis is implemented as follows

1. Insert the statement block **Home axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the command **Home axis** in the command group.

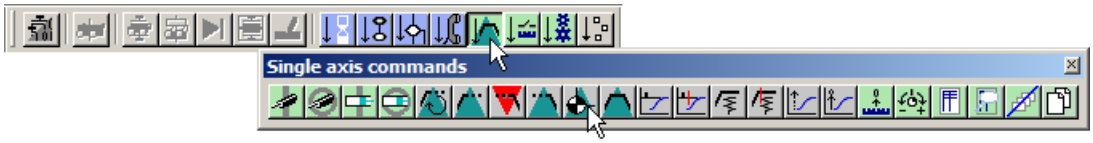


Figure 3-277 MCC editor toolbar, Home axis command

   The **Home axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize axis homing:
   - Double-click the inserted **Home axis** command block in the MCC chart. The **Home axis** window appears.
   - In the **Axis** field, select the axis that is controlled by the program.
   - In the **Parameters** tab, select the homing type **Set home position**. Enter the value **0** in the **Home position coordinates** field.
     With this homing type, the value of the home position coordinates are assigned to the current position of the axis (actual value). There is no active traversing motion.

– Leave the other parameters at the default values.

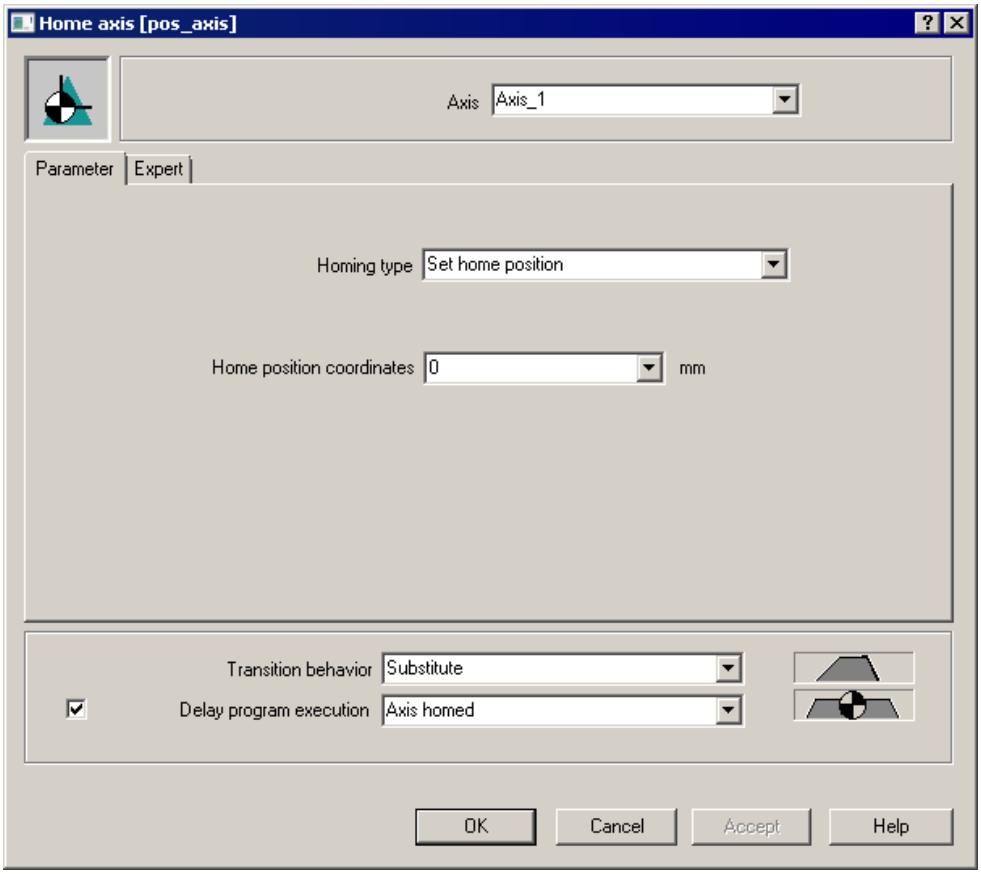


Figure 3-278 MCC chart, Home axis

3. Click **OK** to close the window.
   The command is parameterized.

Position the axis to the target position

The program is to traverse the axis at a velocity of 500 mm/s to the position 2000 mm.

You traverse the axis to the target position as follows

1. Insert the statement block **Position axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Position axis** command in the command group.

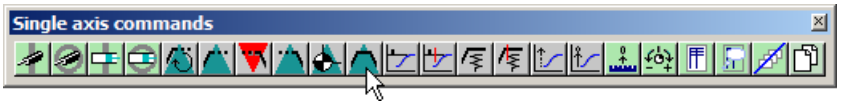


Figure 3-279 MCC editor toolbar, Position axis command

The **Position axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the axis motion:
   - Double-click the inserted **Position axis** command block in the MCC chart. The **Position axis** window appears.
   - In the **Axis** field, select the axis that is controlled by the program.
   - Enter the position 2000 in the **Parameters** tab. Set the velocity to 500 mm/s.
   - Leave the other parameters at the default values.

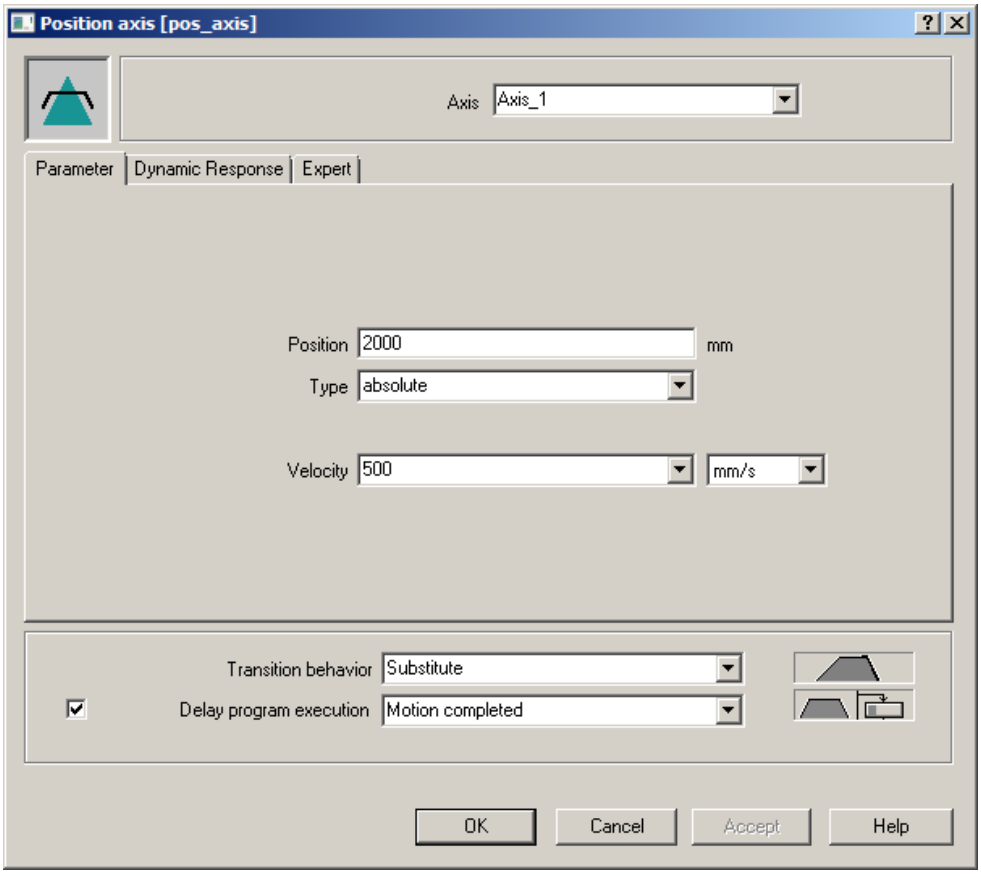


Figure 3-280 MCC chart, position the axis to the target position

3. Click **OK** to close the window.

The command is parameterized.

Position the axis to the starting position

The program is to return the axis to the starting position 0 mm at a velocity of 500 mm/s.

You return the axis to the starting position as follows

Proceed as for the step Position the axis to the target position (Page 531).

1. Insert the statement block **Position axis** into the MCC chart in front of the end node.
2. Parameterize the axis motion:
   - Enter the position 0 in the **Parameters** tab. Set the velocity to 500 mm/s.
   - Leave the other parameters at the default values.

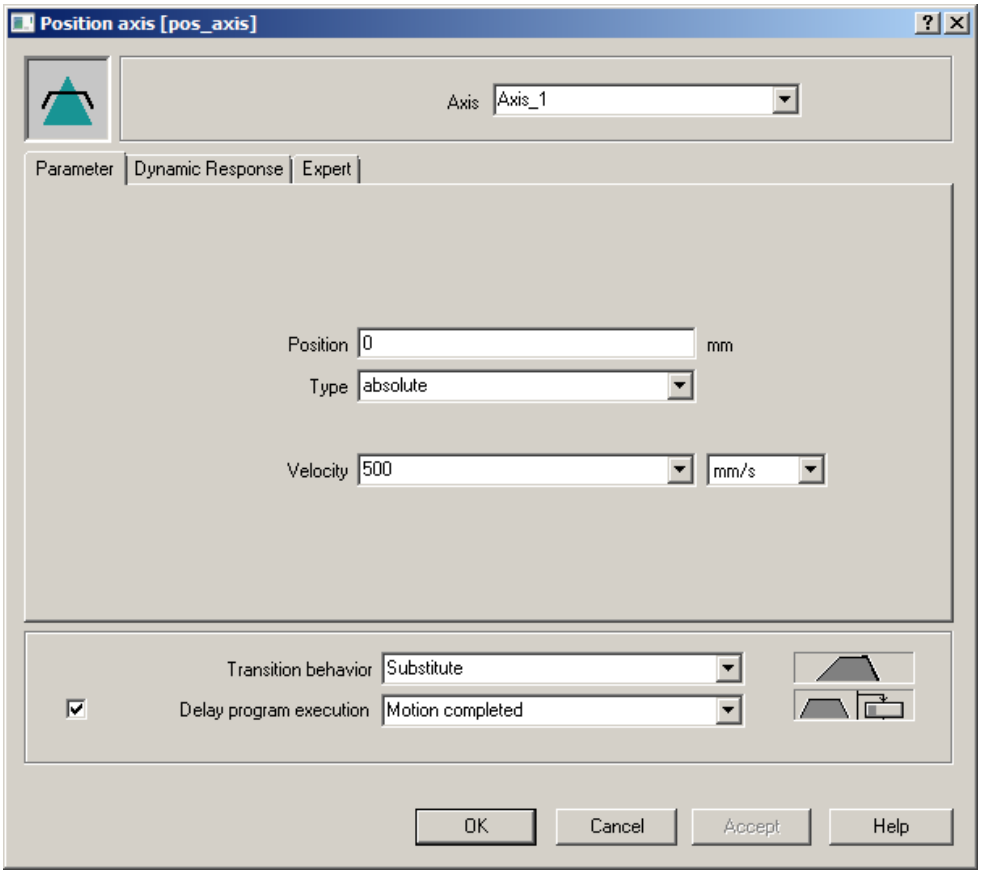


Figure 3-281 MCC chart, position the axis to the starting position

3. Click **OK** to close the window.
   The command is parameterized.

Disable axis

You disable the axis as follows

1. Insert the statement block **Disable axis** into the MCC chart in front of the end node:
   - Click the last block before the end node. The node is marked in blue.
   - Open the **Single axis commands** command group in the **MCC editor** toolbar. Click the **Disable axis** command in the command group.

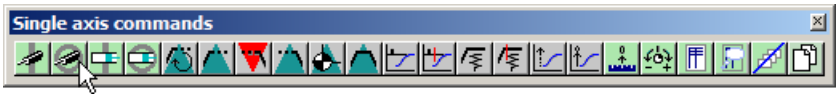


Figure 3-282 MCC editor toolbar, Disable axis command

The **Disable axis** command block is inserted into the MCC chart after the selected block and marked in blue.

2. Parameterize the command:
   - Double-click the inserted **Disable axis** command block in the MCC chart. The **Disable axis** window is displayed.

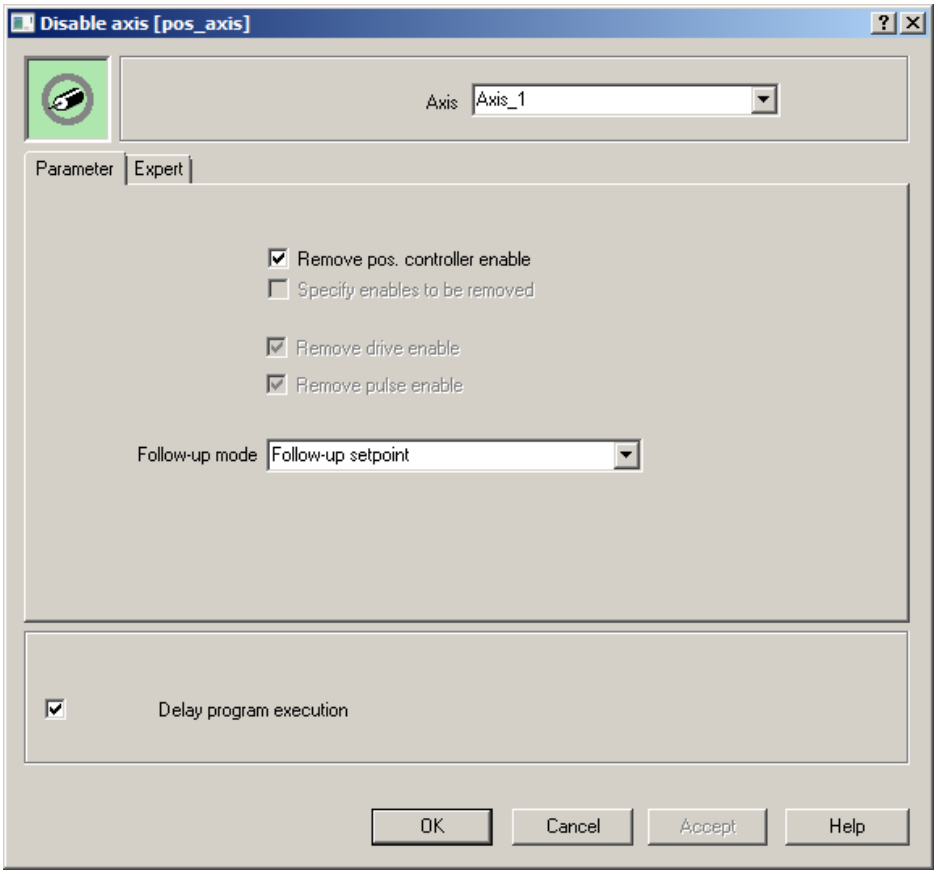


Figure 3-283 MCC chart, Disable axis

   - In the **Axis** field, select the axis that is controlled by the program.

– Leave the other parameters at the default values.

3. Click **OK** to close the window.
   The command is parameterized.

Variable assignment g_bo_start:=false / g_bo_ready:=true

The axis motion is finished. The variable **g_bo_start** is set to FALSE. The variable **g_bo_ready** is set to TRUE.

You assign the value FALSE to the variable g_bo_start as follows

Proceed as for the step Variable assignment g_bo_start:=true (Page 526).

1. Insert the statement block **Variable assignment** into the MCC chart in front of the end node.
2. Assign the value **false** to the variable **g_bo_start** and the value **true** to the variable **g_bo_ready**.
3. Click **OK** to close the window.
   The command is parameterized.

Variable assignment g_bo_ready:=true

The axis motion is finished. The variable **g_bo_ready** is set to TRUE.

You assign the value TRUE to the variable g_bo_ready as follows

Proceed as for the step Variable assignment g_bo_ready:=false / g_bo_start:=true (Page 523).

1. Insert the statement block **Variable assignment** into the MCC chart in front of the end node.
2. Assign the value **true** to the variable **g_bo_ready**.
3. Click **OK** to close the window.
   The command is parameterized.

3.4.13.5 Expanding the MCC sample program: control of the infeed

Program flow

If you are using an infeed with DRIVE-CLiQ interface, the programmed controller must switch on the infeed before the axis commands can then be issued. The program sequence shown below handles this task. The **pos_axis** program created until now must be expanded accordingly.

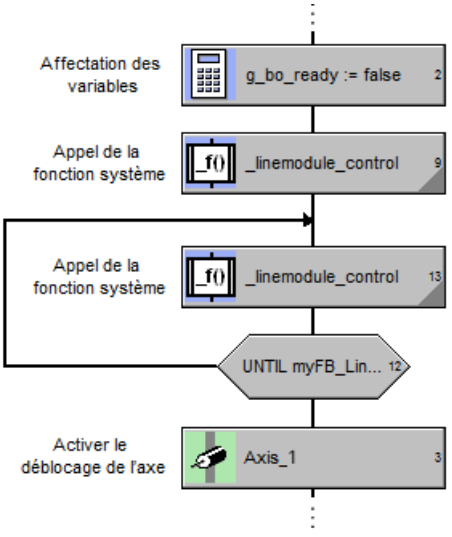
The finished MCC chart

The two times two blocks **_linemodule_control** system function call and **LineModule_STW** variable assignment are located in the MCC chart **pos_axis** before the first command for axis control **Switch axis enable**.

Command group

Single command

MCC chart of the sample program, control of the infeed



The first two blocks before the **UNTIL** loop reset the operating parameters of the infeed to the start values. Without this initialization, it may not be possible to switch the infeed on. In addition, this also acknowledges alarm messages that prevent enabling of the infeed.

The blocks within the **UNTIL** loop switch the infeed on. The loop repeats the switching command until the infeed is ready for operation.

Note

To avoid making excessive demands on the performance of RoundRobin execution level (Background Task, Motion Tasks) in infinite loops, it is better to use a _waitTime(T#0ms) in the loop for active waiting for an event."

System function call _LineModule_control[FB]

You create the system function call _LineModule_control[FB] as follows

1. Insert the statement block **System function call** into the MCC chart:
   - Click the command block **g_bo_start := true**.
     The node is marked in blue.
   - Select the **Basic commands > System function call** command in the **MCC editor** toolbar.

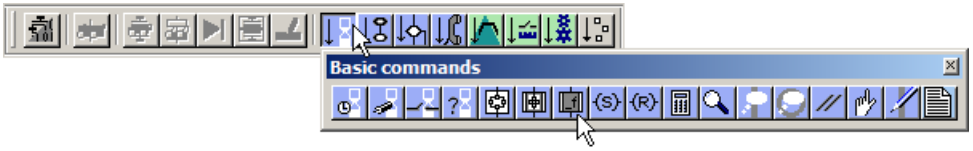


Figure 3-284 MCC editor toolbar, System function call command

The command block is inserted into the MCC chart.

2. Parameterize the function call:
   - Double-click the empty **System function call** command block in the MCC chart.
     The **System function call [pos_axis]** dialog appears.
   - Open the command library. You can find the command library in the screen area of the project navigator as a separate tab.

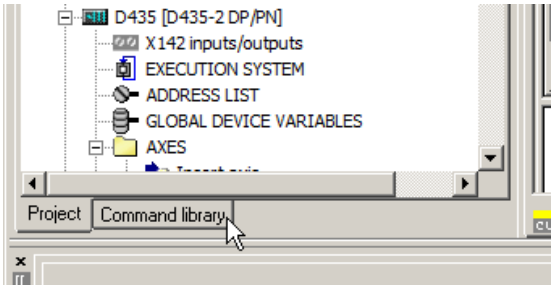


Figure 3-285 Open the command library

   - Go to the **Drives > SINAMICS** branch of the command library.
   - Drag & drop the **_LineModule_control[FB]** function call from the command library to the **System function** field of the **System function call[pos_axis]** window.

3. Define an instance for the system function call:
   – Enter the instance name **myFB_LineControl** in the **Instance** field of the **System function call [pos_axis]** window.
   As soon as you exit the field, e.g. by clicking on another element or with the TAB or RETURN keys, the **Variables declaration** dialog appears. The instance variable **myFB_LineControl** is assigned to the variable type **VAR** as default. The variable is thus valid locally within the program **pos_axis**.

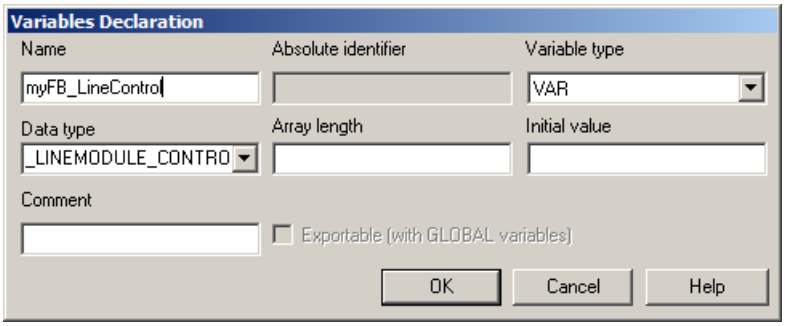


Figure 3-286 Variable declaration of the instance variable myFB_LineControl

Accept the default. Click **OK** to confirm the dialog.

– Set the input variables VAR_INPUT of the instance to the following values. Make use of the convenience of the autocomplete operator function **Ctrl+space bar**:

Variable name	Value	Meaning
reset	true	Reset the infeed
periIn	LineModule_ZSW	Status word of the infeed
typeLM	ACTIVE_LINE_MODULE SMART_LINE_MODULE BASIC_LINE_MODULE	Type of the module to be controlled depending on the infeed used
periOut	LineModule_STW	Control word, sequence control infeed

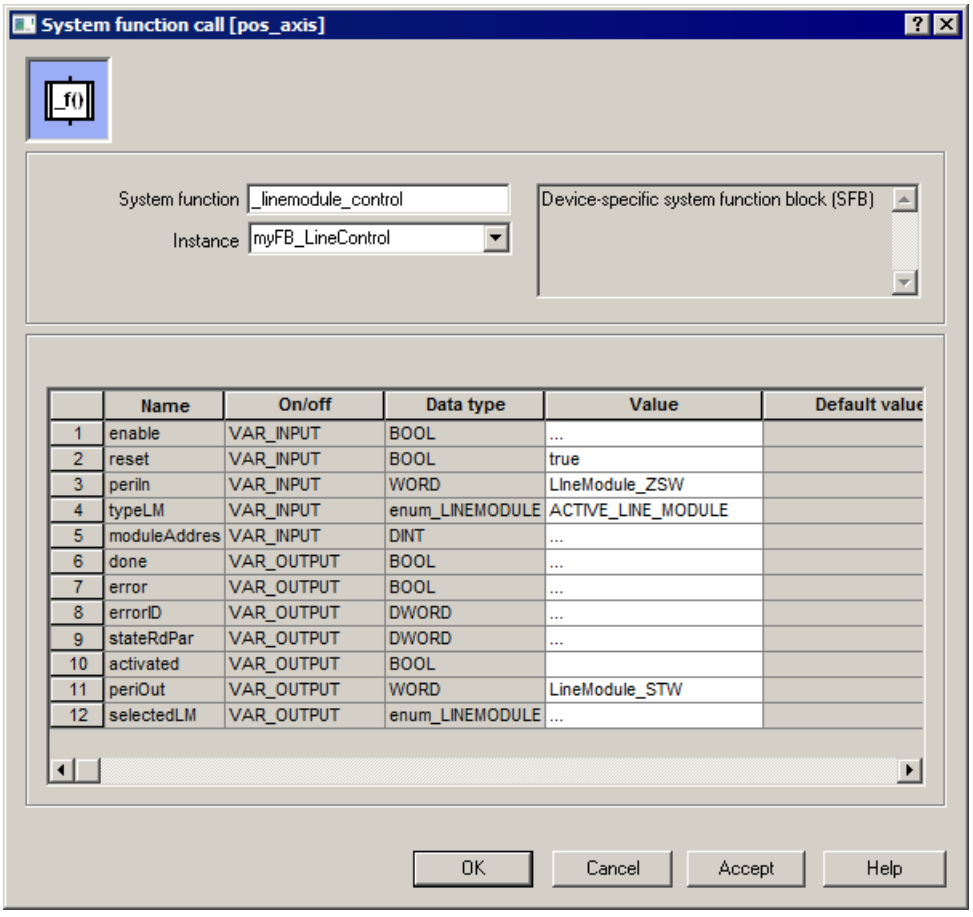


Figure 3-287 MCC chart, system function call _linemodule_control

4. Confirm with **OK**. The command is parameterized.

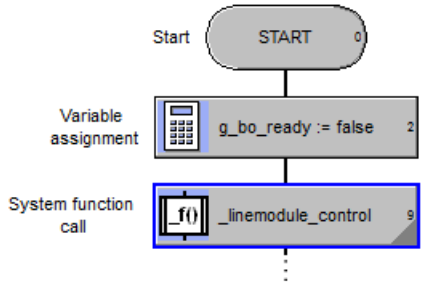


Figure 3-288 MCC chart, executed variable assignment _linemodule_control

Variable assignment LineModule_STW:=myFB_LineControl.periOut

You assign the myFB_LineControl.periOut variable to the LineModule_STW variable as follows

1. Create a **Variable assignment** block after the **System function call** block.
2. Assign the **myFB_LineControl.periOut** variable to the **LineModule_STW** I/O variable. You can use autocomplete or drag & drop in both **Variable** and **Expression** fields.

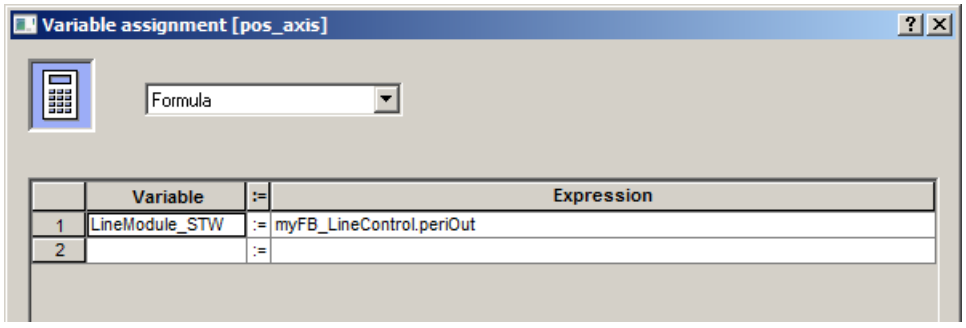


Figure 3-289 MCC chart, variable assignment LineModule_STW:=myFB_LineControl.periOut

3. Click **OK** to close the window.
   The command is parameterized.

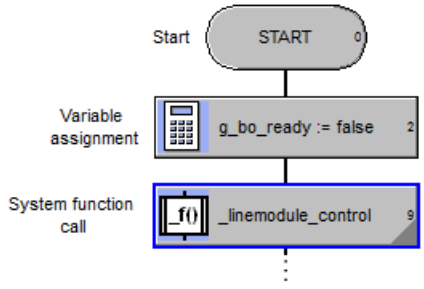


Figure 3-290 MCC chart, system function call and variable assignment

Create UNTIL loop

You create the UNTIL loop as follows

1. Click the connecting line between the blocks **System function call** and the subsequent axis command **Switch axis enable**.
   The reference point is marked in blue.
2. In the **MCC editor** toolbar, select the command **Program structures > UNTIL: Loop with condition at end**

Figure 3-291 MCC editor toolbar, UNTIL loop

The **UNTIL** block with loop is inserted.

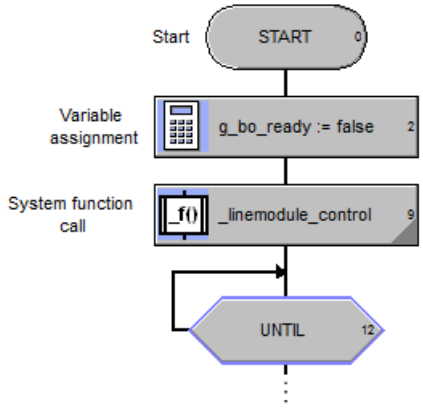


Figure 3-292 MCC editor, UNTIL loop

3. Parameterize the loop condition:
   The loop is to be run through until the infeed is ready for operation.
   - Double-click the UNTIL block. The window **UNTIL: Loop with condition at end[]pos_axis]** opens.
   - Select the value **Formula** in the field at top left.
   - Enter the loop condition in the **Until** field: **myFB_LineControl.activated=TRUE**.
     You can use the autocomplete function **Ctrl+space bar** in this field too.
4. Confirm with **OK**.
   The loop is created and parameterized in the program.

Copy blocks

You copy the statement blocks in the MCC chart as follows

Copy the statement blocks **System function call _LineModule_control[FB]** and **Variable assignment LineModule_STW:=myFB_LineControl.periOut** into the **UNTIL** loop. Use the operator functions **Copy** and **Paste** for this purpose.

1. Mark the blocks:
   - Hold the **Shift** key down.
   - Click the blocks one after the other. Both blocks are edged in blue.
2. Copy the selected blocks to the clipboard:
   - Open the context menu with the right mouse key.
   - Select the **Copy** command in the context menu.

   Or use the **Ctrl+C** key function.
3. Select the target of the copy operation in the MCC chart:
   - Click the connecting line within the **UNTIL** loop. The connecting piece is marked in blue.

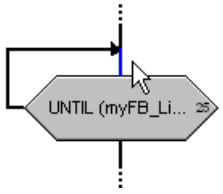


Figure 3-293 MCC chart, mark the UNTIL loop internal area

4. Insert the copied blocks at the target position:
   - Open the context menu again with the right mouse key.
   - Select the **Paste** command in the context menu.

   Or use the **Ctrl+V** key function.
   The blocks are inserted within the loop.

Adapt the system function call _linemodule_control

You change the system function call _linemodule_control as follows

The system function call **_linemodule_control** within the **UNTIL** loop switches the infeed on.

1. Open the system function block by double-clicking.
2. Adapt the parameter variables of the **myFB_LineControl** instance as shown below:

Variable name	Value	Meaning
enable	true	Switches the infeed on
reset		Empty field. The variable is thus set to **false**
periIn	LineModule_ZSW	Status word of the infeed
typeLM	ACTIVE_LINE_MODULE SMART_LINE_MODULE BASIC_LINE_MODULE	Type of the module to be controlled depending on the infeed used

3. Confirm with **OK**.

3.4.13.6 Create additional MCC programs for the sample project

Handling system events

The execution system that you will set up in the later section Configure execution system (Page 552) requires two further MCC programs:

- technology_fault
- peripheral_fault

These programs are used for handling system events. Handling is generally necessary. If a system event occurs that is not handled by the user program, the CPU goes to the STOP mode.

Within the scope of the sample project, no specific handling of system events is necessary. The programs technology_fault and peripheral_fault therefore remain empty.

You create the MCC programs technology_fault and peripheral_fault as follows

1. Create the MCC unit **fault**.
   Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert MCC unit**. Assign the name **fault** to the MCC unit.
2. Create the MCC chart **technology_fault**.
   Open the MCC unit **fault** in the **PROGRAMS** folder. Double-click **Insert MCC chart**. Assign the name **technology_fault** to the MCC chart. Select the creation type **Program**.
   The program remains empty.

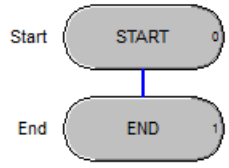


Figure 3-294 Empty program technology_fault

3. Create the MCC chart **peripheral_fault** within the MCC unit **fault**.
   This program also remains empty.

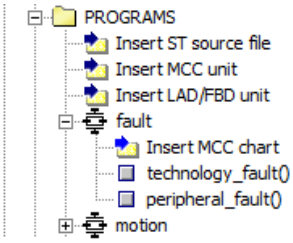


Figure 3-295 MCC programs technology_fault and peripheral_fault in the MCC unit fault

3.4.13.7 Back up MCC sample programs

Back up the created MCC programs.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.

Additional information

As an alternative to the command **Save project and compile changes**, you can find the command **Accept and compile** in the **MCC editor** toolbar.

This command compiles the currently selected program as well as all other programs of the same unit.

However, the command does not save the changes.

You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

3.4.13.8 LAD/FBD ladder logic/function block diagram

The LAD and FBD programming languages

LAD (ladder logic)

LAD stands for ladder logic. LAD is a graphical programming language. The syntax for the instructions is similar to a circuit diagram. LAD enables simple tracking of the signal flow between conductor bars via inputs, outputs and operations. The LAD statements comprise elements and blocks that are connected graphically to form networks. LAD operations follow the rules of Boolean logic.

FBD (function block diagram)

FBD stands for function block diagram. FBD is a graphical programming language. To represent the logic relationships, it uses the logic boxes familiar from Boolean algebra. In addition, complex functions (e.g. mathematical functions) can be represented directly in conjunction with the logic boxes.

System in SIMOTION SCOUT

The **PROGRAMS** folder under the SIMOTION device contains the LAD/FBD units created in the project.

A LAD/FBD unit contains the LAD/FBD programs that are to run on the SIMOTION device. A LAD/FBD unit can contain several LAD/FBD programs.

"LAD/FBD program" is a collective term for different program organization units (POE).

A POE can be a program, a function, or a function block. The type of the POE is indicated in the project navigator by an icon:

- Program
- Function
- Function block

Program creation steps

Creation of a LAD/FBD program encompasses the following steps:

1. Creating a LAD/FBD unit.
2. Creating a LAD/FBD program in the LAD/FBD unit.
3. Inserting LAD/FBD commands in the LAD/FBD program and parameterizing the commands.
4. Saving and compiling the LAD/FBD program.

Switching the programming language

SIMOTION SCOUT TIA allows simple switching between ladder logic and function block diagram. The LAD/FBD editor contains the command **LAD/FBD program > Switch to FBD** or **Switch to LAD**.

Create LAD/FBD unit

You create the LAD/FBD unit **bg_out** for the sample project as follows.

You create a LAD/FBD unit in the project as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator. Double-click **Insert LAD/FBD unit**. The **Insert LAD/FBD unit** window appears.
2. Enter the name **bg_out** for the unit.
3. Go to the **Compiler** tab. For diagnostics purposes, activate the option **Permit program status**. In this way, you can monitor program execution later in online mode.
4. Confirm with **OK**.
   The LAD/FBD unit is created.
   – The LAD/FBD unit **bg_out** appears in the **PROGRAMS** folder.

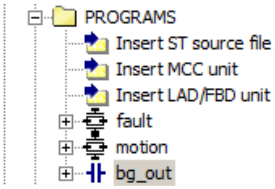


Figure 3-296 MCC and LAD/FBD units in the PROGRAMS folder

   – In the working area of the workbench, the declaration table of the unit opens. The variables declared there apply within the LAD/FBD unit and can be linked in other units. No other variable declaration is required for the sample project. You have already created the necessary variables as global device variables in the symbol browser.

Create LAD/FBD program

You create the LAD/FBD program **LAD_1** within the LAD/FBD unit **bg_out** as follows.

You create a LAD/FBD program in a LAD/FBD unit as follows

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator.
2. Open the LAD/FBD unit **bg_out** in the **PROGRAMS** folder. Double-click **Insert LAD/FBD program**. The **Insert LAD/FBD program** window appears.
3. Enter the name **LAD_1**. The name must be unique throughout the project.

4. Select the creation type **Program**.

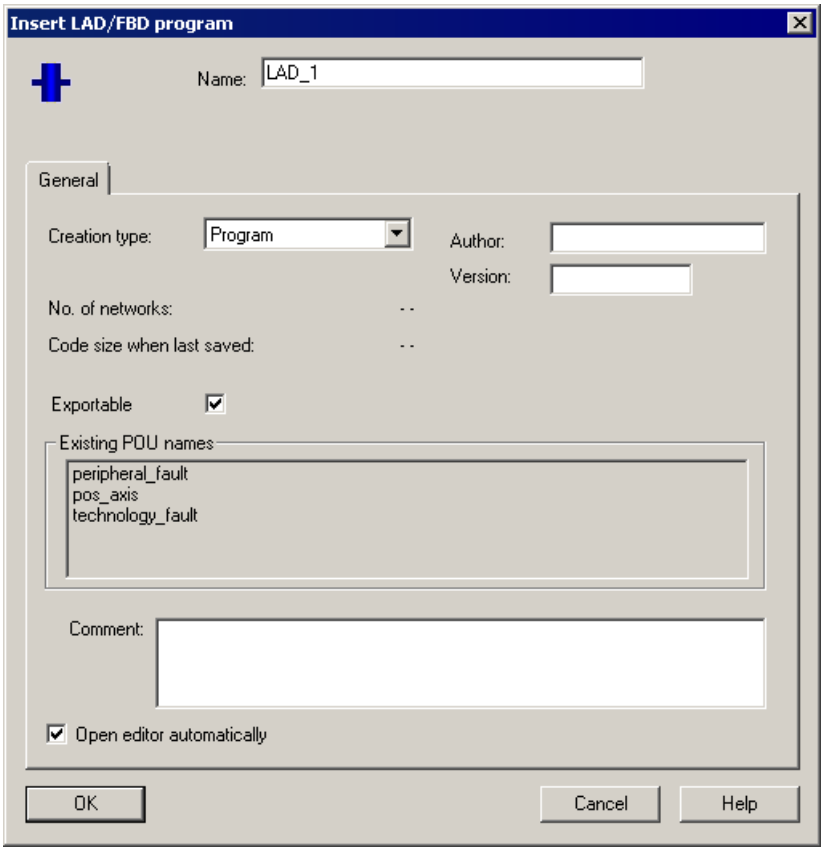


Figure 3-297 Insert LAD/FBD program

5. Confirm with **OK**.
   The LAD/FBD program **LAD_1** is created in the project.
   – The LAD/FBD program appears in the **PROGRAMS** folder.

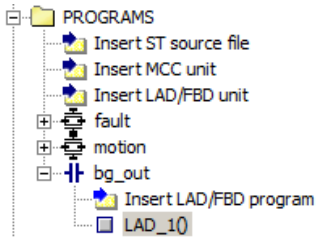


Figure 3-298 LAD program LAD_1 in the PROGRAMS folder

   – The LAD/FBD editor is opened in the working area of the workbench. You can start programming.

Creating a LAD sample program

You write the LAD program **LAD_1** for the sample project.

In this program, the global device variables **g_bo_start** and **g_bo_ready** are read cyclically, and the I/O variables **q_bo_output0** and **q_bo_output1** are set accordingly.

Transfer of the program status to the digital outputs DO 0 and DO 1 could also be programmed direct in the MCC chart **pos_axis**. However, Getting Started implements the assignment in an autonomous LAD program to introduce you to LAD programming.

You create the LAD program LAD_1 for the sample project as follows.

1. Open the LAD/FBD program if it is not already visible in the working area of the workbench.
   For this purpose, double-click the LAD/FBD program under the LAD/FBD unit in the **PROGRAMS** folder of the project navigator.
   In the sample program, the empty LAD/FBD program **LAD_1** is already open.

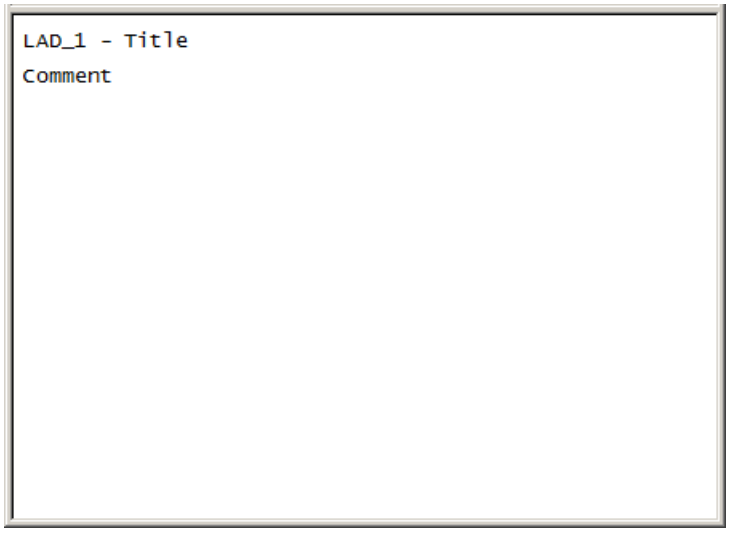


Figure 3-299 Empty LAD program following first-time creation in the LAD/FBD unit

2. Insert a network:
   – Click **Insert network** in the LAD/FBD toolbar.

   The **001** block with a coil is inserted.

3. Insert an NO contact in front of the coil:
   – Select the coil.

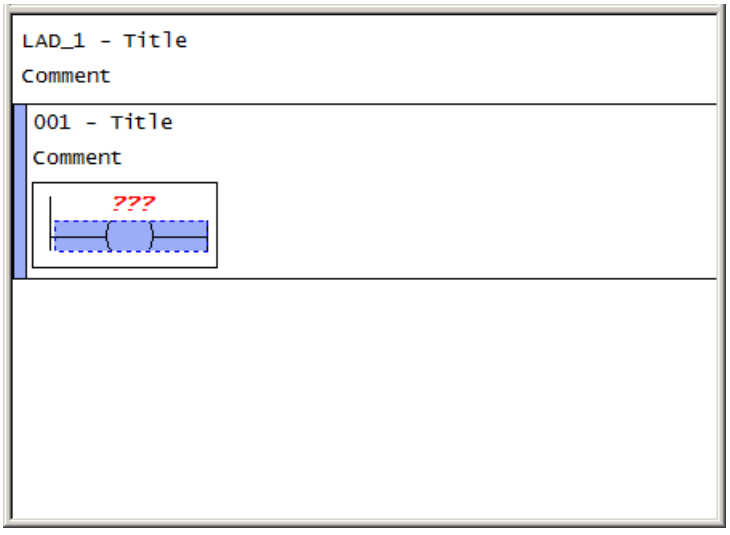


Figure 3-300 LAD program, selected coil

   – Click **NO Contact** in the toolbar.

   The NO contact is inserted in front of the coil.

4. Assign the **g_bo_start** variable to the NO contact and the **q_bo_output0** variable to the coil:
   – Click above the symbols for NO contact and coil on **???**.

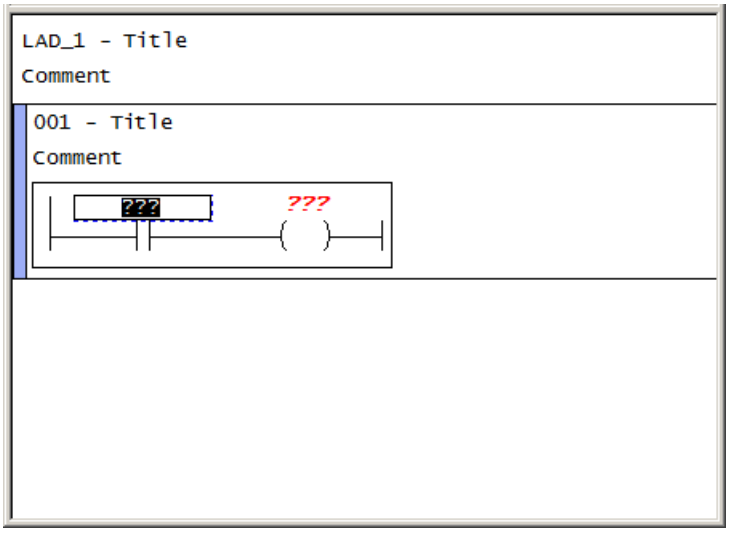


Figure 3-301 LAD program, opened field for entering the variable name

   – Transfer the variable names from the symbol browser or the ADDRESS LIST by dragging and dropping. Or use the Autocomplete function. Use RETURN to confirm each entry.

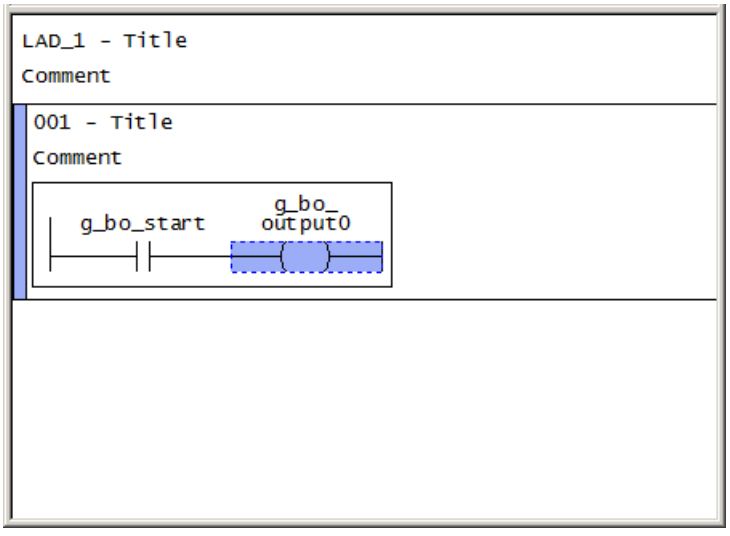


Figure 3-302 LAD program with inserted variables

5. Insert a network **002**:
   – Click the block **001** to select the block.
   A selected block has a blue background on the left edge.
   – Click **Insert network** in the LAD/FBD toolbar.
   The block **002** is inserted after the block **001**.

6. Insert an NO contact in front of the coil in network **002**.
7. Assign the **g_bo_ready** variable to the NO contact and the **q_bo_output1** variable to the coil. The LAD program is finished.

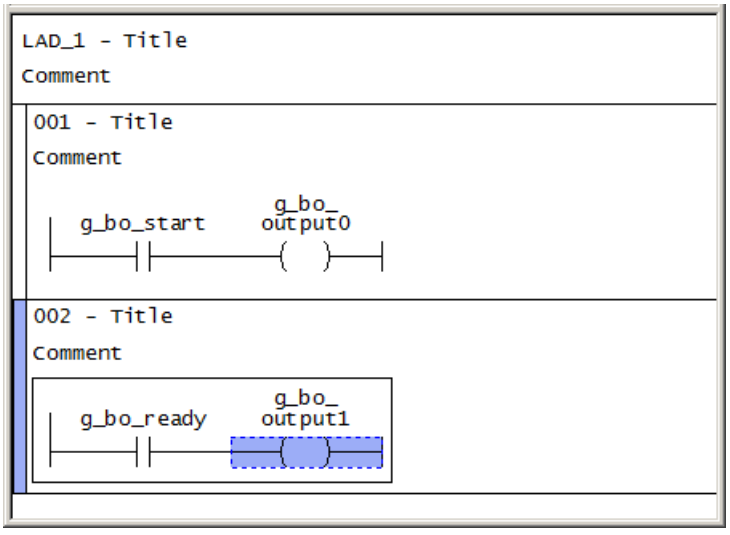


Figure 3-303 Finished LAD program LAD_1

Back up LAD/FBD sample program

Back up the created LAD/FBD sample program.

To do so, click in the toolbar on the **Save project** or **Save project and compile changes** button.

Additional information

As an alternative to the command **Save project and compile changes**, you can find the command **Accept and compile** in the **LAD/FBD toolbar**.

This command compiles the currently selected program as well as all other programs of the same unit.

However, the command does not save the changes.

You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

3.4.13.9 Further programming options

SIMOTION SCOUT TIA provides other programming languages in addition to MCC and LAD/FBD.

- **ST (Structured Text)**
  ST is a text-based, PASCAL-based programming language. The language is based on the international standard IEC 61131-3. This standard harmonizes the programming languages for programmable logic controllers (PLC). ST is based on the Structured Text part of this standard. In addition to the standardized programming language in accordance with IEC 61131-3, SIMOTION ST features technological commands.
  An easy-to-use text editor is provided for creating programs. The ST compiler compiles the edited program into executable code and indicates every syntax error, specifying the program line and the cause of the error.
  In the Command library tab of the project navigator, the commands and functions required for programming are shown in a tree. You can use these in all programming languages, e.g. when programming.

  Note

  In SIMOTION SCOUT V4.5 and higher, ST can also be used as a language for object-oriented programming. Further information can be found in the SIMOTION SCOUT Online Help and in the SIMOTION ST Structured Text Programming and Operating Manual.

- **Libraries**
  Libraries allow modular software development and provide you with user-defined data types, functions, and function blocks that you can use from all SIMOTION devices.
  Libraries can be written in all programming languages and used in all program sources (e.g. ST units, MCC units).

All programming languages enable you to structure the application using programs, functions and function blocks so that the application is manageable and re-usable.

To test the programs, there are comprehensive test functions available to you with program status and breakpoints in all languages. You can visualize and test your programs online.

For further information on programming languages, please refer to the respective manuals.

3.4.13.10 Result in the sample project

You have created the programs required for program-controlled traversing of the axis in the sample project of Getting Started.

3.4.14 Configure execution system

3.4.14.1 Overview

Aim of Getting Started

In this part of Getting Started, you get to know the SIMOTION execution system.

You assign the sample project programs to the tasks of the execution system. You thus transfer the programs from the SIMOTION SCOUT TIA engineering system to the SIMOTION runtime system. The programs control the system as soon as the operating mode of the SIMOTION device is switched to RUN.

Requirements

You have created the programs pos_axis, technology_fault, peripheral_fault, and LAD_1 for the sample project, see the section Programming the SIMOTION application (Page 512).

Execution levels and tasks

Execution levels define the chronological sequence of tasks in the execution system. A level can contain several tasks. The tasks provide the framework for program execution. A task may comprise several programs. By assigning the created programs to the tasks, you can, for example, define the priority, the time frame or the order in which the programs are to be executed.

3.4.14.2 Assign programs to tasks

Below you will assign the sample project programs to the tasks of the execution system.

Table 3-37 Assignment of programs to tasks

Programs of the sample project	Tasks
pos_axis	MotionTask_1
lad_1	BackgroundTask
technology_fault	TechnologicalFaultTask
peripheral_fault	PeripheralFaultTask

You assign programs to tasks in the execution system as follows

1. Double-click in the project navigator under the SIMOTION device on **EXECUTION SYSTEM**.
   The **EXECUTION SYSTEM** window appears in the working area.
2. Assign the MCC program **motion.pos_axis** to the task **MotionTask_1**:
   – In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > MotionTasks > MotionTask_1**.

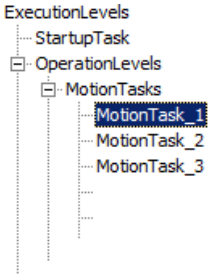


Figure 3-304 Execution system, tree view of the execution levels and tasks

The **MotionTasks** window appears on the right of the working area. The programs **pos_axis** and **LAD_1** as well as auxiliary programs of the **fault** unit are visible on the **Program assignment** tab under **Programs**.

– Select the MCC program **motion.pos_axis** and click the **>>** button. The program is displayed under **Programs used**. It is thus assigned to the task **MotionTask_1**.

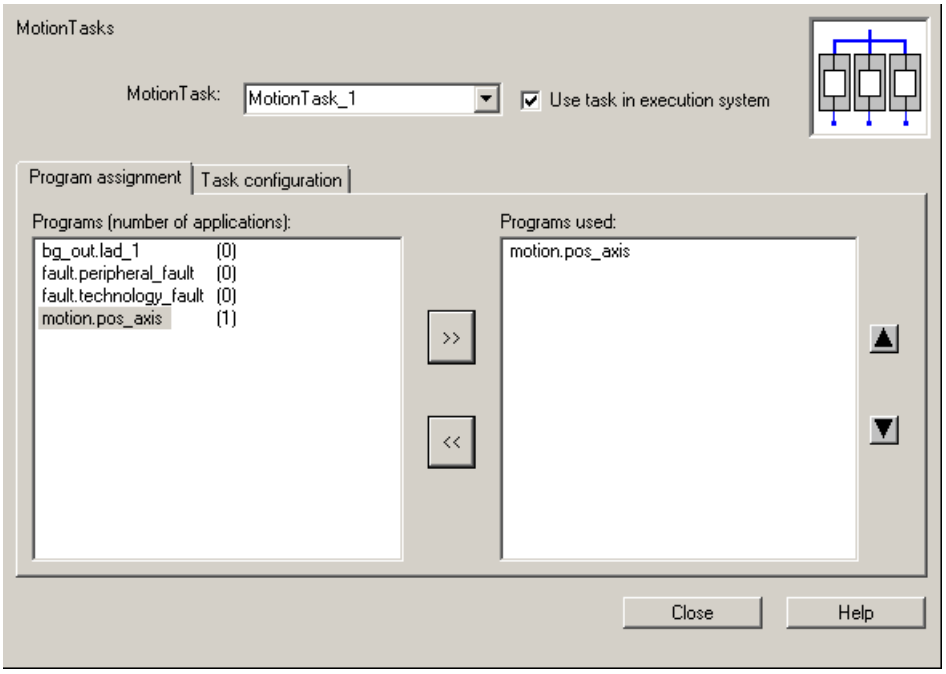


Figure 3-305 Execution system, MotionTasks window

The assignment is visible in the tree of the execution system. The program **motion.pos_axis** appears below the branch **MotionTask_1**.

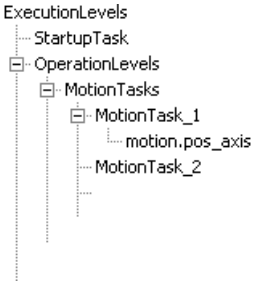


Figure 3-306 Execution system, MotionTask_1 with assigned program pos_axis

– Activate the checkbox **Activation after StartupTask** in the **Task configuration** tab. This executes the MCC program immediately after the SIMOTION device is started. If this checkbox is not activated, the program must be started explicitly by the call from another program that is assigned to the StartupTask or another active task.

3. Assign the LAD program **bg_out.kop_1** to the task **BackgroundTask**:

   – In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > BackgroundTask**. Assign the LAD program **bg_out.kop_1** to this task.

4. Assign the error handling routines:
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > TechnologicalFaultTask**. Assign the MCC program **fault.technology_fault** to this task.
   - In the tree of the execution system, select the branch **ExecutionLevels > OperationLevels > SystemInterruptTasks > PeripheralFaultTask**. Assign the MCC program **fault.peripheral_fault** to this task.
5. Click **Close**. Confirm with **Yes** if you are prompted to save.
   The execution system is configured.

Loading the configured execution system to the target system

Save and compile the project. Go online. Download the sample project with the configured execution system to the target system.

3.4.14.3 Download the configured execution system to the target system

Download the sample project with the configured execution system to the target system.

Steps

1. Save project and compile changes
2. Establish the online connection
3. Download the project to the target system

Save and compile changes

Click in the toolbar on the **Save project and compile changes** button.

Establish the online connection

Click in the toolbar on the **Connect to selected target devices** button.

Download to target system

Click in the toolbar on the **Download project to target system** button.

3.4.14.4 Result in the sample project

Configuring the axis control is thus completed:

- You have set up an axis in the sample project.
- You have created a program for traversing an axis, as well as other programs that are necessary for operation.
- You have assigned these programs to the tasks of the SIMOTION runtime system.

In the following configuring steps, you will start the axis control on the SIMOTION device and monitor the program-controlled axis motion.

3.4.15 Starting and stopping the system

3.4.15.1 Overview

Aim of Getting Started

You switch the SIMOTION CPU of the sample project to the RUN mode to start execution of the **pos_axis** program. You can see on the hardware that the axis rotates twice for approximately 4 seconds. After execution of the program, switch back to the STOP state.

You switch the operating mode in the dialog **Control Operating Mode**.

Requirements

- You have created the programs pos_axis, technology_fault, peripheral_fault and LAD_1 for the sample project, and you have assigned them to the tasks of the SIMOTION execution system, see the sections Programming the SIMOTION application (Page 512) and Configure execution system (Page 552).
- The project has been compiled and downloaded to the target system, see the section Download the configured execution system to the target system (Page 556).
- SIMOTION SCOUT TIA is in online mode.

3.4.15.2 RUN and STOP operating modes

Operating modes

In the SIMOTION SCOUT TIA dialog **Control Operating Mode**, you can switch a SIMOTION CPU to the RUN or STOP mode.

RUN mode

SIMOTION executes the user program and the associated system services:

- Read process image input
- Execution of the user programs assigned to the execution system
- Write process image output

The technology packages are active in this state. They can execute commands from the user program.

STOP mode

SIMOTION does not process any user program.

- It is possible to load a complete user program.
- All system services (communication, etc.) are active.
- The I/O modules are in a secure state. (that is, for example, the digital outputs are at "LOW level" and the analog outputs are de-energized)
- The technology packages are inactive, that is, all enables are deleted. No axis motions can be executed.

You can find the full description of the operating states in the online help under **SIMOTION device: Operating mode**.

3.4.15.3 Mode selector switch on the software side and the hardware side

Control Operating Mode dialog

The menu command **Target system > Control Operating Mode** or the button **Control Operating Mode** in the toolbar of the workbench open the dialog **Control Operating Mode**.

The dialog lists all configured CPUs. The **State** column shows the current operating mode. The assigned switches in the **Control** column switch the CPU to the RUN or STOP mode. The switches are deactivated when the CPU is in offline mode.

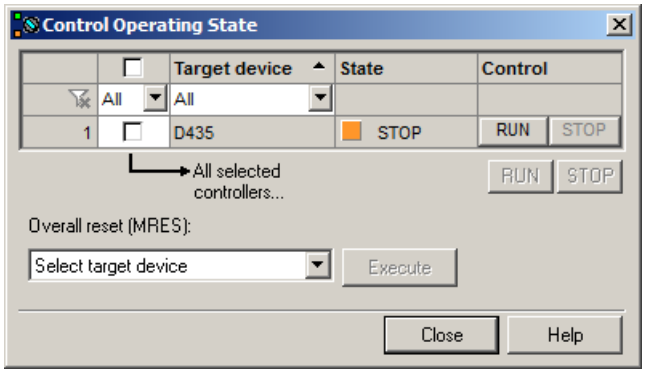


Figure 3-307 Control Operating Mode dialog

Priority of the mode selector switch on the SIMOTION device

The setting of the mode selector switch on the SIMOTION device has priority. SIMOTION SCOUT TIA can switch a SIMOTION device to the RUN operating mode only if the mode selector on the device is set to 0 (RUN).

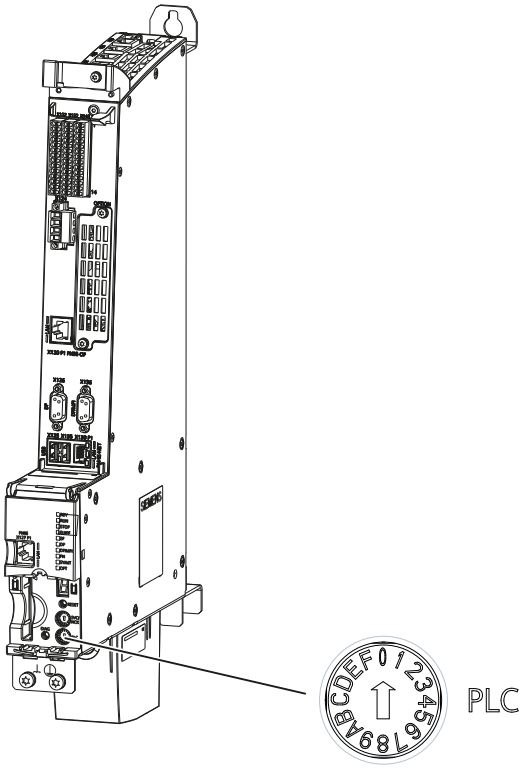


Figure 3-308 Mode selector switch of the SIMOTION D435-2, switch position 0 (RUN)

You can find detailed information on the mode selector switch of the SIMOTION D4x5-2 devices in the Manuals and Commissioning Manuals, as well as in the online help.

3.4.15.4 Start program control of the sample project

You start the axis control of the sample project on the SIMOTION device as follows

⚠ **WARNING**
Danger to life through unexpected machine movement
Make sure this presents no hazard to personnel or property.

1. Open the **Control Operating Mode** dialog. To do so, click **Control Operating Mode** in the toolbar.

2. Switch the SIMOTION CPU of the sample project to the RUN mode. To do so, click the assigned **RUN** switch in the **Control** column of the **Control Operating Mode** dialog.
   The axis starts to rotate immediately. The axis returns to a standstill after approximately 8 seconds.
   The SIMOTION device remains in RUN mode after execution of the **pos_axis** program.
3. Switch the SIMOTION device to STOP mode. To do so, click the assigned **STOP** switch in the **Control** column.

Note

If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

3.4.16 Monitor the application

3.4.16.1 Overview

Aim of Getting Started

In this part of Getting Started, you monitor the program-controlled axis motion.

- You monitor program execution.
- You monitor the values in the symbol browser.
- You compile certain values in a watch table.
- You record the course of the axis motion with the trace.

Requirements

- You have created the programs pos_axis, technology_fault, peripheral_fault and LAD_1 for the sample project, and you have assigned them to the tasks of the SIMOTION execution system, see the section Configure execution system (Page 552).
- The project has been compiled and downloaded to the target system, see the section Download the configured execution system to the target system (Page 556).

- SIMOTION SCOUT TIA is in online mode.
- Operating mode of the SIMOTION device:
  - The SIMOTION device of the sample project has been switched to the STOP mode on the software side. The SIMOTION SCOUT TIA mode selector switch, **Control Operating Mode** dialog, shows the STOP mode.
  - The SIMOTION device of the sample project has been switched to the RUN mode on the hardware side. The mode selector switch on the SIMOTION device is at position **0** (RUN).

3.4.16.2 Monitoring program execution

SIMOTION SCOUT TIA provides several functions for monitoring program execution.

In the sample project, you use the function **Monitor**. With this function, you can track execution of the command blocks in an MCC chart. The currently executed command block is shown against a yellow background in the chart. The marking runs at the same speed as the actual program execution. The marking is therefore only visible with the "slow" commands that wait for the machine to implement the command.

You monitor the execution of the commands in the MCC chart pos_axis as follows

1. Open the **Control Operating Mode** dialog.

   The SIMOTION device must be in **STOP** mode.
2. Open the MCC chart **pos_axis**.
3. Switch the monitoring function on:
   - To do so, select the command **MCC chart > Monitor** in the menu bar of the workbench. The function is switched on if a checkmark is visible next to the menu item.
   - Or click in the **MCC editor** toolbar on **Monitor**.

4. Start program control: Switch the SIMOTION device to the **RUN** mode in the **Control Operating Mode** dialog box.
   The **pos_axis** program is run through once. The currently executed command block is shown against a yellow background.

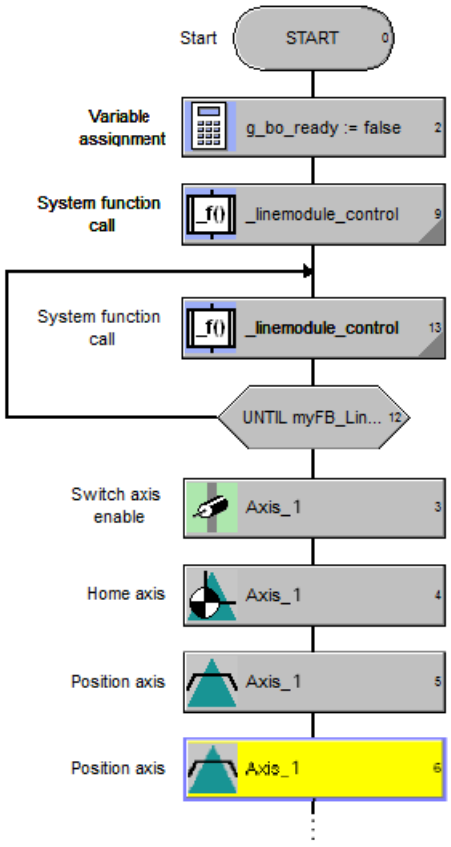


Figure 3-309 Monitor the program execution of the MCC chart pos_axis

5. Switch the SIMOTION device to **STOP** mode in the **Control Operating State** dialog box.

Further information

You can find further information on program monitoring in the online help under **Monitoring program execution**.

3.4.16.3 Monitoring variables

In the symbol browser, you can monitor variables (read out status value) or assign values to them (assign control values).

For the sample project, you monitor the actual position of the axis during the program run.

Monitor variables in the symbol browser

1. In the **AXES** folder of the project navigator, select the axis created in the sample project. The system variables and configuration data of the axis are displayed on the **Symbol browser** tab of the detail view.
2. Open the system variable **positioningstate.actualposition** (actual position of the axis) in the symbol browser.
   You find the variable as follows:
   - Filter the list: You can specify a filter criterion in the filter line of the symbol browser. The last 5 criteria are saved and can be selected for re-use.
     Enter a suitable filter term, for example, **positioningstate**, in the filter line. Press **RETURN** to confirm.
   - Search for variable: As an alternative to the filter function, you can search for the variable. Select the menu command **Edit > Find**. Enter a suitable search term in the **Find** dialog box, for example, **positioningstate**. Click **Find next**.

   The actual position of the axis is displayed in the **Status value** column of the symbol browser.
3. Start the program control via the dialog box **Control Operating Mode**. The program is run through once. During this time, the changing values of the actual axis position are displayed in the symbol browser.
   After the program has completed this run, the SIMOTION device is in the RUN state.
4. Switch the SIMOTION device to STOP mode in the **Control Operating State** dialog box.

Monitoring variables in watch table

Different variables, even of several devices, can be combined into a table in order to monitor them at the same time.

To add a variable to a watch table, you must first create a watch table.

Proceed as follows:

1. In the project navigator under **MONITOR**, double-click on the entry **Insert watch table** and confirm your configuration.
2. In the symbol browser, right-click on the variable you want to add to the watch table. Select the **Add to watch table** command in the shortcut menu.
   The selected variable is displayed in the table. In this way, you can add further variables to the watch table and monitor them.

You open a watch table as follows

You will find all created watch tables in the **MONITOR** folder of the project navigator. Double-click a watch table to open it.

You can find detailed information on this topic in the online help under **Working with the SCOUT Workbench > Working with lists > Watch table**.

3.4.16.4 Recording signals with the trace

Trace

You can use the trace to record and save signal characteristics and variable values. The recorded data can be used, for example, for diagnostics purposes in machine motion sequences, and for troubleshooting in user programs.

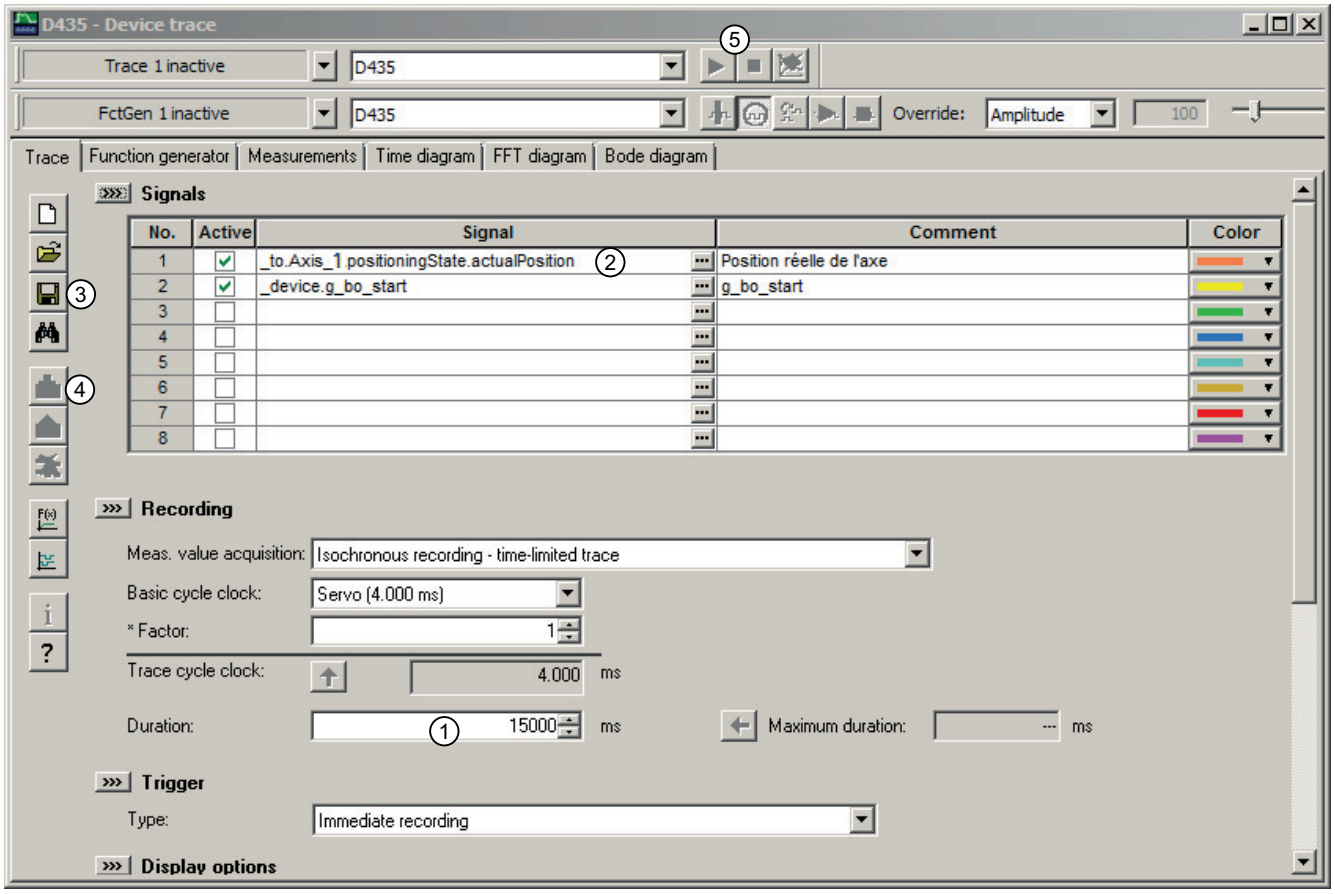
For the sample program, you record the actual position of the axis over time and represent it in the diagram.

Working with the trace

You call the trace as follows

1. Highlight the SIMOTION device in the project navigator.
2. Select the menu command **Target system > Device trace** or click the **Device trace function generator** button in the **Trace toolbar**.

The **Device trace** window appears in the working area.



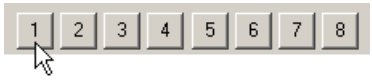
① ... ⑤ Reference is made to the circled digits in the text below.

Figure 3-310 Device trace window

You parameterize the trace for recording as follows

1. In the **Duration** field ① of the **Trace** tab, enter the recording duration 15000 ms.
2. Click the button ... ② in line **No 1** of the **Signals** table.
   The **Trace Signal Selection** window appears.

3. In the tree, select the branch **Sample_1 > (**SIMOTION device, e.g. **D435) > TO > Axis_1 > positioningstate**.
4. Select the system variable **actualPosition** in the variable table.
5. Click the button **1** to accept the system variable for channel 1.



The variable is displayed under **Signal name**.

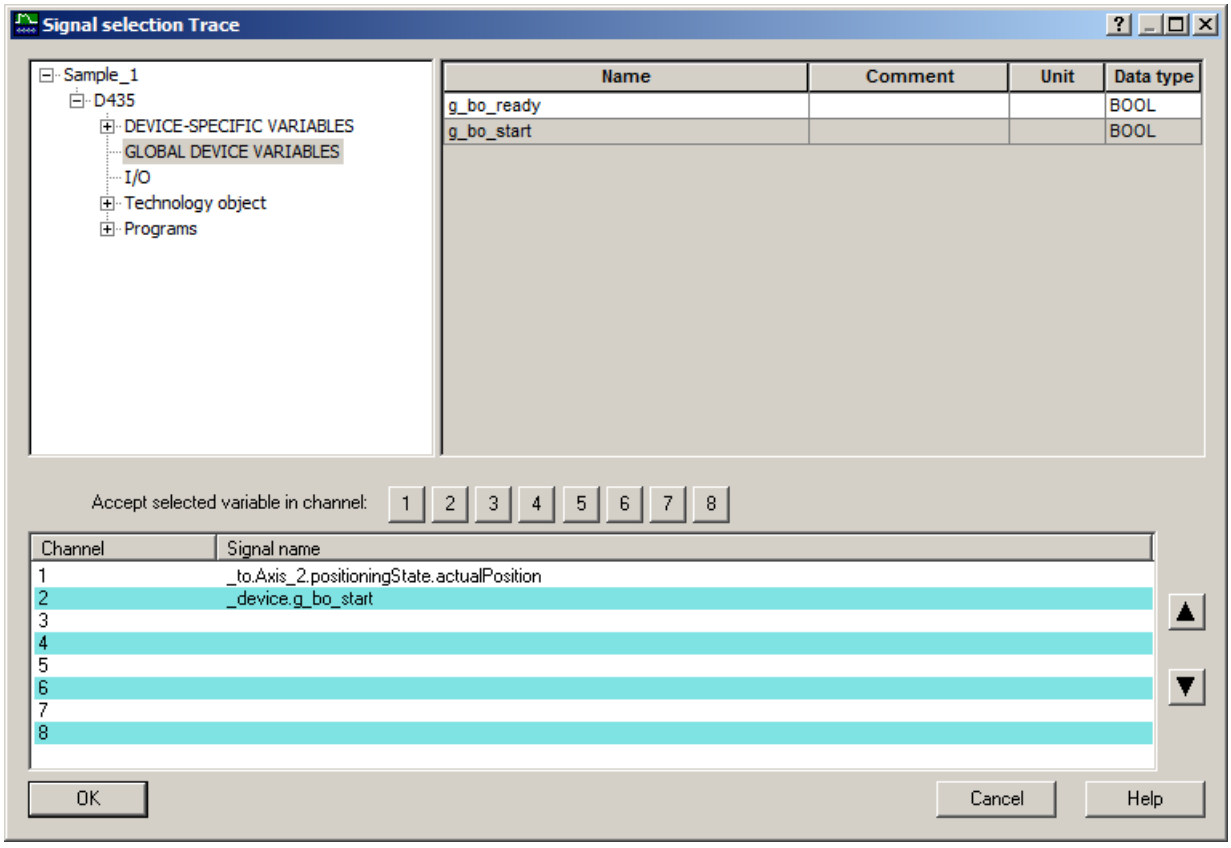


Figure 3-311 Trace signal selection

6. Repeat the channel assignment for the variable **g_bo_start**:
   Select the branch **(Project) > (SIMOTION device) >GLOBAL DEVICE VARIABLES**. Select the system variable **g_bo_start** in the variable table. Click the button **2**.

   Note

   You can also drag & drop the variables from the symbol browser or the watch table to the signal field of the Trace dialog. For how to drag & drop, see Section Variable assignment g_bo_ready:=false / g_bo_start:=true (Page 523).

7. Confirm with **OK**.
   The **Trace Signal Selection** window is closed.
   The trace is now parameterized for recording.

You save the parameterization of the trace as follows

The trace parameterization is not saved in the project data. When you close the project, the trace parameterization is deleted.

To save the parameterization, click the **Accept in catalog** button ③ on the **Trace** tab. In the **Catalog entry** field, enter the name under which the setting/parameterization is to be saved in the catalog of the trace.

You record with the trace as follows

1. Go online.
2. Download the parameterization of the trace to the target system:
   - Click the **Download parameterization** button ④ on the **Trace** tab.

     
   - Click **OK** to confirm the dialog that appears following successful downloading.
3. Open the **Control Operating Mode** dialog.

   The SIMOTION device must be in **STOP** mode.
4. Open the **Time diagram** tab in the **Device trace** window.

5. Start recording of the trace:
   - Click **Trace start**⑤ in the **Device trace** window.

   - Then immediately switch the SIMOTION device to the RUN mode in the **Control Operating Mode** dialog box.

   The program is started. The actual position of the axis is recorded and represented in the time diagram. After expiry of the recording duration, the signal profile of the actual position is displayed.

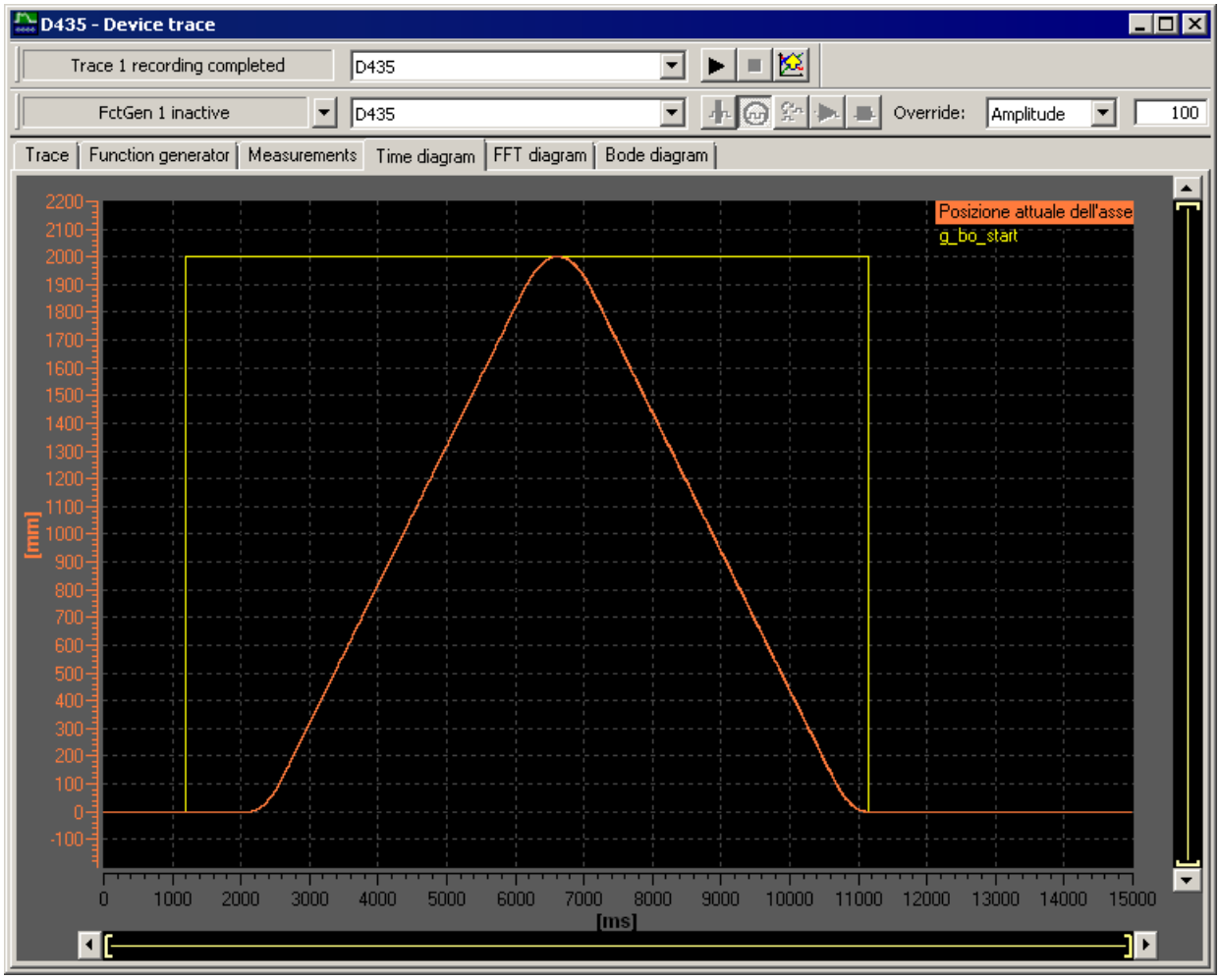


Figure 3-312 Time diagram of the axis

6. If the recorded curve is only partially displayed, select the menu item **Auto-scaling** in the context menu of the time diagram.
7. Switch the SIMOTION device to the STOP mode in the **Control Operating Mode** dialog.

Carrying out several measurements

You can carry out several measurements. They are shown on the **Measurements** tab and can be displayed in the diagram.

You can save and re-open recorded measurements for documentation purposes.

3.4.16.5 Result in the sample project

Creation of the trace time diagram concludes Getting Started.

3.4.17 ESD directives

3.4.17.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

3.4.17.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

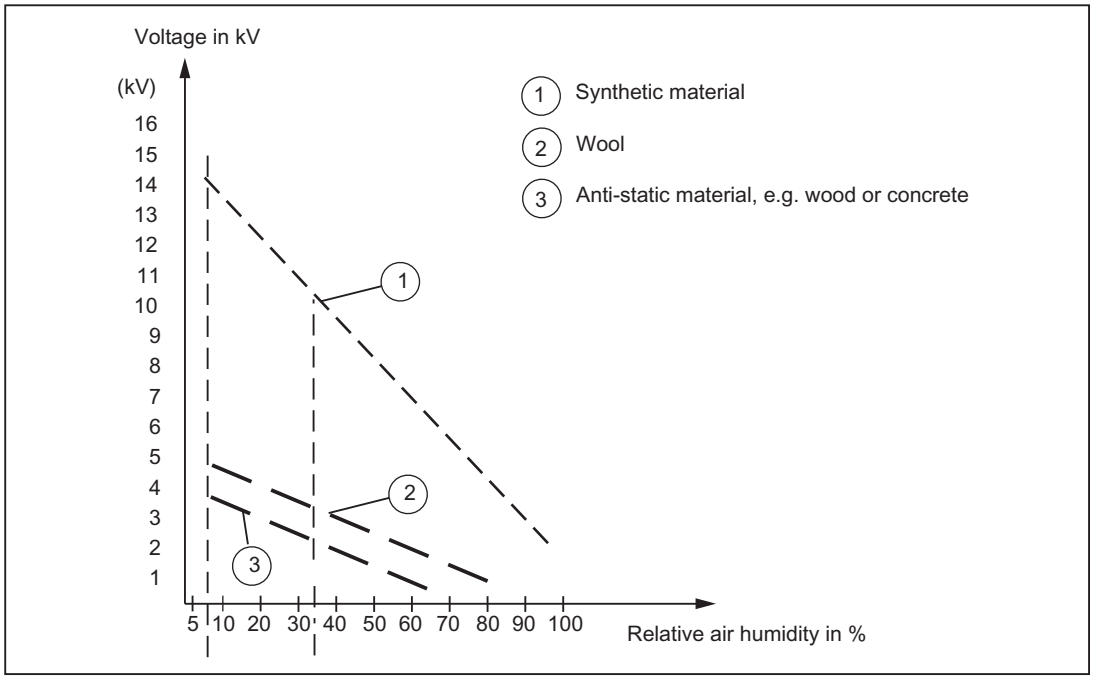


Figure 3-313 Electrostatic voltage that can accumulate on operating personnel

3.4.17.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

3.5 SIMOTION SCOUT TIA

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT TIA and it comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/de/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT TIA, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

3.5.1 Fundamental safety instructions

3.5.1.1 General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

3.5.1.2 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

3.5.1.3 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

3.5.1.4 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

3.5.1.5 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

3.5.1.6 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

3.5.1.7 Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

3. Hazardous shock voltages caused by, for example:
   – Component failure
   – Influence during electrostatic charging
   – Induction of voltages in moving motors
   – Operation and/or environmental conditions outside the specification
   – Condensation/conductive contamination
   – External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

3.5.2 Introduction

3.5.2.1 Target group and content of the configuration manual

The SIMOTION SCOUT TIA Configuration Manual is a general description of the software for new users and experienced users switching from SIMOTION SCOUT.

To differentiate the two versions from each other, the SIMOTION SCOUT designation refers to SCOUT in the SIMATIC Manager environment and the SIMOTION SCOUT TIA designation to SCOUT in the TIA Portal environment.

At the beginning of each section a note is included that indicates whether each configuration step is performed in the TIA Portal or in SIMOTION SCOUT TIA.

For an overview of the configuration step and its place of execution, refer to section Configuration - Overview.

Not all available software functions are described in this document. For detailed, topic-specific information, refer to the information system of the TIA Portal, the context-sensitive online help of SIMOTION SCOUT TIA, and the corresponding documentation.

Important notes and information on the SIMOTION Motion Control system are described in the following catalog:

- SIMOTION, Equipment for Production Machines, PM 21 Catalog

Note

Getting Started with SIMOTION SCOUT TIA

We recommend that you work through "Getting Started with SIMOTION SCOUT TIA" in the online help or the "SIMOTION SCOUT TIA Getting Started" tutorial. There you find guidelines how you work with SIMOTION SCOUT TIA, for example, how you create a project, compile and save it, insert a SIMOTION device, insert and parameterize a technology object, and how to create a program. When you have worked through all these steps, you will be able to create more complex projects.

Additional references

The "SIMOTION SCOUT TIA Getting Started" tutorial is available at (https://support.industry.siemens.com/cs/ww/en/view/109474299)

3.5.2.2 SIMOTION in the TIA Portal

By integrating the SIMOTION motion control system in the **T**otally **I**ntegrated **A**utomation Portal, Siemens has expanded the efficient engineering framework within the "Integrated Drive System" concept to include another powerful tool. The full range of SIMOTION motion control functions can now also be used in the TIA Portal. All the SIMOTION components can be configured and engineered simply and efficiently using the TIA Portal's intuitive user interface.

This enables the efficient and user-friendly mechanisms of this engineering framework to be used for commissioning, configuring and designing SIMOTION components. That includes, for example, merging of the hardware and network configuration into one consistent editor, thus enabling intuitive and fully graphical configuration and powerful diagnostics of the configured and networked components. In the device view, the SIMOTION CPU is displayed fully graphically with all its interfaces and properties. All the parameters can be selected and modified intuitively. In the network view, further automation components such as the HMI (Human Machine Interface) or drives can be connected to the CPU, and configuration of the PROFINET network topology is just as intuitive.

With SIMOTION SCOUT TIA, configuration is as easy as ever. Not only can you configure telegrams and integrate technology and drive objects, you can also configure automatic communication for distributed synchronous operation between two CPUs.

Figure 3-314 SIMOTION in the TIA Portal

SIMOTION devices in the TIA Portal

As of Version 13 of the TIA Portal, you can also configure SIMOTION devices.

A SIMOTION configuration comprises the following steps:

- Configuring the hardware
- Configuring the technology

The following overview shows which configuration steps you perform in the TIA Portal and which configuration steps you perform in SIMOTION SCOUT TIA:

Configuration steps in the TIA Portal

- Managing a project
- Configuring the hardware
- Saving and compiling the hardware configuration
- Configuring communication
- Creating F-programs

- Integrating HMI
- Set up and test online connection

Configuration steps in SIMOTION SCOUT TIA

- Configure the drive
- Configure the technology (e.g. axis, external encoder)
- Create user programs
- Saving and compiling
- Establishing online/offline connection
- Using diagnostic functions
- Using trace functions

Integration of SIMOTION SCOUT TIA as an option package of the TIA Portal has resulted in the following special features of use:

- You can only start SIMOTION SCOUT TIA from the TIA Portal.
- As soon as you close the TIA Portal, SIMOTION SCOUT TIA is also closed.
- When you switch to SIMOTION SCOUT TIA, you cannot undo the actions you last performed in the TIA Portal.

3.5.2.3 SIMOTION SCOUT TIA Engineering System

Introduction

The SIMOTION motion control system provides not only a range of ready-to-use functions, but can also be individually parameterized and programmed. High-performance tools, which provide optimum support and ease of use for the necessary engineering steps, are required for this.

The SIMOTION SCOUT TIA Engineering System is the environment for the uniform automation of production machines with SIMOTION and integrates into the SIMATIC environment in accordance with TIA (Totally Integrated Automation).

SIMOTION SCOUT TIA provides a uniform, function-oriented view for your automation task and it is also very user-friendly.

The possible SIMOTION applications range from simple, parameterizable, speed-controlled single axes through to complex, mechatronically-coupled and programmable multi-axis machines. Therefore, SIMOTION SCOUT TIA provides views that are adapted to the task, and it can be expanded by additional tools.

SIMOTION SCOUT TIA provides all the necessary tools for the following functions:

- Configuration
- Parameterization
- Programming

- Testing
- Diagnostics

The following tasks are graphically supported with operator guidance:

- Creation of the hardware and network configuration
- Creation, configuration and parameterization of technology objects such as axes, output cams and cams

3.5.2.4 SIMOTION hardware platforms

To meet the complex requirements of machine construction, SIMOTION offers three hardware variants with different performance, packaging formats and expandability options. The basic system characteristics, like the engineering, are identical.

SIMOTION D (Drive-based)

SIMOTION D is a compact, drive-based version of SIMOTION based on the SINAMICS S120 drives family. For SIMOTION D, the SIMOTION runtime environment and the SINAMICS drive software run concurrently on the controller hardware in the SINAMICS S120 packaging format.

SIMOTION P (PC-based)

SIMOTION P is a PC-based, open motion control system from SIMOTION. Control, motion control, and HMI functions are executed together with standard PC applications on the SIMOTION P hardware platform.

SIMOTION P combines the openness of the Windows operating system with the real-time capability of SIMOTION P Runtime.

SIMOTION C (Controller-based)

SIMOTION C is the modular controller variant in the tried and trusted packaging system of the SIMATIC S7-300 with its very varied expandability options on the I/O bus. The SIMOTION C240 high-performance motion controller is available for control and motion control tasks. The integrated interface for four analog coupled drives makes the SIMOTION C particularly suitable for compact applications with the control of analog electrical drives and the operation of hydraulic axes. SIMOTION C also supports operation of four stepper motors at these interfaces. SIMOTION C240 PN offers a PROFINET interface instead of the encoder and drive interfaces.

Note

In the current version, SIMOTION SCOUT TIA supports only specific SIMOTION devices.

To find out which SIMOTION devices are supported, see Supported devices (Page 584).

3.5.2.5 Supported devices

Not all devices that are available in SIMOTION SCOUT are currently supported by SIMOTION SCOUT TIA.

The following table lists the devices that are supported in the current version of SIMOTION SCOUT TIA:

SIMOTION devices	SINAMICS devices
As of SIMOTION firmware/kernel V4.3 and SINAMICS Integrated V4.5: • SIMOTION C240/C240 PN • SIMOTION D410-2 / D4x5-2 / CX32-2 As of SIMOTION firmware/kernel V4.5: • SIMOTION P320-4	SINAMICS V4.5 and as of 4.7: • SINAMICS S120 CU310-2 / CU320-2

Note

The SINAMICS devices listed in the table can be found in the TIA Portal in the "hardware catalog" task card at "Controller > SIMOTION > SIMOTION drives".

You can parametrize these devices in SIMOTION SCOUT TIA only if you have networked them with a SIMOTION device via PROFIBUS/PROFINET.

The following devices are not supported:

- Single drive devices

See also

Inserting a SIMOTION drive (Page 654)

Migrating a SIMOTION SCOUT project (Page 632)

SIMOTION hardware platforms (Page 583)

Overview (Page 824)

3.5.2.6 Supported functionalities

SIMOTION SCOUT TIA generally supports the same functionality as SIMOTION SCOUT and the TIA Portal.

The following functionality from SIMOTION SCOUT is not currently supported in SIMOTION SCOUT TIA:

- DCC SIMOTION / DCC SINAMICS
- Scripting for HWCN data
- XML import/export of hardware data
- SIMOTION Easy Project

See also

Migrating a SIMOTION SCOUT project (Page 632)

3.5.2.7 Programming languages

SIMOTION provides different programming languages for the solution of Motion Control tasks, control logic, arithmetic calculations, etc. During runtime, the selected programming language has no effect – except in the different displays when debugging. You can create user applications in different programming languages and use them jointly in a project.

The following programming languages are available in SIMOTION SCOUT TIA:

- **Motion Control Chart (MCC)**
  Graphical programming as a flow chart.
  In particular, for sequential tasks with a high level of motion control functionality.
- **Ladder Logic / Function Block Diagram (LAD/ FBD)**
  Graphical Programming as Ladder Logic / Function Block Diagram, supplemented by Motion Control Functions via PLCopen Function Blocks.
  In particular, for cyclic tasks with a high logic proportion.
- **Structured Text (ST)**
  Textual Programming in a High-level Language.
  As the base language of the SIMOTION system, ST supports all system features and functions of the technology packages and is thus suitable for all tasks.

Sources (units)

Programs are created in program containers, the so-called sources (units). They are compiled in the engineering system. Any errors or warnings that occur during compilation are output in the diagnostics window. Sources compiled without error can then be loaded into the associated controller.

A source contains any number of programs, functions, function blocks and classes. Each executable part of a source (program, function, function block, class) is called a POU (Program Organization Unit).

A source is divided into an interface and an implementation section.

Interface section

All parts exported by the source are defined in the interface section. Other sources and external components (e.g. HMI systems) can access these parts. These include user-defined data types, data (variables and constants) as well as names of programs, functions, and function blocks.

Implementation section

The data types and data defined in the implementation section are global throughout the source and can be used by all POEs. The implementation section also contains the program code of the POUs. Any POUs not specified in the interface section can only be used within the source.

Note

Source concept

The source concept with encapsulation of code and data allows you to structure applications. For example, the functionality of an entire machine module with a defined external interface can be implemented in a single source.

Tasks

Programs are processed in tasks. A task is a job which is executed in a certain chronological sequence. The advantage of the task system (execution system) is that processes appended to the appropriate task levels can run simultaneously.

The SIMOTION Motion Control system uses high-performance CPUs on which a real-time operating system - suitable for fast control processes - is implemented. Each task is allocated a slice of the computing time. The organization of the task executions is performed by the operating system. A differentiation is made between user and system tasks that are independent of one another.

Additional references

For detailed information on the programming languages and the execution system, refer to the online help of SIMOTION SCOUT TIA and the respective programming and operating manuals.

See also

Motion Control Chart (MCC) (Page 587)

Ladder Logic / Function Block Diagram (LAD/FBD) (Page 588)

Structured Text (ST) (Page 589)

Motion Control Chart (MCC)

MCC (Motion Control Chart) is a "flow diagram language" that graphically formulates the process sequences in production machines in a simple manner. The result is one or more flow diagrams, comprising MCC blocks that describe the time execution of the individual machine actions. Due to its special means of expression, an MCC is ideally suited to programming sequential processes.

Motion Control Chart supports the simple description of the motion sequences of machines using powerful motion control commands, such as reference axis, position axis, synchronize or desynchronize cam, and many more.

To control the machine sequence, commands are available for awaiting conditions and for formulating computations, as well as for programming various control structures, such as polling (IF), case determination (CASE) and loops (FOR, WHILE, UNTIL). Several MCC programs may be created to describe different process situations. For example, you can create one MCC program to bring the machine to a defined initial state when it is switched on, a second MCC program for the normal production sequence, and a third MCC program to specify what the machine has to do in the event of a fault.

All MCC blocks – a selection of the most important SIMOTION functions – are available in tool bars. They are grouped according to function and are automatically inserted in the flow diagram at the marked point by means of a click. A click on the individual elements opens specific dialogs for parameterization. Obviously, you can also add your own comments for the further documentation of the process sequence. Functions from the SIMOTION command library that are not individually offered as MCC blocks can be used in an MCC program by means of a special command.

Performance features:

- Easy-to-use due to graphical illustration in the form of flow diagrams.
- Hierarchical command library for motion control, PLC, and technology functions
- Control structures (IF, WHILE, CASE, etc.)
- Zooming for LAD, FBD and ST
- Subroutine calls (FB/FC, programs, etc.)
- Structuring based on command module generation, i.e. combination of command sequences to form a module command
- User-friendly debug functions for online test and diagnosis, for example, single-step, program status or breakpoints for easier troubleshooting (debugging)
- Monitor, trace

Note

Implicit conversion to ST

When being compiled, programs written in MCC are implicitly converted to ST programs and then compiled.

You can export the intermediate result as an ST and use it as a basis for your own ST programs.

Additional references

Detailed information can be found in the online help of SIMOTION SCOUT TIA and in the SIMOTION MCC Motion Control Chart Programming and Operating Manual.

Ladder Logic / Function Block Diagram (LAD/FBD)

LAD/FBD is a graphical programming language and is available for Ladder Logic / Function Block Diagram. The statement syntax corresponds to a circuit diagram (LAD) or a function block diagram (FBD). LAD/FBD enable simple tracking of the signal flow between power rails via inputs, outputs, and operations. LAD and FBD programs are usually suitable for application in cyclic tasks (in particular, BackgroundTask).

LAD/FBD programs consist of elements and boxes that are graphically connected to networks. Their operations work mostly according to the rules of Boolean logic or simple arithmetic expressions and equations. Therefore, they are suitable only for control-relevant programs or also for motion control tasks by using the PLCopen blocks.

Functions, function blocks and programs can be programmed in LAD/FBD. A source can contain several LAD and FBD blocks. Only one POU can be implemented in an LAD or FBD block.

LAD/FBD also include commands for SIMOTION system control using standard logic functions. These commands are added from the command library. Motion Control tasks are preferably programmed with PLCopen blocks. Blocks which have been programmed in other SIMOTION languages can be called. User-friendly functions such as "on the fly" variable declarations or automatic syntax checks are available when programming in LAD or FBD. It is possible to switch over between LAD and FBD in the editor at any time. A program can therefore be viewed and processed in either LAD or FBD.

The following user-friendly debug functions are available for online testing and diagnostics:

- Monitor variables in the symbol browser or a watch table
- Program status
- Trace tool with measurement functions for drives and function generators
- Breakpoints

Note

Direct editing of motion commands is not recommended. Instead, it is better to use the PLCopen blocks. These blocks are designed for integration in logic-oriented programs.

Additional references

Detailed information can be found in the online help of SIMOTION SCOUT TIA and in the SIMOTION LAD/FBD Programming and Operating Manual.

Structured Text (ST)

Structured Text

ST is a high-level, PASCAL-based programming language. ST is based on the IEC 61131-3 standard. This standard harmonizes programming languages for programmable logic controllers (PLC).

The basic command scope is sufficient for the implementation of everything related to data management, arithmetic functions, control structures and I/O access. The addition of technology packages for Motion Control expands the scope of commands by other comprehensive, extremely flexible Motion Control commands.

In addition, applications can be subdivided into any number of sections. Such a section might be a program allocated to a runtime level, an instantiatable function block with its own memory, or a function without its own memory. In this case, the function blocks and functions are not allocated to a runtime level, but are instead called in programs.

Performance features:

- Motion Control, PLC and technology functions in a single language
- Well-structured programs with comment capability
- Powerful editor functions, such as:
  - Syntax coloring
  - Automatic indenting
  - Automatic completion
  - Bookmarks
  - Fold (show and hide blocks)
  - Displaying sets of parentheses
  - Select text, e.g. by column
  - Using the command library
- Convenient debug functions for online testing and diagnostics, e.g. display of up-to-date variable content of the code sequence selected in the editor (program status) and breakpoints.

Note

Object-oriented programming (OOP)

As of version V4.5, the Structured Text programming language supports object-oriented programming.

Additional references

Detailed information can be found in the online help of SIMOTION SCOUT TIA and in the SIMOTION ST Structured Text Programming and Operating Manual.

3.5.2.8 CamEdit cam editor

CamEdit can be used to describe curves by means of either interpolation points or segments. A combination is not possible. If the curve is to be created from segments using polynomials, SIMOTION SCOUT TIA provides the VDI wizard as assistance. Cam geometries are created in offline mode.

Information on the graphical creation of cams can be found in the section CamTool options package (Page 590).

Additional references

Detailed information on the topic can be found in the SIMOTION SCOUT TIA online help and in the SIMOTION Motion Control, Synchronous Operation TO, TO Cam Function Manual.

3.5.2.9 CamTool options package

SIMOTION CamTool is a powerful, graphical editor for creating and optimizing cams.

SIMOTION CamTool can be used as an expansion package for SIMOTION SCOUT TIA and is completely integrated in the SIMOTION SCOUT TIA user interface.

Basic functions

SIMOTION CamTool provides the following basic functions:

- Inserting and editing cams.
  Cams can be inserted in a SIMOTION SCOUT TIA project using SIMOTION CamTool. You can also edit a cam created with CamEdit using CamTool: Cams can also be imported from a text file or read from a SIMOTION device.
- Customize display of the cam in CamTool.
  In SIMOTION CamTool, you can show and hide diagrams, change display parameters of the axes and diagrams and adjust the lines and fonts. You can also display auxiliary lines in the diagram.
- Convert cams from SIMOTION CamTool to SIMOTION CamEdit.
  To edit a cam that is edited in SIMOTION CamTool using SIMOTION CamEdit, the cam needs to be converted.
- Export cams to a text file.
- Load cams into a SIMOTION device.

Additional references

Detailed information can be found in the online help of SIMOTION SCOUT TIA and in the SIMOTION CamTool Configuration Manual.

See also

CamEdit cam editor (Page 590)

3.5.2.10 Technology packages and technology objects

Technology packages in SIMOTION SCOUT TIA

Technology packages combine software functions which are required for automation in mechanical engineering in various sectors. They are loaded into the controller during configuration and expand the basic functionality through additional system functions. The functions of the technology packages can be accessed easily in the SIMOTION SCOUT TIA command library during engineering. Access to the technology package functions is provided by additional language commands and system variables. Programming of motional sequences is therefore simple and integrated.

The following standard technology packages are available for SIMOTION SCOUT TIA:

CAM technology package

The SIMOTION CAM technology package contains the basic Motion Control technologies, such as drive axis, position axis, following axis, synchronous object, cam, output cam, cam track, and measuring input.

PATH technology package

The SIMOTION PATH technology package contains additionally the path interpolation technology.
Path interpolation generates the traversing profile for the path, calculates the path interpolation points in the interpolation cycle, and uses the kinematic transformation to derive the axis setpoints for the interpolation cycle points.

CAM_EXT technology package

The CAM_EXT technology package also contains objects for preparing technological data at the system level, e.g. addition object, formula object.

TControl technology package

The SIMOTION technology package for temperature control (TControl) provides temperature channels with extensive functions. These functions are also accessed via additional language commands and system variables.

More sector-specific technology packages are also available as separate products.

The loadable technology packages support the creation of technology objects (e.g. positioning and synchronous axis, cam tracks, external encoders) which can be accessed over system functions and system variables for use in every SIMOTION programming language.

Technology objects in SIMOTION SCOUT TIA

SIMOTION SCOUT TIA uses technology objects (TOs) to represent the functionality of axes, cams, output cams, etc. After creating a technology object (e.g. axis) and configuring it (e.g. as positioning axis), you can access it when programming with system functions.

Note

Missing licenses

All TOs outside the basic functionality (Motion Control Basic) must be licensed. Missing licenses are indicated by a flashing group error LED. The number and type of missing licenses is stated by the online diagnostics. They are also displayed during downloading.

Additional references

Detailed information is provided in the online help of SIMOTION SCOUT TIA and in the following documents:

- SIMOTION Runtime Basic Functions Function Manual
- Function Manual: SIMOTION Motion Control, TO Axis, Electric/Hydraulic, TO External Encoder
- Function Manual: SIMOTION Motion Control, Synchronous Operation TO, TO Cam
- Function Manual: SIMOTION Motion Control, Supplementary Technology Objects
- Function Manual: SIMOTION Motion Control Output Cams and Measuring Inputs
- Function Manual: SIMOTION Motion Control, TO Path Object
- Function Manual: SIMOTION Motion Control, Basic Functions for Modular Machines

See also

Licensing of the runtime components (Page 897)

3.5.2.11 CLib Studio option package

SIMOTION CLib Studio

If required, further functions and function blocks can be created in user libraries (SIMOTION CLib Studio) by means of C/C++ programming in the Windows environment.

They can be used in all SIMOTION languages (MCC, LAD, FBD, ST).

That allows the creation of application-specific and high-performance function extensions as well as adaptations including know-how protection.

3.5.3 Installation

3.5.3.1 SIMOTION SCOUT TIA and TIA Portal system preconditions

Note

Note, as of V14, TIA Portal supports only 64-bit operating systems.

Further details for the system requirements are contained in the *Readme* file in the start menu at "Start > All programs > Siemens Automation > Documentation > Readmes".

Compatibility details can be found in the Internet compatibility list at: Compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317)

3.5.3.2 Install SIMOTION SCOUT TIA

Requirements

The following requirements apply to the installation of SIMOTION SCOUT TIA:

- Hardware and software of the PG/PC meet the system preconditions.
- You require administrator rights on your PG/PC.
- All running programs are closed.
- The TIA Portal (framework) is installed, for example with the STEP 7 or WinCC software package.
- The SIMOTION SCOUT TIA version to be installed (not part of the TIA Portal) is compatible with the installed TIA Portal version.

Side-by-side installations

The following combinations are possible:

- Different versions of the TIA Portal
- Different SIMOTION SCOUT TIA versions (V4.4 and V4.5 or V4.4 and V5.4)
- SIMOTION SCOUT (as of V4.4) and different SIMOTION SCOUT TIA versions
- STARTER (as of V4.4) and different SIMOTION SCOUT TIA versions
- S7T Config and different SIMOTION SCOUT TIA versions

Note

You cannot install STARTER, S7T Config and SIMOTION SCOUT at the same time.

Procedure

To install SIMOTION SCOUT TIA, proceed as follows:

1. Insert the installation data medium into the appropriate computer drive.
   The Setup program will start automatically provided Autostart is not disabled on the PG/PC.
   If the Setup program does not start automatically, start it manually by double-clicking the "Start.exe" file.
2. If necessary, click the "Yes" button for the Windows confirmation prompt.
3. Select the installation language of the Setup program.
4. Read the product information and installation instructions. To do this, click the "Read installation notes" or "Read product information" button.
5. Close the online help and click the "Next >" button.
6. The "Product Languages" dialog box shows those languages that were installed with the TIA Portal Framework. All languages supported by the product are installed automatically for SIMOTION SCOUT TIA from the activated languages. Click the "Next >" button.
7. In the following dialog box, select the product configuration.
   The required memory and the target directory are also listed. Click the "Next >" button.
8. Read and accept all the listed licensing agreements. Click the "Next >" button.
9. Read and accept all the listed safety instructions for safe operation. Click the "Next >" button.
10. All installation settings are listed in the overview. If you want to change the settings, navigate with the "< Back" button to the appropriate installation dialog box, and perform the change there.
    Otherwise, click the "Install" button to start the installation.

    Note

    License transfer

    If a license key is found during installation, you can transfer it to your PC. If you skip the license transfer, you can transfer it later using the Automation License Manager.

    You will find further information on licensing in section Licensing for SIMOTION SCOUT TIA (Page 596).

11. The computer may need to be restarted. Activate the "Yes, restart computer now" checkbox. Then click the "Restart" button.
12. If the computer does not need to be restarted, click the "Exit" button.

Result

SIMOTION SCOUT TIA has been installed in the specified directory on the PG/PC.

3.5.3.3 Uninstall SIMOTION SCOUT TIA

Requirements

The following requirements are valid for uninstalling SIMOTION SCOUT TIA:

- You require administrator rights on your PG/PC.
- All running programs are closed.

Note

Uninstalling TIA Portal

When an upgrade is made to a different TIA Portal version, the TIA Portal and SIMOTION SCOUT TIA must be uninstalled beforehand.

Procedure

To uninstall SIMOTION SCOUT TIA, proceed as follows:

1. Open the Control Panel via "Start > Control Panel".
2. Double-click the "Programs and functions" entry.
   The "Uninstall or Change a Program" dialog opens.
3. Double-click the "Siemens Totally Integrated Automation portal Vxx" entry.
4. If necessary, click the "Yes" button for the Windows confirmation prompt.
5. Select the installation language and click the "Next >" button.
   The dialog for selecting the products to be uninstalled opens.
6. Select SIMOTION SCOUT TIA and click the "Next >" button.
7. Check in the "Overview" dialog the list of products to be uninstalled. To make changes, click the "Back" button.
8. Click the "Uninstall" button.
   The uninstall is started.
   The expected time until completion of uninstalling is displayed in the dialog.
9. The computer may need to be restarted. Activate the "Yes, restart computer now" checkbox. Then click the "Restart" button.
10. If the computer does not need to be restarted, click the "Exit" button.

Result

The selected SIMOTION SCOUT TIA version is uninstalled.

3.5.3.4 Licenses

Licensing for SIMOTION SCOUT TIA

To use SIMOTION SCOUT TIA, in addition to the product DVDs, you will also receive a data medium containing the authorization data medium on which you will find the license key for SIMOTION SCOUT TIA.

Because the license key is identical for SIMOTION SCOUT TIA and SIMOTION SCOUT, it is valid for both.

The *Automation License Manager* program manages the license.

Installing the license for SIMOTION SCOUT TIA

1. Insert the authorization data medium with the license key.
2. Start the *Automation License Manager* program as follows:
   – In the start menu, select "Start > All programs > Siemens Automation > Automation License Manager".
   Or double-click the "Automation License Manager" symbol.
3. In the navigation area, select the drive with the authorization data medium.
   The license key is displayed in the right-hand window.
4. Select the license key and drag-and-drop it to the target drive.
5. Exit the Automation License Manager.
6. Remove the authorization data medium.

Result

SIMOTION SCOUT TIA has been licensed.

Installing the license upgrade for SIMOTION SCOUT TIA

1. Insert the authorization data medium with the upgrade license key.
2. Start the *Automation License Manager* program as follows:
   – In the start menu, select "Start > All programs > Siemens Automation > Automation License Manager".
   Or double-click the "Automation License Manager" symbol.

   In the navigation area, select the drive where the authorization for the old version of SIMOTION SCOUT TIA is located. Generally, the authorization will be installed on a hard disk drive on the PG/PC.
3. Transfer the authorization to the data medium containing the upgrade license key.
   To do this, select the license in the right-hand window and select "License key > Transfer..." in the menu.
   The "Transfer license key" dialog opens.

4. Select the connected data medium and click "OK" to start the transfer.
   When the transfer is complete, the authorization and the upgrade license key will be stored on the data medium.
5. Select the "Upgrade..." command in the "License Key" menu.
   The older authorization will be deleted. After the upgrade, you will have a new floating license.

   Note

   Do not interrupt the upgrade while it is in progress. Interrupting this process can result in the license key being lost.

6. Now transfer the new floating license onto the hard disk drive.
   Proceed as described under item 3.
7. Exit the Automation License Manager.
8. Remove the automation data medium.

Result

SIMOTION SCOUT TIA has been upgraded.

Additional references

Detailed information about license keys is provided in the online help of the *Automation License Manager*.

3.5.4 User interfaces

3.5.4.1 Introduction

The TIA Portal serves as framework for the hardware and network configuration (HWCN).

Call SIMOTION SCOUT TIA from this framework with the "SIMOTION configuration" entry. SIMOTION SCOUT TIA is the workbench for configuring and programming the motion control.

As you are working both in the TIA Portal and in SIMOTION SCOUT TIA, both user interfaces are briefly described in the following sections.

Additional references

Detailed information about the user interfaces is provided in the information system of the TIA Portal and in the help system of SIMOTION SCOUT TIA.

See also

TIA Portal (Page 598)

SIMOTION SCOUT TIA (Page 603)

3.5.4.2 TIA Portal

Views in the TIA Portal

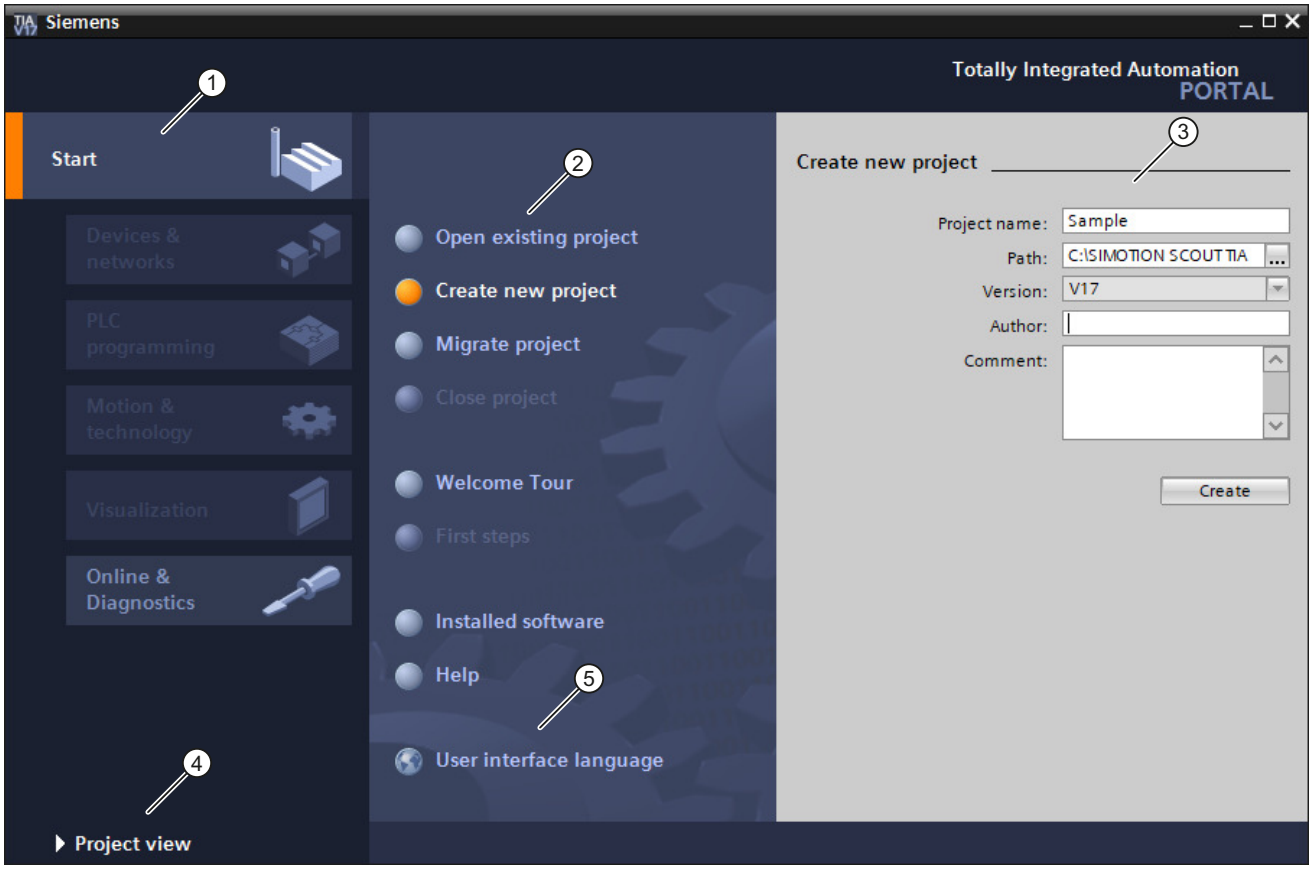
The following views are available in the TIA Portal:

- Portal view
- Project view

Portal view

The portal view provides an overview of all configuration steps and enables a task-based entry to your automation solution.

The individual portals ("Start", "Devices & Networks", "PLC Programming", "Visualization", "Online & Diagnostics", etc.) provide the basic functions for the individual tasks.



① **Portals for the various tasks**

The portals provide the basic functions for the individual task areas. The portals provided in the portal view depend on the products that have been installed.

Start SIMOTION SCOUT TIA from the "Motion & Technology" portal.

② **Actions for the selected portal**

Depending on the selected portal, the actions are offered that can be executed in this portal. You can open context-sensitive help in every portal.

③ **Selection window for the selected action**

The selection window is available in all portals. The content of the window adapts to your current selection.

④ **Change to the project view**

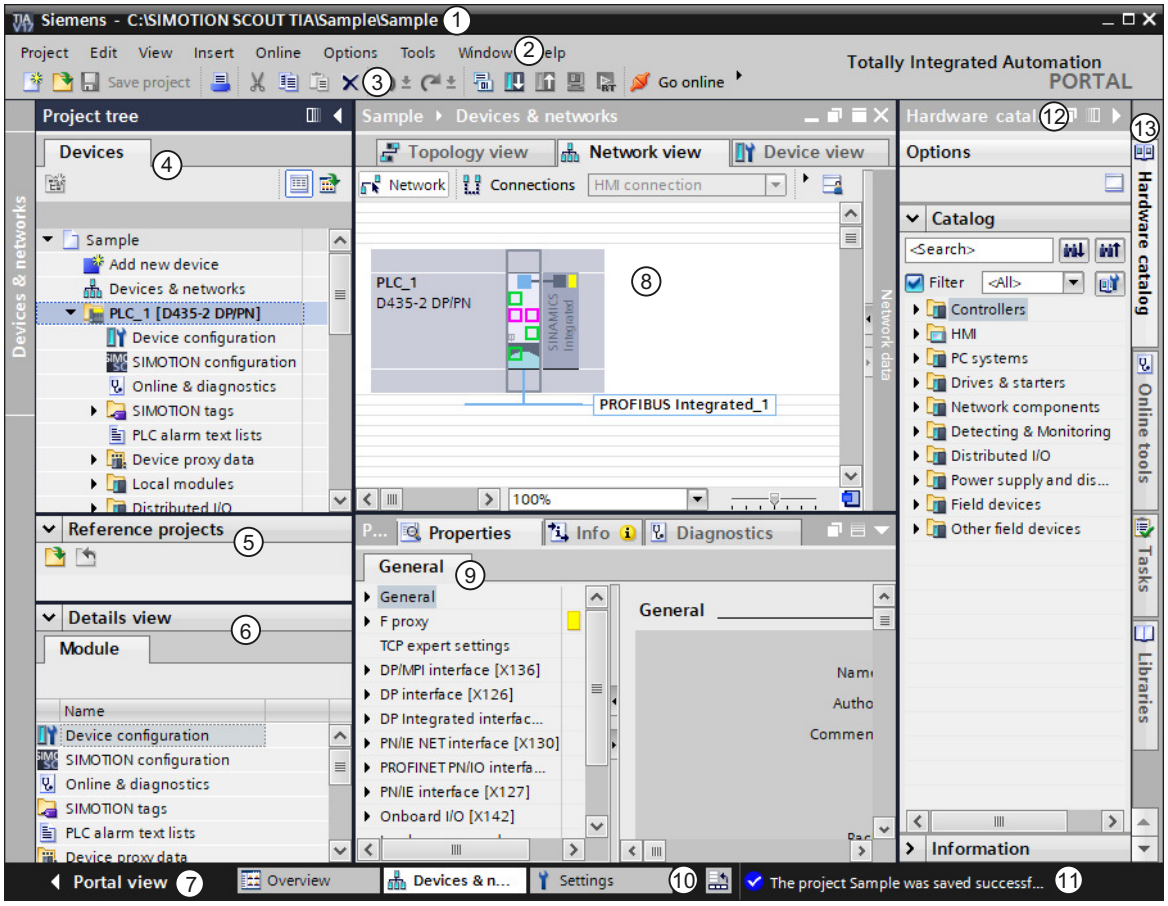
Use the "Project view" link to change to the project view.

⑤ **Select user interface language**

Use the "User interface language" link to set the user interface language.

Project view

The project view is a hierarchically structured view of all components in a project. The project view enables quick access to all objects in the project, the relevant working areas and editors. The various work windows show all the associated data for the selected objects.



① **Title bar**

The name of the project is displayed in the title bar.

② **Menu bar**

The menu bar contains all the commands that you require for your work.

③ **Toolbar**

The tool bar provides you with buttons for commands you will use frequently. This allows you to access these commands faster.

④ **Project tree**

The project tree provides access to all components and project data. For example, you can perform the following actions in the project navigation:

- Add new components
- Edit existing components
- Query and change the properties of existing components

⑤ **Reference projects**

In the "Reference projects" view, you can open further projects in addition to the current project. Although these reference projects are write-protected, you can copy individual devices from a reference project and add them to your current project where they can be further processed.

⑥ **Details**

The details view shows certain contents of a selected object. Possible contents include text lists and variables.

⑦ **Changing to the portal view**

Use the "Portal view" link to change to the portal view.

⑧ **Working area**

The objects that you can open for editing purposes are displayed in the working area.

⑨ **Inspector window**

The Inspector window displays additional information on a selected object or on executed actions.

⑩ **Editor bar**

The opened editors are displayed in the editor bar. If you have many open editors, they are shown grouped. The editor bar can be used to change quickly between the opened elements.

⑪ **Status bar with progress display**

The status bar contains the progress display for those background processes that are running currently.

⑫/⑬ **Task cards**

Depending on the edited or selected objects, task cards are provided to perform further actions, such as

- Select objects from a library or from the hardware catalog
- Search for objects in the project and replace
- Drag objects to the working area

The available task cards are found in a bar at the right-hand edge of the screen. You can expand and collapse the task cards at any time.

Additional references

For detailed information about the user interface and operation of the TIA Portal, refer to the "Introduction to the TIA Portal" section in the information system of the TIA Portal.

TIA Portal - language settings

User interface language

The TIA Portal allows you to switch the user interface language during operation. SIMOTION SCOUT TIA takes over the language setting of the TIA Portal. However, change in the language setting takes effect only after a restart of the SIMOTION SCOUT TIA.

The TIA Portal supports the following languages:

- German
- English
- French
- Italian

- Spanish
- Chinese (simplified)

Note

Not all languages of the TIA Portal are supported by SIMOTION SCOUT TIA.

If you have selected a user interface language not supported by SIMOTION SCOUT TIA, the SIMOTION SCOUT TIA user interface is displayed in English.

Changing the user interface language

Proceed as follows to change the user interface language:

1. Switch to the TIA Portal.
2. Switch to the "General" group in the "Tools > Settings" menu.
3. Select the preferred language from the "User interface language" drop-down menu of the "General Settings".

Note

Multiple languages in SIMOTION SCOUT TIA

After switching the language, the SIMOTION SCOUT TIA texts and messages may be displayed in both the previous and the new selected user interface language.
In this case, exit SIMOTION SCOUT TIA and restart the TIA Portal.

TIA Portal - Using Help

TIA Portal information system

The TIA Portal has an extensive information system. Start the information system in the Portal view via the "Help" button or in the Project view via the "Help -> View Help" menu and read how to work with the help in Section "Introduction to the TIA Portal".

Additional references

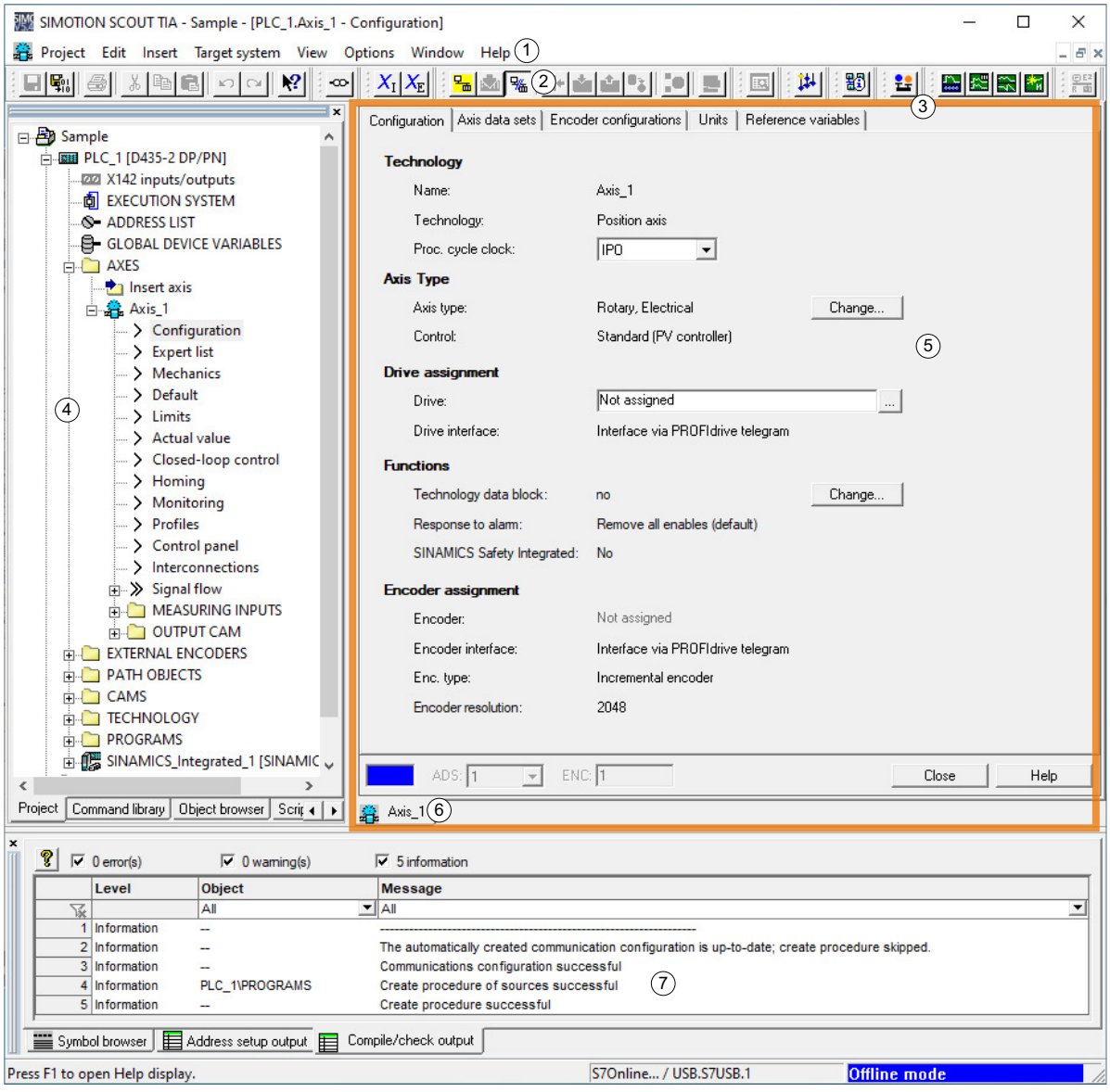
You will find detailed information on all steps that you perform for SIMOTION configuration in the TIA Portal (e.g. configuring hardware, setting up communications, etc.) in the information system of the TIA Portal in the "Configuring SIMOTION devices" section.

3.5.4.3 SIMOTION SCOUT TIA

SIMOTION SCOUT TIA - using workbench

The SIMOTION SCOUT TIA workbench is the common framework for all other tools of the engineering system. The workbench is the navigation center of the individual engineering steps and offers a uniform and integrated view of all data and programs.

The following figures shows an example of the Workbench components:



① Menu bar
② Tool bars
③ Working area
④ Project navigator
⑤ Snap-in
⑥ Tab
⑦ Detail view

Workbench components

The workbench comprises the following components:

- **Menu bar**:
  You call the functions of SIMOTION SCOUT TIA via the menus in the menu bar.
- **Tool bars:**
  Frequently used menu commands are also available in tool bars, which can be activated or deactivated as required. These provide quick access to the functions. The tool bars can be undocked from the header and relocated to a different position (e.g. at right, left, lower border) or as a window.
- **Working area**:
  The task-specific windows are displayed in the working area. In these windows, you can perform the configuration with wizards for the axis configuration and drive configuration. You also create programs in the working area. Further information about the active window in the working area is provided in the detail view.
- **Project navigator**:
  The project navigator provides an overview of the entire project. All defined elements, such as devices, drives, axes, etc., are displayed in a tree structure.
  The following tabs are also available in the project navigator:
  - Command library - depending on the configuration, contains all the commands and functions required for the programming
  - Object browser - clearly displays the object structure of the configuration in a hierarchical structured tree
  - Scripting library - contains all scripting objects and methods in alphabetical order
- **Snap-in / tab:**
  A Snap-in is a program integrated automatically in the working area of the SIMOTION SCOUT TIA Workbench. Snap-ins provide functions for processing SIMOTION SCOUT TIA projects and are displayed as a work window in the working area of the workbench. You can open several snap-ins. Open Snap-ins are displayed in the working area as tabs. The active Snap-in is visible in the foreground.
  The following Snap-Ins are available:
  - Program editors
  - Wizards for the configuration of technology objects
  - Device diagnostics
  - Drive navigator
- **Detail display:**
  In the detail display, you can show more detailed information about the element selected in the project navigator and the active window in the working area, for example, variables of a program, system variables of a technology object, protocols of compiled program sources.

Additional references

Detailed information can be found in the SIMOTION SCOUT TIA online help.

SIMOTION SCOUT TIA - using the working area

The workbench displays all of the Snap-in work windows in the working area. Each Snap-in provides its own work window. You can open multiple instances of these windows. You can, for example, open several programs at the same time for editing.

The following table provides an overview of the active functions in the working area of the workbench depending on the project mode (offline/online):

Function in project navigator	**Function in working area offline mode (Example)**	**Function in working area online mode (Example)**
Programs (MCC, ST, LAD/FBD) For further information, please go to: • SIMOTION MCC Motion Control Chart • SIMOTION ST • SIMOTION LAD/FBD	Create program	Monitor program
Execution system	Assign program	-
Technology objects (TO configuration)	Create and configure TO element	Change configuration data during RUN
Cam editor CamEdit/CamTool	Create cam	Change cam
Trace	Load and evaluate old recordings	Create current recordings

Window in the working area

You can change the size of the windows in the working area:
Click the edge of the window, hold down the left mouse button, and drag the window to the required size.

To maximize or restore a window, press <Ctrl+F11>.

Each window opened in the working area can be accessed via a tab at the bottom edge of the working area.

The following options are available to bring a window into the foreground:

- Click the relevant tab.
- Select the appropriate item in the "Window" menu.

To close a window, proceed as follows:

- Configuration dialogs:
  Click the "Close" button.
- Editors for MCC, ST and LAD/FBD:
  Click the "X" button in the top right-hand corner.

Configuration dialogs and editors can also be closed with <Ctrl+F4>.

SIMOTION SCOUT TIA - using the project navigator

By default, the project navigator comprises the "Project" tab and the "Command library", "Object browser" and "Scripting library" tabs.

Project tab

The "Project" tab displays the entire project structure and is used for managing elements within the projects.

In addition to the elements created automatically via the hardware configuration, create the following elements manually in the project navigator:

- Technology objects
  e. g. axes, external encoders, measuring inputs, cams
- Programs
  e. g. ST source files, MCC chart
- Watch tables
- Libraries

You can change the properties, e.g. the name of an element.

To improve clarity in the project navigator, you can sort and group lower-level objects into topics.

Command library tab

Depending on the configuration, the commands and functions required for the programming are displayed in a tree on the "Command library" tab. You can use these commands to program in the ST editor, LAD/FBD editor and MCC editor, e.g. to create conditions. You can search in the command library or set filters.

Object browser tab

The object structure of the configuration is clearly displayed in a tree in the "Object browser" tab. The tree is structured hierarchically and clearly shows the individual objects, classes, interfaces and references.

Scripting library tab

The scripting structure is clearly displayed in a tree in the "Scripting library" tab. The tree contains the individual objects and methods of scripting in alphabetical order.

SIMOTION SCOUT TIA - menus

SIMOTION SCOUT TIA menus are subdivided into static and dynamic menus.

Static menu

Static menus are primarily used to control the workbench or a project. Static menus are permanently displayed.

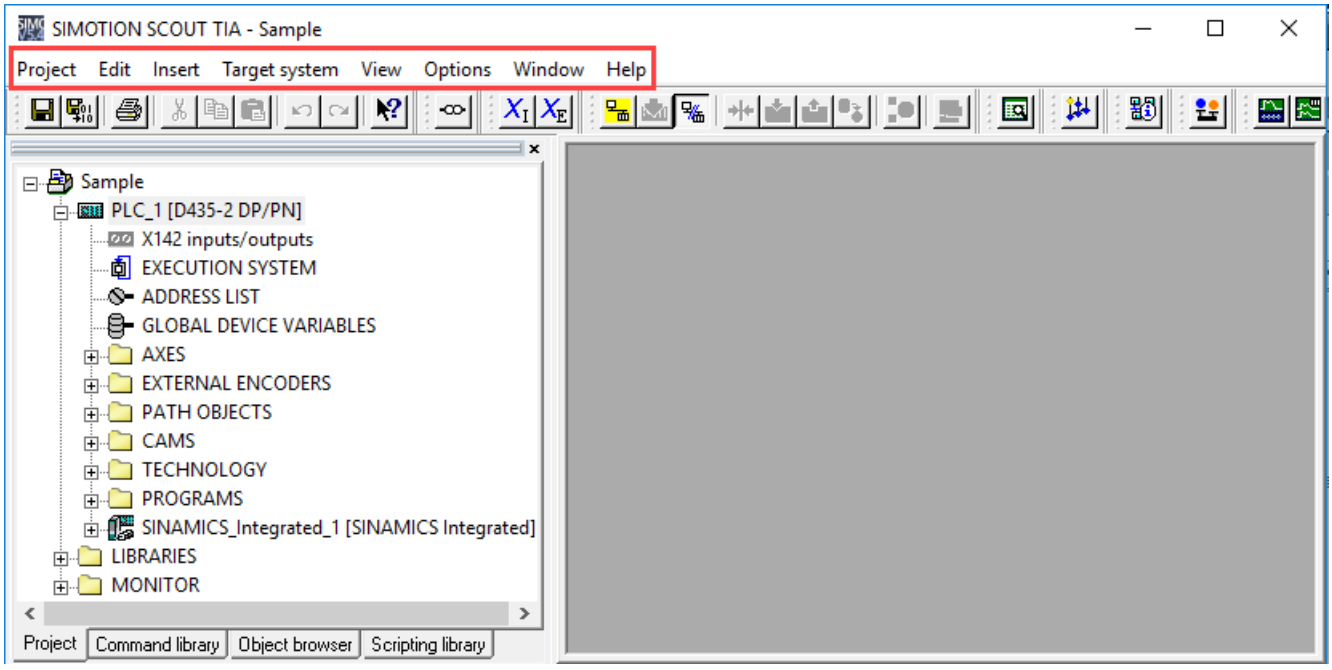


Figure 3-315 Static menu

Dynamic menu

Dynamic menus are provided by Snap-ins and are added to the static menu. The menu with active Snap-in in the working area is always displayed.

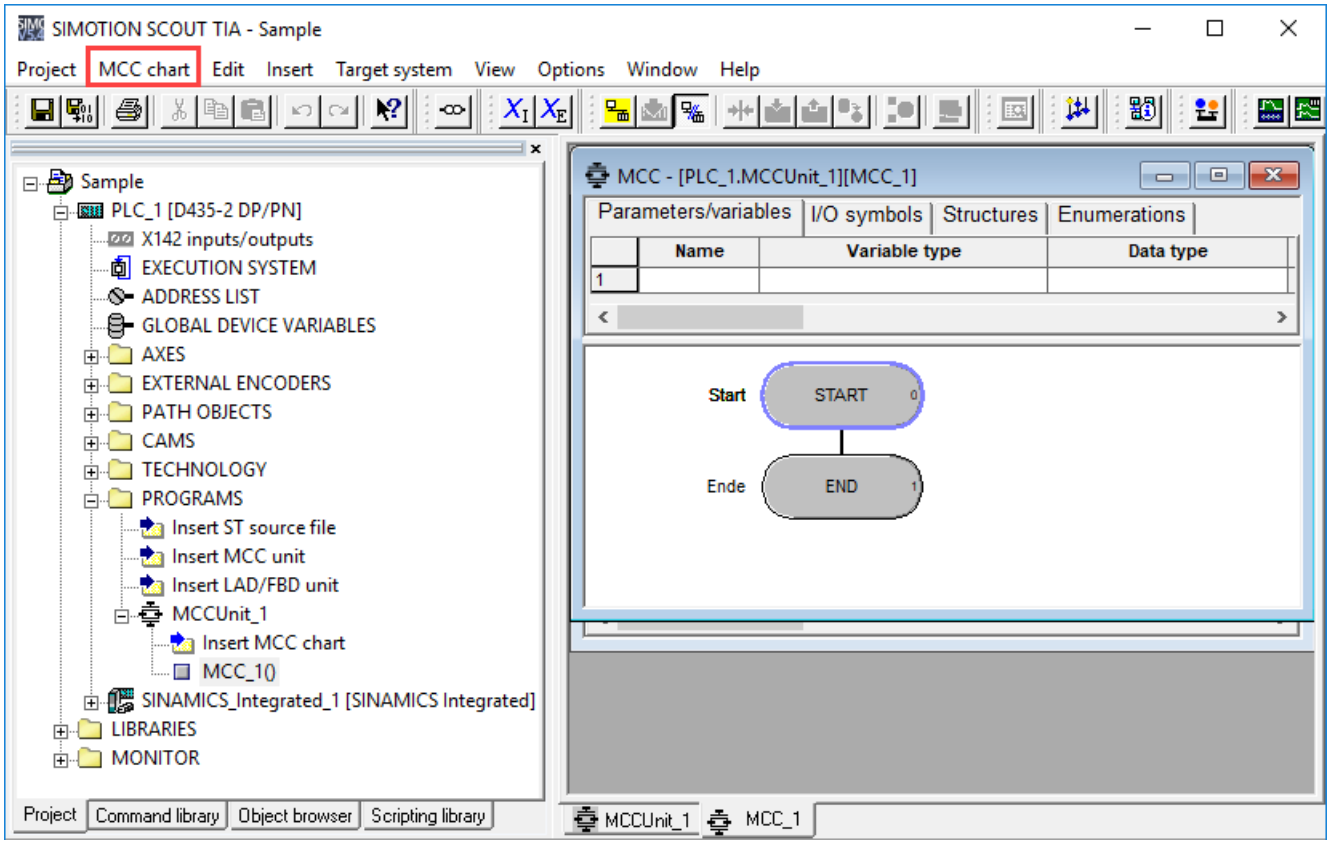


Figure 3-316 Example of a dynamic menu

Table 3-38 Structure of the menu bar

Menu	Comment
Project	Static menu, always visible
(Dynamic menu)	See "Dynamic menus" table
Edit	Static menu, always visible
Paste	Static menu, always visible
Target system	Static menu, always visible
View	Static menu, always visible
Options	Static menu, always visible
Window	Static menu, always visible
Help	Static menu, always visible

Table 3-39 Dynamic menus

Dynamic menu	Comment
Axis	Visible only when the project is open and the associated Snap-in is active in the working area. The dynamic menus appear in the menu bar between the "Project" and "Edit" static menus.
Output cam	
Measuring input	
Synchronous operation	
Cam	
External encoder	
Sensor	
Controller object	
Drive	
Fixed gear	
Formula object	
Addition object	
Drive control panel	
Trace	
ST source	
MCC source	Visible only when a project has been loaded and the associated Snap-in is active in the working area. The dynamic menus appear in the menu bar between the "Project" and "Edit" static menus.
MCC chart	
LAD/FBD source	
LAD/FBD program	

SIMOTION SCOUT TIA - menu items

You can use the shortcuts listed in the table to call the menu items available in SIMOTION SCOUT TIA.

Table 3-40 Menu items in SIMOTION SCOUT TIA

Shortcuts	Menu item	Response
Project...		
Ctrl+Alt+K	Check consistency	Checks project consistency
Ctrl+Alt+B	Save and compile changes	Saves the project and compiles the changes made in the project since the last compilation
Ctrl+P	Print	Prints the selected window
Alt+F4	Exit	Exits the SIMOTION SCOUT TIA program

Shortcuts	Menu item	Response
Edit...		
Ctrl+Z	Undo	Undoes the last action
Ctrl+Y	Redo	Redoes the last action
Ctrl+X	Cut	Cuts the selection
Ctrl+C	Copy	Copies the selection
Ctrl+V	Paste	Inserts the clipboard contents

Shortcuts	Menu item	Response
Edit...		
Del	Delete contents	Deletes the selection
F2	Rename	Renames the selected tree object
Alt+ENTER	Object properties	Displays the properties of an object in the project tree
Ctrl+Alt+O	Open object	Opens a new object of the selected tree object
Ctrl+A	Select all	Selects the entire contents in the ST and MCC Snap-ins

Ctrl+F	Search	Opens the Search window
F3	Find next	Continues the search from the current position
Ctrl+Shift+F	Search in the project	Opens the Search window
Ctrl+Shift+G	Replace in the project	Opens the "Search and Replace" window

CTRL+H	Replace	Opens the "Replace" window
CTRL+J	Display next position	-
Ctrl+G	Go to in the project navigator...	Opens the "Go To in the Project Navigator" dialog box

Shortcuts	Menu item	Response
Target system...		
Ctrl+L Is only possible online	Download / target device	Loads individual target device
Ctrl+D Is only possible online	Device diagnostics	Opens device diagnostics
Ctrl+I Is only possible online	Control operating state	Opens the dialog box for controlling the operating state

Shortcuts	Menu item	Response
View...		
Ctrl+F11	Maximize working area	Maximizes and restores the view of the working area
Ctrl+F12	Maximize detail view	Maximizes and restores the view of the detail view
Ctrl+Num+ (Plus key on the numeric keypad)	Zoom in	Enlarges the graphic in the program editors
Ctrl+Num+ (Minus key on the numeric keypad)	Zoom out	Scales down the graphic in the program editors
F5	Refresh	Refreshes the view

Shortcuts	Menu item	Response
Options...		
Ctrl+Alt+E	Settings	Opens the "Options > Settings" dialog box

Shortcuts	Menu item	Response
Window...		
Ctrl+Shift+F5	Arrange cascading	Arranges the opened windows in the working area
Ctrl+Shift+F3	Arrange vertically	

Shortcuts	Menu item	Response
Snap-in menu...		
Ctrl+F4	Close	Closes the selected window
Ctrl+B	Accept and compile	Compiles the active object
Ctrl+E	Expert list	Opens the expert list for the current technology object

Shortcuts	Menu item	Response
Help...		
F1	Help topics	Opens the entire help available for SIMOTION SCOUT TIA
SHIFT+F1	Context-sensitive help	Opens the context-sensitive help function for the selected object, parameter, etc.

Note

List of all shortcuts

The online help of SIMOTION SCOUT TIA includes a complete list of all the shortcuts available in SIMOTION SCOUT TIA and the programming and operating instruction manuals.

SIMOTION SCOUT TIA - keyboard operation and shortcuts

There are various keyboard assignments and shortcuts for the menu items to facilitate your work in SIMOTION SCOUT TIA.

The following table provides an overview of the keyboard assignments or shortcuts that you can use for SIMOTION SCOUT TIA.

Table 3-41 Keyboard action

Keyboard action / shortcuts	Meaning
Workbench: Change window...	
Alt+0	Project navigator
Alt+1	Working area

Keyboard action / shortcuts	Meaning
Workbench: Change window...	
Alt+2	Detail view
Ctrl+F6	Next window in the working area
Ctrl+F11	Minimize/maximize working area in relation to the whole desktop
Ctrl+F12	Minimize/maximize detail view in relation to the whole desktop

Project navigator	
Left mouse button	Selects the tree object at the cursor position; detail view displays the associated details
Double-click with left mouse button	Selects the tree object at the cursor position; detail view displays the associated details; corresponding Snap-in opens
Right mouse button	Selects the tree object at the cursor position; detail view displays the associated details; corresponding context menu opens
UP/DOWN arrow keys	Selects the tree object at the cursor position
"ENTER"	Snap-in for the selected tree object opens
"Context menu key"	Context menu for the selected tree object opens

SIMOTION SCOUT TIA - using the context menus

The tree elements of the project navigator have context menus. Context menus give you fast access to all important functions permitted for this tree element.

To call the desired function of the tree element via context menus, proceed as follows:

1. Select the appropriate element in the project navigator.
2. Open the context menu with a right-click.
3. Left-click the appropriate command to select it.

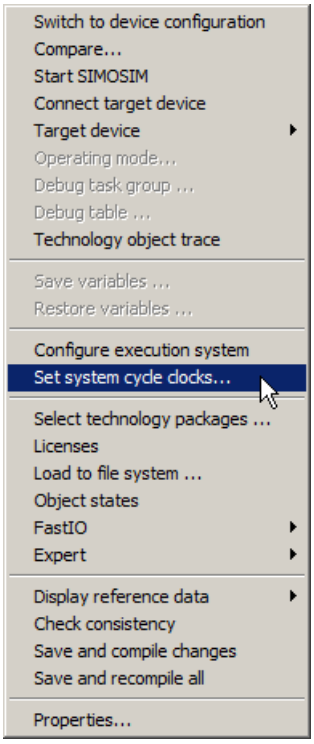


Figure 3-317 Context menu for SIMOTION device

SIMOTION SCOUT TIA - detail view

Using the detail view

When you select an element in the project navigator, the associated detail view will appear in the lower area of the workbench.

To display the detailed view for the address list and the watch table, double-click this element in the project navigator.

Depending on the selected element, different tabs are offered and information about the element is displayed therein.

The tabs available are determined by the project mode (offline/online) and the active snap-ins.

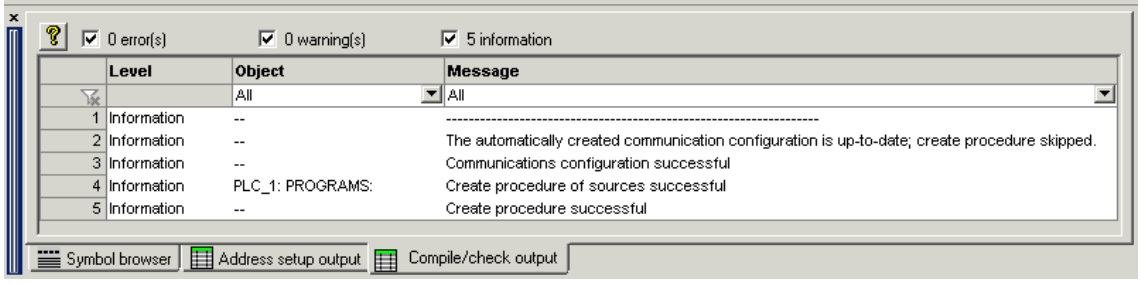


Figure 3-318 Example of detail view

Each tab is only opened once. This means:

- The active tab shows the details of the selected element/action.

You can maximize or minimize each tab with <Ctrl+F12>.

Using the symbol browser

The symbol browser is a tab in the detail view and is always displayed. The symbol browser displays status values of the variables for the element selected in the project navigator.

Permanently retain the display in the symbol browser

You can permanently retain the display in the symbol browser. The display remains even when, for example, other device or technology objects are selected in the project navigator.

To permanently retain the display in the symbol browser, proceed as follows:

1. Click the pin at the top right of the symbol browser view.

The active element remains displayed until this function is deactivated. To deactivate this function, click the pin again.

Using the address list

The address list is where you create I/O variables, which you then assign to a particular item of hardware. You can monitor I/O variables in the address list, and, when necessary, modify them. The I/O variables are assigned either manually or via an assignment wizard.

Opening the address list

To open the address list, proceed as follows:

1. Browse to the folder for the device in the project navigator.
2. Double-click the "Address list" element below the device.
   The address list opens in the details area.

Additional references

Further information about the the address list is available in the SIMOTION SCOUT TIA online help.

Using the watch table

With the symbol browser you can view the variables belonging to one object in your project; with the program status you can view the variables belonging to a selected monitoring area in the program. With watch tables, in contrast, you can monitor selected variables from different sources as a group (e.g. program sources, technology objects, SINAMICS drives - even on different devices). You can sort the variables in the watch table in any way, add comments and combine them into groups. You can hide individual variables to make the watch table more manageable and also control variables via the watch table directly.

Creating a watch table

To create a watch table and assign your variables, proceed as follows:

1. Open the "Monitor" folder in the project navigator.
2. Double-click the "Insert watch table" entry to create a watch table.
3. Enter the name of the watch table.
   A watch table is created with this name in the "Monitor" folder.
4. In the project navigator, click the object from which you want to move variables to the watch table.
5. In the symbol browser, in the address list, or in the expert list, select the corresponding variable line by clicking its number in the left column.
6. From the context menu, select the "Add to watch table" command and the appropriate watch table, e.g. "Watch_table_1".
7. If you click the watch table, you will see in the detail view of the"Watch table" tab that the selected variable is now in the watch table.
8. Alternately, you can copy-and-paste variables to the watch table.
9. To monitor the variables of various objects, repeat steps 3 to 6.

You can also create a watch table direct by selecting a variable followed by "Add to watch table > New watch table" in the context menu.

The new watch table is created automatically and filled with the selected variables.

If you are connected to the target system, you can monitor the variable contents.

Using multiple watch tables (>Watchtable_temp<)

To monitor variables simultaneously in different watch tables, create temporary watch tables.

To create a temporary watch table ">Watchtable_Temp<", proceed as follows:

1. Open the "Monitor" folder in the project navigator.
2. Double-click ">Watchtable_Temp<".
   The temporary watch table is displayed in the detail view.
3. Drag-and-drop the desired watch tables from the project navigator to the temporary watch table.
   The watch tables are arranged below each other.
   The contents are lost when the temporary watch table is closed.

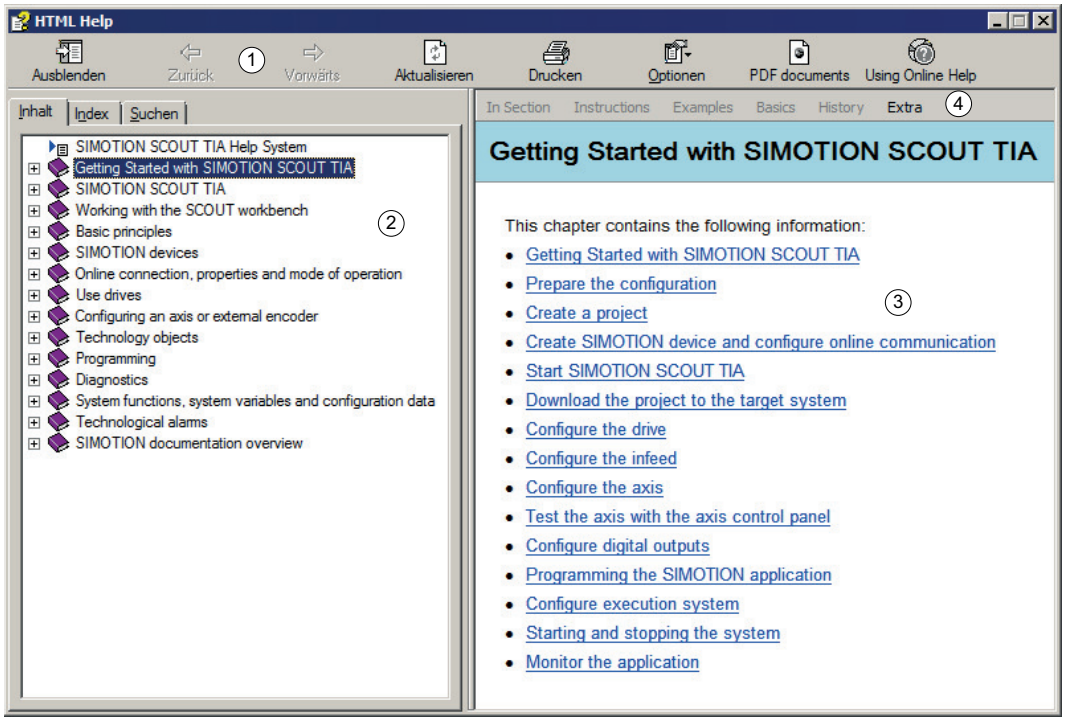
Additional references

Detailed information about the use of watch tables is contained in the "Watch table" section in the SIMOTION SCOUT TIA online help.

SIMOTION SCOUT TIA - using the help

SIMOTION SCOUT TIA has a comprehensive context-sensitive help. The following is a description of how to work with the help system using different examples.

Structure of the Online help window



1 Menu bar

You can use buttons in the menu bar to configure the help or display general information.

- Click "PDF documents" to display the available function diagrams.
- Click "Use online help" to display the help page for the full text search on the "Search" tab.

2 Navigation area

In the navigation area you can navigate through the help, open the index or search through the entire help.

- Click the "Contents", "Index", or "Search" tab to navigate through the help.
- Click the "+" in front of a book to open the table of contents.
- Click a book or help page to display its contents in the contents area.

3 Contents area

The contents for the help pages selected in the navigation area are displayed in the contents area.

Within the contents area, you can use symbols to display hidden information:

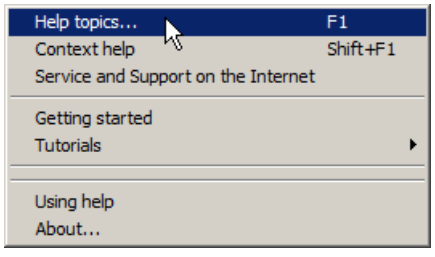
- Click the arrow to display or hide additional information.
- Click the minimized view to display a hidden figure.
- Click the copy symbol. The following program code will be saved automatically in the clipboard. You can then insert this in a program editor.

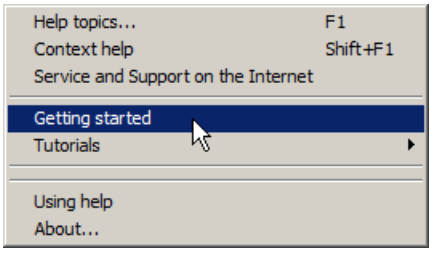
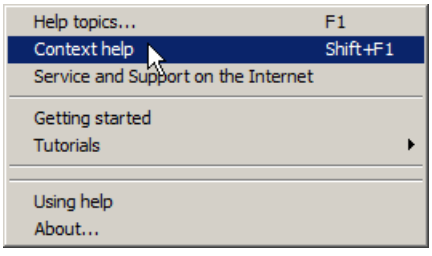
4 Link bar

The header area of many help pages has a link bar. This link bar displays further information for the selected help page. The link bar contains the following entries:

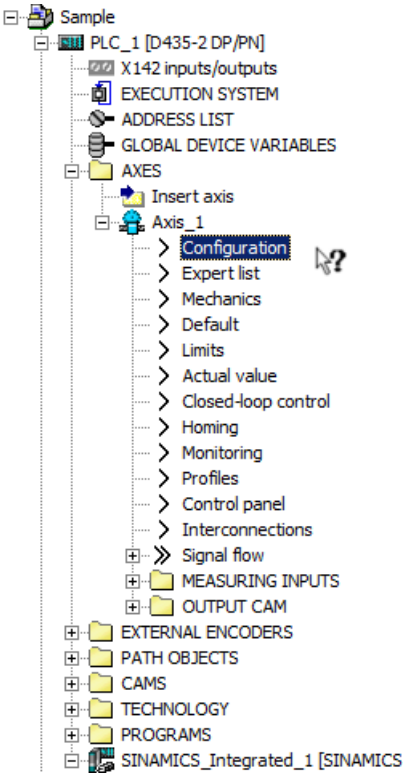
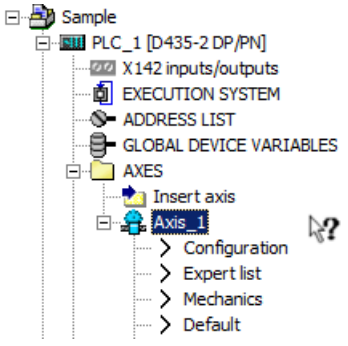
- In Section
  Links to headers of the displayed topic.
- Programming Manuals
  Links to handling instructions.
- Examples
  Links to examples.
- Fundamentals
  Links to background information, such as definitions or details.
- History
  List of the most recently opened help pages.
- Options
  Link to the start page.
  Forwards and backwards in the previously opened help pages.

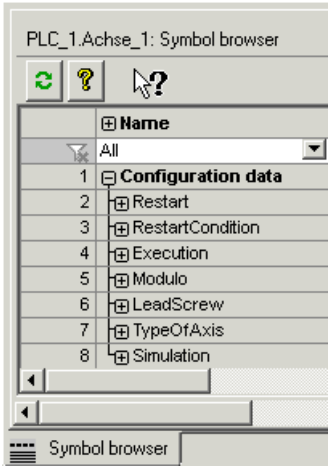
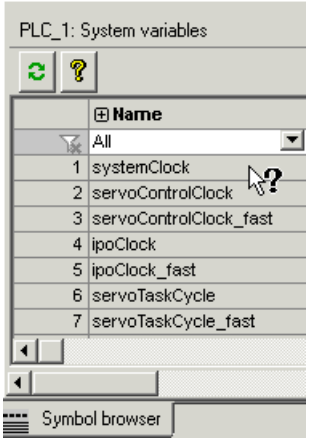
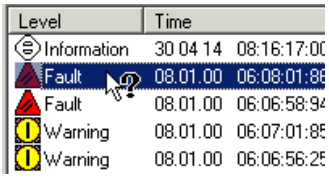
Features of online help

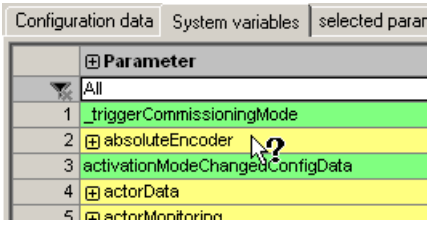
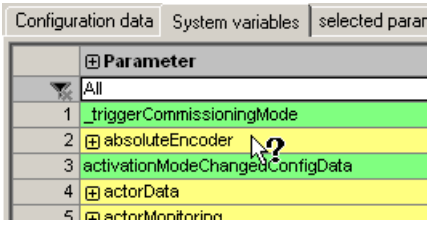
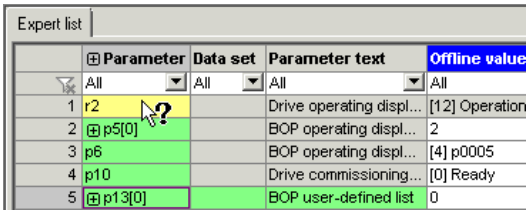
Features of online help	To open the help:	
Open entire help	Key <F1>	Press <F1>.
	or	
		Select "Help > Help topics" from the menu.
	The entire help opens.	

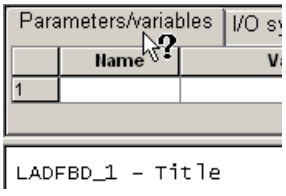
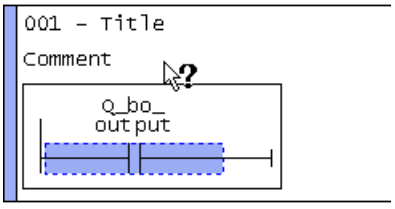
Features of online help	To open the help:	
Open Getting Started		Select "Help > Getting Started" from the menu.
	Getting Started opens.	
Help button	 or 	Click the Help button of the respective dialog box or window.
	The context-sensitive help for the dialog box opens.	
Context-sensitive help	<Shift+F1>	• Press <Shift+F1>.
	• or	
		• Click the button in the menu bar.
	• or	
		• In the menu, select "Help > Context help".
	The mouse pointer changes to a question mark.	
		• Then click with the changed mouse cursor on the dialog box, the parameter, the input field or the menu item.
	The context-sensitive help for the selected entry opens.	

Examples of the context-sensitive help

Help in the project navigator		
General help		Press <Shift+F1>. Then click with the question mark on the project navigator. The general help for the project navigator opens.
Help for the technology object (only SIMOTION)		Press <Shift+F1>. Then click with the question mark on a technology object in the project navigator. The help for this technology object opens.

Help for the detail view		
General help for the symbol browser		Press <Shift+F1>. Then click with the question mark in the margin of the symbol browser. Alternatively, you can also open the help for the symbol browser by clicking . The general help for the symbol browser opens.
System variables in the symbol browser		In the project navigator, select the SIMOTION device or the technology object. The system variables of the element are displayed in the symbol browser. Press <Shift+F1>. Then click with the question mark on the system variable in the symbol browser. The associated help opens.
Help for alarms		Press <Shift+F1>. Then click with the question mark on the alarm shown on the "Alarms" tab. The help for the drive alarm or technological alarm opens.

Help for the expert list		
Configuration data		Press <Shift+F1>. Then click with the question mark on the configuration data item in the expert list. The help opens.
System variables		Press <Shift+F1>. Then click with the question mark on the system variable in the expert list. The help opens.
Parameters of the drives		Press <Shift+F1>. Then click with the question mark on the parameter in the expert list. The help opens.

Help for the LAD/FBD editor (SIMOTION only)		
Declaration area		Press <Shift+F1>. Then click with the question mark on a tab in the declaration area of the LAD/FBD editor. The help opens.
Editor area		Press <Shift+F1>. Then click with the question mark on an element in the editor area of the LAD/FBD editor. The help opens.

3.5.5 Basics of SIMOTION configuration in the TIA Portal

3.5.5.1 Configuration overview

Configuring requirements

The following requirements apply to configuring:

- The TIA Portal (framework) is installed, for example with the STEP 7 or WinCC software package.
- SIMOTION SCOUT TIA has been installed and licensed for the PG/PC.
- The hardware has been installed and wired.
- The PG/PC has been connected to the SIMOTION device via the PROFIBUS, Ethernet or PROFINET IO interface.
- For configuring the drives with the drive wizard, you need the article number of the drive unit and the components (power supply, power unit, motors and encoders).

The following table provides an overview of the individual configuring steps and where to implement them.

Configuring step	Where
① Create a new project	TIA Portal
② Insert SIMOTION devices	TIA Portal
③ Insert a SIMOTION drive	TIA Portal
④ Compile hardware configuration	TIA Portal
⑤ Configure online access	TIA Portal
⑥ Configure communication	TIA Portal
⑦ Configure HMI connection	TIA Portal
⑧ Go online	SIMOTION SCOUT TIA
⑨ Configure the drives	SIMOTION SCOUT TIA
⑩ Configure the technology objects	SIMOTION SCOUT TIA
⑪ Create user programs	SIMOTION SCOUT TIA
⑫ Test and diagnostics	SIMOTION SCOUT TIA

See also

Target group and content of the configuration manual (Page 579)

3.5.5.2 Managing projects

Basic principles

Projects are used for the orderly storage of data and programs that result in the creation of an automation solution. You create projects in the TIA Portal and manage them there. The data summarized in a project shall include, in particular:

- Configuration data on the hardware structure and parameter data for modules
- Configuration data for communication via networks
- Configuration data for the devices
- Reports
- Configuration data
- User programs
- Motion profiles
- Drive data

The following table provides an overview of the individual configuration steps and where to implement them.

Project management	Where
Creating a new project	TIA Portal
Open project	TIA Portal
Delete a project	TIA Portal
Close a project	TIA Portal
Save project as	TIA Portal
Migrating a project	TIA Portal
Save project	SIMOTION SCOUT TIA
Save project and compile changes	SIMOTION SCOUT TIA
Save project and recompile everything	SIMOTION SCOUT TIA
Load project to the target system	SIMOTION SCOUT TIA
Saving and exporting a project	SIMOTION SCOUT TIA

See also

Creating a new project (Page 625)

Transferring SIMOTION SCOUT projects (Page 631)

Creating a new project

Note

You can only create a project in the TIA Portal.

Procedure

To create a project, proceed as follows:

1. Start the TIA Portal.
2. Select the "Create new project" entry in the secondary navigation in the Portal view.
3. Assign a name for the new project below "Project name".
4. Click the "..." button to set the storage location.
5. Enter any required comments in the comment field.
6. Create the project using the "Create" button.

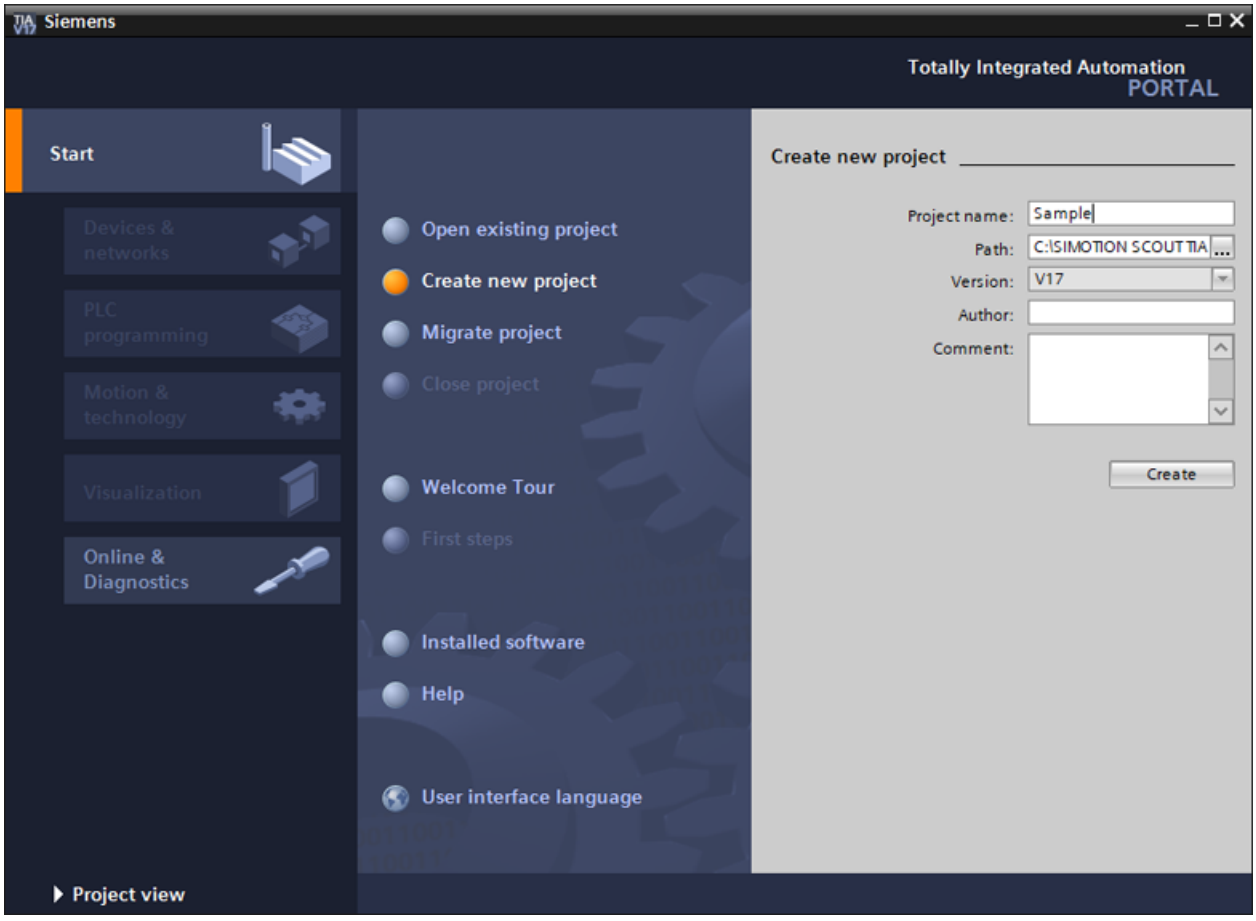


Figure 3-319 Creating a project

Result

You have created a project that was saved in the specified directory.

Note

Maximum number of characters

If necessary, change the storage location or project name because a maximum of 94 characters are available for the project name and 143 characters for the path.

Open project

Note

You can only open a project in the TIA Portal.

Procedure

To open a project, proceed as follows:

1. Start the TIA Portal.
2. Select the "Open existing project" entry in the secondary navigation in the Portal view.
3. Select the project you want to open in the right-hand window at "Recently used".
4. Click the "Open" button to open the project.

If the project to be opened is not listed, proceed as follows:

1. Click the "Browse" button and navigate to the project you want to open in the subsequent "Open existing project" dialog.
2. Double-click the project file.

Result

You have opened the project and are in the Portal view.
Switch to the Project view to continue with the configuration.

Close a project

Note

You can only close a project in the TIA Portal.

Procedure

To close a project, proceed as follows:

1. Switch to the TIA Portal.
2. Open the "Close" entry in the "Project" menu in the Portal view.
   Or select the "Close project" command in the secondary navigation in the Portal view.
3. If you have made changes in the project, you will be prompted in "The project has been changed" dialog whether or not you want to save the changes.

Result

You have closed the selected project. SIMOTION SCOUT TIA has been closed.

Save project as

Note

You can only perform "Save project as" in the TIA Portal.

Procedure

To save a project at the required target location, proceed as follows:

1. Switch to the TIA Portal.
2. Select the "Save project as ..." command in the "Project" menu in the Portal view.
3. In the "Save current project as ..." dialog, navigate to the storage location where you want to save the project.
4. Assign a file name to the project.
5. Save the project at the set storage location by clicking the "Save" button.

Result

You have saved the project in the specified location.

Note

Maximum number of characters

If necessary, change the storage location or project name because a maximum of 94 characters are available for the project name and 143 characters for the path.

Delete a project

Note

You can only delete a project in the TIA Portal.

Requirement

- The project is not opened.

Procedure

To delete a project, proceed as follows:

1. In the "Project" menu, select the "Delete project..." command.
   The "Delete project" dialog opens and a list of recently used projects is displayed. The project currently open is not displayed in the list.
2. Select the project you want to delete.
   If the required project is not in the list, click the "Browse" button. Navigate to the project folder and open the project file.
3. Click the "Delete" button.
4. Confirm the prompt with "Yes" to delete the project permanently.

Result

The complete project folder has been deleted from the file system.

Search and replace

Browsing a project

You can use the following search options within the TIA Portal:

- Browse the complete project
- Search and replace within an editor
- Browse the hardware catalog

Note

Search in SIMOTION SCOUT TIA

To search for any text or variables within the Motion Control configuring, switch to SIMOTION SCOUT TIA in the "Edit > Search in the project..." menu.

Procedure

To start the search editor in the TIA Portal, proceed as follows:

1. In the "Edit" menu, select the "Browse project" command.
   The search editor opens.
2. Enter the search text as "Search for:".
3. If the search results need to be restricted, activate the appropriate options for "Restrict search to:".
4. Start the search with the "Start search" command.

Result

All found objects are listed. Double-click a hit to open the source.

Search and replace in the project

The following methods are available within an editor to search for and replace texts.

Note

Searching and replacing in SIMOTION SCOUT TIA

To search for or replace any text or variables within the Motion Control configuring, switch to SIMOTION SCOUT TIA in the "Edit > Replace in the project..." menu.

Procedure

To start the editor for the "Search and replace " function in the TIA Portal, proceed as follows:

1. In the "Edit" menu, select the "Search and replace" command.
   The "Search and replace" editor opens.
2. Enter the search text as "Search for:".
3. Activate the desired additional options for the search.
4. Click "Search".
5. The first search hit is selected in the editor.
6. Enter the text with which you want to replace the search hit as "Replace".
7. To replace the selected hit, click "Replace".

Result

The found term is replaced with the desired text.

Print project

Print project content

The "Print" function allows you to print the complete project, individual objects, visible information from an editor or a compact project overview.

Note

Printing in SIMOTION SCOUT TIA

To print a complete project overview or individual parts from the configuring within the Motion Control configuring, e.g. programs or technology objects, switch to SIMOTION SCOUT TIA, select the element to be printed, and select the "Project > Print..." menu.

Procedure

To print content from the hardware configuration in the TIA Portal, proceed as follows:

1. Select in the project navigation the element to be printed.
2. Select the "Print project..." menu.
   The "Print" dialog opens.
3. Select the desired print options.
4. To start the print operation, click "Print".

Result

The selected project content is printed.

Archiving a project

The archiving of a project has the advantage of reducing the project size for large projects, for example, to send a project as e-mail.

Procedure

To archive a project, proceed as follows:

1. Save the currently open project.
2. Select the "Archive..." command in the "Project" menu in the project view.
3. Select the required archiving settings.
   To save a compressed project archive, activate the "Archive as compressed file" option.
   If you do not want to back up restorable data when the project is archived, select the option "Discard restorable data".
   To automatically add the date and time of day to the file name, select the option "Add date and time to target name".
4. Under "Target path:", navigate to the storage location where the archive should be saved.

5. Assign a file name for the archive.
6. Archive the project at the set storage location by clicking the "Archive" button.

Result

A compressed archive with the file extension ".zapVersion" is created. A project archived in TIA Portal V17 has the file extension ".zap17".

Additional references

Detailed information about creating project archives can be found in the information system of the TIA Portal under "Archiving and retrieving projects".

Transferring SIMOTION SCOUT projects

To transfer a SIMOTION SCOUT project to SIMOTION SCOUT TIA, the following possibilities are available:

- Migration with the migration tool in the TIA Portal (complete, including the hardware and network configuration, and the SCOUT data).
- Migration with the migration tool outside the TIA Portal as dual-computer solution and import the SIMOTION SCOUT project using an intermediate file into the TIA Portal.
- Export of the SIMOTION SCOUT project and import into SIMOTION SCOUT TIA (only SCOUT data).

Note

Readiness Check Tool TIA Portal

With the TIA Portal Readiness Check Tool, you can easily check whether the hardware included in your projects is supported by the TIA Portal.

Detailed information about the tool and a download link online are available in the Internet at Readiness Check Tool TIA Portal (https://support.industry.siemens.com/cs/ww/en/view/60162195).

See also

Migrating a SIMOTION SCOUT project (Page 632)

Exporting and importing projects (Page 637)

Examples of export/ import (Page 641)

Migrating a SIMOTION SCOUT project

Migration of a SIMOTION SCOUT project

Note

Migration, including the hardware configuration

You can migrate a SIMOTION SCOUT project only together with the hardware configuration.

A SIMOTION SCOUT TIA project cannot be converted to a SIMOTION SCOUT project. However, you can export the SIMOTION SCOUT TIA data without the hardware configuration and import it into SIMOTION SCOUT by XML export/import. See Section Exporting and importing projects (Page 637). In this case, you must manually create the hardware configuration in SIMOTION SCOUT so that it is identical to the TIA Portal project.

Note

Alarm_S messages

Alarm_S messages are incremented in the number range starting at 60,000,000 hex for the migration from SIMOTION SCOUT to SIMOTION SCOUT TIA.

As of SIMOTION SCOUT TIA V5.2, you can edit the number value in this range.

Note that the Alarm_S information text is limited to the same characters as the Alarm_S text.

Note that the associated values in the Alarm_S text can be only displayed formatted appropriately for the data type.

Note the TIA Portal specifications for data type formattings for previously configured Alarm_S messages.

If you reconfigure Alarm_S messages in SIMOTION SCOUT, check the message text for compatibility with TIA Portal.

To do this, activate the "Check compatibility with TIA Portal" option.

Detailed information is contained in the information system of the TIA Portal at "Configuring messages".

You must migrate a SIMOTION SCOUT project to use it in SIMOTION SCOUT TIA.

The following procedures are available for this purpose:

- Migration with the migration tool in the TIA Portal (complete, including the hardware and network configuration, and the SCOUT data).
- Migration with the migration tool outside the TIA Portal as dual-computer solution and import the SIMOTION SCOUT project using an intermediate file into the TIA Portal.

Migration in the TIA Portal

During the migration, the project data from a SIMOTION SCOUT project can be converted for processing in SIMOTION SCOUT TIA. The migration is performed in the TIA Portal using the migration tool.

Requirements

The following requirements must be satisfied for a migration:

- The TIA Portal migration tool (integral part of SIMATIC STEP 7 Professional as of V13) and the migration tool plug-in for SIMOTION SCOUT TIA (included on the installation DVD for SIMOTION SCOUT/SIMOTION SCOUT TIA) are installed.
- The TIA Portal migration tool and the migration tool plug-in must be mutually compatible in the version.
- STEP 7 as of V5.5 SP4 is installed.
- SIMATIC STEP 7 Safety Advanced is installed so that Safety projects can be migrated.
- WinCC flexible 2008 as of SP3 is installed provided that an HMI is available.
- SIMOTION SCOUT as of V4.4 is installed.
- SIMOTION SCOUT TIA as of V4.4 is installed.
- SIMOTION SCOUT and STEP 7 must be mutually compatible in the version.
- The SIMOTION SCOUT project is consistent and can be compiled without error.
- You have saved the SIMOTION SCOUT project in SIMOTION SCOUT as of version 4.4.
- The SIMOTION SCOUT project does not use any devices, objects or functionality not supported by SIMOTION SCOUT TIA.
  (See Section Supported devices (Page 584) and Section Supported functionalities (Page 584).)

Note

SIMOTION SCOUT TIA V4.5 and V5.1 in the same project format

TIA Portal projects that you created with SIMOTION SCOUT TIA V5.1 still have the project format of version V4.5.

Once you have used SIMOTION Version V5.1 devices in your TIA Portal project, a TIA Portal project becomes version V14 SP1. You then require a compatible version for the migration tool and the SCOUT version for the migration.

Note

Save and compile in SIMOTION SCOUT

Save and compile the project in SIMOTION SCOUT with the "Save and recompile everything" command before you start the migration.

Procedure

To migrate a SIMOTION SCOUT project, proceed as follows:

1. Start the TIA Portal.
2. Select the "Migrate project" entry in the secondary navigation in the Portal view.
3. Use the "..." button to set the source path to the SIMOTION SCOUT project that you want to migrate.
4. Activate the "Include hardware configuration" option.
5. Assign a name to the migrated project.
6. Click the "..." button to set the target path.
7. Enter any required comments in the comment field.
8. Start the migration with the "Migrate" button.

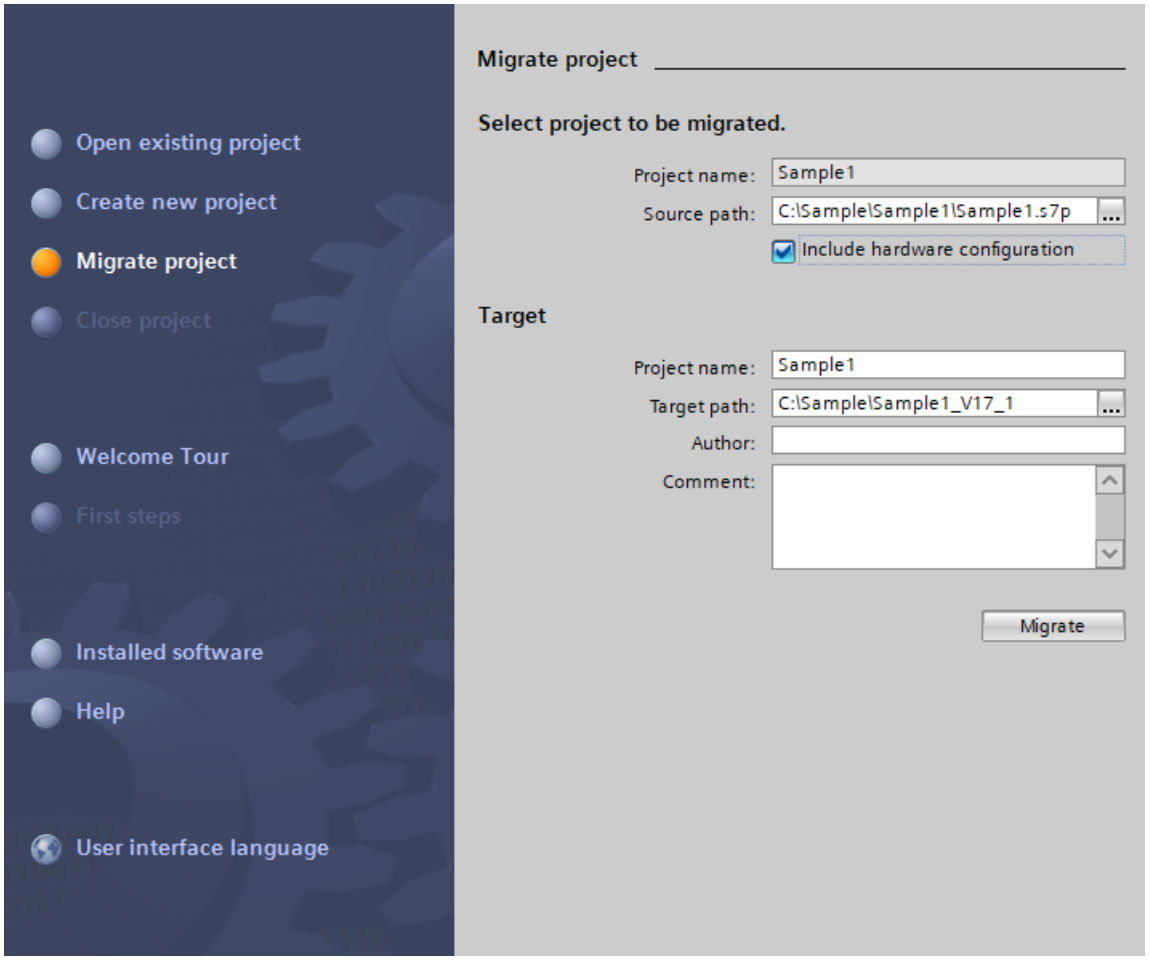


Figure 3-320 Migration

Result

The migrated project opens in the TIA Portal.

You can find the migrated project in the selected target directory.

Note

Save and compile in SIMOTION SCOUT TIA

Save and compile the project in SIMOTION SCOUT TIA with the "Save and recompile everything" command before you start working with the project.

Note

Abort of the migration

The migration is aborted with an error message if the SIMOTION SCOUT project to be migrated uses SIMOTION/SINAMICS devices that are not supported by SIMOTION SCOUT TIA, or the SIMOTION SCOUT project contains objects that are not supported.

In the event of an error, open the project in SIMOTION SCOUT and remove the devices/objects that are not supported, or upgrade the firmware of the devices used.

For information on which devices/objects/functionalities are supported, see Section Supported devices (Page 584) and Section Supported functionalities (Page 584).

Detailed information on the firmware upgrade can be found in Section "Upgrade" of the SIMOTION SCOUT TIA online help.

If the project includes further components in addition to SIMOTION components, such as SIMATIC components and HMI, then the migration conditions specified therein are also applicable.

Migration outside TIA Portal as dual-computer solution

In many cases, the project that you want to migrate is not located on the same PG/PC on which the TIA Portal is installed. The migration tool converts the initial project into a compatible format. This file can be imported and further process in the TIA Portal.

Procedure

To migrate a SIMOTION SCOUT project, proceed as follows:

1. Start the migration tool on computer 1.

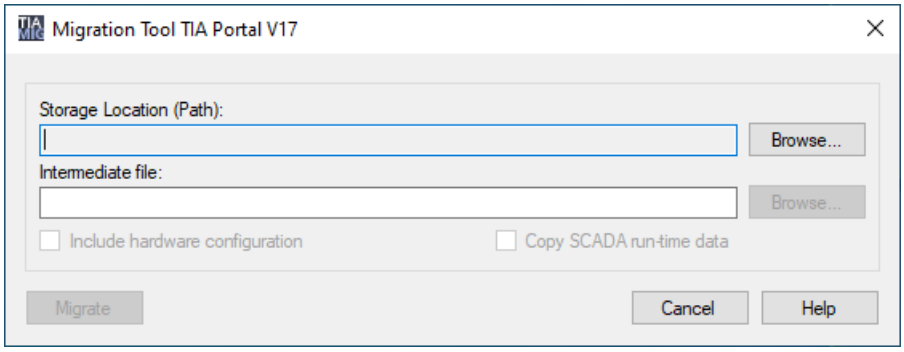


Figure 3-321 Migration tool

2. Navigate to your source project.
3. At "Intermediate file:", set the target path under which the migration file is to be saved.

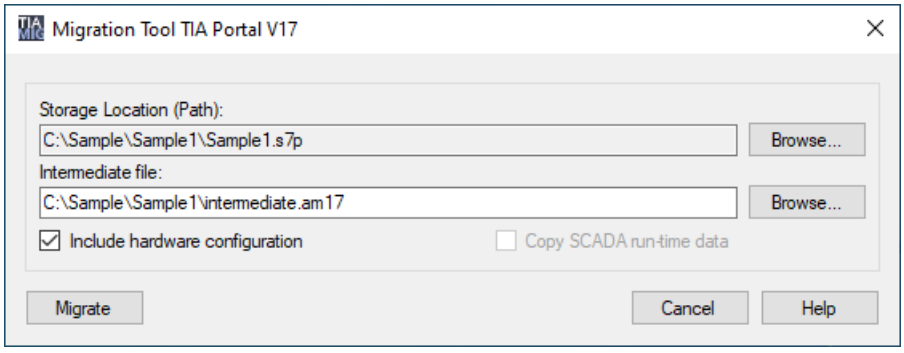


Figure 3-322 Setting the paths

4. Activate the "Include hardware configuration" option (include hardware configuration).
5. Migrate the source project in the migration file format with the appropriate file extension ".amxx".
6. Copy the migration file to the target system, computer 2.
7. Perform the migration within the TIA Portal using the procedure described under "Migration in the TIA Portal".
   Under "Source path", select the generated migration file ".amxx" as the project to be migrated.

Result

The migrated project opens in the TIA Portal.

You can find the migrated project in the selected target directory.

Note

Save and compile in SIMOTION SCOUT TIA

Save and compile the project in SIMOTION SCOUT TIA with the "Save and recompile everything" command before you start working with the project.

Exporting and importing projects

Fundamentals

You can export a whole project or individual projects and subsequently import them into another project.

This functionality is available to you both in SIMOTION SCOUT and in SIMOTION SCOUT TIA. An exchange of project data is thus possible via export and import.

Restrictions

You can only export the SCOUT data with SIMOTION SCOUT TIA because hardware and network configuration is performed in the TIA Portal. Similarly, you can only import projects in SIMOTION SCOUT TIA without the hardware configuration.

Note

Export/import is only possible in offline mode.

Export and import functions

The following functions are available to you in each of the applications:

SIMOTION SCOUT	SIMOTION SCOUT TIA
Export/import of complete projects (with and without hardware configuration)	Export/import of a project without hardware configuration
Export/import of individual devices	Export/import of an existing project without hardware configuration
Export/import of individual data or objects	Export/import of technology objects, watch tables, programs, drive objects, scripts (see restrictions in Section Appendix A (Page 948)), and global device and I/O variables.

The export and import functions for complete projects are described below.

Requirements

Complete projects can only be exchanged between SIMOTION SCOUT and SIMOTION SCOUT TIA via the export/import function if the hardware is identical.

The SCOUT data exported from SIMOTION SCOUT must have identical device properties (name, type, version) in order to be able to import it into SIMOTION SCOUT TIA. From the point of view of configuration, a SIMOTION D435-2 DP V4.5 and a D435-2 DP/PN V4.5 are the same type.

Procedure for exporting from SIMOTION SCOUT

To export a project from SIMOTION SCOUT, proceed as follows:

1. Start SIMOTION SCOUT.
2. Open the project that you want to export.
3. Compile the project.
4. Change to offline mode.
5. Mark the project in the project navigator, right-click to open the context menu and select "Project > Expert > Save and export...".
6. In the next dialog, enter a target directory for the export.

   Note

   Access violation to target directory

   Before the export, all files in the specified target directory are deleted. Therefore make sure that no application accesses a subdirectory or file of the specified target directory (e.g. Windows Explorer). In case of an error, export will be interrupted with an error message. In this case, close the associated application or the opened file and restart the export.

7. If necessary, activate the "Use optimized export format" option.
   In this format, the storage volume is reduced because only the data required for a running system is exported. For example, all comments, minimum and maximum values of parameters from files are suppressed.
8. If necessary, deactivate the "Export, including hardware configuration (STEP 7 data)" option.
9. Confirm with "OK".

Result

The project has been exported into the specified target directory.

Procedure for exporting from SIMOTION SCOUT TIA

To export a project from SIMOTION SCOUT TIA, proceed as follows:

1. Start the TIA Portal.
2. Open the project that you want to export.
3. Switch to SIMOTION SCOUT TIA.
4. Compile the project.

5. Change to offline mode.
6. Mark the project in the project navigator, right-click to open the context menu and select "Project > Expert > Save and export...".
7. In the next dialog, enter a target directory for the export.

   Note

   Access violation to target directory

   Before the export, all files in the specified target directory are deleted. Therefore make sure that no application accesses a subdirectory or file of the specified target directory (e.g. Windows Explorer). In case of an error, export will be interrupted with an error message. In this case, close the associated application or the opened file and restart the export.

8. If necessary, activate the "Use optimized export format" option.
   In this format, the storage volume is reduced because only the data required for a running system is exported. For example, all comments, minimum and maximum values of parameters from files are suppressed.

Procedure for importing into SIMOTION SCOUT

To import XML data for projects that you exported previously from SIMOTION SCOUT or SIMOTION SCOUT TIA, proceed as follows:

1. Open the project into which you want to import the data.
2. Change to offline mode.
3. Mark the project in the project navigator, right-click to open the context menu and select "Project > Expert > Import SCOUT data...".
4. Select the XML file to be imported under "Source path and name of the import file" in the next dialog box.
5. Select the target path and target name of the project file.
   This displays the target path and target name of the currently opened project.
6. Confirm with "OK".
7. Compile the project.

Result

You have imported the project.

Procedure for importing into SIMOTION SCOUT TIA

You only import the SIMOTION SCOUT data without hardware configuration into SIMOTION SCOUT TIA. If there is any hardware configuration data in the import file it will be ignored. You can optionally delete any libraries, watch tables, messages contained in the target project when you import. Please note that you can only import SCOUT data if a corresponding hardware configuration exists in the target project. You overwrite the existing SCOUT data when you import.

Note

Backup copy before importing

If necessary, make a backup copy of the project before you import because deleted and replaced data cannot be restored after being imported.

To import the SIMOTION SCOUT data, proceed as follows:

1. Open the project into which you want to import the data.
2. Change to offline mode.
3. Select "Project > Expert > Import SCOUT data..." or select the project in the project navigator and select "Expert > Import SCOUT data..." in the context menu.
4. In the dialog that then opens, select the source path and the source name for the import.
5. Confirm with "OK".
6. Compile the project.

Note

Alarm_S messages

Alarm_S messages are incremented in the number range starting at 60,000,000 hex for the migration from SIMOTION SCOUT to SIMOTION SCOUT TIA.

As of SIMOTION SCOUT TIA V5.2, you can edit the number value in this range.

Note that the Alarm_S information text is limited to the same characters as the Alarm_S text.

Note that the associated values in the Alarm_S text can be only displayed formatted appropriately for the data type.

Configure Alarm_S messages in SIMOTION SCOUT TIA in accordance with the TIA Portal specifications for data type formatting.

If you reconfigure Alarm_S messages in SIMOTION SCOUT, check the message text for compatibility with TIA Portal.

To do this, activate the "Check compatibility with TIA Portal" option.

Note also for previously configured Alarm_S messages the TIA Portal specifications for data type formatting.

Detailed information is contained in the information system of the TIA Portal at "Configuring messages".

Result

You have imported the project.

Additional references

You will find detailed information about exporting and importing individual objects or devices, for example, in section "Exporting and importing" of the SIMOTION SCOUT TIA online help.

See also

Examples of export/ import (Page 641)

Migrating a SIMOTION SCOUT project (Page 632)

Examples of export/ import

The migration of a project via export and import by two examples is illustrated below.

Export/import of SIMOTION SCOUT data to SIMOTION SCOUT TIA (Example 1)

A project of SIMOTION SCOUT is migrated to SIMOTION SCOUT TIA via export and import of the SIMOTION SCOUT data.

Requirements

- A project with two C240 CPUs with distributed synchronization via PROFIBUS or PROFINET.
- A SIMOTION D CPU with an axis interconnected with a drive to the SINAMICS Integrated.
- An ST program with a P1 program that runs in the BackgroundTask.
- The project can be compiled without errors.

Procedure

1. Export the project to SIMOTION SCOUT.
   – Deactivate the "Export including hardware configuration (STEP 7 data)" option during export.
2. Create a new project in the TIA Portal.
3. Configure the hardware in the TIA Portal same as in SIMOTION SCOUT.
4. Open SIMOTION SCOUT TIA and import the previously exported SIMOTION SCOUT data.
5. Compile the project.

Result

- The distributed synchronization is established.
- The connection of the axis to the drive is established.

- Program P1 is entered in the execution system (BackgroundTask).
- The project can be compiled without errors.

Export/import of SIMOTION SCOUT TIA data to SIMOTION SCOUT (Example 2)

A project of SIMOTION SCOUT TIA is migrated to SIMOTION SCOUT via export and import of the SIMOTION SCOUT data.

Requirements

Modify the sample project previously imported into SIMOTION SCOUT TIA as follows:

- Move the program P1 from the BackgroundTask to the MotionTask_1.
- Add a further drive to the SINAMICS Integrated.
- The axis is connected with the new drive.
- The project can be compiled without errors.

Procedure

1. Export the project to SIMOTION SCOUT TIA.
   – SCOUT data without hardware configuration
2. Start SIMOTION SCOUT.
3. Open the previously exported project from example 1.
4. Import the previously exported project from example 2.
5. Compile the project.

Result

- The axis is connected with the new drive.
- The program P1 is entered in the MotionTask_1.
- The program can be compiled without errors.

See also

Commissioning the drives (Page 824)

Converting projects

Project conversion is necessary when a SIMOTION SCOUT TIA project was created with an older version of the TIA Portal (<V14) and is to be processed further in a newer version of the TIA Portal (V15.1).

The conversion creates a copy of the project for the current version. The project data is converted and saved in the current version.

Note

Converting projects from TIA Portal V13 to TIA Portal ≥V14

To convert a SIMOTION SCOUT TIA V4.4 project from TIA Portal V13 to a SIMOTION SCOUT TIA V4.5 project in TIA Portal V14 or to a SIMOTION SCOUT TIA V5.4 project in TIA Portal V17, you must first upgrade the project to V13 SP1 of the TIA Portal.

Upgraded projects can no longer be edited with the old TIA Portal version.

Note

Converting projects from TIA Portal V14 to TIA Portal V14 SP1

In versions V14 and V14 SP1 of the TIA Portal, you can open and process projects alternately within a SCOUT TIA version provided you do not use any new functions.

If in TIA Portal V14 SP1 you use new functions or devices in your project or different SCOUT TIA versions, you must upgrade the project.

Upgraded projects can no longer be edited with the old TIA Portal version.

Note

Global libraries

If the project contains libraries, they must also be upgraded.

When a global library in the current version of the TIA Portal opens, a copy of the library is created and it is upgraded to the current version of the TIA Portal.

3.5.5.3 Configure the device.

Adding a SIMOTION device

Note

You can add a SIMOTION device only in the TIA Portal.

A SIMOTION device can be added in several ways.

SIMOTION SCOUT TIA does not support all devices that can be configured with SIMOTION SCOUT.

To find out which SIMOTION devices are supported, see Supported devices (Page 584).

To find out which functionality is supported, see Supported functionalities (Page 584).

Procedure

To add a SIMOTION device via the project tree, proceed as follows:

1. Switch to the project view.
2. Double-click "Add new device" in the project navigation.
   The "Add new device" dialog opens.
3. Under "Controller," open the desired device variant, e.g. SIMOTION D - Drive based.
4. Select the desired SIMOTION device.
5. To display device-specific information, click the article number.
6. Select the firmware version of the installed SIMOTION device at "Version".

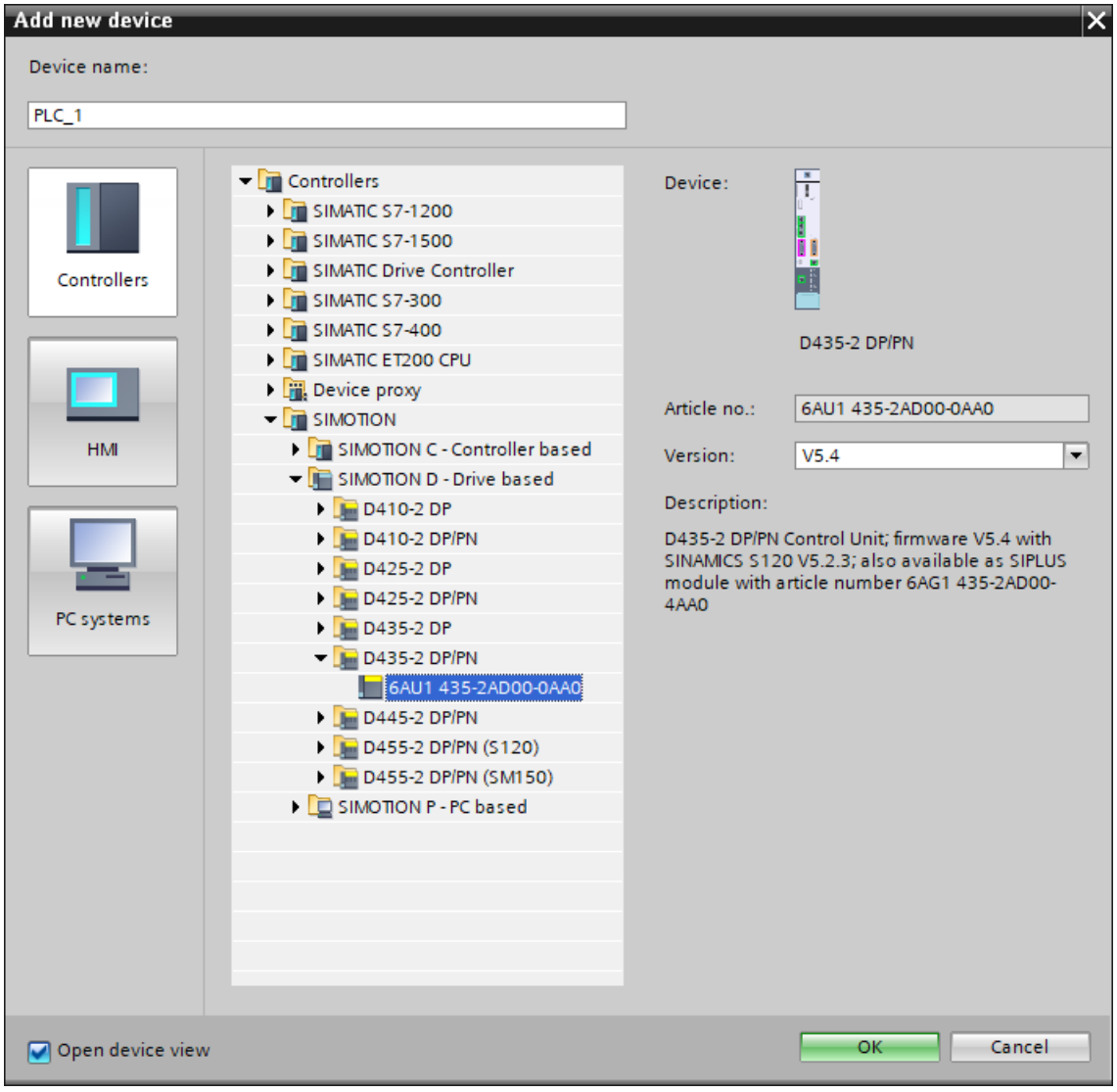


Figure 3-323 Project tree - Add device

7. Click "OK" to apply the settings and add the SIMOTION device.

Result

You have now created a SIMOTION device.

In the device view, the SIMOTION device is displayed in full graphics, showing all interfaces and properties.

In the Inspector window at "Properties", you can configure and parameterize the interfaces easily and intuitively.

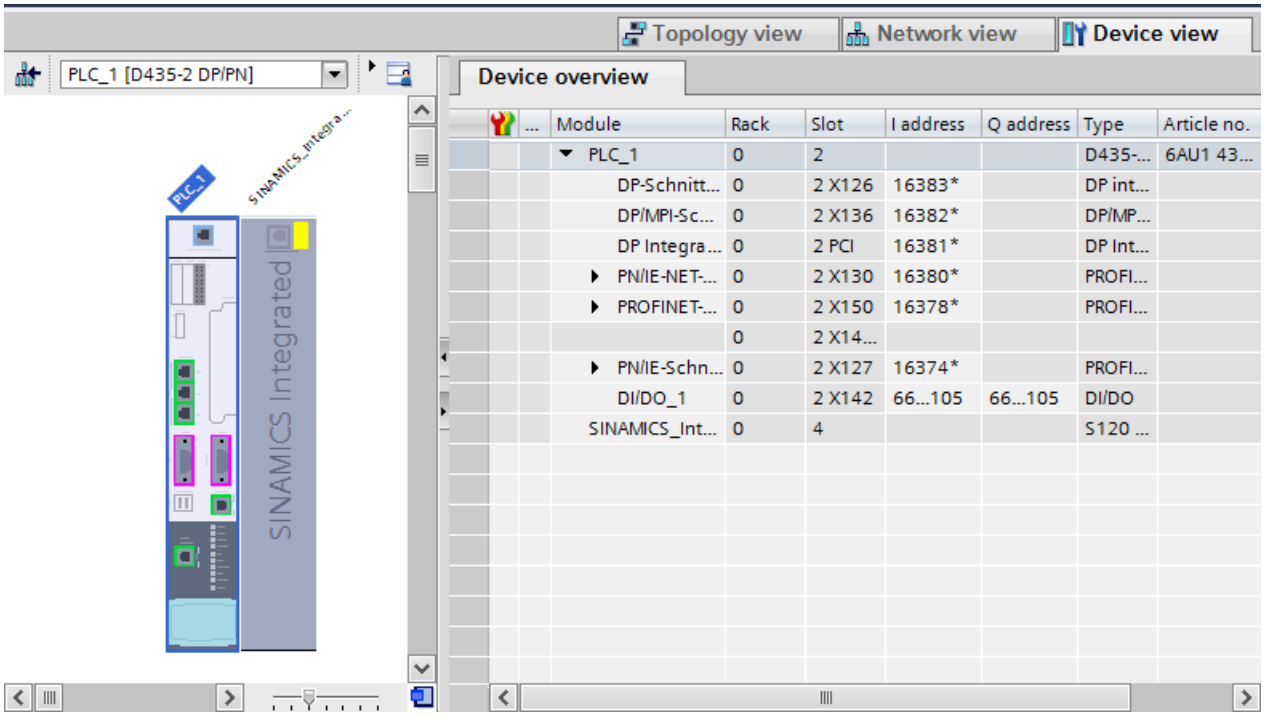


Figure 3-324 SIMOTION device in the device view

All inserted devices are also visible in the project tree.

As soon as you have inserted the SIMOTION device in the TIA Portal, it is also visible in SIMOTION SCOUT TIA.

Note

The selected firmware version must match the firmware version on the memory card of the device, otherwise you receive an error message when you switch to online mode.

Replacing a SIMOTION device

Note

You can replace a SIMOTION device only in the TIA Portal.

Procedure

To replace a SIMOTION device, proceed as follows:

1. In the TIA Portal, select the SIMOTION device that you want to replace in one of the views (topology view, network view or device view).
2. In the context menu, select the "Replace device..." command.
   The "Replace device" dialog opens.

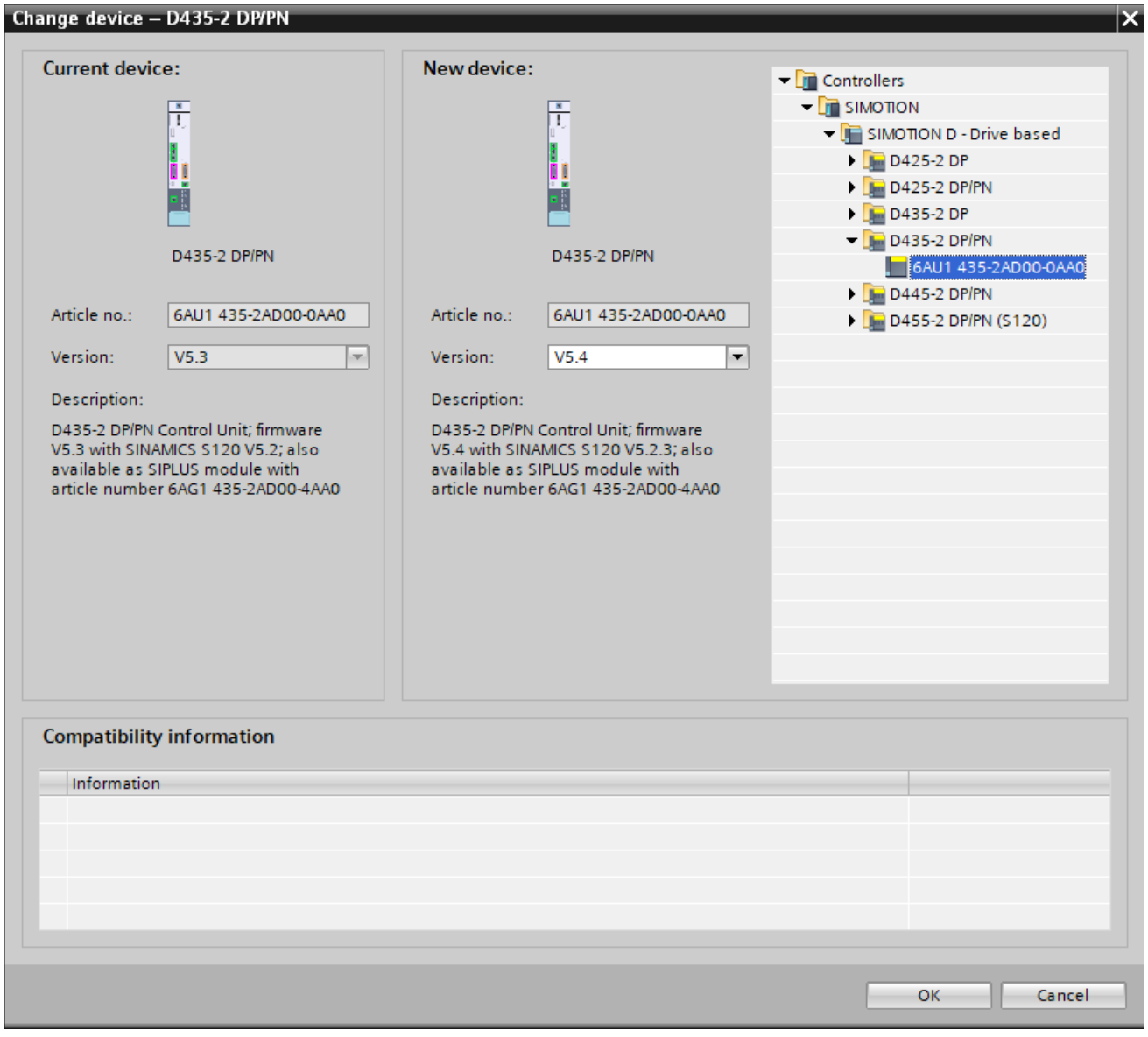


Figure 3-325 Replace device

   – The device that you want to replace is displayed in the "Current device:" area.
   – In the right-hand area, only those devices are listed that are compatible with the device that you want to replace.
3. In the right-hand area, select the new device.
   The selected device is displayed in the "New device:" area.

4. Select the correct firmware version at "Version".
   Observe the compatibility information.
5. Close the dialog with the "OK" button.
6. Save the project.

Result

You have replaced a SIMOTION device.

When you replace the device in the TIA Portal and save the project, it will also be replaced in SIMOTION SCOUT TIA.

Note

Devices must be mutually compatible.

You can only replace devices with devices that are compatible. If you select an incompatible device or an incompatible firmware version, the "OK" button remains inactive.

A SIMOTION D can be replaced with another SIMOTION D only when the associated SINAMICS version is the same or higher. It is not possible to downgrade to an earlier SINAMICS version

Replacing hardware components may result in inconsistencies in the project. The consistency check indicates any inconsistencies.

Eliminate these inconsistencies as appropriate.

Deleting a SIMOTION device

Note

You can delete a SIMOTION device only in the TIA Portal.

Procedure

To delete a device, proceed as follows:

1. Select the device you want to delete from one of the following views.
   - Device view
   - Network view
     In the graphical view or in the "Network overview" tab.
   - Topology view
     In the graphical view or in the "Topology overview" tab.
   - Project navigation
     In the tree structure of the project.
2. Press "Del".
3. Save the project.

Result

The SIMOTION device has been deleted.

Once you have deleted the device in the TIA Portal, it is also deleted in SIMOTION SCOUT TIA.

Note

Inconsistencies caused by deleting hardware components

Deletion of hardware components may lead to inconsistencies in the project.

The inconsistencies are indicated in the consistency check.

Eliminate these inconsistencies as appropriate.

Configuring an Ethernet interface

Creating an Ethernet subnet

You set up online communication of the PG/PC with the SIMOTION device via PROFIBUS, PROFINET, or Industrial Ethernet.

The most common application is the communication via Industrial Ethernet (communication via PROFINET protocol).

To configure I/O devices, you must create a subnet. The following step shows how a subnet is created.

Requirement

You have connected the PG/PC and the SIMOTION device with an Ethernet cable.

Procedure

To add a subnet to the SIMOTION device in the network view, follow these steps:

1. Click the Ethernet "X127" interface of the SIMOTION device where you would like to create a subnet.
2. Open the context menu with a right-click and select the "Add subnet" command.

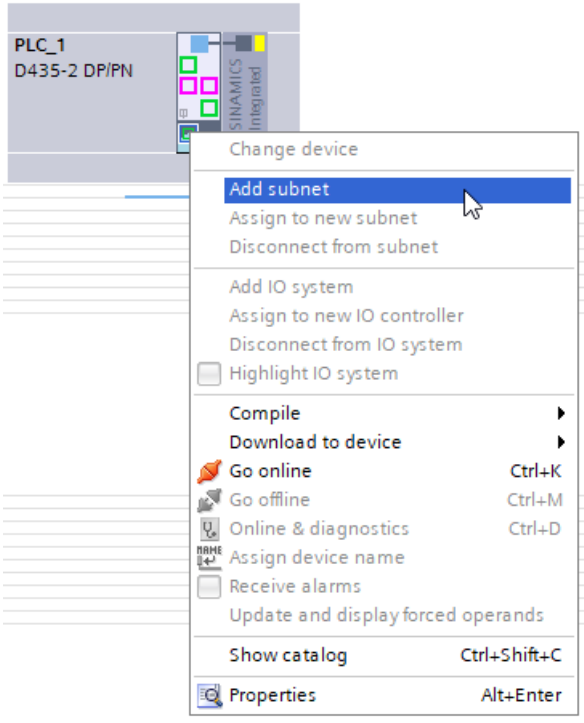


Figure 3-326 Adding a subnet

Result

You have added a subnet.

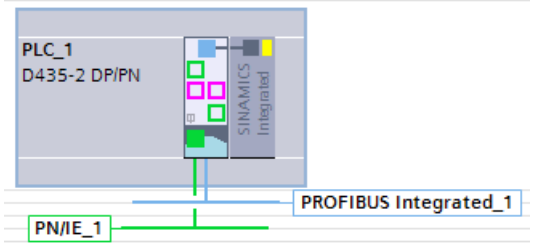


Figure 3-327 Subnet added

Add the expansion module SIMOTION - CX32-2

Note

You can insert a CX32-2 only via the hardware catalog in the TIA Portal.

If the computing performance of SINAMICS Integrated is not sufficient and you wish to use additional drives, you can insert an CX32-2 to increase performance. This module is connected via DRIVE-CLiQ and is thus networked via PROFIBUS Integrated.

Requirement

You have now created a SIMOTION device.

Procedure

To insert a CX32-2 in the network view, proceed as follows:

1. Switch to the "Hardware Catalog" task card and open the "Controller" component.
2. Open the "SIMOTION" folder and then "SIMOTION D - Drive-based".
3. At "Controller extensions", select the CX32-2 that you wish to use.

4. Drag-and-drop the CX32-2 to the PROFIBUS Integrated.
   The "Wiring between controller and CX" dialog opens.

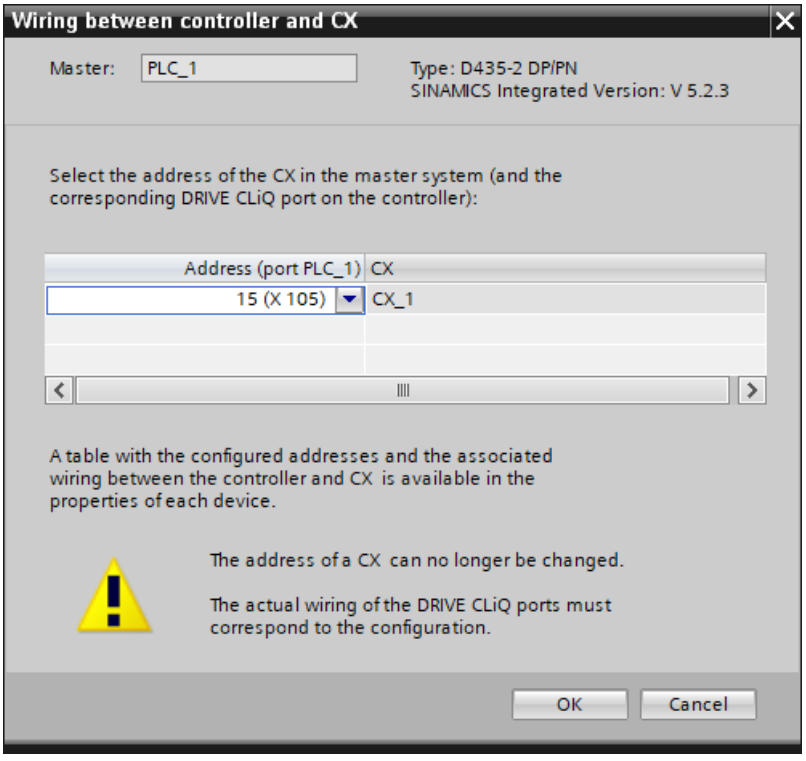


Figure 3-328 CX32-2 wiring

5. Select the address (DRIVE-CLiQ interface) to which your CX32-2 is connected. Make sure that the configured wiring is the same as the real wiring.

DP address of the CX32-2 in the master system	DRIVE-CLiQ socket on the SIMOTION device
10	X100
11	X101
12	X102
13	X103
14	X104
15	X105

6. Click OK to apply the settings.

Note

The configured address cannot be changed.

Once you have created the CX32-2 in the network view, you can no longer change the configured address.

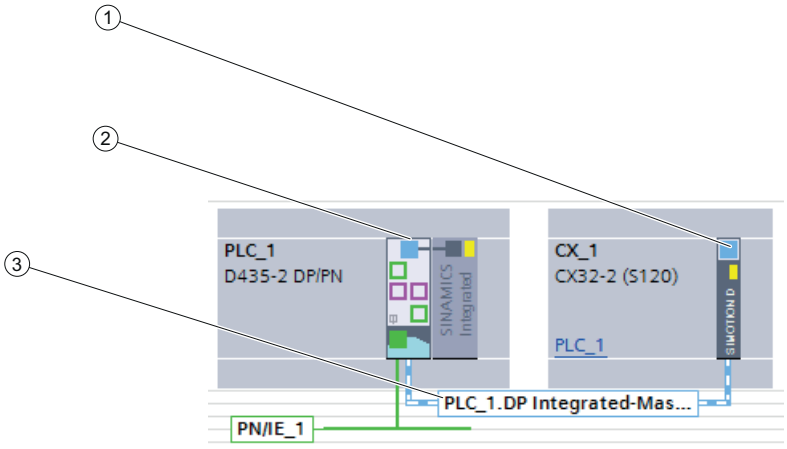
If you want to change the address anyway, you must delete the CX32-2 in the network view and then create a new one.

Result

The CX32-2 is inserted in PROFIBUS Integrated and assigned to the SIMOTION module.

PROFIBUS Integrated and the interfaces are configured in the "Properties" tab in the Inspector window.

The following figure shows a CX32-2 on a SIMOTION D435-2.



① DRIVE-CLiQ interface CX32-2
② DRIVE-CLiQ interface SIMOTION D435-2
③ PROFIBUS Integrated

Note

Configuring the CX32-2

Configure the CX32-2 interface in SIMOTION SCOUT TIA.

Using communication modules

CBE20

The device is connected to PROFINET IO using the CBE20 communication board for SINAMICS S120. The module supports PROFINET IO with isochronous Realtime Ethernet (IRT), PROFINET IO with RT, and standard TCP/IP communication.

The following SINAMICS drive devices support the onboard version.

- SINAMICS S120 CU320-2 DP CBE20 V4.5
- SINAMICS S120 CU320-2 DP CBE20 V4.7
- SINAMICS S120 CU320-2 DP CBE20 V4.8

- SINAMICS S120 CU320-2 DP CBE20 V5.1
- SINAMICS S120 CU320-2 DP CBE20 V5.1.1
- SINAMICS S120 CU320-2 DP CBE20 V5.2
- SINAMICS S120 CU320-2 PN CBE20 V4.5
- SINAMICS S120 CU320-2 PN CBE20 V4.7
- SINAMICS S120 CU320-2 PN CBE20 V4.8
- SINAMICS S120 CU320-2 PN CBE20 V5.1
- SINAMICS S120 CU320-2 PN CBE20 V5.1.1
- SINAMICS S120 CU320-2 PN CBE20 V5.2
- SINAMICS S120 CU320-2 PN CBE20 V5.2.3

When selecting a drive unit from the hardware catalog, you can decide whether to use the CBE20 communication board.

Choose the appropriate variant of the SINAMICS S120 CU320-2. You can find the CBE20 communication modules in the hardware catalog under "Controllers > SIMOTION > SIMOTION drives > SINAMICS S120".

CBE30-2

A second PROFINET interface can be implemented for the SIMOTION D4x5-2 DP/PN with the CBE30-2 communication board.

It is not possible to use a CBE30-2 in SIMOTION D4x5-2 DP controllers.

Adding a communications module

To add the CBE30-2 communication module, proceed as follows:

1. Select the SIMOTION device.
2. Switch to the device view.
3. Switch to the "Hardware Catalog" task card and open the "Controller" component
4. Open the folder "SIMOTION D – Drive-based".
5. Select the CBE30-2 communication module in the "Communication modules" folder under "PROFINET".
6. Drag-and-drop the communication board into the device view onto the interface of the device.

Inserting a SIMOTION drive

Note

You can insert a SIMOTION drive only via the hardware catalog in the TIA Portal.

A SIMOTION drive in the current version of SIMOTION SCOUT TIA is a drive of the SINAMICS S120 type (interconnection via PROFINET or PROFIBUS) interconnected with a SIMOTION CPU. The explicit SIMOTION drives you can insert can be found in Supported devices (Page 584).

These SIMOTION drives can be configured and parameterized only under these conditions (interconnection with SIMOTION CPU). The latter is configured in SIMOTION SCOUT TIA.

Procedure

To insert a SIMOTION drive (SINAMICS S120), proceed as follows:

1. Insert a SIMOTION device, if not available yet.
2. Create the bus system, e.g. DP master system or PN/IO system.
3. Navigate in the hardware catalog to the "Controller" component and open the "SIMOTION" folder.
4. Open the "SINAMICS S120" sub-folder in the "SIMOTION drives" folder and select the desired drive, in the example "CU310-2 PN".
5. Drag-and-drop the drive to the bus system.
6. Assign the master system.
   To do this, click "Unassigned" and select the device with which the drive is to be networked.

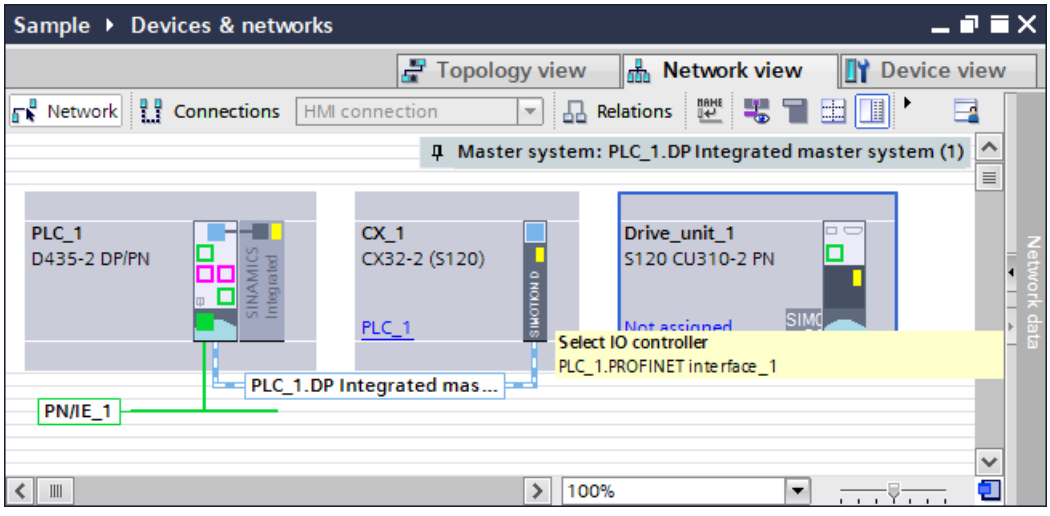


Figure 3-329 Networking the drive

Result

You have networked the drive with the SIMOTION device via the bus system.

The settings on the bus system are adapted in the "Properties" tab in the Inspector window.

Some the following settings can be made:

- Specify the transmission rate
- Configure the line
- Set the bus parameters and constant bus cycle time.

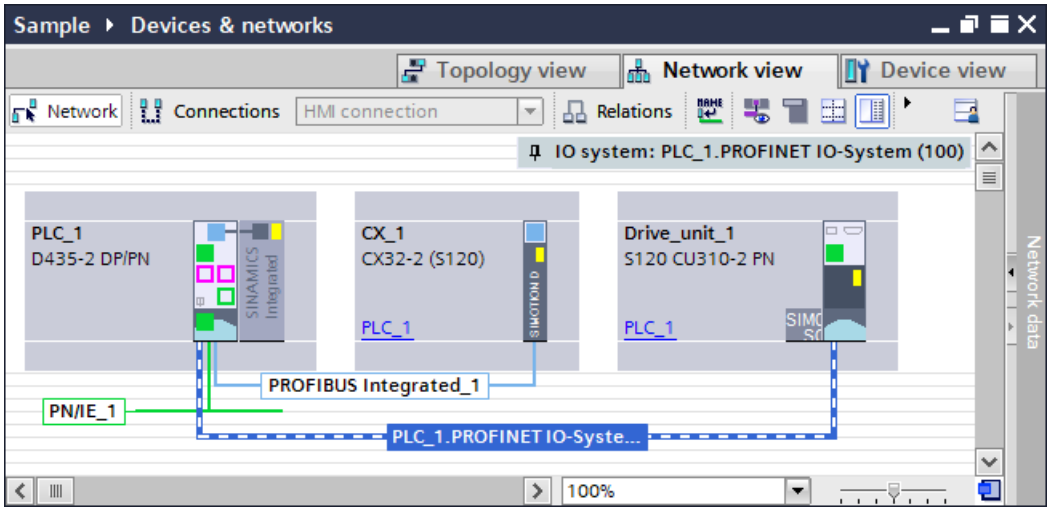


Figure 3-330 Networked drive

Note

The drive is configured in SIMOTION SCOUT TIA.

Inserting a Startdrive drive

Requirements

- You have installed the compatible "Startdrive" option package for the TIA Portal.
- You have now added a SIMOTION device.

Procedure

Inserting a drive and assigning it to a controller

To insert a Startdrive drive (e.g. SINAMICS G120, SINAMICS G110M,...), proceed as follows:

1. Add a higher-level controller, e.g. SIMOTION D435-2.
2. Switch to the network view.
3. Double-click "Add new device" in the project navigation.
   The "Add new device" dialog opens.
4. Open the variant of the drive you require at "Drives".
5. Then select the desired drive.

6. To display device-specific information, click the drive.
7. Select the firmware version of the drive at "Version".

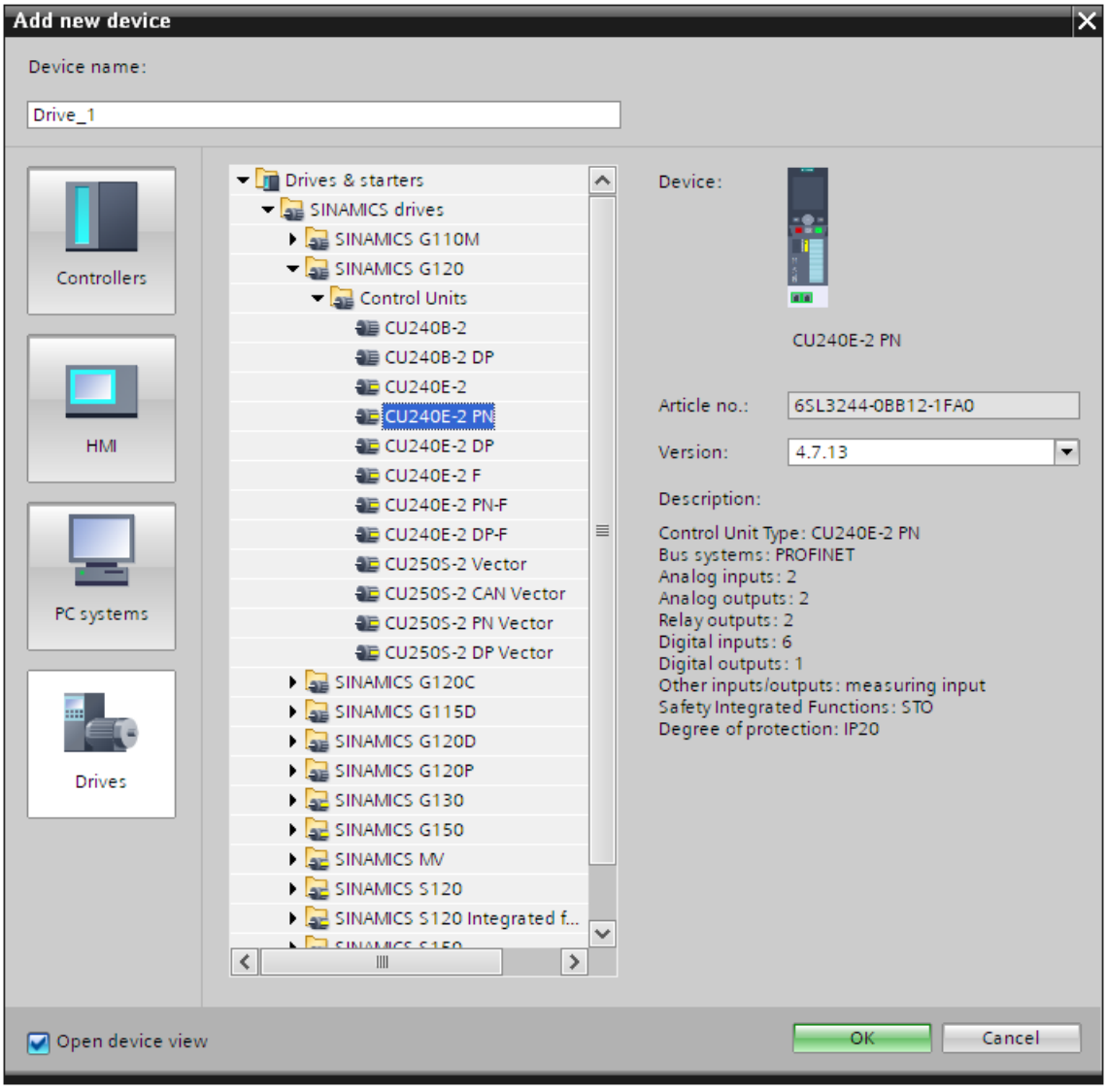


Figure 3-331 Selecting a Startdrive drive

8. Click "OK" to apply the settings and add the Startdrive drive.

Adding a Power Module

To add a Power Module, proceed as follows:

1. Switch to the device view.
2. Switch to the "Hardware Catalog" task card and open the "Power Modules" component.
3. Select the Power Module that you want to use, a PM240 in the example.

4. Drag-and-drop the Power Module onto the drive.
5. Assign the master system. To do this, click "Unassigned" and select the device with which the drive is to be networked.

Result

You have created a Startdrive drive and added the Power Module.

You have networked the drive with the SIMOTION device.

The device view displays a full graphical view of the Startdrive drive, showing all interfaces and properties.

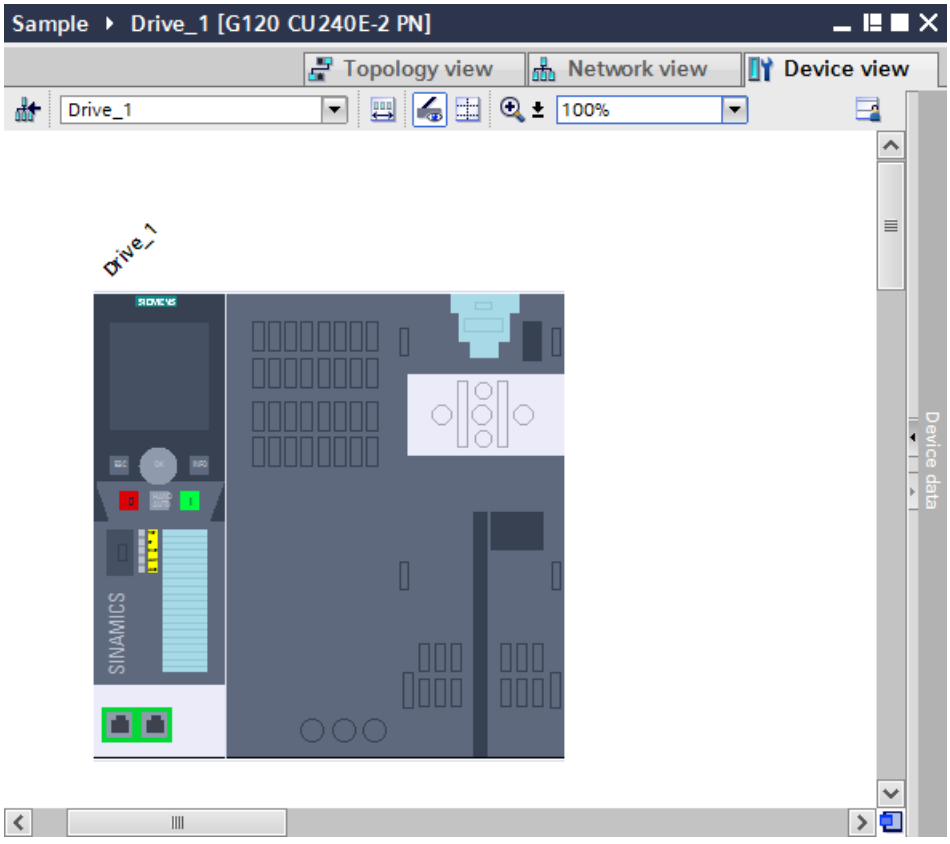


Figure 3-332 Startdrive drive in the device view

The inserted Startdrive drive is visible in the project tree.

As soon as you have inserted the Startdrive drive in the TIA Portal, it will also be visible in SIMOTION SCOUT TIA.

Note

Interconnecting and configuring Startdrive drives

Like GSD drives, Startdrive drives can be interconnected with technology objects in SIMOTION SCOUT TIA. The drives are configured with Startdrive.

Inserting a GSD drive

If you are using a drive that cannot be configured via SIMOTION SCOUT TIA or Startdrive, insert this drive as a GSD drive. Various Siemens drives, such as SINAMICS S150 CU320-2, are available as a GSD drive in the TIA Portal.

Procedure

Inserting a GSD drive and assigning it to a controller

To insert a GSD drive and assign it to a controller, proceed as follows:

1. Add a higher-level controller, e.g. SIMOTION D435-2.
2. Switch to the network view.
3. Switch to the "Hardware Catalog" task card and open the "Other field devices" component.

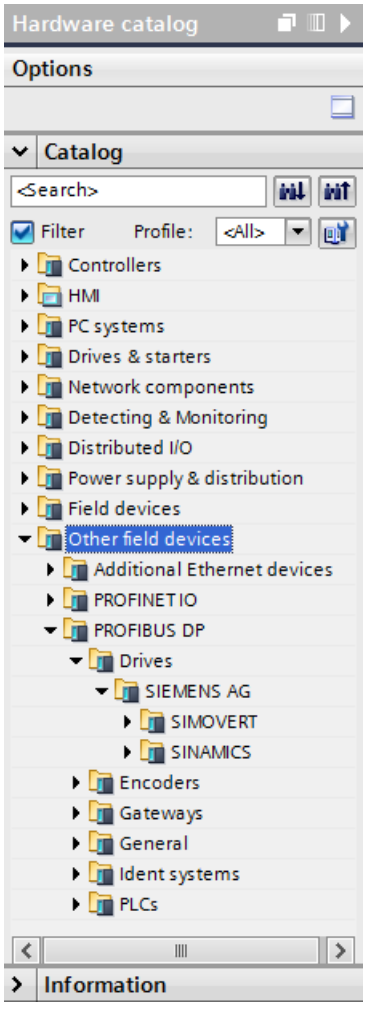


Figure 3-333 Selecting field devices

4. Select the GSD drive in the hardware catalog, SINAMICS GM150 in the example.

5. Drag-and-drop the GSD drive to the "Network view".
6. Assign the master system.
   To do this, click "Unassigned" and select the device with which the GSD drive is to be networked.

Configuring telegrams for the communication

Communication between the controller and the drive is performed via PROFIdrive telegrams. Different telegrams are available, depending on the task.

To add a telegram for the communication between controller and drive, proceed as follows:

1. Select the GSD drive and switch to the device view.
2. Display the device overview.
3. Switch to the "Hardware catalog" task card.
   The telegrams available for this drive are displayed.

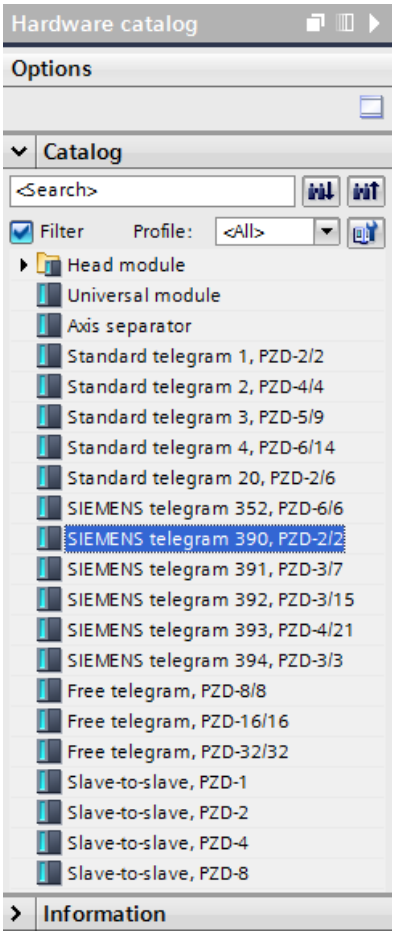


Figure 3-334 Hardware catalog telegrams

4. Select a telegram.
   In the device view, the possible positions for insertion are indicated by blue lines.

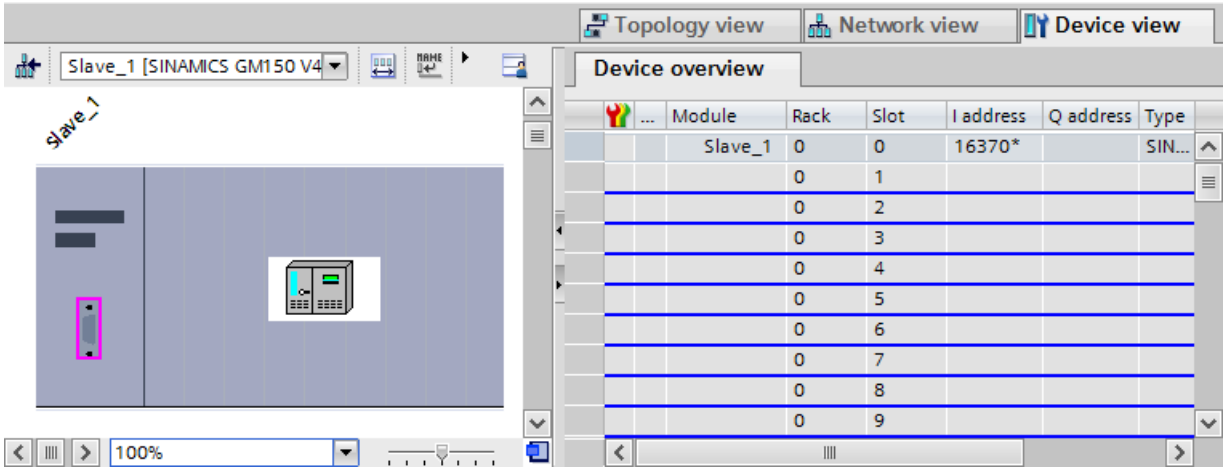


Figure 3-335 Selecting a telegram

5. Drag-and-drop the telegram into the device overview and the desired position.
   The I/O addresses are assigned automatically.
6. If required, adapt the I/O addresses for the Motion Control applications.

Result

You have created a GSD drive and networked it with the SIMOTION device.

You have inserted a telegram for the communication between controller and drive.

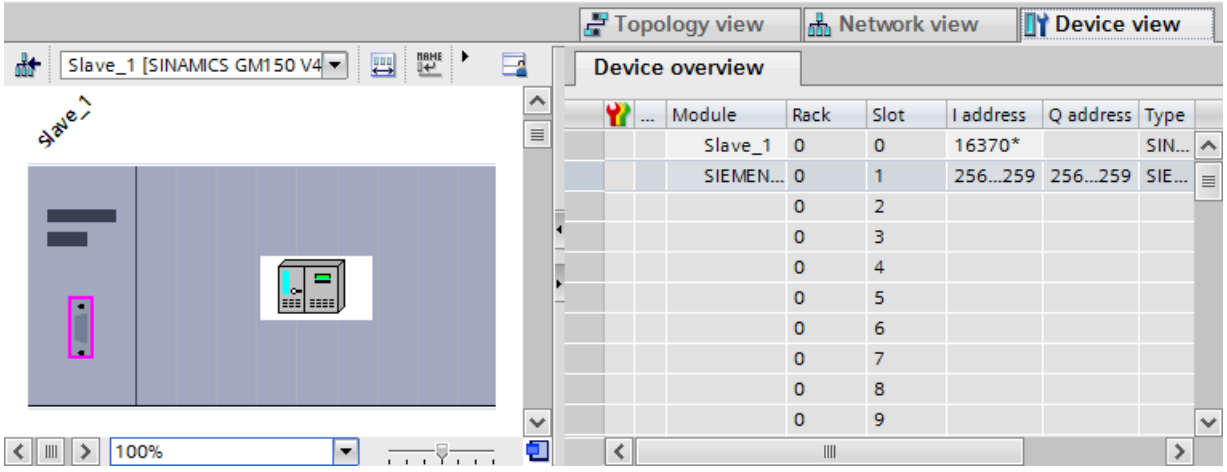


Figure 3-336 Telegram inserted into device overview

As soon as you have inserted the GSD drive in the TIA Portal, it will also be visible in SIMOTION SCOUT TIA.

3.5.5.4 Configuring online access

Online access of the project

Displaying available nodes in the TIA Portal

In the "Online access" folder in the project navigation, you will find all active interfaces at your PG/PC. At each interface icon, you receive information about the current status. You can view the available nodes and use the context menu to show and change the properties of an interface.

Procedure

To display the available nodes on a single interface of the PG/PC, proceed as follows:

1. Open the "Online access" folder in the project navigation.
2. To make all objects ordered below the interface visible, click the arrow at the left next to the "Online access".
3. Double-click "Update accessible nodes" below the interface.
   All nodes that can be accessed via this interface are displayed in the project navigation.

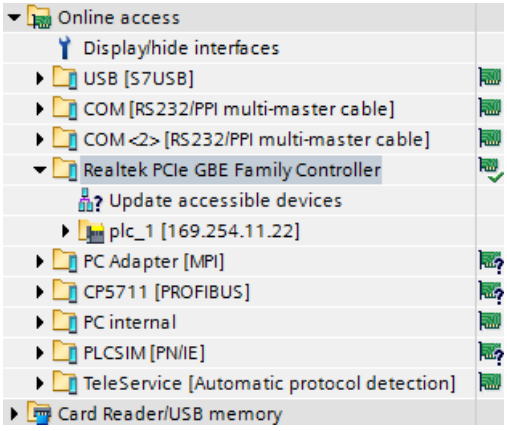


Figure 3-337 Accessible nodes

Setting up the PG/PC communication

Note

You assign the PG/PC interface in the TIA Portal.

Communication between the SIMOTION device and the PG/PC requires a conditioner card (for PROFIBUS) or an Ethernet interface. You configure, parameterize, program and test using the PG/PC.

The following options to assign the PG/PC interface are available:

- "Connect online" function
  The function is available until the PG/PC interface has been successfully set up.
- "Online & diagnostics" function
- "Online access" function

Assign PG/PC interface

The following procedure describes the process for the Ethernet interface type by using the "Online access" function.

To assign the interfaces, proceed as follows:

1. Navigate to the appropriate interface in the project tree at "Online access".
2. Select the "Properties..." command in the context menu.

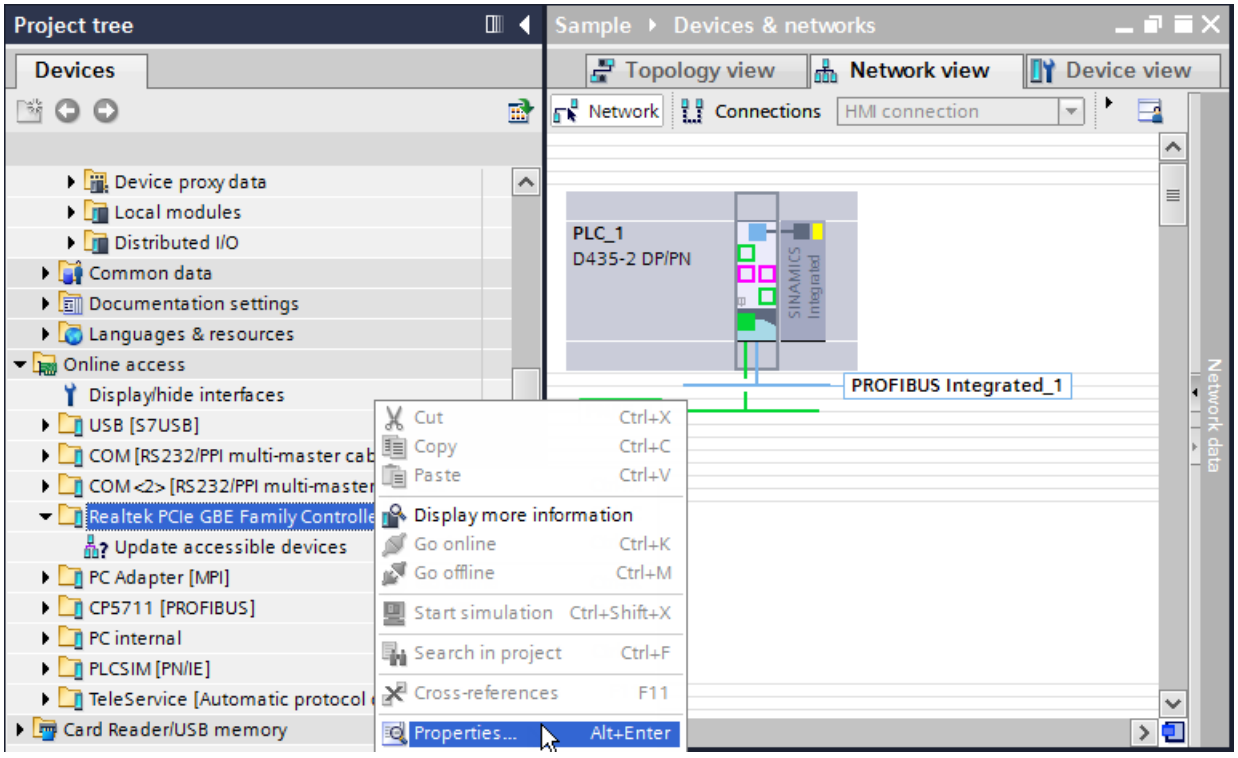


Figure 3-338 Online access properties

3. In the next step, select the subnet and apply the setting with "OK".

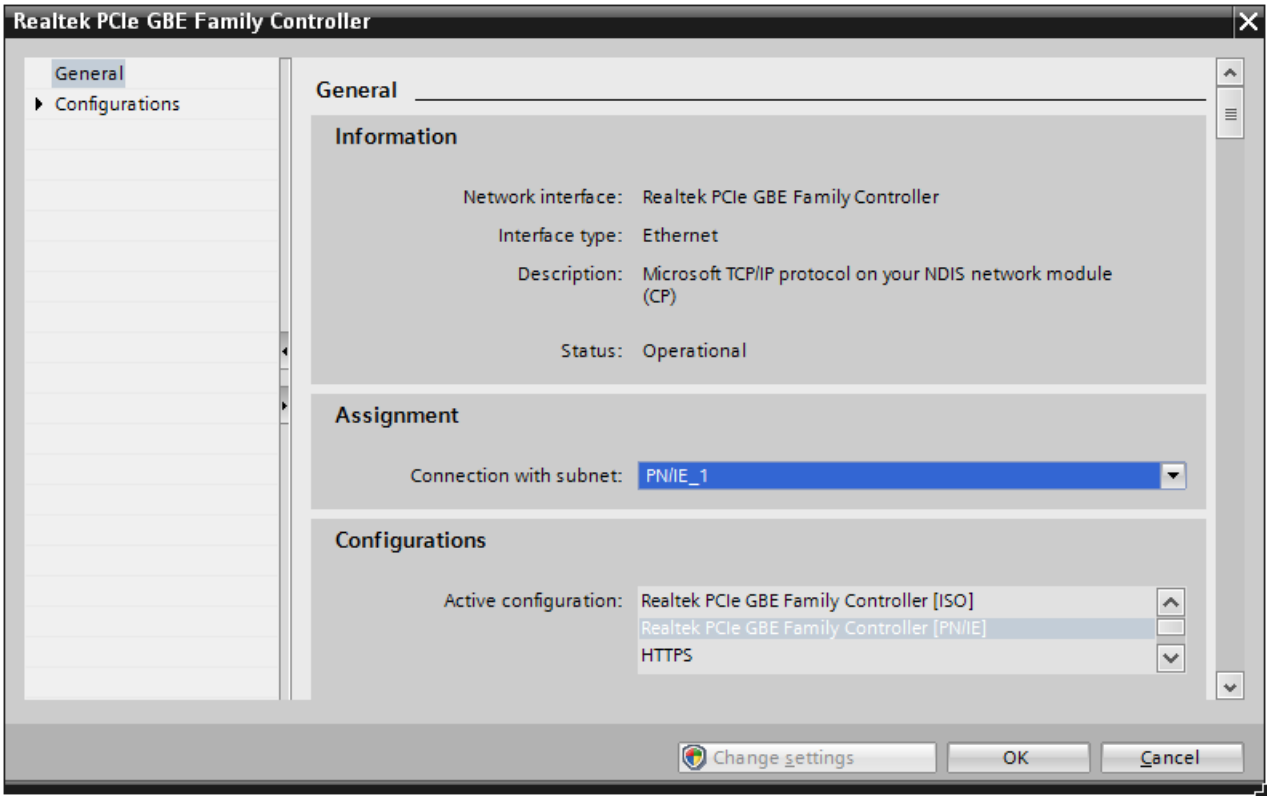


Figure 3-339 Assigning a subnet

Adding an IP address in the subnet

If the SIMOTION device and the PG/PC are not in the same subnet, add a suitable IP address from the subnet of the device to the PG/PC.

To automatically add an IP address in the subnet, proceed as follows:

1. Click Go online in the toolbar.
   The "Connect online" dialog opens.
2. Click "Start search" to search for accessible devices.

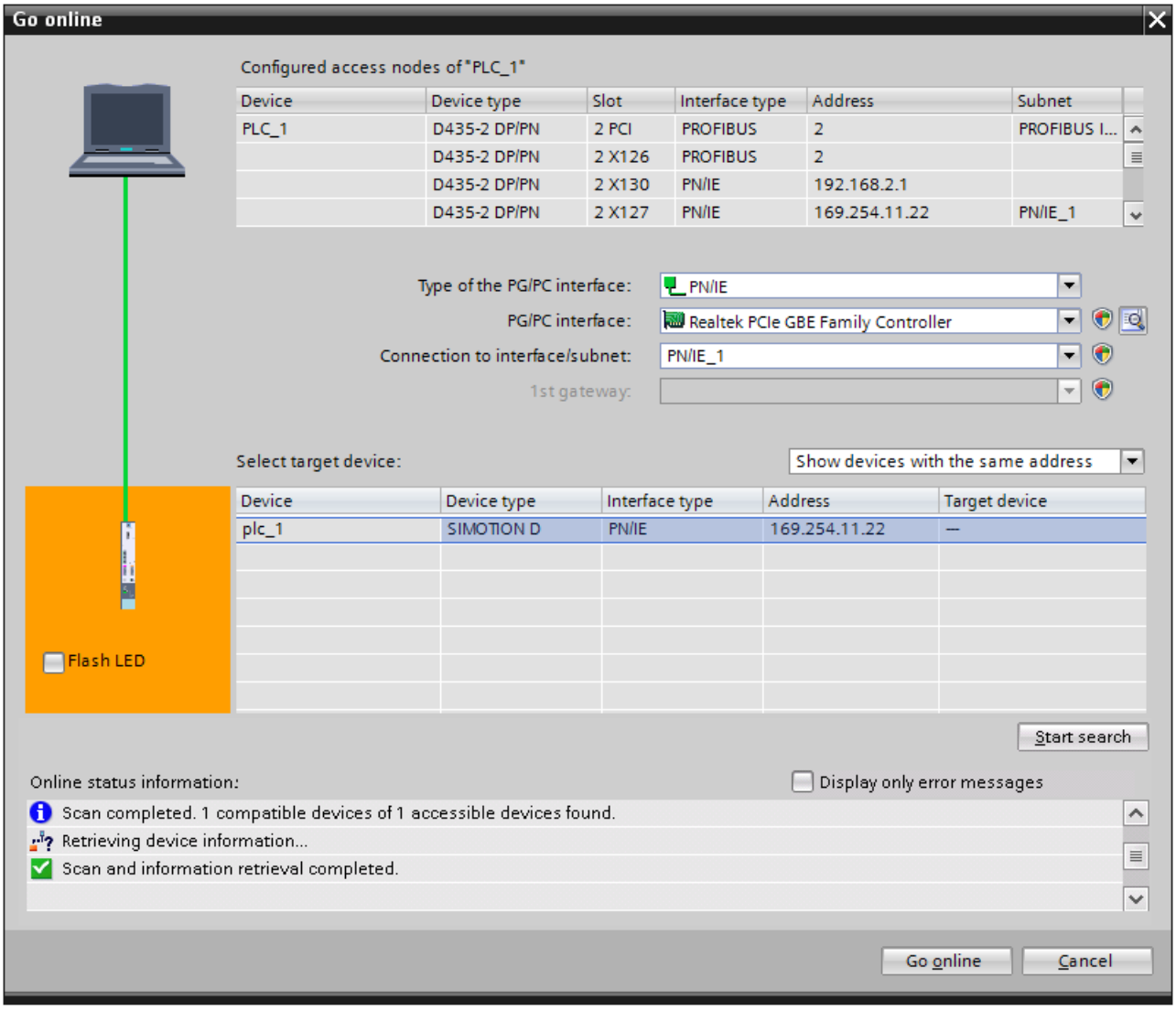


Figure 3-340 Connect online - select a device

3. Click "Connect" in order to establish a connection to the device located by the search.

4. If the SIMOTION device and the PG/PC are not in the same subnet, a message is displayed offering you the option of temporarily assigning a suitable IP address from the subnet of the device.

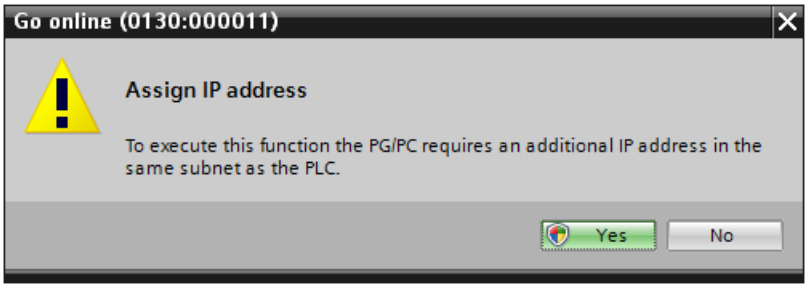


Figure 3-341 Assigning an IP address

5. Confirm with "Yes".

Figure 3-342 IP address assigned

6. To use the connection path specified in the settings as default for the online access, click "Yes".

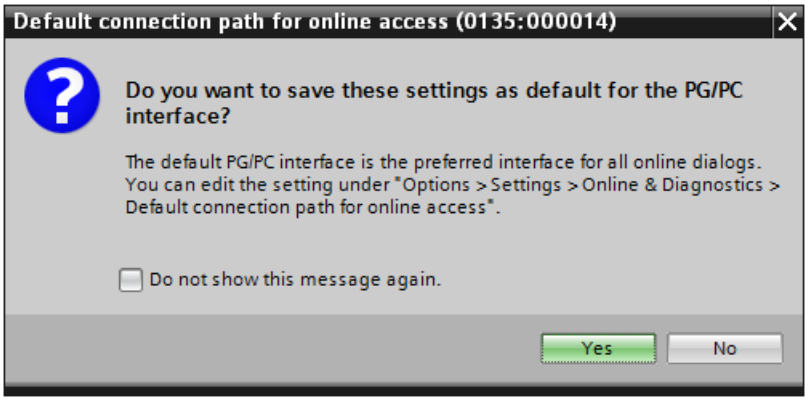


Figure 3-343 Default connection path

Result

- You have assigned the PG/PC interface.
- The TIA Portal has assigned an IP address within a project.
- The online connection has been established.
  The title bar of the project navigation is orange colored.

Displaying and deleting temporary IP addresses

You can display and also delete all temporarily assigned addresses.

To display and delete temporary IP addresses, proceed as follows:

1. Navigate in the project navigator at "Online access" to the appropriate interface.
2. Select "Properties" in the context menu.
3. At Configuration, select the "IE-PG access" entry.

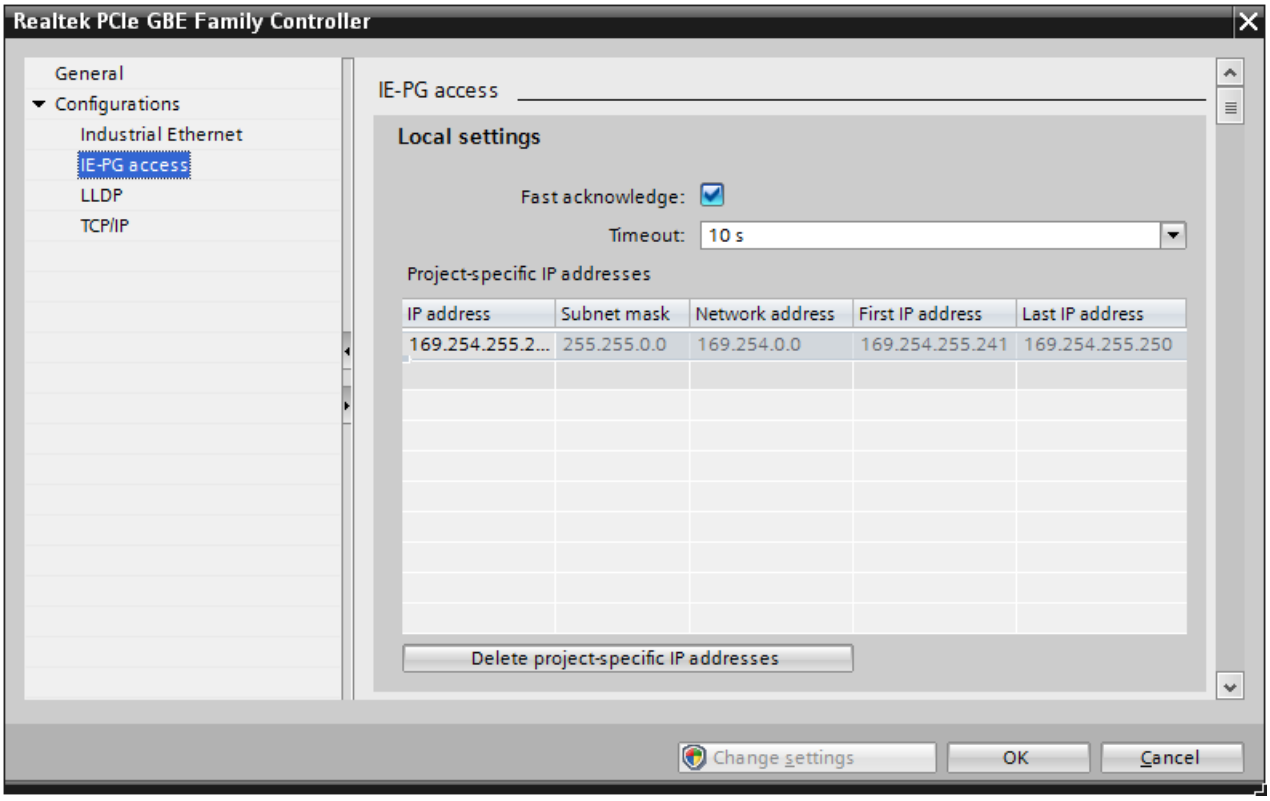


Figure 3-344 Displaying and deleting temporary IP addresses

Compiling the project and downloading the hardware configuration to the SIMOTION device

Note

Downloading the hardware configuration

Before you continue configuring in the SIMOTION SCOUT TIA, download the hardware configuration in the TIA Portal to the SIMOTION device.

Downloading to the SIMOTION device from the TIA Portal is necessary, for example, if you want to change the device names of the IP address. In addition, the loading in the TIA Portal results in the routing information being loaded.

Requirement

The online connection to the SIMOTION device is disconnected.

Procedure

To compile the hardware configuration and download to the SIMOTION device, proceed as follows:

1. Select the SIMOTION device and select "Compile > Hardware (only changes)" in the context menu.

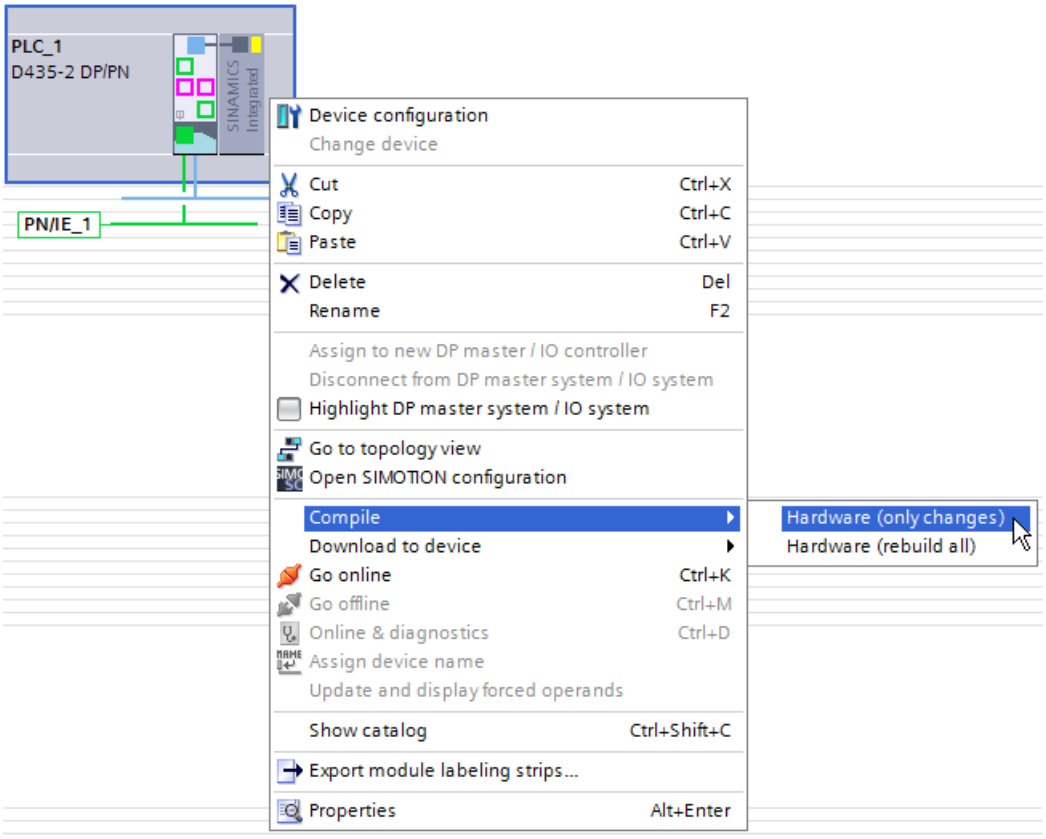


Figure 3-345 Compile hardware

2. Select the SIMOTION device and select "Load to device > Hardware configuration" in the context menu.

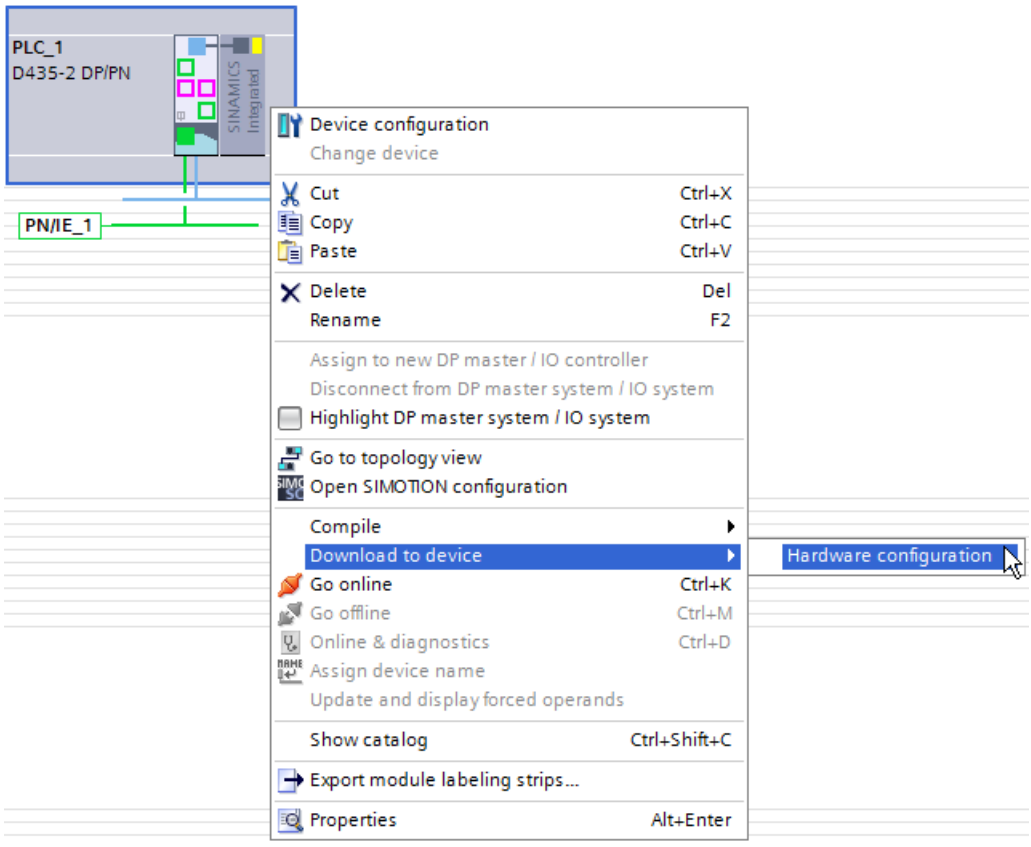


Figure 3-346 Load hardware

– If you have already established an online connection, the "Load Preview" dialog opens. In this dialog, messages are displayed and actions required for loading suggested.
– If you have not yet established any online connection, the "Advanced Load" dialog opens and you must first select the interfaces with which the online connection to the device should be established.

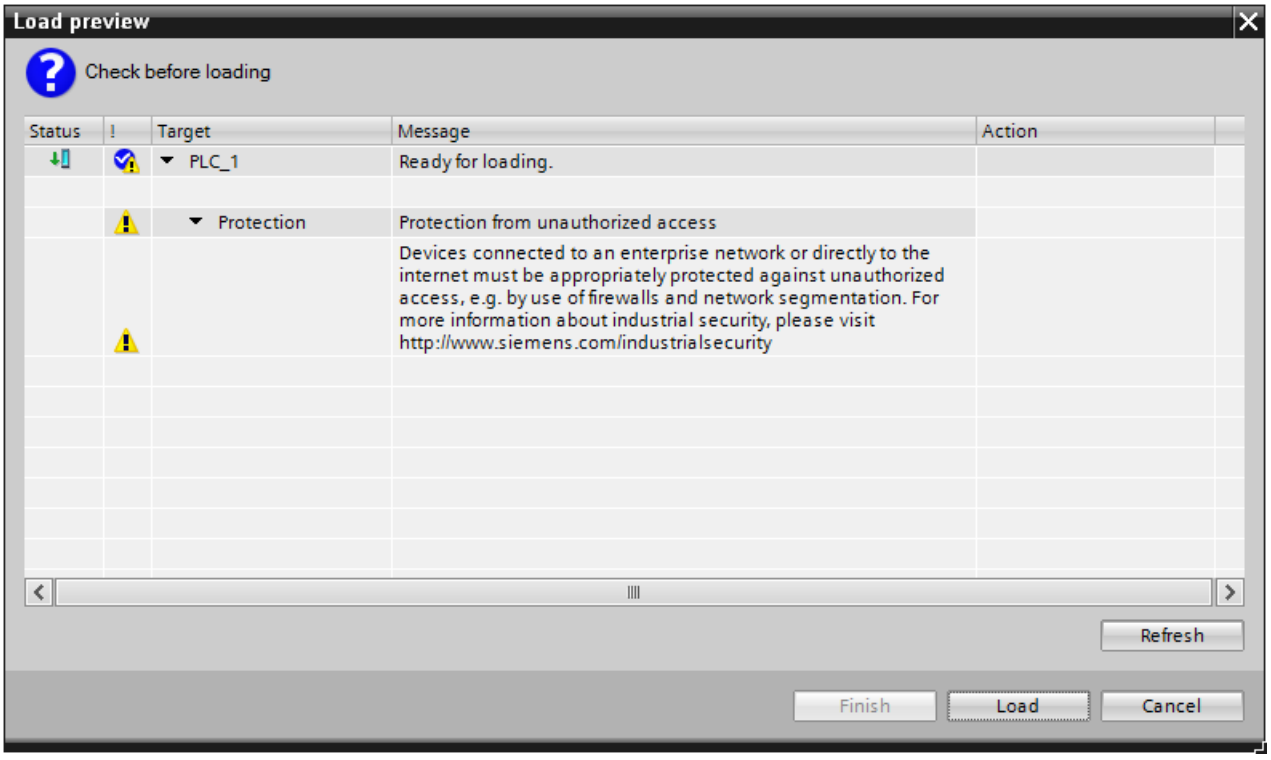


Figure 3-347 Loading the preview

3. Click "Load".

Result

You have compiled the current hardware configuration and loaded it to the target device.

Additional references

You will find a detailed description of how to set up the Ethernet interfaces in SIMOTION SCOUT TIA in the online help "SIMOTION SCOUT TIA Getting Started".

For detailed information about the configuration of the online access, refer to the information system of the TIA Portal "Configuring online access".

3.5.5.5 Diagnostics and functions

Overview

The following device diagnostics functions are available, for example, via the online and diagnostics view:

- Display diagnostics status of a module
- Read out diagnostics buffer of a CPU
- Assign an IP address to a PROFINET IO device

- Determine and set the time of a CPU
- Assign PROFINET device name
- Reset the parameters of the PROFINET interface

Calling up the online and diagnostics view

To call up the online and diagnostics view, proceed as follows:

1. Open the device view.
2. Select the module to be diagnosed.
3. Select "Online> Online & diagnostics" from the main menu or select "Online & diagnostics" in the context menu.

Result

The online and diagnostics view of the module to be diagnosed is started.

Note

Establish the online connection

If no online connection is available when starting the online and diagnostics view, no online information can be displayed. The corresponding display fields remain empty.

Structure of the online and diagnostics view

Via the "Online access" group it is displayed if there is currently an online connection to the associated target or not. Furthermore, you can also establish or cancel an online connection.

The "Diagnostics" folder contains several diagnostic groups for the selected module.

The "Functions" folder contains several groups in which you can change the settings on the selected module, or output commands to the module.

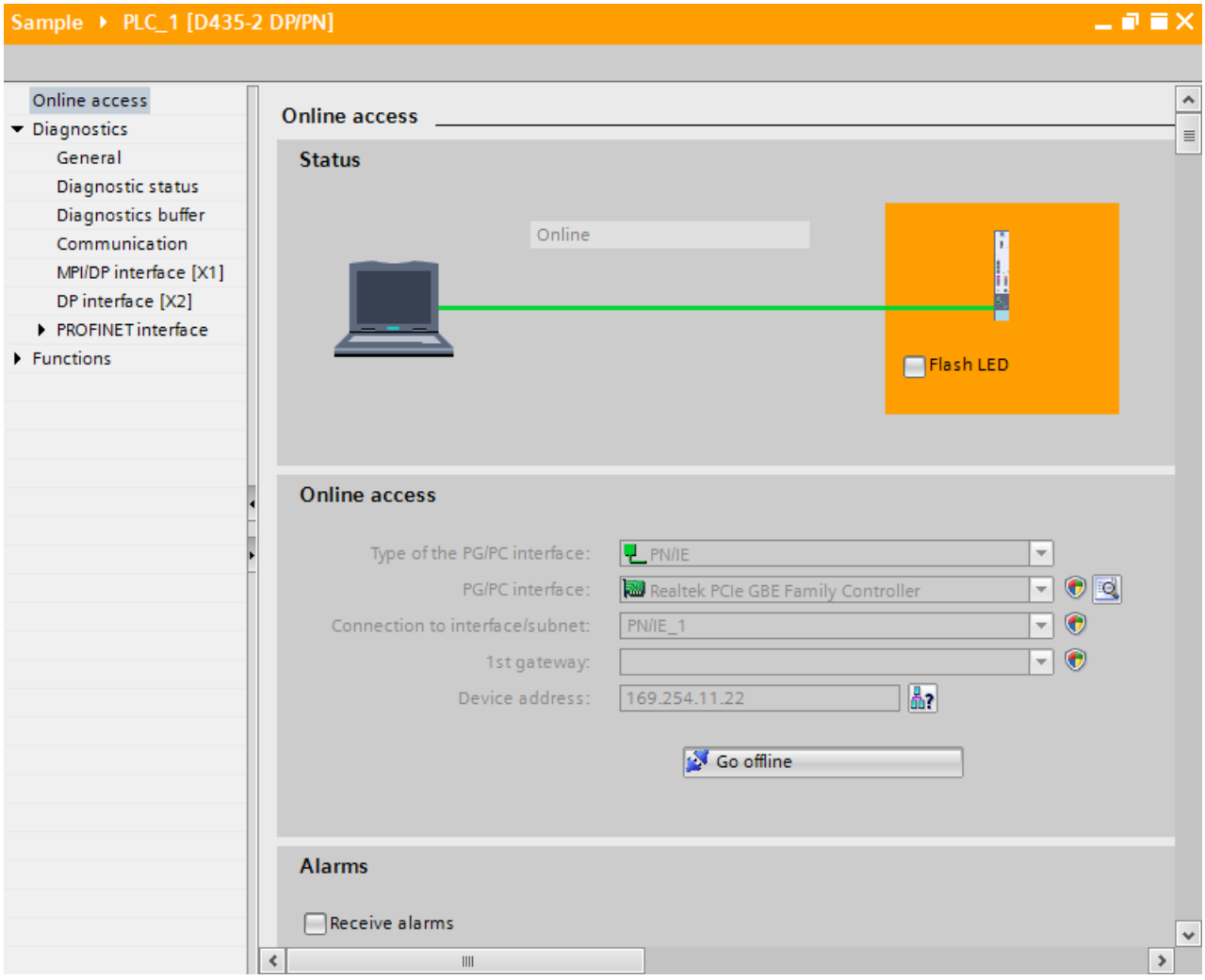


Figure 3-348 Online and diagnostics view

Diagnostics

General

The "General" group shows the general characteristics and system-relevant information of a module.

The following areas are displayed:

Module	The "Module" area contains information such as short designation, article number, firmware, hardware, rack and slot.
Module information	The "Module information" area contains information such as device name, module name, plant ID and location designation.
Manufacturer information	The "Manufacturer information" area contains information, such as manufacturer description, serial number, copyright and profile details.

Diagnostics status

The "Diagnostics status" group displays the status of the module from the perspective of the CPU.

Diagnostics buffer

The diagnostics buffer is used as a log file for the diagnostic events that have occurred on the CPU and its associated modules. The diagnostic results are entered in the order in which they occurred, with the most recent event shown at the top.

The following areas are displayed:

Events	Activate the "CPU time stamp considers local PG/PC time" option if you want the diagnostics buffer entries to be displayed with the time that is calculated using the following formula: Displayed time = module time + time zone offset of your PG/PC. It is assumed that the module time is identical with UTC. Use this setting if you want to see the local time of your PG/PC as the time-of-day in the diagnostics buffer entries of the module. If you activate or deactivate the checkbox, the time information of the diagnostics buffer entries is immediately adjusted. The events with consecutive number, date and time, and short description of the event are listed in the events table. If you click the "Freeze display" button, the current display of the diagnostics buffer entries will be frozen. If there is a quick succession of diagnostic events, this lets you examine the entries without a hurry. The CPU continues to enter events in the diagnostics buffer. Click "Cancel freezing" to update the diagnostics buffer entries again.
Event details	Detailed information such as the consecutive number of the event in the diagnostics buffer, event ID, description of the event with event-related additional details, time stamp, and information as to whether it is an incoming or outgoing event is displayed in the "Events" area. If you click the "Help for event" button, the selected event will be explained in greater detail and, if applicable, remedies are suggested. If you click the "Open in editor" button, the referenced block is opened in the offline view at the programming instruction causing the error if reference is made in the diagnostic event to a relative address of a block. This allows you to check the source code of the block at the specified location and, if necessary, change and subsequently load it back into the CPU. If the diagnostic event was triggered by a module, the "Open in editor" function opens the device view of the module concerned. Click "Save as..." if you want to save the contents of the diagnostics buffer in a text file.
Settings	In the "Settings" area, you can configure how the events should be displayed.

Communication

The following areas are displayed:

Connection resources, maximum number	Specifies the maximum number of available connection resources of the module. In each case, a certain number of available connection resources is reserved for PG, OP communication, S7 communication, and S7 basic communication. However, you can create additional PG or OP connections; the number of unassigned connection resources is then reduced accordingly.
Cycle load through communication	The CPU's cycle time can be extended due to communication processes. These communication processes include, for example: • Data transfer to another CPU • Loading of blocks is triggered via a programming device (PG) The duration of these communication processes can be controlled to a certain extent by means of the CPU parameter "Cycle load through communication". Via the parameter "Cycle load through communication", you indicate the percentage of total CPU processing power that is to be available to the communication processes.

Displaying interfaces and interface properties of a module

The interfaces and interface properties of a module can be found in the following groups:

- MPI/DP interface
- DP interface
- PROFINET interface

Additional references

Detailed information about diagnostic functions can be found in the information system of the TIA Portal at "Diagnose hardware".

Functions

Assigning an IP address

If the IP address of a device is located in a different subnet than the IP address of the network card, you must first assign the network card an additional IP address with the same subnet address as that of the device. Only then is communication between the device and the PG/PC possible.

Select "Connect online", for example, in order to automatically assign a temporary address in the same subnet to the PG/PC. A prompt is displayed automatically if the current IP address of the PG/PC is not already located in the correct subnet.

A temporarily assigned IP address is valid until the next restart of the PG/PC or until it is manually deleted.

Setting the time-of-day

In the "Set time-of-day" folder, you can determine and change the time of a CPU. This is only possible if there is an existing online connection.

The following areas are displayed:

PG/PC time:	Here the set time zone, the current date, and the current time-of-day of your PG/PC is displayed.
Module time	The "Module information" area contains information such as device name, module name, plant ID and location designation. This displays the current date and time values that were read from the module (e.g. CPU) and converted to local time. If the "Import from PG/PC" checkbox is activated, clicking on the button "Apply" triggers transfer of the date and the PG/PC time-to-day converted to UTC time) to the module. If the "Import from PG/PC" checkbox is deactivated, you can specify the date and time-of-day for the internal clock of the module. After clicking the "Apply" button, the date and time-of-day of day (converted to UTC time) will be transferred to the module.
Time system	In the "Time system" area, the following factors are considered for the time-of-day and synchronization: • **Resolution**: Specifies the accuracy of the calculation and output of the time-of-day. • **Real-time clock**: Indicator whether the CPU has a real-time clock: "available" when yes; "not available" when no. • **Correction factor**: Parameterized correction factor for the real-time clock of the module. The daily deviation of the clock is corrected using the correction factor. Synchronization can be achieved through synchronization mechanisms between the clocks of various sub-systems. The time synchronization settings are made during module parameterization. The integrated real-time clock - if present - can synchronize the clocks of other modules (master) or can be synchronized by the clocks of other nodes (slave), or it does not participate in the synchronization. Synchronization can be performed in the AS or MPI.

Assigning a PROFINET device name

Before an IO device can be addressed by an IO controller, it must have a device name. In the case of PROFINET, this method was chosen because names are easier to handle than complex IP addresses.

The assignment of a device name for a PROFINET IO device can be compared with setting the PROFIBUS address of a DP slave.

In its delivery state, an IO device does not have a device name. An IO device can only be addressed by an IO controller, for example, for the transfer of project engineering data (including the IP address) during startup or for user data exchange in cyclic operation, after it has been assigned a device name with the PG/PC.

Additional references

Detailed information about diagnostic functions can be found in the information system of the TIA Portal at "Diagnose hardware".

3.5.6 Configuring communication

3.5.6.1 Devices, networks and communication services in the TIA Portal

Configure and parametrize communication

Editing devices and networks

In the TIA Portal, you edit devices and networks in the hardware and network editor. For a detailed description, refer to the corresponding section in the information system of the TIA Portal at "Editing devices and networks". Note that modules can be fully parameterized only if they are assigned to a CPU. Therefore, PROFIBUS or PROFINET modules need to be networked first, so that they form a master system or IO system.

Only then, for example, can the addresses of the components be edited.

In this section, only the settings relevant to SIMOTION are listed below:

- PROFIBUS Integrated on SIMOTION D
- SIMOTION on PROFIBUS DP
- SIMOTION and PROFINET IO
- SIMOTION drives on PROFIBUS DP
- SIMOTION drives on PROFINET IO

Note

SIMOTION communications

The theoretical principles of the communication for SIMOTION are contained in the *Motion Control Communication* Manual.

Representation in the user interface

Devices and networks are shown in the device view and the network view.

Device view

In the device view, the device is illustrated with all interfaces. Additional modules can also be added, e.g. communication boards.

In the device overview, the parameters of the individual interfaces are shown in tabular form.

If you select an interface or the whole device, you will see the corresponding parameters displayed in tabs in the Inspector window.

Network view

In the network view, you perform the following tasks:

- Configure and parametrize devices
- Network devices with each other

For SIMOTION devices, this means they are interconnected with each other, with drives, SIMATIC modules or PN IO devices (e.g. ET200SP, third-party devices) in the network view.

In the network overview, the parameters of the modules are shown in tabular form.

If you select a device or the bus, you will see the corresponding parameters displayed in tabs in the Inspector window.

Configuring telegrams

Telegrams are generally configured in SIMOTION SCOUT TIA. Telegrams are configured in the TIA Portal for drives and other hardware that have been inserted, for example, via GSD. In SIMOTION SCOUT TIA, you specify the telegrams or allow specification of the telegrams by the symbolic assignment. The telegrams configured for the respective drive objects are displayed in the device overview of the device in the TIA Portal.

Device overview

Module	Rack	Slot	I address	Q address	Type	Order no.
Drive unit_1	Rack_0	0	16369*		S120 CU 320-2 PN	6SL3 040-1MA01
PROFINET interface_1	Rack_0	0 X1	16368*		PROFINET-Interface	
Drive_1	Rack_0	1			Drive object	
Module access point_1	Rack_0	1 1	16365*		Module access point	
SIEMENS telegram 105	Rack_0	1 2	288...307	288...307	SIEMENS telegram 105, PZD-1...	
	Rack_0	1 3				
	Rack_0	1 4				
Supply	Rack_0	2			Drive object	
Module access point_1	Rack_0	2 1	16364*		Module access point	
SIEMENS telegram 370	Rack_0	2 2	320...321	320...321	SIEMENS telegram 370, PZD-1/1	
	Rack_0	2 3				
	Rack_0	2 4				
Control_Unit	Rack_0	3			Drive object	
Module access point_1	Rack_0	3 1	16363*		Module access point	
SIEMENS telegram 390	Rack_0	3 2	352...355	352...355	SIEMENS telegram 390, PZD-2/2	
	Rack_0	3 3				

Figure 3-349 View telegrams

Note

Handling telegrams

Deleting telegrams in the TIA Portal can result in synchronization errors between SCOUT TIA and the TIA Portal.

- Avoid deleting telegrams in the TIA Portal.

Move the slots in the TIA Portal device overview

- The moving of individual slots in the TIA Portal device overview can cause problems when saving and compiling or during the telegram alignment. The "Value outside permissible range" message appears. For this reason, do not move any slots in the device overview.

Telegram configuration and symbolic assignment

Introduction

The communication and also the communication roles of PROFINET IO controllers / IO devices or master-slave relationships for PROFIBUS DP are based on the use of telegrams. The telegrams and necessary interconnections can be automatically created during configuration in SIMOTION SCOUT TIA via the symbolic assignment. This in turn is a requirement for the automatic setting of the communication relationships in the TIA Portal.

Symbolic assignment in SIMOTION SCOUT TIA

SIMOTION SCOUT TIA supports the symbolic assignment on SINAMICS drive objects (DOs, Drive Objects) during the configuration of technology objects (TOs) and I/Os. This simplifies the configuration of the technological relationships, including communication between IO controllers and IO devices (e.g. drives) for PROFINET IO or master and slave for PROFIBUS DP.

After the configuration of the hardware in the TIA Portal (devices created, ports interconnected), you should configure the drives and technology objects (axes) in SIMOTION SCOUT TIA. Because of the selected functions for the configuration of the drives and technology objects, the required telegrams are determined. They are used as basis for the communications configuring in the TIA Portal. If a telegram configuration exists, the TIA Portal makes the required settings automatically, e.g. the Sync-Master and Sync-Slave synchronization roles are preassigned automatically and a synchronous mode set. Note this for further actions.

Note

Before communications are configured in the TIA Portal, the drive and axis should be configured with active symbolic assignment in SIMOTION SCOUT TIA.

For further information about the symbolic assignment, see the *SIMOTION Runtime Basic Functions* Function Manual.

Time-of-day synchronization via NTP

Introduction

To ensure that all components throughout the system have the same time of day, they must be synchronized to a time of day and one component must be the time generator for all the others. One possibility is to synchronize the time of day via an NTP server.

For NTP, the device sends time of day queries in regular time intervals to the NTP server in the subnet. A router must be used if the NTP server is located outside the subnet. The most accurate time of day is determined based on the responses and used to synchronize the station time of day. NTP has the advantage that the synchronization is also performed over subnet limits.

NTP time-of-day synchronization properties

- The NTP server is configured at the PN/IE interface of the controller. The PN interface via which communication with the time-of-day server actually takes place is determined on the basis of the IP address configured in the TCP/IP stack.
  - X7 with C240
  - X130 with D4x5
  - X127 with D410
  - X1 with P320-4
- Up to four servers can be configured as time generator.
- NTP does not support any automatic adjustment between summer and winter time.
- Time zones cannot be set. UTC (Universal Time Coordinated) is always transferred. It is identical with GMT (Greenwich Mean Time).

Configuring NTP in SIMOTION SCOUT

Introduction

The NTP time-of-day synchronization of a SIMOTION CPU can be configured in the TIA Portal.

Procedure

How to configure the NTP time synchronization:

1. In the device view, click the PN/IE interface (e.g. for D4x5 X130).
2. In the Inspector window, select "Properties > General" and click "Time-of-day synchronization".
3. Activate the "Activate time-of-day synchronization via NTP server" checkbox.

4. Enter the IP addresses for as many as four NTP servers. A router must be used when the IP address lies outside the subnet of the SIMOTION controller.
5. Select the refresh interval.

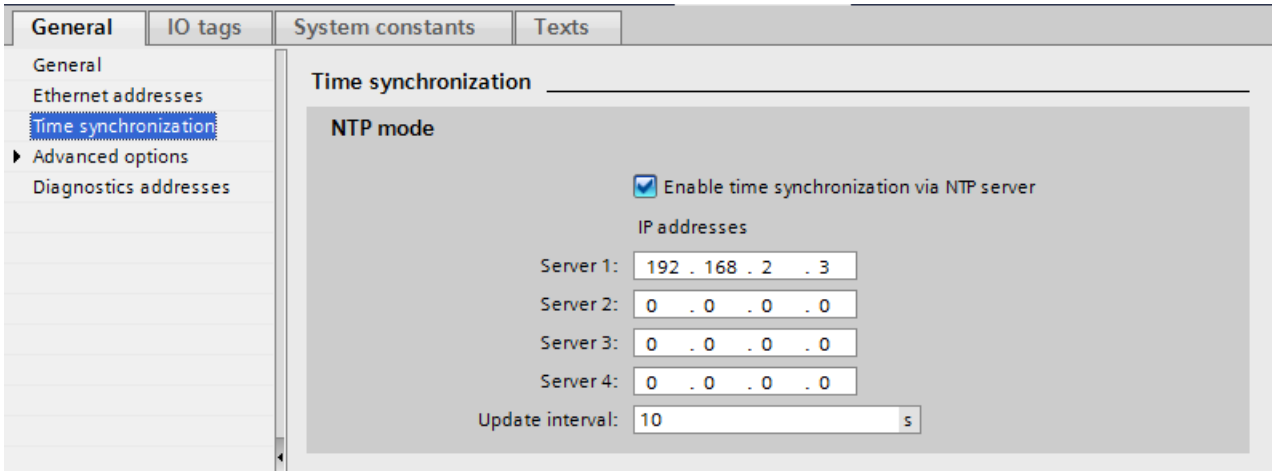


Figure 3-350 Time-of-day synchronization via the NTP protocol in the TIA Portal

TCP expert settings

You set the TCP expert settings for SIMOTION controllers in the TIA Portal at "Properties > General". With these settings, you configure the behavior of the host (SIMOTION) for TCP/IP connections. TCP is a connection-oriented protocol, the sender receives a receive confirmation via the reception of the sent data.

Set the times for the timeout so that the sender does not have to wait continuously for a receive confirmation when there is a connection abort. You can set these and the number of new send attempts here. Only change these values when the communication is frequently faulted due to timeouts. This can be the case with slow communication connections with high latency:

The following settings are required:

- Connection timeout [ms]
  Within this period, the sender waits for a receive conformation from the recipient.
- Max. retransmission timeout [ms]
  After this time has expired, a retransmission is started until the connection timeout or the maximum number of retransmissions is reached.
- Max. number of retransmissions
  Maximum number of attempts for a retransmission. If the maximum number is exceeded, there is no retransmission attempt.

3.5.6.2 PROFINET IO

Device settings on the PROFINET IO

Introduction

This section briefly explains the basic concepts and properties of PROFINET IO.

PROFINET IO

PROFINET, as an Ethernet-based automation standard specified by PROFIBUS International (PI - PROFIBUS & PROFINET International), defines a manufacturer-independent communication, automation and engineering model.

PROFINET IO systems

A PROFINET IO system consists of a PROFINET IO controller and its assigned PROFINET IO devices. The PROFINET IO controller is a controller (e.g. SIMOTION D4x5-2 DP/PN) that controls the automation task. A PROFINET IO device is a device (e.g. SINAMICS S120) controlled and monitored by an IO controller.

RT classes and isochronous mode

PROFINET offers two transmission protocols that are adapted to the requirements of automation. These are PROFINET IO with RT and PROFINET IO with IRT. For typical motion control applications, an isochronous application with IRT is required. As well as synchronizing the transmission network, IRT enables the application (e.g. SIMOTION position controller and interpolator) to be synchronized in the devices (isochronous application).

Addressing of PROFINET IO devices

In addition to the MAC and the IP address, PROFINET also uses a device name (NameOfStation) to identify PROFINET devices.

Creating a PROFINET subsystem

To create a new PROFINET subsystem, proceed as follows:

1. Right-click the PN interface at which you want to create a subsystem.
2. Select "Create subnet" if you want to create a new subnet.
3. Select "Assign subnet" if you want to assign an existing subnet to the PN interface.

Editing an IP address

To set an IP address, proceed as follows:

1. In the network view / device view, select the interface whose IP address you want to edit.
2. In the Inspector window, select the "Properties" tab and click the "General" tab.
3. Select "Ethernet addresses".
   You will now see the IP address and subnet mask below the "IP protocol".

   Note

   Make sure that all IP addresses and subnet masks are in the same area.

4. On this tab, you can select the subnet assigned to the interface with "Interface networked with".

Assigning device names

Before an IO device can be addressed by an IO controller, it must have a device name. In the case of PROFINET, this method was chosen because names are easier to handle than complex IP addresses.

The PROFINET device name must conform to the DNS naming conventions. Further information is available in the online help of the TIA Portal with the "Address and naming convention for PROFINET devices" keyword.

To change the device name, proceed as follows:

1. In the network view / device view, select the interface whose device name you want to edit.
2. In the Inspector window, select the "Properties" tab and click the "General" tab.
3. Select "Ethernet addresses".
   You now see at "PROFINET" the setting options of the device name.
4. Change the device name or select "Automatically generate PROFINET device name" for automatic generation.

Note

Converted device names

The TIA Portal supports PROFINET device names that do not conform with the DNS name conventions. A "converted name" is generated from the non-conform PROFINET device name. This is the device name that is actually loaded to the device. The PROFINET device name is converted only when it does not conform to the IEC 61158-6-10 rules. The PROFINET device name and the converted name are displayed for the properties of the PN interface. You cannot change the converted name; it is generated automatically.

Additional references

You will find detailed information and SIMOTION-specific settings and parameters described in detail in the *SIMOTION Communication* Function Manual.

Brief introduction communication configuration SIMOTION controller with SIMOTION drive via PROFINET IO

SIMOTION controller with PROFINET IO drive

Introduction

This section explains how to get started quickly with communication configuration of a PROFINET IO system using the examples of a SIMOTION D455-2 DP/PN with a SINAMICS S120 CU320-2 PN drive.

Configuration example	Description
SIMOTION D455-2 DP/PN IO controller	Sync master in the sync domain (IRT) Servo isochronous mode of the subnet
IO device SINAMICS S120 CU320-2 PN	Sync slave in the sync domain (IRT) IO device isochronous mode with the servo cycle

Note

For the following steps, symbolic assignment must have been activated in SIMOTION SCOUT TIA and the drive and axis must have been configured before communication configuration.

Additional references

Detailed information about the basic theoretical principles is contained in the *SIMOTION Communication* System Manual or in the information system for the TIA Portal.

General overview of the configuration sequence

To configure an isochronous PROFINET IO system, proceed as follows:

1. Insert a SIMOTION D4x5-2 DP/PN (PROFINET IO controller) and a SINAMICS S120 CU320-2 PN (PROFINET IO device) (Inserting devices (Page 685)).
2. Configure a PROFINET IO system (Inserting a subnet (Page 686)).
3. Interconnect the topology. This is necessary if you are using PROFINET IO with IRT (Interconnecting topology (Page 688)).
4. Switch to SIMOTION SCOUT TIA and configure the drive and the telegrams (Telegram configuration (Page 688)).
5. Switch to the TIA Portal. All necessary settings for communication configuration are performed automatically (Communication configuration (Page 690)).

Result

The PROFINET IO system is fully configured. Further operating steps are only required if you want to make changes to the default settings.

A detailed description of the individual steps is given in the next sections.

Inserting SIMOTION D455-2 DP/PN and SINAMICS S120 CU320-2 PN

Requirement

You have created a new project in the TIA Portal.

Inserting a device and drive

1. Drag a SIMOTION D455-2 DP/PN (S120) from the folder "Controller > SIMOTION" in the hardware catalog and drop it into the network view. The IP address and the PROFINET device name are assigned automatically. You can leave the default settings unchanged or adjust them accordingly.
2. Drag a SINAMICS S120 CU320-2 PN from the folder "Controller > SIMOTION > SIMOTION drives > SINAMICS S120" in the hardware catalog and drop it into the network view. The IP address and the PROFINET device name are assigned automatically. You can leave the default settings unchanged or adjust them accordingly.

After inserting the hardware, the network view is represented as follows. The drive device is shown as "Not assigned".

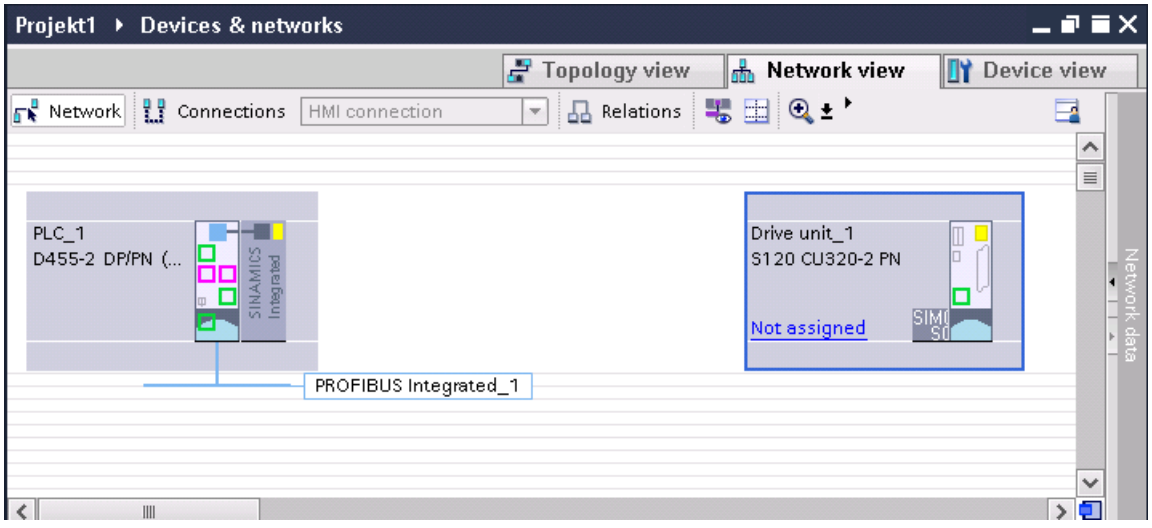


Figure 3-351 SIMOTION D455-2 DP/PN and SINAMICS S120 CU320-2 PN in the network view

In the next step, you will create a PROFINET IO system.

Creating a PROFINET IO system

Introduction

After inserting the hardware, it must first be assigned to a subnet. Unassigned devices are displayed as "Not assigned". Nodes of a subnet form a PROFINET IO system. A sync domain is also needed for synchronizing PROFINET IO devices. The sync domain ensures that all nodes are synchronized. When a subnet is created, a new sync domain is also created.

Creating a sync domain

To assign the devices to a sync domain, proceed as follows:

1. In the network view, click the "Not assigned" link on the SINAMICS S120 CU320-2 PN. All IO controllers in the project are displayed.

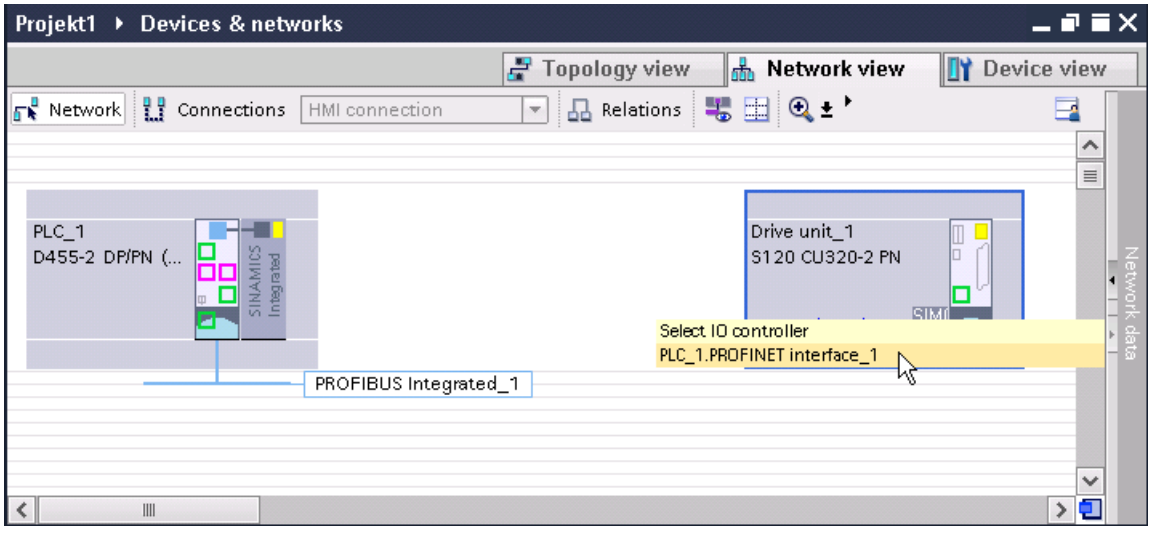


Figure 3-352 Assigning an IO device to an IO controller

2. Click the displayed PROFINET device name of the SIMOTION D455-2 DP/PN to connect it to the SINAMICS S120 CU320-2 PN IO device. A PROFINET IO system and a sync domain with the two nodes is automatically created.
   If you assign another IO device to the IO system of the IO controller, the IO device will automatically be assigned to the sync domain of the IO controller.

After you have assigned the drive unit, a sync domain ① and a PROFINET IO system ② will be created automatically. The IO controller is created as a sync master and the drive unit is created as an IO device without synchronization ③. The RT class and the synchronization role of the drive unit is automatically generated when the drive and the axis are configured in SIMOTION SCOUT TIA.

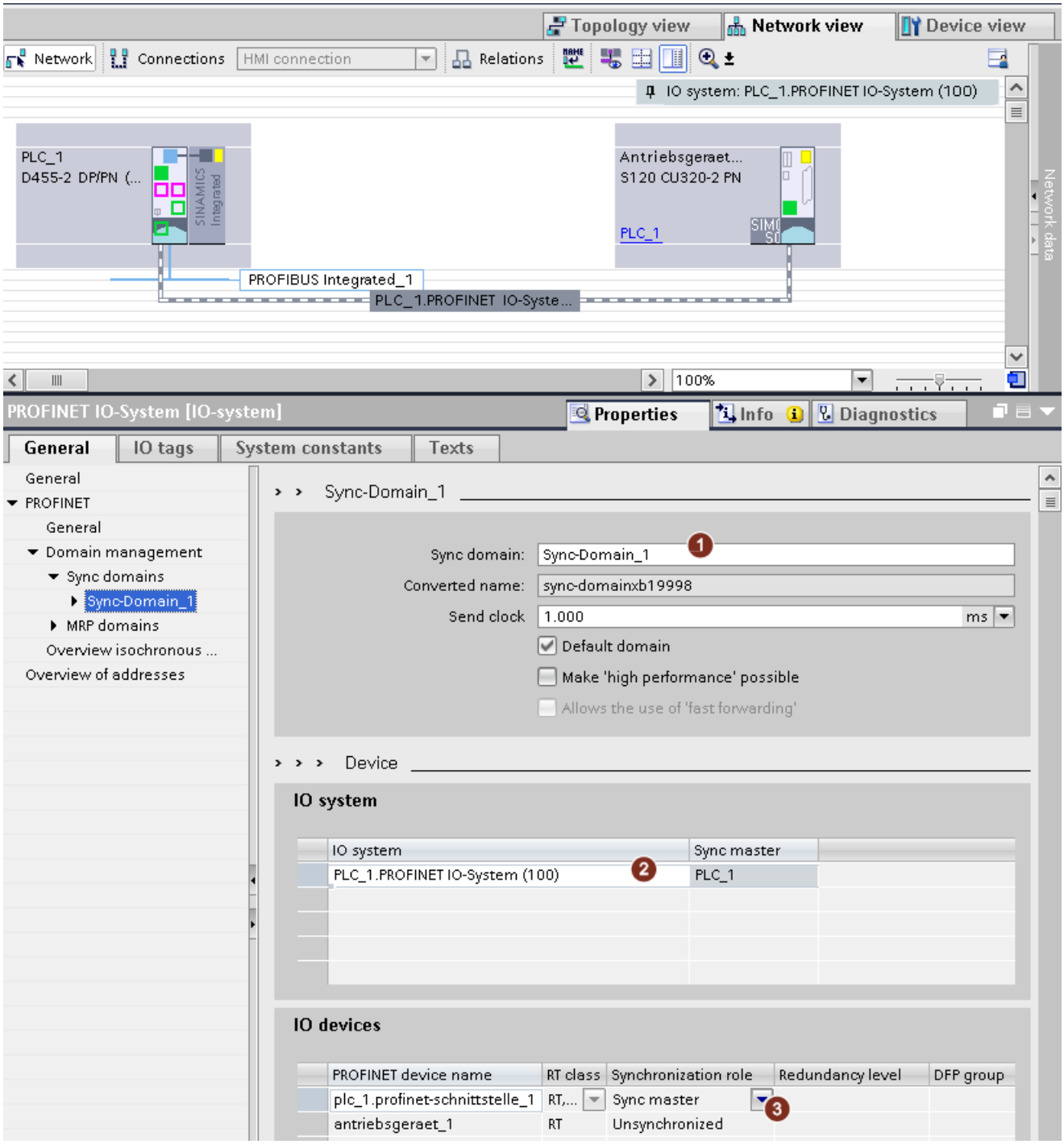


Figure 3-353 Sync domain and PROFINET IO system

The topology is interconnected in the next step.

Interconnecting the topology

Introduction

A requirement for isochronous IRT is the topology configuration. In the example, a connection from the port of the IO controller (SIMOTION D455-2 DP/PN) to the port of the IO device (SINAMICS S120 CU320-2 PN) must be configured. The configuration is performed in the topology view.

Configuring the topology

1. Switch to the "Topology view" tab.
2. Click the port of SIMOTION D455-2 DP/PN. The topmost port (X150 P1) is used in the example.
3. Hold down the left mouse button and draw a connection to the port of the SINAMICS S120 CU320-2 PN. The topmost port (X150 P1) is used in the example.

The connection between the ports is created.

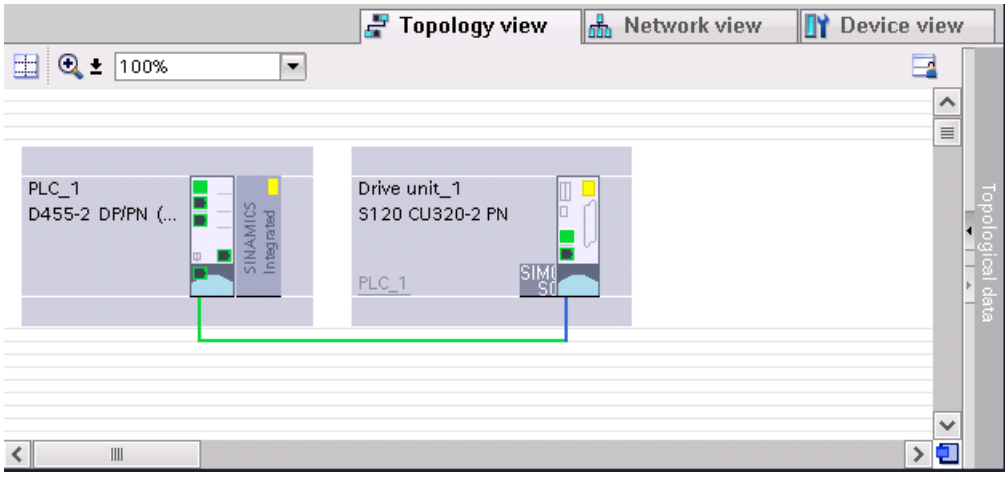


Figure 3-354 Configuration of the topology in the topology view

After this step, configuration in the TIA Portal is complete. The next steps are performed in SIMOTION SCOUT TIA.

Configuration of technology and telegrams in SIMOTION SCOUT TIA

Introduction

After you have inserted the hardware in the TIA Portal and configured the topology, switch to the SIMOTION SCOUT TIA to configure the drive and the axis (technology). This step is necessary for automatic communication configuration. With configuration of an axis on the drive, the TIA Portal automatically configures communication, e.g. the drive as a sync slave that is isochronous with the servo cycle.

Procedure

Note

You will find the precise procedure for configuration of the drive and the axis in the "SIMOTION SCOUT TIA" Configuration Manual in the Motion Control parameterization/programming section.

1. Open SIMOTION SCOUT TIA.
2. Configure the drive. In the example, a SINAMICS S120 CU320-2 PN.
3. Create an axis and assign it to the drive in the axis wizard.
   If only one axis and one drive are present in the project, the correct assignment is automatically displayed.

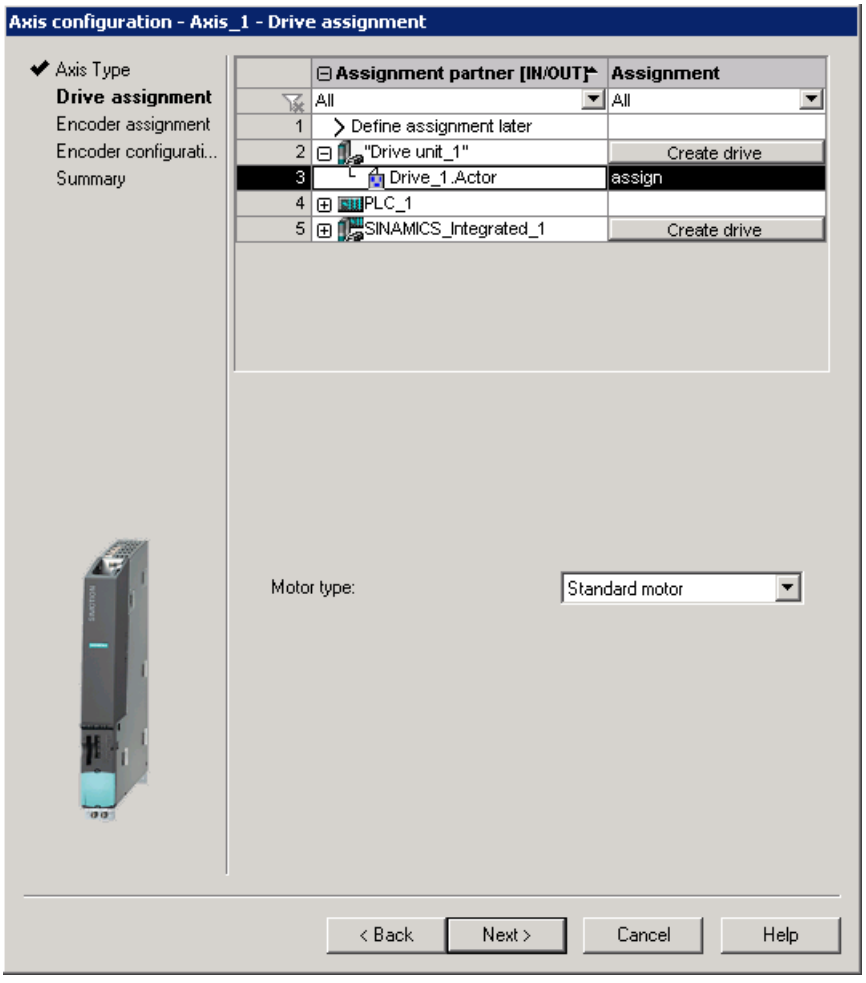


Figure 3-355 Axes are assigned to the drive during configuration

4. "Save and compile" in SIMOTION SCOUT TIA. During active symbolic assignment, the technological relationships and the telegrams are configured in the SIMOTION SCOUT TIA automatically.
5. Open the telegram configuration below the drive unit to view the configured telegrams.

Result

The telegrams and addresses have been synchronized with the TIA Portal. With the "Telegram configuration" entry below the drive unit, you can switch to the telegram overview in which blue checkmarks are now shown.

Configuration in SIMOTION SCOUT TIA has now been completed. The communication configuration has therefore been created automatically in the TIA Portal.

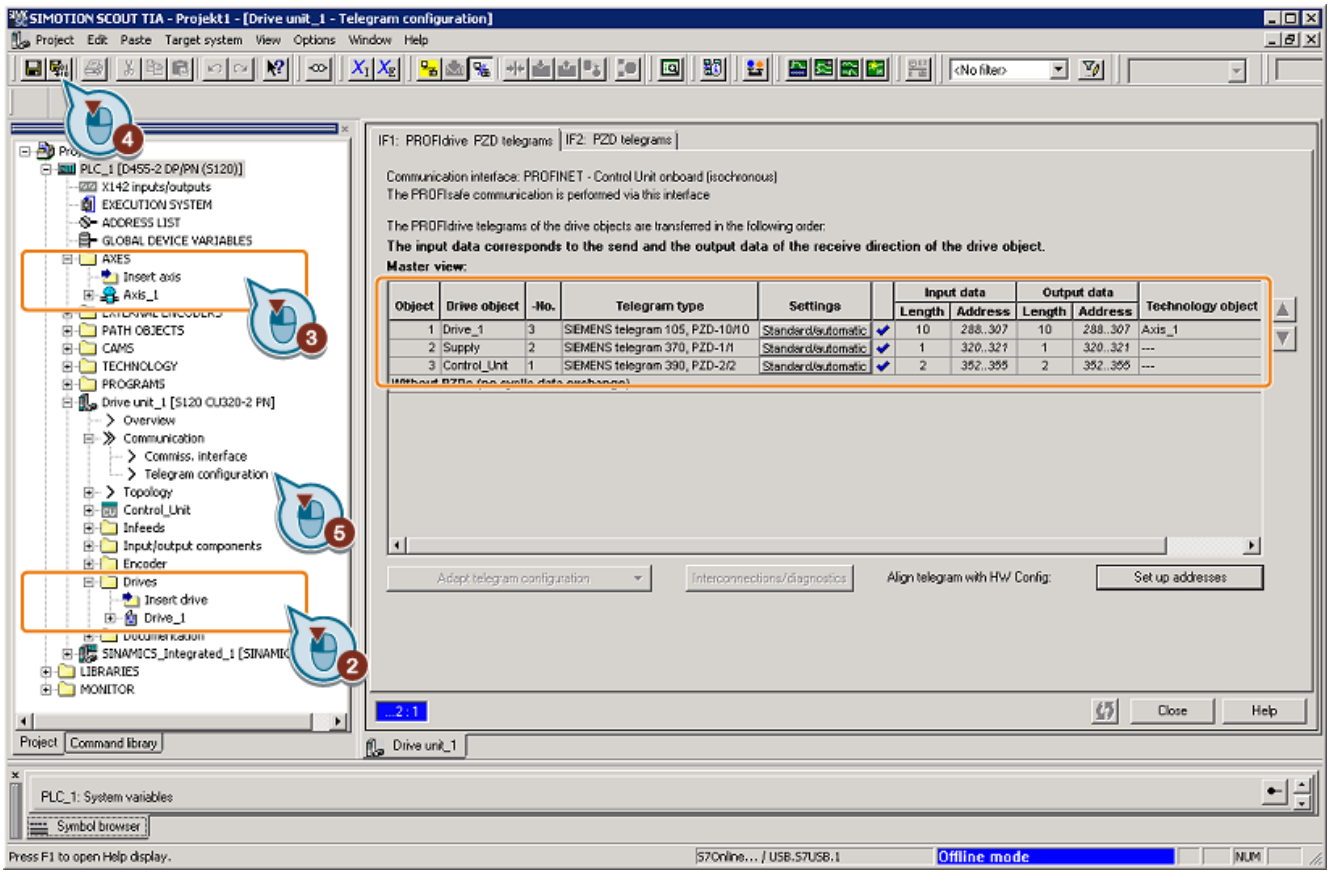


Figure 3-356 Drive and axis configuration in the SIMOTION SCOUT TIA

Automatic communication configuration in the TIA Portal

Introduction

After synchronization of the telegrams of SIMOTION SCOUT TIA to the TIA Portal, the communication configuration will automatically be adapted in the TIA Portal. All necessary communication settings required because of the technology configuration in SIMOTION SCOUT TIA have been adapted automatically. You can check the automated settings below. The communication configuration has now been completed.

Checking for automatic communication configuration

In the table, you can see the necessary communication configuration settings. You can check them.

Configuration example	Description
SIMOTION D455-2 DP/PN IO controller	Sync master in the sync domain (IRT) ① Servo isochronous mode of the subnet ③
IO device SINAMICS S120 CU320-2 PN	Sync slave in the sync domain (IRT) ② IO device isochronous mode with the servo cycle ④

Checking sync domain, sync master, sync slave ①②

1. Select the PROFINET IO system in the network view.
2. Select the sync domain in the secondary navigation "PROFINET > Domain management > Sync domains" on the "General" tab. You can check the settings there.

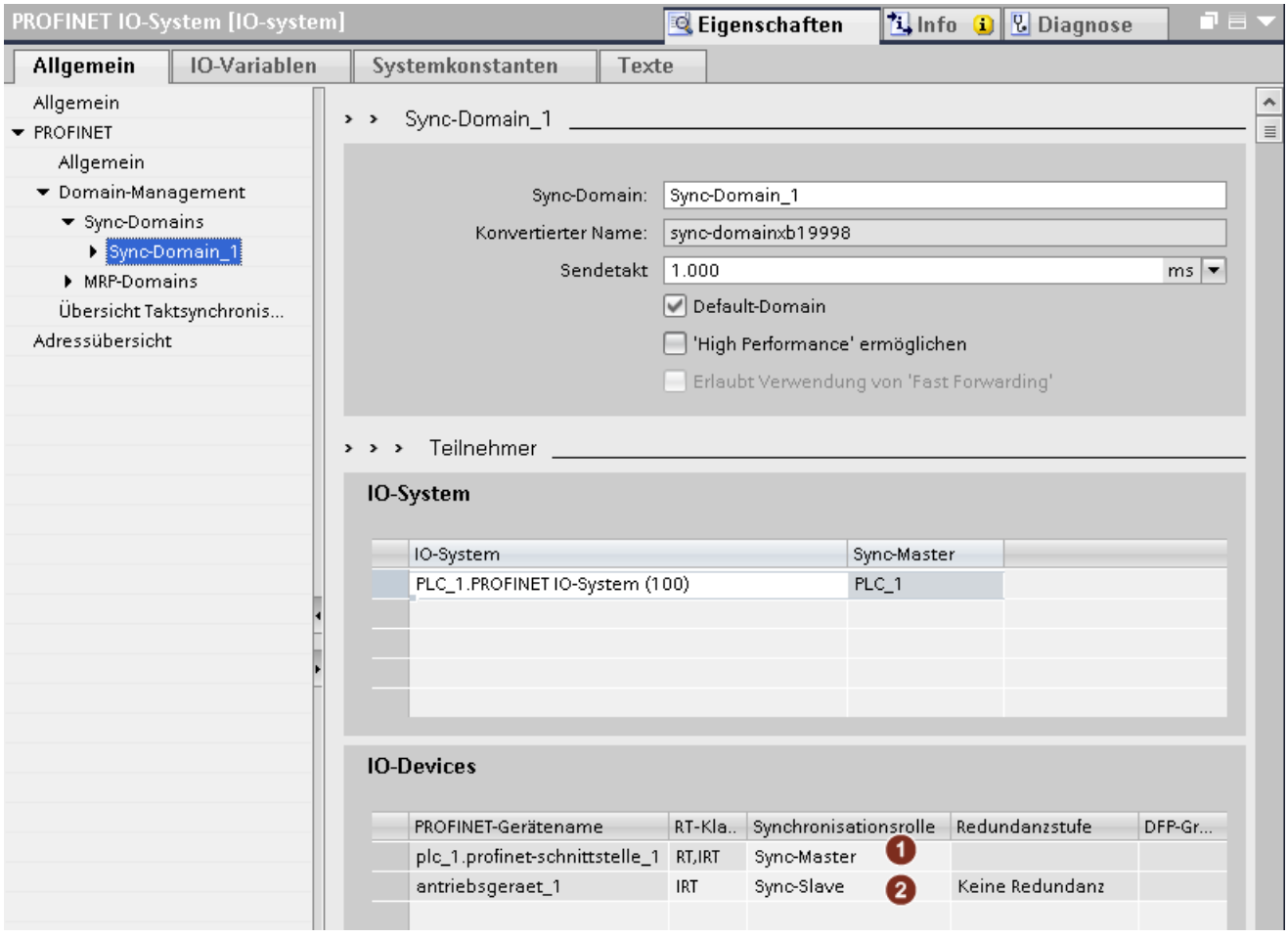


Figure 3-357 Checking sync domain, sync master, sync slave

Servo isochronous mode of the subnet ③

1. Select the PROFINET IO system in the network view and click the "Overview isochronous mode" entry on the "General" tab of the Inspector window. You can check the settings there.

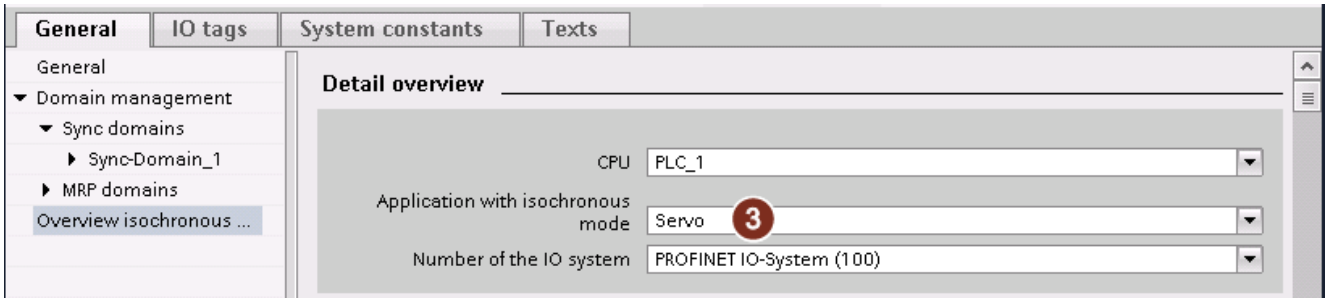


Figure 3-358 Isochronous mode of the subnet

IO device (sync slave) isochronous with the servo cycle ④

1. In the network view, select the PN interface of the IO device and click the "isochronous mode" entry on the "General" tab of the Inspector window. You can check the settings there. The isochronous telegram of the drive is shown in the detailed view.

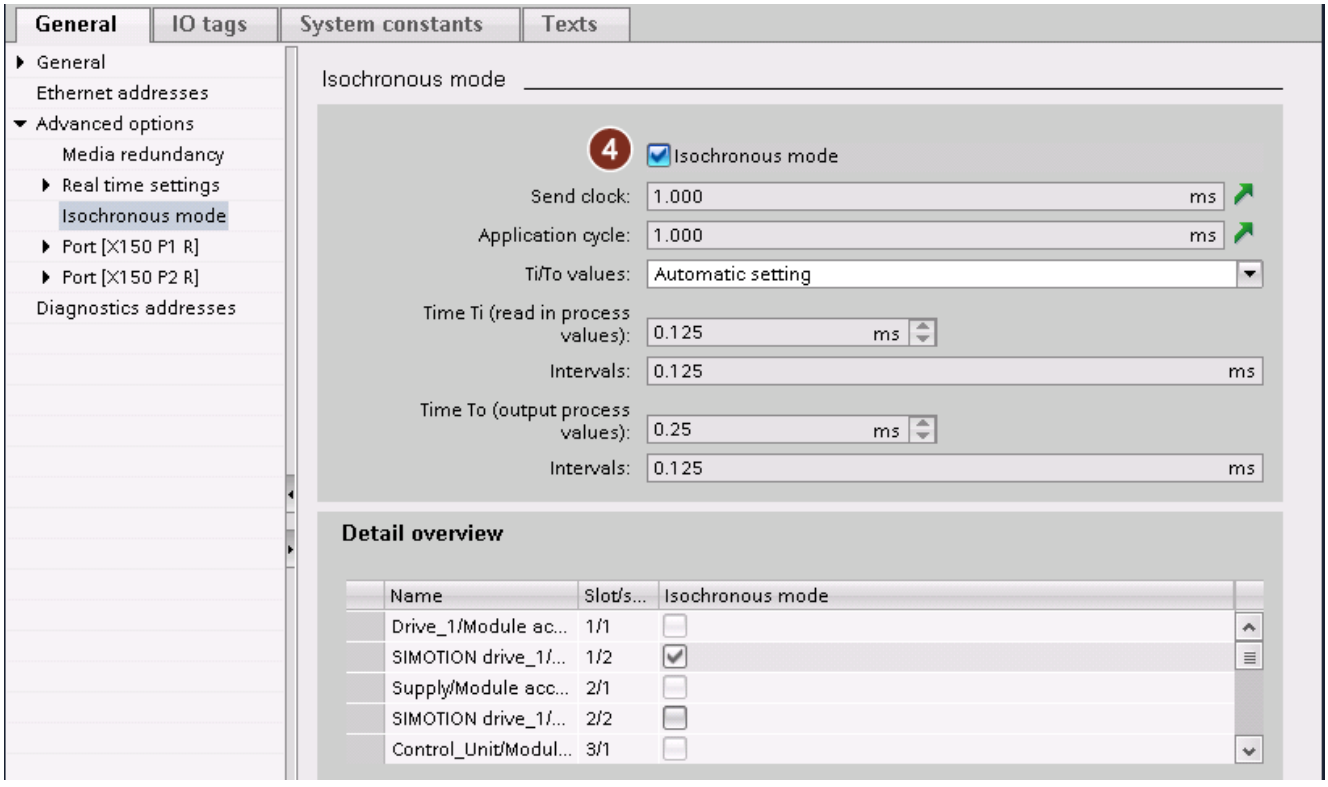


Figure 3-359 Isochronous mode of the PROFINET IO device in the servo cycle

If you would like to change the preset values, e.g. the send cycle, you will find more detailed explanations in the following sections.

Sync domain, topology, and isochronous mode

IO device (drive) on PROFINET IO

Introduction

A SIMOTION device should be connected with an IO device via PROFINET, in our example, a drive device. You have already inserted a SIMOTION device with integrated PROFINET interface into your project.

Section "Inserting a SIMOTION device (Page 643)" describes inserting a SIMOTION device in detail.

Note

Drive devices that are not networked or assigned to an IO controller are not displayed in SIMOTION SCOUT TIA.

Inserting and editing drives

To operate a drive as an IO device, proceed as follows:

1. Select a SIMOTION drive in the hardware catalog at "Controller> SIMOTION> SIMOTION drives" and drag it into the network view.
2. Click the link on the unassigned device in the network view (in the example, a SINAMICS S120).
   All possible IO controllers in the project are displayed.

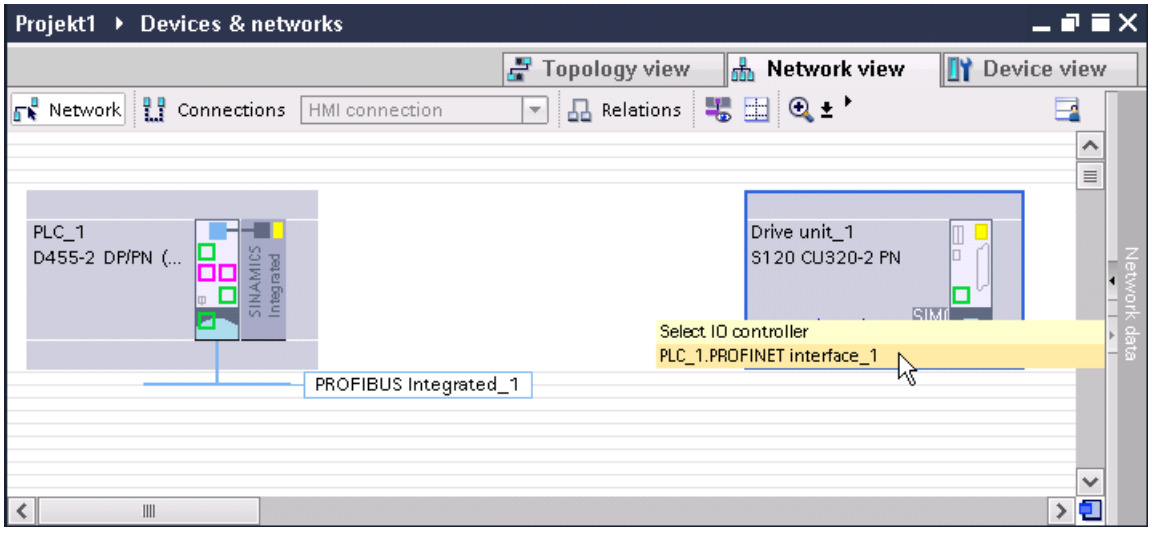


Figure 3-360 Assigning an IO device to an IO controller

3. Assign the drive to a SIMOTION controller. A PROFINET IO system and a sync domain with the two nodes are created automatically. If you assign another IO device to the IO system of the IO controller, the IO device is assigned automatically to the sync domain of the IO controller.

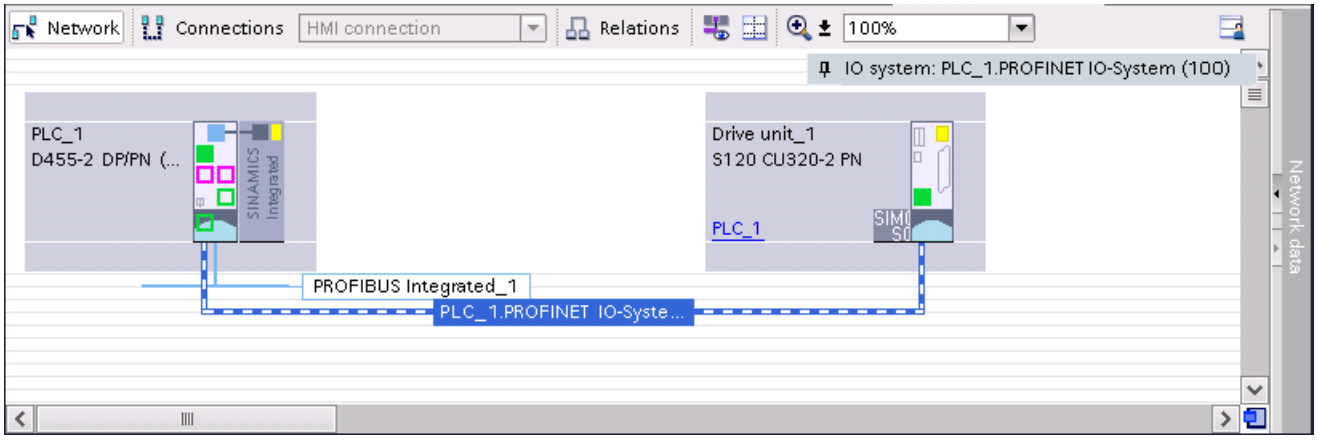


Figure 3-361 SINAMICS drive device on SIMOTION D4x5-2 via PROFINET

Note

You need a telegram for the communication between a SIMOTION controller and a drive. Telegrams are configured in SIMOTION SCOUT TIA. This requires that you configure the drive and the axis in SIMOTION SCOUT TIA. If "Symbolic assignment" is activated, the required telegrams are configured automatically for saving and compiling.

After inserting the IO device, you can configure the sync domain and the isochronous mode of the IO device. Isochronous mode is required for typical motion control applications.

See also

Real-time settings on IO devices and I devices (Page 714)

Configure sync domains and send clock (Page 695)

Configure sync domains and send clock

Configuring sync domains

A sync domain is needed for synchronizing PROFINET IO devices. The sync domain ensures that all nodes that belong to it are synchronized. After being inserted, the nodes of the sync domain are not synchronized.

After inserting a SINAMICS CPU and a SINAMICS S120 drive device, they must first be assigned to a subnet. Unassigned devices are displayed as "Not assigned".

To configure the sync domain, proceed as follows:

1. Select the PROFINET IO system in the network view.
2. In the Inspector window, click the "General" tab and select "Domain management > Sync domain > Sync domain_1". You can set the parameters in the tab.
3. You can set a new name at "Sync domain". It is converted automatically to a conformant name that is displayed at "Converted name".
4. Select the sync master (in the example, SIMOTION D445-2 DP/PN) and select the RT class for the IO device (in the example, SINAMICS S120). When you select IRT, the synchronization role of the IO device is set automatically to sync slave.
5. Select the "send cycle" for the sync domain.

   Note

   You can set the send cycle only when you have configured the synchronization role and the RT class at "IO devices".

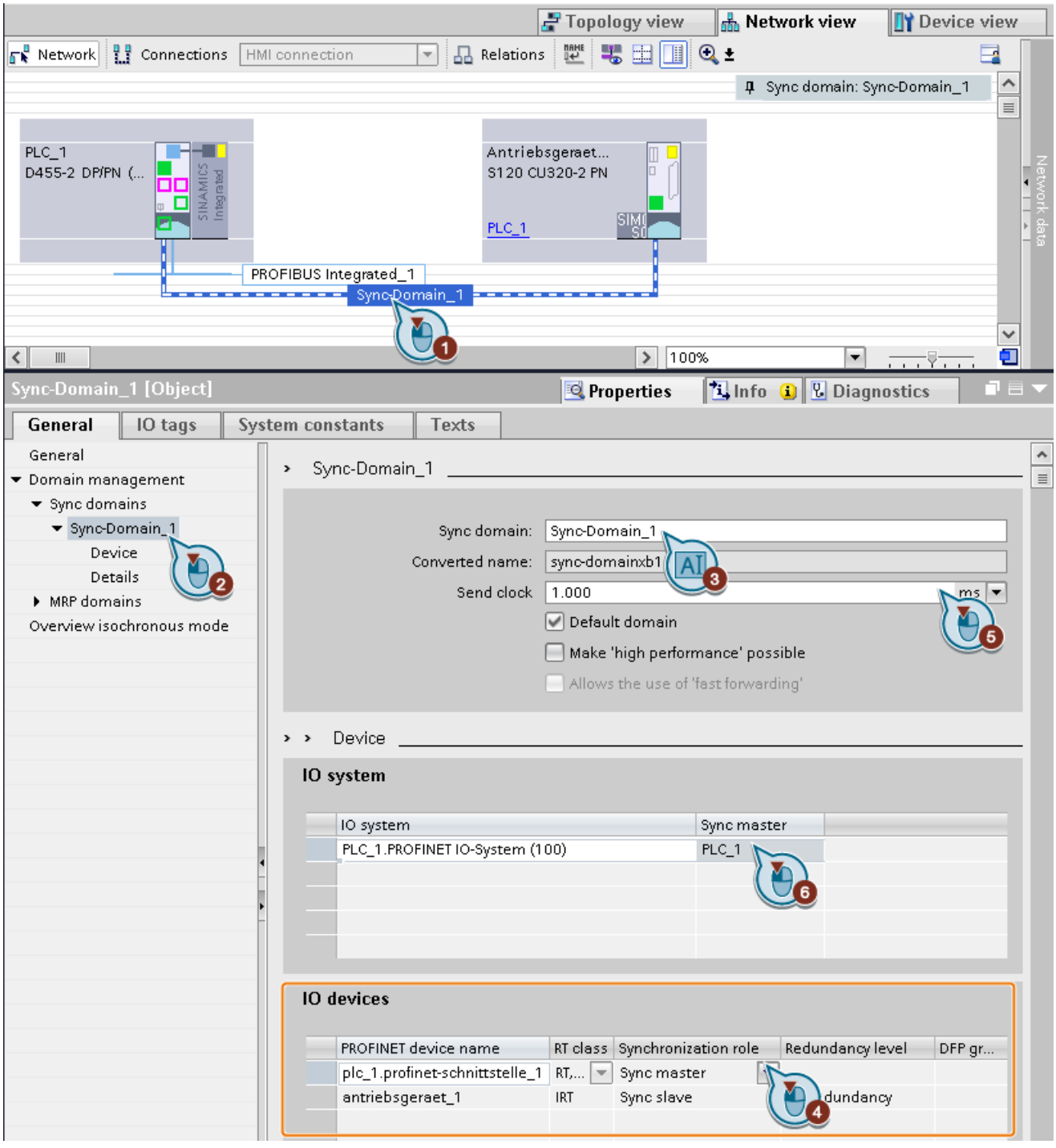


Figure 3-362 Configuring the sync domain

Explanation of the parameters

Parameter	Description
③ Sync domain	Name of the sync domain. This name is converted automatically if it does not conform with the DNS conventions.
④ IO devices	Here, you can see a list of the IO devices of the sync domain. You can select the following parameters for the nodes: • RT class: Select either RT (Real Time) or IRT (Isochronous Real Time). • Synchronization role: Choose between sync master, redundant sync master or sync slave. The synchronization role is displayed as "unsynchronized" by default, e.g. for devices in RT networks. Information about the RT classes and synchronization roles is provided in the SIMOTION SCOUT Communication Function Manual.
⑤ Send cycle	Select the send cycle for the sync domain here.
⑥ Node	The nodes of the PROFINET IO system are listed here.

See also

IO device (drive) on PROFINET IO (Page 693)

Set up the sync master and sync slave.

Requirement

You have performed the following configured steps:

- IO controller and IO device configured with RT
- Port interconnections are configured

Note

The sync master and sync slave are automatically configured in the TIA Portal if you have configured the drive and axis as well as saved and compiled the project in SIMOTION SCOUT TIA. You will find further information and a description of the configuring in SIMOTION SCOUT TIA in Section Isochronous mode (Page 702).

Configuring the sync master

To configure a sync master, proceed as follows:

1. Select the PROFINET interface of the sync master in the network view.
   The interface settings are displayed in the Inspector window.
2. Click "Extended options > Real-time settings > Synchronization" on the "General" tab.
3. Select the "Sync master" entry at "Synchronization role".

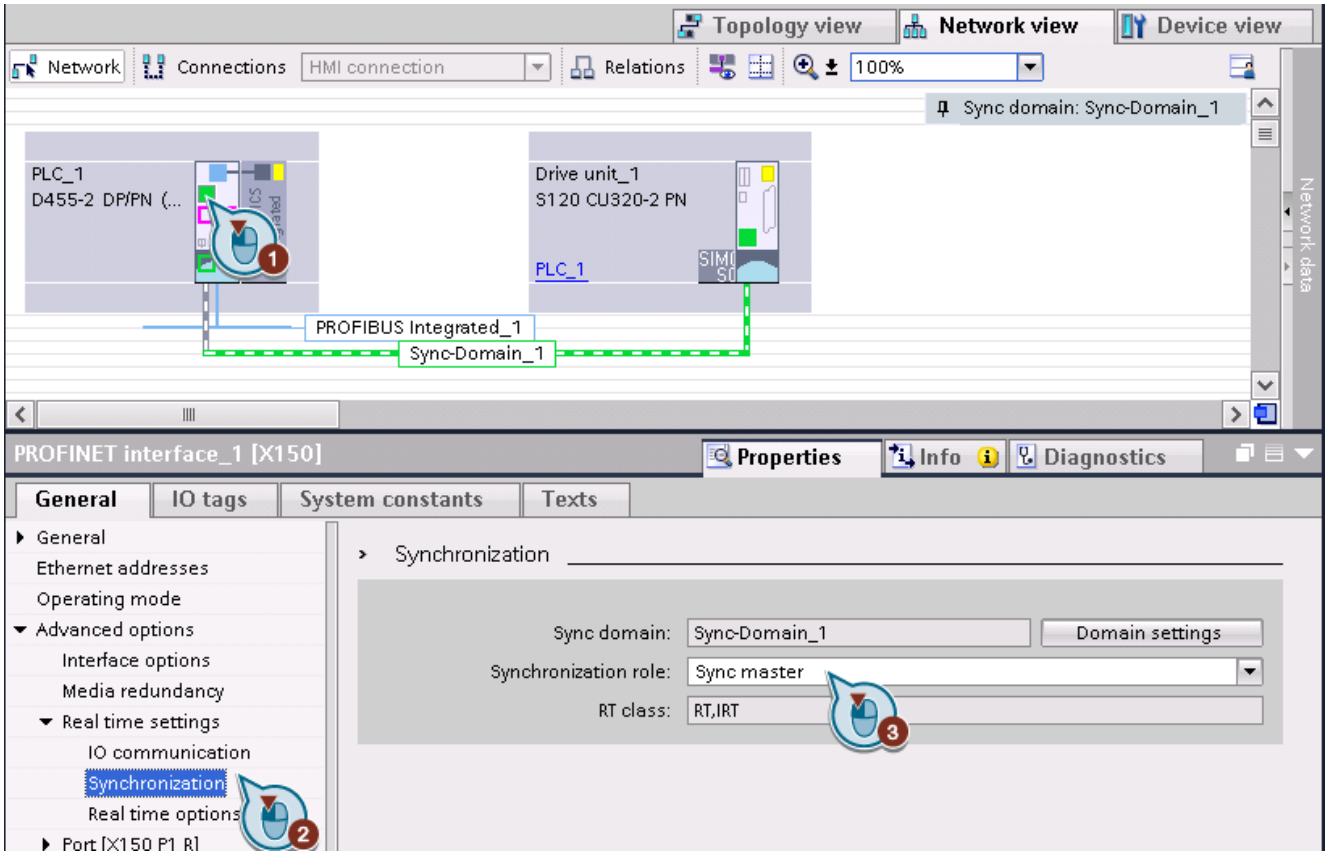


Figure 3-363 Configuring the sync master

Configuring the IO device as a sync slave

To configure a device within a PROFINET IO system as a sync slave, proceed as follows:

1. Select the PROFINET interface for the sync slave in the network view.
   The interface settings are displayed in the Inspector window.
2. Click "Extended options > Real-time settings > Synchronization" on the "General" tab.
3. Select "IRT" as RT class. In the example, the "sync slave" entry is selected automatically as "synchronization role".

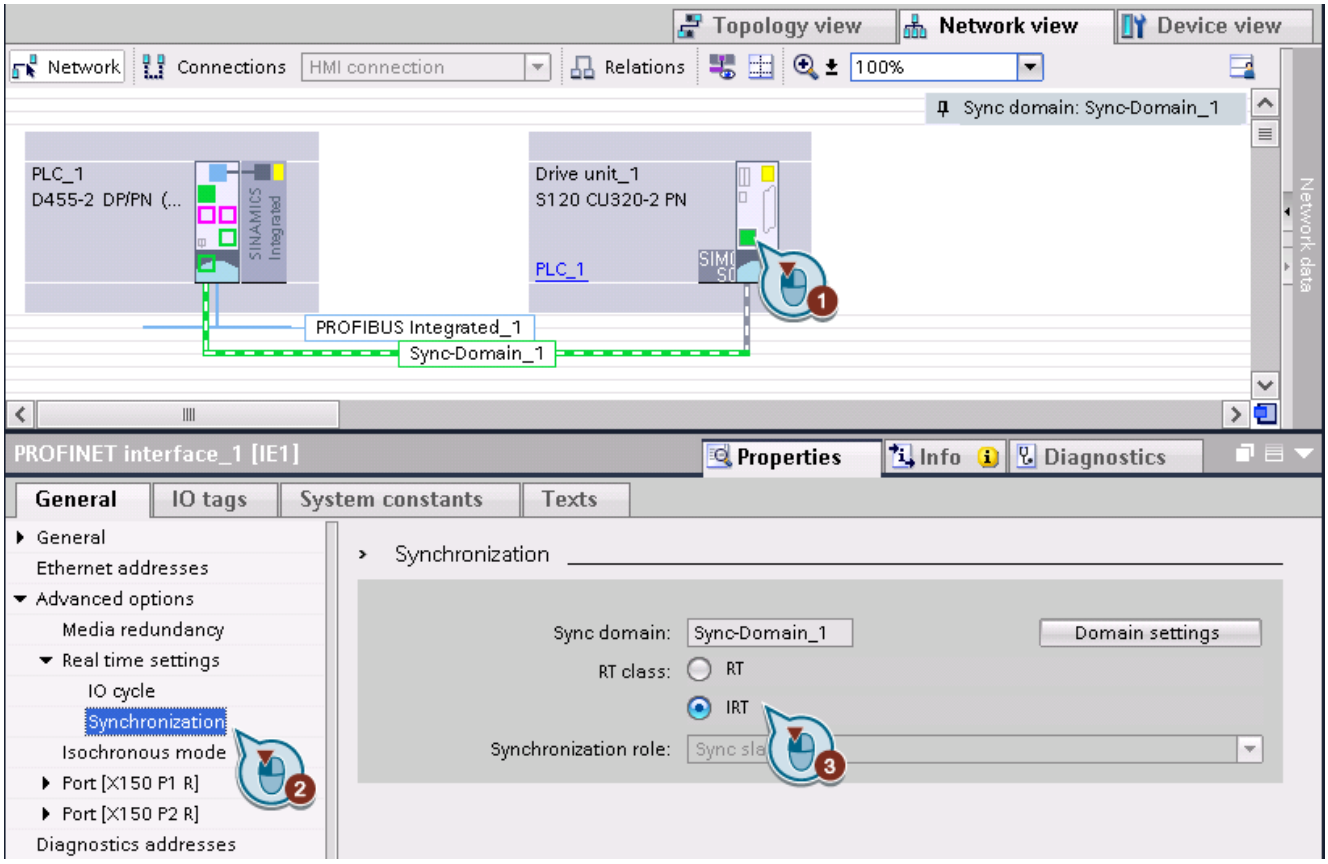


Figure 3-364 IO device as a sync slave

See also

Configure isochronous IO device (Page 701)

Configuring the topology, interconnecting ports

Interconnecting ports

A requirement for IRT is that the topology is configured and settings made to determine which device is to be connected via which port to which other devices.

There are two options for defining the properties of the cables between the ports of the switches:

- Using the topology view
- Using the object properties

Use the topology view

With the topology view you have an overview of all ports in the project and can interconnect them centrally.

To configure the topology, proceed as follows:

1. Click the "Topology view" tab.
   You can perform the following in the topology view:
   - Interconnect ports
   - Modify the properties of the interconnection
   - Insert passive components
   - Arrange for an offline/online comparison to be displayed in online mode
2. If you hover the mouse over the ports, port names are displayed in the tooltip. In the Inspector window, the properties for each selected port/ device are displayed.
3. Click on the source port and drag-and-drop the connection to the target port.
   The topology view then shows the connection between the two ports.

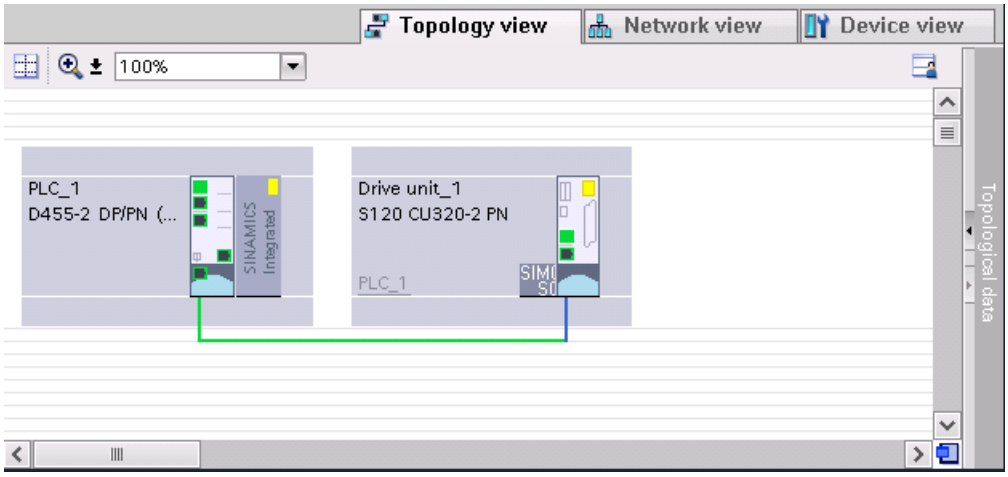


Figure 3-365 Topology view

Port interconnection via the properties

After selecting a device or an interface, you see the entries for the ports in the Inspector window under "PROFINET PN/IO interface > Extended options" in the "General" tab.

1. Select the interface, whose ports you want to edit, from the Inspector window.
2. Select the interface you wish to interconnect.
3. Select the "Port interconnection" entry.
   The "Local port" and "Partner port" dialogs open.
4. Open the "Partner port" drop-down menu.
   All devices with their PROFINET interfaces are shown.
5. Below the respective interface, select the port you want to connect to the selected port.
   Ports that you cannot interconnect are marked in red. Ports that are already occupied are not displayed.
   The connection between the ports is displayed in the topology view.
6. If necessary, edit the other parameters of the port.

Additional references

For more information on this topic, refer to the Online Help of the TIA Portal.

Configure isochronous IO device

Configuring isochronous mode for IO devices

The drive and controller together form a sync domain. For motion control applications of the PROFINET IO system, you must parameterize the IRT operating mode (for isochronous drives) and the PROFINET settings. Isochronous mode for the application on the bus is only possible for PROFINET IO with IRT. In the case of PROFINET IO with IRT, a sync master generates a synchronization telegram to which all sync slaves synchronize themselves. To use isochronous mode, you must configure it for the IO device.

The PROFINET IO system is isochronous with the servo clock. If you want to use the Servo_fast clock, you must select it under "Isochronous mode" on the IO controller.

Note

Isochronous mode only applies if isochronous submodules are configured in the IO device so that the IO controller has isochronous IO submodules. Otherwise, the Save and compile in the TIA Portal will be terminated with an error message.

To configure isochronous mode in the TIA Portal, you should therefore have already configured the drive and an axis in the SIMOTION SCOUT TIA. In this case, the necessary telegrams and isochronous submodules will be generated on Save and compile (symbolic assignment in SIMOTION SCOUT TIA). This is precondition for configuring the isochronous mode in the TIA Portal. After the configuration in the SIMOTION SCOUT TIA, isochronous mode is usually preset.

To configure isochronous mode, proceed as follows:

1. Select the IO device in the network view and click the PN interface.
   The interface properties are displayed in the Inspector window.
2. Click "Extended options > Isochronous mode" on the "General" tab.
3. Activate the "isochronous mode" option.
4. In the "detailed overview", the drive objects with the appropriate telegrams are displayed. The objects and telegrams must have been configured previously in SIMOTION SCOUT TIA. The telegrams are parameterized automatically during the symbolic assignment.
5. If necessary, activate the "isochronous mode" option besides the drive object with the appropriate telegram. This setting cannot be changed for telegrams for which isochronous mode is essential (e.g. telegram 105). In the example, telegram 105 is created automatically during symbolic assignment in SIMOTION SCOUT TIA.

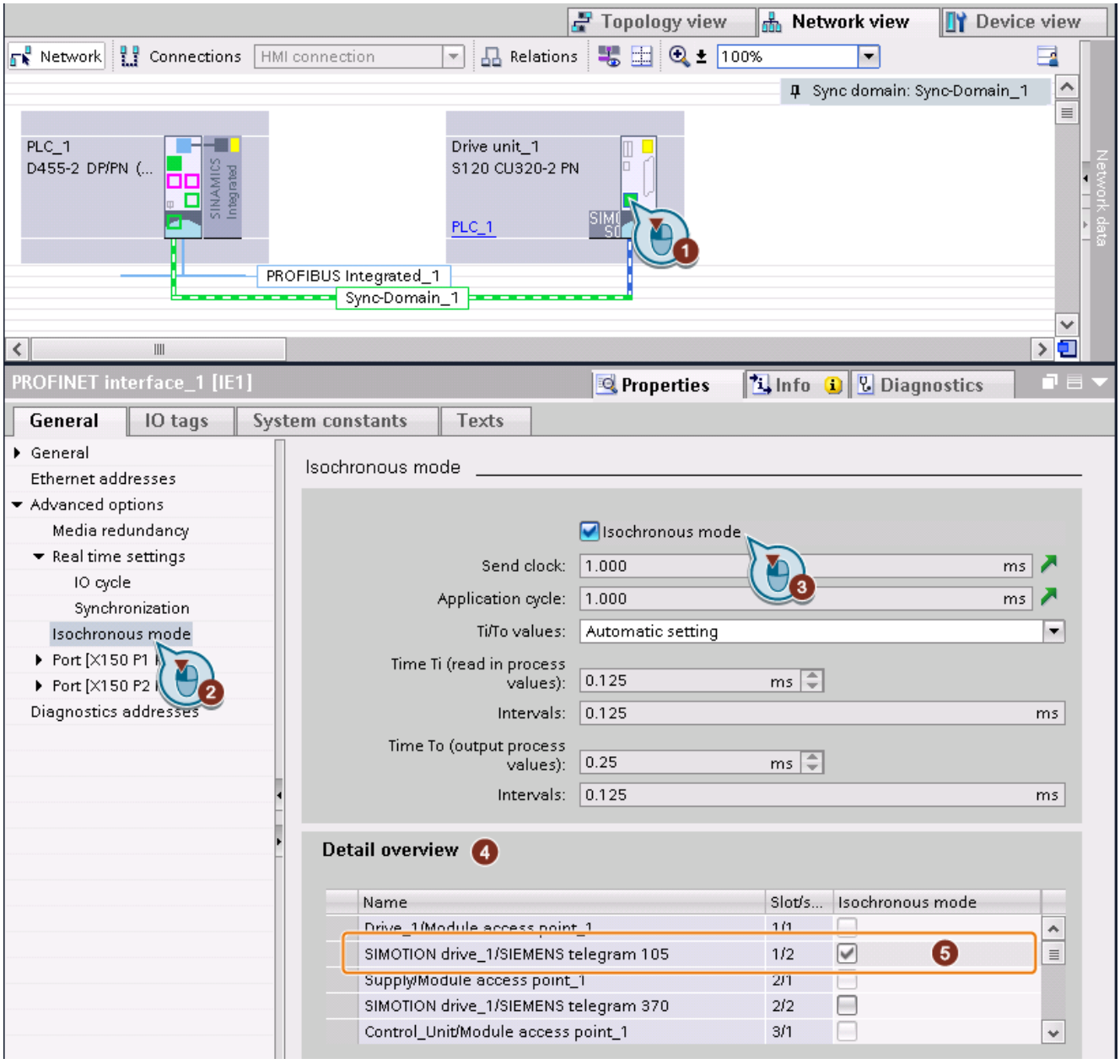


Figure 3-366 Configure isochronous IO device

Parameterizing isochronous mode

Isochronous mode can be individually activated or deactivated for each IO device or module in the IO device. You will find a more detailed description of the possible settings for isochronous mode in the following table.

Parameter	Description
Isochronous mode	Activate this option if you want to operate the drive (IO device) in isochronous mode with the PROFINET IO.
Send clock	Shows the send cycle clock of the PROFINET IO system. Click the link to edit the setting.
Application cycle	Shows the application cycle (multiple of the data cycle).
Ti/To values	Three settings are possible for the Ti/To values: • Automatic: Ti/To is automatically calculated. • Manual: Ti/To can be edited. • Task: Ti/To values are transferred to the OB. Ti, To: can only be edited if "manual" is set. Ti_{Min}/Ti_{Max}: minimum or maximum time for Ti. To_{Min}/To_{Max}: minimum or maximum time for To. Interval: defined interval for Ti/To values.
Time Ti (actual value)	Specifies at what time Ti, the actual value will be read in before the cycle starts.
Intervals	Indicates in which interval the value will be read in. The value should not be changed.
Time To (setpoint)	Specifies at what time To, the setpoint will be read out after the end of the cycle.
Interval	Indicates in which interval the value will be read in. The value should not be changed.

See also

Isochronous mode with servo cycle clock (Page 703)

Isochronous mode with Servo_fast cycle clock (Page 705)

Isochronous mode with servo cycle clock

Isochronous mode with the servo clock

For isochronous mode, the drive unit (IO device) must be synchronized with the PROFINET IO clock. You make these settings on the PN interface of the drive unit. In the previous chapter, you have already configured this. You can check isochronous mode in the IO controller.

Displaying isochronous mode in the TIA Portal

To display isochronous mode for a PROFINET IO system or DP master system, proceed as follows: A PROFINET IO system is shown in the example.

1. Select the device (interfaces) in the network view.
   The interface settings are displayed in the Inspector window.
2. Select "General > Isochronous mode" in the Inspector window.
   The TIA Portal initializes the settings automatically in accordance with their configuration; the settings cannot be changed.
   The application cycle is calculated and is derived from the settings in the TIA Portal and SIMOTION SCOUT TIA.

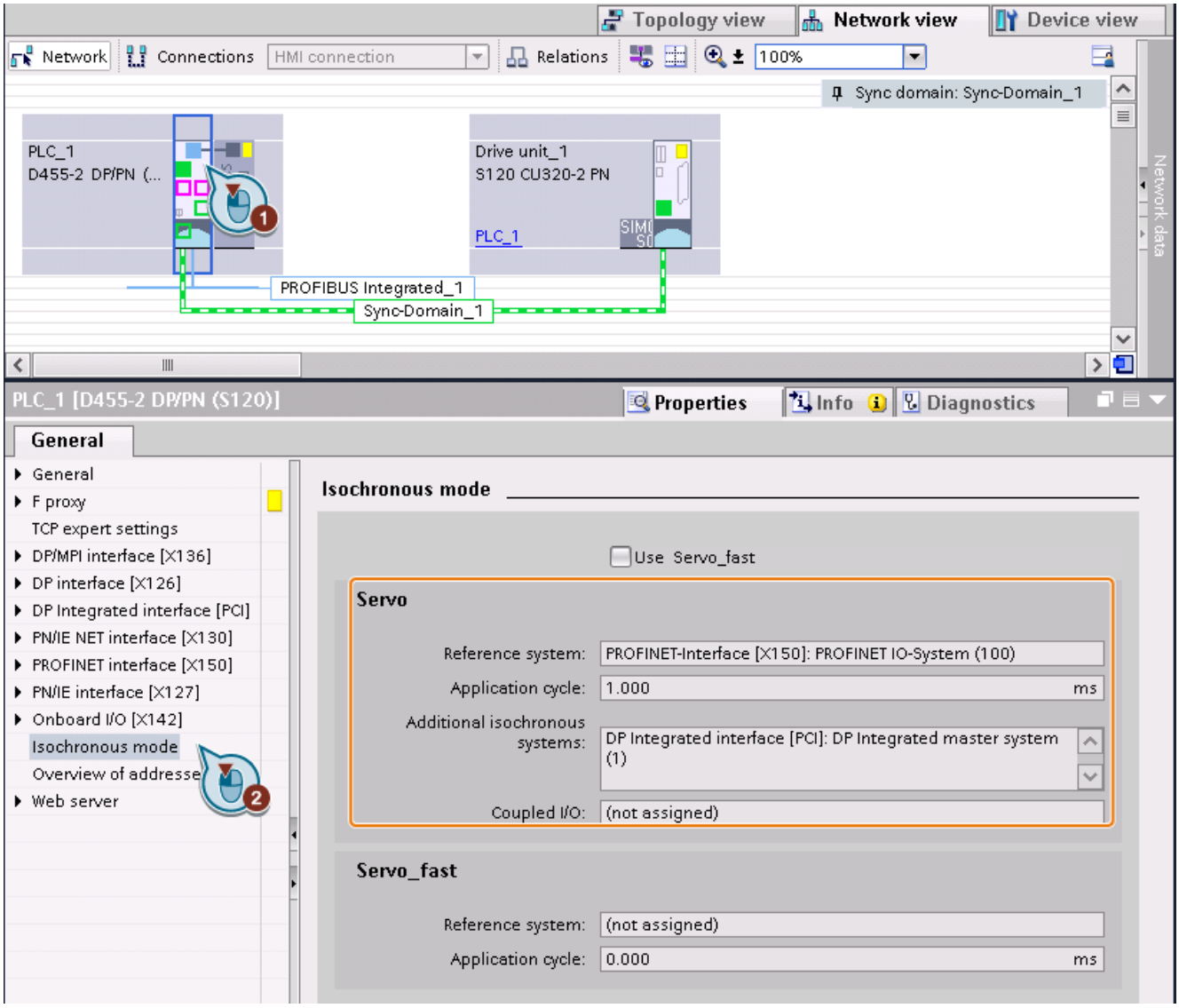


Figure 3-367 Setting the servo

The following parameters are displayed:

Parameter	Description
Reference system	Displays the reference system for the servo cycle in the following form: "Type of interface [interface name]: Master/IO system (number of master/ IO systems)". In the example, the reference system (clock generator) is the PN interface X150.
Application cycle	Shows the application cycle. The application cycle corresponds to the servo clock and can be set in SIMOTION SCOUT TIA. 1. Right-click the controller in the project navigator and perform "Set system cycle clocks." 2. Select an appropriate value in the "Factor" column.
Other isochronous systems	Displays the other (DP) master systems that are operated in isochronous mode.
Coupled IO	Displays the IO system on the [X1400] with the isochronously operated drive unit as the coupled I/O.

See also

Isochronous mode with Servo_fast cycle clock (Page 705)

Isochronous mode with Servo_fast cycle clock

Configuring Servo_fast

Servo_fast is configured like the Servo. That means the IO devices must be operated isochronously and the application of Servo_fast must be activated explicitly at the IO controller.

Note

Servo_fast is only supported by SIMOTION D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN. The requirements and constraints that apply to the Servo_fast can be found in the "SIMOTION SCOUT Communication" Function Manual.

For detailed information, read the descriptions of the servo settings:

Configure isochronous IO device (Page 701).

In the following example, a SIMOTION D455-2 DP/PN with two SINAMICS S120 CU310-2 PN drive devices is configured. The drives operate isochnronously. The drive device at the integrated PN interface (X150) should be operated isochronous with the Servo_fast; the drive device on the CBE30-2 (X1400) should be operated isochronous with the servo.

To configure Servo_fast, proceed as follows:

1. Select the device in the network view.
   The interface settings are displayed in the Inspector window.
2. In the Inspector window, select the "Properties > General" tab and click "Isochronous mode".

3. Activate the "Use Servo_fast" option.
   The settings are "read-only" and are filled according to the configuration. The application cycle is calculated and is derived from the settings in the TIA Portal and SIMOTION SCOUT TIA. The reference systems in this example for Servo and Servo_fast are X1400 CBE30-2 interface and X150 PN Integrated, respectively.

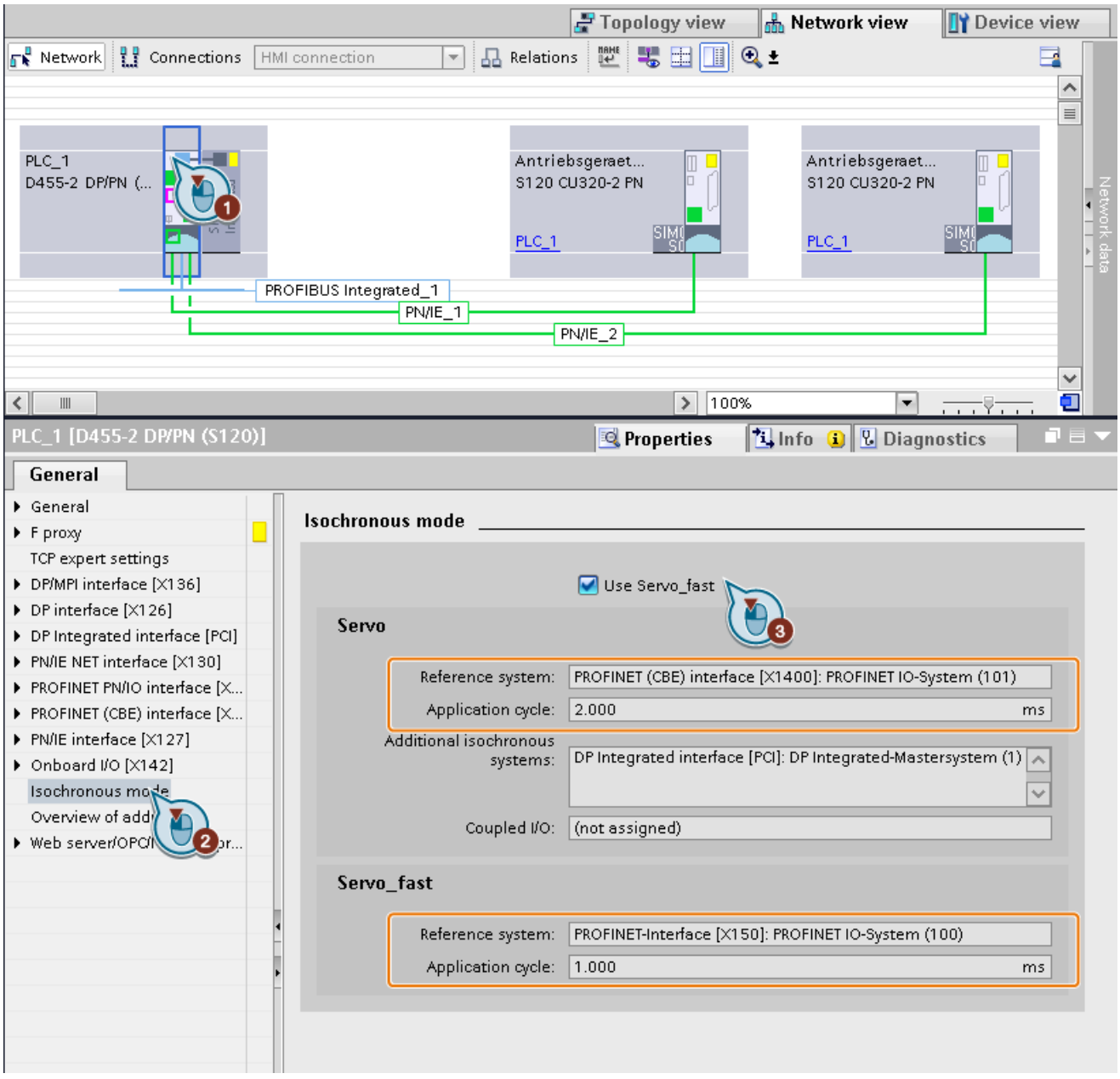


Figure 3-368 Assign Servo_fast

4. The application cycle corresponds to the servo cycle and can be set in SIMOTION SCOUT TIA. Switch to SIMOTION SCOUT TIA, right-click the controller and select "Set system cycles" in the context menu. In the opened dialog, select an appropriate value in the "Factor" column. In the example, Servo_fast is set to 1 ms and Servo to 2 ms.

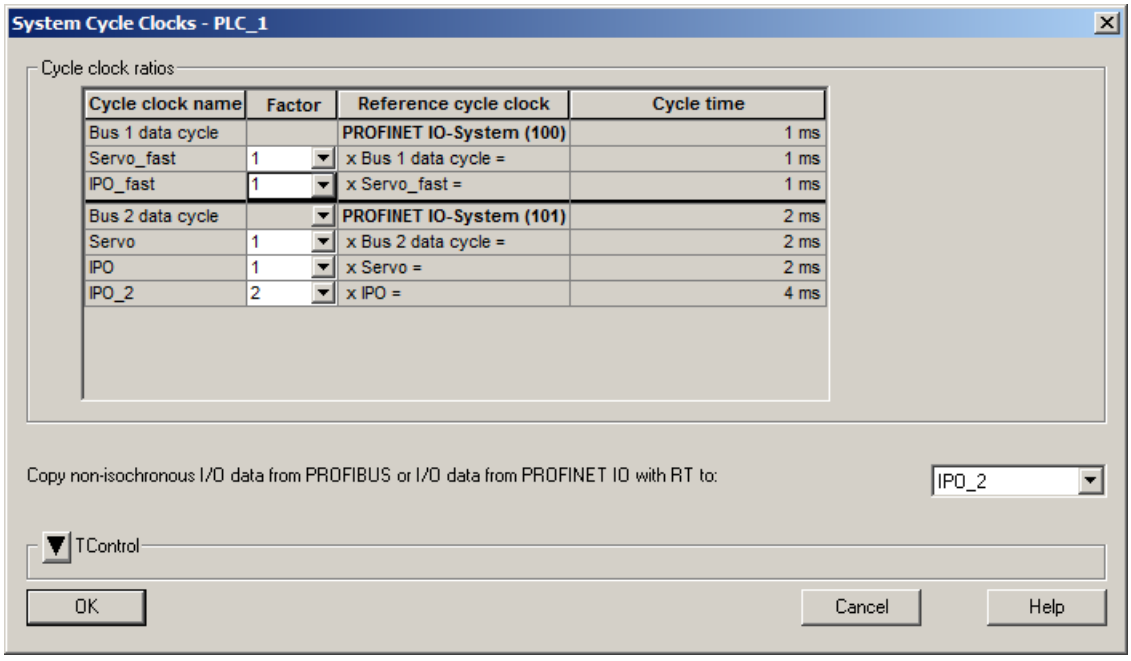


Figure 3-369 Setting system cycles for SIMOTION SCOUT TIA

5. Switch to the TIA Portal. There you can check the isochronous mode, the send cycle and the application cycle for the IO devices. A send cycle has been entered as 1 ms (i.e. synchronously to the Servo_fast cycle) at the IO device at the integrated PN interface (X150).

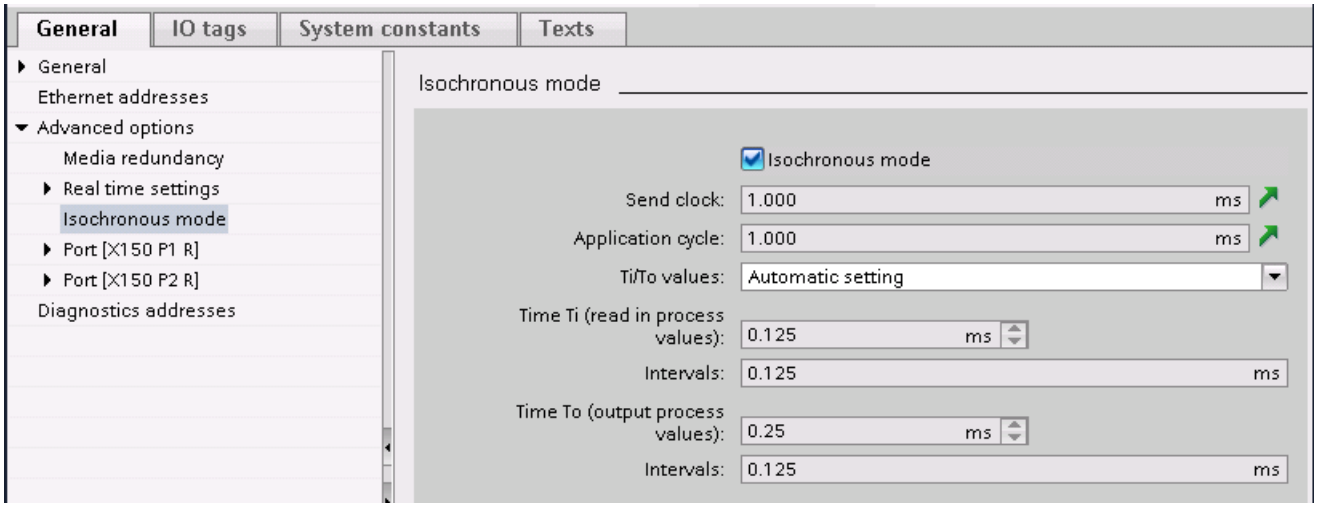


Figure 3-370 IO device configured isochronous to the Servo_fast

6. At the IO device at the PN interface of the CBE30-2, the send cycle has been entered with 2 ms, which means synchronously to the servo cycle.

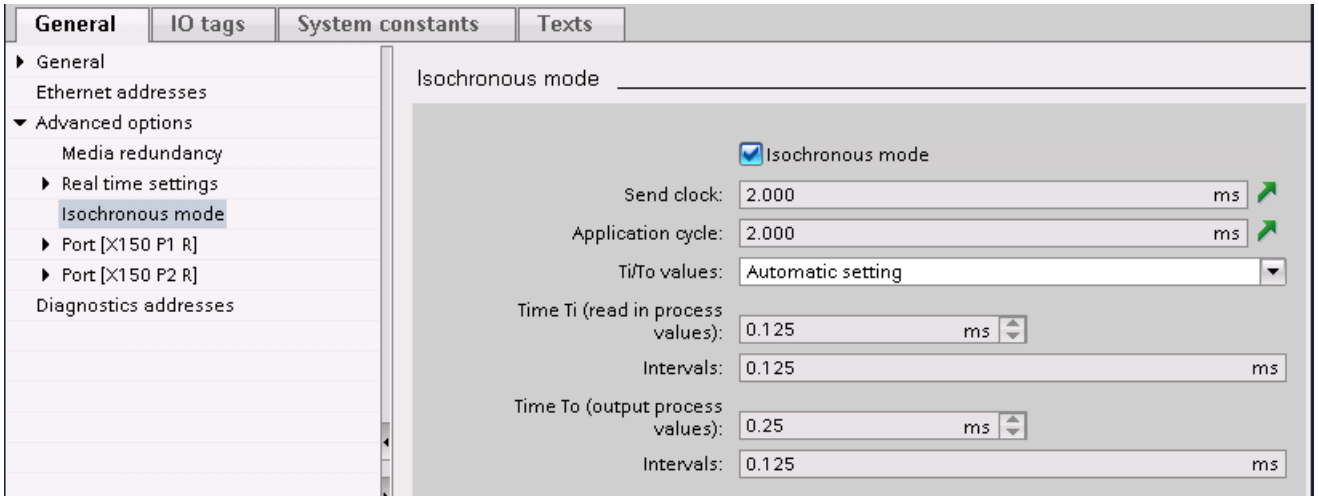


Figure 3-371 IO device configured isochronous to the servo

IO controller as IO device (I-Device)

Configuring I devices

I-Device description

The "I-Device" (intelligent IO device) functionality of a controller enables exchange of data with an IO controller. Here the I-Device is connected as an IO device to a higher-level IO controller. Communication can be established in both directions (bidirectional) via I/O areas using the I-Device functionality. This function also allows both controllers to operate in separate projects. When operating as an I-Device, a SIMOTION device can be used for data exchange with a SIMATIC station, for example.

Additional references

You will find detailed information in the information system of the TIA Portal under the keyword "I-Device" and in the "SIMOTION SCOUT Communication" Function Manual.

I-Device properties

With SIMOTION, the I-Device is available for PROFINET IO with RT and IRT. The restrictions that exist for operation as IRT or RT are contained in the following table.

Table 3-42 RT and IRT I-Device combination options with SIMOTION

SIMOTION functions	**Possible additional functions**			
	RT I-Device	RT controller	IRT I-Device	IRT controller
RT I-Device	-	X	-	X
RT controller	X	-	X*	X*
IRT I-Device	-	X	-	-
IRT controller	X	X	-	-

*Either an IRT I-Device or an IRT controller

Configuring I-Device

In the following example, a SIMOTION D455-2 DP/PN with isochronous SINAMICS S120 operates as I-Device on a SIMATIC S7 CPU 1516-F. If you want to operate a device as an I-Device, you must first activate the "IO device" operating mode.

In order to create an I-Device, proceed as follows:

1. In the network view, select the SIMOTION controller and the interface over which the I-Device is to be operated. In the example, X150.
2. In the Inspector window, select the "Properties > General" tab and click "Operating mode".
3. Activate the "IO device" option.

4. Assign the controller to which the I-Device is to be assigned at "Assigned IO controller".

Note

If the I-Device is to be used in another project, select "not assigned". To be able to use the I-Device, you must export the GSD and reinstall it. In the hardware catalog, this is displayed at "Other field devices" in a subfolder.

5. Activate the "Parameterization of the PN interface by higher-level IO controller" option if the interfaces of the I-Device and its ports are to be configured by the controller. This is the case if the I-Device is to be operated with IRT.
   If the I-Device is to be operated isochronously with IRT, you must also activate the "Operate in isochronous mode" option. This option is only available when the "Parameterization of the PN interface by the higher-level IO controller" option has been activated. If isochronous operation of the I-Device is not possible (e.g. already configured as IRT controller), you cannot activate the "Operate in isochronous mode" option.

Note

If you operate the I-Device with a lower-level IO system, the PROFINET interface (e.g. port parameter) of the I-Device cannot be parameterized by the higher-level IO controller.

Note

Restriction to configuring an isochronous IRT I-Device

Configuring an isochronous IRT I-Device in a single project in the TIA Portal is not supported. When an isochronous IRT I-Device is configured, the configuration must be divided between two projects.

This restriction also applies to migrated SIMOTION SCOUT projects. The SIMOTION SCOUT project must be divided between two projects before it is migrated.

This restriction does not apply to RT I-Devices.

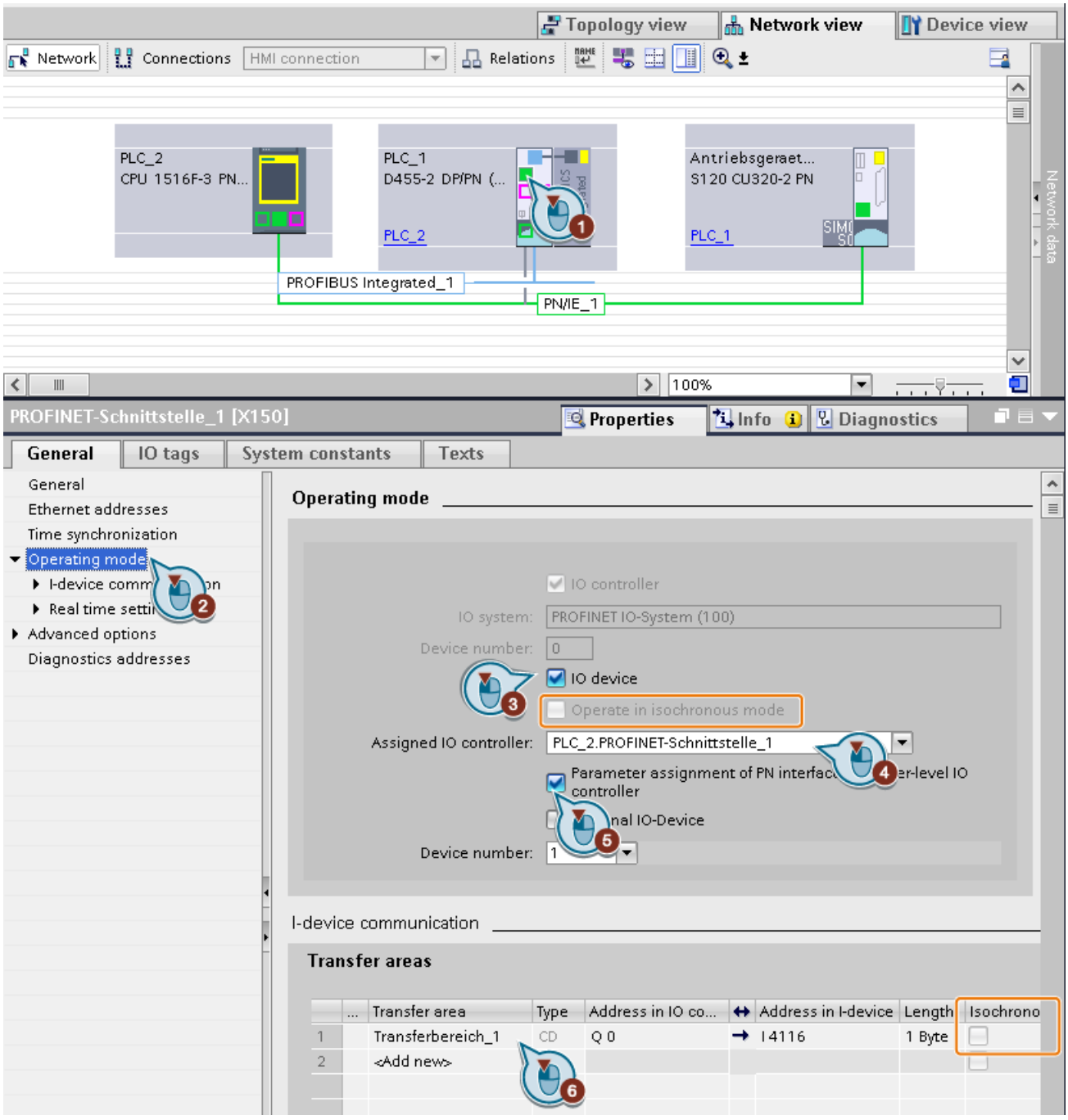


Figure 3-372 Setting the I-Device

6. Once the "IO device" option has been activated for the controller, you can define the transfer area. It is necessary to define a transfer area for the data exchange.

Parameter		Description
⑥ Transfer areas		Transfer areas are the peripheral areas over which the I-Device communicates data with the higher-level IO controller.
	Add new	Add a transfer area. A name is created automatically for the transfer area, the addresses are assigned in the IO controller and I-Device, and the length of the transfer area is set. You can still edit the following data:
	Type	Select the "Type" CD (Controller Device communication relationship).
	TransferArea_x	After you have created a transfer area, "Details of the transfer area" are displayed, where you can view the settings and change them, if necessary.
Diagnostic address of the communication		Here, you see the diagnostic addresses for each module and its ports/interfaces.
Export the device description file (GSD)		If you use the I-Device in another project or if the I-Device is used in another engineering system, then configure the higher-level IO controller and the I-Device as described here, in principle. Click the "Export" button after configuration of the transfer areas to create a GSD file from the I-Device. This GSD file is the representative of the I-Device configured in the other project.

Exporting and installing an I-Device

You can also export an I-Device as a GSD file and then install it in the TIA Portal. With this functionality, you can insert the I-Device in other projects via the hardware catalog.

1. In the "Operating mode" window, select the "not assigned" entry at "Assigned IO controller".
2. Configure the transfer area.
3. The next step is to export the GSD (generic station description). Click the "Export" button in the same window in the field "Export generic station description (GSD)".
4. In the dialog that opens, enter a name and description and specify the export path.
5. Confirm with "Export". The GSD file will be stored in the specified folder.
6. Select "Options" > "Install GSD file" in the menu.
7. Navigate to the storage location of the exported GSD file and select it in the window that opens.
8. Click "Install". The GSD file will be installed. In the hardware catalog, the I-Device is displayed at "Other field devices" in a subfolder.
9. With a drag-and-drop operation, you can, for example, insert the I-Device into another project.

Real-time settings on IO devices and I devices

Real-time settings for IO devices

The real-time settings are displayed when you set the "IO device" operating mode. Accordingly, these settings are also available for I-Devices:

1. Mark the SIMOTION CPU and select the "Real-time settings" for the PN interface in the "Properties" tab in the Inspector window.
   The table shows the possible options.

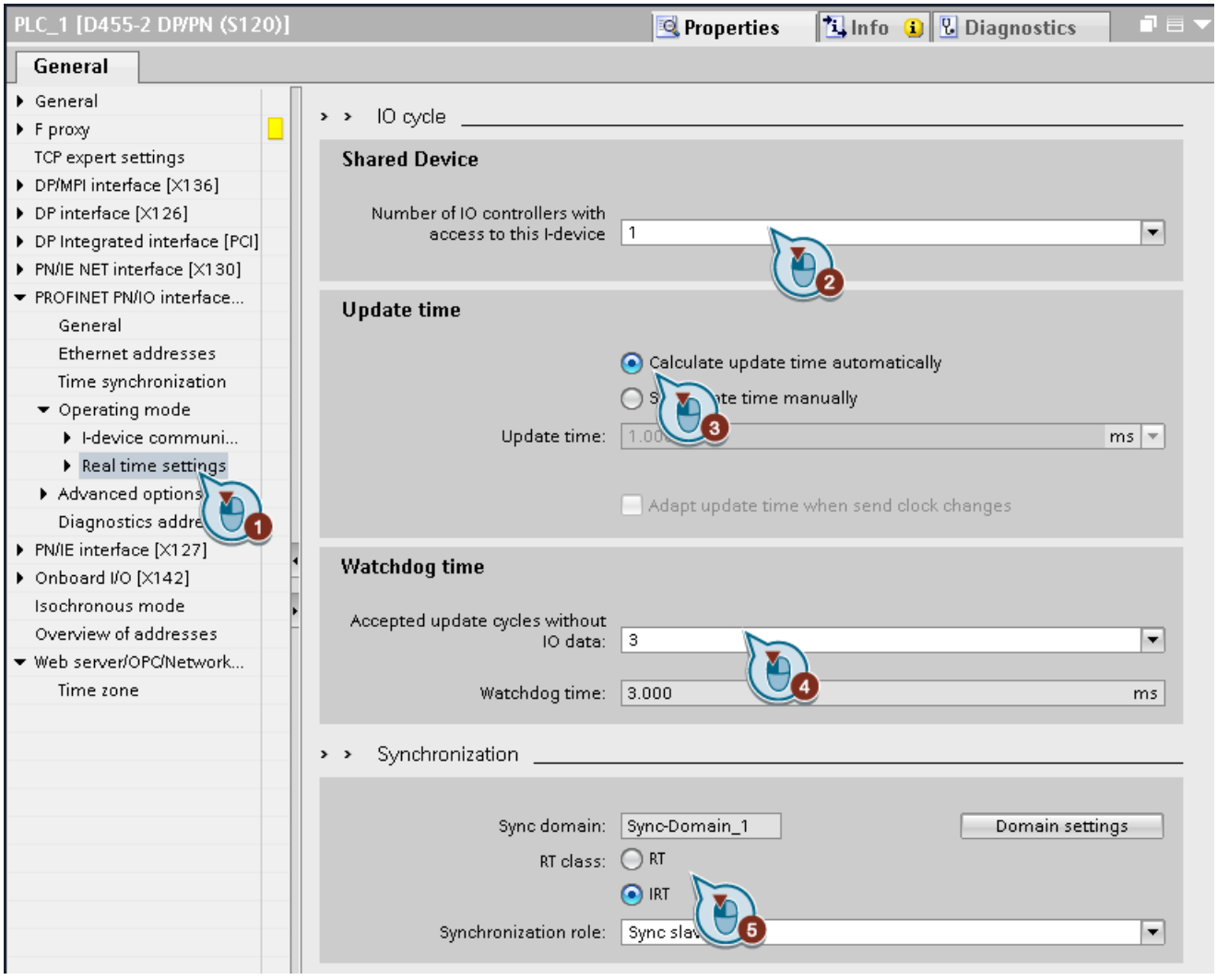


Figure 3-373 Real-time settings on the I-Device / IO device

You can set the following parameters:

Parameter		Description
② Shared device		From the drop-down menu, select the number of IO controllers to have access to this I-Device (only possible if you have configured an I-Device). **Note:** A SIMOTION I-Device can be operated only on **one** IO controller. As shared I-Device is not possible. A number > 1 causes an error message during the compilation.
③ Update time		Within this time interval an IO device / IO controller in the PROFINET IO system is supplied with new data by the IO controller / IO device. The update time can be configured separately for each IO device and determines the interval at which data is sent from the IO controller to the IO device (outputs) as well as from the IO device to the IO controller (inputs).
	Automatic	Automatically calculates the update time for each IO device of the PROFINET IO system, taking into account the data volume and the send cycle set.
	Adjustable	Activate the option to enter the update time manually. However, this can lead to errors if the bandwidth is not sufficient, or for example, too many nodes are configured.
	Adjust the update time when changing the send cycle	Automatically adjusts the update time.
④ Response monitoring time		If the IO device is not supplied with input or output data (IO data) by the IO controller within the response monitoring time, it switches to the safe state.
	Accepted update cycles without IO data	Enter here the number of cycles in which the IO device may be without data from the IO controller.
	Response monitoring time	Displays the current response monitoring time.
⑤ Synchronization		Can only be edited if the "Parameterization of the PN interface by the higher-level IO controller" option has been activated.
	Sync domain	Displays the name of the assigned sync domain. Click the "Domain setting" to edit the sync domain.
	RT class	Select the current RT class here. The following are available: • RT; for the cyclic data transmission • IRT; for isochronous real-time data transmission **Note:** The I-Device can operate only in IRT mode when IO devices connected to the I-Device operate only in RT mode.
	Synchronization role	Indicates the synchronization role the IO device has. The following are possible: • Unsynchronized • Sync master • Sync slave

See also

IO device (drive) on PROFINET IO (Page 693)

Direct data exchange via PROFINET IO

Direct data exchange

Direct data exchange between two SIMOTION controllers

IO data areas can be exchanged cyclically via IRT between two or more SIMOTION controllers. This is also referred to as controller-controller data exchange broadcast. Controller-controller data exchange broadcast is possible only between SIMOTION controllers via PROFINET IO with IRT. The following requirements must be met:

- The controllers are interconnected to the subnet.
- The topology is configured (ports interconnected).
- The PN-IO system is configured.
- The synchronization roles are assigned.
- The devices are in a single sync domain.

Procedure

To configure controller-controller data exchange broadcast, proceed as follows:

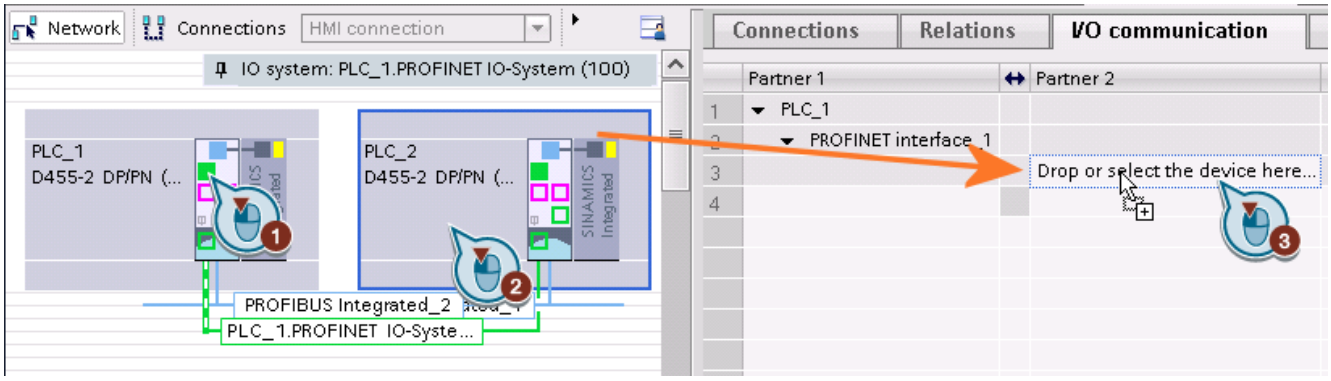


Figure 3-374 Drag-and-drop direct data exchange

1. Call the network view and click the "I/O communication" tab in the foreground below "Network data".
2. At the transmitter module, select the interface that you want to use for direct data exchange. At "I/O communication", the modules that are configured in this subnet are displayed.
3. Drag-and-drop the receiver module onto the transmitter interface.
   The module is displayed with the corresponding direction of communication.
4. Repeat the operation to configure the reverse direction of communication.
5. Select the "receiver module" to configure it as a transmitter and drag the "transmitter module" onto the interface in the "I/O communication".
   The two configured communication paths are now displayed.

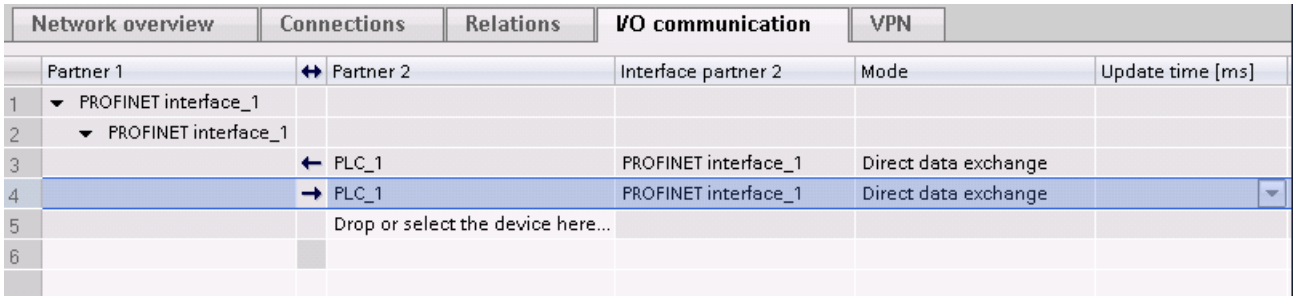


Figure 3-375 I/O communication

Configure transfer areas

It is necessary to define transfer areas for direct data exchange. Once you have selected the interface in the "I/O communication", the "Direct data exchange" area is displayed below the "General" tab in the Inspector window.

To configure a transfer area, proceed as follows:

1. On the "General" tab, click the "Add new..." entry in the "Transfer areas" table.
   The transfer area is created and displayed with the given data.
2. Repeat the process to create the transfer area for the reverse direction.

Note

The transfer areas must be configured both in the transmit and the receive direction (bidirectional data communication).

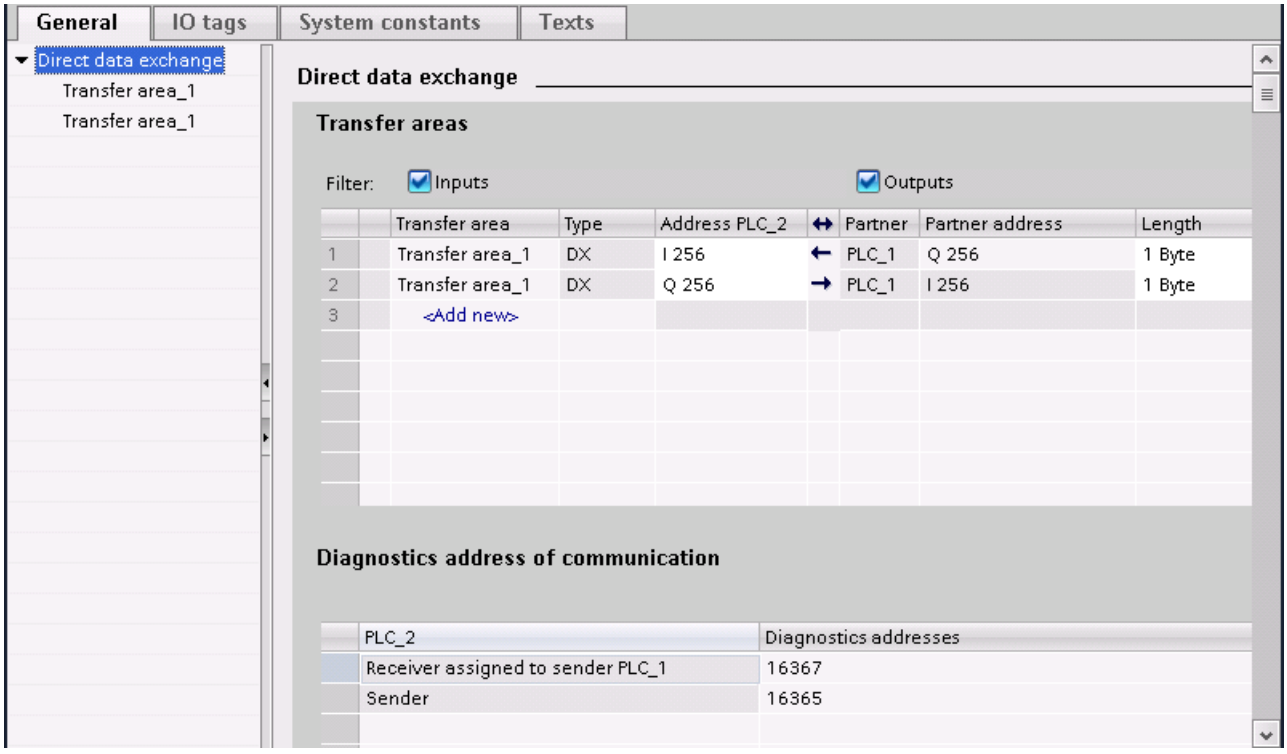


Figure 3-376 Configure transfer areas

3. In order to parameterize the transfer areas further, click the "Transfer area 1", ... "Transfer area x" entries in the secondary navigation.
   The corresponding parameters are displayed for each transfer area.

The diagnostic addresses for transmitter and receiver are illustrated at "Diagnostic address of communication".

If you want to configure direct data exchange for additional controllers, add them to the subnet, and perform the configuration similarly.

Type of transfer areas

In the TIA Portal, different types of communication (PROFIBUS, PROFINET) can be configured. Depending on this configuration, different short codes are shown for the transfer area type.

- PROFINET IO
  For Direct Data Exchange, "DX" is displayed as the type in the TIA Portal. For synchronous and distributed synchronism configurations that are generated automatically by SIMOTION SCOUT TIA, "M-DX" is displayed as the type. M-DX means Motion Control with Direct Data Exchange.

See also

Set up the sync master and sync slave. (Page 697)

Configuring the redundancy second sync master

Configuring the redundant sync master

There are machines/plants that consist of two submachines / plant units. For such machines/ plants (complete systems), it can be necessary that both submachines / plant units (subsystems) must be able to be operated synchronously, alone (separated) and in the complete system. This requires that a sync master is defined/configured on each subsystem. Only one sync master is permitted in the complete system. Consequently, a sync master is configured on one subsystem and a redundant sync master is configured on the other subsystem.

If the complete system is operated combined, either the sync master or the redundant sync master handles the synchronization of the complete system (depending on the switch-on sequence; for simultaneous switch-on, the sync master).

Example of a redundant sync master

In the following example, simple topology is configured with a redundant sync master. The project contains two SIMOTION D455-2 DP/PNs and one SINAMICS S120 CU320-2 PN. One SIMOTION D455-2 DP/PN is configured as the sync master (PLC_2) and the other as the redundant sync master (PLC_1). On the redundant sync master, the SINAMICS S120 is connected in isochronous mode via PROFINET.

For configuration of the sync master and sync slave, refer to Section Communication configuration SIMOTION controller with SIMOTION drive (Page 684).

To configure a redundant sync master, proceed as follows:

1. Select the PROFINET interface in the network view.
   The interface settings are displayed in the Inspector window.
2. Click "Synchronization" at "Real-time settings".

3. Select the "Redundant sync master" entry.

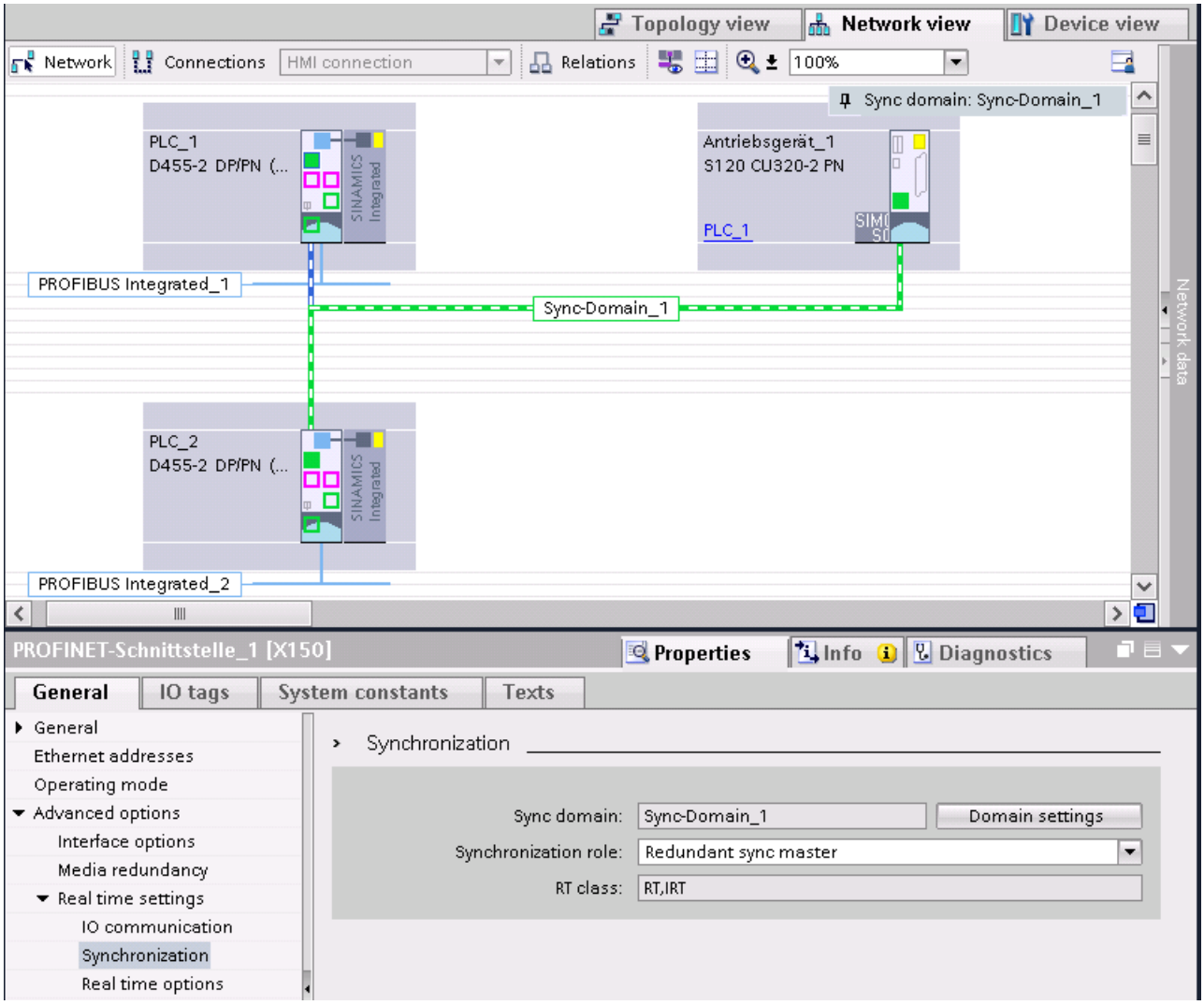


Figure 3-377 Redundant sync master

4. Switch to the topology view.
5. Interconnect the ports of the two SIMOTION controllers.

Note

In the topology view, the redundant sync master should be between the sync master and the sync slaves.

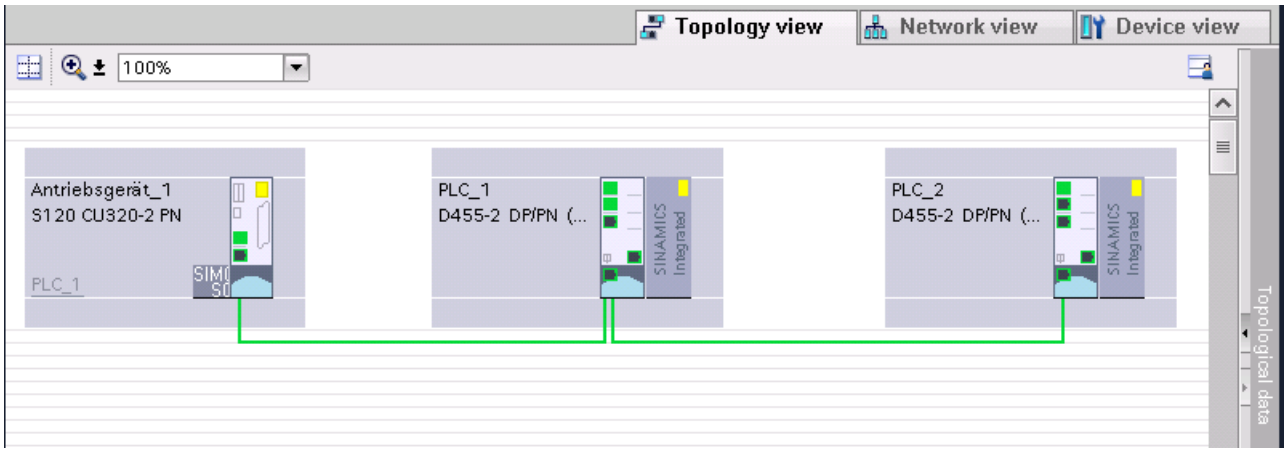


Figure 3-378 Topology view with redundant sync master

Media redundancy (MRP and MRPD)

Configuring MRP media redundancy

Configuring MRP (PROFINET RT)

It is possible to establish redundant networks via the so-called Media Redundancy Protocol (MRP). Redundant transmission links (ring topology) ensure that an alternative communication path is made available when a transmission link fails. The PROFINET devices that are part of this redundant network form an MRP domain. The Redundancy Manager coordinates the communication. All other members in the MRP ring are so-called redundancy clients, e.g. SINAMICS S120 drives.

Only devices with MRP-capable ports may be inserted in an MRP ring. In SIMOTION controllers, the ring ports are marked R. In MRP, no IRT communication can be configured, i.e. no PROFINET interface is set to "sync master" or "sync slave".

Additional references

You will find detailed information in the information system of the TIA Portal under the keyword "Media redundancy" and in the "SIMOTION SCOUT Communication" Function Manual.

Requirements

1. Generate a ring via the port interconnections in the topology view. If no rules are violated, a MRP domain is automatically created.
2. Check the media redundancy roles, e.g. who assumes the role of the "Redundancy Manager".
3. Select the device and click "Properties" in the Inspector window and then the "General" tab. You edit the parameters at the "Media redundancy" entry.

Port interconnection ring topology for MRP

In the topology view, you must configure the port interconnection of the manager with the clients to create a ring. In the example, a SCALANCE X-200 IRT as the redundancy manager, a SIMOTION D435-2 DP/PN, and a SINAMICS S120 CU310-2 PN as clients are configured.

1. Open the topology view.
2. Interconnect the individual ports of the devices. The interconnected ports must be ring ports. They are marked with an R.
3. Connect the two end points of the line topology with each other in order to close the ring, e.g. the redundancy manager with a redundancy client.

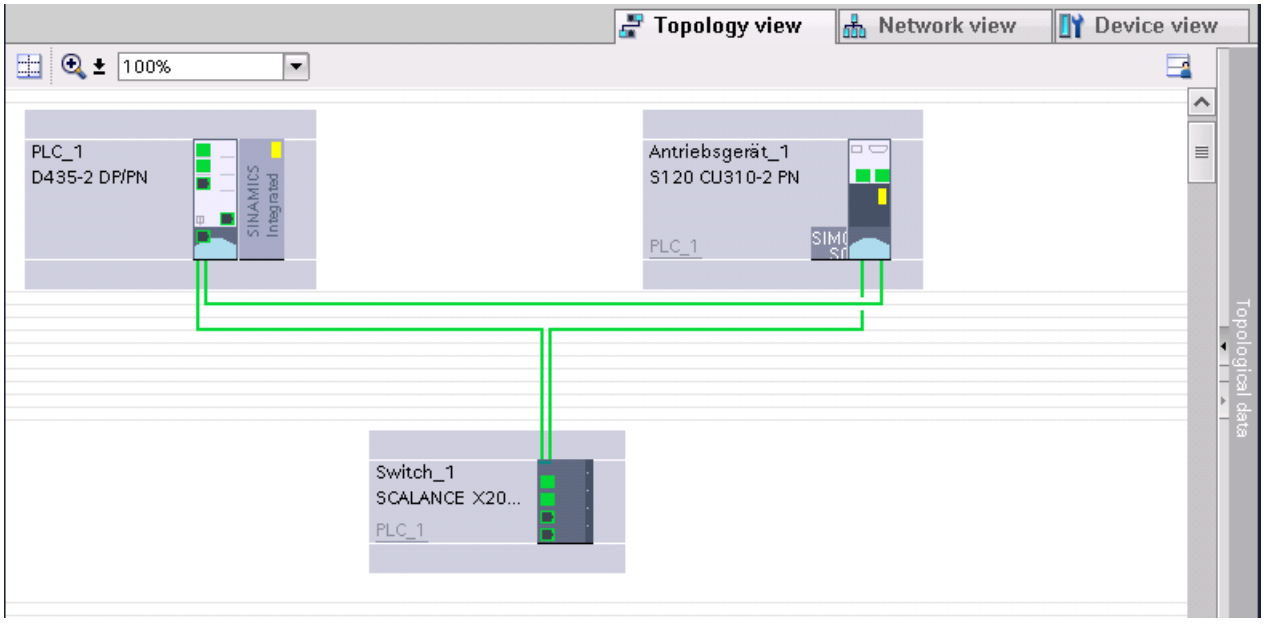


Figure 3-379 Port interconnection of a ring topology

MRP parameters

You can set the following parameters for media redundancy:

Parameter	Description
MRP domain	Shows the MRP domain that is assigned to the device. The assignment is made at Domain Management, see below.
Media redundancy role	Please select here the role the device is to perform in the ring: • Manager (Auto) if the device is to assume the manager role. The device must be able to fulfill this role. • Client if the device is part of the MRP ring. • Not node within the ring if the unit is not to participate in the MRP.
Ring port 1	Displays the interconnection of ring port 1.
Ring port 2	Displays the interconnection of ring port 2.
Diagnostic alarms (in the manager only)	Activate the option if you want to display diagnostic alarms.
Alternative redundancy (in the manager only)	Select this option if the properties of the media redundancy are parameterized by alternative mechanisms or tools.
Domain settings	Click this button to edit the domain settings. Then switch to the "Properties > General" tab of the PROFINET system. There, you can edit the settings for sync domains and MRP domains, see also Configure sync domains and send clock (Page 695).

In the following example, you see the media redundancy settings of a SIMOTION D435-2 DP/PN, which is configured as a client in the ring. The SCALANCE X-200 IRT is configured as a manager and the SINAMICS S120 CU310-2 PN as a further client.

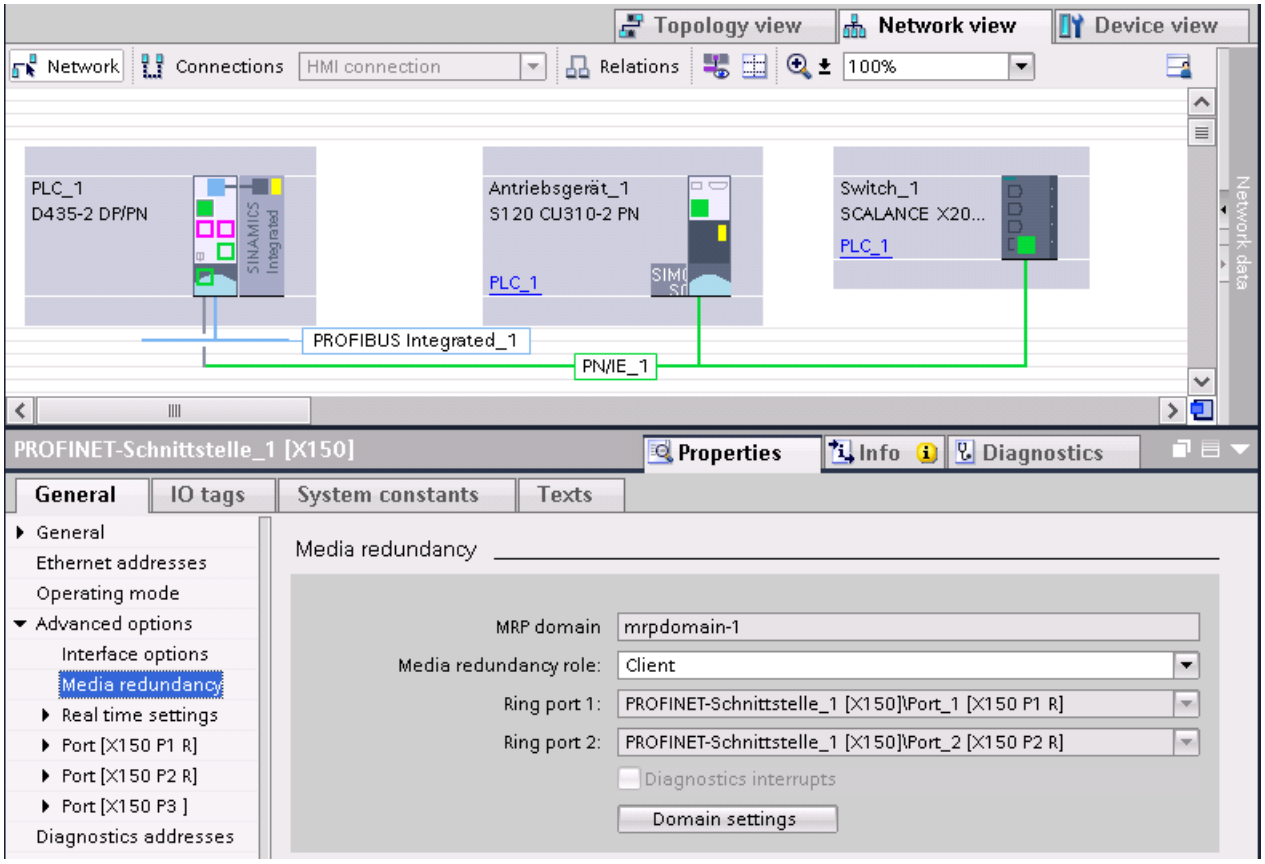


Figure 3-380 Media redundancy settings of a SIMOTION D435-2 DP/PN as redundancy client

Configuring the MRP domain

MRP domain

All devices that want to participate in the MRP must be part of an MRP domain. The ring topology can be accessed via the corresponding port interconnection.

Configuring the MRP domains

To configure the nodes of an MRP domain, proceed as follows:

1. Select the PROFINET IO system in the network view.
2. In the Inspector window, select the "General" tab and click the "MRP domain" entry.

The following parameters can be edited:

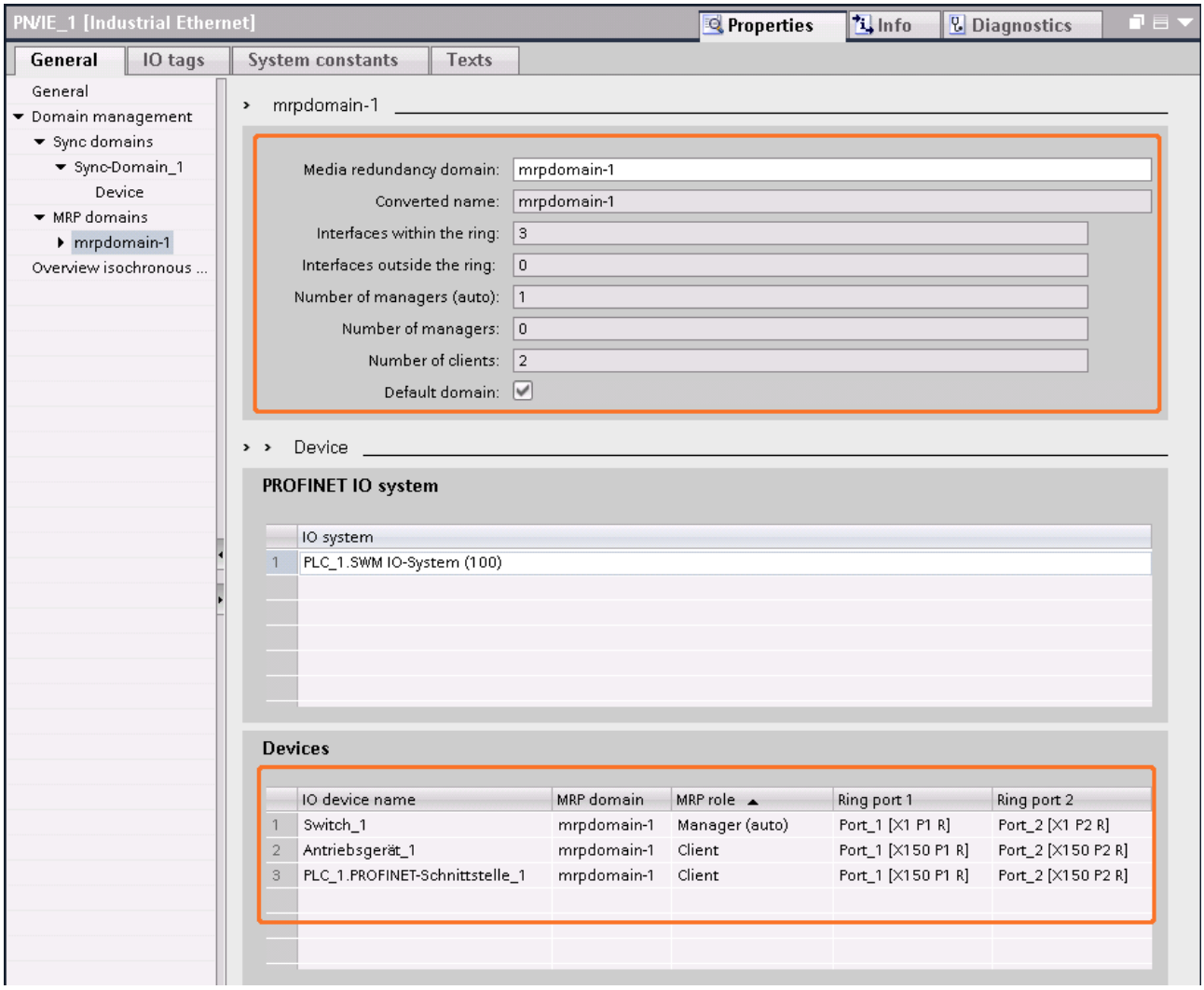


Figure 3-381 Configure MRP domain

Parameter		Description
MRP domains		
	Media redundancy domain	Displays the MRP domain to which the device is assigned.
	Converted name	Displays the converted name of the MRP domain.
	Interfaces within the ring	Displays the number of PROFINET interfaces that are located within the ring.
	Interfaces outside of the ring	Displays the number of PROFINET interfaces that are located outside of the ring.
	Number of managers	Displays the number of Redundancy Managers.
	Number of clients	Displays the number of Redundancy Clients.
	Default domain	Indicates whether the displayed MRP domain is the default domain.
PROFINET IO System		
	IO system	Displays the IO systems that are part of the MRP domain.
Devices		
	IO device name	Lists the device names of the PROFINET IO system nodes.

Parameter		Description
	MRP domain	Drop-down menu to select the MRP domain
	MRP role	Drop-down menu to select the role of the device in the MRP domain: • Not node within the ring • Client; is a redundancy client within the ring • Master; is a redundancy master within the ring
	Ring port 1	Drop-down menu to select the port through which the device is connected to the ring.
	Ring port 2	Drop-down menu to select the second port through which the device is connected to the ring.

Configuring MRPD media redundancy for IRT

Configuring MRPD (PROFINET IRT)

MRPD (Media Redundancy for Planned Duplication) is a method for smooth media redundancy with PROFINET IRT. MRPD also requires MRP.

The combination of MRP with MRPD provides bumpless PROFINET operation for short cycle times in the event of a fault in the ring. MRPD is based on IRT and ensures bumpless operation by the transmitter sending the cyclic data in both directions which the recipient then receives twice. The first received telegram is used by the recipient. The second telegram is discarded automatically. If the ring is interrupted at one location (e.g. through the failure of a ring node), receipt of the cyclic data via the problem-free side of the ring is still guaranteed.

Note

Bumpless media redundancy MRPD always requires the activation of MRP in the individual rings.

Requirements for media redundancy with MRPD

- All nodes in the ring must support MRPD.
- IRT must be configured for all components.
- All nodes in the ring must be configured as node.
- Devices not contained in the ring must assume the MRP role "Not node of the ring".

Configure MRPD and redundancy level

MRPD does not need to be activated explicitly. The function is available automatically once all MRPD requirements are satisfied. If MRPD is available, the redundancy level is displayed for each IO device. The redundancy level specifies the extent to which the real-time communication is affected in the event of a network interruption between an IO device and its IO controller.

In the following example, a SCALANCE X-200 IRT as the redundancy manager, a SIMOTION D455-2 DP/PN and a SINAMICS S120 CU320-2 PN as clients are configured. MRPD is set automatically when all nodes satisfy the requirements. For active MRPD, the "Redundancy level" column with the associated value is displayed for the IO devices.

Displaying the redundancy level

1. Select the PROFINET IO system in the topology view.
2. In the Inspector window, select "Properties > General" and open the "Nodes" of the sync domain in the secondary navigation. The redundancy level is displayed in the "IO devices" table.

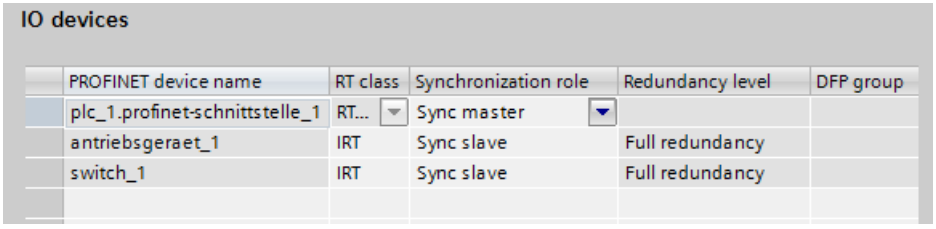


Figure 3-382 Redundancy level for available MRPD in the ring

Series machine projects (modular machine)

Series machine projects

Introduction

Series machine projects (basis for modular machine projects) are designed to help you configure and commission flexible automation solutions for series or modular machines.

A hardware configuration comprising an IO controller and any number of connected IO devices represents a "PROFINET IO system master". This master is a "maximum configuration" and a template from which various options can be derived for a variety of series machines with, for example, different design variants of the IO system.

SIMOTION SCOUT TIA provides you with a range of functions that will support you as you configure a master project from which you can derive modular machine instances.

Characteristics

- You can derive different variants (instances) from a project with maximum configuration.
- An IO system can be integrated using simple tools into an existing on-site network.

Functions for series machine projects

- Addresstailoring (reusable IO systems) Adaptation of IP addresses and device names on-site using simple tools.
- Machinetailoring (configuration control for IO systems) Vary the number of on-site IO devices

Additional references for TIA Portal

Additional information and application examples on the subject of Configuration control with SIMATIC S7 (https://support.industry.siemens.com/cs/ww/en/view/29430270) can be found in the Service & Support Portal.

Assigning device name and IP address via user program / DCP

Introduction

With the SIMOTION system functions, the IP addresses and the device names of the PN IE interfaces and the PN IO interfaces can be assigned from the user program or the DCP (Discovery and Configuration Protocol). An additional reboot is no longer necessary for the PN interfaces. This mechanism allows the IP addresses and device names to be adjusted without making changes to the project. The IP settings can be changed locally, in particular for series machines.

System functions

New system functions have been introduced as of SIMOTION V4.4 for assigning the IP address and device name.

- _setPnNameOfStation
- _getPnNameOfStation
- _getPnPortNeighbour (only PN IO interfaces)
- _setPnIpConfig
- _getPnIpConfig

Assigning the device name based on the MAC address (V5.2)

As of SIMOTION SCOUT V5.2, the device name can also be assigned based on the MAC address. The PN devices are named with the system function _assignNameOfStationToDevice().

Note

A detailed description of the system functions can be found in the online help of SIMOTION SCOUT TIA, in the *SIMOTION System Functions/Variables Devices* List Manual, and in the *SIMOTION Communication System Manual*.

Permit adaption of IP addresses and device names directly on the device

Introduction

The IP address and the device name can be assigned via a user program. The hardware must consider this during the configuration.

Procedure

1. Select in the network view or device view of the hardware and network editor of STEP 7, the PROFINET interface of an IO controller.
2. Navigate in the Inspector window to "Ethernet addresses".
3. Select the "Permit adaptation of the IP address directly on the device" option in the "IP protocol" area.
4. Activate the "Permit adaptation of the PROFINET device name directly on the device" checkbox in the "PROFINET" area.

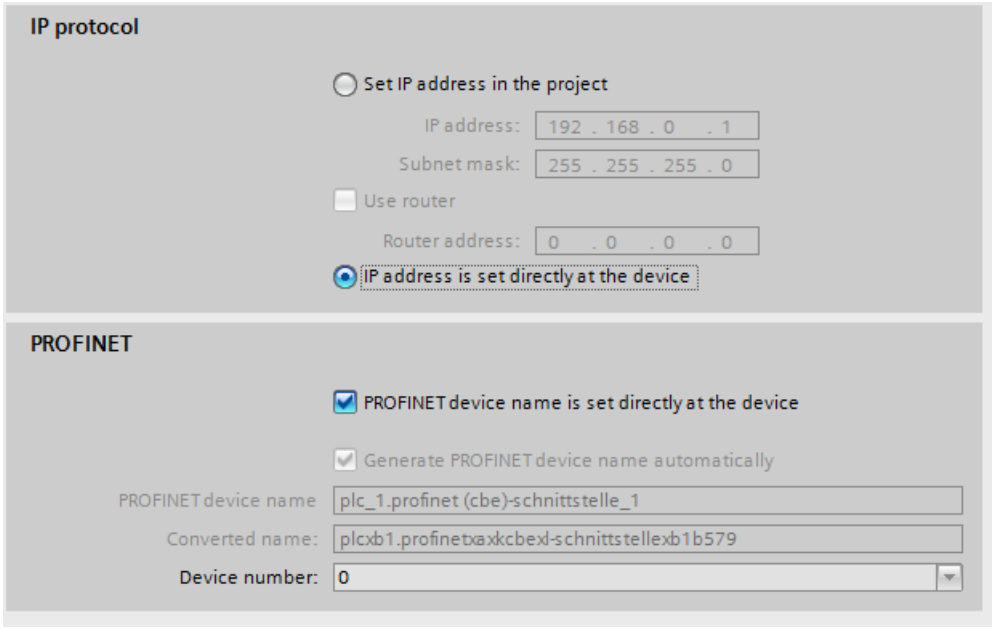


Figure 3-383 Permit assignment of device names and IP addresses differently (e.g. user program)

The IP address and the device name for the commissioning can be assigned via the user program or, for example, also via the PST (Primary Setup Tool) or PRONETA.

Permit overwriting the PN device name

Introduction

Previously, a topological naming of an IO device was possible only when the IO device did not have any device name (NameOfStation). To change existing device names of IO devices, they had to be reset. Only then could the devices be renamed by the IO controller. As of SIMOTION V4.5, overwriting a device name can be activated without reset for the hardware configuration. This option, for example, makes it easier for you to replace a device during automatic commissioning.

⚠ CAUTION

Unstable error states during partial commissioning

The activated option can cause the system to reach an unstable error state in the case, for example, of a wiring error, and wiring errors are difficult to diagnose.

After the wiring has been rectified, you must manually delete, for example, the device names of the IO devices to ensure that the names are correctly assigned.

⚠ WARNING

Incorrect PROFINET device names from the configuration due to incorrectly connected devices

Depending on the I/O connected, malfunctions may pose a danger to life or risk of serious injury or property damage if PROFINET device names are entered incorrectly because the device connections are incorrect.

To eliminate any risks, always check when a device is replaced whether the appropriate replacement device is connected and the port interconnection matches the configured set topology!

Principle

With activated "Permit overwriting the PROFINET device name" option, the IO controller (CPU) can overwrite the PROFINET device names of the IO devices in the IO system.

Note

The "Permit overwriting the PROFINET device name" option must be activated for address tailoring and is set automatically when "Reusable IO system" is selected.

Procedure

To change the "Permit overwriting the PROFINET device name" option, proceed as follows:

1. Select in the network view or in the device view the PROFINET interface or the CPU for which the option is to be changed.
2. Select the "Extended options" area, "Interface options" section.
3. Change the option.

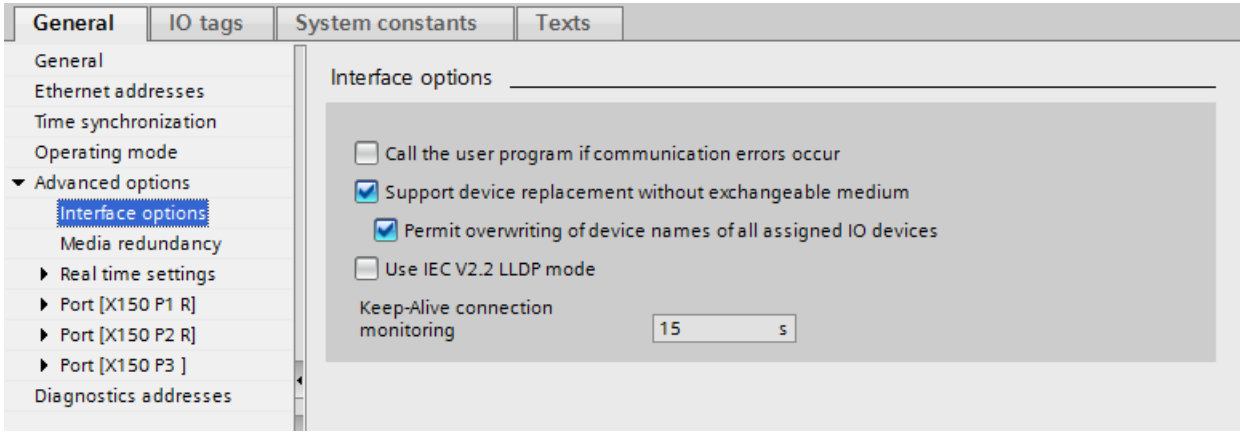


Figure 3-384 Permit overwriting the PROFINET device name

Reusable IO systems - Addresstailoring

Introduction

The Addresstailoring function assigns the IP address and NameOfStation for IO controllers and IO devices via system functions at commissioning time. This feature allows a machine consisting of an IO controller and IO devices to be supplied with IP addresses and NameOfStation, and permits multiple instances of the machine to be commissioned on a single physical Ethernet without the engineering system.

There is only one project (configuration and programs). It can be loaded onto machines of the same type without change. To integrate the machine in an existing network infrastructure, only limited customization is required for the on-site commissioning (IP address, NameOfStation). The user program assigns the IP address and the NameOfStation during the first controller startup.

Note

Addresstailoring can be configured only for SIMOTION SCOUT TIA.

Example of an automation solution

The following figure shows how an automation solution with a reusable IO system is loaded into various automation systems and one of these is then adapted on-site to the existing network infrastructure.

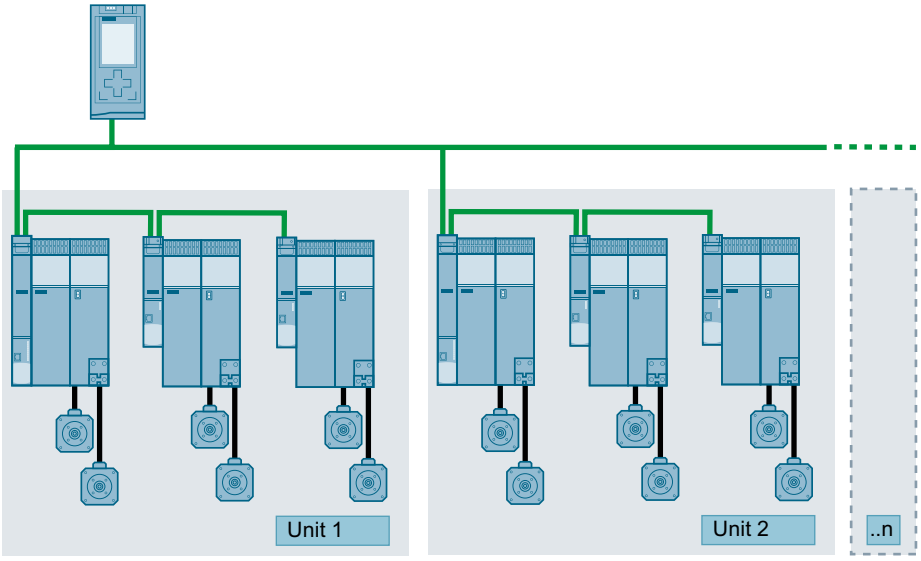


Figure 3-385 Addresstailoring: Series machine with the same units with a different name and IP address

Rules

The following rules apply to a reusable IO system:

- A series machine project consists of an IO controller and the associated IO devices. Consequently, configure only one CPU as IO controller and the associated IO devices in the series machine project.
- No IO device may be configured as shared device.
- If IRT is configured, all IO devices must belong to the same sync domain and the sync domain must not contain any further IO devices.

Alternatively to Addresstailoring via two PN interfaces

In the SIMOTION environment, under some circumstances, rather than using Addresstailoring, the same functionality can also be achieved by using the two PN interfaces. One PN interface is configured as I device and connected with the higher-level controller. The second PN interface is the IO controller for the IO devices. In this case, a second PN interface (e.g. CBE30-2) is required.

In the previously described configuration, the IO devices are located within a local network and so can have the same NameOfStation and IP address in all instances.

Additional references

Further information can be found in the *Basic Functions for Modular Machines Function Manual* and in the *online help for the TIA Portal*.

Configuring the project for Addresstailoring

Introduction

The Addresstailoring function ensures that all devices in the physical network are assigned a unique IP address and NameOfStation. This must be configured in the project for the IO controller and for the IO devices.

Procedure

The configuration of a series machine is described in the following using a SIMOTION D CPU as an example.

Configure a series machine project with Addresstailoring as follows

1. Create a project.
2. Configure the SIMOTION CPU as IO controller.
3. Configure the required IO devices (e.g. SINAMICS S120) and assign the IO devices to the PROFINET IO controller.
4. Configure the port interconnections between the devices.
5. Select the IO system so that you can edit the properties in the Inspector window.
6. Activate the "Reusable IO system" checkbox in the "General" area of the inspector window.

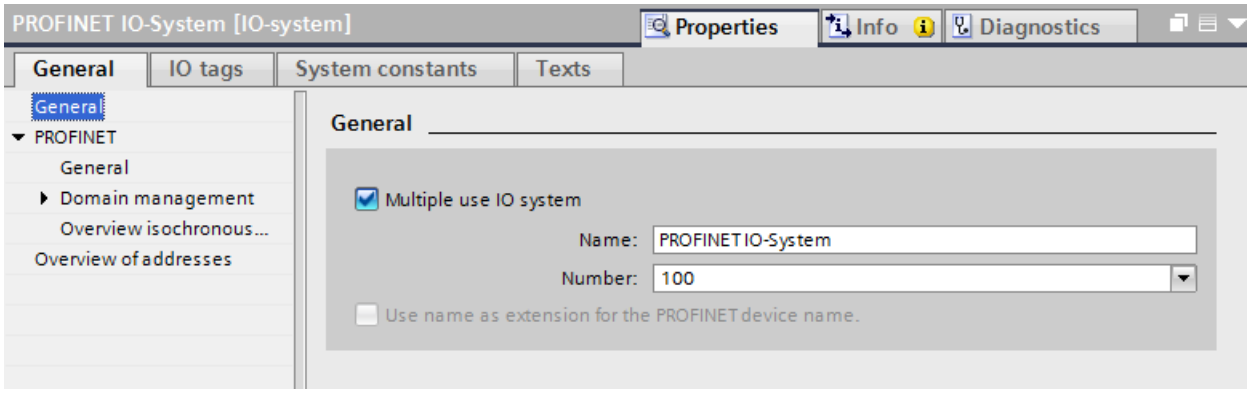


Figure 3-386 Configuring a reusable IO system (Addresstailoring)

Result in the TIA Portal

The following settings are made automatically through the configuration in the TIA Portal.

- The device name of the IO controller (CPU) in the series machine project is set to "Permit adaptation of the PROFINET device name directly on the device". At first, the IO controller (CPU) does not have a PROFINET device name.
- The IP protocol of the IO controller (CPU) is set to "Permit adaptation of the IP address directly on the device". At first, the CPU does not have an IP address.

- The "Enable device replacement without exchangeable medium" option is activated automatically. This option enables an automatic commissioning. A commissioning engineer does not have to assign device names on site to the IO devices. Based on the target topology and other settings, the IO controller assigns the device name and the IP address to the IO devices during startup.
- The device name of the IO devices is set to "Automatically generate PROFINET device name" (from the configured name of the IO device).
- The IP protocol for the IO devices is set to "Permit adaptation of the IP address by the IO controller". At first, the IO devices do not have an IP address. If an IO device is not a typical distributed I/O system (e.g. ET 200 system), but rather another device, e.g. an HMI device, then change the option to "Permit adaptation of the IP address directly on the device", see below.
- The device numbers for the IO devices are assigned automatically and used locally to make the IP address unique.

In order that the IO controller can adapt the device names later at the operator, the "Permit overwriting of the PROFINET device name" option must be activated (CPU parameters, property of the PROFINET interface, Ethernet addresses area). This option is deactivated per default.

Local commissioning

During the plant startup, the IP address and the NameOfStation are assigned to the IO devices. IP address and name are generated according to the following rules:

- IP address: Sum of the IP address of the IO controller and the station number of the IO device
- NameOfStation: <Configured name of the IO device from the series machine project>.<Name of the associated IO controller set on the device>

Diagnostics

If an error occurs during startup, e.g. incorrect IP address or NameOfStation is too long, an error message is entered in the diagnostic buffer.

Configuration control for IO systems - Machinetailoring

Introduction

The configuration control for IO systems (Machinetailoring) allows several specific instances or configurations of the series machine to be generated from a single series machine project. The configuration of an IO system can vary flexibly for a specific application provided the real configuration can be derived from the configured configuration. The configured configuration (series machine project) so forms the superset of all derivable real configurations.

You can activate or deactivate a PROFINET IO system (IO devices, I devices) in a specific system. Different variants can be operated from a configured maximum configuration of an IO system. You prepare for a series machine project a modular system consisting of IO devices that can be adapted flexibly for the wide range of configurations via the configuration control.

Machinetailoring allows the following customizations to be made on-site (without engineering system):

- Variation of the number of associated modules (IO devices). This is also possible with restrictions with IRT (IRT node as leaf for the IRT node part in a line).

Note

Machinetailoring can be configured only for SIMOTION SCOUT TIA.

PROFINET IRT

Machinetailoring is possible for PROFINET IRT only when minor adaptations need to be made to the IRT devices of the master machine. Minor adaptation means that IRT IO devices are only bridged or individual branches and leaves are removed.

Example of an automation solution

The following figure shows an example from a series machine project how an instance can be created with a different number of IO devices. The IO devices in the middle and at the end can be optional.

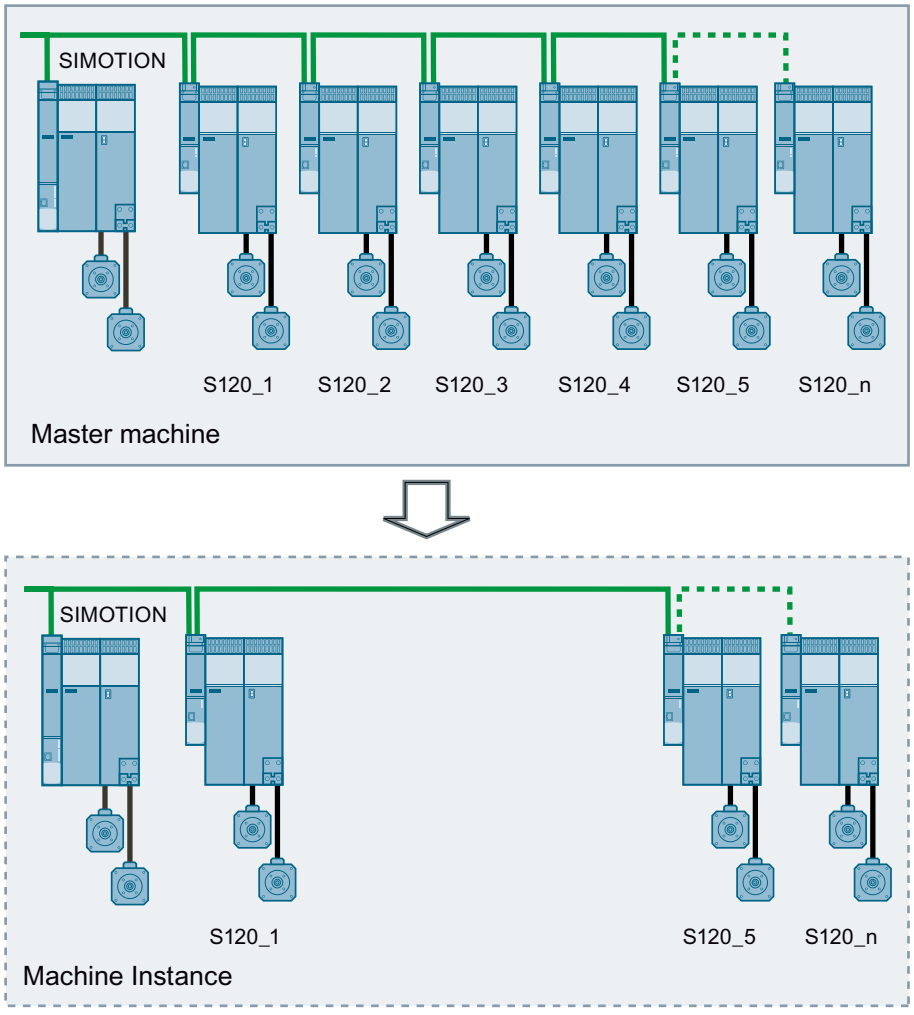


Figure 3-387 Instance of the machine with a different number of IO devices

Restrictions

- The number of optional IO devices in series is limited.
- The data exchange between IO controllers (controller-controller slave-to-slave communication) is possible only from the user program.
- Because it cannot be guaranteed that the optional IO devices actually exist in the machine instance, data exchange between IO devices is not meaningful.

Additional references

Further information can be found in the *Basic Functions for Modular Machines Function Manual* and in the *online help for the TIA Portal*.

Configuring with optional IO devices

Principle

In the project, configure the maximum configuration of the series machines. The supplied machines contain all or individual IO devices of the maximum configuration. The user program transfers a parameterized data set for configuring the supplied machines. After the reconfiguration, the optional IO devices are activated.

Requirement

You must observe the following for the configuration of the optional IO devices:

IO controller

- No special settings need to be made on the IO controller.
- The IO controller must be assigned a subnet.

IO devices

- The deployed interface modules have a PROFINET interface.
- The distributed peripheral stations are assigned the same subnet as the IO controller.
- The topology of the IO devices is defined.

Configuring the optional IO devices

For the configuration, settings must be made on the IO devices that should be optional.

1. Open the "Devices & networks" overview.
2. Double-click the IO device in the project navigator. The selected IO device is displayed in the device view.
3. Click the PROFINET interface of the IO device. Open the "Advanced options".
4. Activate the "Optional IO device" checkbox for "Interface options" in the Inspector window.
5. Repeat the steps for all optional IO devices.

SIMOTION controller as intelligent IO device (I device) on a higher-level IO controller

If you operate a SIMOTION controller as I device together with optional IO devices, you must take the following into account during the configuration of the SIMOTION I device. The parameterization for the PN interface of the SIMOTION I device must be performed as for a "normal" IO device by the higher-level controller.

Note

Parameterization by the higher-level controller

In order that the "Optional IO device" function can be activated, all IO devices and I devices in the PROFINET IO system must be parameterized by the higher-level IO controller, irrespective of whether they are used optionally. This means that "Parameter assignment of PN interface by higher-level PN controller" must be configured for all I devices.

To configure the SIMOTION I device:

1. Activate the following options for the operating mode of the I device.
   - IO device
   - Operate isochronously
   - Parameterization of the PN interface higher-level PN controller
2. Save and compile the hardware.
3. Export the SIMOTION I device.
4. You can now connect the SIMOTION I device as intelligent IO device to the higher-level controller.

SIMOTION variant management function block

The IO devices to be activated are specified with a data set. A function block transfers this to the IO controller, which activates the IO devices accordingly. The function block and the associated documentation (SIMOTION variant management) with the detailed procedure are available as download in the Service & Support Portal (https://support.industry.siemens.com/cs/ww/en/view/109747768).

Performance upgrade - low send cycle clocks

Optimized data transfer / 125 µs PROFINET send clock

PROFINET V2.3 Performance Upgrade

All SIMOTION Control Units support PROFINET in accordance with the PROFINET International specification PN V2.3. The optional **Performance Upgrade** functionality that provides extended functional scope is available for SIMOTION D455-2 DP/PN.

- Minimum PROFINET send cycle clock 125 µs (Fast Send Clock)
- Optimization of data transfer (e.g. by Dynamic Frame Packing - DFP (Page 740))

- Fast forwarding
- Fragmentation

Note

Validity

The PROFINET Performance Upgrade (Fast Send Clock 125 µs and Dynamics Frame Packing) is only approved for SIMOTION SCOUT TIA.

Note

You will find further information on this on the web site of the PI Organization (http://www.profibus.com) and the following technical specifications:

- Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.722
- Application Layer services for distributed I/O and distributed automation, Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.712

With optimized data transfer, the following options are possible:

- More devices can be operated with the same cycle time.
- The cycle time can be reduced for the same number of devices.

Requirements

General conditions for using the 125 µs send cycle:

- The 125 µs send cycle is supported only by the onboard PROFINET interface X150 in conjunction with Servo_fast / IPO_fast.
- Only one port can be used on X150; the other two ports must therefore be deactivated in the interface properties of X150 (the ports cannot be used for TCP/IP or UDP communication either).
- X150 can only be used as a PROFINET IO controller (not as an I-Device).
- No MRP/MRPD media redundancy (only one port is available on X150).
- Use of STEP 7 V14, V14; for other operating conditions, see also *Readme* for SIMOTION SCOUT V4.5.

Note

Quantity structures in the case of very short send cycles

When very short send cycles (125 µs, 250 µs) are used in a configuration with a large number of PROFINET devices, SIMOTION will be operating close to its upper performance limits. It is not possible to reliably predict all the restrictions applicable to the configuring of such systems. Only during runtime will it be possible to identify overload situations. Overload conditions will be indicated by appropriate messages (cycle violation, inconsistent data transmission). When you are using these very short cycle times, you should therefore restrict the number of devices to the absolute minimum necessary.

Components that can be used

The 125 µs send cycle is only supported by selected PROFINET nodes (e.g. by the ET 200SP I/O system). No SCALANCE switches are currently available for the 125 µs send cycle.

- SIMOTION D455-2 DP/PN as of V4.5

Note

For the PROFINET nodes that support a send cycle of 125 µs and the general conditions that have to be met, see the relevant product documentation.

Dynamic Frame Packing - DFP

Description

For PROFINET V2.2, a separate Ethernet frame is used for each IO device for the cyclic transfer of the IO data. This leads to a large overhead due to the Ethernet header, especially in the case of IO devices with a small amount of IO data.

With the PN V2.3 Performance Upgrade, SIMOTION D455-2 DP/PN supports Dynamic Frame Packing (DFP) via the onboard PROFINET interface X150. With Dynamic Frame Packing, the cyclic data of multiple IO devices is summarized in one Ethernet frame. DFP significantly reduces the bandwidth required to exchange the cyclic data, especially in the case of IO devices with a small amount of IO data. This makes it possible to either operate more IO devices with the same cycle time or to reduce the cycle time in order to achieve a shorter response time.

In the outbound direction (from the IO controller to the last IO device), the Ethernet frame is automatically reduced. This is achieved by having each device remove its own data from the telegram when forwarding. In the inbound direction (from the last IO device to the IO controller), each IO device adds its own data to the Ethernet frame.

Dynamic Frame Packing is used automatically when send cycles < 250 µs are set and the requirements (see "Requirements" section) are met.

The figure below shows examples of how package groups can be formed. The IO devices (IO-D) with red backgrounds are automatically grouped into package groups.

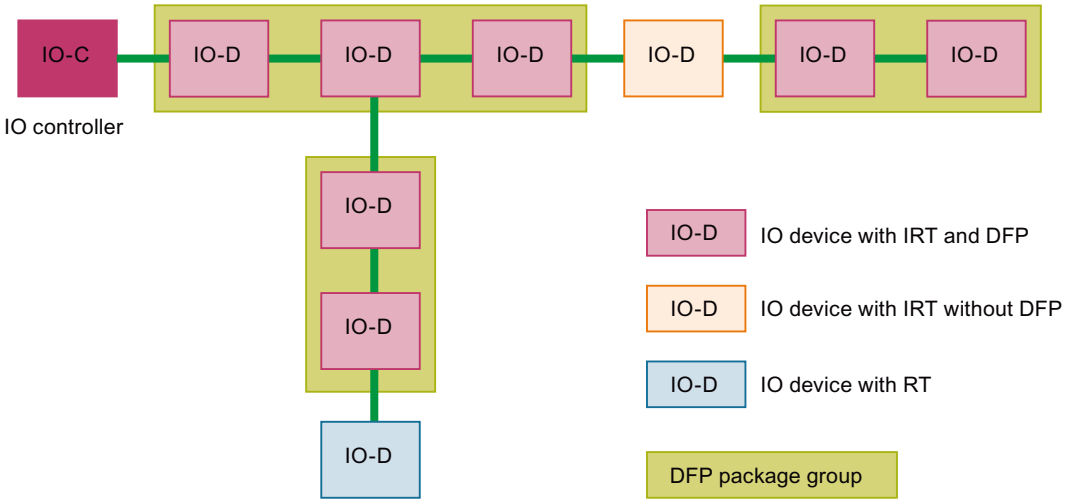


Figure 3-388 Example of DFP package group formation

Requirements

DFP can only be used with the following restrictions:

- DFP can be used only on the onboard PROFINET interface X150 of the SIMOTION D455-2 DP/PN and on the ET200SP High Speed IO-Devices.
- DFP is supported by SIMOTION only as an IO controller. The SIMOTION I-Device does not support DFP.
- A DFP device can be used as a shared device by a second IO controller. Whereby the shared device restrictions apply. One IO controller communicates with DFP via IRT; the other via RT.
- In the case of MRPD, the DFP devices must be present in a spur line. IO devices in the ring are not supported.
- IO devices that do not support DFP terminate a package group; i.e. the grouped IO devices must form a line with no gaps.
- A maximum of 63 IO devices can be packed in one group.
- The size of a package frame is limited to 128 bytes in the outbound direction and 256 bytes in the inbound direction.

Configuring IRT with Performance Upgrade

Introduction

To use PN V2.3 Performance Upgrade on the SIMOTION CPU with SIMOTION SCOUT TIA, you must activate this option on the sync domain.

Activating High Performance and increasing the bandwidth

To use send cycles < 250 µs and other Performance Upgrade functions, proceed as follows:

1. Deactivate all ports (except for port 1) of the SIMOTION D455-2 DP/PN.
2. Select the PROFINET IO system in the network view.
3. In the Inspector window, select the "Properties > General" tab.
4. Select "Domain-Management" in the navigation and then the appropriate sync domain.
5. Select the send cycle 0.125 ms.
6. Activate "Permit High Performance" and "Fast Forwarding" optionally.

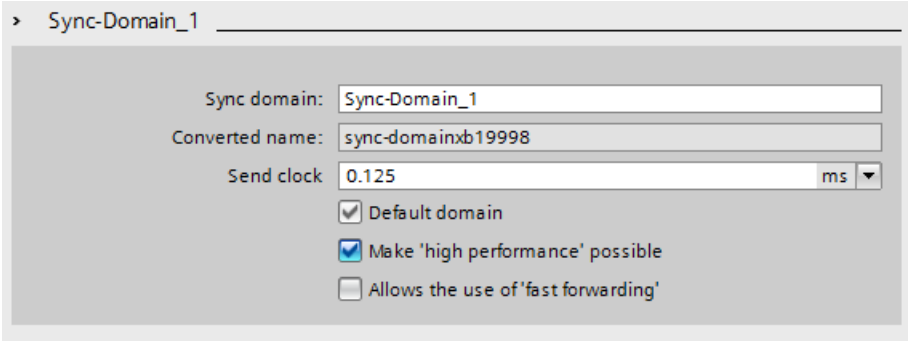


Figure 3-389 Activating High Performance for SIMOTION

7. Click "Details" in the navigation below the sync domain.
8. Select "Maximum 90% cyclical IO data..." for the bandwidth use.

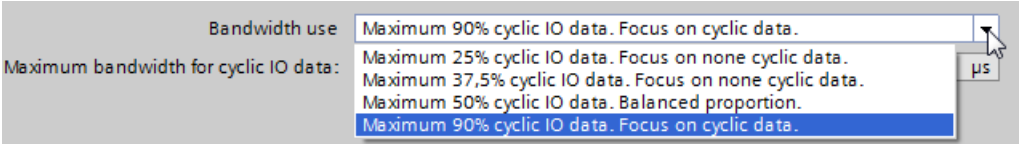


Figure 3-390 Increasing the bandwidth for cyclical IO data

3.5.6.3 Using PROFIsafe

PROFIsafe for SIMOTION

Description

The drive-based safety functions in the drive can be controlled either by using safe terminals directly on the drive or from a fail-safe control (F control) via PROFIBUS/PROFINET.

The control signals for the drive-based safety functions, as well as the feedback relating to the safety function status, are safety-oriented and must be transferred via a communication channel that is secured by means of the PROFIsafe protocol.

Mechanisms with PROFIsafe

Basically, the PROFIsafe communication can be performed via PROFINET and also via PROFIBUS. The following mechanisms are available for PROFINET and PROFIBUS:

- PROFINET I device F-proxy
  - I device directly on the F-host (a project for F-CPU and SIMOTION CPU)
  - I device as GSDML file coupled to F-host (two projects)
  - Shared I device (like I device as GSDML file, the I device is also "shared")
- PROFINET Shared Device
- PROFIBUS slave-to-slave communication
  - I slave to slave

For all the mechanisms, the fail-safe data is managed by the F-CPU and the motion control data by the SIMOTION controller. The required properties of the PROFIsafe communication are set in the F-parameters during the hardware configuration.

Note

Mixed operation in a PROFIsafe configuration is **not** possible with **one** Control Unit. If, for example, in the case of a SINAMICS drive, the safety configuration for one axis is carried out via F-Proxy and those of the other axes are configured via Shared Device, this configuration will be rejected with a consistency error during compilation. The F-Proxy variants are recommended for the PROFIsafe configuration.

Supported number of F-Proxy submodules

A maximum of 128 I device submodules are supported. Of these, up to 64 submodules can be used for Safety by default. The other 64 submodules are available for standard IOs. The maximum number of 64 F-Proxy submodules can be configured **independently** of the PROFIsafe telegram for all SIMOTION CPUs.

Up to 128 F-Proxy submodules supported (as of SIMOTION V4.5)

As of firmware version 4.5, a maximum of 128 F-Proxy submodules, and thus a maximum of 128 safety axes, can be configured for the SIMOTION D455-2. The maximum number of F-Proxy submodules **depends** on the configured PROFIsafe telegram.

In the following tables, the input/output data, as well as the maximum number of F-Proxy submodules, are shown depending on the PROFIsafe telegram and the SIMOTION CPU.

SINAMICS PROFIsafe telegrams

PROFIsafe telegram	F-Proxy submodules SIMOTION D455-2	F-Proxy submodules SIMOTION CPU (other)	Single telegram Input	Single telegram Output	Single telegram Total
30	128	64	6	6	12
31	128	64	8	8	16

PROFIsafe telegram	F-Proxy submodules SIMOTION D455-2	F-Proxy submodules SIMOTION CPU (other)	Single telegram Input	Single telegram Output	Single telegram Total
901	89	64	14	10	24
902	79	64	16	10	26
903	89	64	14	10	24

Configuring more than 64 F-Proxy submodules

The following must be taken into account when configuring more than 64 F-Proxy submodules:

- A SIMOTION I device supports a maximum of 128 subslots. If, for example, 128 subslots are assigned to F-Proxy axes, no further user data can be configured on the I device interface (see table above).
- A SIMOTION I device supports a maximum of 1440 bytes of user data (including IOPS/CS user data qualifier). Depending on the selected safety telegram and the number of F-Proxy axes, only limited (or no) space is available for user data in the I device.

F proxy with SIMOTION

F-Proxy description

SIMOTION features integrated F-Proxy functionality for the purpose of PROFIsafe connection of drives that are controlled by SIMOTION but are in a different communication domain than the F-CPU, for example. The F-Proxy functionality enables transparent routing of safety telegrams from the SIMOTION I-Device interface to the respective SIMOTION controller interface on which the drive is configured. Despite the SIMOTION routing function, PROFIsafe communication between the F-CPU and drive is secure, as the PROFIsafe drivers at the end points (F-CPU, drive) securely monitor communication.

In order to use F-Proxy functionality, the two paths of communication - from the F-CPU to SIMOTION and from SIMOTION to the drive - need to be configured separately.

The following drive connections can be used on the SIMOTION controller for the F-Proxy:

- Drive on SINAMICS_Integrated (SIMOTION D)
- Drive on PROFIBUS DP
- Drive on PROFINET IO Onboard PN interface
- Drive on PROFINET IO CBE30-2 PN interface

Properties of the I-Device F-Proxy

- The F-Proxy submodules on the SIMOTION controller are RT and can be operated non-isochronously.
- The PROFIsafe standard telegrams 30 and 31 and the PROFIsafe SIEMENS telegrams 901, 902 and 903 are supported.

Additional references

Detailed information about the F-Proxy can also be found in the *SIMOTION Communication* Function Manual.

Note

Connection between SIMOTION and F-CPU only via PROFINET IO

With the TIA Portal, you can interconnect F-Proxy slots between the SIMOTION controllers and the F-CPU only via PROFINET IO. A so-called PROFIBUS I slave F-Proxy is not supported.

Basic procedure for an F-Proxy configuration

The F-Proxy configuration is performed in the TIA Portal and in SIMOTION SCOUT TIA:

- Creating the device (F-CPU, SIMOTION controller, drives) in the TIA Portal
- Networking the devices and creating the topology in the TIA Portal
- Creating and configuring the drives, axes, and configuration of PROFIsafe in the drive in SIMOTION SCOUT TIA
- Changing the operating mode of the PN interface of the SIMOTION controller to IO device (I-Device) in the TIA Portal
- Interconnecting the F-CPU with the drive units via the PN interface that functions as I-Device in the TIA Portal

A configuration example (Page 746) is described in detail in the next section.

F-Proxy settings on the SIMOTION controller

If you want to run a SIMOTION controller as F-Proxy, you must perform the following steps:

1. Change the operating mode for the SIMOTION device to IO device and select whether the F-CPU is in the same or in another project.
2. In the Inspector window below "General", select the "F-Proxy" entry.
3. Configure the interconnections of the F-Proxy.

See also

Direct data exchange via PROFIBUS DP (Page 789)

Direct data exchange (Page 716)

Checking project-wide PROFIsafe addresses (Page 757)

Configuration example for I device F-Proxy

Introduction

This section describes a configuration example for an I-Device F-Proxy. The configuration is largely identical for all drives at the SIMOTION controller. The difference is essentially whether the F-CPU is in the same or in another project. As an example, the configuration is shown in the same project.

Controllers and drives used

- SIMOTION D455-2 DP/PN (Motion Control) [PLC_1]
- SINAMICS S120 CU320-2 PN
- SIMATIC S7-300 CPU317F-2 PN/DP (F-CPU) [PLC_2]

Requirement

- Project created in the TIA Portal.
- Controllers and drives inserted.
- PROFINET IO system configured with SIMOTION D455-2 DP/PN and SINAMICS S120 CU320-2 PN.
- Ports are interconnected.
- Drive and the axis are configured in SIMOTION SCOUT TIA.

Also note Section Brief introduction to SIMOTION control and SINAMICS drive via PROFINET IO (Page 684).

Configuring PROFIsafe in SIMOTION SCOUT TIA

1. In the project navigator in SIMOTION SCOUT TIA below the drive, click "Functions > Safety Integrated".
   The window for safety configuration opens in the working area.
2. Select a PROFIsafe function such as "Extended functions via PROFIsafe".
   You can view the "PROFIsafe address" in the window "Configuration PROFIsafe". 0001H is used in this example.

   Note

   The PROFIsafe address for the safety configuration is the F-destination address for the later configuration of the F-Proxy. These **must** be identical. You enter the F-destination address in the interconnection table when interconnecting the F-Proxy.

   The F-destination address must be configured at three places and must be identical:

   - in the drive (PROFIsafe-address)
     As of V5.2, you can apply in the drive the PROFIsafe address (F-destination address) configured in the TIA Portal.
   - at the F-Proxy interconnection
   - at the exported I-Device if the F-CPU is in another project

   Observe the procedure for checking PROFIsafe addresses in the project in section Checking project-wide PROFIsafe addresses (Page 757).
3. Activate Safety Integrated for the axis. Double-click "Configuration" below the axis in the project navigator. The window opens in the working area.
4. In the "Functions" field, click the "Change..." button and select the "Standard (support of the entire safety functionality via DSDB)" option in the "SINAMICS Safety Integrated" tab.
5. After configuration, you have to select the PROFIsafe telegram. In the SIMOTION SCOUT TIA project navigator, below the drive unit double-click "<Drive unit_xx"> - Communication > Telegram configuration".
   The telegram configuration is opened in the working area.
6. Mark the appropriate drive in the tab "IF1: PROFIdrive PZD telegram" in the telegram overview, and in the bottom part of the window at "Adapt telegram configuration", select the entry "Add PROFIsafe".
   The PROFIsafe telegram is inserted. Dependent on the drive, you can select from several telegrams. The PROFIsafe standard telegram 30 is inserted by standard. You can change this.

Figure 3-391 Adding the PROFIsafe telegram on the drive

7. Save and compile the project.

8. Click "Set up address" to run an address alignment between SIMOTION SCOUT TIA and TIA Portal. A successful alignment is indicated with a blue checkmark. For the drive, telegrams are indicated with red checkmarks, since during the alignment the process data was extended due to the automatic telegram extension. This is for information purposes only. Configuration in SIMOTION SCOUT TIA is completed by saving.

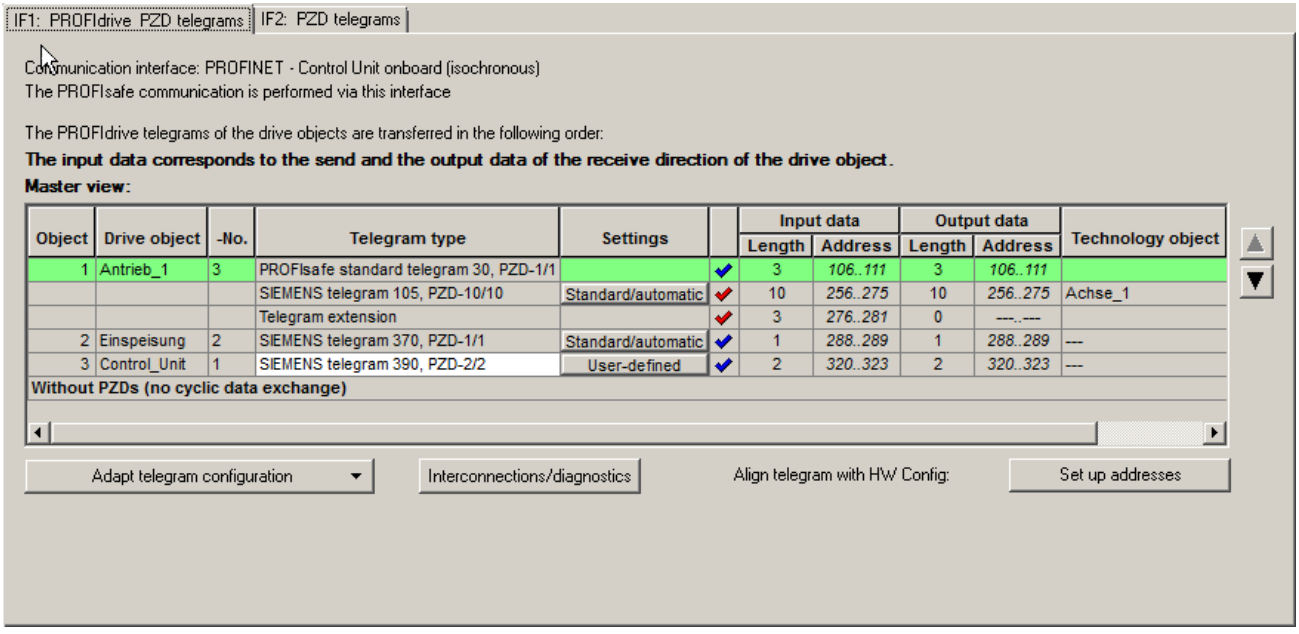


Figure 3-392 Telegram configuration for drive with PROFIsafe

9. Switch to the TIA Portal and create the I-Device.

Creating an I-Device in the TIA Portal

1. In the network view, select the SIMOTION controller and the interface via which the I-Device is to be operated. In the example, X150.
2. In the Inspector window, select "Properties > General" and click "Operating mode".
3. Activate the "IO device" option.

4. Assign the controller to which the I-Device is to be assigned at "Assigned IO controller". In this case, the F-CPU. A subnet is automatically inserted between the F-CPU and the SIMOTION controller.

   Note

   From that point on, the configuration is different if the F-CPU is in another project. "Not assigned" is selected as IO controller and the transfer area is defined and the F-Proxy is configured. A GSD file of the I-Device is then created.

5. Define the transfer area.
   You configure the F-Proxy in the next step.

   Note

   Transfer areas

   In the transfer area, the automatically created fail-safe I/O transfer areas are **not** displayed.

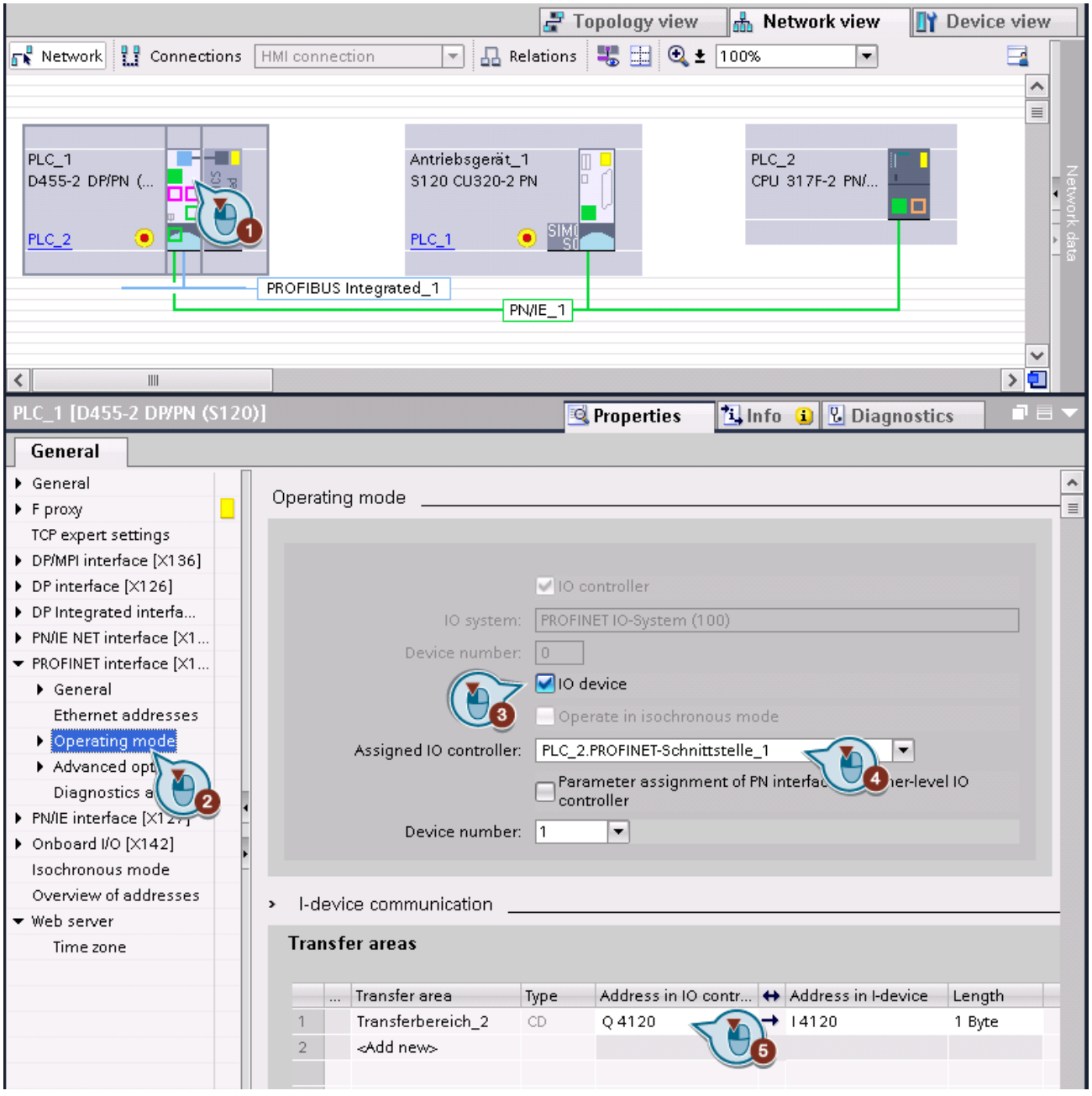


Figure 3-393 Creating an I-Device for I-Device F-Proxy

Configuring F-Proxy in the TIA Portal

1. Click the SIMOTION controller in the network view.
2. Click "F-Proxy" on the "General" tab. The F-Proxy interconnection is opened.
3. In the column "F-CPU", select the IO controller for PROFIsafe monitoring. All controllers in the project are displayed. Choose "PLC_2.PROFINET_Interface_1" in the example.

4. If the F-CPU is already interconnected with the SIMOTION controller, the PN interface used is automatically entered into the column "F-Proxy interface". If there is no interconnection yet, you can select the PN interface of the SIMOTION controller here.
5. An address is specified in the "F-destination address" column. This address must be identical with the PROFIsafe address of the drive in SIMOTION SCOUT TIA. Check if these addresses are identical and change them. Address 0001H is used in this example. Click the field to change the address. This will finish the configuration.
   More information about the interconnection table can be found in Section Editing F-Proxy settings (Page 753).

Note

The F-destination address must be assigned for each drive. If you have configured further drives at the SIMOTION controller, they are shown in the table and are configured identically. You must also enter the specific F-destination addresses of the safety configuration here for each drive.

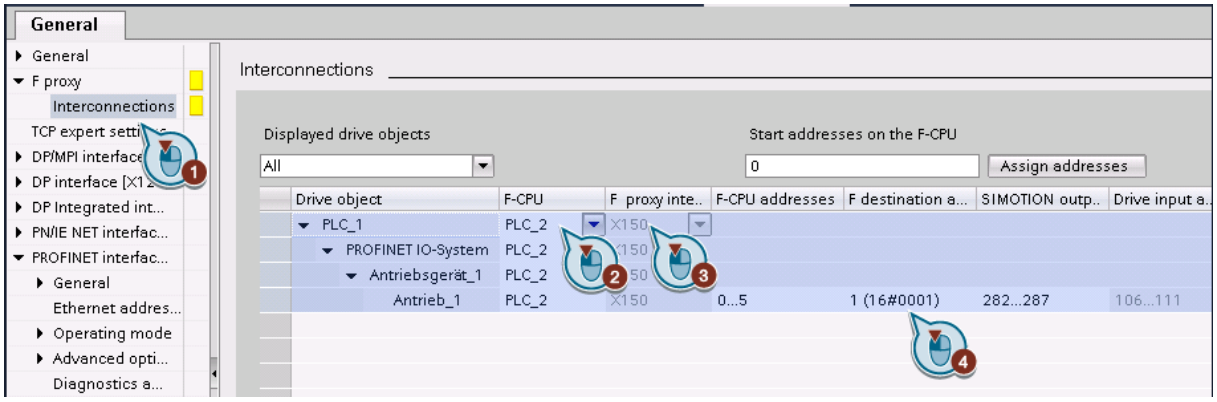


Figure 3-394 Configuring interconnections for F-Proxy

F-CPU in another project

If the F-CPU exists in another project, you must pay attention to some points.

1. When configuring the I-Device, select the entry "Not assigned" at "Assigned IO controller" and then configure the I-Device as described above.
2. Configure the interconnection of the F-Proxy. For the F-CPU, select "F-CPU in another project" and then configure the further parameters of the F-Proxy as described above.

3. Change back to the "Operating Mode" window. Now export the I-Device as device description file (GSD). The procedure is described in Section Configuring an I-Device (Page 709).

Note

Configuring an F-Proxy prior to the I-Device export

Before you create the I-Device, you must have finished the F-Proxy interconnections. If changes are made to the F-Proxy interconnection (e.g. other F-destination addresses), you must export the I-Device again.

4. You may insert the installed I-Device with a drag-and-drop operation into the other project with the F-CPU and connect it to the subnet of the F-CPU.

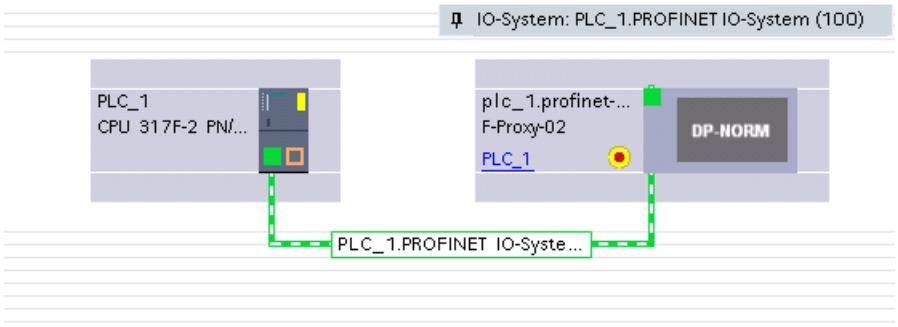


Figure 3-395 F-CPU in another project with I-Device F-Proxy on the subnet

Automatic transfer of the F-destination addresses

Accept F-destination address (V5.2 and higher)

All axes are configured and commissioned during initial commissioning of a machine. Next the PROFIsafe telegrams are added in the telegram configuration of SIMOTION SCOUT or Starter. As soon as the telegram is used for the F-Proxy it can no longer be changed. The logical address is specified during "Save and compile" with symbolic assignment.

After the slot for the PROFIsafe telegram has been accepted in HW Config, the PROFIsafe telegram also receives its F-destination address. Up to V5.2, this address had to be entered manually for all drives with PROFIsafe while commissioning Safety Integrated. To avoid incorrect entries, the "Accept F-destination address" button is available during Safety commissioning as of V5.2 and higher.

F-destination address during Safety Integrated commissioning

You apply the F-destination address for PROFIsafe during drive commissioning.

1. In the SIMOTION SCOUT project navigator, click on "Functions > Safety Integrated" below the drive. The Safety configuration window opens in the work area. PROFIsafe is configured as Safety function.
2. Click the "PROFIsafe configuration" button.
3. You can view the "F_Dest_add" in the "PROFIsafe configuration" window.

4. Click "<" to accept the F-destination address as PROFIsafe address.

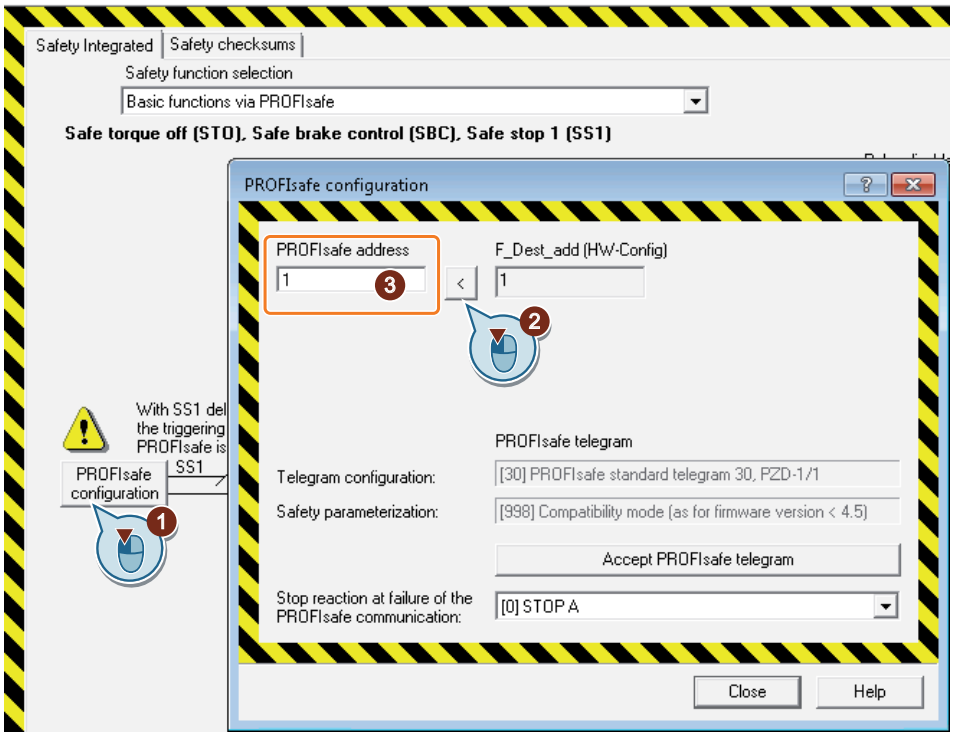


Figure 3-396 Accepting the F-destination address as PROFIsafe address

5. Click "Close" to confirm.

Scripting

The "Accept F-destination address" function can also be programmed via scripting. To do so, program the following scripting commands.

- GetFDestAdressFromHWCN
- SetUpAddresses

Editing F-Proxy settings

Working with the F-Proxy interconnection table

With an interconnection table, you interconnect all drives (also termed drive objects or DOs) with a F-CPU. The drives must be connected to the SIMOTION controller and a PROFIsafe telegram must be set up.

The table is hierarchically organized. If you change a setting (F-CPU, F-Proxy interface) on a higher-level entry, the new setting will be propagated to all lower-level objects.

- To edit the settings below a SIMOTION controller, edit the settings at the controller level to interconnect all the interfaces and drive objects below it.
- If you only want to interconnect the drive objects of one particular bus system, edit the relevant entry of the interface to interconnect all the drive objects below it.
- If you only want to configure a particular drive object, change the settings on this drive object only.

In the first example figure, for example, the F-CPU with the name PLC_2 is used for all drive objects.

In the following figures, you can see an example of an F-proxy interconnection with an F-CPU in the same project and an F-CPU in another project. You will find the explanations of the numerals in the table. In the interconnection table, you can edit only the field with a light-blue background.

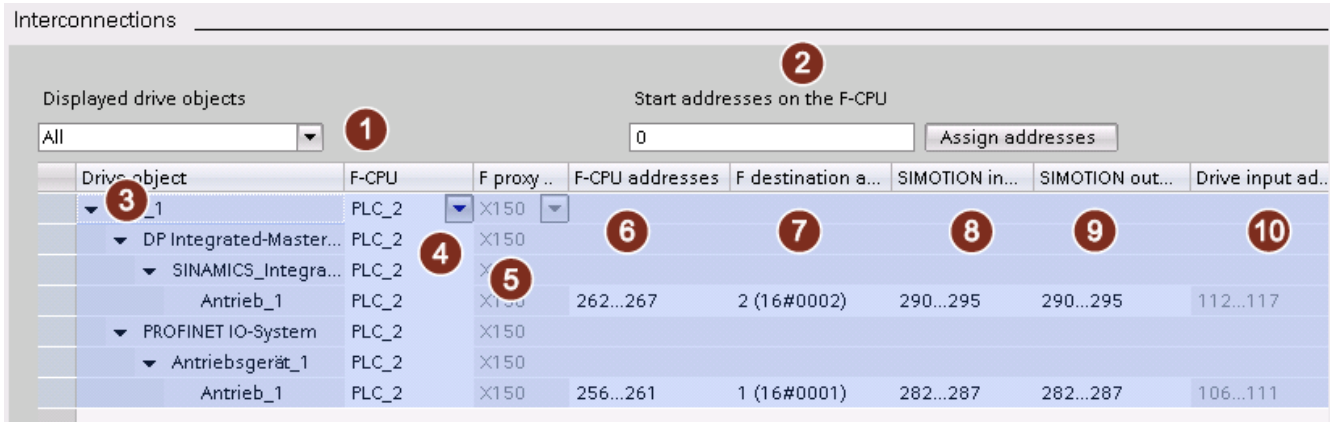


Figure 3-397 F-proxy interconnection table with two drive objects and F-CPU in the same project

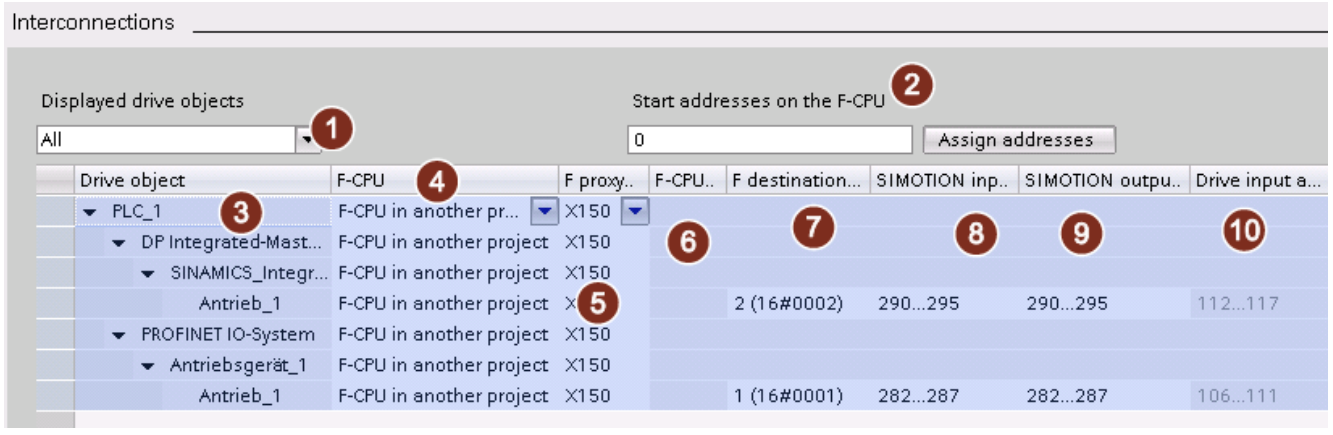


Figure 3-398 F-proxy interconnection table with two drive objects and F-CPU in another project

Explanations of the interconnection table

Parameter	Description
Displayed drive objects ①	At "Displayed drive objects", you can filter the display of the drive objects. You can have only the interconnected or only the non-interconnected objects displayed.
Start address in the F-CPU ②	Enter a start address as the basis for address assignment in the F-CPU here. If no drive object has yet been interconnected with the F-CPU, this button is not active.
Drive object ③ (not editable)	The drive objects are displayed sorted hierarchically below the controller and interface.
F-CPU ④	Select the F-CPU with which interconnection of the SIMOTION controller is to be performed via the F-proxy. • All F-CPUs connected to the SIMOTION controller are listed (F-CPU in the same project) In the example, the SIMOTION controller (PLC_1) is interconnected to the F-CPU (PLC_2). • F-CPU in another project • None With this selection, existing interconnections will be deleted.
F-proxy interface ⑤	Here, you can select the PN interface of the SIMOTION controller via which interconnection with the F-CPU is performed. All PN interfaces that are in I-Device mode are displayed (X150, X1400). In the example, it is the X150 interface. If the F-CPU is in the same project and is already interconnected with the SIMOTION controller, the interface will be entered automatically.
F-CPU addresses ⑥	Address of the drive objects in the IO controller of the F-CPU. On the first drive object, the start address is used that was entered at "Start address in the F-CPU".
F-destination address ⑦	You enter the F-destination address here. This is preset to a default value. The F-destination address is identical to the PROFIsafe address in the drive.
Input address ⑧	Enter the input address of the I-Device here (SIMOTION controller).
Output address ⑨	Enter the output address of the I-Device here (SIMOTION controller).
Drive input address ⑩ (not editable)	The logical address of the PROFIsafe telegram of the drive object is displayed here. You can change this address in SIMOTION SCOUT TIA.

Configuring a table

When you right-click the table header, a shortcut menu opens.

1. Select "Display/hide" if you wish to configure the displayed columns. Please note that some of the columns including, for example, F-CPU address are required for programming the F-CPU application.
2. "Show all columns" if you require the entire table.

F-CPU in another project

If the fail-safe module is located in another project, for example, after migrating a project, select the entry "F-CPU in another project" for the F-CPU.

For this setting, at least one PN interface of the SIMOTION controller must be configured as an I-Device and no F-CPU must be present in the project. When you have made all the parameter settings, you must export the I-Device with a GSD export and then install it and reinsert and interconnect it in the F-CPU.

Filtering the display

With the "Displayed drive objects" filter, you can filter displayed drive objects, for example, for the following application scenarios:

- You want to interconnect additional drive objects subsequently.
- You have a project with a large number of drive objects.
- For checking whether all drive objects have been interconnected.

The entries that do not correspond to the filter criterion are hidden.

Start address of the F-CPU

The start address defines an address range in the F-CPU. All components are then clearly listed within the address band.

1. Enter a start address as the basis for address assignment in the F-CPU here.
   If no drive object has yet been interconnected with the F-CPU, this button is not active.
2. First enter the start address.
3. The make the interconnection.
   When you then interconnect the object, the addresses will be assigned from the starting value in ascending order.

If you want to change the address range after having made an interconnection, enter a new start address and click "Assign address".
The addresses are automatically moved into the address range.

Deleting components

If you delete an F-CPU, for example, all interconnections and transfer areas will be deleted.

Changing CRC backed-up data of a PROFIsafe telegram

Improved configuring/commissioning of drives with safety technology

For plants and machines with comprehensive safety technology, it must be possible to detect changes to safety-related data quickly and precisely. Each safety element usually has safety features, e.g. CRC (Cyclic Redundancy Check) backed-up data. For machines with 128 safety axes, for example, large data storage areas such as F-parameters are protected for the PROFIsafe telegrams. This means changed data must be displayed in a clearly structured manner. As of V5.2 and higher, the F-Proxy Interconnections table is extended by the **CRC change** column.

CRC change in the F-Proxy Interconnections table

Each change of CRC backed-up data of a PROFIsafe telegram is recorded. When the signature of the F-parameters (Failsafe_F_Par_CRC/F-parameter signature with addresses) is changed, this is indicated visually for the F-Proxy in the Interconnections table. If there is no CRC change, the column remains empty. CRC changes are displayed by a symbol for the respective safety axis. Check the respective F-parameters of the safety axis when a change is detected.

The detected difference only serves as information. The generation of online data is not prevented.

Working with the CRC change

1. Save and compile the modified project. A check is performed during compiling to see whether a CRC change exists.
2. If changes are detected, the comment "Unconfirmed CRC changes present" is displayed in the "Info > Compile" detailed view.
3. Click on the "Go to" column in the information row to jump to the Interconnections table with the detected CRC changes.
4. Check the displayed changes.

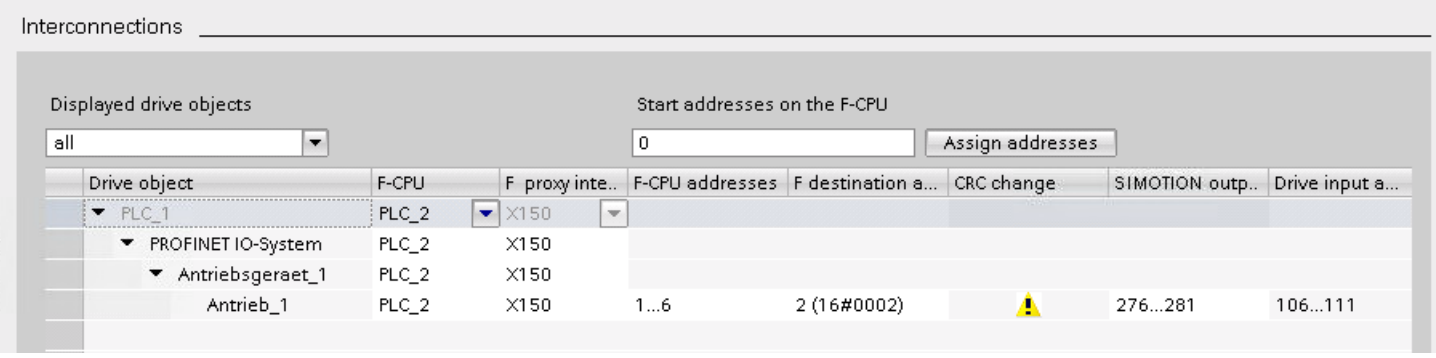


Figure 3-399 CRC change detected at safety axes

5. By using the filter selection "With CRC changes" for "Displayed drive objects", only the rows with changed CRC data are displayed.
6. At the bottom of the Interconnections table, click on the "Confirm CRC changes" button. The markings at the changed safety axes are deleted.

Checking project-wide PROFIsafe addresses

Unique PROFIsafe address

To ensure reliable communication, unique PROFIsafe addresses are required throughout the CPU and the network.

For this reason, you need to carefully check the settings of the PROFIsafe addresses.

⚠ WARNING

Unique PROFIsafe addresses

You must ensure the unique assignment of the PROFIsafe address throughout the network and the CPU.

- The fail-safe I/O of PROFIsafe address type 1 is uniquely addressed by its F-destination address.
- The F-destination address of the fail-safe I/O (drive units in this case) must be unique for all the fail-safe I/Os throughout the network and the CPU (system-wide). The fail-safe I/O of PROFIsafe address type 2, e.g. modules of the ET 200SP type, must also be taken into account.
- Note also the corresponding documentation in the TIA Portal online help in section "SIMATIC Safety - Configuration and programming".
  (*SDR001*)

1. Make sure that the following parameters are set for each drive:
   – Parameter p9610 in the parameter view = "F address" (F_Dest_Add) in the Inspector window under "Cyclic data exchange".
   Or
   – Parameter p9610 in the function view in dialog "F-DI / F-DO PROFIsafe" = F address (F_Dest_Add) in the Inspector window under "Cyclic data exchange".
2. Check the messages in the Inspector window while the control safety program is being compiled (compile CPU in the project tree). An error will be displayed for any address that is not unique:

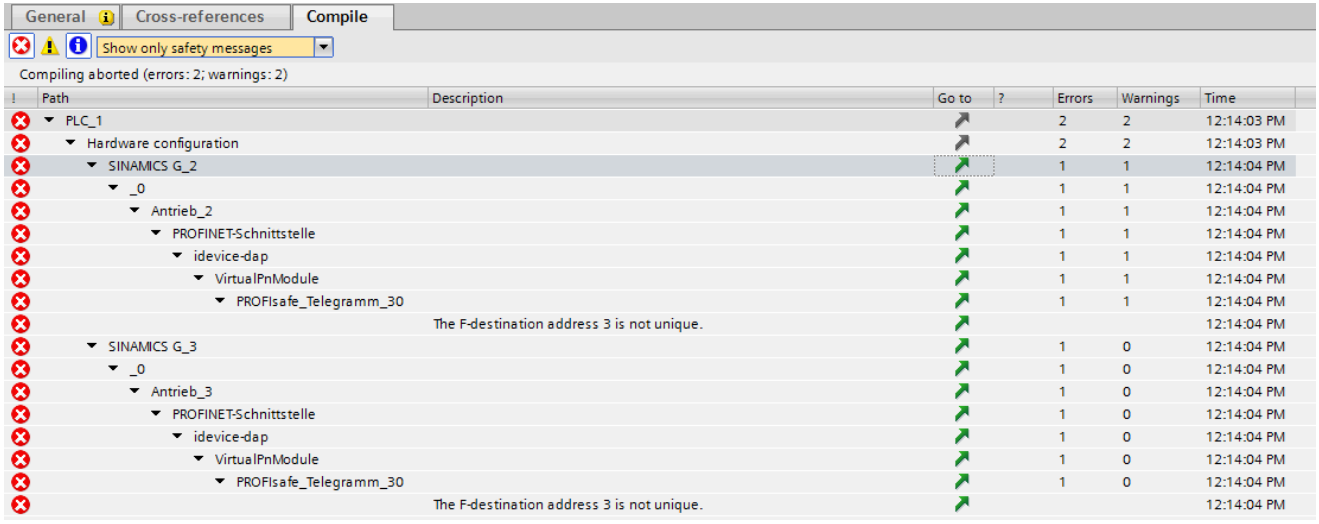


Figure 3-400 Compiler error message "F destination address is not unique"

Navigate into the telegram configuring screen of the affected device and assign a different PROFIsafe address.

3. Check the safety printout of the PLC you are using.
   Create the printout by right clicking on "Safety Administration" to call the context menu and then select "Print".
   All of the relevant data is listed in the printout. Below you can see an example for a drive that is networked as an IO device; the PROFIsafe address is displayed under "F-destination" address.

Note

Error-free hardware and software compilation

Error-free hardware and software compilation is the prerequisite for the creation of the safety printout for acceptance purposes. Only then is it assured that all consistency checks have been performed and therefore the safety printout created for a consistent project.

PROFINET-Schnittstelle : PROFINET IO-System PN/IE_1 idevice-dap					
Slot	Module	Start address	F-destination address	F-monitoring time	Parameter signature (w/o addresses)
0	PROFIsafe_Telegramm_30	8	1101	50 ms	0xEE5 (3813)

PROFIsafe_Telegramm_30 : idevice-dap, PositionNumber 0		
Module data		Channel parameters
Hardware		Info
Name	PROFIsafe_Telegramm_30	Module has no channels.
PositionNumber	0	
TypeName	Virtual.F.PnSubmodule.Drive	
--	--	
Start address input	8	
Start address output	8	
Laddr	276	
F_WD_Time	150 ms	
F_Dest_Add	1101	
F_Par_CRC_WithoutAddresses	0xEE5 (3813)	
F_Par_CRC	0xD884 (56244)	
Behavior after channel error	Passivate the complete module	
RIO for FA-Safety	No	
F_Par_Version	V2-mode	
PROFIsafe protocol version	Loop-back extension (LP)	
Software		
F-I/O DB number	30002	
F I/O DB name	F00008_SINAMICSG_1	
Used in F-runtime group	No	

Figure 3-401 PROFIsafe address in the safety printout

Compare this PROFIsafe address with the value of the drive in p9610.
Compare this PROFIsafe address with the values of all other nodes and make sure that the addresses are unique.

Typical PROFIsafe topologies

Overview of topologies

This topic shows examples of topologies in which SIMOTION PROFIsafe is used.

1. F-CPU in another project (GSDML export)

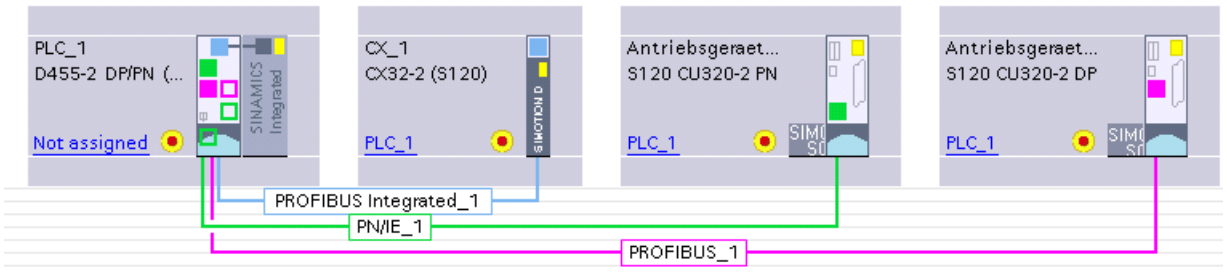


Figure 3-402 F-CPU in another project

2. F-CPU with SIMOTION via PROFINET as an I-Device (F-Proxy)

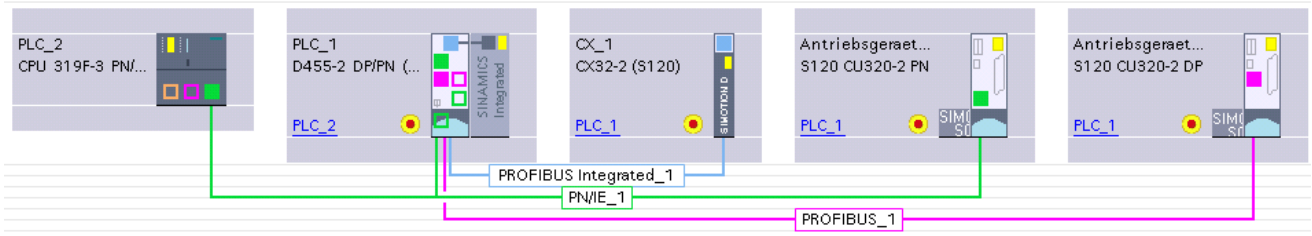


Figure 3-403 F-CPU with SIMOTION via PROFINET as an I-Device

Check of F-destination addresses

3. F-CPU with SIMOTION via PROFINET as an IO device (GSDML import)

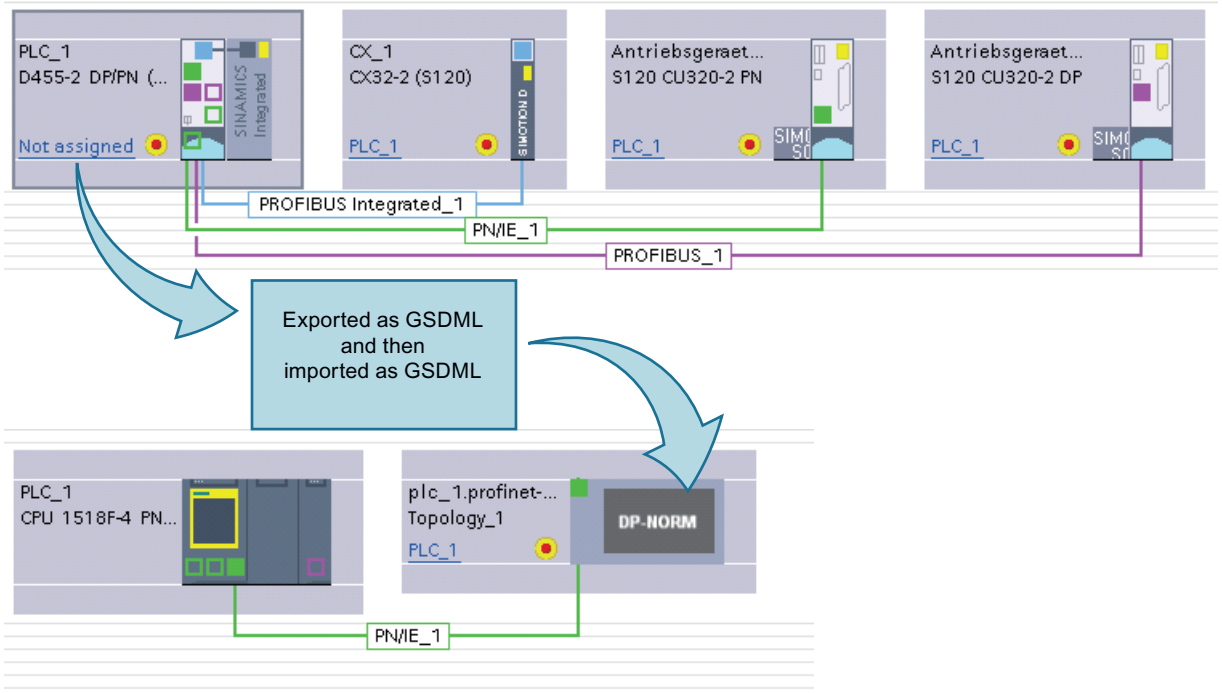


Figure 3-404 F-CPU with SIMOTION via PROFINET as an IO device

4. F-CPU with SINAMICS via PROFIBUS as a DP slave (F-DX-Mod)

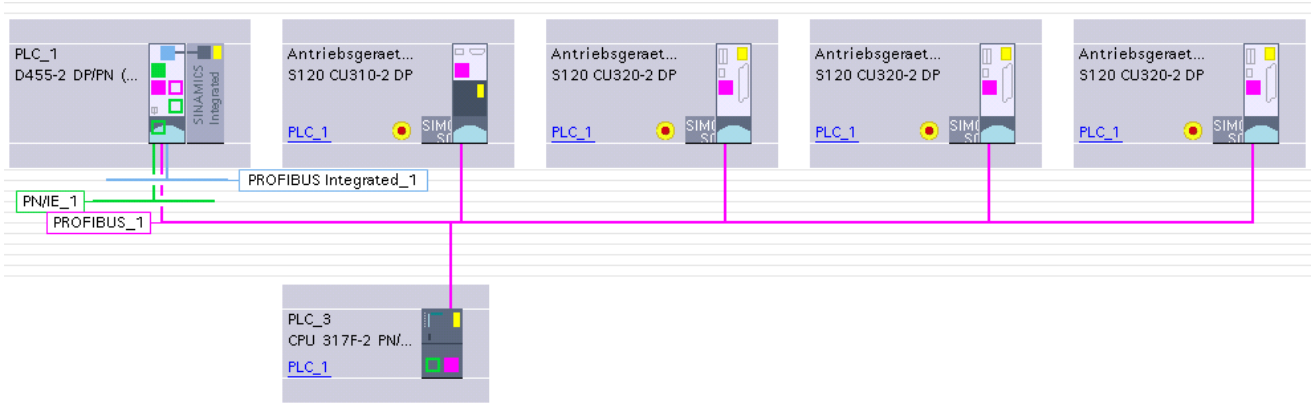


Figure 3-405 F-CPU with SINAMICS via PROFIBUS as a DP slave

5. F-CPU with SIMOTION as shared IO device via PROFINET (GSDML import)

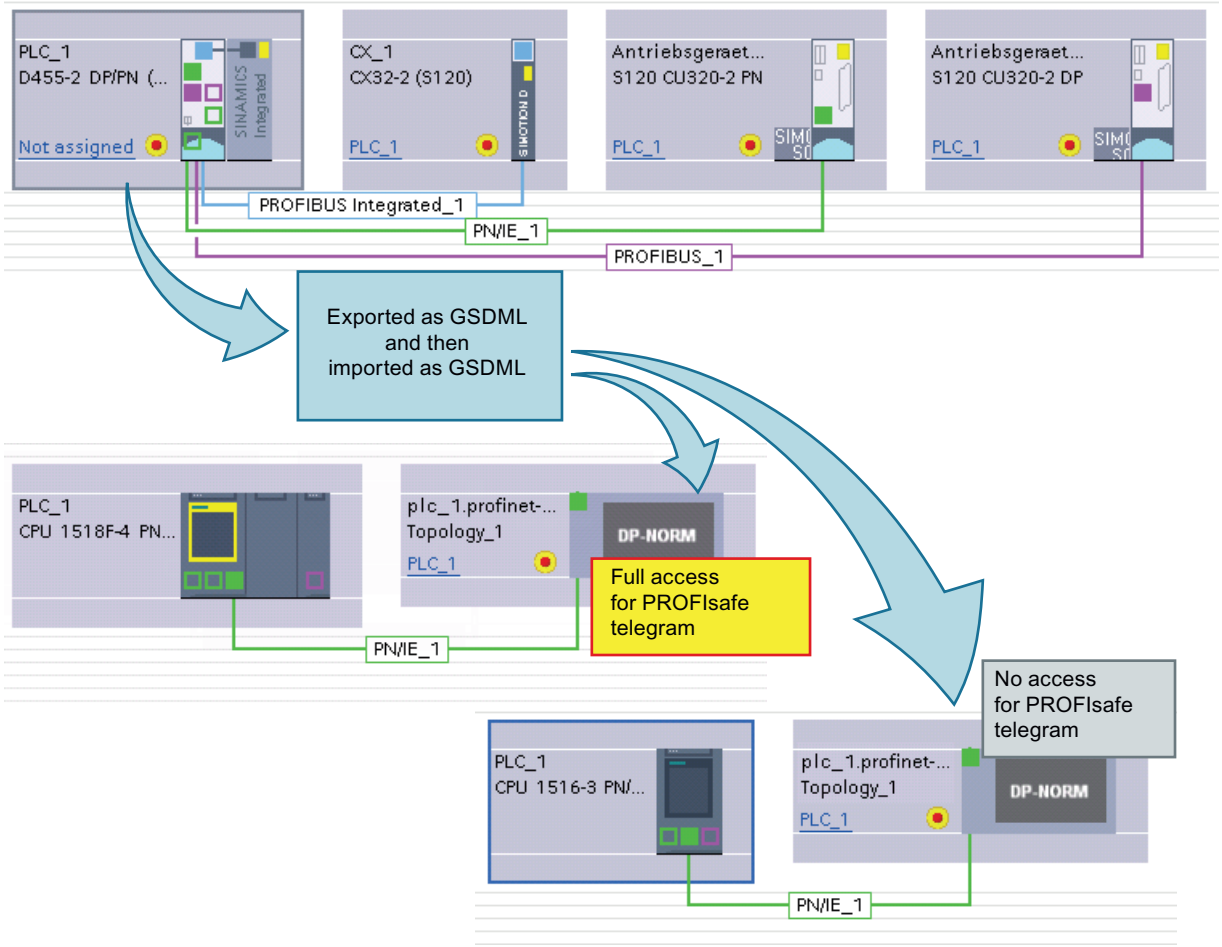


Figure 3-406 F-CPU with SIMOTION as shared device via PROFINET

Unique PROFIsafe addresses

You must check the uniqueness of the PROFIsafe addresses for all nodes in every topology. Typical points to be considered with respect to the topologies illustrated are described in the table.

Topology	F-destination address (F_Dest_Add)
Re 2:	Compare this F-destination address with the values of all other nodes and make sure that the addresses are unique. The nodes in the area between the F-CPU and the coupled SIMOTION CPU must be checked.
Re 3:	Compare this F-destination address with the values of all other nodes and make sure that the addresses are unique. The nodes in the F-CPU area must be checked. The nodes of the SIMOTION CPU in the imported project (topology 1) must also be checked.
Re 4 and 5:	Compare this F-destination address with the values of all other nodes and make sure that the addresses are unique. The nodes in the F-CPU area must be checked.

Shared Device and PROFIsafe

Shared Device and PROFIsafe

Description

With the Shared Device functionality, you can configure access to an IO device with several IO controllers using PROFINET. This enables channels/modules to be flexibly assigned to different IO controllers. This option is available for inputs and outputs. Access to the submodules of the Shared Device is divided between the individual IO controllers. Each submodule of the Shared Device can be assigned exclusively to one IO controller.

You can use this mechanism to access the fail-safe data of a drive configured below a SIMOTION CPU via the F-CPU, for example.

Note

When configuring PROFIsafe, it is recommended that you use the I-Device F-Proxy functionality instead of the Shared Device. The configuration as Shared Device is only relevant when the F-Proxy functionality cannot be used because of the number of axes.

Schematic diagram of Shared Device

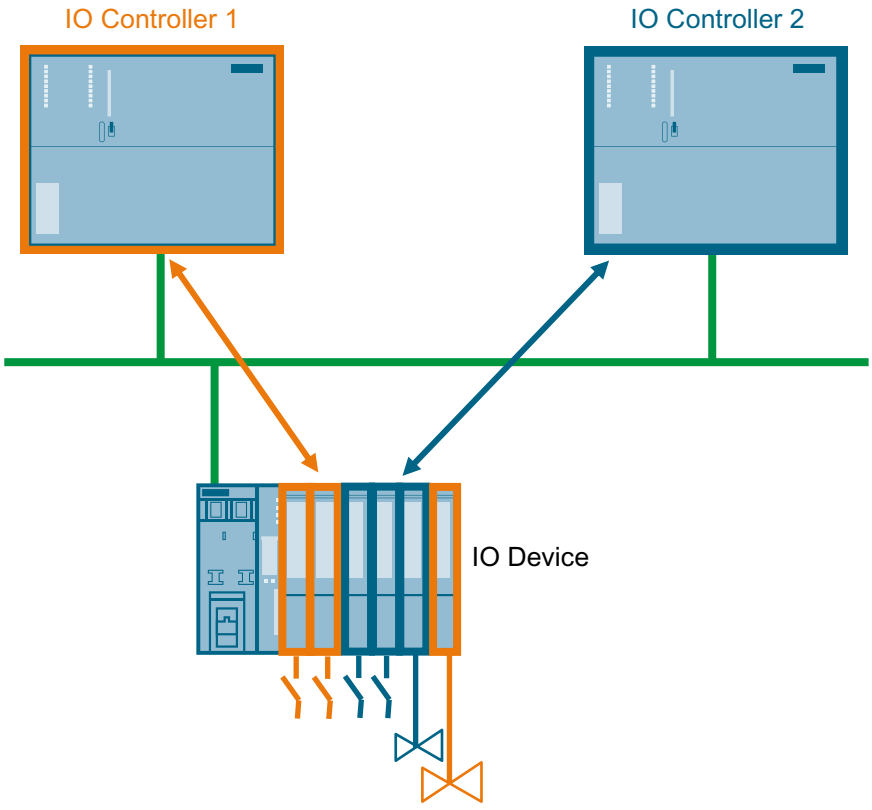


Figure 3-407 Schematic diagram of Shared Device

SINAMICS G120 shared by SIMOTION and F-CPU

Introduction

SINAMICS G120 as Shared Device only supports a fixed division of the subslots between the two sharing IO controllers. The Shared Device functionality is available only for the SINAMICS G120 GSD drives.

The following rule applies:

- Only the safety subslots are assigned to a second IO controller as F-CPU.
- All other subslots are assigned to the automation CPU, i.e. SIMOTION.

Note

The F-Proxy for SINAMICS G120 is**not** supported via GSD and Startdrive.

Configuring

A SINAMICS G120 can only be configured as Shared Device via GSDML. The configuration via GSD is performed in the same way as for a SINAMICS S120. The drive commissioning must be performed separately.

- SINAMICS G120 as GSD drive on SIMOTION CPU (project 1)
- SINAMICS G120 as GSD drive on SIMATIC F-CPU (project 2)
- Drive configuration and SINAMICS G120 commissioning in Startdrive (project 3)

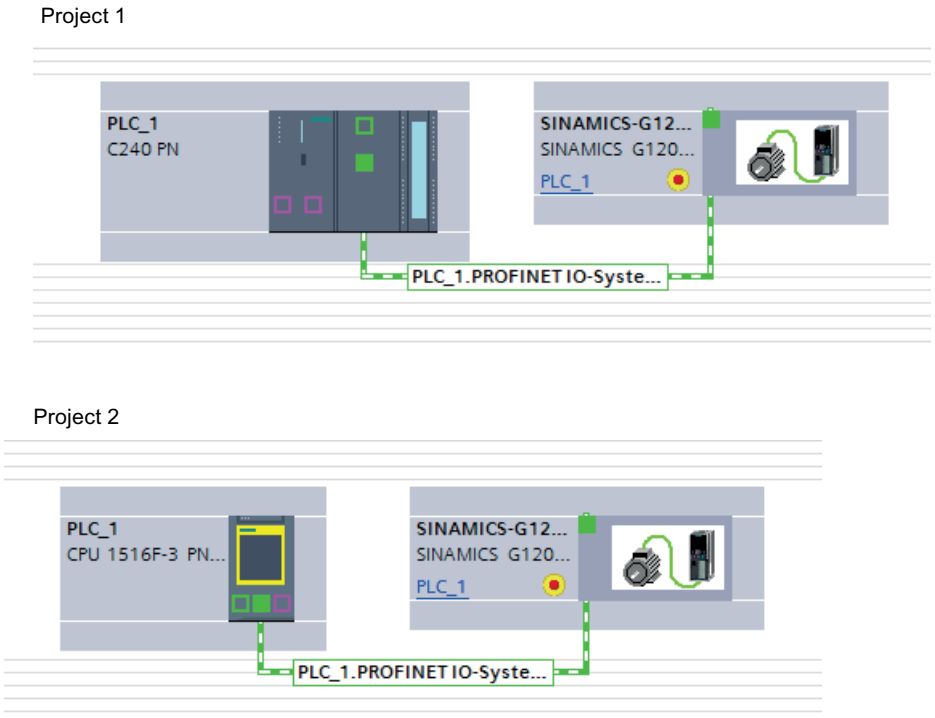


Figure 3-408 Two projects with SINAMICS G120 shared between SIMOTION CPU and F-CPU

Procedure

In the following example, you configure a project with a SIMOTION C240 PN with SINAMICS G120 CU240E-2 PN and a second project with a SIMATIC CPU 1516F-3 PN/DP, which accesses the fail-safe data of the shared SINAMICS G120. You perform the drive commissioning in Startdrive in a third project.

Creating project 1 with SIMOTION CPU and SINAMICS G120

1. Create a new project with a SIMOTION CPU.
2. Insert the SINAMICS G120 CU240E-2 PN (V4.7) IO device via the GSD file. You can find the GSD file in the hardware catalog at "Further field devices > PROFINET IO > Drives > SIEMENS AG > SINAMICS".
3. Assign the IO device to the controller.
4. You must adapt the SINAMICS G120 modules that are to be shared in the device overview.

5. Insert the modules and submodules from the hardware catalog using drag-and-drop. Only the safety modules have to be configured.
   - PROFIsafe telegram (PROFIsafe telegram 30 in the example)
   - SIEMENS telegram (SIEMENS telegram 352 in the example)

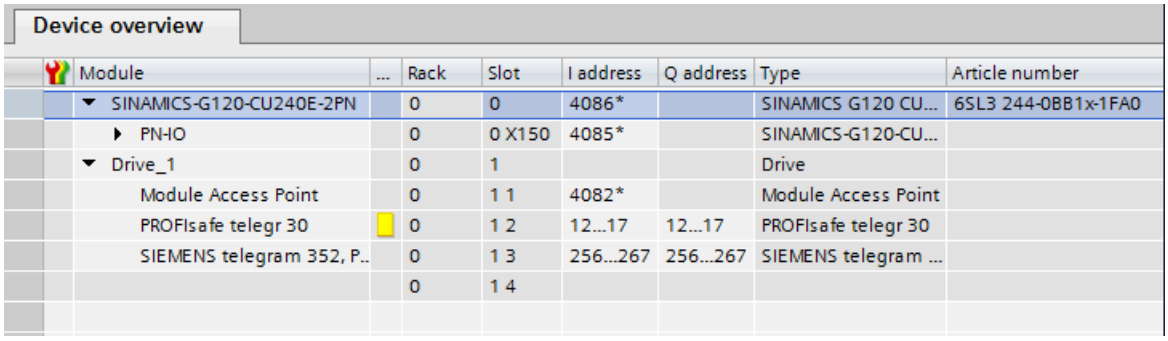


Figure 3-409 SINAMICS G120 inserted via GSD, Shared Device PROFIsafe modules via F-CPU

6. The following parameters and addresses must be identical in all three projects:
   - I/O addresses of the PROFIsafe telegram and the F-parameters (F-source/F-destination address).
   - The device name (also the converted device name) and the IP address.
7. Save the project.

Creating project 2 with F-CPU and SINAMICS G120 via GSD

1. Create a new project with an F-CPU, e.g. SIMATIC CPU 1516F-4 PN/DP.
2. Insert the SINAMICS G120 IO device and proceed as described above in project 1 as of step 3.

Creating project 3 with SINAMICS G120 (Startdrive must be installed)

1. Create a new project with a SINAMICS G120 CU240E-2 PN in the TIA Portal.
2. Run through the drive commissioning in Startdrive and configure the drive and PROFIsafe.
3. To use the SINAMICS G120 as Shared Device, parameter p8929 of the Control Unit must be set to "(2) Automation and safety".
4. Continue as described above in project 1 as of step 6.

Configuring access to the modules of the SINAMICS G120 (Shared Device)

1. Open project 1.
2. Click the SINAMICS G120 drive unit in the network view.
3. In the Inspector window, navigate to the "PROFINET interface > Extended options > Real-time settings > I/O cycle" area.
4. Select the number of project-external IO controllers at "Shared Device" (one in the example).
5. Select "Properties > General > Shared Device" in the Inspector window.

6. Configure the F-CPU access to the safety module. To do this, select the value "---" in the "Access" column in the "PROFIsafe telegram" row. In this way, another controller (the F-CPU in this case) can access these modules.

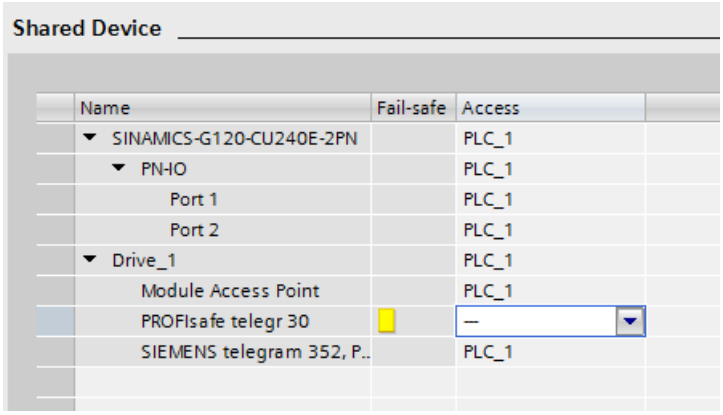


Figure 3-410 SINAMICS G120 Shared Device, automation assigned to the SIMOTION CPU

Note

If you select the "---" option, you cannot change the associated parameters for the PROFINET interface and the ports. They are read-only. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. Irrespective of this, the ports can be connected in both projects.

7. Open project 2.
8. In the Inspector window, navigate to the "PROFINET interface > Extended options > Real-time settings > I/O cycle" area.
9. Select the number of project-external IO controllers at "Shared Device" (one in the example). Select "Properties > General > Shared Device" in the Inspector window.
10. Configure the access to the safety module with the F-CPU in the same project. To do this, select the value "PLC_1" in the "Access" column in the "PROFIsafe telegram" row for the local CPU. All other modules are set to the value "---".

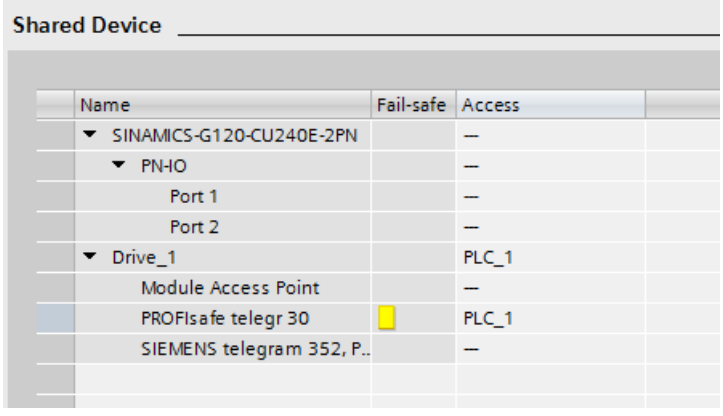


Figure 3-411 SINAMICS G120 Shared Device, PROFIsafe modules assigned to the F-CPU

11. Check whether the same IP address parameters and device names have been set for the Shared Device in all projects and that the modules have not been assigned twice. Only one IO controller may have access to a module.
12. Save all projects.

ET200SP as a shared device on SIMOTION CPU

Introduction

The SIMATIC ET200SP can be operated as Shared Device on SIMOTION controllers as of V4.5. The SIMATIC ET200SP is integrated as IO device, as GSD or via the hardware catalog (MDD). Two higher-level IO controllers share the submodules of the SIMATIC ET200SP. A typical application is a SIMOTION CPU as automation CPU and an F-CPU for the safety application.

Note

Isochronous mode and MRPD are only supported by SIMATIC ET200SP HS.

Applications

In principle, there are three typical applications for the SIMATIC ET200SP as Shared Device with two higher-level controllers.

1. SIMOTION CPU not isochronous and F-CPU
2. SIMOTION CPU isochronous and F-CPU (variation of application 1)
3. SIMOTION CPU isochronous and second CPU not isochronous (e.g. SIMOTION or SIMATIC)

Applications 1 and 2:SIMOTION CPU not isochronous or isochronous and F-CPU

The application is configured in two separate projects.

- SIMATIC ET200SP HS or SIMATIC ET200SP HF as Shared Device
- SIMOTION CPU (CPU 1)
  Accesses the standard modules of the ET200SP station via PROFINET with IRT for isochronous applications.
  Variant:
  SIMOTION accesses the standard modules of the ET200SP via PROFINET RT for non-isochronous applications.
- SIMATIC F-CPU (CPU 2)
  Accesses the safety modules of the ET200SP via PROFINET RT and the safety telegrams of the integrated drives of SIMOTION via the I-Device F-Proxy of SIMOTION.

Support of MSI and MSO

The modules/submodules are logical copies of real IO modules. Isochronous operation is not supported in MSI/MSO mode.

Schematic diagram

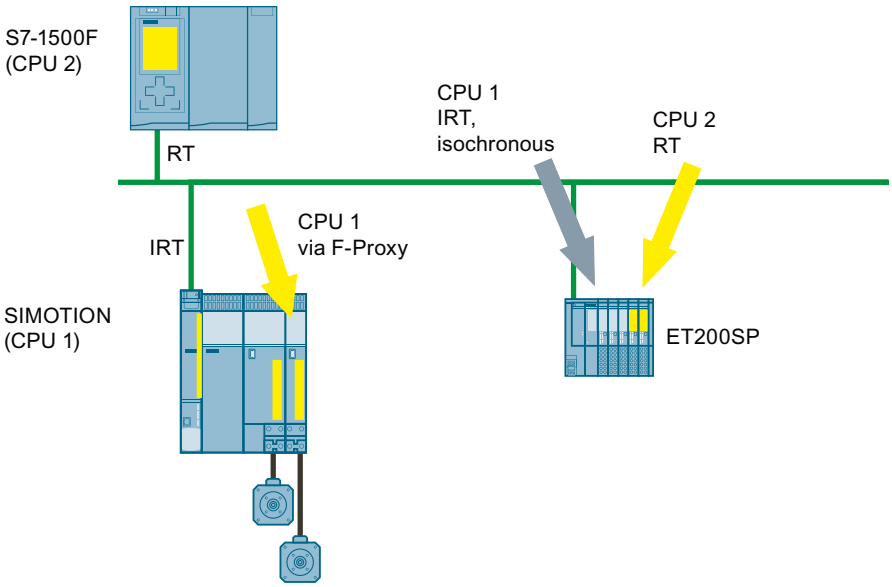


Figure 3-412 SIMATIC ET200SP as Shared Device between SIMOTION CPU IRT isochronous and F-CPU RT (application 2)

Application 3:SIMOTION CPU isochronous and second CPU not isochronous (e.g. SIMOTION or SIMATIC)

The application is configured in two separate projects.

- SIMATIC ET200SP HS as Shared Device
- SIMOTION CPU (CPU 1)
  Accesses the standard modules of the ET200SP station via PROFINET with IRT for isochronous applications.
- SIMOTION or SIMATIC CPU (CPU 2) accesses the non-safety modules via PROFINET RT for non-isochronous applications.
  SIMOTION accesses the standard modules of the ET200SP via PROFINET RT for non-isochronous applications. CPU 2 can also be any SIMATIC CPU, e.g. S7-1500.

Supported modules

Modules can each be DI, DQ, AI, AQ. The modules/submodules are logical copies of real IO modules. Isochronous operation is not supported in MSI/MSO mode.

Schematic diagram

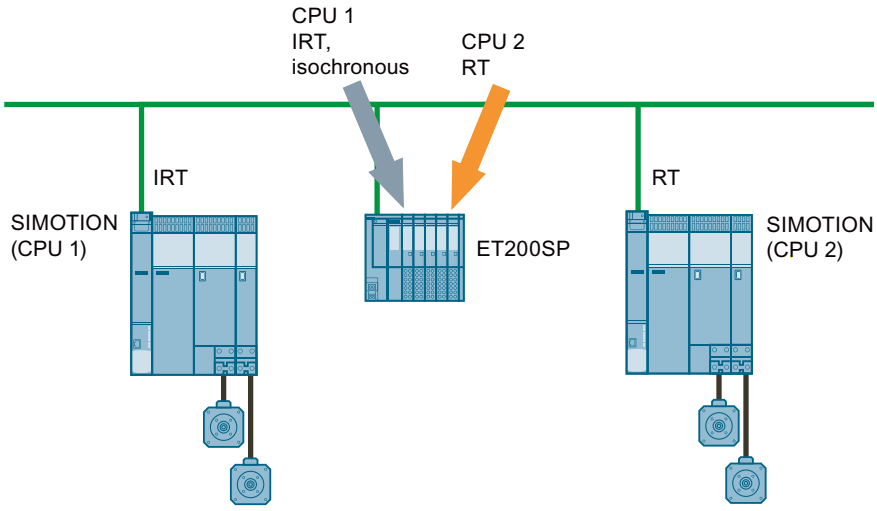


Figure 3-413 SIMATIC ET200SP as Shared Device between SIMOTION CPU IRT isochronous and second SIMOTION/SIMATIC CPU RT

Configuration example of SIMOTION CPU isochronous and F-CPU (application 2)

In the following example, an ET200SP HS isochronous is shared with a SIMOTION CPU and an F-CPU. The F-CPU accesses the PROFIsafe modules via RT.

Table 3-43 Hardware used

Project 1	Project 2
• SIMATIC S71516F-3 PN/DP • ET200SP HS (shared): IM 155-6 PN HS with the following modules DQ 8x24 VDC HF DI 8x24 VDC / 0.5 A HF F-DQ 4x24 VDC / 2 A PM HF F-DI 8x24 VDC HF	• SIMOTION D455-2 PN/DP • ET200SP HS (shared copied): IM 155-6 PN HS with the following modules DQ 8x24 VDC HF DI 8x24 VDC / 0.5 A HF F-DQ 4x24 VDC / 2 A PM HF F-DI 8x24 VDC HF

Configuring project 1 with F-CPU

1. Create a new project 1 in the TIA Portal with the F-CPU and an ET200SP HS with the modules described above.
2. Configure the PN subnet. The ET200SP HS is operated in RT mode.
3. Click the ET200SP HS station in the network view.
4. In the Inspector window, navigate to the "PROFINET interface > Extended options > Real-time settings > I/O cycle" area.
5. Select the number of project-external IO controllers at "Shared Device" (one in the example).
6. Select "Properties > General > Shared Device" in the Inspector window.

7. Configure the F-CPU access to the safety module. To do this, select the value "---" for all modules that are to have access through the SIMOTION CPU from another project. The F-DI and F-DQ modules remain assigned to the F-CPU.
8. Save and compile the project.

Configuring project 2 with SIMOTION CPU

1. Create a new project 2 in a new instance of the TIA Portal with the SIMOTION CPU.
2. Configure the SIMOTION CPU with the appropriate integrated drives and PROFIsafe.
3. Switch to project 1, click the ET200SP HS in the network view and select "Copy" in the shortcut menu.
4. Insert the copied ET200SP HS station in project 2 and assign it to the SIMOTION CPU. The IP address and the device name must be identical and in the same subnet.
5. Click the inserted ET200SP HS station in the network view.
6. In the Inspector window, navigate to the "PROFINET interface > Extended options > Real-time settings > I/O cycle" area.
7. Select the number of project-external IO controllers at "Shared Device" (one in the example).
8. Select "Properties > General > Shared Device" in the Inspector window.
9. Configure the access of the SIMOTION CPU to the modules. The F-DI and F-DQ modules are assigned to the F-CPU and are assigned the value "---". Assign the other modules to the SIMOTION CPU, e.g. value "PLC_1".
10. Configure the SIMOTION CPU as master and the ET200SP HS station as slave in the IRT isochronous mode to the servo.
11. Switch to the device view of the inserted ET200SP HS station and select "Servo" as process image for the DI and DQ modules for the I/O addresses.

    Note

    An ON delay must not be activated for the inputs of the modules that are to be operated in isochronous mode. "Servo" must be selected as process image for the I/O addresses of the isochronous modules.

12. Save and compile the project.

Note

Supplementary condition for ET200SP as a shared device

The PROFINET send cycle must be identical in both projects (sync domains of the controllers). The first higher-level controller operates the SIMATIC ET200SP in IRT isochronous mode, e.g. with a send cycle of 2 ms. The second higher-level controller which has shared access to the SIMATIC ET200SP in RT mode must also operate in a 2 ms send cycle. Only if the two controllers have the same send cycle can the ET200SP establish a connection with both of them. If they have different send cycles, the shared device randomly establishes contact with one controller and cannot be accessed by the other controller.

Shared I-Device and PROFIsafe

Shared I-Device and PROFIsafe

Description

An I-Device can be operated on two higher-level IO controllers as an IO device. This functionality is known as shared I-Device.

It provides the following application areas:

- One of the higher-level controllers works as the "Automation CPU" and uses the SIMOTION I-Device as an isochronous I-Device in the IRT operating mode.
- The other higher-level IO controller works as the "Safety CPU" and uses the submodules configured as the F-Proxy in the RT operating mode.

This means that an I-Device can communicate with two higher-level controllers as an IO device. At the same time, higher-level controllers can access certain modules, e.g. an F-CPU can access the F-Proxy submodules, using Shared Device.

The configuration is performed in three projects. The two higher-level IO controllers are configured in two projects and the SIMOTION CPU in an extra project. The SIMOTION CPU is connected to the higher-level IO controllers via GSD. Only two higher-level IO controllers are supported.

Schematic diagram

The schematic diagram below shows two IO controllers which share the submodules on the I-Device of a third IO controller.

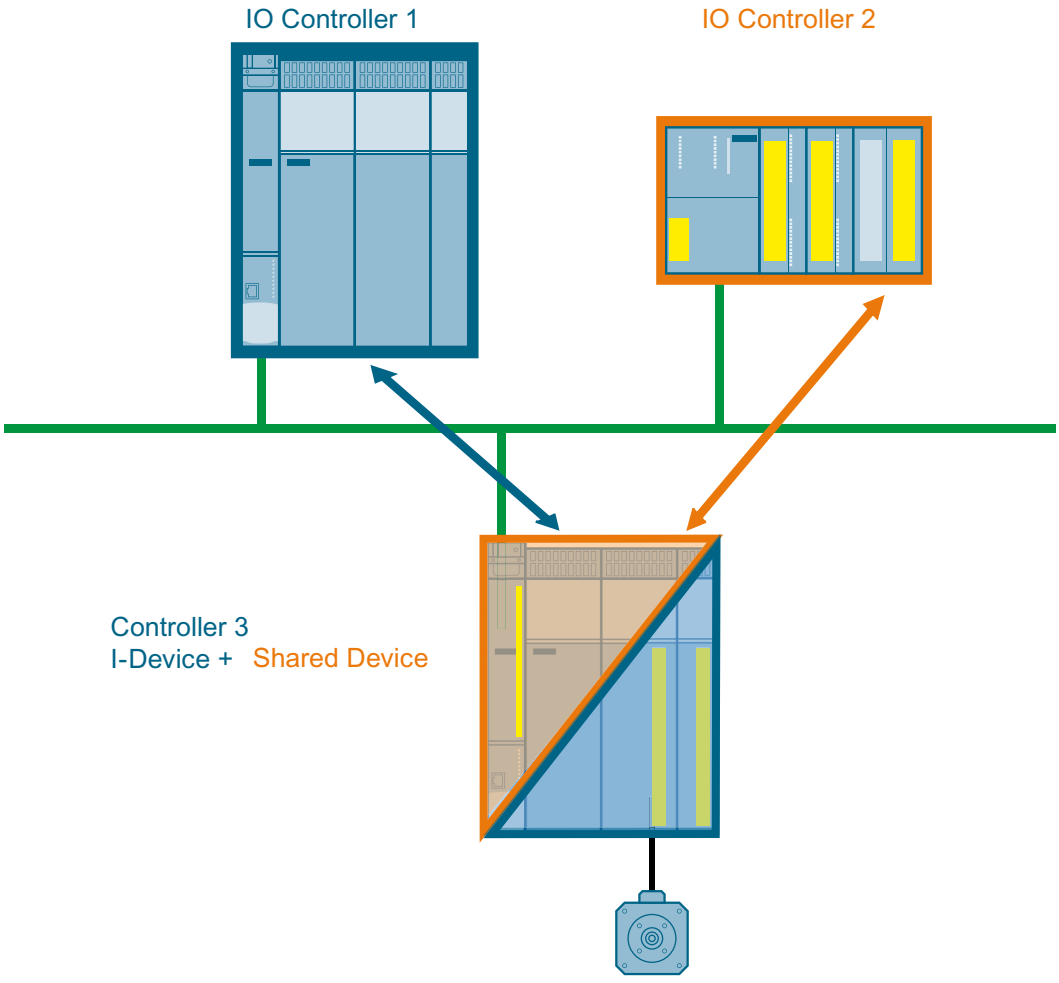


Figure 3-414 I-Device as Shared Device

Note

In this configuration, only the submodules that are used by the "Automation CPU" are configured as isochronous (IRT).

Applications

In principle, there are five typical applications for the SIMOTION CPU as shared I-Device with two higher-level controllers.

1. Shared I-Device via RT (or IRT not isochronous) - RT
   - SIMOTION CPU is a shared I-Device on PN Integrated or CBE30-2
   - Automation CPU 1 SIMATIC (RT or IRT not isochronous)
   - Automation CPU 2 SIMATIC (RT)
2. Shared I-Device via IRT isochronous
   - SIMOTION CPU is a shared I-Device on PN Integrated or CBE30-2
   - Automation CPU 1 SIMOTION (IRT isochronous)
   - Automation CPU 2 SIMOTION (RT)
3. Shared I-Device via RT (or IRT not isochronous) - PROFIsafe
   - SIMOTION CPU is a shared I-Device on PN Integrated or CBE30-2
   - Automation CPU 1 SIMATIC (RT or IRT not isochronous)
   - F-CPU SIMATIC (RT)
4. Shared I-Device via IRT isochronous - PROFIsafe
   - SIMOTION CPU is a shared I-Device on PN Integrated or CBE30-2
   - Automation CPU 1 SIMOTION (IRT isochronous)
   - F-CPU SIMATIC (RT)
5. Shared I-Device via PROFIsafe - PROFIsafe
   - SIMOTION CPU is a shared I-Device on PN Integrated or CBE30-2
   - F-CPU 1 SIMATIC (RT)
   - F-CPU 2 SIMATIC (RT)

Configuration example of a shared I-Device and PROFIsafe

Configuration example

In the following example, a SIMATIC S7-1500 is used as automation CPU and an S7-15xxF for safety for the central machine logic. The SIMOTION CPU performs the motion control task and is coupled to the automation CPU and the F-CPU via PROFINET RT as shared I-Device. The F-CPU accesses the SIMOTION drives via F-Proxy. The configuration must be performed in three projects.

Extended application

Several SIMOTION CPUs can be used as shared I-Device as extended application The SIMOTION CPUs work in a synchronized group using direct data exchange. The shared I-Devices are all PROFINET RT.

In the following example, only one SIMOTION CPU is configured as shared I-Device.

Schematic diagram

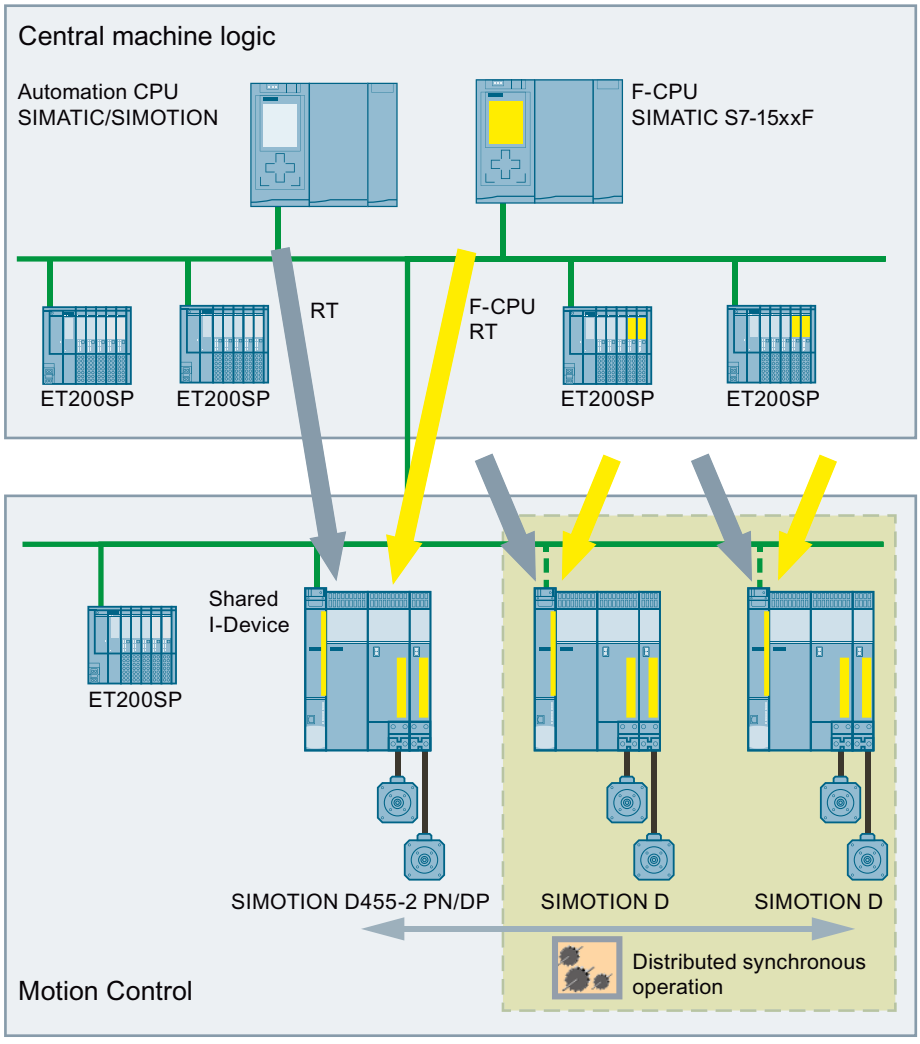


Figure 3-415 Two higher-level SIMATIC IO controllers access the SIMOTION CPU via shared I-Device (RT). Optionally, also additional SIMOTION CPUs in synchronous operation.

Hardware used in the projects

Project 1 (I-Device)	Project 2 (automation CPU)	Project 3 (F-CPU)
• SIMOTION D455-2 PN/DP • SINAMICS S120 on the PROFIBUS Integrated axis assigned with Safety Integrated	• SIMATIC S7-1516-3 PN/DP • SIMOTION I-Device via PROFINET RT	• SIMATIC S7-1516F-3 PN/DP • SIMOTION I-Device via PROFINET RT

Project 1: Configuring the SIMOTION CPU, drive and PROFIsafe

In project 1 you configure the SIMOTION CPU and create the SIMOTION I-Device.

1. Create a new project in the TIA Portal with a SIMOTION D455-2 PN/DP.
2. In SIMOTION SCOUT TIA, configure the drive on the SINAMICS Integrated and insert an axis, and assign it to the drive.
3. Configure PROFIsafe:
   - Activate "Safety via PROFIsafe" on the drive.
   - Activate the functions of the SINAMICS Safety Integrated axis.
   - Insert a PROFIsafe telegram during the telegram configuration of the SINAMICS Integrated.
4. Save and compile the project in SIMOTION SCOUT TIA. The addresses are set up.
5. Switch to the TIA Portal and in the network view click the PN interface of the SIMOTION CPU in order to configure the operating mode as I-Device.
6. In the Inspector window, switch to "Properties > General > Ethernet addresses".
7. Assign a unique "IP address" and a unique "PROFINET device name".
8. In the Inspector window, click "Operating mode".
   - Activate "IO-Device".
   - Add the transfer areas.
   - Select the number of IO controllers that are to have access to the I-Device ("Operating mode > Real-time settings > IO cycle", "Shared Device" section).
9. Click the SIMOTION CPU in the network view and select the "F-Proxy" section on the "General" tab in the Inspector window.
   - Enter the "Start address" of the F-CPU.
   - Select "F-CPU in another project".
   - Select the "F-Proxy interface" X150.
10. Save the project.
11. In the Inspector window, switch to "Properties > General > Operating mode > I-Device communication" tab.
12. Click "Export" in the "Export device description file (GSD)" section, select the designation and path and export the GSD file.
13. You must install the exported GSD in the TIA Portal. Select "Options > Manage device description file GSD" in the menu, navigate to the storage path and install the GSD file. The I-Device is displayed at "Other field devices" in the "PROFINET IO > PLCs & CPs" subfolder in the hardware catalog.

Note

The IP address and the device name must be identical for the SIMOTION CPU and the I-Device. Check that the parameters are identical in other projects in which you use the I-Device because with Shared Device, it is the same device.

Project 2: Coupling SIMOTION as shared I-Device to the automation CPU

In project 2, you couple the SIMOTION I-Device to the automation CPU and assign the modules to the IO controller.

1. Create a new project in the TIA Portal with a SIMATIC S7-1516-3 PN/DP and configure the CPU.
2. Insert the I-Device from the hardware catalog (hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
3. Assign the IO controller to the I-Device.
4. In the properties of the I-Device, select the "Shared Device" area. All transfer areas and the PROFINET interface are assigned the local IO controller (PLC_1) in the table.
5. Specify the transfer areas to which the automation CPU should not have access. Select the entry "---" for these areas. These transfer areas are intended for the F-CPU.
6. Save the project.

Project 3: Coupling SIMOTION as shared I-Device to the F-CPU

In project 3, you couple the SIMOTION I-Device to the F-CPU and assign the modules to the IO controller.

1. Create a new project in the TIA Portal with a SIMATIC S7-1516F-3 PN/DP and configure the CPU.
2. Insert the I-Device from the hardware catalog (hardware catalog: Other field devices > PROFINET IO > PLCs & CPs) or copy the I-Device from project 2 and insert it in the network view.
3. Assign the IO controller to the I-Device.
4. In the properties of the I-Device, select the "Shared Device" area. All transfer areas and the PROFINET interface are assigned the local IO controller (PLC_1) in the table.
5. Specify the transfer areas to which the F-CPU CPU should not have access. Select the entry "---" for these areas. For the safety module,. select "PLC_1".
6. Save the project.

Compiling and loading

You only have to compile the configurations for the IO controller and load them in succession to the CPUs. Errors in the configuration and consistency errors are not detected during the compilation. They can cause communication and access errors.

PROFIsafe via PROFIBUS with failsafe direct data exchange

Introduction

This section describes a configuration example for fail-safe slave-to-slave communication. During fail-safe slave-to-slave communication, the SIMOTION CPU is the DP master. The SIMATIC F-CPU is the I slave on PROFIBUS DP and controls fail-safe communication, e.g. with a CU320 of the SINAMICS S120.

Controllers and drives used

- SIMOTON D410-2 DP (motion control)
- SINAMICS S120 CU320-2 DP
- SIMATIC S7-300 CPU317F-2 PN/DP (F-CPU)

Requirements

- Project created in the TIA Portal
- Controllers and drives inserted in TIA Portal
- PROFIBUS DP system with SIMOTION D410-2 DP, SINAMICS S120 CU320-2 DP, and SIMATIC S7-300 CPU317F-2 PN/DP configured in the TIA Portal

Configuring fail-safe slave-to-slave communication

1. In network view, create an F-CPU, a SIMOTION D410 controller, and a SINAMICS S120 in accordance with the hardware installed. The devices are located in the same PROFIBUS DP subnet.

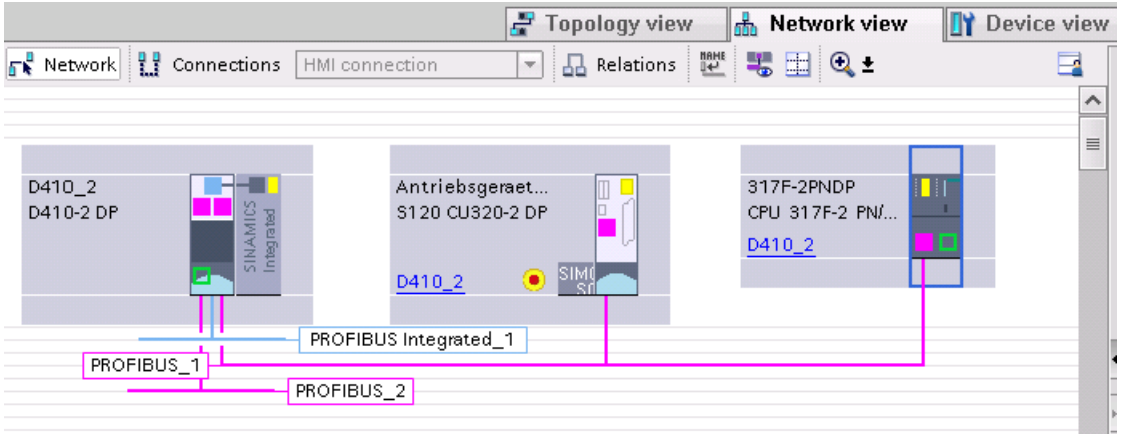


Figure 3-416 Configuration example of fail-safe slave-to-slave communication on the PROFIBUS DP

2. You configure the SIMOTION CPU as a DP master and the SINAMICS S120 as a DP slave, which is isochronously synchronized to the constant DP cycle.

3. You configure the SIMATIC F-CPU as an I slave and enter at least one transfer area per transfer direction.

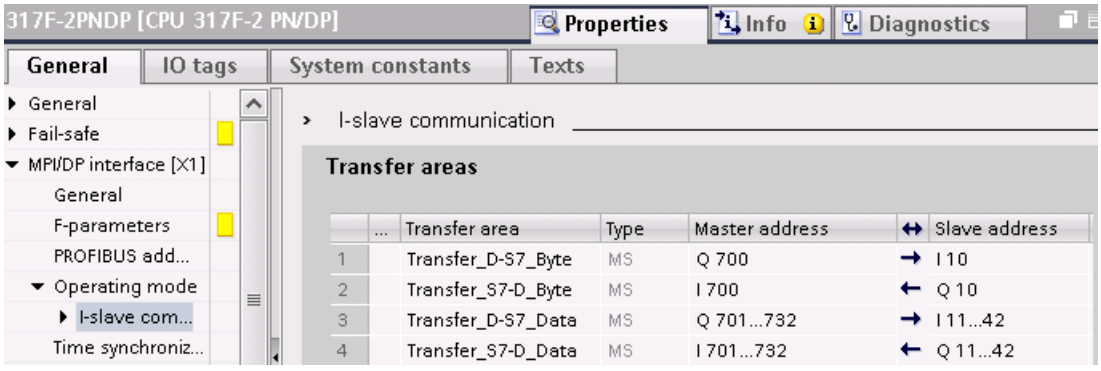


Figure 3-417 I slave communication transfer areas

4. Configure the SINAMICS drive unit (safety configuration) in SIMOTION SCOUT TIA in accordance with your hardware configuration.
5. Then insert a new TO axis in SIMOTION SCOUT TIA and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding telegram will automatically be created (symbolic assignment).
6. Save and compile the project in SIMOTION SCOUT TIA.
7. Create a PROFIsafe slot in SIMOTION SCOUT TIA in the configuration of the SINAMICS drive unit.
   For this, on tab "IF1: PROFIdrive PZD telegrams", select the drive object that will communicate with the SIMATIC F-CPU via PROFIsafe. Click the "Adapt telegram configuration" button and select "Add PROFIsafe". In the example, a PROFIsafe telegram 901 is configured.

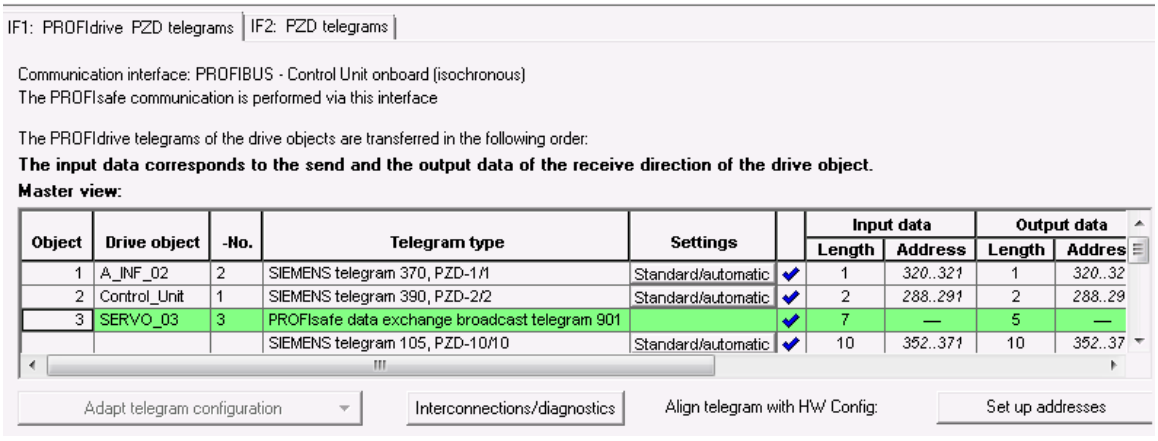


Figure 3-418 Adding a PROFIsafe telegram

8. Transfer the new PROFIsafe slot by clicking the "Set up address" button.
9. Switch to the TIA Portal and in the network view click SIMATIC F-CPU.
10. At the network data, select the "I/O communication" tab. You can show the network data on the right-hand side of the network view.

11. On the transmitter module (SIMATIC F-CPU), select the interface that you want to use for direct data exchange. At "I/O communication", the modules that are configured in this subnet are displayed. Click the SINAMICS drive unit. It is inserted for slave-to-slave communication.

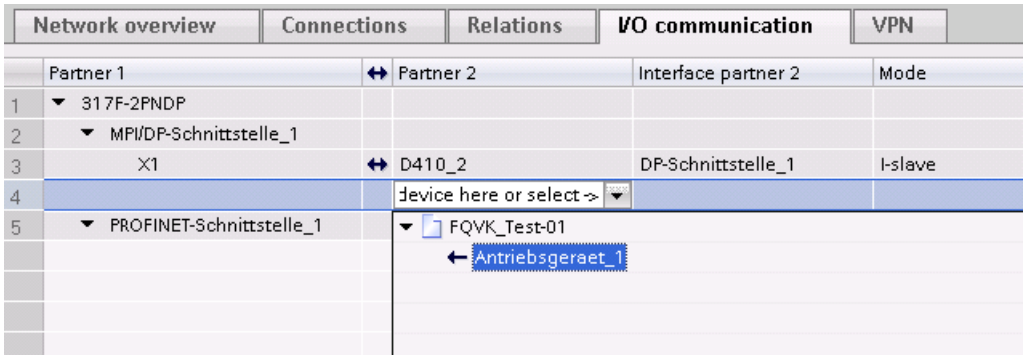


Figure 3-419 Setting up slave-to-slave communication in the TIA Portal

12. On the "I/O communication" tab, click the inserted drive unit. In the "Inspector window", go to the "Properties" tab and then "General" to add a transfer area for fail-safe slave-to-slave communication.

13. Click "<Add new>" and then insert a transfer area of type "F-DX-Mode" and adapt the addresses accordingly.

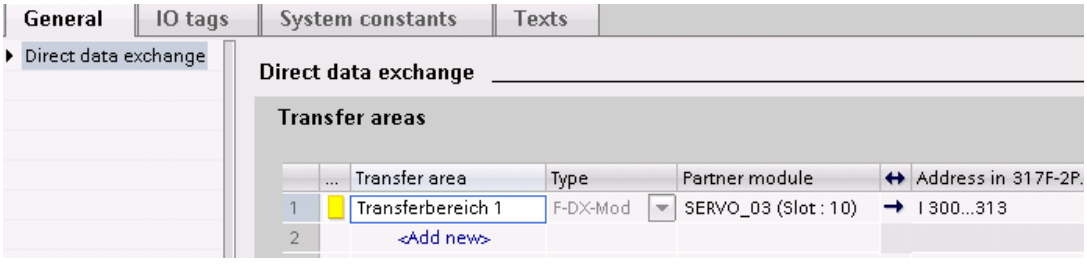


Figure 3-420 Adding a transfer area F-DX-Mode in the TIA Portal

14. Save and compile the project in the TIA Portal. Fail-safe slave-to-slave communication is configured.

3.5.6.4 Configuring onboard IO X142

SIMOTION D4x5-2 I/Os (terminal X142)

The I/Os of terminal X142 are permanently assigned to the SIMOTION D4x5-2. The configuration is performed in the device view of the TIA Portal. The digital I/Os on the X142 connector are provided for the connection of sensors and actuators.

The following types of digital inputs/outputs are available:

- Digital inputs (DI)
- Bidirectional digital I/Os (DI/DO, IN/OUT)

Bidirectional digital I/Os can be configured individually as digital inputs or outputs. Assignment of the I/Os to functions can be parameterized as required. Special functions (e.g. input of measuring input and cam output) can also be assigned to the I/Os.

Note

If you want to interconnect a measuring input or an output cam to the terminals X142 in the SIMOTION SCOUT TIA, you can select all of the I/Os (measuring inputs / output cams) configured here.

You can set the following parameters:

Table 3-44 Channels 0-7

Field/button		Meaning/instruction
IN/OUT X142		Inputs/outputs (IN/OUT 0-7) of the X142
Inverter		Button for the inversion
Function		
	DI	Digital input
	DO	Digital output
	Output cam	Output for output cam
	Measuring input	Input for measuring input
IN filter time		
	Filter time	The selection list has the values 1 µs and 125 µs and is available only for the function "DI" and "measuring input".
Logical addr.		PI, PQ for output cams and measuring inputs The displayed logical address of a channel is only required when symbolic assignments are not used.

Table 3-45 Digital inputs and outputs

Field/button	Meaning/instruction
Input addresses	
Start address	Enter the start address of the digital inputs here.
End address	The end address is determined by the length.
Output addresses	
Start address	Enter the start address of the digital outputs here.
End address	The end address is determined by the length.

Configuration of the D4x5-2 I/Os (terminal X142)

The following figure shows an example interconnection to DI, DO, measuring input, and output cam.

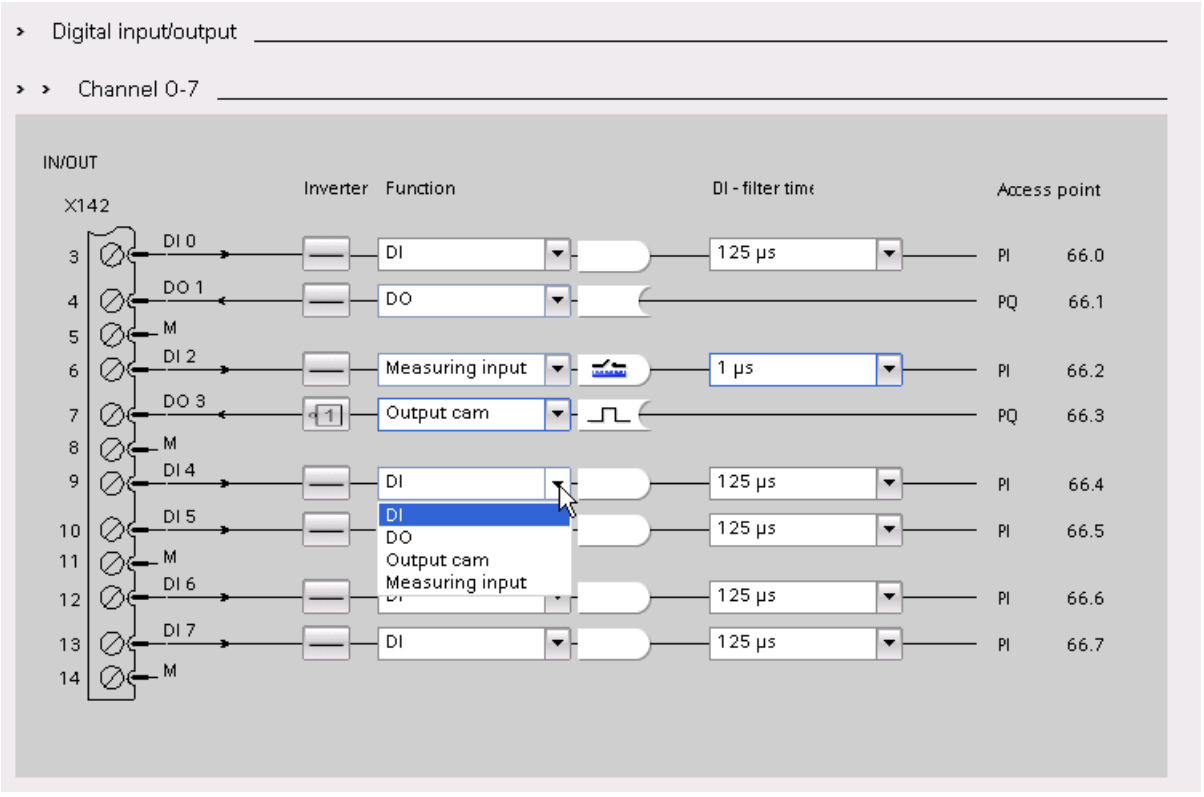


Figure 3-421 Configuration of the D4x5-2 I/Os (terminal X142)

In the SIMOTION SCOUT TIA, you can interconnect the technology object to the configured I/Os of the X142 in the configuration of the technology objects measuring inputs or output cams. In the example, the input X142.6 with the measuring input.

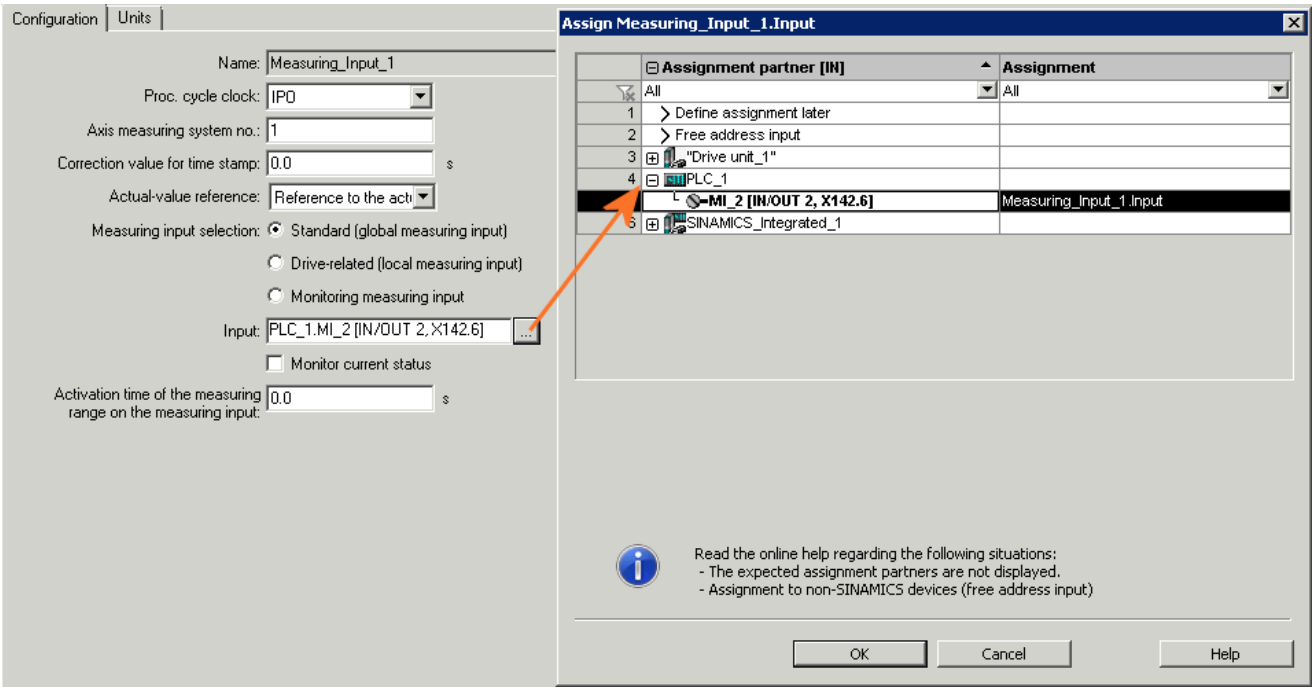


Figure 3-422 Configuring a measuring input at the input X142

3.5.6.5 PROFIBUS DP

Settings on the PROFIBUS

Parameterize PROFIBUS DP

PROFIBUS DP is used for communication between the SIMOTION controller and other devices such as drives or other SIMOTION or SIMATIC controllers. Via DRIVE-CLiQ, you can connect additional modules (CX32-2) to the PROFIBUS Integrated.

For the settings that are possible for the different devices, refer to the documentation of the devices that you use.

Creating a PROFIBUS subnet or master system

To create a new PROFIBUS subnet or a DP master system, proceed as follows:

1. Right-click the interface at which you want to create a system.
2. Select "Create subnet" if you want to create a PROFIBUS subnet at this interface.
3. Select "Assign subnet" if you want to assign an existing subnet to the interface.
4. Select "Create master system" if your device is to act as a DP master.

PROFIBUS address

In general, the PROFIBUS address of the interface is assigned automatically.

Reserve PROFIBUS address "0" for a service PG and "1" for a service HMI device which are connected to the subnet as required.

For service purposes, it is recommended reserving the address "2" for a SIMOTION module. This prevents occurrence of duplicate addresses when a SIMOTION D4x5-2 is installed in the subnet using default settings (for example, when replacing a SIMOTION D4x5-2). You should therefore assign addresses greater than "2" to additional units on the subnet.

The PROFIBUS address set in the TIA Portal must match the address that is set on the device.

You edit The PROFIBUS address either in the device view or in the network view:

- In the device view or network view by selecting the device and then clicking the "PROFIBUS address" entry below the "Inspector window > General > <Interface>". There, set the PROFIBUS address in the "Address" drop-down menu.
- In the device view or network view by directly selecting the interface and then clicking the "PROFIBUS address" entry below the "Inspector window > General". There, set the PROFIBUS address in the "Address" drop-down menu.

Operating mode

To create a DP master system, you need a DP master and at least one DP slave. Once you connect a DP master with a DP slave, a master-slave coupling occurs.

You can change the operating mode subsequently:

1. To do this, select the device in the network view.
2. Change to the Inspector window and call the respective interface from the "Operating mode".
3. Activate the option you want to use for your device.

PROFIBUS Integrated

The PROFIBUS Integrated is operated only equidistantly. You can view the settings by clicking the bus in the network view and the "General" tab in the Inspector window under "Properties".

In order to parameterize the isochronous cycle for the PROFIBUS Integrated, proceed as follows:

1. Select the PROFIBUS Integrated in the network view.
2. Select the "General" tab in the Inspector window and click "Equidistance" to display the parameters for the equidistant PROFIBUS.
   The "Activate equidistant bus cycle" setting is permanently selected and cannot be edited.

The other parameters are set to default settings and can be edited.

See also

Drives on PROFIBUS DP (Page 787)

Cycles with SINAMICS_Integrated (<500 µs)

Introduction

As of SIMOTION SCOUT/SIMOTION SCOUT TIA V4.5, SINAMICS_Integrated for drives on SIMOTION D4x5-2 can use bus cycles of less than 500 µs. It supports cycles between 250 µs and 375 µs. SINAMICS_Integrated remains assigned to the servo cycle. Servo_fast is not supported.

Supplementary conditions

The supplementary conditions for deployment of cycles < 500 µs are listed below.

PROFIBUS and Servo_fast

- External PROFIBUS interfaces cannot be used isochronously
- Since a factor of 4 between Servo and Servo_fast is prescribed for cycles of ≤ 250 µs, Servo_fast with 125 µs cannot be used in parallel to SINAMICS_Integrated with 250 µs.
- Because the Servo cycle must be a multiple of the Servo_fast cycle, Servo_fast cannot be used:
  - Servo_fast > 250 µs: N = 2, 4, 8, 16, 32, ...
  - Servo_fast ≤ 250 µs: N = 4, 8, 16, 32, ...

Number of axes

With regard to utilization, SINAMICS_Integrated is subject to the same restrictions as those applicable to an external SINAMICS drive.

If the SINAMICS_Integrated is operated with a DP Integrated cycle clock of 250 µs or 375 µs, the resultant utilization is higher in comparison to a cycle clock of ≥ 500 µs. Depending on the utilization situation (e.g. by deployed ALMs, CXs, TMs), this can result in a reduction in the number of axes (e.g. from six to five servo drives).

The higher utilization can result in a reduction of the quantity structure for vector and V/f-controlled drives.

CBE30

CBE30 supports a minimum send cycle of 500 µs. The SINAMICS_Integrated can only be operated with 250 µs if no isochronous communication is configured for the CBE30. If the CBE30 operates isochronously (assignment to the Servo), the Servo has a minimum cycle of 500 µs.

Setting the equidistant PROFIBUS

Configuring the equidistance

The PROFIBUS properties of "isochronicity" and "equidistance" form the basis for synchronized editing cycles.

Equidistance ensures exactly the same interval for bus cycles. The "Constant bus cycle time" function ensures that the DP master always starts the PROFIBUS DP bus cycle within a constant time interval. Therefore, the connected slaves also obtain their data from the master in exactly constant time intervals. This is also called "clock of the bus cycle."

Equidistance is the requirement for isochronous operation. For instructions on setting isochronicity, refer to Drives on PROFIBUS DP (Page 787).

Equidistant PROFIBUS

In order to perform the settings for the PROFIBUS, select the bus in the network view.

1. In the Inspector window, select the "Properties > General" tab and click "Equidistance".
2. Parameterize the bus settings there.

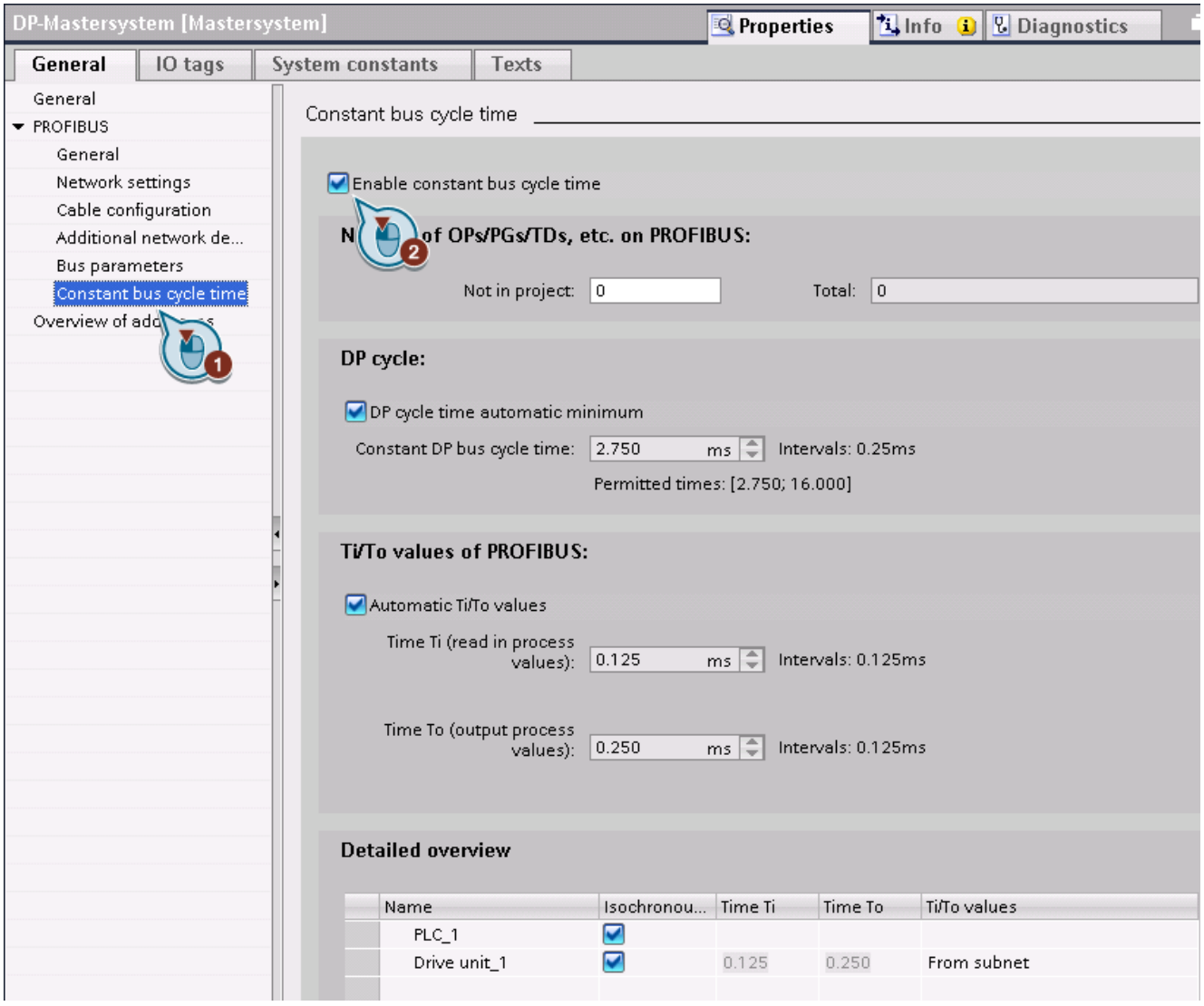


Figure 3-423 Configuring the equidistant PROFIBUS

Parameter		Description
Activate the equidistant bus cycle		Activate this option if you want to operate the bus isochronously.
DP cycle		
	DP cycle time automatically minimal	Activate this option if you want to assign the values automatically with the minimum value.
	Equidistant DP cycle	The equidistant DP cycle in which the master exchanges data with all slaves is set here.
Ti/To values of the PROFIBUS		
	Ti/To values automatic	Activate this option if you want to assign the values automatically.
	Time Ti (read in process values)	Specifies at what time Ti before the start of the "DP cycle", the actual position value (process value) is recorded.
	Time To (output process values)	Specifies at what time To after the start of the "DP cycle" the controller transfers the setpoint (e.g. speed setpoint).

Detailed overview

For an overview of the current settings and nodes on the equidistant PROFIBUS DP, select the "Properties > General" tab in the Inspector window and click "Equidistance". The "Details view" is located in the lower area of the "Equidistance" tab.

Drives on PROFIBUS DP

Settings on PROFIBUS DP

Drives are normally operated on the PROFIBUS DP as slave on a higher-level controller. There is also the possibility of operating the bus equidistantly and isochronously.

For an isochronous mode, you have to click the "Activate equidistant bus cycle" option on the bus, see also Settings on the PROFIBUS (Page 783).

More settings on the drive (DP slave) are necessary for an isochronous PROFIBUS.

For detailed information, refer to the information system of the TIA Portal under "Isochronous" and "Equidistant" operation.

Drive units that are not networked or are assigned to a DP master are not displayed in SIMOTION SCOUT TIA.

Set the isochronization on the drive

1. Click the interface of the drive in the network view.
2. In the "General" inspector window, select "Isochronization."
3. Activate the "Synchronize DP slave to constant DP cycle" option.

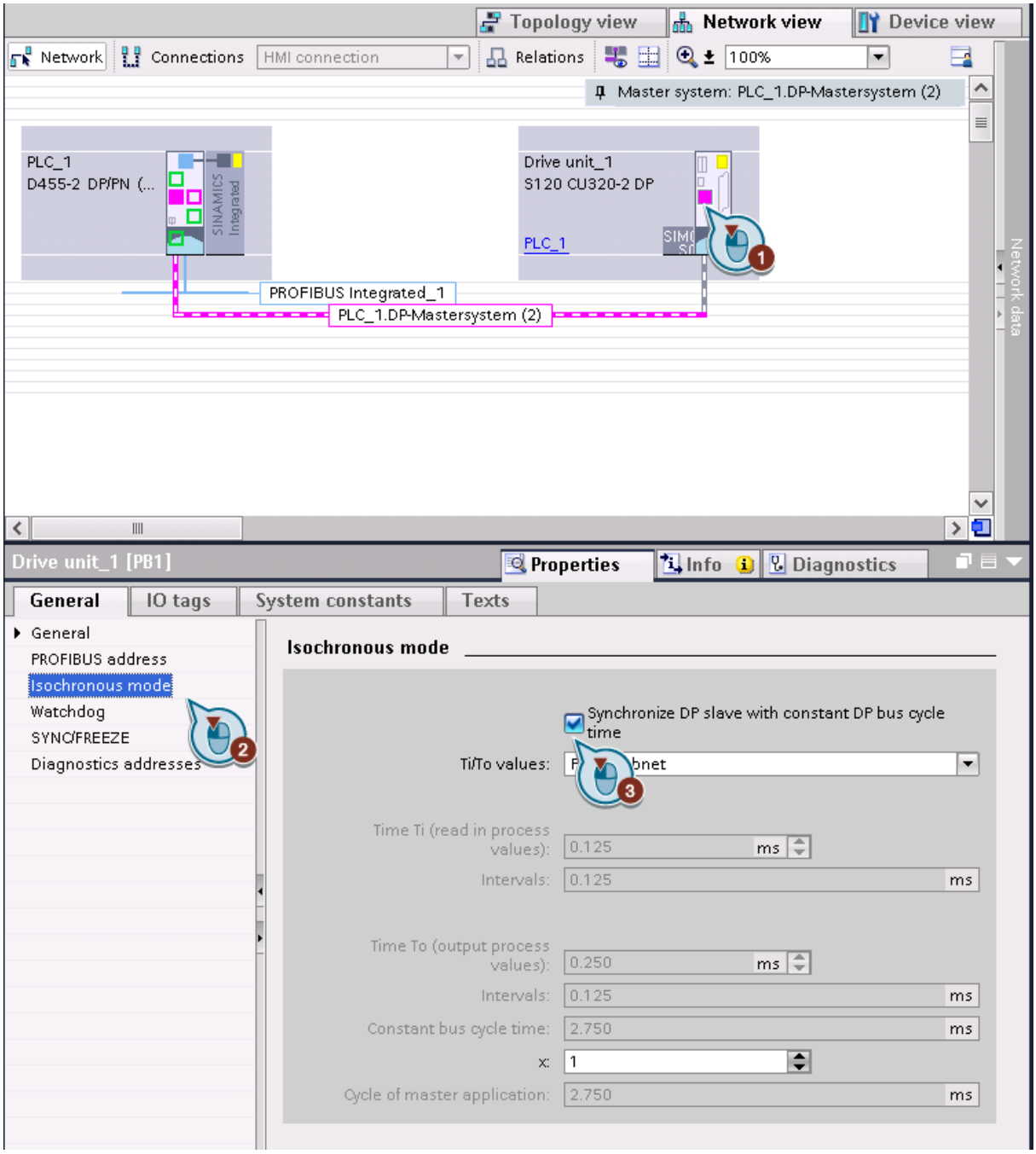


Figure 3-424 Isochronous drives on PROFIBUS DP

The following settings can be made:

Parameter	Description
Synchronize the DP slave to equidistant DP cycle	Activate this option if you want to synchronize the drive (DP slave) to the equidistant DP cycle.
Ti/To values	From the drop-down menu, select whether you want to enter the Ti/To values manually or obtain them from the subnet. The values for Ti/To are then set the same for all DP slaves.

Parameter	Description
Time Ti (read in process values)	Specifies at what time Ti before the start of the "DP cycle", the process value (actual position value) is recorded. If several drives operate as group, the same value for all axes should be entered here.
Grid	Indicates in which field the value is read.
Time To (output process values)	Specifies at what time To after the start of the "DP cycle" the controller transfers the process value (e.g. speed setpoint). If several drives operate as group, the same value for all axes should be entered here.
Grid	Indicates in which field the value is read.
Equidistant DP cycle	Displays the calculated cycle time in the equidistant PROFIBUS.
x	Factor which is multiplied by the cycle time.
Cycle of the master application	Enter the master application cycle here. The cycle describes the reduction ratio between the DP cycle and the cycle of a higher-level controller, e.g. the servo cycle of a SIMOTION controller.

Direct data exchange via PROFIBUS DP

Configuring direct data exchange between two controllers

Master-slave communication is possible between two SIMOTION controllers via PROFIBUS DP using direct data exchange.

You can find more information in the information system of the TIA Portal under "Direct data exchange."

One module is the DP master, the other module is the I slave. This makes, for example, distributed synchronous operation on two modules possible.

Procedure

To configure communication for this topology, proceed as follows:

1. Insert two SIMOTION modules in the TIA Portal.
2. Create a DP master system. Preferably, you should create the system on the module that is to be the DP master and drag the PROFIBUS DP onto the second module.
3. In the network view, select a module which is to be the master.
4. In the inspector window on the "General" tab under the interface used under operating mode, select the option "DP Master."
5. In the network view, select a module which is to be the I slave.
6. In the Inspector window on the "General" tab under the interface used, select the item "Operating mode".
7. Select the "DP Slave" option.
   The direct data exchange is automatically created and displayed under I/O communication.

8. In the "Assigned DP master" dropdown list, select the controller with the interface being used. Additional parameters are listed under "I-Slave communication":

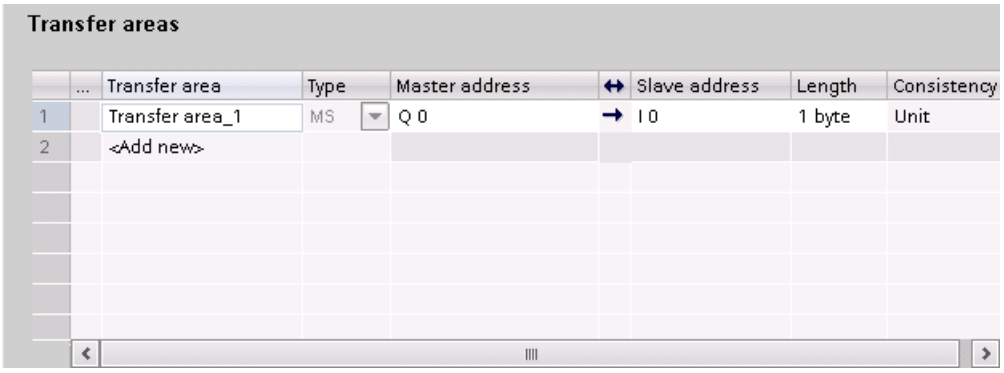


Figure 3-425 Transfer area

9. Under "Transfer areas" click "Add new". A new transfer area is created.
10. Select type "MS" (Master - Slave).
    The addresses are entered automatically.
11. In the "Transfer direction" column, click the arrow to reverse the communication direction.
    This activates data transfer from the "sender" (master) to the "receiver" (I-Slave) and vice versa.

Type of transfer areas

In the TIA Portal, different types of communication (PROFIBUS, PROFINET) can be configured. Depending on this configuration, different short codes are displayed as the transfer area type.

- PROFIBUS DP
  In master I-slave communication, "MS" is displayed as the type. For synchronous and distributed synchronism configurations that are generated automatically by SIMOTION SCOUT TIA, "M-MS" is displayed as the type. M-MS means motion control with master I-slave communication.

See also

F proxy with SIMOTION (Page 744)

3.5.7 Configuring an HMI connection

3.5.7.1 Supported HMI panels

An overview of the HMI panels supported with SIMOTION SCOUT TIA can be found in the Internet at: Compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317).

3.5.7.2 Adding an HMI

Note

You can only add an HMI device in the TIA Portal.

An HMI device can communicate with a SIMOTION device via the PROFINET, Ethernet and PROFIBUS bus systems. The SIMOTION device and the HMI operator panel can exchange information about variables for technology objects, system variables and global user variables.

Procedure

To add an HMI, proceed as follows:

1. In the network view, browse to the "Hardware Catalog" task card and open the "HMI" component.
2. Open the folder of the panel type, "SIMATIC Comfort Panel" in the example.
3. In the folder for the display size, select the installed panel, "15" Display > TP1500 Comfort" in the example.
4. Drag the HMI panel into the network view of your project.

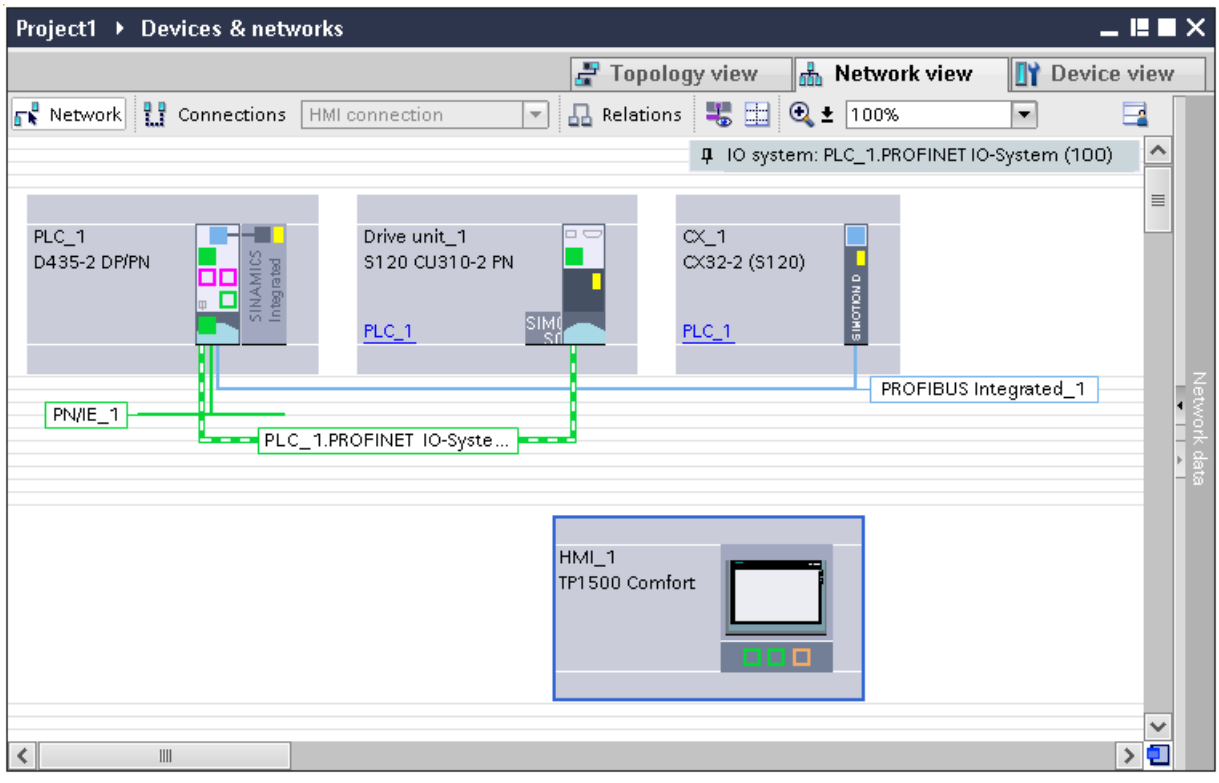


Figure 3-426 HMI in the network view

Result

The inserted HMI is visible in the project navigator.

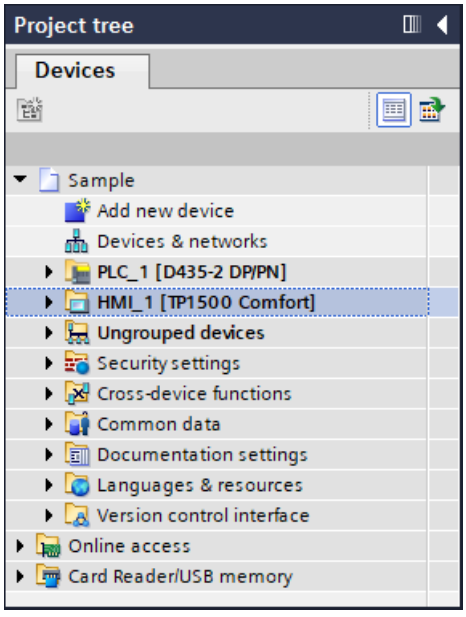


Figure 3-427 HMI in the project navigator

3.5.7.3 Synchronizing SIMOTION variables and messages

So that they can be displayed on an HMI, the variables and messages of SIMOTION SCOUT TIA must be transferred to the TIA Portal.

Procedure

To determine when the variables and messages will be transferred, proceed as follows:

1. Switch to SIMOTION SCOUT TIA.
2. Select "Options > Settings" in the menu.
3. Switch to the "WinCC" tab and select when the data is to be synchronized, e.g. "Automatically with Save and compile".
4. Save and compile your project.

Result

The variables have been transferred to the TIA Portal and can be called there in the project navigation below the SIMOTION device with the "SIMOTION variables > Show all variables" entry.

Note

SIMOTION variables

The following SIMOTION variables are visible after synchronization in the TIA Portal:

- System variables of the technology objects and of the SIMOTION device
- Configuration data of the technology objects
- Global device variables
- Global unit variables
- I/O variables

User-defined data types are also available in the TIA Portal.

The messages (user alarms and alarms of the technology objects) have been transferred to the TIA Portal and are available there for the message configuration for the HMI.

Note

SIMOTION messages

The following messages are also available in the TIA Portal after transfer and/or synchronization:

- Alarm_S messages (TIA Portal text lists are supported)
- TO alarms (separate alarm class in the TIA Portal)
- User-defined diagnostic buffer entries
- System-side diagnostic buffer entries

The display language for the entire project including message texts is set in the project navigation of the TIA Portal at "Languages & Resources".

Additional references

You will find detailed information on creating and compiling message texts in Section "Message configuration" in the SIMOTION SCOUT TIA Online Help.

See also

Using tags (Page 847)

3.5.7.4 Creating an HMI connection

To make this connection, the SIMOTION device and the HMI device must have been created in HWCN. The HMI device must support SIMOTION CPUs.

There are two ways of establishing the communications connection between the SIMOTION CPU and the HMI device:

- The communications connection is configured manually in the network view.
- The communications connection is generated automatically when a SIMOTION variable is used in an HMI image.

Configuring an HMI connection manually

To configure the HMI connection manually, proceed as follows:

1. Switch to the network view.
2. Click Connections and select "HMI connection" in the drop-down list.
   The connection option is then highlighted in color.

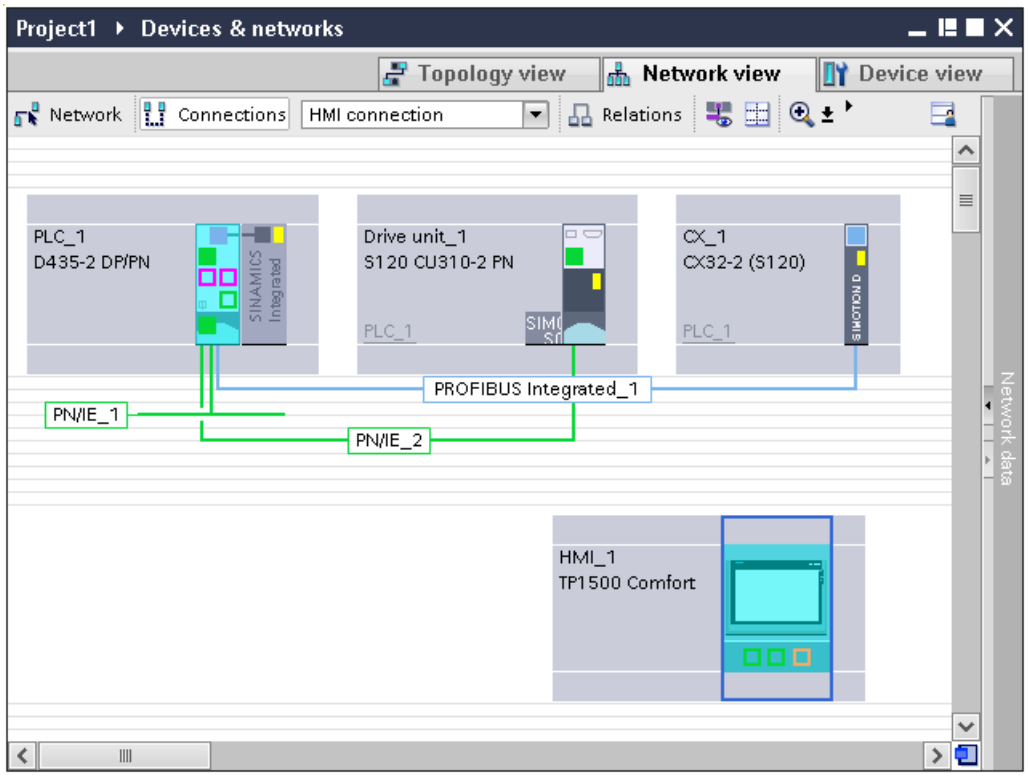


Figure 3-428 Displaying an HMI connection

3. Move a connection from the HMI device to the SIMOTION device with a drag-and-drop operation.

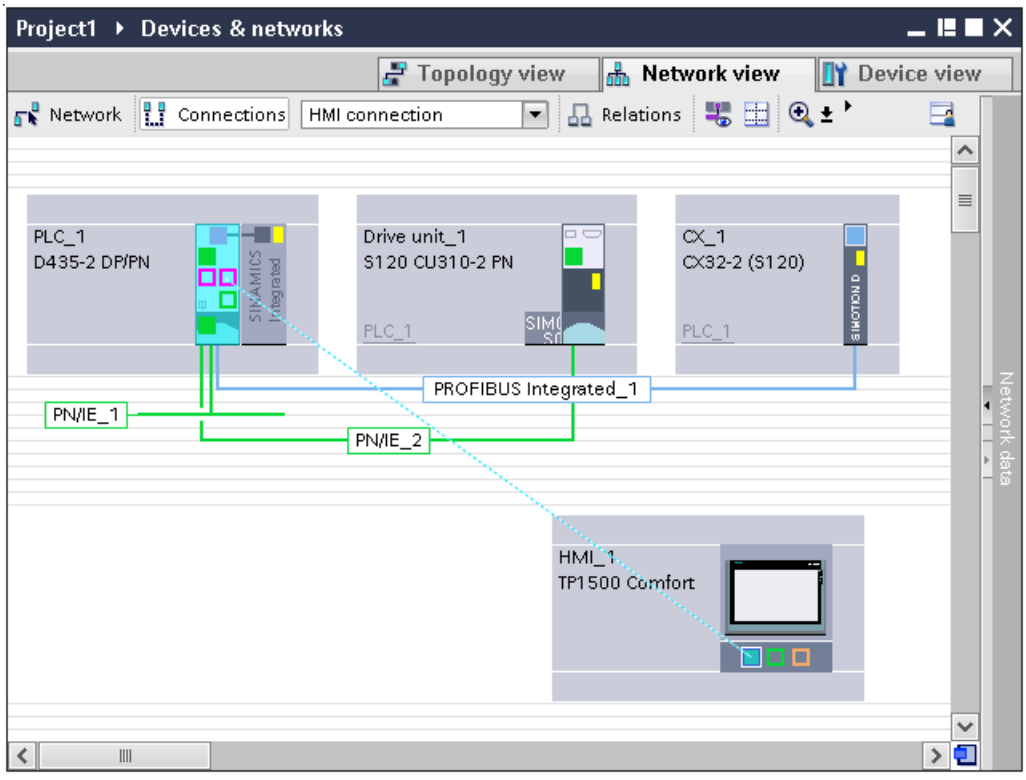


Figure 3-429 Configuring an HMI connection

The parameters of this connection are displayed in the "Properties" tab of the inspector window.

You can specify the connection subsequently here and/or create a suitable connection path to the partner via "Find connection path".

Result

The HMI connection has been established.

You can connect parameters in the "Connections" editor.

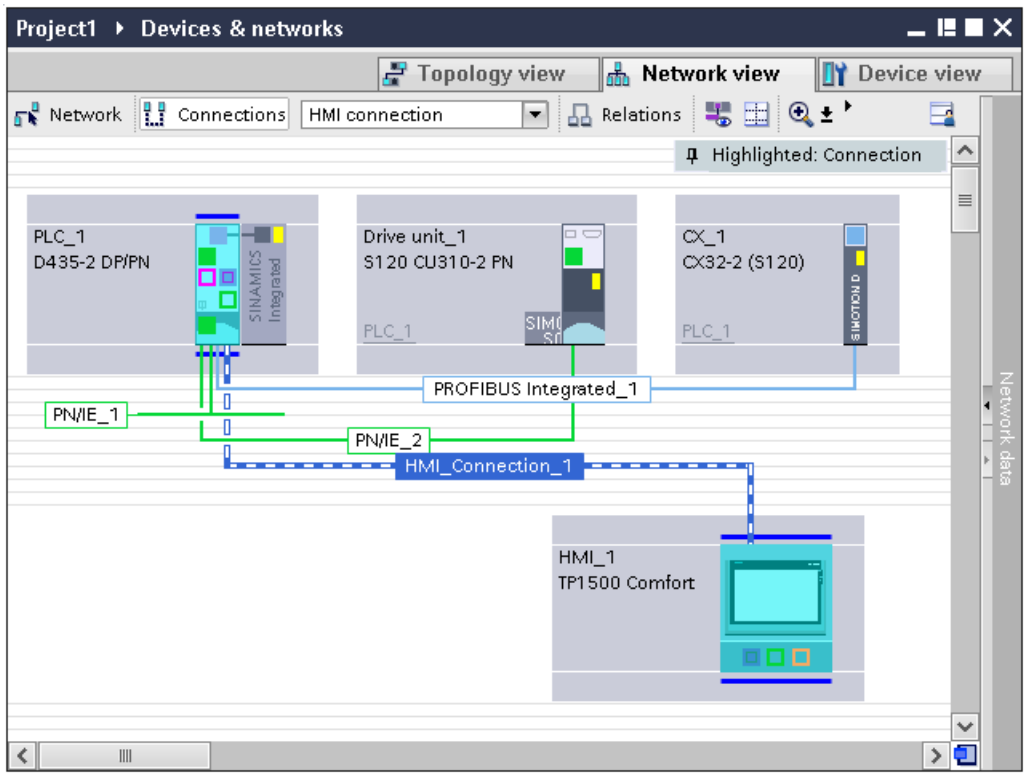


Figure 3-430 HMI connection created

Generating an HMI connection using variables

To generate the HMI connection using variables, proceed as follows:

Creating an HMI screen

1. In the TIA Portal, browse to the HMI in the project navigator.
2. Click "Images" and double-click "Insert new image".

Adding I/O fields

1. In the "Tools" window, select the desired element.

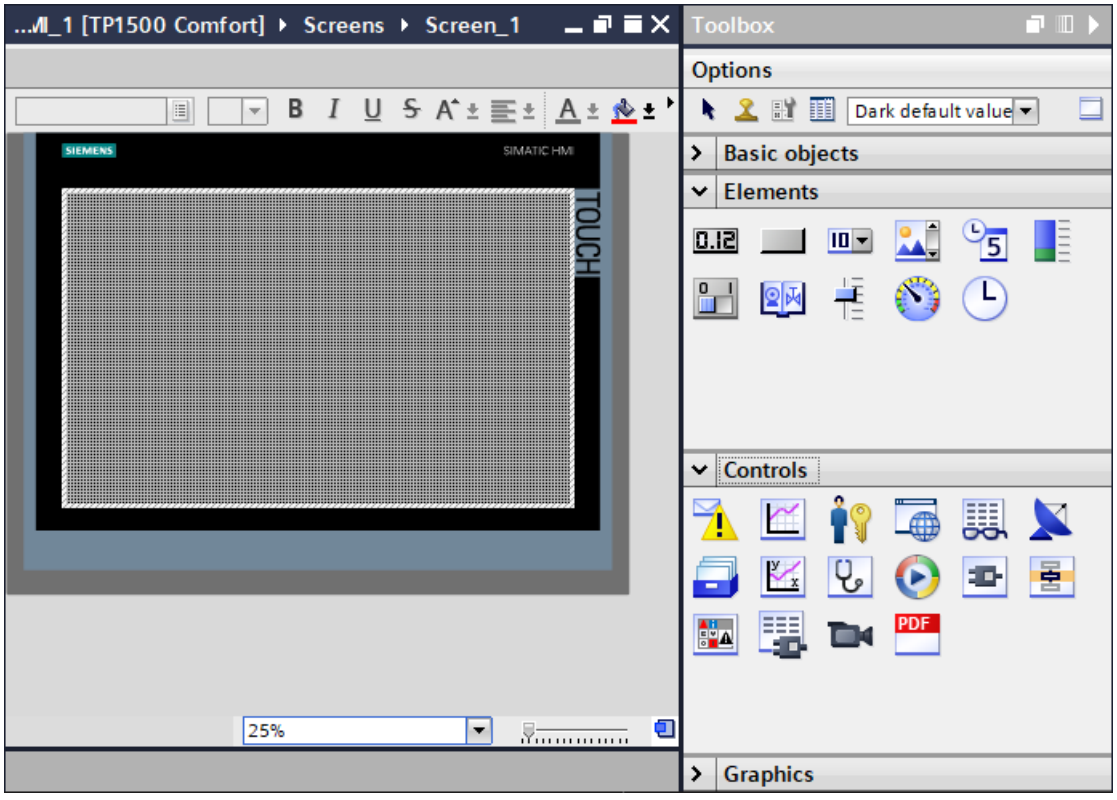


Figure 3-431 Selecting an element

2. Drag the element to the approximate position on the screen at which you want it to appear later.
3. Select the element and switch to the properties window.
4. In the navigation, select the entry "General".
   In this area, you can configure the connection of the process variables and settings to the mode and displayed format.
5. Drag the variable from the project navigator to the properties window onto the "Variables" entry.

Result

The system automatically creates an HMI variable assignment.

You can see the assignment of the connections in the network view of the device configuration.

3.5.7.5 Testing the connection

To test the created connection with the HMI device in the TIA Portal, proceed as follows:

1. Configure an alarm window on the HMI device.
   To do this, select the object "Alarm control" under "Controls".
2. Define "Alarm buffer" as display setting.
3. Compile the project.
4. Load the project to the controller and to the HMI device.
5. Switch the controller's operating mode to RUN.

Result

Buildup of the connection is displayed in the alarm window.

3.5.8 Motion Control parameterization/programming in SIMOTION SCOUT TIA

3.5.8.1 Start SIMOTION SCOUT TIA

Note

You can only start SIMOTION SCOUT TIA from the TIA Portal.

After you have created the project in the TIA Portal and you have configured the hardware and communication, start SIMOTION SCOUT TIA to configure the technology.

Procedure

You can start SIMOTION SCOUT TIA in the following ways:

In the portal view

1. Click the "Motion & Technology" function.
2. Click the "Open SIMOTION configuration" entry in the secondary navigation.

In the project view

1. Start SIMOTION SCOUT TIA via the "Project > Open SIMOTION configuration" menu.
   Or double-click the "SIMOTION configuration" menu entry below the SIMOTION device in the project navigation.

Note

Disconnecting the online connection to the TIA Portal

You should not go online simultaneously in the TIA Portal and SIMOTION SCOUT TIA.

Always disconnect the online connection in the TIA Portal before going online with SIMOTION SCOUT TIA. You need to do this so that programs can be compiled in SIMOTION SCOUT TIA.

Making basic settings

In SIMOTION SCOUT TIA, you can make various basic settings via the "Options > Settings..." menu.

To make the settings, proceed as follows:

1. In the "Options" menu, select the "Settings..." command.
   The "Settings" dialog opens.
2. Switch to the desired tab.
3. Make the settings.
4. Confirm with "OK".

Additional references

You will find detailed information on the individual tabs and settings in the SIMOTION SCOUT TIA Online Help.

3.5.8.2 Going online/offline with SIMOTION SCOUT TIA

Selecting the access point and target devices

Online access points

You have two access points for the communication of SIMOTION SCOUT TIA with controllers and drive units.

- The properties of the access point **S7ONLINE** are defined in the TIA Portal. With this access point, you go online via the interface that you have configured in the TIA Portal. The settings in SIMOTION SCOUT TIA are not used and are not effective.
  The settings for this access point are taken from the TIA Portal.

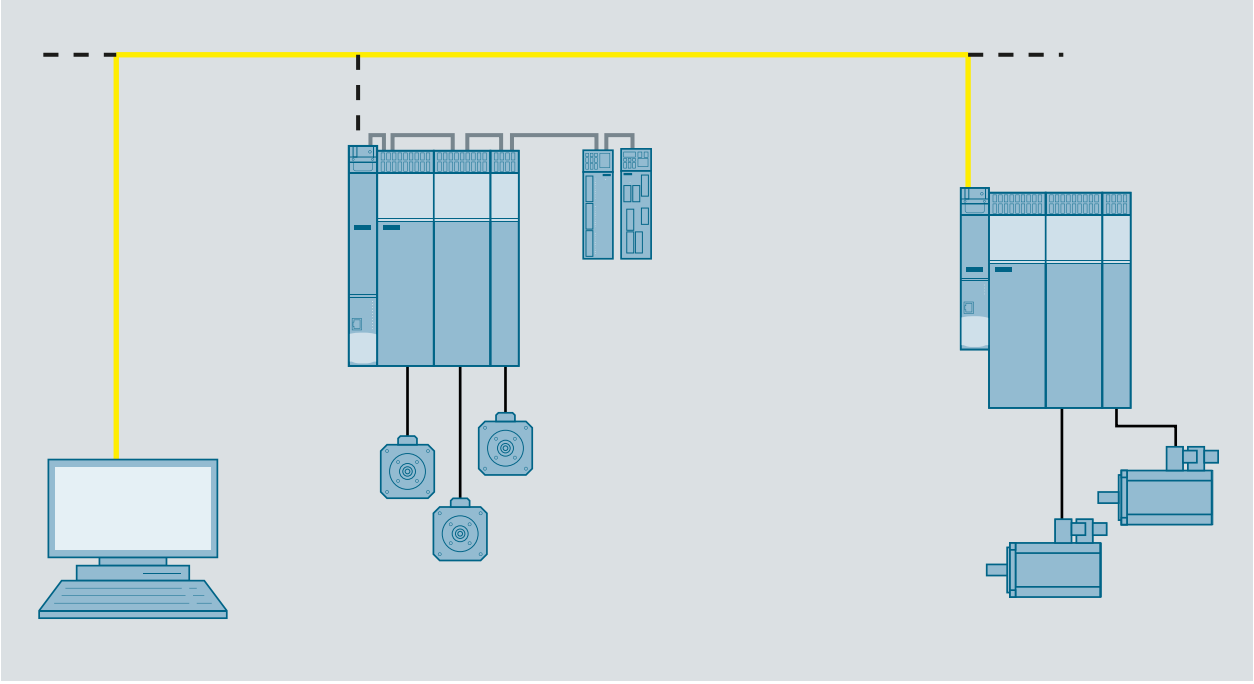


Figure 3-432 S7ONLINE

- The **DEVICE (STARTER, SCOUT)** access point provides the option of connecting SIMOTION SCOUT TIA (either in parallel or alternatively to S7ONLINE) directly to a device, e.g. via the Ethernet interface. In this way, you can communicate quickly, without changing the project settings, with the device either via the system network or via a separate connection in order, for example, to adapt the parameterization or read out the diagnostics information. The DEVICE access point does not exist in the TIA Portal. Thus, the settings for this access point can be configured only in SIMOTION SCOUT TIA.

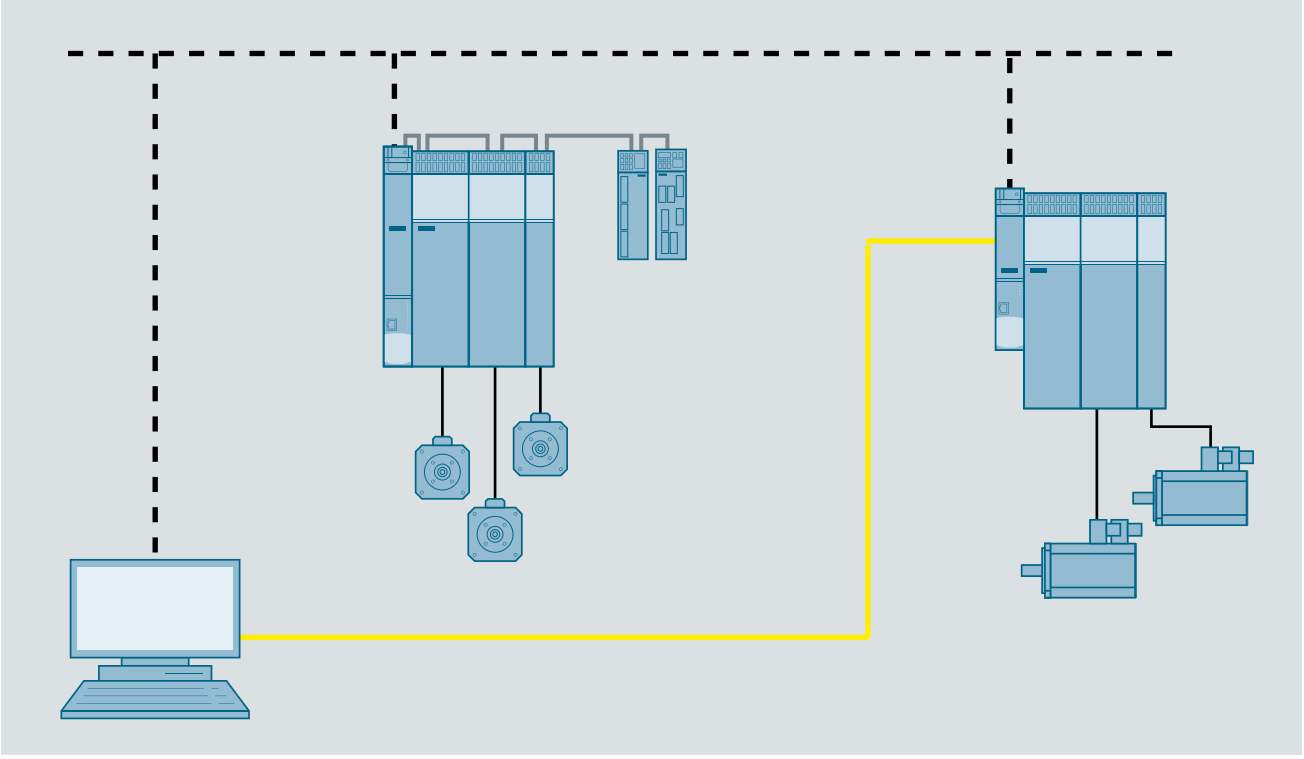


Figure 3-433 DEVICE

It is therefore possible to configure both access points for a device and to go online via these points.

See also

Select target devices (Page 805)

Available nodes

The function "Accessible nodes" in SIMOTION SCOUT TIA enables you to identify devices in a network and go online with them.

Note

The search for accessible nodes assumes that no online connection with an opened project exists.

Displaying accessible nodes

Proceed as follows to identify accessible nodes:

1. Select "Project > Accessible nodes" in the menu.
   All the devices found in the network are listed in the working area below the folder "Accessible Nodes". Type information is also displayed for the devices.

Assigning interface parameterization to an access point

To create a connection between the access point, the interface parameterization, and the interface, proceed as follows:

1. Click the "PG/PC..." button to open the "Set PG/PC Interface" dialog.
2. At "Access point of the application", select the DEVICE access point to which you assign the interface parameterization.
3. Select one of the interface parameterizations to assign it to the access point, modify its properties, copy it, or delete it.
4. Click "OK" to accept the settings.
5. In the working area, click "Update" to restart the search for accessible nodes.

Note

The settings in SIMOTION SCOUT TIA are only accepted for the access point DEVICE. The access point S7ONLINE must be configured in the TIA Portal.

Searching for accessible nodes via the IP address

With TCP/IP interface parameterization you can search for nodes via the IP address. Enter the IP address in the "IP address of the sought node:" field.

For the automatic detection of accessible nodes via a PG/PC interface with TCP/IP, the nodes must be connected to the same physical Ethernet subnet as the PG/PC. Once a device is downstream of an IP router and so located in another physical Ethernet subnet, this device will no longer be automatically detected in "Accessible nodes".

You can also enter an IP address and check whether it is possible to access a node downstream of an IP router with the configuration currently set on the Ethernet adapter located in the PG/PC (IP address / subnet mask).

Displaying additional information

To display additional information about identified devices, proceed as follows:

1. Right-click on a device or drive unit and select, for example, one of the following options from the shortcut menu:
   - "Device Diagnostics"
   - "Operating state..."
   - "Licenses"
   - "Edit Ethernet Node..."
   - "Flashing"

See also

Setting the access point on the PG/PC (Page 804)

Select target devices (Page 805)

Setting the access point on the PG/PC

Note

The data for the PG/PC interface can only be changed when the interface is not online. If several instances of SIMOTION SCOUT TIA have been started, an online connection can only be established from one of the instances. An attempt to go online simultaneously from several SIMOTION SCOUT TIA applications to different CPUs results in errors.

Using S7ONLINE

SIMOTION SCOUT TIA takes the settings for the S7ONLINE access point from the TIA Portal. At "Online accesses", select the following settings:

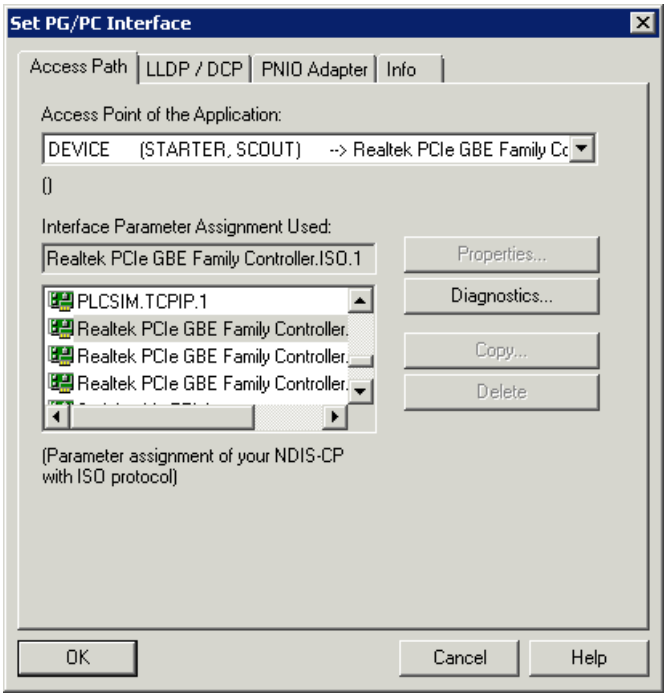
1. PG/PC interface type
2. PG/PC interface
3. Connection with interface/subnet
4. "1st gateway"
5. Device address

The settings for S7ONLINE in the "Set PG/PC Interface" dialog box are not applied.

Setting the DEVICE access point

Proceed as follows to set the DEVICE access point:

1. Select "Options > Set device access interface..." in the menu.
   The "Set PG/PC Interface" dialog box opens.
2. Select the DEVICE access point at "Access point of the application":



3. At "Interface parameterization in use", select the interface with which you want to go online via the selected access point.
4. Confirm your selection with OK.

Select target devices

With the target device selection, you specify whether an online connection to this device will be established and which access point will be used when you perform "Connect to selected target devices".

Procedure

To select a target device, proceed as follows:

1. Select "Target system -> Select target devices ..." in the menu.
   The "Target Device Selection" dialog opens.

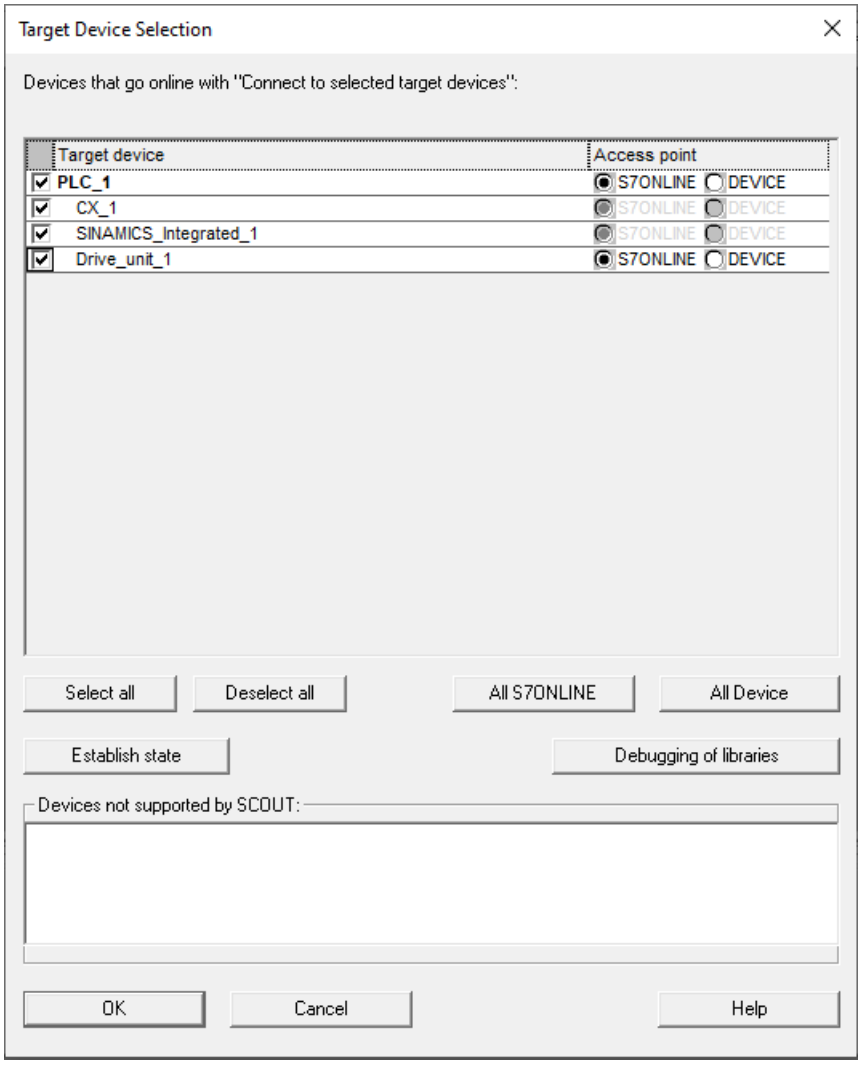


Figure 3-434 Selecting target devices and access points

2. Activate the checkbox for the respective device.
3. Click the "Establish state" button.

- Establish a connection to a device that was deselected when going online and that has been subsequently selected.
- Separate a connection to a device that was selected when going online and that has been subsequently deselected.

Save project and compile changes

Overview

Note

You can only "save a project and compile changes" in SIMOTION SCOUT TIA.

"Save project" in SIMOTION SCOUT TIA automatically performs a "Save project" in the TIA Portal.

Always cancel the online connection in the TIA Portal before you perform "Save and compile project" in SIMOTION SCOUT TIA.

To download a project to the target system, it must first be saved and compiled into an executable code. Only the compiled code can be downloaded to the target system and executed.

By default, "Save and compile" is performed automatically for a download if the appropriate option is activated under "Options > Settings..." in the "Download" tab.

Save project

With "Save project", the project is saved to the hard disk. The changes are accepted into the project. No further processes (such as compilation or consistency checking) are triggered for the project.

Procedure

To save a project, proceed as follows:

1. Select "Project > Save" in the menu.

Save project and compile changes

"Save project and compile changes" searches through the entire project for changes. If a source is found which has been changed or has no compilation results, this and any linked sources are compiled and saved (e.g. during an FB call). Therefore only the changes are compiled.

Procedure

To save and compile a project, proceed as follows:

1. Select "Project > Save and compile changes" in the menu.
   Or click the "Save project and compile changes" button in the toolbar.

The compilation run is logged in the detail view of the workbench. Information, warning and compilation errors are shown there in plain text.

Saving projects and recompiling all

With "Save and recompile all", all sources of the entire project are recompiled. The command is suitable if you are quite sure that all the old data from older versions should be removed and replaced with new compilation results. The function comprises the following steps:

- Project-wide deletion of all compilation results
- Recompilation of all objects

Procedure

To save and recompile data, proceed as follows:

1. Select the "Project > Save and recompile all" menu command.

The compilation run is logged in the detail view of the workbench. Information, warnings and compilation errors are shown there in plain text.

Check consistency

Overview

Use "Check consistency" to check the current configuration and parameterization of the project for formal errors in the data.

This allows you to determine that the programmed configuration is consistent with the defined hardware configuration.

Procedure

To check the consistency of a project, proceed as follows:

1. Select "Project > Check consistency" in the menu.

The results of the consistency check are logged in the detail view of the workbench in the "Compile/check output" tab.

Connect to selected target devices

Overview

Use the "Connect to selected target devices" function to go online with the device connected to the PG/PC. You can send and receive data.

Requirements

- You have created a subnet for the PG/PC interface in the TIA Portal.
- You have assigned the PG/PC interface to the subnet in the TIA Portal (see also Setting up the PG/PC communication (Page 661)).

- In the TIA Portal, you have loaded the hardware configuration into the SIMOTION device (see also Setting up the PG/PC communication (Page 661)).
- You have disconnected the online connection in the TIA Portal.

Establishing an online connection

To establish an online connection, proceed as follows:

1. Select "Project > Connect to selected target devices" in the menu.
   Or click the "Connect to selected target devices" button in the toolbar.
   The "Target Device Selection" dialog opens.
2. In this dialog, check the settings of the target device selection and the access points and confirm with "OK".
   Devices not selected in this dialog but still online will not be disconnected from the target system.

Establishing a selective online connection

To establish a selective online connection, proceed as follows:

1. Select a CPU or a drive in the project navigator.
2. Perform "Connect target device" in the context menu to establish a specific connection to this drive.

Result

- The connection will be established and the project checked for consistency.
- The footer area indicates that the system is online.
- The detail view shows the drives control panel.
- In the project navigator, the status of the online connection is indicated in a color by the connector icons.

Consistency display (connector icons)

Each element in the project navigator has its characteristic icon. In online mode, the icons are highlighted in color or additional icons (connector icons) are displayed which are characteristic for the PG/PC to target system connection.

The following table provides an overview of the meaning of the various icons.

Icon	Description
	The SINAMICS drive unit / element of a drive unit (DO) is in online mode. There is an inconsistency in terms of the data contents (parameter values) of the drive unit / element. SINAMICS drive unit / element of a drive unit (DO) is in online mode. There is an inconsistency in terms of the data contents (parameter values) of the drive unit / element. Possible reasons for the inconsistency: • At least one parameter value differs between the project data and the actual values in the target system. • Sequential parameterization: After a parameter assignment has been downloaded, sequential parameterization may be required once data has been received in the drive. In this case, you must perform an upload for alignment purposes. Note: • If the inconsistency is displayed by means of the icon in the project navigator, you can display the inconsistency in detail using the object comparison. Depending on the result, you can then decide whether or not an upload or download is necessary. • If this icon is displayed, it may be that, depending on the mechanism used, the parameters of a DO that are marked with this icon will be marked as "equal" in the object comparison. The object comparison is based on a direct comparison of all the values for the element and, therefore, provides the correct difference.
	The component (drive object) is in online mode. The component is physically present in the RAM of the drive unit or project data in the main memory of the PG/PC is identical with the project data of the component present in the RAM.
	The component (drive object) is not in online mode. The component is not physically present in the RAM of the drive unit or the connection to the target device is broken.

Icon	Description
	The device was not selected during the change to online mode. There is no connection to this device.
	Device/element is in online mode. There is an inconsistency between the project data of the device/element in the PG/PCs and the actual values of the device/element in the target system. • Possible reasons for the inconsistency – SIMOTION CPUs/TOs/Units: – Configuration data of a technology object changed / technology objects added – Change in a program source, in a global device variable, or in the address list – Change to the default value of a system variable • Possible reasons for the inconsistency – drive unit / DO, e.g. due to: – Change in the name of a DO – Change in the type of drive object (DO) from SERVO -> VECTOR – Selection/deselection of different function modules – Addition/removal of DDS/CDS – Addition of a DCC chart to a drive object (DO) Structural changes have been made to the DO. If this inconsistency is present, integrated online diagnostics cannot be performed on the device/element. Remedy: In the event of this inconsistency, save the project and reload the new project data to the target system. Note: If an inconsistency is displayed by means of the icon in the project navigator, you can display the inconsistency in detail using the object comparison.

No connection could be established to SINAMICS Integrated/drive units

1. Check the messages in the output window.
2. Perform a project download to load the project data to the device and establish a connection. Select "Project > Download to target system" in the menu.

You will find more information in Section Loading data to the target system (Page 814).

Note

You must make sure that you go online via the selected interfaces. If, for example, you have configured the incorrect interface for the S7ONLINE access point, you will not go online. Therefore, if required, check the settings made there.

Additional references

You will find more information in Section "Set interface" of the SIMOTION SCOUT TIA Online Help.

Problems when going online

Not possible to connect to target system

If you cannot go offline with the project despite proceeding as described above, check the following settings:

- Network connections (TIA Portal)
- PG/PC interface access point S7ONLINE (TIA Portal)
- Access point Device (SIMOTION SCOUT TIA)
- Bus address of the devices or IP address (TIA Portal)
- Firmware version of the device (SIMOTION SCOUT TIA)

Subsequently we describe the procedures that are particularly easy to implement.

You have the choice of communicating with the controller via PROFINET/Ethernet or PROFIBUS. These options also provide maximum flexibility in the choice of a teleservice connection (teleservice via PROFINET/Ethernet or PROFIBUS). This access method, which is specified once, enables you to use routing to access other controllers and drives connected to the SIMOTION controller via PROFINET/Ethernet or PROFIBUS.

You can also access the SIMOTION controller via existing routing connections to other controllers, without there being a direct connection between the PG and the SIMOTION controller.

However, these options require that an online access is specified during the initial commissioning. This description explains the procedure for the initial commissioning (or when changing the access settings). When this configuration has been performed correctly and completed, going online then is practically only at the touch of a button.

Only procedures that always and safely result in success are described here. Depending on the initial situation, a different procedure is also possible with sufficient knowledge.

PG/PC interface (access point S7ONLINE) configured in the TIA Portal?

Were you online with the TIA Portal and have you configured the PG/PC interface?

Before you can go online with the SIMOTION SCOUT TIA, you must configure the interface on which you want to go online. Perform this step in the TIA Portal.

Possible sources of error include:

- You are still online with the TIA Portal. Disconnect the online connection.
- You have not assigned the controller to any subnet. Instructions for assigning the controller to a subnet are described in the SCOUT TIA Online help in the TIA Portal.
- IP address and subnet screen of the PG/PC and the controller are not on the same subnet. Adapt the IP address / subnet of the PG/PC in the Windows Control Panel or automatically assign a temporary IP address via the TIA Portal. Associated information is contained in the SCOUT TIA online help in the TIA Portal.
- In the TIA Portal, check with which interface you attempt to go online at "Online & diagnostics".

Before going online with the controller, you must clarify the following questions:

Is the project to be loaded (or consistent project) already on the controller and have no changes been made to the interface settings?

In this case, activate the interface configured in the project and go online in SIMOTION SCOUT TIA using the  button, "Connect to selected target devices". The devices to which a connection is established can be seen under "Target system > Select target devices...".

Have the interface settings been changed?

When do changes have to be made to the interface settings?

- An overall reset has been performed on the controller (MRES).
- The controller has been replaced.
- The project is to be loaded to a new (or unknown) controller.
- Interface settings have been changed in the project (different PROFIBUS baud rate selected; access addresses changed).

Requirement: You have a completed HW configuration and a project that can be downloaded.

Do I require an online connection to all devices/components present in the project?

In online operation, SIMOTION SCOUT TIA tries to perform online operation with all hardware components contained in the project. This means that the time needed for going online increases. We recommend that you make settings for SIMOTION SCOUT TIA so that online operation is made only with those components currently needed. The setting can be found with "Target system > Select target devices..." in the menu. The selection and deselection of the devices in the online state can be made from the "Online connection" context menu on the device.

This procedure is also advantageous for SIMOTION D when configuration of the SINAMICS Integrated is complete. Without completely going offline with all hardware components, the connection may simply be deselected on the object SINAMICS_Integrated via the context menu.

See also

Setting up the PG/PC communication (Page 661)

Troubleshooting

If you still cannot go online with the project despite proceeding as described above, check the settings of CX32-2.

To be able to go online on a CX32-2, at least the hardware configuration (HWCN) must be loaded to the D4x5-2 and the CX32-2 connected without errors via the configured DRIVE-CLiQ ports on the D4x5-2.

Online access to the drives is not possible if HWCN is not loaded at the time you initially connect to the target system.

Loading data to the target system

Overview

Note

Loading data to the target system can only be done in SIMOTION SCOUT TIA.

You have to download the project data that you have created with SIMOTION SCOUT TIA to the target system.

The target system can contain several CPUs, e.g. SIMOTION controllers or drives. The project data contains the programs (e.g. MCC) that you have created and compiled, the hardware configuration and the technology packages that you have created and parameterized.

You must download a project to the target system if you have made the following changes:

- You have created or changed programs.
- You have changed definitions of global variables or symbolic I/O variables.
- You have made changes in the execution system.
- You have inserted or changed technology objects.
- In the TIA Portal, you have loaded the hardware configuration of a SIMOTION controller with the "Upload device as new station (hardware and software)..." function into a project.

Requirements

The following conditions must be met before the project can be downloaded to the target system:

- All ports (cables) must be plugged in and the interfaces configured
- All source codes are compiled without errors.
- The project is consistent.
- SIMOTION SCOUT TIA is in online mode.

Responses in the event of an error

The download is canceled in response to a fault: Information can be found in the "Target system output" tab.

Download variants

You can either load the entire project to the target system or perform a download for a specific device:

- Download a project to the target system (all target devices)
- Download CPU / drive unit to target device
- Download a program subset and single units (sources) to the target device

Additional references

You will find detailed information on downloading in the SIMOTION Runtime Basic Functions Function Manual and in the SIMOTION SCOUT TIA Online Help at "Downloading data to the target device".

See also

Load project to the target system (Page 815)

Load CPU / drive unit to target device (Page 817)

Downloading a program subset and single units (sources) to the target device (Page 819)

Load project to the target system

Note

You can only perform a project download in the STOP operating state and for all target devices with which you are ONLINE.

Procedure

To load a project to the target system, proceed as follows:

1. Select "Project -> Download to target system" in the menu or click the "Download project to target system" button in the toolbar.
   The "Download to Target System" dialog opens.

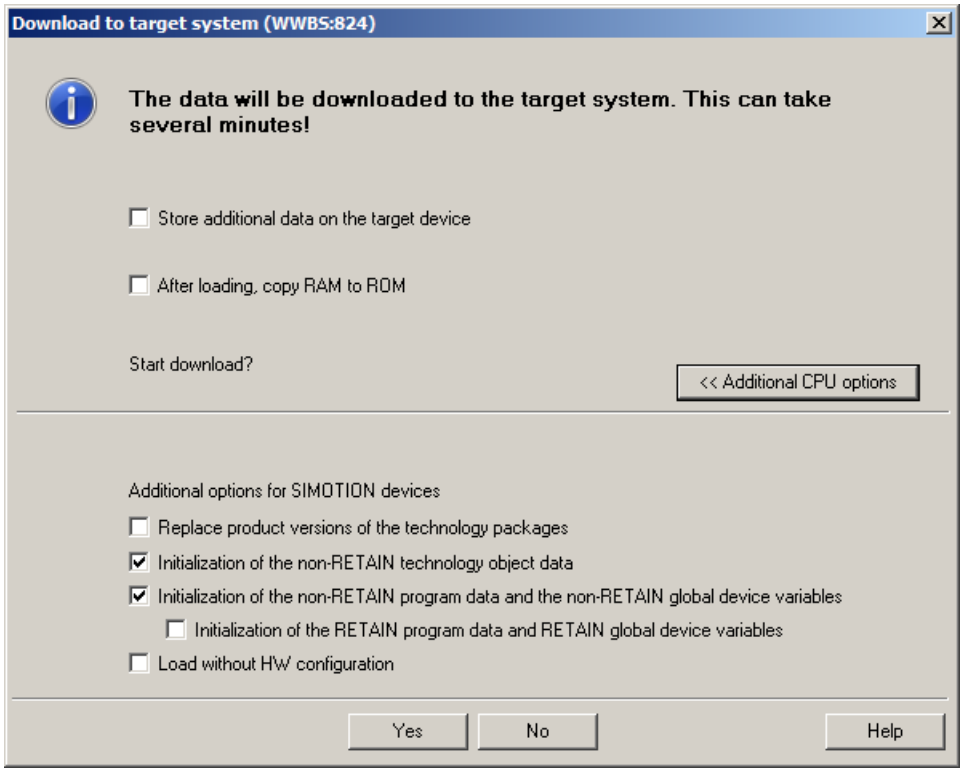


Figure 3-435 Download to target system

   Optionally activate the following options, depending on the device:

   – Store additional data on the target device
   – Copy RAM to ROM after download

2. Click the "Additional CPU options" button to display the following options:
   – "Replace product versions of the technology packages"; this loads the technology packages selected in the "Select Technology Packages" dialog to the target system.
   – "Initialization of the non-RETAIN technology object data"; only non-RETAIN data can be initialized for TOs. This does not affect the program data (variable values).
   – "Initialization of the non-RETAIN program data and the non-RETAIN global device variables"
     The "RETAIN program data and the RETAIN global device variables" can also be initialized. This does not affect the technology object data. The global device variables are initialized together with the settings for the program data (variable values) - only works in STOP mode.
   – Load without HW configuration

3. Select additional options when required.
4. Click "Yes" to start the download.

Note

Global download settings

You make the global presetting in SIMOTION SCOUT TIA under "Options > Settings... > Download".

Note

Default names in the station manager and offline project for SIMOTION P

Note that the default names in SIMOTION SCOUT in the station manager and the offline project must match.

If you have performed a download from SIMOTION SCOUT TIA and then want to start a download from SIMOTION SCOUT, make sure that the default names in the station manager and the offline project match.

Result

The project data has been loaded to the target device and an online connection established.

If the connection was successful, the connector icons are highlighted in green.

Load CPU / drive unit to target device

In addition to a project download, you can also selectively load data to each CPU / drive unit. The standard procedure is the download in STOP operating state. However, under certain circumstances you can also perform a download in RUN operating state.

Procedure

Proceed as follows to load selective data to a CPU / drive unit:

1. First select the device to which you want to load the data in the project navigator. The device must be online.
2. Click the "Download CPU / drive unit to target device" button.
   Or right-click the device in the project navigator and select "Target device > Download..." in the context menu.
   The "Download" dialog opens. The contents depend on the device for which you want to download data.

Loading the CPU to the target device

A download can be performed for the entire CPU (without drive unit) but also separately for individual related download units.

- Download CPU without drive unit
- Download all technology objects of the CPU
- Download all programs of the CPU

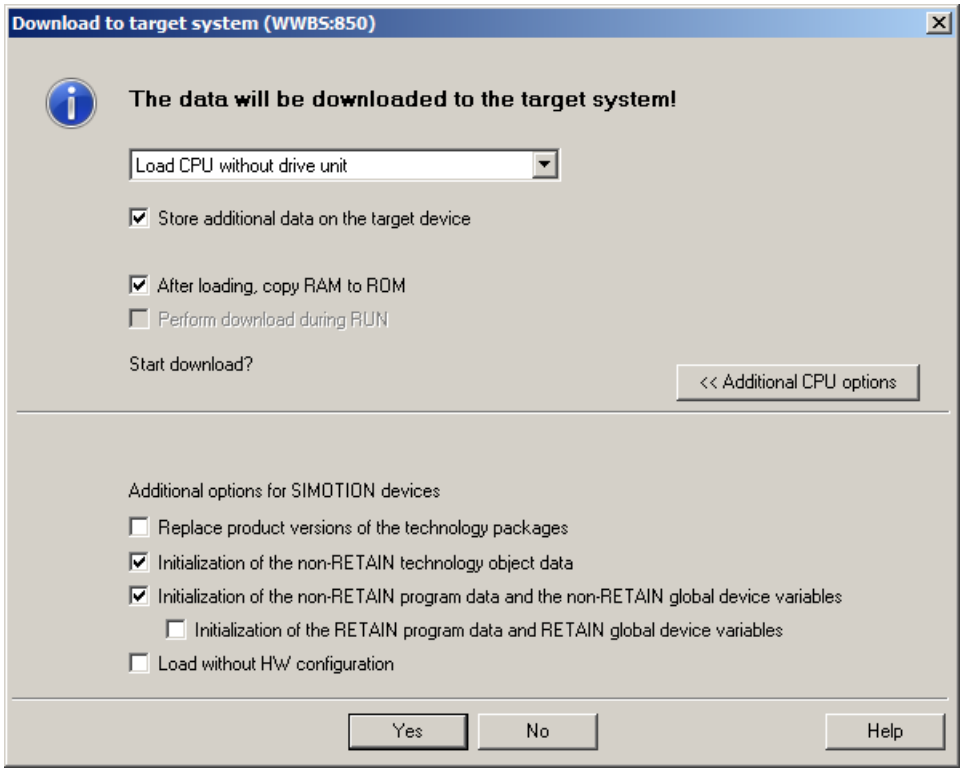


Figure 3-436 Download to CPU / drive unit

You can select the following options:

- "Store additional data on the target device"; you can use this option to store additional data, e.g. program sources, on the target device.
- "After loading, copy RAM to ROM"
- "Perform download during RUN"; enables a download to be performed in RUN operating state.

Click "Additional CPU options" to display the following options:

- "Replace product versions of the technology packages"
- "Initialization of the non-RETAIN technology object data"
- "Initialization of the non-RETAIN program data and the non-RETAIN global device variables"
- "Load without HW configuration"

Downloading the drive to the target device

For drives, you can only download the drive data (parameterization, etc.) to the drive unit.

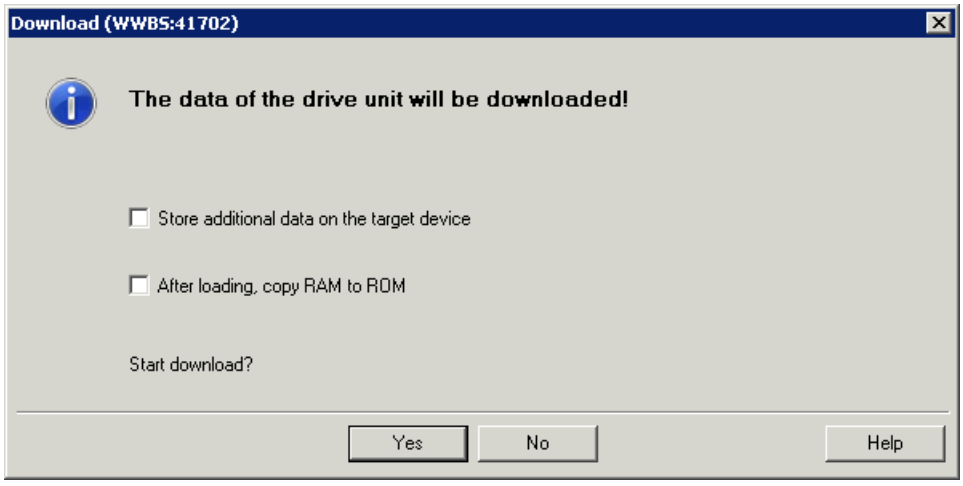


Figure 3-437 Download to drive unit

By default, the download is performed in STOP. After a successful download, the CPU can be restored to its previous state.

You can select the following options:

- "Store additional data on the target device"; you can use this option to store additional data, e.g. program sources, on the target device.
- "After loading, copy RAM to ROM"; copies RAM to ROM to the selected device once the download is complete.

Additional references

You will find detailed information on downloading in the SIMOTION Runtime Basic Functions Function Manual and in the SIMOTION SCOUT TIA Online Help at "Downloading data to the target device".

Downloading a program subset and single units (sources) to the target device

Download of selected units

You can load single or multiple units to the SIMOTION device in the RUN and STOP operating states and in this way specifically load only a selected subset of programs.

This enables, for example, two users who are connected online to the target device at the same time with SIMOTION SCOUT TIA to download separate program subsets independently.

To load selected units to the target device, proceed as follows:

1. In the project navigator, right-click the unit and select the "Download" command in the context menu.

Additional references

You will find detailed information on downloading in the SIMOTION Runtime Basic Functions Function Manual and in the SIMOTION SCOUT TIA Online Help at "Downloading data to the target device".

Loading data from the target device

Load to PG

With Load to PG, the SIMOTION SCOUT TIA data is loaded from the target device to the PG/PC. In this way, SIMOTION SCOUT TIA data can be loaded from the target device to the existing SIMOTION SCOUT TIA offline project on the PG/PC.

Load the CPU / drive unit to the PG

- Select this option if you want to load all the project data from the target device to the PG. Note the information in the dialog.
- Select the "Save before loading to PG" option if you want to save the project on your PG before loading.
- Select the "Overwrite available libraries" option if you want to overwrite the libraries of the project available locally on your PG.

Only load configuration data to PG

- Select the "Only load configuration data to PG" option if you want to load the current configuration data present in the target device to the PG.

Transfer current data to RAM

If you want to load the current data present in the current data memory of the target device to the RAM before Load to PG, activate the "Transfer current data to RAM" option. If you do not transfer the current data to the RAM, it is not loaded to the PG.

When "Load to PG" is executed, adapted values for configuration are detected and "Transfer current data to RAM" is also automatically preselected.

Load to PG via project comparison (object granular)

With the project comparison functionality, you can also load the data for individual objects to the PG.

Additional references

You will find detailed information in Section "Loading data from the target device" of the SIMOTION SCOUT TIA Online Help.

Detailed information on the project comparison can be found in the SIMOTION SCOUT TIA Online Help and in the SIMOTION Project Comparison Function Manual.

See also

Device upload (Page 947)

Load to file system

Select "Edit > Load to file system..." to save data for a device to a memory card/CF card directly via a card adapter on the PC. Alternatively, the device data can also be saved locally on a hard disk in order to copy it to a memory card / CF card at a later point in time. First delete the USER directory on the card as otherwise inconsistencies can occur. The directory structure of the versions of the hard disk and CF card may not be the same and data will not be overwritten.

Note

Kernel version and device type

A project is always created for exactly one kernel version and one device type and will only run on the relevant SIMOTION device if this exact kernel version and this device type are available. If this is not the case, the SIMOTION device remains in the STOP operating state.

Note

Saving only possible in offline mode

To be able to load data into the file system, SIMOTION SCOUT TIA must be in OFFLINE mode and the device must be selected in the project navigator.
When the project data is saved, an executable project results.

Additional references

Detailed information can be found in the SIMOTION SCOUT TIA Online Help.

Disconnect from target system

Use "Disconnect from target system" to switch to offline mode. The device is disconnected from the PC/PG.

Disconnecting from the target system

To disconnect the connection to the target system, proceed as follows:

1. Select "Project > Disconnect from target system".
   Or click the "Disconnect from target system" button in the toolbar.

Result

The connection is interrupted and SIMOTION SCOUT TIA goes into offline mode.

Disconnecting a device

To disconnect a device selectively, proceed as follows:

1. In the project navigator, select the drive that you want to disconnect.
2. Select "Disconnect target device" in the context menu.

Result

The drive goes offline.

If this is the last drive that was online, SIMOTION SCOUT TIA goes into offline mode.

3.5.8.3 Selecting technology packages

The technology packages (e.g. TP CAM, TP PATH) for SIMOTION are available for installation in various versions.

You can only use the functions of the technology objects selected if the technology objects are available in the target system.

You can select the technology packages and their versions (as well as the service packs and hotfix versions) for each SIMOTION device.

In the "Select Technology Packages" dialog, you can adopt a more selective approach when choosing the technology packages you want to use. To open this window,

Procedure

To select technology packages, proceed as follows:

1. Select a SIMOTION device in the project navigator.
2. Select "Select technology packages..." in the context menu.
   The "Select Technology Packages" dialog opens.

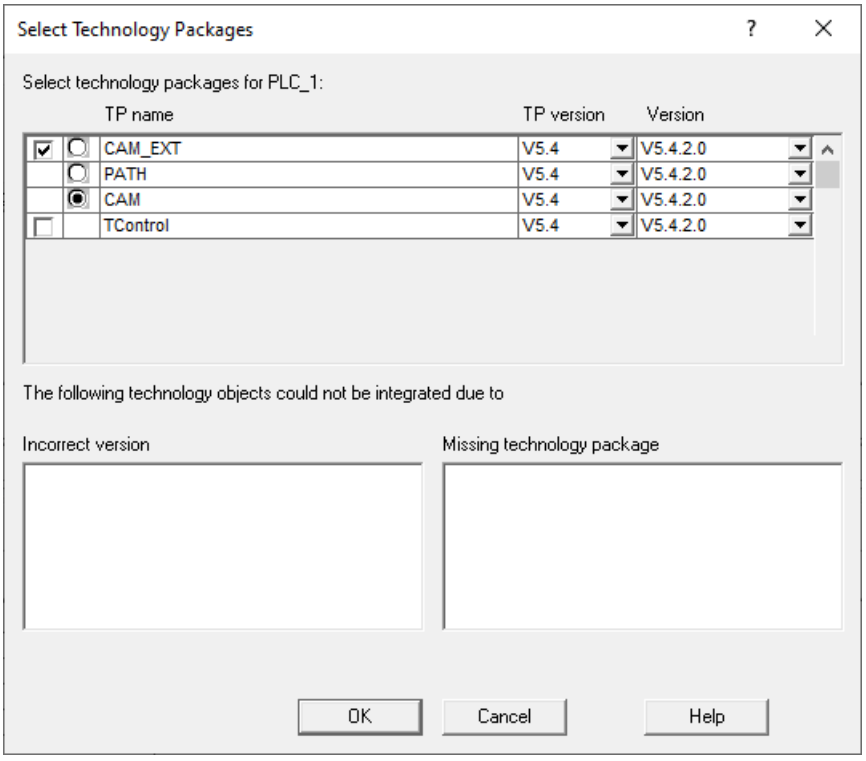


Figure 3-438 Selecting technology packages

3. Select the technology package, version and product version.
4. Confirm with "OK".

Note

Device diagnostics can provide information on which technology package product version has been loaded to a CPU.

Loading technology packages to the target device

Technology packages are only loaded to the target device if no technology package has been loaded previously. If a technology package version changes, the technology package must be explicitly reloaded to the target device.

To download a changed technology package to the target system, proceed as follows:

1. In the "Project" menu, select the "Download to target system" command.
   The "Download to Target System" dialog opens.
2. Click the "Additional CPU options" button.
3. Activate the "Replace product versions of the technology packages" option.
4. Confirm with "OK".

Additional references

You will find detailed information on downloading in the SIMOTION Runtime Basic Functions Function Manual and in the SIMOTION SCOUT TIA Online Help at "Downloading data to the target device".

3.5.8.4 Commissioning the drives

Overview

Note

Creating a SIMOTION drive

You configure a SIMOTION drive in SIMOTION SCOUT TIA. A SIMOTION drive in the current version of SIMOTION SCOUT TIA is a drive of the SINAMICS S120 type (interconnection via PROFINET or PROFIBUS) interconnected with a SIMOTION CPU. You will find the SIMOTION drives you can insert explicitly in Section Supported devices (Page 584). These SIMOTION drives can be configured and parameterized only under these conditions (interconnection with SIMOTION CPU).

You can configure the drive both in offline and online modes.

After configuring the drive, you can test the drive with the drive control panel.

Configure drives online

Requirements:

- You have created a drive in the project.
- The project has been downloaded to the target system.
- SIMOTION SCOUT TIA is in online mode.
- The drive is not being used by a current project (TO axis linked to this drive) in the RUN state.

Procedure

To configure drives automatically, proceed as follows:

1. In the project navigator, navigate to the SINAMICS drive and double-click "Automatic configuration".

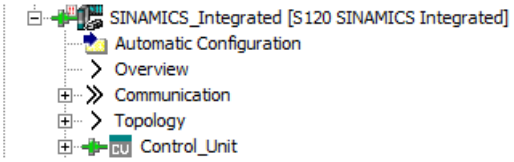


Figure 3-439 Starting automatic configuration

The "Configuration" dialog opens.

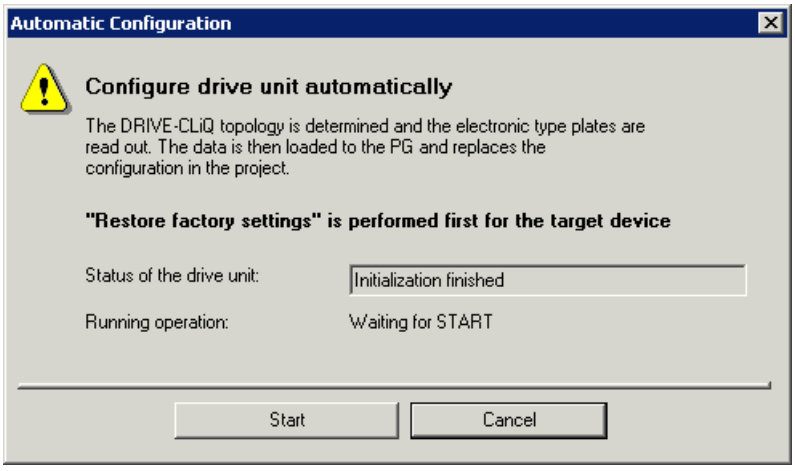


Figure 3-440 "Automatic Configuration" dialog

2. Click the "Start" button and reset the device configuration in the following "Restore Factory Settings" dialog.
   The confirmation prompt appears if the drive unit is not in the "First commissioning" state.

3. In the "Automatic Commissioning" dialog, you can specify whether you are using a drive object of the type "servo" or "vector". Confirm your selection with "Create". Automatic configuration is started.

   Note

   Firmware update

   If the firmware version on the DRIVE-CLiQ components is different to that on the CF card, a firmware update is performed automatically at this position.

   In this case, proceed as follows:

   - Wait for the procedure to finish. This can take several minutes.
   - Go offline.
   - Switch the power supply to the SIMOTION device off and on again.

   As soon as automatic commissioning has finished, SIMOTION SCOUT TIA carries out an upload. With this upload, the configuration data of the component is uploaded to the SIMOTION SCOUT TIA project.

4. Close automatic configuration and select "Stay ONLINE" to test the drive with the drive control panel.

Result

The drive is displayed in the "Drives" folder in the project navigator.

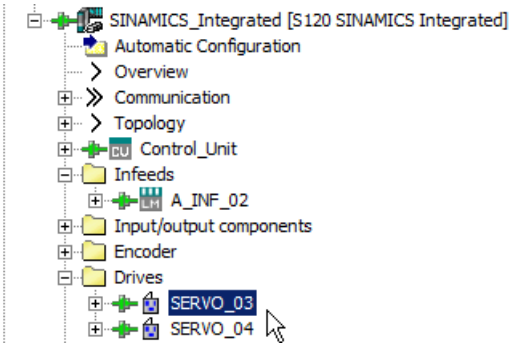


Figure 3-441 Drives inserted

Infeed

- If an infeed with DRIVE-CLiQ connection has already been created, the ready signal of the infeed (r0863.0) is automatically interconnected with "Infeed operation, p0864" of the drive when drives are inserted (only applies to drives that are attached to the same drive unit as the infeed).
- If you are using an infeed without a DRIVE-CLiQ interface, e.g. a Smart Line Module, you must wire the ready signal of the infeed via terminals, see Section Configuring the infeed (Page 831).

Configure a drive offline

The drive is configured in offline mode in two steps:

1. Insert a drive into the project.
2. Configure the drive with the Wizard.

Procedure

To insert a drive, proceed as follows:

1. In the project navigator, navigate to the drive unit and double-click "Configure drive unit".

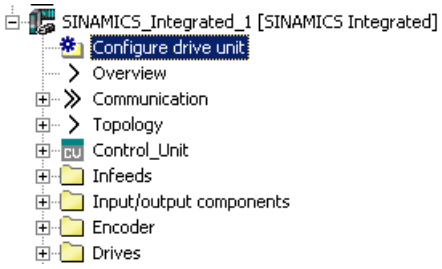


Figure 3-442 Configuring a drive unit

The "Configuration - Option Module" dialog opens.

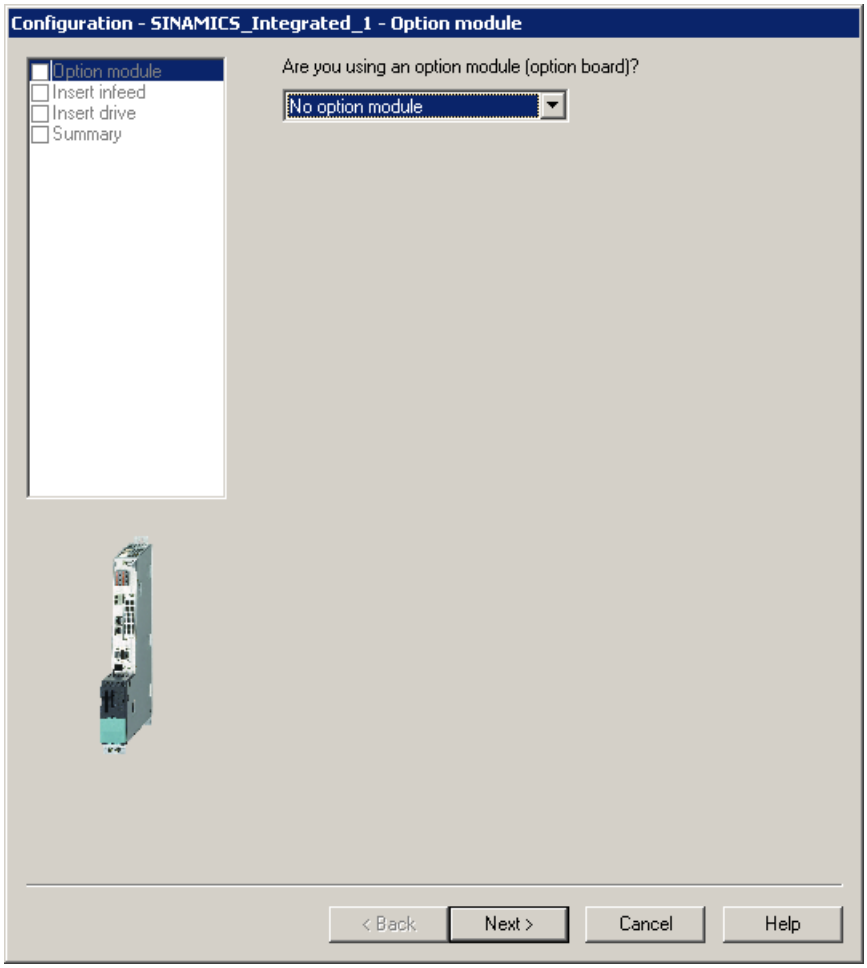


Figure 3-443 Option module configuration

2. Select the option module used at "Option module" when required. Confirm with "Next>".

3. At "Insert infeed", select whether you want to use an infeed with or without DRIVE CLiQ interface.
   - If you are using an infeed with DRIVE-CLiQ interface, the ready signal of the infeed is interconnected automatically.
   - If you are using an infeed without a DRIVE-CLiQ interface, e.g. a Smart Line Module, you must wire the ready signal of the infeed via terminals, see Section Configuring the infeed (Page 831).
4. At "Insert drive," confirm with "YES" that you would like to create a drive. Click "Continue>". The "Configuration - Drive Properties" dialog opens.

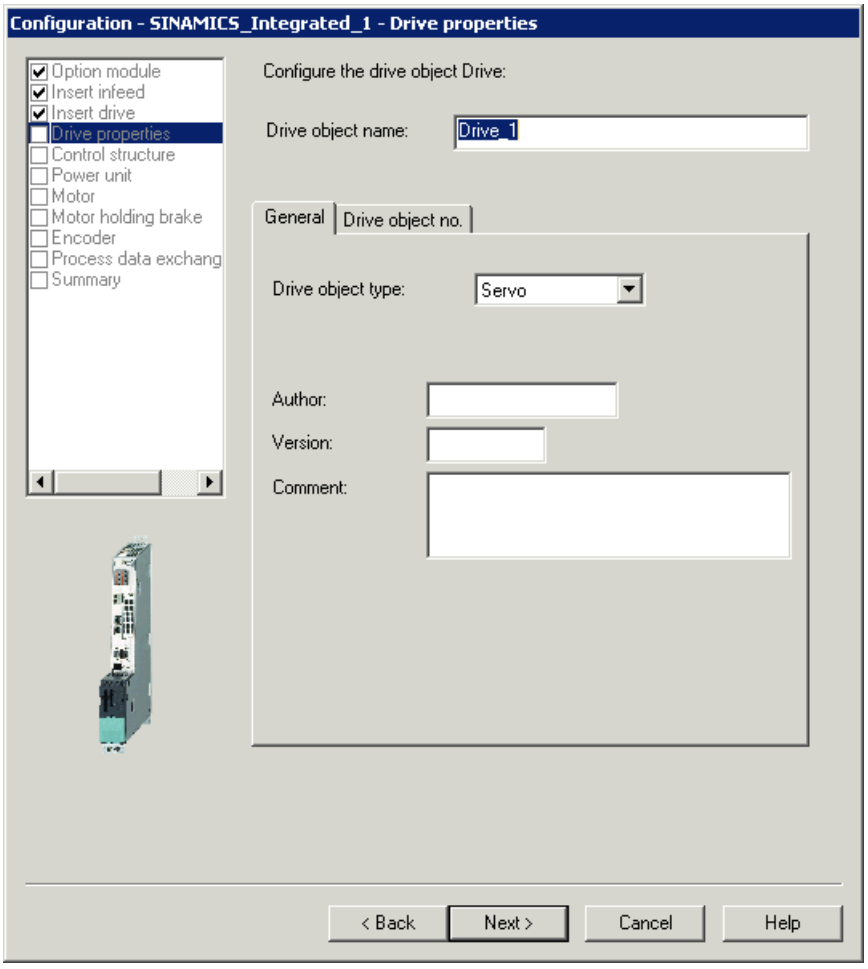


Figure 3-444 Configuring drive properties

5. Assign a name for the drive. Optionally, you can enter Author, Version, and Comment.
6. Confirm with "Next>".
7. At "Control structure", specify the control mode and the function modules depending on the operating mode (servo or vector). Confirm with "Next>".
8. At "Power unit", select the required power unit. Further narrow down your selection of possible power units by using the prescribed filter criteria supply voltage, cooling method, type. Confirm with "Next>".

9. If you are using an infeed without a DRIVE-CLiQ interface, you must manually interconnect the ready signal of the infeed in the next step "Power unit BICO". In the example, parameter p0864 is set to 1 (high) to simplify matters.
   From the perspective of the drive, the infeed is thus always ready, regardless of its actual state of readiness.
10. At "Motor", define the motor for the SINAMICS drive. Confirm with "Next>".
11. At "Motor holding brake", you can configure the motor holding brake of the selected motor. Depending on the motor, different parameters are displayed. Confirm with "Next>".
12. At "Encoder", specify the encoder that you want to use. This step is skipped if you selected speed control without encoder as the control type. Confirm with "Next>".
13. Adopt the preset option at "Process data exchange" and confirm with "Next>". The telegram configuration is done automatically.
14. Once you have worked through all the steps of the configuration, the configured configuration is displayed once again for you to confirm.
    Close the dialog with "Finish".

Result

The drive is displayed in the "Drives" folder in the project navigator.

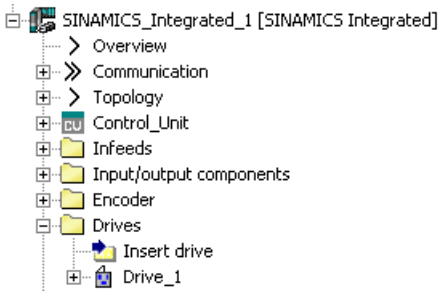


Figure 3-445 Drive inserted

Change configuration

To change the configuration of a drive that has already been configured, proceed as follows:

1. Open in the project navigator the folder of the drive whose configuration you want to change.
2. Double-click "Configuration".
3. Click "Configure DDS" or "Wizard", depending on the drive, to start the drive wizard.

Download the drive configuration to the target system

Save and compile the changes and load the drive configuration to the target system to be able test the drive function with the control panel in the next step.

Additional references

You will find detailed information in Section "Configure SINAMICS drives" of the SIMOTION SCOUT TIA Online Help.
Further information on the interconnection of the ready signal can be found in Section Configuring the infeed (Page 831).

Configuring the infeed

Configuring an infeed without a DRIVE-CLiQ interface

An infeed without DRIVE-CLiQ interface provides the ready signal (p0863.0) via an output terminal. In the project, you specify on which input (r0722) of SINAMICS Integrated the signal is active. The drive supplied by the infeed uses the signal as a ready signal (p0864).

Note

Infeed ready

A drive on SINAMICS Integrated of a SIMOTION D can only be moved if the infeed is ready.

Interconnecting the ready signal of the infeed

Requirements

- You have configured a drive.
- SIMOTION SCOUT TIA is in online mode.

Procedure

In the next step, in order to wire the ready signal of the infeed (terminal "DO: Ready" of the infeed) with the DI 0 of the D4x5-2, proceed as follows in case you have not already established the interconnection during processing using the drive wizard:

1. Navigate in the project navigator to the "Drives" folder and double-click "Configuration" below the drive.

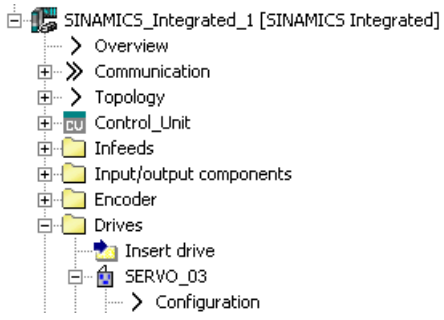


Figure 3-446 Configuring a drive

2. Click the "Configure DDS" button with a yellow background in the working area.

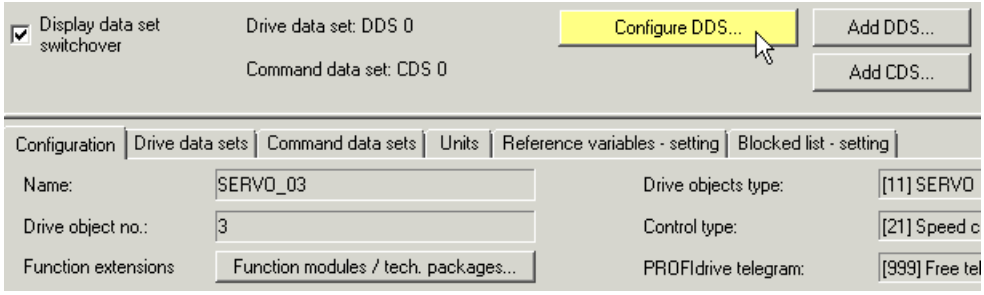


Figure 3-447 Configuring DDS

The drive wizard opens.

3. Click "Next" in the drive wizard until you reach the "Configuration - SINAMICS_Integrated - BICO Power Unit" dialog.
4. In the input field "p0864", select the digital input (e.g. DI 0) to which the ready signal of the infeed is wired.

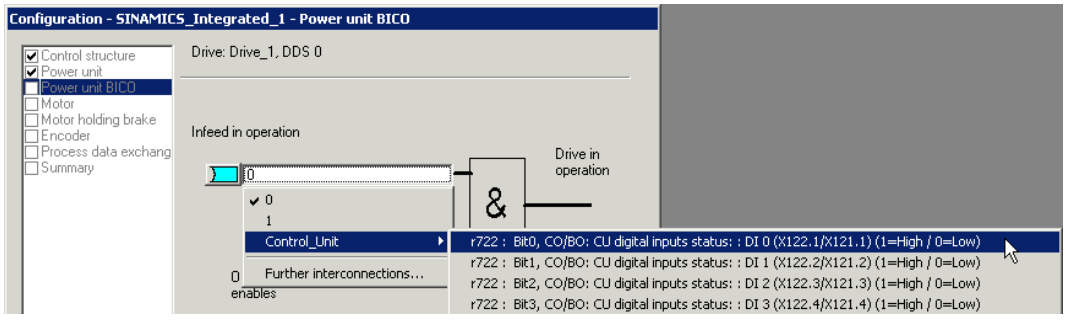


Figure 3-448 BICO interconnection

5. Click "Next". Run through all other dialogs without change until the final "Summary" dialog.
6. Click "Finish".
Thus, the configuration is finished.

See also

Configure a drive offline (Page 827)

Configure drives online (Page 824)

Testing the drive with the drive control panel

Drive control panel

The drive control panel is used to control and monitor individual drives. With the drive control panel, you can traverse drives and can:

- Test each part of the system individually before the drives are traversed in a coordinated manner by means of a program.
- Test in a fault situation whether the individual drives can be traversed by the drive control panel at all, or whether there are already problems here.

⚠ WARNING

This software may be used only when the associated safety instructions are observed! The non-observance can cause injury to persons or damage to material!

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns from the higher-level control have no effect.

The **Stop with preconfigured braking ramp** function is not guaranteed in all operating states. Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Requirements

- You have created and fully configured a drive in the project.
- You have stored and compiled the project. The project is consistent.
- SIMOTION SCOUT TIA is in online mode.
- The project with the drive parameterization has been loaded to the target system.

Opening the drive control panel

To open the drive control panel, proceed as follows:

1. In the project navigator, navigate to the drive unit and open the "Drives" folder.
2. Open the "Commissioning" entry and double-click "Control panel".
   The drive control panel opens in the detail view.

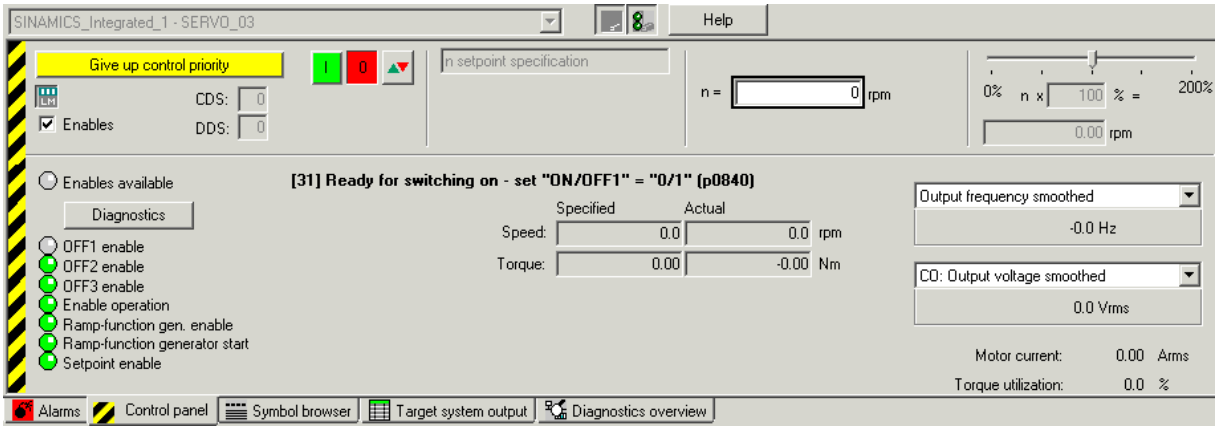


Figure 3-449 Drive control panel

Using the drive control panel

Assuming control priority

To assume control priority, proceed as follows:

1. Click the button to show the control area to test the motor speed specified at "n=".
2. Click the button to show the diagnostics area to display and evaluate defined parameters. You can determine the status of the required enables with the "Diagnostics" button.
3. In the selection list, select the drive that you want to control or monitor using the PG/PC, "SINAMICS_Integrated_1 - SERVO_03" in the example.
4. Specify the parameters that you want to display in the selection lists on the bottom right-hand side of the diagnostics area. The values are displayed below.

5. Select "Assume control priority" to establish the connection to the PG/PC. The "Assume Control Priority" dialog opens.

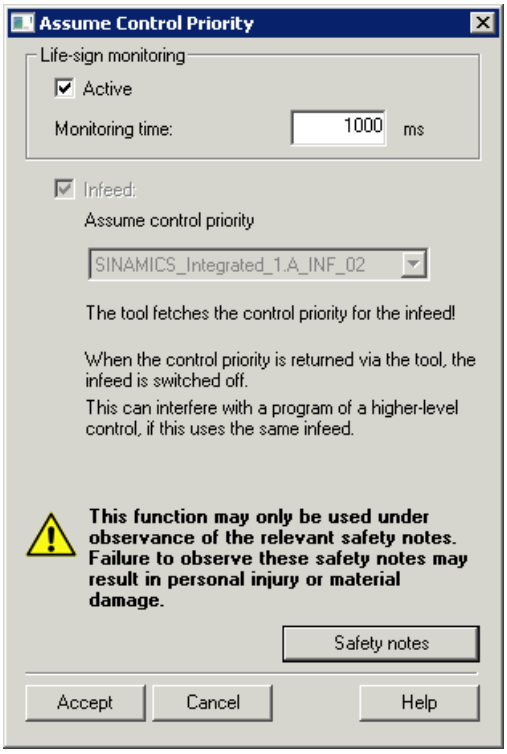


Figure 3-450 Assuming control priority

6. Consider the following infeed options for the type of connection you are using.
   - If you are using an infeed with a DRIVE-CLiQ interface, select the infeed for which the control priority is to be assumed at "Infeed" in the "Assume Control Priority" dialog. Activate the "Infeed" option if the infeed control priority is to be assumed and activated.
   - If you are using an infeed without a DRIVE-CLiQ interface, you will have to interconnect the "Infeed operation" signal (drive parameter p0864) yourself. In this case, the dialog does not contain any fields for selecting infeed.
   - If the signal of the "Closed-loop control operation" infeed is already BICO interconnected with the drive, the infeed is permanently specified. No selection is possible in the dialog, the fields are grayed out.
7. Read the safety instructions and confirm these with "Accept".

The PG/PC has the control priority; the label of the button has changed to "Return control priority".

Note

Stopping the drive

You can also stop the drive at any time by pressing the SPACEBAR.

Activating the infeed

To switch on the infeed, proceed as follows:

1. Click the button for more information.

Activating enables

To activate the enables, proceed as follows:

1. Activate the "Enables" option to enable the drive.
   The ON and OFF buttons and are activated and the enable states are indicated by the LEDs.
   Click the "Diagnostics" button to show any missing enables.

Testing the drive

To test the drive, proceed as follows:

1. Enter a speed setpoint in the "n=" field.
2. Use the slider to set the current speed **setpoint** in relation to the maximum speed **setpoint (100%)**.
3. Click the green switch to start the drive.
   Or click the switch to control the drive in jog mode. The motion continues as long as you keep the switch pressed.
   In the diagnostics area you can monitor the set speed/torque as well as the actual speed/ torque.

Exiting the drive control panel

To exit the drive control panel, proceed as follows:

1. Click the red button to stop the drive.
2. Deactivate the "Enables" option.
   The LEDs are grayed out.
3. Deactivate the infeed. To do this, click "Infeed on/off".
4. Click the "Return control priority" button to return the control priority of the control panel.
5. On the menu bar, select "Project > Disconnect from target system" to switch to offline mode.

Result

The drive can be operated. The drive configuration is complete.

3.5.8.5 Creating and testing an axis

TO axis technology object

Note

You create the technology object (TO) axis in SIMOTION SCOUT TIA.

You can create the axis both in offline and online mode.

Technology objects represent the respective real objects, e.g. a position axis, in the controller.

The technology object axis offers the user a technological view of the drive and the encoder (actuator and sensor) and provides technological functions, such as communication with the drive, actual value processing, position control and positioning functionality. It executes control and motion commands and indicates states and actual values.

Requirements

If you want to insert a real axis, you require a functional drive.

If you want to insert a virtual axis, you do not require a functional drive.

Note

Virtual axis

All axis types can also be virtual axes, i.e. they do not control real drives but are used as auxiliary axes, e.g. as a leading axis for several following axes (line shaft).

Axis technologies

You can use the following axis technologies:

- Speed-controlled axis
  Motion control is performed using a speed specification without position control.
- Positioning axis
  Motions are position-controlled.
- Synchronous axis
  The synchronous axis creates a grouping of the following axis and synchronous object.
- Path axis
  The path axis type can be interconnected with a path object.
  The path object can be used to calculate and traverse a linear, circular or polynomial path in the 2D/3D coordinate system for at least two path axes and up to three path axes.

Axis wizard

SIMOTION SCOUT TIA provides the Axis wizard for the creation of a new axis. Using the wizard, you interconnect the TO axis with a drive.

Note

Axis wizard - restriction

The wizard can only be run through once.

Subsequent changes to the axis configuration are possible in the corresponding dialogs of the TO axis.

Configuring Axes

Procedure

To insert an axis with the axis wizard and to interconnect with drive and encoder, proceed as follows:

1. Navigate to the folder in the project navigator "AXES" and open it by double-clicking.
2. Select "Insert axis".
   The "Insert Axis" dialog opens.

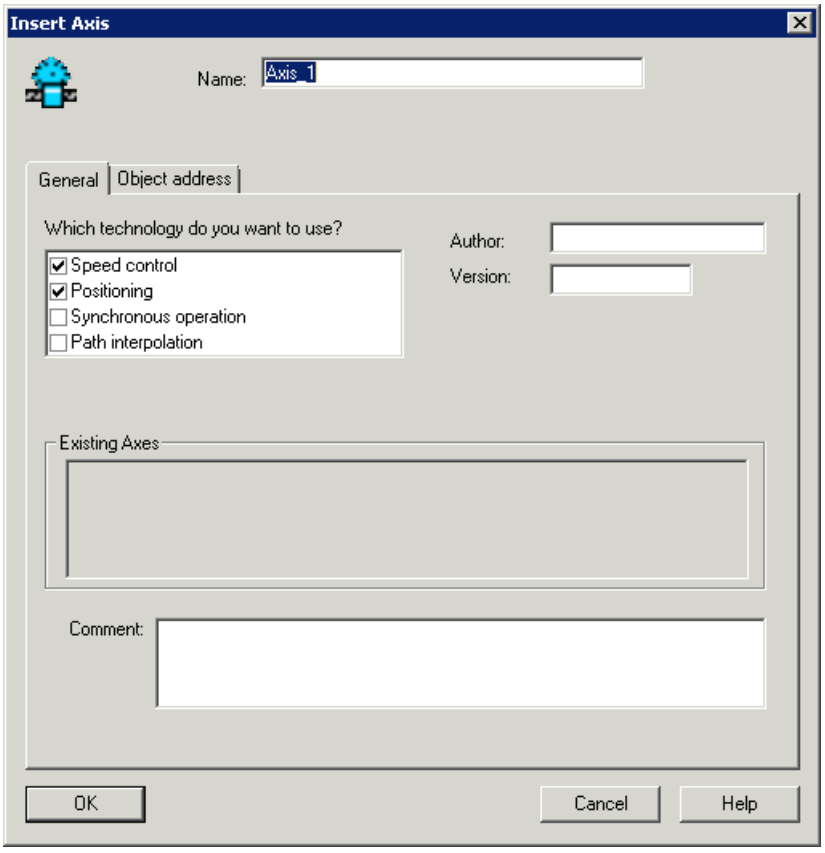


Figure 3-451 Creating an axis

3. Name the axis and select the axis technology that you want to use.
   Note here that the "Positioning" technology always requires the "Speed control" technology. Thus, this technology cannot be deactivated.

4. Click "OK" to exit the dialog.
   The "Axis Configuration - Axis Type" dialog opens.

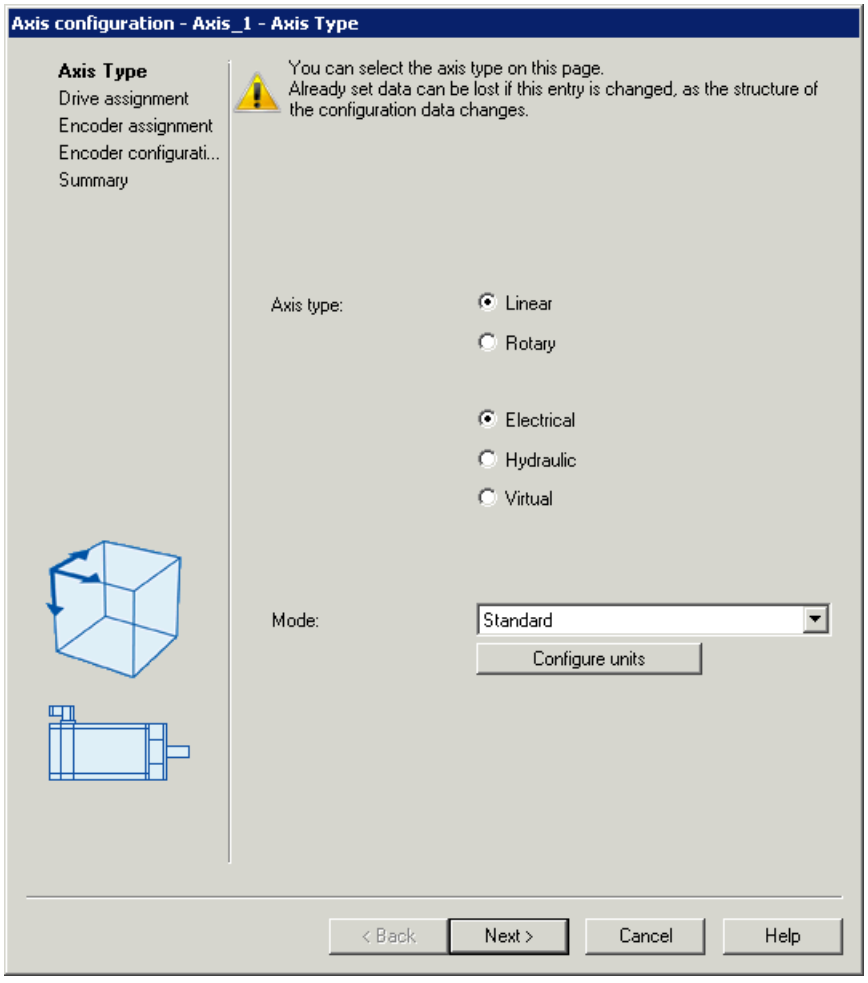


Figure 3-452 Axis configuration - defining the axis type

5. Select the axis type, and specify it. Select the mode for electric axes or hydraulic axes and confirm with "Next>".
   The "Axis Configuration - Drive Assignment" dialog opens.

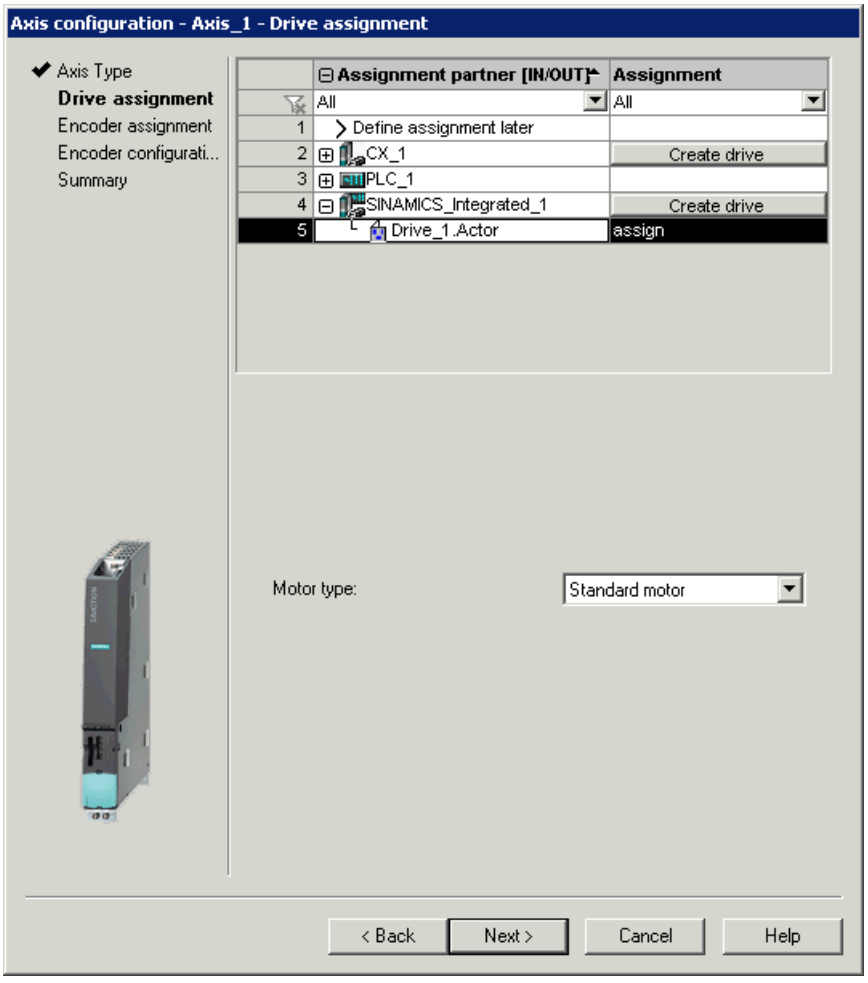


Figure 3-453 Axis configuration - Drive assignment

6. Depending on the selected axis type, with an electric axis you assign the drive which you have configured previously in the "Drive assignment" step.
   Several drives are configured in the example: "Drive_unit_1" (CU310-2), "CX_1" (CX32-2) and "SINAMICS_Integrated_1" (SIMOTION D SINAMICS Integrated).

   Note

   As an alternative to assigning the axis to an already configured drive, the axis wizard offers two further selection options:

   - Define the axis/drive assignment later:
     The axis is to be created and not assigned to a drive until later. Programming and simulation of the axis are also possible here.
   - Create drive:
     The drive wizard can be called up from the axis wizard (offline configuring). The axis can thus be created in one step along with the drive, and assigned to the drive.

   The alternative approaches are not being considered at this point.

   With an hydraulic axis, you configure the output for a Q-valve in the step "Configuration of Q-output"; the Q-valve represents the actuator for a volume flow.
   Confirm with "Next>".
7. Specify the encoder to be used by the axis or the active axis data set in the "Encoder assignment" step.
   Confirm with "Next>".
8. Specify the parameters for the encoder used in the "Encoder configuration". Only the encoder type has to be specified when using the symbolic assignment.
9. Once you have worked through all the steps of the configuration of a TO axis, the configured configuration is displayed once again for you to confirm.
   Close the dialog with "Finish".

Result

The axis is displayed in the "AXES" folder of the project navigator.

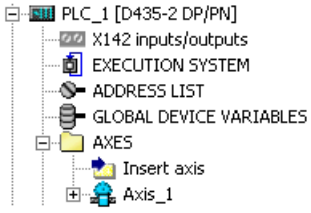


Figure 3-454 Axis created

Automatic settings of the engineering system

The engineering system automatically defines the PROFIdrive axis telegrams required for communication with the drive, as well as the addresses used.

In the same way, telegrams are extended and interconnections automatically created in the drive, depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

Drive and encoder data, as well as reference variables, maximum variables, torque limits, and granularity in torque reduction of the SINAMICS S120 are accepted automatically for the configuration of the SIMOTION technology objects "TO axis" and "TO externalEncoder". This data no longer has to be entered in SIMOTION.

Loading the axis configuration to the target system

Save and compile the changes and download the axis configuration to the target system in order to be able to test the function of the axis in the next configuration step.

Additional references

For detailed information, refer to the section "Configuring the axis and external encoder" in the SIMOTION SCOUT TIA Online Help.

Testing the axis with the axis control panel

Using the axis control panel you can control and monitor individual axes.

You can perform the following tasks with the axis control panel:

- Test axes before you traverse them via a program.
- Testing as to whether you can move the axis using the axis control panel if a fault is detected.
- Set/remove axis enable.

Requirements

- You have created, configured and connected an axis to the drive.
- SIMOTION SCOUT TIA is in online mode.
- You have loaded the project to the target system.

Opening the axis control panel

To open the axis control panel, proceed as follows:

1. Open the "AXES" folder in the project navigator.
2. Double-click the "Control panel" entry below the axis to be tested.

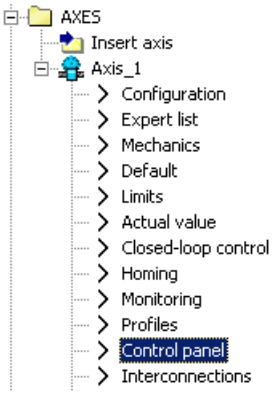


Figure 3-455 Select axis control panel

The axis control panel opens in the detail view.

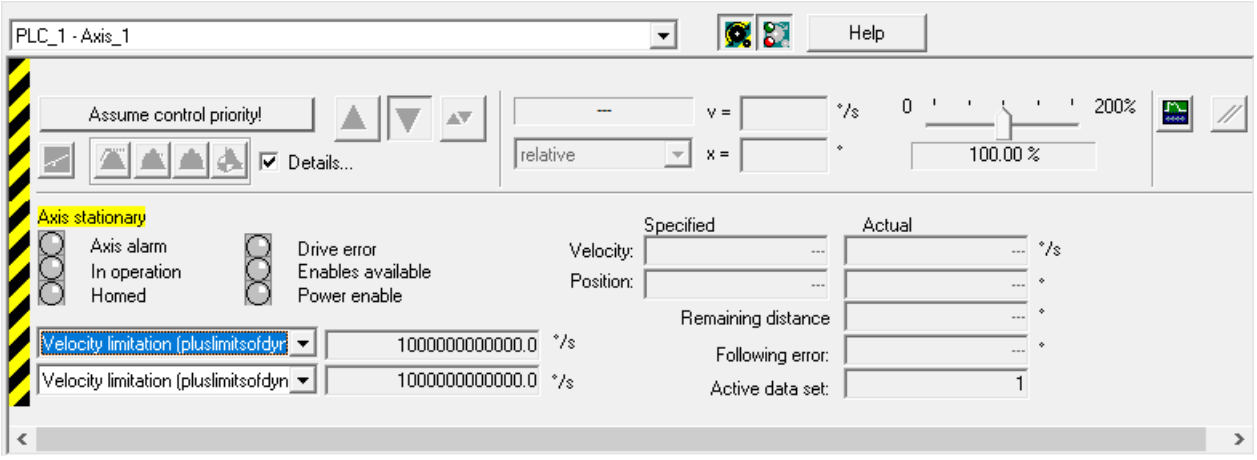


Figure 3-456 Axis control panel

Using the axis control panel

⚠ WARNING

Activate sign-of-life monitoring

Use the axis control panel in control mode only with activated sign-of-life monitoring and a suitably short monitoring time! Otherwise, if problems occur in the communication link between the control PC and the device, the axis may start moving in an uncontrollable manner.

Assume control priority

To assume control priority, proceed as follows:

1. Show the control area by clicking the button in order to test, for example, programmed traversing motions in this mode.
2. Show the diagnostic area by clicking the button in order to test traversing motions that are caused by the motion commands sent to the axis in this mode.
3. Select the axis that you want to traverse and monitor from the selection list, PLC_1 - Axis_1 in the example.
4. Specify the parameters you want to display in the lower left side of the Diagnostics area using the selection lists. The corresponding values are shown next to them.
5. Select "Assume control priority" to establish the connection to the PG/PC.
   The "Assume Control Priority" dialog opens.

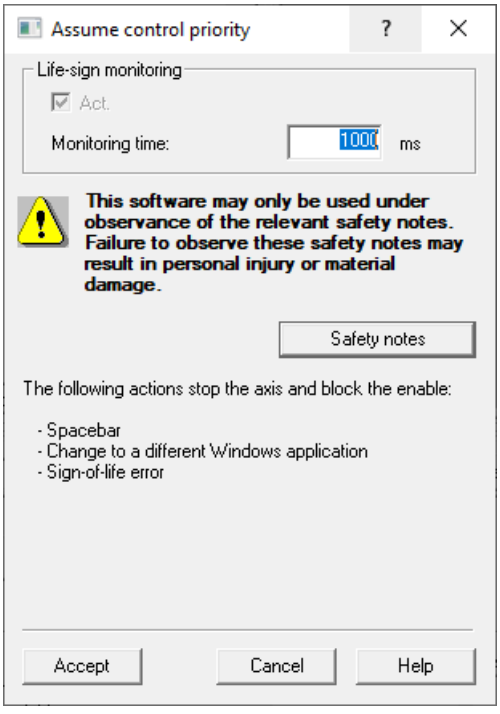


Figure 3-457 Assume control priority

6. Read the safety instructions and confirm these with "Accept".

The PG/PC has the control priority; the label of the button has changed to "Return control priority".

Note

Stopping the axis

By pressing the SPACEBAR, you can stop the axis at any time.

Activating enables

To activate the enables, proceed as follows:

1. Click the "Set/remove enables" button to enable the axis.
2. Confirm the dialog "Switch axis enable" with "OK".
   The LEDs indicate the status of the enables.

Specifying motion commands for axis

1. Specify motion commands for the axis.
   The following functions are available:

Button	Meaning/function
	Speed default This command specifies that the axis is started speed-controlled. Specify a speed setpoint to which the axis can be ramped up via a velocity profile. The command is possible for all axis types; position and following axes are operated speed-controlled.
	Start axis position-controlled This command specifies that the axis is started position-controlled. Specify a speed setpoint to which the axis can be ramped up via a velocity profile. The command is possible for position and following axes.
	Position axis With this command, you specify positioning of the axis (positioning or synchronous axis) at a specific position. The position value entered can be absolute or relative. Modulo axes can also be positioned via the shortest path. The programmed position must be within the software limit switches.
	Home axis With an absolute measuring system, homing is only required once during commissioning. When this is completed, the position value will be known when the machine is switched on. In the case of an incremental measuring system, the machine must be homed every time it is switched on.

Start axis motion

To start the axis motion, proceed as follows:

1. Click the switch to start the motion command parameterized last.
   Or click the switch to start the "Speed specification" or "Start axis position-controlled" motion commands in jog mode.
   The motion continues as long as the right-hand mouse button is pressed.

Exit axis control panel

To exit the axis control panel, proceed as follows:

1. To stop the system, click the switch.
2. Deactivate the "Set/remove releases" button.

3. Confirm the "Remove Axis Enable" dialog with "OK".

Note
Acknowledging alarms

If you want to traverse the axis again, acknowledge all alarms in the "Alarms" window.

4. Click "Give up control priority" to return the control priority to the control panel.
5. On the menu bar, select "Project > Disconnect from target system" to switch to offline mode.

Result

The axis can be traversed. Axis configuration is complete.

Potential problems

The axis cannot be traversed.

If the axis cannot be traversed using the axis control panel, test to see if you can traverse the drive using the drive control panel.

- If you cannot traverse the drive using the drive control panel, there is a problem with the drive.
- If you can traverse the drive with the drive control panel, there is a communication problem. Check the drive.

See also

Testing the drive with the drive control panel (Page 833)

3.5.8.6 Program SIMOTION application

Using tags

Variable types

Variables are used to structure programs. They are wild cards in a program and can assume values.

In SIMOTION, different types of variables are distinguished:

- **System variables**
  Each SIMOTION device and technology object has specific system variables. You can access system variables within the SIMOTION device from all programs.
- **I/O variables**
  An I/O variable is a symbolic variable name that is assigned to an I/O address of the SIMOTION device or to the I/O. Thus, direct access to the peripherals is possible.
  I/O variables are valid for all devices. All programs of the SIMOTION device have access to them.
- **Global device variables, unit variables and local variables** are user-defined variables with limited scope:
  These global device variables can be accessed from all parts of the user program. They can also be accessed from HMI devices.
  Unit variables can be accessed by all programs, function blocks and functions defined within the same source, e.g. ST source, MCC source, LAD/FBD source.
  A source is a logic unit that you can create in your project and that can contain programs, functions and function blocks.
  Local variables can only be accessed within the program, the function or function block in which they are defined.
  If variables have been defined in the interface section of a source, they can be accessed by other sources and external components, e.g. HMI systems.

Creating global device variables

Note

Global device variables can only be created in offline mode.

To create global device variables, proceed as follows:

1. Click the "GLOBAL DEVICE VARIABLES" element in the project navigator below the SIMOTION device.
   The "Global device variables" table is displayed in the symbol browser.
2. In the "Name" column, click the first empty cell and enter the variable name, "g_bo_start" in the example.
   Press the RETURN or TAB key. The input focus jumps to the "Data type" field. Alternatively, you can click on the field to move the input focus to it.
3. Enter the data type in the "Data type" field, "BOOL" in the example.
4. Press <RETURN> to confirm.
   The variable is created and available in the project. In the symbol browser, a new line is opened for input.

5. Create other global device variables in the same way, "g_bo_ready" in the example.

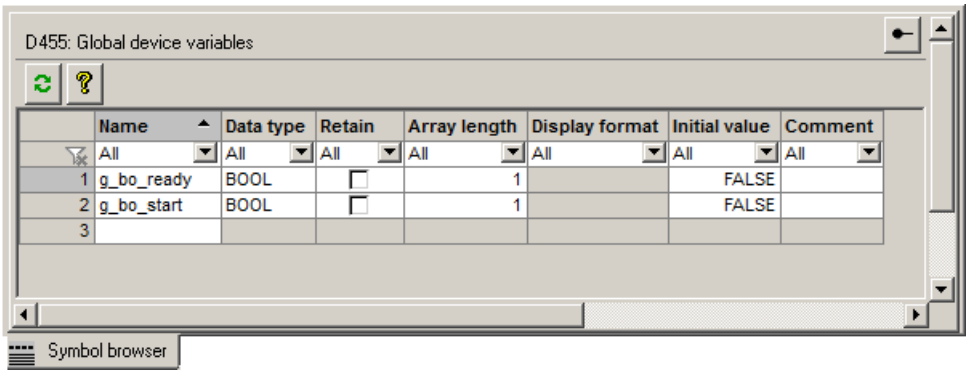


Figure 3-458 Creating global device variables

6. Select "Project > Save and compile changes" in the menu to accept the expansions in the project.

Creating I/O variables

Note

I/O variables can only be created in offline mode.

To create I/O variables, proceed as follows:

1. Double-click the "ADDRESS LIST" element below the SIMOTION device in the project navigator.
   In the detail view, the "Address list" tab opens.
2. Click on the first free cell in the "Name" column. Enter a name for the variable.
3. Press the <RETURN> or <TAB> key. The cursor moves to the "I/O address" field.
4. Select the "IN" entry for input variables or "OUT" for output variables in the I/O address column.
5. Click the three dots in the "Assign" column to open the "Assignment" dialog.
6. In the "Assignment" dialog, select the required variable and select the "Assign" entry in the "Assignment" column.
7. Confirm the dialog with "OK".
8. Select "Project > Save and compile changes" in the menu to accept the expansions in the project.

Note

Steps 4, 5, 6 and 7 only apply for devices that support symbolic assignment (e.g. CU320-2).

Additional references

You will find detailed information on the creation of variables in the SIMOTION SCOUT TIA Online Help and the relevant Programming and Operating Manuals.

Use MCC

Overview

Program creation steps

Creation of an MCC program encompasses the following steps:

1. Creating the MCC unit.
2. Creating the MCC chart in the MCC unit.
3. Inserting MCC commands in the MCC chart and parameterizing the commands.

Creating the MCC unit

Procedure

To insert an MCC unit, proceed as follows:

1. In the project navigator, below the SIMOTION device, open the "PROGRAMS" folder.
2. Double-click "Insert MCC unit".
   The "Insert MCC Unit" dialog opens.

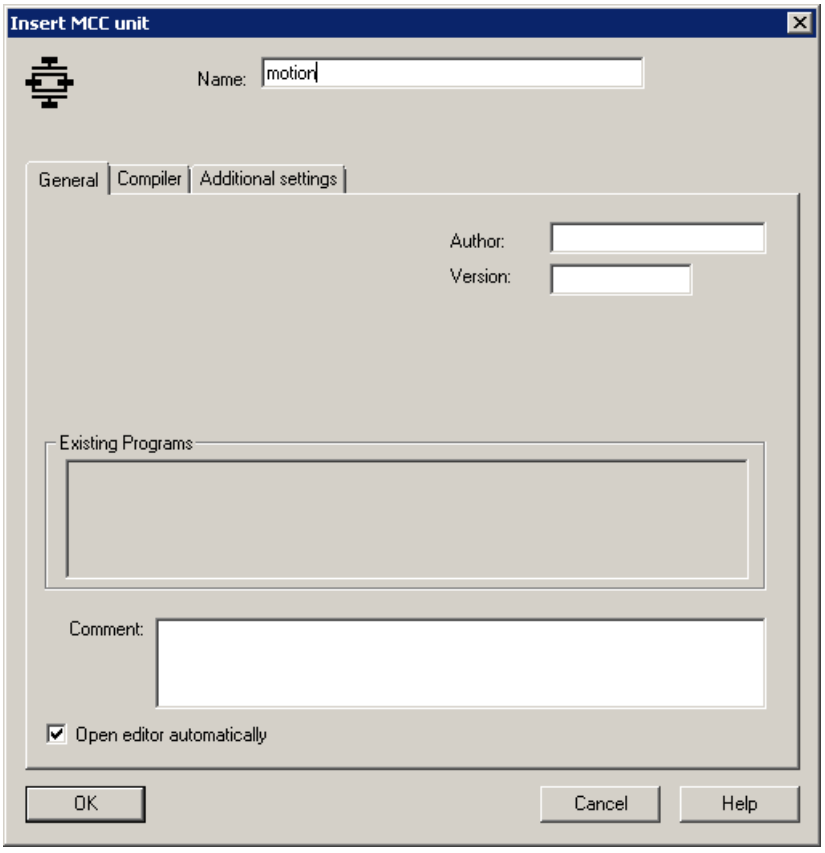


Figure 3-459 Inserting an MCC unit

3. Assign a name, "motion" in the example.
4. Go to the "Compiler" tab. For diagnostics purposes, activate the "Permit program status" and "Permit single step" options. In this way, you can monitor program execution later in online mode.
5. Confirm with "OK".

Result

The MCC unit is created.

- The unit appears in the project navigator below the "PROGRAMS" branch.
- The declaration table of the source file opens in the working area of the workbench. The variables declared there apply within the MCC unit and can be linked in other units.

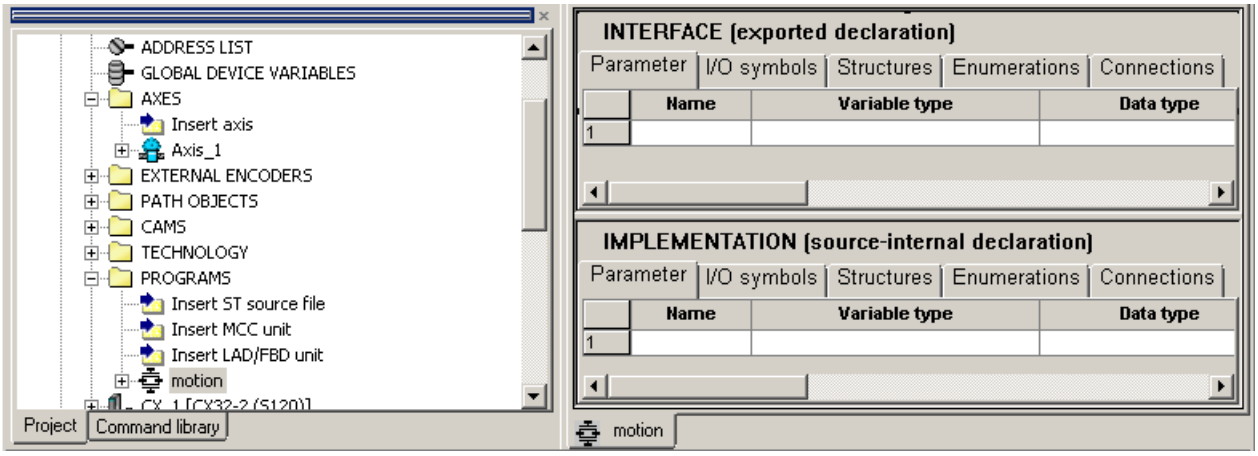


Figure 3-460 MCC unit "motion" inserted

Creating an MCC chart

Procedure

To insert an MCC chart, proceed as follows:

1. In the project navigator, below the SIMOTION device, open the "PROGRAMS" folder.
2. In the "PROGRAMS" folder, open the MCC unit, "motion" in the example.

3. Double-click "Insert MCC chart".
   The "Insert MCC Chart" dialog opens.

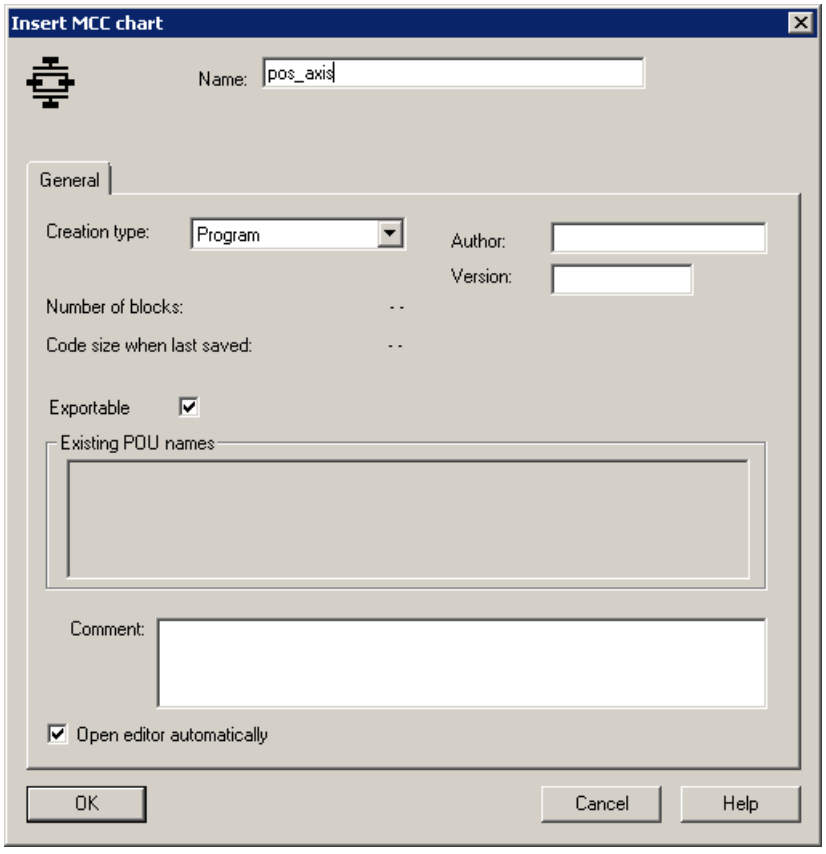


Figure 3-461 Inserting an MCC chart

4. Assign a name, "pos_axis" in the example.
5. For the creation type, select "Program".
6. Confirm with "OK".

Result

The MCC chart is created in the project.

- The created MCC chart "pos_axis" appears in the PROGRAMS folder below the "motion" unit.
- The MCC editor is opened in the working area of the workbench. The start and end nodes are already pre-defined. You can start MCC programming.

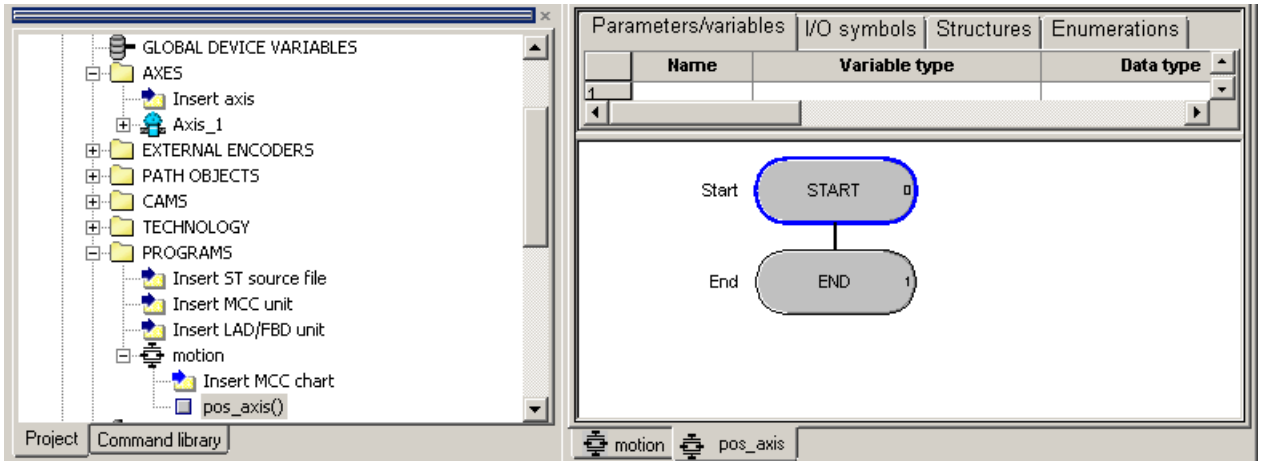


Figure 3-462 MCC chart "pos_axis" inserted

Using MCC command blocks

Every newly created MCC chart already contains a start and an end node.

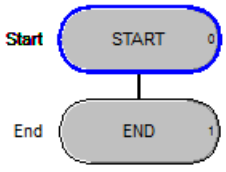


Figure 3-463 MCC chart, start/end nodes

You insert the MCC command blocks between these. The commands are processed in the direction from the start to the end node.

The MCC commands are available to you via:

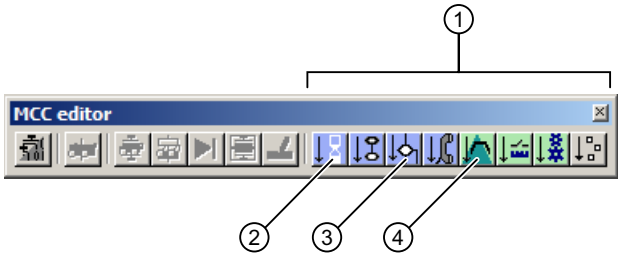
- "MCC editor" toolbar
- "MCC chart > Paste" menu command
- Context menu of the command block

Using the MCC editor toolbar

Open the toolbar

The "MCC editor" toolbar is displayed in the workbench as soon as you open an MCC chart.

The commands are arranged into command groups.



① Command groups
② Basic commands
③ Program structures
④ Single axis commands

Note

Displaying the MCC toolbar

If you do not see the toolbar, check that the display is switched on: Open the "View > Toolbars" menu. Select the checkbox for "MCC editor" in the "Toolbars" window.

Opening the command group

Move the cursor across the colored buttons of the toolbar to display the command groups.

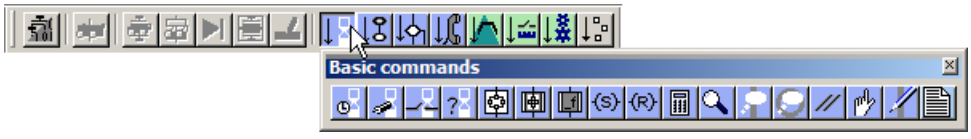


Figure 3-464 Opening the command group

Keeping command groups open or closed continuously

Click the window title of a command group to keep the command group open continuously.

Select "Hide" in the context menu of a command group to close the command group.

Placing the command group as required

Drag the toolbar or the command groups of the toolbar with the mouse to any location on the workbench.

Docking the toolbar

Drag the toolbar or the command groups of the toolbar with the mouse to the edge areas of the workbench to dock them there.

Showing a tooltip for the command

Hold the mouse pointer briefly over a command button. The designation of the command is shown.

Inserting MCC editor commands in the MCC chart

In order to insert an MCC editor command in the created MCC chart, proceed as follows:

1. In the active MCC chart, click the connecting line between two commands, or click the command after which the new command is to be inserted.
   The connecting line or the border line of the command button is marked blue. The marking flashes.
2. On the "MCC editor" toolbar, select the command group.
3. Click the desired command in the command group.

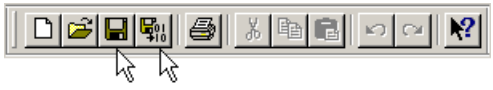
Result

The command was inserted into the chart and can now be parameterized.

Backing up the MCC program

To back up the MCC programs, proceed as follows:

1. On the toolbar, click the "Save project" or "Save project and compile changes" button.



Alternatively you can click the "Accept and compile" command in the MCC editor toolbar.

This command compiles the currently selected program as well as all other programs of the same unit.
However, the command does not save the changes.
You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

Additional references

For further information, refer to the SIMOTION MCC Motion Control Chart Programming and Operating Manual.

The "Getting Started section of the SIMOTION SCOUT TIA" Online Help contains a detailed description of a sample configuration.

Use LAD/FBD

Overview

Program creation steps

Creation of a LAD/FBD program encompasses the following steps:

1. Creating an LAD/FBD unit.
2. Creating an LAD/FBD program in the LAD/FBD unit.
3. Inserting LAD/FBD commands in the LAD/FBD program and parameterizing the commands.
4. Saving and compiling the LAD/FBD program.

Create LAD/FBD unit

Procedure

In order to set up an LAD/FBD unit, proceed as follows:

1. In the project navigator, below the SIMOTION device, open the "PROGRAMS" folder.
2. Double-click "Insert LAD/FBD unit".
   The "Insert LAD/FBD Unit" dialog opens.

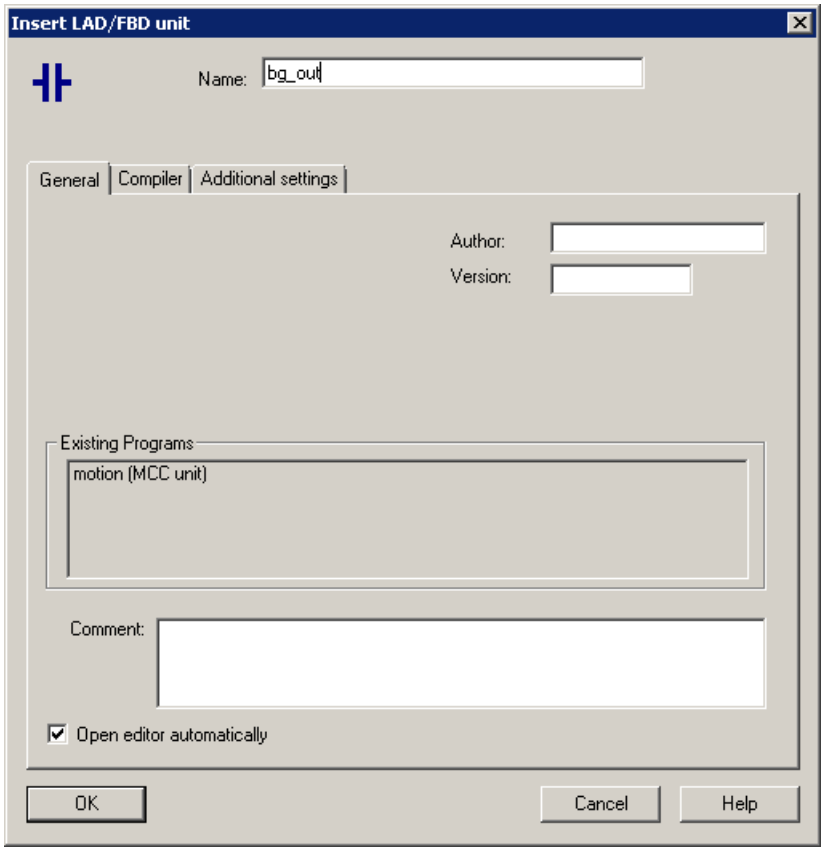


Figure 3-465 Inserting the LAD/FBD unit

3. Assign a name, "pos_axis" in the example.
4. Switch to the Compiler tab.
5. Activate the "Permit program status" option for diagnostics purposes. In this way, you can monitor program execution later in online mode.
6. Confirm with "OK".

Result

The LAD/FBD unit is created.

- The LAD/FBD unit "bg_out" appears in the "PROGRAMS" folder.
- The declaration table of the source file opens in the working area of the workbench. The variables declared there apply within the LAD/FBD unit and can be linked in other units.

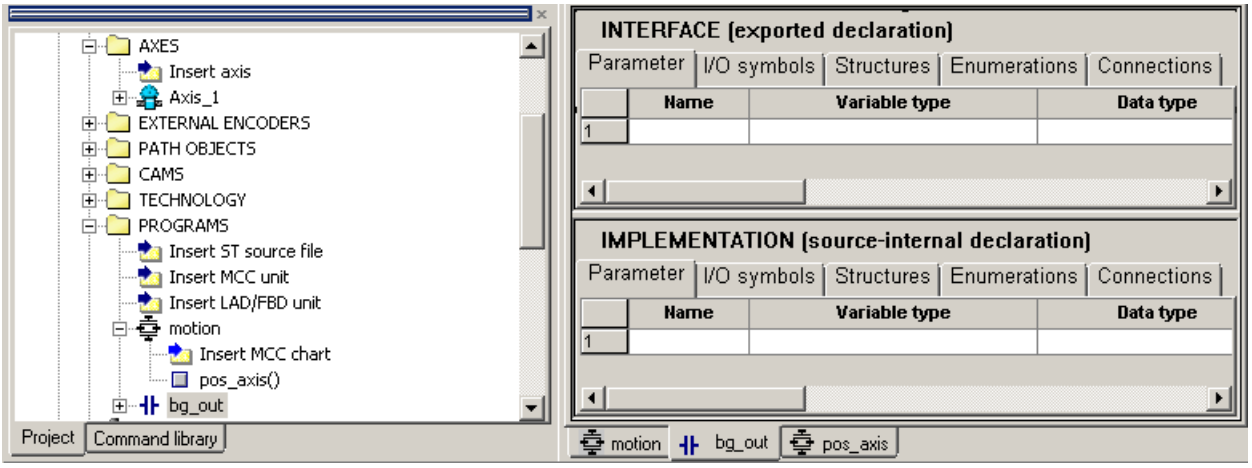


Figure 3-466 LAD/FBD unit "bg_out" inserted

Create LAD/FBD program

Procedure

In order to set up an LAD/FBD program, proceed as follows:

1. In the project navigator, below the SIMOTION device, open the "PROGRAMS" folder.
2. Open the LAD/FBD unit "bg_out" in the "PROGRAMS" folder.

3. Double-click "Insert LAD/FBD program".
   The "Insert LAD/FBD Program" dialog opens.

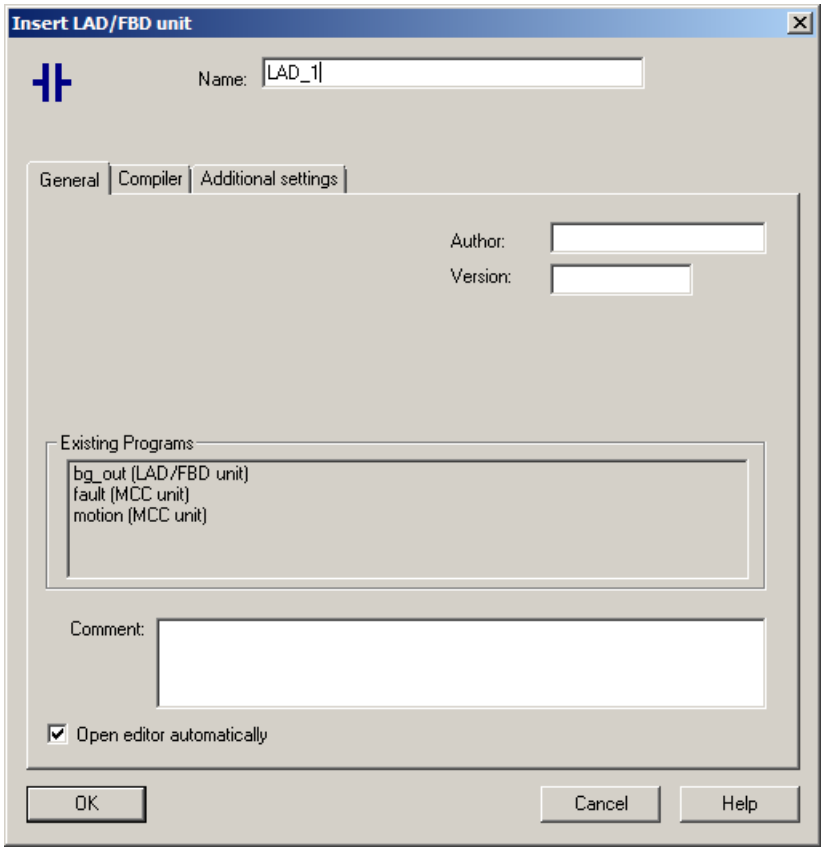


Figure 3-467 Inserting the LAD/FBD program

4. Assign a name, "LAD_1" in the example. The name must be unique throughout the project.
5. For the creation type, select "Program".
6. Confirm with "OK".

Result

The LAD/FBD program "LAD_1" is created in the project.

- The LAD/FBD program appears in the "PROGRAMS" folder.
- The LAD/FBD editor is opened in the working area of the workbench. You can start programming.

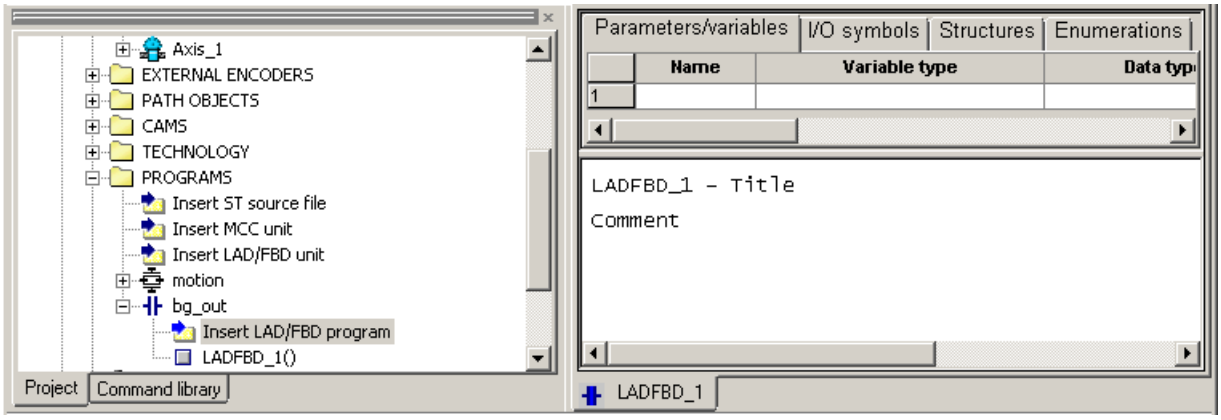


Figure 3-468 LAD/FBD program "LAD_1" inserted

Using the LAD/FBD toolbar

Opening the LAD/FBD toolbar

The LAD/FBD toolbar becomes visible in the workbench as soon as you open an LAD/FBD program.

Move the cursor across the active buttons of the toolbar to display the functions.

Note

Displaying the LAD/FBD toolbar

If you do not see the toolbar, check if the display is switched on: Open the "View > Toolbars" menu. In the "Toolbars" window, activate the "LAD/FBD toolbar" checkbox.

Switching the programming language

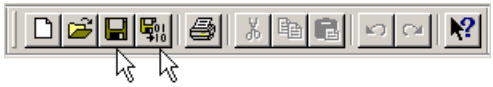
SIMOTION SCOUT TIA allows simple switching between ladder logic and function block diagram. The LAD/FBD editor contains the command "LAD/FBD program > Switch to FBD" or "Switch to LAD".

Backing up the LAD/FBD program

Procedure

In order to back up an LAD/FBD program, proceed as follows:

1. On the toolbar, click the "Save project" or "Save project and compile changes" button.



Alternatively you can click the "Accept and compile" command in the LAD/FBD toolbar.

This command compiles the currently selected program as well as all other programs of the same unit.
However, the command does not save the changes.
You thus have the option of accepting changes to a program into the project without having to save or compile the entire project again.

Additional references

For further information, refer to the Online Help and SIMOTION LAD/FBD Programming and Operating Manual.

The "Getting Started section of the SIMOTION SCOUT TIA" Online Help contains a detailed description of a sample configuration.

Using ST

Overview

Program creation steps

The creation of an ST program encompasses the following steps:

1. Creating an ST source file.
2. Saving and compiling the ST program.
3. Executing the ST program.

Note

Template for example unit

An extensively annotated *template for example unit* is also available in the online help.

Creating an ST source file

Procedure

To create an ST source file, proceed as follows:

1. In the project navigator, below the SIMOTION device, open the "PROGRAMS" folder.
2. Double-click Insert ST source file.
   The "Insert ST Source File" dialog opens.

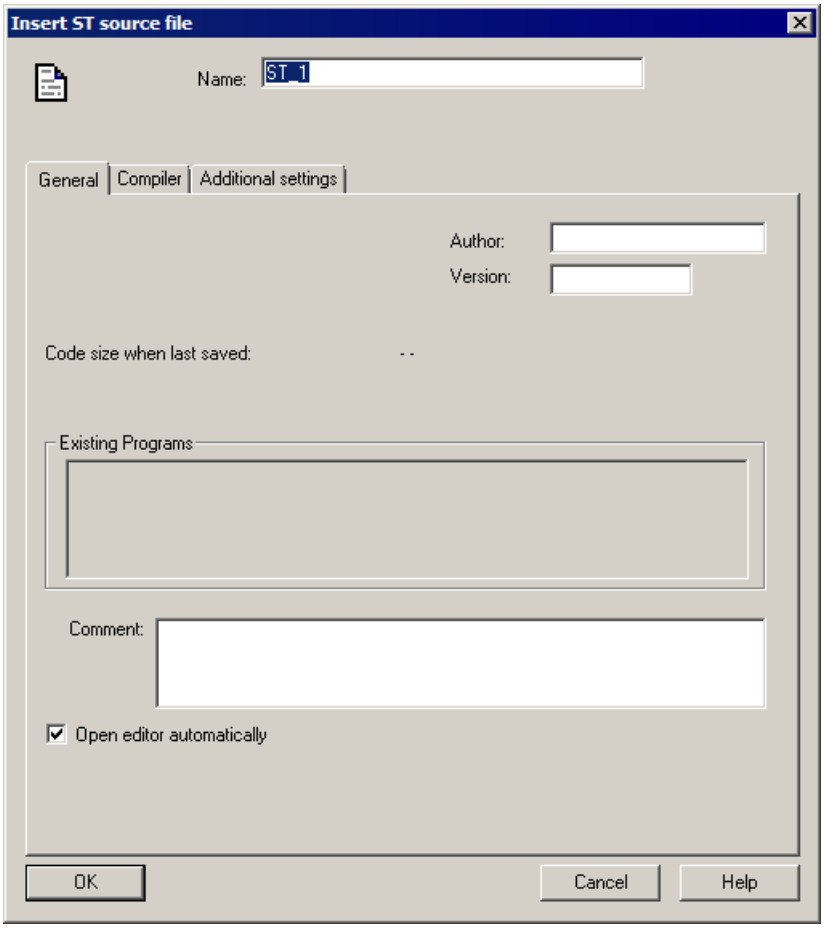


Figure 3-469 Inserting an ST source file

3. Assign a name, "ST_1" in the example.
4. Confirm with "OK".

Result

The ST source file has been created.

- The unit appears in the project navigator below the "PROGRAMS" branch.
- The declaration table of the source file opens in the working area of the workbench.

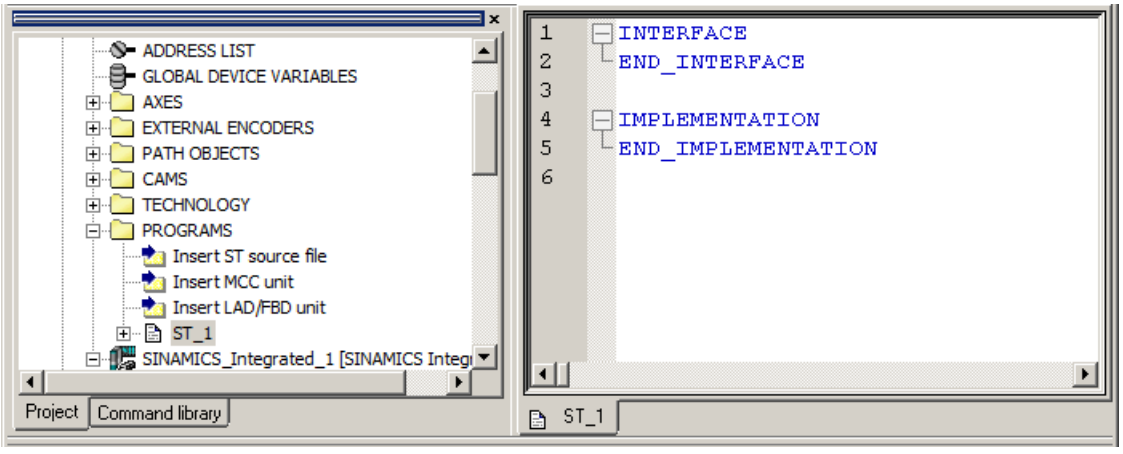


Figure 3-470 ST source file "ST_1" inserted

Using the Snippet Editor

With the Snippet Editor, you can create your own snippets for frequently used code blocks. You assign a keyword to the code block and insert the code block into your ST program by calling the keyword.

To call the Snippet Editor, click in the active work area of your ST source and select "View > Snippet Editor" in the context menu.

Additional references

Detailed information on the Snippet Editor can be found in the SIMOTION SCOUT TIA online help, section "Code Snippets".

Using the ST toolbar

Using the ST toolbar

The ST toolbar is displayed in the workbench as soon as you have created an ST source file.

Move the cursor across the buttons of the toolbar to display the functions.

Backing up the ST program

Procedure

To back up an ST program, proceed as follows:

1. Save the entire project.
   Select the "Save" command in the "Project" menu.
2. Click the "Accept and compile" button in the toolbar to run and test the program.

Executing the ST program.

Before you can execute the sample program, you must assign it to an execution level or a task. When you have done this, you can establish the connection to the target system, download the program to the target system, and then start it.

Additional references

Detailed information can be found in the SIMOTION SCOUT TIA Online Help and in the SIMOTION ST Structured Text Programming and Operating Manual.

The "Getting Started section of the SIMOTION SCOUT TIA" Online Help contains a detailed description of a sample configuration.

Using libraries

In SIMOTION SCOUT TIA, you can create general functions or standard functions as a library and use them in ST source files, LAD/FBD programs or MCC charts for programming.

You can create user and standard libraries for user-defined data types, unit variables, functions (FC) and function blocks (FB).

You create libraries in the project navigator in the "LIBRARIES" folder. You store the ST source files/ LAD/FBD programs/MCC charts for the library below the library. A library's individual ST source files, LAD/FBD programs or MCC charts can be protected against unauthorized opening, reading and modification via the know-how protection for programs.

Inserting libraries

To insert a library, proceed as follows:

1. Open the "LIBRARIES" folder in the project navigator.
2. Double-click "Insert library".
   The "Insert library" dialog opens.

3. Enter the name of the library.

   Note

   Long library names

   If the library name is longer than 9 characters, the name may be shortened depending on the set column width. A tooltip shows the full library name.

   For example, a user has assigned the name "myVeryVeryLongLibName_1". "myV ... e_1" is shown.

   If there is only a small number of libraries, the displayed name can be longer, depending on the window size.

   CPU names can also be long, so their display mode is limited in the same way as the library names, except that the displayed length is 15 characters. The full name is displayed via a tooltip here also.

4. Go to the "Technology packages" tab and select the device and the technology package. You can select specific technology packages for each device.
5. Switch to the "Compiler" tab and enable the current settings for the library. This setting applies to all ST source files of this library. The current settings overwrite the global settings.
6. Confirm with "OK". The library is created.
7. To create the ST source files/LAD programs/MCC charts of the library, double click on "Insert ST source file"/"Insert MCC unit"/"Insert LAD/FBD unit" under the folder of the created library You can also insert ST, MCC or LAD/FBD source files from the "PROGRAMS" folder into the library by copying or drag-and-drop, or import them via a selective export from another project.

Using libraries

To use libraries, follow these steps:

1. For MCC charts and LAD/FBD programs, select the connection type "Library" in the "Connections" tab in the active work area of the source. Select the name of the library. For ST source files, integrate the libraries via the USELIB command.

A library is always specifically created for one SIMOTION device and one technology package. These settings are taken into consideration during compilation and the library can only be run on this set device with the selected technology package. If the library is also to be capable of being run with other devices or technology packages, you have to change the library's properties accordingly and perform the compilation again. When changing the technology package, you must make sure that the system functions used in the library are also to be found in the technology package. If there are any errors, they will be displayed during compilation.

Debugging libraries

Debugging a library is only possible if it is connected to a device. You can link libraries to CPUs to define the context in the editors so that the files are available from the corresponding CPU via "Diagnostics".

Proceed as follows to connect the library with a CPU and select it:

1. Select the "LIBRARIES" folder in the project navigator.
2. Right-click and select "Debugging by target device..." in the context menu.
   Only the CPUs available online are displayed.
   CPUs that are already connected are marked with a checkmark.

   Note

   A library can only be connected with one CPU at any time

   Please note that a library can only be connected with one CPU at any time.

   To disconnect a device, click on the CPU again.

3. Set the connection.
   The libraries are connected.

   Note

   The same CPU can be connected with multiple libraries

   You can connect multiple libraries to the same CPU.

Additional references

Detailed information on the topic of libraries can be found in the SIMOTION SCOUT TIA online help, section "Libraries".

Searching in variables and programs

In the open project, you can search all the project data for variables or any text.

If you select a variable search, you can also search within the sources. All global and local variables at the declaration and use points are recognized.

Global search

To start the search, proceed as follows:

1. In the "Edit" menu, select the "Search in the project..." command.
   The "Search in the Project" dialog opens.

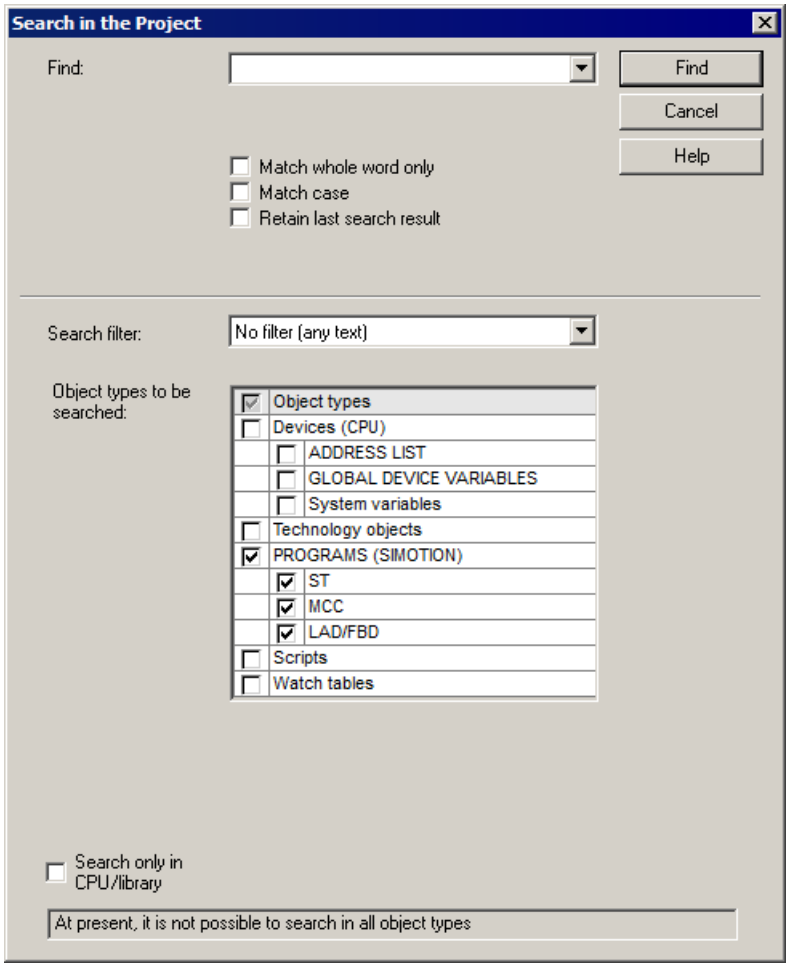


Figure 3-471 Search in the project

2. Enter the search text and specify additional criteria for the search.
3. Start the search by clicking "Find".

Result

The search results are displayed in the detail view in the "Search result" tab.

Local search

You can search locally for a specific text in the various editors and the declaration tables.

To start the local search, proceed as follows:

1. Open the appropriate editor, e.g. for a program or created global device variable.
   The focus in on the editor window.
2. In the "Edit" menu, select the "Search..." command.
   The "Search" dialog opens.
3. Enter the search text.
4. To start the search, click "Find next".

Result

The result of the search is highlighted in the editor window.

3.5.8.7 Configure execution system

Execution levels define the chronological sequence of tasks in the execution system. A level can contain several tasks. The tasks provide the framework for program execution. A task may comprise several programs. By assigning the created programs to the tasks, you can, for example, define the priority, the time frame or order in which the programs are to be executed.

Assigning programs to tasks

To assign programs to tasks in the execution system, proceed as follows:

1. In the project tree, under the SIMOTION device, double-click "EXECUTION SYSTEM". The "EXECUTION SYSTEM" window opens in the working area.
2. Assign the required program to the preferred task. In the sample MCC program, "motion.pos_axis" and "MotionTask_1" task.
   – Select the "ExecutionLevels > OperationLevels > MotionTasks > MotionTask_1" branch in the tree of the execution system.

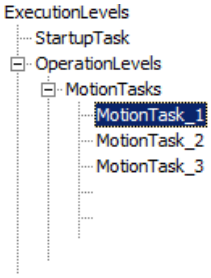


Figure 3-472 Execution system, tree view of the execution levels and tasks

The "MotionTasks" window opens on the right of the working area. On the "Program assignment" tab, under "Programs," the programs "pos_axis" and "KOP_1" and the programs of the source "fault" are visible.

– Select the MCC program "motion.pos_axis" and click the ">>" button. The program is displayed under "Programs used". Thus, the "MotionTask_1" task is assigned.

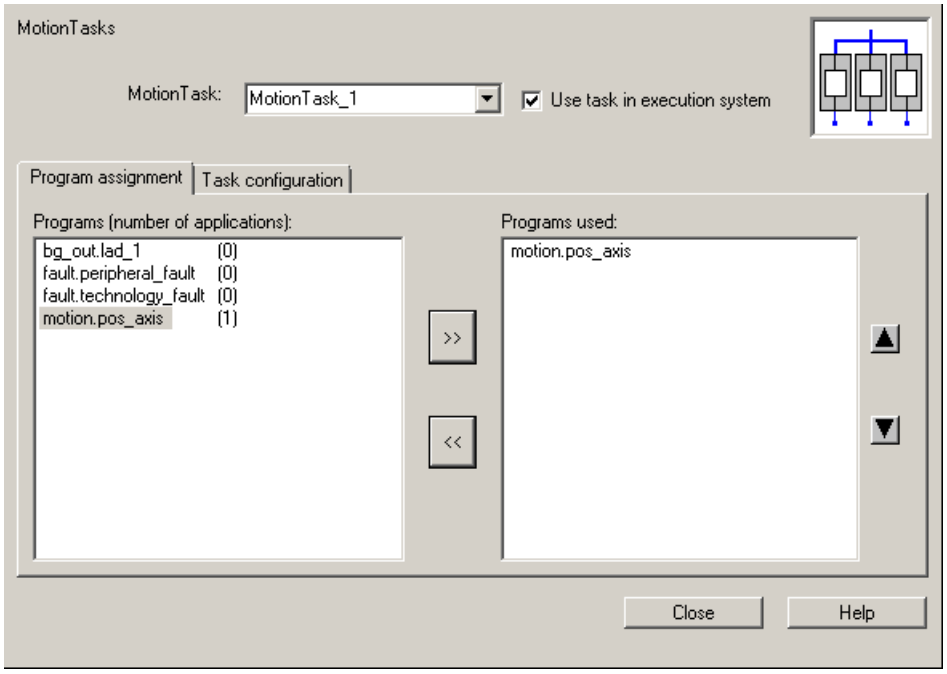


Figure 3-473 Execution system, MotionTasks window

The assignment is visible in the tree of the execution system. The "motion.pos_axis" program appears below the "MotionTask_1" branch.

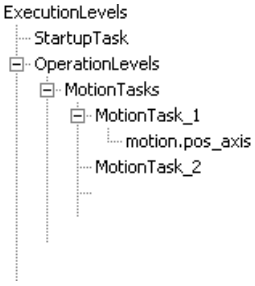


Figure 3-474 Execution system, MotionTask_1 with assigned pos_axis program

– In the "Task configuration" tab, activate the "Activation after StartupTask" option. This executes the MCC program immediately after the SIMOTION device is started. If this option is not activated, the program must be started explicitly by the call from another program that is assigned to the StartupTask or another active task.

3. Assign the LAD program "bg_out.kop_1" to the task "BackgroundTask":

   – In the tree of the execution system, select the branch "ExecutionLevels > OperationLevels > BackgroundTask". Assign the LAD program "bg_out.kop_1" to this task.

4. Assign the fault handling routines:
   - In the tree of the execution system, select the branch "ExecutionLevels > OperationLevels > SystemInterruptTasks > TechnologicalFaultTask". Assign the MCC program "fault.technology_fault" to this task.
   - In the tree of the execution system, select the "ExecutionLevels > OperationLevels > SystemInterruptTasks > PeripheralFaultTask" branch. Assign the MCC program "fault.peripheral_fault" to this task.
5. Click the "Close" button. Confirm with "Yes" if you are prompted to save.

Result

The execution system is configured.

Loading the configured execution system to the target system

Save and compile the changes, go online and load the configured execution system into the target system.

3.5.8.8 Controlling the target system

RUN and STOP operating states

Operating states

In the SIMOTION SCOUT TIA dialog "Control Operating State", you can switch a SIMOTION CPU to the RUN or STOP state.

RUN operating state

SIMOTION executes the user program and the associated system services:

- Reading process image of inputs
- Execution of the user programs assigned to the execution system
- Writing process image of outputs

The technology packages are active in this state. They can execute commands from the user program.

STOP operating state

SIMOTION does not process a user program.

- It is possible to load a complete user program.
- All system services (communication, etc.) are active.

- The I/O modules are in the safe state (this means, for example, digital outputs are "LOW" and analog outputs are de-energized or at zero current).
- The technology packages are inactive, i.e. all releases are deleted. No axis motions can be executed.

Additional references

Detailed information about operating states can be found in the section "SIMOTION device: Operating state**"** in the SIMOTION SCOUT TIA Online Help.

Control operating mode

⚠ WARNING

Danger to life through unexpected machine movement

If the operating state is not switched under controlled conditions, this may endanger the safety of personnel and the machine.

- Observe the safety regulations before you control a SIMOTION device via the mode selector switch in SIMOTION SCOUT TIA.

Switching SIMOTION devices to the RUN or STOP operating state

To control the operating state, proceed as follows:

1. Select "Target system -> Control operating state..." in the menu to open the "Control Operating State" dialog.
   Or click on the "Control operating state" icon in the toolbar.

   The call is possible if at least one CPU of the project is in online mode.
2. Click the assigned button to select the desired "RUN" or "STOP" operating state for the displayed devices in the "Control Operating State" dialog.
   The "State" field indicates whether the operating state change was performed.

Control Operating State dialog

The dialog shows the operating state of all configured CPUs. The display can be filtered. The change of operating state is possible per device, or for several devices simultaneously.

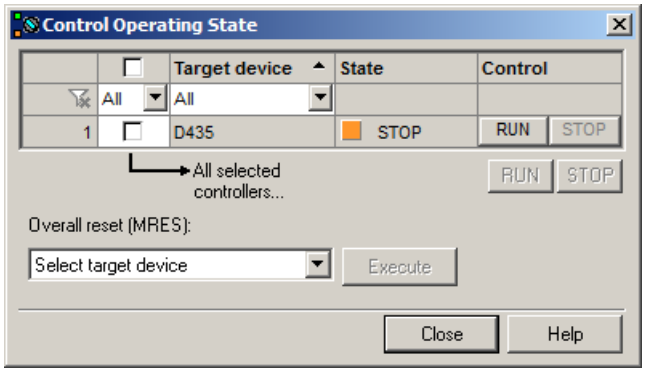


Figure 3-475 STOP operating state

Note

"Control Operating State" dialog with all configured devices

As of Version 4.4, the "Control Operating State" dialog represents all configured SIMOTION devices. The previous dialog had to be opened individually for each CPU.

Overall reset is possible in the dialog as before. You will find additional information about overall reset in the section Overall reset (Page 880).

The SIMOTION P State tool is retained unchanged.

Listed devices

The "Target device" column shows the names of the CPUs available in the project. The order corresponds to the order of the CPUs in the project navigator.

Offline/online distinction

If a CPU is in offline mode, the name of the CPU and all other assigned elements are grayed out in the table line.

The checkbox is always open for input.

Filter

The list of the CPUs can be filtered according to the following criteria:

- Selection by checkbox
- Name, or part of the name, of the CPU
- CPU in online mode

Filtering the display with the checkbox

The display can be restricted to CPUs that are activated in the checkbox column. The field in the header of the checkbox column provides a choice of the following filter values.

Table 3-46 Filter values in the checkbox column

Filter value	Filter result
All	All CPUs are displayed, irrespective of whether a check mark is set or not.
Set	Only CPUs with a set check mark are displayed.
Not set	Only CPUs without a set check mark are displayed.

If the dialog is called from the context menu of a CPU, this CPU is automatically preselected ("Context menu > Target device > Operating state").

All CPUs can be selected or deselected with the checkbox in the header.

Filtering the display by operating state or CPU name

The selection field in the header of the "Target device" column offers the following filter values:

Table 3-47 **Filter values in the Target device column**

Filter value	Filter result
All	All CPUs are displayed, regardless of whether they are in online or offline mode
Online	Only CPUs that are in online mode are displayed.
User-definable	The field can be edited, thus enabling filtering according to parts of names. Up to five user-defined filters remain for selection.

Operating states

State column: Shows the state of the CPUs in text form and via static LEDs.

Table 3-48 Operating states of a SIMOTION device

Operating state		Description
Text	**LED**	
STOP	(orange)	• Technology objects inactive (enables deleted, no axis motion) • User program is not executed • Loading a user program is possible • All system services are active (communication, etc.) • All analog and digital outputs set to "0" • The I/O modules (signal modules) are in the safe state (SIMOTION D)
STOP U	(orange/white)	• Technology objects active • Technology objects can execute jobs for test and commissioning functions • Otherwise identical to STOP operating state • STOP U means stop user program • User program is not executed
RUN	(green)	• Technology objects active • Execution of the user programs assigned to the execution system • Loading a user program is possible • The process image of the inputs and outputs is read or written
STOP	(orange/white)	• All tasks shut down, operating system stopped, real-time clock continues to run
STARTUP	(orange/white)	• The display only appears if the state persists longer than 1 second, or in the event of a fault
SERVICE	(orange/white)	• Display, e.g. if master control has been fetched by the axis control panel
SHUTDOWN	(orange/white)	• Display appears only if the state persists for longer than 1 second
(empty)	(empty)	• CPU is in offline mode

Control operating state

A CPU can be switched to the designated operating state with the RUN and STOP switches. The switching options are dependent on the position of the mode selector switch on the SIMOTION device. The setting on the device takes priority.

Table 3-49 Switching options of the software switch dependent on the position of the mode selector switch on the SIMOTION device.

Mode selector switch position of the SIMOTION device	Switching option in the SCOUT "Control Operating State" dialog
STOP	STOP
STOP U	STOP
RUN	RUN, STOP
MRES	STOP

Note

Manual switching to the "STOP U" operating state is no longer possible.

As of V4.4, manual switching to the "STOP U" operating state is no longer possible in the "Control Operating State" dialog of SCOUT.

If a CPU is in offline mode, both the RUN and STOP buttons are deactivated.

Simultaneous control of several CPUs

CPUs that are in online mode and that are selected in the filter column can be switched simultaneously to the RUN or STOP operating state. To do so, click the RUN or STOP switch below the list.

Observe that the switches only switch the CPUs that are visible in the dialog. For this reason, verify the effect of the name filter in the "Target device" column.

Error messages

A CPU that can no longer assume the required state is indicated in color. The "State" cell changes to red or yellow/orange.

- Red: Error
- Yellow/orange: Note

A single click on the corresponding cell causes an error text or information text to appear in a roll-out tip.

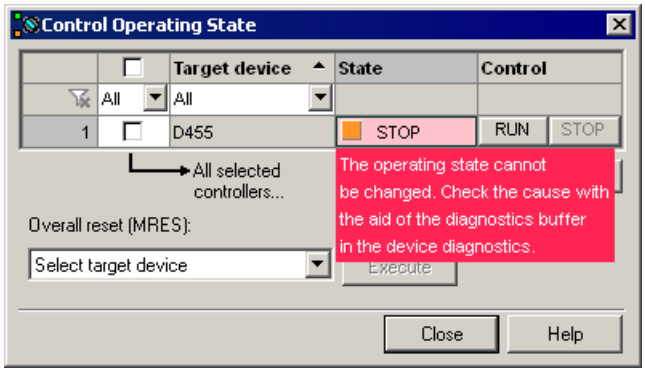


Figure 3-476 Roll-out tip with error message

Priority of the mode selector switch on the SIMOTION device

The setting of the operating state on the SIMOTION device has priority. SIMOTION SCOUT TIA can switch a SIMOTION device to the RUN operating mode only if the mode selector on the device is set to "0" or "RUN".

- **SIMOTION D**
  You can find the mode selector switch of the D410-2 and D4x5-2 in the lower area of the front behind the blanking cover.

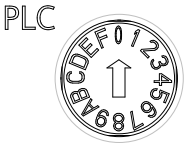


Figure 3-477 D410-2 and D4x5-2, mode selector switch, switch position 0 (RUN)

- **SIMOTION P**
  The LEDs, mode selector switch and CF card handling indicators, which are implemented as hardware on other SIMOTION platforms, are visualized on the screen for SIMOTION P.
  The display is performed via the *SIMOTION P State* application.
  The following figures show *SIMOTION P State* in the started state, using the SIMOTION P320-4 as an example.

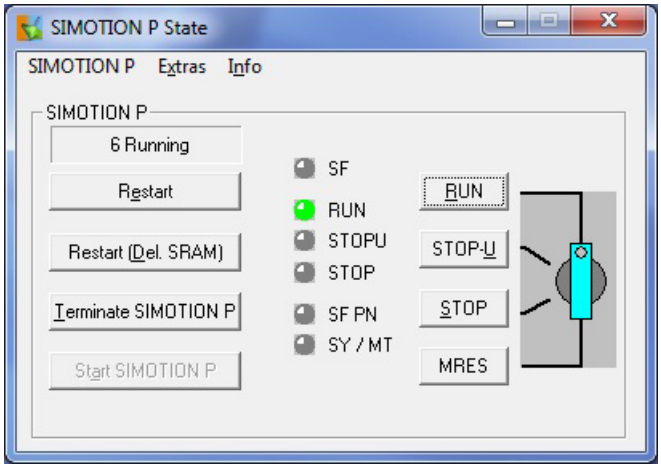


Figure 3-478 SIMOTION P State - state display

- **SIMOTION C**

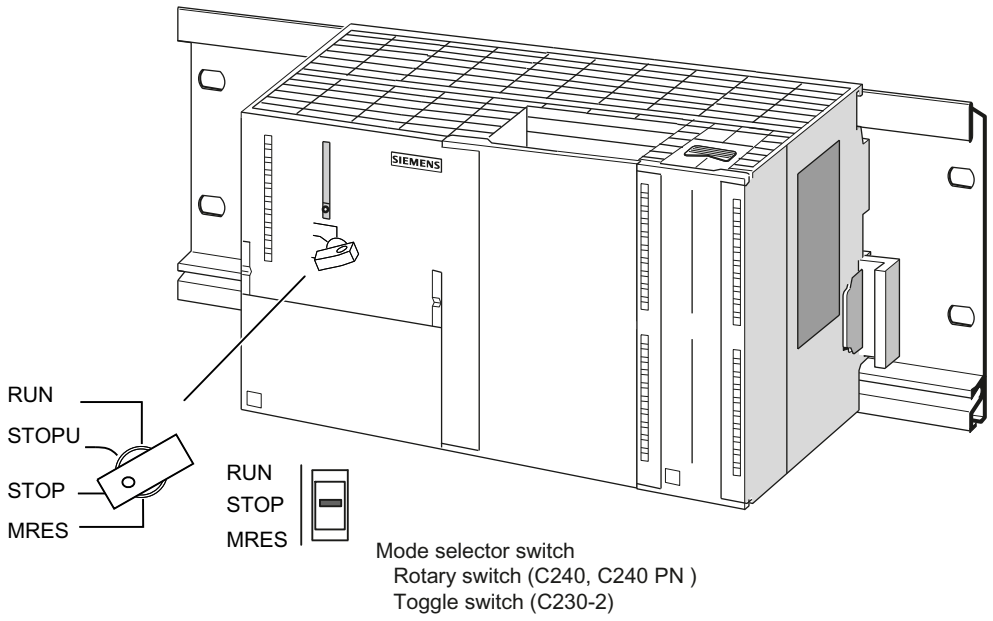


Figure 3-479 SIMOTION C module front

Additional references

Detailed information can be found in:

- SIMOTION C Operating Instructions
- SIMOTION D4x5-2 Manual
- SIMOTION D4x5-2 Commissioning and Hardware Installation Manual
- SIMOTION D410-2 Manual
- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- SIMOTION P320-4 E / P320-4 S Manual
- SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual
- SIMOTION SCOUT TIA Online Help

Overall reset

Requirement

The SIMOTION device for which overall reset is to be performed must be online.

Procedure

To execute an overall reset, proceed as follows:

1. Select "Target system -> Control operating state" in the menu to open the "Control Operating State" dialog.
   Or click "Control operating state" in the toolbar.

   The call is possible if at least one CPU of the project is in online mode.
2. Switch the SIMOTION device for which an overall reset is to be performed to the "STOP" operating state in the "Control Operating State" dialog.
3. Select the SIMOTION device under "Overall reset (MRES)". Click the "Execute" button. Confirm the command by clicking "Yes".

Result

The memory reset will now be performed.

See also

Control operating mode (Page 873)

3.5.8.9 Know-how protection

Know-how protection

The know-how protection in SIMOTION SCOUT TIA prevents unauthorized viewing and editing of your programs or parameters directly in the drive unit. A distinction is made between the following two know-how protection types:

- Know-how protection for programs
- Know-how protection for drive units (as of SINAMICS V4.5)

Note

Write protection for drive units

In addition to the know-how protection, write protection can be set up for drive units.

See also

Know-how protection for programs (Page 882)

Know-how protection for drive units (Page 885)

Write protection for drive unit (Page 888)

Know-how protection for programs

The know-how protection protects the programs and libraries in your project. Unauthorized viewing and editing of your programs is prevented when the know-how protection is activated.

Note

If a program is copied using copy and paste, the copied program also remains locked. A program retains its know-how protection even when it is imported or exported.

Setting up a password

By specifying a login and password, the know-how protection is activated for the program. The programs included in the project are visible to the user in the project navigator, but the programs are locked. Only by entering the password can you remove the lock and open the program for editing.

Procedure

To set up a password, proceed as follows:

1. Select "Project > Know-how protection for programs > Edit default login..." in the menu. The "Edit Default Login" dialog opens.

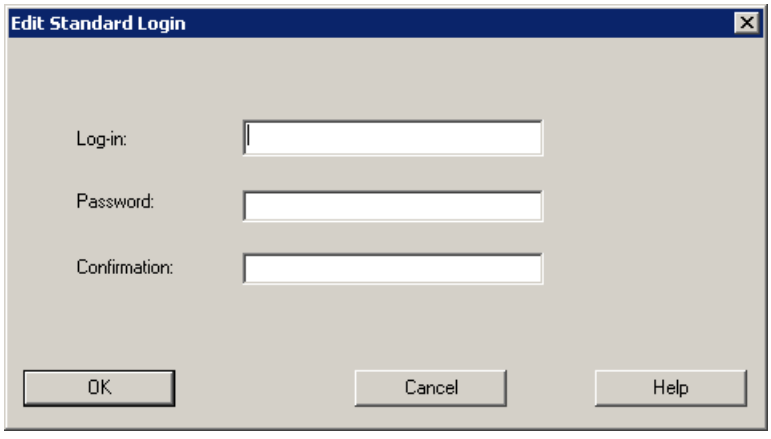


Figure 3-480 Setting up know-how protection password

2. Select the login and specify a password.

Note

The set default login remains valid for the whole project until it is deleted or SIMOTION SCOUT TIA is closed.

Configuring know-how protection

To configure the password security level, proceed as follows:

1. Select "Project > Know-how protection > Configure..." in the menu.
   The "Configure Know-how Protection for Programs" dialog opens.

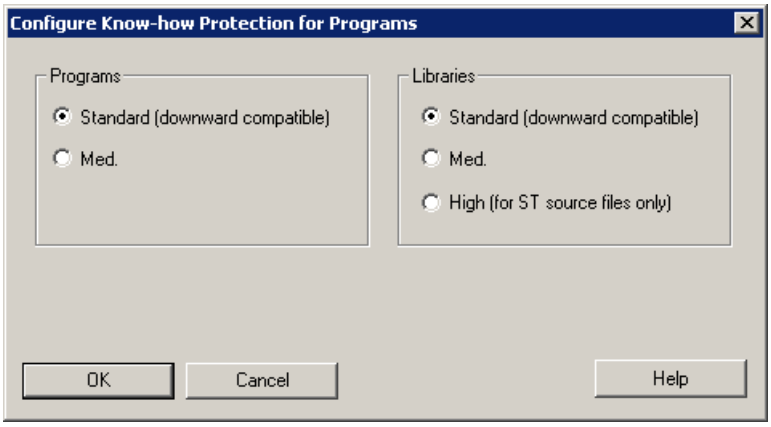


Figure 3-481 Configuring know-how protection

2. Select the required security level for programs and libraries.

Additional references

For detailed information, refer to Section "Know-how protection" in the SIMOTION SCOUT TIA Online Help.

Activating/ deactivating know-how protection

Activate know-how protection

Proceed as follows to activate know-how protection for programs:

1. Open the project and, in the project navigator, select the "PROGRAMS" folder or a program, depending on whether you want to set know-how protection for all programs or for selected individual programs.
2. Right-click to open the context menu and select "Know-how protection > Set".

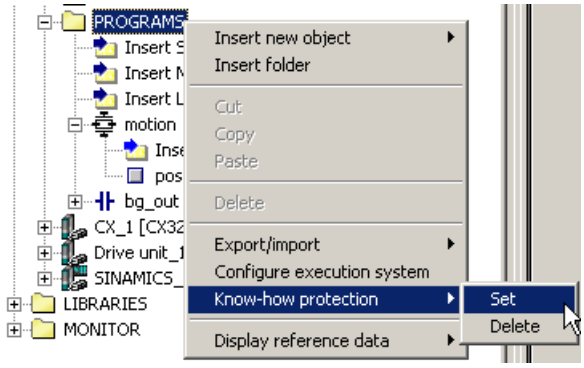


Figure 3-482 Set know-how protection

Result

The selected programs are protected and then locked. Protected programs are identified in the project navigator symbolically by a lock.

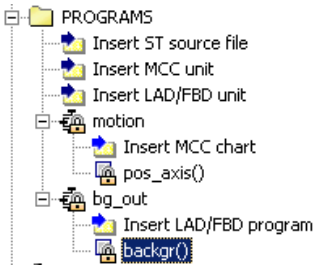


Figure 3-483 Result: Programs are protected

Only by entering the password can you remove the lock and open the program.

Deactivating know-how protection

Proceed as follows to deactivate know-how protection for programs:

1. Open the project and, in the project tree, select the "PROGRAMS" folder or the program, depending on whether you want to deactivate the know-how protection for all or only for specific programs.
2. Right-click to open the context menu and select "Know-how protection > Delete".

Result

The know-how protection for all programs or for the selected individual programs will be deleted.

Search in know-how-protected sources

If you have set know-how protection on sources, these are no longer included in the search. A search is only possible if you open the source in the appropriate editor or if you remove know-how protection again.

Note

Setting know-how protection for all programs

In ST source files, MCC charts and LAD/FBD programs, you can protect all programs in one action by setting the know-how protection directly in the "PROGRAMS" folder.

Know-how protection for drive units

The Drive unit know-how protection only applies online and is used to protect intellectual property, in particular, the know-how of machine manufacturers, against unauthorized use or reproduction of their products.

Activating know-how protection for a drive unit

Proceed as follows to activate the know-how protection for the drive unit:

1. Open the project, and in the project navigator, select the drive unit to which you want to apply know-how protection.
2. Right-click to open the context menu and select "Know-how protection for drive unit > Activate ...".
   The "Activate Know-how Protection for Drive Unit" dialog opens.

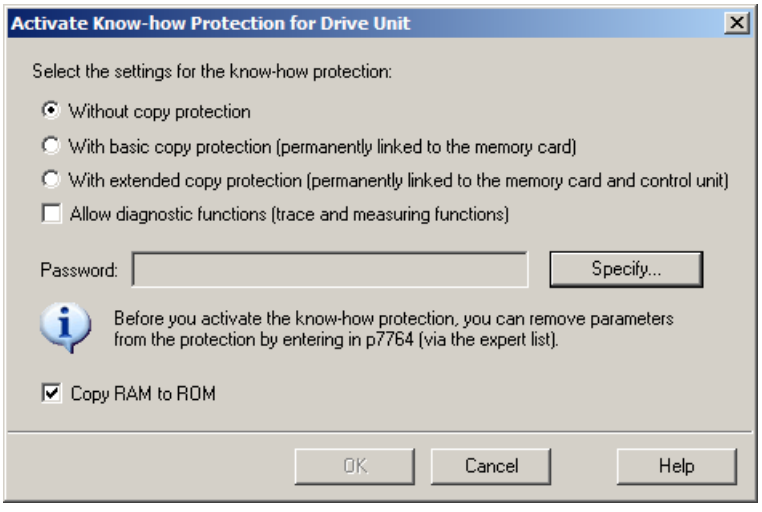


Figure 3-484 Know-how protection for drive unit

3. Specify whether the know-how protection should be set with or without copy protection.

4. Activate "Copy RAM to ROM" if the know-how protection is to be maintained permanently.
5. To set a password, click the "Set" button.
   The "Know-how Protection for Drive Unit - Define Password" dialog opens.
6. Define a password.
7. Confirm with "OK" to close the dialog.
8. Confirm with "OK" to activate the know-how protection for the drive unit.

Deactivating know-how protection for a drive unit

Proceed as follows to deactivate the know-how protection for the drive unit:

1. Open the project, and in the project navigator, select the drive unit whose know-how protection you want to deactivate.
2. Right-click to open the context menu and select "Know-how protection for drive unit > Deactivate ...".
   The "Deactivate Know-how Protection for Drive Unit" dialog opens.

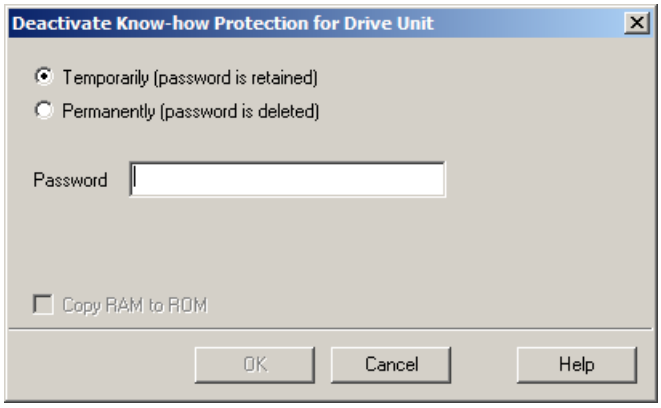


Figure 3-485 Deactivating know-how protection for a drive unit

3. Specify whether the know-how protection should be temporarily deactivated or whether the know-how protection should be deleted permanently.
   For temporary deactivation, you can activate the know-how protection via the context menu. Copying from RAM to ROM is not possible here.
4. Enter the password and confirm with "OK".

Additional references

For detailed information, refer to Section "Know-how protection" in the SIMOTION SCOUT TIA Online Help.

3.5.8.10 Access protection for technology objects

Overview

With the access protection for technology objects, you can protect important configuration data of the machine, e.g. axis settings, cams, coupling types, controller settings, etc., from unauthorized access.

With enabled access protection, the following actions can only be performed after input of a password:

- Uploading data from the controller
- Reading out data with the Web server
- Reading/analyzing data from a CF card

Note

Protect memory card from unauthorized access

It makes sense to extend the security measure and ensure that unauthorized persons have no access to the memory card.

Note

Longer controller ramp-up times

Please note that ramp-up may take longer for controllers with enabled access protection.

Managing access protection

Access protection is based on SIMOTION IT and is managed via SIMOTION IT.

You set up access protection via "Manage Config > SIMOTION IT > Access Control".

We distinguish between the following applications:

1. No access protection is set up yet.
   Set up access protection via "Manage Config > SIMOTION IT > Access Control" and assign a password.
2. Access protection is already set up. You as user are authorized to access the configuration data and have logged in successfully.
   In this mode, you can remove access protection, change the password for access protection, or enable access protection.
3. Access protection is set up and active. You have not yet logged in with a password.
   On the login page, enter your login data with password to log in.

When access protection is enabled and without login with password, the following functions are blocked:

- IT-Diag
- FTP (even when enabled in HW Config)

- OPC UA (even when enabled in HW Config)
- Upload of SIMOTION SCOUT TIA

Additional references

You can find detailed information on access protection in the "SIMOTION IT Diagnostics and Configuration" Diagnostics Manual and in the SIMOTION SCOUT TIA online help.

3.5.8.11 Write protection for drive unit

Drive unit with activated write protection

As of SINAMICS V4.5 there is a write protection for drive units. You can only activate/deactivate this write protection in online mode.

The write protection applies only online and is used for protection against accidental changes by the user. When write protection is set, p parameters can be read, but not written.

Activate/deactivate the write protection via the "Write protection for drive unit" context menu of the drive unit or via parameter p7761 of the expert list. A password is not required to activate/deactivate the write protection. You can query the status of the write protection via parameter r7760.0 in the expert list (value = 0 means that the write protection is not set).

Note

Permanent write protection

If you want to activate the write protection permanently, execute the "Copy RAM to ROM" function.

Features for drive units with activated write protection

Note the following restrictions for activated write protection of drive units:

- The write protection for drive unit always applies to the device, not to individual drive objects.
- Write protection only applies online.
- All r parameters can still be read and are visible as usual.
- All p parameters can be read, but not written. p parameters are displayed hatched.
- Parameters with the WRITE_NO_LOCK attribute can be written despite the write protection.

The following functions are available even when write protection is activated for the drive units:

- Restoring factory settings
- Upload
- Acknowledging alarms
- Control panel

- Trace
- Function generator
- Measuring functions
- Reading out the diagnostic buffer

Note
Restoring factory settings

You can restore the factory settings via the expert list (parameters p9, p976).

The following functions are not available when write protection for drive unit is activated:

- Download
- Automatic controller setting
- Stationary/rotating measurement
- Delete fault buffer (p952 must not be defined as an exception parameter)
- Delete drive unit completely or individual components thereof
- Rename drive unit

Activating/deactivating write protection for drive unit

Activate write protection

Proceed as follows to activate the Drive unit write protection:

1. Connect to the target device online.
2. Select the drive unit to which write protection is to be applied in the project navigator.
3. Right-click the drive unit and select "Drive unit write protection > Activate" in the context menu.

Result

Write protection has been activated.

Copy RAM to ROM

You can select whether the write protection for drive unit is to be set permanently by activating "Copy RAM to ROM".

Deactivating write protection

Proceed as follows to deactivate the Drive unit write protection:

1. Connect to the target device online.
2. In the project tree, select the drive unit, whose write protection you want to remove.
3. Right-click the drive unit and select "Drive unit write protection > Deactivate" in the context menu.

Result

Write protection has been deactivated.

Note

Activating/deactivating write protection in the expert list

You can activate and deactivate the Drive unit write protection also directly in the expert list via the parameter p7761. No password is required to modify the parameter p7761. Similarly, you can activate and deactivate the Drive unit write protection via script.

Additional references

For detailed information, refer to Section "Drive unit write protection" in the SIMOTION SCOUT TIA Online Help.

3.5.8.12 Configuring multilingual messages

Multilingual messages and comments can be configured with the "Language-dependent texts" function. This is necessary, for example, if you want to output multilingual user-defined Alarm_S messages or diagnostic buffer entries.

Procedure

To configure multilingual messages, proceed as follows:

1. Select "Project > Language-dependent texts..." in the menu.
   The "Select Language" dialog box opens.
2. At "Relevant text sources", select the source for which you want to configure multilingual texts, e.g. Alarm_S.
3. Add the languages for which you want to configure texts in the selected text source. To do so, click the (>) arrow key to transfer the languages from the "Other languages" field to the "Languages in the project" field.
   Dual-language messages, for example, German and English, are then in the "Languages in the project" field.

4. Set the current configuring language. To do so, select the language in the "Languages in the project" field and click the arrow-down key (v). The selected language is displayed as "Current language".
   The language-dependent texts for the selected text source are now configured in this language setting.
5. Confirm with "OK".
6. Select "Project -> Messages > Configuration" in the menu.
   The "Message Configuration" dialog box opens.
7. Check the language set at "Configuration language".
   The message texts and remarks are displayed in the language in which they were originally configured. These texts must then be translated into the new configuration language.
8. Double-click the messages for which you want to enter texts in the new configuration language.
   The "Edit Message" dialog box opens.
9. Enter the new message text and remarks in the changed language.
   The symbol name must not be changed in foreign language messages, as this symbol name is used for programming.
   Only the message text and remarks are language-dependent.
10. Confirm with "OK".
    The modified message text is entered in the Message configuration table.

Exporting and importing text sources for translation

Using the "Language-dependent texts... > Export" function, you can export the language-dependent texts within the same project for translation, and then import them again.

Proceed as follows to export text sources:

1. Select "Project > Language-dependent texts..." in the menu.
   The "Select Language" dialog box opens.
2. At "Affected text sources", select the source that you want to export for translation, e.g. Alarm_S.
3. Click the "Export..." button.
4. Navigate to the target directory and assign a file name.
5. Confirm the selection with "Save".
   A *.txt text file has been imported that contains placeholders for all languages defined in the project.

Proceed as follows to import the translated text sources again:

1. Select "Project > Language-dependent texts..." in the menu.
   The "Select Language" dialog box opens.
2. Click the "Import..." button.
3. Navigate to the translated text file.

4. Click "Open".
   The translated text source has been imported and is available in the project in the respective configuration language.
5. Select "Project > Save and compile changes" in the menu to save the changes.

Additional references

Detailed information can be found in Section "Message configuration" and in Section "Exporting/ importing language-dependent texts" in the SIMOTION SCOUT TIA online help.

Observe in particular the constraints applying to the use of language-dependent texts described in the online help under "Notes on language-dependent texts".

3.5.8.13 Exporting OPC data

Overview

The OPC interface (**O**LE for **P**rocess **C**ontrol) is the specification of a common, manufacturer independent software interface based on OLE (**O**bject **L**inking and **E**mbedding). It has been designed as an industry standard by leading automation companies with the support of the Microsoft Corporation.

Up to now, applications that accessed process data depended on manufacturer-specific interfaces. With the standardized OPC interface, you can now access process data uniformly via the OPC interface when using, for example, operator control and monitoring software.

For OPC data export, the following symbols are exported in the symbol file:

- System variables of the device and the technology objects
- Global device user variables
- Symbolic input and output variables
- Interface variables of user programs (ST source file, MCC, LAD/FBD)

Note

In the interface area of a program, a maximum of 64 KB data can be addressed for operator control and monitoring. Data beyond this area get lost during the export.

In addition, the following data can be exported for OPC Alarm/Event:

- Configured alarms (Alarm_S/Q)
- Alarms from technology objects (technology object alarms)
- All messages/alarms of the diagnostics buffer

The data is saved as an XML file in the selected target directory (OPC_AE.xml).

Exporting OPC data

Procedure

Proceed as follows to export the OPC data:

1. In SIMOTION SCOUT TIA, select "Options > Export OPC data..." in the menu.
   The "Setting the Data for Export" dialog opens.

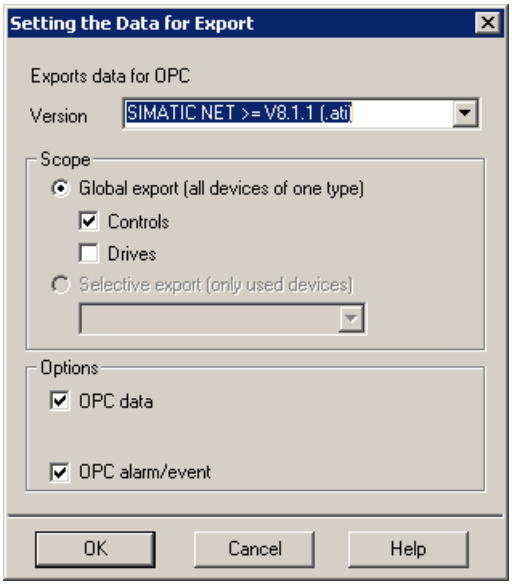


Figure 3-486 Exporting OPC data

2. Select the intended use of the data to be exported and confirm with OK.
   The "Exporting OPC data" dialog opens.
3. Select a target directory and confirm with "OK".
4. If several interfaces with different bus addresses are available on the device, also select the protocol, e.g. PROFIBUS or TCP/IP, and the interface for the export. The "Parameterize Interfaces" dialog opens. If the interfaces possess the same bus addresses or only one is being used, this dialog will not be displayed. Select the interface and protocol and confirm with "OK". An interface must be selected for all SIMOTION devices in the project. Repeat this step when required.
   The export is started. A successful export is displayed in the "Export OPC data status display" tab in the detail view.
5. If the SIMOTION devices are distributed across multiple subnets and you want to address a SIMOTION device with a SIMATIC OPC server via a router, you must configure this router again in a window after the file export in SIMOTION SCOUT TIA. If you do not want to use a router, confirm the dialog with "No".
   The following information is displayed in the "OPC Routing Information for HMI Devices" dialog:
   - All configured networks
   - All SIMOTION devices contained in the project
6. First select the subnet (location) of the OPC server.

7. Select the network station and then the first router for each network station to be addressed via the OPC server in the displayed window.
8. Confirm with "OK".

Setting the data to be exported

Note

OPC data can only be exported if a project is open. The export is possible both in offline and online mode.

Note

Default export setting

The latest version is set as default for the OPC export. As of SIMATIC Net V8.1.1 the OPC server can use the format "*.ati" and as of SIMOTION SCOUT TIA V4.4.

The data to be exported is selected in this window.

The following parameters can be selected:

Field/button	Meaning/note
Exports data for SIMATIC NET OPC	
SIMATIC NET versions	Select the SIMATIC NET version for which the export data is to be used. Different versions are not fully compatible. The SIMATIC NET >= V8.1.1 (.ati) Export is the fastest export version. The symbol file to be exported is first configured in the main memory and then exported. For this operation, you require sufficient main memory for the number of symbols. Before the export is started, the maximum number of symbols that can be exported is displayed. Check first whether the maximum possible number of symbols is greater than the number of symbols in your export.
Scope	
Global export	Under "Global export", activate the data (control data, drive data) that you want to export. All the required variables of the controller (SIMOTION CPU) are exported with the control data. Only drive data of drives of the SINAMICS family can be exported. Whereby the drive must be inserted below a SIMOTION CPU. The export of data of single drives and drives below a SIMATIC CPU is not possible.
Selective export	Activate "Selective export" if you only want to export certain variables. You must first have compiled these variables in watch tables in SIMOTION SCOUT TIA. In the selection list, you can then select the watch table whose variables are to be exported. A selective export is only possible when you have inserted watch tables in the project. You can create watch tables in the project tree by clicking "Insert watch table" in the "MONITORING" folder. The drive parameters are copied to the watch table using the expert list of the associated drive object.
Options	

Field/button	Meaning/note
OPC data	Activate OPC data if the symbol names and variable names are to be exported from the current SIMOTION project. If the option is deactivated, no data is exported.
Arrays with single elements	Activate the option if the arrays' single elements are to be exported with them. If the option is deactivated, only the first address of the array is exported and the data export can be shortened and the data volume reduced.
OPC alarm/event	Activate OPC alarm/event if you want to export the configured alarms and alarm texts. All existing languages of the alarms are exported to the file. If the option is deactivated, no alarms are exported.

OPC export - OPC data

The target directory is defined in the "Project Export" window.

You can set the following parameters:

Field/button	Meaning/note
Export settings	
Enter a target directory for the export	To select a target directory for the files to be exported, proceed as follows: 1. Click Browse to the right of the text field. A window opens. 2. Specify a target directory. 3. Confirm with "Open". Your selection is displayed in the text box.
Project path	The path of the currently opened project is displayed here. The symbol name of the current project is exported as a file or files.

Exporting OPC data, ati export (main memory)

SIMATIC NET >= V8.1.1 (.ati) export

The SIMATIC NET >= V8.1.1 (.ati) export is faster than the old versions of the OPC export. However, it requires more main memory as the symbol file must be configured in the main memory. For this operation, you require sufficient main memory for the number of symbols.

Therefore, at the start of the export, a calculation is made as to how many symbols can be exported with the available main memory. Check first whether the maximum possible number of symbols is greater than the number of symbols in your export. If the calculated number is not sufficient for the export of the project data, then you still have to use the other export versions, e.g. SIMATIC NET V7.x/V8.0.

Assigning the connection parameters

Assign the parameters for the connection in this window.

The following parameters can be selected:

Field/button	Meaning/note
Device	The device for which you must select the protocol and the interface is displayed here. You must specify the interface settings for each device in the project. You must specify these settings in this window depending on the number of devices.
Connection name	With V4.3, it is possible to use device names that do not satisfy the ST conventions. If the name of the device violates these conventions, a name valid for the OPC export is displayed at "Connection name". The following is modified in the name: • Invalid characters are replaced by a "_". • If the first character is not a letter, an "A" is set as prefix. • A number (_<n>) is added at the end, if not already available, so that the connection name is unique. You can rename the name modified in this way. Since the connection name and the device name may be different, the device/connection assignment is also exported in a file (OPC_device_connection.dat).
Protocol	Selects the transmission protocol for the SIMOTION device. **PROFIBUS** Protocol for PROFIBUS **TCP/IP** Protocol for Ethernet
Interface	Use "Interface" to select the interface for the symbol names to be exported. The SIMOTION device is addressed via this interface from the HMI software. Each symbol name must be uniquely assignable to a hardware address (bus address). This assignment is set via the "Interface" selection. **Notice** : If the interface is parameterized for isochronous bus cycle, then it must not be used for an OP.

OPC routing information (routers)

If the SIMOTION devices are distributed across multiple subnets and you want to address a SIMOTION device with a SIMATIC OPC server via a router, you must configure this router again in a dialog after the file export in SIMOTION SCOUT TIA.

In the "OPC routing information for HMI devices" dialog, the following is displayed:

- All configured networks
- All SIMOTION devices contained in the project

First select a location of the OPC server and then the first router for each node to be addressed via the OPC server.

Several projects interconnected

To communicate with the SIMOTION devices in a network interconnection through several projects, the SIMATIC NET OPC server must be forwarded the following files and items of information:

- "OPC Alarm/Event" files, which may originate from several SIMOTION SCOUT TIA projects
- Time zones
- Router

An auxiliary program, "SIMOTION OPC File Manager", enables you to configure this data for the SIMATIC NET OPC server.

Exporting OPC data, status display

The status is displayed in the "Export OPC data status display" tab in the detail view during the export.

The following information can be displayed during export:

Information displayed	Meaning/note
Error messages	Error messages are displayed in plain text if export fails. The following errors are possible: • 64k overshoot. • Export file with symbol names already exists and is being used by another application.
Location and name of the exported data	Displays the storage location of the exported files after the symbol names have been successfully exported.

3.5.8.14 Licensing of the runtime components

Licensing of the runtime components

Overview for the licensing

Licensing of the technology functions

Functions can be licensed using the following software options:

- Motion control technology functions
  The licensing is performed axis-specifically for:
  - POS; use of the technology functions for positioning axis
  - GEAR; use of the technology functions for following axis and path axis
  - CAM; use of the technology functions for cam axis

The GEAR technology function contains the POS technology function, while the CAM technology function contains the POS and GEAR technology functions.

The MultiAxes package permits a simple licensing of the Motion Control technology functions. It contains the license for the unlimited use of the POS/GEAR/CAM technology function on a SIMOTION device, e.g. a D4x5-2.

- TControl technology function
  The TControl technology package functions are licensed on a channel-specific basis in packages of eight temperature channels.

Note

For SIMOTION C, D and P, licenses for Runtime software can also be ordered as pre-installed software (via order code / Z option) on the memory card.

Additional references

You will find information on further licensed technology functions and detailed information on licensing the runtime software and the order data under:

- SIMOTION, SINAMICS S120 & SIMOTICS, equipment for production machines, PM 21 catalog
- PM 21 catalog, section titled SIMOTION runtime software
- Configurator for SIMOTION Runtime licenses in the Industry Mall (http://mall.industry.siemens.com)

Licenses and license key

Depending on the type and number of runtime components used in the project, licenses must be acquired as part of the licensing procedure for SIMOTION. The licenses required for a SIMOTION device are assigned to a license key. This license key is stored on the storage medium of the SIMOTION device during the licensing procedure.

There are two ways of ordering licenses:

- Preinstalled licenses
  The license key is already stored on the card.
- Ordered licenses (Certificate of License)
  These licenses must be assigned to the storage medium using the Web License Manager. The Web License Manager can be found on the Internet under:
  Web License Manager (http://www.siemens.com/automation/license).

  Note

  Web License Manager login

  You can find the required login data, e.g. license number and the delivery note number, on the license sticker on the software.

  The license key is transferred to the hardware with SIMOTION SCOUT TIA.

You require the following information to obtain the license key:

- The serial number of the SIMOTION device memory medium
  You can obtain the serial number from the memory medium or have it displayed online in the SIMOTION SCOUT TIA (licensing wizard).
- The serial number of the CoL (Certificate of License)
  You have this number on paper.

Table 3-50 The serial number on the SIMOTION hardware assigned to the SIMOTION device

SIMOTION device	Hardware serial number of the module
SIMOTION C2xx	SIMOTION Micro Memory card
SIMOTION D	SIMOTION Compact Flash card
SIMOTION P	You can find this information on the rating plate of the SIMOTION P, starting with SVP...

License keys can be generated separately from the licensing.

Note

Archiving the licensing data

When the SIMOTION memory card is deleted or formatted, the licensing data is also deleted. Archive the licensing data in order to be able to transfer it again to the memory medium in such a case. If the data is not backed up, you have to perform the licensing again. You can display the entered license key in the Web License Manager.

You will find additional information in Section License key delete protection (Page 902).

Determining licensing requirements

License requirement

Determine your license requirement only if you have completed the configuration, and before you load it into the target device! Before you begin, the project must have been saved and compiled.

If you have not acquired any licenses yet, under-licensing is displayed.

You can determine the licenses in the following ways:

- **Offline mode** with open project
  The required licenses are displayed.
- **Online mode** with open project
  A comparison of the required and actual licenses is displayed.
- **Online mode** without project
  The actual licenses of the selected SIMOTION device are displayed.

Procedure

To determine the license requirement, proceed as follows:

1. Select the SIMOTION device in the project tree.
2. Select "Licenses" in the context menu.
   The required licenses for the project or a comparison of required and actual licenses are displayed.
3. You can close the window with "X" or continue with "Perform licensing...".

The license check, i.e. the inspection of the license key, is carried out in the target system. Possible responses in the case of under-licensing can be found in Section Underlicensing (Page 902).

Memory cards can be purchased with integrated runtime licenses, which do not require separate licensing.

Note

A license which has already had a license key allocated can no longer be taken.

It is possible, however, to add licenses to an already generated license key.

Display the available licenses of the SIMOTION device

Displaying via accessible nodes

You can use the list of "Accessible nodes" to determine the specific licenses that have already been assigned to the SIMOTION device. You can access the data of the SIMOTION device directly.

Note

This step is not necessary if the required and actual licenses are displayed within a project.

Requirements

- SIMOTION SCOUT TIA has been started.
- SIMOTION SCOUT TIA is in offline mode.

Procedure

To display the licenses assigned to the SIMOTION device, proceed as follows:

1. Select the menu "Project" > "Accessible nodes" menu.
   The list of accessible nodes is displayed in the working area.
2. Select the relevant SIMOTION device.
3. Select the "Licenses" command in the context menu.
   The "Licenses" dialog opens. The actual licenses of the selected SIMOTION device are displayed.
4. You can close the window with "X" or continue with "Perform licensing...".

Performing the licensing

If there are no pre-installed licenses, you can acquire the licenses you need and then generate the license key required.

Requirements

Licensing is subject to the following requirements:

- The configuration has been completed.
- The project has been saved and compiled.
- The required licenses have been determined.
- The license key has been determined or the serial numbers of the memory medium and the CoL are available.
- SIMOTION SCOUT TIA is in online mode.

Procedure

To perform the licensing, proceed as follows:

1. Select the relevant SIMOTION device in SIMOTION SCOUT TIA.
2. Select the "Licenses" command in the context menu.
3. In the "Licenses" dialog, click "Perform licensing...".
   If the "Use wizard" option is activated, a wizard guides you through the licensing procedure.
   If the option is not activated, the window for expert licensing opens. You can enter the license key there without running through the wizard.
   If you have not yet generated the license key, the wizard gives you the option of switching to the Web tool to generate a license key. Then return to the wizard.
4. If you have an online connection, continue with item 5.
   Otherwise, you can establish an online connection with "Online" in the (Step 2 of 3) window.
5. Enter the license key in the (Step 3 of 3) window.
6. Click the "Finish" button.
   The wizard closes. Licensing is complete.

Note

The license key is written to the retentive memory when the project data is transferred to the target system.

Changing the license key

Through changes in the project, such as through the inclusion of an additional axis, the license key for the project can lose its validity. This is why under-licensing is displayed when the project is downloaded and the SF LED flashes at 0.5 Hz.

After you have determined the actual requirement and purchased the necessary licenses, generate the license key again. Now replace the license key already entered with the newly generated one.

Licensing during hardware replacement

For the replacement of licensed SIMOTION components (MMC, CF, etc.), the associated license key must be assigned to the new SIMOTION component. In this case, contact the Customer Support for assistance.

License key delete protection

The license key is stored in the "KEYS" directory on the SIMOTION Memory Card.

When the controller starts up for the first time, the license key will be saved in the boot sector of the card and from this time is protected from being lost.

Deleting the license key in the boot sector through user operation is not possible, also not by formatting the card or by using the "Write boot sector..." function.

If the license key is no longer present on the card, it will be written again during startup from the boot sector to the "KEYS" directory. This means that the system will repair any deletion on the "Key" file.

The license key can be changed at any time, e.g. through relicensing. At the next startup, the license key will be saved again in the boot sector.

Underlicensing

If SIMOTION SCOUT TIA detects the presence of under-licensing during license verification, an entry is made in the diagnostics buffer. The verification is repeated every hour, and an entry is made in the diagnostics buffer each time under-licensing is detected.

The following information can be read from the diagnostics buffer entry:

- Number of required licenses
- Number of actual licenses
- Operating mode

As an additional warning signal, the SF LED flashes at 0.5 Hz as long as under-licensing is present on the system. Underlicensing will only be displayed if no acknowledgeable technological event is pending, as the same SF LED is used to indicate this as well.

See also

Determining licensing requirements (Page 899)

3.5.9 Service and diagnostics

3.5.9.1 Diagnostic functions in SIMOTION SCOUT TIA

There are extensive diagnostic options in the online mode within the SIMOTION SCOUT TIA for the operation of SIMOTION devices. These diagnostic options are summarized in the diagnostics overview:

- The diagnostics overview is a tab in the detail view and is available by default in the online mode. You can call up detailed displays from here.
- An "Alarms" tab is also available in the detail view. This provides a tabular overview of
  - Technological alarms (from technology objects)
  - Alarm_S messages (from user programs)
    The alarms can be acknowledged either individually or all together.
- The "Address list" tab in the detail area offers extended functions in terms of I/O diagnostics and hardware availability.
- With the device diagnostics, you determine extensive diagnostic information (e.g. diagnostics buffer, system utilization, task status, etc.).
- You can record signal charts with the trace tool. The values of system variables can be recorded during runtime for diagnostic purposes.
- Program testing and debugging, e.g. modify variable, program status, breakpoints.
- Back up diagnostic data including non-volatile data (retain data) to CompactFlash card (for SIMOTION D) or MMC (for SIMOTION C).
- Back up the HTML pages, including the current contents for diagnostic purposes, on the CF card or MMC.
- Restore backed up non-volatile data (retain data).
  Further information can be found in the FAQ section on the Utilities & Applications DVD under: FAQs > Engineering > Backing up diagnostic data and non-volatile data.
  SIMOTION IT also offers comprehensive diagnostic options that can be easily accessed via an Internet browser.

Additional references

Detailed information can be found in:

- SIMOTION ST Structured Text Programming and Operating Manual
- SIMOTION MCC Motion Control Chart Programming and Operating Manual
- SIMOTION LAD/FBD Programming and Operating Manual
- SIMOTION IT Ethernet-based HMI and Diagnostic Function Diagnostics Manual
- SIMOTION SCOUT TIA Online Help

3.5.9.2 Using the diagnostics overview

The diagnostics overview is available as a tab in the detail view when the project is in online mode.

- In the detail view, select the "Diagnostics overview" tab.

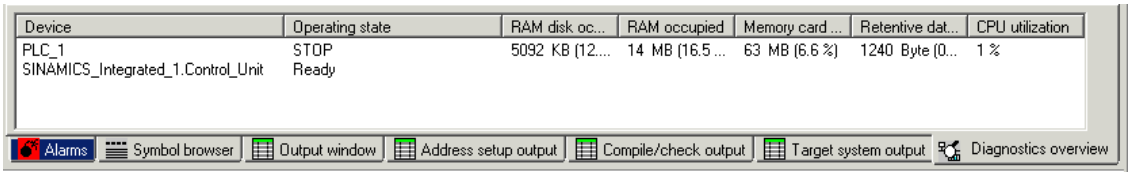


Figure 3-487 Diagnostics overview in the detail view (online mode)

The following are displayed for each accessible SIMOTION device:

- Operating mode
- Memory used (absolute and percentage display)
  RAM disk, RAM, memory card, retentive data
- CPU utilization (percentage display)

The drive units are also displayed. To obtain a detailed display of the individual devices, open the device diagnostics.

3.5.9.3 Device diagnostics

In online mode, the device diagnostics function enables you to obtain a comprehensive display of diagnostics results of the individual SIMOTION devices.

To display the diagnostic results of the individual devices, follow these steps:

1. Select the desired SIMOTION device in the project navigator.
2. Select the "Target system > Device diagnostics" menu.
   Or double-click the SIMOTION device in the Diagnostics overview tab in the detail view.

Note

Device diagnostics for several SIMOTION devices

You may open the device diagnostics for several SIMOTION devices simultaneously. This allows you to compare different devices.

You can also access these device diagnostics via the "Accessible nodes" function.

The "Device Diagnostics" window will open in the working area of the workbench. This window provides you with the following information:

- General information
- Diagnostics buffer
- Task runtimes
- Memory utilization
- System utilization
- Userlog file

- Syslog file
- Alarms

You can work with this information as follows:

- Print:
  Select the "Project > Print" menu.
- Save it as a text file:
  Click "Save".
- Refresh:
  Click "Refresh" or press the <F5> function key.

You can also monitor and change the operating state:

- Click "Control operating state".

General information on the SIMOTION device

To obtain general information on the SIMOTION device, proceed as follows:

- Select the "General" tab in the device diagnostics.

The following information will be displayed:

- Name and system ID of the SIMOTION device
- Operating state of the SIMOTION device
- MAC addresses
- IP addresses
- Subnet mask

- Standard gateway
- Article numbers and designations of the deployed components, e.g. SIMOTION device, Motion Control technology package.

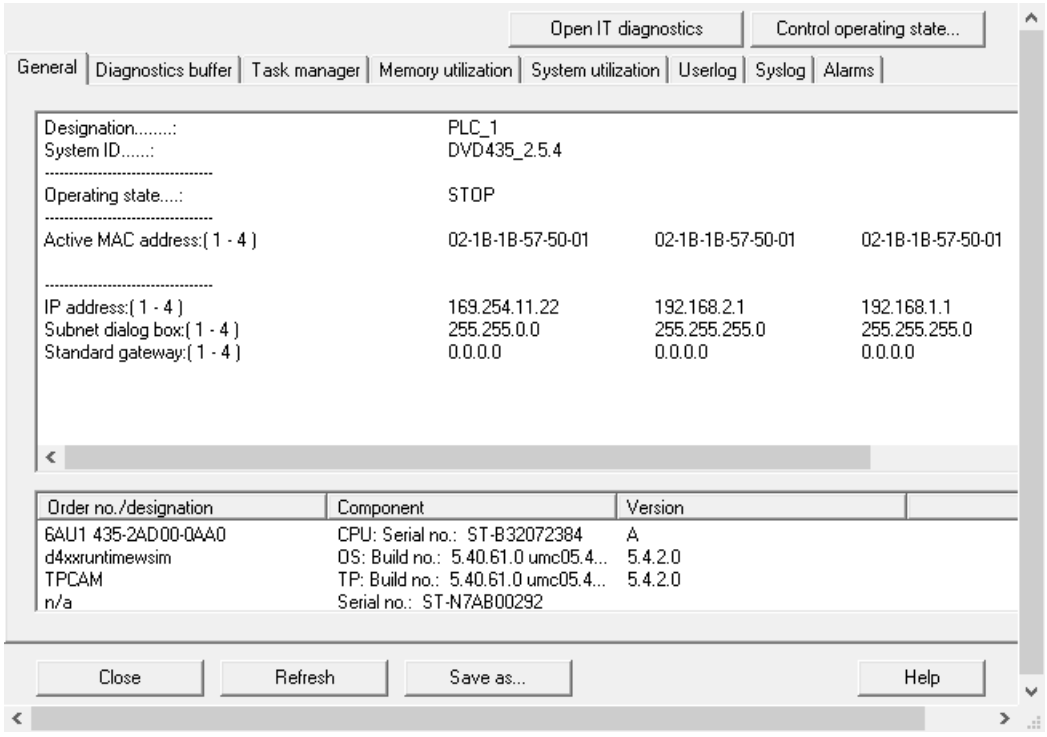


Figure 3-488 Display of general information in the device diagnostics

Diagnostics buffer

The diagnostics buffer is part of the system status list. It is possible to jump to the error position from the diagnostics buffer. It logs important events (e.g. changes in module state) in the order in which they occur. These include the following:

- Faults in a module
- Faults in the process wiring
- System errors in the CPU
- CPU operating state transitions
- User-defined diagnostic events
- Technology object alarms
- Alarm_S messages
- Errors in the user program
- User-defined entries with the _writeAndSendMessage() function
- Compatibility errors, e.g. between the drive software and SIMOTION (SIMOTION D)

To display the diagnostic buffer events, proceed as follows:

1. Select the "Diagnostics buffer" tab in the "Device Diagnostics" dialog.
   The saved events are displayed in tabular form.
2. Select the event for which you want to display more information.
   Detailed information for the selected event is displayed in the lower pane of the window.

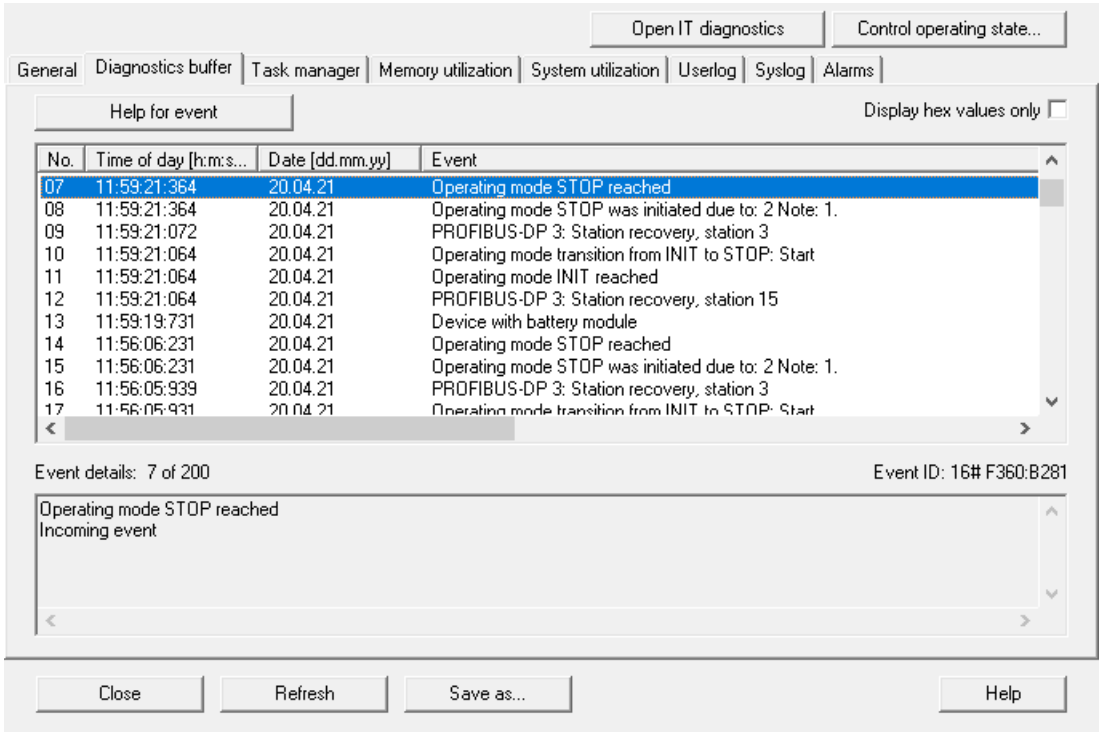


Figure 3-489 Example of the diagnostics buffer display in the device diagnostics

Task Manager

You can display the task runtimes and the status of the tasks set up in the project if you are connected online with the device. The resolution of the displayed task runtimes is performed in the servo cycle clock.

Note

The task runtimes are calculated to the µs and indicate the effective level runtime of the respective task (including the interrupt times). Thus, these correspond to the values of the "effectiveTaskruntime" device variables.

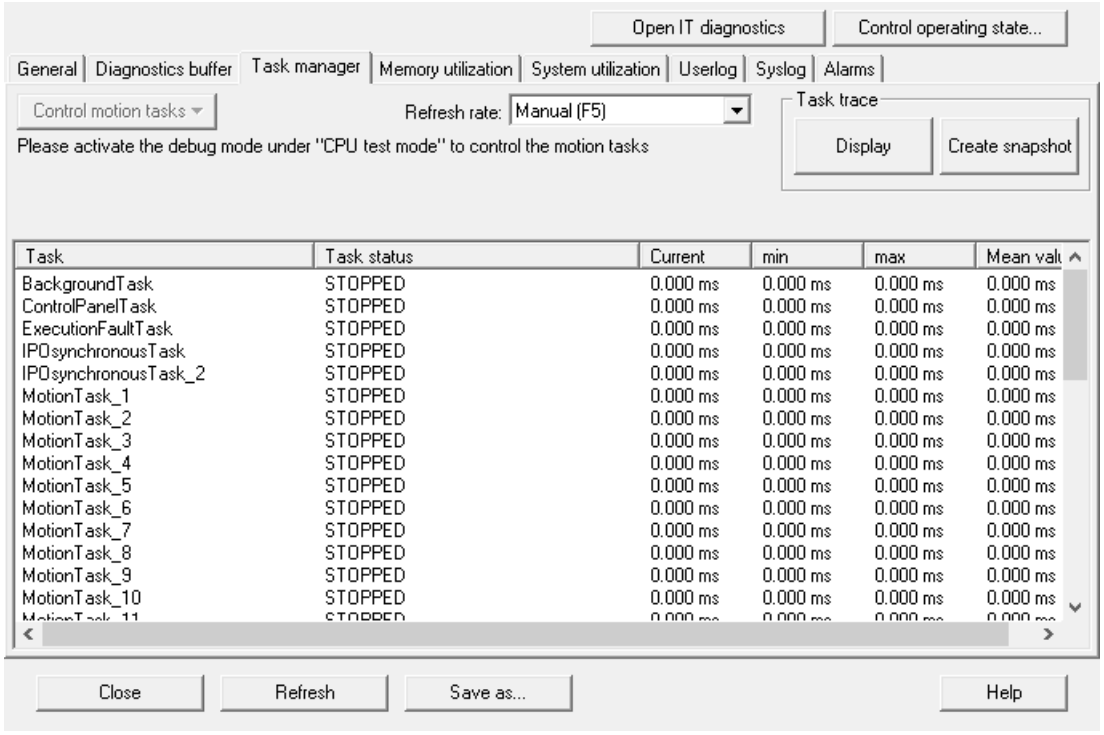


Figure 3-490 Example of the displayed task runtimes in the device diagnostics - STOPPED status

The display is refreshed according to the refresh rate selected. The status and the following values are then displayed:

- Current runtime (current):
  Value of last polling
- Minimum runtime (min.):
  Minimum value since last transition from STOP to RUN
- Maximum runtime (max.):
  Maximum value since last transition from STOP to RUN
- Mean runtime (mean value):
  Value averaged from the last 10 cycles

The runtimes measured include the interruptions by higher-priority tasks.

Meaning of the various status displays:

- RUNNING (TASK_STATE_RUNNING)
  Task running, e.g.:
  - By "_startTask" function
  - As an active cyclic task
- RUNNING_SCHEDULED (TASK_STATE_RUNNING_SCHEDULED)
  Task is interrupted by the system.
  If the task status RUNNING_SCHEDULED remains pending for a long time, it identifies a long running user task, e.g. a programmed endless loop.

- STOP_PENDING (TASK_STATE_STOP_PENDING)
  Task has received signal to stop; it is in a state between RUNNING and STOPPED.
  Actions may be performed until the task has stopped.
- STOPPED (TASK_STATE_STOPPED)
  Task stopped (e.g. via "_resetTask" function), completed or not yet started.
- SUSPENDED (TASK_STATE_SUSPENDED)
  Task suspended by function "_suspendTask".
  Use "_resumeTask" *(name)* to cancel this command. The task then resumes from the point at which it was interrupted.
- WAITING (TASK_STATE_WAITING)
  Task is waiting due to the function "_waitTime" or "WAITFORCONDITION".
- WAITING_FOR_NEXT_CYCLE (TASK_STATE_WAIT_NEXT_CYCLE)
  TimerInterruptTask waiting for start trigger.
- WAITING_FOR_NEXT_INTERRUPT (TASK_STATE_WAIT_NEXT_INTERRUPT)
  SystemInterruptTask or UserInterruptTask is waiting for the triggering event to occur. When an interrupt occurs, the SystemInterruptTasks are started and executed once. Up to eight incoming interrupts can be stored in the buffer. If another interrupt occurs, the buffer overflows and the CPU goes into STOP operating state.
- LOCKED (TASK_STATE_LOCKED)
  Task locked by function "_disableScheduler".
  This status prevents the activation of all user tasks (except the IPOSynchronousTask and IPOSynchronousTask_2) until the "_enableScheduler" command is called. It does not, however, affect system tasks. The time watchdog for cyclic tasks is **not** suspended.

  Note

  This also prevents the activation of the SystemInterruptTasks and UserInterruptTasks.

Controlling MotionTasks

It is possible to control MotionTasks via SIMOTION SCOUT TIA without a user program. Consequently, you can test programs and influence MotionTask sequences in a very specific way. Selected MotionTasks can be stopped, and locked or restarted for the sequence.

This means that programs in MotionTasks can also be downloaded in RUN mode. If you have made changes to sources and want to reload them in RUN mode, an active MotionTask can prevent this. To avoid this problem, you can terminate MotionTasks specifically with SIMOTION SCOUT TIA and then carry out the download in RUN mode.

Additional references

More detailed information on downloading in RUN mode can be found in:

- SIMOTION Runtime Basic Functions Function Manual
- SIMOTION SCOUT Task Trace Function Manual

Checking memory utilization

To display the memory utilization, proceed as follows:

1. Select the "Memory utilization" tab in the device diagnostics.

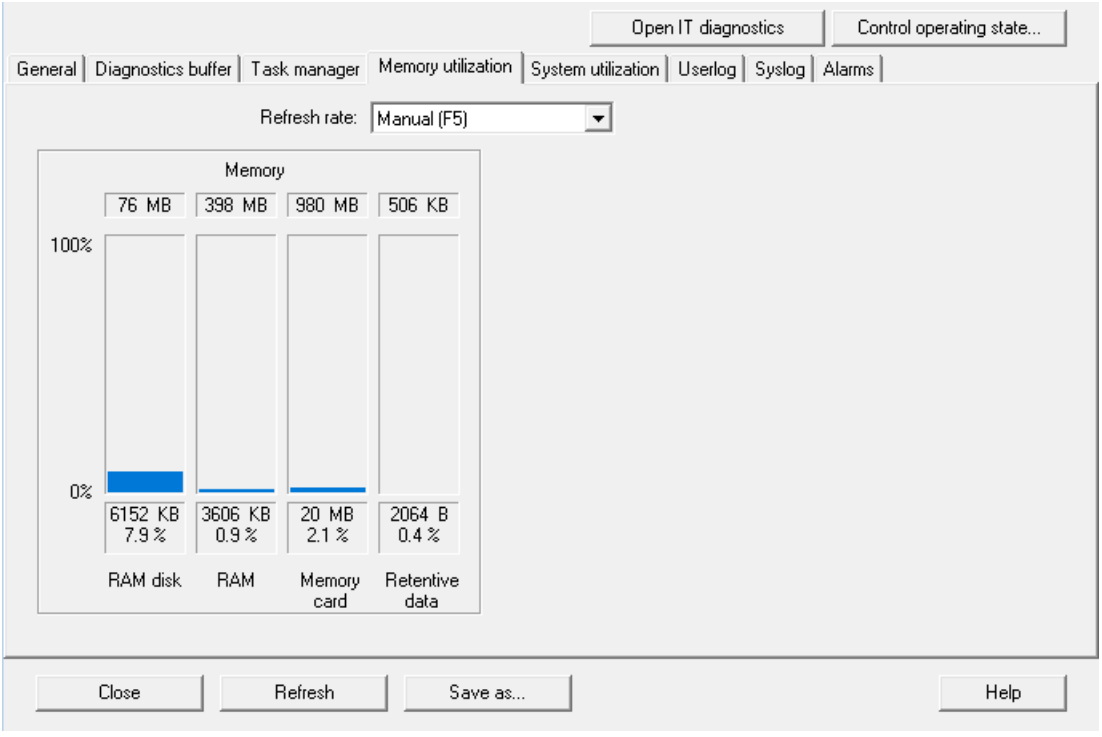


Figure 3-491 Example of the memory utilization display in the device diagnostics

Additional references

More detailed information on this topic can be found in:

- SIMOTION Runtime Basic Functions Function Manual, Section "Overview of the memory in the target device"
- SIMOTION SCOUT TIA Online Help

Checking the system utilization

To display the system utilization, proceed as follows:

- Select the "System utilization" tab in the device diagnostics.

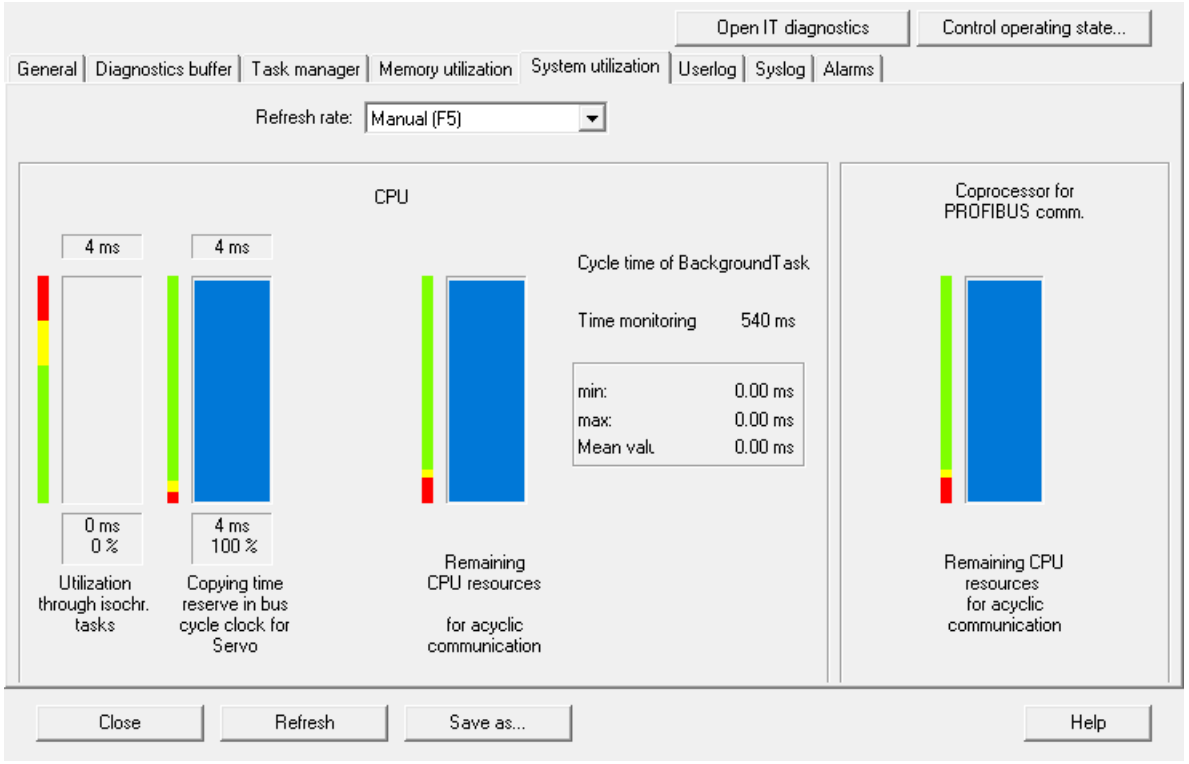


Figure 3-492 Example of the system utilization display in the device diagnostics

Additional references

More detailed information on this topic can be found in:

- SIMOTION Runtime Basic Functions Function Manual, Section "Overview of the memory in the target device"
- SIMOTION SCOUT TIA Online Help

User log file

With the Userlog file, you can store your own texts in the RT system. This is necessary, for example, when changes, which are to be documented, are made in the SIMOTION system on a plant which has already been commissioned.
Changes can be written in the SIMOTION SCOUT TIA. These are loaded to the ROM of the target device. When required, the text strings can be read out again. The text editor for the Userlog file is integrated as a tab in the device diagnostics Snap-in. This function is only available in online mode.

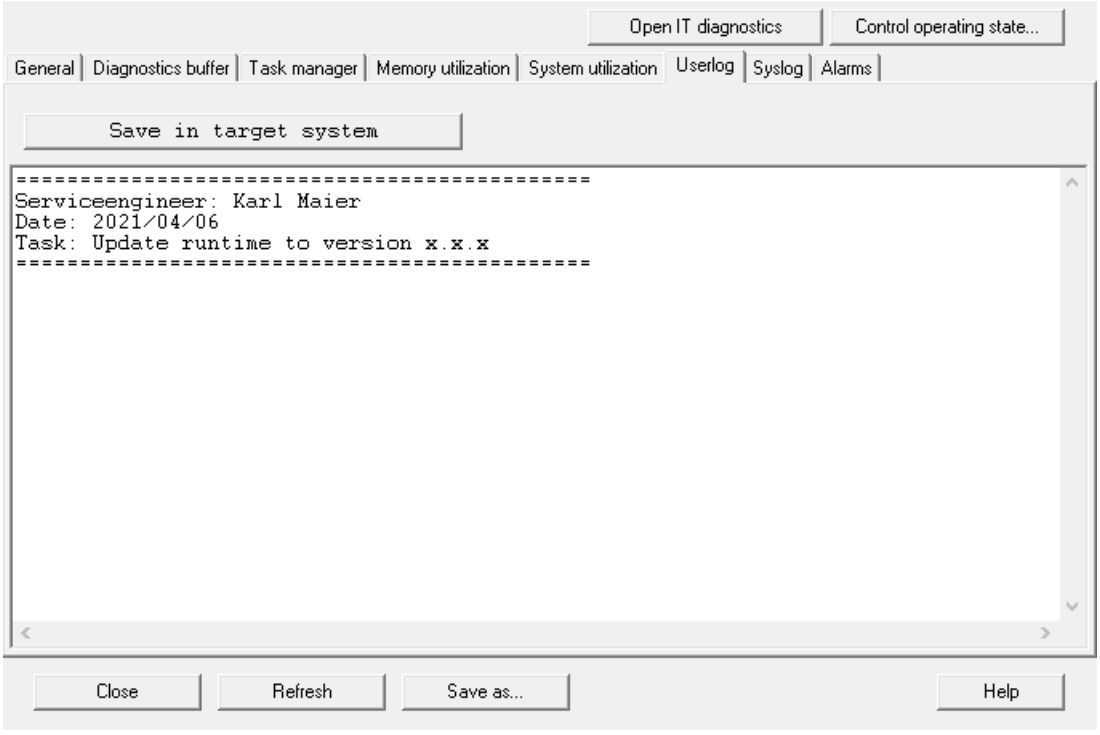


Figure 3-493 Example of the Userlog file display in the device diagnostics

How to work in the Userlog file

- In the Device diagnostics, select the "Userlog" tab.
  The editor is in edit mode, i.e. you can write or delete immediately.
  The system does not add any additional system content, such as date/time.
  Enter the information.
- To save, click "Save as..." The Userlog file is saved as a .txt file.
  All text entries can be changed or deleted at any time.
  Access protection is not available.
- The Userlog file can also be read without the project.
  The online mode is necessary for this.
- The Userlog file remains after "Delete user data".

Syslog file

In addition to the user-defined Userlog file, the SIMOTION device also has a Syslog file. The ROM actions entered therein facilitate a subsequent diagnosis. This function is only available in online mode. The information of the Syslog file can also be read without a project.

The Syslog file logs the following actions:

- RAM2ROM
- Overall reset
- Formatting the card from SIMOTION SCOUT TIA

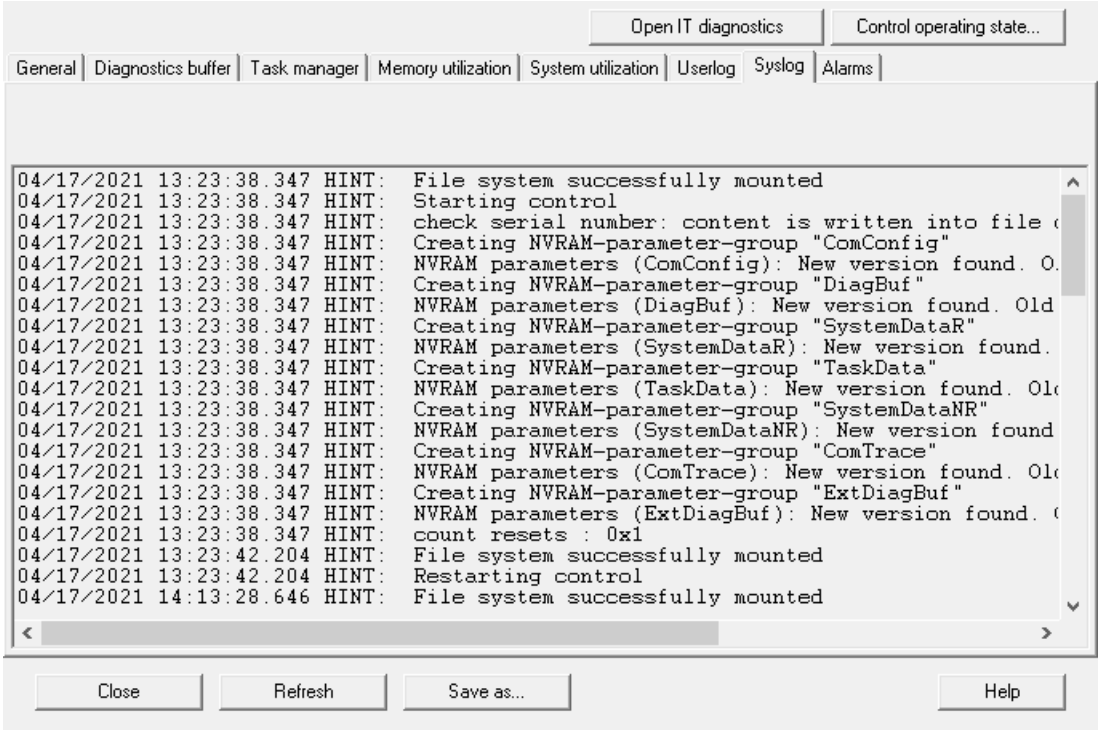


Figure 3-494 Example of the Syslog file display in the device diagnostics

Alarms

In the device diagnostics "Alarms" tab, pending alarms and configured messages are displayed in the same way as in the "Alarms" tab in the detail view.

Note

Alarm_S messages

When configuring Alarm_S messages, observe the specifications for the TIA Portal data type formattings.

Detailed information is contained in the information system of the TIA Portal at "Configuring messages".

Detailed information can be found in the SIMOTION SCOUT TIA Online Help, in the Alarms output window.

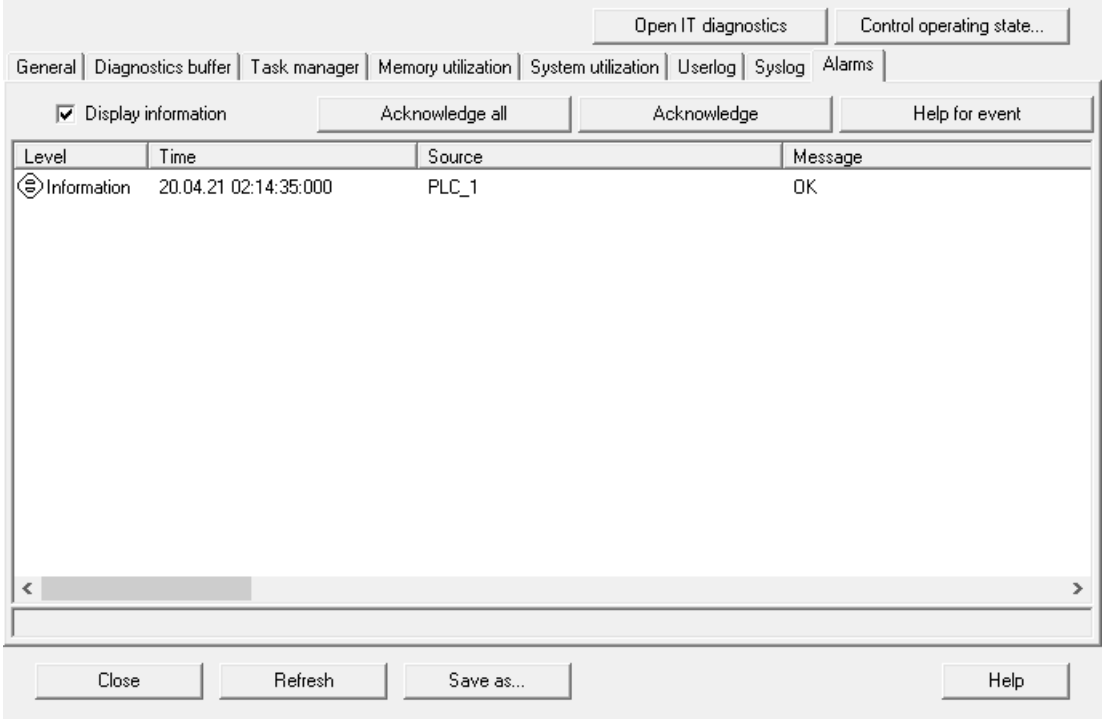


Figure 3-495 Example of the displayed alarms in the device diagnostics

3.5.9.4 Diagnostic functions in the address list

The Address list provides more features for I/O diagnostics. In online mode, the information is displayed in the "Availability" column. Detailed information is available via a tool tip if you move the cursor over the relevant cell.

To open the address list, proceed as follows:

1. Browse to the folder for your device in the project navigator.
2. Double-click the "ADDRESS LIST" item.
   The address list opens in the detailed area.

The following diagnostic information is recorded:

- I/O stations that have failed completely
- Modules have been removed, e.g. for ET200SP
- Deactivated I/O stations
- I/O variables working with substitute values
- I/O stations whose set topology differs from the actual topology
- I/O stations configured to be isochronous are not isochronous
  - Distributed synchronous operation
  - Drive units
  - Isochronous I/O

- Partner device, e.g. I-Device, I-Slave, is in STOP state
- For PROFINET devices: Provider state / consumer state displays errors
  - Controller
  - I/O device
  - Module
  - Submodule

Additional references

Further information can be found in the description of the "_quality()" system function in Section *Detailed status of I/O variables (as of kernel V4.2)* in the SIMOTION ST Structured Text Programming and Operating Manual.

For further information on diagnostics with the address list, please refer to the Online Help.

3.5.9.5 Trace and measuring functions

Trace, measuring function, and automatic controller setting

Overview

SIMOTION SCOUT TIA provides support for the commissioning and optimization of technology objects and user programs. The following functions are available:

- Device trace / function generator
- System trace
- Measuring function
- Automatic controller setting

Calling trace and measuring functions

To call the trace and measuring functions in SIMOTION SCOUT TIA, proceed as follows:

1. Select the SIMOTION device in the project navigator.
2. Select "Target system" in the menu and select the desired function.
   Or click the desired function in the toolbar.

① ② ③ ④

① Device trace / function generator
② System trace
③ Measuring function
④ Automatic controller setting

Device trace / function generator

You can use the device trace to record and evaluate parameters as well as system and program variables.

The device trace for system variables is mainly used to analyze time-synchronous sequences in the real-time system.

An automatic multiple trace is available as of SINAMICS V4.6. It can be used to trigger a recording automatically according to the trigger conditions. The measurement is stored as a YDB file (SIMOTION) or ACX file (SINAMICS) in a ring buffer on the memory card. At least five measurements can be stored in the ring buffer.
You can parameterize the multiple trace on the Trace tab under "Save to device (memory card)".

Program variables can be traced in order to find logical errors in the execution system or in user programs. For this purpose, rather than a time-triggered measuring task, an event-triggered measuring task in the RT system can be used. The event that causes the measurement recording is the execution of a specific code position in the user program by the RT processor. To start recording, a trigger event based on the variable can be selected on the Device trace tab (on a positive edge, a tolerance range or a bit map, for example).

The function generator can be used for test purposes to dynamically generate setpoints with defined shapes (e.g. rectangle, sine) for various system variables. With the aid of the trace, for example, the system response can be recorded in order to optimize the controllers. Two function generators are provided which you can start separately or simultaneously. You can also output the signals of the two function generators to a system variable at the same time. The signals are then superimposed.

System trace

You can use the system trace to record and evaluate parameters, system variables, and program variables from multiple CPUs at the same time. It is essential that the CPUs communicate via PROFINET. An isochronous connection must exist between the CPUs, and the PROFINET Sync Master must be a SIMOTION device.

Function generator, mathematical processing and bit tracks are not available for recording with the system trace.

Measuring function

The measuring function is used for controller optimization.

The SIMOTION measuring functions are used to commission the axis controller without requiring a user program.

With the SINAMICS measuring function, you can directly inhibit the influence of higher-level control loops by means of simple parameterization, and analyze the dynamic response of individual drives. The free measuring function measures and averages several measurement series with parameterizable noise sources without master control. This measuring function is suitable for controller settings and for avoiding whirling resonance with magnetic suspension bearing applications with rotor systems.

Automatic controller setting

The automatic controller setting can be used to configure the speed controller in the drive and the DSC position controller in the controller (SCOUT TIA only) for SINAMICS drive units.

Additional references

Detailed information can be found in the SIMOTION SCOUT TIA Online Help under Diagnostics.

Task Trace

Application area

The SIMOTION Task Trace supports you when troubleshooting in the SIMOTION multitasking environment. The SIMOTION Task Trace records the sequence of individual tasks, identifies user events that you can generate via a program command, and displays these graphically.

Structure of the Task Trace

The SIMOTION Task Trace includes two main components:

- The SIMOTION Task Tracer, which writes the task change and events to a buffer on the target device, and
- The SIMOTION Task Profiler, an application for displaying the recorded data.

Additional references

Detailed information on this topic can be found in:

- Function Manual: Task Trace
- SIMOTION IT Diagnostics and Configuration Diagnostics Manual
- SIMOTION SCOUT TIA Online Help

Trace technology object

With the Trace technology object, you can monitor commands at a technology object and track access to the system variables and configuration data

Overview

In SIMOTION, technology objects can be influenced by configuration data, system variables and commands during runtime. In order to represent the influences on a technology object during runtime through the user program, a TO trace / object trace records the last actions at the technology object in chronological order.

Note

Simultaneous use of the **Trigger through program call** trigger condition for the TO trace and the device trace is not permitted.

If the TO trace and the device trace are active simultaneously, different trigger conditions must be used.

The recording can be read out from the technology object on request and displayed in the ES.

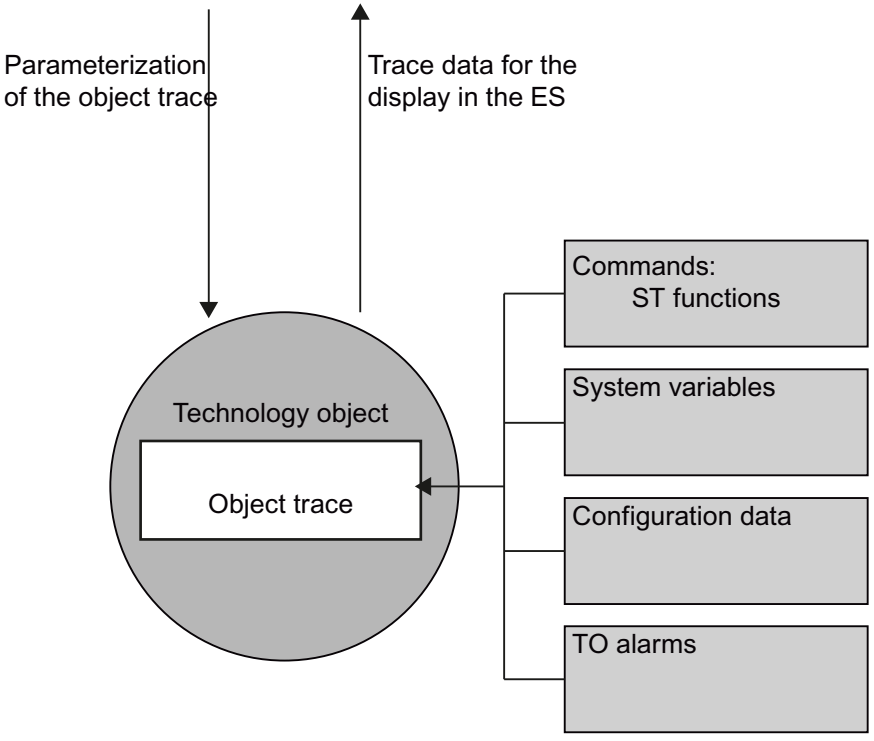


Figure 3-496 Principle of the object trace

The TO trace is used for the diagnostics of technology objects and displays the information and the events that influence a technology object during runtime in tabular form. The TO trace can be configured separately for each technology object.

It records commands and access to configuration data and system variables with their command and error statuses when the TO trace is read out. As of SIMOTION V4.4, alarms of the individual technology objects can also be recorded and an automatic trace is available. It can be used to trigger a recording automatically according to a trigger condition immediately after a restart. You parameterize the automatic trace in the "Settings > Save in the device (memory card)" tab.

The TO trace is displayed in a table/list with a time stamp.

The TO trace is available for the following technology objects:

- Drive axis
- Position axis
- Following axis
- Path interpolation
- Following object
- External encoder
- Measuring input
- Output cam
- Cam track
- Addition object
- Formula object
- Sensor
- Temperature channel
- Fixed gear
- Controller object

Additional references

Additional information on this topic can be found in the SIMOTION Runtime Basic Functions Function Manual.

3.5.9.6 Interconnection overview

The interconnection overview allows you to display all motion input and output interconnections of technology objects (TO) within the project. This overview is displayed in the SIMOTION SCOUT TIA working area in the form of an interconnection tree.

The tree display enables the synchronous operation interconnections to be displayed in cascades. In the interconnection table below, you can see all the TOs interconnected on the input and output sides for the technology object selected in the interconnection tree.

Call the interconnection overview with "Edit > Interconnection overview" in the menu.

Note

Device names with special characters

As of V4.3, it is possible to assign device names that do not comply with the ST naming convention. Names with special characters such as "." are displayed in the interconnection overview in " ".

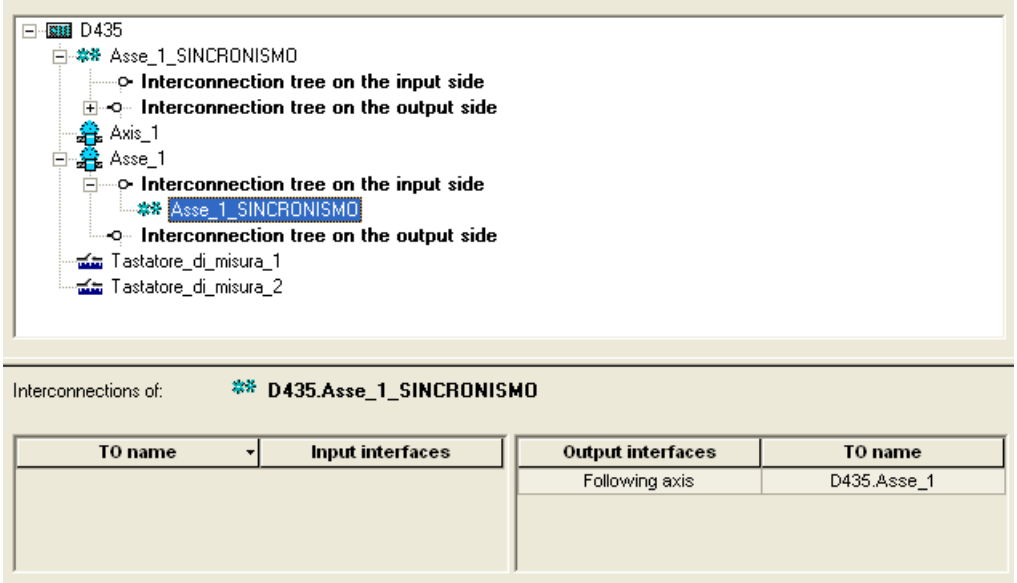


Figure 3-497 Example of an interconnection overview

Additional references

Further information on this topic can be found in:

- SIMOTION Runtime Basic Functions Function Manual
- SIMOTION SCOUT TIA Online Help

3.5.9.7 Service Overview

In online mode, the service overview shows a tabular complete overview of all configured axes in the project. The current state (including values from system variables) is displayed along with fault conditions.

The service overview is called with "Target system > Service overview" in the menu.

Error states are indicated by a red light and an active state by a green light.

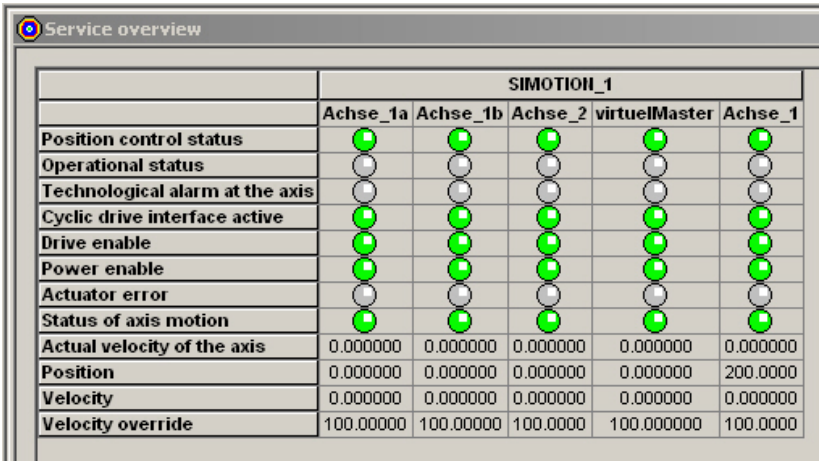


Figure 3-498 Service overview

3.5.9.8 Program testing and debugging

Comprehensive program testing and debugging functions are available in SIMOTION SCOUT TIA.

Overview

Various SIMOTION device operating modes are available for program testing.

To choose the operating mode, select the SIMOTION device and the "Operating mode..." command in the context menu.

Process mode

Program execution on the SIMOTION device is optimized for maximum system performance. The following diagnostic functions are available:

- Monitor variables in the symbol browser or a watch table
- Program status (restricted)
- Trace tool (restricted)

Test mode

The diagnostic functions of the process mode are available to the full extent.

The following diagnostic functions are available:

- Monitor variables in the symbol browser or a watch table
- Program status
- Trace tool

Debug mode

In addition to the functions of the test mode, you can use the following functions:

- Breakpoints
- Controlling MotionTasks

You can only select the debug mode for one SIMOTION device.

Note

Parameterizing the sign-of-life monitoring

Observe the safety information before using debug mode.

You will find detailed information on this topic in the SIMOTION SCOUT TIA online help under "Modes of the SIMOTION devices > Important information about the life-sign monitoring".

Additional references

Please refer to the following documents on this subject:

- SIMOTION ST Structured Text Programming and Operating Manual
- SIMOTION MCC Motion Control Chart Programming and Operating Manual
- SIMOTION LAD/FBD Programming and Operating Manual
- SIMOTION SCOUT TIA Online Help

3.5.9.9 Project comparison

You can use the "Compare..." function to compare objects within the same project or objects from other projects (online or offline).

Objects are devices, programs, technology objects (TOs) or drive objects (DOs) and libraries.

Procedure

To compare objects, proceed as follows:

1. Start the project comparison by clicking the "Object comparison" button .
   The "Select Comparison Partners" dialog opens.
2. Select the projects to be compared.

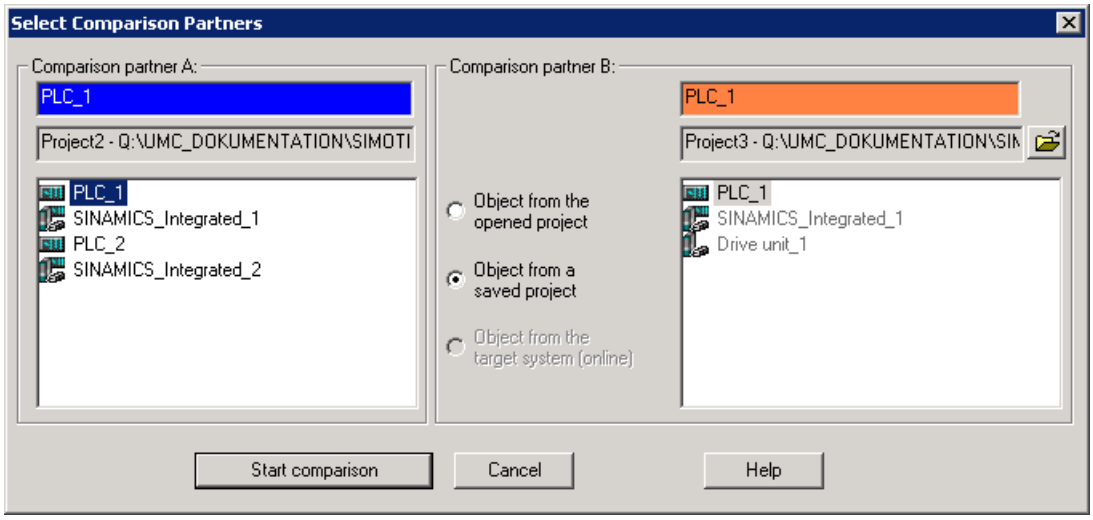


Figure 3-499 "Select Comparison Partners" dialog

3. Click the "Start comparison" button.

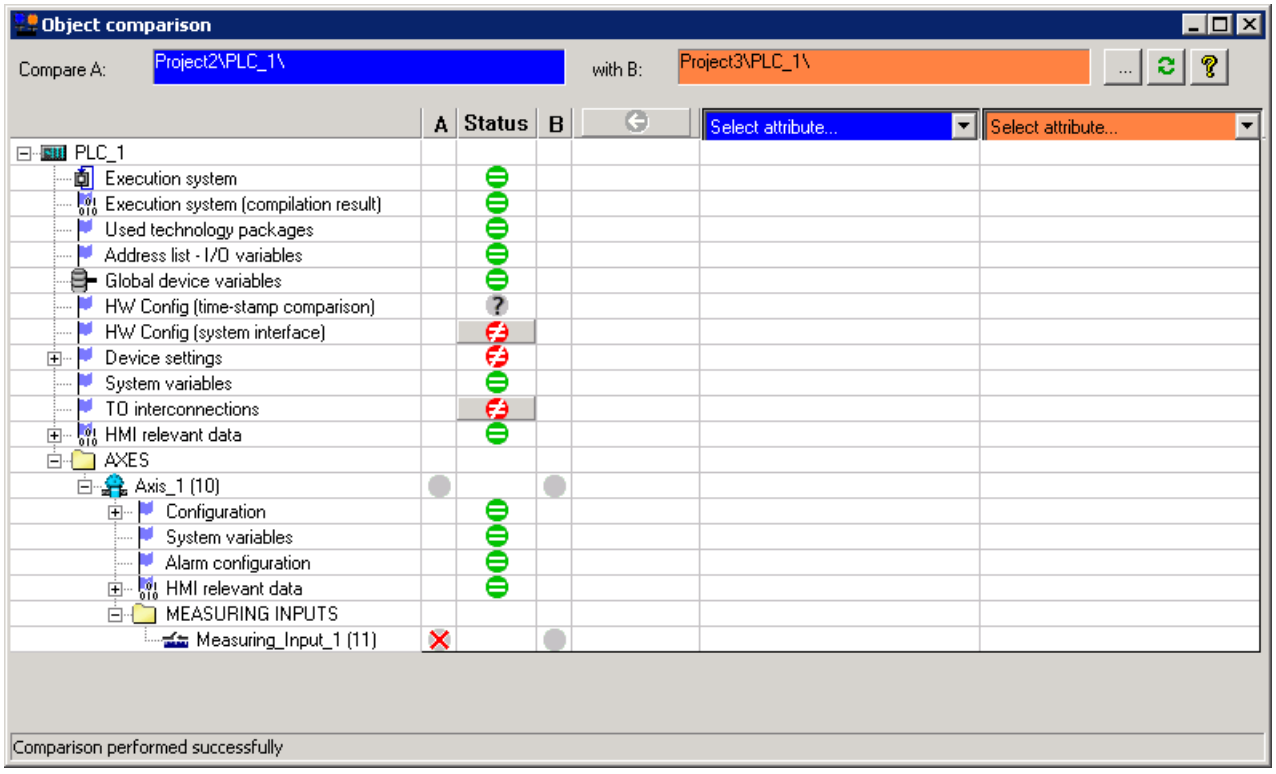


Figure 3-500 Synchronizing project data via the project comparison function

4. Move the mouse over the symbols highlighted in color to display information about the status.

Note

A detailed comparison is only possible if supplementary data (e.g. program sources) has also been downloaded to the target device ("Options > Settings... > Download").

Additional references

Detailed information on the topic can be found in the SIMOTION SCOUT TIA Online Help and in the SIMOTION Project Comparison Function Manual.

3.5.9.10 Version Control Interface

With Version Control Interface (VCI), SIMOTION SCOUT TIA provides the option of tracking and comparing changes to the source files within a version control.

Version Control Interface is characterized by the following features:

- Versioning automatically manages changes for all files that have been added to the version control system (VCS).
- Versioning allows complete traceability of all of the versions of every file, the author of these versions, and complete undoing of all changes from the start of the version control.

Advantages

Via the Version Control Interface of SIMOTION SCOUT TIA, you have the option of connecting any version control system (VCS) in SIMOTION SCOUT TIA.

You exchange data with the VCS via a work directory that you have defined as workspace.

Working with VCI offers the following advantages:

- You can export program sources of the project to a work directory as XML and back them up. You can transfer this data to the VCS as new version.
- You can restore a previous version in the work directory. This can be necessary, for example, when different versions were saved in the VCS via the work directory.
- You can work together in parallel in a team in a traceable manner and merge changes made.

Default settings

Before working with Version Control Interface, you can make the following settings:

- Define a default work directory.
  You can only define the work directory when a project is open. The work directory is assigned to the project.
- Connect the VCS using a configuration script.
  With a configuration script, it is possible to use the script commands in SIMOTION SCOUT TIA via the context menu or the operator controls.

  Note

  Scripts changed without authorization are not executed

  Make sure that unknown, unmanaged and unverified scripts are not called and/or executed with the login information of the user of the Siemens software. SIMOTION SCOUT TIA creates a checksum for every known script and saves it in encrypted form. The validity of the checksum is checked before each execution of a known script is initiated from SIMOTION SCOUT TIA. If the current and saved checksums of the script differ, SIMOTION SCOUT TIA does not execute the relevant script and reports a fault.
  If such an error occurs, make sure that the scripts have not been changed without authorization.

To make the settings for the work directory and/or a configuration script, proceed as follows:

1. Select the "Options > Settings..." menu.
   The "Settings" dialog opens.
2. Switch to the "Version Control" tab and make your settings there.

Starting Version Control Interface in SIMOTION SCOUT TIA

Note

Version Control Interface in offline mode

You can only start Version Control Interface in offline mode.

To start Version Control Interface in SIMOTION SCOUT TIA, proceed as follows:

1. In the toolbar in SIMOTION SCOUT TIA, click on the button  .
   The VCI window opens in the work area.
2. To start the comparison for the current project with the project data from the selected workspace, click the button  .
   If differences are detected between two projects, the status is displayed with an active button. Click the corresponding button to go to comparison mode and view the details.

Additional references

Detailed information on the topic of Version Control Interface in the SIMOTION SCOUT TIA can be found in the SIMOTION SCOUT TIA online help, section "Version Control Interface".

3.5.9.11 Creating a Project Overview by Script

In the Utilities_Applications\src\Scripts\ReportingScripts folder on the product DVD VOL3, you will find sample scripts that allow you to create reports on the following objects of the current SIMOTION project:

- TOs
- Programs, function blocks, functions
- Sources
- Tasks and assigned programs

The script lists the relevant specific objects of the SIMOTION project in an HTML document, arranged in a tabular format. Once generated, the HTML document opens automatically in the Internet Explorer. The information gathered can also be saved from there in HTML or CSV format or as an Excel file.

3.5.9.12 Services and diagnostics without an engineering system

Activating the Web server

SIMOTION devices have an integrated Web server. The Web server supports the display of diagnostics and system data in standard Internet browsers and the carrying out of project/ firmware updates, even in the absence of an engineering system.

As of version V4.5, SIMOTION SCOUT TIA supports the OPC Unified Architecture (UA) communication standard in addition to OPC XML-DA for access to process data.

Note

Protection against unauthorized HTTP requests

To protect the SIMOTION IT Web pages and the Web server, proven techniques have been implemented that prevent unauthorized access to the data.

As of V5.2, Cross Site Request Forgery (CSRF) protects all applications or Web services for which it could be possible to generally impair the status of the Web server and/or of the controller.

You will find detailed information on how CSRF works in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual, section "CSRF protection".

Procedure

To enable the Web server in the TIA Portal, proceed as follows:

1. Select the SIMOTION device in the network view or in the device view.
2. In the Inspector window, select the "Properties" tab and then click the "General" tab.
3. Select "Webserver/OPC/Netzwerkprotokolle"??.
   The Web server is disabled in the basic setting.
4. To display the CPU Web pages, you must activate the corresponding checkbox.

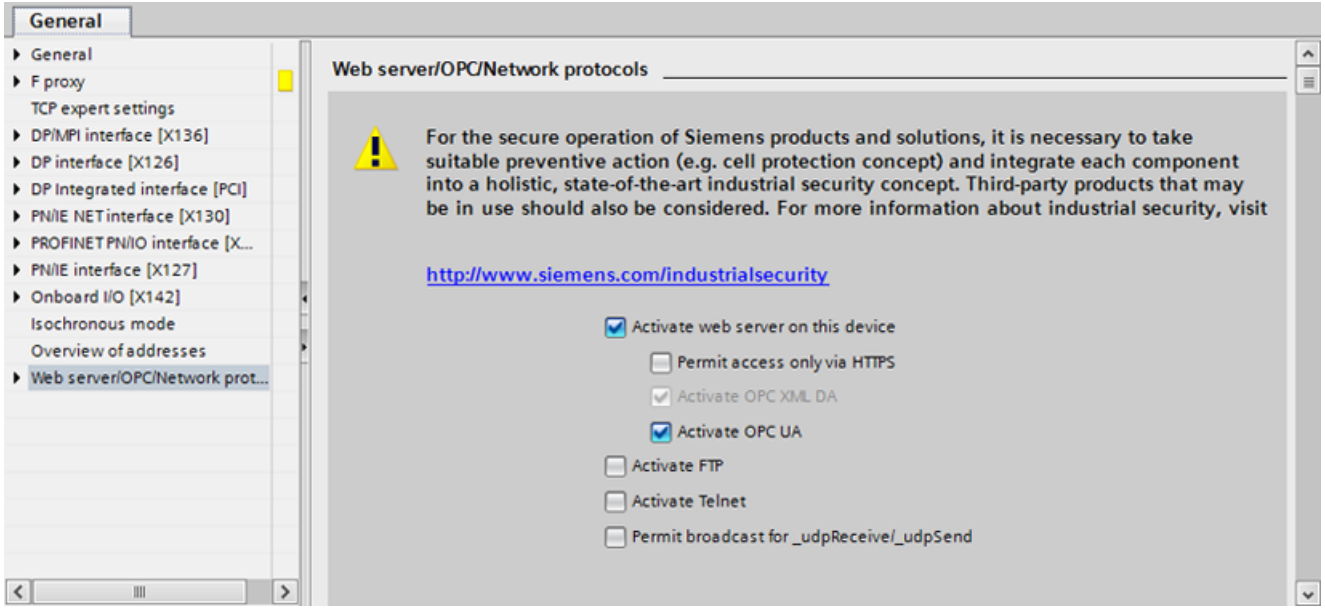


Figure 3-501 Activating the Web server

The Web server is addressed via HTTP/S. FTP and Telnet are only connected to the user administration.

Security concept of HTTP/S, FTP, and Telnet access on the Web server

As of version V4.4, access to the SIMOTION IT Web server is protected by a multi-level security concept.

The security state of the Web server is indicated by the security level on the Website. There are three possible security levels: Low, normal, high.

You will find detailed information on the security concept of the connection via FTP/Telnet in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual, section "Security concept".

Calling HW Config from SIMOTION SCOUT TIA

With SIMOTION SCOUT TIA, you can switch directly to the appropriate tab of the Inspector window in the TIA Portal.

To call HW Config from SIMOTION SCOUT TIA, proceed as follows:

1. Select the SIMOTION device in the project navigator.
2. Select the "Properties..." command in the context menu.
   The "Properties" dialog opens.

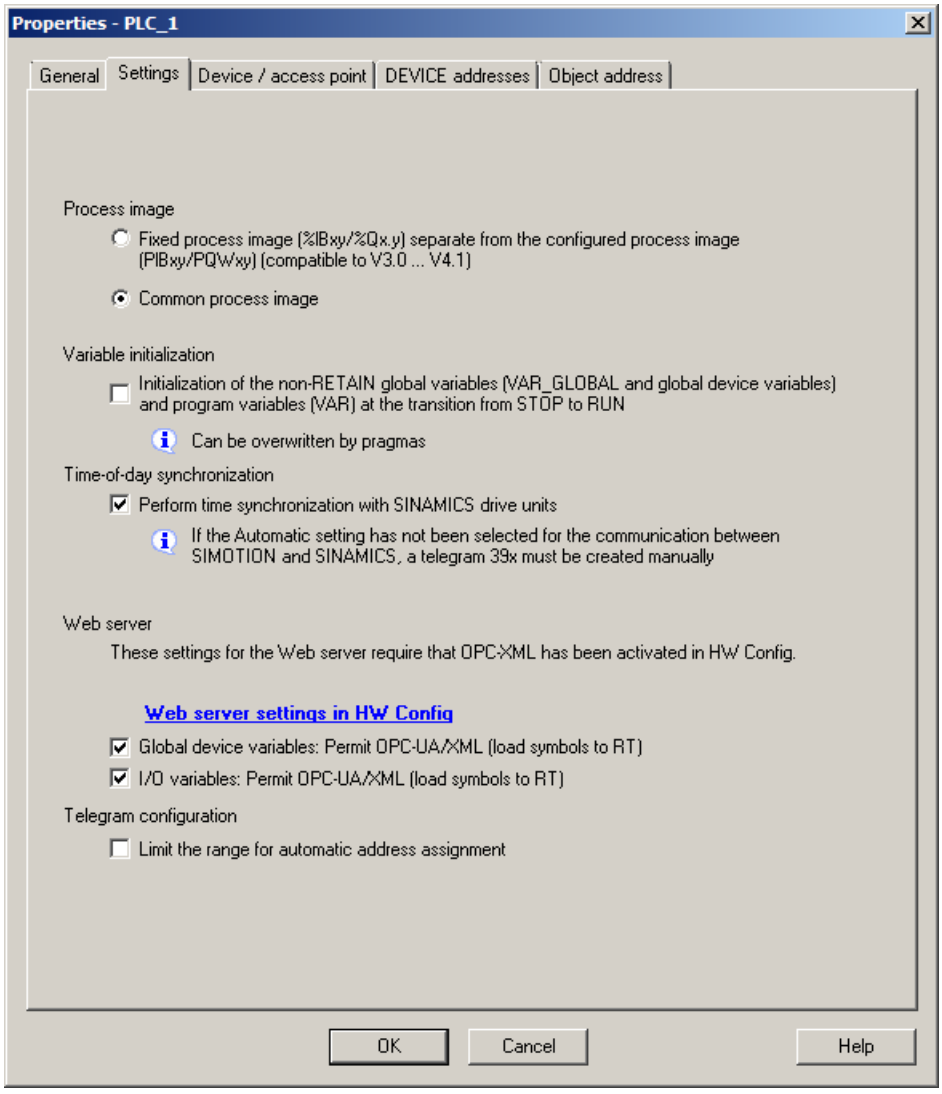


Figure 3-502 Calling HW Config from SIMOTION SCOUT TIA

3. Switch to the "Settings" tab and click the link "Web server settings in HW Config".
   Now you can change the Web server settings in the TIA Portal.

Additional references

Detailed information on this topic can be found in:

- SIMOTION IT Programming and Web Services Programming Manual
- SIMOTION IT Diagnostics and Configuration Diagnostics Manual
- SIMOTION SCOUT TIA Online Help

3.5.10 SIMOTION Runtime Simulation

3.5.10.1 Overview

Note

SIMOTION Runtime Simulation only for SIMOTION devices D4x5-2 as of V5.1

SIMOTION Runtime Simulation (SIMOSIM) is supported for SIMOTION devices D4x5-2 as of V5.1.

With SIMOTION Runtime Simulation, you can test and evaluate the general workflow of a SIMOTION runtime without connected hardware.

The simulation is so ideal for training purposes or, for example, to test and debug programs before they go online on a real module.

The projects are loaded into SIMOTION SCOUT TIA and can be switched retentive with RamToRom. SINAMICS Integrated and external I/Os are not simulated.

As of SIMOTION SCOUT TIA V5.3 you can perform the following simulation tasks with the new PROFIdrive state machine implemented in SIMOSIM:

- Switch the drive selectively to the S1 - S4 states with _enableAxis(BY_STW_BIT)
- Monitor drive enables with STW1 via the system variable on the TO axis
- Monitor status information of the drive with ZSW1 via the system variable on the TO axis

Note

Simulation of SIMOTION devices

Note that not more than one device can be simulated concurrently within a project.

Restrictions

The following functions are not supported:

- CLib
- OA packages
- TPDCBADM, TPTControl, TPdcblib technology packages

Note

Technology package product versions

Note that only those technology packages integrated in the simulation (TPCAM, TPPATH, TPCAM_EXT) are supported.

3.5.10.2 Using SIMOSIM

Requirements

- You have installed SIMOTION SCOUT TIA.
  SIMOSIM is part of the installation up to SIMOTION SCOUT TIA V5.2.
  SIMOSIM will be supplied as a dedicated component in a separate setup as of SIMOTION SCOUT TIA V5.2 SP1 HF2.
- You have activated the "Siemens SIMOSIM Virtual Ethernet Adapter" in the network settings of your PC.
- You have created a project in TIA Portal and use SIMOTION D4x5-2 devices as of V5.1 in your project.
- SIMOTION devices and Siemens SIMOSIM Virtual Ethernet Adapter are contained in the same IP subnet.
- No active online connection to a real SIMOTION device exists.

Starting SIMOSIM in SIMOTION SCOUT TIA

Note

Starting SIMOSIM for side-by-side use of SIMOTION SCOUT and SIMOTION SCOUT TIA

You can start SIMOSIM just once, either in SIMOTION SCOUT or in SIMOTION SCOUT TIA.

To start SIMOSIM and simulation mode in SIMOTION SCOUT TIA, proceed as follows:

1. Switch to SIMOTION SCOUT TIA.
2. In the project navigator, select the device you want to simulate.

3. Click "Start SIMOSIM" in the menu bar.
   The "Start Simulation Mode and SIMOSIM" dialog box opens.

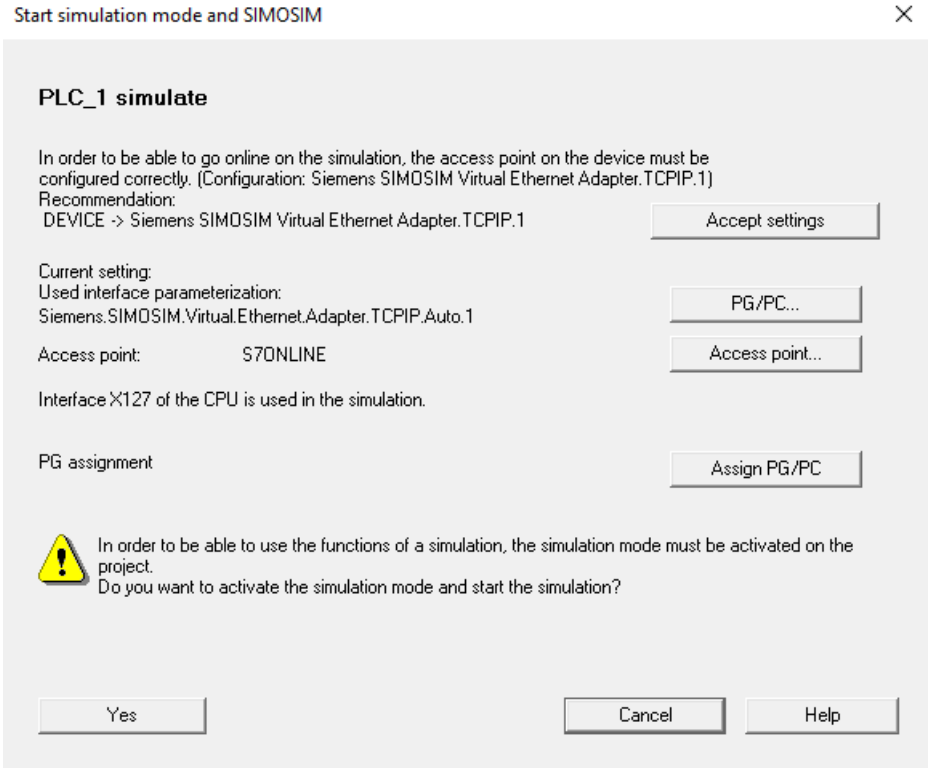


Figure 3-503 Starting simulation mode and SIMOSIM

4. Click "PG/PC...".
   The "Set PG/PC Interface" dialog box opens.

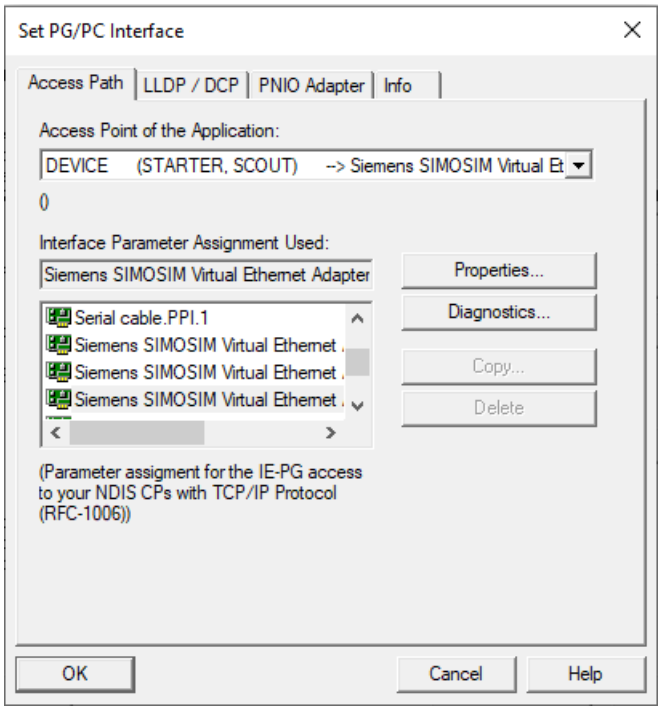


Figure 3-504 Setting the SIMOSIM PG/PC interface

5. Select "DEVICE ->Siemens SIMOSIM Virtual Ethernet Adapter.TCPIP.1" as access point of the application.
   Note the prompt that an online connection via router is no longer possible with this setting and confirm with "OK".
6. Confirm the selected access point with "OK".
   Note the prompt about the changed access path and confirm with OK.

Note

Accepting Siemens SIMOSIM Virtual Ethernet Adapter automatically as setting

To accept Siemens SIMOSIM Virtual Ethernet Adapter automatically as setting, click "Accept settings" in the "Start Simulation Mode and SIMOSIM" dialog box.

7. Click "Access point..." in the "Start simulation mode and SIMOSIM" dialog box.
   The "Properties > Device / access point" dialog box opens.

Figure 3-505 Access point used by SIMOSIM

8. Activate "DEVICE" as used access point.
9. Click "Set DEVICE addresses".
   The "DEVICE addresses" tab appears.

10. Activate X127 as virtual interface.
    You can use online access to the simulation only via interface X127.

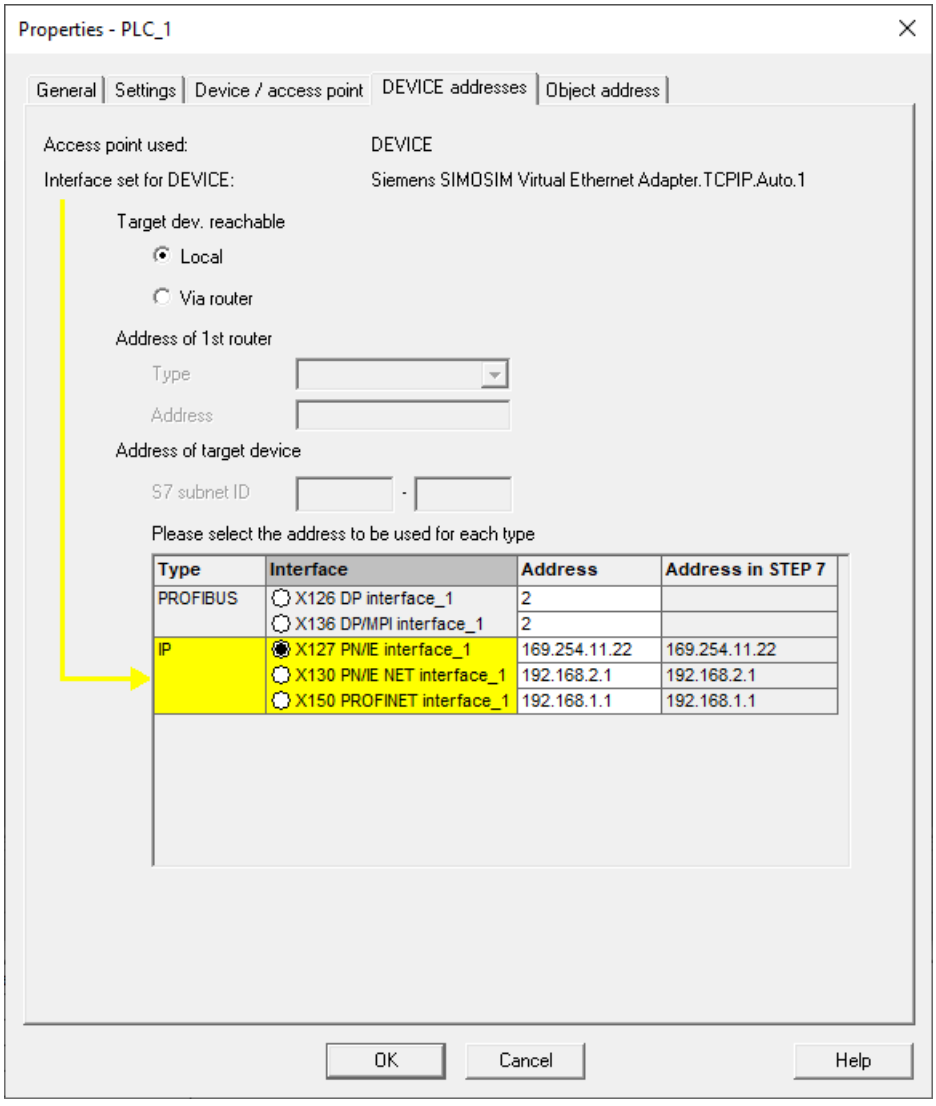


Figure 3-506 Setting the SIMOSIM DEVICE address

11. Click "OK" to confirm the selection.

12. Check your settings for the online interface.

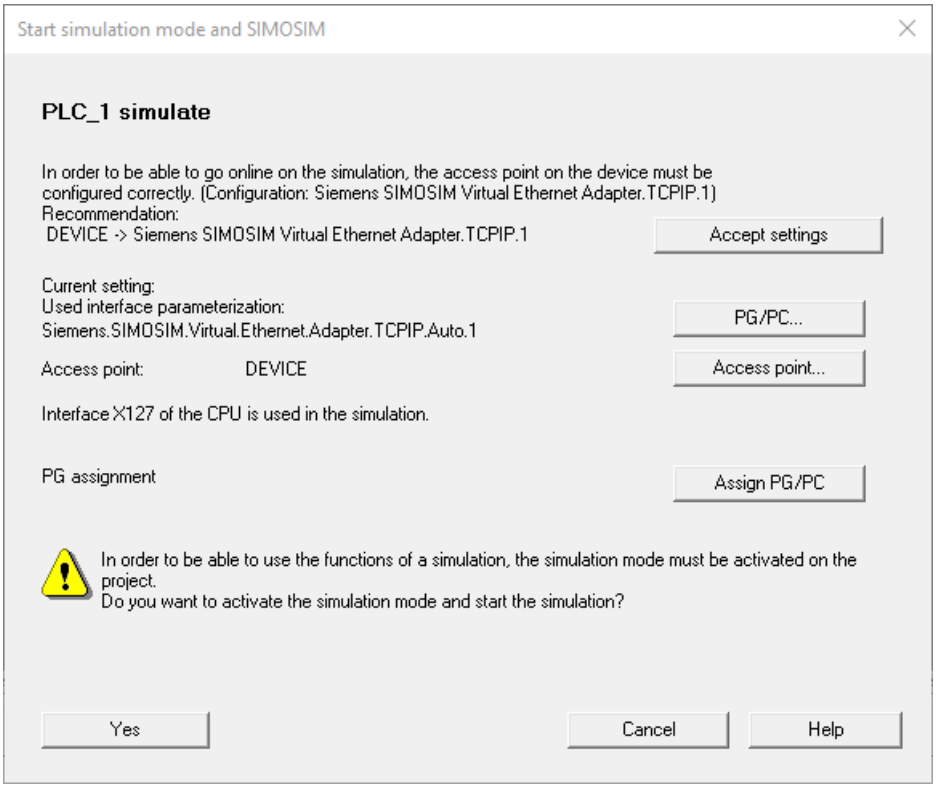


Figure 3-507 Starting simulation mode.

13. To start simulation mode, click "Yes".
Simulation mode is activated, SIMOTION simulation is started and a virtual controller is shown.

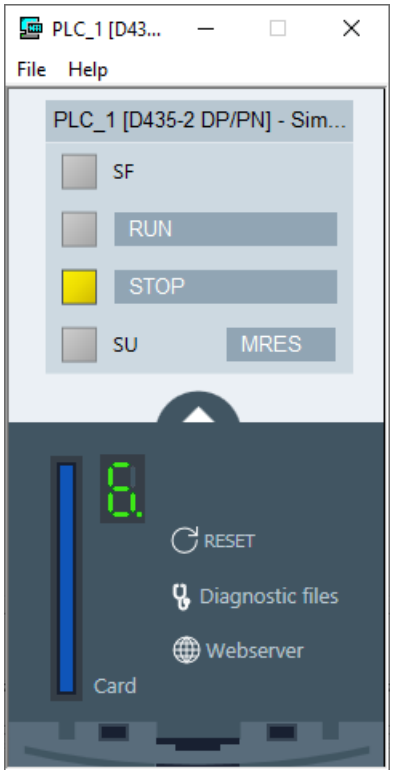


Figure 3-508 Virtual controller

The graphic user interface allows you to:

- Control the CPU operating state
- Open the Web server in the browser window
- Access the diagnostic files
- Reset the controller
- Access data on the memory card

Accessing simulation data

At every start of a new simulation of a device, a directory is created in which the memory image of the device is stored.

The data stored in the NVRAM, RAM disk and CF card can be found in this directory.

To access data in this directory, proceed as follows:

1. Start the SIMOTION simulation.
2. Click the slot of the CF card in the displayed virtual controller.
   The Windows Explorer opens and the directory of the CF card is displayed.
3. To access the memory image of the NVRAM and RAM disk, switch from the CF card to the root directory of the virtual controller in the Windows Explorer.

Note

Check and clear the data memory regularly.

To ensure that no memory bottlenecks occur and that all functions can be used without restrictions, regularly delete the simulation data records from the directory.

To delete data from the simulation directory, proceed as follows:

1. Select "Project -> Delete SIMOSIM data..." in the menu.
   The "Delete SIMOSIM Data" dialog box opens.
2. Select the simulation project whose data you want to delete from the directory.
3. Click "Delete" to start the deletion.

Starting SIMOSIM in simulation mode

To start the simulation in activated simulation mode, proceed as follows:

1. Click  "Start SIMOSIM" in the menu bar.
2. The "Start SIMOSIM" dialog box opens.

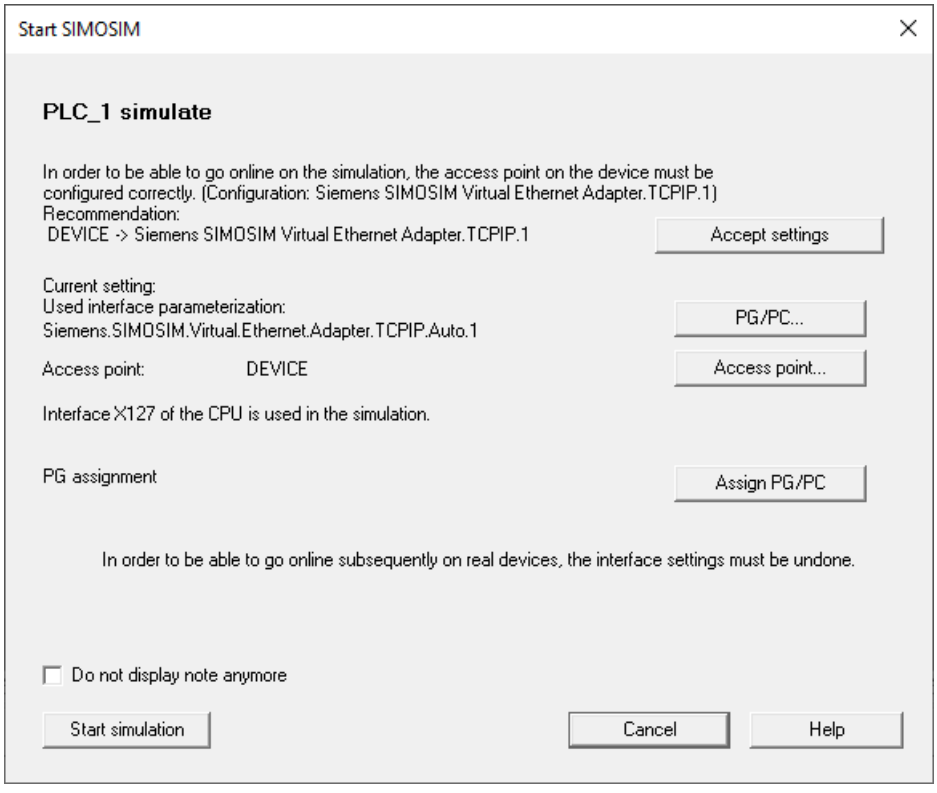


Figure 3-509 Restarting SIMOSIM

3. Click "Start simulation".

Result

Simulation mode and SIMOSIM have been started.

You can now go online with a target device and download your project data into the virtual target system.

You can then simulate programs, axes, etc. You so allow SIMOSIM to write directly to the inputs.

Note

Going online in simulation mode

Note that only one device can go online in simulation mode.

Note

Simulation mode active/inactive

When simulation mode is active, this is color-highlighted in the SIMOTION SCOUT TIA status bar.

To exit simulation mode, close the controller window by clicking the "X" in the upper area of the window. Deactivate the "Use simulation mode" option in the "Project" menu.
The flag in the status bar is reset. Simulation mode is exited.

Note

Emergency stop deactivated in debug mode

Note that in debug mode, no emergency stop is made with the space bar or by switching the application.

Note

SIMOSIM Web server access via HTTPS

Note that the certificates must be created at the first access via HTTPS. This may take some time. Acknowledge any messages that are displayed.

Setting the online access point in the TIA Portal

You now find yourself in SIMOSIM simulation mode.

To set the online access point in the TIA Portal, proceed as follows:

1. Switch to the TIA Portal.
2. Select the "Online > Online & Diagnostics" menu.
   The "Online Accesses" dialog box opens.

3. Select SIMOSIM and X127 as access point

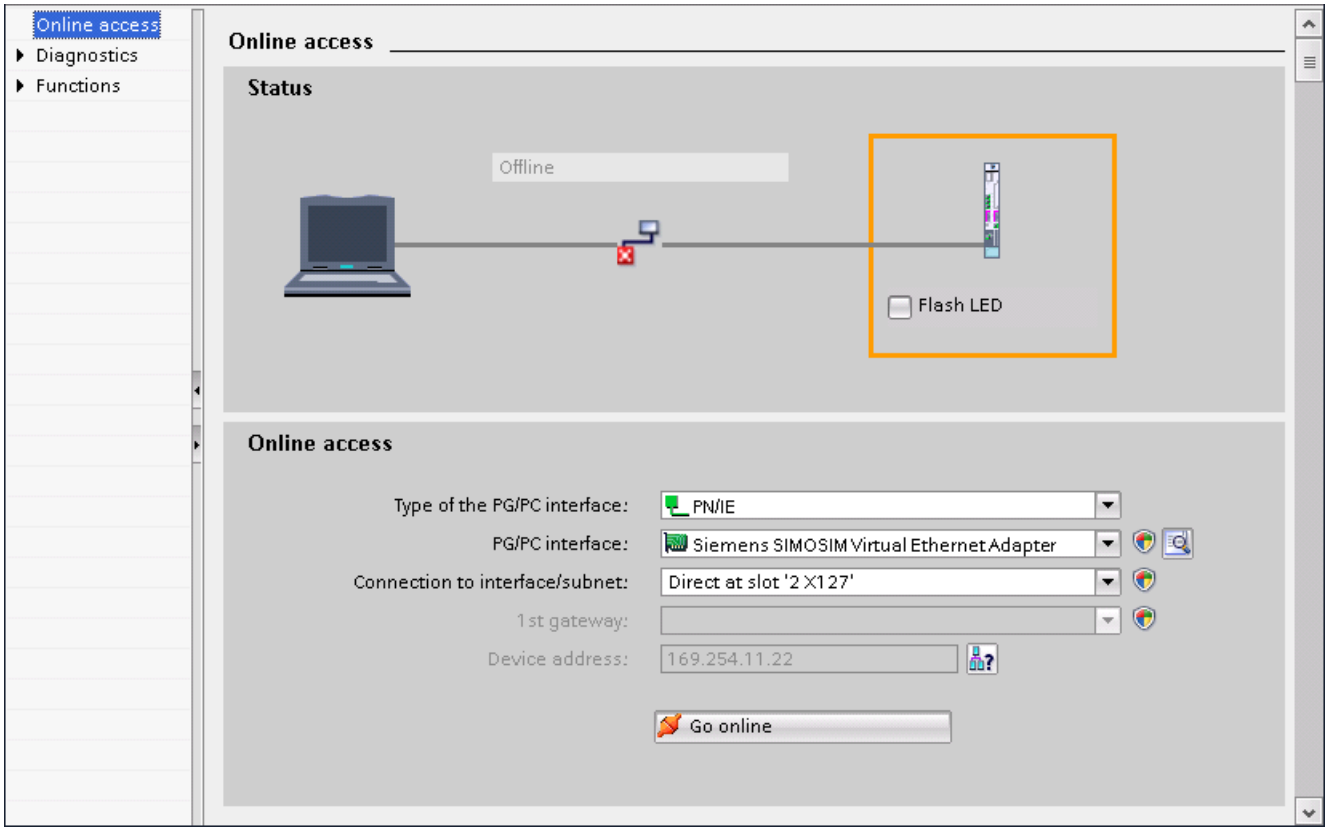


Figure 3-510 Setting the SIMOTION simulation access point

4. Click "Connect online"
   A device is displayed that can be accessed and simulated via the access point.

3.5.11 Multiuser engineering

3.5.11.1 Overview

With Multiuser Engineering, the TIA Portal offers you the possibility, in the context of the Team Engineering functionality, to work together with multiple users and to perform various configuration tasks in a project at the same time (Multiuser Engineering). You can then commission your configuration together using the TIA Portal functionality "Commission Engineering".

This significantly reduces the configuration and commissioning times. You can commission your plant and start to work productively much quicker.

The multiuser engineering in the TIA Portal functions as follows:

- The project management is on a local or an external server.
- Different users work in local sessions, based on the projects managed by the server.

- Users work independently from each other in local sessions.
- Changes from local sessions are transferred by checking into the server project.
- Checked-in changes from other users are displayed and can be applied.

The Multiuser Commissioning in the default synchronous mode in the TIA Portal functions as follows:

- As with commissioning, project management is located either on a local or external server.
- Different users work in local sessions, based on the projects managed by the server.
- Users can work independently from each other in local sessions also during commissioning.
- The changes from the local sessions are first always downloaded to the common CPU and only transferred to the server project after successful download.
- Downloaded changes from other users are displayed and can be conveniently applied.
- The server project is already synchronized on the project server. This means consistent versions of the server project are always downloaded to the CPU. Time-consuming synchronization of different project versions on the CPU therefore becomes obsolete.

The following description of the functionality of Multiuser Engineering in the TIA Portal serves only as a brief insight into team-oriented configuration within the TIA Portal.

Functionalities that go beyond this, e.g. commissioning together at the same time in a team (Multiuser Commissioning) as part of the TIA Portal functionality "Team Engineering", do not form part of this documentation.

Additional references

Detailed information on the concept of Team Engineering in the TIA Portal can be found in the information system of the TIA Portal at "Using Team Engineering".

3.5.11.2 Working with Multiuser Engineering

Requirements

- The TIA project server is installed.
  You can install the TIA project server together with the TIA Portal or standalone via a separate installation process.
  The local project server with limited functionality is installed automatically with the TIA Portal.

  Note

  License required

  To be able to use the project server with extended functionality, for example because you want to use it as external server for long-term cooperation, you need an additional license.

  While you are only using the local project server available with limited functionality, you do not need an additional license for the installed TIA Portal.

- You have created an executable project.

Creating a local session

To be able to work on a server project together with other users in the context of Multiuser Engineering, each user needs to create their own local session.

The respective user can insert their changes in one or more local sessions and then check them into the server project and publish them.

After check-in, the changes from the local session are available again to all users in the server project. They can continue to be used and modified again in subsequent local sessions.

The following steps are required to create a local session:

Configuring the local project server

To configure a local project server, proceed as follows:

1. Select the "Options > Settings > Project server" command in the TIA Portal menu.
2. Double-click on "Local project server" under "Project server" in the table.
   The "Edit server connection" dialog opens.
3. Enter the desired port.
4. Confirm your entries with "OK".

Starting the local project server

Proceed as follows to start the project server:

1. Select the "Project > Project server > Manage server projects..." command in the TIA Portal menu.
   The "Manage server projects" dialog opens.
2. Under "Select server", choose the local project server in the drop-down list. If server projects have already been created, you will see all server projects for the selected server connection listed in the table.
   The "Start local project server" dialog opens.
3. To start the local project server, confirm with "Yes".

Adding the server project

To add a project to the project server, proceed as follows:

1. Select the "Project > Project server > Manage server projects" command in the TIA Portal menu.
2. To add a server project to the server, click "<Add project to server>".
   The "Add project to project server "Local Project Server"" dialog opens.
3. Select the project that you want to make into the server project under "Source path:".
4. Click "Add".

Creating a local session

Follow these steps to create a local instance:

1. Select the "Project > Project server > Manage server projects" command in the TIA Portal menu.
2. Select "Manage server projects" in the dialog and right-click to select "Create new local session" in the context menu.
3. Select the type for the session, check the path for the session and change it if necessary.
   - Select "Multiuser Engineering" if you want to work with multiple users in parallel.
   - Select "Exclusive Engineering" if you want to work exclusively in the local session.
4. Click "Create".

Result

The session folder has been created in the Windows Explorer and the server project created with the file extension "als17".

3.5.11.3 Editing the server project

So that the changes made by different users in separate sessions can be combined and published in the context of Multiuser Engineering, the respective changes from the session or the server project view must be checked in and the server project updated.

Note

Editing objects in SIMOTION SCOUT TIA

The SIMOTION configuration cannot be opened in the session in V17.

To make changes in SIMOTION SCOUT TIA, switch to the server project view in which a temporary local copy of the server project is opened.
During this time, other users cannot open a server project view and check-in changes.

Procedure

Editing objects in the session

To edit a server project in the session, proceed as follows:

1. Select the "Open" command in the "Project" menu.
2. Click "Browse" in the "Open project" dialog.
3. Navigate to the created session and select the file with the extension "als17".
   The selected session opens and the associated toolbar is displayed.

Figure 3-511 Local session toolbar

4. Edit the desired objects in the hardware configuration.
5. Select each changed object. To do this, click the grayed-out flag.
   In the example shown, the HMI variable "HMI_Variable_1" has been marked for check-in with a blue flag.

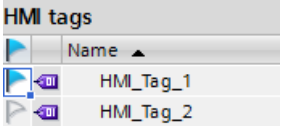


Figure 3-512 Selecting an object

6. To transfer the changes to the local server project, click "Check in" in the local session toolbar.
   All changes pending for check-in are shown in the work area.
7. Click "Start check-in".
   A message may be output prompting you to adjust the options for compiling. You make this setting in the TIA Portal under "Options > Settings > Multiuser".

Note

Changes to objects that cannot be selected

To make changes to objects that cannot be selected, you need to open the server project view.

If you have made changes to objects that cannot be selected in the local sessions, these changes are not applied on check-in.

Editing objects in the server project view

To edit a multiuser server project in the server project view, proceed as follows:

1. Select the "Open" command in the "Project" menu.
2. Click "Browse" in the "Open project" dialog.
3. Navigate to the created session and select the file with the extension "als17".
   The selected session opens and the associated toolbar is displayed.

Figure 3-513 Local session toolbar

4. Change to the server project view. To do so, click in the multiuser toolbar.
   The server project view opens under the session. The session is now in "read-only" mode.
5. Open the SIMOTION configuration in the server project view and make the desired changes.
6. Switch to the TIA Portal and click "Save changes".
7. Enter a meaningful comment for the change history.
8. To close the server project view when required, click .
   The server project view closes. You are now in your local session again.
9. To transfer the changes to the local server project, click "Check in" in the local session toolbar.
   All changes pending for check-in are shown in the work area.
10. Click "Start check-in".
    A message may be output prompting you to adjust the options for compiling.
    You make this setting in the TIA Portal under "Options > Settings > Multiuser".

Result

All changes to objects in the project have been saved and checked in.

1. To synchronize any newer changes from editing of the server project view in the local session, click on "Refresh local session" in the local session toolbar.

Note

Showing the change history

To show the change history to the local session, start the Administration Tool under Windows via "Start > All Programs > Siemens Automation > TIA project server Vxx - Administration".

Add the corresponding server. All server projects assigned to the server are shown in the left-hand area of the tool. You can also display the local sessions for this project via the arrow on the left next to the server project and view the history of the changes made.

3.5.11.4 Comparing project versions in SIMOTION SCOUT TIA

Thanks to the project comparison, device upload and download with supplementary data functions, several users can exchange project data that has been modified via the target device, online. This allows them to synchronize the status of their own SIMOTION project with that of another one, thereby updating it.

The following functions can be performed in parallel operation in SIMOTION SCOUT TIA:

- Reading and controlling of system variables and configuration data
- Execution of measuring functions and axis control panel
- Uploading of configuration data
- Uploading of programs or technology objects and other settings (e.g. execution system)
- Editing and downloading of programs (following prior alignment, if applicable)
- Downloading of individual program sources even when other sources are inconsistent

See also

Project comparison (Page 922)

Loading data to the target system (Page 814)

Loading data from the target device (Page 820)

Device upload (Page 947)

3.5.12 Updating with the Device Update Tool

3.5.12.1 Updating SIMOTION devices

SIMOTION offers a convenient solution for updating SIMOTION devices or SIMOTION projects for machine manufacturers and machine operators.

Updating does not simply refer to an update to a higher version of firmware; rather, in general terms it refers to switching to a defined configuration, e.g. a project update. It is also possible to return (restore) to a previous configuration. Update or restore procedures can easily be performed on SIMOTION devices locally or remotely. The data can be imported to a SIMOTION device via a portable, easy-to-use storage medium or a communication connection.

You can create update or restore data based on the opened project with the Device Update Tool. The data contains all the information required for updating or restoring the data on a SIMOTION device.

This includes:

- SIMOTION project
- Technology packages
- User data
- Firmware

Once the update data has been created, it can either be saved directly to an update medium or stored as an update archive.

Depending on the SIMOTION device, the storage or update medium can be:

- CF/MMC card
- USB memory stick or
- SIMOTION IT DIAG file
- File system (update archive)

Update data can also be imported from the update medium to the SIMOTION device, irrespective of time and place, using the Device Update Tool.

Note

Exiting SIMOTION SCOUT TIA terminates the Device update tool functionality

Note that when exiting SIMOTION SCOUT TIA or closing the project in SIMOTION SCOUT TIA, the functionality of the Device update tool is also terminated.

When required, close the open "Device Update Tool" dialog box with "Cancel".

Additional references

Detailed information can be found in the SIMOTION SCOUT TIA Online Help at "Updating SIMOTION devices" and in the "Updating SIMOTION devices" Operating Instructions.

3.5.13 Device upload

3.5.13.1 Upload functions

As of TIA Portal V14, the hardware configuration of a SIMOTION CPU can be uploaded in the TIA Portal.

Note

Note that the device that you want to upload in the TIA Portal must have been loaded with the TIA Portal (≥V14).

Note

Note that the upload of SIMOTION PC stations is not supported.

The following upload functions are supported:

- Uploading of the hardware configuration in the TIA Portal for a SIMOTION CPU.
- Uploading of the SIMOTION data in SIMOTION SCOUT TIA including the drive data, e.g. for a project comparison, a data adaptation of the configuration data, for multiuser engineering or for debugging.
- Small changes, e.g. in an ST source file.

Alternatively to an upload, you can store the archived project on the CF card.

Additional references

Detailed information on the upload functions in the TIA Portal can be found in the information system of the TIA Portal.

See also

Project comparison (Page 922)

Loading data from the target device (Page 820)

3.5.14 Appendix A

3.5.14.1 Scripting functionality

Automation of sequences with scripts

Overview

You can use the scripting functionality to automate the configuration of devices, such as drive objects (DOs), e.g. SINAMICS drives and SIMOTION technology objects (TOs), such as axes, using an easy-to-learn script language.

Standard scripts can be adapted to special situations occurring during runtime with interactive queries that modify the script processing depending on the query results. This facilitates and speeds up commissioning.

Other application possibilities include the documentation of the settings that have been made and the repetition of complex settings without error.

The implementation basis is VBScript by MICROSOFT, which has been expanded with special objects and methods for SIMOTION SCOUT TIA.

The application of the scripting functionality assumes that you are familiar with script programming.

For scripts and related documents, refer to the installation DVD 3 in the Utilities_Applications\src \Scripts directory.

The selection of documents and scripts you will find there should help you quickly get to grips with scripting in SIMOTION. As well as demonstration-only scripts for studying the code, there are also scripts and script libraries you can actually use yourself.

Restrictions

In the context of SIMOTION SCOUT TIA (TIA Portal), only scripts that influence/use pure SIMOTION SCOUT/SIMOTION SCOUT TIA data/functionalities can be used.

A scripting of the HWCN data/functionalities and project handling of the TIA Portal (framework) is not possible.

The following is explicitly not possible via scripting:

- Creation, deletion, and renaming of objects that have a representation in the TIA Portal (e.g. all SIMOTION devices).
- Creating, deleting, and renaming of projects.
- File system access to TIA Portal projects.
- Utilization of functionality provided by the TIA Portal (framework), e.g. archiving/retrieval.
- Any handling of TIA Portal data.

Additional references

For detailed information, refer to Section "Scripts for sequence automation" in the Online Help.

Scripting with TIA Portal Openness

Introduction

External scripting of the SIMOTION data is only possible in combination with scripting in the TIA Portal. The project and device handling is controlled from the TIA Portal.

To be able to access the valid object model and the Application Program Interface, you must install the *TIA Openness V14.0* option package.

As of STEP 7 or WinCC V13 SP1, the TIA Portal Openness is contained in the scope of delivery of STEP 7 or WinCC in the TIA Portal. You can find the components in the installation files at "Support".

As of TIA Portal V14 SP1, TIA Openness is included in the functional scope of the setup.

Requirement

In order to be able to access a C# project on the SIMOTION application object, the following references must be entered: Siemens.MC.Simotion.Scripting.dll

In order to be able to access the TIA Portal application object, the following references must be entered: Siemens.Engineering.dll and Siemens.Engineering.Hmi.dll

You can find these DLLs in the PublicAPI folder of the TIA Portal installation under "...\Siemens \Automation\Portal V17\PublicAPI\V17".

Set the "Local copy" property of the DLLs to "False".

If no SIMOTION SCOUT is installed on your computer, you must add the reference to the following file in your project: Siemens.MC.Simotion.SideBySide.manifest.

You can find this file in the Bin folder of the TIA Portal installation under "...\Siemens\Automation \Portal V17\Bin".

Establishing version references in the configuration file

To ensure that TIA Portal and SIMOTION SCOUT can operate side by side in the correct version, you must enter the relevant references in the configuration file.

The configuration file "myScript.exe.config" is stored in the same directory as the script and must be transferred together with the script.

An example of how to adapt the configuration file to the TIA Portal V14 version with reference to the installation directories is given below. Adjust this to suit your installation.

```
<?xml version="1.0"?>
<configuration>
 <runtime>
 <assemblyBinding>
```

```
 <dependentAssembly>
 <assemblyIdentity name="Siemens.Engineering" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
 <codeBase version="14.0.0.0" href="FILE://F:\Program Files
(x86)\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.dll"/>
 </dependentAssembly>
 <dependentAssembly>
 <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
 <codeBase version="14.0.0.0" href="FILE://F:\Program Files
(x86)\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.Hmi.dll"/>
 </dependentAssembly>
 <dependentAssembly>
 <assemblyIdentity name="Siemens.MC.Simotion.Scripting"
culture="neutral" publicKeyToken="d29ec89bac048f84"/>
 <codeBase version="1.0.0.0" href="FILE://F:\Program Files
(x86)\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.MC.Simotion.Scripting.dll"/>
 </dependentAssembly>
 </assemblyBinding>
 </runtime>
</configuration>
```

Use of scripts

The procedure for external scripting of the SIMOTION data via TIA Portal is explained using an example in the following steps.

TIA Portal uses scripts based on .Net. C# is used in the following example.

1. Create the TIA Portal application object:
   ```
   TiaPortal tiaPortal = new TiaPortal();
   ```
2. Open an existing project:
   ```
   ProjectAggregation projects = tiaPortal.Projects;
   string projectPath = @"C:\Demo\AnyCompanyProject.ap14";
   Project tia_proj = projects.Open(projectPath);
   ```

3. Synchronize the hardware configuration data of TIA Portal with the SIMOTION data in SIMOTION SCOUT TIA:
```
Siemens.Engineering.IEngineeringServiceProvider iServProv =
project as Siemens.Engineering.IEngineeringServiceProvider;
Siemens.Engineering.Simotion.SimotionProvider iSimotionProv =
iServProv.GetService<Siemens.Engineering.Simotion.SimotionProvider
>();
System.String strSimotionAppID = iSimotionProv.Initialize();
```
4. Create an application object for the SIMOTION scripting.
```
Type comType = Type.GetTypeFromProgID(strSimAppID);
Siemens.MC.Simotion.Scripting.Application SimApp =
Activator.CreateInstance(comType) as
Siemens.MC.Simotion.Scripting.Application;
```
5. Assign the project path to the SIMOTION application object:
```
SIMOTIONLib.Project SimProj = SimApp.AttachToProject(projectPath);
```
You can now automate configuration sequences, e.g. for drive objects (DOs), with the scripting function in SIMOTION SCOUT TIA.
6. Before you close the TIA Portal project, release the project again in the SIMOTION application object.
```
SimApp.DetachFromProject();
tia_proj.Close();
```

Additional references

Detailed information on scripts in the TIA Portal can be found at:

https://support.industry.siemens.com/cs/ww/en/view/108716692 (https://support.industry.siemens.com/cs/ww/en/view/108716692)

Detailed information on scripts in SIMOTION SCOUT TIA can be found in Section "Scripts for sequence automation" in the online help.

3.6 SIMOTION SCOUT TIA device proxy

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT TIA, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

3.6.1 Fundamental safety instructions

3.6.1.1 General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

3.6.1.2 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

3.6.1.3 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

3.6.1.4 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

3.6.1.5 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

3.6.1.6 Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

3.6.2 Overview

3.6.2.1 Comfort Panels with SIMOTION SCOUT

Inter Project Engineering (IPE) in the TIA Portal

The TIA Portal makes it possible to carry out the technological and HMI configuration in various projects. For example, one processor can carry out the HMI configuration and a second processor can carry out the technological configuration. In the TIA Portal, this working method is called "Inter Project Engineering".

Note

The "Device Proxy" functionality in the TIA Portal can only be used for SIMOTION when SIMOTION SCOUT TIA V4.4 has been installed.

Using Comfort Panels with SIMOTION SCOUT

Inter Project Engineering is primarily used with SIMOTION to make flexible HMI panels available for SIMOTION SCOUT, SIMATIC STEP 7 V5 and WinCC,which can only be configured in the TIA Portal.

The following application scenarios are relevant to SIMOTION SCOUT:

- Migration of an existing HMI configuration from a SIMOTION SCOUT project.
- Use of panels in SIMOTION SCOUT/WinCC flexible that can only be configured via the TIA Portal, e.g. Comfort Panels.

For this purpose you configure a device proxy in the TIA Portal with which you connect the Comfort Panel for the HMI visualization. The device proxy is a proxy object in the TIA Portal for devices that are not part of the TIA project. Through initialization of the device proxy, the configuration of the SIMOTION controller is simulated in the TIA Portal.

Distributed working with SIMOTION SCOUT TIA

It is also possible to carry out the technological and HMI configuration in various projects for SIMOTION SCOUT TIA via a device proxy.

Procedure for using Comfort Panels with SIMOTION SCOUT

The following steps must be performed in the TIA Portal for the editing:

1. Create a project in the TIA Portal
2. Create a device proxy.
3. Initialize the device proxy, see also Create a device proxy and initialize via a project file (Page 963)
   The project file of the source project must be available and suitable.

4. Create an HMI Panel and establish a connection to the device proxy, see also Configuring Comfort Panels in a SIMOTION SCOUT project (Page 974).
5. Interconnect the HMI tags.

Procedure for "Distributed working with SIMOTION SCOUT TIA"

A controller with the technological configuration is already present.

The following steps must be performed in the TIA Portal for the editing:

1. Create the device proxy data in the TIA Portal.
2. Export the settings in the TIA Portal as an IPE file, see also Creating and configuring device proxy data (Page 968).

In order to create the HMI visualization in a separate project, the following steps must be carried out:

1. Create a project in the TIA Portal
2. Create a device proxy.
3. Initialize the device proxy with the exported IPE file, see also Initialize a device proxy via an IPE file (Page 969)
   The IPE file of the source project must be available and suitable.
4. Create the HMI Panel and establish a connection to the device proxy.
5. Interconnect the HMI tags.

See also

Project with integrated WinCC flexible HMI configuration (Page 976)

Basics of Inter Project Engineering (IPE)/device proxy (Page 960)

3.6.2.2 Basics of Inter Project Engineering (IPE)/device proxy

Introduction to Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to extract the HMI data in a source project to exchange it using a device proxy. You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration.

You can exchange the following PLC data between projects via the device proxy for SIMOTION SCOUT:

- Variables
- Messages

The following data is automatically included in the exchange:

- Controller communication interfaces
- Configured communication processors and communication modules

This automatic data exchange ensures that the interfaces and the configured communication are transferred along with the data you selected. This data is required to allow you to continue working consistently and without problems in your target project.

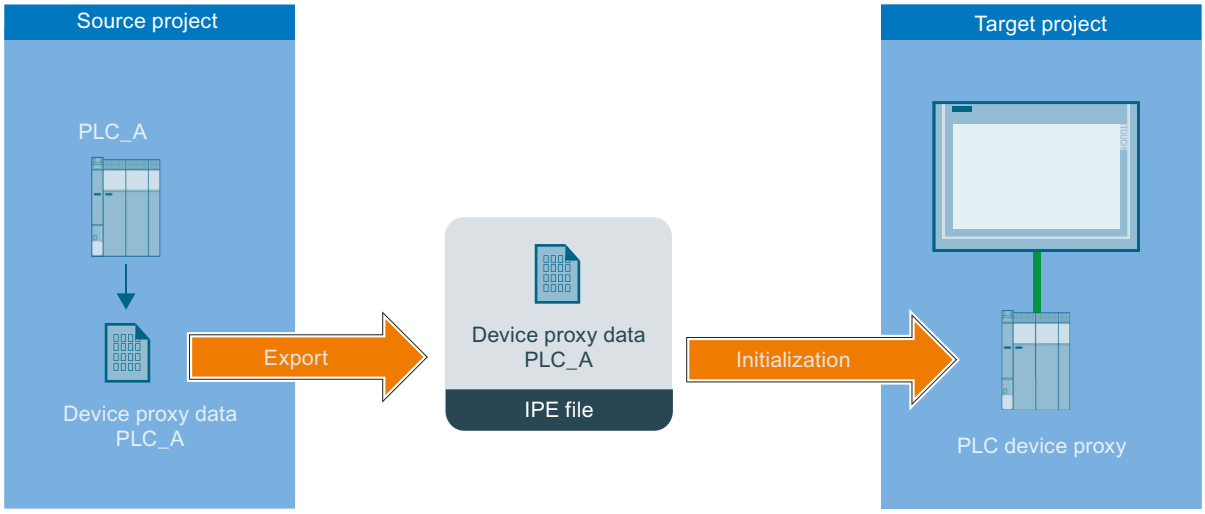
Cross-project data exchange

The following options are available for cross-project data exchange with Inter Project Engineering:

- Exchanging PLC data via a project file
  - for data from SIMOTION SCOUT; the project file may contain several controllers.
- Exchanging PLC data via an IPE file
  - for data from TIA Portal projects V13 or higher; an IPE file must also be generated for each controller.

Exchanging PLC data via a project file

The figure below shows cross-project data exchange via a project file.

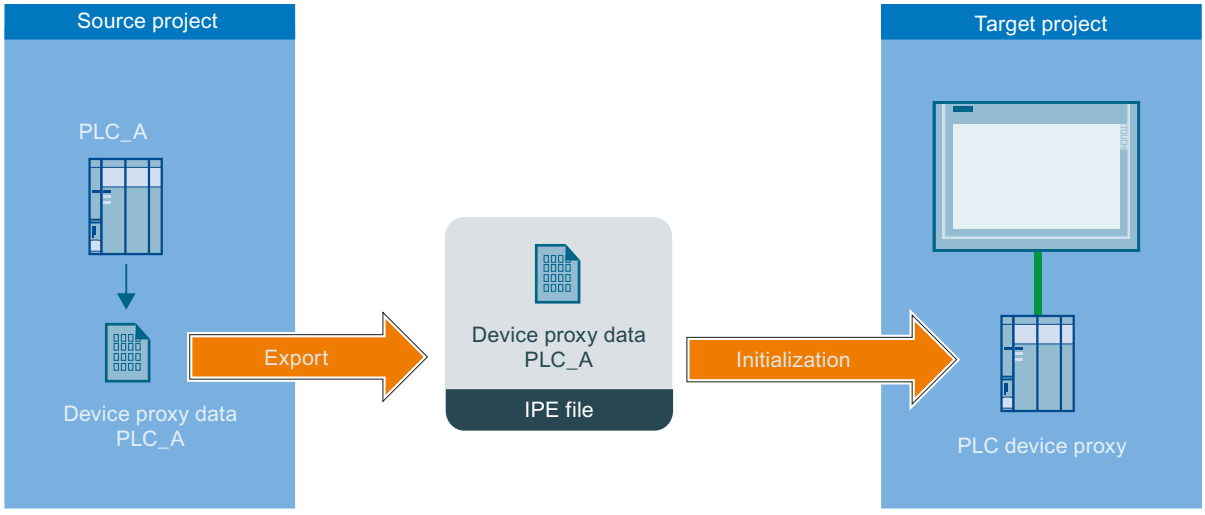


The PLC data from the source project is transferred to the PLC in the target project via a SIMOTION project file (*.mcp).

You can transfer data from STEP7/SIMOTION SCOUT projects that are not integrated in the TIA Portal relating to the HMI visualization conveniently into your current TIA Portal projects and continue to use it there.

Exchanging PLC data via IPE file

The figure below shows cross-project data exchange via an IPE file. Both projects are created in the TIA Portal.



The PLC data from the source project is transferred to the PLC in the target project via the object "Device Proxy Data".

PLC data that is exchanged via an IPE file must originate from TIA Portal projects of V13 or higher. You must generate an IPE file for each controller.

3.6.2.3 Requirements for Inter Project Engineering (IPE) / device proxy

Software and hardware requirements

The following requirements must be met in order for you to use Comfort Panels in SIMOTION SCOUT:

- You have installed TIA Portal as of V13 and the SIMOTION SCOUT TIA V4.4 option package to use Comfort Panels
- You have installed the SIMOTION SCOUT software package as of V4.4 and therefore also SIMATIC STEP 7 V5.5 SP4 (but at least SIMATIC STEP 7 V5.5 SP1).
- You are using a SIMOTION C, P or D module version as of V4.3.
- When Comfort Panels are used, WinCC Comfort, WinCC Advanced or WinCC Professional as of V13 must be installed.
- For the migration of an integrated HMI project, SIMATIC WinCC flexible 2008 SP3 Update 3 must also be installed as a minimum.
- The same firmware version must be installed on all controllers involved.

Functionality of IPE in TIA Portal

- You have installed TIA Portal as of V13 and the SIMOTION SCOUT TIA option package as of V4.4.

Note

Operating systems

The SIMOTION SCOUT and STEP 7 V5.5 software packages are only approved for Windows XP and Windows 7. The TIA Portal is approved for Windows 7 and Windows 8.x. The device proxy for SIMOTION SCOUT is therefore only available to you for Windows 7.

Requirements for IPE and use of Comfort Panels

The requirements for accepting PLC data from projects are as follows:

- You have a project with hardware configuration that is to be transferred to another project.
- You have ensured that the project is consistent before you transfer the PLC data using the device proxy data.

3.6.3 Create and configure device proxy.

3.6.3.1 Initialize a device proxy via a project file

Create a device proxy and initialize via a project file

Introduction

You initialize the device proxy using a project file.

A device proxy must be created for every controller that has a connection to an HMI panel. If the project file contains several controllers, you can select the controller via the "Initialize device proxy" dialog and assign to the device proxy.

Note

PROFIBUS interfaces

Three PROFIBUS interfaces are shown in the network view and device view for SIMOTION D4xx devices, although there are only two interfaces on the devices. The third interface symbolizes the PROFIBUS Integrated. Open the device overview to find out which icon represents the PROFIBUS Integrated interface.

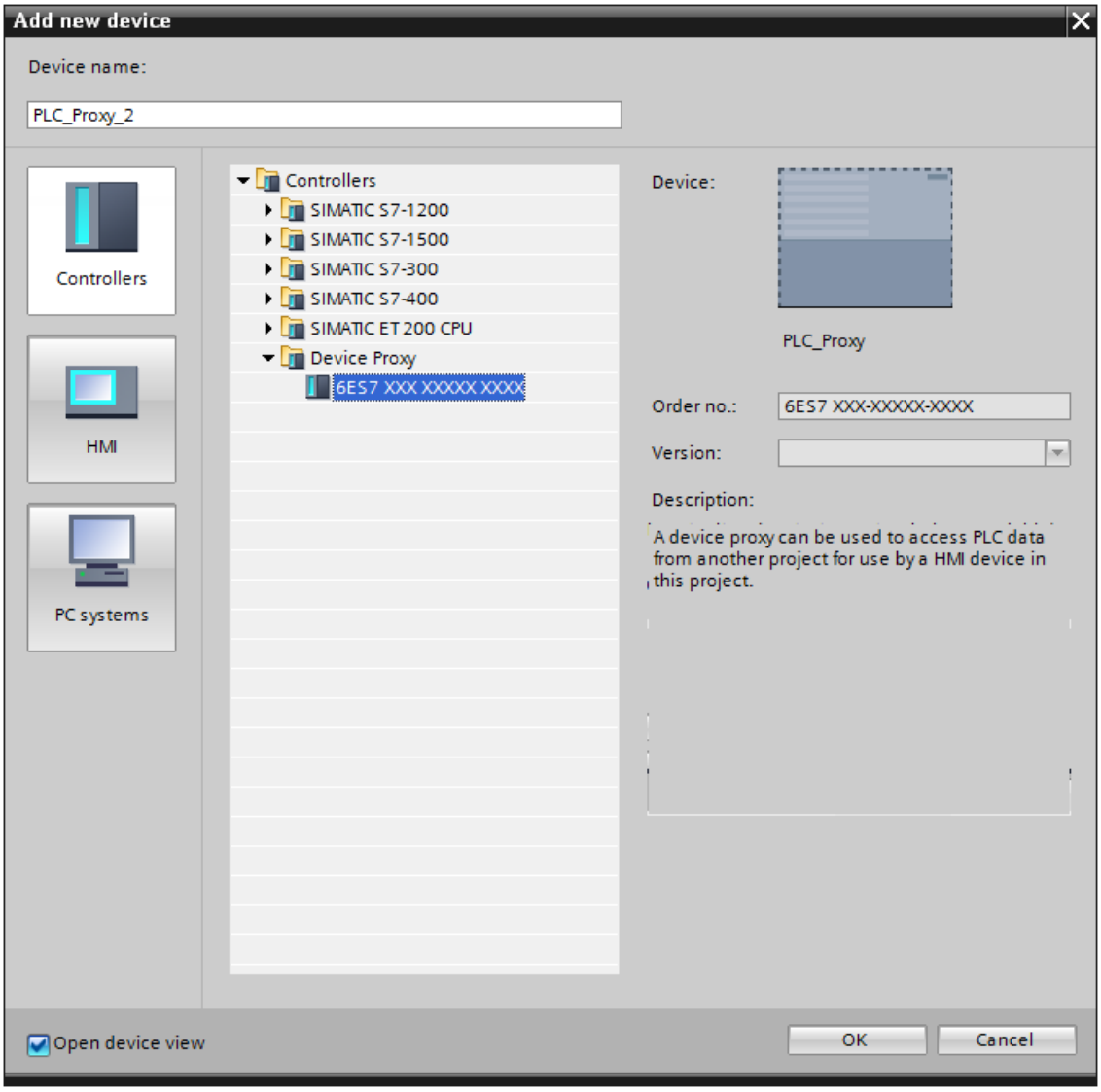
Requirement

One of the following projects exists:

- SIMOTION V4.4 project (*.mcp); you will find the mcp file in the project folder in the "u7" folder.
- TIA Portal V13 Project (*.ap13)

Procedure

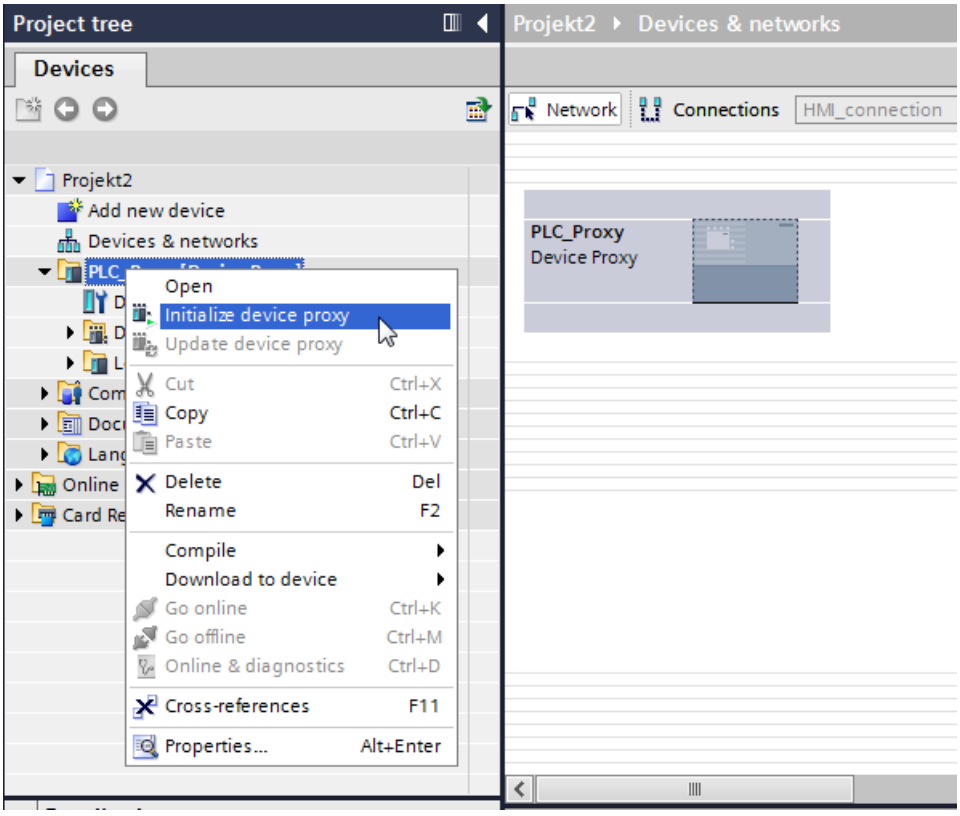
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in net view.

4. Click "Initialize device proxy" in the shortcut menu.



5. In the "Open device proxy data source" dialog box, select from the following entries:
   - "SIMOTION Classic Project (*.mcp)" for initialization via a SIMOTION project file (for editing Comfort Panels)
   - "TIA Project (*.ap13)" for initialization via a TIA Portal project file (only for SIMOTION SCOUT TIA).
6. Select the project file and click "Open".

7. The "Initialize device proxy" dialog box opens.

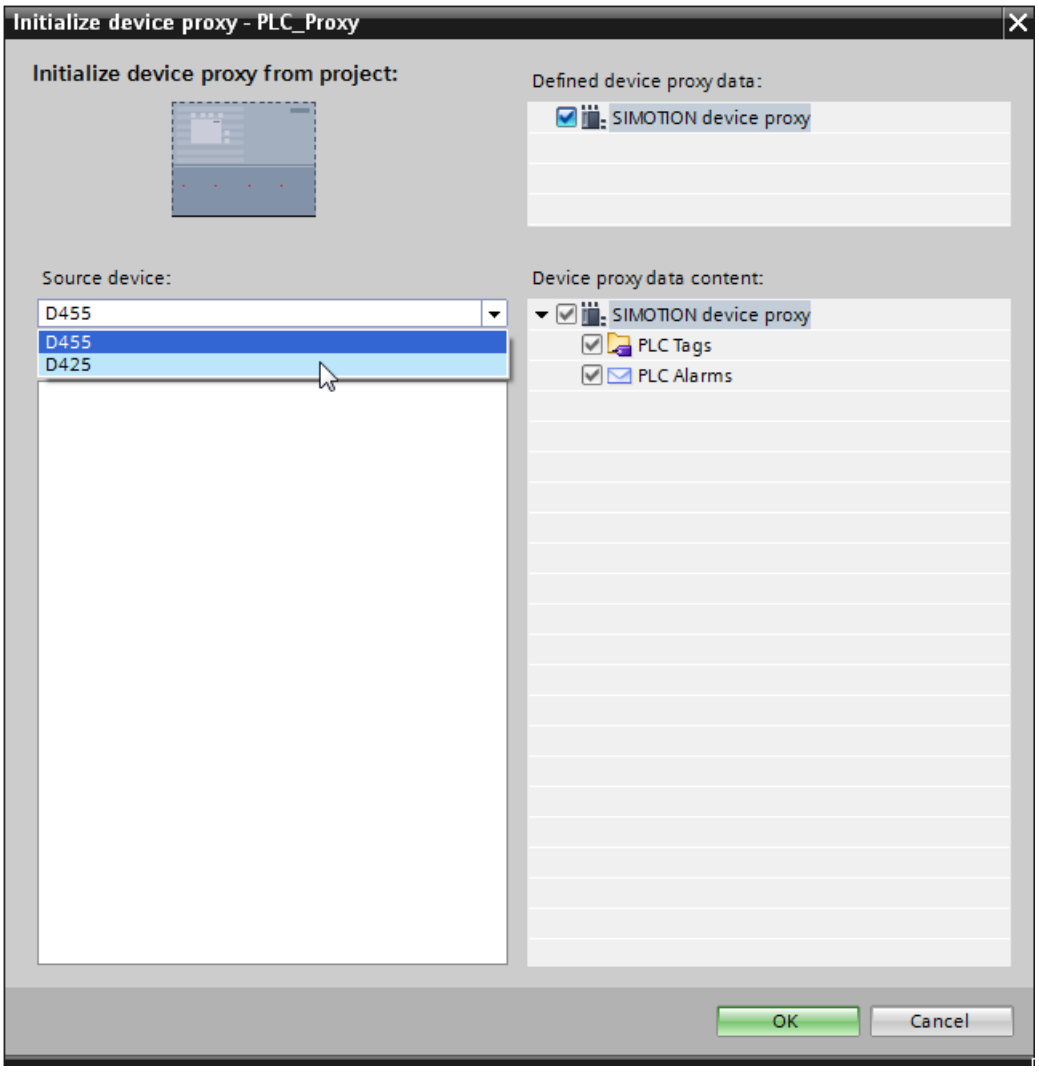


Figure 3-514 Initializing the device proxy

8. Select the source device if several controllers are configured in your project. You cannot change any other settings.
9. Click "OK".

Result

Following initialization, the PLC data from the project file in the selected device proxy data object is stored in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

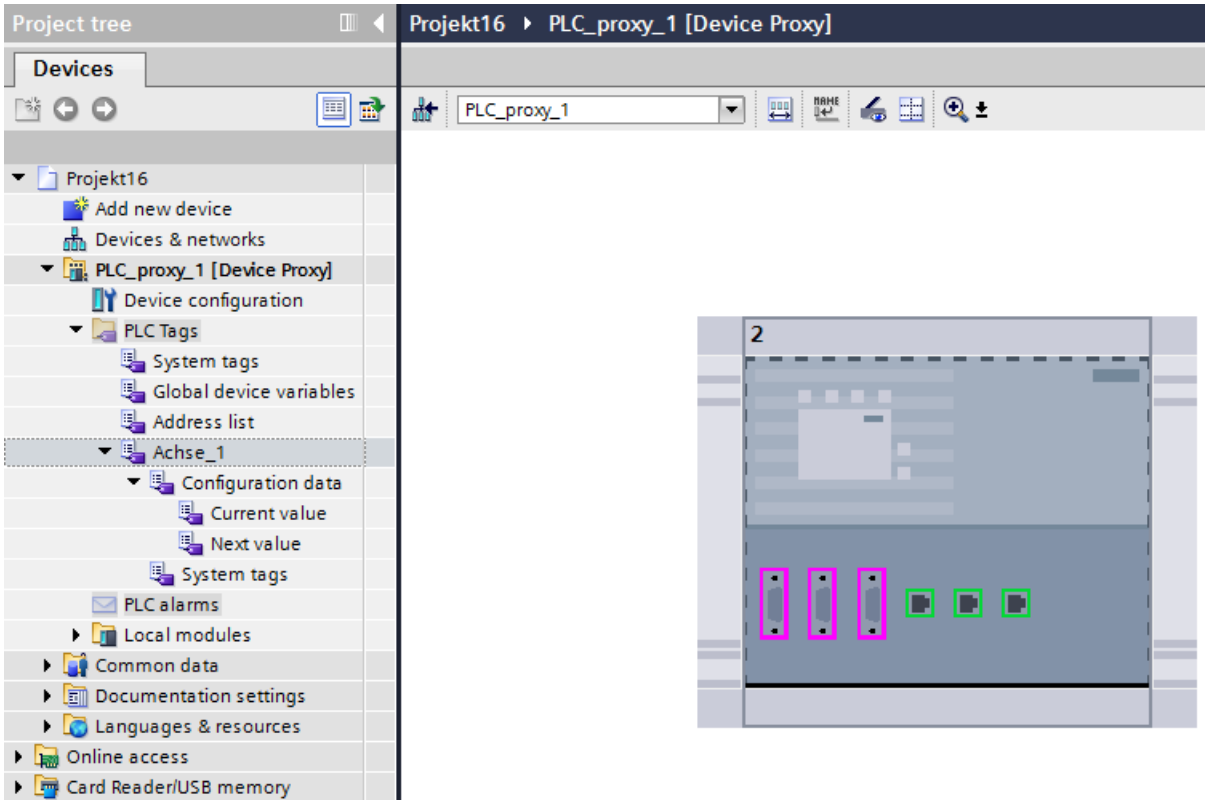


Figure 3-515 Device proxy initialized

See also

Configuring Comfort Panels in a SIMOTION SCOUT project (Page 974)

Updating a device proxy via a project file

Introduction

If changes have been made to the PLC data or the communications configuration in the source project of the device proxy, you can update the device proxy in your TIA Portal project.

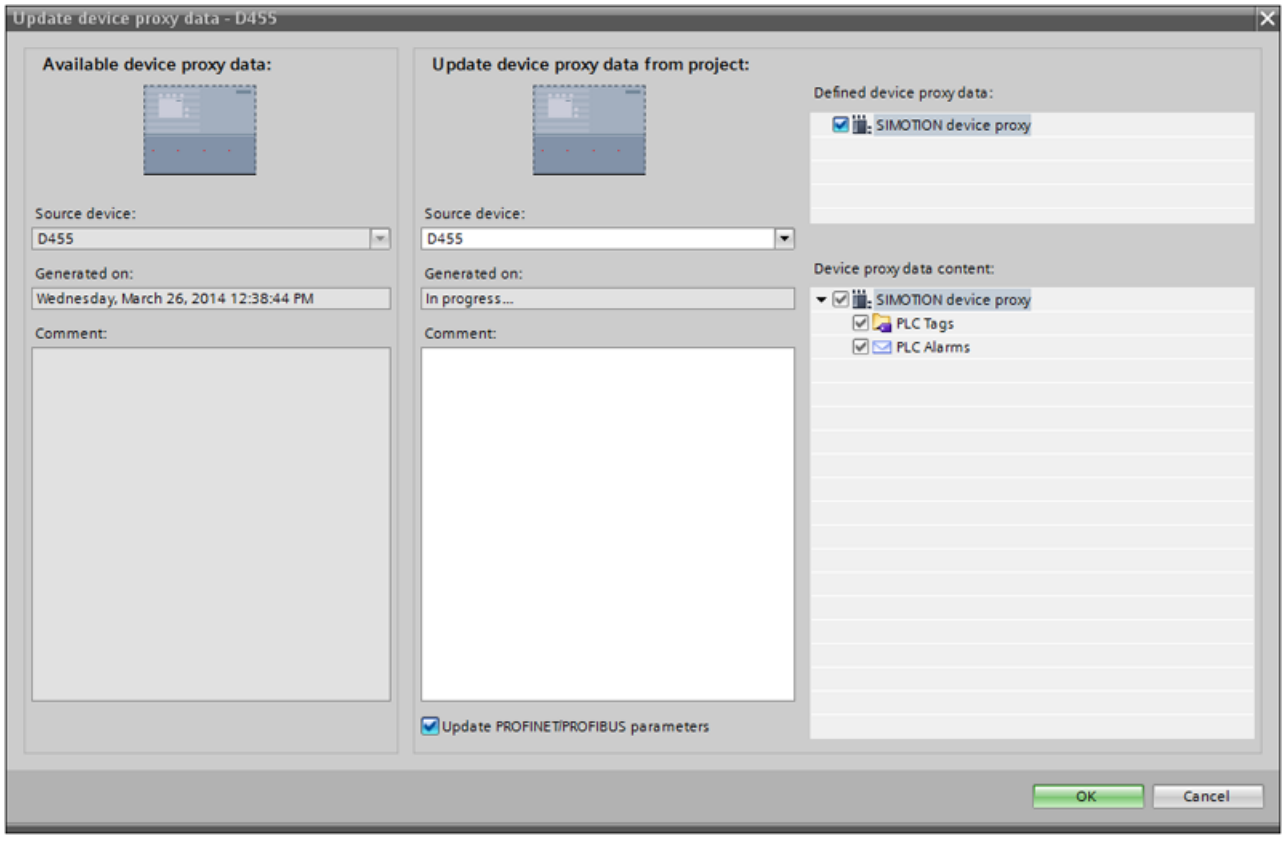
Requirement

- The project file has been generated from the source project of the device proxy.
- A device proxy that has already been initialized is available in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Select the "Update device proxy" entry in the shortcut menu.

3. Select the project file.
   The "Update device proxy" dialog opens.



4. Select a device.
5. Select whether you wish to update the PROFINET or PROFIBUS parameters.
6. Click "OK".

3.6.3.2 Initialize a device proxy via an IPE file

Creating and configuring device proxy data

Creating and editing device proxy data

You need device proxy data to generate an IPE file. With this file you can then replace and transfer configurations in the TIA Portal.

To create device proxy data, follow these steps:

1. Open the "Device Proxy Data" folder below the CPU to which you want to create device proxy data.
2. Double-click the "Add new device proxy data" command.
   A new entry is created.

3. Double-click the entry.
   In the working area, the settings for the device proxy data are now displayed.

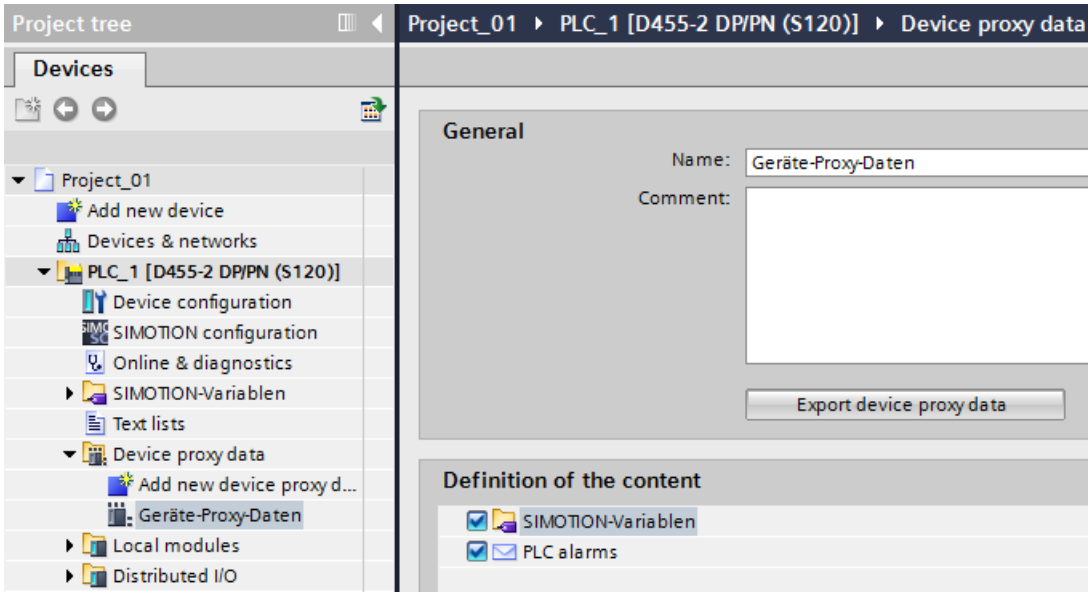


Figure 3-516 Exporting device proxy data

4. Select the content of the data:
   - PLC tags
   - PLC messages
5. Click the "Export proxy data" button to open the "Export device proxy data" dialog box.
6. Enter a meaningful name and click the "Save" button to generate the IPE file.

Initialize a device proxy via an IPE file

Introduction

You initialize device proxy data using a dialog on the device proxy in the TIA Portal.

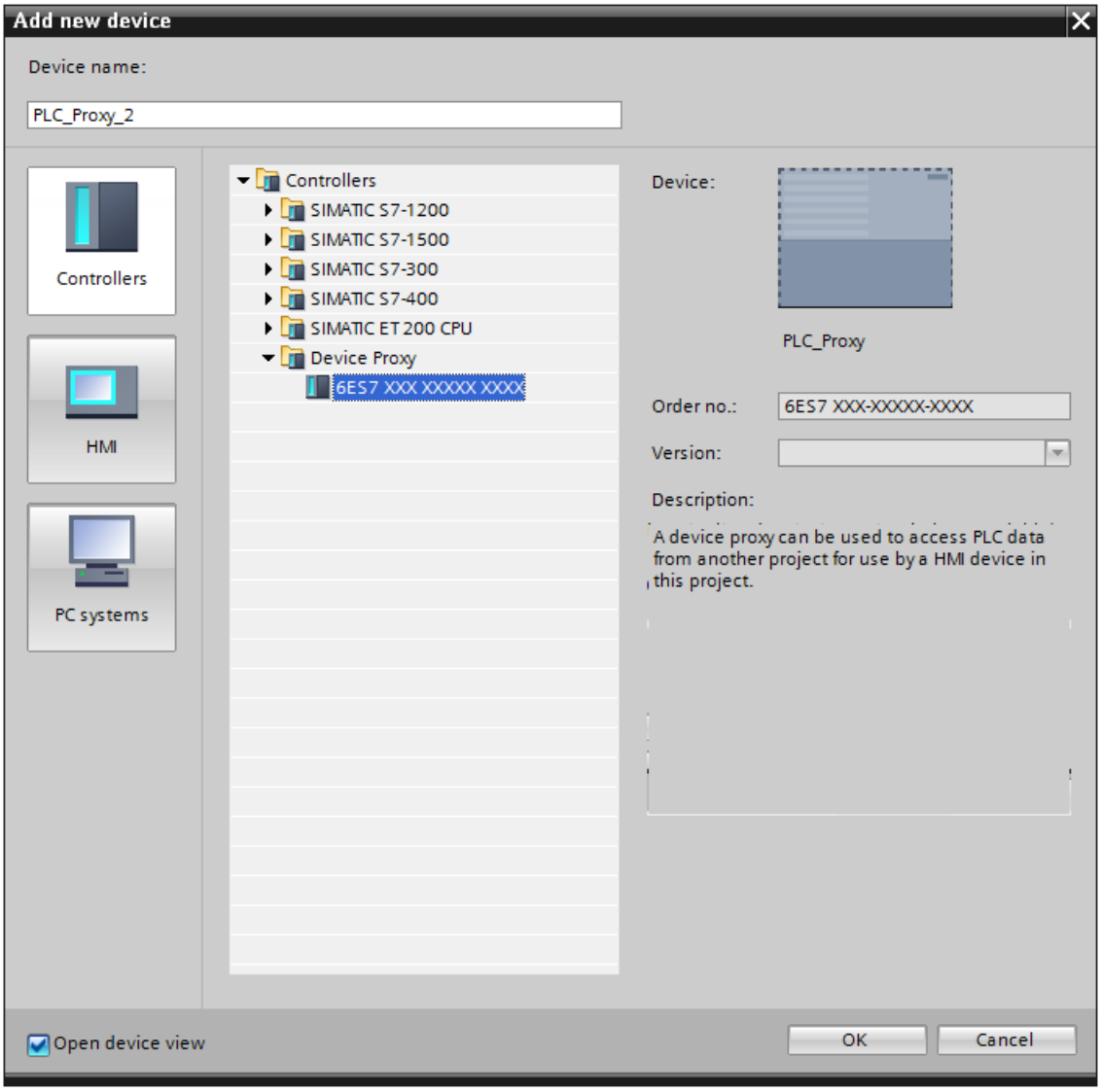
The IPE file contains the device proxy data from the source project.

Requirement

IPE file is available.

Procedure

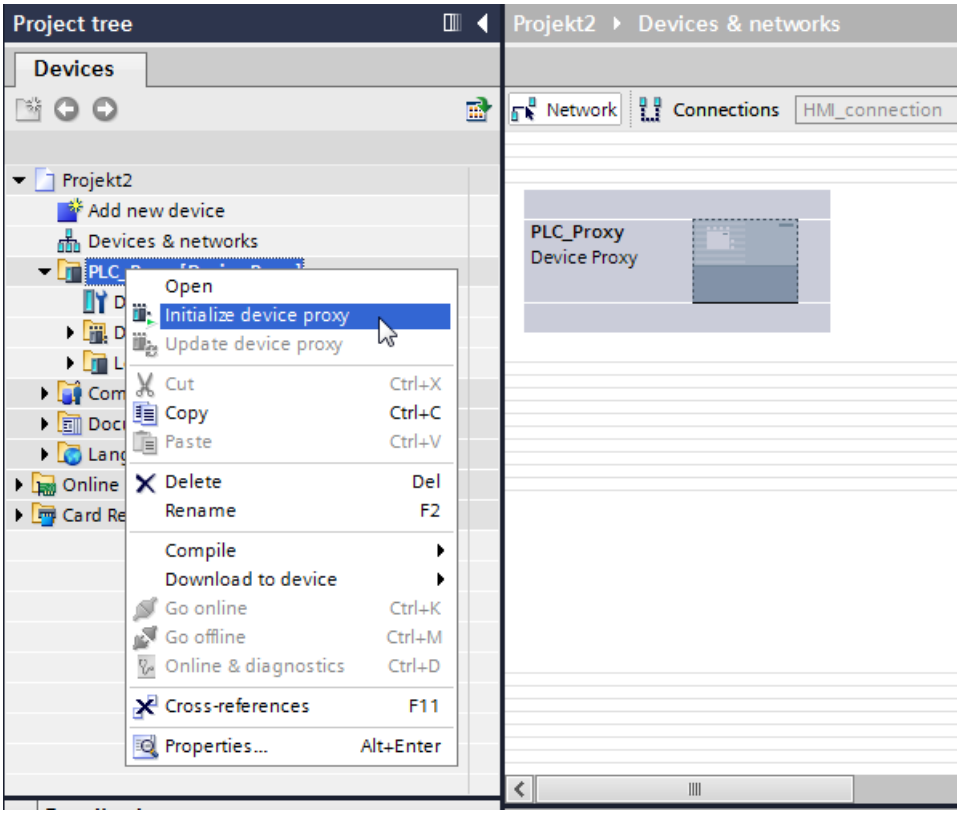
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in net view.

4. Click "Initialize device proxy" in the shortcut menu.



5. Select the IPE file and click the "Open" button.
   The "Initialize device proxy" dialog opens.

   Note

   You cannot select the device proxy data before initialization in the target project.

   All device proxy data included in the IPE file is transferred.

   Click "OK".
   Initialization of the device proxy is started.

Result

Following initialization, all device proxy data from the IPE file is contained in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

See also

Create a device proxy and initialize via a project file (Page 963)

Updating a device proxy via an IPE file

Introduction

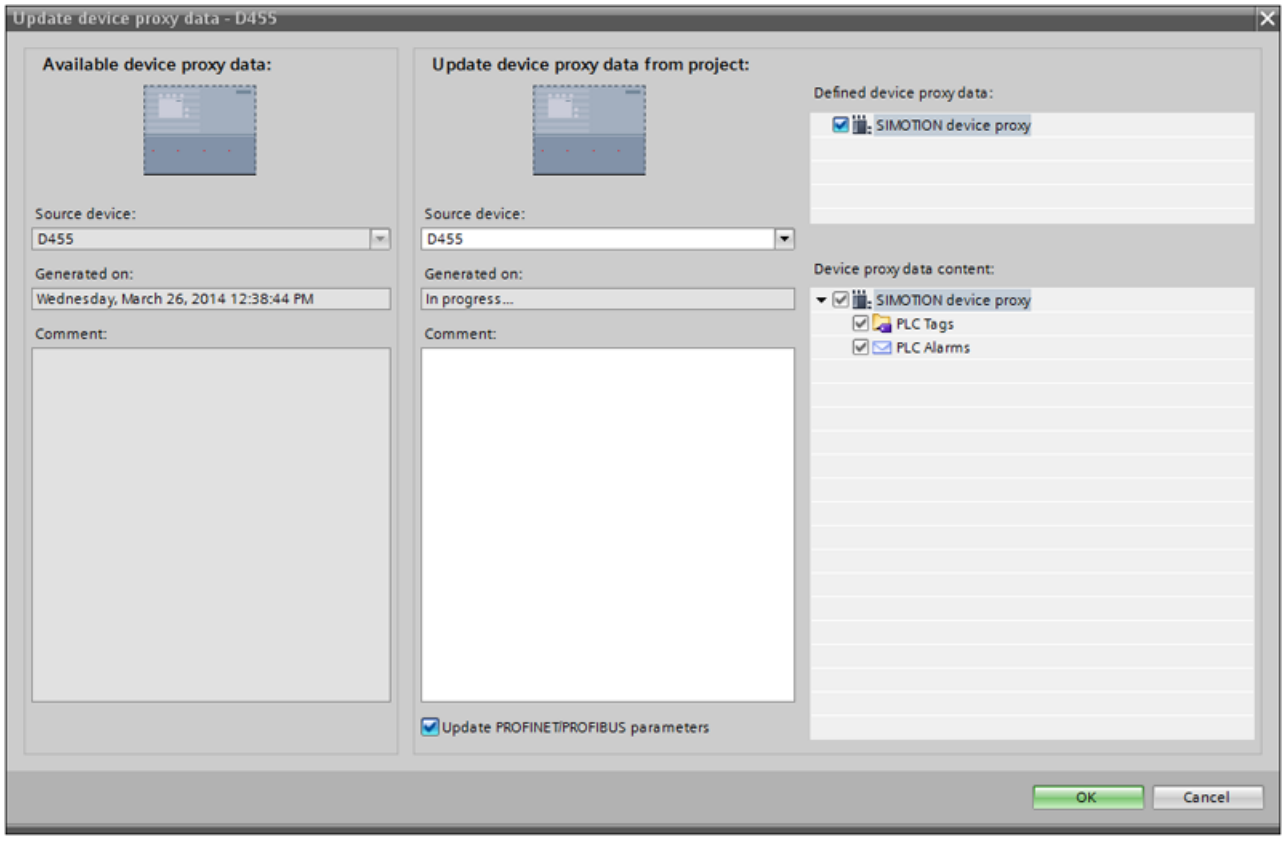
If changes were made to the device proxy data in the device proxy source project, the device proxy data can be updated in the target project.

Requirement

- The new IPE file contains the device proxy data from the source project.
- A device proxy has already been initialized using an IPE file in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Select the "Update device proxy" entry in the shortcut menu.
3. Select the IPE file.



4. Select whether you wish to update the PROFINET/PROFIBUS parameters.
5. Click "OK".

3.6.4 SIMOTION SCOUT with Comfort Panels

3.6.4.1 SIMOTION SCOUT data

Data for the device proxy

With the initialization, the device proxy accepts the data set present in the project file:

- System variables
- Config data
- Global variables
- Message configuration
  The device proxy accepts the message configuration of the source. When you create a new message, you must update the device proxy accordingly.
  TO alarms are also synchronized and created in the TIA Portal.
  Alarm_S messages are created only when they are used in the source project via a message call in a program. If you use the Alarm_S messages only after the initialization, you must update the device proxy.
- Communication configuration
  During the initialization, the settings for the communication interfaces are transferred automatically (e.g. IP address, subnet mask). If you change the settings in the source project, you must update the device proxy.

3.6.4.2 Using Comfort Panels with SIMOTION SCOUT

Using HMI panels

The device proxy is intended for applications in which the automation is still to be configured with SIMOTION SCOUT but the HMI connection made via the TIA Portal.

This is the case, for example, when functions are used in the project that are not available in the TIA Portal (e.g. SIMOTION Drive Control Chart, DCC) and on the other hand HMI panels are being used that are only available in the TIA Portal.

In this case, the automation project remains in SIMOTION SCOUT and only the HMI configuration is carried out in the TIA Portal.

See Requirements for Inter Project Engineering (IPE) / device proxy (Page 962) for an overview of the software requirements.

Note

Comfort Panels are available only for WinCC Comfort, WinCC Advanced and WinCC Professional V13 or higher.

This may result in the following application scenarios:

- You are setting up a new project in SIMOTION SCOUT and using the TIA Portal for HMI configuration.
- You already have a project and would like to reconfigure an HMI in the TIA Portal.
- You already have a project with an HMI configuration and would like to switch to a Comfort Panel.

You generally proceed as follows:

- Configure the SIMOTION components in SIMOTION SCOUT.
- Configure the HMI in the TIA Portal.
- For the SIMOTION part, perform a download from SIMOTION SCOUT.
- For visualization in HMI, perform the download from the TIA Portal.

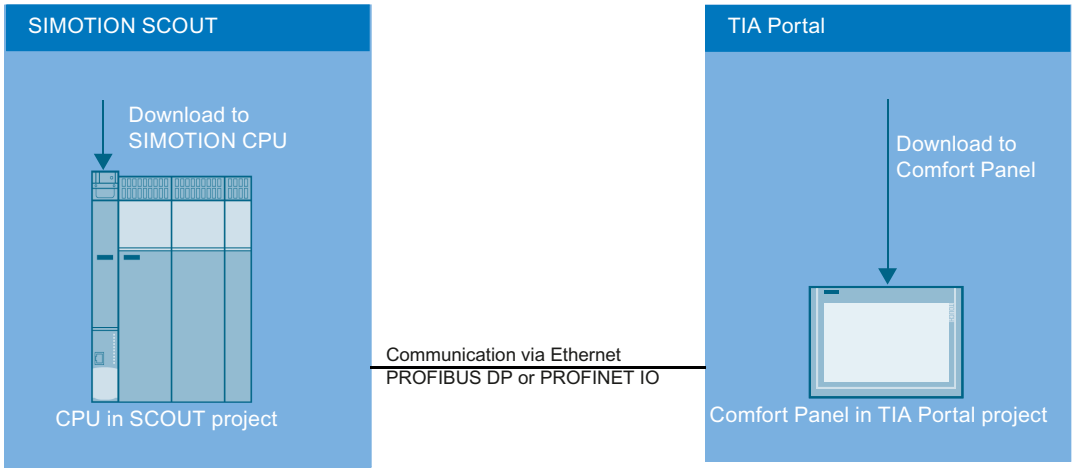


Figure 3-517 SIMOTION SCOUT with Comfort Panel

3.6.4.3 Configuring Comfort Panels in a SIMOTION SCOUT project

Integrating a Comfort Panel via a device proxy

If you use Comfort Panels, you require a connection between the TIA Portal and SIMOTION SCOUT.

If you use an older project, you must first upgrade it to SIMOTION SCOUT V4.4.

Note

PROFIBUS interfaces

Three PROFIBUS interfaces are shown in the network view and device view for SIMOTION D4xx devices, although there are only two interfaces on the devices. The third interface symbolizes the PROFIBUS Integrated. Open the device overview to find out which icon represents the PROFIBUS Integrated interface.

To configure an HMI Panel, proceed as follows:

1. Create a project in SIMOTION SCOUT and add the modules that you need.
2. Configure the communication and the technology.
   – If you have not fully configured the project, you can update the device proxy at a later date (see also Updating a device proxy via a project file (Page 967))
3. Save and compile the project.
4. Open the TIA Portal (as of V13).
5. Create a project and insert a device proxy.
6. Initialize the device proxy using a project file (see also Create a device proxy and initialize via a project file (Page 963)).
7. To do that, in the device view right-click the device proxy and execute "Initialize device proxy". The "Open data source for device proxy" dialog box opens.
8. Select "SIMOTION Project" as the filter for the data source.
9. In the file system, navigate to your project and select the project file.
10. Apply the setting with "Open."

Result

The device proxy has now been initialized with the settings of the SIMOTION module.

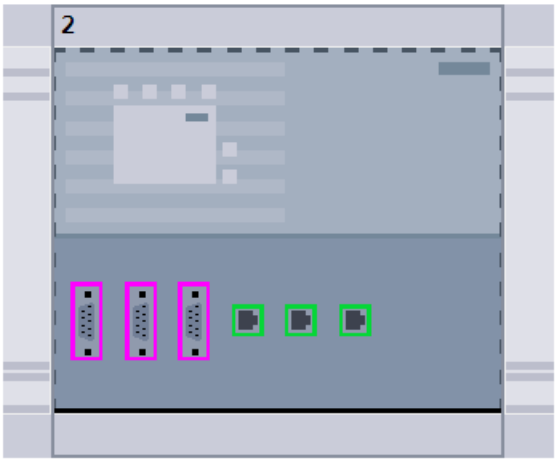


Figure 3-518 Device proxy initialized

Inserting and connecting HMI

Once you have initialized the device proxy, add the HMI and configure the connection.

1. Change to the network view.
2. Select a Comfort Panel in the hardware catalog ("HMI > SIMATIC Comfort Panels").
3. Drag the Comfort Panel into net view.
4. Configure the communication between device proxy and HMI panel in the same way as the SIMOTION SCOUT project:
   – Use the same interfaces and ports (PROFINET only) for the network connection with PROFIBUS or PROFINET as in SIMOTION SCOUT.
5. Change the net view from "interconnect" to "connections", select the "HMI connection" entry in the drop-down list and drag a connection from the device proxy to the HMI panel.
   The connection is created.

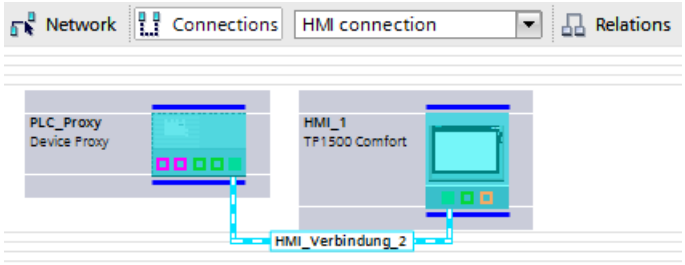


Figure 3-519 Device proxy HMI connection

This connection is an integrated connection and like the connections with SIMOTION devices, it cannot be edited in the WinCC connection editor.

Download

1. Carry out the additional configuration in SIMOTION SCOUT and in the TIA Portal.
2. Load the SIMOTION configuration from SIMOTION SCOUT into the controller.
3. Load the HMI configuration from the TIA Portal into the Comfort Panel.

3.6.4.4 Project with integrated WinCC flexible HMI configuration

Migrating a Classic HMI configuration

If you have a project that contains an HMI configuration, you can migrate this HMI project from WinCC flexible to the TIA Portal. To do this, you must extract the integrated HMI configuration from the SIMOTION SCOUT V4.4 project. This migration can also be used to switch from a panel to a Comfort Panel.

You will find an FAQ in the Internet about extracting the HMI configuration and subsequent migration:

Migrating an integrated project from WinCC flexible to WinCC (TIA Portal) (http://support.automation.siemens.com/WW/view/en/54695062)

After the migration, the project contains the migrated HMI with the old connections.

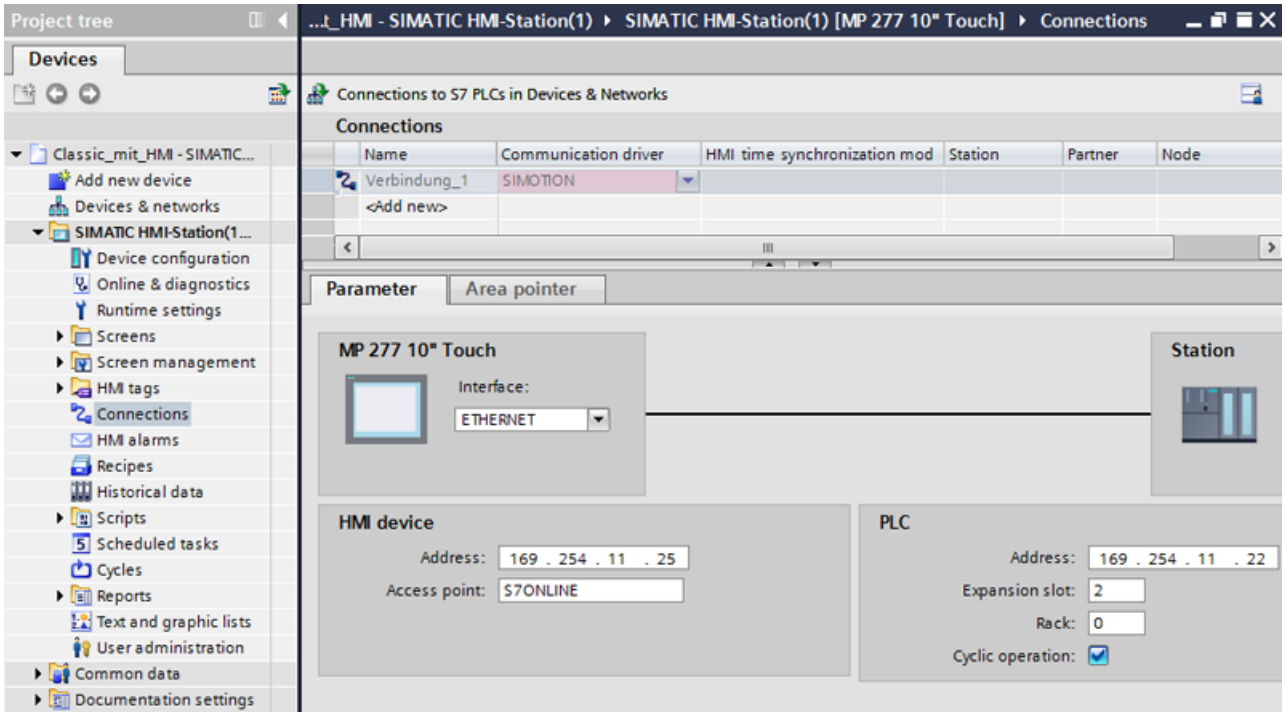


Figure 3-520 HMI migration

You must then perform the following additional steps:

1. Insert a device proxy and initialize it to the Classic project (see also Configuring Comfort Panels in a SIMOTION SCOUT project (Page 974)).
   The Classic project must reference the same HMI tags as the migrated project.
2. If you want to use a Comfort Panel, perform a device replacement.
3. Right-click the panel in the net view and select the "Device / change version" entry.
   The "Replace device" dialog opens.

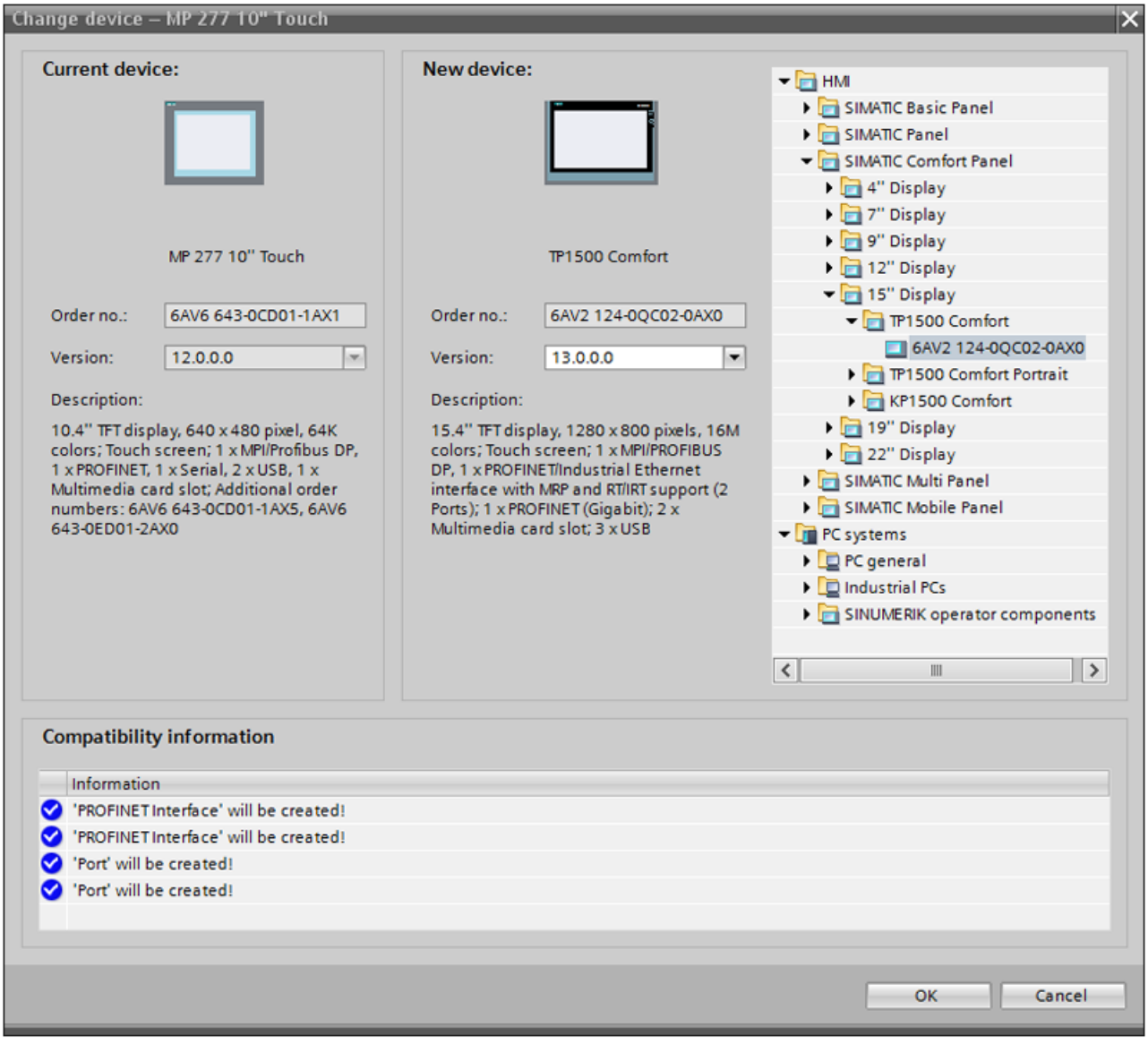


Figure 3-521 Replacing a device

Note

Note the compatibility information.

Establishing a connection and synchronizing HMI tags

1. Establish a connection between the device proxy and the HMI. This connection is an integrated connection and like the connections with SIMOTION devices, it cannot be edited in the WinCC connection editor.
2. Click the "Connections" button and select the device in the net view. Also select "HMI connection" in the drop-down list.
3. Move one device to the other one using drag-and-drop. The TIA Portal creates an HMI connection.

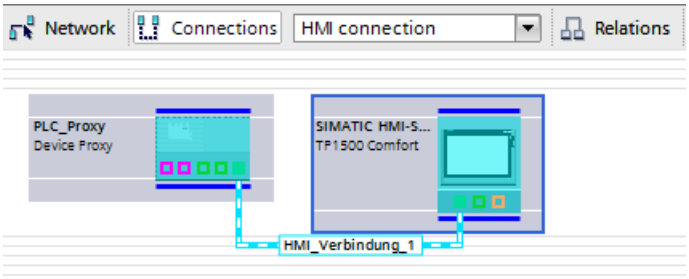


Figure 3-522 HMI device proxy connection

4. Double-click the "Connections" entry under HMI variables in the project navigation. The panel settings are displayed.

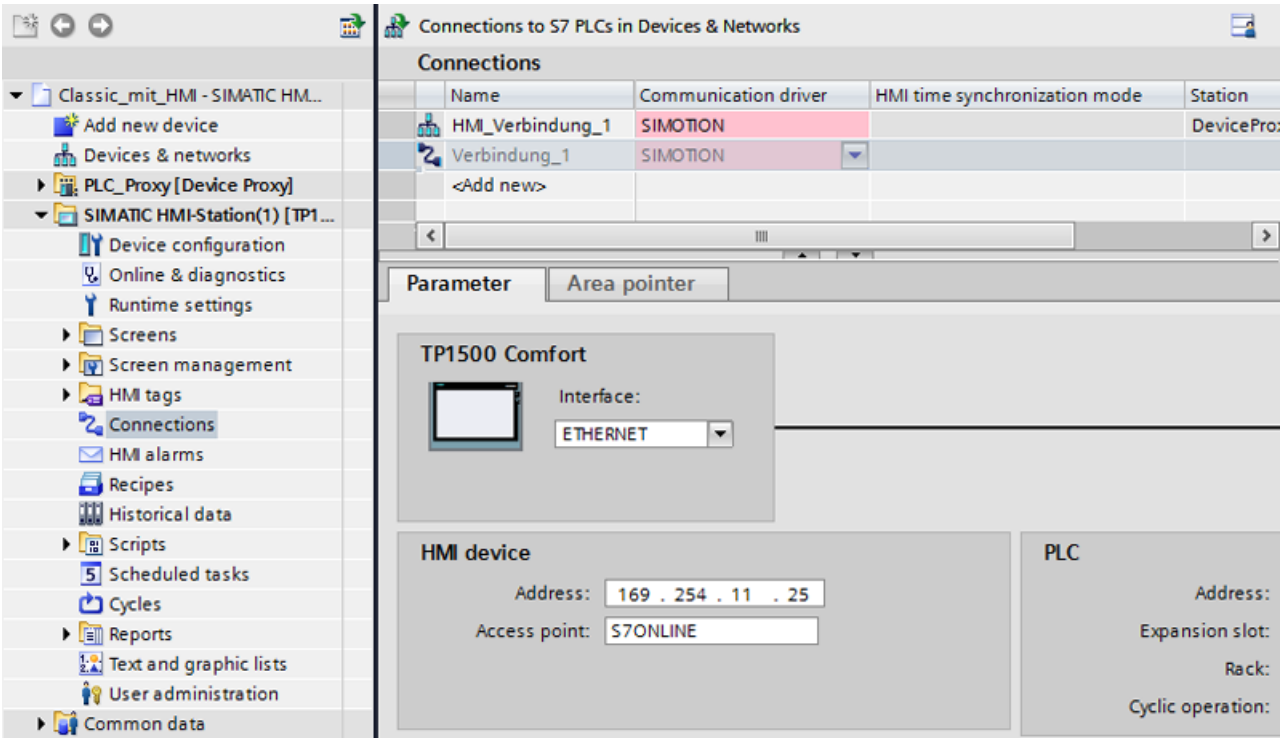


Figure 3-523 Customizing connections

5. Note the name of the old connection (here "Connection_1") and delete this connection. The old connection is invalid after the migration.
6. Change the name of the new connection (between panel and device proxy) by entering the name of the old connection.

7. Perform a comparison of the old HMI variables.
8. Right-click the "HMI variables" entry in the project tree and "Synchronize for comparison with the PLC variable".
   The following dialog opens:

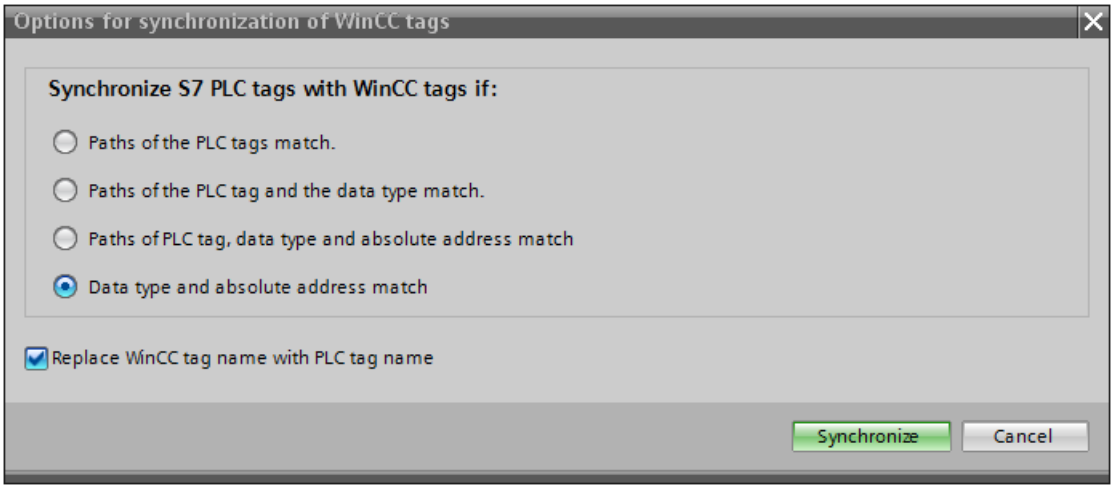


Figure 3-524 Synchronizing PLC variables dialog

9. Activate the "Data type and absolute address match" option. If you have renamed one of the variables, it is possible that it is connected with an incorrect variable. Consequently, renamed variables should be reassigned manually.
   If a variable is incorrectly connected, this error will not occur until during RT communication.
10. Activate the "Replace WinCC variable name with PLC variable name" option.

The project that was used for initializing the device proxy must contain the same set of tags as the old project.

Add new tags after the migration and synchronization, and interconnect them. If you change the source project settings in SIMOTION SCOUT, you must update the device proxy.

3.6.4.5 Using direct keys

Use of panel keys as direct keys

The direct keys are used to directly influence the machine status from the HMI Panel. For example, a direct key can be pressed to stop an axis.

The following files must exist for the configuring:

- For PROFIBUS DP, a GSD file for the panel
- For PROFINET IO, a GSDML file for the panel

Requirement

You already have a project with a device proxy that is connected with a Comfort Panel, and a SIMOTION controller exists.

Configuring in HW Config

As you cannot select the Comfort Panel in the HW Config of SIMOTION SCOUT, you must add the device to the HW Config using GSD import. Only then can you configure the direct keys. The GSD/GSDML files are not currently available; contact SIEMENS Customer Support in this connection.

To configure a Comfort Panel, proceed as follows:

Configuring in HW Config

1. Switch to SIMOTION SCOUT and open HW Config.
2. Perform "Options > Install GSD files".
   The "Install GSD files" dialog box is displayed.
3. Select the GSD/GSDML file, and click "Install" and "Close".
   The file will be installed.
4. Navigate in the hardware catalog, for example to the "PROFINET IO > Further field devices > HMI" entry.
   The installed Comfort Panels are displayed in the folder.
5. Drag the panel to the PROFIBUS master system or to the PROFINET IO subsystem.
   Ensure that the panels have the same bus addresses (PROFIBUS address, IP address, etc.) as the devices configured in the TIA Portal.

The help system of the TIA Portal describes how you complete the configuration of a direct key with WinCC flexible.

Configuring the HMI in the TIA Portal for the use of direct keys

If you intend to use direct keys, you need to take a number of general conditions into account when configuring the Comfort Panel in the TIA Portal. The PROFINET device name and the converted device name of the HMI must be identical. The Comfort Panel must not be configured as an IO device. The configuration must be saved as a GSD/GSDML before the Comfort Panel is exported.

The proxy is set up in the TIA Portal and connected to the panel in the network configuration. The HMI connection has been configured.

Configuration in the TIA Portal

1. Check to make sure that the Comfort Panel is not configured as an IO device. The IO device operating mode must not be activated.

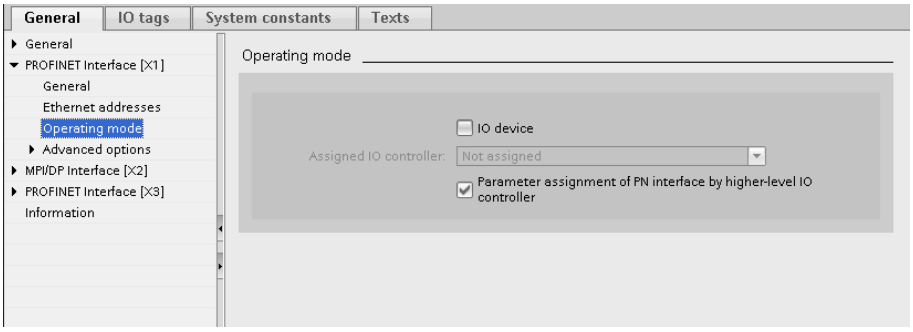


Figure 3-525 HMI operating mode not as IO device

2. Configure the PROFINET device name at the connected interface PNxIO (150) port X1.

3. Enter the device name, e.g. "hmi1_panel". The TIA Portal automatically generates a converted device name. In this example, this is hmi1xbpanel7da2.

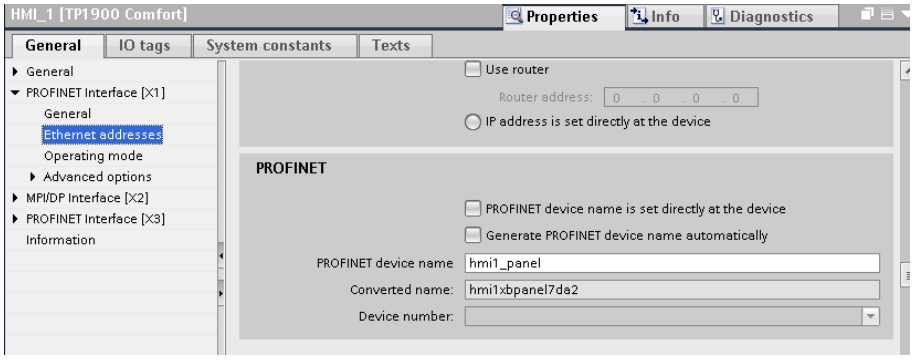


Figure 3-526 Device name of the Comfort Panel with converted device name

4. Copy the converted name into the field labeled "PROFINET device name".

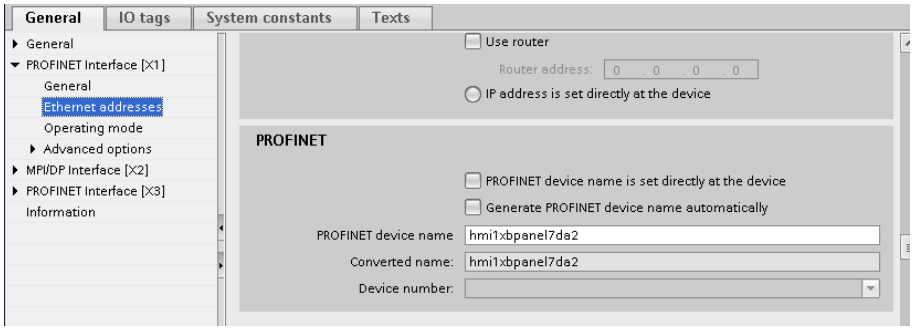


Figure 3-527 Converted name and PROFINET device name are identical

5. All essential configuring steps have now been completed. Continue the configuring process as described in previous chapters.

You can check which device name has been configured by selecting the Startcenter at the fully configured panel.

1. Select the "Startcenter" at the panel.
2. Select "Settings > PROFINET". The PROFINET window appears. The converted name of the panel and the PROFINET device name configured in the TIA Portal must be identical in this window. The name is hmi1xbpanel7da2 in our example.

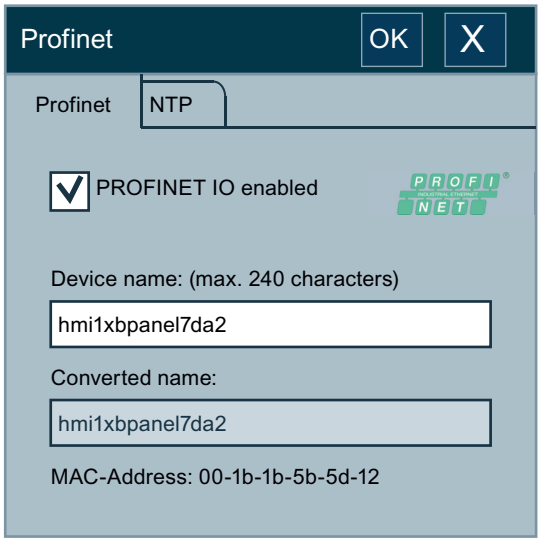


Figure 3-528 Converted name and PROFINET device name at the panel (schematic representation of the Startcenter)

See also

Create a device proxy and initialize via a project file (Page 963)

System and Function Descriptions 4

4.1 Basic Functions for Modular Machines

Preface

Preface

This document is part of the **SIMOTION System and Function Descriptions** documentation package.

Scope

This document applies to SIMOTION SCOUT, the engineering system of the SIMOTION product version V5.2 in conjunction with:

- A SIMOTION device with the following versions of the SIMOTION Kernel:
  - V5.2
  - V5.1
  - V4.5
  - V4.4
  - V4.3
  - V4.2
  - V4.1
  - V4.0
  - V3.2
- The relevant version of the following SIMOTION Technology Packages, depending on the kernel:
  - Cam
  - Path (as of kernel V4.1)
  - Cam_ext
  - TControl

Sections in this manual

The following is a list of sections included in this manual along with a description of the information presented in each section.

- **Fundamental safety instructions** (Section 1)
- **Overview of the functionality of modular machines** (Section 2)
  This section provides an overview of the functionalities for modular machines that are automated using SIMOTION and PROFIBUS DP or PROFINET IO.
- **Synchronizing SIMOTION devices with a higher-level bus block cycle** (Section 3)
  This section describes the synchronization of a SIMOTION device and its PROFIBUS DP interfaces with a higher-level bus block cycle.
- **Setting the communication addresses via the user program** (Section 4)
  This section describes the options for changing the communication addresses for PROFIBUS DP, PROFINET IO, and Ethernet via the user program.
- **Activating and deactivating components and technology objects** (Section 5)
  This section describes how to activate or deactivate DP slaves (in PROFIBUS), IO devices (in PROFINET), or technology objects in the user program.
- **Changing the active configuration or the active kernel** (Section 6)
  This section describes how to perform a controlled change of configurations or the SIMOTION Kernel via a user program.
- **Appendix**
  u7mkcnfx.exe Command line application, creating and loading the SCOUT scripting card images, storing configuration files for SIMOTION D4xx devices up to kernel version V4.1.4
- **Index**
  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P

- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.1.1 Fundamental safety instructions

4.1.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

4.1.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

4.1.1.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.1.2 Overview of the functionality of modular machines

This section provides an overview of which functionalities are offered by the SIMOTION MC system to enable the creation of modular machines with SINAMICS drives, PROFIBUS DP, and PROFINET IO.

The SIMOTION system offers several ways of designing a machine on a modular basis and optimizing the rated conditions:

- The targeted and automated creation of a SIMOTION project in the SIMOTION SCOUT engineering system, for example by:
  - Using ready-made and tested modules such as library components, for example
  - Automating the configuration and parameterization of machine components (e.g. the drives or the components) via so-called scripting.
    This procedure is clearly described in the online help section of the SCOUT engineering system and will not be further explained here.
  - Changing the configuring via the user-friendly SIMOTION easyProject project generator.
    As of version V4.4, it can be called directly from SIMOTION SCOUT.
    Further information is available at:
    On the SIMOTION Utilities & Applications DVD that is supplied free of charge with SIMOTION SCOUT.
    On the Internet (http://w3.siemens.com/mcms/mc-systems/en/automation-systems/mc-system-simotion/motion-control-software/easy-project/Pages/simotion-easy-project.aspx).
    In SIOS (https://support.industry.siemens.com/cs/document/62049135/project-generator-simotion-easyproject?dti=0&lc=en-WW).
- Changing the configuration during user program runtime. This is done via system functions provided by the SIMOTION kernel. An engineering system is not required.
  This procedure is described in detail in this document.
  The procedure for distributed synchronous operation, which can apply to the entire project, is described in the function manual titled *SIMOTION Motion Control Technology Objects Synchronous Operation, Cam*.

4.1.2.1 Synchronizing SIMOTION devices with a higher-level bus cycle clock

SIMOTION devices provide several communication interfaces to which distributed I/O, for example, or other SIMOTION devices can be connected. This enables the synchronous running of axes that are controlled and regulated via the various SIMOTION devices (distributed synchronous operation). For this, the devices must be interconnected via an isochronous communication network (PROFIBUS DP or PROFINET IO) and synchronized with a common bus cycle clock.

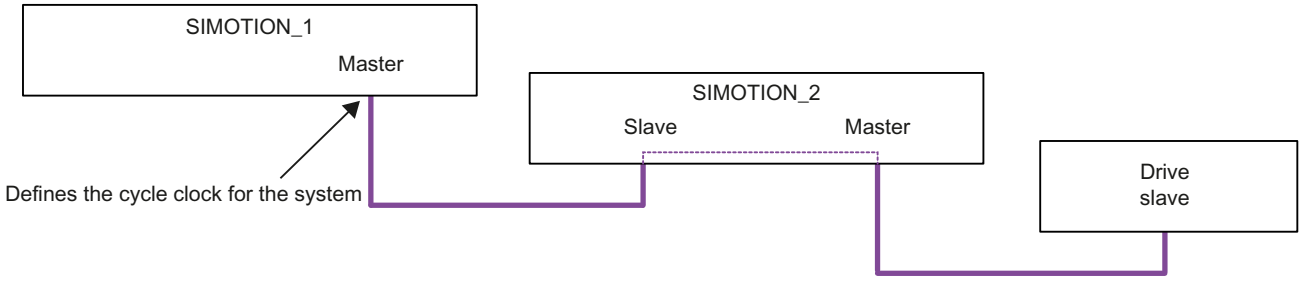


Figure 4-1 Sample configuration for synchronizing a SIMOTION device with an isochronous master

The SIMOTION_1 device delivers a bus cycle clock to the connected subnet via an isochronous communication interface that has been configured as a DP master (in PROFIBUS DP) or as a sync master (in PROFINET IO).

The SIMOTION_2 device should be synchronized with this higher-level bus cycle clock. For this reason, the communication interface reached by the bus cycle clock in the SIMOTION_1 device must be configured as a DP slave (in PROFIBUS DP) or as a sync slave (in PROFINET IO). General information about synchronizing a SIMOTION device with the bus cycle clock (Page 1000) provides an overview of cycle clock generation and synchronization in PROFIBUS DP and PROFINET IO.

Isochronous DP master interfaces in the SIMOTION_2 device can be synchronized with the higher-level bus cycle clock and the cycle clock delivered to lower-level drives. The synchronization behavior of the SIMOTION_2 device depends on whether or not the PROFIBUS DP interfaces have been configured as isochronous DP masters:

- No further isochronous DP master interface available: The synchronization of the device (with the internal cycle clocks) occurs automatically when the higher-level bus cycle clock has been recognized at the corresponding slave interface. See Synchronizing a SIMOTION device without an isochronous DP master interface (Page 1006)
- At least one isochronous DP master interface is available (for example in SIMOTION D). Synchronization of the isochronous DP master interfaces is controlled by the user and occurs when the user calls up a system function. Synchronization of the device varies, depending on whether the higher-level bus cycle clock is fed in via a PROFIBUS DP interface or a PROFINET IO interface. See Synchronization of a SIMOTION device with an isochronous DP master interface (Page 1008)

4.1.2.2 Changing communication addresses via the user program

Nodes are clearly identified within a communication network by means of, for example

- The PROFIBUS address of the interface in PROFIBUS DP
- The device name (NameOfStation) in PROFINET IO
- The IP address of the interface on Ethernet

These communication addresses are set during configuration with the help of the SIMOTION SCOUT engineering software. They can be changed later by a user program and the requirements customized at each respective destination.

Some application examples for this functionality are described in the following.

Application examples for setting the communication address

Base machine with several modules

A machine consists of a base machine (base_machine) with several machine modules (module_1 to module_n). The base machine and all modules are controlled via separate SIMOTION devices. These are interconnected via a communication network. This is shown schematically:

- In the PROFIBUS DP example: A base machine with several identical modules, which communicate via PROFIBUS DP
- In the PROFINET IO example: A base machine with several identical modules, which communicate via PROFINET IO

The same SIMOTION project (i.e. the same SIMOTION devices, drives, technology objects, user programs) is loaded on all machine modules. The user program of each module determines the necessary communication address and sets it up.

Note

As an alternative to changing the communication address of the machine module, you can also activate different configurations (Activating the configuration (Page 998)).

Example of communication via PROFIBUS DP

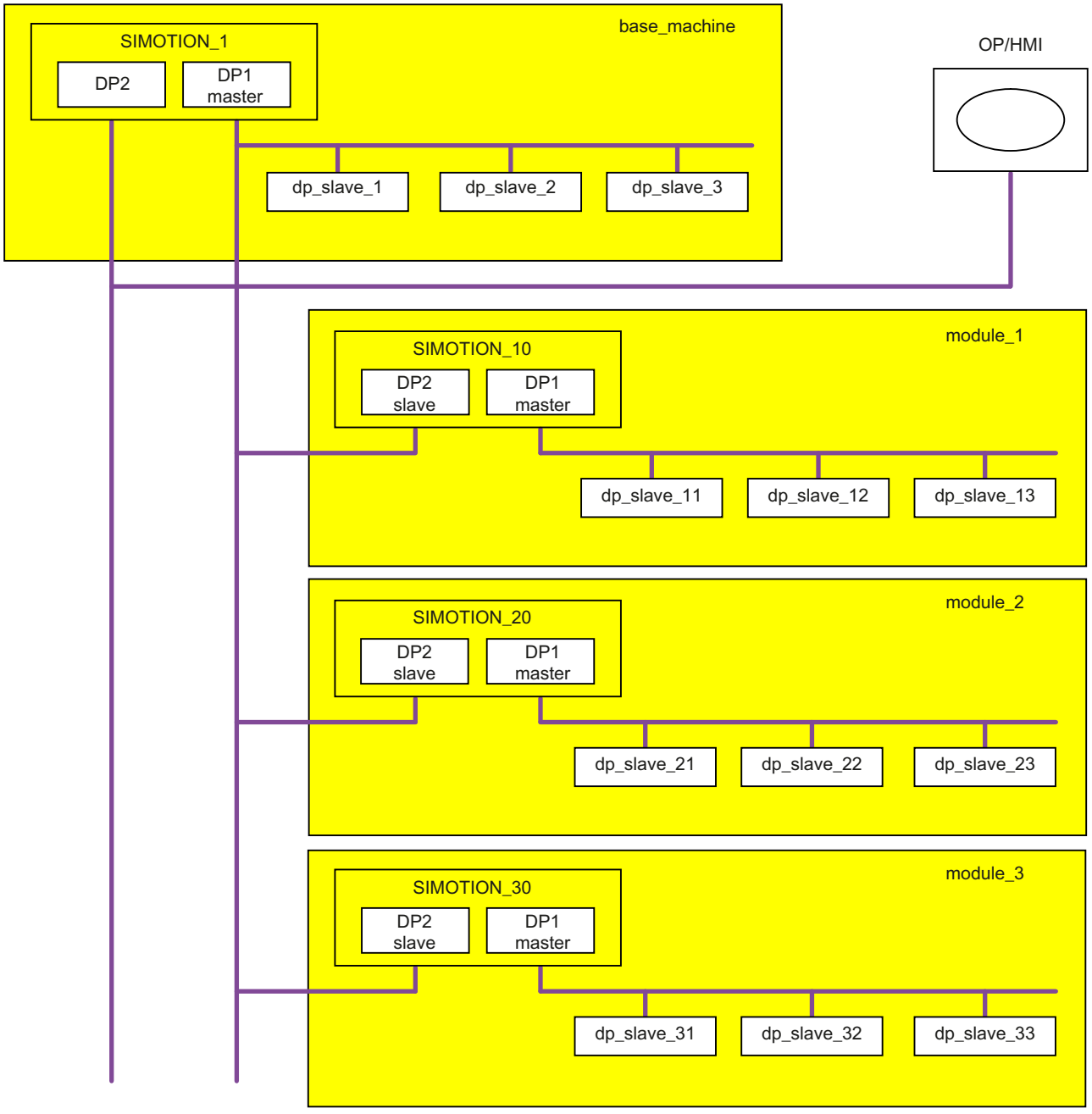


Figure 4-2 Base machine with several identical modules (communication via PROFIBUS DP)

Example of communication via PROFINET IO

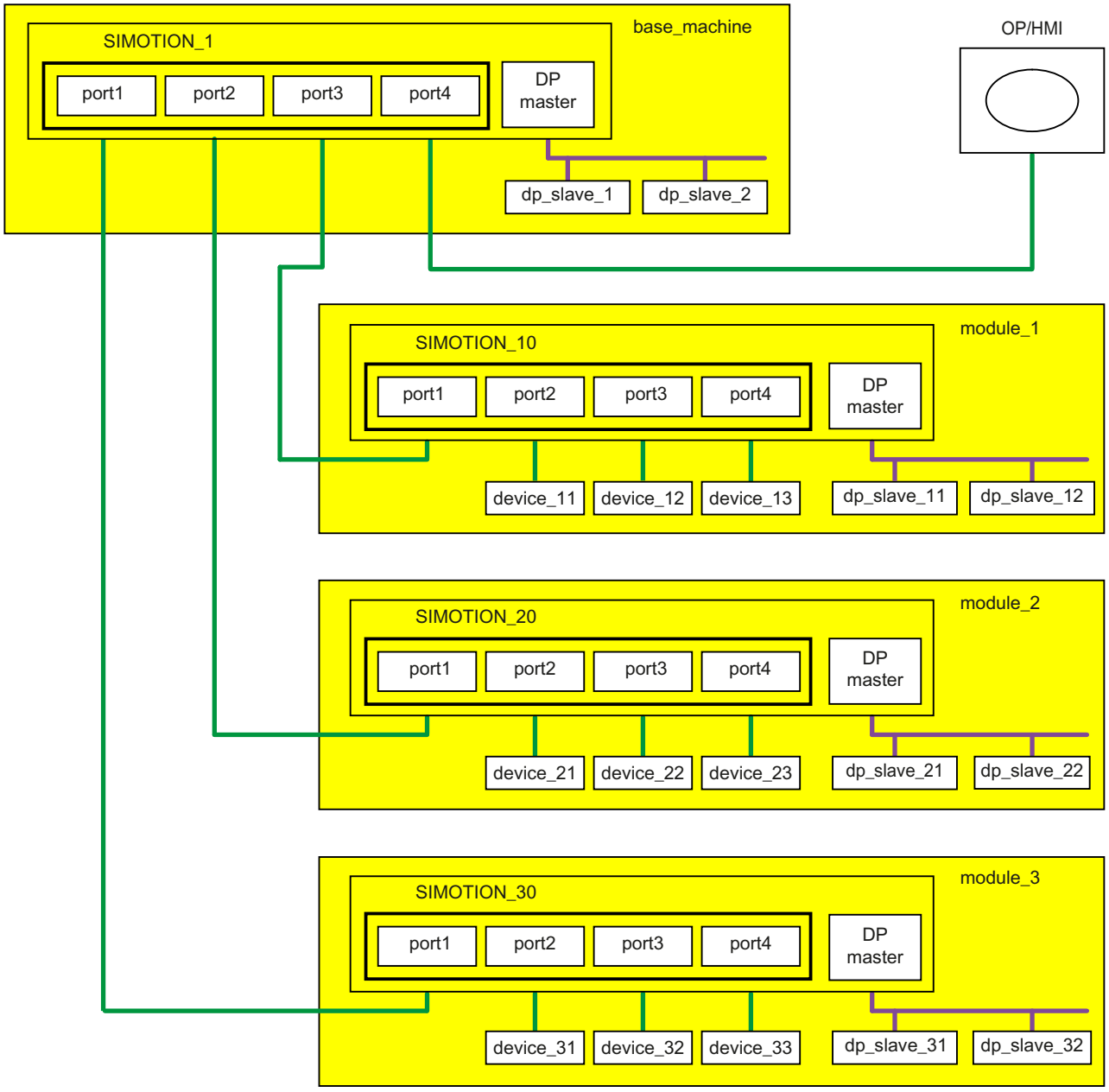


Figure 4-3 Base machine with several identical modules (communication via PROFINET IO)

Replacing a module on the base machine

The user connects a replacement module to the base machine. A standard project with a preset communication address is loaded on the replacement module. The user program of each module determines the necessary communication address and sets it up.

Operating a module on several base machines

A module (with dedicated SIMOTION control) is alternately connected to different base machines (e.g. a transport vehicle travels to different assembly sites and connects there). At each base machine, the module is to be integrated in the respective communication network and contact established with the higher-level control.

A standard project with a preset communication address is loaded on the module. The communication address valid for the network of the respective base machine is set by the user program of the module according to the specifications.

User program for changing the communication address

The user program checks which communication address is to be set for the respective network, e.g. by:

- Reading in and evaluating a slot coding
- Evaluating a program variable (which can be changed, for example, by a user input on the HMI device)

The determined communication address then sets the user program. System functions are provided for this purpose, for example:

- To change the PROFIBUS address of a DP slave (see Setting the PROFIBUS address (Page 1014)):
  - _getActiveDpSlaveAddress
  - _setDpSlaveAddress
  - _activateDpSlaveAddress
- To change the device name (NameOfStation) of an IO device on PROFINET IO (see Setting the device name (NameOfStation) of an IO device on PROFINET IO (Page 1020)):
  As of SIMOTION Kernel version V4.4, the following functions should be used in new projects:
  - _getPnNameOfStation (as of Kernel V4.4, replaces _getActiveNameOfStation)
  - _setPnNameOfStation (as of Kernel V4.4, _setNameOfStation _activateNameOfStation)

  Up to SIMOTION Kernel version V4.3, the following functions should be used:
  - _getActiveNameOfStation
  - _setNameOfStation
  - _activateNameOfStation
- To change the IP address of an Ethernet node (see Setting the IP address (on Ethernet) (Page 1026)):
  As of SIMOTION Kernel version V4.4, the following functions should be used in new projects:
  - _getPnIpConfig (as of Kernel V4.4, replaces _getIpConfig)
  - _setPnIpConfig (as of Kernel V4.4, replaces _setIpConfig)

  Up to SIMOTION Kernel version V4.3, the following functions should be used:
  - _getIpConfig
  - _setIpConfig

4.1.2.3 Operating only parts of a configured maximum configuration

Here, for example, a machine consists of a base machine with several machine modules or several axes. The user configures and programs the base machine with all possible machine modules and axes (maximum configuration). The real machine differs from the configured maximum configuration, e.g.:

- Not all configured machine modules or axes are available (actual structure differs from configured structure)
- Not all machine modules or axes are required in an operation phase.

SIMOTION provides system functions for the following purposes:

- The distributed I/O on PROFIBUS DP or PROFINET IO (e.g. drives or other DP slaves or IO devices) can be activated or deactivated. In addition, the current state of individual or all DP slaves or IO devices can be queried.
- Technology objects (e.g. axes assigned to drives that are not required) can be activated or deactivated.

The computational load of the SIMOTION device can be reduced through specific use of these system functions; the computing capability can be concentrated on the drives or technology objects which are currently required. No bus fault is signaled, even if bus nodes are not available.

Note

Licenses are required for all axes of the maximum configuration.

Fundamental procedures

As an example, different procedures are described for a machine during ramp-up or during production.

- During ramp-up, the user program determines the actual structure of the distributed I/O (e.g. drives, other DP slaves) of the machine. Here, the system function calls _getStateOfAllDpSlaves and evaluates its return value.
  All configured distributed I/O are activated, by default. The user program now deactivates the DP slaves that are not available and the technology objects assigned to the drives.
- During a defined production phase, certain plant units (e.g. machine modules, including drives and axes) are required, while others are not. In accordance with the relevant operator specifications, the user program deactivates plant units which are no longer required (along with the assigned technology objects) and activates the newly added ones.

User program

You can use the following SIMOTION system functionalities in the user program:

- System functions are provided (see Activating and deactivating nodes on the PROFIBUS or PROFINET IO (Page 1031)) to activate or deactivate a bus node (DP slaves or IO devices) and to call up its status; for example:
  - _activateDpSlave
  - _deactivateDpSlave
  - _getStateOfAllDpSlaves or _getStateOfAllDPStations
- Use the parameters of a SINAMICS drive (see Activating and deactivating SINAMICS components (Page 1064)) to activate or deactivate the drive and to query its status.
- System functions are provided (see Activating and deactivating technology objects (Page 1066)) to activate or deactivate a technology object and to query its status; for example:
  - _activateTo
  - _deactivateTo
  - _getStateOfTo

Note

The functions can also be integrated easily into the user program via the library **LMoMa**. This library is included on the DVD "SIMOTION Utilities & Applications". This DVD also contains other useful blocks such as those required to read out the DriveCLiQ topology, for example.

System behavior

The cycle time calculated from the configuration for exchanging data with all distributed I/O on the PROFIBUS DP remains unchanged.

However, a deactivation of the distributed I/O still results in the following advantages:

- The data volume on the PROFIBUS DP to be transferred cyclically is reduced, and the time required within a system cycle clock for communication decreases. The time saved is available for other services on the PROFIBUS (e.g. acyclic services for operator control and monitoring or diagnostics)
- Deactivated technology objects hardly require any computing time. The time saved is available for user tasks
- Alarms of these I/O (e.g. _SC_DIAGNOSTIC_INTERRUPT, _SC_STATION_DISCONNECTED, _SC_STATION_RECONNECTED) are suppressed and no longer evaluated.

4.1.2.4 Activating the configuration or the SIMOTION kernel

The "configuration server" functionality integrated in SIMOTION permits configurations on the memory card to be added to the SIMOTION device. For example, the following applications can be implemented:

- A configuration adapted for each task (e.g. device configuration, technology objects, user program) can be added (see Activating the configuration (Page 998)):
  - By coupling different machine modules (e.g. in different production phases), the required configuration can be automatically determined (using connector coding, for example); this configuration can then be added
  - The memory and computing capability of the SIMOTION device are only used for the configured functionalities
  - Cycle times and cycle clocks (e.g. DP cycle clock, position control cycle clock, IPO cycle clock) can be optimized for each requirement.
- The configurations (e.g. of the user programs) or the SIMOTION kernel can be updated using the user program:
  The user transfers the new configurations and, if necessary, the SIMOTION kernel to the memory card of the SIMOTION device and starts updating via the user program
  (see Activating the configuration (Page 998) and Activating the SIMOTION kernel (Page 999))
- Automatic recognition of the required configuration and its activation during the ramp-up of a SIMOTION device (SelfAdaptingConfiguration) is possible (see Selecting and activating a configuration using an initial configuration (Page 999)).

Activating the configuration

Various machine modules (e.g. DP slaves) are alternately connected to a SIMOTION device (base machine). The data for each arrangement (base machine with connected machine module) is stored as its own configuration on the memory card of the SIMOTION device.

In the example (see figure), a transport device with a SIMOTION control transports a workpiece to various machining stations. The following sequences can be observed here:

1. Drawing a blank from the store and transporting it to the machining station
2. Machining the blank
3. Unloading the semi-finished part at the packing station.

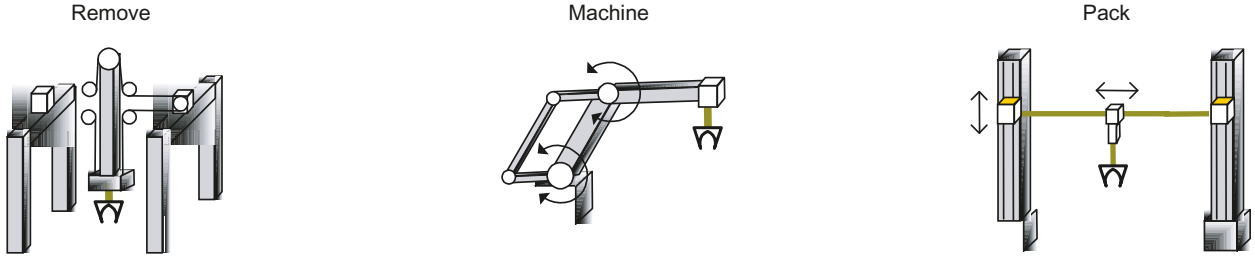


Figure 4-4 Sequences of an automation task

For each machining step, the associated optimized configuration is stored on the memory card of the transport device. After coupling to a machining station, the transport device identifies this, e.g. by evaluating a connector coding, and loads the required configuration using the _activateConfiguration system function. It then completes all work at the station and moves to the next one.

See also Activating a configuration (Page 1078).

Activating the SIMOTION kernel

You can also exchange the active kernel of the SIMOTION device using the user program and, for example, carry out an update to a new version without having to change the memory card of the device. In this regard, you must replace all configurations (configuration data) of the SIMOTION device.

The SIMOTION kernel and the associated configurations (configuration data) are stored on the memory card.

Using the _activateConfiguration system function you can ensure that a new kernel and a configuration are loaded and activated in the user program. The activating configuration can be specified explicitly or determined automatically using an initial configuration.

See also Activating a kernel (Page 1084).

Selecting and activating a configuration using an initial configuration

A SIMOTION device (machine module) is connected to a different SIMOTION device (base machine). After POWER ON, the machine module ramps up, identifies its position at the base machine (e.g. using a connector coding), and automatically loads the required configuration.

For this, the data of all configurations is saved to the memory card of the machine module; the machine module can then pick up this data on the base machine. The initial configuration also stored there will be loaded automatically after POWER ON. This identifies the required configuration (e.g. by using a connector coding) and loads it using the _activateConfiguration system function.

See also Selecting and activating a configuration using an initial configuration (Page 1089).

Procedure

All configurations required for a SIMOTION device (and possibly the SIMOTION kernel to be added) must be stored on the memory card of the SIMOTION device in a prescribed data format. The following section gives a summary of how to create the required data.

1. Configure and program each required configuration as a unique device in the engineering system (e.g. SIMOTION SCOUT).
2. Create the files required for the file system on the memory card of the SIMOTION device.
3. Transfer this card image to the associated memory card.

This process is described in detail in Procedure for creating the card images of configurations (Page 1098).

4.1.3 Synchronizing SIMOTION devices with a higher-level bus cycle clock

4.1.3.1 General information about synchronizing a SIMOTION device with the bus cycle clock

SIMOTION devices provide a range of interfaces for connecting to PROFIBUS DP or PROFINET IO. The devices and their applications (e.g. in ServoSynchronousTask or IPOSynchronousTask) can be operated isochronously to the bus cycle clocks occurring on the interfaces.

The table below provides an overview of the number and type of PROFIBUS DP and PROFINET IO interfaces of the various SIMOTION devices.

Table 4-1 Isochronous PROFIBUS DP and PROFINET IO interfaces of SIMOTION devices

SIMOTION device	PROFIBUS DP interfaces	PROFINET IO interfaces
D410 DP	• 1 interface • 1 internal master interface for SINAMICS Integrated	0 interfaces
D410 PN	• 0 interfaces • 1 internal master interface for SINAMICS Integrated	1 interface (2 ports)
D410-2 DP	• 2 interfaces • 1 internal master interface for SINAMICS Integrated	0 interfaces
D410-2 DP/PN	• 1 interface • 1 internal master interface for SINAMICS Integrated	1 interface (2 ports)
D425 D435 D445 D445-1	• 2 interfaces • 1 internal master interface for SINAMICS Integrated	1 interface optional (CBE30, 4 ports)
D425-2 DP D435-2 DP	• 2 interfaces • 1 internal master interface for SINAMICS Integrated	0 interfaces
D425-2 DP/PN D435-2 DP/PN D445-2 DP/PN D455-2 DP/PN	• 2 interfaces • 1 internal master interface for SINAMICS Integrated	• 1 interface (3 ports) • 1 interface optional (CBE30-2, 4 ports)
C230-2 C240	2 interfaces	0 interfaces
C240 PN	2 interfaces	1 interface (3 ports)
P320-3	0 interfaces	1 interface (3 ports)
P320-4	2 interfaces optional (IsoPROFIBUS board)	1 interface (3 ports)
P350-3	2 interfaces	1 interface optional (MCI-PN board, 4 ports)

The PROFIBUS DP interfaces can be operated as either slave or master, and the PROFINET IO interfaces as either controller and/or device.

The following provides a description of cycle clock generation and the various synchronization mechanisms.

Cycle clock generation and synchronization in PROFIBUS DP

Clock generation in PROFIBUS DP

It is necessary to distinguish between the following cases for clock generation in PROFIBUS DP:

- Precisely one PROFIBUS DP interface is configured as DP slave and is operated isochronously: In this case, this interface supplies the basic cycle clock for the SIMOTION device.
  - If a DP cycle is available at this interface, the SIMOTION device must be synchronized with this cycle clock. The synchronization behavior depends on whether the PROFIBUS DP interfaces of the SIMOTION device have been configured as the isochronous DP master. See the following section, "Synchronization in PROFIBUS DP".
  - If no DP cycle clock is available at this interface (e.g. if the associated DP master fails), a replacement cycle clock of the same size is generated.
- In all other cases (e.g. if no PROFIBUS DP interface has been configured as DP slave, or the interface configured as DP slave is not operated isochronously), the following applies: The basic cycle clock is derived from an internal cycle clock which is determined during configuration (configured basic cycle clock).

The figure below displays the cycle clock generating principle in PROFIBUS DP.

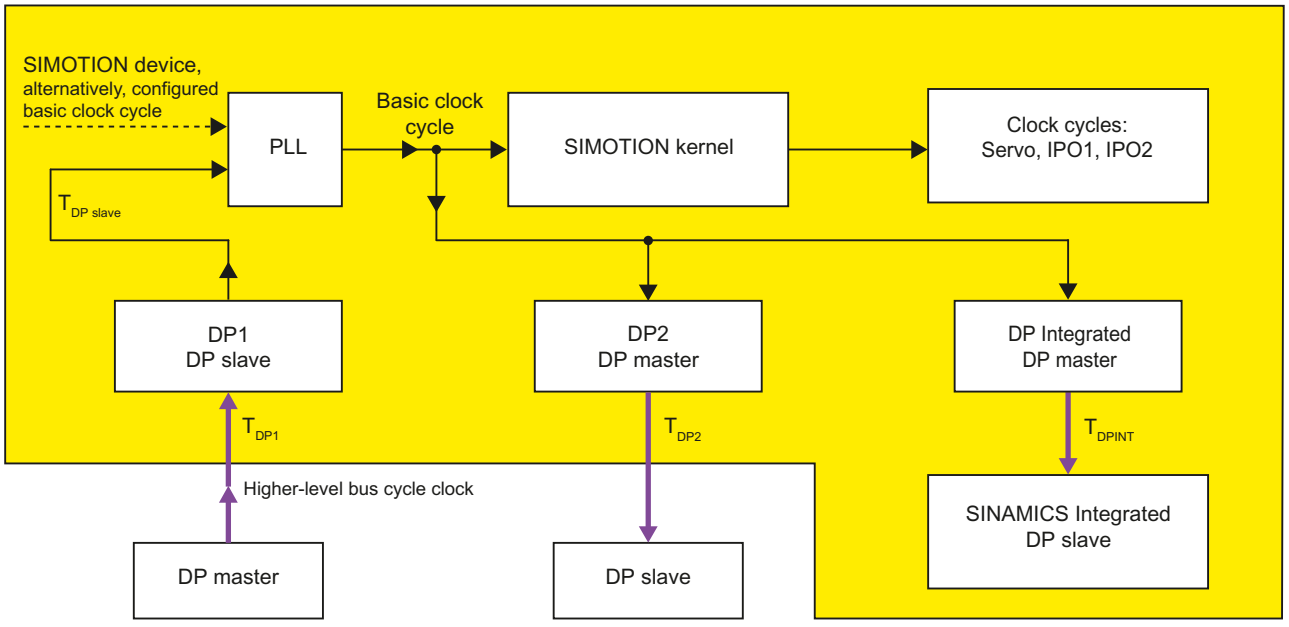


Figure 4-5 Cycle clock generation in a SIMOTION device in PROFIBUS DP

Synchronization in PROFIBUS DP

If the DP slave interface is operated isochronously, the SIMOTION device must be synchronized with the higher-level bus cycle clock. The synchronization behavior depends on whether the PROFIBUS DP interfaces of the SIMOTION device have been configured as the isochronous DP master:

- None of the PROFIBUS DP interfaces have been configured as isochronous DP master for the lower-level drives:
  The device is automatically synchronized with the higher-level bus cycle clock.
  After successful synchronization, the cycle clocks of the device (e.g. Servo, IPO, IPO2) and the programs running in the SynchronousTasks are both synchronous with the higher-level bus cycle clock.
  See: Synchronization of a SIMOTION device without isochronous DP master interface (Page 1006).
- At least one of the PROFIBUS DP interfaces is configured as isochronous DP master for lower-level drives:
  Users must themselves activate synchronization of the device with the higher-level bus cycle clock using the **_synchronizeDpInterfaces** system function (user-controlled synchronization). The user can thereby determine the time of synchronization and take suitable precautions, such as deactivating affected DP slaves, to avoid system failures caused by the synchronization process.
  Only after successful synchronization are the device cycle clocks (e.g. Servo, IPO, IPO2), the programs running in SynchronousTasks, and also the isochronous DP master interfaces all synchronous with the higher-level bus cycle clock.
  See: Synchronization of a SIMOTION device with isochronous DP master interface (Page 1008).

Note

Synchronization without the configured Safety functions

A phase jump occurs during synchronization of an isochronous DP master interface with a higher-level bus cycle clock. This can lead to failures in the connected DP slaves (e.g. SINAMICS drives).

You should, therefore, deactivate the DP slaves concerned and the technology objects assigned to them before synchronizing drives that are already in operation. Once synchronization has been successfully completed, you can reactivate the DP slaves concerned and their assigned technology objects. See Activating and deactivating components and technology objects (Page 1031).

Note

In SIMOTION D devices, the DP Integrated interface is always configured as the isochronous DP master interface.

Therefore, if these devices are connected to a higher-level isochronous master, they must always be synchronized.

Note

Synchronization with the configured Safety functions

Note that if SINAMICS drives with Safety Integrated are connected to the isochronous DP master interface of a SIMOTION device and the actuation is performed via PROFIsafe:

A Safety error in the drive which cannot be acknowledged can be triggered during the synchronization of the SIMOTION device with the higher-level bus cycle clock. This error occurs if the phase jump caused by the synchronization occurs while the Safety functionality in the drive is booting up.

Therefore, the synchronization must not be performed until after Safety Integrated in the drive is ready. In order to achieve this, use the user-controlled synchronization, see synchronization of a SIMOTION device with isochronous DP master interface (Page 1008). Wait with the call of the system function **_synchronizeDPInterfaces** until:

1. The drives have powered up (system variable **cyclicInterface** of the axis = ACTIVE).
   And
2. Safety Integrated in the drives is ready (additional 250 ms = 10 * MAX (p9500)).

Cycle clock generation and synchronization in PROFINET IO

Clock generation in PROFINET IO

The following applies for clock generation in PROFINET IO:

If a PROFINET IO interface is available, the base cycle clock of a SIMOTION device is supplied by this interface providing the interface has been configured as follows:

1. The synchronization type has been set as sync master or sync slave.
2. IRT with the "High Performance" option has been set as the real-time class.

In this configuration, the basic cycle clock is the send cycle clock set at the associated sync domain.

It is necessary to distinguish between the following cases:

- The PROFINET IO interface is configured as a Sync-Master:
  This interface provides the cycle clock for the SIMOTION device including its isochronous PROFIBUS DP interfaces. Synchronization with a higher-level bus cycle clock is not necessary.
- The PROFINET IO interface is configured as a Sync-Slave:
  In this case, this interface supplies the basic cycle clock for the SIMOTION device.
  - If a cycle clock is available at this interface, the SIMOTION device automatically synchronizes itself with this cycle clock.
    If PROFIBUS DP interfaces of the SIMOTION device have been configured as isochronous DP master, their synchronization must be initiated by the user. See the following section, "Synchronization in PROFINET IO".
  - If no cycle clock is available at this interface (e.g. if the associated sync master fails), a replacement cycle clock of the same size is generated.

Note

Only a single cycle clock source is permitted on a SIMOTION device. For this reason:

If a SIMOTION device contains a PROFINET IO interface, a PROFIBUS DP interface can no longer be operated as an isochronous DP slave.

The following figure shows cycle clock generation in PROFINET IO:

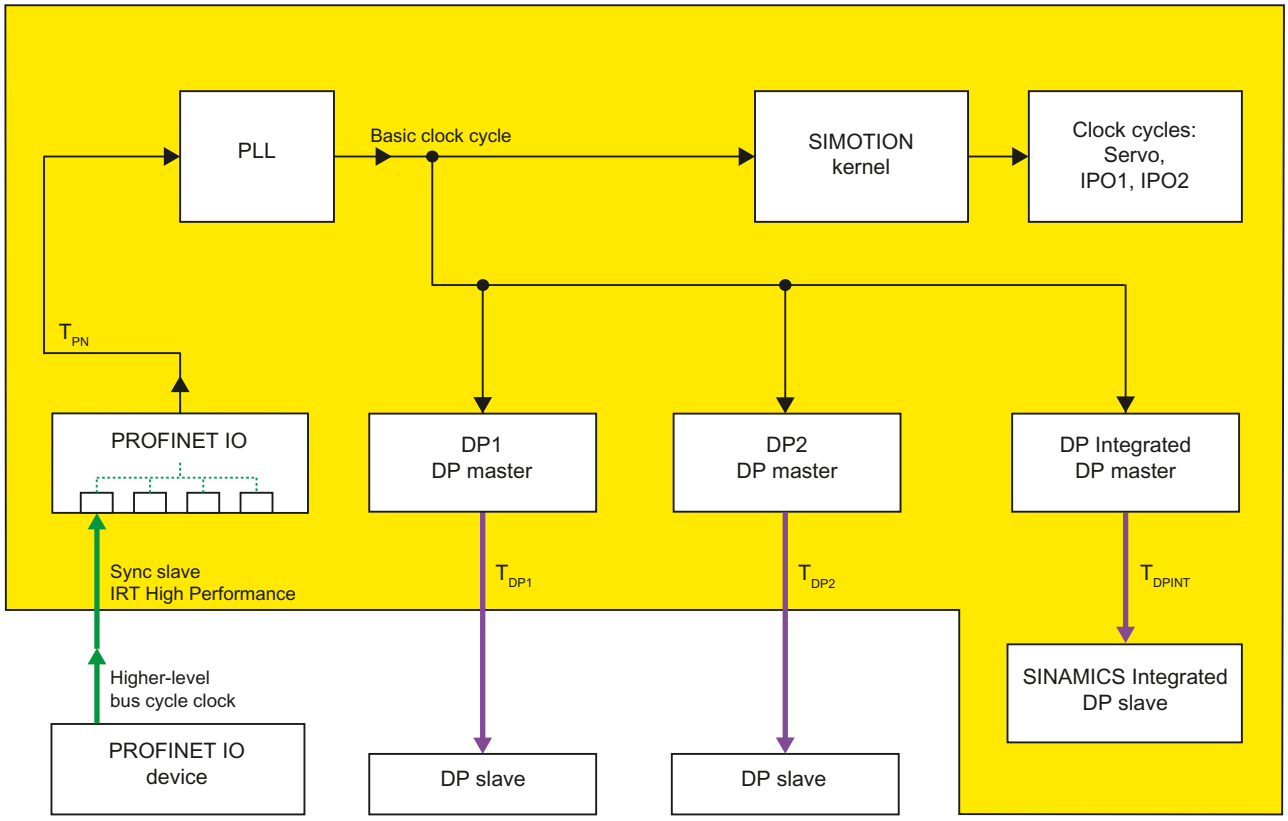


Figure 4-6 Cycle clock generation in a SIMOTION device in PROFINET IO

Synchronization in PROFINET IO

If the PROFINET IO interface has been configured as a Sync-Slave, the SIMOTION device must be synchronized with the higher-level bus cycle clock. The synchronization behavior depends on whether the PROFIBUS DP interfaces of the SIMOTION device have been configured as isochronous DP master:

- None of the PROFIBUS DP interfaces have been configured as isochronous DP master for the lower-level drives:
  The device is automatically synchronized with the higher-level bus cycle clock.
  After successful synchronization, the cycle clocks of the device (e.g. Servo, IPO, IPO2) and the programs running in the SynchronousTasks are both synchronous with the higher-level bus cycle clock.
  See Synchronizing a SIMOTION device without an isochronous DP master interface (Page 1006).
- At least one of the PROFIBUS DP interfaces is configured as isochronous DP master for lower-level drives:
  Synchronization of the device and the isochronous DP master interfaces is different:
  - The device (not including the DP master interfaces) is automatically synchronized with the higher-level bus cycle clock.
    After successful synchronization, the cycle clocks of the device (e.g. Servo, IPO, IPO2) and the programs running in the SynchronousTasks are both synchronous with the higher-level bus cycle clock.
  - Synchronization of the isochronous DP master interfaces with the higher-level bus cycle clock must be activated by the users using the **_synchronizeDpInterfaces** system function. The user can thereby determine the time of synchronization and take suitable precautions, such as deactivating affected DP slaves, to avoid system failures caused by the synchronization process.
    Only after successful synchronization has been completed are the isochronous DP master interfaces synchronous with the higher-level bus cycle clock.
    See Synchronizing a SIMOTION device with an isochronous DP master interface (Page 1008).

Note

A phase jump occurs during synchronization of an isochronous DP master interface with a higher-level bus cycle clock. This can lead to failures in the connected DP slaves (e.g. SINAMICS drives).

You should, therefore, deactivate the DP slaves concerned and the technology objects assigned to them before synchronizing drives that are already in operation. Once synchronization has been successfully completed, you can reactivate the DP slaves concerned and their assigned technology objects. See Activating and deactivating components and technology objects (Page 1031).

Note

In SIMOTION D devices, the DP Integrated interface is always configured as the isochronous DP master interface.

Therefore, if these devices are connected to a higher-level isochronous master, they must always be synchronized.

Note

Note that if SINAMICS drives with Safety Integrated are connected to the isochronous DP master interface of a SIMOTION device and the actuation is performed via PROFIsafe:

A safety error in the drive which cannot be acknowledged can be triggered during the synchronization of the SIMOTION device with the higher-level bus cycle clock. This error occurs if the phase jump caused by the synchronization occurs while the safety functionality in the drive is booting up.

Therefore, the synchronization must not be performed until after Safety Integrated in the drive is ready. In order to achieve this, use the user-controlled synchronization, see synchronization of a SIMOTION device with isochronous DP master interface (Page 1008). Wait with the call of the system function **_synchronizeDPInterfaces** until:

1. The drives have powered up (system variable **cyclicInterface** of the axis = ACTIVE).
   And
2. Safety Integrated in the drives is ready (additional 250 ms = 10 * MAX (p9500)).

4.1.3.2 Synchronizing a SIMOTION device without an isochronous DP master interface

Prerequisites

1. None of the PROFIBUS interfaces in the SIMOTION device have been configured as an isochronous DP master for lower-level drives:
   This is the case e.g. in SIMOTION C230-2, C240 or P350 devices if no lower-level isochronous drives are connected via PROFIBUS DP.
2. An interface receives a higher-level bus cycle clock and is configured accordingly, i.e. as DP slave in PROFIBUS DP or as sync slave in PROFINET IO.

Example configuration

The following example configuration is used:

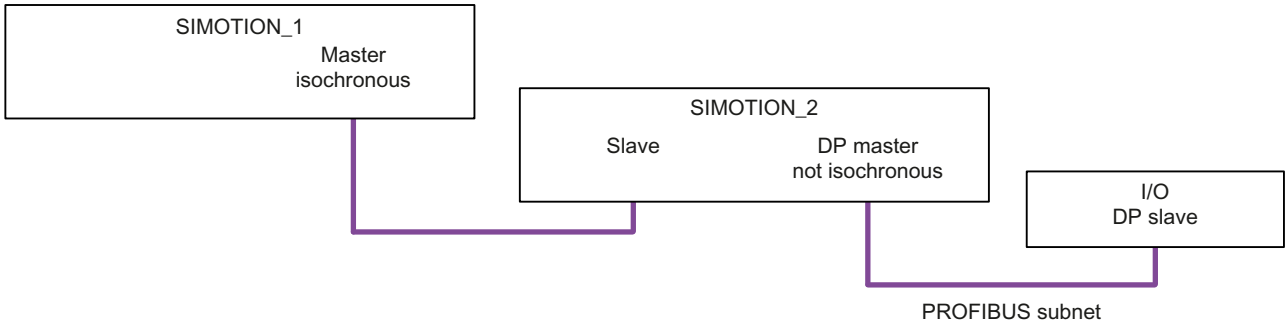


Figure 4-7 Example configuration for synchronizing a SIMOTION device with the higher-level bus cycle clock without an isochronous DP master interface

Synchronization sequence

As soon as a higher-level bus cycle clock is available at the corresponding interface (DP slave in PROFIBUS DP, sync slave in PROFINET IO) of the SIMOTION_2 device, the interface automatically synchronizes itself with the defined bus cycle clock.

Note

Only a single cycle clock source is permitted on a SIMOTION device. For this reason:

If a SIMOTION device contains a PROFINET IO interface, a PROFIBUS DP interface can no longer be operated as an isochronous DP slave.

After successful synchronization, the device cycle clocks (e.g. Servo, IPO, IPO2) and the programs running in the SynchronousTasks are both synchronized with the received bus cycle clock.

Should the bus cycle clock fail, a replacement cycle clock will be generated with the same cycle time.

The current synchronization state is shown in the **stateOfDpSlaveSynchronization** system variable. You can, for example, evaluate the synchronization state in the user program so that in the event of a synchronization failure, execution of program sections that assume synchronization would be prevented.

In addition, you can activate the alarms for initiating the PeripheralFaultTask. To do this, call up the **_enableDpInterfaceSynchronizationMode** system function using the parameter dpInterfaceSyncMode = SLAVE_ALARMMESSAGES_1:

- The _SC_DP_SLAVE_SYNCHRONIZED (= 209) and _SC_DP_SLAVE_NOT_SYNCHRONIZED (= 210) events then initiate the PeripheralFaultTask and can be queried in their TaskStartInfo (TSI#InterruptId).
- The **modeOfDpInterfaceSynchronization** system variable is set to SLAVE_ALARMMESSAGES_1.

See also the status diagram in the figure below.

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

If the alarms for initiating the PeripheralFaultTask are activated, they must be activated in the execution system and an appropriate program assigned. Otherwise the SIMOTION device enters the STOP operating mode when this event occurs.

Status diagram

For a SIMOTION device without an isochronous DP master interface, the status diagram in the figure below describes:

- Both synchronization states
- The values of the corresponding system variables
- The state transitions and the associated initiated alarms (TDI#InterruptId), provided they are activated.

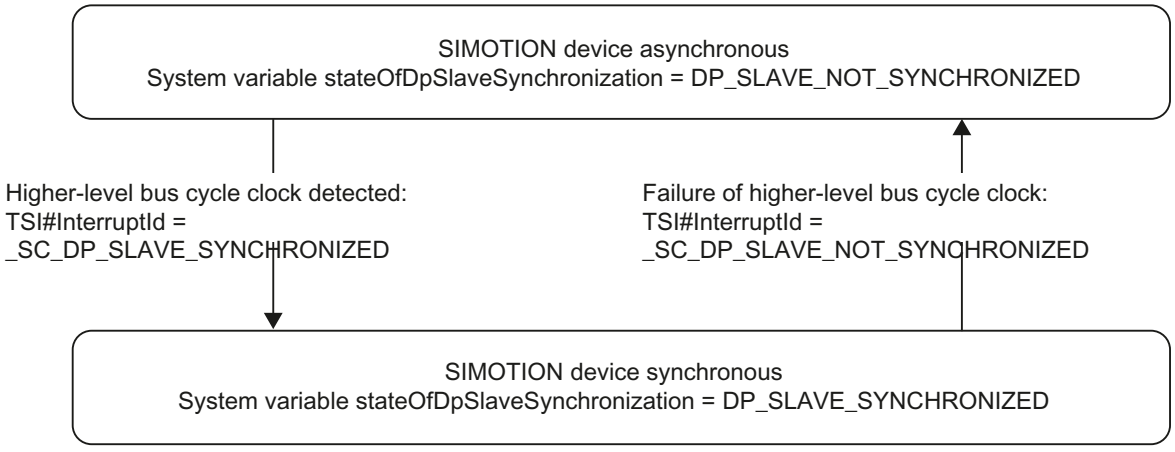


Figure 4-8 Status diagram for synchronization of a SIMOTION device without an isochronous DP master interface

4.1.3.3 Synchronization of a SIMOTION device with an isochronous DP master interface

Prerequisites

1. At least one PROFIBUS interface in the SIMOTION device has been configured as an isochronous DP master.
   This prerequisite is always met with SIMOTION D4xx devices; the interface "DP Integrated" is always configured as an isochronous DP master interface.
2. An interface receives a higher-level bus cycle clock and is configured accordingly, i.e. as a DP slave in PROFIBUS DP or as a sync slave in PROFIBUS IO.
3. The same bus cycle clock must be configured for all isochronously operated interfaces.

Example configuration

The following figures show example configurations for synchronization of a SIMOTION device which has an isochronous DP master interface with a higher-level bus cycle clock. The synchronization behavior depends on whether a PROFIBUS DP subnet or a PROFINET IO subnet supplies the higher-level bus cycle clock.

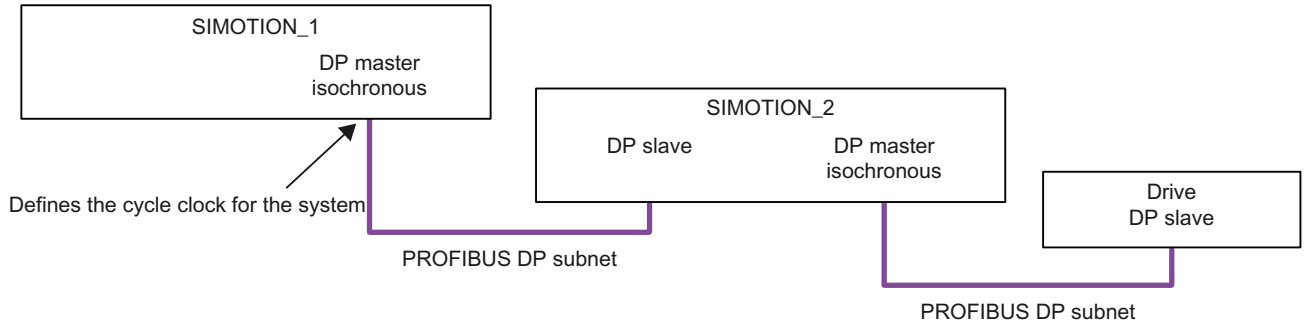


Figure 4-9 Example configuration for synchronization of a SIMOTION device which has an isochronous DP master interface with a higher-level PROFIBUS DP cycle clock

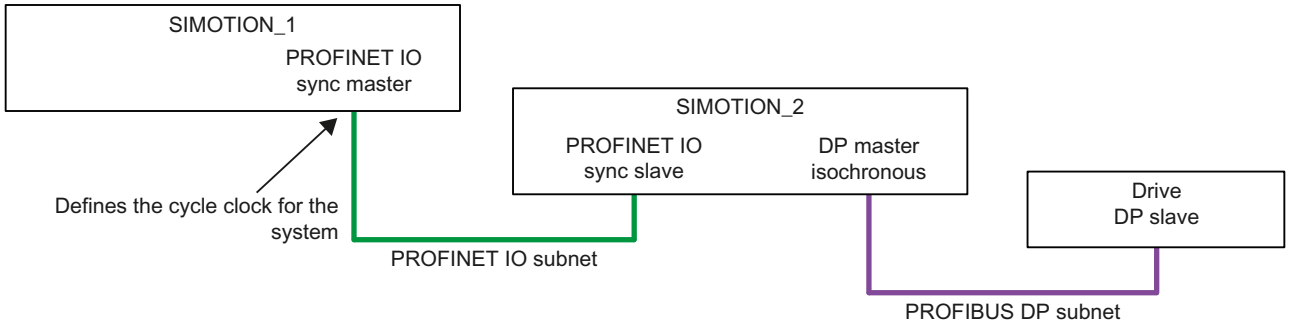


Figure 4-10 Sample configuration for synchronization of a SIMOTION device which has an isochronous DP master interface with a higher-level PROFINET IO cycle clock

Synchronization sequence

Following a configured/programmed system startup, the SIMOTION_2 device with the interface configured as an isochronous DP master (with drives as DP slaves) is to be synchronized with the higher-level bus cycle clock determined by the SIMOTION_1 device.

The interface which receives the higher-level bus cycle clock (DP slave in PROFIBUS DP, sync slave in PROFINET IO) automatically synchronizes itself with the defined bus cycle clock.

Synchronization of the isochronous DP master interface on the SIMOTION_2 device with the higher-level bus cycle clock must be activated by users themselves using a system function. The user can thereby determine the time of synchronization and take suitable precautions, such as deactivating affected DP slaves, to avoid system failures caused by the synchronization process. See following sections: "Procedure for automatic synchronization" and "Procedure for user-controlled synchronization".

Preferably, synchronization should be started while the SIMOTION devices are ramping up (in the StartupTask), before the drives start to move.

Only after successful synchronization has been completed are the isochronous DP master interfaces synchronous with the higher-level bus cycle clock.

For synchronization of the internal cycle clocks of the SIMOTION device and the programs running in the SynchronousTasks, see section: "Synchronization of the cycle clocks of the SIMOTION device".

Synchronization of the cycle clocks of the SIMOTION device

The following differences apply as regards synchronization of the SIMOTION device - which features internal cycle clocks (e.g. Servo, IPO, IPO2) - with the programs running in the SynchronousTasks:

- In PROFIBUS DP:
  Synchronization of the SIMOTION device and synchronization of the isochronous DP master interface run at the same time.
  Therefore, the following applies to both the device cycle clocks (e.g. Servo, IPO, IPO2) and the programs running in SynchronousTasks: Only after successful synchronization of the isochronous DP master interfaces are they synchronous with the higher-level bus cycle clock (see also Cycle clock generation and synchronization in PROFIBUS DP (Page 1001)).
- For PROFINET IO:
  Synchronization of the SIMOTION device with the higher-level bus cycle clock is automatic.
  Therefore, the following applies to both the device cycle clocks (e.g. Servo, IPO, IPO2) and the programs running in SynchronousTasks: They are already synchronous with the received bus cycle clock once successful automatic synchronization of the interface receiving the higher-level bus cycle clock is complete (see also Cycle clock generation and synchronization in PROFINET IO (Page 1003)).

Procedure for automatic synchronization

So that the interface configured as an isochronous DP master in the SIMOTION_2 device automatically synchronizes itself with the higher-level bus cycle clock, you should proceed as follows:

1. If drives (e.g. SINAMICS drives) are already in operation:
   Deactivate the DP slaves concerned and their assigned technology objects (see Activating and deactivating components and technology objects (Page 1031)).
2. Start the automatic synchronization. To do this, call the **_enableDpInterfaceSynchronizationMode** system function using the parameter dpInterfaceSyncMode = AUTOMATIC_INTERFACE_SYNCHRONIZATION.
   - The **modeOfDpInterfaceSynchronization** system variable will be set to AUTOMATIC_INTERFACE_SYNCHRONIZATION.
   - If a higher-level bus cycle clock is recognized at the corresponding interface, the interface configured as an isochronous DP master will be automatically synchronized.
     You can find out about the current synchronization state using the stateOfDpInterfaceSynchronization system variable.
   - The alarm for initiating the PeripheralFaultTask on loss of synchronization is activated.

   The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).
3. Once synchronization has been successfully completed, activate the DP slaves which you previously deactivated and their assigned technology objects (see Activating and deactivating components and technology objects (Page 1031)).
   The synchronized state continues to apply unless it is disrupted, e.g. if the constant bus cycle clock should fail.
   The event _SC_DP_SYNCHRONIZATION_LOST (= 208) initiates the PeripheralFaultTask and can be queried in the associated TaskStartInfo (TSI#InterruptId).

Note

If the alarms for initiating the PeripheralFaultTask are activated, they must be activated in the execution system and an appropriate program assigned. Otherwise the SIMOTION device enters the STOP operating mode when this event occurs.

Note

Preferably, automatic synchronization should be started while the SIMOTION devices are ramping up (in the StartupTask), before the drives start to move.

Procedure for user-controlled synchronization

In order to synchronize the interfaces configured as isochronous DP masters in the SIMOTION_2 device with the higher-level bus cycle clock, you should proceed as follows:

1. First, you must activate the corresponding alarms for initiating the PeripheralFaultTask. To do this, call up the **_enableDpInterfaceSynchronizationMode** system function using the parameter dpInterfaceSyncMode = MASTER_SLAVE_ALARMMESSAGES_1.
   - The events _SC_DP_CLOCK_DETECTED (= 207) and _SC_DP_SYNCHRONIZATION_LOST (= 208) then initiate the PeripheralFaultTask and can be queried in their TaskStartInfo (TSI#InterruptId).
   - The **modeOfDpInterfaceSynchronization** system variable is set to MASTER_SLAVE_ALARMMESSAGES_1.

   The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).
2. If a higher-level bus cycle clock is recognized at the corresponding interface (this event is signaled by TSI#InterruptId = _SC_DP_CLOCK_DETECTED (= 207)), synchronization can be carried out:
   - Deactivate all DP slaves and assigned technology objects that are connected to the isochronous DP master interfaces and in which faults could occur during a phase jump (e.g. SINAMICS drives) (see Activating and deactivating components and technology objects (Page 1031)).
   - Call the **_synchronizeDpInterfaces** system function.
     You can find out about the current synchronization state using the stateOfDpInterfaceSynchronization system variable.
   - Once synchronization has been successfully completed, activate the DP slaves which you previously deactivated and their assigned technology objects (see Activating and deactivating components and technology objects (Page 1031)).

   The synchronized state continues to apply unless it is disrupted, e.g. if the constant bus cycle clock should fail.
   The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

If the alarms for initiating the PeripheralFaultTask are activated, they must be activated in the execution system and an appropriate program assigned. Otherwise the SIMOTION device enters the STOP operating mode when this event occurs.

Status diagram of user-controlled synchronization

The status diagram in the figure below describes the fundamental procedure for synchronizing isochronous DP master interfaces in a SIMOTION device with a higher-level bus cycle clock.

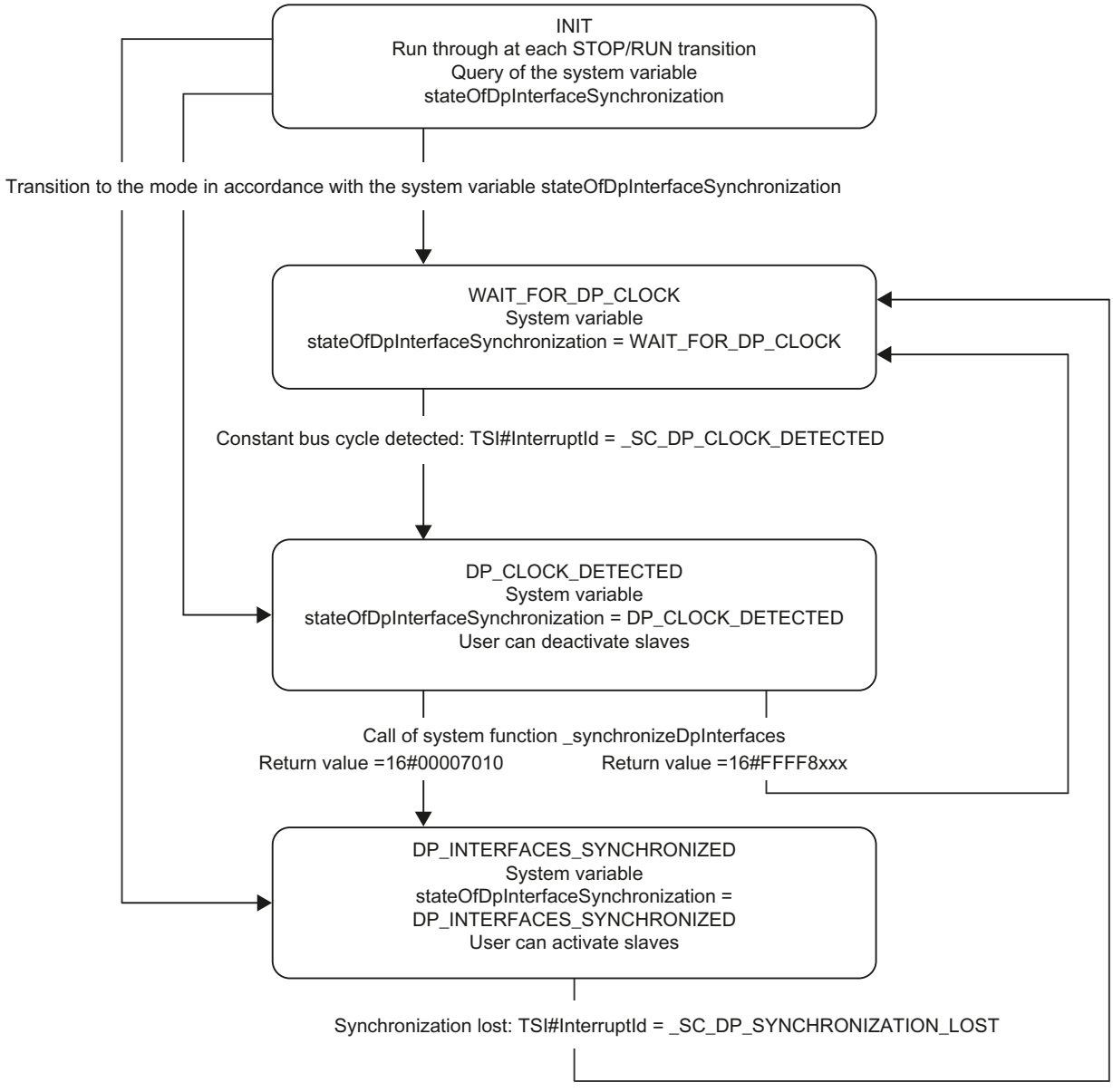


Figure 4-11 Status diagram for user-controlled synchronization of an isochronous DP master interface with a higher-level bus cycle clock

4.1.3.4 Behavior in the STOP and RUN operating modes

The SIMOTION device exhibits the following behavior in the operating modes and in transition between them. Observe this information when configuring/programming synchronization variants in accordance with Synchronizing a SIMOTION device without an isochronous DP master interface (Page 1006) and Synchronization of a SIMOTION device with an isochronous DP master interface (Page 1008).

Behavior in the STOP operating mode

If the interfaces of a SIMOTION device are not consecutively synchronized, the input cycle clock is monitored and a signal generated that leads to the _SC_DP_CLOCK_DETECTED event in the RUN operating mode.

If the interfaces are consecutively synchronized, the synchronicity and the input cycle clock are monitored. If the input cycle clock fails, a signal will be generated that results in the _SC_DP_SYNCHRONIZATION_LOST event in the RUN operating mode.

The state of the DP slaves does not change. An activated slave remains activated and a deactivated slave remains deactivated.

Behavior during the transition from STOP operating mode to RUN operating mode

The system variables are updated.

Processing of the user program will be delayed until, for example, any internal processing of the system functions _activateDpSlave, _deactivateDpSlave, or _getStateOfDpSlave (see Activating and deactivating nodes on the PROFIBUS or PROFINET IO (Page 1031)) which is still running has been completed. There is no separate time monitoring.

Behavior in the RUN operating mode

In the RUN operating mode, the events result in the PeripheralFaultTask being started with the associated TSI#InterruptId. System functions can be called up in the user program or in the PeripheralFaultTask.

The user can use the stateOfDpInterfaceSynchronization system variable and the _getStateOfDpSlave or _getStateOfAllDpStations system function to determine the current system state and restart its application.

Behavior during the transition from RUN operating mode to STOP operating mode

The transition can be made at any time. It is possible that slaves may have been started already or even deactivated. Interfaces can be synchronous or asynchronous.

All called system functions end in a defined state:

- The _synchronizeDpInterfaces system function will be cancelled.
- The system functions _activateDpSlave, _deactivateDpSlave, and _getStateOfDpSlave will undergo further internal processing.

4.1.4 Setting the communication addresses via the user program

Nodes are clearly identified within a communication network by means of, for example:

- The PROFIBUS address of the interface in PROFIBUS DP
- The device name (NameOfStation) in PROFINET IO
- The IP address of the interface on Ethernet

These communication addresses are set during configuration with the help of the engineering software (e.g. SIMOTION SCOUT). They can be changed later by a user program and customized according to requirements.

The address to be set can be determined in the user program in various ways, e.g.:

- Reading in and evaluating a slot coding
- Evaluating a program variable (which can be changed, for example, by a user input on the HMI device)

Note

In a communication network where PROFIsafe is also used, communication addresses may not be changed in the user program.

Note

You will find additional information on PROFIBUS and PROFINET in the SIMOTION Communication System Manual.

Note

The freely available and SIEMENS-supported PRONETA tool can be used to set and configure the PROFINET communication addresses. Associated information and the download are available:

- on the Internet (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert).
- in SIOS (https://support.industry.siemens.com/cs/document/67460624/proneta-2-3-0-26-commissioning-and-diagnostics-tool-for-profinet?dti=0&lc=en-WW).

4.1.4.1 Setting the PROFIBUS address

Using system functions, the user program can set and activate the required address (node number) for the SIMOTION device on PROFIBUS DP. The activation initiates a restart of the SIMOTION device.

It is only possible to change the address at the interface when this is set to the DP slave operating mode.

Note

After changing the configured address, it is no longer possible to route via this interface into other devices or interfaces (see "Routing in the communication network" in the section titled System behavior (Page 1017)).

If it must still be possible to route via this interface into other devices or interfaces, use the "Activating a configuration" (Page 1078) functionality instead. The PROFIBUS interface for the configuration to be activated must be configured with the desired address.

System functions

- **_getActiveDpSlaveAddress**
  This function is used to determine the active DP address (node number) of a DP interface on a SIMOTION device.
- **_setDpSlaveAddress**
  This function is used to set a new DP address (node number) for a DP interface on a SIMOTION device.
- **_activateDpSlaveAddress**
  This function is used to activate the DP address (node number) for a DP interface on a SIMOTION device.
  This function initiates a restart of the SIMOTION device.

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

You specify the DP interface of the SIMOTION device via its project-wide, unique, logical, diagnostics address. This diagnostics address is obtained, e.g. in HW Config, via the object properties of the interface (Addresses tab).

Example for reading, setting, and activating the DP slave address

```
// Variable declaration
VAR
    // Diagnostics address of the PROFIBUS interface
    logAddrDpAdapter : DINT := 1023;
    // PROFIBUS address of the local slot
    locDpSlaveAddress : SINT := 4;
    // Variables for the return values
    retDpSlaveAddress : structRetDpSlaveAddress;
    locRetVal : DINT;
    neededSetDpAddress : DINT := 0;
END_VAR

// Read active DP slave address
retDpSlaveAddress := _getActiveDpSlaveAddress (
    logicalAddressCommunicationAdapter := logAddrDpAdapter
    // Diagnostics address of the PROFIBUS interface
    );

IF (0 = retDpSlaveAddress.functionResult) THEN
    // Check if new DP slave address is necessary
    IF (retDpSlaveAddress.dpSlaveAddress <> locDpSlaveAddress) THEN
        // Need to set new DP slave address
        neededSetDpAddress := 1;
    ELSE
        ; //User-defined error response
    END_IF;
END_IF;

IF (1 = neededSetDpAddress) THEN
    // Set new DP slave address
    locRetVal := _setDpSlaveAddress (
        logicalAddressCommunicationAdapter := logAddrDpAdapter,
        // Diagnostics address of the PROFIBUS interface
        dpSlaveAddress := locDpSlaveAddress
        // PROFIBUS address of the local slot
        );
    IF (0 = locRetVal) THEN
        locRetVal := _activateDpSlaveAddress (
            logicalAddressCommunicationAdapter := logAddrDpAdapter
            // Diagnostics address of the PROFIBUS interface
            );
    ELSE
        ; // User-defined error response
    END_IF;
END_IF;
```

System behavior

Configuring the DP slave interface

The following should be noted when configuring the SIMOTION device interface that has been configured as a DP slave:

The configured quantity structure of the data to be exchanged must be identical in both the DP slave and the higher-level DP master.

Only then can the interface function as a DP slave to the higher-level DP master.

Routing in the communication network

The specific routing information is stored in each device corresponding to the communication addresses configured there. When the address is changed, the appropriate routing information no longer remains in the device.

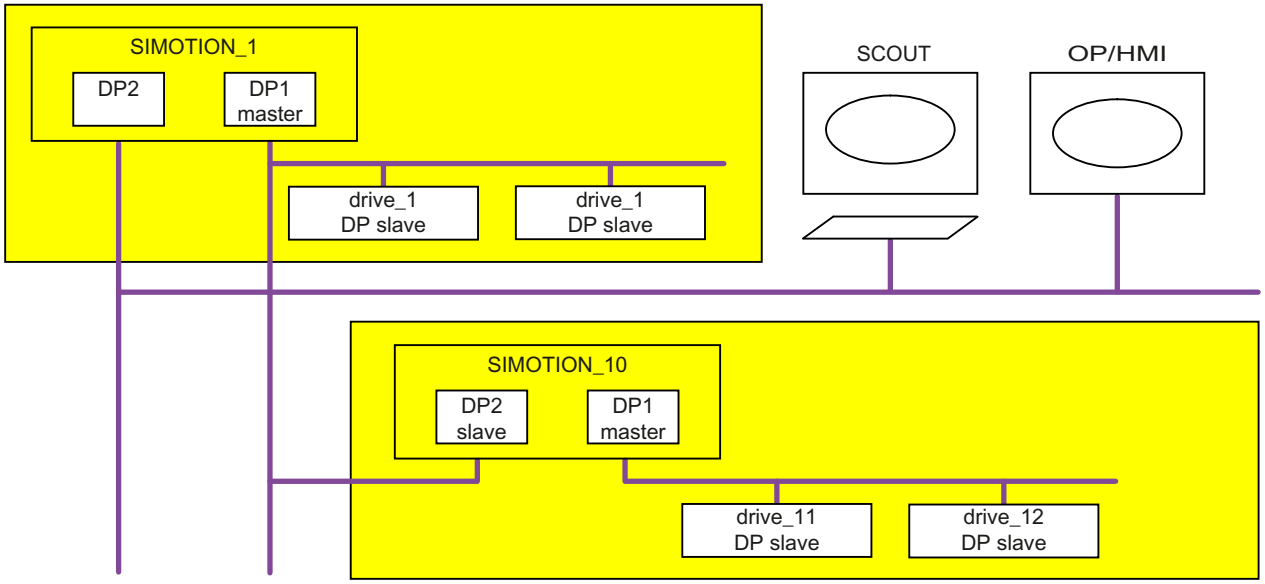


Figure 4-12 Example: Routing in the communication network

In SIMOTION_10, the DP slave address of the DP2 interface is changed in the user program. Routing from the OP or SCOUT via SIMOTION_1 (DP2/DP1) to SIMOTION_10 (DP2/DP1) and to the drives (drive_11, drive_12, etc.) is then no longer possible. No further control, loading, or monitoring actions are possible from the SCOUT.

Communication networks with PROFIsafe

In a communication network where PROFIsafe is also used, communication addresses may not be changed in the user program.

4.1.4.2 Topology detection in PROFINET IO

In PROFINET IO, one port belonging to a node is connected to exactly one port belonging to another node. A SIMOTION device cyclically identifies the connected neighbors for all the local ports, regardless of the configured transfer procedure.

Note

No topology detection is possible in PROFIBUS DP due to the other existing bus topology.

The result for each local port can be read out using the following system functions.

- As of SIMOTION Kernel version V4.4, the following function should be used in new projects: **_getPnPortNeighbour** (as of Kernel V4.4, replaces _getPnInterfacePortNeighbour)
- Up to SIMOTION Kernel version V4.3, the following function should be used: **_getPnInterfacePortNeighbour**

As a result, you receive the following data on the connected neighbor node, which allows for clear identification of this node:

- The device name (NameOfStation) of the connected interface of the PROFINET IO node.
- The number of the connected port (value range 1 to 255)
- The number of the slot (detection of the PROFINET IO interface) containing the connected port (value field 0 to 65535)

Exceptions:

- If no PROFINET IO node is connected to the local port, you receive the following return value:
  - As a device name: An empty string
  - And as a number for the port and the slot: 16#FFFFFFFF in each case
- If the connected PROFINET IO node only contains a single PROFINET IO interface, 16#FFFFFFFF will be returned as the number of the slot.
- If the connected PROFINET IO node has not yet been assigned a device name (NameOfStation), the return value for the device name will comprise:
  - PROFINET IO node is not a SIMOTION device: The hardware address (MAC address) will be returned. The MAC address will be shown in the usual notation xx:xx:xx:xx:xx:xx, where x is a hexadecimal number [0 to 9, A to F].
  - PROFINET IO node is a SIMOTION device: A character string containing the device type and the MAC address hexadecimal numbers will be returned, e.g. SIMOTION-D-08-00-06-73-6C-E6.

The syntax of these system functions is described in detail in the List Manual (reference list) of the SIMOTION devices and in the online help (see index).

For information about the syntax of the device name (NameOfStation), see Setting the device name (NameOfStation) of an IO device on PROFINET IO (Page 1020).

An example of the _getPnPortNeighbour system function is contained in Section "System functions" for setting the device name (Page 1021).

The figure below displays an example for a topology in PROFINET IO. Topology detection is carried out on the SIMOTION devices for various ports using the _getPnPortNeighbour or _getPnInterfacePortNeighbour system function. The result is given in the table.

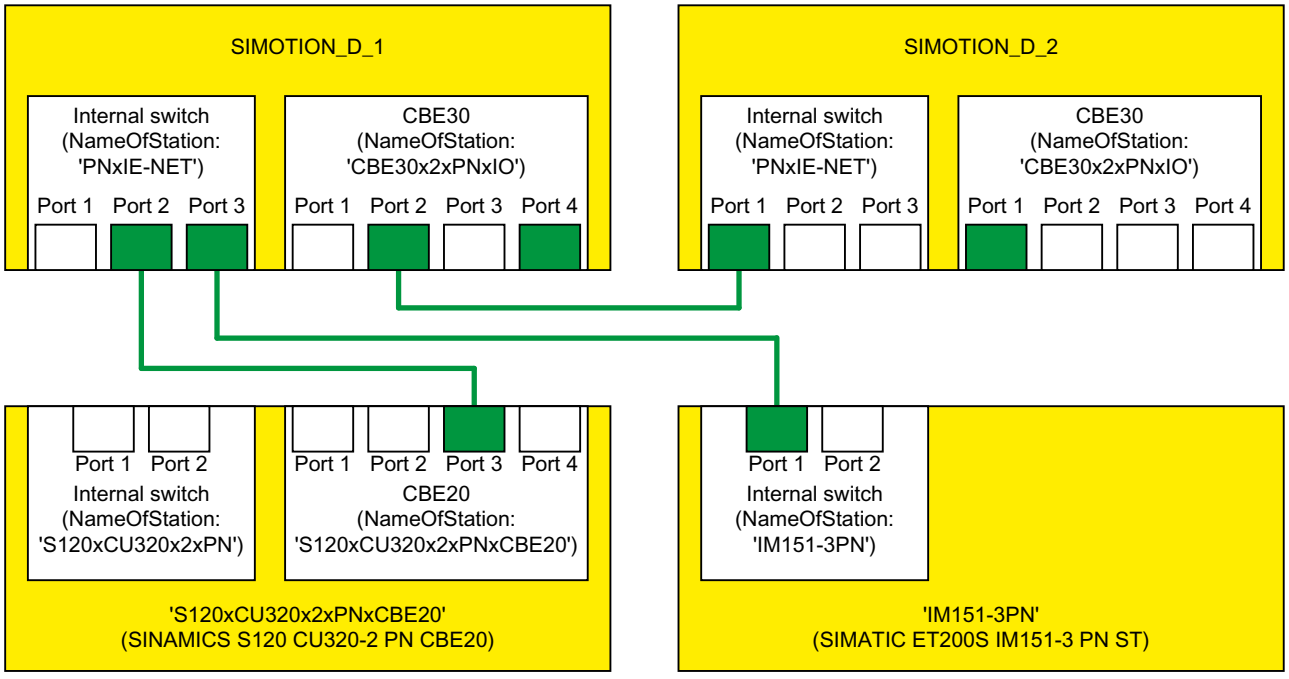


Figure 4-13 Example of a topology with PROFINET IO

Table 4-2 Topology detection (call up of the _getPnPortNeighbour or _getPnInterfacePortNeighbour function) for the topology example in the previous figure

Topology detection for device (interface, port)	Result of the topology detection (components of the return value)			
	functionResult	nameOfStation	pnPortNumber	pnSlotNumber
SIMOTION_D_1 (internal switch, port 2)	0	'S120xCU320x2xPNxCBE30'	3	0
SIMOTION_D_1 (internal switch, port 3)	0	'IM151-3PN'	1	0
SIMOTION_D_1 (CBE30, port 2)	0	'PNxIE-NET'	1	0
SIMOTION_D_1 (CBE30, port 4)	16#FFFF80A2 [1]	' '	-1	-1
SIMOTION_D_2 (internal switch, port 1)	0	'CBE30x2xPNxIO'	2	0
SIMOTION_D_2 (CBE30, port 1)	16#FFFF80A2 [1]	' '	-1	-1

[1] Only with the _getPnPortNeighbour function. The following applies with the _getPnInterfacePortNeighbour function: functionResult = 0.

4.1.4.3 Setting the device name (NameOfStation) of an IO device on PROFINET IO

The user program can change and activate the device name (NameOfStation) for the SIMOTION device on the PROFINET IO using system functions. Activation with the system function _activateNameOfStation initiates a restart of the SIMOTION device.

Note

After changing the configured device name (NameOfStation), the following should be noted:

- It is no longer possible to route via this interface into other devices or interfaces (see *Routing in the communication network* in chapter System behavior (Page 1017)).
- Controller-controller data exchange broadcast is no longer possible.

If these functionalities are still required, follow Activating a configuration (Page 1078) instead. The desired device name (NameOfStation) must be configured for the configuration to be activated.

Syntax of the device name (NameOfStation)

The device name (NameOfStation) must adhere to the following conventions:

- Permissible length: 1 to 127 characters
- Organization using periods "." into several labels is permissible; length of a single label: 1 to 63 characters.
- Characters permitted within a label:
  - Letters "A" to "Z" and "a" to "z".
  - Numbers "0" to "9" (not at the beginning of the label).
  - Special character hyphen "-" (not at the beginning or end of a label).
  - Other special characters (such as umlauts, blank spaces, brackets, underscores) are not permitted.
- The following designations are not permitted for device names:
  - Identifiers that start with "port-xyz-" (x, y, z = 0 ... 9).
  - Identifiers of the form n.n.n.n (n = 0 ... 999).

Note

Device names which do not satisfy the general rules for identifiers (i.e. containing hyphens or periods) must be enclosed in double inverted commas (", ASCII code $22) for use in SIMOTION SCOUT (e.g. in the programming languages).

System functions

As of SIMOTION Kernel version V4.4, the following functions should be used in new projects:

- **_getPnNameOfStation** (as of Kernel V4.4)
  You can use this function to determine the active device name (NameOfStation) of a PROFINET IO interface on the SIMOTION device.
  If the PROFINET IO interface has not yet been assigned a device name (NameOfStation), a character string containing the device type and the hexadecimal numbers of the MAC address will be returned, e.g. SIMOTION-D-08-00-06-73-6C-E6.
- **_setPnNameOfStation** (as of Kernel V4.4)
  You can use this function to set a new device name (NameOfStation) for a PROFINET IO interface on the SIMOTION device and activate this name.

Up to SIMOTION Kernel version V4.3, the following functions should be used:

- **_getActiveNameOfStation**
  You can use this function to determine the active device name (NameOfStation) of a PROFINET IO interface on the SIMOTION device.
  If the PROFINET IO interface has not yet been assigned a device name (NameOfStation), a character string containing the device type and the hexadecimal numbers of the MAC address will be returned, e.g. SIMOTION-D-08-00-06-73-6C-E6
  As of SIMOTION Kernel version V4.4, the _getPnNameOfStation function should be used in new projects:
- **_setNameOfStation**
  You can use this function to set a new device name (NameOfStation) for a PROFINET IO interface on the SIMOTION device.
  As of SIMOTION Kernel version V4.4, the _setPnNameOfStation function should be used in new projects:
- **_activateNameOfStation**
  This function activates all of the device names (NameOfStation) of the PROFINET IO interfaces previously set by the system function _setNameOfStation and initiates a restart of the SIMOTION device.
  As of SIMOTION Kernel version V4.4, this function is integrated in _setPnNameOfStation.

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

For information about synchronizing the PROFINET IO interface, see Cycle clock generation and synchronization in PROFINET IO (Page 1003).

Examples

The following examples demonstrate the reading and setting of the device name.

Example with _getPnPortNeighbour, _getPnNameOfStation, _setPnNameOfStation (as of V4.4, asynchronous call)

```
// Program bgrName is assigned to a cyclic task.
// (e.g. BackgroundTask)
INTERFACE
  VAR_GLOBAL CONSTANT
    // D4x5-2 diagnostics address X150 -----------
    X150_INTERFACE_LOG_ADDRESS : DINT := 16378;
    X150_PORT_1_LOG_ADDRESS    : DINT := 16377;
    X150_PORT_2_LOG_ADDRESS    : DINT := 16376;
    X150_PORT_3_LOG_ADDRESS    : DINT := 16375;
    // D4x5-2 diagnostics address X127 -----------
    X127_INTERFACE_LOG_ADDRESS : DINT := 16374;
    X127_PORT_1_LOG_ADDRESS    : DINT := 16373;
    // D4x5-2 diagnostics address X130 -----------
    X130_INTERFACE_LOG_ADDRESS : DINT := 16389;
    X130_PORT_1_LOG_ADDRESS    : DINT := 16379;
  END_VAR
  TYPE
    bgrPortNeighbourStruct  : STRUCT
        ioId                : EnumIoIdType;
        logAddressPort      : DINT;
        requestMode         : EnumReqSysFunctMode;
        commandId           : commandIdType;
        nextCommand         : enumNextCommandMode;
        startAgain          : BOOL;
        retPortNeighbour    : StructRetDeviceNameOfStation;
    END_STRUCT;

    bgrNameStruct   : STRUCT
        ioId                : EnumIoIdType;
        logAddressPort      : DINT;
        nameOfStation       : STRING;
        storePermanent      : EnumYesNo;
        commandId           : commandIdType;
        nextCommand         : enumNextCommandMode;
        startAgain          : BOOL;
        retNameOfStation    : StructRetDeviceNameOfStation;
        ret                 : DINT;
    END_STRUCT;
  END_TYPE
  VAR_GLOBAL
    bgrStart                : BOOL := FALSE;
    bgrNameState            : DINT := 0;
    bgrNamePortNeighbour    : bgrPortNeighbourStruct;
    bgrNameOfStation        : bgrNameStruct;
  END_VAR

  PROGRAM bgrName;
END_INTERFACE
```

Example with _getPnPortNeighbour, _getPnNameOfStation, _setPnNameOfStation (as of V4.4, asynchronous call)

```
IMPLEMENTATION
  PROGRAM bgrName
    CASE bgrNameState OF
    0:      IF bgrStart THEN
                bgrNameState := 1000;
                bgrStart := FALSE;
            END_IF;
    1000:   // Initialization for _getPnPortNeighbour
            bgrNamePortNeighbour.ioId           := INPUT;
            bgrNamePortNeighbour.logAddressPort := X150_INTERFACE_LOG_ADDRESS;
            bgrNamePortNeighbour.commandId      := _getCommandId ();
            bgrNamePortNeighbour.nextCommand    := IMMEDIATELY;
            bgrNamePortNeighbour.retPortNeighbour.functionResult := 0;

            bgrNameState := 1001;
    1001:   // Call of _getPnPortNeighbour
            bgrNamePortNeighbour.retPortNeighbour := _getPnPortNeighbour (
                    ioId                  := bgrNamePortNeighbour.ioId,
                    logicalAddressOfPnPort := bgrNamePortNeighbour.logAddressPort,
                    nextCommand           := bgrNamePortNeighbour.nextCommand,
                    commandId             := bgrNamePortNeighbour.commandId);
            CASE bgrNamePortNeighbour.retPortNeighbour.functionResult OF
            16#0000:      // Function terminated successfully
                        bgrNameState := 2000;
            16#7001:    ; // Function started
            16#7002:    ; // Function active
            16#7003:    ; // Function aborted
            ELSE          // Error
                        bgrNameState := 10009;
            END_CASE;
    2000:   // Initialization for _getPnNameofStation
            bgrNameOfStation.ioId           := INPUT;
            bgrNameOfStation.logAddressPort := X150_INTERFACE_LOG_ADDRESS;
            bgrNameOfStation.commandId      := _getCommandId ( );
            bgrNameOfStation.nextCommand     := IMMEDIATELY;
            bgrNameOfStation.retNameOfStation.functionResult := 0;
            bgrNameState := 2001;
    2001:   // Call of _getPnNameofStation
            bgrNameOfStation.retNameOfStation := _getPnNameOfStation (
                    ioId                      := bgrNameOfStation.ioId,
                    logicalAddressPnInterface := bgrNameOfStation.logAddressPort,
                    nextCommand               := bgrNameOfStation.nextCommand,
                    commandId                 := bgrNameOfStation.commandId);
            CASE bgrNameOfStation.retNameOfStation.functionResult OF
            16#0000:      // Function terminated successfully
                        bgrNameState := 3000;
            16#7001:    ; // Function started
            16#7002:    ; // Function active
            16#7003:    ; // Function aborted
            ELSE          // Error
                        bgrNameState := 20009;
            END_CASE;
```

Example with _getPnPortNeighbour, _getPnNameOfStation, _setPnNameOfStation (as of V4.4, asynchronous call)

```
    3000:   // Initialization for _setPnNameofStation
            bgrNameOfStation.ioId           := INPUT;
            bgrNameOfStation.logAddressPort := X150_INTERFACE_LOG_ADDRESS;
            bgrNameOfStation.nameOfStation  := 'blaalb';
            bgrNameOfStation.storePermanent := NO;
            bgrNameOfStation.commandId      := _getCommandId ( );
            bgrNameOfStation.nextCommand    := IMMEDIATELY;
            bgrNameOfStation.ret            := 0;
            bgrNameState := 3001;
    3001:   // Call of _setPnNameofStation
            bgrNameOfStation.ret := _setPnNameOfStation(
                    ioId                        := bgrNameOfStation.ioId,
                    logicalAddressPnInterface   := bgrNameOfStation.logAddressPort,
                    NameOfStation               := bgrNameOfStation.nameOfStation,
                    storeNameOfStationPermanent := bgrNameOfStation.storePermanent,
                    nextCommand                 := bgrNameOfStation.nextCommand,
                    commandId                   := bgrNameOfStation.commandId);
            CASE bgrNameOfStation.ret OF
            16#0000:      // Function terminated successfully
                        bgrNameState := 0;
            16#7001:    ; // Function started
            16#7002:    ; // Function active
            16#7003:    ; // Function aborted
            ELSE          // Error
                        bgrNameState := 30009;
            END_CASE;
    ELSE
            ; // Error handling
    END_CASE;
  END_PROGRAM
END_IMPLEMENTATION
```

Example with _getActiveNameOfStation, _setNameOfStation, _activateNameOfStation (up to V4.3, synchronous call)

```
// Program assigned to a MotionTask
// Variable declaration
VAR
    // Diagnostics address of the PROFINET IO interface
    logAddrPnAdapter : DINT := 1023;
    // Device name of the PROFINET IO interface
    locNameOfStation : STRING[240];
    // Variables for the return values
    retNameOfStation : StructRetDeviceNameOfStation;
    locRetVal : DINT;
    neededSetNameOfStation : DINT := 0;
    locCommandId : CommandIdType;
END_VAR
```

Example with _getActiveNameOfStation, _setNameOfStation, _activateNameOfStation (up to V4.3, synchronous call)

```
// Read active device name
retNameOfStation := _getActiveNameOfStation (
    logicalAddressPnInterface := logAddrPnAdapter,
    // Diagnostics address of the PROFINET IO interface
    requestMode := REQUEST_TRUE,
    // Function will be executed
    commandId := locCommandId,
    nextCommand := WHEN_COMMAND_DONE
    // Synchronous call; wait until the function has terminated
    );
IF (0 = retNameOfStation.functionResult) THEN
    // Check if new device name is necessary
    IF (retNameOfStation.nameOfStation <> locNameOfStation) THEN
        // Need to create a new device name
        neededSetNameOfStation := 1;
    ELSE
        ; // User-defined error response
    END_IF;
END_IF;
IF (1 = neededSetNameOfStation) THEN
    // Set new device name
    locRetVal := _setNameOfStation (
        logicalAddressPnInterface := logAddrPnAdapter,
        // Diagnostics address of the PROFINET IO interface
        nameOfStation := locNameOfStation,
        // Device name of the PROFINET IO interface
        requestMode := REQUEST_TRUE,
        // Function will be executed
        commandId := locCommandId,
        nextCommand := WHEN_COMMAND_DONE
        // Synchronous call
        );
    IF (0 = locRetVal) THEN
        locRetVal := _activateNameOfStation (
            logicalAddressPnInterface := logAddrPnAdapter
            // Diagnostics address of the PROFIBUS interface
            );
    ELSE
        ; // User-defined error response
    END_IF;
END_IF;
```

System behavior

Configuring the PROFINET IO interface

The following should be noted when configuring the PROFINET IO interface of the SIMOTION device (IO device): The configured quantity structure of the data to be exchanged must be identical in both the IO device and the higher-level IO controller.

Only then can this interface function as an IO device to the higher-level IO controller.

Routing in the communication network

The specific routing information is stored in each device corresponding to the communication addresses configured there. After the device name (NameOfStation) has been changed, no further appropriate routing information remains in the device.

Data exchange broadcast

Controller-controller data exchange broadcast is no longer possible. The data exchange broadcast partner can no longer be found after the device name (NameOfStation) has been changed.

Communication networks with PROFIsafe

In a communication network where PROFIsafe is also used, the device names (NameOfStation) may not be changed in the user program.

4.1.4.4 Setting the IP address (on Ethernet)

The IP configurations (IP address, subnet mask, gateway address) can be read in, set, or changed using the system functions _setIpConfig and _getIpConfig. SIMOTION remains in the RUN operating mode.

System functions

As of SIMOTION Kernel version V4.4, the following functions should be used in new projects:

- **_getPnIpConfig** (as of Kernel V4.4)
  You can use this function to determine the IP configuration of an Ethernet interface which you can specify via the logical diagnostics address of the port.
- **_setPnIpConfig** (as of Kernel V4.4)
  You can use this function to set the IP configuration of an Ethernet interface which you can specify via the logical diagnostics address of the port.

Up to SIMOTION Kernel version V4.3, the following functions should be used:

- **_getIpConfig**
  You can use this function to determine the IP configuration of an Ethernet interface; you can specify the interface by inputting the corresponding enumeration value (IE_01, IE_02).
- **_setIpConfig**
  You can use this function to set the IP configuration of an Ethernet interface; you can specify the interface by inputting the corresponding enumeration value (IE_01, IE_02).

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

The factory settings of the IP configurations (e.g. default IP address) vary from one SIMOTION device to the next. Please check the relevant value in the Commissioning and Hardware Installation Manual or in the Operating Instructions of the respective SIMOTION device.

Examples

The following examples demonstrate the setting of an IP address.

Example with _getPnIpConfig and _setPnIpConfig (as of V4.4, asynchronous call)

```
// Program bgr assigned to the BackgroundTask
INTERFACE
  VAR_GLOBAL CONSTANT
    // D4x5-2 diagnostics address X150 -----------
    X150_INTERFACE_LOG_ADDRESS : DINT := 16378;
    X150_PORT_1_LOG_ADDRESS    : DINT := 16377;
    X150_PORT_2_LOG_ADDRESS    : DINT := 16376;
    X150_PORT_3_LOG_ADDRESS    : DINT := 16375;
    // D4x5-2 diagnostics address X127 -----------
    X127_INTERFACE_LOG_ADDRESS : DINT := 16374;
    X127_PORT_1_LOG_ADDRESS    : DINT := 16373;
    // D4x5-2 diagnostics address X130 -----------
    X130_INTERFACE_LOG_ADDRESS : DINT := 16380;
    X130_PORT_1_LOG_ADDRESS    : DINT := 16379;
  END_VAR
  TYPE
    pnIpConfigStruct    : STRUCT
        result_getIpConfig : StructRetPnIpConfig;
        result_setIpConfig : UDINT;
        ioId               : EnumIoIdType;
        logAddress         : DINT;
        ip                 : ARRAY [0..3] OF USINT;
        mask               : ARRAY [0..3] OF USINT;
        gate               : ARRAY [0..3] OF USINT;
        storePermanent     : EnumYesNo;
        nextCommand        : EnumNextCommandMode;
        commandId          : commandIdType;
    END_STRUCT;
  END_TYPE
  VAR_GLOBAL
    bgrStart      : BOOL  := FALSE;
    bgrState      : DINT  := 0;
    bgrI          : DINT  := 0;
    bgrPnIpConfig : pnIpConfigStruct;
  END_VAR

  PROGRAM bgr;
END_INTERFACE
IMPLEMENTATION
  PROGRAM bgr
    CASE bgrState OF
    0:      IF bgrStart THEN
                bgrState := 1000;
                bgrStart := FALSE;
            END_IF;
```

Example with _getPnIpConfig and _setPnIpConfig (as of V4.4, asynchronous call)

```
1000:   // Initialization
        bgrPnIpConfig.ioId        := INPUT;
        bgrPnIpConfig.logAddress  := X150_PORT_1_LOG_ADDRESS; // 16378;
        bgrPnIpConfig.nextCommand := IMMEDIATELY;
        bgrPnIpConfig.commandId   := _getCommandId ();
        bgrPnIpConfig.result_getIpConfig.functionResult := 0;

        bgrState := 1001;
1001:   // Call of _getPnIpConfig
        bgrPnIpConfig.result_getIpConfig := _getPnIpConfig (
                ioId                      := bgrPnIpConfig.ioId,
                logicalAddressPnInterface := bgrPnIpConfig.logAddress,
                nextCommand               := bgrPnIpConfig.nextCommand,
                commandId                 := bgrPnIpConfig.commandId );
        CASE bgrPnIpConfig.result_getIpConfig.functionResult OF
        16#0000:      // Function terminated successfully
                    bgrState := 2000;
        16#7001:    ; // Function started
        16#7002:    ; // Function active
        16#7003:    ; // Function aborted
        ELSE          // Error
                    bgrState := 10009;
        END_CASE;
2000:   // Initialization
        bgrPnIpConfig.ioId            := INPUT;
        bgrPnIpConfig.logAddress      := X150_PORT_1_LOG_ADDRESS; // 16378;
        bgrPnIpConfig.ip [0]          := 169;
        bgrPnIpConfig.ip [1]          := 250;
        bgrPnIpConfig.ip [2]          := 11;
        bgrPnIpConfig.ip [3]          := 169;
        bgrPnIpConfig.mask [0]        := 255;
        bgrPnIpConfig.mask [1]        := 255;
        bgrPnIpConfig.mask [2]        := 0;
        bgrPnIpConfig.mask [3]        := 0;
        bgrPnIpConfig.gate [0]        := 169;
        bgrPnIpConfig.gate [1]        := 250;
        bgrPnIpConfig.gate [2]        := 11;
        bgrPnIpConfig.gate [3]        := 1;
        bgrPnIpConfig.storePermanent  := NO;
        bgrPnIpConfig.commandId       := _getCommandId ();
        bgrPnIpConfig.nextCommand     := IMMEDIATELY;
        bgrPnIpConfig.result_setIpConfig := 0;

        bgrState := 2001;
```

Example with _getPnIpConfig and _setPnIpConfig (as of V4.4, asynchronous call)

```
    2001:   // Call of _setPnIpConfig
            bgrPnIpConfig.result_setIpConfig := _setPnIpConfig (
                    ioId                        := bgrPnIpConfig.ioId,
                    logicalAddressPnInterface   := bgrPnIpConfig.logAddress,
                    ipAddress                   := bgrPnIpConfig.ip,
                    subnetMask                  := bgrPnIpConfig.mask,
                    gatewayAddress              := bgrPnIpConfig.gate,
                    storeIpConfigPermanent      := bgrPnIpConfig.storePermanent,
                    nextCommand                 := bgrPnIpConfig.nextCommand,
                    commandId                   := bgrPnIpConfig.commandId );
            CASE bgrPnIpConfig.result_setIpConfig OF
            16#0000:      // Function terminated successfully
                        bgrState := 0;
            16#7001:    ; // Function started
            16#7002:    ; // Function active
            16#7003:    ; // Function aborted
            ELSE          // Error
                        bgrState := 20009;
            END_CASE;
    ELSE
            ; // Error handling
    END_CASE;
  END_PROGRAM
END_IMPLEMENTATION
```

Example with _getIpConfig and _setIpConfig (up to V4.3)

```
// Variable declaration
VAR
    ipRead : ARRAY [1..2] OF StructRetIpConfig;
    setIpAddress : ARRAY [0..5] OF USINT;
    setIpSubnet  : ARRAY [0..5] OF USINT;
    setIpGateway : ARRAY [0..5] OF USINT;
    retUdint : UDINT;
    (* IP address of the configuration will be read from the input with address 0. *)
    ip_conf_id AT %IB0 : USINT;
END_VAR
// Read the current data of the Ethernet interfaces
ipRead[1] := _getIpConfig (
    ethernetInterface:= IE_01    );
ipRead[2] := _getIpConfig (
    ethernetInterface:= IE_02    );
```

Example with _getIpConfig and _setIpConfig (up to V4.3)

```
// Change of the Ethernet interface IE_01.
// Check whether a change is necessary.
IF ( ip_conf_id <> ipRead[1].ipAddress[3] ) THEN
    // Make the data to be changed available
    setIpAddress := ipRead[1].ipAddress;
    setIpAddress[3] := ip_conf_id;
    setIpSubnet := ipRead[1].subnetMask;
    setIpGateway := ipRead[1].gatewayAddress;
    (* Check whether the two Ethernet interfaces are different after the subnet masks have
been changed: *)
    IF ( ipRead[2].subnetMask <> setIpSubnet ) THEN
        // Set the data of the Ethernet interface IE_01
        retUdint := _setIpConfig (
            ethernetInterface := IE_01,
            ipAddress := setIpAddress,
            subnetMask := setIpSubnet,
            gatewayAddress := setIpGateway
            );
    ELSE
        ; // No change, error response.
          // Also change interface IE_02, if necessary.
    END_IF;
END_IF;
```

4.1.4.5 Automatic address assignment with PROFINET IO systems (address tailoring)

The address tailoring function can be used to assign an IP address and a NameOfStation to IO controllers and IO devices by means of system functions at the commissioning stage. This feature allows a machine consisting of an IO controller and IO devices to be supplied with IP addresses and NameOfStation, and permits multiple instances of the machine to be commissioned on a single physical Ethernet without the engineering system.

There is only one project (configuration and programs). It can be loaded onto machines of the same type without change. To integrate the machine into an existing network infrastructure, only limited customization is required during on-site commissioning (IP address, NameOfStation). The user program assigns the IP address and the NameOfStation during the first controller startup.

Note

Address tailoring can be configured only for SIMOTION SCOUT TIA.

You can find further information about the following subjects in the SIMOTION SCOUT TIA Configuration Manual and in the online help:

- Reusable IO systems - Address tailoring
- Configuring the project for address tailoring

4.1.5 Activating and deactivating components and technology objects

This section describes how PROFIBUS DP slaves and technology objects are activated and deactivated.

4.1.5.1 Activating and deactivating nodes on the PROFIBUS or PROFINET IO

The user has configured and programmed a maximum configuration. The user program deactivates bus nodes that are not available or not required in order to achieve the following:

- The cyclic load (data exchange) between the DP master and the DP slaves (PROFIBUS) or the controller and the IO device (PROFINET IO) is reduced.
- The processing of alarms is suppressed for deactivated nodes.
- No bus fault is signaled for PROFIBUS or PROFINET IO nodes which are not available.
- DP slaves (drives) re-synchronize with the DP cycle clock.

The user program can also activate further nodes that are required, for example, in another production phase.

Note

If the DP slave or the IO device is integrated in PROFIsafe communication (e.g. SINAMICS with SINAMICS Safety Integrated Extended Functions): Deactivation of a bus node will result in PROFIsafe errors with corresponding responses in the failsafe control (F-CPU). If necessary, the control will shut down all other PROFIsafe nodes.

System functions

- _activateDpSlave (Page 1032)
- _deactivateDpSlave (Page 1032)
- _getStateOfSingleDpSlave (Page 1040)
- _getStateOfAllDpSlaves (Page 1040)
- _getStateOfAllDPStations (Page 1040)
- _getStateOfDpSlave (Page 1039)
- _getLogDiagnosticAddressFromDpStationAddress (Page 1041)
- _getDpStationAddressFromLogDiagnosticAddress (Page 1041)
- _getGeoAddressFromLogAddress (Page 1041)

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

Technology objects are often assigned to the distributed I/O (DP slave, IO device); therefore, please note the information in Procedure for deactivating and activating components (Page 1075).

System functions for activating and deactivating bus nodes

The example below shows a machine design consisting of a central unit and several machine modules, each with several drives that communicate over PROFIBUS.

- **Central unit (SIMOTION_1)**
  The PROFIBUS DP1 interface for the central unit is configured as a DP master for the subnet dp_subnet_1. Machine modules 10 and 20 are connected to this subnet.
- **Machine module 10 (SIMOTION_10)**
  The PROFIBUS DP2 interface for the machine module is configured as a DP slave for the subnet dp_subnet_1.
  The PROFIBUS DP1 interface for the machine module is configured as a DP master for the subnet dp_subnet_10. The drives are connected as DP slaves to this subnet.
- **Machine module 20 (SIMOTION_20)**
  The PROFIBUS DP2 interface for the machine module is configured as a DP slave for the subnet dp_subnet_1.
  The PROFIBUS DP1 interface for the machine module is configured as a DP master for the subnet dp_subnet_20. The drives are connected as DP slaves to this subnet.

This example can also be applied to PROFINET IO in comparable form.

Example structure of a modular machine

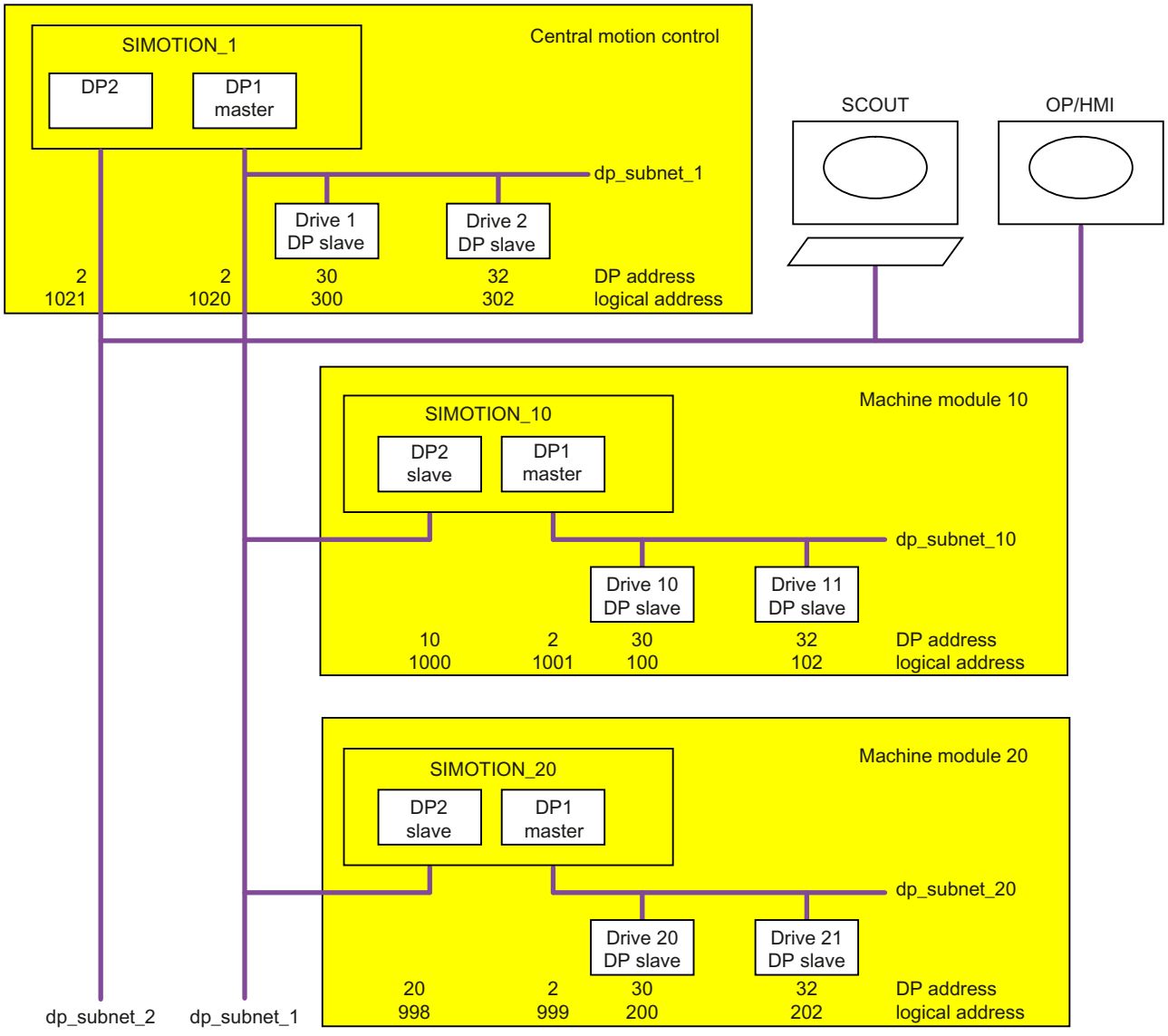


Figure 4-14 Example of a modular machine with differentiation between DP address and logical diagnostics address

System functions

You can use the **_deactivateDpSlave** and **_activateDpSlave** system functions to execute the following actions during operation:

- In the user program of the central unit: Activate or deactivate the machine modules 10 and 20.
- In the user program of the individual machine modules: Activate or deactivate the respective drives.

You can specify the node to be deactivated or activated (DP slave or IO device) using the logical diagnostics address for its interface; this address is unique across the project. The functions can be used on all DP slaves or IO devices (e.g. drives, ET 200, DP standard slaves, other stations).

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index). Please also note the information relating to the status diagram (Page 1034), asynchronous calls (Page 1035) and synchronous calls (Page 1037).

Functions (Page 1041) are available for converting a PROFIBUS address or device number for PROFINET IO into the logical diagnostics address and vice versa.

Note

The technology objects assigned to the nodes (e.g. the axes assigned to the drives) must be activated and deactivated separately (see Activating and deactivating technology objects (Page 1066)).

Observe the correct order (see Procedure for deactivating and activating components (Page 1075)).

Status diagram for the functions _activateDpSlave and _deactivateDpSlave

The figure below shows the states of the system functions _activateDpSlave and _deactivateDpSlave.

The functions can be called asynchronously and synchronously.

- With an asynchronous call (parameter **nextCommand** := IMMEDIATELY), the user program is continued immediately after the start of the system function.
  You must regularly query the state of the function to be able to determine when a command has been executed in full.
  The asynchronous call is required in cyclic tasks to ensure that their time monitoring function does not start.
- With a synchronous call (parameter **nextCommand** : =WHEN_COMMAND_DONE), the user program waits until the system function has been completed. The return value directly indicates whether the function call was successful.
  The synchronous call is recommended for MotionTasks.

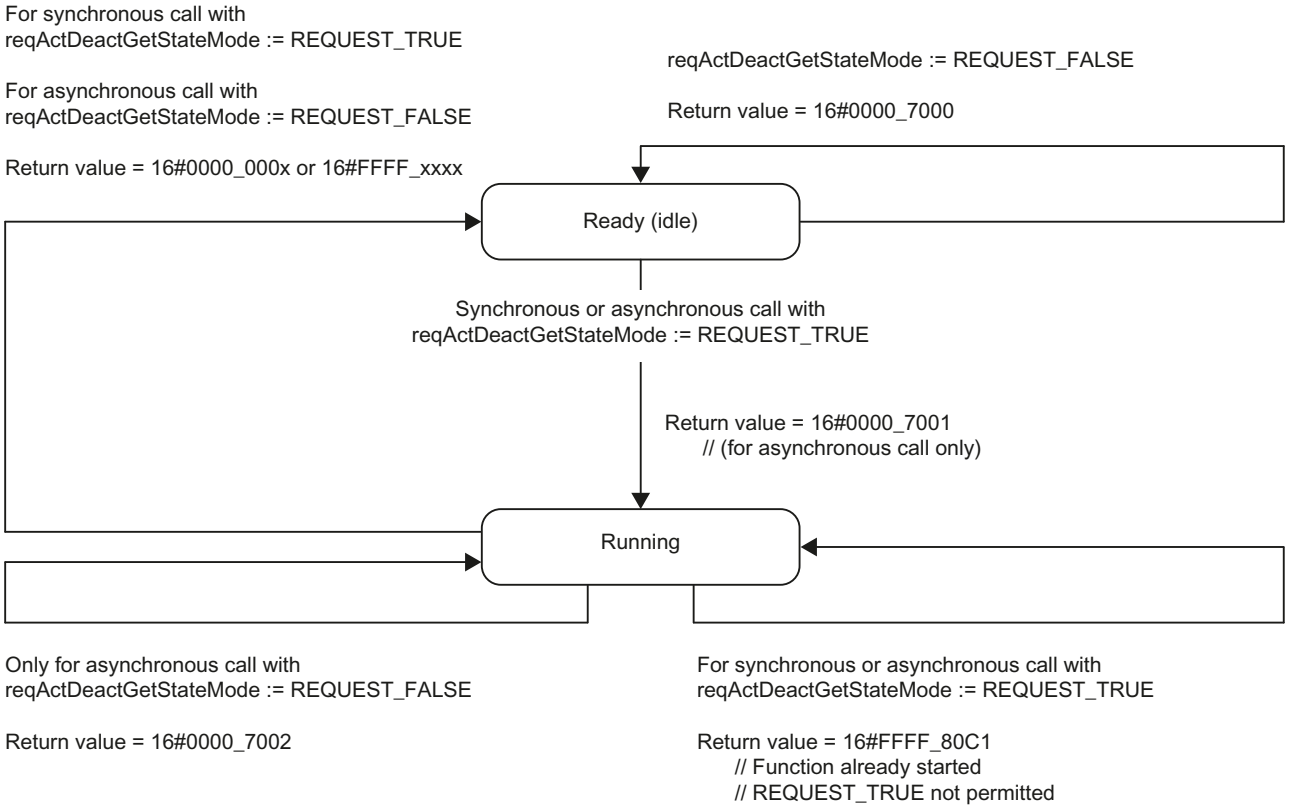
To execute the function (e.g. activation or deactivation of a DP slave or I/O device), start this with parameter **reqActDeactGetStateMode :=** REQUEST_TRUE.

Start the status query of the function with parameter **reqActDeactGetStateMode :=** REQUEST_FALSE. Use the same command ID as for the first call.

Note

To correctly complete the function with an asynchronous call, you must call the function with the parameter **reqActDeactGetStateMode :=** REQUEST_FALSE as often as necessary until you obtain 16#0000_000x as the return value or an error message.

You can execute the function only once at the same time for each bus node.



Return values:

16#0000_0001	// Activation successful, station recovery alarm is issued
16#0000_0002	// Deactivation successful, station failure alarm is issued
16#0000_0005	// Station already activated, station recovery alarm is not issued
16#0000_0006	// Station already deactivated, station failure alarm is not issued
16#0000_7000	// Function not active, initial call required with REQUEST_TRUE
16#0000_7001	// Function started (only for asynchronous call)
16#0000_7002	// Function active
16#FFFF_xxxx	// Function terminated with error

Figure 4-15 Status diagram of the system functions _activateDpSlave and _deactivateDpSlave

Asynchronous call for the functions _activateDpSlave and _deactivateDpSlave

The asynchronous call of the functions is required in cyclic tasks to ensure that the time monitoring function for the task does not start.

With an asynchronous call, the user program is continued immediately after the start of the system function. You must query the state of the function regularly to determine whether it has been completed without errors.

Proceed as follows for the asynchronous call:

1. Call the following parameters to start the system function:
   - reqActDeactGetStateMode := REQUEST_TRUE
   - nextCommand := IMMEDIATELY
2. Check the return value.
   - The return values 16#0000_0001, 16#0000_0002, 16#0000_0005, and 16#0000_0006 indicate that the function has been completed without errors.
   - The return value 16#0000_7001 indicates that the function has been started successfully, but has not yet been completed.
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
3. For the state query and to check whether the system function has been completed, call these again, but with the following parameters:
   - reqActDeactGetStateMode := REQUEST_FALSE
   - nextCommand := IMMEDIATELY
4. Check the return value:
   - Return value 16#0000_7002 indicates that the function is still running.
     Repeat step 3.
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
   - The return values 16#0000_0001, 6#0000_0002, 16#0000_0005, and 16#0000_0006 indicate that the function has been completed without errors.

The example program below shows the asynchronous call of the **_activateDpSlave** function in the BackgroundTask.

Example program for the asynchronous call for the _activateDpSlave function

```
INTERFACE
    PROGRAM backGround;

    VAR_GLOBAL
        bgrRequestActivate : DINT := 0;
        bgrResultActivate  : DINT := 0;
        bgrLogAddrDpSlave  : DINT := 16#FFFF_FFFF;
        bgrReqActDeactMode : enumReqActDeactGetStateMode := REQUEST_TRUE;
        bgrDpAlarmMode     : enumDeviceDpAlarmMode := SET_DP_ALARM;
        bgrNextCommand     : enumNextCommandMode := IMMEDIATELY;
        bgrTimeToWait      : UDINT := 30; // 30 seconds
    END_VAR
END_INTERFACE
IMPLEMENTATION
PROGRAM backGround // Assigned to the BackgroundTask
    VAR
        retVal : DINT;
    END_VAR
    // ... Further instructions
```

Example program for the asynchronous call for the _activateDpSlave function

```
    IF ( 0 <> bgrRequestActivate ) THEN
        retVal := _activateDpSlave (
            logicalAddressOfDpStation := bgrLogAddrDpSlave,
            reqActDeactGetStateMode := bgrReqActDeactMode,
            dpAlarmMode := bgrDpAlarmMode,
            nextCommand := bgrNextCommand,
            timeToWaitForStationAlarm := bgrTimeToWait
            (* As of SIMOTION Kernel version V4.4
               Waiting time for station ramp-up *)
            );
        CASE retVal OF
            16#0000_0001: bgrRequestActivate := 0;
                // Node successfully activated.
            16#0000_0005: bgrRequestActivate := 0;
                // Node was already activated, no alarm.
            16#0000_7001: bgrReqActDeactMode := REQUEST_FALSE;
                // Function started successfully
            16#0000_7002: ; // Function running.
        ELSE // Troubleshooting
            bgrRequestActivate := 0;
        END_CASE;
        IF ( 0 = bgrRequestActivate ) THEN
            // Function completed, prepare next call.
            bgrReqActDeactMode := REQUEST_TRUE;
            bgrResultActivate := retVal;
        END_IF;
    END_IF;
    // ... Further instructions
END_PROGRAM
END_IMPLEMENTATION
```

Synchronous call for the functions _activateDpSlave and _deactivateDpSlave

The synchronous call is recommended for MotionTasks.

With a synchronous call, the user program waits until the system function has been completed. The return value directly indicates whether the function call was successful.

Proceed as follows for the synchronous call:

1. Call the following parameters to start the system function:
   - reqActDeactGetStateMode = REQUEST_TRUE
   - nextCommand = WHEN_COMMAND_DONE
2. Check the return value:
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
   - The return values 16#0000_0001, 16#0000_0002, 16#0000_0005, and 16#0000_0006 indicate that the function has been completed without errors.

The example program below shows the synchronous call of the **_activateDpSlave** function in a MotionTask.

Example program for the synchronous call for the _activateDpSlave function

```
INTERFACE
    PROGRAM motion_1;

    VAR_GLOBAL
      motion_1_RequestActivate : DINT := 0;
      motion_1_ResultActivate  : DINT := 0;
      motion_1_LogAddrDpSlave  : DINT := 16#FFFF_FFFF;
    END_VAR
END_INTERFACE

IMPLEMENTATION
PROGRAM motion_1 // Assigned to a MotionTask
    VAR
        retVal : DINT := 0;
    END_VAR
    // ... Further instructions
    IF ( 0 <> motion_1_RequestActivate ) THEN
        retVal := _activateDpSlave (
            logicalAddressOfDpStation := motion_1_LogAddrDpSlave,
            reqActDeactGetStateMode := REQUEST_TRUE,
            dpAlarmMode := SET_DP_ALARM,
            nextCommand := WHEN_COMMAND_DONE,
            timeToWaitForStationAlarm := 30 // 30 seconds
            (* As of SIMOTION Kernel version V4.4
               Waiting time for station ramp-up *)
            );
        CASE retVal OF
            16#0000_0001: motion_1_RequestActivate := 0;
                // Node successfully activated.
            16#0000_0005: motion_1_RequestActivate := 0;
                // Node was already activated, no alarm.
        ELSE // Troubleshooting
            motion_1_RequestActivate := 0;
        END_CASE;
        IF ( 0 = motion_1_RequestActivate ) THEN
            // Function completed.
            motion_1_ResultActivate := retVal;
        END_IF;
    END_IF;
    // ... Further instructions
END_PROGRAM
END_IMPLEMENTATION
```

Querying activation process and status of the nodes (DP slaves or IO devices)

For the PROFIBUS and PROFINET IO nodes (DP slaves, IO devices) you can:

- Query the status of the activation process using system functions (Page 1039)
- Query the status using system functions (Page 1040)

Note

The freely available and SIEMENS-supported PRONETA tool can be used to set and configure the PROFINET communication addresses. Associated information and the download are available:

- on the Internet (http://w3.siemens.com/mcms/automation/en/industrial-communications/profinet/productportfolio/proneta/Pages/proneta.aspx).
- in SIOS (https://support.industry.siemens.com/cs/document/67460624/proneta-2-3-0-26-commissioning-and-diagnostics-tool-for-profinet?dti=0&lc=en-WW):

Querying the status of the activation process of the DP slaves and IO devices using system functions

System function

The **_getStateOfDpSlave** system function is used to specifically query the status of the activation or deactivation procedure of a DP slave or IO device, which is defined via its logical diagnostics address.

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Functions (Page 1041) are available for converting a PROFIBUS address or device number for PROFINET IO into the logical diagnostics address and vice versa.

Status diagram, asynchronous call and synchronous call

As regards the main features, the same information applies as for the _activateDpSlave and _deactivateDpSlave functions, see:

- Status diagram for the functions _activateDpSlave and _deactivateDpSlave (Page 1034)
- Asynchronous call for the functions _activateDpSlave and _deactivateDpSlave (Page 1035)
- Synchronous call for the functions _activateDpSlave and _deactivateDpSlave (Page 1037)

Querying the status of the DP slaves and IO devices using system functions

System functions

The system functions **_getStateOfAllDPStations**, **_getStateOfAllDpSlaves**, and **_getStateOfSingleDpSlave** can be used to query the status of all or just one DP slave or IO device in a user program. In addition to the activation status, information on availability is also provided.

- With the **_getStateOfAllDPStations** function you obtain:
  - For **PROFIBUS**: the status of all DP slaves configured on a DP master with the logical diagnostics addresses of their PROFIBUS interfaces. You can specify the DP master using the logical diagnostics address for its PROFIBUS interface.
  - For **PROFINET IO**: the status of all IO devices that you can reach from a controller (also via routers), as well as the logical diagnostics addresses for their PROFIBUS interfaces. You can specify the controller using the logical diagnostics address for its PROFINET IO interface.

  With one call of the function, you obtain the status for a maximum of 10 DP slaves or IO devices. If more DP slaves and/or I/O devices are available, you will receive the value 16#0000_0001 in the functionResult component of the return value. Following this, keep calling up the function using the parameter RequestMode := REQUEST_FALSE as often as is necessary until the functionResult component of the return value contains the value 16#0000_0000. Use the same command ID as for the first call.
- With the **_getStateOfAllDpSlaves** function you obtain:
  - For **PROFIBUS**: for every legal PROFIBUS address on a DP master, whether a DP slave is configured for this address and, if necessary, its status. You can specify the DP master using the logical diagnostics address for its PROFIBUS interface.
    This function exists only with PROFIBUS DP.
- With the **_getStateOfSingleDpSlave** function you obtain:
  - For **PROFIBUS**: the status of an individual DP slave that you can specify using the logical diagnostics address for its PROFIBUS interface.
  - For **PROFINET IO**: the status of an individual IO device that you can specify using the logical diagnostics address for its PROFINET IO interface.

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" Parameter Manual (reference list) and in the online help (see index).

Functions (Page 1041) are available for converting a PROFIBUS address or device number for PROFINET IO into the logical diagnostics address and vice versa.

Status diagram, asynchronous call and synchronous call

As regards the main features, the same information applies as for the _activateDpSlave and _deactivateDpSlave functions, see:

- Status diagram for the functions _activateDpSlave and _deactivateDpSlave (Page 1034)
- Asynchronous call for the functions _activateDpSlave and _deactivateDpSlave (Page 1035)
- Synchronous call for the functions _activateDpSlave and _deactivateDpSlave (Page 1037)

Converting a PROFIBUS address or device number for PROFINET IO into the logical diagnostics address and vice versa

The PROFIBUS address and device number for PROFINET IO are unique in the related PROFIBUS subnet or PROFINET IO segment, but not across the project. Only the logical diagnostics addresses are unique across the project.

Using the **_getLogDiagnosticAddressFromDpStationAddress** function you can convert the PROFIBUS address for a bus node (or device number for PROFINET IO) into the associated logical diagnostics address in the user program by specifying the related local interface.

Using the **_getDpStationAddressFromLogDiagnosticAddress** function you can convert the logical diagnostics address into the PROFIBUS address for a bus node (or device number for PROFINET IO), as well as the related local interface, in the user program.

By executing the **_getGeoAddressFromLogAddress** function you will obtain, in the user program, the geographical address for a logical diagnostics address across the project (e.g. bus system, slot number).

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

4.1.5.2 Activating and deactivating SIMATIC ET200 components (option handling)

Introduction

The activation and deactivation of SIMATIC ET200 components is referred to below as "configuration control".

Configuration control (option handling) is used to operate various standard machine configuration levels in one project without changing the configuration or the user program.

Detailed information is provided in the *SIMATIC ET 200SP Distributed I/O System* System Manual.

Operating principle of configuration control

You can use configuration control to operate different standard machine configurations with a single configuration of the ET 200SP distributed I/O system.

- A station master is configured in a project (maximum configuration). The station master comprises all modules needed for all possible plant parts of a modular standard machine.
- The project's user program provides for several station options for various standard machine configuration levels as well as selection of a station option. A station option uses, for example, only some of the configured modules of the station master and these modules are inserted in the slots in a different order.
- The standard machine manufacturer selects a station option for a configuration of the standard machine. To do this, the project need not be modified, and it is not necessary to load a modified configuration.

You use a control data record you have programmed to notify the CPU/interface module as to which modules are missing or located on different slots in a station option as compared to the station master. The configuration control does not have an impact on the parameter assignment of the modules.

Configuration control allows you to flexibly vary the centralized/distributed configuration. This is only possible if the station option can be derived from the station master.

The following figure shows three configurations of a standard machine with the corresponding station options of the ET 200SP distributed I/O system.

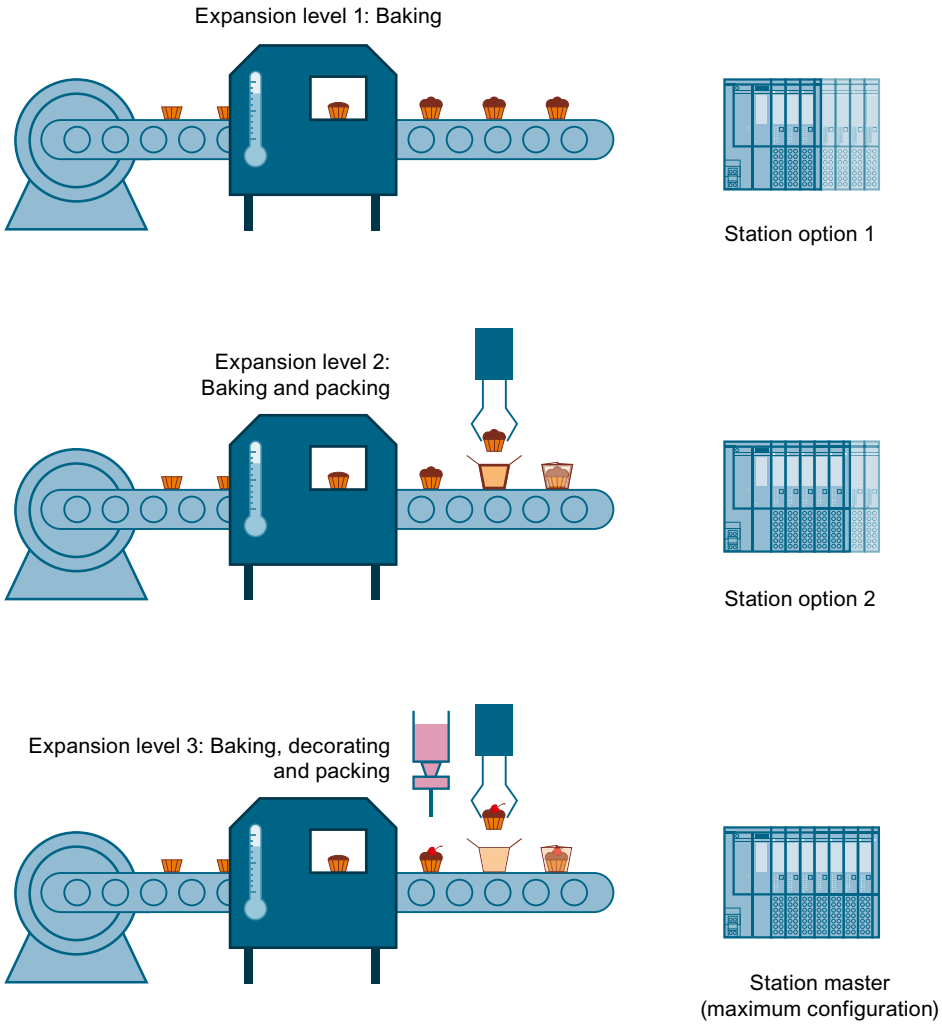


Figure 4-16 Various configuration levels of a standard machine with the corresponding station options of the ET 200SP distributed I/O system.

Advantages

- Simple project management and commissioning by using a single STEP 7 project for all station options.
- Simple handling for maintenance, versioning and upgrade:
- Hardware savings: Only those I/O modules are installed that are required for the machine's current station option.

- Savings potential in the creation, commissioning and the documentation for standard machines.
- Simple station expansion by using pre-wired empty slots. To expand, you simply exchange the BU cover for the new model.

Procedure

To set up the configuration control, follow these steps:

Step	Procedure	See...
1	Enable configuration control in STEP 7	Section Configuring (Page 1043)
2	Create control data record	Section Creating the control data record (Page 1045)
3	Transfer control data record	Section Transferring the control data record in the startup program of the CPU (Page 1055)

Application examples

A number of application examples are available to download (https://support.industry.siemens.com/cs/#document/29430270?lc=en-WW) from the Internet.

Block library "OH_S71x00_Library"

The block library OH_S71x00_Library is available to download (https://support.industry.siemens.com/cs/#document/29430270?lc=en-WW) from the Internet. The block library contains data types with the structure of the control data records for the ET 200SP distributed I/O system. These data types will make it easy for you to integrate your flexible automation solution into the SIMOTION system.

Configuring

Requirements

Configuration control is supported by the ET 200SP distributed I/O system with both an ET 200SP CPU and with interface modules via PROFINET IO and PROFIBUS DP.

Centrally for ET 200SP CPU:

- STEP 7 Professional V13 Update 3 or higher
- CPU 1510SP-1 PN/CPU 1512SP-1 PN

- Firmware version V1.6 or higher
- All modules of the CPU must be able to start up even with different configurations.
  - The startup parameter "Comparison preset to actual configuration" of the CPU is set to "Startup CPU even if mismatch" (default setting) and the module parameter "Comparison preset to actual module" of the module is set to "From CPU" (default setting).
    or
  - The module parameter "Comparison preset to actual module" for the module is set to "Startup CPU even if mismatch".

Distributed via PROFINET IO:

- Engineering Tool (e.g. STEP 7)
- IM 155-6 PN BA/ST/HF
- You have assigned the interface module to an IO controller.

Distributed via PROFIBUS DP:

- Engineering Tool (e.g. STEP 7)
- IM 155-6 DP HF
- You have assigned the interface module to a DP master.
- The startup parameter is set to "Operate if preset configuration does not match actual configuration"

Required steps

Enable the "Allow to reconfigure the device via the user program" parameter when configuring the CPU/interface module in STEP 7 (TIA Portal).

- The "Allow to reconfigure the device via the user program" parameter is located in the "Configuration control" area for an ET 200SP CPU.
- The "Allow to reconfigure the device via the user program" parameter is located in the "Module parameter" area under "General" for an IM 155-5 PN interface module.

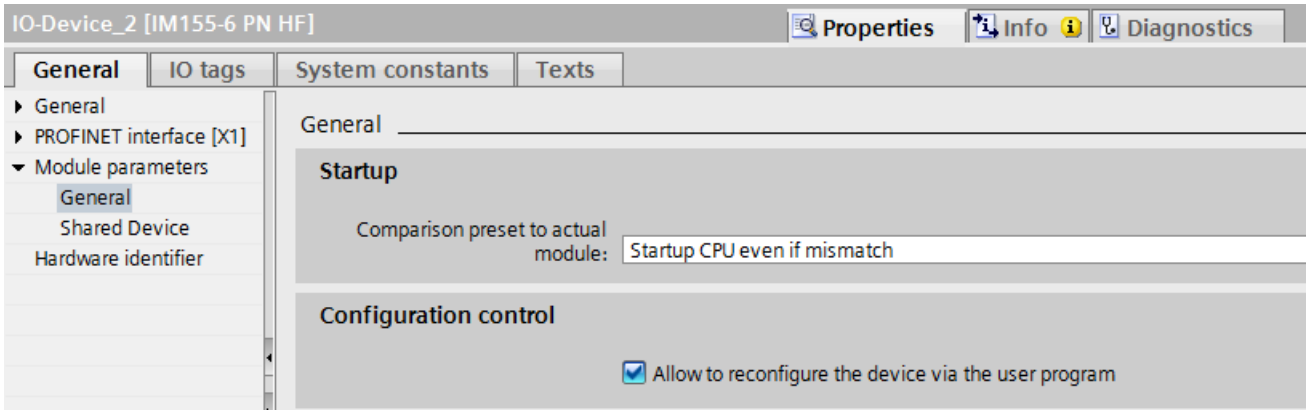


Figure 4-17 Enabling configuration control using an IM 155-6 PN HF as an example

You can find more information on the Internet (https://support.industry.siemens.com/cs/#document/29430270?lc=en-WW).

Creating the control data record

Introduction

Required steps

To create a control data record for the configuration control, follow these steps:

1. Create a PLC data type which contains the structure of the control data record.
   The following figure shows a PLC data type "CTR_REC", which contains the structure of the control data record for an ET 200SP interface module.

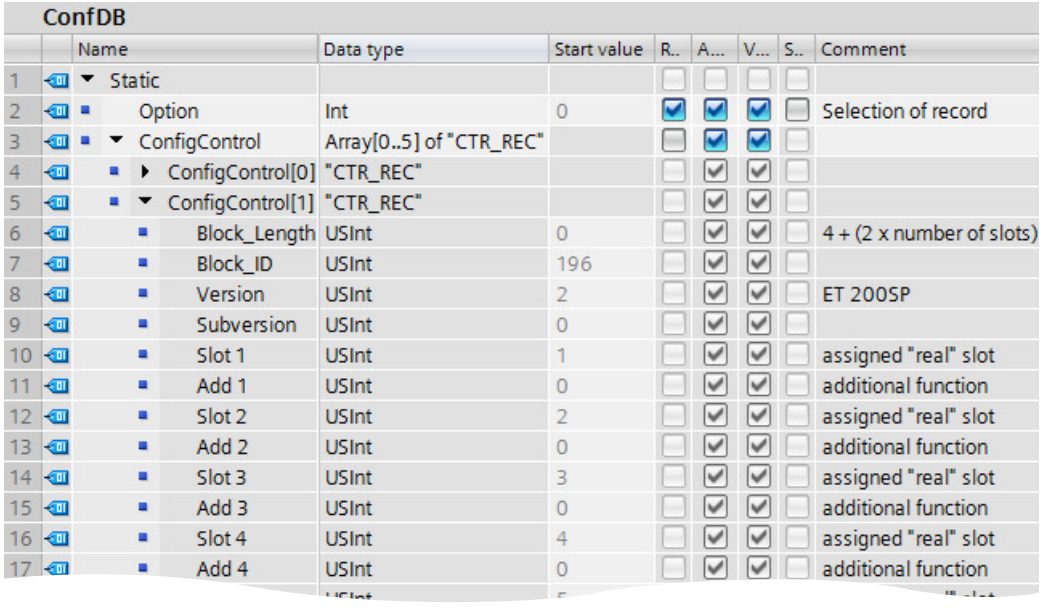
CTR_REC

	Name	Data type	Default value	A...	V...	S...	Comment
1	Block_Lenght	USInt	134	☑	☑	☐	4 + (2 x number of Slots)
2	Block_ID	USInt	196	☑	☑	☐	
3	Version	USInt	2	☑	☑	☐	ET 200SP
4	Subversion	USInt	0	☑	☑	☐	
5	Slot 1	USInt	1	☑	☑	☐	assigned "real" slot
6	Add 1	USInt	0	☑	☑	☐	additional function
7	Slot 2	USInt	2	☑	☑	☐	assigned "real" slot
8	Add 2	USInt	0	☑	☑	☐	additional function
9	Slot 3	USInt	3	☑	☑	☐	assigned "real" slot
10	Add 3	USInt	0	☑	☑	☐	additional function
11	Slot 4	USInt	4	☑	☑	☐	assigned "real" slot
12	Add 4	USInt	0	☑	☑	☐	additional function

Figure 4-18 Creating control data record 196 using an IM 155-6 PN HF as an example

2. Create a global data block.

3. In the data block, create an array that is based on the created PLC data type.
4. In the control data records, enter the slot assignments in the "Start value" column.
   The figure below shows the global data block "ConfDB". The data block "ConfDB" contains an array [0..5] of the PLC_DataType "CTR_REC".



ConfDB

	Name	Data type	Start value	R..	A...	V...	S..	Comment
1	Static			☐	☐	☐	☐	
2	Option	Int	0	☑	☑	☑	☐	Selection of record
3	ConfigControl	Array[0..5] of "CTR_REC"		☐	☑	☑	☐	
4	ConfigControl[0]	"CTR_REC"		☐	☑	☑	☐	
5	ConfigControl[1]	"CTR_REC"		☐	☑	☑	☐	
6	Block_Length	USInt	0	☐	☑	☑	☐	4 + (2 x number of slots)
7	Block_ID	USInt	196	☐	☑	☑	☐	
8	Version	USInt	2	☐	☑	☑	☐	ET 200SP
9	Subversion	USInt	0	☐	☑	☑	☐	
10	Slot 1	USInt	1	☐	☑	☑	☐	assigned "real" slot
11	Add 1	USInt	0	☐	☑	☑	☐	additional function
12	Slot 2	USInt	2	☐	☑	☑	☐	assigned "real" slot
13	Add 2	USInt	0	☐	☑	☑	☐	additional function
14	Slot 3	USInt	3	☐	☑	☑	☐	assigned "real" slot
15	Add 3	USInt	0	☐	☑	☑	☐	additional function
16	Slot 4	USInt	4	☐	☑	☑	☐	assigned "real" slot
17	Add 4	USInt	0	☐	☑	☑	☐	additional function

Figure 4-19 Data block for configuration control

Rules

Observe the following rules:

- Slot entries in the control data record outside the station master are ignored by the CPU/ interface module.
- The control data record must contain the entries up to the last slot of the station option.
- Multiple configured slots may not be assigned to the same actual slot. In other words, each station option slot may be present only once in the control data record.

Control data record for an ET 200SP CPU

Slot assignment

The following table shows the possible slots for the various modules for an ET 200SP CPU:

Table 4-3 Slot assignment

Modules	Possible slots	Comment
CPU	1	Slot 1 is always the CPU
I/O modules	2 - 65	Downstream of CPU
Server module	2 - 66	The server module completes the configuration of the ET 200SP station after the CPU/the last I/O module.

Control data record

For the configuration control of an ET 200SP CPU, you define a control data record 196 V2.0, which includes a slot assignment. The maximum slot corresponds to the slot of the server module.

The table below shows the structure of a control data record with explanations of the individual elements.

Table 4-4 Configuration control: Structure of control data record 196

Byte	Element	Code	Explanation
0	Block length	4 + (number of slots × 2)	Header
1	Block ID	196	
2	Version	2	
3	Version	0	

Byte	Element	Code	Explanation
4	Slot 1 of the station master	Slot assignment 1 in the station option (always 1, because the CPU is always in slot 1)	**Control element** Contains the information on which module is inserted in which slot. The value that you need to enter in the corresponding byte depends on the following rule: • If the module exists in the station option, enter the slot number of the module. • If the module exists as empty slot (with BU cover), enter the slot number of the module + 128. (Example: module as empty slot on slot 3: Enter 131 in the control element) • If the module does not exist in the station option, enter 0. **Additional function** Contains information on whether a new potential group will be opened in the station option - by replacing a dark-colored BaseUnit with a light-colored BaseUnit. • If you replace a dark-colored BaseUnit with a light-colored BaseUnit, enter 1 as additional function. • If you accept the BaseUnit from the station master, enter 0 as additional function.
5	Additional function for slot 1		
6	Slot 2 of the station master	Slot assignment in the station option	
7	Additional function for slot 2		
8	Slot 3 of the station master	Slot assignment in the station option	
9	Additional function for slot 3		
:	:	:	
4 + ((max. slot - 1) × 2)	Server module slot	Server module slot assignment in the station option*	
4 + ((max. slot - 1) × 2) + 1	Additional function for server module slot		

* The server module must be present in the station option and must not be marked as empty slot (BU cover).

Control data record for an interface module

Slot assignment

The following table shows the possible slots for the various modules for an ET 200SP interface module:

Table 4-5 Slot assignment

Modules	Possible slots	Comment
Interface module	0	The interface module (slot 0) is not an element of the configuration control, but instead controls this.
Station extension BA-Send	1	For mixed configuration with ET 200AL modules, BA-Send is always on slot 1.

Modules	Possible slots		Comment
ET 200SP I/O module	1 - 12	for IM 155-6 PN BA	Downstream from the interface module
	1 - 32	for IM 155-6 PN ST, IM 155-6 DP HF	
	1 - 64	for IM 155-5 PN HF	
Server module	1 - 13	for IM 155-6 PN BA	The server module completes the configuration of the ET 200SP station after the last I/O module.
	1 - 33	for IM 155-6 PN ST, IM 155-6 DP HF	
	1 - 65	for IM 155-5 PN HF	
ET 200AL I/O module	34 - 49	for IM 155-6 DP HF	For mixed configuration with ET 200AL modules
	66 - 81	for IM 155-6 PN ST, IM 155-6 PN HF	

Simplified control data record (V1)

For the configuration control of interface modules of the ET 200SP distributed I/O system, you define a control data record 196 V1.0, which includes a slot assignment. The maximum slot of the configuration corresponds to the slot of the server module or ET 200AL I/O module (in a mixed ET 200SP / ET 200AL configuration).

The table below shows the structure of a control data record with explanations of the individual elements.

Table 4-6 Structure of the simplified control data record V1.0

Byte	Element	Code	Explanation
0	Block length	4 + maximum slot	Header
1	Block ID	196	
2	Version	1	
3	Version	0	
4	Slot 1 of the station master	Slot assignment in the station option	**Control element ET 200SP** Contains the information on which ET 200SP module is inserted in which slot. The value that you need to enter in the corresponding byte depends on the following rule: • If the module exists in the station option, enter the slot number of the module. • If the module exists as empty slot (with BU cover), enter the slot number of the module + 128. (Example: module as empty slot on slot 3: Enter 131 in the control element) • If the module does not exist in the station option, enter 0.
5	Slot 2 of the station master	Slot assignment in the station option	
:	:	:	
4 + (slot server module - 1)	Server module slot	Assignment of server module slot in the station option*	
:	:	:	:

Byte	Element	Code	Explanation
4 + (first slot ET 200AL - 1)	First slot ET 200AL	Slot assignment in the station option	**Control element ET 200AL** Contains information on which ET 200AL module is inserted in which slot. The value that you need to enter in the corresponding byte depends on the following rule: • If the module exists in the station option, enter the slot number of the module. • If the module does not exist in the station option, enter 0.
:	:	:	
4 + (last slot ET 200AL - 1)	Last slot ET 200AL	Slot assignment in the station option	

* The server module must be present in the station option and must not be marked as empty slot (BU cover).

Control data record (V2)

If you change the potential groups in the station option compared to the station master, define a control data record 196 V2.0 for the ET 200SP interface module which contains a slot assignment. The maximum slot of the configuration corresponds to the slot of the server module or ET 200AL I/O module (in a mixed ET 200SP / ET 200AL configuration).

The table below shows the structure of a control data record with explanations of the individual elements.

Table 4-7 Structure of control data record 196 V2.0

Byte	Element	Code	Explanation
0	Block length	4 + (maximum slot x 2)	Header
1	Block ID	196	
2	Version	2	
3	Version	0	

<table>
<tr><th>Byte</th><th>Element</th><th>Code</th><th>Explanation</th></tr>
<tr><td>4</td><td>Slot 1 of the station master</td><td>Slot assignment in the station option</td><td rowspan="11">Control element ET 200SP
Contains the information on which ET 200SP module is inserted in which slot.
The value that you need to enter in the corresponding byte depends on the following rule:
• If the module exists in the station option, enter the slot number of the module.
• If the module exists as empty slot (with BU cover), enter the slot number of the module + 128. (Example: module as empty slot on slot 3: Enter 131 in the control element)
• If the module does not exist in the station option, enter 0.
Additional function
Contains information on whether a new potential group will be opened in the station option - by replacing a dark-colored BaseUnit with a light-colored BaseUnit.
• If you replace a dark-colored BaseUnit with a light-colored BaseUnit, enter 1 as additional function.
• If you accept the BaseUnit from the station master, enter 0 as additional function.</td></tr>
<tr><td>5</td><td colspan="2">Additional function for slot 1</td></tr>
<tr><td>6</td><td>Slot 2 of the station master</td><td>Slot assignment in the station option</td></tr>
<tr><td>7</td><td colspan="2">Additional function for slot 2</td></tr>
<tr><td>8</td><td>Slot 3 of the station master</td><td>Slot assignment in the station option</td></tr>
<tr><td>9</td><td colspan="2">Additional function for slot 3</td></tr>
<tr><td>:</td><td>:</td><td>:</td></tr>
<tr><td>4 + ((server module slot - 1) × 2)</td><td>Server module slot</td><td>Server module slot assignment in the station option*</td></tr>
<tr><td>4 + ((server module slot - 1) × 2) + 1</td><td colspan="2">Additional function for server module slot</td></tr>
<tr><td>:</td><td>:</td><td>:</td><td>:</td></tr>
<tr><td>4 + ((first slot ET 200AL - 1) x 2)</td><td>First slot ET 200AL</td><td>Slot assignment in the station option</td><td rowspan="5">Control element ET 200AL
Contains information on which ET 200AL module is inserted in which slot.
The value that you need to enter in the corresponding byte depends on the following rule:
• If the module exists in the station option, enter the slot number of the module.
• If the module does not exist in the station option, enter 0.</td></tr>
<tr><td>4 + ((first slot ET 200AL - 1) x 2) + 1</td><td colspan="2">Reserved</td></tr>
<tr><td>:</td><td>:</td><td>:</td></tr>
<tr><td>4 + ((last slot ET 200AL - 1) x 2)</td><td>Last slot ET 200AL</td><td>Slot assignment in the station option</td></tr>
<tr><td>4 + ((last slot ET 200AL - 1) x 2) + 1</td><td colspan="2">Reserved</td></tr>
</table>

* The server module must be present in the station option and must not be marked as empty slot (BU cover).

Note

If a BU cover or no I/O module is plugged on a light-colored BaseUnit, you should enter 1 in the additional function for the slot.

The function "Group diagnostics: Missing supply voltage L+" requires proper assignment of the slots to a shared supply voltage L+ (potential group). All light-colored BaseUnits must be known to the interface module. By entering 1 in the additional function, you make a light-colored BaseUnit known to the interface module, even if an I/O module is not inserted.

Combination of configuration control and shared device (for PROFINET)

The configuration control function in a shared device is therefore only for the I/O modules of the IO controller to which the interface module has subscribed. I/O modules that are assigned to no controller or a different controller behave like a station without activated configuration control.

You cannot make any change to the slot assignment for modules that are assigned to another IO controller or are not assigned to an IO controller (shared device on module level). The CPU assumes a one-to-one assignment for the modules.

If additional IO controllers subscribe to a module intended for configuration control (shared device on submodule level), only one-to-one assignment is permitted for this module. It is not possible to deselect such a module using the control data record (code 0 for this slot in the control data record). This means the combination of "Configuration control" and "Shared device on submodule level" is only possible to a limited extent.

Please note that all modules affected by the configuration control including all assigned submodules are reset when you change the module assignment. Submodules that are assigned to a second IO controller are affected as well.

Feedback data record for interface modules

Operating principle

The feedback data record informs you about the accuracy of the module assignment and gives you the option of detecting assignment errors in the control data record. The feedback data record is mapped via a separate data record 197 V2.0. The feedback data record exists only with configured configuration control.

Slot assignment

The feedback data record refers to the configured station configuration and always includes the maximum configuration limits. The maximum configuration limits comprise 13/49/81 slots depending on the interface module in use. Partial reading of the feedback data record is possible.

The following table shows the slot assignment of the modules:

Table 4-8 Slot assignment

Modules	Possible slots		Comment
Station extension BA-Send	1		For mixed configuration with ET 200AL modules, BA-Send is always on slot 1.
ET 200SP I/O module	1 - 12	for IM 155-6 PN BA	Downstream from the interface module
	1 - 32	for IM 155-6 PN ST, IM 155-6 DP HF	
	1 - 64	for IM 155-5 PN HF	
Server module	1 - 13	for IM 155-6 PN BA	The server module completes the configuration of the ET 200SP station after the last I/O module.
	1 - 33	for IM 155-6 PN ST, IM 155-6 DP HF	
	1 - 65	for IM 155-5 PN HF	
ET 200AL I/O module	34 - 49	for IM 155-6 DP HF	For mixed configuration with ET 200AL modules
	66 - 81	for IM 155-6 PN ST, IM 155-6 PN HF	

Feedback data record

Table 4-9 Feedback data record

Byte	Element	Code	Explanation
0	Block length	4 + (number of slots x 2)	Header
1	Block ID	197	
2	Version	2	
3		0	
4	Slot 1 status	0/1	Status = 1: • Module from station master is inserted in the station option • Slot is marked as not available in the control data record Status = 0: • Module pulled • Incorrect module inserted in the station option*
5	Reserved	0	
6	Slot 2 status	0/1	
7	Reserved	0	
:	:	:	
4 + ((max. slot - 1) × 2)	Max. slot status	0/1	
4 + ((max. slot - 1)× 2) + 1	Reserved	0	

* Not possible if the slot is marked as not available.

Note

The data in the feedback data record is always mapped for all modules. In a shared device configuration, it is therefore irrelevant which IO controller the respective modules are assigned to.

As long as no control data record has been sent, a one-to-one module assignment is assumed for the compilation of data record 197 (station master → station option).

Error messages

In case of error, the RDREC instruction returns the following error messages via the STATUS block parameter while reading the feedback data record:

Table 4-10 Error messages

Error code	Meaning
$80B1_H$	Invalid length; the length information in data record 197 is not correct.
$80B5_H$	Configuration control not configured
$80B8_H$	Parameter error The following events cause a parameter error: • Incorrect block ID in the header (not equal to 197) • Invalid version identifier in the header • A reserved bit was set • Multiple slots in the station master are assigned to the same slot in the station option

Data records and functions

Supported data records and functions

The table below shows a comparison of the supported data records and functions depending on the CPU/interface module used.

	CPU...		Interface module (IM...)			
Supported data records and functions	**1510SP-1 PN 1510SP F-1 PN**	**1512SP-1 PN 1512SP F-1 PN**	**155-6 PN HF**	**155-6 PN ST**	**155-6 PN BA**	**155-6 DP HF**
Control data record (V2)	✓	✓	✓	✓	✓	✓
Simplified control data record (V1)	--	--	✓	✓	✓	--
Read back control data record *	✓	✓	✓	✓	✓	✓
Read feedback data record	--	--	✓	✓	✓	✓

* You can read back the control data record with the RDREC instruction.

Transferring the control data record to the ET200SP

The example program shown below creates the control data record for the example options in section Examples of configuration control (Page 1059) and transfers it to the ET200SP. The program must be assigned to the BackgroundTask.

```
INTERFACE
  PROGRAM pConfigControlET200SP_Station1;

  TYPE
    eConfigControlET200SPoptionType : (STATION_OPTION_1, STATION_OPTION_2,
                                       STATION_OPTION_3, STATION_OPTION_4);
  END_TYPE

  VAR_GLOBAL
  gboConfigControlET200SPDataRecordWrittenStation1 : BOOL;
       // TRUE: control data record written
  END_VAR
  VAR_GLOBAL RETAIN
    greConfigControlET200SPNextOptionStation1 : eConfigControlET200SPoptionType;
        // option to be activated with next restart (STOP -> RUN transition)
  END_VAR
END_INTERFACE
IMPLEMENTATION
  TYPE
    sET200SPslotType : STRUCT
        u8ControlElement      : USINT;
            // 0: module does not exist,
            // (slot number): module exists,
            // (slot number + 128): module exists as empty slot (with BU cover)
        u8AdditionalFunction  : USINT;
            // 0: same potential group as configured,
            // 1: open a new potential group (light-colored BaseUnit)
    END_STRUCT;
  END_TYPE

  VAR_GLOBAL CONSTANT
    MODULE_DOES_NOT_EXIST              : USINT := 0;
    SAME_POTENTIAL_GROUP_AS_CONFIGURED : USINT := 0;
    OPEN_NEW_POTENTIAL_GROUP           : USINT := 1;
  END_VAR
```

```
PROGRAM pConfigControlET200SP_Station1
  //this program is designed to be called in BackgroundTask
  VAR CONSTANT
    MAX_NUMBER_OF_SLOTS : USINT := 32;
        // e.g. 32: IM 155-6 PN ST, 64: IM 155-5 PN HF, ...
    LOG_ADDRESS_OF_ET200SP : DINT := 16369;
        // corresponding logical base address of the I/O module (ET200SP)
  END_VAR

  TYPE
    sET200SPcontrolDataRecordV2Type : STRUCT
        u8BlockLength : USINT := 4 + (MAX_NUMBER_OF_SLOTS * 2);
            // 4 + (maximum number of slots * 2) bytes
        u8BlockID : USINT := 196;
        u8Version : USINT := 2;
        u8SubVersion : USINT := 0;
        asET200SPslot : ARRAY [1..MAX_NUMBER_OF_SLOTS] OF sET200SPslotType;
    END_STRUCT;
  END_TYPE

  VAR
    sET200SPcontrolDataRecordV2 : sET200SPcontrolDataRecordV2Type;
    ab8Record : ARRAY [0..(4 + (MAX_NUMBER_OF_SLOTS * 2) - 1)] OF BYTE;
    FBwriteVariableRecord : _writeVariableRecord;
    boRestart : BOOL := TRUE;
        // start writing the control data record
        // (variable will be initialized at STOP -> RUN transition)
    boExecute : BOOL;
  END_VAR
  IF boRestart THEN
      // STOP -> RUN transition -> write control data record to ET200 SP station
    gboConfigControlET200SPDataRecordWrittenStation1 := FALSE;
        // reset "written" flag
    CASE greConfigControlET200SPNextOptionStation1 OF
        eConfigControlET200SPoptionType#STATION_OPTION_1:
              // Station option 1 with module that is not present
              // The module that is located in slot 3 in the station master
              // is not present in the station option 1.
              //Slot 3 must be designated in the control data record
              // accordingly with 0 (= not present). The server module
              // is located in slot 3 in the station option.
            sET200SPcontrolDataRecordV2.asET200SPslot[1].u8ControlElement := 1;
            sET200SPcontrolDataRecordV2.asET200SPslot[1].u8AdditionalFunction :=
                SAME_POTENTIAL_GROUP_AS_CONFIGURED;
            sET200SPcontrolDataRecordV2.asET200SPslot[2].u8ControlElement := 2;
            sET200SPcontrolDataRecordV2.asET200SPslot[2].u8AdditionalFunction :=
                SAME_POTENTIAL_GROUP_AS_CONFIGURED;
            sET200SPcontrolDataRecordV2.asET200SPslot[3].u8ControlElement :=
                MODULE_DOES_NOT_EXIST;
            sET200SPcontrolDataRecordV2.asET200SPslot[3].u8AdditionalFunction :=
                SAME_POTENTIAL_GROUP_AS_CONFIGURED;
            sET200SPcontrolDataRecordV2.asET200SPslot[4].u8ControlElement := 3;
            sET200SPcontrolDataRecordV2.asET200SPslot[4].u8AdditionalFunction :=
                SAME_POTENTIAL_GROUP_AS_CONFIGURED;
```

```
    eConfigControlET200SPoptionType#STATION_OPTION_2:
          // Station option 2 with modified order of modules
          // The order of the modules at slots 2 and 3 is interchanged.
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8ControlElement := 1;
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8ControlElement := 3;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8ControlElement := 2;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8ControlElement := 4;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
    eConfigControlET200SPoptionType#STATION_OPTION_3:
          // Station option 3 with empty slot
          // The module that is located in slot 3 in the station master
          // occupies an empty slot with BU cover in the station option.
          // Enter the value 130 in slot 3 in the control data record.
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8ControlElement := 1;
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8ControlElement := 2;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8ControlElement := 131;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8ControlElement := 4;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
    eConfigControlET200SPoptionType#STATION_OPTION_4:
          // Station option 4: Opening a new potential group
          // A new potential group is opened at slot 3 of station option 4.
          // Compared to the station master, a dark-colored BaseUnit
          // is replaced by a light-colored BaseUnit.
          // Enter the value 1 as additional function.
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8ControlElement := 1;
        sET200SPcontrolDataRecordV2.asET200SPslot[1].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8ControlElement := 2;
        sET200SPcontrolDataRecordV2.asET200SPslot[2].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8ControlElement := 3;
        sET200SPcontrolDataRecordV2.asET200SPslot[3].u8AdditionalFunction :=
            OPEN_NEW_POTENTIAL_GROUP;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8ControlElement := 4;
        sET200SPcontrolDataRecordV2.asET200SPslot[4].u8AdditionalFunction :=
            SAME_POTENTIAL_GROUP_AS_CONFIGURED;
ELSE
    ;
END_CASE;
```

```
      // convert control data record to array of byte
      ab8Record := ANYTYPE_TO_BIGBYTEARRAY (
            anyData := sET200SPcontrolDataRecordV2
         // , offset := 0
            );
      //write control data record to ET200SP
      FBwriteVariableRecord (
            EXECUTE := boExecute
          , IOID := INPUT
          , LOGADDR := LOG_ADDRESS_OF_ET200SP
              //logical address from device configuration (e.g. 16369)
          , DSNR := sET200SPcontrolDataRecordV2.u8BlockID
          , LEN := sET200SPcontrolDataRecordV2.u8BlockLength
          , RECORD := ab8Record
         // , DONE =>
         // , BUSY =>
         // , ERROR =>
         // , ERRORID =>
         );
      boExecute := TRUE;
          // needed to create a rising edge
          // at the EXECUTE input of FBwriteVariableRecord
      IF FBwriteVariableRecord.DONE THEN
            //successfully completed
          boRestart := FALSE;
          gboConfigControlET200SPDataRecordWrittenStation1 := TRUE;
              // set "written" flag
      ELSIF FBwriteVariableRecord.ERROR THEN
            // error handling ...
          boExecute := FALSE;
              // try to write the control data record again
      END_IF;
    END_IF;
  END_PROGRAM
END_IMPLEMENTATION
```

Behavior during operation

Effect of discrepancy between station master and station option

For the online display and for the display in the diagnostics buffer (module OK or module faulty), the station master is always used and not the differing station option.

Example: A module supplies diagnostic information. This module is configured in slot 4 in the station master, but is inserted in slot 3 in the station option (missing module; see example in the next section). The online view (station master) shows a faulty module in slot 4. In the real configuration, the module in slot 3 indicates an error via an LED display.

Response when modules are missing

If modules are entered as not present in the control data record, the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostics and their status is always OK. The value status is OK.
- Direct write access to the outputs that are not present or write access to the process image of the outputs that are not present: Remains without effect; no access error is signaled.
- Direct read access to the inputs that are not present or read access to the process image of the inputs that are not present: Value "0" is supplied; no access error is signaled.
- Write data record to module that is not present: Remains without effect; no error is signaled.
- Read data record from module that is not present: An error is signaled because a valid data record cannot be returned.

Examples of configuration control

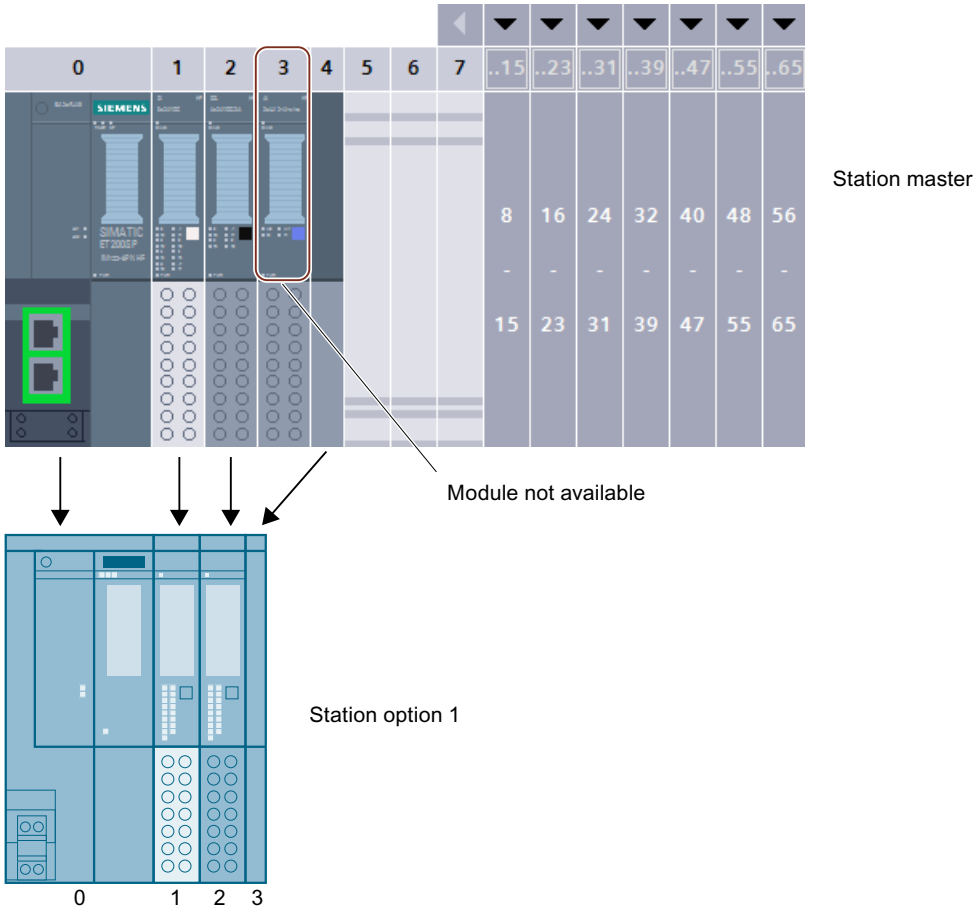
A station master consisting of an interface module, three I/O modules and the server module is configured in STEP 7 in the following section.

Four station options are derived from the station master with the configuration control:

- Station option 1 with module that is not present
- Station option 2 with modified order of modules
- Station option 3 with empty slot
- Station option 4: Opening a new potential group

Station option 1 with module that is not present

The module that is located in slot 3 in the station master is not present in the station option 1. Slot 3 must be designated in the control data record accordingly with 0 (= not present). The server module is located in slot 3 in the station option.

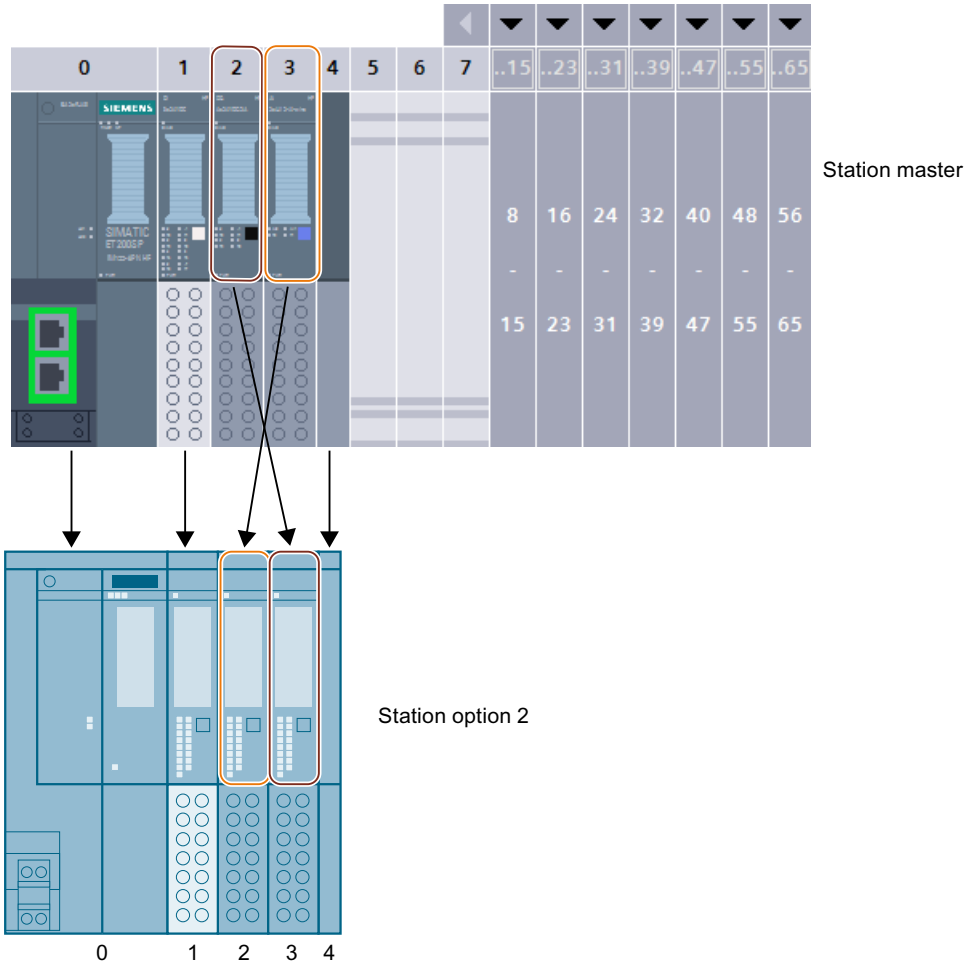


5			ConfigControl[1]	"CTR_REC"			☑	☑	
6			Block_Lenght	USInt	134		☑	☑	4 + (2 x number of slots)
7			Block_ID	USInt	196		☑	☑	
8			Version	USInt	2		☑	☑	ET 200SP
9			Subversion	USInt	0		☑	☑	
10			Slot 1	USInt	1		☑	☑	assigned "real" slot
11			Add 1	USInt	0		☑	☑	additional function
12			Slot 2	USInt	2		☑	☑	assigned "real" slot
13			Add 2	USInt	0		☑	☑	additional function
14			Slot 3	USInt	0		☑	☑	assigned "real" slot
15			Add 3	USInt	0		☑	☑	additional function
16			Slot 4	USInt	3		☑	☑	assigned "real" slot
			Add 4	USInt	0		☑	☑	additional function

Figure 4-20 Example: Hardware configuration of station option 1 with the associated control data record in STEP 7

Station option 2 with modified order of modules

The order of the modules at slots 2 and 3 is interchanged.

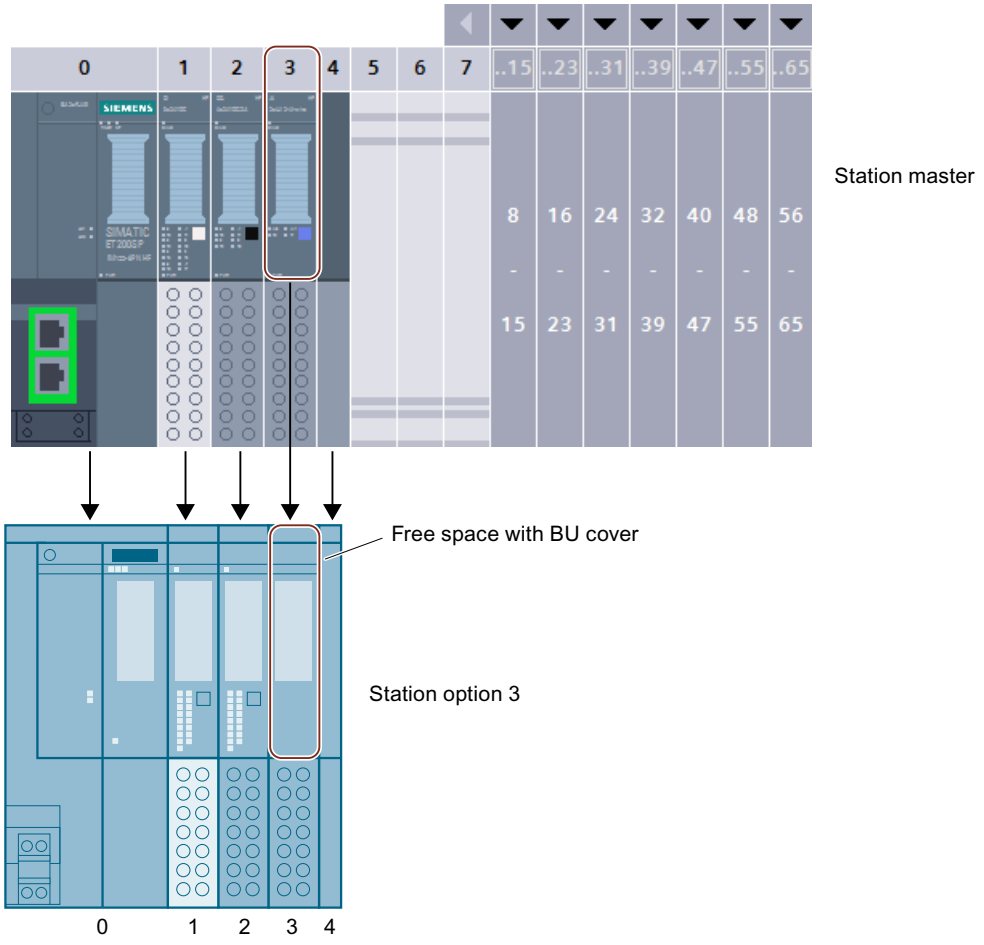


6	ConfigControl[2]	"CTR_REC"			
7	Block_Lenght	USInt	134		4 + (2 x number of slots)
8	Block_ID	USInt	196		
9	Version	USInt	2		ET 200SP
10	Subversion	USInt	0		
11	Slot 1	USInt	1		assigned "real" slot
12	Add 1	USInt	0		additional function
13	Slot 2	USInt	3		assigned "real" slot
14	Add 2	USInt	0		additional function
15	Slot 3	USInt	2		assigned "real" slot
16	Add 3	USInt	0		additional function
17	Slot 4	USInt	4		assigned "real" slot

Figure 4-21 Example: Hardware configuration of station option 2 with the associated control data record in STEP 7

Station option 3 with empty slot

The module that is located in slot 3 in the station master occupies an empty slot with BU cover in the station option. Enter the value 130 in slot 3 in the control data record.

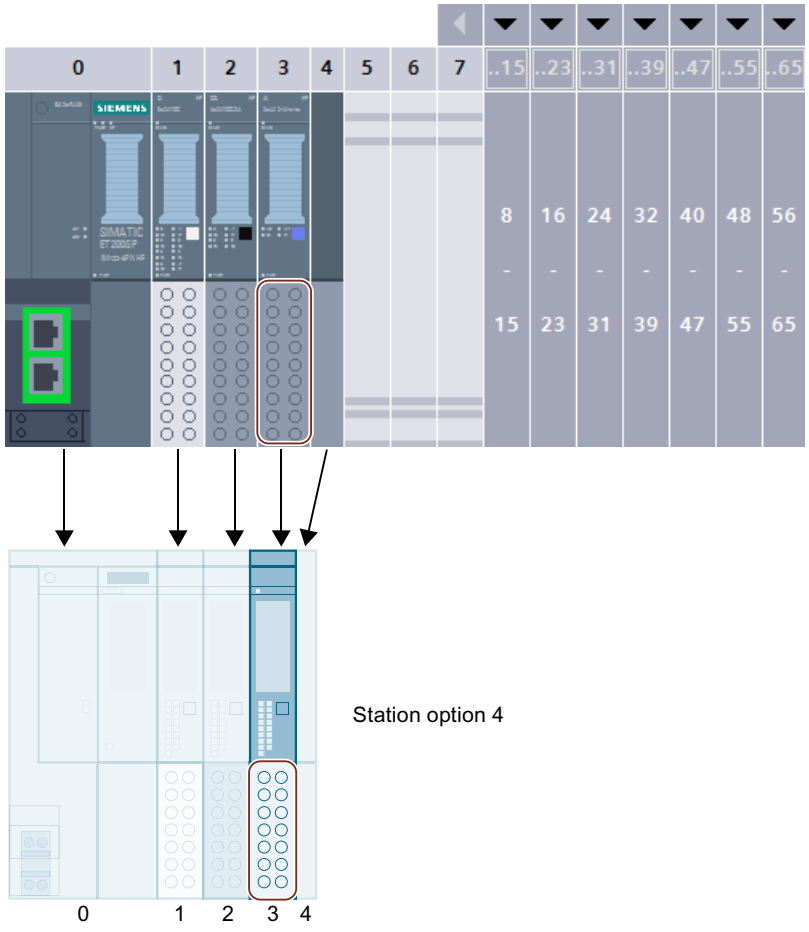


7			ConfigControl[3]	"CTR_REC"					
8			Block_Lenght	USInt	134				4 + (2 x number of slots)
9			Block_ID	USInt	196				
10			Version	USInt	2				ET 200SP
11			Subversion	USInt	0				
12			Slot 1	USInt	1				assigned "real" slot
13			Add 1	USInt	0				additional function
14			Slot 2	USInt	2				assigned "real" slot
15			Add 2	USInt	0				additional function
16			Slot 3	USInt	131				assigned "real" slot
17			Add 3	USInt	0				additional function
			Slot 4	USInt	4				assigned "real" slot

Figure 4-22 Example: Hardware configuration of station option 3 with the associated control data record in STEP 7

Station option 4: Opening a new potential group

A new potential group is opened at slot 3 of station option 4. Compared to the station master, a dark-colored BaseUnit is replaced by a light-colored BaseUnit. Enter the value 1 as additional function.



8		ConfigControl[4]	"CTR_REC"		☐	☑	☑	☐	
9		Block_Lenght	USInt	134	☐	☑	☑	☐	4 + (2 x number of slots)
10		Block_ID	USInt	196	☐	☑	☑	☐	
11		Version	USInt	2	☐	☑	☑	☐	ET 200SP
12		Subversion	USInt	0	☐	☑	☑	☐	
13		Slot 1	USInt	1	☐	☑	☑	☐	assigned "real" slot
14		Add 1	USInt	0	☐	☑	☑	☐	additional function
15		Slot 2	USInt	2	☐	☑	☑	☐	assigned "real" slot
16		Add 2	USInt	0	☐	☑	☑	☐	additional function
17		Slot 3	USInt	3	☐	☑	☑	☐	assigned "real" slot
18		Add 3	USInt	1	☐	☑	☑	☐	additional function
19		Slot 4	USInt	4	☐	☑	☑	☐	assigned "real" slot

Figure 4-23 Example: Hardware configuration of station option 4 with the associated control data record in STEP 7

4.1.5.3 Activating and deactivating SINAMICS components

There is a fundamental difference between how individual components are activated and deactivated within a SINAMICS S120 multi-axis group and the corresponding procedures for, say, I/O components. This is due to the specific architecture involved.

Drives, I/O components in the SINAMICS S120 series, and their control instances are described in a complex parameter object model that is stored in the central control unit. Entire drives, including their control or individual components, can be activated or deactivated there. The following parameters are used for this purpose:

- Activate/deactivate drive objects (also on Terminal Modules or Controller Extension CX32):
  - Adjustable parameter p0105 (Activate/deactivate drive object)
  - Display parameter r0106 (Drive object active/inactive)
- Activate/deactivate power units:
  - Adjustable parameter p0125[PDS] (Activate/deactivate power unit component)
  - Display parameter r0126[PDS] (Power unit components active/inactive)
- Activate/deactivate encoders (on motor modules):
  - Adjustable parameter p0145[EDS] (Activate/deactivate encoder interface)
  - Display parameter r0146[EDS] (Encoder interface active/inactive)
- Activate/deactivate Voltage Sensing Module (on line modules):
  - Adjustable parameter p0145[VSM] (Activate/deactivate Voltage Sensing Module)
  - Display parameter p0146[VSM] (Voltage Sensing Module active/inactive)

There is a description of these parameters in the SINAMICS S120/S150 List Manual.

Note

Pay attention to the boundary conditions for the individual parameters described in the SINAMICS S List Manual.

Example of a sub-topology

The figure below shows an example of a sub-topology.

Parameters have been set on a machine for 1 active line module and 2 motor modules. The related configuration is saved on the CompactFlash Card of the control unit for the SINAMICS drive (reference topology).

- With the machine switched off, "drive 1" is removed and the DRIVE-CLiQ cable from the control unit is connected directly to "drive 2" (sub-topology, actual topology).
- On ramping up, the control unit detects a difference between the reference topology and the actual topology.
- The user program must then execute the following actions:
  - Set the drive object (DO) "drive 1" to "deactivate and not available". For this purpose the program must set the parameter p0105 = 2 in this DO.
  - Save all parameters in non-volatile memory on the CompactFlash Card; for this purpose the program must set the parameter p0977 = 1 in the control unit.

The actual topology is then saved as the reference topology. No further errors are triggered on ramping up.

Note

You can use the same procedure to deactivate all drive objects on a Controller Extension CX32 at the same time. For this purpose, set the parameter p0105 = 2 in the drive object CU_LINK.

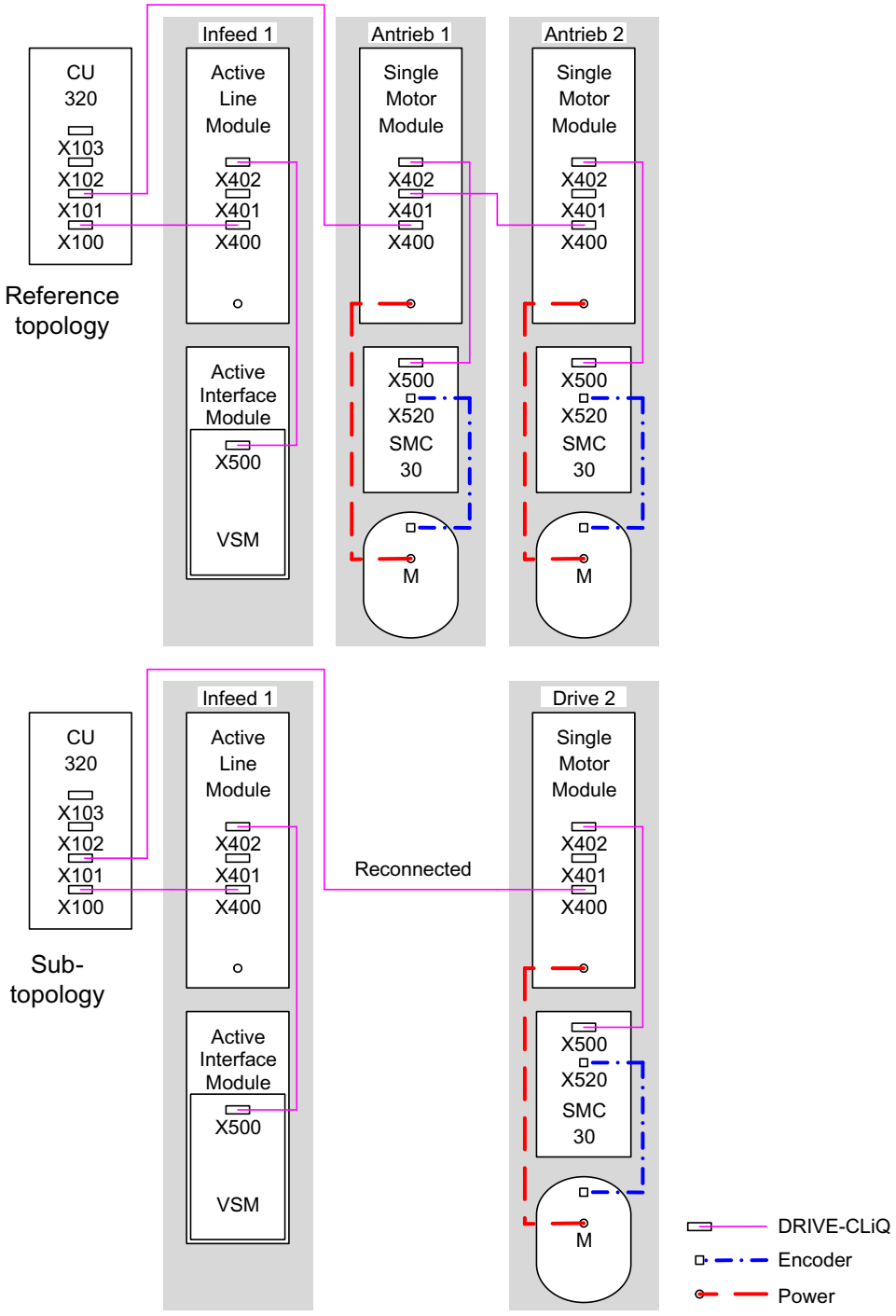


Figure 4-24 Example of a sub-topology

4.1.5.4 Activating and deactivating technology objects

The user has configured and programmed a maximum configuration. Technology objects which are not required (e.g. those assigned to deactivated DP slaves or IO devices) can be deactivated without restarting during operation.

The required technology objects (together with the DP slaves or IO devices to which they are assigned) can also be activated during operation.

This has the following advantages:

- The processor performance is concentrated on the activated technology objects. The computing time available for the MotionTasks and the BackgroundTask is increased.
- The system cycle clocks do not have to be dimensioned for calculating all the technology objects present in the maximum configuration, but only for the maximum number of simultaneously active technology objects. They can then generally be selected with shorter times: This increases the time resolution and reduces any response times; motion control is improved.
- A single project is suitable for various machine configurations; changes to the machine configuration can be made without a restart during operation.
- A default setting can be specified in the project as to which technology objects are active during the transition to the RUN operating mode.

Note

Licenses are required for all axes of the maximum configuration.

System functions

- _activateTo (Page 1067)
- _deactivateTo (Page 1067)
- _getStateOfTo (Page 1074)

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

Technology objects are often assigned to the distributed I/O (DP slave, IO device); therefore, please note the information in Procedure for deactivating and activating components (Page 1075).

System functions for activating and deactivating technology objects

System functions

Use the functions **_deactivateTo** and **_activateTo** to activate and deactivate the technology objects during runtime.

The related technology object is transferred in parameter **TO_Instance** (data type ANYOBJECT).

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index). Please also note the

information relating to the status diagram (Page 1068), asynchronous calls (Page 1069) and synchronous calls (Page 1071).

Note

The temperature channel technology object can only be activated and deactivated in the engineering system (see Activating or deactivating technology objects in the engineering system (Page 1074)).

Status diagram for the functions _activateTo and _deactivateTo

The figure below shows the states of the system functions _activateTo and _deactivateTo. It also applies to the _getStateOfTo system function, see also Querying the activation state of technology objects (Page 1074).

The functions can be called asynchronously and synchronously.

- With an asynchronous call (Page 1069) (parameter **nextCommand** := IMMEDIATELY), the user program is continued immediately after the start of the system function.
  You must regularly query the state of the function to be able to determine when a command has been executed in full.
  The asynchronous call is required in cyclic tasks to ensure that their time monitoring function does not start.
- With a synchronous call (Page 1071)
  (parameter **nextCommand** : =WHEN_COMMAND_DONE), the user program waits until the system function has been completed. The return value directly indicates whether the function call was successful.
  The synchronous call is recommended for MotionTasks.

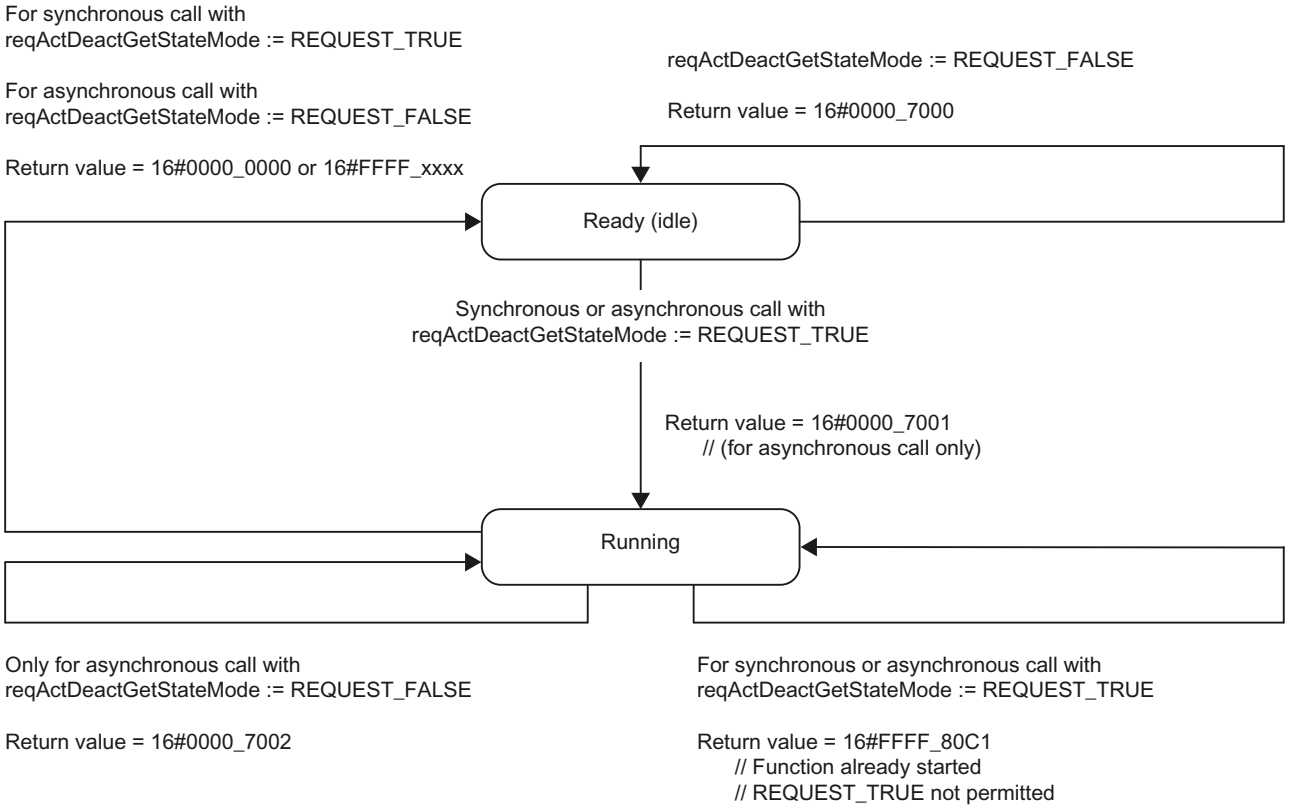
Use parameter **reqActDeactGetStateMode :=** REQUEST_TRUE to start execution of the function (e.g. activation or deactivation of a technology object).

Start the status query of the function with parameter **reqActDeactGetStateMode :=** REQUEST_FALSE. Use the same command ID as for the first call.

Note

To correctly complete the function on an asynchronous call, you must call the function with the parameter reqActDeactGetStateMode := REQUEST_FALSE as often as necessary until you obtain 16#0000_0000 as the return value or an error message.

You can only execute the function once at the same time for each technology object.



Return values

16#0000_0000	// Activation or deactivation successful
16#0000_7000	// Function in ready mode (idle), initial call required with REQUEST_TRUE
16#0000_7001	// Function started (only for asynchronous call)
16#0000_7002	// Function running (only for asynchronous call)
16#FFFF_xxxx	// Function terminated with error

Figure 4-25 Status diagram of the system functions _activateTo and _deactivateTo

Asynchronous call for the functions _activateTo and _deactivateTo

The asynchronous call is required in cyclic tasks to ensure that their time monitoring function does not start.

With an asynchronous call, the user program is continued immediately after the start of the system function. You must query the state of the function regularly to determine whether it has been completed without errors.

Proceed as follows for the asynchronous call:

1. Call the following parameters to start the system function:
   - reqActDeactGetStateMode := REQUEST_TRUE
   - nextCommand := IMMEDIATELY
2. Check the return value.
   - Return value 16#0000_0000 indicates that the function has been completed without errors.
     Continue with step 5.
   - The return value 16#0000_7001 indicates that the function has been started successfully, but has not yet been completed.
     Continue with step 3.
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
3. For the state query and to check whether the system function has been completed, call these again, but with the following parameters:
   - reqActDeactGetStateMode := REQUEST_FALSE
   - nextCommand := IMMEDIATELY
   - commandId: Use the same command ID as for the first call.
4. Check the return value:
   - Return value 16#0000_7002 indicates that the function is still running.
     Repeat step 3.
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
   - Return value 16#0000_0000 indicates that the function has been completed without errors.
5. On error-free completion of the functions _activateTo and _deactivateTo note the information below.

Note

Note for versions up to and including V4.0 of the SIMOTION Kernel:

The activation or deactivation of the technology object is not yet finished on completion without errors of the functions _activateTo or _deactivateTo.

Therefore, keep calling the _getStateOfTo (Page 1074) system function until the commandIdState component of the return value (data type EnumStateOfTo) has the value ACTIVE or INACTIVE.

Example program for the asynchronous call for the _activateTo function

```
INTERFACE
    USEPACKAGE Cam;
    PROGRAM backGround;

    VAR_GLOBAL
        bgrRequestActivate : DINT := 0;
        bgrResultActivate  : DINT := 0;
        bgrToInstance      : posaxis;
        bgrReqActDeactMode : enumReqActDeactGetStateMode := REQUEST_TRUE;
        bgrCommandId       : CommandIdType;
        bgrNextCommand     : enumNextCommandMode := IMMEDIATELY;
    END_VAR
END_INTERFACE

IMPLEMENTATION
PROGRAM backGround // Assigned to the BackgroundTask
    VAR
        retVal : DINT := 0;
    END_VAR

    // ... Further instructions

    IF ( 0 <> bgrRequestActivate ) THEN
        retVal := _activateTo (
                    TO_Instance             := bgrToInstance,
                    reqActDeactGetStateMode := bgrReqActDeactMode,
                    commandId               := bgrCommandId,
                    nextCommand             := bgrNextCommand );

        CASE retVal OF
            16#0000_0000: bgrRequestActivate := 0;
                // Node successfully activated.
            16#0000_7001: bgrReqActDeactMode := REQUEST_FALSE;
                // Function started successfully
            16#0000_7002: ; // Function running.
        ELSE // Troubleshooting
             bgrRequestActivate := 0;
        END_CASE;

        IF ( 0 = bgrRequestActivate ) THEN
            // Function completed, prepare next call.
            bgrReqActDeactMode := REQUEST_TRUE;
            bgrResultActivate := retVal;
        END_IF;
    END_IF;

    // ... Further instructions

END_PROGRAM
END_IMPLEMENTATION
```

Synchronous call for the functions _activateTo and _deactivateTo

The synchronous call is recommended for MotionTasks.

With a synchronous call, the user program waits until the system function has been completed. The return value directly indicates whether the function call was successful.

Proceed as follows for the synchronous call:

1. Call the following parameters to start the system function:
   - reqActDeactGetStateMode = REQUEST_TRUE
   - nextCommand = WHEN_COMMAND_DONE
2. Check the return value:
   - A return value < 0 indicates that the function has been terminated with errors. You can program error routines, depending on the return value.
   - Return value 16#0000_0000 indicates that the function has been completed without errors.

For the functions _activateTo and _deactivateTo note the information below.

Note

Note for versions up to and including V4.0 of the SIMOTION Kernel:

The activation or deactivation of the technology object is not yet finished on completion without errors of the functions _activateTo or _deactivateTo.

Therefore, keep calling the _getStateOfTo (Page 1074) system function until the commandIdState component of the return value (data type EnumStateOfTo) has the value ACTIVE or INACTIVE.

Example program for the synchronous call for the _activateTo function

```
INTERFACE
    USEPACKAGE Cam;
    PROGRAM motion_1;

    VAR_GLOBAL
      motion_1_RequestActivate  : DINT := 0;
      motion_1_ResultActivate   : DINT := 0;
      motion_1_ToInstance       : posaxis;
      motion_1_CommandId        : CommandIdType;
    END_VAR
END_INTERFACE

IMPLEMENTATION
PROGRAM motion_1 // Assigned to a MotionTask
    VAR
        retVal : DINT := 0;
    END_VAR

    // ... Further instructions

    IF ( 0 <> motion_1_RequestActivate ) THEN
        retVal := _activateTo (
                    TO_Instance             := motion_1_ToInstance,
                    reqActDeactGetStateMode := REQUEST_TRUE,
                    commandId               := motion_1_CommandId,
                    nextCommand             := WHEN_COMMAND_DONE );

        CASE retVal OF
            16#0000_0000: motion_1_RequestActivate := 0;
                // Node successfully activated.
        ELSE // Troubleshooting
            motion_1_RequestActivate := 0;
        END_CASE;

        IF ( 0 = motion_1_RequestActivate ) THEN
            // Function completed.
            motion_1_ResultActivate := retVal;
        END_IF;
    END_IF;

    // ... Further instructions

END_PROGRAM
END_IMPLEMENTATION
```

Activating or deactivating technology objects in the engineering system

You can also activate or deactivate technology objects in the engineering system (e.g. SIMOTION SCOUT). After downloading the project to the target system, these technology objects are activated or deactivated during ramp-up of the system (default setting: all technology objects are activated).

Note

You can only make the settings with the engineering system in offline mode. They only take effect after the project is downloaded to the target system.

1. Make sure that the engineering system (e.g. SIMOTION SCOUT) is in offline mode.
2. Select the SIMOTION device (e.g. D435) in the project navigator.
3. Select the **Edit > Object states** menu.
   A list of all technology objects configured on this SIMOTION device is displayed. The check box in front of the identifier of each technology object indicates whether it is activated during the transition from STOP to RUN operating mode.
4. For each technology object, define whether the object is to be activated when the system is ramped up:
   – Activate the check box if the relevant technology object is to be activated (default setting).
   – Deactivate the check box if the relevant technology object is not to be activated.
5. Repeat steps 2 to 4 for all SIMOTION devices.
6. Change to online mode and download the project to the target system.

After the system is ramped up, you can change the activation and deactivation settings by calling the system functions **_activateTo** and **_deactivateTo** (e.g. in the StartupTask).

You can monitor the current state in online mode using the engineering system (see Querying the activation state of technology objects (Page 1074)).

Querying the activation state of technology objects

You can now carry out the following for individual technology objects:

- Query the activation state using system functions (Page 1074)
- Monitor the activation state in the engineering system (e.g. SIMOTION SCOUT) (Page 1075).

Querying the activation state of technology objects using system functions

System function

The **_getStateOfTo** system functions can be used to query the status of an individual technology object in a user program.

The following information is contained in the return value of the function (data type StructRetGetStateOfTo):

- The state of the function in the functionResult component (data type DINT)
- The activation state of the technology object in the commandIdState component (data type EnumStateOfTo) (only when component functionResult = 16#0000_0000)

The syntax of these system functions is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Status diagram, asynchronous and synchronous call

As regards the main features, the same information applies as for the _activateTo and _deactivateTo functions, see: .

- Status diagram for the functions _activateTo and _deactivateTo (Page 1068)
- Asynchronous call for the functions _activateTo and _deactivateTo (Page 1069)
- Synchronous call for the functions _activateTo and _deactivateTo (Page 1071)

Monitoring the activation state of technology objects in the engineering system

You can monitor the current activation state of the technology objects of a SIMOTION device in the engineering system (e.g. SIMOTION SCOUT). The engineering system must be in online mode.

Note

You cannot make any settings for the activation or deactivation of technology objects in online mode.

1. Make sure that the engineering system (e.g. SIMOTION SCOUT) is in online mode.
2. Select the SIMOTION device (e.g. C240) in the project navigator.
3. Select the **Edit > Object states** menu.
   A list of all technology objects configured on this SIMOTION device is displayed. The check box in front of the identifier of each technology object indicates the activation state after the transition to RUN operating mode and the current activation state:
   - Check mark in the check box: activation state after the transition to RUN operating mode (setting in offline mode)
   - Color of the check box: current activation state (green = activated).

4.1.5.5 Procedure for deactivating and activating components

The following is a brief summary of the steps that must be performed when deactivating and activating each component (e.g. machine modules, DP slave with assigned technology objects):

- Deactivating components (Page 1076)
- Activating components (Page 1077)

Note

If possible, deactivate technology objects and components first before activating other technology objects and components. In this way you will avoid level overflows.

Deactivating components

Proceed in this sequence if you want to deactivate technology objects and the distributed I/O to which they are assigned:

1. Terminate all commands that are present on the technology objects to be deactivated (e.g. motion, homing).
2. Remove all enables for these technology objects with the specific system function for the technology object in question, e.g. _disableAxis.
   Also remove the enables for dependent technology objects (for axes, e.g. measuring inputs, output cams, synchronous objects).
3. Deactivate these technology objects individually:
   - Observe the correct sequence if technology objects are dependent on one another: First deactivate all dependent technology objects (e.g. measuring inputs, output cams, synchronous objects) before deactivating their base technology object (e.g. axis).
   - Start the deactivation process with the system function _deactivateTo (Page 1067).
   - Only up to version V4.0 of the SIMOTION kernel:
     After error-free completion of the _deactivateTo system function, keep calling the _getStateOfTo (Page 1074) system function until the commandIdState component of the return value (data type EnumStateOfTo) has the value INACTIVE.
4. When all technology objects that are assigned to a distributed I/O (e.g. drive) are deactivated:
   - Deactivate the distributed I/O (e.g. drive) with the system function _deactivateDpSlave (Page 1032)
     Or
   - Deactivate individual drive objects in the SINAMICS S120 drive line-up with parameter p0105, see Activating and deactivating SINAMICS components (Page 1064).
5. If required, remove the component (e.g. machine module).

Activating components

Proceed in this sequence if you want to activate the distributed I/O and the technology objects assigned to it:

1. Connect the component (e.g. machine module).
2. Activate the distributed I/O:
   - Activate the distributed I/O (e.g. drive) with the system function _activateDpSlave (Page 1032)
     Or
   - Activate individual drive objects in the SINAMICS S120 drive line-up with parameter p0105, see Activating and deactivating SINAMICS components (Page 1064).
3. Activate the assigned technology objects individually:
   - Observe the correct sequence if technology objects are dependent on one another: First activate the base technology object (e.g. axis) before activating the dependent technology objects (e.g. measuring inputs, output cams, synchronous objects).
   - Start the activation process with the system function _activateTo (Page 1067).
   - Only up to version V4.0 of the SIMOTION kernel:
     After error-free completion of the _activateTo system function, keep calling the _getStateOfTo (Page 1074) system function until the commandIdState component of the return value (data type EnumStateOfTo) has the value ACTIVE.
4. Set the required enables for these technology objects with the specific system function for the technology object in question, e.g. _enableAxis.
5. If required, home the axis with the _homing system function.
6. Start further motion commands.

4.1.6 Changing the active configuration or the active kernel

A configuration contains all configuration data for a SIMOTION device (e.g. device configuration, communications configuration, technology objects, global device variables, I/O variables, user programs). The active configuration is stored on the SIMOTION device's memory card and is loaded with each system ramp-up (after POWER ON).

This section describes how you can perform the following actions in the user program:

1. Replace the active configuration by another one whose card image is present on the memory card. You can explicitly specify the configuration to be activated (see Activating a configuration (Page 1078)) or select it via an initial configuration (see Selecting and activating a configuration using an initial configuration (Page 1089)).
2. Replace the active kernel of the SIMOTION device with another one whose card image is present on the memory card (see Activating a kernel (Page 1084)).

In this way, you can update the configuration and/or the SIMOTION Kernel without replacing the memory card.

Note

If need be, you may have to update the corresponding firmware of interlinked SINAMICS drive units.

4.1.6.1 Activating a configuration

A configuration contains all configuration data for a SIMOTION device (e.g. device configuration, communications configuration, technology objects, global device variables, I/O variables, user programs). The active configuration is stored on the SIMOTION device's memory card and is loaded with each system ramp-up (after POWER ON).

This section describes how you can replace the active configuration by another one whose card image is present on the memory card by means of a user program. In this case, the configuration to be activated is specified explicitly or determined by the user program. In this way, a configuration adapted for each task can be added:

- The memory and computing capability of the SIMOTION device are only used for the configured functionalities
- Cycle times and cycle clocks (e.g. DP cycle clock, position control cycle clock, IPO cycle clock) can be optimized for each requirement.

A typical application case is as follows: On a SIMOTION device (base machine), various machine modules are connected in turn. The configuration data for each arrangement (base machine with connected machine module) is stored as its own configuration on the memory card of the SIMOTION device. After the machine module is connected to the base machine, the user program identifies the relevant machine module.

System function

When the **_activateConfiguration** system function is initiated, the user program loads the required configuration and restarts the SIMOTION device. Provide the configuration to be activated in parameter projectDataId (data type UDINT). The hexadecimal number to be provided corresponds to the last six characters of the relevant file name assigned to the card image of the respective configuration on the memory card (example: File name = PR000001.ZIP, projectDataId = 16#000001).

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

Note when activating a configuration:

- Retentive global device variables are always initialized.
- The values of the following variables of the last active configuration can be preserved: Retentive unit variables and retentive system variables of the technology objects. Observe the measures described in Retaining retentive data in the event of a configuration change (Page 1095).
- If the SINAMICS Safety Integrated Functions configuration is used, an acceptance test is mandatory when the SIMOTION device is restarted (see "SINAMICS S120 Safety Integrated" Function Manual).
- After restarting the SIMOTION device, check the Activation state of the configuration (Page 1094) with the _getConfigurationData system function.

The activated configuration remains after the SIMOTION device is turned off. After the device is turned on (POWER ON), the configuration is reloaded.

A detailed description of how the card images of configurations are created is provided in Section "Procedure for creating the card images of configurations" (Page 1098).

Note

The following applies up to version V4.2 of the SIMOTION Kernel:

- If one or more CX32/CX32-2 controller extensions are connected to a SIMOTION D4x5/ D4x5-2, note the following when activating a configuration with the _activateConfiguration function:
  The activated configuration will only take effect on the CX32/CX32-2 controller extension after the CX32/CX32-2 has been restarted. This restart must be specifically initiated by the user, as the _activateConfiguration function does not perform a restart automatically.
- Note the information in the Section "Special considerations when a CX32/CX32-2 controller extension is connected" (Page 1080).

As of SIMOTION Kernel version V4.3, restart of the CX32/CX32-2 takes place automatically with the function _activateConfiguration.

Note

For the implementation of a modular machine, several projects with their drive configurations can be stored on the SIMOTION memory card. These configurations can then be activated, and it is possible to switch between the different configurations. You must therefore ensure that all projects with their drive configurations (incl. safety configuration) stored on the SIMOTION memory card for SINAMICS Integrated function correctly on the entire system and have been subject to a full safety acceptance test. For projects with mixed operation, the user is responsible for the installation of passive (e.g. marking or suitable safety precautions) or active (e.g. monitoring of the relevant axes) safety measures.

Mixed operation is the switchover between safety and non-safety axes. This also includes the replacement of safety axes with the same safety parameterization with regard to the communication with the safety module in the different configurations.

Note

If the size of the retentive data blocks changes in the new configuration, fragmentation of the retentive memory may make it impossible to activate the configuration.

The following applies as of version V4.5 of the SIMOTION Kernel:

- The retentive memory will be reorganized when the device is restarted.
  The values of the retentive global unit variables are retained if the symbol information is saved in the device. For this purpose, the compiler option "Permit OPC-UA / -XML (load symbols to RT)" must be activated for all relevant program sources during compilation.
- Please also note the relevant information in section "Retaining retentive data" (Page 1095).

The following applies up to version V4.4 of the SIMOTION Kernel:

- You must perform a memory reset on the device.
  The retentive data is lost.

Special considerations when a CX32/CX32-2 controller extension (up to Kernel V4.2) is connected

The following applies up to version V4.2 of the SIMOTION Kernel:

- If one or more CX32/CX32-2 controller extensions are connected to a SIMOTION D4x5/ D4x5-2, note the following when activating a configuration with the _activateConfiguration function:
  The activated configuration will only take effect on the CX32/CX32-2 controller extension after the CX32/CX32-2 has been restarted. This restart must be specifically initiated by the user, as the _activateConfiguration function does not perform a restart automatically.

Forcing restart (procedure)

To force the CX32/CX32-2 controller extension to restart, proceed as follows (3 options):

- Power Off/On for the entire system
- Power Off/On of the CX32/CX32-2 controller extension
- Set p0972 = 1 on the CX32/CX32-2 controller extension:
  - In the user program: with the function _writeDriveParameter, for example
    Or
  - In SIMOTION SCOUT: In the expert list for the "Control_Unit" of the CX32/CX32-2 controller extensions

  This restarts the CX32/CX32-2 immediately.

⚠ DANGER
Avoid unsafe states
Ensure the system is in a safe state!
No attempt should be made to access the memory card of the SIMOTION D4x5/D4x5-2.

Example for the activation of a configuration

Description

In the figure below, a transport device with a SIMOTION control transports a workpiece to various machining stations. The following sequences can be observed here:

1. Drawing a blank from the store and transporting it to the machining station.
2. Machining the blank.
3. Unloading the semi-finished part at the packing station.

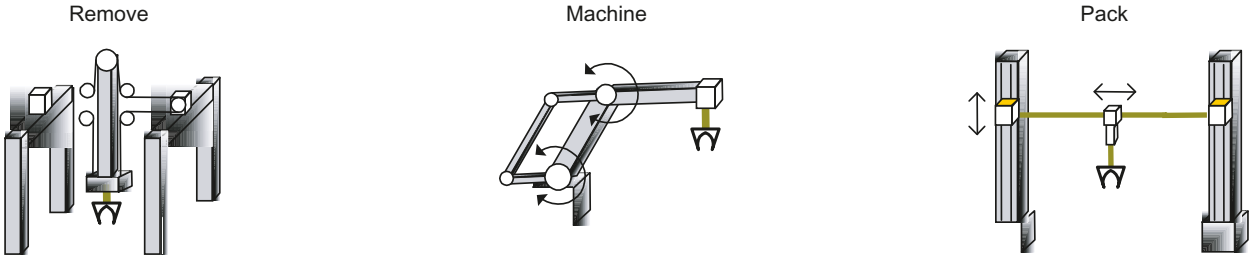


Figure 4-26 Sequences of an automation task

Implementation

For each machining step, the associated optimized configuration is stored on the memory card of the transport device. After the transport device has been coupled to one of the machining stations, it identifies the relevant machining station (e.g. by evaluating a connector coding) and loads the necessary configuration. The table below shows examples of the following for the machining stations shown in the figure above:

- Their connector coding
- The file name of the associated configuration on the memory card of the transport device and
- The projectDataId associated with the file name, which is required when calling the **_activateConfiguration** system function

Table 4-11 Identifying the configuration

Designation of the machining station	Connector coding	File name of the configuration	projectDataId
Remove	001	PR000001.ZIP	16#000001
Machine	010	PR000002.ZIP	16#000002
Pack	100	PR000003.ZIP	16#000003

Example program

The following programming example shows:

- The identification of the required configuration in accordance with the table above
- The activation of the required configuration

Example program for the identification and activation of a required configuration

```
// Variable declaration
VAR
    retActivateConf  : StructRetConfiguration;
    plugCode AT %IB0 : BYTE;  // Connector coding
    configId         : UDINT; // projectDataId of the
                              // required configuration
END_VAR

// Identify the required configuration
// by evaluating the connector coding
CASE plugCode OF
    2#001   : configId := 16#000001; // Remove  (PR000001.ZIP)
    2#010   : configId := 16#000002; // Machine (PR000002.ZIP)
    2#100   : configId := 16#000003; // Pack  (PR000003.ZIP)
    ELSE
         ; // ... Error response
END_CASE;

// Activate the required configuration
retActivateConf := _activateConfiguration (
        projectData   := YES,
        projectDataId := configId,
        timeOut       := T#5m       // 5 minutes
    );

IF ( 0 <> retActivateConf.result ) THEN
    // Error response
    ; // ...
END_IF;
```

About parameter timeOut

The timeOut parameter specifies the maximum time the SIMOTION device can require after the restart until the RUN mode is reached. When this time is exceeded, the system performs the following actions:

1. The system tries to activate the initial configuration (if present), and to use this to select a configuration (see Selecting and activating a configuration using an initial configuration (Page 1089)).
2. In the event of an error, the device switches to the STOP operating mode.

Monitoring is deactivated when timeOut = T#0s.

Note

Make sure that the selected system cycle clocks are long enough, otherwise a timeout (internal error) may occur while a configuration is powering up.

4.1.6.2 Activating a kernel

Activating a kernel

This section describes how you can exchange the active kernel of the SIMOTION device using the user program and, for example, carry out an update to a new version without having to replace the memory card of the device. In this regard, you must replace all configurations (configuration data) of the SIMOTION device.

To do this, you transfer the card images of the kernel and all necessary configurations to the INSTALL\SIMOTION\VExxxxxx directory on the SIMOTION device's memory card. In the directory name VExxxxxx there is a 6-digit hexadecimal number xxxxxx. You specified this directory name when creating the card images with the /i"VExxxxxx" option. A detailed description of how the card images of the kernel and configurations are created is provided in Section "Procedure for creating the card images of configurations (Page 1098)".

Note

With SIMOTION P3xx devices, it is not possible to exchange the SIMOTION Kernel in the way described above. It is necessary to make additional changes in the Windows system that necessitate replacement of the entire image.

The required data is saved in the following subdirectories in the VExxxxxx directory:

- INITIAL directory: The card image of an initial configuration (file name INITIAL.ZIP), if present
- KERNEL directory:
  - For SIMOTION C2xx devices
    The card image of the SIMOTION Kernel (file name KERNEL.ZIP).
  - For the SIMOTION D devices:
    The card image of the SIMOTION Kernel including the firmware for the SINAMICS Integrated (file name KERNEL.ZIP)
- PACKAGE directory: The card images of the technology objects
- PROJECT directory: The card images for all configurations (file name PRyyyyyy.ZIP, where yyyyyy is a hexadecimal number that corresponds to the projectDataId)

Note

Select a designation for the VExxxxxx directory from which the version of the SIMOTION Kernel can be clearly identified, e.g. VE004156 for version V4.1 SP5 HF6.

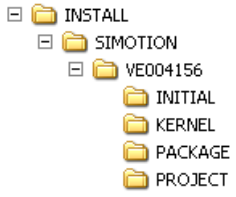


Figure 4-27 Directory structure on the memory card

System function

To activate the SIMOTION Kernel stored on the card, call in the user program the **_activateConfiguration** system function with the kernel := YES parameter. In the kernelId parameter, specify the last six characters of the directory name on the memory card (example: Directory name = VE004156, kernelId = 16#004156).

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Specify the configuration to be activated in the projectData and projectDataId parameters:

- projectData := NO: The projectDataId parameter will be ignored; the initial configuration will be activated. In this initial configuration, the required configuration will be determined, for example, by evaluating a connector coding, and loaded by calling the _activateConfiguration system function again.
  See "Example *without* details of a configuration to be activated" in "Examples for activating a kernel" (Page 1086).
  For information on the initial configuration, see Selecting and activating a configuration using an initial configuration (Page 1089).
- projectData := YES: The configuration specified in the projectDataId parameter (UDINT data type) will be activated. The hexadecimal number to be transferred corresponds to the last six characters of the relevant file name of the configuration on the memory card (for example, File name = PR000001.ZIP, projectDataId = 16#000001).
  See "Example *with* details of a configuration to be activated" in "Examples for activating a kernel" (Page 1086).

Note

When activating a kernel and the associated configuration, please note:

- Retentive global device variables are always initialized.
- The values of the following variables of the last active configuration can be preserved: Retentive unit variables and retentive system variables of the technology objects. Observe the measures described in "Retaining retentive data in the event of a configuration change" (Page 1095).
- If the SINAMICS Safety Integrated Functions configuration is used, an acceptance test is mandatory when the SIMOTION device is restarted (see "SINAMICS S120 Safety Integrated" Function Manual).
- After restarting the SIMOTION device, check the Activation state of the kernel and the configuration (Page 1094) with the _getConfigurationData system function.

The chronological sequence of the activation of a kernel and the required configuration is described in the Status diagram (Page 1089).

The activated kernel and the activated configuration are retained after the SIMOTION device is turned off. After the device is turned on (POWER ON), they are reloaded.

User administration for SIMOTION IT

The SIMOTION IT user administration depends on the SIMOTION Kernel version:

- The following applies up to SIMOTION Kernel version V4.3:
  The user administration is integrated in the `webcfg.xml` file. This file is on the memory card of the SIMOTION device under the path: `\user\simotion\hmicfg`
- The following applies as of SIMOTION Kernel version V4.4:
  The user administration is contained in the `userdatabase.xml` file. This file is on the memory card of the SIMOTION device in path: `\user\simotion\hmicfg \userdatabase`

When changing the SIMOTION Kernel version from 4.3 to V4.4, an attempt is made to convert the previous user administration into the new format. However, this is not always possible (e.g. "simotion" user available with standard password). Note Section "Firmware upgrade to V4.4" in the "SIMOTION IT Diagnostics and Configuration" Diagnostics Manual as well as further sections in Chapter "Manage Config".

For this reason, it is recommended that you integrate the user administration in the configurations when changing the kernel. Refer to the information on "User administration for SIMOTION IT" in Section "Saving the configuration files in the work directories on the PC/PG" (Page 1102).

Example programs for activating a kernel

The following two examples for activating a kernel make a distinction between two scenarios:

- The configuration to be activated is not specified (see "Example *without* details of a configuration to be activated").
- The configuration to be activated is specified (see "Example *with* details of a configuration to be activated").

Example *without* details of a configuration to be activated

The following example shows the activation of a kernel without details of a configuration to be activated.

After activating the kernel and restarting the system, the initial configuration will be loaded. In this initial configuration, the required configuration will be determined, for example, by evaluating a connector coding, and activated by calling the _activateConfiguration system function again.

For information on the initial configuration, see "Selecting and activating a configuration using an initial configuration" (Page 1089).

Example for activating a kernel without details of a configuration to be activated

```
// Variable declaration
VAR
    retActivateConf : StructRetConfiguration;
    newKernelId : UDINT;    // kernelId of the
                            // kernel to be loaded
END_VAR

newKernelId := 16#004156;   // Corresponds to directory
                            // VE004156

// Activate the required configuration
retActivateConf := _activateConfiguration(
        projectData     := NO,          // Mandatory parameters
        projectDataId   := UDINT#MAX,   // 16#FFFF_FFFF
        kernel          := YES,
        kernelId        := newKernelId,
        timeOut         := T#5m         // 5 minutes
    );

IF ( 0 <> retActivateConf.result ) THEN
    // Error response
    ; // ...
END_IF;
```

The timeOut parameter is described in detail in Section "Example of the activation of a configuration" (Page 1081). The time should be at least five minutes for the activation of a kernel.

Example *with* details of a configuration to be activated

The following example shows the activation of a kernel with details of a configuration to be activated.

After activating the kernel and restarting the system, the configuration specified in the projectDataId parameter will be loaded.

Example for activating a kernel with details of a configuration to be activated

```
// Variable declaration
VAR
    retActivateConf : StructRetConfiguration;
    newKernelId : UDINT; // kernelId of the
                         // kernel to be loaded
    configId    : UDINT; // projectDataId of the
                         // required configuration
END_VAR

newKernelId := 16#004156; // Corresponds to directory
                          // VE004156
configId    := 16#000001; // Corresponds to file
                          // PR000001.ZIP

// Activate the required configuration
retActivateConf := _activateConfiguration (
        projectData := YES,
        projectDataId := configId,
        kernel := YES,
        kernelId := newKernelId,
        timeOut := T#5m // 5 minutes
    );

IF ( 0 <> retActivateConf.result ) THEN
    // Error response
    ; // ...
END_IF;
```

The timeOut parameter is described in detail in Section "Example of the activation of a configuration" (Page 1081). The time should be at least five minutes for the activation of a kernel.

Status diagram for activating a kernel

The figure below shows the chronological sequence for the activation of a kernel depending on the projectData parameter (specification of a configuration to be activated).

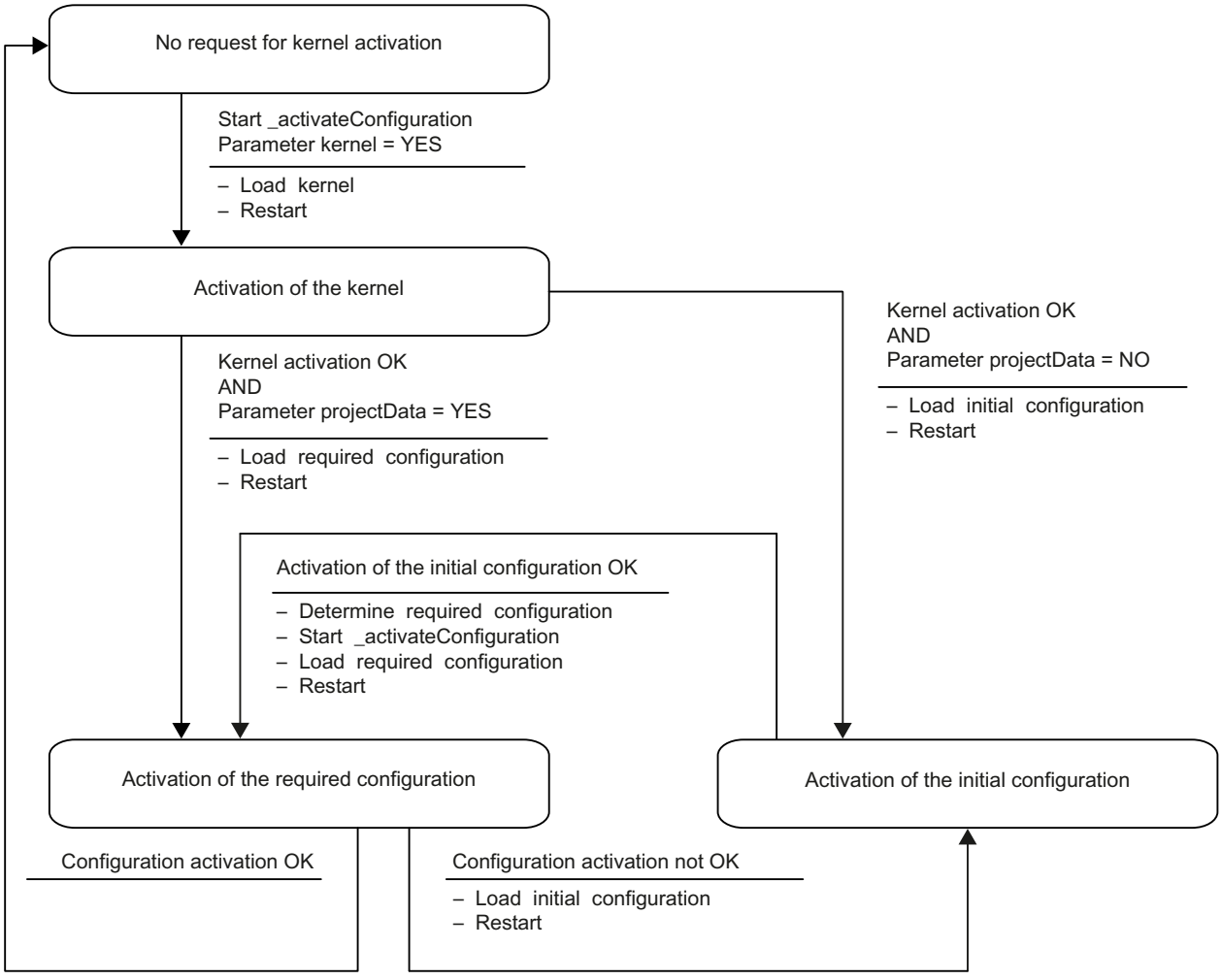


Figure 4-28 Status diagram for activating a kernel depending on the projectData parameter

4.1.6.3 Selecting and activating a configuration using an initial configuration

Selecting and activating a configuration using an initial configuration

Loading an initial configuration during ramp-up (after POWER ON) of a SIMOTION device ensures that it can be integrated failure-free into a running communication network of another device after switch-on.

For example, a SIMOTION device (machine module) is connected to a different SIMOTION device (base machine). After POWER ON, the machine module ramps up, identifies its position at the base machine (e.g. using a connector coding), and automatically loads the required configuration.

The card images for all configurations that the SIMOTION device (machine module) can transfer to the base machine is stored on its memory card. The initial configuration also stored there will be loaded automatically after POWER ON. This identifies (e.g. using a connector coding) the required configuration and loads it.

Note

Note the following restrictions on configuring the initial configuration:

1. The interface to the base machine must be configured with a communication address (e.g. PROFIBUS address or IP address) that is not used in the associated communication network of the base machine.
   This does not disturb communication in this network.
2. There must be no retentive data (e.g. retentive unit variables) in the user program. You can check this with **Display reference data > Code attributes**.
   The retentive unit variables of the last activated configuration thereby remain intact.
3. No instances of technology objects may be configured.
   This prevents retentive instance data of the technology objects. The retentive system variables of the technology objects of the last active configuration remain intact.
4. No drives should be configured.
   This reduces the loading time for the initial configuration.

The card image of the initial configuration (file name INITIAL.ZIP) is located in the INITIAL subdirectory of the relevant kernel version (e.g. INSTALL\SIMOTION\VE004156). A detailed description of how the card images of the initial configuration and the configurations to be activated are created is provided in "Procedure for creating the card images of configurations" (Page 1098).

SelfAdaptingConfiguration functionality

The initial configuration will be started automatically during ramp-up of the SIMOTION device if the SelfAdaptingConfiguration functionality has been activated on its memory card. This will be the case if the following actions have been performed:

- The /a"ON" option was also specified when creating the card images of the configurations (Page 1106) with the u7mkcnfx.exe program (/m"MAKE_STORE" option).
- In the last user program to run (before POWER OFF),
  the **_setModeSelfAdaptingConfiguration** system function was called with the mode := ON parameter.

This makes a corresponding entry on the memory card, which causes the initial configuration to be loaded after POWER ON.

You can disable the SelfAdaptingConfiguration functionality by calling the **_setModeSelfAdaptingConfiguration** system function with the mode := OFF parameter. This makes a corresponding entry on the memory card, which causes the last active configuration to be loaded after POWER ON.

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Note

The activation and the deactivation of the SelfAdaptingConfiguration functionality only take effect the next time the SIMOTION device is switched on (POWER ON).

Note

In a communication network (PROFIBUS subnet or PROFINET or Ethernet segment), only one SIMOTION device may be active simultaneously with the initial configuration.

The existence of several nodes with the same communication addresses in a communication network causes a fault.

Note

When activating a configuration through an initial configuration, please note:

- Retentive global device variables are always initialized.
- The values of the following variables of the last active configuration can be preserved: Retentive unit variables and retentive system variables of the technology objects. Please note in this regard:
  - The restrictions for the initial configuration given in the **Notice** information at the start of this section
  - The measures for the configuration to be activated described in "Retaining retentive data in the event of a configuration change" (Page 1095)

Note

If one or more CX32 controller extensions are connected to a SIMOTION D4x5, please note the following when activating a configuration with the _activateConfiguration function:

The activated configuration will only take effect on the CX32 controller extension after the CX32 has been restarted. This restart must be specifically initiated by the user, as the _activateConfiguration function does not perform a restart automatically.

Note the information in the section titled "Special considerations when a CX32 controller extension is connected" (Page 1080).

Status diagram for what happens when a SIMOTION device is switched on depending on whether or not the SelfAdaptingConfiguration functionality is active

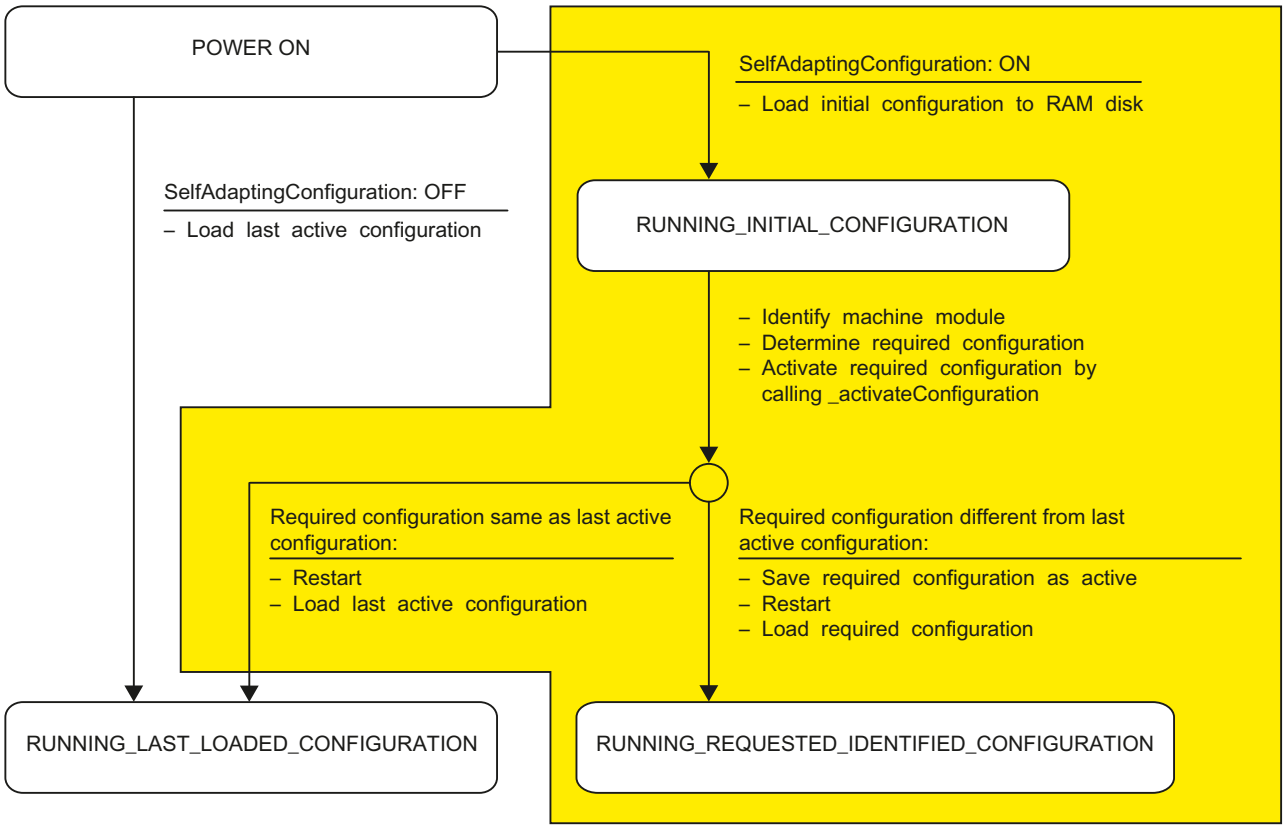


Figure 4-29 Status diagram for what happens when a SIMOTION device is switched on depending on whether or not the SelfAdaptingConfiguration functionality is active

When the device is switched on (POWER ON), the SIMOTION performs a one-off run through the status diagram:

- If the SelfAdaptingConfiguration functionality is activated, the initial configuration will be loaded.
  The initial configuration identifies the position of the machine module on the base machine, determines the required configuration, and activates it using the _activateConfiguration system function.
  The system function checks whether the configuration to be activated is identical to the most recently active configuration:
  - **Identical**: After the device restart, the most recently active configuration will be loaded.
  - **Not identical**: The required configuration will be saved as the active configuration and will be loaded after a device restart.
- If the SelfAdaptingConfiguration functionality is deactivated, the most recently active configuration will be loaded.

Example program for an initial configuration

The following example program shows the determination and activation of a configuration in the initial configuration:

Example for activating a configuration in an initial configuration

```
// Variable declaration
VAR
    ipRead          : ARRAY [1..2] OF StructRetIpConfig;
    setIpAddress    : ARRAY [0..5] OF USINT;
    setIpSubnet     : ARRAY [0..5] OF USINT;
    setIpGateway    : ARRAY [0..5] OF USINT;
    retUdint        : UDINT;
    retActivateConf : StructRetConfiguration;
    // IP address and projectDataId of the configuration are
    // read from output with address 0.
    ip_conf_id AT %IB0 : USINT;
END_VAR

// Read the current data of the Ethernet interfaces
ipRead[1] := _getIpConfig (
        ethernetInterface:= IE_01);
ipRead[2] := _getIpConfig (
        ethernetInterface:= IE_02);

// Change of the Ethernet interface IE_01.
// Check whether a change is necessary.
IF ( ip_conf_id <> ipRead[1].ipAddress[3] ) THEN
    // Make available the data to be changed
    setIpAddress    := ipRead[1].ipAddress;
    setIpAddress[3] := ip_conf_id;
    setIpSubnet     := ipRead[1].subnetMask;
    setIpGateway    := ipRead[1].gatewayAddress;
    // Check whether, following the change, the subnet masks
    // of the two Ethernet interfaces are different:
    IF ( ipRead[2].subnetMask <> setIpSubnet ) THEN
        // Set the data of the Ethernet interface IE_01
        retUdint := _setIpConfig (
            ethernetInterface := IE_01,
            ipAddress         := setIpAddress,
            subnetMask        := setIpSubnet,
            gatewayAddress    := setIpGateway
          );
        // Activate the required configuration
        retActivateConf := _activateConfiguration(
            projectData   := YES,
            projectDataId := ip_conf_id,
            timeOut       := T#1m // 1 minute
          );
    ELSE
      ; // No change, error response,
        // change interface IE_02 too, if necessary.
    END_IF;
END_IF
```

4.1.6.4 Reading the activation state of configurations

You can use the **_getConfigurationData** function to query the status of the last two calls of the **_activateConfiguration** system function.

The syntax of this system function is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

A description of some input parameters follows:

- You specify in the **configurationInfoId** parameter (EnumConfigurationInfoId data type), the configuration whose status you want to query:
  - ACTUAL_ACTIVATED: The status of the currently active configuration (last call of the _activateConfiguration function) will be queried.
  - PREVIOUS_ACTIVATED: The status of the previously active configuration (next to last call of the _activateConfiguration function) will be queried.
- In the **requestMode** parameter (EnumReqSysFunctMode data type), you specify whether the actual function (query of the configuration state) is to be executed or whether the status of the command is to be queried.
  - REQUEST_TRUE: The function (query of the configuration state) will be executed.
  - REQUEST_FALSE: The query of the status of the command will be executed (required for asynchronous call).
    The same principles illustrated by the Status diagram for the functions _activateTo and _deactivateTo (Page 1068) and _getStateOfTo, apply.
- You specify in the **nextCommand** parameter (EnumNextCommandMode data type) when the next command is to be executed:
  - IMMEDIATELY: Immediately (asynchronous call - required for cyclic tasks).
    See "Asynchronous call for the functions _activateTo and _deactivateTo" (Page 1069).
  - WHEN_COMMAND_DONE: After the completion of the command (synchronous call - recommended for MotionTasks).
    See "Synchronous call for the functions _activateTo and _deactivateTo" (Page 1071).

The return value (**StructRetGetConfigurationData** data type) shows

- In the functionResult component (DINT data type), the general result of the function call
- In the configuration component (StructDeviceConfigurationData data type):
  - The general status of the _activateConfiguration function
  - The value of the timeOut parameter

  And for each functionality to be activated (configuration, technology object, drive, kernel)
  - Its activation status
  - The activation request placed in _activateConfiguration
  - The ID of the functionality to be loaded (e.g. the projectDataId of the configuration)

The example program below shows the status query for a configuration with a synchronous call.

Example program for the _getConfigurationData (synchronous call) function

```
// Variable declaration
VAR
    configurationInfoId     : EnumConfigurationInfoId;
    locRequest              : EnumReqSysFunctMode;
    nextCommand             : EnumNextCommandMode;
    locCommandId            : CommandIdType;
    retConfigurationInfo    : StructRetGetConfigurationData;
END_VAR

// Provide the parameter data
configurationInfoId := PREVIOUS_ACTIVATED;
locRequest      := REQUEST_TRUE; // Start request
nextCommand     := WHEN_COMMAND_DONE; // Synchronous call
locCommandId    := _getCommandId ();

// Query status
retConfigurationInfo := _getConfigurationData (
        configurationInfoId := configurationInfoId,
        requestMode         := locRequest,
        commandId           := locCommandId,
        nextCommand          := nextCommand
    );

// Check and evaluate result
If ( 0 <> retConfigurationInfo.functionResult ) THEN
    ; // ... Error response
ELSE
    ; // ... Evaluate the structure
      //     retConfigurationInfo.configuration
END_IF;
```

4.1.6.5 Retaining retentive data

Retaining retentive data in the event of a configuration change

The following section describes which measures you must adopt so that the following retentive data is retained during a change of configuration:

- Retentive unit variables
- Retentive system variables of the technology objects (e.g. absolute encoder adjustment)

Note

Note when activating a configuration:

- Retentive global device variables are always initialized.
- Without the described measures, retentive unit variables and retentive system variables of the technology objects are also initialized.

Measures for retaining retentive unit variables

Proceed as follows to ensure the values of retentive unit variables are retained (not initialized) during a configuration change:

1. Declare the retentive unit variables in a separate unit with the same identifier and the same object address in all configurations.
2. Ensure that an identical data structure is used in all configurations within this unit - particularly because of HMI device access.
3. Use several declaration blocks in the unit, if necessary.

The retentive unit variables thus receive the same version code in all configurations.

If the retentive unit variables in the most recently activated configuration and the configuration to be activated have the same version code, they are not initialized.

Note

When activating a configuration using an initial configuration, please also note the restrictions for the initial configuration given in Selecting and activating a configuration using an initial configuration (Page 1089) (in the **Notice** information).

Note

If you do not observe the above procedure, the retentive unit variables will be initialized when a different configuration is activated.

Also ensure the following setting is made for the download: The initialization of retentive variables must be disabled (**Options > Settings** menu, **Download** tab).

Alternatively, unit variables even with functions **_exportUnitDataSet** and **_importUnitDataSet** can be exported, saved and imported again.

Measures for retaining the retentive system variables of technology objects

Proceed as follows to ensure the retentive system variables of technology objects are retained (not initialized) during a configuration change.

Configure the SIMOTION devices and the technology objects in the following manner in all configurations:

1. Set the system variable of the SIMOTION device _configurationManagement.preserveToRetainData := YES in offline.
2. Assign identical identifiers and object addresses for the same technology objects.

This prevents the retentive system variables of a technology object from being initialized when the following conditions are met in the last configuration activated and the configuration to be activated:

1. Identifier and object address of the technology object are identical.
2. The structure of the retentive system variables of the technology object is the same.

Note

When activating a configuration using an initial configuration, please also note the restrictions for the initial configuration given in Selecting and activating a configuration using an initial configuration (Page 1089) (in the **Notice** information).

Note

If you do not observe the above procedure, the retentive system variables of a technology object will be initialized when a different configuration is activated.

The _configurationManagement.preserveToRetainData system variable can only be changed in offline mode and takes effect after the configuration has been activated or downloaded. It cannot be changed in online mode or in a user program.

The setting for the download for the initialization of retentive variables (**Options > Settings** menu, **Download** tab) has no effect.

Retaining retentive data when a kernel is activated

Exchanging the active kernel of the SIMOTION device, e.g. when updating to a new version, can result in the loss of retentive data.

Data loss of this kind occurs in cases where the SIMOTION Kernel is updated to a new main version and the system data types used in the declaration blocks for retentive unit variables have been extended or modified (e.g. addition of an enumerator data type) in the new main version. In this case, the version code of the declaration block is changed and the relevant data block initialized.

Workaround for updating to SIMOTION Kernel versions up to V4.3

The only way in which the loss of retentive unit variables can be avoided is to do the following:

1. Export the retentive unit variables using the **_exportUnitDataSet** system function in a user program of the old main version.
   The compiler option "Permit OPC-UA / -XML (load symbols to RT)" must be activated for all relevant program sources during compilation.
2. Import the retentive unit variables using the **_importUnitDataSet** system function in a user program of the new main version.

Workaround for updating to SIMOTION Kernel versions as of V4.4

It is no longer necessary to back up and restore the data using the **_exportUnitDataSet/ _importUnitDataSet** system function. Instead, the backed up symbol information is used to restore the retentive unit variables as far as possible.

Note

The compiler option "Permit OPC-UA / -XML (load symbols to RT)" must be activated for all relevant program sources during compilation.

The contents of retentive unit variables are retained in the following scenarios:

- The name and the data type of a variable are unchanged.
- The name of a variable is unchanged but its data type has changed and the value to be restored fits the value range of the new data type.

Update of versions V4.2 and V4.3:

The symbol information must be manually backed up on the memory card:

1. Create a directory named `\SIMOTION\RT_BACKUP` on the memory card.
2. Copy the directory `\SIMOTION\RT_DIR\UPP` together with all subdirectories and files into the directory `\SIMOTION\RT_BACKUP`.

Update of versions as of V4.4:

The symbol information is automatically backed up on the memory card.

4.1.6.6 Procedure for creating the card images of configurations

The individual steps of how to create the card images of the configurations and the associated kernel are described in the following.

1. Preparing the memory card of the SIMOTION device (Page 1098)
2. Creating the work directories on the PC/PG (Page 1098)
3. Saving the configuration files in the work directories on the PC/PG (Page 1102)
4. Creating the compressed configuration files (Page 1105)
5. Saving the kernel in the work directory on the PC/PG (Page 1106)
6. Creating the card images of the configurations (Page 1106)

Note

The following steps can also be automated by means of SCOUT scripting. Appropriate scripts are shown in the appendix in "Creating the card images for configuration servers by means of SCOUT scripting" (Page 1112).

Preparing the memory card of the SIMOTION device

Check the following:

1. The SIMOTION Kernel for the associated SIMOTION device is present on the card.
2. The `INSTALL` directory in the root directory of the card does not contain any files or subdirectories.

Creating the work directories on the PC/PG

You must create the required work directories in the file directory of the PC/PG on which the SIMOTION SCOUT engineering software is installed.

The directory structures for the following applications are described in the following:

1. Activation of a kernel (e.g. update to a new version) with several associated configurations
2. Activation of several kernels (e.g. various firmware versions) each with several associated configurations

Directory structure for a kernel with several associated configurations

You want to activate, for example, a new kernel with three configurations and create the required card images. The following is a detailed description of how to create the required work directories on the PC/PG.

You require a work directory for storing all the generated data (e.g. `D:\modular_machine`). In this directory you require three further directories each with specific subdirectories for each individual configuration:

- One directory (e.g. `download`) for storing the uncompressed (executable) configuration files.
  In this directory, you require a subdirectory with the `DExxxxxx` designation for each configuration, where `xxxxxx` is a hexadecimal number. With this number, you specify the associated configuration for the call of the _activateConfiguration (Page 1078) system function.
- One directory (e.g. `zip`) for storing the compressed configuration files.
  In this directory too, you require a subdirectory with the `DExxxxxx` designation for each configuration, where `xxxxxx` is a hexadecimal number. With this number, you specify the associated configuration for the call of the _activateConfiguration (Page 1078) system function.
- One directory (e.g. `cardfiles`) for storing the card images.
  You must duplicate the directory structure on the memory card in this directory. This is described in Section "Activation of a kernel" (Page 1084).

You create these directories with the corresponding subdirectories with the command line application `u7mkcnfx.exe` as follows:

1. In the Windows Explorer, create a directory that you want to use to store all generated files (e.g. `D:\modular_machine`).
2. Create a batch file (e.g. `make_dirs.bat`).

3. Edit this batch file. Write a command line to start the u7mkcnfx.exe application with the following options:
   - /m"Mode" option: /m"MAKE_DIRS".
   - /l"Load Path" option: Specification of the complete path for the directory for storing the uncompressed configuration files (e.g. /l"d:\modular_machine\download").
   - /s"Source Path" option: Specification of the complete path for the directory for storing the compressed configuration files (e.g. /s"d:\modular_machine\zip").
   - /d"Destination Path" option: Specification of the complete path for the directory for storing the card images (e.g. /d"d:\modular_machine\cardfiles").
   - /i"Identifier" option: Specification of the identifier for the version of the SIMOTION Kernel in the form VEyyyyyy, where yyyyyy is a hexadecimal number (e.g. /i"VE004156"). With this number you specify the kernel version for the call of the _activateConfiguration system function, see Activating a kernel (Page 1084).
   - /n"Number of Projects" option: Specification of the number of configurations for which the subdirectories are to be created.

   Example:
   ```
   rem make_dirs.bat Create work directories
   rem The command must be written in a command line
   u7mkcnfx /m"MAKE_DIRS"
       /l"d:\modular_machine\download"
       /s"d:\modular_machine\zip"
       /d"d:\modular_machine\cardfiles"
       /i"VE004156" /n"3"
   pause
   ```
   The syntax of the command line application u7mkcnfx.exe is described in the appendix in Command line application u7mkcnfx.exe (Page 1109).

4. Execute the batch file.
   The directories for the uncompressed and compressed configuration files as well as the directory for the card images are created. The following is also created:
   - In the directories for the uncompressed and compressed configuration files (e.g. download), the following subdirectories:
     DE000001, DE000002, etc.: Consecutively numbered directories for the files of the associated configuration, in accordance with the number of configurations. The identifier of these directories (DExxxxxx), corresponds to xxxxxx of a hexadecimal number with which you identify the respective configuration when calling the _activateConfiguration system function, see Activating a configuration (Page 1078).
     INITIAL: Directory for the files of the initial configuration
     KERNEL: Directory for storing the kernel
   - In the directory for the card images (e.g. cardfiles), the following INSTALL\SIMOTION subdirectory containing a subdirectory with the identifier specified with the /i"Identifier" option (e.g. VE004156).

   Further directories, not specified here, are created within these subdirectories.

The figure below shows an example of the directory structure for three configurations identified by means of 16#000001, 16#000002, and 16#000003.

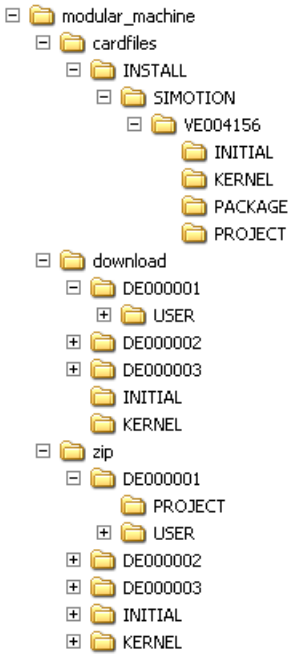


Figure 4-30 Directory structure for one kernel with three configurations

Note

The steps described in this chapter can also be automated by means of SCOUT scripting. A corresponding script is shown in the appendix under "Creating the work directories on the PC/PG" (Page 1113).

Directory structure for several kernel versions with the respective associated configurations

If you want to create the card images for several kernel versions with their respective associated configurations, proceed as follows:

1. Create the work directories (e.g. `download`, `zip`, `cardfiles`) as described above.
2. Create the new directories corresponding to the number of kernel versions and assign them unique names (e.g. `VE004153` and `VE004156`).
3. Copy or move the `download` and `zip` directories to the directories that you have just created (e.g. `VE004153` and `VE004156`).
4. In the `cardfiles\INSTALL\SIMOTION` directory, copy the `VEyyyyyy` directory (e.g. `VE004156`) corresponding to the number of kernel versions. Assign unique names to the copies in the form of `VEyyyyyy`, where `yyyyyy` is a hexadecimal number (e.g. `VE004153`).

The following figure shows the resulting directory structure:

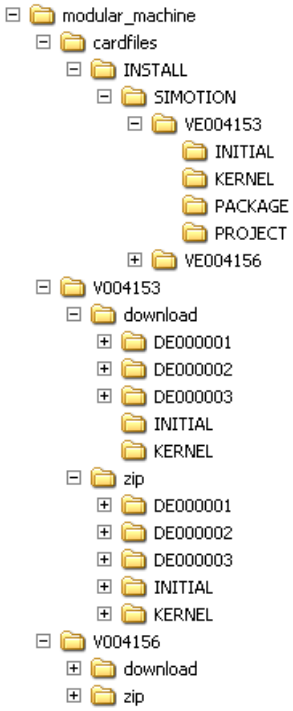


Figure 4-31 Directory structure for two kernels each with three configurations

Saving the configuration files in the work directories on the PC/PG

This section describes how you can save different configurations for a kernel (e.g. for a kernel update) in the work directories on the PC/PG.

Requirements

You have created a project in the SIMOTION SCOUT engineering system in which the required configurations are configured as dedicated devices (with technology objects, programs, etc.).

This project has been compiled without errors.

If an initial configuration (Page 1089) is required, it must also be configured as a dedicated device.

Note

All SIMOTION devices must be configured with the same version of the SIMOTION Kernel.

Procedure

This description applies to:

- All SIMOTION C2xx, P3xx and D4xx-2 devices, regardless of the SIMOTION Kernel version
- The SIMOTION D4xx device, as of SIMOTION Kernel version V4.1.5 (V4.1 SP5)

Note

For the procedure with SIMOTION D4xx devices up to version V4.1.4 (V4.1 SP4) of the SIMOTION Kernel: See the appendix, "Storing configuration files for SIMOTION D4xx devices up to kernel version V4.1.4 (Page 1119)".

To store the configuration files in the work directories on the PC/PG:

1. Select the appropriate device in the project navigator.
2. Select (alternatively):
   - The **Edit > Load to file system** menu or
   - The **Load to file system** context menu
3. In the window, select: **Save normally**.
4. Click the **Select target** button to select the directory.
5. Select the directory for the uncompressed configuration files (e.g. `D:\modular_machine\download`).
   In this directory, select the `DExxxxxx` folder with which you can identify the configuration by the hexadecimal number in the file name (`xxxxxx`), and confirm with **OK**.
   The device with all technology objects and programs is compiled and saved in the specified directory in the `USER` subdirectory.
6. Repeat steps 1 to 5 for all devices, other than the device that represents the initial configuration.
   Save the configuration files of the initial configuration in the `INITIAL` directory.
7. If required, copy the the user administration files for SIMOTION IT to the `DExxxxxx` directories. These files depend on the SIMOTION Kernel version:
   Refer to the appropriate information in this section.

Note

It is absolutely necessary to save configuration files that differ from each other (e.g. different devices or different configuring of the same device) in the folders `DExxxxxx`.
Otherwise, activation of the configuration (Page 1078) with the system function _activateConfiguration will not be possible later.

Note

The steps described in this chapter can also be automated by means of SCOUT scripting. A corresponding script is shown in the appendix at "Storing the configuration files in the file system" (Page 1114).

User administration for SIMOTION IT

The SIMOTION IT user administration depends on the SIMOTION Kernel version:

- The following applies up to SIMOTION Kernel version V4.3:
  The user administration is integrated in the `webcfg.xml` file. This file is on the memory card of the SIMOTION device under the path: `\user\simotion\hmicfg`.
- The following applies as of SIMOTION Kernel version V4.4:
  The user administration is contained in the `userdatabase.xml` file. This file is on the memory card of the SIMOTION device under the path: `\user\simotion\hmicfg \userdatabase`.

The required users must be taken into the files specified above.

The user administration and login administration are described in detail in the "SIMOTION IT Diagnostics and Configuration" Diagnostics Manual of the corresponding SIMOTION Kernel version.

Note

In the following cases, it may be necessary to save the user administration to the `DExxxxxx` directory of the respective configuration:

- The user administration depends on the configuration to be activated.
- The SIMOTION Kernel version changes (particularly for a change from a version up to 4.3 to a version as of 4.4) with the configuration to be activated.

To make sure that the user administration suitable for the SIMOTION Kernel version is activated:

1. Copy the `webcfg.xml` or `userdatabase.xml` files specified above from the memory card (depending on the SIMOTION Kernel version) to a temporary directory on the PC.
   Insert the memory card in a card reader connected to the PC or use the FTP access to the SIMOTION device.
2. If required, edit the files to make the changes to the user administration.
3. Depending on the SIMOTION Kernel version, proceed as follows for each configuration (`DExxxxxx` directory):
   - For a SIMOTION Kernel version up to 4.3:
     In the `DExxxxxx` directory, create the `user\simotion\hmicfg` subdirectory.
     Copy the `webcfg.xml` file to this subdirectory.
   - For a SIMOTION Kernel version as of 4.4:
     In the `DExxxxxx` directory, create the `user\simotion\hmicfg\userdatabase` subdirectory.
     Copy the `userdatabase.xml` file to this subdirectory.

Creating the compressed configuration files

Proceed as follows to compress the configuration files

1. Create a batch file (e.g. `make_zip.bat`).
2. Edit this batch file. Write a command line for each configuration to start the `u7mkcnfx.exe` application with the following options:
   - /m"Mode" option: `/m"MAKE_ZIP"`
   - /s"Source Path" option: Specification of the complete path in which the uncompressed files of the respective configuration are stored (e.g. `/s"d:\modular_machine\download\DE000001"`).
   - /d"Destination Path" option: Specification of the complete path for the directory for storing the compressed configuration files (e.g. `/d"d:\modular_machine\zip"`).
   - /i"Identifier" option: Specification of the subdirectory in which the compressed files of the respective configuration are stored. This name is identical to the name of the corresponding subdirectory for the uncompressed files (e.g. `/i"DE000001"`).
     You use the hexadecimal number in the name to specify the configuration for the call of the _activateConfiguration system function, see Activating a configuration (Page 1078).
     You create the compressed files of the initial configuration with the `/i"INITIAL"` option.

   Example:

```
rem make_zip.bat Compress configuration files
rem he command must be written in a command line
u7mkcnfx /m"MAKE_ZIP"
         /s"d:\modular_machine\download\DE000001"
         /d"d:\modular_machine\zip" /i"DE000001"
u7mkcnfx /m"MAKE_ZIP"
         /s"d:\modular_machine\download\DE000002"
         /d"d:\modular_machine\zip" /i"DE000002"
u7mkcnfx /m"MAKE_ZIP"
         /s"d:\modular_machine\download\DE000003"
         /d"d:\modular_machine\zip" /i"DE000003"

rem Compress files of the initial configuration
u7mkcnfx /m"MAKE_ZIP"
        /s"d:\modular_machine\download\initial"
        /d"d:\modular_machine\zip" /i"INITIAL"
pause
```

   The syntax of the command line application `u7mkcnfx.exe` is described in the appendix in Command line application u7mkcnfx.exe (Page 1109).
3. Execute the batch file.
   The compressed files are created in the specified directories (the `PROJECT.ZIP` file in each `PROJECT` subdirectory).

Note

The steps described in this chapter can also be automated by means of SCOUT scripting. A corresponding script is shown in the appendix in "Creating the compressed configuration files" (Page 1115).

Saving the kernel in the work directory on the PC/PG

You can obtain the kernel (firmware) suitable for the stored configurations in two ways:

- As of version V4.1 of the SIMOTION Kernel, directly from a SIMOTION device that is connected via Ethernet to the PC/PG:
  You can download the kernel from the device with the SIMOTION IT DIAG function package contained in the kernel, which enables direct diagnostics of the SIMOTION device via an HTML standard browser.
  To do this, call the **Manage Config** standard page and also the **Device update** tab. Activate the **FW** checkbox.
  The procedure is described in the "SIMOTION IT Diagnostics and Configuration" Diagnostics Manual.
  You are provided with a ZIP archive (file name *name*.zip) that contains the `KERNEL.ZIP` file.
  SIMOTION P3xx devices do not support kernel download.
- Irrespective of the version of the SIMOTION Kernel as download from the SIMOTION Service and Support pages on the Internet:
  http://support.automation.siemens.com/WW/view/en/10805436
  Rename the ZIP archive that you have downloaded to `KERNEL.ZIP`

The `KERNEL.ZIP` file contains the SIMOTION Kernel of the respective device as well as the SINAMICS Integrated firmware for the SIMOTION D devices.

Proceed as follows:

1. Open the directory for the compressed configuration files (e.g. `D:\modular_machine\zip`).
2. Copy the `KERNEL.ZIP` file to the `KERNEL\PROJECT` subdirectory.

Creating the card images of the configurations

Requirement

A read/write device for the memory card is connected to the PC.

Proceed as follows to create the card images and copy them to the memory card

1. Create a batch file (e.g. `make_store.bat`).
2. Edit this batch file. Write a command line to start the `u7mkcnfx.exe` application with the following options:
   - /m"Mode": `/m"MAKE_STORE"` option.
   - /s"Source Path" option: Specification of the complete path for the directory for storing the compressed configuration files (e.g. `/d"d:\modular_machine\zip"`).
   - /d"Destination Path" option: Specification of the complete path for the directory for storing the card images (e.g. `/d"d:\modular_machine\cardfiles"`).
   - /a"ModeSelfAdaptingConfiguration" option: Specification of whether the initial configuration, in which the appropriate configuration is selected, should be automatically loaded after POWER ON, see Selecting and activating a configuration using an initial configuration (Page 1089):
     `/a"ON"`: The initial configuration is started after each POWER ON.
     `/a"OFF"`: The initial configuration is not started after POWER ON.
     /i"Identifier" option: Specification of the identifier for the version of the SIMOTION Kernel (as described in Creating the work directories on the PC/PG (Page 1098)). The identifier has the form `VEyyyyyy`, where `yyyyyy` is a hexadecimal number (e.g. `/i"VE004156"`). With this number you specify the kernel version for the call of the _activateConfiguration system function, see Activating a kernel (Page 1084).

   Example:

```
rem make_store.bat Create card images
rem The command must be written in a command line
u7mkcnfx /m"MAKE_STORE" /s"d:\modular_machine\zip"
         /d"d:\modular_machine\cardfiles"
         /a"ON" /i"VE004156"
pause
```

   The syntax of the command line application `u7mkcnfx.exe` is described in the appendix in Command line application u7mkcnfx.exe (Page 1109).

3. Execute the batch file.
   The card images are created in the \INSTALL\SIMOTION\VEyyyyyy subdirectory of the directory specified for the card images (e.g. D:\modular_machine\cardfiles):
   - The PROJECT subdirectory contains the card images of the configurations in the form PRxxxxxx.ZIP, where xxxxxx is the hexadecimal number which was defined in the DExxxxxx identifier when the compressed files were created, see Creating the compressed configuration files (Page 1105).
   - The INITIAL subdirectory contains the card image of the initial configuration INITIAL.ZIP.
   - The PACKAGE subdirectory contains the card image of the technology packages.
   - The KERNEL subdirectory contains the card image of the SIMOTION Kernel KERNEL.ZIP.

   A BOOT.INI file is also created in the specified directory (e.g. D:\modular_machine\cardfiles).

   Note

   Existing card images will be deleted and overwritten

   The command line application u7mkcnfx.exe deletes all existing data in the \INSTALL \SIMOTION subdirectory of the directory specified for the card images (e.g. D:\modular_machine\cardfiles) with the /m"MAKE_STORE" option.

   If you want to create the card images for several kernel versions, proceed as follows:
   1. Create the card images for the first kernel version as described above.
   2. Copy the VEyyyyyy directory from the \INSTALL\SIMOTION subdirectory to an arbitrary directory outside this directory structure (e.g. D:\temp).
      You can use the following command line, for example, in the batch file:
      `xcopy /e /f d:\modular_machine\cardfiles\INSTALL\SIMOTION d:\temp`
   3. Create the card images for a further kernel version as described above.
   4. Repeat steps 2 to 3 for each additional kernel.
   5. Copy the VEyyyyyy directories from the temporary directory back to the \INSTALL \SIMOTION subdirectory.
      You can use the following command line, for example, in the batch file:
      `xcopy /e /f d:\temp d:\modular_machine\cardfiles\INSTALL \SIMOTION`

4. Copy the following from the directory of the card images (e.g. D:\modular_machine \cardfiles) to the memory card's root directory:
   - The INSTALL directory with all subdirectories and files
   - The BOOT.INI file

   Overwrite any files and directories that exist.
5. Back up the symbol information of retentive variables if necessary.
   See Retaining retentive data when a kernel is activated (Page 1097).

Note

The steps described in this chapter can also be automated by means of SCOUT scripting. A corresponding script is shown in the appendix in "Creating the card images of the configurations" (Page 1117).

4.1.7 Appendix

4.1.7.1 Command line application u7mkcnfx.exe

Command line application for Windows, with which the card images of the configurations and the SIMOTION Kernel are created in several steps (/m"Mode" option).

Options

The following options are available:

/m"Mode"

Specification of the functional principle of the command line application.

/m"MAKE_DIRS" Creating the work directories

The following additional options must be specified:

- /s"Source Path"
- /d"Destination Path"
- /i"Identifier"

The following options can be specified:

- /l"Load Path"
- /n"Number of projects"

For application information and an example, see Chapter Creating the work directories on the PC/PG (Page 1098) and Creating the work directories on the PC/PG (Page 1113) in the appendix.

/m"MAKE_ZIP" Compressing individual configurations

The following additional options must be specified:

- /s"Source Path"
- /d"Destination Path"
- /i"Identifier"

For application information and an example, see Chapter Creating the compressed configuration files (Page 1105) and Creating the compressed configuration files (Page 1115) in the appendix.

/m"MAKE_ZIP_ALL"	Compressing all configurations The following additional options must be specified: • /s"Source Path" • /d"Destination Path"
/m"MAKE_STORE"	Creating card images The following additional options must be specified: • /s"Source Path" • /d"Destination Path" • /a"Mode Self Adapting Configuration" • /i"Identifier" For application information and an example, see Chapter Creating the card images of the configurations (Page 1106) and Creating the card images of the configurations (Page 1106) in the appendix.
/m"MAKE_STORE_ARCHIVE"	Archiving card images The following additional options must be specified: • /s"Source Path" • /d"Destination Path" • /i"Identifier" For application information and an example, see Creating the card images of the configurations (Page 1117) in the appendix.

/s"Source Path"

Specification of the complete path for the source directory

- For options /m"MAKE_DIRS" and /m"MAKE_STORE"
  Specification of the complete path for the directory for storing the compressed configuration files (e.g. `/s"d:\modular_machine\zip"`).
- For option /m"MAKE_STORE_ARCHIVE"
  Specification of the complete path for the directory for storing the card images (e.g. `/s"d:\modular_machine\cardfiles"`).
- For option /m"MAKE_ZIP"
  Specification of the complete path in which the uncompressed files of the respective configuration are stored (e.g. `/s"d:\modular_machine\download\DE000001"`).
- For option /m"MAKE_ZIP_ALL"
  Specification of the complete path for the directory for storing the uncompressed configuration files (e.g. `/s"d:\modular_machine\download"`).

/d"Destination Path"

Specification of the complete path for the source directory

- For options /m"MAKE_DIRS" and /m"MAKE_STORE"
  Specification of the complete path for the directory for storing the card images (e.g. `/d"d:\modular_machine\cardfiles"`).
- For option /m"MAKE_STORE_ARCHIVE"
  Specification of the complete path for the directory for storing the card images (e.g. `/d"d:\modular_machine\cfes"`).
- For options /m"MAKE_ZIP" and /m"MAKE_ZIP_ALL"
  Specification of the complete path for the directory for storing the compressed configuration files (e.g. `/d"d:\modular_machine\zip")`.

/l"Load Path"

Optional, only for option /m"MAKE_DIRS"

Specification of the complete path for the directory for storing the uncompressed configuration files (e.g. `/l"d:\modular_machine\download"`).

/i"Identifier"

Only for options m/"MAKE_DIRS", /m"MAKE_STORE" and m/"MAKE_ZIP

- For options /m"MAKE_DIRS", /m"MAKE_STORE" and /m"MAKE_STORE_ARCHIVE"
  Specification of the identifier for the version of the SIMOTION Kernel. The identifier has the form `VEyyyyyy`, where `yyyyyy` is a hexadecimal number (e.g. `/i"VE004156")`. With this number you specify the kernel version for the call of the _activateConfiguration system function (see Activating a kernel (Page 1084)).
- For option /m"MAKE_ZIP"
  Specification of the subdirectory in which the compressed files of the respective configuration are stored. This name is identical to the name of the corresponding subdirectory for the uncompressed files (e.g. `/i"DE000001")`.
  You use the hexadecimal number in the name to specify the configuration for the call of the _activateConfiguration system function (see Activating a configuration (Page 1078)).
  You create the compressed files of the initial configuration with the `/i"INITIAL"` option.

/a"Mode Self Adapting Configuration"

Only for option /m"MAKE_STORE"

Specification of whether the initial configuration, in which the appropriate configuration is selected, should be automatically loaded after POWER ON (see Selecting and activating a configuration using an initial configuration (Page 1089)).

/a"ON" The initial configuration is started after each POWER ON.

/a"OFF" The initial configuration is not started after POWER ON.

/n"Number of projects"

Only for option /m"MAKE_DIRS"

Specification of the number of subdirectories to be created for the individual configurations. For `/n"3"`, for example, three subdirectories `DE000001`, `DE000002` and `DE000003` are created. These configurations are identified by the hexadecimal numbers 16#00001, 16#00002 and 16#00003.

4.1.7.2 Creating the card images for the configuration server by means of SCOUT scripting

This function is available in SIMOTION Kernel as of version V4.1.

The steps for creating the card images of configurations (as described in Procedure for creating the card images of configurations (Page 1098)) can be largely automated by means of SCOUT scripting.

After you have set up the folder for the scripts (Creating the script folder in the SCOUT project (Page 1112)), create the individual scripts for the following tasks:

- Creating the work directories on the PC/PG (Page 1113)
- Saving the configuration files in the work directories on the PC/PG (Page 1114)
- Creating the compressed configuration files (Page 1115)
- Creating the card images of the configurations (Page 1117)

Script to call up the individual scripts to create the card image (Page 1118) in the appendix defines another script which calls up the ones specified above.

Creating the script folder in the SCOUT project

First, you must set up the folder in the engineering system (e.g. SIMOTION SCOUT) in which the scripts will be stored.

Proceed as follows:

1. Select the project in the project navigator.
2. Select the **Expert > Insert script folder** context menu.
   The SCRIPTS folder is created in the project. The scripts defined in the following sections are stored in this folder.

Proceed as follows to create a script in this folder:

1. Select the folder.
2. Select the **Insert new object > Script** context menu.
3. Specify the name.
4. Confirm with **OK**.

Creating the work directories on the PC/PG

The following script "createDirsModMach" deletes any directories present in the `d:\modular_machine` directory and also sets up the required directories there, see also Creating the work directories on the PC/PG (Page 1098).

```
' createDirsModMach

App.LogActive = True
APP.LogMode = 1

Set objFso = CreateObject("Scripting.FileSystemObject")

basePath = "d:\modular_machine"

App.PrintToLog "remove dirs ...\cardfiles ..." _
    & "\download ...\zip ...\cfes"
If (objFso.FolderExists(basePath)) Then
    App.PrintToLog "delete folder basePath"
    call objFso.deleteFolder(basePath)
End If

If not (objFso.FolderExists(basePath)) Then
    App.PrintToLog "create basePath ...\modMach"
    Call objFso.createFolder(basePath)
End If
createPath = basePath & "\cardfiles"
App.PrintToLog "path: " & createPath

If not (objFso.FolderExists(createPath)) Then
    App.PrintToLog "create folder ...\cardfiles"
    Call objFso.createFolder(createPath)
End If
createPath = basePath & "\cfes"
App.PrintToLog "path: " & createPath

If not (objFso.FolderExists(createPath)) Then
    App.PrintToLog "create folder ...\cfes"
    Call objFso.createFolder(createPath)
End If
createPath = basePath & "\download"
App.PrintToLog "path: " & createPath

If not (objFso.FolderExists(createPath)) Then
    App.PrintToLog "create folder ...\download"
    Call objFso.createFolder(createPath)
End If
createPath = basePath & "\zip"
App.PrintToLog "path: " & createPath

If not (objFso.FolderExists(createPath)) Then
    App.PrintToLog "create folder ...\zip"
    Call objFso.createFolder(createPath)
End If
```

```
createPath = basePath & "\export"
App.PrintToLog "path: " & createPath

If not (objFso.FolderExists(createPath)) Then
    App.PrintToLog "create folder ...\export"
    Call objFso.createFolder(createPath)
End If

App.LogActive = False
```

Saving the configuration files in the work directories on the PC/PG

The following "downl" script stores the configuration files of the configured device in the `download` directory, see also Storing the configuration files in the file system (Page 1102).

Note

The SIMOTION IT user administration depends on the SIMOTION Kernel version.

Note the information for the user administration in Section Storing the configuration files in the file system (Page 1102).

```
' downl
' download project-data into filesystem of hd

App.LogActive = True
APP.LogMode = 1

'App.PrintToLog "update project"
basePath = "d:\modular_machine"

App.PrintToLog "download into filesystem"
'APP.Time 1
App.PrintToLog "D435_1"
downloadPath = basePath & "\download\1"
call PROJ.Devices("D435_1").DownloadFileSystem _
    ( downloadPath, "overwrite" ) ', FALSE )
App.PrintToLog "D435_2"
downloadPath = basePath & "\download\2"
call PROJ.Devices("D435_2").DownloadFileSystem _
   ( downloadPath, "overwrite" ) ', FALSE )
App.PrintToLog "D435_3"
downloadPath = basePath & "\download\3"
call PROJ.Devices("D435_3").DownloadFileSystem _
    ( downloadPath, "overwrite" ) ', FALSE )
```

```
App.PrintToLog "D435_initial"
downloadPath = basePath & "\download\initial"
call PROJ.Devices("D435_initial").DownloadFileSystem _
    ( downloadPath, "overwrite" ) ', FALSE )

App.PrintToLog "end script download"
App.LogActive = False
```

Creating the compressed configuration files

The following script "makezip" compresses the configuration files present in the `download` directory and stores them in the `zip` directory, see also Creating the compressed configuration files (Page 1105).

```
' makezip

Dim WshShell, oExec', script

Set WshShell = CreateObject("WScript.Shell")

basePath = "d:\modular_machine"
App.LogActive = True
App.PrintToLog "make_zip 1"
App.LogActive = false

sourcePath = basePath & "\download\1"
destinationPath = basePath & "\zip"

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_ZIP""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""DE000001""")

' needed !!! for waiting end execute zip
Do While oExec.Status = 0
    APP.time 1
Loop
```

```
App.LogActive = True
App.PrintToLog "make_zip 2"
App.LogActive = false

sourcePath = basePath & "\download\2"
destinationPath = basePath & "\zip"

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_ZIP""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""DE000002""")

' needed !!! for waiting end execute zip
do While oExec.Status = 0
    APP.Time 1
loop
App.LogActive = True
App.PrintToLog "make_zip 3"
App.LogActive = false

sourcePath = basePath & "\download\3"
destinationPath = basePath & "\zip"
'   App.LogActive = false

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_ZIP""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""DE000003""")

' needed !!! for waiting end execute zip
do While oExec.Status = 0
    APP.Time 1
loop
App.LogActive = True
App.PrintToLog "make_zip initial"
App.LogActive = false

sourcePath = basePath & "\download\initial"
destinationPath = basePath & "\zip"

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_ZIP""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""INITIAL""")

'   needed !!! for waiting end execute zip
do While oExec.Status = 0
    APP.Time 0.1
loop
```

Creating the card images of the configurations

The following script "makestore" creates from the compressed configuration files in the `zip` directory the corresponding card images and stores them in the `cardfiles` directory. These card images are then archived in the `cfes` directory, see also Creating the card images of the configurations (Page 1106).

```
' makestore

Dim WshShell, oExec', script

Set WshShell = CreateObject("WScript.Shell")

basePath = "d:\modular_machine"
App.LogActive = True
App.PrintToLog "make_store"
App.LogActive = false

sourcePath = basePath & "\zip"
destinationPath = basePath & "\cardfiles"

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_STORE""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""VE000041""" _
    & " /a""ON""")

' needed !!! for waiting end execute zip
do While oExec.Status = 0
    countTime = countTime + 1
loop
App.LogActive = True
App.PrintToLog "make_archive"
App.LogActive = false

sourcePath = basePath & "\cardfiles"
destinationPath = basePath & "\cfes"

Set oExec = WshShell.Exec _
("C:\Program Files\siemens\step7\s7bin\u7mkcnfx " _
    & " /m""MAKE_STORE_ARCHIVE""" _
    & " /s""" & sourcePath & """" _
    & " /d""" & destinationPath & """" _
    & " /i""ARCH0041""")

' needed !!! for waiting end execute zip
do While oExec.Status = 0
    APP.Time 1
loop

App.LogActive = True
App.PrintToLog "ende make_archive"
App.LogActive = false
```

The card images in the `cardfiles` directory and the `BOOT.INI` file can then be transferred to the memory card of the SIMOTION device via a read/write device connected to the PG/PC (Creating the card images of the configurations (Page 1106)).

The archived card images in the `cfes` directory as well as the `BOOT.INI` file can be transferred directly to the corresponding SIMOTION device. Corresponding scripts are shown in the appendix under Loading the card image to the target device by means of SCOUT scripting (Page 1118).

Script to call up the individual scripts to create the card image

The individual scripts mentioned above are executed by a central script. In this, only the actions that run in the engineering system are summarized (no target required). This is possible by means of the syntax listed below:

```
'   allToDo

WBScript.addCode(Scripts("createFoldersModMach").code)
WBScript.addCode(Scripts("downl").code)
WBScript.addCode(Scripts("makezip").code)
WBScript.addCode(Scripts("makestore").code)
```

4.1.7.3 Loading the card image to the target device by means of SCOUT scripting

This function is available in SIMOTION Kernel as of version V4.1.

The following scripts show examples of how an archived card image stored in the subfolder of the `d:\modular_machine\cfes` directory can be transferred to the corresponding target device.

For other target devices, the scripts are to be changed accordingly.

After loading the card images, the SIMOTION devices must be restarted in order for the card images to become effective as the active configuration.

Setting up an online connection to the target device

The following script "online" sets up an online connection to the SIMOTION device with the identifier "D435_1."

```
' online

' connection to device "D435_1"

App.LogActive = True
PROJ.Devices("D435_1").EnableOnline = TRUE
PROJ.Online = true
```

Loading the card image to the target device

The following script "dwldtarg" loads the archived card image of the configuration into the SIMOTION device with the identifier "D435_1."

```
' dwldtarg

' enable log activities
App.LogActive = True
App.PrintToLog "begin download into filesystem of target"

basePath = "d:\modular_machine"
bootIniPath = basePath _
    & "\cfes\INSTALL\SIMOTION\BOOT.INI"
archivePath = basePath _
    & "\cfes\INSTALL\SIMOTION\ARCH0041.ZIP"

call PROJ.Devices _
    ("D435_initial").DownloadConfigServerData( _
    bootIniPath, _
    archivePath )

App.PrintToLog "end download into filesystem of target"

' disable log activities
App.LogActive = False
```

Ending the online connection to the target device

The following script "online" breaks the online connection to the SIMOTION device with the identifier "D435_1."

```
' offline

PROJ.Devices("D435_1").EnableOnline = false
PROJ.Online = false
```

4.1.7.4 Storing configuration files for SIMOTION D4xx devices up to kernel version V4.1.4

Requirements

1. You have created a project in the SIMOTION SCOUT engineering system in which the required configurations are configured as dedicated devices with technology objects, programs, etc.
2. This project has been compiled without errors.
3. If an initial configuration (Page 1089) is required, it must also be configured as a dedicated device.

4. The following options are **deactivated** (**Options > Settings** menu **Download** tab) in the SIMOTION SCOUT engineering system:
   – Compile all programs before loading
5. The PC with the SIMOTION SCOUT engineering system is connected to the SIMOTION device (e.g. via PROFIBUS).
6. The PC is included in the network of the SIMOTION device (e.g. with NetPro).
7. A read/write device for the memory card is connected to the PC.

Procedure

1. Insert the memory card in the SIMOTION device and switch it on.
2. Execute the following steps in succession in the SIMOTION SCOUT engineering system:
   – Select the appropriate device in the project navigator.
   – Select the **Edit > Open object** menu to open the HW configuration of the device.
   – In the HW Config program, select the **Target system > Load to module** menu to load the hardware configuration to the device.
   Then exit the HW Config program.
   The appropriate device is still selected in the project navigator of the SIMOTION SCOUT engineering system.
   – In the SIMOTION SCOUT engineering system, select the **Project > Connect to target system** menu.
   – Select the **Target system > Load > Load CPU / drive unit to target device** menu to load the configuration data to the device.
   – Select the **Target system > Copy RAM to ROM** menu to store this data on the memory card.
   – Select the **SINAMICS_Integrated** element below the respective device in the project navigator.
   – Select the **Target system > Load > Load CPU / drive unit to target device** menu to load the drive data to the device.
   – Select the **Target system > Copy RAM to ROM** menu to store this data on the memory card.
   – Select the **Project > Disconnect from target system** menu.
3. Switch off the SIMOTION device and remove the memory card.
4. In the Windows Explorer in the directory for uncompressed configuration files (e.g. `D:\modular_machine\download`), select the `DExxxxxx` folder whose hexadecimal number you want to use to identify the configuration in the file name (`xxxxxx`).

5. Copy the configuration files from the memory card to the selected directory:
   – Insert the memory card into a card reader on your PC.
   – Copy the USER directory from the memory card (e.g. B:\USER) to the selected directory DExxxxxx. Overwrite any existing USER directory.
6. Repeat steps 1 to 5 for all devices, other than the device that represents the initial configuration.
   Proceed as described above for the initial configuration, but copy the USER directory from the memory card (e.g. B:\USER) to the INITIAL directory.

4.2 Basic functions

Preface

Content

This document is part of the **System and Function Descriptions documentation package**.

Scope of validity

This manual is valid for SIMOTION SCOUT product version V5.4:

- SIMOTION SCOUT V5.4 (engineering system of the SIMOTION product family)
- SIMOTION Kernel V5.4, V5.3, V5.2, V5.1, V4.5, V4.4, V4.3, V4.2, V4.1, V4.0, V3.2
- SIMOTION technology packages CAM, PATH (as of kernel V4.1), CAM_EXT and TControl in the version for the associated kernel.

Sections in this manual

This manual describes the generally applicable functions of SIMOTION and technology objects.

- **System overview**
  General information on SIMOTION.
- **Technology Packages and Technology Objects**
  Basic information on the technology packages and the technology objects.
- **Symbolic assignment**
  Information on symbolic assignment of SINAMICS objects to SIMOTION.
- **Programming of Technology Objects**
  Information on how and with which system functions technology objects can be programmed.
- **Error Handling in Technology Objects**
  General information on technological alarms and command return values.

- **Execution System, Tasks, and System Cycles**
  Information on the execution system and tasks.
- **Programming Execution System / Tasks / System Cycles**
  Information on the programming of the execution system, tasks, and system cycles.
- **Programming of General System Functions**
  Information on which general system functions exist and how they are used.
- **Programming of General System Function Blocks**
  Information which general system function blocks exist and how they are used.
- **SIMOTION Memory Concept (in the target device)**
  Information on the memory concept in the target device.
- **Downloading Data to the Target Device**
  Information on downloading data to the target device or target system.
- **Appendix**
  Information on symbolic constants and identifiers
- **Index**
  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.2.1 Fundamental safety instructions

4.2.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.2.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.2.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.2.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.2.2 System overview

4.2.2.1 System architecture

SIMOTION system architecture

The SIMOTION system architecture enables trends towards decentralization, heterogeneous target systems, and distributed intelligence to be implemented.

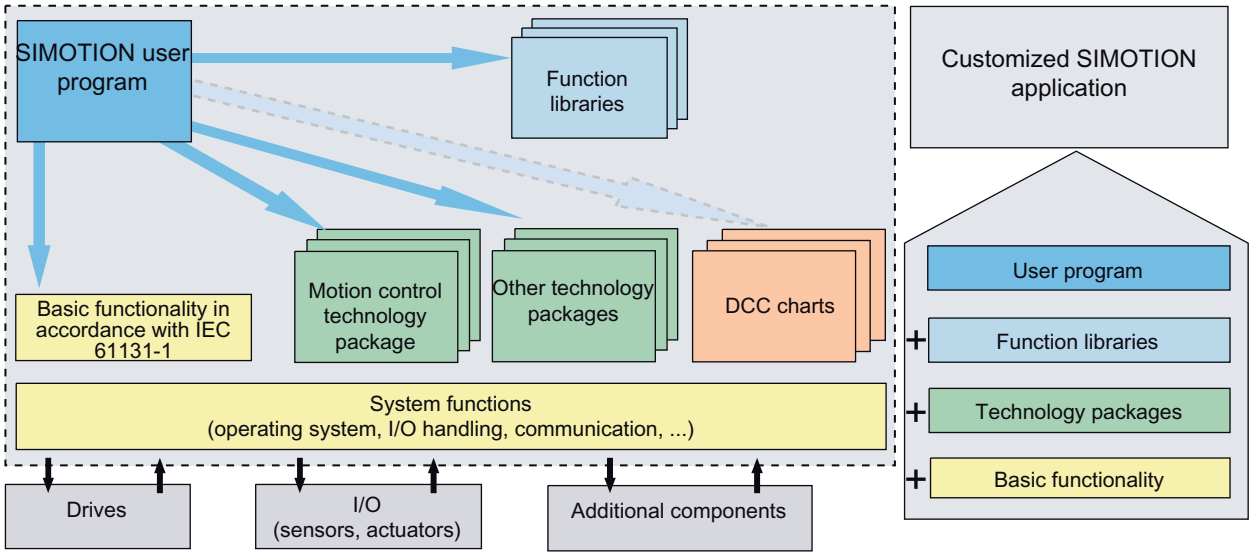


Figure 4-32 SIMOTION system architecture

The basic functionality of the device (SIMOTION Kernel) includes functions for open-loop and closed-loop control as well as logic and arithmetic. Program execution can be cyclical, time-triggered or interrupt-triggered. As a result, the SIMOTION Kernel contains the functions needed for virtually all applications and corresponds in essence to a PLC with the IEC 61131-3 command set plus system functions for controlling various components, such as inputs and outputs.

You can expand the SIMOTION Kernel of the device by downloading technology packages. You access the technology packages from the user program using special language commands, in the same way that you access the SIMOTION Kernel.

For particular tasks, you can either use existing applications or you can program and link the required applications yourself. The applications are programmed in compliance with IEC 61131-3 and can be adapted to your specific task.

In addition, SIMOTION provides function libraries that include the system functions and motion functions. The function libraries contain functions and access to system variables of a technology object and are linked to the associated device and technology package in SIMOTION SCOUT.

For special tasks, such as closed-loop control functions, you can use wiring diagrams and execute them in the corresponding DCC execution tasks using blocks connected with a graphical tool.

SIMOTION SCOUT Engineering System

The following activities are required for machine automation:

- Selection/configuration of the hardware and software components, configuration of hardware and software, including the communication networks with **HW Config**
- Creation and configuration of the technology objects
- Creation of the user program
- Testing and commissioning of the drive units
- Linking of the machine operator control (HMI)
- Concluding steps, such as the generation of machine documentation.

The **SIMOTION SCOUT** engineering system offers a uniform user view and flexible functionality.

Individual automation tasks for production machines are formulated in a uniform and consistent user interface.

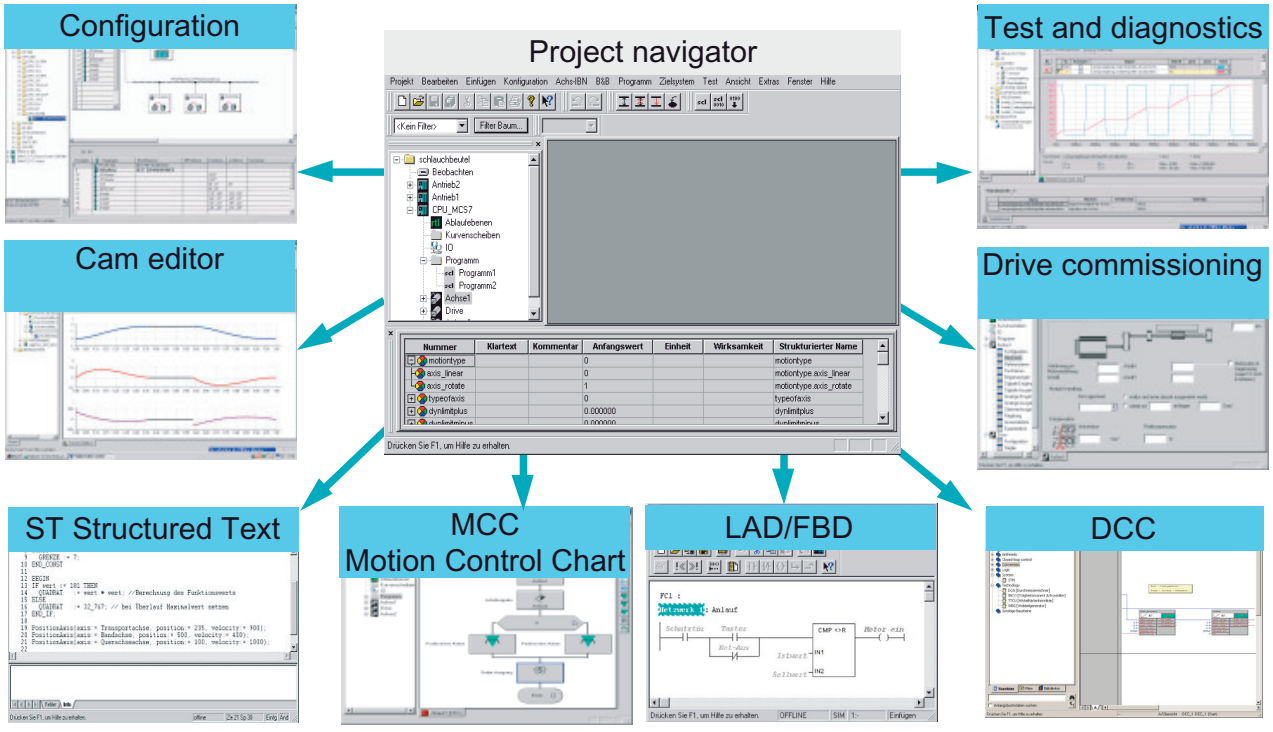


Figure 4-33 SIMOTION SCOUT engineering system

The **SIMOTION SCOUT** engineering system is a powerful tool that acts as the PC development environment to provide optimal support for the required engineering steps in a user-friendly way. It is integrated in the SIMATIC landscape, where it operates as an option package for **STEP7**. Special attention has been given to optimum usability and a comprehensive, function-oriented view of the automation task.

SIMOTION project

The **project** includes all SIMOTION devices, drives, etc. The associated devices are hierarchically assigned to technology objects and programs. This hierarchical structure is displayed in the **Project navigator**.

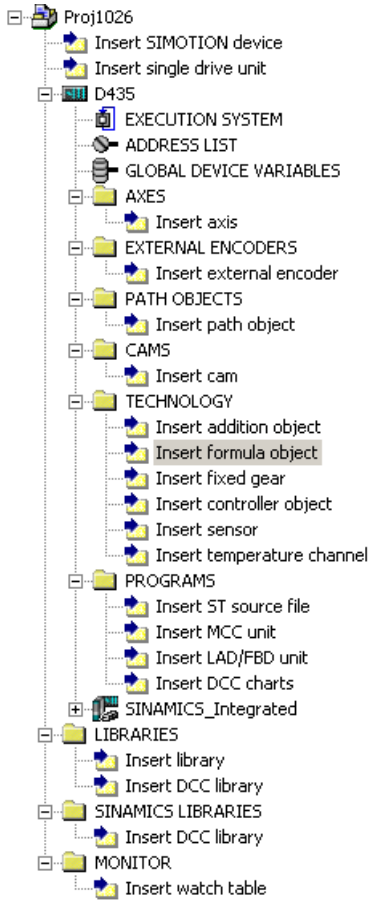


Figure 4-34 SIMOTION SCOUT project navigator

The project is the highest level in the data management hierarchy. SIMOTION SCOUT saves all data which belongs, for example, to a production machine, in the project directory.

Offline/online mode

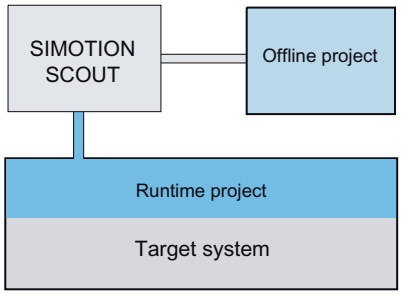


Figure 4-35 Connection between offline and runtime project with SIMOTION SCOUT

SIMOTION SCOUT can be used in two modes:

- In **offline mode**, you can create projects and user programs.
  In offline mode, you work with the SCOUT engineering system without connection to the runtime system. The data and technology objects are maintained in the engineering system. The offline mode is the *leading* work mode, i.e. changes of system variables and configuration data should be made offline.
- In **online mode**, you can load projects and data into the target system, control and monitor the application.
  The data in the SCOUT engineering system is retained as offline project.
  Configuration data changed online can be reloaded into the offline project; this is not possible with system variables changed online.

Programming

The open-loop control and motion control tasks are predefined in the user program.

The following programming languages are available to create your user programs:

- **MCC - Motion Control Chart**
  Is a graphical programming language for programming logic and motion control functions. MCC charts are created.
- **ST - Structured Text**
  Is a text-based programming language compliant with IEC 61131-3 that enables you to create an ST source that can comprise several programs.
  You can also edit an ST source file using an external ST editor.
  Programs created using MCC can be converted to ST, but *not vice versa*.
- **LAD/FBD - Ladder Logic / Function Block Diagram**
  Are graphics-based programming languages in compliance with IEC 61131-3 for programming logic as well as motion control via PLCopen blocks.
- **DCC - Drive Control Chart**
  Is a graphic programming language for programming drive functionality.
  It is possible to edit a Drive Control Chart with an external DCC editor.
- **CLib Studio (C/C++ programming)**
  Is a creation environment in Windows to program functions / function blocks in C/C++ that are stored in user libraries.
  When imported into SIMOTION SCOUT, they can be used in all SIMOTION languages (MCC, ST, LAD/FBD, DCC).

Detailed information for the programming languages is provided in the Programming and Operating Manuals **SIMOTION MCC**, **SIMOTION ST**, **SIMOTION LAD/FBD** and **SINAMICS / SIMOTION DCC editor description**.

Programming in SIMOTION provides the following advantages:

- Programs of different programming languages can be used in one project (**MCC**, **ST** and **LAD/FBD**)
- Programming is independent of the hardware platform
- PLC, motion control and technology can be implemented in one program
- There are functions for direct access to drive parameters available via PROFIDRIVE

Modular machines are supported by the following:

- Modular software development with libraries
- Division into individual machine modules
- Activation/deactivation of DP slaves and technology objects
- Commands for the synchronization

Execution system

SIMOTION provides an execution system with a cyclic task as well as sequential, time-controlled, isochronous, and event-triggered tasks. User programs may be "hooked into" each task; it is even possible for multiple programs to be "hooked into" a task.

Each block is able to use the entire range of SIMOTION functions, i.e. logic, motion, and technology functions.

There is no need to program calls, coordination/synchronization, and monitoring for programs; instead, these functions can be entirely delegated to the system.

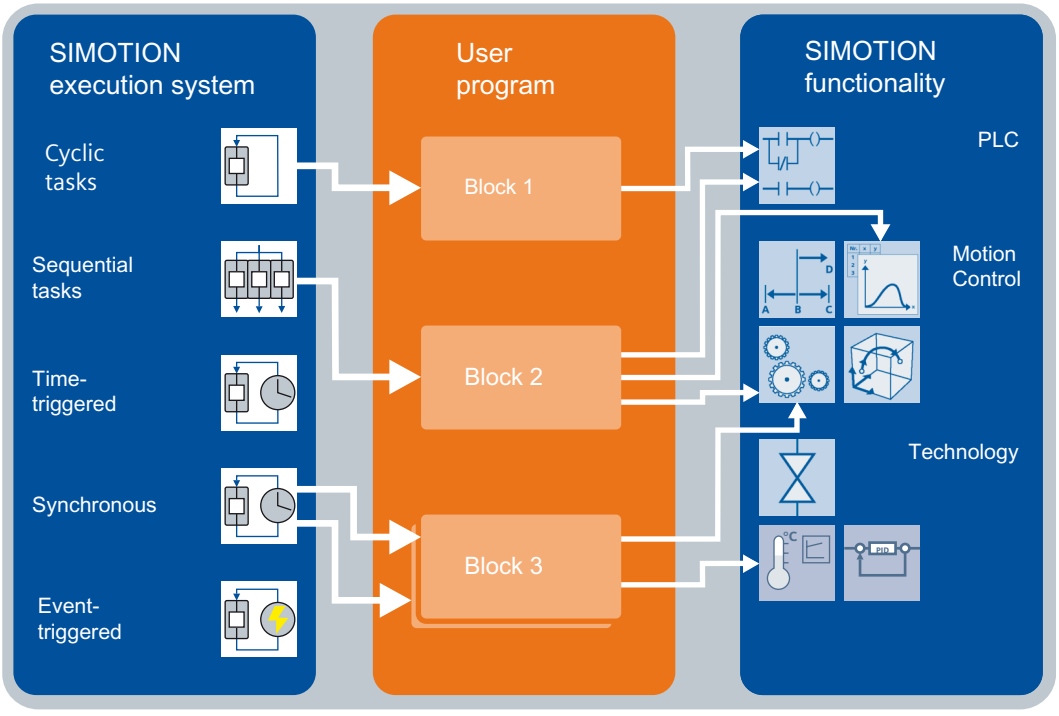


Figure 4-36 SIMOTION execution system

4.2.2.2 SIMOTION motion control

SIMOTION offers an optimized system platform for automation and drive solutions with emphasis on motion control applications and technology tasks.

SIMOTION is the answer to the latest trends in production machines:

- **Mechatronics**
  Mechatronics means that a machine is regarded not only from the mechanical aspect, but as a complete system in which mechanical components, electrical components, and control and software technologies are integrated equally. As part of mechatronics, relatively inflexible mechanical components (such as cams, gears, couplings, line shafts, etc.) are replaced by intelligent software solutions.
- **Fusion of PLC functionality and motion control technology**
  The historical separation of pure automation functions and motion functions has been eliminated. These functions are combined both on the hardware and on the software side.
- **Usability**
  A uniform engineering system (SIMOTION SCOUT) ensures consistency in configuration, parameterization and programming. Automation tasks and motion control tasks are programmed in the same language.
- **Standards**
  Industrial automation is increasingly governed by standards in the PC world such as Microsoft Windows and Ethernet. Standardized global programming languages have greatly facilitated system handling for the customer.
- **Modular machine concepts**
  The trend towards standardization is also encompassing machine designs. As a consequence, attempts are being made to break down machine design into various subcomponents. By virtue of this modularity, it is possible to standardize individual subcomponents and install them as *standard* components in different types of machines.

4.2.2.3 Fields of application

An efficient control system requires that today's tasks and future trends in the field of production machines are implemented using open control concepts.

SIMOTION is a uniform **motion control system** that focuses on the automation of production machines. The uniformity involves engineering, programming, communication, data management and human machine interface (HMI), and thus encompasses the whole system - even on different hardware platforms.

These trends relate primarily to the following mechanical engineering sectors, for which our motion control system is particularly well-suited:

- Packaging machines
- Plastics processing machines
- Metal forming technology
- Textile machines
- Printing machines
- Machines used in the wood, glass, and ceramics industries
- And other machines

In addition to the standard logic and drive-related tasks, the automation solutions in these sectors are also increasingly requiring the incorporation of integrated, uniform motion control and technology tasks.

The SIMOTION modular system consists of the SCOUT engineering system and a common runtime system for various hardware platforms. The SCOUT engineering system is identical for all hardware platforms. Configuration, parameter assignment, and programming are performed using graphics-based or text-based methods. This also provides the basis for sector-specific solutions.

You decide which runtime software (position, gear, ...) to load to enhance the basic functions based on your application. The fact that SIMOTION runs on a variety of hardware platforms makes it a versatile solution that satisfies all requirements

4.2.2.4 Fusion of PLC and motion control

SIMOTION combines open-loop control, technology and motion control. Thus, there are no hardware and software interfaces between the controls required.

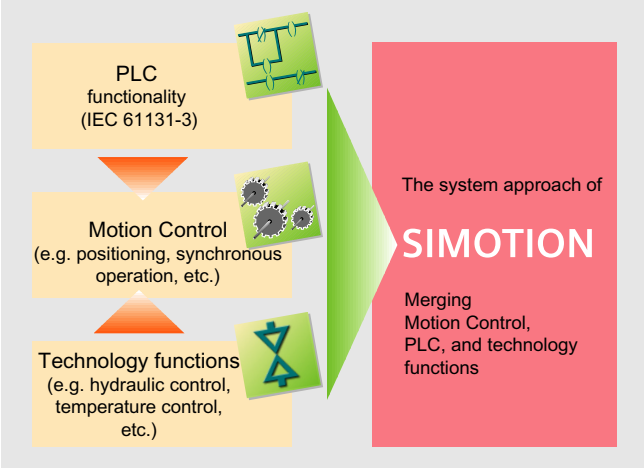


Figure 4-37 Fusion of PLC and motion control

On the hardware side, this means that the programmable controller is able to process motion functions. The hardware platform can be selected individually and hardware components reduced.

On the software side, the fusion of automation functions and motion functions makes for simpler engineering. This starts with the configuration and continues through parameter assignment and programming.

This fusion also means there are no delay times, which results in quick responses and higher cycle rates compared with systems that have separate tasks/CPUs for motion control and the PLC. Not only this, but it is also possible to determine response times more accurately, ensuring high reproducibility of machine behavior and thus of products. In turn, this has a direct impact on product quality.

Consistency with SIMATIC is another essential feature, since both systems can be used together in one plant.

4.2.2.5 Totally Integrated Automation

Threefold consistency in programming and configuration, data management, and communication is at the heart of Totally Integrated Automation: In this way, every automation task solved by SIMATIC can also make full use of the many advantages of Totally Integrated Automation:

- Marked reduction in engineering costs
- No system breaks within the automation landscape
- One software basis for all components

It makes no difference whether your automation task involves implementation of customized solutions in machines or systems, or small-scale automation tasks. Totally Integrated Automation with SIMATIC includes all of the technologies that your automation landscape requires, including programmable controllers, PC-based control, automation computers, distributed I/O, HMI systems, communication networks or process control systems. Because of its modularity, you can use one complete, uniform system to implement the exact solution that satisfies your process requirements and is economically feasible.

SIMOTION and SIMATIC are integrated in the sense of Totally Integrated Automation. This is achieved by integrating SIMOTION SCOUT into SIMATIC Manager and by adopting the same engineering philosophy for comparable activities. Engineering processes that are not available in SIMATIC (motion control, output cam controller, etc.) or that are required by SIMOTION to meet the demands of a distributed system are selected based on optimal usability. Moreover, SIMOTION is integrated into PROFINET and includes all of the requisite system features for this purpose.

4.2.2.6 Hardware platforms

SIMOTION supports various hardware platforms. The choice of hardware components is largely driven by your requirements. You can also distribute your automation tasks to different target systems.

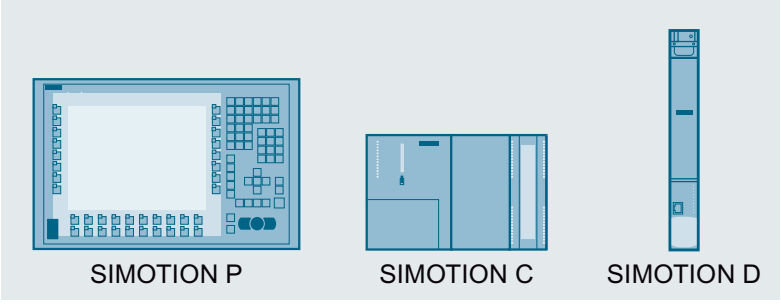


Figure 4-38 SIMOTION hardware platforms

The following platforms are available:

- PC-based (**SIMOTION P**)
  As a PC-based motion control system, SIMOTION P works with the Windows XP operating system, equipped with a real-time expansion for SIMOTION. Apart from SIMOTION applications, other PC applications can also be started.
  SIMOTION P is suitable for:
  - Applications requiring an open architecture to the PC world
  - Applications requiring hardware-based control and visualization
  - Extensive data storage, evaluation and logging
- Controller-based (**SIMOTION C**)
  This controller in SIMATIC S7-300 mounting technology comprises integrated analog drive interfaces and several digital inputs and outputs.
  Moreover, it can be expanded by I/O modules from the SIMATIC S7-300 product range. Two PROFIBUS connections with PROFIdrive interface and one Industrial Ethernet connection allow for communication with other machine parts.
  PROFIBUS can also be used for communication with operator panels (such as those from the SIMATIC HMI range) or with higher-level controls such as SIMATIC S7.
  SIMATIC HMI panels as well as PCs with WinCC flexible or OPC interfaces are suitable as operator systems.
  SIMOTION C enables:
  - Freedom in the selection of the drives
  - Wide range of process signals
  - Retrofit applications via integrated analog interfaces
  - Direct connection of analog drives and stepper drives
- Drive-based (**SIMOTION D**)
  SIMOTION D is a compact, drive-based version of SIMOTION based on the SINAMICS S120 drives family.
  The SIMOTION D Control Units are available in the following variants:
  - SIMOTION D410-2 are compact Control Units for single-axis applications with multi-axis option. The Control Units are available in variants D410-2 DP and D410-2 DP/PN and are snapped onto the SINAMICS S120 PM240-2/PM340 Power Modules in blocksize format. In order to create a multi-axis grouping with SIMOTION D410-2, additional SINAMICS S110/S120 Control Units are connected to the SIMOTION D410-2 by means of PROFIBUS or PROFINET. Motion control is performed centrally by the SIMOTION D410-2 using the SIMOTION technology objects.
  - SIMOTION D4x5-2 are Control Units for multi-axis applications in the SINAMICS S120 booksize format and are available in the four performance variants (D425-2, D435-2, D445-2 and D455-2).
    A SIMOTION D4x5-2 axis grouping comprises a SIMOTION D4x5-2 Control Unit, a Line Module (infeed unit) as well as one or more Motor Modules (power units) to control the motors.
    The fine scalability of SIMOTION D ensures a quick response to changing requirements in automation without having to change the system.

4.2.3 Technology Packages and Technology Objects

4.2.3.1 Introduction

SIMOTION basic functionality

A SIMOTION system without technology packages and without technology objects provides the execution system and basic functionality of a control system in accordance with IEC 61131-3.

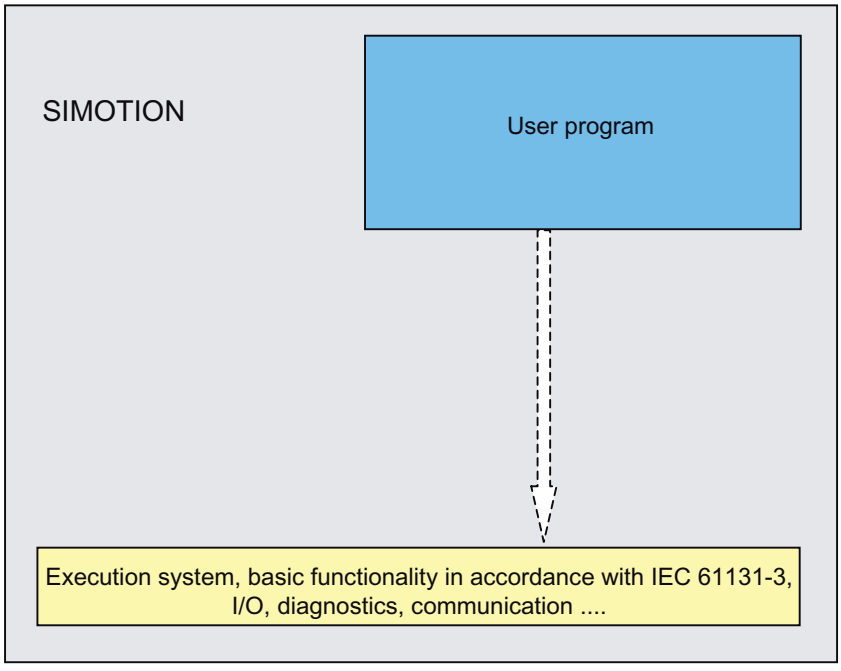


Figure 4-39 SIMOTION basic functionality

The user programs can arbitrarily be assigned to the various tasks.

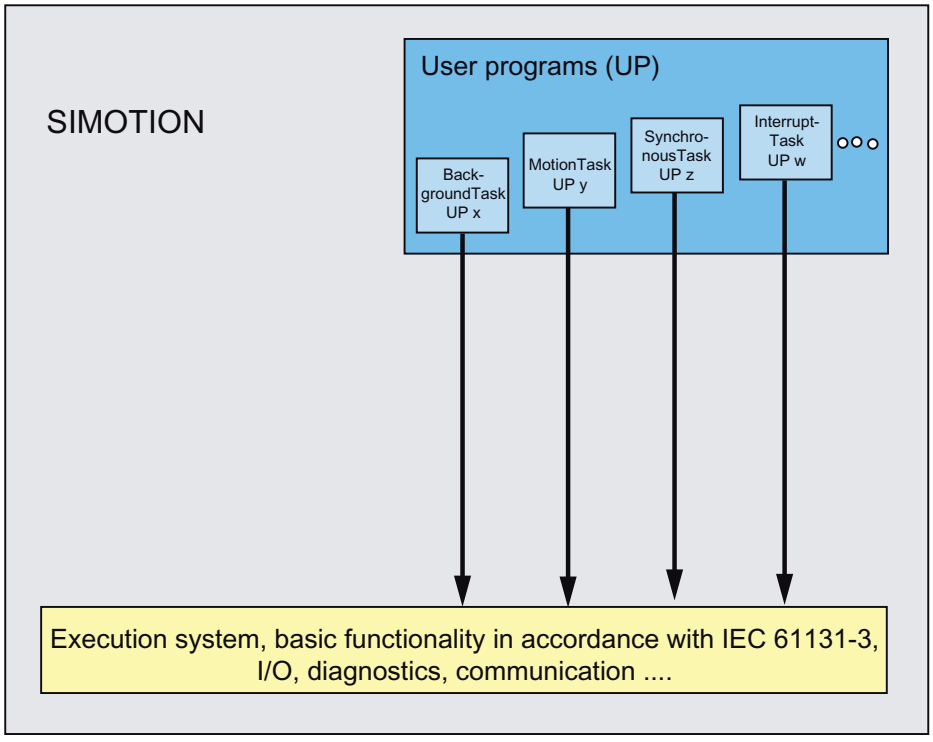


Figure 4-40 Assignment of the user programs to tasks

Technology Objects

The SIMOTION runtime system is implemented in an object-oriented rather than a function-oriented manner, i.e. independent technology objects (TO) are used. These technology objects are available for technology and motion control. Technology objects have a high degree of functionality integrated into them. For example, a TO axis contains the capability for communication with the drive, as well as measured value processing, position control, and positioning functions.

The TOs are created and parameterized by means of configuration and then run automatically in the core of the system. Their functions are called in the user program using appropriate commands. The TOs then process the jobs independently and provide relevant status messages (in a similar way to a driver).

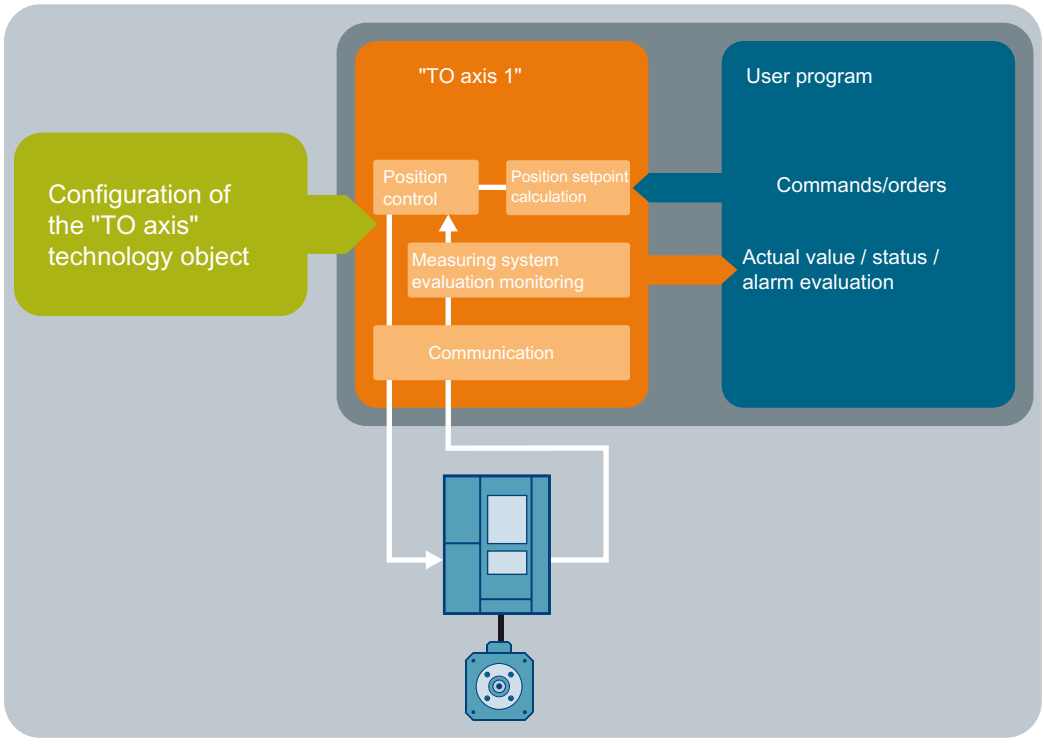


Figure 4-41 SIMOTION technology objects

In principle, there are no limits concerning the number of objects; in other words, you can have any number of axes, synchronous operation links, output cams/cam tracks at an axis, external encoders as master values, etc. The TOs can be activated or deactivated from the program. This enables straightforward adaptation to various machine configurations (e.g. if an axis is not present in a particular configuration).

The technology objects offer the user a technological view of actuators and sensors and provide technological functions for these, for example:

- the TO axis for drive and encoder
- the TO external encoder for one encoder only
- the TO outputCam/camTrack for an output to be switched in a defined way
- the TO measuringInput for a measuring input

Moreover, technology objects for the preparation of technological data on the system level are available, for example:

- the TO FollowingObject for synchronous operation between two axes or one axis on an encoder value
- the TO path for traversing path axes along a path and a position axis synchronous to the path
- the TO cam for representing complex programmable functions
- the TO additionObject, TO formulaObject for the processing of motion data and technology data on the system side

The technology objects are implemented by the system in system tasks, e.g. IPO task, IPO_2 task or servo task.

Instantiation

Technology objects are provided by the system as **technology object types** adapted to the specific application using instantiation.

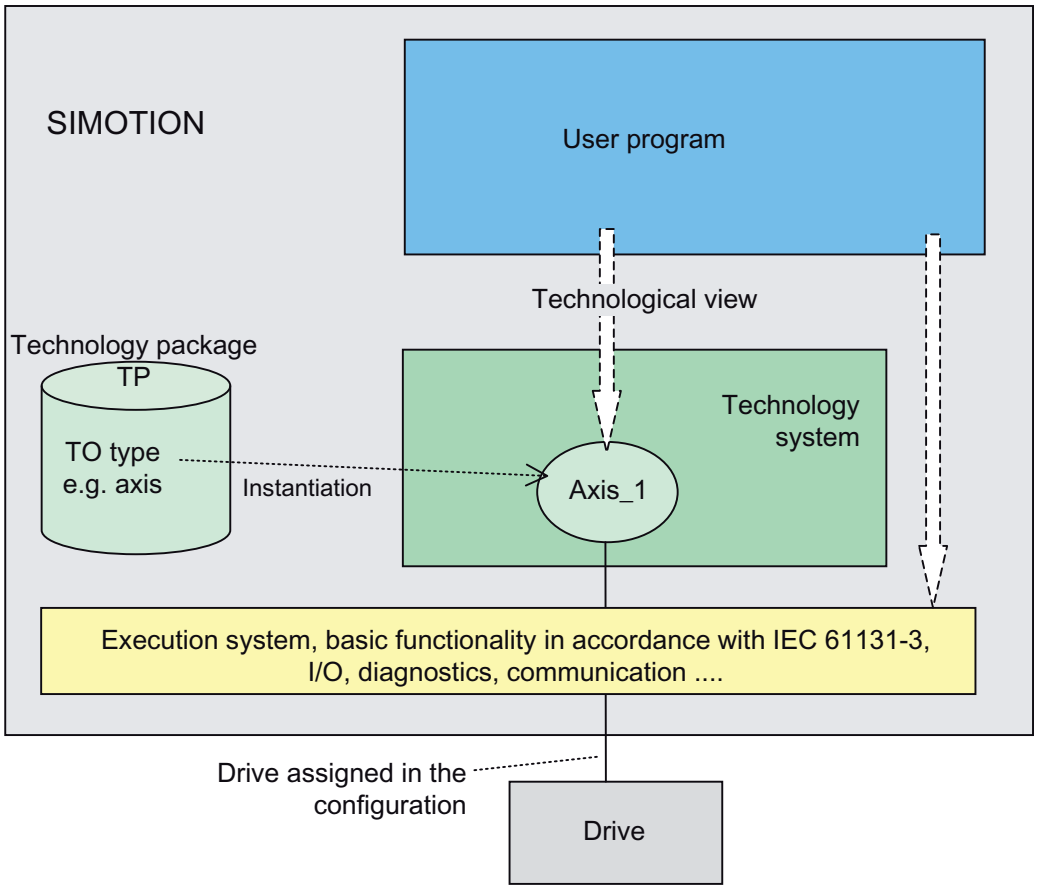


Figure 4-42 Programming model on technology object instances, e.g. on TO axis

Technology object types are combined in one technology package and loaded to the runtime system.

4.2.3.2 Technology packages

In SIMOTION, the technology object types are provided via **technology packages**. They are loaded to the SIMOTION runtime system together with the project.

The following technology packages are available:

- **TP CAM** contains the basic technologies for motion control, such as drive axis, position axis, following axis, synchronous object, cam, output cam, cam track, and measuring input.
- **TP Path** contains TP CAM and also the Path technology.
- **TP CAM_ext** contains TP Path and also objects for the preparation of technological data at system level, e.g. addition object, formula object
- **DCC** contains interconnectable blocks for drive-based controller functions.
- **TControl** contains temperature controller technology.

Shell diagram for technology packages

The range of functions offered by the technology packages can be graphically represented in the form of a shell diagram.

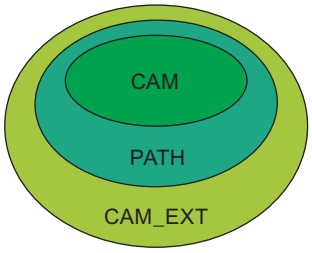


Figure 4-43 Shell diagram for technology packages

Other technology packages

Other sector-specific technology packages are also available as separate products.

4.2.3.3 Technology objects (TO)

Technology objects / technology object types provide the functionality for technology and motion control, thus comprising technological system functions and hiding the concrete hardware connection.

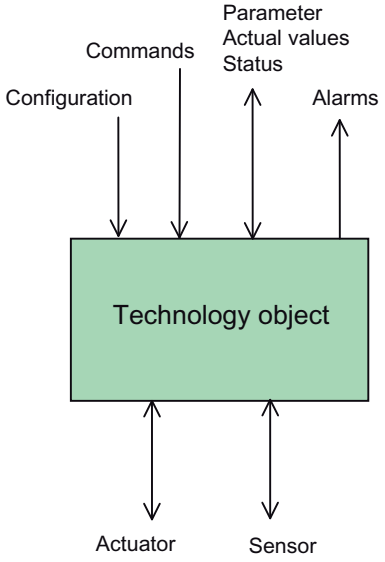


Figure 4-44 Possible interfaces to technology objects

For the cross-TO processing of technological data on the system level, the technology objects provide defined input and output interfaces.

See also

Available technology objects (Page 1148)

Interconnections (Page 1144)

Instantiation and configuration

With **Add <technology object>**, you create an instance of the corresponding TO type in **SIMOTION SCOUT**. (Data, parameters, alarm lists, etc. are created in the system.)

If necessary, the technology object type must be specified, e.g. for an axis: drive axis, position axis, following axis.

The hardware/software interfaces to the actuator/sensor are defined, e.g. hardware addresses, data width, message frame type for PROFIdrive message frames.

Basic settings are defined, e.g. the processing of the technological system functions in the IPO or IPO_2 cycle clock, see Task runtimes (Page 1358).

SIMOTION SCOUT provides wizards and dialogs for the **configuration** of the technology objects. The basic functionality of a technology object is defined by the configuration data.

All entries in the configuration and parameterization screen forms are output in configuration data and system variables, which are displayed in the screen forms as tooltips.

For a detailed description of the configuration data and system variables, please refer to the SIMOTION Reference Lists.

For a description of the procedure of creation and configuration of the individual technology objects, please refer to the respective **function manuals**.

A majority of the configuration data can be modified during the runtime via the user program or the expert list.

See also

Programming of general standard functions - overview (Page 1469)

System variables (Page 1245)

Configuration data (Page 1249)

Parameter assignment

System variables comprise technological parameters and display values of technology objects.

SIMOTION SCOUT usually provides screen forms for the setting of technological parameters and standard values/defaults.

System variables are individual values or structures that are read out consistently.

Additional information on the system variables can be found in System variables (Page 1245).

Programming

The technological functions are activated/deactivated using specific commands.

Synchronous/asynchronous command execution

The commands can be set *synchronously* or *asynchronously*.

With the synchronous setting of motion commands, it is possible to wait for a certain motion status or for the end of the motion, e.g. of a positioning, on the command. This setting is especially advantageous when programming motion sequences in sequential tasks (motion tasks).

Return value

The return value supplies the result of the function call, e.g. function was/is carried out as planned or there is an error.

CommandId

CommandId can be used to uniquely identify and trace a TO command.

For information on parameters, step enabling conditions, diagnostics, etc., please refer to the following:

- Functions for CommandID (Page 1541) and various examples in this manual.
- **SIMOTION MCC Motion Control Chart**, Programming MCC Charts

For a detailed description of the TO commands, please refer to the SIMOTION Reference Lists.

Programming model

The commands are assigned to tasks which are in turn assigned to the execution levels of the task system.

Commands can be issued from all user program tasks of the system.

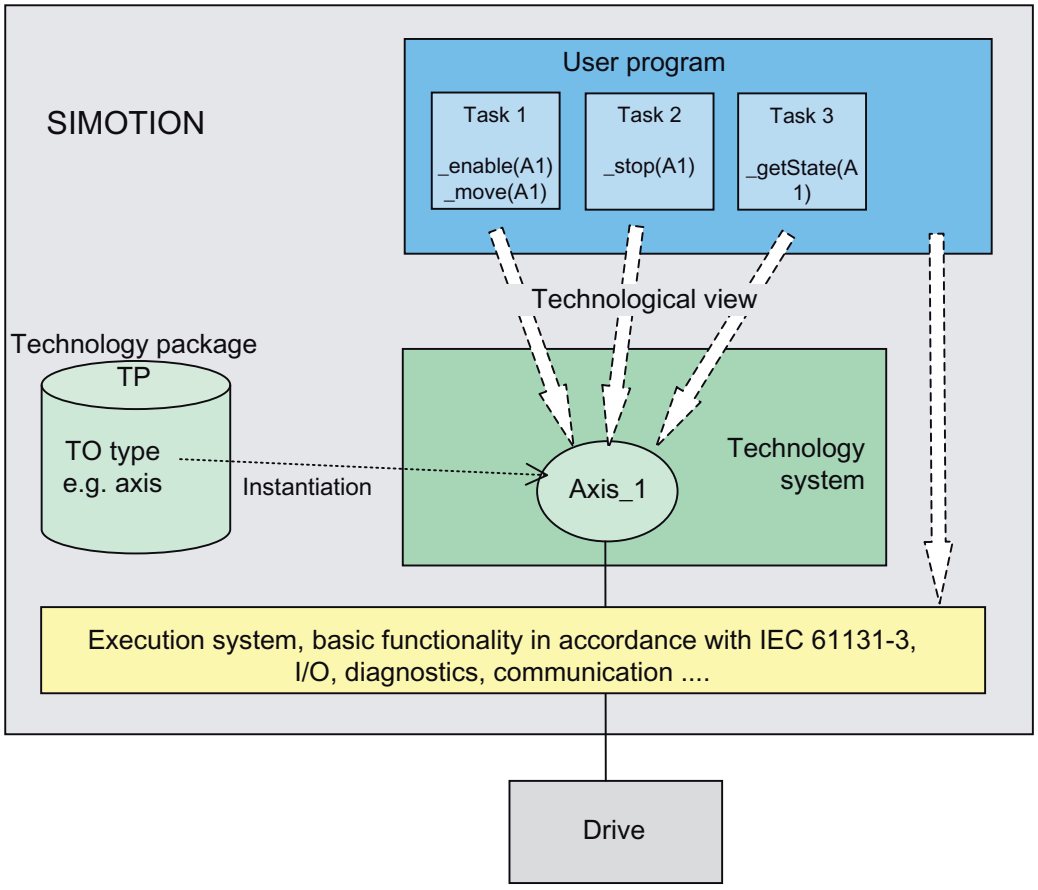


Figure 4-45 Programming model on technology object instances, e.g. on TO axis

The execution time of a command on the technology object is the only factor that determines the effectiveness of the command.

If commands are issued from multiple tasks, programming consistency must be ensured by the user program.

Command execution on the TO / effectiveness of TO commands

The commands issued from the user program, the user program tasks, to the TO can generally be subdivided as follows:

- Commands which are executed immediately in the context/sequence of the user program tasks
  These are handled like a function within the user program tasks.
  These commands are synchronous since the user program is continued only upon the return of the function result.
  Example: Reading TO values
- Commands which are entered in a command buffer and which overwrite each other
- Commands which are entered in a command buffer and which are rejected if the command buffer is occupied

In the TO commands you specify the response of the commands on the TO if more than one command is issued to a command group between two processing cycle clocks, e.g. replace/ overwrite or, command rejection.

Information on the command buffer and command activation can be found in the **function manuals** of the individual technology objects.

Execution properties

The execution properties of technology objects are object-specific.

The following synchronous execution levels are provided for the execution of the technology objects: DP cycle clock, servo cycle clock and IPO or IPO_2 cycle clock (except for temperature controllers).

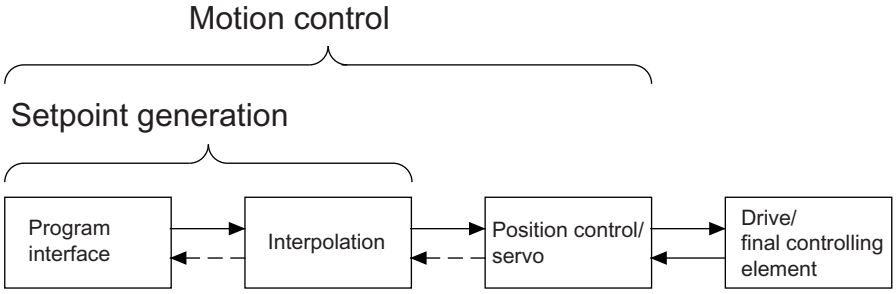


Figure 4-46 Flow chart

The instances of the technology objects are processed in different execution levels.

- The command evaluation and motion control are processed in the IPO/IPO_2 cycle clock.
- The position and setpoint control are processed in the servo cycle clock.
- Communication with the drive is processed via PROFIBUS DP in the DP cycle clock or PROFINET IO with IRT in the PN cycle clock.

The processing can be influenced by means of the system and object configurations.

- System cycle clock setting
  Setting the system cycle clocks also sets the sampling time for the motion control of axes (IPO, IPO_2), the position control (servo), and the communication via PROFIBUS DP or PROFINET IO with IRT.
- When configuring objects, you can specify whether motion control is to be executed in the IPO or IPO_2 cycle clock. This allows you to distinguish between operations that are time critical and those that are not.
- Technology objects are controlled from the user program through system function calls.

With V4.2 and higher, Servo_fast and IPO_fast can also be used as an option in a second application cycle clock.

Alarms

A technology object monitors the execution and executability of technological functions as well as the I/O required for the TO, and creates a **technological alarm**, if necessary.

A **technological alarm** has a TO-local alarm response, e.g. stop motion, and a global response, e.g. stop system or call alarm task in which further responses can be specified. The local and global responses can be set for specific alarms.

For a detailed description of the TO alarms, please refer to the SIMOTION Reference Lists.

See also

Execution system (Page 1298)

Differences between cyclical and sequential programming (Page 1227)

Process Alarms (Page 1281)

Interconnections

The technology objects can be flexibly interconnected (for example, following axes can be switched over or interlinked) and even distributed among several SIMOTION devices. The functionality required for this is in the object and does not need to be programmed. This enables maximum flexibility in terms of functionality and distributability.

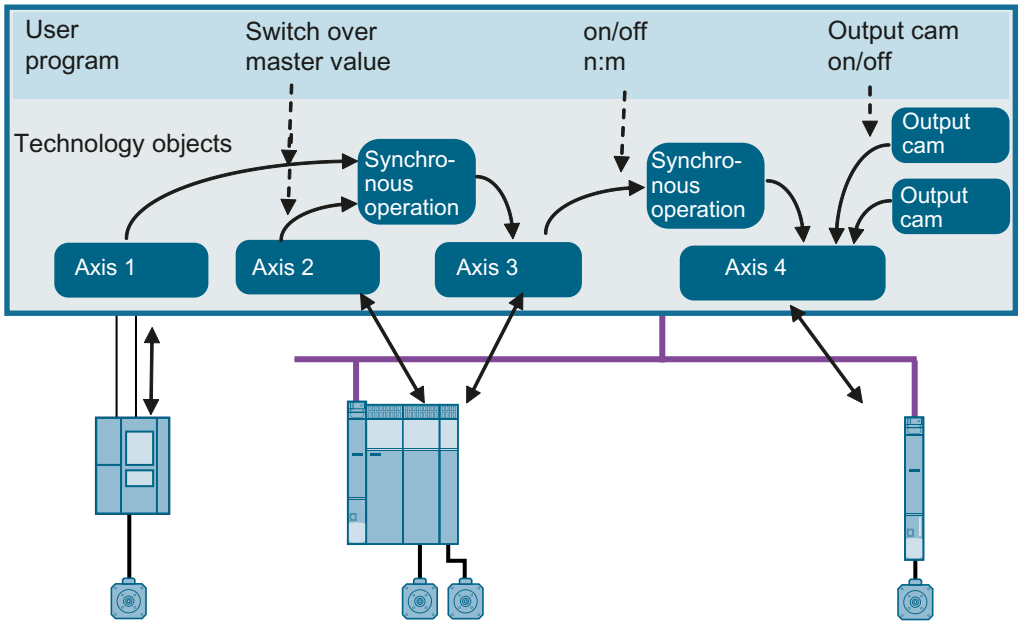


Figure 4-47 Interconnection example

There are interconnection interfaces defined on the technology objects for the exchange of data/information between the technology objects on the system level. These interfaces generally allow for the bidirectional exchange of data between individual technology objects.

The type of the external interfaces can be different on each TO type. There are no interconnections or actuators/sensors required.

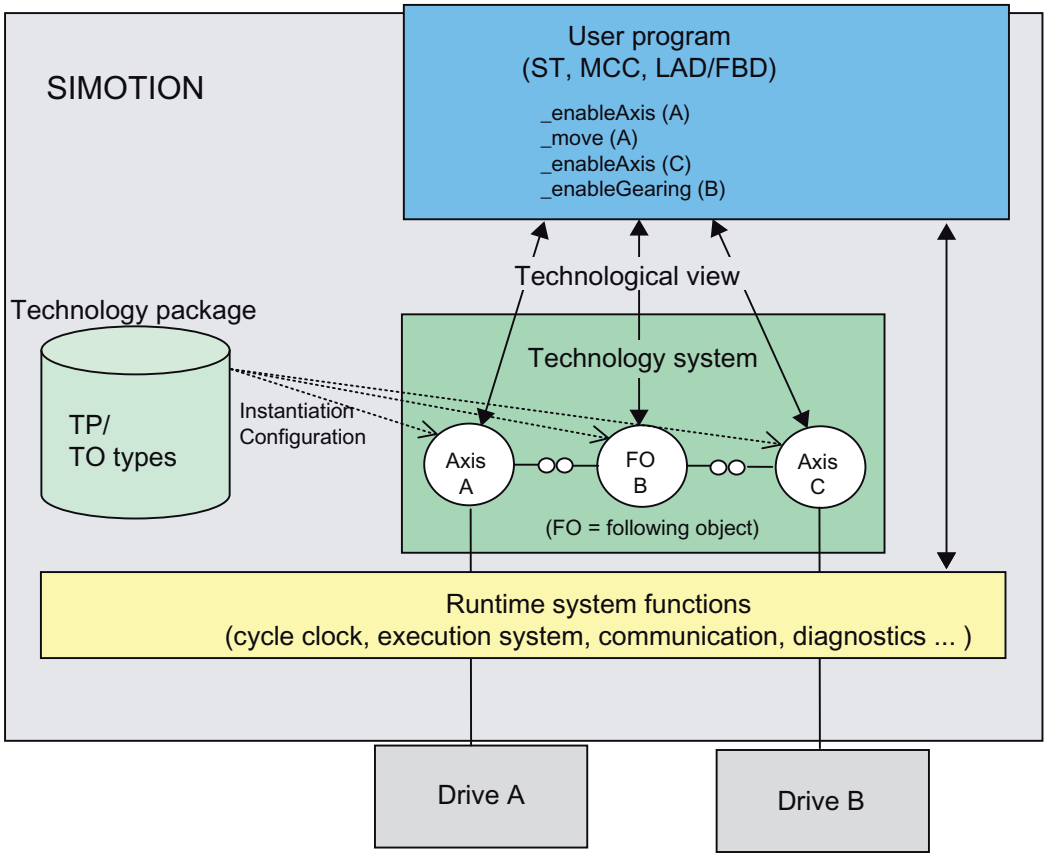


Figure 4-48 Interconnection between technology objects, example: Master axis - Following object - Slave axis

Depending on the interface type, interconnection interfaces can exchange different data in cyclic mode (cyclic motion data, e.g. s,v,a for the motion vector of type motion) and during initialization (e.g. modulo information, units).

Implicit interconnection

If an interconnection is mandatory and unambiguous, it is implicitly created by the SIMOTION SCOUT engineering system, e.g. measuring input with axis, cam with axis, synchronous operation with axis. The corresponding TO types are offered under the axis.

Interconnection using technological interconnection screen forms

There are specific screen forms provided for further interconnections, e.g. for:

- Following axis with master axis/master value
- Following object with cams
- Profile inputs with cams

Interconnection using general interconnection screen form (for experts only)

There is a general interconnection screen form provided for special interconnections, e.g. for:

- MotionIn interface interconnection (for motion vectors)
- Torque input interface interconnection

See also

Interconnection interface type 'Motion' (for experts) (Page 1167)

Interconnection interface type 'LREAL' (for experts) (Page 1168)

Interconnection of technology objects (Page 1159)

Technology objects and DCC

Description

The data exchange between DCC and TO is performed by DCC charts

- using the direct interconnection of block inputs and block outputs with system variables of the TO;
  for TO axes, it is possible to transfer cyclically the data specified in system variables, such as override and setpoints, without commands needing to be issued each time in the user program.
- or by forwarding values calculated in DCC via commands and variables in the user programs to the TO. The TOs have specific function-related interconnection interfaces.

The TOs themselves are not available as blocks in the DCC charts.

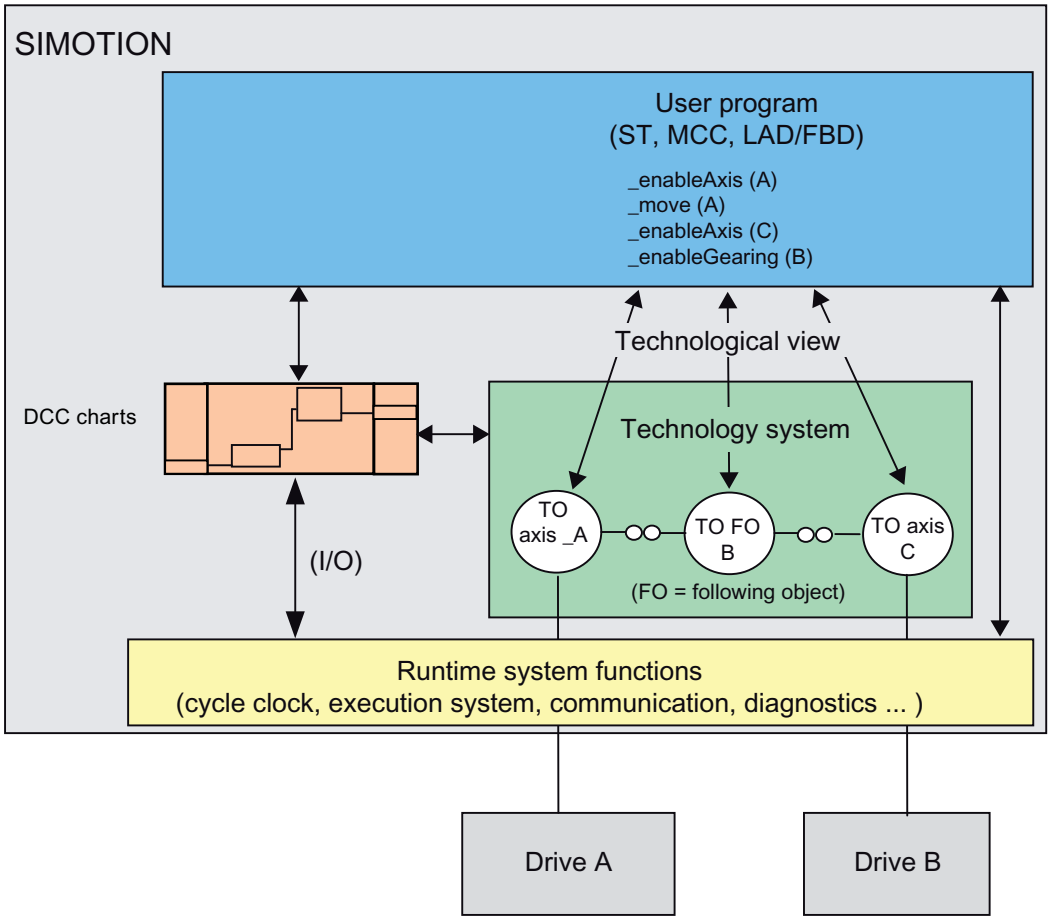


Figure 4-49 Technology and DCC

See also

Sequence model for DCC blocks (DCB) (Page 1394)

Available technology objects

Table 4-12 Overview of the technology objects available in SIMOTION

Technology object	Symbol	Brief description
Axis		Axes are available in various versions. There are: • Real or virtual axes • Position axes or drive axes • Electrical axes or hydraulic axes • Standard axes or force/pressure controlled • Modulo axes • Following axes • Path axes For detailed information, refer to the Motion Control TO Axis Electric/Hydraulic, External Encoder Function Manual.
Following object		If an axis is created with the synchronous operation technology, the following object will then be created below the axis. The settings for the synchronous operation are stored in the following object. For detailed information, refer to the Motion Control TO Synchronous Operation, Cam Function Manual.
Path object (as as of V4.1)		You can use the Path object technology object to configure the path interpolation. For detailed information, refer to the Motion Control Path Interpolation Function Manual.
Measuring input		Measuring inputs are used for fast, accurate measurement of actual positions. For detailed information, refer to the Motion Control TO for Output Cam and Measuring Input Function Manual.
Output cam		The TO outputCam creates position-dependent switching signals and can be assigned to position axes, following axes or external encoders. For detailed information, refer to the Motion Control TO for Output Cam and Measuring Input Function Manual.
Cam track		Cam tracks are used to group several output cams to form a technology object. For detailed information, refer to the Motion Control TO for Output Cam and Measuring Input Function Manual.
External encoder		The technology object external encoder provides the functionality in the system for connecting an external encoder without an axis, e.g. a shaft encoder on a press. For detailed information, refer to the Motion Control TO Axis Electric/Hydraulic, External Encoder Function Manual.
Cam		The TO cam can be used to define a transmission function and apply it with other technology objects. For detailed information, refer to the Motion Control TO Synchronous Operation, Cam Function Manual.
Temperature channel		The TO temperatureChannel can be used to configure temperature controls in SIMOTION. For detailed information, refer to the Motion Control Additional Technology Objects Function Manual.

Technology object	Symbol	Brief description
Additional technology objects (for experts): The advantage of these technology objects is that they are executed on the system level like other TOs. The interconnected signals are processed directly in the technology objects within a system cycle clock. Applicative solutions in structured text, in contrast, bring about frequent changes between system and application and thus dead times. The additional technology objects are used for the implementation of winder applications, for example.		
Fixed gear		You can use the fixed gear technology object to implement a fixed synchronous operation (without synchronization/desynchronization) using a specified gear ratio. For detailed information, refer to the Motion Control Additional Technology Objects Function Manual.
Addition object		You can use addition objects to add up to four input vectors to produce an output vector. For detailed information, refer to the Motion Control Additional Technology Objects Function Manual.
Formula object		With formula objects, you can use mathematical operations on scalar (LREAL, DINT) and on motion vectors of the Motion type. For detailed information, refer to the Additional Technology Objects Function Manual.
Sensor		The TO sensor can be used to record scalar measured values. For detailed information, refer to the Motion Control Additional Technology Objects Function Manual.
Controller object		The controller object can be used to prepare and control scalar variables. For detailed information, refer to the Motion Control Additional Technology Objects Function Manual.

4.2.3.4 Expert list

In addition to accessing the configuration data and system variables via wizards and parameterization screen forms, you also have the option of direct access via the **expert list**.

Since all important configuration data and system variables can be parameterized via dialogs, the expert list is, however, only required in exceptional cases (such as online modification of configuration data).

These lists state the most important configuration data and system variables for the programming and diagnostics of axes, following object and external encoder.

Starting in V4.1, the expert list has a separate view containing a selection of the most important configuration data and system variables (**Selected Parameters**). This gives you easy access to the most important configuration data and system variables for programming and diagnostics for each individual technology object.

As of V4.2, it is no longer possible to create user-defined lists of values/parameters. The only way to convert existing user-defined lists of values/parameters without encountering any problems is to open them in a watch table with exactly the same range of functions.

Note

Special knowledge of the system is required to modify parameters in the expert list.

- You can overwrite entries made using the expert list by calling wizards and parameterization screen forms.
- No check is made on dependencies with other parameters.

Invoking the expert list

1. Highlight the required technology object in the project navigator.
2. Select **Expert list** in the object context menu or press **Ctrl+E** on the keyboard (if, for example, an axis dialog is open).

Or

- Double-click on the **Expert list** entry in the project navigator below a TO.

Expert list for configuration data and system variables

The **system variables** and **configuration data** for the technology object are displayed in the first two tabs in the **expert list** .

You can change the configuration data values in OFFLINE mode and in ONLINE mode. Parameters which can be written are displayed in green, those which cannot be written are displayed in yellow. Grayed-out values cannot be changed.

As of V4.2, it is no longer possible to create user-defined lists of values/parameters. The only way to convert existing user-defined lists of values/parameters without encountering any problems is to open them in watch tables with exactly the same range of functions.

Table 4-13 The following information is displayed in the expert list:

Button / Table column	Meaning/Note
Configuration data	The **Configuration data** tab lists the configuration data of the TO in alphabetical order. All configuration data can generally be modified by the user (displayed in green).
System variables	The **System Variables** tab lists the system variables of the TO in alphabetical order. There are system variables which can be written (displayed in green) and which cannot be written (displayed in yellow).
Selected parameters	On the **Selected Parameters** tab, the most important system variables and configuration data of a technology object are displayed in alphabetical order, separated according to system variables and configuration data. The tab can be closed, but it will be displayed again at the next call. The selection is a factory setting and cannot be changed.

Button / Table column		Meaning/Note
Convert list into watch table 		Click this button to convert a user-defined list created using an earlier version to a watch table. A message containing more detailed information appears before the conversion takes place. When you have confirmed the message, you can select the list in a file dialog. Once selected, the list is converted and opens as a watch table.
Quick search 		In this text field, you can carry out a quick search within the expert list. The columns Parameter, Parameter text and Value are searched.
Compare configuration data or system variables 		Click this button if you want to compare configuration data and system variables of 5 technology objects max. and the differences in ONLINE and OFFLINE mode of an object. For further information, see Comparing expert lists (Page 1156).
Collect changes 		While the button is activated, all the configuration data changed by you with an effectiveness of "immediately" and "restart" is collected but the new values do not take effect in the target system. The collected changes to the configuration data only take effect in the target system when you click **Activate changes**. See below.
Activate changes 		• If all the collected changes to the configuration data **only** have an effectiveness of **immediately**, the changes take effect as soon as you click **Activate changes**. • If some of the collected changes to the configuration data have an effectiveness of **restart**, **all** changes only take effect after a technology object **restart**.
Activate TO restart 		For configuration data that only take effect after a restart of the TO, the restart can be performed by either activating the system variable "restartActivation" or by clicking the **Restart** button. The button restarts the axis. The button becomes active when you have changed an effective data item after a restart, but not yet accepted it.
Configuration selection list		Select whether the parameters for a linear axis or rotary axis are to be displayed or changed for standard, force or pressure in the table in the expert list.
Parameter 		The name of the configuration data item or the system variable is displayed here. Click the plus sign to open the entire structure.
Parameter text		A short description of the system variable or the configuration data is displayed.
Offline value		The value of the configuration data item or the system variable is displayed here. Depending on the type of parameter, you can enter the value directly as a numerical value or select a symbolic identifier from the selection list.
Configured value	(ONLINE only, configuration data)	The configured value of the configuration data item is displayed here. This value is saved to the RAM of the target device.

Button / Table column		Meaning/Note
Current value	(ONLINE only)	The current value of the system variable or the configuration data is displayed here. You can change the value of the system variable directly in ONLINE mode. Changes to the configuration data can only be made in the **Next value** column. Current values are stored in the **current data memory**. To transfer these values to the RAM, **Target system > Copy current data to RAM** must first be selected in the menu. **Note:** System variables changed ONLINE cannot be copied to RAM. This also means that it is not possible to **Save to memory card (Ram2Rom)** or **"Save in the engineering project (load configuration data in PG)**. See SIMOTION memory concept.
Next value	(ONLINE only, configuration data)	You can enter the **Next value** of the configuration data item in ONLINE mode here. This value is then taken over as Current value depending on when the configuration data item is to take effect. With Immediately, the new value takes immediate effect in the target system after pressing the ENTER key or selecting a value.
Units		The unit of the system variable or the configuration data is displayed.
Effectiveness		This displays when the value of the changed configuration data item takes effect, e.g. with **restart**, the changed value takes effect in the target system after a restart. You cannot change certain configuration data in ONLINE mode. With V4.1.2 and higher, this column is also displayed OFFLINE.
Data type		The **data type** of the system variable or the configuration data item is displayed here (e.g. INT for an integer value).
Minimum		The minimum value of the system variable or the configuration data item is displayed.
Maximum		The maximum value of the system variable or the configuration data item is displayed.
Filter		This line allows you to enter filtering criteria for the contents of the various columns in the table. Alternatively, you can select predefined filters in the selection box. To reset the filters, select the criterion **All** in the selection box or click the  symbol.

See also

Comparing expert lists (Page 1156)

Using the expert list

Displaying the help for the system variables and the configuration data

1. Select **Help > Context help**.
   Or
   Press the **SHIFT+F1** keys.
   Or
   Select **Parameter help** in the context menu.
   The cursor changes to a question mark.
2. Click the configuration data item or the system variable in the expert list for which help is required.
   The help for this is displayed.

Changing the data in the expert list offline

1. In the expert list, select the Configuration data or System variables tab.
   A tabular overview is displayed.
2. In the **Parameter** column of the expert list, click the plus sign in front of an entry to display subentries.
   You cannot change entries highlighted in yellow.
3. Click the **Offline value** column of the entry to be changed.
4. Enter the value directly or select a value when a selection list appears.

Carrying out a quick search

When entering a search term, the cursor jumps to the first line in which the search term is found.

- Enter the search term and press the ENTER key.

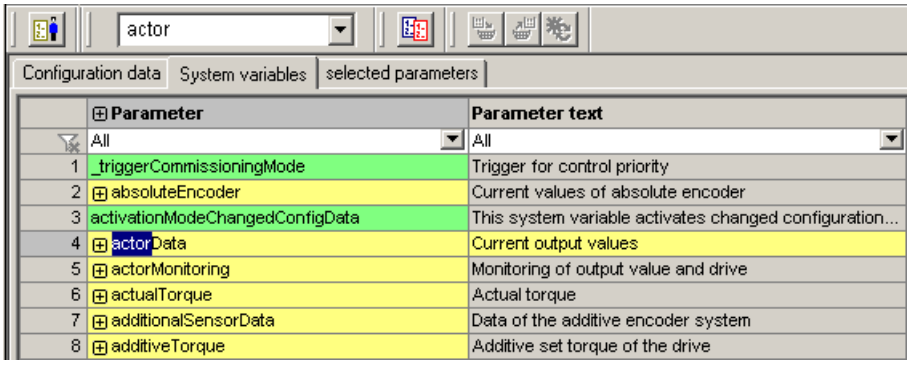


Figure 4-50 Example of quick search

If it is a writable parameter, the cursor jumps directly to the value field.

If a structure shell is found that does not have a separate parameter value, the jump is made directly to the cell in which the term was found.

Continue search:

- Press the **F3** key.
  The next cell containing this term is searched.

If the search function does not find any further entry in the list, a message is output.

Navigating within the expert list using the arrow keys

Each cell within the expert list can be selected using the arrow keys.

1. Press an arrow key.
   The cell located next is selected.
2. With the **Ctrl+arrow** key combination, the cursor jumps to the end or beginning of the respective line or column.

Selecting cells using the keyboard

If you want to select the complete expert list, including the cells in the non-visible area:

- Press the **Ctrl+A** key combination.

In online mode, this key combination also initiates a read process for all of the parameter values (with technology objects as of V4.0).

Individual cell areas can be selected by navigating with the arrow keys or with the mouse:

- The cells are selected in the course of navigating by simultaneously pressing the **Shift** key.

Cell areas can also be selected using the mouse:

- Place the cursor on a line, press the left mouse button and drag the mouse pointer across the cell area to be selected.

Copying cell contents into other Windows applications

Selected rows or cells can be copied into other Windows applications using the Windows clipboard:

- Press the **Ctrl+C** key combination or select **Copy** in the context menu.

From the clipboard, the cell content can be inserted, e.g. in an Excel or Word document.

How to create a watch table

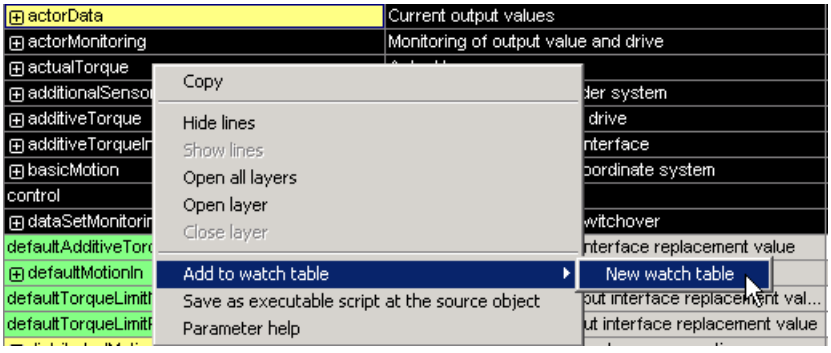


Figure 4-51 Context menu in the expert list (right-click)

Table 4-14 Context menu in the expert list

Menu command	Meaning/function
Copy	Copies the parameters selected
Hide lines	Hides the marked lines.
Show lines	Shows the hidden lines again.

Menu command		Meaning/function
Open all levels		Opens all levels
Open level		Opens all the levels selected
Close level		Closes any levels which are open
Add to watch table		
	New watch table	Creates a new watch table and opens it in the **<Watch table name>** tab of the detail view
	Watch table name	Adds parameters to an existing watch table
Add to watch table, values as control values		
	New watch table	Creates a new watch table and opens it in the **<Watch table name>** tab of the detail view
	Watch table name	Adds parameters to an existing watch table
Save as executable script at the source object		Saves as executable script
Parameter help		Calls the help for the parameter selected

Procedure for creating a watch table

1. Select the lines in the expert list to be transferred to the watch table.
2. In the context menu (right-click), select **Add to watch table > New watch table**. The **Insert Watch Table** dialog opens.

   Note

   To add to an existing watch table, select **Add to watch table > <Watch table name>**. No more dialogs are called. The values are added to the existing watch table.

3. In the **Insert Watch Table** dialog, you can enter the name, author, version, and comment. The name is preassigned automatically.

Note

You can also save the results of an expert list comparison as a watch table or as an executable script, see Comparing expert lists (Page 1156).

Notes on the structure of scripts:

- Each converted script receives a header with date and origin, and then an empty line.
- The logging mechanism of the script engine is activated in the next line. This causes each write and read action to be output in the script output window.
- Headings in the watch table are entered in the script as comments with a leading additional empty line.
- Comments in the watch table are entered in the script as simple comments without an additional empty line.

- For each write parameter in the watch table, a write request is entered in the script. Numbers and enumerations are written as integer values. The parameter text of the parameter is added as comment at the right-hand side next to the write request.
- Read-only parameters in the watch table are no longer saved in scripts as of V4.2. When parameters are saved as a watch table, only the parameter (without a value) is added to the watch table.

Example for the use of scripts:

- Any changed unit systems will not be used when the script is executed. You must yourself ensure that the acceptance of the parameter values is sensible in the current context.
- Parameters that change the structure (e.g. TypeOfAxis.typeOfAxis) are not treated differently. You must yourself ensure that these parameters are at the top of the parameter selection.
- Only the current value is written; this means the script may not be executed online if it contains configuration data.

Application: You can also transfer the parameter assignment from one axis to the other with scripts.

Comparing expert lists

Description

In this window, you can compare the differences in parameter settings of a maximum of five objects (TOs) and the differences in ONLINE and OFFLINE mode of an object.

The displayed data is always the current data in effect. For TOs in ONLINE mode, this data is the actual values. System variables are not permanently updated, only if you update the representation with **Start comparison**.

You can change the values of the comparison objects as well as the values of the reference objects. Objects that cannot be changed are indicated with "???".

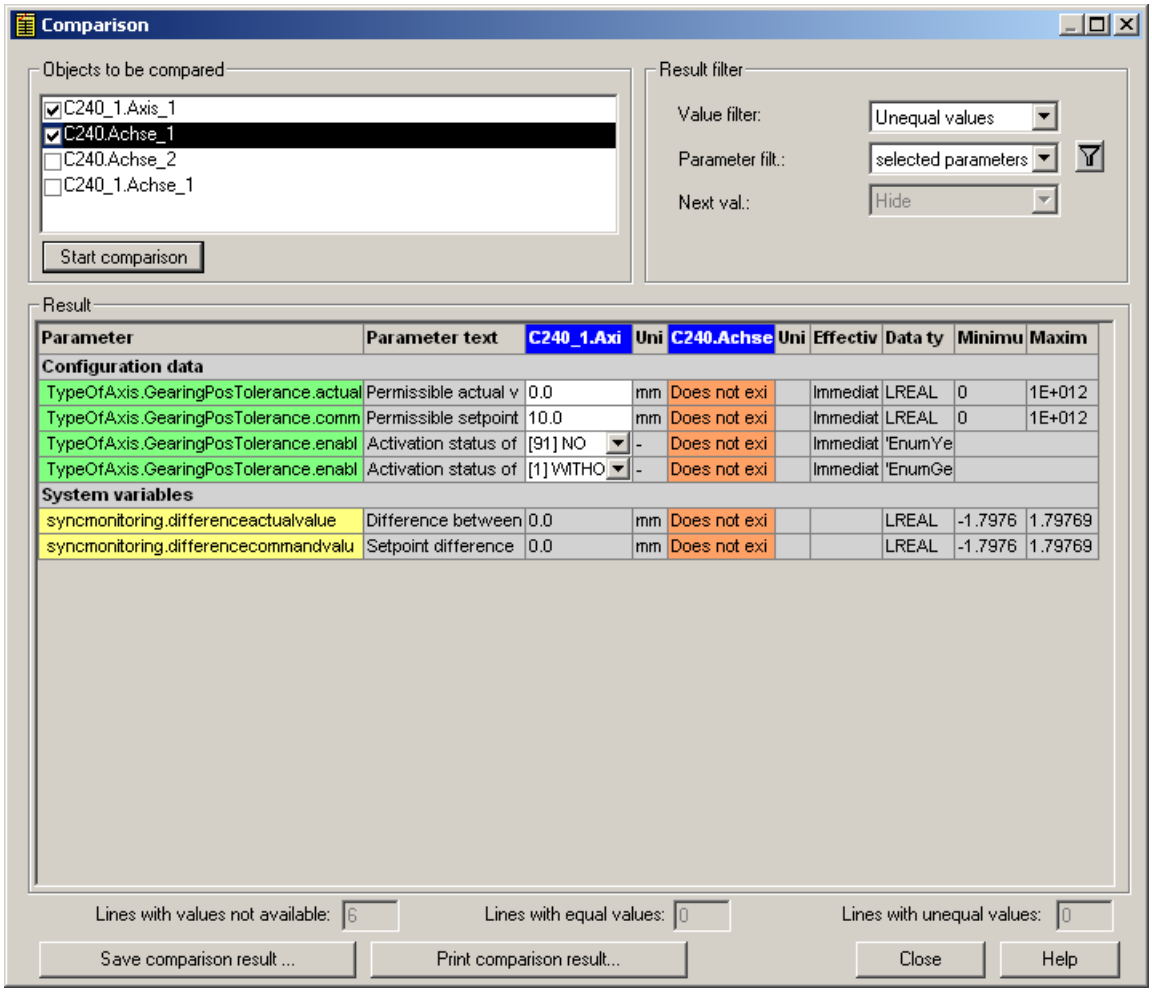


Figure 4-52 Experts list comparison

Parameters with different settings or parameters not available for an object are presented in a list.

Note

A comparison can only be made within groups of compatible objects (TOs). Only the eligible objects for the comparison will be made available.

Button / table column	Meaning/information
Objects to be compared	Select the checkbox for the objects you want to compare. You can compare up to five objects with one another and objects in OFFLINE and ONLINE mode. In ONLINE mode, separate checkboxes are available for ONLINE and OFFLINE parameterization of an object. The uppermost list, which is grayed out, is used as a reference list. The display is always dictated by the structure and, thus, the parameter set of the reference object. As a result, for comparison objects it can only be indicated whether or not they have the parameter of the reference object. Parameters that only the comparison objects have but not the reference object are not displayed. If the values of a setting are different, this is indicated by the appearance of a light-orange colored background in the relevant cell in the columns of the relevant comparison objects. However, the relevant cell of the reference object does not display a colored background.
Start comparison	Click **Start comparison** to compare the parameter settings of the selected objects. The compared settings are listed under **Result**.
Value filter	Select the parameter types that are to be displayed under **Result** after the comparison.
All	All the parameters that can be compared are displayed under **Result**.
Unequal	Only the parameters whose values are unequal will be displayed under **Result**.
Equal	Only the parameters whose values are equal will be displayed under **Result**.
Parameter filter	Only the type of parameters are displayed which you select in the drop-down list.
Display filter	Click the button to open the **Display Filter** window. Here you can edit the display of the columns.
Next value	The next values (values in the next memory) are also displayed.
Result	**Result** displays the parameters found to have different settings during comparison of the objects and other parameters that are not available for an object. Diverging values and non-comparable values of parameters are displayed with a colored background.
Save comparison result	Click the button to save the comparison results as a new watch table or as an executable script. The button is active only if the parameters of a device are displayed under **Result**. For further information, see Save expert list comparison (Page 1159).
Print comparison result	Click the button to print the comparison results in a tabular form. The parameter number, the value for the respective object and the unit will be printed in each case.

See also

Save expert list comparison (Page 1159)

Save expert list comparison

Description

In this window, you can save the result of the comparison.

The button is active only if the parameters of a device are displayed under Result.

- Click **As watch table** to create a new object-granular watch table. The left column is always the reference column, i.e. only the parameters of the reference object are compared and saved.
- Click **Executable script on source object** and select the object to which you want to assign the script. A SCRIPTS folder will then be created under this object.

See also Comparing expert lists (Page 1156).

You can save only the comparison results of one object. Prior to storing, you must specify which object is to be saved.

4.2.3.5 Interconnection of technology objects

Technology objects are automatically interconnected when they are created (synchronous axis, output cam, measuring input). Additional interconnections can be established using a general interconnection screen form.

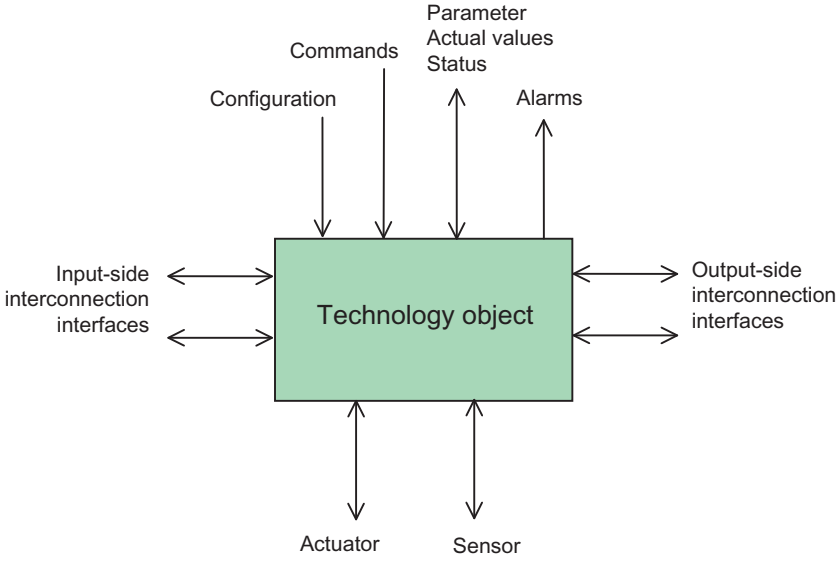


Figure 4-53 Interconnection interfaces to technology objects

Interconnection overview for technology objects

Introduction

With the interconnection overview you can display all motion input and output interconnections of technology objects in the project via an interconnection tree. The tree display enables the interconnections to be displayed in cascades.

Note

As of V4.3, it is possible to assign device names that do not comply with the ST naming convention. Names with special characters such as "." are displayed in the interconnection overview in " ".

An interconnection table shows the technology objects interconnected on the input or output side with interface designation for the technology object selected in the interconnection tree.

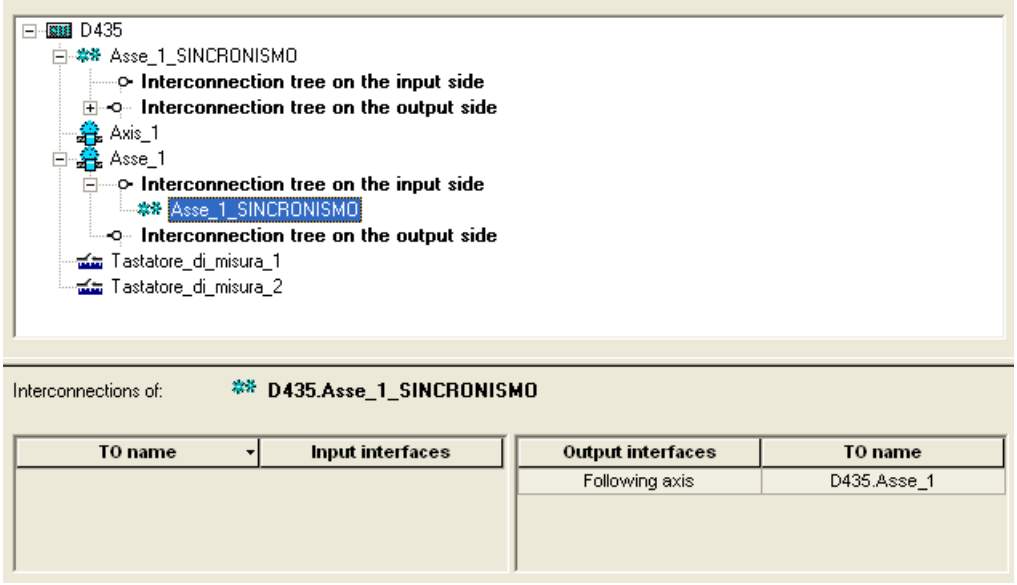


Figure 4-54 Interconnection overview

Properties of the interconnection overview

The interconnection overview can be started with the button or via **Edit > Interconnection overview**.

- After the interconnection overview has been started, the device level (CPUs) is displayed first. You can then display all instantiated TOs under each CPU by opening the appropriate interconnection tree of the CPU. The TOs are sorted according to TO type and displayed in alphabetical order within a TO type.
- Two interconnection trees are offered for the TOs that are displayed directly below the CPUs:
  - Input-side interconnection tree; indicates the motion input interconnection
  - Output-side interconnection tree; indicates the motion output interconnection

- All technology objects that have motion inputs and outputs interconnected to other technology objects, e.g axes, synchronous objects or encoders, can be opened further. This is possible at every hierarchy level of the interconnection tree.
- Recursion is shown if the interconnection tree contains a node which already exists on the path to the root of the tree. The lower part of the tree is not shown for the sake of simplicity. Continue to the node which is closer to the root.

The interconnection trees can only be opened up sequentially, not all at once with one click.

Displaying interconnection table

Select a TO in the interconnection table.

The associated interconnection table is displayed below the interconnection overview. The input and output interconnections of all interconnected TOs are listed.

Implicit interconnection when creating

Description

Interconnections are generated automatically when the following TOs are created:

- Synchronous axis; synchronous object is interconnected to the synchronous axis (following axis)
- Output cam; this is interconnected to the master value
- Cam track; this is interconnected to the master value
- Measuring input; this is interconnected to the master value

These interconnections are also displayed in the interconnection overview (Page 1160).

Interconnection using general interconnection screen form in SCOUT (for experts)

SCOUT offers the **Interconnection** view for interconnection of various technology objects.

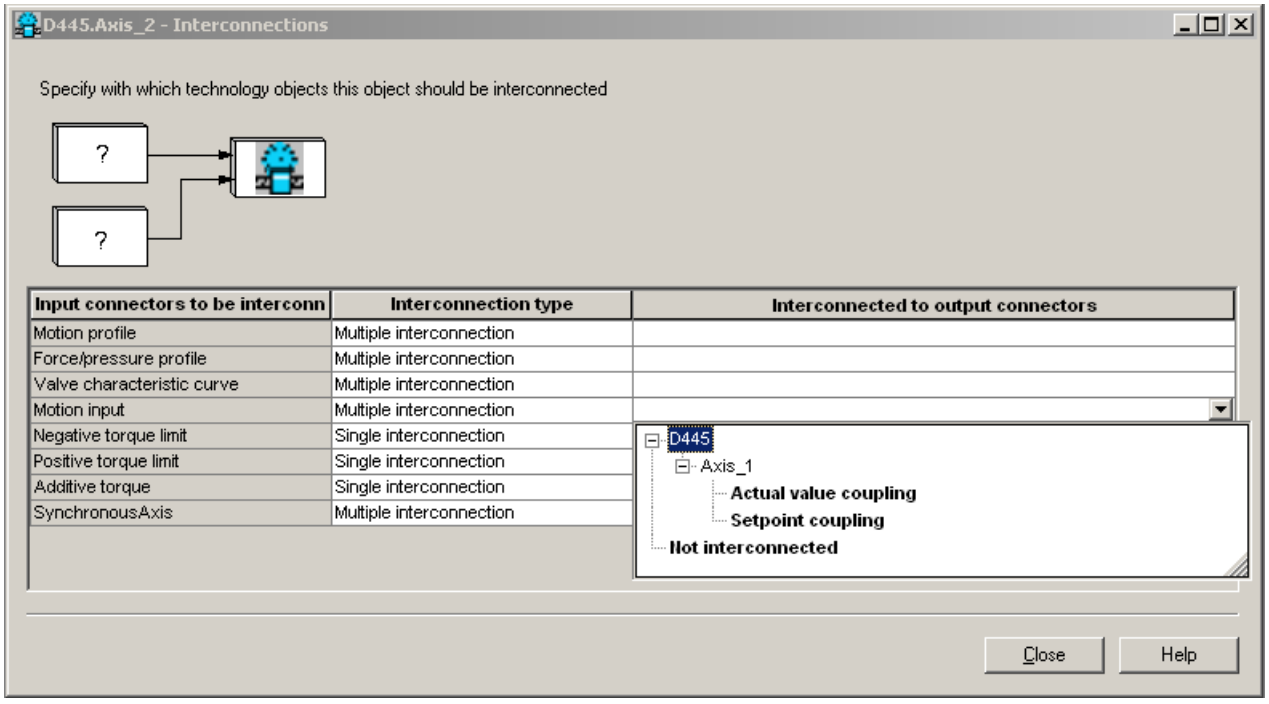


Figure 4-55 Interconnection in the SCOUT - axis example

On the left-hand side, all input-side interconnection interfaces of the selected object are listed, on the right-hand side, the output-side interconnection interfaces of the same type. Only the output-side interfaces displayed bold can be selected.

- When one of these lines is highlighted (clicked), you can open a tree that displays all interconnection possibilities.

Only those elements displayed **bold** can be selected.

The display of the devices and technology objects is used only for orientation purposes.

- Direct input can be made in the input line.

Only complete and correct inputs are accepted. The input field cannot be exited if incorrect inputs are made.

If the input-side interconnection interface has the characteristic of multiple interconnectability and can therefore be interconnected with several output-side interconnection interfaces, another line is added for the interconnection of the input-side interconnection interface after the interconnection with an output-side interconnection interface.

- An interconnection can be removed by deleting a line.

The value will be accepted when the input line is exited.

General properties of interconnection interfaces

Configuration of interconnection interfaces

The input-side interconnection interfaces are created:

- Implicitly with the instantiation of the TO
- Via technology-specific interconnection screen forms
- Via general interconnection screen forms

Enabling/disabling

The input-side interconnection interfaces are activated with a command on the associated TO or they are active by default.

In the case of multiple interconnectability of an input-side interconnection interface, enabling/disabling is carried out with a command for the technological function, e.g.:

- Selection of the cam in **_enableCamming()** via the **cam** parameter
- Selection of the technology object for the master value in the **_setMaster()** command via the **master** parameter
- Selection of the technology object for the reference value of the MotionIn interface in the **_run...basedMotionIn...()** command via the **mastertype.reference** parameter
- Selection of a profile or the cam in the **_run...Profile()** commands via the **profile** parameter

For information on this topic, please refer to the function manuals of the respective technology objects.

Indicator for status and interconnection values

The indicator for the status of the input-side interconnection and the interconnection values is defined on the TO for the individual interface types.

For information on this topic, please refer to the function manuals of the respective technology objects.

Validity

Interconnection interfaces have valid values when the interconnection is established and the interconnection values have the 'valid' status; invalid interconnections / interconnection values have the value 0.

If a (virtual) axis is set to simulation, the output-side setpoints will continue to be updated, the actual values will be frozen.

The status available in the system variables can be read out to determine whether the interconnection value or the substitute value is valid after ramp-up.

Alarm Response

In the case of a faulty interconnection, a technological alarm is only issued when a function on the interconnection interface is activated.

In the case of multiple interconnectability of an input-side interconnection interface, a reference TO must be selected.

Programmed functions will be aborted during the transition from **STOP U** to **STOP**; interconnection functions remain active during the transition from **STOP U** to **STOP**.

Interconnection interfaces on the TOs (for experts)

The input-side interconnection interfaces can only be interconnected with output-side interconnection interfaces of the same type.

TO	Interface	Type
Drive axis		
	Input-side interconnection interfaces:	
	Motion input	Motion
	Torque limit, positive	LREAL
	Torque limit, negative	LREAL
	Additive torque	LREAL
	Motion profile	MotionProfile
	Force/pressure profile	ForceProfile
	Valve characteristic (hydraulic axis)	ValveCharacteristics
	Output-side interconnection interfaces:	
	Motion setpoint	Motion
	Motion actual value with extrapolation	Motion
	Actual torque	LREAL
Position axis		
	Input-side interconnection interfaces:	
	As for drive axis	
	Output-side interconnection interfaces:	
	As for drive axis plus additionally:	
	Interface for output cam, cam track (implicitly interconnected with the axis by the system when creating an output cam, cam track)	Specific
	Interface for measuring input (implicitly interconnected with the axis by the system when creating a measuring input)	Specific
Following axis		
	Input-side interconnection interfaces:	
	As for position axis, plus:	
	Interface for synchronous object (implicitly interconnected by the system when creating a following axis/following object)	Specific
	Output-side interconnection interfaces:	
	As for position axis, plus:	
Path axis		
	Input-side interconnection interfaces	
	Interface as for the following axis, in addition:	
	Path motion	Specific
	Output-side interconnection interface	
	As for position axis	
Following object		

TO	Interface	Type
	Input-side interconnection interfaces:	
	Motion input	Motion
	Cam	CAM
	Output-side interconnection interfaces:	
	Interface for slave axis (implicitly interconnected by the system when creating a following axis/following object)	Specific
Path object		
	Input-side interconnection interfaces:	
	Velocity profile	MotionProfile
	Output-side interconnection interfaces:	
	Path motion (path axis 1)	Specific
	Path motion (path axis 2)	Specific
	Path motion (path axis 3)	Specific
	Path synchronous motion	Specific
	Motion output (path object.x)	Motion
	Motion output (path object.y)	Motion
	Motion output (path object.z)	Motion
External encoder:		
	Input-side interconnection interfaces:	
	None	
	Output-side interconnection interfaces:	
	Interface for output cam, cam track (implicitly interconnected by the system when creating an output cam, cam track)	Specific
	Interface for measuring input (implicitly interconnected by the system when creating a measuring input)	Specific
	Motion setpoint	Motion
	Motion actual value with extrapolation	Motion
Measuring input		
	Input-side interconnection interfaces:	
	Measuring input interface (implicitly interconnected with the axis, external encoder by the system when creating a measuring input)	Specific
	Input interface 'Listening measuring input' (as of V4.0) (via general interconnection screen form)	Specific
	Output-side interconnection interfaces:	
	Output interface for 'Listening measuring input' (as of V4.0)	Specific
Output cam, cam track		
	Input-side interconnection interfaces:	
	Output cam interface (implicitly interconnected with the axis, external encoder by the system when creating an output cam, cam track)	Specific

TO	Interface	Type
Cam		
	Input-side interconnection interfaces:	
	None	
	Output-side interconnection interfaces:	
	Cam	CAM
	Motion profile	MotionProfile
	Force/pressure profile	ForceProfile
	Valve characteristic (hydraulic axis)	ValveCharacteristics
Temperature controller		
	No interconnection interfaces	
Addition object		
	Input-side interconnection interfaces:	
	Motion output 1	Motion
	Motion input 2	Motion
	Motion input 3	Motion
	Motion input 4	Motion
	Output-side interconnection interfaces:	
	Motion output	Motion
Formula object		
	Input-side interconnection interfaces:	
	Motion input 1	Motion
	Motion input 2	Motion
	Motion input 3	Motion
	LREAL input 1	LREAL
	LREAL input 2	LREAL
	LREAL input 3	LREAL
	LREAL input 4	LREAL
	DINT input 1	DINT
	DINT input 2	DINT
	DINT input 3	DINT
	DINT input 4	DINT
	Output-side interconnection interfaces:	
	Motion output 1	Motion
	Motion output 2	Motion
	Motion output 3	Motion
	LREAL output 1	LREAL
	LREAL output 2	LREAL
	LREAL output 3	LREAL
	LREAL output 4	LREAL
	DINT output 1	DINT
	DINT output 2	DINT
	DINT output 3	DINT
	DINT output 4	DINT

TO	Interface	Type
Fixed gear		
	Input-side interconnection interfaces:	
	Motion input	Motion
	Output-side interconnection interfaces:	
	Motion output	Motion
Sensor		
	Input-side interconnection interfaces:	
	None	
	Output-side interconnection interfaces:	
	Sensor value	LREAL
	Sensor value derivation	LREAL
	Motion output	Motion
Controller object		
	Input-side interconnection interfaces:	
	Setpoint	LREAL
	Actual value	LREAL
	Feedforward control	LREAL
	Motion input setpoint	Motion
	Motion input actual value	Motion
	Output-side interconnection interfaces:	
	Output value on motion output	Motion
	Output value	LREAL

Interconnection interface type 'Motion' (for experts)

The interconnection interface type **Motion** contains the motion vector (s, v, a), internal information such as validity status and, if applicable, modulo information.

Motion basis

The motion vector can have a different motion basis: position or velocity.

The position component is zero for the "velocity" motion basis.

A drive axis provides only the motion vector with the velocity motion basis on the output side.

Commands / command parameters (TO axis) or the configuration (TO additionObject, TO fixedGear) are used to consider/set the motion basis.

The components of the motion vector are maintained consistent at the output side of the TO axis, TO fixedGear (the derivation of the position value is the velocity value, the derivation of the velocity value is the acceleration).

The components of the motion vector are not maintained consistent at the output side of the TO additionObject and TO formulaObject.

The velocity and the acceleration can be specifically defined in this item for the position.

'MotionInput' input-side interconnection interfaces of the 'Motion' type

The TO type specifies whether a 'MotionInput' input-side interconnection interface of the Motion type can be interconnected once or several times.

The following technology objects have, for example, an input-side interconnection interface 'MotionInput' of the 'Motion' type:

- Drive axes
- Positioning axes
- Following axes
- Fixed gear
- Addition object
- Formula object
- Following object (master value)

Output-side interconnection interfaces of the 'Motion' type

The following technology objects have, for example, an output-side interconnection interface of the 'Motion' type:

- Drive axes
- Positioning axes
- Following axes
- Path object
- Fixed gear
- Addition object
- Formula object
- External encoder

Interconnection interface type 'LREAL' (for experts)

There is an interconnection interface of the LREAL type available for scalar variables.

Input-side interconnection interface on the axis 'TorqueLimitPositive' and 'TorqueLimitNegative' for specifying the torque limit.

To specify the **B+/B-** torque limits, the 'TorqueLimitPositive' and 'TorqueLimitNegative' input-side interconnection interfaces of the axis can be interconnected with output-side interconnection interfaces of the LREAL type of other TOs.

Prerequisite is the activation of the process-related field on the TO axis.

The input-side interfaces 'TorqueLimitPositive' and 'TorqueLimitNegative' cannot be interconnected several times and they cannot be distributed.

Output-side interconnection interfaces of the LREAL type have, for example:

- TO axis with the output-side interconnection interface 'Torque'
- TO formula object with the output-side interconnection interface 'LRealOut'

- Sensor TO
- Controller Object TO

'Torque' output-side interconnection interface on the axis for providing the actual torque

There is a 'Torque' output-side interconnection interface available on the axis for issuing the actual torque to other TOs.

Prerequisite is the activation of the process-related field on the TO axis.

The actual torque supplied by the drive in the additive process-related field is issued.

It is issued in the unit set on the axis.

The output-side interconnection interface can be interconnected several times.

The output-side interconnection interface 'ActualTorque' cannot be distributed.

'AdditiveTorque' Input-side interconnection interface on the axis for specifying an additive torque

There is a 'AdditiveTorque' input-side interconnection interface of the LREAL type available on the axis for specifying an additive torque to the drive using the additive process-related field.

To specify an additive torque depending on other technology objects, the input-side interconnection interface 'AdditiveTorque' can be interconnected with output-side interconnection interfaces of the LREAL type of other technology objects on the axis.

The input-side interface 'AdditiveTorque' cannot be interconnected several times and it cannot be distributed.

Output-side interconnection interfaces of the LREAL type have, for example:

- TO axis with the output-side interconnection interface 'Torque'
- TO formula object with the output-side interconnection interface 'LRealOut'
- Sensor TO
- Controller Object TO

4.2.3.6 Technology object trace

Introduction

You can monitor commands at a technology object and track access to the system variables and configuration data.

In SIMOTION, technology objects can be influenced by configuration data, system variables, and commands during runtime. In order to represent the influences on a technology object during runtime through the user program, a TO trace / object trace, which records the last n actions at the technology object and arranges these in chronological order, is available as of V4.2.

Note

Simultaneous use of the **Trigger through program call** trigger condition for the TO trace and the device trace is not permitted.

If the TO trace and the device trace are active simultaneously, different trigger conditions must be used.

The recording can be read out from the technology object on request and represented in the ES.

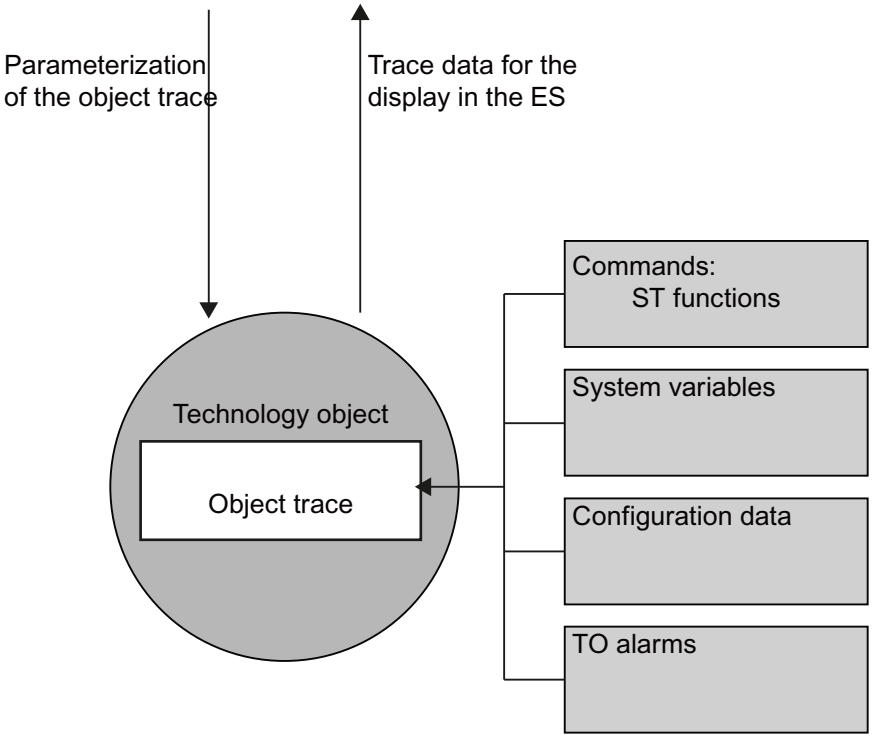


Figure 4-56 Principle of the object trace

The TO trace is used for the diagnostics of technology objects and displays the information and the events that influence a technology object during runtime in tabular form. The TO trace can be configured separately for each technology object.

It records commands and access to configuration data and system variables with their command and error statuses when the TO trace is read out. As of SIMOTION V4.4, alarms of the individual technology objects can also be recorded and an automatic trace is available. It can be used to trigger a recording automatically according to the trigger conditions immediately after a restart. You can parameterize the automatic trace in the Settings tab (Page 1180) at **Save in the device (memory card)**.

The TO trace is represented in a table/list with a time stamp.

The TO trace is available for the following technology objects:

- Drive axis
- Position axis
- Following axis
- Path interpolation
- Following object
- External encoder
- Measuring input
- Output cam
- Cam track
- Addition object
- Formula object
- Sensor
- Temperature channel
- Fixed gear
- Controller object

Events

Events

The TO trace supports recording of events which influence a technology object during runtime.

Events

These events include:

- Command execution
- Writing of system variables and
- Writing or activation of configuration data
- Alarms

TO trace for PLCopen function blocks

Block calls of PLCopen function blocks are entered in the command buffer. Edge-triggered blocks are entered in the buffer with their rising edge. Level-triggered blocks are entered in the buffer each time the input level changes.

TO trace for commands or function blocks

You can use the TO trace to read out the last commands issued to the technology object.

As command execution advances, the more information can be read out concerning the relevant commands. This means you can determine the status (executed or aborted) of commands which have already been completed/terminated as well as any error code. As the TO trace takes a snapshot, this information is based on the situation at the time of reading out. The status is not updated automatically once the command is finished. You can trigger an update by performing a further readout.

Note

The command value on the TO displayed for commands or function blocks is the value transferred on the interface.

The value that is active on the TO, e.g. through limits, is not displayed.

The TO trace provides the following information:

Table 4-15 Information on commands in the TO trace:

Parameter	Description
Command type	The command name in plain text can be ascertained from the command type.
Time stamp 1	The RT time for when the command is issued is stored in the time stamp.
Time stamp 2	The RT time for when the command takes effect is stored in the time stamp.
Time stamp 3	The RT time for when the command is completed is stored in the time stamp.
Taskstack	The call point for the command in the user program can be ascertained from the taskstack.

Parameter	Description
Command status / block status	Status of the command. The phase of the current motion is also given for motion commands. Possible values are as follows: • BUFFERED • WAIT_SYNC_START • IN_EXECUTION • IN_ACCELERATION • IN_CONSTANT_MOTION • IN_DECELERATION • INTERPOLATION_DONE • EXECUTED • ABORTED
Error code	Return value when the command is finished. The error code is only valid if the motion status is EXECUTED or ABORTED.
n command parameters	The TO trace records the transfer parameters of the command. The programmed value and the command value on the TO are recorded for each command parameter.
n return values	All the return values of the commands are also recorded, in addition to the command parameters. Recording takes place in the user program when the command is finished.

TO trace for configuration data

The TO trace records write accesses

Table 4-16 Information on configuration data in the TO trace

Parameter	Description
Configuration data element	Name of configuration data element in plain text
Time stamp	The RT time for when the configuration data element is written is stored in the time stamp.
Target data range	The target data range shows the effectiveness of the configuration data element. Possible values are the NEXT data range or the ACTUAL data range.
Data type	Date type of configuration data element
New value	Value written in the configuration data element

TO trace for system variables

The write accesses to the system variables through the user program are recorded with the TO trace

Table 4-17 Information on system variables in the TO trace

Parameter	Description
System variable name	System variable name in plain text
Time stamp	The RT time for when the system variable is written is stored in the time stamp.
Data type	Data type of the system variables
New system variable value	Value written in the system variable

TO trace for alarms

The TO trace records alarms

Table 4-18 Information on alarms in the TO trace

Parameter	Description
Time stamp	The following times are saved in the time stamp: • Time alarm occurred • Time alarm was acknowledged
Alarm number	Number of the alarm

Call

The TO trace can be called in a number of ways

Call via the main menu

You can call the TO trace by selecting **Target system -> Technology object trace**. You need to select a device in the project navigator; otherwise, the menu command remains hidden. Where there is more than one device, you can call the TO trace individually for each device.

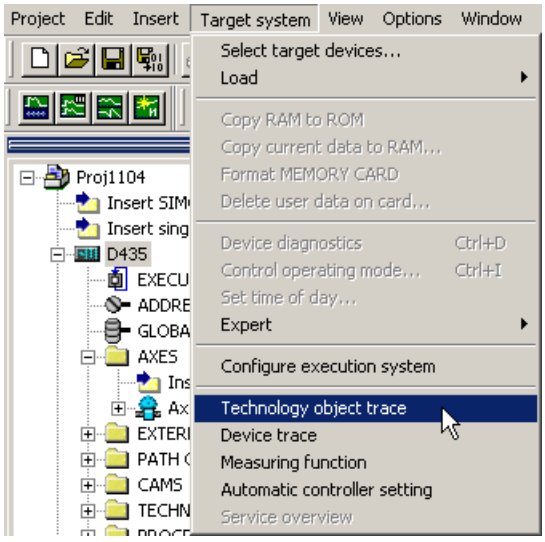


Figure 4-57 Call via the main menu

Call via the context menu

You can call the TO trace via the device's context menu.

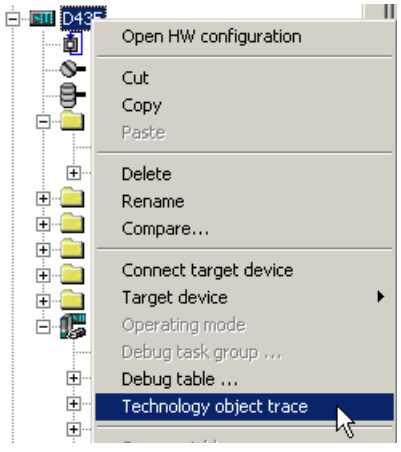


Figure 4-58 Call via the context menu

Call via the toolbar

The toolbar contains the button for calling the TO trace. The button is activated when a device is selected in the project navigator.

Parameterization

Overview of the parameterization

The parameterization and trace data are available both offline and online.

Only part of the overall functionality is available offline:

- Parameterization of the TO trace
- Displaying the latest TO trace to be uploaded
- Saving the data to a TO trace file
- Opening an existing TO trace

Additional online functionality:

- Parameterizing the TO trace and downloading it to RT
- Starting, stopping the trace recording
- Uploading and downloading the parameterization
- Delete the parameterization in the target system

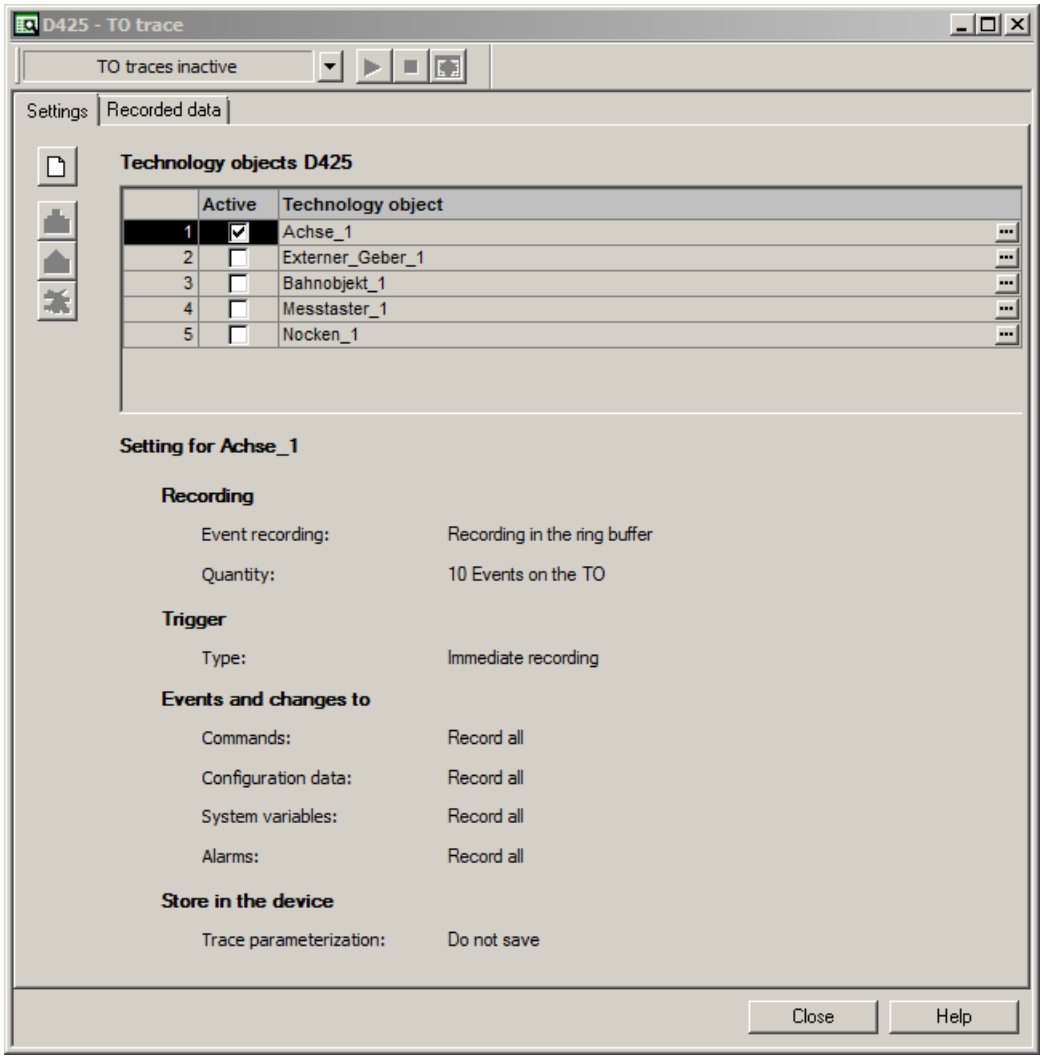


Figure 4-59 Dialog for parameterization

Toolbar

You control the recordings via the TO trace toolbar. The toolbar is displayed with the snap-in. This toolbar can be used to start or stop the trace for all technology objects and to upload a recording. The device to which the trace applies is fixed and cannot be changed. A separate trace window is opened for each device.

Table 4-19 Possible settings

Field/button	Description
TO traces inactive	Current state of the trace for all technology objects The field displays the state and availability of the TO trace. **Possible states:** • TO trace inactive (if all are complete or yet to start) • TO trace parameterization faulty (if at least one has an incorrect parameterization) • TO trace active (if at least one is running or waiting for a trigger)
	Opens a drop-down list to display the states of the individual TOs **Possible states:** • Inactive • Ready • Waiting for trigger • Running • Parameterization faulty • Recording finished
	Start TO trace
	Stop TO trace
	Upload data TO trace data can be uploaded at any time. If an endless trace is set at the time of the upload, the recording is appended to the existing trace in each case. If the time-limited trace is activated, the existing trace is overwritten.

Settings

The tab displays the technology objects created below the device. The settings of the Technology object trace event selection (Page 1180) dialog box are displayed below the table for the technology object selected in the table.

Table 4-20 Possible settings

Function	Description
Menu commands	
	Reset parameterization
	Download parameterization
	Upload parameterization

Function	Description
	Delete the parameterization in the target system
Technology objects	All the technology objects created in the project are displayed under the **Technology objects** setting. To select a line, click any cell within it. The settings for a line are displayed under the technology objects. Use the  button to change the settings for a technology object. All the technology objects are deselected under the default setting. The relevant technology object must be selected for the recording.
No.	Number of the technology object
Active	Activates or deactivates the trace for the technology object
Technology object	Name of the technology object
	The Technology object trace event selection (Page 1180) dialog box is called with this command.

Recorded data

The tab displays the recorded data and enables results to be saved and loaded in XML format.

Table 4-21 Possible settings

Function	Description
Menu commands	
	Load result
	Save results in XML format.
Reference time	Time specification to which the time stamp of the recorded events refer. The absolute time of the controller is used for the TO trace recording. Through the specification of the absolute time of the controller from the device or system trace recording as reference time, TO trace recordings can be chronologically assigned to the trace recordings. The absolute time of the controller in the device or system trace recording is displayed with the measuring cursor.
Time specifications relative to the reference time	Activation or deactivation of the reference time for the display
Columns	
Type	Type of recording
Time stamp	Time stamp of recording
Technology object	The reference object
Event	The event recorded
Current status	Status of recording
Detailed information for the selection	Detailed information is displayed in tabular form. Chronological sequence of events is structured as follows for time stamp, code position, event and current status: - Issued - Has become effective - Completed/aborted

Note

With PLCopen blocks, the return values are also current values and change while the block is running. In addition, the time stamp for **issued** corresponds to the **execute edge** on the block.

Technology object trace event selection

Commands, configuration data, and system variables are always selected as the default setting. Signals can be deselected on the individual tabs.

Make the settings for the recording for the selected TO in the **Technology Object Trace Event Selection** dialog. You can select or deselect certain commands, configuration data, system variables and alarms for the recording in separate tabs. Activate or deactivate the checkbox of a node to select or deselect the whole of the branch beneath.

The commands are sorted and grouped in the same way in which they are displayed in the command library.

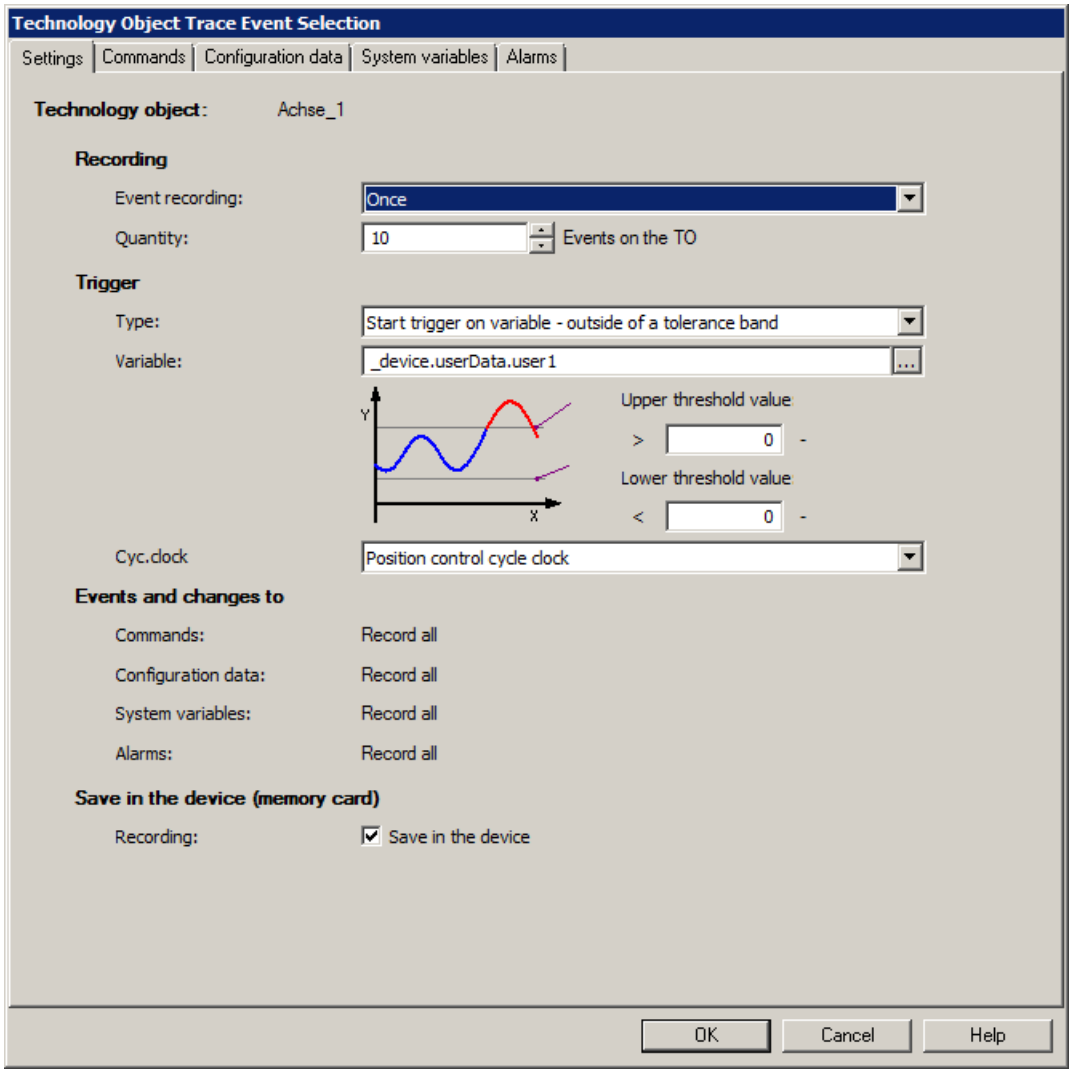


Figure 4-60 Technology Object Trace Event Selection

Table 4-22 You can set the following parameters:

Field/button	Meaning/instruction
Record	
Event recording	Recording of the events once or endlessly in a ring buffer With the setting **Once**, the trigger acts as the start trigger, with **Recording in the ring buffer** as the stop trigger.
Number	Buffer size of the number of events on the TO Up to 100 events can be recorded. The default setting is 10 events.
Trigger (the description is formulated for a start trigger and analogously is also valid for the stop trigger)	
Type	**Record immediately** Recording is started with the start of the TO trace.

Field/button	Meaning/instruction		
	Start trigger on variable ... Recording is started when a certain trigger threshold of the variable value is reached. The threshold value to be reached and a specific system variable can be specified for the trigger. Select the **cycle clock** for this trigger type. • **Positive edge** If the entered trigger threshold is exceeded, the signal recording will be started. You can enter the trigger threshold under **Threshold value**. • **Negative edge** If the entered trigger threshold is undershot, the signal recording will be started. You can enter the trigger threshold under **Threshold value**. • **Within a tolerance band** The signal recording is started when the variable value enters a tolerance band. Enter the **lower** and **upper threshold value**. • **Outside a tolerance band** The signal recording is started when the variable value leaves a tolerance band. Enter the **lower** and **upper threshold value**.		
		Variable	Variable used as trigger
		Threshold value	Value that is to be used as the trigger threshold
		Upper/lower threshold value	Upper and lower threshold value for the tolerance band
		Cycle clock	Cycle clock in which the monitoring is to be performed for when the trigger event occurs. **Servo cycle clock** The servo cycle clock is used as the basis for the trigger. **IPO cycle clock** The IPO cycle clock is used as the basis for the trigger. **IPO2 cycle clock** The IPO2 cycle clock is used as the basis for the trigger.
	Start trigger through 'TraceTrigger1' program call and **Start trigger through 'TraceTrigger2' program call** The recording starts when the program call is triggered by the trigger event. In MCC, use the "Activate Trace" command and in ST, you can set the tracecontrol[0].tracetrigger system variable to the value "Start". **Note** Simultaneous use of the **Trigger through program call** trigger condition for the TO trace and the device trace is not permitted. If the TO trace and the device trace are active simultaneously, different trigger conditions must be used.		
	Start trigger at code position The recording of variables is triggered by a defined code position in the program once the code position has been executed in the program sequence. You can parameterize the code position at **Program source**, **Program line** and **Trigger in the task**.		

Field/button	Meaning/instruction		
		Program source	Name of the program from which the program variables are to be recorded. Note that only the name of the source is to be entered without a preceding controller name.
		Program line	Enter here the program line of the code position where the recording is to be started. Note that only lines containing executable code or program-local variable declarations will result in a recording. Empty program lines, lines that only contain comments or program lines with global declarations will result in an error message. How to calculate the program line: 1. Open the cross-reference list. 2. Select a line or cell in the list. If the **Calculate program line...** button is selected then the line can be used for the trigger. 3. Open the dialog to display the program line using the **Calculate program line...** button. 4. Adopt the line number displayed in the dialog as the **program line** for the recording. **Note** With ST programs the number displayed in the cross-reference list under **Line/Block** can be adopted as the **program line**. With MCC or LAD/FBD programs the **program line** must always be calculated using the **Calculate program line...** button. The program line calculated here does not match the number displayed under **Line/Block**.

Field/button	Meaning/instruction		
		After the n-th pass	How often the code position is to be executed until the measured value is acquired. The measured value acquisition is performed cyclically after each n-th pass.
		Trigger in the task	Task in which the program variables are to be recorded. If you have assigned the program to several tasks, only the task selected here will be considered for the recording.
Events and changes of			
	Commands	Displays whether all or only selected objects are to be recorded.	
	Configuration data	Displays whether all or only selected objects are to be recorded.	
	System variables	Displays whether all or only selected objects are to be recorded.	
	Alarms	Displays whether all or only selected objects are to be recorded.	
Save in the device (memory card) (as of SIMOTION V4.4)			
	Trace parameterization	Activation of saving in the device The trace parameterization is stored in the device or in the file system. Activate the TO trace in the Toolbar (Page 1178) with the **Start TO trace** button. The TO trace is activated automatically during ramp-up. These measurements are retained after the controller is switched off. Read out the measurements with SIMOTION SCOUT (toolbar - **Upload data** button) or SIMOTION IT DIAG.	

Examples/applications

Saving the TO trace recording to the device (memory card)

The TO trace recordings can be saved retentively in the device with the **Save in the device** setting. In this way, a loaded TO trace can be automatically activated again the next time the controller is switched on.

Requirement

- The TO trace has been opened.
- An online connection has been established to the device.

Procedure

To parameterize a TO trace and save the recording in the device, proceed as follows:

1. In the Settings (Page 1178) tab of the TO trace for a technology object, call the Technology Object Trace Event Selection (Page 1180) dialog box.
2. Activate the **Save in the device** checkbox.
3. Close the Technology Object Trace Event Selection (Page 1180) dialog box with **OK**.

4. Load the parameterization to the device.
   The icon for deleting the parameterization in the target system is activated.

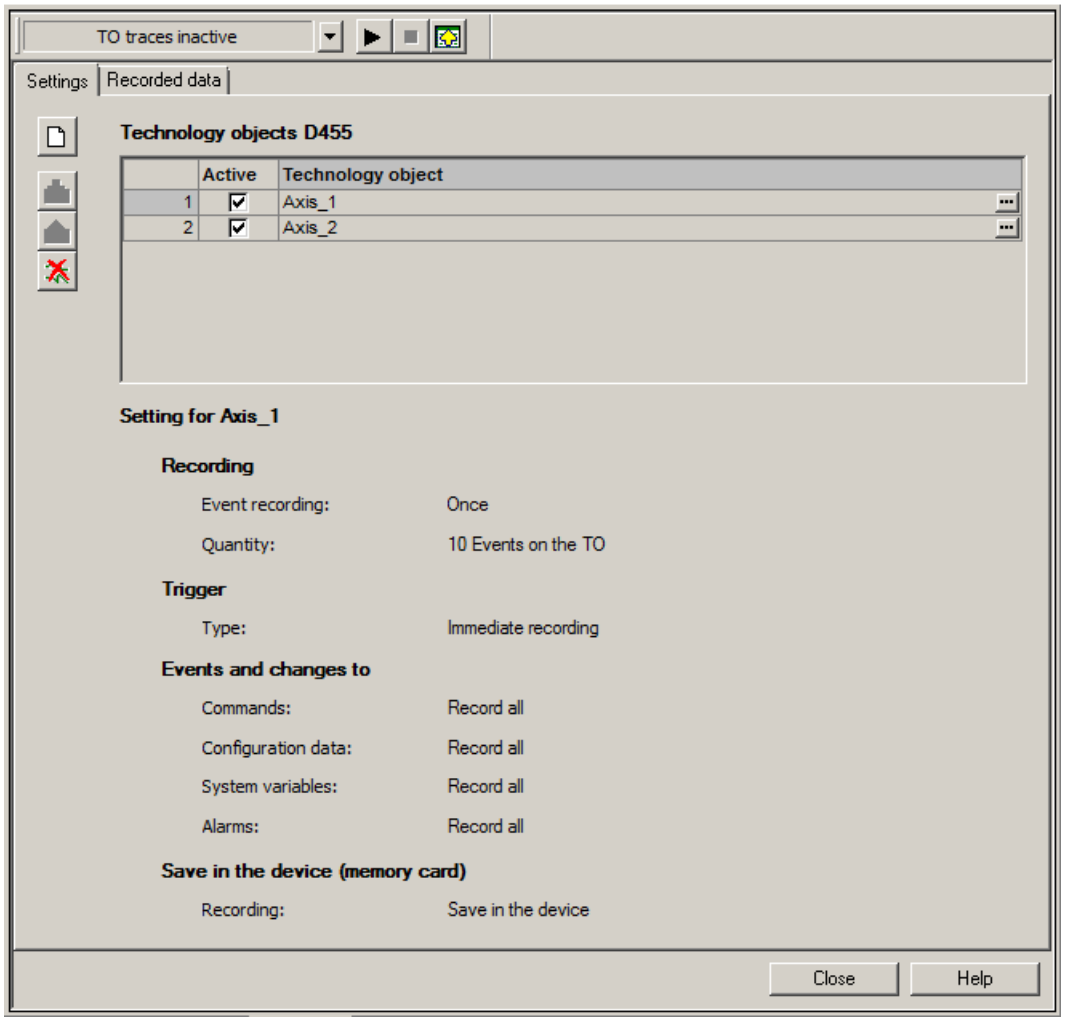


Figure 4-61 Settings tab for loaded parameterization in the device

5. Start the trace recording.
   The status is displayed in the toolbar.

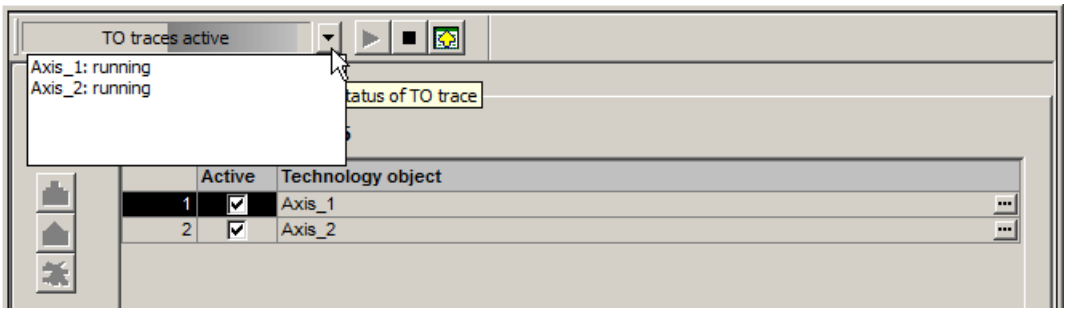


Figure 4-62 Status of the individual TOs

Reading out the TO trace recording from the device after a reconnection

TO trace recordings with the **Save in the device** setting can be read out with the TO trace even after disconnection of the online connection.

Note

A download of the TO trace parameterization may delete existing recordings in the device.

Requirement

The TO trace has been opened.

Procedure

To read out a recording saved in the device with TO trace, proceed as follows:

1. Establish an online connection to the device.
   The activated buttons to upload the parameterization and delete the parameterization in the target system indicate that a TO trace parameterization is available on the device.

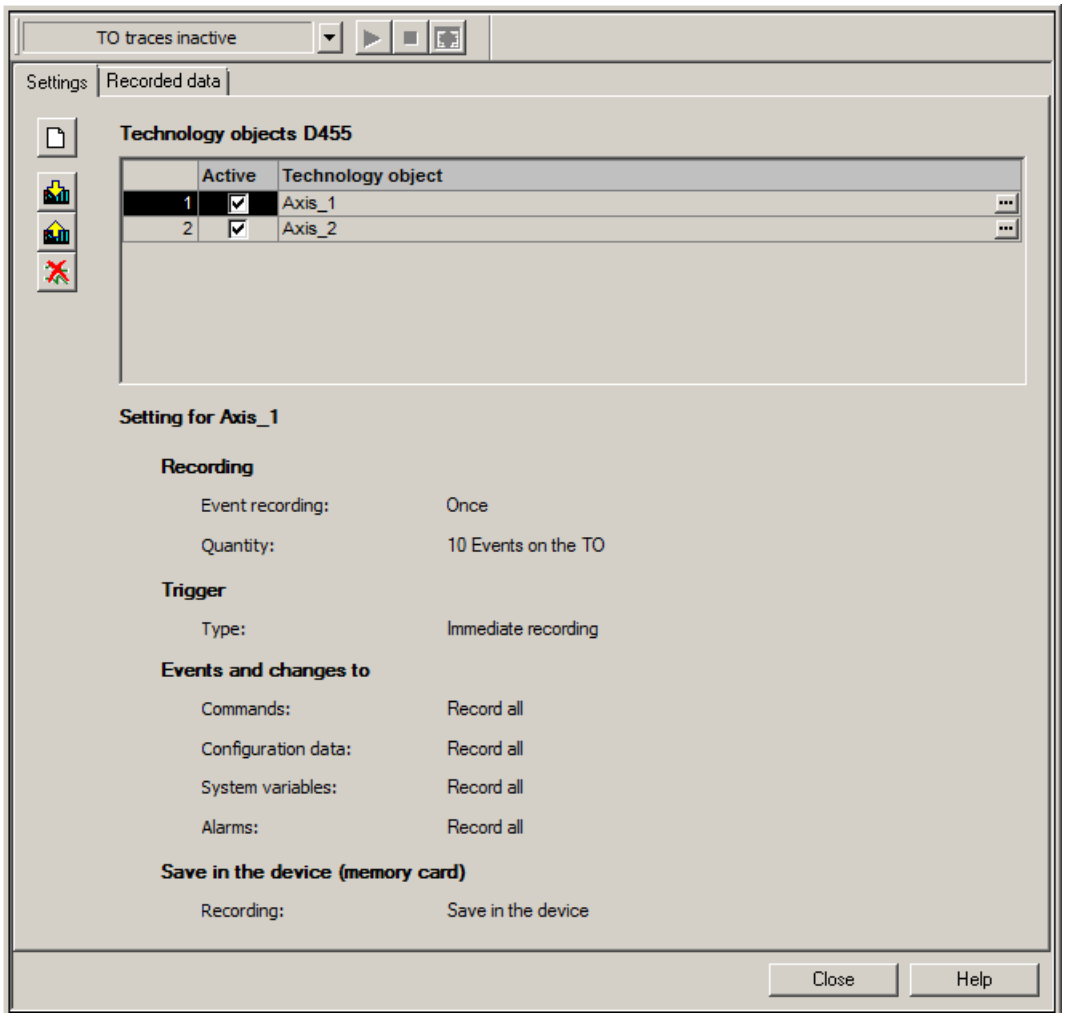


Figure 4-63 Settings tab after a reconnection when a TO trace parameterization is available on the device

2. Click the button to upload the parameterization.

Result

The elements of the offline parameterization are replaced by the online parameterization.

TO trace shows the status and, if required, the results gathered since the start of the trace. If the TO trace is still active in the device, further results are recorded.

See also

Technology object trace event selection (Page 1180)

Call (Page 1174)

4.2.4 Symbolic assignment (as of V4.2)

4.2.4.1 Symbolic assignment - introduction

Symbolic assignment of axis and drive

As of version V4.2, SIMOTION supports symbolic assignment to SINAMICS drive objects (DOs, drive objects) in the context of the configuration of technology objects (TOs) and I/Os.

This simplifies the configuration of the technological links including communication between the controller and the drive.

By means of symbolic assignment:

- Only compatible assignment partners are listed for selection in an assignment dialog
- The engineering system sets up communication between axis and drive automatically, along with the required PROFIdrive axis telegrams and the addresses used
- Telegrams are extended and assignments are created automatically in the drive depending on the selected TO technology (e.g. SINAMICS Safety Integrated)
- Axis and drive configuration can initially be carried out independently of one another
- Communication links are established automatically during the configuration of I/O variables on SINAMICS I/Os (telegrams and addresses are set up automatically and the I/Os are interconnected with the telegram).

Thus, other than symbolic assignment, no more configuration settings are required for communication. Since addresses no longer have to be configured, the connection is maintained even if addresses change.

Note

You can deactivate **automatic telegram configuration** and **automatic telegram adaptation** when configuring drive objects (DO drive, DO encoder, etc.) and in the dialog for telegram configuration.

If deactivated you will lose many of the benefits referred to above, we recommend it only in justifiable exceptional circumstances.

The TO axis, TO external encoder, TO output cam, TO cam track, and TO measuring input support symbolic assignment. It is also possible to establish symbolic assignment involving the onboard I/O of a SIMOTION D, a SINAMICS S110/S120 Control Unit, the TB30 and selected Terminal Modules.

The following components support symbolic assignment:

Module	Supports symbolic assignment
SIMOTION C, P, D	As of SIMOTION V4.2
Controller Extension • CX32-2 • CX32	As of SIMOTION V4.2
SINAMICS S110 CU305	As of SINAMICS V4.3
SINAMICS S120 • CU310 • CU310-2 • CU320 • CU320-2	 • As of SINAMICS V2.6.2 • As of SINAMICS V4.4 • As of SINAMICS V2.6.2 • As of SINAMICS V4.3
TB30, TM15 DI/DO, TM31	As of SIMOTION V4.2
TM41 (not DI/DO or AI)	As of SIMOTION V4.2
TM15, TM17 High Feature	As of SIMOTION V4.2

Note

The previous method of drive/axis and I/O configuration continues to be available. Symbolic assignment must be deactivated in such cases. Symbolic assignment is used by default for newly created projects.

Where projects are being upgraded to V4.2, symbolic assignment is deactivated by default and needs to be activated if required.

Symbolic assignment can be activated/deactivated in SIMOTION SCOUT via the menu "Project" > "Use symbolic assignment", see Using symbolic assignment (Page 1218).

Adaptation

In addition to the benefits offered by symbolic assignment, configuration is also made easier as of SIMOTION V4.2 with the addition of automatic adaptation of SINAMICS S120 data. When SIMOTION devices ramp up, reference variables, along with drive and encoder data for SINAMICS S120, are transferred automatically for the configuration data of the SIMOTION technology objects "Axis" and "External Encoder". This data no longer has to be entered in SIMOTION.

Events, after which an adaptation is carried out:

- When the controller boots
- STOP->RUN transition, if no adaptation has been made yet
- In the case of a restart of a TO
- _activateTO
- For _enableAxisInterface, if the status "not adapted" is present at the interface

- Via a user command (driveControlConfig)
- After a data set changeover; encoder changeover does not cause adaptation

For further information, see Setting as a real axis with digital drive coupling in the Axis TO Electrical/Hydraulic, External Encoder Function Manual.

4.2.4.2 Symbolic assignment of TOs

Assigning the TO axis and TO external encoder

Description

In the axis wizard, the assignment dialog shows the available drives which can be symbolically assigned to the axis.

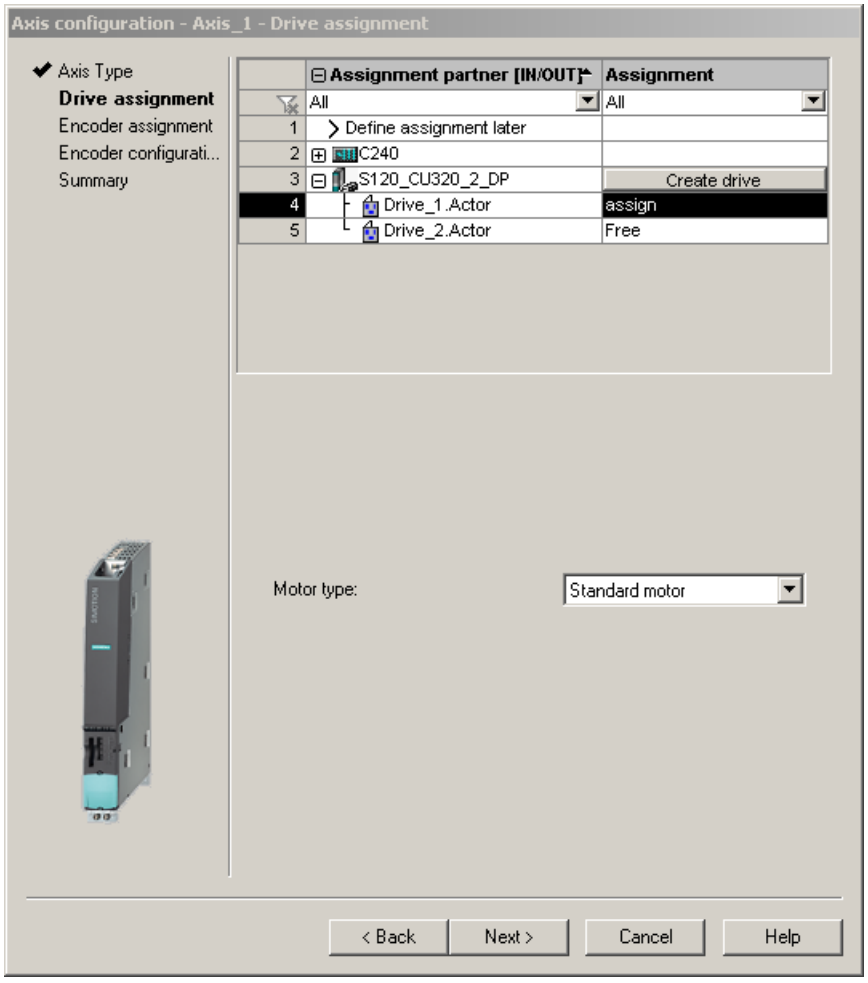


Figure 4-64 Assign axis and drive

Setting options

The following setting options are available:

- Assign drive
  Assign a previously configured drive
- Assign later
  The axis should not be assigned to a drive until a later point in time.
  Accordingly:
  - A programmer can use technology objects (Axis TO, for example) to make all the necessary configuration settings for the PLC and motion control functions and load them to the device; drive know-how is not required.
  - The drives can be configured and optimized separately by a drive expert and then, at a later point in time, the technology objects can be symbolically assigned to the drive objects via an interconnection dialog
- Create drive
  This function, which can be accessed in the assignment dialog, creates a new drive on an existing drive unit (e.g. S120 CU320-2 or SINAMICS Integrated) and assigns it to the axis directly. This allows the axis, including the drive, to be created in a single operation. It is not necessary to configure a drive before creating an axis.

Note

If you want to operate the drive directly via an application and not via a technology object, you must deactivate the "Automatic telegram setting" on the drive object, see also Message frame configuration (Page 1212).

Encoder assignment

With a position axis, encoder 1 is also created at the Axis TO (motor encoder) and automatically assigned to the first encoder on the drive.

If encoder 2 (direct encoder) is created at the Axis TO, it is assigned to the second encoder of the drive control. See also axis wizard overview.

On completion of the axis wizard, the symbolic drive assignment can be viewed using the following methods:

- Via the "configuration" of the axis
- Via the address list (view all addresses)

The assignment dialog can be called again from these dialogs using the " ... " button.

Instead of calling the assignment dialog, it is also possible to edit the input field containing the symbolic name directly.

Technology data block (TDB) and drive safety data block (DSDB)

The activation of the technology data block and support for SINAMICS Safety Integrated functions by the technology object can be set in the configuration dialog of the Axis TO for functions under the "Change" button. The assignment is always made to the drive DO of the axis actuator. The system automatically generates a telegram extension and the BICO interconnection of the relevant SINAMICS parameters.

Assignment of the I/O signals on the Axis TO

For the assignment of I/O signals at the Axis TO, e.g. the inputs for the homing output cam or hardware limit switches, call the assignment dialog from the axis dialogs or from the address list (view all addresses) by clicking the " ... " button.

The detailed structure and the identifiers of the component types for symbolic assignment can be found in Assignment types for symbolic assignment (Page 1783).

For further information about drive assignment, see the Axis Function Manual.

Address list

The address list in the "All addresses" view provides an overview of the assignments to all interfaces of the Axis TO. From this view, the assignments can also be changed via the assignment dialog (... button).

See also

Assigning I/O variables via the address list (Page 1202)

Using the assignment dialog (Page 1207)

Assigning the TO measuring input

Description

The assignment dialog displays the available encoders can be symbolically assigned to the external encoder.

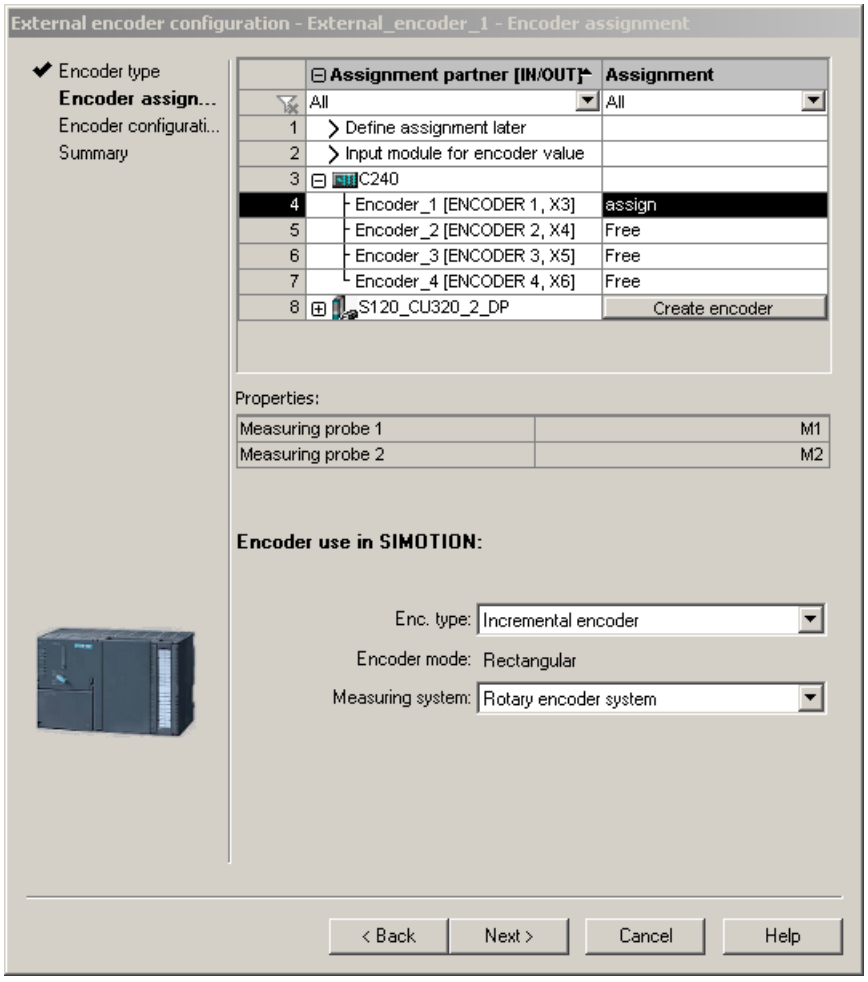


Figure 4-65 Assign External Encoder TO

For information about the axis wizard, which can also be used to configure the external encoder DO, see Overview of axis wizard.

See also

Using the assignment dialog (Page 1207)

Assigning the TO output cam and cam track

Description

Before configuring the technology object, the functionality of the terminals must be configured accordingly (configuration of a DI/DO as a digital output, for example).

The configuration settings for the technology objects to access I/Os are made symbolically, whereby only "function-compatible" I/O channels are listed for selection in the assignment dialog.

Example 1:

Only MI (measuring input) type symbolic assignments are listed for selection for the Measuring Input TO.

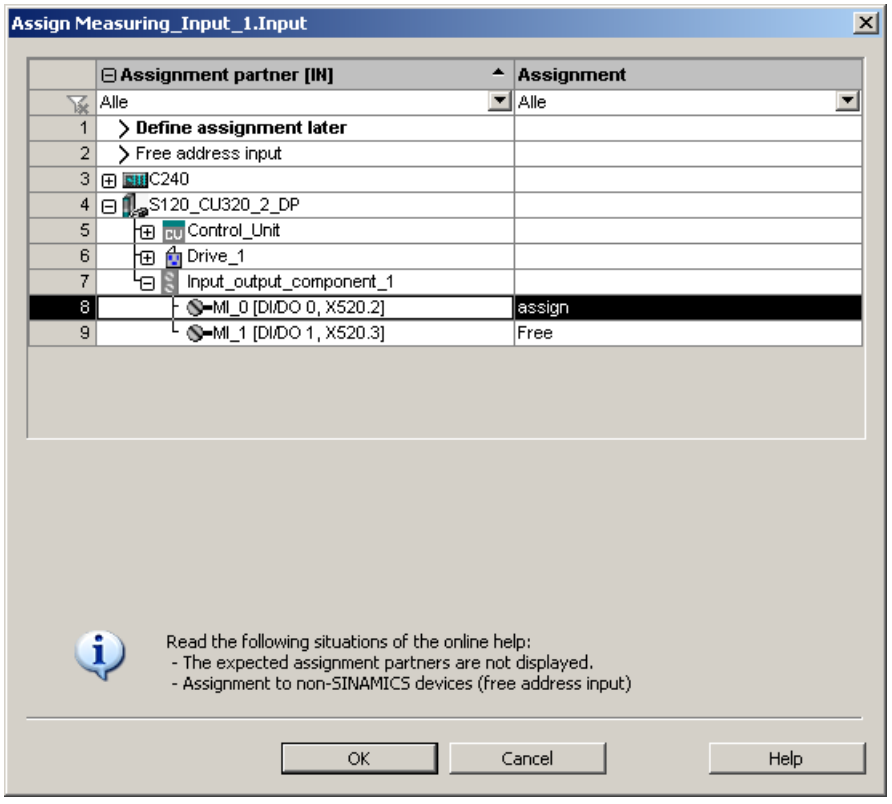


Figure 4-66 Assign measuring input

For further information, see the Output Cams and Measuring Inputs Function Manual:

- Assigning the TO output cam
- Assigning the TO cam track
- Assigning the TO measuring input

For information on the assignment dialog, see Using the assignment dialog (Page 1207).

Assigning the TO sensor

Description

It might be necessary to configure the functionality of the terminals prior to assignment.

The analog inputs are listed for selection of the TO sensor in the assignment dialog.

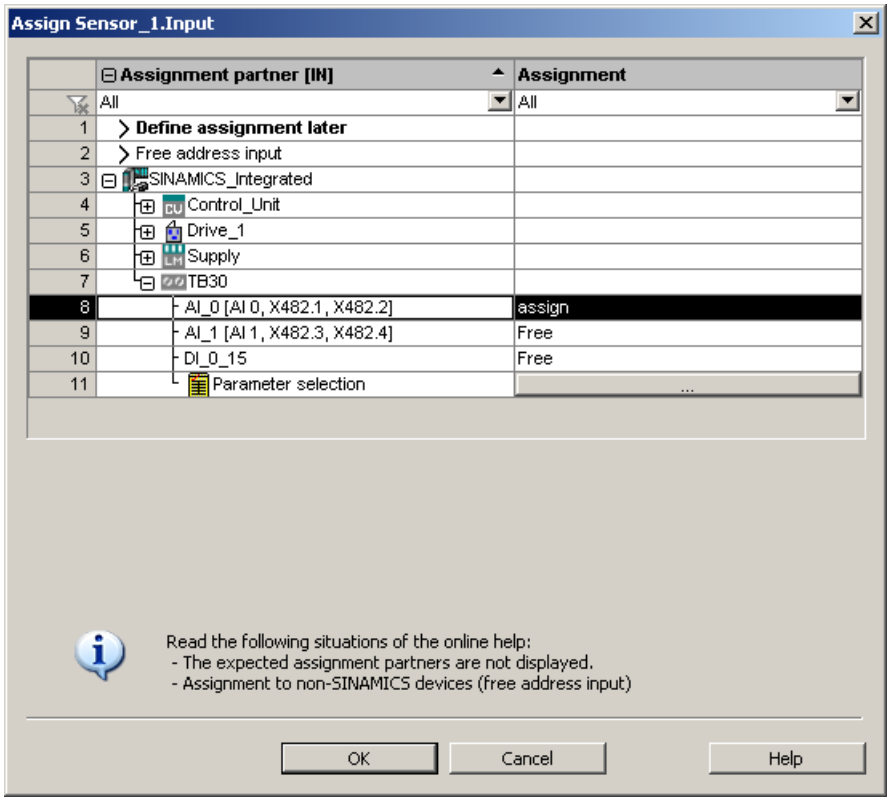


Figure 4-67 Assigning the TO sensor

For a description of the TO sensor, see the section titled TO Sensor in the "Additional technology objects" Function Manual.

The section titled Using the assignment dialog (Page 1207) contains a description of the assignment dialog.

4.2.4.3 Symbolic assignment of I/O variables

Symbolic assignment of I/O variables to the PROFIdrive message frame of the Axis TO

Description

You can assign I/O variables from the address list which you require for display and diagnostic purposes, for example, to individual components (status word, for example) of the PROFIdrive telegram using the assignment dialog. Only components suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, the data type this is determined by the assignment partner after selection.

To assign an I/O variable to an individual component of a PROFIdrive telegram

1. Open the address list and create a variable.
2. During I/O variable configuration, enter the identifier IN or OUT (or make a selection from the drop-down list box) under the I/O address for the direction.
3. Select the data type and open the assignment dialog by clicking the "..." button.
   The assignment dialog will then only show assignment targets which are suitable for this data type. If you do not select a data type, all parameters will be offered. The assignment dialog returns the parameter data type when it is closed.

4. Under Assignment partner, select the component (e.g. the word) that you wish to assign.
5. Click **OK** to assign the component.

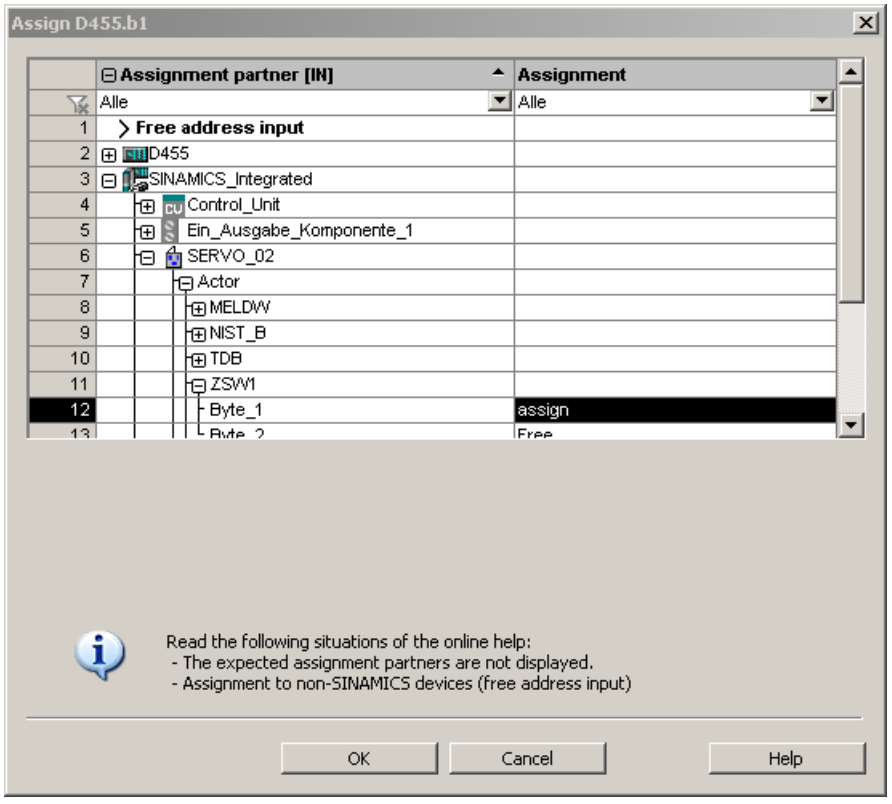


Figure 4-68 Assigning I/O variables to the PROFIdrive telegram of TO Axis

The detailed structure and the identifiers of the component types for symbolic assignment can be found in Assignment types in standard telegrams (Page 1788).

Interconnecting parameters via BICO

Description

I/O variables from the address list can be assigned to drive parameters using the assignment dialog. Only parameters suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, this is determined following parameter selection.

The system automatically extends the standard telegram for the drive for the transfer of the parameters.

BICO interconnection

You can interconnect a BICO parameter with a telegram by specifying the parameter numbers, the transmission width (WORD, DWORD), and the transmission direction in the assignment dialog.

As part of this, you can also select the parameter of a different signal source (SINAMICS DO). The parameter will then be transmitted automatically along with the telegram of the assignment partner.

How to assign I/O variables to drive parameters

1. Open the address list.
2. During I/O variable configuration, enter the identifier IN or OUT (or make a selection from the drop down list box) under the I/O address for the direction.
3. Select the data type and open the assignment dialog by clicking the "..." button.
   The assignment dialog will then only show assignment destinations which are suitable for this data type. If you do not select a data type, all parameters will be offered. The assignment dialog returns the parameter data type when it is closed.

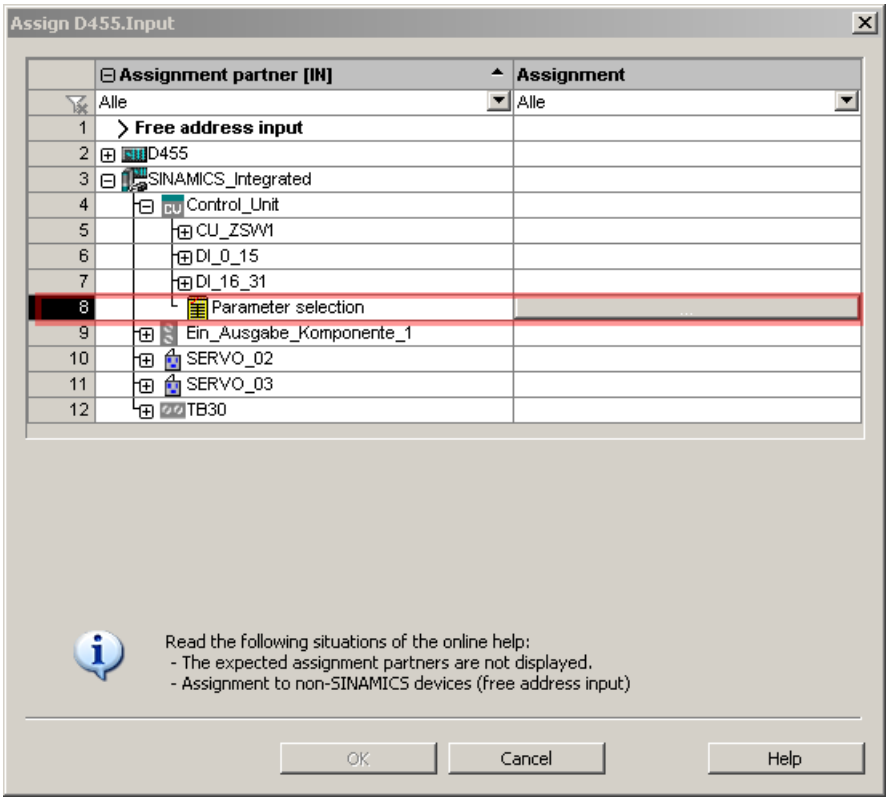


Figure 4-69 Assignment dialog for drive parameters

4. Click the ... button in the parameter selection line to open the parameter list.

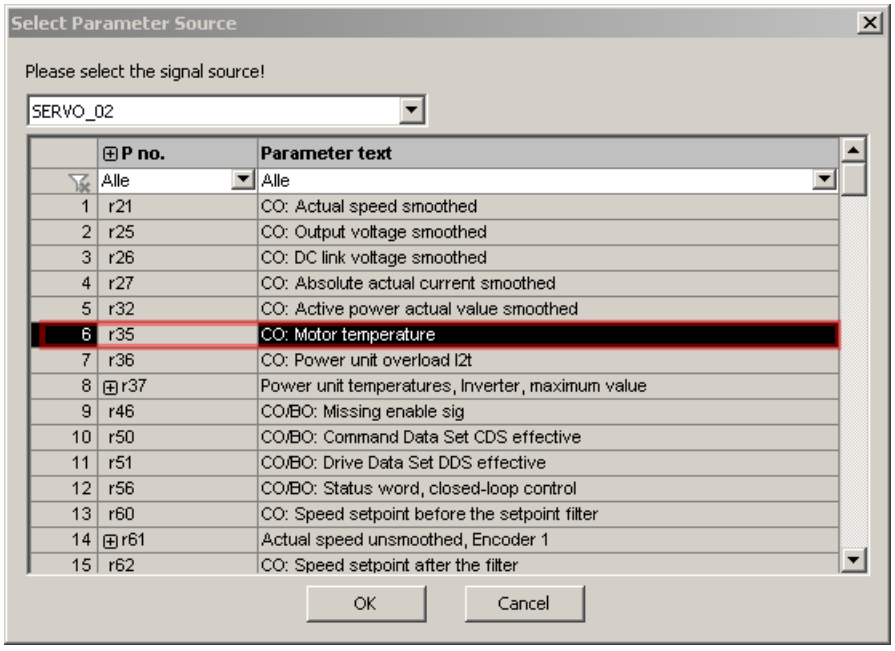


Figure 4-70 Dialog for parameter and DO selection

5. If necessary, select the signal source (SINAMICS DO) which makes the parameter available, followed by the parameter itself.
6. Click **OK** to accept the selection.

7. A SINAMICS parameter is assigned to the I/O variable in the assignment dialog.

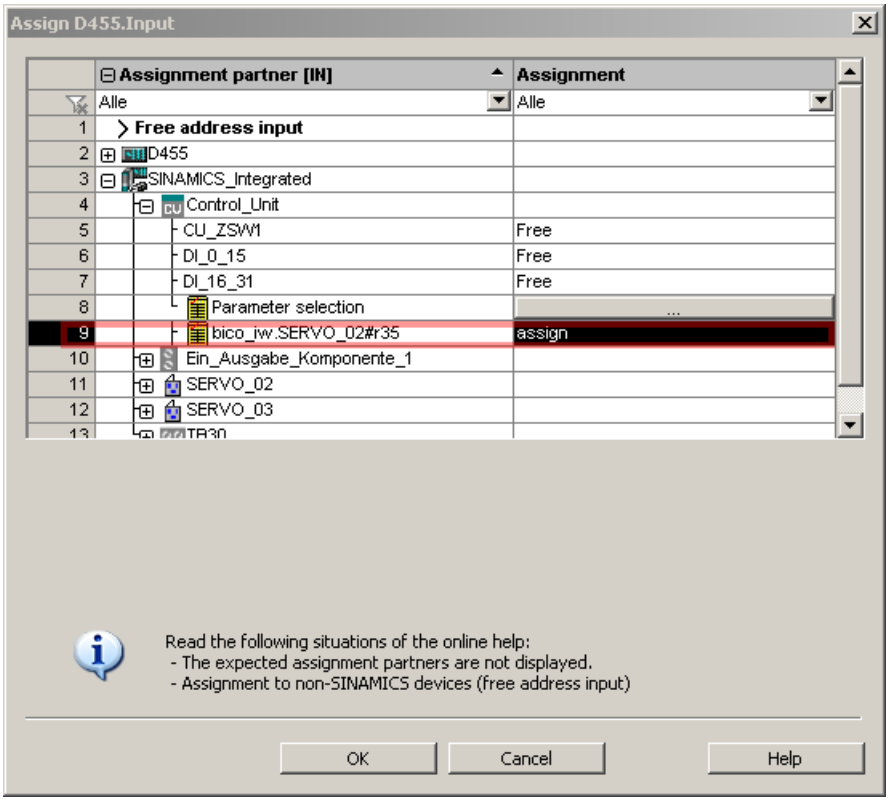


Figure 4-71 Assigned drive parameter

8. Click **OK** to accept the assignment.

Assignment types

The following table shows the possible types of assignment:

Name of the BICO interface	Data type	Direc-tion	Transferrable BICO parameters
BICO_IW.<parameter number>	WORD	Input	All CO parameters (BICO source)
BICO_QW.<parameter number>	WORD	Output	All CI parameters (BICO sink)
BICO_ID.<parameter number>	DWORD	Input	All CO parameters (BICO source)
BICO_QD.<parameter number>	DWORD	Output	All CI parameters (BICO sink)

Syntax of the assignment names:

- A number of parameters (separated by periods) are specified for outputs (SINAMICS side = received data) which can be interconnected with a number of BICO sinks.
- If the transferred parameter is on another DO, the DO name precedes the parameter. "#" is used as a separator between the DO name and the parameter.
- Individually transferred bits of a parameter appear in brackets [x].

Assigning I/O variables via the address list

Description

Prior to assignment, the functionality of the terminals might have to be configured accordingly (configuration of a DI/DO as a digital output, for example).

There are several ways to assign an I/O variable to terminals:

- Assignment of a terminal signal via the SIMOTION preferential interconnection. To do this, you must use the SIMOTION preferential interconnection for the corresponding input/ outputs of the SINAMICS DOs. The BICO interconnection is automatically executed.
- Assignment of a terminal signal via PZD interfaces, e.g. DI_0_15 or DO_0_15. The interfaces are, for example, assigned to PZD 2 (A_DIGITAL or E_DIGITAL).
  Please note for these signals that the BICO interconnection is not executed (e.g. for TB30, TM31 or TM15DIDO), although a q of the appropriate length is generated.

To activate the SIMOTION preferential interconnection for all terminals in a SINAMICS DO

1. In the project navigator, open the folder inputs/outputs of the relevant drive DOs.

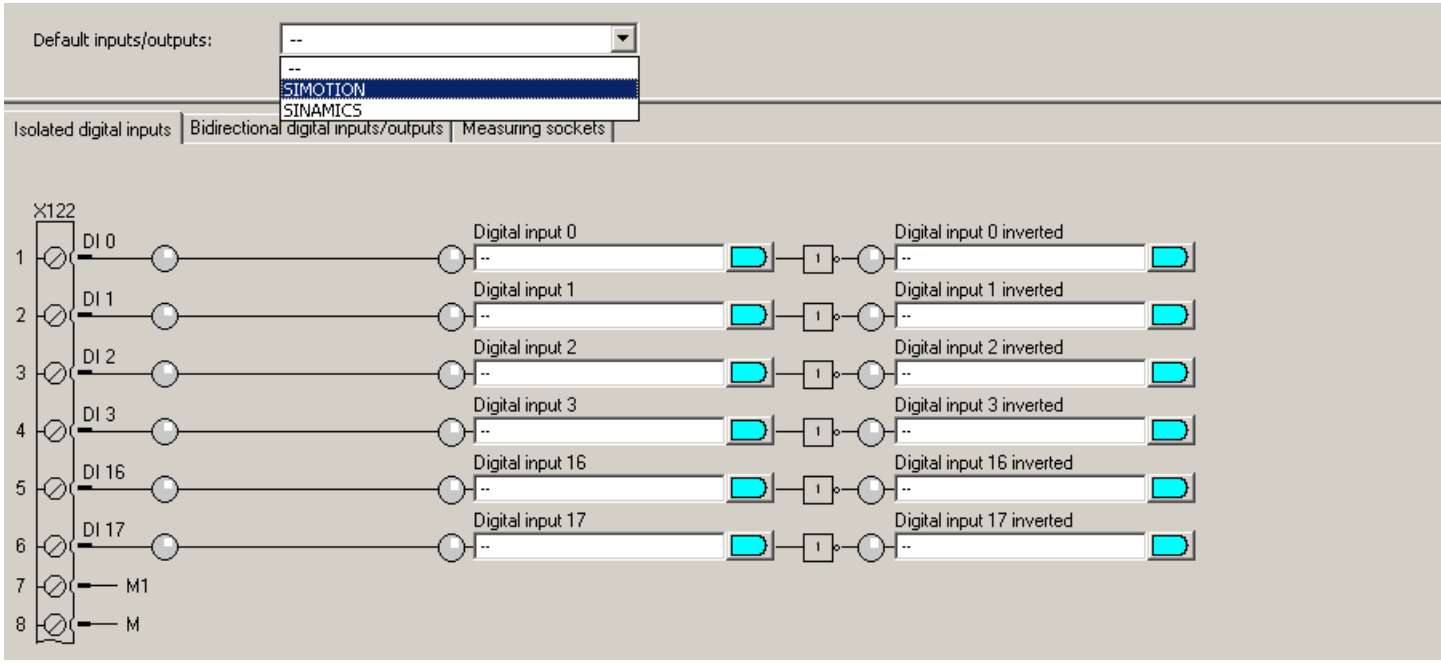


Figure 4-72 SIMOTION preferential interconnection

2. Select the entry **SIMOTION** in the **Default inputs/outputs** drop-down list box. All inputs/outputs are assigned to the SIMOTION preferential interconnections.

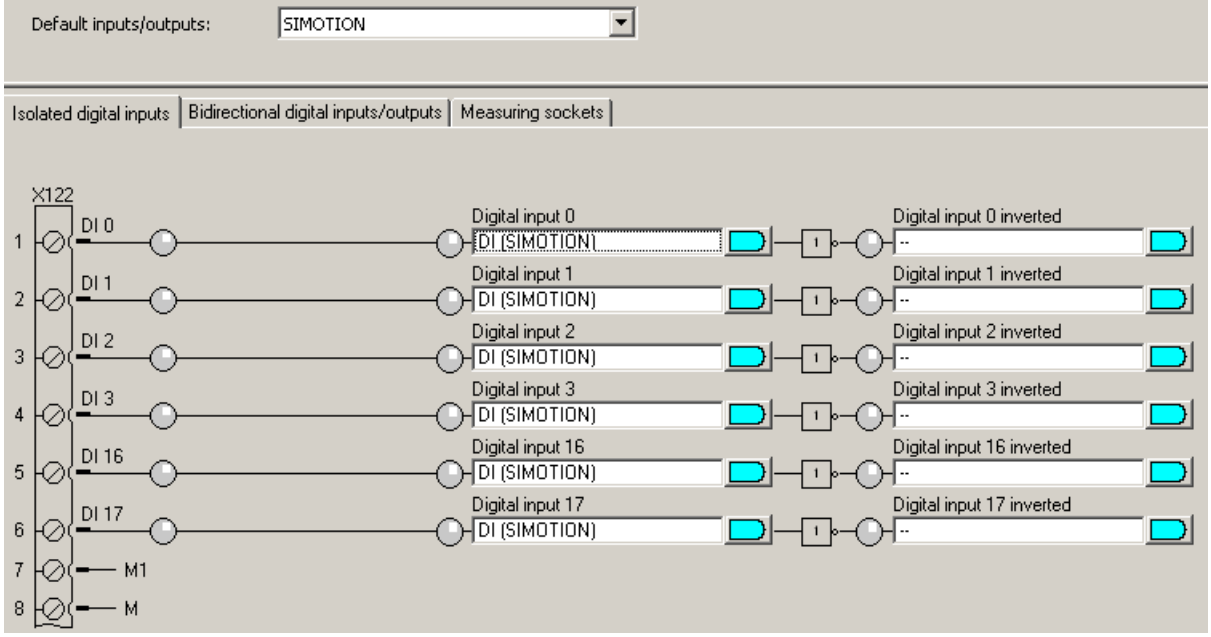


Figure 4-73 SIMOTION preferential interconnection active

With this setting, the BICO interconnections will be performed automatically.

If you wish to undo the interconnection, you must select SINAMICS in the drop-down list box for pre-assignment of the inputs/outputs.

To switch a single terminal

1. In the project navigator, open the folder inputs/outputs of the relevant drive DOs.
2. Click the tab in the foreground which shows the required inputs/outputs.
3. Click in the BICO field to display the possible interconnection targets.

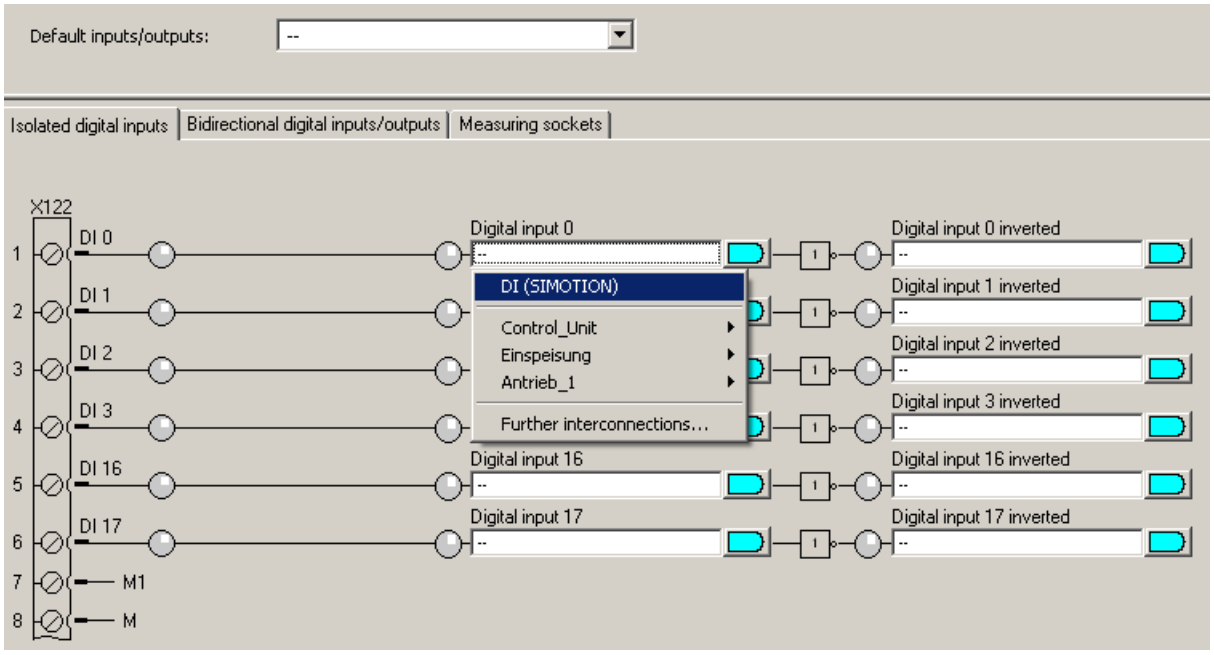


Figure 4-74 Setting the SIMOTION preferential interconnection locally

4. Select, for example, the entry DI (SIMOTION), to assign a digital input.

This is the way to proceed with a symbolic assignment with the SIMOTION preferential interconnection:

1. Activate in the drive DO the SIMOTION preferential interconnection (see above).
2. Open the address list in SCOUT.
3. For the configuration of the I/O variables, enter the identifier IN or OUT (or make a selection from the drop-down list box) under the I/O address for the direction, select the data type, and open the assignment dialog by clicking the " ... " button.

4. The assignment dialog will then only show assignment targets which are suitable for this data type.
   If no data type is selected, every known assignment parameter is listed for selection (e.g. bit, WORD, DWORD, input/output terminals). The assignment dialog returns the data type of the assignment partner when it is closed.
5. Select the appropriate terminal signal and click **OK**.

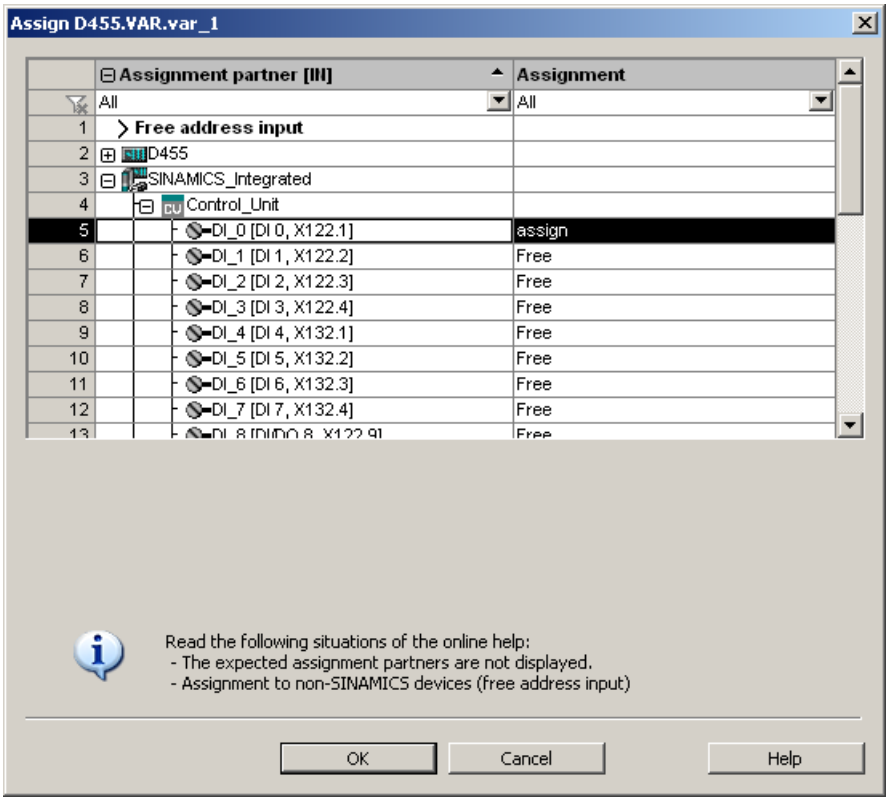


Figure 4-75 Assigning an I/O variable

Note

If the SIMOTION preferential interconnection is not activated, you only see entries like DI_0_15 in the dialog (see below). In this case, only interconnect the I/O variable to one bit in DI_0_15. The BICO interconnections must then be created manually, otherwise the interconnection does not function.

This is the way to proceed with a symbolic assignment of telegrams via PZDs:

1. Open the address list in SCOUT.
2. For the configuration of the I/O variables, enter the identifier IN or OUT (or make a selection from the drop-down list box) under the I/O address for the direction, select the data type, and open the assignment dialog by clicking the " ... " button.

3. Select the interface, to which you wish to assign an I/O variable, e.g. DI_0_15.
   You can also selectively assign the variable to a bit.

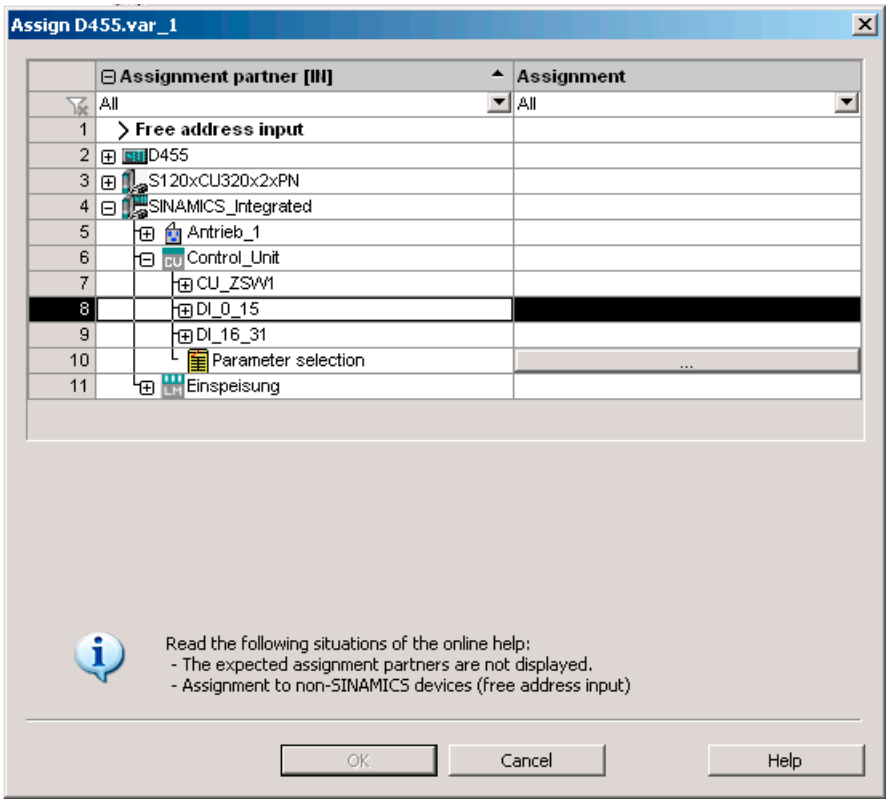


Figure 4-76 Assignment dialog for PZDs

4. Click **OK** to accept the assignment.
5. Now change to the dialog **communication** of the relevant SINAMICS DO. You can now see the individual bits of the PZD (e.g. E_Digital or A_Digital) listed there.
6. Interconnect the corresponding bit of the PZD with a signal, e.g. **External warning active**. This will execute the BICO interconnection.

Note

When you have activated the symbolic assignment, the telegrams 39x will be automatically assigned for the SINAMICS Control Units (at least telegram 390). The following interconnection of the terminals is, however, not executed in line with the original definition of the telegrams (without symbolic interconnection) since the terminal configuration is defined by it with the symbolic interconnection and managed by the symbolic interconnection.

Assignment of I/O variables with substitute values

Substitute values cannot be specified for I/O variables of data type BOOL. If you do need substitute values, however, proceed as follows:

1. Assign a Bool-type variable to, for example, the actuator interface of a drive, Drive_1.Control_Unit.DI_0_15
2. Create a global variable (with WORD as the minimum data type).
3. Assign the actuator interface of the drive to the variable in the assignment dialog, Drive_1.Control_Unit.MELDW.
   Bit1 of the substitute value must then contain the substitute value for the Bool variable.
   You can assign a substitute value to a BICO parameter using the same kind of method.

Additional, higher-level interfaces for assigning substitute values are available for various SINAMICS DOs, see SINAMICS assignment types (Page 1786).

See also

Assigning the TO axis and TO external encoder (Page 1191)

Assigning the drive I/O symbolically (Page 1407)

4.2.4.4 Working with the assignment dialog

Using the assignment dialog

Description

A SIMOTION object can be assigned to a SINAMICS object using the assignment dialog.

You can call the assignment dialog from various technology object configuration dialogs and from the address list. The display of the dialog is dependent on the assignment for which you call it; for example, if an input, an output, or an actuator/encoder is to be assigned.

The analog outputs of a C240 cannot be interconnected symbolically. These must be interconnected using the input of addresses.

Signal or assignment type not found

The assignment dialog only shows assignment types which are suitable for the SIMOTION type. If you were expecting a different assignment type, check:

- The data type of the assignment type (e.g. for I/O variables)
- The configuration of the SINAMICS DO (input/output)
- Whether symbolic assignment is switched on (**Project > Symbolic assignment**)

Structure of the assignment dialog with symbolic assignment activated

The assignment dialog with symbolic assignment activated is available in the axis wizard, the address list, and TO configuration screen forms.

The assignment dialog has the following structure:

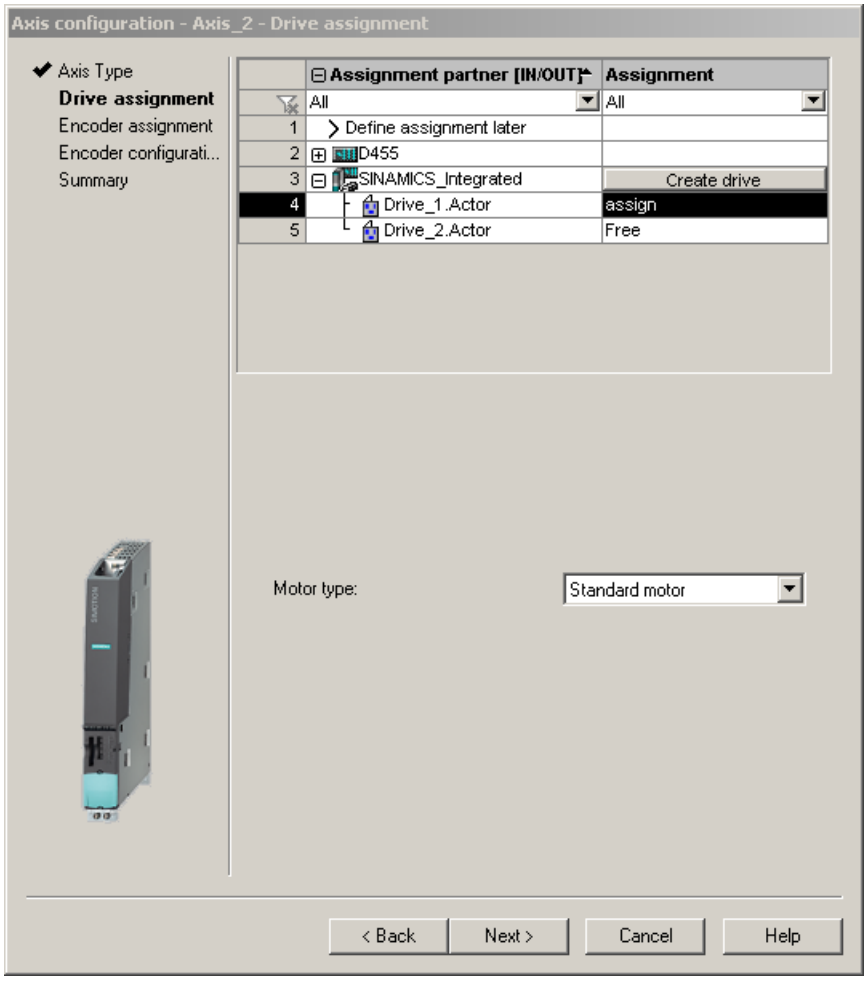


Figure 4-77 Assignment dialog with symbolic assignment activated

Field	Description
Title bar	The title bar contains the name of the SIMOTION interface or I/O variable to be interconnected. The dialog only shows the assignment parameters which are suitable for this interface.
Filter line	The filter line enables you to filter what is displayed in the table according to the criteria **all**, **free**, and **assigned**.
Assignment partner	This column lists all the devices and DOs. Click the "+" sign to expand the groups and display suitable interfaces, e.g. under the control unit. Note: If you assign an I/O variable to an I/O terminal of SINAMICS modules, you must still execute the BICO interconnection in the drive DO when using the interfaces DI_0_15 or DO_0_15 (name can vary depending on the number of terminals), see also Assigning I/O variables via the address list (Page 1202).

Field		Description
	Define assignment later	Select **Assign later** if you want to assign the SIMOTION interface at a later time.
	Free address input	Select this entry if you need/want to enter the address for the assignment partner yourself (e.g. for devices not assigned symbolically, hydraulic axes, or analog modules). You then have to read out the address in HW Config and enter it here.
	Parameter selection	Under Parameter selection you can select parameters of the SINAMICS DO. See also Interconnecting parameters via BICO (Page 1198).
Assignment		This column shows whether the SIMOTION interface is already assigned: • Blank or empty, the interface is still unassigned. • Assign; the interface is assigned to the selected assignment partner. • <assigned interface>; the interface is already assigned. The assigned SIMOTION interface is shown.
Create drive (Axis wizard - Assign drive)		Click the button to call the drive wizard for creating and configuring a SINAMICS drive.
Create encoder (Axis wizard - Assign encoder)		Click on the button to call the wizard for configuring the encoder DO.

Structure of the assignment dialog with symbolic assignment deactivated

The assignment dialog with symbolic assignment deactivated is only available in the axis wizard (drive assignment and encoder assignment).

The assignment dialog has the following structure:

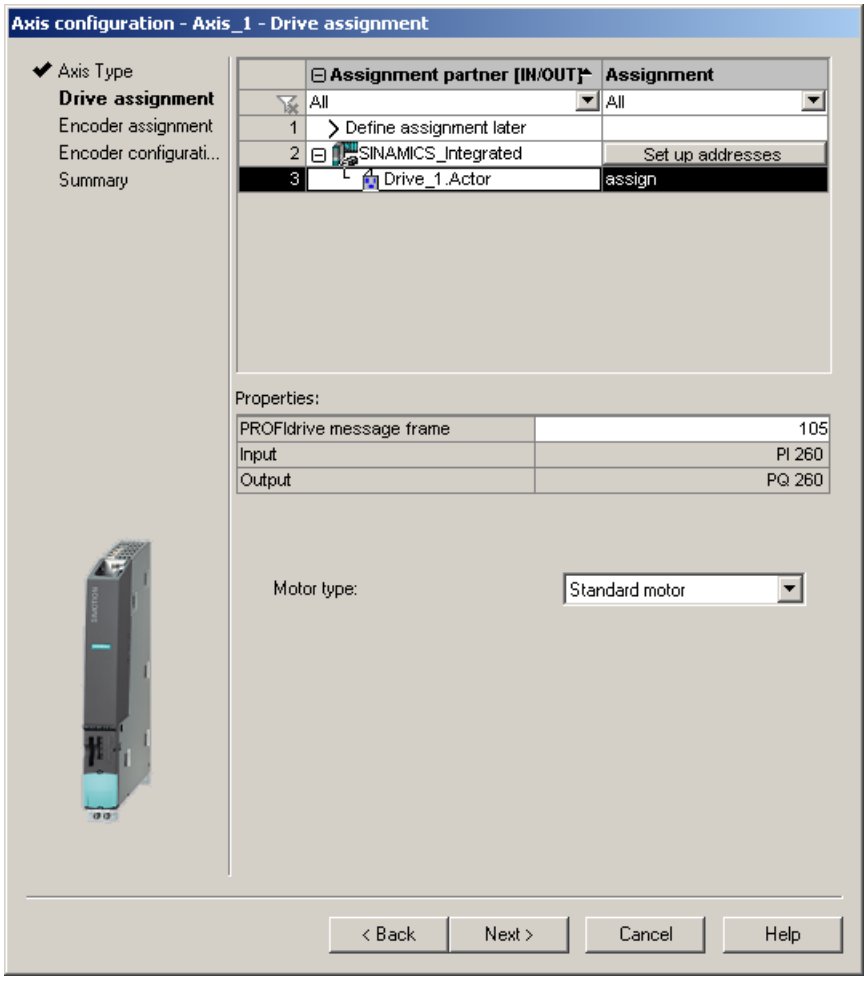


Figure 4-78 Assignment dialog

Field		Description
Title bar		The title bar contains the name of the SIMOTION interface or I/O variable to be interconnected. The dialog only shows the assignment parameters which are suitable for this interface.
Filter line		The filter line enables you to filter what is displayed in the table according to the criteria **all**, **free**, and **assigned**.
Assignment partner		This column lists all the devices and DOs. Click the "+" sign to expand the groups and display suitable interfaces, e.g. under the control unit.
	Free address input	Select this entry if you need/want to enter the address for the assignment partner yourself (e.g. for devices not assigned symbolically or hydraulic axes). You then have to read out the address in HW Config and enter it here.
	Parameter selection	Under Parameter selection you can select parameters of the SINAMICS DO.
Assignment		This column shows whether the SIMOTION interface is already assigned: • Blank or empty, the interface is still unassigned. • Assign; the interface is assigned to the selected assignment partner. • <assigned interface>; the interface is already assigned. The assigned SIMOTION interface is shown.

Field		Description
	Set up address	This button is for transferring the addresses, message frames, etc. of devices that are not symbolically assigned to HW Config.
Properties		In this table you can enter settings for the drive or encoder.

Assignment dialog with symbolic assignment deactivated or for drives which do not support symbolic assignment

The drives are only fully visible once you have performed "Set up addresses" (using HW Config). This affects SIMODRIVE or MASTERDRIVES drives, for example, and the PROFIBUS modules ADI4 or IM174. See also AUTOHOTSPOT.

Free address input

In addition to the symbolic assignment, you can also enter a hardware address directly under "Free address input". This is necessary, for example, if the hardware components do not support symbolic assignment.

The address entry in the text box is checked and depends on the data type

specified when calling the dialog.

- If a data type was not entered when the dialog was called, the correct syntax of the address is checked and the data type is determined on the basis of the address. It is then entered in the address list, for example.
- The address of an output variable cannot be entered for input variables, and vice versa.

4.2.4.5 Setting up addresses and message frames

Setting up addresses and message frames - overview

Overview

After all SINAMICS components have been configured, addresses must be determined for the process data exchange between the drive and the controller.

This procedure depends on whether **symbolic assignments** are used.

- With **symbolic assignment**, the addresses are determined automatically by the engineering system; see the section titled Setting up communication for symbolic assignment (Page 1212).
- Without **symbolic assignment**, the addresses must be determined manually; see the section titled Message frame configuration (Page 1212).

See also

Acyclic communication with the drive (Page 1604)

Setting up communication for symbolic assignment

You can set up communication for symbolic assignment using the following methods:

- Via the SCOUT menu (call the following in the menu: **Project > Set up communication for symbolic assignment**)
- By downloading to the target system
- By saving the project and compiling changes

When you set up communication, the message frames, BICO interconnections, and addresses are set up for the entire project.

See also

Setting up addresses and message frames - overview (Page 1211)

Message frame configuration (Page 1212)

Downgrading to versions <V4.2 (Page 1221)

Message frame configuration

Requirement

You have configured the drive unit.

Note

You only need to carry out these steps if symbolic assignment is not active or cannot be used (as is the case with devices < V4.2, for example).

On the basis of this configuration, one or more of the following actions should be performed:

- The automatic PROFIdrive telegram setting for a drive object should be activated/deactivated
- The automatic telegram extension for a drive object should be activated/deactivated
- The automatic address adaptation for a drive object should be activated/deactivated
- PROFIdrive telegrams should be configured for drive objects
- The addresses should be set up
- Telegrams should be extended manually.

Note

Telegram synchronization

The telegram synchronization process cannot delete any safety slots for which F-proxy is activated. For this reason, telegram synchronization cannot take place.

- Remove F-proxy, perform telegram synchronization and enable F-proxy again.

Procedure

To do so, proceed as follows:

- In the project navigator, open the entry for the SINAMICS drive unit, e.g. an external S120 or a SIMOTION D.
- In the project navigator, open the **Communication > Telegram configuration entry** under **Control Unit** (SINAMICS standalone CU3xx or **SINAMICS_Integrated)**.
  The **Telegram configuration** dialog is displayed on the tab **IF1 PROFIdrive PZD telegrams**.

The dialog lists all the available drive objects. The possible setting options are described in the following.

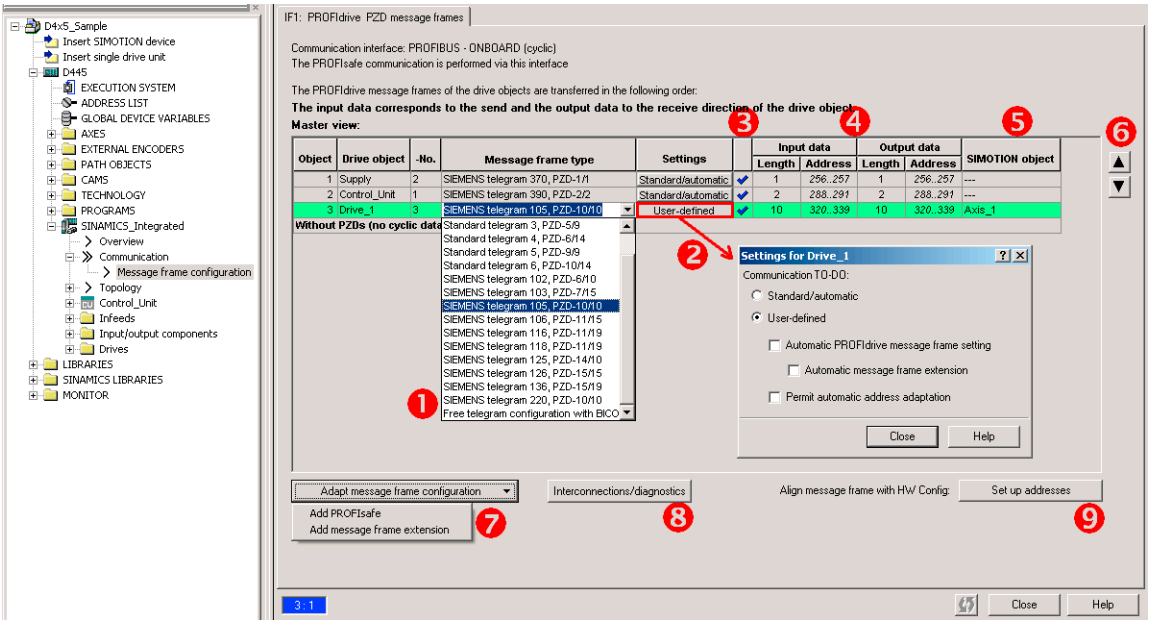


Figure 4-79 Telegram configuration

Table 4-23 Explanation of the figure

Number	Meaning
1	Selection of a telegram • The drive telegrams (telegrams 1 ... 6 and telegram 1xx) are defined in accordance with the PROFIdrive specification and can be selected based on the required functional scope. • You can transfer the signals of the I/Os or the global measuring inputs for the Control Unit, for example, via the telegrams 39x. Telegram 39x is also required for the time-of-day synchronization between SIMOTION and SINAMICS. • Free telegram configuration with BICO allows you to define your own telegram. • Free telegram configuration with p915/p916 (for TM15/17). • Telegrams 37x for control of the infeed.
2	The setting "Standard/automatic" and "User-defined" is visible only if "Use symbolic assignment" is activated. It is generally recommended to use the setting "Standard/automatic". With the "User-defined" setting, the automatic telegram setting, telegram extension and address adjustment can be deactivated and/or activated. • With "Automatic PROFIdrive telegram setting", the telegram is set by the system depending on the configured technology (telegram selection, e.g. for infeed, drive and Control Unit incl. onboard I/O). • With "Automatic telegram extension", the telegram is extended by the system depending on the configured technology (e.g. when the technology data block is activated in the axis configuration). • With "Allow automatic address adjustment", the addresses can be adapted for address displacements, for example, by the system. Address displacements can occur, for example, when a telegram is extended and the adjacent addresses are already occupied by other telegrams. For the TM15 / TM17 High Feature, deactivation of the "Automatic PROFIdrive telegram setting", the "Automatic telegram extension" and the "Automatic address adjustment" is in principle not possible since with these drive objects the telegram is always set up according to the parameterized terminal functionality (DI, DO, output cams, measuring inputs) and cannot be extended. In case the telegrams are to be manually configured for TM15 DI/DO, TM31 and TB30, and BICO is to be interconnected, then "Automatic PROFIdrive telegram setting" and "Automatic telegram extension" must be deactivated. See also Performing message frame configuration in the Online Help.
3	The icons in the status column show the following information:
	 The telegram is configured differently in HW Config. You must align the configuration with HW Config.
	 You are using a predefined standard telegram or free BICO interconnection.
	 You are using an altered standard telegram, which you have extended to include supplementary data.
	 You are using a telegram for which one of the two telegram lengths (I or O) is too long. The drive object cannot process this entry.
4	Length: Displays the size of the telegram component. Address: Address range in HW Config The addresses will be displayed only if they have been defined.
5	Displays the SIMOTION object that is interconnected to the SINAMICS object (e.g. axis or encoder).
6	Change the telegram order. Before the alignment, all drive objects without input/output addresses ("---..---") must be moved behind the objects with valid input/output addresses or those still to be aligned ("???..???").
7	"Manual" adaptation of the telegram configuration (e.g. when additional data, such as a motor temperature, is to be transferred via the telegram)

Number	Meaning
8	Display of the individual control and status words of the associated telegram.
9	Setting up of the addresses (alignment of the addresses with HW Config)

Note

If the telegrams for drive objects change (drives, Terminal Modules, etc.), you must set up the addresses again. The addresses are not updated automatically.

Error correction (symbolic assignment deactivated)

SIMOTION generates further configuration information (FastIO configuration) for the following functions using the 39x telegram:

- Time-of-day synchronization SIMOTION↔ SINAMICS
- Use of onboard I/Os of SIMOTION CU or CX
- Use of output cams and global measuring inputs
- _setDriveObjectSTW system function

If the telegrams are manually defined (symbolic assignment is deactivated), a 39x telegram must be created in the telegram configuration. The telegram must then be compared to the HW Config via "Set up addresses".

If the above functions cannot be used, create a new FastIO configuration. To do this, select the associated SIMOTION CPU in the project tree and right-click to open the "Fast IO" > "Create new configuration" context menu. Compile the project and load it into the CPU. Restart the computer.

The FastIO configuration is also used for the telegram of the Terminals Modules TM15 and TM17 High Feature. Follow the same procedure in the event of problems.

See also

Setting up addresses and message frames - overview (Page 1211)

Setting up communication for symbolic assignment (Page 1212)

Assigning the TO axis and TO external encoder (Page 1191)

4.2.4.6 Switching over projects to symbolic assignment

Working with and without symbolic assignment

Description

The symbolic assignment simplifies the configuration of the technological relationships including the communication between controller and drive significantly. The names of the symbolic assignments in plain text is also advantageous for project maintenance.

The symbolic assignment is therefore recommended for new projects as of V4.2 and is automatically activated.

If symbolic assignment is used in a project, telegrams, interconnections and addresses are automatically created by the engineering system per default.

The engineering system sets the "optimum" PROFIdrive telegrams and telegram extensions for the system, makes the required BICO interconnections and determines the addresses.

Activating the symbolic assignment later

It is also possible to change updated projects to symbolic assignment. This must be decided individually, as follow-up work may be required. Particularly in the case of free telegram configurations, comprehensive follow-up work is to be expected.

NOTICE
Possible changes in telegrams and BICO interconnections If symbolic assignment is subsequently activated for a project in which telegrams have already been configured and interconnected, these can be changed together with the BICO interconnections! For this reason, make a backup copy of your project before activating the symbolic assignment.

If symbolic assignment is subsequently activated and telegrams have already been configured and interconnected in the project, the engineering system attempts to identify symbolic assignment for this configuration.

If a symbolic assignment was able to be identified for all communication connections, when the symbolic assignment is activated for the first time, the automatic telegram determination/ extension and address adaptation must then be activated for all drive objects (DOs).

It is not possible to determine a symbolic assignment for all communication connections. This is signaled via a dialog.

The corresponding warnings are then issued in the "Set up assignments" dialog.

Should such a case arise, one of the following measures is required PRIOR TO COMPILATION:

- Option 1: Deactivate automatic for drive object
  In the "Communication" > "Telegram Configuration" drive dialog, the automatic telegram determination/extension and address adaptation for the affected drive objects must be deactivated (user-defined setting, all check marks deselected).
  When the symbolic assignment is activated for the first time, the automatic telegram determination/extension and address adaptation is deactivated for all drive objects (DOs) as standard.
- Option 2: Reconfigure assignments
  The symbolic assignments must be made via the corresponding assignment dialogs of the TO configuration or the address list

Basic information

If necessary, the degree of reconfigurations required is thereby determined, as well as how well "already defined telegram/interconnections" correspond to the settings that the engineering system makes for optimization.

Given the individuality of the "Free telegram configuration with BICO" and the "Telegram extension", it is particularly important to take into account the fact that a symbolic assignment cannot always be determined.

As there is no standard telegram for TB30, TM15 DI/DO and TM31, and "Free telegram configuration with BICO" or "Telegram extension" is thereby applied, these components are particularly affected.

The same applies to the onboard I/Os of a Control Unit (D4x5-2, CX32-2, CU320-2, etc.), if the "Free telegram configuration with BICO" was used here.

Example 1:

If, for example, onboard I/Os of a Control Unit are connected via "Free telegram configuration with BICO", the telegram length, telegram format and interconnection are very individual.

If symbolic assignment is now activated, the image is usually generated on the basis of standard telegram 39x, whereby the telegram length, telegram format and interconnection change.

This then causes disturbances, for example, to the addressing of an onboard I/O that is to be used by SIMOTION.

Example 2:

Telegram 106 is configured for a drive (transfers two encoder values). Only one encoder is used, however.

If symbolic assignment is now activated, optimization of telegram 105 automatically occurs (transfers one encoder value).

NOTICE
Possible changes in telegrams and BICO interconnections
If symbolic assignment is subsequently activated for a project in which telegrams have already been configured and interconnected, these can be changed together with the BICO interconnections!
For this reason, make a backup copy of your project before activating the symbolic assignment.
TB30, TM15 DI/DO and TM31 are especially affected.

See also

Upgrading projects <V4.2 (Page 1219)

Using symbolic assignment

Description

Symbolic assignment of SIMOTION and SINAMICS interfaces can be switched on or off as follows. As of V4.2, symbolic assignment is activated by default when new projects are created.

Where projects < V4.2 are being upgraded, symbolic assignment is deactivated by default and needs to be activated if required.

Activating symbolic assignment

1. If the menu entry **Project>Use symbolic assignment** is checked, the function is already activated. If it is not checked, select **Project>Use symbolic assignment**.
   A message dialog opens.
2. Click **OK** to confirm.
   Symbolic assignment is activated and the necessary steps are executed:
   - Assignments between the objects are determined and created symbolically based on the logical addresses
   - Addresses are set up (at the latest automatically before the download)

Note

If you are using symbolic assignment, SCOUT is entirely responsible for managing the telegram between SIMOTION and the drive DO. In other words, SCOUT independently creates telegrams that all contain the necessary signals according to the technological configuration. SCOUT automatically manages the positioning of signals in the telegram and its size; the user is no longer able to influence this process or make any changes.

If you have upgraded a project to V4.2 or started a project initially without symbolic assignment and then select symbolic assignment later on, SCOUT automatically extracts the signal objects contained in the existing telegram configuration and uses them to generate new telegrams automatically based on the pattern you defined. As a specific consequence of this, the number and size of telegram modules (PROFIBUS) or telegram submodules (PROFINET) are bound to change if symbolic assignment is selected (e.g. the number of telegram modules/submodules may be reduced in order to optimize performance).

Deactivating symbolic assignment

The menu item **Project > Use symbolic assignment** is checked if the function is activated.

1. Perform **Project > Use symbolic assignment**.
   A message dialog is displayed indicating that addresses and symbolic names will be set up and checked accordingly.
2. Click **No** to leave symbolic assignment activated.
3. Click **Yes** to deactivate symbolic assignment.

The logical addresses are reassigned priority for the connections between the objects. If this process results in invalid addresses, a message dialog brings this to your attention.

Upgrading projects <V4.2

Description

Should you wish to upgrade projects with a version <V4.2 in order to work with symbolic assignment, you must carry out the following steps where necessary. See also Working with and without symbolic assignment (Page 1216).

NOTICE
Possible changes in telegrams and BICO interconnections
If symbolic assignment is subsequently activated for a project in which telegrams have already been configured and interconnected, these can be changed together with the BICO interconnections!
For this reason, make a backup copy of your project before activating the symbolic assignment.
TB30, TM15 DI/DO and TM31 are especially affected.

Procedure

- Open project in SCOUT: The project is opened with deactivated symbolic assignment. In the **Telegram configuration** dialog of the corresponding SINAMICS Control Unit, the dialogs for the symbolic assignment telegram settings are deactivated. As a result, the telegrams, addresses and BICO interconnections that have already been configured are retained.
- Upgrading device version to ≥ V4.2. Only devices as of version V4.2 support symbolic assignment. Devices can be upgraded in HW Config, for example.
- Activating symbolic assignment: When symbolic assignment is activated, the assignments are introduced between axes and drives and are first updated when the project is saved and compiled. The addresses and telegram configuration of SINAMICS DOs continue to remain intact until the automatic telegram configuration is activated in the **Telegram configuration** dialog. In the **Set up assignments** tab, error messages that occur during assignment are output. Using these error messages, you can check and, if necessary, change the TO-DO interconnections.

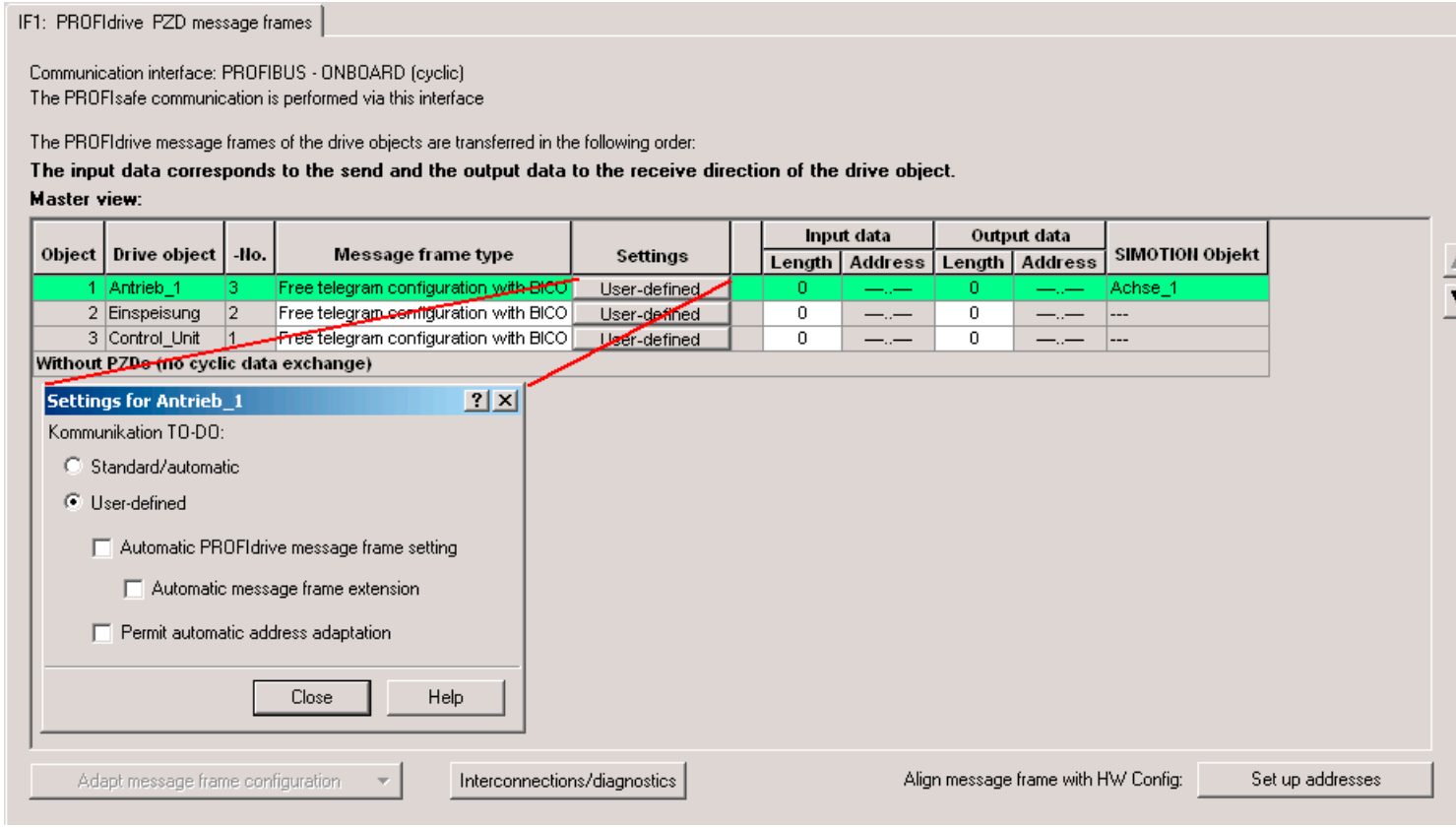


Figure 4-80 Telegram configuration according to high converting

- Assigning SINAMICS DOs: In order to ensure that the SINAMICS DOs are able to participate in the symbolic assignment, the following steps must be carried out in the **Telegram configuration** dialog for each DO:
  - Call dialog **Setting for <DO>** and select the **Standard/Automatic** option when you want to automatically execute the telegram configuration.
    Or
    Call dialog **Setting for <DO>** and select the option **User-defined** if you wish to execute the telegram configuration user-defined, e.g. if you wish to use the setting **Free telegram configuration via BICO** (deselct **Automatic PROFIdrive telegram setting**).
- Setting up communication for symbolic assignment: When you have checked all interconnections, you can run **Project>Setting up communication for symbolic assignment**. All telegrams, addresses and BICO interconnections are then updated via symbolic assignment.

Downgrading to versions <V4.2

Description

If a SIMOTION device is downgraded by replacing a device in HW Config, you must take into account the fact that symbolic assignments are only available as of version V4.2.

If symbolic assignments are used and downgrading to a SIMOTION device <V4.2 occurs, the "Communication for symbolic assignment" must be set up before replacing the device in HW Config. As a result of this, the addresses that are required for a project "without symbolic assignment" are identified.

See also Setting up communication for symbolic assignment (Page 1212) for more information.

Safety data block for a SINAMICS device version less than V4.x

Description

Problems occur with older SIMOTION modules and SINAMICS CUs that have a firmware version less than V4.x in the following combination:

- Used STANDARD telegram 4/6 or SIEMENS telegram 103/106 (each with and without a technology data block)
- Safety data block
- Servo drive
- Double-encoder system

Occurring problems

Upgraded projects:

The following message is output during subsequent activation of the symbolic interconnection for upgraded projects:

"Address/value of I/O variable 'Axis_x: TypeOfAxis.TechnologicalData.DriveSafetyExtendedFunctionsInfoDataIn.logAdress' (address: PIB xxx[x]) is invalid or is not supported by the hardware. (If required, perform FastIO "Create new configuration".)"

Newly created projects:

With newly created projects (and therefore activated symbolic assignment), the SIDB cannot be created by the system because the limited velocity must be interconnected to a double word.

Message output at activation of SINAMICS Safety Integrated Extended Functions on the TO axis: "SINAMICS_Integrated: Drive_x: Not enough space to set up the telegram extension".

Procedure

Upgraded projects:

In the "Telegram configuration", do not set the settings to "Standard/automatic" and deactivate the automatic PROFIdrive telegram setting and the automatic telegram extension, or make the appropriate settings.

If the system has deleted the telegram extension due to the "Standard/automatic" setting, make the settings as described above and interconnect the safety data block again (parameter 9733[0] to the first free word after the safety data block and leave the following word empty).

Newly created projects:

Deselect "Standard/automatic" in the telegram configuration settings and deactivate the automatic PROFIdrive telegram setting and the automatic telegram extension. Interconnect parameter 9733[0] to the first free word after the safety data block and leave the following word empty.

With upgraded and new projects:

Go to the address list and set the view to "All addresses". At the appropriate axis ("Axis name".Actor.SIDB), enter the address PIxxx at which the safety data block is located. For example, SIEMENS telegram 106 has 15 PZD and therefore the following applies:

Logical start address of the drive +30 / when using the technology data block +32

For further information, see also AUTOHOTSPOT in the TO Axis Function Manual.

4.2.5 Programming with Technology Objects

4.2.5.1 Definitions

The following section provides an introduction to the motion components primarily from the perspective of the ST programming language. This mainly involves system functions, system variables and configuration data. Some definitions are provided below:

- System functions are functions used for the system management. They provide access to technology-neutral functionality of the device. System functions are always available.
- A technology object (TO) represents a technological functionality (for example, positioning an axis, assigning parameters for an output cam) in the SIMOTION user program.
- Technology object functions or technology commands are language commands provided by the individual technology objects, i.e. functions for a technology object.
- A technology package contains one or more technology object types, from which the respective TO instance is generated with <Insert technology object>.
- System variables and configuration data are attributes of the technology objects and the basic system. You can use system variables to assign parameters for technology objects and the basic system or to read their status.

Note

You will find additional information about the basics, configuration and programming of motion control technology and, in particular, technology objects, in the SIMOTION Motion Control <Technology Objects> Function Manuals.

The specified topics are only briefly described in this documentation.

4.2.5.2 Programming technology objects (TOs)

Using technology functions in a program

In order to use technology functions (TO functions), you must select a technology package once. Technology objects (TOs) and, thus, TO functions are contained in the technology package. Formally, these are functions with a function name, input parameters and, usually, a return value. Below is an overview of the codes and components of the technology objects and TO functions except for the input parameters, which are described in the **Function parameters of the technology functions** section.

The following is presented from the ST programming perspective. For LAD/FBD and MCC programming see the respective manuals.

Selecting a technology package

You use the following command to select the technology package:

`USEPACKAGE` *tp-name*

Here *tp-name* is the name of the technology package (see next table).

Table 4-24 Technology packages

Technology package	Description of contents
CAM	• Basic motion control commands, positioning, gearing and discontinuous synchronous operation (cam) • Commands for external encoders, measuring inputs, output cams, and cam tracks
PATH[1]	As for TP CAM plus commands for path interpolation
CAM_EXT	As for TP PATH plus commands for supplementary technology objects (fixed gear, addition object, formula object, sensor, controller object)
TControl	Commands for temperature controllers

[1] Available as of version V4.1.

Technology object (TO) codes

A technology package provides technology objects (TOs); they are instantiated in the SIMOTION SCOUT. An instantiated TO is addressed (referenced) in the program by means of its name.

Codes of technology functions (TO functions)

TO functions are primarily commands that execute specific actions on the technology object, which is why they are also referred to as TO commands. You can also use TO functions in user-defined FBs. For more information about the format rules of user-defined FCs and FBs, please refer to the SIMOTION ST Programming Manual. The following table shows an overview of the formal codes of the TO functions.

Table 4-25 Codes of the TO functions

Code	Description
Name	All TO function names (*_enableAxis* in the example) are defined identifiers in the SIMOTION system that always begin with _ (underscore). To maintain a distinction between these TO functions and user-defined FBs and FCs, you should not create source file sections beginning with the _ character.
Input parameter	When called, TO functions can contain one or more input parameters and always supply a return value to the call location. Output parameters are not supported. For additional information, see Input parameters (Page 1574).
Return value	All commands normally return a double-precision integer (DINT data type). This indicates whether the command was successfully transferred to the system and processed normally (return value of zero) or whether an error occurred (return value other than zero).

Example

If the application requires you to enable a virtual axis, you could program a source file like the one shown in the following figure.

The following conditions must be fulfilled:

- An instance of a position axis or speed axis has been created in SIMOTION SCOUT as a virtual axis called Axis_1.
- The *myPos* program has been assigned to *MotionTask_1*, for example. The **Activation after StartupTask** option has been selected in the task configuration of *MotionTask_1*.
- The source file has been downloaded to the target system.

After the CPU has changed to RUN mode, virtual axis *Axis_1* will issue an enable. You can verify the status of the axis enable in system variable *Axis_1.control*.

Table 4-26 Example for using TO functions in a program

```
INTERFACE
    USEPACKAGE CAM;
    PROGRAM myPos;
END_INTERFACE
IMPLEMENTATION
(* The following program must be assigned to a MotionTask. The "Activation
after StartupTask" option must be selected in the task configuration. *)
    PROGRAM myPos
        VAR
            retVal : DINT;
        END_VAR
        // Axis is enabled for positioning.
        retVal := _enableAxis (
            axis := Achse_1,
            // TO instance identifier
            nextCommand := WHEN_COMMAND_DONE,
            // Condition for program advance.
            commandId := _getCommandId() );
            // Unique command ID
    END_PROGRAM
END_IMPLEMENTATION
```

Note

Use the long form for passing parameters (with value assignment) described in **Function parameters of the technology functions**. This form is clearer and more flexible. For example, the program code therefore remains compatible with new function parameters for a function extension.

You will find tips on efficient use of parameters in system functions in **Error sources and efficient programming**.

See also

Function parameters of the technology functions (Page 1228)

Sample program with namespace option

You can use the optional *namespace* add-on AS to define a name space. You can then also access types, variables, functions and function blocks in the technology package that have the same name as those in the ST source file.

The following example shows how to select the CAM technology package, assign it the namespace Cam1, and use the namespace:

Table 4-27 Example of selecting a technology package and using a namespace

```
INTERFACE
    USEPACKAGE CAM AS Cam1;
    USES ST_2;
    FUNCTION function1;
END_INTERFACE

IMPLEMENTATION
    FUNCTION function1 : VOID
    VAR_INPUT
        p_Axis : posAxis;
    END_VAR
    VAR
        retVal : DINT;
    END_VAR

    retVal:= Cam1._enableAxis (
            axis := p_Axis,
            nextCommand := Cam1.WHEN_COMMAND_DONE,
            commandId := _getCommandId() );
    END_FUNCTION
END_IMPLEMENTATION
```

Note

If a namespace is defined for an imported library or technology package, this must **always** be specified if a function, function block or data type from this library or technology package is being used. See above example: Cam1._enableAxis, Cam1.WHEN_COMMAND_DONE.

See also

Function parameters of the technology functions (Page 1228)

Efficient programming - overview (Page 1686)

Differences between cyclical and sequential programming

Cyclic tasks

Cyclic tasks (such as the BackgroundTask) will be started by the system after their completion or automatically after a defined time frame. The values of the static variables of the assigned programs are retained. Cyclic tasks have a time monitoring and a defined error response should the time monitoring be exceeded. This means the code contained in cyclic tasks must perform its tasks quickly and efficiently. Tasks with a waiting character (for example, wait for the positioning of an axis) can only be performed in several call cycles of the cyclic task. This means TO system commands must normally be called with the IMMEDIATELY step enabling condition for the nextCommand parameter. The subsequent call cycles must then check the result of the system command for successful processing or error. This procedure is also called asynchronous execution.

Sequential tasks

After start, sequential tasks (e.g. MotionTasks) are executed once and then terminated. At each start, all local variables of the assigned programs are initialized; see Influence of the compiler on variable initialization (Page 1419) for variable initialization. Because sequential tasks do not have any time monitoring, they can attain a run time of any length. Sequential tasks are subject only to the control of the application. This means the application can start, stop, pause and resume tasks. The code contained in sequential tasks processes tasks successively, where the following task is normally performed only when the previous task has been completed. For example, an axis is first released and then homed. This means TO system command calls should be performed using the WHEN_COMMAND_DONE step enabling condition for the nextCommand parameter. The system function returns to the calling sequential task only when the command has been processed. This is also called synchronous execution.

General Procedure

In general, it is better to program sequential sequences in MotionTasks. The differences between a sequential programming in a MotionTask and the cyclical programming in the BackgroundTask are:

- As part of the sequential processing of a the MotionTask, one and only one (but nevertheless linked) step enabling condition can be awaited at any point in time. The step enabling condition is checked with high priority in the interpolator cycle clock. To reduce the response time for continuing the MotionTask, its priority can be increased temporarily (-> WAITFORCONDITION).
- A normally cyclical program checks in each cycle a number of signal states and step enabling conditions. This heavily loads and extends the cycle time. The advantage compared with the sequential type of programming lies in the parallel processing of requests and sequences.

Synchronous and asynchronous command execution

Synchronous command execution:

Further processing of the program is suspended at the command until the command has been fully executed (preferably in synchronous tasks).

Asynchronous execution:

The function is activated via a command and program processing continues (preferably in cyclic tasks) while the function is still being executed by the system.

For more information, please read Transition and step enabling condition (Page 1233).

Function parameters of the technology functions

The function parameters of the TO functions are:

- *mandatory*, i.e. they must be specified, for example, the TO instance and the target position for positioning;
- *optional predefined*, i.e. they can be specified; otherwise, a default setting for the system is activated, for example, IMMEDIATELY for the step enabling condition;
- *optionalUserDefault*, i.e. they can be specified, but if they are not, a user-defined default behavior takes effect (programmable or configurable in the associated userDefault variable). For information about the use of parameter values in motion control programs, refer to the documentation for the technology objects (TOs).

The TO instance name must always be specified for the TO commands as they cannot be preset by the system. In our **Example of variables to be specified**, you must specify which axis is to be activated (axis := myaxis).

Short and long form of the function parameter specification

The function parameters can be specified with or without function parameter identifiers.

The following variants are supported (see IEC1131/3 2nd Edition, 11/98):

- **Short form**
  "Call by value" as call with fixed arrangement and number of input variables corresponding to the function declaration
- **Long form**
  "Call by name" as call with variable arrangement and number of input variables

Short form

The function parameters (parameter identifiers) are omitted and only the current data values (parameter values) are specified. An assignment operator is not permitted.

All parameter values (even optional ones) must be separated by commas *in the correct order*!

Only a few system functions (see **Functions for the runtime measurement of tasks**, **Task control commands** and **Functions for the message configuration**) require the short form of parameter transfer when called. This is specified explicitly for the relevant functions.

Long form

The transfer parameters are assigned to the formal operands. Transfer parameters with assigned default values can be specified optionally, but this is not essential.

Note

Use the described long form (with value assignment). This form is clearer and more flexible. For example, the program code therefore remains compatible with new function parameters for a function extension.

Mandatory rules for specifying function parameters

The following table shows the rules you must follow when specifying function parameters.

Table 4-28 Rules for specifying function parameters

Rule	Example
Short form	
All function parameters must be transferred.	In the **Example of variables to be specified** the actual values *myAxis* (variable) and *IMMEDIATELY* (value) are assigned to the function parameters *axis* and *nextCommand*.
The order of the function parameters according to the function declaration must be maintained.	-
Long form	
Only function parameters that are not optional must be specified. The default value is used for parameters that are not specified.	In the **Example of variables to be specified**, the following is also possible: *_enableAxis (axis:=myAxis)*
The order of the function parameters is optional.	In the **Example of variables to be specified**, the following is also possible: *_enableAxis (nextCommand:= IMMEDIATELY, axis:=myAxis)*
Several function parameters	
Several function parameters are separated by commas.	In the **Example of variables to be specified**: *_enableAxis (axis:=myAxis, nextCommand := IMMEDIATELY)*
Actual values as variables	
The advantage is that you can reuse the source file sections that use these variables. For example, a user-defined function block (FB) is to call a TO function with variable axis identifiers as function parameters. It would be awkward to have to call the function several times using constant axis names and would also mean that you could not reuse the FB. You can also use other actual values as variables.	In the **Example of variables to be specified**, the user-defined function block *posFB* defines the TO instance *myAxis* (*myAxis:PosAxis*). When you call a TO function from this FB, the TO instance is used as actual values (*Axis := myAxis*). The values for the TO instance are derived not from the user-defined FB, but from the program called by this FB with *myFB (myAxis := Axis)*. For the reasons named, you can also call the FB and, thus, the TO function with *myAxis := Axis2* or *myAxis := Axis3*, etc. This means you can use the FB in all programs of the ST source file and in programs of other source files with the aid of the export function.
Actual values as values (enumerators)	

Rule	Example
If you enter actual values as values (*next Command: = IMMEDIATELY* in the example), you must usually choose a value from various specific states (only for enumerators). Enumerators are a derived data type. You will find basic information about enumerators in Chapter "User-defined data types" in the ST Programming Manual. You can also specify direct values for other data types.	In the **Example of variables to be specified**, enumerators IMMEDIATELY and WHEN_COMMAND_DONE are available for function parameter *nextCommand* in TO function *_enableAxis*. The compiler rejects other values during compilation of the program.
Parameters for the transition and step enabling condition	
With many motion commands, you can specify when the TO function is executed on the technology object and when the next statement is processed (see **Transition and step enabling condition**)	-
Parameters for command identification	
All motion commands must contain a parameter for the command identification. This allows you to track the status of the command execution (see **Diagnosis of the command execution**).	-

Note

All system data types for enumerations (in system functions and system variables) begin with the word *Enum*. For example, function parameter *nextCommand* has enumeration data type *EnumNextCommandEnable* (with values *IMMEDIATELY* or *WHEN_COMMAND_DONE*).

All system data types for structures (in system functions and system variables) begin with the word *Struct*.

If you do not begin user-defined data types with character string *Enum* or *Struct*, names cannot overlap. For a detailed description of of the name spaces, see ST Programming Manual.

Appendix D of this manual contains all reserved identifiers of the ST (Structured Text) programming language, the system functions, the system blocks and the SIMOTION devices.

The reserved identifiers of the SIMOTION technology packages can be found in the List Manuals for the SIMOTION technology packages.

Reference and value specification for motion variables

The parameters of motion variables (e.g. velocity, acceleration) are defined using a reference parameter and a value parameter.

- The reference parameter specifies which system variable the motion variable to be transferred references and, if required, how the following value parameter is to be interpreted.
  The identifier of a reference parameter is formed from the identifier of the motion variable with the suffix *Type*, e.g. *velocityType*.
- The value parameter specifies:
  - For reference parameter = DIRECT: The numerical value of the motion variable.
  - For reference parameter = USER_DEFAULT: The scaling factor of the default value stored in the system variable.

  The value parameter has no significance for other reference parameter settings.
  The identifier of a value parameter is the identifier of the motion variable, e.g. *velocity*.

Table 4-29 Frequent reference parameters and effect of the associated value parameters

Reference parameter	Value parameter	Value of the motion variable
DIRECT	Absolute value	Content of the value parameter (with the unit of the motion variable specified during the configuration of the technology object). See example in the table below.
USER_DEFAULT	Relative value [%]	Default setting * value parameter / 100 The default settings for the motion variable are stored in a system variable. See example in the table below.
ACTUAL	–[1]	Current actual value
CURRENT	–[1]	Current setpoint of the interpolator
EFFECTIVE	–[1]	Last programmed value
[1] Value parameter is ignored		

Table 4-30 Example of velocityType (reference parameter) and velocity (value parameter)

velocityType	velocity	Value of the motion variable
DIRECT	(No details)	100 [configured unit]
	50.0	50 [configured unit]
	200.0	200 [configured unit]
USER-DEFAULT	(No details)	100% of the system variable value[1]
	50.0	50% of the system variable value[1]
	200.0	200% of the system variable value[1]
[1] For motion commands of an axis: system variable *userDefault.velocity*		

Data type of TO instance variables

You can use TO instance variables, for example, for object specification in TO function calls.

Note

New TO data types (as of V4.0) begin with a "_" to prevent confusion with user variables.

You must first declare the variables, however, and you must choose from a predefined selection of data types (see table for **Data types for technology objects**). In our **Use of TO functions in the program** examples in this section (**Example of variables to be specified**), the data type is *PosAxis*, because you want to create TO instance variables for position axes. The *Cam* (*discontinuous synchronous operation*) technology package you have selected contains the *PosAxis* data type for the *Position Axis* technology object.

Table 4-31 Data types of technology objects (names of technology objects)

Technology object	Data type	Contained in the technology package
Drive axis	DriveAxis	CAM, PATH[1], CAM_EXT
External encoder	ExternalEncoderType	CAM, PATH[1], CAM_EXT
Measuring input	MeasuringInputType	CAM, PATH[1], CAM_EXT
Output cam	OutputCamType	CAM, PATH[1], CAM_EXT
Cam track	_CamTrackType	CAM, PATH[1], CAM_EXT
Position axis	PosAxis	CAM, PATH[1], CAM_EXT
Following axis	FollowingAxis	CAM, PATH[1], CAM_EXT
Following object	FollowingObjectType	CAM, PATH[1], CAM_EXT
Cam	CamType	CAM, PATH[1], CAM_EXT
Path axis[1]	_PathAxis	PATH[1], CAM_EXT
Path object[1]	_Pathobjecttype	PATH[1], CAM_EXT
Fixed gear	_FixedGearType	CAM_EXT
Addition object	_AdditionObjectType	CAM_EXT
Formula object	_FormulaObjectType_	CAM_EXT
Sensor	_SensorType	CAM_EXT
Controller object	_ControllerObjectType_	CAM_EXT
Temperature channel	TemperatureControllerType	TControl
General data type, to which every TO can be assigned	ANYOBJECT	

As of Version V4.1

Variables of the technology object (TO) data type are initialized by default with the value TO#NIL. You can use this to check whether a valid TO was assigned to a variable; see **Example of variables to be specified**.

Example of variables to be specified

An example of variables to be specified with a technology object data type (an example of optional use was described in **Data types for technology objects**) follows. A reusable FB will be written, which will enable any axis. Since the axis name is variable, you must define a variable with the data type of the axis.

The axis to be enabled is provided when the FB is called. This is verified for safety reasons. If no TO is available (TO#NIL), execution of the FB is interrupted.

Table 4-32 Example of variables to be specified with a technology object data type

```
// ...
FUNCTION_BLOCK posFB
    VAR_INPUT
        myAxis: posAxis;
    END_VAR

    VAR_OUTPUT
        // Return value of the TO function,
        // also output parameter of the FB
        return_value: DINT := -1;
    END_VAR
    // Check for valid TO
    IF myAxis = TO#NIL THEN RETURN; END_IF;
        // Example of call with variable of data type of TO
        return_value := _enableAxis (
        axis:= myAxis, // TO function
        nextCommand:= IMMEDIATELY, //optional
        commandId    := _getCommandId() );
END_FUNCTION_BLOCK
PROGRAM Example
    VAR
        myFB: posFB;
    END_VAR
    myFB (myaxis := Axis1);
    // Name is created on start-up // in the SIMOTION SCOUT.
    myFB (myaxis := Axis2);
    // Name is created on start-up // in the SIMOTION SCOUT.

END PROGRAM
//...
```

Transition and step enabling conditions

Fundamentals for the processing of a TO function (command execution)

The TO functions are transferred as commands to the technology function for execution. The TO executes or activates these commands in the processing cycle clock, which was specified during their configuration (e.g. IPO cycle clock).

The outputCam, measuringInput, externalEncoder and cam technology objects have a direct command execution. A new command on the same technology object supersedes a command which is active there.

In addition to commands for direct command execution, motion commands can be issued on the following technology objects: driveAxis, posAxis, followingAxis, and followingObject. The technology objects have structure elements for command management.

An example of the command management for axes is described below.

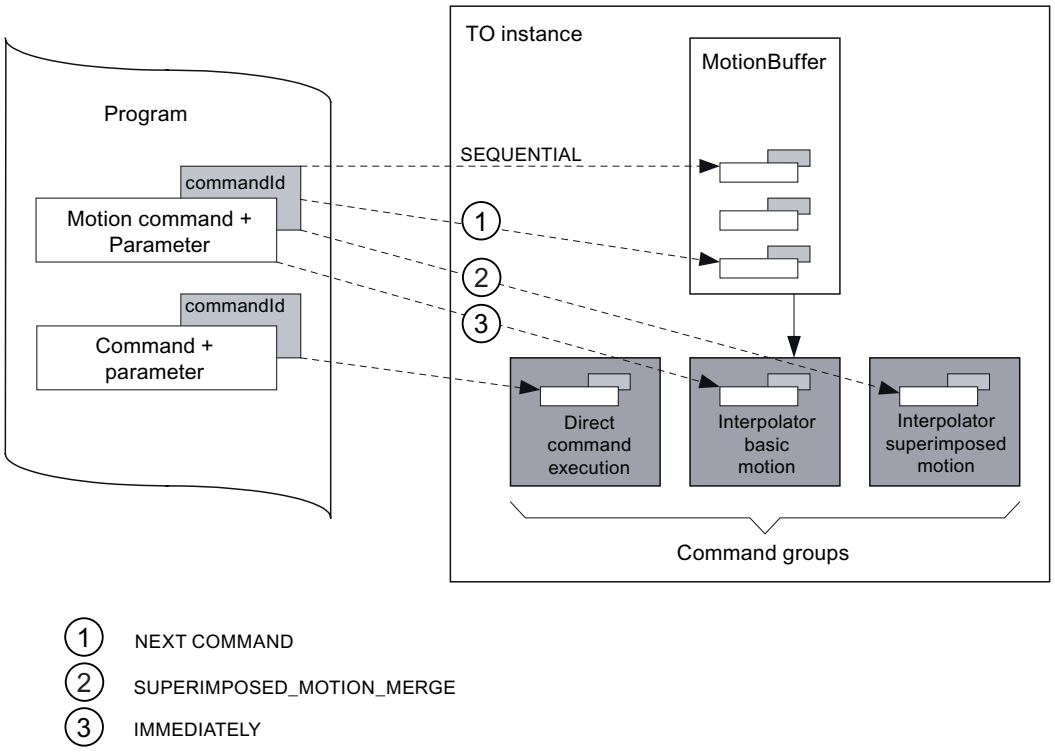


Figure 4-81 Command execution on the axes

The MotionBuffer stores motion commands that are executed sequentially by the interpolator. The number of motion commands that can be stored in the motion buffer is defined via the configuration data *TypeOfAxis.DecodingConfig.numberOfMaxbufferedCommandId*. Multiple motion commands can therefore be issued on the technology object, irrespective of the execution status of the active command.

A **commandId** is assigned to each command when it is issued. This is stored in the command and provides a reference to the issued command.

The commands are assigned to command groups. The pending commands in the existing command groups are processed in parallel by the interpolator. Certain commands become

active on the technology object immediately and form a queue in the motion buffer; other commands, i.e. superimposing commands, take effect immediately.

Note

The behavior of the command groups and command buffers, e.g. MotionBuffer, is TO-specific. Thus, for example, you cannot call the _enableaxistorquelimitpositive and _enableaxistorquelimitnegative commands simultaneously within one IPO2 cycle clock. Only one of the commands will be executed.

An exact description of command groups and command buffers can be found in the TO manuals, e.g. in the "TO Axis Electric / Hydraulic, External Encoder" function manual.

Transition behavior of the currently active motion command

You specify the transition behavior of the currently active motion command on the technology object in the TO function with the parameter *MergeMode*. Here you specify, how the TO function is placed in the command execution sequence on the technology object, or to which command group it is assigned.

Table 4-33 Frequent merge modes

Merge mode	Description
IMMEDIATELY	The motion specified in the command becomes active immediately; already active motions are replaced, pending commands/motions are aborted.
NEXT_MOTION	Execute after the active motion and delete further pending commands/motions.
SEQUENTIAL	Attach to previous commands/motions.
SUPERIMPOSED_ MOTION_MERGE	Superimposed motion on the axis is possible in addition to the basic motion.

Command step enabling condition

A step enabling condition can be programmed for TO functions that are executed in the interpolator. It specifies when the next statement in the program source is executed.
The *nextCommand* parameter is used for this purpose. Refer to the SIMOTION Cam Technology Package, System Functions List Manual (reference list) for the permissible conditions for each command.

The main difference is between asynchronous and synchronous command execution:

- Asynchronous command execution:
  The TO function is transferred to the technology object and the program continued immediately. The *nextCommand* = IMMEDIATELY is set for this.
  In this case, the application must ensure that TO functions are issued only once and checkback signals are evaluated explicitly by scanning the axis or command status (see **Diagnosis of the command execution**).
  Example: see the following two figures
  This method of motion control is referred to as *cyclic programming*. It is permitted in all system tasks and is intended especially for the programming of cyclic tasks.
  The asynchronous command execution is the default setting when parameter *nextCommand* is not specified.
- Synchronous command execution:
  The TO function together with the parameterization for the step enable are transferred to the technology object and the called task stopped. The technology object executes the function and calls for program execution to be resumed as soon as the specified step enabling condition is satisfied or the command has been aborted. For this, *nextCommand* is set to the desired step enabling condition.
  Example: see **Asynchronous program execution (sequential programming)**.
  This method of motion control is referred to as *sequential programming*. It is supported especially by the MotionTasks.
  Programming command sequences in cyclic tasks leads to task timeouts and therefore to runtime errors.

Table 4-34 Example of asynchronous program execution (cyclic programming) - Part 1

```
INTERFACE
USEPACKAGE CAM;
    PROGRAM ProgramCycle;
END_INTERFACE

IMPLEMENTATION

PROGRAM ProgramCycle
    VAR
        boStartCommand : BOOL; // Command - issue command
        boCommandStarted : BOOL; //Auxiliary variable - command issued
        boCommandDone : BOOL; // Auxiliary variable - command executed
        i32Ret : DINT; // Return value of system functions
        sCommandId : CommandIdType; // CommandId
        // Return value - _getStateOfAxisCommand
        sRetCommandState : StructRetCommandState;
        // Instance of the system FB for the edge detection
        r_trig_1 : R_TRIG;
    END_VAR

    r_trig_1 (boStartCommand); // Call the edge detection

    IF r_trig_1.q THEN
        // Request for a system-wide unique commandId
        sCommandId := _getCommandId ();
        // Register commandId at the TO
        // --> Diagnostics of end or abort of command possible
        i32Ret := _bufferAxisCommandId (
        axis := Axis_1,
        commandId := sCommandId );
        // Evaluation of return value of system function
        // ...
        // Issuing of a command - motion with USER_DEFAULT values
        i32Ret := _move(
        axis := Axis_1,
        nextCommand := IMMEDIATELY,
        commandId := sCommandId );
        // Evaluation of return value of system function
        // ...
        // Auxiliary variables for coordination of command execution
        boCommandStarted := TRUE;
        boCommandDone := FALSE;
//-----------------------------------------------------------------------
// Continuation follows
```

Table 4-35 Example of asynchronous program execution (cyclic programming) - Part 2

```
// Continuation
//---------------------------------------------------------------------
    ELSIF boCommandStarted AND NOT boCommandDone THEN
        // Query command execution status
        sRetCommandState := _getStateOfAxisCommand(
        axis := Axis_1,
        commandId := sCommandId );

    IF sRetCommandState.functionResult = 0 THEN
        IF sRetCommandState.commandIdState = EXECUTED THEN
            // Command has been executed (completed)
            boCommandStarted := FALSE;
            boCommandDone := TRUE;
            // Remove registered commandId on TO
            i32Ret := _removeBufferedAxisCommandId(
            axis := Axis_1,
            commandId := sCommandId );
        END_IF;
    ELSE
        // Error handling for _getStateOfAxisCommand function call
        // ...
        ;
    END_IF;

    ELSIF boCommandDone THEN
        // Execution after command execution
        // ...
        ;
    END_IF;

 // Further user program as of here
 // ...

END_PROGRAM

END_IMPLEMENTATION
```

Table 4-36 Example of synchronous program execution (sequential programming)

```
INTERFACE
    USEPACKAGE CAM;
    VAR_GLOBAL
        g_boCommandStarted : BOOL; //Auxiliary variable - command issued
        g_boCommandDone : BOOL; // Auxiliary variable - command executed
        END_VAR
        PROGRAM ProgramSequential;
END_INTERFACE

IMPLEMENTATION
    PROGRAM ProgramSequential
        VAR
            i32Ret : DINT: // Return value of system function
        END_VAR;
        g_boCommandStarted := TRUE;
        g_boCommandDone := FALSE;
        // Statements executed before the motion.
        // ...
        i32Ret := _move(
        axis := Axis_1,
        nextCommand:= WHEN_MOTION_DONE,
        commandId:= _getCommandId () );

        // Statements executed after the motion.
        // ...
        // Evaluation of return value of system function
        // ...

        g_boCommandStarted := FALSE;
        g_boCommandDone := TRUE;

    END_PROGRAM
```

See also

CommandID overview (Page 1541)

Command execution diagnostics

Command identification – commandId

When a system function issues a command, a commandId is transferred. While the command is being executed by the technology object, it stores the commandId in the command, thus identifying the command.

A project-wide unique commandId is obtained with the *_getCommandId* system function. This ensures that no further command with the same commandId exists in the system (unique reference to the command).

Table 4-37 Example of the use of the commandId

```
//...
VAR
    myCommandId : CommandIdType;
END_VAR
//...
// Save unique ID
myCommandId := _getCommandId ();
// Execute function with ID
myFC := _pos (axis := myAxis,
        position        := position_1,
        nextCommand        :=IMMEDIATELY,
        commandId        :=myCommandId);
//...
```

The description how you can track the execution status of a command with the aid of the commandId follows.

System functions for scanning the command/execution status

Technology objects, on which several commands are issued for execution, have *_getStateOf...Command* system functions (e.g. *_get StateOfAxisCommand*). The return value of data type *StructRetCommandState* provides information on the execution status of a motion command by the interpolator in the *EnumCommandIdState* component.

With *EnumCommandIdState* = ABORTED, the abort reason is specified in the component *abortId* (data type DINT). The values correspond to the reason in the technological alarm "30002 Command aborted".

Status check after completion or abort of a command

Once a command has been aborted or has finished executing, it is normally deleted from the internal command management system of the technology object. As a result, it is not possible to diagnose the command status *End* or *Abort* by calling the system functions indicated above.

To enable the status of a command to be scanned even after it has been aborted or its execution is complete, the commandId of the relevant command must be made known to the internal command management of the technology object. This is performed using the *_buffer...CommandId* system function (e.g. _bufferAxisCommandId).

After the *End* or *Abort* status has been evaluated, the commandId must be explicitly removed from the command management of the technology object. This is performed via the system function *_removeBuffered...CommandId* (e.g. *_removeBufferedAxisCommandId*).

Example 1	The _buffer... command is not issued, and the _pos command for which the query is to be made is already finished.
Result	Because a matching command for the CommandId specified in the _getStateOf...CommandId was not found (_pos is already finished), NOT_EXISTENT ('commandId' is not known or command is already finished) is returned.
Example 2	The _buffer... command is issued, and the _pos command for which the query is to be made is already finished.
Result	The _pos command is no longer found, but the result was stored in the CommandId buffer. The result is now either EXECUTED (command processing finished) or ABORTED (command processing aborted).

The size of the CommandID buffer is limited and can be set, for example, for the axis with configuration data item *TypeOfAxis.DecodingConfig.NumberOfMaxBufferedCommandId*. Thus, you can notify the TO regarding the maximum number of commands it has to manage simultaneously.

On the STOP-RUN transition, the buffered Command IDs will be deleted. The CommandID buffer is then empty.

Example: see **Asynchronous program execution (cyclic programming), Part 1 and Part 2**.

The behavior of the "buffer and removeBuffer" commands is the same in all TOs that support this functionality (exceptions: names of commands and name of the configuration data item for the buffer size).

Note

The description applies analogously to the external encoder, as well.

Status check after resetting a technology object

The CommandID can be buffered in such a way that it is not deleted when a technology object is reset. It is then also available after resetting a technology object.

- To do this, call the TO function *__buffer...CommandId* with the parameter *deleteCommandIdWithReset = NO*. The commandId can then only be deleted explicitly with the command *_removeBuffered...CommandId*.
- If you call the *_buffer...CommandId* TO function with the *deleteCommandIdWithReset= YES* (default setting) parameter, the commandId is also deleted with the *_reset...* function (e.g. *_resetAxis*) when resetting the technology object.

See also

CommandID overview (Page 1541)

Identifiers of technology object instances

The identifiers of technology object instances are defined during their configuration in SIMOTION SCOUT. They must be defined uniquely within a SIMOTION device.

In general, you can call the instance of a technology object with its identifier.

However, if the same identifier is defined as data type, variable, function or function block in ST source files or if a global device variable or I/O variable of the same name was created, these identifiers cover the technology object instance.

You can use the predefined name space *_to* so that you can still access the technology object instance, for example to access its system variables or configuration data (see **Name spaces** in the ST Programming Manual). Place the name space identifier in front of the corresponding name, separated by a period, for example *_to.to-name*.

If you want to access the instance of a technology object on another SIMOTION device, place the name of the SIMOTION device in front of the instance identifier, separated by a period, for example *dev-name.to-name* or *dev-name._to.to-name*.

If the identifier of a device is covered, you can use the predefined name space *_project*, for example *_project.dev-name.to-name* or *_project.dev-name._to.to-name*.

Note

With a project-wide unique identifier for the technology object instance, you can use the predefined name space *_project* also for identifying the instance.

Determining the TO name at runtime (as of V4.4)

The system function *_getObjectName* as string is used to determine the configured TO name at runtime.

E.g., you can use it to determine the name of an object causing an error and to display it on an HMI.

When the function is called in the TechnologicalFaultTask, you can transfer TSI#toInst directly from the TaskStartInfo in the function's input parameter.

Conversion of TO data types

Type conversions within the hierarchical data types

The *driveAxis*, *posAxis*, *followingAxis* TO data types are hierarchically structured by the functional scope.

- A position axis (*posAxis* data type) contains the functionality of a speed-controlled axis (*driveAxis* data type).
- A following axis (*followingAxis* data type) contains the functionality of a position axis (*posAxis* data type) and, therefore, also of a speed-controlled axis (*driveAxis* data type).

Type conversions are only possible within these hierarchical data types and with the general ANYOBJECT technology object data type.

Note

Other type conversions are not possible (for example, between a measuring input and following object).

Implicit type conversion

Variables (with TO data type) or TO instances can be assigned to the following variables without specifying a conversion function:

- Variables of a lower hierarchy TO data type:
  - *followingAxis* to *posAxis* or *driveAxis*
  - *posAxis* to *driveAxis*
- Variables of general ANYOBJECT TO data type

See the table below.

Table 4-38 Example of implicit type conversion

```
// The following TO instance (axis) is configured in the project navigator:
fol_axis_real as a following axis

VAR
    drv_axis1 : driveAxis;
    pos_axis1 : posAxis;
    any_obj1 : ANYOBJECT;
END_VAR

drv_axis1 := pos_axis1;
any_obj1  := fol_axis_real;...
```

Explicit type conversion

The *anyObject_to_Object* type conversion function is used to assign variables (with a TO data type) to variables with higher hierarchy TO data types.

A requirement for this is that the source variable (with the hierarchically lower TO data type) must refer to a TO instance that at least corresponds hierarchically to the TO data type of the target variable (see example in the following table).

Table 4-39 Example of successful type conversions

```
// The following TO instances (axes) are configured in the
 // project navigator:
 // pos_axis_real as position axis
 // fol_axis_real as a following axis

VAR
    drv_axis1 : driveAxis;
    pos_axis1 : posAxis;
    any_obj1 : ANYOBJECT;
END_VAR

// Implicit type conversions
drv_axis1 := pos_axis_real;
any_obj1  := fol_axis_real;

// Successful type conversions

pos_axis1 := anyObject_to_Object (in := drv_axis1);
 // Type conversion successful,
 // because drv_axis1 refers to a position axis.

pos_axis1 := anyObject_to_Object (in := any_obj1);
 // Type conversion successful,
 // because any_obj1 refers to a following axis.

//...
```

In the event of a failed type conversion, the value TO#NIL is assigned to the target variable:

Table 4-40 Example of a failed type conversion

```
// The following TO instance (axis) is configured in the
 // project navigator:
 // drv_axis_real as a drive axis

VAR
    pos_axis1 : posAxis;
    any_obj1 : ANYOBJECT;
END_VAR

// Implicit type conversions
any_obj1  := drv_axis_real;

// Type conversion failed
pos_axis1 := anyObject_to_Object (in := any_obj1);
        // Type conversion is not required,
        // because any_obj1 refers to a speed-controlled axis.
        // pos_axis1 has the value TO#NIL.
```

System variables

You can use system variables to assign parameters for technology objects and the basic system or to read their status.

Syntax of system variables

System variables are queried using a structure access with the following syntax. The following table shows the significance of the individual syntax components.

[_to.]TO name.variable.component or

[_device.]variable.component

Table 4-41 Syntax components of system variables

Syntax component	Meaning
TO name	*TO name* stands for the name of a technology object (TO), e.g. of an axis. You have performed one of the following: • Inserted the technology object in SIMOTION SCOUT. • Or declared a variable of the data type of this technology object in the program. You will find a list of all the data types for technology objects in **Function parameters of the technology functions**.
_to	The optional word *_to* identifies the predefined name space for technology objects. It should be used to unambiguously specify a technology object with *TO name*. Also see **Name spaces** in the ST programming manual.

Syntax component	Meaning
_device	The optional word *_device* identifies the predefined name space for device-specific variables (system variables of the SIMOTION devices, I/O variables, global device variables). It should be used to unambiguously specify the variable as system variable of the SIMOTION device. These system variables are also available when no technology object is loaded. Also see **Name spaces** in the ST programming manual.
Variable	*Variable* stands for the name of the system variable as found in the list of all system variables (see List Manual for the SIMOTION technology package system variables).
Component	*Component* stands for the part of the structure you want to query. This can be an additional structure that contains further components, as well. The depth of the structure depends on the system variable and can be zero.

Using system variables

Read access to a system variable is always possible. A system variable can be assigned to a variable in two ways:

- With a value assignment (see also **Value assignments** in the ST programming manual). This means ExecutionFaultTask is called in the event of an error (please also refer to *Reading out the substitute value or last values* at the end of the chapter). For more information on the error reaction, see Accesses to system variables (Page 1258).
- With the *_getSafeValue* and *_setSafeValue* system functions (see _getSafeValue function (Page 1514), _setSafeValue function (Page 1517), and Accesses to system variables and inputs/outputs (Page 1514)). In this case it is possible to program the desired response in the event of an error.

The use in an expression or as a parameter in a function or function block is also possible. In this case, ExecutionFaultTask is also called in the event of an error (see Accesses to system variables (Page 1258)).

If a system variable can be written is specified in the List Manual (reference list) of the system variables of the technology objects (TO) in the following entry:

- Effective: *immediately* (can be read and written to)
- Effective: *read only* (can only be read).

If a system variable can be written to, it can be assigned a value in two ways:

- With a value assignment (see also **Value assignments** in the ST programming manual). This means ExecutionFaultTask is called in the event of an error (please also refer to *Reading out the substitute value or last values* at the end of the chapter). For more information on the error reaction, refer to Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258).
- With the system function _setSafeValue (see function _setSafeValue (Page 1517)). In this case it is possible to program the desired response in the event of an error.

TO#NIL is a special variable value. You can use this value to check whether a valid technology object is present (see the example in **Function parameters of the technology functions**).

Note

For performance reasons, you should only access system variables when absolutely necessary. Instead, save their contents in a local variable of the same data type. Local variable accesses need much fewer resources because less processor time is used. For more information, see **Efficient programming**.

Also note that a source of error can be comparing REAL variables, LREAL variables, and system variables (for example, axis position) with each other, see **Compare REAL or LREAL variables**.

Note

As of SIMOTION Kernel V4.4, system variables can be accessed from synchronous user tasks.

Note

System variables saved ONLINE cannot be saved with "Save to memory card (Ram2Rom)" or "Save in the engineering project (load configuration data to PG)" is not possible.

So that values of system variables can also be saved in the engineering project and on the memory card, the default value of system variables must be changed OFFLINE and then loaded to the target system per download and saved. See Storage concept in the target system (Page 1637).

Scope of system variables

1. System variables, for example, the status flag, may only exist for just one IPO cycle clock.
2. All system variables have the documented **Update** property.
3. If you want to query the status of a task with a lower cycle time, the status should be linked with the following OR status. The OR operation ensures that all states of the application are used. The operation ensures that the subsequent ENUMS of the status flag are grouped.

Examples of system variables

You want to check the axis position and dynamic axis state of *Axis1*.

Requirements:

- You have created Axis1 in SIMOTION SCOUT or have defined and initialized it in the program, e.g. using the *myAxis* variable of *PosAxis* data type.
- You have defined variables in the program for recording the axis position and the dynamic axis state. The data type of these variables must match the data type of the variables to be checked, e.g.:

```
VAR
act_pos : LREAL;
```

```
act_motionState : EnumAxisMotionState;
END_VAR
```

Example of accessing a system variable using a structure element of an elementary data type:

```
act_pos := Axis1.positioningState.actualPosition;
```

PositioningState is the system variable and *actualPosition* is the structure element of data type LREAL that will be queried.

Example of querying a system variable with an enumeration element:

```
act_motionState := Axis1.motionStateData.motionState;
```

motionStateData is the system variable and *motionState* is the structure element of enumerator data type *EnumAxisMotionState* that will be queried.

Initialization of system variables

System variables are not reinitialized as a rule when you download the project; their actual values are not reset to the initial values. In general, system variables are only reinitialized when the SIMOTION device is restarted.

Substitute value or the last value of system variables at TO restart or where TO is deactivated (as of V4.1)

The access to system variables is also possible at RESTART of the TO or at activated TO, without the system going to STOP. Instead of reading out the system variables using the _getSaveValue function, you can configure the following by means of an entry in the config data (restart.behaviorInvalidSysvarAccess) to enable direct access:

- Read out last value (LAST_VALUE = default)
- Read out default value (=value when loading the project; DEFAULT_VALUE)
- Go to STOP (STOP_DEVICE)

Exception

Variables that deliver the current TO status, also return the correct status at RESTART. This affects the system variable restartActivation, which you can access via _getSafeValue.

Note

TO system variables can be written to during a TO restart or where a TO is deactivated (apart from STOP_DEVICE). The values will be applied or are effective after the RESTART. If writing system variables exceeds the limits, the CPU will go into STOP mode. As of V4.2, writing can take place outside the valid limits. The limit value will be accepted in this case.

See also

Function parameters of the technology functions (Page 1228)

Configuration data

Configuration data defines the basic functionality of a technology object. It is normally set during the configuration of the technology object with SIMOTION SCOUT and for the most part cannot be modified during runtime.

A major part of this configuration data can, however, be modified while the program is running. Whether or not a configuration data item can be modified during runtime is specific to each configuration data item and is documented in the List Manual for SIMOTION technology package configuration data.

Depending on the configuration data item, the following options are available:

- Cannot be modified online:
  This configuration data can only be modified during configuration of the technology object with SIMOTION SCOUT.
- Can be modified online, effective after restart:
  This configuration data can be changed by variable access from the user program. The change does not take effect until the technology object is restarted (see **Resetting a technology object**).
- Can be modified online, immediately effective:
  This configuration data can be changed by variable access from the user program. Change is immediately effective.

Note

If you make a change to configuration data which only takes effect after a TO restart, subsequent changes to "effective immediately" configuration data will also only take effect after the TO restart.

Note

As of SIMOTION Kernel V4.4, configuration data can be accessed from synchronous user tasks.

Note

Configuration data changed ONLINE can be saved on a memory card with "Copy current data to RAM" and subsequent "Copy from RAM to ROM" or saved with "Load to PG" in the engineering project. See Storage concept in the target system (Page 1637).

Reading the configuration data

You can read each configuration data item of a technology object and assign it to a variable, for example:

- The value saved in the SIMOTION device's RAM (can be read in the "Next value" column in the expert list)
  To do so, use the syntax: *TO-name*.setconfigdata.*config-date*.
- Value currently in effect on the technology object
  To do so, use the syntax: *TO-name*.activeconfigdata.*config-date*.

See example in the table below.

Only access to individual variables of the configuration data is permitted.

Table 4-42 Read access to a configuration data item of a technology object

```
VAR
    lreal_var : LREAL;
    drive_var : driveaxis;
END_VAR

lreal_var :=
    drive_var.setconfigdata.TypeOfAxis.MaxJerk.maximum;
    // Read access to saved value
lreal_var :=
    drive_var.activeconfigdata.TypeOfAxis.MaxJerk.maximum;
    // Read access to value currently in effect
```

A configuration data item can be assigned to a variable in two ways:

- With a value assignment, such as the example in the previous table (see also **Value assignments** in the ST Programming Manual). This means *ExecutionFaultTask* is called in the event of an error (please also refer to *Substitute value or last value* at the end of the chapter). For more information about the error reaction, see Errors when accessing configuration data (Page 1514).
- With the *_getSafeValue* system function (see function _getSafeValue (Page 1514)). In this case it is possible to program the desired response in the event of an error.

The use in an expression or as a parameter in a function or function block is also possible. In this case, the *ExecutionFaultTask* is called in the event of an error.

Modifying configuration data at runtime (online modification)

Configuration data is modified online by a simple variable write access to the value stored in the RAM. To do so, use the syntax:
TO-name.setconfigdata.*config-date*.

The effectiveness (adoption as the current value in effect on the technology object) is determined by the configuration data item (effective immediately / effective after restart).

See example of **write access to a configuration data item**.

Only access to individual variables of the configuration data is permitted.

Table 4-43 Write access to a configuration data item of a technology object

```
VAR
    lreal_var : LREAL;
    drive_var : driveaxis;
END_VAR

drive_var.setconfigdata.TypeOfAxis.MaxJerk.maximum :=
    200000.0;
    // Write access to saved value
```

A configuration data item can be assigned a value in two ways:

- With a value assignment, such as the example in the previous table (see also **Value assignments** in the ST programming manual). This means *ExecutionFaultTask* is called in the event of an error (please also refer to *Substitute value or last value* at the end of the chapter). For more information about the error reaction, see Errors when accessing configuration data (Page 1258).
- With the system function *_setSafeValue* (see function _setSafeValue (Page 1517)). In this case it is possible to program the desired response in the event of an error.

In addition, there is the option to control the activation via a system variable.

Use technology object system variable *activationModeChangedConfigData* to define when the modified configuration data is to take effect:

- If *activationModeChangedConfigData = ACTIVATE_CHANGED_CONFIG_DATA* is set, the modified data is set immediately active. If the system variable is set to this value, data collected up to this point is also activated.
- If *activationModeChangedConfigData = COLLECT_CHANGED_CONFIG_DATA* is set, the modified data is collected.
  They are activated as a body as soon as this system variable is set to *ACTIVATE_CHANGED_CONFIG_DATA*.
  The collected, modified configuration data can be deleted (without activation) by calling the corresponding technology object system function (e.g. *_resetAxisConfigDataBuffer*).

Note

When *activationModeChangedConfigData*:= *ACTIVATE_CHANGED_CONFIG_DATA*, it takes a certain amount of time for a modified configuration data item to take effect after it has been written.

In particular, when several configuration data items are changed simultaneously, timeouts can occur in the tasks.

Note

If you want to change several configuration data items at the same time, it is advisable to collect them first with *activationModeChangedConfigData = COLLECT_CHANGED_CONFIG_DATA* and then activate them as a body in a sequential task using the *ACTIVATE_CHANGED_CONFIG_DATA* setting.

Configuration data changed at runtime can be saved on card and in the ES project, see Memory access (Page 1640).

Note

When accessing configuration data from the user program (e.g. in a general FB), make sure that the respective configuration data is also available depending on TO type.

Access to configuration data for TO references and axis arrays

Access to configuration data for TO references is possible. To access configuration data of an axis array you must use an intermediate variable. If you attempt to use axis arrays directly you will get a compiler message that an intermediate variable should be used.

Example

```
VAR_TEMP
    myPosAxis  :posaxis;
END_VAR
myPosAxis      :=Pos[2]

myPosAxis.setconfigdata.TypeOfAxis.MaxJerk.maximum :=
    200000.0;
```

Because you are accessing the axis directly via a reference, no further assignment is necessary to access the value. You do not have to reassign pos[2] to myPosAxis.

Note

Write operations to configuration data are possible during a TO restart or where a TO is deactivated (apart from STOP_DEVICE). The values will be applied or are effective after the RESTART.

Substitute value or the last value of configuration data at TO restart or where TO is deactivated (as of V4.1)

Configuration data can also be accessed when a RESTART of the TO is performed or where the TO is deactivated without the system going to STOP. Instead of reading out the configuration data using the _getSaveValue function, you can configure the following by means of an entry in the configuration data (restart.behaviorInvalidSysvarAccess):

- Read out last value (LAST_VALUE = default)
- Read out default value (=value when loading the project; DEFAULT_VALUE)
- Go to STOP (STOP_DEVICE)

If writing configuration data exceeds the applicable limits, the CPU will go to STOP. As of V4.2, writing can take place outside the valid limits. The limit value will be accepted in this case.

Resetting a technology object

⚠ CAUTION
Resetting a technology object aborts the current motion without an error message.

To integrate configuration data that require a restart of the technology object, you must reset the technology object. The procedure depends on the *restart.restartActivationSetting* configuration data item of the technology object:

- For the setting *restart.restartActivationSetting* = RESTART_BY_COMMAND:
  The technology object can only be restarted by means of the corresponding system function (e.g. *_restartAxis*). To do so, set the parameter *activateRestart* = ACTIVATE_RESTART.
- For the setting *restart.restartActivationSetting*= RESTART_BY_SYSVAR_AND_COMMAND:
  The restart can be performed in two ways:
  - By calling the corresponding system function (e.g. *_restartAxis*). By setting parameter *activateRestart* = ACTIVATE_RESTART.
  - By assigning a value to the technology object system variable: *restartActivation*= ACTIVATE_RESTART. The system variables are initialized, the technology object loses all of its status information, such as axis homed.

  The restart is always performed asynchronously. After a successful restart of the technology object, this system variable has the value NO_RESTART_ACTIVATION.

Use of technology packages in libraries

Libraries can also contain TO functions and accesses to the system variables of a technology object.

You specify the SIMOTION device and the technology object for which the library is being compiled in the object properties for the library as follows:

1. Select the library in the project navigator.
2. Select the **Edit > Object properties** menu command.
3. Then select the **Technology packages** tab.
4. Select the SIMOTION devices (including the version number) and the technology packages for which the library is to be compiled.

Note

To compile a project without errors, observe the rules governing the selection of SIMOTION devices and technology packages in the following table!

You can also specify the technology package in the library's ST source files if you want (with the USEPACKAGE command), however, this is not necessary.

Table 4-44 Selecting devices and technology packages in a library

Selection	Description
Device-independent	You must also select: • The technology packages • The version number of the selected technology packages **Note:** 1. The library is compiled without reference to a SIMOTION device or a version of the SIMOTION kernel. **For this reason, the following must not be used:** – System functions of SIMOTION devices – System variables of SIMOTION devices – Version-dependent system functions – Configuration data of the technology objects 2. The library is compiled precisely to the version selected. The use of system functions or variables which are not available in the selected version will result in a compilation error. 3. If a device-independent library is to be made available for another version it must be copied and inserted under a different name. This copy must be recompiled with a different version reference.
SIMOTION device including version (multiple selection possible)	Only those technology packages are displayed that are available for all of the selected devices. **Note:** 1. The library is compiled for all of the selected devices and technology packages (of the selected device versions). 2. The use of system functions or variables which are not available for one of the selected devices, or the technology package of the respective device version, will result in a compilation error. 3. The library can only be used for the selected devices and technology packages. When you use the library in an ST source file, the following is therefore checked: – Whether the library is compiled for the SIMOTION device (including version) that contains the importing ST source file. – Whether the technology package set on the SIMOTION device and specified in the ST source file with the USEPACKAGE command corresponds to the one in the library. Any inconsistencies will result in compilation errors.

4.2.5.3 Response to faults and events

Evaluating faults and events

The SIMOTION system has various execution levels for scheduling programs. One of these execution levels is the interrupt execution level, which is started in response to specific events, such as errors.

Options for responding to faults and events

There are two classes of events that can start tasks in the interrupt execution level:

- When events are user-defined, they are called UserInterruptTasks; the programs assigned to these tasks are also user-defined.
- If the events are system or technology-related (system errors or technology objects), we refer to SystemInterruptTasks. The following table lists the available SystemInterruptTasks. Messages generated by the system are known as alarms. The reaction is defined in the alarm configuration (see below).

Table 4-45 Specified SystemInterruptTasks

SystemInterruptTask	Called for the following events
ExecutionFaultTask	Execution errors in programs
PeripheralFaultTask	Process and diagnostic alarms from I/O modules
TechnologicalFaultTask	Alarms, warnings, and notes from technology objects
TimeFaultBackgroundTask	Timeout for the BackgroundTask
TimeFaultTask	Timeouts for TimerInterruptTasks

Note

Message generation is another system feedback option. User-assigned messages can be used in programs, e.g. if certain events occur (tank empty, etc.). For more information, see Programming messages (Page 1563).

Alarm configuration

You can define the system behavior for technological alarms in SIMOTION SCOUT (alarm configuration). You can choose between:

- STOP: Transition to STOP mode (all system and user tasks are stopped.)
- STOP U: Transition to STOP U mode (only user program tasks are stopped.)
- START TechnologicalFaultTask: Starts the associated SystemInterruptTask
- NONE: No response

Each alarm has a default response. Details for the alarm configuration can be found at Error handling for technology objects (Page 1281).

When you select **START TechnologicalFaultTask**, you must assign a program to the TechnologicalFaultTask that responds to the associated alarm.

Besides a configurable response to program execution, alarms also have a reaction in the technology object (see SIMOTION Motion Control Technology Objects function manuals).

Execution errors in programs

You set the response to execution errors in programs in the task configuration (see **Configuring the execution system**).

This affects the following operations, for example:

- Invalid operation with floating-point numbers (Page 1257) (e.g. logarithm of a negative number)
- Incorrect type conversions
- Division by zero
- Violation of array limits

The possible error responses are listed in the following table.

Table 4-46 Error response in case of execution errors in the program

Error response	Task type	Description
CPU to STOP	All tasks	SIMOTION device switches to STOP mode and the ShutdownTask is started.
ExecutionFaultTask	Sequential (non-cyclic)	The ExecutionFaultTask is started. The task in which the error occurred, is aborted; the aborted task can be dispatched again by the user.
	Cyclic	The ExecutionFaultTask is started. The task in which the error occurred is aborted. When the ExecutionFaultTask is finished, the SIMOTION device switches to STOP mode; the ShutdownTask is started.

See also

Specifications for the configuring (Page 1417)

Error during operations with floating-point numbers (FPU exceptions)

Invalid floating-point numbers

The data types for REAL and LREAL floating-point numbers are implemented with their bit patterns according to IEEE 754. Accordingly, the following bit patterns for invalid floating-point numbers are also supported:

- Signaling NaN (NaNs): Invalid bit pattern (Not a Number), which triggers an error (FPU exception) on each operation.
- Quiet NaN (NaNq): Invalid bit pattern (Not a Number), which triggers an error (FPU exception) only on certain operations.
- Infinity: Bit pattern for + infinity or – infinity.

Note

You can create an NaN or infinity from the corresponding bit pattern, e.g. with system function DWORD_TO_REAL, BigByteArray_to_AnyType or LittleByteArray_to_AnyType.

Note

When a floating-point number is displayed in the engineering system (e.g. in the symbol browser of SIMOTION SCOUT), no distinction is made between a signaling and a quiet NaN.

FPU exceptions

The operations with floating-point numbers are also implemented according to IEEE 754. If any of the indicated errors for the operations below occur, an FPU exception will be triggered:

- Any operation with a signaling NAN (NaNs).
- Addition: Both addends are infinity but have different signs.
- Subtraction: Minuend and subtrahend are infinity and have the same sign.
- Multiplication: One factor is 0 and the other is infinity.
- Division:
  - Both operands are 0 or infinity.
  - Divisor is 0.
- Modulo division (x MOD y): x = infinity or y = 0.

- The argument of a system function is outside the domain, e.g.:
  - SQRT: The radicand is < 0.
  - LN, LOG: The argument is ≤ 0.
  - EXPT or Operator "**": Base is ≤ 0:
    Exceptions as of Version 4.1 of the SIMOTION Kernel:
    Base is < 0 and exponent has an integer value.
    Base is = 0 and exponent is > 0.
  - SIN, COS, TAN: The argument is infinity.
  - ASIN, ACOS: Argument is > 1
- Conversion of a floating-point number to an integer or its assigned bit data type with the corresponding system function (e.g. LREAL_TO_DINT, REAL_VALUE_TO_DWORD):
  - Range of target data type is exceeded.
  - Argument is a signaling or quiet NaN (NaNs or NaNq).
  - Argument is infinity.
- Comparison operations:
  - At least one operand is a signaling or quiet NaN (NaNs or NaNq).
  - At least one operand is infinity.
- Range is exceeded for operations with valid floating-point numbers

The behavior specified for the processing errors in programs (Page 1256) will be carried out. If the ExecutionFaultTask is called, then TSI#executionFaultType = _SC_INVALID_FLOATING_POINT_OPERATION, see TaskStartInfo of the ExecutionFaultTask (Page 1267).

Note

No error (no FPU exception) is triggered:

- For operations with quiet NaN (NaNq), if not mentioned explicitly above.
  For example, the addition of a valid floating-point number to a quiet NaN (NaNs) produces the same quiet NaN (NaNs).
- For operations with + infinity or - infinity, if not mentioned explicitly above.
  For example, the addition of a valid floating-point number to + infinity produces +infinity again.

Access errors to system variables and configuration data, as well as I/O variables for direct access

This section describes the behavior when errors occur while accessing system variables, configuration data or I/O variables with the usual methods (using the variable identifier in an expression or variable assignment), see also TaskStartInfo of the ExecutionFaultTask (Page 1267).

Errors in system variables and configuration data:

As of V4.1 SP2/V4.1 SP3, system variables and configuration data can also be accessed when a RESTART of the TO is performed or where the TO is deactivated without the system entering STOP mode.

You can configure the following by means of an entry in the configuration data (restart.behaviorInvalidSysvarAccess):

- Read out last value (LAST_VALUE = default)
- Read out default value (=value when loading the project; DEFAULT_VALUE)
- Go to STOP (STOP_DEVICE)
  The ExecutionFaultTask is started, and the additional error response depends on the task type (sequential or cyclic) in which the error occurs (see the following table).
  Response at start of ExecutionFaultTask for incorrect access to system variables and configuration data

Task type	Description
Sequential (non-cyclic)	The ExecutionFaultTask is started. The task in which the error occurred is aborted; the aborted task can be dispatched again by the user.
Cyclic	The ExecutionFaultTask is started. The task in which the error occurred is aborted. When the ExecutionFaultTask is finished, the SIMOTION device switches to STOP mode; the ShutdownTask is started.

Writing of system variable values outside the applicable limits is not affected by the configuration data referred to above (e.g. STOP_DEVICE). As of V4.2, the limit value is written. See also System variables (Page 1245), Configuration data (Page 1249), General information on accessing system variables and inputs/outputs (Page 1514) or Errors when accessing system data with _get/_setSafeValue (Page 1295).

Definition of DEFAULT_VALUE and LAST_VALUE

- DEFAULT_VALUE
  The DEFAULT_VALUE is the configured value. This is the value transferred during a download (system variables and configuration data).
- LAST_VALUE
  - Configuration data
    The last value can be the configured value. The value will then be equivalent to the value shown under "Current value" in the expert list.
    OR
    The last value can be the last configured value. The value will then be equivalent to the value shown under "Next value" in the expert list.
    Either of these values can be output, depending on the configuration data entered.
  - System variables
    With system variables, the last value is the value which is currently set and effective.

Errors for I/O variables (direct access to inputs and outputs)

The error response is specified when the I/O variables are defined (see **Direct access and process image of the cyclical tasks** in the ST programming manual).

- CPU stop: The ExecutionFaultTask is started. The SIMOTION device then switches to STOP mode; the ShutdownTask is started.
- Substitute value: The substitute value specified when the I/O variable was defined is adopted, and the task is continued.
- Last value:
  With read access (to inputs or outputs): The last valid value is applied, and the task is continued.
  With write access (to outputs): The value is written to the variable. However, it will not be active at the output until the output becomes available again. The task is continued.

In specific cases it is necessary to respond differently to errors, to deviate from the defined error response, or to avoid errors by performing queries beforehand. The functions _getSafeValue (Page 1514) and _setSafeValue (Page 1517), and getInOutByte (Page 1520) serve this purpose. The functions *_getSafeValue* and *_setSafeValue* are very time intensive.

Errors when generating the process image

This section describes the behavior when I/O access errors occur during updating of the process image with the assigned task. Possible causes:

- The I/O module is not present.
- The I/O module is switched off.
- The connection to the I/O module is missing or faulty.
- The I/O module is signaling an error.

The behavior is as follows:

- For the process image of the cyclic tasks (see **Direct access and process image of the cyclic tasks** in the programming manuals)
  The error response is specified when the I/O variables are defined:
  - CPU stop: For response information, see the following table.
  - Substitute value: The substitute value specified when the I/O variable was defined is adopted, and the cyclic task is continued.
  - Last value:
    Process input image (reading of inputs): The value of the process image at the address is not changed; the cyclic task is continued.
    Process output image (writing to outputs): The value only takes effect at the output with the address when the output is available again; the cyclic task is continued.
- For the fixed process image of the BackgroundTask (see **Accesses to the fixed process image of the BackgroundTask** in the programming manuals):
  The error response depends on whether direct access was defined at the same address using I/O variables:
  - **No** direct access is defined: Error response is always CPU STOP; for response information, see the following table.
  - Direct access is defined: The error response specified in the definition of the I/O variables does apply (see above, like process image of cyclic tasks).

Table 4-47 Error response during process image update for CPU STOP response

Event	Description
Error occurs	1. An incoming message is generated once. 2. If no program is linked to the PeripheralFaultTask, the SIMOTION device goes to STOP mode, and the ShutdownTask is started. 3. Otherwise: – The PeripheralFaultTask is started once **immediately** (rather than in the next IPO cycle clock): *TSI#interruptId* = _SC_IMAGE_UPDATE_FAILED. *TSI#logBaseAdrIn* or *TSI#logBaseAdrOut* contains the address at which the error occurred. See TaskStartInfo of the PeripheralFaultTask (Page 1269). – Process input image: The substitute value is assigned to the value of the process image at the address. Process output image: Value will not take effect at the output with the address until the output becomes available again. – The cyclic task in which the error occurred is continued.
Error persists	• No additional messages are generated. • The PeripheralFaultTask is not restarted. • Process input image: The value of the process image at the address is not changed. Process output image: Value will not take effect at the output with the address until the output becomes available again.
Error disappears	1. An outgoing message is generated once. 2. The PeripheralFaultTask is started once **immediately** (rather than in the next IPO cycle clock): *TSI#interruptId* = _SC_IMAGE_UPDATE_OK, see TaskStartInfo of the PeripheralFaultTask (Page 1269). 3. The cyclic task is continued.

Using Taskstartinfo

Important information about starting the task is stored in the Taskstartinfo for each task, e.g.:

- Start time of task
- For the TechnologicalFaultTask: the triggering instance of the Technology Object and the alarm number,
- For the TimeFaultTask: TimerInterruptTask that caused the timeout error.

Within a task, you can query the relevant Taskstartinfo of this task. To do this, use the **TSI#<info>** system variable; where **<info>** is the particular information to be queried. The content and scope of the TaskStartInfo and the associated system variables depend on the relevant task. This is described in the following sections.

Details on the TaskStartInfo of the

StartupTask (Page 1263)
MotionTasks (Page 1264)
BackgroundTask (Page 1264)
TimerInterruptTasks (Page 1264)
UserInterruptTasks (Page 1265)
SynchronousTasks (Page 1265)

SystemInterruptTasks
- ExecutionFaultTask (Page 1267)
- PeripheralFaultTask (Page 1269)
- TechnologicalFaultTask (Page 1277)
- TimeFaultBackgroundTask (Page 1278)
- TimeFaultTask (Page 1279)

ShutdownTask (Page 1279)

The TaskStartInfo query is used mainly for SystemInterruptTasks, see example for using the TaskStartInfo of the TechnologicalFaultTask (Page 1280).

See also

Evaluating in the user program (Page 1293)

TaskStartInfo of the StartupTask

Table 4-48 TaskStartInfo of the StartupTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use

TSI#taskType	:	ENUM...	Type of task
			As of version V4.5 of the SIMOTION Kernel
			Data type: EnumTaskType
			Value: STARTUP
			`TYPE EnumTaskType`
			`  STARTUP  // (1)`
			`  MOTION  // (2)`
			`  BACKGROUND  // (3)`
			`  SERVO_SYNCHRONOUS  // (4)`
			`  IPO_SYNCHRONOUS  // (5)`
			`  PWM_SYNCHRONOUS  // (6)`
			`  INPUT_SYNCHRONOUS  // (7)`
			`  POST_CONTROL  // (8)`
			`  SYSTEM_INTERRUPT  // (9)`
			`  TIMER_INTERRUPT  // (10)`
			`  USER_INTERRUPT  // (11)`
			`  SHUTDOWN  // (12)`
			`END_TYPE`

TaskStartInfo of the MotionTasks

Table 4-49 TaskStartInfo of the MotionTasks

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: MOTION

TaskStartInfo of the BackgroundTask

Table 4-50 TaskStartInfo of the BackgroundTask

Designation	:	Data type	
TSI#startTime	:	DT	Time of cycle control point
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is non-equidistant and cyclic)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is non-equidistant and cyclic) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: BACKGROUND

TaskStartInfo of the TimerInterruptTasks

Table 4-51 TaskStartInfo of the TimerInterruptTasks

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Time of cycle control point
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms

Designation	:	Data type	Meaning
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: TIMER_INTERRUPT

TaskStartInfo of the UserInterruptTasks

Table 4-52 TaskStartInfo of the UserInterruptTasks

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: USER_INTERRUPT

TaskStartInfo of the SynchronousTasks

Table 4-53 TaskStartInfo of the SynchronousTasks

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of task (= cycle clock with which the task runs synchronously)
TSI#currentTaskId	:	StructTaskId	TaskId of task

Designation	:	Data type	Meaning
TSI#cycleTime	:	TIME	Configured cycle time of task in ms • ServoSynchronousTask: Position control cycle clock • ServoSynchronousTask_fast: Fast position control cycle clock • IPOsynchronousTask: Interpolator cycle clock (IPO) • IPOsynchronousTask_fast: Fast interpolator cycle clock IPO • IPOsynchronousTask_2: Interpolator cycle clock (IPO_2) • PWMsynchronousTask: Pulse-width modulation cycle clock • InputSynchronousTask_1: Input1 cycle clock • InputSynchronousTask_2: Input2 cycle clock • PostControlTask_1: Control1 cycle clock • PostControlTask_2: Control2 cycle clock
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs • ServoSynchronousTask: Position control cycle clock • ServoSynchronousTask_fast: Fast position control cycle clock • IPOsynchronousTask: Interpolator cycle clock (IPO) • IPOsynchronousTask_fast: Fast interpolator cycle clock IPO • IPOsynchronousTask_2: Interpolator cycle clock (IPO_2) • PWMsynchronousTask: Pulse-width modulation cycle clock • InputSynchronousTask_1: Input1 cycle clock • InputSynchronousTask_2: Input2 cycle clock • PostControlTask_1: Control1 cycle clock • PostControlTask_2: Control2 cycle clock SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Values: • SERVO_SYNCHRONOUS • IPO_SYNCHRONOUS • PWM_SYNCHRONOUS • INPUT_SYNCHRONOUS • POST_CONTROL

TaskStartInfo of the ExecutionFaultTask

Table 4-54 TaskStartInfo of the ExecutionFaultTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use

Designation	:	Data type	Meaning
TSI#executionFault Type	:	UDINT	Type of execution error in user program • _SC_DIVISION_BY_ZERO (= 500) Error during division or modulo division of integers (data type ANY_INT or ANY_BYTE): Divisor is 0. • _SC_INVALID_FLOATING_POINT_OPERATION (= 501) Error during operations with floating-point numbers (FPU exceptions) (Page 1257) • _SC_ARRAY_BOUND_ERROR_READ (= 502) Array boundaries exceeded during read access • _SC_ARRAY_BOUND_ERROR_WRITE (= 503) Array boundaries exceeded during write access • _SC_VARIABLE_ACCESS_ERROR_READ (= 504) Error during read access: – To a system variable of a Technology Object: Violation of array limits; Access while Technology Object is being reset (RESET); Access to a Technology Object data type variable with a value of TO#NIL; – To an I/O variable by means of direct access. See Errors when accessing system variables and configuration data as well as I/O variables for direct access (Page 1258). • _SC_VARIABLE_ACCESS_ERROR_WRITE (= 505) Error during write access: – To a system variable of a Technology Object: Violation of array limits; Access while Technology Object is being reset (RESET); Access to a Technology Object data type variable with a value of TO#NIL; – To an I/O variable by means of direct access. See Errors when accessing system variables and configuration data as well as I/O variables for direct access (Page 1258). • _SC_TO_INSTANCE_NOT_EXISTENT (= 506) Access to a non-existent TO instance: A Technology Object data type variable with a value of TO#NIL is used in a TO function. • _SC_CLASS_INSTANCE_NOT_EXISTENT (= 507) Access to a non-existent class: An interface variable or pointer with value NULL is used. • _SC_TASK_STACKSIZE_FAULT (= 508) Stack size exceeded
TSI#taskId	:	StructTaskId	TaskId of task in which the execution error has occurred.
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SYSTEM_INTERRUPT

TaskStartInfo of the PeripheralFaultTask

Table 4-55 TaskStartInfo of the PeripheralFaultTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#interruptID	:	UDINT	• _SC_PROCESS_INTERRUPT (= 200) Process alarm occurred at I/O module Additional information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#details – TSI#eventClass – TSI#faultId For information about process alarms, refer to **Process alarms** below. • _SC_DIAGNOSTIC_INTERRUPT (= 201) Diagnostic interrupt occurred at I/O module Additional information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#logDiagAdr – TSI#details – TSI#eventClass – TSI#faultId • _SC_STATION_DISCONNECTED (= 202) PROFIBUS DP: A DP slave station has failed PROFINET IO: Station failure of an I/O device Additional information in the following TSI: – TSI#logDiagAdr – TSI#eventClass – TSI#faultId

Designation	:	Data type	Meaning
TSI#interruptID (continued)	:	UDINT	• _SC_STATION_RECONNECTED (= 203) PROFIBUS: Station reconnection of a DP slave PROFINET IO: Station reconnection of an I/O device Additional information in the following TSI: – TSI#logDiagAdr – TSI#eventClass – TSI#faultId • _SC_IMAGE_UPDATE_FAILED (= 204) Error during process image update (for DP slave: in connection with station failure) Additional information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#loDiagAdr See Errors when generating the process image (Page 1260). • _SC_PC_INTERNAL_FAILURE (= 205) Error in local controller module Additional information in the following TSI: – TSI#details • _SC_IMAGE_UPDATE_OK (= 206) Process image update works again (for DP slave: In connection with station reconnection) Additional information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – #logDiagAdr See Errors when generating the process image (Page 1260). • _SC_DP_CLOCK_DETECTED (= 207) Clock signal detected for the first time, and valid PRM telegram is received • _SC_DP_SYNCHRONIZATION_LOST (= 208) Multiple cycle failure or PLL has slipped (in internal DP_INTERFACES_SYNCHRONIZED state). PLL switches to uncontrolled mode • _SC_DP_SLAVE_SYNCHRONIZED (= 209) PLL has slipped into synchronized operation • _SC_DP_SLAVE_NOT_SYNCHRONIZED (= 210) Multiple cycle failure or PLL has slipped (in internal DP_SLAVE_SYNCHRONIZED state). PLL remains in controlled mode

Designation	:	Data type	Meaning
TSI#interruptID (continued)	:	UDINT	• _SC_IO_MODULE_SYNCHRONIZED (= 214) e.g. onboard SINAMICS measuring input (Control Unit) e.g. TM15 / TM17 High Feature / DO1 with telegram 39x: Synchronization attained. More information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#logDiagAdr • _SC_IO_MODULE_NOT_SYNCHRONIZED (= 215) e.g. TM15/TM17 High Feature/DO1 with telegram 39x: Synchronization failed. More information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#logDiagAdr • _SC_PULL_PLUG_INTERRUPT (= 216) PROFINET IO: Unplugging or plugging of modules in an I/O device. More information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#eventClass – TSI#faultId – TSI#logDiagAdr • _SC_DRIVE_OBJECT_FAULT (= 217) Fault message from DO1; cause of fault stored in the fault buffer, can be read out with _readDriveFaults. The alarm must be acknowledged with _resetDriveObjectFault. More information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#logDiagAdr • _SC_DRIVE_OBJECT_ALARM (= 218) Warning message from DO1; warnings are stored in the warning parameters, can be read out with _readDriveParameter. The alarm cannot be acknowledged. It is triggered by the arrival and departure of the last warning (see TSI#details). More information in the following TSI: – TSI#logBaseAdrIn – TSI#logBaseAdrOut – TSI#logDiagAdr – TSI#details

Designation	:	Data type	Meaning
TSI#logBaseAdrIn	:	DINT	Logic start address for following *TSI#interruptID* if the alarm was caused by an input range on the module: • _SC_PROCESS_INTERRUPT (= 200) • _SC_DIAGNOSTIC_INTERRUPT (= 201) • _SC_IMAGE_UPDATE_FAILED (= 204) • _SC_IMAGE_UPDATE_OK (= 206) • _SC_PULL_PLUG_INTERRUPT (= 216) • _SC_IO_MODULE_SYNCHRONIZED (= 214) • _SC_IO_MODULE_NOT_SYNCHRONIZED (= 215) Otherwise _SC_INVALID_ADDRESS (= -1)
TSI#logBaseAdrOut	:	DINT	Logic start address for following *TSI#interruptID* if the alarm was caused by an output range on the module: • _SC_PROCESS_INTERRUPT (= 200) • _SC_DIAGNOSTIC_INTERRUPT (= 201) • _SC_IMAGE_UPDATE_FAILED (= 204) • _SC_IMAGE_UPDATE_OK (= 206) • _SC_PULL_PLUG_INTERRUPT (= 216) • _SC_IO_MODULE_SYNCHRONIZED (= 214) • _SC_IO_MODULE_NOT_SYNCHRONIZED (= 215) Otherwise _SC_INVALID_ADDRESS (= -1)
TSI#logDiagAdr	:	DINT	Diagnostics address of a DP slave or I/O device for following *TSI#interruptID* • _SC_DIAGNOSTIC_INTERRUPT (= 201) • _SC_STATION_DISCONNECTED (= 202) • _SC_STATION_RECONNECTED (= 203) • _SC_IMAGE_UPDATE_FAILED (= 204) • _SC_IMAGE_UPDATE_OK (= 206) • _SC_IO_MODULE_SYNCHRONIZED (= 214) • _SC_PULL_PLUG_INTERRUPT (= 216) • _SC_DRIVE_OBJECT_FAULT (= 217) • _SC_DRIVE_OBJECT_ALARM (= 218) Otherwise _SC_INVALID_ADDRESS (= -1)
TST#logDiagAdrIoType		DINT	Specifies the I/O type of a diagnostics address for the following *TSI#interruptID* • DSC_SVS_DEVICE_INPUT (= 198) • DSC_SVS_DEVICE_OUTPUT (= 199)
TSI#details	:	DWORD	Detailed information depending on the TSI#interruptID. See **Meaning of the TSI#details** table
TSI#eventClass	:	UINT	Event class depending on the TSI#interruptID Description in connection with TSI#faultId: See **Significance of the TSI#eventClass and TSI#faultId** table

Designation	:	Data type	Meaning
TSI#faultId	:	UINT	Fault ID depending on the TSI#interruptID Description in connection with TSI#eventClass: See **Significance of the TSI#eventClass and TSI#faultId** table
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SYSTEM_:INTERRUPT

Table 4-56 Significance of TSI#details depending on TSI#interruptID (PeripheralFaultTask)

TSI#interruptID	Significance of TSI#details
_SC_PROCESS_INTERRUPT (= 200)	• Interrupt data of module signaling interrupt The structure of these data is presented in the module manual. • If interrupt source is in a SIMOTION I-slave: Call parameters of the *_sendProcessInterrupt()* system function
_SC_DRIVE_OBJECT_ALARM (= 218)	• 1 = Incoming warning • 0 = Outgoing warning
_SC_DIAGNOSTIC_INTERRUPT (= 201)	DS0 (= byte 0 - 3 of DS1) of module/station signaling interrupt The structure of the DS0 is described in the "Communication" function manual in the section "PROFINET IO - diagnostics and alarm behavior".

TSI#interruptID	Significance of TSI#details	
_SC_PC_INTERNAL_FAILURE (= 205)	The cause of the interrupt is represented as an OR operation to the following causes (sum of hexadecimal values).	
	Value	**Cause**
	16#00000001	"Fatal error" of host operating system (Windows blue screen - SIMOTION P).
	16#00000002	Temperature error (SIMOTION P, SIMOTION D)
	16#00000004	Temperature returned to normal (SIMOTION P, SIMOTION D)
	16#00000008	Battery warning (battery voltage low, but still sufficient - SIMOTION P, SIMOTION D)
	16#00000010	Battery voltage returned to normal (SIMOTION P, SIMOTION D)
	16#00000020	Battery failure (battery voltage too low – SIMOTION P, SIMOTION D)
	16#00000040	Battery module removed (SIMOTION P, SIMOTION D)
	16#00000080	Fan or dual fan failed (SIMOTION P, SIMOTION D)
	16#00000100	UPS in buffer state (SIMOTION P)
	16#00000200	UPS non-chargeable interval expired, PC is booting (SIMOTION P)
	16#00000400	UPS OK again (SIMOTION P)
	16#00000800	UPS battery warning (SIMOTION P)
	16#00001000	UPS battery error (battery no longer operational – SIMOTION P)
	16#00002000	UPS battery warning (SIMOTION P)
	16#00004000	One fan in dual fan failed (SIMOTION P, SIMOTION D)

Table 4-57 Significance of TSI#eventClass and TSI#faultId depending on TSI#interruptID (PeripheralFaultTask)

TSI#interruptID	TSI#eventClass	TSI#faultId	Bus system[1]	Meaning
_SC_PROCESS_INTERRUPT (= 200)	16#11	16#41	PROFIBUS DP PROFINET IO I/O bus	Process alarm
_SC_DIAGNOSTIC_INTERRUPT (= 201)	16#39[2]	16#42	PROFIBUS DP PROFINET IO I/O bus	Incoming diagnostic interrupt
	16#38[3]	16#42	PROFIBUS DP	Last error on the slot has gone
			PROFINET IO I/O bus	Last error on the subslot has gone

TSI#interruptID	TSI#eventClass	TSI#faultId	Bus system[1]	Meaning
_SC_STATION_DISCONNECTED (= 202)	16#39[2]	16#C4	PROFIBUS DP	A DP slave station has failed
		16#CA	PROFINET IO	System error PROFINET IO[4]
		16#CB	PROFINET IO	Station failure of a connected I/O device Shared I-device: No submodule of the I-device is currently subscribed through higher-level I/O controllers.
		16#CC	PROFINET IO	IO device fault present. Channel diagnostics or manufacturer-specific diagnostics pending. **Note:** 16#39:16#CB is always used for station failure and 16#38:16#CC is always used for the arrival with error
		16#F8	PROFINET IO	Partial station failure Shared iDevice: Submodules of the iDevice are partially subscribed to by the higher level IO controller.
_SC_STATION_RECONNECTED (= 203)	16#38[3]	16#C4	PROFIBUS DP	A DP slave station has been reconnected
		16#CB	PROFINET IO	Station reconnection of an IO device with error or warning. Shared iDevice: All submodules of the iDevice are currently subscribed to by the higher level IO controller.
		16#CC	PROFINET IO	Station reconnection of an IO device with error or warning.
		16#CD	PROFINET IO	An I/O device has been reconnected, but error: Set configuration <> actual configuration
		16#CE	PROFINET IO	An I/O device has been reconnected, but error in module parameterization
		16#F8	PROFINET IO	Partial station reconnection Shared iDevice: Submodules of the iDevice are partially subscribed to by the higher level IO controller.

TSI#interruptID	TSI#eventClass	TSI#faultId	Bus system[1]	Meaning
_SC_PULL_PLUG_INTERRUPT (= 216)	16#39[2]	16#51	PROFINET IO	PROFINET IO module has been removed or cannot be addressed.
		16#61	PROFIBUS DP	PROFINET DP module has been removed or cannot be addressed.
		16#54	PROFINET IO	PROFINET IO submodule has been removed or cannot be addressed.
		16#61	PROFIBUS DP	Station is faulted or module has been removed.
	16#38[3]	16#54	PROFINET IO	PROFINET IO module or submodule has been inserted, module type OK (actual configuration = set configuration)
		16#55	PROFINET IO	PROFINET IO module or submodule has been inserted, but wrong module type (actual configuration <> set configuration)
		16#56	PROFINET IO	PROFINET IO module or submodule has been inserted, but error during module parameterization
		16#58	PROFINET IO	I/O status of a module has changed from BAD to GOOD
		16#61	PROFIBUS DP	PROFINET DP module or submodule has been inserted, module type OK
		16#63	PROFIBUS DP	PROFINET DP module or submodule has been inserted, but wrong module type or module inserted

1 Bus system which signals the TSI#interruptID with the specified TSI#eventClass and TSI#faultId

2 Signals incoming event

3 Signals outgoing event

4 Outgoing event (TSI#eventClass = 16#38) is signaled for each existing station as station reconnection. Depending on the error status, the values 16#CB, 16#CD or 16#CE are displayed in TSI#faultId

Process alarms

Process alarms are indicated using the TSI "_SC_PROCESS_INTERRUPT".

Once a module triggers a process alarm, the alarm is read out, the PeripheralFaultTask is called and the alarm is acknowledged. If a process alarm of the same module occurs again during this time, note the following:

- If the process alarm occurs in the same channel that previously triggered a process alarm, then the relevant alarm is lost.
- If the process alarm occurs in a different channel of the same module, then no process alarm can be triggered at that moment. However, this alarm is not lost, but rather is triggered after the first active process alarm has been acknowledged. The behavior is analogous if a process alarm is triggered:
  - On another module of the same station
  - On another module of another station

TaskStartInfo of the TechnologicalFaultTask

Table 4-58 TaskStartInfo of the TechnologicalFaultTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#alarmNumber	:	DINT	Number of the triggered alarm (see description in the SIMOTION Alarms Diagnostics Manual) The parameters output in the alarm message are available in *TSI#alarmP1_DINT to TSI#alarmP5_LREAL* (e.g. *TSI#alarmP3_UDINT* is parameter 3 with data type UDINT).
TSI#toInst	:	ANYOBJECT	TO instance responsible for error, can be converted with the *AnyObject_to_Object* function. Comparison to the instance of the technology object is also possible, see example for using the TaskStartInfo of the TechnologicalFaultTask (Page 1280).
TSI#commandId.low	:	UDINT	CommandId of triggering command (least significant word)
TSI#commandId.high	:	UDINT	CommandId of triggering command (most significant word)
TSI#alarmP1_DINT TSI#alarmP1_UDINT TSI#alarmP1_LREAL	: : :	DINT UDINT LREAL	Parameters 1 to 5 (associated value) of the message for the triggered alarm in the respective data type. Example: *TSI#alarmP3_UDINT* contains parameter 3 with UDINT data type. You can obtain the significance of the parameter from the description of the alarm in the SIMOTION Alarms Diagnostics Manual. This states the number and data type of the parameter with the following syntax: */n/%A* Where: • */n/* : Number of the parameter • *%A* : Abbreviation for the data type – *%d*: DINT – *%X*: UDINT – *%lf* : LREAL Example: */3/%X* means parameter 3 with UDINT data type. Only the parameters documented in the SIMOTION Alarms Diagnostics Manual for the triggered alarm are released for use. For information about process alarms, refer to *Process alarms* below.
TSI#alarmP2_DINT TSI#alarmP2_UDINT TSI#alarmP2_LREAL	: : :	DINT UDINT LREAL	
TSI#alarmP3_DINT TSI#alarmP3_UDINT TSI#alarmP3_LREAL	: : :	DINT UDINT LREAL	
TSI#alarmP4_DINT TSI#alarmP4_UDINT TSI#alarmP4_LREAL	: : :	DINT UDINT LREAL	
TSI#alarmP5_DINT TSI#alarmP5_UDINT TSI#alarmP5_LREAL	: : :	DINT UDINT LREAL	
TSI#AlarmType	:	ENUM...	Alarm type (as of version V4.5 of the SIMOTION kernel) TYPE EnumTechAlarmType FAULT WARNING INFO HIDDEN END_TYPE

Designation	:	Data type	Meaning
TSI#localReaction	:	ENUM...	Local action of the alarm (as of version V4.5 of the SIMOTION kernel) TYPE EnumTechAlarmLocalReaction NONE DECODE_STOP END_OF_MOTION_STOP MOTION_STOP MOTION_EMERGENCY_STOP MOTION_EMERGENCY_ABORT FEEDBACK_EMERGENCY_STOP OPEN_POSITION_CONTROL RELEASE_DISABLE OUTPUT_CAM_DISABLE MEASURING_INPUT_DISABLE FIXED_GEAR_DISABLE SIMULATION_STOP SIMULATION_ABORT ENCODER_DISABLE FOLLOWING_OBJECT_DISABLE DEPENDING_ON_OUTER_INPUT_LIMITCHECK_STATUS HEAT_OUTPUT_ZERO HEAT_AND_COOL_OUTPUT_ZERO DISABLE_MEASURE_AND_AVERAGE_OUTPUT_VALUE DISABLE_MEASURE_AND_MANUAL_OUTPUT_VALUE DISABLE_OUTPUT DISABLE_ALL CAMTRACK_DISABLE DISABLE CONTROL_STOP DISABLE_CONTROLLER FIXED_GEAR_DISABLE STOP_SPECIFIC_FORMULA STOP_ALL_FORMULA CONFIGURED_OUTPUT_VALUE END_TYPE
TSI#Alarmgroup	:	DINT	Alarm group (as of version V4.5 of the SIMOTION kernel)
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SYSTEM_:INTERRUPT

TaskStartInfo of the TimeFaultBackgroundTask

Table 4-59 TaskStartInfo of the TimeFaultBackgroundTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:

Designation	:	Data type	Meaning
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#interruptID	:	UDINT	Triggering event: • _SC_BACKGROUND_TIMER_OVERFLOW (= 300) Additional events are not defined at this time.
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SYSTEM_:INTERRUPT

TaskStartInfo of the TimeFaultTask

Table 4-60 TaskStartInfo of the TimeFaultTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use
TSI#interruptID	:	UDINT	Triggering event: • _SC_CYCLE_TIMER_OVERFLOW (= 301) Additional events are not defined at this time.
TSI#taskId	:	StructTaskId	TaskId of TimerInterruptTask during which the time watchdog responded.
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SYSTEM_:INTERRUPT

TaskStartInfo of the ShutdownTask

Table 4-61 TaskStartInfo of the ShutdownTask

Designation	:	Data type	Meaning
TSI#startTime	:	DT	Start time of the task
TSI#currentTaskId	:	StructTaskId	TaskId of task
TSI#cycleTime	:	TIME	Configured cycle time of task in ms (= 0, because the task is sequential)
TSI#cycleTime_us	:	UDINT	Configured cycle time of task in µs (= 0, because the task is sequential) SIMOTION Kernel as of version V4.4:
TSI#dwuser_1	:	DWORD	Reserved for internal use
TSI#dwuser_2	:	DWORD	Reserved for internal use

Designation	:	Data type	Meaning
TSI#shutDownInitiator	:	UDINT	Initiate transition to STOP: • _SC_MODE_SELECTOR (= 400): Mode selector switch • _SC_DEVICE_COMMAND (= 401): System function in user program • _SC_EXTERNAL_COMMAND (= 402): Command in SIMOTION SCOUT • _SC_EXCEPTION (= 403): Exception • _SC_ALARM_CONFIGURATION (= 404) Configured response to technological alarm Example of exceptions: • Execution errors in programs • Level overflow • Withdrawing a central module in RUN mode
TSI#taskType	:	ENUM...	Type of task As of version V4.5 of the SIMOTION Kernel Data type: EnumTaskType, see StartupTask (Page 1263) Value: SHUTDOWN

Example for using the TaskStartInfo of the TechnologicalFaultTask

The following example shows you how to query the TO instance that initiated the alarm and the alarm number using the Taskstartinfo in the TechnologicalFaultTask. The *TO_AlarmProg* program must therefore be assigned to the TechnologicalFaultTask.

Table 4-62 Example of polling for Taskstartinfo in the TechnologicalFaultTask

```
PROGRAM TO_AlarmProg
    VAR
        dintVar  : DINT;
        dtVar    : DT;
    END_VAR;
    dtVar   := TSI#startTime;
    dintVar := TSI#alarmNumber;
    IF TSI#toInst = axis_1 THEN // Instance of the technology object
        ; // Statements
    END_IF;
    IF TSI#alarmNumber = 30002 THEN    // Triggering alarm
        ; // Statements
   END_IF;
END_PROGRAM
```

For an additional example, refer to Evaluating in the user program (Page 1293).

4.2.6 Error Handling in Technology Objects

4.2.6.1 Possible errors in technology objects

The following basic errors are possible in the programming of technology objects:

- The technology object itself cannot execute the function required by the application or reports certain events or states:
  → A **technological alarm** is output.
  You can find information on the individual alarms in the SIMOTION Reference Lists.
- The command issued to a technology object cannot be executed:
  → The **return value** of the command provides information about the cause.
  You can find information on the return values of the commands in the SIMOTION Reference Lists.
- Error while accessing configuration data, system variables or I/O variables
  The ExecutionFaultTask is called in the event of errors when configuration data or variables are being read or written

See also

Process Alarms (Page 1281)

Return values of commands (Page 1294)

Errors when accessing system data with _get/_setSafeValue (Page 1295)

4.2.6.2 Process Alarms

If an event (error, note) occurs on a technology object, the object issues a **technological alarm**.

A maximum of 160 technology object alarms can be stored in the system at one time. In this regard per TO identical alarms are counted as one alarm. If this buffer overflows because more than 160 alarms are pending simultaneously, the system switches to STOP mode with a **Message buffer overflow** diagnostics entry. The user cannot read out the buffer level.

If a series of the same alarms occur in succession, the detailed information will only be displayed on the first alarm. For the subsequent alarms the information will only be displayed when the first alarm that occurred has been acknowledged.

Effects of alarms

Technological alarms cause subsequent responses in the system. A distinction is made between:

- Effects on the affected technology object itself: **Local response**
- Effects on other technology objects or the execution system: **Global response**

For each alarm, certain effects have a default setting. However, you can adapt these settings to suit your requirements.

- By specifying the **error activation**, you can define whether the alarm is to be activated immediately, after repeated occurrences of an error, or after a certain period of time.
- You can hide some alarms. This allows you, for example, to suppress unimportant information.

Alarm group association

TO alarm numbers are assigned to an alarm group. Some alarm groups are defined for each TO type, whereas others are TO-specific. For example, alarm numbers 20006 and 20011 are assigned for a TO from the "Configuration error" alarm group.

The alarm group association of the pending technological alarms is displayed via the "errorGroup" system variable. A bit of this variable is assigned to each alarm group. Bit 1 (the lowest is bit 0) is assigned to the "Configuration error" alarm group. If bit 1 is set, this means there is an alarm from the "Configuration error" group at the TO. The following table lists the various alarm groups.

Description	Bit number	Meaning
SYSTEM_FAULT	0	System error
CONFIG_FAULT	1	Configuration error
USER_FAULT	2	User error
PERIPHERAL_FAULT	3	I/O error
FUNCTION_FAULT	4	Function/command error
FUNCTION_ABORTED	5	Function/command abort
RESET_RESTART_FAULT	6	Reset/restart error
DISTRIBUTED_MOTION_FAULT	7	Error during distributed synchronous operation
FOLLOWING_ERROR	8	Dynamic following error monitoring
STANDSTILL_POSITIONING_ERROR	9	Standstill/positioning monitoring error
DYNAMIC_LIMIT	10	Limitation of velocity, acceleration or jerk
CLAMPING_ERROR	11	Clamping monitoring error
SOFTWARE_LIMIT	12	Software limit position switch
LIMIT_SWITCH	13	Hardware limit position switch
SENSOR_FAULT	14	Encoder error
REFERENCE_NOT_FOUND	15	Homing output cam/zero mark error
OUTPUT_LIMIT	16	Output variable limitation
FORCE_DYNAMIC_LIMIT	17	Limitation of force and/or pressure
ADDITIONAL_SENSOR_FAULT	18	Error on additional encoder
SYNCHRONOUS_MOTION_FAULT	19	Synchronous operation error
FOLLOWING_OBJECT	20	Following object error
PATH_SYNCHRONOUS_MO-TION_FAULT	21	Synchronous path operation error
PATH_MOTION_FAULT	22	Path motion error
PATH_OBJECT	23	Path object error

See also

Local response (Page 1283)

Global response (Page 1283)

Error activation (Page 1284)

Configure technological alarms (Page 1285)

Displaying and acknowledging technological alarms (Page 1287)

Acknowledging via the user program (Page 1288)

Evaluating in the user program (Page 1293)

Local response

With the *Local response* setting, you specify how the relevant technology object is to react when the alarm occurs and how subsequent commands for this TO will be handled.

Alarm responses are prioritized. Only one response can be in progress at any given time. This is the response with the highest priority of all the alarms pending at that particular point in time. When an alarm response occurs (except for NONE), the command decoder is always stopped. Any programmed commands issued subsequently are rejected. Command execution can continue after the alarm has been acknowledged in cases where the global error response for the alarm does not automatically require a Power ON.

You can select specific response options depending on the individual technology object and technology object alarm.

For information on the specific local response, please refer to the function manuals of the respective technology objects.

Global response

The *global response* describes the effect of a technology object alarm on the execution system. You can select the following response options depending on the respective TO alarm.

- **NONE**
  System does not respond when the alarm occurs.
- **START TechnologicalFaultTask**
  The **TechnologicalFaultTask** is started. Programs assigned to this task are started. This provides the user with the option of programming an application-specific response to the TO alarm. If there is no program assigned to this task, the system switches to STOP mode.
- **STOP**
  The system switches to **STOP** mode. In this mode, all technology objects are inactive, the user program is not executed, and all outputs are at zero. All system services are still active, and user programs can be loaded.
- **STOP U**
  The system switches to **STOP U** mode. The program execution is terminated, the axis enable and thus the position control is deactivated. The technology objects also remain active and can still process requests for testing and commissioning functions. Otherwise, identical to STOP mode.

Error activation

Type of error activation (TypeOfActivation)

Alarms and their associated alarm responses can be triggered in a variety of ways. Depending on the error type, the error activation is preset to one of the following values.

Table 4-63 Type of error activation

TypeOfActivation parameter	Meaning
ACTIVATE_IMMEDIATELY	The alarm is activated immediately when the error occurs.
ACTIVATE_AFTER_NTIME	The alarm is activated after the error occurs n times. The number n of error repetitions is set in the Continue parameter, see Error reproducibility table.
ACTIVATE_AFTER_NTIME_CONSECUTIVE	The alarm is activated after time t. The error must be pending without interruption over the entire time t. The time is set in the Time parameter, see Error response time table.

Error repeatability (Continue)

Table 4-64 Error repeatability

Continue parameter	Meaning
NO_TIME	The alarm is activated immediately when the error occurs.
TIME_n	The alarm is activated after the error occurs n times.

This parameter is relevant only if the **TypeOfActivation** parameter is set to ACTIVATE_AFTER_NTIME.
TIME_n can be set to values TIME_10, TIME_100, and TIME_1000, depending on the alarm.

Error response time (Time)

Table 4-65 Error response time

Time parameter	Meaning
NO_TIME	The alarm is activated immediately when the error occurs.
TIME_n	The alarm is triggered after a temporally uninterrupted error presence of n milliseconds.

This parameter is relevant only if the **TypeOfActivation** parameter is set to ACTIVATE_AFTER_NTIME_CONSECUTIVE.
TIME_n can be set to values TIME_1, TIME_10, TIME_100, and TIME_1000, depending on the alarm.

Type

You can set the **Type** parameter to specify whether certain technology object alarms are to be displayed. If you select the *Hidden* setting, an alarm message is not displayed when this alarm occurs and no entry is written to the diagnostics buffer. You can thereby prevent an overflow of the alarm buffer when a particular technology alarm is issued frequently, or suppress a note message that is not important to you.

Configure technological alarms

Individual responses are preset for each alarm. To change these default settings, proceed as follows:

1. Select the **Execution System** path in the project navigator. The **Execution System** window opens. Select the path **SystemInterruptTasks → TechnologicalFaultTask**. Then click the **Alarm configuration** button in the window.

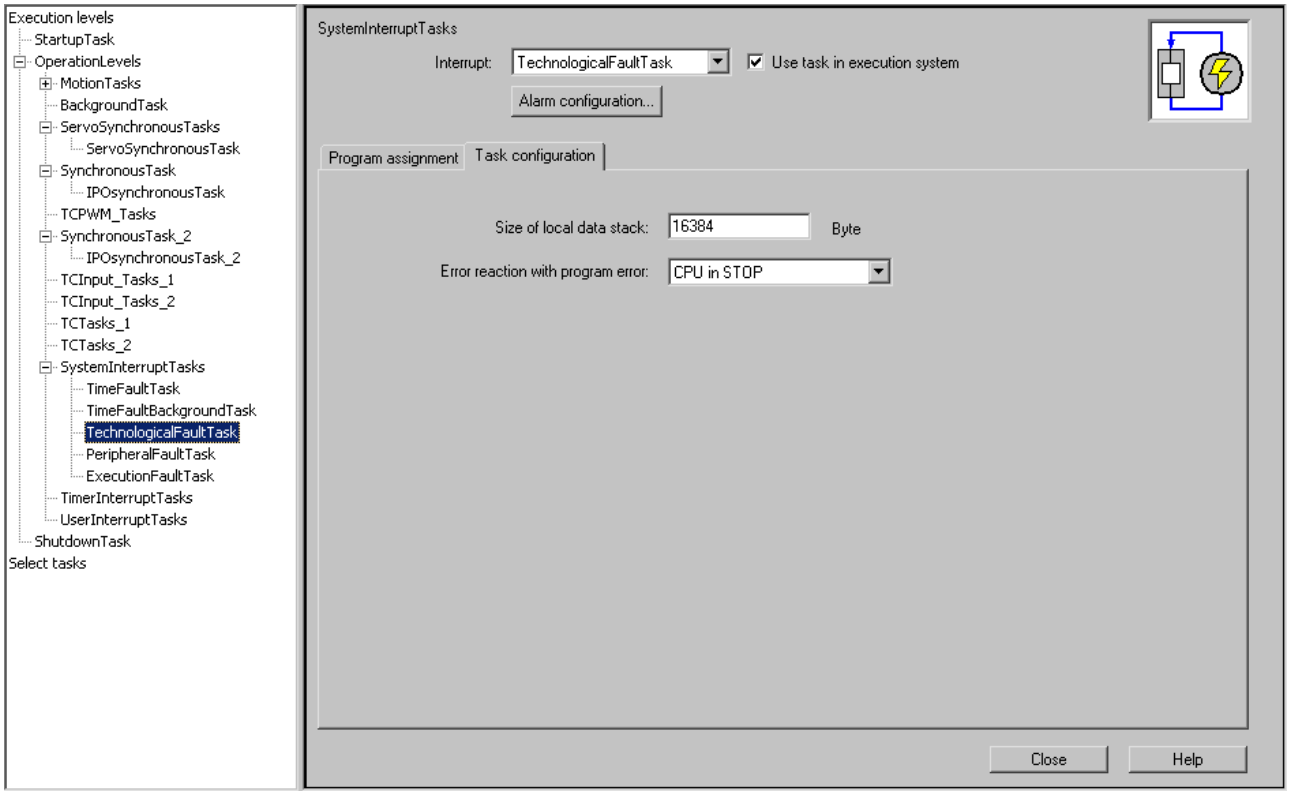


Figure 4-82 Configuring a technological alarm

2. In the combo box, select the technology object for which you want to configure alarms. The alarms for the technology object are displayed.

3. Select the alarm for which you want to change the response.
4. Select the required response from the list for the corresponding technology object alarm. The options available depend on the type of alarm.

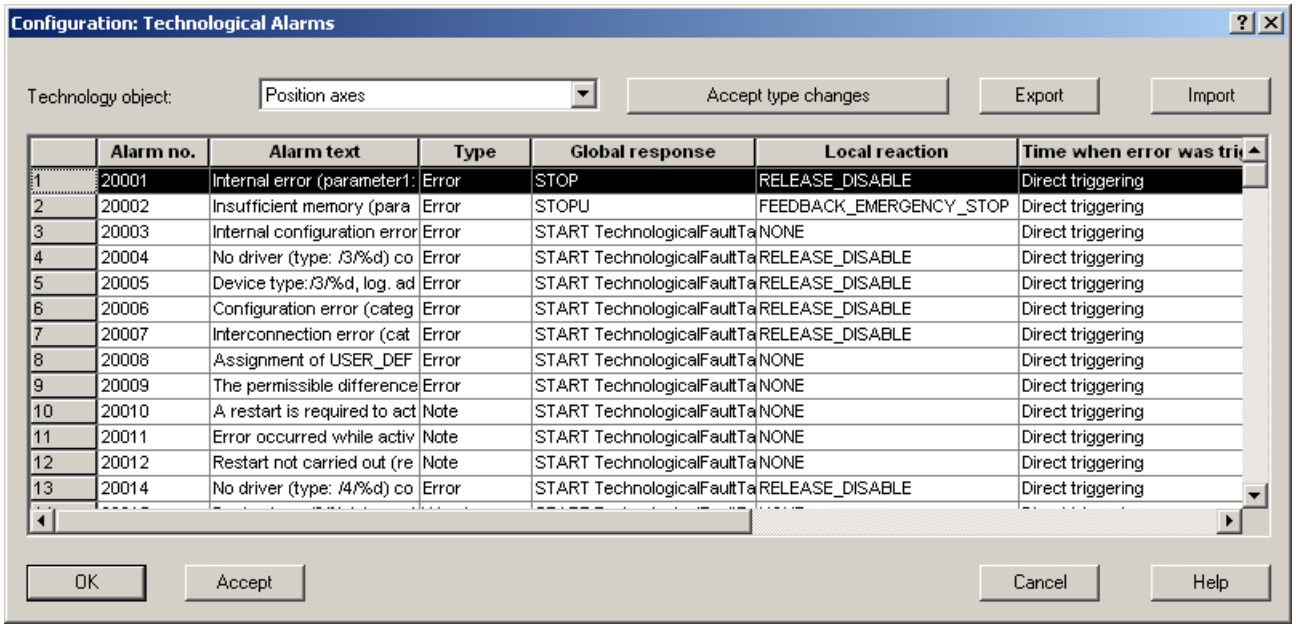


Figure 4-83 Selecting a technological alarm

Note

The technology object whose alarms you want to configure must already be configured.

The alarm configuration can be transferred to other technology objects via the corresponding buttons in the **Alarm configuration** dialog (via export/import).

Exporting or importing technological alarms

Note

So that all changes that you have made in the dialog **Configuration: Technological alarms** are exported, the button **Export** remains inactive until you have applied all changes using **Apply**.

To export all TO alarms in XML format, proceed as follows:

1. Open the **Configuration: Technological alarms** dialog.
2. Click on export to select the **Export alarm configuration** dialog.
3. Select a path for the export and confirm with **OK**.
   The alarms are stored there.

To import TO alarms in XML format, proceed as follows:

1. In the **Configuration: Technological alarms** dialog click on Import.
   This displays the **Import alarm configuration** dialog.
2. Under **Source path and source name of the import** select the desired XML file.
3. Click **OK** to import the data.
   The alarms will then be displayed in the dialog.

Configuring messages (alarms) for all TOs of a certain TO type

If several technology objects of the same TO type are configured on the SIMOTION device, you can assign one alarm configuration to all TOs of this type. For example you can assign a configuration to all position axes.

1. Open the **Configuration: Technological alarms** dialog.
2. Select the TO type from the list, e.g. **- All positioning axes**.
3. Click on **OK** to assign the settings to all preciously configured TOs of the selected type.

Displaying and acknowledging technological alarms

Technological alarms can be evaluated and acknowledged in different ways:

- When SIMOTION SCOUT is in **online mode**, alarms and messages are displayed on the **Alarms** tab in the detail view of the workbench.
  With **Acknowledge**, all alarms of the associated type are deleted.
- Alarms can be output, displayed, and acknowledged via the **H**uman **M**achine **I**nterface (HMI).
- All pending or individually selected alarms of a technology object can also be queried, evaluated, and acknowledged via the user program.

Note

If during the acknowledgement of TO alarms, a new alarm occurs at the TO with the same or lower priority, then the new alarm is no longer reported in the system. This behavior must be taken into account during error acknowledgement.

Acknowledging via SIMOTION SCOUT

Acknowledging individual alarm:

1. Select the alarm in the **Alarms** tab of the detail view.
2. Click **Acknowledge**.

Acknowledging all alarms:

- Click on **Acknowledge all**.

All alarms of the associated type are then deleted.

Note

Because drive alarms usually generate technology object alarms as well, the drive alarms are also deleted with the **Acknowledge (TO)** switch. If however the cause of a drive alarm still exists then a new TO alarm will be triggered immediately. In this case first correct the cause of the drive alarm.

Display and acknowledge via HMI

1. Connecting via WinCC flexible
   Alarms are displayed in a configured message line or in message windows.
   WinCC devices are acknowledged via the ACK key or via user-configured softkeys or buttons.
   (For operating instructions, see **WinCC flexible** description)
2. Link via OPC
   Alarms can be displayed and acknowledged in SimaticNet as of V6.0 SP4 OPC Alarms & Events. In addition the diagnostic buffer can be read out and acknowledged if necessary.
   (For handling, see the *SIMOTION IT Diagnostics and Configuration* Function Manual which you can find on the SIMOTION SCOUT CD documentation)

Acknowledging via the user program

Acknowledge all pending technology object alarms

_resetTechnologicalErrors → Acknowledge all currently pending technology object alarms

ST call example: Acknowledge all pending alarms

```
INTERFACE
    USEPACKAGE CAM;
    PROGRAM EXAMPLE;
END_INTERFACE

IMPLEMENTATION
    PROGRAM EXAMPLE
        VAR
            s_i_RetVal : DINT;
        END_VAR;
        (* Acknowledge all TO alarms *)
        s_i_RetVal := _resetTechnologicalErrors();
    END_PROGRAM
END_IMPLEMENTATION
```

MCC call: Acknowledge all pending alarms

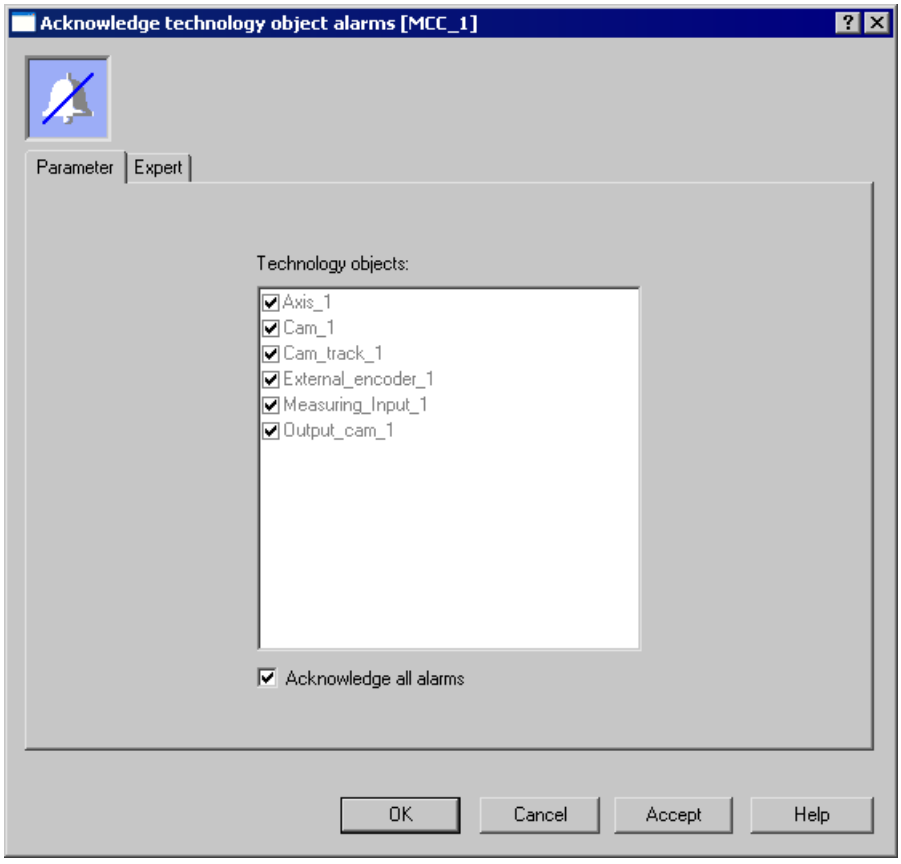


Figure 4-84 MCC call: Acknowledge all pending alarms

Acknowledge all pending alarms of a technology object

ST call example: Acknowledge all alarms on a TO axis, TO measuring input and TO output cam

```
INTERFACE
    USEPACKAGE CAM;
    PROGRAM EXAMPLE;
END_INTERFACE

IMPLEMENTATION
    PROGRAM EXAMPLE
        VAR
            s_i_RetVal : DINT;
        END_VAR;
        (* Acknowledge TO alarms ('ResetTOAlarms') *)
        s_i_RetVal := _resetAxisError(axis := Axis_1);
        s_i_RetVal := _resetMeasuringInputError(
                measuringInput := Measuring_input_1);
        s_i_RetVal := _resetOutputCamError(
                outputCam := Output_cam_1);
    END_PROGRAM
END_IMPLEMENTATION
```

MCC call: Acknowledge all alarms on a TO axis, TO measuring input and TO output cam

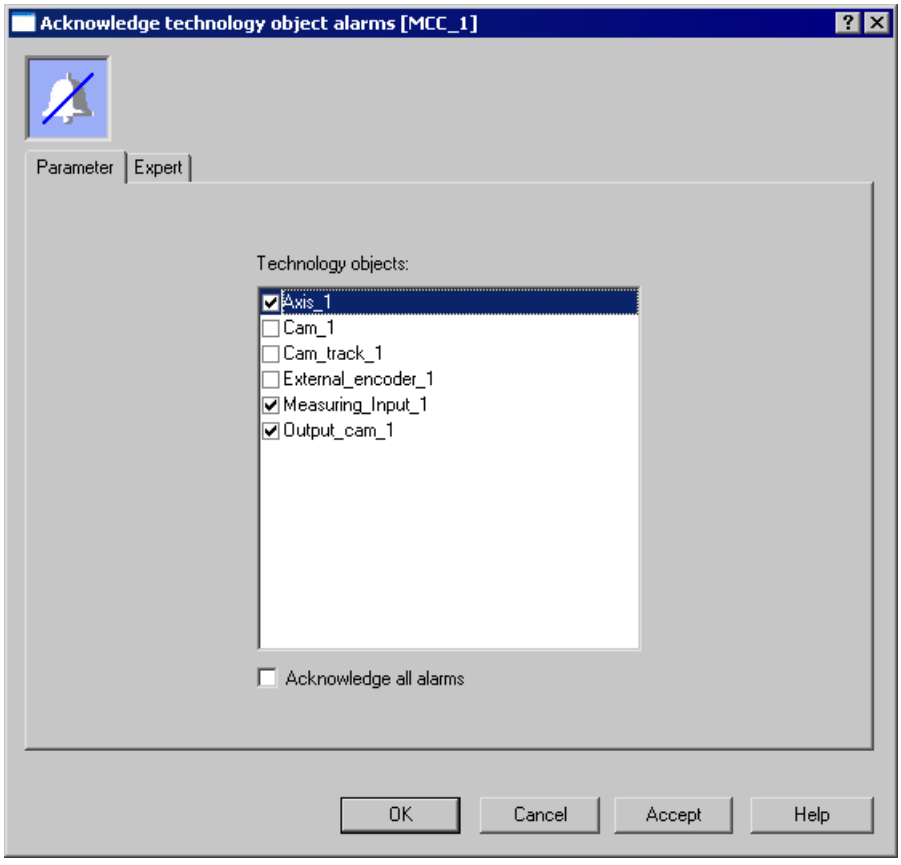


Figure 4-85 MCC call: Acknowledge all alarms on a TO axis, TO measuring input and TO output cam

Acknowledging a specific alarm of a technology object

By adding **errorResetMode** := SPECIFIC_ERROR and **errorNumber** := XXXXX to the **_reset..** commands, it is possible to specifically acknowledge a certain alarm.

Example:

_resetAxisError (..., **errorResetMode** := SPECIFIC_ERROR, **errorNumber** := 30002)

ST call example: Acknowledge alarm 30002 on a TO axis

```
INTERFACE
    USEPACKAGE CAM;
    PROGRAM EXAMPLE;
END_INTERFACE

IMPLEMENTATION
    PROGRAM EXAMPLE
        VAR
            s_i_RetVal : DINT;
        END_VAR;
        (* Acknowledge specific TO alarm ('ResetSingleTOAlarm') *)
        s_i_RetVal:= _resetAxisError(axis := Axis_1,
                errorResetMode:= SPECIFIC_ERROR,
                errorNumber := 30002);
    END_PROGRAM
END_IMPLEMENTATION
```

MCC call: Acknowledge alarm 30002 on a TO axis

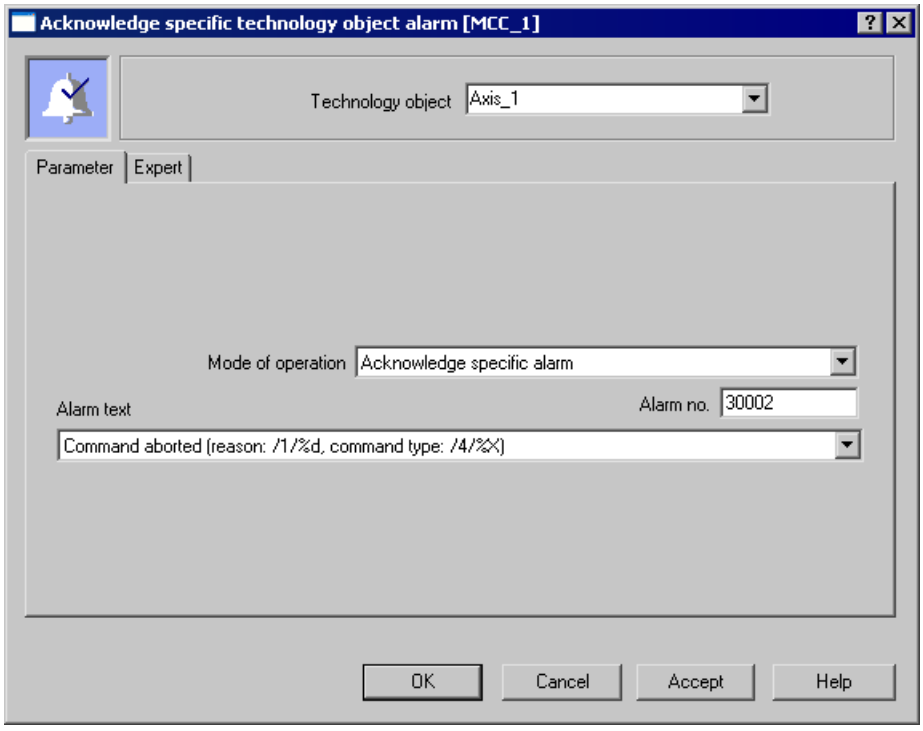


Figure 4-86 MCC call: Acknowledge alarm 30002 on a TO axis

Resetting a technology object

The technology object is set to its initial state and all pending alarms are acknowledged.

Note

In addition to troubleshooting the commands for reset have other effects on the TO as well. Therefore, as a general rule, errors should be acknowledged with the **_reset...Error** commands.

In contrast to the **_reset...Error** command, the **_reset...** commands also bring the respective TO to a safe state. In addition to acknowledging the alarms, the following actions are performed for each TO type:

- Stop the active commands
- Clear the command buffer
- Reset system variables (parameter userDefaultData)
- Execute a TO restart (parameter activateRestart)

In addition, actions dependent on the TO type are executed:

- Axes: Generate a braking ramp
- Output cam and cam track: Switch off the hardware output
- Cam: Delete the curve geometry
- Path object: Stop the path group

Call example: Resetting TO axis, TO measuring input and TO output cam

```
INTERFACE
    USEPACKAGE CAM;
    PROGRAM EXAMPLE;
END_INTERFACE

IMPLEMENTATION
    PROGRAM EXAMPLE
        VAR
            s_i_RetVal : DINT;
        END_VAR;
        (* Reset object ('ResetObject') *)
        s_i_RetVal := _resetAxis(
                axis := Axis_1,
                userDefaultData := DO_NOT_CHANGE);
        s_i_RetVal := _resetMeasuringInput(
                measuringInput := Measuring_input_1,
                userDefaultData := DO_NOT_CHANGE);
        s_i_RetVal := _resetOutputCam(
                outputCam := Output_cam_1,
                userDefaultData := DO_NOT_CHANGE);
    END_PROGRAM
END_IMPLEMENTATION
```

Evaluating in the user program

In the case of alarms for which the **StartTechnologicalFaultTask** global response is configured, the **TechnologicalFaultTask** is called once with every alarm occurrence. The number of the pending alarm and the technology object triggering the alarm can be scanned in this task. The information is passed to the **TechnologicalFaultTask** via the task start info (TSI).

You can add a program to the **TechnologicalFaultTask** and thus program an individual error response for specific alarms or, for example intercept alarms and forward them to higher-level alarm evaluation.

Note

If you have not added any program to the **TechnologicalFaultTask**, the CPU will switch to **STOP** mode if the task is called by an alarm.

The following parameters are transferred in **TechnologicalFaultTask** for evaluation:

TSI#startTime	→	time at which the alarm was registered
TSI#alarmNumber	→	alarm number
TSI#toInst	→	name of the technology object that triggered the alarm (e.g. Axis_1)

Programming example

Every time **Alarm 30002** occurs on Axis_1, a counter (**s_i_Count**) is to be incremented and the alarm acknowledged automatically.

Note: This sample program must be added to the **TechnologicalFaultTask** in the execution system!

```
UNIT ST_1;
INTERFACE
  USEPACKAGE CAM;
  PROGRAM TO_AlarmProg;
END_INTERFACE

IMPLEMENTATION
  PROGRAM TO_AlarmProg
    VAR
      s_i_Count : INT;
      s_i_RetVal: DINT;
    END_VAR;

    (* Query whether alarm 30002 is pending for Axis_1 *)
    IF (TSI#alarmNumber = 30002) AND
                        (TSI#toInst = Achse_1) THEN
      (* Increment counter *)
      s_i_Count:= s_i_Count + 1;
      (* Acknowledge specific technology object alarm *)
      s_i_RetVal := _resetAxisError (
                        axis:=Achse_1,
                        errorResetMode := SPECIFIC_ERROR,
                        errorNumber := 30002);
    END_IF;
  END_PROGRAM
END_IMPLEMENTATION
```

4.2.6.3 Return values of commands

The user program indicates whether or not an executed command was able to be executed successfully during runtime.

Any error information is contained in the return value of the block. You do not have to acknowledge this error information.

Evaluating the return value

When a command returns a value of zero, this signifies that the command has been executed without error. If an error has occurred, the return value contains an error number that can be used to identify the cause of the error.

The possible error numbers for the respective commands can be found in the SIMOTION Reference Lists.

You can respond to possible errors during system function execution in the user program, thus preventing secondary errors.

Likewise for MCC commands you can program a specific error response by entering a return variable with the return value of the command (see MCC Programming Manual, section Expert tab).

Call example

If an error occurs when the **_pos** command is being executed, the **g_bo_error** variable is to be set to **true**, the return value entered in the **g_i_errornumber** parameter and the block aborted.

Note: This sample program must be added to a **MotionTask** in the execution system!

```
UNIT ST_1;
INTERFACE
    USEPACKAGE CAM;
        VAR_GLOBAL
            g_bo_error: BOOL;
            g_i_errornumber: DINT;
            g_i_RetVal: DINT;
        END_VAR
    PROGRAM RETURN_VALUE;
END_INTERFACE
IMPLEMENTATION
    PROGRAM RETURN_VALUE

         (* Position axis ('Pos') *)
        g_i_RetVal:= _pos(axis:=Axis_1,
        direction:=SHORTEST_WAY,
        positioningMode:=ABSOLUTE,
        position:=222,
        velocityType:=DIRECT,
        velocity:=1000,
        velocityProfile:=TRAPEZOIDAL,
        blendingMode:=INACTIVE,
        mergeMode:=IMMEDIATELY,
        nextCommand:=WHEN_MOTION_DONE,
        commandId:=_getCommandId());
         (*Evaluation of the return value *)
        IF g_i_RetVal <> 0 THEN
            g_i_errornumber := g_i_RetVal;
            g_bo_error := true;
            Return;  // End block
        END_IF;
    // Further user program as of here.
    END_PROGRAM
END_IMPLEMENTATION
```

4.2.6.4 Errors when accessing system data with _get/_setSafeValue

Error while accessing configuration data, system variables or I/O variables

With V4.1.3 and higher, these system functions are not necessarily required for system variables and config data. In the event of access errors (e.g. technology object in restart), a "replacement value" or "last value" can be configured; see System variables (Page 1245) or Configuration data (Page 1249).

Where direct access is involved, ExecutionFaultTask is called in the event of errors occurring when configuration data or system variables are being read or written (see Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258)).

However, in certain cases it is necessary to avoid calling the ExecutionFaultTask, to respond differently to an error or to deviate from the configured error response. The functions _getSafeValue, _setSafeValue, and getInOutByte serve this purpose. Configuration data item *restartInfo.behaviorInvalidSysvarAccess* can be used for accessMode = CONFIGURED (see System variables (Page 1245) or Configuration data (Page 1249)).

Replacement value strategy and errors when accessing system variables, configuration data, and I/O variables

There are various situations in which the reading or writing of a system variable, a configuration data item or an I/O variable can fail.

Table 4-66 Possible causes for the read/write failure

Cause	System variable	Config data item	I/O variable
Variable temporarily not available	TO in restart TO deactivated	TO in restart TO deactivated	Faulty input
	-	Data not visible in the current configuration	Faulty output
Illegal value (for write)	Value lies outside the limits	Value lies outside the limits	-
	Value not in the grid	Value not in the grid	-

Depending on the cause, the following responses are possible:

Table 4-67 System response in the event of an error when using _getSafeValue

Required response		System variable	Config data item	I/O variable
A value that temporarily cannot be accessed while reading:				
Go to STOP	Call = accessMode	STOP_DEVICE	STOP_DEVICE	STOP_DEVICE
	Answer	→ STOP	→ STOP	→ STOP
	Value	Unchanged	Unchanged	Unchanged
Do not read	Call = accessMode	NO_CHANGE	NO_CHANGE	Not implemented
	Answer	NO_CHANGE	NO_ACCESS	
	Value	Last value	Configured value	
Read substitute value	Call = accessMode	DEFAULT_VALUE	DEFAULT_VALUE	DEFAULT_VALUE
	Answer	NO_ACCESS	INVALID_VALUE	DEFAULT_VALUE
	Value	Default value	Default value	Substitute value
Read last valid value	Call = accessMode	CONFIGURED	CONFIGURED	NO_CHANGE
	Answer	(In acc. with configuration data item behavior, etc.)	(In acc. with configuration data item behavior, etc.)	NO_CHANGE
	Value			Last valid value

Table 4-68 System response in the event of an error when using _setSafeValue - when writing a temporarily inaccessible value

Required response		System variable	Config data item	I/O variable
Go to STOP	Call = accessMode	STOP_DEVICE	STOP_DEVICE	STOP_DEVICE
	Answer	→ STOP	→ STOP	→ STOP
	Value	Unchanged	Unchanged	Current value
	Return value setValue	Current value	Current value	-
Do not write	Call = accessMode	NO_CHANGE	NO_CHANGE	Not implemented
	Answer	NO_CHANGE	NO_ACCESS	
	Value	Unchanged	Unchanged	
	Return value setValue	Current value	Current value	
Do not write	Call = accessMode	DEFAULT_VALUE	DEFAULT_VALUE	Not implemented
	Answer	NO_ACCESS	NO_ACCESS	
	Value	Unchanged	Unchanged	
	Return value setValue	Current value	Current value	
Do not write	Call = accessMode	CONFIGURED	CONFIGURED	Not implemented
	Answer	(In acc. with configuration data item behavior, etc.)	(In acc. with configuration data item behavior, etc.)	
	Value			
	Return value setValue	Current value	Current value	
Accept substitute value, will take effect later	Call = accessMode	Not implemented	Not implemented	DEFAULT_VALUE
	Answer			DEFAULT_VALUE
	Value			Substitute value
Accept current value, will take effect later	Call = accessMode	Not implemented	Not implemented	NO_CHANGE
	Answer			NO_CHANGE
	Value			Current value

Table 4-69 System response in the event of an error when using _setSafeValue - error when writing an invalid value

Required response		System variable	Config data item	I/O variable
Go to STOP	Call = accessMode	STOP_DEVICE	STOP_DEVICE	Does not occur
	Answer	→ STOP	→ STOP	
	Value	Unchanged	Unchanged	
	Return value setValue	Current value	Current value	
Do not write	Call = accessMode	NO_CHANGE	NO_CHANGE	
	Answer	NO_CHANGE	NO_CHANGE	
	Value	Unchanged	Unchanged	
	Return value setValue	Current value	Current value	
Write default value	Call = accessMode	DEFAULT_VALUE	DEFAULT_VALUE	
	Answer	DEFAULT_VALUE	INVALID_VALUE	
	Value	DEFAULT_VALUE	Unchanged	
	Return value setValue	Current value	Current value	

Legend for tables above

- DEFAULT_VALUE
  The DEFAULT_VALUE is the configured value. This is the value transferred during a download (system variables and config data).
- LAST VALUE - configured value (config data)
  The last value can be the configured value. The value will then be equivalent to the value shown under "Current value" in the expert list.
  OR
  The last value can be the last configured value. The value will then be equivalent to the value shown under "Next value" in the expert list.
  Either of these values can be output, depending on the configuration data entered.
- LAST VALUE - last value (system variables)
  With system variables, the last value is the value which is currently set and effective.

See also

System variables (Page 1245)
_getSafeValue function (Page 1514)
_setSafeValue function (Page 1517)
_getInOutByte function (Page 1520)
Configuration data (Page 1249)

4.2.7 Execution System, Tasks, and System Cycle Clocks

4.2.7.1 Execution system

The SIMOTION **execution system** provides various **execution levels**.

- The execution levels are assigned to **user program tasks**.
  - **Programs** are assigned to the user program tasks.

  All programs - and thus also tasks - can contain PLC and motion control tasks.

The following execution levels are available:

- Synchronous execution levels: synchronous with control and interpolator cycle clocks
- Time-driven execution levels
- Event-driven execution levels
- Interrupt-controlled execution levels
- Sequential execution levels
- Free-running execution levels

Various **tasks** with different execution properties are available for specific tasks:

System tasks

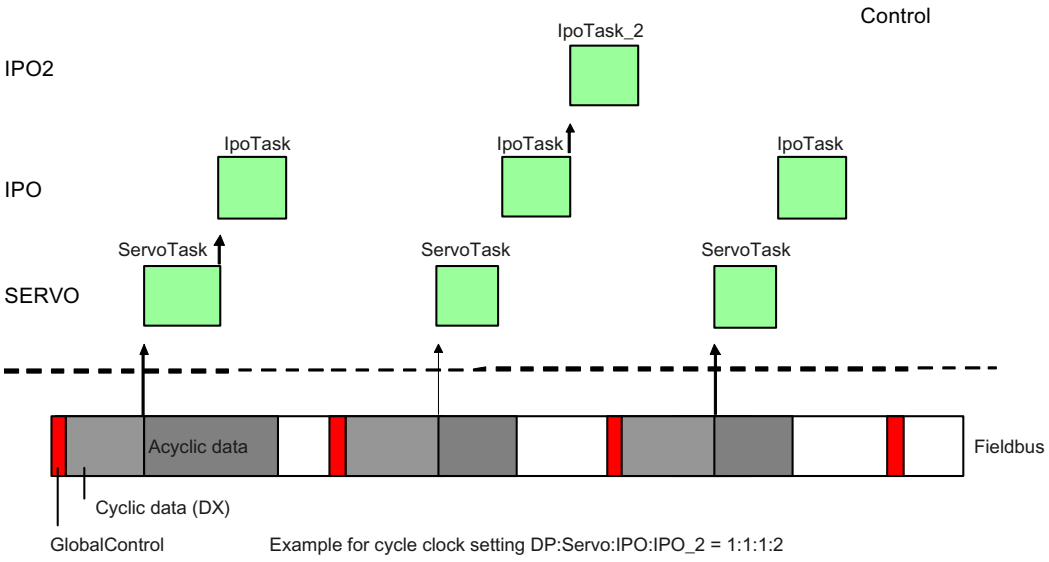
System tasks are regularly executed by the system.

The system cycle clock can be specified.

The following tasks are executed by system tasks:

- **Communication**
  System tasks for the connection to the isochronous PROFIBUS, to PROFINET IO with IRT or the system cycle clock and for I/O processing.
- **Motion control**
  - Base cycle clock/bus cycle clock
    The base/bus cycle clock (DP cycle clock or PN cycle clock) is based on the cycle of the isochronous bus and determines the intervals for data exchange with the DP/PN I/O. The base/bus cycle clock is used as the basis for setting further cycle clocks.
  - Position control cycle clock
    Among other things, position control and monitoring of axes, drive communication, and I/O processing take place during the time slice of the position control cycle clock. The cycle for the position control cycle clock determines the interval between two position controller cycle clocks. The position control cycle clock can be set as a multiple of the base/bus cycle clock. A cycle clock ratio of 1:1 is recommended.
  - IPO cycle clock
    The time slice of the interpolator cycle clock is used, amongst other things, to calculate the setpoints. The cycle for the interpolator cycle clock determines the interval between two interpolator cycle clocks. Time slice requirements may increase briefly at the start of one or more commands. The IPO cycle clock can be set as a multiple of the position control cycle clock. A cycle clock ratio of 1:1 is recommended.
  - IPO2 - cycle clock 2
    Interpolator cycle clock 2 handles the same tasks as the interpolator cycle clock and can be used for technology objects of lower priority classes. The IPO2 cycle clock can be set as a multiple of the IPO cycle clock.
  - Servo_fast/IPO_fast
    An additional faster position control cycle clock option and an IPO option can be configured as of V4.2, see also Servo_fast (application cycle clock for fast bus system) (Page 1364).

  Adjustable gear ratios between bus task, servo, and IPO support appropriate load distribution and optimum system utilization.
  With digital drives, these execution levels are synchronized with the isochronous PROFIBUS or PROFINET IO with IRT. For analog drives without isochronous PROFIBUS, the execution levels are supplied with an adjustable system cycle clock from an internal timer.



- **Temperature control**
  In connection with the **TControl** technology package, execution levels are available for the temperature control: Actual value measurement, control and pulse width modulation of the output signals.

User programming

Execution levels for the task-related programming are available for the user programming (**user program tasks**): Motion control, logic and technological functions.

See also

Specifications for the configuring (Page 1417)

Execution levels / tasks

The execution levels define the chronological sequence of programs in the execution system. Each execution level contains one or more tasks.

A task provides the execution framework for the programs. Each task is executed when specific conditions are satisfied. You can assign one or more user programs to each task and specify their order within the task.

Besides user program tasks there are also several system tasks, the contents and execution sequence of which you cannot influence.

Execution levels

The following figure shows the execution levels with their tasks.

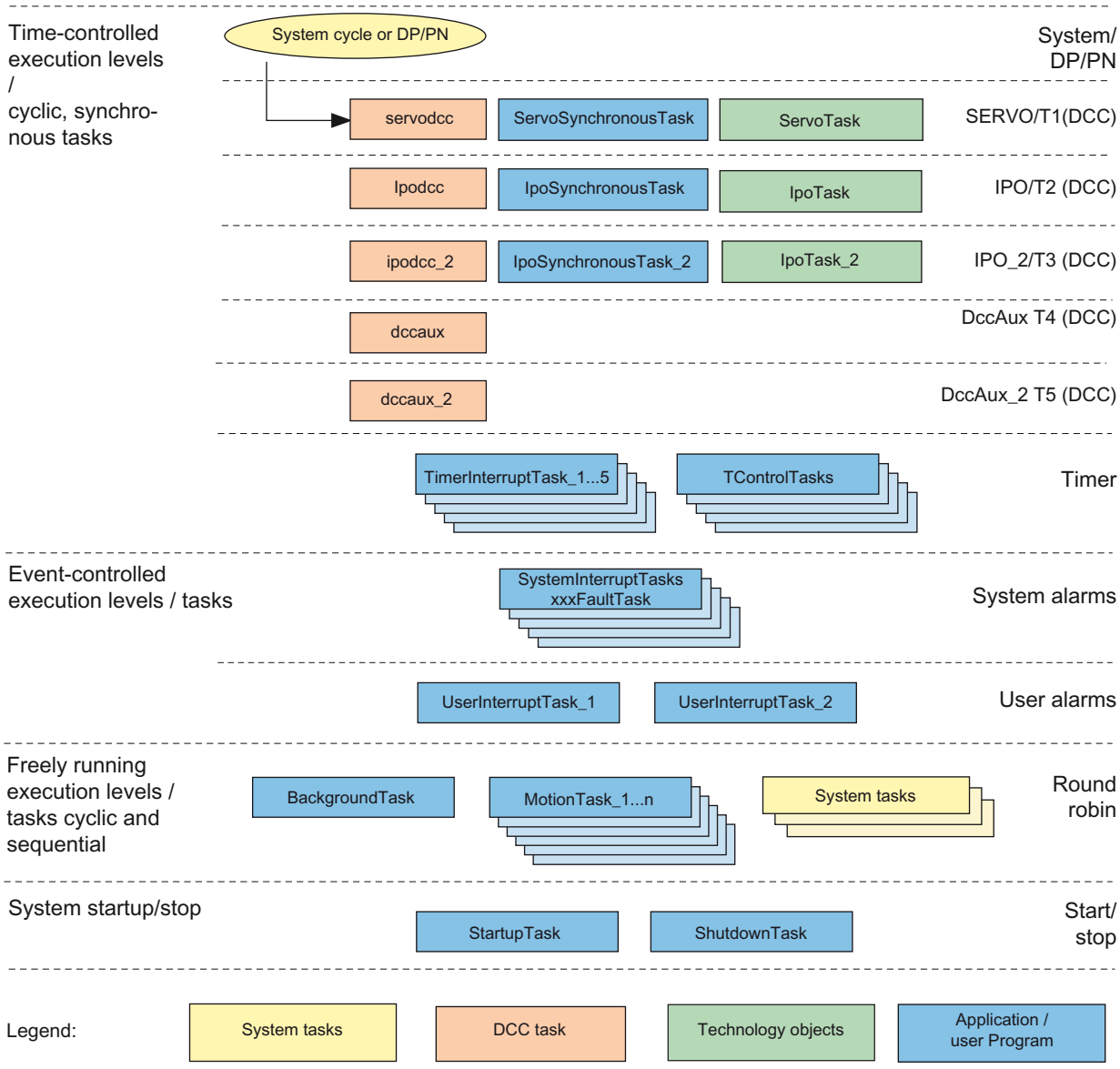


Figure 4-87 Execution levels and tasks in the SIMOTION runtime system

When you use technology packages, the execution levels provided by the system are automatically assigned. The user program cannot influence the processing of technology functions in these execution levels. System tasks are acyclic communication (Ethernet, PROFIBUS DP, PROFINET IO), debug services or trace preprocessing, for example.

The DCC levels and tasks are not visible in the execution system in the SCOUT user interface. The DCC plans are arranged in the DCC editor via execution groups, see **Description of the DCC editor**.

The following example for the Axis technology object demonstrates how the system and user program tasks interact with each other. At the end of cyclic data transmission, the ServoSynchronousTask and ServoTask are started, followed by the IPOSynchronousTask and IPOTask.

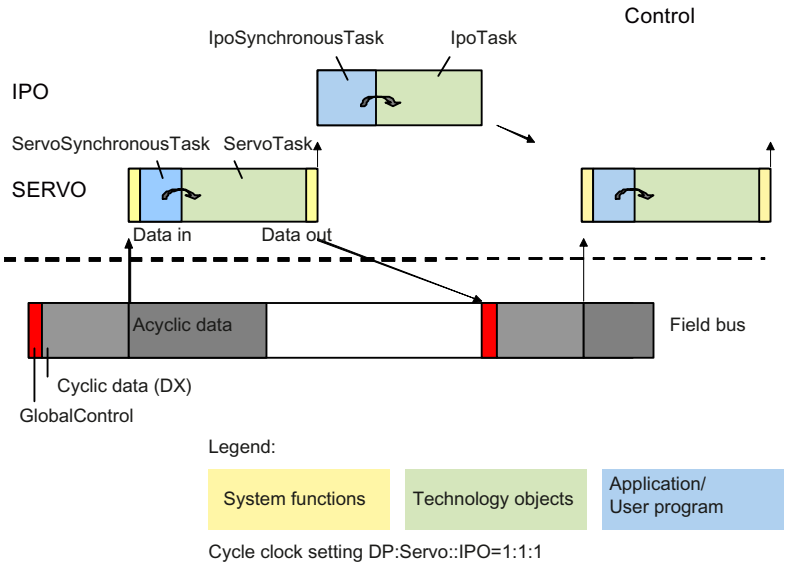


Figure 4-88 Task interaction

For more information, go to:

Isochronous I/O processing on fieldbus systems (Page 1383)

Dynamic response with respect to data processing in the control (Page 1389)

Tasks

One or more tasks are available in each execution level for user programming.

The main features of the tasks are:

- Start behavior: When and under what conditions a task is started. (refer to Table)
- Priority: Which task is interrupted by which other task. (refer to the table for the task priorities in the next section)

The following table shows the tasks available for user programs.

Table 4-70 Tasks in the SIMOTION runtime system

Task		Description
StartupTask		The **StartupTask** is executed once at the transition from **STOP** or **STOP U** mode to **RUN** mode. It is intended for initialization and resetting of technology objects.
Free-running tasks:		In the **round robin execution level**, the **MotionTasks** and **BackgroundTask** are executed by the system in the background in the time-slice procedure.
	MotionTasks	**MotionTasks** are intended for the programming of sequences, for programmed motion control or other *sequential* executions. MotionTasks are started by user programs and executed once.
	BackgroundTask	The **BackgroundTask** is provided for the programming of *cyclic* sequences without a fixed time frame. The BackgroundTask is started after system start-up and then executed cyclically and free-running.
Time-driven tasks and synchronous tasks:		**Cyclic tasks**. They are called cyclically in a certain time frame and are automatically restarted after the execution of the assigned programs.

Task		Description
	TimerInterruptTasks	**TimerInterruptTasks** are intended for the periodical starting of programs.
	SynchronousTasks	**SynchronousTasks** are started periodically, synchronous with a specified system cycle clock.
Event-driven tasks:		**Sequential tasks**. They are started and executed once when an event occurs and then terminated.
	SystemInterruptTasks	**SystemInterruptTasks** are started and executed once when a system event occurs.
	UserInterruptTasks	**UserInterruptTasks** are started and executed once when a user-defined event occurs.
ShutdownTask		The **ShutdownTask** is executed once at the transition from **RUN** mode to **STOP** or **STOP U** mode.

Note

In addition to the user program tasks, there are various system-internal tasks that the user cannot influence.

The **ControlPanelTask** is such a system-internal task, for example. It is visible during the task run times of the device diagnosis, but not in the execution system.

As of V4.1.2, the TaskTrace is also available. The SIMOTION Task Trace records the sequence of the individual tasks, labels "user events", which you can generate using a program command, and represents these graphically; see the Task Trace Function Manual.

See also

StartupTask (Page 1314)

MotionTasks (Page 1316)

BackgroundTask (Page 1319)

TimerInterruptTasks (Page 1323)

SynchronousTasks (Page 1326)

SystemInterruptTasks (Page 1334)

UserInterruptTasks (Page 1338)

ShutdownTask (Page 1343)

Isochronous I/O processing on fieldbus systems (Page 1383)

Execution system in SIMOTION SCOUT

All tasks used in the execution system can be viewed in SIMOTION SCOUT.

- Select the SIMOTION device in the project navigator and select **Target system > Configure execution system** in the menu or double-click **EXECUTION SYSTEM**.

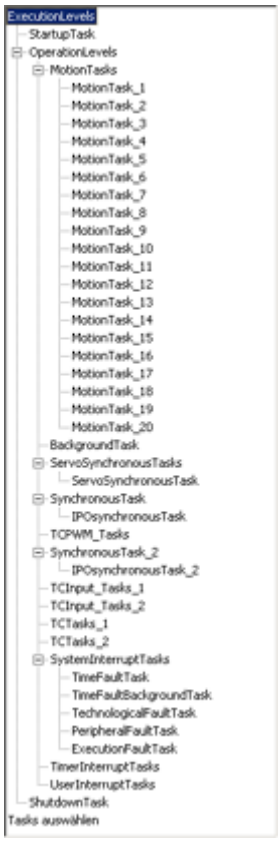


Figure 4-89 Overview of the tasks in SCOUT

Execution system

The execution system with the execution levels and the assigned tasks are displayed here.

Execution levels tree

The execution levels tree displays the available execution levels / tasks as fixed entries.

The **OperationLevels** folder contains the tasks that are available in the **RUN** mode.

- To open the **OperationLevels** folder, click the plus sign in front of it.

The list below each execution level or task name shows the configured tasks and the programs assigned to them.

After you have assigned programs to tasks, they are displayed in the execution levels tree.

Use task in execution system

Select the tasks from the list that should be used in the execution system.

- Activate the checkbox of the respective task.

Only those tasks that have been selected will be shown in the execution levels tree.

Execution system - context menu

You can select the following functions:

Function		Meaning/Note
Open		Use this to open the execution level configuration.
Expert		
	Set system cycle clocks	Use this to set the ratio between the system cycle clocks (e.g. between the interpolator cycle clock and the position control cycle clock) depending on a parameterized isochronous operation at the PROFIBUS or PROFINET interface.
Print		Use this to print the contents of the execution system. All tasks with the associated configuration are printed.
Print preview		Use this to open the print preview for the execution system.

Task priorities

Tasks and the programs assigned to them are started and executed at a certain time. If several tasks are to be started and executed at the same time, the task priority determines which task will be executed first.

Table 4-71 Overview of the task priorities

Priority	Execution level	Task	Cyclic/ sequential	Meaning
High	Servo, DP communication	-	z	
	Servo/T1 (DCC)	**Servodcc** **ServoSynchronousTask** **ServoTask**	z	Task of a DCC runtime group (T1) User program task in the servo cycle clock System task
	IPO/T2 (DCC)	**Ipodcc**	z	Task of a DCC runtime group (T2)
		IPOSynchronousTask • Check UserInterrupt • User program	z	User program task in the interpolator cycle clock
		IPOTask • Check the condition for WAITFORCONDITION (IPO, system)	z	System task in the IPO cycle clock
	IPO_2/T3 (DCC)	**Ipodcc_2**	z	Task of a DCC runtime group (T3)
		IPOSynchronousTask_2	z	User program task in the interpolator cycle clock 2
		IPOTask2	z	System task in the IPO cycle clock 2
	TControlTasks	**PWMsynchronousTask**	z	Temperature controller task
		PWMTask	z	System task
		InputTask_1	z	System task
		InputSynchronousTask_ 1	z	Temperature controller task
		InputTask_2	z	System task
		InputSynchronousTask_ 2	z	Temperature controller task
		ControlTask_1	z	System task
		PostControlTask_1	z	Temperature controller task
		ControlTask_2	z	System task
		PostControlTask_2	z	Temperature controller task
	DccAux (DCC)	**Dccaux**	z	Task of a DCC runtime group (T4)
	DccAux_2 (DCC)	**Dccaux_2**	z	Task of a DCC runtime group (T5)
	SystemInterruptTasks	**TimeFaultTask**	s	System alarm (in the order of events)
		TimeFaultBackgroundTask	s	System alarm (in the order of events)
		TechnologicalFaultTask	s	System alarm (in the order of events)
		PeripheralFaultTask	s	System alarm (in the order of events)
		ExecutionFaultTask	s	System alarm (in the order of events)
	UserInterruptTasks	**UserInterruptTask_1**	s	User alarm (in the order of events)
		UserInterruptTask_2	s	User alarm (in the order of events)

Priority	Execution level	Task	Cyclic/ sequential	Meaning
	TimerInterruptTasks	**TimerInterruptTask1 ... TimerInterruptTask5**	z ... z	Timer alarms
	Round robin	**BackgroundTask**	z	User program task (the sequence cannot be specified)
		MotionTask_1 ... MotionTask_32	s ... s	User program tasks (the sequence cannot be specified)
Low				

Priorities

The priority of a task *cannot* be changed by the user. For information on the execution levels of Servo_fast and IPO_fast, see also Priorities and sequences of synchronous tasks (Page 1369).

Note

Tasks run according to their priorities. Thus, higher priority tasks supersede lower priority tasks.

Therefore, fluctuations of the cycle time can occur for lower priority tasks.

- **TimerInterruptTasks:**
  The shorter a time slice, the higher its priority.
- **UserInterruptTasks:**
  All tasks have the same priority and are executed in the order of their activation events.
- **Wait for condition / WAITFORCONDITION** temporarily increases the priority of a MotionTask:
  - The condition is checked with the same priority as for **UserInterruptTasks**.
  - If the condition is satisfied, the MotionTask (that was previously pending) is reactivated.
  - The commands enclosed between **WAITFORCONDITION** and **ENDWAITFORCONDITION** are executed with increased priority (between **SystemInterruptTasks** and **TimerInterruptTasks**).

Further information on **Wait for condition** / **WAITFORCONDITION** can be found in the **SIMOTION MCC** or **SIMOTION ST** programming manuals.

Checking the condition for UserInterruptTask

The condition for the UserInterruptTasks is checked in the IPOSynchronousTask, prior to the execution of the user program of the IPOSynchronousTask.

Any errors that occur for the UserInterruptTask condition will be handled as if they came from the IPOsynchronousTask.

The time required for checking the condition of the UserInterruptTask is part of the runtime of the IPOsynchronousTask.

Checking the wait condition

If wait conditions can be used, such as **Wait for condition** / **WAITFORCONDITION**, the system performs the check, the task is not started for the check.

A synchronous setting on the command (e.g. Wait until motion end) is reasonable especially in sequential tasks.

The WAITFORCONDITION condition is checked after the user program for the IPOSynchronousTask at the start of the IPOTask.

Any errors that occur for WAITFORCONDITION will be handled as if they came from the associated MotionTask.

The time required for checking the WAITFORCONDITION condition(s) is part of the runtime of the IPOTask.

Task start sequence

When the StartupTask is completed, **RUN** mode is reached.

The following tasks are then started:

- SynchronousTasks
- TimerInterruptTasks
- BackgroundTask
- MotionTasks for which the attribute for automatic start is set

Note

The priorities of the execution levels or their tasks do not indicate the order in which the MotionTasks, BackgroundTask and time-triggered tasks are started after **RUN** mode is reached.

Priorities of the temperature controller tasks

The PWM tasks have a separate layer; their priority is located between servo level and IPO level. In terms of priority, all other temperature controller tasks are sorted between the DccAux task and the Fault tasks and are located on the same layer.

Runtime model in SIMOTION

The following diagram shows the general sequence and (from top to bottom) the priorities of the tasks in SIMOTION.

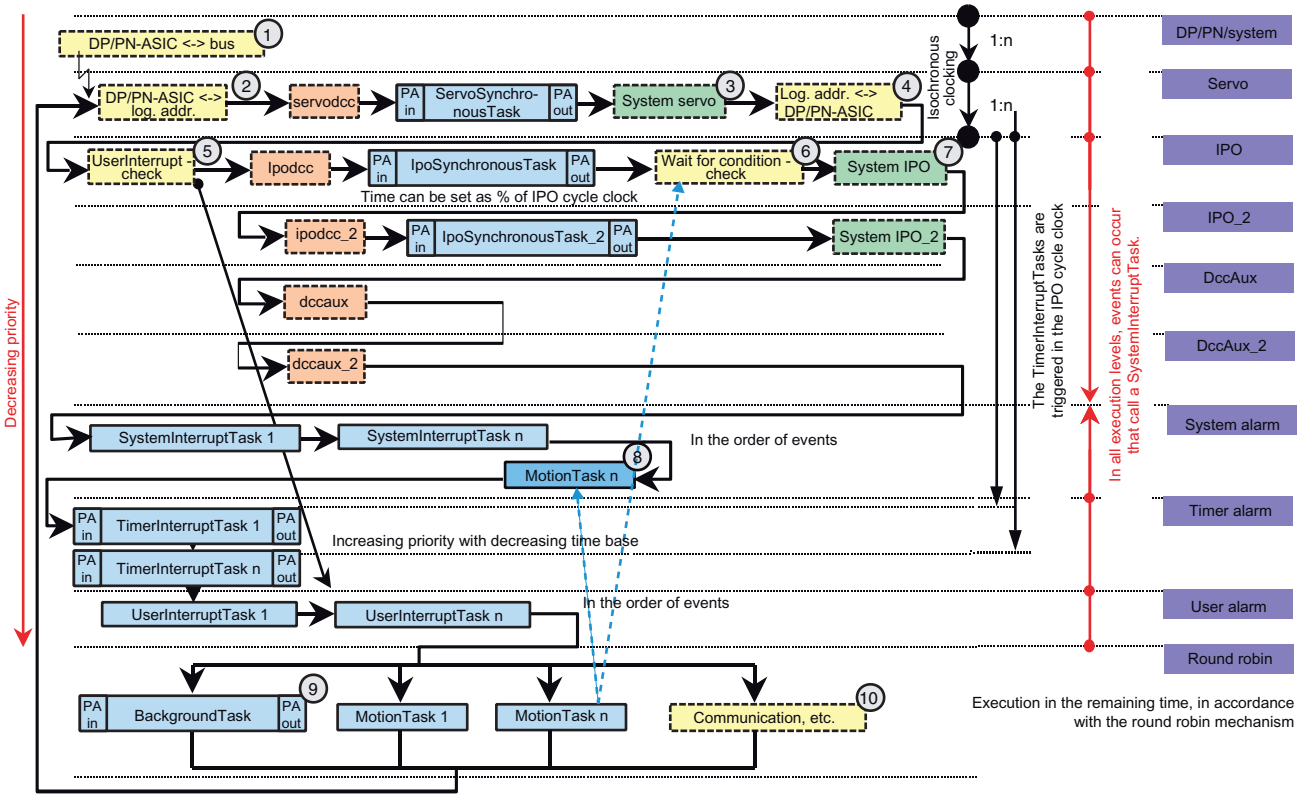


Figure 4-90 Runtime model in SIMOTION - general sequence, priorities

Explanations for the figure:

The display applies to a ratio between DP, Servo and IPO of 1:1:1

- Color blue/green: available to the user (application / technology objects)
- Color yellow (dashed): not available to the user (system tasks)

Explanation of the system tasks

1. **DP/PN-ASIC <-> Bus:**
   Data is copied from the communications chip to the PROFIBUS or PROFINET IO with IRT or fetched from there. The transfer of the I/O inputs on the logical addresses (2) is started at the end of the copy action.
   The copy action runs on its own processor and thus asynchronously to the remaining execution system. The synchronization point is the end of the data exchange between the bus and the ASIC.
2. **DP/PN-ASIC -> log. addr.:**
   The I/O inputs are loaded from the communications chip.

3. **System servo:**
   System calculations in the servo cycle clock (position controller, etc.).
   If all tasks of the servo level cannot be calculated in a single cycle (bus cycle clock), a level overflow occurs and the system enters STOP mode, the start-up lock is set and a corresponding entry is made in the diagnostic buffer. Only after a ramp-up (power on/off) or download can the system return to RUN mode.
4. **Log. addr. -> DP/PN-ASIC:**
   The I/O outputs are written to the communications chip.
5. **UserInterrupt - Check:**
   The conditions of the two user interrupts are checked.
6. **Wait for condition - Check:**
   The conditions for WAITFORCONDITION (wait for axis, wait for signal, etc.) are checked.
7. **System IPO/IPO_2:**
   The system-side components of the IPO cycle clock are calculated (motion control: Positioning profiles, synchronous operation, etc.).
8. **MotionTask n:**
   A MotionTask that is waiting with a WAITFORCONDITION will be switched in preference when the condition occurs (with higher priority).
9. **BackgroundTask:**
   The **BackgroundTask** runs here.
   The updating of the background process image (PI) is performed at the start and after completion of the complete BackgroundTask.
   The runtime model normally runs several times between the reading of the input image and the writing of the output image. I.e. depending on the size of the user program, the BackgroundTask is interrupted several times by higher priority tasks (starting with the ServoTask).
10. **Communication:**
    Communication functions (HMI, PG/PC, etc.).

See also

SynchronousTasks (Page 1326)

BackgroundTask (Page 1319)

Execution system in the symbol browser

Description

You can monitor and, if required, change program instance data of programs that are used in the execution system. This data includes the values of local variables that are normally not monitored on the unit.

For example, you have used a function block several times in the execution system on MotionTasks. You can then view the program instance data under every MotionTask in the symbol browser of the execution system.

To display the symbol browser for the execution system:

1. Select the execution system in the project navigator.
2. Click the symbol browser into the foreground in the detail view.
   The available program instance data is then displayed for each task.

Note

If you have selected "Only create program instance data once" as the compiler option, the program instance data is no longer displayed. It is displayed locally on the unit.

Synchronism of all components

Description

All components (the SIMOTION system, user programs, drives, bus system, and accurately-timed I/O modules) work deterministically and in strict synchronism. The individual components are synchronized via an isochronous fieldbus (PROFIBUS or PROFINET).

SIMOTION synchronizes itself with the bus cycle clock and passes this on to a second bus as well.

This ensures determinism for all the components, even if multiple machine modules and bus segments are being used. Distributed synchronous operation is also possible, e.g. as shown in the figure below:

Although axes 1 to 4 and axes 5 and 6 are controlled by different SIMOTION devices, they can still be operated synchronously.

This consistent synchronism results in high dynamic response and precision throughout the entire machine.

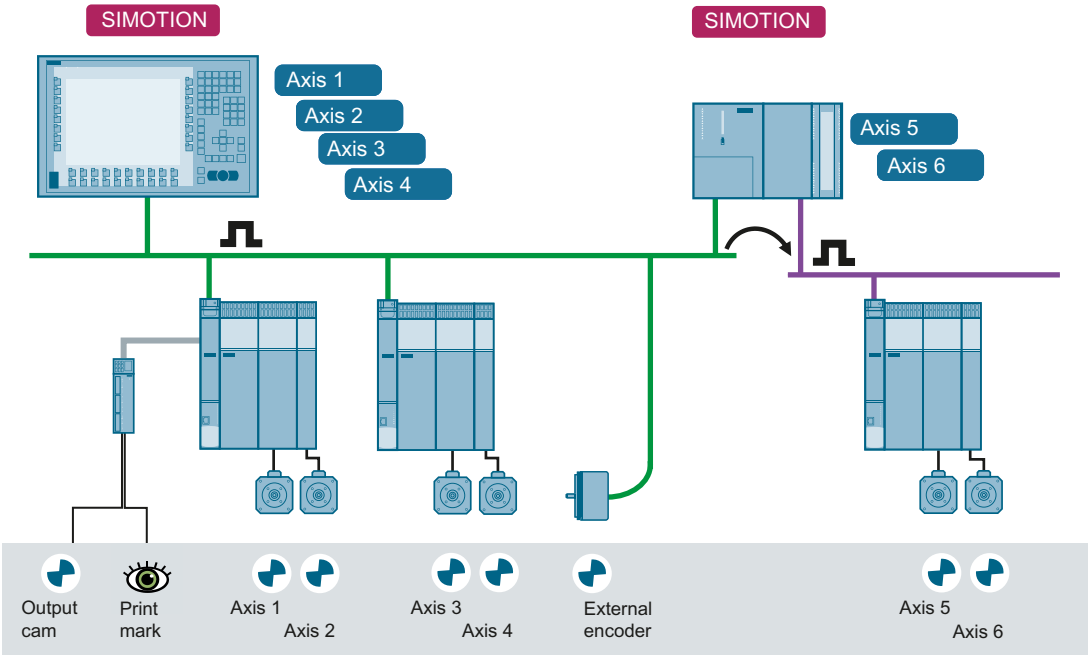


Figure 4-91 Synchronism with SIMOTION

Identifying the task name and type (as of Kernel V4.5)

With SIMOTION Kernel version V4.5 and higher, you can identify during runtime the name and type of the task assigned to a program.

To identify the task name, use the _getCurrentTaskName() function which is called without specification of a parameter. The return value of data type STRING contains the task name. A description of the function can be found in the "System Functions/Variables Devices" List Manual (reference list).

The task type is specified in the TaskStartInfo TSI#taskType. This TaskStartInfo contains the type of the task as an EnumTaskType. This data type is described in the TaskStartInfo of the StartupTask (Page 1263).

4.2.7.2 Description of the user program tasks

StartupTask

The **StartupTask** is provided for the one-time initialization and the resetting of the technology objects.

It is active when the operating mode switches from **STOP** or **STOP U** to **RUN**.

It must *not* be used for process start-up or homing or for setting up axes (motion commands must not be used).

While the StartupTask is being executed, no other user program tasks except for the **SystemInterruptTask** and the **UserInterruptTask** are active.

Access to process image and symbolic I/O variables is restricted. The process image of the inputs is updated before the startup task and remains constant for the duration of the startup task. The process image of the outputs is set to zero before the startup task, and output after the startup task. Direct accesses to inputs provide the current values. Write I/O variables can be set to initial values. These values, however, act only after the startup task with the enable of all outputs on the terminals.

When the StartupTask is completed, **RUN** mode has been reached. The following tasks are now started:

- **SynchronousTasks**
- **TimerInterruptTasks**
- **BackgroundTask**
- **MotionTasks**, in which the automatic start attribute is set

Note

The priorities of the execution levels or their tasks do not indicate the order in which the MotionTasks, BackgroundTask and time-triggered tasks are started after **RUN** mode is reached.

Select the **Program assignment** tab to assign the created and compiled programs to the StartupTask and define their execution sequence.

Configuring the StartupTask

1. Click **StartupTask** in the Execution Levels tree.

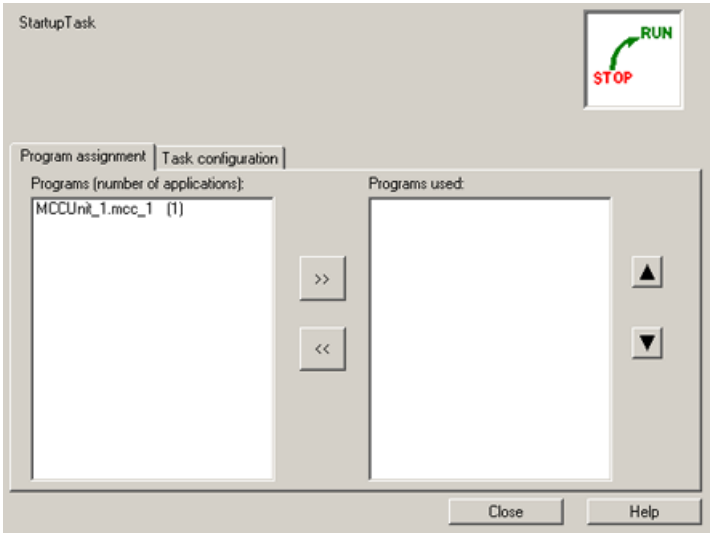


Figure 4-92 Configuration of the StartupTask (program assignment)

2. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
3. Switch to the **Task configuration** tab.

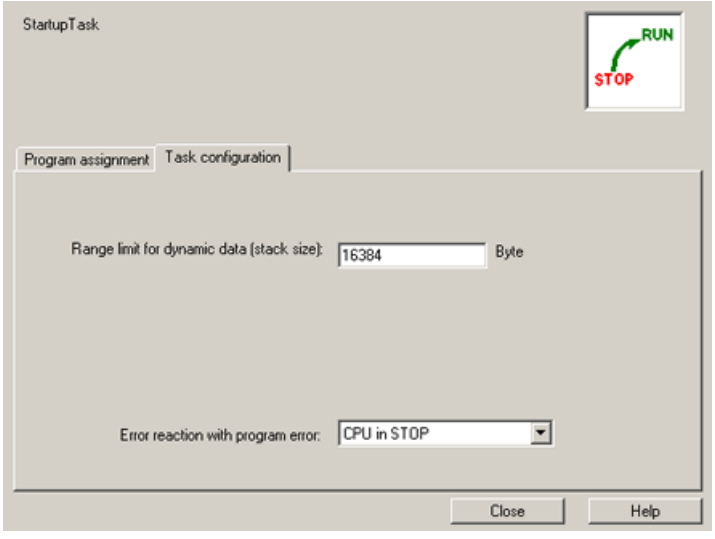


Figure 4-93 Configuration of the StartupTask (task configuration)

4. If required, enter the **Range limit for dynamic data** (**stack size**).
5. Specify the **Error response to program error** (e.g. **ExecutionFaultTask**).

Task configuration - StartupTask

In the **Task configuration** tab, parameterize the error response to program errors.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

Assigning programs to the execution levels/tasks (Page 1345)

SystemInterruptTasks (Page 1334)

MotionTasks

MotionTasks are intended for the programming of sequences, for programmed motion control or other sequential executions.

Example for the sequential execution: An axis traverses to a target position, waits for an enable signal, and then traverses to the next target position.

Several MotionTasks are available:

- With SIMOTION D and SIMOTION P only: 32 MotionTasks **MotionTask_1** to **MotionTask_32**

The names of the MotionTasks can be changed, see below.

MotionTasks are executed in the round robin execution level.

Note

C2xx devices only have 20 MotionTasks; all other CPUs (see above) have 32. With the C2xx, the excess MotionTasks may be deleted. Once deleted, these MotionTasks cannot be recreated.

MotionTasks and **BackgroundTask** share the free time apart from the higher-priority system and user program tasks. The relationship of the time slices between both levels can be parameterized, see Setting the time allocation (Page 1376).

There is no fixed sequence of execution for MotionTasks and BackgroundTask.

Information on the influencing of the task execution is provided in Overview of the task control commands (Page 1435).

Starting a MotionTask

MotionTasks are usually controlled from the user program via task control commands such as **_startTaskID**, **_stopTaskID,**, ... With the corresponding configuration (set attribute), a MotionTask starts automatically when the **RUN** mode has been reached. You can scan the current task status using the **_getStateOfTaskID** system command.

A MotionTask does not have a time monitoring, i.e. once a MotionTask is started, it can remain active for an indefinite period.

A MotionTask that waits for a synchronous command remains active with regard to its status.

Completing a MotionTask

A MotionTask is completed when the task has been completed or at the transition to the **STOP** or **STOP U** mode (start of the **ShutdownTask**).

One way to automatically suspend the task is to use wait commands **Wait for condition** / **WAITFORCONDITION**.

The task is suspended when the wait command is issued. The condition specified in the command is checked in the IPO cycle clock. When the condition is fulfilled, the MotionTask is automatically resumed. Depending on where you position the wait command, you can influence the task priority when execution continues.

The commands (with MCC in the gray area behind the command) enclosed between **WAITFORCONDITION** and **ENDWAITFORCONDITION** are executed with increased priority (between **SystemInterruptTasks** and **TimerInterruptTasks**).

Select the **Program assignment** tab to assign the created and compiled programs to the MotionTasks and define the execution sequence.

You can set the following parameters:

Field/button	Meaning/information
MotionTask	Under MotionTask, select the MotionTasks you want to assign the programs to. You can assign several programs to one MotionTask.
MotionTask_1 to MotionTask_n	Predefined names of the possible MotionTasks.
Use task in execution system	Activate the checkbox to display and use the task in the execution system. If the checkbox is deactivated, you cannot assign any programs to this task.

Configuring MotionTasks

You can define which tasks are to be started automatically when the **RUN** mode is reached. Otherwise MotionTasks must be explicitly started via programmed task control commands.

1. Click **MotionTasks** in the execution level tree.
2. Select the required task in the **MotionTask** list. The task names can be changed.

3. To do so click in the **MotionTask** field and enter a new name. Umlauts and special characters are not permitted. The name is updated in the task list.

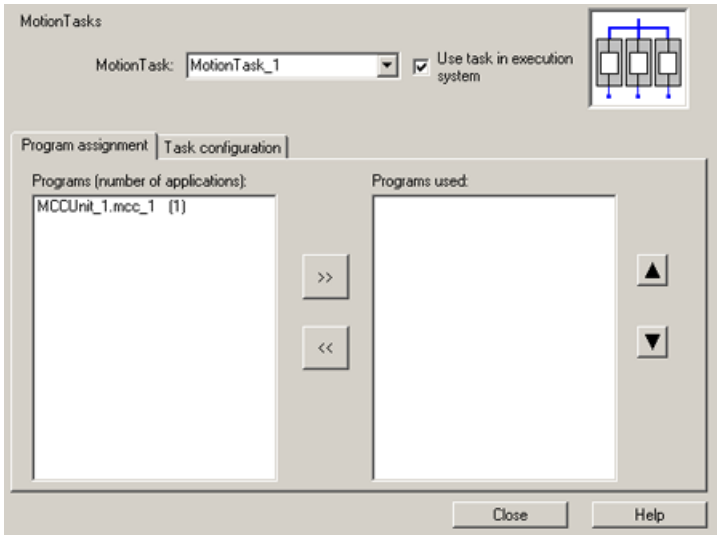


Figure 4-94 Configuration of the MotionTasks (program assignment)

4. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
5. Select the **Task configuration** tab.

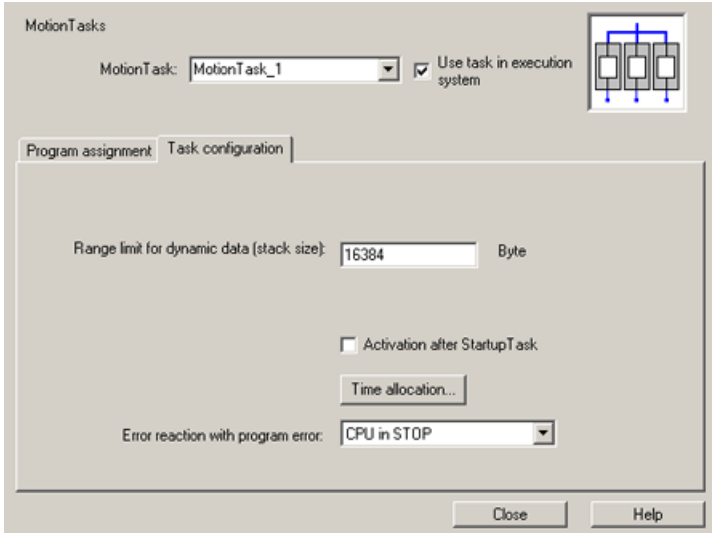


Figure 4-95 Configuration of the MotionTasks (task configuration)

6. Activate the **Activation after StartupTask** option to start the MotionTask once after the StartupTask.
7. If required, enter the **Range limit for dynamic data** (**stack size**).
8. Specify the **Error response to program error** (e.g. **ExecutionFaultTask**).
9. Repeat steps 2 through 7 for all the MotionTasks to be configured.
10. Specify the **time allocation in the round robin execution level** between MotionTasks and BackgroundTasks, see Setting the time allocation (Page 1376).

Task configuration - MotionTasks

In the **Task configuration** tab, parameterize the time allocation between the MotionTasks and BackgroundTasks in the round robin execution level, and define when the MotionTasks are activated.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Activation after StartupTask		Select this if you want the MotionTask to start once when **RUN** mode is reached (StartupTask is completed).
Time allocation		Clicking this button opens the screen form for **time allocation in the round robin execution level**. This is where you can parameterize the time allocated to MotionTasks and BackgroundTasks in the round robin execution level.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

BackgroundTask (Page 1319)

Assigning programs to the execution levels/tasks (Page 1345)

SystemInterruptTasks (Page 1334)

Time allocation in the round robin execution level (Page 1373)

Assigning programs to the tasks (Page 1417)

Specifications for the configuring (Page 1417)

Description of the user program tasks (Page 1314)

BackgroundTask

The **BackgroundTask** is provided for the programming of cyclic sequences without a fixed time frame.

It is executed cyclically in the round robin execution level, which means it will be automatically restarted on completion.

The BackgroundTask is used in programs that have to be executed cyclically, e.g. interlocking tasks, PLC tasks.

The cycle time of the BackgroundTask is monitored. When the cycle time monitoring responds, the **TimeFaultBackgroundTask** is started. The CPU will switch to **STOP** mode if the task is not configured or there is no program assigned to it.

The process image of the inputs and outputs is generated for the BackgroundTask in address space 0.0 to 63.7. The process image remains consistent during the time the BackgroundTask is being processed.

You can use the BackgroundTask to implement lower-priority, cyclical logic functions, interlocks, calculations, monitoring functions.

The **BackgroundTask** shares available CPU time with the **MotionTasks**.

Note

The BackgroundTask shares CPU time with MotionTasks and SystemTasks (e.g. communication tasks). Relative to time allocation you must consider that the runtime, i.e. the performance, is influenced by the settings (time allocation of the round robin execution level).

Starting the BackgroundTask

The BackgroundTask is started automatically when the **RUN** mode has been reached or when the task has been completed (with the next position control cycle clock).

Completion of the BackgroundTask

A BackgroundTask is completed at the transition to the **STOP** or **STOP U** mode (start of the **ShutdownTask**).

Select the **Program assignment** tab to assign the created and compiled programs to the BackgroundTasks and define the execution sequence.

Configuring the BackgroundTask

1. Click **BackgroundTask** in the Execution Levels tree.

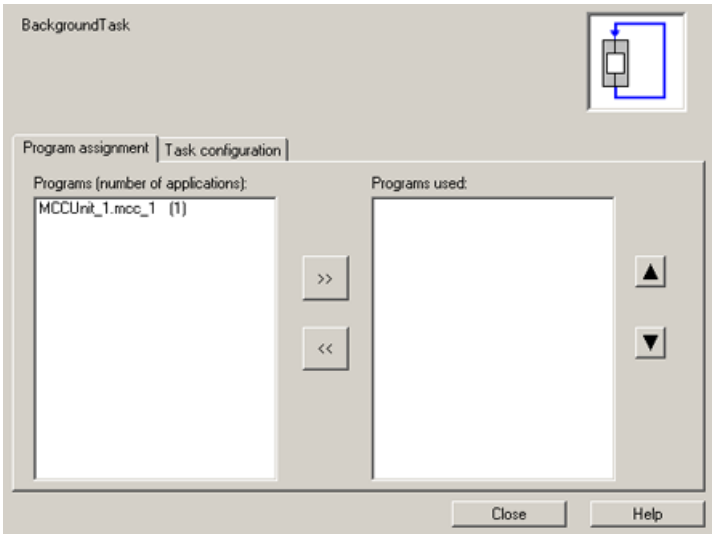


Figure 4-96 Configuration of the BackgroundTask (program assignment)

2. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
3. Switch to the **Task configuration** tab.

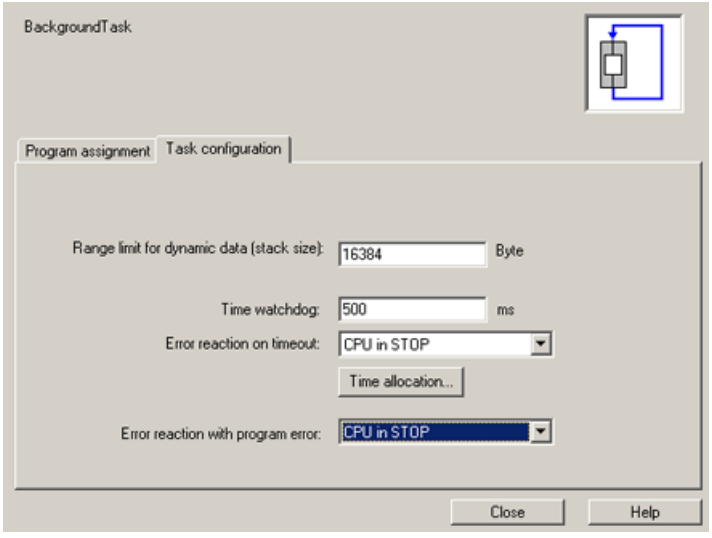


Figure 4-97 Configuration of the BackgroundTask (task assignment)

4. If required, enter the **Range limit for dynamic data (stack size)**.
5. Enter a value for the **time monitoring**.
   If this time is exceeded, the relevant SystemInterruptTask (**TimeFaultBackgroundTask**) can be called or the **CPU to STOP** can be set, 0 ms = no monitoring.
6. Specify the **Error response after timeout**.
   CPU in STOP or call **TimeFaultBackgroundTask**.

7. Specify the **time allocation in the round robin execution level** between MotionTasks and BackgroundTasks, see **Setting the time allocation**.
8. Specify the **Error response to program error** (e.g. **ExecutionFaultTask**).

Task configuration - BackgroundTask

You can parameterize the time monitoring and the error response in the **Task configuration** tab.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Time monitoring		Specify the time monitoring in ms within which the BackgroundTask must be calculated here. This allows you to specify the required cycle time for the BackgroundTask. The time monitoring is inactive if you enter 0 or no value.
Error response after timeout		You can select the error response, if the cycle for the BackgroundTask is not terminated within the time frame specified under Time monitoring.
	CPU to Stop	The CPU switches to **STOP** mode.
	TimeFaultBackgroundTask	The TimeFaultBackgroundTask is started. All programs assigned to this task are started.
Time allocation		Clicking this button opens the screen form for **time allocation in the round robin execution level**. This is where you can parameterize the time allocated to MotionTasks and BackgroundTasks in the round robin execution level.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to Stop	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. The CPU subsequently changes to **STOP**.

See also

MotionTasks (Page 1316)

Assigning programs to the execution levels/tasks (Page 1345)

Watchdog (Page 1363)

Setting of the time allocation (Page 1376)

SystemInterruptTasks (Page 1334)

Time allocation in the round robin execution level (Page 1373)

TimerInterruptTasks

TimerInterruptTasks are intended for the periodical starting of programs.

Five TimerInterruptTasks, i.e. **TimerInterruptTask_1** to **TimerInterruptTask_5**, are available for different time levels.

TimerInterruptTasks are periodically started and executed in the configured fixed time frame (e.g. 100 ms).

This time frame must be a multiple of the interpolator cycle clock.

In this task, you can implement closed-loop control or monitoring functions that require a reproducible time reference without a direct link to I/O or motion control of the axes.

Additional system and user program tasks are provided for the TControl technology package. The TControl technology package is processed in the system tasks, whereas the application-specific adaptations are processed in the user program tasks.

You can find additional information in the functional description of the temperature controller, refer to the **Motion Control Additional Technology Objects** Function Manual.

Starting a TimerInterruptTask

TimerInterruptTasks are started periodically. A start delay can be set.

Completing a TimerInterruptTask

TimerInterruptTasks are completed automatically after completion of the programs assigned to the TimerInterruptTask.

You can assign the created and compiled programs to the selected TimerInterruptTask and define their execution sequence in the **Program assignment** tab.

You can set the following parameters:

Field/button	Meaning/information
For task	Use this to select one of the five TimerInterruptTasks to which you want to assign the programs. You can assign several programs to one TimerInterruptTask.
TimerInterruptTask_1 to **TimerInterruptTask_5**	Predefined names of the five TimerInterruptTasks.
Defined time level	Use this to select the time frame for restarting the TimerInterrupt-Task. You can choose time levels from the available list or enter additional ones as required. The value must be a multiple of the IPO cycle clock.
Use task in execution system	Activate the checkbox to display and use the task in the execution system. If the checkbox is deactivated, you cannot assign any programs to this task.

Configuring TimerInterruptTasks

1. Click **TimerInterruptTasks** in the Execution Levels tree.
2. Select the required task in the **For task** selection box. The names (**TimerInterruptTask_1** to **TimerInterruptTask_5**) are preset.

3. Activate the **Use task in execution system** checkbox.

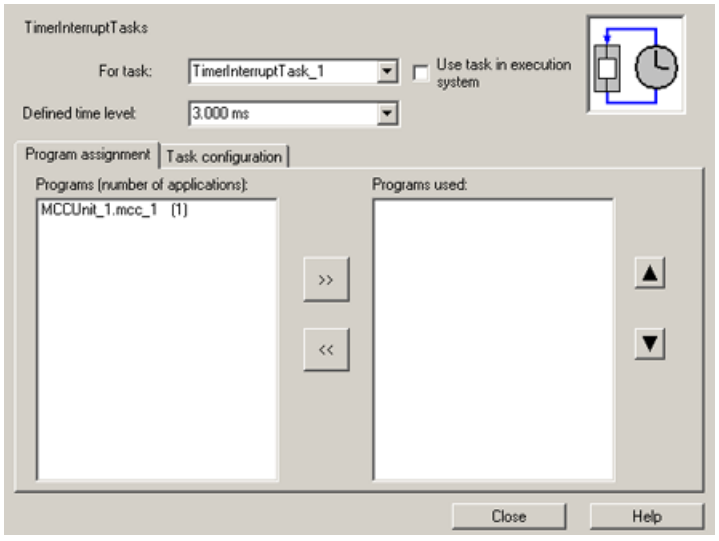


Figure 4-98 Configuration of the TimerInterruptTasks (program assignment)

4. Select a **Defined time level** or enter an arbitrary integer value. This value must be an integral multiple of the interpolator cycle clocks (IPO cycle clocks). Different settings are rounded up to the next integral multiple of the IPO cycle clock.
5. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
6. Switch to the **Task configuration** tab.

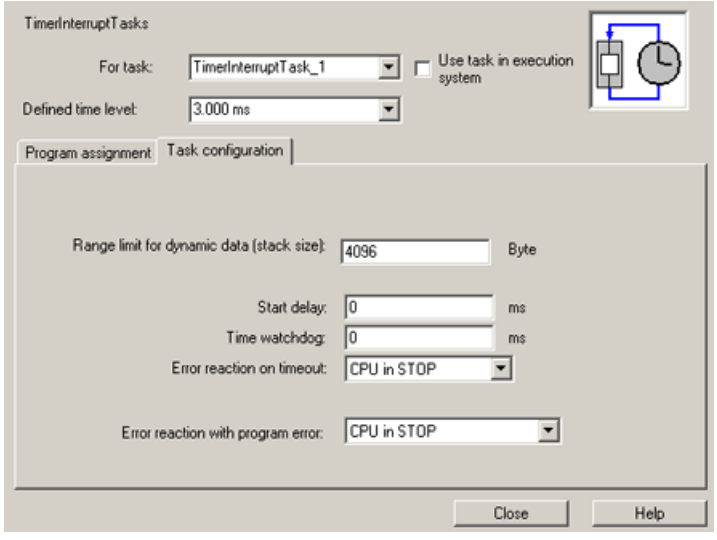


Figure 4-99 Configuration of the TimerInterruptTasks (task configuration)

7. If required, enter the **Range limit for dynamic data (stack size)**.

8. Enter the **Start delay** for this task, if applicable.
   To prevent different time-triggered tasks from starting at the same time, you can define a start delay for each task. This prevents different TimerInterruptTasks from being executed at the same time.
   Example:
   IPO cycle clock: 4 ms
   TimerInterruptTask_1: 8 ms
   TimerInterruptTask_2: 16 ms
   Both TimerInterruptTasks are started simultaneously in every fourth IPO cycle clock. By delaying the start of one of the two tasks by 4 ms, you can prevent them from being executed at the same time. In this way, the load on the system is distributed more evenly, and the behavior of the lower-priority tasks can be reproduced.
9. Enter a value for the **time monitoring**.
   If this time is exceeded, the associated SystemInterruptTask (**TimeFaultTask**) can be called or the **CPU to STOP** can be set, 0 ms = no monitoring.
10. Specify the **Error response with timeout**.
    CPU in STOP or call **TimeFaultTask**.
11. Specify the **Error response to program error** (e.g. **ExecutionFaultTask**).
12. Repeat steps 3 to 10 for all the TimerInterruptTasks to be configured.

Task configuration - TimerInterruptTasks

You can define the start delay, time monitoring and the error response for the TimerInterruptTasks in the **Task configuration** tab.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Start delay		Enter a time value in ms for the task start delay. The start of TimerInterruptTask is delayed by this time value. This delay ensures that different TimerInterruptTasks will not start simultaneously, resulting in a timeout.
Time monitoring		Specify the cycle time in ms for executing the TimerInterruptTask. Enter a value for time monitoring. The time monitoring is inactive if you enter 0 or no value.
Error response after timeout		You can select the error response if the TimerInterruptTask is not terminated within the time frame specified under Time monitoring.
	CPU to Stop	The CPU switches to **STOP** mode.
	TimeFaultTask	The TimeFaultTask is started. All programs assigned to this task are started.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.

Field/button		Meaning/information
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

Assigning programs to the execution levels/tasks (Page 1345)

Watchdog (Page 1363)

SystemInterruptTasks (Page 1334)

SynchronousTasks

SynchronousTasks are started synchronously to the specified system cycle clock on a periodic basis. They run with high priority in the time-triggered execution level.

The following SynchronousTasks are available:

- **ServoSynchronousTasks**
  - **ServoSynchronousTask** (as of V4.0): Synchronous with the position control cycle clock.
  - **ServoSynchronousTask_fast** (as of V4.2): Synchronous with the Servo_fast_clock_cycle (optional, only available for specific SIMOTION modules, see Servo_fast (application cycle clock for fast bus system) (Page 1364)).

  In the servo-synchronous tasks, you can implement time-critical terminal - terminal responses for I/O or fast influencing of setpoints on the servo level.
  See also isochronous I/O processing on fieldbus systems (Page 1383).
  Application, such as for fast terminal - terminal response. If a TO is configured for execution in the position control cycle clock then TO and motion commands can also be issued.
- **IPOsynchronousTasks**
  - **IPOsynchronousTask**: Synchronous with interpolator cycle clock IPO.
  - **IPOsynchronousTask_2**: Synchronous with interpolator cycle clock IPO_2
  - **IPOSynchronousTask_fast** (as of V4.2): Synchronous with the interpolator cycle clock IPO_fast (optional, only available for specific SIMOTION modules, see Servo_fast (application cycle clock for fast bus system) (Page 1364)).

  In IPO-synchronous tasks, you can implement time-critical functions that have a direct effect on the functions of the technology object. The user program is executed prior to the interpolator, i.e. the programmed functions can be effective in the same IPO cycle clock.
  Application, e.g. for a fast start depending on the event, fast response to events terminal - axis

Additionally, there are further SynchronousTasks for TControl:

- **PWMsynchronousTask**: Synchronous with the PWM cycle clock
- **InputSynchronousTask_1/InputSynchronousTask_2**: Synchronous with the Input1/2 cycle clock before temperature control
- **PostControlTask_1/PostControlTask_2**: Synchronous with the Control1/2 cycle clock after temperature control

You can assign the created and compiled programs to the selected SynchronousTask and define their execution sequence on the **Program assignment** tab.

You can set the following parameters:

Field/button		Meaning/information
For cycle clock level		Use this to select the SynchronousTask to which you want to assign the programs. You can assign several programs to one SynchronousTask.
	ServoSynchronousTask_fast	Optional. SynchronousTask which is started in the application cycle for the fast bus system (Servo_fast).
	IPOsynchronousTask_fast	Optional. SynchronousTask which is started in the interpolator cycle for the fast bus system (IPO_fast).
	ServoSynchronousTask	SynchronousTask which is started in the position control cycle clock
	IPOsynchronousTask	SynchronousTask which is started in the interpolator cycle clock (IPO cycle clock).
	IPOsynchronousTask_2	SynchronousTask which is started in the second interpolator cycle clock (IPO2 cycle clock).
	PWMsynchronousTask (TCPWM_Tasks)	SynchronousTask which is used for the actuating signal output of the TO temperature controller (temperature channel). Defines the basic cycle clock for further tasks of the TO temperature controller.
	InputSynchronousTask_ 1/2 (TCInput_Tasks_1/2)	SynchronousTask which is used for the actual value measurement of the TO temperature controller.
	PostControlTask _1/2 (TCTasks_1/2)	SynchronousTask which is used for the control of the TO temperature controller.
Use task in execution system		Activate the checkbox to display and use the task in the execution system. If the checkbox is deactivated, you cannot assign any programs to this task.

ServoSynchronousTask/ServoSynchronousTask_fast

ServoSynchronousTasks are intended for applications in which fast processes are to be implemented using isochronous mode.

ServoSynchronousTasks run within a position control cycle clock and behave similar to IPOsynchronousTasks.

ServoSynchronousTasks_fast run within a Servo_fast cycle clock and behave similar to the ServoSynchronousTasks.

The ServoSynchronousTasks are suitable for:

- Fast responses (e.g. for short terminal-terminal times for I/O processing)
- Influencing servo-effective system variables
  Information on servo effectiveness of the system variables is provided in the SIMOTION reference list technology package CAM system variables.
- For motion commands if a TO (in exceptional cases) is configured for execution in the position control cycle clock.

They are not suitable for:

- Communication
- For motion commands if a TO is configured for execution in the IPO cycle clock (since this is evaluated in the IPO cycle clock)

If a TO is configured for execution in the position control cycle clock then TO and motion commands can also be issued.

The same characteristics apply as for the IPOSynchronousTasks.

The ServoSynchronousTask can be configured in the execution system. The cycle time of the servo execution level is set with the servo. The default setting is active.

The ServoSynchronousTask_fast can be configured in the execution system. The default setting is inactive.

There is a process image available for the ServoSynchronousTask as for the other cyclic tasks. This only results in an improved performance if the same I/O addresses are accessed several times.

As of V4.1, the monitoring of the ServoSynchronousTasks can also be set. In this way, MOTION commands are also permitted in the ServoSynchronousTask, where the IPO share is executed in the servo (all commands with the step enabling condition IMMEDIATELY).

You can set the following parameters:

Field/button	Meaning/information
Range limit for dynamic data (stack size)	Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Monitoring when executing synchronous functions	You can select the error response if the ServoSynchronousTask is not completed within the system cycle clock.

Field/button		Meaning/information
	Discontinue time monitoring and interrupt the task	Compatible with the functionality up to V4.0 With synchronous functions, time monitoring is suspended and one servo cycle clock is lost - all old projects have this setting, too, if they have been updated to V4.1 (CPU replacement).
	Leave time monitoring active	Time monitoring is not suspended, i.e. the CPU goes to **STOP** with timeout when synchronous functions are called which really interrupt the ServoSynchronousTask. If overflows are tolerated, it might be possible to avoid **STOP** - this is the default setting for newly created CPUs. Diagnostics buffer entry: "STOP by execution system, cause: Timeout".
	CPU goes to STOP	There is no synchronous function permitted. In the case of a synchronous function, the CPU goes to **STOP** - even if the ServoSynchronousTask is not interrupted in the specific case. Diagnostics buffer entry: "Impermissible calling of a system/package function".
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

IPOsynchronousTask/IPOsynchronousTask_2/IPOsynchronousTask_fast

IPOSynchronousTasks are intended for applications that require, for example, fast and deterministic responses or correction movements. Optimum time transfer of motion commands to the motion control system is thus possible.

IPOsynchronousTasks run immediately before the interpolator within an IPO cycle clock. Thus, commands in these tasks can directly affect the motion control.

The **IPOSynchronousTask** is started synchronously to the IPO_clock_cycle while the **IPOSynchronousTask_2** is started synchronously to the IPO_clock_cycle_2, i.e. a reduced IPO cycle clock.

The **IPOSynchronousTask** is executed before the internal IPOTask and **IPOsynchronousTask_2** is executed before IPOTask_2.

The **IPOSynchronousTask_fast** is started synchronously with IPO_fast and, depending on the version, before the position control cycle clock.

The following properties must be specified for the **IPOsynchronousTasks**:

- Task configuration
- Number of level overflows in the IPO/IPO_2 cycle clock
  Not for IPOsynchronousTask_fast
  If the tasks in the IPO level (**IPOSynchronousTask**, **IPOTask**) are not completed within an IPO cycle clock, an overflow will occur. An overflow of a task in the cycle clock (n) must be processed in the following cycle clock (n+1).
  The same applies for the tasks in the IPO_2 level (**IPOSynchronousTask_2**, **IPOTask_2**).
  You can set the number of overflows which should be tolerated in sequence (n = 0 ... 5). The ratio can be violated n-times. The internal monitoring counter is reset when a task has been performed without overflowing.
  The accumulated level overflows are displayed in the **numberOfSummarizedTaskOverflow** system variables.
  No level overflows are permitted for **ServosynchronousTask_fast** and **IPOSynchronousTask_fast**.
- Error response after timeout (level overflow): The CPU goes to **STOP** mode and a starting lockout is set.
- **IPOsynchronousTask/IPO-clock_cycle or IPOsynchronousTask_2/IPO_2-clock_cycle**:
  Time ratio between IPOsynchronousTask and IPO cycle clock or IPOsynchronousTask_2 and IPO_2 cycle clock.
  A timeout occurs if an IPO-synchronous task exceeds this ratio.
  You also define the time frame for the IPOTask in the system cycle clocks. In the IPOSynchronousTask/IPOClockCycle parameter, you can specify the percentage of time that is to be made available for the IPOSynchronousTask per cycle clock.
  If, for example, 4 ms is set as IPO cycle clock and 25% set as ratio, the runtime of the IPOSynchronousTask must not exceed 1 ms, including possible interrupts due to higher priority tasks.
  Recommendation: With a gear ratio of servo: IPO > 1, enter a percentage as high as possible.

Note

If a synchronous system function is called that suspends the IPOSynchronousTask, the CPU will go into **STOP** mode with a diagnostic buffer entry.

- It is possible to configure whether time monitoring is to be performed.
- Note that the copy times of the non-equidistant I/O data are included in the runtime of the IPO/IPO_2 or the IPO-synchronous user tasks.

Configuring SynchronousTasks

1. Click the appropriate task in the Execution Level tree.
2. Select the required SynchronousTask in the **For cycle clock level** selection box.

3. Activate the **Use task in execution system** checkbox.

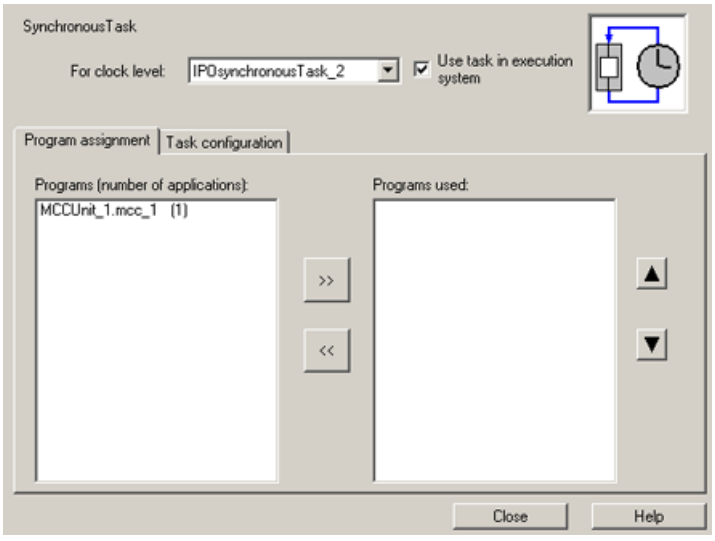


Figure 4-100 Configuration of the SynchronousTasks (program assignment)

4. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
5. Switch to the **Task configuration** tab.

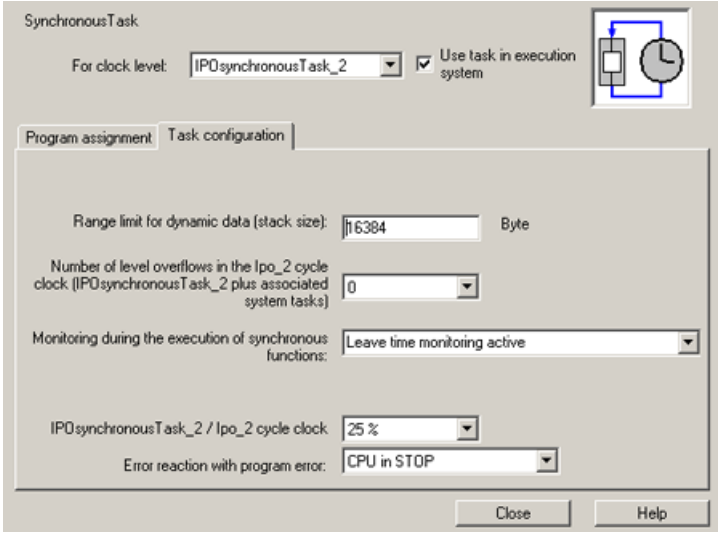


Figure 4-101 Example: Configuration of the IPOsynchronousTask (task configuration)

6. The following options are also available:
   - **Stack size** as an option in the Task configuration tab
   - Specify the number (0-5) of permitted level overflows in the **Level overflows** selection box
     No level overflows are permitted for ServoSynchronousTask_fast and IPOsynchronousTask_fast.
   - Select the ratio between SynchronousTask and IPO task
   - Select the **Error response to program error**
   - Configure the IPOSynchronousTask/IPOTask time monitoring
7. Repeat steps 3 through 5 for all the synchronous tasks to be configured.

Task configuration - SynchronousTasks

On the **Task configuration** tab you can select the error response for the SynchronousTask.

You can set the following parameters:

Table 4-72 Task configuration IPOsynchronousTasks

Field/button	Meaning/information
Range limit for dynamic data (stack size)	Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task.
Number of level overflows in the IPO cycle clock or IPO2 cycle clock	Not for IPOsynchronousTask_fast. Enter the number of tolerable level overflows in the IPO/IPO_2 cycle clock for the IPOsynchronousTasks. If the number of overflows is less than the entered overflows, there will be no error response. When arithmetic operations are processed, level overflows in the IPO/IPO_2 cycle clock can occur that can be tolerated by the user for the SynchronousTask and SynchronousTask_2 execution levels. Execution level overflows may occur if: • IPOSynchronousTask + IPO task > IPO cycle clock or • IPOSynchronousTask_2 + IPO_2 task > IPO_2 cycle clock This set tolerance level causes the next IPO cycle clock to be used for terminating the arithmetic operation of the previous cycle clock without triggering the set error response. When the set number of overflows is exceeded, or if no tolerance has been set, an entry is made in the diagnostic buffer and the error response (CPU to STOP with starting lockout) results. A maximum of 5 execution level overflows can be set.
Monitoring when executing synchronous functions	You can select the error response if the SynchronousTask is not completed within the system cycle clock.

Field/button		Meaning/information
	Discontinue time monitoring and interrupt the task	With synchronous functions, the time monitoring function is suspended and one IPO cycle is lost.
	Leave time monitoring active	Time monitoring is not suspended, i.e. the CPU goes to **STOP** with timeout when synchronous functions are called which really interrupt the IPOSynchronousTask. If IPO overflows are tolerated, it might be possible to avoid **STOP** - this is the default setting for newly created CPUs. Diagnostics buffer entry: "STOP by execution system, cause: Timeout".
	CPU goes to STOP	There is no synchronous function permitted. In the case of synchronous functions, the CPU goes to **STOP** - even if the IPOSynchronousTask is not interrupted in the concrete case. Diagnostics buffer entry: "Impermissible calling of a system/package function". The behavior applies to the IPOsynchronousTask, IPOsynchronousTask_2 and PWMTask.
IPOsynchronousTask / IPO-ClockCycle or **IPOsynchronousTask_2 / IPO_2Clock-Cycle**		Not for IPOsynchronousTask_fast. Here you select the duration for the specified tasks as a percentage of the cycle clock (e.g. IPOsynchronousTask / IPO cycle clock). If the task requires more time than set here, the system will respond according to the settings for error response. To configure the system cycle clocks, select the CPU in the project navigator and select Target system > Expert > Set system cycle clocks in the menu. **25%**: The maximum duration of the task is 25% of the cycle clock. **50%**: The maximum duration of the task is 50% of the cycle clock. **75%**: The maximum duration of the task is 75% of the cycle clock.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

Isochronous I/O processing on fieldbus systems (Page 1383)

Timeouts and level overflows (Page 1360)

Assigning programs to the execution levels/tasks (Page 1345)

Sequence model for DCC blocks (DCB) (Page 1394)

Setting size of local data stack (Page 1685)

SystemInterruptTasks

SystemInterruptTasks are started and executed once when a system event occurs.

The following SystemInterruptTasks are available:

- **TimeFaultTask**: Starts in the event of a TimerInterruptTask timeout
- **TimeFaultBackgroundTask**: Starts in the event of a BackgroundTask timeout
- **TechnologicalFaultTask**: Starts in the event of an error on a technology object
- **PeripheralFaultTask**: Starts in the event of an error on the I/O
- **ExecutionFaultTask**: Starts in the event of an error when executing a program

Note

An exception in the DCC tasks does not call a SIMOTION fault task.

Starting a SystemInterruptTask

An internal system task checks approx. every 10 ms whether a configured event has occurred and then starts the applicable SystemInterruptTask automatically. The cycle clock for this internal system task is independent of the defined cycle clock for IPO or servo.

The SystemInterruptTask must be used in the execution system and a program must be assigned to this task. If this is not the case and an event occurs which starts this SystemInterruptTask, the CPU enters **STOP** state.

Up to eight different interrupts can be stored in the buffer. If another interrupt occurs, the buffer overflows and the CPU goes into **STOP** mode, too.

The events that caused a SystemInterruptTask to be called can be queried using the associated **TaskStartInfo** for this task and are described under Using Taskstartinfo (Page 1262).

Completing a SystemInterruptTask

A SysteminterruptTask is completed automatically after completion of the programs assigned to the SystemInterruptTasks.

You can assign the created and compiled programs to the selected SystemInterruptTask and define their execution sequence in the **Program assignment** tab.

You can set the following parameters:

Field/button		Meaning/information
For task		Use this to select one of the SystemInterruptTasks to which you want to assign the programs. You can assign several programs to one SystemInterruptTask.
	ExecutionFaultTask	Program processing error, for example, division by zero.
	PeripheralFaultTask	I/O alarms such as process alarms, diagnostic alarms, removing and inserting modules.
	TechnologicalFaultTask	Technological alarms such as alarms, warnings and notes.
	TimeFaultBackgroundTask	Timeout in the BackgroundTask.
	TimeFaultTask	Timeouts, for example in the TimerInterruptTasks, system runtime and cycle time.

Field/button	Meaning/information
Alarm configuration	This displays the window for the configuration of the technological alarms. You can configure the alarm response to a technological alarm for the SystemInterruptTasks.
Use task in execution system	Activate the checkbox to display and use the task in the execution system. If the checkbox is deactivated, you cannot assign any programs to this task.

TimeFaultTask

The **TimeFaultTask** is started when the time monitoring of a **TimerInterruptTask** responds. The CPU will switch to **STOP** mode if the TimeFaultTask is not configured or if there is no program assigned to it.

In the **TimeFaultTask**, you can program the response to timeouts of TimerInterruptTasks.

For additional information, see Using Taskstartinfo (Page 1262).

TimeFaultBackgroundTask

The **TimeFaultBackgroundTask** is started when the time monitoring of the BackgroundTask responds. The CPU will switch to **STOP** mode if the **TimeFaultBackgroundTask** is not configured or if there is no program assigned to it.

In the **TimeFaultBackgroundTask**, you can program how timeouts of **BackgroundTasks** are handled.

TechnologicalFaultTask

The **TechnologicalFaultTask** is started when the technology package generates an alarm or information message. The CPU will switch to **STOP** mode if the TechnologicalFaultTask is not configured or if there is no program assigned to it.

Generally, alarm messages have a direct effect on the behavior of the technology package and must be acknowledged before technology functions can be activated again.

In the **TechnologicalFaultTask** you can, for example, acknowledge errors directly and/or initiate further responses with respect to the sequence of operations on the machine.

For additional information, see Using Taskstartinfo (Page 1262).

PeripheralFaultTask

The **PeripheralFaultTask** is started immediately in accordance with its priority for I/O access errors.

I/O access errors can occur, for example, when the load voltage supply for the I/O module fails or other errors occur on the I/O module.

For further information, refer to the descriptions of the *I/O modules*.

The task for which an error occurred during its I/O access, *will not be terminated*.

The CPU will switch to **STOP** mode if the **PeripheralFaultTask** is not configured or if there is no program assigned to it.

For additional information, see Using Taskstartinfo (Page 1262).

ExecutionFaultTask

The **ExecutionFaultTask** is started immediately in accordance with its priority for program run errors.

Examples of program run errors:

- Faulty operations with floating-point numbers, such as logarithms of negative numbers, invalid numbers, etc.
- Division by zero
- Violation of array limits
- Error while accessing system variables

The task in which the error occurred is terminated.

The CPU will switch to **STOP** mode if the **ExecutionFaultTask** is not configured or if there is no program assigned to it.

The CPU to **STOP** error response is possible for all tasks and starts the **ShutdownTask**. The SIMOTION device switches to **STOP** mode.

An error response for the ExecutionFaultTask restarts the ExecutionFaultTask.

The following tasks can be restarted by commands in the program of the ExecutionFaultTask:

- StartupTask
- ShutdownTask
- MotionTasks

For the following tasks, the SIMOTION device switches to **STOP** mode once the ExecutionFaultTask is completed and the ShutdownTask is started:

- BackgroundTask
- TimerInterruptTasks
- SynchronousTasks

For additional information, see Using Taskstartinfo (Page 1262).

Note

Program errors in the **ExecutionFaultTask** and in the **ShutdownTask** switch the system to **STOP** mode immediately.

Configuring SystemInterruptTasks

1. Click **SystemInterruptTasks** in the Execution Levels tree.
2. Select one of the specified tasks in the **For task** selection box.

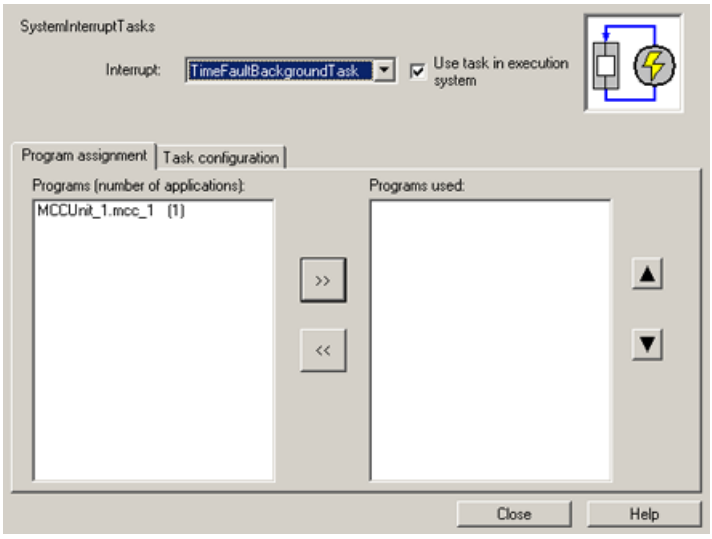


Figure 4-102 Configuration of the SystemInterruptTasks (program assignment)

3. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
4. Switch to the **Task configuration** tab.

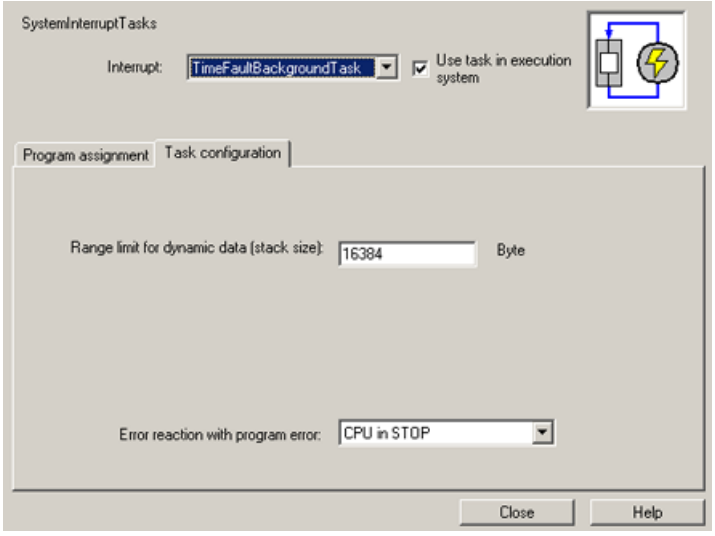


Figure 4-103 Configuration of the SystemInterruptTasks (task configuration)

5. Configure the task.
6. Repeat steps 3 to 5 for all the SystemInterruptTasks to be configured.
7. To configure the technological alarms:
   Click **Alarm configuration**.

Task configuration - SystemInterruptTasks

In the **Task configuration** tab, parameterize the error response to program errors.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

Information on starting a task: TaskStartInfo (TSI) (Page 1363)

UserInterruptTasks

UserInterruptTasks are intended for user-defined actions.

Two UserInterruptTasks are available: **UserInterruptTask_1** and **UserInterruptTask_2**.

A defined condition must be specified for the UserInterruptTask. Each time the condition is fulfilled, the UserInterruptTask is started.

If you want to use a **UserInterruptTask**, the **IPOsynchronousTask** must be used in the execution system.

UserInterruptTasks are *not active* during a **StartupTask** and a **ShutDownTask**.

Starting a UserInterruptTask

UserInterruptTasks are started automatically as soon as the user-defined interrupt condition is fulfilled. The interrupt condition is checked in the interpolator cycle clock.

If both UserInterruptTasks are started at the same time, **UserInterruptTask_1** is processed before **UserInterruptTask_2**.

Note

If a user-defined interrupt occurs during shutdown, the UserInterruptTask is not started anymore.

During shutdown (ShutdownTask), starting a UserInterruptTask is only possible using the _startTaskId() command.

Completing a UserInterruptTask

UserInterruptTasks are completed automatically after completion of the programs assigned to the UserInterruptTask.

You can assign the created and compiled programs to the selected UserInterruptTask and define their execution sequence in the **Program assignment** tab.

You can set the following parameters:

Field/button		Meaning/information
For task		Use this to select one of the two UserInterruptTasks to which you want to assign the programs. You can assign several programs to one UserInterrupt-Task.
	UserInterruptTask_1 and **UserInterrupt-Task_2**	Predefined names of the UserInterruptTask.
Defined condition		Here you define the condition for the selected UserInterruptTask according to IEC 61131. During operation, the specified condition is checked in the interpolator cycle clock. If the condition is fulfilled, the UserInterruptTask is started with the assigned programs. You enter the condition in the input field by using the symbol browser (variables via drag and drop) and the **Command library** tab in the project navigator (operators via drag and drop). Only simple conditions (e.g. logic operations of inputs and local CPU variables) and Boolean variables are permissible (see below).
Use task in execution system		Activate the checkbox to display and use the task in the execution system. If the checkbox is deactivated, you cannot assign any programs to this task.

Configuring UserInterruptTasks

1. Click **UserInterruptTasks** in the Execution Levels tree.
2. Select a UserInterruptTask in the **For task** selection box.

3. Specify the condition for starting this task.

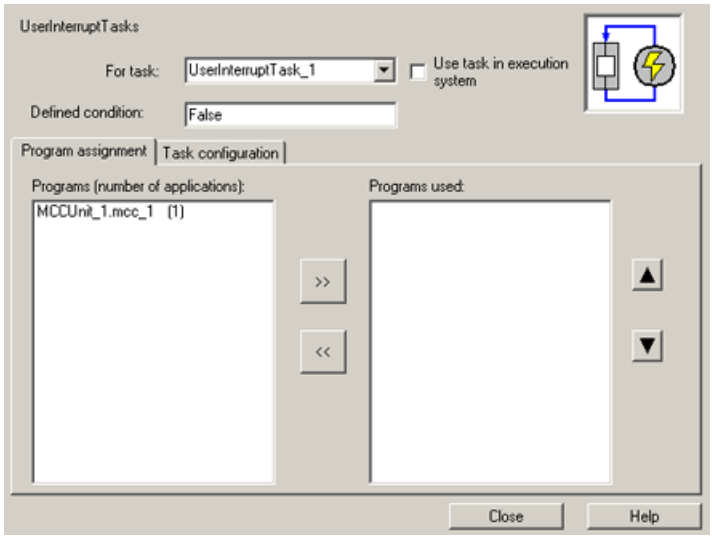


Figure 4-104 Configuration of the UserInterruptTasks (program assignment)

4. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
5. Switch to the **Task configuration** tab.

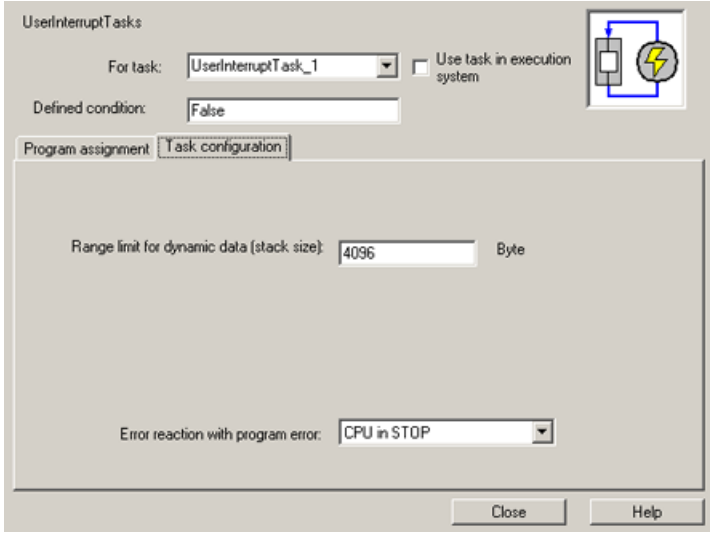


Figure 4-105 Configuration of the UserInterruptTasks (task configuration)

6. Configure the task.
7. Repeat steps 3 to 6 for the second UserInterruptTask.

Formulating a condition for a UserInterruptTask

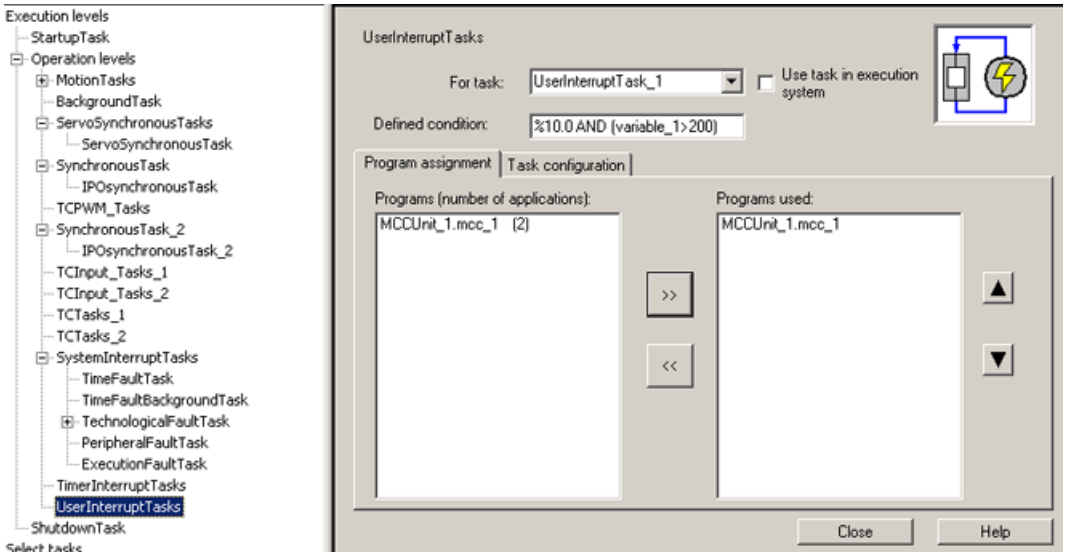


Figure legend - example:

When the digital input 0.0 = TRUE and the global variable "variable_1" has a value > 200.0, then the UserInterruptTask_1 is executed once.

Figure 4-106 Configuration of a UserInterruptTasks with condition

You enter the condition for starting a UserInterruptTask as a formula according to IEC 61131 (Structured Text). You can use the following variables:

- I/O variables
- Global device variables
- Unit variables in the interface section of a program source (ST/MCC or LAD/FBD unit). Syntax for the use of a *var_name* unit variable from the *src_name* program source:
  - Up to and including SIMOTION Kernel V4.3:
    Only specification of the variable name <VariableName>.
  - As of SIMOTION Kernel V4.4:
    The program source must also be specified in the form _device \<UnitName>.<VariableName>
    (example _device\mySourceFile.myVar).
    As previously, global device variables can only be specified directly via the variable names.

SIMOTION SCOUT helps you create the formula.

To formulate a condition for starting a UserInterruptTask:

1. Under **For task**, select the UserInterruptTask for which you want to define the start condition.
2. Take the required operators from the **Command library** tab and insert them in the **Defined condition** text field using drag and drop.
   Alternatively, you can enter the operators directly in the text field.
3. Take the operands from the symbol browser and insert them in the **Defined condition** text field using drag and drop or copy and paste. Alternatively, you can enter the operands directly in the text field.

Creating conditions with the command library

The **Command library** tab contains a list of mathematical operations and functions. You can use these to create conditions.

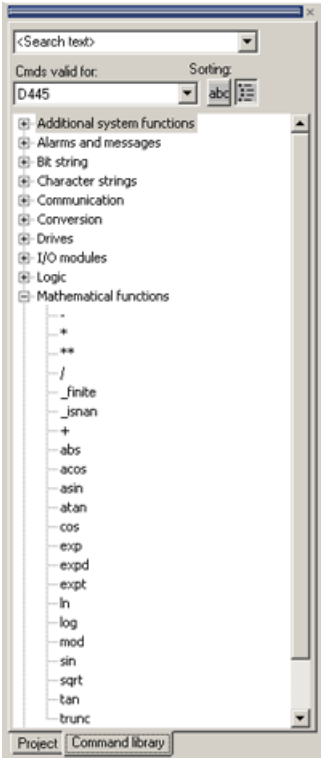


Figure 4-107 Command library in SCOUT

To open the individual operations, click the plus sign in front of the folders. You can drag the operators to the required text field via drag and drop. You can use these commands, for example, to define conditions.

Note

The following applies when updating the version of SIMOTION Kernel as of V4.4:

If you use unit variables of a program source in the start condition of a UserInterruptTask, you may have to adapt the syntax of the start condition.

Task configuration - UserInterruptTasks

In the **Task configuration** tab, parameterize the error response to program errors.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode and the ShutdownTask is started.
	ExecutionFaultTask	The ExecutionFaultTask is started. All programs assigned to this task are started. If no programs are assigned, the CPU switches to **STOP** mode. The task in which the error occurred is terminated.

See also

Assigning programs to the execution levels/tasks (Page 1345)

ShutdownTask

The **ShutdownTask** is intended for selective intervention in the transition from **RUN** to **STOP U**/**STOP** mode or for programming stop sequences with a preassigned braking ramp, e.g. selected setting of outputs, defined shutdown of axes.

The ShutdownTask is *not* called in the case of a power failure.

The **time monitoring** must be specified in the **Task configuration** for the ShutdownTask: You can configure the maximum duration for the execution of the ShutdownTask; 0 ms = no monitoring. After this time, the CPU switches to **STOP** mode.

The I/O can be accessed directly in the ShutdownTask. Access to process image and symbolic I/O variables is restricted. It is *not practical* to access the I/O by means of the process image, as the process image is no longer being updated.

For additional information, see **SIMOTION ST Structured Text**, "Access to inputs and outputs (process image, I/O variables)"

Select the **Program assignment** tab to assign the created and compiled programs to the ShutdownTask and define their execution sequence.

Note

Program errors in the **ExecutionFaultTask** and in the **ShutdownTask** switch the system to **STOP** mode immediately.

Configuring the ShutdownTask

1. Click **ShutdownTask** in the Execution Levels tree.

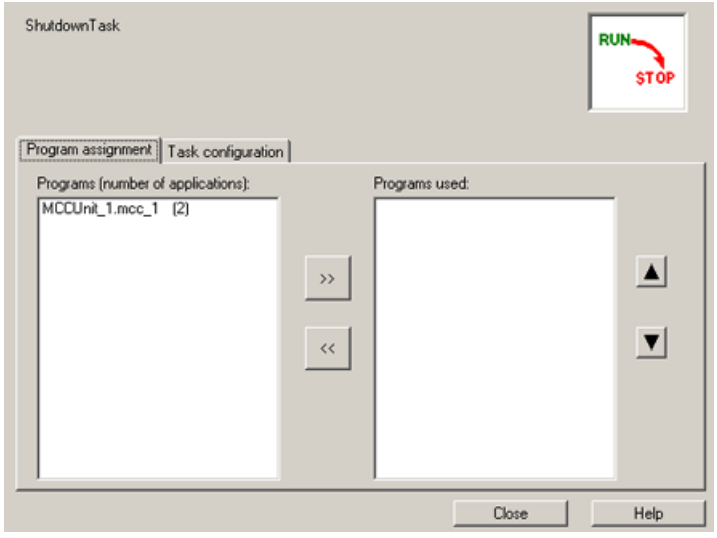


Figure 4-108 Configuration of the ShutdownTask (program assignment)

2. In the **Program assignment** tab, assign the required programs to this task and define the execution sequence.
3. Switch to the **Task configuration** tab.

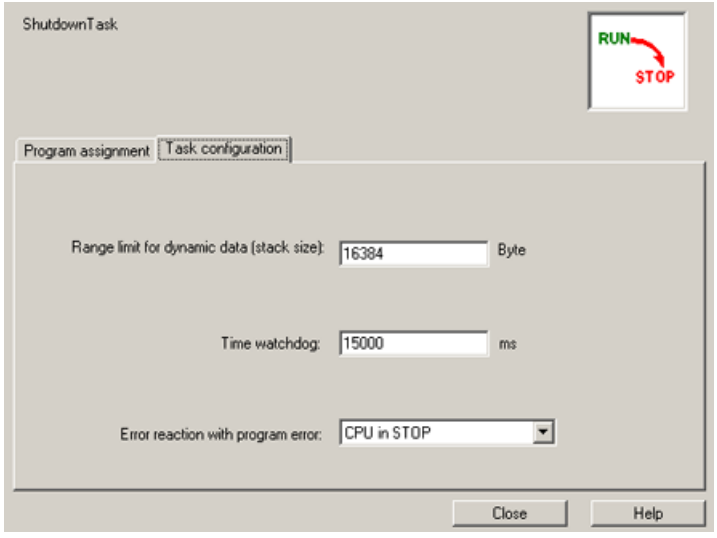


Figure 4-109 Configuration of the ShutdownTask (task configuration)

4. If required, enter the **Range limit for dynamic data (stack size)**.
5. Enter a value for the **time monitoring**.
6. Specify the **Error response to program error** (e.g. **ExecutionFaultTask**).

Task configuration - ShutdownTask

Select the **Task configuration** tab to parameterize the time monitoring.

You can set the following parameters:

Field/button		Meaning/information
Range limit for dynamic data (stack size)		Enter the local data stack size for this task in bytes. When the programs assigned to this task are executed, this size is made available for data in the stack. The guide value is 16 KB for a task. See also: • Setting the size of the local data stacks (Page 1685). • "Memory requirement of the variables on the local data stack" in the SIMOTION ST Programming Manual.
Time monitoring		Specify the cycle time in ms for executing the ShutdownTask. Enter a value for time monitoring. The time monitoring is inactive if you enter 0 or no value.
Error response to program error		Select the error response for errors that occur while processing programs. Program errors are, for example, faulty operations with floating-point numbers, division by zero, and overshooting array limits.
	CPU to STOP	The CPU switches to **STOP** mode.

See also

Assigning programs to the execution levels/tasks (Page 1345)

Watchdog (Page 1363)

4.2.7.3 Configure execution system

Configuration of the execution system involves the following steps:

- Assignment of user programs and definition of the task properties
- Enabling of the tasks used
- Selection of the cycle clock source and setting of the system cycle clocks

Assigning programs to the execution levels/tasks

The programs must be assigned to execution levels. Only then are the programs executed.

After creation, you can assign the programs of an MCC source, an ST source or a LAD/FBD source to one or more tasks with SIMOTION SCOUT.

You can assign more than one program to a task.

The assigned programs are then executed in the order in which they are listed; this order can be specified and modified using SIMOTION SCOUT.

If several programs are assigned to a task, execution of the first program must be finished before the next program in the same task can be started. If, for example, the first program is in a continuous loop, the second program will never be executed.

You can also assign one program to several tasks, which are then executed independently of each other.

With the program assignment, you specify the following:

- Priority with which the programs are executed
- The execution behavior: sequential/cyclic
- The initialization behavior of program variables
  See the **SIMOTION MCC** or **SIMOTION ST** Programming Manual - "Time of variable initialization" and Influence of the compiler on variable initialization (Page 1419).

Please note the following when assigning a program to one or more tasks:

- Before programs can be assigned, they must be compiled without error.
- The assignment must be made before downloading the program to the target system.
- It is possible that the program (if multiple tasks are assigned) may be called by another task while it is being executed. No measures are taken in the system to ensure data consistency.
- Once you have assigned a program to a task, it will remain assigned even in the event of recompilation.

Note

DCC tasks are assigned via the DCC editor, see the description of the DCC editor and Sequence model for DCC blocks (DCB) (Page 1394).

Execution system - program assignment

You can assign the created and compiled programs to the various tasks of the execution levels in the **Program assignment** tab.

You can set the following parameters:

Field/button	Meaning/note
Programs (number of uses)	A list of all compiled programs that are available in the project is displayed here. Non-compiled programs are not displayed. The number after the program name indicates how often the program has been assigned to the different tasks of the execution levels. MCC charts and LAD/FBD programs are displayed immediately after entering as long as they have been entered as "exportable".
›› **Assign**	Use this to assign selected programs to the task. 1. Select the program in the "Programs" list. 2. Click the ›› button or drag-and-drop the program into the "Used programs" list. The program is assigned to the task and displayed in the list of used programs.
‹‹ **Remove**	Use this to remove programs assigned to a task. 1. Select the program in the "Used Programs" list. 2. Click the ‹‹ button or drag-and-drop the program into the "Programs" list. The program is removed from the task.
Used Programs	A list of all programs assigned to this task is displayed here. The sequence of the programs in the list corresponds to the execution sequence when the program is executed. The program at the top of the list is executed first.

Field/button	Meaning/note
Arrow up	Use the arrow to move the selected program up one position within the task. In this way you can determine the order of program execution within the task.
Arrow down	Use the arrow to move the selected program down one position within the task. In this way you can determine the order of program execution within the task.

Assigning programs to the tasks

1. Select the SIMOTION device in the project navigator and select **Target system > Configure execution system** in the menu or double click **EXECUTION SYSTEM**.
   The **EXECUTION SYSTEM** window opens in the working area of the workbench.

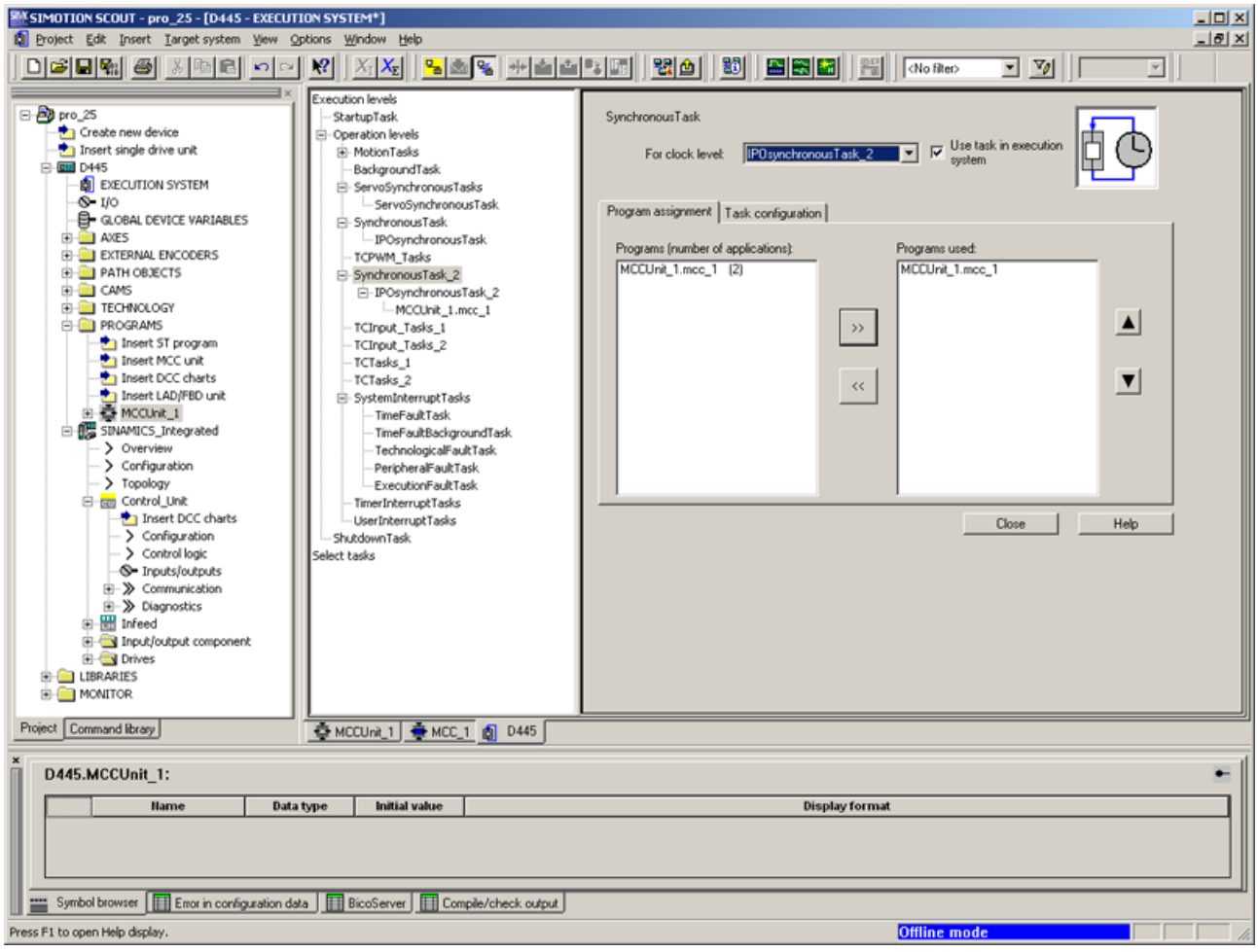


Figure 4-110 Configuration of the execution system in SCOUT

In the left-hand pane of the window, you can see the execution levels tree. It displays as entries the execution levels / tasks for which **Use task in execution level** has been clicked. The **OperationLevels** folder contains the tasks that are available in the **RUN** mode. The list below each execution level or task name shows the configured tasks and the programs assigned to them.

2. Select the task to be configured.

3. Select the **Program assignment** tab.
   The left-hand list box lists all available programs (ST programs, MCC charts and LAD/FBD with Program creation type).
4. Select the programs in the list box on the left that you want to assign to the task.
5. Click ».
   The assigned programs are listed in the right-hand list box.
   They are still listed in the left-hand list box. You can assign programs to several tasks. The number of assignments made appears in brackets.
6. Select the **Task configuration** tab and make any additional settings, such as:
   - Error response to program error
   - Time watchdogs for cyclic tasks
   - Start behavior of MotionTasks

   See the detailed description for the relevant task in Section "Description of the user program tasks (Page 1314)".

After you have assigned a program to one or more tasks, you can establish the connection to the target system, download the project to the target system, and start it.

Change execution sequence

Programs are executed in the order entered. You can change this order.

1. Select the entry to be moved in the right-hand list box.
2. Click ▲ or ▼ to move the element up or down.
3. Repeat Step 2 as often as required.

Task names

The names of the MotionTasks can be changed.

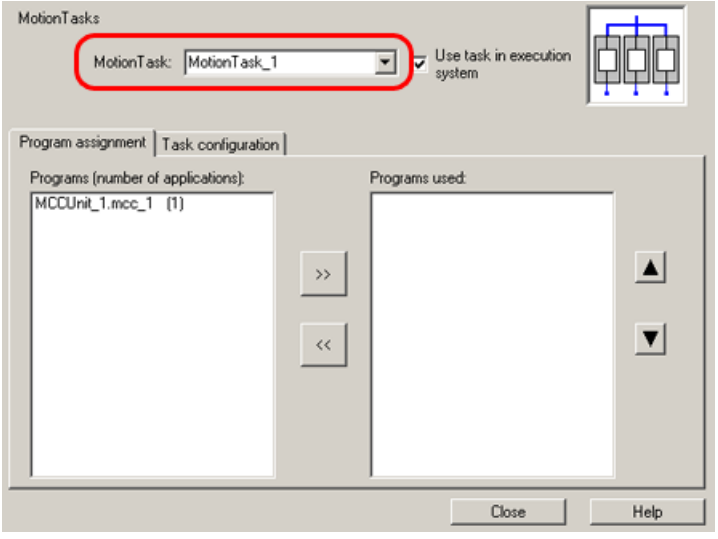


Figure 4-111 Task names in SCOUT

1. Select the desired task on the left-hand side.
2. Enter the desired new task name in the drop-down list on the right-hand side.

Task control

Various commands are available for the task control (e.g start, stop, etc.), see Section "Task control commands (Page 1435)". See also Task commands in the SIMOTION MCC programming language

Information is contained there about how to use these task control commands and several examples.

Stack size

In SIMOTION SCOUT, the stack size (limit for dynamic data) of the associated task can be set on the Task configuration tab of the Task Configuration window. A default value is specified for each task.

For further information, refer to the **SIMOTION ST** Programming Manual, "Setting the size of the local data stack".

Assignment of the program to a MotionTask

In the program assignment, you can see which MotionTasks the program is assigned to via a tooltip.

See also

Task priorities (Page 1307)

Selecting the cycle clock source

Selection of the cycle clock source is implemented in the device configuration in the **HW configuration**.

As soon as you parameterize one of the interfaces as isochronous DP/PN interface (select **"Isochronous mode"** in the Properties dialog box), the cycle clock setting is used as the bus cycle clock.

The DP/PN communication level and servo and interpolator levels are synchronized with the bus cycle clock. This setting is essential if you want to utilize the motion control functions of the **TP CAM/TP PATH/TP CAM_ext** technology package in combination with digital drives in accordance with PROFIDRIVE V3 on PROFIBUS DP/PROFINET. Drives that support and require this communication are, for example: SIMODRIVE 611U, MASTERDRIVES MOTION CONTROL, SINAMICS. Synchronization to the bus cycle clock if you must isochronously access I/O from your application.

You can still operate drives that do not support isochronous mode as drive axes on the isochronous bus. Examples of these drives are Micromaster MM4 and MASTERDRIVES VC.

If you are not using an isochronous DP/PN interface, you can set the basic system cycle clock. The servo and interpolator levels are synchronized to the basic cycle clock. You can select this setting if you are not using the **TP CAM/TP PATH/TP CAM_ext** technology package or are using it exclusively with analog drives on SIMOTION.

Defining system cycle clocks

Once the cycle clock source is selected, you specify the sampling times of the isochronous execution levels derived from the basic cycle clock. An additional faster servo cycle clock option and an IPO option can be configured as of V4.2, see also Servo_fast (application cycle clock for fast bus system) (Page 1364).

Included here:

- **Bus cycle clock (DP cycle clock / PN cycle clock)**: refer to the table for the execution system - system cycle clocks
  The "base cycle clock" is the basic cycle clock if no isochronous DP/PN interfaces have been parameterized in HW Config. The "bus cycle clock" is based on the equidistant bus cycle and is set in HW Config. The designation changes in accordance with the setting in the **Set system cycle clocks** dialog. The basic/bus cycle clock (DP cycle clock or PN cycle clock) is used as a basis for setting further tasks.
- **Servo_fast (fast position control cycle clock (as of V4.2))**
  Inputs/outputs are updated in the Servo_fast cycle clock (optional). Servo_fast is assigned to the faster bus system (PROFINET IO) and set in HW Config.
- **IPO_fast (fast interpolator cycle clock (as of V4.2))**
  The motion control for the axes on the fast bus system is calculated in the IPO_fast cycle clock (optional).
- **Servo cycle clock (position control cycle clock) / T1(DCC) cycle clock**
  Inputs/outputs are updated in the servo cycle clock.
  The position control for the axes and the processing of the centralized and distributed I/O are carried out in the servo cycle clock. The servo cycle clock can be operated relative to the bus cycle clock at a ratio of 1:1 or 1:2. The **ServoSynchronousTask** is being processed in this clock cycle.
  For isochronous PROFINET IO, the following applies:
  - The DP master interface must run in the same cycle clock as the servo cycle clock.
  - For SIMOTION C, P and D, the bus cycle clock for the external PROFIBUS interfaces must be at least 1 ms in order for these to be operated isochronously.
  - If the PN cycle clock is scaled down (e.g. 0.5 ms), the servo cycle clock must then be the same as the PROFIBUS cycle clock (for SIMOTION D: also the same as the bus cycle clock of PROFIBUS integrated).

  Note

  The reduction ratio between the bus and servo cycle clocks must also be set as the master application cycle on the drive. This setting is necessary to enable reciprocal life-sign monitoring. For further information, refer to the drive descriptions.
- **Interpolator cycle clock (IPO cycle clock) / T2(DCC) cycle clock**
  The axis motion control is calculated in the IPO cycle clock.
  The **IPOSynchronousTask** is executed in this cycle clock. The IPO cycle clock can be reduced relative to the servo cycle clock at a ratio of 1:1 to 1:6.
- **Interpolator cycle clock 2 (IPO_2 cycle clock) / T3(DCC) cycle clock**
  The IPO cycle clock 2 is the basis for motion control of lower-priority axes.
  The **IPOSynchronousTask_2** and the **PWM Task** (TControl) are executed in this cycle clock.
  The IPO_2 cycle clock can be reduced relative to the IPO cycle clock at a ratio of 1:2 to 1:64.

- **DCCAux T4(DCC) - cycle clock**
  The DCCAux cycle clock can be reduced relative to the T3(DCC) cycle clock at a ratio of 1:2 to 1:32.
- **DCCAux_2 T5(DCC) cycle clock**
  The DCCAux_2 cycle clock can be reduced relative to the DCCAux cycle clock at a ratio of 1:2 to 1:32.
- **PWM cycle clock**: For the TControl technology package

Note

The settings of the system cycle clocks affect the system operating sequence considerably. You must therefore exercise caution when making these settings.

Set system cycle clocks

1. Call the configuration window in the main menu via **Target system > Expert > Set system cycle clocks...**, or in the context menu of the device, or in the **EXECUTION SYSTEM** context menu via **Expert > Set system cycle clocks**.

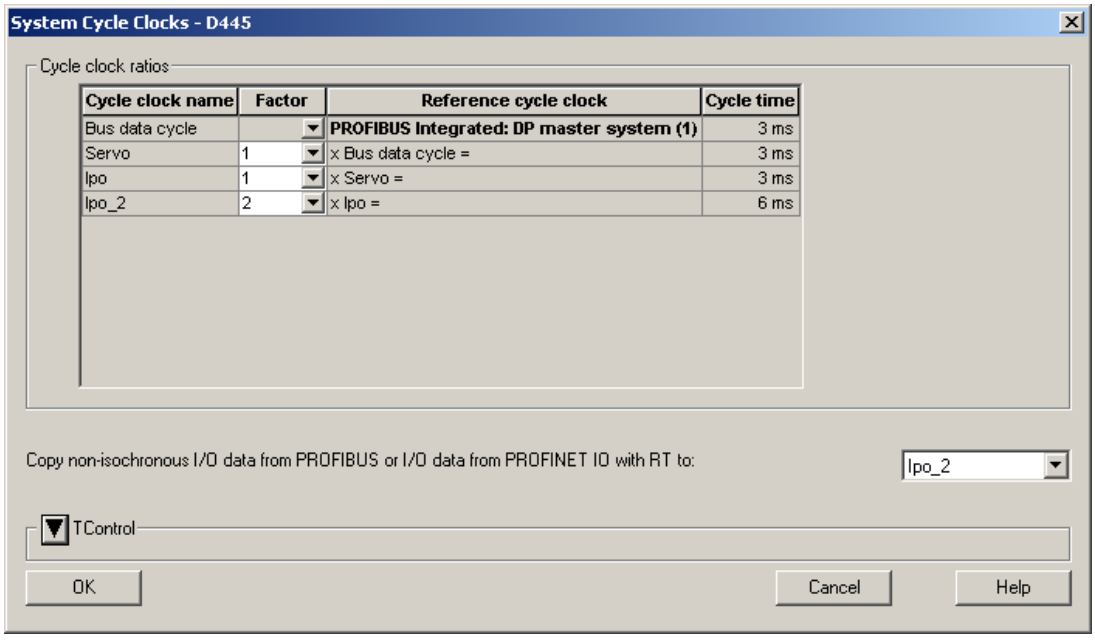


Figure 4-112 Setting of the system cycle clocks in SCOUT

2. Specify the basic/bus cycle clock (only if the isochronous mode is not configured). The bus cycle clock is specified in HW Config for the equidistant bus.
   Possible values: 1.0 ... 8.0 ms for PROFIBUS
   Possible values for PROFINET:
   - As of V4.0 of SIMOTION P and SIMOTION D4x5/D4x5-2: 0.5 ... 4.0 ms
   - As of V4.1 for SIMOTION P: 0.25 ms to 4 ms.
   - As of V4.2 for SIMOTION D: 0.25 ms to 4 ms.
   - As of V44 for SIMOTION D455-2 DP/PN: 0.125 ms to 4 ms.

   The cycle time for PROFIBUS must be an integer multiple of 0.125 ms, or of 0.25 ms in the case of SIMOTION C.
   The cycle time for PROFINET must be an integer multiple of 0.125 ms. Change the value in **HW Config** if this is not the case.
3. Specify the ratio between the cycle clocks.

Note

If the basic/bus cycle clock has been set too small for a large hardware configuration, this can result in the CPU not switching to the RUN mode. In this case, note the entries in the diagnostics buffer for timeouts.

Set system cycle clocks with two servo cycle clocks (Servo_fast)

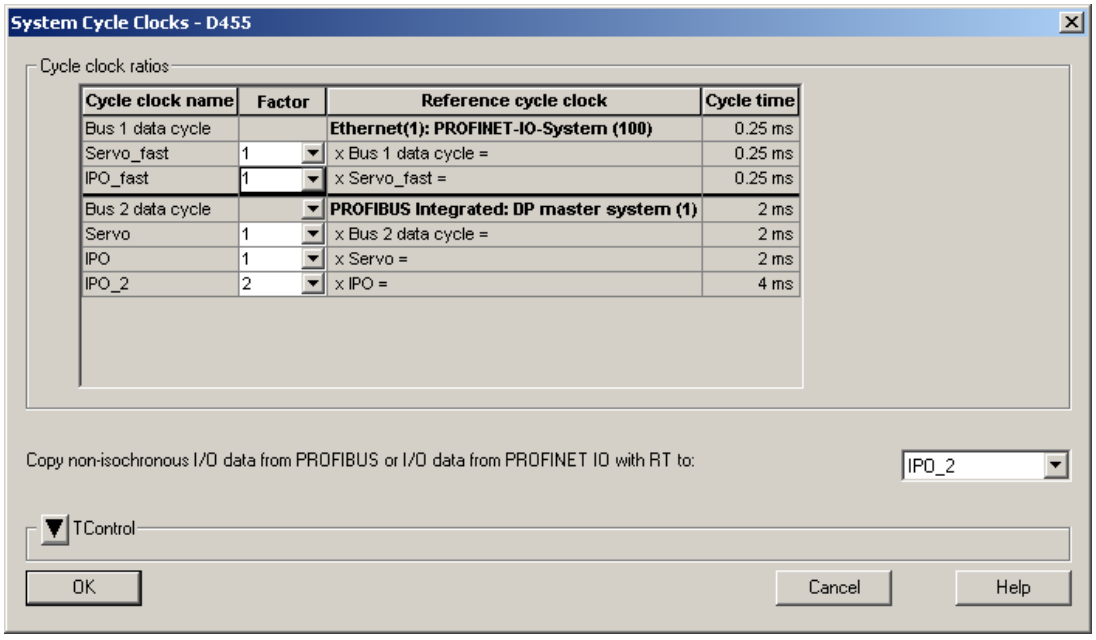


Figure 4-113 Servo and Servo_fast system cycle clocks

A description of how to configure a second servo cycle clock can be found in the *Communication* Function Manual under Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface.

System cycle clocks for DCC

The entries for DCC are displayed as soon as a chart has been added under programs. DCC charts are started automatically when the appropriate task is started.

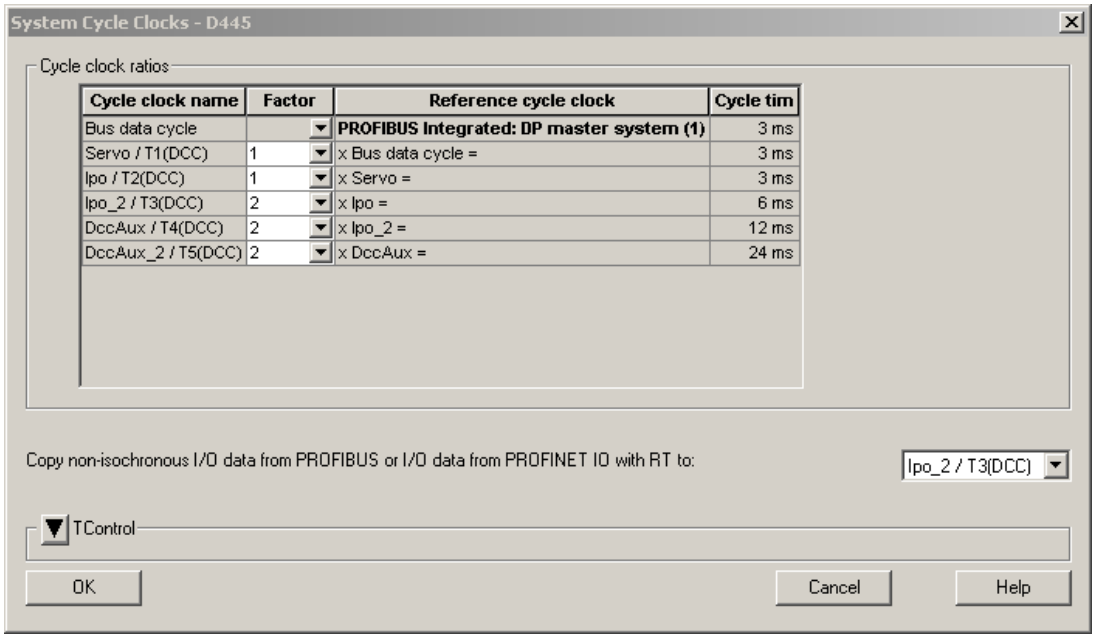


Figure 4-114 Setting of the system cycle clocks in SCOUT with DCC

Setting the system cycle clocks for TControl

1. You can use the **TControl** button in the **System Cycle Clocks** configuration window to display the settings for the TControl system tasks.
2. You can use the **Use system tasks for TControl** checkbox to activate or deactivate the TControl system tasks.
   If **Use system tasks for TControl** is activated, the special tasks for the temperature channels are displayed in the execution system.
   If you have not configured a temperature channel, you should deactivate the system tasks for TControl as they require unnecessary computation time.
3. Furthermore, you can specify here the cycle clock ratios of the TControl tasks.
   - The **PWM** cycle clock must be an integer multiple of the position control cycle clock.
   - The cycle clock of **InputTask_1/2** depends on the PWM cycle clock.
   - The cycle clock of **PostTask_1/2** depends on the cycle clock of **InputTask_1/2**.

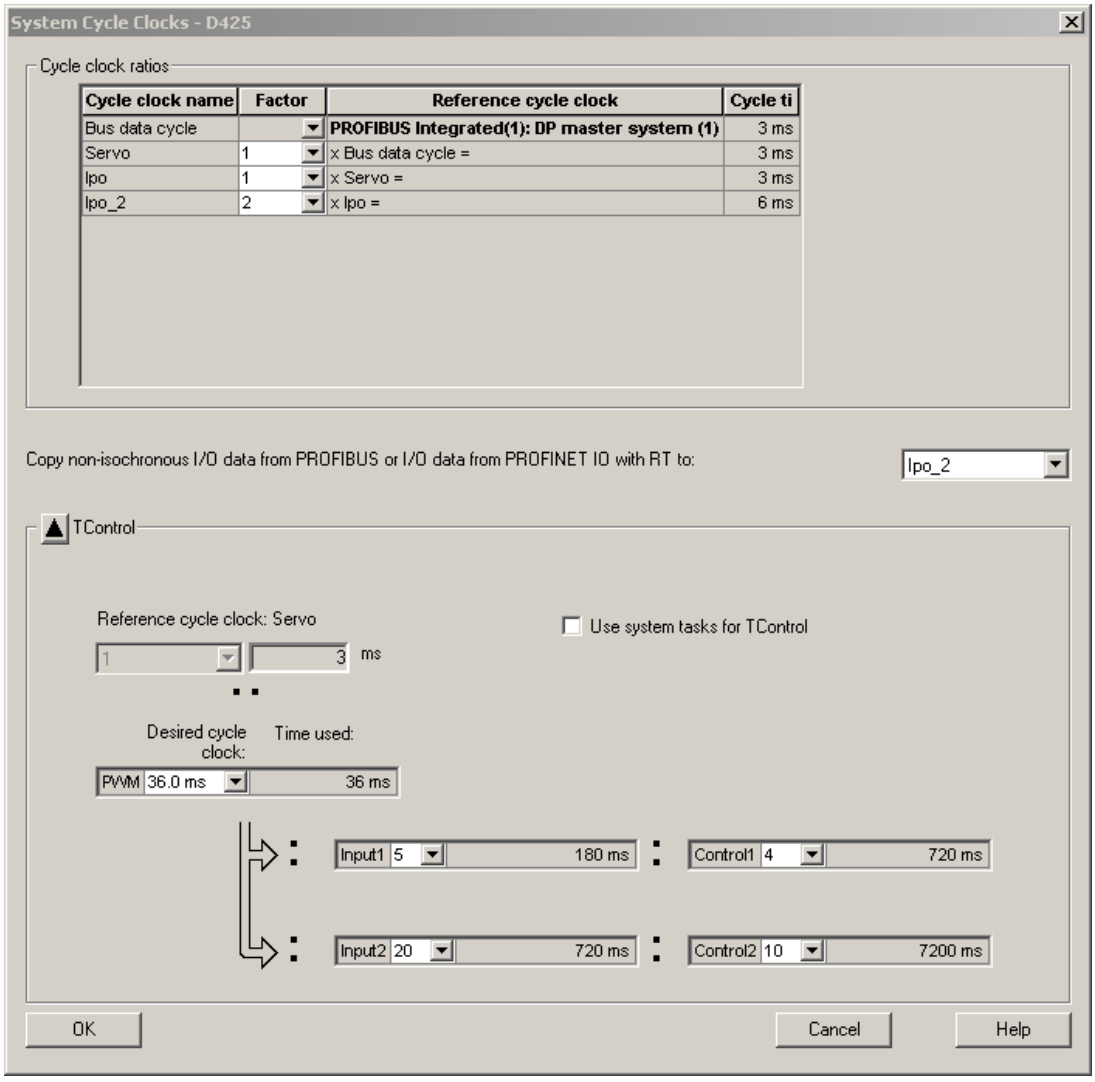


Figure 4-115 Setting of the system cycle clocks for TControl in SCOUT

Execution system - system cycle clocks

You can set the following parameters:

Field/button		Meaning/information
Cycle clock ratios		
	Basic cycle clock or **bus cycle clock**	The base cycle clock is the basic cycle clock if no isochronous DP/PN interfaces have been parameterized in HW Config. The "bus cycle clock" cannot be set in this dialog box if the "Equidistant bus cycle" setting is activated. The "bus cycle clock" is set in HW Config. This means the equidistant bus cycle forms the basic cycle clock for the system cycle clocks. The basic/bus cycle clock (DP cycle clock or PN cycle clock) is used as a basis for setting further tasks.
	Servo_fast	You can set the Servo_fast cycle clock here (optional). The reference cycle clock is the isochronous PROFINET IO cycle clock (for the faster bus system) set in HW Config.
	IPO_fast	You can set the IPO_fast cycle clock here. The reference cycle clock is the Servo_fast cycle clock.
	Servo / T1(DCC)	You can set an integer ratio between the servo cycle clock and the bus/Servo_fast cycle clock here. Among others, the position control of the axes is calculated in this cycle clock. Normally the factor 1 should be used. If you set the factor 2, although the dynamic performance of the controller will deteriorate, more computing time will be available for processing other tasks. The reduction ratio between the bus cycle and the servo cycle clock must also be set as the "master application cycle" on the drive. This setting is necessary to enable reciprocal life-sign monitoring. For further information, refer to the drive descriptions. If isochronous mode is not configured, the default setting for the basic time to servo cycle clock ratio is 1:1. It cannot be changed then! PROFIBUS DP: The servo cycle clock can be operated at a ratio of 1:1, 1:2, 1:3, and 1:4. PROFINET IO: The servo cycle clock can be operated at a ratio of 1:1, 1:2, 1:3, 1:4, 1:6, 1:8, 1:10, 1:12, 1:14, and 1:16.
	Ipo / T2(DCC)	You can set an integral ratio between the IPO cycle clock and servo cycle clock here. As standard, the motion control of the axes will be calculated in the interpolator cycle clock. The interpolator cycle clock factor is used to determine how fast the drive setpoints are calculated.
	Ipo2 / T3(DCC)	You can set an integral ratio between the IPO_2 cycle clock and IPO cycle clock here. The interpolator cycle clock 2 is used for the motion control of low-priority axes. The factor is used to determine how fast the setpoints of the low-priority drives are calculated.
	DCCAux T4(DCC)	You can set an integer ratio between the DCCAux DCC cycle clock and T3(DCC) DCC cycle clock here.
	DCCAux_2 T5(DCC)	You can set an integral ratio between the DCCAux_2 DCC cycle clock and DCCAux DCC bus cycle clock here.
TControl		Click the arrow to open the configuration of the system tasks for the TP TControl. There, you can set the ratio of the cycle clocks for the task of the temperature channels to the servo cycle clock.

Field/button		Meaning/information
	Use system tasks for TControl	Activate the checkbox if the special tasks for the temperature channels in the execution system should be displayed. If you have not configured a temperature channel, you should deactivate the system tasks for TControl as they require unnecessary computation time.
	Servo Reference Clock Cycle	The servo cycle clock is displayed here for information purposes. To change this, you must select the value further up in the dialog box. All other cycle clocks of the temperature channel are a multiple of the servo cycle clock.
	PWM (Pulse Width Modulation)	Cycle clock which is used for the actuating signal output of the temperature channel. Specifies the basic cycle clock for further cycle clocks. Select the cycle clock in ms or in Hz. If you select Hz, the ratios to the input cycle clock and the control cycle clock are specified automatically.
	Input1/2	Cycle clock which is used for the actual value measurement of the temperature channel. Select the ratio of this cycle clock to the PWM cycle clock. The duration is displayed in the grayed-out field.
	Control1/2	Cycle clock which is used for the closed-loop control of the temperature channel. Select the ratio of this cycle clock to the input cycle clock. The duration is displayed in the grayed-out field.

Copying asynchronous cyclic I/O data

As of V4.2, you can choose when the asynchronous cyclic I/O data is to be copied. This means you can now choose whether data is copied in the IPO (with DCC T2 (DCC)) or IPO_2 (with DCC T3 (DCC)).

- To do this, select the relevant cycle clock from the drop down list box **Copy non-equidistant I/O data from PROFIBUS or I/O data from PROFINET IO with RT to**.

The inputs are always copied at the start of the first task on a level. The outputs are always copied at the end of the last task on a level. If there is no DCC or user program task, the BackgroundTask serves as both the first and the last task on the level concerned.

Note

Copied are all non-equidistant data on the **PROFIBUS DP**, **PROFINET IO**, and**I/O bus**.

Assigning system cycle clocks to technology objects

Assigning system cycle clocks

You can make better use of the performance reserves of the SIMOTION CPU by assigning priorities to your technology objects and specifically assigning them to system cycle clocks. The definition of lower-priority tasks allows you to gain performance reserves for higher-priority tasks.

Consequently, a higher time level can be used for the TOs, particularly for uniform motions without large accelerations/decelerations. (e.g. call of the TO in the interpolator cycle clock 2).

Change the standard setting if you detect:

- A job processing that lasts too long
- Too great a utilization of the technology in the CPU

Assign the interpolator cycle clock 2 to the axis and external encoder technology objects with lower-priority tasks, and assign the IPO cycle clock or the interpolator cycle clock 2 to the output cam and the measuring input technology objects.

Assign the interpolator cycle clock or the servo cycle clock to the technology objects with high-priority tasks.

You can assign the following cycle clocks to the technology objects:

Motion control task	**High priority**	**...**	**Lower priority**
Technology object	Servo cycle clock	Interpolator cycle clock	Interpolator cycle clock 2
Drive axis	(x)	Standard	X
Position axis	(x)	Standard	X
Following axis	(x)	Standard	X
External encoder	(x)	Standard	X
Output cam	X	X	X
Cam track	X	X	X
Measuring input	X	X	X

The system cycle clocks for output cams and measuring inputs, as well as for axes, are set in SIMOTION SCOUT in the **Configuration** dialog.

As of V4.1 in exceptional cases the IPO share can also be calculated in the servo, in this regard see **Motion control / interpolator** in the **TO axis** manual.

See also

Second position control cycle clock (Servo_fast) (Page 1364)

Task runtimes

You can use the task runtimes to check whether the computer performance of the system is sufficient for the demands of the application.

The task runtimes are displayed in the **Taskruntime** and **effectiveTaskruntime** device variables.

A distinction is made between:

- The runtimes of the individual tasks within a level (**Taskruntime**)
  The **Taskruntime** displays the sum of the net runtimes of the task (without the interrupt times).
- The runtime of the level (**effectiveTaskruntime**)
  The **effectiveTaskruntime** displays the effective runtime of a level (including the interrupt times). This is the time from the start of the level until the end of the last task in the level.

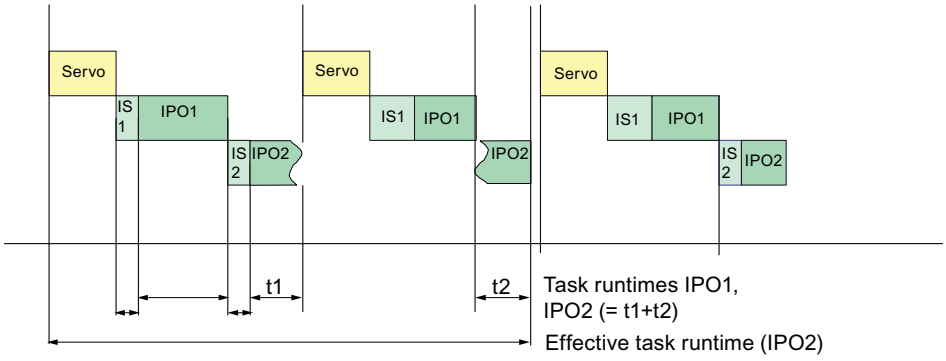


Figure 4-116 Representation of the Taskruntime and effectiveTaskruntime (cycle clock ratio IPO1 : IPO2 = 2)

Legend:

Servo: Servo cycle clock

IS1: IPOSynchronousTask

IS2: IPOSynchronousTask_2

IPO1: IPOTask

IPO2: IPOTask2

The following table shows the tasks and the levels for which a Taskruntime (tasks) and an effective Taskruntime (levels) can be determined.

Tasks (Taskruntime)	Levels (effective Taskruntime)
servodcc	Servo / T1(DCC)
servo	
servosynchronous	
servo_fast	Servo_fast
Ipodcc	Ipo / T2(DCC)
ipo	
iposynchronous	
ipo_fast	IPO_fast
ipodcc_2	Ipo_2 / T3(DCC)
ipo_2	
Iposynchronous_2	
dccaux	DCCAux T4(DCC)

Tasks (Taskruntime)	Levels (effective Taskruntime)
dccaux_2	DCCAux_2 T5(DCC)
background	Background[1]

[1] With "Background" level: Difference between 2 start times of the BackgroundTask

Note

To determine the task runtimes, the system variable **taskRuntimeMonitoring** must be set to **enable**.

The task runtimes are also shown in the SCOUT device diagnostics.

Note

Even when the module is in the **Stop** operating state, system tasks use computing time.

See also

Functions for runtime measurement of tasks - overview (Page 1446)

Optimizing the execution system (Page 1688)

Efficient programming - overview (Page 1686)

Timeouts and level overflows

Timeouts and/or level overflows may occur when executing computing processes. To monitor task runtimes, you can use the device diagnostics in SCOUT (ONLINE only). The evaluation of system variables is described in **Monitoring of timeouts and level overflows** (see below).

You have the possibility of configuring various responses for the system response to overflows of certain execution levels.

Timeout

You can configure maximum execution cycles for the execution levels **BackgroundTask**, **TimerInterruptTask**, **SynchronousTasks** and **ShutdownTask**.

If the set maximum duration is exceeded, the associated **SystemInterruptTask** (**TimeFaultTask** or **BackgroundFaultTask**) or a standard function (e.g. CPU to **STOP**) can be called up.

Level overflow

A toleration of level overflows can be set for the execution levels **IPO1** and **IPO_2**.

A level overflow occurs when a level (**IPOTask/IPOSynchronousTask** or **IPOTask2/ IPOSynchronousTask_2**) has not been executed within the set system cycle clock, IPO or IPO_2.

Note

A cyclic user task (IpoSync, Ipo2Sync) must be finished within its cycle.

A task overflow caused by a failed start attempt of a lower priority task produces an entry in the diagnostic buffer.

If the clock cycles be set too low so that level overflows continuously occur, the CPU will no longer function.

With a set tolerance, the next IPO cycle clock is used in order to:

- Complete the computing process of the previous cycle clock without triggering the set error response
- Start the computing process of the current cycle clock.

If another level overflow occurs in the next level cycle (if no tolerance is set or the number has been exceeded), the CPU is set to **STOP** in order to prevent a blockade of the entire system. This results in an entry in the diagnostic buffer and the error response (CPU to **STOP**) with starting lockout.

You can set a maximum of 5 level overflows to be tolerated in the task configuration in the execution system.

Examples

The following are examples of the execution relationships for level overflows:

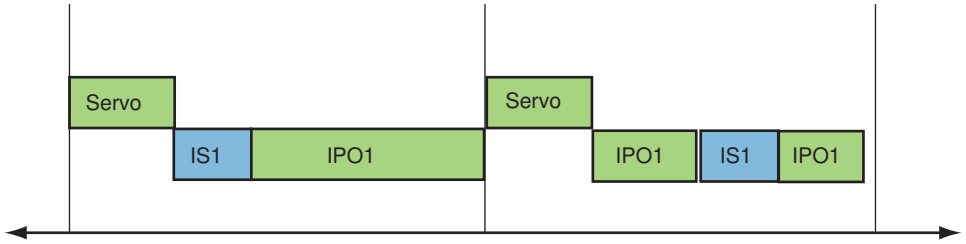
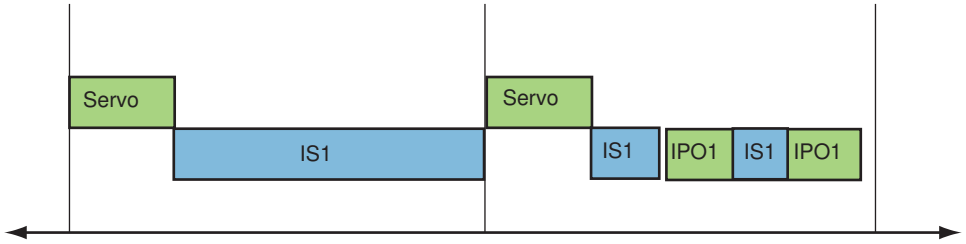


Figure 4-117 IPO overflow example



Servo: Servo cycle clock
IS1: IPOSynchronousTask
IPO1: IPOTask

Figure 4-118 Overflow of the IPOSynchronousTask example

High-performance programming

If level overflows occur, but it is not possible to increase the DP/servo cycle clock, the following measures can be taken:

- In task configuration for IPOSynchronousTask
  Ratio of IPOSynchronousTask : IPO cycle clock = 75%
  Tolerate two IPO overflows
- Use optimized PROFIBUS
  User-defined profile: HSA=2 (highest PROFIBUS address)
  with, for example, two nodes on the PROFIBUS, RetryLimit = 1, switch off cyclic distribution of the bus parameters (then, however, a PG can no longer be connected to the isochronous PROFIBUS)
- Change ratio BackgroundTask : MotionTasks
  If, for example, focus is placed on the MotionTask, an event in the BackgroundTask results in a MotionTask being quickly started and handled with few interruptions.
- Read system variables only once and temporarily store them in a local variable for later use

For additional information on high-performance programming, refer to Optimizing access to inputs and outputs (Page 1687) .

Monitoring of timeouts and level overflows

The system variables **Taskruntime** and **effectiveTaskruntime** can be used to determine whether a level overflow or a timeout has occurred.

- If the **Taskruntime** in the IPO/IPO_2 cycle clock is *the same* as the **effectiveTaskruntime** in the IPO/IPO_2 cycle clock, then the task has *not* been interrupted by a higher-priority task.
- If the **Taskruntime** in the IPO/IPO_2 cycle clock is *not the same* as the **effectiveTaskruntime** in the IPO/IPO_2 cycle clock, then the task has been interrupted by a higher-priority task.
  Recommendation: With a ratio of servo : IPO : IPO_2 > 1, enter a percentage value of the IPO cycle clock for the duration which is as large as possible in the task configuration of the **SynchronousTasks**.
- If the **effectiveTaskruntime** of the level which has overflowed is *less* than the set cycle clock of the level, a *timeout* has occurred.
  You can prevent this by adjusting the monitoring time of the levels to the values of the system variable **effectiveTaskruntime**.
- If the **effectiveTaskruntime** of the level which has overflowed is *very close to or greater* than the set cycle clock of the level, a *level overflow* has occurred.
  You can prevent this by adjusting the system cycle clocks, or you can tolerate these overflows.

See also

SynchronousTasks (Page 1326)

Task runtimes (Page 1358)

Information on starting a task: TaskStartInfo (TSI)

Every time a task is started, important information is stored in the **TaskStartInfo**, e.g.:

- The starting time of the task
- For the **TechnologicalFaultTask**: The triggering instance of the technology object and the alarm number
- For the **TimeFaultTask**: The TaskId of the **TimerInterruptTask** which caused the timeout error
- for **the ExecutionFaultTask**: The triggering event and the TaskId of the task in which the event occurred.

Within a task, you can query the relevant **TaskStartInfo** of this task, see Using Taskstartinfo (Page 1262).

Watchdog

The maximum duration for the execution of a task (cycle time) can be configured. If this time is exceeded, the associated SystemInterruptTask (**TimeFaultTask** or **TimeFaultBackgroundTask**) or the response CPU to **STOP** mode can be called.

The cycle time monitoring can be activated for the following tasks:

- BackgroundTask
- TimerInterruptTasks
- SynchronousTasks
- ShutdownTask

Configure cycle watchdog

- Switch to the **Task configuration** tab in the Configuration window for the task.

For BackgroundTask, TimerInterruptTask, ShutdownTask:

- Enter the maximum execution time in the **Time monitoring** field (0 ms = no monitoring).

For IPOsynchronousTask:

- Choose the ratio between the maximum execution period and the IPO cycle clock.

For tasks other than ShutdownTask:

- Select as error response to timeout either the associated **SystemInterruptTask** or the CPU to **STOP** mode response.

Note

For time-triggered tasks: If the task is restarted before it has been executed, there is a timeout error. The SystemInterruptTask is started or the CPU goes to **STOP** mode.

See also

SystemInterruptTasks (Page 1334)

4.2.7.4 Second position control cycle clock (Servo_fast)

Servo_fast (application cycle clock for fast bus system)

Description

Synchronization (data copying, cyclic life-sign, time stamp for output cam and measuring input) with all the synchronous I/O (drive, external encoder, measuring input input, output of output cam, digital and analog inputs/outputs) occurs in the position control cycle clock. Further synchronous cycle clocks (IPO, IPO_2, DccAux, DccAux_2) are based on this cycle clock.

With only one servo, there is only one common application cycle clock available for all the bus systems. This application cycle clock may not be shorter than the cycle clock of the slowest bus system (example: PROFIBUS with 1 ms). Although the faster bus system (e.g. PROFINET IO with 250 µs send clock) is able to transfer the data using a faster cycle clock, the signals are only evaluated in the slow application cycle clock.

Second position control cycle clock (Servo_fast) for the fast bus system

As of V4.2, you can operate synchronous devices (drive, measuring input or output of output cam, I/Os) on the fast bus system (PROFINET) via Servo_fast. Servo_fast always finishes during the first send clock of the fast bus system with this type of arrangement. This makes it possible to operate two bus systems in different application cycle clocks. An assigned position control cycle clock and IPO cycle clock are available for each of the two application cycle clocks.

A description of how to configure a second position control cycle clock can be found in the *Communication* Function Manual under Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface.

A second position control cycle clock is only necessary in applications with special requirements, e.g.:

- For fast I/O processing via PROFINET IO (for particularly short sampling times and response times)
- If, in addition to the electric axes (e.g. servo drives), hydraulic axes with particularly high-performance pressure/position control are implemented
- If the electric axes have to be split into two performance classes (fast servo - slow servo)

Note

The following SIMOTION modules support the second position control cycle clock:

- V4.2 and higher SIMOTION D445-2 DP/PN and SIMOTION D455-2 DP/PN.
- V4.3 and higher SIMOTION D435-2 DP/PN.

If the axes have to be split into two performance classes, a split on the IPO level (IPO, IPO_2) is usually sufficient; a split on the servo level (Servo, Servo_fast) is only required in exceptional circumstances.

- The reference variable calculation/motion profile calculation of the axes is performed on the IPO level.
- The position control and monitoring of the axes is performed on the servo level.

Please also note that an additional Servo_fast/IPO_fast increases the overall load on the SIMOTION runtime system.

In addition, with DSC (Dynamic Servo Control) for servo axes, the dynamically effective part of the position controller in the drive is performed in the frequency of the speed control loop (i.e. usually with 125 µs in the case of SINAMICS S120). If symbolic assignment with automatic message frame determination is used, DSC is activated automatically.

Servo_fast and IPO_fast

This means the additional fast application cycle clocks can also be selected for the technology objects:

Expansion to include Servo_fast and IPO_fast	Expansion to include IPO_fast
TO axis	TO AdditionObjectType
TO external encoder	TO path object
TO synchronous operation	TO fixed gear
TO measuring input	TO formula Object
TO output cam	TO sensor
TO cam track	TO controller object

See also

Cycle clock model based on a single position control cycle clock (Page 1365)

Cycle clock model based on a single position control cycle clock

Description

The use of Servo_fast is optional and is only intended for certain applications. See also Servo_fast (application cycle clock for fast bus system) (Page 1364). An application cycle clock is sufficient for standard applications.

The graphic below shows an arrangement based on a single position control cycle clock.

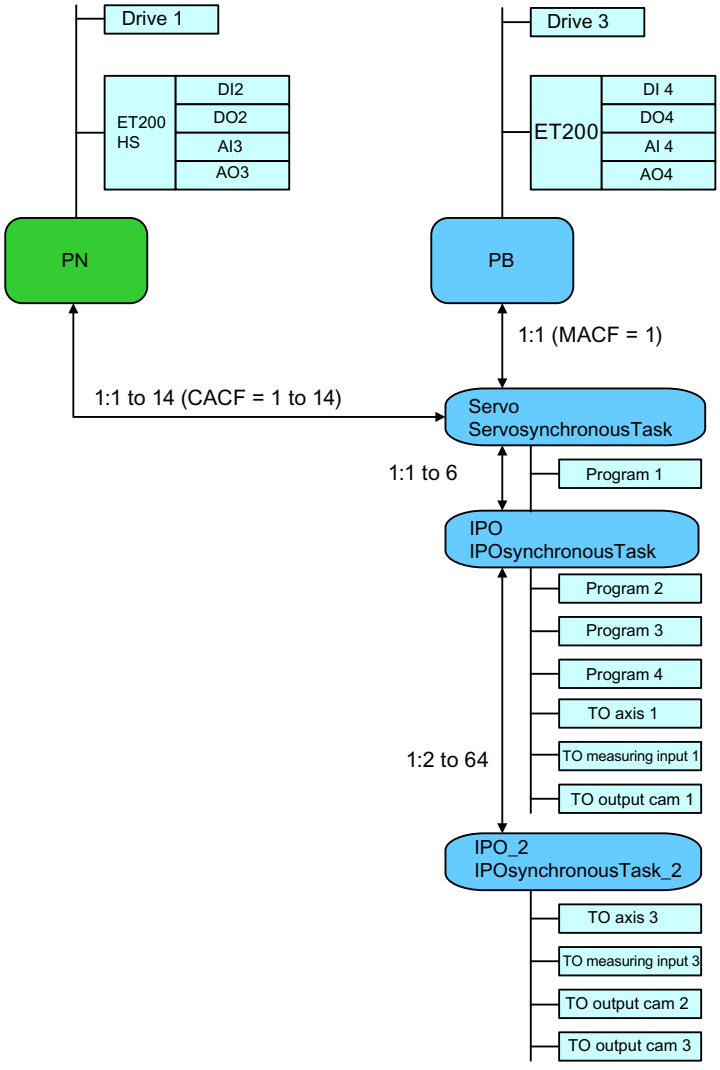


Figure 4-119 I/O with a single position control cycle clock

- MACF: The Master Application Cycle Factor describes the reduction ratio of the PROFIBUS DP send clock to the cycle clock of a higher-level control.
- CACF: The Controller Application Cycle Factor describes the reduction ratio of the PROFINET IO send clock to the cycle clock of a higher-level control.

Cycle clock model based on two position control cycle clocks

Description

The second servo cycle clock enables you to operate two bus systems in different application cycle clocks. An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This enables you to divide your application into slow and fast sections (Servo_fast and IPO_fast).

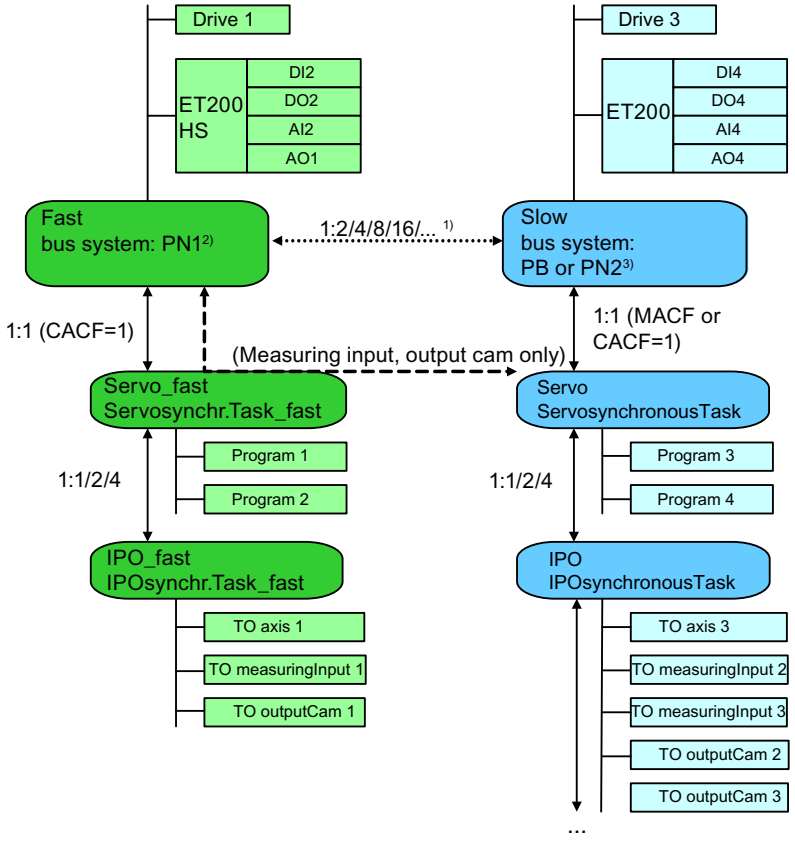
The basic principle is as follows:

- The TOs and I/Os on the fast bus system are processed isochronously in the fast Servo_fast/ IPO_fast.
- The TOs and I/Os on the slow bus system are processed isochronously in the slow Servo/IPO/ IPO_2.

Exception:

- The TO output cam and measuring input in the slow Servo/IPO/IPO_2 can also use the I/O of the fast bus system.

The graphic below shows an arrangement involving two servo cycle clocks.



1) Factor = 2 only for PN1 send cycle clock > 250 µs (for < V4.4 only factor 2, 4, 8 and 16 for all PN1 send cycle clocks)
2) PN1: Onboard PROFINET IO interface for D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN
3) PN2: Optional second PROFINET IO interface with CBE30-2 (as of V4.3)

Figure 4-120 I/O with two servo cycle clocks

Generally, a PROFINET send cycle of max. 4 ms and a PROFIBUS bus cycle of max. 8 ms can be set.

For information on CACF and MACF, see also Cycle clock model based on a single position control cycle clock (Page 1365).

Cycle clock ratios and conditions with two servo cycle clocks

Description

For isochronous use, the application cycle is set in HW Config on the I/O module:

- The MACF (master application cycle) is set for PROFIBUS DP slaves. Only MACF=1 is supported if you are also using PROFINET IO. All PROFIBUS devices run in the slow servo cycle clock.
- The CACF is set for PROFINET IO devices. As of V4.2, you can also select the servo cycle clock (Servo_fast) on I/O devices. As of V4.3, a second PROFINET IO interface can be connected in D4x5-2 DP/PN Control Units. This allows the integrated PROFINET interface (X150) to be operated in the Servo_fast clock cycle and the second PROFINET IO interface (CBE30-2) in the Servo clock cycle.

The following settings are possible:

Product version	Property	Application cycle of the devices on		
		PROFINET IO Integrated (X150)	PROFINET IO (CBE30-2)	PROFIBUS DP
Before V4.2	1 servo cycle clock	--	--	Servo
As of V4.2 in addition	2 servo cycle clocks	Servo_fast	--	Servo
As of V4.3 in addition	2 servo cycle clocks	Servo_fast	Servo	Servo

Application cycle on the I/O device

The application cycle set in HW Config on the I/O device has the following effects on the isochronous use of technology objects and user programs:

- Standard I/O are only used in a precisely isochronous manner in the relevant servo. An "IPO-synchronous" process image of the outputs can be output in the 1st, 2nd, 3rd, etc. servo cycle clock within an IPO (with scaling down of the cycle clocks, servo to IPO) (depending on the servo + IPO runtimes). This means the interval between 2 outputs may fluctuate.
- Drive and encoder telegrams are synchronized in the relevant servo between the controller and device. They can be used isochronously by the TO axis in the associated IPO.
- Measuring inputs and output cam outputs with time stamps can also be used synchronously in a slower cycle clock.

Application cycle clock set	Cycle clocks for the isochronous use of		
	Standard I/O	Drive, external encoder	Measuring input, output cam output
Servo_fast	Servo_fast	Servo_fast IPO_fast	Servo_fast IPO_fast Servo IPO IPO_2
Servo	Servo	Servo IPO IPO_2	Servo IPO IPO_2

Factors and reference cycle clocks with two servo cycle clocks

The table below contains the factors and reference cycle clocks that are possible with 2 servos:

Task name	Factors which can be set	Reference cycle clock
Servo_fast	1	Send cycle clock of the onboard PROFINET IO interface (X150)
IPO_fast	1, 2, 4	Servo_fast
Servo	1	PROFIBUS DP bus cycle clock and send cycle clock of the optional second PROFINET IO interface (CBE30-2)
IPO	1, 2, 4	Servo
IPO_2	2, 3, 4, 5, ..., 64	IPO
DccAux	2, 3, 4, 5, ..., 32	IPO_2
DccAux_2	2, 3, 4, 5, ..., 32	DccAux

Possible cycle clock settings for equidistant cycle clocks

The following conditions must also be satisfied when using two servo cycle clocks:

- PROFIBUS DP bus cycle clock = N x send cycle clock of the onboard PROFINET IO interface X150
  N = 2, 4, 8, 16, ... (N = 2 only for send cycle clock > 250 µs)
  (for < V4.4: N = 2, 4, 8, 16 for all send cycle clocks)
- Send cycle clock of an optional second PROFINET IO interface (CBE30-2) = PROFIBUS DP bus cycle clock
- IPO ≥ IPO_fast
- Servo_fast and IPO_fast are always assigned to the onboard PROFINET IO interface X150
- The onboard PROFINET interface can only be operated as sync master (consequently with distributed synchronous operation via PROFINET, the D4x5-2 cannot participate as sync slave when Servo_fast/IPO_fast is used)
  If the onboard PROFINET interface is configured as sync slave, the faulty configuration is indicated via a diagnostic buffer entry and the controller goes into starting inhibit.

Priorities and sequences of synchronous tasks

Cycle clock division when there are two position control cycle clocks

Description

The following rules apply in terms of task priorities:

- Task levels with a shorter cycle time have a higher priority than task levels with a longer cycle time.
- If the cycle time is the same, the servo level has a higher priority than the IPO level.
- If the cycle time is the same, the fast IPO level (IPO_fast) has a higher priority than the slow IPO level (IPO).

There are two synchronous task sequences dependent upon the cycle clock settings. The following graphic shows the two options for the priority and sequence of the task levels.

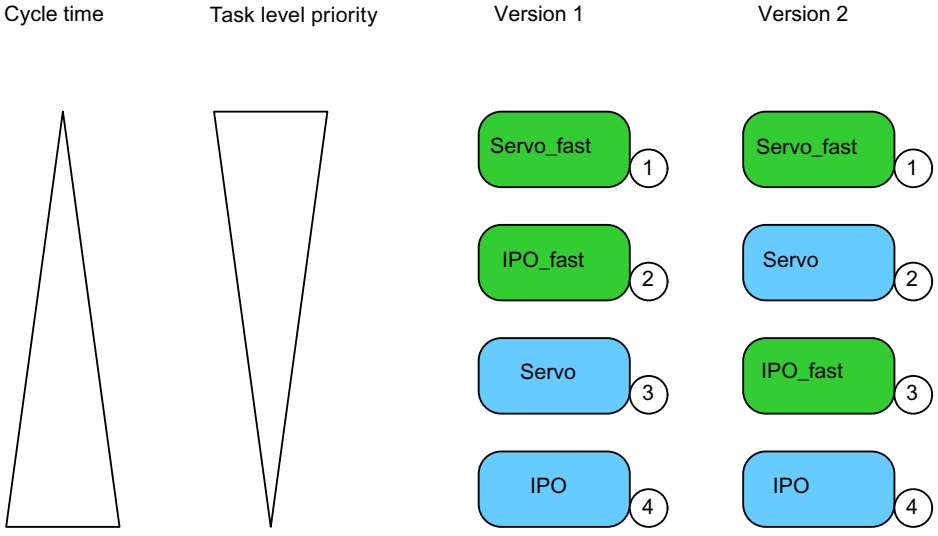


Figure 4-121 Dividing cycle clocks when there are two position control cycle clocks

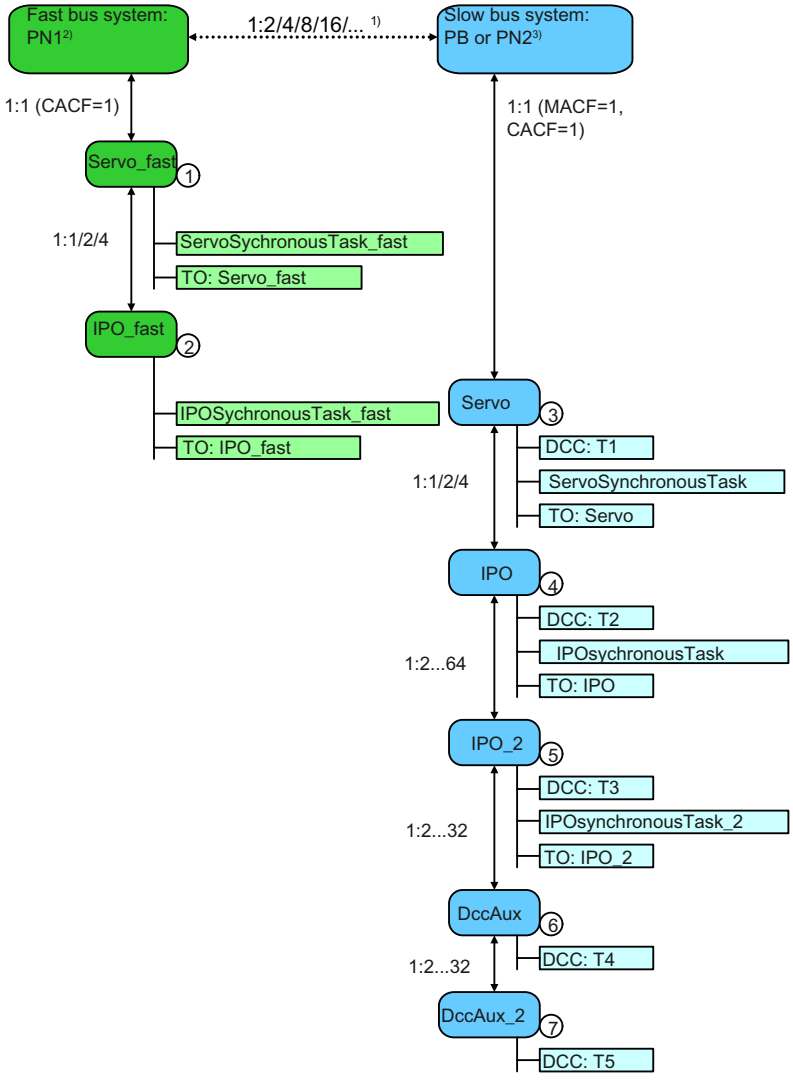
Rules for cycle clock division

- If the setting is made for scaling down of cycle clocks between PROFINET IO and PROFIBUS DP as well as between Servo_fast and IPO_fast, the user has a choice between version 1 (Servo_fast – IPO_fast – Servo – IPO) and version 2 (Servo_fast – Servo – IPO_fast – IPO).
- If IPO_fast has a shorter runtime than the servo, the priorities for version 1 are set within the execution level system.
- A very fast **terminal-terminal response** requires a very short cycle clock for Servo_fast. If there is a strong possibility that IPO_fast may not be finished in good time before Servo due to a high system load, then IPO_fast must be set to the same speed or a slower speed than Servo. The priorities for version 2 are then set within the execution level system.
- The IPO_fast cycle clock is not longer than the IPO cycle clock.
- The new Servo_fast and IPO_fast cycle clocks must not overrun.
- In the event of a level overflow, the system goes into STOP mode. The starting lockout is set and an appropriate entry is made in the diagnostics buffer. Only after a ramp-up (power on/off) or download can the system return to RUN mode.

Cycle clock division for the 2nd servo, version 1

Task assignment, version 1 (Servo_fast – IPO_fast – Servo – IPO)

The graphic below shows version 1 of cycle clock assignment.



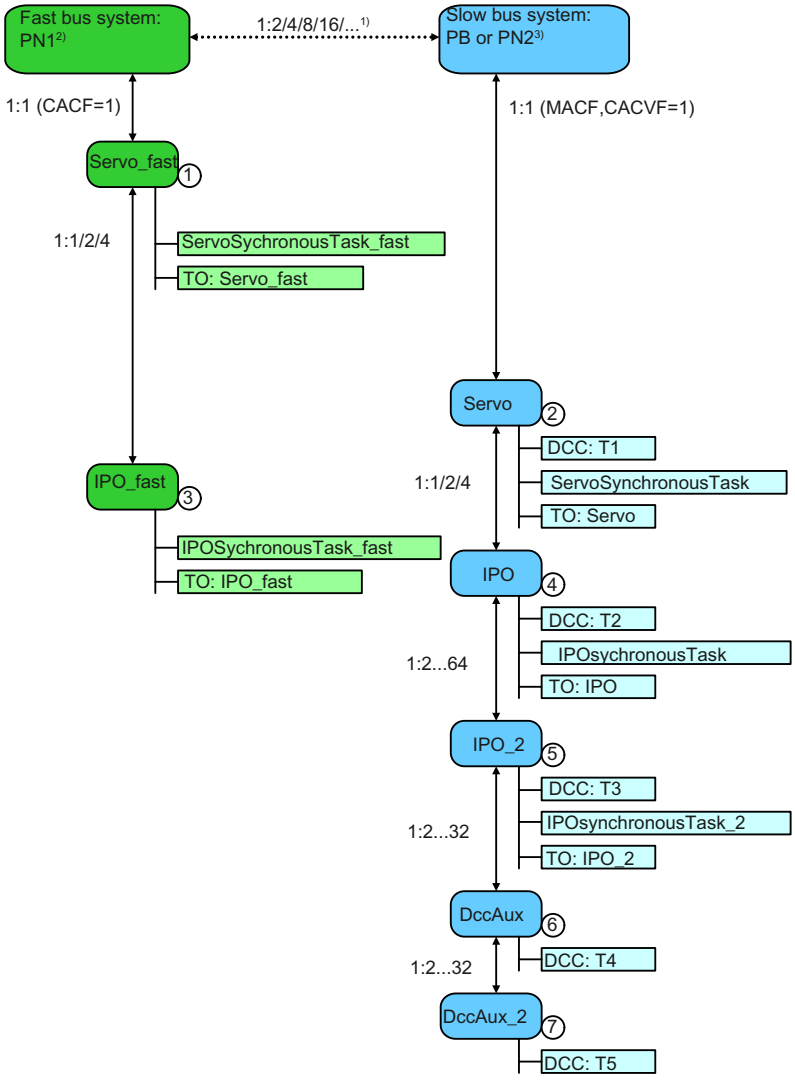
[1)] Factor = 2 only for PN1 send cycle clock > 250 µs (for < V4.4 only factor 2, 4, 8 and 16 for all PN1 send cycle clocks)
[2)] PN1: Onboard PROFINET IO interface for D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN
[3)] PN2: Optional second PROFINET IO interface with CBE30-2 (as of V4.3)

Figure 4-122 Cycle clock division, version 1

Cycle clock division for the 2nd servo, version 2

Task assignment, version 2 (Servo_fast– Servo– IPO_fast – IPO)

The graphic below shows version 2 of cycle clock assignment.



[1] Factor = 2 only for PN1 send cycle clock > 250 µs (for < V4.4 only factor 2, 4, 8 and 16 for all PN1 send cycle clocks)
[2] PN1: Onboard PROFINET IO interface for D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN
[3] PN2: Optional second PROFINET IO interface with CBE30-2 (as of V4.3)

Figure 4-123 Cycle clock assignment, version 2

Synchronization of cycle clock and bus systems

Description

Synchronization between the slower and faster bus systems occurs automatically, whereby the cycle clock of Servo_fast is always an integer multiple of the servo.

Example of cycle clock and bus synchronization

This example shows how the individual cycle clocks behave during a cycle.

- Fast servo cycle: 250 µs
- Fast IPO cycle: 500 µs
- Slow servo cycle: 1 ms
- Slow IPO cycle: 2 ms

Cycle clock	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Time	0000	0250	0500	0750	1000	1250	1500	1750	2000	2250
Start faster, servo_fast	x	x	x	x	x	x	x	x	x	x
Start faster, IPO_fast	x		x		x		x		x	
Start slower, servo	x				x				x	
Start slower, IPO	x								x	

4.2.7.5 Time allocation in the round robin execution level

The time *remaining* after high-priority user and system tasks have been processed is used by the **MotionTasks** and the **BackgroundTask**.

This time is allocated to the BackgroundTask and the MotionTasks according to the *round robin principle*.

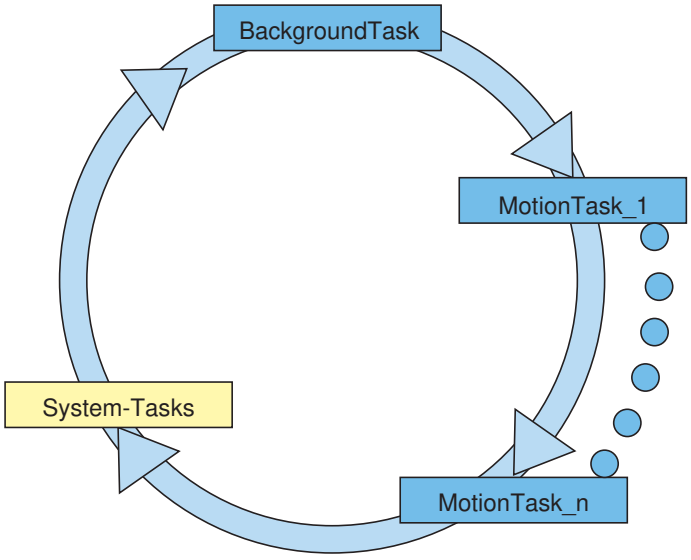


Figure 4-124 Overview of the time allocation in the round robin execution level

Round robin principle

Computing time that is not consumed by higher priority tasks is distributed over the remaining tasks in the round robin principle (Background Task, MotionTasks and Systemtasks); this means that these tasks run one after the other. Thus a quasi parallel (quasi "concurrent") processing of these tasks take place.

The next task is always started when the previous task gives up the computing time (task completed or in the "wait" state). The computing time of a round robin task is also limited to the maximum number of servo cycle clocks. In the next round robin cycle the computing time is continued at the point of interruption. The system ensures that a round robin task always gets at least the computing time of a servo cycle clock. BackgroundTasks can be favored so that the computing time of multiple successive servo cycle clocks can be made available to these tasks (slider execution system).

Additional tasks

In addition to the user program tasks (BackgroundTask, MotionTask), other system tasks (e.g. for communication) that require computing time in the round robin cycle also run in the round robin execution level.

Sequence

In general, the tasks run successively in the round robin cycle. (The sequence is *non-deterministic* and can, for example, change each time via a download, PowerON.)

Time allocation

A task can successively run a specified number of servo cycle clocks (remaining runtime) before it must transfer the computation time to the next task. However, it can also be transferred beforehand to the next task if it "does not have anything to do".

For the constructed case that all tasks in the round robin execution level have already been handled once, the first task is restarted. In this way, there is *no* IDLE time.

There are times in which no user program is being executed. This, for example, is the case when the BackgroundTask and the MotionTask are very brief (shorter than the remaining runtime in a servo cycle clock); the system tasks will then use the remaining runtime.

Restart / process image

After finishing, the BackgroundTask is not started again until the process image has been updated. The update is performed in the **ServoTask**.

Performance

A continuous loop in a MotionTask, without wait commands or synchronous motion commands, causes the MotionTask to use its maximum computation time. Consequently, the cycle of a BackgroundTask is additionally utilized (a larger part of the computation time goes into the MotionTask). Consequently, the system tasks in the round robin level are called at longer time intervals, which, for example, may influence the communication.

Note

MotionTasks with (continuous) loops without _waitTime (0 s) burden the round robin execution level as the MotionTask uses two complete servo cycle clocks.

During the allocation of the round robin execution level for the BackgroundTask and MotionTasks, the BackgroundTask in a complete round robin cycle receives at least one time slice, a MotionTask receives two time slices (if, for example, they are required because of a continuous loop).

Cyclic MotionTasks

Note when Motion Tasks should run cyclically:

If you want to use continuous loops in MotionTasks, then issue, for example, a **_waitTime(0s)** in each cycle. This MotionTask then passes to the following MotionTask.

Note

MotionTasks with (continuous) loops without _waitTime (0 s) burden the round robin execution level as the MotionTask needs two complete servo cycle clocks.

If you want a MotionTask to be executed cyclically, it is better to end the task in the normal way and restart the MotionTask again from a different task (e.g. BackgroundTask) rather than use a continuous loop or a step command at the start of the program. This also has advantages for a download during **RUN**.

See also

Optimizing the execution system (Page 1688)

Setting of the time allocation

You can use a slider accessible via the **Time allocation** button for **MotionTasks** or for the **BackgroundTask** to set the chronological weighting of the BackgroundTask to the remaining tasks of the round robin execution level.

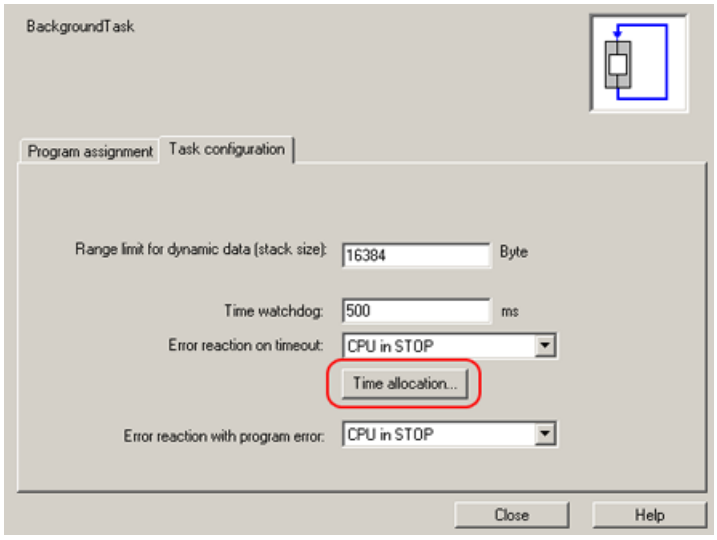


Figure 4-125 Task configuration in SCOUT

Specifying the time allocation in the round robin execution level

1. Select the **Task configuration** tab in the configuration window of a **MotionTask** or the **BackgroundTask**.
2. Click **Time allocation**.
   The Time Allocation in the Round Robin Execution Level window opens.
3. Use the slider to set the ratios of the computation times.
4. Click **OK** to confirm.

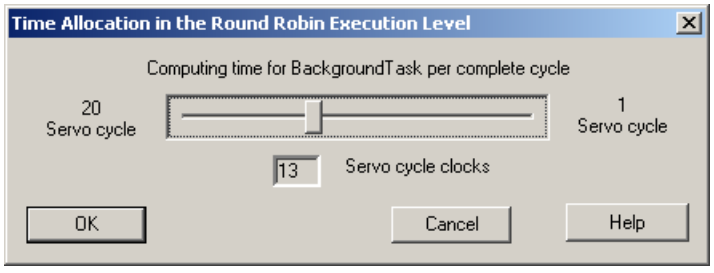


Figure 4-126 Time allocation in the round robin execution level

Time allocation in the round robin execution level

This is where you specify the ratio of the time allocation between the BackgroundTask and all other round robin tasks (MotionTasks and Systemtasks, see Execution levels/tasks (Page 1301)).

The BackgroundTask gets a (maximum) of n-times the computing time of a different round robin task (all other round robin tasks, e.g. MotionTask_1, each have the same maximum share of computing time). You can set this factor with the slider.

When the BackgroundTask has run to an end, it will be restarted when the inputs are updated, at the earliest (after the next servo cycle).

If the computing time of the BackgroundTask is set high, and thus disadvantages the other round robin tasks then, for example, this will slow the communication to SCOUT/OP. Such a setting should only be made for extremely time-critical applications.

Table 4-73 Effect of computation time allocation to MotionTasks and BackgroundTask

Increased computation time for	**Effect**
MotionTasks	BackgroundTask takes longer to run from beginning to end; time monitoring may respond in extreme cases.
BackgroundTask	Programs in the MotionTask may take longer to execute under some circumstances.
The computation time for the BackgroundTask also includes the time required for acyclic communication (via PROFIBUS or PROFINET IO with IRT).	

Settings (examples)

To explain the time allocation in the round robin execution level, two examples are provided here for setting the slider.

Case 1: Slider at the far right-hand side

The BackgroundTask runs for at least one servo cycle clock.

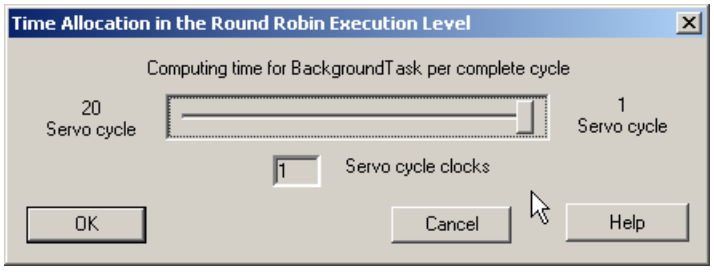


Figure 4-127 Time allocation in the round robin: Setting of the computation time for MotionTasks

In this setting, the BackgroundTask runs for one servo cycle clock, then all MotionTasks run (each MotionTask runs for maximum two servo cycle clocks, unless it enters the suspend mode beforehand), then the BackgroundTask runs again for one servo cycle clock, etc.

Table 4-74 Example:

Servo cycle clock 1	Servo cycle clock 2-3	Servo cycle clock 4-5	Servo cycle clock 6	Servo cycle clock 7-8
Background Task	MotionTask 1	MotionTask 2	Background Task	MotionTask 1

Sample program:

MotionTask 1 and 2 run concurrently with the BackgroundTask;
servo cycle clock = 3 ms

- A counter in a While loop is incremented in the BackgroundTask (green line).
- A counter in a While loop is incremented in MotionTask 1 (red line).
- A counter in a While loop is incremented in MotionTask 2 (blue line).

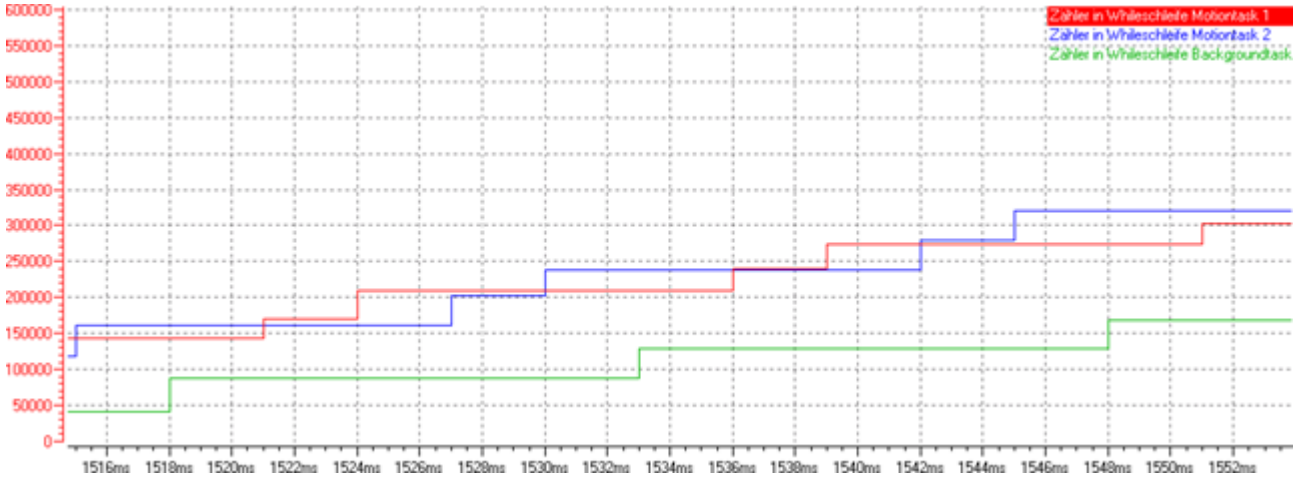


Figure 4-128 Execution example for the time allocation in the round robin: Setting of the computation time for MotionTasks

t = 1518 ms	BackgroundTask runs for one servo cycle clock
t = 1521 ms, 1524 ms	MotionTask 1 runs for two servo cycle clocks
t = 1527 ms, 1530 ms	MotionTask 2 runs for two servo cycle clocks
t = 1533 ms	BackgroundTask runs for one servo cycle clock

Case 2: Slider at the far left-hand side

The BackgroundTask runs for 20 servo cycle clocks.

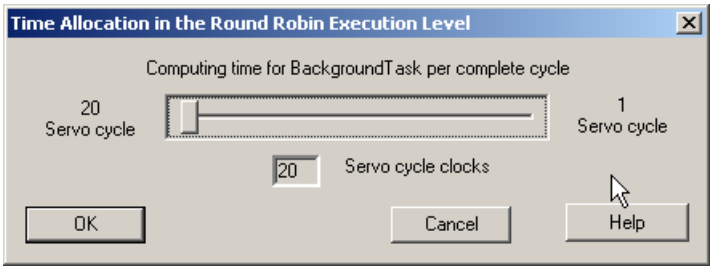


Figure 4-129 Time allocation in the round robin: Setting the computation time for BackgroundTask

In this setting, the BackgroundTask runs for 20 servo cycle clocks, then all MotionTasks run (each MotionTask runs for maximum two servo cycle clocks, unless it enters the suspend mode beforehand), then the BackgroundTask runs again for another 20 servo cycle clocks, etc.

Table 4-75 Example:

Servo cycle clock 1-20	Servo cycle clock 21-22	Servo cycle clock 23-24	Servo cycle clock 25-40	Servo cycle clock 41-42
Background Task	MotionTask 1	MotionTask 2	Background Task	MotionTask 1

Sample program:

MotionTask 1 and 2 run concurrently with the BackgroundTask;
servo cycle clock = 3 ms

- A counter in a While loop is incremented in the BackgroundTask (green line).
- A counter in a While loop is incremented in MotionTask 1 (red line).
- A counter in a While loop is incremented in MotionTask 2 (blue line).

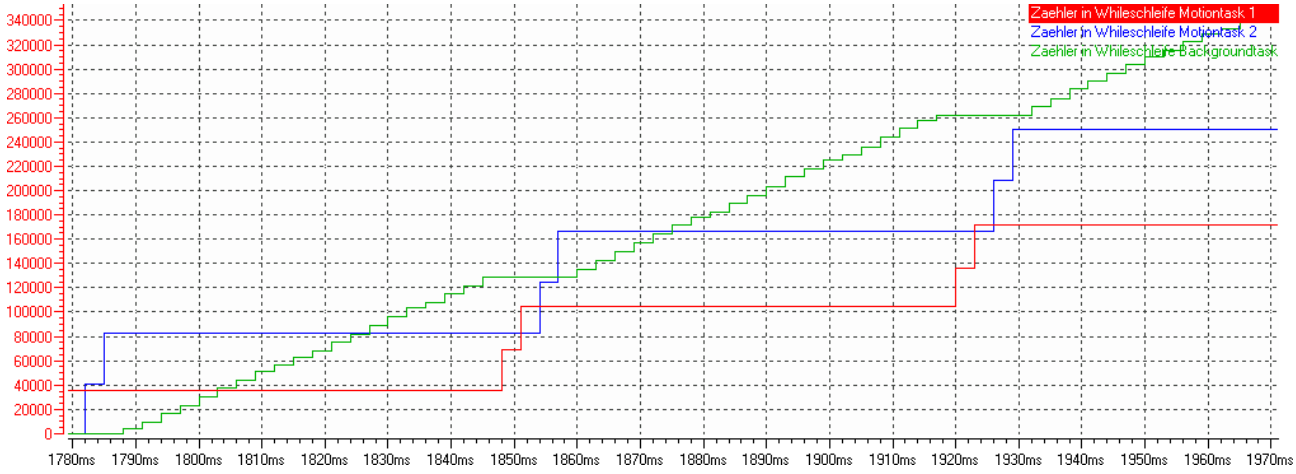


Figure 4-130 Execution example for the time allocation in the round robin: Setting the computation time for BackgroundTask

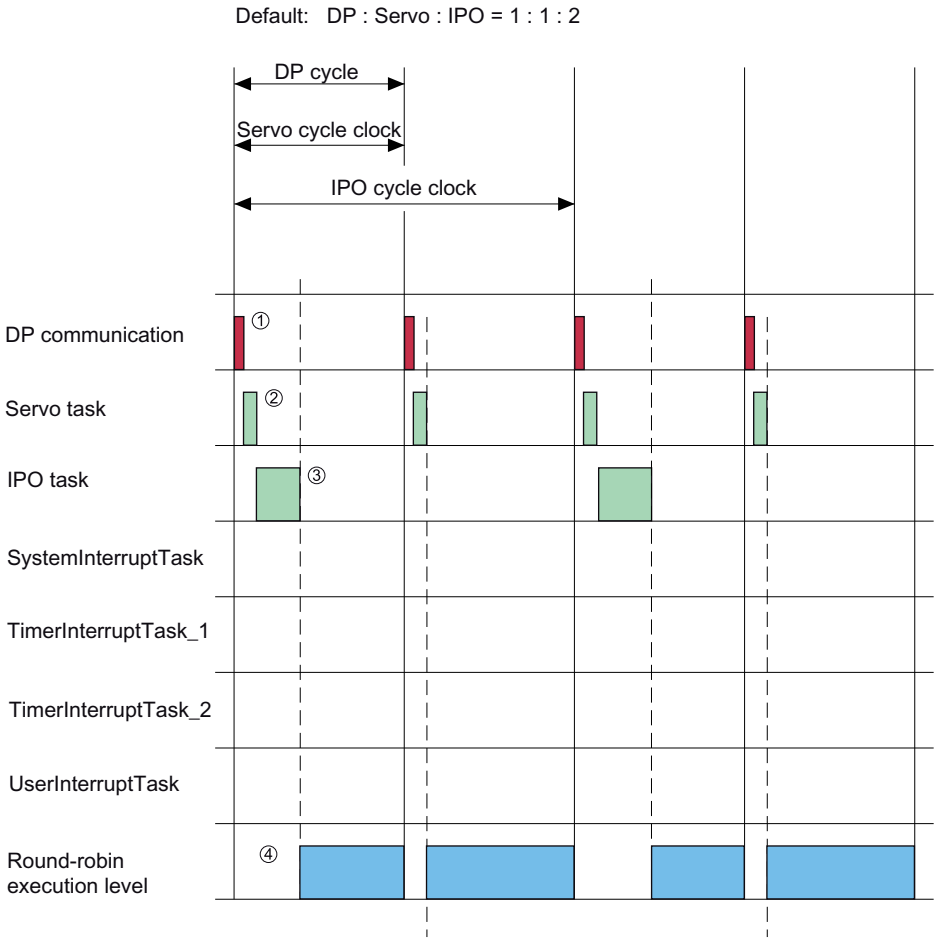
t = 1788-1845 ms	The BackgroundTask runs for 20 servo cycle clocks
t = 1848 ms, 1851 ms	MotionTask 1 runs for two servo cycle clocks
t = 1854 ms, 1857 ms	MotionTask 2 runs for two servo cycle clocks
t = 1860-1917 ms	The BackgroundTask runs for 20 servo cycle clocks

Processing of the tasks (examples)

The following examples illustrate the chronological execution of the various tasks.

The examples refer to PROFIBUS and apply to PROFINET IO with IRT analogously.

Example 1: Chronological execution when no InterruptTask is active



1 In the execution system, the cycle clocks have been selected as follows for this example:
DP cycle: Servo cycle clock: IPO cycle clock: 1:1:2
This means that the servo task is executed in each DP cycle and the IPO task is only executed in every second DP cycle.

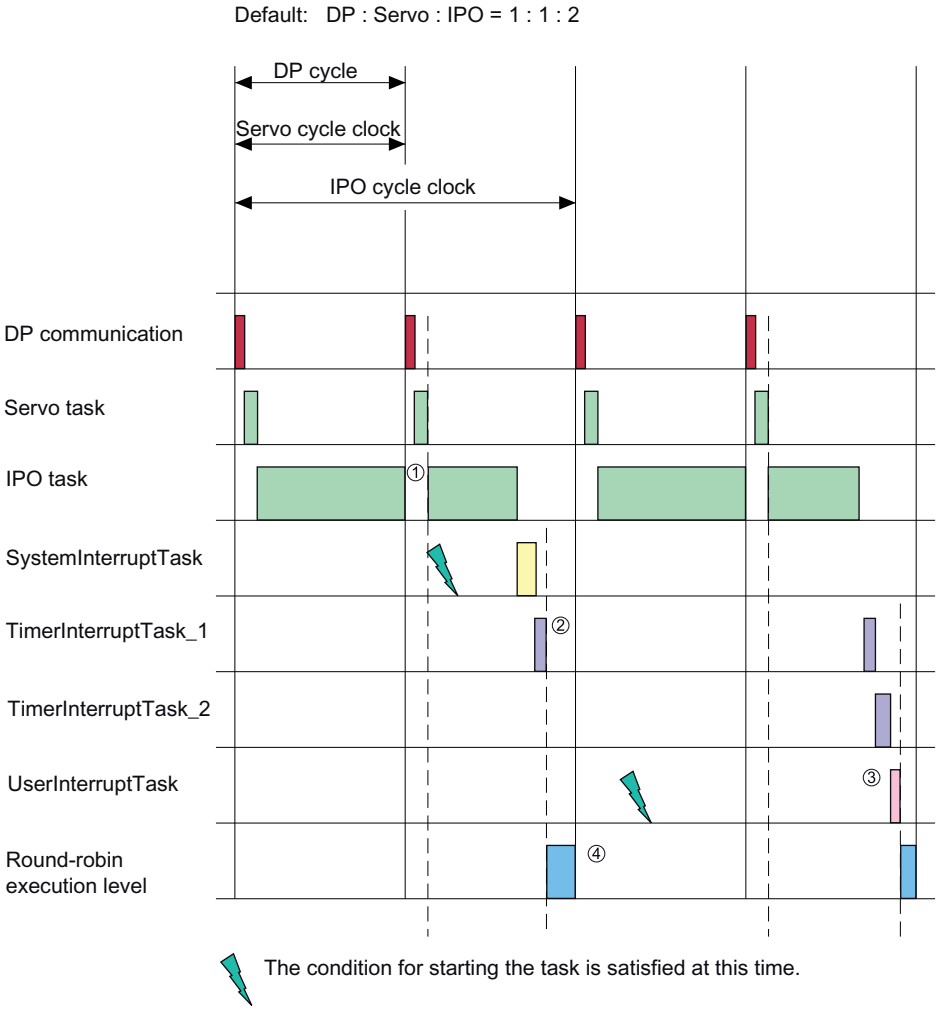
2 DP communication has the highest priority followed by the servo task.

3 IPO tasks are executed after servo tasks.

4 The round robin execution level is executed in the remaining time.

Figure 4-131 Chronological task execution, example: no InterruptTask active

Example 2: Chronological execution when an InterruptTask is active and IPOTask lasts longer than one servo cycle clock



1 The program executed in the IPO task lasts longer than the servo cycle clock. As a result, the IPO task is interrupted and the DP communication and the servo task are executed. Then, execution of the IPO task resumes.

2 After the IPO task is completed, the SystemInterruptTask is executed. Then the lower-priority TimerInterruptTask is started.

3 The UserInterruptTask is also not executed until the higher-priority tasks are completed, even if the condition for the UserInterruptTask was previously satisfied.

4 The round robin execution level is executed in the remaining time.

Figure 4-132 Chronological task execution, example: with InterruptTask active and IPOTask lasts longer than one servo cycle clock

Example 3: Special features for task change in multitasking

However the behavior of the multitasking with task changes, e.g., in case of wait commands or synchronous commands, must be taken into account when querying conditions in MotionTasks and also in the BackgroundTask.

Example:

Table 4-76 Example:

	BackgroundTask	**MotionTask1**	**MotionTask2**
Sources:	x=5	...	...
	... x=4 ...	if x <> 5 _enableAxis Endif	if x = 5 _stopAxis Endif
		...	...
	BackgroundTask	**MotionTask1**	**MotionTask2**
Execution sequence:	x=5		
	Task change →	if x <> 5 Condition not fulfilled endif	
		Task change →	if x = 5 Condition fulfilled Task change →
	x=4		
	Task change →	...	
		Task change →	...
			_stopAxis

Although the execution seems clear in MotionTask2, it may occur that the execution is other than intended due to a task change.

After the renewed change in the BackgroundTask (x=4), the if-queries have already been processed and the query results from before the task change are still valid. This means that although the condition x=5 is not fulfilled anymore at this time:

- _enableAxis is not executed in MotionTask1
- _stopAxis is executed in MotionTask2

The execution depends on when the task change is performed.

Note

To ensure multitasking runs smoothly, it is better not to use global variables from different tasks.

4.2.7.6 Task Trace overview

Using Task Trace

Description

The Task Trace is a tool for troubleshooting in the SIMOTION multitasking environment. The Task Trace offers the following options:

- Graphic display of the sequence of individual tasks and user events (generated using a program command).
- Trace of individual user tasks.
- Ability to configure the Task Trace using IT Diag or via the user program (system functions).
- Storage of the trace file on a memory card.
- Start of the SIMOTION Task Profiler as a separate application using IT Diag or the SCOUT device diagnostics.

For detailed information regarding the Task Trace, see the *Task Trace* function manual.

4.2.7.7 Isochronous I/O processing on fieldbus systems

The general sequence of isochronous I/O processing in a control system with I/O connection via a fieldbus system is presented below:

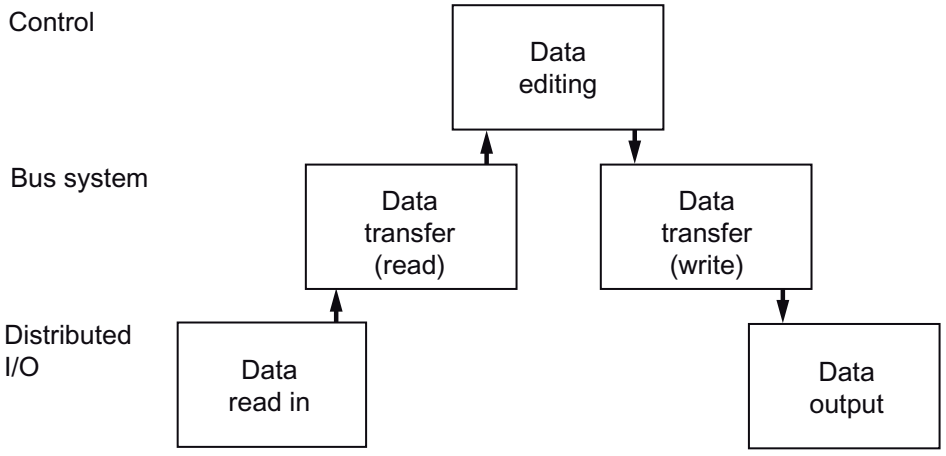


Figure 4-133 Data flow of isochronous I/O processing on fieldbus systems

In an isochronous system, the individual actions are chronologically synchronized with each other. Thus, short and reproducible response times (i.e. with the same length) are achieved.

Cycle clock and time reference are preset by an isochronous, i.e. clocked bus system.

Data protocol on PROFIBUS DP

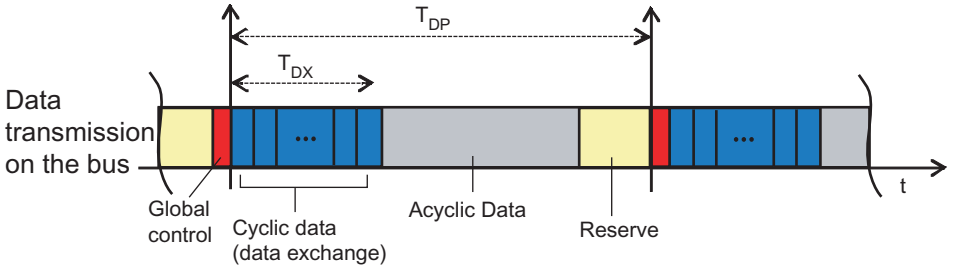


Figure 4-134 Data protocol on PROFIBUS DP

The data protocol on the bus contains the following:

- An isochronous **global control message frame (GC)** that defines the bus cycle clock
- **Cyclic data**: Data transmitted between the nodes in each cycle clock
  Cyclic data communication allows for especially short and reproducible process response times. The transfer of information is done *in each cycle*.
  A user program accesses this data via the process image or via direct access to the I/O.
- **Acyclic data**: Data transmitted only if required and in larger quantities
  The acyclic transmission is suitable especially for the non-time-critical transmission of larger data quantities; the data can be distributed to several tasks.
  Example of acyclic data: Alarms, diagnostic services and data sets
  A user program accesses this acyclic data, e.g. using the commands _readrecord, _writerecord.
- **Reserve**: Remaining time until the next global control.
  The residual time differs depending on the current running acyclic communication, and is a maximum of T_{DP}-T_{DX}.

Data protocol on PROFINET IO

Description

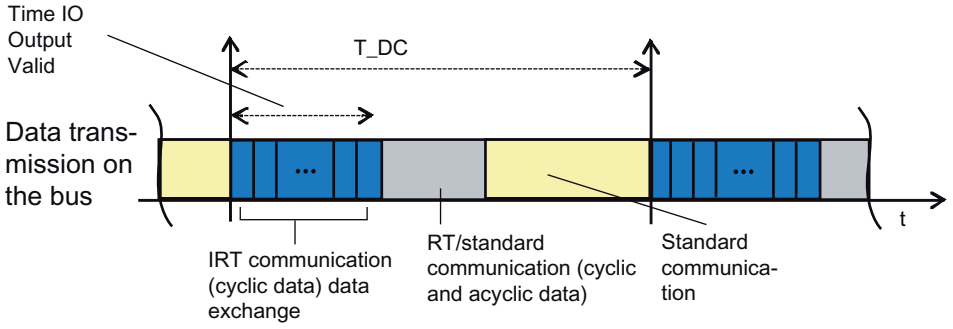


Figure 4-135 Data protocol on PROFINET IO

The data protocol on the bus contains the following:

- The IRT communication that transfers cyclic data in a planned time interval.
- The RT and standard communication in which the following data can be transferred:
  - Cyclic RTC message frames with which I/O data can be transferred, for example
  - Acyclic RTA telegrams with which alarm data can be transferred, for example
  - NRT data (standard communication), standard Ethernet telegrams with which all standard Ethernet-based protocols can be transferred
- Standard communication (last interval prior to synchronization telegram); here only communication can take place that is concluded by the time the synchronization telegram is used.

TIME IO Output Valid describes the time during which the new output data are available.

Isochronous data processing

Isochronous application

With an isochronous application, the processing cycle clock of the controller and any drives connected or distributed I/O are synchronized to the bus cycle.

Isochronous data transmission on PROFIBUS DP

For PROFIBUS the bus cycle clock is synchronized to the Global Control Telegram (GC). The GC embodies the bus cycle clock and is also the system cycle clock for the entire system.

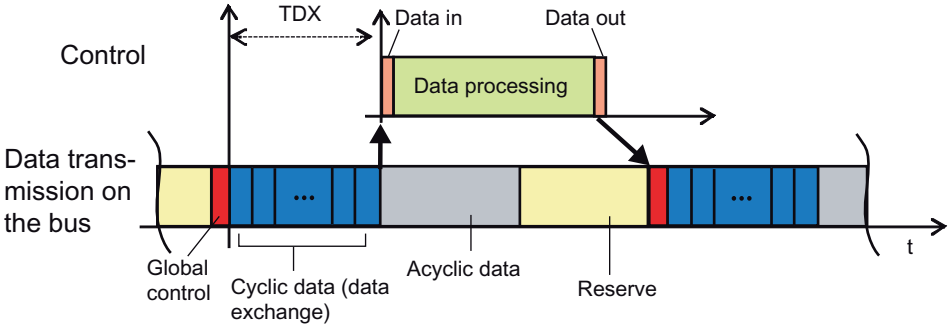


Figure 4-136 Structure of a bus cycle for PROFIBUS DP

Isochronous data transmission on PROFINET IO

For PROFINET IO the processing cycle clock of the controller is synchronized to the IRT communication.

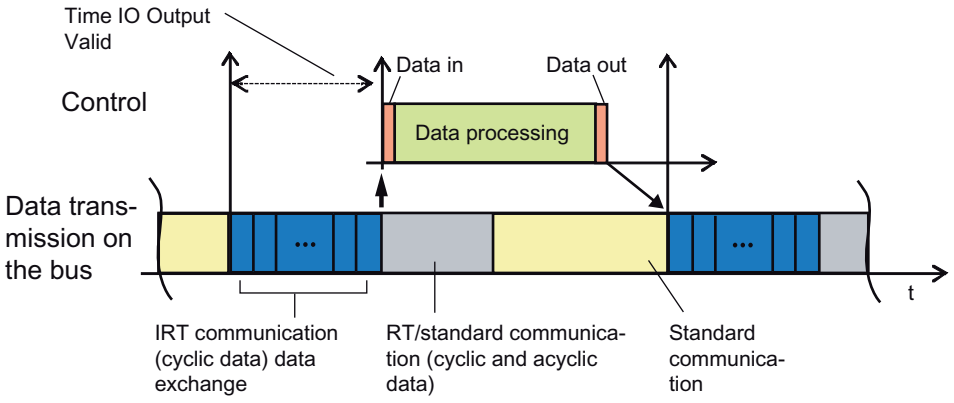


Figure 4-137 Structure of a bus cycle for PROFINET IO

The data transmitted via cyclic data communication can thus be processed isochronously.

Isochronous data acquisition with PROFIBUS DP

In an isochronous complete system, data acquisition, data processing and data output refer to the total system cycle clock.

The application is started when the cyclic data transfer has been completed (T_{DX}).

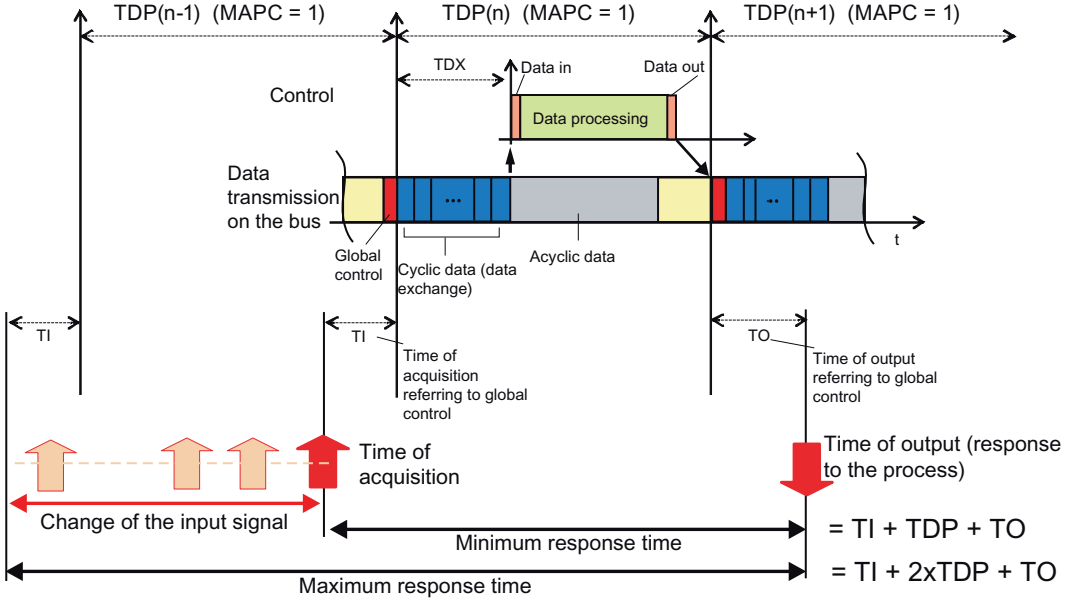


Figure 4-138 Calculation of the reaction times (PROFIBUS DP)

Isochronous data transmission with PROFINET IO

The application is started when the cyclic data transfer has been completed (Time IO Output Valid).

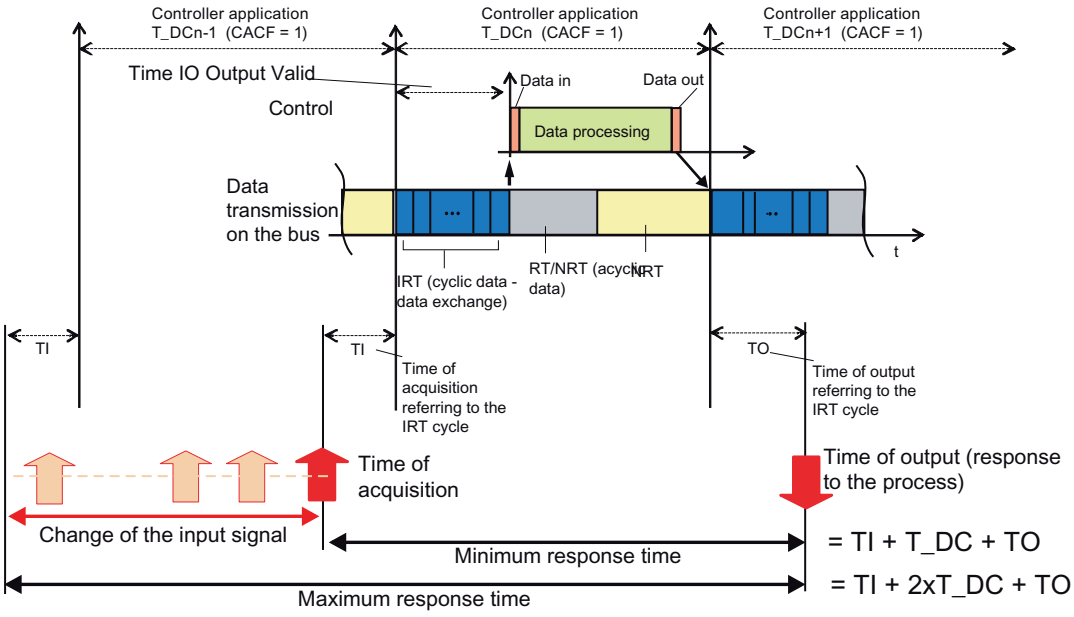


Figure 4-139 Calculation of the reaction times (PROFINET IO)

Calculation of the reaction times (PROFIBUS DP and PROFINET IO)

The read-in process in the distributed I/O must be brought forward, offset by time T_I, so that a consistent state of the inputs can be read in at the starting time of the new bus cycle.

The time T_I comprises at least the signal processing, and conversion time on the electronic modules, and in the case of a modular ET 200 I/O system also the transmission time of the inputs on the ET 200 backplane bus. In general T_I is selected in such a manner that it is the same for all modules, the slowest module determines the time in this regard.

The time T_O ensures that the process reactions of the user program (data processing) are switched through simultaneously and consistently to the "terminals" of the distributed I/O on the bus. analogously for drives the set point is made available for the drive control. The time T_O comprises at least the time for the cyclic data exchange of all nodes on the bus; in the case of a modular ET 200 I/O system the transmission time of the outputs on the ET 200 backplane bus and the signal processing, and conversion time on the electronic modules.

Reaction times or dead times (for drives)

By synchronizing the individual cycles it is possible to read the input data in the cycle clock "n-1", to transmit and process the data in the cycle clock "n", and to transmit the calculated output data at the beginning of the cycle clock "n+1", and to connect the output data to the "terminals" in the same cycle clock.

PROFIBUS

This results in a real process reaction time of:

- $T_I + T_{DP} + T_O$ minimum
- $T_I + 2 \cdot T_{DP} + T_O$ maximum

PROFINET

This results in a real process reaction time of:

- minimum $T_I + T\_DC + T_O$
- maximum $T_I + 2 \cdot T\_DC + T_O$

Dynamic response with respect to data processing in the control

Data processing in the ServoSynchronousTask

- For data processing in the ServoSynchronousTask, for the bus cycle clock: Servo = 1 : 1 the setting can produce a processing time of one bus cycle clock in the controller.
  This applies, for example, for the setting of output data, e.g. DO or AO, depending on the input signals.

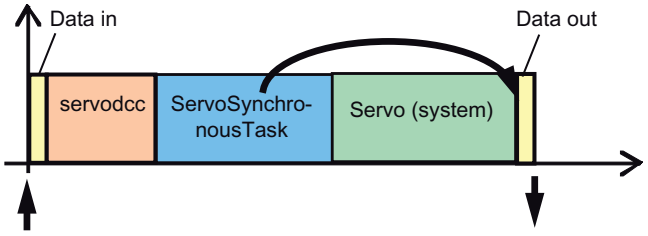


Figure 4-140 Data processing in the control - setting the output data

- When influencing axis data/axis motions, the processing time depends on the fact if the data is already active in the servo (e.g. superimposed setpoint or output value).

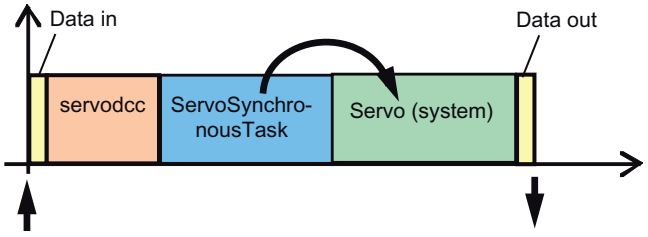


Figure 4-141 Data processing in the control - influencing motion data

- If the data or commands become effective not until the IPO (e.g. issuing of motion commands), the output data on the outputs only become effective with the next servo.
  Note: To keep the runtime of the ServoSynchronousTask low (and thus the time until Data Out), it is preferable to implement the programming of such tasks in the IPOSynchronousTask.

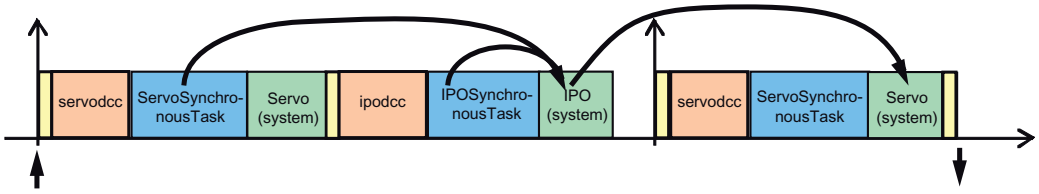


Figure 4-142 Data processing in the control - output data effective in the IPO

Data processing in the IPOSynchronousTask

- In the data processing in the IPOSynchronousTask, the output data on the I/O become effective with the next servo.

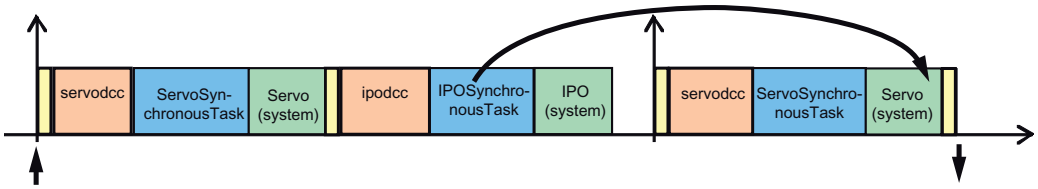


Figure 4-143 Data processing in the control - output data effective in the next servo

- Or: Motion data in the next IPO or servo (see above)

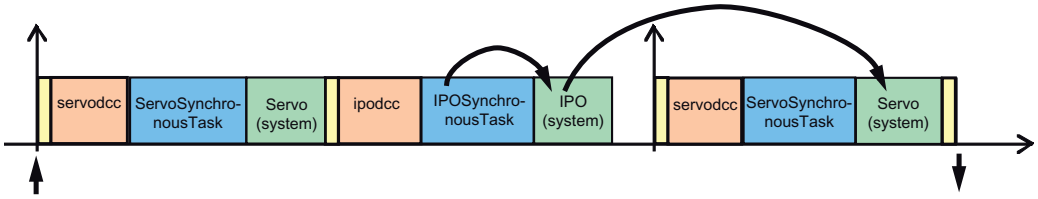


Figure 4-144 Data processing in the control - output data effective in the IPO - influencing motion data

In those cases where the output data of the I/O are only transmitted in the next servo, the reaction time is increased by one bus cycle clock each (for bus cycle clock: servo : IPO = 1 : 1 : 1).

Scenario	Recommendation for user program task
Fast terminal-terminal response on I/O	Use ServosynchronousTask
Fast influencing of the setpoint (servo level)	Use ServosynchronousTask
Switching/superimpositioning the motion (e.g. _stop, _move, _pos, ...) e.g. print mark	Use IPOsynchronousTask

As of V4.1 in exceptional cases the IPO share can also be calculated in the servo, in this regard see **Motion control / interpolator** in the **TO axis** manual.

Dynamic response with respect to data transmission from the acquisition to the processing

When using a PROFIBUS or PROFINET bus system, the specified times for data transmission (T_{DX}) must be taken into account in the calculation of the system cycle clock. Small T_{DX} allow for shorter cycle clocks or more processing time for the application with the same cycle clock length.

- For PROFIBUS DP T_{DX} is shown by the HW config.
- On SIMOTION D, a transmission time in the range of 125 µs ≦ T_{DX} ≦ 375 µs is set automatically on PROFIBUS integrated depending on the degree of extension of the SINAMICS project (I/O extension on SIMOTION D and SIMOTION CX32).
- For PROFINET IO $T_{DX} = 0.5 \bullet T_{DP}$ is set automatically

Dynamic response for data acquisition and data output

When configuring time T_O take T_{DX} into consideration. The following always applies: $T_{DX} \leqq T_O$.

- The times can be set in HW Config.
- When using TM15/TM17 High Feature modules for acquisition and output, the following times are to be set:
  - For the acquisition: T_I + 1 DRIVE-CLiQ cycle (typically 125 µs)
  - For the output T_O + 1 DRIVE-CLiQ cycle (typically 125 µs)

See also

Determination of Tdp, Ti and To using HW Config for ET 200 I/O devices on the PROFIBUS (Page 1391)

Determination of Tdp, Ti and To using HW Config for ET 200 I/O devices on the PROFIBUS

In HW Config in the **Isochronous mode** screen form (called from the **Edit > Isochronous mode** menu), the times for T_{DP}, T_I and T_O can be determined.

The dialog box gives an overview of the parameters and times set for the isochronous mode of the involved objects PROFIBUS, slaves and modules in the respective slaves.

All times in the dialog are specified in milliseconds.

- T_I/T_O setting
  - **in the network** means that the times T_I and T_O are centrally set "same for all slaves"
  - **in the slave** means that the times T_I and T_O are set individually on each slave
- T_I, T_O, T_{DP}
  Shows the currently set or calculated values T_I, T_O and T_{DP} for the selected DP master system.

For a detailed description of the screen form and all other parameters, please refer to the online help for HW Config.

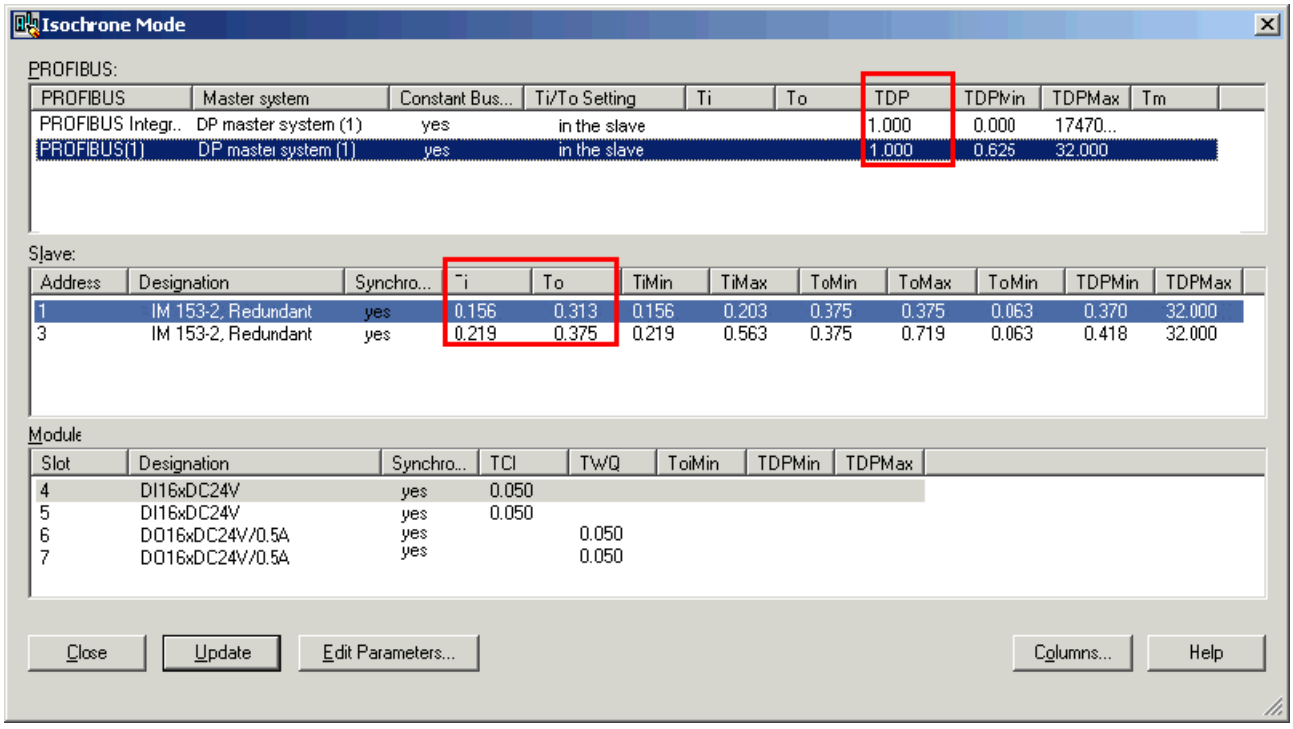


Figure 4-145 Isochronous mode screen form in HW Config

Note

Full "terminal-to-terminal" support of isochronous mode is only possible if all components within the sequence support the "isochronous mode" system property. When selecting from the PM10 catalog or the hardware catalog of HW Config, pay attention to the entry **Isochronous mode** in the information field of the module.

A current list of isochronous I/O modules can be found on the Internet under http://support.automation.siemens.com/WW/view/de/14747353.

Handling cycle clock scaling

The following must be observed for cycle clock scaling:

- At the end of an IPO synchronous task the process image is output with the next possible servo (Data Out) (= reaction time optimized). When scaling the servo cycle clock to the IPO cycle clock sooner or later this can cause the data of one or more servo cycle clocks to be output within an IPO cycle clock, if the I/O access are executed via the IPOSynchronousTask.
- At the end of the servo execution level, the process image of the ServoSynchronousTask is output with the next possible bus cycle clock (= reaction time optimized).
  - For PROFINET and downscaling PROFINET cycle clock to servo cycle clock this can cause the data of one or more bus cycle clocks to sooner or later be output within one servo cycle clock if the I/O accesses are executed via the ServoSynchronousTask and the runtime of the server execution level fluctuates beyond one bus cycle clock.
  - For PROFIBUS the data are always output with the first bus cycle clock as the servo execution level must always be concluded with the first bus cycle clock.

Thus with a different runtime of the servo execution level in the individual cycle clocks, the terminal-terminal time may vary.

If an always constant reaction time is to be achieved instead of a reaction time optimized behavior, the following must be set:

- For PROFIBUS:
  - A reduction ratio servo: IPO = 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode.
    Comment: I/O accesses from the ServoSynchronousTask are always isochronous for PROFIBUS
- For PROFINET:
  - A reduction ratio bus cycle clock : Servo: IPO = 1 : 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode
  - A reduction ratio bus cycle clock : servo = 1 : 1 so that the I/O accesses from the ServoSynchronousTask are always implemented in isochronous mode

4.2.7.8 Integrating DCC into the SIMOTION execution system

Introduction

Introduction

Drive Control Chart (DCC) allows you to implement your regulating and control tasks in a drive-related manner. For this there is a set of blocks (DCB) available to you that you can interconnect graphically via a configuration tool (DCC editor) in so-called charts. The DCBs are available to you in the form of a library (DCBLIB).

Note

One exception in the DCC tasks does not call up a SIMOTION fault task.

Thus the charts created can be executed on the SIMOTION platforms as well as on SINAMICS drives.

Only DCC for Simotion is described below.

Additional references

DCB lib DCC editor description

Sequence model for DCC blocks (DCB)

Description

The individual DCC blocks (DCB – Drive Control Block) are organized as runtime groups in the DCC editor. You can freely assign these runtime groups to the DCC tasks in the DCC editor (task T1 to T5). The DCC tasks in the DCC editor correspond to the DCC tasks in the SIMOTION execution system.

Task in the DCC editor	Execution level	Task
T1	Servo	servoDcc
D2	IPO	ipoDcc
T3	IPO_2	IpoDcc_2
T4	DccAux	dccAux
T5	DccAux_2	dccAux_2

Execution groups can be activated or deactivated via binary outputs of blocks. Detailed description in the editor.

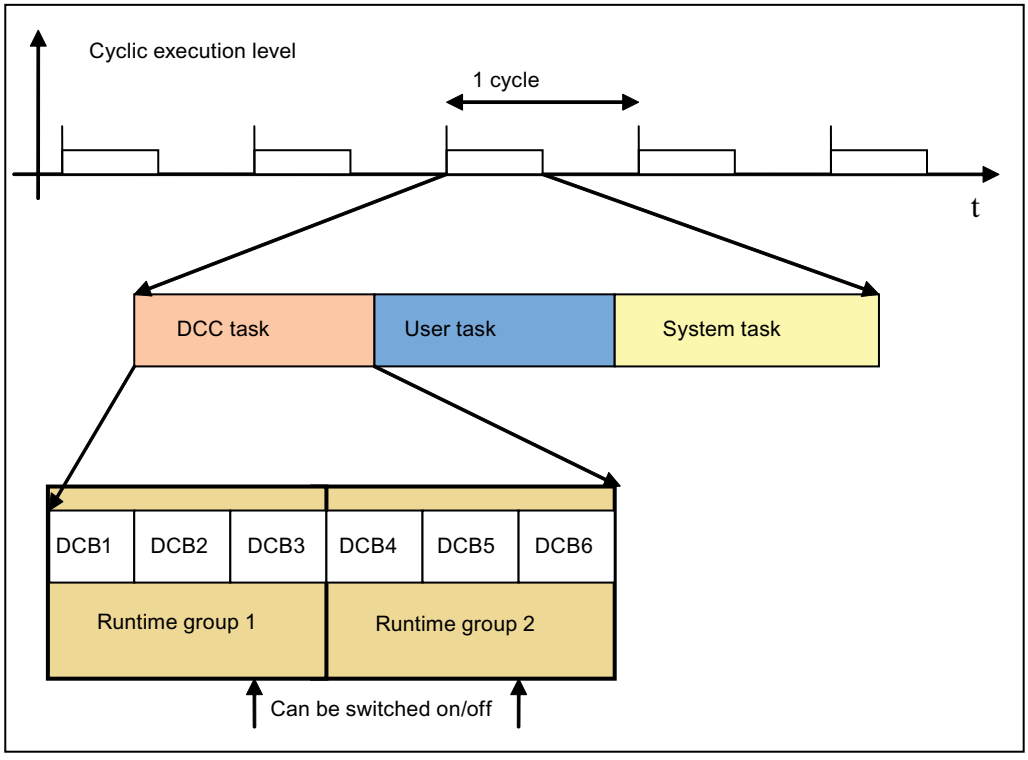


Figure 4-146 Sequence model of the runtime groups for DCBs

In the user model, the DCC task assigns itself in the execution level next to the user program tasks (execution environment for user programs in ST, MCC, LAD/FBD) and the system tasks. If required, the DCC task is created by the Engineeringsystem (ES); this means it is *not* present in the system as standard.

Execution of the DCC diagrams

The execution of DCC diagrams is linked to the task status but rather to the status of the runtime group assigned to it. They are enabled and disabled depending on the mode changes.

Mode	Action
STARTUP->RUN	ON runtime groups
SHUTDOWN->STOPU	OFF runtime groups
SHUTDOWN->STOP	OFF runtime groups

All other changes in the mode cause no changes to the runtime groups.

See also

servoDccTask in the servo level (Page 1395)

ipoDcc Task in the IPO level (Page 1396)

ipoDcc_2 Task in the IPO2 level (Page 1398)

Execution levels for DccAux and DccAux_2 (Page 1398)

servoDccTask in the servo level

Description

The T1(DCC) task runs in the servo level of the SIMOTION execution system. The servo level is called synchronous to the data exchange with the isochronous I/O and used primarily for fast control tasks.

- At the start, the cyclic I/O data for the technology objects and servo-synchronous I/O variables is copied from the interface to the isochronous I/O in the execution system.
- At conclusion, the cyclical I/O data is copied again from the execution system and, for example, transferred with PROFIBUS to a PROFIBUS node.

The following graphic shows the chronological sequence of a task in the servo level.

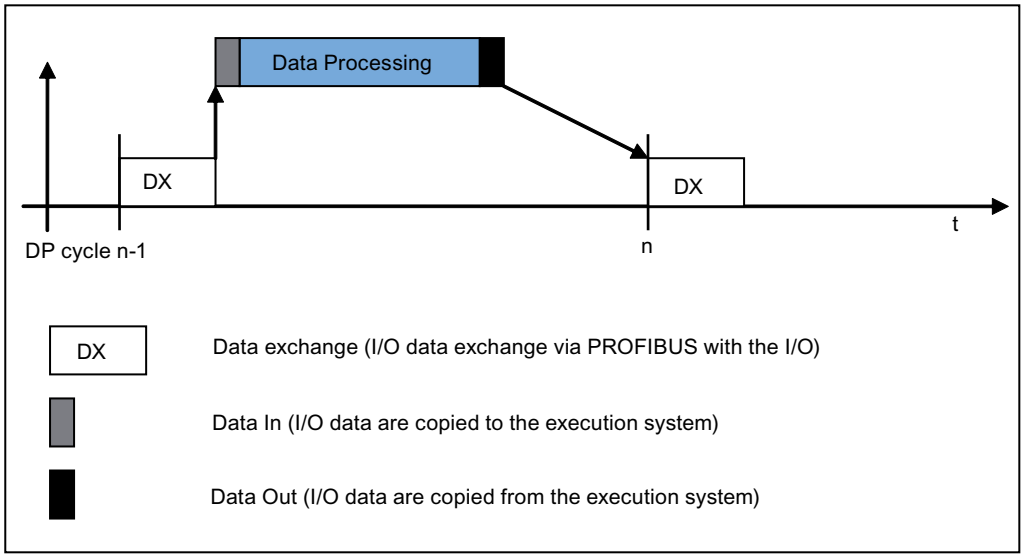


Figure 4-147 T1(DCC) task in the servo level

The T1(DCC) task is automatically started and executed at the begin of the servo level.

The following graphic shows the sequence of the tasks in the servo level.

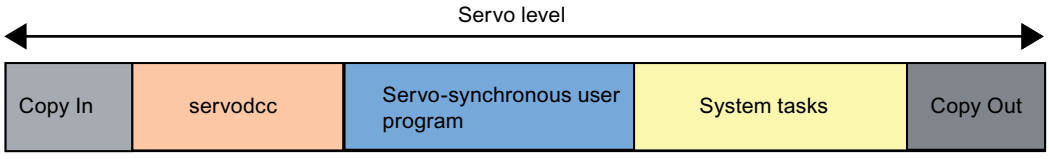


Figure 4-148 Servo level with T1(DCC) task

Level overflow in the servo level

Tasks in the servo level run with the same priority and so do not interrupt each other. If all tasks of the servo level cannot be calculated in a single cycle, a level overflow occurs and the system enters STOP mode and the start-up lock is set. Only after a new startup (power on/off) or download can the system return to RUN mode.

See also

Defining system cycle clocks (Page 1350)

ipoDcc Task in the IPO level

Description

The IPO level runs with the same cycle time as the servo level or in an integer reduction ratio. Tasks in the servo level have a higher priority and so can interrupt tasks in the IPO level. In addition, in the IPO level all I/O data not copied in the servo level are copied into the execution system. The data can come both from the isochronous and the non-isochronous I/O.

The command processing and the setpoint generation of the technology objects run in the system task of the IPO level.

The following graphic shows the chronological sequence in the IPO level.

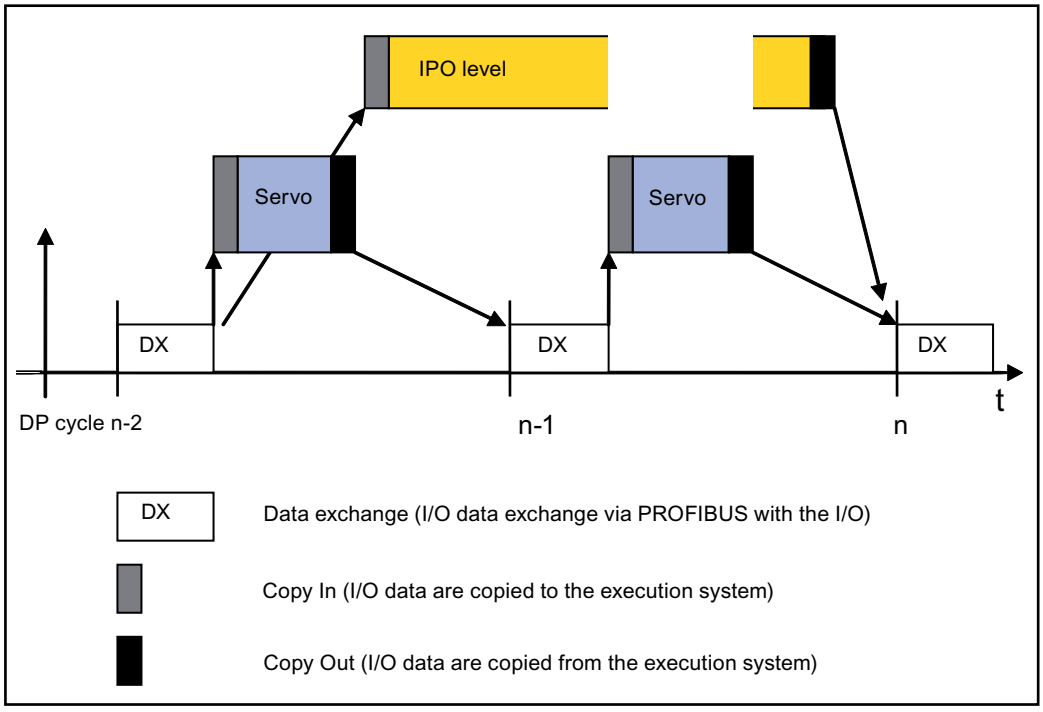


Figure 4-149 IPO level with T2(DCC) task

The T2(DCC) task is automatically started and executed at the begin of the IPO level.

The following graphic shows the execution sequence in the IPO level.

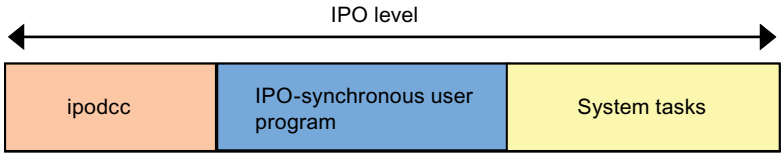


Figure 4-150 Execution sequence in the IPO level

Execution overflow in the IPO level

Tasks in the IPO level run with the same priority and so do not interrupt each other. A task in the higher priority servo level can at anytime interrupt the processing in the IPO level; this can possibly cause a level overflow, because not all tasks in the level can be processed.

With the standard behavior, for a level overflow the device enters into STOP mode and the start-up lock is set. Only after a new startup (power on/off) or download can the system return to RUN mode.

However, you can also configure the system for toleration of up to five successive overflows. You must set the behavior appropriately in the task configuration.

See also

Defining system cycle clocks (Page 1350)

Timeouts and level overflows (Page 1360)

SynchronousTasks (Page 1326)

Isochronous data processing (Page 1385)

ipoDcc_2 Task in the IPO2 level

Description

The cycle time of the IPO2 level is an integer multiple of the cycle time of the IPO level. However, it is not possible to set the IPO2 cycle clock equal to the IPO cycle clock.

The cycle time of the IPO2 level is shorter than that of the IPO level. Consequently, both the IPO level and the servo level can interrupt the processing.

The T3(DCC) task is assigned in the IPO2 level as follows:

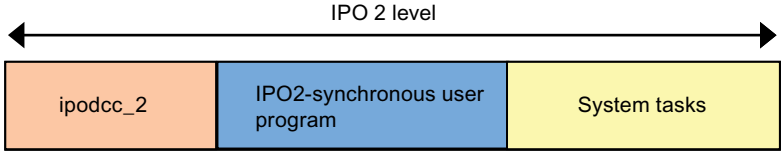


Figure 4-151 Order of processing in the IPO2 level

Level overflow in the IPO2 level

In the case of a level overflow, the IPO2 level behaves like the IPO level.

See also

ipoDcc Task in the IPO level (Page 1396)

Defining system cycle clocks (Page 1350)

Execution levels for DccAux and DccAux_2

Description

The two DCC execution levels, DccAux and DCCAux_2, are called synchronous to the system.

The DCCAux and DCCAux_2 levels have the following properties:

- DccAux has a multiple of the Ipo_2 cycle clock.
- DccAux_2 has a multiple of the DccAux cycle clock.
- DccAux has a higher priority than DccAux_2
- Synchronous trace recordings are possible.

- Up to five level overflows are tolerated (the default is 1). The current number of level overflows can be fetched.
- In the case of a level overflow, the DCCTask is calculated to completion in the next task.

Table 4-77 System variables for fetching level overflows

System variable	Data type	Meaning
_device.numberOfSummarized-TaskOverflow.DccAux	UDINT	Number of overflows of the DccAux level since system startup
_device.numberOfSummarized-TaskOverflow.DccAux_2	UDINT	Number of overflows of the DccAux_2 level since system startup

See also

Defining system cycle clocks (Page 1350)

Data exchange between blocks

Data exchange between blocks (overview)

Overview

In the DCC editor, you can interconnect the connections of various blocks. Three basic scenarios must be differentiated for the data exchange between the blocks:

- You have connected connections of blocks with each other that lie in the same execution level.
- You have interconnected the output of one block with the input of a block in a higher execution level.
- You have interconnected the output of one block with the input of a block in a lower execution level.

See also

Data exchange between blocks in the same level (Page 1400)

Data for blocks from a lower-priority level (Page 1402)

Data for blocks from a higher-priority level (Page 1405)

Data exchange between blocks in the same level

Description

If you have interconnect the blocks in the same execution level, the data will be transferred in accordance with the execution sequence of the blocks. The guarantees that the value is current at the input of the execution sequences.

If you have defined the execution sequence in accordance with the data flow, no additional dead time results in the level.

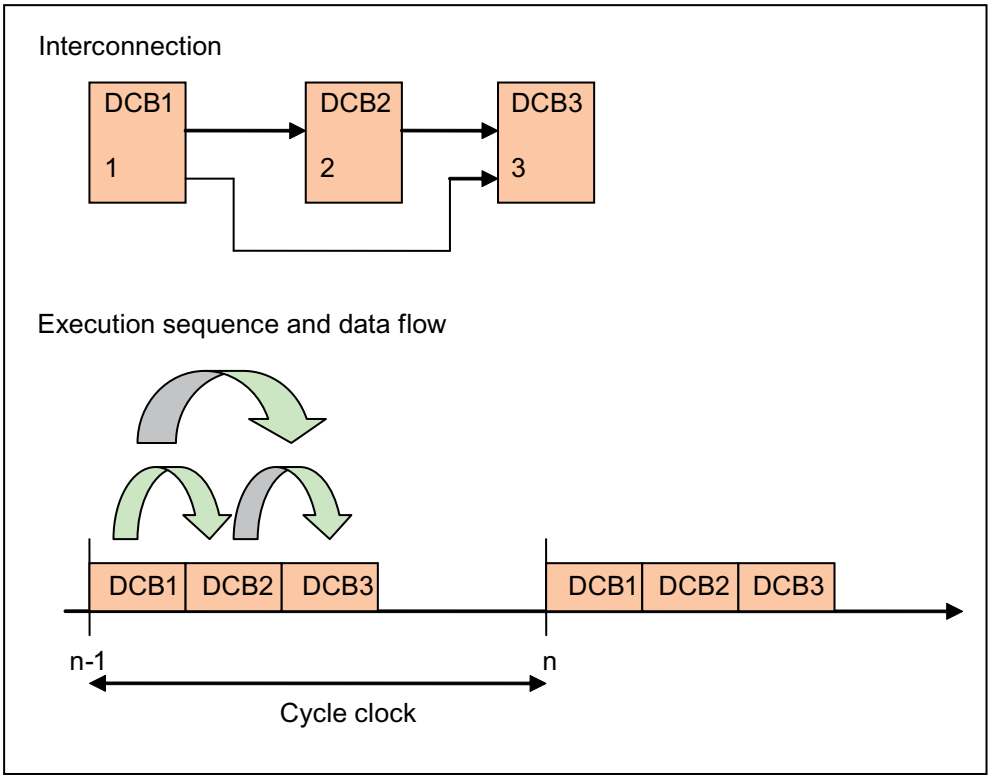


Figure 4-152 Interconnection sequence and execution sequence are identical

If the execution sequence of the blocks in the level does not correspond to the data flow, additional dead times result, because at the input access has been made to values from the previous cycle clock.

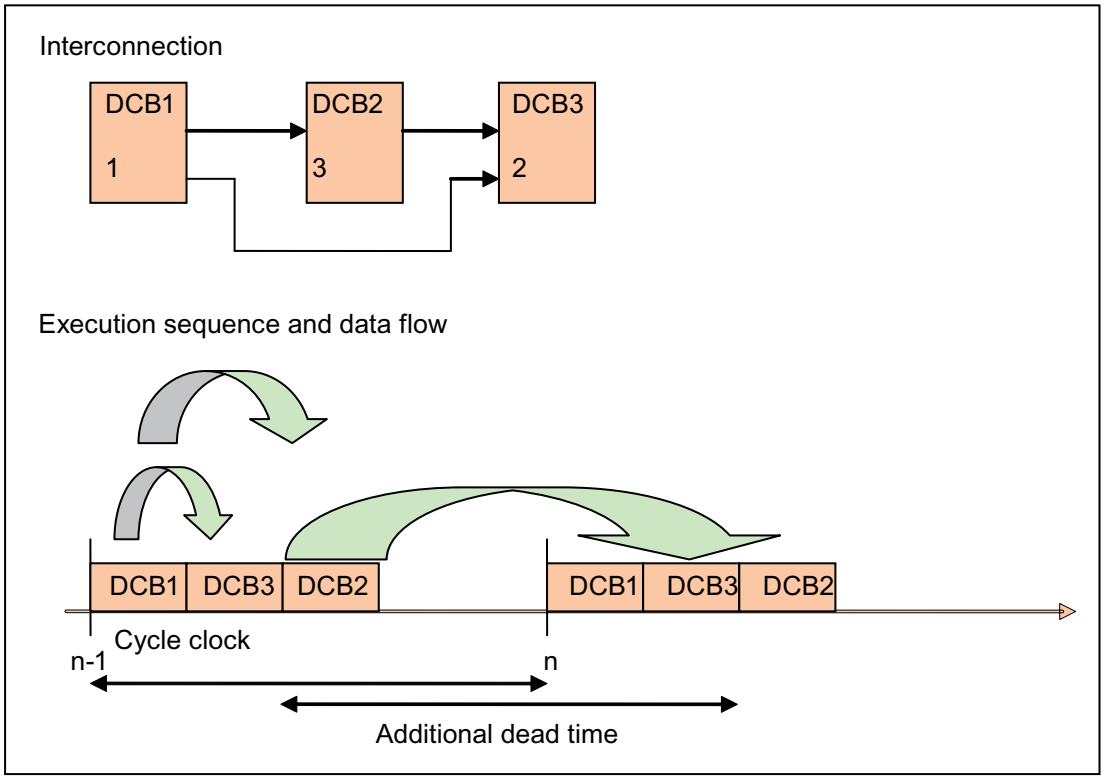


Figure 4-153 Interconnection sequence and execution sequence differ

To optimize the dead times, always orient the execution sequence on the data flow.

Data for blocks from a lower-priority level

Description

All blocks in an execution level must be calculated before their output values are made available to blocks at the input in a different execution level. Output values from a higher-priority level are accessed in a dedicated cycle clock (acceptance cycle clock) in order to ensure isochronism of input values. The following graphic illustrates the isochronous value transfer in a higher-priority level.

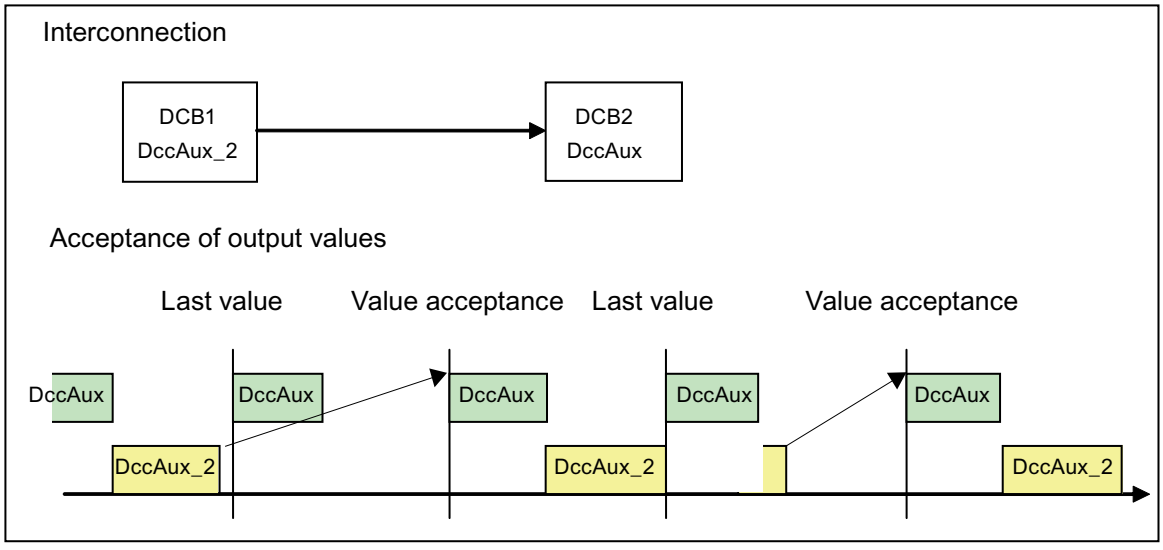


Figure 4-154 Example for the data exchange from a lower priority level

The cycle clock ratio between the levels is 1:2. The task in the higher-priority level always accepts the value in a second cycle clock, independent of whether the lower-priority task was already calculated to the end in the previous cycle clock.

Level overflow of the lower-priority level

If a level overflow of the lower-priority level occurs, the current values have not yet been calculated at the acceptance cycle clock. In this case, the old value is used for calculations at the input until the next acceptance cycle clock.

The following graphic illustrates the overflow behavior for a cycle clock ratio of 1:2. A level overflow for T2 occurs. This means the old values are transferred to the higher-priority task at the acceptance time (cycle clock 1). A new value is accepted at the next acceptance time, namely two cycle clocks later.

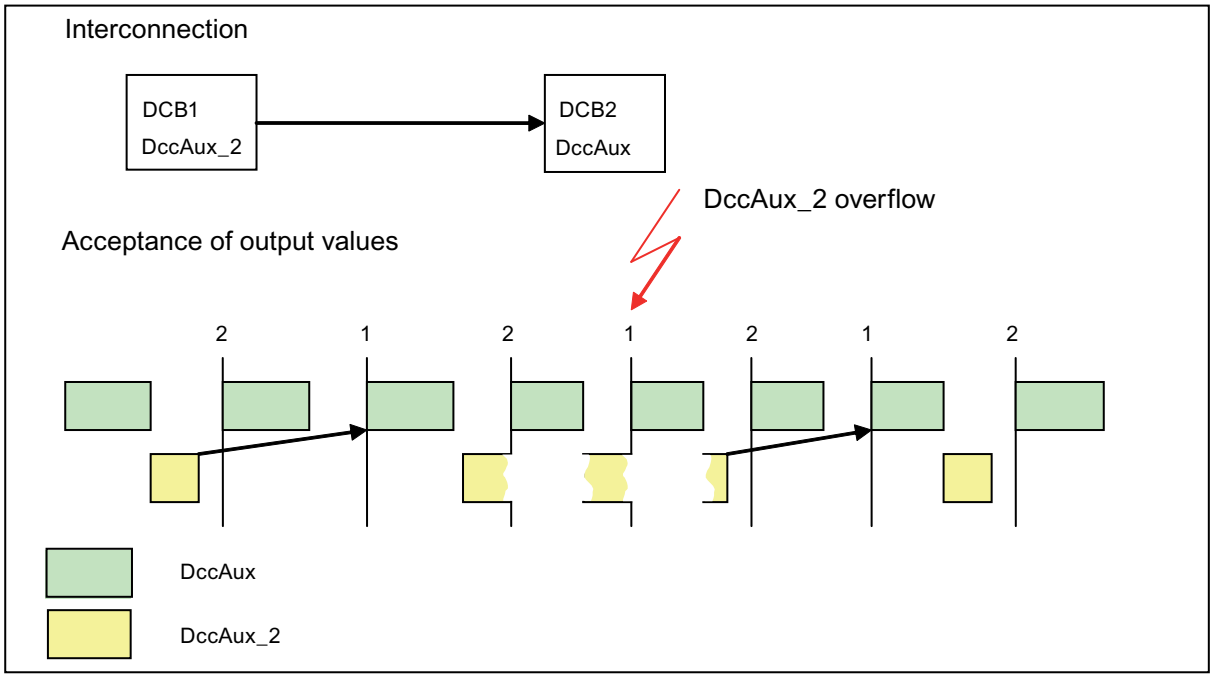


Figure 4-155 Data exchange from a lower-priority level with overflow

The described overflow scenario is true only for an ideal system. In a real application, user tasks and system tasks also run in a level together with the DCC task.

Example consideration of user programs and system tasks

If overflow occurs during the processing of the iposynchronous UP or of the IPO system task, the values have already been calculated in the IPoDCC task and can be transferred to the higher-priority level. However, no new value is calculated in the next cycle clock. If the ipoDCC task already causes an overflow, a correct value will only be accepted again two cycle clocks later. In the two cycle clocks between, the value is accepted delayed by one cycle clock.

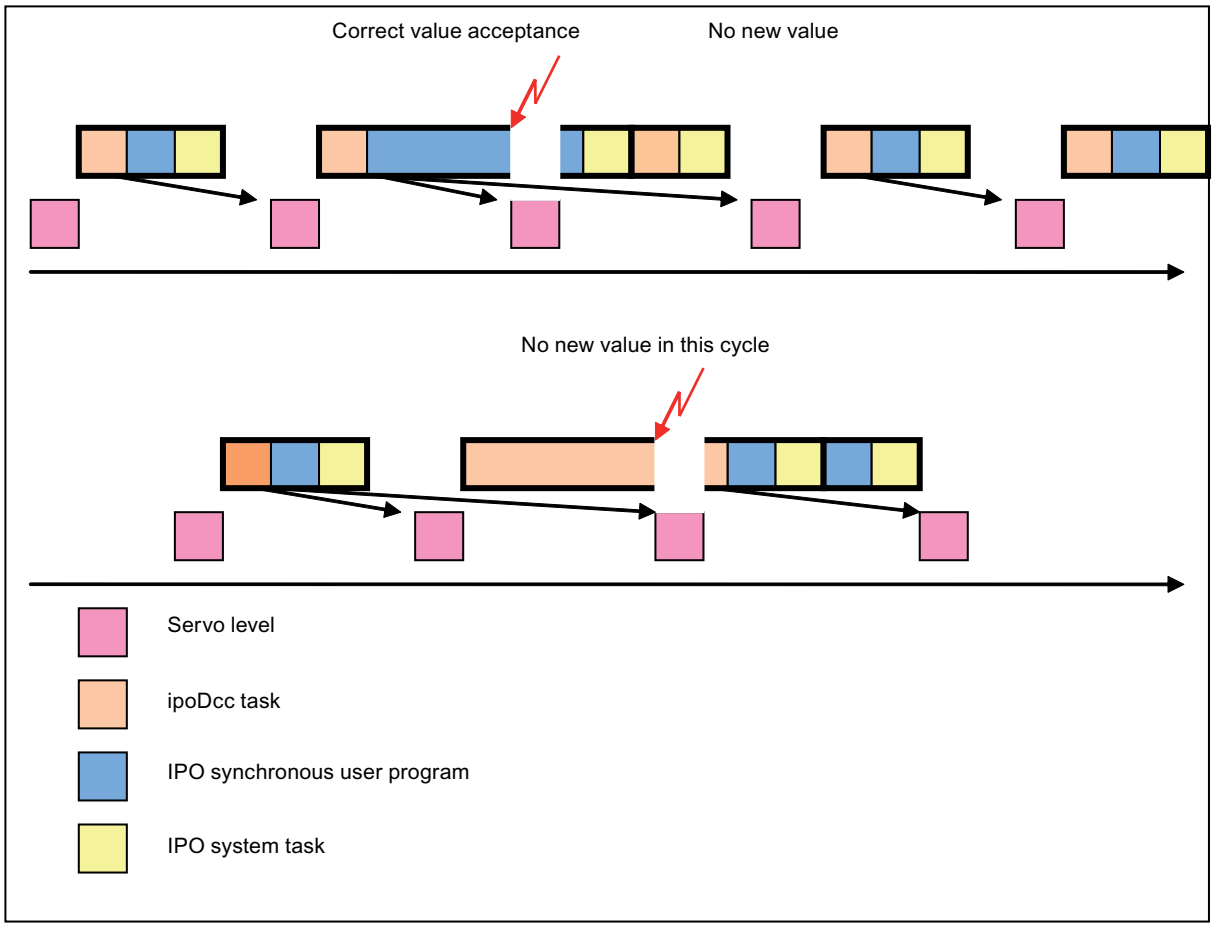


Figure 4-156 Overflow in the execution context with other tasks

Note

To ensure an isochronal data transfer between the ipoDCC task in the IPO level and the servoDCC in the servo level, the ipoDcc task in the IPO level considered by itself should not already cause a level overflow. If this is the case, the higher-priority task does not have available any updated values for the transfer of the values from the lower-priority level. The updated values are then transferred only after the next cycle of the lower-priority levels.

Data for blocks from a higher-priority level

Description

If an access is made to the output values from a lower-priority level, the values must be transferred from a dedicated cycle clock of the higher-priority level in order to guarantee the isochronism of input values.

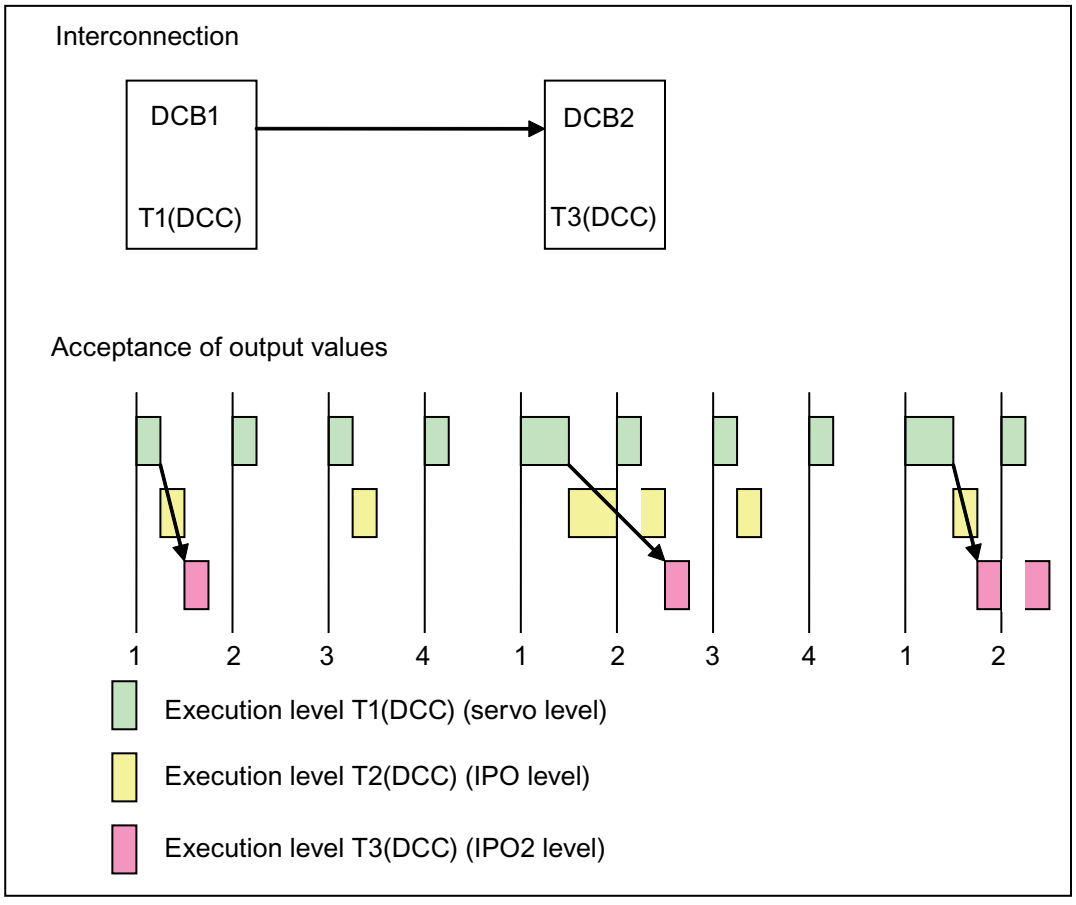


Figure 4-157 Data transfer from a higher-priority level

Interconnection of blocks with variables

Interconnection with variables

Overview of the interconnection possibilities with variables

In addition to the possibility of interconnecting blocks, a PIN can also be interconnected to an external variable. In this way, you can establish a connection with the I/O, the technology objects, a user program and an HMI.

The following user variables can be interconnected with blocks:

- **I/O variables**.
- **System variables**
- The following **user variables** can be interconnected with DCC:
  - Global device variables
  - Variables in the interface of a source

Interconnection of technology objects

- Interconnection using system variables and cyclic interface
- System functions of TOs cannot be called directly from DCCs.

Interconnect PIN with variables

You can interconnect the various inputs and outputs of a block in the DCC editor. The description of the DCC editor contains detailed information.

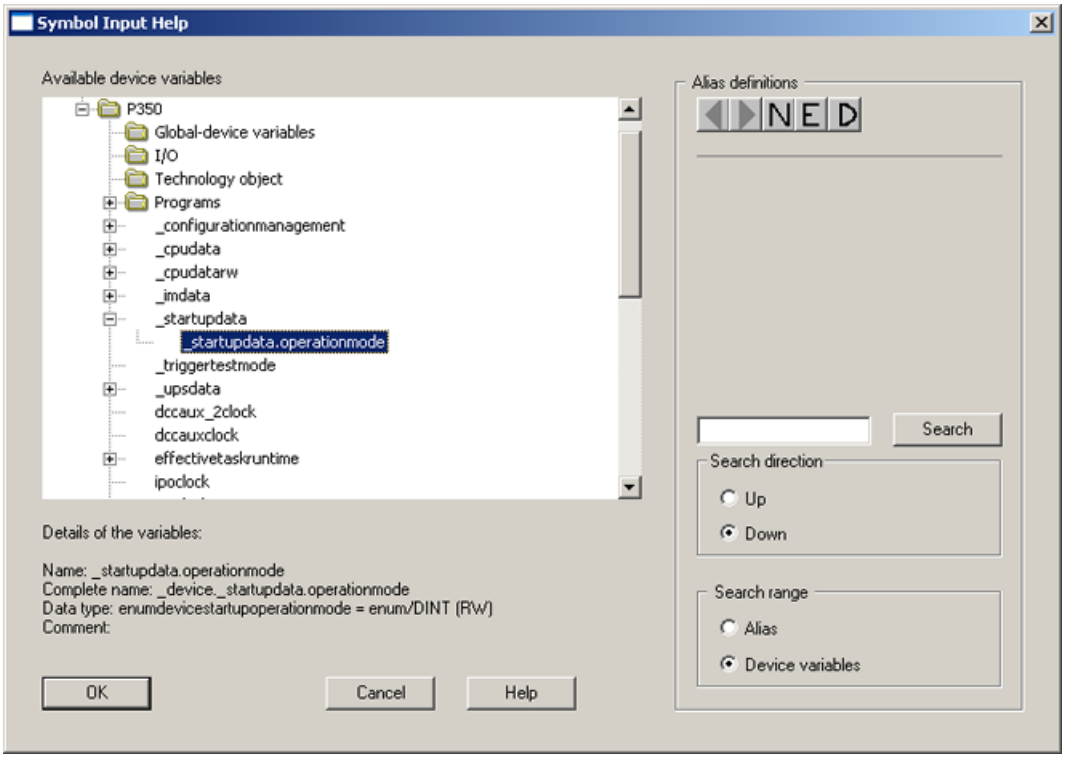


Figure 4-158 Interconnection of user variables in the DCC editor

Note

There are no restrictions as to how often an output can be interconnected to a variable. The output value of the most recently calculated block is always effective.

Behavior for FPU exceptions

Description

DCC Tasks exhibit the following behavior for FPU (Floating-Point Unit) exceptions:

Exception	Description	SIMOTION response
Invalid Operation	Invalid operation (e.g. 0/0, INF/ INF, INF - INF, SQRT(-1)	Device switches to STOP mode.
Division by Zero	Division by zero	Device switches to STOP mode.
Overflow	The absolute result of an operation is larger than abs(+/-- Nmax)	Device switches to STOP mode.
Underrun	The absolute result of an operation is less than abs(+/-- Nmax)	The result is 0.
Inexact	The result cannot be represented exactly.	The result is rounded.

See Error during operations with floating-point numbers (FPU exceptions) (Page 1257).

4.2.7.9 Include drive I/O

Assigning the drive I/O symbolically

Description

As of V4.2, the drive I/O can be assigned symbolically to the I/O variables. For more information, refer to Assigning I/O variables via the address list (Page 1202).

Terminal modules TM15 and TM17 High Feature

Description

The TM15 and TM17 High Feature terminal modules must be evaluated isochronously by the SIMOTION motion control system. Details for the timing can be obtained from the TM15 / TM17 High Feature Terminal Modules Commissioning Manual.

Terminal modules TMxx, TB30 and onboard I/Os on SIMOTION D or CX32/CX32-2 and CU3xx/ CU3xx-2

Description

TM31, TM41, TM15 DI/DO, TB30 and the onboard I/Os operate free-running (not isochronous) for SINAMICS.

The updating cycle is specified by the following drive parameters with 4 ms as default setting:

- p4099 for TM31, TM41 and TM15 DI/DO or
- p0799 for the onboard I/O

The accesses are performed in a SINAMICS background task (for example, every 4 ms). Access time to the inputs and outputs within the set updating cycle can vary from cycle to cycle; this means that the system merely ensures that within each cycle an update is made with a possible fluctuation range in accordance with the updating cycle. This can be influenced using the setting for p4099 or p0799.

Note

The following figure only provides a schematic presentation of the times and does not provide information on the absolute amount.

The I/O produces the following timing:

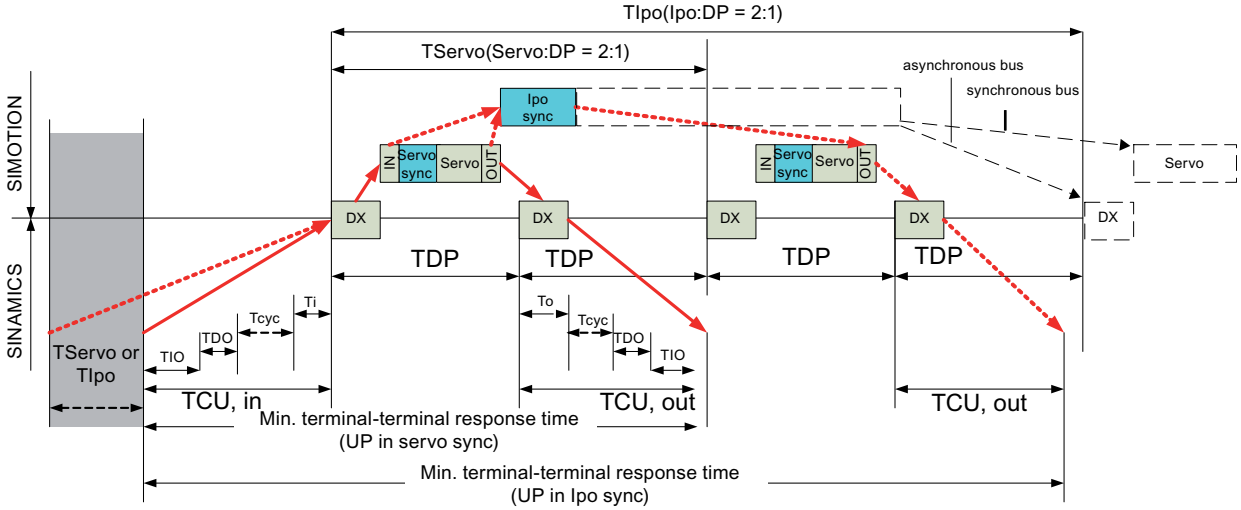


Figure 4-159 I/O Timing

Legend

- T_{DP}: Clock cycle PROFIBUS (see setting in HW Config)
- T_i: Latch time of the inputs for isochronous PROFIBUS (see setting in HW Config for the drive)
- T_o: Delay of the outputs for isochronous PROFIBUS (see setting in HW Config for the drive)
- T_{IO}: Module-specific signal delay (corresponds to a DRIVE-CLiQ cycle + the input delay time for digital input or the load-dependent output delay time for digital outputs)
- T_{cyc}: Updating cycle in the SINAMICS (p4099 or p0799 parameter)
- Servo: SIMOTION servo cycle clock; multiple of T_{DP} (see cycle clock settings for SIMOTION)
- IPO: SIMOTION IPO-cycle clock; multiple of the servo cycle clock (see cycle clock settings for SIMOTION)
- T_{DQ}: DRIVE-CLiQ clock cycle (for V4.1, always corresponds to the current controller cycle clock)

- UP: User program
- Output time on the CU: $T_{CU\_out} = T_O + T_{cyc} + T_{DQ} + T_{IO}$
- Read in time on the CU: $T_{CU\_in} = T_i + T_{cyc} + T_{DQ} + T_{IO}$

The figure shows all times that affect the terminal-terminal response time. Worst case values should be used for the dashed times, thus:

- The set updating cycle T_{cyc} (= maximum time for the access time)
- The time T_{servo} or T_{IPO} in the gray area. Depending on the time the input event occurs the maximum terminal-terminal reaction time can be up to one position control or IPO cycle clock longer than the minimum terminal-terminal reaction time.

The gray area shows the range in which the input event is acquired and processed with the next call of the user program (UP). The size of this range depends on whether the user program runs in the servo-synchronous task or IPO-synchronous task.

Note

Please note that a reduction of the update cycle leads to higher utilization of the SINAMICS closed-loop control or the SIMOTION D, which may result in a reduction in the quantity frame for drives, terminal modules, etc., under certain circumstances.

It is possible to access I/O components in the SINAMICS drive unit from the SIMOTION user program:

- As of V4.2, this can be carried out by means of symbolic assignment of the I/O variable to the SINAMICS I/O. SIMOTION automatically configures the required BICO interconnections and telegrams
- In V4.1, use can be made of this function with special telegrams (39x)
- In earlier versions, the I/O components are inserted in the PROFIdrive message frame as I/O PZD (process data) using a BICO interconnection (refer to the SIMOTION D Commissioning and Hardware Installation Manual, section "Message frame configuring for onboard I/O and drive objects").

The timing depends on which I/O component is accessed. The following table specifies the maximum delay times. To determine the terminal-terminal times, the values for the "terminal → user program" and "user program → terminal" must be added.

Note

The dashed line areas in the **IO Timing** graphic for the IPO synchronous task only apply for PROFINET IO. Analogously the times for PROFINET IO are presented in the following table in the **UP in IPOsynchronous task** column. For PROFIBUS DP in the line **OUT** T_{IPO} must be replaced by T_{Servo}. The dotted line area is irrelevant for PROFIBUS as the servo cannot run there over multiple bus cycles.

			Maximum delay times		
I/O module			**UP in servo-synchronous task**	**UP in iposynchronous task**	**T_{cyc}**
TM31, TM41, TM15 DI/O	Out	UP → terminal	$T_{DP} + T_{CU\_out}$	$T_{IPO} + T_{DP} + T_{CU\_out}$	p4099
	In	Terminal → UP	$T_{Servo} + T_{CU\_in}$	$T_{IPO} + T_{CU\_in}$	
TB30	Out	UP → terminal	$T_{DP} + T_o + T_{cyc} + T_{IO}$	$T_{IPO} + T_{DP} + T_o + T_{cyc} + T_{IO}$	p4099[3)]
	In	Terminal → UP	$T_{Servo} + T_i + T_{cyc} + T_{IO}$	$T_{IPO} + T_i + T_{cyc} + T_{IO}$	
Onboard I/O SIMOTION D, CX32, CX32-2, CU3xx, CU3xx-2	Out	UP → terminal[1)]	75 µs	75 µs	p0799[4)]
		UP → terminal[2)]	$T_{DP} + T_o + T_{cyc}$	$T_{IPO} + T_{DP} + T_o + T_{cyc}$	
	In	Terminal → UP	$T_{Servo} + T_i + T_{cyc}$	$T_{IPO} + T_i + T_{cyc}$	

1) Access to SIMOTION D integrated drives in conjunction with standard telegram 39x for the DO1. Without the standard message frame, the same timing behavior applies as for CX32 or CX32-2.

2.) Access to CX32, CX32-2 as well as via external PROFIBUS or PROFINET IO on CU3xx and CU3xx-2

3) For isochronous operation, T_{cyc} = max (T_{DP}, p4099)

4) For isochronous operation, T_{cyc} = max (T_{DP}, p0799)

Further information

You can also find more information on this topic:

- In the SIMOTION D410/D410-2 and D4x5/D4x5-2 Commissioning Manuals
- In an FAQ on the Utilities & Applications CD under FAQs
- On the Internet at http://support.automation.siemens.com/WW/view/de/27585482.

4.2.7.10 Time for SIMOTION and SINAMICS

SIMOTION time (real-time clock)

All SIMOTION devices have an integrated real-time clock. All events on a module (alarms, messages, etc.) are "time-stamped" based on the time shown by this real-time clock.

Setting the SIMOTION time of day

How to set the clock from SIMOTION SCOUT:

1. Select the SIMOTION device in the project navigator
2. Then select the **Target system > Set time of day** menu.

Alternatively, the clock can be set using the RTC system function block.

Synchronizing SINAMICS time of day

SINAMICS system runtime (operating hours counter)

With the SINAMICS Integrated of a SIMOTION D, controller extensions and the SINAMICS S120 control units, faults and warnings are "time-stamped" on the basis of the system runtime. This means that events are recorded by default on the basis of operating hours rather than a particular time of day or date.

System runtime

The entire system runtime is displayed in CU parameter r2114.

- r2114[0] shows the system runtime in milliseconds. After reaching 86,400,000 ms (24 hours), the value is reset.
- r2114[1] shows the system runtime in days.

The counter value is saved when the power is switched off. After the drive unit has been switched on, the counter continues to run with the value stored when the power was last switched off.

As a result, the drive displays the system runtime from 00:00:00 on 01.01.1992 in both the alarm window in SIMOTION SCOUT and the diagnostic buffer for entries.

If faults and warnings need to be "time-stamped" based on a time of day, "Time stamp operating hours" needs to be changed to "Time stamp UTC format" as described below.

Requirements

Telegram 39x is required for the time synchronization. If the automatic PROFIdrive telegram setting is selected for the control unit, this telegram is used automatically.

If the telegrams are specified manually, telegram 39x must be set up (see Section Telegram configuration (Page 1212)).

So that drive units can be synchronized with the SIMOTION time, they must support telegram 39x and the UTC time format.

For precise time synchronization, the drive unit must also be operated via an isochronous/ equidistant bus on SIMOTION. The SINAMICS Integrated for SIMOTION D and the CX32/CX32-2 controller extension always have an isochronous/equidistant connection.

The following control units support time synchronization:

- SINAMICS Integrated of the SIMOTION D
- CX32/CX32-2 controller extensions
- SINAMICS S120, CU310, CU310-2, CU320, CU320-2 control units, connected via PROFIBUS or PROFINET

Procedure

Proceed as follows to convert the SINAMICS clock to UTC format and to synchronize this with the SIMOTION clock:

1. Select the SIMOTION device in the project navigator.
2. Select the **Edit > Object Properties** menu.
3. Select the **Settings** tab.
4. Activate the option **Perform time synchronization with SINAMICS drive units**.
5. Confirm with OK.

Note

This setting is automatically activated for new projects as of V4.2 and applies for all drive units connected to the SIMOTION device. The SINAMICS clock is automatically synchronized with the SIMOTION clock for all drive units with configured telegram 39x.

The first time synchronization is performed after the SIMOTION device has reached the RUN operating state.

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically resynchronized at regular intervals.

Using the system variable **_driveStates.allClocksSynchronized** on the SIMOTION device, the user program can query whether the automatic time synchronization is activated (=YES) or deactivated (=NO).

Before the first synchronization, alarms and messages are stored with the time stamp valid in the SINAMICS at this time, all subsequent alarms and messages with the synchronized time.

The first time synchronization after switching on is entered with the status of the operating hours counter and the time (UTC time, synchronized with SIMOTION) in the diagnostic buffer of the drive (e.g. SINAMICS Integrated).

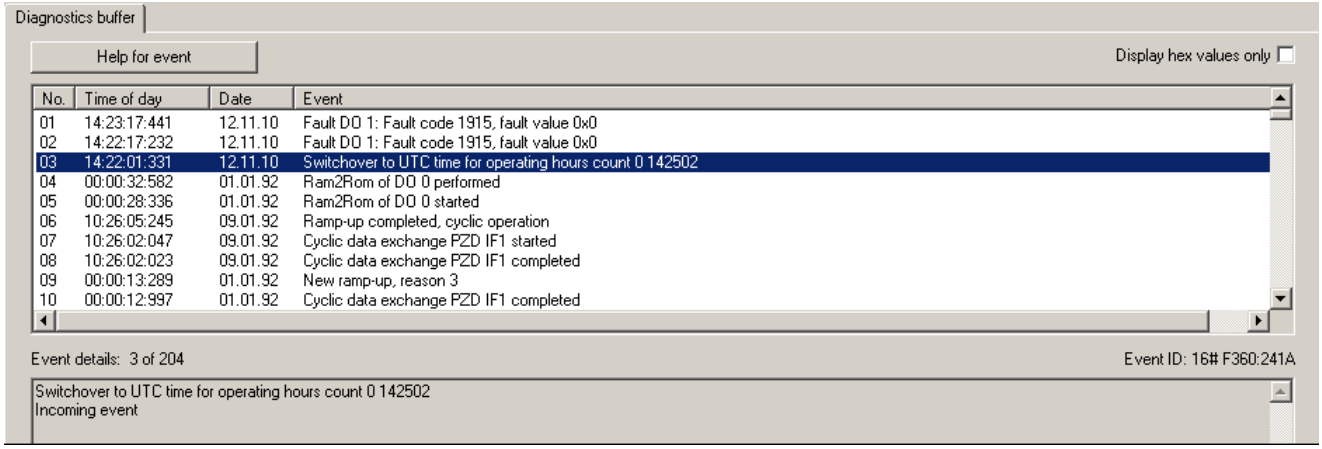


Figure 4-160 SINAMICS time synchronization diagnostic buffer entry

Error correction

If problems occur during the time synchronization, this may be due to faulty configuration information (Fast IO configuration) when symbolic assignment is deactivated.

For further information, see Section Message frame configuration (Page 1212).

Compensation of runtime deviations

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically resynchronized at regular intervals.

The following behavior must be taken into consideration when setting the SIMOTION time:

- "Time/date to be set" is after "Time/date on SINAMICS":
  Time and date are corrected on the SINAMICS.
- "Time/date to be set" is before "Time/date on SINAMICS":
  The SINAMICS clock must be stopped until the SINAMICS "Time/date" has caught up with "Time/date to be set".

Adopting this procedure ensures that the sequence of SINAMICS diagnostic buffer entries remains the same, even when the runtime differences are aligned.

The SINAMICS clock operates with a resolution of 1 ms. A synchronization accuracy of 1 ms can be achieved for all bus cycle clocks that can be divided exactly by 1 ms (e.g. 1 ms, 2 ms, 3 ms, etc.).

Due to system considerations, a slightly lower synchronization accuracy is achieved for all bus cycle clocks that cannot be divided exactly by 1 ms (e.g. 1.25 ms).

If the drive unit does not have an isochronous/equidistant connection on SIMOTION, this can result in runtime differences of milliseconds (depending on the set cycle clock ratios).

Resetting the time

Requirement:

- SIMOTION D4xx-2: As of SIMOTION V4.3
- SINAMICS S120: As of SINAMICS firmware version V4.5

A threshold value is defined using the CU parameter 3109 and is effective as follows:

- In the case of negative time jumps less than the threshold value (p3109), the time is halted (for details, see "Compensation of runtime deviations").
- In the case of negative time jumps greater than the threshold value (p3109), the time is reset.
  The factory setting of p3109 is 100 ms
  This means that in the case of negative time jumps greater than 100 ms, the time is reset.
  The factory setting is configured so that normal runtime deviations (drift of the quartzes) are below the threshold value.
  If the SIMOTION clock is reset by more than 100 ms, this is interpreted as a "specific reset of the time" and the time of the drives is also immediately reset.

If the real-time clock is reset by more than 60 seconds, a diagnostic buffer entry is also made in the drive:

Time correction (reset) by <correction value> seconds

After a re-synchronization (negative time gap greater than the threshold value), the CU parameter r3107 shows:

- In r3107[0..1] the UTC time after the synchronization
- In r3107[2..3] the UTC time before the synchronization

The milliseconds are in indices [0] and [2] and the days in [1] and [3].

Note

The diagnostic buffer entries are not converted to the new time when the time is changed.

4.2.8 Programming Execution System/Tasks/System Cycle Clocks

4.2.8.1 Execution system

Introduction to the execution system

All programs are executed in the SIMOTION device execution system. The execution system provides a series of execution levels with various execution properties.

Programs must therefore be assigned to the execution levels in order to be executed. To this end, programs of the source files are assigned to one or more tasks.

This defines the sequence in which they are to be executed.

Note

Before programs can be assigned to the execution levels the ST/MCC/LAD/FBD sources must be translated.

Execution levels and tasks

Execution levels define the chronological sequence of programs in the execution system. Each execution level contains one or more tasks.

A task provides the execution framework for the programs. You can assign one or more user programs to each task and specify their order within the task.

Besides user program tasks there are also several system tasks, the contents and execution sequence of which you cannot influence.

The table shows the execution levels with their tasks that are available for the user programs (user program tasks).

You can use task control commands to influence the execution system.

Table 4-78 Execution levels in SIMOTION

Execution level	Description
Time-controlled	Cyclic tasks, automatically restarted once assigned programs have been executed.
• SynchronousTasks	Tasks are started periodically, synchronous with specified system cycle clock. • ServoSynchronousTask: synchronous with the position control cycle clock (as of SIMOTION Kernel V4.0) • ServoSynchronousTask_fast: Synchronous with Servo_fast (Version 4.2 of SIMOTION Kernel and higher) as an option • IPOsynchronousTask: Synchronous with interpolator cycle clock IPO • IPOsynchronousTask_fast: Synchronous with interpolator cycle clock IPO_fast (Version 4.2 of SIMOTION Kernel and higher) as an option • IPOsynchronousTask_2: Synchronous with interpolator cycle clock IPO_2 • Synchronous tasks for the Tcontrol technology package: – PWMsynchronousTask: Synchronous to the PWM cycle clock InputSynchronousTask_1: Synchronous with cycle clock Input1 – InputSynchronousTask_2: Synchronous with cycle clock Input2 – PostControlTask_1: Synchronous with cycle clock Control1 – PostControlTask_2: Synchronous with cycle clock Control2
• TimerInterruptTasks	Tasks are started periodically in a fixed time frame. This time frame must be a multiple of interpolator cycle clock IPO.
Interrupts	Sequential tasks, executed once after start and then terminated.
• SystemInterruptTasks	Started when a system event occurs: Detailed description • ExecutionFaultTask: Error processing a program • PeripheralFaultTask: Error on I/O • TechnologicalFaultTask: Error on the technology object • TimeFaultBackgroundTask: BackgroundTask timeout • TimeFaultTask: TimerInterruptTask timeout
• UserInterruptTasks	Started on edge trigger when a user-defined event occurs.
Round robin	MotionTasks and BackgroundTasks share the free time remaining after execution of the higher-priority system and user tasks. The proportion of the two levels can be assigned.

Execution level	Description
• MotionTasks	Sequential tasks, executed once after start and then terminated. Start takes place: • Explicitly via a task control command in a program assigned to another task. • Automatically when RUN mode is attained if the corresponding attribute was set during task configuration. The priority of a MotionTask can be increased temporarily using the system function WAITFORCONDITION (see **Let MotionTask wait until a condition is satisfied**).
• BackgroundTask	Cyclic task, restarted automatically once the assigned programs have been executed; task cycle time depends on runtime.
StartupTask	Task is executed once when there is a transition from STOP or STOP U mode to RUN mode. SystemInterruptTasks are started by their triggering system event.
ShutdownTask	Task is executed once when there is a transition from RUN mode to STOP or STOP U mode. STOP or STOP U mode is reached by: • Corresponding mode selector switch position • Corresponding system function of the SIMOTION device • Alarm (fault) with corresponding error response SystemInterruptTasks and PeripheralFaultTasks are started by their triggering system event.
For information on behavior of sequential and cyclic tasks: • For the initialization of local program variables, refer to the Initialization of local program variables depending on the task execution behavior (Page 1418) table. • In the event of execution errors in the program, see Processing errors in programs (Page 1256) section. For information about options for accessing the process image and I/O variables, see the *Important properties for direct access and process image* section, which can be found in the various programming manuals.	

Task start sequence

When the StartupTask is completed, RUN mode is reached.

The following tasks are then started:

- SynchronousTasks
- TimerInterruptTasks
- BackgroundTask
- MotionTasks with startup attribute.

Note

The sequence in which these tasks are first started after RUN mode has been reached does not conform to the task priorities.

Configure execution system

Specifications for the configuring

When configuring the execution system, you assign the programs you want to execute in each task. In so doing, you specify the following:

- Priority with which the programs are executed
- Execution behavior (sequential, cyclical)
- Initialization behavior of local program variables

Assignment of a program to one or more tasks can only be performed after compilation and must occur before the program is loaded onto the target system.

When you have assigned a program to one or more tasks, you can establish the connection to the target system, download the program to the target system, and start it.

See also

Configure execution system (Page 1345)

Assigning programs to the execution levels/tasks (Page 1345)

Assigning programs to the tasks

In the *Execute sample program* section, you set up a task assignment using the sample program. The basic procedure is as follows:

1. Select the SIMOTION device in the project navigator, and select the **Target system > Configure execution system** menu command.
   The configuration window for the execution system of the SIMOTION device opens.
2. Select the task to be configured.
3. Select the **Program assignment** tab, and assign the desired programs to the task.
   See Section "Assigning programs to the execution levels / tasks (Page 1345)".
4. Select the Task configuration tab and make any additional settings, such as:
   - The local data stack size in the Area limit for dynamic data field, see SIMOTION ST Programming Manual
   - Error response to program error
   - Time watchdogs for cyclic tasks
   - Start behavior of MotionTasks

   See the detailed description for the relevant task in Section "Description of the user program tasks (Page 1314)".

Effect of the task execution behavior on the variable initialization

Time for the initialization of local program variables

The execution behavior of the task (sequential or cyclic) determines the initialization of local variables in the assigned programs. These descriptions also apply to instances of function blocks that were declared as local variables in the programs.

A summary of all variable types and their time of initialization can be found in the "Time of the variable initialization" chapter which can be found in the various Programming Manuals.

Table 4-79 Initialization of local program variables according to task execution behavior

Execution behavior	Tasks	Initialization of local program variables
Sequential (non-cyclic)	MotionTasks, UserInterruptTasks, SystemInterruptTasks, StartupTask, ShutdownTask.	After start, sequential tasks are executed once and then terminated. On each start, all local variables of the assigned programs are initialized. The time for data initialization is included in the task runtime. Behavior with compiler option "Only create program instance data once": • The local variables of the assigned programs will only be initialized once (during the download). See Download of changed sources in RUN (Page 1656) and Influence of the compiler on variable initialization (Page 1419). • Setting is possible so that the initialization of local variables is performed for every STOP - RUN transition, see Initialization of data during a STOP - RUN transition (Page 1664).
Cyclic	BackgroundTask, SynchronousTasks, TimerInterruptTasks	Cyclic tasks are automatically restarted upon completion, and the values of static local variables of the assigned programs (declared in VAR/ END_VAR) are retained. • Static variables are initialized once only during the transition from STOP to RUN. • Temporary variables (declared in VAR_TEMP / END_VAR) are initialized every time the task is started. This one-time initialization of static variables is not included in the task runtime. Behavior with compiler option "Only create program instance data once": • The local variables of the assigned programs will only be initialized once (during the download). See Download of changed sources in RUN (Page 1656) and Influence of the compiler on variable initialization (Page 1419). • Setting is possible so that the initialization of local variables is performed for every STOP - RUN transition, see Initialization of data during a STOP - RUN transition (Page 1664).

Assigning initial values to unit variables

Unit variables and global device variables are not initialized during the transition from STOP or STOPU mode to RUN mode (see the "Time of the variable initialization" chapter which can be found in the various Programming Manuals).

If you nevertheless want to assign initial values to these variables, use the StartupTask to do so. For information about the initialization for a STOP - RUN transition and the pragma BlockInit_OnDeviceRun (ST) or pragma lines (MCC and LAD/FBD), see also Initialization of data during a STOP - RUN transition (Page 1664).

Note

The task start sequence is not defined after the transition to RUN mode (refer to the **Task start sequence** section).

Correct initial value assignment is not ensured if a task other than the StartupTask is used.

Use multiple VAR_GLOBAL, VAR_GLOBAL RETAIN blocks

Description

You can create multiple VAR_GLOBAL, VAR_GLOBAL RETAIN blocks in the interface and implementation area of a UNIT (as of V4.1).

In the interface area and in the implementation area you can specify multiple declaration blocks in any sequence. Each of these blocks will be versioned separately. Changes within a block (whether direct or indirect via data type change) thus result in reinitialization of these blocks when reloading. Thus the (RETAIN) data retain works block-by-block.

New blocks can be added on the end and the changed sources can be reloaded in the RUN, without influencing the existing data blocks. If in one source section (statement applies separately for interface and implementation) a block is added in front of an existing block of the same type (RETAIN or not RETAIN) then all the subsequent blocks change and will be reinitialized at download. Thus download in the RUN after such a change is not possible.

Via the functions _saveUnitDataSet /_loadUnitDataSet and _exportUnitDataSet / _importUnitDataSet, the unit data block information can be saved. If when reading a data set one or multiple blocks are missing this can be detected on the return code, the remaining blocks however will be read. See also General information on saving data sets from the user program (Page 1522) and Data backup and data initialization from user program - functions and instructions (Page 1573).

Loss of retain data due to initialization

This data can be saved first in SCOUT via the "Save variables" function and read in again via the "Restore variables" function, see Online help.

Alternatively you can also use the _exportUnitDataSet (Page 1528) and _importUnitDataSet (Page 1531) system functions in the application.

Influence of the compiler on variable initialization

Compiler option "Only create program instance data once"

The compiler option can be set on each source and thus overwrites the global setting. It works regardless of the creation language and consequently it can also be used in LAD/FBD and MCC.

The compiler option controls how the instance data must be created in a PROGRAM contained in a source. Instance data of a PROGRAM are formed by the content of the VAR declaration block.

Generally, the following is valid: A program can be hooked into the execution system once and only once per task.

If "Only create program instance data once" is not set (DEFAULT), then the following behavior applies:

- The program cannot be called from a different LAD.
- The instance data of a program will be created separately for each task to which the program is assigned.
- The data initialization of the instance data is executed with the start of the TASK; (sequential task with task start, cyclic tasks for the STOP-RUN transition)
- Instance data of PROGRAMs in sequential tasks can be changed in RUN mode (if a loading is possible in principle).

Activation of the compiler option "Only create program instance data once" (also when using a program in different tasks) causes the following:

- Instance data of programs compiled in this manner will only be created once. The instance data is in the source, where the PROGRAM is declared.
- Each time a PROGRAM created in this manner is used, the same data is involved. This affects the assignment to (possibly multiple) tasks and the call in other PROGRAMS or function blocks.
- The data initialization is executed according to the rules of global data initialization (see download settings on the project) with the download of the source / of the code in which the program is declared. See also Download of changed sources in RUN (Page 1656).

NOTICE
Changed behavior for the data initialization when "Only create program instance data once" has been selected. With sequential tasks, data initialization is no longer performed with a task start or for cyclic tasks with the STOP - RUN transition, but generally only during a download. If necessary, the data must now be initialized by the application in the StartUp task or at the beginning of the programs in sequential tasks. An initialization during a STOP - RUN transition is possible • With a pragma BlockInit_OnDeviceRun (ST) or pragma line (MCC and LAD/FBD) • As of V4.2 using the dialog **Properties > Settings** via the context menu on the device. See Initialization of data during a STOP - RUN transition (Page 1664). For information on the influence of the compiler option in conjunction with the download in RUN, see Download of changed sources in RUN (Page 1656). If several tasks are assigned to one program, the same data is used as the basis for all of them.

Initialization of variables during a STOP - RUN transition

For information about the initialization for a STOP - RUN transition and pragma BlockInit_OnDeviceRun, see Initialization of data during a STOP - RUN transition (Page 1664).

Compiler option "Permit language extensions" (for non-IEC conformity)

The compiler option can be set on each source and thus overwrites the global setting. It works regardless of the creation language and consequently it can also be used in LAD/FBD and MCC.

The compiler option allows the following:

- Direct bit access, bit addressing for bit string variables (except data type BOOL)
- Read and write access to the input parameter (VAR INPUT) from function blocks outside the function block.
- Permissible call "program in program".

A program can be called within a different program, like a global FB instance within a program organization unit, such as calling "myprog" within a different program, within a different FB, but not within a function.

Global availability of the instance data is a prerequisite for "program in program". This can be satisfied either in that the PROGRAM does not have any instance data, or by using the compiler option "Only create program instance data once" when compiling the PROGRAM that will be called (see **Only create program instance data once**).

Note

If you do not set the compiler options, then the behavior remains unchanged compared to V4.0 (corresponds to DEFAULT).

Re-initialization of variable blocks

In VAR_GLOBAL, VAR_GLOBAL RETAIN blocks (interface and implementation sections of the source/unit):

- `BlockInit_OnChange := true`; causes (IMPLEMENTATION only) a reinitialization of the data to be executed with the values specified in the source when changes are made to the block structure for the download in RUN.
- With MCC and LAD/FBD, as of V4.2 the behavior can be set in the declaration tables using pragma lines.

This pragma is also applicable in VAR declarations of PROGRAMs. However, it will only work if the "Only create program instance data once" compiler option is selected.

See also Download of changed sources in RUN (Page 1656).

Marking HMI relevant data

As standard, the following applies for the HMI relevance of retentive unit variables (VAR_GLOBAL RETAIN) and non-retentive unit variables (VAR_GLOBAL):

- Unit variables declared in the interface section are available for operator control and monitoring (OCM).
  OCM addresses are generated for the variables contained therein. Changes have an effect on the HMI consistency.
- Unit variables declared in the implementation section are not exported to HMI devices. Changes have no effect on the HMI consistency.

Marking data as "non-HMI-relevant data"

You can mark individual declaration blocks in the interface section as non-HMI-relevant.

- In the SIMOTION ST programming language:
  Via the pragma `{ HMI_Export := FALSE; }` at the start of the declaration block (see the following example).
- In the SIMOTION MCC and SIMOTION LAD/FBD programming languages (as of V4.2):
  Via a pragma line in the declaration table.

OCM addresses are then no longer generated for the variables contained therein. For SIMOTION devices as of version V4.1, changes in this declaration block have no effect on the HMI consistency.

Example

```
VAR_GLOBAL
    { HMI_Export := FALSE; }
    x : INT;
    y : INT;
END_VAR
```

Marking data as "HMI-relevant data"

You can mark individual declaration blocks in the interface section as non-HMI-relevant.

- In the SIMOTION ST programming language:
  Via the pragma `{ HMI_Export := TRUE; }` at the start of the declaration block (see the following example).
- In the SIMOTION MCC and SIMOTION LAD/FBD programming languages (as of V4.2):
  Via a pragma line in the declaration table.

OCM addresses are then generated for the variables contained therein. Changes to this declaration block have an effect on the HMI consistency.

Example

```
VAR_GLOBAL
    { HMI_Export := TRUE; }
    x : INT;
    y : INT;
END_VAR
```

HMI address space and HMI consistency

The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.

Within the HMI address space, the variables are arranged in the following order:

- Retentive unit variables (VAR_GLOBAL RETAIN) of the interface section
- Retentive unit variables (VAR_GLOBAL RETAIN) of the implementation section
- Non-retentive unit variables (VAR_GLOBAL) of the interface section
- Non-retentive unit variables (VAR_GLOBAL) of the implementation section

Within these segments, the variables are arranged according to order of their declaration.

Note

For SIMOTION devices as of version V4.1:

Only HMI-relevant unit variables of the interface and implementation section are assigned to the HMI address space.

For SIMOTION devices up to version V4.0:

The following unit variables are assigned to the HMI address space:

- HMI-relevant **and non-HMI-relevant** unit variables of the interface section.
- HMI-relevant unit variables of the implementation section.

If variables of an HMI-relevant declaration block lie outside the HMI address space (64 KB), the compiler issues information 32050 that variables are not visible for the HMI. The first variable of the declaration block that exceeds the limit of the HMI address space is specified in this information.

Note

If variables are inserted in non-HMI-relevant declaration blocks, the HMI can still access SIMOTION variables even when the consistency check is activated.

A new compilation of the HMI project and a new transfer of the HMI project to an HMI device are then not necessary.

Exception for SIMOTION devices up to version V4.0:

HMI consistency is only retained when you enter or change variables in the following areas:

- In non-HMI-relevant declaration blocks of the **implementation section**
- In the **last declaration block** for non-retentive unit variables (VAR_GLOBAL) **of the interface section** when the following two conditions are satisfied:
  - This declaration block is marked as non-HMI-relevant.
  - No declaration blocks for non-retentive unit variables are marked as HMI-relevant in the implementation section.

  If the last condition is satisfied, you can add any number of declaration blocks for non-retentive unit variables (VAR_GLOBAL) at the **end of the interface section**, if they are marked as non-HMI-relevant.

See also HMI (Human Machine Interface) connection (Page 1611) and HMI variables in one unit (Page 1690).

For detailed information on the behavior of HMI-relevant data, see *SIMOTION ST Programming Manual*, Section Variables and HMI devices.

Task status

Querying and meaning of the task status

The task status can be queried using the function _getStateOfTaskId (Page 1437). This function requires the *TaskId* as an input parameter and returns a value of data type DWORD.

The table shows the possible combinations of task statuses in hex representation and as symbolic constants. Task status combinations are possible and are displayed as the sum of the hexadecimal values.

Table 4-80 Task statuses and their meaning

Symbolic constant	Hex notation	Symbolic constant
TASK_STATE_INVALID	16#0000	The task does not exist.
TASK_STATE_STOP_PENDING	16#0001	Task has received signal to stop; it is in a state between TASK_STATE_RUNNING and TASK_STATE_STOPPED. Actions may be performed until the task has stopped.
TASK_STATE_STOPPED	16#0002	Task stopped (e.g. using _resetTaskID (Page 1439) function), completed or not yet started.
TASK_STATE_RUNNING	16#0004	Task running, e.g.: • using _startTaskID (Page 1441) function, • As an active cyclic task
TASK_STATE_WAITING	16#0010	Task waiting due to _waitTime function or WAITFORCONDITION.
TASK_STATE_SUSPENDED	16#0020	Task suspended by function _suspendTaskid (Page 1442).
TASK_STATE_WAIT_NEXT_CYCLE	16#0040	TimerInterruptTask waiting for start trigger.
TASK_STATE_WAIT_NEXT_INTERRUPT	16#0080	SystemInterruptTask or UserInterruptTask is waiting for the triggering event to occur.
TASK_STATE_LOCKED	16#0100	Task locked by function *_disableScheduler*.

Combinations of task statuses

_getStateOfTaskId (Page 1437) often delivers a return value in the form of combinations found in the *Keywords for the declaration of static and temporary variables* table, depending on the task status named by the source file section. These combinations are generated by an OR logic operation. Frequent combinations are listed in the table.

Table 4-81 Frequently occurring task status combinations

Combination	Hex notation	Meaning
TASK_STATE_WAITING OR TASK_STATE_RUNNING	16#0014	Task running but currently waiting, for example, because of *_waitTime* or WAITFORCONDITION.
TASK_STATE_SUSPENDED OR TASK_STATE_RUNNING	16#0024	Task running but currently suspended by _suspendTaskid (Page 1442).
TASK_STATE_WAIT_NEXT_CYCLE OR TASK_STATE_RUNNING	16#0044	TimerInterruptTask running, but currently waiting for its start trigger.
TASK_STATE_WAIT_NEXT_CYCLE OR TASK_STATE_SUSPENDED OR TASK_STATE_RUNNING	16#0064	TimerInterruptTask running but currently waiting for its start trigger and suspended by _suspendTaskid (Page 1442).
TASK_STATE_WAIT_NEXT_INTERRUPT OR TASK_STATE_RUNNING	16#0084	SystemInterruptTask or UserInterruptTask running, but currently waiting for its triggering event.
TASK_STATE_WAIT_NEXT_INTERRUPT OR TASK_STATE_SUSPENDED OR TASK_STATE_RUNNING	16#00A4	SystemInterruptTask or UserInterruptTask running, but currently waiting for its triggering event, and is suspended by *_suspendTaskid*.
TASK_STATE_LOCKED OR TASK_STATE_RUNNING	16#0104	Task running but currently locked by *_disableScheduler*.

Other combinations are possible.

Example of the task status in use

In the following example, the task status of a MotionTask is queried to decide whether it can be started. The relevant bits of the return value are evaluated for this. If the result is positive, the MotionTask is started.

Table 4-82 Example of a query whether a MotionTask can be started

```
ret_dword := _getStateOfTaskId (id := _task.motionTask_1);
    IF (ret_dword AND
        (TASK_STATE_STOPPED OR TASK_STATE_STOP_PENDING)
        ) <> 0 THEN
        // MotionTask can be started.
        ret_dword := _restartTaskId (id := _task.motionTask_1);
    ELSE
        ; // MotionTask cannot be started.
    END_IF;
```

For an example with MCC: See Section "Parameter overview for the task state" in the MCC Programming Manual.

Waiting in MotionTask for a condition to be satisfied

Syntax of the condition of the EXPRESSION

The following applies for programming in ST. For MCC, definition of the wait condition is executed directly in the commands.

You can use the WAITFORCONDITION command in a MotionTask to wait for a condition to be satisfied (e.g. an event to occur). The MotionTask in which the statement is called is kept in TASK_STATE_WAITING status until the condition is satisfied.

When the VAR_IN_OUT in/out parameter is used, you can enable a time watchdog for WAITFORCONDITION.

This condition is formulated in the form of an EXPRESSION.

The expression is a special type of function declaration (for information on the syntax, see the **Expressions** section in the ST Programming Manual):

- The data type of the return value can be defined as BOOL and is not specified explicitly.
- The use of the VAR_IN_OUT in/out parameter and the VAR_INPUT input parameter is possible.

An expression can only be declared in the implementation section of the unit.

Optionally, local (temporary) variables can be declared in the declaration section. No other declarations (e.g. of input parameters or constants) are possible.

The following can be accessed in the statement section:

- Local variables for the expression
- Unit variables
- Global device variables, I/O variables, and the process image

An expression of the BOOL data type must be assigned to the expression identifier in the statement section of the expression (see **Expressions** section in the ST Programming Manual).

Note

The statement section of the expression cannot contain any function calls or loops.

Syntax of the WAITFORCONDITION statement

The following syntax is used to call the command for a waiting MotionTask subject to an expression:

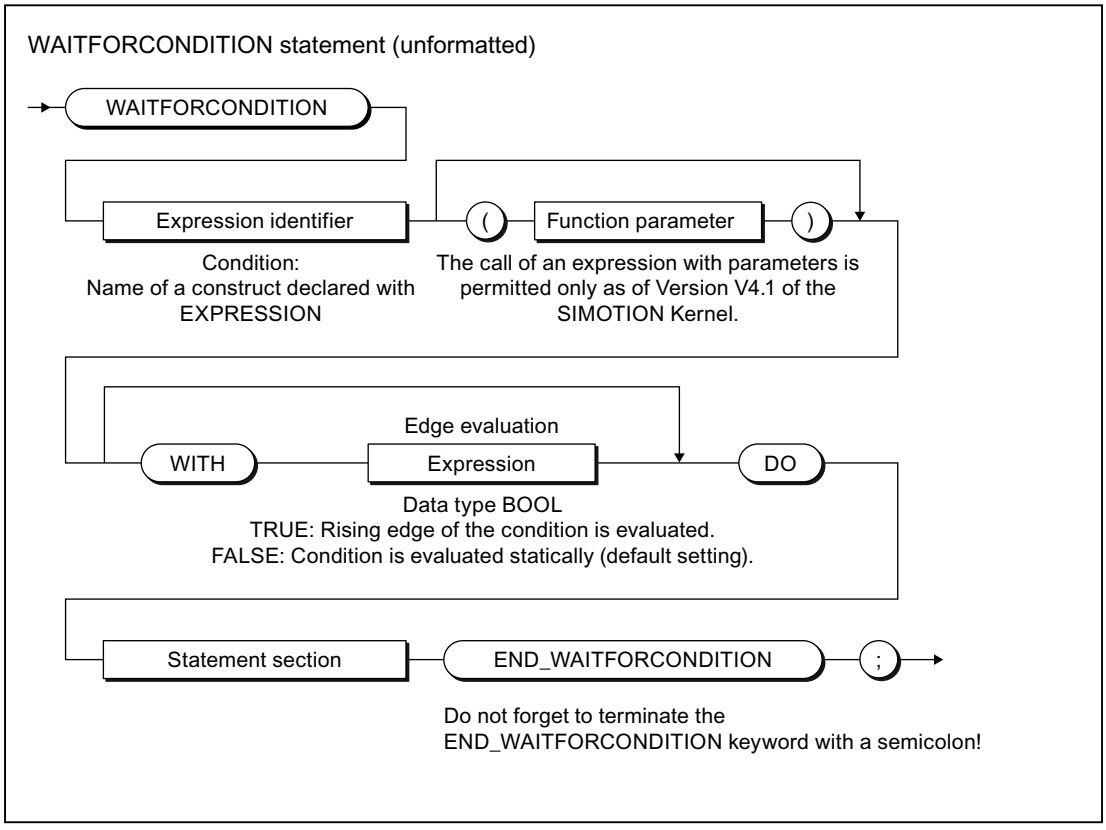


Figure 4-161 Syntax: WAITFORCONDITION statement

Expression identifier is a construct declared with EXPRESSION; its value defines (together with *WITH edge evaluation*, if necessary) whether the condition is considered as satisfied.

The *WITH edge evaluation* sequence is optional. *Edge evaluation* is an expression of data type BOOL; it determines how the value of *expression identifier* is interpreted:

- *Edge evaluation* = TRUE: The rising edge of *expression identifier* is interpreted; i.e. the condition is satisfied when the value of *expression identifier* **changes** from FALSE to TRUE.
- *Edge evaluation* = FALSE: The static value of *expression identifier* is evaluated; i.e. the condition is satisfied when the value of *expression identifier* **is** TRUE.

If is not specified, the default setting is FALSE, i.e. the static value of *expression identifier* is evaluated.

The statement section must contain at least one statement (empty statements also possible).

Effect of the WAITFORCONDITION statement

The WAITFORCONDITION statement sets the task that called it to the TASK_STATE_WAITING state until the condition (expression) is TRUE.

The priority of the task is increased if the expression is true. The priority is higher than for UserInterruptTasks and lower than for SystemInterruptTasks, i.e. the MotionTask has its turn before the other task in the round-robin execution level, as well as before the UserInterruptTasks and TimerInterruptTasks.

The increased task priority applies to all statements between the WAITFORCONDITION command and the END_WAITFORCONDITION command. The END_WAITFORCONDITION command ends the increased task priority.

The statement section must contain at least one empty statement.

Please note the following when using the WAITFORCONDITION command:

- The command is used to wait for an event in a MotionTask. If it is programmed within another task, it is ignored.
- There is no time watchdog in a MotionTask.
  Therefore, make sure that the condition does actually become true. Otherwise, the MotionTask would always be in the wait state and this would produce a timeout error.
- As of V4.1 a time watchdog for WAITFORCONDITION can be programmed in a Motion Task.
- The WAITFORCONDITION ... END_WAITFORCONDITION structure may not be nested.
- Within the round-robin level, the waiting task is the next to be executed when the event occurs, as long as it is not hindered by higher-priority tasks (such as alarms).
- The time slice of the active task in the round robin level is interrupted.
- For the behavior in the event of task interruption, see *_suspendTaskid*).
- The expression is checked in the IPO cycle clock with high priority.

See also

Task priorities (Page 1307)

MotionTasks (Page 1316)

Example of use of WAITFORCONDITION

The following example assumes that the *feeder* program is running in a MotionTask. The option **Activation after StartupTask** is selected for this MotionTask.

The assignment of programs to tasks is performed in SIMOTION SCOUT (see **Assigning programs to the execution levels / tasks**).

Table 4-83 Example for use of the WAITFORCONDITION condition

```
INTERFACE
 USEPACKAGE CAM;
 PROGRAM feeder;
END_INTERFACE

IMPLEMENTATION
    // Condition for WAITFORCONDITION in MotionTask_1
    EXPRESSION automaticExpr
        automaticExpr := IOfeedCam; // Digital input
    END_EXPRESSION
    // feeder (MotionTask_1)
    PROGRAM feeder
        VAR
            retVal : DINT;
    END_VAR ;
    retVal := _enableAxis(axis := realAxis,
        enableMode := ALL,
        servoCommandToActualMode := INACTIVE,
        nextCommand := WHEN_COMMAND_DONE,
        commandId := _getCommandId() );
        // Wait for automatic start
    WAITFORCONDITION automaticExpr WITH TRUE DO
        retVal := _pos(axis := realAxis,
            positioningMode := RELATIVE,
            position := 500,
            velocityType := DIRECT,
                                    velocity:=300,
            positiveAccelType := DIRECT,
            positiveAccel:= 400,
            negativeAccelType := DIRECT,
            negativeAccel := 400,
            velocityProfile:= TRAPEZOIDAL,
                                    mergeMode:=IMMEDIATELY,
                                    nextCommand:=WHEN_MOTION_DONE,
            commandId:= _getCommandId() );
            // Reduce priority after WAITFORCONDITION
    END_WAITFORCONDITION;
    retVal := _disableAxis(axis := realAxis,
    disableMode := ALL,
    servoCommandToActualMode := INACTIVE,
    nextCommand := WHEN_COMMAND_DONE,
    commandId := _getCommandId() );
    END_PROGRAM
END_IMPLEMENTATION
```

Example for time verification via FB

Example of using WAITFORCONDITION with time verification via FB (as of V4.1)

The following example shows time monitoring of the *expressions* (WAITFORCONDITION). The bonding of the expression at the call point to instance data of a calling function block inclusive time monitoring is presented alongside. This is a TON type reference variable. The call is executed within the expression with query of the output.

```
INTERFACE
    VAR_GLOBAL
        v1, v2 : INT;
        t1, t2 : TIME;
        Evaluation: INT;
    END_VAR
    PROGRAM Test_1;
END_INTERFACE
IMPLEMENTATION
    EXPRESSION mcccont1
        VAR_IN_OUT
            v : INT;
            t : TON;
        END_VAR
        t();
        mcccont1 := v > 100 or t.q ;
    END_EXPRESSION
    FUNCTION_BLOCK waitfb
        VAR_IN_OUT
            refpar1 : INT;
            refpar2 : TIME;
        END_VAR
        VAR_TEMP
            expr_timeout : TON;
        END_VAR
        expr_timeout(pt := refpar2, IN := TRUE);
        // Set monitoring time and
        //activate timer
        Evaluation: = 0;
        WAITFORCONDITION mcccont1(v := refpar1, t := expr_timeout ) DO
           Evaluation: = 100; //statement
            IF (expr_timeout.q) THEN
                // Error handling feasible, if Time-Out
                Evaluation: = 20005;
            END_IF;
        END_WAITFORCONDITION;
        expr_timeout( IN := FALSE);
        ;
    END_FUNCTION_BLOCK
    PROGRAM test_1
        VAR
            my_waitfb1 : waitfb;
        END_VAR
        my_waitfb1 (refpar1 :=v1, refpar2:=t1);
        //Wait until V1>100 and time
    END_PROGRAM
END_IMPLEMENTATION
```

The call of the waitfb type instance can then be executed at any point with different variables in each case. These global variables are then updated by cyclic tasks:

```
my_waitfb_1(refpar1 := v1, refpar2:=t1); //Wait until V1>100 and
time t1 not elapsed

my_waitfb_2(refpar1 := v2, refpar2:=t1);
```

Example for using WAITFORCONDITION with time monitoring directly in non-cyclic task / Motion Task

Description

Example of time monitoring of the expression (WAITFORCONDITION).

The call is executed directly in a MotionTask.

The time monitoring is type TON, with call within the expression and query of the output.

```
INTERFACE
    TYPE
        eDiagType : (TIMEOUT, COUNTER_REACHED, NOTHING);
    END_TYPE
    VAR_GLOBAL
        eDiag   : eDiagType := NOTHING;
        v1      : INT  := 0;
        t1      : TIME := T#0d_0h_0m_3s_0ms;
    END_VAR
    PROGRAM testExpression;
    PROGRAM increaseV1;
END_INTERFACE
IMPLEMENTATION
    EXPRESSION expr
        VAR_IN_OUT
            v : INT;
            t : TON;
        END_VAR
        t();
        expr := v > 500 OR t.q;
    END_EXPRESSION
                        //MotionTask
    PROGRAM testExpression
        VAR_TEMP
            expr_timeout : TON;
        END_VAR
                       // Set monitoring time, and activate timer
expr_timeout(pt := t1, IN := TRUE);
                      //Wait until v1 > 500 or time t1 has elapsed
        WAITFORCONDITION expr(v := v1, t := expr_timeout) DO
            IF (expr_timeout.q) THEN
                      // Error handling
                      // Time t1 has elapsed without v1 > 500 becoming true
                eDiag := TIMEOUT;
                … ;
            ELSE
                      // Good case,  v1 > 500
                eDiag := COUNTER_REACHED;
                 … ;
            END_IF;
        END_WAITFORCONDITION;
    END_PROGRAM
                       //BackgroundTask
    PROGRAM increaseV1
        v1 := v1 + 1;
    END_PROGRAM
END_IMPLEMENTATION
.....
```

You can update v1 in any cyclical tasks.

Making tasks wait a defined period

Place task in the wait state

You can place a task in the TASK_STATE_WAITING status for a specific period of time (see Task status (Page 1424)). To do so, use the *_waitTime* function. This places the calling task in the wait state for the specified period.

Note

A task that is in the wait state does not (or hardly) requires any CPU time. The only load on the target system is the periodic check to establish whether the wait time has expired. This check takes place in the IPO cycle clock.

The call of *_waitTime (timeValue := T#0ms)* in a MotionTask temporarily inactivates these and returns the program control to the scheduler. This is recommended, for example for longer loops if the program control will knowingly be transferred to the next round robin task (also possible in BackgroundTask).

Note

The function should be used in MotionTasks only; using it in cyclic tasks may lead to time monitoring errors!

- With SynchronousTasks: You can configure whether the time watchdog is suspended. Time monitoring is active by default.
  With IPOsynchronousTask, additionally take the following into account: UserInterruptTasks will no longer be started by their triggering event!
- With other cyclic tasks (BackgroundTask, TimerInterruptTasks): Time monitoring is always active.

In cyclic tasks, use the timer system function blocks (see *Timers* section) to implement waiting times.

Use an expression of data type TIME as an input parameter, the return value of data type DINT is always 0.

For more information on the function (syntax), refer to the description of the **_waitTime** function.

The following sample program puts the MotionTask to which it is assigned in the wait state for ten seconds:

Table 4-84 Example of _waitTime function

```
INTERFACE
    PROGRAM waitTime;
END_INTERFACE

IMPLEMENTATION
    PROGRAM waitTime
        VAR
            retVal :DINT;
        END_VAR;
        retVal := _waitTime(timeValue := T#10s);
    END_PROGRAM
END_IMPLEMENTATION
```

4.2.8.2 Task control commands

Overview of the task control commands

The commands listed in the table are available for task control.

Table 4-85 Task control commands in SIMOTION ST

Task control command	Meaning
_startTaskId (Page 1441)	Starts a MotionTask that is in the state TASK_STATE_STOPPED; its startup code is executed. Can only be applied to MotionTasks. Command must not directly follow *_resetTaskId*. Use *_restartTaskId* instead.
_resetTaskId (Page 1439)	Resets a MotionTask to TASK_STATE_STOPPED. Can only be applied to MotionTasks. *_startTaskId* must not directly follow this command. Use *_restartTaskId* instead.
_restartTaskId (Page 1439)	Resets a MotionTask to TASK_STATE_STOPPED and restarts it; the startup code for this MotionTask is executed. Can only be applied to MotionTasks.
_suspendTaskId (Page 1442)	The task in question will be suspended (set to TASK_STATE_SUSPENDED), for cyclic tasks time monitoring will be suspended. Cannot be used on SynchronousTasks, StartupTask, or ShutdownTask.
_resumeTaskId (Page 1440)	The relevant task, which is located in the TASK_STATE_SUSPENDED state, is resumed. For cyclic tasks, time monitoring is activated again. Cannot be used on SynchronousTasks, StartupTask, or ShutdownTask.

Task control command	Meaning
_getStateOfTaskId (Page 1437)	Queries the state of the relevant task.
_retriggerTaskIdControlTime (Page 1441)	The monitoring time is reset once for the relevant task.

The task is specified for these functions via a unique TaskId (Page 1436).

An example of a task control command can be found in Section "Example for using a task control command (Page 1437)".

See the SIMOTION MCC Programming Manual (Chapters "Incoming message" and "Outgoing message") for programming functions in MCC.

See also Task status (Page 1424).

Commands for controlling the scheduler

In addition, SIMOTION devices provide system functions for controlling the scheduler. These are listed in the table (for a description, see the List Manuals for SIMOTION devices):

Table 4-86 Commands for controlling the scheduler

Task control command	Meaning
_disableScheduler()	Prevents all user program tasks (except SynchronousTasks) from being loaded until *_enableScheduler* is called. It does not, however, affect system tasks. **Note**: The time monitoring for cyclic tasks is **not** suspended. **Note**: This command also prevents exchanging of SystemInterruptTasks and UserInterruptTasks!
_enableScheduler()	Cancels the effect of *_disableScheduler*.

TaskId

Tasks are specified via the associated TaskId.

You can obtain the TaskId for a task (data type StructTaskId) as follows:

- As _task.*name* variable (e.g. _task.motionTask_1). Explanation of terms:
  - _task
    Predefined name space for TaskId, see Section "Name spaces" in the SIMOTION ST Programming Manual. The identifier for the name space is separated by a period from the following identifier.
  - *name*
    The identifier of the task as specified in the execution system (e.g. motionTask_1).
- With the _getTaskId (*name*) function, see _getTaskId (Page 1444) function.
  name is the identifier of the task as assigned in the execution system.

You can use the _checkEqualTask function to check whether a TaskId belongs to a particular task (see _checkEqualTask (Page 1445) function).

Note

The _checkEqualTask and _getTaskId functions may only be called in the short form, i.e. with a complete listing of all parameter values, but without specification of the formal parameters.

The _checkEqualTask and _getTaskId functions and variables of the form _task.*name* must not be used in libraries.

Example for using a task control command

The following example requires that the following assignments have been made in SIMOTION SCOUT:

- The *myStart* program is assigned to the StartupTask.
- The *myMotion* program is assigned to MotionTask_1.

After controller startup (transition from STOP to RUN), the StartupTask runs automatically. The *myStart* program is executed in this task.

The *_startTaskId* task control command is used in the *myStart* program to start MotionTask_1. In MotionTask_1, the *myMotion* program is executed.

```
INTERFACE
    PROGRAM myStart;
    PROGRAM myMotion;
END_INTERFACE

IMPLEMENTATION
    PROGRAM myStart
        VAR
            RetDWord : DWORD;
        END_VAR

        RetDWord := _startTaskid (_task.motionTask_1);
        // Start MotionTask_1
    END_PROGRAM

    PROGRAM myMotion
        ;// Here, commands in the program, e.g. axis commands
    END_PROGRAM
END_IMPLEMENTATION
```

_getStateOfTaskId function

Description

This functions returns the state of the relevant task. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function can be applied to all tasks except StartupTask and ShutdownTask.

See also Section "Task status (Page 1424)". An example is also provided of how to decide if a MotionTask can be started, based on the task status, see Example for use of the task status (Page 1425).

Declaration

```
_getStateOfTaskId (
             { id     : StructTaskId // TaskId
             }
             ) : DWORD
```

Input parameter

id (optional)

Data type: StructTaskId

Default: TaskId of the current task in which the function is called.

TaskId of the task to be controlled, see TaskId (Page 1436).

Return value

Data type: DWORD

Status of the task is displayed as an OR operation on the following values, see also Task status (Page 1424):

16#0000	Task does not exist or Task ID is invalid TASK_STATE_INVALID
16#0001	Task in transition from running to stopped TASK_STATE_STOP_PENDING
16#0002	Task stopped TASK_STATE_STOPPED
16#0004	Task running TASK_STATE_RUNNING
16#0010	Task waiting TASK_STATE_WAITING
16#0020	Task suspended TASK_STATE_SUSPENDED
16#0040	TimerInterruptTask waiting for its start trigger TASK_STATE_WAIT_NEXT_CYCLE
16#0080	SystemInterruptTask or UserInterruptTask waiting for the occurrence of an initiating event TASK_STATE_WAIT_NEXT_INTERRUPT
16#0100	Task locked (by _disableScheduler) TASK_STATE_LOCKED

_resetTaskId function

Description

The function resets a MotionTask to TASK_STATE_STOPPED. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is only applicable to MotionTasks.

The _startTaskId (Page 1441) function must not directly follow this function. Use the _restartTaskId (Page 1439) function instead.

Declaration

```
_resetTaskId (
         id    : StructTaskId    // MotionTasks only
         ) : DWORD
```

Input parameter

id

Data type: StructTaskId

For the TaskId of the task to be controlled, see TaskId (Page 1436) – MotionTasks only.

Return value

Data type:	DWORD
0	No error
16#FFFF_FFFE	TaskId does not refer to a MotionTask
16#FFFF_FFFF	TaskId is invalid

_restartTaskId function

Description

The function resets a MotionTask to TASK_STATE_STOPPED and restarts it; its data is initialized. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is only applicable to MotionTasks.

To prevent this function from being used to stop a running MotionTask, its status can be queried and evaluated, see _getStateOfTaskId (Page 1437) function and the Example for use of the task status (Page 1425).

Declaration

```
_restartTaskId (
         id    : StructTaskId  // MotionTasks only
         ) : DWORD
```

Input parameter

id

Data type: StructTaskId

For the TaskId of the task to be controlled, see TaskId (Page 1436) – MotionTasks only.

Return value

Data type:	DWORD
0	No error
16#FFFF_FFFE	TaskId does not refer to a MotionTask
16#FFFF_FFFF	TaskId is invalid

_resumeTaskId function

Description

The function resumes a task that is in the TASK_STATE_SUSPENDED state. This state has been achieved using the _suspendTaskId (Page 1442) function. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is applicable to MotionTasks, BackgroundTask, TimerInterruptTasks, UserInterruptTasks, SystemInterruptTasks.

In the case of cyclic tasks (BackgroundTask, TimerInterruptTasks), the time watchdog for the task is enabled again.

Declaration

```
_resumeTaskId (
         id    : StructTaskId
         ) : DWORD
```

Input parameter

id

Data type: StructTaskId

TaskId of the task to be controlled, see TaskId (Page 1436).

Return value

Data type:	DWORD
0	No error
16#FFFF_FFFE	TaskId refers to an illegal task
16#FFFF_FFFF	TaskId is invalid

_retriggerTaskIdControlTime function

Description

The function resets the monitoring time once for the relevant task. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is applicable to MotionTasks, BackgroundTask, TimerInterruptTasks, UserInterruptTasks, SystemInterruptTasks.

Declaration

```
_retriggerTaskIdControlTime (
                    { id        : StructTaskId
                    }
                ) : DWORD
```

Input parameter

id (optional)

Data type:	StructTaskId
Default:	TaskId of the current task in which the function is called.

TaskId of the task to be controlled, see TaskId (Page 1436).

Return value

Data type:	DWORD
0	Function executed normally.
Not equal to 0	Function not executed normally.

_startTaskId function

Description

The function starts a MotionTask that is in the TASK_STATE_STOPPED state; its data is initialized. The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is only applicable to MotionTasks.

It has no effect on MotionTasks in the following states:

- TASK_STATE_RUNNING
- TASK_STATE_WAITING
- TASK_STATE_SUSPENDED

Before using the function, the state of the MotionTask can be queried and evaluated, see _getStateOfTaskId (Page 1437) function and Example for use of the task status (Page 1425).

It must not directly follow the _resetTaskId (Page 1439) function. Use the _restartTaskId (Page 1439) function instead.

Declaration

```
_startTaskId (
         id    : StructTaskId    // MotionTasks only
         ) : DWORD
```

Input parameter

id

Data type: StructTaskId

TaskId of the task to be controlled, see TaskId (Page 1436) – MotionTasks only.

Return value

Data type:	DWORD
0	No error
16#FFFF_FFFE	TaskId does not refer to a MotionTask
16#FFFF_FFFF	TaskId is invalid

_suspendTaskId function

Description

The function suspends the relevant task (sets it to TASK_STATE_SUSPENDED). The task is specified by means of a project-wide unique TaskId (see TaskId (Page 1436)).

The function is applicable to MotionTasks, BackgroundTask, TimerInterruptTasks, UserInterruptTasks, SystemInterruptTasks.

In the case of cyclic tasks (BackgroundTask, TimerInterruptTasks), the time watchdog for the task is stopped.

The task and its time monitoring are resumed using the _resumeTaskId (Page 1440) function.

Declaration

```
_suspendTaskId (
             id        : StructTaskId
         ) : DWORD
```

Input parameter

id

Data type: StructTaskId

TaskId of the task to be controlled, see TaskId (Page 1436).

Return value

Data type:	DWORD
0	No error
16#FFFF_FFFE	TaskId refers to an illegal task
16#FFFF_FFFF	TaskId is invalid

Interruption of the evaluation of a condition for WAITFORCONDITION

When the wait condition of a WAITFORCONDITION statement in a MotionTask is checked, the task status is not taken into account automatically.

You have programmed the wait condition for WAITFORCONDITION statement, e.g. wait for overtravel of an initiator, in an EXPRESSION. The associated MotionTask is now interrupted with _suspendTaskId before the wait condition is satisfied. Subsequently the condition is temporarily satisfied, e.g. through manual overtravel of an initiator. After _resumeTaskId, the MotionTask is executed as if the condition had been satisfied in the normal process.

However, you can avoid this by additionally querying the task status in the EXPRESSION.

```
// Condition for WAITFORCONDITION statement in MotionTask_1
EXPRESSION automaticExpr
    automaticExpr := IOfeedCam AND        // Digital input
        (task_status = (TASK_STATE_RUNNING OR TASK_STATE_WAITING));
END_EXPRESSION
```

_getTaskId function

Description

The function generates a project-wide unique TaskId (Page 1436) from the task name (as in the execution system). This TaskId can be assigned to a variable of data type StructTaskId and used as an input parameter in the following functions:

- Task control commands (Page 1435)
- Functions for runtime measurement of tasks (Page 1446)

Note

This function must not be used in libraries.

This function may only be called in the short form, i.e. with a complete listing of all parameter values, but without specification of the formal parameters.

Declaration

```
_getTaskId (  // Only the short form is permitted
         name    : Task_Name  // Name of the task
         ) : StructTaskId
```

Input parameter

name (only short form permitted)

Name of task as defined in the execution system.

Return value

Data type: StructTaskId

Return value contains the TaskId of the task.

_checkEqualTask function

Description

This function indicates whether a TaskId (Page 1436) is associated with a specified task.

Note

This function must not be used in libraries.

This function may only be called in the short form, i.e. with a complete listing of all parameter values, but without specification of the formal parameters.

Declaration

```
_checkEqualTask (  // Only the short form is permitted
        id     : StructTaskId,  // TaskId
        Name   : TaskName       // Name of the task
    ) : BOOL
```

Input parameter

id

(only short form permitted)

Data type: StructTaskId

TaskId to be checked, e.g. TSI#taskId.

Name

(only short form permitted)

Data type: TaskName

Name of a task, as defined in the execution system.

Return value

Data type: BOOL

The return value indicates whether the TaskId is associated with the specified task:

TRUE: TaskId is associated with the specified task.

FALSE TaskId is not associated with the specified task.

4.2.8.3 Functions for runtime measurement of tasks

Functions for runtime measurement of tasks - overview

The functions for runtime measurement are permissible for all tasks. However, the measurement is not supported by the IPOsynchronousTask, the ServosynchronousTask and the ShutdownTask. The functions for runtime measurement of tasks supply the runtime in ms.

The following runtimes can be measured:

- The maximum runtime of the task since the last STOP-RUN transition, see _getMaximalTaskIdRunTime (Page 1446) function
- The minimum runtime of the task since the last STOP-RUN transition, see _getMinimalTaskIdRunTime (Page 1447) function
- The runtime from the preceding task pass, see _getCurrentTaskIdRunTime (Page 1448) function
- The average task runtime from the last ten preceding passes, see _getAverageTaskIdRunTime (Page 1449) function

The task is specified for these functions via a unique TaskId.

Note

You can also use the Track Trace to determine the runtime of tasks. For detailed information regarding the Task Trace, see the Task Trace function manual.

Functions for accurate runtime determination

You can use the following system functions to measure the time since the system started with µs precision. Thus, you can measure the time precisely for sections within the application in order, for example, to optimize the application.

- Function _getInternalTimeStamp (Page 1451)
- _getTimeDifferenceOfInternalTimeStamps function (Page 1451)

See also

Task runtimes (Page 1358)

_getMaximalTaskIdRunTime function

Description

The _getMaximalTaskIdRunTime function provides the maximum runtime of the task since the last STOP-RUN transition, including all interruptions by higher-priority tasks. The task is specified by means of a project-wide unique task ID, see description in the section Overview of the task control commands (Page 1435).

The following functions do not interrupt the measurement:

- _suspendTaskId (Page 1442)
- _disableScheduler (see List Manuals for SIMOTION devices)

The determined runtime is a multiple of the position control cycle clock; for runtimes less than the position controller cycle, T#MIN (= T#0ms) is returned as measured value.

The function is permitted for all tasks. However, the measurement is not supported by the IPOsynchronousTask and the ShutdownTask. Measured value T#MIN (= T#0ms) is returned when called with these tasks.

Declaration

```
_getMaximalTaskIdRunTime (
                { id  : StructTaskId
                }
            ) : TIME
```

Input parameter

id (optional)

Data type: StructTaskId

Default: TaskId of the current task in which the function is called.

TaskId of the task whose runtime should be measured, see description in the section Overview of the task control commands (Page 1435).

Return value

Data type:	TIME
T#MIN (= T#0ms = T#1ms * UDINT#0)	Measurement is not supported or is not yet finished.
Greater than T#MIN and less than T#MAX	Greatest runtime that has occurred.
T#MAX (= T#49d_17h_2m_47s_295ms = T#1ms * UDINT#16#FFFF_FFFF)	TaskId is invalid

_getMinimalTaskIdRunTime function

Description

The _getMinimalTaskIdRunTime function provides the minimum runtime of the task since the last STOP-RUN transition, including all interruptions by higher-priority tasks. The task is specified by means of a project-wide unique task ID, see description in the section Overview of the task control commands (Page 1435).

The following functions do not interrupt the measurement:

- _suspendTaskId (Page 1442)
- _disableScheduler (see List Manuals for SIMOTION devices)

The determined runtime is a multiple of the position control cycle clock; for runtimes less than the position controller cycle, T#MIN (= T#0ms) is returned as measured value.

The function is permitted for all tasks. However, the measurement is not supported by the IPOsynchronousTask and the ShutdownTask. Measured value T#MIN (= T#0ms) is returned when called with these tasks.

Declaration

```
_getMinimalTaskIdRunTime (
                { id  : StructTaskId
                }
            ) : TIME
```

Input parameter

id (optional)

Data type: StructTaskId

Default: TaskId of the current task in which the function is called.

TaskId of the task whose runtime should be measured, see description in the section Overview of the task control commands (Page 1435).

Return value

Data type:	TIME
T#MIN (= T#0ms = T#1ms * UDINT#0)	Measurement is not supported or is not yet finished.
Greater than T#MIN and less than T#MAX	Minimum runtime that has occurred.
T#MAX (= T#49d_17h_2m_47s_295ms = T#1ms * UDINT#16#FFFF_FFFF)	TaskId is invalid

_getCurrentTaskIdRunTime function

Description

The _getCurrentTaskIdRunTime function provides the runtime from the preceding pass of the task, including all interruptions by higher-priority tasks. The task is specified by means of a project-wide unique task ID, see description in the section Overview of the task control commands (Page 1435).

The following functions do not interrupt the measurement:

- _suspendTaskId (Page 1442)
- _disableScheduler (see List Manuals for SIMOTION devices)

The determined runtime is a multiple of the position control cycle clock; for runtimes less than the position controller cycle, T#MIN (= T#0ms) is returned as measured value.

The function is permitted for all tasks. However, the measurement is not supported by the IPOsynchronousTask and the ShutdownTask. Measured value T#MIN (= T#0ms) is returned when called with these tasks.

Declaration

```
_getCurrentTaskIdRunTime (
                { id  : StructTaskId
                }
            ) : TIME
```

Input parameter

id (optional)

Data type: StructTaskId

Default: TaskId of the current task in which the function is called.

TaskId of the task whose runtime should be measured, see description in the section Overview of the task control commands (Page 1435).

Return value

Data type:	TIME
T#MIN (= T#0ms = T#1ms * UDINT#0)	Measurement is not supported or is not yet finished.
Greater than T#MIN and less than T#MAX	Runtime that occurred during the preceding pass.
T#MAX (= T#49d_17h_2m_47s_295ms = T#1ms * UDINT#16#FFFF_FFFF)	TaskId is invalid

Function _getAverageTaskIdRunTime

Description

This _getAverageTaskIdRunTime function provides an average runtime of the task from the last ten preceding cycles, including all interruptions by higher-priority tasks. The task is specified by means of a project-wide unique task ID, see description in the section Overview of the task control commands (Page 1435).

The following functions do not interrupt the measurement:

- _suspendTask (Page 1442)
- _disableScheduler (see List Manuals for SIMOTION devices)

The determined runtime is a multiple of the position control cycle clock; for runtimes less than the position controller cycle, T#MIN (= T#0ms) is returned as measured value.

The function is permitted for all tasks. However, the measurement is not supported by the IPOsynchronousTask and the ShutdownTask. Measured value T#MIN (= T#0ms) is returned when called with these tasks.

Declaration

```
_getAverageTaskIdRunTime (
                { id  : StructTaskId
                }
            ) : TIME
```

Input parameter

id (optional)

Data type	StructTaskId
Default:	TaskID of the task for which runtime is to be measured, as defined in the execution system.

TaskId of the task whose runtime should be measured, see description in the section Overview of the task control commands (Page 1435).

Return value

Data type:	TIME
T#MIN (= T#0ms = T#1ms * UDINT#0)	Measurement is not supported or is not yet finished.
Greater than T#MIN and less than T#MAX	Average of runtimes that have occurred.
T#MAX (= T#49d_17h_2m_47s_295ms = T#1ms * UDINT#16#FFFF_FFFF)	TaskId is invalid

Functions for the precise runtime measurement of tasks

Description

With the following system functions you can measure the time since system start with µs precision. Thus for sections within the application you can measure the time precisely, in order to optimize the application.

You can use the following functions:

- Function _getInternalTimeStamp (Page 1451)
- _getTimeDifferenceOfInternalTimeStamps function (Page 1451)

Function _getInternalTimeStamp

Description

The _getInternalTimeStamp function supplies a specific internal time stamp as UDINT. You can set a time stamp, e.g. at the start and at the end of a task. You can then use the two time stamps as parameters t1 and t2 of the _getTimeDifferenceOfInternalTimeStamp (Page 1451) function.

Note

The function is suitable for the measurement of short times (e.g. performance optimization of algorithms). In the case of extended measurements, there may be differences of the real time and the expired system cycle clocks. These differences can vary for different CPU types.

Declaration

```
_getInternalTimeStamp ()     : UDINT;
```

Input parameter

None

Return value

Data type: UDINT

Internal time stamp.

_getTimeDifferenceOfInternalTimeStamps function

Description

The _getTimeDifferenceOfInternalTimeStamps function supplies a time value from two internal time stamps (UDINT return value). Both time stamps must be generated via the _getInternalTimeStamp (Page 1451) function.

Note

When calling the function, pay attention to the order of the parameters t1 and t2 for the internal time stamp. The numeric values of the time stamp are not significant for this.

Declaration

```
_getTimeDifferenceOfInternalTimeStamps (
        t1  : UDINT,  // Internal time stamp 1
        t2  : UDINT   // Internal time stamp 2
    ): UDINT;         // Time difference in µs
```

Input parameter

t1

Data type: UDINT

Internal time stamp 1.

The start time of the interval to be measured, created with _getInternalTimeStamp function.

t2

Data type: UDINT

Internal time stamp 2.

The end time of the interval to be measured, created with _getInternalTimeStamp function.

Return value

Data type: UDINT

Time difference in µs.

Application

With this function you can measure, for example, the runtimes of code sections or communication times in an IPOsynchronous task.

4.2.8.4 Functions for message programming (AlarmS)

General information for the message programming

You can send messages configured in SIMOTION SCOUT of the AlarmS type (e.g. error messages) to logged-on display devices and check their status.

In particular, you can:

- Send a message that does not require acknowledgment, see _alarmSId (Page 1453) function.
- Send a message that requires acknowledgment, see _alarmSqId (Page 1456) function.
- Check the status of a message and its acknowledgment, see _alarmScId (Page 1459) function.
- List all pending alarms, see _getPendingAlarms (Page 1462) function (as of V4.1).

- Set messages to "outgoing", see _resetAlarmId (Page 1463) and _resetAllAlarmId (Page 1464) functions (as of V4.1).
- Acknowledge messages, see _acknowledgeAlarmSqId (Page 1466) and _acknowledgeAllAlarmSqId (Page 1467) functions (as of V4.4).

The message is specified for these functions using a unique AlarmId. For more information about the AlarmId, see the following section (Page 1453).

Examples of message programming are included in Programming messages. (Page 1563)

See the SIMOTION MCC Programming Manual (Chapters "Incoming message" and "Outgoing message") for programming functions in MCC.

AlarmId

Messages are specified via the associated AlarmId.

You can obtain the AlarmId (data type StructAlarmId) for a configured message name in the following way:

- As variable _alarm.*name*. Explanation of terms:
  - _alarm
    Predefined name space for AlarmId, see Section "Name spaces" in the SIMOTION ST Programming Manual. The identifier for the name space is separated by a period from the following identifier.
  - *name*
    Identifier of the message as configured in SIMOTION SCOUT.
- With the _getAlarmId (*name*) function, see _getAlarmId (Page 1461) function.
  name is the identifier of the message as configured in SIMOTION SCOUT.
  This function may only be called in the short form, i.e. with a complete listing of all parameter values, but without specification of the formal parameters.

Note

The _getAlarmId function and variables of the form _alarm.*name* must not be used in libraries.

_alarmSId function

Description

When a level change occurs in the triggering signal (Sig), the function generates a message that does not require acknowledgment and that is sent to all display devices logged on to receive such signals. The message to be generated is specified by means of a project-wide, unique AlarmId, see AlarmId (Page 1453).

As option, associated values can be appended:

- Independent of the SIMOTON kernel version:
  One associated value of the general data type ANY_NUM or ANY_BIT
- As of version V4.5 of the SIMOTION kernel:
  Several associated values are possible. They are passed in an ARRAY [0..11] OF BYTE.

A maximum of 40 messages can be processed simultaneously.

Declaration

```
_alarmSId (
         Sig    : BOOL
         Ev_Id  : StructAlarmId
         { sd   : ANY_NUM, ANY_BIT, ARRAY [0..11] OF BYTE
         }
         )  : DWORD
```

Input parameter

Sig

Data type: BOOL

The signal triggering the message is interpreted as follows:

If the signal represents a rising edge – relative to the last call with this AlarmId – an incoming message is generated. An incoming message is also generated if the signal state is TRUE on the first call with this message name.

If the signal represents a falling edge – relative to the last call with this AlarmId – an outgoing message is generated.

Ev_Id

Data type: StructAlarmId

AlarmId of the message to be generated, see AlarmId (Page 1453).

sd (optional)

Data type:	ANY_NUM, ANY_BIT (independent of the kernel version) ARRAY [0..11] OF BYTE (as of kernel V4.5)
Default:	0

Associated value:

Associated values must be specified when associated values have been defined during the message configuring in SIMOTION SCOUT.

- Independent of the SIMOTON kernel version:
  One associated value possible, all elementary data types are permissible.
  Variables or previously defined identifiers for constants belonging to one of the permitted data types are possible.
  Values can be entered, but it is not recommended. The format is determined only by the numerical value, which is particularly problematic for floating-point values. If a value is nonetheless specified directly, use the typed literal notation string for constants. A warning is generated if non-typed numerical values are entered.
- As of version V4.5 of the SIMOTION kernel:
  A maximum of 12 associated values of data type BYTE (or SINT, USINT) are possible; they are passed in an ARRAY [0..11] OF BYTE.
  Associated values with data types that comprise several bytes (e.g. WORD, DINT, REAL) are possible; they reduce the number of possible associated values. The user must ensure that their values are stored in the permissible array elements in accordance with the Big Endian byte order. The AnyType_to_BigByteArray (Page 1492) function can be used for this purpose.

During compilation, a check cannot be performed with the data types of the associated values configured in the message.

Return value

Data type: DWORD

The return value informs the user about the result of the call.

The specified values can be represented as an OR operation between the ALARMS_ERROR constants and the DSC_SVS_DEVICE_ALARMS_xxx constants.

16#0000	No error For an incoming message, an entry is made in the message list. For an outgoing message, an entry was deleted from the message list.
16#8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED

16#8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#8004	Double message, message rejected (call with incoming or outgoing message for the second time in succession). Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_CALL
16#8005	No display device registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#8006	Message name is already being processed in a lower-priority level. Does not occur for SIMOTION. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_IN_USE
16#8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#8008	Message name already assigned. Does not occur for SIMOTION. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_SFC_TYP
16#8009	Internal error ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#8010	Entry was rejected; message acknowledgment memory full. Only for _alarmSqId ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

Example

See Example of message generation (Page 1566), Example of messages with several associated values (as of kernel V4.5) (Page 1567).

_alarmSqId function

Description

When a level change occurs in the triggering signal (Sig), the function generates a message that requires acknowledgment and that is sent to all display devices logged on to receive such signals. The message to be generated is specified by means of a project-wide, unique AlarmId, see AlarmId (Page 1453).

As option, associated values can be appended.

- Independent of the SIMOTON kernel version:
  One associated value of the general data type ANY_NUM or >ANY_BIT
- As of version V4.5 of the SIMOTION kernel:
  Several associated values are possible. They are passed in an ARRAY [0..11] OF BYTE.

A maximum of 40 messages can be processed simultaneously.

Declaration

```
_alarmSqId (
         Sig    : BOOL
         Ev_Id  : StructAlarmId
         { sd   : ANY_NUM, ANY_BIT, ARRAY [0..11] OF BYTE
         }
         )  : DWORD
```

Input parameter

Sig

Data type: BOOL

The signal triggering the message is interpreted as follows:

If the signal represents a rising edge – relative to the last call with this message name – an incoming message is generated. An incoming message is also generated if the signal state is TRUE on the first call with this message name.

If the signal represents a falling edge – relative to the last call with this message name – an outgoing message is generated.

If no signal is generated with falling edge, an alarm of type SQ remains in the message buffer, even if it has been acknowledged using _acknowledgeAllAlarmSqId.

Ev_Id

Data type: StructAlarmId

AlarmId of the message to be generated, see AlarmId (Page 1453).

sd (optional)

Data type:	ANY_NUM, ANY_BIT (independent of the kernel version) ARRAY [0..11] OF BYTE (as of kernel V4.5)
Default:	0

Associated value

Associated values must be specified when associated values have been defined during message configuring in SIMOTION SCOUT.

- Independent of the SIMOTON kernel version:
  One associated value possible, all elementary data types are permissible.
  Variables or previously defined identifiers for constants belonging to one of the permitted data types are possible.
  Values can be entered, but it is not recommended. The format is determined only by the numerical value, which is particularly problematic for floating-point values. If a value is nonetheless specified directly, use the typed literal notation string for constants. A warning is generated if non-typed numerical values are entered.
- As of version V4.5 of the SIMOTION kernel:
  A maximum of 12 associated values of BYTE (or SINT, USINT) data type is possible; they are passed in an ARRAY [0..11] OF BYTE.
  Associated values with data types that comprise several bytes (e.g. WORD, DINT, REAL) are possible; they reduce the number of possible associated values. The user must ensure that their values are stored in the permissible array elements in accordance with the Big Endian byte order. The AnyType_to_BigByteArray (Page 1492) function can be used for this purpose.

During compilation, a check cannot be performed with the data type of the associated value configured in the message.

Return value

Data type: DWORD

The return value informs the user about the result of the call.

The specified values can be represented as an OR operation between the ALARMS_ERROR constants and the DSC_SVS_DEVICE_ALARMS_xxx constants.

16#0000	No error For an incoming message, an entry is made in the message list. For an outgoing message, an entry was deleted from the message list.
16#8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED

16#8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#8004	Double message, message rejected (call with incoming or outgoing message for the second time in succession). Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_CALL
16#8005	No display device registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#8006	Message name is already being processed in a lower-priority level. Does not occur for SIMOTION. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_IN_USE
16#8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#8008	Message name already assigned. Does not occur for SIMOTION. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_SFC_TYP
16#8009	Internal error ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#8010	Entry was rejected; message acknowledgment memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

Example

See Example of message generation (Page 1566), Example of messages with several associated values (as of kernel V4.5) (Page 1567).

_alarmScId function

Description

The function displays the status of a message and its acknowledgment status. The message is specified by means of a project-wide, unique AlarmId, see AlarmId (Page 1453).

Alarms that do not have to be acknowledged are always treated as acknowledged in "incoming" status.

Declaration

```
_alarmScId (
         Ev_Id    : StructAlarmId
         )    : DWORD
```

Input parameter

Ev_Id

Data type: StructAlarmId

AlarmId of the relevant message, see AlarmId (Page 1453).

Return value

Data type: DWORD

The return value informs the user about the result of the call and the status of a message. The constant values and symbolic constants below can be used with equal priority.

First check for errors:

16#8000	Filter for errors ALARMS_ERROR
16#8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID

Second check for message status:

16#0000	Outgoing message, not acknowledged (a).
16#0001	Incoming message, not acknowledged (b). ALARMS_STATE
16#0010	There is no message for this message name. Three options: 1. Message never triggered 2. Message triggered via _alarmSId, but also sent 3. Message triggered via _alarmSqId, sent and acknowledged at the display device
16#0101	Incoming message, acknowledged (c). ALARMS_STATE OR ALARMS_QSTATE

Messages (a) ... (b) imply _alarmSqId.

Message (c) implies _alarmSqId and _alarmSId.

Example

See Checking the error number and status of a message (filtering return values) (Page 1569).

_getAlarmId function

Description

The _getAlarmId function generates the AlarmId (Page 1453) of the message from a configured, application-specific message name. This AlarmId can be assigned to a variable of data type StructAlarmId and used as an input parameter in the following functions:

- _alarmSId (Page 1453) function
- _alarmSqId (Page 1456) function
- _alarmScId (Page 1459) function
- _resetAlarmId (Page 1463) function
- _acknowledgeAlarmSqId (Page 1466) function

Note

This function must not be used in libraries.

This function may only be called in the short form, i.e. with a complete listing of all parameter values, but without specification of the formal parameters.

Declaration

```
_getAlarmId (  // Only the short form is permitted
        Al_Name : Alarm_Name // Name of the message
        ) : StructAlarmId
```

Input parameter

Al_Name

This is a message name configured specifically for the application that is created when the message is configured in SIMOTION SCOUT.

Return value

Data type: StructAlarmId

The return value contains the AlarmId of the configured message, see AlarmId (Page 1453).

Function _getPendingAlarms

Description

The _getPendingAlarms provides information on the pending alarms. It provides:

- The number of entries in the message list (maximum of 40 entries).
- For each entry in the message list:
  - The AlarmId (Page 1453) of the message
  - The message type:
    Message that does not require acknowledgement generated with the _alarmSId (Page 1453), or
    Message that requires acknowledgement generated with the _alarmSqId (Page 1456).
  - The status of the message: "Incoming" or "Outgoing".

The function is available for SIMOTION devices as of version V4.1.

Declaration

```
_getPendingAlarms ()  :StructRetGetPendingAlarms
```

Input parameter

None

Return value

Data type: StructRetGetPendingAlarmState

The return value is a structure of data type *StructRetGetPendingAlarmState*.

It comprises the following:

- A numberOfPendingAlarms component: UINT
  This provides information on the number of entries in the message list.
- An alarm component: ARRAY [0..39] OF StructPendingAlarmState
  This provides information about the state of the individual messages in the list (maximum of 40 entries).
  The state of the individual messages is a structure of data type *StructPendingAlarmState*. It comprises the following:
  - An Id component: StructAlarmId
    This provides information about the project-wide, unique AlarmId (Page 1453) of the message
  - A _type component: EnumAlarmIdType
    This provides information about the message type (requires acknowledgement or not)
  - A state component: EnumAlarmIdState
    This provides information about the message state (incoming or outgoing)

Data types

```
TYPE
  EnumAlarmIdType  : (
    ALARM_S ,   // Message that does not require acknowledgement
    ALARM_SQ ); // Message that requires acknowledgement
  EnumAlarmIdState : (
    INCOMING_ALARM ,   // Incoming message
    OUTGOING_ALARM );  // Outgoing message
  StructPendingAlarmState    : STRUCT
    Id     : StructAlarmId;
    _type  : EnumAlarmIdType;
    state  : EnumAlarmIdState;
  END_STRUCT;
  StructRetGetPendingAlarmState : STRUCT
    numberOfPendingAlarms  : UINT;
    alarm  : ARRAY [0..39] OF StructPendingAlarmState;
  END_STRUCT;
END_TYPE
```

_resetAlarmId function

Description

The _resetAlarmId function sets a message to "outgoing". The message is specified by means of a project-wide, unique AlarmId, see AlarmId (Page 1453).

If this is a message that requires acknowledgment that has been generated with the _alarmSqId (Page 1456) function, the following applies:

- A message that has not been acknowledged must also be acknowledged, e.g.
  - Via the HMI or SIMOTION SCOUT.
  - With the _acknowledgeAlarmSqId (Page 1466) or _acknowledgeAllAlarmSqId (Page 1467) functions.
- An already acknowledged message is deleted.

A message that does not require acknowledgment that has been generated with the _alarmSId (Page 1453) function is deleted:

The function is available for SIMOTION devices as of version V4.1.

Declaration

```
_resetAlarmId (
        Id  : StructAlarmId
        ) : UDINT
```

Input parameter

Id

Data type: StructAlarmId

AlarmId of the message that is to be set to "outgoing", see AlarmId (Page 1453).

Return value

Data type	UDINT
The return value informs the user about the result of the call.	
16#0000_0000	No error Message could be acknowledged without errors.
16#0000_8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#0000_8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#0000_8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#0000_8005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#0000_8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#0000_8009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#0000_8010	Entry was rejected; message acknowledgment memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

_resetAllAlarmId function

Description

The _resetAllAlarmId function sets all messages to "outgoing".

For messages that require acknowledgment that have been generated with the _alarmSqId (Page 1456) function, the following applies:

- Messages that have not been acknowledged must also be acknowledged, e.g.
  - Via the HMI or SIMOTION SCOUT.
  - With the _acknowledgeAlarmSqId (Page 1466) or _acknowledgeAllAlarmSqId (Page 1467) functions.
- Already acknowledged messages are deleted.

Messages that do not require acknowledgment that have been generated with the _alarmSId (Page 1453) function are deleted:

The function is available for SIMOTION devices as of version V4.1.

Declaration

```
_resetAllAlarmId  () : UDINT
```

Input parameter

None

Return value

Data type	UDINT
The return value informs the user about the result of the call.	
16#0000_0000	No error Message could be acknowledged without errors.
16#0000_8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#0000_8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#0000_8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#00008005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED

16#0000_8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#0000_8009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#0000_8010	Entry was rejected; message acknowledgment memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

_acknowledgeAlarmSqId function

Description

The _acknowledgeAlarmSqId function acknowledges a message that requires acknowledgment that has been generated with the _alarmSqId (Page 1456) function. The message is specified by means of a project-wide, unique AlarmId, see AlarmId (Page 1453).

This enables the user, for example, to acknowledge a message via the signal of an I/O input and the call of the function.

The function is available for SIMOTION devices as of version V4.4.

Declaration

```
_acknowledgeAlarmSqId (
        Id  : StructAlarmId
        ) : DWORD
```

Input parameter

Id

Data type: StructAlarmId

AlarmId of the message that is to be acknowledged, see AlarmId (Page 1453).

Return value

Data type DWORD

The return value informs the user about the result of the call.

16#0000_0000	No error Message could be acknowledged without errors.
16#0000_8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID

16#0000_8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#0000_8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#0000_8005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#0000_8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#0000_8009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#0000_8010	Entry was rejected; message acknowledgment memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

_acknowledgeAllAlarmSqId function

Description

The _acknowledgeAllAlarmSqId function acknowledges all messages that require acknowledgment that have been generated with the _alarmSqId (Page 1456) function.

This enables the user, for example, to acknowledge all messages via the signal of an I/O input and the call of the function.

The function is available for SIMOTION devices as of version V4.4.

Declaration

```
_acknowledgeAllAlarmSqId  () : DWORD
```

Input parameter

None

Return value

Data type	DWORD
The return value informs the user about the result of the call.	
16#0000_0000	No error Message could be acknowledged without errors.
16#0000_8001	Message name not permitted. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
16#0000_8002	Loss of messages due to overflow (no more space in the message list). All 40 entries of the message list are occupied. Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
16#0000_8003	Loss of messages due to overflow (signal not yet sent, signal overflow). Send buffer for notification of the clients is still occupied by the last event. Entry is not made in the message list. Error may also occur if function calls come quickly one after the other with rising and falling edge. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
16#0000_8005	No display unit registered (message is entered into the list anyway). ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
16#0000_8007	No job has been started yet with this message name (first call with Sig = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) Entry is not made in the message list. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
16#0000_8009	Internal error. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
16#0000_8010	Entry was rejected; message acknowledgment memory full. ALARMS_ERROR OR DSC_SVS_DEVICE_ALARMS_NO_ENTRY

4.2.9 Programming of general standard functions

4.2.9.1 Programming of general standard functions - overview

SIMOTION provides a series of standard functions you can call in your sources to solve common tasks.

Note

Some of the following standard functions can only be called in short form, i.e. with a complete list of all parameter values, but without specifying the formal parameters:

- Result := function name (1st parameter value, 2nd parameter value)
  instead of
- Result := function name (formal parameter1 := 1st parameter value, etc.)

This is noted in the description of each function.

For information about the general data types (e.g. ANY_INT, ANY_BIT), see **Elementary data types**, General data types table in the ST programming manual.

Using the command library

The command library is a tab in the project navigator. It contains the available system functions, system function blocks, and operators.

You can use a drag-and-drop operation to move these elements from the command library:

- To the window of the ST editor
- To a network of the LAD/FBD editor
- To the system function call command of the MCC editor

Comparison of the system functions for SIMOTION and SIMATIC

You can find a comparison of the SIMATIC S7 and SIMOTION system functions on the Utilities & Applications CD under FAQs.

4.2.9.2 Numeric standard functions

Special features of a numeric function

Every numeric standard function has an input parameter. The result is always the return value. The general numeric, the logarithmic and the trigonometric standard functions each specify a group of numeric standard functions through the function name and the data type.

General numeric standard functions

General numeric standard functions are used for:

- Calculation of the absolute value of a variable
- Calculation of the square root of a variable
- Truncating a variable to its integer part

Table 4-87 General numeric standard functions

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
ABS	in	ANY_NUM	ANY_NUM[1]	Absolute value
SQRT	in	ANY_REAL	ANY_REAL[1]	Square root
TRUNC	in	ANY_REAL	ANY_INT	Truncation of value to integer part (0 direction)

[1] Data type of input parameter *in*.

The following table shows possible general numeric standard function calls and their results:

Table 4-88 General numeric standard function calls

Call	Result
`Result := ABS (-5);` `Result := ABS (in := -5);`	`5`
`Result := SQRT (81.0);` `Result := SQRT (in := 81.0);`	`9.0`
`Result := TRUNC (-3.141_59);` `Result := TRUNC (in := -3.141_59);`	`-3`

Logarithmic standard functions

Logarithmic standard functions are the functions for the calculation of an exponential value or a logarithm.

Table 4-89 Logarithmic standard functions

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
EXP	in	ANY_REAL	ANY_REAL[1]	e^x (e function)
EXPD	in	ANY_REAL	ANY_REAL[1]	10^x
EXPT	in1	ANY_REAL[2]	ANY_REAL[3]	Exponentiation (see also Operator ** in **Arithmetic expressions** in the ST programming manual)
	in2	ANY_REAL		

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
LN	in	ANY_REAL	ANY_REAL[1]	Natural logarithm
LOG	in	ANY_REAL	ANY_REAL[1]	Common logarithm

1 Data type of input parameter *in*.

2 The input parameter *in1* must be greater than zero.
Exceptions as of SIMOTION Kernel V4.1:
– If *in2* is an integer, *in1* can also be less than zero.
– If *in2* is positive, *in1* can also be equal to zero.
The following applies up to SIMOTION Kernel V4.0: If *in1* is equal to zero, an error message can be caught with ExecutionFaultTask.

3 Data type of input parameter *in1*.

The following table shows possible logarithmic standard function calls and their results:

Table 4-90 Logarithmic standard function calls

Call	Result
`Result := EXP (4.1);` `Result := EXP (in := 4.1);`	`60.3403 ...`
`Result := EXPD (3.0);` `Result := EXPD (in := 3.0);`	`1_000.0`

`Result := EXPT (2.0,4.0);`
`Result := EXPT (in1 := 2.0, in2 := 4.0);`
`Result := 2.0 ** 4.0;`
	`16.0`

`Result := LN (2.718_281);` `Result := LN (in := 2.718_281);`	`1.0`
`Result := LOG (245);` `Result := LOG (in := 245);`	`2.389_166`

Trigonometric standard functions

The trigonometric standard functions shown expect and calculate variables of angles in radians.

Table 4-91 Trigonometric standard functions

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
ACOS	in	ANY_REAL	ANY_REAL[1 2]	Arc cosine (main value)
ASIN	in	ANY_REAL	ANY_REAL[1 2]	Arc sine (main value)
ATAN	in	ANY_REAL	ANY_REAL[1 2]	Arc tangent (main value)
COS	in	ANY_REAL[2]	ANY_REAL[1]	Cosine (radian measure input)
SIN	in	ANY_REAL[2]	ANY_REAL[1]	Sine (radian measure input)
TAN	in	ANY_REAL[2]	ANY_REAL[1]	Tangent (radian measure input)

1 Data type of input parameter *in*.

2 Radian angle

The following table shows possible trigonometric standard function calls and their results:

Table 4-92 Trigonometric standard function calls

Call	Result

`PI := 3.141592; //PI is a variable!`
`Result := SIN (PI / 6);`
`Result := SIN (in := PI / 6);`
	`0.5`

`Result := ACOS (0.5);` `Result := ACOS (in := 0.5);`	`1.047_197 //equals PI/3`

Bit string standard functions

Each bit string standard function has two input parameters:

- in (data type ANY_BIT):

Bit string to be manipulated by bit shift operations

- n (data type USINT):
  - Number of places to be rotated for ROL and ROR
  - Number of places to be shifted for SHL and SHR

The result is always the return value. The following table shows the function names and the data types of the two input parameters and of the return value.

Table 4-93 Bit string standard functions

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
ROL	in n	ANY_BIT USINT	ANY_BIT[1]	The bit string in parameter *in* is **left-rotated** through the number of places specified by the content of parameter *n*.
ROR	in n	ANY_BIT USINT	ANY_BIT[1]	The bit string in parameter *in* is **right-rotated** through the number of places specified by the content of parameter *n*.
SHL	in n	ANY_BIT USINT	ANY_BIT[1]	The bit string contained in parameter *in* is **left-shifted** and replaced by 0, as specified by the content of parameter *n* [2].

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
SHR	in n	ANY_BIT USINT	ANY_BIT[1]	The bit string contained in parameter *in* is **right-shifted** and replaced by 0, as specified by the content of parameter *n* [2].
[1] Data type of input parameter *in* [2] Only the five least-significant bits of the parameter in are evaluated, for example: SHL (16#FFFF_FFFF,32) = 16#FFFF_FFFF SHR (16#FFFF_FFFF,32) = 16#FFFF_FFFF				

1 Data type of input parameter *in*.

2 Only the five least-significant bits of the parameter in are evaluated, for example:
SHL (16#FFFF_FFFF,32) = 16#FFFF_FFFF
SHR (16#FFFF_FFFF,32) = 16#FFFF_FFFF

Note

If numerical values are used as input parameters, the smallest possible data type is assumed (for example, BOOL if 1, BYTE if 2).

The following table shows possible bit string standard function calls and their results.

Table 4-94 Examples of bit string standard function calls

Call	Result
`Result := ROL (2#1101_0011, 5);` `// 1st parameter corresponds to 211 decimal`	`2#0111_1010` `(= 122 decimal)`
`Result := ROR (2#1101_0011, 2);` `// 1st parameter corresponds to 211 decimal`	`2#1111_0100` `(= 244 decimal)`
`Result := SHL (2#1101_0011, 3);` `// 1st parameter corresponds to 211 decimal`	`2#1001_1000` `(= 152 decimal)`
`Result := SHR (2#1101_0011, 2);` `// 1st parameter corresponds to 211 decimal`	`2#0011_0100` `(= 52 decimal)`
`Result := SHL (1, 3);` `// 1st parameter data type BOOL`	`2#0000_0000` `(= 0 decimal)`
`Result := SHL (2, 3);` `// 1st parameter data type BYTE`	`2#0001_0000` `(= 16 decimal)`

4.2.9.3 Access to bits in bit strings

_getBit function

Description

This function reads the bit string variable and returns the value of the specified bit.

Note

The access to bits of an I/O variable or system variable can be interrupted by other tasks. There is therefore no guarantee of consistency.

Access to I/O variables of the BOOL data type is consistent.

Declaration

```
FUNCTION _getBit (
        in  : ANY_BIT,  // Bit string variable
        n   : USINT     // Number of the bit
    ) : BOOL
```

Input parameter

in

Data type ANY_BIT

Bit string variable

n

Data type USINT

Number of the bit for which the value is to be output

Permissible values:		
	[0..7]	For BYTE
	[0..15]	For WORD
	[0..31]	For DWORD

When specifying impermissible values, the function supplies the return value FALSE.

Return value

Data type: BOOL

Value of the bit

Example

```
myBit := _getBit (
                in  := myBitString,
                n   := 5);
```

The user variable *myBit* contains bit 5 of the user variable *myBitString*.

Direct bit access

You can also access an individual bit of a bit string variable by specifying the number of the bit as constant separated by a point behind the variable. The number of the bit must be specified as a constant of the ANY_INT data type, see also SIMOTION ST Programming and Operating Manual.

To be able to use this option, you must activate the "Permit language extensions" compiler option (Page 1419), see also SIMOTION ST Programming and Operating Manual.

```
FUNCTION f : VOID
    VAR CONSTANT
        BIT_7 : INT := 7;
    END_VAR
    VAR
        dw : DWORD;
        b  : BOOL;
    END_VAR
    b := dw.BIT_7; // Access to bit number 7
    b := dw.3;     // Access to bit number 3
//  b := dw.33;    // Compilation error:
//  Bit offset not permitted for the data type!
END_FUNCTION
```

Note

The access to bits of an I/O variable or system variable can be interrupted by other tasks. There is therefore no guarantee of consistency.

_setBit function

Description

The function reads the specified bit string variable and returns its value. The specified bit is set to the prespecified Boolean value (TRUE or FALSE).

To change a bit value to a variable with this function, reassign the return value to this variable.

Example:

```
myVar := _setBit (in := myVar, n := 2, value := TRUE);
// Bit 2 is set to the value 1 in the myVar variable.
```

Note

A function call is generally executed in 3 steps:

- Read the input variables
- Function execution
- Result assignment

The execution sequence can be interrupted by other tasks. This also applies to the _setBit function.

When you use the function to modify bits of a variable, limit all function calls for the same variable to one task, e.g. BackgroundTask, IpoSynchronousTask, MotionTask_x.

The use for I/O variables (address list) is not recommended because interruptions by other accesses to the I/O variables are possible, e.g. access to lower-level BOOL variable declared in the address list, update of the process image. This can result in undesirable data corruption.

Declaration

```
FUNCTION _setBit (
        in      : ANY_BIT,    // Bit string variable
        n       : USINT,      // Number of the bit
      { value   : BOOL        // Value of the bit
        }
    ) : ANY_BIT
```

Input parameter

in

Data type: ANY_BIT

Bit string variable

n

Data type: USINT

Number of the bit for which the value is to be set.

Permissible values: [0..7] For BYTE
[0..15] For WORD
[0..31] For DWORD

If illegal values are specified (without an additional message), the unchanged value of the bit string variable is returned.

value (optional)

Default setting: TRUE

Value assigned to the bit to be set

Return value

Data type: ANY_BIT
Data type of input parameter *in*.
Value of the bit string variable with modified bit.

Example

```
myBitString := _setBit (
                    in    := myBitString,
                    n     := 5,
                    value := FALSE);
```

Bit 5 of the *myBitString* user variable is set to FALSE (logic 0).

Direct bit access

You can also access an individual bit of a bit string variable by specifying the number of the bit as constant separated by a point behind the variable. The number of the bit must be specified as a constant of the ANY_INT data type, see also SIMOTION ST Programming and Operating Manual.

To use this option, you must activate the "Permit language extensions" compiler option (Page 1419), see also SIMOTION ST Programming and Operating Manual.

```
FUNCTION f : VOID
    VAR CONSTANT
        BIT_7 : INT := 7;
    END_VAR
    VAR
        dw : DWORD;
        b  : BOOL;
    END_VAR
    b  := 1;
    dw.BIT_7 := b; // Write to bit number 7
    dw.3     := b; // Write to bit number 3
//  dw.33    := b; // Compilation error:
//  Bit offset not permitted for the data type!
END_FUNCTION
```

Note

When you set or reset multiple bits of a variable, limit all function calls for the same variable to one task, e.g. BackgroundTask, IpoSynchronousTask, MotionTask_x.

The use for I/O variables (address list) is not recommended because interruptions by other accesses to the I/O variables are possible, e.g. access to lower-level BOOL variable declared in the address list, update of the process image. This can result in undesirable data corruption.

_toggleBit function

Description

This function reads the specified bit string variable and returns its value, whereby the specified bit is inverted.

Change the specified bit in the variable by assigning the return value to the variable again.

Note

A function call is generally executed in 3 steps:

- Read the input variables
- Function execution
- Result assignment

The execution sequence can be interrupted by other tasks. This also applies to the _setBit function.

When you use the function to modify bits of a variable, limit all function calls for the same variable to one task, e.g. BackgroundTask, IpoSynchronousTask, MotionTask_x.

The use for I/O variables (address list) is not recommended because interruptions by other accesses to the I/O variables are possible, e.g. access to lower-level BOOL variable declared in the address list, update of the process image. This can result in undesirable data corruption.

Declaration

```
FUNCTION _toggleBit (
        in  : ANY_BIT,  // Bit string variable
        n   : USINT,    // Number of the bit
    ) : ANY_BIT
```

Input parameter

in

Data type: ANY_BIT

Bit string variable

n

Data type: USINT

Number of the bit whose value is to be inverted (set from TRUE to FALSE or from FALSE to TRUE).

Permissible values:		
	[0..7]	For BYTE
	[0..15]	For WORD
	[0..31]	For DWORD

If illegal values are specified (without an additional message), the unchanged value of the bit string variable is returned.

Return value

Data type: ANY_BIT
Data type of input parameter *in*.
Value of the bit string variable with inverted bit

Example

```
myBitString := _toggleBit (
                    in  := myBitString,
                    n   := 5);
```

Bit 5 of the *myBitString* user variable is inverted.

Direct bit access

You can also access an individual bit of a bit string variable by specifying the number of the bit as constant separated by a point behind the variable. The number of the bit must be specified as a constant of the ANY_INT data type, see also SIMOTION ST Programming and Operating Manual.

To be able to use this option, you must activate the "Permit language extensions" compiler option (Page 1419), see also SIMOTION ST Programming and Operating Manual.

```
FUNCTION f : VOID
    VAR CONSTANT
        BIT_7 : INT := 7;
    END_VAR
    VAR
        dw : DWORD;
        b  : BOOL;
    END_VAR
    b  := 1;
    dw.BIT_7 := NOT dw.BIT_7; // Write to bit number 7
    dw.3     := NOT dw.3;     // Write to bit number 3
//  dw.33    := NOT dw.33;    // Compilation error:
//  Bit offset not permitted for the data type
END_FUNCTION
```

Note

When you set or reset multiple bits of a variable, limit all function calls for the same variable to one task, e.g. BackgroundTask, IpoSynchronousTask, MotionTask_x.

The use for I/O variables (address list) is not recommended because interruptions by other accesses to the I/O variables are possible, e.g. access to lower-level BOOL variable declared in the address list, update of the process image. This can result in undesirable data corruption.

4.2.9.4 Bit operations on numeric data types

The following functions enable bit operations on numeric data types. Each bit of the return value is generated from the corresponding bits of the input parameters.

Table 4-95 Bit operators on numeric data types

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
_NOT	in	ANY_INT	ANY_INT[1]	Bit-by-bit negation
_AND	in1 in2	ANY_INT[32] ANY_INT[32]	ANY_INT[43]	Bit-by-bit conjunction (AND operation): A bit of the return value is only 1 when all the corresponding bits of the input parameters are 1, otherwise 0.
_OR	in1 in2	ANY_INT[32] ANY_INT[32]	ANY_INT[4]	Bit-by-bit disjunction (OR operation): A bit of the return value is only 1 when at least one of the corresponding bits of the input parameters is 1, otherwise 0.
_XOR	in1 in2	ANY_INT[32] ANY_INT[32]	ANY_INT[43]	Bit-by-bit exclusive disjunction (exclusive OR operation): A bit of the return value is only 1 when exactly one of the corresponding bits of the input parameters is 1, otherwise 0.

1 Data type of the input parameter.

2 It must be possible to convert the data types of in1 and in2 implicitly to a common general ANY_INT data type.

3 Smallest common data type to which the input parameters can be converted implicitly.

4.2.9.5 String processing

Functions for the string editing

The following functions enable the processing of variables of data type STRING.

Table 4-96 Functions for the string processing

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
CONCAT	in1 in2	STRING[254] STRING[254]	STRING[254]	Attaches string in2 to string in1[1]. Error flags: TSI#ERRNO := 1, if LEN (in1) + LEN (in2) > 254
DELETE	in l p	STRING[254] INT INT	STRING[254]	Deletes l character(s) from string in, starting at position p[2 4 6]. Error flags: TSI#ERRNO := 2, if l < 0, p < 1, p > LEN (in)

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
FIND	in1 in2	STRING[254] STRING[254]	INT	Returns the position at which string in2 begins in string in1. Result is 0 if string in2 is not contained in string in1. Error flags: None
INSERT	in1 in2 p	STRING[254] STRING[254] INT	STRING[254]	Inserts string in2 in string in1, starting at position p[1 2 5 7]. Error flags: TSI#ERRNO := 2, if p < 0, p > LEN (in1) TSI#ERRNO := 1, if LEN (in1) + LEN (in2) > 254
LEFT	in l	STRING[254] INT	STRING[254]	Returns the first l characters in string in[2 3]. Error flags: TSI#ERRNO := 2, if l < 0
LEN	in	STRING[254]	INT	Returns the number of characters in string in. Error flags: None
MID	in l p	STRING[254] INT INT	STRING[254]	Returns l characters from string in, starting at position p[2 3]. Error flags: TSI#ERRNO := 2, if l < 0, p < 1, p > LEN (in)
REPLACE	in1 in2 l p	STRING[254] STRING[254] INT INT	STRING[254]	Replaces l characters from string in1 with string in2, starting at position p[2 4 7]. Error flags: TSI#ERRNO := 2, if l < 0, p < 1, p > LEN (in1) TSI#ERRNO := 1, if LEN (in1) + LEN (in2) - l > 254
RIGHT	in l	STRING[254] INT	STRING[254]	Returns the last l characters in string in[2 3]. Error flags: TSI#ERRNO := 2, if l < 0

1 If LEN (in1) + LEN (in2) > 254: String is truncated.

2 If l < 0 or p < 0: Empty string is returned.

3 If l = 0 or p = 0: Empty string is returned.

4 If l = 0 or p = 0: String in or in1 remains unchanged.

5 If p = 0: String in1 is attached to string in2.

6 If p > LEN(in) : String in remains unchanged.

7 If p > LEN(in1): String in2 is attached to string in1.

Table 4-97 Examples for calls of the functions for the string processing

Call	Result
`A := CONCAT (in1 := 'ASTRING', in2 := '123');`	`'ASTRING123'.`

`A := DELETE (in1 := 'ASTRING', l := 2, p := 4);`
`A := DELETE (in1 := 'ASTRING', l := 2, p := 0);`
`A := DELETE (in1 := 'ASTRING', l := 2, p := 8);`
`A := DELETE (in1 := 'ASTRING', l := 0, p := 4);`
`A := DELETE (in1 := 'ASTRING', l := 10, p := 4);`
`A := DELETE (in1 := 'ASTRING', l := -1, p := 4);`
`A := DELETE (in1 := 'ASTRING', l := 2, p := -1);`
	`'ASTNG'.`
	`'ASTRING'.`
	`'ASTRING'.`
	`'ASTRING'.`
	`'AST'.`
	`''.`
	`''.`

`B := FIND (in1 := 'ASTRING', in2 := 'RI');` `B := FIND (in1 := 'ASTRING', in2 := 'RB');`	`4.` `0.`

`A := INSERT (in1 := 'ASTRING', in2 := '123', p := 1);`
`A := INSERT (in1 := 'ASTRING', in2 := '123', p := 0);`
`A := INSERT (in1 := 'ASTRING', in2 := '123', p := 10);`
`A := INSERT (in1 := 'ASTRING', in2 := '123', p :=-1);`
	`'A123STRING'.`
	`'123ASTRING'.`
	`'ASTRING123'.`
	`''.`

`A := LEFT (in := 'ASTRING', l := 3);`
`A := LEFT (in := 'ASTRING', l := 10);`
`A := LEFT (in := 'ASTRING', l := -1);`
	`'AST'.`
	`'ASTRING'.`
	`''.`

`B := LEN (in := 'ASTRING');`	`7.`

`A := MID (in := 'ASTRING', l :=3, p :=2 );`
`A := MID (in := 'ASTRING', l :=3, p :=6 );`
`A := MID (in := 'ASTRING', l :=3, p :=8 );`
`A := MID (in := 'ASTRING', l :=3, p :=0 );`
	`'STR'.`
	`'NG'.`
	`''.`
	`''.`

`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 4, p := 2);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 4, p := 1);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 0, p := 2);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 4, p := 0);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 2, p := 10);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 4, p := 5);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := 4, p := -1);`
`A := REPLACE (in1 := 'ASTRING', in2 := '123', l := -1, p : =2);`
	`'A123NG'.`
	`'123ING'.`
	`'ASTRING'.`
	`'ASTRING'.`
	`'ASTRING123'.`
	`'ASTRI123'.`
	`''.`
	`''.`

`A := RIGHT (in := 'ASTRING', l := 3);`
`A := RIGHT (in := 'ASTRING', l := 10);`
`A := RIGHT (in := 'ASTRING', l := -1);`
	`'ING'.`
	`'ASTRING'.`
	`''.`

For information about conversion functions for STRINGs, see Functions for the conversion of numeric data types to STRING data type (Page 1490).

Error analysis during the string editing

Description

Errors which occur during string functions are stored separately for each task in the Taskstartinfo. It is therefore implemented in the task context and can then be subsequently queried accordingly in the same task, e.g. BackgroundTask.

```
Variable:TSI#ERRNO : DINT
```

Value 0 indicates free of errors. For the string functions, the errors in P (position in the string) and L (number of characters) are stored separately from the violation of the maximum string length with different values.

- Value 0: for free of errors
- Value 1: for violation of the maximum string length
- Value 2: for p or l outside the value range

Note

TSI#ERRNO cannot be used to examine the string length (applies to DINT_TO_STRING, UDINT_TO_STRING, REAL_TO_STRING, and LREAL_TO_STRING).

In this case, the 0 error code will be set in TSI#ERRNO (string length exceeded).

You must, therefore, ensure that the string you enter is long enough or should check the length in the user program prior to conversion.

Resetting of the TSI#ERRNO value:

- The error remains in Taskstartinfo until you explicitly reset the content of TSI#ERRNO.
- The value of TSI#ERRNO is only rewritten in the case of an error when executing a new string function.
  A correctly executed string function does not overwrite any error ID present in TSI#ERRNO.
- The error ID is reset when the task is restarted.
  Cyclic execution of a task (e.g. IpoSynchronousTask, BackgroundTask) does not overwrite the error ID. The error ID is retained, as is the content of local variables.

Table 4-98 Example

```
VAR
    A : STRING [254];
END_VAR

A := DELETE (in := 'ASTRING', l:= -1, p := 4);
// Result is an empty string
IF (TSI#ERRNO = 2)THEN  // Error during string processing
    // l < 0, p < 1, p > LEN (in)
   A := 'ERROR';
END_IF;
```

4.2.9.6 Standard functions for data type conversion

Functions for the conversion of numeric data types and bit data types

Explicit conversion is always required if information could be lost, for example, if the value range is decreased or the accuracy is reduced, as is the case for conversion from LREAL to REAL.

The compiler outputs warnings when it detects conversions associated with loss of precision.

Note

The type conversion result may cause errors when the program is running, the error response set in the task configuration will then be triggered, see Execution errors in programs (Page 1256).

Special attention is required when converting DWORD to REAL. The bit string from DWORD is taken unchecked as the REAL value. You must make sure that the bit string in DWORD corresponds to the bit pattern of a normalized floating-point number in accordance with IEEE. To do this, you can use the _finite (Page 1503) and _isNaN (Page 1504) functions.

Otherwise, an error can be triggered (FPU exception (Page 1257)) as soon as the REAL value is first used for an arithmetic operation (for example, in the program or when monitoring in the symbol browser).

Explicit data type conversion is performed using standard functions which are listed in the following table.

- **Input parameter**
  Each function for the conversion of a data type has exactly one input parameter named *in*.
- **Return value**
  The result is always the return value of the function. The respective conversion rule is specified in the following table.
- **Names**
  As the data types of the input parameter and the return value come from the respective function name, they are not listed separately in the following table: For example, with the BOOL_TO_BYTE function, the data type of the input parameter is BOOL and the data type of the return value is BYTE.

Table 4-99 Functions for converting numeric data types and bit data types

Function name	Conversion rule	Implicitly possible
BOOL_TO_BYTE	Accept as least significant bit and fill the rest with 0.	Yes
BOOL_TO_DWORD	Accept as least significant bit and fill the rest with 0.	Yes
BOOL_TO_WORD	Accept as least significant bit and fill the rest with 0.	Yes
BOOL_VALUE_TO_DINT	Accept Boolean value as DINT value (0 or 1).	No
BOOL_VALUE_TO_INT	Accept Boolean value as INT value (0 or 1).	No
BOOL_VALUE_TO_LREAL	Accept Boolean value as LREAL value (0.0 or 1.0).	No
BOOL_VALUE_TO_REAL	Accept Boolean value as REAL value (0.0 or 1.0).	No
BOOL_VALUE_TO_SINT	Accept Boolean value as SINT value (0 or 1).	No
BOOL_VALUE_TO_UDINT	Accept Boolean value as UDINT value (0 or 1).	No
BOOL_VALUE_TO_UINT	Accept Boolean value as UINT value (0 or 1).	No
BOOL_VALUE_TO_USINT	Accept Boolean value as USINT value (0 or 1).	No
BYTE_TO_BOOL	Accept the least significant bit.	No
BYTE_TO_DINT	Accept bit string as least significant byte and fill the rest with 0.	No
BYTE_TO_DWORD	Accept bit string as least significant byte and fill the rest with 0.	Yes
BYTE_TO_INT	Accept bit string as least significant byte and fill the rest with 0.	No

Function name	Conversion rule	Implicitly possible
BYTE_TO_SINT	Accept bit string as SINT value.	No
BYTE_TO_UDINT	Accept bit string as least significant byte and fill the rest with 0.	No
BYTE_TO_UINT	Accept bit string as least significant byte and fill the rest with 0.	No
BYTE_TO_USINT	Accept bit string as USINT value.	No
BYTE_TO_WORD	Accept bit string as least significant byte and fill the rest with 0.	Yes
BYTE_VALUE_TO_LREAL	Interpret bit string as USINT value and accept this value.	No
BYTE_VALUE_TO_REAL	Interpret bit string as USINT value and accept this value.	No
DINT_TO_BYTE	Accept the least significant byte as bit string.	No
DINT_TO_DWORD	Accept bit string.	No
DINT_TO_INT	Accept the 2 least significant bytes as INT value.	No
DINT_TO_LREAL	Accept value.	Yes
DINT_TO_REAL	Accept value (accuracy may be lost).	No
DINT_TO_SINT	Accept the least significant byte as SINT value.	No
DINT_TO_UDINT	Accept value as bit string.	No
DINT_TO_UINT	Accept the 2 least significant bytes as UINT value.	No
DINT_TO_USINT	Accept the least significant byte as USINT value.	No
DINT_TO_WORD	Accept the 2 least significant bytes as bit string.	No
DINT_VALUE_TO_BOOL	FALSE (0), if DINT value = 0; else TRUE (1).	No
DWORD_TO_BOOL	Accept the least significant bit.	No
DWORD_TO_BYTE	Accept the least significant byte as bit string.	No
DWORD_TO_DINT	Accept bit string as DINT value.	No
DWORD_TO_INT	Accept the 2 least significant bytes as INT value.	No
DWORD_TO_REAL	Accept bit string as REAL value (validity check of the REAL number is not carried out!). Note the important information at the start of this section!	No
DWORD_TO_SINT	Accept the least significant byte as SINT value.	No
DWORD_TO_UDINT	Accept bit string as UDINT value.	No
DWORD_TO_UINT	Accept the 2 least significant bytes as UINT value.	No
DWORD_TO_USINT	Accept the least significant byte as USINT value.	No
DWORD_TO_WORD	Accept the 2 least significant bytes as bit string.	No
DWORD_VALUE_TO_LREAL	Interpret bit string as UDINT value and accept this value.	No
DWORD_VALUE_TO_REAL	Interpret bit string as UDINT value and accept this value (accuracy can be lost).	No
INT_TO_BYTE	Accept the least significant byte as bit string.	No
INT_TO_DWORD	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	No
INT_TO_DINT	Accept value.	Yes
INT_TO_LREAL	Accept value.	Yes
INT_TO_REAL	Accept value.	Yes
INT_TO_SINT	Accept the least significant byte as SINT value.	No
INT_TO_USINT	Accept the least significant byte as USINT value.	No
INT_TO_UDINT	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	No
INT_TO_UINT	Accept bit string as UINT value.	No

Function name	Conversion rule	Implicitly possible
INT_TO_WORD	Accept bit string.	No
INT_VALUE_TO_BOOL	FALSE (0), if INT value = 0; else TRUE (1).	No
LREAL_TO_DINT	Round off to next integer[1].	No
LREAL_TO_INT	Round off to next integer[1].	No
LREAL_TO_REAL	Accept value (accuracy may be lost).	No
LREAL_TO_SINT	Round off to next integer[1].	No
LREAL_TO_UDINT	Round off value to next integer[1].	No
LREAL_TO_UINT	Round off value to next integer[1].	No
LREAL_TO_USINT	Round off value to next integer[1].	No
LREAL_VALUE_TO_BOOL	FALSE (0), if LREAL value = 0.0; else TRUE (1).	No
LREAL_VALUE_TO_BYTE	Round off value to next integer[1] and accept value as bit string.	No
LREAL_VALUE_TO_DWORD	Round off value to next integer[1] and accept value as bit string (only for positive numbers[2])	No
LREAL_VALUE_TO_WORD	Round off value to next integer[1] and accept value as bit string.	No
REAL_TO_DINT	Round off to next integer[1].	No
REAL_TO_DWORD	Accept bit string.	No
REAL_TO_INT	Round off to next integer[1].	No
REAL_TO_LREAL	Accept value.	Yes
REAL_TO_SINT	Round off to next integer[1].	No
REAL_TO_UDINT	Round off value to next integer[1].	No
REAL_TO_UINT	Round off value to next integer[1].	No
REAL_TO_USINT	Round off value to next integer[1].	No
REAL_VALUE_TO_BOOL	FALSE (0), if REAL value = 0.0; else TRUE (1).	No
REAL_VALUE_TO_BYTE	Round off value to next integer[1] and accept value as bit string.	No
REAL_VALUE_TO_DWORD	Round off value to next integer[1] and accept value as bit string.	No
REAL_VALUE_TO_WORD	Round off value to next integer[1] and accept value as bit string.	No
SINT_TO_BYTE	Accept bit string.	No
SINT_TO_DINT	Accept value.	Yes
SINT_TO_DWORD	Accept bit string as least significant byte and fill the rest with 0.	No
SINT_TO_INT	Accept value.	Yes
SINT_TO_LREAL	Accept value.	No
SINT_TO_REAL	Accept value.	No
SINT_TO_UDINT	Accept bit string as least significant byte and fill the rest with 0.	No
SINT_TO_UINT	Accept bit string as least significant byte and fill the rest with 0.	No
SINT_TO_USINT	Accept bit string.	No
SINT_TO_WORD	Accept bit string as least significant byte and fill the rest with 0.	No
SINT_VALUE_TO_BOOL	FALSE (0), if SINT value = 0; else TRUE (1).	No
StructAlarmId_TO_DINT	Accept bit string.	No
UDINT_TO_BYTE	Accept the least significant byte as bit string.	No
UDINT_TO_DINT	Accept bit string.	No
UDINT_TO_DWORD	Accept bit string.	No
UDINT_TO_INT	Accept the 2 least significant bytes as INT value.	No

Function name	Conversion rule	Implicitly possible
UDINT_TO_LREAL	Accept value.	Yes
UDINT_TO_REAL	Accept value (accuracy may be lost).	No
UDINT_TO_SINT	Accept the least significant byte as SINT value.	No
UDINT_TO_UINT	Accept the 2 least significant bytes as UINT value.	No
UDINT_TO_USINT	Accept the least significant byte as USINT value.	No
UDINT_TO_WORD	Accept the 2 least significant bytes as bit string.	No
UDINT_VALUE_TO_BOOL	FALSE (0), if UDINT value = 0; else TRUE (1).	No
UINT_TO_BYTE	Accept the least significant byte as bit string.	No
UINT_TO_DINT	Accept value.	Yes
UINT_TO_DWORD	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	No
UINT_TO_INT	Accept bit string.	No
UINT_TO_LREAL	Accept value.	No
UINT_TO_REAL	Accept value.	Yes
UINT_TO_SINT	Accept the least significant byte as SINT value.	No
UINT_TO_UDINT	Accept value.	Yes
UINT_TO_USINT	Accept the least significant byte as USINT value.	No
UINT_TO_WORD	Accept bit string.	No
UINT_VALUE_TO_BOOL	FALSE (0), if UINT value = 0; else TRUE (1).	No
USINT_TO_BYTE	Accept bit string.	No
USINT_TO_DINT	Accept value.	No
USINT_TO_DWORD	Accept bit string as least significant byte and fill the rest with 0.	No
USINT_TO_INT	Accept value.	Yes
USINT_TO_LREAL	Accept value.	No
USINT_TO_REAL	Accept value.	No
USINT_TO_SINT	Accept bit string as SINT value.	No
USINT_TO_UDINT	Accept value.	Yes
USINT_TO_UINT	Accept value.	Yes
USINT_TO_WORD	Accept bit string as least significant byte and fill the rest with 0.	No
USINT_VALUE_TO_BOOL	FALSE (0), if USINT value = 0; else TRUE (1).	No
WORD_TO_BOOL	Accept the least significant bit.	No
WORD_TO_BYTE	Accept the least significant byte as bit string.	No
WORD_TO_DINT	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	No
WORD_TO_DWORD	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	Yes
WORD_TO_INT	Accept bit string as INT value.	No
WORD_TO_SINT	Accept the least significant byte as SINT value.	No
WORD_TO_UDINT	Accept bit string as least significant word (2 bytes) and fill the rest with 0.	No
WORD_TO_UINT	Accept bit string.	No
WORD_TO_USINT	Accept the least significant byte as USINT value.	No
WORD_VALUE_TO_LREAL	Interpret bit string as UINT value and accept this value.	No
WORD_VALUE_TO_REAL	Interpret bit string as UINT value and accept this value.	No

[1] If the distance to the two next integers is the same: Round off to next even integer.

[2] The LREAL_VALUE_TO_DWORD function behaves the same as the LREAL_TO_UDINT function. Consequently only positive numbers are converted. If a floating-point number with sign is to be converted to a bit string, the combination of LREAL_TO_DINT and DINT_TO_DWORD must be used.

Functions for converting date and time data types

The following standard functions are available for date and time data types.

For information about arithmetic expressions with date and time data types, see "Arithmetic expressions" in the SIMOTION ST Programming Manual.

Table 4-100 Standard functions for date and time

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
CONCAT_DATE_TOD	in1 in2	DATE TIME_OF_DAY	DATE_AND_TIME	Combine DATE and TIME_OF_DAY to DATE_AND_TIME (DT).
DATE_AND_TIME_TO_TIME_OF_DAY or DT_TO_TOD	in	DATE_AND_TIME	TIME_OF_DAY	Accept time of day.
DATE_AND_TIME_TO_DATE or DT_TO_DATE	in	DATE_AND_TIME	DATE	Accept date.
INT_TO_TIME	in	INT	TIME	Accept value as time (unit is ms) The function does not return any useful results for negative values.
REAL_TO_TIME	in	REAL	TIME	Round off to unsigned integer component and accept the value as indication of time (in ms).
TIME_TO_INT	in	TIME	INT	Accept time (unit is ms) as value.
TIME_TO_REAL	in	TIME	REAL	Accept time (unit is ms) as value.
TIME_TO_UDINT	in	TIME	UDINT	Accept time (unit is ms) as value.
UDINT_TO_TIME	in	UDINT	TIME	Accept value as time (unit is ms). The function does not return any useful results for negative values.

Functions for the conversion of enumeration data types

You can obtain the numeric value of an enumeration data type element with the ENUM_TO_DINT function. When calling the function, you specify the data type of the enumeration element by placing the identifier of the data type and the character # in front of the identifier of the enumeration element (*enum_type#enum_value*).

Table 4-101 Standard functions for date and time

Function name	Input parameter		Return value	Description
	Name	Data type	Data type	
ENUM_TO_DINT	in	any enumeration data type	DINT	Supplies the numeric value of the enumeration element

All user-defined enumeration data types organize the values in ascending order starting with 0.

Enumeration data types that are defined as system data types (e.g. of the SIMOTION devices or technology objects), deviate from this. You will find the values in the relevant List Manuals (reference lists).

Conversions between BYTE and STRING

Description

The implicit conversion from BYTE to STRING and from STRING to BYTE enables:

- A byte to be written to a string or
- A byte to be read from a string (ASCII format, 1 byte per character).

Note

Strings are interpreted as ARRAY[1...stringsize].

Conversion from BYTE to STRING

The conversion is performed through direct assignment of the byte to string element n:

```
mySTring[n] := myByte;
```

Rules:

1. If n > len(myString) and n < maxlen(myString), the length of the string is adjusted.
   All characters between myString[len(myString)] ... myString[n] are assigned the value "0".
2. If n > maxlen(myString), TSI#ERRNO is set to value 1 (maximum string length exceeded) and myString remains the same.

Conversion from STRING to BYTE

The conversion is performed through direct assignment of string element k to the byte:

```
myByte := myString[k];
```

Rule

1. If k > len(myString), TSI#ERRNO is set to value 2 (value exceeds the valid range) and myByte is assigned the value 0.

Applications

- Read out of internal variable and, if required, conversion from INT to STRING or DINT to STRING.
- Direct output of STRING on HMI, e.g. WIN CC Op.
- Conversion of STRING to ARRAY OF BYTE and thus output on HMI.

Functions for the conversion of numeric data types to STRING data type

Description

The following functions are used for the conversion of numbers to strings for the purpose of displaying numbers of the DINT/UDINT/REAL/LREAL data types.

Explicit data type conversion is performed using standard functions which are listed in the following table.

- Input parameter
  Each function for the conversion of a data type has exactly one input parameter named *in*.
- Return value
  The result is always the return value of the function.

Rules for the conversion from DINT/UDINT/REAL/LREAL to STRING

- Values are written left-justified in the STRING as decimal number or real number.
- If signs are present, they are written in front of the numerals.
- If the string length is not sufficient, the numeric sequence is truncated on the right (violation of the string length).
- With the conversion to STRING, the number representation is decimal.

Rules for the conversion from STRING to DINT/UDINT/REAL/LREAL

1. Leading white spaces are not taken into account; blanks and tabs are recognized as white spaces.
2. Conversion stops at the end of the string or at the first character that is not a numeral.
3. The function uses the "." as decimal character. If you use a "," as decimal character, the numbers after the comma are not taken into account.
   Example:
   - 1.1 -> STRING_TO_REAL -> 1.1
   - 1.1 -> STRING_TO_REAL -> 1.0

4. If the string does not contain a valid number or the value range is violated, TSI#ERRNO is set to value 3 (invalid number representation) and 0.0 (REAL/LREAL) output.
5. Leading zeros are omitted.

Function	Example	Description
DINT_TO_STRING	myString:=DINT_TO_STRING(myDint) myString := DINT_TO_STRING(mySint)	Observe rules
UDINT_TO_STRING	myString:=UDINT_TO_STRING(myUDint) myString:=UDINT_TO_STRING(myUsint)	
STRING_TO_DINT	myDin t:= STRING_TO_DINT(myString)	A valid number has the form [white space [sign][digits] Decimal numbers are required for conversion from STRING. Octal and hexadecimal notation is not supported.
STRING_TO_UDINT	myUDint:=STRING_TO_UDINT(myString)	
REAL_TO_STRING	myString := REAL_TO_STRING(myReal)	Observe rules
LREAL_TO_STRING	myString := LREAL_TO_STRING(myLReal)	
STRING_TO_REAL	myReal := STRING_TO_REAL(myString)	A valid number has the form [white space [sign][digits][digits][{ e I E } [sign]digits].
STRING_TO_LREAL	myLReal := STRING_TO_LREAL(myString)	

Application

- An HMI fetches texts from the file system (recipe memory) and loads the text via a sequence into the run-time system of SIMOTION (unit variable).
- The data is saved with _saveUnitDataSet or _exportUnitDataSet in the run-time system. Extension of the STRINGs with current SIMOTION data (e.g. actual position).
- The text is output via the serial interface (e.g. ET200).

4.2.9.7 Conversion between any data types and defined byte order

General

Variables of any data type (elementary data types, standard data types of technology packages and devices, and user-defined data types) can be converted to the defined big endian or little endian byte order, and vice versa, using the functions specified below:

- To byte arrays of the big endian or little endian type:
  AnyType_to_BigByteArray function, AnyType_to_LittleByteArray function (Page 1492)
- From byte arrays of the big endian or little endian type:
  BigByteArray_to_AnyType function, LittleByteArray_to_AnyType function (Page 1494)
- To the same data type with big endian or little endian byte order:
  TO_BIG_ENDIAN function, TO_LITTLE_ENDIAN function (Page 1496)
- From the same data type with big endian or little endian byte order:
  FROM_BIG_ENDIAN function, FROM_LITTLE_ENDIAN function (Page 1498)

For further information (e.g. for the ordering of the byte arrays, application example) see Converting between any data types and byte arrays (marshalling) (Page 1593).

These functions are commonly used to create defined transmission formats for data exchange between various devices (see also Communication functions (Page 1596)).

AnyType_to_BigByteArray function, AnyType_to_LittleByteArray function

Description

The functions convert a variable of any data type (elementary data types, user-defined data types, system data types of technology packages, and devices) to a byte array.

- For AnyType_to_BigByteArray:
  Big endian-type byte array (most significant byte at low memory address)
- For AnyType_to_LittleByteArray:
  Little endian-type byte array (least significant byte at low memory address)

The array index of the first element to be occupied in the array is an optional constant offset (default = 0). It must fall within the array limits.

When an ST source file is compiled, a check is made to determine whether the offset falls within the array limits and whether the variable can be displayed completely in the byte array (between the offset and the upper array limit).

Only those byte array elements that are covered by variables to be converted are occupied with values. Other elements of the byte array remain unchanged.

Note

Where possible, the function call and the processing of the result should be made in the same task.

If the function call and the processing of the result are made in different tasks, they must be synchronized using appropriate means (e.g. _testAndSetSemaphore (Page 1512), _releaseSemaphore (Page 1513)). Otherwise data consistency is not guaranteed and undefined values can result.

If the return value of the function should be assigned to a global variable that is evaluated in a higher priority task, the following procedure is recommended (also for simple data types):

1. Conversion to a temporary variable.
2. Assign the temporary variable to the global target variable.

Note

Each variable of data type BOOL (including variables that are components within a structured data type) occupies one byte in the byte array.

For the big endian and little endian byte order as well as the use of the functions, see Converting between any data types and byte arrays (marshalling) (Page 1593).

Declaration

```
FUNCTION AnyType_to_BigByteArray (
        anydata : ANY,         // Any data type
        { offset  : DINT         // Only constants permitted
        }
        ) : ARRAY [..] OF BYTE       // Big endian
FUNCTION AnyType_to_LittleByteArray (
        anydata : ANY,         // Any data type
        { offset  : DINT         // Only constants permitted
        }
        ) : ARRAY [..] OF BYTE  // Little endian
```

Input parameter

anydata

Data type: ANY

Variable of any data type.

The following data types are permitted:

- Elementary data types (ANY_ELEMENTARY)
- User-defined data types (UDT)
- System data types of technology packages and devices

The following data types are **not** permitted:

- Technology object data types
- Object-oriented interfaces
- Structures with overlapping memory areas (STRUCT OVERLAP)
  Components within these structures are, however, permitted.

offset (optional)

Data type: DINT

Default 0

Constant, specifies the first element to be assigned in the array.

Return value

Data type: ARRAY [..] OF BYTE

- For AnyType_to_BigByteArray:
  In big endian order (most significant byte at low memory address)
- For AnyType_to_LittleByteArray:
  In little endian order (least significant byte at low memory address)

BigByteArray_to_AnyType function, LittleByteArray_to_AnyType function

Description

The functions convert a byte array to a variable of any data type (elementary data types, user-defined data types, system data types of technology packages and devices, data types of technology objects).

- For BigByteArray_to_AnyType
  Big endian-type byte array (most significant byte at low memory address)
- For LittleByteArray_to_AnyType
  Little endian-type byte array (least significant byte at low memory address)

The array index of the first element to be evaluated in the array is an optional constant offset (default = 0). It must fall within the array limits.

When an ST source file is compiled, a check is made to determine whether the offset falls within the array limits and whether the byte array (between the offset and the upper array limit) completely covers the variable.

Note

Where possible, the function call and the processing of the result should be made in the same task.

If the function call and the processing of the result are made in different tasks, they must be synchronized using appropriate means (e.g. _testAndSetSemaphore (Page 1512), _releaseSemaphore (Page 1513)). Otherwise data consistency is not guaranteed and undefined values can result.

If the return value of the function should be assigned to a global variable that is evaluated in a higher priority task, the following procedure is recommended (also for simple data types):

1. Conversion to a temporary variable.
2. Assign the temporary variable to the global target variable.

Note

One byte from the byte array is assigned to each variable of data type BOOL (including variables that are components within a structured data type).

For the big endian and little endian byte order as well as the use of the functions, see Converting between any data types and byte arrays (marshalling) (Page 1593).

Declaration

```
FUNCTION BigByteArray_to_AnyType (
    byteArray : ARRAY [..] OF BYTE, // Big endian
    { offset  : DINT        // Only constants permitted
    }
    ) : ANY
```

```
FUNCTION LittleByteArray_to_AnyType (
    byteArray : ARRAY [..] OF BYTE, // Little endian
    { offset  : DINT        // Only constants permitted
    }
    ) : ANY
```

Input parameter

byteArray

Data type: ARRAY [..] OF BYTE

- For BigByteArray_to_AnyType
  In big endian order (most significant byte at low memory address)
- For LittleByteArray_to_AnyType
  In little endian order (least significant byte at low memory address)

offset (optional)

Data type: DINT

Default 0

Constant, specifies the first element to be assigned in the array.

Return value

Data type: ANY

Any data type.

The following data types are permitted:

- Elementary data types (ANY_ELEMENTARY)
- User-defined data types (UDT)
- System data types of technology packages and devices

The following data types are **not** permitted:

- Technology object data types
- Object-oriented interfaces
- Structures with overlapping memory areas (STRUCT OVERLAP)
  Components within these structures are, however, permitted.

Note

The marshalling function result may cause errors when the program is running, the error response set in the task configuration will then be triggered, see Execution errors in programs (Page 1256).

Proceed with caution when converting byte arrays to the general ANY_REAL data type or to structures that contain this data type. The bit string from the byte array is taken unchecked as the ANY_REAL value. You must make sure that the bit string of the byte array corresponds to the bit pattern of a normalized floating-point number in accordance with IEEE 754. To do this, you can use the _finite (Page 1503) and _isNaN (Page 1504) functions or the IS_VALID (Page 1505) function.

Otherwise, an error can be triggered (FPU exception (Page 1257)) as soon as the ANY_REAL value is first used for an arithmetic operation (for example, in the program or when monitoring in the symbol browser).

TO_BIG_ENDIAN function, TO_LITTLE_ENDIAN function

Description

The functions store a variable of any data type (elementary data types, user-defined data types, system data types of technology packages, and devices) in a variable of the same byte array data type. Whereby, elementary data types that comprise several bytes are stored as follows:

- For TO_BIG_ENDIAN:
  In big endian byte order (most significant byte at low memory address)
- With TO_LITTLE_ENDIAN:
  In little endian byte order (least significant byte at low memory address)

To use this function, you must activate the "Permit language extensions IEC61131-3rd Edition" compiler option, see SIMOTION ST Programming and Operating Manual.

Note

Where possible, the function call and the processing of the result should be made in the same task.

If the function call and the processing of the result are made in different tasks, they must be synchronized using appropriate means (e.g. _testAndSetSemaphore (Page 1512), _releaseSemaphore (Page 1513)). Otherwise data consistency is not guaranteed and undefined values can result.

If the return value of the function should be assigned to a global variable that is evaluated in a higher priority task, the following procedure is recommended (also for simple data types):

1. Conversion to a temporary variable.
2. Assign the temporary variable to the global target variable.

Note

Each variable of data type BOOL (including variables that are components within a structured data type) occupies one byte in the byte array.

For information about big endian and little endian byte ordering, see Converting between any data types and byte arrays (marshalling) (Page 1593).

Declaration

```
FUNCTION TO_BIG_ENDIAN (
        in  : ANY,        // Any data type
        )   : ANY       // Big endian
FUNCTION TO_LITTLE_ENDIAN (
        in  : ANY,        // Any data type
        )   : ANY       // Little endian
```

Input parameter

in

Data type: ANY

Variable of any data type.

The following data types are permitted:

- Elementary data types (ANY_ELEMENTARY)
- User-defined data types (UDT)
- System data types of technology packages and devices

The following data types are **not** permitted:

- Technology object data types
- Object-oriented interfaces
- Structures with overlapping memory areas (STRUCT OVERLAP)
  Components within these structures are, however, permitted.

Return value

Data type: ANY

The same data type as the *in* input parameter.

- For TO_BIG_ENDIAN:
  Elementary data types that comprise several bytes to the big endian byte order (most significant byte at low memory address)
- For TO_LITTLE_ENDIAN:
  Elementary data types that comprise several bytes to the little endian byte order (least significant byte at low memory address)

FROM_BIG_ENDIAN function, FROM_LITTLE_ENDIAN function

Description

The functions store a variable of any data type (elementary data types, user-defined data types, system data types of technology packages and devices, data types of technology objects) in a variable of the same data type. Whereby, elementary data types in the input parameter that comprise several bytes have the following byte order:

- For FROM_BIG_ENDIAN:
  The big endian byte order (most significant byte at low memory address)
- For FROM_LITTLE_ENDIAN:
  The little endian byte order (least significant byte at low memory address)

To use this function, you must activate the "Permit language extensions IEC61131-3rd Edition" compiler option, see SIMOTION ST Programming and Operating Manual.

Note

Where possible, the function call and the processing of the result should be made in the same task.

If the function call and the processing of the result are made in different tasks, they must be synchronized using appropriate means (e.g. _testAndSetSemaphore (Page 1512), _releaseSemaphore (Page 1513)). Otherwise data consistency is not guaranteed and undefined values can result.

If the return value of the function should be assigned to a global variable that is evaluated in a higher priority task, the following procedure is recommended (also for simple data types):

1. Conversion to a temporary variable.
2. Assign the temporary variable to the global target variable.

Note

One byte from the byte array is assigned to each variable of data type BOOL (including variables that are components within a structured data type).

For information about big endian and little endian byte ordering, see Converting between any data types and byte arrays (marshalling) (Page 1593).

Declaration

```
FUNCTION FROM_BIG_ENDIAN (
    in : ANY // Any data type, big endian
    )  : ANY
FUNCTION FROM_LITTLE_ENDIAN (
    in : ANY // Any data type, little endian
    )  : ANY
```

Input parameter

in

Data type: ANY

Any data type.

- For FROM_BIG_ENDIAN
  Elementary data types that comprise several bytes to the big endian byte order (most significant byte at low memory address)
- For FROM_LITTLE_ENDIAN
  Elementary data types that comprise several bytes to the little endian byte order (least significant byte at low memory address)

The following data types are permitted:

- Elementary data types (ANY_ELEMENTARY)
- User-defined data types (UDT)
- System data types of technology packages and devices

The following data types are **not** permitted:

- Technology object data types
- Object-oriented interfaces
- Structures with overlapping memory areas (STRUCT OVERLAP)
  Components within these structures are, however, permitted.

Return value

Data type: ANY

The same data type as the *in* input parameter.

4.2.9.8 Combining bit-string data types

General information for combining bit-string data types

The following functions enable several bit-string data-type variables to be combined into a higher-level data-type variable.

The inverse functions are implemented as function blocks, see Separating bit-string data types (Page 1630).

_BYTE_FROM_8BOOL function

Description

This function combines eight variables of data type BOOL into one variable of data type BYTE.

Declaration

```
FUNCTION _BYTE_FROM_8BOOL (
       { bit0,          // Least significant bit
         bit1, bit2, bit3, bit4, bit5, bit6,
         bit7 : BOOL   // Most significant bit
         }
    ) : BYTE
```

Input parameter

bit0 (optional)
...
bit7 (optional)

Data type: BOOL
Default FALSE

Up to eight variables of data type BOOL which are to be combined into a variable of data type BYTE

bit0: Least significant bit
...
bit7: Most significant bit

Return value

Data type: BYTE
Byte formed by combining input parameters.

_WORD_FROM_2BYTE function

Description

This function combines two variables of data type BYTE into one variable of data type WORD.

Declaration

```
FUNCTION _WORD_FROM_2BYTE (
       { byte0,           // Less significant byte
         byte1   : BYTE   // More significant byte
         }
    ) : WORD
```

Input parameter

byte0 (optional)
byte1 (optional)

Data type: BYTE
Default BYTE#0

Up to two variables of data type BYTE, which are to be combined into a variable of data type WORD

byte0: Less significant byte
byte1: More significant byte

Return value

Data type: WORD
Word formed by combining input parameters.

_DWORD_FROM_2WORD function

Description

This function combines two variables of data type WORD into one variable of data type DWORD.

Declaration

```
FUNCTION _DWORD_FROM_2WORD (
       { word0,          // Less significant word
         word1 : WORD    // More significant word
         }
    ) : DWORD
```

Input parameter

word0 (optional)
word1 (optional)

Data type: WORD
Default WORD#0

Up to two variables of data type WORD, which are to be combined into a variable of data type DWORD

word0: Less significant word
word1: More significant word

Return value

Data type: DWORD
Double word formed by combining input parameters.

_DWORD_FROM_4BYTE function

Description

This function combines four variables of data type BYTE to one variable of data type DWORD.

Declaration

```
FUNCTION _DWORD_FROM_4BYTE (
       { byte0          // Least significant byte
         byte1, byte2,
         byte3 : BYTE   // Most significant byte
         }
    ) : DWORD
```

Input parameter

byte0 (optional)
byte1 (optional)
byte2 (optional)
byte3 (optional)

Data type: BYTE
Default BYTE#0
Up to four variables of data type BYTE, which are to be combined into a variable of data type DWORD
byte0: Least significant byte
...
byte3: Most significant byte

Return value

Data type: DWORD
Double word formed by combining input parameters.

4.2.9.9 Conversion of technology object data types

AnyObject_to_Object function

Description

The function converts variables of a hierarchical TO data type (driveAxis, posAxis, followingAxis) or of the general ANYOBJECT type to a compatible TO data type.

You will find examples and additional information in Conversion of TO data types (Page 1242).

Declaration

```
FUNCTION AnyObject_to_Object (
        in    : ANYOBJECT
    ) : ANYOBJECT
```

Input parameter

in

Data type: ANYOBJECT

Variable of a TO data type (or ANYOBJECT) or a TO instance

Return value

Data type: ANYOBJECT

Value is TO#NIL, if type conversion is not possible

4.2.9.10 Functions for verification of floating-point numbers

_finite function

Description

The function checks whether the input parameter complies with the bit pattern for infinite in accordance with IEEE 754.

In combination with the _isNaN (Page 1504) function, it is used in particular to check whether bit strings converted to floating-point numbers correspond to the bit pattern of a normalized floating-point number in accordance with IEEE 754.

This prevents the error response specified during task configuration from being triggered (see "Execution errors in programs" (Page 1256)) as soon as an invalid floating-point number is used

for an arithmetic operation for the first time (for example in the program or when monitoring in the symbol browser).

Declaration

```
_finite (
        in : ANY_REAL
     ) : BOOL
```

Input parameter

in

Data type: ANY_REAL
Variable to be checked

Return value

Data type:	BOOL
FALSE	Bit pattern for infinite in accordance with IEEE 754
TRUE	No bit pattern for infinite in accordance with IEEE 754, i.e. valid floating-point number within the value range or invalid bit pattern (NaN Not a Number)

Example

```
var_real := DWORD_TO_REAL (var_dword);
IF NOT _finite (var_real) OR _isNaN (var_real) THEN
    ; // Error handling
ELSE
    var_real := SQRT (var_real);
END_IF;
```

_isNaN function

Description

The function verifies whether the input parameter corresponds to an invalid bit pattern of a floating-point number in accordance with IEEE 754 (is Not a Number NaN).

In combination with the _finite (Page 1503) function, it is used in particular to check if bit strings converted to floating-point numbers correspond to the bit pattern of a normalized floating-point number in accordance with IEEE 754.

This prevents the error response specified during task configuration from being triggered (see "Execution errors in programs" (Page 1256)) as soon as an invalid floating-point number is used

for an arithmetic operation for the first time (for example in the program or when monitoring in the symbol browser).

Declaration

```
_isNaN (
         in : ANY_REAL
      ) : BOOL
```

Input parameter

in

Data type: ANY_REAL
Variable to be checked

Return value

Data type:	BOOL
FALSE	Valid bit pattern or bit pattern for infinite in accordance with IEEE 754
TRUE	Invalid bit pattern in accordance with IEEE 754 (NaN Not a Number)

Example

```
var_real := DWORD_TO_REAL (var_dword);
IF NOT _finite (var_real) OR _isNaN (var_real) THEN
    ; // Error handling
ELSE
    var_real := SQRT (var_real);
END_IF;
```

IS_VALID function

Description

The function checks whether the input parameter complies with the bit pattern of a normalized floating-point number in accordance with IEEE 754.

This prevents the error response specified during task configuration from being triggered (see "Execution errors in programs" (Page 1256)) as soon as an invalid floating-point number is used for an arithmetic operation for the first time (for example in the program or when monitoring in the symbol browser).

To use this function, you must activate the "Permit language extensions IEC61131-3rd Edition" compiler option, see SIMOTION ST Programming and Operating Manual.

Declaration

```
IS_VALID (
        in : ANY_REAL
     ) : BOOL
```

Input parameter

in

Data type: ANY_REAL

Variable to be checked

Return value

Data type:	BOOL
FALSE	Bit pattern for infinite in accordance with IEEE 754 or invalid bit pattern (NaN Not a Number)
TRUE	Valid floating-point number within the value range

Example

```
var_real := DWORD_TO_REAL (var_dword);
IF IS_VALID (var_real) THEN
    var_real := SQRT (var_real);
ELSE
    ; // Error handling
END_IF;
```

4.2.9.11 Functions for selection

SEL function

Description

Binary selection

The return value is one of the input parameters in0 or in1, depending on the value of input parameter g.

Declaration

```
FUNCTION SEL (
        g    : BOOL,
        in0  : ANY,
        in1  : ANY
        ) : ANY
```

Input parameter

g

Data type: BOOL

Selection of the input parameter in0 or in1

in0, in1

Data type: ANY

The input parameters in0 and in1 must either be of the same data type or must be convertible to a common data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

Return value

Data type: ANY

Selected input parameter

in0 If g = 0 (FALSE)

in1 If g = 1 (TRUE)

The data type corresponds to the common data type of the input parameters in0 and in1.

MUX function

Description

Expandable multiplex function

The return value is one of the *n* input parameters in0 to in*n*-1, depending on the value of input parameter k.

Note

This description is not relevant to LAD/FDB. The corresponding function is described in the SIMOTION LAD/FDB Programming Manual.

Declaration

```
FUNCTION MUX (
        k       : ANY_INT,
        in0     : ANY,
        ...
        inn-1   : ANY
        ) : ANY
```

Input parameter

k

Data type: ANY_INT

Selection of the input parameter in0 to in*n*-1.

The value range depends on the number n of the input parameters in0...in*n*-1: 0 ≤ k ≤ *n*-1.

If illegal values are specified, input parameter in0 will be selected.

in0

...

in*n*–1

Data type: ANY

The number n of the input parameters in0 and in*n*-1 may vary.

All input parameters in0 and in*n*-1 must either be of the same data type or must be convertible to a common data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

If *n* formal parameters in0 to in*n*-1 are specified when calling the function, this must be done in ascending, uninterrupted order; for four input parameters, for example, the identifiers of the formal parameters are: in0, in1, in2, in3.

Return value

Data type: ANY

Input parameter in*m* if input parameter k has the value *m*.

The data type corresponds to the common data type of the input parameters in0 to in*n*-1.

MAX function

Description

Expandable maximum function

The return value is the maximum value of *n* input parameters in0 to in*n*-1.

Note

This description is not relevant to LAD/FDB. The corresponding function is described in the SIMOTION LAD/FDB Programming Manual.

Declaration

```
FUNCTION MAX (
        in0    : ANY_ELEMENTARY,
        ...
        inn-1  : ANY_ELEMENTARY
        ) : ANY_ELEMENTARY
```

Input parameter

in0
...
in*n*–1

Data type: ANY_ELEMENTARY

The number n of the input parameters in0 and in*n*-1 may vary.

All input parameters in0 and in*n*-1 must either be of the same data type or must be convertible to the most powerful data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

If *n* formal parameters in0 to in*n*-1 are specified when calling the function, this must be done in ascending, uninterrupted order; for four input parameters, for example, the identifiers of the formal parameters are: in0, in1, in2, in3.

Return value

Data type: ANY_ELEMENTARY

Maximum of the input parameters

The data type corresponds to the most powerful data type of the input parameters in0 to in*n*-1.

MIN function

Description

Expandable minimum function

The return value is the minimum value of *n* input parameters in0 to in*n*-1.

Note

This description is not relevant to LAD/FDB. The corresponding function is described in the SIMOTION LAD/FDB Programming Manual.

Declaration

```
FUNCTION MIN (
        in0    : ANY_ELEMENTARY,
        ...
        inn-1  : ANY_ELEMENTARY
        ) : ANY_ELEMENTARY
```

Input parameter

in0
...
in*n*–1

Data type: ANY_ELEMENTARY

The number n of the input parameters in0 and in*n*-1 may vary.

All input parameters in0 and in*n*-1 must either be of the same data type or must be convertible to the most powerful data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

If *n* formal parameters in0 to in*n*-1 are specified when calling the function, this must be done in ascending, uninterrupted order; for four input parameters, for example, the identifiers of the formal parameters are: in0, in1, in2, in3.

Return value

Data type: ANY_ELEMENTARY

Minimum of the input parameters

The data type corresponds to the most powerful data type of the input parameters in0 to in*n*-1.

LIMIT function

Description

Limiting function

The input parameter *in* is limited to values between the lower limiting value *mn* and the upper limiting value *mx*.

The function value is calculated as follows with the MIN (Page 1509) and MAX (Page 1508) selection functions:

```
LIMIT = MIN (MAX (in, mn), mx)
```

Declaration

```
FUNCTION LIMIT (
        mn  : ANY_ELEMENTARY,
        in  : ANY_ELEMENTARY,
        mx  : ANY_ELEMENTARY
        ) : ANY_ELEMENTARY
```

Input parameter

mn

Data type: ANY_ELEMENTARY

Lower limiting value

All input parameters must either be of the same data type or must be convertible to the most powerful data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

in

Data type: ANY_ELEMENTARY

Value to be limited

All input parameters must either be of the same data type or must be convertible to the most powerful data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

mx

Data type: ANY_ELEMENTARY

Upper limiting value

All input parameters must either be of the same data type or must be convertible to the most powerful data type by implicit conversion (see "Elementary data type conversion" in the SIMOTION ST Programming Manual).

Return value

Data type: ANY_ELEMENTARY

MIN (MAX (in, mn), mx)

The data type corresponds to the most powerful data type of the input parameters.

4.2.9.12 Consistent access to global variables of derived data types (UDT)

General

When accessing **global** variables of derived data types (see **User-defined data types (UDT)** in the ST programming manual), the user is responsible for ensuring data consistency when multiple tasks access the same user variables (I/O variables, system variables, system variables of technology objects, global device variables, and global unit variables, see **Variable model** in the programming manuals).

Note

Consistent data access is always ensured within a task.

You can work with semaphores to ensure that global variables of derived data types (UDT) are written and read consistently.

A global variable of data type DINT serves as a semaphore. Use the following functions to change and test the status of the semaphore:

- _testAndSetSemaphore (Page 1512)
- _releaseSemaphore (Page 1513)

Consistent data access to global variables is ensured under the following conditions:

1. All tasks signal access to global variables by setting a semaphore.
2. All tasks access global variables only when the semaphore is enabled.

For more information on consistent data access and semaphores, refer to "Consistent reading and writing of variables (semaphores)" (Page 1570).

_testAndSetSemaphore function

Description

This function checks whether the semaphore is set.

The semaphore is always set when the function ends. Additional calls of this function (even from other programs) return a value of FALSE until the _releaseSemaphore (Page 1513) function is called.

Declaration

```
_testAndSetSemaphore     (
                          sema    : DINT
                         ) : BOOL
```

Input parameter

sema

Data type DINT

Sema is a **global** variable of the DINT data type; it serves as semaphore. It must not be indexed. If it is an element of an array, the index must be specified when compiling (e.g. a[2]).

Return value

Data type: BOOL

This return value determines whether the semaphore is set:

TRUE	The semaphore was successfully set/assigned by this call. If TRUE, the semaphore must be released again after completion of the critical code section by calling the **_releaseSemaphore** function so it can be assigned at another location.
FALSE	The semaphore was already set/assigned by another program.

Example

See Example: Consistent data access with semaphores (Page 1572).

_releaseSemaphore function

Description

This function releases the semaphore. The next call of the _testAndSetSemaphore (Page 1512) function (even from different programs) results in a return value of TRUE.

Declaration

```
_releaseSemaphore     (
                       sema    : DINT
                       ) : VOID
```

Input parameter

sema

Data type: DINT

Sema is a **global** variable of the DINT data type; it serves as semaphore. It must not be indexed. If it is an element of an array, the index must be specified when compiling (e.g. a[2]).

Return value

none

Example

See Example: Consistent data access with semaphores (Page 1572).

4.2.9.13 Access to system variables and inputs/outputs

General information on accessing system variables and inputs/outputs

The _getSafeValue and _setSafeValue functions allow a special error response for access to system variables, configuration data, or I/O variables. It is possible to specify a different response from what has been configured in the event of an error (see Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258)).

The system functions _getSafeValue and _setSafeValue require a lot of execution time. Therefore, only use them when necessary, e.g. in an IF statement, to wait for the restart of a TO.

With V4.1 and higher, you can also specify a "replacement value" or "last value" for accesses to system variables and config data (V4.1.3 and higher) in the event of an error (e.g. TO is in restart). The configuration data restartInfo.behaviorInvalidSysvarAccess is relevant for this. See System variables (Page 1245) or Configuration data (Page 1249).

The _getInOutByte function enables direct read access to individual I/O bytes by specifying the address.

See also

Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258)

System variables (Page 1245)

Configuration data (Page 1249)

_getSafeValue function

This function reads the specified system variable (or configuration data element) or I/O variable and returns the value in another variable.

When this function is used (instead of a variable assignment), the transition to STOP mode can be prevented if an error occurs when accessing system variables, configuration data or I/O variables (e.g. while restarting a technology object, or in the event of an I/O failure).

Note

Runtime of the function

- Up to SIMOTION Kernel V4.3
  The runtime can be very long. Therefore, this function is not suitable for use in fast cyclic tasks.
- As of SIMOTION Kernel V4.4
  The runtime has been optimized. Access from synchronous user tasks is possible.

Specifying the error response

You can control the error response using the *accessmode* parameter:

- CONFIGURED (default): The error response defined by *restart.behaviorInvalidSysvarAccess* is used, see Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258).
- NO_CHANGE:
  - System variables and configuration data:
    Variable is not read and the value remains undefined, see System variables (Page 1245).
  - I/O variables:
    With read access (to inputs or outputs): The last valid value is applied.
- DEFAULT_VALUE: Substitute value or limit value is read.
- STOP_DEVICE: SIMOTION device switches to STOP mode. When a transition to STOP mode occurs, the ExecutionFaultTask is not started.

You can use the return value to determine whether the access was successful.

Declaration

```
_getSafeValue (
         variable      : ANY,        //System variable,
                                     // Configuration data or
                                     // I/O variable
         accessMode    : EnumAccessMode,
         getValue      : ANY
         )             : EnumSetAndGetSafeValue
```

Input parameter

variable

Data type: ANY

System variable, configuration data item or I/O variable to be read.

accesssMode

Data type: EnumAccessMode

Response when an error occurs during read access.

```
TYPE EnumAccessMode : (
// Response as configured:
    CONFIGURED  // System variables and config data:
                // Behavior depends on config data item
                // restartInfo.behaviorInvalidSysvarAccess
                // For I/O variables:
                // Strategy specified during creation
                // of the I/O variables
// Different response than configured:
    NO_CHANGE      // For system variables and
                   // configuration data: Variable
                   // do not read.
                   // For I/O variables: Accept
                   // last available (valid)
                   // value.
    DEFAULT_VALUE  // Configured value or
                   // Substitute value for I/O variables
    STOP_DEVICE    // Device switches to STOP mode.
END_TYPE
```

getvalue

Data type: ANY

Name of the variable to which the current value of the system variable (or configuration data item) or I/O variable is written.

The following applies for the data type:

- It must be the same as the data type of the variable to be read, or
- The data type of the variable to be read must be able to be converted to this data type by means of implicit type conversion.

Return value

Data type: EnumSetAndGetSafeValue

The return value indicates whether the access was successful.

```
TYPE EnumSetAndGetSafeValue : (
 // Access successful:
 OK            // Access was successful.
 // Access error:
, NO_CHANGE  // For system variables and configuration data: Variable was
             // not read, last value
             // For I/O variables: Last
             // available (valid) value was
             // accepted.
, DEFAULT_VALUE    // Only for I/O variables:
                   // Substitute value was accepted.
, INVALID_VALUE    // Only for configuration data:
                   // Configuration data item was
                   // not read, default value is returned,
                   // impermissible parameter
                   //(accessMode = DEFAULT_VALUE).
, NO_ACCESS )      // Only for configuration data:
                   // Configuration data item was
                   // not read, default value is returned
END_TYPE
```

See also

Errors when accessing system data with _get/_setSafeValue (Page 1295)

_setSafeValue function

The function behaves differently as of V4.1.3. It writes the specified value to the system variable, configuration data item or I/O variable and there is an option for it to return the currently written value in another variable. The response in the event of a fault can be specified differently from the configured response.

When this function is used (instead of a variable assignment), the transition to STOP mode can be prevented if an error occurs when accessing system variables, configuration data or I/O variables (e.g. while restarting a technology object, or in the event of an I/O failure).

Note

Runtime of the function

- Up to SIMOTION Kernel V4.3
  The runtime can be very long. Therefore, this function is not suitable for use in fast cyclic tasks.
- As of SIMOTION Kernel V4.4
  The runtime has been optimized. Access from synchronous user tasks is possible.

Specifying the error response

You can control the error response using the *accessmode* parameter:

- CONFIGURED (default): The error response specified in *restart.behaviorInvalidSysvarAccess* is used, see Errors when accessing system variables and configuration data as well as I/O variables for direct access (Page 1258).
- NO_CHANGE:
  - System variables and configuration data:
    Value is not written and remains unchanged or retains the last available value.
  - I/O variables:
    With write access (to outputs): The value is written to the variable. However, it will not be active at the output until the output becomes available again.
- DEFAULT_VALUE: Substitute value or limit value is written.
- STOP_DEVICE: SIMOTION device switches to STOP mode. When a transition to STOP mode occurs, the ExecutionFaultTask is not started.
- Value range for variable

The value is restricted to the value range and is not affected by the accessMode parameter.

You can use the return value to determine whether the access was successful.

Declaration

```
_setSafeValue (
        variable     : ANY,         //System variable,
                                   // Configuration data or
                                   // I/O variable
        value        : ANY,
        accessMode   : EnumAccessMode,
        setValue     : ANY
         ) : EnumSetAndGetSafeValue
```

Input parameter

variable

Data type: ANY

System variable, configuration data item or I/O variable to be written.

value

Data type**:** ANY

Value to be written to the system variable (or configuration data item) or I/O variable. The value data type must be the same as the data type of the variable to be written to, or it must be able to be converted to this data type by means of implicit type conversion.

accessMode

Data type: EnumAccessMode

Response when an error occurs during write access:

```
TYPE EnumAccessMode : (
 // Response as configured:
     CONFIGURED       // System variables and config data:
                       // Behavior depends on config data item
                       // restartInfo.behaviorInvalidSysvarAccess
                       // For I/O variables:
                       // Strategy specified during creation
                       // of the I/O variables
 // Different response than configured:
    , NO_CHANGE        // For system variables and
                       // configuration data: Current value
                       // Is not written.
                       //For I/O variables
                       // The value transferred in the
                       // parameter is written to the
                       // written. It is not active at the output
                       // until the output is
                       // again available.
, DEFAULT_VALUE
                       // For system variables and config data:
                       // Variable is not described.
                       // For I/O variables:
                       // Variable is described with the
                       // substitute value specified
                       // when creating the variable.
, STOP_DEVICE )        // Device goes into STOP
                       // mode
END_TYPE
```

setvalue

Data type: ANY

Name of a variable to which the current value of the system variable, configuration data item, or I/O variable is written.

The following applies for the data type:

- It must be of the same data type as the variable to be written, or
- The data type of the variable to be written must be able to be converted to this data type by means of implicit type conversion.

If, for example, 100 is to be written, but only 90 was entered, *setValue* is given a value of 90.

Return value

Data type: EnumSetAndGetSafeValue

The return value indicates whether the access was successful.

```
TYPE EnumSetAndGetSafeValue : (
 // Access successful:
 OK                // Access was successful.
 // Access error:
, NO_CHANGE        // For system variables and config data:
                   // Value is not written.
                   // For I/O variables: The value transferred
                   // in the parameter was
                   // written. It is not active at the output
                   // until the output is
                   // again available.
, DEFAULT_VALUE    // For system variables: Limitation
                   // active, limit value was
                   // written.
                   // For I/O variables: Substitute value was
                   // written. It is not active at the output
                   // until the output is
                   // again available.
, NO_ACCESS )     // Only for configuration data and system variables:
                   // Value of the configuration data item was
                   // not changed,
                   // Configuration data item
                   // not available).
END_TYPE
```

See also

Errors when accessing system data with _get/_setSafeValue (Page 1295)

_getInOutByte function

This function enables direct read access to individual I/O bytes through specification of the input/output address.

In the event of an access error, the PeripheralFaultTask is not called; rather a corresponding value is returned in the functionResult component of the return value.

If an I/O variable (for direct access or the process image of the cyclic task) (see **Direct access and process image of the cyclical tasks** in the ST Programming Manual) is defined for the specified I/O address and a replacement value is specified, this replacement value is returned in the event of an access error.

Evaluation of the return value can determine, for example, whether an input or an output is assigned to the address in the hardware.

Note

The runtime of this function can be very long. Therefore, this function is not suitable for use in fast cyclic tasks.

Declaration

```
_getInOutByte (
            mode          : EnumInOutDirection,
            logAddress    : DINT
         )                 : StructRetGetInOutByte
```

Input parameters

mode

Data type: EnumInOutDirection

Specifies whether logAddress is used to access the input or output.

```
TYPE EnumInOutDirection : (
      IN            // Access to input
 ,    OUT )        // Access to output
END_TYPE
```

logAddress

Data type: DINT

Logic address of input or output

Return value

Data type: StructRetGetInOutByte

Function call result and read byte value

```
TYPE StructRetGetInOutByte : STRUCT
    functionResult    : DINT;    // Result of the
                                 // function call
    value            : BYTE;     // Value of byte read
 END_STRUCT;
END_TYPE
```

Possible values of functionResult (result of function call):

0	No error, access okay.
16#FFFF_FFFA	Not enough memory available
16#FFFF_FFFE	Access error
16#FFFF_FFFF	Input/output not available

4.2.9.14 Backing up data from the user program

General information on saving data sets from the user program

The functions below are for:

- Saving, loading, or initializing the following values:
  - Non-retentive or retentive global unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)
  - Non-retentive or retentive global device variables (declared via project navigator)

  Data backup is performed by selecting the relevant function
  - Binary: The backed-up data set can no longer be read after the version ID of the data segment has been changed.
  - In XML format: The backed-up data set can be read after the version ID of the data segment has been changed.
- Managing the data set in which the values are backed up

The values are backed up in a data set that is identified uniquely by specifying the storage location, the name of the unit, and a data set number.

The application of these functions - in particular, the step enabling condition - is described in detail in Data backup and initialization from user program (Page 1573).

As well as saving retentive variables as individual data sets, it is also possible to back up all the retentive data to the memory card using the _savePersistentMemoryData function. For more information, see the *System Functions/Variable Devices* Parameter Manual.

_saveUnitDataSet function

Description

The values of the following variables are saved as a binary data set:

- Non-retentive or retentive global unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit, LAD/FBD unit)
- Non-retentive or retentive global device variables (declared via project navigator)

You can select the location at which the data set will be stored:

- Temporary data storage (RAM disk), erased in the event of a power failure
- Permanent data storage (memory card) is retained if there is a power failure

Pay attention to consistency of the data to be backed up, see Consistent reading and writing of variables (semaphores) (Page 1570).

When calling this function in short form, all parameters (including all optional parameters) must be specified.

The "Save variables" and "Restore variables" SCOUT functions can be used to retain data sets saved with _saveUnitDataSet, even if there is a change of version or data segments have been

changed. For additional information, refer to the SIMOTION SCOUT Configuration Manual or the online help.

Declaration

```
_saveUnitDataSet (
            unitName        : STRING,
            id              : UDINT,
            storageType     : EnumDeviceStorageType,
          { path            : STRING,
            overwrite       : BOOL,
            nextCommand     : EnumNextCommandMode,
            dataScope       : EnumDeviceDataScope,
            kindOfData      : EnumDeviceKindOfData,
            }
        ): StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit, LAD/FBD unit) whose unit variables will be saved.

The name must be indicated in lower case and inside single quotation marks (e.g. 'st_unit1').

If '_device' is specified, the global device variables will be saved.

id

Data type: UDINT

Number of the data set under which the variable values are saved (maximum of 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location at which the data is saved.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING
Default: ' ' (empty string)
Destination path, if storageType = USER_STORAGE.
Only the default value may be specified. Otherwise, an error message is output.

overwrite (optional)
Data type: BOOL
Default: FALSE
If TRUE, existing data set is overwritten.

nextCommand (optional)
Data type: EnumNextCommandMode
Default: IMMEDIATELY
Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

dataScope (optional)
Data type: EnumDeviceDataScope
Default _INTERFACE

```
TYPE EnumDeviceDataScope : (
    _INTERFACE,
        // Interface section
    _IMPLEMENTATION,
        // Implementation section
    _INTERFACE_AND_IMPLEMENTATION )
        // Interface and implementation section
END_TYPE
```

If unitName = '_device' is specified, only the values _INTERFACE or _INTERFACE_AND_IMPLEMENTATION are permissible for dataScope.

kindOfData (optional)
Data type: EnumDeviceKindOfData
Default NO_RETAIN_GLOBAL
Specification of whether non-retentive or retentive global variables will be saved.

```
TYPE EnumDeviceKindOfData : (
    NO_RETAIN_GLOBAL,
        // Non-retentive variables
    _RETAIN,
        // Retentive variables
    ALL_GLOBAL )
        // Retentive and non-retentive variables
END_TYPE
```

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A functionResult : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A handle : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

Table 4-102 Data type return value

```
TYPE
  EnumDeviceUnitDataSetCommand: (
    DONE,       // Execution or start successful
    ACTIVE,                 // Being processed
    INTERNEL_ERROR,         // Internal error
            // (please call the hotline)
    COMMAND_FAILED,         // Command cannot be executed
    NO_COMMAND_BUFFER_AVAILABLE,  // Command buffer full
    COMMAND_NOT_FOUND,      // Command (handle) not found
    DATASET_ID_NOT_VALID,   // Data set number is invalid
    READ_ERROR,             // Data read error
                            // (defective storage medium)
    NO_STORAGE_AVAILABLE,   // No memory space available
    ACCESS_DENIED,          // Access denied
                            // (missing write/read rights)
    DATASET_ALREADY_EXISTS, // File already exists
    DATASET_NOT_FOUND,      // Data set not found
    UNIT_NOT_FOUND,         // Unit not found
            // (e.g. ST source file, MCC unit) not available
    VERSION_MISSMATCH,      // Incorrect version ID
            // of the data area to be imported
    DATA_INCOMPLETE,    // Incomplete import of the data
    SYMBOL_INFORMATION_NOT_AVAILABLE,
            // Symbol information not available
            //(Activate the OPC-XML export at the program source)
    DATA_MISMATCH,      // Data area to be imported
                        // is not contained in the data set
    DATA_INCOMPATIBLE )
            // Data set contains additional data segments or
            // selected data segments are missing in the data set

  StructRetUnitDataSetCommand  : STRUCT
    functionResult : EnumDeviceUnitDataSetCommand;
    handle : UDINT;
  END_STRUCT;
END_TYPE
```

For information about data backup, see "Data backup and data initialization from user program" (Page 1573)

_loadUnitDataSet function

Description

The values of the following variables are loaded from a binary data set saved with _saveUnitDataSet (Page 1522):

- Non-retentive or retentive global unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit, LAD/FBD unit)
- Non-retentive or retentive global device variables

The data set can only be loaded if the version codes of all the data segments to be loaded (e.g. non-retentive and retentive variables of the interface section of the unit) have remained unchanged since the data set was saved. For data segments and their version ID, see Section "Version ID of global variables and their initialization during download" in the Programming Manuals, e.g. SIMOTION ST.

A subset of the data segments backed up in the data set can be loaded (e.g. if the version codes of some data segments have changed).

When calling this function in short form, all parameters (including all optional parameters) must be specified.

The "Save variables" and "Restore variables" SCOUT functions can be used to retain data sets saved with _saveUnitDataSet, even if there is a change of version or data segments have been changed. For additional information, refer to the SIMOTION SCOUT Configuration Manual or the online help.

Declaration

```
_loadUnitDataSet (
            unitName        : STRING,
            id              : UDINT,
            storageType     : EnumDeviceStorageType,
          { path            : STRING,
            nextCommand     : EnumNextCommandMode,
            dataScope       : EnumDeviceDataScope,
            kindOfData      : EnumDeviceKindOfData,
            }
        ): StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit), whose unit variables will be loaded.

The name must be indicated in lower case and inside single quotation marks (e.g. 'st_unit1').

If '_device' is specified, the global device variables will be loaded.

id

Data type: UDINT

Number of the data set from which the variable values are loaded (maximum of 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location at which the data is saved.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        // is retained if there is a power failure
    USER_STORAGE )
        // with path specification,
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: '' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

nextCommand (optional)

Data type: EnumNextCommandMode

Default: IMMEDIATELY

Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

dataScope (optional)

Data type: EnumDeviceDataScope

Default _INTERFACE

Specification of the section whose unit variables are to be saved.

```
TYPE EnumDeviceDataScope : (
    _INTERFACE,
        // Interface section
    _IMPLEMENTATION,
        // Implementation section
    _INTERFACE_AND_IMPLEMENTATION )
        // Interface and implementation section
END_TYPE
```

If unitName = '_device' is specified, only the values _INTERFACE or _INTERFACE_AND_IMPLEMENTATION are permissible for dataScope.

kindOfData (optional)

Data type: EnumDeviceKindOfData
Default NO_RETAIN_GLOBAL

Specification of whether non-retentive or retentive global variables will be loaded.

```
TYPE EnumDeviceKindOfData : (
    NO_RETAIN_GLOBAL,
        // Non-retentive variables
    _RETAIN,
        // Retentive variables
    ALL_GLOBAL )
        // Retentive and non-retentive variables
END_TYPE
```

If retentive and non-retentive variables are stored in the saved data set, it is possible to load retentive or non-retentive variables selectively.

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand. component.
  This supplies information on errors and the current state.
- A *handle* : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section **Return value** of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573)

_exportUnitDataSet function

Description

The values of the following variables are exported in XML format as a data set:

- Non-retentive global unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit or LAD/FBD unit)
- Retentive global unit variables of the interface or the implementation section of a unit

You can select the location at which the data set will be stored:

- Temporary data storage (RAM disk), erased in the event of a power failure
- Permanent data storage (memory card) is retained if there is a power failure

Non-retentive or retentive global device variables can only be backed up with the _saveUnitDataSet (Page 1522) function.

Make sure the symbol information of the unit variables in the SIMOTION device is available for the relevant unit. Therefore, activate the **Permit OPC-XML** option in the local settings of the compiler at the program source, see Section "Local settings of the compiler" in the Programming Manuals, e.g. SIMOTION ST.

Pay attention to consistency of the data to be exported, see Consistent reading and writing of variables (semaphores) (Page 1570).

The exported data set can also be imported with the _importUnitDataSet (Page 1531) function if the version ID of the data area (e.g. retentive variables of the interface section of the unit) has changed (e.g. by a change to the data structure).

When calling this function in short form, all parameters (including all optional parameters) must be specified.

Declaration

```
_exportUnitDataSet (
            unitName        : STRING,
            id              : UDINT,
            storageType     : EnumDeviceStorageType,
          { path            : STRING,
            overwrite       : BOOL
            nextCommand     : EnumNextCommandMode,
            dataScope       : EnumDeviceDataScope,
            kindOfData      : EnumDeviceKindOfData,
            }
        ): StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit), whose unit variables will be exported.

The name must be indicated in lower case and inside single quotation marks (e.g. 'st_unit1').

Specifying '_device' (for exporting global device variables) is not permissible here (only possible with _saveUnitDataSet (Page 1522)).

id

Data type: UDINT

Number of the data set under which the variable values are saved (maximum of 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location at which the data is saved.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: ' ' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

overwrite (optional)

Data type: BOOL

Default: FALSE

If TRUE, existing data set is overwritten.

nextCommand (optional)

Data type: EnumNextCommandMode

Default: IMMEDIATELY

Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

dataScope (optional)

Data type: EnumDeviceDataScope

Default _INTERFACE

Specification of the section whose unit variables will be exported.

```
TYPE EnumDeviceDataScope : (
    _INTERFACE,
        // Interface section
    _IMPLEMENTATION,
        // Implementation section
    _INTERFACE_AND_IMPLEMENTATION )
        // Interface and implementation section
END_TYPE
```

kindOfData (optional)

Data type: EnumDeviceKindOfData

Default NO_RETAIN_GLOBAL

Specification of whether non-retentive or retentive global variables will be exported.

```
TYPE EnumDeviceKindOfData : (
    NO_RETAIN_GLOBAL,
        // Non-retentive variables
    _RETAIN,
        // Retentive variables
    ALL_GLOBAL )
        // Retentive and non-retentive variables
END_TYPE
```

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A *handle* : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section **Return value** of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573)

_importUnitDataSet function

Description

The values of the following variables are imported from a data set that was exported with _exportUnitDataSet (Page 1528).

- Non-retentive global unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit or LAD/FBD unit)
- Retentive global unit variables of the interface or the implementation section of a unit

Make sure the symbol information of the unit variables in the SIMOTION device is available for the relevant unit. Therefore, activate the **Permit OPC-XML** option in the local settings of the compiler at the program source, see Section "Local settings of the compiler" in the Programming Manuals, e.g. SIMOTION ST.

The data set can also be imported if the version code of a data segment to be loaded has changed since the data set was saved (e.g. non-retentive variables of the interface section of the unit):

- Variables that no longer exist are ignored.
- The value of added variables remains unchanged.

- In the case of a variable with a changed data type, the value is transferred if it can be converted to the new data type; otherwise the value of the variable is retained.
- The returned value of the function is DATA_INCOMPLETE.

To avoid unwanted values in variables, you can initialize the relevant data segments before importing the data set with the _resetUnitData function (see List Manual for the system functions of the SIMOTION devices).

For data segments and their version ID, see Section "Version ID of global variables and their initialization during download" in the Programming Manuals, e.g. SIMOTION ST.

A subset of the data segments exported in the data set can be imported.

When calling this function in short form, all parameters (including all optional parameters) must be specified.

Declaration

```
_importUnitDataSet (
            unitName        : STRING,
            id              : UDINT,
            storageType     : EnumDeviceStorageType,
          { path            : STRING,
            nextCommand     : EnumNextCommandMode,
            dataScope       : EnumDeviceDataScope,
            kindOfData      : EnumDeviceKindOfData,
            }
        ): StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit), whose unit variables will be imported.

The name must be indicated in lower case and inside single quotation marks (e.g. 'st_unit1').

Specifying '_device' (for importing global device variables) is not permissible here (only possible with _loadUnitDataSet (Page 1526)).

id

Data type: UDINT

Number of the data set under which the variable values are saved (maximum of 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location at which the data is saved.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: ' ' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

overwrite (optional)

Data type: BOOL

Default: FALSE

If TRUE, existing data set is overwritten.

nextCommand (optional)

Data type: EnumNextCommandMode

Default: IMMEDIATELY

Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

dataScope (optional)

Data type: EnumDeviceDataScope

Default _INTERFACE

Specification of the section whose unit variables will be imported.

```
TYPE EnumDeviceDataScope : (
    _INTERFACE,
        // Interface section
    _IMPLEMENTATION,
        // Implementation section
    _INTERFACE_AND_IMPLEMENTATION )
        // Interface and implementation section
END_TYPE
```

kindOfData (optional)

Data type: EnumDeviceKindOfData

Default NO_RETAIN_GLOBAL

Specification of whether non-retentive or retentive global variables will be imported.

```
TYPE EnumDeviceKindOfData : (
    NO_RETAIN_GLOBAL,
        // Non-retentive variables
    _RETAIN,
        // Retentive variables
    ALL_GLOBAL )
        // Retentive and non-retentive variables
END_TYPE
```

If retentive and non-retentive variables are stored in the exported data set, it is possible to import retentive or non-retentive variables selectively.

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A *handle* : UDINT. component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section **Return value** of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573)

_deleteUnitDataSet function

Description

A single data set containing the stored values of the following variables is deleted:

- Backed-up non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)
- Backed-up non-retentive or retentive global device variables.
- Exported non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)

When calling this function in short form, all parameters (including all optional parameters) must be specified.

Declaration

```
_deleteUnitDataSet (
            unitName       : STRING,
            id             : UDINT,
            storageType    : EnumDeviceStorageType,
          { path           : STRING,
            nextCommand    : EnumNextCommandMode,
            }
        ): StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit); the name must be written in lower case and placed inside single quotation marks (e.g. 'st_unit1').

If '_device' is specified, a data set for global device variables will be deleted.

id

Data type: UDINT

Number of the data set to be deleted (maximum of 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location from which the data set is to be deleted.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: ' ' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

nextCommand (optional)

Data type: EnumNextCommandMode
Default: IMMEDIATELY
Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A *handle* : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section Return value of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573)

See also

_deleteAllUnitDataSets function (Page 1539)

_getStateOfUnitDataSetCommand function

Description

It returns the state of the functions for data backup.

Declaration

```
_getStateOfUnitDataSetCommand (
            handle  : UDINT
        ) : EnumDeviceUnitDataSetCommand
```

Input parameter

handle

Data type: UDINT

Handle of the data backup function for which the state is to be checked. You received this handle as a component of the return value of the data backup function.

Return value

Data type: EnumDeviceUnitDataSetCommand

This supplies information on errors and the current state.

For further information on the EnumDeviceUnitDataSetCommand data type, see Section **Return value** of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573).

_checkExistingUnitDataSet function

Description

A check is performed to determine whether the specified data set containing the stored values of the following variables is available on the storage medium:

- Backed-up non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)
- Backed-up non-retentive or retentive global device variables.
- Exported non-retentive or retentive unit variables of the interface section of a unit (e.g. ST source file, MCC unit)

When calling this function in short form, all parameters (including all optional parameters) must be specified.

Declaration

```
_checkExistingUnitDataSet (
            unitName       : STRING,
            id             : UDINT,
            storageType    : EnumDeviceStorageType,
          { path           : STRING,
            nextCommand    : EnumNextCommandMode,
            }
        ) : StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit); the name must be written in lower case and placed inside single quotation marks (e.g. 'st_unit1').

If '_device' is specified, a data set for global device variables will be checked (possible as of Version 3.2 of the SIMOTION Kernel).

id

Data type: UDINT

Number of the data set (max. 1_000_000 data sets per unit).

storageType

Data type: EnumDeviceStorageType

Location at which the data is stored.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: ' ' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

nextCommand (optional)

Data type: EnumNextCommandMode

Default: IMMEDIATELY

Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A *handle* : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section Return value of the _saveUnitDataSet (Page 1525) function.

For information about data backup, see "Data backup and data initialization from user program" (Page 1573).

_deleteAllUnitDataSets function

Description

All data sets containing stored values for the following variables are deleted.

- Backed-up non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)
- Backed-up non-retentive or retentive global device variables.
- Exported non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit)

When calling this function in short form, all parameters (including all optional parameters) must be specified.

Declaration

```
_deleteAllUnitDataSets (
            unitName       : STRING,
            storageType    : EnumDeviceStorageType,
          { path           : STRING,
            nextCommand    : EnumNextCommandMode,
            }
        ) : StructRetUnitDataSetCommand
```

Input parameter

unitName

Data type: STRING

Name of the unit (e.g. ST source file, MCC unit); the name must be written in lower case and placed inside single quotation marks (e.g. 'st_unit1').

If '_device' is specified, all data sets for global device variables will be deleted (possible as of Version 3.2 of the SIMOTION Kernel).

storageType

Data type: EnumDeviceStorageType

Location from which the data set is to be deleted.

```
TYPE EnumDeviceStorageType : (
    TEMPORARY_STORAGE,
        // Temporary data storage (RAM disk),
        // is deleted if there is a power failure
    PERMANENT_STORAGE,
        // Permanent data storage (memory card),
        //  is retained if there is a power failure
    USER_STORAGE )
        // with path specification
        // (only default permissible)
END_TYPE
```

path (optional)

Data type: STRING

Default: ' ' (empty string)

Destination path, if storageType = USER_STORAGE.

Only the default value may be specified. Otherwise, an error message is output.

nextCommand (optional)

Data type: EnumNextCommandMode

Default: IMMEDIATELY

Advance to next command

```
TYPE EnumNextCommandMode : (
    IMMEDIATELY,
        // Immediately
    WHEN_COMMAND_DONE )
        // After completion or abort of the command
END_TYPE
```

Return value

Data type: StructRetUnitDataSetCommand

The return value is a structure of data type StructRetUnitDataSetCommand. It comprises the following:

- A *functionResult* : EnumDeviceUnitDataSetCommand component.
  This supplies information on errors and the current state.
- A *handle* : UDINT component.
  This provides the possibility, by means of the _getStateOfUnitDataSetCommand (Page 1536) function of checking the current state of a data backup function (especially useful with step enabling condition IMMEDIATELY).

For further information on data types StructRetUnitDataSetCommand and EnumDeviceUnit-DataSetCommand, see Section **Return value** of the _saveUnitDataSet (Page 1522) function.

For information about data backup, see "Data backup and data initialization from user program".

4.2.9.15 Functions for commandId

CommandID overview

Description

A CommandId is transferred every time a command is issued. This is saved on the command and represents a reference to the issued command.

A command is thus uniquely identifiable and traceable using the CommandId.

Note

If the CommandId is not assigned for the command call, the default value of the parameter takes effect with (0.0).

Further information

Further information can be found at Command execution diagnostics (Page 1239), Transition and step enabling conditions (Page 1233) and Using the commandId parameter correctly (Page 1680).

_getCommandId function

Description

This function supplies a project-wide unique commandId, which can be used for explicit identification of commands.

A commandId is always produced, there is no error feedback.

Declaration

```
_getCommandId ( ) : CommandIdType
```

Input parameters

None

Return value

Data type: CommandIdType

Project-wide unique CommandId for tracking the command status

```
TYPE
 CommandIdType : STRUCT
    SystemId_low        : UDINT;    // Lower-order part
    SystemId_high       : UDINT;    // Higher-order part
 END_STRUCT
END_TYPE
```

See also

Using the commandId parameter correctly (Page 1680)

_getSyncCommandId function

Description

This function supplies the user with a project-wide unique syncCommandId. This ID can be transferred to system functions BEGIN_SYNC and _startSyncCommands (see Parameter Manuals for SIMOTION devices) to start motion sequences synchronously.

A syncCommandId is always produced, there is no error feedback.

Declaration

```
_getSyncCommandId ( ) : CommandIdType
```

Input parameter

None

Return value

Data type: CommandIdType

Project-wide unique syncCommandId for tracking the command status.

```
TYPE
 CommandIdType : STRUCT
    SystemId_low        : UDINT;    // Lower-order part
    SystemId_high       : UDINT;    // Higher-order part
 END_STRUCT
END_TYPE
```

See also

Using the commandId parameter correctly (Page 1680)

4.2.9.16 Defining the waiting time

_waitTime function

Description

This function interrupts the task triggering this function until the time specified in the call has expired.

Note

The function should be used in MotionTasks only; using it in cyclic tasks may lead to time monitoring errors!

- With SynchronousTasks: You can configure whether the time watchdog is suspended. Time monitoring is active by default.
  With IPOsynchronousTask, additionally take the following into account: UserInterruptTasks will no longer be started by their triggering event!
- With other cyclic tasks (BackgroundTask, TimerInterruptTasks): Time monitoring is always active.

In cyclic tasks, use the system function blocks Timers (Page 1627) to implement wait times.

The _waitTime function is always executable, its return value = 0.

For information on how you can make a task wait for a certain time, see Making tasks wait a defined period (Page 1434).

Declaration

```
_waitTime (
        timeValue    : TIME    // Wait time
        ) : DINT               // Always = 0
```

Input parameter

timeValue

Data type: TIME

Indicates the time interval during which task processing is interrupted.

Return value

Data type: DINT

Is always 0.

See also

Time allocation in the round robin execution level (Page 1373)

Wait times in cyclic tasks (Page 1679)

4.2.9.17 Device-specific functions

_getDeviceId function

Description

The function reads the hardware ID of the SIMOTION device from its hardware information block. You specify the type of ID to be read as input parameter when calling the function.

Declaration

```
_getDeviceId (
              idType : EnumDeviceIdType
             ) : StructRetGetDeviceId
```

Input parameter

idType

Data type: EnumDeviceIdType

Specification of the ID to be read

```
TYPE EnumDeviceIdType : (
    SERIAL_NUMBER ,    // CPU serial number
    HW_TYPE ,          // Module type
    SPECIFIC_NUMBER ,  // Special OEM number
    ORDER_ID )         // Order number of the module
END_TYPE
```

Return value

Data type: StructRetGetDeviceId

The return value is a structure of data type StructRetGetDeviceId. It comprises the following:

- A *functionResult* component: DINT.
  This provides information on errors.
- An *id* component: STRING[254]
  This contains the read hardware ID of the memory card.

```
TYPE
  StructRetGetDeviceId : STRUCT
    functionResult : UDINT; // Error status
        // 16#00000000 : No error
        // 16#FFFF80C3 : Information not available
        // 16#FFFF8090 : Incorrect transfer parameter
        // 16#FFFF8099 : Internal error
    id : STRING[254];       // Read-out hardware ID
  END_STRUCT;
END_TYPE
```

_getMemoryCardId function

Description

The function reads the hardware ID of a memory card from its hardware information block. You specify the type of ID to be read as input parameter when calling the function (at present only serial number possible).

Declaration

```
_getMemoryCardId (
                  idType : EnumMemoryCardIdType
                 ) : StructRetGetMemoryCardId
```

Input parameter

idType

Data type: EnumMemoryCardIdType

Specification of the ID to be read

```
TYPE EnumMemoryCardIdType : (
    MEMORY_CARD_SERIAL_NUMBER ) // Memory card serial number
END_TYPE
```

Return value

Data type: StructRetGetMemoryCardId

The return value is a structure of data type StructRetGetMemoryCardId. It comprises the following:

- A *functionResult* component: DINT.
  This provides information on errors.
- An *id* component: STRING[254]
  This contains the read hardware ID of the memory card.

```
TYPE
  StructRetGetMemoryCardId : STRUCT
    functionResult : UDINT; // Error status
        // 16#0000_0000 : No error
        // 16#FFFF_FFFD : Internal error
        // 16#FFFF_FFF8 : Incorrect parameter
    id : STRING[254];       // Read-out hardware ID
  END_STRUCT;
END_TYPE
```

Note

Any leading "S" in the memory card serial number will be ignored and not returned in the return value. For determining the license key in the Web License Manager, it is irrelevant whether or not the serial number is specified with a leading "S".

_setDeviceErrorLED function

Description

The function sets the **Underlicensing of technology/option objects** error on the SIMOTION device. The corresponding LED flashes in red on the SIMOTION device (see Manual of the SIMOTION device).

Declaration

```
_setDeviceErrorLED ( ) : DINT
```

Input parameter

None

Return value

Data type:	DINT
16#0000_0000	No error
16#FFFF_FFFD	Internal error

_setDriveObjectSTW function

Description

The _setDriveObjectSTW system function can be used to set individual bits of the CU_STW control word for drive object 1 (DO1) of the drive unit. This control word is part of the manufacturer-specific PROFIdrive telegrams 390 ff (telegrams 39x) for the control unit of the drive unit.

No bits can be set that have been reserved for the communication between the SIMOTION device and the drive unit or for other purposes.

This function is available in the SIMOTION device as of version V4.1 SP2.

Declaration

```
_setDriveObjectSTW (
            logAddress  : DINT;
            STW1BitMask : UINT;
            STW1BitSet  : UINT
           ) : DINT
```

Input parameter

logAddress

Data type DINT

Logical base address of the send telegram for the drive object.

STW1BitMask

Data type UNIT

Bit mask for selecting the affected bits in the CU_STW control word of the send telegram.

Bits that are operated autonomously in the connection to the drive object of SIMOTION (e.g. for the synchronization, fault handling, etc.) cannot be accessed by this function (see the table below for the assignment and reservation of the bits in the CU_STW).

Bits that can be accessed via the function: 16#007E = 2#0000_0000_0111_1110.

STW1BitSet

Data type UNIT

Bit-coded value for writing in the CU_STW control word of the send telegram. Only those bits selected via STW1BitMask are written.

Return value

Data type: DINT

The return value informs the user about the result of the call.

16#0000_0000	Request successfully completed • Bits have been written in the control word • Bits in STW1 were already in the defined state
16#FFFF_8090	Job aborted Logical address is not available
16#FFFF_8091	Job aborted Addressed drive object does not support the function
16#FFFF_8093	Job aborted STW1BitMask contains bits blocked for the function
16#FFFF_809F	Job aborted Internal error, function cannot be executed, e.g. function called on SIMOTION devices of version V4.1 or V4.1 SP1

Bit assignment and reservation in CU_STW of the 39x telegrams

Bit	Name	Meaning	Access right
12-15	MLZ	Dyn. master sign-of-life	SIMOTION FW
11	-	Reserved - system	Reserved
10	DAG	Demand from AG	SIMOTION FW
9	-	Reserved - system	Reserved
8	-	Reserved - system	Reserved
7	FACK	Fault acknowledge	_resetDriveObjectFault

Bit	Name	Meaning	Access right
6	-		_setDriveObjectSTW
5	-		_setDriveObjectSTW
4	-		_setDriveObjectSTW
3	-		_setDriveObjectSTW
2	-		_setDriveObjectSTW
1	PING	Start pulse for time synchronization	_setDriveObjectSTW
0	SYN	Dyn. synchronization flag for system time cycle	SIMOTION FW

4.2.9.18 Determine the memory size of a variable or of a data type

_sizeOf function

Description

The function returns the memory size in bytes required for a variable or data type:

- By default, the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Exception - only as of Version 4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

Declaration

```
_sizeOf (
         in  : ANY  // Identifier of the data type or
                    // of the variables
         )   : DINT
```

Input parameter

in

Data type: ANY

Identifier of the variable or data type, whose size is to be determined.

Return value

Data type: DINT

Required memory size in bytes.

The memory size is specified taking account of the natural layout, i.e. in accordance with the assignment possibilities of the data types in the memory. Therefore, the effective size is determined, which is required for the use of the data type in an ARRAY.

The actual required size may be less.

Example

```
TYPE
    a_type  : STRUCT
        a   : LREAL;    // 8 bytes
        b   : BOOL;     // 1 byte
    END_STRUCT;
END_TYPE
//...
x := _sizeOf (in := a_type);    // Returns value 16
```

_firstIndexOf function

Description

This function returns the lower index limit for an ARRAY (variable or data type):

- In the case of an ARRAY with a defined length (i.e. the index limits are declared), the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Only as of Version V4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

Note

The function must not be used for variables of the STRING data type or directly on this data type. The function then returns the value 1.

Declaration

```
_firstIndexOf (
        in  : ARRAY [..] OF ANY  // Identifier of the ARRAY
                                 // (Data type or variable)
        )   : DINT
```

Input parameter

in

Data type: ARRAY [..] OF ANY

Identifier for the ARRAY (variable or data type) whose lower index limit is to be determined

Return value

Data type: DINT

Lower index limit

Example

```
TYPE
    array_1  : ARRAY [5..29] OF DINT;
    string_1 : STRING[80];
END_TYPE
// ...
x := _firstIndexOf (in := array_1);      // Returns value 5
y := _firstIndexOf (in := string_1);     // Returns value 1
```

_lastIndexOf function

Description

This function returns the upper index limit for an ARRAY (variable or data type):

- In the case of an ARRAY with a defined length (i.e. the index limits are declared), the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Only as of Version V4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

Note

The function must not be used for variables of the STRING data type or directly on this data type. The function then returns the last index permitted for indexed access (**not** the currently assigned length of the string).

Declaration

```
_lastIndexOf (
        in  : ARRAY [..] OF ANY  // Identifier of the ARRAY
                                 // (Data type or variable)
        )   : DINT
```

Input parameter

in

Data type: ARRAY [..] OF ANY

Identifier for the ARRAY (variable or data type) whose upper index limit is to be determined

Return value

Data type: DINT

Upper index limit.

Example

```
TYPE
    array_1  : ARRAY [5..29] OF DINT;
    string_1 : STRING[80];
END_TYPE
// ...
x := _lastIndexOf (in := array_1);      // Returns value 29
y := _lastIndexOf (in := string_1);     // Returns value 80
```

_lengthIndexOf function

Description

This function returns the number of array elements for an ARRAY (variable or data type):

- In the case of an ARRAY with a defined length (i.e. the index limits are declared), the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Only as of Version V4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

Note

The function must not be used for variables of the STRING data type or directly on this data type. The function returns the usable length of the string (identical with the return value of the _lastIndexOf function.

Declaration

```
_lengthIndexOf (
        in  : ARRAY [..] OF ANY  // Identifier of the ARRAY
                                 // (Data type or variable)
        )   : DINT
```

Input parameter

in

Data type: ARRAY [..] OF ANY

Identifier of the ARRAY (variable or data type) for which the number of array elements is to be determined

Return value

Data type: DINT

Number of array elements

The following applies:
_lengthIndexOf (*array name*) := _lastIndexOf (*array name*) - _firstIndexOf (*array name*) +1

Example

```
TYPE
    array_1 : ARRAY [5..29] OF DINT;
END_TYPE
// ...
x := _lengthIndexOf (in := array_1);      // Returns value 25
```

LOWER_BOUND function

Description

This function returns the lower index limit for an ARRAY (variable or data type):

- In the case of an ARRAY with a defined length (i.e. the index limits are declared), the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Only as of Version V4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

To use this function, you must activate the "Permit language extensions IEC61131-3rd Edition" compiler option, see SIMOTION ST Programming and Operating Manual.

Note

The function must not be used for variables of the STRING data type or directly on this data type. The function then returns the value 1.

Declaration

```
LOWER_BOUND (
        arr   : ARRAY [..] OF ANY  // Identifier of the ARRAY
                                   // (Data type or variable)
        { dim : UDINT          // ARRAY dimension
        }                      // (Only constant value 1 permitted)
        )     : DINT
```

Input parameter

arr

Data type: ARRAY [..] OF ANY

Identifier for the ARRAY (variable or data type) whose lower index limit is to be determined

dim (optional)

Data type: UDINT

Default setting: 1

ARRAY dimension (only constant value 1 permitted)

It may be specified only when a variable was passed in *arr*.

Return value

Data type: DINT

Lower index limit

Example

```
TYPE
    array_1  : ARRAY [5..29] OF DINT;
    string_1 : STRING[80];
END_TYPE
// ...
x := LOWER_BOUND (arr := array_1);      // Returns value 5
y := LOWER_BOUND (arr := string_1);     // Returns value 1
```

UPPER_BOUND function

Description

This function returns the upper index limit for an ARRAY (variable or data type):

- In the case of an ARRAY with a defined length (i.e. the index limits are declared), the return value is determined at the time of compilation. The function can then be used in constant expressions (e.g. during initialization).
- Only as of Version V4.2 of the SIMOTION kernel:
  In the case of an ARRAY with a dynamic length (i.e. the index limits are not declared), the return value is determined during runtime.

To use this function, you must activate the "Permit language extensions IEC61131-3rd Edition" compiler option, see SIMOTION ST Programming and Operating Manual.

Note

The function must not be used for variables of the STRING data type or directly on this data type. The function then returns the last index permitted for indexed access (**not** the currently assigned length of the string).

Declaration

```
UPPER_BOUND (
        arr   : ARRAY [..] OF ANY  // Identifier of the ARRAY
                                   // (Data type or variable)
        { dim : UDINT          // ARRAY dimension
        }                      // (Only constant value 1 permitted)
        )     : DINT
```

Input parameter

arr

Data type: ARRAY [..] OF ANY

Identifier for the ARRAY (variable or data type) whose upper index limit is to be determined

dim (optional)
Data type: UDINT
Default setting: 1
ARRAY dimension (only constant value 1 permitted)
It may be specified only when a variable was passed in *arr*.

Return value

Data type: DINT
Upper index limit.

Example

```
TYPE
    array_1  : ARRAY [5..29] OF DINT;
    string_1 : STRING[80];
END_TYPE
// ...
x := UPPER_BOUND (arr := array_1);      // Returns value 29
y := UPPER_BOUND (arr := string_1);     // Returns value 80
```

4.2.9.19 Determination of the logical address for an I/O variable

Function _getLogicalAdressOfIoVariable

Description

The function determines the start address of an I/O variable for direct access or process image of the cyclical tasks.

This function is especially useful if a symbolic assignment of the I/O variables to the addresses (Page 1197) is performed. Some functionalities, such as the evaluation of alarms or TSI, require the knowledge of the logical address for an I/O variable in advance.

When using the function, the standard rules for using I/O variables apply.

Note

Every change in the address list for I/O (e.g. changes to the I/O addresses, new I/O variables) requires a recompilation of all program sources which contain this function.

Declaration

```
_getLogicalAddressOfIoVariable (
                in : <IO_VARIABLE> // Identifier of an I/O variable
            ) : DINT
```

Input parameter

in

Data type: <IO_VARIABLE>

Identifier of an I/O variable for direct access or the process image of the cyclical tasks. The I/O variable must be defined in the address list of the detail view.

With arrays of I/O variables, the specification of a constant index is permitted.

Return value

Data type: DINT

Logical start address of the I/O variable. It is returned in the following value ranges:

- For I/O variables for inputs: 16#0000 .. 16#FFFF
- For I/O variables for outputs: 16#7000_0000 .. 16#7000_FFFF

4.2.9.20 Copy areas of arrays

_ArrayCopy function

Description

The function copies an area of an array (copy source) to any location of another array of the same base data type (copy target). This area is specified by:

- Specification of the area length to be copied (number of array elements to be copied)
- Optional specification of the area start to be copied within the copy source.
- Optional specification of the area start to be written within the copy target.

The base data type of the copy source and the copy target must be identical; the index specifications may, however, differ. For information about the base data type and the index specification: see SIMOTION ST Programming and Operating Manual.

The copy target may also be identical with the copy source; the correct copy sequence of the array elements is selected automatically.

Note

If the area to be copied or written lies completely or partially outside the array limits of the copy source or copy target, the following applies: The array limits determine the maximum number of copied array elements.

If the compiler already determines that no array elements are copied (return value = 0), a warning will be issued.

The function can be interrupted by other tasks. The consistency of the copy target so cannot be guaranteed.

Declaration

```
_ArrayCopy (
        Dest   : ARRAY [..] OF ANY  // Copy target
        Source : ARRAY [..] OF ANY  // Copy source
        Length : DINT    // Number of array elements to be copied
        { DestIndex   : DINT  // Start of the area to be written
          SourceIndex : DINT  // Start of the area to be copied
         }
         )     : DINT
```

In-out parameter

Dest

Data type: ARRAY [..] OF ANY

Copy target.

Array into which the area is to be copied.

The copy source and the copy target must be arrays of the same base data type, although they can have different index specifications. For information about the base data type and the index specification: see SIMOTION ST Programming and Operating Manual.

Source

Data type: ARRAY [..] OF ANY

Copy source.

Array from which the area is to be copied.

The copy source and the copy target must be arrays of the same base data type, although they can have different index specifications. For information about the base data type and the index specification: see SIMOTION ST Programming and Operating Manual.

Input parameter

Length

Data type: DINT

Length of the area to be copied (number of array elements to be copied)

DestIndex (optional)

Data type: DINT

Default setting: 0

Start of the area within the copy target into which the copied array elements are written (index of the copy target into which the first copied array element is written).

The value must lie within the declared array limits of the copy target.

SourceIndex (optional)

Data type: DINT

Default setting: 0

Start of the area to be copied within the copy source (index of the first copied array).

The value must lie within the declared array limits of the copy source.

Return value

Data type: DINT

Number of copied array elements

For a successful and error-free execution of the function, the return value represents the length of the area to be copied.

Example

```
VAR
    src_array  : ARRAY [6..36] OF DINT;
    dest_array : ARRAY [0..25] OF DINT;
    copy_length : DINT := 15;
END_VAR
// ...
ret := _ArrayCopy (
        Dest        := dest_array,
        Source      := src_array,
        Length      := copy_length,
        DestIndex   := 20,
        SourceIndex := 10);
IF ret = copy_length THEN
    ; // Copy successful
ELSE
    ; // Error handling
END_IF;
```

4.2.9.21 Display the intermediate results for "program status"

_trcVal function

Description

The function extends the display options for the "Program status" (Page 4949) diagnostic function for the SIMOTION ST programming language

By default, the program status shows only the values for simple variables of elementary data types. The intermediate results of formulas can also be displayed with the _trcVal function. To do this, pass the expression to be displayed as actual parameter to the _trcVal function.

The "Program status" display window shows the return value for each _trcVal function call in the form: "`_TRCVAL = ...`".

Note

If the _trcVal function is used for variables that cannot be displayed in the "Program status" (e.g. no simple variable), the compiler issues an error message.

The function cannot be used for enumerations.

For information about the "Program status" diagnostic function, see SIMOTION ST Programming and Operating Manual (Page 4951).

Declaration

```
_trcVal (
        in  : ANY_ELEMENTARY
        )   : ANY_ELEMENTARY
```

Input parameter

in

Data type: ANY_ELEMENTARY

The expression to be displayed in the "Program status".

Return value

Data type: ANY_ELEMENTARY

The same data type as the *in* input parameter.

Example

In the following example, the intermediate results from the calculation of the distance between two points in the Cartesian coordinate system are displayed in the "Program status".

```
VAR
    Distance,
    x_koord0, x_koord1,
    y_koord0, y_koord1,
    z_koord0, z_koord1 : LREAL;
END_VAR
// ...
Distance := SQRT (_trcVal (_trcVal (x_koord1 - x_koord0)**2) +
                  _trcVal (_trcVal (y_koord1 - y_koord0)**2) +
                  _trcVal (_trcVal (z_koord1 - z_koord0)**2));
```

The function return value is displayed six times in the form "`_TRCVAL = ...`" in the "program status". The sequence of the display corresponds to the sequence of occurrence in the _trcVal function in the formula.

4.2.9.22 Creating general references

REF function

Description

The function returns the reference for a variable (address information). This can, for example, be assigned to a variable declared as reference to the data type of the passed actual parameter (REF_TO).

Note

Note for instances of function blocks passed as in/out parameter (VAR_IN_OUT):

- The REF function cannot be used for static variables (VAR) of function blocks imported from technology packages or device-independent libraries.

The restriction does not apply to class instances.

To use this function, you must activate the "Permit object-oriented programming" compiler option, see SIMOTION ST Programming and Operating Manual.

Declaration

```
REF (
    in  : ANY
    )   : REF_TO ANY
```

Input parameter

in

Data type: ANY

The variable for which the reference should be formed

The following are permitted:

- Retentive and non-retentive unit variables or instances of classes or function blocks or their elements declared as such.
- In methods within classes or function blocks:
  - Static variables (VAR) of the higher-level class or function block (including instance variables for classes or function blocks).
  - The THIS keyword. It permits access to the instance variable of the higher-level class or function block.
- In function blocks:
  - Static variables (VAR) of the function block (including instance variables of classes or function blocks).
  - The THIS keyword. Permits access to the instance variable of the function block.

The variable must be able to be read and written.

Not permitted are:

- Variables of technology object data types
- Variables of object-oriented interfaces
- Variables of general references
- Retentive local variables within classes or function blocks
- I/O variables
- Global device variables
- Constants

Return value

Data type: REF_TO ANY

Reference to the variable.

The data type is "Reference to the data type of the input parameter *in*".

4.2.9.23 Additional available system functions

In SIMOTION additional system functions are available that are introduced, for example, by the SIMOTION devices and technology objects. The following table lists provides an overview of where these functions are described.

Table 4-103 Overview of additional system functions and system function blocks in SIMOTION ST

System function	Description
System functions of technology objects	SIMOTION Cam Technology Package, System Functions List Manual (reference list) SIMOTION TControl Technology Package List Manual (reference list) Additionally see the function manuals on the technology objects
System functions of SIMOTION devices	List Manual of the SIMOTION devices (reference list)
System functions for controlling axes in accordance with the PLCopen standard	SIMOTION Cam Technology Package, System Functions List Manual (reference list)
Standard functions for controlling I/O modules and drive components	List Manual for the corresponding I/O module and drive component (reference list)

4.2.9.24 Application of certain system functions

Programming messages

General

You can use the following functions to program messages, e.g. error messages, or check their status:

- _alarmSId (generation of a message without acknowledgment)
- _alarmSqId (generation of a message with acknowledgment)
- _alarmScId (query about message status)

However the prerequisite is an application-specific configured message name.

Note

You can use the *_writeAndSendMessage* function to make user-defined entries in the diagnostic buffer. For a description of this function, refer to the List Manuals for SIMOTION devices.

See also General information for the message programming (Page 1452).

Overview of the functions

The functions described here can be used in libraries.

During each call, **_alarmSId** generates a **message that does not require acknowledgment**. The message is triggered according to a signal and an associated value can be appended to it. The message is transferred to all display devices registered for this purpose.

During each call, **_alarmSqId** generates a **message that requires acknowledgment**. The message is triggered according to a signal and an associated value can be appended to it. The message is transferred to all display devices registered for this purpose and can be acknowledged at these devices.

Note

No display device registered

If no display device is registered, the acknowledgment status of an Alarm_SQ is not reset when there is a new "incoming" message from the same alarm instance.

In this status, an Alarm_SQ is no longer displayed by the _getPendingAlarms() function.

Therefore, use _alarmScId() to read the status of the message.

You use the following as input signals for the functions:

1. The signal that triggered the message:
   This is interpreted as follows:
   - If the signal represents a rising edge – relative to the last call with this message name – an incoming message is generated. An incoming message is also generated if the signal state is TRUE on the first call with this message name.
   - If the signal represents a falling edge – relative to the last call with this message name – an outgoing message is generated.
2. The message to be compiled:
   It is specified via a unique, project-wide AlarmId (Page 1453).
3. Optional associated values provided they have been specified in the message configuration.
   - Independent of the SIMOTON kernel version:
     One associated value of the general data type ANY_NUM or ANY_BIT possible
   - As of version V4.5 of the SIMOTION kernel:
     A maximum of 12 associated values of BYTE (or SINT, USINT) data type is possible; they are passed in an ARRAY [0..11] OF BYTE.
     Associated values with data types that comprise several bytes (e.g. WORD, DINT, REAL) are possible; they reduce the number of possible associated values. The user must ensure that their values are stored in the permissible array elements in accordance with the Big Endian byte order. The AnyType_to_BigByteArray function can be used for this purpose.

The **_alarmScId** function queries the status of a message and its acknowledgment status. The relevant message is specified via a unique AlarmId (Page 1453).

For information about the formal structure of functions, see:

- _alarmSId function (Page 1453).
- _alarmSqId function (Page 1456).
- _alarmScId function (Page 1459).

For message names configured application-specifically, see online help.

Note

You should only generate an outgoing message after an incoming one, otherwise an error message is output.

The associated value (optional) must be a variable of an elementary data type. Avoid the use of constant values in the function call!

Buffer management of AlarmS

Description

A message list with 40 buffer areas is available for the AlarmS messages. Incoming messages are entered with their ID in this message list. For each outgoing message, an incoming message with the same ID must exist in the message list.

For each of the total 40 list entries there is also a send buffer. This send buffer is used to organize the notification of the registered client (HMI or SIMOTION SCOUT).

Message list and send buffer

Message list and send buffer are used as follows:

Return value		Meaning
Return values for incoming message (function call with rising edge *sig*)		
	16#8002	All entries in the message list are already occupied. Message entry is not made in the message list.
	16#8003	There is still an outgoing message with this ID in the message list and the send buffer is still occupied. Message entry is not made in the message list.
	16#8004	There is already an incoming message for this ID in the message list. Message entry is not made in the message list.
	16#0000	A message with this ID has not been found in the message list and the send buffer is no longer occupied. • ID of the message is entered in the message list. • The associated send buffer is occupied. • The system function returns with return value 16#0000. • The registered clients will be notified. • The send buffer is released again after the successful notification of the clients. • The ID of the message remains as incoming message in the message list.
Return values for outgoing message (function call with falling edge *sig*)		

Return value		Meaning
	16#8003	There is still an incoming message for this ID in the message list and the send buffer is still occupied. Message entry is not made in the message list.
	16#8004	There is already an outgoing message for this ID in the message list. Message entry is not made in the message list.
	16#8007	An incoming entry has not been found for the ID. Message entry is not made in the message list.
	16#0000	An incoming message for the ID has been found in the message list and the send buffer is no longer occupied. • The entry in the message list will be overwritten with the outgoing message. • The associated send buffer is occupied. • The system function returns with return value 16#0000. • The registered clients will be notified. • The send buffer is released again after the successful notification of the clients. • The entry in the message list will be deleted.

Example of message generation

The example in the table checks the temperature and generates an incoming message with one associated value which does not have to be acknowledged (e.g. *Temperature too high, incoming*), if the temperature is too high. If the temperature drops below the maximum value specified, an outgoing message is generated (the incoming message disappears).

The message named *SCOUT_alarm_name* has been configured in SIMOTION SCOUT, as for example: `Temperature too high: @1I%2d@ degrees.` A status variable prevents the same message from being repeated. The *handleAlarm* program is assigned to the BackgroundTask.

This example is independent of the SIMOTON kernel version.

Example of message generation

```
INTERFACE
  PROGRAM handleAlarm;
END_INTERFACE

IMPLEMENTATION
  PROGRAM handleAlarm
    VAR
      retVal         : DWORD;         // Return value
      temperature    : INT;           // Status to be queried
      maxTemperature : INT  := 60;    // Signal comparison value for status
      mySignal       : BOOL := FALSE; // Signal yes/no
    END_VAR
    //...
```

Example of message generation

```
    IF temperature > maxTemperature THEN
      IF mySignal = FALSE THEN
        // Incoming message, does not require acknowledgment
        retVal := _alarmSId (
            Sig   := TRUE,
            Ev_id := _alarm.SCOUT_alarm_name,
            sd    := temperature);
        mySignal := TRUE;
      END_IF;
    ELSE
      IF mySignal = TRUE THEN
        // Outgoing message, does not require acknowledgment
        retVal := _alarmSId (
            Sig   := FALSE,
            Ev_id := _alarm.SCOUT_alarm_name,
            sd    := temperature);
        mySignal := FALSE;
      END_IF;
    END_IF;
    //...
  END_PROGRAM
END_IMPLEMENTATION
```

Example of messages with several associated values (as of kernel V4.5)

As of version V4.5 of the SIMOTION kernel, several associated values can be specified for message configuring.

The _alarmSId (Page 1453) and _alarmSqId (Page 1456) functions are passed the associated value in an ARRAY FIELD [0..11] OF BYTE.

Associated values in the ARRAY [0..11] OF BYTE

The following table shows which array elements in the ARRAY [0..11] OF BYTE are assigned the associated value of a specific data type, as well as the addressing of the associated values in the message configuring.

The following applies to an associated value with a data type comprising several bytes:

- It must be stored in consecutive array elements. The first index must be divisible by the number of bytes assigned to the data type.
- It must be stored in the array with big endian byte order (most significant byte at low memory address). The AnyType_to_BigByteArray (Page 1492) function can be used for this purpose.

Table 4-104 Ordering of the associated values in the ARRAY [0..11] OF BYTE and access to the message configuring

Data type	Access to the associated values depending on the index of the byte array											
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
BOOL	@1B...	@2B...	@3B...	@4B...	@5B...	@6B...	@7B...	@8B...	@9B...	@10B...	@11B...	@12B...
BYTE, USINT	@1Y...	@2Y...	@3Y...	@4Y...	@5Y...	@6Y...	@7Y...	@8Y...	@9Y...	@10Y...	@11Y...	@12Y...
WORD, UINT	@1W...		@2W...		@3W...		@4W...		@5W...		@6W...	

Data type	Access to the associated values depending on the index of the byte array											
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
INT	@1I...		@2I...		@3I...		@4I...		@5I...		@6I...	
DWORD, UDINT	@1X...				@2X...				@3X...			
DINT	@1D...				@2D...				@3D...			
REAL	@1R...				@2R...				@3R...			
LREAL	@1O...											
Combination examples												
Example 1	@1W...		@3Y...	@4Y...	@2R...							
Example 2	@1X...				@3I...		@7Y...	@8Y...	@3R...			
Example 3	@1W...		@3Y...	@4Y...	@5Y...				@3R...			

For example, an associated value of the INT data type can be stored with indexes [4] and [5] in the array elements. During the message configuring, access is made to this associated value with `@3I...`, i.e. with the format code `%4d` (decimal number with at least 4 digits), the output of this associated value is configured with `@3I%4d@`.

Example

In the message configuring, a *SCOUT_alarm* message is configured with five associated values:

- One associated value of DWORD data type
- One associated value of INT data type
- Two associated values of BYTE data type
- One associated value of REAL data type

The associated values are placed in ARRAY [0..11] OF BYTE passed to the _alarmSId (Page 1453) and _alarmSqId (Page 1456) functions in accordance with combination example 2 in the above table. The message text can, for example, be configured as follows:

```
Value 1 @1X%6X@, value 2 @3I%5d@, value 3 @7Y%2X@, value 4 @8Y%2X@,
value 5 @3R%8.2f@
```

The following program example shows how the associated values are written in big endian byte order in an ARRAY [0..11] OF BYTE and passed to the _alarmS function. For information about big endian byte ordering, see Converting between any data types and byte arrays (marshalling) (Page 1593).

Table 4-105 Example of message programming with several associated values

```
PROGRAM alarm_example
  VAR
    alarm_array : ARRAY [0..11] OF BYTE;
    value_1     : DWORD := 100;
    value_2     : INT   := -10;
    value_3     : BYTE  := 1;
    value_4     : BYTE  := 2;
    value_5     : REAL  := 3.14;
    ret_val     : DWORD;
  END_VAR
```

```
  alarm_array := AnyType_to_BigByteArray (anydata := value_1, offset := 0);
  alarm_array := AnyType_to_BigByteArray (anydata := value_2, offset := 4);
  alarm_array := AnyType_to_BigByteArray (anydata := value_3, offset := 6);
  alarm_array := AnyType_to_BigByteArray (anydata := value_4, offset := 7);
  alarm_array := AnyType_to_BigByteArray (anydata := value_5, offset := 8);
  // Incoming message
  ret_val := _alarmSId (Sig   := TRUE,
                        Ev_id := _alarm.SCOUT_alarm,
                        sd    := alarm_array);
END_PROGRAM
```

Checking the error number and status of a message (filtering return values)

_alarmSId and _alarmSqId functions

The return value of the _alarmSId (Page 1453) and _alarmSqId (Page 1456) functions contains the error number and thus indicates whether an error occurred during execution. As with most system functions, return value = 0 indicates error-free execution.

_alarmScId function

However, the return value of the _alarmScId (Page 1459) function indicates both the error number and the status of a message. Therefore, proceed as follows when checking the status with these functions:

1. First, filter the return value with the constant ALARMS_ERROR (= 16#8000).
   This enables you to determine whether an error occurred while the function was being executed. The filter and the error numbers are selected such that they are true when combined in an AND operation.
2. If no error has occurred, you can evaluate the status of the message.

You can find a complete listing of error numbers and message states in the description of the _alarmScId function (Page 1459).

Consequently, you can check for errors when the *_alarmScId function* is called (the retVal (Page 1459) variable of data type DWORD contains the return value of the function) as follows:

Example of error and status check with the _alarmScId function

```
retVal := _alarmScId (Ev_id := _alarm.SCOUT_alarm_name);
// Error check here
IF (retVal AND ALARMS_ERROR) <> 0 THEN
    // Condition satisfied, therefore an error has occurred.
    // Error number check
    IF retVal = (ALARMS_ERROR OR
        DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID) THEN
        ; // Message number not permitted.
    END_IF;
ELSE
    // Condition not satisfied, therefore no error has occurred.
    // Check of the message and acknowledgment state
    IF retVal = 16#0000 THEN
        ; // Outgoing message, not acknowledged
    ELSIF retVal = ALARMS_STATE THEN
        ; // Incoming message, not acknowledged
    ELSIF retVal = 16#0010 THEN
        ; // Message not available
    ELSIF retVal = (ALARMS_QSTATE OR ALARMS_STATE) THEN
       ; // Incoming message, acknowledged
    END_IF;
    //...
END_IF;
```

Note

You can use constant values and symbolic constants on an equal footing to check for errors, see _alarmScId function (Page 1459).

Consistent reading and writing of variables (semaphores)

Consistent data access

All accesses to variables of elementary data types (see Section *Elementary data types*) are managed consistently by the system. The system ensures that these variables do not change while you are processing them.

When accessing **global** variables of derived data types (see Section *User-defined data types (UDT)*), the user is responsible for ensuring data consistency when multiple tasks access the same variables (symbolic I/O variables, system variables of SIMOTION devices, system variables of technology objects, global device variables and unit variables, see Section *Variable model*).

Access to **local** variables of derived data types are always consistent, since they can only be used inside the program (or function or function block) in which they are defined.

Note

Consistent data access is always ensured within a task.

Semaphores

To ensure consistent reading and writing of global variables, you work with semaphores.

A global variable (e.g. *semaA*) of data type DINT is used as a semaphore. If the variable is an element of an array, the index must be specified when compiling (e.g. a[2]).

Use the following functions to change and test the status of the semaphore:

- _testAndSetSemaphore (sema : DINT) : BOOL
  This function is used to check whether the semaphore is set:
  - Return value TRUE: The semaphore is enabled.
  - Return value FALSE: The semaphore is set.

  When the function is ended, the semaphore is always set. Additional calls of this function (even from other programs) return a value of FALSE, until the _releaseSemaphore (semaA) function is called.
- _releaseSemaphore (sema: DINT) : VOID

For information on the formal structure of functions, see Section *Working with variables*.

The semaphore is released.

Consistent data access to global variables is ensured under the following conditions:

1. All tasks signal access to global variables by setting a semaphore.
2. All the tasks only access global variables when the semaphore is released.

Example: Consistent data access with semaphores

The example in the table illustrates the use of semaphores in a program that reads data and a program that writes data.

Table 4-106 Example for ensuring consistent access to global variables using semaphores

```
IMPLEMENTATION
    VAR_GLOBAL
        myArray : ARRAY [0..1] OF DINT;
        semaA : DINT;
    END_VAR

    PROGRAM Writer
        // Consistent writing of variables
        IF _testAndSetSemaphore(sema := semaA) THEN
            myArray[0] := 18;
            myArray[1] := 19;
            _releaseSemaphore(sema := semaA);
            // The semaphore must be released in the TRUE
            // branch of the query; this ensures that
            // it is only released when it has been
            // reset.
        ELSE
            ; // Error handling
        END_IF;
        // _releaseSemaphore(sema := semaA);
        // The release would be incorrect at this point;
        // the semaphore would always be released.
    END_PROGRAM
    PROGRAM Reader
        VAR
            var0        : DINT;
            var1        : DINT;
        END_VAR

        // Consistent reading of variables
        IF _testAndSetSemaphore(sema := semaA) THEN
            var0 := myArray[0];
            var1 := myArray[1];
            _releaseSemaphore(sema := semaA);
            // The semaphore must be released in the TRUE
            // branch of the query; this ensures that
            // it is only released when it has been
            // reset.
        ELSE
            ; // Error handling
        END_IF;
        // _releaseSemaphore(sema := semaA);
        // The release would be incorrect at this point;
        // the semaphore would always be released.
    END_PROGRAM

END_IMPLEMENTATION
```

Data backup and initialization from user program

Data backup and data initialization from user program - functions and instructions

From a user program, it is possible to save, load, or initialize the values of the following variables in data sets:

- Non-retentive or retentive unit variables of the interface or implementation section of a unit (e.g. ST source file, MCC unit).
- Non-retentive or retentive global device variables.

Various functions are available for this:

- _saveUnitDataSet (Page 1522): The values are stored as a binary data set.
- _loadUnitDataSet (Page 1526): The values are loaded from a binary data set saved with *_saveUnitDataSet*.
  Loading the data set is only possible if since saving the data set the version IDs of all data segments to be loaded have remained unchanged.
  Please also pay attention to the note below.
- _exportUnitDataSet (Page 1528): The values are exported in XML format as ZIP archive (file name *.dat).
- _importUnitDataSet (Page 1531): The values are imported from a data set that was exported in XML format as ZIP archive (file name *.dat) with *_exportUnitDataSet*.
  Importing the data set is also possible if, since the data set was saved, the version ID of a data segment to be loaded has changed:
  - Variables that no longer exist are ignored.
  - The value of added variables remains unchanged.
    You can, for example, initialize the relevant data segments with *_resetUnitData* before importing a data set to avoid unwanted values in variables.
  - In the case of a variable with a changed data type, the value is transferred if it can be converted to the new data type; otherwise the value of the variable is retained.
- _deleteUnitDataSet (Page 1534): A single data set is deleted.
- _checkExistingUnitDataSet (Page 1537): A check determines whether the specified data set exists on the storage medium.
- _deleteAllUnitDataSets (Page 1539): All data sets are deleted.
- _resetUnitData: The values of the variables are initialized, see *System Functions of Devices* List Manual.

For data segments and their version ID, see Section "Version ID of global variables and their initialization during download" in the Programming Manuals, e.g. SIMOTION ST.

The description for *_resetUnitData* can be found in the List Manual (reference list) for the system functions of the SIMOTION devices.

The input parameters (Page 1574) and the return value (Page 1575) of the functions are explained in detail in the following sections.

NOTICE
Data loss possible If the data structure of a data segment is changed in a unit or in the global device variables, the following applies to loading the project into the SIMOTION device: • All temporarily backed-up data sets are deleted. • None of the binary data sets that have been permanently saved (with *_saveUnitDataSet*) can be read for this data segment any longer.

Up to 16 data backups can run simultaneously on a device.

The number of data sets that can be saved - up to 1_000_000 - depends on the available memory space.

Pay attention to consistency of the data to be backed up or exported, see Consistent reading and writing of variables (semaphores) (Page 1570).

Note

You can upload data sets saved with *_saveUnitDataSet* or *_exportUnitDataSet* from the SIMOTION device. Use the **Save variables** function in SIMOTION SCOUT. The data sets saved with *_saveUnitDataSet* are automatically converted to XML format.

You can use the **Restore variables** function to download these data sets and variables back to the SIMOTION device. For this purpose select the relevant SIMOTION device in the project navigator and select the function from the context menu. For more information, refer to the SIMOTION SCOUT Configuration Manual.

This makes it possible, for example, to obtain the data saved with *_saveUnitDataSet*, even if it is initialized by a project download or if it becomes unusable (e.g. due to a SIMOTION version change).

Input parameters

The most important parameters are briefly outlined below. For a more detailed description of the individual functions and their parameters, refer to Backing up data from the user program (Page 1522).

- *unitName*: Name of the unit (e.g. ST source file, MCC unit)
  If '*_device*' is specified, the data backup function is applied to global device variables (possible with _saveUnitDataSet).
- *id*: Data set number
  Specifying a data set number as an index enables several versions of the unit variables to be saved and then downloaded selectively.
  id < 1_000_000

- *storageType*: Specifying the storage location allows you to select the following:
  - TEMPORARY_STORAGE: Data is saved to the RAM disk (deleted after a power failure).
  - PERMANENT_STORAGE: Data is saved to a memory card (and retained after a power failure).

  Note also the additional information on the Storage location and memory requirement (Page 1576).
- *nextCommand*: Step enabling condition
  Specifying the step enabling conditions enables you to select from the following options:
  - Execute next command in the ST source file immediately (IMMEDIATELY).
  - Wait until command is done (WHEN_COMMAND_DONE).

  Note also the additional information on the Step enabling condition (Page 1577).
- *dataScope*
  It enables the section of the unit to which the data backup function is applied to be selected.
  - _INTERFACE: Function is applied to the interface section of a unit.
  - _IMPLEMENTATION: Function is applied to the implementation section of a unit.
  - _INTERFACE_AND_IMPLEMENTATION: Function is applied to the interface and implementation section of a unit.

  If *unitName* = '*_device*' is specified, only the values _INTERFACE or _INTERFACE_AND_IMPLEMENTATION are permissible for *dataScope*.
- *kindOfData*:
  It permits selection of whether the data backup function will be applied to retentive or retentive global variables.
  - NO_RETAIN_GLOBAL: Function is applied to non-retentive global variables.
  - _RETAIN: Function is applied to retentive global variables.
  - ALL_GLOBAL: Function is applied to retentive and non-retentive global variables.

  If retentive and non-retentive variables are stored in a data set, it is possible to load or import retentive or non-retentive variables selectively.

Return value

The return value is a structure of data type *StructRetUnitDataSetCommand*. It comprises the following:

- A *functionResult* component: *EnumDeviceUnitDataSetCommand*
  This supplies information on errors and the current status.
- A *handle* component: UDINT
  This allows you to check the current status of a data backup function by means of the *_getStateOfUnitDataSetCommand* function (this is required for step enabling condition IMMEDIATELY).

Storage location and memory requirement

You define the storage location with input parameter *storageType*:

- TEMPORARY_STORAGE: Data sets are saved to the RAM disk.
- PERMANENT_STORAGE: Data sets are saved to the Memory Card (under \USER\SIMOTION \USER_DIR\).

The number of data sets that can be saved depends on the available memory space at the respective memory location. Information on the available memory space can be obtained by means of the Device diagnostics, **System utilization** tab (see online help).

Note

Do not fill the entire available memory space with data sets! Otherwise, it will no longer be possible to download a project to the target system or copy RAM to ROM if project data is increased!

You can estimate the required memory space for binary data sets (saved with _saveUnitDataSet (Page 1522)) with the following information:

- Elementary data types occupy their natural data width on the memory (see Table *Bit widths and value ranges of the elementary data types* in Chapter *Elementary data types*; data type BOOL occupies 1 byte).
- Additional memory space is required due to:
  - Adaptation of the memory addresses to word or double-word boundaries
  - Consistency information (approx. 100 bytes per data set)
  - Usual supplementary data of a file system (e.g. sector headers, directory, occupy whole sectors only)

For data sets exported in XML format (with _exportUnitDataSet (Page 1528)), the memory requirement is much higher and cannot be ascertained in this way.

Step enabling condition

General information

The step enabling condition is indicated in input parameter *nextCommand*.

- **Synchronous call**
  The function is called with parameter *nextCommand* := WHEN_COMMAND_DONE. The next command in the program source when the function has been completed.
  The return value contains the result of the executed function in its *functionResult* component (see example).
  This procedure is mainly used in sequential tasks (e.g. MotionTasks).
- **Asynchronous call**
  The execution of this function may take quite a long time. Therefore, the time watchdog might respond in connection with cyclic tasks (e.g. BackgroundTask).
  For this reason, the function can also be executed asynchronously by setting parameter *nextCommand* := IMMEDIATELY. In this case, the function is started, and then the next command in the source is immediately processed.
  From the return value you can see:
  - If the start was successful (component *functionResult* = DONE)
  - A handle for further status query (*handle* component)

  If the command start was successful, you must check the current status of the data backup function using the _getStateOfUnitDataSetCommand (Page 1536) function and the handle until the result is something other than ACTIVE (see example).

Example of a synchronous call (in sequential tasks)

The data backup function _loadUnitDataSet (Page 1526) is called with the step enabling condition WHEN_COMMAND_DONE.

```
VAR_GLOBAL
    ds_ret : StructRetUnitDataSetCommand;
    error  : BOOL := FALSE;
END_VAR

PROGRAM save_data_seq
    // Program is assigned to a sequential task.
    // Function is executed synchronously:
    ds_ret := _loadUnitDataSet (
            unitName := 'ds3',
            id := 1,
            storageType := TEMPORARY_STORAGE,
            nextCommand := WHEN_COMMAND_DONE);
    // Function completed, evaluate result
    IF (ds_ret.functionResult <> DONE) THEN
        error := TRUE;    // Fehler
    END_IF;
END_PROGRAM
```

Example of an asynchronous call (in cyclic tasks)

The data backup function _saveUnitDataSet (Page 1522) is called with the step enabling condition IMMEDIATELY. The _getStateOfUnitDataSetCommand (Page 1536) is then called until the _saveUnitDataSet function has been completed

```
VAR_GLOBAL
    error : BOOL := FALSE;
    ds_rslt      : EnumDeviceUnitDataSetCommand;
    ds_ret       : StructRetUnitDataSetCommand;
    cmd_busy     : BOOL := FALSE;
    cmd_done     : BOOL := FALSE;
END_VAR

PROGRAM save_data_cycl
    // Program is assigned to a cyclic task.
    IF NOT cmd_busy THEN
        cmd_busy := TRUE;
        // Function is executed asynchronously:
        ds_ret := _saveUnitDataSet (
                unitName := 'ds1',
                id := 1,
                storageType := TEMPORARY_STORAGE,
                overwrite   := TRUE,
                nextCommand := IMMEDIATELY);
        IF (ds_ret.functionResult <> DONE) THEN
            cmd_busy := FALSE;
            error := TRUE; // Start of the function has failed
            // (e.g. too many services)
        END_IF;
    ELSE
        // Function is running, wait for result:
        ds_rslt := _getStateOfUnitDataSetCommand (
                handle := ds_ret.handle);
        IF (ds_rslt <> ACTIVE) THEN
            cmd_busy := FALSE;
            IF (ds_rslt = DONE) THEN
                cmd_done := TRUE;
                // Function terminated successfully
            ELSE
                error := TRUE;
                // Function has failed
            END_IF;
        END_IF;
    END_IF;
END_PROGRAM
```

Accessing files from a user program (as of V4.4)

As of version 4.4 of the SIMOTION Kernel, the user can access the file system of the SIMOTION device via a program. The user can access the permanent storage medium (e.g. the memory card) or the temporary storage medium (e.g. the RAM disk) and read or write files. The user can also manage the files (e.g. copy, rename and delete).

The user can only use selected areas of the file system on the applicable storage medium so that SIMOTION function is not impaired. These areas are addressed by symbolic names.

The following sections describe how to:

- read or write files (Page 1579), and
- manage files (Page 1589).

Note

Memory cards are not designed to be rewritten an unlimited number of times. With this in mind, you should avoid writing user data from the application to the memory card cyclically. Depending on the system, a write operation from the application may trigger one or more write operations on the memory card. Therefore, we recommend you adopt a conservative approach in terms of the number of writing processes. In other words, do not perform more than 100,000 write access instances from the user program over the estimated service life of the application.

Never switch off a SIMOTION device during write accesses to the memory card. If the SIMOTION device is switched off during write accesses, this can result in destruction of the data and, in the worst case, damage the file system (FAT Table = table of contents) on the memory card.

Follow the instructions in the section "Recommended method of handling CF cards" in Commissioning and Hardware Installation Manual SIMOTION D4x5-2.

Reading and writing files from a user program (as of V4.4)

As of version 4.4 of the SIMOTION Kernel, it is possible to read and write files from a user program. These files can be on the permanent storage medium (e.g. memory card) or the temporary storage medium (e.g. RAM disk). The following function blocks are provided for this purpose. Their functionality and use is described below.

Brief description of function blocks

- _filehandle function block
  See the information about "Filehandle" in this section.
- _fileOpen function block
  This function block opens the file specified in the parameters STORAGETYPE, LOCATION and FILENAME and links them to the instance of the _filehandle function block transferred in the HANDLE parameter.
  For information about the STORAGETYPE, LOCATION and FILENAME parameters, see "General information about the storage locations of a file" in the following section (Page 1589).
  Specify using the OPENFILE parameter whether the file should only be opened with write authorization or with read and write authorization. A file can either be opened multiple times for read-only access or only once exclusively for read and write access.
  If a file that is not present is opened for read and write, it is created with a fixed length (as well as the path, if not present). The fixed length is specified in the FILESIZE parameter. If the file already exists under the specified path, this parameter is ignored.
  A maximum of five files can be opened at the same time.

- _fileSetPosition function block
  This function block determines the position within an open text file.
  This file is identified via the _filehandle function block instance transferred in the HANDLE parameter.
  The position is specified relative to the FILEPOSITION parameter (FILEPOSITIONOFFSET parameter).
- _fileRead function block
  This function block reads binary data from the current position of the open file and copies it to an ARRAY OF BYTE (BUFFER parameter).
  The file to be read is identified via the _filehandle function block instance transferred in the HANDLE parameter.
  The number of bytes to be read is specified in the DATALENGTH parameter. The BUFFERPOSITION parameter specifies the index of the ARRAY OF BYTE from which the data is copied into the array.
- _fileWrite function block
  This function block copies binary data from an ARRAY OF BYTE (BUFFER parameter) and writes it into the open file from the current position.
  The file to be written is identified via the _filehandle function block instance transferred in the HANDLE parameter.
  The number of bytes to be written is specified in the DATALENGTH parameter. The BUFFERPOSITION parameter specifies the index of the ARRAY OF BYTE from which the data is copied from the array.
- _fileReadLn function block
  This function block reads data from the current position up to the end of line (CR/LF - $0D$0A) from the open text file and copies it into a string (LINE parameter).
  The text file to be read is identified via the _filehandle function block instance transferred in the HANDLE parameter.
- _fileWriteLn function block
  This function block writes a string (LINE parameter) into the open text file from the current position and ends the line with CR/LF ($0D$0A).
  The text file to be written is identified via the _filehandle function block instance transferred in the HANDLE parameter.
  The written data is visible in the WRITTENLINE parameter.
- _fileClose function block
  This function closes the open file and releases the system resources.
  The file to be closed is identified via the _filehandle function block instance transferred in the HANDLE parameter.
  If the file was open for read and write access, potential changes are saved on the storage medium.

The syntax of these function blocks is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

Operation of the function blocks

The interfaces of the function blocks behave in accordance with the PLCopen specification "Technical Specification, Function blocks for motion control, Version 1.1".

The rules described in the "SIMOTION PLCopen Blocks" Function Manual in Chapter "General rules for the PLCopen FB interface" apply for the PLCopen function blocks.

The function blocks work asynchronously, i.e. after calling an instance of these function blocks, they are processed in the background.

1. They declare an instance of the function block, and class this instance in a program which is integrated into a cyclic task (e.g. BackgroundTask).
   As long as the value is set to FALSE in the EXECUTE input parameter, the instance of the function block is not executed.
2. A positive edge in the EXECUTE input parameter starts the function block. The BUSY output parameter is set to TRUE.
   As long as the job is being processed, then BUSY = TRUE. Further positive edges in the EXECUTE input parameter are then ignored and an instance of the function block can no longer be called.
   The instance of the function block is processed in the background.
3. When the job is finished, BUSY is set to FALSE.
   - If execution was successful, the DONE output parameter is set to TRUE.
   - If execution completed with errors, the ERROR output parameter is set to TRUE and the cause of error is displayed in the ERRORID output parameter.

   Only one of the three output parameters BUSY, DONE or ERROR can have the value TRUE at any one time.
4. A negative edge in the EXECUTE input parameter resets the DONE and ERROR output parameters to FALSE.
   If the negative edge occurs at EXECUTE while a job is active (BUSY = TRUE), the running job is not aborted. After the job has finished, the DONE or ERROR output parameters are, however, only in TRUE status for one cycle time.

The instances of the function blocks can also be called in sequential tasks (e.g. MotionTasks). However, the user must ensure the cyclic execution. Refer to the information in "Basic procedure for reading and writing files" later in this section.

Filehandle

Filehandle is an instance of the _filehandle function block. It is transferred to the function blocks for open, read, write and close, and identifies the file to be read or written uniquely. It provides the system resources to access the file system and its structure.

Calling an instance of the _fileOpen function block links the filehandle to a file in the file system The system resources are assigned.

The location at which the instance of the _filehandle function block has been declared determines its initialization behavior and therefore the life of the system resources. See also the section "Initialization of instances of function blocks (FBs)" in the SIMOTION ST Programming Manual.

The system resources are released again by closing the file with an instance of the _fileClose function block.

Note

- Accesses to the storage medium are buffered. For this reason, e.g. for write accesses, only once the file has been closed successfully with an instance of the _fileClose function block can it be ensured that the data have been written in full to the storage medium.
- If the instance of the _filehandle function block is declared as a static variable that is saved in the user memory of the task, the following applies:
  On transition to STOP mode, an existing job is processed first before the system resources are released. On rapid transition to RUN mode, it may therefore be the case that the system resources are still assigned.
  For information about variables in the user memory of the task, see section "Memory areas of the variable types" in the SIMOTION ST Programming Manual.

A separate call of an instance of the _filehandle function block provides information about the associated filehandle and the storage medium (available memory and number of open files).

The syntax of the _filehandle function block is described in detail in the "System Functions/ Variables Devices" List Manual (reference list) and in the online help (see index).

Basic procedure for reading and writing files

For an asynchronous call in a cyclic task (e.g. BackgroundTask)

The following basic procedure is recommended:

1. Declare instances of the _filehandle, _fileOpen and _fileClose function blocks and the other necessary function blocks.
2. Provide a start signal. For this purpose declare, e.g. a unit variable of data type BOOL with initialization value FALSE.
3. Call the instance of the _fileOpen function block first. Assign these values to the following parameters:
   - HANDLE: the instance of the _filehandle function block.
   - EXECUTE: the start signal.
   - STORAGETPYE, LOCATION and FILENAME: as per the file in the file system.
     See "General information about the storage locations of a file" in the next section (Page 1589).
   - OPENFILE: READONLY or READWRITE.
4. Call the required instances of the function blocks to read write or position. Assign these values to the following parameters:
   - HANDLE: the instance of the _filehandle function block.
   - EXECUTE: the DONE output parameter of the preceding instance.

5. Release the system resources. To do so, call the instance of the _fileClose function block. Assign these values to the following parameters:
   - HANDLE: the instance of the _filehandle function block.
   - EXECUTE: the DONE output parameter of the preceding instance.
6. Assign the program to a cyclic task in the execution system (e.g. BackgroundTask).
7. In RUN operating mode, assign the value TRUE to the start signal at the required time. Execution starts. The instances of the function block are processed sequentially in the background.

The following examples 1 and 2 explain this procedure.

For a synchronous call in a sequential task (e.g. MotionTask)

In order to use the function blocks in sequential tasks, proceed in a similar way as for asynchronous execution.

However, the user must ensure the cyclic execution, e.g through the following procedure:

1. Include each used instance of a function block in a loop (e.g. REPEAT ... END_REPEAT).
2. Also include a call of the system function _waitTime (Page 1543) with the parameter timeValue = T#0s in each of these loops.
   This ensures that the next MotionTask is executed. Also refer to information on "Cyclic MotionTasks" in Section "Time allocation in the round robin execution level" (Page 1373).
3. Use a suitable expression with the output parameters BUSY, DONE or ERROR as an abort condition for the loop.

Assign the program to a sequential task in the execution system (e.g. StartupTask, MotionTask).

This procedure is explained in the following example 3.

Examples

Example 1: Create file and write line (asynchronous call)

This example shows how to create a file and write a line in the file.

The program is called in a cyclic task (e.g. BackgroundTask). A positive edge at the bWrite variable starts execution.

```
INTERFACE
    PROGRAM WriteLineToFile;
    VAR_GLOBAL
        bWrite : BOOL := FALSE;
    END_VAR
END_INTERFACE

IMPLEMENTATION
  PROGRAM WriteLineToFile
    VAR
        myFilehandle    : _filehandle;
        myOpenFileFB    : _fileOpen;
        myWriteLnFileFB : _fileWriteLn;
        myCloseFileFB   : _fileClose;
    END_VAR

    myOpenFileFB (
        EXECUTE  := bWrite,
        STORAGETYPE := CARD,
        LOCATION := USERFILES,
        FILENAME := 'MyData/file_name.txt',
        OPENFILE := READWRITE,
        FILESIZE := 512,
        HANDLE   := myFilehandle);

    myWriteLnFileFB (
        EXECUTE := myOpenFileFB.DONE,
        LINE    := 'This is the text that is written.',
        HANDLE  := myFilehandle);

    myCloseFileFB (
        EXECUTE := myWriteLnFileFB.DONE,
        HANDLE  := myFilehandle);
  END_PROGRAM
END_IMPLEMENTATION
```

Example 2: Open file and read line (asynchronous call)

This example shows how to open an available file and read a line from a specific position.

The program is called in a cyclic task (e.g. BackgroundTask). A positive edge at the bRead variable starts execution.

```
INTERFACE
    PROGRAM ReadLineFromFile;
    VAR_GLOBAL
        bRead : BOOL := FALSE;
    END_VAR
END_INTERFACE

IMPLEMENTATION
  PROGRAM ReadLineFromFile
    VAR
        myFilehandle   : _filehandle;
        myOpenFileFB   : _fileOpen;
        myReadLnFileFB : _fileReadLn;
        mySetFilePosFB : _fileSetPosition;
        myCloseFileFB  : _fileClose;
        lineBuffer     : STRING[254];
    END_VAR

    myOpenFileFB (
        EXECUTE  := bRead,
        STORAGETYPE := CARD,
        LOCATION := USERFILES,
        FILENAME := 'MyData/file_name.txt',
        HANDLE   := myFilehandle);

    mySetFilePosFB (
        EXECUTE := myOpenFileFB.DONE,
        FILEPOSITION := BEGINNING_OF_FILE,
        FILEPOSITIONOFFSET := 12,
        HANDLE  := myFilehandle);

    myReadLnFileFB (
        EXECUTE := mySetFilePosFB.DONE,
        HANDLE  := myFilehandle,
        LINE    => lineBuffer);

    myCloseFileFB (
        EXECUTE := myReadLnFileFB.DONE,
        HANDLE  := myFilehandle);
  END_PROGRAM
END_IMPLEMENTATION
```

Example 3: Create file and write lines (synchronous call)

This example shows how to create a file and write some lines in the file.

The program is called in a sequential task (e.g. StartupTask, MotionTask). A positive edge at the bWrite variable starts execution.

```
INTERFACE
    PROGRAM writeFile_From_MotionTask;
    VAR_GLOBAL
        bWrite : BOOL := FALSE;
    END_VAR
END_INTERFACE

IMPLEMENTATION
  FUNCTION_BLOCK write
    VAR_IN_OUT
        execute : BOOL;
    END_VAR

    VAR
        mydret           : DINT;
        myFilehandle     : _filehandle;
        myOpenFileFB     : _fileOpen;
        myWriteLnFileFB  : _fileWriteLn;
        myCloseFileFB    : _fileClose;
        myOpenFileFB2    : _fileOpen;
        myWriteLnFileFB2 : _fileWriteLn;
        myCloseFileFB2   : _fileClose;
        myPosFileFB      : _fileSetPosition;
    END_VAR

    REPEAT
        myOpenFileFB (
            EXECUTE     := execute,
            STORAGETYPE := CARD,
            LOCATION    := USERFILES,
            FILENAME    := 'MyData2/file_name2.txt',
            OPENFILE    := READWRITE,
            FILESIZE    := 512,
            HANDLE      := myFilehandle);
        IF execute = TRUE THEN
            execute := FALSE;
        END_IF;
        mydret := _waittime (timeValue := T#0ms);
        UNTIL myOpenFileFB.DONE OR myOpenFileFB.ERROR
    END_REPEAT;

    REPEAT
        myWriteLnFileFB (
            EXECUTE := myOpenFileFB.DONE,
            LINE    := 'This is the text that is written.',
            HANDLE  := myFilehandle);
        mydret := _waittime (timeValue := T#0ms);
        UNTIL myWriteLnFileFB.DONE OR myWriteLnFileFB.ERROR
    END_REPEAT;

    REPEAT
        myCloseFileFB (
            EXECUTE  := myWriteLnFileFB.DONE,
            HANDLE   := myFilehandle);
        mydret := _waittime (timeValue := T#0ms);
```

```
      UNTIL myCloseFileFB.DONE OR myCloseFileFB.ERROR
  END_REPEAT;

  REPEAT
      myOpenFileFB2 (
          EXECUTE     := myCloseFileFB.DONE,
          STORAGETYPE := CARD,
          LOCATION    := USERFILES,
          FILENAME    := 'MyData2/file_name2.txt',
          OPENFILE    := READWRITE,
          FILESIZE    := 512,
          HANDLE      := myFilehandle);
      mydret := _waittime (timeValue := T#0ms);
      UNTIL myOpenFileFB2.DONE OR myOpenFileFB2.ERROR
  END_REPEAT;

  REPEAT
      myPosFileFB (
          EXECUTE            := myOpenFileFB2.DONE,
          FILEPOSITION       := BEGINNING_OF_FILE,
          FILEPOSITIONOFFSET := 50,
          HANDLE             := myFilehandle);
      mydret := _waittime (timeValue := T#0ms);
      UNTIL myPosFileFB.DONE OR myPosFileFB.ERROR
  END_REPEAT;

  REPEAT
      myWriteLnFileFB2(
          EXECUTE := myPosFileFB.DONE,
          LINE    := 'This is the second text.',
          HANDLE  := myFilehandle);
      mydret := _waittime (timeValue := T#0ms);
      UNTIL myWriteLnFileFB2.DONE OR myWriteLnFileFB2.ERROR
  END_REPEAT;

  REPEAT
      myCloseFileFB2(
          EXECUTE := myWriteLnFileFB2.DONE,
          HANDLE  := myFilehandle);
      mydret := _waittime (timeValue := T#0ms);
      UNTIL myCloseFileFB2.DONE OR myCloseFileFB2.ERROR
  END_REPEAT;
END_FUNCTION_BLOCK

PROGRAM writeFile_From_MotionTask
  VAR
      mywrite : write;
      biwrite : BOOL;
  END_VAR
  bwrite := TRUE;
  IF bwrite THEN
      bwrite  := FALSE;
      biwrite := TRUE;
      mywrite (execute := biwrite);
  END_IF;
END_PROGRAM
```

```
END_IMPLEMENTATION
```

Managing files from a user program (as of V4.4)

Files can be managed from a user program (e.g. copy, rename and delete) as of version 4.4 of the SIMOTION Kernel. The files can be on the permanent storage medium (e.g. memory card) or the temporary storage medium (e.g. RAM disk). The following function blocks are provided for this purpose. Their functionality and use is described below.

Brief description of function blocks

- _fileCopy function block
  This function block creates a copy of a file.
  The file to be copied is specified by the SOURCESTORAGETYPE, SOURCELOCATION and SOURCEFILENAME parameters.
  The copy destination is specified by the DESTINATIONSTORAGETYPE, DESTINATIONLOCATION and DESTINATIONFILENAME parameters.
- _fileRename function block
  This function block renames a file inside a directory.
  The file to be renamed is specified by the STORAGETYPE, LOCATION and FILENAME parameters.
  Enter the new file name in the NEWFILENAME parameter without specifying the path.
- _fileDelete function block
  This function block deletes a file from a folder.
  The file to be deleted is specified by the STORAGETYPE, LOCATION and FILENAME parameters.
  The file must not be open. If necessary, the _fileClose function block must be used first.
- _directoryPathDelete function block
  This function block deletes a folder including all sub-folders and their contents.
  The folder to be deleted is specified by the STORAGETYPE, LOCATION and DIRECTORYPATHNAME parameters.
- _getStateOfFile function block
  This function block provides status information about a file or a folder (e.g. size and modified date).
  The file or the folder is specified by the STORAGETYPE, LOCATION and FILENAME parameters.

The syntax of these function blocks is described in detail in the "System Functions/Variables Devices" List Manual (reference list) and in the online help (see index).

General information about the storage locations of a file

For the above function blocks and the _fileOpen (Page 1579) function block, a file is specified in the file system via three entries and the associated parameters.

1. The storage medium (e.g. STORAGETYPE parameter).
   The following values are permissible:
   - RAMDISK: temporary, non-retentive storage medium, e.g. RAM disk.
   - CARD: permanent storage medium, e.g. memory card.

   Note

   Files that have been created using the above function blocks or the _fileOpen function block on the temporary storage medium (e.g. RAM disk) are not copied to the permanent storage medium (e.g. memory card) using the "Copy RAM to ROM" SIMOTION SCOUT function.

2. The location on the storage medium (e.g. LOCATION parameter).
   The files are stored in the selected folders on the storage medium. These folders are selected via symbolic names. This ensures that the SIMOTION functions are not affected.
   The following values (symbolic names) are permitted:
   - USERFILES
     Corresponds to path: '\USER\SIMOTION\HMI\USERFILES'
   - USERLOG
     Corresponds to path: '\USER\SIMOTION\HMI\USERLOG'
   - USERDATASETS
     Corresponds to path: '\USER\SIMOTION\USER_DIR\UPP\UNITDS'
     All data and sub-folders in this folder are deleted when the following SIMOTION SCOUT function is used: "Delete user data on card" when the unit data sets checkbox is activated.
   - USERWEB
     Corresponds to path: '\USER\SIMOTION\HMI\FILES'
   - USERTEMP
     Corresponds to path: '\USER\SIMOTION\USER_DIR\USERTEMP'
     All data and sub-folders in this folder are deleted when the following SIMOTION SCOUT function is used: "Delete user data on card"

   Any folder structure can be created beneath the specified folder.

3. The path and the file name (e.g. FILENAME parameter)
   This allows files to be accessed in any folder tree beneath the folders selected using LOCATION.
   Permissible characters for file name and path:
   - All displayable characters from the ASCII character set are permitted as name of the file and sub-folders, except the following symbols:
     '"'($22), '*'($2A), '/'($2F), ':'($3A), '<'($3C), '>'($3E), '?'($3F), '\'($5C), '|'($7C).
   - The following symbols may be used as separator between the folder name and file name:
     '/'($2F), '\'($5C).

Operation of the function blocks

The function blocks are designed for asynchronous calls.

For information about this functionality, see the previous section (Page 1579).

Basic procedure to manage files

The following basic procedure is recommended:

1. Declare instances of the required function blocks.
2. Provide a start signal. For this purpose declare, e.g. a unit variable of data type BOOL with initialization value FALSE.
3. Call the required instances of the function blocks sequentially. Assign the following values to the EXECUTE parameter:
   - For the first instance: the start signal.
   - For the subsequent instances: the DONE output parameter of the preceding instance in each case.
4. Assign the program to a cyclic task in the execution system (e.g. BackgroundTask).
5. In RUN operating mode, assign the value TRUE to the start signal at the required time. Execution starts. The instances of the function block are processed sequentially in the background.

Example

This example demonstrates the following file management operations:

1. Rename a file
2. Copy the file from the permanent storage medium (e.g. memory card) to the temporary storage medium (e.g. RAM disk)
3. Delete the original file from the permanent storage medium

The program is called in a cyclic task. A positive edge at the bStart variable starts execution.

```
INTERFACE
    PROGRAM fileSystemOperation;
    VAR_GLOBAL
        bStart : BOOL := FALSE;
    END_VAR
END_INTERFACE

IMPLEMENTATION
  PROGRAM fileSystemOperation
    VAR
        myDeleteFileFB     : _fileDelete;
        myRenameFileFB     : _fileRename;
        myCopyFileFB       : _fileCopy;
        myGetStateOfFileFB : _getStateOfFile;
    END_VAR

    myRenameFileFB (
            EXECUTE  := bStart,
            STORAGETYPE := CARD,
            LOCATION := USERFILES,
            FILENAME := 'MyData/file_name.txt',
            NEWFILENAME := 'new_name.txt');

    myCopyFileFB (
            EXECUTE := myRenameFileFB.DONE,
            SOURCESTORAGETYPE := CARD,
            SOURCELOCATION    := USERFILES,
            SOURCEFILENAME    := 'MyData/file_name.txt',
            DESTINATIONSTORAGETYPE := RAMDISK,
            DESTINATIONLOCATION    := USERFILES,
            DESTINATIONFILENAME    := 'MyTemplates/file.txt');

    myGetStateOfFileFB (
            EXECUTE  := myRenameFileFB.DONE,
            STORAGETYPE := CARD,
            LOCATION := USERFILES,
            FILENAME := 'MyData/new_name.txt');

    IF myGetStateOfFileFB.FILEEXISTING = YES THEN
        myDeleteFileFB (
                EXECUTE  := myGetStateOfFileFB.DONE,
                STORAGETYPE := CARD,
                LOCATION := USERFILES,
                FILENAME := 'MyData/new_name.txt');
    END_IF;
  END_PROGRAM
END_IMPLEMENTATION
```

Converting between any data types and byte arrays (marshalling)

Conversion of variables of any data type to byte arrays, and vice versa, is frequently used to create defined transmission formats for data exchange between different devices (see also section Communication functions (Page 1596)).

Variables of any data type (elementary data types, standard data types of technology packages and devices, and user-defined data types) can be converted to byte arrays, and vice versa, using the following functions:

- AnyType_to_BigByteArray (Page 1492)
- AnyType_to_LittleByteArray (Page 1492)
- BigByteArray_to_AnyType (Page 1494)
- LittleByteArray_to_AnyType (Page 1494)

For all functions, an offset can be specified optionally for the first element to be assigned or evaluated in the byte array.

A distinction is made as to:

- Conversion direction (to or from byte arrays)
- Byte order in the array (see table):
  - Big endian: Most significant byte at low memory address (Motorola, SUN Sparc, SIMATIC S7)
  - Little endian: Least significant byte at low memory address (Intel, DEC Alpha)

Table 4-107 Examples of byte order (big endian and little endian)

Address	Number 34677374 = 16#2_11_22_7E (data type UDINT)		Character string "Byte" (data type DWORD)	
	Big endian	Little endian	Big endian	Little endian
2#...11	16#7E = 126	16#02 = 2	16#65 = "e"	16#42 = "B"
2#...10	16#22 = 34	16#11 = 17	16#74 = "t"	16#79 = "y"
2#...01	16#11 = 17	16#22 = 34	16#79 = "y"	16#74 = "t"
2#...00	16#02 = 2	16#7E = 126	16#42 = "B"	16#65 = "e"

Note

TO data types cannot be converted. See section “Conversion of technology object data types” (Page 1503) for conversion of TO data types.

If structures and arrays are to be converted, consistency is guaranteed only at the elementary variable level. Any higher level of consistency must be ensured by the user (see the following sections:

- "Consistent reading and writing of variables (semaphores)" (Page 1570) and
- Consistent access to global variables of derived data types (UDT) (Page 1512).

The following data types are not portable convertible, i.e their transmission formats are not defined across systems. Conversions between SIMOTION devices are possible without any restrictions:

- Time types: For transmission format, see Table
- Enumerators (enumeration data types)

When converting these data types, the compiler outputs a warning (16013).

Table 4-108 Transmission formats of time data types for marshalling functions

Data type	Transmission format
TIME	Time specification, in ms (UDINT)
TIME_OF_DAY (TOD)	Time of day starting with TOD#0:0:0 in ms (UDINT)
DATE	Number of days since DATE#1992_01_01 (UDINT). DATE#1992_01_01 corresponds to 1
DATE_AND_TIME (DT)	Summary of TOD and DATE in the order specified.

Note

The marshalling function result may cause errors when the program is running, the error response set in the task configuration will then be triggered, see Execution errors in programs (Page 1256).

Proceed with caution when converting byte arrays to the general ANY_REAL data type or to structures that contain this data type. The bit string from the byte array is taken unchecked as the ANY_REAL value. You must make sure that the bit string of the byte array corresponds to the bit pattern of a normalized floating-point number in accordance with IEEE 754. To do this, you can use the _finite (Page 1503) and _isNaN (Page 1504) functions or the IS_VALID (Page 1505) function.

Otherwise, an error can be triggered (FPU exception (Page 1257)) as soon as the ANY_REAL value is first used for an arithmetic operation (for example, in the program or when monitoring in the symbol browser).

Table 4-109 Example of marshalling function use

```
TYPE
    t_struct_1 : STRUCT
        m_word       : WORD;
        m_byte       : ARRAY [0..1] OF BYTE;
    END_STRUCT;
    t_struct_2 : STRUCT
        m_struct     : ARRAY [0..3] OF t_struct_1;
        m_lreal      : LREAL;
    END_STRUCT;
END_TYPE

VAR
    gsbVar           : t_struct_2;
    big_b_Array      : ARRAY [0..23] OF BYTE;
    lit_b_Array      : ARRAY [0..23] OF BYTE;
END_VAR

// Assignment of the values to the structure
gsbVar.m_struct[0].m_word     := WORD#16#7FF1;
gsbVar.m_struct[0].m_byte[0]  := BYTE#16#F9;
gsbVar.m_struct[0].m_byte[1]  := BYTE#16#E8;
gsbVar.m_struct[1].m_word     := WORD#16#9FF7;
gsbVar.m_struct[1].m_byte[0]  := BYTE#16#80;
gsbVar.m_struct[1].m_byte[1]  := BYTE#16#A1;
gsbVar.m_struct[2].m_word     := WORD#16#A881;
gsbVar.m_struct[2].m_byte[0]  := BYTE#16#BC;
gsbVar.m_struct[2].m_byte[1]  := BYTE#16#CD;
gsbVar.m_struct[3].m_word     := WORD#16#CF9C;
gsbVar.m_struct[3].m_byte[0]  := BYTE#16#D8;
gsbVar.m_struct[3].m_byte[1]  := BYTE#16#E7;
gsbVar.m_lreal                := LREAL#-12345.6789e123;
// Conversion to big endian
big_b_Array := AnyType_to_BigByteArray (
                    anyData  := gsbVar,
                    offset   := 0);
 // Content of the elements of big_b_array (big endian):
 // See 2nd column in the following table

// Conversion to little endian
lit_b_Array := AnyType_to_LittleByteArray (
                    anyData  := gsbVar,
                    offset   := 0);
 // Content of the elements of lit_b_array (little endian):
 // See 3rd column in the following table

// Conversion from big endian
gsbVar := BigByteArray_to_AnyType (
                    bytearray := big_b_Array,
                    offset    := 0);

// Conversion from little endian
gsbVar := LittleByteArray_to_AnyType (
                    bytearray := lit_b_Array,
                    offset    := 0);
```

Table 4-110 Content of the array elements of big_b_array and lit_b_array from example

Byte array big_b_array or lit_b_array			Components of the gsbVar variable	
Array index	big_b_array (big endian)	lit_b_array (little endian)	Name	Value
23	BYTE#16#07	BYTE#16#DA	m_lreal	LREAL#-12345.6789e123
22	BYTE#16#F0	BYTE#16#52		
21	BYTE#16#43	BYTE#16#3C		
20	BYTE#16#68	BYTE#16#EC		
19	BYTE#16#EC	BYTE#16#68		
18	BYTE#16#3C	BYTE#16#43		
17	BYTE#16#52	BYTE#16#F0		
16	BYTE#16#DA	BYTE#16#07		
15	BYTE#16#D8	BYTE#16#D8	m_struct[3].m_byte[1]	BYTE#16#D8
14	BYTE#16#E7	BYTE#16#E7	m_struct[3].m_byte[0]	BYTE#16#E7
13	BYTE#16#9C	BYTE#16#CF	m_struct[3].m_word	WORD#16#CF9C
12	BYTE#16#CF	BYTE#16#9C		
11	BYTE#16#CD	BYTE#16#CD	m_struct[2].m_byte[1]	BYTE#16#CD
10	BYTE#16#BC	BYTE#16#BC	m_struct[2].m_byte[0]	BYTE#16#BC
9	BYTE#16#81	BYTE#16#A8	m_struct[2].m_word	WORD#16#A881
8	BYTE#16#A8	BYTE#16#81		
7	BYTE#16#A1	BYTE#16#A1	m_struct[1].m_byte[1]	BYTE#16#A1
6	BYTE#16#80	BYTE#16#80	m_struct[1].m_byte[0]	BYTE#16#80
5	BYTE#16#F7	BYTE#16#9F	m_struct[1].m_word	WORD#16#9FF7
4	BYTE#16#9F	BYTE#16#F7		
3	BYTE#16#E9	BYTE#16#E9	m_struct[0].m_byte[1]	BYTE#16#E9
2	BYTE#16#F9	BYTE#16#F9	m_struct[0].m_byte[0]	BYTE#16#F9
1	BYTE#16#F1	BYTE#16#7F	m_struct[0].m_word	WORD#16#7FF1
0	BYTE#16#7F	BYTE#16#F1		

Communication functions

Available functions

ST provides the following functions for communication over non-configured connections:

- _Xsend, see Parameter description for _Xsend (Page 1597)
- _GetStateOfXCommand, see Parameter description for _GetStateOfXCommand (Page 1599)
- _Xreceive, see Parameter description for _Xreceive (Page 1599)
- _tcpsend, see Communication via Ethernet with TCP/IP protocol (Page 1603)
- _tcpreceive
- _udpsend, see Communication via Ethernet with UDP protocol (Page 1604)
- _udpreceive

These communication functions are used to send and receive data:

- Between SIMOTION devices
- Between SIMOTION and SIMATIC S7 devices
  (S7-300, S7-400, M7-300, M7-400, etc.).
- Via acyclic communication; see also Acyclic communication with the drive (Page 1604).

The dual functions **_Xsend** and **_Xreceive** enable transparent transmission of data packets that are sent coordinately by the user program of the client and received coordinately by the user program of the server.

Further information

Additional information is also available in the **Communication System/Configuration Manual** or in Acyclic communication with the drive (Page 1604).

The program on the receiving device recognizes whether it is the data packet to be received on the basis of a user-definable integer appended to the data packet. The data exchange is only successful if the user program accepts and does not reject the data packet.

The sent data is in the form of byte sequences in an array, i.e. the data has no logical structure. SIMOTION devices can send or receive up to 200 bytes at once; the actual user data length depends on the communication peer.

You can check the status of an XSend or XReceive request with the **_GetStateOfXCommand** command.

Parameter description for _Xsend

The _Xsend function sends a data packet with transparent data to a communication peer. For the detailed syntax of the parameters, refer to the List Manual for the SIMOTION devices.

An overview is provided below:

- You can choose between two communication modes (*communicationMode* parameter): The connection is or is not retained after the data transmission.
- The *address* parameter specifies the destination address of the communications peer. The parameter is of data type *StructXSendDestAddr*. The following table lists the meaning of the individual components.

Table 4-111 Structure of destination address

Parameter (data type)	Meaning/values	Values	
deviceId (USINT)	Point of the connection	SIMOTION C230-2, C240, C240 PN	**1** for X8 **2** for X9
		SIMOTION P350, P350-3	**1** for X101 **2** for X102
		SIMOTION D410 DP	**1** for X21
		SIMOTION D410-2 DP	**2** for X21 **1** for X24
		SIMOTION D410-2 DP/PN	**2** for X21
		SIMOTION D4x5, D4x5-2	**1** for X126 **2** for X136
remoteSubnetIdLength (USINT)	Length of the subnet dialog box	**0** for MPI, PROFIBUS	
remoteStaddrLength (USINT)	Length of the station address (station number) of the destination system.	**1** for MPI, PROFIBUS	
nextStaddrLength (USINT)	Length of the router address	**0** for MPI, PROFIBUS	
remoteSubnetId (ARRAY [0..5] OF USINT)	Subnet mask	(Irrelevant for MPI, PROFIBUS)	
remoteStaddr (ARRAY [0..5] OF USINT)	Station address of the target system (actual destination address)	Station number for MPI, PROFIBUS, e.g. remoteStaddr[0] = 25	
nextStaddr (ARRAY [0..5] OF USINT)	Router address	(Irrelevant for MPI, PROFIBUS)	

Additional parameters of the _Xsend function are:

- The receiver identifies the data packet from the **job identifier** (*messageId* parameter) that you append to the data packet.
- You can select between two **modes of data transmission** (*nextCommand* parameter): synchronous or asynchronous.
  - During synchronous data transmission, the program waits until the receiver acknowledges receipt of the data packet before resuming execution. This happens automatically on receipt of the data packet.
  - During asynchronous data transmission, the program is resumed immediately after the command is issued. You can check the status of the command with *_GetStateOfXCommand*
- The mandatory **CommandId** parameter is used for internal command detection in ST. The parameter value should be saved to a local variable (data type *CommandIdType*) with the _getCommandId (Page 1541) function. This variable can be used as a parameter value.
- The **data packet** (*data* parameter) involves a list containing 200 entries of 1 byte each. The list does not have to be this long if the amount of data you are sending is less than the maximum.

- The **data length** (*dataLength* parameter) is used to specify the actual length of the data packet to be transmitted.
- You can determine whether the command was successfully executed (return value = 0) on the basis of the **return value**.
  If a return value other than 0 is returned, an error has occurred (see the command syntax in the List Manual for the SIMOTION devices).

Parameter description for _Xreceive

The *_Xreceive* function is used to receive transparent data sent by a communication peer with *_Xsend*.

Below is a brief overview of the function parameters (see also the section on system functions in the List Manual for the SIMOTION devices):

- You identify the anticipated data packet from the **job identifier** (*messageId* parameter) appended by the sender.
- As when sending data, you can choose between two **modes of data reception** (*nextCommand* parameter): synchronous or asynchronous.
  - During synchronous data receipt, the program waits until the data packet has arrived before it resumes execution. Receipt of the data packet is then acknowledged automatically to the sender.
  - During asynchronous data transmission, the program is resumed immediately after the command is issued. You can check the status of the commands with *_GetStateOfXCommand*.
- The mandatory **CommandId** parameter is used for internal command detection in ST. The parameter value should be saved to a local variable (data type *CommandIdType*) with the _getCommandId (Page 1541) function. This variable can be used as a parameter value.
- The **return value** is a structure:
  - You can determine whether the command was successfully executed (functionResult = 0) on the basis of the *functionResult* element.
    If a value other than 0 is returned, an error has occurred (see the command syntax in the List Manual for the SIMOTION devices).
  - The *dataLength* element represents the data length of the data packet received.
  - The *data* element represents the data packet received (array of up to 200 entries of one byte each).

Parameter description for _GetStateOfXCommand

You can check the status of an *_Xsend* or *_Xreceive* command with the _GetStateOfXCommand function.

Below is a brief overview of function parameters (see also documentation for the technology functions):

- The status query can tell which command is involved by the mandatory *commandId* parameter uniquely assigned to each send and receive function (see information about this parameter for *_Xsend* and *_Xreceive* above).
- The return value consists of an error number (zero when command execution is okay, otherwise greater than zero) and the status of the command being checked (see command syntax in the system functions for the SIMOTION devices).

Communication between SIMOTION and SIMATIC S7 devices

You must consider the following for communication between SIMOTION and SIMATIC S7 devices:

- Communication is only possible with *_Xsend* and *_Xreceive*.
- The maximum volume of data that can be transmitted in one packet is limited to 76 bytes. If you transmit larger data packets, an error message is output.
- The SIMOTION interface must be connected to the **MPI interface** of the SIMATIC S7 devices. The baud rate on the SIMOTION interface must be set to correspond to the baud rate of the SIMATIC S7device. For example, the baud rate for an SIMATIC S7-300 must be configured to 187.5 Kbits/s (see documentation for the relevant SIMATIC S7 devices).

Note

The parameters for the *_Xsend* and *_Xreceive* commands have different names in the SIMOTION system are in some cases have a different meaning than those you used in your previous SIMATIC S7 system. You will find a comparison in the following tables.
The *_GetStateOfXCommand* command does not exist in an SIMATIC S7 system, thus eliminating the need for comparison.

Table 4-112 Comparison of the parameters for _Xsend in SIMATIC S7 and SIMOTION devices

SIMATIC S7 device (parameters for SFC 65 X_SEND)	SIMOTION device (parameters for _Xsend)
REQ (Request to activate, data type BOOL)	– (Not available)
DEST_ID (MPI station number of the communications peer)	address (Destination address of the communication peer as a structure of data type StructXSendDestAddr)
CONT (Keep connected, Boolean value TRUE or FALSE)	communicationMode (Keep connected, enum. value M_TRUE or M_FALSE)
– (Not available)	nextCommand (Data transmission mode, enum values for synchronous and asynchronous)
REQ_ID (Job identifier, value of data type DWORD)	messageId (Job identifier, value of data type UDINT)

SIMATIC S7 device (parameters for SFC 65 X_SEND)	SIMOTION device (parameters for _Xsend)
SD (Variable address for sent data, value of data type ANY pointer, max. 76 bytes long)	data (Data packet to be sent, one-dim. array with max. 200 values of data type USINT)
– (Not available)	dataLength (Data length of data packet to be sent in bytes, value of data type UDINT)
– (Not available)	commandId (Internal command identifier, see description in parameter description above)
BUSY (Activation status, values of data type BOOL)	– (Not available)
<Return value> (= 0, if no error occurred; <> 0, if error occurred; see SIMATIC S7 documentation)	<Return value, (= 0, if no error occurred; <> 0, if error occurred; see Parameter Manual for the SIMOTION devices)

Table 4-113 Comparison of the parameters for _Xreceive in SIMATIC S7 and SIMOTION devices

SIMATIC S7 device (parameters for SFC 66 X_RCV)	SIMOTION device (parameters for _Xreceive)
EN_DT (Enable Data Transfer, data type BOOL)	– (Not available)
REQ_ID (Job identifier, value of data type DWORD)	messageId (Job identifier, value of data type UDINT)
– (Not available)	nextCommand (Data transmission mode, enum values for synchronous and asynchronous)
– (Not available)	commandId (Internal command identifier, see description in parameter description above)
RD (variable address for received data, value of data type ANY pointer, max. 76 bytes long)	<Return value, data structure element> (Received data packet; an array of up to 200 entries of 1 byte each)
– (Not available)	<Return value, dataLength structure element> (Data length of received data package)
<Return value> (= 0, if no error occurred; <> 0, if error occurred; see SIMATIC S7 documentation)	<Return value, functionResult structure element> (= 0, if no error occurred; <> 0, if error occurred; see Parameter Manual for the SIMOTION devices)

Example of send and receive program

The following figures show the source text for the send and receive program.

Table 4-114 Example of send program

```
INTERFACE
    PROGRAM xsend_control;
END_INTERFACE

IMPLEMENTATION
                          // The following program must be assigned
                          // to a MotionTask.
                          // In the task configuration, the "Activation
                          // after Startup Task" option must be selected.
PROGRAM xsend_control
VAR
    retVal                          : DINT;
    myAddress                       : StructXSendDestAddr;
    myStaddr                        : ARRAY[0..4] OF BYTE;
    myData                          : ARRAY[0..199] OF BYTE;
END_VAR
                          // Specify destination address and PROFIBUS interface ////////
myAddress.deviceID                  := 1;
                          // PROFIBUS interface X8
myAddress.remoteStaddrLength        := 1;
                          // must always be assigned a value of 1
myAddress.remoteStaddr[0]           := 4;
                          // PROFIBUS address of the receiving station

                          // Send data
myData[0] := 170;

                          // Call of send function
retVal := _Xsend( communicationMode := ABORT_CONNECTION
             , address              := myAddress
             , messageId            := 1
             , nextCommand          := WHEN_COMMAND_DONE
             , commandId            := _getCommandId()
             , data                 := myData
             , dataLength           := 1
             );
END_PROGRAM
END_IMPLEMENTATION
```

Table 4-115 Example of receive program

```
INTERFACE
    PROGRAM xreceive_control;
END_INTERFACE

IMPLEMENTATION
VAR_GLOBAL
    retVal                          : StructRetXReceive;
END_VAR
                          // The following program must be assigned
                          // to a MotionTask.
                          // In the task configuration, the "Activation
                          // after Startup Task" option must be selected.
PROGRAM xreceive_control

// Call of receive function
retVal := _Xreceive( messageId       := 1
            , nextCommand            := WHEN_COMMAND_DONE
            , commandId              := _getCommandId()
            );
END_PROGRAM
END_IMPLEMENTATION
```

Communication via Ethernet with TCP/IP protocol

For SIMOTION devices with Ethernet interface, communication via the TCP/IP protocol is also possible.

The following table lists the individual steps to establish communication between a sender (client) and a receiver (server):

Table 4-116 Communication steps of a TCP/IP connection and the corresponding system functions

Communication steps		System function
Establish the connection		
1	Receiver (server) waits for communication request.	_tcpOpenServer
2	Sender (client) requests connection establishment to the receiver.	_tcpOpenClient
3	Receiver (server) has established communication request.	
Communicating		
4	Sender sends data to the receiver.	_tcpSend
5	Receiver receives data from the sender.	_tcpReceive
Terminating communication connection		
6	Sender does not send any more data and closes the connection.	_tcpCloseConnection
7	There is no further connection required.	_tcpCloseServer

The system functions are described in detail in the List Manual for the SIMOTION devices, in the chapter on system functions.

Note

A sender or receiver can be a client as well as a server when establishing a connection.

There must be at least one client and one server to establish TCP/IP connection.

The client-server relationship is only valid until the connection is established. After connection is established, both communication peers are equivalent, i.e. each of the two can send or receive or close the connection at any point of time.

For detailed information, please refer to the *Frequently Asked Questions* (FAQs) on *SIMOTION Utilities & Applications* (DVD2 - CD_14).

Communication via Ethernet with UDP protocol

For SIMOTION devices with Ethernet interface, communication via the UDP protocol is also possible.

- The *_udpSend* function sends a data packet to a communication peer which is specified with IP address and port number.
  The data to be sent is transferred to the function as ARRAY [0..1399] OF BYTE.
- The *_udpReceive* function is used to receive a data packet which a communication peer has sent with *_udpSend*.
  The received data is stored in a variable of data type ARRAY [0..1399] OF BYTE, whose identifier is transferred to the function as parameter.

The system functions are described in detail in the Parameter Manual for the SIMOTION devices, in the chapter on system functions.

A detailed description is available in the FAQs on *SIMOTION Utilities & Applications* (DVD2 - CD_14).

Acyclic communication with the drive

Overview

PROFIdrive drive units are supplied with control signals and setpoints by the controller and return status signals and actual values. These signals are normally transferred cyclically (i.e. continuously) between the controller and the drive.

For SINAMICS S110/S120, configure the axis message frames for data exchange (see Setting up addresses and message frames - overview (Page 1211)).

As well as offering cyclic data exchange, PROFIdrive drive units have an acyclic communication channel. In particular, this is used for reading and writing drive parameters (e.g. error codes, warnings, controller parameters, motor data, etc.).

As a result, data can be transferred on an "acyclic" as opposed to a "cyclic" basis when required. Acyclic reading and writing of parameters for PROFIdrive drives is based on the DP V1 services "Read data set" and "Write data set".

The acyclic DP V1 services are transferred in parallel to the cyclic communication via PROFIBUS or PROFINET. The PROFIdrive profile specifies precisely how these basic mechanisms are used for read/write access to parameters of a PROFIdrive-compliant drive.

The PROFIdrive standard states that "pipelining" of jobs on PROFIdrive drives is not supported. This means:

- Only one "Write/read data set" can be performed at any one time on a drive unit (e.g. SINAMICS S110/S120 control unit or the SINAMICS Integrated of a SIMOTION D).
- However, if several PROFIdrive drive units are connected to a controller, a job can be processed for each of these drive units at the same time. In this case, the maximum total number of jobs will depend on the controller (maximum of eight jobs simultaneously with SIMOTION).

For acyclic data exchange with SINAMICS drives, this means you will have to coordinate the write/read jobs with each other (buffer management). An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive unit.

Additional references

Additional information on how to use DP V1 services can be found in the *SIMOTION Communication* System Manual.

SIMOTION Utilities & Applications also has a DP V1 library with functions that are capable of performing coordination tasks commonly associated with acyclic communication. The library not only coordinates access to the system functions (_ReadRecord/_WriteRecord/_readDriveParameter/_writeDriveParameter/, etc.), but also expands the range of functions for frequently required tasks, e.g. the reading of faults and alarms from the drive unit.

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

The following functions are available in the DP V1 library:

- Buffer management (coordination of a number of parallel DP V1 services)
- StartUp (function for coordinating the power-up of the SINAMICS drive with SIMOTION)
- TimeSync (applicative time synchronization: Transfer of SIMOTION time of day to the SINAMICS drives)
- SetActIn (activating/deactivating objects in SIMOTION and SINAMICS)
- RwnPar (reading and writing of drive parameters)
- GetFault (reading errors and warnings from the drive)

See also

Available functions (Page 1596)

Synchronous start

On synchronous start, several commands are started within an IPO cycle clock or IPO_2 cycle clock.

Follow these steps:

1. First, obtain a unique SyncCommandId from the system. You will need this SyncCommandId as a unique designation for the synchronized commands.
   For this purpose, use the _getSyncCommandId() system function.
2. Enclose the commands to be executed synchronously between the BEGIN_SYNC(SyncCommandId) and END_SYNC() functions. This defines them for synchronized start.
   All motion commands are permissible within the structure BEGIN_SYNC/END_SYNC. The value IMMEDIATELY must be transferred as the nextCommand parameter.
3. The synchronous start itself occurs with the _startSyncCommand (SyncCommandId) function. The commands defined for the synchronized start are processed in parallel.

See sample program.

Note

Synchronous start for motion commands is only ensured if the following conditions are satisfied:

1. Commands included in a synchronous start must act on various technology objects.
2. The technology objects involved must be in the same execution level (IPO or IPO_2).
3. When you use the Synchronous Start command in MCC, you must generate the UserInterruptTask_1 since this is called in the event of an error. You can program an error response in this task, see UserInterruptTasks (Page 1338).

The task controller is temporarily shut down during the synchronous start. It is not restarted until the following conditions are met:

- All single-axis commands and synchronous operation and cam commands in the branches have been started
- All basic commands in the branches have been completed

Start interruptions by other tasks (except SynchronousTasks) are thus prevented. This can cause a timeout to occur in cyclic tasks (BackgroundTask, TimerInterruptTasks). This error can be detected and caught by programming the TimeFaultBackgroundTask or TimeFaultTask appropriately.

The BEGIN_SYNC, END_SYNC, and *_startSyncCommand* functions are system functions of SIMOTION devices; for more information, refer to the parameter manual for the relevant SIMOTION device.

Note

Synchronous start is often used in conjunction with the WAITFORCONDITION construct. In this case, the system waits for a condition to be fulfilled before the synchronized start. The *_startSyncCommand* function must not occur within the WAITFORCONDITION construct.

Table 4-117 Sample program for synchronized start of two axes with WAITFORCONDITION

```
INTERFACE
    USEPACKAGE cam;
    PROGRAM sync_motion;
END_INTERFACE

IMPLEMENTATION

    EXPRESSION wait_sync_expression
        wait_sync_expression := TRUE;
    END_EXPRESSION

    PROGRAM sync_motion
        VAR
            ret_val : DINT;
            sync_id : CommandIdType;
        END_VAR

        sync_id := _getSyncCommandId();
        BEGIN_SYNC(sync_id);
        (* Positioning axis ('Pos') *)
        ret_val := _pos    (axis := Axis_1,
                    positioningMode := ABSOLUTE,
                    position := 100,
                    mergeMode := IMMEDIATELY,
                    nextCommand := IMMEDIATELY,
                    commandId := _getCommandId() );
        (* Positioning axis ('Pos') *)
        ret_val := _pos    (axis := Axis_2,
                    positioningMode := ABSOLUTE
                    position := 50
                    mergeMode := IMMEDIATELY
                    nextCommand := IMMEDIATELY,
                    commandId := _getCommandId() );
        END_SYNC();

        WAITFORCONDITION wait_sync_expression DO
            ;
        END_WAITFORCONDITION;

        ret_val := _startSyncCommands(sync_id);
        // Here in the event of an error Start of the UserInterruptTask
        IF (_ret_val <> 0) THEN
            _startTaskId(_task.UserInterruptTask_1);
        END_IF;

    END_PROGRAM
END_IMPLEMENTATION
```

To check the correct completion of the synchronous start, you can query whether the commands of the two axes started in the previous example have been completed without error.

The following MCC sample program shows a synchronous start with two axes and subsequent query of the group variable _MccRetSyncStart. This is not equal to 0 when a command within the synchronous start had a return value not equal to 0.

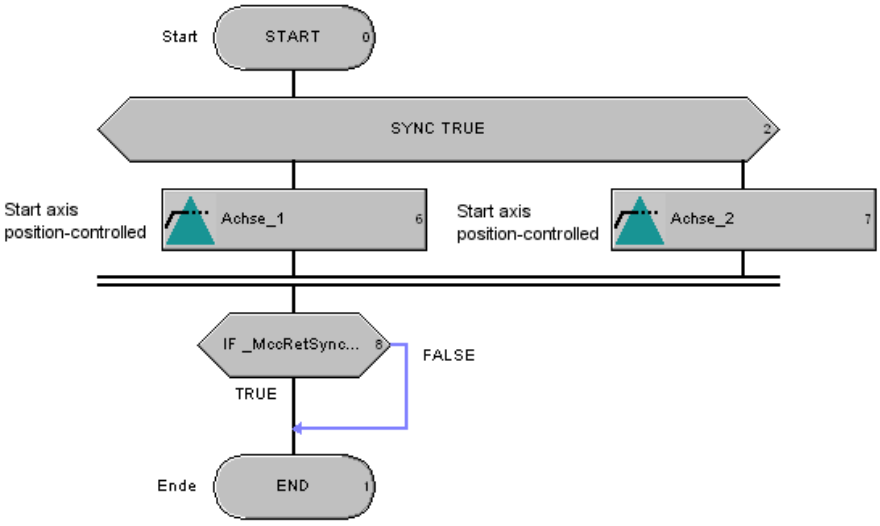


Figure 4-162 MCC synchronous start

The following is the section from the appropriate ST program (generated here via export from MCC unit)

Table 4-118 ST synchronous start

```
(* Synchronous start ('StartSync') *)
_MccCommand1 := _getCommandId();
_MccCommand2 := _getCommandId();

_MccSync := _getSyncCommandId();
BEGIN_SYNC(_MccSync);
    (* Start axis position-controlled ('Move') *)
    _MccRetDINT_1:=_move(axis:=Achse_1, velocityType:=DIRECT,
        velocity:=v_blue, moveTimeOutType:=WITHOUT_TIME_LIMIT,
        mergeMode:=IMMEDIATELY, nextCommand:=IMMEDIATELY,
        commandId:=_MccCommand1, movingMode:=POSITION_CONTROLLED);
    (* Start axis position-controlled ('Move') *)
    _MccRetDINT_2:=_move(axis:=Achse_2, velocityType:=DIRECT,
        velocity:=v_red, moveTimeOutType:=WITHOUT_TIME_LIMIT,
        mergeMode:=IMMEDIATELY, nextCommand:=IMMEDIATELY,
        ommandId:=_MccCommand2, movingMode:=POSITION_CONTROLLED);
(* Synchronous start ('EndSync') *)
END_SYNC();

WAITFORCONDITION _MccMCC_1Condition1 DO
    ;
END_WAITFORCONDITION;

_MccRetDINT := _startSyncCommands(_MccSync);
// Start of the UserInterruptTask
IF (_MccRetDINT <> 0) THEN
    _startTaskId(_task.UserInterruptTask_1);
END_IF;

_MccRetStructRetMotionCommandState := _getMotionStateOfAxisCommand
    (Achse_1, _MccCommand1);

// Query whether both axis commands are completed

WHILE ((_MccRetStructRetMotionCommandState.functionResult = 0 )
   AND (_MccRetStructRetMotionCommandState.motionCommandIdState <>
   IN_CONSTANT_MOTION) AND
   (_MccRetStructRetMotionCommandState.motionCommandIdState <>
    NOT_EXISTENT)) DO
    _MccRetDINT := _waitTime(T#0d_0h_0m_0s_0ms);
    _MccRetStructRetMotionCommandState :=
    _getMotionStateOfAxisCommand(Achse_1, _MccCommand1);
END_WHILE;

_MccRetStructRetMotionCommandState :=
_getMotionStateOfAxisCommand(Achse_2, _MccCommand2);

WHILE ((_MccRetStructRetMotionCommandState.functionResult = 0 )
   AND (_MccRetStructRetMotionCommandState.motionCommandIdState <>
   IN_CONSTANT_MOTION) AND
   (_MccRetStructRetMotionCommandState.motionCommandIdState <>
   NOT_EXISTENT)) DO
   _MccRetDINT := _waitTime(T#0d_0h_0m_0s_0ms);
   _MccRetStructRetMotionCommandState :=
```

```
   _getMotionStateOfAxisCommand(Achse_2, _MccCommand2);
END_WHILE;

_MccRetSyncStart := 0; // Query whether both axis commands are running without error
IF (_MccRetDINT_1 + _MccRetDINT_2 <> 0) THEN
    _MccRetSyncStart := -1;
END_IF;

(* IF: Program branch ('If') *)
IF (_MccRetSyncStart <> 0) THEN
    ;// Error handling
(*  ('Else') *)
ELSE
    ;
(*  ('EndIf') *)
END_IF;
```

4.2.9.25 HMI (Human Machine Interface) connection

Interface between HMI and SCOUT or SIMOTION

SIMOTION allows communication with HMI devices (operator devices for the end user), e.g. with operator panels.

If one or more SIMOTION devices are connected on the PROFIBUS or PROFINET, the HMI device can display variables, messages and alarms of the SIMOTION devices. Likewise, you can initiate functions programmed using the HMI device and a program in the SIMOTION device.

The WinCC flexible software package is available for implementing such tasks on the HMI device. You can use WinCC flexible to read and write to system variables and unit variables declared in the interface section. See also HMI variables in one unit (Page 1690).

Note

If you would like to use a SIMOTION module as DP slave on an HMI, you must make the following settings in HW Config for the object properties of the relevant DP interface:

1. Select the **Operating mode** tab.
2. Select **DP Slave** as the setting.
3. Activate the Option **Programming, status/control or other PG functions and non-configured communication connection possible**.

SIMOTION also provides an OPC server. You can use this server to access process data with non-proprietary HMI software. The configuration software for WinCC flexible allows direct access to the symbolic variables of a SIMOTION device. Likewise, this is possible via OPC. For this purpose,

the configuration data of WinCC flexible must be located in the same project as the SIMOTION devices.

Note

- Variables that are meant to be available in HMI must always be created as unit variables in the interface section of a source.
- HMI tags can only be linked to SIMOTION variables of a simple data type (e.g. DINT). Other variables such as ANYOBJECT or configuration data cannot be linked.

Further information about WinCC flexible

Notes on the configuration with WinCC flexible can be found:

- In the online help for WinCC flexible
- In the online help for SCOUT when WinCC flexible is integrated in SCOUT

Further FAQs are also available online at:

http://support.automation.siemens.com/WW/view/en/28767398

http://support.automation.siemens.com/WW/view/en/23751257

Additional information on the topics below is available in the FAQs in SIMOTION Utilities & Applications:

- Time synchronization SIMOTION
- WinCC/WinCCFlexible
- SIMOTION and HMI (parallel work on SIMOTION and HMI projects)

Examples of HMI configurations can be found among the standard applications contained within the Applications section on SIMOTION Utilities & Applications.

- Examples of HMI implementation

See also HMI variables in one unit (Page 1690).

Alarms

Everything that is displayed in the alarm list in SCOUT can also be displayed in WinCC flexible with the appropriate configuration, except SINAMICS alarms!

All SIMOTION and SINAMICS alarms function with OPC Alarm&Event.

The present mechanism is as follows:

When the HMI project is generated, the SCOUT texts (axis names, axis alarms, Alarm_S, etc.) are synchronized with the HMI, whereby each technology object has a number.

Therefore, when an alarm occurs the TO number, alarm number, coefficient, etc. are signaled and the HMI then writes the TO number, alarm text with coefficient, time stamp, etc.

With WinCC flexible, warning and error messages of the technology objects (axes, output cams, measuring inputs, etc.) are output directly on the HMI and are acknowledged by the operator. The AlarmS concept also provides a method for configuring and programming user messages. User messages are configured in SIMOTION SCOUT; messages are called and acknowledged

during runtime by calling the appropriate system commands. OPC supports the same message mechanisms. For the OPC export, an additional OPC alarm/event must be activated.

Consistent data access with HMI devices (example)

HMI devices can also access user data of the SIMOTION device consistently. The following example includes an ST program on the SIMOTION device (see example) and an application (Visual Basic program) on the HMI device (see example).

The user requests consistent read access to user variables (including arrays) from within the HMI application (in this case, Visual Basic; but it could also be Visual C++ ...). An uneven value is written to the consistencyFlag variable. The SIMOTION user program then copies the data. The HMI application waits until the SIMOTION user program confirms that it has completed copying the data by writing an even value to the *consistencyFlag* variable. Then, the HMI reads out this variable consistently, since no changes have been made in the meantime.

The ST program can be assigned to a cyclic task; it may be necessary to take the IPOsynchronousTask if, for example, axis positions and velocities are to be read at the same time (as a snapshot).

Examples:

Table 4-119 ST program for consistent data access from HMI devices

```
INTERFACE
    VAR_GLOBAL
        consistencyFlag : DINT;
        myDint    : DINT;
        myArray : ARRAY[0..10] OF LREAL;
    END_VAR
    PROGRAM OPC_Prog;
END_INTERFACE

IMPLEMENTATION
    PROGRAM OPC_Prog
        IF (consistencyFlag MOD 2) = 1 THEN
            myDint := 99;
            myArray[0] := 0.0;
            myArray[1] := 1.0;
            myArray[10] := 10.0;
            consistencyFlag := consistencyFlag + 1;
        END_IF;
    END_PROGRAM
END_IMPLEMENTATION
```

Table 4-120 HMI application for consistent data access (Visual Basic)

```
Option Explicit
Option Base 1

    Dim g_Server As OPCServer
    Dim g_GroupObj As OPCGroup

    Dim g_myItem1 As OPCItem
    Dim g_myItem2 As OPCItem

    Dim g_myItem3 As OPCItem
    Const OPC_DS_DEVICE = 2

    Dim consistencyFlag As Long

Private Sub Form_Load()
    Set g_Server = New OPCServer
    g_Server.Connect ("OPC.SimaticNet")
    Set g_GroupObj = g_Server.OPCGroups.Add("Test1")
    g_GroupObj.IsActive = False
    Set g_myItem1 = g_GroupObj.OPCItems.AddItem("C240.consistencyFlag", 2)
    Set g_myItem2 = g_GroupObj.OPCItems.AddItem("C240.myDint", 2)
    Set g_myItem3 = g_GroupObj.OPCItems.AddItem("C250.myArray", 3)
    consistencyFlag = 1
End Sub
Private Sub Form_Unload(Cancel As Integer)
    Set g_myItem1 = Nothing
    Set g_myItem2 = Nothing
    Set g_myItem3 = Nothing
    Set g_GroupObj = Nothing
    Set g_Server = Nothing
    End
End Sub

Private Sub cmdRead_Click()
    Static reentrancyFlag As Boolean
    Dim var1 As Variant
    Dim var2 As Variant
    Dim var3 As Variant

    If reentrancyFlag = False Then
        reentrancyFlag = True
        consistencyFlag = consistencyFlag + 2
        g_myItem1.Write consistencyFlag
    Do
        g_myItem1.Read OPC_DS_DEVICE, var1
    Loop Until var1 = consistencyFlag + 1
        g_myItem2.Read OPC_DS_DEVICE, var2
        g_myItem3.Read OPC_DS_DEVICE, var3
        reentrancyFlag = False
    End if
End Sub
```

4.2.10 Programming of general system function blocks

4.2.10.1 Overview of the function blocks

ST contains a series of system function blocks that you can use in your ST source files without having to declare them beforehand. You only have to create an instance and assign the necessary parameters.

Overview of system function blocks

Below is an overview of all system function blocks implemented in the system. You will find definitions and a further specification in the following sections.

Table 4-121 System function blocks

Collective term	Name	Input parameter		Output parameter	
	Definition	Name	: Type	Name	: Type
Bistable function blocks (Page 1617)	**SR**[1] Priority set	S1 R	: BOOL; : BOOL;	Q1	: BOOL;
	RS[1] Priority reset	SR1	: BOOL; : BOOL;	Q1	: BOOL;
Edge detection (Page 1619)	**R_TRIG**[1] Rising edge	CLK	: BOOL;	Q	: BOOL;
	F_TRIG[1] Falling edge	CLK	: BOOL;	Q	: BOOL;

Collective term	Name	Input parameter		Output parameter	
	Definition	Name	: Type	Name	: Type
Counters (Page 1622)	**CTU**[1]	CU	: BOOL;	Q	: BOOL;
	Up counter	R	: BOOL;	CV	: INT;
	Data type: INT	PV	: INT;		
	CTU_DINT[1]	CU	: BOOL;	Q	: BOOL;
	Up counter	R	: BOOL;	CV	: DINT;
	Data type: DINT	PV	: DINT;		
	CTU_UDINT[1]	CU	: BOOL;	Q	: BOOL;
	Up counter	R	: BOOL;	CV	: UDINT;
	Data type: UDINT	PV	: UDINT;		
	CTD[1]	CD	: BOOL;	Q	: BOOL;
	Down counter	LD	: BOOL;	CV	: INT;
	Data type: INT	PV	: INT;		
	CTD_DINT[1]	CD	: BOOL;	Q	: BOOL;
	Down counter	LD	: BOOL;	CV	: DINT;
	Data type: DINT	PV	: DINT;		
	CTD_UDINT[1]	CD	: BOOL;	Q	: BOOL;
	Down counter	LD	: BOOL;	CV	: UDINT;
	Data type: UDINT	PV	: UDINT;		
	CTUD[1]	CU	: BOOL;	QU	: BOOL;
	Up/down counter	CD	: BOOL;	QD	: BOOL;
	Data type: INT	R	: BOOL;	CV	: INT;
		LD	: BOOL;		
		PV	: INT;		
	CTUD_DINT[1]	CU	: BOOL;	QU	: BOOL;
	Up/down counter	CD	: BOOL;	QD	: BOOL;
	Data type: DINT	R	: BOOL;	CV	: UINT;
		LD	: BOOL;		
		PV	: DINT;		
	CTUD_UDINT[1]	CU	: BOOL;	QU	: BOOL;
	Up/down counter	CD	: BOOL;	QD	: BOOL;
	Data type: UDINT	R	: BOOL;	CV	: UDINT;
		LD	: BOOL;		
		PV	: UDINT;		
Timers (Page 1627)	**TP**[1]	IN	: BOOL;	Q	: BOOL;
	Pulse	PT	: TIME;	ET	: TIME;
	TON[1]	IN	: BOOL;	Q	: BOOL;
	ON delay	PT	: TIME;	ET	: TIME;
	TOF[1]	IN	: BOOL;	Q	: BOOL;
	ON delay	PT	: TIME;	ET	: TIME;
	RTC[1]	SET	: BOOL;	CDT	: DT;
	Real-time clock	READ	: BOOL;		
		PDT	: DT;		

Collective term	Name	Input parameter		Output parameter	
	Definition	Name	: Type	Name	: Type
Splitting bit-string data types (Page 1630)	**_BYTE_TO_8BOOL**	n	: BYTE	bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7	: BOOL;
	_WORD_TO_2BYTE	in	: WORD	byte0, byte1	: BYTE;
	_DWORD_TO_2WORD	in	: DWORD	word0, word1	: WORD;
	_DWORD_TO_4BYTE	in	: DWORD	byte0, byte1, byte2, byte3	: DWORD;
Emulation of SIMATIC S7 commands (Page 1634)	**_S7_COUNTER**[1]	CU CD PV S R	: BOOL; : BOOL; : INT; : BOOL; : BOOL;	Q CV	: BOOL; : INT;
	_S7_TIMER[1]	T_Type PR S R TV TV_BCD	: WORD; : BOOL; : BOOL; : BOOL; : TIME; : WORD;	Q BI	: BOOL; : TIME;
[1] These function blocks require a repeated call (e.g. in a cyclic task).					

Comparison of the system function blocks for SIMOTION and SIMATIC

You can find a comparison of the SIMATIC S7 and SIMOTION system function blocks in the 2_FAQ directory on the Utilities & Applications CD.

4.2.10.2 Bistable elements (set flip-flop)

Description

In ST you can use system function block SR to set/reset the flip-flop (priority set) and system function block RS to reset/set the flip-flop (priority reset).

Bistable function block SR (priority set)

The stored value can be obtained at output Q1. Output Q1 is 1 when S1 is 1. If S1 and R are equal to 0, output Q1 does not change.

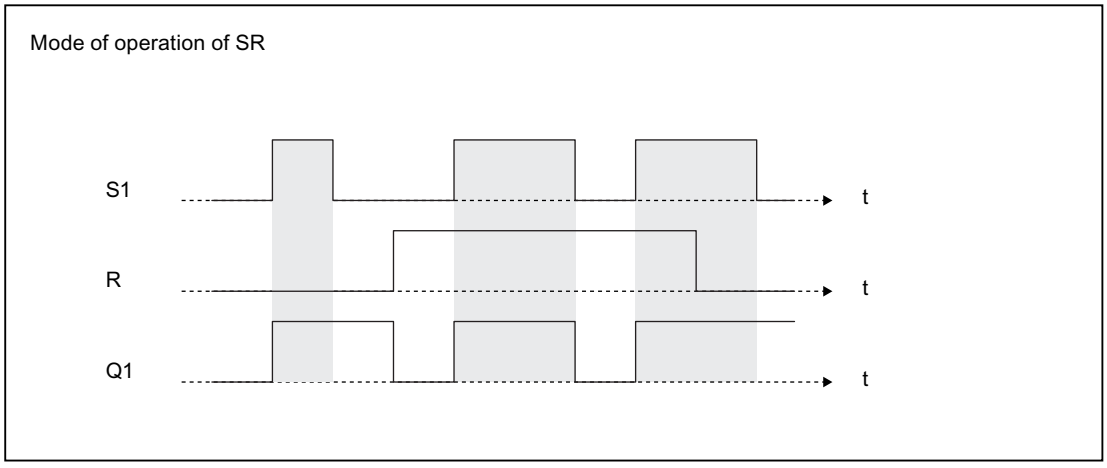


Figure 4-163 Mode of operation of the SR bistable function block

Program code of the SR bistable function block

```
FUNCTION_BLOCK SR

    VAR_INPUT
        S1, R    : BOOL;
    END_VAR

    VAR_OUTPUT
        Q1       : BOOL;
    END_VAR

    Q1 := S1 OR (NOT R AND Q1);

END_FUNCTION_BLOCK
```

Table 4-122 Call parameters for SR

Identifier	Parameter	Data type	Description
S1	Input	BOOL	Set
R	Input	BOOL	Reset
Q1	Output	BOOL	Signal state

Bistable function block RS (priority reset)

The stored value can be obtained at output Q1. Output Q1 is 0 when R1 is 1. If R1 and S are equal to 0, output Q1 does not change.

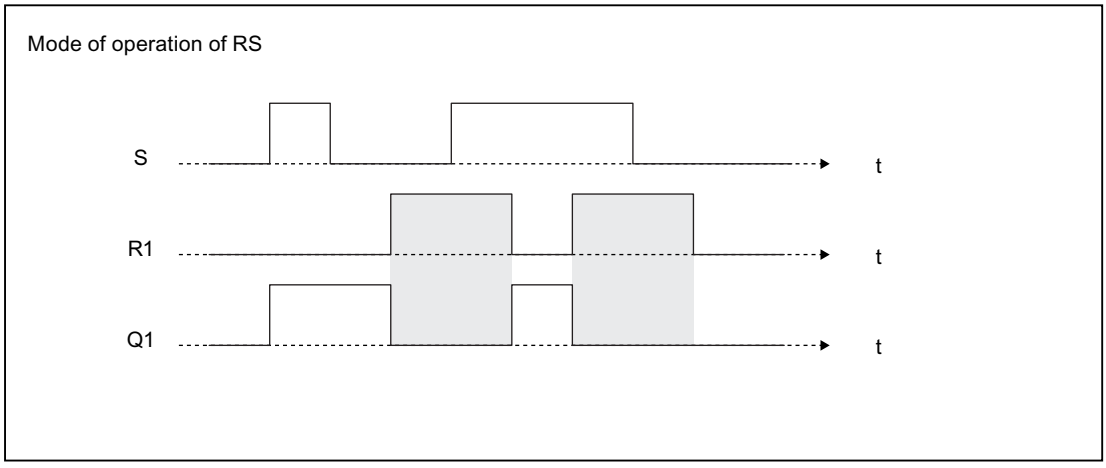


Figure 4-164 Mode of operation of the RS bistable function block

Program code of RS bistable function block

```
FUNCTION_BLOCK RS

    VAR_INPUT
        R1, S      : BOOL;
    END_VAR

    VAR_OUTPUT
        Q1         : BOOL;
    END_VAR

    Q1 := NOT R1 AND (S OR Q1);

END_FUNCTION_BLOCK
```

Table 4-123 Call parameters for RS

Identifier	Parameter	Data type	Description
S	Input	BOOL	Set
R1	Input	BOOL	Reset
Q1	Output	BOOL	Signal state

4.2.10.3 Edge detection

Description

System function block R_TRIG can be used to detect a rising edge; F_TRIG can detect a falling edge. You can use this function, for example, to set up a sequence of your own function blocks.

Detection of rising edge R_TRIG

If a rising edge (R_TRIG, Rising Trigger), i.e. a status change from 0 to 1 is present at the input, 1 is applied at the output for the duration of one cycle.

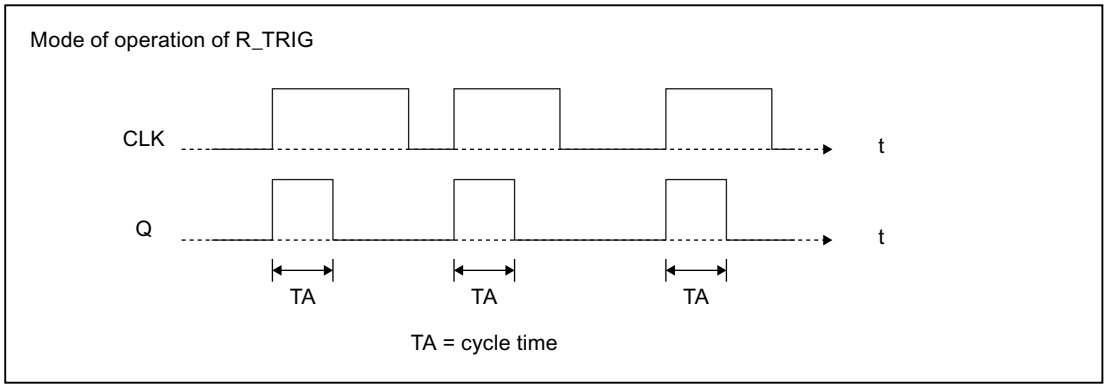


Figure 4-165 Mode of operation of R_TRIG (rising edge) function block

Program code for function block R_TRIG (rising edge)

```
FUNCTION_BLOCK R_TRIG

    VAR_INPUT
        CLK     : BOOL;
    END_VAR

    VAR_OUTPUT
        Q       : BOOL;
    END_VAR

    VAR
        M       : BOOL := 0;
    END_VAR

    Q := CLK AND NOT M;
    M := CLK;

END_FUNCTION_BLOCK
```

Table 4-124 Call parameters for R_TRIG

Identifier	Parameter	Data type	Description
CLK	Input	BOOL	Input for edge detection
Q	Output	BOOL	Status of edge

Detection of falling edge F_TRIG

When a falling edge (F_TRIG, falling trigger), i.e. a status change from 1 to 0, occurs at the input, the output is set to 1 for the duration of one cycle time.

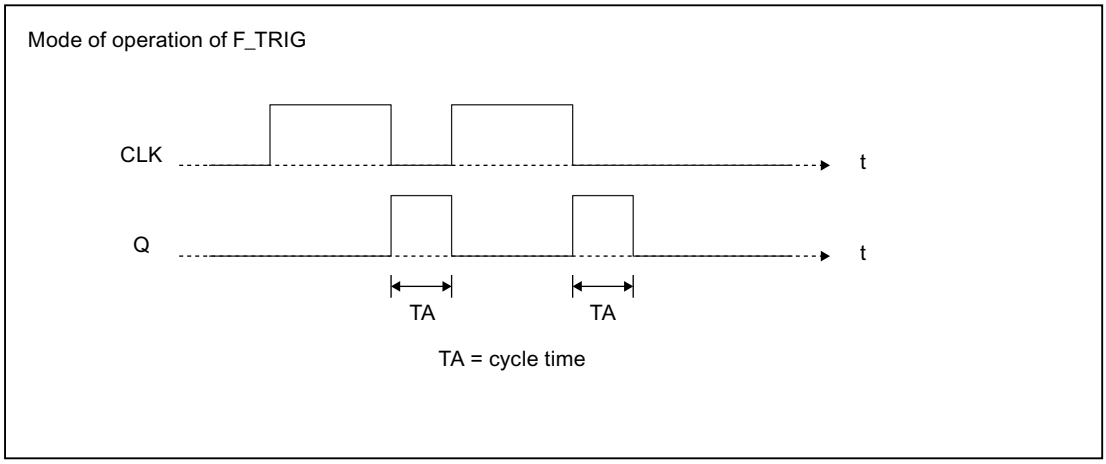


Figure 4-166 Mode of operation of F_TRIG (falling edge) function block

Program code for function block F_TRIG (falling edge)

```
FUNCTION_BLOCK F_TRIG

    VAR_INPUT
        CLK  : BOOL;
    END_VAR

    VAR_OUTPUT
        Q    : BOOL;
    END_VAR

    VAR
        M    : BOOL    := 1;
    END_VAR

    Q := NOT CLK AND NOT M;
    M := NOT CLK;

END_FUNCTION_BLOCK
```

Table 4-125 Call parameters for F_TRIG

Identifier	Parameter	Data type	Description
CLK	Input	BOOL	Input for edge detection
Q	Output	BOOL	Status of edge

4.2.10.4 Counters

General information on counters

ST provides a series of system function blocks for counters that you can use in your ST program.

CTU up counter

Description

The CTU counter allows you to perform upward counting operations:

- If the input is R = TRUE when the FB is called up, then the CV output is reset to 0.
- If the CU input changes from FALSE to TRUE (0 to 1) when the FB is called (rising edge), then the CV output is incremented by 1.
- Output Q specifies whether CV is greater than or equal to comparison value PV.

The CV and PV parameters are both INT data types, which means that the maximum counter value possible is 32_767 (= 16#7FFF).

User interface prototype

```
FUNCTION_BLOCK CTU
    VAR_INPUT
        CU    : BOOL;        // Count:
        R     : BOOL;        // Reset
        PV    : INT;         // Comparison value
    END_VAR

    VAR_OUTPUT
        Q     : BOOL;        // Status
        CV    : INT;         // Counter value
    END_VAR

    // ... (Code)

END_FUNCTION_BLOCK
```

Input parameter

CU

Data type: BOOL

Count up if value changes from FALSE to TRUE (rising edge)

R

Data type: BOOL

TRUE: Reset the counter to 0

PV

Data type: INT

Comparison value

Output parameter

Q

Data type: BOOL

Status of counter (CV ≥ PV)

CV

Data type: INT

Counter value

CTU_DINT up counter

The method of operation is the same as for the CTU up counter (Page 1622) except for the following:

The CV and PV parameters are both DINT data types, which means that the maximum counter value possible is 2_147_483_647 (= 16#7FFF_FFFF).

Table 4-126 Parameters for CTU_DINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (rising edge)
R	Input	BOOL	Reset the counter to 0
PV	Input	DINT	Comparison value
Q	Output	BOOL	Status of counter (CV ≥ PV)
CV	Output	DINT	Counter value

CTU_UDINT up counter

The method of operation is the same as for the CTU up counter (Page 1622) except for the following:

The CV and PV parameters are both UDINT data types, which means that the maximum counter value possible is 4_294_967_295 (=16#FFFF_FFFF).

Table 4-127 Parameters for CTU_UDINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (rising edge)
R	Input	BOOL	Reset the counter to 0
PV	Input	UDINT	Comparison value

Identifier	Parameter	Data type	Description
Q	Output	BOOL	Status of counter (CV ≥ PV)
CV	Output	UDINT	Counter value

CTD down counter

The CTD counter allows you to perform downward counting operations.

- If the LD input = TRUE when the FB is called, then the CV output is reset to start value PV.
- If the CD input changes from FALSE to TRUE (from 0 to 1) when the FB is called (= positive edge), then the CV output is decremented by 1.
- Output Q specifies whether CV is less than or equal to 0.

The CV and PV parameters are both INT data types, which means that the minimum counter value possible is –32_768 (= 16#8000).

Table 4-128 Call parameters for CTD

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	INT	Start value of counter
Q	Output	BOOL	Status of counter (CV ≤ 0)
CV	Output	INT	Counter value

CTD_DINT down counter

The method of operation is the same as for the CTD down counter (Page 1624) except for the following:

The CV and PV parameters are both DINT data types which means that the minimum counter value possible is –2_147_483_648 (= 16#8000_0000).

Table 4-129 Call parameters for CTD_DINT

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	DINT	Start value of counter
Q	Output	BOOL	Status of counter (CV ≤ 0)
CV	Output	DINT	Counter value

CTD_UDINT down counter

The method of operation is the same as for the CTD down counter (Page 1624) except for the following:

The CV and PV parameters are both UDINT data types, which means that the minimum counter value possible is 0.

Table 4-130 Call parameters for CTD_UDINT

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	UDINT	Start value of counter
Q	Output	BOOL	Status of counter (CV = 0)
CV	Output	UDINT	Counter value

CTUD up/down counter

The CTUD counter allows you to perform both upward and downward counting operations.

- Reset the CV count variable:
  - If the input is R = TRUE when the FB is called up, then the CV output is reset to 0.
  - If the LD input = TRUE when the FB is called, then the CV output is reset to start value PV.
- Count:
  - If the CU input changes from FALSE to TRUE (0 to 1) when the FB is called (rising edge), then the CV output is incremented by 1.
  - If the CD input changes from FALSE to TRUE (from 0 to 1) when the FB is called (= rising edge), then the CV output is decremented by 1.
- Counter status QU or QD:
  - Output Q specifies whether CV is greater than or equal to comparison value PV.
  - Output QD specifies whether CV is less than or equal to 0.

Table 4-131 Parameters for CTUD

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (rising edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
R	Input	BOOL	Reset the counter to 0 (up counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	INT	Comparison value (for up counter) Start value (for down counter)
QU	Output	BOOL	Status as up counter (CV ≥ PV)
QD	Output	BOOL	Status as down counter (CV ≤ 0)
CV	Output	INT	Counter value

CTUD_DINT up/down counter

The method of operation is the same as for the CTUD up/down counter (Page 1625) except for the following:

The CV and PV parameters are both DINT data types.

Table 4-132 Parameters for CTU_DINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (rising edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
R	Input	BOOL	Reset the counter to 0 (down counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	DINT	Comparison value (for up counter) Start value (for down counter)
QU	Output	BOOL	Status as up counter (CV ≥ PV)
QD	Output	BOOL	Status as down counter (CV ≤ 0)
CV	Output	DINT	Counter value

CTUD_UDINT up/down counter

The method of operation is the same as for the CTUD up/down counter (Page 1625) except for the following:

The CV and PV parameters are both UDINT data types.

Table 4-133 Parameters for CTU_UDINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (rising edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (rising edge)
R	Input	BOOL	Reset the counter to 0 (up counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	UDINT	Comparison value (for up counter) Start value (for down counter)
QU	Output	BOOL	Status as up counter (CV ≥ PV)
QD	Output	BOOL	Status as down counter (CV = 0)
CV	Output	UDINT	Counter value

4.2.10.5 Timers

Description

Timers are elements in your program used to execute and monitor time-driven processes. ST provides a series of system function blocks that you can access with ST. You can use timers in your program to do the following:

- Set waiting times
- Enable monitoring times
- Generate pulses
- Measure times

TP pulse

With a signal state change from 0 to 1 at the IN input, time ET is started. Output Q remains at 1 until elapsed time ET is equal to programmed time value PT. As long as time ET is running, the IN input has no effect.

The following figure illustrates the mode of operation of the TP pulse timer.

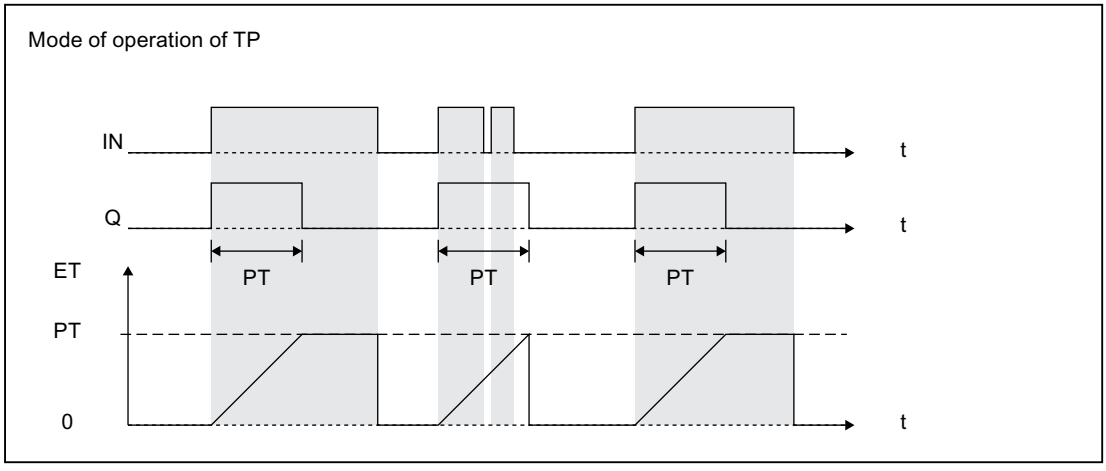


Figure 4-167 Mode of operation of the TP pulse

The following table shows the call parameters.

Table 4-134 Call parameters for TP

Identifier	Parameter	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of pulse
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

TON ON delay

With the signal state change from 0 to 1 at the IN input, time ET is started. The output signal Q only changes from 0 to 1 if the time ET = PT has elapsed and the input signal IN still has a value of 1, i.e. the output Q is switched on with a delay. Input signals of shorter durations than that of programmed time PT do not appear at the output.

The following figure illustrates the mode of operation of the TON ON delay timer.

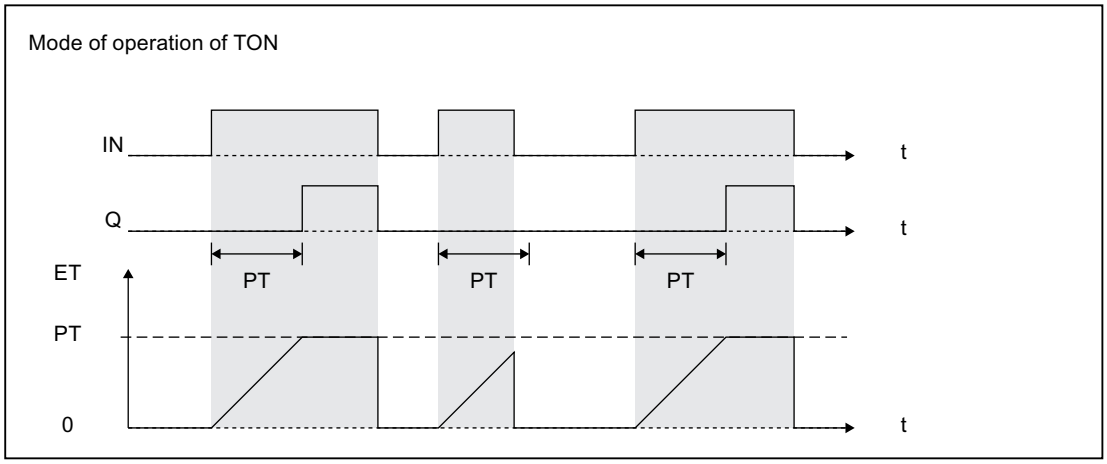


Figure 4-168 Mode of operation of the TON ON delay

The following table shows the call parameters.

Table 4-135 Call parameters for TON

Identifier	Parameter	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration for which the rising edge at input IN is delayed
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

Example

```
VAR
    Mytimeout : TON;
END_VAR

Mytimeout (PT := T#2s, IN := TRUE);
IF (mytimeout.Q) THEN
    ; //Time has expired
END_IF;
```

TOF OFF delay

With a signal state change from 0 to 1 at start input IN, state 1 appears at output Q. If the state at the start input changes from 1 to 0, then time ET is started. If a change occurs at input IN from 0 to 1 before time ET has elapsed, then the timer operation is reset. A start is initiated again when the state at input IN changes from 1 to 0. Only after the duration ET = PT has elapsed does output Q adopt a signal state of 0. This means that the output is switched off with a delay.

The following figure illustrates the mode of operation of the TOF OFF delay timer.

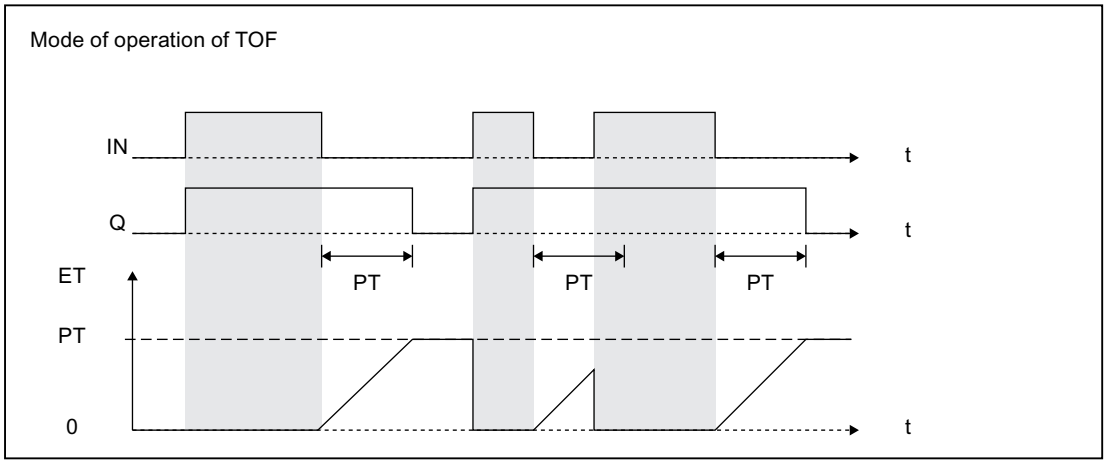


Figure 4-169 Mode of operation of the TOF OFF delay timer

The following table shows the call parameters.

Table 4-136 Call parameters for TOF

Identifier	Parameter	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Period for which the falling edge at input IN is delayed
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

Real-time clock RTC

The rising edge on SET sets the real-time clock to the value of PDT; PDT is copied to CDT. If READ is set to TRUE, then the current system time is read and is available at output CDT.

Table 4-137 Call parameters for the RTC

Identifier	Parameter	Data type	Description
SET	Input	BOOL	Set time, default FALSE
READ	Input	BOOL	Read time, default FALSE

Identifier	Parameter	Data type	Description
PDT	Input	DT	Value to which the real-time clock is to be set, default DT#0001-01-01-0:0:0 If the value is less than the default value of the SIMOTION device real-time clock, the real-time clock will be set to its default value (e.g. with C2xx: DT#1994-01-01-00:00:00)
CDT	Output	DT	Current system time

The granularity for setting the real-time clock is 1 ms; the accuracy depends on the position control cycle clock.

4.2.10.6 Splitting bit-string data types

General information for splitting bit-string data types

The following functions enable a variable of a bit string data type to be split into multiple variables of a lower-level data type.

The inverse functions are implemented as functions, see Combining bit-string data types (Page 1499).

_BYTE_TO_8BOOL function block

Description

This function splits a variable of data type BYTE into eight variables of data type BOOL.

User interface prototype

```
FUNCTION_BLOCK _BYTE_TO_8BOOL
    VAR_INPUT
        bytein : BYTE;
    END_VAR
    VAR_OUTPUT
        bit0,               // Least significant bit
        bit1, bit2, bit3, bit4, bit5, bit6,
        bit7 : BOOL;        // Most significant bit
    END_VAR
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameter

bytein

Data type: BYTE

Variable of data type BYTE to be split into eight variables of data type BOOL.

Output parameter

bit0
...
bit7

Data type: BOOL

Eight variables with the individual bits of the input parameter

bit0: Least significant bit

...

bit7: Most significant bit

_WORD_TO_2BYTE function block

Description

This function splits a variable of data type WORD into two variables of data type BYTE.

User interface prototype

```
FUNCTION_BLOCK _WORD_TO_2BYTE
    VAR_INPUT
        wordin : WORD;
    END_VAR
    VAR_OUTPUT
        byte0,           // Less significant byte
        byte1 : BYTE;    // More significant byte
    END_VAR
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameter

wordin

Data type: WORD

Variable of data type WORD to be split into two variables of data type BYTE.

Output parameter

byte0
byte1

Data type: BYTE

Two variables with the individual bytes of the input parameter

byte0: Less significant byte
byte1: More significant byte

_DWORD_TO_2WORD function block

Description

This function splits a variable of data type DWORD into two variables of data type WORD.

User interface prototype

```
FUNCTION_BLOCK _DWORD_TO_2WORD
    VAR_INPUT
        dwordin : DWORD;
    END_VAR
    VAR_OUTPUT
        word0,           // Less significant word
        word1 : WORD;    // More significant word
    END_VAR
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameter

dwordin

Data type: DWORD

Variable of data type DWORD to be split into two variables of data type WORD.

Output parameter

word0
word1

Data type: WORD

Two variables with the individual words of the input parameter

word0: Less significant word
word1: More significant word

_DWORD_TO_4BYTE function block

Description

This function splits a variable of data type DWORD into four variables of data type BYTE.

User interface prototype

```
FUNCTION_BLOCK _DWORD_TO_4BYTE
    VAR_INPUT
        dwordin : DWORD;
    END_VAR
    VAR_OUTPUT
        byte0,           //  Least significant byte
        byte1, byte2,
        byte3 : BYTE;    // Most significant byte
    END_VAR
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameter

in

Data type: DWORD

Variable of data type DWORD to be split into four variables of data type BYTE.

Output parameter

byte0
byte1
byte2
byte3

Data type: BYTE

Four variables with the individual bytes of the input parameter

byte0: Least significant byte

...

byte3: Most significant byte

4.2.10.7 Emulation of SIMATIC S7 commands

General

The following function blocks are interface-compatible with the commands for the SIMATIC S7 counters and timers; see the reference manual SIMATIC Statement List (STL) for S7-300/400.

Note

You should generally use the function blocks in accordance with IEC 61131-3 (**counters** and **timers**).

See also

General information on counters (Page 1622)

Timers (Page 1627)

_S7_COUNTER function block

User interface prototype

```
FUNCTION_BLOCK _S7_COUNTER
    VAR_INPUT
        CU: BOOL;
        CD: BOOL;
        PV : INT;
        S: BOOL;
        R : BOOL;
    END_VAR
    VAR_OUTPUT
        Q : BOOL;
        CV : INT;
    END_VAR;
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameters

CU

Data type: BOOL

Count up (with edge)

CD

Data type: BOOL

Count down (with edge)

PV

Data type: INT

Preset value

S

Data type: BOOL

Set preset value (with edge)

R

Data type: BOOL

Reset counter (static)

Output parameters

Q

Data type: BOOL

Counter status

CV

Data type: INT

Counter value

_S7_TIMER function block

This function block is used to emulate SIMATIC S7 time functions.

User interface prototype

```
FUNCTION_BLOCK _S7_TIMER
    VAR_INPUT
        T_TYPE : WORD;
        FR     : BOOL;
        S      : BOOL;
        R      : BOOL;
        TV     : TIME;
        TV_BCD : WORD;
    END_VAR
    VAR_OUTPUT
        Q      : BOOL;
        BI     : TIME;
    END_VAR
    // ... (Code)
END_FUNCTION_BLOCK
```

Input parameters

T_TYPE

Data type: WORD

S7 timer function to be executed

TYPE_S7T_SI_ = 16#0001 SI SP
TYPE_S7T_SV_ = 16#0002 SV SEE
TYPE_S7T_SE_ = 16#0004 SE SD
TYPE_S7T_SS_ = 16#0008 SS SS
TYPE_S7T_SA_ = 16#0010 SA SF
TYPE_S7T_BCD_IN = 16#0020 SA SF
TYPE_S7T_SI_ = 16#0040 R FR
TYPE_S7T_SI_ = 16#0080 L LC (U; UN; X ...)

FR

Data type: BOOL

Release

S

Data type: BOOL

Set preset value

R

Data type: BOOL

Reset timer

TV

Data type: TIME

Preset value (as data type TIME)

TV_BCD

Data type: WORD

Preset value in S7 coding

Output parameters

Q

Data type: BOOL

Timer status

BI

Data type: TIME

Current time

4.2.11 SIMOTION memory concept (in the target device)

4.2.11.1 Overview of the memory in the target device

Memory types

SIMOTION has a staggered memory management concept. A distinction is made in the target device between the DRAM (RAM disk and RAM) and the SRAM/NVRAM. The data (TOs, units and TPs) in the DRAM are lost when the device is switched off whereas the SRAM/NVRAM can retentively store smaller data quantities. A removable data medium (CF card or memory card) is also available as ROM.

RAM disk

The RAM disk is a virtual memory that can be controlled in the same way as a hard disk, but can be accessed much faster. A fixed memory area is reserved for the RAM disk in the DRAM of the target device. After the download, the RAM disk contains the hardware and device configuration, technology packages (TP), configuration data of the technology objects, and the program units. With **Copy RAM to ROM**, the contents of the RAM disk are written to the ROM (on the card). During an upload, the data is loaded from the RAM disk to the programming device. You can display the utilization of the RAM disk in the device diagnostics under System utilization (ONLINE only).

RAM

User data (user RAM) and system data (system RAM) are stored in the RAM. The user RAM contains the TO Current data memory, the TO Next memory of the technology objects (for more information, see Changing system variables and configuration data online (Page 1640)) and is also used to store technology packages, units (variables and code). During power-up or download, the data is loaded from the RAM disk to the RAM.

The user RAM containing the TPs and units can be divided into a data memory (e.g. variables) and a code memory (compiled user programs) for clarity. Retain variables are saved retentively (powerfail-proof) in the SRAM/NVRAM.

The kernel, kernel data and further settings such as communication settings, cycle clock settings and the contents of the diagnostic buffer are stored in the system RAM.

You can display the utilization of the RAM in the device diagnostics under System utilization (ONLINE only).

ROM (CF card or memory card)

Data is saved retentively on the memory card (ROM). To do this, you must execute the command **Copy RAM to ROM**. The data saved last with Copy RAM to ROM is loaded from the ROM to the RAM disk during power-up and from there to the RAM. You can display the utilization of the CF/ memory card in the device diagnostics under System utilization (ONLINE only).

SRAM/NVRAM (retentive data - non-volatile data)

Data is saved retentively and in a powerfail-proof manner in the SRAM/NVRAM. The data is loaded from the SRAM/NVRAM to the RAM during power-up. You can display the allocation of the SRAM/NVRAM in the device diagnostics under System utilization (retentive data, ONLINE only).

Non-volatile data available in SIMOTION devices:

Non-volatile data	Contents
Kernel data	Last operating mode
	IP parameters (IP address, subnet mask, router address)
	DP parameters (PROFIBUS DP address, baud rate)
	Diagnostics buffer
Retain variables	Variables in the interface or implementation section of a unit declared with VAR_GLOBAL RETAIN
	Local variables in programs, function blocks or classes declared with VAR RETAIN. As of SIMOTION Kernel version V4.5
	Global device variables set with the "RETAIN" attribute
Retain TO	Absolute encoder offset
DCC blocks	SAV blocks and user-defined blocks with retain behavior

The user program can use the "_savePersistentMemoryData" system function to save the contents of non-volatile data to the memory card. This prevents the retain variables and the absolute encoder position from being lost if, for example, a component is replaced (for more details, see the *System Function/Variables* List Manual).

The following graphic shows the general structure of the memory in the target device

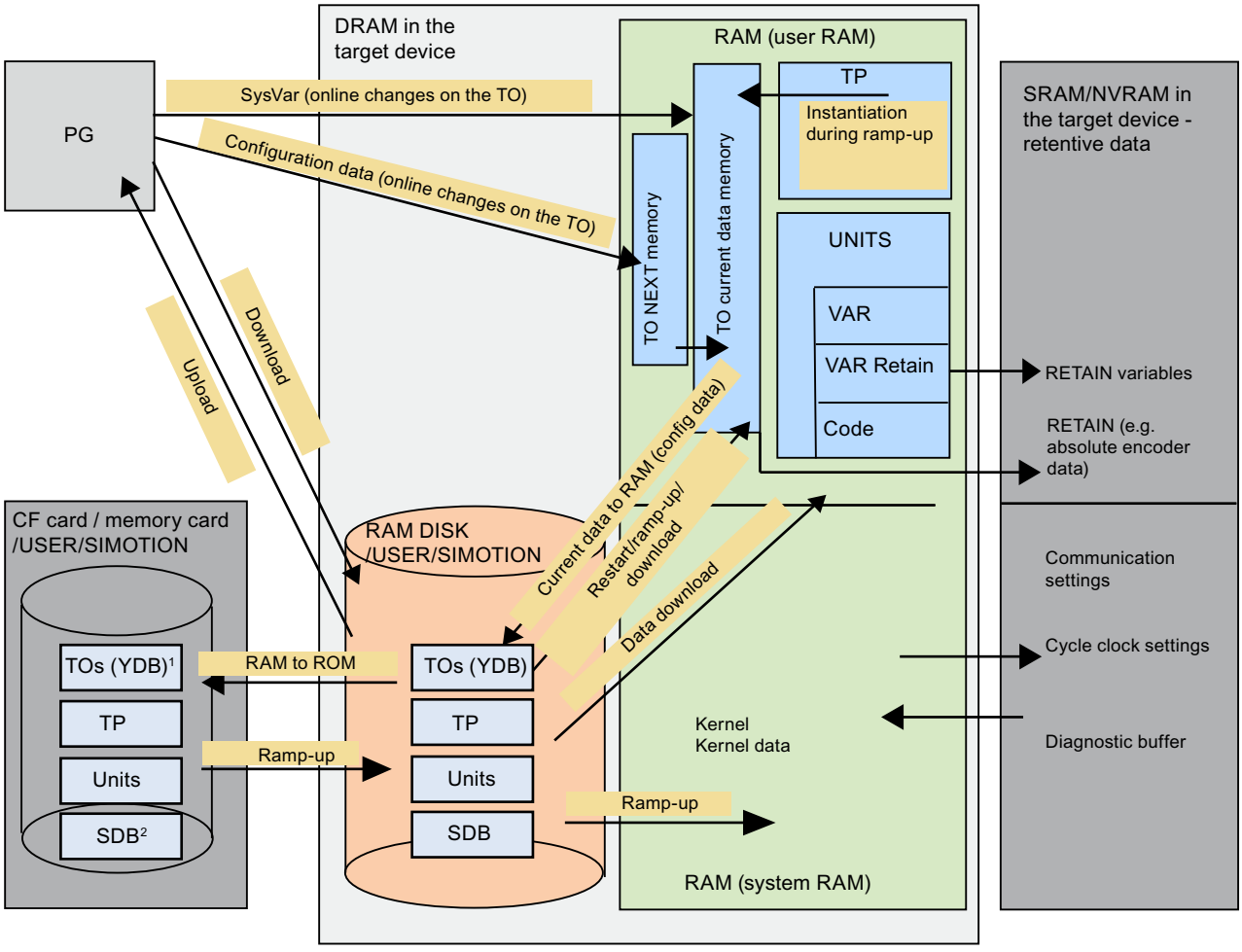


Figure 4-170 Memory concept, general representation

[1] YDBs contain the configuration data of technology objects, such as system variables, configuration data or alarms.

[2] SDBs are system data blocks

See also

Overview of the data download (Page 1643)

Copy RAM to ROM (Page 1676)

Download in RUN (Page 1655)

4.2.11.2 Memory access

Power-up of the target device

The data of the TOs, technology packages, units and further data is transferred from the ROM via the RAM disk to the USER RAM during a power-up. The TO data is loaded to the TO current data memory, the technology packages (TP) and units to the USER-RAM. The individual TOs are instantiated and assigned the appropriate values. Communication settings, cycle clock settings, and the content of the diagnostic buffer (if available) are transferred from the SRAM/NVRAM to the system RAM, along with the retain data.

Note

IP and DP parameters in non-volatile data

If there is configuration data on the memory card, the IP and DP parameters are loaded from the memory card during ramping up and used by the SIMOTION device. The addresses defined in the configuration data are used to go online. During ramping up, the IP and DP parameters on the memory card are also written to the non-volatile data. If the SIMOTION device is then powered up with a CF card with no configuration, the IP and DP parameters are retained in the non-volatile data and are used by the device. This means the SIMOTION device can continue to go online if a configuration has been loaded with SIMOTION SCOUT at least once or if the SIMOTION device is powered up with a memory card containing a configuration.

Power On and overall reset

Data (configuration data, default values of system variables) is loaded from ROM (memory card) into the RAM disk and then into the TO current data memory. During the memory reset operation, the RAM and the retentive data in the SRAM/NVRAM, except for the communication configuration (baud rates, network addresses, etc.), are deleted prior to the power-up. In addition, the unit data is deleted during the memory reset and are then reloaded from the ROM during power-up.

Restarting a technology object

Data of the TO (configuration data, default values of system variables, cams) is loaded from RAM disk into the TO current data memory.

STOP - RUN transition

There is no access to any of the various memories during the transition from STOP to RUN.

Download to the target device

During a download from the programming device to the target device, the data is first written to the RAM disk and then from there to the RAM, with optional data initialization. See also Overview of the data download (Page 1643) and Separate initialization of source and TO data during a download (Page 1655).

Additional data (e.g. sources) can be included in the download to the target device.

This data is required for

- Online object comparison (e.g. additional properties)
- Various detailed comparisons (e.g. ST source file comparison)
- "Load to PG" function (complete upload to offline project)
- Synchronization with online objects

In order to be able to load sources or additional data of a project to the PG, this must be specified in the project under **Options > Settings > Download > Store additional data on the target device**.

Changing system variables and configuration data online

If you change the values of system variables, the system variable values are effective immediately in the TO current data memory. New configuration data values are initially stored in the Next memory. Immediately effective configuration data is transferred automatically to the TO current data memory. Configuration data that only becomes active following a RESTART on the technology object (set the *restartactivation* system variable to ACTIVATE_RESTART) is only written to the TO current data memory after the RESTART.

To save the configuration data changed online to the offline project, you must first transfer the content of the TO current data memory to the RAM disk with the menu command **Target system > Copy current data to RAM**.

The configuration in SCOUT is then no longer consistent with the configuration in the target device as the data of the RAM disk is used for the consistency check. Read the data from the RAM disk with the menu command **Target system > Load > Load to PG** (only for the configuration data) to re-establish a consistent system state. To ensure that the configuration is saved on the CompactFlash card or memory card with protection against power failure, execute the menu command **Target system > Copy RAM to ROM**. This menu command can also be used to implicitly execute the **Copy current data to ROM** and **Load to PG** functions.

As of V4.2, configuration data is completed during ramp-up using parameter values of the SINAMICS drive (adaptation). For more detailed information, see "Adaptation" under Symbolic assignment - introduction (Page 1189). When "Copy current data to RAM" or "Copy RAM to ROM" is executed, adapted values are detected and "Load to PG" is automatically preselected as well.

Note

Copy current to RAM does not transfer the values of the system variables to the RAM memory. This means that "Save to memory card (Copy RAM to ROM)" or "Save in the engineering project (Load to PG)" is not possible.

So that values of system variables can also be saved in the engineering project and on the memory card, the default value of system variables must be changed OFFLINE and then loaded to the target device per download and saved.

For information on how TOs, variables and programs behave during a download in RUN, see Download in RUN (Page 1655)

Backing up non-volatile SIMOTION data

You have the following options for backing up non-volatile SIMOTION data on the memory card:

- In the user program:
  The "_savePersistentMemoryData" system function allows the user program to back up the content of non-volatile SIMOTION data on the memory card. This ensures that the retain variables and the absolute encoder position are backed up in the event that a spare part is used.
- Using the switch/pushbutton (service selector switch or DIAG pushbutton the SIMOTION D) or IT DIAG.

Backing up non-volatile SINAMICS data

As of SINAMICS FW version V4.5, the backup of the non-volatile SINAMICS data (NVRAM data) is carried out by setting the CU parameter p7775 to the value 1. The data is stored on the CF card under /USER/SINAMICS/DATA/NVRAM/PMEMORY.ACX.

- For SIMOTION D4xx-2 with SINAMICS Integrated and CX32-2 controller extension
- For SINAMICS drives (e.g. S120) with CU310-2 or CU320-2 control units

Restoring non-volatile SIMOTION data

SIMOTION data backed up on a CompactFlash card with _savePersistentMemoryData is restored in the following cases:

1. After an overall reset
2. By selecting a switch position on SIMOTION D
3. SRAM content lost due to a discharged battery. In this case, the entire contents of the SRAM are restored from the file.

If, after a module has been replaced, retain data is available on the module, it is accepted and remains valid as long as the TO name or unit name is consistent. Otherwise, this data is invalid.

This means that once a module has been replaced, if the project is not consistent with the saved retain data, you should proceed as described in 1 or 2.

The backed-up data in the USER/SIMOTION/PMEMORY.XML file is copied back to the SRAM/NVRAM when the controller is being ramped up.

System variable device.persistentDataPowerMonitoring.persistentDataState indicates the state of the non-volatile SIMOTION data after ramp-up. The FROM_RAM state indicates that the non-volatile SIMOTION data has not been lost and has been downloaded from the SRAM/NVRAM.

Table 4-138 State of non-volatile data after ramp-up (persistentDataState system variable)

State	Meaning
FROM_RAM (1)	Non-volatile SIMOTION data in the SIMOTION device is used
FROM_FILE (2)	Non-volatile SIMOTION data restored from the backup file

State	Meaning
FROM_BACKUP (3)	Non-volatile SIMOTION data is restored from the backup copy of the backup file
INVALID (4)	Data in the non-volatile SIMOTION data and in the backup file / backup copy of backup file is invalid or not available/deleted. The SIMOTION device has copied the default settings to the non-volatile SIMOTION data and used this data to power up.

Restoring non-volatile SINAMICS data

The restoration of the SINAMICS-non-volatile data

- Is performed automatically in the event of a module replacement:
  A module replacement is detected on the basis of the serial number.
- Can be performed manually:
  Restore can be initiated manually by setting the CU parameter p7775 to 2.

4.2.12 Downloading data to the target device

4.2.12.1 Overview of the data download

Description

You have to download the project data that you have created with the SCOUT to the target system. The target system can contain several CPUs (SIMOTION controllers or drives). The project data contains the programs (ST, MCC, LAD/FBD and DCC) that you have created and compiled, the hardware configuration and the technology objects (with their technology packages) that you have created and parameterized.

You must reload a project to the target system, if you have:

- Created or changed programs
- Changed definitions of global variables or symbolic I/O variables
- Made changes to the execution system
- Changed technology object configurations

Download requirements

The following conditions must be met before the project can be downloaded to the target system:

- All cables must be plugged in and the interfaces configured
- All program sources are compiled (Project > Save and compile changes) or the program sources will be compiled automatically before the download.
- The project has been checked for consistency (Project > Check consistency)
- The system status is ONLINE (menu: Project > Connect to target system)

Note

The processing work increases with the size of the project. Plan the required components first in order to avoid any download errors. Note the technical specifications for the relevant device, technology object, communications interface, etc.

If an error occurs, transfer of data to the target system is aborted. The **Target system output** tab in which the transfer sequence and any errors are logged indicates the possible cause of the error.

Note

If a power failure occurs on the SIMOTION device during the download (RAM2ROM), a message will appear in the diagnostics buffer: Starting lockout set, after you have switched to RUN mode. Go online again and carry out the download once more.

In earlier versions, this can lead to various faults that cannot be corrected by the system. Unexplainable error messages may appear.
If this happens:
Go online again and carry out the download once more.

Download scope

You can either download the entire project or perform a download for a specific device.

- Downloading a project to the target system (all target devices) (Page 1647)
- Downloading CPU/drive unit to target device (Page 1650)
- Downloading a program subset and single units (sources) to the target device (Page 1652)

Operating state

You can perform a download in the STOP mode, and also in the RUN mode.

Additional download options in OFFLINE mode

Additional functions in OFFLINE mode are also available for transferring the project data in SCOUT to the memory card:

- Download direct to memory card or hard disk, see Download direct to memory card or hard disk (Page 1671)
- Device Update Tool, see Downloading using the device update tool (Page 1673).

4.2.12.2 Save and compile

The project must be saved and compiled first before the download. SIMOTION SCOUT distinguishes three commands in the main project menu that act differently when saving and compiling:

- Save; the project is saved to the hard disk
- Save and compile changes
- Save and recompile all

In addition, there is the following command in the context menu of a source, a device and a library:

- Accept and compile (a single source)

Save

The project is saved to the hard disk. The changes are accepted into the project. No other processes, such as compilation or consistency checking, are initiated for the project.

Save and compile changes

On this command, the whole project is searched for changes. If a source is found which has been changed or has no compilation results, this and any linked sources are compiled and saved (e.g. during an FB call). Therefore only the changes are compiled. Use this command for day-to-day operations within a SCOUT version.

Save and recompile all

All sources of the entire project are recompiled with this command. DCC libraries are also recompiled.

The **Save and recompile all** command is suitable if you are entirely sure that all the old data from older SCOUT versions has been removed and should be replaced with new compilation results. It consists of the following steps:

- Project-wide deletion of all compilation results
- Recompilation of all objects (except libraries, see also AUTOHOTSPOT)

Use this command if you specifically want to convert a project from an earlier SCOUT version to a later one. All the error correction and optimization features are accepted into the new SCOUT as part of the process, Thus, the compilation results in SCOUT and SIMOTION RT can become inconsistent. The project navigator and object comparison then display the objects as "inconsistent" in ONLINE mode. You need to load the whole project to the target system in order to debug it.

Accept and compile

The Accept and compile command compiles the changes of the individual source and accepts them in the project. The data, together with the project, is only saved to the hard disk if you select **Save**, **Save and compile changes** or **Save and recompile all**.

Comparison of compilation results in the project and in SIMOTION RT

Before SIMOTION SCOUT V4.0, compilation results were compared on the basis of their time stamp. As of V4.0, SCOUT uses a hash function to compare compilation results. A hash function compares larger amounts of data based on the hash code. Compilation results are only displayed as "consistent" if their hash code is the same in both the project and SIMOTION RT. The hash code is only deemed to be the "same" if code and compiler (note the SCOUT version) are the same.

Information about errors

Detailed information about transfer and compilation errors is output during the compilation of single sources (OFFLINE **Compile/Check Output** and ONLINE **Target System Output**). Double-clicking the error message in the detail window causes the cursor to jump to the error location in the source. Correct the errors, for example, by:

- Changing the configuration of the technology objects
- Changing the programs

Repeat the steps as often as required until no more errors are displayed.

Note

You have the option to output detailed error information even for project-wide compilation (with Save and (re)compile all). To do this, execute **Options > Settings > Compiler > Create detailed message log during compilation of the project/devices**

4.2.12.3 Editing a project with a more recent SCOUT version

Opening a converted project

The project was opened and edited with a more recent SCOUT version, and saved in the current project format. This can cause inconsistencies in online mode for objects that were not edited. It suffices to compile the project to avoid this problem. Click **Yes** to compile the project immediately. If you click **No**, you have the possibility to resolve the inconsistencies subsequently with the **Save and compile changes** function.

4.2.12.4 Performing a consistency check

Checking the project for consistency and compiling the project

Before you download the project to the target system, the program sources must be compiled without error and the consistency ensured. In this process, I/O addresses or TO configurations are checked, for example.

1. Select the **Project > Save and compile changes** menu.
   SIMOTION SCOUT compiles all changed sources.
   Further information about saving and compiling the project can be found in Save and compile (Page 1645).
2. Select the **Project > Consistency check** menu.
   SIMOTION SCOUT checks, for example, whether all technology objects have been configured and that the sources have been compiled without errors. Prior to a download, a consistency check can be performed automatically if the corresponding option is selected under **Options > Download > Check consistency before loading**.
   During the download, a consistency check is always performed in the target system. If download errors occur, their possible causes are displayed by entries in various output windows (ONLINE in the **Target System Output** window) of the detail view.

Note

SINAMICS components for the consistency check

SINAMICS components are considered only for a cross-project consistency check. For the consistency check on the device, e.g. on the SIMOTION D4x5-2, the SINAMICS component (SINAMICS Integrated or Controller Extensions) is not checked.

4.2.12.5 Downloading a project to the target system (all target devices)

During a project download, the entire project is loaded to all devices that are connected online (you can tell which devices these are from the green or red/green plug icons in the project navigator). This loads data to all devices selected in the **Target Device Selection** dialog. This includes all devices which are located below a SIMOTION device (e.g. SINAMICS Int, external PROFIBUS/PROFINET slaves). If you go offline with a specific device prior to running the download, data will not be downloaded to that device. If you change the selection in the **Target Device Selection** dialog after going online, this does not affect the download at first (device remains connected).

Note

You can only perform a project download for all devices connected ONLINE and their subordinate drive units, in the STOP operating state.

You cannot download to drives that cannot be configured in SCOUT, such as MASTERDRIVE or 611U.

Note

Project change (downloading a new project to the target system)

Delete the user data on the CF card before downloading the new project.

We also recommend that you reset the controller to the factory settings.

The procedure is described in the Commissioning Manuals.

Procedure

1. Click the **Download project to target system** button. The **Download to Target System** dialog box is displayed.

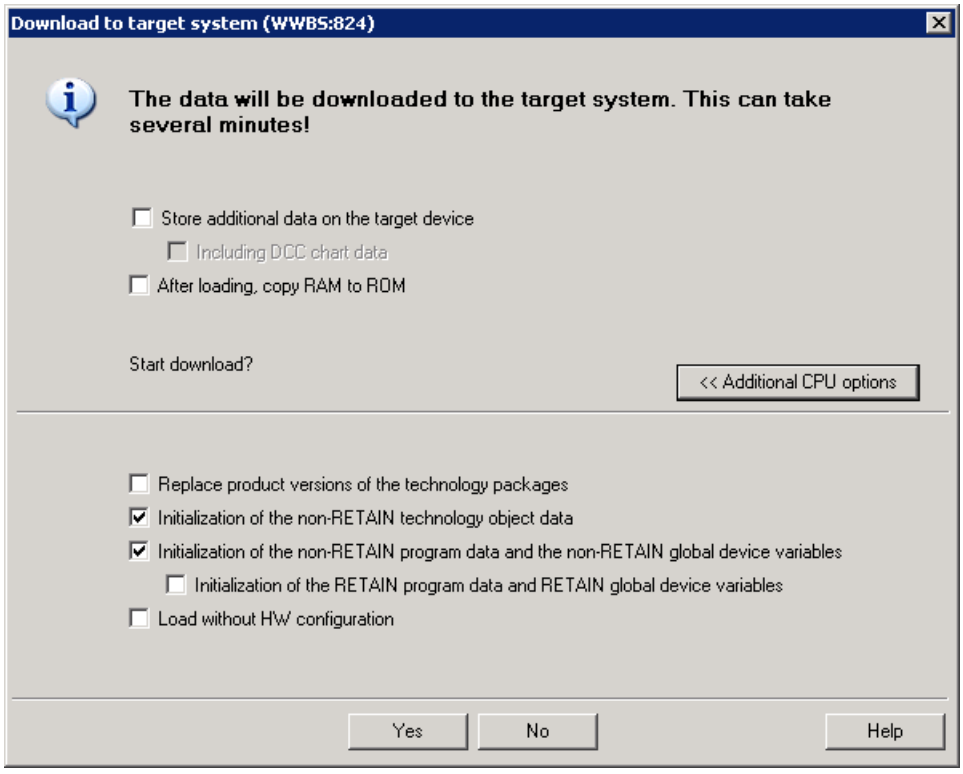


Figure 4-171 Download to target system

You can select the following options depending on the device:

- **Store additional data on the target device**; you can use this option to store supplementary data (e.g. program sources) on the target device.
  - **Including DCC chart data**
    With DCC, it is possible to store graphical chart data in addition to the supplementary data that enables a functional chart to be uploaded, meaning that the chart in question is in the full source text and can, therefore, be edited further.
  - For further information, see also Project download - settings in the SCOUT online help.
- **After loading, copy RAM to ROM**; copies RAM to ROM after the download for all target devices.
  After the download, the data is copied from RAM to ROM for all target devices. The loaded configuration is then stored retentively on the memory card.

Click **Additional CPU options** to display the following options:

- **Replace product versions of the technology packages**; this loads the technology package versions selected in the Select technology packages dialog to the target system. See also Downloading the selected product version of the technology packages (Page 1654).
- **Initialization of the non-RETAIN technology object data** (see Separate initialization of source and TO data during a download (Page 1655))
- **Initialization of the non-RETAIN program data and the non-RETAIN global device variables** - only effective in STOP mode
  - **Initialization of the RETAIN program data and RETAIN global device variables** (see Separate initialization of source and TO data during a download (Page 1655))
- **Load without HW configuration** (see Download without HW Config information (Page 1667))

The global settings are made at **Options > Settings > Download**.

Accepting or rejecting settings

- Click **Yes** to start the download or **No** to cancel the download.

The additional options are also active when they are not visible. The settings with which the last download was performed are then used. The only exception is the **Replace product versions of the technology packages** option which is only active for one action.

For drive units, you can only select **After loading, copy RAM to ROM** and **Supplementary data... incl. DCC chart data**.

Troubleshooting possible errors from older SCOUT versions:

If an error occurs when using projects from older SCOUT versions that can be reproduced in those versions, the error status can be resolved for the affected device and the project re-downloaded using the **Save and recompile all** command. If necessary, run **Delete user data on card** first.

Error messages:

- "IOM configuration inconsistent" (although there is no overlap of I/O variables).
- "UPP transfer of the source failed"

See also

Save and compile (Page 1645)

4.2.12.6 Downloading CPU/drive unit to target device

Description

In addition to a project download, you can also selectively load data to each CPU / drive unit. The standard procedure is the download in STOP mode. You can also perform a download in RUN operating state. For further information, see Download in RUN (Page 1655).

Procedure

1. First select the device to which you want to load the data in the project navigator (device must be ONLINE).
2. Click the **Download CPU / drive unit to target system**
   or
   right-click the device in the project navigator and perform **Target device > Download** in the context menu.
3. The **Download** dialog box is displayed. The contents depend on the device for which you want to download data.

Loading the CPU to the target device

A download can be performed for the entire CPU (without drive unit) but also separately for individual related download units.

- Download CPU without drive unit
- Download all technology objects of the CPU
- Download all programs of the CPU

You can select the following options:

- **Store additional data on the target device**; you can use this option to store supplementary data (e.g. program sources) on the target device.
  - **Including DCC chart data**
    With DCC, it is possible to store graphical chart data in addition to the supplementary data that enables a functional chart to be uploaded, meaning that the chart in question is in the full source text and can, therefore, be edited further.
  - For further information, see also Project download - settings in the SCOUT online help.
- **Copy RAM to ROM after download**
  After the download, the data is copied from RAM to ROM for the selected target device. The loaded configuration is then stored retentively on the memory card**.**
- **Perform download during RUN**; enables a download to be performed in RUN mode.
  Click **Additional CPU options** to display the following options:
- **Replace product versions of the technology packages**; this loads the technology packages selected in the Select Technology Packages (Page 1654)dialog to the target device.

- **Initialization of the non-RETAIN technology object data** - only effective in STOP mode (see Separate initialization of source and TO data during a download (Page 1655))
- **Initialization of the non-RETAIN program data and the non-RETAIN global device variables** - only effective in STOP mode
  - **Initialization of the RETAIN program data and RETAIN global device variables** (see Separate initialization of source and TO data during a download (Page 1655))
- **Load without HW configuration** (see Download without HW Config information (Page 1667))

You set the global default under **Options > Settings > Download** or **CPU download**.

The additional options are also active when they are not visible. The settings with which the last download was performed are then used. The only exception is the **Replace product versions of the technology packages** option which is only active for one action.

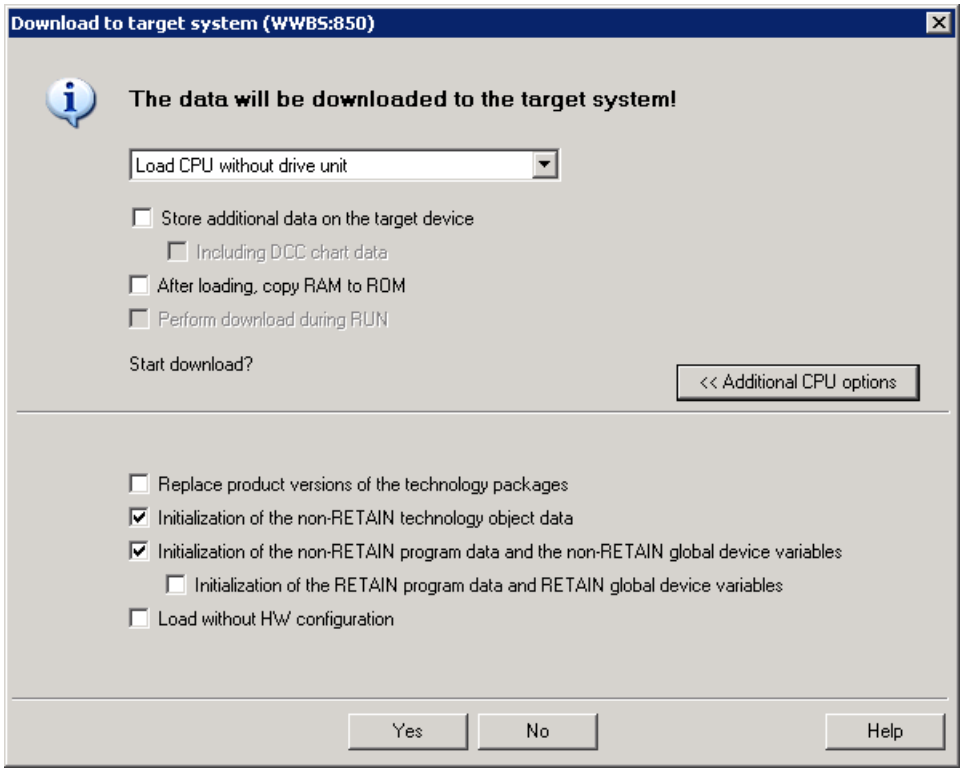


Figure 4-172 Download to CPU / drive unit

Accepting or rejecting settings

- Click **Yes** to start the download or **No** to cancel the download.

Downloading the drive to the target device

For drives, you can only download the drive data (parameterization, etc.) to the drive unit.

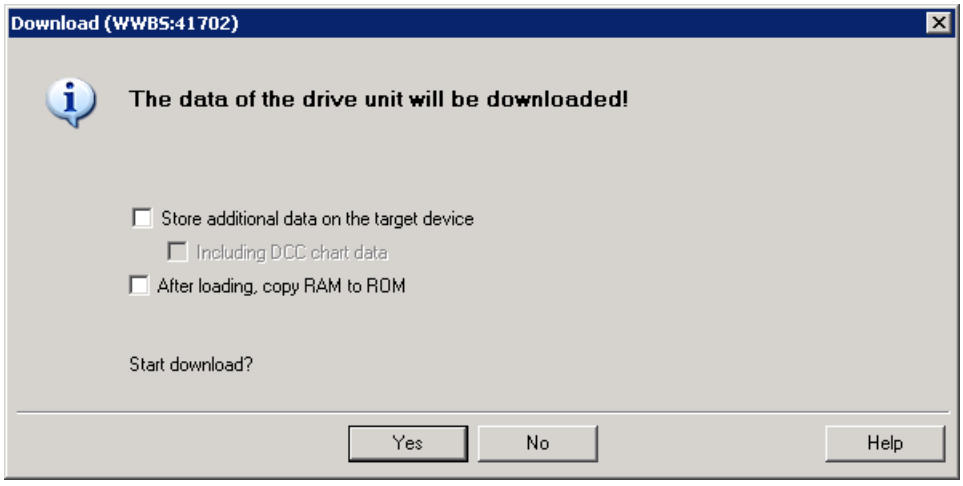


Figure 4-173 Download to drive unit

By default, the download is performed in STOP. After a successful download, the CPU can be restored to its previous status.

You can select the following options:

- **Store additional data on the target device**; you can use this option to store supplementary data (e.g. program sources) on the target device.
  - **Including DCC chart data**
    With DCC, it is possible to store graphical chart data in addition to the supplementary data that enables a functional chart to be uploaded, meaning that the chart in question is in the full source text and can, therefore, be edited further.
- **After loading, copy RAM to ROM**; copies RAM to ROM to the selected device once the download is complete.

See also

Download of changed technology objects (Page 1668)

4.2.12.7 Downloading a program subset and single units (sources) to the target device

Download of selected units

As of SIMOTION V4.4, you can download a single of several units to the SIMOTION device in the RUN and STOP states.

You no longer have to download all units that have been displayed as inconsistent, but can download a specific program subset. This enables, for example, two users who are connected online to target device at the same time with SCOUT to download separate program subsets independently.

To do this, right-click the unit in online mode and execute **Download...** in the context menu.

Note

The selective download is not possible for DCC charts.

When downloading a subset, the system ensures that the project as a whole is available again in a consistent state. For this purpose, it may be necessary to download additional units for the selection. This operation must be confirmed in a dialog box.

Note the following applications in this respect:

- You change the interface of a unit and this interface is used by a second unit. This unit is then also downloaded.
  Example: You change the signature of a function.
- You load only a subset of the changed units. After checking the dependency, independent units can be downloaded as a subset.
- You are referencing a library that has not been downloaded, in a unit that you want to download, or you change the interface of a library used in the unit. The library is downloaded with the unit.
- You delete a program (POU) of a unit that is called from the execution system. When downloading the unit, the execution system must also be downloaded.
- If programs have been downloaded to the controller that are not in the project, these programs are removed from the controller during a complete download. If you only load a subset, then programs are only deleted when there are no longer any dependencies.
  If the programs that have not been removed reference the programs that are to be downloaded (uses), then the behavior of these programs may be changed.
  Therefore, perform a detailed comparison to display the programs online and offline.

Selective download not possible

A selective download is not possible for the following reasons:

- If the relevant CPU has not been compiled error-free or the consistency check is not error-free.
- If an I/O variable has been changed in the address list.
- If a global device variable has been inserted, deleted, renamed or changed structurally (e.g. data type changed, length of the array changed); a change of the initial value does not prevent the download.
- If a TO has been inserted, deleted, renamed or changed structurally.
- If technology packages have to be loaded.
- If DCC is affected, see above.
- If tasks have been activated, deactivated for an execution system, or the cycle clocks have been changed.

4.2.12.8 Downloading the selected product version of the technology packages

Description

As of Version V4.2, you can select the product version of technology packages within a particular version (service packs and hotfixes). **Replace product versions of the technology packages** is used to download the selected technology package versions to the target system. This enables you to overwrite newer or older versions of the technology package in the target system. Further information can be found at Selecting or replacing technology packages in the SCOUT online help.

How to load a specific version of a technology package to the target system

1. Right-click a SIMOTION device in the project navigator and then click **Select Technology Packages**.
   The dialog opens.
2. Select the technology package you want to update.
3. Select the version under TP version and Version. Hotfix versions can also be selected here as of V4.2.
4. Click **OK** to close the dialog and accept these settings.
5. Go online with the device and click **Download project to target system** or **Download CPU to target system**.
   The dialog opens.
6. First click **Additional options** and select **Replace product versions of the technology packages**, then click **OK.**
7. The technology package is downloaded to the target system, even if a technology package already exists there.
   The SCOUT's detail view shows the loading process for the technology packages.

Boundary conditions for replacing technology packages

- If no technology package version is selected in the **Select Technology Packages** dialog, the latest technology package of the appropriate firmware version is downloaded to the target system.
- If you are performing an upload, the versions of the technology packages already present in the target system are uploaded too, entered on the device, and appear in the **Select Technology Packages** dialog.
- Technology package versions are exported and imported at the same time.
- If the version selected is not already on the PG/PC, an error message is output.

See also

Downloading a project to the target system (all target devices) (Page 1647)

Downloading CPU/drive unit to target device (Page 1650)

4.2.12.9 Separate initialization of source and TO data during a download

Description

As of V4.1.2, you can initialize non-RETAIN and RETAIN data of programs (units) and global device variables as well as non-RETAIN and technology objects separately via two settings. The data is initialized together at the first or complete download of the project. During servicing or if you only want to reload programs (units, sources) or technology objects, you can initialize the data separately.

- Initialization of the non-RETAIN technology object data; only non-RETAIN data can be initialized for TOs. This does not affect the program data (variable values).
- Initialization of the non-RETAIN program data (variables) and the non-RETAIN global device variables The RETAIN program data and RETAIN global device variables can also be initialized. This does not affect the technology object data. The global device variables are initialized together with the settings for the program data (variable values).

The initialization only takes effect in the STOP operating state.

To make the settings, see Downloading CPU/drive unit to target device (Page 1650).

The "Time of variable initialization" section in the programming manuals contains general information on variable initialization.

4.2.12.10 Download in RUN

General information for downloading in RUN

Perform download in RUN

As for a download in STOP, all selected changes are always downloaded in RUN. Therefore, only manageable changes should be made for a download in RUN. You can load an entire target device or even just parts of the target device.

A number of issues presented below must be noted for a successful download in RUN

Allowing a download in RUN

Settings in SCOUT

1. In SCOUT, select **Enable download / copy RAM to ROM during RUN** under **Options > Settings > CPU download**. This activates the option **Perform download during RUN** in the **Download to target system** dialog.
2. Click the option **Perform download during RUN**.

Download of changed sources in RUN

Change to sources – general

If a source (unit) contains several programs, FBs or FCs (POUs), then the entire unit is always loaded.

A unit can generally be exchanged in RUN. At the time of transfer, neither code (of the programs, FBs or FCs) nor the data (variables) of this unit may be used in a task.

Units are loaded at the cycle control point.

At this point in time, all cyclic tasks are restarted. As no source is accessed at the time, no source is therefore blocked in cyclical tasks for exchange in RUN.

If Motion Tasks are also used and they have been started at the time of exchange, the system determines at the cycle control point which POUs are currently accessed and blocks the applicable units from exchange in RUN.

Several seconds are spent attempting to load the changed units simultaneously at the cycle control point. If that is not possible, the process is terminated.

Options for download in RUN:

- The number of cyclic tasks in which programs (POUs) can be loaded in RUN is limited to four.
- If the number of sources (units) and relevant tasks is too high for time or volume reasons, the sum of the changes cannot be processed (loaded). This problem is displayed in the output window and the complete set of changes is not transferred to the sequence. The old data remains effective.

Dependencies on other sources

SIMOTION applies the unit concept, so that you can access the global variables, data types, functions (FCs), function blocks (FBs), and programs of other source files.

You can connect to other program sources to use their exported components by using a USES instruction in ST sources or via the **Connections** tab on the declaration tables of MCC and LAD/FBD sources.

Changes to connected sources also affect download in RUN. Additional units may have to be loaded.

If units depend on another changed unit, these units must also be compiled before download. They may use data or POUs that have been changed. On download, these units must only be exchanged if data or POUs are used.

As of SIMOTION SCOUT V4.4, the dependencies between the units are taken into account. Only called (used) POUs require the unit to be replaced. There are therefore significantly fewer dependencies compared to older SCOUT versions. This behavior applies to all supported RT versions that are processed using SIMOTION SCOUT V4.4.

In projects that have been converted from older versions of SCOUT, the feature is only available after full re-compilation using SCOUT V4.4. Execute the function **Save and recompile all** on the **Project** menu for this purpose. Note that recompilation in this case creates an online inconsistency.

In the event of a simple change to the code of a POU and in the event of any changes to the data in the implementation part of a unit, only the modified unit has to be replaced. Please note that data changes in RUN are only possible if the data is initialized (see Changes to the data declaration (Page 1659)).

Reduction of dependencies between the units

In the Engineering System, individual parameters of the units are checked for changes as of SIMOTION SCOUT V4.4.

The decision regarding whether changes in a used unit are relevant to your unit reduces the dependencies and therefore the necessary number of units to be replaced.

The following parameters are changes when a unit is used:

- Data declarations (if data from the unit is used)
- POU interfaces of used POUs (if POUs from the unit are used)
  - The following is checked for FUNCTION, FUNCTION_BLOCK and PROGRAM: Return value, VAR_INPUT and VAR_IN_OUT
  - For FUNCTION_BLOCK and PROGRAM the following are also checked: VAR and VAR_OUTPUT

Example 1

If the code of unit_2 is only processed in the BackgroundTask, the modified unit_2 can be replaced at time t4. However, as unit_1 has started a call for code (POU) from unit_2 that at time t4 has not yet returned, unit_2 cannot be replaced at this point in time.

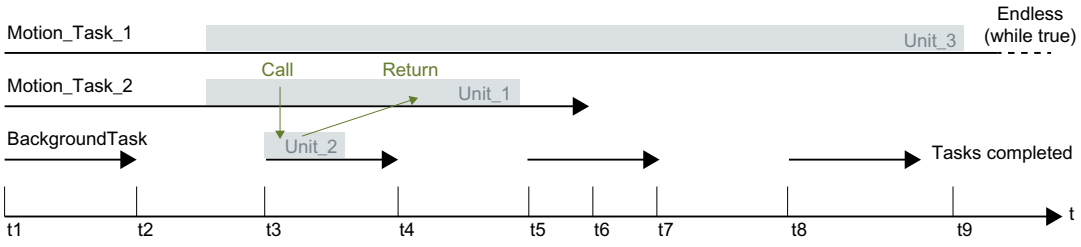


Figure 4-174 Dependencies on other sources – example 1

Unit_2 can be replaced at time t7 at the earliest.

A unit also cannot be replaced for as long as it is being processed by a MotionTask. In the example, unit_3 in the endless MotionTask can never be replaced.

The Motion_Task_1 can be stopped explicitly by the user to allow replacement to take place.

Example 2

In unit ST_2, data (data types and variables) and POUs (programs, functions, function blocks) used by ST_1 are indicated via the interface area:

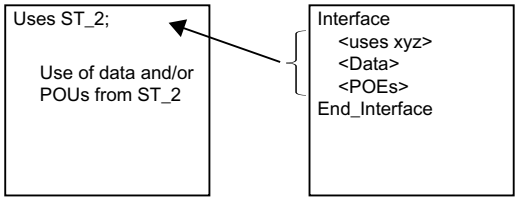


Figure 4-175 Dependencies on other sources – example 2

A change to the interface area of ST_2 has an impact on ST_1:

- ST_1 must also be re-compiled
- If ST_2 is to be replaced (loaded), ST_1 must also be replaced. Replacement depends on the scope of the change (see above) and the method for checking dependencies.

If, for example, a program is running in a cyclic task in ST_2 and a program is running in an endless MotionTask in ST_1, the unit ST_2 cannot be replaced in RUN. Unit ST_1 would also have to be replaced.

System unit of the execution system

Also note the system unit of the execution system in addition to the user units to be loaded This system unit primarily implements the execution system configuration and is therefore the connecting element between the programs and the tasks.

In the event of the following changes, the system unit must also be replaced:

- Changes to the program assignment and the size of the local data stacks in the execution system
- Changes to the data declaration of programs that are used in the execution system (PROGRAM container)
  (This point does not apply if the following compiler option is not selected for the unit that provides the program: "Create program instance data only once")
- Where a UserInterruptTask has been configured with condition on a global variable of a unit (from the interface area), if the interface (data structure of exported POUs) of this unit changes.

Where the system unit of the execution system is replaced, no task may be running simultaneously (long-running tasks and endless MotionTasks are problematic).

As of SIMOTION SCOUT V4.4, the Engineering System checks whether changes in the interface area of a user unit only related to the interfaces of the published programs or to the conditions of UserInterruptTasks. Only then is a change of the system unit of the execution system necessary, which must then be loaded and replaced. This procedure applies to all RT versions.

Changes to the data declaration

Changes to the following variables declarations are generally not permitted for a download in RUN, and the download in RUN is terminated. At the time of loading, the current values would otherwise have to be saved for all variables and, after the new variable structure had been created, the existing current values would have to be set as active again. This is not possible during the short period of loading.

If applicable measures are taken (permit initialization of the current variable values), download in RUN can, however, generally be performed anyway:

- Changes in VAR_GLOBAL blocks.
  Remedy: Permit initialization via compiler pragma BlockInit_OnChange or pragma lines in the declaration tables, see Recommended settings for download of changed sources in RUN (as of SIMOTION V4.1) (Page 1661).
- Changes in VAR blocks by programs (program instance data) that are used in cyclic tasks. The program instance data of the cyclic task is retained over the run of the tasks and therefore cannot be changed.
  Remedy: Permit initialization via compiler pragma BlockInit_OnChange or pragma lines in the declaration tables and additionally set the compiler option "Only create program instance data once" at the unit that declares the program (PROGRAM). See Recommended settings for download of changed sources in RUN (as of SIMOTION V4.1) (Page 1661).

The following variables declarations can always be changed:

- VAR_TEMP in programs and FBs, as well as all declaration blocks in FCs
- The program instance data of non-cyclic tasks (MotionTasks), as they are reinitialized on every start.
  (This only applies if the compiler option "Only create program instance data once" has not been activated for these programs (PROGRAM).)

Programmer notes for download of changed sources in RUN

Appropriate programming for download of changed sources in RUN

Use appropriate programming to enable a download of changed units in RUN mode.

- When structuring the units, ensure that data and POUs that are used both by cyclic and by sequential tasks are stored in separate units. This reduces the number of dependencies.
- Use the USES instruction (connection to other program sources to use their exported components) where possible in the implementation section and not in the interface section. This ensures there are fewer dependencies between the units, as USES instructions are not inherited in the connected source. Fewer units are affected, even at the compile stage. Changes therefore have an impact on fewer units. For more information, see "*USES statement in an importing unit*" in the *ST programming manual*.
- Configure the size of the local data stack in the task configuration for large sources (see Configure execution system (Page 1345) and Setting size of local data stack (Page 1685)):
  Take into account a reserve for download in RUN to avoid changes in the execution system. For example, additional storage may be required on the local data stack as the result of a change (e.g. for new local variables and function parameters).

In addition, for new variables:

- An additional variables declaration block may be used in the interface and implementation section of a unit with new declarations for TYPE and global variables. The variables declaration block must be introduced after the pre-existing declaration sections.
- Instead of using an additional variables declaration block, you can also create a new UNIT with the new data and then link it with USES (in the implementation section). This is also possible in MCC and LAD/FBD.

In addition, when using MotionTasks:

- Create units in accordance with how tasks and programs are assigned to one another. This minimizes dependencies and increases programming clarity.
  Negative example: One unit contains two programs. One of the two is assigned to the BackgroundTask and the other to a MotionTask.
  Download in RUN in this and other units is not possible while the MotionTask is active.
- If the program of the modified unit is assigned to a MotionTask: Provide options so that the MotionTask is not active when you want to replace the unit:
  - Avoid continuous loops in MotionTasks! For example, instead of a WHILE loop, use the *_restartTaskId* function.
  - Make provision for an operator function (such as single clock mode) so that you can reset one MotionTask, several MotionTasks, or all MotionTasks. MotionTasks can also be reset via SCOUT as of V4.1.2 (see Manage MotionTasks via SCOUT (as of V4.1.2) (Page 1666)).
- Store the program parts in their own units, if the units run permanently or are called ad hoc. This is particularly advantageous in the case of MotionTasks. A sequential control system in MCC, for example, is a permanently running program part.
  These called program parts can be replaced as long as they are not being accessed.

Error messages if download is not possible

Description

Error messages are output during the download if a change is not possible in RUN, e.g. UPP change in RUN not possible (UPP = User Program Processing). UPP indicates that an error has occurred when a user program is loaded. The old program is retained in the controller, the change is rejected.

Any changes made to the source in the SCOUT project can be easily undone by reloading the current data on the controller to the Engineering System. This ensures the SCOUT project is consistent online once more.
Requirement: Supplementary data have previously been loaded at the same time.

See Loading data from the target device to the programming device (PG)/PC (Page 1673).

Recommended settings for download of changed sources in RUN (as of SIMOTION V4.1)

Overview of settings

Download in RUN can be further improved by the following settings:

- Set the compiler switch "Create program instance data only once"
  The system unit of the execution system is then not affected by changed instance data of a program.
- Use the "BlockInit_OnChange := True;" compiler pragma or as of SCOUT V4.2, an appropriate pragma line in the declaration tables (MCC, LAD/FBD) to allow changed variables to be initialized for download in RUN.

Compiler option "Only create program instance data once"

Description

The program data instantiation and the storage of the program instance data are important in the context of download in RUN.

Compiler option "Only create program instance data once" is set (preferable for download in RUN)

- Instance data associated with programs compiled in this way is created just once, even if the program is used in multiple tasks. The instance data is then in the source or in the program code (for diagnostic purposes, this data is in the symbol browser for the unit and no longer under the execution system).
  If several tasks are assigned to one program, the same instance data is used as the basis for all of them.
  This setting has advantages for a download in RUN as the system unit of the execution system remains unchanged with changed instance data of a program.
- Local program variable data initialization is performed for sequential and cyclic tasks once when the source (unit) is downloaded and the CPU ramps up.
- A compiler pragma or the device properties can be adjusted so that the local variables are initialized on each STOP-RUN transition. See Initialization of data during a STOP - RUN transition (Page 1664).
- Where instance data has been changed, a program within a cyclic or sequential task can even be downloaded in RUN while other tasks (cyclic or sequential) are active.
  Requirement: Allow initialization by compiler pragma BlockInit_OnChange or pragma lines in the declaration tables

NOTICE
Changed behavior for data initialization Changed behavior for the data initialization when "Only create program instance data once" has been selected (see above).

Information on which tasks can run sequentially or cyclically can be found under Execution levels / tasks (Page 1301).

Compiler option "Only create program instance data once" is not activated (default)

- The instance data of all programs is in a central memory area (for diagnostic purposes, this data is in the symbol browser for the execution system). If several tasks are assigned to the same program, then separate program instance data will be created for each task.
  If instance data of a program has been changed, all non-cyclic tasks must be terminated (via central storage in the system unit of the execution system) for activation to be successful. This gives rise to restrictions for download in RUN.
- The data initialization of local program variables is performed:
  - For non-cyclic tasks, at start of task.
  - For cyclic tasks, at STOP-RUN transition.
- A program within a non-cyclic task can also be downloaded in RUN (as long as this task and other non-cyclic tasks are not active) even if the instance data is changed, as the program data is initialized at the start of the task.

Setting of the compiler option

- For the global default of the compiler settings for the program sources, activate **Only create program instance data once** under **Options > Settings > Compiler**.
- For the local setting, when inserting a program source, select **Only create program instance data once** under Compiler in the **Insert xxx** or **Properties** dialog of the source (xxx = ST, MCC, LAD, FBD unit).
  The local settings overwrite the global settings.
  The compiler switch must be placed in RUN before a download (i.e. the source must have been loaded previously with a download in STOP).

See also Influence of the compiler on variable initialization (Page 1419).

Compiler pragma for re-initialization of program instance data for download in RUN

Description

Pragma lines in the declaration tables or the pragma "{BlockInit_OnChange := TRUE;}" in ST sources can initiate a re-initialization of the data with the default values specified in the source in the event of changes to the block structure during a download in RUN.

The pragma can be used in the following cases:

- Changes to TYPE and global variables (VAR_GLOBAL) in the interface and implementation section of a unit
  (alternatively an additional variable block can also be used)
- Changes to instance data of programs (VAR)
  (when the “Only create program instance data once” option is active)

Changes made in VAR_GLOBAL blocks in the INTERFACE section may affect other sources, which will then also need to be loaded. This applies to sources which import the amended source via USES (ST) or are linked with the changed source in the declaration table (MCC, LAD/FBD) and

access variables or POUs of this source. By contrast, only the changed source is affected during the download where changes are made in VAR_GLOBAL blocks in the IMPLEMENTATION section.

Where changes are made in VAR_GLOBAL blocks in the INTERFACE section, any MotionTasks which may be running prevent download in RUN despite the pragmas mentioned above (for details of how to avoid this, see Manage MotionTasks via SCOUT (as of V4.1.2) (Page 1666)).

The pragma may only be placed at the beginning of a block.

Example of the syntax of BlockInit_OnChange:

```
VAR_GLOBAL
    {BlockInit_OnChange := TRUE;}
    Interface_Var_Global1 : INT;
    Interface_Var_Global2 : REAL;
END_VAR
```

NOTICE
Unexpected behavior after initializing data areas in RUN
Initializing data areas in RUN can cause unexpected behavior, as any saved values from the previous task cycle are no longer available.

In LAD/FDB and MCC, initialization can be activated in the declaration tables by inserting pragma lines (context menu); see the relevant LAD/FDB and MCC Programming Manuals. The pragma can be set and deleted as required at any time (see also *Adding pragma lines in variable definition* in the MCC and LAD/FBD Programming Manuals).

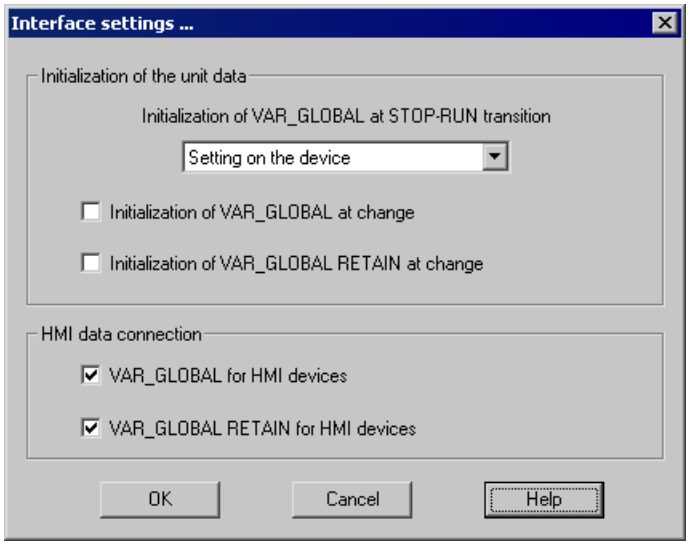


Figure 4-176 Pragma settings for unit variables (interface section)

Example

If it is not possible to perform a download in RUN because of a change to a VAR block within the program, the pragma can be subsequently set in the VAR block. Following compilation, a download in RUN will be possible, because the VAR block will have been reinitialized.

The pragma can be removed again prior to the next download. Where this is not the case, however, re-initialization in RUN is only performed in the event of a further change in the VAR block.

Simple changes to the pragma (add/delete) can be loaded in RUN at any time. You are not required to stop sequential tasks.

Initialization of data during a STOP - RUN transition

Description

If the "Only create program instance data once" compiler option has been activated, data initialization for program variables only occurs during a download or when the CPU is started.

You can set the configuration so that global unit variables (in the interface and implementation section) and program variables are also initialized during a STOP-RUN transition. The initialization is performed immediately before the start of the StartupTask. You therefore always have the ability to perform an initialization of the variables during a STOP-RUN transition:

- As of V4.2 via the Device properties dialog.
- As of V4.1.2 with the compiler pragma "BlockInit_OnDeviceRun" in the Units (ST) or pragma lines in the declaration tables of the sources or programs/charts (MCC and LAD/FBD).

Initialization via the Device properties dialog (as of V4.2)

The **Initialization of the non-RETAIN global variables (VAR_GLOBAL and global device variables) and program variables (VAR) at the transition from STOP to RUN** option in the Properties dialog of a device enables you to trigger initialization immediately before the StartupTask is started.

This global setting can be overwritten locally with a compiler pragma or a pragma line (see below).

Always / never initialize data in ST sources

In ST sources, the initialization can be influenced during a STOP-RUN transition using the compiler pragma "BlockInit_OnDeviceRun" at the start of the variable blocks.

If you use this pragma, the setting on the device is no longer valid for the applicable variable block.

If you set this pragma to "ALWAYS" in the VAR_GLOBAL block or in the VAR block of programs, the variables are initialized at each STOP - RUN transition. The setting is only valid for the variable block in which the pragma is used.

Set the pragma with the "DISABLE" setting to prevent initialization at the STOP - RUN transition in a targeted way.

The compiler issues a warning when the pragma is used at a position where it has no effect.

See also Section "Controlling compiler with attributes" in the SIMOTION ST Programming and Operating Manual.

Example of "always" initialize:

```
VAR
      {BlockInit_OnDeviceRun := ALWAYS;}
      Test_1 : REAL;
      Test_2 : REAL;
  END_VAR
```

Example of "never" initialize (even if, in the device settings, the option "Initialization of the non-RETAIN global variables (VAR_GLOBAL and global device variables) and program variables (VAR) at the transition from STOP to RUN " is active):

```
  VAR_GLOBAL
      {BlockInit_OnDeviceRun := DISABLE;}
      Test_1 : REAL;
      Test_2 : REAL;
  END_VAR
```

Always / never initialize data in MCC or LAD/FBD sources

In MCC or LAD/FBD sources and/or the associated programs/charts, the initialization during a STOP - RUN transition can be influenced by inserting a pragma line in the declaration table. Click the "Pragmas" button and select the appropriate setting under "Initialization ... at the transition from STOP to RUN".

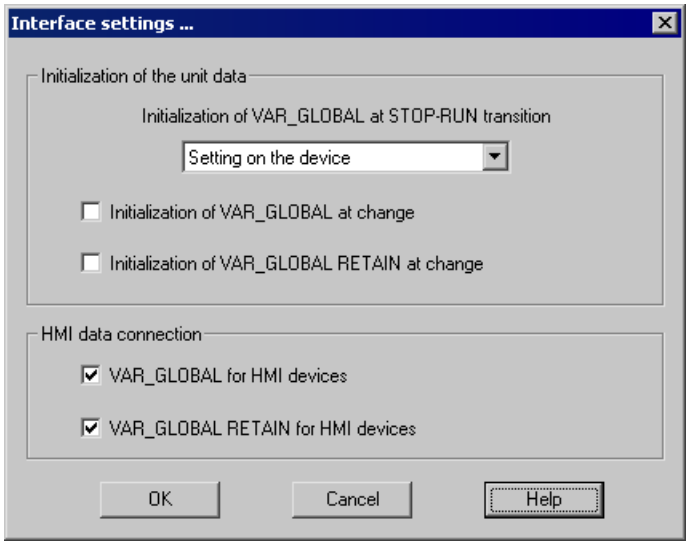


Figure 4-177 Pragma settings for unit variables (interface section)

The default value of "Setting on the device" means:

- As of V4.2: The setting made on the SIMOTION device applies.
- Up to V4.1: Variables are not initialized.

See also section "Adding pragma lines in variable definition" in the MCC Programming and Operating Manual or in the LAD/FBD Programming and Operating Manual.

Download in RUN before programs in Motion Tasks

Description

If you have made changes to sources and want to reload these using download in RUN, an active MotionTask may prevent a change-over time from being found for the interchange of the sources.

Make provision for an operator function (such as single clock mode) so that you can reset one MotionTask, several MotionTasks, or all MotionTasks.
MotionTasks can also be reset via SCOUT as of V4.1.2.

Manage MotionTasks via SCOUT (as of V4.1.2)

Description

You can control MotionTasks from SCOUT even without having created a user program.

In this way you can test programs and directly affect the execution of MotionTasks. To perform a download in RUN, you can terminate MotionTasks individually via SCOUT and restart them after download.

Switching on debug mode and controlling MotionTasks

Requirement

To control MotionTasks, you must be ONLINE and switch to debug mode.

1. Right-click the relevant device and execute **Operating mode** in the context menu.
2. Then select the **Debug mode** option in the **Operating mode** dialog, read the safety instructions, accept these by clicking the option and then clicking **OK**.

Procedure

1. After you have activated the debug mode, call up the device diagnostics via **Target system > Device diagnostics**.
2. Select a MotionTask on the **Task status tab** and right-click it.
   A context menu is displayed.
3. Select **Disable task start** if you want to block the start of the MotionTask. This prevents the task from restarting (TASKSTART_LOCKED in task status).
4. Select **Reset task** if you want to set the task in the STOPPED state. A download in RUN can be performed in this state.

5. Select **Enable task start** if you want to release the start of the task. You can then restart the task from the program or via SCOUT.
6. Select **Start task** if you want to start the MotionTask.

After executing the task control commands, it takes some time before the display in SCOUT is refreshed. You can also click the **Update (F5)** button.

Controlling several MotionTasks simultaneously

1. Select the desired tasks by holding down the Ctrl key and clicking the left mouse button. The tasks are selected.
2. Click the **Control MotionTasks** button and select the corresponding command in the context menu that appears.

Note

The task control commands depend on the system task control commands. You can read the task status in the **Task runtimes** tab under **Task status**.

Behavior when changing the operating mode

If all tasks are enabled when leaving the debug mode, the CPU retains its operating state.

The system displays the following behavior if you intentionally or unintentionally exit the **Debug mode** operating mode and at least one task is disabled.

- If you exit the debug mode before all tasks are enabled, the CPU switches to STOP and the disabling is canceled.
- If you go OFFLINE before all tasks are enabled, the CPU goes into STOP and the disables are cancelled. If you go ONLINE, the CPU switches to process mode again (debug mode is canceled).
- A RUN - STOP - RUN transition does not affect the task status. If the task was disabled before the start, it is also disabled for the start.
- The CPU goes into STOP if SCOUT unintentionally loses communication to the controller. SCOUT goes OFFLINE at the same time.

Download without HW Config information

Download with inconsistent HW Config

As of V4.1.2, it is also possible to perform a download with inconsistent HW Config. In the event of significant changes to the hardware configuration, the download is still canceled and the HW Config also has to be loaded.

The following parameter changes require a download with HW Config

- Logical addresses
  - Address moved in HW Config
  - Adding separate slots to DP nodes or to existing nodes
- Telegram length
  - Telegram length changed
- Cycle clock settings
  - DP cycle time and PN send cycle clock changed

Note

The settings for the telegram length, address and data types are adapted in HW Config in the **DP Slave Properties** dialog box in the **Configuration** tab.
The cycle clock settings are made in the **Isochronous operation** tab.

Setting download without HW Config

You can select this additional option in **Additional CPU options** in the **Load to target system** and **Load to CPU/drive unit** dialogs. You can preset the additional option in **Options > Settings > Download**.

Download of changed technology objects

Description

You can reload changes in the configuration data and system variables of a TO you made offline in RUN (using restart and effective immediately). If necessary, a TO restart is performed after the download.

Downloading TOs

It is possible to download a changed TO under the following circumstances or prerequisites:

- Only technology objects whose structure is retained can be downloaded (see below)
- A download is also performed when a technology object is used by a program, e.g. axis is enabled. If necessary, TO alarms are then output.
- The substitute values of the TO supplied during the download are returned according to the setting of "restart.behaviorInvalidSysvarAccess" (LAST_VALUE, DEFAULT_VALUE). If a restart is performed during a download of TOs, access errors from the user program may occur. The reaction of the CPU or the return value then depends on the settings of the configuration data item "restart.behaviorInvalidSysvarAccess". If this configuration data item is set to STOPPED, the CPU goes to STOP. For more information, refer to System variables (Page 1245).

- If the change to a TO cannot be loaded during the download or an error occurs, the original configuration is retained or re-established (situation as prior to the download).
- The technology objects are loaded in succession and take effect immediately.

Behavior when loading multiple TOs

If you are loading a group of TOs and one of the TOs cannot be downloaded:

- The configuration for already loaded TOs remains active
- The download for the technology object that could not be loaded is rejected.
- The download for the TOs not yet loaded continues

Permitting a TO restart

Before the download is performed, you can use an option to select whether a RESTART of the TOs is to be permitted or not. In this way, you can directly influence a TO RESTART. A RESTART is then executed independently of the setting in the TO in configuration data element RestartCondition.

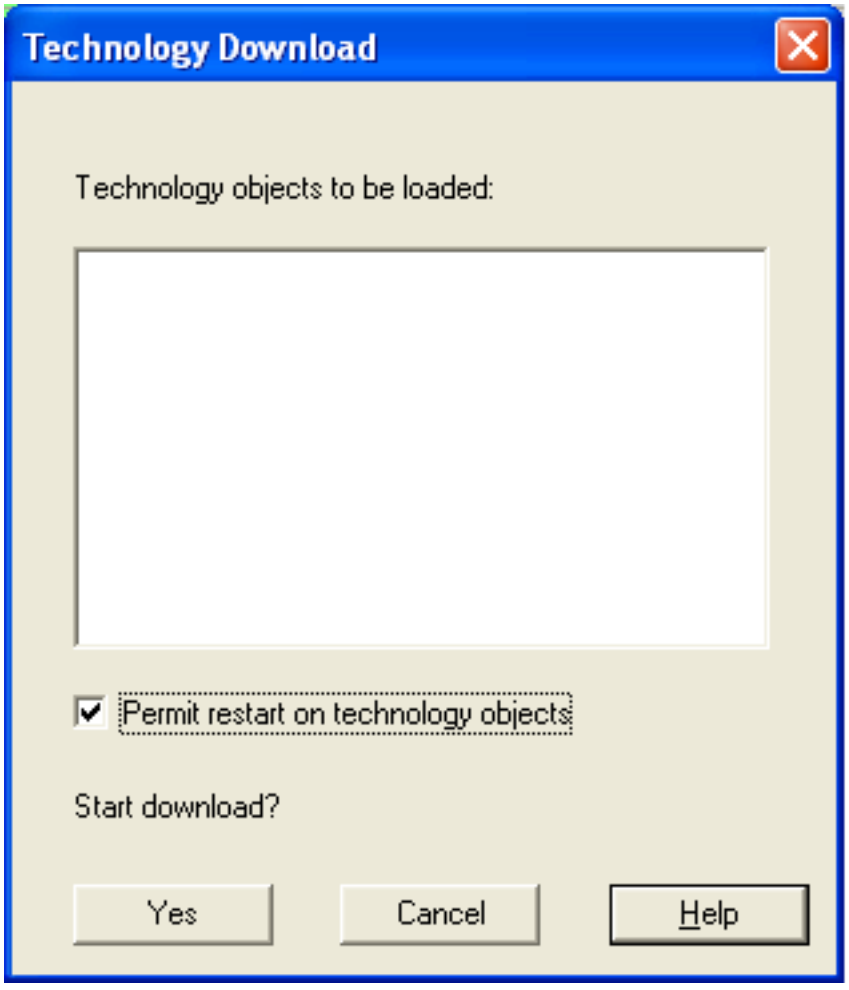


Figure 4-178 TO overview for download

The **Permit restart on technology objects** option is selected by default. If you deselect the option, this setting is saved for further downloads with this project. With configuration data that can only be changed via restart, the TO then behaves as though the download would not have functioned.

Changeable parameters that allow a download in RUN

Configuration data that can be changed via a download always acts on the structure of the TO. In addition, changes to units, interconnections, alarm configuration, and profiles also act on the structure of a TO in such a way that a download in RUN is no longer possible.

In view of this, you are able to change the following data for a download in RUN:

- Immediately effective configuration data
- Configuration data effective at a restart
- System variables

In the expert list of the TOs, the **Effectiveness** column allows you to recognize whether the change in a configuration data item takes effect immediately, requires a restart or is possible only with a download (in STOP). This applies to the configuration data in ONLINE as well as OFFLINE mode.

Copying RAM to ROM in RUN (as of V4.2)

Description

If you have performed a download in RUN, you can also save the changes on the memory card in a powerfail-safe manner.

The Copy RAM to ROM option can be selected beforehand in the download dialog. After a download, this is also possible by means of a separate function (using the **Copy RAM to ROM** button in the toolbar or in the device context menu under Target device).

To preset the option in the download dialog, select **Options > Settings > CPU download**. Here, it can also be specified whether or not the actual values of the TO configuration data are transferred to RAM before the copy RAM to ROM operation.

Note

During "Copy RAM to ROM in RUN", CPU utilization must not be at a critical level (see device diagnostics) as this function generates additional load.

Download with RAM2ROM in RUN in projects with older runtime versions (< V4.1) is not possible and is actively prevented.

Inconsistent TO after a download in RUN

If a TO configuration data element has been changed in the application, it may be the case that the corresponding TO is displayed as inconsistent after a download in RUN.

Scenario with the following settings:

- **Download > Copy RAM to ROM**
- **Settings > CPU download > Accept actual values when copying RAM to ROM**

The configuration data changed by the application will then be in the current data memory. The above settings ensure they are copied from current memory to the RAM and then to the memory card (ROM).
The configured values in the project and the values in the RAM are therefore different and the TO is displayed as inconsistent in the project navigator.

You will then be able to proceed as follows:

- Ignore TO inconsistency
- Analyze the TO inconsistency by means of a project comparison.
- Load the online TO configuration data to the programming device using **Load to PG**.
- Load the offline TO configuration data by performing a download to the device.

See also Copy RAM to ROM (Page 1676).

4.2.12.11 Download direct to memory card or hard disk

With **Load to File System**, you can save the executable project data from SCOUT to the hard disk of the PC/PG or directly to the memory card using a card adapter.

Two methods of saving are available:

- With the "Save normally" option, the user directory is created containing the SIMOTION and SINAMICS subdirectories.
- The "Save compressed" option creates the same content in the form of a Zip file that can be used, for example, for loading via SIMOTION IT Diag.

The loading to the memory card can be viewed the same as a download with subsequent copying of RAM to ROM. This function is very useful, for example, during series commissioning.

Note that the non-volatile data saved on the memory card is overwritten with **Load to file system**. If required, back up the non-volatile data first as described in Memory access (Page 1640).

Procedure

To save the data, SCOUT must be in OFFLINE mode and the device must have been selected in the project navigator for this function to be available.

- Perform **Edit > Load to file system** or **Load to file system** using the context menu of the CPU to save the data of a device on a memory card / CF card or locally on a hard disk.

SCOUT does not automatically recognize the connected read and storage media automatically. The storage/read device must be explicitly selected using **Select target**. An executable project is produced when the project data is saved.

Note

When you save data (either to hard disk or CF card), a USER directory is created automatically so that you can copy data directly to the ROOT directory of a CF card.

⚠ CAUTION

Data loss on the memory card

If you perform **Load to file system**, then the **OEM** folder on the memory card is also deleted. This data is not restored for OEM applications that are not known in the engineering system (are not installed there). You must copy the data manually to the card.

Loading data from the hard disk to the card

You can copy the data from the hard disk to the CF card. As of V4.2, only the contents of the following directories are deleted or overwritten:

- \USER\SIMOTION\RT_DIR
  The subdirectory UPP can be backed up as of version 4.4.
  To do this, check the box "Restore the retentive unit global variables on the device". If the program sources with the compiler option "Permit OPC-UA / -XML (load symbols to RT)" have been compiled, retentive data can be restored by this method. See section "Retaining retentive data" in the "Basic Functions for Modular Machines" Function Manual.
- \USER\SINAMICS\DATA\RT_DIR

The user data in the directories listed below is retained.

Examples of user data saved on the card:

- Backing up retain data (non-volatile data saved using _savePersistentMemoryData) stored in the directory:
  - user\simotion\pmemory.xml
- Backing up IT DIAG user files, settings (e.g. trace.xml), task trace data, log files, and Java files (classes, archives, user file system, etc.), stored in the following directories:
  - user\simotion\hmicfg
  - user\simotion\hmi
- Backing up unit data (data saved on the CF card using _saveUnitDataSet/ _exportUnitDataSet), stored in the following directory:
  - user\simotion\user dir\<unitname>

Note

The stored data must match the firmware of the card. If, for example, you load an old configuration to a card with the latest firmware, you receive error messages during ramp-up.

4.2.12.12 Downloading using the device update tool

Description

SIMOTION offers a convenient solution when updating SIMOTION devices or SIMOTION projects for machine manufacturers and machine operators. Upgrading does not simply mean updating to a higher firmware version; it is a general term for switching to a defined configuration, e.g. a project update. It is also possible to return (restore) to a previous configuration. Update or restore procedures can easily be performed on SIMOTION devices in situ or remotely. The data can be imported to a SIMOTION device via a portable, handy storage medium (e.g. memory stick) or a communication connection.

Note

The firmware of connected SINAMICS drives will only be updated if the automatic firmware update function has been set in parameter p7826 (on every drive unit to be updated).

Detailed information can be found under Updating SIMOTION devices.

4.2.12.13 Retaining retentive data during download

Retentive unit variables and retentive local variables are retained when a download is performed provided that no changes are made to the data blocks.

Please also note the relevant information in section "Retaining retentive data" in the "Modular Machines" Function Manual.

Reorganization of the retentive memory

If the size of the retentive data blocks changes in the new configuration, fragmentation of the retentive memory may make it impossible to perform the download.

The following applies as of version V4.5 of the SIMOTION Kernel:

- The device must be restarted manually
- The retentive memory will then be reorganized.
  The values of the retentive global unit variables are retained if the symbol information is saved in the device. For this purpose, the compiler option "Permit OPC-UA / -XML (load symbols to RT)" must be activated for all relevant program sources during compilation.
- Please also note the relevant information in section "Retaining retentive data" in the "Modular Machines" Function Manual.

The following applies up to version V4.4 of the SIMOTION Kernel:

- You must perform a memory reset on the device.
  The retentive data is lost.

4.2.12.14 Loading data from the target device to the programming device (PG)/PC

With **Load to PG**, the data is loaded from the target device to the PG/PC. In this way, the currently loaded project can be loaded from the target device with all settings to the local hard disk.

Sources or additional data of a project can only be loaded to a programming device when they have previously been loaded to the target system. **Options > Settings > Download > Store additional data on the target device** must have been set in the project for this.

The menu item **Load to PG** is only active when you are ONLINE and a device has been selected.

Note

This function is not suited to restore the original project in SIMOTION SCOUT. Use the **Target system > Load > Copy archived project from card to PG/PC** function for this purpose.

The function is more suited to restore an archived original project, see the following Section "Loading with an existing original project".

Further information on the archiving of a project and saving of the archived project on the memory card can be found in the Online help.

Load to PG with available original project (maintenance scenario)

You are placing commissioned equipment into service, along with a project, or loading the archived project from the memory card of the controller. The project is not consistent online, but is essentially the same as the project that will run on the target device. You can use **Load to PG** when performing the upload to create online consistency between the SCOUT project and the CPU. Afterwards, possible errors are searched for and changes downloaded to the target device.

Note

Objects in the OFFLINE project will be overwritten. As a result, objects which are only present OFFLINE will be deleted. If you want to still be able to access the OFFLINE project after the upload, click **Save before loading to PG** to save the current version of the OFFLINE project. You can retain the OFFLINE project again by exiting the project after the upload without saving it.

Alternative: Archive the OFFLINE project first or save it under another name.

Please note that various functions (e.g. program status) remain inactive and cannot be used until the uploaded objects are saved with the project.

The offline and online projects can also be aligned using the comparison functions (project comparison).

Procedure where there is an existing project

1. Perform **Target system > Load > Load CPU / drive unit to PG**.
   The **Load to PG** dialog box is displayed.
2. Select the option **Load target device to PG** if you want to load all the project data from the target device to the programming device. Note the information in the dialog box.
3. Select the option **Save before loading to PG** if you want to save the project on your PG before uploading.
4. Select the option **Overwrite available libraries** if you want to overwrite the libraries of the project available locally on your programming device.

Only load configuration data to PG

1. Select the **Only load configuration data to PG** option if you want to load the configuration data from the RAM to the programming device.

Transfer current data to RAM

If you want to load the amended configuration data in the current data memory of the target device to the RAM before loading to the programming device, select the **Transfer current data to RAM** checkbox. If you do not transfer the current data to the RAM, it is not loaded to the programming device.

As of V4.2, configuration data is completed during ramp-up using parameter values of the SINAMICS drive (adaptation). For more detailed information, see **Adaptation** under Symbolic assignment - introduction (Page 1189). When "Load to PG" is executed, adapted values are detected and "Transfer current data to RAM" is automatically preselected as well.

Loading to the programming device via a project comparison (object granular)

- With the project comparison functionality, you can also load the data for individual objects to the programming device. For more detailed information, please refer to the online help on project comparison or the sections relating to project comparison in the SIMOTION manual.

See also

Overview of the data download (Page 1643)

4.2.12.15 Copy current data to RAM

Description

Use **Copy current data to RAM** to copy the current configuration data, changed during the RUN mode, to the RAM. Depending on the data item, changes to the configuration data during RUN take effect in different ways (e.g. immediately or after a restart of the technology object).

After the restart on the technology object, the changed configuration data is stored in the current data memory. To copy the values from the current data memory to the RAM, you must execute this function explicitly. Only after copying is the data in the RAM, and is then uploaded at the next ramp-up and takes effect. See also Overview of the memory in the target device (Page 1637).

Note

So that the project data in SCOUT is consistent with the project data in the target system if configuration data has been changed online, you must perform an upload to the PG/PC (menu **Target system > Load > Configuration data to PG**).

Adaptation of configuration data

When SIMOTION devices ramp up, reference variables, along with drive and encoder data for SINAMICS S120 (SINAMICS Integrated), are transferred automatically for the configuration data of the SIMOTION technology objects "TO axis" and "TO external encoder". Therefore, this data no longer has to be entered in SIMOTION. The adapted data is located in the current data memory. When **Copy current data to RAM** is executed, these values are added to the RAM. This makes the technology object inconsistent online. To resolve this inconsistency, you also need to load the configuration data to the PG. This is detected in the dialog and the appropriate preselections are made. If you have changed configuration data online yourself, this will also be loaded to the PG. For more detailed information on adaptation, see Symbolic assignment - introduction (Page 1189).

Transferring current data to RAM

1. Select **Target system > Copy current data to RAM**.
   The **Copy current data to RAM** dialog opens. The dialog indicates whether the configuration data has been changed as a result of adaptation.
2. Select one of the following options:
   – Copy RAM to ROM
   – Load to PG
3. Click **Yes** to trigger the copying process and any options selected.

For more setting options to trigger **Copy current data to RAM** select:

- **Load to PG** dialog
- **Options > CPU download >Copy RAM to ROM > Transfer current values to RAM**

See also

Copy RAM to ROM (Page 1676)

4.2.12.16 Copy RAM to ROM

Description

Select **Copy RAM to ROM** to save the project from the volatile memory (RAM) to the retentive memory (ROM) of the memory card. A power failure must be avoided when writing to the memory card.

For a detailed description, see Overview of the memory in the target device (Page 1637).

Possible errors when copying from RAM to ROM

Errors can occur when copying from RAM to ROM Possible causes:

- An access error has occurred.
- Rotating measurement (p1960) is activated.

- Insufficient storage space is available. Check the memory card and restart the action.
- The ramp up of the drive unit has not yet completed, check the operating state of the diagnostic overview.

Note

Check the entries in the diagnostic buffer and the **target system output**.

Copying RAM to ROM with adapted configuration data

When SIMOTION devices ramp up, reference variables, along with drive and encoder data for SINAMICS S120 (SINAMICS Integrated), are transferred automatically for the configuration data of the SIMOTION technology objects "TO axis" and "TO external encoder". Therefore, this data no longer has to be entered in SIMOTION. The adapted data is located in the current data memory, **not** in the RAM.

When **Copy RAM to ROM** is executed, the system detects whether values have been adapted. **Copy current data to RAM** and **Load configuration data to the PG** are also preselected in order to back up this configuration data to the memory card (ROM) and to carry out alignment with the engineering project as well. See also Copy current data to RAM (Page 1675).

See also

Copying RAM to ROM in RUN (as of V4.2) (Page 1670)

4.2.12.17 Deleting user data on the memory card

You will need to delete the user data on the memory card, for example:

- If you want to save a different (new) project to the memory card and, therefore, delete any user data relating to an "old project" (e.g. unit data sets) which might be stored on the memory card.
- If you want to delete the technology packages on the memory card, e.g.:
  - If you want to use a smaller technology package (e.g. CAM instead of CAM_EXT)
  - To force an upgrade of the technology package **within a version** in line with a new hotfix or service pack

    Note

    User data does not have to be deleted in the event of a change of version. In such cases, the technology packages on the memory card are always updated.

- You can delete the user data with SIMOTION SCOUT. As of V4.2, you can select which data you want to delete.

This means you can still access the SIMOTION module online with your PG/PC. The licenses on the memory card are retained.

Deleting user data

1. In SIMOTION SCOUT, open the project you want to edit.
2. Go online with the module.
3. Select the module in the project navigator and then select **Target system > Delete user data on card**.
   The **Delete user data on card** dialog appears. You can select which data you want to delete in this dialog:
   - Project data (programs, technology objects, TPs, SINAMICS Integrated, etc.)
   - Unit data sets
   - Non-volatile data (overall reset)
   - Archived project
4. Activate the options which you want to delete.
5. Click **OK** to delete the data.

Note

Non-volatile data and unit data sets may only be deleted together with project data.

4.2.13 Error sources and efficient programming

4.2.13.1 Error sources during programming

Error sources during programming

Explanations of the most important error sources associated with programming are provided below. The section also provides you with solution possibilities for eliminating these error sources.

Checking data types when assigning arithmetic expressions

In arithmetic expressions, the result is always calculated in the largest number format contained in the expression.

An expression value can only be assigned to a variable in the following cases:

- The calculated expression and the variable to be assigned are of the same data type.
- The data type of the calculated expression can be implicitly converted to the data type of the variable to be assigned.

Therefore, if you want to assign an expression to a variable:

- Ensure that the variable is of a large enough data type.
- Perform an explicit data type conversion for that part of the expression that is too large (see Functions for the conversion of numeric data types and bit data types (Page 1483)). However, as a result information could be lost, for example, if the value range is decreased or the accuracy is reduced, as is the case for conversion from LREAL to REAL.
- If applicable, specify the data type explicitly for numbers (e.g. UINT#127, if the number 127 is to be of data type UINT instead of USINT).

Checking start of functions in cyclic tasks every time

Technology object functions, such as functions for positioning axes, should only be started in cyclic tasks if they are not already running. Otherwise, the commands in your program are executed every time the cycle is repeated.

You can program a condition for the technology object function start in cyclic tasks, e.g. depending on the contents of an auxiliary variable, which is set when the command is executed.

Table 4-139 Example for correct TO function start in a cyclic task

```
//...
IF myStart = 0 THEN       // If auxiliary variable has not yet been set
    myStart := 1;        // Set auxiliary variable (function started)
    myCommandID := _getCommandId ();
    myFC := _pos (axis := myAxis, // Execute function
             position        := position_1,
             nextCommand     := IMMEDIATELY,
             commandID       := myCommandID);
END_IF;
//...
IF myAxis.positioningState.actualPosition = position_1 THEN
    myStart := 0;         // Reset auxiliary variable, if
                          // function execution required.
END_IF;
//...
```

Wait times in cyclic tasks

If you link command transitions to conditions when using system functions in cyclic tasks, e.g. for the *_pos* command, a timeout can occur, causing a CPU stop.

This can occur in any system function where the *nextCommand* parameter assumes a value other than IMMEDIATELY, e.g. the value WHEN_MOTION_DONE.

The timeout occurs if the cycle time configured in SIMOTION SCOUT is exceeded as a result of a step enabling condition or programmed wait times, e.g. using *_waitTime*.

Note

Do not use commands with wait times in cyclic tasks, e.g. *_waitTime*.

Use only input parameter **nextCommand := IMMEDIATELY** for system functions in cyclic tasks.

Using the commandId parameter correctly

All TO commands must contain a parameter for command identification, see **Function parameters of technology functions.**

Prior to calling the appropriate command you can obtain a project-wide unique command ID with the command *_getCommandId*. Save the command ID in a local variable and use it as a parameter in the technology object command; alternatively, use the function call in *commandId:=_getCommandId()* directly as a parameter.

Note

If the CommandId is not assigned for the command call, the default value of the parameter takes effect with (0.0).

The CommandID is not only used by TO commands.

You must use this unique command ID to check the status of the motion command, e.g. to check the status of a positioning motion with *_getStateOfAxisCommand*. The system can only uniquely identify a motion command from its command ID!

Table 4-140 Example of using a TO function with command identification

```
//...
VAR
    myCommandID                 : commandIdType;
    myState                     : StructRetCommandState;
END_VAR
//...
myCommandID := _getCommandId ();
// Save unique ID
myFC := _pos (axis := myAxis,
// Execute function with ID
             position           := position_1,
             nextCommand     := IMMEDIATELY,
             commandID          := myCommandID);
myState := _getStateOfAxisCommand (axis:=myAxis,
             commandID          := myCommandID);
// Status check
IF myState.commandIdState = WAITING_FOR_SYNC_START THEN
    ;
//...
END_IF;
//...
```

See also

Function parameters of the technology functions (Page 1228)

_getCommandId function (Page 1541)

_getSyncCommandId function (Page 1542)

CommandID overview (Page 1541)

Locating errors (ST programs)

The following error message can occur during compilation:

Error 7000, 7010, 7011, or 7014 in Line ...

A syntax error has occurred. Possible causes:

- *Incorrectly terminated control structures (e.g. END_IF missing)*
- *Statements not terminated with ;*
- *Missing parentheses*

First, check the specified line to see whether the error actually occurred there, i.e. perhaps you have not terminated a control structure, or have omitted a semicolon at the end of a line, or a parenthesis is missing.

If you cannot find any of the specified errors, you must locate the error:

1. Comment out the program block-by-block before the line containing the error message, i.e. enclose the selected section between the character pair (* and *). The line with the error message must not be commented out.
2. Recompile the program.
3. If an error message still appears after steps 1 and 2, the commented-out section is too small. Expand it until the error message disappears.
4. When the error message no longer appears, the error is contained in the commented-out section. Now reduce the size of the commented-out section line by line and recompile the program each time until the error message appears again. The last enabled line is the line with the error message.

Note

You can consult a list of all compiler error messages in the appendix of the ST Programming Manual.

Errors on download

If an error message occurs when your program is downloaded, the log is held in the SIMOTION SCOUT detail view. Look for the error cause in the log. Check your hardware configuration or the program, e.g. for addresses that do not exist.

For more information about hardware configuration and addressing, refer to the SIMOTION SCOUT configuration manual.

Error messages during the download that contain the abbreviation UPP (User Program Processing) occur when a user program is being processed, e.g. when changes cannot be performed in RUN.

CPU does not switch to RUN

If the CPU switches back to STOP mode immediately after you have started your programs, check the device diagnostics and alarm window in SCOUT. The causes of the STOP are entered in the diagnostics buffer. The alarms of the technology objects that can also cause a CPU STOP are displayed in the Alarms window.

CPU goes to STOP

Description

If the CPU changes to STOP mode then check the device diagnostics and the alarm window in the SCOUT. The causes of the STOP are entered in the diagnostics buffer. The alarms of the technology objects that can also cause a CPU STOP are displayed in the Alarms window.

Possible reasons, why a CPU can go into STOP are:

- Incorrect direct I/O access
- Configuration data access or system variable access to TO if in RESTART (as of V4.1, however, substitute values for system variables are possible; see System variables (Page 1245))
- Missing PeripheralFaultTask (if there is incorrect access to the process image)
- Missing TechnologicalFaultTask (if there are errors on the technology object)
- Processing error in the programs (or missing ExecutionFaultTask)
- Monitoring time overflow of the IPO/servo tasks
- Monitoring time overflow of the BackgroundTask
- Monitoring overflow of a TimerInterruptTask
- Sign-of-life monitoring Simotion - drive
- Technology object alarms with STOP response (e.g. 20001)
- Set mode using system variable

Displaying and changing the operating mode

Use the device system variable *modeOfOperation* to display or change the current operating mode.

Syntax example

```
modeOfOperation  :EnumDeviceModeOfOperation  //readable, writable,
immediately effective
EnumDeviceModeOfOperation:
    [
     _STOP |
     _RUN
    ]
```

For example, with this, you can switch a SIMOTION CPU that has gone into STOP into RUN again from a local HMI by writing the device system variables.

Detailed information about the system variable *modeOfOperation* can be found in the *System Functions/Variables of the Devices* Manual or in the online help.

See also

Execution errors in programs (Page 1256)

Access errors to system variables and configuration data, as well as I/O variables for direct access (Page 1258)

SystemInterruptTasks (Page 1334)

Possible errors in technology objects (Page 1281)

Checking and setting system clocks

Often, incorrectly set system cycle clocks (servo cycle clocks, interpolator cycle clocks, PWM cycle clock) cause the SIMOTION device to go into STOP mode.

Check the runtimes of the SynchronousTasks (ServoSynchronousTask, ServoSynchronousTask_fast, IPOsynchronousTask, IPOsynchronousTask_fast, IPOsynchronousTask_2, tasks for the TControl technology package). It could be that the user programs or system programs for individual tasks need more time than is set in the system cycle clocks in SIMOTION SCOUT.

Also try to minimize the runtime of the SynchronousTasks. Shift the programs to MotionTasks as much as possible, and split up your programs accordingly, if necessary.

Make sure there is an integer ratio between individual tasks. Otherwise, low-priority SynchronousTasks will not be started at periodic intervals.

Deactivate unnecessary tasks.

Note

The system tasks for the TControl technology package can be disabled in the system cycle clock settings window.

Refer to the online help for information about how to check the device diagnostics, interpret the alarm window and check/change your system clocks.

System functions and a Task Trace are available for checking the runtimes. The Task Trace can be used to graphically display the sequence of individual tasks and user events (generated per program command); see Task Trace function manual.

See also Isochronous data processing (Page 1385) , Functions for message programming (AlarmS) (Page 1452).

Comparing REAL or LREAL variables

When you compare REAL variables or LREAL variables (also corresponding system variables, e.g. axis position) with one another, you should never use "=". Because of the different internal number representation, the compared numbers are never identical. Instead, you should evaluate, for example, the direction of motion and use ">" or "<" or the system variable for "Position window reached".

Sequential task is interrupted

Description

Sequential tasks (MotionTasks) can assume different states, including TASK_STATE_WAITING. The status can be read out in the diagnostics display of the CPU.

If you do not know why the task is waiting, you must carry out an extensive search to identify the point in the program at which the task is waiting on a condition.

You use the following functions to find these points:

- Display program run function
- Display code position (e.g. line of an ST source file) that runs a MotionTask

For detailed information refer to Section *Program run* in the *ST programming manual*.

Checking for range violations

Range violations, i.e. the violation of range limits for a data type, are not signaled by the compiler. Therefore, if you use variables in arithmetic operations, you should always check for possible range violations.

Table 4-141 Example of checking for a range violation

```
PROGRAM myRange
    VAR
        a,b  : SINT := 100;
        c: SINT;
    END_VAR

    c := a + b;
    // If c is outside range, then exit program.
    IF (a > 0) AND (c < a) AND (c < b) OR
       (a < 0) AND (c > a) AND (c > b) THEN
       // Range violation
       RETURN;
    ELSE
        ; // OK
    END_IF;
END_PROGRAM
```

Setting size of local data stack

When configuring the execution system, set the size of the reserved local data stack for each task in the *Task configuration* tab, see Description of the respective user program task (Page 1314).

Use the "Program structure" function to obtain information about the memory requirements of a program with all called program organization units (FCs and FBs) on the stack, see the SIMOTION ST Programming Manual. You will also find information on the memory requirement of variables on the local data stack in this manual.

During configuration, ensure that there is sufficient spare. For example, temporary additional storage may be required on the local data stack during downloading in RUN.

The download operation will be aborted with an error message if the reserved local data stack exceeds the available memory space in the RAM.

The program will be aborted during execution if the reserved local data stack is insufficient for program execution.

In the event of an error, check the following:

- RAM utilization in the device diagnostics
- The sources (programs, etc.) for large data structures (arrays, structures), e.g. using the cross-reference list
- Size of local data stack for the task

See also

Specifications for the configuring (Page 1417)

Sources of errors during multitasking

The main sources of errors during multitasking are:

- Using FB instances in different tasks
- Using global variables in different tasks

Using FB instances and global variables in different tasks carries the risk of this data not being processed correctly. If global variables are used in different tasks, a change of task and processing in a different task may cause an unwanted change in the value, for example.

4.2.13.2 Efficient programming

Efficient programming - overview

The following discrepancy always occurs in many control systems and/or the associated programming environments:

- Well structured and concise user programs with special emphasis on modifiability and expandability do not perform optimally with regard to runtime.
- On the other hand, runtime-optimized programs are difficult to expand or modify.

The following description provides information for runtime-optimizing programming and for change-optimizing programming. Depending on the task or with local optimizations, you should focus on one of these options.

Runtime-optimized programming

The following description contains information about runtime-optimized programming. Note that instructions for runtime-optimizing programming are often at odds with the rules of structured programming.

Optimizing access to inputs and outputs

Access to the process image of cyclic tasks is significantly faster than direct access to inputs or outputs (see Direct access and process image of the cyclic tasks in the ST Programming Manual). Therefore, you should assign an I/O variable to the process image of the task in which the variable is used. In addition to quicker access, another advantage of this is that the I/O variable will be consistent during the entire runtime of the task.

Optimizing access to system variables

It takes considerably longer to access system variables (see System variables (Page 1245)) than to access variables that are stored in the dynamic memory (local variables, non-retentive unit variables).

Therefore, only a few instances of direct access to system variables are possible in high-speed cyclic tasks (e.g. SynchronousTasks). For this reason, when a system variable is accessed many times, it is recommended that you copy the entire system variable structure at the beginning of a cycle (program in the cyclic task) to a local variable of the corresponding system data type. Then access this variable during the program cycle.

The data types for declaring the local variables can be found in the List Manuals for SIMOTION technology objects (TOs).

Optimal variable declaration

Arrange the variables within a declaration block (e.g. VAR/END_VAR) in order of ascending size. This enables you to make optimal use of the memory space.

Initialize variables with values other than 0 only when necessary. Variable initialization at the start of a task or POU requires time. This is especially the case for temporary variables and variables of programs that are assigned to sequential tasks.

Optimizing access to function block parameters

When creating function blocks (FBs) that are to process values, you can select one of two methods. You can assign input variables in the FB with VAR_INPUT, process the variables there, and assign the results to output parameters with VAR_OUTPUT.

If a large data volume is to be transferred to the FB, it can be faster to use in/out parameters (VAR_IN_OUT) than to use input and output parameters (VAR_INPUT and VAR_OUTPUT). Parameters can be transferred more quickly because copying is eliminated, however, it could take longer to access the variable from the FB under certain conditions.

Note

When you access unit variables or global device variables with in/out parameters, note that other tasks can access these variables at the same time.

Optimizing program structure

Make sure that your ST source files and the POUs they contain are structured clearly and concisely. However, avoid modularizing your source files too much, since it takes longer to access functions and variables of imported units than functions and variables within a unit.

Optimizing the execution system

Assign the IPOsynchronousTask to a single program only. This reduces the risk of a timeout during runtime.

Use functions that are executed synchronously in MotionTasks only. (During synchronous processing, the next command is not executed until processing of the pending command satisfies a condition, e.g. it is complete.)

Make sure that not too many MotionTasks are active simultaneously.

For more efficient programming of cyclic tasks (primarily the BackgroundTask) go to sequence controlled programming. This means you consciously control the program flow via case decisions and only run through the portions of code that are relevant in the current status (e.g. via CASE statement).

Moreover edge triggered program instead of level-triggered programming is recommended.

Do not run through the relevant code cyclically (if the level of the condition is TRUE) but rather just once. This is done by querying the satisfied condition on rising edge (new status of the condition is TRUE, the old status on the other hand was FALSE).

Use of USEPACKAGE for multiple technology packages

Syntax for the use of "USEPACKAGE" for several technology packages

If you want to use more than one technology package simultaneously, you can select it again using the following syntax. See also Using technology functions in a program (Page 1223).

Table 4-142 Formal description

```
usepackagestatement ::= 'USEPACKAGE' packagelist ';'
packagelist         ::= packageentry {',' packageentry }
packageentry        ::= simplepackname | namespacepackname
simplepackname      ::= packagename
namespacepackname ::= packagename 'AS' namespaceident
//packagename and namespaceident must be valid identifiers
```

Change-optimized programming

The following description contains information about change-optimized programming.

For information about downloading in RUN, please see Download of changed sources in RUN (Page 1656) .

Notes on the draft program

Description

Useful notes on the following, for example:

- How best to structure the program
- How best to use the different tasks (MotionTask, BackgroundTask, IPO task, etc.)
- How best to divide your program between the different tasks

can be found in the application guidelines for SIMOTION.

You can find the application guidelines for SIMOTION in **SIMOTION Utilities & Applications**, which is included in the scope of delivery of SIMOTION SCOUT (under **Documentation > Application guidelines for SIMOTION**).

Generic programming

Description

Program axes in arrays and loops. This offers advantages, especially when new axes are being added or errors corrected (only once in the loop and not separately for each axis), and results in a more compact code. The readability is significantly improved.

With MCC and LAD/FDB sources, program initialization can also be activated using a pragma line in the declaration tables ("BlockInit_OnChange" pragma).

You can also information on axis arrays under **Documentation > Application guidelines for SIMOTION** in the section titled **Generic programming** in SIMOTION Utilities & Applications, which is part of the SIMOTION SCOUT scope of delivery.

Declaring retentive variables in one unit

Declare all retentive variables in the interface section in one single unit. This has the following advantage:

- It enables you to make optimal use of limited memory space.
- The variables are only initialized if the interface section in this unit has been changed.
- As of V4.1 you can also use multiple declaration blocks, see Use multiple VAR_GLOBAL, VAR_GLOBAL RETAIN blocks (Page 1419) .

HMI variables in one unit

Declare variables that are exported to HMI devices (see HMI (Human Machine Interface) coupling (Page 1611)). In the case of larger projects or strictly delimited software modules, a separate HMI unit per module is appropriate.

- The HMI project (e.g. WinCC flexible) must only be downloaded again if the interface section of this unit has been changed.
- You also have the option of disabling the consistency check during commissioning (**Options > Settings > Download**), though you do so at your own risk. Please note that, as there is no check for inconsistencies (e.g. for valid hardware addresses), access to processes may go unmonitored.
- You can also mark HMI relevant data, see Marking HMI relevant data (Page 1422).

See also HMI (Human Machine Interface) connection (Page 1611).

Using device global variables versus unit global variables

Description

Use of global unit variables in sources is preferable to use of device global variables (via the project navigator).

Benefits:

- Variable structures can be used.
- Initialization (initial values) of the variables for the STOP-RUN transition is possible (via program in StartupTask)
- For newly created global unit variables a download in RUN is also possible (in a new declaration block)

Procedure

1. Define unit-global variables in a source file in the interface section.
   Retain variables and HMI variables should likewise each be declared in a separate source file or declaration block (see above).
2. Assign the program with the initialization of the variables to the StartupTask.
   The variables will be set to a defined initial value in the startup.

Parameter | I/O symbols | Structures | Enumerations | Connections

	Structure name	Element name	Data type	Array length	Initial value
1	MyStruct	IntValue	INT		0
2		RealValue	REAL		0.0
3		BitValue	BOOL		FALSE
4					

Figure 4-179 Example of the declaration of structures in an MCC source file

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Array length	Initial value
1	m	VAR_GLOBAL CONSTANT	INT		16
2	BitArray	VAR_GLOBAL	BOOL	m	16(False)
3	IntArray	VAR_GLOBAL	INT	m	16(0)
4	ReadArray	VAR_GLOBAL	REAL	m	16(0.0)
5	StructVar	VAR_GLOBAL	MyStruct		
6					

Figure 4-180 Example of the declaration of parameters in an MCC source file

Table 4-143 Example declaration and program in an ST source file (unit)

```
INTERFACE
    //global types
    TYPE
       MyStruct : STRUCT
                   Intvalue : INT;
                   Realvalue: REAL;
                   Bitvalue : BOOL;
                  END_STRUCT
    END_TYPE
    //global constants
    VAR_GLOBAL CONSTANT
        n : INT := 0;
        m: INT := 15;
    END_VAR
    //global variables
    VAR_GLOBAL
        Bitarray : ARRAY [n..m] OF BOOL := [16 (FALSE)];
        Intarray : ARRAY [0..15] OF INT := [16 (0)    ];
        Realarray: ARRAY [0..15] OF REAL:= [16 (0.0)  ];
        StructVar: MyStruct;
    END_VAR
    //programs
    PROGRAM Init;
    //end of the interface
END_INTERFACE

IMPLEMENTATION
 PROGRAM Init
  ////////////////////////////////////////////////
  //initialization of the variables during startup//
  ////////////////////////////////////////////////
  StructVar.Intvalue :=0;
  StructVar.Realvalue:=0;
  StructVar.Bitvalue :=FALSE;
 END_PROGRAM
END_IMPLEMENTATION
```

The program is then assigned to the StartupTask.

Whenever unit global variables are to be used in a different source, the sources that contain the declaration must be combined (USES). See also **Connection to other program sources or to libraries** in the MCC Programming Manual or **Use of the USES statement in the interface or implementation section of an importing unit** in the ST programming manual.

Centralized starting and resetting of MotionTasks

To enhance programming clarity, program starting and resetting of MotionTasks in one centralized location.

4.2.14 Appendix

4.2.14.1 Symbolic constants

The following table presents names of reserved constants that you must not use for individual variable names.

Table 4-144 Symbolic constants of the Taskstartinfo

Symbolic constant	Data type	Value
_SC_ALARM_CONFIGURATION	UDINT	404
_SC_ARRAY_BOUND_ERROR_READ	UDINT	502
_SC_ARRAY_BOUND_ERROR_WRITE	UDINT	503
_SC_BACKGROUND_TIMER_OVERFLOW	UDINT	300
_SC_CYCLE_TIMER_OVERFLOW	UDINT	301
_SC_DEVICE_COMMAND	UDINT	401
_SC_DIAGNOSTIC_INTERRUPT	UDINT	201
_SC_DIVISION_BY_ZERO	UDINT	500
_SC_DP_CLOCK_DETECTED	UDINT	207
_SC_DP_SLAVE_NOT_SYNCHRONIZED	UDINT	210
_SC_DP_SLAVE_SYNCHRONIZED	UDINT	209
_SC_DP_SYNCHRONIZATION_LOST	UDINT	208
_SC_DRIVE_OBJECT_FAULT	UDINT	217
_SC_DRIVE_OBJECT_ALARM	UDINT	218
_SC_EXCEPTION	UDINT	403
_SC_EXTERNAL_COMMAND	UDINT	402
_SC_IMAGE_UPDATE_FAILED	UDINT	204
_SC_IMAGE_UPDATE_OK	UDINT	206
_SC_INVALID_ADDRESS	DINT	–1
_SC_INVALID_FLOATING_POINT_OPERATION	UDINT	501
_SC_IO_MODULE_NOT_SYNCHRONIZED	UDINT	215
_SC_IO_MODULE_SYNCHRONIZED	UDINT	214
_SC_MODE_SELECTOR	UDINT	400
_SC_PC_INTERNAL_FAILURE	UDINT	205
_SC_PULL_PLUG_INTERRUPT	UDINT	216
_SC_PROCESS_INTERRUPT	UDINT	200
_SC_STATION_DISCONNECTED	UDINT	202
_SC_STATION_RECONNECTED	UDINT	203
_SC_TO_INSTANCE_NOT_EXISTENT	UDINT	506
_SC_VARIABLE_ACCESS_ERROR_READ	UDINT	504
_SC_VARIABLE_ACCESS_ERROR_WRITE	UDINT	505

Table 4-145 Symbolic constants of the task status

Symbolic constant	Data type	Hex notation
TASK_STATE_INVALID	DWORD	16#0000
TASK_STATE_LOCKED	DWORD	16#0100
TASK_STATE_RUNNING	DWORD	16#0004
TASK_STATE_STOP_PENDING	DWORD	16#0001
TASK_STATE_STOPPED	DWORD	16#0002
TASK_STATE_SUSPENDED	DWORD	16#0020
TASK_STATE_WAIT_NEXT_CYCLE	DWORD	16#0040
TASK_STATE_WAIT_NEXT_INTERRUPT	DWORD	16#0080
TASK_STATE_WAITING	DWORD	16#0010

Table 4-146 Symbolic constants for alarm messages

Symbolic constant	Data type	Hex notation
ALARMS_ERROR	DWORD	16#8000
ALARMS_QSTATE	DWORD	16#0100
ALARMS_STATE	DWORD	16#0001
DSC_SVS_DEVICE_ALARMS_EVENT_ID_IN_USE	DWORD	16#0006
DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED	DWORD	16#0005
DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID	DWORD	16#0001
DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR	DWORD	16#0009
DSC_SVS_DEVICE_ALARMS_IV_CALL	DWORD	16#0004
DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL	DWORD	16#0007
DSC_SVS_DEVICE_ALARMS_IV_SFC_TYP	DWORD	16#0008
DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED	DWORD	16#0002
DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED	DWORD	16#0003
DSC_SVS_DEVICE_ALARMS_NO_ENTRY	DWORD	16#0010

Table 4-147 Symbolic constants for the value range limits of elementary data types

Symbolic constant	Data type	Value	Hex notation
SINT#MIN	SINT	-128	16#80
SINT#MAX	SINT	127	16#7F
INT#MIN	INT	-32768	16#8000
INT#MAX	INT	32767	16#7FFF
DINT#MIN	DINT	-2147483648	16#8000_0000
DINT#MAX	DINT	2147483647	16#7FFF_FFFF
USINT#MIN	USINT	0	16#00
USINT#MAX	USINT	255	16#FF

Symbolic constant	Data type	Value	Hex notation
UINT#MIN	UINT	0	16#0000
UINT#MAX	UINT	65535	16#FFFF
UDINT#MIN	UDINT	0	16#0000_0000
UDINT#MAX	UDINT	4294967295	16#FFFF_FFFF
T#MIN TIME#MIN	TIME	T#0ms	16#0000_0000[1]
T#MAX TIME#MAX	TIME	T#49d_17h_2m_47s_295ms	16#FFFF_FFFF[1]
TOD#MIN TIME_OF_DAY#MIN	TOD	TOD#00:00:00.000	16#0000_0000[1]
TOD#MAX TIME_OF_DAY#MAX	TOD	TOD#23:59:59.999	16#0526_5BFF[1]
[1] Internal display only			

Table 4-148 Symbolic constants for invalid values of selected data types

Symbolic constant	Data type	Meaning
STRUCTALARMID#NIL	StructAlarmId	Invalid AlarmId
STRUCTTASKID#NIL	StructTaskId	Invalid TaskId
TO#NIL	ANYOBJECT	Invalid technology object

4.2.14.2 Identifiers with defined meaning in SIMOTION

This section contains an alphabetical listing of identifiers with a predefined meaning in SIMOTION. The list contains:

- The reserved and protected identifiers of the SIMOTION ST (Structured Text) programming language
- The reserved and protected identifiers of other programming languages
- The general standard functions and standard function blocks with the associated data types
- The functions for task control, runtime measurement and message programming with the associated data types
- The system functions and system variables of SIMOTION devices with their associated data types
- The technology object data types
- The system functions, system variables (and configuration data) of technology packages

The response when using this identifier is different, e.g.:

- Compiler error with reserved identifiers
- Possible masking and resulting inability to obtain the identifiers
  Under certain circumstances, the compiler may not issue a warning if, for example, the associated technology package is not imported.

Icons

_abortAllReadWriteDriveParameterJobs

_abortReadWriteRecordJobs

_activateConfiguration

_activateDpSlave

_activateDpSlaveAddress

_activateNameOfStation

_activateTo

_adaptAxisConfigData

_adaptExternalEncoderConfigData

_AdditionObjectType

_addPointToCam

_addPolynomialSegmentToCam

_addSegmentToCam

_addSegmentToCam_V2_0

_alarmS

_alarmSc

_alarmScId

_alarmSId

_alarmSq

_alarmSqId

_allocateTokenToVariableName

_AND

_BOOL

_bufferActuatorCommandId

_bufferAdditionObjectCommandId

_bufferAxisCommandId

_bufferAxisCommandId_V3_1

_bufferCamCommandId

_bufferCamTrackCommandId

_bufferControllerObjectCommandId

_bufferExternalEncoderCommandId

_bufferExternalEncoderCommandId_V3_1

_bufferFixedGearCommandId

_bufferFollowingObjectCommandId

_bufferFollowingObjectCommandId_V3_1
_bufferFormulaObjectCommandId
_bufferMeasuringInputCommandId
_bufferOutputCamCommandId
_bufferPathObjectCommandId
_bufferSensorCommandId
_BYTE_FROM_8BOOL
_BYTE_TO_8BOOL
_calculateTControllerParameter
_CamTrackType
_cancelAxisCommand
_cancelExternalEncoderCommand
_cancelFollowingObjectCommand
_cancelPathObjectCommand
_changeEnableModeOfAdditionObjectIn
_changeEnableModeOfControllerObjectIn
_changeEnableModeOfFormulaObjectIn
_changeEnableOfFormula
_changeOperationMode
_checkEqualTask
_checkExistingUnitDataSet
_configurationManagement
_continue
_continue_V3_1
_continuePath
_continuePath_V4_2
_ControllerObjectType
_copyTControllerShadow
_cpuData
_cpuDataRW
_DATE
_DATE_AND_TIME
_deactivateDpSlave
_deactivateTo
_deallocateTokenToVariableName

_defineFormula
_deleteAllUnitDataSets
_deleteUnitDataSet
_DINT
_disableAdditionObjectIn
_disableAxis
_disableAxis_V2_0
_disableAxis_V3_1
_disableAxisAdditiveTorque
_disableAxisInterface
_disableAxisSimulation
_disableAxisTorqueLimitNegative
_disableAxisTorqueLimitPositive
_disableCamming
_disableCamming_V3_0
_disableCamTrack
_disableCamTrackSimulation
_disableCommandToActual
_disableControllerObject
_disableControllerObjectIn
_disableExternalEncoder
_disableExternalEncoderSimulation
_disableFixedGearing
_disableFixedGearMotionIn
_disableFollowingObjectSimulation
_disableFollowingObjectSimulation_V3_1
_disableForceLimiting
_disableFormula
_disableFormulaObjectIn
_disableGearing
_disableGearing_V3_0
_disableMaster_V3_0
_disableMeasuringInput
_disableMeasuringInputSimulation
_disableMonitoringOfEncoderDifference

_disableMovingToEndStop
_disableOutputCam
_disableOutputCamSimulation
_disablePathObjectSimulation
_disableQFAxis
_disableQFAxis_V3_0
_disableQFAxis_V3_1
_disableScheduler
_disableSensor
_disableTorqueLimiting
_disableVelocityGearing
_disableVelocityLimiting
_disableVelocityLimiting_V4_3
_driveStates
_DWORD_FROM_2WORD
_DWORD_FROM_4BYTE
_DWORD_TO_2WORD
_DWORD_TO_4BYTE
_enableAdditionObjectIn
_enableAxis
_enableAxis_V2_0
_enableAxis_V3_1
_enableAxis_V3_2
_enableAxisAdditiveTorque
_enableAxisInterface
_enableAxisSimulation
_enableAxisTorqueLimitNegative
_enableAxisTorqueLimitPositive
_enableCamming
_enableCamming_V3_0
_enableCamTrack
_enableCamTrackSimulation
_enableCommandToActual
_enableControllerObject
_enableControllerObjectIn

_enableDistributedMotionDelayValueCalculation
_enableDpInterfaceSynchronizationMode
_enableExternalEncoder
_enableExternalEncoderSimulation
_enableFixedGearing
_enableFixedGearMotionIn
_enableFollowingObjectSimulation
_enableFollowingObjectSimulation_V3_1
_enableForceControlByCondition
_enableForceControlByCondition_V2_1
_enableForceControlByCondition_V3_1
_enableForceControlByCondition_V4_0
_enableForceLimitingByCondition
_enableForceLimitingByCondition_V4_0
_enableForceLimitingValue
_enableForceLimitingValue_V3_1
_enableFormula
_enableFormulaObjectIn
_enableGearing
_enableGearing_V3_0
_enableGearing_V3_1
_enableMaster_V3_0
_enableMeasuringInput
_enableMeasuringInputCyclic
_enableMeasuringInputCyclic_V3_2
_enableMeasuringInputSimulation
_enableMonitoringOfEncoderDifference
_enableMotionInPositionLockedForceLimitingProfile
_enableMotionInPositionLockedVelocityLimitingProfile
_enableMotionInPositionLockedVelocityLimitingProfile_V4_0
_enableMotionInPositionLockedVelocityLimitingProfile_V4_3
_enableMovingToEndStop
_enableMovingToEndStop_V2_0
_enableOutputCam
_enableOutputCamSimulation

_enablePathObjectSimulation
_enablePathObjectTracking
_enablePathObjectTrackingSuperimposed
_enablePositionLockedForceLimitingProfile
_enablePositionLockedForceLimitingProfile_V3_1
_enablePositionLockedVelocityLimitingProfile
_enablePositionLockedVelocityLimitingProfile_V4_0
_enableQFAxis
_enableQFAxis_V3_0
_enableQFAxis_V4_0
_enableScheduler
_enableSensor
_enableTimeLockedForceLimitingProfile
_enableTimeLockedForceLimitingProfile_V3_1
_enableTimeLockedVelocityLimitingProfile
_enableTimeLockedVelocityLimitingProfile_V4_0
_enableTimeLockedVelocityLimitingProfile_V4_3
_enableTorqueLimiting
_enableTorqueLimiting_V2_0
_enableVelocityGearing
_enableVelocityLimitingValue
_enableVelocityLimitingValue_V4_0
_enableVelocityLimitingValue_V4_3
_exportUnitDataSet
_exportUnitDataSet_01
_exportUserDataSet
_FALSE
_FB_MF_MultiAxis
_FB_MF_SingleAxis
_FB_MF_STGP
_FB_STGP_BasicMC
_FB_STGP_Cam
_FB_STGP_Cam_Ext
_FB_STGP_Gear
_FB_STGP_Path

_FB_STGP_Position
_finite
_firstIndexOf
_FixedGearType
_forceTControllerIdentification
_FormulaObjectType
_getActiveDpSlaveAddress
_getActiveNameOfStation
_getActuatorErrorNumberState
_getActuatorErrorState
_getAdditionObjectErrorNumberState
_getAdditionObjectErrorState
_getAlarmId
_getAverageTaskIdRunTime
_getAverageTaskRunTime
_getAxisDataSetParameter
_getAxisDataSetParameter_V3_1
_getAxisDataSetParameter_V4_1
_getAxisErrorNumberState
_getAxisErrorState
_getAxisInternalPosition
_getAxisSpecificState
_getAxisSpecificState2
_getAxisStoppingData
_getAxisUserPosition
_getBit
_getCamErrorNumberState
_getCamErrorState
_getCamFollowingDerivative
_getCamFollowingMinMax
_getCamFollowingValue
_getCamLeadingValue
_getCamSpecificState
_getCamSpecificState2
_getCamTrackErrorNumberState

_getCamTrackErrorState
_getCamTrackSpecificState
_getCircularPathData
_getCircularPathData_V4_2
_getCircularPathGeometricData
_getCommandId
_getConfigurationData
_getControllerObjectErrorNumberState
_getControllerObjectErrorState
_getCurrentTaskIdRunTime
_getCurrentTaskRunTime
_getDataByToken
_getDeviceId
_getDoIndexNumberFromLogAddress
_getDpStationAddressFromLogDiagnosticAddress
_getExternalEncoderErrorNumberState
_getExternalEncoderErrorState
_getExternalEncoderSpecificState
_getExternalEncoderSpecificState2
_getFixedGearErrorNumberState
_getFixedGearErrorState
_getFollowingObjectErrorNumberState
_getFollowingObjectErrorState
_getForceControlDataSetParameter
_getForceControlDataSetParameter_V3_1
_getForceControlDataSetParameter_V4_1
_getFormulaObjectErrorNumberState
_getFormulaObjectErrorState
_getGearAxisSpecificState
_getGearAxisSpecificState2
_getGeoAddressFromLogAddress
_getInOutByte
_getInternalTaskIdIdx
_getInternalTaskIdx
_getInternalTimeStamp

_getIO_Part_4
_getIPConfig
_getLinearPathData
_getLinearPathData_V4_2
_getLinearPathGeometricData
_getLogDiagnosticAddressFromDpStationAddress
_getLogicalAdressOfIoVariable
_getMasterValue
_getMaximalTaskIdRunTime
_getMaximalTaskRunTime
_getMeasuringInputErrorNumberState
_getMeasuringInputErrorState
_getMeasuringInputSpecificState
_getMeasuringInputSpecificState2
_getMemoryCardId
_getMinimalTaskIdRunTime
_getMinimalTaskRunTime
_getMotionStateOfAxisCommand
_getMotionStateOfFollowingObjectCommand
_getMotionStateOfPathObjectCommand
_getNextLogAddress
_getOutputCamErrorNumberState
_getOutputCamErrorState
_getOutputCamSpecificState
_getOutputCamSpecificState2
_getPathAxesData
_getPathAxesPosition
_getPathCartesianData
_getPathCartesianPosition
_getPathGeometricData
_getPathObjectBcsFromOcsData
_getPathObjectErrorNumberState
_getPathObjectErrorState
_getPathObjectOcsFromBcsData
_getPendingAlarms

_getPnInterfacePortNeighbour
_getPNSyncCounter
_getPolynomialPathData
_getPolynomialPathData_V4_2
_getPolynomialPathGeometricData
_getPosAxisSpecificState
_getPosAxisSpecificState2
_getProgrammedTargetPosition
_getQFAxisDataSetParameter
_getQFAxisDataSetParameter_V3_1
_getSafeValue
_getSegmentIdentification
_getSensorErrorNumberState
_getSensorErrorState
_getSlaveValue
_getStateOfActuatorCommand
_getStateOfAdditionObjectCommand
_getStateOfAllDpSlaves
_getStateOfAllDpStations
_getStateOfAxisCommand
_getStateOfCamCommand
_getStateOfCamTrackCommand
_getStateOfControllerObjectCommand
_getStateOfDiagnosticDataCommand
_getStateOfDpSlave
_getStateOfExternalEncoderCommand
_getStateOfFixedGearCommand
_getStateOfFollowingObjectCommand
_getStateOfFormulaObjectCommand
_getStateOfIO
_getStateOfMeasuringInputCommand
_getStateOfMotionBuffer
_getStateOfOutputCamCommand
_getStateOfPathObjectCommand
_getStateOfPathObjectMotionBuffer

_getStateOfProcessInterruptCommand
_getStateOfRecordCommand
_getStateOfSensorCommand
_getStateOfSingleDpSlave
_getStateOfTask
_getStateOfTaskId
_getStateOfTo
_getStateOfUnitDataSetCommand
_GetStateOfXCommand
_getStationType
_getSyncCommandId
_getSystemTime
_getTaskId
_getTimeDifferenceOfInternalTimeStamp
_getTimeDifferenceOfInternalTimeStamps
_homing
_imData
_IMPLEMENTATION
_importUnitDataSet
_importUnitDataSet_01
_importUserDataSet
_INT
_INTERFACE
_INTERFACE_AND_IMPLEMENTATION
_internalTest
_interpolateCam
_isNaN
_lastIndexOf
_lengthIndexOf
_LINT
_loadUnitDataSet
_loadUnitDataSet_01
_LREAL
_MC_CamIn
_MC_CamIn_V4_0

_MC_CamInMode
_MC_CamOut
_MC_CamSwitch
_MC_CamSwitchDirection
_MC_CamSwitchOff
_MC_Direction
_MC_DisableMode
_MC_EnableMode
_MC_EncoderMode
_MC_GearIn
_MC_GearInMode
_MC_GearOut
_MC_Home
_MC_HomingMode
_MC_Jog
_MC_MoveAbsolute
_MC_MoveAdditive
_MC_MoveRelative
_MC_MoveSuperimposed
_MC_MoveVelocity
_MC_Phasing
_MC_PositionProfile
_MC_Power
_MC_Power_V4_0
_MC_ReadActualPosition
_MC_ReadAxisError
_MC_ReadBoolParameter
_MC_ReadBoolParameter_V4_0
_MC_ReadParameter
_MC_ReadParameter_V4_0
_MC_ReadStatus
_MC_Reset
_MC_Stop
_MC_StopMode
_MC_VelocityProfile

_MC_WriteBoolParameter
_MC_WriteBoolParameter_V4_0
_MC_WriteParameter
_MC_WriteParameter_V4_0
_modifyCountES
_modifyCountRT
_move
_movePathCircular
_movePathCircular_V4_1
_movePathCircular_V4_3
_movePathLinear
_movePathLinear_V4_1
_movePathLinear_V4_3
_movePathPolynomial
_movePathPolynomial_V4_1
_movePathPolynomial_V4_3
_movePointToPoint
_movePointToPoint_V4_1
_MRES
_NOT
_OR
_PathAxis
_PathObjectTpye
_pos
_readDiagnosticData
_readDriveFaults
_readDriveMultiParameter
_readDriveMultiParameterDescription
_readDriveParameter
_readDriveParameterDescription
_readRecord
_readSystemClock
_readVariableDiagnosticData
_readVariableRecord
_REAL

_redefineExternalEncoderPosition
_redefinePathObjectOcs
_redefinePosition
_releaseSemaphore
_removeBufferedActuatorCommandId
_removeBufferedAdditionObjectCommandId
_removeBufferedAxisCommandId
_removeBufferedCamCommandId
_removeBufferedCamTrackCommandId
_removeBufferedControllerObjectCommandId
_removeBufferedExternalEncoderCommandId
_removeBufferedFixedGearCommandId
_removeBufferedFollowingObjectCommandId
_removeBufferedFormulaObjectCommandId
_removeBufferedMeasuringInputCommandId
_removeBufferedOutputCamCommandId
_removeBufferedPathObjectCommandId
_removeBufferedSensorCommandId
_reset_AllAlarmId
_resetActuator
_resetActuatorConfigDataBuffer
_resetActuatorError
_resetAdditionObject
_resetAdditionObjectConfigDataBuffer
_resetAdditionObjectError
_resetAlarmId
_resetAllAlarmId
_resetAxis
_resetAxis_V2_0
_resetAxis_V3_0
_resetAxisConfigDataBuffer
_resetAxisError
_resetAxisError_V3_0
_resetCam
_resetCam_V2_0

_resetCam_V3_0
_resetCam_V4_2
_resetCamConfigDataBuffer
_resetCamError
_resetCamError_V3_0
_resetCamTrack
_resetCamTrackConfigDataBuffer
_resetCamTrackError
_resetControllerObject
_resetControllerObjectConfigDataBuffer
_resetControllerObjectError
_resetDriveObjectFault
_resetExternalEncoder
_resetExternalEncoder_V2_0
_resetExternalEncoder_V3_0
_resetExternalEncoderConfigDataBuffer
_resetExternalEncoderError
_resetExternalEncoderError_V3_0
_resetFixedGear
_resetFixedGearConfigDataBuffer
_resetFixedGearError
_resetFollowingObject
_resetFollowingObject_V2_0
_resetFollowingObject_V3_0
_resetFollowingObject_V3_1
_resetFollowingObjectConfigDataBuffer
_resetFollowingObjectError
_resetFollowingObjectError_V3_0
_resetFormulaObject
_resetFormulaObjectConfigDataBuffer
_resetFormulaObjectError
_resetMeasuringInput
_resetMeasuringInput_V2_0
_resetMeasuringInput_V3_0
_resetMeasuringInputConfigDataBuffer

_resetMeasuringInputError
_resetMeasuringInputError_V3_0
_resetMotionBuffer
_resetOutputCam
_resetOutputCam_V2_0
_resetOutputCam_V3_0
_resetOutputCamConfigDataBuffer
_resetOutputCamError
_resetOutputCamError_V3_0
_resetPathObject
_resetPathObjectConfigDataBuffer
_resetPathObjectError
_resetPathObjectMotionBuffer
_resetSensor
_resetSensorConfigDataBuffer
_resetSensorError
_resetTask
_resetTaskId
_resetTController
_resetTControllerError
_resetTechnologicalErrors
_resetUnitData
_restartTask
_restartTaskId
_resumeTask
_resumeTaskId
_RETAIN
_retriggerTaskControlTime
_retriggerTaskIdControlTime
_RUN
_runMotionInPositionLockedForceProfile
_runMotionInPositionLockedVelocityProfile
_runPositionBasedMotionIn
_runPositionBasedMotionIn_V3_2
_runPositionLockedForceProfile

_runPositionLockedVelocityProfile
_runTimeLockedForceProfile
_runTimeLockedForceProfile_V3_1
_runTimeLockedPositionProfile
_runTimeLockedVelocityProfile
_runVelocityBasedMotionIn
_runVelocityBasedMotionIn_V3_2
_S7_COUNTER
_S7_TIMER
_saveConfigData
_savePersistentMemoryData
_saveUnitDataSet
_saveUnitDataSet_01
_SC_ALARM_CONFIGURATION
_SC_ARRAY_BOUND_ERROR_READ
_SC_ARRAY_BOUND_ERROR_WRITE
_SC_BACKGROUND_TIMER_OVERFLOW
_SC_BATTERY_WARNING_LEVEL1
_SC_BATTERY_WARNING_LEVEL2
_SC_CYCLE_TIMER_OVERFLOW
_SC_DEVICE_COMMAND
_SC_DIAGNOSTIC_INTERRUPT
_SC_DIVISION_BY_ZERO
_SC_DP_CLOCK_DETECTED
_SC_DP_SLAVE_NOT_SYNCHRONIZED
_SC_DP_SLAVE_SYNCHRONIZED
_SC_DP_SYNCHRONIZATION_LOST
_SC_DRIVE_OBJECT_ALARM
_SC_DRIVE_OBJECT_FAULT
_SC_EXCEPTION
_SC_EXTERNAL_COMMAND
_SC_IMAGE_UPDATE_FAILED
_SC_IMAGE_UPDATE_OK
_SC_INVALID_ADDRESS
_SC_INVALID_FLOATING_POINT_OPERATION

_SC_IO_MODULE_NOT_SYNCHRONIZED
_SC_IO_MODULE_SYNCHRONIZED
_SC_MODE_SELECTOR
_SC_PC_INTERNAL_FAILURE
_SC_PROCESS_INTERRUPT
_SC_PULL_PLUG_INTERRUPT
_SC_STATION_DISCONNECTED
_SC_STATION_RECONNECTED
_SC_TO_INSTANCE_NOT_EXISTENT
_SC_VARIABLE_ACCESS_ERROR_READ
_SC_VARIABLE_ACCESS_ERROR_WRITE
_sendProcessInterrupt
_SensorType
_SERVICE
_setAndGetEncoderValue
_setAxisDataSetActive
_setAxisDataSetParameter
_setAxisDataSetParameter_V3_1
_setAxisDataSetParameter_V4_1
_setAxisSTW
_setAxisSTW_V4_0
_setBit
_setCammingOffset
_setCammingOffset_V1_1
_setCammingOffset_V3_1
_setCammingScale
_setCammingScale_V1_1
_setCammingScale_V3_1
_setCamOffset
_setCamScale
_setCamTrackState
_setControllerObjectPIDControl
_setDataByToken
_setDeviceErrorLED
_setDpSlaveAddress

_setDriveObjectSTW
_setExternalEncoderValue
_setFixedGearingOffset
_setFixedGearMaster
_setForceCommandValue
_setForceCommandValue_V3_1
_setForceControlDataSetParameter
_setForceControlDataSetParameter_V3_1
_setForceControlDataSetParameter_V4_1
_setFormula
_setFormulaObjectOutputValue
_setGearingOffset
_setGearingOffset_V3_1
_setIO_Part_4
_setIPConfig
_setIpoClockBorderToPosition
_setMaster
_setMaster_V3_0
_setMasterValue_V3_0
_setModeSelfAdaptingConfiguration
_setNameOfStation
_setOutputCamCounter
_setOutputCamState
_setOutputCamState_V4_0
_setPathObjectOcs
_setQFAxisDataSetParameter
_setQFAxisDataSetParameter_V3_1
_setQFAxisFCharacteristics
_setQFAxisQCharacteristics
_setSafeValue
_setSystemTime
_setTControllerActualIdentificationType
_setTControllerControlRangeParameter
_setTControllerCycleParameter
_setTControllerCycleParameterSecondary

_setTControllerDPIDParameter
_setTControllerDPIDParameterSecondary
_setTControllerIdentificationModifiedTangentMethodParameter
_setTControllerIdentificationModifiedTangentMethodProcessParameter
_setTControllerIdentificationStandardTangentMethodParameter
_setTControllerIdentificationStandardTangentMethodProcessParameter
_setTControllerInputDisplayValueParameter
_setTControllerInputFilterParameter
_setTControllerInputGradientCheckParameter
_setTControllerInputLimitCheckParameter
_setTControllerIntegrator
_setTControllerIntegratorSecondary
_setTControllerLowerPlausibilityParameter
_setTControllerLowerPlausibilityParameterSecondary
_setTControllerManualOutputValue
_setTControllerOperatingMode
_setTControllerProcessModeParameter
_setTControllerPWMParameter
_setTControllerPWMParameterSecondary
_setTControllerSetpoint
_setTControllerUpperPlausibilityParameter
_setTControllerUpperPlausibilityParameterSecondary
_SHUTDOWN
_SINT
_sizeOf
_startSyncCommands
_startTask
_startTaskId
_STARTUP
_startupData
_stop
_stop_V3_0
_stop_V4_2
_stopEmergency
_stopEmergency_V3_0

_stopEmergency_V4_2
_stopPath
_stopPath_V4_1
_STOPU
_StructCamTrackArrayOfSingleCamSettings
_StructCamTrackSingleCamSettings
_suspendTask
_suspendTaskDebug
_suspendTaskId
_suspendTaskIdDebug
_synchronizeDpInterfaces
_synchronizeExternalEncoder
_taskTraceStart
_taskTraceStop
_taskTraceUserEvent
_taskTraceWriteout
_tcpCloseConnection
_tcpCloseServer
_tcpOpenClient
_tcpOpenServer
_tcpReceive
_tcpSend
_testAndSetSemaphore
_testSFBSysDataInit
_TIME
_TIME_OF_DAY
_toggleBit
_triggerCommissioningMode
_triggerTestMode
_TRUE
_UDINT
_udpAddMulticastGroupMembership
_udpDropMulticastGroupMembership
_udpReceive
_udpSend

A

ACTIVATED
ACTIVATED_NO_ALARM
activationModeChangedConfigData
ACTIVE
ACTIVE_ABSOLUTE
ACTIVE_AND_WAITING_FOR_CAM_TRACK_OUTPUT_INACTIVE
ACTIVE_AND_WAITING_FOR_DATA
ACTIVE_AND_WAITING_FOR_NEXT_CYCLE
ACTIVE_HOMING
ACTIVE_RELATIVE
ACTIVE_WITH_CIRCULAR_MOVE
ACTIVE_WITH_CONSTANT_LIMITS
ACTIVE_WITH_DYNAMIC_ADAPTION
ACTIVE_WITH_POLYNOMIAL_MOVE
ACTIVE_WITH_VARIABLE_LIMITS
ACTIVE_WITHOUT_DYNAMIC_ADAPTION
activeCam
activeMaster
ACTOR_ADAPTION_DATA
actorData
actorMonitoring
ACTUAL
ACTUAL_ACTIVATED
ACTUAL_AND_DEFAULT_VALUE
ACTUAL_DIRECTION_PASSIVE
ACTUAL_VALUE
ACTUAL_VALUE_INSIDE_POSITIONING_WINDOW
ACTUAL_VALUE_OUT_OF_POSITIONING_WINDOW
actualCycleData
ActualDPIDData
actualIdentificationModifiedTangentMethodData
actualIdentificationStandardTangentMethodData
actualIdentificationType
actualInputData
actualInputLimitCheckData

actualInputState
actualTorque
actualValueDefault
actualValueIn
ADAPTED_AND_DIFFERENT
ADAPTED_AND_EQUAL
ADAPTION_ERROR
ADD
ADD_DT_TIME
ADD_TIME
ADD_TOD_TIME
additionalSensorData
additionResult
ADDITIVE
additiveTorque
additiveTorqueIn
ADVANCED_TYPE
ALARM_S
ALARM_SQ
ALARMS_ERROR
ALARMS_QSTATE
ALARMS_STATE
ALL
ALL_ADAPTION_DATA
ALL_AXIS_MOTION
ALL_EDGES
ALL_ERRORS
ALL_GLOBAL
ALL_ID
ALL_POSITION_RELATED_MOTION
ANALOG
ANALOG_TEMPERATURE
AND
ANDN
ANYOBJECT

ANYOBJECT_TO_OBJECT
AnyType_to_BigByteArray
ANYTYPE_TO_LITTLEBYTEARRAY
APPLICATION
APPROACH_MOVE
APPROACH_NEGATIVE_PASSIVE
APPROACH_POSITIVE_PASSIVE
ARRAY
AS
ASIN
ASSEMBLY_BASE_DRIVE
ASSEMBLY_BASE_EXTERN
ASSEMBLY_BASE_LINEAR
ASSEMBLY_BASE_LOAD
AT
AT_DECELERATION_START
AT_END_OF_COMMAND
AT_MOTION_START
AT_PROFILE_START
AT_SYNCHRONIZING_START
AT_THE_END_OF_CAM_CYCLE
ATAN
ATTACHED_STEADILY
AUTOMATIC_INTERFACE_SYNCHRONIZATION
AUTOMATICALLY
AVAILABLE
AVERAGING
AXIS_HOMED
AXIS_STOPPED_AT_POSITION

B

B_SPLINE
BASIC
BASIC_AND_SUPERIMPOSED_MOTION_ACTIVE
BASIC_AND_SUPERIMPOSED_POS_MOTION_ACTIVE

BASIC_MOTION
BASIC_MOTION_ACTIVE
BASIC_MOTION_ACTIVE_WITH_LENGTH_OUTPUT
BASIC_NON_POS_AND_SUPERIMPOSED_POS_MOTION_ACTIVE
BASIC_POS_AND_SUPERIMPOSED_NON_POS_MOTION_ACTIVE
BASIC_POS_MOTION_ACTIVE
basicMotion
Baudrate_1
Baudrate_2
Baudrate_3
Baudrate_4
Baudrate_5
BCD_TO_BYTE
BCD_TO_DINT
BCD_TO_DWORD
BCD_TO_INT
BCD_TO_LWORD
BCD_TO_SINT
BCD_TO_WORD
bcs
BE_SYNCHRONOUS_AT_POSITION
BEGIN_SYNC
BEGIN_TO_STOP_WHEN_POSITION_REACHED
BigByteArray_to_AnyType
BIGBYTEARRAY_TOANYTYPE
BIN_CODE
BOOL
BOOL_TO_BYTE
BOOL_TO_DWORD
BOOL_TO_WORD
BOOL_VALUE_TO_DINT
BOOL_VALUE_TO_INT
BOOL_VALUE_TO_LREAL
BOOL_VALUE_TO_REAL
BOOL_VALUE_TO_SINT

BOOL_VALUE_TO_UDINT
BOOL_VALUE_TO_UINT
BOOL_VALUE_TO_USINT
BOTH
BOTH_DIRECTION
BOTH_EDGES
BOTH_EDGES_FIRST_FALLING
BOTH_EDGES_FIRST_RISING
BUFFERED
BY
BY_CAM_TRACK_END
BY_CENTER_AND_ARC
BY_COMMAND
BY_END_POSITION
BY_PROFILE_END
BY_START_POSITION
BY_STW_BIT
BY_VALUE
BYTE
BYTE_TO_BCD
BYTE_TO_BOOL
BYTE_TO_DINT
BYTE_TO_DWORD
BYTE_TO_INT
BYTE_TO_SINT
BYTE_TO_UDINT
BYTE_TO_UINT
BYTE_TO_USINT
BYTE_TO_WORD
BYTE_VALUE_TO_LREAL
BYTE_VALUE_TO_REAL

C

C_SPLINE
CACHE_ID

CAL
CALC
CALCN
CAM_AND_ZM_PASSIVE
CAM_DATA_RESET
CAM_PASSIVE
CAMMING
cammingAdjustments
CAMTRACK_DISABLE
camTrackPosition
CamType
CASE
CHANGE_DS_IS_LOCKED
CHANGE_FAILED
CHANNEL_0
CHANNEL_1
CHANNEL_2
CHANNEL_3
CHANNEL_4
CHANNEL_5
CHANNEL_6
CHANNEL_7
CIRCULAR
circularPathCommand
CLAMP_BY_FOLLOWING_ERROR_DEVIATION
CLAMP_WHEN_TORQUE_LIMIT_REACHED
CLOSE_ON_EXIT
CMD_CONTINUEAXIS
CMD_DISABLEAXIS
CMD_DISABLEAXISSIMULATION
CMD_ENABLEAXIS
CMD_ENABLEAXISSIMULATION
CMD_ENABLEMEASURINGINPUT
CMD_HOMING
CMD_INIT

CMD_MOVE
CMD_POSABS
CMD_POSREL
CMD_RESETAXIS
CMD_SETDATASETPARAM
CMD_STOPAXIS
CMD_STOPEMERGENCY
COLLECT_CHANGED_CONFIG_DATA
COMMAND_DONE
COMMAND_FAILED
COMMAND_ID
COMMAND_NOT_FOUND
COMMAND_VALUE
COMMAND_VALUE_BASIC_MOTION
COMMAND_VALUE_SUPERIMPOSED_MOTION
CommandIdType
COMPLETE_RANGE
CONCAT
CONCAT_DATE_TOD
CONDITION_1
CONDITION_1_AND_CONDITION_2
CONDITION_1_OR_CONDITION_2
CONDITION_2
CONFIG_TO_SHADOW
CONFIGURATION
CONFIGURATION_DELETED
CONFIGURATION_ERROR
CONFIGURATION_ID_NOT_FOUND
CONFIGURATION_NOT_FOUND
CONFIGURATION_VERSION
CONFIGURED
CONFIGURED_OUTPUT_VALUE
CONNECTED
connection
CONSTANT

CONSTANT_FORCE
CONSTANT_MOVE
CONTINUOUS
control
CONTROL_STOP
controlDeviation
controllerOutput
controlRangeParameter
COS
counterCamData
counterMeasuredValue1
counterMeasuredValue2
CRITICAL
CTD
CTD_DINT
CTD_LINT
CTD_UDINT
CTD_ULINT
CTU
CTU_DINT
CTU_LINT
CTU_UDINT
CTU_ULINT
CTUD
CTUD_DINT
CTUD_LINT
CTUD_UDINT
CTUD_ULINT
CURRENT
CURRENT_MODE
currentMasterData
currentSlaveData
currentSyncPosition
cycleParameter
cycleParameterSecondary

D

DEPENDING_ON_OUTER_INPUT_LIMITCHECK_STATUS
derivedValue
DESYNCHRONIZING
DIFFERENTIATION
DINT
DINT_TO_BCD
DINT_TO_BYTE
DINT_TO_DWORD
DINT_TO_INT
DINT_TO_LREAL
DINT_TO_REAL
DINT_TO_SINT
DINT_TO_STRING
DINT_TO_UDINT
DINT_TO_UINT
DINT_TO_USINT
DINT_TO_WORD
DINT_VALUE_TO_BOOL
DINTIn1
DINTIn1Default
DINTIn2
DINTIn2Default
DINTIn3
DINTIn3Default
DINTIn4
DINTIn4Default
DINTOut1
DINTOut1Default
DINTOut2
DINTOut2Default
DINTOut3
DINTOut3Default
DINTOut4
DINTOut4Default
DIRECT

DIRECT_HOMING
DIRECT_HOMING_RELATIVE
DIRECT_OUTPUT
DIRECT_TO_SHADOW
DIRECT_VALUE
DISABLE
DISABLE_ALL
DISABLE_CONTROLLER
DISABLE_DRIVE_IMMEDIATELY
DISABLE_MEASURE_AND_AVARAGE_OUTPUT_VALUE
DISABLE_MEASURE_AND_MANUAL_OUTPUT_VALUE
DISABLE_MOTION
DISABLE_OUTPUT
disableCommand
DISCONTINUOUS
distributedMotion
DISTURBED
DIV
DIVTIME
DO
DO_NOT_CHANGE
DO_NOT_CLAMP
DO_NOT_CLOSE_ON_EXIT
DO_NOT_SET_DP_ALARM
DONE
DP_1
DP_2
DP_3
DP_CLOCk_DETECTED
DP_INTERFACES_SYNCHRONIZED
DP_SLAVE_NOT_SYNCHRONIZED
DP_SLAVE_SYNCHRONIZED
DP_TEL1_STANDARD
DP_TEL101_611U_VELCTRL_NO_ENCODER
DP_TEL102_611U_POSCTRL_1_ENCODER

DP_TEL103_611U_POSCTRL_2_ENCODER
DP_TEL105_611U_DSC_1_ENCODER
DP_TEL106_611U_DSC_2_ENCODER
DP_TEL2_STANDARD
DP_TEL3_STANDARD
DP_TEL4_STANDARD
DP_TEL5_STANDARD
DP_TEL6_STANDARD
DP_TEL81_STANDARD
DP_TEL83_STANDARD
DPIDParameter
DPIDParameterSecondary
DPMASTER
DRIVE
DRIVE_INTERFACE
DriveAxis
driveData
DSC_SVS_DEVICE_ALARMS_EVENT_ID_IN_USE
DSC_SVS_DEVICE_ALARMS_EVENT_ID_NOT_USED
DSC_SVS_DEVICE_ALARMS_ILLEGAL_EVENT_ID
DSC_SVS_DEVICE_ALARMS_INTERNAL_ERROR
DSC_SVS_DEVICE_ALARMS_IV_CALL
DSC_SVS_DEVICE_ALARMS_IV_FIRST_CALL
DSC_SVS_DEVICE_ALARMS_IV_SFC_TYP
DSC_SVS_DEVICE_ALARMS_LAST_ENTRY_USED
DSC_SVS_DEVICE_ALARMS_LAST_SIGNAL_USED
DSC_SVS_DEVICE_ALARMS_NO_ENTRY
DT
DT_TO_DATE
DT_TO_TOD
DWORD
DWORD_TO_BCD
DWORD_TO_BOOL
DWORD_TO_BYTE
DWORD_TO_DINT

E

END_EXPRESSION
END_FOR
END_FUNCTION
END_FUNCTION_BLOCK
END_IF
END_IMPLEMENTATION
END_INTERFACE
END_LABEL
END_OF_INTERPOLATION
END_OF_MOTION_STOP
END_POINT
END_PROGRAM
END_REPEAT
END_RESOURCE
END_STEP
END_STRUCT
END_SYNC
END_TRANSITION
END_TYPE
END_VAR
END_WAITFORCONDITION
END_WHILE
ENDAT
ENO
Enum_STGP_CMD
Enum_STGP_STATE
ENUM_TO_DINT
EnumAbsBaudrate
EnumAbsMsgFormat
EnumAbsMsgLength
EnumAbsoluteRelative
EnumAbsState
EnumAcceleration
EnumAccessMode
EnumActivatedDeactivated

EnumActiveAbsRelInactive
EnumActiveInactive
EnumActiveInactiveNoChange
EnumActiveNoChange
EnumActiveWaitingForDataInactive
EnumActualDirect
EnumActualValueFormat
EnumAdditionExecution
EnumAdditionObjectErrorReaction
EnumAlarmIdState
EnumAlarmIdType
EnumApproachPositionType
EnumAxisAbortAcceleration
EnumAxisActuatorInterfaceAllocationConfig
EnumAxisAdditionalSensorType
EnumAxisApproachDirection
EnumAxisCompareMode
EnumAxisControllerOutput
EnumAxisControllerType
EnumAxisCyclicSetUpInForceLimiting
EnumAxisDataAdaption
EnumAxisDelayValueCalculationMode
EnumAxisDelayValueState
EnumAxisDioActorType
EnumAxisDriverMode
EnumAxisEnableDscSpline
EnumAxisEnableMovingMode
EnumAxisEncoderAssemblyType
EnumAxisEncoderIdentification
EnumAxisEncoderMode
EnumAxisEncoderSystem
EnumAxisEncoderType
EnumAxisEncoderValueType
EnumAxisErrorReaction
EnumAxisExtrapolatedVelocitySwitch

EnumAxisFilterMode
EnumAxisFineInterpolatorMode
EnumAxisFollowingErrorConfigMode
EnumAxisFollowingMotionState
EnumAxisForceCommand
EnumAxisForceDerivativeLimitingMode
EnumAxisForceExtrapolationType
EnumAxisForceLimitingCommandType
EnumAxisForceProfileModeConditionCommand
EnumAxisForceProfileProcessingState
EnumAxisForceState
EnumAxisForceValueConditionCommand
EnumAxisFrictionReference
EnumAxisHomingMode
EnumAxisHomingReverseCamType
EnumAxisHomingState
EnumAxisIdentification
EnumAxisInterfaceActivationConfig
EnumAxisIntervalCounterMode
EnumAxisKindOfDataAdaption
EnumAxisLimitingProfileProcessingState
EnumAxisMotionCommand
EnumAxisMotionInCommandState
EnumAxisMotionState
EnumAxisMotorType
EnumAxisOperatingMode
EnumAxisPassiveApproachDirection
EnumAxisPassiveHomingMode
EnumAxisPathMotionState
EnumAxisPathPosToleranceCommandValue
EnumAxisPosCommandSpecification
EnumAxisProfileProcessingState
EnumAxisProgrammedTargetPosition
EnumAxisPulsesEnabledEvaluation
EnumAxisReferenceCamType

EnumAxisReferenceMaxNominal
EnumAxisSensorInterfaceAllocationConfig
EnumAxisServoMonitoringPositioningState
EnumAxisSideInput
EnumAxisStateMachineControl
EnumAxisSwitchingCondition
EnumAxisSwitchingCondition_1
EnumAxisSwitchingCondition_2
EnumAxisSwLimitModeSpecificMonitoring
EnumAxisTelegramType
EnumAxisTorqueForceReductionGranularity
EnumAxisType
EnumAxisUpdateCycle
EnumAxisUserDefaultHighLow
EnumAxisValueReferenceType
EnumAxisVelocityLimitingCommandType
EnumAxisVelocityLimitingDirection
EnumAxisVelocityLimitingValueMode
EnumBackLashDiff
EnumBackLashType
EnumBalanceFilterMode
EnumBlendingMode
EnumCamBSplineInterpolation
EnumCamCSplineInterpolation
EnumCamData
EnumCamDataMode
EnumCamDefinitionState
EnumCamErrorReaction
EnumCamExtremumType
EnumCamFctNoSolFRCause
EnumCamFctNoSolLRCause
EnumCamFctWrongSpecLRCause
EnumCamFollowingRangeSpecificationMode
EnumCamInterpolationMode
EnumCamLeadingRangeStartPoint

EnumCammingActivationMode
EnumCammingDirection
EnumCammingMode
EnumCamMode
EnumCamModify
EnumCamPositionMode
EnumCamRange
EnumCamRangeSpecification
EnumCamReferenceBySlaveModeRelative
EnumCamRepresentation
EnumCamTrackActivationMode
EnumCamTrackErrorReaction
EnumCamTrackNextCommand
EnumCamTrackOutputType
EnumCamTrackPositionReference
EnumCamTrackSetState
EnumCamTrackStartMode
EnumCamTrackState
EnumCamTrackStopMode
EnumCamTrackTaskLevel
EnumCamTrackType
EnumCamTrackValue
EnumCamValueType
EnumChangeMode
EnumCommandIdState
EnumCommandValueQuantizationMode
EnumCompactControllerControllerType
EnumConfigurationInfoId
EnumContinueDynamicType
EnumContinueSpecification
EnumControllerObjectErrorBehaviorMode
EnumControllerObjectErrorReaction
EnumControllerObjectInputType
EnumControllerObjectISetMode
EnumControllerObjectPrecontrolAddupMode

EnumControllerObjectValueBehaviorMode
EnumConversionDataType
EnumDataDefault
EnumDataSetChangingState
EnumDataType
EnumDecodeSequentialMotionCommand
EnumDefaultValueDirect
EnumDeviceAbortReadWriteJobs
EnumDeviceCommandIdState
EnumDeviceConfigurationActivationState
EnumDeviceDataActivationState
EnumDeviceDataScope
EnumDeviceDpAlarmMode
EnumDeviceIdType
EnumDeviceKindOfData
EnumDeviceModeOfOperation
EnumDeviceStartupOperationMode
EnumDeviceStateOfDpSlave
EnumDeviceStorageType
EnumDeviceUnitDataSetCommand
EnumDirectEffectiveUserDefault
EnumDirection
EnumDirectionType
EnumDisableQFAxisOutputEnableMode
EnumDisableQFAxisOutputMode
EnumDoNotChangeDirect
EnumDpInterfaceSyncMode
EnumDpInterfaceSyncState
EnumDpSegmentId
EnumDpSlaveSyncState
EnumDriveFaultsType
EnumEffectiveUserDefault
EnumEnableAxisMode
EnumEnableDisable
EnumEncoderErrorReaction

EnumEncoderIdentification
EnumEndBehaviourOfProfile
EnumErrorReporting
EnumErrorReset
EnumEventClass
EnumExtendedValue
EnumExtendedValueType
EnumFanBattery
EnumFctGenStartStop
EnumFctGenState
EnumFixedGearDirectEffectiveUserDefault
EnumFixedGearDirection
EnumFixedGearDisableMode
EnumFixedGearEnableMode
EnumFixedGearErrorBehaviorMode
EnumFixedGearErrorReaction
EnumFixedGearGearingMode
EnumFixedGearGearingType
EnumFixedGearMergeModeDisableGearing
EnumFixedGearMergeModeEnableGearing
EnumFixedGearMotionOutBehaviorMode
EnumFixedGearNextCommandDisableGearing
EnumFixedGearNextCommandEnableGearing
EnumFollowingObjectDynamicMergeMode
EnumFollowingObjectDynamicReference
EnumFollowingObjectErrorReaction
EnumFollowingObjectMotionImpact
EnumFollowingObjectStateSetMasterCommand
EnumFollowingObjectSynchronizeWithLookAhead
EnumFollowingObjectSynchronizingDirection
EnumFollowingObjectSynchronizingState
EnumFollowingObjectSyncMode
EnumFollowingObjectVelocityMode
EnumFollowingState
EnumFooActMasterConnectFailMaster

EnumFooInActMasterConnectFailMaster
EnumForceControlByConditionCommandState
EnumForceController
EnumForceControllerFilterType
EnumForceControllerState
EnumForceControllerType
EnumForceControllerTypeOfSensorData
EnumForceDirection
EnumForceLimitingByConditionCommandState
EnumFormulaErrorReaction
EnumFormulaObjectErrorBehaviorMode
EnumFormulaObjectOutBehaviorMode
EnumGearingActivationMode
EnumGearingDirection
EnumGearingMode
EnumGearingPosToleranceCommandValue
EnumGearingType
EnumGetValue
EnumHomingMode
EnumInactiveNoChange
EnumIncomingOutgoing
EnumInOutDirection
EnumInSensorIdentification
EnumInSensorMode
EnumInSensorSystem
EnumInSensorType
EnumInSensorValueType
EnumInsertMode
EnumIntegratorMode
EnumInterfaceID
EnumInterfaceValueDefaultValue
EnumInterpolationState
EnumIoIdType
EnumJerk
EnumLastValidInterfaceValueDefaultValue

EnumLeadingRangeEndPoint
EnumLimitExceededOk
EnumLogicOperation
EnumMasterMode
EnumMeasuredEdge
EnumMeasuringInputAccess
EnumMeasuringInputCyclicMode
EnumMeasuringInputErrorReaction
EnumMeasuringInputInputType
EnumMeasuringInputTaskLevel
EnumMeasuringRangeMode
EnumMeasuringState
EnumMemoryCardIdType
EnumMergeMode
EnumMergeModeDisableCamming
EnumMergeModeDisableGearing
EnumMergeModeEnableCamming
EnumMergeModeEnableGearing
EnumMergeModeEnableMotionIn
EnumMergeModeForceTimeProfile
EnumMergeModeStop
EnumModeSelfAdaptingConfiguration
EnumMotionBaseType
EnumMotionBufferState
EnumMotionCommandIdState
EnumMountSwitch
EnumMoveTimeOut
EnumMovingMode
EnumMovingModeStopCommand
EnumNextCommand
EnumNextCommandCamming
EnumNextCommandCancelCommand
EnumNextCommandDataAdaption
EnumNextCommandDelayValueCalculation
EnumNextCommandDisableClamping

EnumNextCommandDisableForceLimiting
EnumNextCommandDisableLimiting
EnumNextCommandDisableTorqueLimiting
EnumNextCommandEnable
EnumNextCommandEnableAxisInterface
EnumNextCommandEnableClamping
EnumNextCommandEnableForceControl
EnumNextCommandEnableForceLimitingValue
EnumNextCommandEnableLimitingValue
EnumNextCommandEnableMeasuring
EnumNextCommandEnableMotionIn
EnumNextCommandEnableTorqueLimiting
EnumNextCommandEncoderSimulation
EnumNextCommandForceProfile
EnumNextCommandForceValue
EnumNextCommandGearing
EnumNextCommandHoming
EnumNextCommandIpoClockBorderToPosition
EnumNextCommandMode
EnumNextCommandPathMove
EnumNextCommandProfile
EnumNextCommandReset
EnumNextCommandScaling
EnumNextCommandSensor
EnumNextCommandSetMaster
EnumNextCommandSyncEncoder
EnumNextEnablePathObjectMotionTracking
EnumOffsetMode
EnumOkFaulted
EnumOnOff
EnumOperationMode
EnumOutputCamErrorReaction
EnumOutputCamInvert
EnumOutputCamOutputType
EnumOutputCamPosition

EnumOutputCamState
EnumOutputCamTaskLevel
EnumOutputCamType
EnumOutputCamValue
EnumPathBlendingMode
EnumPathCartesianKinematicsType
EnumPathCircularDirection
EnumPathCircularType
EnumPathDataSource
EnumPathDynamicAdaption
EnumPathDynamicsIn
EnumPathDynamicsInLengthMode
EnumPathErrorReaction
EnumPathIjkMode
EnumPathKinematicsConfig2D
EnumPathKinematicsType
EnumPathMergeMode
EnumPathMode
EnumPathMotionBufferState
EnumPathMotionCommand
EnumPathMotionCommandIdState
EnumPathMotionState
EnumPathObjectAttachPath
EnumPathObjectBlendingAcceleration
EnumPathObjectCsType
EnumPathObjectOcsSettingType
EnumPathObjectRedefineOcsMode
EnumPathObjectTrackingSynchronizingMode
EnumPathPlane
EnumPathPointType
EnumPathPolynomialMode
EnumPathPositionIndication
EnumPathStopMode
EnumPathSyncAxisMotionState
EnumPathTransitionState

EnumPathTransitionType
EnumPathTransitionVelocityMode
EnumPathVelocity
EnumPathVelocityProfile
EnumPathVelocityProfileProcessingState
EnumPathWDirection
EnumPathWDynamics
EnumPathWMode
EnumPersistentDataState
EnumPositioningMode
EnumPositiveNegative
EnumPosValue
EnumProfile
EnumProfileDynamicsLimiting
EnumPulsesEnabledActiveInactiveNotEvaluated
EnumQFAxisOutputEnableMode
EnumQFAxisOutputMode
EnumQFAxisOutputSetMode
EnumRange
EnumRecognitionMode
EnumRedefineMode
EnumRedefineSpecification
EnumRemoveMode
EnumReqActDeactGetStateMode
EnumReqSysFunctMode
EnumRestartAxisCondition
EnumRestartDeniedCause
EnumSaveState
EnumScaleSpecification
EnumScalingSpecification
EnumScopeOfIO
EnumSensorDataType
EnumSensorErrorReaction
EnumSensorState
EnumSensorValueBehaviorMode

EnumSetAndGetSafeValue
EnumSetNoSet
EnumSlaveMode
EnumStartStop
EnumStateOfDpSlave
EnumStateOfDpStation
EnumStateOfIO
EnumStateOfMaintenanceOfIO
EnumStateOfTo
EnumStopDriveMode
EnumStopMode
EnumStopSpecification
EnumSyncCommandState
EnumSynchronizingMode
EnumSyncModeCamming
EnumSyncModeGearing
EnumSyncOffModeCamming
EnumSyncOffModeGearing
EnumSyncOffPositionReference
EnumSyncPositionReference
EnumSyncProfileReference
EnumSystemDirect
EnumTControllerActivationModeChangedConfigData
EnumTControllerBinaryIObits
EnumTControllerCommandDestination
EnumTControllerControllerType
EnumTControllerCopyShadow
EnumTControllerDataResetMode
EnumTControllerErrorReaction
EnumTControllerErrorResetMode
EnumTControllerExecutionLevel
EnumTControllerIdentificationAvailableType
EnumTControllerIdentificationModifiedTangentMethodStage
EnumTControllerIdentificationStandardTangentMethodStage
EnumTControllerIdentificationTransitionMode

EnumTControllerIdentificationType
EnumTControllerInputGradientCheckMode
EnumTControllerInputLimitCheckMode
EnumTControllerInputLimitCheckState
EnumTControllerInputType
EnumTControllerIntegrationLimitState
EnumTControllerIntegrationMode
EnumTControllerNextCommand
EnumTControllerOperatingMode
EnumTControllerOutputLimitState
EnumTControllerOutputType
EnumTControllerPlausibilityCheckMode
EnumTControllerPlausibilityState
EnumTControllerProcessControlState
EnumTControllerProcessMode
EnumTControllerRestartActivation
EnumTControllerRestartActivationSetting
EnumTControllerRestartCondition
EnumTControllerSimulationMode
EnumTcpNextCommandMode
EnumToActivationModeSetConfigData
EnumToCommandExecution
EnumToExecutionLevel
EnumToInvalidSysvarAccess
EnumToRestartActivation
EnumToRestartActivationSetting
EnumTorqueLimitUnitType
EnumToSetStateOfTo
EnumTraceState
EnumTrackingState
EnumTransferSuperimposedPosition
EnumTrueFalse
EnumUdpCommunicationMode
EnumUpsBatteryState
EnumUpsState

F

FCTGEN_DISABLED
FCTGEN_ENABLED
FCTGEN_LIMIT_EXCEEDED
FCTGEN_PARAM_VALID
FCTGEN_RUNNING
FCTGEN_START
FCTGEN_START_FG1
FCTGEN_START_FG2
FCTGEN_STARTING
FCTGEN_STOP
FCTGEN_STOPPING
FCTGEN_WAIT_FG1
FCTGEN_WAIT_FG2
FEEDBACK
FEEDBACK_EMERGENCY_STOP
FILTER_AND_CONSTANT_LIMIT
FIND
FINISHED
FIXED_GEAR_DISABLE
fixedGearValue
FOLLOWING_OBJECT_DISABLE
FOLLOWING_RANGE
FollowingAxis
FollowingObjectType
followingRange
followingRangeSettings
FOR
FORCE_AND_INPUT
FORCE_AND_POSITION
FORCE_AND_TIME
FORCE_CONDITION
FORCE_CONTROLLED
FORCE_LIMITED
FORCE_OR_INPUT
FORCE_OR_POSITION

G

GT

H

HEAT_AND_COOL_OUTPUT_ZERO
HEAT_OUTPUT_ZERO
HEATING
HIGH
HIGH_VELOCITY
HOLD_CONNECTION
HOME_POSITION_ENTRY_MOVE
homingCommand
HW_TYPE

I

IDENTIFICATION
identificationModifiedTangentMethodParameter
identificationStandardTangentMethodParameter
IE_01
IE_02
IE_1
IE_2
IF
IMMEDIATELY
IMMEDIATELY_AND_BE_SYNCHRONOUS_AT_MASTER_POSITION
IMMEDIATELY_AND_SLAVE_POSITION
IMMEDIATELY_BY_CAM_TRACK_OUTPUT_INACTIVE
IMMEDIATELY_BY_TIME_PROFILE
IMMEDIATLY
IMPLEMENTATION
IN
IN_ACCELERATION
IN_ACTIVATION
IN_ADAPTION
IN_CALCULATION
IN_CONSTANT_MOTION

IN_DEACTIVATION
IN_DECELERATION
IN_DEFINED_TIME
IN_EXECUTION
IN_INTERPOLATION
IN_INTERPOLATION_BUT_NOT_ON_PROFILE
IN_MOTION
IN_OPERATION
INACTIVE
INACTIVE_AND_WAITING_FOR_NEXT_CYCLE
INCOMING
INCOMING_ALARM
INCREMENT_COUNTER
INCREMENT_DURATION
INITIAL_STEP
INPUT
INPUT_CONDITION
inputDisplayValueParameter
inputFilterParameter
inputGradientCheckParameter
inputLimitCheckParameter
INSERT
INT
INT_TO_BCD
INT_TO_BYTE
INT_TO_DINT
INT_TO_DWORD
INT_TO_LREAL
INT_TO_REAL
INT_TO_SINT
INT_TO_TIME
INT_TO_UDINT
INT_TO_UINT
INT_TO_USINT
INT_TO_WORD

LAST_VALUE
LD
LDN
LE
LEADING_RANGE
LEADING_RANGE_END
LEADING_RANGE_START
LEADING_RANGE_VALUE
leadingRange
leadingRangeSettings
LEFT
LEN
LENGTH_13
LENGTH_21
LENGTH_25
LESS
lifeSign
LIMIT
LIMIT_EXCEEDED
LIMITING_BY_DIRECT_VALUE
LIMITING_BY_USER_DEFAULT_VALUE
limitsOfPathDynamics
LINEAR
LINEAR_CONVERSIONDATA
LINEAR_SYSTEM
linearPathCommand
LINT
LittleByteArray_to_AnyType
LITTLEBYTEARRAY_TOANYTYPE
LN
LOAD_CONFIGURED_DATA
LOCAL_EXTREMUM
LOG
LONG_RUN_NEGATIVE
LONG_RUN_POSITIVE

LOW
LOW_VELOCITY
LOWER_PRIMARY_FAILED
LOWER_SECONDARY_FAILED
lowerPlausibilityParameter
lowerPlausibilityParameterSecondary
LREAL
LREAL_TO_DINT
LREAL_TO_INT
LREAL_TO_REAL
LREAL_TO_SINT
LREAL_TO_STRING
LREAL_TO_UDINT
LREAL_TO_UINT
LREAL_TO_USINT
LREAL_VALUE_TO_BOOL
LREAL_VALUE_TO_BYTE
LREAL_VALUE_TO_DWORD
LREAL_VALUE_TO_WORD
LREALIn1
LREALIn1Default
LREALIn2
LREALIn2Default
LREALIn3
LREALIn3Default
LREALIn4
LREALIn4Default
LREALOut1
LREALOut1Default
LREALOut2
LREALOut2Default
LREALOut3
LREALOut3Default
LREALOut4
LREALOut4Default

LT
LWORD
LWORD_TO_BCD

M

MAINTAIN_PROGRAMMED_DATA
MAINTENANCE_DEMANDED
MAINTENANCE_NO_INFO
MAINTENANCE_REQUIRED
MANDATORY
manualOutputValue
MASTER_RANGE
MASTER_SLAVE_ALARMMESSAGES_1
masterState
matchPosition
MAX
MAX_DYNAMICS
mcs
measuredValue1
measuredValue2
MEASURING_AND_MANUAL_OUTPUT
MEASURING_AND_OUTPUT_ZERO
MEASURING_ERROR
MEASURING_INPUT_DISABLE
MeasuringInputType
MEMORY_CARD_FULL
MEMORY_CARD_SERIAL_NUMBER
MID
MIN
MIN_MAX
minusLimitsOfDynamics
MOD
MODE_1
MODE_2
modeOfDpInterfaceSynchronization

modeOfOperation
MODIFICATION_ACTIVE
MODIFIED_TANGENTMETHOD
modulo
monitorings
MOTION_DONE
MOTION_EMERGENCY_ABORT
MOTION_EMERGENCY_STOP
MOTION_IN_POSITION_LOCKED_PROFILE
MOTION_STOP
motionBuffer
motionIn
MotionIn1
MotionIn1Default
MotionIn2
MotionIn2Default
MotionIn3
MotionIn3Default
MotionIn4
MotionIn4Default
MotionInDefault
MotionOut
MotionOut1
MotionOut1Default
MotionOut2
MotionOut2Default
MotionOut3
MotionOut3Default
MotionOutDefault
motionState
motionStateData
motionType
moveCommand
MOVING_WITH_REDUCED_VELOCITY
movingToEndStopCommand

MUL

MULTIME

MUX

N

NE

NEGATIVE

NEGATIVE_DIRECTION

NEXT_CAM_CYCLE

NEXT_CAM_TRACK_CYCLE

NEXT_MOTION

NEXT_WITH_REFERENCE

NO

NO_ACCESS

NO_ALARMMESSAGES

NO_CHANGE

NO_COMMAND_BUFFER_AVAILABLE

NO_CONSTRAINTS

NO_CYCLIC

NO_DP_SEGMENT

NO_HOMING_MODE

NO_POS_MOTION_ACTIVE

NO_REQUEST_TO_ACTIVATE

NO_RESSOURCES_AVAILABLE

NO_RESTART_ACTIVATION

NO_RETAIN_GLOBAL

NO_SET

NO_STORAGE_AVAILABLE

NO_TELEGRAM

NO_TYPE

NO_VALID_CONFIGURATION_ID

NO_VALID_STATE

NOCYCLIC

NODEF

NON_AVAILABLE

NONE
NOT
NOT
NOT_ACTIVATED
NOT_ADAPTED
NOT_APPLICABLE
NOT_EMPTY
NOT_ENOUGH_RAM
NOT_EVALUATED
NOT_EXISTENT
NOT_EXISTING
NOT_INTERPOLATED
NOT_LIMITED
NOT_MANDATORY
NOT_POSSIBLE
NOT_PRESENT
NOT_VALID
NOTSUPP
numberOfSummarizedTaskOverflow

O

O_K_
OBJECT
OCS
OF
OFF
OFFSET_MOVE
OFFSET_SCALE
OK
ON
ON_MASTER_AND_SLAVE_POSITION
ON_MASTER_POSITION
ON_POSITION
ON_PROFILE
ON_SLAVE_POSITION

P

PD
PERMANENT_STORAGE
persistentDataPowerMonitoring
PID
PID_ACTUAL
pidController
pidControllerDefault
pidControllerEffective
pidControllerMonitorings
PINETREE
plusLimitsOfDynamics
PN_1
PN_2
POLYNOMIAL
polynomialPathCommand
PosAxis
posCommand
POSITION
POSITION_AND_DIRECT_NIST_B
POSITION_AND_INPUT
POSITION_AND_PROFIDRIVE_ENCODER_NIST_B
POSITION_AND_PROFIDRIVE_NIST_B
POSITION_AND_TIME
POSITION_CONDITION
POSITION_CONTROLLED
POSITION_LOCKED_PROFILE
POSITION_OR_INPUT
POSITION_OR_TIME
positionBasedMotionInCommand
positioningState
positionTimeProfileCommand
POSITIVE
POSITIVE_DIRECTION
POWER
precontrol

precontrolValueDefault
precontrolValueIn
PREDEFINED_APPLICATION_EVENTS
PREDEFINED_MODULE_INFORMATION
PREVIOUS_ACTIVATED
PRIMARY
processedValue
processModeParameter
PROFIDRIVE
PROFIDRIVE_NIST_A
PROFIDRIVE_NIST_B
PROFILE_DONE
PROFILE_END
PROGRAM
PROGRAM
PT1
PT2
PV
PWM
PWMParameter
PWMParameterSecondary

R

R_EDGE
R_TRIG
RAM_DISK_FULL
RANGE_EXTREMUM
rawValue
READ_AND_WRITE_JOBS
READ_ERROR
READ_JOBS
REAL
REAL_AXIS
REAL_TO_DINT
REAL_TO_DWORD

REAL_TO_INT
REAL_TO_LREAL
REAL_TO_SINT
REAL_TO_STRING
REAL_TO_TIME
REAL_TO_UDINT
REAL_TO_UINT
REAL_TO_USINT
REAL_VALUE_TO_BOOL
REAL_VALUE_TO_BYTE
REAL_VALUE_TO_DWORD
REAL_VALUE_TO_WORD
RECTANGLE_TTL
REDUNDANT
reference
REFERENCE_CAM_AND_EXTERNAL_ZM_PASSIVE
RELATE_SYNC_PROFILE_TO_LEADING_VALUE
RELATE_SYNC_PROFILE_TO_TIME
RELATIVE
RELEASE_DISABLE
REPEAT
REPLACE
REQUEST_ABORT
REQUEST_FALSE
REQUEST_TRUE
reset
RESOLVER
RESOURCE
RESTART_BAD_CONFIG
RESTART_BY_COMMAND
RESTART_BY_SYSVAR_AND_COMMAND
RESTART_COND_MISS
RESTART_NONE
RESTART_NOT_LAST_CONFIG
RESTART_NOT_READY

RESTART_SYSVAR_FAILED
restartActivation
RESULTING
RET
RETAIN
RETC
RETCN
RETURN
REVERSE
RIGHT
RIGHT_MARGIN
RISING_EDGE
RISING_EDGES_ONLY
ROL
ROR
ROTATORY
ROTATORY_SYSTEM
RS
RTC
RUN

S

SAFETY
SAME_DIRECTION
SAVE_ABORTED
SAVE_FINISHED
SEARCH_ENDPOINT
SEARCH_INFLECTIONPOINT
SEARCH_STARTPOINT
SECONDARY
SEL
SEMA
SENSOR_ABSOLUTE
SENSOR_ANALOG
SENSOR_CYCLIC_ABSOLUTE

SENSOR_INCREMENTAL
SENSOR_POSITION_DIFFERENCE_MEASUREMENT
sensorData
sensorMonitoring
sensorSettings
SEQUENTIAL
SERIAL_NUMBER
SERVO
SERVO_FAST
servoControlClock
servoControlClock_fast
servodata
servoMonitoring
servoSettings
servoTaskCycle
servoTaskCycle_fast
SET
SET_ACTUAL_VALUE
SET_DP_ALARM
SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION
SET_VALUE
setForceCommandValueCommand
setMasterCommand
setpoint
SETPOINT_VALUE
setpointDefault
setpointIn
SETTING_OF_COEFFICIENTS
SETTING_UP
SHADOW
SHADOW_TO_DIRECT
SHL
SHORTEST_WAY
SHR
SIDE_INPUT_NEG

SIDE_INPUT_NO_SIDE
SIDE_INPUT_POS
SIMULATION
SIMULATION_ABORT
SIMULATION_STOP
SIN
SINGLE
SINGLE_PATH
singleCamState
SINT
SINT_TO_BCD
SINT_TO_BYTE
SINT_TO_DINT
SINT_TO_DWORD
SINT_TO_INT
SINT_TO_LREAL
SINT_TO_REAL
SINT_TO_UDINT
SINT_TO_UINT
SINT_TO_USINT
SINT_TO_WORD
SINT_VALUE_TO_BOOL
SINUS_1VPP
SINUSOIDAL
SLAVE_ALARMMESSAGES_1
SLAVE_RANGE
SLOW
SMOOTH
SPECIFIC
SPECIFIC_AXIS_MOTION
SPECIFIC_ERROR
SPECIFIC_ID
SPECIFIC_NUMBER
SPECIFIC_PART_OF_RANGE
SPECIFIC_POINT

SPECIFIC_START_DATA
specificVelocityProfile
SPEED_CONTROLLED
speedMode
SQRT
SR
SSI_MODE
ST
STANDARD
STANDARD_TANGENTMETHOD
STANDARD_TYPE
STANDSTILL
STANDSTILL_MONITORING_ACTIVE
START
START_POINT
STARTPOINT_ENDPOINT
state
STATE_HOMED
STATE_HOMING
STATE_INITIAL
STATE_MACHINE_CONTROL_BY_APPLICATION
STATE_MOVING
STATE_POSABS
STATE_POSITION_END
STATE_POSREL
STATE_STOPPED
STATE_TOACTIVE
stateCammingAdjustments
stateGearingAdjustments
stateOfDpInterfaceSynchronization
stateOfDpSlaveSynchronization
stateSetMasterCommand
STEP
STEPMOTOR
STN

STOP
STOP_ALL_FORMULA
STOP_AND_ABORT
STOP_AND_ABORT_AND_HOLD
STOP_DEVICE
STOP_IN_DEFINED_TIME
STOP_SPECIFIC_FORMULA
STOP_SYMMETRIC_WITH_POSITION
STOP_WITH_COMMAND_VALUE_ZERO
STOP_WITH_DYNAMIC_PARAMETER
STOP_WITH_MAXIMAL_DECELERATION
STOP_WITHOUT_ABORT
stopEmergencyCommand
STOPPED
STOPPED_WITHOUT_ABORT
STOPU
STRING
STRING_TO_DINT
STRING_TO_LREAL
STRING_TO_REAL
STRING_TO_UDINT
STRUCT
StructAdditionalSensorNumber
StructAdditionObjectMotionIn
StructAdditionObjectMotionOut
StructAlarmId
StructAlarmId_TO_DINT
StructAxisAbsoluteEncoder
StructAxisActorData
StructAxisActorMonitoring
StructAxisActualTorque
StructAxisAdditionalSensorData
StructAxisAdditiveTorque
StructAxisAdditiveTorqueIn
StructAxisDataSetReadWrite

StructAxisDataSetReadWrite_V3_1
StructAxisDataSetReadWrite_V4_1
StructAxisDefaultType
StructAxisDistributedMotion
StructAxisDriveData
StructAxisDriveSafetyExtendedFunctionsInfoData
StructAxisDynamicLimit
StructAxisDynamicQFData
StructAxisExtrapolationData
StructAxisForceActualValueSettings
StructAxisForceControlByConditionCommand
StructAxisForceControlDataSet
StructAxisForceControlDataSet_V3_1
StructAxisForceControlDataSet_V4_1
StructAxisForceControllerData
StructAxisForceControllerMonitorings
StructAxisForceControllerSettings
StructAxisForceLimitingByConditionCommand
StructAxisForceLimitingCommand
StructAxisForceMotionInPositionProfileCommand
StructAxisForcePositionProfileCommand
StructAxisForceStateData
StructAxisForceTimeProfileCommand
StructAxisHomingCommand
StructAxisHomingDefault
StructAxisInvertQOutput
StructAxisInvertSetPointHydraulicType
StructAxisModuloData
StructAxisMotionData
StructAxisMotionInData
StructAxisMotionStateData
StructAxisMoveCommand
StructAxisMovingToEndStopCommand
StructAxisOperatingData
StructAxisOverride

StructAxisPathMotion

StructAxisPathSyncMotion

StructAxisPosCommand

StructAxisPositionBasedMotionInCommand

StructAxisPositioningDefault

StructAxisPositioningState

StructAxisPositionTimeProfileCommand

StructAxisSensorData

StructAxisSensorMonitoring

StructAxisSensorSettings

StructAxisServoData

StructAxisServoMonitoring

StructAxisServoSettings

StructAxisSetForceCommandValueCommand

StructAxisSwLimit

StructAxisSwLimitState

StructAxisTorqueLimitIn

StructAxisTorqueLimitingCommand

StructAxisUserDefaultClamping

StructAxisUserDefaultForceLimiting

StructAxisUserDefaultTorqueLimit

StructAxisVelocityBasedMotionInCommand

StructAxisVelocityLimitingCommand

StructAxisVelocityMotionInPositionProfileCommand

StructAxisVelocityPositionProfileCommand

StructAxisVelocityTimeProfileCommand

StructCamFollowingRange

StructCamInterpolation

StructCammingAdjustments

StructCammingSettings

StructCamRange

StructCamScaleRange

StructCamSettings

StructCamTrackArrayOfSingleCamSettings

StructCamTrackEffectiveData

StructCamTrackSingleCamSettings
StructCamTrackUserDefault
StructCamUserDefault
StructClampingMonitoring
StructControllerDynamic
StructControllerObjectControlDeviation
StructControllerObjectOutputData
StructControllerObjectOutputMonitoring
StructControllerObjectPControllerData
StructControllerObjectPControllerDefault
StructControllerObjectPIDControllerData
StructControllerObjectPIDControllerDefault
StructControllerObjectPIDControllerEffective
StructControllerObjectPIDControllerMonitorings
StructControllerObjectPrecontrolData
StructControllerObjectValueIn
StructControllerType
StructControllerType_V3_1
StructCpuDriveStates
StructCyclicMeasuringEnableCommand
StructDataSetMonitoring
StructDeviceConfigurationData
StructDeviceConfigurationManagement
StructDeviceCpuData
StructDeviceCpuDataRW
StructDeviceDataActivationState
StructDeviceFanBattery
StructDeviceIm0
StructDeviceIm0Softwareversion
StructDeviceIm0Version
StructDeviceImData
StructDeviceStartUp
StructDeviceUpsData
StructDpStationAddressType
StructDynamicComp

StructDynamicComp_V4_1
StructDynamicData
StructDynamicFollowing
StructDynamicFollowing_V4_1
StructEffectiveTaskRuntimeType
StructEncoderEffective
StructEncoderMotionState
StructEncoderNumber
StructEncoderSyncCommand
StructEncoderUserDefault
StructFilterForceControl
StructFixedGearingAdjustments
StructFixedGearingOffset
StructFixedGearingSettings
StructFixedGearMotionIn
StructFixedGearMotionOut
StructFixedGearUserDefault
StructFollowingObjectCurrentSyncPosition
StructFollowingObjectDistributedMotion
StructFollowingObjectMasterDynamics
StructFollowingObjectUserDefault
StructForceControllerData2
StructForceControllerData2_V3_1
StructForceControllerData2_V4_1
StructForceControllerDifference
StructForceControlSwitchingData
StructForceControlUserDefault
StructForceLimitingSwitchingData
StructFormulaObjectDINTIn
StructFormulaObjectDINTOut
StructFormulaObjectLREALIn
StructFormulaObjectLREALOut
StructFormulaObjectMotionIn
StructFormulaObjectMotionOut
StructGear

StructGearingAdjustments
StructGearingOffset
StructGearingSettings
StructInternalServoSettings
StructInternalToTrace
StructLimitsOfPathDynamics
StructMeasuringInputEffective
StructMeasuringInputUserDefault
StructMotionVector
StructOutputCamCounterData
StructOutputCamEffectiveData
StructOutputCamUserDefault
StructOutputControllerOutput
StructOutputData
StructOutputLimits
StructOutputMonitoring
StructOutputSettings
StructPathAbortPosition
StructPathAxisMotionVector
StructPathBcsData
StructPathBlending
StructPathCircularCommand
StructPathData
StructPathDynamics
StructPathDynamicsInState
StructPathKinematicsData
StructPathLinearCommand
StructPathMcsData
StructPathMotionActual
StructPathMotionVector
StructPathObjectConnection
StructPathObjectCsFrame
StructPathObjectMotionBuffer
StructPathObjectOcs
StructPathObjectUserDefaultOcs

StructPathOverride
StructPathPolynomialCommand
StructPathSettings
StructPathSpecificVelocityProfile
StructPathUserDefault
StructPathVector
StructPathWSettings
StructPDController
StructPendingAlarmState
structPersistentDataPowerMonitoringtype
StructPID_Controller
StructPID_Controller_V3_1
StructPID_Controller_V4_1
StructPIDController
StructPIDController_V3_1
StructPNSyncCounter
StructProcessModel
StructPVController
StructPVController_V3_1
StructQFAxisDataSet
StructQFAxisDataSet_V3_1
StructQFAxisMaxDerivative
StructQFAxisUserDefault
StructRetAllocateToken
StructRetCommandState
StructRetConfiguration
StructRetDeviceCommandState
StructRetDeviceGetStateOfAllDpStations
StructRetDeviceGetStateOfDpSlave
StructRetDeviceGetStateOfIO
StructRetDeviceNameOfStation
StructRetDiagnosticData
StructRetDpSlaveAddress
StructRetEncoderValue
StructRetGetAxisProgrammedTargetPosition

StructRetGetAxisSpecificState
StructRetGetAxisSpecificState2
StructRetGetAxisStoppingData
StructRetGetCamFollowingDerivative
StructRetGetCamSpecificState
StructRetGetCamSpecificState2
StructRetGetCamTrackSpecificState
StructRetGetCircularPathData
StructRetGetCircularPathData_V4_2
StructRetGetCircularPathGeometricData
StructRetGetConfigurationData
StructRetGetDataByToken
StructRetGetDeviceId
StructRetGetDoIndexNumberFromLogAddress
StructRetGetDpStationAddressFromLogDiagnosticAddress
StructRetGetErrorNumberState
StructRetGetExternalEncoderSpecificState
StructRetGetExternalEncoderSpecificState2
StructRetGetFollowingMinMax
StructRetGetForceControlDataSetParameter
StructRetGetForceControlDataSetParameter_V3_1
StructRetGetForceControlDataSetParameter_V4_1
StructRetGetGearAxisSpecificState
StructRetGetGearAxisSpecificState2
StructRetGetGeoAddressFromLogAddress
StructRetGetInOutByte
StructRetGetLinearPathData
StructRetGetLinearPathData_V4_2
StructRetGetLinearPathGeometricData
StructRetGetLogDiagnosticAddressFromStationAddress
StructRetGetMeasuringInputSpecificState
StructRetGetMeasuringInputSpecificState2
StructRetGetMemoryCardId
StructRetGetNextLogAddress
StructRetGetOutputCamSpecificState

StructRetGetOutputCamSpecificState2
StructRetGetPathGeometricData
StructRetGetPathMotionBuffer
StructRetGetPathObjectBcsFromOcsData
StructRetGetPathObjectOcsFromBcsData
StructRetGetPendingAlarms
StructRetGetPendingAlarmState
StructRetGetPolynomialPathData
StructRetGetPolynomialPathData_V4_2
StructRetGetPolynomialPathGeometricData
StructRetGetPosAxisSpecificState
StructRetGetPosAxisSpecificState2
StructRetGetQFAxisDataSetParameter
StructRetGetQFAxisDataSetParameter_V3_1
StructRetGetSegmentIdentification
StructRetGetStateOfAllDpSlaves
StructRetGetStateOfSingleDpSlave
StructRetGetStateOfTo
StructRetGetStationType
StructRetGetToError
StructRetGetValue
StructRetIPConfig
StructRetMotionBuffer
StructRetMotionCommandState
StructRetPathAxesData
StructRetPathAxesPosition
StructRetPathCartesianData
StructRetPathCartesianPosition
StructRetPathMotionCommandState
StructRetReadDriveFaults
StructRetReadDriveMultiParameter
StructRetReadDriveMultiParameterDescription
StructRetReadDriveParameter
StructRetReadDriveParameterDescription
StructRetReadGetAxisDataSet

StructRetReadGetAxisDataSet_V3_1
StructRetReadGetAxisDataSet_V4_1
StructRetReadRecord
StructRetTcpOpenClient
StructRetTcpOpenServer
StructRetTcpReceive
StructRetUdpReceive
StructRetUnitDataSetCommand
StructRetWriteDriveMultiParameter
StructRetWriteDriveParameter
StructRetXCommandState
StructRetXreceive
StructScaleOffset
StructSensorMonitorings
StructStateOfDpStations
StructStateOfIO
StructSyncDynamics
StructSyncMonitoring
StructSyncPositions
StructSyncProfileDefinitions
StructSystemLoad
StructTaskId
StructTaskOverflowType
StructTaskRuntimeType
StructTaskRuntimeValues
StructTCOFctGenOverride
StructTControllerActualDPIDData
StructTControllerActualIdentificationModifiedTangentMethodData
StructTControllerActualIdentificationStandardTangentMethodData
StructTControllerActualInputData
StructTControllerActualInputLimitCheckData
StructTControllerActualInputSingleLimitCheckData
StructTControllerControlRangeParameter
StructTControllerCycleParameter
StructTControllerDPIDParameter

StructTControllerIdentificationModifiedTangentMethodParameter
StructTControllerIdentificationModifiedTangentMethodProcessParameter
StructTControllerIdentificationStandardTangentMethodParameter
StructTControllerIdentificationStandardTangentMethodProcessParameter
StructTControllerIdentificationStaticCondition
StructTControllerInputDisplayValueParameter
StructTControllerInputFilterParameter
StructTControllerInputGradientCheckParameter
StructTControllerInputLimitCheckParameter
StructTControllerInputSingleLimitCheckParameter
StructTControllerOutputValue
StructTControllerPlausibilityParameter
StructTControllerProcessModeParameter
StructTControllerPWMParameter
StructTorqueLimit
StructTraceControl
StructTraceState
StructUserData
StructValueAndDerivedMasterValue
StructValueAndDerivedSlaveValue
StructVelocityGearingSettings
StructXsendDestAddr
SUB
SUB_DATE_DATE
SUB_DT_DT
SUB_DT_TIME
SUB_TIME
SUB_TOD_TIME
SUB_TOD_TOD
SUPERIMPOSED_MOTION
SUPERIMPOSED_MOTION_ACTIVE
SUPERIMPOSED_MOTION_MERGE
SUPERIMPOSED_POS_MOTION_ACTIVE
superimposedMotion
SWITCH_BY_VALUE

T

TCOFctGenOverride
TemperatureControllerType
TEMPORARY_STORAGE
THEN
TIME
TIME_AND_INPUT
TIME_CONDITION
TIME_LOCKED_PROFILE
TIME_OF_DAY
TIME_OR_INPUT
TIME_OUT
TIME_TO_INT
TIME_TO_REAL
TIME_TO_UDINT
TO
TO_CONNECTION
TO_EXECUTION
TOD
TOF
TON
TORQUE
torqueLimitingCommand
torqueLimitNegative
torqueLimitNegativeIn
torqueLimitPositive
torqueLimitPositiveIn
TOTAL_MOVE
tOutput
TP
TRACE_ABORTED
TRACE_FINISHED
TRACE_INACTIVE
TRACE_MISMATCH
TRACE_NO_TIME
TRACE_PARAM_VALID

U

UINT_TO_INT
UINT_TO_LREAL
UINT_TO_REAL
UINT_TO_SINT
UINT_TO_UDINT
UINT_TO_USINT
UINT_TO_WORD
UINT_VALUE_TO_BOOL
ULINT
UNDER
UNI_DIRECTION
UNIT
UNIT_NOT_FOUND
UNTIL
UPPER_PRIMARY_FAILED
UPPER_SECONDARY_FAILED
upperPlausibilityParameter
upperPlausibilityParameterSecondary
USELIB
USEPACKAGE
USER
USER_DEFAULT
USER_DEFINED
USER_EVENTS_1
USER_EVENTS_2
USER_STORAGE
userData
userDefault
userDefaultClamping
userDefaultDynamics
userDefaultForceControl
userDefaultForceLimiting
userDefaultHoming
userDefaultOcs
userDefaultPositioning

V

W

WHEN_FORCE_CONTROL_ENABLED
WHEN_FORCE_LIMIT_REACHED
WHEN_FORCE_LIMITING_ACTIVATED
WHEN_FUNCTION_DISABLED
WHEN_GEARING_START
WHEN_INTERPOLATION_DONE
WHEN_LIMIT_REACHED
WHEN_LIMITING_COMMAND_ACTIVATED
WHEN_MOTION_DONE
WHEN_PROFILE_DONE
WHEN_TORQUELIMIT_GONE
WHEN_TORQUELIMIT_REACHED
WHEN_TRIGGER_OCCURED
WHILE
WITH
WITH_CAM_TRACK_ACTIVATION
WITH_COMMAND_VALUE_ZERO
WITH_DYNAMICS
WITH_INTERPOLATION
WITH_MAXIMAL_DECELERATION
WITH_NEXT_SYNCHRONIZING
WITH_RADIUS_AND_ENDPOSITION
WITH_SPECIFIC_AREA
WITH_TIME_LIMIT
WITHOUT_CHANGE
WITHOUT_INTERPOLATION
WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM
WITHOUT_LIMITING
WITHOUT_SPECIFIC_AREA
WITHOUT_TIME_LIMIT
WORD
WORD_TO_BCD
WORD_TO_BOOL
WORD_TO_BYTE
WORD_TO_DINT

WORD_TO_DWORD
WORD_TO_INT
WORD_TO_SINT
WORD_TO_UDINT
WORD_TO_UINT
WORD_TO_USINT
WORD_VALUE_TO_LREAL
WORD_VALUE_TO_REAL
WRITE_JOBS
WRITEABLE

X

X_Y
X_Y_Z
XOR
XOR
XORN

Y

Y_Z
YES

Z

Z_X
ZERO_VALUE
ZM_PASSIVE

4.2.14.3 Assignment types for symbolic assignment

Interface types of the technology objects

Description

SIMOTION technology objects and SINAMICS objects provide interfaces and assignment types for symbolic assignment; these are described in detail below.

Only interfaces of the same type can be assigned.

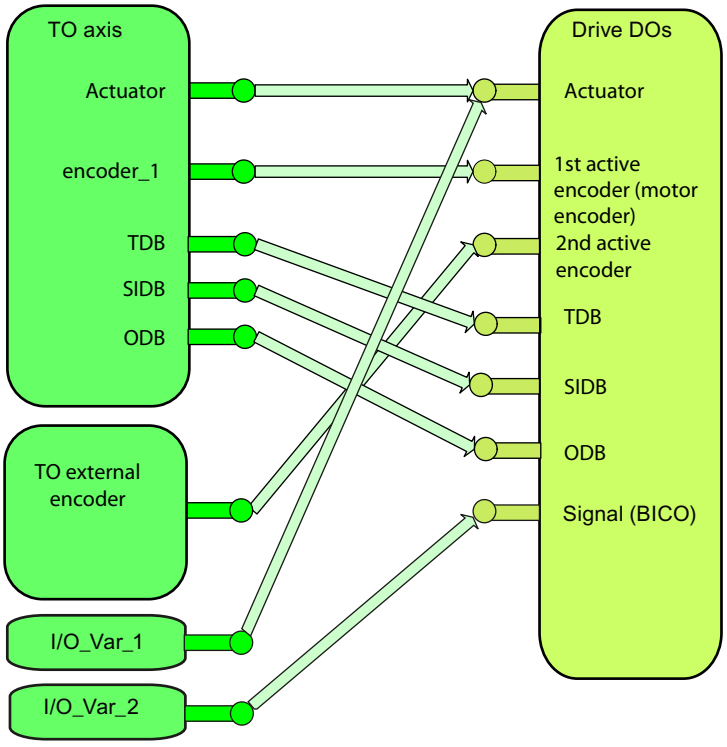


Figure 4-181 Example TO - DO assignment

Assignment types for the Axis TO

Assignment partner designation	Assignment type	Description
Actuator	• ActorProfidrive • ActorOnboard • WordOutput	Axis setpoint interface • Via PROFIdrive message frame • Onboard • Analog output for hydraulics
Technology data	TDB	Technology data on the axis
Safety data	SIDB	Safety data on the axis
encoder_<n>	BitInput	Encoder interface of the nth encoder
StatusExternalZeroMark	BitInput	External zero mark status of a digital drive. For incremental encoders
Homing output cam	BitInput	Homing output cam
HomingOutputCamPassive-Homing	BitInput	Homing output cam for passive homing
Ready status detection	BitInput	Ready status detection for analog sensor
Error status detection	BitInput	Error status detection for analog sensor
Update counter	BitInput	Update counter, analog sensor
Positive reversing cam	BitInput	Positive reversing cam
Negative reversing cam	BitInput	Negative reversing cam

Assignment partner designation	Assignment type	Description
HW limit switch positive	BitInput	Hardware limit switch positive
HW limit switch negative	BitInput	Hardware limit switch negative
Q-valve	WordOutput	Q output for hydraulics, Q-valve
Q-valve enable	BitOutput	Enable bit for Q-valve
P-valve	WordOutput	F output for hydraulics, P-valve
P-valve enable	BitOutput	Enable bit for P-valve
Pressure sensor	Wordinput	Pressure sensor
Digital input	BitInput	Digital input for fast switching to pressure control

Assignment types for the External Encoder TO

Assignment partner designation	Assignment type	Description
encoder_1	• sensorPROFIdrive • sensorOnboard • wordInput • dwordInput	Encoder interface of the encoder
StatusExternalZeroMark	bitInput	External zero mark status of a digital drive. For incremental encoders
Homing output cam	bitInput	Homing output cam
HomingOutputCamPassive-Homing	bitInput	Homing output cam for passive homing
Ready status detection	bitInput	Ready status detection for analog sensor
Error status detection	bitInput	Ready status detection for analog sensor
Update counter	bitInput	Update counter, analog sensor
Measuring input		Measuring input input for local measuring

Assignment types for the Output Cam TO

Assignment partner designation	Assignment type	Description
Output	• bitOutput • bitOutputWithTi-meStamp	Output cam output

Assignment types for the Measuring Input TO

Assignment partner designation	Assignment type	Description
Input	bitInputWithTimeStamp	Measuring input input

Assignment types for the TControl TO

The following interface types are available on the TO sensor:

Assignment partner designation	Assignment type	Description
Temperature input	wordInput	Analog input, temperature
PWM signal	bitInput	Output, PWM signal
PWM signal_2	bitInput	Output, PWM signal_2

Assignment types for the Sensor TO

The following interface type is available on the TO sensor:

Assignment partner designation	Assignment type	Description
Analog sensor	wordInput	Analog input

SINAMICS interface types

Description

The various SINAMICS DOs offer the following I/O interfaces to SIMOTION interfaces.

DO servo, vector, and TM41 (drive DOs)

Name	Interface type	Description
Actor	actorProfidrive	Axis setpoint interface (via PROFIdrive message frame)
Encoder_1	sensorPROFIdrive	Encoder 1 on the drive
Encoder_2	sensorPROFIdrive	Encoder 2 on the drive
Bico_iw.<BICO> Bico_id.<BICO>	wordInputDriveParameter	Additional parameter transferred via BICO interconnection (input side)
Bico_qw.<BICO> Bico_qd.<BICO>	dwordInputDriveParameter	Additional parameter transferred via BICO interconnection (output side)
For information on interfaces, see: TM41 inputs/outputs in the SCOUT online help.		

Encoder DO

Name	Interface type	Description
encoder	sensorPROFIdrive	PROFIdrive encoder interface
Bico_iw.<BICO> Bico_id.<BICO>	wordInputDriveParameter	Additional parameter transferred via BICO interconnection (input side)

Name	Interface type	Description
Bico_qw.<BICO> Bico_qd.<BICO>	dwordInputDriveParameter	Additional parameter transferred via BICO interconnection (output side)
For information on interfaces, see: Encoder DO in the SCOUT online help.		

DO TM15/TM17 High Feature

Name	Interface type	Description
DI_<n> MI_<n> DO_<n> CAM_<n>	bitInput bitInputWithTimeStamp bitOutput bitOutputWithTimeStamp	Interfaces for DIs, DOs, measuring inputs, output cams for terminals X520, X521, X522; interface type depends on terminal configuration TM15: 24 channels, TM17: up to 16 channels
DI_0_15 DI_16_23 DO_0_15 DO_16_23	_wordInputInterfaceType wordOutputInterfaceType	Group interfaces for replacement values (Wordinput, Wordoutput)
For information on interfaces, see: TM17 inputs/outputs TM15 inputs/outputs in the SCOUT online help.		

DO control unit

Name	Interface type	Description
DI_<n>	bitInput	Digital inputs of the control unit with SIMOTION D, CX32/CX32-2, and SINAMICS S110/S120 control units
DI_<n> MI_<n> DO_<n> CAM_<n>	bitInput bitInputWithTimeStamp bitOutput bitOutputWithTimeStamp	Bidirectional digital inputs/outputs DI interfaces (bidirectional digital input) Measuring input input DO interfaces (bidirectional digital output) Output of output cam for the terminals
DI_0_15 DO_0_15	_wordInputInterfaceType _wordOutputInterfaceType	Group interfaces for replacement values (Wordinput, Wordoutput)
Bico_iw.<BICO> Bico_id.<BICO>	wordInputDriveParameter	Additional parameter transferred via BICO interconnection (input side).
Bico_qw.<BICO> Bico_qd.<BICO>	wordOutputDriveParameter	Additional parameter transferred via BICO interconnection (output side)
For information on interfaces, see: Control Unit CU320 inputs/outputs in the SCOUT online help.		

Additional SINAMICS DO (such as TM31, TB30, and TM15 DI/DO)

Name	Interface type	Description
Interfaces depending on the standard message frame		
Bico_iw.<BICO>	WordInputDriveParameter	Additional parameter transferred via BICO interconnection (input side). The number of interconnectable parameters depends on the message frame used.
Bico_qw.<BICO>	WordOutputDriveParameter	Additional parameter transferred via BICO interconnection (output side). The number of interconnectable parameters depends on the message frame used.

Interfaces in standard message frames

Description

Standard message frames are standardized and transfer a set number of interfaces with certain properties.

They contain the following interfaces:

Standard message frame 1

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_A	WORD	Actor interface, speed setpoint A, 16 bits
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_A	WORD	Actor interface, actual speed value A, 16 bits

Standard message frame 2

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2

Standard message frame 3

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2

Standard message frame 4

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_2.Gn_STW	WORD	Encoder 2, control word
encoder_2.Gn_ZSW	WORD	Encoder 2, status word
encoder_2.Gn_XIST1	DWORD	Encoder 2, act. position val. 1
encoder_2.Gn_XIST2	DWORD	Encoder 2, act. position val. 2

Standard message frame 5

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word

Name	Data type	Description
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
actor.XERR	DWORD	Actor interface, position deviation
actor.KPC	DWORD	Actor interface, position controller gain factor

Standard message frame 6

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_2.Gn_STW	WORD	Encoder 2, control word
encoder_2.Gn_ZSW	WORD	Encoder 2, status word
encoder_2.Gn_XIST1	DWORD	Encoder 2, act. position val. 1
encoder_2.Gn_XIST2	DWORD	Encoder 2, act. position val. 2

SIEMENS message frame 101

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word

SIEMENS message frame 102

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits

Name	Data type	Description
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2

SIEMENS message frame 103

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_2.Gn_STW	WORD	Encoder 2, control word
encoder_2.Gn_ZSW	WORD	Encoder 2, status word
encoder_2.Gn_XIST1	DWORD	Encoder 2, act. position val. 1
encoder_2.Gn_XIST2	DWORD	Encoder 2, act. position val. 2

SIEMENS message frame 105

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word

Name	Data type	Description
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
actor.XERR	DWORD	Actor interface, position deviation
actor.KPC	DWORD	Actor interface, position controller gain factor

SIEMENS message frame 106

Name	Data type	Description
actor.STW1	WORD	Actor interface, control word 1
actor.NSOLL_B	DWORD	Actor interface, speed setpoint B, 32 bits
actor.STW2	WORD	Actor interface, control word 2
actor.ZSW1	WORD	Actor interface, status word 1
actor.NIST_B	DWORD	Actor interface, actual speed value B, 32 bits
actor.ZSW2	WORD	Actor interface, status word 2
actor.MOMRED	WORD	Actor interface, torque reduction
actor.MELDW	WORD	Actor interface, message word
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_2.Gn_STW	WORD	Encoder 2, control word
encoder_2.Gn_ZSW	WORD	Encoder 2, status word
encoder_2.Gn_XIST1	DWORD	Encoder 2, act. position val. 1
encoder_2.Gn_XIST2	DWORD	Encoder 2, act. position val. 2
actor.XERR	DWORD	Actor interface, position deviation
actor.KPC	DWORD	Actor interface, position controller gain factor

Standard message frame 81

Name	Data type	Description
encoder_1.STW2	WORD	Control word 2
encoder_1.ZSW2	WORD	Status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2

Standard message frame 82

Name	Data type	Description
encoder_1.STW2	WORD	Control word 2
encoder_1.ZSW2	WORD	Status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_1.NIST_A	WORD	Encoder 1, actual speed value A, 16 bits

Standard message frame 83

Name	Data type	Description
encoder_1.STW2	WORD	Control word 2
encoder_1.ZSW2	WORD	Status word 2
encoder_1.Gn_STW	WORD	Encoder 1, control word
encoder_1.Gn_ZSW	WORD	Encoder 1, status word
encoder_1.Gn_XIST1	DWORD	Encoder 1, act. position val. 1
encoder_1.Gn_XIST2	DWORD	Encoder 1, act. position val. 2
encoder_1.NIST_B	DWORD	Encoder 1, actual speed value B, 32 bits

Structure of message frame extension TDB

Name	Data type	Description
M_ADD	WORD	Additional torque
M_LIMIT_PLUS	WORD	Positive torque limit
M_LIMIT_MINUS	WORD	Negative torque limit
M_ACTUAL	WORD	Current torque

Structure of message frame extension SIDB

Name	Data type	Description
S_ZSW1B	WORD	Safety status word
S_SPEED_LIMIT	DWORD	Velocity limit signal

Designations for interface types

Description

The following assignment types have been defined for SIMOTION:

Axis and encoder interface types

Table 4-149 Axis interface types

Name under ST nomenclature	Name in the address list
_actorProfidriveInterfaceType	ActorProfidrive
_actorOnboardInterfaceType	ActorOnboard
_wordOutputInterfaceType	WordOutput

Table 4-150 Encoder interface types

Name under ST nomenclature	Name in the address list
_sensorProfidriveInterfaceType	SensorProfidrive
_sensorOnboardInterfaceType	SensorOnboard
_wordInputInterfaceType	WordInput
_dwordInputInterfaceType	DWordInput

Table 4-151 Data blocks (extended communication)

Name under ST nomenclature	Name in the address list
_tdbInterfaceType	TDB
_sidbInterfaceType	SIDB

Output cam, measuring input

Name in ST naming scheme of the output cam assignment types	Name in the address list
_bitOutputInterfaceType	BitOutput
_bitOutputWithTimeStampInterfaceType	BitOutputWithTimeStamp

Name in ST naming scheme of the measuring input assignment types	Name in the address list
_bitInputWithTimeStampInterfaceType	BitInputWithTimeStamp

Standard I/O interfaces

Name under ST nomenclature	Name in the address list
_bitInputInterfaceType	BitInput
_bitOutputInterfaceType	BitOutput
_byteInputInterfaceType	ByteInput
_byteOutputInterfaceType	ByteOutput
_wordInputInterfaceType	WordInput
_wordOutputInterfaceType	WordOutput

Name under ST nomenclature	Name in the address list
_dwordInputInterfaceType	DWordInput
_dwordOutputInterfaceType	DWordOutput

SINAMICS parameter interface types

Name under ST nomenclature	Name in the address list
_wordInputDriveParameterInterfaceType	WordInputDriveParameter
_dwordInputDriveParameterInterfaceType	DWordInputDriveParameter
_wordOutputDriveParameterInterfaceType	WordOutputDriveParameter
_dwordOutputDriveParameterInterfaceType	DWordOutputDriveParameter

Syntax of the assignments

You can also enter the symbolic assignments for the I/O variable using text fields in the address list and TO configuration dialogs (in addition to the buttons for calling the assignment dialog). The syntax required for this purpose is specified in the table below.

Table 4-152

Interconnection target	Syntax
Onboard I/Os of SIMOTION D4x5/ D4x5-2, terminal X122/X132 (SINAMICS Integrated);	<device>.<DO_name>.<interface_name> [<terminal_name>] **Example of D4x5-2:** SINAMICS_Integrated.Control_Unit.DO_8 [DI/DO 8, X122.9] -> Digital output "DO_8" on "SINAMICS_Integrated" device, "Control_Unit" drive object, terminal "X122.9", pin designation "DI/DO 8"
Onboard I/Os of SIMOTION D4x5-2, terminal X142	<device>.<<interface_name> **Example:** D455-2, X142: D455.CAM_6 [DI/DO 6, X142.12] -> Output with output cam quality "CAM_6" on "D455" device, terminal "X142.12", pin designation "DI/DO 6"
Onboard I/Os of SINAMICS S120 CU3xx and CX32/32-2 controller extension	<device>.<DO_name>.<interface_name> [<terminal_name>] **Example:** SIMOTION_CX32.Control_Unit.DI_1 [DI 1, X122.2] S120_CU320.Control_Unit.DI_1 [DI 1, X122.2] -> Digital input "DI_1" on "SIMOTION_CX32" or "S120_CU320" device, "Control_Unit" drive object, terminal "X122.2", pin designation "DI 1"

Interconnection target	Syntax
TM15, TM17, TM41	<device>.<DO_name>.<interface_name> [<terminal_name>] **Examples:** TM15: SINAMICS_Integrated.TM15.MI_8 [DI/DO 8, X521.2] -> Input with measuring input quality "MI_8" on "SINAMICS_Integrated" device, "TM15" drive object, terminal "X521.2", pin designation "DI/DO 8" TM41: SINAMICS_Integrated.TM41.AI_0 [AI 0, X523] -> Analog input "AI_0" on "SINAMICS_Integrated" device, "TM41" drive object, terminal "X523", pin designation "AI 0"
TB30, TM31, TM15 DI/DO	<device>.<DO_name>.<interface_name> [<terminal_name>] **Examples:** TB30: SINAMICS_Integrated.TB30.AI_0 [AI 0, X482.1, X482.2] -> Analog input "AI_0" on "SINAMICS_Integrated" device, "TB30" drive object, terminals "X482.1" and "X482.2", pin designation "AI 0" TM31: SINAMICS_Integrated.TM31.DI_2 [DI 2, X520.3] -> Digital input "DI_2" on "SINAMICS_Integrated" device, "TM31" drive object, terminal "X520.3", pin designation "DI 2" TM15 DI/DO: SINAMICS_Integrated.TM15DIDO.DO_23 [DI/DO 23, X522.9] -> Digital output "DO_23" on "SINAMICS_Integrated" device, "TM15DIDO" drive object, terminal "X522.9", pin designation "DI/DO 23"
SINAMICS free message frame extensions	<device>.<DO_name>.BICO_xx.<parameter_number> with (xx = IW, QW, ID, QD) for interconnection to a parameter in the same drive object (DO), to which the message frame is addressed <device>.<DO_name>. BICO_xx.<DO_name2>#<parameter_number> for interconnection to a parameter in drive object (DO) <DO_name2>, whereby the message frame is configured for DO <DO_name>
Message frame components	<device>.<DO_name>.<interface_name>.<subinterface_name> **Examples:** SINAMICS_Integrated.Axis_Red.actor.STW1 SINAMICS_Integrated.Axis_Blue.actor.ZSW1 SINAMICS_Integrated.Axis_Red.encoder_1.Gx_ZSW

Interconnection target	Syntax
Message frame of the infeed unit	If the automatic message frame determination is activated, the message frame for the infeed is specified automatically (PROFIdrive message frame 370). You can now create two I/O variables in WORD format via symbolic assignment in the address list for the status and control words (ZSW1/STW1) of the infeed. You require these I/O variables for the **FB _LineModule_control** for the control of the line module. For further information refer to the *Standard Function for SINAMICS S120 Line Modules* Function Manual
Onboard I/Os of SIMOTION C	<device>.<interface_name> **Examples:** C240_1.AnalogActor_3 [ANALOG/STEPPER 3, X2] C240_1.Encoder_2 [ENCODER 2, X4] C240_1.DI_0 [I 0, X1.28] C240_1.DO_7 [Q7, X.19] C240_1.M1 [M1, X1.26] C240_1.B2 [B2, X1.23]

4.3 Output Cams and Measuring Inputs

4.3.1 Preface

This **document** is part of the
System and Function Descriptions documentation package.

Scope

This manual is valid for SIMOTION SCOUT in conjunction with the SIMOTION Cam, Path or Cam_ext technology package for product version V4.5.

Chapters in this manual

This manual provides information about the functions, operation, command execution, and technology alarms of the technology objects.

- **Output Cam technology object** (part I)
  Functions and operation
- **Cam Track technology object** (part II)
  Functions and operation
- **TO measuring input** (part III)
  Functions and operation
- **Index**
  Keyword index for locating information

4.3.1.1 SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

4.3.1.2 Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.3.2 Fundamental safety instructions

4.3.2.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

4.3.2.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

4.3.2.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.3.3 Output Cam TO - Part I

4.3.3.1 Overview of Output Cam TO

General information about the Output Cam TO

The **Output Cam** technology object

- Generates position-dependent switching signals
- Can be assigned to positioning axes, synchronous axes, path axes or external encoders
- The axes can be real or virtual.

Different switching signals distinguish different types of output cam:

- **Software cam**
  Switching signals are used internally in the user program by evaluating the relevant **state** system variable.
- **Hardware cam**
  Switching signals are output **externally** on I/O devices by assigning a digital output to the Output Cam TO. For example, digital output modules from the ET 200 I/O system can be used for the cam output.

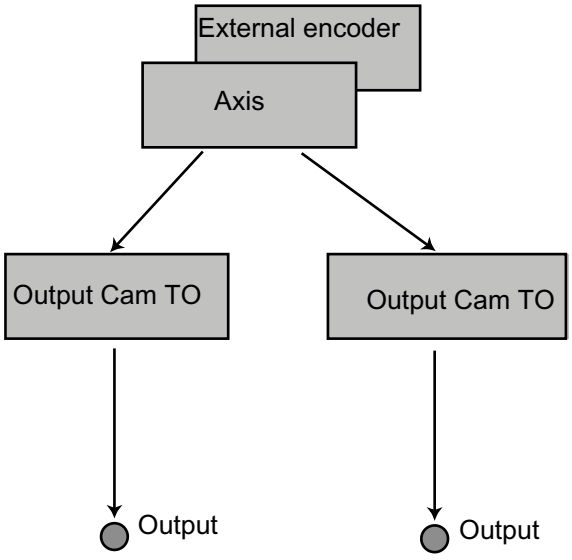


Figure 4-182 Interconnection options for the output cam TO

A range of output cam types with different switching behaviors are available.

- **Position-based cam**
  The switching signal is supplied between the switch-on position and the switch-off position.
- **Time-based cam**
  The switching signal is supplied for a specified time period after the switch-on position is reached.
- **Unidirectional output cam**
  The switching signal is supplied when the axis reaches the switching position and is then reset by the user.
- **Counter cam**
  Counter cams are not a separate output cam type, but rather position-based or time-based type cams. Counter cams can be configured so that they are output for every switching or for every nth switching. They can only be programmed and activated in the user program.
- **Cam output types**
  While output cams are usually output in the IPO cycle clock or the position control cycle clock, high-speed output cams provide better output accuracy than the position control cycle clock because the switching edges are positioned within the position control cycle clock.

Functionality

It is possible to define an **effective direction** for the Output Cam TO, i.e. the output cam is only active when the direction of motion of the axis is the same as the effective direction.

The output cam can be calculated in servo cycles, IPO cycles, or IPO_2 cycles.

Note

With modules D435-2 DP/PN, D445-2 DP/PN, and D455-2 DP/PN, cam calculation can be performed in Servo_fast or IPO_fast. For more information, see the chapter Second servo cycle clock (Servo_fast) in the Motion Control Basic Functions manual.

The reference values of the output cam depend on the axis type or the external encoder:

Table 4-153 Reference to the actual or set position

Technology object	Reference to actual position possible	Reference to set position possible
Real drive axis	-	-
Real position axis	X	X
Real synchronized axis	X	X
Virtual axes	-	X
External encoder	X	-

Possible configuration of the cams with actual value reference:

- **Reference to the actual value on the encoder without considering Ti**. Reference to the actual position in the controller before the position filter.
- **Reference to the actual value after the position filter**. Reference to the actual position in the controller after the position filter.
- **Reference to the actual value on the encoder. Ti is taken into account**. Reference to the actual value on the encoder module / drive. The transmission time Ti from the encoder module / drive to the controller is taken into account by the system.

Possible configuration of the cams with setpoint reference:

- **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoints will already have been reached at the end of this IPO cycle, e.g. if a setpoint is calculated for the IPO cycle clock with the virtual axis and also displayed at the end of the cycle. Reference is made directly to the setpoint applicable at the end of the cycle.
- **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoint calculated in the interpolator will be output completely in the following cycle and the calculated position setpoint will therefore be reached at the end of the following cycle.
- **Reference to the setpoint on the drive**. Calculation of the cam switch points according to the setpoint output with the current settings on the drive.

In this case, the output cam functionality can be applied to axes or external encoders with or without modulo properties.

The output cam is also effective for axes that have not been homed.

The Output Cam technology object is assigned to exactly one output during configuration. Output can be achieved via:

- Onboard I/O
- Drive I/O (e.g. TB30, TM31, TM1x)
- SIMOTION C centralized I/O
- Distributed I/O; PROFIBUS DP IO (e.g. ET 200M) and PROFINET IO (e.g. ET200xP TM Timer DI/DQ)

However, the output must not be in the process image.

The switching accuracy is dependent on the following:

- Output accuracy of the I/O
- How the output cam is allocated in the task system
- How constant delay times are compensated

Several Output Cam TOs can be connected to the same output (see Chapter **Logical operation**). Alternatively, the Cam Track TO can be used for this purpose.

Example

Lines of glue are applied to a wooden board. The output cams are assigned to an external encoder. Output cams assigned to outputs Q 0 to Q 4 are switched on and off at specified positions.

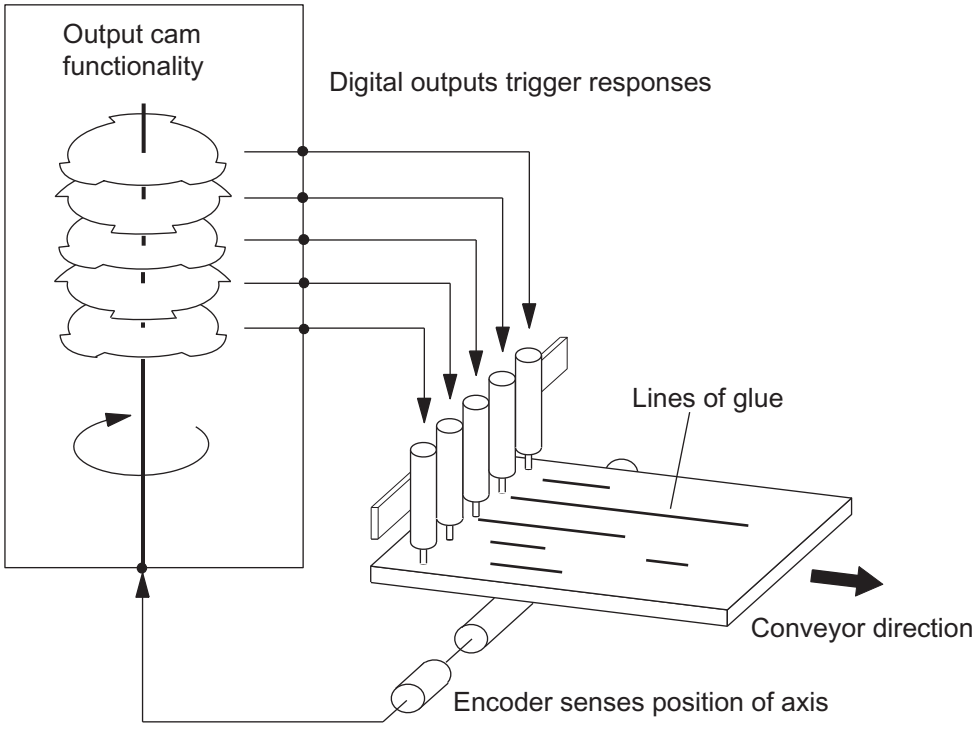


Figure 4-183 Example of an electronic cam control

See also

Logical operation (Page 1822)

Comparison of Output Cam TO and Cam Track TO

Depending on the application, it may be practical to use either the Cam Track TO or one or more Output Cam TOs. The table below should help you to decide which TO should be used in which case.

Table 4-154 Comparison of Output Cam TO and Cam Track TO

Features	Output Cam TO	Cam Track TO
Availability	• As of Version 1.0	• As of Version V3.2
Supported output cams	• Position-based cam • Time-based output cam • Unidirectional output cam • Counter cam • Exact time setting of an output, exact time output cams (as of V4.1)	• Position-based cam • Time-based output cam • Time-based cam with maximum ON length
Several output cams on one output	• Via logical operation (AND/OR)	• Maximum 32 output cams of the same type in one track • No cam track logical operations (AND/OR)
Different types of output cam on one output	• Via AND/OR	• Not available
Output cam definition	• Related to axis • Via system variables	• Related to cam track (cam track can be mapped as required on axis) • Via system-variables array
Hysteresis	• Available	• Available
Effective direction	• Available	• Not available
Derivative-action times	• Separate for power ON/power OFF	• Separate for power ON/power OFF
Deactivation time for time-based cam	• As of Version V3.2	• As of Version V3.2
Activation/deactivation types	• Active immediately	• Start and stop mode parameterizable
Types of output	• Cyclic	• Cyclic • Once
Output cam status	• System variable	• Status of single output cams over one array of byte
Output cam enable	• Via_enableOutputCam	• via_enableCamTrack • Validity of single output cams configurable via system variables

Features	Output Cam TO	Cam Track TO
Performance	• Depends on number of single output cams	• When 5 or more output cams are used in one output cam track instead of 5 single output cams, the output cam track performs better. This performance advantage amounts to at least a factor of 2 for 32 single output cams.
MCC command available	• Available	• Available (V4.0 and higher)

4.3.3.2 Output cam TO basics

Output cam type

Software cam

Switching signals are used **internally** in the user program by evaluating the relevant **state** system variable.

Hardware cam

Description

Switching signals are output **externally** on I/O devices by assigning a digital output to the Output Cam TO.

The following can be used as digital outputs:

- Onboard outputs (SIMOTION C, D, ...)
- Centralized I/O (SIMOTION C)
- Distributed I/O via PROFIBUS DP (e.g. ET 200M) and PROFINET IO (e.g. ET 200S, ET200xP TM Timer DI/DQ)
- Drive I/O (for example, TM15 and TM17 High Feature Terminal Modules)

Hardware for output cams

Cam output on cam output (I/O channel is configured as CAM)

- SIMOTION D410-2
- SIMOTION D4x5-2
- TM15, TM17 High Feature
- PROFINET IO (e.g. ET200xP TM Timer DI/DQ)

Cam output on high-speed output with direct access (I/O channel is configured as DO)

- SIMOTION D4xx / D4x5-1
- SIMOTION C240, C240 PN

Cam output on standard output (I/O channel is configured as DO)

- SIMOTION C/D/CX onboard I/O
- SINAMICS onboard I/O
- TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30
- Standard DO (SIMATIC ET200, ...)
- PROFINET IO (e.g. ET200xP TM Timer DI/DQ)

For more information, see cam output types. (Page 1811)

Position-based cam

Direction-neutral switching

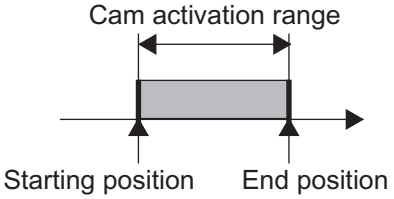


Figure 4-184 Position-controlled output cam with start position less than end position

Limits imposed by start and end positions

The output cam is activated:

- Axis position is within the switch-on area
- Axis position value is shifted into the switch-on area of the output cam
  The position value of the interconnected object can change abruptly, for example, when it is homed or when its coordinate system is shifted with the **_redefinePosition** command.

The output cam is switched off:

- When the axis position is outside the start or end position
- When the axis position value is shifted outside the switch-on area
- When commands are issued that deactivate the output cam, e.g. **_disableOutputCam**, **_setOutputCamState**, and **_resetOutputCam**

Switch-on area

The switch-on area of the output cam is defined from the start position to the end position in a positive counting direction, i.e. within a range between the start position and the end position. If the end position is greater than the start position, the switch-on area is defined by the start and end positions (see figure above).

The switch-on area is outside the area between the end and start positions if the end position is less than the start position (see figure below).

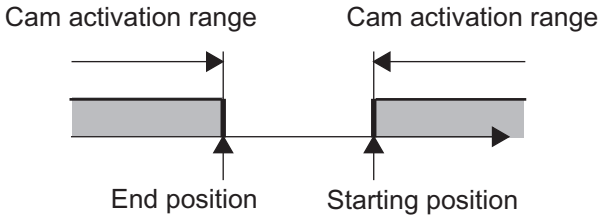


Figure 4-185 Position-controlled output cam with end position less than start position

Note

This definition of the switch-on area is possible for all modulo and non-modulo axes.

ON duration

The ON duration of the output cam depends on the velocity at which the axis traverses the output cam length.

Direction-dependent switching

The output cam is activated:

- When the axis position is between the start and end positions, and the axis is moving in the programmed effective direction

The output cam is switched off:

- When the axis position is outside the start or end position
- When the motion direction is not the same as the assigned effective direction
- When the axis position value is shifted outside the switch-on area
- When commands are issued that deactivate the output cam, e.g. **_disableOutputCam**, **_setOutputCamState**, and **_resetOutputCam**

Time-based output cam

Direction-neutral switching

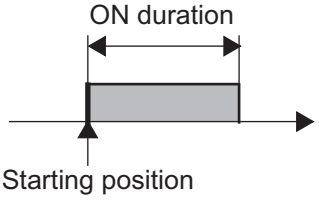


Figure 4-186 Time-controlled output cam

Limits imposed by starting position and ON duration

The output cam is switched on:

- At the starting position.
  If the starting position is overrun again during the ON duration, the time-based cam is not switched on again. It is not possible to retrigger a time-based cam.

The output cam is switched off:

- When the assigned time period expires
- When commands are issued that deactivate the output cam, e.g. **_disableOutputCam**, **_setOutputCamState** and **_resetOutputCam**

Output cam length

The output cam length is dependent on the velocity at which the assigned axis traverses during ON duration of the output cam.

Direction-dependent switching

The output cam is switched on:

- At the starting position if the traversing direction is the same as the effective direction

The output cam is switched off:

- When the assigned time period expires
- When commands are issued that deactivate the output cam, e.g. **_disableOutputCam**, **_setOutputCamState** and **_resetOutputCam**

A change of direction will not lead to the output cam being switched off if the time-based cam has already been activated.

Unidirectional output cam

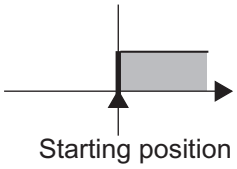


Figure 4-187 Unidirectional output cam

Limits imposed by starting position

The output cam is switched on:

- At the starting position if the axis is moving in the programmed effective direction

The output cam is switched off:

- By the **_disableOutputCam**, **_setOutputCamState** and **_reset** commands

Note

The unidirectional output cam does not switch unless the starting position is explicitly crossed, e.g. by setting the actual value.

An end position is not defined for the unidirectional output cam. The output cam signal depends solely on the switching criteria when the output cam is crossed over. The unidirectional cam can be reset via the program (e.g. by calling up the system function **_enableOutputCam** again).

Counter cam

For a counter cam, it can be specified whether the output cam is to be output **every** time it switches or every **nth** time it switches.

Note

Counter cams can only be configured for position-based and time-based cams. A counter cam is used via the **_setOutputCamCounter** system function.

Counter cams can only be defined in the user program. In configuring the output cam, the output cam type **cannot** be defined as counter cam.

Every counter cam has a starting count value and a current count value.

The current count value for the output cam is reduced by 1 every time the output cam switches. If the current count reaches 0, the output cam is output (**state** system variable and output cam output). At the same time, the current count value is reset to the starting count value. If the current count value does not reach 0, the output cam output is suppressed. The default setting of the starting count value and current count value is 1. The starting count value and current count value are programmed by means of the **_setOutputCamCounter**. The current count values can be scanned with the **counterCamData.actualValue** and **counterCamData.startValue** system variables. No resetting of the values by the system takes place, e.g. after **_enableOutputCam** or **_disableOutputCam**.

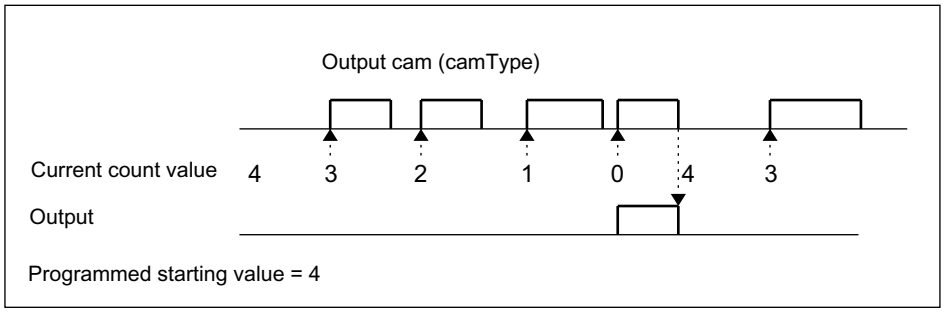


Figure 4-188 Example of a counter cam

Cam output types

The cam calculations are performed in the processing cycle clock (IPO or IPO_2 cycle clock or in the servo cycle clock). For the possible setting of Servo_fast or IPO_fast, see Section Second servo cycle clock (Servo_fast) in the SIMOTION Runtime Basic Functions manual.

The temporal resolution of the cam output depends on the hardware used and the setting in the configuration. In standard applications, the setting is undertaken using screen forms. The configuration data can also be set via the expert list.

The possible setting options for cam output are described below:

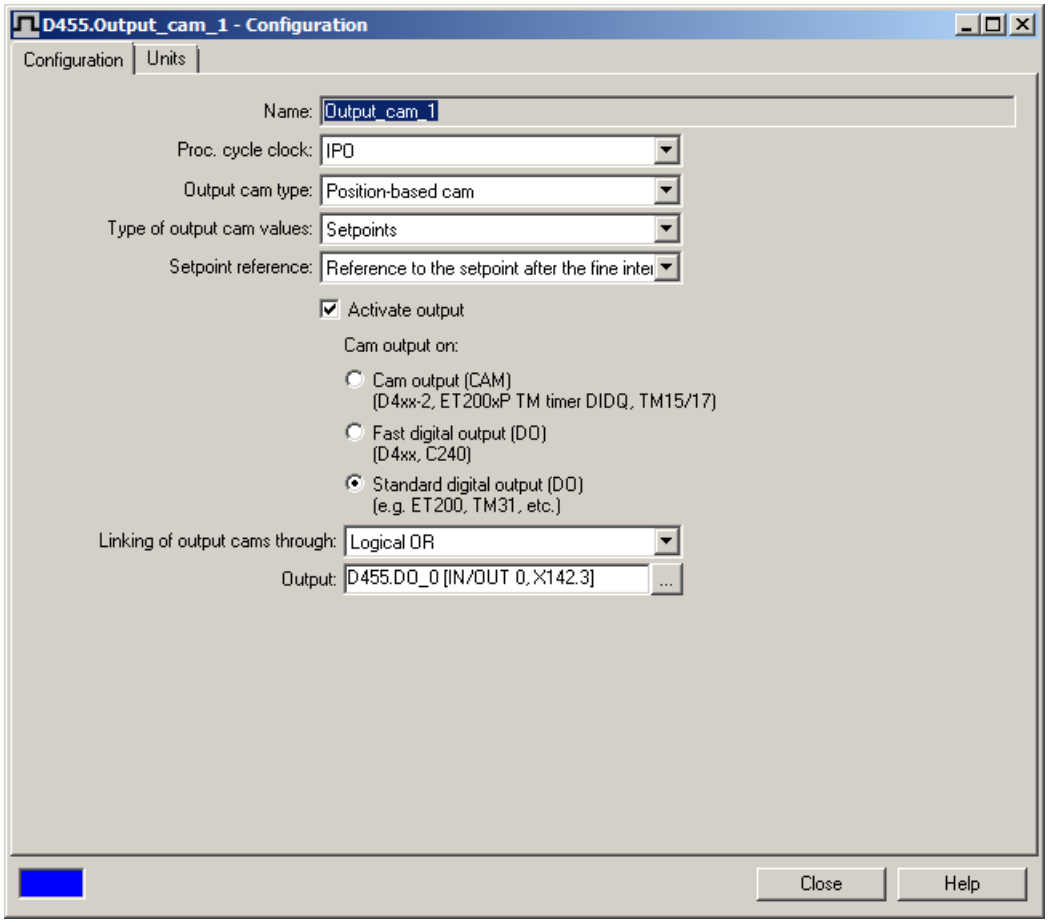


Figure 4-189 Output cam configuration using the example of a position-based cam

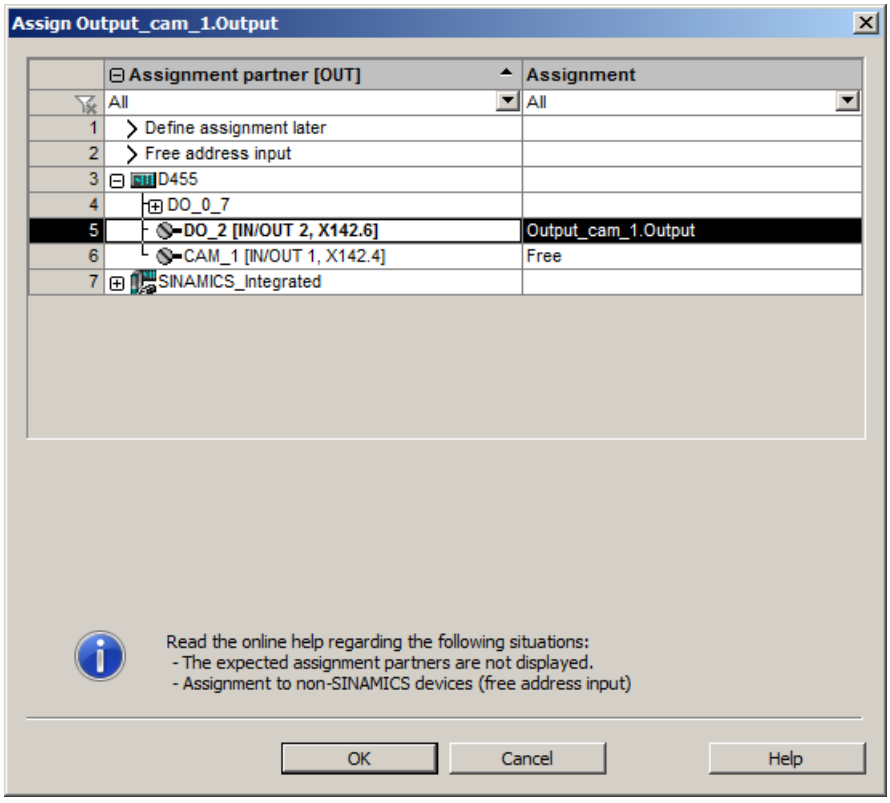


Figure 4-190 Assignment dialog

Symbolic assignment is activated by default in projects as of V4.2 (**Project > Use symbolic assignment**)

Cam output (CAM)

The cam output is performed on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. In the case of D4x5-2 and the TM17 High Feature, the resolution is 1 µs.

Hardware supported

- SIMOTION D410-2
- SIMOTION D4x5-2 (X142)
- TM15, TM17 High Feature
- ET200xP TM Timer DI/DQ

The I/O channel must be configured as CAM.

SIMOTION D410-2

The digital inputs/outputs are used for the cam output in the D410-2. The digital inputs/outputs can be used as the cam output (CAM) from the user program.

As of the editorial deadline of this documentation, the output cam resolution for D410-2 was not yet certain. The information can be found at the following website Additional information about the SIMOTION D manuals (http://support.automation.siemens.com/WW/view/en/27585482).

SIMOTION D4x5-2 onboard outputs (interface X142)

The D4x5-2 onboard outputs can be used as cam output (CAM) from the user program. The D4x5-2 onboard outputs are permanently assigned to SIMOTION. The X142 I/Os are configured using HW Config.

The X142 configuration screen form can be accessed directly from the project navigator in SIMOTION SCOUT.

With SIMOTION D4x5-2, output cams are output at the X142 interface with a resolution of 1 µs.

TM15 / TM17 High Feature Terminal Modules

The TM15 and TM17 High Feature Terminal Modules can be used to set up cam outputs (CAM) within the SIMOTION Motion Control system. The Terminal Modules are connected directly to SIMOTION D or CX32/CX32-2 via DRIVE-CLiQ for this purpose.

Alternatively, TM15 and TM17 High Feature can be connected to a SINAMICS S120 CU320/ CU320-2/CU310/CU310-2 Control Unit with higher-level SIMOTION C, P or D.

Output cams on the TM15 operate with DRIVE-CLiQ cycle-clock resolution (typically 125 µs). Output cams on the TM17 High Feature have a resolution of 1 µs.

If current controller cycle clocks other than 125 µs are used, the parameter calculations of the drive must be taken over into the PG and the Fast IO configuration must be recreated when using cam outputs on TM15/TM17 High Feature. (For more information, see chapter Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs in the TM15 / TM17 High Feature Terminal Modules Commissioning Manual).

ET200xP TM Timer DI/DQ

The digital inputs/outputs are used for the cam output in ET200xP TM Timer DI/DQ. The digital inputs/outputs can be used as the cam output (CAM) from the user program. Output cams on the ET200xP TM Timer DI/DQ have a resolution of 1 µs.

High-speed digital output (DO)

The cam output is performed via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle clock.

The time that it takes for the axis to reach the output cam switching position with reference to the processing cycle is calculated by linear extrapolation. Calculated from the beginning of the 1st position control cycle, the output cam function is triggered by a hardware time when this time is reached.

Hardware supported

The onboard I/O of the following CPUs is used:

- SIMOTION D4x5 (interface X122, X132), 8 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**)
- SIMOTION D410 (interface X121), 4 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**)
- SIMOTION C240, C240 PN (interface X1), 8 high-speed cam outputs

SIMOTION D410/D4x5 onboard outputs

Output cams are output via **a high-speed digital output (DO).**

- Up to and including SIMOTION V4.1 SP5, all D410/D4x5 onboard I/Os configured as digital outputs are exclusively available to SIMOTION
- As of SIMOTION V4.2, D410/D4x5 onboard I/Os configured as digital outputs can be switched over to SINAMICS using BICO interconnection (channel granular)

Standard digital output (DO)

The output cam calculations are performed in processing cycles (IPO or IPO_2 cycle clock or servo cycle clock).

Actual cam output is performed in servo cycles. The temporal resolution of the cam output is usually reduced by the output cycle of the I/O used.

Therefore the resolution

- with standard I/O (e.g. ET 200) depends on the cycle time of the bus system (PROFIBUS DP / PROFINET IO)
- with TM15 / TM17 depends on the cycle time of the bus system (PROFIBUS Integrated / PROFIBUS DP / PROFINET IO)
- with TM15 DI/DO, TM31, TM41, TB30 depends on the configured sampling time
  - cu.p0799 (CU inputs/outputs sampling time) for the TB30 and onboard outputs
  - p4099 (TMxx inputs/outputs sampling time) for TM15 DI/DO, TM31 and TM41

Hardware supported

- Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx)
- Centralized I/O (SIMOTION C)
- Distributed I/O via PROFIBUS DP/PROFINET I/O (e.g. ET 200, ET200xP TM Timer DI/DQ)
- Drive I/O TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30

Configuration data of cam output types in expert list

Table 4-155 Setting options for cam output

Selection in configuration screen	Setting in expert list
Cam output (CAM) (D4xx-2, ET200xP TM Timer DIDQ, TM15/17)	OcaBaseCfg.outputType = [1] TIME_STAMP OcaBaseCfg.hwTimer = [91] NO
High-speed digital output (DO) (D4xx, C240)	OcaBaseCfg.outputType = [0] STANDARD OcaBaseCfg.hwTimer = [173] YES
Standard digital output (DO) (Standard DO, e.g. ET200, TM31)	OcaBaseCfg.outputType = [0] STANDARD OcaBaseCfg.hwTimer = [91] NO

Output cams on **cam output (CAM)** or on **high-speed digital output (DO)** are also referred to below as high-speed, hardware-supported output cams.

Note

Further information and the output accuracy for high-speed output cams is described in the PM21 Catalog and in the respective product brief or commissioning/equipment manuals.

Commissioning Manual *Terminal Modules TM15/TM17 High Feature*
Operating Instructions *SIMOTION C2xx*
Commissioning Manual *SIMOTION D410*
Commissioning Manual *SIMOTION D410-2*
Commissioning and Hardware Installation Manual *SIMOTION D4x5*
Commissioning and Hardware Installation Manual *SIMOTION D4x5-2*

Commissioning Manual *Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA*

See also

Configuring cams on SIMOTION D4xx onboard (Page 1835)

Exact time setting of an output, exact time output cams (as of V4.1)

You can switch a high-speed output cam On/Off at an exact time within an execution cycle (position control, IPO cycle) via the **timeOffset** parameter of the **_setOutputCamState** system function.

Enter an offset of the switching edge in the configured unit (e.g. s) of the Output Cam TO in the **timeOffset** parameter. The reference point of the offset is the start of the next execution cycle of the Output Cam TO. You can read out the value of the time offset (system-dependent execution time between the execution cycle and the output cycle) in the **tOutput** system variable. The time in **tOutput** is the earliest possible time to switch the output cam. The **timeOffset** is added to this time.

Features

- The offset must be less than the cycle time of the processing cycle clock. The offset is limited automatically and a technological alarm is output when the cycle time is exceeded.
- It is possible to switch on or off within one cycle.
- When this function is used, there is a dependency on the processing cycle clock and the set cycle clock times.
- The offset is valid for every output cam type. If only switched once, the unidirectional output cam type is recommended.
- The offset can only be stated if the output cam TO on the outputs is configured with a time stamp (cam output (CAM)).
- For output cams without time stamp (e.g. C240-2 onboard output cams), **tOutput** (time stamp) is set to 0.0.

- If multiple activation or deactivation signal edges are output within one cycle, the most recently written values apply.
- The start of the output cycle is shifted with respect to the beginning of the processing cycle by the value output in the **tOutput** system variable and the specified **timeOffset**.

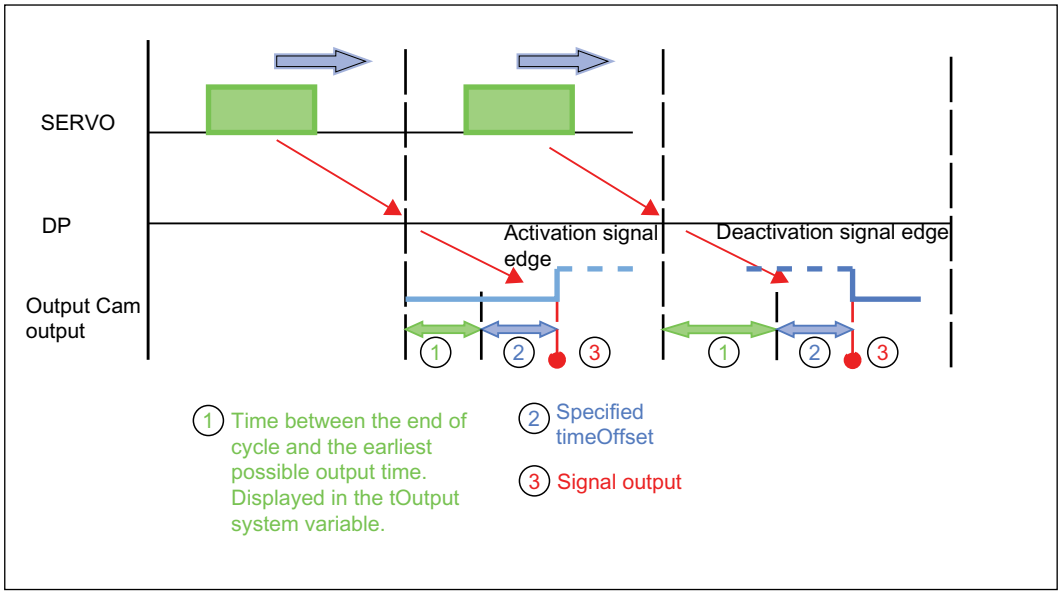


Figure 4-191 Exact-time output setting for DP:POSITION CONTROL=1:1

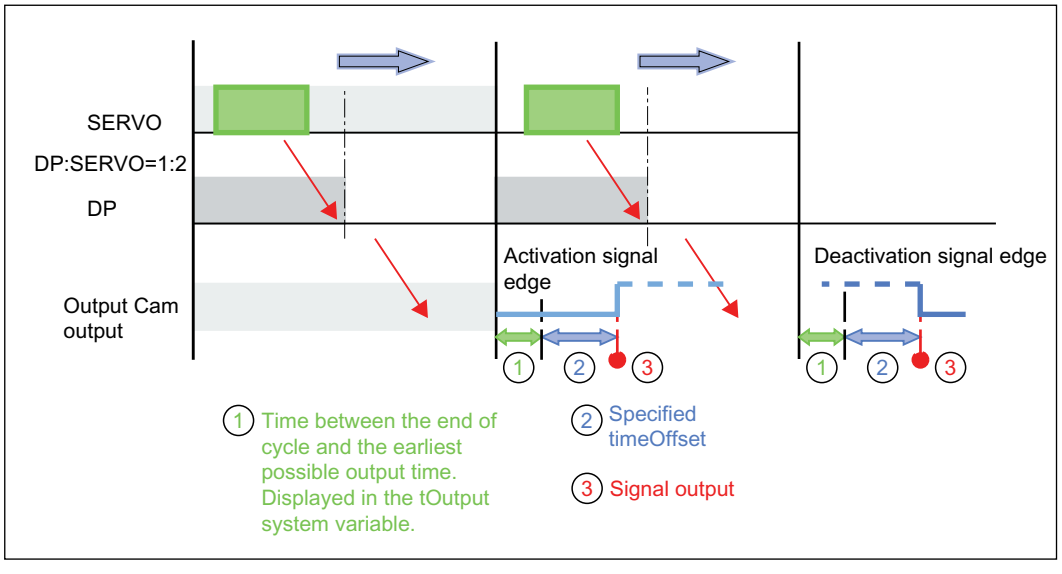


Figure 4-192 Exact-time output setting for DP:POSITION CONTROL=1:2

Cam parameters

Reaction, effective direction

Behavior

The following diagram shows output cam behavior when switching on and off, without hysteresis, activation, or deactivation time.

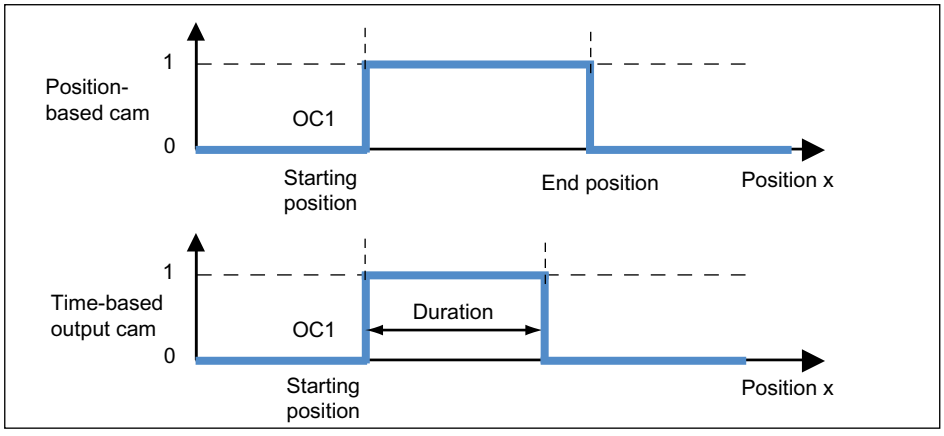


Figure 4-193 Output cam behavior when switching on/off

The switching action depends on the position only (position setpoint or actual position).

Effective direction

You can define a default effective direction when you activate output cams. The output cam only switches when the motion direction and effective direction are identical.

Options:

Table 4-156 Effective direction and behavior

Effective direction	Behavior
Positive	The output cam is activated only in positive direction of motion.
Positive and negative	The output cam is activated independent of the direction of motion.
Negative	The output cam is activated only in negative direction of motion.
Last programmed direction of rotation	With this setting, the output cam switches for the last programmed direction of rotation. If no direction of rotation has been previously programmed, the default setting is used.

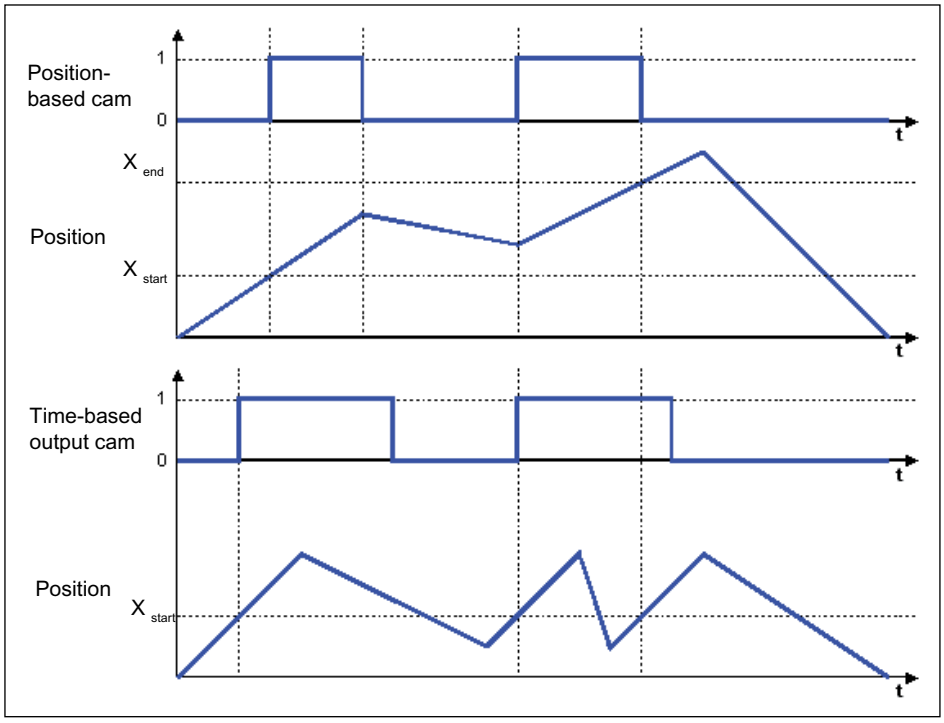


Figure 4-194 Positive effective direction and output cam switching behavior

Hysteresis

If the actual position value tends to fluctuate due to mechanical influences, specification of a **hysteresis** prevents the output cam from unintended switch status changes.

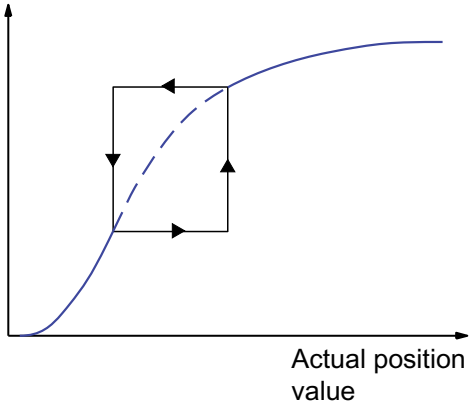


Figure 4-195 Hysteresis

Hysteresis range conditions

- Hysteresis is not activated until the direction has been reversed.
- The direction of motion is not redefined within the hysteresis.

- Within the hysteresis, the switching state of position-based cams is not changed.
- If modified switching conditions for the output cam are detected when the output cam is outside the hysteresis range, this current switching state is set.

Example: position-based cam hysteresis

Output cam configuration:
output cam type: position-based cam; switch-on position, 20 mm; switch-off position, 200 mm; hysteresis, 20 mm; effective direction: positive

Axis positions:
0 mm -> 100 mm -> 10 mm -> 50 mm -> 0 mm -> 150 mm -> 0 mm

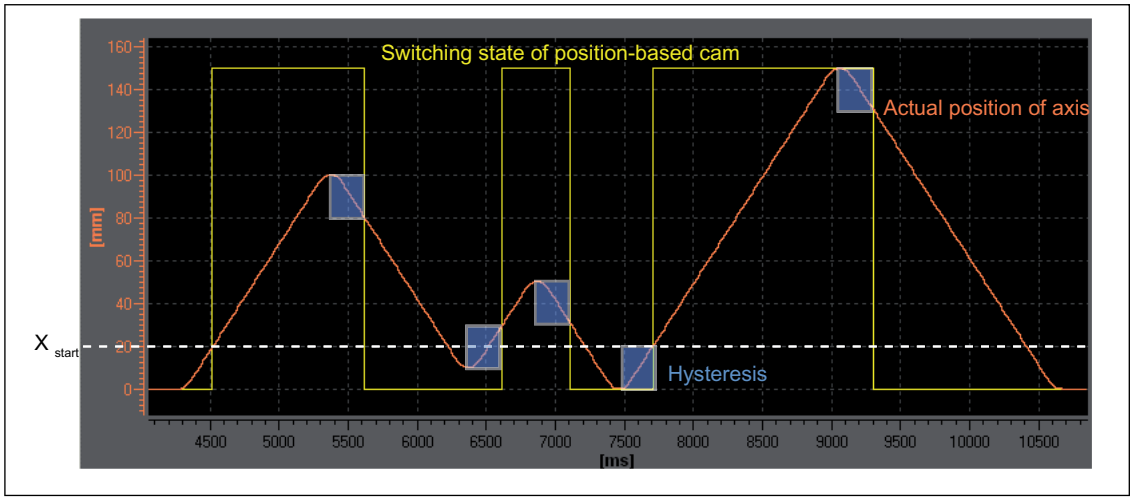


Figure 4-196 Hysteresis range (height of blue sections) and behavior of a position-based cam, positive effective direction

Output cam's second switch-on point is moved to position 30 mm, due to hysteresis (see figure above).

Example: time-based cam hysteresis

Output cam configuration:
output cam type: time-based cam; switch-on position, 40 mm; ON duration, 0.5 s; hysteresis, 20 mm; effective direction: positive

Axis positions:
0 mm -> 100 mm -> 20 mm -> 60 mm -> 30 mm -> 80 mm -> 10 mm -> 150 mm

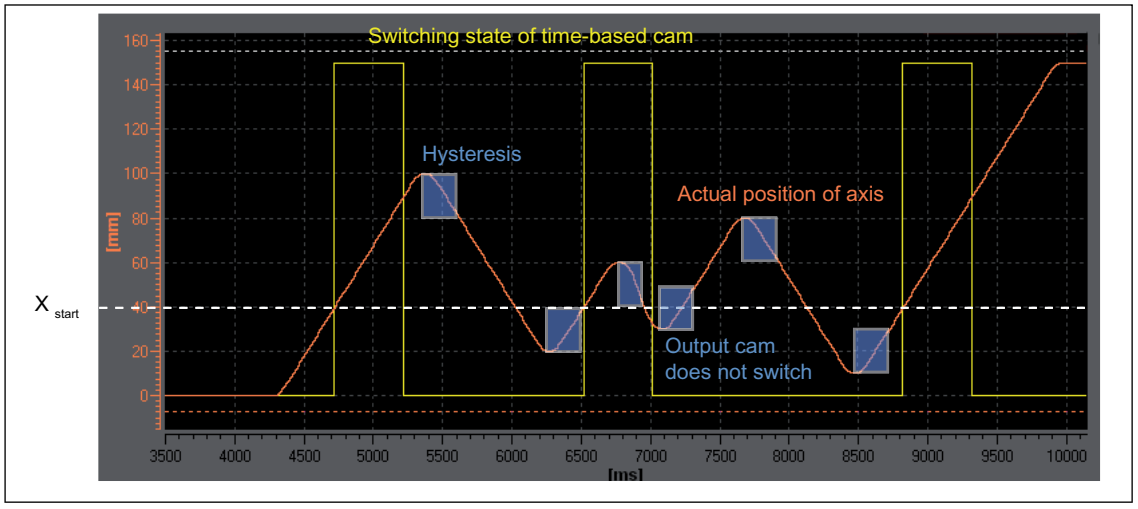


Figure 4-197 Hysteresis range (height of blue sections) and behavior of a time-based cam, positive effective direction

Time-based cam switches off only after ON duration has expired, not after change of direction.

Time-based cam with a start position within the hysteresis range is not output (see figure above).

Hysteresis range

The upper limit of the hysteresis range is set at 25% of the working range for a linear axis, and 25 % of the rotary axis range for a rotary axis. If you violate this maximum setting, an error message is issued. In practice, a lower setting is used for the hysteresis range.

- **Path-controlled output cam**
  The hysteresis becomes active after direction reversal is detected. If only a positive or only a negative effective direction has been parameterized for an output cam, the output cam does not switch off after a reversal of direction until it has left the hysteresis.
- **Time-based cam**
  The switching behavior of a time-based cam is determined by the ON duration, not by the hysteresis. This means that an entered hysteresis range has no influence on the ON duration of an output cam. It only has an influence on the switch-on time (start position).

Note

If a time-based cam's start position lies within the hysteresis, it is **not** output.

Derivative-action times (activation/deactivation time)

To compensate for the switching times of digital outputs and connected switching elements, or of propagation delays, it is possible to specify **actuation times**. Actuation times are calculated from the sum of all delay times and can be specified separately for activation and deactivation edges as an actuation time at the activation edge (activation time) or an actuation time at the deactivation edge (deactivation time).

The activation/deactivation times of the TO outputCam are dynamically compensated by means of the derivative-action times. In this way, output cams are dynamically shifted depending on the actual velocity.

For example, a valve that should open at 200°, with an activation time of 0.5 s

- Must be controlled at 195° at a velocity of 10°.
- Must be controlled at 190° at a velocity of 20°.

This dynamic shift takes place automatically by means of the Output Cam TO.

Settings for the activation and deactivation times can contain positive or negative values.

A negative activation time must be entered if the output cam is to be switched before the programmed start of the output cam.

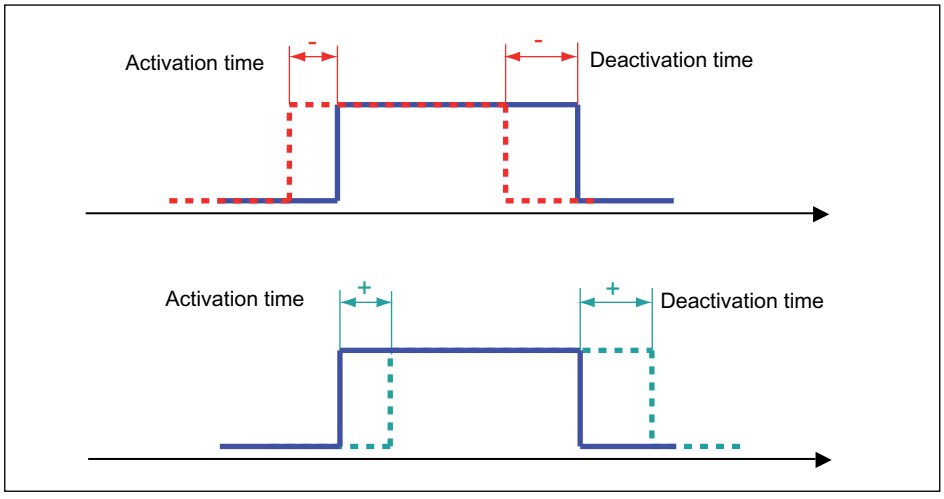


Figure 4-198 Switching behavior at varying actuation times

Note

The time of output for the output cam in the controller is relevant for calculation of the dynamic adjustment. If velocity changes up to signal output, these changes are no longer taken into account.

Dead times, e.g. PROFIBUS DP communication times, output delay times on digital outputs, etc., are taken into account in the actuation time.

Long actuation times exceeding one modulo cycle may lead to heavy fluctuation of the switching position of actual value output cams (actual value curve). Here, setpoint output cams should be used or the actuation time should be considerably less than one modulo cycle.

The system takes into account the specified actuation times when the output cams are calculated and managed. The switching positions of the output cams are calculated taking into account the activation time and deactivation time in relation to the present velocity. If, allowing for actuation times, the output cam was switched, then the system deems this operation to have occurred, and it does not switch the output cam again even if any subsequent current velocity changes occur.

The dynamic actuation of modulo axes can be greater than one modulo length. However, the number of switching operations is not collected by the system, i.e. for actuation times longer than one modulo length, a switching operation cannot take place in each modulo cycle. One switching operation is active in the system at any given point in time. A switching operation is completed when the output cam is switched off.

Actuation times and cycle clock settings

A change of cycle clock settings does not have to be taken into account for the actuation time settings (activation/deactivation time). These are, for example:

- Changing the Servo/IPO/IPO_2 clock settings (for example, from "1/1/1 ms" to "2/2/2 ms").
- Change of processing cycle clock of the TO outputCam (setting: Servo cycle clock, IPO cycle clock, IPO_2 cycle clock).

Deactivation time for time-based cam

Deactivation time is also taken into account in setting a time-based cam.

Deactivation time must be:

- Deactivation time ≤ activation time + ON duration

Activation and deactivation times can vary independently of the I/O and can, therefore, influence the ON duration of the time-based cam.

To achieve compatibility with older software versions (<V3.2), deactivation time for time-based cams can be activated or deactivated in the **Defaults** window, by means of the **Use deactivation time** checkbox.

See also

Determining derivative-action times for output cams (dead time compensation) (Page 1833)

Logical operation

Through a setting in the **LogAdress.logicOperation** configuration data element you can specify whether the output cam is connected to the output using an AND or OR operation.

That is, all ORed output cams will be grouped and then logically linked at the output with the output cams linked by AND logic.

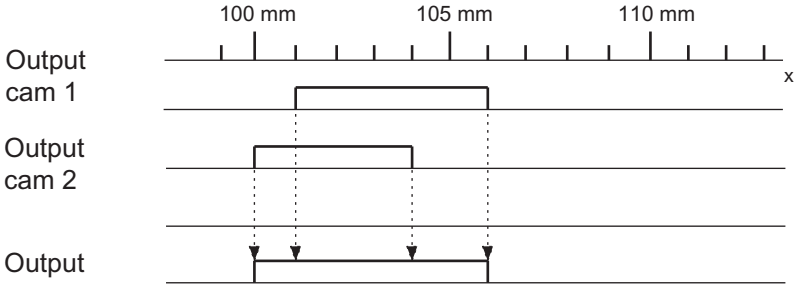


Figure 4-199 OR operation of two output cams

Note

If hardware output cams are configured, you can configure an I/O variable in the symbol browser for monitoring.

See also

Output cam configuration (Page 1826)

Simulation

Operation can be simulated by means of the simulation commands on the output cam. The output cam status is then not output to the hardware output. In simulation mode, a hardware cam behaves as a software cam. It is then only used for programming purposes.

If an active output cam is switched to simulation mode (**_enableOutputCamSimulation**), the output cam status remains the same, and only the control of the output is reset or interrupted.

Inversion

The inversion of single output cams is available and is set on the **_enableOutputCam** command by a parameter (**invertOutput**).

Configure Units

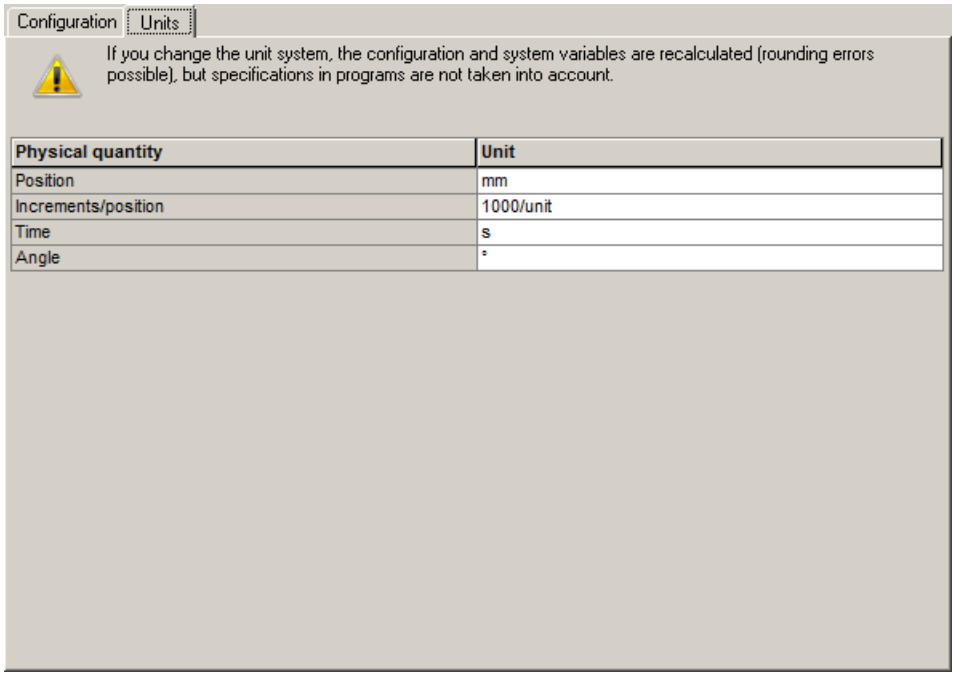
You can define the basic units for each technology object. The same physical variables can have different units in different technology objects. These are converted:

How to configure the units:

1. In the project navigator, open the context menu for the technology object.
2. In the context menu, select **Expert > Configure units**. The **Configure Units** window appears in the working area.
3. Select the **unit** for the **physical variables**. These units are used for the technology object, e.g. s for time units.

or

1. In the project navigator, open the **Configuration** under the TO.
2. Select the **Units** tab.



You can set the following parameters:

Field/button	Meaning/instruction
Table with units	
Physical variable column	Shows the physical variable. The physical variables which are used by the TO are available for the configuration.
Unit column	Displays and configures the unit. A drop-down list for selecting the unit appears when you click on the cell.
Toolbar	
	Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display
Close	Button for closing the dialog.
Help	Button for opening the online help for the dialog.

4.3.3.3 Configuring the Output Cam technology object

Insertion of Output Cam

Note

Before you insert an output cam, the axis (position or synchronous axis) or external encoder to which the output cam is assigned has to be created.

If the cam is output to a TM15/TM17 High Feature or ET200xP TM Timer DI/DQ, the module must be added and configured before cam configuration.

To insert a new output cam:

1. In the project navigator, highlight the **OUTPUT CAMS** folder under the relevant axis or external encoder.
2. Select **Insert > Technology object > Output cam** or double-click **Insert output cam** in the project navigator under the axis or external encoder in the OUTPUT CAMS folder. The **Insert output cam** window appears.
3. Enter a **name** for the output cam. You can also enter a **comment**.
   Names must be unique throughout the project and must comply with ST syntax conventions. For this reason, all the existing output cams are displayed under **Available output cams**.
4. Confirm with **OK**. In the working area, the window for the configuration is displayed and the created output cam TO is shown in the project navigator.

Parameterize Output Cam technology object

General information about configuration data and system variables

Two data classes are distinguished when parameterizing a TO.

Configuration data defines the principal functionality of a TO. They are set within the object configuration framework with the SCOUT engineering system and are not normally changed during runtime.

System variables provide status data of the TO for the user program and a parameterization interface on the TO. System variables can be changed during runtime.

Note

You will find more information on technology objects in the *SIMOTION Runtime Basic Functions functional description*.

To parameterize an output cam:

1. In the project navigator under the folder **OUTPUT CAMS**, find the output cam TO that you want to parameterize. Double-click the output cam TO to display the associated objects.
2. Double-click Configuration or Default in the project navigator. The window appears on the workspace.
   - **Configuration**:
     Define the **configuration data** of the output cam here. This includes, for example, output cam type.
   - **Default**:
     Define the output cam defaults of the **system variables** here. This includes, for example, the effective direction.
3. Change the configuration data and output cam defaults.
4. Click **Close** to accept the changes.
5. Repeat steps 2 to 4 for all objects in which you want to change the configuration data and output cam defaults.

See also

Output cam configuration (Page 1826)

Defining output cam defaults (Page 1831)

Using the expert list for output cams

For standard SIMOTION applications, necessary parameters (configuration data and system variables) are parameterized into the Output Cam technology object directly in screen forms or are defined automatically.
It may be necessary to change automatically defined parameters for special SIMOTION applications. These configuration data and system variables can only be displayed and changed in the expert list.

Note

You will find more information on working with the expert list in the *SIMOTION Runtime Basic Functions functional description*.

Output cam configuration

In the **Configuration** window, define the configuration data values for the output cam.

Double-clicking in the project navigator below the output cam on the **Configuration** element displays the window in the working area.

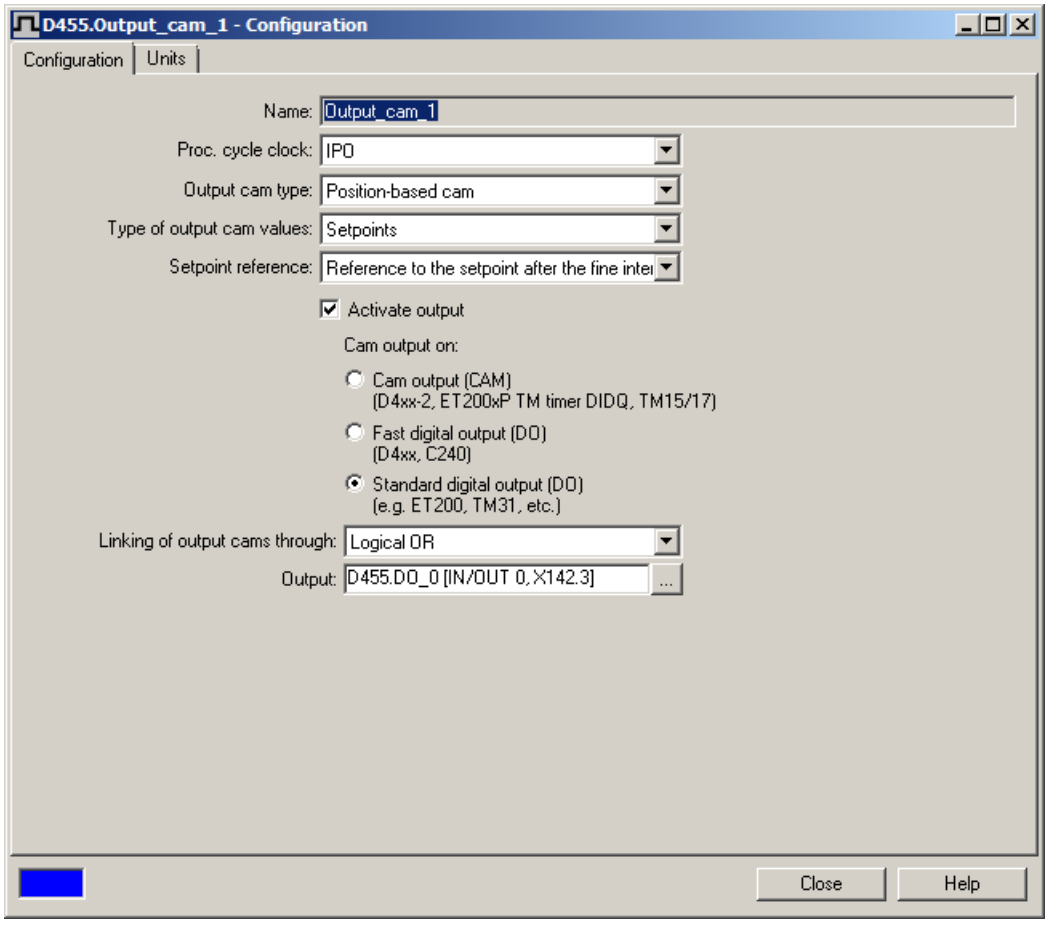


Figure 4-200 Output cam configuration using the example of a position-based cam

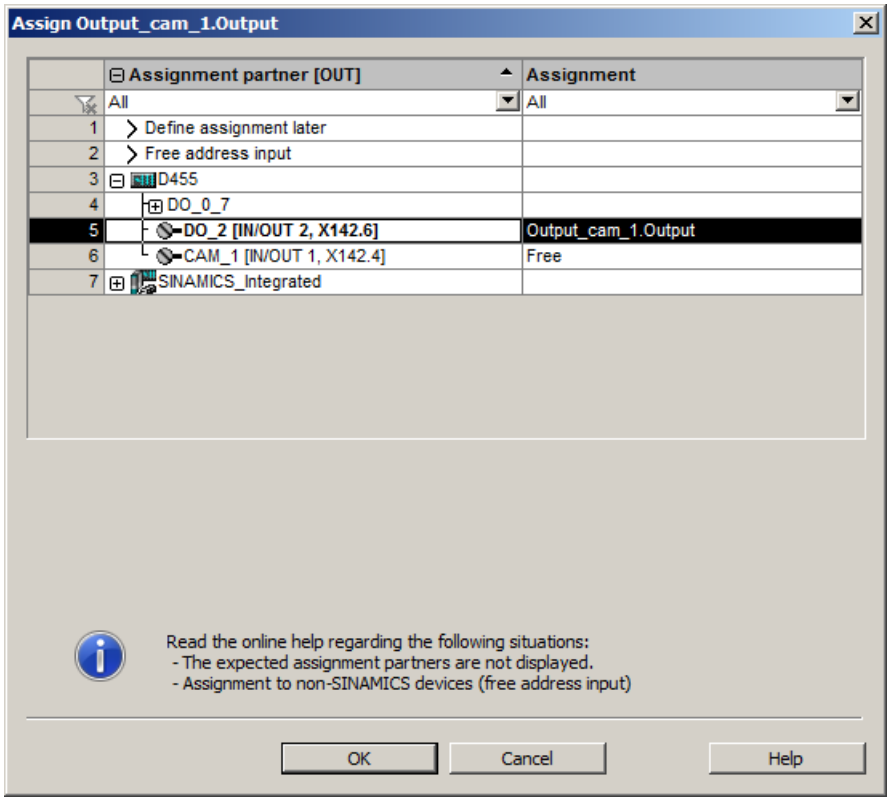


Figure 4-201 Assignment dialog

You can set the following parameters:

Table 4-157 Output cam configuration data

Field/button	Meaning/information
Name	The name of the created output cam is displayed here.
Output cam type	Choose **Output cam type** to select the type of output cam. **Position-based cam** (default value)The switching signal is active when the position of the axis lies between two markers (start and end position). **Time-based cam** The switching signal is on for a specific period of time after reaching the switching position (start position). **Uni-directional output cam** The switching signal changes when the axis reaches the switching position (start position). The output cam remains switched on even if the start position is overtraveled several times. The output cam must be explicitly reset.

Field/button	Meaning/information
Processing cycle clock	Choose **Processing cycle clock** to select the system cycle clock used to update the output cam signal at the output or in the system variables. The cam calculations are performed in IPO or IPO_2 cycles or in servo cycles. The processing cycle clock is set in the configuration by means of the **OcaBaseCfg.taskLevel** configuration data element. **IPO** (default value) The output cam signal is updated in the interpolator cycle clock. **IPO_2** The output cam signal is updated in the interpolator cycle clock 2. The IPO_2 cycle clock length is at least twice that of the IPO. **Servo** The output cam signal is updated in servo cycles. **The following configurations of the processing cycle clock are possible:** • Axis in IPO cycles and output cam in IPO_2 cycles • Output cam in servo cycles and axis in IPO or IPO_2 cycles For the possible setting of IPO_fast and Servo_fast with D435-2, D445-2 and D455-2, see Section Second servo cycle clock (Servo_fast) in the SIMOTION Runtime Basic Functions Manual. **Note:** • It is not possible to configure the axis in the servo cycle clock and the output cam in the IPO or IPO_2 cycle clock. • It is not possible to configure the axis in the IPO cycle clock and the output cam in the IPO_2 cycle clock if it is a setpoint output cam. • It is not possible to configure the axis in the IPO_2 cycle clock and the output cam in the IPO cycle clock.
Type of output cam value	Select the position value that is the reference for the output cam during processing.
Actual value reference	Select the following settings here depending on the type of the output cam values: • **Reference to the actual value on the encoder without considering Ti**. Reference to the actual position in the controller before the position filter. • **Reference to the actual value after the position filter**. Reference to the actual position in the controller after the position filter. • **Reference to the actual value on the encoder. Ti is taken into account**. Reference to the actual value on the encoder module / drive. The transmission time Ti from the encoder module / drive to the controller is taken into account by the system.
Setpoint reference	Select the following settings here depending on the type of the output cam values: • **Reference to the setpoint before the fine interpolator**. The cam switch points are calculated as if the setpoints will already have been reached at the end of this IPO cycle, e.g. if a setpoint is calculated for the IPO cycle clock with the virtual axis and also displayed at the end of the cycle. Reference is made directly to the setpoint applicable at the end of the cycle. • **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoint calculated in the interpolator will be output completely in the following cycle and the calculated position setpoint will therefore be reached at the end of the following cycle. • **Reference to the setpoint on the drive**. Calculation of the cam switch points according to the setpoint output with the current settings on the drive.

Field/button		Meaning/information
Activate output		Activate the checkbox if the output cam signal is to be applied to a digital output. Parameters are displayed.
Cam output on		
	Cam output (CAM)	If the output checkbox is activated and the cam output (CAM) radio button selected, the cam output is on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. In the case of D4x5-2 and the TM17 High Feature, the resolution is, for example, 1 µs. Supported hardware: • SIMOTION D410-2 • SIMOTION D4x5-2 (X142) • TM15, TM17 High Feature • ET200xP TM Timer DI/DQ The I/O channel must be configured as **CAM**. For more details, see cam output types. (Page 1811) **Note** Cam output (CAM) or high-speed digital output (DO) are also known as high-speed, hardware-supported output cams.
	High-speed digital output (DO)	If the output checkbox is activated and the "High-speed digital output (DO)" radio button selected, the output cam is output via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle clock. The time that it takes for the axis to reach the output cam switching position with reference to the processing cycle is calculated by linear extrapolation. Calculated from the beginning of the 1st position control cycle, the output cam function is triggered by a hardware time when this time is reached. Supported hardware: The onboard I/O of the following CPUs is used: • SIMOTION D4x5 (interface X122, X132), 8 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**) • SIMOTION D410 (interface X121), 4 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**) • SIMOTION C240, C240 PN (interface X1), 8 high-speed cam outputs For more details, see cam output types. (Page 1811) **Note** Cam output (CAM) or high-speed digital output (DO) are also known as high-speed, hardware-supported output cams.
	Standard digital output (DO)	If the output checkbox is activated and the "Standard digital output (DO)" radio button selected, the output cam is output in the servo cycle clock. The temporal resolution of the cam output is usually reduced by the output cycle of the I/O used. Supported hardware: • Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx) • Centralized I/O (SIMOTION C) • Distributed I/O via PROFIBUS DP/PROFINET I/O (e.g. ET 200, ET200xP TM Timer DI/DQ) • Drive I/O TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30 For more details, see cam output types. (Page 1811)

Field/button	Meaning/information
Logical operation (Page 1822)	You can assign several TO output cams to an output. Select the logical link of the output cam signal with the output. During the operation, all output cam signals are first grouped together with the logical operation OR. The result of this operation is then combined with the output cam signals to which a logical AND was assigned.
Output	The output can be symbolically assigned via the assignment dialog (see Chapter Symbolic assignment (as of V4.2) in the SIMOTION Runtime Functions Manual) using the ... button in the Output field (symbolic assignment is activated by default in projects as of V4.2). If symbolic assignment is not active or if the CPU version is < V4.2, a physical output is assigned by entering the HW address and bit number in the Output field. Enter the logical HW address of the output to which the output cam signal is to be applied. Only the output cam signal may be present at this address. If other objects are already using this output, an error occurs that is reported following a download to the target system. The logical HW address must be located outside the process image and therefore be greater than 63. For more details, see cam output types. (Page 1811)
...	Button for opening the assignment dialog (see Section Symbolic assignment (as of V4.2) in the SIMOTION Runtime Basic Functions Manual). Select a parameter or an address in the Assign dialog.
	Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display

Defining output cam defaults

You can define the defaults for every output cam. These values are stored in system variables and can be changed by programs.

Double-clicking in the project navigator below the output cam on the **Defaults** element displays the window in the working area.

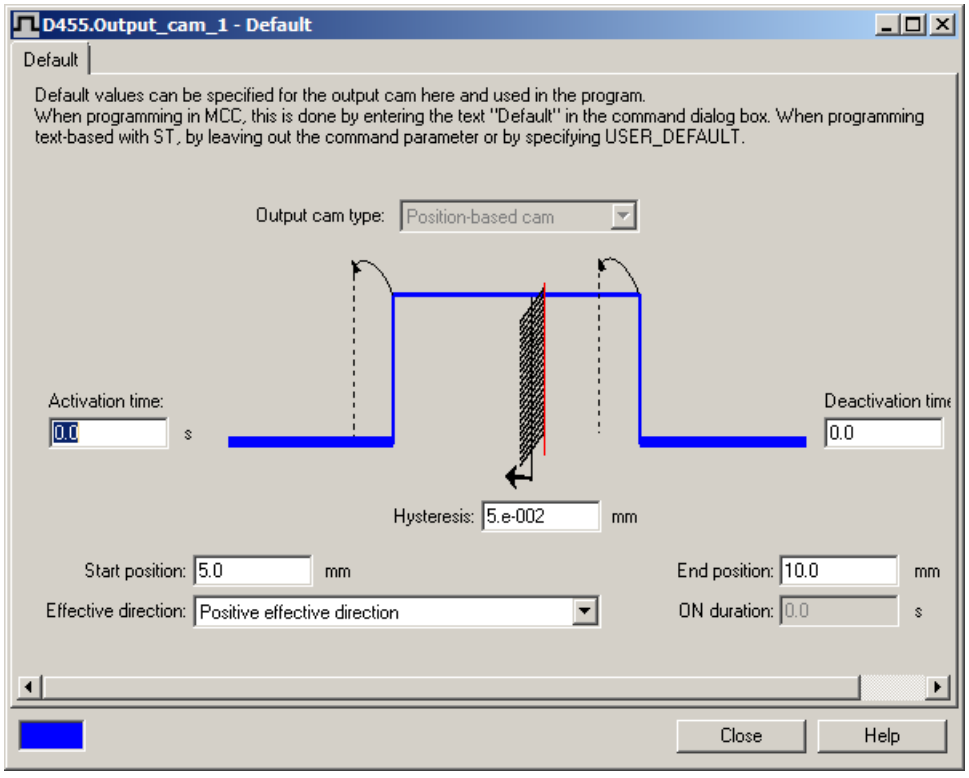


Figure 4-202 Output cam defaults, position-based cam example

You can set the following parameters:

Table 4-158 Defining output cam defaults

Field/Button	Significance/Note
Output cam type	**Output cam type** displays the type of output cam selected in the Configuration window.
Activation time See also the **Actuation times (activation/deactivation time)** section.	Enter the **activation time** here. The output cam switching time is set to the point when the start position is reached, plus this period. The output cam position is adapted dynamically. This allows you to compensate for propagation delays. If a negative value is entered as an activation time, the switching signal is activated before the start position is reached.
Using deactivation time	Activate the checkbox if you want to use a deactivation time when working with time-based cams. If this checkbox is deactivated, you cannot enter a time. In this respect, the time-based cam is compatible with older software versions (<V3.2).
Deactivation time	Enter the **deactivation time** here. The output cam switch-off time is set to the point when the end position is reached, plus this period. The output cam position is adapted dynamically. This allows you to compensate for propagation delays. If a negative value is entered as a deactivation time, the switching signal is activated before the end position is reached.
Hysteresis	Enter a range for the **hysteresis** here. The output cam does not change its switching state in this defined range around the switching position even under changed switching conditions. This prevents a repeated change of the switching state.
Start position See also the **Output cam types** section.	Enter the start position of the output cam. For path-controlled output cams this is the left switching position.

Field/Button	Significance/Note
End position	Enter the end position of the output cam. For path-controlled output cams this is the right switching position.
Effective direction	Enter the **effective direction** for the output cam. The output cam is active only if the current direction of motion of the axis corresponds to the parameterized effective direction. **Positive and negative effective direction (both)** Output cam switches in both directions of motion **Last programmed effective direction (effective)** Output cam switches only in the last programmed effective direction **Negative effective direction (negative)** Output cam switches only for negative direction of motion **Positive effective direction (positive)** Output cam switches only for positive direction of motion
ON duration	Enter the ON duration for time-controlled output cams here. After the axis has passed the switch-on position, the time-based cam output remains on for the ON duration.

See also

Derivative-action times (activation/deactivation time) (Page 1820)

Reaction, effective direction (Page 1817)

Hysteresis (Page 1818)

Determining derivative-action times for output cams (dead time compensation)

Depending on the system and the device, there is a certain time between the setting of a cam output by the program and the actual reaction of the actuator (e.g. solenoid valve). This time is called dead time and depends, for example, on the load-dependent delay times of a digital output, the switching properties of a valve, etc. Usually the exact value for the dead time is not known and can therefore be determined empirically through measurements.

In order that an output cam switches at the correct time, the dead time must be compensated by specifying a derivative-action time, which offsets the cam output by the dead time. Whereby it must be taken into account that the derivative-action times for switching an actuator on and off are usually different.

The empirical determination of the dead times using a difference measurement as an example.

Note

The procedure applies not only to output cams, but also to cam tracks. However, with cam tracks you can only specify a derivative-action time for the entire cam track.

Example

Lines of glue are to be applied to a product at a defined position and with a fixed length. The glue output is controlled by an output cam or a cam track. The glue is output from the start of output cam (switch-on point) to the end of output cam (switch-off point). The offset of the begin and end of output cam with respect to the velocity can be observed on the length and position of the glue line on the product (see figure). The figure below shows the line of glue for two velocities (v_1, v_2) with $v_2 > v_1$.

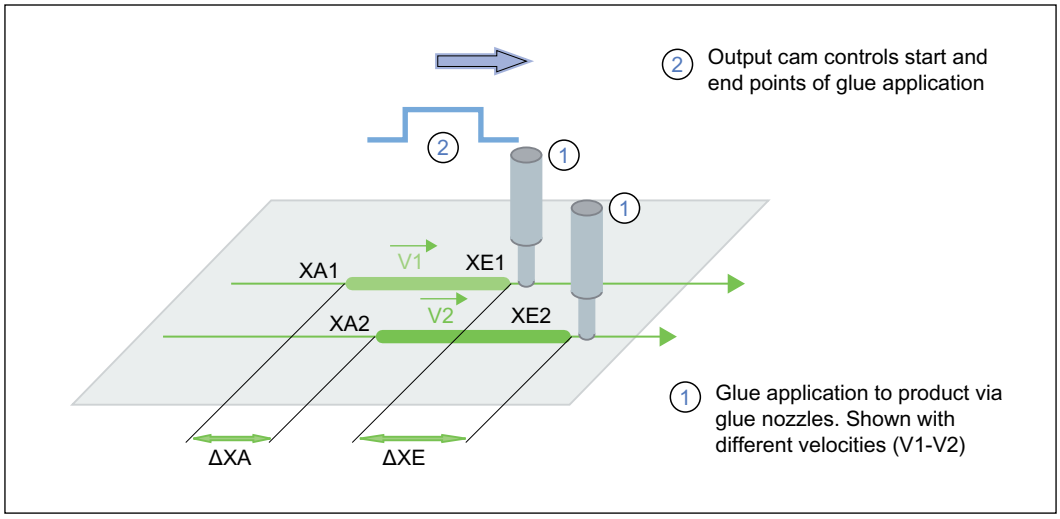


Figure 4-203 Offset of the output of output cam through dead times (dead time compensation)

Procedure:

1. Set all actuation times for start of output cam (activation time) and end of output cam (deactivation time) to 0.
2. Define the velocities for which the positions are to be determined. You should select two velocities that correspond to velocities that occur during production (e.g. minimum and maximum velocity).
3. Start the application and determine the start positions (x_{A1} and x_{A2}) and end positions (x_{E1} and x_{E2}) of the line of glue for the velocities v_1 and v_2.

   Note

   To increase the accuracy, you can perform several comparison measurements and use the average measured values.

4. You can determine the actuation times for the output of output cam using the following formula.
   $\mathbf{t_{activation} = \Delta s/\Delta v = (x_{A2}-x_{A1})/(v_2-v_1)}$
   $\mathbf{t_{deactivation} = \Delta s/\Delta v = (x_{E2}-x_{E1})/(v_2-v_1)}$

5. Enter the calculated actuation times as **activationtime** for the start of output cam and as **deactivationtime** for the end of output cam. Note that the actuation time must be entered as a negative when the output time is to be before the programmed output cam switching time.
6. After you have determined the activation time and the deactivation time for the output of output cam, you should perform a control measurement and check the result.

Note

Depending on the application, it may be, e.g. with eccentric presses, that there is no linear relationship between dead time and velocity (e.g. non-linear response of an applied brake). You have to dynamically adapt the dead time to the respective velocity for these applications. This can be implemented in the application with a user program. After the actuation time has been changed, you have to activate the output cam again with **_enableOutputCam** or the cam track with **_enableCamTrack**.

See also

Derivative-action times (activation/deactivation time) (Page 1820)

Configuring cams on SIMOTION D4xx onboard

Output cams and cam tracks can be configured for standard outputs, or as high-speed, hardware-based output cams / cam tracks.

Cams can be configured on SIMOTION D4xx onboard as follows:

1. In the project navigator, switch to the Control Unit via **SINAMICS_Integrated > Control_Unit.**
2. Double-click **Inputs/outputs** below the control unit. The window appears on the workspace.
3. Switch to the **Bidirectional digital inputs/outputs** tab.
4. Click the **button** to switch between the input and output for the digital inputs/outputs (**DO8 to DO15**). In each case, switch the DI/DO to the output you wish to use as the output of output cam. The designation at the terminal strip of DI or DO switches to DO. Outputs of the output cam can only be used if they have been defined as an output. DO 8 is configured as an output in the diagram. For the output, select the **DO (SIMOTION)** setting.

Note

Mixed use of the SIMOTION D4xx DI/O as high-speed outputs (of output cams) and inputs of measuring inputs is possible.

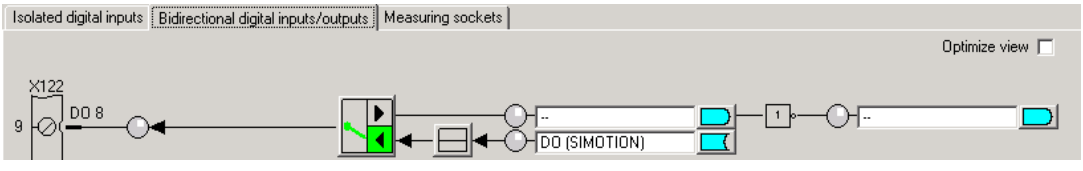


Figure 4-204 SIMOTION D4xx digital inputs/outputs

5. Click **Close**.
6. Insert a new output cam or a new cam track or use an existing one.
7. Parameterize the TO Output Cam / Cam Track
8. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
9. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. Should you wish to configure a high-speed output cam, select the **Activate output** check box and select the **High-speed digital output (DO)** radio button.
10. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
11. Click **OK** to close the window and select **Project > Save**.

To determine the logical hardware address for outputs on SIMOTION D4xx onboard (only if symbolic assignment is not activated)

1. In the project navigator, below the SIMOTION D device, select **SINAMICS_Integrated > Communication > Telegram configuration**.
2. Double-click Configuration and, in the window which appears, select the tab **IF1: PROFIdrive PZD telegram**. The components are displayed there with address range (input/output data).

3. Select SIEMENS telegram 390, 391 or 392 as telegram type. A maximum of eight output cams can be configured for each telegram. The number of DI/DO is limited to eight, i.e. only two output cams can be configured for telegram 392 if you are already using six measuring inputs. Therefore consider whether you also want to use measuring inputs during the telegram selection.

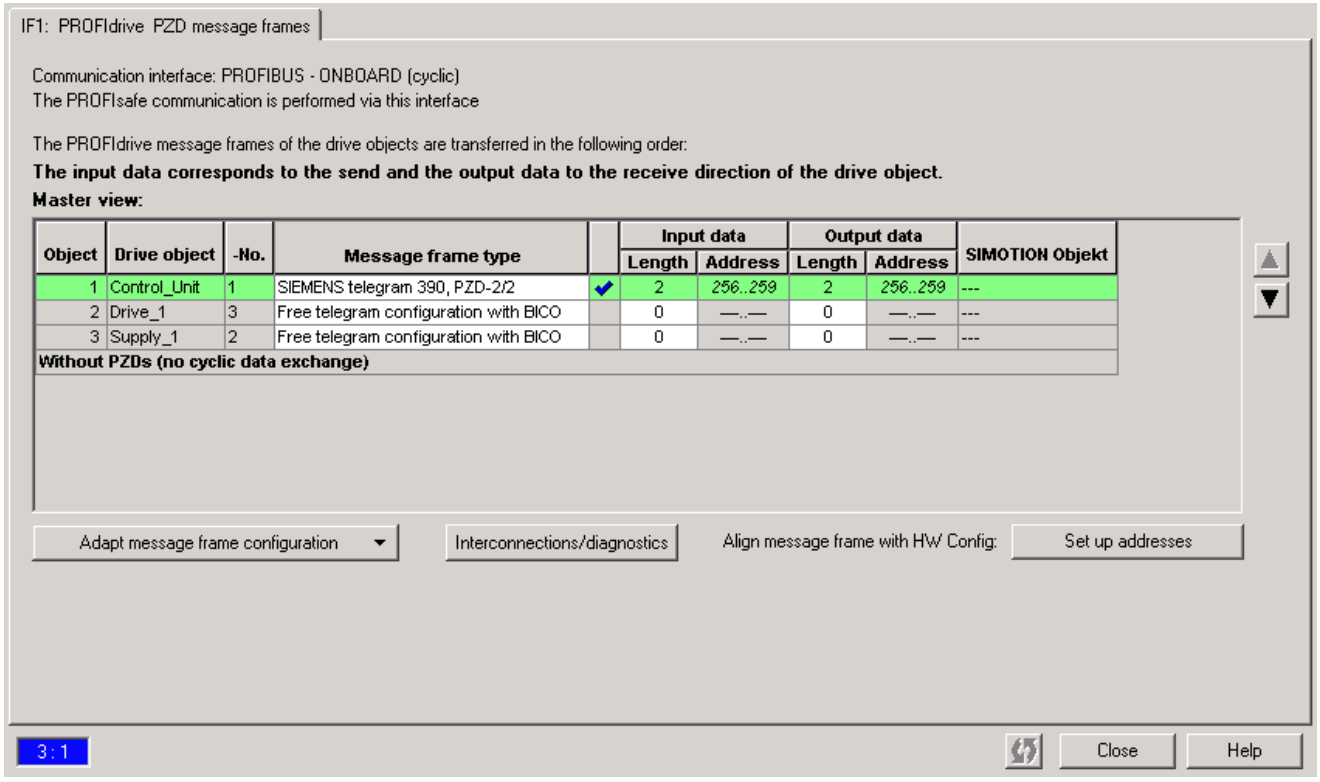


Figure 4-205 Determining the hardware address of the components

4. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If there are question marks in the fields instead of I/O addresses, you must also perform an alignment.
5. Now calculate the hardware address by adding the base output address (first value of the output data) of the Control Unit to the offset (for example 298 + 3 = 301). The offset always has the value **3**. Enter this calculated address under Measuring input > Configuration > Input (e.g. PI 301.1)."]
6. The onboard measuring inputs in the expert list of the Control Unit must be set in parameters 680[0] to 680[5] (e.g. 680[0] -> [1] DI/DO 9 (X122.8/X121.8))." You will find the bit number in the following table. The inputs are set in parameters 680.0 to 680.7 of the Control Unit, e.g. bit 1 in parameter 680[0].

Table 4-159 Bit numbers for D410 and D4x5

Output D4x5	Output D410	Bit number
X122.7 (DI/DO 8)	X121.7 (DI/DO 8)	Bit 0
X122.8 (DI/DO 9)	X121.8 (DI/DO 9)	Bit 1
X122.10 (DI/DO 10)	X121.10 (DI/DO 10)	Bit 2

Output D4x5	Output D410	Bit number
X122.11 (DI/DO 11)	X121.11 (DI/DO 11)	Bit 3
X132.7 (DI/DO 12)	-	Bit 4
X132.8 (DI/DO 13)	-	Bit 5
X132.10 (DI/DO 14)	-	Bit 6
X132.11 (DI/DO 15)	-	Bit 7

Note

In the case of versions earlier than V4.2, when using 39x telegrams, the onboard D4x5 outputs are to be assigned exclusively to SIMOTION. During a consistency check in SIMOTION SCOUT, no check is made as to whether the entered HW address actually belongs to a **high-speed digital output (DO)**.

See also

Insertion of Output Cam (Page 1825)

Parameterize Output Cam technology object (Page 1825)

Cam output types (Page 1811)

Configuring output cams on SIMOTION D410-2

1. In the project navigator, switch to the **Control Unit** via **SINAMICS_Integrated > Control_Unit**.
2. Double-click **Inputs/outputs** below the control unit. The window appears on the workspace.
3. Switch to the **Bidirectional digital inputs/outputs** tab.
4. Click the button to switch between the input and output for the digital inputs/outputs (**DO 8 to DO15**). In each case, switch the DI/DO to the output you wish to use as the output of output cam. The designation at the terminal strip switches to DO. Cam outputs must always be defined as outputs so that they can be used. DO 8 is configured as an output in the diagram. For the output, select the **Output cam (SIMOTION)** setting.

   Note

   Mixed use of the SIMOTION D410-2 DI/DO as high-speed (output cam) outputs and inputs of measuring inputs is possible.

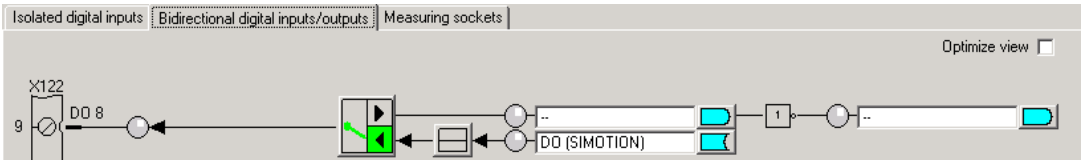


Figure 4-206 Digital inputs/outputs onboard

5. Click Close.

6. Insert a new output cam or a new cam track or use an existing one.
7. Parameterize the TO Output Cam / Cam Track
8. Double-click Configuration below the output cam or the cam track in the project navigator. The Configuration window appears in the working area.
9. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. Should you wish to configure a high-speed output cam, select the **Activate output** checkbox and select the **Output cam output (CAM)** radio button.
10. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
11. Click OK to close the window and select **Project > Save** from the menu.

Configuring cams on SIMOTION D4x5-2 onboard

With SIMOTION D4x5-2 the outputs on the interface X142 are used for cam output

1. The **Inputs/outputs X142** entry in the project navigator can be used to open the configuration screen in HW Config.
2. For the selected I/O channel, select **Output cam** as the function.

Note

If you do not use symbolic assignments (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual), you must note the logical address. (see Figure I/O Properties) This address must be configured at the TO output cam

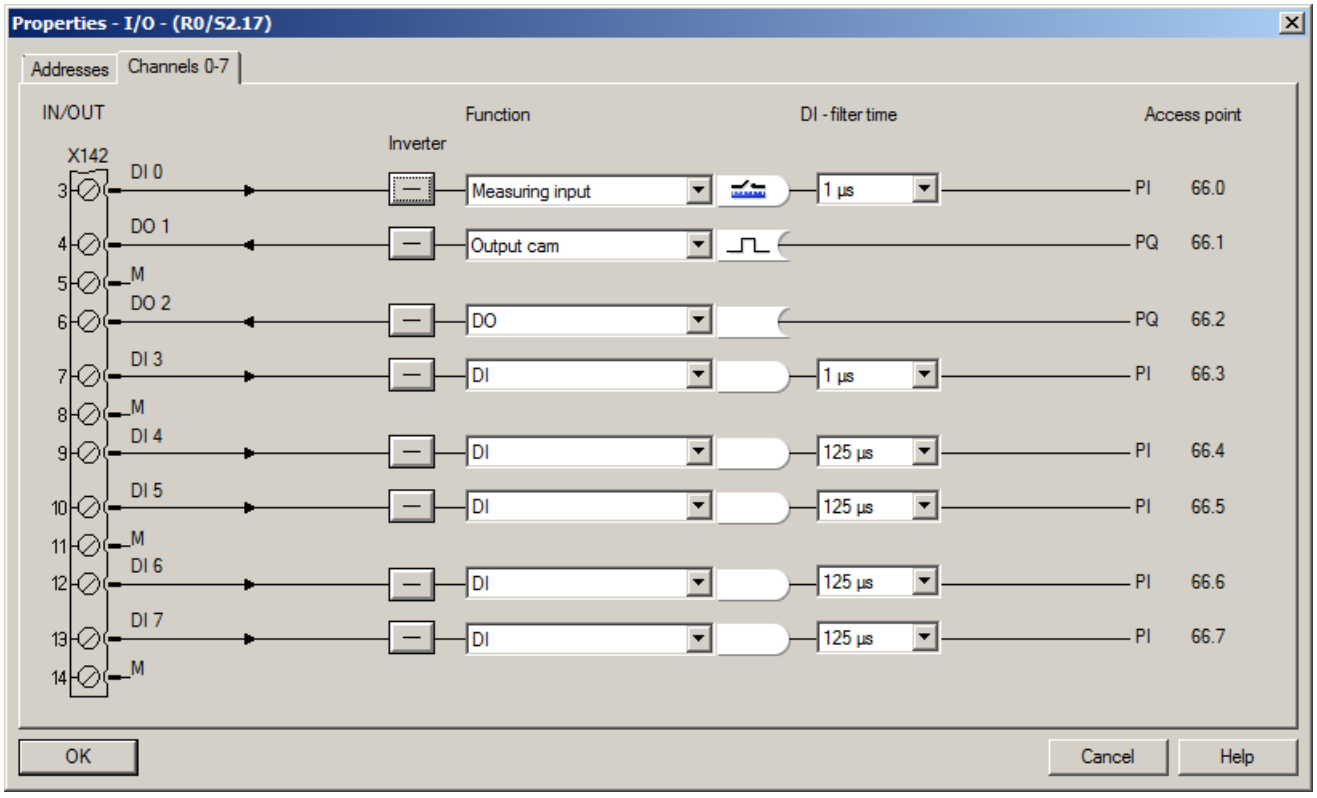


Figure 4-207 I/O Properties

3. Click OK.
4. Insert a new output cam or a new cam track or use an existing one.
5. Parameterize the TO Output Cam / Cam Track
6. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
7. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used.
   If you would like to configure a high-speed output cam, select the Activate output check box and select the output cam output (CAM) radio button.

8. Assignment of an output to an output cam/cam track is supported as of V4.2 using symbolic assignment or by entering the HW address.
9. Click OK to close the window and select **Project > Save** from the menu.

Configuring an output cam on a TM15/TM17 High Feature

1. In the project navigator, below the input/output component (TM15/TM17) that you want to use, double-click the entry **Inputs/outputs.** The **Bidirectional Digital Inputs/Outputs** window is displayed.
2. For the selected I/O channel, select **Output cam** as the function.

   Note

   If you do not use symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual), you must note the offset (e.g. 3.1).

3. Insert a new output cam or a new cam track or use an existing one.
4. Parameterize the TO Output Cam / Cam Track
5. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
6. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used.
   If you would like to configure a high-speed output cam, select the **Activate output** check box and select the **Cam output (CAM)** radio button.
7. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
8. Click OK to close the window and select **Project > Save** from the menu.

To determine the logical hardware address for TM15/TM17 High Feature outputs (only if symbolic assignment is not activated)

1. In the project navigator, below the SIMOTION device or the SINAMICS drive unit
   - for SIMOTION D, select: **SINAMICS_Integrated > Communication > Telegram configuration**
   - for SINAMICS S/G drive unit (position axis only): **Communication > Telegram configuration**
2. Double-click **Telegram configuration** and, in the window that opens, select tab **IF1: PROFIdrive PZD telegram**. The components are displayed there with the address ranges (e.g. TM17 output data 304 to 315).

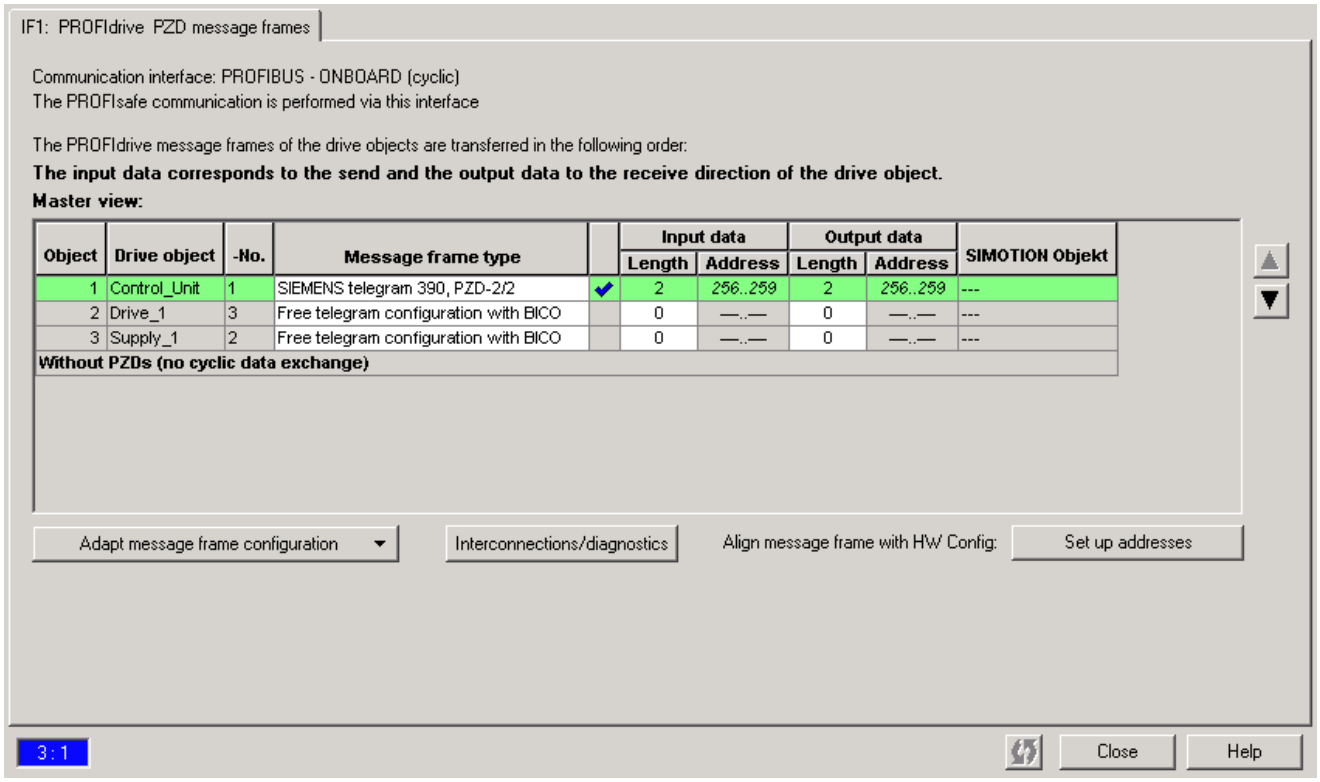


Figure 4-208 Determining the hardware address of the components

3. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If question marks are entered in the fields instead of I/O addresses, either alignment has not yet taken place, or the address is not recognized by SIMOTION SCOUT. In this case, you must perform an alignment.
4. Now calculate the HW address by adding the base output address (first value of the address range) of the TM to the offset (e.g. 304 + 3 = 307).
5. The bit number is defined by means of the offset. For example, a 3.1 offset of an output cam on DO 1 results in a bit number of 1.

Configuring cams on SIMATIC ET200SP and ET200MP

Overview

Description

You can configure TO output cam on the ET200SP technology module TM Timer DIDQ 10x24V and ET200MP technology module TM Timer DIDQ 16x24V.

- Configuration on the ET200SP technology module TM Timer DIDQ 10x24V is possible both using Step7 in connection with SIMOTION SCOUT Classic and with SIMOTION SCOUT TIA in connection with the TIA Portal. Further information can be found in the section Configuring output cams to TM Timer DIDQ 10x24V (Page 1843).
- Configuration on the ET200MP technology module TM Timer DIDQ 16x24V is only possible using SIMOTION SCOUT TIA in connection with the TIA Portal. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA.

Further information

Further information on the functions of the TM Timer DIDQ technology modules can be found in the SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V and SIMATIC S7-1500/ET 200MP Technology Module TM Timer DIDQ 16x24V.

Configuring output cams on TM Timer, using DIDQ 10x24V as example

Description

You can configure TO output cams on the ET200SP technology module TM Timer DIDQ 10x24V in SIMOTION SCOUT or SIMOTION SCOUT TIA in connection with the TIA Portal. The configuration in the TIA Portal is described in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA. The configuration in SIMOTION SCOUT in connection with Step7 is described below:

In order to parameterize the ET200SP TM Timer DIDQ 10x24V with Step7 in connection with SIMOTION SCOUT Classic, a Hardware Support Package (HSP) for Step7 is required. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA in the section on hardware and software requirements.

For the TM Timer DIDQ technology module, the digital outputs are used for the cam output. Irrespective of whether you work with SIMATIC STEP 7 or SIMOTION SCOUT, the parameterization is always performed in HW Config:

Parameterization of TM Timer DIDQ in HW Config

1. Open the HW configuration in SIMOTION SCOUT or SIMATIC STEP 7.
2. Select the header module (IM) with the TM Timer DIDQ in HW Config.
   The TM Timer DIDQ appears in the detail view.
3. To open the context menu, select the TM Timer DIDQ with the right mouse key

4. Select object properties.
   You can view or change the following properties in the object properties:
   - General
   - Addresses
   - Identification
   - Parameters
5. Open the tab **Parameter** > **DQ0 / DI0**
6. Set the parameter value **Configuration DQ/DI group** to **Use input/output individually**.

   Note

   With this setting, the DQ0 channels can be interconnected as cam output CAM or DI0 as measuring input MI under SIMOTION SCOUT.

7. Press **OK** to close the dialog.
8. Open the menu **Station** > **Save and Compile**.

Note

The technology modules and the individual channels may only be reparameterized in offline state. All changes only take effect after saving, compiling and subsequent download of your changed user project to the controller. The download can be performed in HW Config via **Target system** > **Download to module** or in SCOUT via **Load CPU / drive unit to target device**.

Parameterization of CAM in SCOUT

1. Insert a new output cam or a new cam track or use an existing one.
2. Parameterize the TO Output Cam/Cam Track.
3. Double-click Configuration below the output cam or the cam track in the project navigator. The Configuration window appears in the working area.
4. Activate the selection box **Activate output** and select the option **Cam output (CAM).**

5. Assignment of an output to an output cam/cam track is supported as of V4.2 using symbolic assignment or by entering the HW address.

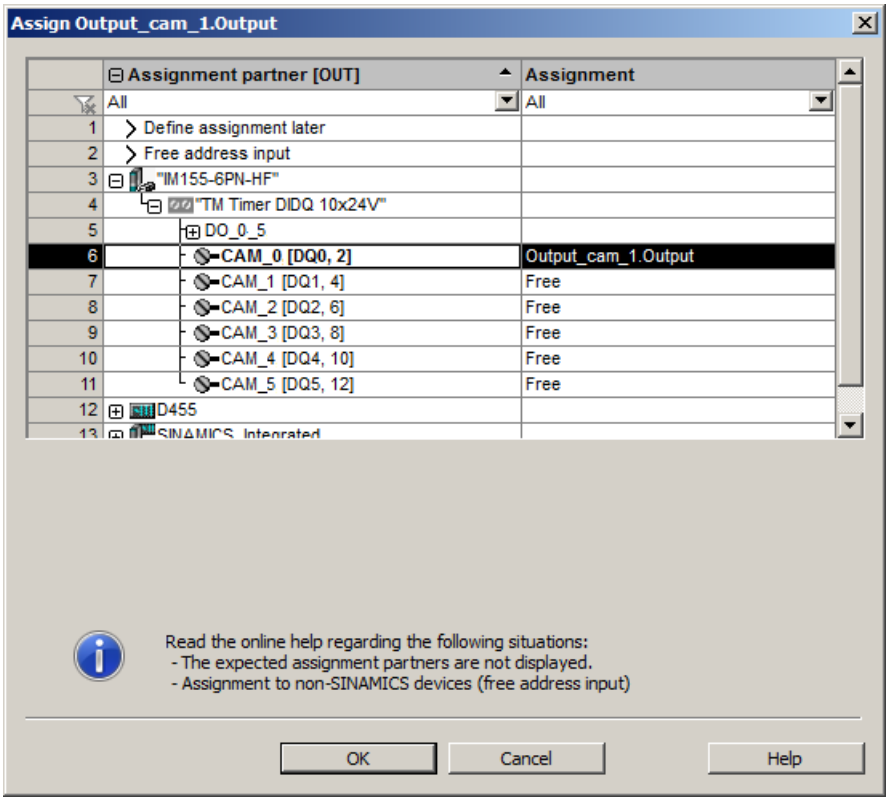


Figure 4-209 Assigning a cam output by means of symbolic assignment

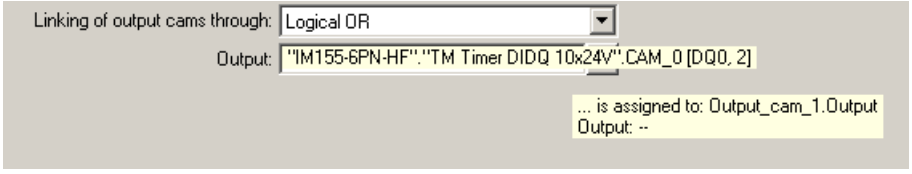


Figure 4-210 Assigning a cam output by entering the HW address

6. Click OK to close the window and select **Project** > **Save**.

Configuring cams on SIMOTION C240

1. Insert a new output cam or a new cam track or use an existing one.
2. Parameterize the TO Output Cam / Cam Track
3. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
4. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. If you would like to configure a high-speed output cam, select the **Activate output** check box and select the **High-speed digital output (DO)** radio button

5. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address. During a consistency check in SIMOTION SCOUT, no check is made as to whether the entered HW address actually belongs to a **high-speed digital output (DO)**.
6. Click OK to close the window and select **Project > Save** from the menu.

HW enable for Output Cam TO

You can make the output of output cams dependent on a hardware-supported enable (only for TM17 High Feature).

Because hardware enables are mainly used for output cam tracks, this function is described under Cam Track TO (Page 1909).

4.3.3.4 Programming/references of Output Cam TO

Programming

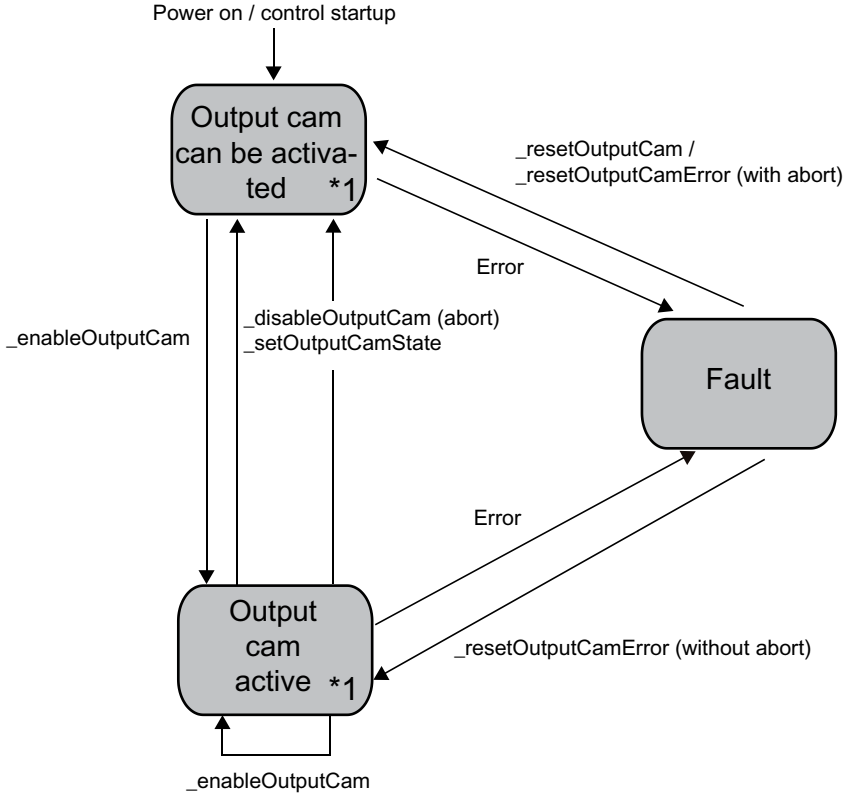


Figure 4-211 Programming and execution model for Output Cam technology object

***1** The following commands are effective in the TO states **can be activated** and **active**:

- **_disableOutputCamSimulation**
- **_enableOutputCamSimulation**

The simulation commands are modal / act in parallel and do not replace any existing _enableOutputCam commands.

Commands

The Output Cam technology object can be addressed in the user program using the following commands:

Table 4-160 Output Cam TO system functions

Commands	Description	Application
_enableOutputCam	Activate output cam	Output cam analysis is activated. If the switching condition for the output cam is fulfilled, the output or **state** system variable is set.
_disableOutputCam	Deactivate output cam	Output cam analysis is deactivated. If the switching condition for the output cam is fulfilled, the output or **state** system variable is not set. Controlled output cams are reset immediately.
_enableOutputCamSimulation	Activate simulation mode. This function simulates an output cam by disconnecting the output.	Values are calculated, but not forwarded to the hardware. Hardware output cams act as software cams. The output cam remains internally active, the status is retained, the output of output cam is not switched. If an active output cam is switched to simulation mode, the output cam status remains the same, and only the control of the output is reset or interrupted.
_disableOutputCamSimulation	The output cam is reset from simulation mode.	The output of output cam is switched according to the output cam status and the signal inversion.
_setOutputCamState	Deactivate the output cam function and set the output cam status to the specified value.	This is used if the output should not be controlled by the output cam TO. **Example:** A glue nozzle is controlled by the output cam TO (applying glue dots). As a service function, it should also be possible to rinse the nozzle while constantly controlling it. This is achieved via **_setOutputCamState**.
_resetOutputCamError	Reset error on output cam TO.	E.g. acknowledge configuration errors after entering correct values.
_setOutputCamCounter	Change starting count for a counter cam.	Output cam is output on every nth switching operation.
_resetOutputCam	This function sets the output cam to an initial state. Pending errors are deleted. Modified configuration data is reset on request.	Create initial state of output cam TO.

Commands	Description	Application
_resetOutputCamConfigDataBuffer	This function deletes the configuration data collected in the buffer since the last activation without activating it.	Changing configuration data in the RUN state discards the accumulated modifications.
_getStateOfOutputCamCommand (V3.2 and higher)	This function returns the execution state of a command.	Check whether the output cam switching has already taken place, i.e. whether the command ID is still available or has already been deleted.
_bufferOutputCamCommandId (V3.2 and higher)	This function enables **commandId** and the associated command status to be saved beyond the execution period of the command. The **commandId** parameter is used to define the command for which the respective status is to be saved. The maximum number of saveable command status is specified in the **decodingConfig.numberOfMaxBufferedCommandId** configuration data element.	Subsequent check of how command was terminated, e.g. error-free or number of error that occurred.
_removeBufferedOutputCamCommandId (V3.2 and higher)	This function ends saving of **commandId** and the associated command status beyond the execution period of the command.	Explicit deletion of previously saved command IDs.

For further information on the system functions, please refer to the *SIMOTION TP CAM Reference Lists*.

Process Alarms

You can predefine local alarm responses via SIMOTION SCOUT.

Note

For more information, refer to the *Motion Control Technology Objects Basic Functions* functional description.

How to configure the alarm response:

1. Double-click **Execution system** in the project navigator below the SIMOTION device. The execution system opens.
2. In the execution level tree, select **SystemInterruptTasks > TechnologicalFaultTask**.
3. Then click the **Alarm Response** button in the displayed window. The **Alarm Response** window appears. You can configure the alarm response for every TO here.

A system variable **error** indicates that a technology alarm has been generated. The response to the alarm is displayed in the **errorReaction** variable.

Table 4-161 Possible alarm responses

Alarm Response	Description	Application
NONE	No response	-
DECODE_STOP	Command processing is aborted, the output cam function remains active. Execution on the technology object can continue after **_resetOutputCam** or **_resetOutputCamError**.	The Output Cam TO can only be reactivated after the error has been acknowledged.
OUTPUTCAM_DISABLE	Command processing is aborted, current output cam function is aborted. Execution on the technology object can continue after **_resetOutputCam** or **_resetOutputCamError**.	The Output Cam TO can only be reactivated after the error has been acknowledged.

Output Cam TO menus

Output cam menu

Grayed-out menu functions cannot be selected. The menu is only active if an output cam window is active in the working area.

You can select the following functions:

Table 4-162 Output cam TO menu

Function		Significance/Note
Close		Select **Close** to close the configuration window for the output cam that is open in the working area.
Characteristics		Select **Properties** to display the properties of the output cam highlighted in the project navigator.
Configuration		Select **Configuration** to determine the configuration data (for example output cam type) of the output cam.
Default		Select **Default** to define the default settings of the system variables (e.g. effective direction) for the output cam.
Expert		
	Expert list	Select **Expert list** to open the expert list for the highlighted output cam. The configuration data and system variables can be displayed and changed in this list.
	Configure units	Select **Configure units** to open the **Configure units of the object** window in the working area. You can configure the units used for the selected object here.

Output cam context menu

Grayed-out functions in the context menu cannot be selected.

You can select the following functions:

Table 4-163 Output cam context menu

Function		Significance/Note
Open configuration		Select **Open configuration** to display the window for configuring the output cam in the working area. Enter the configuration data (for example, output cam type) for the output cam in this window.
Expert list		Select **Expert list** to open the expert list for the highlighted output cam. The configuration data and system variables can be displayed and changed in this list.
Cut		Select **Cut** to remove the selected object and save it to the clipboard.
Copy		Select **Copy** to copy the selected object. It is stored in the clipboard.
Paste		Select **Paste** to insert the output cam stored in the clipboard.
Delete		Select **Delete** to delete the highlighted output cam. The entire data of the output cam is deleted permanently.
Rename		Use **Rename** to rename the object selected in the project navigator. Note that with name changes, name references to this object are not adapted.
Expert		
	Insert script folder	**Insert script folder** enables you to insert a folder below the TO. You can create scripts in this folder in order to, for example, automate the configuration.
	Import object	**Import object** imports the data of a SIMOTION object from another project which was previously created with a selective XML export. You cannot import the entire project, only the data of the SIMOTION object.
	Save project and export object	**Save project and export object** exports selected data of the selected object in XML format. This data export can then be reimported into other projects. Only the data of the selected object, not the entire project, is exported.
Print		Select **Print** to print the configuration of the output cam. All system variables and configuration data with the associated values are printed.
Print preview		Select **Print preview** to open the preview of the output cam data to be printed.
Default		Select **Default** to define the default setting of the system variables (e.g. effective direction) of the output cam.
Properties		Select **Properties** to display the properties of the output cam highlighted in the project navigator.

4.3.4 Cam Track TO - Part II

4.3.4.1 Overview of TO Cam Track

General information about Cam Track TO

Cam tracks allow several output cams to be output as a track on one output.

The **Cam Track** technology object

- Generates position-dependent switching signals
- Can be assigned to positioning axes, synchronous axes or external encoders
- The axes can be real or virtual.

Different switching signals distinguish different types of output cam on the cam track:

- **Software cam**
  Switching signals are used **internally** in the user program by evaluating the relevant **state** system variable.
- **Hardware cams**
  Switching signals are output **externally** on I/O by assigning a digital output to the cam track TO. For example, digital output modules from the ET 200 I/O system can be used for cam track output.

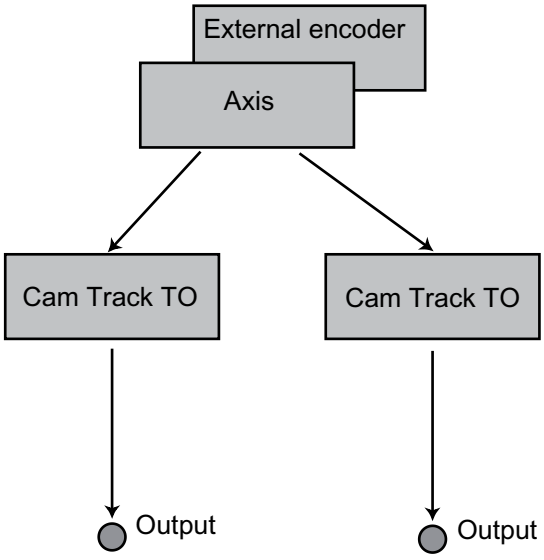


Figure 4-212 Interconnection options for the cam track TO

A range of output cam types with different switching behaviors are available on a cam track:

- **Position-based cam**
  The switching signal is supplied between the switch-on position and the switch-off position.
- **Time-based cam**
  The switching signal is supplied for a specified time period after the switch-on position is reached.
- **Time-based cam with maximum ON length**
  A maximum ON length can be defined for time-based cams. This means that the time-based cam is deactivated once it has covered the maximum length, even though the parameterized time has not yet expired.
- **High-speed/accurate output cam (hardware-based output cam)**
  Although output cams are usually output in IPO cycles or the servo cycles, high-speed output cams achieve output accuracy that is better than the servo cycle clock because the switching edges are output within servo cycles.

Functionality

Cam track functionality

- Cam tracks allow up to 32 output cams to be configured within one TO and allow, for example, the switching point for all output cams to be shifted collectively.
- The switching of several output cams is dependent on the same setpoint/actual value, and they are output on one output.
- Any number of cam tracks can be used per axis. The only restriction placed on this number is the system performance.
- The cam track can be calculated in servo cycles, IPO cycles, or IPO_2 cycles.
- All output cams on one track are of the same type (position or time-based cams).
- Cam tracks can be activated once or cyclically.
- Various modes are available for activating and deactivating cam tracks, e.g. **active immediately**, **next track cycle**, etc.
- Cam track output can be inverted.
- The status of each single output cam (controlled/not controlled) can be read over one array of byte.
- Single output cams on a cam track can also be defined as valid/invalid.
- In connection with the TM17 High Feature terminal module, the cam track output can be controlled via a high-speed hardware enabling signal.

Reference to axis

The reference values of the cam track depend on the axis type or the external encoder:

Table 4-164 Reference to the actual or set position

Technology object	Reference to actual position possible	Reference to set position possible
Real drive axis	-	-
Real position axis	X	X
Real synchronized axis	X	X
Virtual axes	-	X
External encoder	X	-

Possible configuration of the cam tracks with actual value reference:

- **Reference to the actual value on the encoder without considering Ti**. Reference to the actual position in the controller before the position filter.
- **Reference to the actual value after the position filter**. Reference to the actual position in the controller after the position filter.
- **Reference to the actual value on the encoder. Ti is taken into account**. Reference to the actual value on the encoder module / drive. The transmission time Ti from the encoder module / drive to the controller is taken into account by the system.

Possible configuration of the cam tracks with setpoint reference:

- **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoints will already have been reached at the end of this IPO cycle, e.g. if a setpoint is calculated for the IPO cycle clock with the virtual axis and also displayed at the end of the cycle. Reference is made directly to the setpoint applicable at the end of the cycle.
- **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoint calculated in the interpolator will be output completely in the following cycle and the calculated position setpoint will therefore be reached at the end of the following cycle.
- **Reference to the setpoint on the drive**. Calculation of the cam switch points according to the setpoint output with the current settings on the drive.

In this case, the cam track functionality can be applied to axes or external encoders with or without modulo properties.

The cam track is also effective for axes that have not been homed.

The cam track is defined independently of the axis. The cam track is mapped on the axis via an axis reference position, once the track is activated. This enables cam tracks to be operated in a particularly flexible way (e.g. relative output of a cam track on the basis of a measured edge on the measuring input, cam track offset, etc.).

Output on one output

The cam track TO is assigned to one output only during configuration. Output can be achieved via:

- Onboard I/O
- Drive I/O (for example TB30, TM31, TM1x)
- SIMOTION C centralized I/O
- Distributed I/O; PROFIBUS DP I/O (e.g. ET 200M)

However, the output must not be in the process image.

The switching accuracy is dependent on the following:

- Output accuracy of the I/O
- How the cam track is allocated in the task system
- How constant delay times are compensated

Comparison of Output Cam TO and Cam Track TO

Depending on the application, it is practical to use either the Cam Track TO or one or more Output Cam TOs. The table below should help you to decide which TO should be used in which case.

Table 4-165 Comparison of Output Cam TO and Cam Track TO

Features	Output Cam TO	Cam Track TO
Availability	• As of Version 1.0	• As of Version V3.2
Supported output cams	• Position-based cam • Time-based output cam • Unidirectional output cam • Counter cam • Exact time setting of an output, exact time output cams (as of V4.1)	• Position-based cam • Time-based output cam • Time-based cam with maximum ON length
Several output cams on one output	• Via logical operation (AND/OR)	• Maximum 32 output cams of the same type in one track • No cam track logical operations (AND/OR)
Different types of output cam on one output	• Via AND/OR	• Not available
Output cam definition	• Related to axis • Via system variables	• Related to cam track (cam track can be mapped as required on axis) • Via system-variables array
Hysteresis	• Available	• Available
Effective direction	• Available	• Not available
Derivative-action times	• Separate for power ON/power OFF	• Separate for power ON/power OFF
Deactivation time for time-based cam	• As of Version V3.2	• As of Version V3.2
Activation/deactivation types	• Active immediately	• Start and stop mode parameterizable
Types of output	• Cyclic	• Cyclic • Once
Output cam status	• System variable	• Status of single output cams over one array of byte
Output cam enable	• Via_enableOutputCam	• via_enableCamTrack • Validity of single output cams configurable via system variables
Performance	• Depends on number of single output cams	• When 5 or more output cams are used in one output cam track instead of 5 single output cams, the output cam track performs better. This performance advantage amounts to at least a factor of 2 for 32 single output cams.
MCC command available	• Available	• Available (V4.0 and higher)

4.3.4.2 TO Cam Track basics

Cam track features

A cam track has parameters that are valid for the track as a whole, and parameters that can be configured for each single output cam on a track.

Track data

Track data is valid for all output cams on a track and is, therefore, configured for the cam track as a whole.

- **Output cam type**
  Position-based, time-based, etc.
- **Cam track start**
  Always defined from "0"
- **Track length**
  Cam track start to cam track end
- **Hysteresis**
  Even if the switching conditions change, the output cam does not change its switching state in this defined range around the switching position.
- **Actuation times**
  Actuation times can be specified to compensate for the switching times of digital outputs and connected switching elements.
- **Axis reference position**
  Cam tracks are defined independently of the axis. The axis reference position is used to define how the cam track is mapped on the axis, or from which axis position the cam track should be output.
- **Cyclic or non-cyclic activation mode**
  If the cam track is output non-cyclically, it will have to be reactivated after execution.
- **Start mode and stop mode**
  Start mode or stop mode can be used, for example, to define whether a cam track will be output immediately or not until the next track cycle.

Output cam data

Output cam data can be configured separately for each single output cam on a cam track.

- **Output cam parameters**:
  Depending on output cam type, start position, end position, ON duration, maximum ON length.
- **Validity of single output cam**
  Single output cams on a defined cam track can be parameterized as "invalid." This output cam is completely suppressed and is not output. It also has no status indication.

Example of a cam track definition

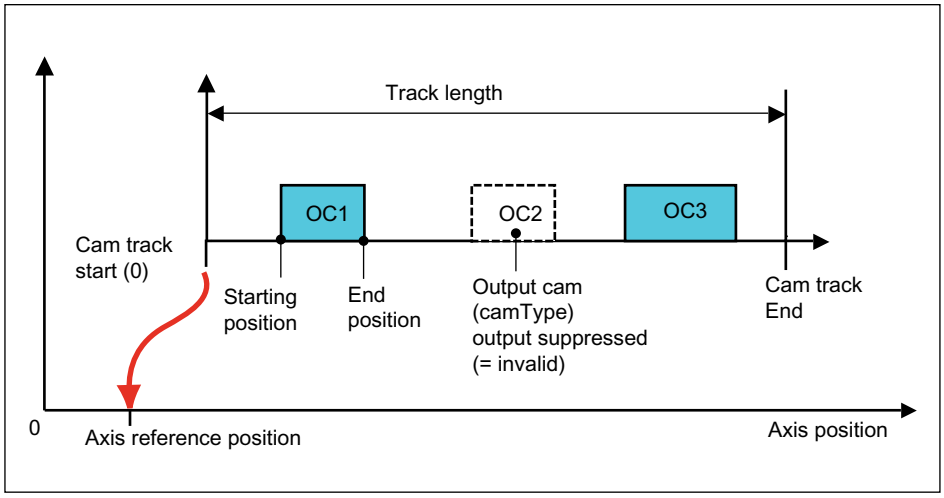


Figure 4-213 Definition of a cam track with 3 output cams

Output cam types of the single output cams on a track

The following chapter provides an overview of the output cam types within a cam track. All output cams on a cam track are always of the same output cam type.

Software cam (Page 1856)

Hardware cam (Page 1856)

Position-based cam (Page 1857)

Time-based cam (Page 1858)

Time-based cam with maximum ON length (Page 1859)

Cam output types (Page 1860)

Software cam

Switching signals are used **internally** in the user program by evaluating the relevant **state** system variable.

Hardware cam

Description

Switching signals are output **externally** on I/O devices by assigning a digital output to the Cam Track TO.

The following can be used as digital outputs:

- Onboard outputs (SIMOTION C, D, ...)
- Centralized I/O (SIMOTION C)

- Distributed I/O via PROFIBUS DP (e.g. ET 200M) and PROFINET IO (e.g. ET 200S, ET200xP TM Timer DI/DQ)
- Drive I/O (for example, TM15 and TM17 High Feature Terminal Modules)

Hardware for cam track

Cam output on cam output (I/O channel is configured as CAM)

- SIMOTION D410-2
- SIMOTION D4x5-2
- TM15, TM17 High Feature
- ET200xP TM Timer DI/DQ

Cam output on high-speed output with direct access (I/O channel is configured as DO)

- SIMOTION D4xx / D4x5-1
- SIMOTION C240, C240 PN

Cam output on standard output (I/O channel is configured as DO)

- SIMOTION C/D/CX onboard I/O
- SINAMICS onboard I/O
- TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30
- Standard DO (SIMATIC ET200, ...)
- PROFINET IO (e.g. ET200xP TM Timer DI/DQ)

Position-based cam

Switching behavior

Position-based cams on a cam track switch independently of the direction of motion, i.e. they always have a positive and negative effective direction.

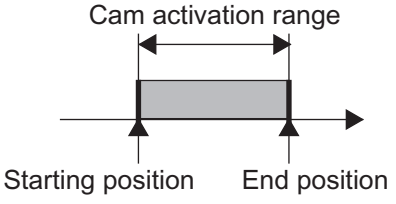


Figure 4-214 Position-controlled output cam with a starting position less than the end position

Limits imposed by starting and end positions

The cam is activated:

- if the axis position is within the activation range
- if the axis position value is shifted into the activation range of the output cam
  The position value of the interconnected object can change abruptly, for example, when it is homed or when its coordinate system is shifted with the **_redefinePosition** command.

The output cam is switched off:

- if the axis position is outside the starting or end position
- if the position value is shifted outside the activation range
- via commands that deactivate the output cam, e.g. **_disableCamTrack**, **_setCamTrackState**, **_resetCamTrack**

Cam activation range

The activation range of the output cam is defined from the start position to the end position in a positive direction of motion, i.e. in the range between the starting position and end position. If the end position is greater than the starting position, the activation range is defined by the starting and end positions (see figure above).

The activation range is outside the range between the end and starting positions if the end position is less than the starting position (see figure below).

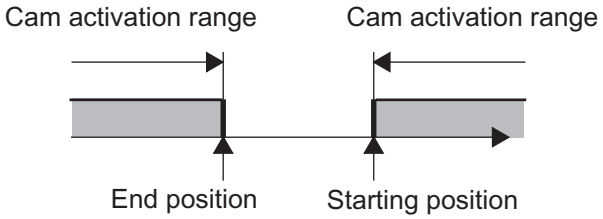


Figure 4-215 Position-controlled cam with an end position less than the starting position

Note

This definition of the activation range is possible for all modulo and non-modulo axes.

ON duration

The ON duration of the output cam depends on the velocity at which the axis traverses the output cam length.

Time-based output cam

Switching behavior

Time-based cams on a cam track switch independently of the direction of motion, i.e. they always have a positive and negative effective direction.

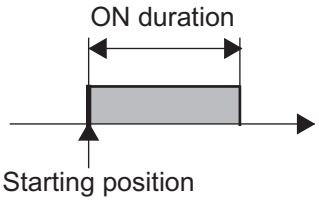


Figure 4-216 Time-controlled output cam

Limits imposed by starting position and ON duration

The output cam is switched on:

- At the starting position.
  If the starting position is overrun again during the ON duration, the time-based cam is not switched on again. It is not possible to retrigger a time-based cam.

The output cam is switched off:

- When the assigned time period expires
- When commands are issued that deactivate the output cam, e.g. **_disableOutputCam**, **_setOutputCamState** and **_resetOutputCam**

Output cam length

The output cam length is dependent on the velocity at which the assigned axis traverses during ON duration of the output cam.

Time-based cam with maximum ON length

Additional limits imposed by maximum ON length

A maximum ON length can also be defined for time-based cams on cam tracks. This means that the time-based cam is deactivated once it has covered the maximum length, even though the parameterized time has not yet expired.

This is the case if, for example, glue dots should be applied to a workpiece and the amount of glue should be independent (constant time -> time-based cam) of the throughput rate.

To avoid the time-based cam still being controlled after the end of the workpiece at high sweep rates, the ON duration can be limited by a maximum ON length (related to the start position of the output cam). This prevents a glue dot being placed adjacent to the workpiece.

The maximum ON length is effective in both traversing directions of the axis, and the cam track's switch-on position is the reference position.

Parameters of a time-based cam with maximum ON length

Every time-based cam on a track has three parameters

- Start of output cam (SOC)
- ON time (t)
- Maximum ON length (Lmax)
  This always relates to the dynamically adjusted start of output cam SOC, i.e. the assigned activation time **is** taken into account. The output cam is then traversed over the maximum ON length, **without** taking into account the deactivation time (see the **Actuation times (activation time/deactivation time** section).

Example of a cam track which controls glue application

In the following example, a cam track with three output cams is used to control the application of glue onto a workpiece. No glue may be applied outside of the predefined areas.

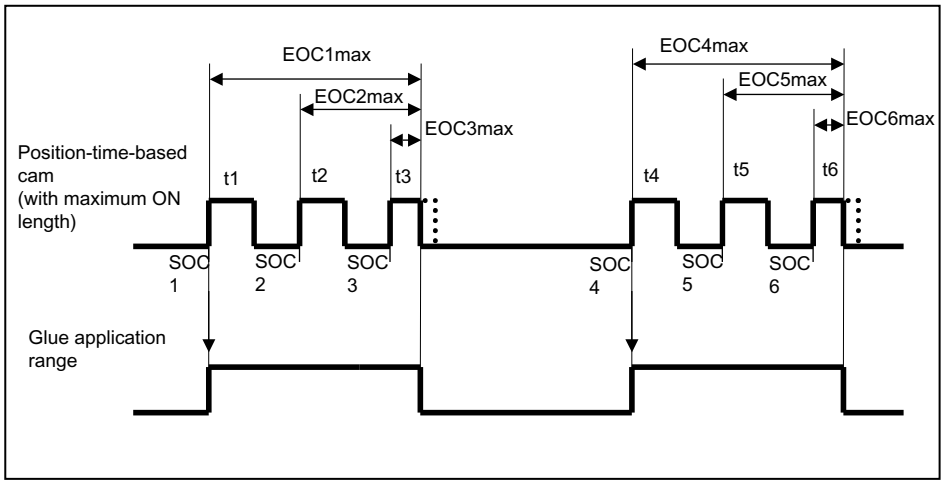


Figure 4-217 Control of glue application via a cam track, based on a time-based cam with maximum ON length

- The start of output cam (here, SOC1 and SOC4) is used to exactly define the start of glue application.
- The ON time (t) is used to ensure that the same amount of glue is applied, independent of the axis speed.
- The maximum ON length Lmax is used to ensure that no glue is applied outside of the defined area. In the example, the output cam ON durations t_3 and t_6 are limited by the maximum ON length.

Cam output types

The cam calculations are performed in the processing cycle clock (IPO or IPO_2 cycle clock or in the servo cycle clock). For the possible setting of Servo_fast or IPO_fast, see Section Second servo cycle clock (Servo_fast) in the SIMOTION Runtime Basic Functions manual.

The temporal resolution of the cam output depends on the hardware used and the setting in the configuration. In standard applications, the setting is undertaken using screen forms. The configuration data can also be set via the expert list.

The possible setting options for cam output are described below:

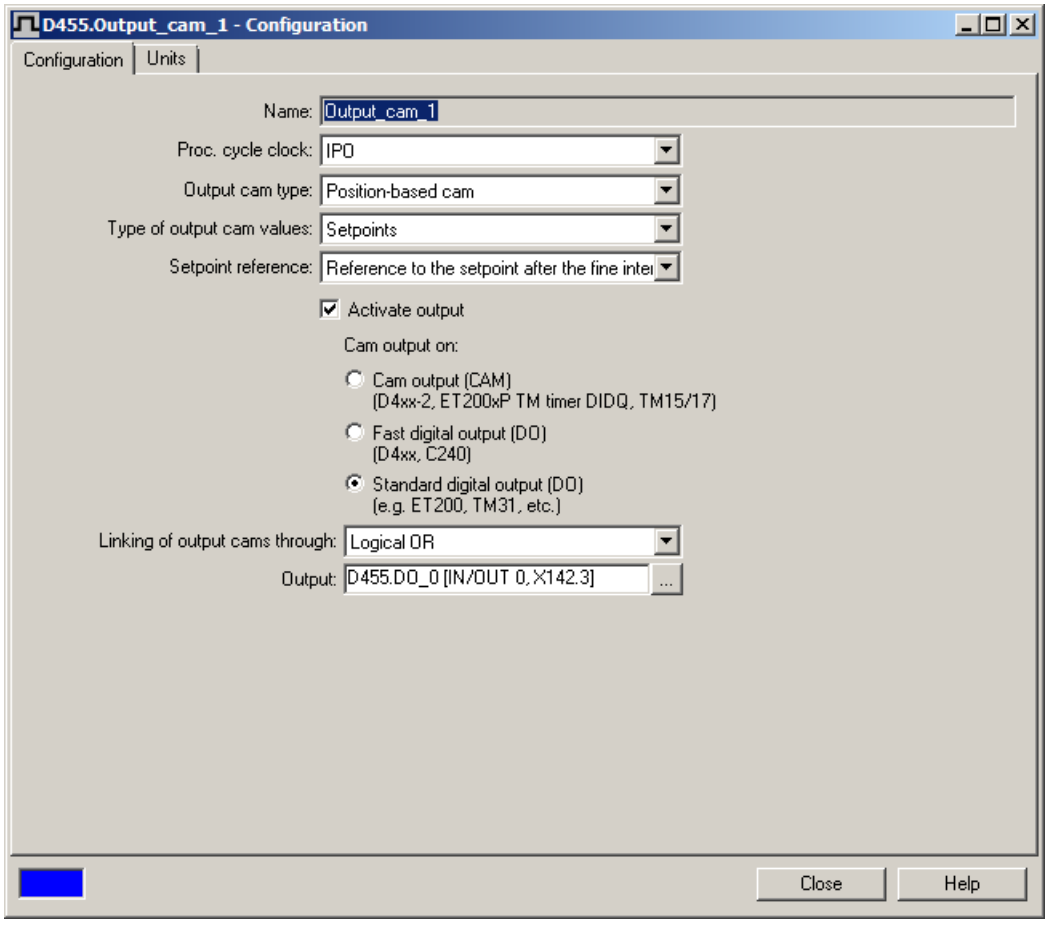


Figure 4-218 Output cam configuration using the example of a position-based cam

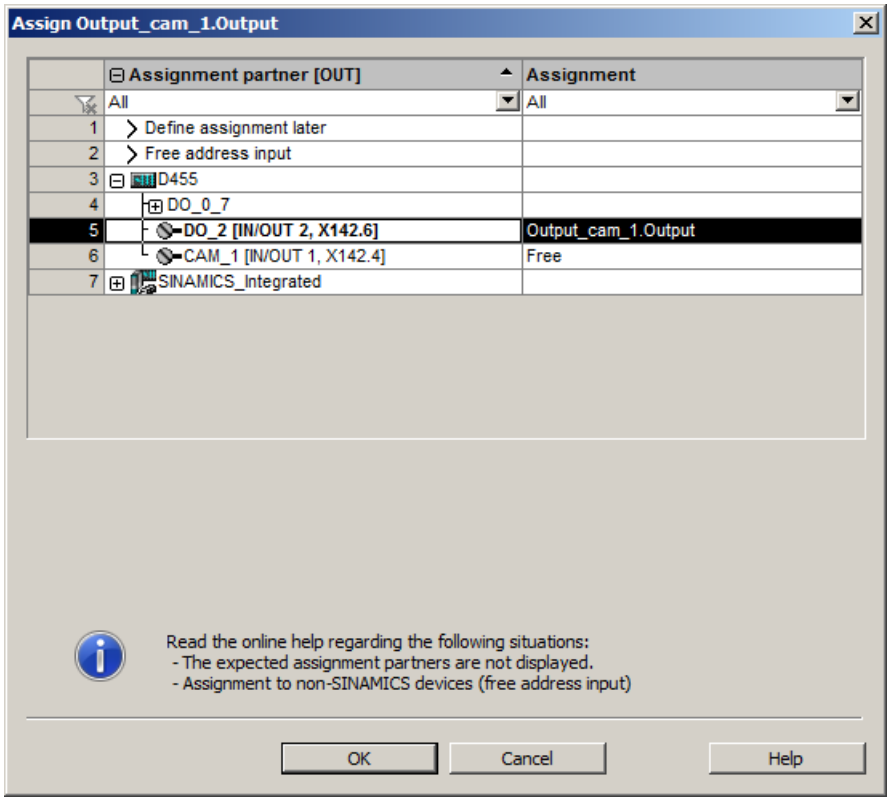


Figure 4-219 Assignment dialog

Symbolic assignment is activated by default in projects as of V4.2 (**Project > Use symbolic assignment**)

Cam output (CAM)

The cam output is performed on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. In the case of D4x5-2 and the TM17 High Feature, the resolution is 1 µs.

Hardware supported

- SIMOTION D410-2
- SIMOTION D4x5-2 (X142)
- TM15, TM17 High Feature
- ET200xP TM Timer DI/DQ

The I/O channel must be configured as CAM.

SIMOTION D410-2

The digital inputs/outputs are used for the cam output in the D410-2. The digital inputs/outputs can be used as the cam output (CAM) from the user program.

As of the editorial deadline of this documentation, the output cam resolution for D410-2 was not yet certain. The information can be found at the following website Internet link (http://support.automation.siemens.com/WW/view/en/27585482).

SIMOTION D4x5-2 onboard outputs (interface X142)

The D4x5-2 onboard outputs can be used as cam output (CAM) from the user program. The D4x5-2 onboard outputs are permanently assigned to SIMOTION. The X142 I/Os are configured using HW Config.

The X142 configuration screen form can be accessed directly from the project navigator in SIMOTION SCOUT.

With SIMOTION D4x5-2, output cams are output at the X142 interface with a resolution of 1 µs.

TM15 / TM17 High Feature Terminal Modules

The TM15 and TM17 High Feature Terminal Modules can be used to set up cam outputs (CAM) within the SIMOTION Motion Control system. The Terminal Modules are connected directly to SIMOTION D or CX32/CX32-2 via DRIVE-CLiQ for this purpose.

Alternatively, TM15 and TM17 High Feature can be connected to a SINAMICS S120 CU320/ CU320-2/CU310/CU310-2 Control Unit with higher-level SIMOTION C, P or D.

Output cams on the TM15 operate with DRIVE-CLiQ cycle-clock resolution (typically 125 µs). Output cams on the TM17 High Feature have a resolution of 1 µs.

If current controller cycle clocks other than 125 µs are used, the parameter calculations of the drive must be taken over into the PG and the Fast IO configuration must be recreated when using cam outputs on TM15/TM17 High Feature. (For more information, see chapter Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs in the TM15 / TM17 High Feature Terminal Modules Commissioning Manual).

ET200xP TM Timer DI/DQ

The digital inputs/outputs are used for the cam output in ET200xP TM Timer DI/DQ. The digital inputs/outputs can be used as the cam output (CAM) from the user program. Output cams on the ET200xP TM Timer DI/DQ have a resolution of 1 µs.

High-speed digital output (DO)

The cam output is performed via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle clock.

The time that it takes for the axis to reach the output cam switching position with reference to the processing cycle is calculated by linear extrapolation. Calculated from the beginning of the 1st position control cycle, the output cam function is triggered by a hardware time when this time is reached.

Hardware supported

The onboard I/O of the following CPUs is used:

- SIMOTION D4x5 (interface X122, X132), 8 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**)
- SIMOTION D410 (interface X121), 4 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as **DO**)
- SIMOTION C240, C240 PN (interface X1), 8 high-speed cam outputs

SIMOTION D410/D4x5 onboard outputs

Output cams are output via **a high-speed digital output (DO).**

- Up to and including SIMOTION V4.1 SP5, all D410/D4x5 onboard I/Os configured as digital outputs are exclusively available to SIMOTION
- As of SIMOTION V4.2, D410/D4x5 onboard I/Os configured as digital outputs can be switched over to SINAMICS using BICO interconnection (channel granular)

Standard digital output (DO)

The output cam calculations are performed in processing cycles (IPO or IPO_2 cycle clock or servo cycle clock).

Actual cam output is performed in servo cycles. The temporal resolution of the cam output is usually reduced by the output cycle of the I/O used.

Therefore the resolution

- with standard I/O (e.g. ET 200) depends on the cycle time of the bus system (PROFIBUS DP / PROFINET IO)
- with TM15 / TM17 depends on the cycle time of the bus system (PROFIBUS Integrated / PROFIBUS DP / PROFINET IO)
- with TM15 DI/DO, TM31, TM41, TB30 depends on the configured sampling time
  - cu.p0799 (CU inputs/outputs sampling time) for the TB30 and onboard outputs
  - p4099 (TMxx inputs/outputs sampling time) for TM15 DI/DO, TM31 and TM41

Hardware supported

- Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx)
- Centralized I/O (SIMOTION C)
- Distributed I/O via PROFIBUS DP/PROFINET I/O (e.g. ET 200, ET200xP TM Timer DI/DQ)
- Drive I/O TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30

Configuration data of cam output types in expert list

Table 4-166 Setting options for cam output

Selection in configuration screen	Setting in expert list
Cam output (CAM) (D4xx-2, ET200xP TM Timer DIDQ, TM15/17)	OcaBaseCfg.outputType = [1] TIME_STAMP OcaBaseCfg.hwTimer = [91] NO
High-speed digital output (DO) (D4xx, C240)	OcaBaseCfg.outputType = [0] STANDARD OcaBaseCfg.hwTimer = [173] YES
Standard digital output (DO) (Standard DO, e.g. ET200, TM31)	OcaBaseCfg.outputType = [0] STANDARD OcaBaseCfg.hwTimer = [91] NO

Output cams on **cam output (CAM)** or on **high-speed digital output (DO)** are also referred to below as high-speed, hardware-supported output cams.

Note

Further information and the output accuracy for high-speed output cams is described in the PM21 Catalog and in the respective product brief or commissioning/equipment manuals.

Commissioning Manual *Terminal Modules TM15/TM17 High Feature*
Operating Instructions *SIMOTION C2xx*
Commissioning Manual *SIMOTION D410*
Commissioning Manual *SIMOTION D410-2*
Commissioning and Hardware Installation Manual *SIMOTION D4x5*
Commissioning and Hardware Installation Manual *SIMOTION D4x5-2*

Commissioning Manual *Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA*

Cam track parameters

Track length

The **camTrackLength** system variable is used to parameterize the track length. The track length is calculated from the start of the cam track (always 0) to the end of the cam track. Usually, the output cams of the cam track are located within the track length. The track length must not be 0. When track length = 0, an error is reported when the cam track of a non-modulo axis is activated. If track length = 0 for a modulo axis during cam track activation, the cam track length is set to the axis modulo length.

Effective direction and behavior

The following diagram shows output cam behavior on switching on and off, without hysteresis, activation or deactivation time.

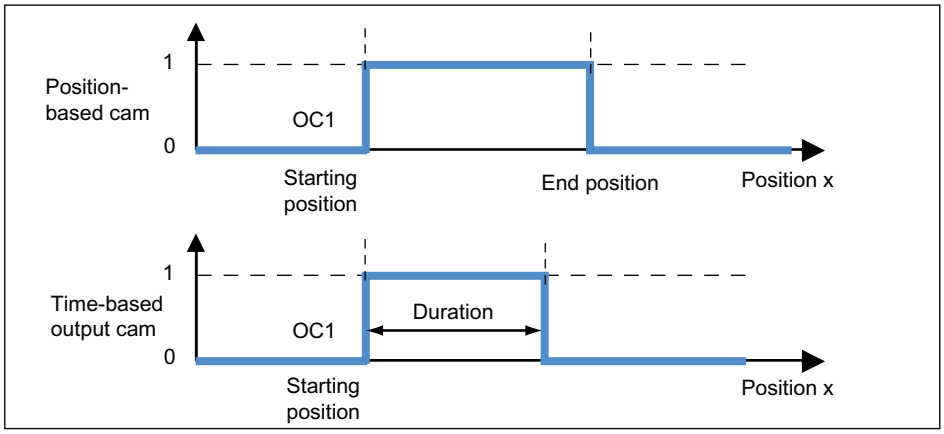


Figure 4-220 Output cam behavior on switching on/off

The switching characteristic depends only on the position (position setpoint or actual position). The cams on a cam track switch independently of the direction of motion, i.e. they always have

a positive and negative effective direction. A position-based cam can be output **repeatedly** on changing the direction of movement. Time-based cams are output **once only**.

Hysteresis

If the actual position value tends to fluctuate due to mechanical influences, specification of a **hysteresis** prevents the output cam from unintended switch status changes.

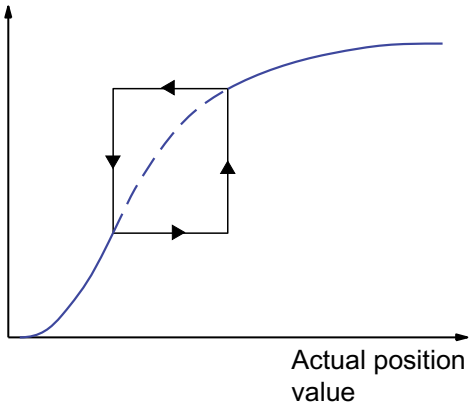


Figure 4-221 Hysteresis

Conditions for the hysteresis range

- Hysteresis is not activated until the direction has been reversed.
- The direction of motion is not redefined within the hysteresis.
- Within the hysteresis, the switching state of position-based cams is not changed.
- If modified switching conditions for the output cam are detected when the output cam is outside the hysteresis range, this current switching state is set.

Example: position-based cam hysteresis

Cam track configuration (only one output cam configured):
Output cam type: position-based cam; switch-on position, 20 mm; switch-off position, 200 mm; hysteresis, 20 mm; effective direction: both

Axis positions:
0 mm -> 100 mm -> 10 mm -> 50 mm -> 0 mm -> 150 mm -> 0 mm

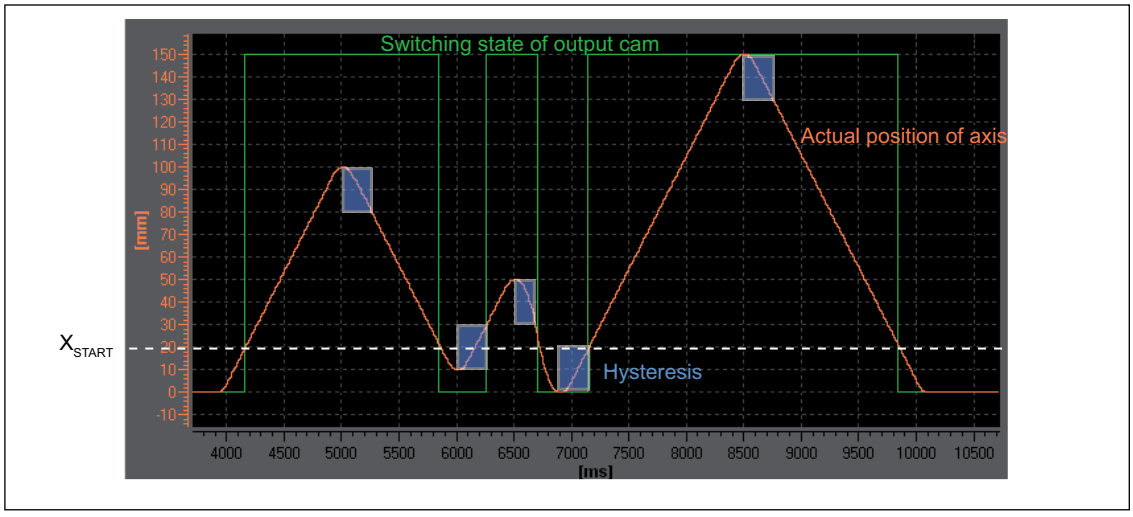


Figure 4-222 Hysteresis range (height of blue sections) and behavior of a cam track with a position-based cam, effective direction in both directions.

As the cam track switches in both directions, the output cam does not switch off after the first reversal of direction. The second switch-on point is moved to position 30, due to active hysteresis.

Example: time-based cam hysteresis

Cam track configuration (only one output cam configured):
Output cam type: time-based cam; switch-on position, 40 mm; ON duration, 0.5 s; hysteresis, 20 mm; effective direction: both

Axis positions:
0 mm -> 100 mm -> 20 mm -> 50 mm -> 30 mm -> 80 mm -> 10 mm -> 150 mm

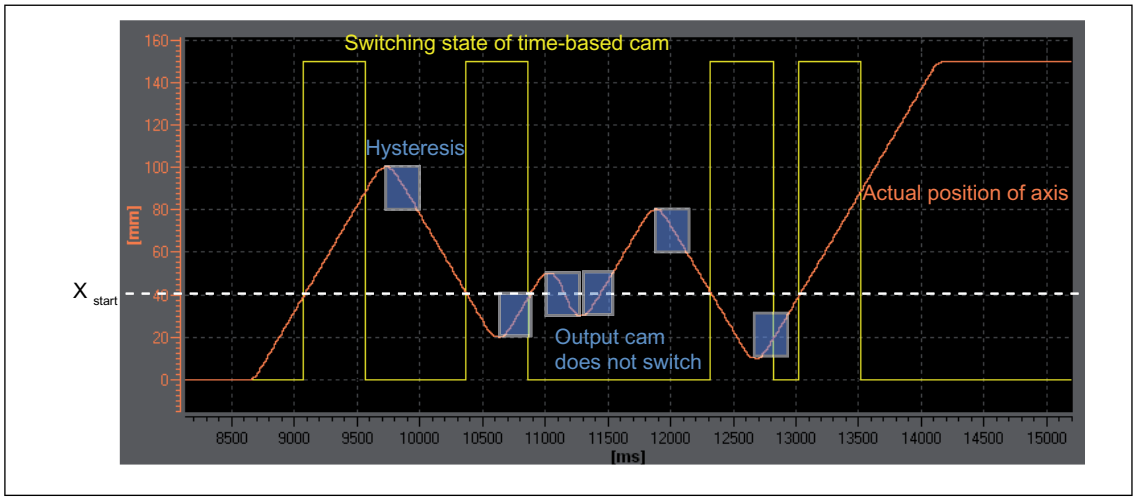


Figure 4-223 Hysteresis range (height of blue sections) and behavior of a cam track with one time-based cam, no effective direction.

Time-based cam switches off only after ON duration has expired, not after change of direction.

Time-based cams with a start position within the hysteresis range are not output (see figure above).

Hysteresis range

The upper limit of the hysteresis range is set at 25% of the working range for a linear axis, and 25 % of the rotary axis range for a rotary axis. If you violate this maximum setting, an error message is issued. In practice, a lower setting is used for the hysteresis range.

- **Path-controlled output cam**
  The hysteresis becomes active after direction reversal is detected. The output cam switches off once the hysteresis has been left and the position is located outside of a defined output cam.
- **Time-based cam**
  The switching behavior of a time-based cam is determined by the ON duration, not by the hysteresis. This means that an entered hysteresis range has no influence on the ON duration of an output cam. It only has an influence on the switch-on time (start position).
- **Time-based cam with maximum ON length**
  The maximum ON length switches off the output cam once the hysteresis has been left and the maximum ON length has been exceeded.

Note

If a time-based cam's start position lies within the hysteresis, it is **not** output.

Derivative-action times (activation time/deactivation time)

To compensate for the switching times of digital outputs and connected switching elements, or of propagation delays, it is possible to specify **actuation times**. Actuation times are calculated from the sum of all delay times and can be specified separately for activation and deactivation edges as an actuation time at the activation edge (activation time) or an actuation time at the deactivation edge (deactivation time).

The activation/deactivation times of the output cam are dynamically compensated by means of the derivative-action times. In this way, output cams are dynamically shifted depending on the actual velocity.

For example, a valve that should open at 200°, with an activation time of 0.5 s

- Must be controlled at 195° at a velocity of 10°.
- Must be controlled at 190° at a velocity of 20°.

This dynamic shift takes place automatically by means of the TO camTrack.

Settings for the activation and deactivation times can contain positive or negative values.

A negative activation time must be entered if the output cam is to be switched before the programmed start of the output cam.

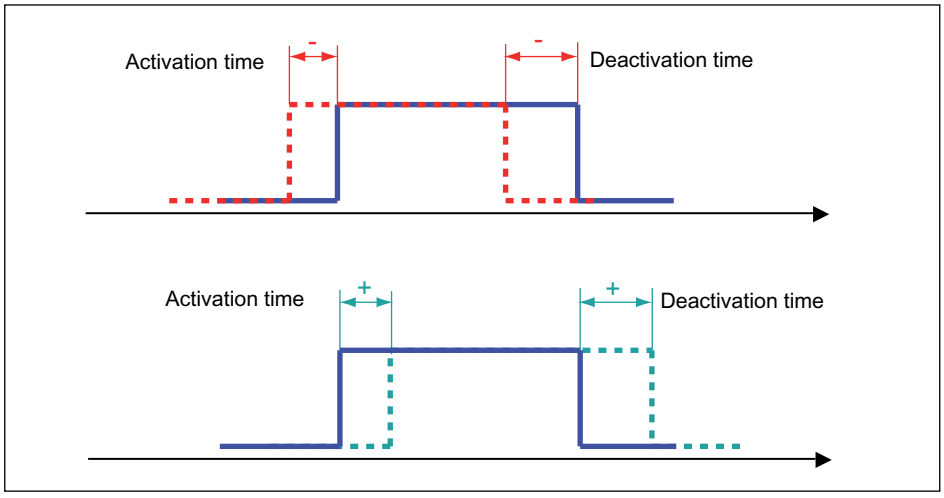


Figure 4-224 Switching behavior at varying actuation times

Note

The time of output for the output cam in the controller is relevant for calculation of the dynamic adjustment. If velocity changes up to signal output, these changes are no longer taken into account.

Dead times, e.g. PROFIBUS DP communication times, output delay times on digital outputs, etc., are taken into account in the actuation time.

Long actuation times exceeding one modulo cycle may lead to heavy fluctuation of the switching position of actual value output cams (actual value curve). Here, setpoint output cams should be used or the actuation time should be considerably less than one modulo cycle.

The system takes into account the specified actuation times when the output cams are calculated and managed. The switching positions of the output cams are calculated taking into account the activation time and deactivation time in relation to the present velocity. If, allowing for actuation times, the output cam was switched, then the system deems this operation to have occurred, and it does not switch the output cam again even if any subsequent current velocity changes occur.

The dynamic actuation of modulo axes can be greater than one modulo length. However, the number of switching operations is not collected by the system, i.e. for actuation times longer than one modulo length, a switching operation cannot take place in each modulo cycle. One switching operation is active in the system at any given point in time. A switching operation is completed when the output cam is switched off.

Actuation times and cycle clock settings

A change of cycle clock settings does not have to be taken into account for the actuation time settings (activation/deactivation time). These are, for example:

- Changing the Servo/IPO/IPO_2 cycle clock settings (for example, from "1/1/1 ms" to "2/2/2 ms").
- Change of processing cycle clock of the cam track TO (setting: Servo cycle clock, IPO cycle clock or IPO_2 cycle clock).

Deactivation time for time-based cam

Deactivation time is also taken into account in setting a time-based cam.

Deactivation time must be:

- Deactivation time ≤ activation time + ON duration

Activation and deactivation times can vary independently of the I/O and can, therefore, influence the ON duration of the time-based cam.

Cam track activation

The **_enableCamTrack** command activates the cam track. On activation, the defaults are transferred to the system variables. If you explicitly want to use other values, these must be transferred with the command.

The following parameters are transferred via the _enableCamTrack command:

- Cam track data
- Output cam data

If you do not transfer any new data when activating the command, the defaults are used.

Cam track deactivation

Cam tracks are deactivated automatically or via a command.

Automatic deactivation

Automatic cam track deactivation is only possible when the configuration data octBaseCfg.keepEnabledOutOfTrackRange has been set to **NO**. In this case deactivation occurs on exiting the domain of the cam track, i.e. the track start (in a negative direction) or the track end (in a positive direction). It is not possible to reverse the direction of movement repeatedly within the track length. Output cams can therefore be output repeatedly. Automatic deactivation is set as default. As of V4.1 you can set the deactivation via the configuration data.

Note

In the cam track **Configuration** window, you can configure the automatic deactivation via **Leave non-cyclic activated cam track active in the axis range**.

Deactivation via command

The **_disableCamTrack** command is used to deactivate the cam track.

You can parameterize the deactivation time for the **_disableCamTrack** command (see Section **Start mode and stop mode**).

Leave cam track active in the axis range (as of V4.1)

Non-cyclic activated cam tracks are deactivated per default when the cam track length is exited. So that the non-cyclic cam track remains active over the entire axis range (also outside of the cam track length), you must set the configuration data **octBaseCfg.keepEnabledOutOfTrackRange** to **YES**. When the cam track length is exited, the non-cyclic cam track remains active and is deactivated, for example, via command.

Note

In the cam track **Configuration** window, you can deselect the automatic deactivation via **Leave non-cyclic activated cam track active outside of the track range**.

Features

- Valid for modulo axes and non-modulo axes
- With modulo axes, the cam track is only switched in the appropriate modulo range of the axis, and **not** in every modulo range. This also ensures a clear assignment when cam track lengths ≥ modulo length.
- Value=**NO**: Non-cyclic cam track is only active within the cam track length.
  If the axis or external encoder moves out beyond the cam track length, the cam track is deactivated. Returning to this length triggers new switching operations.
- Value=**YES**: Non-cyclic cam track is active over the entire axis range, also after leaving the cam track length If the axis or external encoder moves out beyond the cam track length and then back into the cam track range, the configured output cams switch again.

Example

There is a linear axis (non-modulo axis) with a traversing range of -1000 mm to 1000 mm and a cam track with a track length of 200 mm (-100 mm to 100 mm). The cam track is to remain active over the entire axis range and the cam track is to be activated non-cyclically.

After the cam track is exited (2), it remains active and is switched again after the reversal of direction (3) (see figure below).

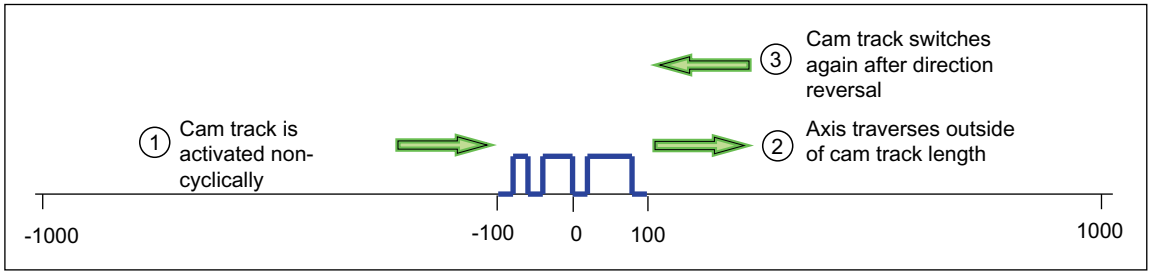


Figure 4-225 Leaving cam track active over the entire axis range

Start mode and stop mode

Start mode and stop mode are used to parameterize behavior on activation or deactivation of a track.

Start mode

The start mode (**startMode**) is used to define when the cam track should become effective after activation, or how tracks should be changed.

The mode is defined either as part of the cam track configuration in the **Default** window, or directly as a default setting using the **_enableCamTrack** command.

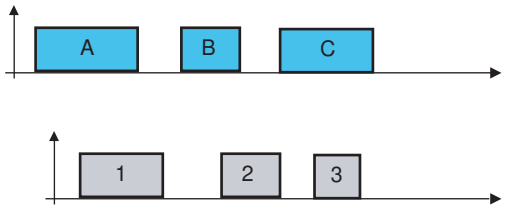
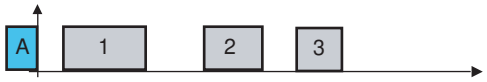
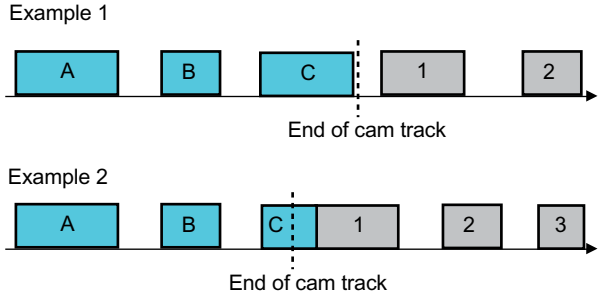
Table 4-167 Types of start mode

Start mode	Description
Effective immediately (default) (IMMEDIATELY)	Track becomes active immediately. If an output cam (or time-based cam) is defined or active at the current position of the old track, the output is aborted. The new track is enabled without delay (as quickly as possible). This enables high-speed exchange of cam tracks. If an output cam is already controlled and new data from the exchanged cam track continues to control the track signal, the track signal is not interrupted.
Immediately when cam track output inactive (IMMEDIATELY_BY_CAM_TRACK_OUTPUT_INACTIVE)	Changeover is made to the new cam track if no single output cam is active (any longer) on the old cam track. An active (output) single output cam on the old cam track is still output completely. If a changeover to the new track has not been made at the start position of a single output cam on the new track, this output cam is not output. Only after the tracks have been exchanged are the subsequent output cams on the new track output.
With next track cycle (NEXT_CAM_TRACK_CYCLE)	Track does not become active until the next track cycle, after either the axis reference position (in the positive traversing direction) or the end position of the cam track (in the negative traversing direction). The cam track end of the old track equals the axis reference position of the new track. Immediately the first output cam on the new track switches, a changeover is made to the new track. Up to that point, a time-based cam on the old track is output. The previous cycle is processed according to the previous command, the next cycle according to the **_enable** command. This allows the next track to be enabled, although another track is currently being processed. It is necessary to use this mode if the enable must occur before the start of the new cam track, for example, if the first output cam lies at the very start, but an output cam at the very end of an old (inactive) track is not to be enabled by mistake.
Last programmed value (EFFECTIVE)	The last programmed stop mode is active. If a stop mode is not programmed, the user default setting is used.

Changing cam track output on the basis of the selected start mode

The effect of the start mode on cam track output is shown in the following table for two cam tracks. The examples refer to activation of the same cam track with new or modified data.

Table 4-168 Start mode examples

Mode	Description	Display on cam track output
	One cam track with different data is given. • Cam track 1 (A to C) active cam track. • Cam track 2 (1 to 3) is activated.	
Effective immediately (IMMEDIATELY)	• New cam track becomes active immediately. • Cam track output becomes inactive. • Output cams 1, 2 and 3 are output.	
Cam track output inactive (IMMEDIATELY_BY_CAM_TRACK_OUTPUT_INACTIVE)	• New cam track becomes active at a cam track output of zero. • Output cam A on the active track is output completely. • Output cams 2 and 3 are output. • Output cam 1 is not output.	
With next track cycle (NEXT_CAM_TRACK_CYCLE)	• Cam track is exchanged at the axis reference position of the new cam track. • Example 1, position-based cam: Position-based cam of the old track is terminated. • Example 2, time-based cam: New cam track becomes active with the first output cam of the new track, at the latest. Up to that point, time-based cams remaining from the old cam track are still output.	

Stop mode

The stop mode (**stopMode**) is used to define the behavior of the cam track on deactivation.

The mode is defined either as part of the cam track configuration in the **Default** window, or directly as a default setting using the **_disableCamTrack** command.

Table 4-169 Types of stop mode

Stop mode	Description
Effective immediately (default) (IMMEDIATELY)	Track is deactivated immediately. If an output cam (or time-based cam) is defined or active at the current position of the track, the output of output cam is aborted.
Immediately when cam track output inactive (IMMEDIATE-LY_BY_CAM_TRACK_OUTPUT_IN-ACTIVE)	If no single output cam is active (any longer), the active cam track is stopped. An active (output) single output cam is still output completely.
At end of cam track (BY_CAM_TRACK_END)	Track is deactivated at its end. Immediately the final output cam on the track switches, the track is deactivated. Up to that point, a time-based cam on the track is output.
Last programmed value (EFFEC-TIVE)	The last programmed start mode is active. If a start mode is not programmed, the user default setting is used.

Output activation mode

Cam tracks can be output in a cyclic or non-cyclic mode. This setting is also transferred when activating the cam tracks (**_enableCamTrack**)

Cyclic output

The **CYCLIC** setting in the **activationMode** parameter is used to predefine the activation mode for cyclic output of the cam track.

The cam track's track length is mapped from the start position and continued/repeated **cyclically**. The cam track switches after the axis reference position and remains active until it is switched off with **_disableCamTrack**.

Non-cyclic output

The **NO_CYCLIC** setting in the **activationMode** parameter is used to predefine the activation mode for non-cyclic output of the cam track.

The cam track is mapped from the start position, output **once only** and terminated automatically after reaching the end position or remains active in the axis range. The performance depends on the value of the configuration date **octBaseCfg. keepEnabledOutOfTrackRange**.

Example of cyclic and non-cyclic output

A cam track is mapped onto a modulo axis. The figure shows a representation of the different activation modes.

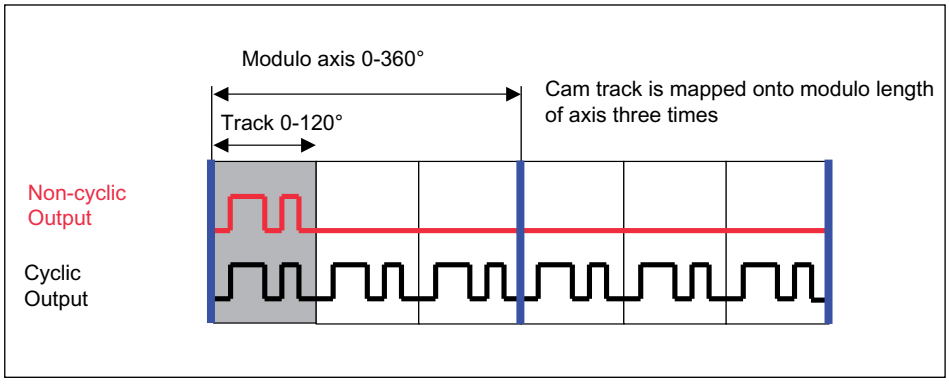


Figure 4-226 Example of cyclic and non-cyclic output on a modulo axis

Axis reference position and cam track offset

Cam tracks are defined independently of the axis. Output cams are not calculated according to the defaults until they are mapped onto the axis. The (**enableCamTrack.axisReferencePosition**) axis reference position is used to define from which position on the axis the cam track should be output. The axis reference position value can be negative or positive. The cam track is always output relative to this position data.

The axis reference position enables you to offset the cam track on the axis as you wish, and therefore to define when the output should take place (see figure in Section **Cam track features**).

A cam track is mapped to the axis range **exactly once**, beginning with the **axis reference position specified** in the enable command. This axis reference position represents the beginning of the cam track (applies for modulo and non-module axes). Upon activation, the cam track is executed once (NO_CYCLIC) or continued cyclically (CYCLIC).

Simulation

Operation can be simulated by means of the simulation commands on the cam track. The cam track status is then not output to the hardware output. In simulation mode, hardware cams behave as software cams. They are then only used for programming purposes.

If an active cam track is switched to simulation mode (**_enableCamTrackSimulation**), the output cam status remains the same, and only the control of the output is reset or interrupted.

Configure Units

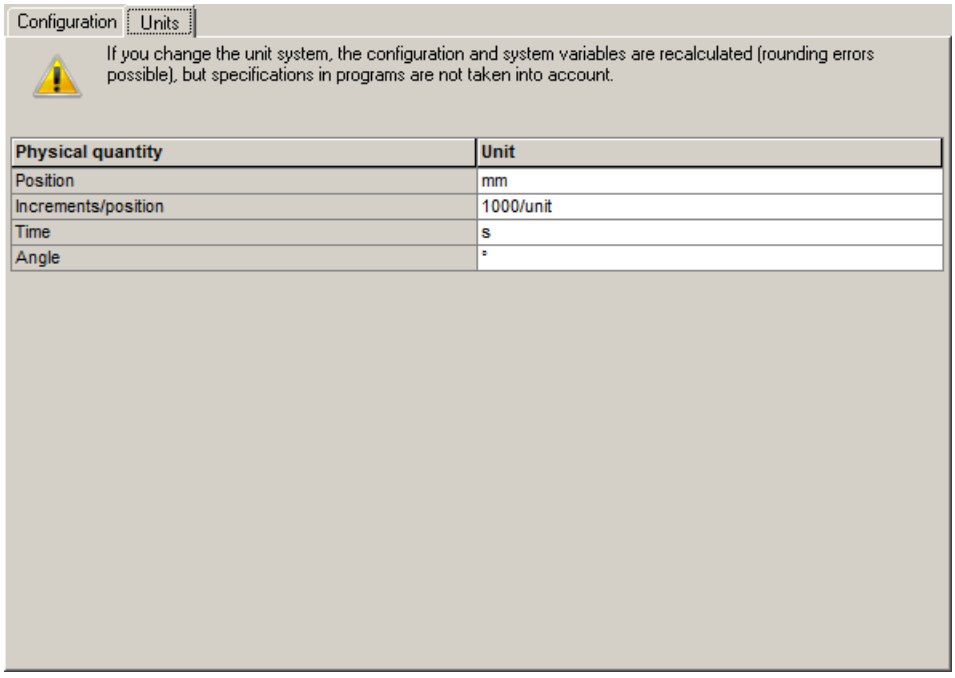
You can define the basic units for each technology object. The same physical variables can have different units in different technology objects. These are converted:

How to configure the units:

1. In the project navigator, open the context menu for the technology object.
2. In the context menu, select **Expert > Configure units**. The **Configure Units** window appears in the working area.
3. Select the **unit** for the **physical variables**. These units are used for the technology object, e.g. s for time units.

or

1. In the project navigator, open the **Configuration** under the TO.
2. Select the **Units** tab.



You can set the following parameters:

Field/button	Meaning/instruction
Table with units	
Physical variable column	Shows the physical variable. The physical variables which are used by the TO are available for the configuration.
Unit column	Displays and configures the unit. A drop-down list for selecting the unit appears when you click on the cell.
Toolbar	
	Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display
Close	Button for closing the dialog.
Help	Button for opening the online help for the dialog.

Mapping a cam track onto an axis

The cam track is defined independently of the axis. On activation, the cam track is mapped onto the axis. Only then are the switching states of the output cams calculated.

See also

Basics of cam track mapping (Page 1877)

Mapping output cams onto the cam track (Page 1877)

Mapping onto negative axis positions (e.g. linear axes) (Page 1878)

Relation of track length, modulo length and activation mode in mapping (Page 1879)

Basics of cam track mapping

- Conversion and mapping of the track onto the axis is identical for modulo and non-modulo axes. Certain points must be noted when mapping onto negative reference positions (see Section **Mapping onto negative axis positions (e.g. linear axes)**).
- The track length can be longer than, shorter than or equal to the modulo length of the axis. Based on this relationship, the switching states of the output cams may differ when they are mapped onto the axis.
- The positions of single output cams always relate to the cam track, not to the axis position. Only on activation of the cam track and entry of the axis reference position is a relationship to the axis position created (start of cam track output).
- If the cam track is activated and the axis rotates negatively, the track will also travel in a negative direction. There is no conversion on the basis of the direction of rotation (see figure). If a cam track should always be output in a positive direction, irrespective of the axis direction, this must be solved in the application.

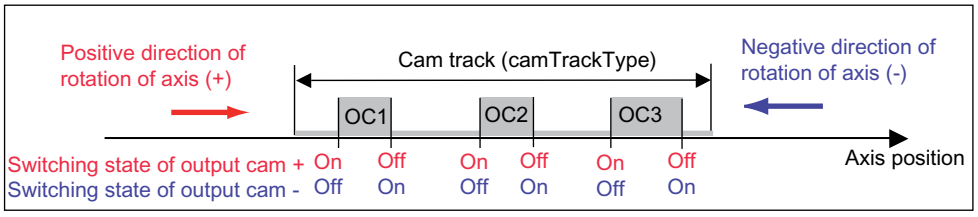


Figure 4-227 Cam track in positive and negative direction, depending on axis direction

- Cam tracks defined outside of the track length limits are mapped or converted during mapping onto the track range.

Mapping output cams onto the cam track

The **_enableCamTrack** command is used to map the starting and end positions of the output cam individually onto the cam track (not onto the axis). Output cams, which have been defined outside of the track range, are converted to this track. Negative starting and end-position values are also converted to the cam track.

"Unfavorable" starting and end-position default settings for output cams, e.g. output cam position outside of the track range, can lead to output cams being shifted, or new output cams being created. This must be taken into account when mapping the cam track onto the axis.

Note

After converting the cam track onto the axis, the effective output cam length is always shorter than or equivalent to the track length. No output cams are defined outside of the track length.

For standard applications, and in the interests of clear programming, automatic conversion of output cams should be avoided. This can be achieved by only defining output cams, which lie within the track length.

Mapping example

A cam track with three output cams (OC1 - OC3) is provided. The end position of output cam OC2 and the entire output cam OC3 are defined outside of the track length. After being mapped onto the axis, OC3 is converted to the track length and mapped to position OC3*. A new output cam OC4* arises from partial output cam OC2.

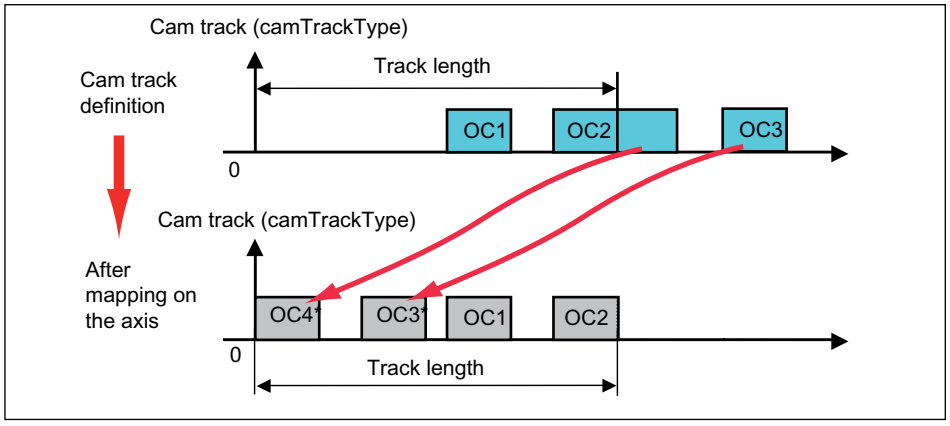


Figure 4-228 Parts of cam track lie outside of the track length

Mapping onto negative axis positions (e.g. linear axes)

Output cams on the cam track are always predefined positively. If you want to output output cams at negative axis positions, the cam track output start must be set in the negative range by means of the axis reference position.

Example of a linear axis with negative axis position

- Range of linear axis: -1000 mm to +1000 mm (non-modulo axis)
- Output of output cam at axis position: -100 mm to -200 mm
- Cam track length: 2000 mm
- Definition of the output cam on the track: SOC=800 mm; EOC=900 mm

By mapping the cam track via the axis reference position, the output cam OC1 can be output at a negative axis position.

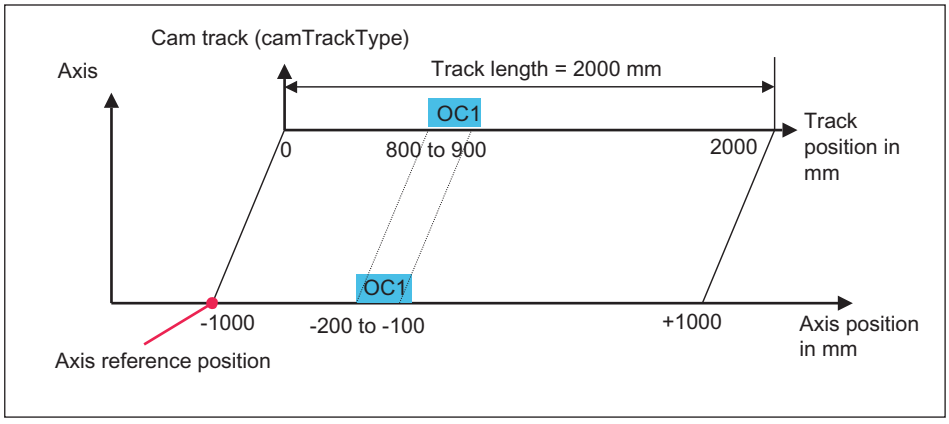


Figure 4-229 Linear axis with output of output cam at negative axis position

Please note that during cyclic output (CYCLIC), cam tracks are continued cyclically even with non-modulo axes, and thus the cam track can be output multiple times to different axis positions.

If a cam track is to remain switched on permanently and only output at one axis position, we recommend the following setting: cam track length ≥ axis traversing range.

This setting prevents the cam track from being continued cyclically in the case of cyclic output of a cam track after execution of the track range (in the figure: axis position - 1000 to +1000).

In the above figure, for example, the cam track would be continued cyclically

- in the positive range from axis position 1000 to 3000; 3000 to 5000 etc
- In the negative range from axis position -1000 to -3000; -3000 to -5000 etc.

.

The same behavior applies also for non-modulo rotary axes.

Relation of track length, modulo length and activation mode in mapping

In conventional output-cam output (comparable to mechanical cam controllers), the track length corresponds to the modulo length of the axis and cyclic output takes place.

With electronic cam controllers, the track length can be shorter or longer than the modulo length of the axis, therefore offering a greater degree of flexibility.

- Track length (tl) < modulo length (m) (integer ratio m to tl)
  Track length is output n-fold (n = m/tl) on modulo length.
- Track length (tl) > modulo length (m) (integer ratio tl to m)
  Output takes place on every nth rotation (n = tl/m). Output always takes place after the first quadrant. If the output takes place differently, a greater track length must be defined and the output cam placed accordingly.
- Non-integer division ratios lead to the cam track being dislocated on every axis rotation.

Use of cyclic output with track length = n x modulo length

If the track length is n times the modulo length, cyclic output allows a repetitive output of output cam to be easily achieved on every nth rotation (e.g. an air nozzle, which is always activated in the same angular range on every nth rotation).

Use of cyclic output with track length = 1/n x modulo length

A cycle scan rate is described using the example of a packaging machine with variable product lengths.

A cam controller controls/triggers all machine functions through a machine cycle of 0-360° (fed from left-hand area, see figure below). The product lengths may vary and are always mapped at 360°.

The machine cycle is subdivided into four identical operation steps at an operating station (right-hand area). The output cam is output cyclically for the operating station, with the track length = 1/4 modulo length of axis. The output cams on a cam track define the operating steps for one of the four identical feeds.

The advantage of this solution is that the product defaults are set in mm of the blister length and calculation/mapping at 360° only has to take place once. In this example, one machine cycle is used for four purposes. The same outputs are required for one use. Therefore, the same configuration does not have to be performed four times. Rather, the repetition factor n is calculated into the track length of the cyclically active track.

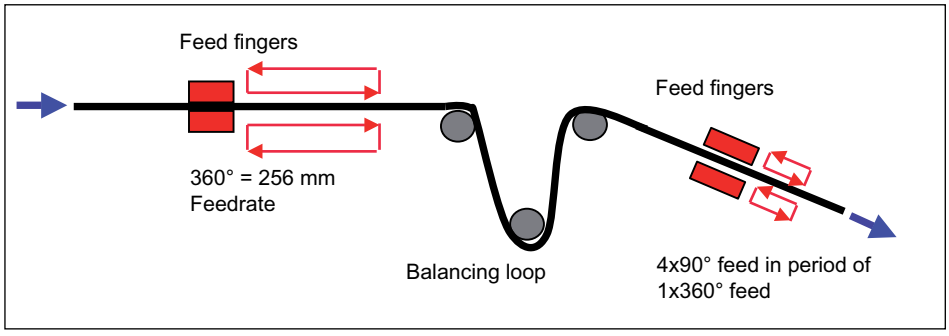
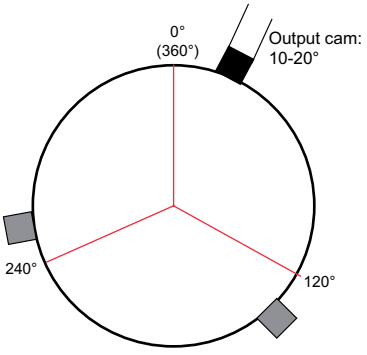


Figure 4-230 Packaging machine with reduced cycle

Example of a modulo axis with cyclic output

Table 4-170 Example of a modulo axis with cyclic output and track length < modulo length

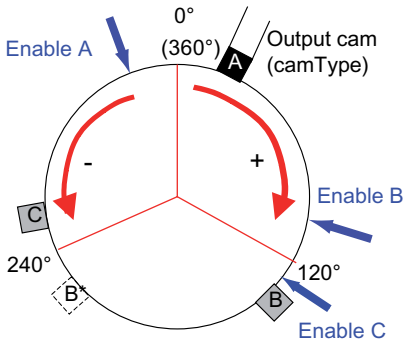
Cam track data/Explanation	Representation
• Modulo length: 360° • Track length: 0-120° • Output cam defaults: SOC=10°, EOC=20° Output cam is output cyclically every 10-20°, 130-140°, 250-260°, etc. If the direction of movement is reversed at 50°, for example, the 10-20° output cam will be output again.	

Example of a modulo axis with non-cyclic output

The example (following table) below shows a modulo axis with non-cyclic output, **next-track-cycle** start mode, and varying enable positions and axis reference positions.

- **Enable A, axis reference position 0°, positive direction of rotation:**
  output cam A is output.
  If the cam track is not exited, output cam A switches multiple times when the direction is reversed. The cam track output is terminated on exiting the cam track.
- **Enable A, axis reference position 240°, negative direction of rotation:**
  output cam C is output during the next cycle.
- **Enable B, axis reference position 120°, positive direction of rotation:**
  output cam B is output.
- **Enable C, axis reference position 240°, positive direction of rotation:**
  output cam C is output.

Table 4-171 Example of a modulo axis with non-cyclic output and track length < modulo length

Cam track data/Explanation	Representation
• Modulo length: 360° • Track length: 0-120° • Output cam defaults: SOC=10°, EOC=20° • Start mode: with next track cycle	

Cam track operating behavior

Changes made to the configuration and defaults of cam tracks or their associated axes during operation affect the cam track, which is active at that time. This section briefly describes the most important changes.

An explanation of how you can determine the status of single output cams and cam tracks is also provided.

The values of system variables are stored in the **userdefault** array. This array is transferred on using **_enableCamTrack** to activate a cam track. These defaults are configured during cam track configuration or other values can be written to them dynamically in the user program.

See also

Changing output cams on a cam track during runtime (Page 1882)

Changing the track length during operation (Page 1882)

Changing the axis configuration when a cam track is active (Page 1883)

Calling up the status of cam tracks and single output cams (Page 1883)

Changing output cams on a cam track during runtime

Changing start and end positions of an output cam

The start and end positions of single output cams can only be changed by transferring the new single positions to the **userdefault** array and using **_enableCamTrack** to activate the changed array.

Validity of single output cams on a track

You can define whether single output cams on a track are valid or invalid. This enables you to define whether single output cams should be output or not. If existing output cams are to be activated on an already active cam track, the relevant **userdefault.singleCamSettings.cam.cam[0-31].validity** system variable can be set with the value **YES** or **NO**, and the **_enableCamTrack** system function must be executed.

When parameterizing the cam track in SIMOTION SCOUT, validity can be set during configuration in the **Default** window and the **Output cam data** tab (see the **Defining cam track defaults** section).

Disabling or enabling valid output cams of a cam track without reactivation via _enableCamTrack (as of V4.1)

Valid output cams of a cam track can be quickly disabled or enabled via the **enableValidCam** system variable without reactivation of the cam track via **_enableCamTrack**.

Default setting of the **enableValidCam** system variable is **0xFFFFFFFF**, i.e. all valid output cams are enabled. By setting the bit of the relevant output cam, e.g. Bit_0 for output cam 0, the valid output cam is enabled with **1** and disabled with **0** (e.g. only output cam 0 is disabled with **0xFFFFFFFE**).

Properties of enableValidCam:

- Value of **enableValidCam** is retained with reactivation by **_enableCamTrack**
- During system ramp-up or with **_resetCamtTrack**, **enableValidCam** is set to the default setting 0xFFFFFFFF.
- Invalid output cams cannot be enabled via **enableValidCam**.
- System variable takes effect immediately without activation of the cam track.

Changing the track length during operation

The **userdefault.camTrackLength** system variable can be used to change the track length of an active cam track during operation. Changes to a track length do not become effective until the cam track is reactivated (**_enableCamTrack**). The track length of the cam track that is already active remains unchanged unless it is overridden by the changed cam track.

Example of a changed track length

- 360° track length is changed to 400° and the new track is activated.
- Output cam defaults: SOC=310°, EOC=30°

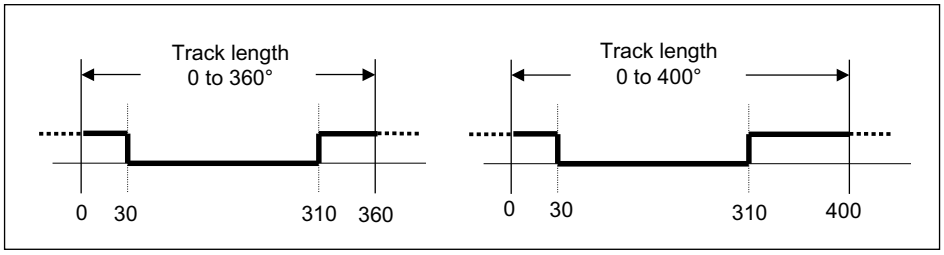


Figure 4-231 Changed track length with an effect on switching states

Changing the axis configuration when a cam track is active

Changing the axis configuration for an assigned, active cam track affects cam track behavior.

Changing modulo length

Changing the modulo length of an axis does not affect the definition or behavior of a cam track when it is enabled.

If the modulo length of an axis is changed, the conversion of the cam track on that axis is not automatically adjusted. If necessary, when changing the modulo length, you must deactivate the active cam track and activate it again, so that the axis can be mapped according to the new modulo length.

Output-cam output is aborted if the modulo length on an axis is changed and the axis is restarted.

Redefine axis

If the axis position is changed during operation, e.g. with **_redefinePosition**, the cam track is aborted and restarted. The change is interpreted as a skip in the modulo range. The cam track is mapped onto the new modulo range of the axis.

Calling up the status of cam tracks and single output cams

You can detect the status of single output cams and cam tracks at any time via system variables, and use the status in the user program.

Table 4-172 Status and position of cam tracks and single output cams

System variable	Meaning	Description
control	Functional status of Cam Track TO	The variable displays the state of the cam track. For example, it can be active, inactive or waiting for the next cam track cycle.
state	Output status	The variable displays if cam track output is in an ON or OFF state.

System variable	Meaning	Description
singleCamState	Status of single output cam	The **singleCamState** system variable is used to read out the status of single output cams. The variable consists of a 32-bit array, in which the lowest bit (bit0) represents output cam 0.
camTrackPosition	Position of the cam track	The **camTrackPosition** system variable is used to read out the actual position of a cam track operation within a cam track cycle. The cam track position is required, because the actual track position of a cam track cannot be determined by means of the axis position of a modulo axis (as, for example, the track length could be longer than the modulo range of the axis). The detected value always lies between the start (always "0") and end (defined by the cam track length) of the track.

Inverting a cam track

If the application requires, you can invert the cam track's activation level. The Cam Track TO retains its positive logic. The cam track switches at level 1 or high.

You can set the inversion via the **OctTechnologicalCfg.invertOutput** configuration data element or in the **Configuration** window of the cam track. The TO must be restarted in order for a change to be made.

Effect of cam track parameters on mapping

This chapter uses examples to explain the effect of configuration changes on cam track mapping onto an axis.

See also

Basic mapping of a simple cam track (Page 1884)

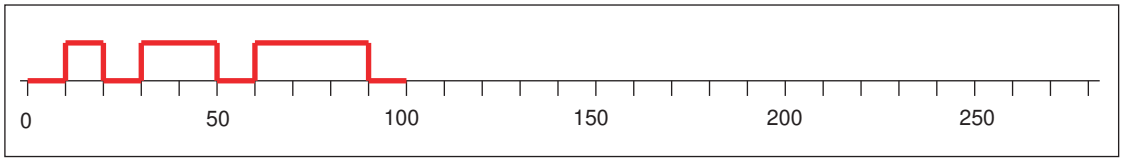
Advanced mappings with shifted output cam positions (Page 1886)

Basic mapping of a simple cam track

One cam track with the following data is given.

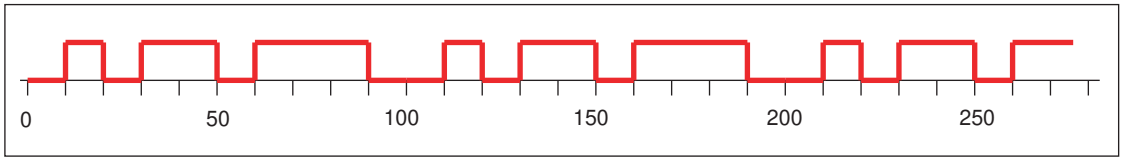
- Three position-based cams (10-20; 30-50; 60-90)
- Activation mode: Non-cyclical cam track activation
- Start mode and stop mode: Effective immediately

- Track length: 100
- All other user-default variables are the default setting unless another setting is mentioned explicitly.

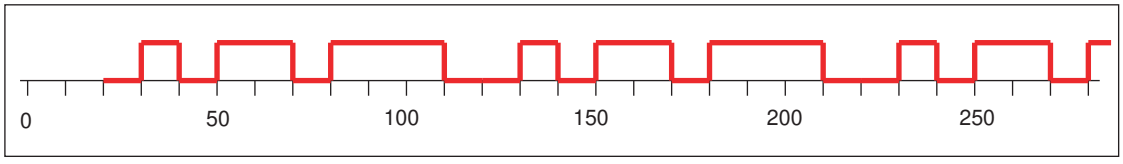


Display of given cam track with modified axis reference position, activation mode, and track length

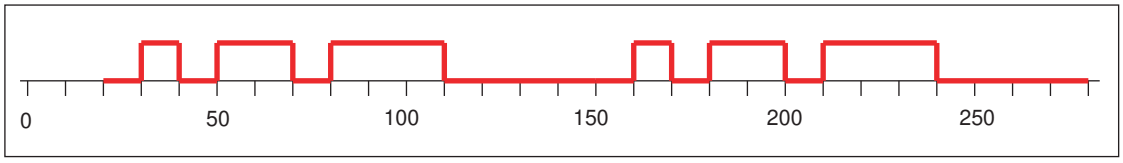
- Activation mode: Cyclic



- Activation mode: Cyclic
- Axis reference position: 20



- Activation mode: Cyclic
- Axis reference position: 20
- Track length: 130



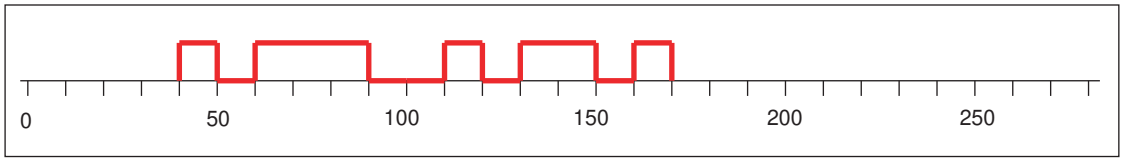
The following applies:

- A change of axis reference position causes a change in the cyclic and non-cyclic mode. The cam track is offset once.
- A track length change only affects the cyclic mode and causes an offset in the cam track cycle.

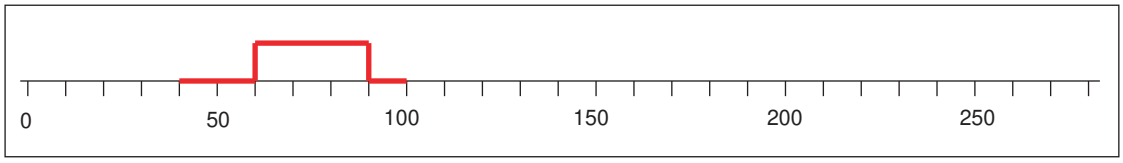
Display of given cam track with modified start/stop mode

- Start mode: Effective immediately
- Position of axis on which _enableCamTrack occurs: 40
- Position at which _disableCamTrack occurs: 170

- Activation mode: Cyclic



- Start mode: Immediate for inactive output cam track output
- Position of axis on which _enableCamTrack occurs: 40

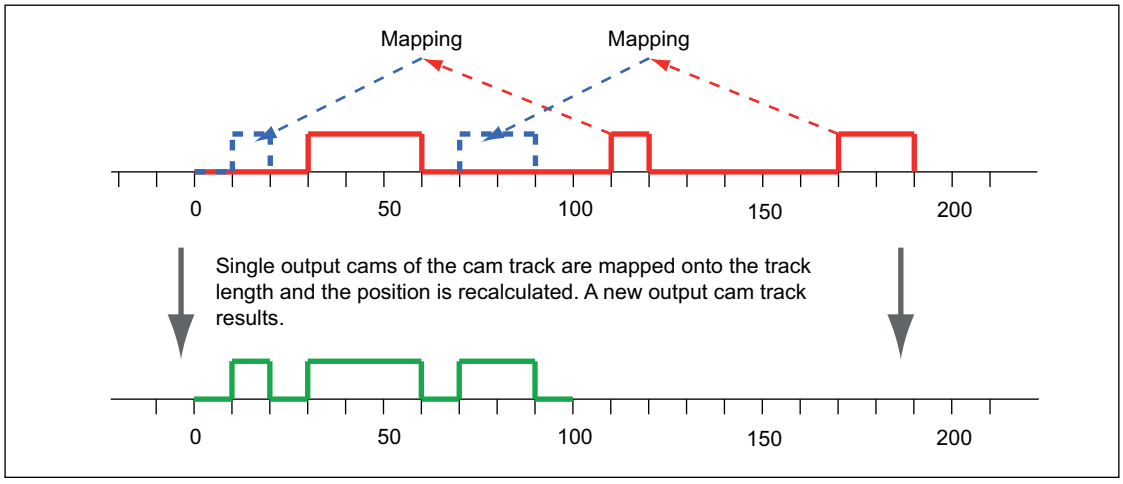


Advanced mappings with shifted output cam positions

The following chapter presents examples of a cam track with shifted single cams, i.e. when they are mapped onto the axis, single cams are shifted to another position if they are defined outside the track length.

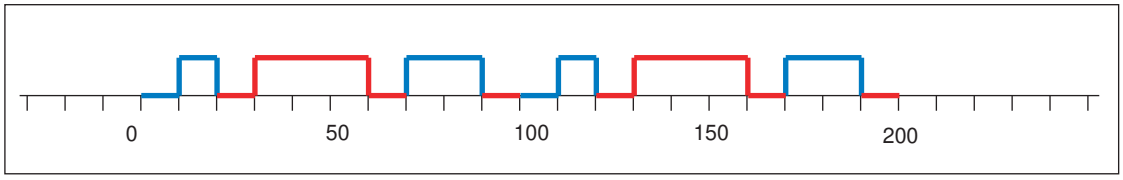
One cam track with the following data is given.

- Three position-based cams (30-60; 110-120; 170-190)
- Track length: 100
- Axis reference position: 0
- All other user-default variables are the default setting unless another setting is mentioned explicitly.

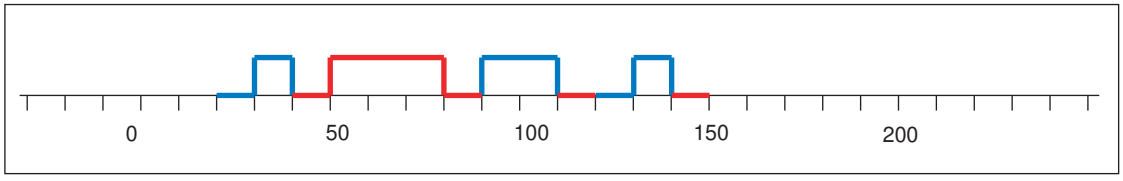


Display of given cam track with modified activation mode and axis reference position

- Activation mode: Cyclic



- Mode: Cyclic
- Axis reference position: 20
- Position at which _disableCamTrack occurs: 150



4.3.4.3 Configuring the TO Cam Track

This chapter describes typical operations used when working with the Cam Track technology object.

See also

Inserting cam tracks (Page 1887)

Parameterizing the Cam Track technology object (Page 1888)

Using expert list for cam tracks (Page 1889)

Configuring a cam track (Page 1889)

Defining cam track defaults (Page 1893)

Determining derivative-action times for cam tracks (dead time compensation) (Page 1907)

Using HW enable for cam tracks (Page 1909)

Inserting cam tracks

Note

Before you insert a cam track, the axis or external encoder to which the cam track is assigned has to be created.

If the cam is output to a TM15/TM17 High Feature or ET200xP TM Timer DI/DQ, the module must be added and configured before cam track configuration.

To insert a cam track:

1. In the project navigator, highlight the **OUTPUT CAMS** folder under the relevant axis or external encoder.
2. Select **Insert > Technology object > Cam track** or double-click **Insert cam track** in the project navigator under the axis or external encoder in the OUTPUT CAMS folder. The **Insert cam track** window appears.
3. Enter a **name** for the cam track. You can also enter a **comment**.
   Names must be unique throughout the project. For this reason, all the existing output cam tracks are displayed under **Available cam tracks**.
4. Confirm with **OK**. In the working area, the window for the configuration is displayed and the created cam track TO is shown in the project navigator.

Parameterizing the Cam Track technology object

General information about configuration data and system variables

Two data classes are distinguished when parameterizing a TO.

Configuration data defines the principal functionality of a TO. They are set within the object configuration framework with the SCOUT engineering system and are not normally changed during runtime.

System variables provide status data of the TO for the user program and a parameterization interface on the TO. System variables can be changed during runtime.

Note

You will find more information on technology objects in the *SIMOTION Runtime Basic Functions* functional description.

To parameterize a cam track

1. In the project navigator under the **OUTPUT CAMS** folder, find the cam track technology object (TO) that you want to parameterize. Double-click the cam track TO to display the associated objects.
2. Double-click **Configuration** or **Default** in the project navigator. The window appears on the workspace.
   - **Configuration** (see chapter **Configuring a cam track**):
     Define the values for the **configuration data** of the cam track here. This includes, for example, output cam type.
   - **Default** (see chapter **Defining cam track defaults**):
     Define the cam track defaults of the **system variables** here. This can include, for example, cam track and output cam data.
3. Changing configuration data and defaults

4. Click **Close** to accept the changes.
5. Repeat steps 2 to 4 for all objects in which you want to change the configuration data and defaults.

Using expert list for cam tracks

Parameters required for standard SIMOTION applications (configuration data and system variables) are parameterized in the Cam Track TO directly by means of screen forms or are defined automatically.
It may be necessary to change automatically-defined parameters for special SIMOTION applications. These configuration data and system variables can only be displayed and changed in the expert list.

Note

You will find more information on working with the expert list in the *SIMOTION Runtime Basic Functions* functional description.

Configuring a cam track

In the **Configuration** window, define the configuration data values for the cam track.

Double-clicking in the project navigator below the cam track on the **Configuration** element displays the window in the working area.

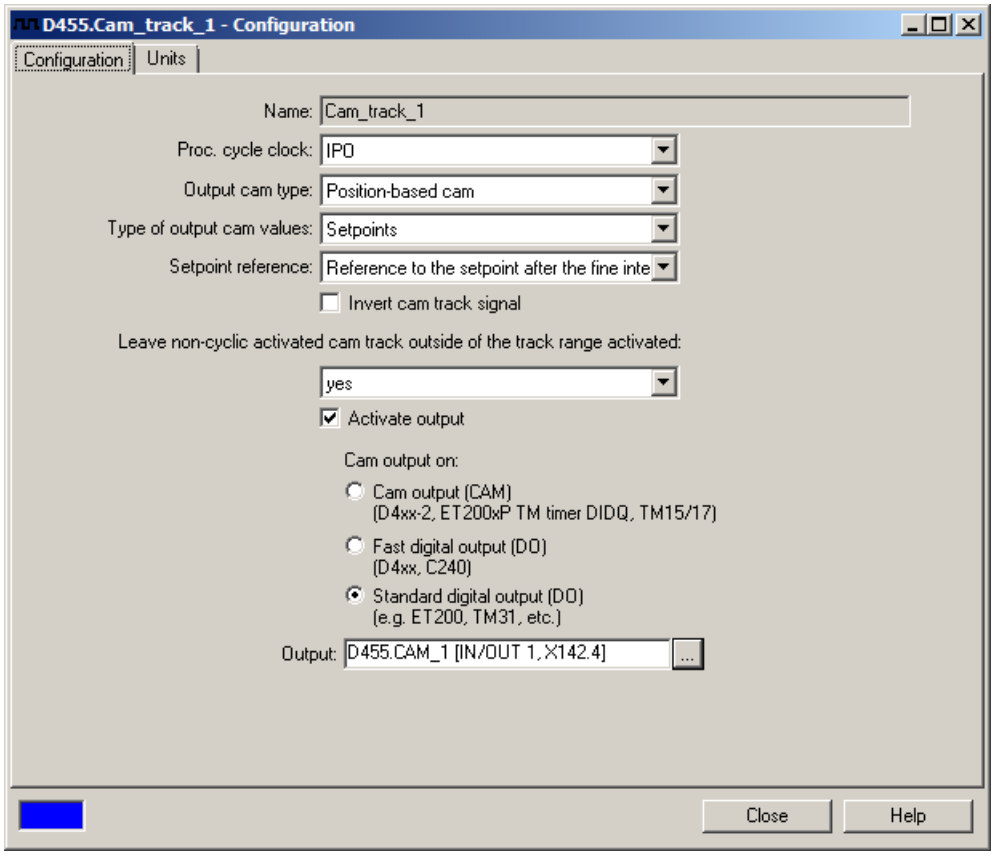


Figure 4-232 Output cam configuration using the example of a position-based cam

You can set the following parameters:

Table 4-173 Cam track configuration data

Field/button	Meaning/information
Name	The name of the created cam track is displayed here.
Output cam type (See Chapter **Output cam types of the single output cams on a track**)	Choose the type of output cam used on the track in **Output cam type**. **Position-based cam** (default value) The switching signal is active when the position of the axis lies between two markers (start and end position). **Time-based cam** The switching signal is on for a specific period of time after reaching the switching position (start position). **Time-based cam with maximum ON length** A maximum ON length can also be defined for time-based cams. This means that the time-based cam is deactivated once it has covered the maximum length, even though the parameterized time has not yet expired.

Field/button	Meaning/information
Processing cycle clock	Choose **Processing cycle clock** to select the system cycle clock used to update the output cam signal at the output or in the system variables. The cam calculations are performed in IPO or IPO_2 cycles or in servo cycles. The processing cycle clock is set in the configuration by means of the **OctBaseCfg.taskLevel** configuration data. **IPO** (default value) The output cam signal is updated in the interpolator cycle clock. **IPO_2** The output cam signal is updated in the interpolator cycle clock 2. The IPO_2 cycle clock length is at least twice that of the IPO. **Servo** The output cam signal is updated in servo cycles. **The following configurations of the processing cycle clock are possible:** • Axis in IPO cycles and output cam in IPO_2 cycles • Output cam in servo cycles and axis in IPO or IPO_2 cycles For the possible setting of IPO_fast and Servo_fast with D435-2, D445-2 and D455-2, see Section Second servo cycle clock (Servo_fast) in the SIMOTION Runtime Basic Functions Manual. **Note:** • It is not possible to configure the axis in the servo cycle clock and the output cam in the IPO or IPO_2 cycle clock. • It is not possible to configure the axis in the IPO cycle clock and the output cam in the IPO_2 cycle clock if it is a setpoint output cam. • It is not possible to configure the axis in the IPO_2 cycle clock and the output cam in the IPO cycle clock. **Note:** When the Servo:IPO ratio is ≠ 1:1, then the greatest possible accuracy for the calculation is reached for "output cams related to position value" when the **servo cycle clock** is set as the processing cycle clock for the cam track TO.
Type of output cam value	Select the position value that is the reference for the output cam during processing.
Actual value reference	Select the following settings here depending on the type of the output cam values: • **Reference to the actual value on the encoder without considering Ti**. Reference to the actual position in the controller before the position filter. • **Reference to the actual value after the position filter**. Reference to the actual position in the controller after the position filter. • **Reference to the actual value on the encoder. Ti is taken into account**. Reference to the actual value on the encoder module / drive. The transmission time Ti from the encoder module / drive to the controller is taken into account by the system.
Setpoint reference	Select the following settings here depending on the type of the output cam values: • **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoints will already have been reached at the end of this IPO cycle, e.g. if a setpoint is calculated for the IPO cycle clock with the virtual axis and also displayed at the end of the cycle. Reference is made directly to the setpoint applicable at the end of the cycle. • **Reference to the setpoint after the fine interpolator**. The cam switch points are calculated as if the setpoint calculated in the interpolator will be output completely in the following cycle and the calculated position setpoint will therefore be reached at the end of the following cycle. • **Reference to the setpoint on the drive**. Calculation of the cam switch points according to the setpoint output with the current settings on the drive.

Field/button		Meaning/information
Activate output		Activate the checkbox if the output cam signal is to be applied to a digital output. Parameters are displayed.
Cam output on		
	Cam output (CAM)	If the output checkbox is activated and the cam output (CAM) radio button selected, the cam output is on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. In the case of D4x5-2 and the TM17 High Feature, the resolution is, for example, 1 µs. Supported hardware: • SIMOTION D410-2 • SIMOTION D4x5-2 (X142) • TM15, TM17 High Feature • ET200xP TM Timer DI/DQ The I/O channel must be configured as **CAM**. For more details, see cam output types. (Page 1811) **Note** Cam output (CAM) or high-speed digital output (DO) are also known as high-speed, hardware-supported output cams.
	High-speed digital output (DO)	If the output checkbox is activated and the "High-speed digital output (DO)" radio button selected, the output cam is output via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle clock. The time that it takes for the axis to reach the output cam switching position with reference to the processing cycle is calculated by linear extrapolation. Calculated from the beginning of the 1st position control cycle, the output cam function is triggered by a hardware time when this time is reached. Supported hardware: The onboard I/O of the following CPUs is used: • SIMOTION D4x5 (interface X122, X132), 8 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as DO) • SIMOTION D410 (interface X121), 4 high-speed cam outputs, as of V4.1 (the I/O channel must be configured as DO) • SIMOTION C240, C240 PN (interface X1), 8 high-speed cam outputs For more details, see cam output types. (Page 1856) **Note** Cam output (CAM) or high-speed digital output (DO) are also known as high-speed, hardware-supported output cams.
	Standard digital output (DO)	If the output checkbox is activated and the "Standard digital output (DO)" radio button selected, the output cam is output in the servo cycle clock. The temporal resolution of the cam output is usually reduced by the output cycle of the I/O used. Supported hardware: • Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx) • Centralized I/O (SIMOTION C) • Distributed I/O via PROFIBUS DP/PROFINET I/O (e.g. ET 200, ET200xP TM Timer DI/DQ) • Drive I/O TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30 For more details, see cam output types. (Page 1856)

Field/button	Meaning/information
Output	The output can be symbolically assigned via the assignment dialog (see Chapter Symbolic assignment (as of V4.2) in the SIMOTION Runtime Functions manual) using the ... button in the Output field (symbolic assignment is activated by default in projects as of V4.2). If symbolic assignment is not active or if the CPU version is < V4.2, a physical output is assigned by entering the HW address and bit number in the Output field. Enter the logical HW address of the output to which the output cam signal is to be applied. Only the output cam signal may be present at this address. If other objects are already using this output, an error occurs that is reported following a download to the target system. The logical HW address must be located outside the process image and therefore be greater than 63. For details see Cam output types (Page 1811)
	Button for opening the assignment dialog (see Chapter Symbolic assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual). Select a parameter or an address in the Assign dialog.
	Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display

Defining cam track defaults

You can define defaults for every cam track. These values are stored in system variables and can be changed by programs.

Double-clicking in the project navigator below the cam track on the **Defaults** element displays the window in the working area.

See also

Track data (Page 1894)

Output cam data (Page 1896)

Track data

The defaults for the track system variables, e.g. track length, are displayed in the **Track Data** tab.

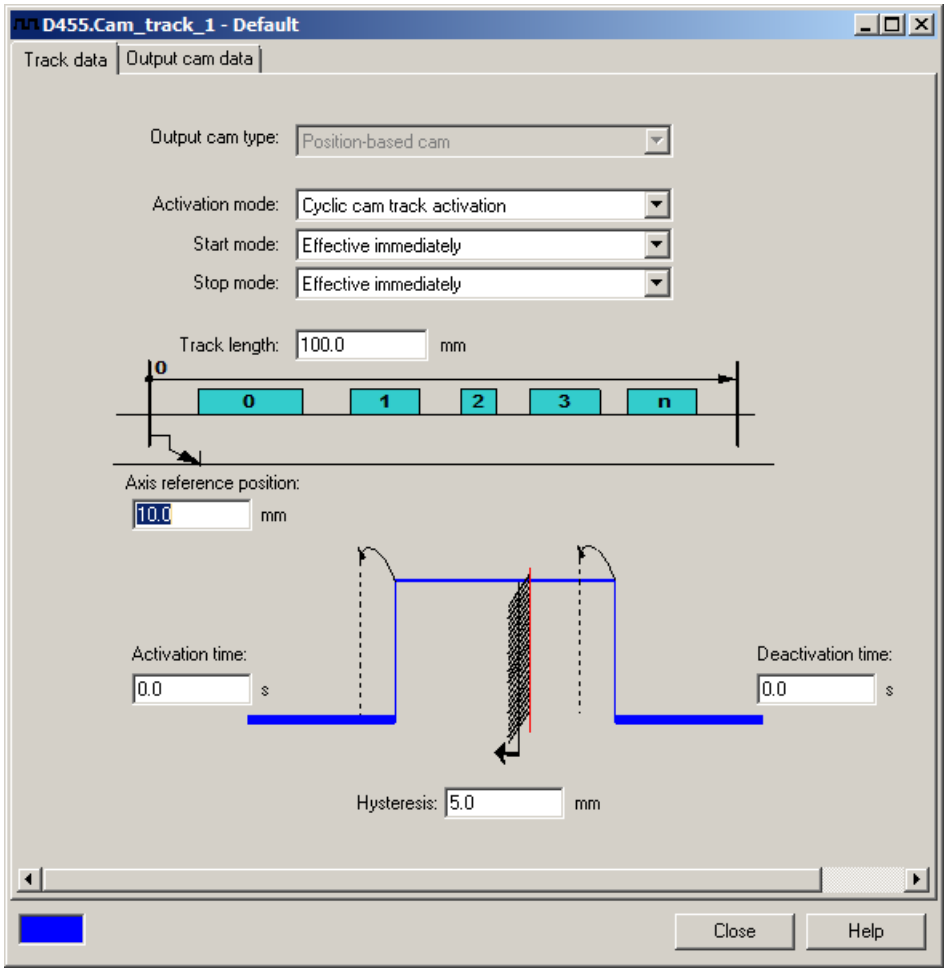


Figure 4-233 Cam track data using the example of a position-based cam

You can set the following parameters:

Table 4-174 Defaults for cam track data

Field/Button	Significance/Note
Output cam type	**Output cam type** displays the type of output cam selected for the cam track in the **Configuration** window.
Activation mode (See the **Output activation mode** section)	Cam tracks can be output in a cyclic or non-cyclic mode. **Cyclic cam track activation** The cam track's track length is mapped from the axis reference position and continued/repeated cyclically. **Non-cyclic cam track activation** Cam track is mapped from the axis reference position, output once, and terminated automatically after exiting the cam track or remains active in the axis range. This setting depends on the value of the **octBaseCfg.disableOutOfTrackRange** configuration data element.

Field/Button	Significance/Note
Start mode (See the **Start mode and stop mode** section)	The start mode (**startMode**) is used to define when the cam track should become effective after activation, or how tracks should be changed. **Last programmed value** The last programmed start mode is active. If a start mode is not programmed, the user default setting is used. **Effective immediately** Track becomes active immediately. If an output cam (or time-based cam) is defined or active at the current position of the old track, the output is aborted. The new track is enabled without delay (as quickly as possible). This enables high-speed exchange of cam tracks. **Immediately with inactive cam track output** Changeover is made to the new cam track if no single output cam is active (any longer) on the old cam track. An active (output) single output cam on the old cam track is still output completely. **With next track cycle** The track becomes active at the next track cycle. Immediately the first output cam on the new track switches, a changeover is made to the new track. Up to that point, a time-based cam on the old track is output.
Stop mode (See the **Start mode and stop mode** section)	The stop mode (**stopMode**) is used to define the behavior of the cam track on deactivation. **Last programmed value** The last programmed stop mode is active. If a stop mode is not programmed, the user default setting is used. **Effective immediately** Track is deactivated immediately. If an output cam (or time-based cam) is defined or active at the current position of the track, the output of output cam is aborted. **Immediately with inactive cam track output** If no single output cam is active (any longer), the active cam track is stopped. An active (output) single output cam is still output completely. **At cam track end** Track is deactivated at its end. Immediately the final output cam on the track switches, the track is deactivated. Up to that point, a time-based cam on the track is output.
Track length	Enter the track length. The track length is calculated from the start of the cam track to the end of the cam track.
Axis reference position (See the **Cam track features** section)	The axis reference position is used to define how the cam track is mapped on the axis, or from which axis position the cam track should be output.
Axis modulo length (See the **Relation of track length, modulo length, and activation mode** section)	The modulo length of the axis, which the cam track is linked to, is displayed here. The modulo length does not have to be identical to the track length. To change the modulo length of the axis, you must work through the axis wizard. It is only displayed when the axis is configured as a modulo axis.
Activation time (See the **Actuation times (activation time/deactivation time)** section)	Enter the **activation time** here. The output cam switching time is set to the point when the start position is reached, plus this period. The output cam position is adapted dynamically. This allows you to compensate for propagation delays. If a negative value is entered as an activation time, the switching signal is activated before the start position is reached.

Field/Button	Significance/Note
Deactivation time	Enter the **deactivation time** here. The output cam switch-off time is set to the point when the end position is reached, plus this period. The output cam position is adapted dynamically. This allows you to compensate for propagation delays. If a negative value is entered as a deactivation time, the switching signal is activated before the end position is reached.
Hysteresis (See the **Hysteresis** section)	Enter a range for the **hysteresis** here. The output cam does not change its switching state in this defined range around the switching position even under changed switching conditions. This prevents a repeated change of the switching state.

Output cam data

The defaults for system variables of single output cams on a track, e.g. starting and end position, are displayed in the **Output Cam Data** tab.

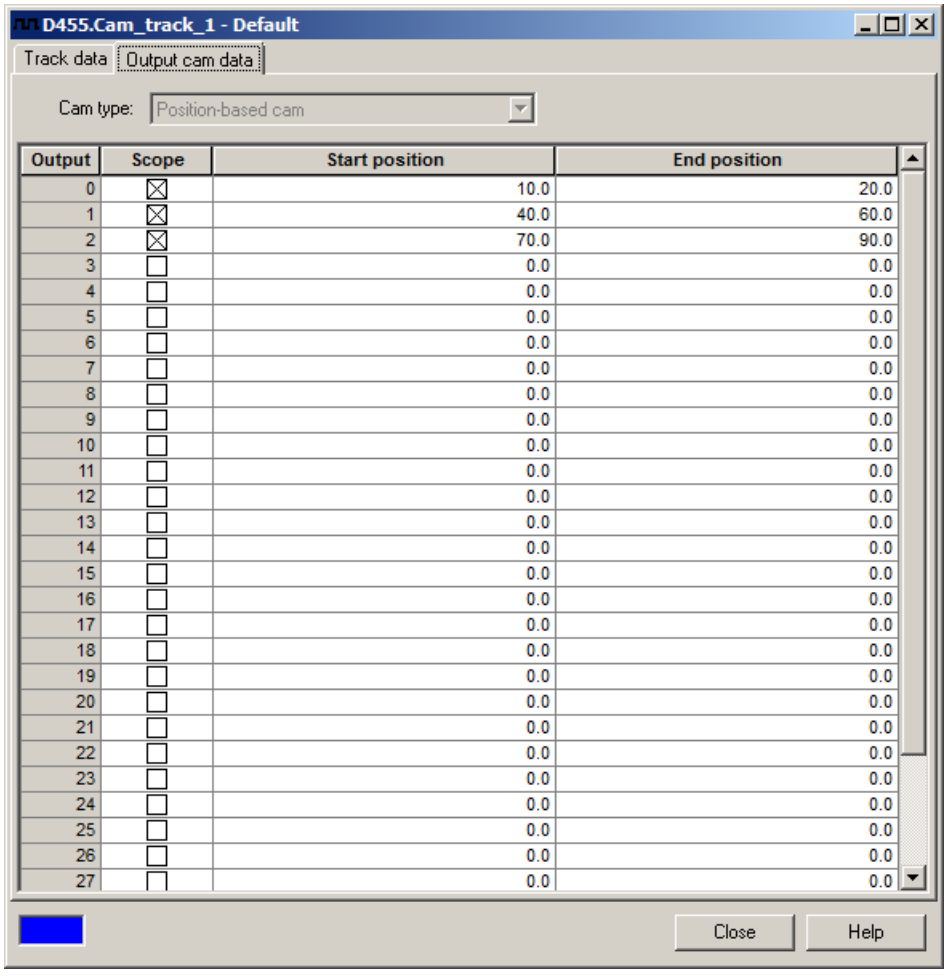


Figure 4-234 Cam track output cam data, position-based cam example

You can set the following parameters:

Table 4-175 Cam track output cam data defaults

Field/Button	Meaning/Note
Output cam (cam-Type)	The output cam number (0 - 31) is displayed here. Each track can have up to 32 output cams.
Validity	Select this checkbox, if you want to set the output validity of single output cams on a track. If the box is activated, the output cam will be output, along with the track. If it is not activated, the output cam is not output.
Starting position	Enter the starting position of the output cam.
End position	Enter the end position of the position-based cam here.
Last programmed ON duration (See Section **Time-based cam**)	Enter the ON duration for time-based cams here. The output cam switches off once the parameterized time has expired.
Max. ON length (See Section **Time-based cam with maximum ON length**)	Enter the maximum ON length for time-based cams with maximum ON length here.

Configuring cam tracks on SIMOTION D4xx onboard

Output cams and cam tracks can be configured for standard outputs, or as high-speed, hardware-based output cams / cam tracks.

A cam track can be configured on SIMOTION D4xx onboard as follows:

1. In the project navigator, switch to the Control Unit via **SINAMICS_Integrated > Control_Unit.**
2. Double-click **Inputs/outputs** below the control unit. The window appears on the workspace.
3. Switch to the **Bidirectional digital inputs/outputs** tab.
4. Click the **button** to switch between the input and output for the digital inputs/outputs (**DO8 to DO15**). In each case, switch the DI/DO to the output you wish to use as the output of output cam. The designation at the terminal strip of DI or DO switches to DO. Outputs of the output cam can only be used if they have been defined as an output. DO 8 is configured as an output in the diagram. For the output, select the **DO (SIMOTION)** setting.

Note

Mixed use of the SIMOTION D4xx DI/O as high-speed outputs (of output cams) and inputs of measuring inputs is possible.

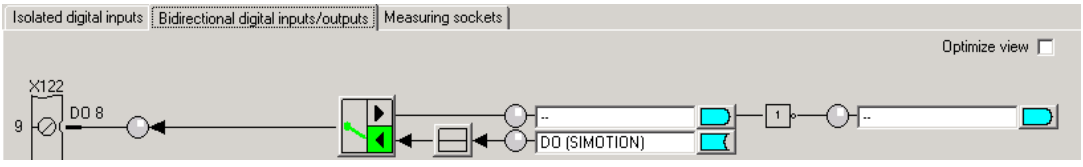


Figure 4-235 SIMOTION D4xx digital inputs/outputs

5. Click **Close**.
6. Insert a new output cam or a new cam track or use an existing one.
7. Parameterize the TO Output Cam / Cam Track
8. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
9. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. Should you wish to configure a high-speed output cam, select the **Activate output** check box and select the **High-speed digital output (DO)** radio button.
10. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
11. Click **OK** to close the window and select **Project > Save**.

To determine the logical hardware address for outputs on SIMOTION D4xx onboard (only if symbolic assignment is not activated)

1. In the project navigator, below the SIMOTION D device, select **SINAMICS_Integrated > Communication > Telegram configuration**.
2. Double-click Configuration and, in the window which appears, select the tab **IF1: PROFIdrive PZD telegram**. The components are displayed there with address range (input/output data).

3. Select SIEMENS telegram 390, 391 or 392 as telegram type. A maximum of eight output cams can be configured for each telegram. The number of DI/DO is limited to eight, i.e. only two output cams can be configured for telegram 392 if you are already using six measuring inputs. Therefore consider whether you also want to use measuring inputs during the telegram selection.

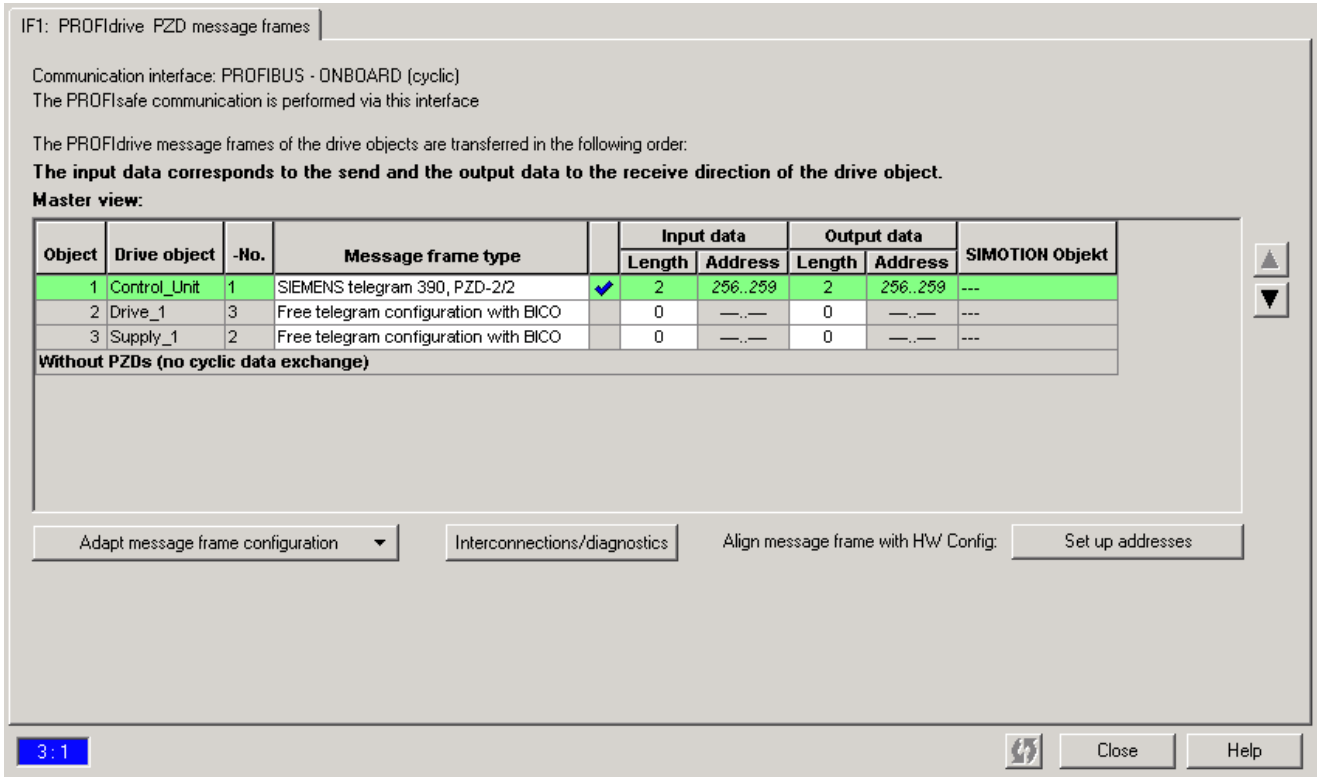


Figure 4-236 Determining the hardware address of the components

4. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If there are question marks in the fields instead of I/O addresses, you must also perform an alignment.
5. Now calculate the hardware address by adding the base output address (first value of the output data) of the Control Unit to the offset (for example, 256 + 3 = 259). The offset always has the value **3**.
6. You will find the bit number in the following table.

Table 4-176 Bit numbers for D410 and D4x5

Output D4x5	Output D410	Bit number
X122.7 (DI/DO 8)	X121.7 (DI/DO 8)	Bit 0
X122.8 (DI/DO 9)	X121.8 (DI/DO 9)	Bit 1
X122.10 (DI/DO 10)	X121.10 (DI/DO 10)	Bit 2
X122.11 (DI/DO 11)	X121.11 (DI/DO 11)	Bit 3
X132.7 (DI/DO 12)	-	Bit 4
X132.8 (DI/DO 13)	-	Bit 5

Output D4x5	Output D410	Bit number
X132.10 (DI/DO 14)	-	Bit 6
X132.11 (DI/DO 15)	-	Bit 7

Note

In the case of versions earlier than V4.2, when using 39x telegrams, the onboard D4x5 outputs are to be assigned exclusively to SIMOTION. During a consistency check in SIMOTION SCOUT, no check is made as to whether the entered HW address actually belongs to a **high-speed digital output (DO)**.

Configuring cam track on the SIMOTION D410-2

1. In the project navigator, switch to the **Control Unit** via **SINAMICS_Integrated > Control_Unit**.
2. Double-click **Inputs/outputs** below the control unit. The window appears on the workspace.
3. Switch to the **Bidirectional digital inputs/outputs** tab.
4. Click the button to switch between the input and output for the digital inputs/outputs (**DO 8 to DO15**). In each case, switch the DI/DO to the output you wish to use as the output of output cam. The designation at the terminal strip switches to DO. Cam outputs must always be defined as outputs so that they can be used. DO 8 is configured as an output in the diagram. For the output, select the **Output cam (SIMOTION)** setting.

   Note

   Mixed use of the SIMOTION D410-2 DI/DO as high-speed (output cam) outputs and inputs of measuring inputs is possible.

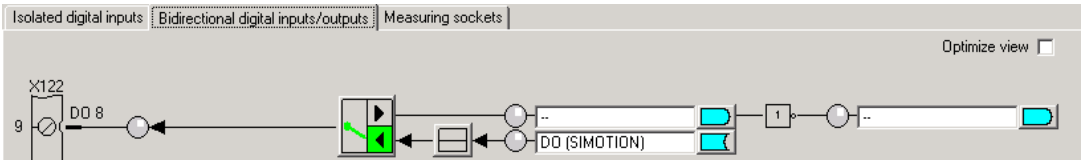


Figure 4-237 Digital inputs/outputs onboard

5. Click Close.
6. Insert a new output cam or a new cam track or use an existing one.
7. Parameterize the TO Output Cam / Cam Track
8. Double-click Configuration below the output cam or the cam track in the project navigator. The Configuration window appears in the working area.
9. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. Should you wish to configure a high-speed output cam, select the **Activate output** checkbox and select the **Output cam output (CAM)** radio button.

10. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
11. Click OK to close the window and select **Project > Save** from the menu.

Configuring cam tracks on SIMOTION D4x5-2 onboard

With SIMOTION D4x5-2 the outputs on the interface X142 are used for cam output

1. The **Inputs/outputs X142** entry in the project navigator can be used to open the configuration screen in HW Config.
2. For the selected I/O channel, select **Output cam** as the function.

Note

If you do not use symbolic assignments (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual), you must note the logical address. (see Figure I/O Properties) This address must be configured at the TO output cam

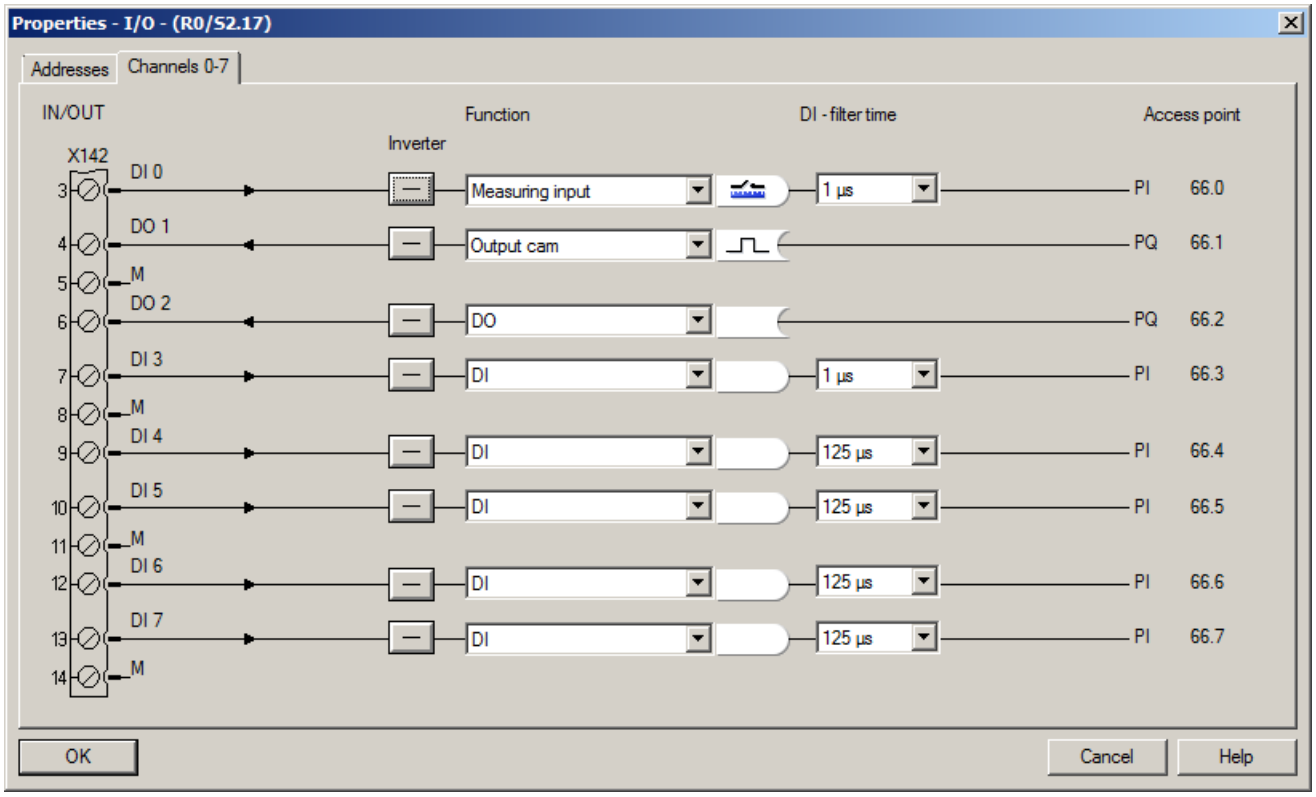


Figure 4-238 I/O Properties

3. Click OK.
4. Insert a new output cam or a new cam track or use an existing one.
5. Parameterize the TO Output Cam / Cam Track

6. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
7. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used.
   If you would like to configure a high-speed output cam, select the Activate output check box and select the output cam output (CAM) radio button.
8. Assignment of an output to an output cam/cam track is supported as of V4.2 using symbolic assignment or by entering the HW address.
9. Click OK to close the window and select **Project > Save** from the menu.

Configuring cam tracks on a TM15/TM17 High Feature

1. In the project navigator, below the input/output component (TM15/TM17) that you want to use, double-click the entry **Inputs/outputs.** The **Bidirectional Digital Inputs/Outputs** window is displayed.
2. For the selected I/O channel, select **Output cam** as the function.

   Note

   If you do not use symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual), you must note the offset (e.g. 3.1).

3. Insert a new output cam or a new cam track or use an existing one.
4. Parameterize the TO Output Cam / Cam Track
5. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
6. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used.
   If you would like to configure a high-speed output cam, select the **Activate output** check box and select the **Cam output (CAM)** radio button.
7. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address.
8. Click OK to close the window and select **Project > Save** from the menu.

To determine the logical hardware address for TM15/TM17 High Feature outputs (only if symbolic assignment is not activated)

1. In the project navigator, below the SIMOTION device or the SINAMICS drive unit
   - for SIMOTION D, select: **SINAMICS_Integrated > Communication > Telegram configuration**
   - for SINAMICS S/G drive unit (position axis only): **Communication > Telegram configuration**
2. Double-click **Telegram configuration** and, in the window that opens, select tab **IF1: PROFIdrive PZD telegram**. The components are displayed there with the address ranges (e.g. TM17 output data 304 to 315).

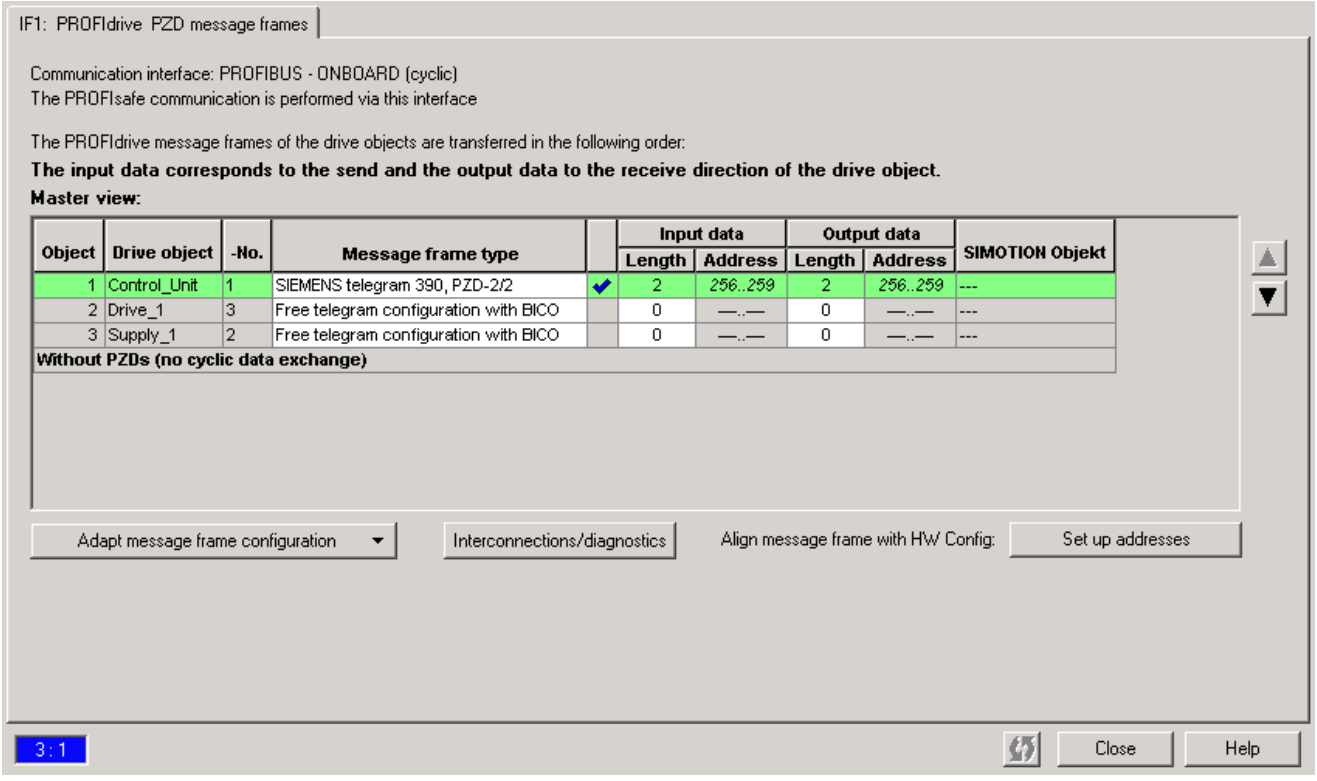


Figure 4-239 Determining the hardware address of the components

3. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If question marks are entered in the fields instead of I/O addresses, either alignment has not yet taken place, or the address is not recognized by SIMOTION SCOUT. In this case, you must perform an alignment.
4. Now calculate the HW address by adding the base output address (first value of the address range) of the TM to the offset (e.g. 304 + 3 = 307).
5. The bit number is defined by means of the offset. For example, a 3.1 offset of an output cam on DO 1 results in a bit number of 1.

Configuring a cam track on SIMATIC ET200SP and ET200MP

Overview

Description

You can configure TO cam track on the ET200SP technology module TM Timer DIDQ 10x24V and ET200MP technology module TM Timer DIDQ 16x24V.

- Configuration on the ET200SP technology module TM Timer DIDQ 10x24V is possible both using Step7 in connection with SIMOTION SCOUT Classic and with SIMOTION SCOUT TIA in connection with the TIA Portal. Further information can be found in the section Configuring output cams to TM Timer DIDQ 10x24V (Page 1904).
- Configuration on the ET200MP technology module TM Timer DIDQ 16x24V is only possible using SIMOTION SCOUT TIA in connection with the TIA Portal. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA.

Further information

Further information on the functions of the TM Timer DIDQ technology modules can be found in the SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V and SIMATIC S7-1500/ET 200MP Technology Module TM Timer DIDQ 16x24V.

Configuring a cam track on TM Timer, using DIDQ 10x24V as example

Description

You can configure TO output cam tracks on the ET200SP technology module TM Timer DIDQ 10x24V in SIMOTION SCOUT or with SIMOTION SCOUT TIA in connection with the TIA Portal. The configuration in the TIA Portal is described in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA. The configuration in SIMOTION SCOUT in connection with Step7 is described below:

In order to parameterize the ET200SP TM Timer DIDQ 10x24V with Step7 in connection with SIMOTION SCOUT Classic, a Hardware Support Package (HSP) for Step7 is required. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA in the section on hardware and software requirements.

For the TM Timer DIDQ technology module, the digital outputs are used for the cam output. Irrespective of whether you work with SIMATIC STEP 7 or SIMOTION SCOUT, the parameterization is always performed in HW Config:

Parameterization of TM Timer DIDQ in HW Config

1. Open the HW configuration in SIMOTION SCOUT or SIMATIC STEP 7.
2. Select the header module (IM) with the TM Timer DIDQ in HW Config.
   The TM Timer DIDQ appears in the detail view.
3. To open the context menu, select the TM Timer DIDQ with the right mouse key.

4. Select object properties.
   You can view or change the following properties in the object properties:
   - General
   - Addresses
   - Identification
   - Parameters
5. Open the tab **Parameter** > **DQ0 / DI0**
6. Set the parameter value **Configuration DQ/DI group** to **Use input/output individually**.

   Note

   With this setting, the DQ0 channels can be interconnected as cam output CAM or DI0 as measuring input MI under SIMOTION SCOUT.

7. Press **OK** to close the dialog.
8. Open the menu **Station** > **Save and Compile**.

Note

The technology modules and the individual channels may only be reparameterized in offline state. All changes only take effect after saving, compiling and subsequent download of your changed user project to the controller. The download can be performed in HW Config via **Target system** > **Download to module** or in SCOUT via **Load CPU / drive unit to target device**.

Parameterization of CAM in SCOUT

1. Insert a new output cam or a new cam track or use an existing one.
2. Parameterize the TO Output Cam/Cam Track.
3. Double-click Configuration below the output cam or the cam track in the project navigator. The Configuration window appears in the working area.
4. Activate the selection box **Activate output** and select the option **Cam output (CAM).**

5. Assignment of an output to an output cam/cam track is supported as of V4.2 using symbolic assignment or by entering the HW address.

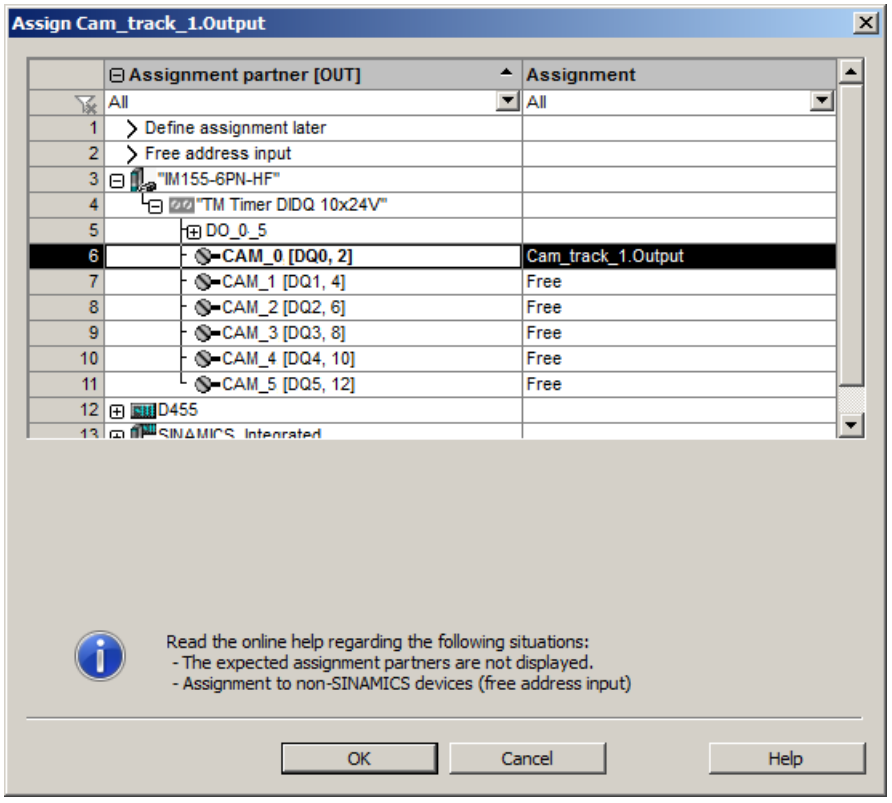


Figure 4-240 Assigning a cam output by means of symbolic assignment

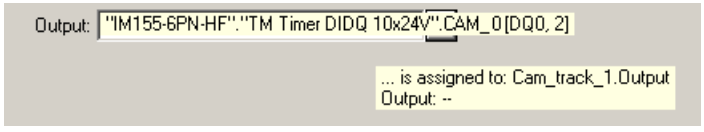


Figure 4-241 Assigning a cam output by entering the HW address

6. Click OK to close the window and select **Project** > **Save**.

Configuring cam tracks on SIMOTION C240

1. Insert a new output cam or a new cam track or use an existing one.
2. Parameterize the TO Output Cam / Cam Track
3. Double-click **Configuration** below the output cam or the cam track in the project navigator. The **Configuration** window appears in the working area.
4. For high-speed, hardware-supported output cams, you can achieve an output accuracy exceeding the servo cycle clock based on the hardware used. Should you wish to configure a high-speed output cam, select the **Activate output** check box and select the **High-speed digital output (DO)** radio button

5. Assignment of an output to an output cam/cam track is supported as of V4.2 either by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address. During a consistency check in SIMOTION SCOUT, no check is made as to whether the entered HW address actually belongs to a **high-speed digital output (DO)**.
6. Click OK to close the window and select **Project > Save** from the menu.

Determining derivative-action times for cam tracks (dead time compensation)

Depending on the system and the device, there is a certain time between the setting of a cam output by the program and the actual reaction of the actuator (e.g. solenoid valve). This time is called dead time and depends, for example, on the load-dependent delay times of a digital output, the switching properties of a valve, etc. Usually the exact value for the dead time is not known and can therefore be determined empirically through measurements.

In order that an output cam switches at the correct time, the dead time must be compensated by specifying a derivative-action time, which offsets the cam output by the dead time. Whereby it must be taken into account that the derivative-action times for switching an actuator on and off are usually different.

The empirical determination of the dead times using a difference measurement as an example.

Note

The procedure applies not only to output cams, but also to cam tracks. However, with cam tracks you can only specify a derivative-action time for the entire cam track.

Example

Lines of glue are to be applied to a product at a defined position and with a fixed length. The glue output is controlled by an output cam or a cam track. The glue is output from the start of output cam (switch-on point) to the end of output cam (switch-off point). The offset of the begin and end of output cam with respect to the velocity can be observed on the length and position of the glue line on the product (see figure). The figure below shows the line of glue for two velocities (v_1, v_2) with $v_2 > v_1$.

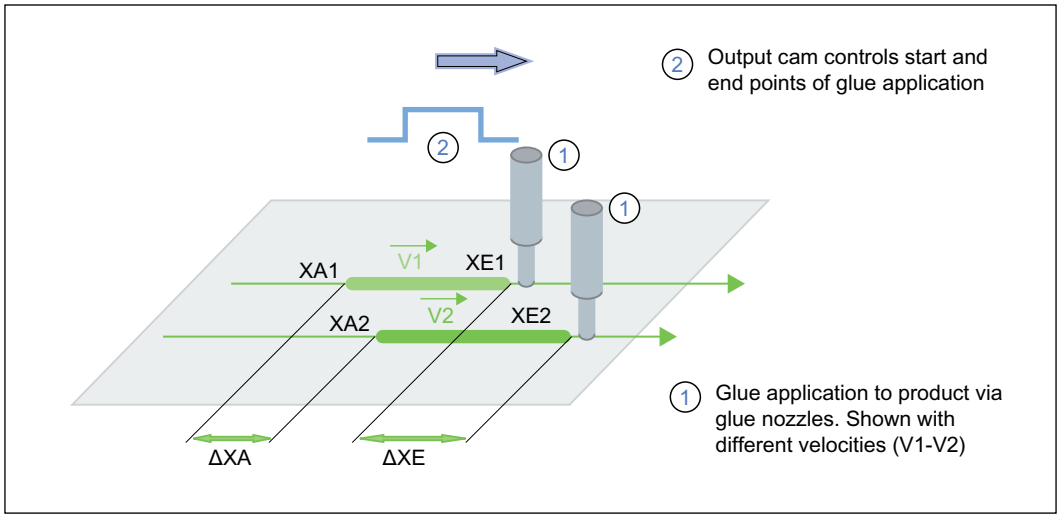


Figure 4-242 Offset of the output of output cam through dead times (dead time compensation)

Procedure:

1. Set all actuation times for start of output cam (activation time) and end of output cam (deactivation time) to 0.
2. Define the velocities for which the positions are to be determined. You should select two velocities that correspond to velocities that occur during production (e.g. minimum and maximum velocity).
3. Start the application and determine the start positions (x_{A1} and x_{A2}) and end positions (x_{E1} and x_{E2}) of the line of glue for the velocities v_1 and v_2.

   Note

   To increase the accuracy, you can perform several comparison measurements and use the average measured values.

4. You can determine the actuation times for the output of output cam using the following formula.
   $\mathbf{t_{activation} = \Delta s/\Delta v = (x_{A2}-x_{A1})/(v_2-v_1)}$
   $\mathbf{t_{deactivation} = \Delta s/\Delta v = (x_{E2}-x_{E1})/(v_2-v_1)}$

5. Enter the calculated actuation times as **activationtime** for the start of output cam and as **deactivationtime** for the end of output cam. Note that the actuation time must be entered as a negative when the output time is to be before the programmed output cam switching time.
6. After you have determined the activation time and the deactivation time for the output of output cam, you should perform a control measurement and check the result.

Note

Depending on the application, it may be, e.g. with eccentric presses, that there is no linear relationship between dead time and velocity (e.g. non-linear response of an applied brake). You have to dynamically adapt the dead time to the respective velocity for these applications. This can be implemented in the application with a user program. After the actuation time has been changed, you have to activate the output cam again with **_enableOutputCam** or the cam track with **_enableCamTrack**.

Using HW enable for cam tracks

You can make the output of cam tracks dependent on a hardware-supported enable (only for TM17 High Feature). The cam track is, for example, output cyclically on the TM17 High Feature. The enable signal (level or edge-controlled) triggers output of the cam track at the output of the TM17 High Feature.

A measuring input TO can be configured to implement an edge-controlled enable at the enable input of the TM17 High Feature. In this case, the cam track is only enabled at the output of the TM17 High Feature once the configured measuring signal occurs.

A hardware enable can be configured for Cam Track TOs, as well as Output Cam TOs. The procedure to be used is described below using the example of a Cam Track TO. Subject to any stated restrictions, this procedure is also used for Output Cam TOs.

In the case of edge-controlled HW enable of a cam output, the input bit **actualInputState** of the assigned "Measuring input" does not provide its terminal level but the level of the enable signal.

Note

You will find more information on the hardware enable in the *Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

See also

Absolute level-controlled (TM17 High Feature) (Page 1910)

Absolute edge-controlled (TM17 High Feature) (Page 1911)

Setting (overriding) the enable via a program (Page 1912)

Relative edge-controlled (Page 1912)

Absolute level-controlled (TM17 High Feature)

Required configuration for level-controlled HW enable

- Cam track TO or output cam TO is configured
- Digital output on TM17 High Feature parameterized for cam track output, and level-triggered enable input set at this output. The appropriate enable input for the enable signal is parameterized automatically.
- Digital output configured for cam track output (HW address)
- Cam track TO or output cam TO must be active.

Level-controlled enable procedure

For cam tracks with a level-controlled enable, output cams are output for as long as the cam track is active in the TO (set with the **_enableCamTrack** command) and as long as a hardware enable signal, which has been parameterized on the TM17 High Feature, is present. This means that a cam track can also be output multiple times with a continuous enable signal. An output cam, which is already active, is still output even if the enable is deactivated.

An output cam cannot be controlled until a hardware enable signal is present at the TM17 High Feature. If the enable takes place within an output cam, it is no longer output. Subsequent output cams are however output. This can be assigned parameters in the track enable of the TO (see Chapter **Start mode and stop mode**).

It is also possible to operate the output with inverted logic, i.e. the enable input on the TM17 High Feature can be operated inversely and then works in LOW-active mode.

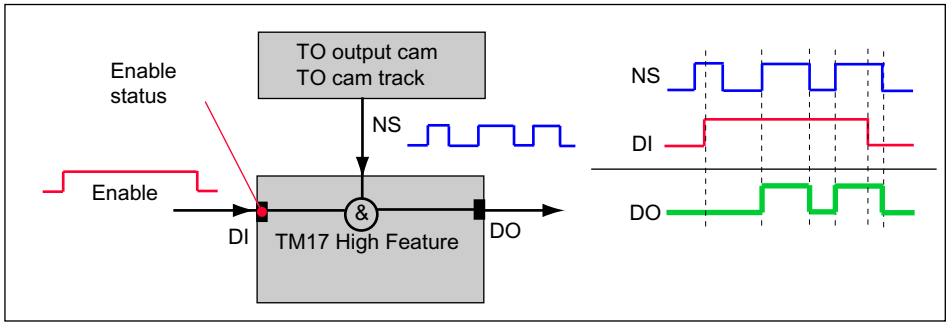


Figure 4-243 Schematic representation of a level-controlled HW enable

Note

With level-controlled HW enables, the **state** system variable does not indicate the status of the output (DO), but rather the status of the cam track signal (C-TS).

Determining the status of the enable

You can determine the status of the enable via the I/O area of the digital input for the enable signal. You will find more information in the *Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

Absolute edge-controlled (TM17 High Feature)

Required configuration for edge-controlled HW enable

- TO camTrack or TO outputCam is configured.
- Digital output on TM17 High Feature parameterized for cam track output, and edge-triggered enable input set at this output. The appropriate enable input for the enable signal is parameterized automatically.
- TO measuringInput configured (measuring range, edge, operating mode of measurement once).
- TO measuringInput is interconnected to the Enable input for the enabling signal.
- Digital output on the TO configured for cam track output (HW address)
- TO camTrack or TO outputCam must be active.

Evaluating edges for the measurement

For the measuring input, only the measurement once mode is permitted with the following edge detection:

- Rising edge
- Falling edge
- Both edges

A measuring of **both edges, first rising** and **both edges, first falling** and the **cyclic measuring** mode is not supported.

Edge-controlled enable procedure

In the case of edge-controlled HW enable inputs, the cam track is output when the track is active (**_enableCamTrack** command set) and the configured edge has been detected by the configured measuring input at the TM17 High Feature input. This edge enables the hardware gate for outputting the active cam track. The edge is detected in the configured measuring range (if configured) or with **_enableMeasuringInput** (without measuring range) on the measuring input.

The measuring range operates with the IPO/IPO_2 or position control cycle clock and the enable input (measuring input) resolution at 1us. The enable signal position can be evaluated in the usual way using the measuring input.

An output cam cannot be controlled until a hardware enable signal is present at the TM17 High Feature (edge). If the enable takes place within an output cam, it is no longer output. Subsequent output cams are, however, output.

As soon as a new measurement job is transmitted (measuring input activated) or a new measuring range begins, the enable for the active output is reset.

To achieve quick response times, the cam track is continually output by the TO (cyclic cam track output), i.e. all output cams are transmitted to the TM17 High Feature.

In principle, non-cyclic cam track output is possible (e.g. if a hardware enable is coupled to a software enable). The cam track is then only output once after the HW enable.

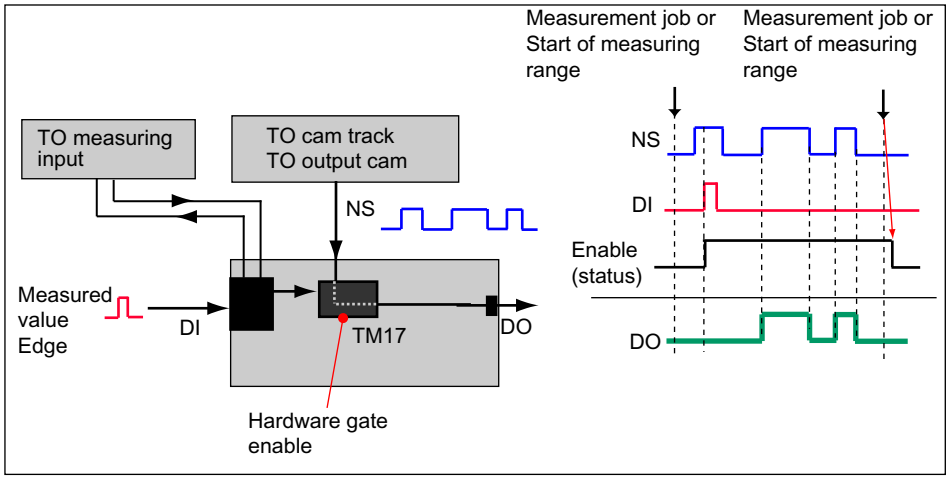


Figure 4-244 Schematic representation of an edge-controlled HW enable

Note

With an edge-controlled HW enable, the **state** system variable does not indicate the status of the output (DO), but rather the status of the cam track signal (C-TS).

Determining the status of the enable

You can determine the status of the enable via the I/O area of the digital input for the enable signal. You will find more information in the *Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

Setting (overriding) the enable via a program

It is possible to set the HW input of the enable for the cam track via a SW enable signal. For this, you have to access the HW input directly via its address in the user program and set the bit.

- **Level-controlled:**
  Enable is set for as long as the bit is set (functions as enable input). This can be achieved by logically ORing the software enable and enable input (i.e. the enable can also be active "outside of" the software enable, if the enable input is active).
- **Edge-controlled:**
  Enable is set for as long as the bit is set, irrespective of the measurement jobs measuring input TO.
  If the measured-value edge is detected at the enable input during software enable, the enable will remain active even after the software enable has been canceled until a new measurement job / new measuring range occurs.

Relative edge-controlled

You can use the user program to implement a relative, edge-controlled enable input for a cam track.

Follow the steps outlined below:

1. Configure the cam track.
2. Configure a measuring input that detects the measured result for exchanging the cam track (e.g. position of a workpiece edge).
   The measuring input can be connected, for example, to a drive via PROFIBUS DP, a SIMOTION CPU, or a TM15/TM17 High Feature. Unintended edges apparent during measuring can be hidden using the measuring range.
3. Depending on the measured position, the configured cam track can be activated with **_enableCamTrack** in the user program and the detected position can be calculated via the axis reference position. Cam track output then occurs relative to the measured position.

Note that output cams cannot be output immediately after the measurement has taken place, due to data transfer times (for example bus runtimes).

This must be taken into account as follows in your application:

- A certain time interval should be left between the measurement and output of the first output cam.
- The edge-detecting sensor should be positioned in the machine accordingly.

4.3.4.4 Programming/References of TO Cam Track

Programming

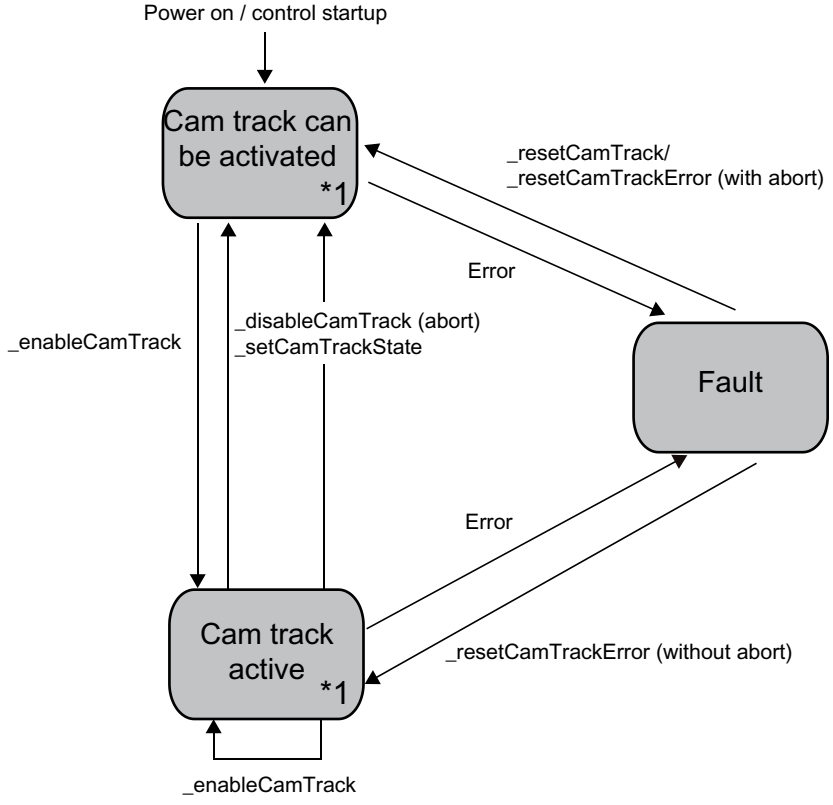


Figure 4-245 Programming and execution model for the Cam Track TO

***1** The following commands are available in the technology object states **can be activated** and **active**:

- **_disableCamTrackSimulation**
- **_enableCamTrackSimulation**

The simulation commands are modal/act in parallel and do not replace any existing _enableCamTrack commands.

Commands

The Cam Track technology object can be addressed in the user program using the following commands:

Table 4-177 Cam Track TO system functions

Commands	Description	Application
_enableCamTrack	Activates cam track execution	Cam track evaluation is activated. If the switching condition for the output of output cam is fulfilled, the output or **state** system variable is set. It is also effective for axes that have not been homed.
_disableCamTrack	Deactivates cam track execution	Cam track evaluation is deactivated. If the switching condition for the output of output cam is fulfilled, the output or **state** system variable is not set.
_enableCamTrackSimulation	This function simulates a cam track by disconnecting the output. The cam track remains internally active, the status is retained, the cam track output is not switched.	Values are calculated, but not forwarded to the hardware. Hardware output cams act as software cams. The cam track remains internally active, the status is retained, the cam track output is not switched. If an active cam track is switched to simulation mode, the output cam status remains the same, and only the control of the output is reset or interrupted.
_disableCamTrackSimulation	The cam track is reset from simulation mode. The cam track output is switched according to the cam track status and the signal inversion.	The output of output cam is switched according to cam track status and signal inversion.
_setCamTrackState	This function deactivates the cam track function and sets the cam track status to the specified value.	This is used if the output should not be controlled by the cam track TO. **Example:** A glue nozzle is controlled by the cam track TO (applying glue dots). As a service function, it should also be possible to rinse the nozzle while constantly controlling it. This is achieved via **_setCamTrackState**.

Commands	Description	Application
_resetCamTrack	This function switches the cam track to an initial state. Pending errors are deleted. Modified configuration data is reset on request.	Creating initial state of cam track TO
_resetCamTrackError	Resets cam track errors. It is terminated with a negative acknowledgment for any errors that cannot be acknowledged at this point.	E.g. acknowledging configuration errors after entering correct values.
_getCamTrackErrorNumberState	Readout of error number status.	Check for occurrence of an error with the specified error number
_getStateOfCamTrackCommand	This function returns the execution state of a command.	Check whether or not cam track switching has already taken place (i.e. the command ID is still available or has already been deleted)
_resetCamTrackConfigDataBuffer	Changed configuration data is collected and stored in a buffer, and is activated with this command in the RUN configuration. This function deletes the configuration data collected in the buffer since the last activation without activating it.	Changing configuration data in the RUN state discards the accumulated modifications.
_bufferCamTrackCommandId	This function enables **commandId** and the associated command status to be saved beyond the execution period of the command. The **commandId** parameter is used to define the command for which the respective status is to be saved. The maximum number of savable command states is specified in the **decodingConfig.numberOfMaxBufferedCommandId** configuration data element.	Subsequent check of how command was terminated, e.g. error-free or number of error that occurred.
_removeBufferedCamTrackCommandId	This function ends the saving of **commandId** and the associated command status beyond the execution period of the command.	Explicit deletion of previously saved command IDs.

For further information on the system functions, please refer to the *SIMOTION TP CAM Reference Lists*.

Process Alarms

You can predefine local alarm responses via SIMOTION SCOUT.

Note

For more information, refer to the *Motion Control Technology Objects Basic Functions* functional description.

How to configure the alarm response:

1. Double-click **Execution system** in the project navigator below the SIMOTION device. The execution system opens.
2. In the execution level tree, select **SystemInterruptTasks > TechnologicalFaultTask**.
3. Click the **Alarm Response** button in the window that then opens. The **Alarm Response** window appears. You can configure the alarm response for every TO here.

A system variable **error** indicates that a technology alarm has been generated. The response to the alarm is displayed in the **errorReaction** variable.

Table 4-178 Possible alarm responses

Alarm Response	Description	Application
NONE	No response	-
DECODE_STOP	Command processing is aborted, the cam track function remains active. Execution on the technology object can continue after **_resetOutputCam** or **_resetOutputCamError**.	The Cam Track TO can only be reactivated after the error has been acknowledged.
CAMTRACK_DISAB LE	Command processing is aborted, current cam track function is aborted. Execution on the technology object can continue after **_resetCamTrack** or **_resetCamTrackError**.	The Cam Track TO can only be reactivated after the error has been acknowledged.

TO Cam Track menus

Cam track menu

Grayed-out menu functions cannot be selected. The menu is only active if a Cam Track TO window is active in the working area.

You can select the following functions:

Table 4-179 Cam track TO menu

Function		Significance/Note
Close		Select **Close** to close the configuration window for the cam track that is open in the working area.
Properties		Select **Properties** to display the properties of the cam track highlighted in the project navigator.
Configuration		Select **Configuration** to determine the configuration data (e.g. output cam type) of the cam track.
Default		Select **Default** to define the defaults of the system variables (e.g. track data and output cam data) of the cam track.
Expert		
	Expert list	Select **Expert list** to open the expert list for the highlighted cam track. The configuration data and system variables can be displayed and changed in this list.
	Configure units	Select **Configure units** to open the **Configure units of the object** window in the working area. You can configure the units used for the selected object here.

Cam track context menu

Grayed-out functions in the context menu cannot be selected.

You can select the following functions:

Table 4-180 Cam track TO context menu

Function		Significance/Note
Open configuration		Select **Open configuration** to display the window for configuring the cam track in the working area. Enter the configuration data for the cam track in this window.
Expert list		Select **Expert list** to open the expert list for the highlighted cam track. The configuration data and system variables can be displayed and changed in this list.
Cut		Select **Cut** to remove the selected object and save it to the clipboard.
Copy		Select **Copy** to copy the selected object. It is stored in the clipboard.
Paste		Select **Paste** to insert the cam track stored in the clipboard.
Delete		Select **Delete** to delete the selected cam track. The entire data of the cam track is deleted permanently.
Rename		Use **Rename** to rename the object selected in the project navigator. Note that with name changes, name references to this object are not adapted.
Expert		
	Insert script folder	**Insert script folder** enables you to insert a folder below the TO. You can create scripts in this folder in order to, for example, automate the configuration.
	Import object	**Import object** imports the data of a SIMOTION object from another project which was previously created with a selective XML export. You cannot import the entire project, only the data of the SIMOTION object.
	Save project and export object	**Save project and export object** exports selected data of the selected object in XML format. This data export can then be reimported into other projects. Only the data of the selected object, not the entire project, is exported.
Print		Select **Print** to print the data of the cam track. All system variables and configuration data with the associated values are printed.
Print preview		Select **Print preview** to open the preview of the output cam data to be printed.
Default		Select **Default** to define the system variables (e.g. track data and output cam data) of the cam track.
Properties		Select **Properties** to display the properties of the cam track highlighted in the project navigator.

4.3.5 Measuring Input TO - Part III

4.3.5.1 Overview of Measuring Input TO

General information about the Measuring Input TO

The **Measuring Input** technology object is used for fast, accurate measurement of actual positions. This is achieved with hardware support (e.g. measuring input on the associated drive unit).

Measurement jobs are activated and configured using the functions of the Measuring Input TO.

Local and global measuring inputs

Depending on the hardware platform, local and global measuring inputs are available for the measuring tasks. Local measuring inputs are axis-related and are mainly implemented in the drive. The actual position value is measured.
Global measuring inputs can be freely assigned to the axes and add an internal time stamp to the measurement result for more precise determination of the axis positions.

Assignment to axes and encoders

The measuring input TO can be assigned to the following axes/encoders:

- Position, synchronized, or path axes
- External encoders
- Virtual axes (only global measuring inputs)

Note

It is not possible to assign a measuring input to speed-controlled axes.

Measurement once

A measurement job is started by a program command. When a signal edge is detected at the measuring input, the current position (for virtual axes, either setpoint or actual value, as required) is stored temporarily. When the current system cycle clock finishes (either Servo cycle clock or interpolator cycle clock), this value is available in a system variable for further processing in low-priority tasks.

Cyclic measurement (global measuring inputs only)

A measurement job is started by a program command. Up to two edges can be detected for each processing cycle clock of the measuring input TO (with onboard measuring inputs of SIMOTION D, CX32, CX32-2, CU310, CU310-2, CU320, CU320-2 max. of two edges for every three position control cycle clocks). These are stored in system variables and remain available until they are overwritten by more recent measurements. The measured values are detected continuously/ cyclically until the program command is deactivated. Cyclic measuring is only possible with global measuring inputs.

Measuring range

By specifying a measuring range, the validity of the measurement can be restricted to this range; the measurement will only be activated when the position lies within the measuring range.

One measuring input for more than one axis (as of V4.0)

By creating a measuring input with the "monitoring measuring input" property, measurements can be made with one measuring input simultaneously on more than one axis/external encoder.

4.3.5.2 Fundamentals of Measuring Input technology object

Measuring input types - local and global measuring inputs

Depending on the hardware platform (measuring input type), local and global measuring inputs are available for the measuring tasks. Compared to local measuring inputs, global measuring inputs have a greater range of functions and enable measurements to be made faster. During the configuration of a Measuring Input TO, you must consider which functions (measurement, interconnection options) are to the used. Depending on the requirements, you have to configure a local or a global measuring input.

Local measuring inputs

At a signal edge on the respective input, the actual values of a SIMOTION C230-2, C240, D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2 or on the drive (for example, SIMODRIVE 611U, MASTERDRIVES MC, SINAMICS) of a connected encoder are acquired with positional accuracy in order to determine lengths or distances. The device on which the measuring system is available, is used for the measuring.

The assignment of inputs is not fixed depending on the hardware and is performed in the SCOUT engineering system during configuration of the Measuring Input TO symbolically or via the HW address.

Local measuring inputs refer to the respective drive. Configuration usually takes place via drive parameters.

Global measuring inputs

With a signal edge at the relevant input, the current actual values of one or more encoders are measured using time stamp functionality with positioning accuracy in order to provide information for determining lengths or distances (possible with any encoders included in the project).

Each measurement result is assigned a very precise "internal" time stamp which is then used to determine the corresponding actual position in SIMOTION.

The assignment of inputs is not fixed depending on the hardware and is performed in the SCOUT engineering system during configuration of the Measuring Input TO symbolically or via the HW address.

Global measuring inputs support extended functionalities

- More than one measuring input on one axis/encoder, whereby these can be active simultaneously.
- More than one measuring input is assigned to one measuring input (monitoring measuring input).
  With this functionality, one measuring input can have a functional effect on more than one measuring input and therefore on more than one axis/external encoder.
- Cyclic measuring
- Measuring on virtual axes
- should be configured on the respective device. (I/O channel is configured as measuring input (**MI)**).

- can be assigned on the SIMOTION-CPU to an axis TO or external encoder TO.
- Configuration by way of symbolic assignment is possible.

If current controller clock cycles <> 125 µs are used, the parameter calculations of the drive must be taken over into the PG and the Fast IO configuration must be recreated when using global measuring inputs (for further information, see chapter entitled Current controller clock cycles <> 125 µs / use of output cams and measuring inputs in the TM15 / TM17 High Feature Terminal Modules Commissioning Manual)

Time stamp functionality

With global measuring inputs, a time value (time stamp) is stored with each measurement. In this way, the exact axis position can still be determined even with different propagation delays between the time of the measurement to the evaluation.

Hardware for measuring inputs

Description

Global measuring inputs are only supported by certain hardware (measuring inputs) (see table below).

Hardware for local and global measuring inputs

The following table provides an overview of which hardware supports local and which hardware supports global measuring inputs.

Table 4-181 Hardware for local and global measuring inputs

Hardware (measuring inputs)	Local measuring inputs	Global measuring inputs
TM15, TM17 High Feature	-	X
SIMOTION C240/C240 PN (B1-B4)	-	X
SIMOTION C230/C240 (M1, M2)	X	-
SIMOTION D4xx	X	X (as of V4.1)
SIMOTION D410-2	X	X
SIMOTION D4x5-2	X (X122/X132 only)	X (X122/X132/X142)
ET200xP TM Timer DI/DQ		X (as of V4.4 HFx)
SIMOTION CX32/CX32-2	X	X (as of V4.1 SP2)
SINAMICS S120 drive CU310, CU310-2, CU320, CU320-2	X	X (as of V4.1 SP2)
MASTERDRIVES MC	X	-
SIMODRIVE 611U	X	-

Hardware (measuring inputs)	Local measuring inputs	Global measuring inputs
ADI4, IM174	X	-
PROFIdrive units	X	-
IM174/ADI4	X	-

Quantity structures for hardware measuring inputs

Table 4-182 Measuring inputs - Overview of quantity structures and functionality

Maximum available quantity structure			
	Maximum number of measuring input inputs	Can be configured as a local measuring input	Can be configured as a global measuring input
CU310, D410, CX32	3	X	X
D4x5, CU320	6	X	X
D4x5-2	16	Max. 8	Max. 16
CU320-2, CU310-2, D410-2, CX32-2	8	X	X
CX32-2	4	X	X
C230-2	2	2 (M1, M2)	-
C240	6	2 (M1, M2	4 (B1-B4)
C240 PN	4	-	4 (B1-B4)
TM15	24	-	X
TM17 High Feature	16	-	X
ET200SP TM Timer DI/DQ 10x24V	4	-	X
ET200MP TM Timer DI/DQ 16x24V	8	-	X

Interconnections

The Measuring Input TO can be linked to all technology objects, such as Axis TO (positioning axis, following axis, path axis) and External Encoder TO.

A TO, such as an Axis TO, can be interconnected simultaneously with several Measuring Input TOs. The assignment can be configured.

With local measuring inputs, the measuring input TO is assigned symbolically or if <V4.2 as part of its configuration. The configuration specifies the number of the measuring input to be used and the number of the encoder on the assigned axis.

(See also Local measuring (Page 1950))

With global measuring inputs, the HW address is used for assignment to the measuring input.

(See also Overview (Page 1951))

With V4.2 and higher, **global measuring inputs** can also be configured with symbolic assignments. Address handling is thereby no longer required.

Measuring input connection options

Measuring input on C2xx controller, analog drive connected

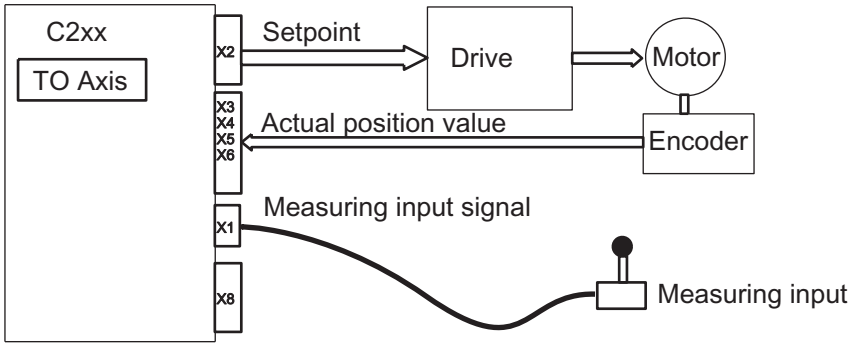


Figure 4-246 Connection of measuring input to the C2xx and an analog axis

Measuring input directly on drive, connected to SIMOTION via PROFIBUS DP

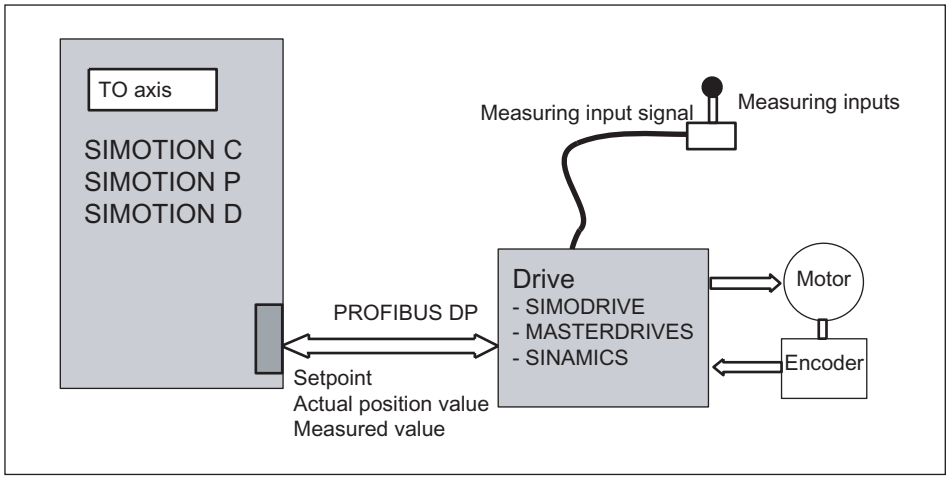


Figure 4-247 Connection of the measuring input on the drive, connected to SIMOTION via PROFIBUS

Measuring inputs on the digital onboard measuring inputs of SIMOTION D or on TM15/ TM17 High Feature

Measuring inputs on the digital onboard measuring inputs of SIMOTION D or on TM15/TM17 High Feature, linked to SIMOTION D via DRIVE-CLiQ.

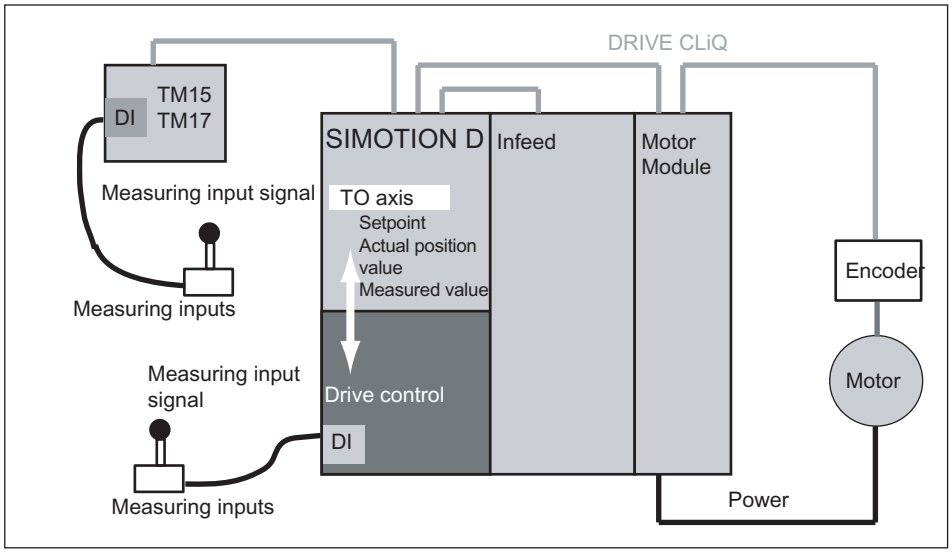


Figure 4-248 Connection of measuring input to SIMOTION D4x5 and TM15/TM17 High Feature

The TO Measuring Input cannot be interconnected with DP I/O or integrated I/O (with the exception of inputs of measuring inputs).

Measuring input at the measuring inputs on ET200xP TM Timer DI/DQ

Measuring input on the digital onboard measuring inputs of SIMOTION D or on an ET200xP TM Timer DI/DQ via PN on a SIMOTION D.

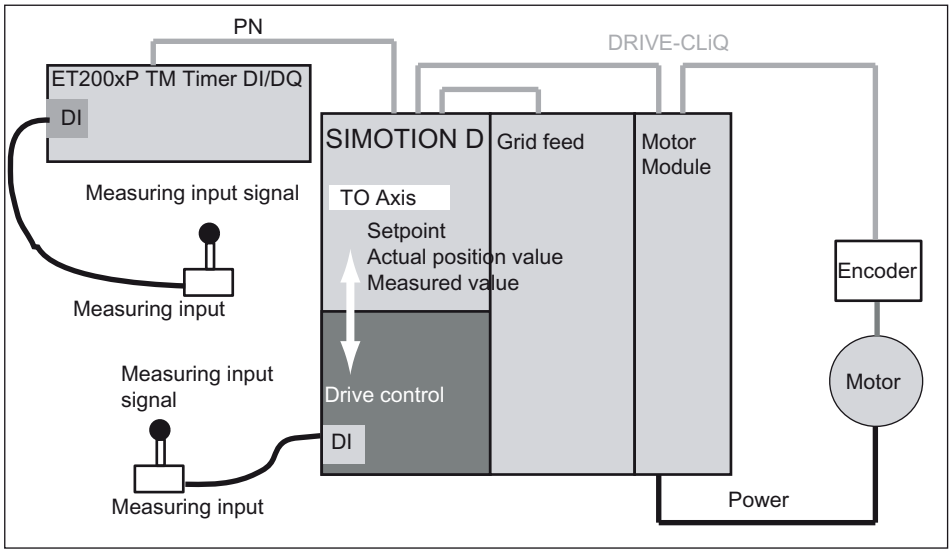


Figure 4-249 Connection of measuring input to ET200xP TM Timer DIDQ

Several Measuring Input TOs on one axis/encoder (as of V3.2)

Several Measuring Input TOs can be assigned to an Axis TO or an External Encoder TO. The number of measuring inputs is determined according to the functionality (local or global).

Local measuring inputs

For local measuring input (see Hardware for measuring inputs (Page 1920)) the following applies:

- Only **two** measuring inputs can be configured per axis TO or external encoder.
- Only **one** measuring input can be active on an axis TO or external encoder.

Global measuring inputs

For global measuring inputs (see Hardware for measuring inputs (Page 1920)) the following applies:

- **More than one** measuring input can be configured per axis TO or external encoder.
- **More than one** measuring input can be active simultaneously on an axis TO or external encoder.

As far as interconnections of measuring input TOs are concerned, **one** local measuring input can be configured on an onboard measuring input and **one** or **more** global measuring inputs can be configured and simultaneously active on measuring inputs of C240/C240 PN (B1-B4), D4xx, D410-2, D4x5-2 (X122/X132/X142), CX32, CX32-2, CU310/310-2/320/320-2 and TM15/TM17 High Feature, ET200xP TM Timer DI/DQ.

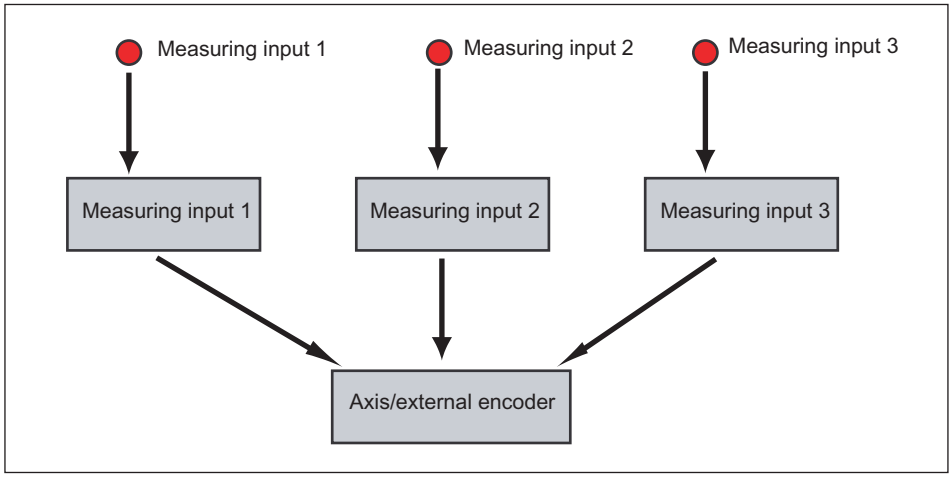


Figure 4-250 Example of interconnection of more than one measuring input with one axis or one external encoder

See also

Measuring input types - local and global measuring inputs (Page 1919)

More than one measuring input TO on a single measuring input (C230-2/C240 only)

It is possible to assign more than one measuring input to a single measuring input (onboard inputs C230-2/C240 (M1-M2)). In this case, however, it must be ensured that only one measuring input is active at any one time.

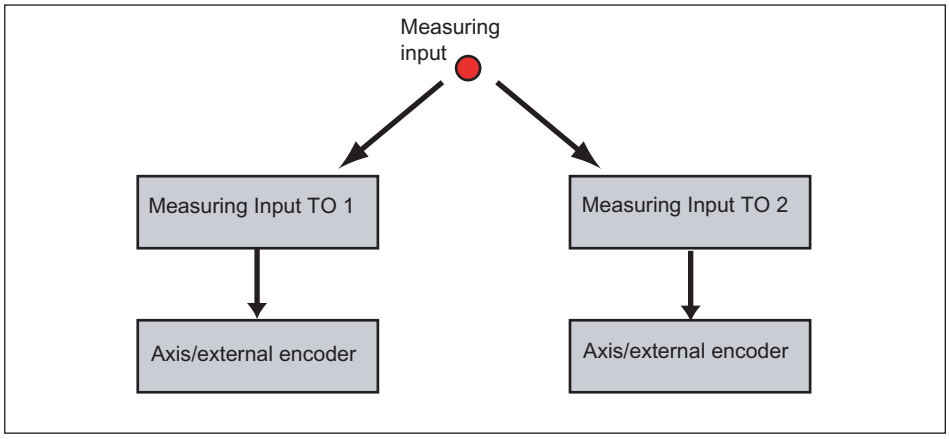


Figure 4-251 Interconnection of more than one measuring input with a single C230-2/C240 (M1-M2) onboard input

Measuring one measurement event on several axes - Listening measuring input (V4.0 and later)

With the "Monitoring measuring input" function, the measurement event of a measuring input can also be measured simultaneously by several Measuring Input TOs. To do so, an original measuring input (hereinafter referred to only as measuring input) is configured on a HW input. Additional Measuring Input TOs can be configured as monitoring measuring inputs and interconnected with the Measuring Input TO. The monitoring Measuring Input TOs "listen in" on the measurement event of the Measuring Input TO.

The monitoring Measuring Input TOs can also be assigned to other axes / external encoders. With this functionality, one measuring input can have a functional effect on several axes / external encoders.

The "listening in" property on the Measuring Input TO is set with the configuration data element **inputAccess**:=TO_INTERFACE.

Note

An event can only be measured on more than one axis/external encoder at the same time if the measuring input TO is configured as a global measuring input (see Hardware for measuring inputs (Page 1920)). Only these inputs have the necessary time stamp functionality.

Procedure

One measuring input is interconnected with one measuring input, as usual. This measuring input is the **original** measuring input. The measuring process is activated and the measurement events are measured on this measuring input. The monitoring measuring inputs are connected internally to the measuring input via an interconnection interface, and the measurement events are communicated simultaneously.

The activation and deactivation of the measurement, as well as the configuration of a measuring range can only take place on the measuring input. Activation and deactivation commands issued on the monitoring measuring input are not executed and are returned with errors. Technology alarm 40011 is issued.

Functionality

- Activation and deactivation of the measuring process on the measuring input only. These commands are not active on the monitoring measuring input.
- Measuring range and edge selection are only available on the measuring input.
- The monitoring measuring input must be configured correctly at the time of the measurement (measuring input cycle clock, system number).
- A monitoring measuring input does not have its own measuring input and cannot perform any measurements of its own.
- The processing cycle clocks of the measuring input and the monitoring measuring input do not have to have the same setting. However, accuracy is lost if the measuring input TO is assigned to the IPO and the monitoring TO measuring input is assigned to IPO_2, and IPO and IPO_2 are configured differently.

The following must be considered when interconnecting more than one measuring input:

- The measuring input and the monitoring measuring input are interconnected with one axis TO or external encoder.
- A measuring input can be interconnected with **more than one** monitoring measuring input **on the output side**.
- A measuring input has **no interconnection on the input side**.
- A monitoring measuring input can only be interconnected with **one** measuring input **on the input side**.
- A monitoring measuring input has **no interconnection on the output side**.
- An axis TO or an external encoder can be interconnected with more than one measuring input, including a mixture of measuring inputs and monitoring measuring inputs. Depending on the hardware (see Chapter **More than one measuring input on one axis/encoder (as of V3.2)**), more than one measuring input may be active simultaneously.

- Local measuring inputs (onboard I/O from SINAMICS and C2xx) cannot be used for the measuring input. Only global measuring inputs can be used (see Hardware for measuring inputs (Page 1920)).
- Only single-stage interconnections are possible.

Note

You will find more information on interconnections in the *SIMOTION Runtime Basic Functions* function manual.

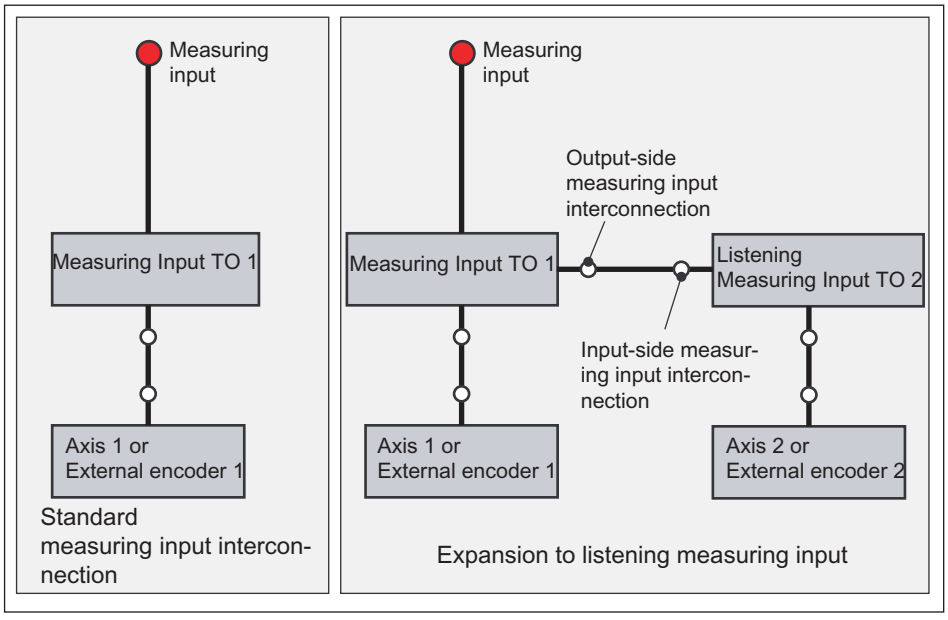


Figure 4-252 Interconnecting monitoring measuring inputs

See also

Several Measuring Input TOs on one axis/encoder (as of V3.2) (Page 1923)

Measuring input types - local and global measuring inputs (Page 1919)

Measurement

The measuring process is divided into one-time and cyclic measurement modes. The operating mode is distinguished using two separate program commands.

Table 4-183 Overview of one-time and cyclic measurement functions

Measurement once	Cyclic measurement (global measuring inputs only)
As of version V1.0	As of Version V3.2
Call up with **_enableMeasuringInput** command.	Call up with **_enableMeasuringInputCyclic** command.
Measurement jobs must be issued individually for each measurement.	Measuring is activated just once and runs cyclically until deactivated with **_disableMeasuringInput**.

Measurement once	Cyclic measurement (global measuring inputs only)
Several interpolation cycle clocks between two measurements	Up to two edges can be measured in each processing cycle clock of the measuring input TO (IPO interpolation cycle clock, IPO_2 interpolation cycle clock or position control cycle clock). The minimum distance between two measurements in SIMOTION D410-2, D4x5, D4x5-2 (X122/X132) onboard, CX32, CX32-2 and CU310/310-2/320/320-2 is 3 position control cycle clocks. The measured values must be read from the user program before they can be overwritten by a new measurement.
Supported by: For hardware for local and global measuring inputs, see Hardware for measuring inputs (Page 1920)	**Supported by:** For hardware for global measuring inputs, see Hardware for measuring inputs (Page 1920) (excluding TM15)
Optional measuring range is possible.	One measuring range can be defined as of V4.0. Prior to V4.0, measuring range for TO cannot be configured (must be solved in the application).

One-time measurement

In one-time measurement mode, the measured result must be waited for. On its occurrence, the measuring process is terminated.

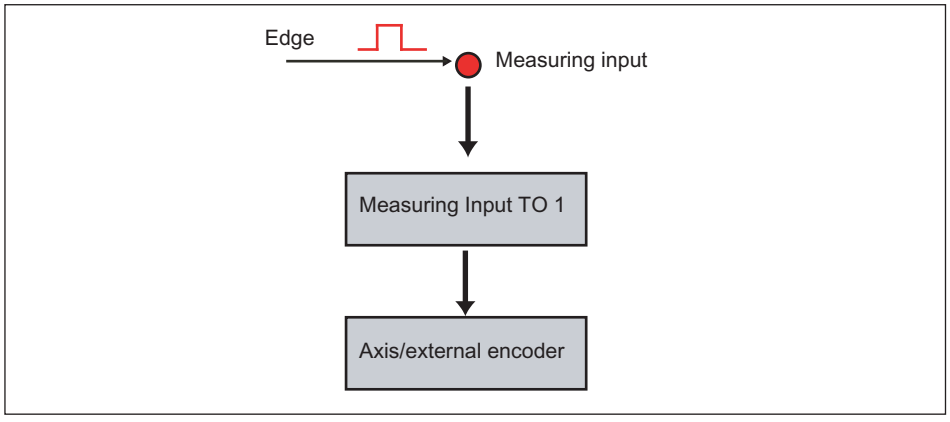


Figure 4-253 TO measuring input, one-time measurement

Measurement process

A measurement is activated by the **_enableMeasuringInput** program command. The **enableCommand** variable indicates the execution status of this command.

The **control** system variable indicates whether the measuring function is active. If, for example, a positive measured edge has been selected and the measuring input is deflected, i.e. a measuring event occurred, the system variable cannot assume the value ACTIVE until the measuring input is no longer deflected.

The **state** variable is set to the value WAITING_FOR_TRIGGER. A rising edge (from 0 to 1) or a falling edge (from 1 to 0) triggers the measuring function. The **measuredEdgeMode** parameter can be used to select which type of edge will be acquired. Acquisition of both measured edges can also be activated by means of a measurement job, in which case you can specify in the command which edge will be acquired first, for example, first the rising edge and then the falling edge.

The drive must be capable of evaluating the signal edge (rising, falling, or both edges) selected by SIMOTION at the measuring input.

When the measurement result is received, the measurement position is stored. Once the measurement has been made, the **state** variable is set to TRIGGER_OCCURED, and the measured values can be evaluated using the **measuredValue1** and **measuredValue2** variables for two measured edges.

Activation/deactivation of measurement job

The measurement job remains active until the measurement result has been obtained or until the job is terminated by a command (e.g. **_disableMeasuringInput**).

The measuring process must be reactivated for each new measurement.

The measuring accuracy depends on the accuracy of the hardware used. It lies in the range of microseconds.

Table 4-184 Storing the measurement results in the system variables

Edges per processing cycle of the TO measuring input	measuredValue1	measuredValue2	Remark
Rising edge	First rising edge	---	Only the first rising edge is acquired (each new edge requires a new measurement job).
Falling edge	First falling edge	---	Only the first falling edge is acquired (each new edge requires a new measurement job).
Both edges	Chronologically the first edge	Chronologically the second edge	The hardware used must provide the possibility of evaluating the edge selected with SIMOTION (rising, falling, or both edges) at the measuring input (e.g. for SIMOTION D, SINAMICS S120 CU320(-2), TM1x).
Both edges, rising edge first	First rising edge	First falling edge	TM1x; D4xx, D4xx-2 (X122/X132), CX32, CX32-2, CU320, CU320-2: The minimum distance of the first edge to be measured from the previous edge must be several servo cycles so that the first edge can be acquired. D4x5-2 (X142): No restrictions With hardware support, the "false" first edge is ignored.
Both edges, falling edge first	First falling edge	First rising edge	

Cyclic measurement (as of V3.2)

In cyclical measurement, up to two edges can be measured per processing cycle of the measuring input TO IPO interpolation cycle clock, IPO_2 interpolation cycle clock, and servo cycle clock in the case of TM17 High Feature, D4x5-2 (X142) and C240 (B1-B4).

In the case of D435-2 DP/PN, D445-2 DP/PN, and D455-2 DP/PN, it is also possible to configure a processing cycle clock in Servo_fast or IPO_fast for the measuring input.

In the onboard measuring inputs of D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310/310-2/320/320-2, the minimum duration between two measurements is three servo cycles. Measuring signals in between are not detected.

Measurements are taken cyclically, until they are terminated with a command.

The measured values must be read from the user program in good time before they are overwritten by a new measurement.

Note

Cyclic measuring is only possible with global measuring inputs (see Hardware for measuring inputs (Page 1920)(excluding TM15)).

You are not prevented from also using local measuring inputs for cyclic measuring during configuration. However, this is a pure software solution, and only unique measuring processes are cyclically collected. The minimum time between two measurements is therefore much longer than with cyclic measuring with global measuring inputs.

When cyclic measurement is activated with local measuring inputs (command **_enableMeasuringInputCyclic**), as of V4.2 **Note 40014** is output stating that during cyclic measuring with local measuring inputs, results may be lost if they are not spaced sufficiently.

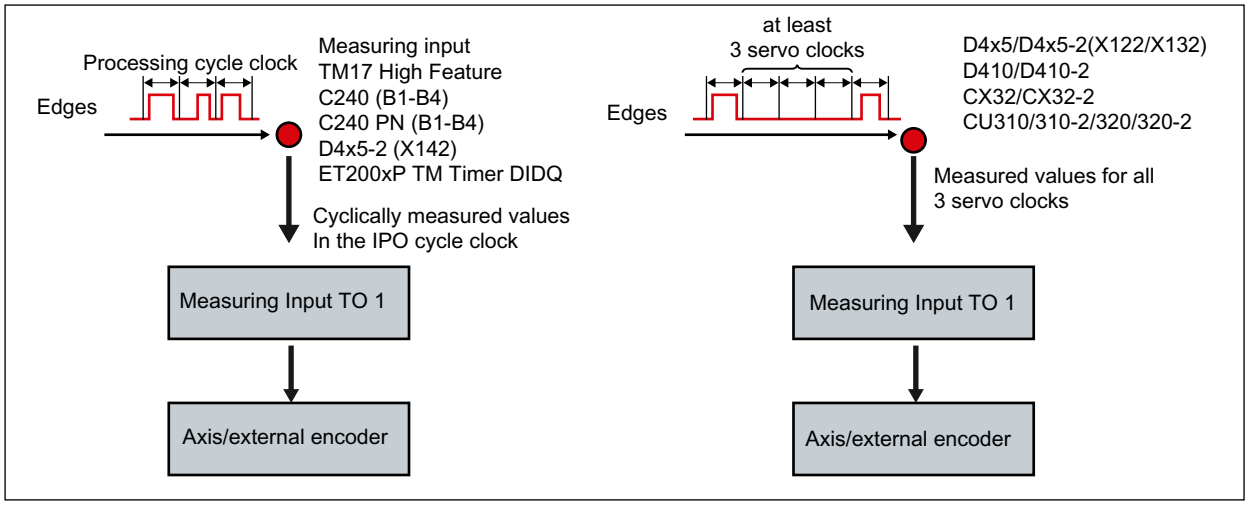


Figure 4-254 Measuring input TO, measuring operation cyclic measurement (example processing cycle clock IPO, no cycle clock scaling IPO:Servo, IPO = Servo)

Measurement process

A measurement is activated by the **_enableMeasuringInputCyclic** program command. The **cyclicMeasuringEnableCommand** variable indicates the execution status of this command.

The **control** system variable indicates whether the measuring function is active. Measuring begins on activation of the program command.

The **userdefault.measurededgecyclicMode** system variable is used to select which type of edge should be acquired. Up to two edges can be measured per processing cycle of the measuring input TO (e.g. IPO interpolation cycle clock, IPO_2 interpolation cycle clock, or position control cycle clock).

The measured values are taken over in the measurement equipment at time Ti and placed in the telegram to the control. After that, the measurement equipment is ready again. In the telegram, two edges per servo can be transmitted to the control. These are provided in a buffer and processed by the measuring input TO in its processing cycle clock.

With cyclic measuring, the **state** variable remains in the WAITING_FOR_TRIGGER state even after the arrival of events, as waiting is continued for further events. Increases on the **countermeasuredvalue1/2** event counter indicate that a measurement event has occurred.

When the measurement event occurs, the measurement position is stored. After successful measurement, the measured values are stored in the **measuredValue1** and **measuredValue2** system variables and can be evaluated. The measured values must be read from the user program before they can be overwritten by a new measurement. For example, if polling is being performed in the IPO-synchronous task, up to two edges can be evaluated per IPO cycle clock.

Table 4-185 Archiving measurement results in the system variables within an IPO cycle clock

Edges per processing cycle of the measuring input TO	measuredValue1	measuredValue2	Description
Rising and falling	First rising edge	First falling edge	- If both edges are to be measured, the measured values of the rising edges are in **measuredValue1**; those of the falling edge in **measuredValue2**, irrespective of the order in which they occur.
Rising only or falling only	First edge	Second edge	-
One edge only	First edge	-	-
More than two edges	First edge	Second edge	- Error status is synchronously reported with the **error** system variable, and the 40009 TO alarm is output.

If the measuring equipment only reports one value in a position control cycle, only one value will be forwarded to the TO and this overwrites the one matching **measuredValue** and increments the counter.

The error condition "more than two edges" signaled via system variable and TO alarm can only occur if scaling of the servo down to the processing cycle clock of the measuring input TO is configured, that is, if more than two edges arrive within one TO processing cycle but across several servo cycles. If more than two edges occur within one servo cycle, the additional edges are already discarded in the measuring equipment. The technology object does not contain any information about this.

counterMeasuredValue system variable

The **counterMeasuredValue1** and **counterMeasuredValue2** counter variables are defined for the **measuredValue1** and **measuredValue2** system variables and are automatically incremented by a value of one for each measuring input. In this way, new events can be traced

immediately, e.g. even if the position value happens to be identical with the previous measurement.

Measuring events can also be read out from tasks that are not synchronous with the processing cycle clock. A measuring event that is not fetched from the user program in time can be detected by larger increments in the counting variables.

The counting variable counterMeasuredValue1 is incremented by "1" each time a new measurement input has been made in **measuredValue1**.

The counting variable counterMeasuredValue2 is incremented by "1" each time a new measurement input has been made in **measuredValue2**.

Measured values lost due to too many edges per unit time cannot be indicated via **measuredValue1** and **measuredValue2**. They are therefore also not counted in the counting variables **counterMeasuredValue1** and **counterMeasuredValue2**.

Counter variables are reset on power-up, reset, restart, and on first activation of cyclic measuring. Counter variables are not reset if cyclic measuring was already active and, for example, only a parameter was changed with the **_enableMeasuringInputCyclic** command.

Activation/deactivation of measurement job

The measurement job remains active until it is deactivated with the **_disableMeasuringInput** command.

See also

Measuring input types - local and global measuring inputs (Page 1919)

Lost edges during cyclic measurement

Lost edges

Lost edges are edges that did exist at the measuring input (after filtering) and were also to be acquired according to the configuration (e.g. only rising edges), but were not signaled via the system variables **measuredValue1** and **measuredValue2**.

Lost edges do not result in incrementation of the counting variables **counterMeasuredValue1** and **counterMeasuredValue2**.

Edges that are filtered out, for example, at X142 via the settable digital filter do not exist for the measuring input logic.

Edges that are not "seen," for example, because the sensor is dirty, are not acquired in any way.

Measured values

Measured values are only acquired for the edges that are also to be calculated according to the configuration of the measuring input TO, that is, optionally

- 2 rising edges,
- 2 falling edges or
- one rising and one falling edge.

During parameterization, generate "only rising edges," that is 2 print marks (comprising 2 rising and 2 falling edges) in total 2 measured values.

Edges can be lost in 2 ways:

1. D4x5-2 (X142), TM17 High Feature and C240 (B1-B4) can acquire up to 2 measured values per position control cycle. If more edges occur that would result in a measured value, these will already be lost in the measuring equipment.
2. Possible scaling of "servo cycle clock" down to "TO processing cycle clock" can result in the measurement hardware communicating the measured values to the measuring input TO, but these not all being made available to the user interface. Only the system variables **measuredValue1** and **measuredValue2** exist.
   Example:
   X142: Servo cycle clock = 1 ms; processing cycle clock measuring input TO in the IPO = 2 ms.
   2 x 2 edges occur in the 2 servo cycles within one IPO cycle.
   ⇒ Of the 2 x 2 edges, only the **first** two edges are entered as measured values

In the case of the onboard measuring inputs of D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2,

CU310/310-2/320/320-2, the minimum period between two measurements is 3 position

control cycles. Measuring signals in between are not detected.

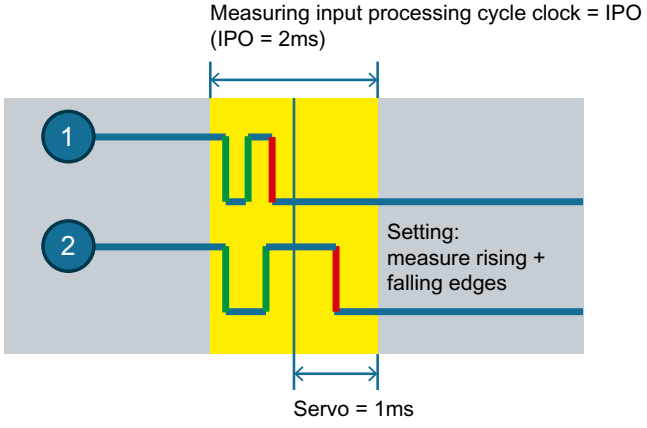


Figure 4-255 (1): Edge lost in measurement hardware(2): Edge lost due to cycle clock scaling

Case 1: Loss of edges in measuring equipment

Edges that result in more than 2 measured values per position control cycle are lost in the measuring equipment. Loss of these edges is not signaled via system variable **error** and TO alarm 40009.

In the case of X142 and ET200xP TM Timer DI/DQ, there is a separate edge counter LEC for lost edges.

Case 2: Loss of edges due to scaling of servo cycle clock down to TO processing cycle clock

Only if the "servo cycle clock" is scaled down to "TO processing cycle clock," can lost edges be signaled via TO alarm.

Example:

Processing cycle clock measuring input TO in the IPO = 2 ms

Servo cycle clock = 1 ms

In this case, the measuring input can transfer up to 2 measured value per 1 ms to the TO (that is, up to 4 in total). The TO can, however, provide the user program with only the first two measured values via the system variables **measuredValue1** and **measuredValue2**. **counterMeasuredValue1** and **counterMeasuredValue2** are each incremented by "1".

In this case, the error condition is signaled synchronously via system variable error and TO alarm 40009 is output.

Behavior of the count variables counterMeasuredValue1/2

Lost edges can result in the counters counterMeasuredValue1 and counterMeasuredValue2 diverging.

Example: Measurement of rising and falling edges

Processing cycle clock measuring input TO	Edges	counterMeasuredValue1	counterMeasuredValue2
N		:	:
N+1		6	6
N+2		6	6
N+3	rising-falling	7	7
N+4	rising	8	7
N+5	falling	8	8
N+6	rising-falling-*rising*	9	9
N+7	falling	9	10
N+8	rising-falling-*rising*	10	11
N+9	falling	10	12
N+10	rising-falling-*rising*	11	13
N+11	falling	11	14

**Italics: Edge is lost*

Hardware-specific aspects

X142 and ET200xP TM Timer DI/DQ / Lost Edge Counter

X142 and ET200xP TM Timer DI/DQ have a counter for lost edges (Lost Edge Counter). Up to 7 lost edges are recorded in one counter. The counter value is transmitted cyclically and can be symbolically interconnected in the controller. (LEC). Because the LEC only counts lost edges, for

example, only the not measured "rising" edges are counted in measuring mode "only rising edges."

The LEC only counts the edges that are lost on the X142 or ET200xP TM Timer DI/DQ. Edges lost due to scaling of the servo cycle clock down to TO processing cycle clock are not acquired by the LEC.

The edges lost on the measuring equipment are only signaled via the LEC. That is, they are not signaled via system variable/TO alarm.

Via an I/O variable, it is possible to access LEC symbolically and evaluate it in the user program.

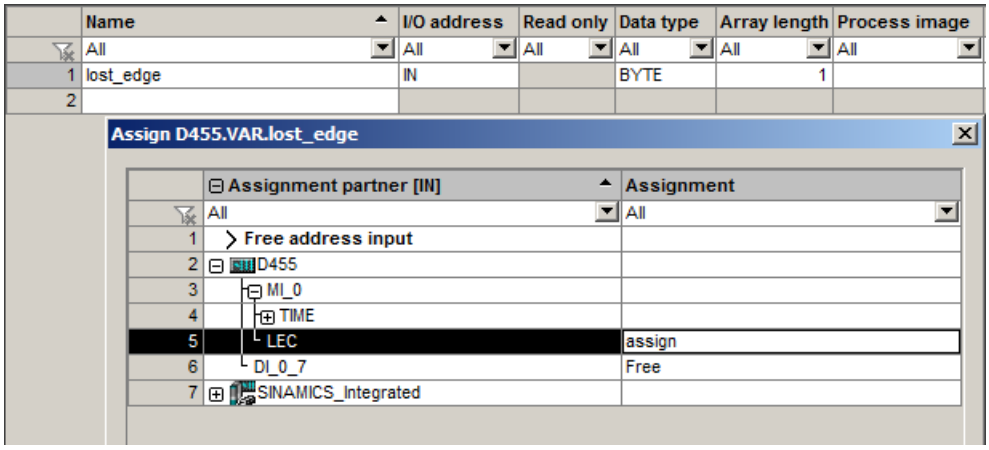


Figure 4-256 Accessing I/O variables on LEC

The value in LEC is automatically reset during cyclic measurement, und the current measured values taken over from the measuring equipment to be transmitted to the controller and the measuring equipment is automatically reactivated for the next cycle.

If the servo cycle clock is scaled down to the TO processing cycle clock, the value must be fetched in good time; otherwise, it will be overwritten.

Example:

Measuring input processing cycle clock: IPO = 2 ms

Servo/bus cycle clock: 1 ms

⇒ the I/O variable assigned to the LEC must be evaluated in the servo-synchronous user task.

TM17 High Feature

If no edges are measured in a cycle, the TM17 High Feature can measure up to 4 edges in the following cycle. In this case, the 3rd and, if necessary, 4th edges are buffered (buffering of up to 2 edges).

In the following cycle, the buffered edges are transferred first, even if new edges have been acquired; these new edges are buffered again.

You will find further information in chapter *System behavior during cyclic measurement* in the Terminal Modules TM15 / TM17 High Feature Commissioning Manual.

Measurement activation times

Various response times (e.g., effects of propagation delays) must be taken into account in the application for the measuring function, depending on the axis/external encoder connection (Onboard C2xx, PROFIBUS axis), the drive used (611U, MASTERDRIVES MC, SINAMICS), and the execution level (IPO/IPO_2, or position control cycle clock).

The measuring process is started with **_enableMeasuringInput** or **_enableMeasuringInputCyclic** in the user program.

The runtime up to the evaluation of the measured edge at the HW input is dependent on the configuration. In order to detect the measured edge correctly, you must ensure in the user program that **_enableMeasuringInput** or **_enableMeasuringInputCyclic** have been executed prior to this runtime.

Utilities & applications

The SIMOTION Utilities & Applications contains examples of and help for SIMOTION. It serves to support SIMOTION users and clarify SIMOTION applications.

SIMOTION Utilities & Applications includes, for example, a tool to estimate:

- The time elapsed from activating the **_enableMeasuringInput** and **_enableMeasuringInputCyclic** command to the measuring input job becoming effective in the drive
- The minimum time between two measurement jobs
- Derivative action time when applying a measuring range (fine range)

The SIMOTION Utilities & Applications is shipped with the SIMOTION SCOUT software package.

Measuring range

A measuring range (system variable **userdefault.measuringRangeMode**) can be predefined for a measurement job. This can either be valid for the whole range, or restricted by entering a measuring range start and end.

Note

The measuring range can be used for single measurement and, as of V4.0, for cyclic measurement. If a measuring range is required for cyclic measurement (prior to V4.0), this can be implemented in the application.

A user program in the IPO-synchronous task can detect up to two edges in each IPO cycle clock. If an "unintended edge" is detected, it simply has to be rejected.

Measuring process with measuring range

When measuring with a measuring range, initially, **_enableMeasuringInput** or **_enableMeasuringInputCyclic** simply records the measurement job in the system. The measurement will then be triggered only when the axis position lies within the measuring range. Measuring is only valid within this range.

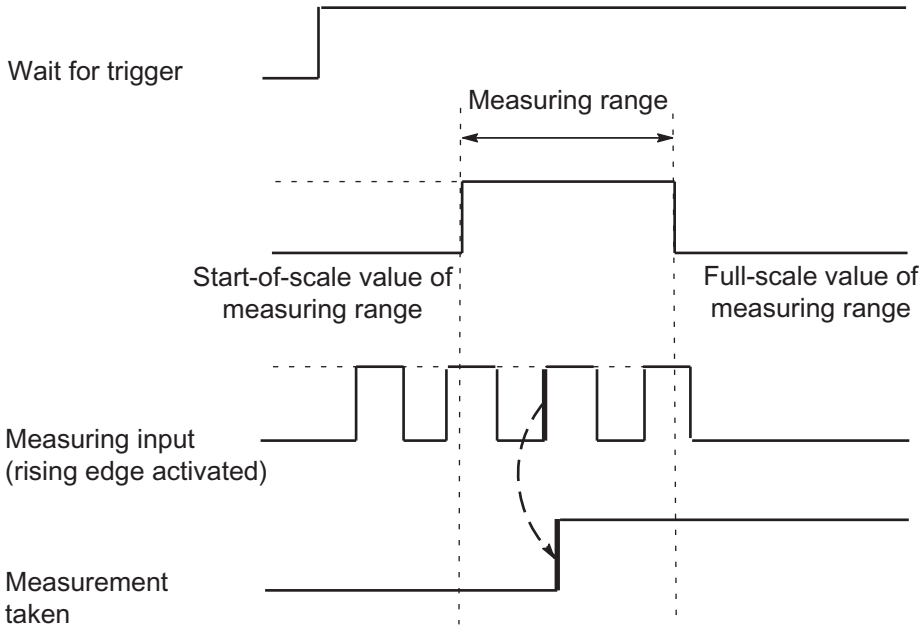

Wait for trigger
Measuring range
Start-of-scale value of measuring range
Full-scale value of measuring range
Measuring input (rising edge activated)
Measurement taken

Figure 4-257 Measuring in the measuring range

The time that elapses after reaching the start of the measuring range on the axis (mechanics) until the measured edge is evaluated at the HW input is dependent on the configuration. In order to activate the measuring function on reaching the desired start of the measuring range on the axis (mechanics), you must preset the start of the measuring range with the **_enableMeasuringInput** or **_enableMeasuringInputCylic** command, depending on the axis velocity and response time.

This procedure is the same for the end of the measuring range. Here, it is important that the measuring function is deactivated as soon as the end of the measuring range on the axis (mechanics) is crossed.

If no measured edge is detected in the measuring range during a single measurement, the measuring job is aborted and a TO alarm is triggered. During cyclic measuring, each measurement result in the measuring range is reported. However, a TO alarm is not output if no measured edge is detected in the measuring range.

During cyclic measuring, the measurement remains active for modulo and non-modulo axes even after the measuring range is exited, and even over several modulo cycles. The measurement is only terminated with **_disableMeasuringInput**.

For non-modulo axes, the sequence in which the start and end of the measuring range are specified is irrelevant. If the start of the measuring range is greater than the end of the measuring range value, the two values are exchanged.

If the start of the measuring range is greater than the end of the measuring range in a modulo axis, the measuring range is extended from the start of the measuring range over the modulo transition of the axis to the end of the measuring range.

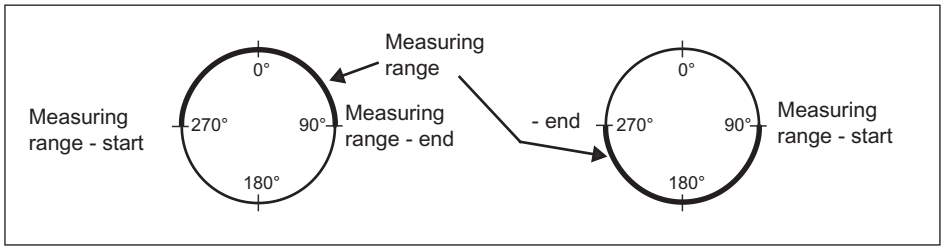


Figure 4-258 Measuring range for modulo axes for the measuring input

Dynamic measuring range

The measuring range can also be switched on or off dynamically. An activation time is allowed for this purpose when switching on or off.

The activation time can be specified by means of the **measuringRange.activationTime** configuration data element.

This time can be used, for example, to compensate for the runtimes during activation via PROFIBUS and the drive.

The following applies here:

- Activation of the measuring range and consideration of the activation time are performed on a position control granular basis.
- This means that the accuracy depends on the position control cycle clock.

- The runtime should be taken into consideration: Position control DP cycle clock activation in the drive (i.e. the position of the position control cycle clock in relation to the DP cycle clock as well).
- The actual position of the encoder is not evaluated directly, instead the filtered actual position is evaluated (see Actual value system).

Configure Units

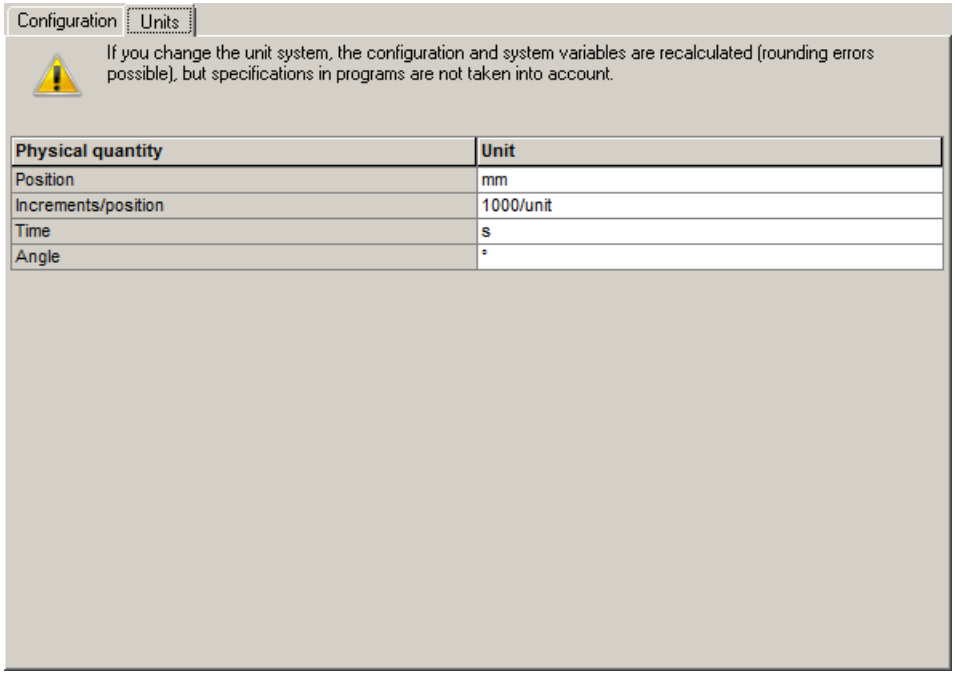
You can define the basic units for each technology object. The same physical variables can have different units in different technology objects. These are converted:

How to configure the units:

1. In the project navigator, open the context menu for the technology object.
2. In the context menu, select **Expert > Configure units**. The **Configure Units** window appears in the working area.
3. Select the **unit** for the **physical variables**. These units are used for the technology object, e.g. s for time units.

or

1. In the project navigator, open the **Configuration** under the TO.
2. Select the **Units** tab.



You can set the following parameters:

Field/button		Meaning/instruction
Table with units		
	Physical variable column	Shows the physical variable. The physical variables which are used by the TO are available for the configuration.
	Unit column	Displays and configures the unit. A drop-down list for selecting the unit appears when you click on the cell.
Toolbar		
		Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display
	Close	Button for closing the dialog.
	Help	Button for opening the online help for the dialog.

Simulation

This function activates the measuring input simulation (**simulation**=active). **Measured result arrived** is set and allocated to the programmed measured value as a measured result.

If the simulation mode is active on the measuring input, the simulated measured value is entered in **Measuredvalue1** using the function **_enableMeasuringInput** (trigger), and **state**=trigger_occured is set.

4.3.5.3 Configuring the Measuring Input technology object

Inserting Measuring Inputs

Note

Before you insert a measuring input, the hardware must be configured and the axis or external encoder to which the measuring input cam is assigned has to be created.

To insert a new measuring input

1. In the project navigator, highlight the **MEASURING INPUTS** folder under the relevant axis or external encoder.
2. Select **Insert > Technology Objects > Measuring Input** , or double-click **Insert Measuring Input** in the project navigator at the axis or external encoder entry in the MEASURING INPUTS folder. The **Insert Measuring Input** window appears.

3. Enter a **name** for the measuring input. You can also enter a **comment**.
   Names must be unique throughout the project. For this reason, all the inserted measuring inputs are displayed under **Available measuring inputs**.
4. Confirm with **OK**. In the working area, the window for the configuration is displayed and the measuring input created is shown in the project navigator.

Parameterization of the Measuring Input technology object

General information about configuration data and system variables

Two data classes are distinguished when parameterizing a TO.

Configuration data defines the principal functionality of a TO. They are set within the object configuration framework with the SCOUT engineering system and are not normally changed during runtime.

System variables provide status data of the TO for the user program and a parameterization interface on the TO. System variables can be changed during runtime.

Note

You will find more information on technology objects in the *SIMOTION Runtime Basic Functions* functional description.

To parameterize a measuring input:

1. In the project navigator under the **MEASURING INPUTS** folder, find the measuring input technology object (TO) that you want to parameterize. Double-click the measuring input to display the associated objects.
2. Double-click **Configuration** or **Default** in the project navigator. The window appears on the workspace.
   - **Configuration** (see chapter **Configuring a measuring input**):
     Define the **configuration data** of the measuring input here. This includes, for example, the processing cycle clock.
   - **Default** (see chapter **Measuring input defaults**):
     Define the measuring input defaults of the **system variables** here. These include the edge, start of measuring range, and end of measuring range.
3. Change configuration data and measuring input defaults.
4. Click **Close** to accept the changes.
5. Repeat steps 2 to 4 for all objects in which you want to change the configuration data and defaults.

See also

Measuring Input Configuration (Page 1942)

Measuring input defaults (Page 1947)

Use Expert List for Measuring Inputs

Parameters required for standard SIMOTION applications (configuration data and system variables) are parameterized into the output cam technology object directly by means of screen forms or are defined automatically.
It may be necessary to change automatically defined parameters for special SIMOTION applications. These configuration data and system variables can only be displayed and changed in the expert list.

Note

You will find more information on working with the expert list in the *SIMOTION Runtime Basic Functions* functional description.

Measuring Input Configuration

Measuring Input Configuration

In the **Configuration** window, define the configuration data values for the measuring input.

Double-clicking in the project navigator below the measuring input on the **Configuration** element displays the window in the working area.

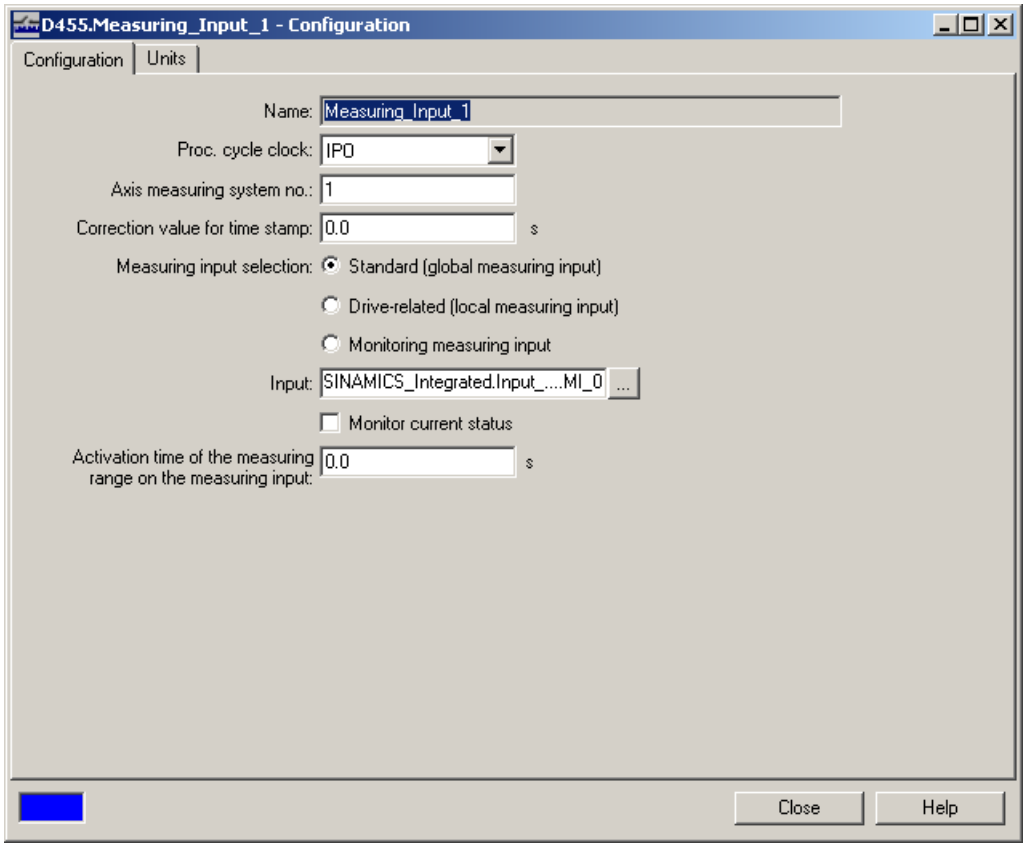


Figure 4-259 Global measuring input

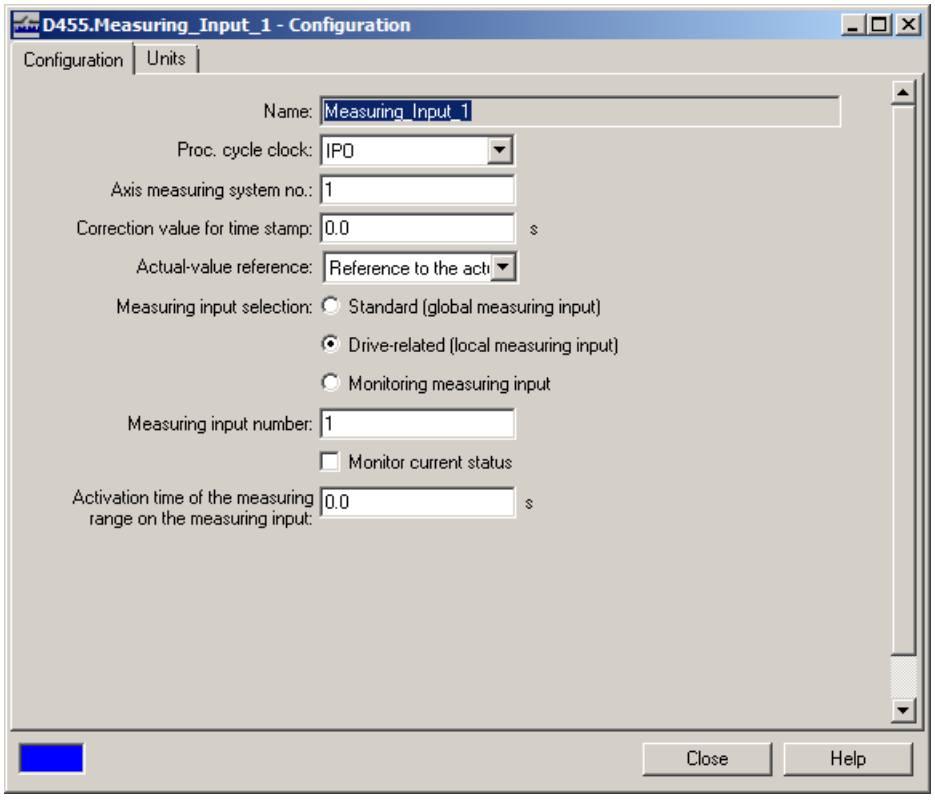


Figure 4-260 Local measuring input

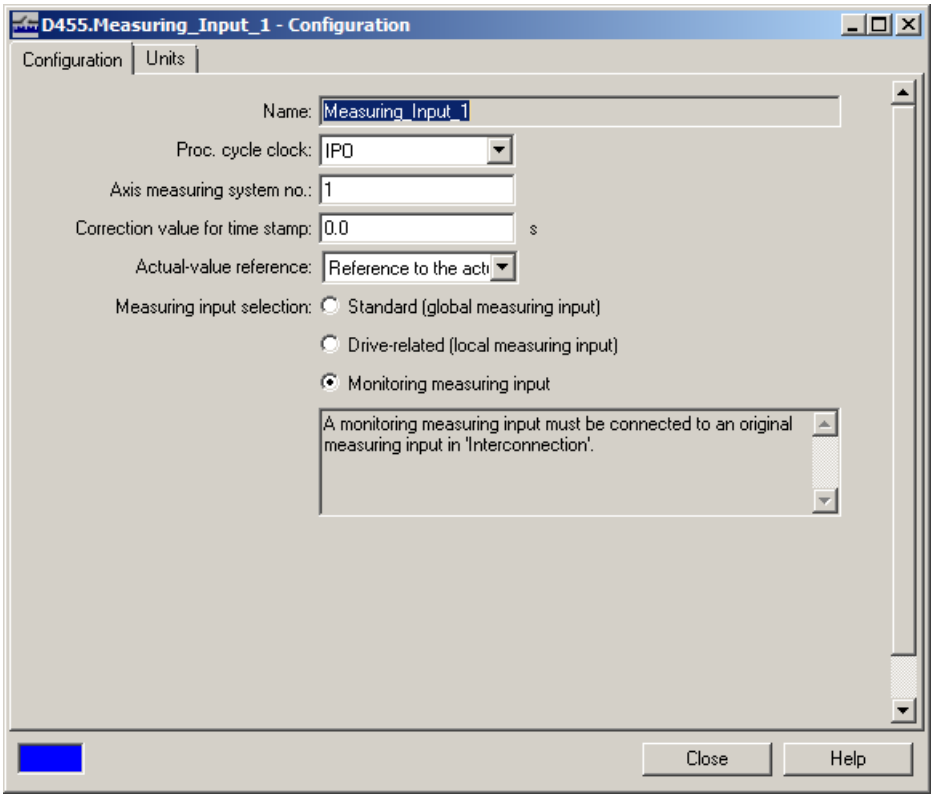


Figure 4-261 Monitoring measuring input

You can set the following parameters:

Table 4-186 Measuring input configuration data

Field/button	Meaning/information
Name	The name of the created measuring input is displayed here.
Processing cycle clock	Use this to select the system cycle during which the measurement result is interpolated and stored in the system variables. **IPO** (Default value) The measurement result is refreshed in interpolation cycles. **IPO_2** Measurement result is refreshed in the interpolation cycle clock 2. The IPO_2 cycle clock length is at least twice that of the IPO. **Servo** The measurement result is refreshed in position control cycles. For the possible setting of IPO_fast and Servo_fast with D435-2, D445-2 and D455-2, see Section Second servo cycle clock (Servo_fast) in the SIMOTION Runtime Basic Functions Manual.

Field/button	Meaning/information
Axis measuring system no.	Under **Axis measuring system no.**, enter the number of the encoder system used on the measuring input (if the axis has more than one encoder). Encoder system 1 is the default setting. An encoder system can be assigned to several measuring inputs. **Note:** With global measuring inputs, please note that measurement is not based on the encoder which is active for the control/Ipo, but on the axis measuring system set on the measuring input. **Note:** With virtual axes, the axis measuring system number is irrelevant because only setpoints are measured.
Time stamp correction value (Page 1946)	Enter the correction time for the evaluation of the time stamp. The correction value will be saved in the configuration data element **TimeStampConfig.correctionTime**. Each measuring result is assigned a time stamp. **Delay times can be corrected in the correction time**: • Delays that occur directly in the measuring input, e.g. the delay caused by mechanical deflection of the probe or the time required to generate the measuring signal before it is transferred to the measuring module; • Switching times in the sensing system, e.g. filter times at the input. When measuring actual values (reference to actual values at the encoder), the delays that occur between sensing of the measurement event and its communication to the SIMOTION system are automatically corrected by interpolation (delays based on bus and cycle clock system). **Special case: Measurement on virtual axis with reference to actual value of a real axis**: When measurements are taken on virtual axes, the actual value matches the setpoint and no delays occur. For example, if you program a measurement for the actual value of an axis in relation to a virtual axis (e.g. master axis), a correction value for the offset time must be taken into account so that the position can be calculated correctly. If the correction value is not taken into account, the measured value position can deviate with different cycle clock settings / velocities.
Actual value reference	Depending on the following settings, you select: • **Reference to the actual value on the encoder without considering Ti**. Reference to the actual position in the controller before the position filter. • **Reference to the actual value after the position filter**. Reference to the actual position in the controller after the position filter. • **Reference to the actual value on the encoder. Ti is taken into account**. Reference to the actual value on the encoder module / drive. The transmission time Ti from the encoder module / drive to the controller is taken into account by the system.
Measuring input selection	

Field/button		Meaning/information
	Standard (global measuring input), e.g. on the TM15/ TM17 module / C240 (B1-B4) or SIMOTION D onboard (See Chapter Overview (Page 1951))	Select the radio button if you want to configure a measuring input for global measuring. This functionality is available for hardware (see Hardware for measuring inputs (Page 1920)). With global measuring, the current actual values of one or more encoders are measured with positioning accuracy with a signal edge on the relevant input in order to determine lengths or distances from these (possible with any encoders existing in the project).
	Drive-related (local measuring input)	Click on the radio button if the measuring input is connected directly to the C230-2/C240, D4xx, D410-2, D410-2, D4x5-2 (X122/X132) or the drive. Measurements can be taken only on the device that has a measuring system. You still have to enter the name of the measuring input number.
	Monitoring measuring input	Select the radio button if you want to configure a monitoring measuring input. If the checkbox is selected, the measurement result of a measuring input TO that is interconnected on another HW input (see Hardware for measuring inputs (Page 1920)) is also measured by the measuring input configured as a monitoring measuring input (listening in on the measurement event). With this functionality, one measuring input can have a functional effect on several axes / external encoders. If you have configured a monitoring measuring input, you can now only select the processing cycle clock and axis measuring system number.
Input (global measuring input)		The output can be symbolically assigned via the assignment dialog using the ... button in the Input field (symbolic assignment is activated by default in the project as of V4.2). For more details, see cam output types. (Page 1811) If symbolic assignment is not active or if the CPU version < V4.2, a physical output is assigned by entering the HW address and bit number in the Input field. The logical HW address must be located outside the process image and therefore be greater than 63.
Measuring input number (local measuring input)		Enter the measuring input at the drive of the axis here. One input can be assigned to several measuring inputs. Input 1 is the default setting.
Monitor current status (measurement once)		If the checkbox is activated, short pulses (short position control cycle clock) will be suppressed at the measuring input. If the measuring input is activated, provided that rising edge is selected under **Edge**, the measuring input will only be activated when the measuring input was at signal status 0 for at least one position control cycle clock.
Activation time of the measuring range on measuring input		Enter an activation time, in seconds, for the activation and deactivation of the measuring range here. With this time, you can compensate, for example, the runtimes during activation via PROFIBUS and drive. The accuracy of the activation depends on the position control cycle clock.
...		Button for opening the assignment dialog. Select a parameter or an address in the Assign dialog.
		Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display

Time stamp correction value

When measuring with measuring inputs, the following should be taken into account when calculating the measurement positions from the measurement result and actual position of axis / external encoder:

- the filter time constant setting on the measuring input (e.g. TM17 smoothing filter),
- the delay times in the transmission section and sensors and
- with global measuring inputs also the Ti setting on axis / external encoder (depending on the actual value reference setting)

The hardware delay times on the measuring inputs can be found in the Manuals and can usually be ignored for values of much less than 125 µs (e.g. with SIMOTION D, SINAMICS Control Units, SIMOTION C240))

If the filter time can be parameterized for the hardware used (e.g. TM17 high feature: 1 µs or 125 µs), set the smallest possible value for the measuring inputs.

The delay times up to the measuring input, due to the transmission line and sensor technology, depend on the components used and are usually unknown. If these values are small, they can also be ignored.

With the setting "actual value reference - reference to the actual value at the encoder," the transmission time Ti of the system is considered. With the other settings, you must also consider the transmission time Ti for "global" measuring inputs. The value can be found in the properties of the SINAMICS drive unit in the HW Config.

Example of how to proceed:

- For SIMOTION D: in HW Config, double-click on **SINAMICS Integrated**
- Under **DP slave properties**, in the tab **Cycle synchronization** read off the value **Ti**

During the measuring input configuration of "global" measuring inputs, also enter the negated value of **Ti** as offset value of the time stamp (**TimeStampConfig.correctionTime**).

If measurements have to be performed with "very high" accuracy or the dead times caused by the filter time constant and delay times from the transmission line, BERO, converter, contacts, etc. cannot be ignored, we recommend the machine-specific determination of the dead times that really occur, through measurements or series of measurements.

Measurements with velocity-dependent displaced measuring results are an indication that the dead times have not been compensated or incorrectly compensated.

The procedure for **empirical/measuring technique determination** of dead times is similar to the process for determining dead times during cam output.

See also

Determining derivative-action times for output cams (dead time compensation) (Page 1833)

Measuring input defaults

You can define the defaults for every measuring input. These values are stored in system variables and can be changed by programs.

Double-clicking in the project navigator below the measuring input on the **Defaults** element displays the window in the working area.

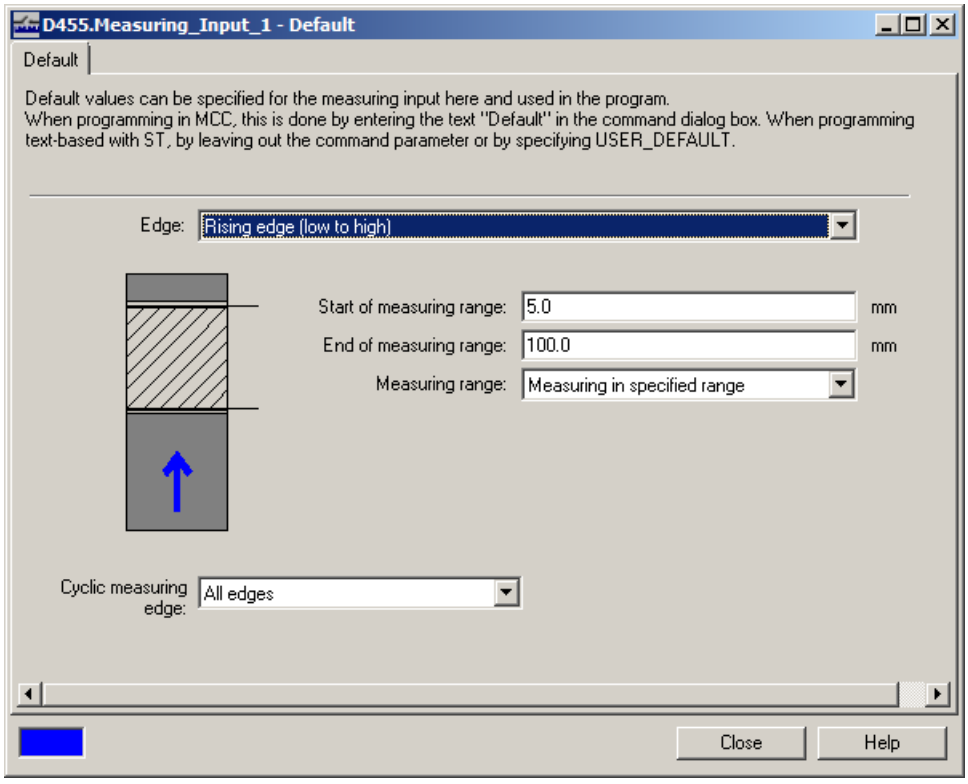


Figure 4-262 Measuring input defaults

You can set the following parameters:

Table 4-187 Measuring input defaults

Field/Button	Meaning/information
Edge (measurement once)	Under **Edge** you select the signal edge that starts the measurement when present at the measuring input and measures the actual position of the axis. The drive must be capable of evaluating the signal edge (rising, falling, or both edges) selected by SIMOTION at the measuring input. **Rising edge (low to high)** The actual position is recorded with the rising edge of the measuring input. **Falling edge (high to low)** The actual position is recorded with the falling edge of the measuring input. **Measure at both edges** The actual position is recorded using both the rising and falling edge of the measuring input. **Measure at both edges, starting with a rising edge (low to high)** The actual position is recorded using both the rising and falling edge of the measuring input, and measurement is begun at the first rising edge. **Measure at both edges, starting with a falling edge (high to low)** The actual position is recorded using both the rising and falling edge of the measuring input, and measurement is begun at the first falling edge.
Start of measuring range/ End of measuring range (measurement once) (See also Chapter **Measuring range**)	Enter the start and end of the measuring range here. If the start of the measuring range is greater than the end of the measuring range for modulo axes, the measuring range extends from the initial value through the modulo transition to the end value. For non-modulo axes the initial and end values are swapped in this case.
Measuring range (measurement once) (See also Chapter **Measuring range**)	Under **Measuring range** you can choose whether or not to apply the defined measuring range. **Measuring without specified range** The measuring input records the measured values in the entire traversing range. **Measuring in specified range** The measuring input only records the measured values within the measuring range defined by the start and end points.
Cyclic measuring edge (Page 1929)	The edges to be detected in cyclic measuring are selected here. Up to two edges can be acquired cyclically per measuring input processing cycle (IPO interpolation cycle, IPO_2 interpolation cycle, or servo cycle clock). Cyclic measuring is only possible with certain hardware (see Hardware for measuring inputs (Page 1920)). **All edges** Both rising and falling edges are measured. **Rising edges only** Only rising edges are measured. **Falling edges only** Only falling edges are measured.

See also

Measuring range (Page 1936)

Local measuring

Introduction

With local measuring inputs, the measuring input is assigned as part of its configuration. The configuration specifies the number of the measuring input to be used and the number of the encoder on the assigned axis (PROFIdrive).

Local measuring on C230-2, C240 (not C240 PN)

With a signal edge at the relevant M1 or M2 input, the current actual values of one or more encoders connected to the onboard encoder interfaces (X3 to X6) are measured with positioning accuracy to determine lengths or distances.

The assignment of inputs is not fixed; the special use is activated in the SCOUT engineering system during configuration of the measuring input via the measuring input number.

Note

As the C240 PN has no onboard encoder interfaces, it has no local inputs of measuring inputs.

Local measurement on D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310, CU310-2, CU320 and CU320-2

If you connect a local measuring input TO to the measuring inputs of D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2 and CU310, CU310-2, CU320 and CU320-2, you will have to set the following parameters (PROFIdrive):

- Parameter p488 on the associated drive for one local measuring input
- Parameters p488 and p489 on the associated drive for two local measuring inputs
- Parameters p728.8 - p728.15 on the control unit as inputs for all DI/DO used as measuring inputs (these settings can also be made using a parameterization mask)

Local measuring on other drives (MASTERDRIVES MC, SIMODRIVE 611U, etc.)

With SIMOTION, the measuring input is assigned as part of its configuration. The configuration specifies the number of the measuring input to be used and the number of the encoder on the assigned axis.

Please refer to the documentation for the respective drive system for details of any additional settings which may be required.

Global measuring

Overview

Description

You can configure TO measuringInput on the ET200SP technology module TM Timer DIDQ 10x24V and ET200MP technology module TM Timer DIDQ 16x24V.

- Configuration to ET200SP technology module TM Timer DIDQ 10x24V is possible both using Step7 in connection with SCOUT Classic and with the TIA Portal in connection with SCOUT TIA. Further information can be found in the section Global measuring on TM Timer using DIDQ 10x24V as example (Page 1904).
- Configuration on the ET200MP technology module TM Timer DIDQ 16x24V is only possible using TIA Portal in connection with SCOUT TIA. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA.

Further information

Further information on the functions of the TM Timer DIDQ technology modules can be found in the SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V and SIMATIC S7-1500/ET 200MP Technology Module TM Timer DIDQ 16x24V.

With global measuring, the current actual values of one or more encoders are measured with positioning accuracy with a signal edge on the relevant input in order to determine lengths or distances from these (possible with any encoders present in the project). Up to two edges can be measured for each position control cycle clock of the Measuring Input TO.

The assignment of the inputs is not fixed, and the special use is activated via the HW address when the Measuring Input TO is configured.

Note

Global measuring is only possible for measuring inputs with time stamp. Hardware for measuring inputs (Page 1920)

See also

Global measurement on D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310, CU310-2, CU320 and CU320-2 (Page 1957)

Configuring and interconnecting a listening Measuring Input TO (Page 1964)

Global measuring on TM15/TM17 High Feature

Description

Digital inputs are available for connection of measuring inputs on the TM15 and TM17 High Feature Terminal Modules. These can be used for one-off or cyclic measuring. Cyclic measuring is only supported by the TM17 High Feature.

Note

For more information about the Measuring Input TO on Terminal Modules, see the *TM15 / TM17 High Feature Terminal Modules* Commissioning Manual.

To configure a measuring input on a TM15/TM17 High Feature

1. In the project navigator, below the input/output component (TM15/TM17) that you want to use, double-click the entry **Inputs/outputs.** The **Bidirectional Digital Inputs/Outputs** window is displayed.
2. For the selected I/O channel, select measuring input as the function.

   Note

   If you do not use symbolic assignment (see Section Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual), you must note the offset (e.g. 3.1).

3. Insert a new measuring input or use an existing one.
4. Assign parameters to the measuring input TO.
5. Double-click **Configuration** below the measuring input in the project navigator. The **Configuration** window appears in the working area.
6. Activate the option **Standard (global measuring input)**. Assignment of an output to a measuring input is supported as of V4.2 by symbolic assignment (see Section Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions Manual) or by entering the hardware address. In order to assign a physical output, the HW address and bit number must be entered in the **Output** field.
7. Click **OK** to close the window and select **Project > Save**.

To determine the logical hardware address for a TM15/TM17 High Feature (only if symbolic assignment is not activated)

1. For TM15/TM17 modules, the hardware addresses result from the configured address range, and the bit numbers from the offset.
2. In the project navigator, below the input/output component (TM15/TM17) that you want to use, double-click the entry **Inputs/outputs.** The **Bidirectional Digital Inputs/Outputs** window is displayed**.**
3. Find the input that you want to use (**Measuring input** must be selected under **Function**) and note the offset (e.g. 3.0).

4. In the project navigator, below the SIMOTION device or the SINAMICS drive unit, select
   - For SIMOTION D:
     SINAMICS_Integrated > Communication > Telegram configuration
   - For the SINAMICS S120 drive unit:
     Communication > Telegram configuration
5. Double-click **Telegram configuration** and, in the window that opens, select the tab **PROFIdrive PZD telegrams.** The components are displayed there with the address ranges (e.g. TM17 input data 288...299).

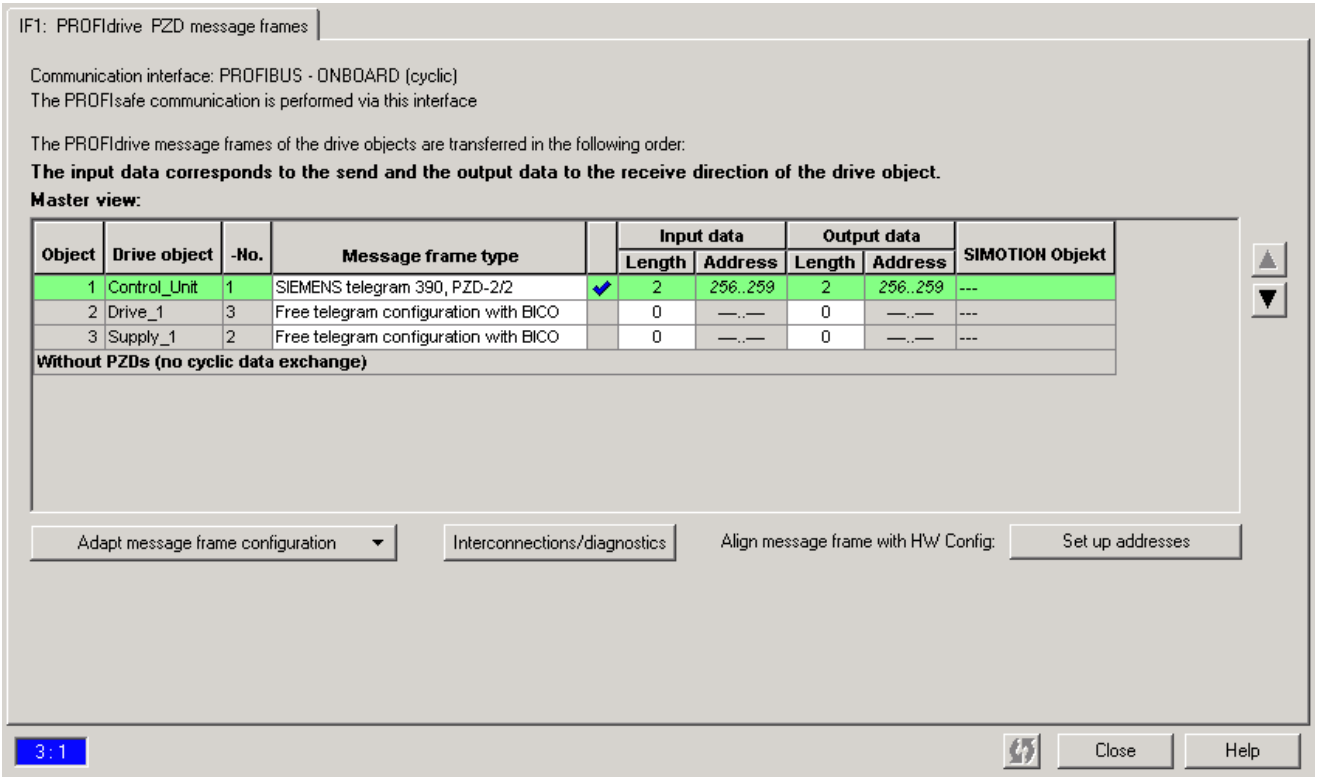


Figure 4-263 Determining the hardware address of the components

6. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If there are question marks in the fields instead of I/O addresses, you must also perform an alignment.
7. Now calculate the HW address by adding the base input address (first value of the address range) of the TM to the offset (for example 288 + 3 = 291).
8. The bit number is defined by means of the offset. For example, an offset of **3.0** results in a bit number of **0**.

See also

Inserting Measuring Inputs (Page 1940)

Parameterization of the Measuring Input technology object (Page 1941)

Global measuring on SIMATIC ET200SP and ET200MP

Overview

Description

You can configure TO measuringInput on the ET200SP technology module TM Timer DIDQ 10x24V and ET200MP technology module TM Timer DIDQ 16x24V.

- Configuration to ET200SP technology module TM Timer DIDQ 10x24V is possible both using Step7 in connection with SCOUT Classic and with the TIA Portal in connection with SCOUT TIA. Further information can be found in the section Global measuring on TM Timer using DIDQ 10x24V as example (Page 1954).
- Configuration on the ET200MP technology module TM Timer DIDQ 16x24V is only possible using TIA Portal in connection with SCOUT TIA. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA.

Further information

Further information on the functions of the TM Timer DIDQ technology modules can be found in the SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V and SIMATIC S7-1500/ET 200MP Technology Module TM Timer DIDQ 16x24V.

Global measuring on TM Timer, using DIDQ 10x24V as example

Description

You can configure TO measuring inputs on the ET200SP Technology Module TM Timer DIDQ 10x24V in SIMOTION SCOUT or SIMOTION SCOUT TIA in connection with the TIA Portal. The configuration in the TIA Portal is described in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA. The configuration in SIMOTION SCOUT in connection with Step7 is described below:

In order to parameterize the ET200SP TM Timer DIDQ 10x24V with Step7 in connection with SIMOTION SCOUT Classic, a Hardware Support Package (HSP) for Step7 is required. Further information can be found in the Commissioning Manual Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA in the section on hardware and software requirements.

For the TM Timer DIDQ technology module, the digital inputs are used for the measuring input. Irrespective of whether you work
with SIMATIC STEP 7 or SIMOTION SCOUT, the parameterization is always performed in HW Config:

Parameterization of TM Timer DIDQ in HW Config

1. Open the HW configuration in SIMOTION SCOUT or SIMATIC STEP 7.
2. Select the header module (IM) with the TM Timer DIDQ in HW Config.
   The TM Timer DIDQ appears in the detail view.
3. To open the context menu, select the TM Timer DIDQ with the right mouse key.

4. Select object properties.
   You can view or change the following properties in the object properties:
   - General
   - Addresses
   - Identification
   - Parameters
5. Open the tab **Parameter** > **DQ0 / DI0**
6. Set the parameter value **Configuration DQ/DI group** to **Use input/output individually**.

   Note

   With this setting, the DQ0 channels can be interconnected as cam output CAM or DI0 as measuring input MI under SIMOTION SCOUT.

7. Press **OK** to close the dialog.
8. Open the menu **Station** > **Save and Compile**.

Note

The technology modules and the individual channels may only be reparameterized in offline state. All changes only take effect after saving, compiling and subsequent download of your changed user project to the controller. The download can be performed in HW Config via **Target system** > **Download to module** or in SCOUT via **Load CPU / drive unit to target device**.

Parameterization of MI in SCOUT

1. Insert a new measuring input or use an existing one.
2. Assign parameters to the measuring input TO.
3. Double-click Configuration below the measuring input in the project navigator. The Configuration window appears in the working area.
4. Activate the measuring input selection **Standard (global measuring input)**.

5. Assignment of an output to an output cam/cam track is supported as of V4.2 using symbolic assignment or by entering the HW address.

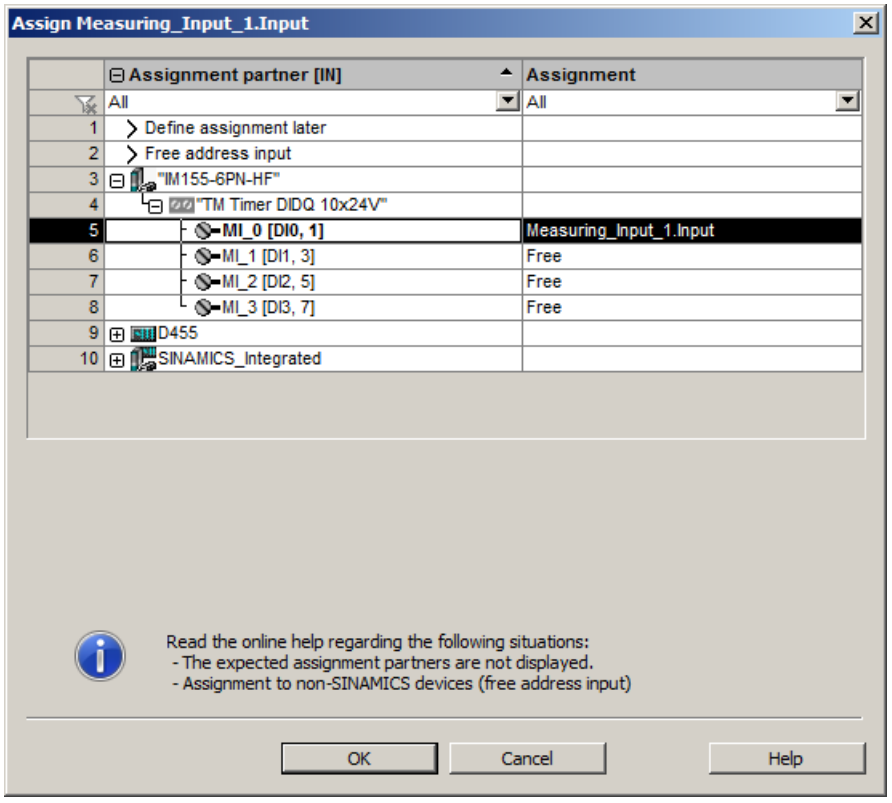


Figure 4-264 Assigning a measuring input by means of symbolic assignment

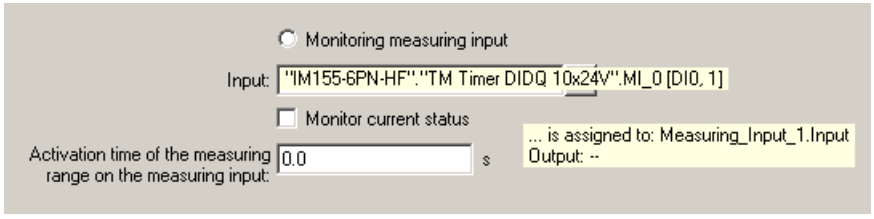


Figure 4-265 Assigning a measuring input by entering the HW address

6. Click OK to close the window and select **Project** > **Save**

Global measuring on C240/C240 PN (B1-B4)

The inputs B1-B4 are available for global measuring on the C240. These can be used for one-off or cyclic measuring.

Note

For more information about the measuring input on C240/C240 PN, refer to the *C2xx* Operating Instructions.

To configure a measuring input on a C240/C240 PN (B1-B4) (global measuring)

1. Insert a new measuring input (see **Inserting measuring inputs** section) or use an existing one.
2. Parameterize the measuring input (see the **Parameterization of the measuring input technology object** section)
3. Double-click **Configuration** below the measuring input in the project navigator.
   The **Configuration** window appears in the working area.
4. Activate the option **Standard (global measuring input)**. If symbolic assignment is not active or if the CPU version < V4.2, a physical input is assigned by entering the HW address and bit number in the Input field.
5. Enter the **HW address** and the **bit number**. The hardware address and the bit number can be ascertained in **HW Config** (e.g. 64.2 for the measuring input Pin 3 of connector X1).
6. Click **OK** to close the window and select **Project > Save**.

See also

Inserting Measuring Inputs (Page 1940)

Parameterization of the Measuring Input technology object (Page 1941)

Global measurement on D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310, CU310-2, CU320 and CU320-2

On D4xx, D410-2, D4x5-2, CX32, CX32-2, CU310, CU310-2 and CU320 you have digital inputs to connect to a global measuring input. These can be used for one-off or cyclic measuring.

If symbolic assignment is not active or if the CPU version < V4.2, the following applies:

The measurement results are transmitted by standard telegrams 391/392 and 393/396 (only D410-2/CU310-2/D4x5-2/CX32-2/CU320-2) and not via the axis telegram.

If a 39x telegram is set, the SINAMICS I/Os are interconnected automatically to this 39x telegram using BICO interconnections and are so available for SIMOTION.

Note

In the case of versions earlier than V4.2, when using 39x telegrams, the onboard D4x5 outputs are to be assigned exclusively to SIMOTION.

Inverting global measuring inputs

When selecting 39x telegram, inverting of the global measuring inputs is activated on Sinamics_Integrated, CX32, CX32-2 and S120 via parameters p490 and p2088[2].

The default assignment of p2088[2] is as follows:

- p2088[2].0 -> Inversion of DI/DO 8
- p2088[2].1 -> Inversion of DI/DO 9
- p2088[2].2 -> Inversion of DI/DO 10

- p2088[2].3 -> Inversion of DI/DO 11
- p2088[2].4 -> Inversion of DI/DO 12
- p2088[2].5 -> Inversion of DI/DO 13
- p2088[2].6 -> Inversion of DI/DO 14
- p2088[2].7 -> Inversion of DI/DO 15

If the default assignment of the digital inputs to PZD2 is changed in 39x telegram, this must be taken account of in p2088[2] correspondingly.

This is how you configure a measuring input on a measuring input of D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2

1. In the project navigator, switch to the **Control_Unit** entry under the control unit for the respective device.
2. Double-click **Inputs/outputs** below the control unit. The window appears on the workspace.
3. Select **Bidirectional digital inputs/outputs**.
4. Select via button  under **Further interconnections** parameters p2082 and the associated bit.
5. Insert a new measuring input (see Chapter **Inserting measuring inputs**) or use an existing one.
6. Assign parameters to the measuring input TO.
7. Double-click **Configuration** below the measuring input in the project navigator.
   The **Configuration** window appears in the working area.
8. Activate the option **Standard (global measuring input)**. Assignment of an output to a measuring input is supported as of V4.2 by symbolic assignment (see Chapter Symbolic Assignment (as of V4.2) in the SIMOTION Runtime Basic Functions manual) or by entering the hardware address. In order to assign a physical output, the HW address and bit number must be entered in the **Output** field.
9. Click **OK** to close the window and select **Project > Save**.

This is how you determine the logical hardware address for the onboard inputs of D4xx, D410-2, D4x5-2 (X122/X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2 (only if no symbolic assignment is activated)

1. In the project navigator, select **Communication > Telegram configuration** under the SIMOTION or SINAMICS device.
2. Double-click **Telegram configuration** and, in the window that opens, select the tab **PROFIdrive PZD telegrams**. The components are displayed there with address range (input/output data).

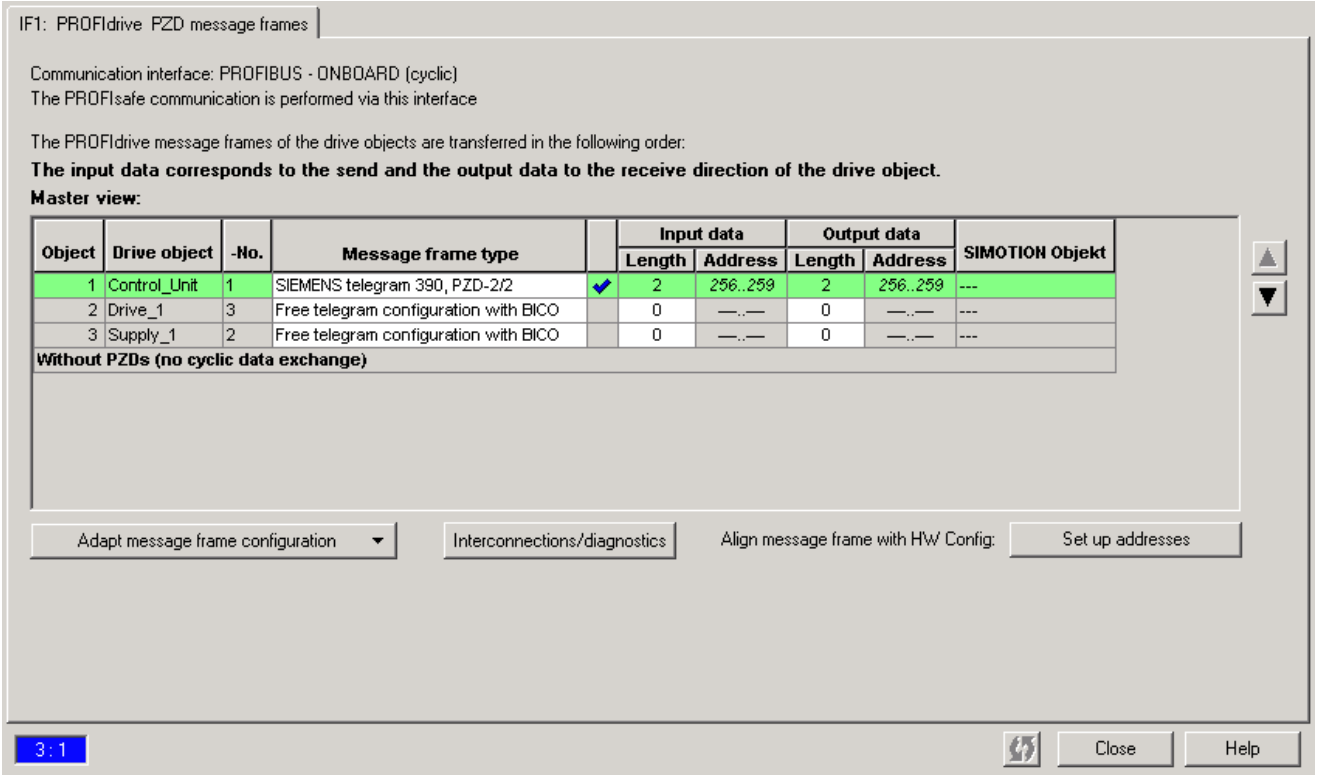


Figure 4-266 Control unit I/O addresses

3. Select the telegram type **SIEMENS telegram 391** (max. 2 measuring inputs), **392** (max. 6 measuring inputs) or 393/396 (max. 8 measuring inputs only D410-2/CU310-2/D4x5-2/ CU320-2). The message depends on the number of measuring inputs configured.
4. Before you determine the hardware address, an alignment between HW Config and SIMOTION SCOUT, with respect to the address, must be performed. If this has not been performed or you have changed the addresses, click on **Set up addresses**. If there are question marks in the fields instead of I/O addresses, you must also perform an alignment.
5. Now calculate the HW address by adding the base input address (first value of the input data) of the control unit to the offset (e.g. 256 + 3 = 301). The offset always has the value **3**.
6. You will find the bit number in the following table. The inputs are set in parameters 680.0 to 680.7 of the control unit, e.g. bit 1 in parameter 680.0.

Table 4-188 Bit numbers for D410, D4x5, D4x5-2 (X122/X132), CX32, CX32-2 CU310, CU320, CU320-2

Input D4x5, CU320	Input D4x5-2, CU320-2	Input D410, CU310, CX32	Input D410-2, CU310-2	Input CX32-2	Bit No.
-	X122.9 (DI/DO 8)	-	X121.7 (DI/DO 8)	X122.9 (DI/DO 8)	Bit 0
X122.8 (DI/DO 9)	X122.10 (DI/DO 9)	X121.8 (DI/DO 9)	X121.8 (DI/DO 9)	X122.10 (DI/DO 9)	Bit 1
X122.10 (DI/DO 10)	X122.12 (DI/DO 10)	X121.10 (DI/DO 10)	X121.10 (DI/DO 10)	X122.12 (DI/DO 10)	Bit 2
X122.11 (DI/DO 11)	X122.13 (DI/DO 11)	X121.11 DI/DO 11)	X121.11 (DI/DO 11)	X122.13 (DI/DO 11)	Bit 3
-	X132.9 (DI/DO 12)	-	X131.1 (DI/DO 12)	-	Bit 4
X132.8 (DI/DO 13)	X132.10 (DI/DO 13)	-	X131.2 (DI/DO 13)	-	Bit 5
X132.10 (DI/DO 14)	X132.12 (DI/DO 14)	-	X131.4 (DI/DO 14)	-	Bit 6
X132.11 (DI/DO 15)	X132.13 (DI/DO 15)	-	X131.5 (DI/DO 15)	-	Bit 7

Global measuring on D4x5-2 (X142)

Unlike (X122/X132), measuring inputs on measuring input D4x5-2 (X142) are configured in HW Config

Configuration of terminals and function as well as the fixed telegram's start address are configured in HW Config.

To configure a measuring input on an input of D4x5-2 (X142)

The dialog box for configuring the X142 can be opened via the input/output X142 in the project navigator. Alternatively, you can access the dialog to be configured via the HW Config by clicking **Open HW Configuration ...** in the device's context menu.

Slot X142 is shown in the detail view and in the station window of the respective D4x5-2 device. You can access the **Properties** dialog via the **Object properties...** context menu or by double-clicking on the X142 line.

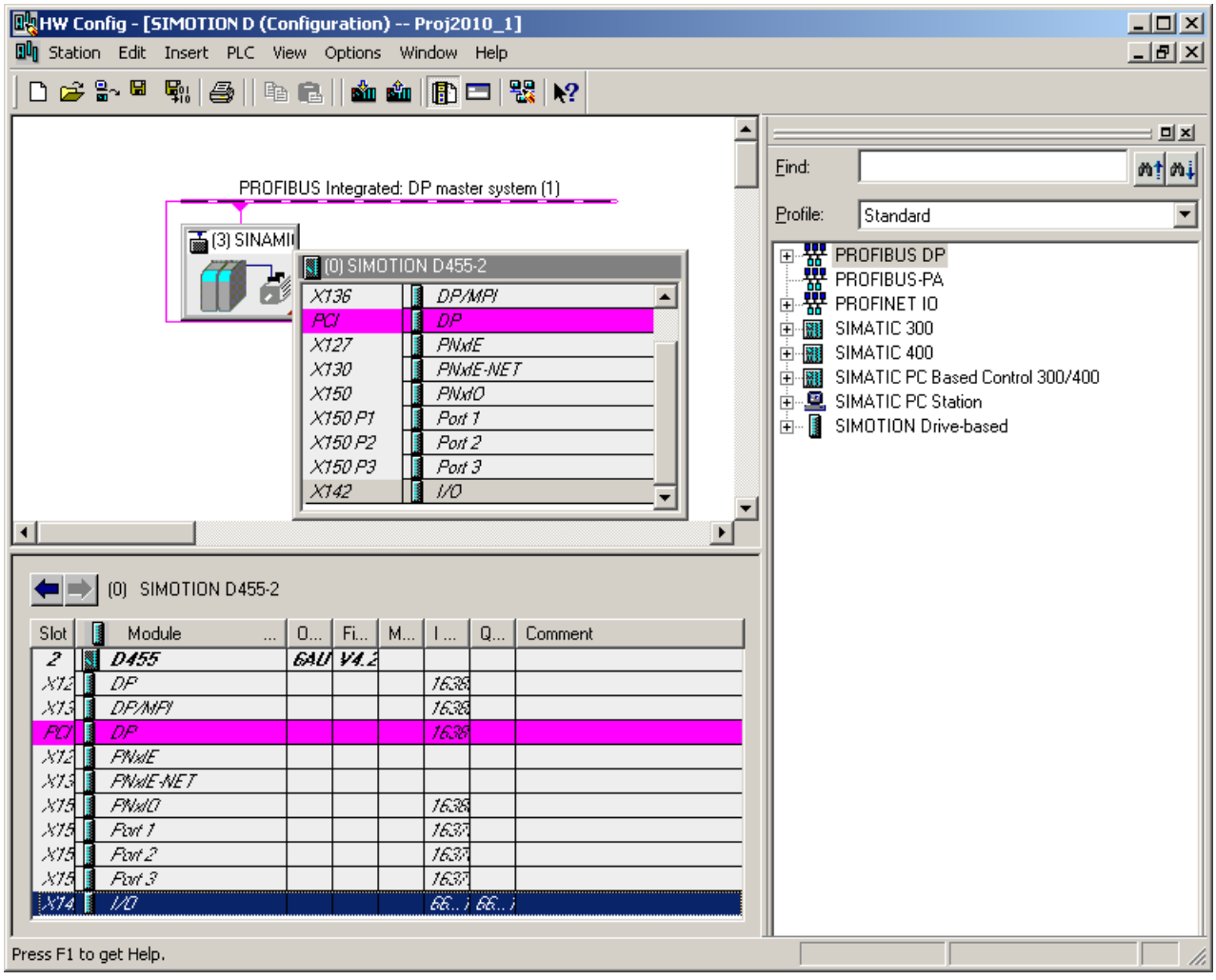


Figure 4-267 Display of D4x5-2 in HW Config

In the **Properties** dialog, you can manage **Addresses** and interconnect **Bidirectional digital inputs/outputs 0-7**.

Addresses

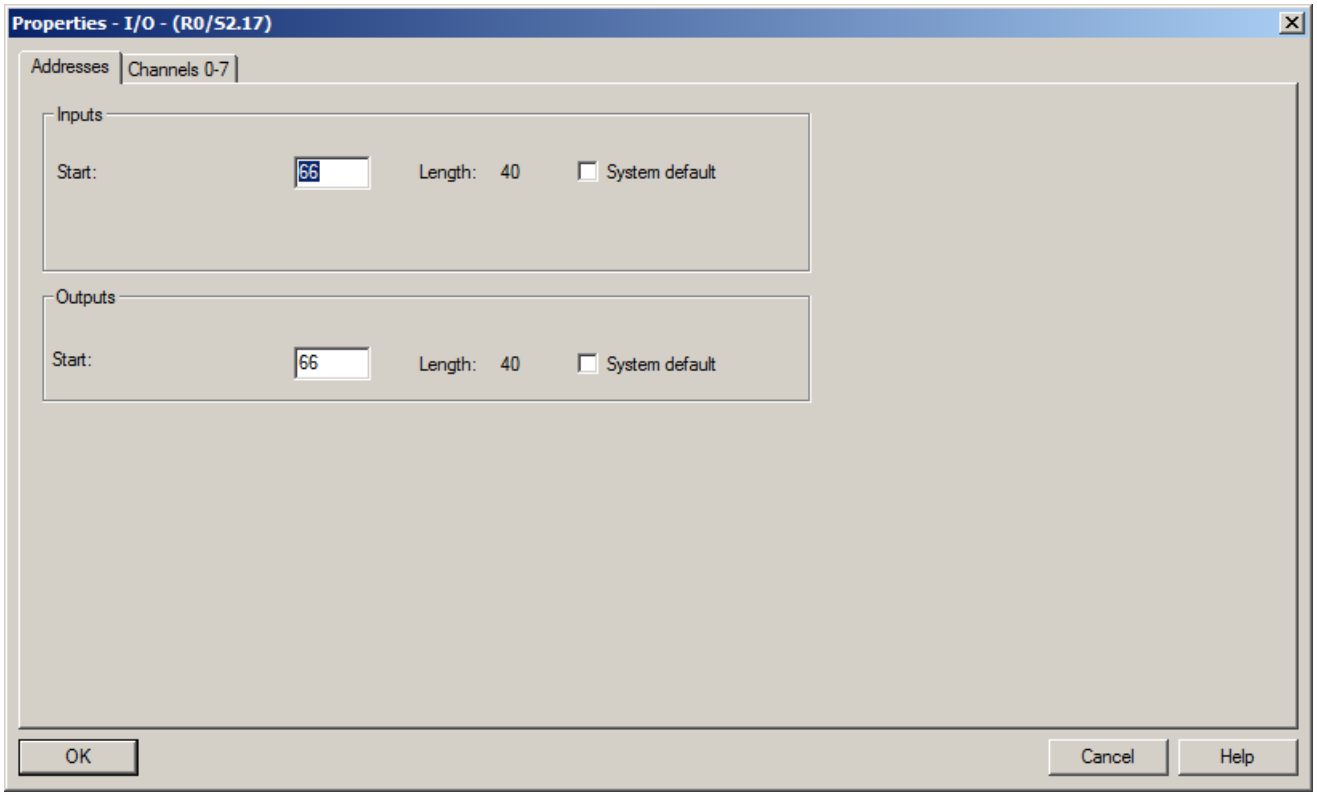


Figure 4-268 Manage addresses

Table 4-189 Addresses

Function		Description
Inputs		
	Start	Under **Start**, enter the start address of the digital inputs. Addresses >= 64 outside the process image are valid.
	Length	The **length** of the digital input address cannot be modified.
	System specification	With **System specification**, you can activate or deactivate the automatic setting of digital input addresses.
Outputs		
	Start	Under **Start**, you can enter the start address of the digital outputs. Addresses >= 64 outside the process image are valid.
	Length	The **length** of the digital output address cannot be modified.
	System specification	With **System specification**, you can activate or deactivate the automatic setting of digital output addresses.

Channels 0-7

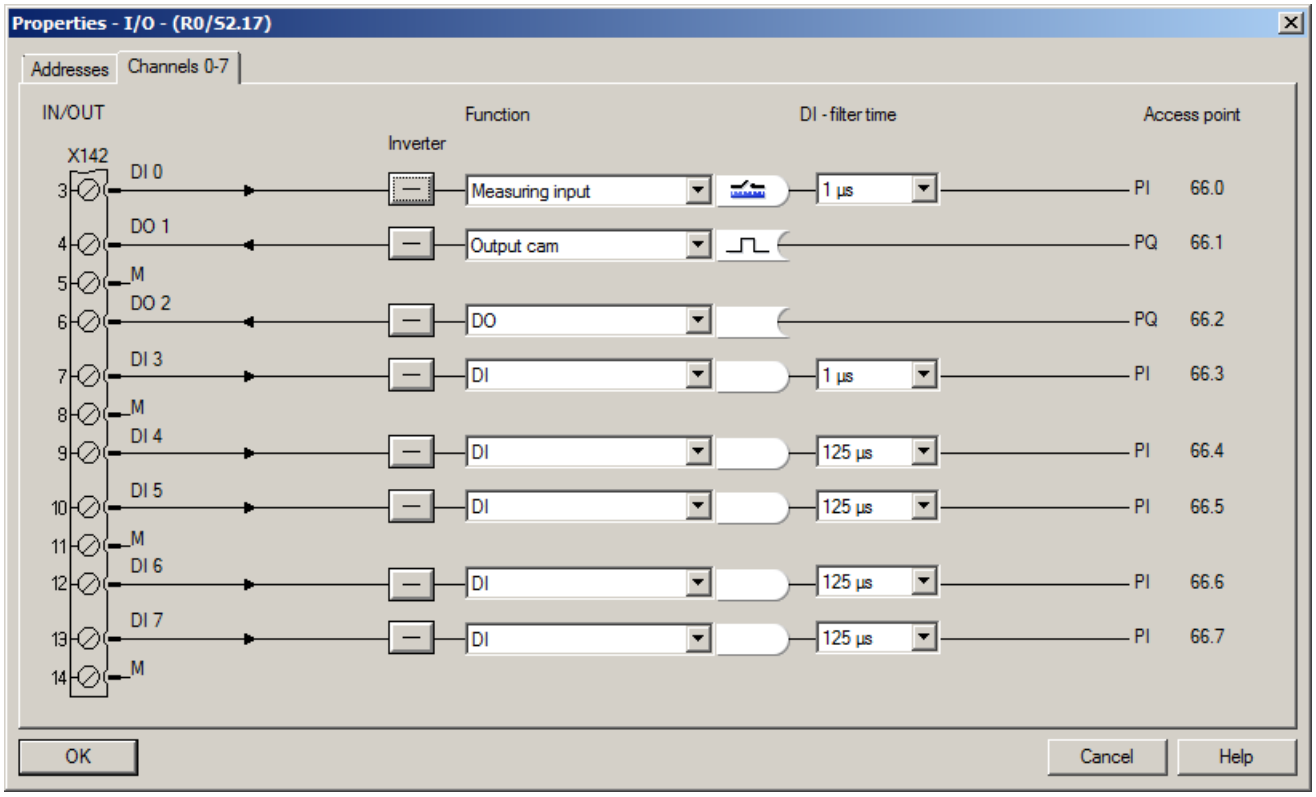


Figure 4-269 Channels 0-7

Table 4-190 Channels 0-7

Function		Description
IN/OUT X142		Inputs/outputs (IN/OUT 0-7) of X142.
Inverter		Button for inverting.
Function		
	DI	Digital input.
	DO	Digital output.
	Output cam	Cam output.
	Measuring input	Measuring input input.
IN filter time		
	Filter time	The drop down list contains 1µs and 125µs values. Only available for **DI** and **measuring input** function.
	Set substitute value/keep last value	The selection options are available for the **DO** and **output cam** function.
Logical address		PI, PQ for output cam and measuring input. Note: Logical address is only required if you work without symbolic assignment.

Configuring and interconnecting a listening Measuring Input TO

With the "Monitoring measuring input" function, the measurement event of a measuring input can also be measured simultaneously by several Measuring Input TOs.

To configure and interconnect a monitoring measuring input:

1. Before you configure a monitoring measuring input, a measuring input must be configured on a measuring input with a HW address. This can be a measuring input (see Hardware for measuring input (Page 1920)). (see chapter Overview (Page 1951)).
2. Insert a new measuring input (see Inserting Measuring Inputs (Page 1940) section).
3. Double-clicking in the project navigator below the measuring input on the **Configuration** element displays the window in the working area.
4. Select **Monitoring measuring input**, and enter the **processing cycle clock** and the **system number** (see the **Measuring input configuration** section).
5. Double-clicking in the project navigator below the monitoring measuring input on the **Interconnections** element displays the window in the working area.
6. In the **EventIn** table row under the **Interconnected to output connectors** column, select the **EventOut** of the measuring input with which you want to interconnect the monitoring measuring input. The axis to which the monitoring measuring input is assigned is automatically displayed under **Reference**.

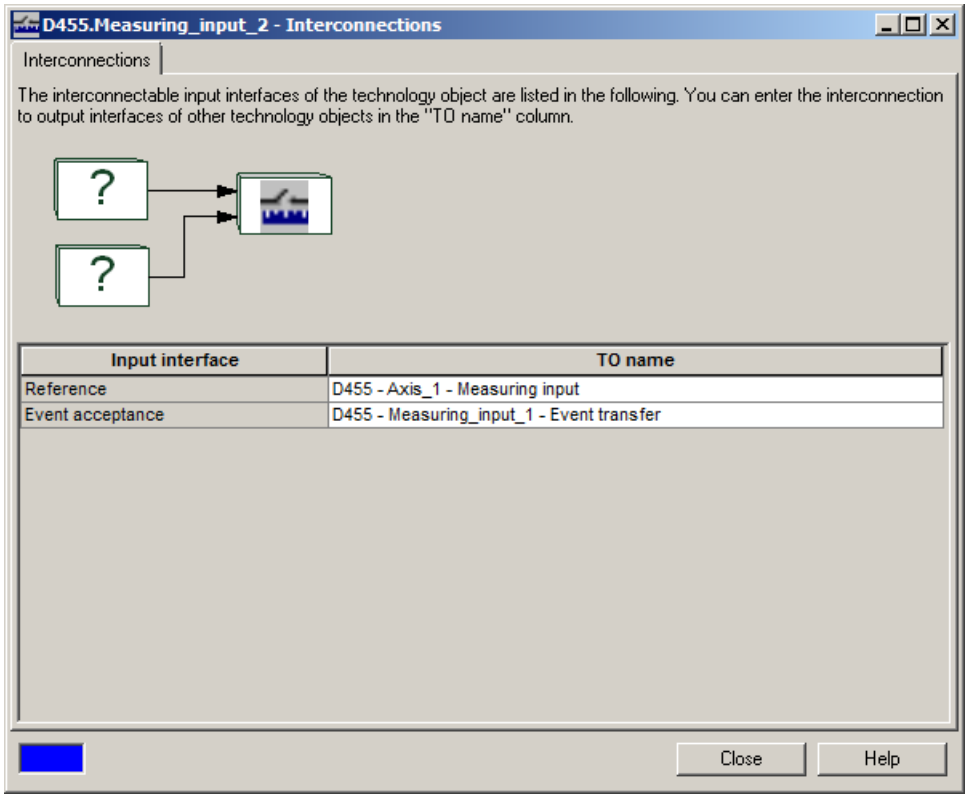


Figure 4-270 Interconnecting a monitoring measuring input

7. Click **Close**. The monitoring measuring input is configured and interconnected.

See also

Measuring Input Configuration (Page 1942)

Measuring one measurement event on several axes - Listening measuring input (V4.0 and later) (Page 1925)

Measuring input with hardware enable (TM17 High Feature)

On the TM17 High Feature, hardware-based enable inputs can be configured for up to six measuring-input inputs. Measuring signals can only be detected at the assigned measuring input if an enable is present (this is a gate function).

Note

To be able to use an enable input it must be configured on the TM17 High Feature (see Section **Global measuring on the TM15/TM17 and C240 (B1-B4)**).

You will find more information on the hardware enable in the *Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

Required configuration for level-controlled HW enable

- Measuring input is configured.
- Digital input on TM17 High Feature parameterized for measuring input and level-triggered enable input set at this input. The appropriate enable input for the enable signal is parameterized automatically.
- Digital input configured for detecting measured values (HW address).
- Measuring input must be active.

Level-controlled enable procedure

The measuring input is activated with **_enableMeasuringInput** or **_enableMeasuringInputCyclic**. No measured values are forwarded to the TO in this state, as no enable signal is present yet (gate closed). Edges are forwarded to the TO by setting the enable (gate open).

Depending on the configured measuring range, measured edges are filtered by the TO in the measuring input processing cycle clock and by the TM17 High Feature with a resolution of several µs.

Edges are not detected if:

- The measuring process is terminated with **_disableMeasuringInput**.
- The edges fall outside the measuring range.
- No enable signal is present (gate closed).

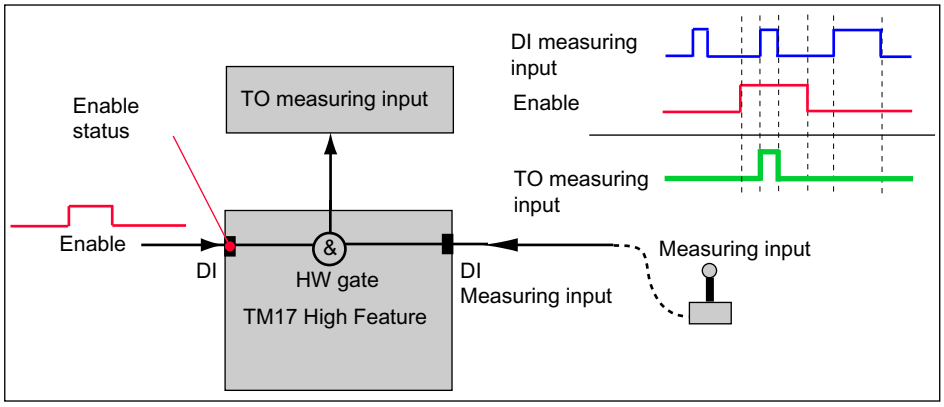


Figure 4-271 Schematic representation of HW enable on the measuring input

It is also possible to execute the enable with inverted logic, i.e. the enable input on the TM17 High Feature can be operated inversely and then works in LOW-active mode.

Overriding the enable

It is possible to override the measuring enable input with a software enable signal. To do this, you have to access the input directly via its address and set the bit. The enable is achieved, as long as the bit is set (functions as enable input).

See also

Overview (Page 1951)

4.3.5.4 Measuring Input technology object programming/references

Programming

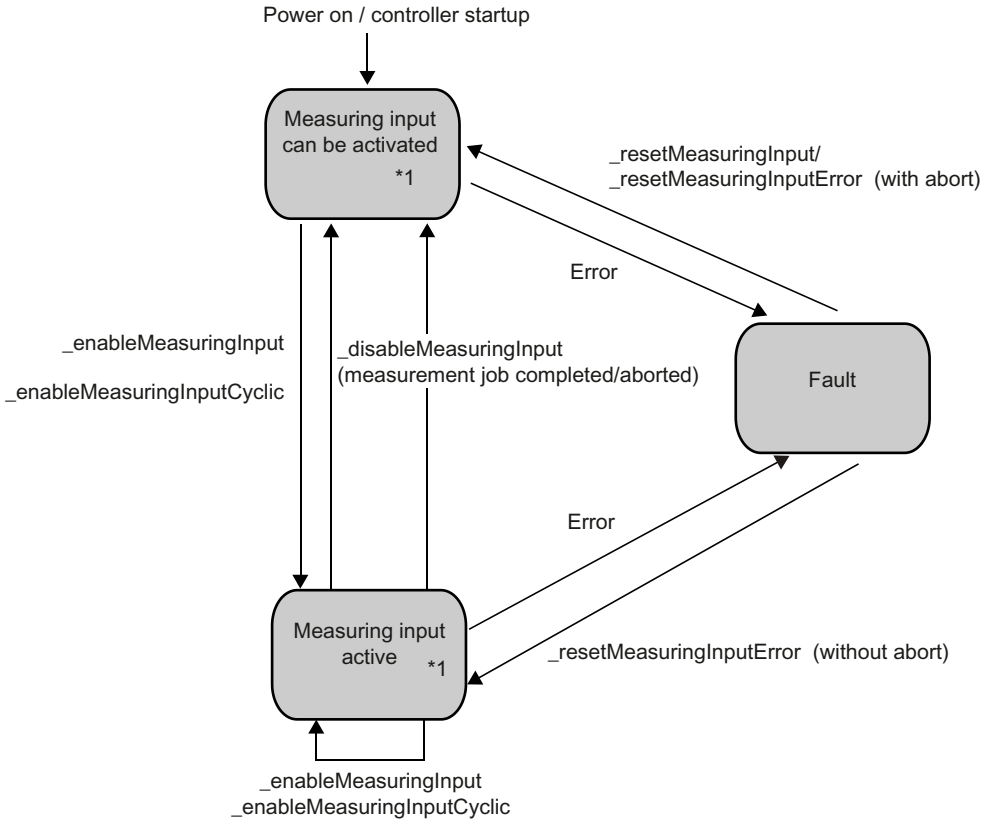


Figure 4-272 Programming and execution model for the Measuring Input TO

***1** The following commands are effective in the TO states **can be activated** and **active**:

- **_disableMeasuringInputSimulation**
- **_enableMeasuringInputSimulation**

Commands

The Measuring Input technology object can be addressed in the user program using the following commands:

Table 4-191 Measuring Input TO system functions

Commands	Description	Application
_enableMeasuringInput	Activate measurement job.	Measurement is activated. Measuring is terminated after the measuring process has been executed.
_enableMeasuringInputCyclic (V3.2 and higher)	This function activates a cyclical measurement job. The results of the last measurement are displayed in the system variables.	Cyclic measuring of axis/encoder positions by activating a measurement job once only
_disableMeasuringInput	Deactivate measurement job.	Measurement is terminated.
_enableMeasuringInputSimulation	Activates measuring input simulation. **Measured result arrived** is set and allocated to the programmed measured value as a measured result.	Simulation of a program run using default settings of programmed measured values
_disableMeasuringInputSimulation	Deactivate simulation mode.	-
_resetMeasuringInput	Reset measuring input.	Create initial state of measuring input.
_resetMeasuringInputError	Reset error on measuring input.	E.g. acknowledge configuration errors after entering correct values.
_resetMeasuringInputConfigDataBuffer	This function clears the configuration data collected in the buffer since the last activation without activating them.	Changing configuration data in the RUN state discards the accumulated modifications.
_getMeasuringInputErrorNumberState (V3.1 and higher)	Readout of error number status.	Check for occurrence of an error with the specified error number
_getStateOfMeasuringInputCommand (V3.2 and higher)	This function returns the execution state of a command.	Check whether or not measurement has already taken place (i.e. the command ID is still available or has already been deleted)
_bufferMeasuringInputCommandId (V3.2 and higher)	This function enables **commandId** and the associated command status to be saved beyond the execution period of the command. The **commandId** parameter is used to define the command for which the respective status is to be saved. The maximum number of saveable command statuses is specified in the **decodingConfig.numberOfMaxBufferedCommandId** configuration data element.	Subsequent check of how command was terminated (e.g. error-free or number of error that occurred)
_removeBufferedMeasuringInputCommandId (V3.2 and higher)	This function ends the saving of **commandId** and the associated command status beyond the execution period of the command.	Explicit deletion of previously saved command IDs

For further information on the system functions, please refer to the *SIMOTION TP CAM Reference Lists*.

Process Alarms

You can predefine local alarm responses via SIMOTION SCOUT.

Note

For more information, refer to the *Motion Control Technology Objects Basic Functions* functional description.

How to configure the alarm response:

1. Double-click **Execution system** in the project navigator below the SIMOTION device. The execution system opens.
2. In the execution level tree, select **SystemInterruptTasks > TechnologicalFaultTask**.
3. Then click the **Alarm Response** button in the displayed window. The **Alarm Response** window appears. You can configure the alarm response for every TO here.

A system variable **error** indicates that a technology alarm has been generated. The response to the alarm is displayed in the **errorReaction** variable.

Table 4-192 Possible alarm responses

Alarm Response	Description	Application
NONE	No response	-
DECODE_STOP	Command processing is aborted, the current measuring function remains active. Further processing on the technology object can continue after **_resetMeasuringInput** or **_resetMeasuringInputError**.	The TO measuring input can only be reactivated after the error has been acknowledged.
MEASURING_INPUT_DISABLE	Stop and abort of all commands. Further processing on the technology object can continue after **_resetMeasuringInput** or **_resetMeasuringInputError**.	The TO measuring input can only be reactivated after the error has been acknowledged.

Measuring input menus

Measuring Input technology object menu

Grayed-out menu functions cannot be selected. The menu is only active if a Measuring Input TO window is active in the working area.

You can select the following functions:

Table 4-193 Measuring input menu

Function		Significance/Note
Close		Select **Close** to close the configuration window for the measuring input that is open in the working area.
Properties		Select **Properties** to display the properties of the measuring input selected in the project navigator.
Configuration		Select **Configuration** to define the configuration data for the measuring input.
Default		Select **Default** to define the values for the system variables of the measuring input.
Expert		
	Expert list	Select **Expert list** to open the expert list for the selected measuring input. The configuration data and system variables can be displayed and changed in this list.
	Configure units	Select **Configure units** to open the **Configure units of the object** window in the working area. You can configure the units used for the selected object here.

Measuring input TO context menu

Grayed-out functions in the context menu cannot be selected.

You can select the following functions:

Table 4-194 Measuring input context menu

Function		Significance/Note
Open configuration		Select **Open configuration** to display the window for configuring the measuring input in the working area. Enter the configuration data of the measuring input in this window.
Expert list		Select **Expert list** to open the expert list for the selected measuring input. The configuration data and system variables can be displayed and changed in this list.
Cut		Select **Cut** to remove the selected object and save it to the clipboard.
Copy		Select **Copy** to copy the selected object. It is stored in the clipboard.
Paste		Select **Paste** to insert the measuring input stored in the clipboard.
Delete		Select **Delete** to delete the selected measuring input. The entire data of the measuring input is deleted permanently.
Rename		Use **Rename** to rename the object selected in the project navigator. Note that with name changes, name references to this object are not adapted.
Expert		
	Insert script folder	**Insert script folder** enables you to insert a folder below the TO. You can create scripts in this folder in order to, for example, automate the configuration.
	Import object	**Import object** imports the data of a SIMOTION object from another project which was previously created with a selective XML export. You cannot import the entire project, only the data of the SIMOTION object.
	Save project and export object	**Save project and export object** exports selected data of the selected object in XML format. This data export can then be reimported into other projects. Only the data of the selected object, not the entire project, is exported.
Print		Select **Print** to print the configuration of the measuring input. All system variables and configuration data with the associated values are printed.
Print preview		Select **Print preview** to open the preview of the measuring input data to be printed.

Function	Significance/Note
Default	Select **Default** to define the values for the system variables of the measuring input.
Properties	Select **Properties** to display the properties of the measuring input selected in the project navigator.

4.4 Additional technology objects

Preface

This **document** is part of the **Description of System and Functions** documentation package.

Validity range

This manual applies to SIMOTION SCOUT in association with the SIMOTION Cam or Cam_ext technology package from product version 4.4.

Chapters in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- **Fixed Gearing** (Part I)
  Function of fixed gearing technology object
- **Addition Object** (Part II)
  Function of addition objects
- **Formula Object** (Part III)
  Function of formula objects
- **Sensor** (Part IV)
  Function of sensor technology object
- **Controller Object** (Part V)
  Function of controller objects
- **Temperature Controller** (Part VI)
  Function of temperature controllers
- **Index**
  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4.3:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

Open source software

Third-party software - License conditions and copyright notes

Copyright notes for the third-party software contained in this product, in particular the open source software, as well as the applicable license conditions, can be found in the READ_OSS.ZIP file on the SIMOTION D CF card or in the corresponding firmware files.

Special note for resellers

The notes and the license conditions contained in the READ_OSS.ZIP file must be passed on to the purchaser in order to avoid the reseller and purchaser from violating the license conditions.

Source code availability

Some license terms of third-party software components used in this product may require us to provide you with the source code and other information for those components. You can find this information directly on or with the product (e.g. on mass storage devices, DVD). If this is not possible for technical reasons, Siemens will be happy to send you this OSS source code in exchange for reimbursement of the processing costs. Please contact the address provided at the end of this section.

Siemens AG

Digital Factory Customer Services

DI CS SD CCC TS

Gleiwitzer Str. 555

D-90475 Nuremberg, Germany

Internet (https://support.industry.siemens.com/cs/ww/en/ps)

Tel.: +49 911 895 7222

4.4.1 Fundamental safety instructions

4.4.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.4.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.4.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.4.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.4.2 Part I - Fixed gearing

4.4.2.1 Overview of Fixed Gearing

Function overview

The **Fixed gear** technology object enables you to implement *fixed synchronous operation* (*without* synchronization/desynchronization) based on a specified gear ratio.

A Fixed gear converts an input variable into an output variable with a configured transmission ratio (gear ratio).

A Fixed gear is interconnected to a motion vector on the input side and the output side.

The basic functionality is the multiplication of the input vector by the configured gear ratio. The individual vector components are thereby multiplied by the gear factor.

Both absolute and relative synchronous operation, with or without offset are possible.

Offset changes, gear changes, and input changeovers are directly enabled.

An on-the-fly master value change is possible (direct transition).

No limiting operations, no transition phases, and no corrections (except the offset value) are taken into account.

You can connect the output interface to a position axis and activate the acceptance of the output values of the fixed gear on the axis with the TO functions _runpositionbasedmotionin or _runvelocitybasedmotionin.

Application

A **Fixed gear TO** can, for example, be used as follows:

- To make allowance for diameters in a master variable
- To implement a fixed gear ratio without clutch
- As a parallel gear on the master,
  slaves are "hung" or "unhung".
  Thus, the gear is always synchronous with the master.
  Example: A paper web also runs in synchronism with the master.

Interconnection

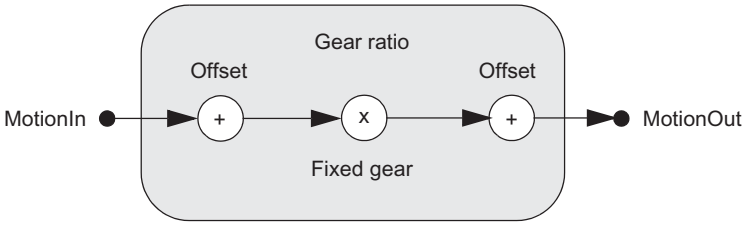


Figure 4-273 Fixed gear object model

A Fixed gear object has one input and one output.

The input vector is type MotionIn and the output vector type MotionOut. These vectors comprise the components distance (s), velocity (v) and acceleration (a).

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Gear ratio

The **gear ratio** can be specified selectively as a rational value (LREAL) or as a ratio of two 32-bit numbers (DINT) in the form of a **numerator/denominator**.

The **gear ratio** can be changed using functions or commands.

Offset

An offset can be specified on both the input side and output side.

An offset only applies in position-related synchronous operation.

The offset is specified in the set user-defined unit.

It is possible to adjust the slave position by setting the master or slave offset. The difference between the two is as follows:

- **Master offset**: The gear ratio is included in the offset.
- **Slave offset**: The gear ratio is *not* included in the offset.

Units

Units can be set on the Fixed gear TO and they apply to both the input and output side.

Example: Axis 1 - Fixed gear - Axis 2

- The units are set to [mm] for axis 1.
- The units are set to [m] for the Fixed gear.
- The units are set to [m] for axis 2.

This means, for example, that a position of three millimeters on axis 1 corresponds to three meters on the input interface of the Fixed gear.

4.4.2.2 Configuring the fixed gearing

Creating fixed gearing

Fixed gearing technology objects are stored across-the-board for a device in the **TECHNOLOGY** folder. They can be interconnected with suitable technology objects of the device.

Proceed as follows

1. To create a new Fixed Gearing TO, double-click **Insert fixed gearing** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing Fixed Gearing TO using the clipboard and insert it under another name.

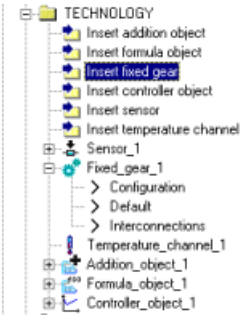


Figure 4-274 Representation of fixed gears in the project navigator

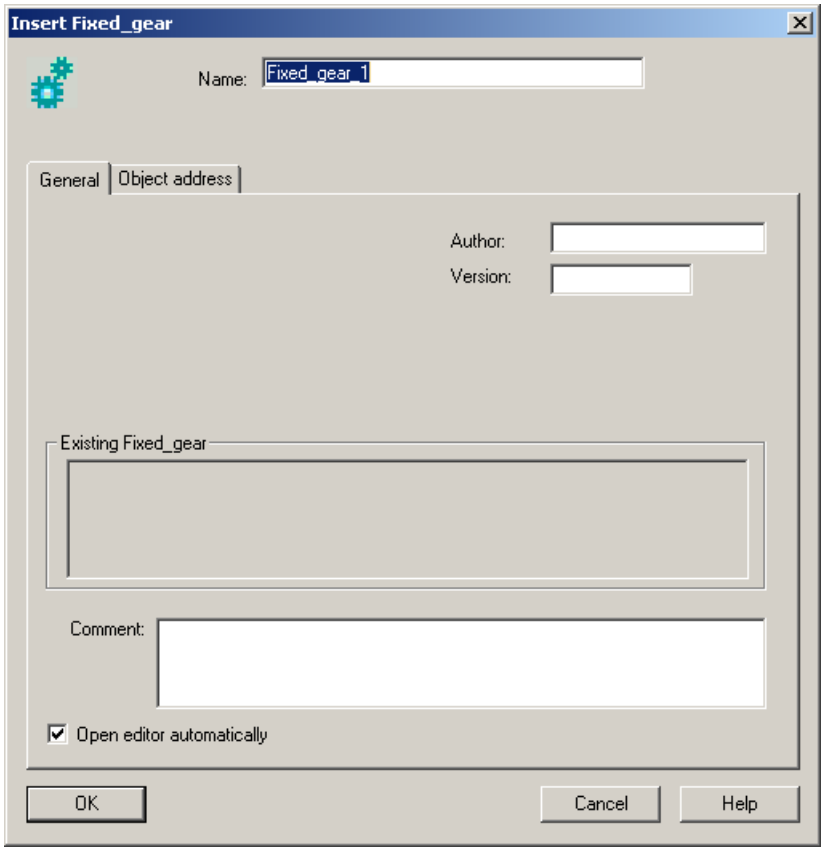


Figure 4-275 Inserting a fixed gear

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new fixed gear TO will be inserted under **TECHNOLOGY**.

Assigning parameters/defaults for fixed gearing

Proceed as follows

In the project navigator, double-click **Defaults** under the object.

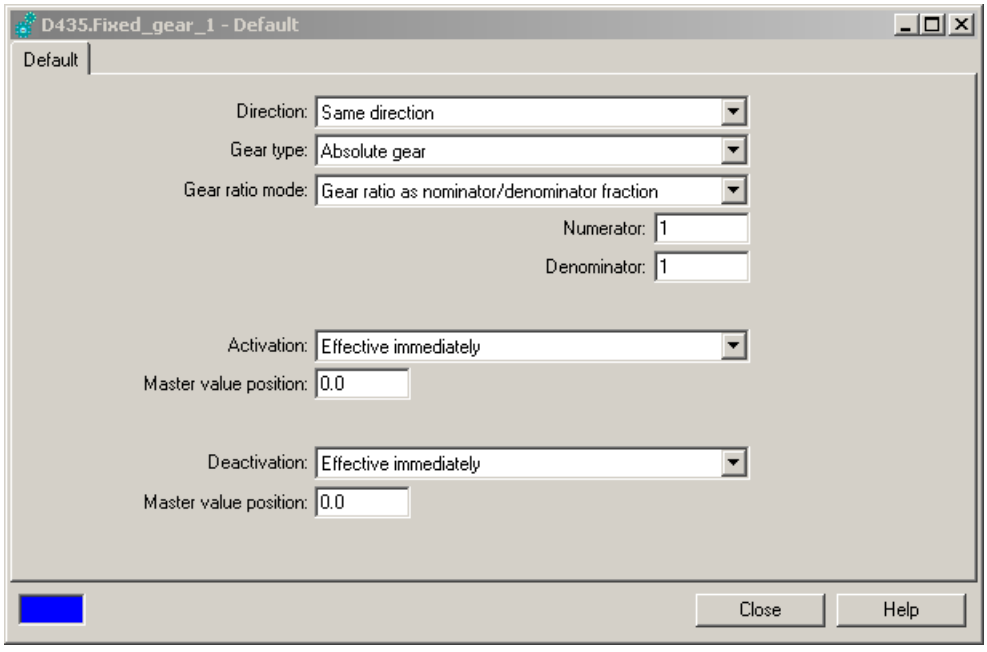


Figure 4-276 Fixed gear TO: Defaults

Fixed gearing - defaults

Specify the parameters (defaults) for calling the fixed gear in this window (**_enableFixedGearing** or **_disableFixedGearing**).

If you do not make any additional specifications during programming, these default values will be used.

You can set the following parameters:

Table 4-195 Fixed gear TO: Parameters that can be set for default purposes

Field/Button	Explanation/Instruction
Direction	Here, you specify the direction of the gearing.
Gear type	Here, you select the gear type (absolute or relative).
Gear ratio mode	Here, you specify the gear ratio mode. Depending on the selected mode, additional parameters are displayed as a rational value (`LREAL`) or as a ratio of two 32-bit numbers (`DINT`) in the form of a numerator/denominator.
Numerator	Here, you can enter the numerator of the gear ratio in the form of a numerator/denominator ratio.
Denominator	Here, you can enter the denominator of the gear ratio in the form of a numerator/denominator ratio.
Activation	Here, you specify the activation method for fixed synchronous operation.

Field/Button	Explanation/Instruction
Master value position	Here, you enter the position of the master value for activation.
Deactivation	Here, you specify the deactivation method for fixed synchronous operation.
Master value position	Here, you enter the position of the master value for deactivation.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION Reference Lists*.

See also

Function overview (Page 1975)

Overview of commands (Page 1983)

Configuring a fixed gearing

Procedure

In the project navigator, double-click **Configuration** under the object.

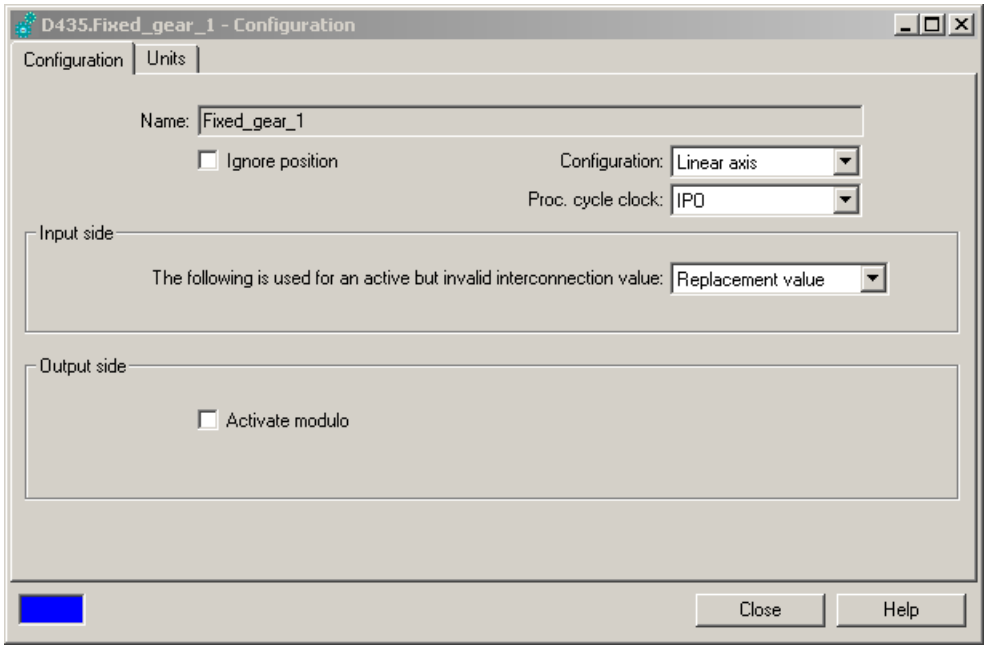


Figure 4-277 Configuring a fixed gearing

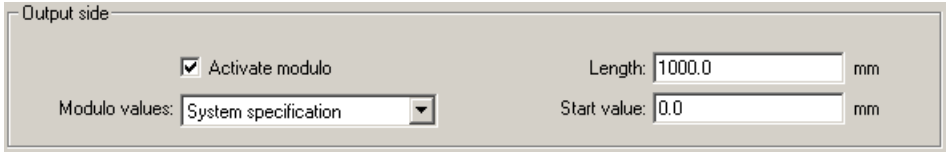


Figure 4-278 Configuring modulo properties of the output of a fixed gear

Fixed gearing - configuration

In this window, specify the configuration of the input and the output.

You can set the following parameters:

Field/Button	Explanation/instructions
Name	Display: name of the fixed gearing
Ignore position	If the check box is selected, the velocity will be used as the gearing basis.
Configuration	Here, you set the units: linear or rotary These apply to the output side. On the input side, the values are applied with their units, or are displayed without units.
Processing cycle clock	Here, you select the IPO cycle clock or IPO_2 cycle clock.
Input side	Here, you specify the validity of the input values or default values.
Output side	Here, you specify the modulo properties of the output. If the check box is selected, the parameters will be displayed.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 1975)

Overview of commands (Page 1983)

Gearing basis

The **gearing basis** (configuration data element **motionBase**) must be specified, i.e.,

- **Position** (POSITION):
  Multiplication of position, velocity, acceleration (s, v, a)
- **Velocity** (VELOCITY):
  Multiplication of velocity, acceleration (v, a)

All components are present in the output vector. When **Velocity** is specified as the gearing basis, the position component in the output vector is set to zero.

Any existing offset is applied to the variable specified by the motion basis.

Modulo properties

The output-side modulo setting is derived from the system or can be set using configuration data element **MotionOut.modulo**.

- The modulo setting is derived from the downstream TO with **system default** (**SYSTEM**) if this TO and its modulo properties can be uniquely determined. (A formula object, for example, does not have any modulo properties.)
- In all other cases, the modulo property set on the TO is used (**DIRECT**).

Interconnecting a fixed gearing

Procedure

1. In the project navigator, double-click **Interconnections** under the object.

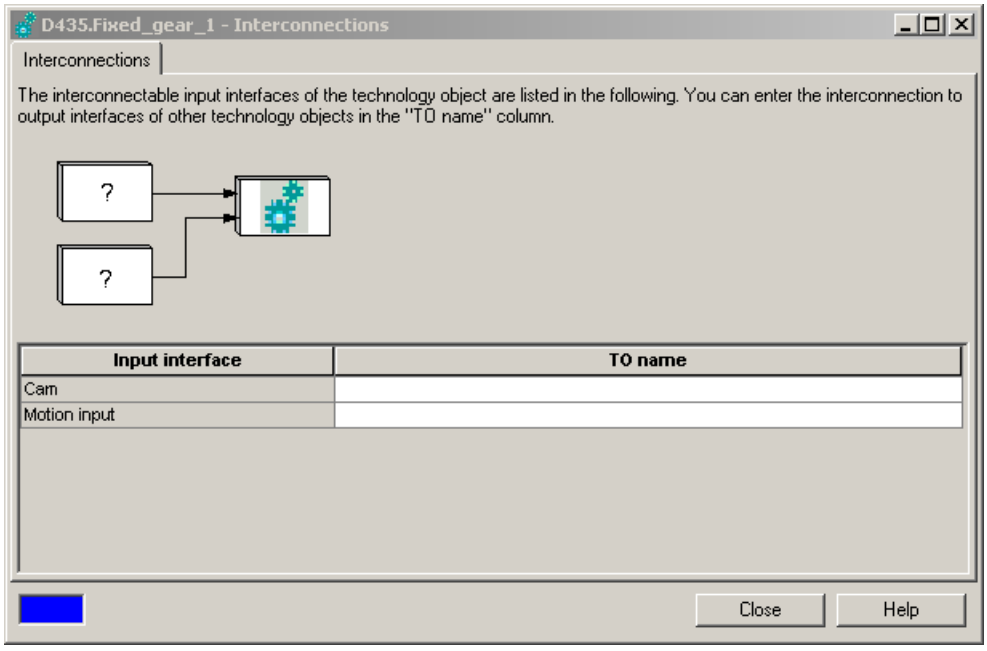


Figure 4-279 Interconnecting a fixed gearing

In this window, interconnect the input of the fixed gear (with axes, for example).

2. To do so, click in the corresponding input field and then select the desired object. (The objects must have already been created.)

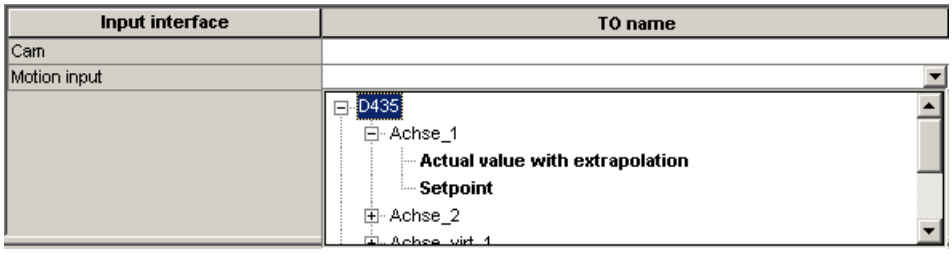


Figure 4-280 Interconnecting a fixed gear with an axis

Note

The interconnection with a cam profile is intended for a later software version and has no function in Version V3.2.

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

4.4.2.3 Programming fixed gearing/references

Programming

Overview of commands

Table 4-196 Fixed gearing TO: Overview of commands

Command	Functionality
_enableFixedGearMotionIn	Enabling/disabling of the input vector
_disableFixedGearMotionIn	
_enableFixedGearing	Activation/deactivation of fixed gear
_disableFixedGearing	
_setFixedGearingOffset	Set offset on the input side/output side
_setFixedGearMaster	Switch over master value
_resetFixedGear	Reset gear
_resetFixedGearError	Reset error
_resetFixedGearConfigDataBuffer	Delete collected configuration data
_bufferFixedGearCommandId	Store CommandId and command status temporarily
_removeBufferedFixedGearCommandId	Delete CommandId
_getStateOfFixedGearCommand	Read out command status
_getFixedGearErrorNumberState	Read out error number status

Note

For a complete list of all commands and their syntax, the system variables, and error messages, please see the SIMOTION Reference Lists.

Commands

Activation/deactivation of the input vector

Note

The interconnection interfaces on the input side are enabled/disabled separately from the functionality.

If the input values are inactive, the default values are applied.

- **_enableFixedGearMotionIn**: Use interconnected input values
- **_disableFixedGearMotionIn**: Disable interconnected input values;
  the default values will be used (**motionInDefault** and **motionOutDefault** system variables).

The commands are synchronous.

The input and the output are active following controller power-up, if they are interconnected. A status check is possible via the **motionIn.state** and **motionOut.state** system variables.

If the input and the output are not interconnected, an error is output. (The alarm is not generated until _enable...In.)

Enabling/disabling of fixed gearing

- **_enableFixedGearing**: Enable gearing functionality
  The command is synchronous.
  - The gearing is started with **_enableFixedGearing** without a transition function.
  - The gearing ratio is specified in the function parameter.
  - The **startPosition...** parameter acts only if **Position** is selected as the gearing basis.
- **_disableFixedGearing**: Disable gearing functionality
  - The gearing is started with **_disableFixedGearing** without a transition function.
  - The values of the interconnection values on the output side depend on the **motionOutBehaviourMode** parameter of the **_disableFixedGearing()** command.

The following can be set for **_disableFixedGearing**:

- Freeze value
- Define default value
- Input a value of "0.0"

Absolute or relative gearing

The fixed gearing can be set to absolute or relative gearing with the **gearingType** parameter of the **_enableFixedGearing** command.

- With *absolute synchronous operation* (**ABSOLUTE**), the input and output values are interpreted "absolutely", i.e., directly coupled.
- With *relative synchronous operation* (**RELATIVE**), the input and output are interpreted "relatively", i.e., coupled with an offset.

Direction

The gear ratio can be set to positive or negative (corresponding to a negative gear ratio) with the **direction** parameter of the **_enableFixedGearing** command.

- **POSITIVE** means that the axes are running in the same direction.
- **NEGATIVE** means that the axes are running in opposite directions.

Validity of input values or default values

The interconnected values are enabled/disabled using a command (**_enableFixedGearMotionIn**/**_disableFixedGearMotionIn**).

Configuration data element **MotionIn.behaviorByInvalidInterface** can be used to specify which of the following values is used if the interconnection values are enabled but invalid:

- Most recent valid value (**LAST_VALID_INTERFACE_VALUE**) or
- Replacement values (**DEFAULT_VALUE**)

(Following system startup, the most recent valid value is 0).

When the interconnection interface on the input side is disabled, the default value is applied.

Setting offset on the input side/output side

_setFixedGearingOffset: Shifts the gearing relative to the master value or slave value.

The offset can be changed over using the **activationMode** parameter of the **_setFixedGearingOffset** command. The changeover applies as follows:

- *For the next synchronous operation and all subsequent synchronous operations* if **DEFAULT_VALUE** is set
- *For the current synchronous operation only* if **ACTUAL_VALUE** is set
- *For the current synchronous operation and all subsequent synchronous operations* if **ACTUAL_AND_DEFAULT_VALUE** is set

An offset with reference to the current synchronous operation is retained only for the duration of the **_enableFixedGearing**, that is, the offset is directly assigned to the active **_enableFixedGearing** command.

The current offset is applied without a compensatory motion, i.e., it is applied directly.

Switch over master value

The active master can be switched over online by means of **_setFixedGearingMaster**.

System variables

The input and output values of a fixed gearing TO can be read out via system variables.

Table 4-197 Fixed gearing TO: System variables

System variable			Type	Description	Remark
motionIn			StructFixedGearMotionIn	input vector	The input vector must be interconnected. Can only be read
	value		StructMotionVector	Components of the input vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	state		EnumActiveInactive	Status	
	lastValidInterface value		StructMotionVector	Last valid values	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	validity		EnumInterfaceValueDefaultValue	Validity	
motionInDefault			StructMotionVector	Default values for input vector	
motionOut			StructFixedGearMotionOut	Output vector	
	value		StructMotionVector	Components of the output vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	fixedGearValue		StructMotionVector	Function result gearing	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
motionOutDefault			StructMotionVector	Default values for output vector	Deactivate with function, see **_disable FixedGearing**

Local alarm response

Technological alarms

The standard technology object alarms, e.g., interconnection error, illegal parameter, are output.

Local responses

If an error occurs, the following local responses are possible:

- No response (**NONE**)
- Stop TO processing (**DISABLE**)
- Stop command decoding (**DECODE_STOP**)

The local reactions can be set under **TechnologicalFaultTask** in the alarm configuration.

Menus

Fixed gearing - menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function		Meaning/Note
Close		Use this function to close the active window in the working area.
Properties		Properties displays the properties of the fixed gearing selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration		This function opens the configuration for the fixed gearing selected in the project navigator. In this window, specify the configuration of the input and the output.
Factory setting (default)		This function opens the defaults for the fixed gearing selected in the project navigator. In this window, specify the parameters for the call of the fixed gearing object (**_enableFixedGearing** or **_disableFixedGearing**).
Interconnections		This function opens the interconnections for the fixed gearing selected in the project navigator. In this window, interconnect the input of the fixed gearing object, e.g. to axes.
Expert		This function opens the submenu for the expert settings.
	Expert list	This function opens the expert list for the fixed gearing selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.

Fixed gearing - context menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function		Meaning/Note
Open configuration		This function opens the configuration for the fixed gearing selected in the project navigator. In this window, specify the configuration of the input and the output.
Factory setting (default)		This function opens the defaults for the fixed gearing selected in the project navigator. In this window, specify the parameters for the call of the fixed gearing object (**_enableFixedGearing** or **_disableFixedGearing**).
Interconnections		This function opens the interconnections for the fixed gearing selected in the project navigator. In this window, interconnect the input of the fixed gearing object, e.g. to axes.
Expert		This function opens the submenu for the expert settings.
	Expert list	This function opens the expert list for the fixed gearing selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.
	Import object	Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.
	Save project and export object	Use Save project and export object to open a window for an XML export. You can define the parameters for the XML export in this window.

4.4.3 Part II - Addition object

4.4.3.1 Overview of Addition Object

Function overview

The **addition object** technology object enables you to add up to four input vectors for one output vector.

The input vectors are type MotionIn and the output vector is type MotionOut.

There are no limitations on the addition object. Transitional phases are not taken into account for non-continuous input signals.

Application

An **addition object** can, for example, be used as follows:

- For adding superimpositions/offsets in the main signal path, e.g., color register, cut-off register to the paper web

Interfaces

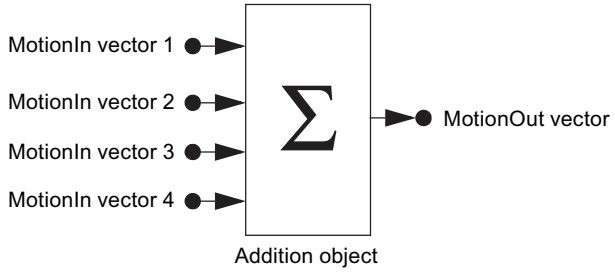


Figure 4-281 Object model for addition object

An addition object calculates the sum of the four input vectors.

The values of the respective input vectors are generated from the interconnection values, the most recent valid interconnection values, or the replacement values.

An interconnected input vector is activated/deactivated using a command; otherwise, the replacement value is applied.

The output vector can be activated or deactivated (frozen).

Interconnection

The first input must be interconnected, while the other inputs and the output may be interconnected.

The first input vector determines which technology variables are added up and are relevant in the output vector.

Non-interconnected input vectors can be specified by means of system variables from the user program.

The input vectors can be interconnected *once only* ('single point') and cannot be changed over; the output vector can be interconnected any number of times ('multi-point').

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Modulo properties

The variables to be added can be modulo variables.

The output vector can be a modulo variable.

The input modulo lengths and the output modulo length can be different.

The modulo relationships can be specified in the addition object on the output side.

The output modulo length is derived from the output-side interconnection, or it can be directly specified (configuration setting).

Subtraction

Subtractions are possible for each input through configuration (inversion).

Units

Units can be set on the addition object, they apply to both the input and output side.

Example: Axis 1, 2 - Addition object - Axis 3

- The units are set to [mm] for axes 1 and 2.
- The units are set to [m] for the addition object.
- The units are set to [m] for axis 2.

This means, for example, that a position of three millimeters on axis 1 corresponds to three meters on the input interface of the addition object.

4.4.3.2 Configuring an Addition Object

Creating an addition object

Addition objects are stored across-the-board for a device in the **TECHNOLOGY** folder. They can be interconnected with suitable technology objects of the device.

Proceed as follows

1. To create a new addition object, double-click **Insert addition object** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing addition object to the clipboard and then insert it under another name.

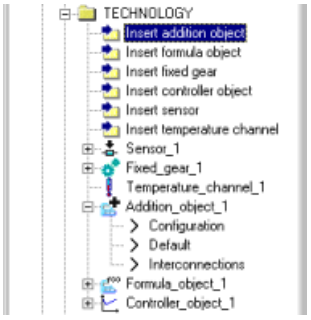


Figure 4-282 Representation of addition objects in the project navigator

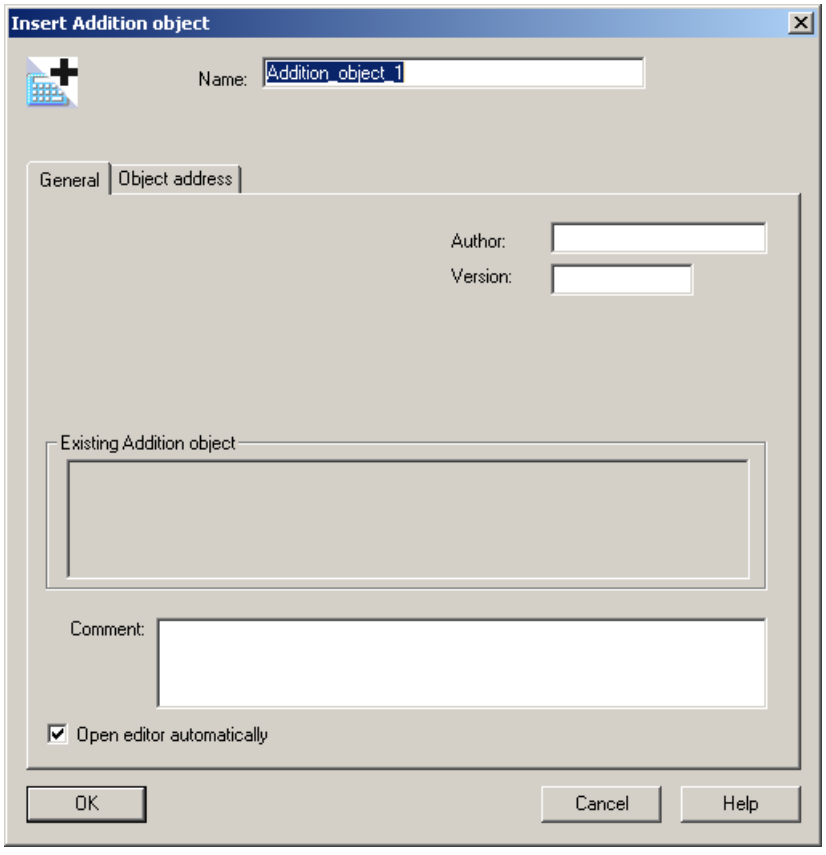


Figure 4-283 Inserting an addition object

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new addition object will be inserted under **TECHNOLOGY**.

Assigning parameters/defaults to an addition object

Procedure

In the project navigator, double-click **Defaults** under the object.

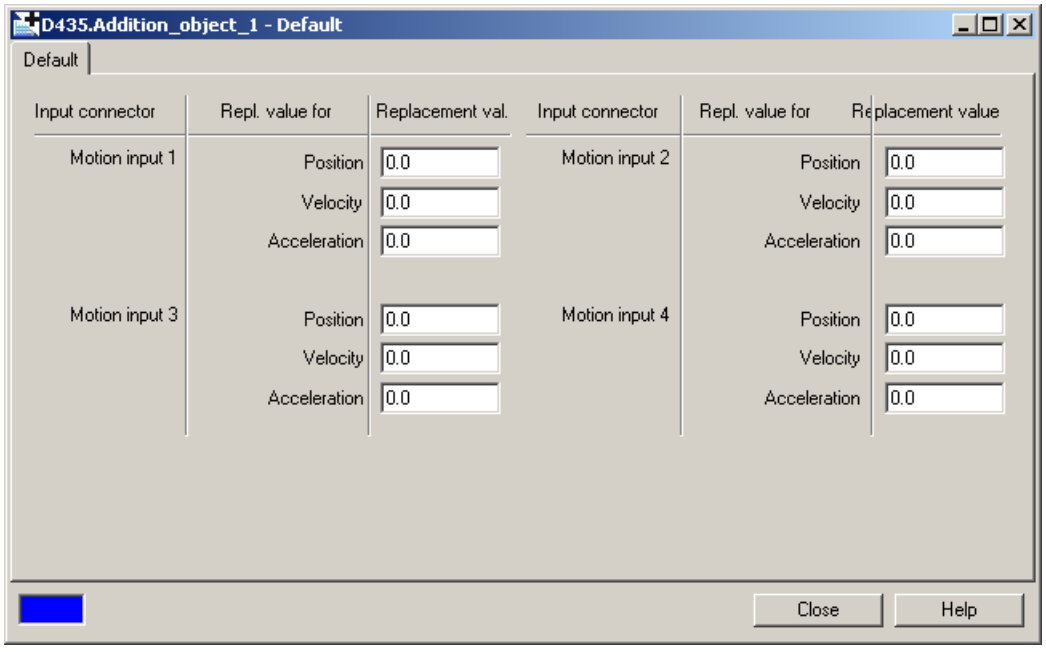


Figure 4-284 Addition object: Defaults

Specify the replacement values (defaults) for calling the addition object in this window (**_enableAdditionObject** or **_disableAdditionObject**).

Addition object - defaults

Specify the input default values (defaults) for calling the addition object in this window (_enableAdditionObject or _disableAdditionObject).

Field/Button	Explanation/instructions
Input connector	Motion inputs 1 to 4
Default value for	Input values are generated from the interconnected values, the most recent valid interconnected values, or the default values. Position, velocity, and acceleration can each be specified with default values.
Default value	Here, you enter the default values.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 1989)

Overview of commands (Page 1997)

Configuring an addition object

Proceed as follows

In the project navigator, double-click **Configuration** under the object.

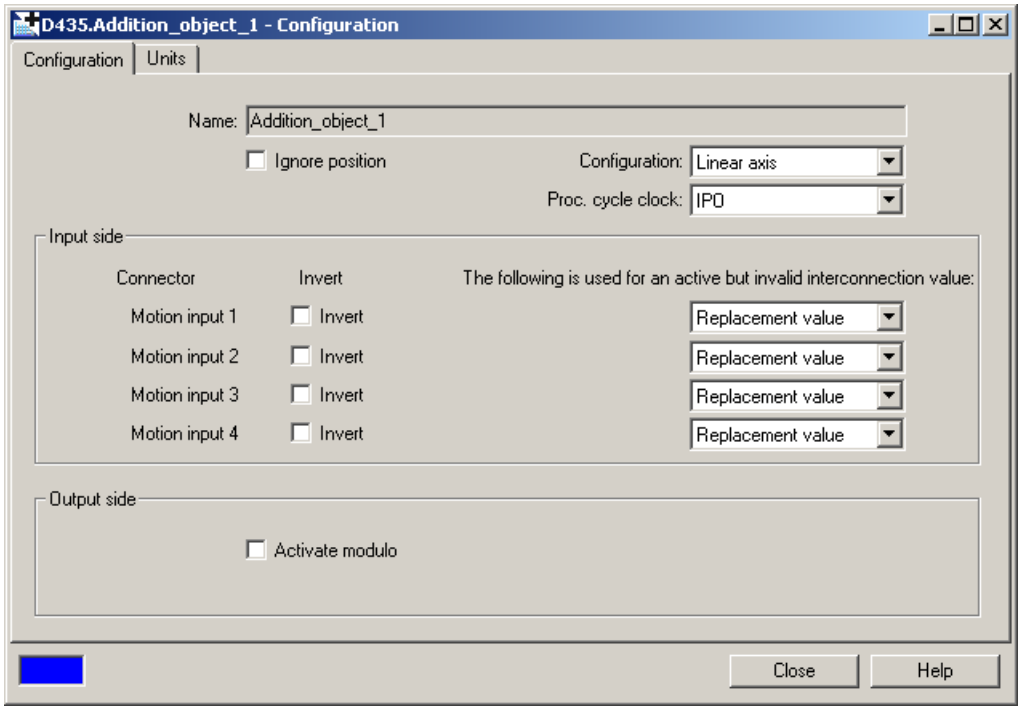


Figure 4-285 Configuring an addition object

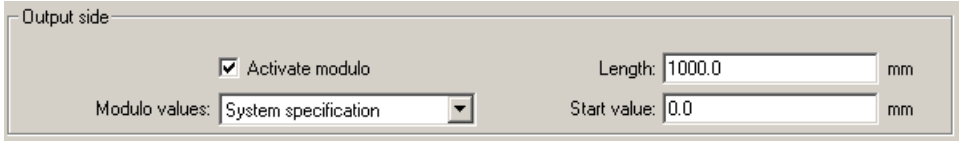


Figure 4-286 Configuring modulo properties of the output of an addition object

Addition object - configuration

In this window, specify the configuration of the inputs and the output.

You can set the following parameters:

Table 4-198 Addition object technology: Parameters that can be set for configuration purposes

Field/button	Meaning/instruction
Name	Display: name of addition object
Ignore position	If the check box is selected, the velocity will be used as the addition basis.
Configuration	Here, you set the units: linear or rotary
Processing cycle clock	Here, you select the IPO-, IPO_2, IPO_FAST, SERVO or SERVO_Fast cycle clock.
Input side	Here, you specify the input inversion and the validity of the input values/substitute values.
Output side	Here, you specify the modulo property of the output. If the check box is selected, the parameters will be displayed.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 1989)

Overview of commands (Page 1997)

Addition basis

The **addition basis** (configuration data element **motionBase**) must be specified, i.e.,

- **Position** (POSITION):
  Addition of position, velocity, acceleration (s, v, a)
- **Velocity** (VELOCITY):
  Addition of velocity, acceleration (v, a)

All input vectors are given the same addition basis.

Only the relevant vector components are then valid in the output vector. The other components are set to zero.

Comment:

A component can be added up selectively in the motion vector, e.g. torque, using a formula object.

Units

The following basic units are available for representing lengths:

- Linear
- Rotary
- No unit

The unit settings apply on the output side. On the input side, the values are added without consideration as to agreement of the units.

That is, all system variables acting on the output side have the configured unit, while all system variables acting on the input side are unitless.

Modulo properties

The output-side modulo setting is derived from the system or can be set using configuration data element **MotionOut.modulo**:

- The "Detect automatically" setting checks the uniqueness of the interconnection(s) on the output side.
  If a unique assignment is possible, the information is used for representing the output vector. If the object interconnected on the output side has a modulo representation, the output vector is already calculated in this modulo representation.
  If a unique assignment is not possible, no adaptation is performed; rather, an error/alarm is output.
- The "Determine from configuration" setting enables a modulo representation to be selected for the output vector itself.

Input inversion

Configuration data element **MotionIn#.invert** can be used to configure a signal inversion for each input prior to addition.

The inversion applies to position, velocity, or acceleration.

The input-side system variable (**motionIn#**) displays the inverted value or writes the value after inversion.

Interconnecting an addition object

Procedure

1. In the project navigator, double-click **Interconnections** under the object.

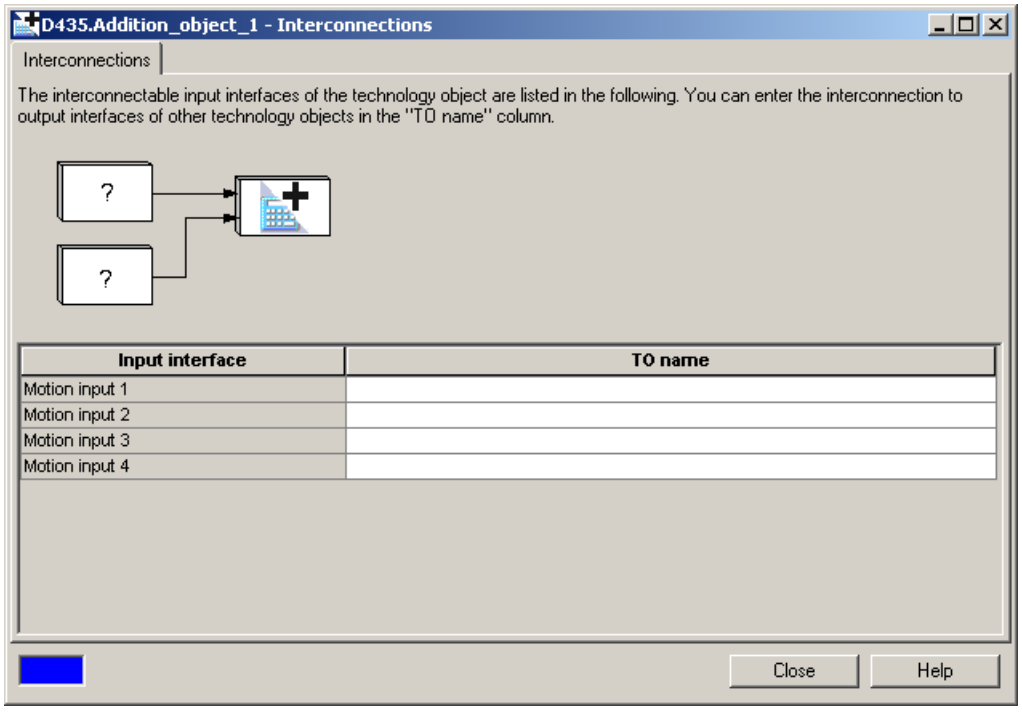


Figure 4-287 Interconnecting an addition object

In the displayed window, interconnect the inputs of the addition object (with axes, for example).

2. To do so, click in the corresponding input field and then select the desired object. (The objects must have already been created.)

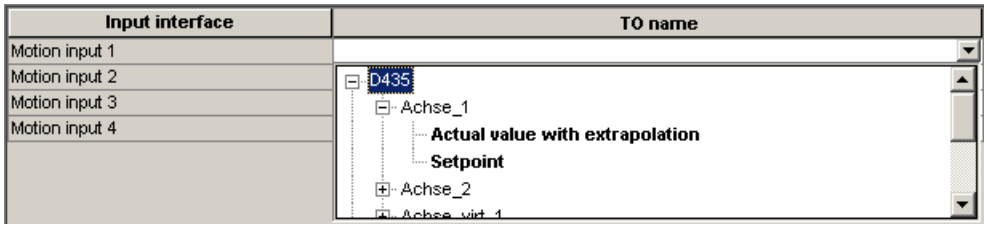


Figure 4-288 Interconnecting an addition object with axes

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

4.4.3.3 Programming an Addition Object/References

Programming

Overview of commands

Table 4-199 Addition object TO: Overview of commands

Command	Functionality
_enableAdditionObjectIn	Enabling and disabling input vectors
_disableAdditionObjectIn	
_changeEnableModeOfAdditionObject	Simultaneous switchover
_resetAdditionObject	Reset addition object
_resetAdditionObjectError	Reset error
_resetAdditionObjectConfigDataBuffer	Delete collected configuration data
_bufferAdditionObjectCommandId	Store CommandId and command status temporarily
_removeBufferedAdditionObjectCommandId	Delete CommandId
_getStateOfAdditionObjectCommand	Read out command status
_getAdditionObjectErrorNumberState	Read out error number status

Note

For a complete list of all commands and their syntax, the system variables, and error messages, please see the SIMOTION Reference Lists.

Commands

Enabling and disabling input vectors

- **_enableAdditionObjectIn**: Enable input vectors
- **_disableAdditionObjectIn**: Deactivate input vectors

The commands are synchronous.

All four input vectors can be enabled/disabled simultaneously with one command.

All of the inputs are always added.

Input values are generated from the interconnected values, the most recent valid interconnected values, or the default values.

- If interconnection values are not enabled, the default values are applied.
- If the interconnection interfaces on the input side are disabled, the most recent valid values or the default values are also used.

The inputs and the output are active following controller power-up if they are interconnected. (The status can be scanned using a system variable.)

If the inputs and the output are not interconnected, an error is output. (The alarm is not generated until _enable...In.)

Validity of input values or default values

Configuration data element **MotionIn#.behaviorByInvalidInterface** can be used to specify which of the following values is used if the interconnection values are enabled but invalid:

- Most recent valid value (**LAST_VALID_INTERFACE_VALUE**)

or

- Replacement values (**DEFAULT_VALUE**)

(Following system startup, the most recent valid value is 0).

System variables

The input and output values of an addition object can be read out via system variables.

Table 4-200 Addition object TO: System variables

System variable			Type	Description	Remark
motionIn1			StructAdditionObjectMotionIn	1. input vector	The first input must be interconnected.
	value		StructMotionVector	Components of 1st input vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	state		EnumActiveInactive	Status	
	lastValidInterface value		StructMotionVector	Last valid values	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	validity		EnumInterfaceValueDefaultValue	Validity	
motionIn1Default			StructMotionVector	Default values for 1st input vector	
motionIn2			StructAdditionObjectMotionIn	2nd input vector	
motionIn2Default			StructMotionVector	Default values for 2nd input vector	
motionIn3			StructAdditionObjectMotionIn	3. input vector	
motionIn3Default			StructMotionVector	Default values for 3rd input vector	
motionIn4			StructAdditionObjectMotionIn	4. input vector	
motionIn4Default			StructMotionVector	Default values for 4th input vector	

System variable			Type	Description	Remark
motionOut			StructAdditionObjectMotionOut	Output vector	
	value		StructMotionVector	Components of the output vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	additionResult		StructMotionVector	Function result addition	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	

Local alarm response

Technological alarms

The standard technology object alarms, e.g. interconnection error or illegal parameter, are output.

Local reactions

If an error occurs, the following local responses are possible:

- No response (**NONE**)
- Stop TO processing (**DISABLE**)

The local reactions can be set under **TechnologicalFaultTask** in the alarm configuration.

Menus

Addition object - menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the addition object selected in the project navigator. You can enter the object name plus author and version in this window.

Function	Meaning/Note
Configuration	This function opens the configuration for the addition object selected in the project navigator. In the displayed window, specify the configuration of the inputs and the output.
Factory setting (default)	This function opens the defaults for the addition object selected in the project navigator. Specify the input default values for calling the addition object in this window (**_enableAdditionObject** or **_disableAdditionObject**).
Interconnections	This function opens the interconnections for the addition object selected in the project navigator. In the displayed window, interconnect the inputs of the addition object, e.g. to axes.
Expert	This function opens the submenu for the expert settings.
Expert list	This function opens the expert list for the addition object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.

Addition object - context menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Open configuration	This function opens the configuration for the addition object selected in the project navigator. In the displayed window, specify the configuration of the inputs and the output.
Factory setting (default)	This function opens the defaults for the addition object selected in the project navigator. Specify the input default values for calling the addition object in this window (**_enableAdditionObject** or **_disableAdditionObject**).

Function		Meaning/Note
Interconnections		This function opens the interconnections for the addition object selected in the project navigator. In the displayed window, interconnect the inputs of the addition object, e.g. to axes.
Expert		This function opens the submenu for the expert settings.
	Expert list	This function opens the expert list for the addition object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.
	Import object	Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.
	Save project and export object	Use Save project and export object to open a window for an XML export. You can define the parameters for the XML export in this window.

4.4.4 Part III - Formula object

4.4.4.1 Overview of Formula Object

Function overview

The **formula object** technology object can be applied in the TO interconnection to LREAL and DINT scalars and to MotionIn and MotionOut motion vectors.

The components of a "generalized motion vector" can be modified individually, but the motion vector is interconnected as a whole.

The formula object is a *stand-alone* technology object, which can be interconnected with other technology objects.

Application

A **formula object** can be used between interconnected objects to modify scalar variables in the main signal path, e.g.:

- Superimposition of torque
- Superimposition of master velocity
- Modification of torque variables B+, B-
- Enabling of torque limitations
- Enabling of torque

Operations

The following operations can be performed (for a complete list, see "Available functions in formulas"):

- Manipulation of scalar variables within the TO interconnection
- Adding/subtracting
- Inverting (changing sign)
- Offsets
- Multiplication
- Division
- Switching, enabling (also using multiplication)
- Limiting (minimum, maximum)
- Logical operations (switch enable)

Interfaces and interconnection

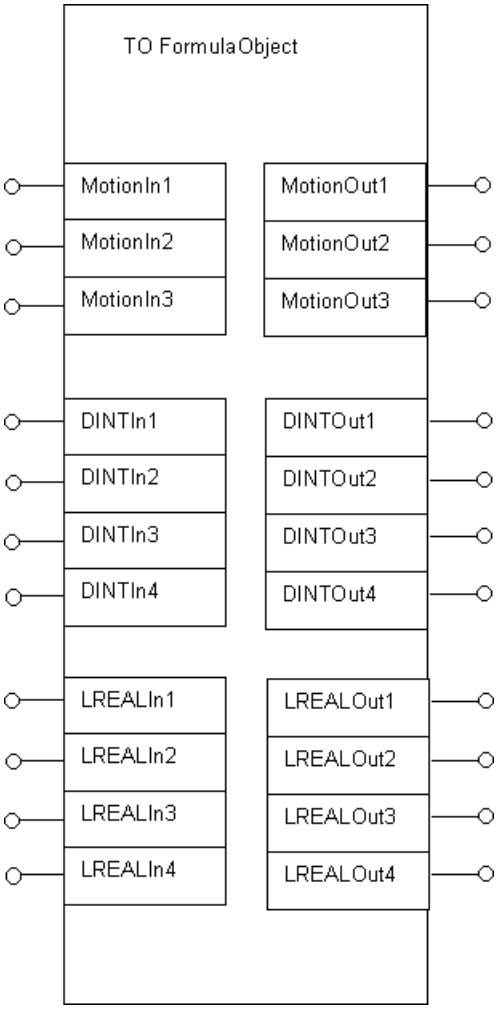


Figure 4-289 Object model for formula object

A formula object has the following inputs/outputs with different formats/types:

- 3 motion vectors/vector components
- 4 scalars `DINT`
- 4 scalars `LREAL`

The interconnection of each input/output is optional and occurs during configuration. However, a check is made to determine whether at least one input is connected (otherwise, an interconnection error is output).

The input interfaces are defined as local connectors, i.e. a device transition does not take place at the interfaces.

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Definition

The formula must be defined in plain text (string with a maximum of 80 characters) with the **_defineFormula** command using (built-in) functions. (For details of possible functions, see "Available functions in formulas (formula operators)".)

In so doing, the associated formula number must be specified.

Mapping rules

The output value at time k can be calculated in relation to the inputs at time k and the outputs at time k-1:

$A_{i,k} = f\ (E_{1,k} \ldots E_{n,k}\ ,\ A_{1,k-1} \ldots A_{m,k-1})$

The following applies here:

- The order of calculation of individual formulas is not relevant.
- The assignment of a formula to an output can be changed online.
- Explicit conversions are not made.
- In the case of vectors, every component must be written to on the output side.
- The output values are retained, if the object is deactivated.
- The output values can be set (e.g. start value for integrator).
- The components of the vector are not kept consistent.

The instructions are calculated sequentially in each cycle.

Units

All scalar values are interpreted as unitless, i.e. all system variables are also unitless.

The following basic units are available for representing lengths (adjustable for vectors):

- Linear
- Rotary

The unit settings apply on the output side. On the input side, the values are used without consideration as to agreement of the units.

That is, all system variables acting on the output side have the configured unit, while all system variables acting on the input side are unitless.

Modulo properties

Modulo functionality is not taken into consideration in the formula object TO.

- Input values are used "as is".
- Output values are transmitted as calculated.

4.4.4.2 Configuring a Formula Object

Creating a formula object

Formula objects are stored across-the-board for a device in the **TECHNOLOGY** folder. They can be interconnected with suitable technology objects of the device.

Proceed as follows

1. To create a new formula object, double-click **Insert formula object** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing formula object to the clipboard and then insert it under another name.

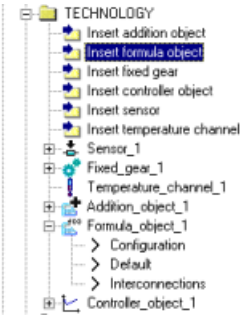


Figure 4-290 Representation of formula objects in the project navigator

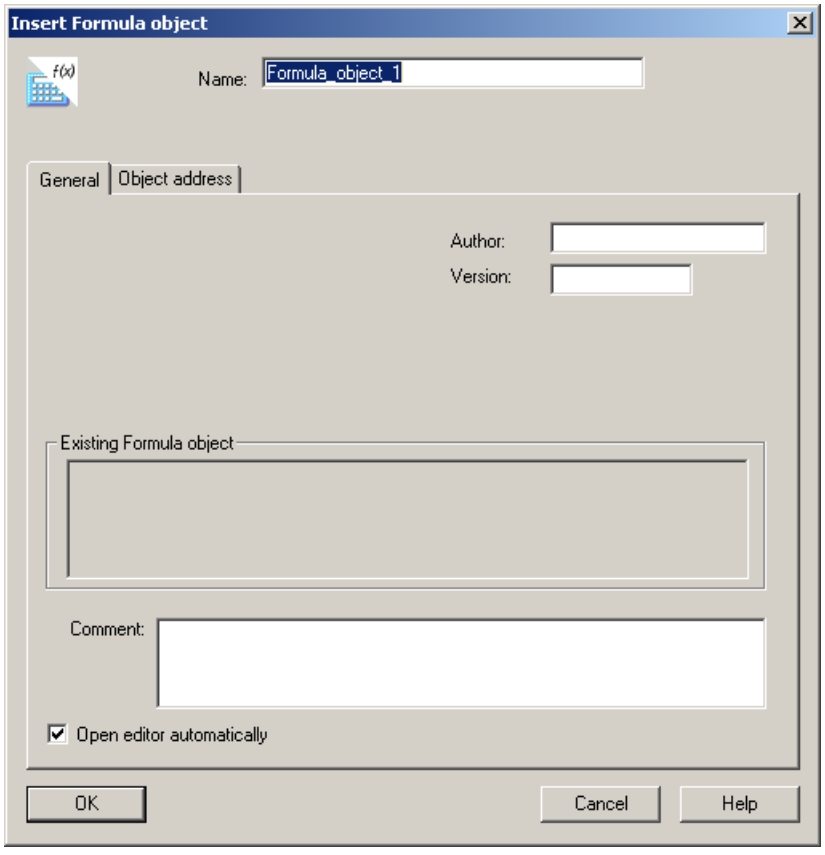


Figure 4-291 Inserting a formula object

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new formula object will be inserted under **TECHNOLOGY**.

Assigning parameters/defaults to a formula object

Proceed as follows

In the project navigator, double-click **Defaults** under the object.

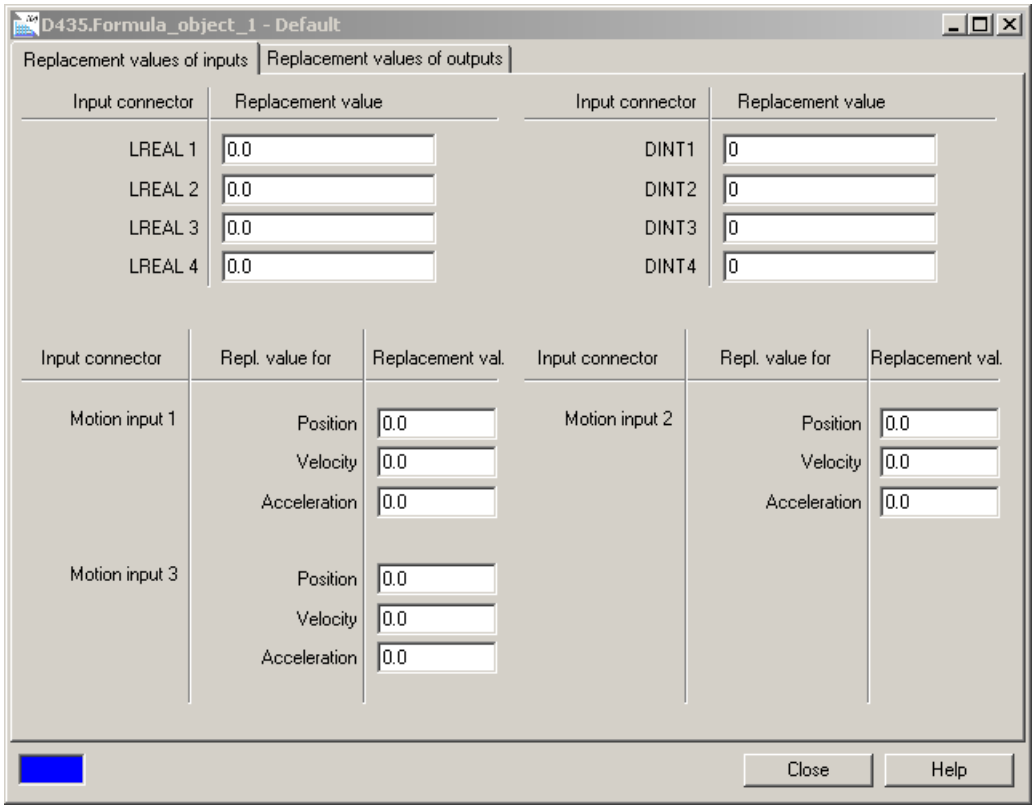


Figure 4-292 Formula object: Defaults

Formula object - defaults

Specify the input and output default values (defaults) for calling the formula object in this window (**_enableFormulaObjectIn** or **_disableFormulaObjectIn**).

You can set the following parameters:

Table 4-201 Formula object TO: Parameters that can be set for default purposes

Field/Button	Explanation/Instruction
Input connector/ Output connector	Motion input/output 1 to 3
Replacement value for	Input/output values are generated from the interconnected values, the most recent valid interconnected values, or the replacement values. Replacement values can be specified for each of the following input/output variables: • 3 motion vectors/vector components • 4 scalars DINT • 4 scalars LREAL
Replacement value	Here, you enter the replacement values.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION Reference Lists*.

See also

Function overview (Page 2001)

Overview of commands (Page 2013)

Configuring a formula object

Proceed as follows

In the project navigator, double-click **Configuration** under the object.

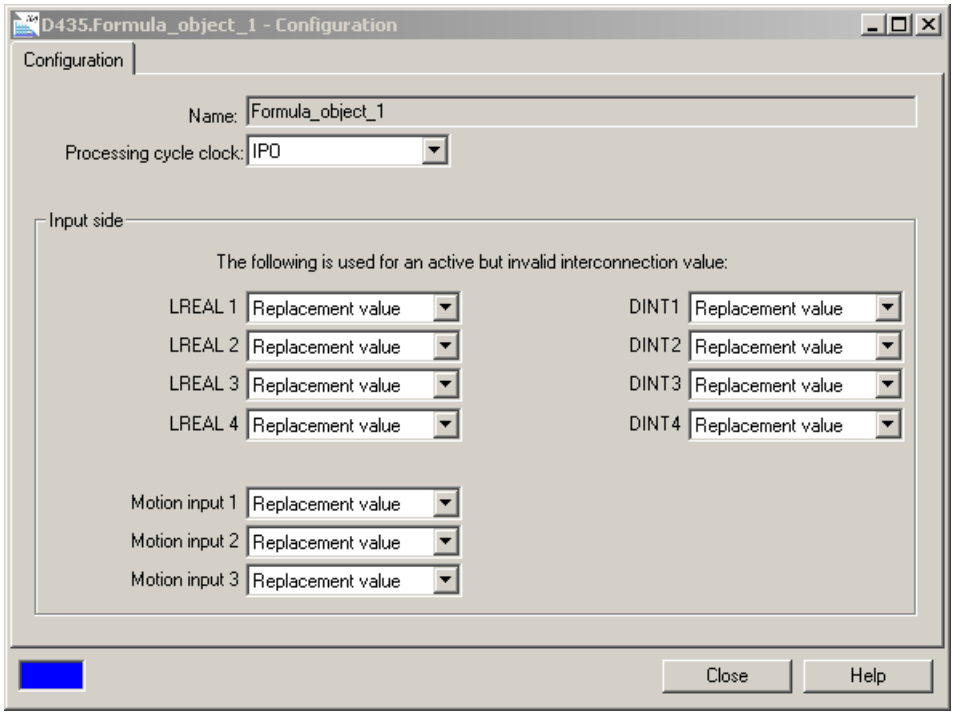


Figure 4-293 Configuring a formula object

Formula object - configuration

In this window, specify the configuration of the inputs and outputs.

You can set the following parameters:

Table 4-202 Formula object TO: Parameters that can be set for configuration purposes

Field/Button	Explanation/instructions
Name	Display: name of formula object
Processing cycle clock	Here, you select the IPO cycle clock or IPO_2 cycle clock.
Input side	Here, you specify the validity of the input values or values. The validity can be specified for each of the following input variables: • 3 motion vectors/vector components • 4 scalars DINT • 4 scalars LREAL

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 2001)
Overview of commands (Page 2013)

Interconnecting a formula object

Procedure

1. In the project navigator, double-click **Interconnections** under the object.

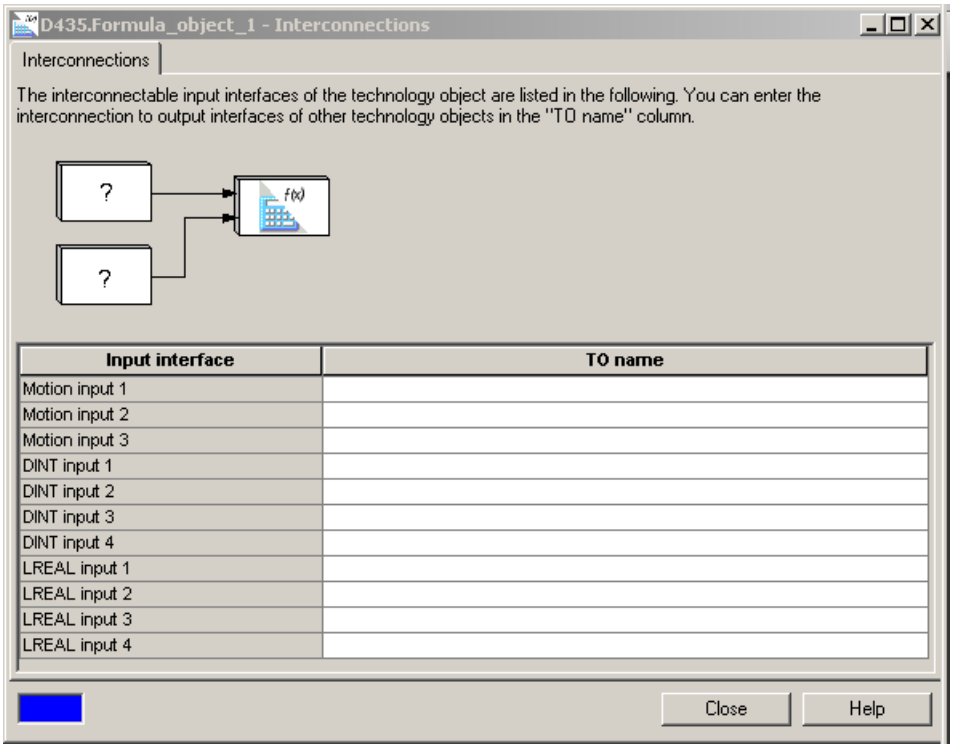


Figure 4-294 Interconnecting a formula object

In this window, interconnect the inputs of the formula object (with axes, for example).

2. To do so, click in the corresponding input field and then select the desired object. (The objects must have already been created.)

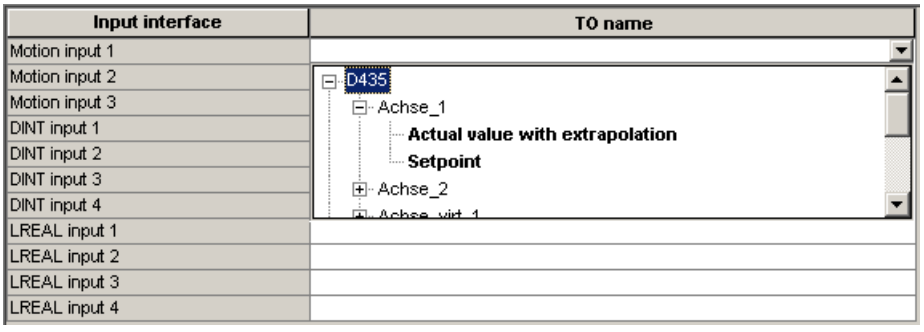


Figure 4-295 Interconnecting the motion input of a formula object

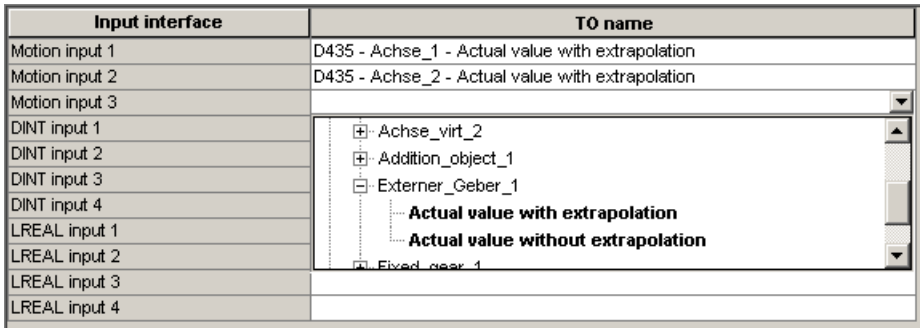


Figure 4-296 Interconnecting the scalar input of a formula object

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Defining a formula

A formula is defined by programming it in the user program.

Definition

The formula must be defined in plain text (string with a maximum of 80 characters) with the **_defineFormula** command using (built-in) functions. (For details of possible functions, see "Available functions in formulas (formula operators)".)

In so doing, the associated formula number must be specified.

The formula syntax is checked when the formula is defined (on execution of command _defineFormula).

See also

Rules for definition of formulas (Page 2016)

4.4.4.3 Programming a Formula Object/References

Programming

Overview of commands

Table 4-203 Formula object TO: Overview of commands

Command	Functionality
_defineFormula	Define a formula
_setFormula	Simultaneous assignment/switchover of formulas to outputs
_enableFormulaObjectIn	Simultaneous activation of all inputs and formula assignment
_disableFormulaObjectIn	Simultaneous deactivation of all inputs
_enableFormula	Simultaneous activation/switching of formulas to outputs
_changeEnableModeOfFormulaObjectIn	Simultaneous activation/deactivation of selected inputs
_changeEnableOfFormula	Simultaneous activation/deactivation of selected formulas
_disableFormula	Simultaneous deactivation of formulas
_setFormulaObjectOutputValue	Selective setting of function values on outputs
_resetFormulaObject	Reset all output values
_resetFormulaObjectError	Reset error
_resetFormulaObjectConfigDataBuffer	Delete collected configuration data
_bufferFormulaObjectCommandId	Store CommandId and command status temporarily
_removeBufferedFormulaObjectCommandId	Delete CommandId
_getStateOfFormulaObjectCommand	Read out command status
_getFormulaObjectErrorNumberState	Read out error number status

Note

For a complete list of all commands and their syntax, the system variables, and error messages, please see the SIMOTION Reference Lists.

Commands

General

The following commands can be executed together in one IPO cycle clock:

- Formula number assignment
- Set output value
- Activate/deactivate

Defining a formula

A formula can be defined with **_defineFormula**.

The command is synchronous.

With **_defineFormula** it is also possible to influence individual components of the motion vector.

See "**Rules for definition of formulas**"

Assigning a formula

Formulas are assigned simultaneously to outputs with **_setFormula** (in one command because several formulas/all formulas are possible in one cycle clock).

Enabling inputs

- All inputs are enabled simultaneously with **_enableFormulaObjectIn ( )**.
  The inputs can be individually specified.

The inputs are active following controller power-up, if they are interconnected. (The status can be scanned using a system variable.)

If the inputs are not interconnected, an error is output. (The alarm is not generated until _enable...In.)

Enabling/disabling specific inputs

- Specific inputs are enabled/disabled simultaneously
  with **_changeEnableModeOfFormulaObjectIn ( )**.
  Input vectors can be enabled individually and are connected as a whole (with components s, v, a) when enabled.
  Input values are generated from the interconnected values, the most recent valid interconnected values, or the default values.

Validity of input values or default values

Configuration data element **MotionIn#.behaviorInvalidInterface** can be used to specify which of the following values is used if the interconnection values are enabled but invalid:

- Most recent valid value (**LAST_VALID_INTERFACE_VALUE**) or
- Replacement values (**DEFAULT_VALUE**)

(Following system startup, the most recent valid value is 0).

If the interconnection interfaces on the input side are disabled, the default values are applied.

Disabling inputs

- All inputs are disabled simultaneously with **_disableFormulaObjectIn ( )**.
  The inputs can be individually specified.

Enabling all formulas

_enableFormula ( ) enables the formula object.

It is possible to enable each input separately. It is, for example, possible to enable only the velocity component of a vector.

Enabling/disabling specific formulas

- Specific formulas are enabled/disabled simultaneously with **_changeEnableOfFormula ( )**.

Disabling all formulas

- Formulas are simultaneously disabled with **_disableFormula ( )**.
  The formulas can be specified individually using output assignment.

Setting function values on outputs

- **_setFormulaObjectOutputValue** can be used to assign function values to each possible output of the formula object.

  Note

  Only one disabled/inactive output can be set.

Example:

_setFormulaObjectOutputValue(MO1sValue=...) assigns a value to the s-component (position) of the motion interface 1 (MO: Motion Out)

Resetting outputs

- **_resetFormulaObject** sets all output values to zero and resets all formulas (formula number 0).

Rules for definition of formulas

The following rules apply when assigning formulas:

- The individual outputs are assigned to formulas in the formula object using formula numbers.
- The formulas can be assigned/changed over for a disabled output and an enabled output respectively.
- Formulas can be assigned simultaneously to multiple outputs, i.e. changeover of formulas in a command.
- It may be that new outputs will have to be enabled at the same time that formulas are changed over to outputs that have already been enabled.
  In other words, formulas can be changed over and inputs/outputs enabled on the basis of IPO-synchronous task commands that all take effect during the same IPO cycle.
- The formula in an output can be changed over online without having to reset or restart the entire formula object.
- Formulas that are not enabled can be written to.
- If a formula is not explicitly assigned to an output, the null formula is active (output = 0).

Formula numbers

The individual formulas in the Formula Object TO are identified using a formula number.

The formula numbers have values of 1 to n.

The formula numbers are assigned to the outputs. The formula calculation is enabled when the output is enabled.

The formula assigned to an output can be changed over during runtime by reassigning a formula number to an output.

The same formula number can be assigned to more than one output at a time and can thus be active a multiple number of times.

After controller power-up, the "null formula" (function value of 0) is assigned to the individual formula numbers.

Valid formula elements

The following elements can be used to define a formula:

- Literals (numbers and letters)
- Identifiers (variables)
- Basic operators
- Provided functions
- Nesting of expressions

Character set

The following characters are permissible:

- Numbers `0-9`
- Letters `a-z, A-Z`
- Special characters `+, -, *, /, (, ), _, #`

No distinction is made between upper and lower case letters.

Number types

Only integer and floating point number types are permitted:

- Integers (DINT) in decimal, octal, and hexadecimal notation
- Floating-point numbers (LREAL) in rational and exponential notation

Vector components are mapped onto these.

Identifier

Only defined symbols for input and output components are permitted as variables, e.g.:

- LI1 (LREALIn1)
- DI1 (DINTIn1)

Vectors are specified component by component because they refer to the components of the motion vector (s, v, a), e.g.:

- MI1.s (MotionIn1, position)
- MI1.v (MotionIn1, velocity)
- MI1.a (MotionIn1, acceleration)

Nesting of expressions

Expressions can be nested by using parentheses `( )`.

Implicit type conversions

- DINT to LREAL:
  In all (completed) expressions in which the highest address is a LREAL value
- DINT to LREAL:
  For function parameters of type LREAL
- DINT to LREAL:
  If the result of the instruction is a LREAL value

Explicit type conversions

- LREAL to DINT:
  Using LD( )

Data type ANY {DINT, LREAL} is implicitly specified for functions. Within a function, ANY is resolved into type DINT or LREAL. This conforms with the parameter assignment and/or the possible implicit type conversions.

Basic operators

In the case of basic operators, only those operators that require integer and/or floating point operand types are used.

Boolean operations/Boolean expressions are defined as functions in order to circumvent explicit type conversions.

- Unary operators:
  -(Negation) [DINT/LREAL]
- Binary operators:
  +, -, *, /, [DINT, LREAL], MOD [DINT]

Note

In contrast to the addition object, separate inversion of an input using configuration data is not possible in the formula object.

Available functions in formulas (formula operators)

Table 4-204 Formula object TO: Formula operators

Function	Description
Logic operators	
Logical **_AND** **AND**(DINT, DINT):DINT	Function for Boolean AND Result is 0 if either of the two parameters is 0; otherwise, it is 1
Logical **_OR** **OR**(DINT, DINT):DINT	Function for Boolean OR Result is 0 if both parameters are 0; otherwise, it is 1
Logical **_XOR** **XOR**(DINT, DINT):DINT	Function for Boolean XOR Result is 1 if one parameter is 0 and the other parameter is not 0; otherwise, result is 0
Logical **_NOT** **NOT**(ANY):DINT	Function for Boolean NOT Result is 1 if parameter is 0; otherwise, it is 0
Equal **EQ** **EQUAL**(DINT, DINT):DINT	Parameters are equal Result is 1 if parameters are identical; otherwise, it is 0
Not equal **LE** **LESS**(DINT, DINT):DINT	Parameters are not equal Result is 1 if parameter 1 is less than parameter 2; otherwise, result is 0
Arithmetic	

Function	Description
SIN(LREAL):LREAL **COS**(LREAL):LREAL **TAN**(LREAL):LREAL **ASIN**(LREAL):LREAL **ACOS**(LREAL):LREAL **ATAN**(LREAL):LREAL **LN**(LREAL):LREAL **LOG**(LREAL):LREAL TRUNC(LREAL):LREAL EXPT(LREAL, DINT):LREAL **ABS**(LREAL):LREAL **SQRT**(LREAL):LREAL	Arithmetic functions Result is the value of the mathematic operation In the event of an error, processing of the expression is aborted and the value is not updated.
Miscellaneous	
Maximum **MAX**(ANY, ANY):ANY	Maximum generation Result is the maximum of the two parameters
Minimum **MIN**(ANY, ANY):ANY	Minimum generation Result is the minimum of the two parameters
Limiting **LIMIT**(ANY, ANY, ANY):ANY	Limit • Parameter 1: Lower limiting value • Parameter 2: Value to be limited • Parameter 3: Upper limiting value Result is the limitation of parameter 2
Selection **SEL**(DINT, ANY, ANY):ANY	Selection function Parameter 1 <> 0 → Result = Parameter 2 Parameter 1 = 0 → Result = Parameter 3

System variables

The input and output values of an formula object can be read out via system variables.

Table 4-205 Formula object TO: System variables

System variable			Type	Description	Remark
motionIn1			StructFormula ObjectMotionIn	1. input vector	At least one input must be interconnected.
	value		StructMotionVector	Components of 1st input vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	state		EnumActiveInactive	Status	
	lastValidInterface value		StructMotionVector	Last valid values	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	validity		EnumInterface ValueDefaultValue	Validity	

System variable		Type	Description	Remark
motionIn1Default		StructMotionVector	Default values for 1st input vector	
motionIn2		StructFormula ObjectMotionIn	2. input vector	
motionIn2Default		StructMotionVector	Default values for 2nd input vector	
motionIn3		StructFormula ObjectMotionIn	3. input vector	
motionIn3Default		StructMotionVector	Default values for 3rd input vector	
LREALIn1		StructFormula ObjectLREALIn	1. input scalar LREAL	
	value	LREAL	Value	
	state	EnumActiveInactive	Status	
	lastValidInterface Value	LREAL	Last valid values	
	validity	EnumInterface ValueDefaultValue	Validity	
LREALIn1Default		StructFormula ObjectLREALIn	Default value for 1st input scalar LREAL	
LREALIn2		StructFormula ObjectLREALIn	2. input scalar LREAL	
LREALIn2Default		StructFormula ObjectLREALIn	Default value for 2nd input scalar LREAL	
LREALIn3		StructFormula ObjectLREALIn	3. input scalar LREAL	
LREALIn3Default		StructFormula ObjectLREALIn	Default value for 3rd input scalar LREAL	
LREALIn4		StructFormula ObjectLREALIn	4. input scalar LREAL	
LREALIn4Default		StructFormula ObjectLREALIn	Default value for 4th input scalar LREAL	
DINTIn1		StructFormula ObjectDINTIn	1. input scalar DINT	
	value	DINT	Value	
	state	EnumActiveInactive	Status	
	lastValidInterface Value	DINT	Last valid values	
	validity	EnumInterface ValueDefaultValue	Validity	
DINTIn1Default		StructFormula ObjectDINTIn	Default value for 1st input scalar DINT	
DINTIn2		StructFormula ObjectDINTIn	2. input scalar DINT	
DINTIn2Default		StructFormula ObjectDINTIn	Default value for 2nd input scalar DINT	
DINTIn3		StructFormula ObjectDINTIn	3. input scalar DINT	

System variable			Type	Description	Remark
DINTIn3Default			StructFormula ObjectDINTIn	Default value for 3rd input scalar DINT	
DINTIn4			StructFormula ObjectDINTIn	4. input scalar DINT	
DINTIn4Default			StructFormula ObjectDINTIn	Default value for 4th input scalar DINT	
motionOut1			StructFormula ObjectMotionOut	1. Output vector	
	value		StructMotionVector	Components of the output vector	
		s	LREAL	Position	
		v	LREAL	Velocity	
		a	LREAL	Acceleration	
	error		EnumYesNo	Error status	
	sFormula		UINT	Formula value position	
	sFormulaEnableState		EnumActiveInactive	Formula status position	
	vFormula		UINT	Formula value velocity	
	vFormulaEnableState		EnumActiveInactive	Formula status velocity	
	aFormula		UINT	Formula value acceleration	
	aFormulaEnableState		EnumActiveInactive	Formula status acceleration	
motionOut2			StructFormula ObjectMotionOut	2. Output vector	
motionOut3			StructFormula ObjectMotionOut	3. Output vector	
motionOut1Default			StructMotionVector	Default values for 1st output vector	Deactivate with function, see **_disableFormula**
motionOut2Default			StructMotionVector	Default values for 2nd output vector	
motionOut3Default			StructMotionVector	Default values for 3rd output vector	
LREALOut1			StructFormula ObjectLREALOut	1. output scalar LREAL	
	value		LREAL	Value	
	error		EnumYesNo	Error status	
	Formula		UINT	Formula value	
	FormulaEnableState		UINT	Formula status	
LREALOut2			StructFormula ObjectLREALOut	2. output scalar LREAL	
LREALOut3			StructFormula ObjectLREALOut	3. output scalar LREAL	
LREALOut4			StructFormula ObjectLREALOut	4. output scalar LREAL	

System variable		Type	Description	Remark
LREALOut1Default		LREAL	Default values for 1st output scalar LREAL	Deactivate with function, see **_disableFormula**
LREALOut2Default		LREAL	Default values for 2nd output scalar LREAL	
LREALOut3Default		LREAL	Default values for 3rd output scalar LREAL	
LREALOut4Default		LREAL	Default values for 4th output scalar LREAL	
DINTOut1		StructFormula ObjectDINTOut	1. output scalar DINT	
	value	DINT	Value	
	error	EnumYesNo	Error status	
	Formula	UINT	Formula value	
	FormulaEnableState	UINT	Formula status	
DINTOut2		StructFormula ObjectDINTOut	2. output scalar DINT	
DINTOut3		StructFormula ObjectDINTOut	3. output scalar DINT	
DINTOut4		StructFormula ObjectDINTOut	4. output scalar DINT	
DINTOut1Default		DINT	Default values for 1st output scalar DINT	Deactivate with function, see **_disableFormula**
DINTOut2Default		DINT	Default values for 2nd output scalar DINT	
DINTOut3Default		DINT	Default values for 3rd output scalar DINT	Deactivate with function, see **_disableFormula**
DINTOut4Default		DINT	Default values for 4th output scalar DINT	

Local alarm response

Responses in the event of an error

The **error status** is displayed in the Formula Object TO if:

- An error is present in the overall object
- An individual error exists

Individual errors are also indicated at the output.

System variable **error** indicates the overall status. This variable is set if:

- The overall object is faulty
- Overall processing is halted
- An individual error exists

Intermediate results are not displayed.

Local reactions

If an error occurs, the following local responses are possible:

- No response (**NONE**)
- Stop processing of the specific formula (**STOP_SPECIFIC_FORMULA**)
- Stop all processing (**STOP_ALL_FORMULA**)
- Stop TO processing (**DISABLE**)

The local reactions can be set under **TechnologicalFaultTask** in the alarm configuration.

Response, e.g., for division by zero

The system performs a check to determine whether division by zero is present.

If yes:

- A technological alarm is triggered
- The output is disabled/frozen
- The formula must be restarted/enabled

Other technological alarms

The standard technology object alarms, e.g., interconnection error, illegal parameter, are output.

Menus

Formula object - menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the formula object selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration	This function opens the configuration for the formula object selected in the project navigator. In this window, specify the configuration of the inputs and outputs.

Function	Meaning/Note
Factory setting (default)	This function opens the defaults for the formula object selected in the project navigator. Specify the input default values for calling the formula object in this window (**_enableFormulaObjectIn** or **_disableFormulaObjectIn**).
Interconnections	This function opens the interconnections for the formula object selected in the project navigator. In this window, interconnect the inputs of the formula object, e.g. to axes.
Expert	This function opens the submenu for the expert settings.
Expert list	This function opens the expert list for the formula object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list

Formula object - context menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Open configuration	This function opens the configuration for the formula object selected in the project navigator. In this window, specify the configuration of the inputs and outputs.
Factory setting (default)	This function opens the defaults for the formula object selected in the project navigator. Specify the input default values for calling the formula object in this window (**_enableFormulaObjectIn** or **_disableFormulaObjectIn**).
Interconnections	This function opens the interconnections for the formula object selected in the project navigator. In this window, interconnect the inputs of the formula object, e.g. to axes.
Expert	This function opens the submenu for the expert settings.
Expert list	This function opens the expert list for the formula object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list

Function		Meaning/Note
	Import object	Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.
	Save project and export object	Use Save project and export object to open a window for an XML export. You can define the parameters for the XML export in this window.

4.4.4.4 Example

A real axis 1 is to be linked to a virtual axis 2 via a formula object.

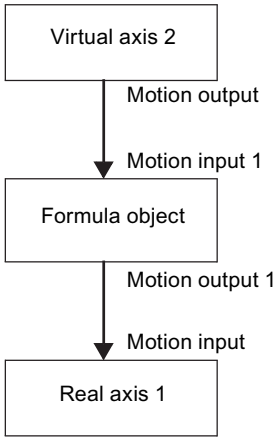


Figure 4-297 Linking two axes via a formula object

Procedure

1. Create all objects.
2. Interconnect motion input 1 of the formula object with the motion output of virtual axis 2.

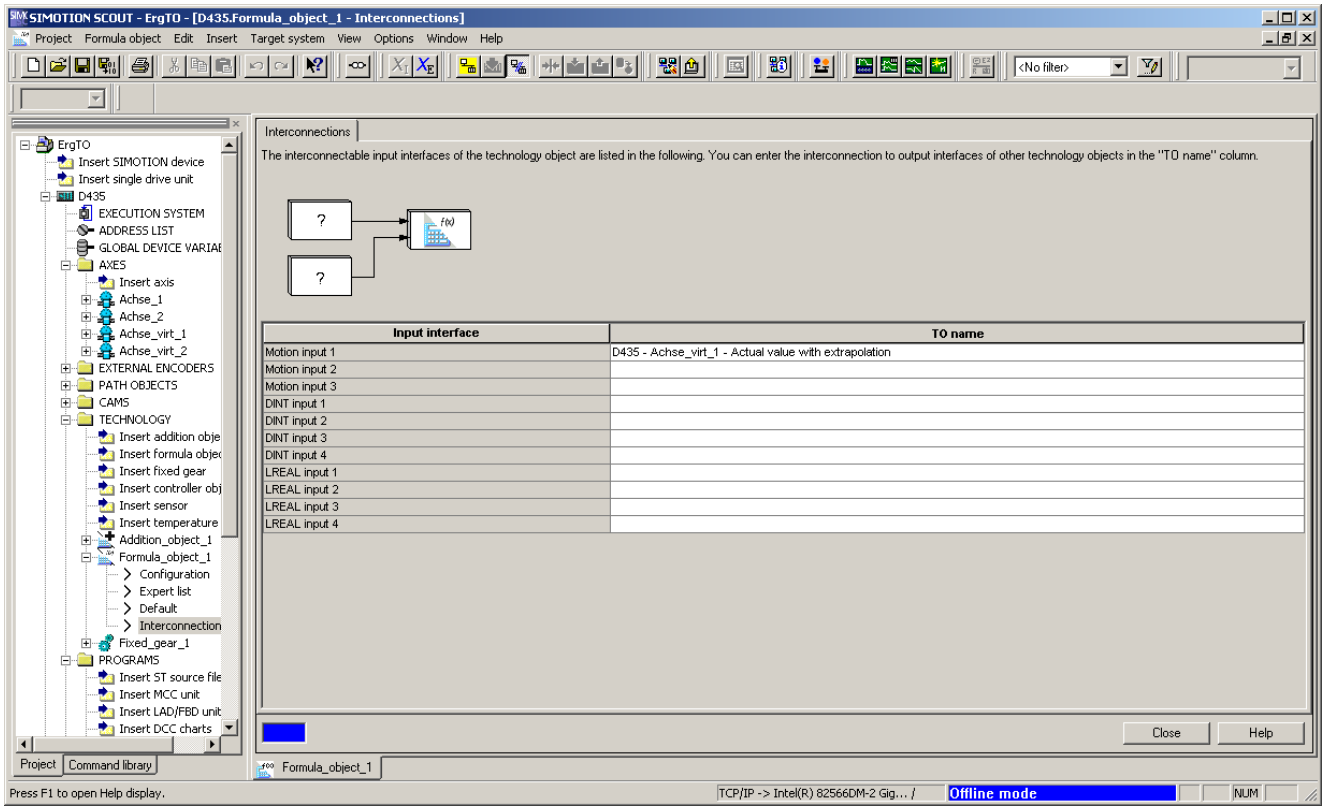


Figure 4-298 Example of interconnection: Formula_object_1 -> Axis_virt_2

3. Interconnect the motion input of axis 1 with the motion output of the formula object.

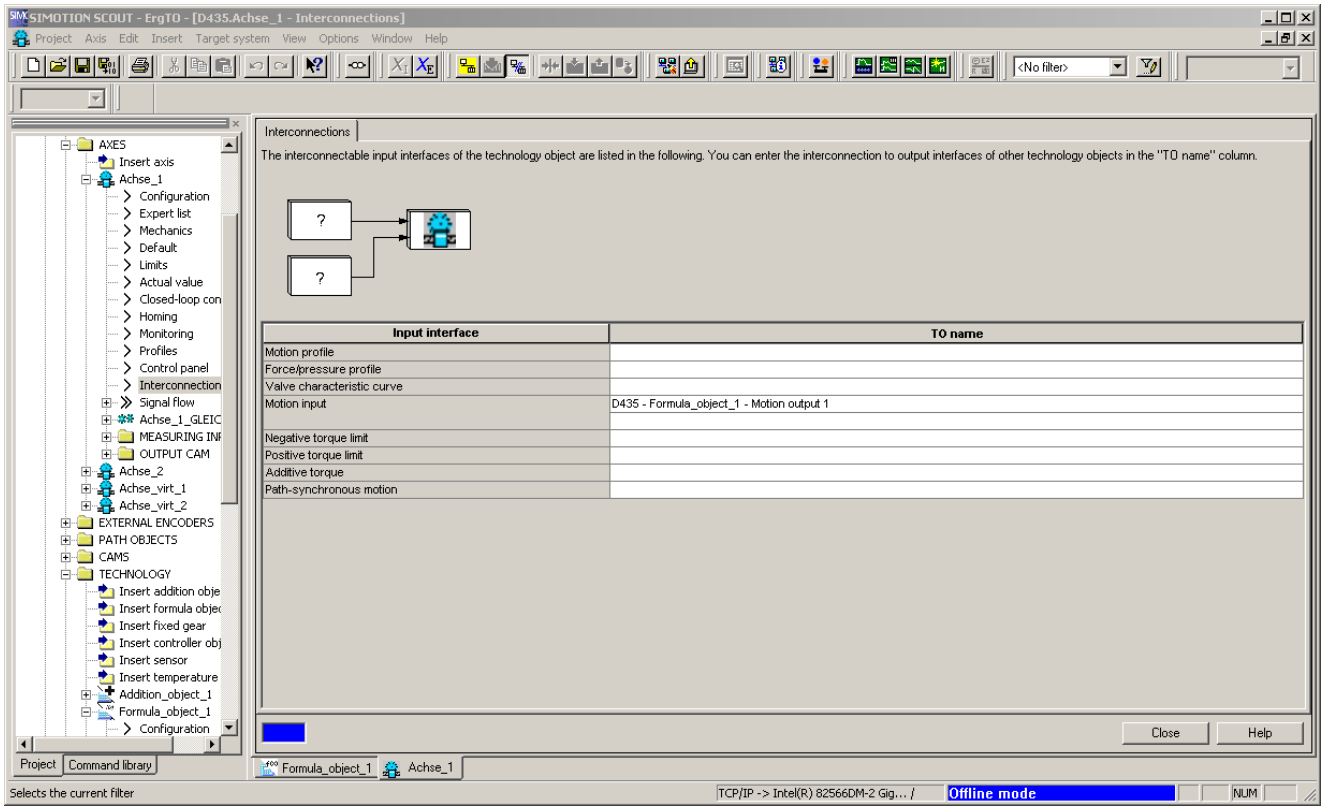


Figure 4-299 Example of interconnection: Axis_1 -> Formula_object_1

4. Use the **_defineFormula** command to influence the components of the motion vector.

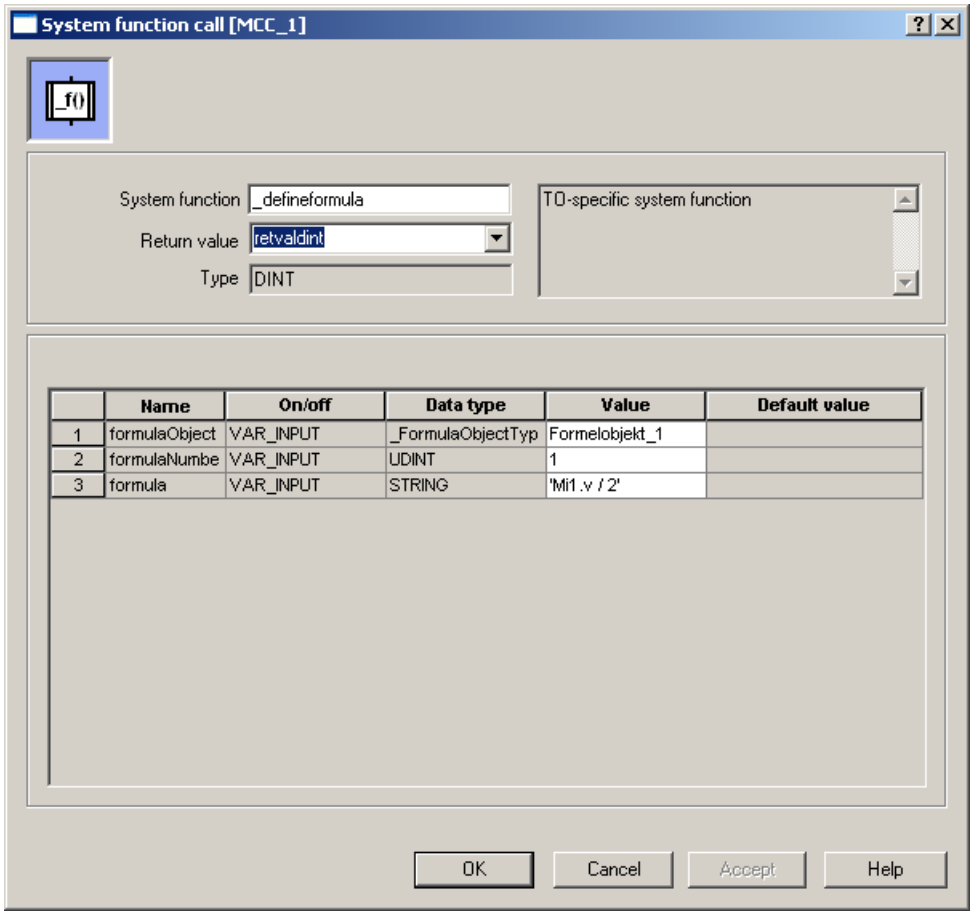
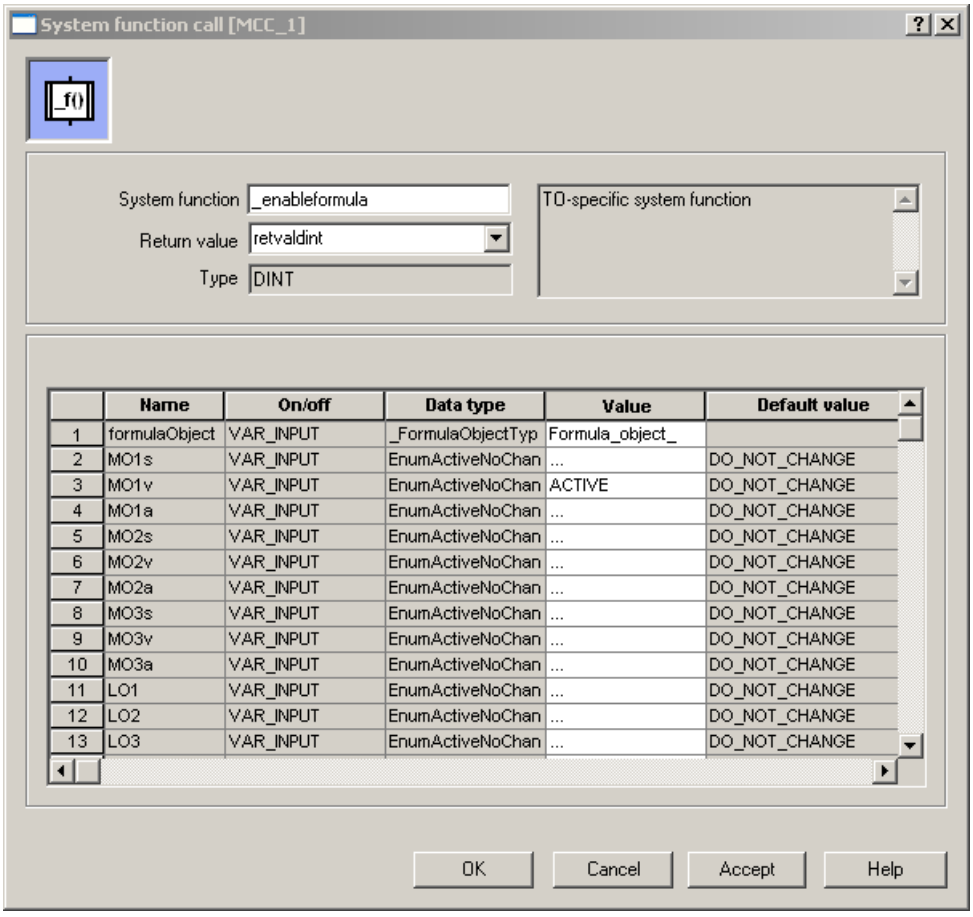


Figure 4-300 Example of calling **_defineFormula** to change a component

The abbreviation "Mi1.v" stands for MotionIn1.value.v (velocity).

5. The **_enableFormula** command activates the formula object; it is also possible to activate each input separately.
   Only the velocity component of the vector is to be activated here.



	Name	On/off	Data type	Value	Default value
1	formulaObject	VAR_INPUT	_FormulaObjectTyp	Formula_object_	
2	MO1s	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
3	MO1v	VAR_INPUT	EnumActiveNoChan	ACTIVE	DO_NOT_CHANGE
4	MO1a	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
5	MO2s	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
6	MO2v	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
7	MO2a	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
8	MO3s	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
9	MO3v	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
10	MO3a	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
11	LO1	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
12	LO2	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE
13	LO3	VAR_INPUT	EnumActiveNoChan	...	DO_NOT_CHANGE

Figure 4-301 Example of calling **_enableFormula** with a component

6. The formula is assigned to the outputs with the **_setFormula** command.

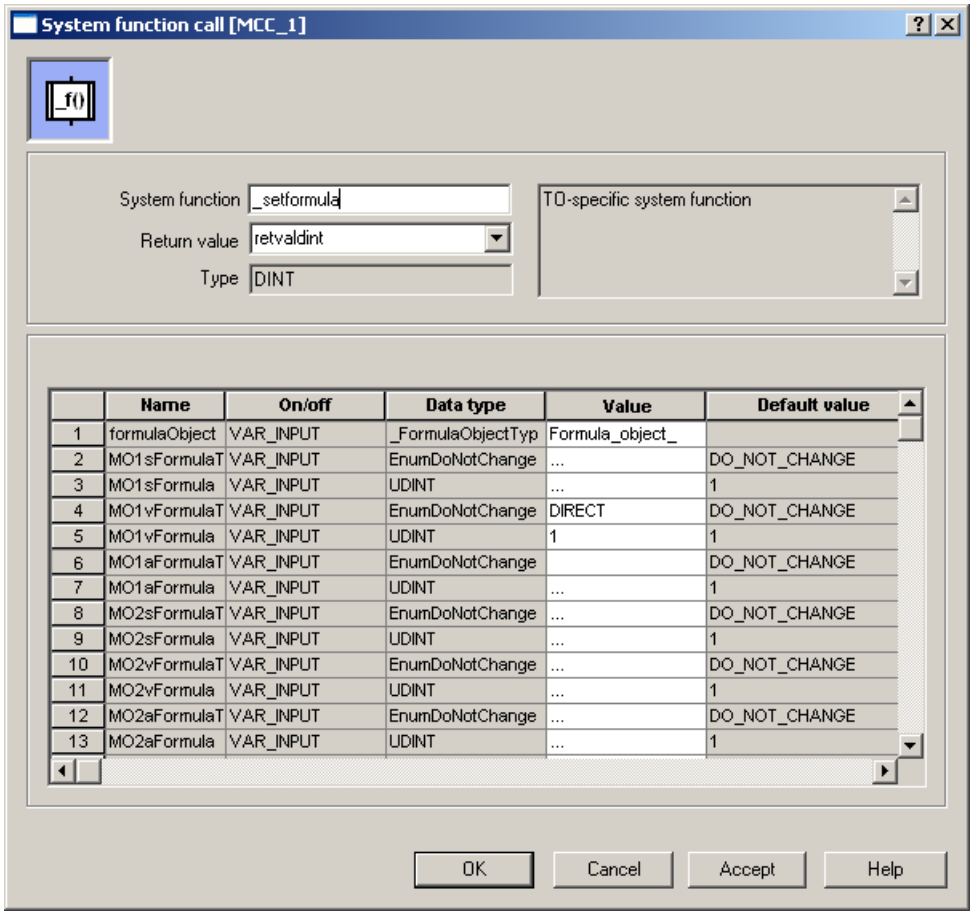


Figure 4-302 Example of calling **_setFormula** with a component

7. Use the **_runVelocityBasedMotionIn** command to activate the motion input of axis 2.

4.4.5 Part IV - Sensor

4.4.5.1 Overview of Sensor

Function overview

The **sensor** TO can be used to record scalar measured values.

The sensor TO reads out a value from the I/O and supplies an actual value as an output signal in standardized formats.

Application

A **sensor TO** can be used to acquire and prepare scalar measured values.

Interfaces

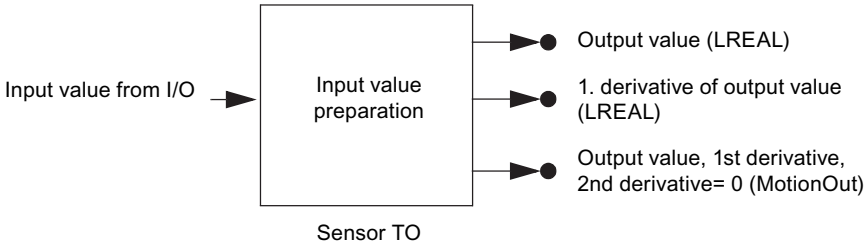


Figure 4-303 Object model for sensor TO

A sensor TO has one input for reading an I/O value and three outputs.

The first output provides the output value, the second output provides its first derivative, and the third output provides the output value as well as its first and second derivatives in the form of a motion vector.

The acceleration component in the motion vector is always 0 (see image).

Interconnection

The input value is obtained from the I/O.

The interfaces are optional and incorporated during configuration.

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Preprocessing

The input value is preprocessed according to the configuration.

The following preprocessing steps can be performed:

- Format adaptations for a wide range of input formats
- Signal monitoring
- Linear signal normalization
- Signal filtering
- Signal differentiation

Input value

The input value and the input value preparation can be activated or deactivated (**_enableSensor** or **_disableSensor**).

When the input value ifs deactivated, it can be specified whether the value is retained or replaced by a zero value, default value, or direct value.

The sensor TO enables up to 32-bit values to be read in from the I/O interface (adjustable). Functions for adaptation to different input formats are possible.

Output value

At the output, it is possible to select whether the prepared input value, the last value, or the default value is output.

The output value can be read.

The sensor TO is active after power-up.

The output value can be interconnected to a **motion vector**:

- The value to the position component (s)
- The derived value to the velocity component (v)
- The value of the acceleration component (a) is zero.

The output values are zero after the TO has powered up.

Units

Units are taken into account (SI units).

The following units are available as of V3.2:

- Force: [N] (newton)
- Distance: [mm] (millimeter)
- Temperature: [°C]
- Angular degrees: [°]
- Voltage: [V] (volt)
- Current: [A] (ampere)
- Unitless [---, %] (normalized variable)

Units can be set on the sensor TO, they apply at the input and output side.

4.4.5.2 Fundamentals of sensors

Function in principle

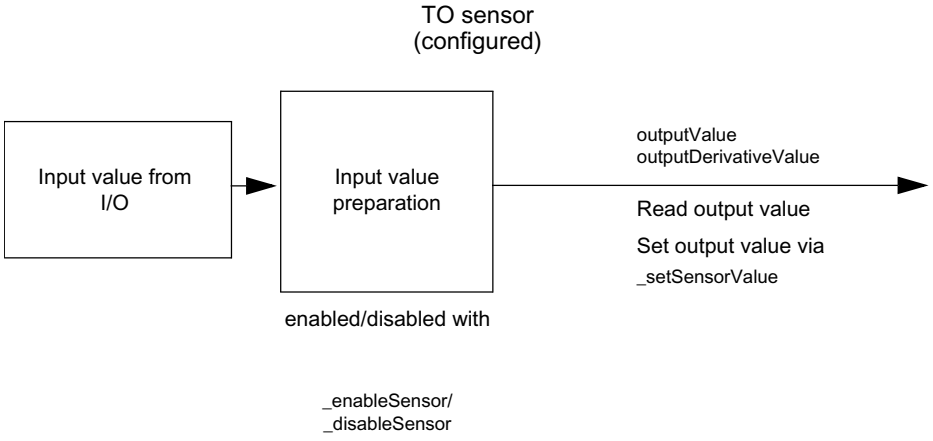


Figure 4-304 Function of sensor TO

The input value is read in from the I/O by the **TO sensor** and preprocessed according to configuration.

The output value is made available directly and as a derived value.

Measured value

The sensor TO enables values to be read in from the I/O interface (up to 32 bits, adjustable).

The **sensor.analogSensorDriverInfo** configuration data element can be used to adapt the input to different input formats.

logAddress	Logical address of the communication module used
resolution	Number of relevant bits (INT32, resolution ADU)
format	Format of the actual value
minValue	Minimum digitized measured value
maxValue	Maximum digitized measured value
errorToleranceTime	Tolerance time for which an error has to be present before an alarm is triggered

Function in detail

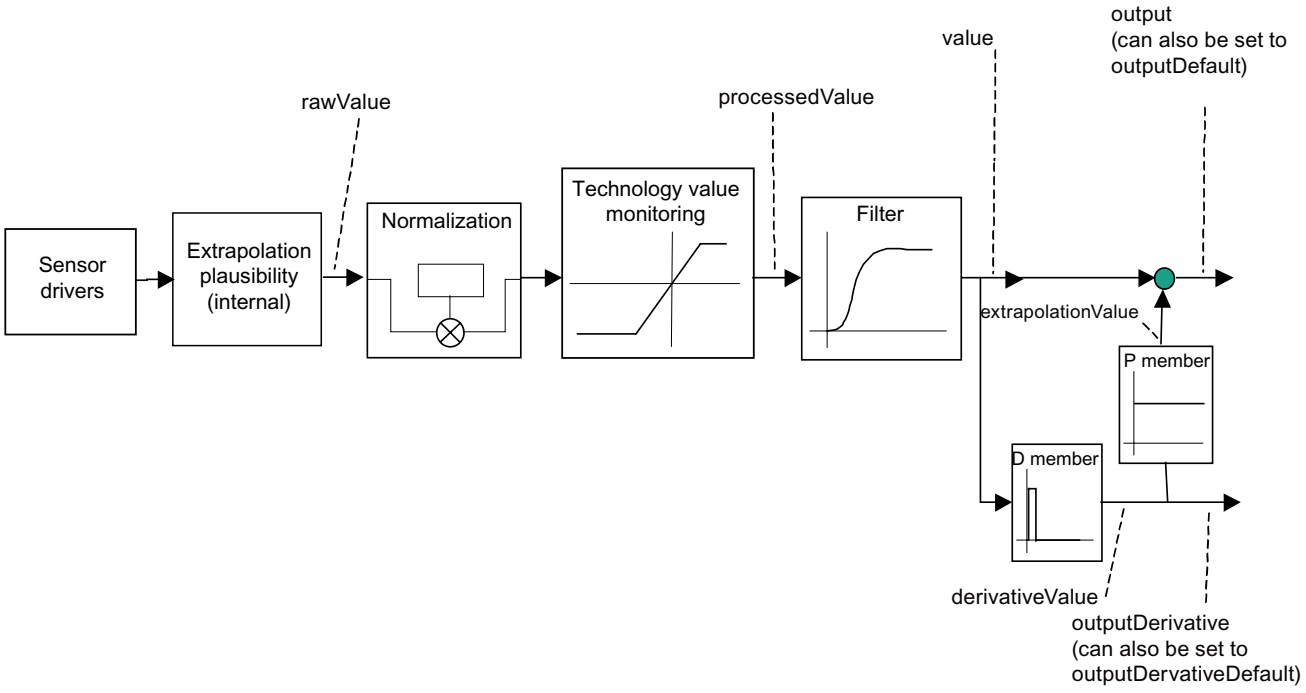


Figure 4-305 Description of functions of sensor TO

Normalization of the input variable

The signals are normalized linearly (configuration data element **sensor.conversiondata._type** =LINEAR).

Configuration data elements **sensor.conversiondata.factor** and **sensor.conversiondata.offset** can be used to specify (linear) normalization.

⚠ WARNING

Weighting factor for analog sensors

The scaling or conversion in the value of the configuration data **ConversionData.factor** for the position of the analog sensor has changed in SIMOTION V4.4.

As of V4.4, the following applies:

If the unit system is configured for inches, then the values are entered directly in inches and are therefore no longer converted from mm to inches. This can result in a higher velocity being detected (by a factor of 25) if the axis is working in inches.

The following applies to SIMOTION < V4.4:

Values are entered in mm and then converted to inches (1 inch corresponds to around 25 mm). Please check the values entered for **ConversionData.factor** in older projects and correct them if necessary.

Raw value monitoring

A permissible raw value range is specified for monitoring of the raw value output by the module. The raw values may fall outside the range for a defined period of time. All values outside the range are regarded as errors, and the raw values are limited.

The permissible raw value range and the time window can be specified in configuration data item **sensor.AnalogsensorDriverInfo.minValue/maxValue**.

The monitoring status is indicated in the **monitorings.rawvalue** system variable.

Technology value monitoring

The technology value is monitored and limited with respect to the limits specified in the **sensor.range** configuration data element.

The monitoring status is indicated in the **monitorings.value** variable.

Filter

A PT1 function is inserted in the signal path for smoothing the output value.

The filter can be switched on and off with the **sensor.filter** configuration data element.

Differentiation

The derivative of the output value is generated by numerical differentiation.

The filtered value is used for this purpose.

Applying the derived value to the output value

The derived value can be applied to the output value with weighting (P element).

The weighting value is specified as a factor and not as an extrapolation time.

The weighting value setting can be made in the **extrapolation.factor** configuration data element.

4.4.5.3 Configuring sensors

Creating a sensor object

Sensor technology objects are stored across-the-board for a device in the **TECHNOLOGY** folder. They can be interconnected with suitable technology objects of the device.

Proceed as follows

1. To create a new sensor TO, double-click **Insert sensor** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing sensor TO using the clipboard and insert it under another name.

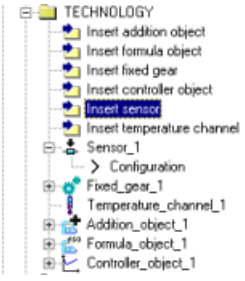


Figure 4-306 Representation of sensors in the project navigator

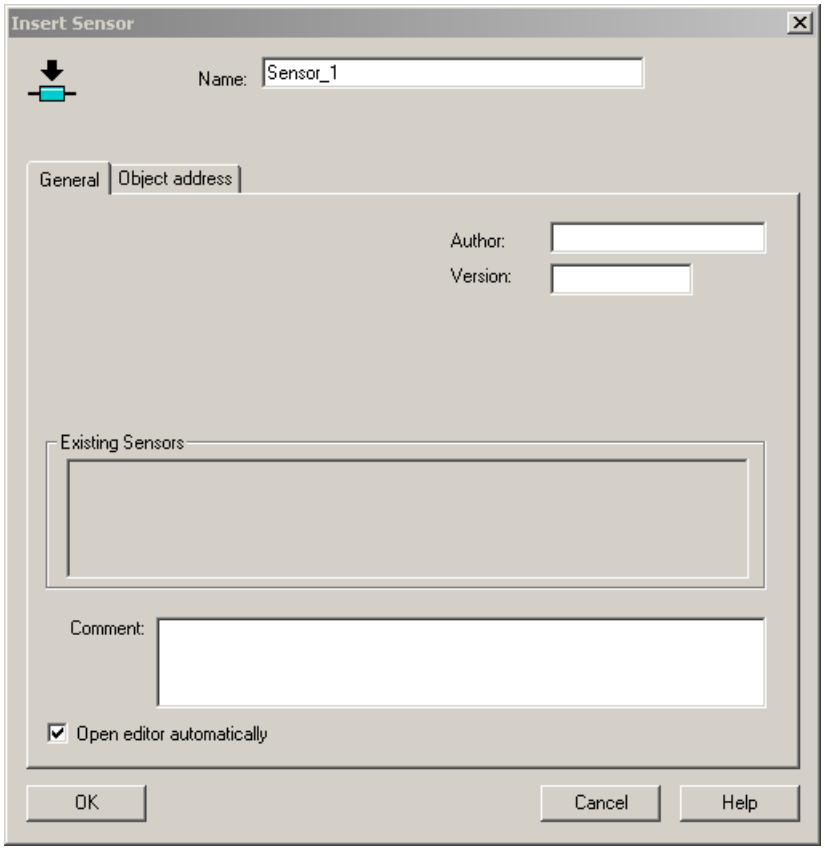


Figure 4-307 Inserting a sensor TO

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new sensor TO will be inserted under **TECHNOLOGY**.

Configuring a sensor

Procedure

In the project navigator, double-click **Configuration** under the object.

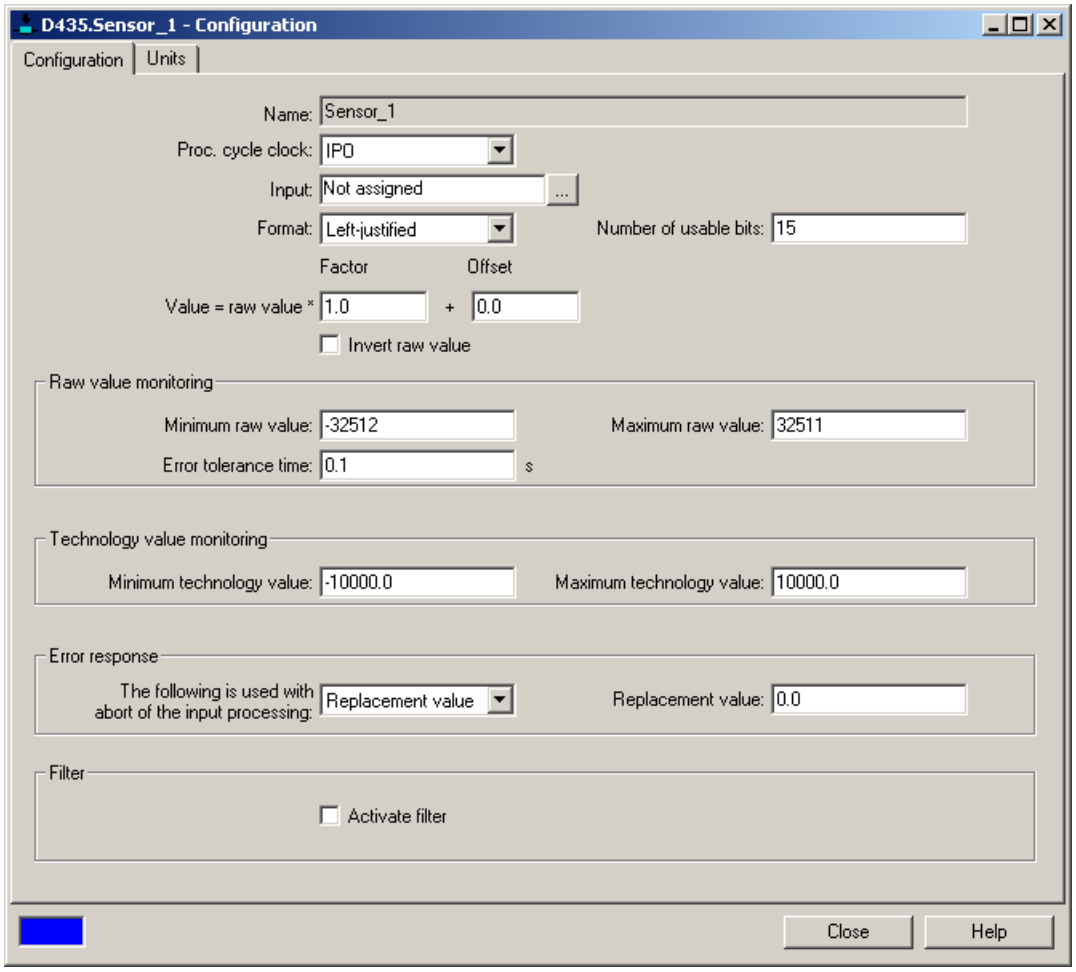


Figure 4-308 Configuring a sensor

Sensor - configuration

Define the properties of the sensor in this window.

You can set the following parameters:

Table 4-206 Sensor TO: Parameters that can be set for configuration purposes

Field/button	Meaning/instruction
Name	Display: name of sensor
Input	The input can be symbolically assigned to the analog signal via the assignment dialog (see the Chapter entitled Symbolic Assignment (from V4.2 onward) in the Motion Control Basic Functions manual) and by using the ... button in the Input field (symbolic assignment is activated by default in projects as of V4.2).
Format	Here, you select the binary format.
Number of usable bits	Here, you enter the number of data bits.
Processing cycle clock	Here, you select the IPO-, IPO_2, Servo, Servo_fast or IPO_servo cycle clock. For the possible setting IPO_fast, see the chapter entitled Second position control cycle clock (Servo_fast) in the Motion Control Basic Functions manual.
Raw value monitoring	Here, you enter the values for raw value monitoring. A permissible raw value range is specified for monitoring of the raw value output by the module. The raw values may fall outside the range for a defined period of time. All values outside the range are regarded as errors, and the raw values are limited.
Technology value monitoring	Here, you enter the values for **Technology Value Monitoring**. The technology value is monitored and limited with respect to the limits specified.
Error response	Here, you select the response in the event of an error.
Filter	Here, you switch the filter on or off. A PT1 function is inserted in the signal path for smoothing the output value. The derived value is generated from the filtered value.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 2030)

Overview of commands (Page 2039)

4.4.5.4 Programming a Sensor/References

Programming

Overview of commands

Table 4-207 Sensor TO: Overview of commands

Command	Functionality
_enableSensor	Enabling/disabling input processing in a sensor TO
_disableSensor	
_resetSensor	Reset sensor
_resetSensorError	Reset error
_resetSensorConfigDataBuffer	Delete collected configuration data
_bufferSensorCommandId	Store CommandId and command status temporarily
_removeBufferedSensorCommandId	CommandId and delete
_getStateOfSensorCommand	Read out command status
_getSensorErrorNumberState	Read out error number status

Note

For a complete list of all commands and their syntax, the system variables, and error messages, please see the SIMOTION Reference Lists.

Commands

Enabling/disabling the sensor TO

The input value of the sensor TO is active after power-up, and thus does not have to be enabled specifically.

The option exists to disconnect the I/O value and specify a default value.

Activating/deactivating input processing in a sensor TO

- **_enableSensor**:
  Enables the input processing
- **_disableSensor**:
  Disables the input processing
  A value setting is not cyclically overwritten.
  The **valueBehaviorMode** parameter can be used to set one of the following functions:
  - Retain the last value (LAST_VALUE)
  - Use the default value (DEFAULT_VALUE)
  - Output the zero value (ZERO_VALUE)

System variables

The input and output signals of the sensor TO are represented using system variables.

Table 4-208 Sensor TO: System variables

System variable	Type	Description	Remark
rawValue	LREAL	Value before linearization	Can only be read
processedValue	LREAL	Value after linearization and limiting	Can only be read
value	LREAL	Technological output value (value after filtering)	Can only be read
derivedValue	LREAL	Differential technological output value	Can only be read
extrapolationValue	LREAL	Extrapolation value	Can only be read
outputDefault	LREAL	Specification for output value	Read/write
outputDerivativeDefault	LREAL	Specification for derivative of output value	Read/write
output	LREAL	Output value	Can only be read
outputDerivative	LREAL	Derivative of output value	Can only be read
motionOut	StructMotionVector	Motion vector	Can only be read
monitorings	StructSensorMonitorings	Monitoring states of sensor TO:	Can only be read
rawValue	EnumLimitExceededOk	Raw value monitoring	
value	EnumLimitExceededOk	Technology value monitoring	
control	EnumActiveInactive	Operational status	Read-only
error	EnumYesNo	Technological alarm on the sensor TO	Read-only
errorReaction	EnumSensorErrorReaction	Active reaction to technological alarm	Read-only
activationModeChanged ConfigData	EnumToActivationModeSet ConfigData	Activation of modified configuration data	Read/write
restartActivation	EnumToRestartActivation	Perform a TO restart	Read/write

Local alarm response

Technological alarms

The standard technology object alarms, e.g. interconnection error or illegal parameter, are output.

Local reactions

If an error occurs, the following local responses are possible:

- No response (**NONE**)
- Stop command decoding (**DECODE_STOP**)
- Cancel input processing and set defined value (**CONFIGURED_OUTPUT_VALUE)**
  You can use configuration data element **ValueOut.outputValueErrorBehaviorMode** to specify whether:
  - The last value is retained (LAST_VALUE)
  - The zero value (ZERO_VALUE) is applied
  - The default value is applied (DEFAULT_VALUE)

  (default: LAST_VALUE)

Menus

Sensor - menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the sensor selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration	This function opens the configuration for the sensor selected in the project navigator. Define the properties of the sensor in this window.
Expert	This function opens the submenu for the expert settings.

Function		Meaning/Note
	Expert list	This function opens the expert list for the sensor selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.

Sensor - context menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function		Meaning/Note
Open configuration		This function opens the configuration for the sensor selected in the project navigator. Define the properties of the sensor in this window.
Expert		This function opens the submenu for the expert settings.
	Expert list	This function opens the expert list for the sensor selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.
	Import object	Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.
	Save project and export object	Use Save project and export object to open a window for an XML export. You can define the parameters for the XML export in this window.

4.4.6 Part V - Controller object

4.4.6.1 Overview of Controller Object

Function overview

The **Controller object** technology object enables you to prepare and control scalar variables.

Application

A controller object can be used as:

- Universal PIDT1 controller for scalar controlled variables,
  can also be used as a PI or P controller

Interfaces

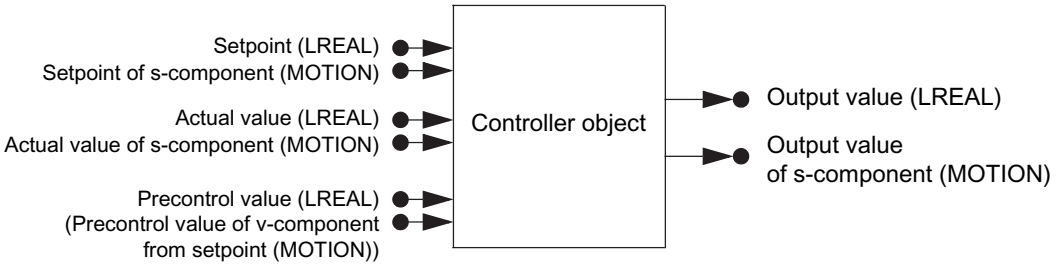


Figure 4-309 Object model for controller object

Each controller object has a setpoint input, actual value input, and precontrol value input.

The inputs are each configurable as a scalar input or as an input for a component of a motion vector.

The output provides the output value as a scalar or as a component of a motion vector.

Interconnection

The interfaces are optional and incorporated during configuration.

The input vectors can be interconnected *once only* ('single point'); the output vector can be interconnected any number of times ('multi-point').

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

Activation/Deactivation

All inputs and the controller object can be activated and deactivated independently of one another.

Output

An activated controller object writes the controller value to the output, provided another behavior is not specified for an error occurrence.

A valid controller output value cannot be overwritten on the output side in the case of an active controller object.

The output value is readable and can be written via **_disableControllerObject**.

Units

Units are taken into account (SI units).

- The following units are available as of V3.2:
  - Force: [N] (newton)
  - Torque: [N] (newton meter)
  - Distance: [m] (meter)
  - Velocity: [m/s] (meters per second)
  - Temperature: [°C]
  - Angular degrees: [°]
  - Unitless [---] (normalized variable)
- The following units are used for controller variables:
  - K_R: [manipulated variable/controlled variable]
  - T_n, T_v: [s] (seconds)

4.4.6.2 Fundamentals of Controller Object

Description of function

Universal PIDT1 controller for scalar controlled variables

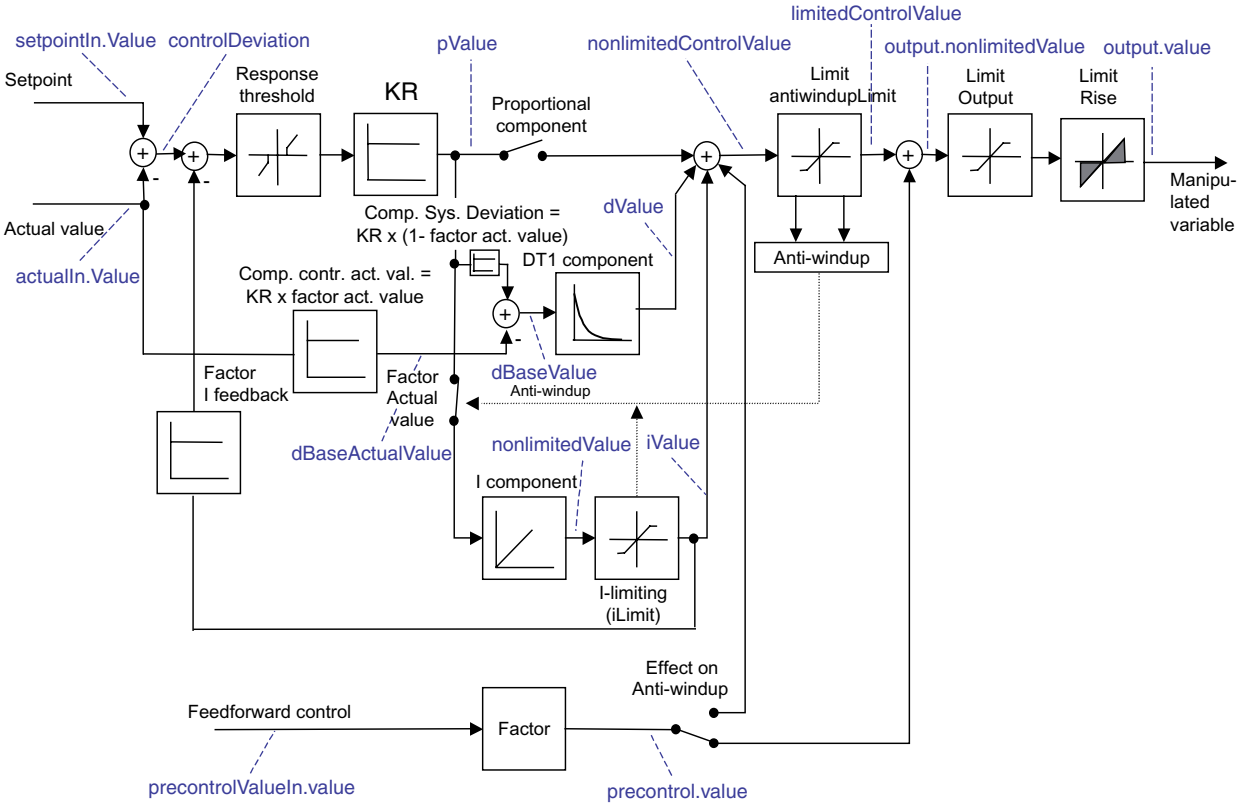


Figure 4-310 Controller object as PIDT1 controller

The controller is designed for systems which can be controlled by the PID algorithm. The control response can be improved by the precontrol function.

Feedforward control is based on a factor (the **precontrol.factor** configuration data item) which is proportional to the actuating signal.

All controller components can be connected and disconnected individually and independently of one another. As a result, the controller can also be employed as a P, PI, PD, and I controller.

In each case, the D and I components are disconnected by setting the associated T_v or T_n controller parameters to zero.

As the proportional gain is upcircuit of all components, the P component is connected via a supplementary parameter (`pmode`).

The upper and lower limits of the actuating signal and the integrator are adjustable. When the selected limits are reached, the signal and integrator are limited and displayed.

Configuration data element **precontrol.addupMode** can be used to indicate whether the feedforward control occurs before or after the anti-windup limiting (default: after).

The response threshold value for control deviation can be specified.

The I component can be fed back with a weighting factor.

It is possible to select whether the D component is generated from the system deviation or directly from the actual value, i.e. the D component from the system deviation and the actual value can be weighted to each other.

The D component is calculated by a differentiator with smoothing.

The I value can be limited.

The I value can be preassigned.

The integrator can be stopped.

The integrator can be stopped using anti-windup, i.e. if the integrator or actuating signal limits are active (configurable), an increment of the integrator is prevented.

P component, I component, and D component are added and limited according to the setting.

The output signal can be limited in value and rise (configuration data elements **output.Limit** and **outputDerivativeLimiting**).

Active limiting operations are indicated.

Limitation and increase limitation of the output value are active:

- For an active controller
- If the output value is set (e.g. with _disableControllerObject)

4.4.6.3 Configuring a Controller Object

Creating a controller object

Controller objects are stored across-the-board for a device in the **TECHNOLOGY** folder. They can be interconnected with suitable technology objects of the device.

Proceed as follows

1. To create a new controller object, double-click **Insert controller object** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing controller object to the clipboard and then insert it under another name.

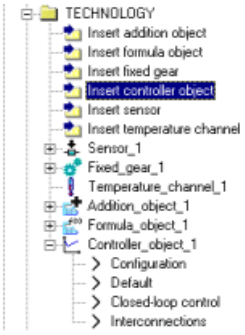


Figure 4-311 Representation of controller objects in the project navigator

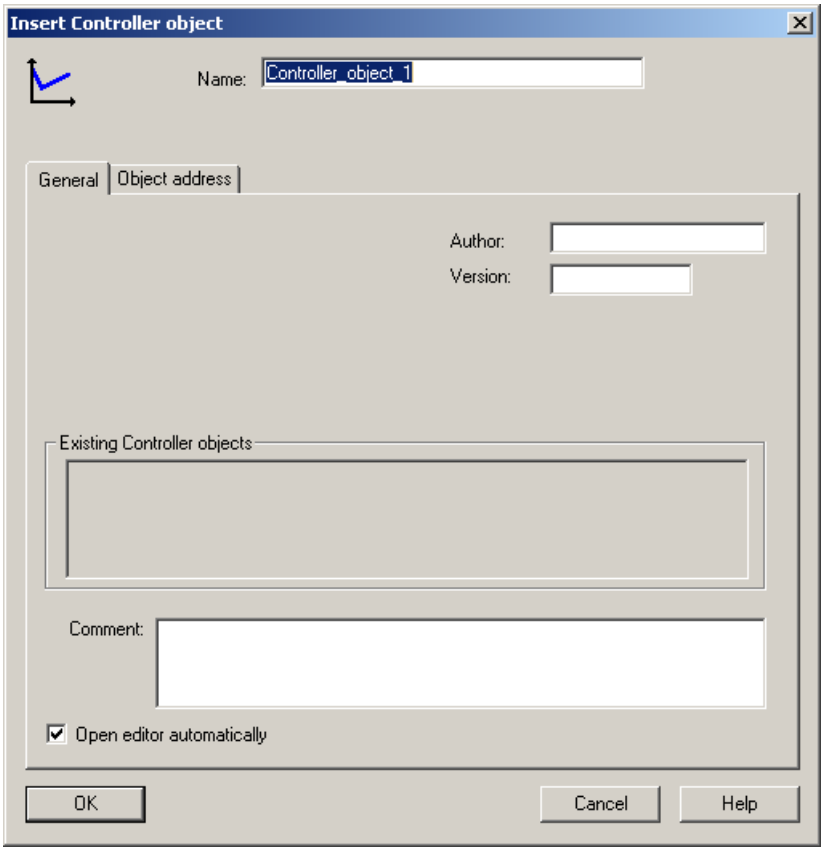


Figure 4-312 Inserting a controller object

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new controller object will be inserted under **TECHNOLOGY**.

Assigning parameters/defaults to a controller object

Proceed as follows

In the project navigator, double-click **Defaults** under the object.

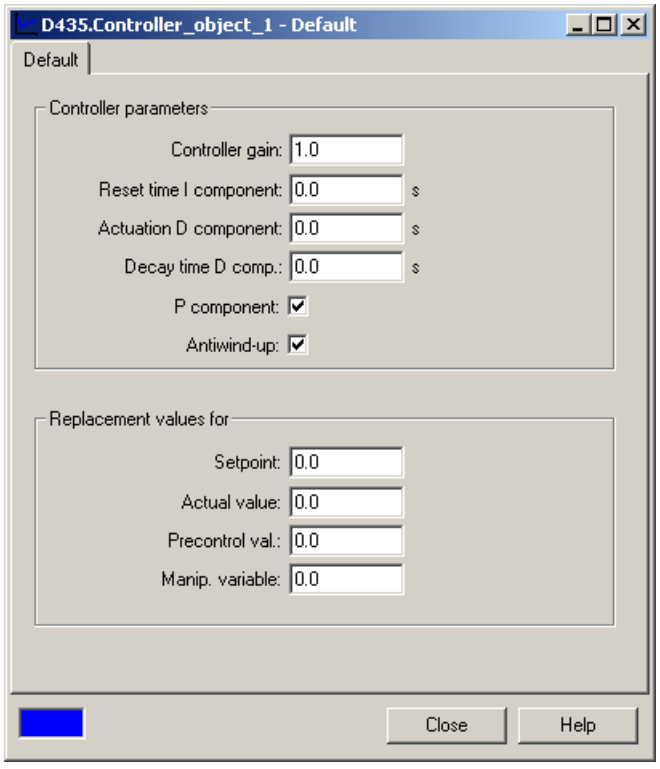


Figure 4-313 Controller object: Defaults

Controller object - defaults

Specify the parameters (defaults) for calling the controller object in this window (**_enableControllerObject** or **_disableControllerObject**). If you do not make any additional specifications during programming, these default values will be used.

You can set the following parameters:

Table 4-209 Controller object TO: Parameters that can be set for default purposes

Field/Button	Explanation/instructions
Controller parameters	Here, you enter the parameters for the controller.
Default value for	Default values can be specified individually for the **setpoint**, **actual value**, **precontrol value**, and **manipulated variable**. Here, you enter the default values.

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Overview of commands (Page 2054)
Function overview (Page 2043)
Description of function (Page 2045)

Configuring a controller object

Proceed as follows

In the project navigator, double-click **Configuration** under the object.

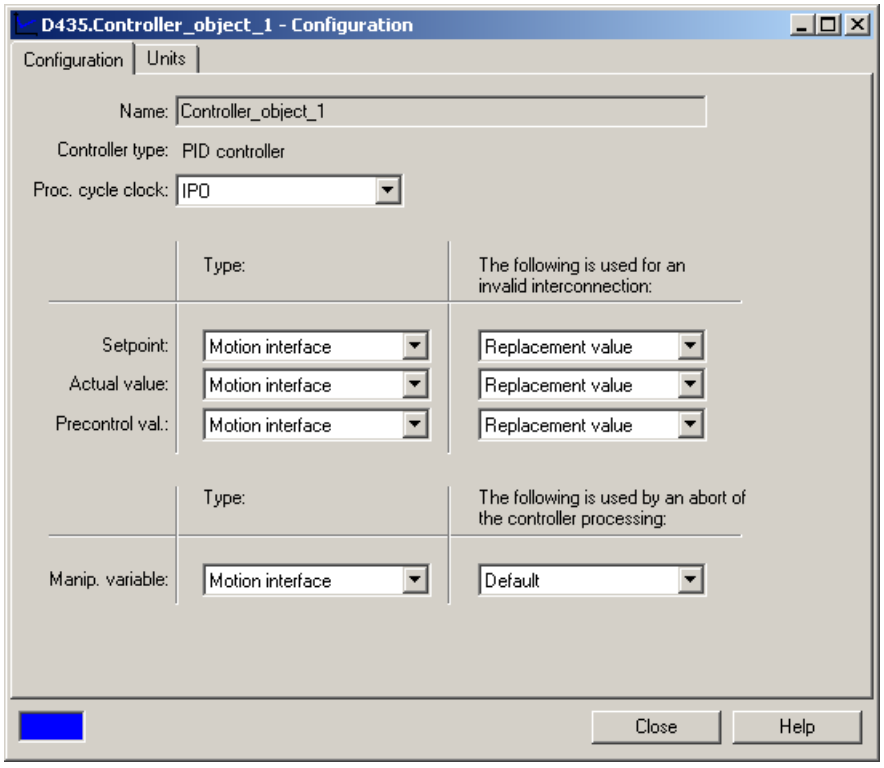


Figure 4-314 Configuring a controller object

Controller object - configuration

Define the properties of the controller inputs in this window.

You can set the following parameters:

Table 4-210 Controller object TO: Parameters that can be set for configuration purposes

Field/button	Meaning/instruction
Name	Display: Name of controller object
Controller type	Display: Controller type

Field/button	Meaning/instruction
Processing cycle clock	Here, you select the IPO-, IPO_2, Servo, Servo_fast or IPO_servo cycle clock.
Type	For **setpoint**, **actual value** and **precontrol value** you can specify whether the LREAL interface or the MOTION interface is connected on the input side. (Configuration data item **setpointIn/actualValueIn/precontrolValueIn._type**) When using the LREAL interfaces, the replacement values are used as input values. (System variables setpointDefault/actualValueDefault/precontrolvalueDefault). For the **manipulated variable**, you can specify whether the LREAL interface or the MOTION interface is connected on the output side. (Configuration data item **outputInterface._type**)
Applicable in the case of an invalid interconnection:	For each **setpoint**, **actual value** and **precontrol value**, you can specify whether the replacement value or the most recent valid value is to be used if the interconnection value is active, but invalid. (Configuration data item **setpointIn/actualValueIn/precontrolValueIn.behaviorByInvalidInterface**)
The following is applied if controller processing is aborted:	For the **manipulated variable**, you can specify whether the replacement value, the last valid value or zero must be output when controller processing is aborted by an alarm. (Configuration data item **outputValueErrorBehaviorMode**)

For additional information on the meaning of parameters and their permissible value ranges, please see the *SIMOTION reference lists*.

See also

Function overview (Page 2043)

Overview of commands (Page 2054)

Description of function (Page 2045)

Configuring closed-loop control

Proceed as follows

In the project navigator, double-click **Closed-loop control** under the object.

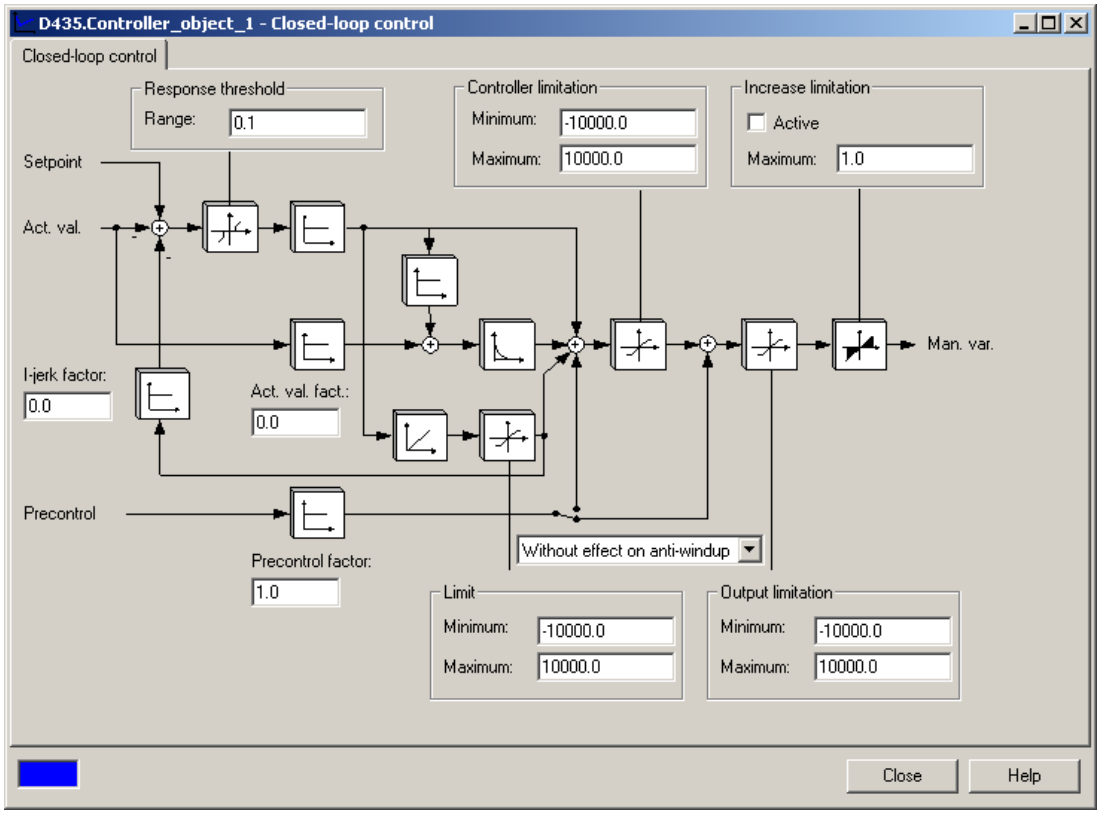


Figure 4-315 Configuring closed-loop control

Controller object - closed-loop control

In this window, specify the parameters for the closed-loop control.

You can set the following parameters:

Table 4-211 Control: Parameters that can be set for configuration purposes

Field/Button	Explanation/instructions
Threshold on	Here, you enter the response threshold for system deviation.
Controller limitation	Here, you enter the limiting for the controller.
Increase limitation	Here, you enter the limiting for the output signal rise.
I-jerk factor	Here, you enter the weighting factor for the I-component.
Actual value	Here, you enter the factor for the actual value.
Switch	Here, you can specify whether precontrol feedforward occurs before or after the anti-windup limiting (default: after).

Field/Button	Explanation/instructions
Feedforward control factor	Here, you enter the factor for the feedforward control.
Limit	Here, you enter the limiting for the actuating signal.
Output limitation	Here, you enter the limit values for the output signal.

See also

Overview of commands (Page 2054)
Function overview (Page 2043)
Description of function (Page 2045)

Interconnecting a controller object

Proceed as follows

1. In the project navigator, double-click **Interconnections** under the object.

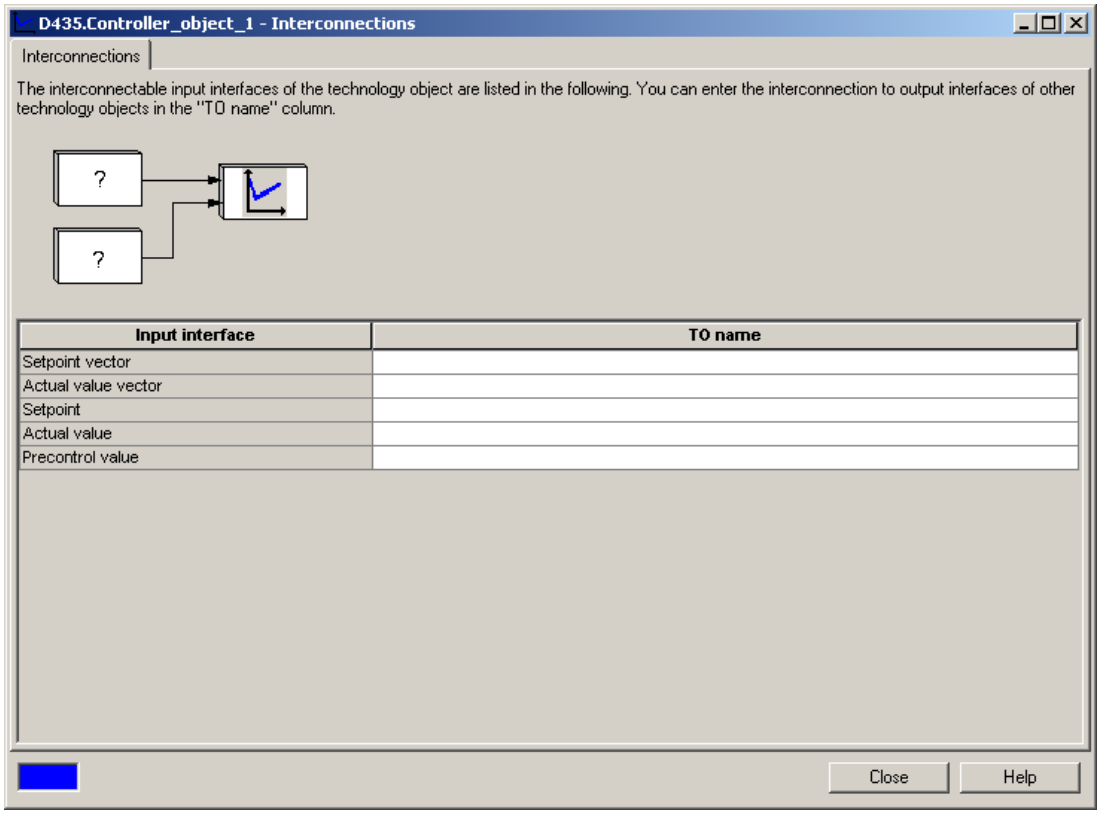


Figure 4-316 Interconnecting a controller object

In this window, interconnect the inputs of the controller object (with axes, for example).

2. To do so, click in the corresponding input field and then select the desired object. (The objects must have already been created.)

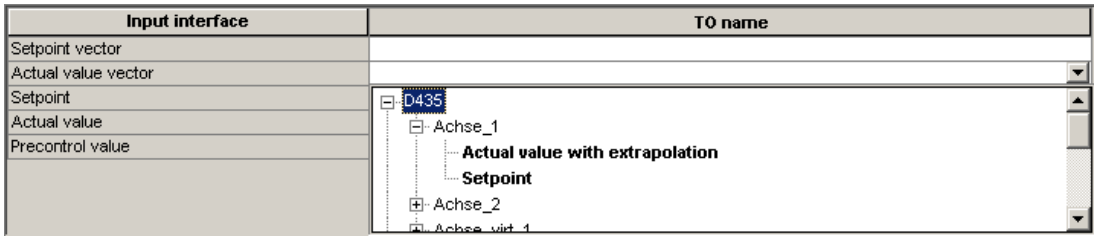


Figure 4-317 Interconnecting a controller object with an axis (motion vector)

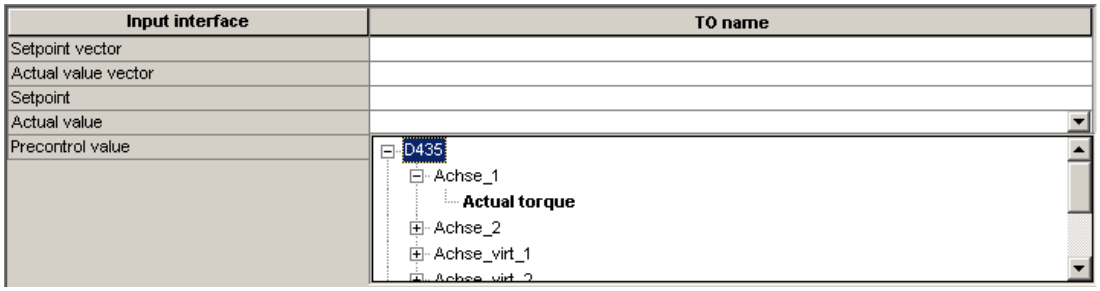


Figure 4-318 Interconnecting a controller object with an axis (scalar)

Note

For additional information, please refer to the following manuals:

Function Manual *Motion Control Basic Functions*, "Interconnection of Technology Objects"

Function Manual *Motion Control Technology Objects Axis, Electric/Hydraulic, External Encoder*, "Traversing using Motion Vectors"

4.4.6.4 Programming a Controller Object/References

Programming

Overview of commands

Table 4-212 Controller object TO: Overview of commands

Command	Functionality
_enableControllerObject	Enabling/disabling a controller
_disableControllerObject	
_enableControllerObjectIn	Activating/deactivating interconnection interfaces on the input side
_disableControllerObjectIn	
_setControllerObjectPIDControl	Set controller parameters
_resetControllerObject	Reset controller object
_resetControllerObjectError	Reset error
_resetControllerObjectConfigDataBuffer	Delete collected configuration data
_bufferControllerObjectCommandId	Store CommandId and command status temporarily
_removeBufferedControllerObjectCommandId	Delete CommandId
_getStateOfControllerObjectCommand	Read out command status
_getControllerObjectErrorNumberState	Read out error number status

Note

For a complete list of all commands and their syntax, the system variables, and error messages, please see the SIMOTION Reference Lists.

Commands

Enabling/disabling interconnection interfaces on the input side

The input-side interconnection interfaces can be enabled or disabled using commands **_enableControllerObjectIn**/**_disableControllerObjectIn**.

- If the inputs are enabled, the input values are used.
  If an interconnection error occurs when interfaces are enabled, configuration data element **behaviorByInvalidInterface** can be used to specify whether the most recent valid value (LAST_VALID_INTERFACE_VALUE) or the default value (DEFAULT_VALUE) is used.
- If the inputs are disabled, the default values are used.
  (The values are applied cyclically or updated when a change occurs.)
  Actual value (parameter **actualValueIn**), setpoint (parameter **setpointIn**), and manipulated variable (parameter **precontrolValueIn**) can be enabled or disabled with a command.

Activating/deactivating a controller

- **_enableControllerObject**: Enables the controller object
  The controller is inactive after TO power-up and is enabled using the command.
  The controller object and interconnections are active.
  The controller parameters must be set in the configuration.
  The following values are active after power-up.
  - When the controller is first enabled, the configuration and defaults values are used first. The values last set are displayed in system variable **pidControllerEffective**.
  - When the controller is next enabled, the currently set values are active.
- **_disableControllerObject**: Disables the controller object
  The **valueBehaviorMode** parameter can be used to set one of the following functions:
  - Retain the last value (LAST_VALUE)
  - Use the default value (DEFAULT_VALUE)
  - Output the zero value (ZERO_VALUE)

Local alarm response

Technological alarms

The standard technology object alarms, e.g. interconnection error or illegal parameter, are output.

Local reactions

If an error occurs, the following local responses are possible:

- No response (**NONE**)
- Stop command decoding (**DECODE_STOP**)
- Abort of command processing (**CONTROLLER_STOP**)
  The **outputValueErrorBehaviorMode** configuration data element can be used to set one of the following functions:
  - Retain the last value (LAST_VALUE)
  - Use the zero value (ZERO_VALUE)
  - Use the default value (DEFAULT_VALUE)

  (default: LAST_VALUE)
- Deactivation of controller (**DISABLE_CONTROLLER**)

Menus

Controller object - menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the controller object selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration	This function opens the configuration for the controller object selected in the project navigator. Define the properties of the controller inputs in this window.
Factory setting (default)	This function opens the defaults for the controller object selected in the project navigator. Specify the parameters for calling the controller object in this window (**_enableControllerObject** or **_disableControllerObject**).
Closed-loop control	This function opens the closed-loop control for the controller object selected in the project navigator. In this window, specify the parameters for the closed-loop control.
Interconnections	This function opens the interconnections for the controller object selected in the project navigator. In this window, interconnect the inputs of the controller object, e.g. to axes.
Expert	This function opens the submenu for the expert settings.

Function		Meaning/Note
	Expert list	This function opens the expert list for the controller object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.

Controller object - context menu

Grayed-out functions cannot be selected.

You can select the following functions:

Function		Meaning/Note
Open configuration		This function opens the configuration for the controller object selected in the project navigator. Define the properties of the controller inputs in this window.
Factory setting (default)		This function opens the defaults for the controller object selected in the project navigator. Specify the parameters for calling the controller object in this window (**_enableControllerObject** or **_disableControllerObject**).
Closed-loop control		This function opens the closed-loop control for the controller object selected in the project navigator. In this window, specify the parameters for the closed-loop control.
Interconnections		This function opens the interconnections for the controller object selected in the project navigator. In this window, interconnect the inputs of the controller object, e.g. to axes.
Expert		This function opens the submenu for the expert settings.
	Expert list	This function opens the expert list for the controller object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list. See Basic functions - expert list
	Configure Units	This function opens the Configure Object Units window in the working area. You can configure the units used for the selected object here.
	Import object	Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.
	Save project and export object	Use Save project and export object to open a window for an XML export. You can define the parameters for the XML export in this window.

4.4.7 Part VI - Temperature controller

4.4.7.1 Overview of Temperature Controller

Function overview

The **temperature channel** technology object provided in SIMOTION allows you to configure temperature controllers that cover all basic functions of temperature control, from actual value measurement and closed-loop control right through to generation of the actuating signal, including identification of temperature control systems and calculation of the resulting channel parameters.

Application

The **temperature channel** technology object is specifically optimized for temperature control tasks in machines, e.g., for plastics processing machines. It can be configured as a heating channel, a cooling channel, or a combination heating/cooling channel.

4.4.7.2 Fundamentals of Temperature Controller

Principles of operation for a temperature controller (temperature channel TO)

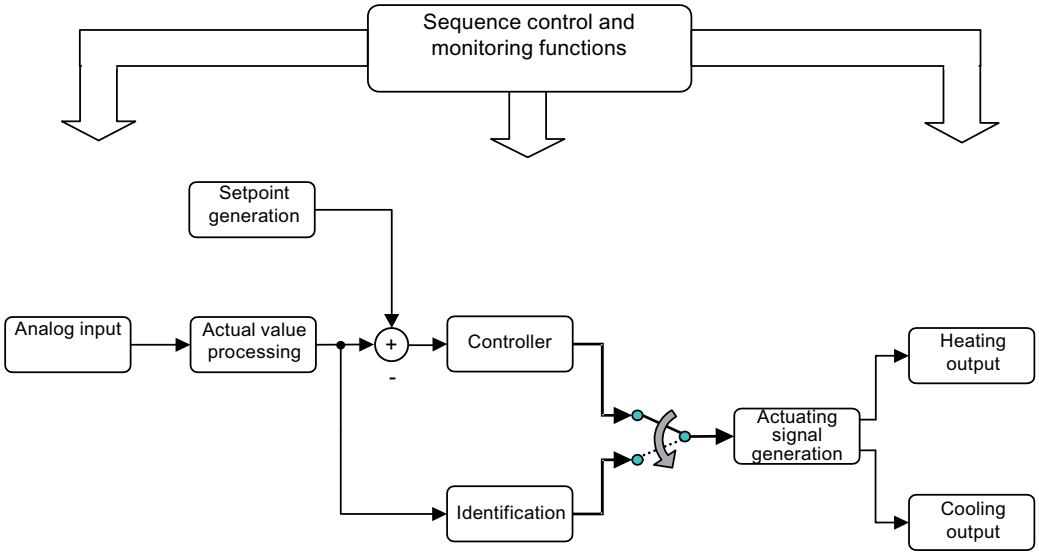


Figure 4-319 Signal processing of the temperature channel

The actual value measured (and linearized according to the sensor characteristic curve) at the analog input is filtered during actual value processing and subjected to a plausibility test to compensate for signal disturbances or sensor errors. This smoothed and tested signal is then processed further, either during closed-loop control or during system identification.

The current system deviation is determined in the controller using the setpoint specified during setpoint generation. For an optimal closed-loop response, different parameter sets are used for

heating and cooling, and the system switches from one parameter set to the other according to the sign of the actuating signal.

The analog actuating signal calculated by the controller or used for system identification is pulse-length modulated during actuating signal generation and routed to the digital outputs for heating and/or cooling.

Note

Temperature controllers with nonexistent physical modules cannot be loaded.

Functional scope

The following table provides an overview of the main functions of the **Temperature Channel** technology object.

Table 4-213 Temperature controller (temperature channel TO) Functional scope

Function	Content
Controller	
Control process	The temperature channel controllers normally operate as DPID controllers. Another type of temperature channel controller provides additional tuning options. Main features of the ADVANCED controller type: • Sampling time adaptation • Parameter adaptation • Control zone • Stop and hold cycles Additional features include: • High level of dynamic response and control performance • Robustness with regard to disturbance variables
Configurable channel structure	You can configure the temperature channel for closed-loop control as a **heating system**, a **cooling system**, or a **combination heating/cooling system**. Separate parameters are used for heating and cooling.
Operating modes	For each temperature channel, one of the following operating modes can be selected: • Inactive • Rules • Actual value acquisition and output of the manual manipulated variable value • Actual value acquisition and output of a manipulated variable value of 0 • Identification (of system parameters for self tuning) In Inactive mode, actual value acquisition does not take place, and a manipulated variable value of 0 is output.
Actual value acquisition and processing	

Function	Content
Actual value plausibility check	Each new actual value is checked for plausibility and corrected before appropriate filtering is applied. Values outside of the tolerance range are considered to be outliers, which, if they occur frequently, can cause error messages to be generated. A plausibility check is carried out, implausible actual values are filtered out and the expected actual value is extrapolated.
Actual value filtering	Filtering takes place separately for the actual control value and the actual display value. A low-pass filter (PT1 function) is used for actual value filtering. Parameters can be assigned for the time constant, or the time constant can be calculated by self-tuning. This ensures that the "smoothed" actual display value is free of the short dips that can be caused, for example by disturbance signals.
Actuating signal preparation and output	
Digital, pulse-length modulated actuating signal	An actuating signal is calculated by the controller as a percent value (in reference to the connected heating/cooling power, +100% to -100%) and output to manipulated variable processing. From this actuating signal, the manipulated variable processing derives a digital, pulse-length modulated signal that is output to the power switch by means of the corresponding output module.
Response times	Because the switching cycle of mechanical relays is supposed to be kept to a minimum, actuating signal pulses that are too small are suppressed by the response time (minimum switch-on duration). Operating (ON) times that are less than the response time or components that are truncated by PWM quantization are not output; rather, they are summed up over several manipulated variable cycles until the specified response time is reached. Conversely, large actuating signal pulses (close to 100 %) can lead to very short turn-off times. If a minimum switch-off duration (e.g., 5%) is specified, these types of pulses are always output at 100%, and the difference relative to 100% is subtracted again from the current manipulated variable during the next output cycle.
Output of manual manipulated variable value	Output of manual manipulated variable value is an operating mode in which a definable manipulated variable value (manual manipulated variable value) is output (manual manipulated variable mode).
Calculation of average manipulated variable value	In closed-loop control mode, the temperature channel calculates an average manipulated variable value, which is read out by the operator or the application and can be used as a substitute manipulated variable value.
Identification	
System identification	In Identification mode, the temperature of the controlled system is changed by specifying a constant manipulated variable value. Process parameters are determined from the time delay and the rate of temperature change. System identification is only possible for a heating controller.

Function	Content
Calculation of parameters	Appropriate temperature channel parameters for the control system can be calculated from the process parameters.
Monitoring and alarm functions	
Actual value monitoring by definition of tolerance bands	The actual values of each channel are monitored for adherence to an inner and outer tolerance band. The inner and outer tolerance bands can be defined independently as absolute or relative tolerance bands: • In an absolute tolerance band, there is no relationship between the limit values and the setpoint; the limit values are fixed. • The relative tolerance band is always defined in relation to the current setpoint, and it changes whenever the setpoint changes. Response when tolerance bands are violated: • When the inner tolerance band is violated, an alarm (warning) is output. When the outer tolerance band is violated, the system reaction can be set separately for the upper and lower tolerance thresholds in the alarm configuration.
Plausibility check	The control loop plausibility check can detect errors in the control loop (sensor-controller-heating), thus preventing hazardous situations from occurring. The control loop plausibility check is only active in *Control* mode. In the case of a control loop plausibility error, an alarm is generated and the output is set to zero.
Gradient monitoring	Temperature changes in the controlled system can be monitored for adherence to maximum gradient values. Repeated violation of the maximum gradients is signaled by means of a system variable and an alarm.
Gradient check for the parameters	All parameters are checked to ensure that they agree mutually and do not exceed limit values.
Alarm functions	If errors occur or if activated monitoring functions respond, corresponding alarms are triggered by the temperature channel. The reaction of the operator/application to an alarm depends on the situation. In the case of serious errors, the temperature channel will react automatically in advance (local response) in order to avoid a dangerous situation (e.g., by outputting a manipulated variable value of "0" at the heating output).

4.4.7.3 Configuring the Temperature Controller

Creating a temperature controller

Temperature controllers (**temperature channel** TOs) are stored globally for all devices in the **TECHNOLOGY** folder.

Proceed as follows

1. To create a new temperature controller (temperature channel TO), double-click **Insert temperature channel** under **TECHNOLOGY** in the project navigator.
   You can also copy an existing **Temperature Channel** TO using the clipboard and insert it under another name.

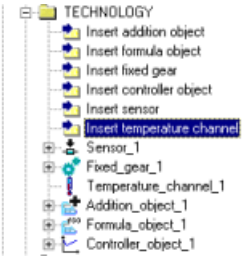


Figure 4-320 Representation of temperature controllers in the project navigator

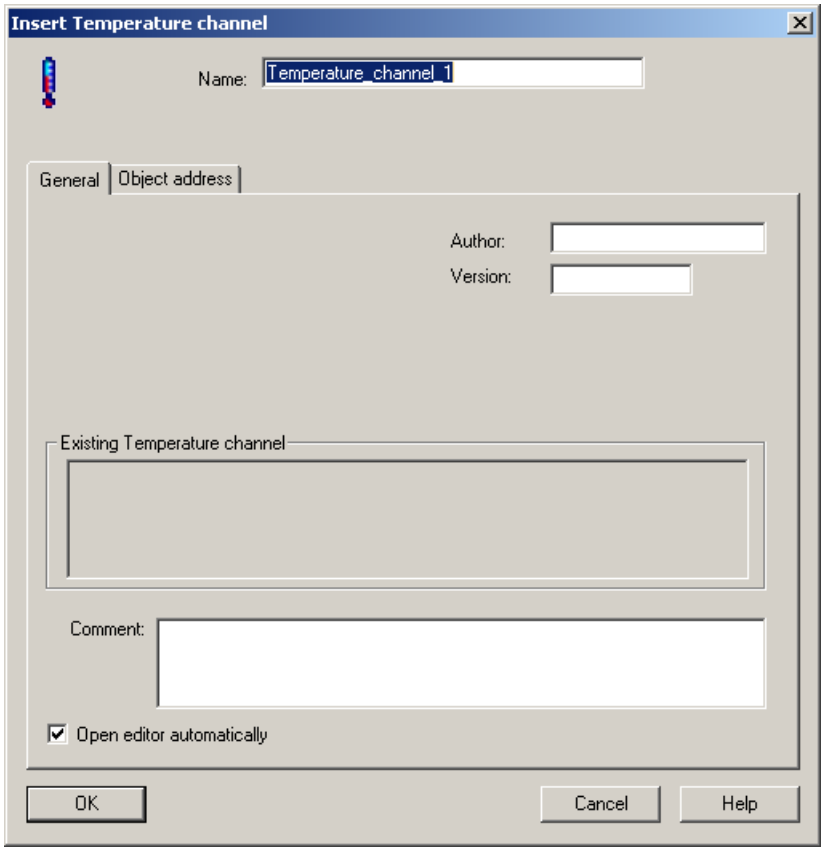


Figure 4-321 Inserting a temperature channel TO

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new temperature channel TO will be inserted under **TECHNOLOGY**.

Configuring the temperature controller in the expert list

Via the expert list, all configuration data and system variables for the temperature controller (**temperature channel** TO) can be displayed and changed in a list.

The configuration data of the **Temperature Channel** TO is already assigned and needs only to be adjusted for your configuration.

Note

The **Temperature Channel** TO can only be configured in the expert list.

For additional information, see the Function Manual *Motion Control Basic Functions*, "Expert List"

Configuring a temperature controller

Default values are already assigned to the parameters of a Temperature Channel TO (configuration data and system variables). The Temperature Channel TO uses these configuration data during power-up.

For each Temperature Channel TO, you can make changes to the configuration and parameter assignment of the following elements:

- Analog Input
- Controller
- Digital outputs

Operating parameters (.generalParameter)

Setting the operating mode for a temperature channel (.generalParameter.operatingMode)

Via the expert list, you can set the operating mode that the selected temperature channel is to assume automatically during the first power-up.

Table 4-214 Temperature controller (temperature channel TO) Operating modes (first power-up)

Configuration parameter	Description
Inactive No actual value acquisition, output of 0	The channel is not active, that is, no actual value measurement takes place, and a manipulated variable value of "0" is output. This is the default operating mode.
Control Control to a particular setpoint	In this operating mode, the predefined setpoint **generalParameter.setpoint** is used for control.
Measuring and manual output Actual value acquisition and output of the manual manipulated variable value	In this operating mode, the actual value is measured, and a manual manipulated variable value **generalParameter.manualOutputValue** is specified as the manipulated variable value.

Configuration parameter	Description
Measuring and output zero Actual value acquisition and output of 0	In this operating mode, the actual value is measured, and a manipulated variable value of "0" is output.
Identification System identification	This operating mode starts system identification, during which a constant manipulated variable value is specified, and the temperature of the controlled system is changed. Process parameters used to calculate appropriate channel parameters are determined from the delay and the rate of temperature change.

Setting the temperature setpoint (.generalParameter.setpoint)

The temperature setpoint for closed loop control mode is specified in this parameter.

Note

If **Control** mode (.generalParameter.operatingMode) is the default setting for the channel, then the system is controlled to this setpoint immediately following initial startup.

Setting the manual manipulated variable value (.generalParameter.manualOutputValue)

In this parameter, you specify the manual manipulated variable value for the selected channel. The manual manipulated variable value is output if you have assigned **Measuring and Manual Output** mode in **generalParameter.operatingmode**.

The manual manipulated variable value **manualOutputValue** can have a value ranging from +100% to -100%.

- **Heating controller: from 0% to 100%**
  (0: no heating power, 100: full heating power)
- **Heating/cooling controller: from -100 % to 100%**
  (-100: full cooling power, 100: full heating power)
  If the manual manipulated variable value is to be output at the cooling output of a heating/cooling controller, then it must have a negative value.
- **Cooling controller: from 0% to 100%**
  (0: no cooling power, 100: full cooling power)

Configuration data for the analog input

Here, you set the address of the analog input and specify the input handling of the actual values.

Entering the address (.input.device)

The address for this input must match the address of the analog input in the hardware configuration data.

Defining the sampling time ratio (.input.analog)

All actual values are read in during an input cycle that is dependent on the controller cycle and are then checked.

The ratio between the sampling time of actual value handling and the sampling time of the controller or identification is critical for optimal filtering of the actual values. For this reason, the sampling time ratio is user-defined.

Note

To filter out disturbances during actual value acquisition, the actual value must be measured more quickly than the controller cycle clock, so that smoothing (PT1 function) can occur. The larger and more frequent the disturbances, the larger the sampling time ratio should be.

Setting parameters for filtering of actual values (.input.analog.filterParameter)

Here, you specify the settings for smoothing of actual values. Not every input value is evaluated as it appears: Rather, a value that is filtered with a PT1 function is calculated.

Actual value handling.

The following functions are performed for each actual value during input handling:

1. Read in all measured actual values during a cycle that is generally dependent on the controller cycle.
2. Use a plausibility test followed by actual value filtering to check for disturbances and smooth the actual value
3. Tolerance tests of the current actual value are then carried out.

Defining time constants (controlTimeConstant, displayTimeConstant)

Actual value filtering takes place separately for the actual control value and the actual display value. For this reason, you can specify the following time constants:

- Time constant for filtering of the actual control value (control input filter)
- Time constant for filtering of the actual display value (display input filter).

A PT1 function is used for actual value filtering. This ensures that the "smoothed" actual display value is free of the short dips that can be caused, for example, by disturbance signals.

Parameters can be assigned for the time constants, or the time constants can be calculated by self-tuning. The time constant of the display input filter is greater than the time constant of the control input filter. The actual display value undergoes a more intense smoothing operation. The actual value smoothed with the display input filter is made available in system variable **TOTCx.actualInputData.displayValue**.

Setting the initialization value of the maximum and minimum measured actual display value (input.analog.displayValueParameter)

The current actual display value - as well as the maximum and minimum actual display values that are likewise calculated during the actual value processing - are always available for display and data recording.

The maximum and minimum actual display values determined during runtime can be reset via the system functions **_setTControllerInputDisplayValueParameter** and **_resetTController**.

Parameters for the actual value plausibility check (.input.analog.gradientCheckParameter)

Each new actual value is checked for plausibility before filtering is applied. The main purpose of the actual value plausibility check is to filter out short-term disturbances (outliers), as such disturbances can have a particularly negative effect on the controller performance due to the differential components in the control algorithm.

Activating/deactivating check mode

The plausibility check can be activated (ACTIVE) or deactivated (INACTIVE) with the **checkMode** parameter. The default setting for the plausibility check is INACTIVE.

Entering the number of tolerated violations

Here, you enter the number of violations of maximum gradients to be tolerated. All values outside the range of the maximum positive and maximum negative actual value gradients are considered to be violations.

If the number of tolerated violations is exceeded in direct succession, an alarm is generated.

Specifying the maximum positive and negative gradients

When you specify the maximum positive and maximum negative actual value gradients, you set the range of the actual value gradients. The input signal change is checked to determine whether it is within the specified value range.

Actual value monitoring by defining tolerance bands (.input.analog.limitCheckParameter)

The actual values of each channel are monitored for adherence to an inner and outer tolerance band.

Defining the inner and outer tolerance bands

The actual control value is checked for tolerance violation. For this purpose, you can assign parameters to each control channel for an inner and an outer tolerance band.

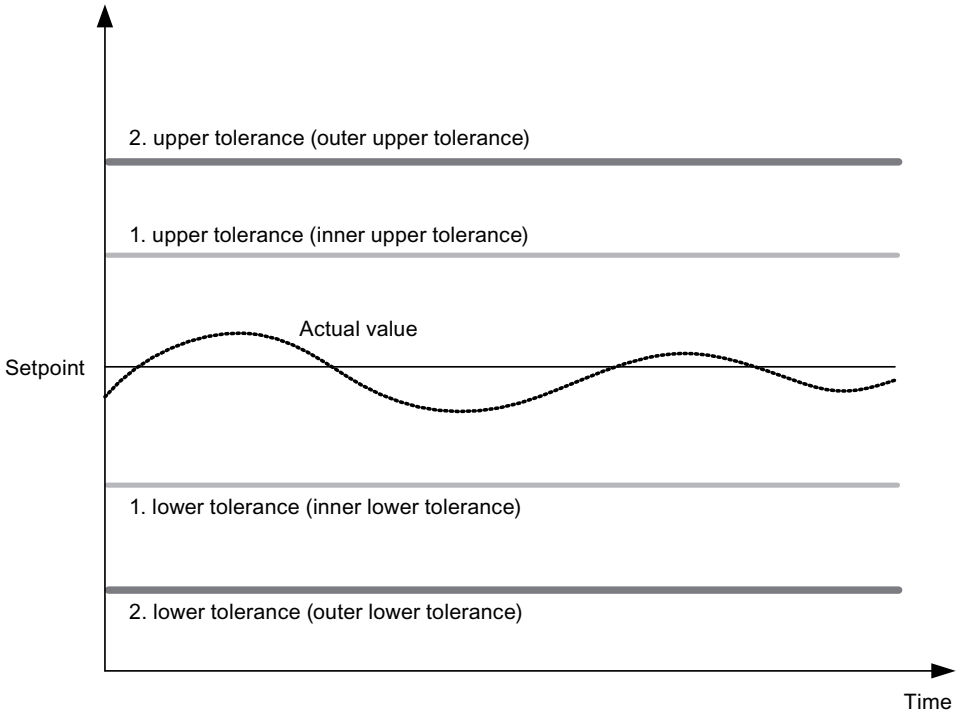


Figure 4-322 Inner and outer tolerance bands

The inner tolerance band is nested inside the outer tolerance band.

If the:

- Inner limit (1st tolerance band) is crossed, a warning (TO alarm) is output. There is no local reaction from the channel.
- Outer limit (2nd tolerance band) is also crossed, an error (TO alarm) is displayed, triggering a local reaction from the channel (for example, a heating output is reduced to 0% when the outer upper tolerance limit is exceeded).

Absolute or relative tolerance band

It is possible to parameterize an upper (**upperLimitValue**) and a lower (**lowerLimitValue**) limit value in each case for the inner and outer tolerance bands. Via **lowerLimitMode** and **upperLimitMode**, the limits can be set to apply as an absolute limit value or as a relative deviation from the current setpoint.

For an absolute tolerance band, the limit values are not dependent on the setpoint; they are fixed. A relative tolerance band is always defined in relation to the current setpoint, and thus it changes relative to changes in the setpoint.

In the case of relative tolerance, if a setpoint step change occurs, the tolerance limit that is further away from the new setpoint is corrected in parallel to the actual value to prevent an unwanted tolerance violation.

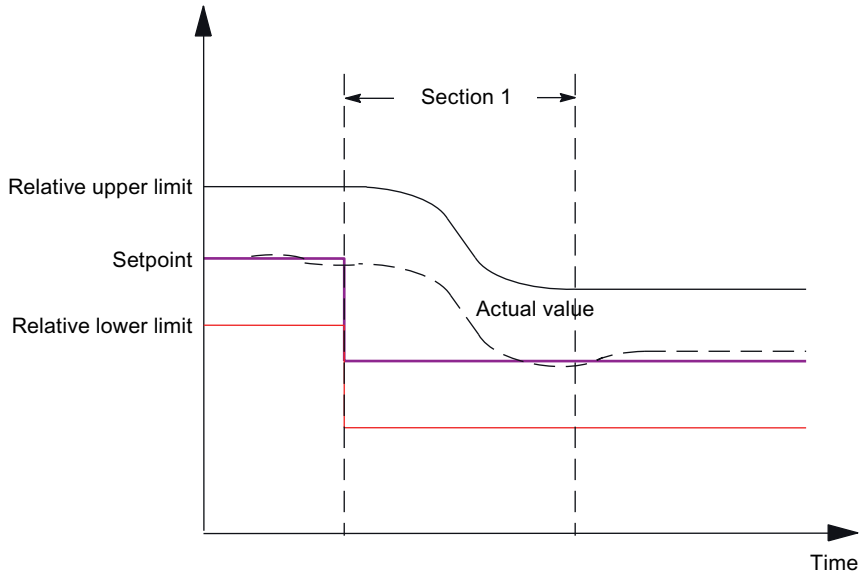


Figure 4-323 Correction of the relative tolerance band

If a relative tolerance band is configured and a setpoint step change occurs during control mode, the tolerance band check is corrected until the actual value has reached the new setpoint. Once this occurs, the tolerance check is once again active.

Tolerance violations

If the actual value is outside of a tolerance band, this means that one of the tolerance limits has been violated. The following mechanisms are used to indicate a tolerance violation:

- TO alarm sent to the application and alarm message on the HMI device
- Display in the **actualInputLimitCheckData** system variable

(See Reference List for temperature channel TO)

Tolerance band monitoring:

Tolerance band monitoring is reset under the following conditions:

- Modification of data for tolerance band monitoring
- Reset of the temperature controller
- Stop/Run
- Power supply off/on

Under these conditions, a relative tolerance band is corrected to the setpoint.

Configuration data for the controller

You set the configuration data in the **controller** data element.

Controller structure

You can configure the controller structure for **heating systems**, **cooling systems**, or **combination heating/cooling systems**.

Heating systems or cooling systems

For heating systems and cooling systems, there is only one controller, the primary controller.

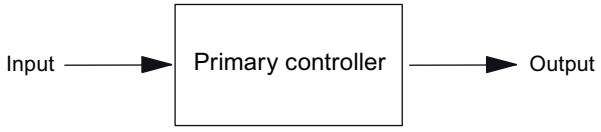


Figure 4-324 Controller structure for heating or cooling controller

- For **heating systems**, the primary controller works in a **positive effective direction**. Temperatures that are too low cause a rise in the output value.
- For **cooling systems**, the primary controller works in a **negative effective direction**. Temperatures that are too high cause a rise in the output value.
- If a control system can only be heated or cooled, only **one parameter set must be entered for a controller**.

Combination heating/cooling systems

Combination heating/cooling systems have two controllers:

- The **primary controller** works in a positive effective direction (**heating controller**). Temperatures that are too low cause a rise in the output value.
- The **secondary controller** works in a negative effective direction (**cooling controller**).
- If a control system can be both heated and cooled, then the **controller parameters must be entered for two controllers**.

During switchover phases, 0% is output at both outputs.

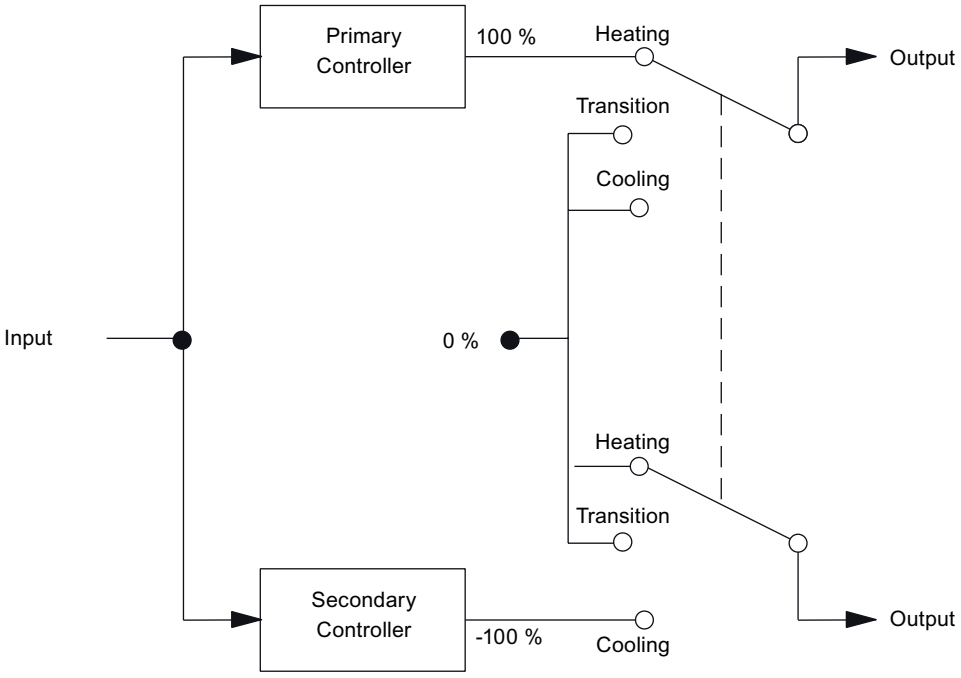


Figure 4-325 Controller structure for combined heating/cooling controllers

Controller parameters

The controller parameters depend on the controller structure:

- A dedicated heating controller uses a parameter set for heating and a dedicated cooling controller uses a parameter set for cooling (for example **controller.standard.DPIDParameter**).
- A combined heating/cooling controller uses two separate parameter sets, one for heating and one for cooling:
  The first parameter set is for heating, and the second parameter set is for cooling (for example, **controller.standard.DPIDParameter** - parameters for a heating controller and **controller.standard.DPIDParameterSecondary** - parameters for a cooling controller).

Selecting the controller type

You can specify the controller type under **controller.type**.

You can select between two DPID controllers that are especially designed to meet the requirements of temperature control systems:

- Standard DPID controller (**STANDARD**)
- Advanced DPID controller (**ADVANCED**)

ADVANCED controller

The ADVANCED controller provides additional tuning options while control mode is in operation (sampling time and parameter adaptation), as well as an assignable control zone:

- **Sampling time adaptation**
  If the absolute value of the system deviation increases sharply, the sampling time is reduced in the possible range by means of sampling time adaptation.
- **Parameter adaptation**
  Adaptation of the controller parameters for each operating point has advantages relating to control performance and dynamic response.
  Parameter adaptation can be especially optimized for:
  - Continuous processes (extruders)
  - Batch processes (injection molding machines, blow molding)
- **Control zone**
  The control zone improves the dynamic response of the control and reduces overshoots. Maximum cooling occurs above the control zone to be defined, and maximum heating takes place below the control zone. Within the control zone, the actuating signal calculated by the controller is output.

The following overview figure shows how a DPID controller operates:

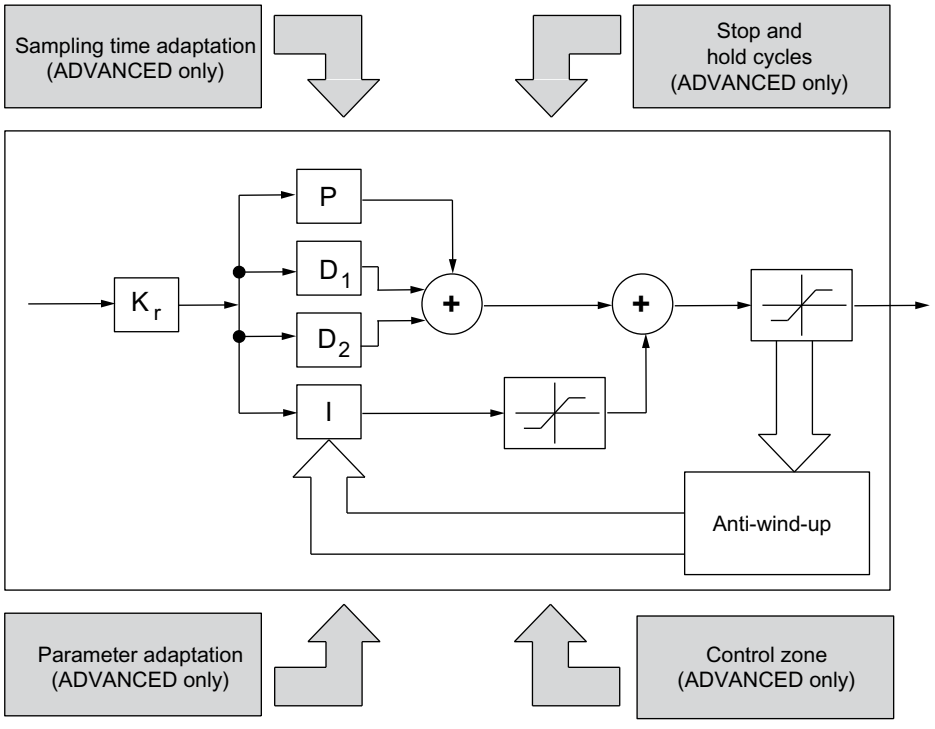


Figure 4-326 Stop and hold cycles for both controller types, plus control zone for ADVANCED

Both controller types have the following features in common:

- **Separate parameter sets for heating and cooling**
  For optimal control performance, the DPID controller uses separate parameter sets for heating and cooling. The controller switches from one parameter set to the other according to the sign of the actuating signal ("+" for heating and "–" for cooling). This also changes the controller sampling time.
- **Anti windup for the I component**
  Anti windup prevents rise of the I component when the actuating signal reaches its limit. If the manipulated variable output is limited, the calculation of the I component is stopped.
- **Stop and hold cycles** (ADVANCED controller type only)
  In operation, the D component is, for example, temporarily switched off to ensure a bumpless parameter transition during the switchover from a heating controller to a cooling controller (for combination controllers) or in the event of a setpoint jump. This behavior is performed by the technology object and cannot be specified.
- **Average manipulated variable value**
  During the control process, an average manipulated variable value is calculated. In settled state, this manipulated variable value can be used to switch temporarily from closed-loop control mode over to output mode without causing a large change in the current working temperature.

DPID parameters (.controller.standard.DPIDParameter[Secondary])

The following DPID parameters are available for the **ADVANCED** and **STANDARD** controller types:

- **Gain (K_R)**
  The gain corresponds to the proportional-action coefficient of the controller and is then calculated as **P = K_R * control deviation** (see the figure titled "Stop and hold cycles for both controller types, plus control zone for ADVANCED").
  You can select values between the controller limit values for **minimum controller gain** (.limits.controller.minGain) and **maximum controller gain** (.limits.controller.maxGain).
- **Switch I component on/off**
  With this parameter, you can switch the integral component of the controller on or off.
- **Reset time**
  With the reset time, you set the I component of the controller.
- **Actuation time**
  With the first order actuation time, you set the D component of the controller, and with the second-order actuation time, you set the D2 component. If you enter a value of 0, the respective D component is switched off. In this way, you can implement a P controller or a PI controller.
- **Start value of integrator following RESET**
  With this start value, you specify the initialization value of the I component when (re)starting the SIMOTION device. The initialization value is also activated by the **_resetTController** system function.

Controller cycle parameter (.controller.standard)

The dynamic response of the controlled system determines the length you must set for the controller cycle time. To control rapid temperature changes, the controller cycle time must be shorter than for control of slower temperature changes.

There are two speed classes (**.execution.executionlevel**) for both the heating and the cooling controller for the controller cycle time **n** (with n = 1, 2, etc.). T_1 and T_2 depend on specific task settings in SIMOTION SCOUT:

- High-speed controllers: n * T_1
- Slow controllers: n * T_2 (with $T_2 > T_1$)

Note

If you specify a controller cycle time that is not within the time base, the controller cycle time is rounded up to the next time-base increment. For slow controllers, the cycle time must not < T_2.

Parameters and sampling time adaptation (.controller.advanced.processModeParameter)

This adaptation type can only be adjusted for the **ADVANCED** controller type. It is determined by the system features.

While the controller is in operation, the controller parameters of adaptive controllers are constantly being adapted. This adaptation has a stronger effect on the DISCONTINUOUS process type than on the CONTINUOUS process type. This affects the following parameters:

- Gain
- Reset time
- Actuation time
- Controller cycle time

Control zone parameters (.controller.advanced.controlRangeParameter)

You can only make control zone parameter settings for the **ADVANCED** controller type.

The control zone is a band on either side of the current setpoint that is formed by a value for the upper limit and a value for the lower limit.

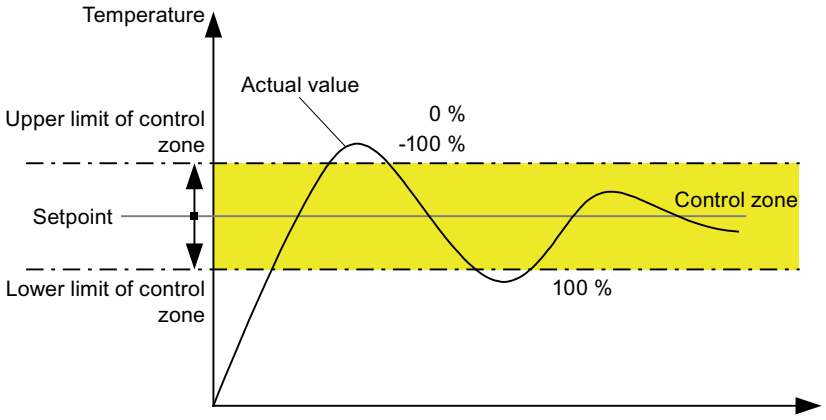


Figure 4-327 Control zone

If the actual value falls outside of the control zone, the controller responds as follows:

- If the actual value falls below the control zone, the controller outputs heat at the maximum manipulated variable value.
- If the actual value exceeds the control zone, the controller outputs a manipulated variable value of 0% (heating controller) or it cools on the basis of the maximum manipulated variable value.

Controller plausibility check (.controller.standard.upper/lowerPlausibilityParameter[Secondary])

Control loop plausibility is used to detect failure of final controlling elements.

The **controller plausibility is violated** if in a specified temperature range (within the upper and lower temperature limits) and starting from a particular output capacity (the lower manipulated variable value limit) a minimum rise of the actual value is **not achieved** within a specified time (delay time). A violation of the controller plausibility is displayed in the **ActualDPIDData.plausibilityState** system variable. Two separate temperature ranges can be distinguished to increase the sensitivity of this monitoring process.

For the primary controller, there are two parameter sets, which are handled separately. If a secondary controller is present, there are two additional parameter sets for this controller.

Configuration data for identification

Identification is available only for dedicated heating systems. The parameters are determined according to the inflectional tangent method.

Identification method (.identification.actualIdentificationType)

Once the system parameters are identified, they can be used to determine appropriate controller parameters for each controller type (ADVANCED or STANDARD). The two methods described below can be used for identification:

- With the **standard tangent method**, the system is activated with a constant, definable actuating signal until it reaches a static equalization. The system parameters determined are time delay (T_U), rise time/equalization time (T_A), and system gain (K_S).
- With the **modified tangent method**, the system is excited with the maximum actuating signal (100 %) until the inflection point is detected. The system parameters determined are delay (T_U) and the maximum temperature rise S_{max}(100 %) with reference to a 100 % actuating signal. With these parameters, only a simplified determination of the system parameters is possible. The advantage of the modified tangent method as compared to the standard tangent method is a considerably shorter identification time.

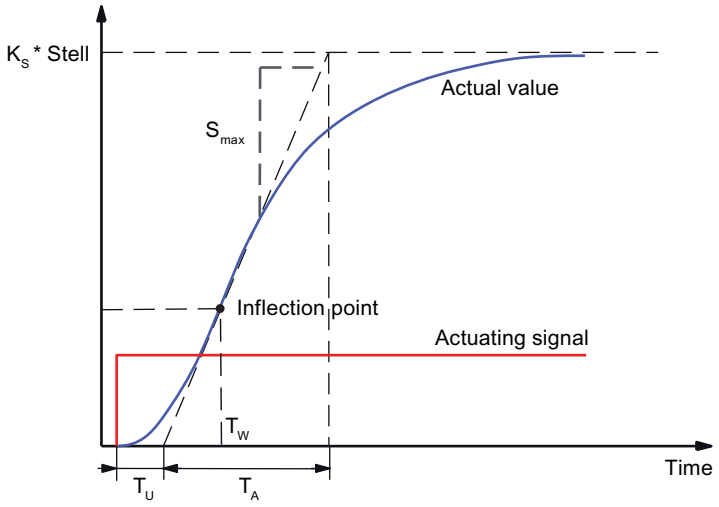


Figure 4-328 Tangent method

Note

The basic functions of the closed-loop control (such as actual value acquisition and actuating signal output) are also used for self-tuning.

The phases of self-tuning are monitored in relation to the runtime, and self-tuning is aborted with an alarm if the time limit is exceeded.

If self-tuning is called at a too high basic cycle clock, it might happen that the internal limit for the temperature rise of 5°C/basic cycle clock is exceeded. In this case, self-tuning goes on running and the determines values for the set basic cycle clock. The following alarm is issued: 30014 "Selected controller sampling time too small for the measured temperature rise".

The sampling time of the identification corresponds to the controller sampling time which can be set with the **.controller.maxControllerCyle** configuration data element.

The following times apply for both methods:

SETTING_UP	10 * max. controller sampling time
SEARCH STARTPOINT	Maximum equalization time

HEATING	5 times maximum delay time
SEARCH INFLECTIONPOINT	Maximum equalization time

Additionally for the standard tangent method:

SEARCH ENDPOINT	Maximum equalization time

Table 4-215 Temperature controller (temperature channel TO) Operating modes (identification)

Variable type	Description
Self-tuning enable (.available)	If self-tuning is available for a heating channel, then you set the AVAILABLE parameter value. If you set the NON_AVAILABLE parameter value, the required program section for self-tuning of a Temperature Channel technology object will not be loaded.
Start mechanism (.transitionMode)	You use the start mechanism to specify whether the temperature controller should switch automatically to the heating phase once the start conditions have been satisfied (AUTOMATICALLY) or whether it should wait for an explicit start command (BY_COMMAND) before switching. **Note:** For interaction with the **TControl** function object, you must set the BY_COMMAND value (any other value will be automatically overwritten by the **TControl** function object with BY_COMMAND).
Start conditions (.startCondition)	Before the heating phase is entered, the actual value must have attained a stable idle state. You set the idle state using the waiting time (.waitingTime) and the tolerated temperature change during the waiting time (.permissableTemperatureChange).
Minimum step size of actual value/setpoint (.minimumStepSize)	Before the heating phase is entered, a check is performed to determine whether the temperature difference between the current actual value and the setpoint to which the system is to be heated for parameter identification is at least as large as the configured minimum step size (.minimumStepSize). If it is not, then self-tuning is aborted.

Output handling of measured values

General

An actuating signal is calculated by the controller as a percent value (-100% to +100%, with reference to the connected heating/cooling capacity).

This value is then used to derive a digital, pulse-length modulated signal that is output to the power switch by means of the corresponding output module.

Actuators of temperature controlled systems are usually digitally-controlled mechanical or solid-state relays with no-voltage releasing magnets that are used to switch on and off electrical strip-

type heaters or heat-exchanger units (valves for liquid coolants, such as water, or a fan for air cooling).

These actuators are activated by means of separate pulse-width modulated, digital outputs for the heating circuit and the cooling circuit.

Output value/control signal

Generation of the actuating signal converts the analog actuating signal to a proportional switch-on time within the controller cycle clock separately for the heating output and the cooling output.

For pulse-length modulation, an actuating signal of 100 % corresponds to a control output being activated all the time. Smaller actuating signals are converted to proportional switch-on times within an manipulated variable cycle clock.

The low pass behavior of the temperature controlled systems has a smoothing effect on this pulsated characteristic if the manipulated variable cycle clock is sufficiently small in relation to the system time constants (manipulated variable cycle clock < 1/10 system time constant, for example sampling time).

The accuracy of the manipulated variable resolution depends on the quantization resolution of the on-time as compared to the manipulated variable cycle clock. Resolutions between 1% and 5% are a good compromise between the system load for identifying the manipulated variable and a reasonable control resolution.

Clocked actuating signal output

The controller outputs the complete actuating signal in one block at the beginning of each controller cycle clock. This might result in a saw-tooth temperature characteristic.

The "stroke of the saw-tooth" can be minimized by outputting in several blocks, e.g. 8 blocks at intervals of 250 ms instead of 1 block at intervals of 2 s. (V4.0 and higher)

The number of blocks can be set via the **numberOfOutputCycles** configuration data element (default: 1).

Making settings for pulse-width modulation (.output.out[Secondary])

Specifying output parameters for pulse width modulation

You specify output parameters for pulse width modulation (upper and lower threshold values for the manipulated variable value) to avoid insufficient switching intervals for the output of a dedicated heating or cooling controller or for the cooling output of a combined heating/cooling controller.

- The following applies to all manipulated variable values that exceed the upper threshold: The controller outputs the maximum manipulated variable (100%).
- The following applies to all manipulated variable values that fall below the lower threshold: The controller outputs the minimum manipulated variable value (0%).

Here, the missing absolute manipulated variable values are summed up and the corresponding switching correction is made.

Specifying the address and number of the digital output

Heating controllers and cooling controllers have a digital output for which you must enter an address.

A combination heating/cooling controller has two digital outputs that operate between 0% and 100%, depending on the manipulated variable value. Parameters of the first digital output are assigned for heating and parameters of the second digital output are assigned for cooling. You must enter addresses for both digital outputs.

Note

The address for this digital output must match the address of the corresponding digital output in the hardware configuration data.

Specification of limit values

You can specify the following types of limit values:

- General limit values (.limits.general)
- Controller limit values (.limits.controller)
- Limit values of the system to be controlled (.limits.process)

System cycle clocks and execution speed

Activation in the execution system and setting the cycle clocks

The Temperature Channel TO utilizes dedicated tasks in the execution system. These must be enabled through selection of **Use system tasks for TControl** when the temperature channels are used.

1. Call the configuration window **System Cycle Clocks** in SCOUT via **EXECUTION SYSTEM > Expert > Set system cycle clocks**.
2. Display the settings for the TControl system tasks via key **TControl**.
3. Activate the **Use system tasks for TControl** checkbox.

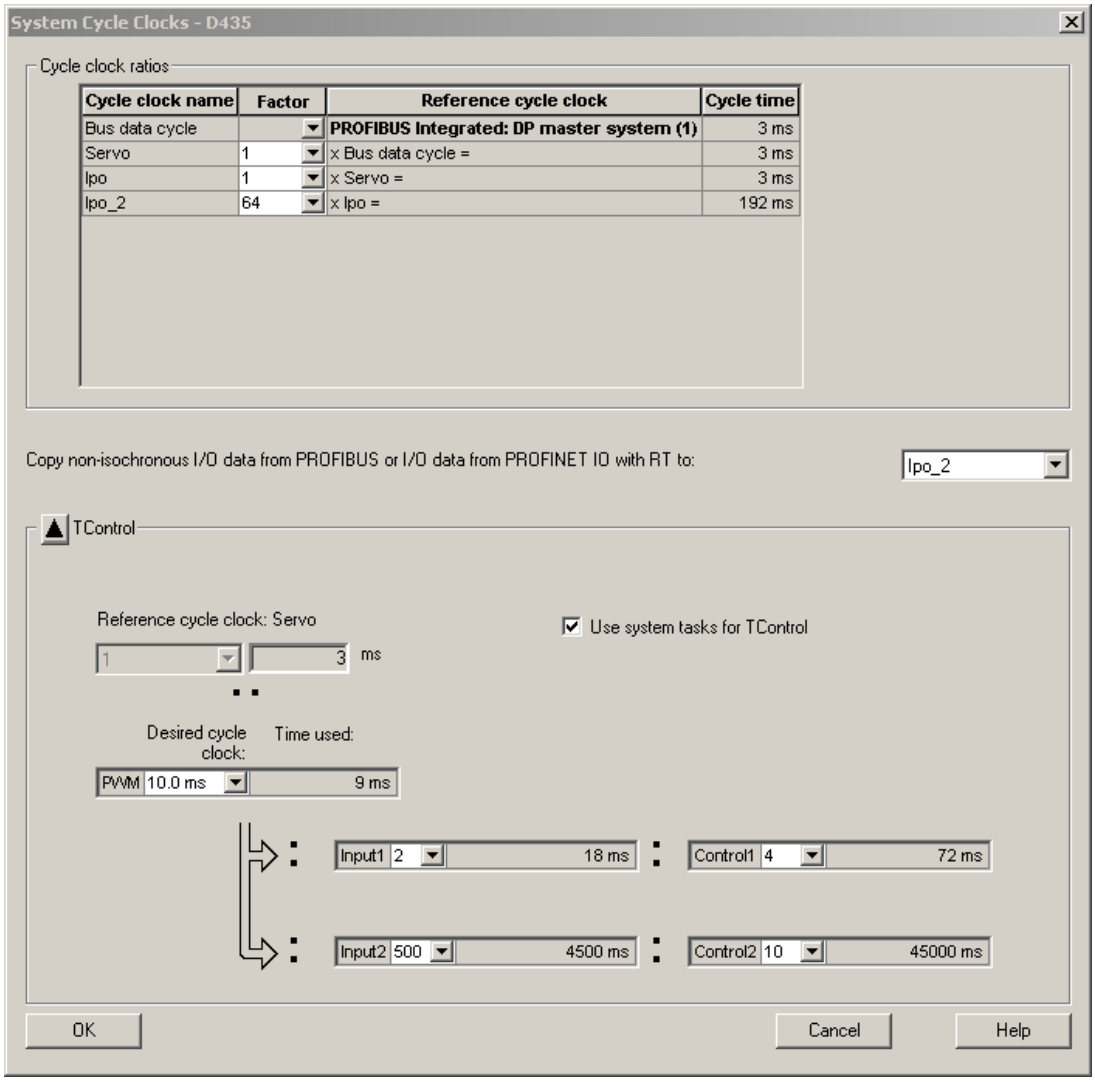


Figure 4-329 System cycle clocks for temperature control

Table 4-216 Temperature controller (temperature channel TO) Meaning of temperature channel tasks

Task	Description
TCInput_Task_1	Actual-value acquisition
TCTask_1	Temperature control
TCPWM_Tasks	Pulse width modulation (PWM) cycle clock time at digital output

Speed classes

Two classes of speed are available depending on which pulse width modulation (**PWM**) is selected.

The speed class specifies the time base to be used as the basis for the controller cycle time. The speed class can be set via configuration data element **controller.execution.executionlevel**.

Table 4-217 Temperature controller (temperature channel TO) Assignment of speed classes

Speed class	classification
FAST (T_1)	Input 1 = Read actual values
	Control 1 = Controller cycle
SLOW (T_2)	Input 2 = Read actual values
	Control 2 = Controller cycle

These times define the cyclical task start times (system time base) for pulse width modulation, controller and actual-value processing.

Controller sampling time

The following configuration data from the **expert list** are available for this purpose:

Table 4-218 Temperature controller (temperature channel TO) Description of configuration data elements

Configuration parameter	Description
controller.standard. cycleParameter	This time defines the actual call time of the controller. The time you set must exceed the time configured under Control1 if you have selected the FAST speed class or, if you have selected the SLOW class speed, the time configured under Control2. The controller will automatically round the values to an integer multiple of Control1/Control2.
Input.analog.relation ControllerCycle-toInputCycle	Ratio between controller and actual-value processing. The ratio set should match the ratio between Input1 and Control1/Input2 and Control2. The actual-value processing is faster by the set factor.

Distribution of actuating signal output

The actuating signal is output on the basis of a specific distribution. As a result, not all controllers are included in the same cycle clock, thereby minimizing peaks in terms of the system load. This is achieved by delaying the controller call within a controller sampling time. In the case of two controllers, for example, the first controller will be called immediately, but the second will only be called once 50% of the sampling time has elapsed. If the ratio is smaller than the number of controller channels, then several controllers will have to be processed at the same time. For example, in the case of 4 channels and a ratio of 3: The first controller will be processed at the beginning, the second once 33% of the time has elapsed, the third once 66% has elapsed and the fourth at the beginning.

Example:

In the example below, 43 channels need to be configured with a cycle time of 1.008 s.

Make the following settings in the expert list:

ContollerCycleTime	1.008 s

Make the following settings in the "System Cycle Clocks" dialog:

Servo cycle clock	Selection 1	3 ms
PWM - Task [1]:	10 ms	Rounded to: 9 ms
System time base of actual-value acquisition (Input 1):	Distribution ratio 2	9 ms (PMV) * 2 --> 18 ms
System time base of controller (Control 1):	Distribution ratio 4	18 ms (Input 1) * 4 --> 72 ms

[1] Set integer multiple of the basic cycle clock. If you specify a controller cycle time that is not within the time base, the controller cycle time is rounded up to the next time-base increment (in the example: 9 ms). This is valid for all controller instances.

The ControllerCycleTime is always rounded up to an integer multiple of Control1.

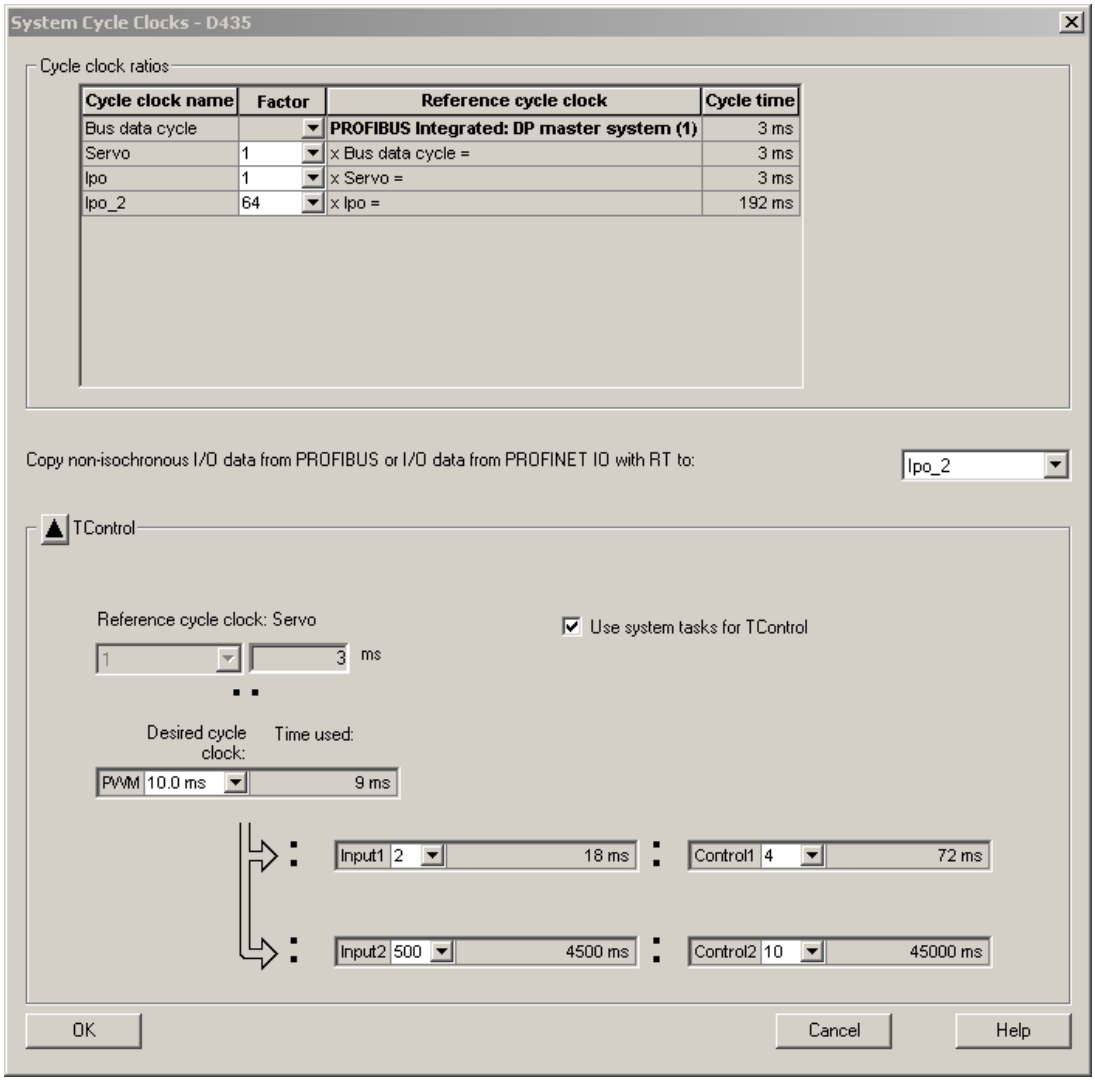


Figure 4-330 System cycle clocks for temperature control

Resulting distribution slices:

Number of distribution slices = ControllerCycleTime/Control1_Time

Number of distribution slices = 1,008 s/72 ms = 14

Thus, the 43 channels are distributed across 14 time slices (three or four channels will always switch at the same time).

Signs of life for temperature channels

Unless **TControl** was activated via the **Use system tasks for TControl** checkbox in the execution system when creating a temperature channel, you will not be notified in automatic mode that this temperature channel *is not being processed*.

Each temperature channel has a sign of life via the **lifeSign** system variable (V4.0 and higher). The sign of life is counted up in each cycle and can be evaluated via the user program.

Nevertheless, there is no checking carried out by the system.

4.4.8 List of abbreviations

4.4.8.1 List of abbreviations

PWM	Pulse width modulation
TO	Technology object

4.5 Communication

Foreword

Foreword

Content

This document is part of the **System and Function Descriptions** documentation package.

Scope of validity

This manual is valid for SIMOTION SCOUT product version V5.4:

- SIMOTION SCOUT V5.4 (engineering system of the SIMOTION product family)
- The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 version V5.x. The described theoretical principles apply to SIMOTION SCOUT and, usually, also to SIMOTION SCOUT TIA.
  Information on configuration in the Engineering Framework Totally Integrated Automation Portal (SIMOTION SCOUT in the TIA Portal) is find in the SIMOTION SCOUT TIA Configuration Manual.

Sections in this manual

This manual describes the communications possibilities for SIMOTION systems.

- **Communications functions and services overview**
  General information about the communications possibilities provided by SIMOTION.
- **PROFIBUS**
  Information about the DPV1 communication, and the setup and programming of the communication between SIMOTION and SIMATIC devices.
- **PROFINET IO**
  Information about configuring PROFINET with SIMOTION

- **Ethernet introduction (TCP/IP and UDP connections)**
  Information about the setup and programming of the Ethernet communication between SIMOTION and SIMATIC devices.
- **Routing - communication across network boundaries**
  General information about routing
- **SIMOTION IT**
  General information about the IT and Web functions provided by SIMOTION.
- **PROFIsafe**
  General information on configuring failsafe controllers.
- **PROFIdrive**
  Description of the PROFIdrive profile.
- **Index**
  Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.5.1 Fundamental safety instructions

4.5.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.5.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.5.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.5.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.5.2 Introduction

4.5.2.1 The communications subject in the SIMOTION documentation

Overview

You can find information on the subject of communication in the individual Manuals, in the Programming Manuals and in this Communication Manual.

Communication manual

This Communication Manual provides, in particular, information that is important for the communication of SIMOTION devices with devices that are not part of the SIMOTION family, especially SIMATIC devices.

This manual contains explanations of the required configuration steps that must be performed on both communication partners in order to obtain an error-free, functioning communication relationship.

Therefore, this manual deals very intensively with the settings and the programming of the SIMATIC S7 stations as communication partners of the SIMOTION devices.

Product manuals and programming manuals

The product manuals deal with the subject of communication from the point of view of the devices themselves, i.e. with respect to the electrical properties of the available interfaces as well as the setting options with the SIMOTION SCOUT engineering system.

You will also find further information in the manuals entitled **Modular Machine Concepts** and **Base Functions**, which are part of the SIMOTION documentation package.

There is no information here how the partner stations are set.

4.5.3 Overview of the communication functions and services

4.5.3.1 Network options

Introduction

As an integral part of "Totally Integrated Automation" (TIA), the SIMOTION and SIMATIC network solutions provide the necessary flexibility and performance characteristic for the communication requirements of your application, irrespective of how simple or complex it is.

Note

This section provides a general description of the communication functions and services included in Siemens' automation technology. This does not necessarily imply that all functions mentioned also are available for SIMOTION. You will find details concerning the functions supported by SIMOTION in chapters 4 - 8.

SIMOTION and SIMATIC networks for all applications

The SIMOTION products support a variety of network options. With these network solutions, you can combine the SIMOTION devices in accordance with the requirements of your application.

For further optimization of the network solutions, SIMOTION products provide integrated communication services and functions to extend the performance capability of the network protocol.

Note

Appropriate protective measures (among other things, IT security, e.g. network segmentation) are to be taken in order to ensure safe operation of the system. You can find more information on Industrial Security on the Internet at:

www.siemens.de/industrialsecurity.

PROFINET

Overview

PROFINET is based on the open Industrial Ethernet standard for industrial automation for company-wide communication and extends the capability for data exchange of your automation components through to the office environment, so that your automation components, even the distributed field devices and drives, can be connected to your local area network (LAN).

Because PROFINET connects all levels of your organization – from the field devices through to the management systems – you can perform the plant-wide engineering using normal IT standards. As for all solutions based on Industrial Ethernet, PROFINET supports electrical, optical and wireless networks.

As PROFINET is based on Industrial Ethernet and not implemented as a derived form of "PROFIBUS for Ethernet", PROFINET can utilize the previously installed Ethernet-compatible devices. Even if PROFINET is not a master/slave system, the PROFINET IO and PROFINET CBA communication services provide the functionality required by automation systems:

- With PROFINET IO, you can connect distributed field devices (e.g. digital or analog signal modules) and drives directly to an Industrial Ethernet subnet.
- PROFINET CBA (Component-Based Automation) supports modular solutions for machine and plant construction. You define your automation system as autonomous components, whereby each component consists of independent, self-contained tasks.

Both communication services provide real-time functionality to make PROFINET a real-time implementation. PROFINET also enables the simultaneous existence of the real-time communication of your automation process and your other IT communication, at the same time in the same network, without the real-time behavior of your automation system being impaired.

The PROFIsafe profile communicates with the fail-safe devices via the PROFINET subnet for further support of fail-safe or "safety-relevant" applications.

Industrial Ethernet

Overview

As Industrial Ethernet provides a communication network for the connection of command levels and cell levels, you can extend the data exchange capability of your automation components into the office environment with Industrial Ethernet.

Industrial Ethernet is based on the standards IEEE 802.3 and IEEE 802.3u for communication between computers and automation systems and enables your system to exchange large data volumes over long distances.

PROFIBUS

Overview

PROFIBUS is based on the standards IEC 61158 and EN 50170 and provides a solution with open field bus for the complete production and process automation. PROFIBUS provides fast, reliable data exchange and integrated diagnostic functions. PROFIBUS supports manufacturer-independent solutions with the largest third-party manufacturer support worldwide. A variety of transmission media can be used for your PROFIBUS subnet: electrical, optical and wireless.

PROFIBUS provides the following communication services:

- PROFIBUS DP (Distributed Peripherals) is a communication protocol that is especially suitable for production automation.
  PROFIBUS DP provides a fast, cyclic and deterministic exchange of process data between a bus DP master and the assigned DP slave devices. PROFIBUS DP supports isochronous communication. The synchronized execution cycles ensure that the data is transmitted at consistently equidistant time intervals.
- PROFIBUS PA (Process Automation) expands PROFIBUS DP to provide intrinsically safe data and power transmission according to the IEC 61158-2 standard.
- PROFIBUS FMS (Fieldbus Message Specification) is for communication on the cell level, where the controllers communicate with one another. Automation systems from different manufacturers can communicate with one another by means of PROFIBUS FMS.
- PROFIBUS FDL (Fieldbus Data Link) has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet.

In addition, PROFIBUS uses profiles to provide communication options for the needs of specific applications, such as PROFIdrive (for the motion control) or PROFIsafe (for fail-safe or "safety-relevant" applications).

MPI (Multi-Point Interface)

Overview

MPIs are integrated interfaces for SIMOTION and SIMATIC products (SIMOTION devices, SIMATIC S7 devices, SIMATIC HMI as well as SIMATIC PC and PG).

MPI provides an interface for PG/OP communication. In addition, MPI provides simple networking capability using the following services: communication via global data (GD), S7 communication and S7 basic communication.

The electric transmission medium for MPI uses the RS 485 standard, which is also used by PROFIBUS.

Point-to-point communication (PtP)

Overview

SIMOTION devices can be programmed so that they exchange data with another controller in the network. Even if the point-to-point communication is not considered as a subnet, the point-to-point connection provides serial transmission (e.g. RS232 or RS485) of data between two stations, e.g. with a SIMATIC controller or even with a third-party device that is capable of communication.

CP modules (e.g. a CP340) or ET200 modules can be used for point-to-point communication to read and write data between two controllers. Point-to-point communication thus represents a powerful and cost-effective alternative to bus solutions, particularly when only a few devices are connected to the SIMOTION device.

Point-to-point communication provides the following capabilities:

- Using standard procedures or loadable drivers to adapt to the protocol of the communication partner
- Using ASCII characters to define a user-specific procedure
- Communication with other types of devices, such as operator panels, printers or card readers

Additional references

You will find additional references concerning point-to-point communication in the descriptions of the CP or ET200 modules.

4.5.3.2 Communications services (or network functions)

Introduction

SIMOTION and SIMATIC devices support a set of specific communication services, which control the data packets that are transmitted via the physical networks. Each communication service defines a set of functions and performance characteristics, e.g. the data to be transferred, the devices to be controlled, the devices to be monitored and the programs to be loaded.

Communication services of the SIMOTION and SIMATIC products

Communication services, also often referred to as network functions, are the software components that utilize the physical hardware of the networks. Software interfaces (e.g. S7 system functions) in the end device (e.g. SIMOTION device, SIMATIC S7 device or PC) provide access to the communication services. However, a software interface does not necessarily have all of the communication functions for the communication service. Such a service can be provided in the respective end system with different software interfaces.

Communicating with SIMATIC

For the communication functions of the SIMATIC controllers, refer to the SIMATIC documentation.

- SIMATIC S7-1500, ET 200MP, ET 200SP communication (https://support.industry.siemens.com/cs/ww/en/view/59192925)
- SIMATIC communication with SIMATIC (https://support.industry.siemens.com/cs/ww/en/view/25074283)

PG/OP communication services

Overview

PG/OP services are the integrated communication functions with which SIMATIC and SIMOTION automation systems communicate with a programming device (e.g. STEP 7) and operator control and monitoring system. All SIMOTION and SIMATIC networks support the PG/OP communication services.

S7 communication services

Overview

S7 communication services provide data exchange using communication system function blocks (SFBs) and communication function blocks (FBs) for configured S7 connections.

All SIMOTION devices and SIMATIC S7 devices have integrated S7 communication services that allow the user program in the controller to initiate the reading or writing of data. These functions are independent of specific networks, allowing you to program S7 communication via any network (MPI, PROFIBUS, PROFINET or Industrial Ethernet).

For transferring data between the controllers, you must configure a connection between both controllers. The integrated communication functions are called up by the SFB/FB in the application. You can transfer up to 64 KB of data between SIMOTION and SIMATIC S7 devices.

You can access data in the controller with your HMI device, programming device (PG), or PC as the S7 communication functions are integrated in the operating system of the SIMOTION devices and SIMATIC S7 devices. This type of peer-to-peer link does not require any additional connection equipment. (However, if you configure a connection to one of these devices, you can access the data via the symbolic names.)

Note

SFBs may not be used with SIMOTION.

S7 basic communication services

Overview

S7 basic communication services provide data exchange using communication system functions (SFCs) for non-configured S7 connections. These SFCs (e.g. X_GET or X_PUT) read or write the data to a SIMATIC controller, so that small data volumes can be transferred via an MPI subnet to another S7 station (S7 controller, HMI or PC).

The SFCs for the S7 basic communication do not communicate with stations in other subnets. You do not need to configure connections for the S7 basic communication. The connections are established when the user program calls the SFC.

Note

You can only use the S7 basic communication services via an MPI connection between SIMATIC S7-300, S7-400 or C7-600 controllers.

"Global data" communication service

Overview

In addition to the other options for the network communication, you can configure a 'global data' communication connection (GD) to provide cyclic data transmission between SIMATIC controllers that are connected to an MPI network. The data exchange runs as part of the normal process image exchange, as the global data communication is integrated in the operating system of the SIMATIC controller.

As the global data communication is a process for transferring data, the receipt of the global data is not acknowledged. A publisher (data source) sends the data to one or several subscriber(s) (data sink) and subscribers receive the data. The publisher does not receive an acknowledgement from the subscribers that they have received the transmitted data.

Note

You can only use the global data communication via an MPI connection between SIMATIC S7-300, S7-400 or C7-600 controllers.

GD communication does not require any special programming or program blocks in your STEP 7 user program. The operating systems of the individual controllers process the global data exchange. Using STEP 7, you configure a global data (GD) table with the source path of the data to be transmitted to the subscribers. This GD table is downloaded with the hardware configuration for both the publisher and the subscribers.

Global data is not available for SIMOTION.

PROFINET communication services

Overview

PROFINET provides the following communication services:

- You can connect I/O devices and drives via a Ethernet physics to the SIMOTION or SIMATIC controller with the communication service PROFINET IO. The user program executed in the controller can process the input and output data of the I/O devices with PROFINET IO. You configure the addressing for PROFINET IO in STEP 7 or SIMOTION SCOUT.
- With PROFINET CBA, you can define your automation system as autonomous subunits or components. These components can be PROFINET IO, PROFIBUS DP or third-party devices or subnets.

If you want to use the PROFINET CBA communication services for a component-based solution, configure the SIMATIC controllers and the I/O devices in individual components in STEP 7. Then configure the communication between the various components with SIMATIC iMAP.

Both PROFINET IO and PROFINET CBA communication services provide the real-time communication required by automation systems.

Note

PROFINET CBA is only available for SIMATIC devices, not for SIMOTION devices.

Industrial Ethernet communication services

Overview

Industrial Ethernet is based on the IEEE 802.3 and IEEE 802.3u standards and connects the automation systems with your business system, so that you also have access to the data in the office.

Industrial Ethernet provides the following communication services:

- The ISO transfer provides services for transmitting data via connections that support error-free data transmission. The ISO transfer is only possible with STEP7.
- TCP/IP allows you to exchange contiguous data blocks between the controllers and computers in PROFINET or Industrial Ethernet networks. With TCP/IP, the controller transmits contiguous data blocks.
- ISO-on-TCP (RFC 1006) supports error-free data transmission. For SIMOTION only when going though SCOUT ONLINE. If the communication is performed from the user program, an RFC must be programmed.
- UDP (User Datagram Protocol) and UDP multi-cast provide simple data transmission without acknowledgment. You can transmit related data blocks from one station to another, such as between a SIMOTION and SIMATIC controller, a PC or a third-party system.
- Information technology (IT) communication allows you to share data using standard Ethernet protocols and services (such as FTP, HTTP and e-mail) via PROFINET or Industrial Ethernet networks.

PROFIBUS communication services

Overview

PROFIBUS provides the following communication services:

- PROFIBUS DP (Distributed Peripherals) supports the transparent communication with the distributed I/O. The SIMOTION/STEP 7 user program accesses the distributed I/O in the same manner as it accesses the I/O on the central rack of the controller (or the PLC). PROFIBUS DP enables the direct communication with the distributed I/O. PROFIBUS DP complies with the EN 61158 and EN 50170 standards.
- PROFIBUS PA (Process Automation) facilitates the direct communication with process automation (PA) instruments. This includes cyclic access to I/O, typically with a PLC master, as well as acyclic access to the potentially large set of device operating parameters, typically with an engineering tool such as Process Device Manager (PDM). PROFIBUS PA complies with the IEC 61158 standard.
- PROFIBUS FMS (Fieldbus Message Specifications) enables the transmission of structured data (FMS variables). PROFIBUS FMS complies with the IEC 61784 standard.
- PROFIBUS FDL (Fieldbus Data Link) has been optimized for the transmission of medium-sized data volumes to support error-free data transmission on the PROFIBUS subnet. PROFIBUS FDL supports the SDA function (Send Data with Acknowledge).

Note

SIMOTION devices only support the PROFIBUS DP communication service.

For fail-safe communication, SIMOTION and SIMATIC devices use the PROFIsafe profile for PROFIBUS DP.

SIMOTION devices use the PROFIdrive profile for communication between SIMOTION devices through to the connected drives.

Additional references

You can find a comparison of the SIMATIC S7 and SIMOTION system functions in the 2_FAQ directory on the Utilities & Applications CD.

Time-of-day synchronization via NTP

Introduction

To ensure that all components throughout the system have the same time of day, they must be synchronized to a time of day and one component must be the time generator for all the others. One possibility is to synchronize the time of day via an NTP server.

For NTP, the device sends time of day queries in regular time intervals to the NTP server in the subnet. A router must be used if the NTP server is located outside the subnet. The most accurate time of day is determined based on the responses and used to synchronize the station time of day. NTP has the advantage that the synchronization is also performed over subnet limits.

NTP time-of-day synchronization properties

- The NTP server is configured at the PN/IE interface of the controller. The PN interface via which communication with the time-of-day server actually takes place is determined on the basis of the IP address configured in the TCP/IP stack.
  - X7 with C240
  - X130 with D4x5
  - X127 with D410
  - X1 with P320-4
- Up to four servers can be configured as time generator.
- NTP does not support any automatic adjustment between summer and winter time.
- Time zones cannot be set. UTC (Universal Time Coordinated) is always transferred. It is identical with GMT (Greenwich Mean Time).

Configuring NTP in SIMOTION SCOUT

Introduction

The NTP time-of-day synchronization of a SIMOTION CPU can be configured in HW Config.

Procedure

How to configure the NTP time-of-day synchronization:

1. Click the PN/IE-NET interface (for D4x5 X130) in the HW Config for the SIMOTION CPU.
2. Select the time-of-day synchronization tab in the displayed dialog.
3. Activate the "Activate time-of-day synchronization in NTP" checkbox.

4. Enter the IP addresses for as many as four NTP servers. A router must be used when the IP address lies outside the subnet of the SIMOTION controller.
5. Enter the refresh interval.

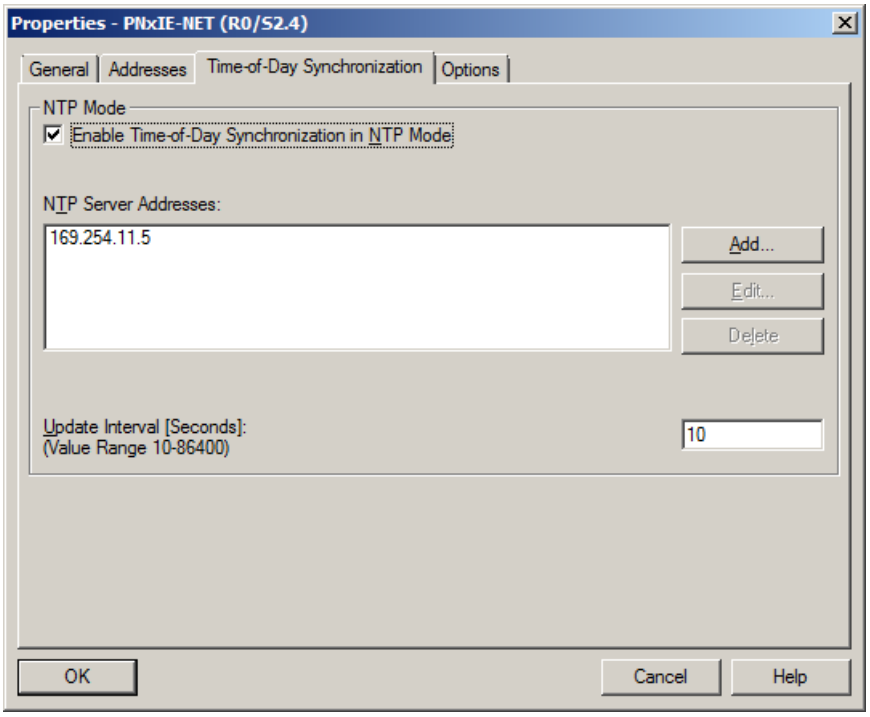


Figure 4-331 Time-of-day synchronization via NTP in HW Config

4.5.3.3 Additional services for the exchange of information

In addition to supporting the standard communication networks, SIMOTION and SIMATIC also provide additional means for sharing information via networks.

Sharing data with other applications via OPC (OLE for Process Control)

With OPC, Windows applications can access process data so that it is easy for devices and applications from different vendors to be combined with each other. OPC not only provides an open, manufacturer-independent interface, but also an easy-to-use client/server configuration for the standardized data exchange between applications (e.g. HMI or office applications) that do not require a specific network or protocol.

The OPC server provides interfaces for connecting the OPC client applications. You configure the client application for access to data sources, e.g. addresses in the memory of a PLC. Because several different OPC clients can access the same OPC server at the same time, the same data sources can be used for any OPC-compliant application.

In addition to OPC servers, SIMATIC NET also provides applications for configuring and testing OPC connections: Advanced PC Configuration (APC) and OPC Scout (used to test and commission an OPC application or OPC server). You use these tools to connect SIMOTION and SIMATIC S7 products to other OPC-compliant applications.

The SIMATIC NET OPC servers support the following communication services:

- PROFINET IO (by means of PROFINET or Industrial Ethernet subnet)
- PROFINET CBA (by means of PROFINET or Industrial Ethernet subnet)
- TCP/IP (by means of PROFINET or Industrial Ethernet subnet)
- PROFIBUS DP (by means of PROFIBUS subnet)
- PROFIBUS FMS (by means of PROFIBUS subnet)
- S7 communication
- S5compatible communication

Using information technology (IT) for sharing data in an office environment

SIMOTION and SIMATIC use standard IT tools (such as e-mail, HTTP Web server, FTP and SNMP) with PROFINET and Industrial Ethernet networks to expand the data-sharing capabilities into the office environment.

For SIMOTION devices, the corresponding functions are made available through SIMOTION IT, see SIMOTION IT Ethernet-based HMI and Diagnostic Functions.

4.5.4 PROFIBUS DP

4.5.4.1 PROFIBUS DP communication

Overview of PROFIBUS DP communication

Description

PROFIBUS DP (Decentralized Peripherals) is designed for fast data exchange at the field level. The communication is performed between a class 1 PROFIBUS DP master (e.g. a SIMOTION controller) and PROFIBUS DP slaves (e.g. a SINAMICS S120 drive). The data exchange with decentralized devices is mainly performed cyclically (DP V0 communication). In this case, the central controller (SIMOTION controller) reads the input information cyclically from the slaves and writes the output information cyclically to the slaves. Moreover, diagnostics functions are made available through the cyclic services. The following figure shows the data protocol on PROFIBUS DP.

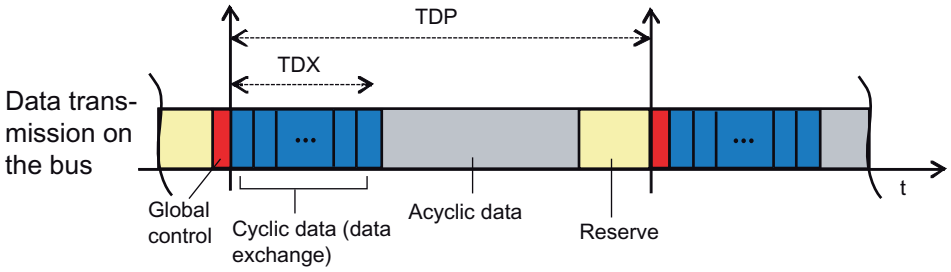


Figure 4-332 Data protocol on PROFIBUS DP

4.5.4.2 Cyclical data exchange between a SIMOTION and SIMATIC controller.

Description of cyclical SIMOTION and SIMATIC controller data exchange

The following section describes the options for having a SIMOTION and a SIMATIC controller communicating with one another via PROFIBUS DP.

There are various possibilities:

- A SIMOTION controller is connected to a PROFIBUS DP master system of a SIMATIC controller as a PROFIBUS DP slave.
- A SIMATIC controller is connected to a PROFIBUS DP master system of a SIMOTION controller as a PROFIBUS DP slave.
- A master-master communication is used between SIMOTION and SIMATIC controls.

Linking as DP slave

For linking as a DP slave, there are two options:

- The controller is connected to the DP master as a "standard" DP slave by means of a GSD file. This is essential if the two controllers are in different projects.
- The controller is connected to the DP master as a so-called I slave (intelligent DP slave). For this purpose, the two controls must be located in one project.

The two options are supported by SIMOTION and SIMATIC controllers.

Note

You will find further information on cyclic data exchange with controllers in the SIMATIC FAQs.

- Linking a S7-400 CPU to a non-Siemens master as a DP slave (https://support.industry.siemens.com/cs/ww/en/view/13091565)
- DP linking CPU 315-2DP (slave) and S7-400 (master) (https://support.industry.siemens.com/cs/ww/en/view/6518822)

Linking a SIMOTION controller to a SIMATIC controller as a PROFIBUS DP slave

Linking by means of a GSD file

Procedure

The GSD files for the various SIMOTION hardware platforms are automatically also installed by the SIMOTION SCOUT installation.

If there is no SIMOTION SCOUT on the engineering PC, the GSD files must first be installed in STEP7 HW Config. You will find the corresponding GSD files on the SIMOTION SCOUT DVD "Add-on" in the respective device directory under Firmware and Version.

Table 4-219 GSD file

Device	Name of the GSD file
SIMOTION C	Si0480aa.gsd
SIMOTION D	Si0280ab.gsd (This file can be used for all SIMOTION D)
SIMOTION P	Si0380fa.gsd

After the GSD files have been installed by selecting **Options > Install GSD file...** in the menu, the corresponding SIMOTION devices will appear in the Hardware Catalog under **PROFIBUS DP >**

Additional FIELD DEVICES > PLC > SIMOTION. These can be inserted into a DP master system from there.

Note

On SIMOTION controllers that are linked to a higher-level controller via a GSD file (separate projects), access is not possible with SIMOTION SCOUT and the access point S7ONLINE via a routed connection.

The alternative access point **DEVICE** in the SIMOTION SCOUT must be used for this! This also has the advantage of being able to go online simultaneously with the two projects to SIMATIC (S7ONLINE) and SIMOTION controller (DEVICE). However, it is not possible to go online to the SINAMICS_Integrated of a SIMOTION D controller if the link with the SIMOTION has already been routed. Only one network transition can be configured (e.g. PC/PG <Ethernet> S7-CPU <PROFIBUS> SIMOTION, but not <PROFIBUS_Integrated> SINAMICS_Integrated in addition).

Note

It is possible to access SINAMICS drives that are linked to a controller via a GSD file with SIMOTION SCOUT (or STARTER) via a routed connection with the exception of the SINAMICS_Integrated drive.

To do this, it is possible to set a network transition point using the S7 subnet ID with the setting **Target device > Online access ...**. Through a network node it is also possible to route to drives that have been inserted as single drives.

Linking using an I slave

Requirement

- STEP 7 and SIMOTION SCOUT must have been installed on the engineering PC.
- The SIMATIC and SIMOTION controllers must be in the same project.

If these requirements are fulfilled, the SIMOTION controller can also be connected to the PROFIBUS DP master system of the higher-level SIMATIC as an I slave.

Procedure

It is recommended that the SIMOTION station is first completely configured as DP slave before it is placed on the DP line of the SIMATIC CPU as an I slave.

Below you will find a description of the procedure for a SIMOTION C. The procedure is identical apart from the selection of the SIMOTION platform.

1. Configuring a station as a DP slave, e.g. SIMOTION C2xx
   Double-clicking on the desired PROFIBUS interface (e.g. DP2/MPI) in HW Config opens its properties. Select the **DP slave** option from the **Operating Mode** tab.
2. Configuring the local I/O addresses
   You can set the local I/O addresses and the diagnostics address on the **Configuration** tab.
3. Switch to the configured SIMATIC station that is to be DP master for the SIMOTION.

4. Creating an I slave
   Drag the station type "C2xx/P3xx/D4xx/D4xx-2-I-Slave" from the **Hardware catalog** window, "Preconfigured stations" folder, and drop it on the DP master system of the SIMATIC controller.
5. Linking an I-slave
   Double-click the I slave proxy to open the properties window. On the **Link** tab, assign the station that is to represent the intelligent DP slave (I slave). This displays all the stations that are already available in the project and that are potential link partners.

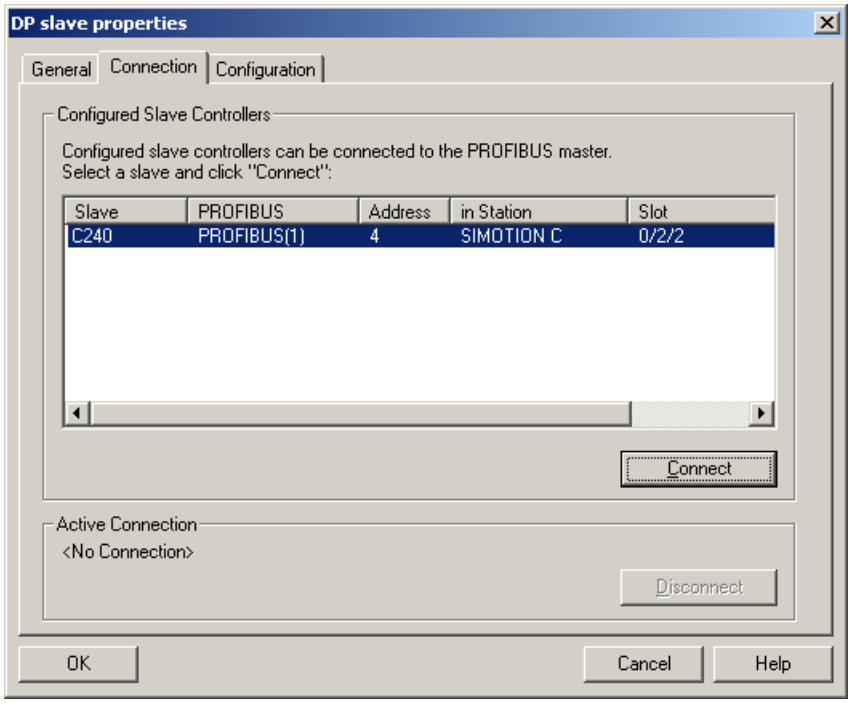


Figure 4-333 I-slave properties

6. Select the appropriate SIMOTION and click **Connect**. The configured SIMOTION station is now connected as intelligent DP slave to the SIMATIC.

7. Select the **Configuration** tab and assign the addresses:

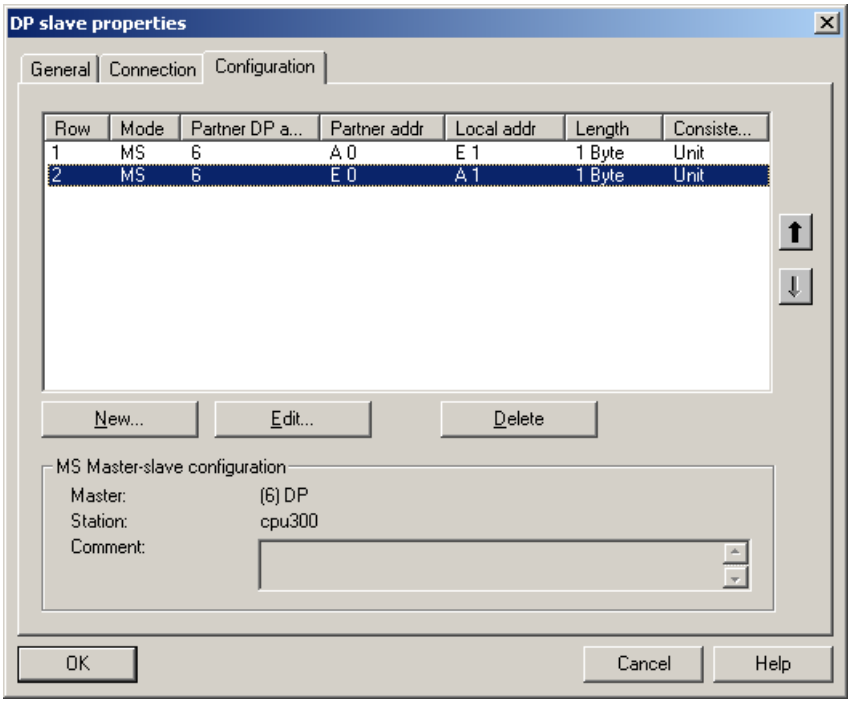


Figure 4-334 Properties - configuration

– For the data exchange with the DP master via I/O areas, select the **MS** (Master-Slave) mode

Note

For direct data exchange with a DP slave (slave-to-slave data exchange broadcast) or DP master (master-to-master data exchange broadcast), it would be necessary to select DX mode (Direct Data Exchange) mode here.

8. Confirm the settings by clicking **OK**.

Configuration of the SIMOTION station as intelligent DP slave on the SIMATIC station is now complete and data can be exchanged via the specified I/O addresses.

Linking a SIMATIC controller to a SIMOTION controller as a PROFIBUS DP slave

Linking by means of a GSD file

Procedure

The GSD files for the various SIMATIC stations must first be installed in STEP 7 HW Config.

Note

You will find the corresponding GSD files in Product Support: PROFIBUS GSD files for SIMATIC.

After the GSD files have been installed by selecting **Options > Install GSD file...** in the menu, the corresponding SIMATIC devices will appear in the Hardware Catalog under **PROFIBUS DP > Additional FIELD DEVICES > PLC > SIMATIC**. These can be inserted into a DP master system from there.

Note

SIMATIC controllers that have been connected to a higher-level controller by means of a GSD file, cannot be accessed with STEP 7 via a routed connection.

See also

PROFIBUS GSD files: SIMATIC (https://support.industry.siemens.com/cs/ww/en/view/113652)

Linking using an I slave

Requirements

- STEP 7 and SIMOTION SCOUT must have been installed on the engineering PC.
- The SIMATIC and SIMOTION controllers must be in the same project.

If these requirements are fulfilled, the SIMATIC controller can also be connected to the PROFIBUS DP master system of the SIMOTION controller as an I slave.

Procedure

It is recommended that the SIMATIC station is first completely configured as a DP slave before it is placed on the DP line of the SIMOTION as an I slave.

Below you will find a description of the procedure for a CPU 315-2 D. The procedure is identical apart from the selection of the CPU types, also for an S7-400.

1. Configure a station, for example, with the CPU 315-2 DP, as DP slave. Double-click on line 2.1 (interface) in the configuration table and select the DP slave option in the **Operating mode** tab.
2. You can set the local I/O addresses and the diagnostics address in the **Configuration** tab.
3. Switch to the configured SIMOTION station that is to be DP master for the SIMATIC.
4. Drag the appropriate station type, CPU 31x or CPU 41x, from the **Hardware Catalog** window (folder of already configured stations) and drop it on the symbol for the DP master system of the SIMOTION station.

5. Double-click the I slave proxy to open the properties window. On the **Link** tab, assign the station that is to represent the intelligent DP slave (I slave). This displays all the stations that are already available in the project and that are potential link partners.

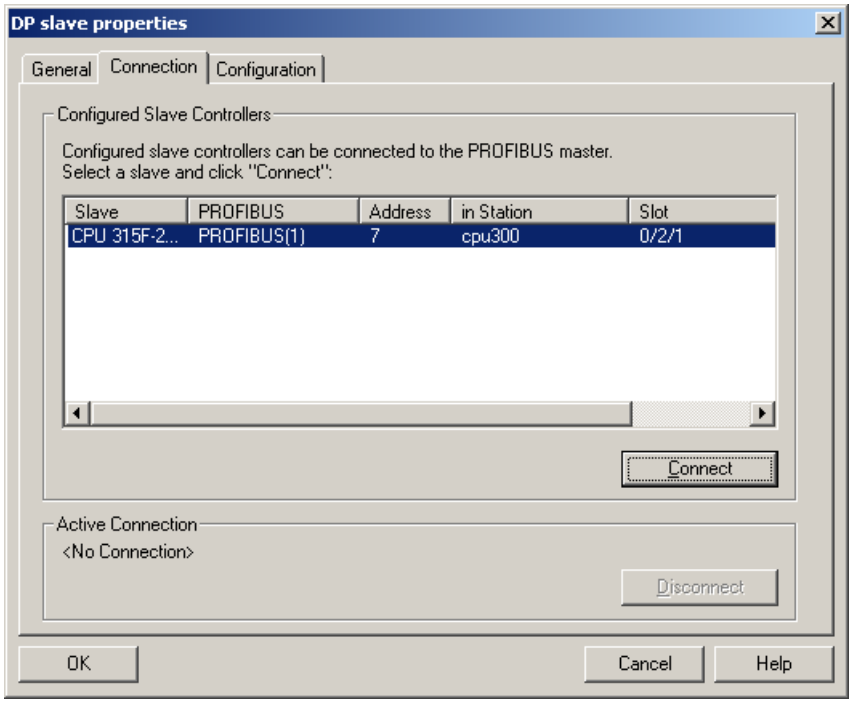


Figure 4-335 Properties - link

6. Select the appropriate S7 station and click **Connect**. The configured S7 station is now connected as intelligent DP slave to the SIMOTION.

7. Select the **Configuration** tab and assign the addresses:

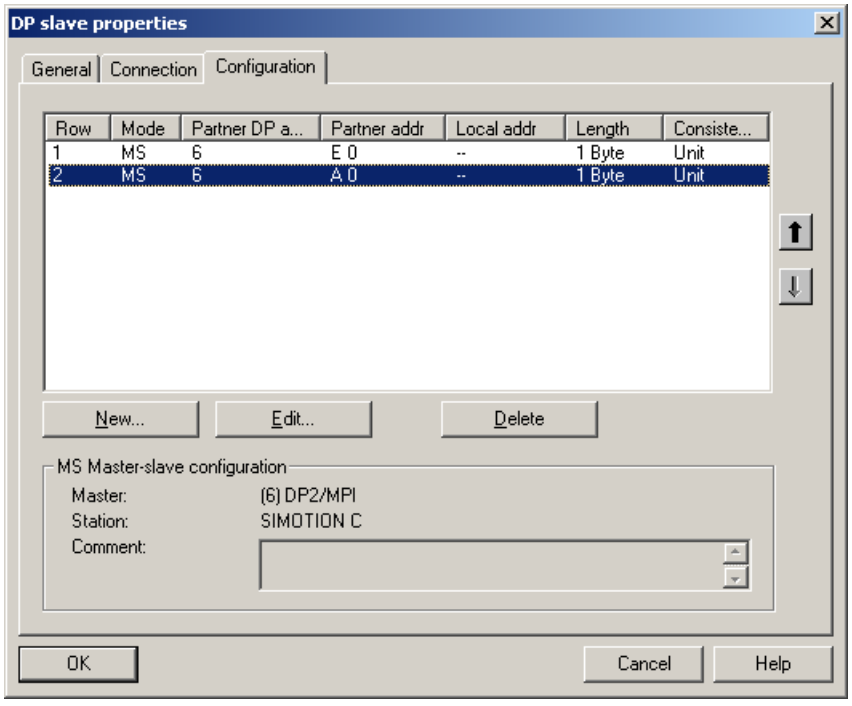


Figure 4-336 Configuration - address selection

– For the data exchange with the DP master via I/O areas, select the **MS** (Master-Slave) mode

Note

For direct data exchange with a DP slave (slave-to-slave data exchange broadcast) or DP master (master-to-master data exchange broadcast), it would be necessary to select DX mode (Direct Data Exchange) mode here.

8. Confirm the settings by clicking **OK**.

The configuration of the SIMATIC station as intelligent DP slave on the SIMOTION station is now completed and data can be exchanged via the specified I/O addresses.

CPU-CPU communication using the S7 basic communication via PROFIBUS

Introduction

CPU-CPU communication

CPU-CPU communication using the S7 basic communication between a SIMATIC and a SIMOTION controller, is set up using the SFC65 (XSEND) and SFC66 (XRECEIVE) system functions on the SIMATIC side and the _Xsend() and _Xreceive() system functions on the SIMOTION side. This application communication is effected via PROFIBUS, or MPI.

Table 4-220 CPU-CPU communication

Protocol	SIMATIC device	Function	SIMOTION device	Function
PROFIBUS	S7-300 CPU S7-400 CPU	SFC65 (XSEND) SFC66 (XRCV)	SIMOTION C SIMOTION D SIMOTION P	_Xsend _Xreceive

PROFIBUS addresses are assigned in HW Config. This must also be passed as a parameter when the system function _Xsend() is called. That is, the parameters that are important for communication are determined by the user and passed in the function call. It is not necessary to configure the communication link in NetPro.

Note

For further information, refer to the online help of each system function.

You will find additional information in the FAQ for SIMATIC communication:

- CPU-CPU communication with SIMATIC controllers (https://support.industry.siemens.com/cs/ww/en/view/20982954)
- S7 basic communication (https://support.industry.siemens.com/cs/ww/en/view/28866132)

SIMOTION functions

The SIMOTION functions for a PROFIBUS connection are explained in more detail below.

System functions Xsend() and Xreceive()

Table 4-221 Xsend()

```
RetVal_PB_Senden:=
    _xsend(
          communicationMode := PB_Senden_CommunicationMode,
          address := PB_Senden_Address,
          messageID := PB_Senden_MessageID,
          nextCommand := PB_Sender_NextCommand,
          commandID := PB_Senden_CommandID,
          data := PB_Sende_Daten,
          datalenght := PB_Sende_Daten_Laenge
          );
```

Example of the call of the SIMOTION function _xsend

If the SIMATIC S7 station and the SIMOTION device communicate via PROFIBUS, the _xsend function is called on the SIMOTION side for sending purposes.

Note

Application transmission and reception via S7 basic communication (_Xreceive() or _Xsend()) is only possible on a PROFIBUS interface of the SIMOTION controller if the **Programming, status/ force, or other PG functions and non-configured communication connections possible** option is set.

The "communicationmode" parameter informs the called function of what is to happen to the connection after the successful data transfer. The function data type can assign the ABORT_CONNECTION or HOLD_CONNECTION values. If ABORT_CONNECTION is assigned to the parameter, the connection will be removed after the data transfer. The HOLD_CONNECTION value is used to parameterize the function so that the connection will be retained after a successful data transfer.

The address parameter contains a structure of the StructXsendDestAddr data type, which also consists of various parameters. This structure contains all the information about the communication partner address of the SIMOTION device.

Parameter structure "StructXsendDestAddr

The individual parameters of the structure are listed and explained in the following.

The deviceid parameter is used for the respective SIMOTION hardware. The physical connection point is specified with the parameter. For example, the value 1 is entered for interface X8 for a SIMOTION C2xx. The value 2 is entered for interface X9. If a SIMATIC S7 station is connected to X101 of a SIMOTION P, the value 1 is assigned in the deviceid parameter. The value 2 is written in the deviceid parameter for the X102 interface. For the SIMOTION D, the value 1 is entered for the X126 interface and the value 2 for the X136 interface in the deviceid parameter.

Because no subnet mask is specified for communication via MPI or PROFIBUS, a value of 0 is pre-assigned to the remotesubnetidlength parameter. Consequently, the assignment of the remotesubnetid parameter is irrelevant.

The value 1 is set in the remotestaddrlength parameter for the MPI or PROFIBUS communication.

The nextstaddrlength parameter specifies the length of the router address. As a router is not used for the MPI or PROFIBUS communication between the SIMATIC S7 station and the SIMOTION device, the value 0 is assigned for this parameter. Consequently, the nextstaddr parameter is also irrelevant (see below).

The following remotesubnetid parameter identifies the subnet mask and has, as already mentioned above, no significance for the communication via MPI or PROFIBUS.

The remotestaddr parameter specifies the actual destination address. The parameter is an array. However, only the first index is used for the MPI or PROFIBUS communication. The other five indices have no significance.

The nextstaddr parameter is used to specify the router address. The same applies for this parameter as for the remotesubnetid parameter. Its assignment is also irrelevant for the communication via MPI or PROFIBUS.

The messageid parameter is assigned by the user for the identification of the SIMOTION on the receive side. The value entered enables an assignment on the SIMATIC S7 station via the REQ_ID parameter. The value can be fetched there from the messageid parameter.

The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command.

When the function is called, a system-wide unique number is assigned in the commandid parameter to allow tracking of the command status.

The send data is specified with the data variable when the function is called.

The datalength parameter specifies the length of the data to be transferred from the send area.

The return value of the _xsend function to the user program is of data type DINT. The various return values indicate any problems that occurred during the execution of the function. There is also a confirmation when the data has been successfully sent.

Table 4-222 Xreceive()

```
RetVal_PB_Empfangen:=
    _xreceive(
          messageID := PB_Empfangen_MessageID,
          nextCommand := PB_Empfangen_NextCommand,
          commandID := PB_Empfangen_CommandID
          );
```

Call example of the SIMOTION _xreceive function

The example shows the use of the _xreceive function. The function is used when data from a SIMATIC S7 station is to be received via PROFIBUS.

The messageid parameter is transferred to the _xreceive function for the identification of the S7 station from which the data is to be received. The entered value is that what was assigned on the S7 page in the REQ_ID parameter of the corresponding _xsend system function.

The behavior of this function with respect to the advance when called is parameterized with the nextcommand parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediately and with the second value, after completion of the command.

When the function is called, a system-wide unique number is assigned in the commandid parameter to allow tracking of the command status.

The structure returned from the function to the user program contains the functionresult, datalength and data parameters. The receive status can be queried via the functionresult parameter. The datalength parameter returns the number of received useful data bytes after a successful call of the _xreceive function. The received useful data can be accessed via the data parameter.

4.5.5 PROFINET IO

4.5.5.1 PROFINET IO overview

PROFINET IO

In machine construction, there is a clear trend toward distributed machine concepts and mechatronic solutions. This increases the demands on the drive networking. A large number of drives and shorter cycle times as well as the use of IT mechanisms are increasingly gaining in importance.

The two successful solutions, PROFIBUS DP and Ethernet, are combined under PROFINET IO. PROFINET IO is based on many years of experience with the successful PROFIBUS DP and combines the normal user operations with the simultaneous use of innovative Ethernet technology concepts. This ensures the smooth migration of PROFIBUS DP into the PROFINET IO world.

PROFIBUS DP is a bus system where only one node can have "send" access to the bus at any one time (half-duplex operation). PROFINET IO uses the switching technology which is also found with Ethernet. This involves separating all the network segments, thereby enabling sending and receiving to be performed on all lines at the same time (full-duplex operation). In this way, the network can be used much more efficiently through the simultaneous data transfer of several nodes. The bandwidth has also been increased to 100 Mbps.

Note

Detailed descriptions on the subject of PROFINET can be found in the *SIMATIC PROFINET System Description* System Manual.

Application model

During the development of PROFINET IO, special emphasis was placed on the protection of investment for users and device manufacturers. The application model is retained for the migration to PROFINET IO. Compared with PROFIBUS DP, the process data view remains unchanged for:

- I/O data (access to the I/O data via logical addresses)
- Data records (storage of parameters and data) and
- Connection to a diagnostic system (reporting of diagnostic events, diagnostics buffer)

This means that the familiar view for access to the process data is used in the user program. Existing programming know-how can continue to be used. This also applies to device profiles, such as PROFIdrive, which is also available with PROFINET IO.

The engineering view also has a familiar "look and feel". The engineering of the distributed I/O is performed in the same way and with the same tools, as already used for PROFIBUS.

PROFINET IO system

A PROFINET IO system consists of an IO controller and the IO devices assigned to it.

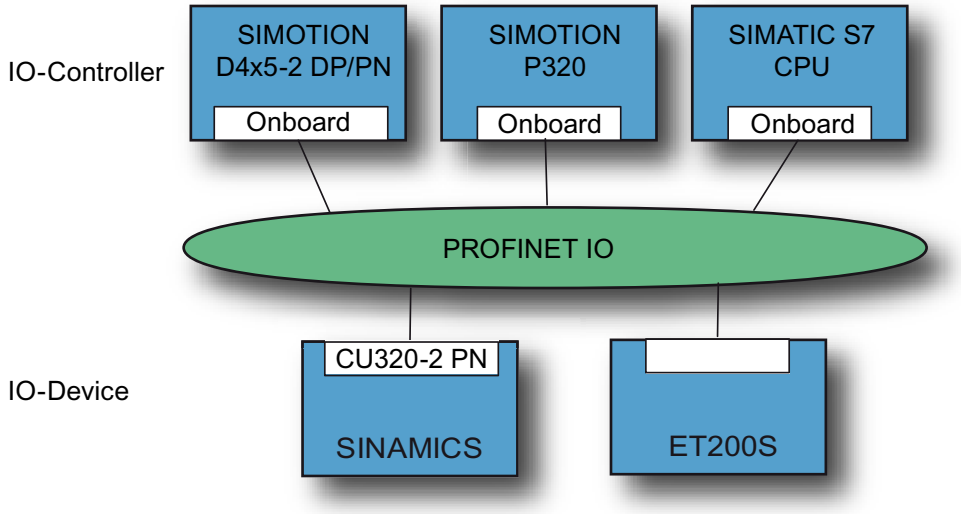


Figure 4-337 PROFINET overview IO Controller - IO Device

IO controller

The PROFINET IO controller has the same functions as the PROFIBUS DP master. The IO controller of, for example, a D4x5-2 DP/PN with onboard PROFINET interface exchanges data cyclically with the I/O devices assigned to it (PROFINET IO devices), such as the SINAMICS S120.

IO device

Distributed field devices such as I/O components (e g. ET200) or drives (e.g. SINAMICS S120 with CU320-2 PN) are referred to as IO devices. The function is comparable to a PROFIBUS DP slave.

See also

Creating an IO device (Page 2213)

PROFINET interface

Overview

PROFINET IO is based on Switched Ethernet full-duplex operation and a transmission bandwidth of 100 Mbit/s.

PROFINET devices have one or more PROFINET interfaces (Ethernet controller/interface). The PROFINET interfaces have one or more ports (physical connections).

PROFINET devices that have interfaces with several ports also have an integrated switch. PROFINET devices with two ports at one interface allow you to establish a system with a linear or ring topology. PROFINET devices with three and more ports at one interface are also suitable for establishing tree topologies.

Transmission medium

You can set the transmission medium in STEP 7 for the ports of a PN interface. **Full duplex with 100 Mbps** is the default setting at all ports. When ports are interconnected, the partner port must use the same transmission medium. **Automatic settings** is recommended as the setting for ports because this will ensure harmonized settings between ports and partner ports.

RT classes

RT classes for PROFINET IO

Description

PROFINET is based on the Ethernet standard. This means that all standard Ethernet-based protocols (e.g. HTTP, FTP, TCP, UDP, IP, etc.) can be transferred via the PROFINET network.

In addition to the protocols associated with the types of office-based applications known throughout the world, PROFINET IO offers two protocols (transmission modes) adapted to the requirements of the automation sector. These are PROFINET IO with RT and PROFINET IO with IRT.

Both these transmission modes are optimized for cyclic IO communication within a network involving small amounts of data.

RT

PROFINET RT uses the option of prioritizing Ethernet telegrams (as described in the Ethernet standard). For more detailed information, see PROFINET IO with RT (Page 2116).

IRT

PROFINET IRT uses a time slot procedure (bandwidth reservation); i.e. 2 time slots occur. The IRT telegrams are transmitted in the first time slot; the standard Ethernet and RT telegrams in the second. By reserving the transmission bandwidth, the transmission of the IRT data is guaranteed for all load/overload situations. For all participating devices to know when the time slot begins, all the participating devices must be synchronized with PROFINET IRT.

In addition to the bandwidth reservation, a schedule for the cyclic telegrams is developed with consideration given to the topology. This makes it possible for the engineering system to determine the required bandwidth for each individual cable.

In addition to the synchronization of the transmission network with PROFINET IRT, the application can also be synchronized in the devices with IRT High Performance (isochronous application; e.g. position controller in SIMOTION or ServoSynchronousTask). This is an essential requirement for closing control loops across the network and isochronous switching of inputs and outputs in the network.

Note

Only PROFINET IRT High Performance is used for SIMOTION. Whenever PROFINET IRT is referred to in the rest of this document, this means IRT High Performance.

For more detailed information, see PROFINET IO with IRT (High Performance) (Page 2118).

Comparing RT and IRT

Table 4-223 The major differences between RT and IRT

Property	RT	IRT (High Performance)
Real-time class	Real-time class 1	Real-time class 3
Transfer mode	Prioritization of cyclic RT telegrams using the so-called VLAN tag.	Bandwidth reservation optimized by the engineering system on the basis of topology information
Determinism	No Send and receive time points of the cyclical RT data are not precise; they can be delayed by standard Ethernet telegrams.	Yes Transmission and receiving times for cyclic IRT data are precisely defined and guaranteed for all kinds of topologies.
Isochronous application	Not supported	Supported
Hardware support using special Ethernet controller	No	Special hardware necessary (e.g. SCALANCE X200 IRT)

Send clock and update time

Description

With PROFINET the send clock and update time are distinguished. The send clock is the basic cycle clock for cyclic communication. The update time indicates the cycle in which a device is supplied with data.

Send clock

This is the period between two successive intervals for IRT or RT communication. The send clock is the shortest possible transmit interval for exchanging data. The send clock therefore corresponds to the shortest possible update time. During this time, IRT data and non-IRT data (RT, TCP/IP) is transmitted. All devices within a sync domain work with the same send clock.

Update time

The update time can be configured separately for each IO device and determines the interval at which data is sent from the IO controller to the IO device (outputs) as well as from the IO device to the IO controller (inputs). The calculated/configured update times are always a multiple (2^n) of the send clock.

Relationship between the update time and send clock

The calculated update times are multiples (1, 2, 4, 8, ..., 512) of the send clock. The minimum possible update time thus depends on the minimum send clock for the IO controller that can be set and the efficiency of the IO controller and the IO device.

Adjustable send clocks and update times

Description

The table below describes the send clocks which can be set for SIMOTION devices with PROFINET IO, as well as the settable update times which are dependent on them and can be set for IRT and RT. Update times are obtained by multiplying the factor and the send clock. The adjustable send clocks are divided into two ranges: the "even" range and the "odd" range.

Note

The following versions generally relate to the use of the position control cycle clock with servo or IPO task, unless otherwise specified. When using the Servo_fast with Servo_fast or IPO_fast Task, other restrictions apply under certain circumstances. If Servo_fast and IPO_fast are used, then the PROFINET must be operated isochronously.

Table 4-224 Adjustable send clocks and update times when using servo or Servo_fast

Send clock		**Factors (update time = factor * send clock)**	
		RT	**IRT**
"Even" range	125 µs *Note 4)*	IO devices cannot be used with RT	1 *Note 2)*
	250, 500, 1,000 µs	1, 2, 4, 8, 16, 32, 64, 128, 256, 512	
	2,000 µs	1, 2, 4, 8, 16, 32, 64, 128, 256	
	4,000 µs	1, 2, 4, 8, 16, 32, 64, 128	
"Odd" range	375, 625, 750, 875, 1,125, 1,250 µs ... 3,875 µs (increment 125 µs)	Cannot be used in front of SIMOTION V4.4 IO devices with RT *Note 1*)	1
Note 1)	**Mixed operation RT / IRT with SIMOTION < V4.4** Send cycles from the "odd" range can be used only if there is no IO device in the IO systems involved in the sync domain. As soon as there are IO devices with RT in a sync domain, it is only possible to set send clocks from the "even" range.		

Send clock		Factors (update time = factor * send clock)	
		RT	IRT
Note 2)	**Settable factors and isochronous operation (isochronous application)** Some IO devices with IRT support the factors 2, 4, 8, and 16 in addition to factor 1. Where IO devices (e.g. ET200S IM151-3 PN HS, SINAMICS S120) are operated isochronously, it is usually only possible to set a factor of 1. In these cases, the mode for the update time must always be set to **Fixed factor** to ensure STEP 7 does not automatically adapt the update time. The update time than always matches the send clock.		
Note 3)	**Mixed operation RT / IRT as of SIMOTION V4.4** As of SIMOTION V4.4, send cycles from the "odd" range can be used only if there is no IO device with RT in the PROFINET IO systems involved in the sync domain. However, this mixed operation is only possible with SIMOTION D4x5-2 devices on the onboard PROFINET interface. The same factors are available as from the "even" range. The factors that can actually be set depends specifically on the set send clock. **Restriction of Shared Device with RT / IRT as of SIMOTION V4.4:** If a Shared Device is simultaneously used with PROFINET IO with RT and PROFINET IO with IRT, PROFINET IO with IRT can only be operated with send clocks from the "even" range.		
Note 4)	The send clock can only be used with D455-2 DP/PN on the onboard PROFINET interface together with Servo_fast as of SIMOTION V4.4. If this send clock is used, only the first port of the onboard PROFINET interface will be active, all other ports are deactivated and cannot be used.		

If there is no sync master configured in a PROFINET IO system (no PROFINET IRT), the send cycle clock for the respective PROFINET IO system can be set individually on the relevant IO controller. You can do this in the properties **<PROFINET Interface>** on the **PROFINET** tab under **Send cycle** or in the properties **PROFINET IO System** on the **Update time** tab. There is a default setting for the send clock of 1 ms. On the **IO cycle** tab, the down-scaling for the update time can be set via the mode **fixed factor** or **fixed update time** and the factor.

As soon as a sync master is configured in the PROFINET IO system, the send clock is defined under the sync domain properties. The IO controllers assigned to the sync domain take over this value. Update times can be set independently for each IO device.

Possible modes for setting the update time

- **Fixed factor**
  Fixed send clock down-scaling for the update time
- **Fixed update time**
  Update time is set
- **Automatic**
  STEP 7 automatically adjusts the down-scaling if the factor selected is too low

Note

It is recommended that you work with the **Fixed factor** setting.

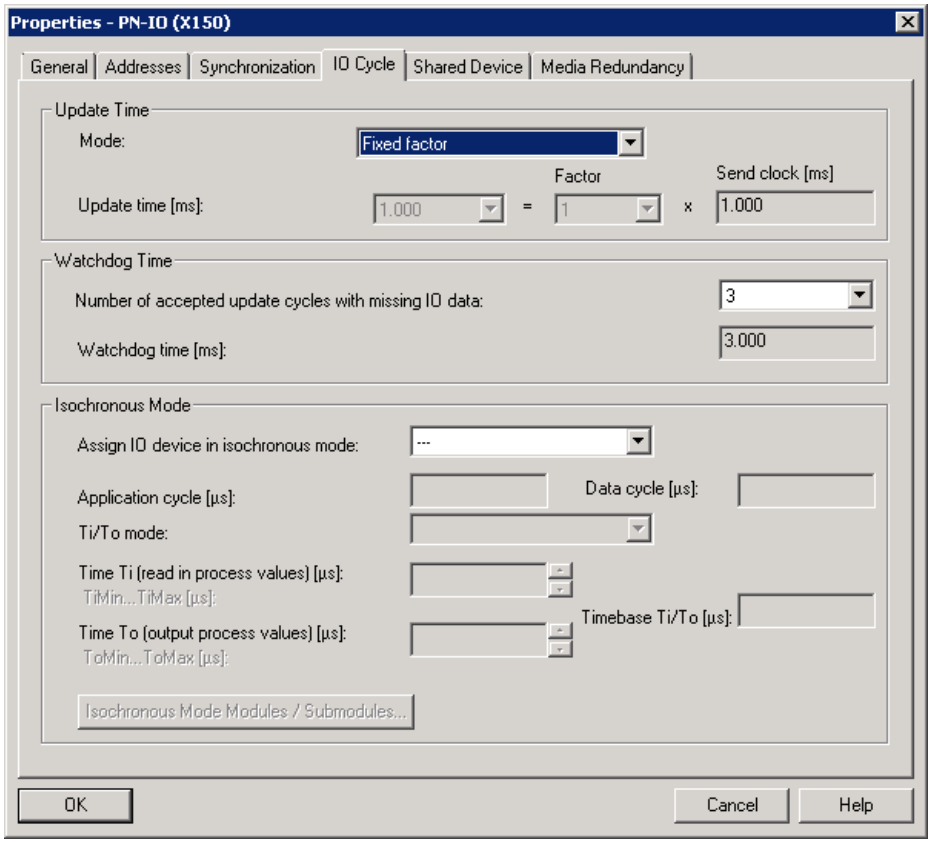


Figure 4-338 The influence of isochronous mode on setting the update time

PROFINET IO with RT

PROFINET IO with RT is the optimal solution for the integration of I/O systems without particular requirements in terms of performance and isochronous mode. This is a solution that uses standard Ethernet and commercially available industrial switches as infrastructure components. A special hardware support is not required.

Not isochronous

Although standard Ethernet and PROFINET IO with RT do not offer any synchronization mechanisms for devices, this does not mean that such arrangements are impossible. However, it does mean that isochronous data transmission is impossible, and there is no isochronous application for motion control as a result.

Data exchange

Communication via PROFINET IO with RT and IRT is based on the Ethernet frame and the MAC address. This means that cross-network communication via a router using RT and IRT is not possible. PROFINET IO message frames have priority over IT message frames in accordance with IEEE802.1Q. This ensures the availability of the real-time properties required in automation applications (e.g. for standard IOs).

Update time

The adjustable update time is in the range of 0.25 - 512 ms. The selected update time depends on the process requirements, the number of devices, and the amount of IO data. Given the improved performance offered by PROFINET compared to field buses, the bus cycle is generally no longer the variable which determines the system cycle.

Device replacement without removable data storage medium

Activate the "Support device replacement without removable data storage medium" checkbox if you want to use the module replacement (the devices connected to the IO controller) without the PG or removable data storage medium.

The IO device replacement is no longer assigned a device name from the removable data storage medium or the PG, but from the IO controller. The IO controller uses the configured topology and the neighboring relationships determined by the IO devices for this purpose.

Note

Device replacement without removable data storage medium

In order that the device replacement without removable data storage medium functions, the plant topology must be configured correctly. Use the topology editor for this purpose.

PROFINET IO with IRT - Overview

Overview

PROFINET IO with IRT satisfies communication requirements which go beyond the sending of standard signals. As far as IRT is concerned, the jitters which may still be encountered with RT during communication are significantly reduced by synchronizing the network.

A time-slot procedure is required at a level above that of the Ethernet network. One time slot is reserved for the IRT telegrams and another slot is reserved for the RT and IP-based telegrams, in which the standard Ethernet communication (NRT (optional)) also runs. This type of approach requires all devices involved in IRT communication to be synchronized.

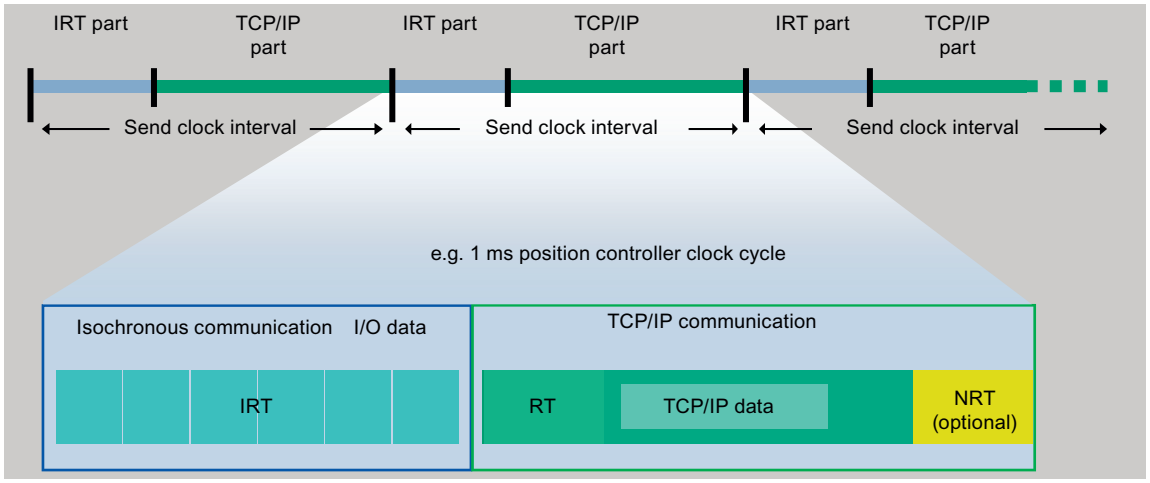


Figure 4-339 IRT Communication - Overview

See also IRT High Performance (Page 2118) with optimized bandwidth reservation and scheduled IRT communication.

For PROFINET IO with IRT, all IRT devices are synchronized on a shared sync master. See also Isochronous operation and isochronous mode with PROFINET (Page 2120).

PROFINET IO with IRT (High Performance)

The performance capability of motion control applications is significantly increased with PROFINET IO IRT (High Performance).

PROFINET is based on Ethernet technology, which is in turn based on point-to-point connections. Point-to-point connections support a significantly higher transmission rate when compared with arrangements based on parallel wiring. PROFINET uses 100 Mbps. Using this in conjunction with switching technology means that all connecting cables are decoupled from each other, enabling each cable to both transmit and receive at the same time.

Scheduling the telegram traffic for IRT High Performance enables data traffic to be optimized to a considerably higher degree, as only the bandwidth which is actually required is reserved.

IRT (High Performance) is particularly suitable for:

- The control and synchronization of axes via PROFINET IO
- A fast, isochronous I/O integration with short terminal-terminal times

Send clock

The send clocks that can be set are described in the table in Section Adjustable send clocks and update times (Page 2114).

The actual send clock that can be used depends on various factors:

- The process; communication should be no faster than required. This reduces the bus and CPU loads.
- The bus load (number of devices and the amount of IO data per device)

- The computing power available in the controller
- The supported send cycles on the PROFINET devices involved in a sync domain.

The supported send clocks can be found in the corresponding manuals of the respective SIMOTION controls.

Isochronous application

Isochronous data transmission and an application synchronized with the bus system satisfy the requirements associated with demanding motion control applications. This makes it possible to close control loops via the bus system and achieve minimum guaranteed response times (terminal-to-terminal time response). In addition, a high-performance and isochronous connection to the application with low load on the application CPU is ensured.

Unlike standard Ethernet and PROFINET IO with RT, transmission of telegrams for PROFINET IO with IRT High Performance is scheduled.

Time-scheduled data transmission

Scheduling is the specification of the communication paths and the exact transmission times for the data to be transferred. The bandwidth can be optimally utilized through communication scheduling and therefore the best possible performance achieved. This requires the network topology to be configured, with the engineering system automatically calculating the communication schedule from the configured topology (see also Topology (Page 2209)). The data relevant to PROFINET IO is transmitted by means of a download from HW Config to the IO controller. The highest determinism quality is achieved through the scheduling of the transmission times which is especially advantageous for an isochronous application connection.

Data exchange

PROFINET IO with IRT (high performance) only runs within a sync domain. These must not however be mixed in one IO system. In other words, a sync domain may consist of two or more IO systems which can all be synchronized with one another. Within the IO system, PROFINET IO with IRT (high performance) is then used.

Sync domain

A sync domain is a group of PROFINET devices synchronized to a common cycle clock. The sync slaves synchronize themselves with the cycle clock set by the sync master. The role of a sync master can assume a control system (IO controller) or a SCALANCE X200 IRT (IO device). A sync domain has just one sync master.

See also

Creating a sync domain (Page 2199)

Isochronous operation and isochronous mode with PROFINET

Description

PROFINET IO with IRT is based on Ethernet with a higher-level time-slot procedure. This arrangement requires all bus interfaces involved in communication to be synchronized. In the case of PROFINET IO with IRT High Performance, a sync master transfers a synchronization message frame to which all sync slaves synchronize themselves.

Sync master, slaves, and domain

The sync master and sync slave device roles are assigned during the configuration. An IO controller or SCALANCE 200 IRT switches can be assigned a sync master role.

A sync domain can comprise PROFINET devices with IRT High Performance. PROFINET devices with RT may also be located at the ends (spur lines), but not between two PROFINET IO devices with IRT (High Performance).

A line (network) may only comprise IRT High Performance, and these must always be connected to each other directly.

Compatibility

Communication between and through different sync domains via PROFINET IO with RT is possible.

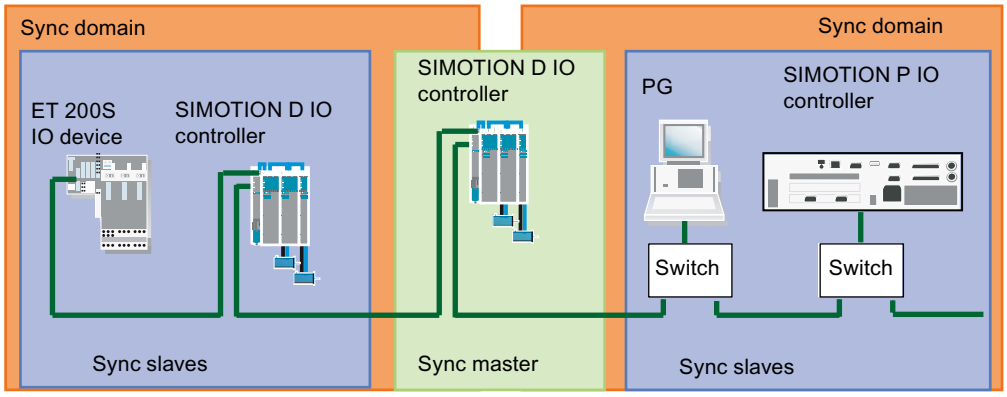


Figure 4-340 PROFINET isochronous mode

IRT-compatible switches, such as SCALANCE X204 IRT, are used in the structure. All devices involved in IRT communication are connected to each other directly. As the programming device in the center of the network is connected via a spur line, it does not interrupt the IRT path.

See also

Isochronous applications with PROFINET (Page 2126)

Addressing of PROFINET IO devices

A globally unique MAC (Media Access Control) address is used for data exchange via Ethernet and forms part of the Ethernet telegram. The MAC address is linked to the hardware and cannot be modified.

Ethernet-based protocols such as HTTP (Web applications) or FTP (file transfer) use the IP protocol. Addressing is based on the IP address. This is a logical address, which can be assigned by the user.

PROFINET uses a device name (NameOfStation) to identify PROFINET devices, in addition to the two items of address information already known in connection with Ethernet. The device name is a string that fulfills the requirements of a DNS (Domain Name Service) name. This device name must be unique across the PROFINET network.

During the commissioning phase, each PROFINET device (identified via the MAC address) is assigned a device name once via the configuration tool and this is stored retentively in the PROFINET device (a process known as node initialization). A device is referenced in the configuration via the device name. If a device is replaced, e.g. because of a defect, the new device has another MAC address. If it is initialized with the same device name as the replaced device (e.g. by reconnecting a removable medium that stores the device name retentively), it can take over the function of the replaced device without any changes in the configuration.

Alternatively, the device can be initialized automatically by the controller on the basis of topology information. This is only possible if topology information (who is wired to whom) is stored in the engineering system. During ramp-up, the controller identifies the connected devices using the device names, before assigning the IP address defined in the engineering system to the device. The station can then be accessed via IP services. The IP address can be taken from a configured sequence of numbers or configured individually.

A PROFINET device has the following addresses by which it can be addressed:

- MAC address (part of the Ethernet telegram, stored on the device, and cannot be modified)
- IP address (IP-based communication such as engineering access, must be assigned to all devices)
- Device name (devices identified by the controller during ramp-up)

See also

Assigning device names and IP addresses to IO devices (Page 2221)

Planning and topology for a PROFINET network

Planning guidance

Fundamental questions must be resolved before implementing the design of a PROFINET network. In this chapter, you will find broad guidelines to support you in defining requirements and creating planning documentation. Planning is an iterative process, i.e. some requirements influence each another and therefore necessitate changes in the overall plan.

Content of the planning documentation

Once the requirements have been specified and the planning process has come to an end, the following information should be available to you:

- System configuration
- Topology
- Selection of components
- Selection of transmission medium
- Connector selection
- Communication relationships
- Estimate of the data volumes to be transferred

Preliminary considerations and analysis during the planning stage

1. **Selecting the devices**
   Create a list of devices. The selection of devices is based, among other things, on the application class (conformance class), the time and communication requirements, the function, environmental influences, and the degree of protection.
2. **Position of the devices in the machine**
   The position of the devices in the machine has implications for the degree of protection, EMC, device dimensions and the cables used, e.g. whether fiber-optic cable should be used rather than copper cable. This in turn influences device selection.
3. **Defining the communication properties**
   The time requirements concerning the application (isochronous/cyclic) and communication via PROFINET must be defined, i.e. does communication take place in real-time (IRT/RT) or is it acyclic via TCP/IP or UDP/IP.

4. **Network planning (topology)**
   Specify the network topology (ring, star, line). Depending on the type of topology selected, the following must be taken into account: switches (with IRT capability), EMC, extent of network, WLAN (IRT not possible) and, if applicable, media redundancy MRP (possible with SIMOTION V4.3 and higher, and SINAMICS V4.5).
   - Set up your PROFINET in a point-to-point architecture where this is useful (for example, use a switch to branch off into a point-to-point topology downstream of a CPU).
   - In the case of PROFINET with IRT, a line structure with 64 IRT devices is permissible. The amount of data transmitted has certain implications. If longer message frame lengths are configured for each device, the possible number of devices per line may be reduced. However, this is detected early on during configuration with HW Config and signaled by means of an error message. The 64 IRT devices in the line are only applicable to PROFINET after V2.2.
   - Maintain a low interconnection depth for the switches. This will reduce the effect of a worst-case jitter scenario with RT communication.

Topology	
Star	If you connect communication nodes to a switch, you automatically create a star-shaped network topology. With this structure (unlike with other structures), if an individual PROFINET device fails, this does not automatically lead to the failure of the entire network. Only the failure of a switch causes the failure of devices downstream of the switch.
Tree	If you interconnect several star-shaped structures, you obtain a tree network topology.
Line	All the communication nodes are connected in series as a bus. If a switch fails, communication downstream of the failed switch is no longer possible. Devices with a 2-port switch must be used in order to set up a linear structure. Linear network structures require the least amount of cabling.
Ring	With V4.3 or higher, you can establish a ring topology for MRP or MRPD. For the ring topology you must define a redundancy manager and redundancy clients.

5. **Defining the communication relationships (logical assignment of partners)**
   Specify which communications partners are connected as well as the spatial and functional assignment of these partners.

6. **Defining the data volumes**
   Define the possible data volumes of the network nodes and those at the communication junctions.

   Note

   Communication within a PROFINET network should only be as fast as the technology of the system requires it to be and not as fast as technically possible.

7. **Defining the sending cycle and update time**
   The sending cycle is determined by the PROFINET device which has to be updated the most frequently. The update time is a multiple of the sending cycle and determines the PROFINET network load. The network load generated by PROFINET should always remain below 50% to allow reserve capacity for peak loads. Please note that the PROFINET network load increases in linear relation to the number of PROFINET devices and the sending cycle.

   Note

   Always configure the update times as required by the process, even if the bus system allows for a very large number of shorter update times. This reduces the PROFINET network load and the load on the PROFINET controller to what is strictly necessary.

8. **Checking the network load**
   In order to determine the network load, you must take into account the PROFINET network load as well as the network load generated by standard Ethernet devices. These can take the form of video cameras for monitoring the system or data servers for production data, for example. Ethernet devices with a low data volume such as engineering workstations or HMIs, for example, are normally non-critical. To prevent the PROFINET RT data stream from being compromised, you should, if necessary, adapt the network topology in cases where high network utilization by Ethernet nodes is anticipated. However, you also need to make sure you have sufficient bandwidth reserves for future expansion.

   Note

   Devices which exert a high IP load on the network should be located, where possible, in a separate area of the network. The diagnostics server and HMI server are examples of such devices.

9. **Connection to the company network**
   Various points must be considered if the automation system is to be connected to the company network. Normally, you should establish the connection via a router or a SCALANCE S with firewall to prevent unauthorized access. PROFINET IRT or RT communication via routers is not possible. You should only establish further connections (such as remote access or VPN, for example) in individual cases following consultation with the IT department.

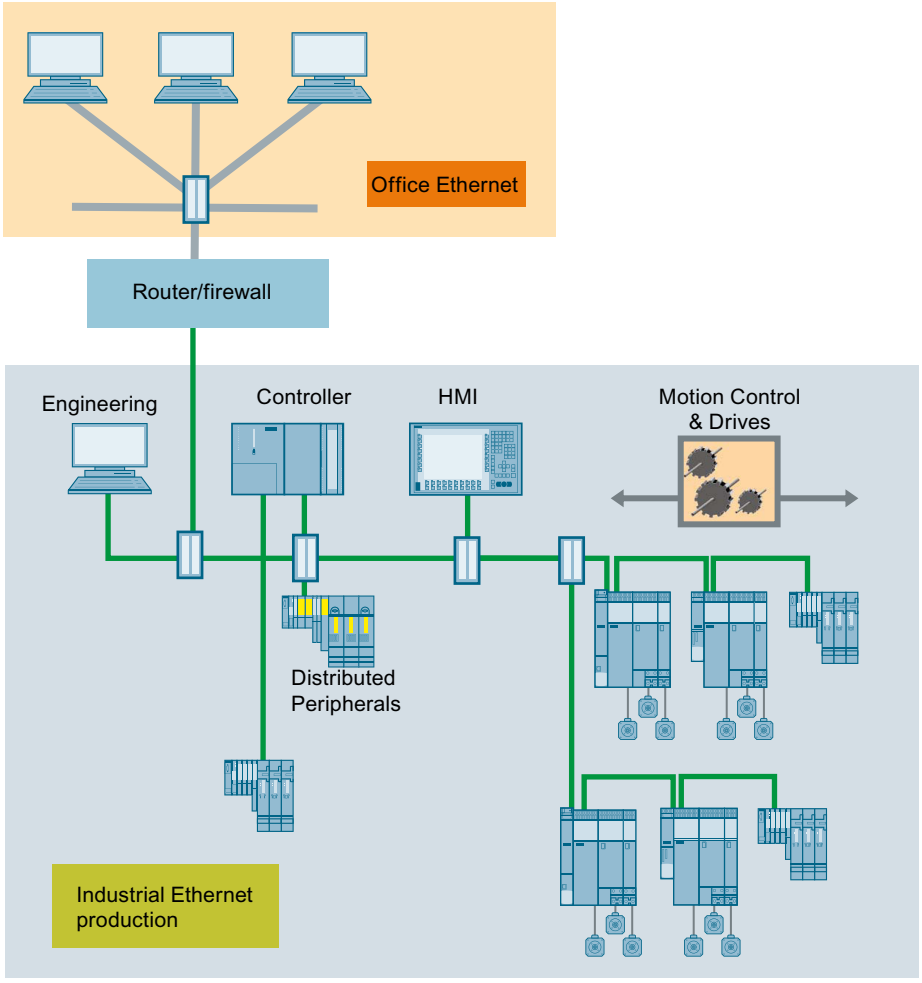


Figure 4-341 Optimized topology of a company network connection

Configuring the topology

Communication scheduling requires knowledge of the network topology. This includes information about interconnecting the individual devices to create a communication network.

Topology scheduling is only relevant for IRT High Performance. The topology editor that has been integrated in the hardware configuration enables user-friendly configuration of the network topology.

Isochronous applications with PROFINET

As with PROFIBUS, in the case of PROFINET IO with IRT High Performance the application can be synchronized with the transmission network's cycle clock. Distributed Motion Control applications and terminal-to-terminal response times of less than a millisecond require all PROFINET devices with IRT High Performance to be synchronized with a common time base.

Note

Isochronous mode for the application on the bus is only possible for PROFINET IO with IRT High Performance.

Configuration model for isochronous mode, V4.4 and higher

The configuration model for isochronous mode has changed as of V4.4. A selection must be made in the controller properties to specify whether isochronous IO devices are being operated. The isochronous task, the clock generator, must be selected. You can find a more detailed outline of the procedure for making settings in isochronous applications in the chapter titled Inserting and configuring the SINAMICS S120 (Page 2215)

Procedure

When configuring isochronous applications, proceed as follows in HW Config:

1. Set the synchronization role for the IO controller to **Sync Master**. For SIMOTION (controller) V4.2 or higher, the RT class is automatically set to **IRT** and **High Performance**.
   (Double-click in HW Config on the PN interface of the IO controller. You can set the **Synchronization role** in the **Synchronization** tab of the Properties window.)

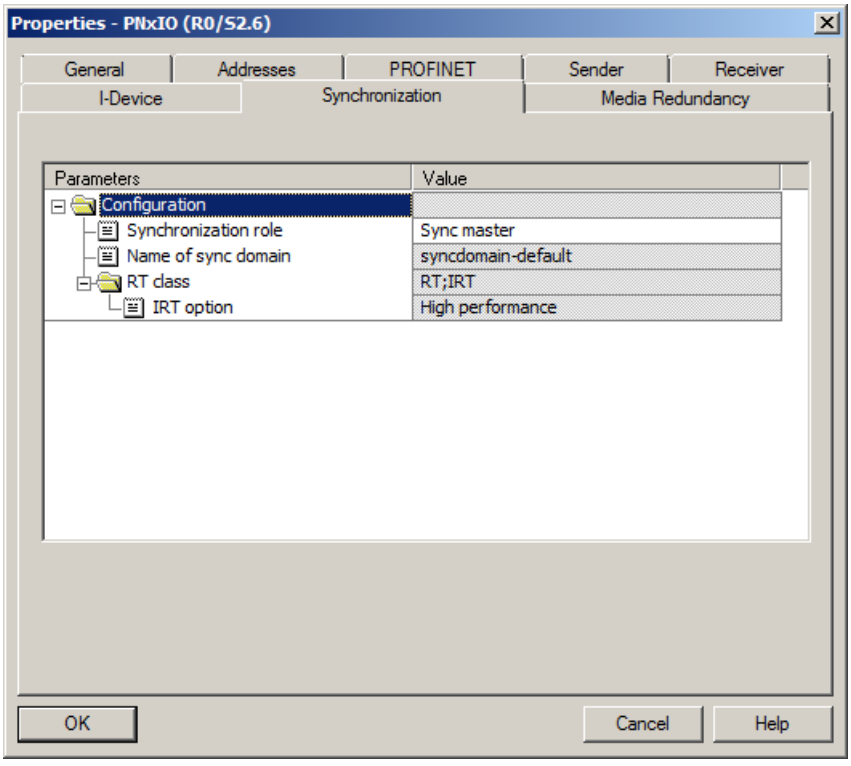


Figure 4-342 Setting the RT class

2. Set the synchronization role on the IO devices to **sync slave**. The RT class must be set to **IRT** and **High Performance**.
   (Double-click in HW Config on the PN interface of the IO devices. You can set the **Synchronization role** in the **Synchronization** tab of the Properties window.)

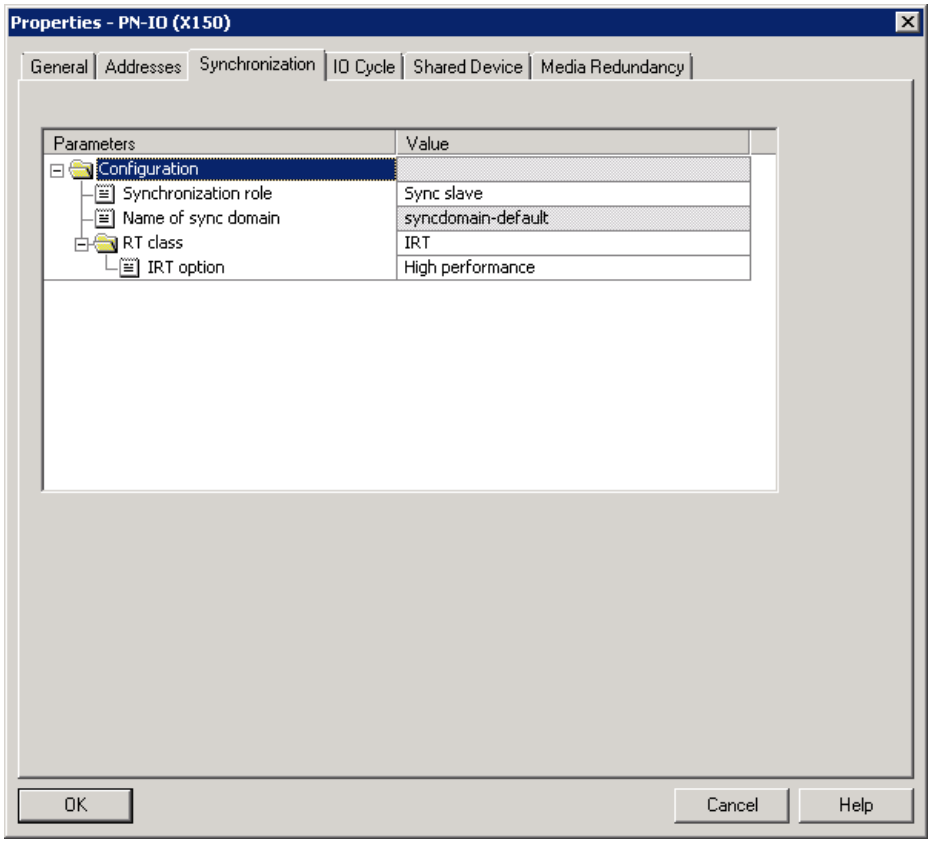


Figure 4-343 Setting IO device synchronization to sync slave and setting IRT High Performance.

3. Select the task (**Servo_fast** or **Servo**) in the **Isochronous tasks** tab in the controller properties. In general, **Servo** is used. The reference cycle clock is shown grayed out. The checkbox must be set for the following configurations:
   - Operation of isochronous IO devices (e.g. drives).
   - Configuration of exchange broadcast data. This is always isochronous.
   - Use of an iDevice with isochronous IO data.

   If two PN interfaces (e.g. X150 and X1400) of the controller are used with isochronous IO data, i.e. both checkboxes in the **Servo** field are activated, this is referred to as "coupled I/O" or "coupled IO systems". The two IO systems are then synchronized with one another.

Note

The checkbox must **not** be set if no isochronous IO data is configured. During compilation, inconsistent settings are displayed with an error message.

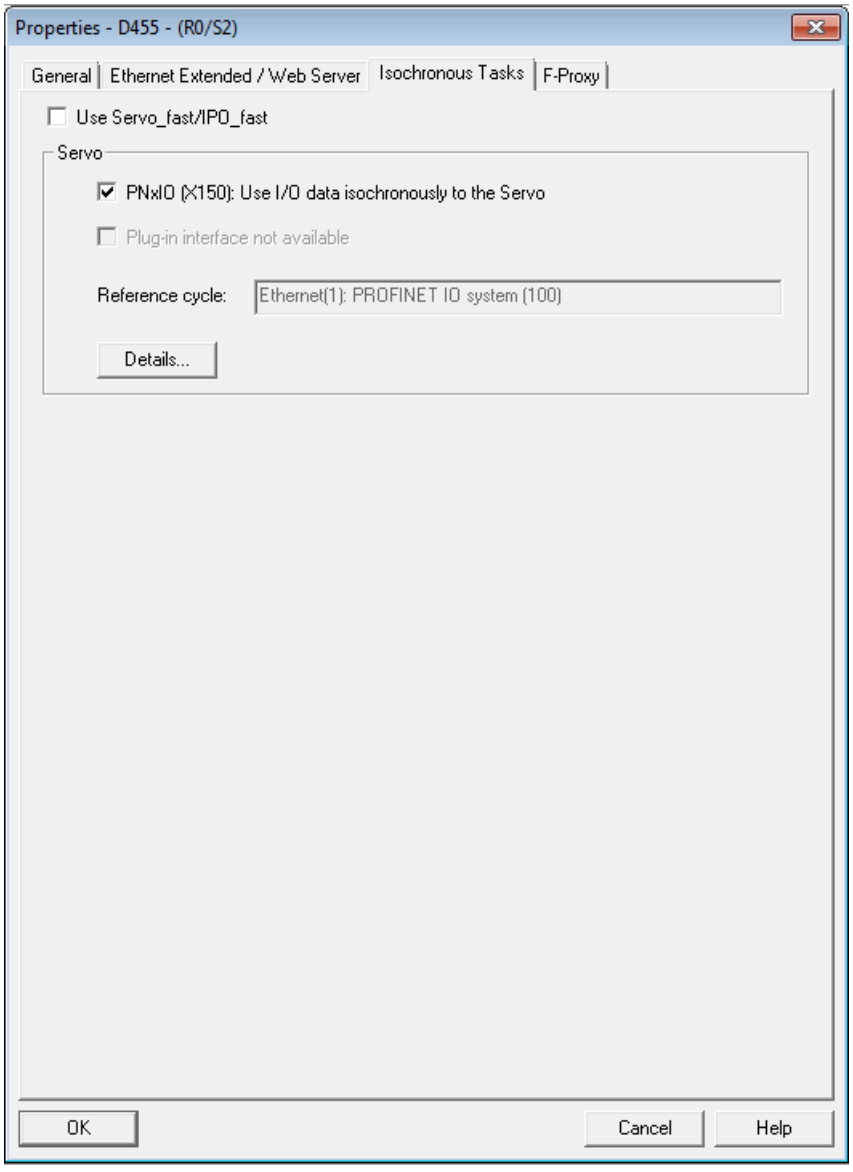


Figure 4-344 Isochronous IO devices are operated on the controller

Note

Servo_fast

If you activate Servo_fast, the other settings for Servo_fast are displayed in the dialog. You can then select **Details..** to enter further settings. If you have activated Servo_fast, you can also select Servo_fast as the reference cycle for the isochronous application (isochronous IO device).

The Servo_fast option is not available for all SIMOTION controllers and is not displayed for such controllers.

4. On the devices, set the **Update time** mode to **Fixed factor** and select the cycle clock (servo or Servo_fast) under **Assign IO device isochronously**. Here, you can select the cycle clock that was set under "Isochronous tasks" in the previous step. **Servo** is used in the example. (Double-click in HW Config on the PN interface of the IO devices. You can configure the settings in the **IO Cycle** tab of the Properties window.)

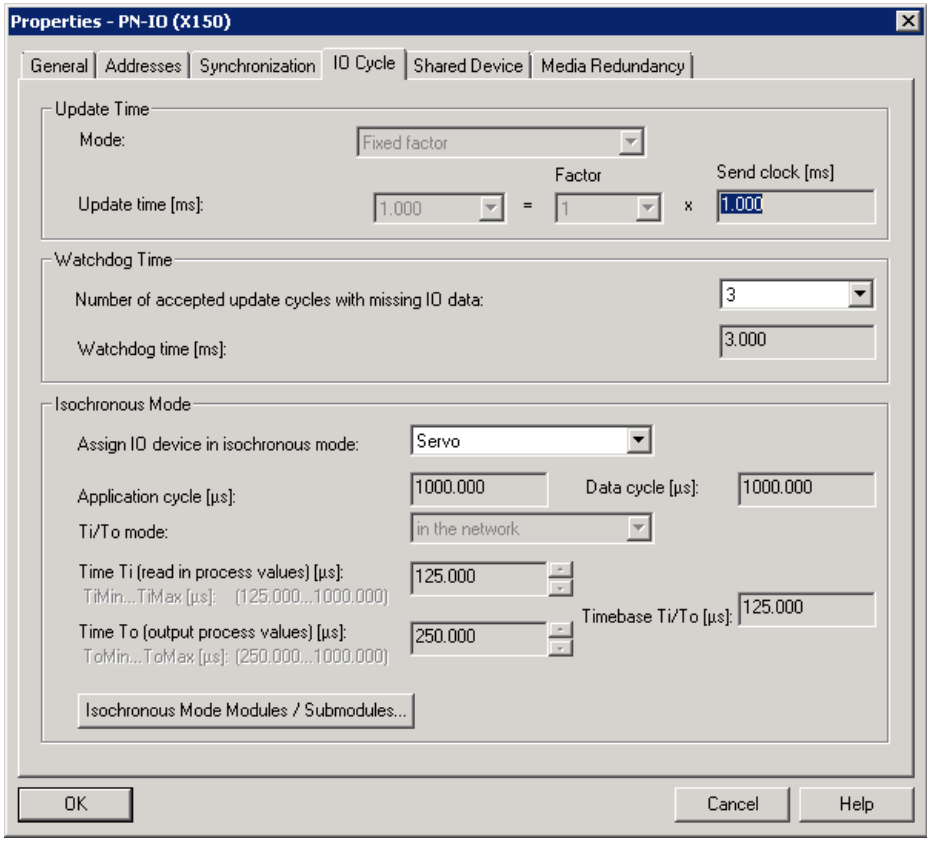


Figure 4-345 Assigning IO cycle parameters

Acyclic communication via PROFINET

Description

Similarly to PROFIBUS DP, it is also possible for PROFINET IO to operate acyclic communication (Base Mode Parameter Access). You will find a detailed description hereof under DP V1 communication (Page 2415).

Shared device

Shared device functionality

Large systems or systems spread out over a large area frequently make use of numerous IO controllers. It can, therefore, happen that sensors that are physically near one another must supply data to different IO controllers. Previously, this could be solved using multiple IO devices which were assigned to the different IO controllers. The shared device functionality makes it possible to divide up the submodules of an IO device between different IO controllers in order to save one or more interface modules.

Further application

Safety engineering is required in a system for certain system sections. Therefore, in addition to the standard CPU, an F-CPU is used which ensures that the critical system sections are shut down safely. With the shared device function, it is possible to assemble an IO device from F modules and standard modules and to assign the individual modules to either the F-CPU or the standard CPU accordingly.

The application with an F-CPU is described briefly in the PROFIsafe chapter (Page 2368).

Principle

Access to the submodules of the shared device is divided between the individual IO controllers. Each submodule of the shared device can be assigned exclusively to one IO controller. The individual submodules are assigned in HW Config.

iDevice

iDevice functionality

The PROFINET I device functionality is comparable with that of the I-slave for PROFIBUS, i.e. a SIMOTION CPU can accept the role of an IO device and thereby exchange data with a different IO controller.

Whereas with PROFIBUS an interface can be either a master or slave only, with PROFINET it is possible to be both an IO controller and IO device at the same time on a single PROFINET interface.

Shared iDevice (V4.4 or higher)

As of SIMOTION V4.4, an iDevice of the PN-IO interfaces can be operated with two higher-level controllers. This functionality is referred to as "shared iDevice" and is comparable with the shared device. A configuration example for a shared iDevice can be found in the chapter Shared iDevice and PROFIsafe (Page 2374).

See also

PROFINET IO and I device (Page 2238)

4.5.5.2 Properties and functions of PROFINET IO with SIMOTION

Introduction

Requirement

To work with SIMOTION using PROFINET IO, you must have at least one PN interface available. This can either already be integrated within the controller or inserted via an (additional) option board.

Connection possibilities:

- SIMOTION D4x5 with Option Board CBE30
- SIMOTION D4x5-2 DP/PN
- SIMOTION D4x5-2 DP/PN with CBE30-2 (from V4.3)
- SIMOTION P350 PN or SIMOTION P320-4
- SIMOTION D410 PN/SIMOTION D410-2 DP/PN
- SIMOTION C240 PN

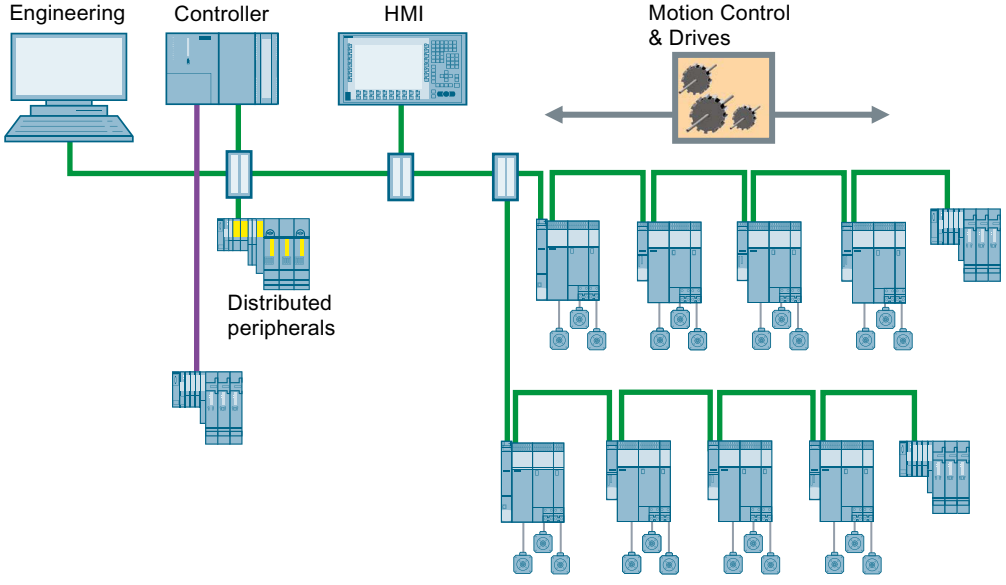


Figure 4-346 System topology with PROFINET

The SIMOTION controllers as PROFINET IO controllers support the simultaneous operation of:

- IRT High Performance - isochronous realtime Ethernet
  - Operation of IRT I/O (e.g. ET200S HS for IRT High Performance)
  - Operation of a SINAMICS S120 as an IRT IO device
  - Data exchange between IO controllers via IRT High Performance (e.g. distributed synchronous operation)
- RT - realtime Ethernet
  - Operation of RT - peripherals (e.g. ET 200S, ET 200pro)
  - ASi link via IE/AS interface link PN IO for the PROFINET IO gateway to AS interface
  - SINAMICS as RT IO device
- TCP/IP, UDP, HTTP, ... standard Ethernet services

Note

With a mixed operation of PROFINET IRT and PROFINET RT, one must be aware that the IRT IO devices have to be connected together directly; i.e. no IO devices which PROFINET IRT does not support are allowed to be connected between two IRT devices The IO device that is not IRT-compatible must be located at the end of the IRT line so that IRT operation is not interrupted.

Note

With SIMOTION SCOUT, it is possible to access a maximum of 10 nodes ONLINE simultaneously. If you have installed SIMATIC NET, it is possible to establish more connections. The exact number is based on the available resources of the network adapter.

Note

For a PROFINET IO device that is connected to a SIMOTION controller, max. submodule sizes are defined. This limit is especially important if a SIMATIC controller is configured as an I-device for a SIMOTION controller . Therefore, please note the possible quantity structures (Page 2162).

PROFINET V2.2/V2.3

SIMOTION SCOUT supports PROFINET V2.2 and is compatible with V2.3. Older versions are no longer supported as standard in the hardware and must be changed over.

When inserting SIMOTION controllers and/or SINAMICS drives, only PROFINET V2.2 versions are inserted in SIMOTION SCOUT. If you want to configure older versions, you need to explicitly add the hardware as PROFINET V2.1 in HW Config.

PROFINET V2.3 for SIMOTION D455-2 V4.4

SIMOTION D455-2 DP/PN supports PROFINET V2.3. In particular, the following functions are available:

- Fast Send Clock
  Isochronous mode with send clock ≥ 125 µs
- Dynamic Frame Packing (DFP)
  With DFP, the cyclic data of multiple IO devices is summarized in one Ethernet frame.

See also

PROFINET V2.3 Performance Upgrade (Page 2175)

Cycles and cycle support

Cycle clock scaling with PROFINET IO on SIMOTION devices

Description (PROFINET IO with IRT High Performance)

An isochronous application (e.g. position controller) on an IO controller must be synchronized with the send clock for IRT High Performance. It can, however, be synchronized with a multiple of the send clock of the data. This multiple is designated as CACF (Controller Application Cycle Factor). Cycle clock scaling is configured in SCOUT with the execution system via **Expert > Set system cycle clocks** in the shortcut menu or via **Set system cycle clocks** from the shortcut menu of the highlighted SIMOTION controller in the project navigator.

Example: The data on the network is transferred with a send clock time of 1 ms. However, the servo should run with 2 msec. Therefore, the application cycle must be 2. For this purpose, set the ratio 2 at **Set system cycle clocks** for the servo.

The CACF is set on the IO device, see e.g. Inserting and configuring the SINAMICS S120 (Page 2215).

V4.2 and higher, two servo cycle clocks (servo, Servo_fast)

With V4.2 and higher, Servo_fast is introduced as a second position control cycle clock for fast applications. The second servo cycle clock enables you to operate two bus systems in different application cycle clocks (PROFINET and PROFIBUS). An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This enables you to divide your application into a slow section (servo and IPO) and a fast section (Servo_fast and IPO_fast). If a Servo_fast is configured, it is linked to the send clock of PROFINET IRT at a ratio of 1:1. The servo cycle clock must be set as a multiple of Servo_fast (at least 2x Servo_fast). In this case, the CACF continues to be the factor between the send clock/Servo_fast and servo.

The functions described below mainly relate to configurations without the option of a second servo (Servo_fast).

Note

You can find a detailed description of the Servo_fast option in the *SIMOTION SCOUT Basic Functions Manual*, Chapter 6.

With V4.3 or higher, two PROFINET IO interfaces with different cycle clocks (SIMOTION D4x5-2 DP/PN with CBE30-2)

With V4.3 or higher, you can operate two PROFINET IO interfaces with different cycle clocks. With V4.2, it was only possible to operate one interface with PROFINET IO and one with PROFIBUS DP in different cycles (see above).

For the rules that apply when using Servo_fast, see Two PROFINET IO interfaces (Page 2144).

Description

Scaling to the send clock with SIMOTION controllers is possible in the case of PROFINET with IRT High Performance under the following conditions:

- The SINAMICS Integrated of a SIMOTION D and an isochronous DP master interface always run in accordance with the servo cycle clock.
- For a SIMOTION P350/C240, the isochronous DP master interface always runs simultaneously with the servo cycle clock.
- For drives (e.g. S120) which are connected via SINAMICS Integrated or as an external drive unit, the servo cycle clock must **always** be counted in the first send clock. This means that down-scaling is possible, although the position controller must be counted in a send clock.

The following general conditions apply to cycle clocks and cycle clock scalings for a SIMOTION controller

- The servo cycle clock/Servo_fast cycle clock is only synchronous with the send clock if the following conditions apply:
  - At least one IO device is operated isochronously.
  - At least one I-device is operated isochronously.
  - One controller-controller communication is configured.

Combinations of cycle clocks and cycle clock sources for PROFIBUS and PROFINET IO

- Servo can be scaled in integral multiples (1, 2, ...n) of the send clock.
- If a Servo_fast is configured, then the CACF = 1 for Servo_fast and Servo is also fixed and cannot be changed (not configurable).
- Cycle clocks for SINAMICS Integrated and isochronous DP master interfaces must run simultaneously with the servo cycle clock.

Cycle clock scaling for IO accesses

Description of PROFINET IRT data transmission

The following must be observed for cycle clock scaling (PROFINET and PROFIBUS):

- PROFINET IO IRT data is always read at the beginning of the servo cycle clock and written at the end of the servo cycle clock.
- PROFINET IO RT data is read at the beginning of the IPO or IPO2 cycle clock and written at the end of the IPO cycle clock.
- At the end of a IPOSynchronousTask, the process image is output with the next possible servo (Data Out) (= response-time-optimized). If the position control cycle clock is down-scaled to the IPO cycle clock (position control < IPO), this can lead to the data being output one or more position control cycle clocks earlier or later within an IPO cycle clock, if the I/O accesses are performed via the IPOSynchronousTask. This is the case if the runtime for the IPO cycle clock is not constant and, as a result, data is transmitted earlier or later on the bus with a faster position control cycle clock.
- At the end of the position control execution level, the process image of the ServoSynchronousTask is output with the next possible bus cycle clock (= response-time-optimized).
- If the PROFINET cycle clock is down-scaled to the position control cycle clock (PROFINET <= servo), the data is always output in the first PROFINET cycle clock.
- For PROFIBUS, the data is always output with the first bus clock cycle, since the servo priority class must always be finished with the first bus clock cycle. In case of a different runtime of the servo priority class in the individual cycles, the terminal-terminal time may vary as a result.

If an always constant response time is to be achieved instead of a response-time-optimized behavior, the following must be set:

- For PROFIBUS:
  - A reduction ratio servo: IPO = 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode.
  - Comment: IO accesses from the ServoSynchronousTask are always isochronous for PROFIBUS
- For PROFINET:
  - A reduction ratio bus clock cycle: Servo: IPO = 1 : 1 : 1 so that the I/O accesses from the IPOSynchronousTask are always implemented in isochronous mode
  - A reduction ratio bus clock cycle: servo = 1 : 1 so that the I/O accesses from the ServoSynchronousTask are always implemented in isochronous mode

Note

With I/O access, cycle clock down-scaling can result in an offset (fixed dead time) for the IPO of a send cycle clock.

Bus cycle clocks that can be adjusted for cycle clock scaling to SIMOTION devices

Overview of the possible bus cycles

	PROFIBUS[2)]	PROFINET IRT High Performance	PROFINET IRT High Performance	Servo Servo_fast
	Minimum	Minimum	Maximum	Minimum
SINAMICS S120 CU310-2 PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S120 CU320-2 PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S110 CU305 PN	1.0 ms	1.0 ms	4.0 ms	1.0 ms
SINAMICS S120 CU320-2 DP with CBE20	1.0 ms	0.25 ms	4.0 ms	0.25 ms
SINAMICS S120 CU320-2 PN with CBE20	1.0 ms	0.25 ms	4.0 ms	0.25 ms
C230-2	1.5 ms	-	-	1.5 ms
C240 PN	1.0 ms	0.5 ms	4.0 ms	0.5 ms
C240	1.0 ms	-	-	0.5 ms
D410 PN	-	0.5 ms	4.0 ms	2.0 ms
D410 DP	2.0 ms	-	-	2.0 ms
D410-2 DP	1.0 ms	-	-	0.5 ms[3)]
D410-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.5 ms[3)]
D425	2.0 ms	0.5 ms	4.0 ms	2.0 ms
D425-2 DP	1.0 ms	-	-	0.5 ms
D425-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.5 ms
D435	1.0 ms	0.5 ms	4.0 ms	1.0 ms
D435-2 DP	1.0 ms	-	-	0.5 ms
D435-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms[1)]
D445/D445-1	1.0 ms	0.5 ms	4.0 ms	0.5 ms
D445-2 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms[1)]
D455-2 DP/PN	1.0 ms	0.125 ms	4.0 ms	0.125 ms [1)]
P350-3	1.0 ms	0.25 ms	4.0 ms	0.25 ms
P320-3	-	0.25 ms	4.0 ms	0.25 ms
P320-4 PN	-	0.25 ms	4.0 ms	0.25 ms
P320-4 DP/PN	1.0 ms	0.25 ms	4.0 ms	0.25 ms
ET200S High Speed	-	0.25 ms	4.0 ms	0.25 ms
ET200SP High Speed	-	0.125 ms	4.0 ms	0.125 ms

1) Explanation:

- 0.5 ms in conjunction with SINAMICS S120 (incl. SINAMICS Integrated/CX32-2)
- 0.25 ms in conjunction with SERVO_fast and IPO_fast (D455-2 DP/PN: 0.125 ms)

2) Explanation:

- The specifications apply to "external" PROFIBUS interfaces and not to PROFIBUS Integrated for SIMOTION D. PROFIBUS Integrated for SIMOTION D: minimum 0.5 ms

3) Explanation:

- 1 ms when using the TO axis and the integrated drive control

Cycle scaling with PROFINET IO

As of V4.3, it is possible to down-scale the PROFIBUS cycle to the position control cycle. Down-scaling is only permitted if PROFINET with IRT has not been configured. It is also possible to down-scale the PROFINET send cycle to the PROFIBUS cycle. For example, PROFINET send cycle = 0.5 ms and PROFIBUS cycle = position control cycle = 1 ms.

The PROFIBUS cycle can be operated relative to the PROFINET send cycle at a ratio of 1:1 to 16:1. The table below shows the possible ratio settings for the system cycles based on the DP cycle of the SINAMICS_Integrated or PROFINET send cycle.

Table 4-225 Ratios of the system cycles (when one servo is used)

Cycle name	Adjustable factors	Reference cycle
PROFIBUS DP bus cycle	1, 2, 3, 4, 6, 8, 10, 12, 14, 16	PROFINET IO send cycle
Servo	As of V4.3: 1, 2, 3, 4, 8[1)] V4.2: 1	PROFIBUS DP bus cycle
IPO	1, 2, 3, 4, 5, 6	Servo
IPO_2	2, 3, 4, 5, ..., 64	IPO

1) Always "1" if PROFINET with IRT has been configured

Table 4-226 Ratios of system cycles (two servos)

Cycle name	Adjustable factors	Reference cycle
Servo_fast	1	Send cycle of the onboard PROFINET IO interface (X150)
IPO_fast	1, 2, 4	Servo_fast
Servo	1	PROFIBUS DP bus cycle and send cycle of the optional second PROFINET IO interface (CBE30-2)
IPO	1, 2, 4	Servo
IPO_2	2, 3, 4, 5, ... 64	IPO

Ratios of the system cycles when using Servo_fast and Servo

- PROFIBUS DP bus cycle = N x send cycle of the onboard PROFINET IO interface X150
  N = 2, 4, 8, 16, 32, 64 (N = 2 only for send cycles > 250 µs)
  (for < V4.4: N = 2, 4, 8, 16 for all send cycles)
- Send cycle of an optional second PROFINET IO interface (CBE30-2) = PROFIBUS DP bus cycle
- IPO ≥ IPO_fast

Cycles with SINAMICS_Integrated (<500 µs)

Introduction

As of SIMOTION SCOUT/SIMOTION SCOUT TIA V4.5, SINAMICS_Integrated for drives on SIMOTION D4x5-2 can use bus cycles of less than 500 µs. It supports cycles between 250 µs and 375 µs. SINAMICS_Integrated remains assigned to the servo cycle. Servo_fast is not supported.

Supplementary conditions

The supplementary conditions for deployment of cycles < 500 µs are listed below.

PROFIBUS and Servo_fast

- External PROFIBUS interfaces cannot be used isochronously
- Since a factor of 4 between Servo and Servo_fast is prescribed for cycles of ≤ 250 µs, Servo_fast with 125 µs cannot be used in parallel to SINAMICS_Integrated with 250 µs.
- Because the Servo cycle must be a multiple of the Servo_fast cycle, Servo_fast cannot be used:
  - Servo_fast > 250 µs: N = 2, 4, 8, 16, 32, ...
  - Servo_fast ≤ 250 µs: N = 4, 8, 16, 32, ...

Number of axes

With regard to utilization, SINAMICS_Integrated is subject to the same restrictions as those applicable to an external SINAMICS drive.

If the SINAMICS_Integrated is operated with a DP Integrated cycle clock of 250 µs or 375 µs, the resultant utilization is higher in comparison to a cycle clock of ≥ 500 µs. Depending on the utilization situation (e.g. by deployed ALMs, CXs, TMs), this can result in a reduction in the number of axes (e.g. from six to five servo drives).

The higher utilization can result in a reduction of the quantity structure for vector and V/f-controlled drives.

CBE30

CBE30 supports a minimum send cycle of 500 µs. The SINAMICS_Integrated can only be operated with 250 µs if no isochronous communication is configured for the CBE30. If the CBE30 operates isochronously (assignment to the Servo), the Servo has a minimum cycle of 500 µs.

Task system and time response

Overview of SIMOTION task system and system cycle clocks

Overview

If IRT data is transmitted via the bus using PROFINET IO, the cycle clock execution times fall between reading and writing the data (e.g. axis data), depending on which task in the execution system the application is executed in. You can find examples of applications in different tasks (execution levels) in the chapters that follow.

BackgroundTask, MotionTask, and IPOSynchronousTask

MotionTask/BackgroundTask

The data is transmitted via the bus using PROFINET IO with IRT High Performance, and accepted by the communication interface at the start of the position control cycle clock. The logic signals are generally evaluated in a MotionTask or BackgroundTask. Here, a distinction is made as to which machine function is activated; for example, "position-controlled traversing of axis". The traversing profile required is counted in the next IPO cycle clock. Based on the position setpoints determined here, the speed setpoints for controlling the axis are calculated in the next position control cycle clock. These are transmitted to the drive in the next cycle, via PROFINET IO with IRT High Performance.

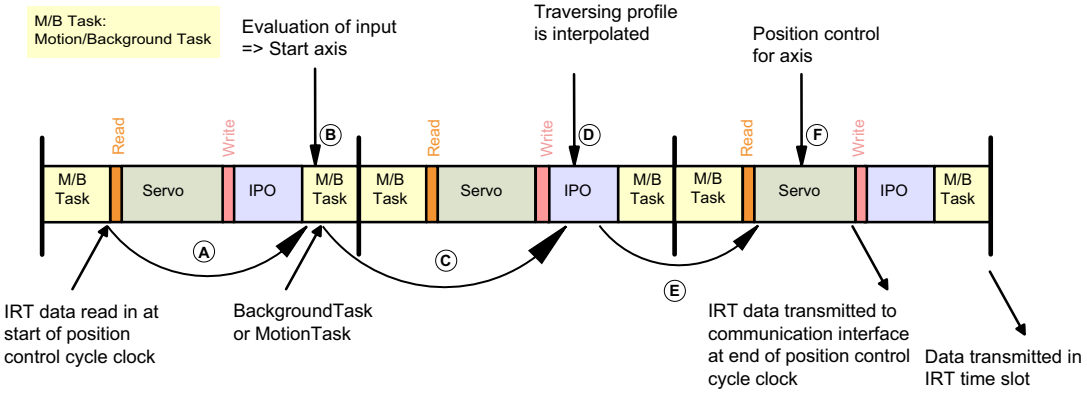


Figure 4-347 Logic evaluation for an axis in the BackgroundTask or MotionTask

Constraints

The option of a second servo clock is not activated, i.e. no Servo_fast and IPO_fast. The bus cycle clock, servo clock and IPO cycle clock are set to a 1:1:1 ratio. Other ratios may result in longer response times. With a 1:1:2 ratio, the IPO execution may be extended to two position control cycle clocks, which can lead to the response time increasing by one position control cycle clock.

Additionally, the BackgroundTask may be processed over several position control cycle clocks, meaning that the system is unable to make sure the data is evaluated in the first position control cycle clock. This may also lead to an increased response time.

Assigning the variables to a process image has an impact on the response time as well. The process images are made available to other tasks or to the communication interface at the end of the respective task, rather than after the variables are updated.

IPOSynchronousTask

In order to optimize the time response and enable synchronous triggering of actions (e.g. starting axes simultaneously), in the IPOSynchronousTask it is possible to process the part of the application that triggers axis commands. If this option is used, it is counted before the IPO. In this way, the axis command can be issued before the IPO is executed, and the resulting position setpoint then calculated in the IPO. Based on this, the speed setpoint for the drive is calculated in the next position control cycle clock. Once the position control cycle clock has finished, the data is passed on to the communication interface and transmitted in the next PROFINET IRT send clock. Unlike processing in a MotionTask/BackgroundTask, where the response time equals the maximum BackgroundTask runtime + one IPO cycle clock + one position control cycle clock, in this case you can be assured that the response time will be one IPO cycle clock + one position control cycle clock until new data is output.

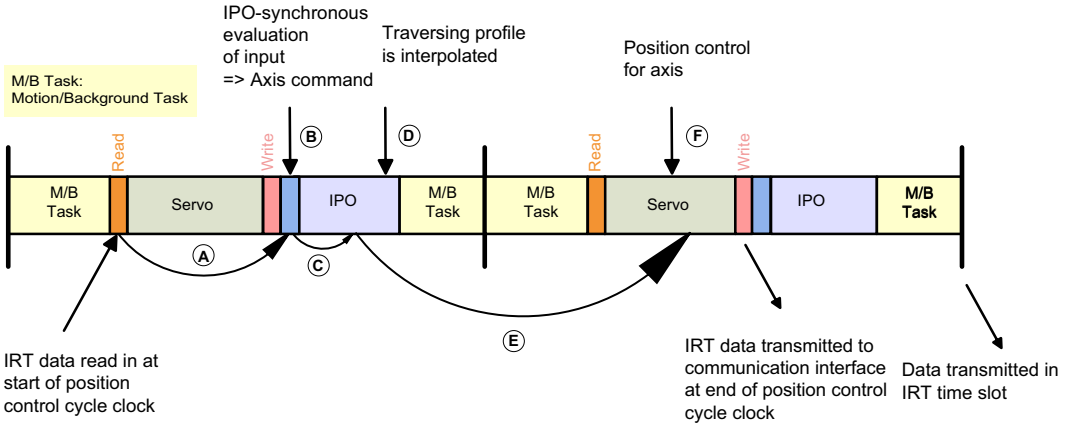


Figure 4-348 Logic evaluation for an axis in the IPOSynchronousTask

ServoSynchronousTask

ServoSynchronousTask

It is possible to optimize the time response even further and reduce the response time to one servo cycle clock. This option can be used for high-speed actual-value synchronous operations, e.g. flying knife/shear. Within this context, the part of the application that triggers axis commands for selected axes is processed in the ServoSynchronousTask. Additionally, the IPO part of the system for the axes involved is counted before the position controller in the servo task. In this way, the speed setpoints may be transmitted as soon as the next IRT time slot.

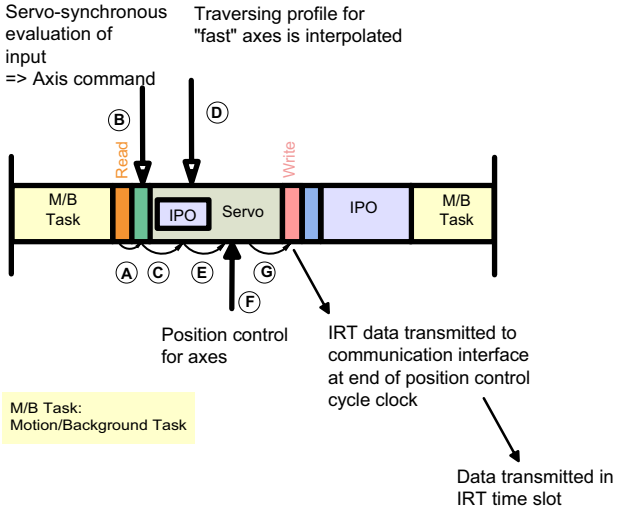


Figure 4-349 Logic evaluation for an axis in the ServoSynchronousTask

Boundary conditions

Using this function increases the CPU load and, therefore, the position control cycle clock; for this reason it should only be used when necessary.

Activating

This feature must be activated explicitly for the axes in SIMOTION SCOUT as part of axis configuration.

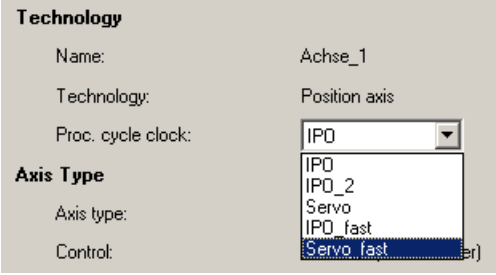


Figure 4-350 Determining the processing cycle clock for the axis

If the option of a second servo cycle clock is configured, then the axis can also be assigned to the Servo_fast processing cycle clock if this is linked to the faster bus.

Position control

The position controller responds within one position control cycle clock. The data is read out from the communication interface at the start of the position control cycle clock. The position controller is counted in the position control cycle clock. The new speed setpoints are copied to the communication interface at the end of the position control cycle clock and, therefore, transmitted in the next IRT time slot.

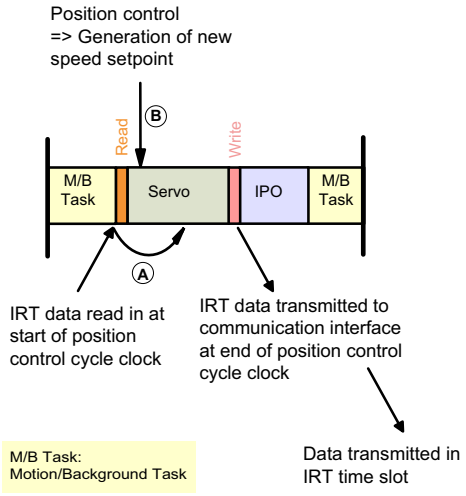


Figure 4-351 Position control time response with ServoSynchronousTask

Fast I/O processing in the ServoSynchronousTask

Fast I/O processing in the ServoSynchronousTask

The evaluation of quick I/Os – e.g. ET200S High Speed – is performed in the ServoSynchronousTask. resulting in a system response time of one cycle. Due to the terminal-terminal response, there is a delay of Ti + servo cycle clock + To.

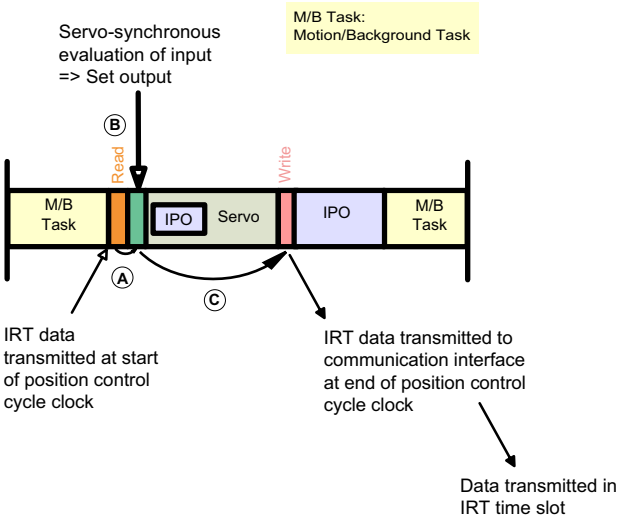


Figure 4-352 Fast IOs in the ServoSynchronousTask

If the **second servo clock** is configured, the fast I/O processing can also be performed in the Servo_fast clock (i.e. in the ServoFastSynchronousTask). This requires the I/O peripherals to be configured on the faster bus.

SIMOTION controllers with two PROFINET IO interfaces

Two PROFINET IO interfaces

SIMOTION controllers with two PROFINET IO interfaces

With the SIMOTION D4x5-2 DP/PN controllers, optionally a second PROFINET IO interface is available in addition to the onboard PROFINET interface (X150) with the Ethernet communication board (CBE30-2, X1400).

The CBE30-2 can only be operated in D4x5-2 DP/PN control units. It is impossible to use it in D4x5-2 DP control units.

Application description

A detailed application description explaining how to operate two different PROFINET networks on one SIMOTION controller at the same time is available to download in the Service and Support Portal.

PROFINET & SIMOTION: Use of Two PROFINET Interfaces with SIMOTION (https://support.industry.siemens.com/cs/ww/en/view/59396321)

See also

Example configuration for "controlled" sync master (Page 2158)

Cycle clock scaling with PROFINET IO on SIMOTION devices (Page 2134)

Use of Safety with MRRT (Page 2173)

Basic rules for using two PN IO interfaces

Introduction

The following rules must be considered in use of two **isochronous** PROFINET interfaces. The most important basic rules for the PN interfaces and the use of Servo_fast are sorted thematically there.

Definition of isochronous PN interfaces

A PN interface is isochronous (i.e. assigned to servo or Servo_fast) if the following conditions apply.

- The PN interface is the IO controller and one of the IO devices is isochronous.
- The I-device is configured isochronously.
- Controller-controller data exchange broadcast is configured.

Note

IO controller and I-device cannot simultaneously be configured isochronously (IRT) on a PN interface. One PN interface can be configured as the IRT controller and the second PN interface as the IRT I-device.

If a PN interface is configured as IRT and does not meet the conditions for isochronous operation, it will work **asynchronously** with respect to the Servo or Servo_fast clocks and therefore also asynchronously with respect to the second isochronous PN interface.

General rules when using the onboard PN interface (X150) and CBE30-2 PN interface (X1400)

- A higher-level, isochronous controller must always be connected via the CBE30-2 PN interface. This applies both to controller-controller data exchange broadcast communication and to I-device communication with a higher-level controller.
- Lower-level isochronous drives or IO devices should be connected via the onboard PN interface.
- Both PN interfaces can be configured as a PROFINET IO I-device and/or as an IO controller.
- If a PN interface is used as an IRT I-device, it cannot be configured as a redundant sync master. Only the sync slave and sync master roles are possible.

Constraints on Sync master and Sync slave

- The CBE30-2 PN interface can be configured as the sync master, sync slave, or redundant sync master.
- The onboard PN interface can only be configured as the sync master.
- In the Sync domain of the onboard PN interface, there must be no redundant sync master if the CBE30-2 is configured as a Sync slave or IRT I-device.

Failsafe I/Os for using two PN interfaces

An F-CPU can only be connected to one of the two PN interfaces, because fail-safe I/O transfer areas can only be configured either on the I-device on the onboard PN interface **or** on the CBE30-2 PN interface (I-device F proxy).

The PROFIsafe telegrams are routed to the following drives:

- Drives on the SINAMICS Integrated and CX32-2
- Drives on the external PROFIBUS
- Drives on the onboard PROFINET interface (X150 interface)
- Drives on the CBE30-2 (X1400 interface)
  The maximum quantity structure for PROFIsafe on PROFINET is therefore not increased by using a second PROFINET interface.

Use of Servo_fast

For the use of Servo_fast, the following rules apply.

- If Servo_fast is not used, either one or both PN interfaces can be assigned to the servo.
- If Servo_fast is used, the onboard PN interface is assigned to Servo_fast and the CBE30-2 PN interface can be assigned to the servo.

Constraints on the use of Servo_fast

- Only the onboard PN interface can be assigned to Servo_fast.
- If Servo_fast is used, only one CACF=MACF=1 (Controller Application Cycle Factor=Master Application Cycle Factor) is permitted Servo and Servo_fast, i.e. the servo clock is taken over from the set send clock in HW Config and does not have to be set especially.
  The following applies:
  - Servo clock = send clock CBE30-2 PN interface X1400
  - Servo_fast cycle clock = send clock onboard PN interface X150
- The onboard PN interface can only be operated as the sync master (not sync slave or redundant sync master).
  This means that **no** controller-controller data exchange broadcast can be configured between the two SIMOTION controllers if the onboard PN interfaces of both SIMOTION controllers are assigned to Servo_fast.
- The onboard PN interface can be an IRT IO controller but not an IRT I-device if assigned to Servo_fast.
- If the Servo_fast is used, it is essential that isochronous IO devices are assigned to it.

Constraints on the clock ratio and cycle time of servo/Servo_fast

- Rules for the cycle clock ratio **servo = N * Servo_fast**
  - Servo_fast > 250 µsec: N = 2, 4, 8, 16, 32, ...
  - Servo_fast <= 250 µsec: N = 4, 8, 16, 32, ...(For 32: Servo_fast = 125 µs, Servo = 4000 µs)
- The minimum send clock for Servo_fast depends on the SIMOTION device (see also "Adjustable cycle clocks")
  - D455-2 DP/PN: 125 µs
  - Other: 250 µs
- The maximum send clock for the servo depends on the source of the IO data.
  - PROFINET is the source of the IO data: 4 ms
  - SINAMICS_Integrated or PROFIBUS is the source of the IO data: 8 ms

Applications for devices with two PROFINET IO interfaces

Applications for two PN IO interfaces

Possible applications

The PROFINET interface of a CBE30-2 can be used, for example, for the following applications:

- The second PROFINET interface is used
  - to increase the maximum number of connectable IO devices
  - to increase the available I/O address space
- IO devices are to be operated with different send cycle clocks and assigned different isochronous tasks (Servo and Servo_fast).
- IO devices should have an independent IP address range or NameOfStation. Devices with identical device configuration at the local interface can thus be connected to each other independently via the other interface in the plant network (e.g. higher-level network as the plant network; local network for machine module; identical machine modules; participants in the "local network" can be addressed independently of nodes in the "plant network").
- I-device and IO controller are to be operated isochronously at the same time.
  If one PROFINET interface is operated simultaneously as an I-device and IO controller, either only the I-device or the IO controller can be operated isosynchronously (i.e. with IRT). With two PROFINET interfaces, it is possible to operate a PROFINET IO controller and an I-device isosynchronously on a D4x5-2 DP/PN at the same time.

Application examples

There are, for example, the following applications for two PROFINET IO interfaces:

1. Increase the number of devices on PROFINET from 64 to 128, see Application 1 - increase in the maximum possible number of IO devices on PROFINET (Page 2149).
2. Fast and slow clock cycle for data to isochronous devices, see Application 2 - fast I/O processing and slower electrical drives on one module (Page 2150).
3. Distributed synchronous operation independent of isochronous devices, see Application case 3 - distributed synchronism over multiple devices and independent local systems (Page 2151).
4. The same project for the machine modules
5. Machine modules are individualized by assigning the IP address and NameOfStation on site, see Application 4 – Modular Machine: Independent IP address and NameOfStation for the iDevice and IO controller (Page 2152).
6. Two independent I-devices, see Application 5 - iDevice on both interfaces (Page 2154).

Application 1 - increase in the maximum possible number of IO devices on PROFINET

Description

The following functions are illustrated in the topology:

- Increased number of possible devices on SIMOTION D4x5 2 DP/PN by 2 subnets from 64 to up to 128.
- IRT and RT in each subnet

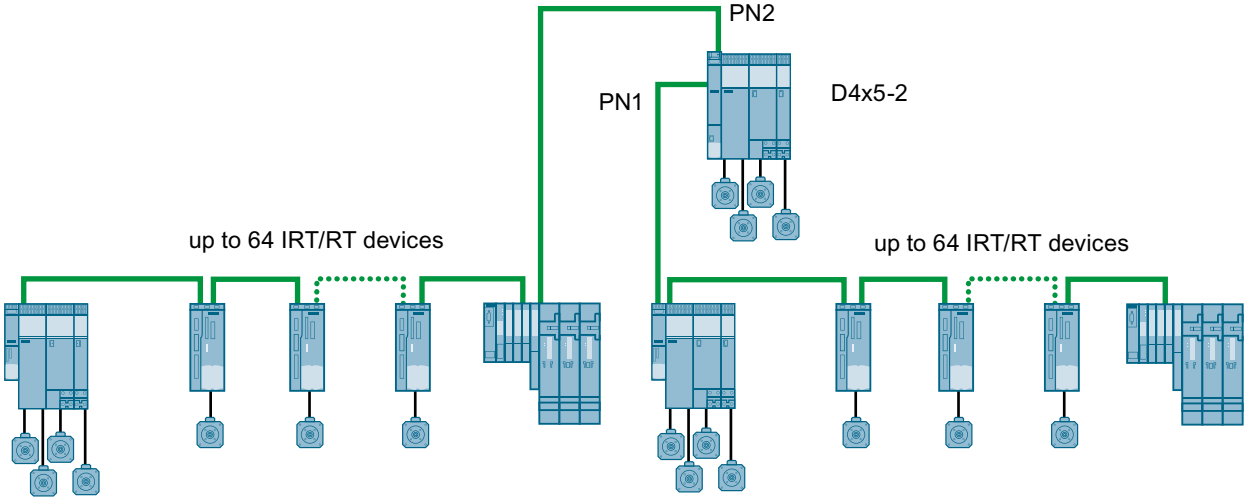


Figure 4-353 Application 1 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 2148)

Application 2 - fast I/O processing and slower electrical drives on one module

Description

The following functions are illustrated in the topology:

- Fast I/O application and slower drive functions in one controller.
- Control of slower SINAMICS drives via PN2 (e.g. servo clock of 1 ms).
- Fast I/O processing (e.g. ET200S High Speed) via PN1 (e.g. Servo_fast clock of 250 µs).

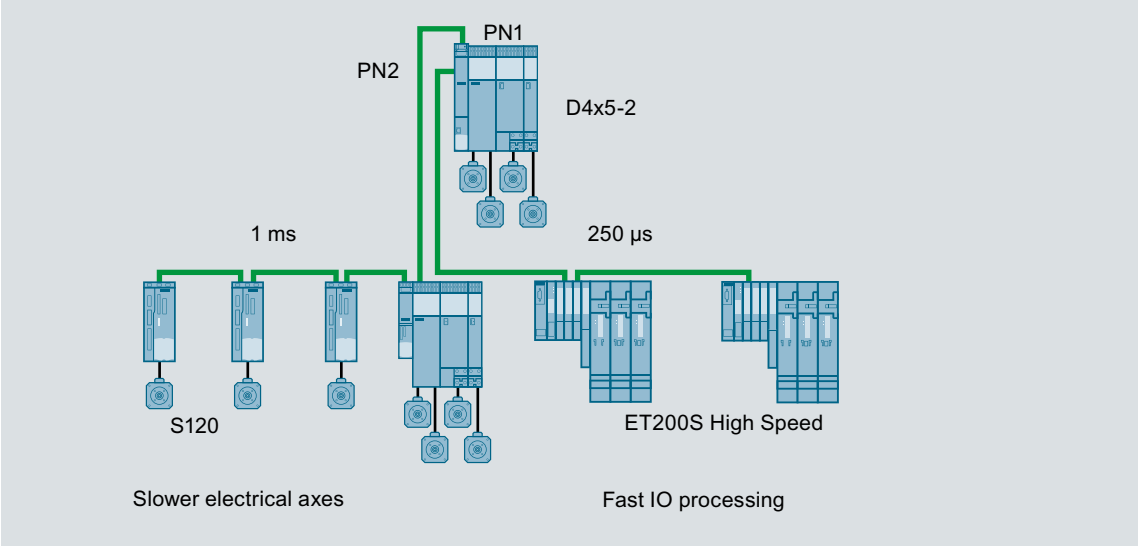


Figure 4-354 Application 2 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 2148)

Application case 3 - distributed synchronism over multiple devices and independent local systems

Description:

The following functions are illustrated in the topology:

- Distributed synchronous operation via controller-controller communication between the SIMOTION controllers of different machine modules in subnet 1.
- Local I/O on each SIMOTION controller of a machine module in separate subnet independent of subnet 1.
  - Independent adjustable communication parameters (send clock, IRT, IP address, NameOfStation, etc.), adapted to the technical requirements.
  - Addresses can be set identically to the other modules, thus facilitating simpler establishment of modular machine concepts
- Use of Servo and Servo_fast

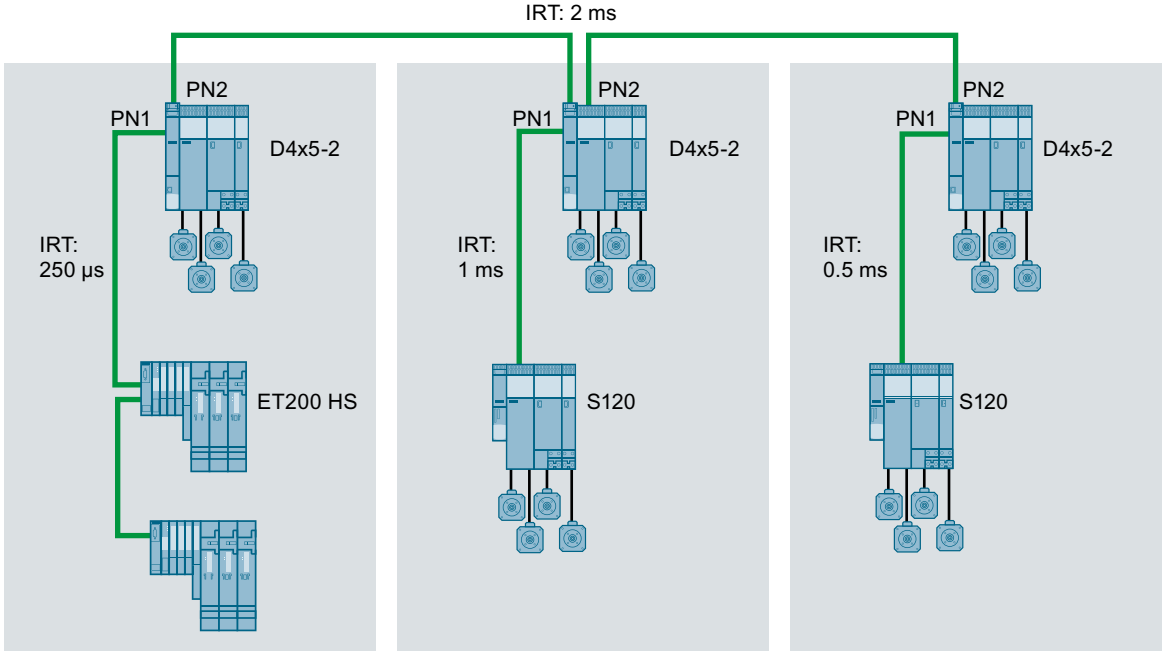


Figure 4-355 Application 3 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 2148)

Application 4 – Modular Machine: Independent IP address and NameOfStation for the iDevice and IO controller

Description

The following functions are illustrated in the topology:

- 2 PN interfaces in the SIMOTION controllers disconnect the subnet of the base machine from the machine modules.
- IP address and NameOfStation of the interface of PN2 can be granted independently of IP address and NameOfStation of the interface of PN1.

  Note

  The IP addresses of the two PN interfaces must be in different IP subnets.

- IP addresses and NameOfStation of the PN interfaces that are operated as an IO controller can be identical in the different machine modules.
- In the base machine project the SIMOTION controllers are added as I devices via a GSD file (I device substitute).

- CBE30-2 as IRT I device and onboard PN interface as IRT IO controller.
- Distributed synchronous operation between the base machine controller and the IRT I devices.
  - Isochronous coupling of the subnets, if PN axes
  - IRT communication on iDevice and controller interface

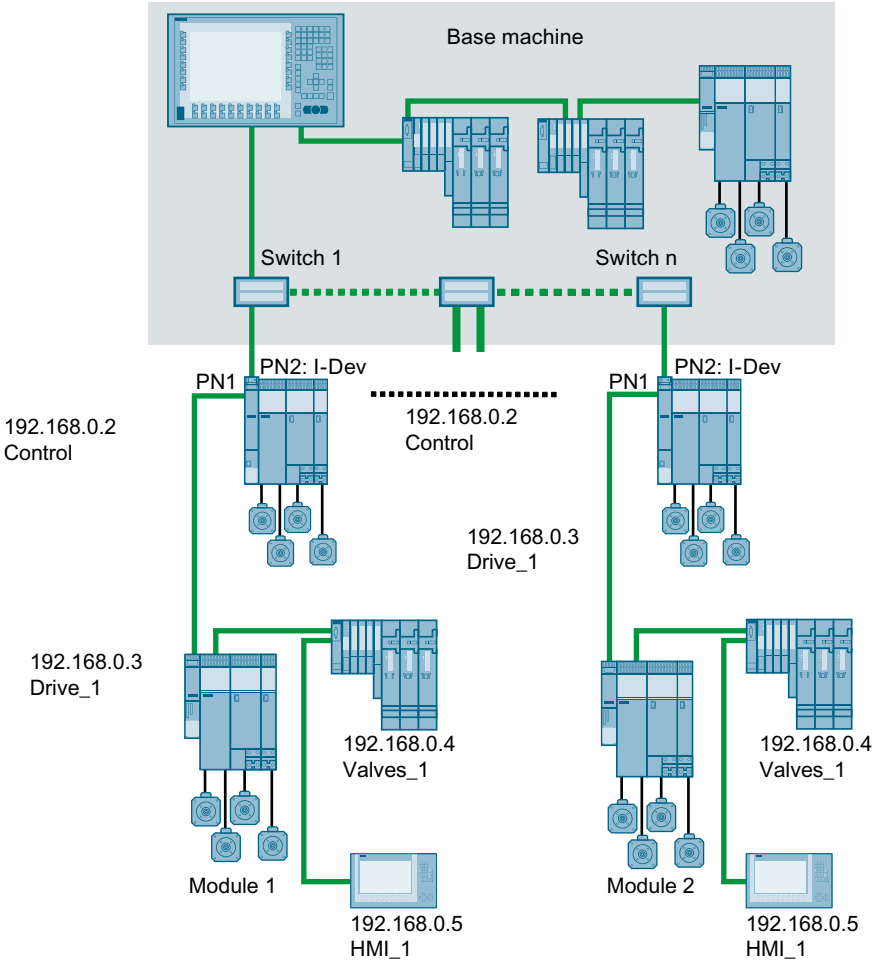


Figure 4-356 Application 4 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 2148)

Application 5 - iDevice on both interfaces

Description

The following functions are illustrated in the topology:

- Distributed synchronous operation between higher-level controller and lower-level SIMOTION controllers via CBE30-2 as an IRT I device
- RT communication between additional higher-level controllers and lower-level SIMOTION controllers via onboard PN interfaces as an RT I device (e.g. for PROFIsafe, diagnostics, process value recording). IRT communication is not possible here.

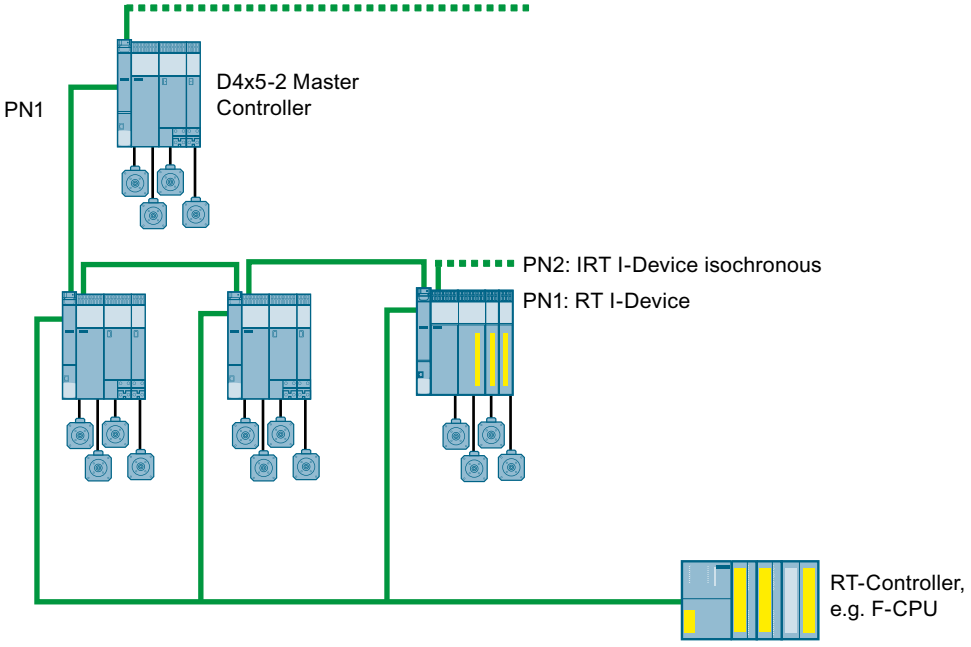


Figure 4-357 Application 5 for two PROFINET IO interfaces

PN1	Integrated PROFINET interface
PN2	CBE30-2

See also

Applications for two PN IO interfaces (Page 2148)

"Controlled" sync master

"Controlled" sync master

The "controlled" sync master synchronizes the PROFINET IRT cycle of the two PROFINET IO interfaces with the bus cycle of Servo/Servo_fast of the higher-level controller. The property "Controlled Sync Master" does not have to be explicitly configured. In the engineering, set "Sync Master" for this. The property "Controlled Sync Master" is automatically activated at the integrated PN interface if isochronous data has been configured on both PN interfaces, i.e. the send cycle clocks of both interfaces are synchronized to servo or Servo_fast.

A "controlled" sync master cannot be used together with a redundant sync master in a sync domain.

Application-specific synchronization is necessary analogously to PROFIBUS if the synchronization is lost and the system must be restarted.

Synchronization example with two PROFINET interfaces (IRT i device and IRT IO controller)

The following illustration shows the synchronization if the CBE30-2 is configured as an IRT I device for a higher-level controller and the onboard PN interface is configured as an IRT IO controller for the lower-level local peripherals.

Requirements for synchronization:

- CBE30-2 (X1400): Sync slave (synchronizes with the higher-level IO controller).
- Onboard PN interface (X150): "Controlled" sync master synchronizes the lower-level local peripherals.
- You can achieve application-specific synchronization with the _synchronizeDPInterfaces function, for example. See also *Clock generation and synchronization in PROFINET IO* in the Function Manual *Basic Functions for Modular Machines*.

  Note

  The application-specific synchronization can also be achieved in the StartupTask using the system function _enableDpInterfaceSynchronization.

- An example configuration can be found at Example configuration for "controlled" sync master (Page 2158).

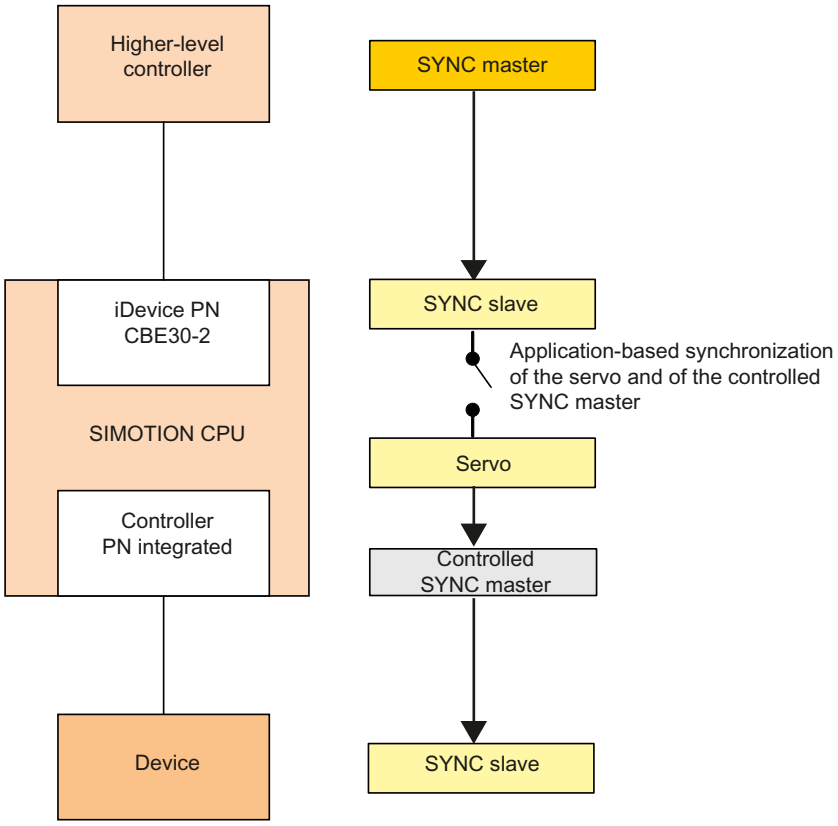


Figure 4-358 "Controlled" sync master

Synchronization example with two PROFINET interfaces (IRT IO controller on both interfaces)

The following illustration shows the synchronization if the CBE30-2 is configured as an IRT IO controller for direct controller-controller communication and the onboard PN interface is configured as an IRT IO controller for the lower-level local peripherals.

Requirements for synchronization:

- Onboard PN interface (X150): "Controlled" sync master synchronizes the lower-level local peripherals.
- CBE30-2 (X1400): Sync master (synchronizes the IO controller configured as a sync slave which is involved in direct data exchange) or sync slave (synchronizes with the IO controller configured as a sync master which is involved in direct data exchange).
- You can achieve application-specific synchronization with the _synchronizeDPInterfaces function, for example. See also *Clock generation and synchronization in PROFINET IO* in the Function Manual *Basic Functions for Modular Machines*.

Note

The application-specific synchronization can also be achieved in the StartupTask using the system function _enableDpInterfaceSynchronization.

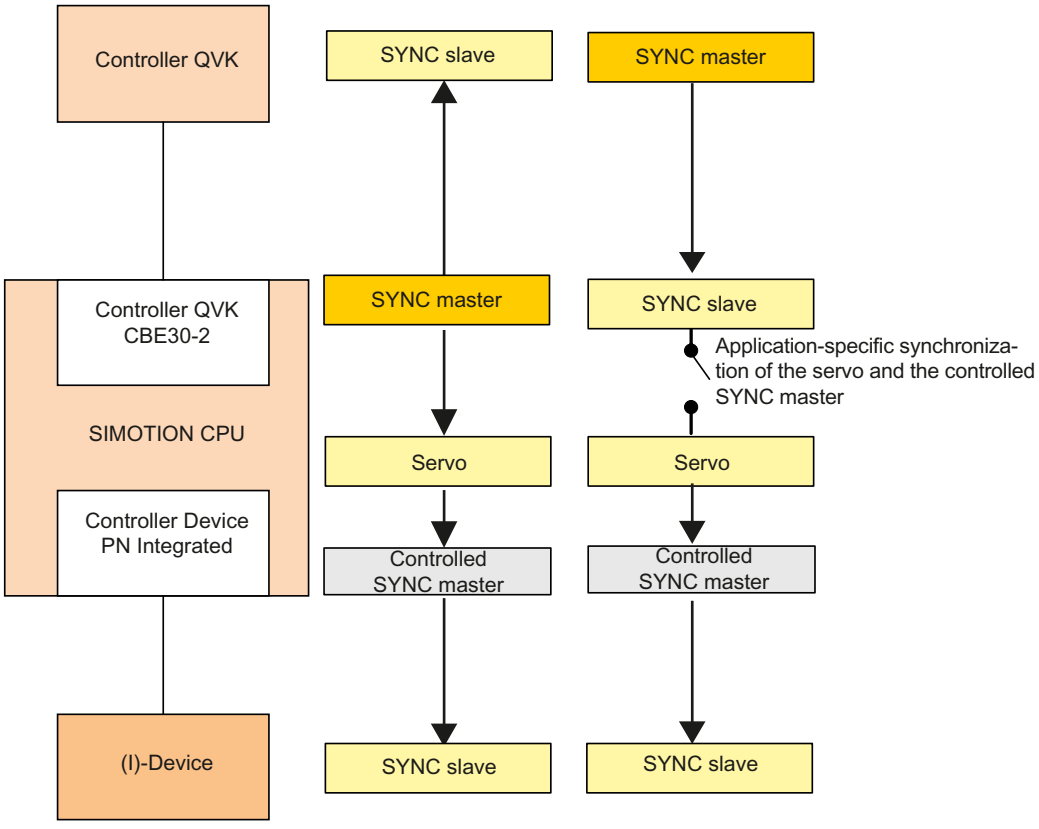


Figure 4-359 Controller direct data exchange with "controlled" sync master

Example configuration for "controlled" sync master

Distributed synchronous operation via two PROFINET interfaces

The example shows which application requires the use of two PROFINET interfaces for SIMOTION controllers. The PROFINET network "1" is configured as the MRPD ring for the distributed synchronous operation. The SIMOTION controller which has been configured as a sync master specifies the bus cycle clock as well as the servo and IPO cycle clock. The two other SIMOTION controllers are integrated into the MRPD ring as sync slaves via the CBE30-2 and synchronize with the sync master. Through their onboard PROFINET interface, they each set up their own PROFINET network and transmit the bus cycle clock as a "controlled" sync master to the lower-level local periphery (SINAMICS drives).

See also "Controlled" sync master (Page 2155) and Two PROFINET IO interfaces (Page 2144).

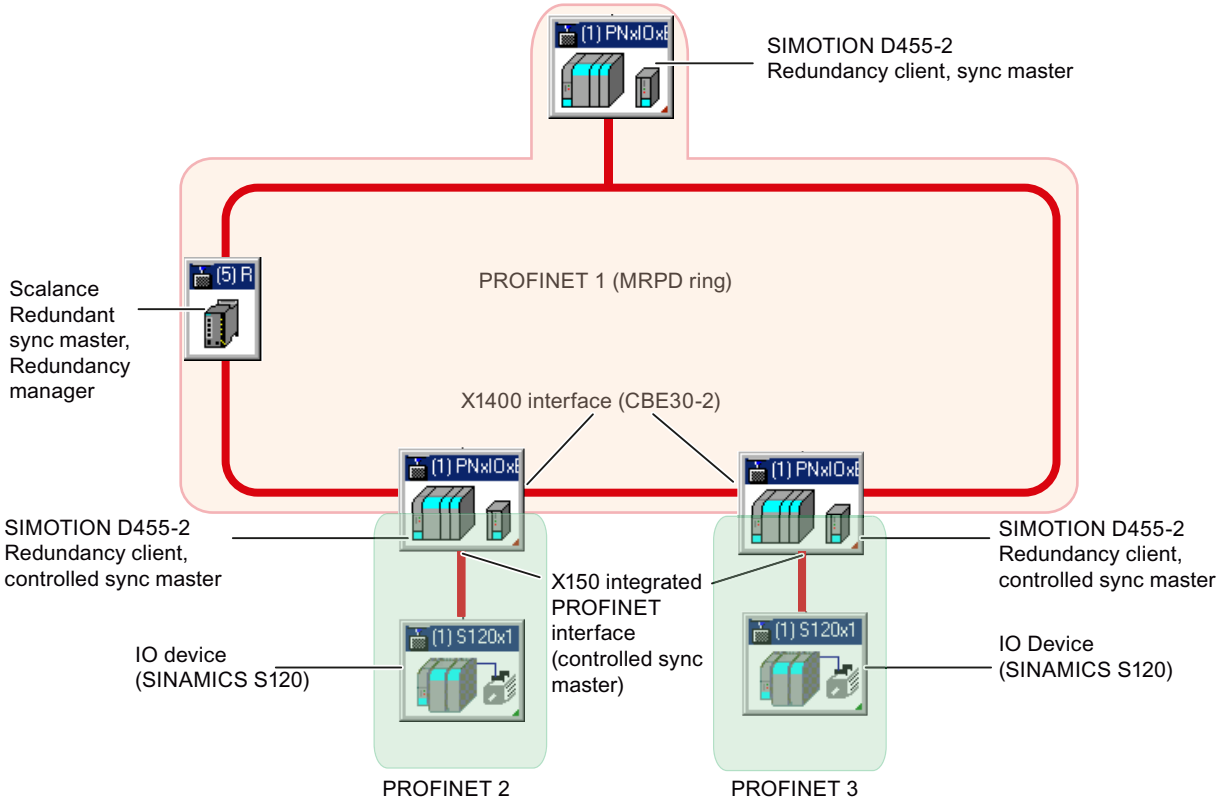


Figure 4-360 Example configuration for controlled sync master

If several sync domains are operated on a physical Ethernet, you should ensure that:

- either all sync domains have different names
- or configure this explicitly as the end of the sync domain at the ports that form the end of the sync domain.
  To do this click on the port that is to be the end of the sync domain and select **Properties** from the context menu. Choose the **Options** tab from the properties dialog box that appears. Activate the option **End of sync domain** and confirm with **OK**.

Relationship between sync domain and PROFINET IO systems

Introduction

The devices of several PROFINET IO systems can be synchronized by a single sync master, provided they are connected to the same Ethernet subnet and belong to a sync domain. Conversely, a PROFINET IO system may only belong to a single sync domain.

Devices with two PROFINET interfaces

The two PROFINET interfaces must belong to different sync domains. Synchronization of the two sync domains then takes place via a "controlled" sync master, see also "Controlled" sync master (Page 2155).

To find out how to create a second sync domain or change the name of an existing sync domain, go to Creating a sync domain (Page 2199).

Redundant sync master

Description

There are machines/plants that consist of two submachines / plant units. For such machines/ plants (complete systems), it may be that both submachines / plant units (subsystems) must be able to be operated synchronously, alone (separated) and in the complete system. This requires that a sync master is defined/configured on each subsystem. Only one sync master is permitted in the complete system. Consequently, a sync master is configured on one subsystem and a redundant sync master is configured on the other subsystem.

If the complete system is operated combined, either the sync master or the redundant sync master handles the synchronization of the complete system (depending on the switch-on sequence; for simultaneous switch-on, the sync master).

Applications

- Complete system (machine) with two subsystems (submachines)
  If it is necessary that a machine/plant consists of two subsystems and the complete system, as well as the two subsystems, must be operated synchronously independently of one another. See also Media redundancy for SIMOTION (Page 2166).
  Examples for the use of subsystems that are operated separately:
  - Part commissioning of the complete system.
  - Failure of the sync master and the first subsystem is permitted, but the second subsystem must continue to run on its own.
  - Shutdown of a subsystem by the user because it is not required for a process or is being serviced.
- Redundant sync master in the case of MRPD
  A redundant sync master is also possible in the case of a ring topology with media redundancy, see also Sample configurations for MRPD rings (Page 2171) and Information on PROFINET with MRPD (Page 2170). For this, an additional SCALANCE switch must be configured in the ring as a redundant sync master.

Special tandem machine application

An application example is a tandem configuration of a printing machine. For this special application, note the FAQ in the Service & Support portal (https://support.industry.siemens.com/cs/ww/en/view/109480700).

Limitations of use

- There can physically be up to a maximum of two additional modules between two sync masters.
- If the transmission link between the sync master and redundant sync master fails, leaving two subnets with one sync master each, both subnets remain synchronized with the remaining sync master in each case. This results in two independent synchronized subnets that drift apart due, among other things, to the temperature drift of the quartzes.
- Once the transmission link has been reestablished, the redundant sync master cannot synchronize with the sync master. The redundant sync master must be switched off and on again (power OFF/ON) before the synchronization can take place.
- If the plant section with the sync master fails as the result of a fault or shutdown, the other plant section with the redundant sync master will continue operating independently. The sync slaves assigned to the sync master will lose their synchronization and cease to operate.
- When both PN-IO interfaces operate isochronously, no redundant sync master is permitted in the sync domain for PN-IO.

Configuring the redundant sync master

1. Add a second SIMOTION module or a SCALANCE switch (if MRPD) and configure the PROFINET communication in accordance with your requirements.
2. Right-click on the PROFINET interface to open the **Properties - <PROFINET interface> -- (R0/S2.6)** dialog.
3. Select the **Sync master (redundant)** entry below **Synchronization role** on the **Synchronization** tab.

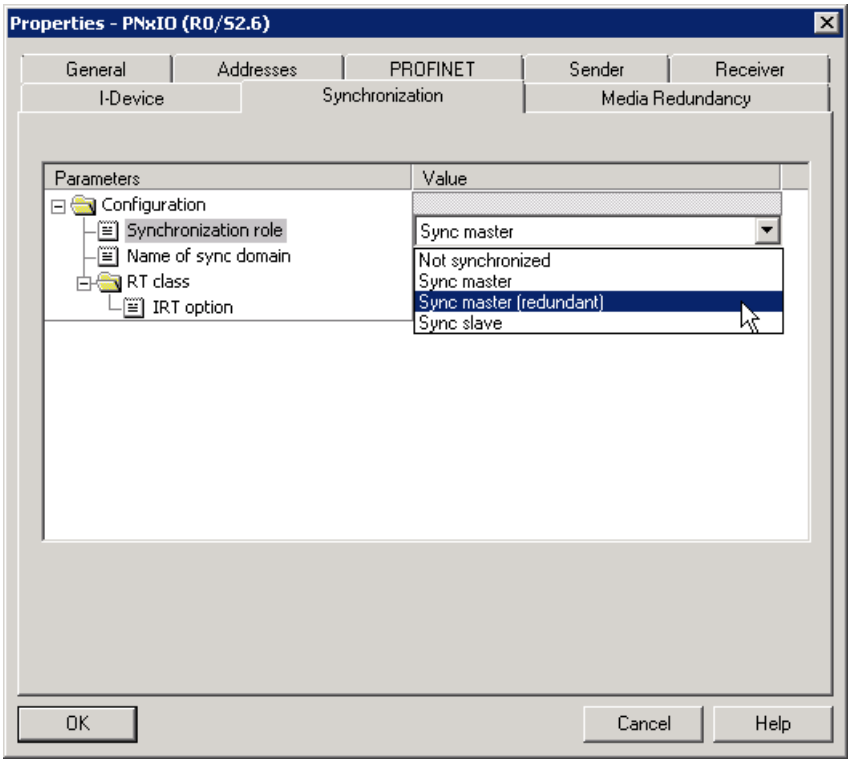


Figure 4-361 Configuring the redundant sync master

Deployment of a redundant sync master using the example of a machine with two machine parts.

The switchover behavior of the sync master and the redundant sync master is subject to several rules That must observed during the project creation and for applicative solutions.

- The topological design of the machine should provide the connection between the two machine parts via the sync master and the redundant sync master.
- If possible, no further device should be present between the sync master and the redundant sync master (e.g. IRT switch).

Supplementary conditions

- A wait time of 60 seconds must be observed for switching from the sync master to the redundant sync master. Only after this interval is it guaranteed that the redundant sync master has assumed its final status rather than the sync master resuming its role after a return.
- If the redundant sync master is switched on after the sync master, all sync slaves remain synchronized to the sync master.
- If the redundant sync master is switched on first and then the sync master after a wait time (> 60 seconds), all sync slaves remain synchronized to the redundant sync master.
- If both sync masters are switched on simultaneously, all sync slaves are synchronized to the sync master (this behavior is guaranteed only for an approximately equal startup time).
- If the connection between the sync master and the redundant sync master is interrupted, both sync masters accept their appropriate role and the topologically assigned sync slaves are synchronized accordingly.
- If the sync master and the redundant sync master are switched on simultaneously without a direct connection, the connection will be restored after a time > 60 seconds. The sync slaves remain synchronized to the sync master or the redundant sync master with which they are connected topologically. The two sync domains are not joined automatically. A diagnostic message will be generated.
- If the sync master is switched off, the redundant sync master handles the synchronization of the topologically-linked sync slaves. The sync slaves remain synchronized, i.e. no failure occurs.
- If the redundant sync master is switched off, the sync slaves connected topologically with the sync master remain synchronized.
- If the redundant sync master and sync master are separated and the redundant sync master has taken over the sync slaves, a combination (operation of the complete system) is only possible again after the complete machine has been shut down.

Quantity structures

The following maximum values apply to IO controllers of the SIMOTION platform.

A maximum of 64 communication relationships are possible; these can be divided up as follows:

- Connection of up to 64 IO devices.
- Maximum 64 RT IO devices.
- Maximum 64 IRT IO devices (High Performance).
- A maximum of up to 64 controller-controller data exchange broadcast relationships may be set up between SIMOTION controllers.
- If a SIMOTION controller has been configured as an I-Device, it also counts as an IO device; in other words, only another 63 IO devices can be connected to this controller.
- A SIMOTION controller can receive data from up to 64 other SIMOTION controllers as part of controller-controller data exchange broadcast yet can send data to any number of SIMOTION controllers.

- The IRT telegrams in controller-controller data exchange broadcast are broadcast telegrams. This is why any number of SIMOTION controllers can access the data of one SIMOTION controller.
- In controller-controller data exchange broadcast, the data volume also has to be considered. A data exchange broadcast only counts as a connection up to a certain data volume. Where a second telegram is required, data exchange broadcast does not require two connections.

Note

For MRPD, halve the quantity structures, since an MRPD IO device requires the resources of two IO devices. The same applies to controller-controller data exchange broadcast.

The IO area doubles if two PROFINET IO interfaces are used.

Mixed operation of IO devices and controller-controller data exchange broadcast

You can calculate the possible number of devices in mixed operation using the following formula:

```
RT + IRT High Performance IO device + data exchange broadcast frames
<= 64
```

Note

In a data exchange broadcast relationship, it is not the number of configured slots (lines on the **Receiver** tab - see Configuring the receiver (Page 2237)) that is intended for IRT High Performance direct communication configuration, but rather the number of Ethernet frames received for the data exchange broadcast. An Ethernet frame can contain up to 768 bytes of data exchange broadcast net data.

One slot has up to 768 bytes and 3,072 bytes of net data can be exchanged during data exchange broadcast (divided into 4 frames of 768 bytes each). The value will depend on the sizes of the slot chosen. This means a transmitter can receive more than one slot, although these must be transmitted within a single telegram.

Each sender sends its data exchange broadcast data in an Ethernet frame. Every other SIMOTION controller can read the data in this frame. This results in a counting connection to each transmitting SIMOTION controller.

For data exchange between an IO controller and an IO device, a frame has a useful data size of 1440 bytes for each transmitting SIMOTION controller.

During the compilation of the project, HW Config verifies the configured quality structure based on the formulas mentioned above.

Address space

A maximum of 4 KB each may be assigned for PROFINET IO data for the input and output data in the logical address space of an IO controller. The rest of the 16 KB large address space can be used, for example, for PROFIBUS data or diagnostic data.

Address space

A maximum of 4 KB each may be assigned for PROFINET IO data for the input and output data in the logical address space (IO address data) of an IO controller. As of SIMOTION V4.5, up to 6 KB can be used for SIMOTION D controllers. The rest of the 16 KB large address space can be used, for example, for PROFIBUS data or diagnostic data.

Definitions:

- IO address space:
  Logical addresses can be assigned in STEP 7 within this space.
- IO address data:
  Address space with assigned data.
- All logical addresses, i.e. diagnostic addresses as well, are included in the IO address space or the IO address data.
- The IO address data includes all logical addresses belonging to the IO system, i.e. I-Devices, slave-to-slave communication, and diagnostic addresses as well in the case of PROFINET (port submodules, for example).
- Only a check of the IO address data across all IO systems/DP systems is implemented in STEP 7 itself (including IO data + diagnostic addresses). The bus-specific check is carried out in the SIMOTION CPU-OM.

Example:

A 4-byte output module occupies 4 bytes of IO address data and can be assigned a logical address at any (free) location within the IO address space, e.g. 14000.

Size of the logical address space

	C240 / C240 PN	P3xx PN / P3xx DP/PN	D410-2 DP/PN	D4x5-2 DP/PN
CPU total [bytes]	4096	4096	16834	16834

Maximum quantity of IO address data

The maximum quantity of IO address data has been increased as of V4.5 from the current 4096 bytes to 6144 bytes for all PN/IO interfaces of SIMOTION D.

Segment	C240 / C240 PN	P3xx PN / P3xx DP/PN	D410-2 DP/PN	D4x5-2 DP/PN
DP ext. (per interface) [bytes]	1024	1024	1024	1024
DP_integrated [bytes]	-	-	512	4096
PROFINET (per interface) [bytes]	4096	4096	6144	6144

For each input and output per bus segment (DP system / IO system)

Submodule size from SIMOTION V4.4

In PROFINET, data for devices (and I-devices) and the controller-controller data exchange broadcast is structured in the form of submodules. Only data from a submodule can be read or written consistently when "non-synchronized" with PROFINET RT or IRT. The max. submodule size of SIMOTION controllers for PROFINET interfaces has been increased for the I-device and for the IO device from 254 bytes to 1024 bytes. The size of the submodule depends on the hardware and the PROFINET interface.

Maximum submodule size and consistency range

The following table defines the maximum submodule size, depending no the SIMOTION controller and the PROFINET interface. The submodule size stated applies to operation of a PROFINET interface with the following function:

- IO controller (the IO controller can operate IO devices with submodules up to the specified size)
- I-device (on the local I-device, submodules up to the specified size can be set up)

SIMOTION module	Onboard PROFINET interface [bytes]	CBE30-2 [bytes]
D455-2 DP/PN	1024	254
D445-2 DP/PN	1024	254
D435-2 DP/PN	1024	254
D425-2 DP/PN	1024	254
D410-2 DP/PN	1024	-
C240 PN	254	-
P320-4 PN	254	-
P320-4 DP/PN	254	-

Note

Controller-Controller data exchange broadcast

For controller-controller data exchange broadcast, the max. submodule size of 254 bytes is retained.

Rule for use of a SIMOTION or SIMATIC I-device on a SIMOTION IO controller

If a SIMOTION controller is configured as an IO controller and a SIMOTION or SIMATIC controller is configured as an I-Device in a project, submodules larger than 254 bytes can be used subject to the following constraints:

- The I-device must support submodules up to 1024 bytes.
- The IO controller must support submodules up to 1024 bytes.

Note

For SIMATIC controllers, a max. submodule size of 1024 bytes applies.

Potential problems when using interfaces with different consistency lengths

If, for example, you configure an I-Device that contains a submodule of 500 bytes in size on the onboard PROFINET interface of a SIMOTION D4x5-2, the I-Device, for example, cannot be operated on a SIMOTION C240 PN that is a higher-level controller. An error message is displayed in STEP 7, stating that the output/input net data length of 254 bytes has been exceeded.

See also

Data exchange through the use of iDevices (Page 2258)

Media redundancy (MRP and MRPD)

Media redundancy for SIMOTION

Media Redundancy Protocol

It is possible to establish redundant networks via the so-called Media Redundancy Protocol (MRP). Redundant transmission links (ring topology) ensure that an alternative communication path is made available when a transmission link fails. The PROFINET devices that are part of this redundant network form an MRP domain.

Note

Devices that support MRP

An overview of all devices that support media and/or system redundancy can be found in the Support area (https://support.industry.siemens.com/cs/ww/en/view/67364686).

MRP (Media Redundancy Protocol for PROFINET RT, as of SIMOTION V4.3)

MRP guarantees media redundancy in the event of a problem in the ring. The switchover of the ring is performed by the Redundancy Manager.

The switchover times depend on the following parameters:

- The actual topology
- The devices used and
- The network load in the relevant network.

The typical reconfiguration time of the communication paths for TCP/IP and RT frames in the event of a fault is < 200 ms.

In most systems, the switchover time of MRP is far above the PROFINET update time for cyclic data, so that a failure for cyclic data is detected. The PROFINET connection therefore fails and is re-established after the switchover by the Redundancy Manager. In this way, an error can be corrected in the network, while the system continues to run *with bumps*.

The figures below show a sample topology with and without fault:

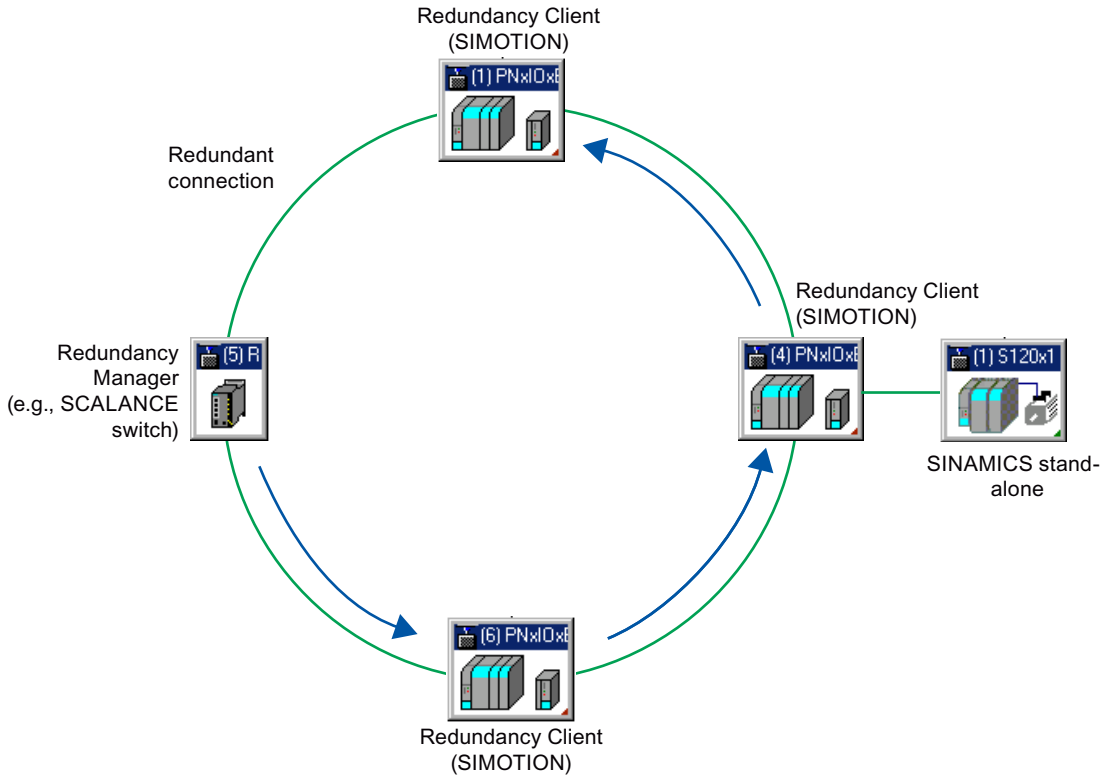


Figure 4-362 Ring topology for media redundancy without fault

After the occurrence of a fault, the Redundancy Manager reconfigures the communication of the ring. In contrast with MRPD, the PROFINET RT telegrams are only sent in one direction in the ring with MRP. Therefore, the redundant communication path is only closed by the redundancy manager in the event of a fault.

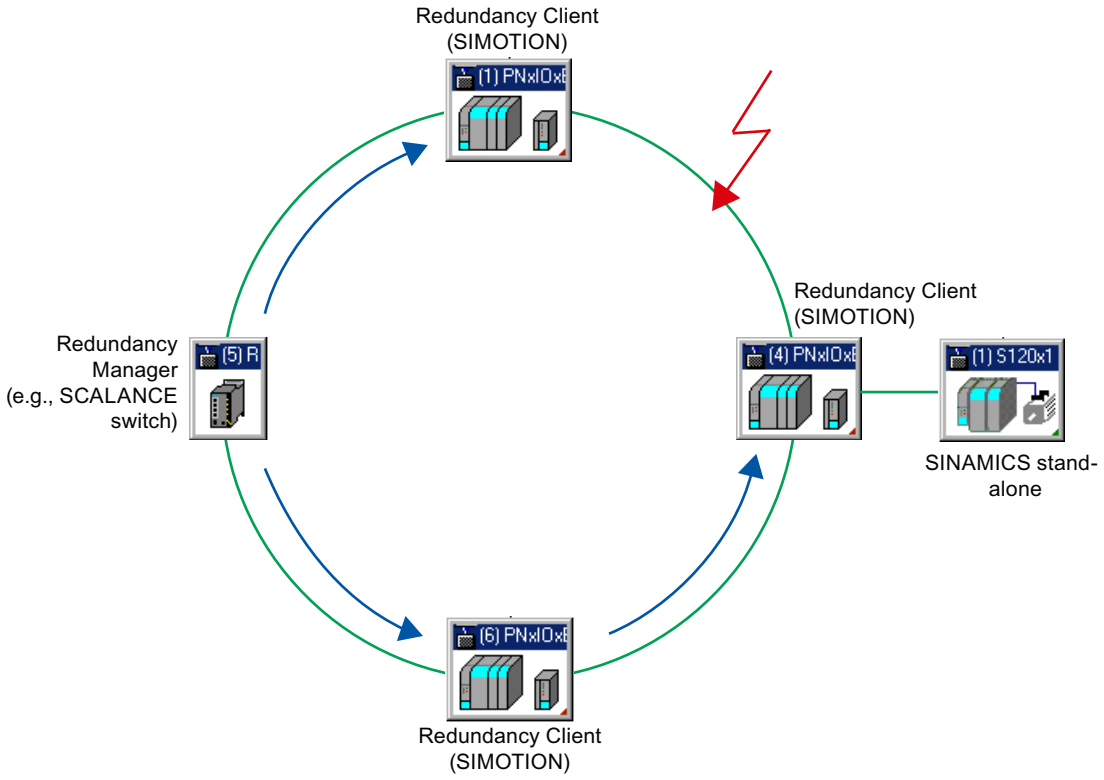


Figure 4-363 Ring topology for media redundancy with a fault

Ring ports

A SIMOTION controller / SINAMICS drive may only be inserted into an MRP ring as a node with MRP-capable ports. For SIMOTION D, the first two ports of the PROFINET IO interfaces are designed as ring ports.

These two ports are marked with an "R" in the module rack in HW Config.

Note

Only devices with MRP-capable ports may be inserted in an MRP ring. If MRP-capable ports are not used, the reconfiguration times can be in the multi-digit seconds range.

For SINAMICS S120 drives (SINAMICS S120 CU320-2/CU310-2), these are ports P1 and P2. However, they are not separately designated as ring ports in the HW Config.

MRPD (Media Redundancy for Planned Duplication) for PROFINET IRT, as of SIMOTION V4.3, SIMOTION SCOUT TIA V4.5

MRPD (Media Redundancy for Planned Duplication) is a method for smooth media redundancy with PROFINET IRT. MRPD also requires MRP.

The combination of MRP with MRPD provides smooth PROFINET operation for short cycle times in the event of a fault in the ring. MRPD is based on IRT and ensures bumpless operation by the sender sending the cyclic data in both directions which the recipient then receives twice. The

first received telegram is used by the recipient. The second telegram is discarded automatically. If the ring is interrupted at one position (e.g., through the failure of a ring node), receipt of the cyclic data via the uninterrupted side of the ring is still guaranteed. The following figure shows a sample configuration:

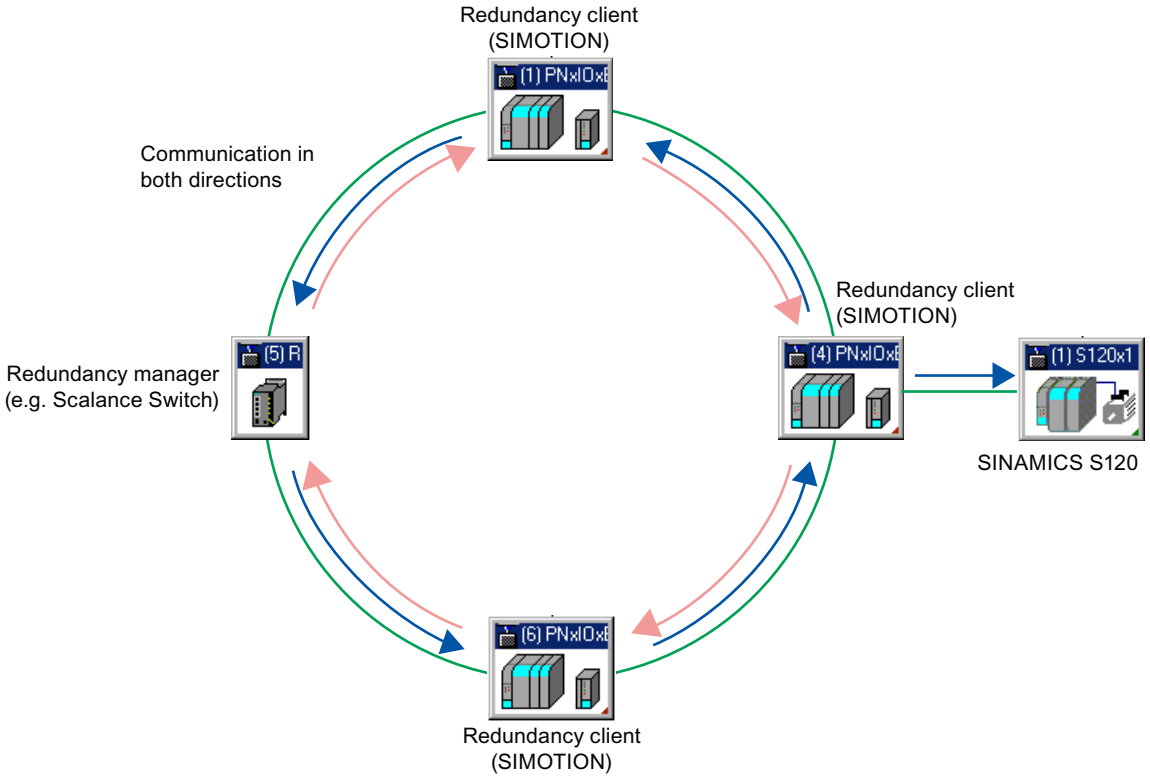


Figure 4-364 Ring communication in MRPD

Smooth media redundancy MRPD always requires the activation of MRP in the individual rings.

Note

Too high a network load or too rapid incoming/outgoing faults caused by delayed or continuous switchovers of the MRP, may lead to failure of the PROFINET connection even with activated MRPD.

Devices that support MRPD

Note

An overview of all devices that support media and/or system redundancy can be found in the Support area (https://support.industry.siemens.com/cs/ww/en/view/67364686).

If the device prevents MRPD, an **error message** is issued in HW Config:

- "The <xxxx> device does not support sync domain with MRPD area."

SIMATIC controllers, such as a SIMATIC 300 or an ET200, currently do not support MRPD.

See also

Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 2207)

Redundant sync master (Page 2159)

Information on PROFINET with MRPD

Smoothness of MRPD

Delayed or incomplete switchovers of MRP/MRPD caused, for example, by a network load that is too high or the too rapid incoming/outgoing of faults, under unfavorable conditions, can lead to the PROFINET connection failing even with activated MRPD.

For example, in the event of two consecutive faults at various parts of the ring, smooth operation is only ensured when the two faults are separated by approx. three seconds (if a fault has "gone", an additional fault may "come" only after a short pause).

Operation of MRPD in conjunction with a redundant sync master

There can be a maximum of two Ethernet nodes between one sync master and one redundant sync master.

If the redundant sync master is used together with MRPD, it is recommended that the redundant sync master (e.g. a SCALANCE IRT switch) be connected directly to the sync master and the two nodes positioned in a common switch cabinet so that the cable connection between both nodes is protected.

The insertion or withdrawal of a connection in an MRPD ring does not have any effect on the synchronization. Only one sync master remains.

In case of an interruption in the distance between sync master and redundant sync master, the system initially continues to run smoothly, however faults may occur when you switch off and then restart the system if the devices between the sync master and the redundant sync master have too great a difference in ramp-up time.

The fault is that the sync domain breaks down into two sub-domains, and one part of the sync slave synchronizes to the sync master, and the other part synchronizes to the redundant sync master. The PN diagnosis "Remote mismatch / Peer PTCP mismatch 0x8008" is present.

If the sync master or the redundant sync master in an MRPD ring is switched off, all sync slaves remain synchronized, because unlike the line, all sync slaves retain their connection to the remaining sync master.

Configuration instructions

You must comply with the following rules to operate PROFINET IRT in connection with MRPD:

- The rules for configuring MRP and PROFINET IRT must be followed.
- All devices in the ring must support MRPD.

- If devices at the stub are in redundant communication via the ring, all devices at the stub must support up to the last MRPD node of the MRPD stub.
- Nodes in the stub that only communicate locally in the stub and not over the ring, do not have to support MRPD. They do, however, require the IRT property "StartupMode=Advanced" so you can operate them in a sync domain together with MRPD nodes. In V4.3, the devices described in Media redundancy for SIMOTION (Page 2166) have this IRT property.

See also

Media redundancy for SIMOTION (Page 2166)

Sample configurations for MRPD rings

MRPD configuration with a ring from SCALANCE switches

The ring consists of SCALANCE switches (redundancy manager and redundancy clients), SIMOTION controllers and IO devices at the stubs. In this configuration, several IO controllers can fail or be deactivated with their IO devices. The rest of the system continues to run smoothly. In addition, it is also possible for a SCALANCE switch to fail, or for the ring to be interrupted.

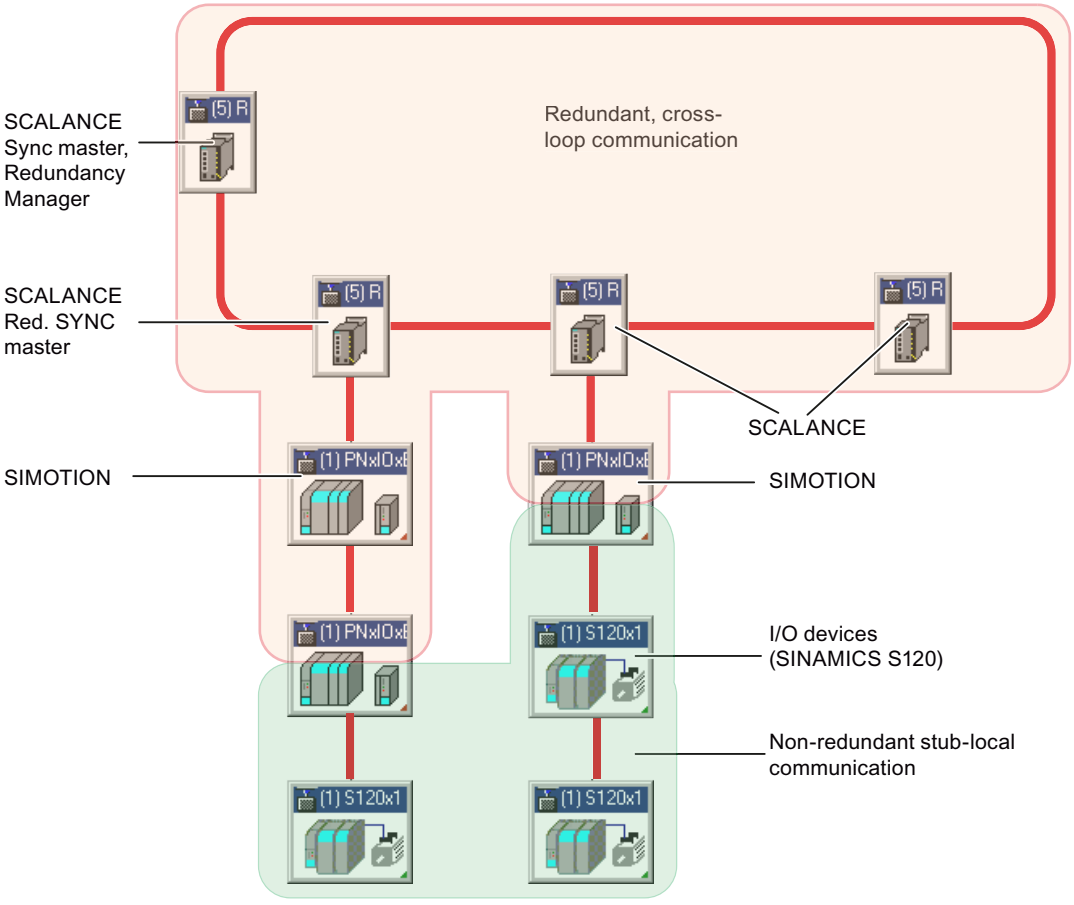


Figure 4-365 Ring configuration with SCALANCES

MRPD configuration with a ring from SIMOTION controllers and a SCALANCE switch

The ring contains a SCALANCE switch as a redundancy manager and SIMOTION controllers as redundancy clients. The SIMOTION controllers with their IO devices are connected with the ring via spur lines.

In this configuration, an IO controller can fail or can be deactivated with its IO devices (at the stub). The rest of the system continues to run smoothly.

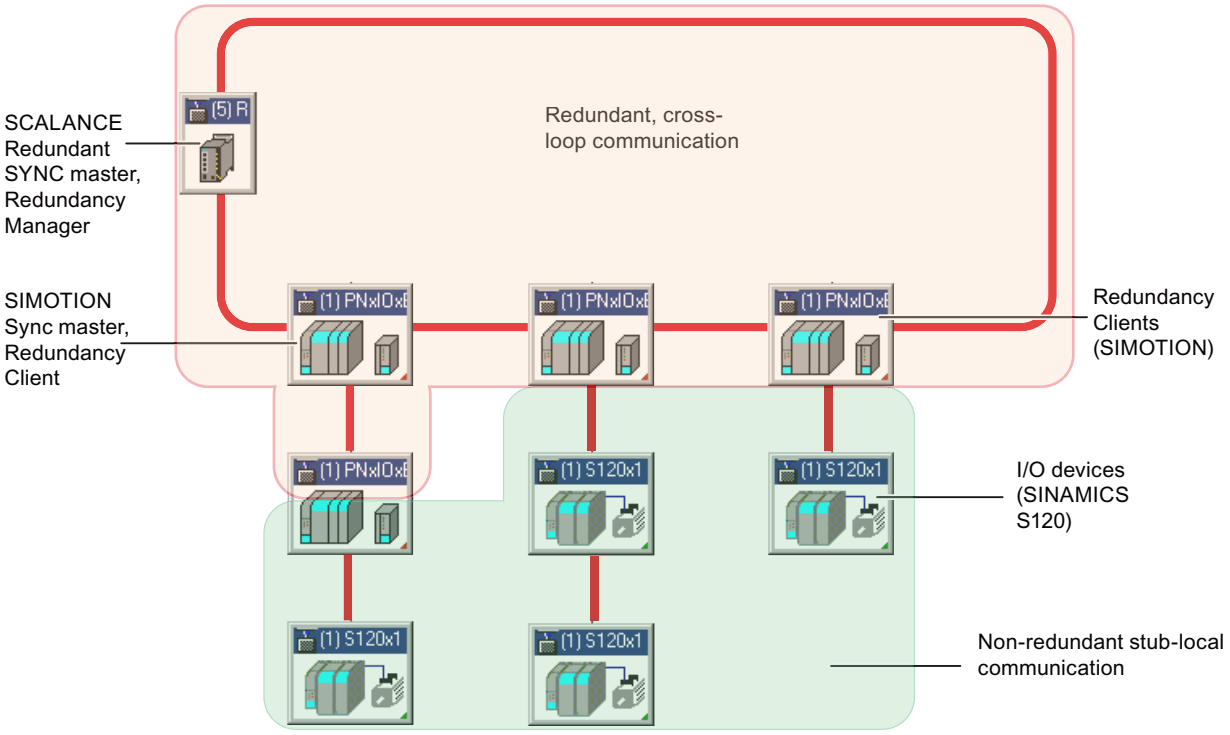


Figure 4-366 Ring configuration with SCALANCE and SIMOTION

MRPD configuration with SCALANCE, SIMOTION controller and SINAMICS I/O devices

The ring contains a SCALANCE switch, a SIMOTION controller and SINAMICS IO devices. An IO device can fail or can be deactivated. The rest of the system continues to run smoothly.

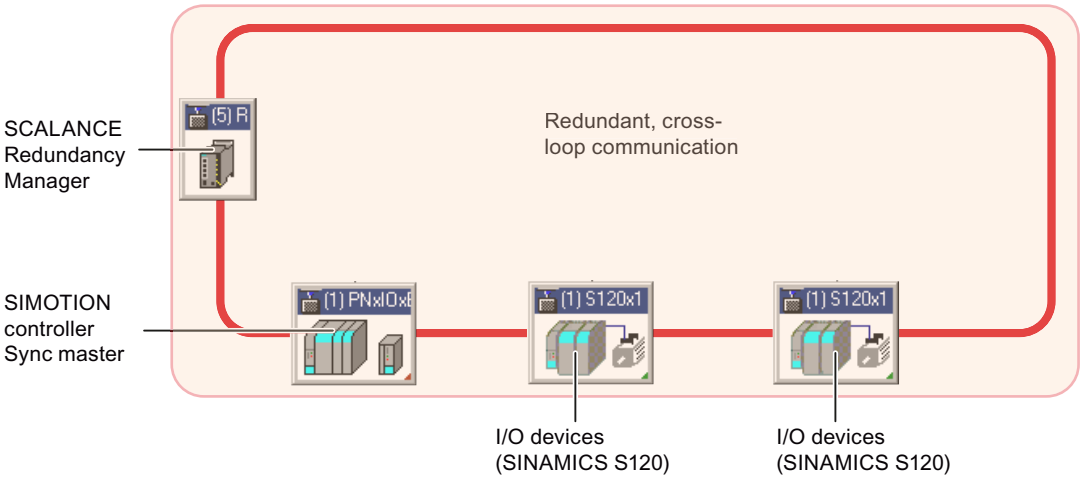


Figure 4-367 Ring configuration with SINAMICS I/O devices

Hierarchical multi-ring system

If, for example, the number of stations per ring or cycle times is exceeded, you can establish hierarchical multi-ring systems and divide the functionality into sub-rings. The sub-rings can continue to operate independently of each other.

For an example, see also Redundant sync master (Page 2159).

Use of Safety with MRRT

No smoothness for Safety via RT and shared device

MRRT offers no smooth redundancy for RT data (but only for IRT data).

In order to guarantee smooth media redundancy with Safety, the safety data within the MRRT ring must be transferred as IRT data. To make sure that this is the case, the SIMOTION controller F-proxy must be used for communication with the F-CPU. Please note that a direct connection must be established between the F-CPU and the SIMOTION controller. Either a free port or the second PROFINET interface, which is not part of the MRPD ring, can be used for this purpose. The

Safety RT data of the F-CPU is then transferred internally from the SIMOTION controller into the IRT data of the MRPD ring.

Note

The use of the shared device (the F-CPU communicates directly with the SINAMICS drive) is not recommended for MRPD, because the RT data deactivates temporarily in the event of a fault and so the PROFIsafe modules are passivated. Use of the I-device F-Proxy is recommended, because the PROFIsafe data is packed together in the IRT data.

Safety via I-device F-Proxy

When using the I-device F-Proxy, the safety data is transmitted from the F-Proxy to the drives in the MRPD ring via PROFINET IRT. The PROFIsafe communication between the F-CPU and the drives is thus also secured in case of a fault with the ring.

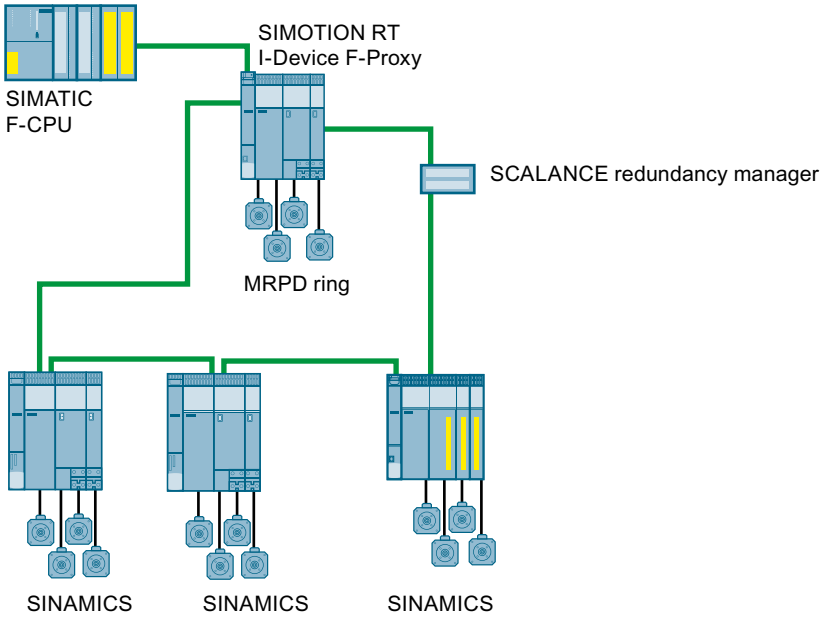


Figure 4-368 Safety via MRRT

In order for the safety data to be transferred smoothly, the communication paths between the F-CPU and the SIMOTION I-device F-Proxy may not go via the ring. As seen in the illustration above, the F-CPU should either be connected directly with an open port of the PN interface of the SIMOTION controller, which has been configured as an I-device F-Proxy, or the second PN interface be used as an I-device F-Proxy. For an exact description of how to configure an I-device F-proxy, please refer to section Principles of I device failsafe proxy (Page 2338).

You can find a description of how to configure Safety via PROFINET with an F-CPU in the FAQ (https://support.industry.siemens.com/cs/ww/en/view/50207350).

See also

Two PROFINET IO interfaces (Page 2144)

PROFIsafe via PROFINET with an F-CPU (Page 2344)

PROFINET V2.3 Performance Upgrade

Optimized data transfer / 125 µs PROFINET send clock

PROFINET V2.3 Performance Upgrade

All SIMOTION Control Units support PROFINET in accordance with the PROFINET International specification PN V2.3. The optional **Performance Upgrade** functionality that provides extended functional scope is available for SIMOTION D455-2 DP/PN.

- Minimum PROFINET send cycle clock 125 µs (Fast Send Clock)
- Optimization of data transfer (e.g. by Dynamic Frame Packing - DFP (Page 2176))
- Fast forwarding
- Fragmentation

Note

Validity

The PROFINET Performance Upgrade (Fast Send Clock 125 µs and Dynamics Frame Packing) is only approved for SIMOTION SCOUT TIA.

Note

You will find further information on this on the web site of the PI Organization (http://www.profibus.com) and the following technical specifications:

- Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.722
- Application Layer services for distributed I/O and distributed automation, Technical Specification for PROFINET, Version 2.3 – Date: October 2010, Order No.: 2.712

With optimized data transfer, the following options are possible:

- More devices can be operated with the same cycle time.
- The cycle time can be reduced for the same number of devices.

Requirements

General conditions for using the 125 µs send cycle:

- The 125 µs send cycle is supported only by the onboard PROFINET interface X150 in conjunction with Servo_fast / IPO_fast.
- Only one port can be used on X150; the other two ports must therefore be deactivated in the interface properties of X150 (the ports cannot be used for TCP/IP or UDP communication either).
- X150 can only be used as a PROFINET IO controller (not as an I-Device).
- No MRP/MRPD media redundancy (only one port is available on X150).
- Use of STEP 7 V14, V14; for other operating conditions, see also *Readme* for SIMOTION SCOUT V4.5.

Note

Quantity structures in the case of very short send cycles

When very short send cycles (125 µs, 250 µs) are used in a configuration with a large number of PROFINET devices, SIMOTION will be operating close to its upper performance limits. It is not possible to reliably predict all the restrictions applicable to the configuring of such systems. Only during runtime will it be possible to identify overload situations. Overload conditions will be indicated by appropriate messages (cycle violation, inconsistent data transmission). When you are using these very short cycle times, you should therefore restrict the number of devices to the absolute minimum necessary.

Components that can be used

The 125 µs send cycle is only supported by selected PROFINET nodes (e.g. by the ET 200SP I/O system). No SCALANCE switches are currently available for the 125 µs send cycle.

- SIMOTION D455-2 DP/PN as of V4.5

Note

For the PROFINET nodes that support a send cycle of 125 µs and the general conditions that have to be met, see the relevant product documentation.

Dynamic Frame Packing - DFP

Description

For PROFINET V2.2, a separate Ethernet frame is used for each IO device for the cyclic transfer of the IO data. This leads to a large overhead due to the Ethernet header, especially in the case of IO devices with a small amount of IO data.

With the PN V2.3 Performance Upgrade, SIMOTION D455-2 DP/PN supports Dynamic Frame Packing (DFP) via the onboard PROFINET interface X150. With Dynamic Frame Packing, the cyclic

data of multiple IO devices is summarized in one Ethernet frame. DFP significantly reduces the bandwidth required to exchange the cyclic data, especially in the case of IO devices with a small amount of IO data. This makes it possible to either operate more IO devices with the same cycle time or to reduce the cycle time in order to achieve a shorter response time.

In the outbound direction (from the IO controller to the last IO device), the Ethernet frame is automatically reduced. This is achieved by having each device remove its own data from the telegram when forwarding. In the inbound direction (from the last IO device to the IO controller), each IO device adds its own data to the Ethernet frame.

Dynamic Frame Packing is used automatically when send cycles < 250 µs are set and the requirements (see "Requirements" section) are met.

The figure below shows examples of how package groups can be formed. The IO devices (IO-D) with red backgrounds are automatically grouped into package groups.

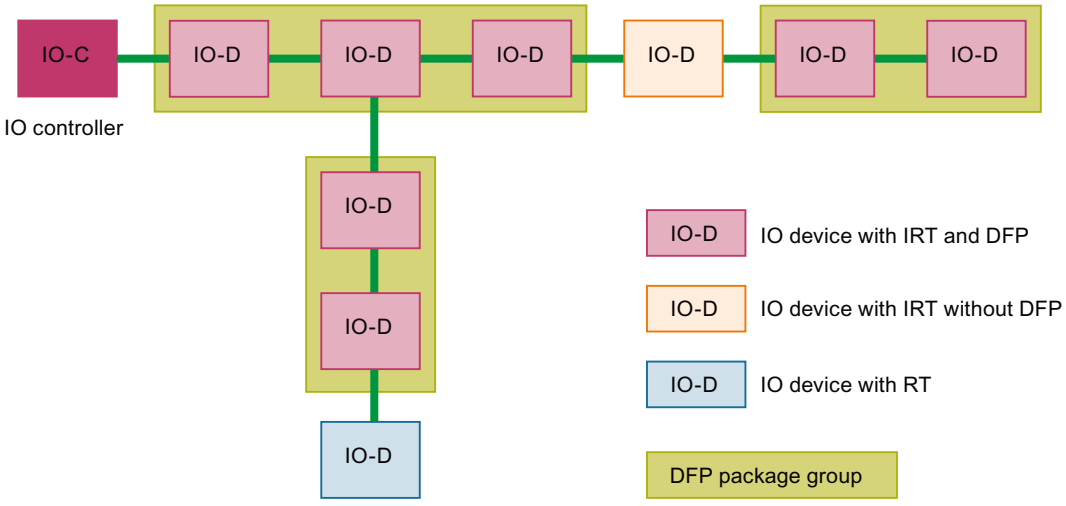


Figure 4-369 Example of DFP package group formation

Requirements

DFP can only be used with the following restrictions:

- DFP can be used only on the onboard PROFINET interface X150 of the SIMOTION D455-2 DP/PN and on the ET200SP High Speed IO-Devices.
- DFP is supported by SIMOTION only as an IO controller. The SIMOTION I-Device does not support DFP.
- A DFP device can be used as a shared device by a second IO controller. Whereby the shared device restrictions apply. One IO controller communicates with DFP via IRT; the other via RT.
- In the case of MRPD, the DFP devices must be present in a spur line. IO devices in the ring are not supported.
- IO devices that do not support DFP terminate a package group; i.e. the grouped IO devices must form a line with no gaps.
- A maximum of 63 IO devices can be packed in one group.
- The size of a package frame is limited to 128 bytes in the outbound direction and 256 bytes in the inbound direction.

See also

Optimized data transfer / 125 µs PROFINET send clock (Page 2175)

4.5.5.3 Configuring PROFINET IO with SIMOTION

New features as of SIMOTION SCOUT V5.3

SINAMICS S210 drive with motion control in SIMOTION SCOUT TIA

SIMOTION SCOUT TIA supports the SINAMICS S210 drive system in conjunction with technology objects. The drive is commissioned in Startdrive or via the Web server and integrated in the project as GSD file. The data exchange is performed via telegrams.

See also

Technology objects with SIMOTION SCOUT TIA and SINAMICS S210 (https://support.industry.siemens.com/cs/ww/en/view/109759892)

Last versions

New as of SIMOTION SCOUT V5.2

Assign device names based on MAC addresses via system function (SIMOTION SCOUT / SIMOTION SCOUT TIA)

With the system function _assignNameOfStationToDevice(), you assign a device name (NameOfStation) to a PN device with a specific MAC address. You can implement modular machine concepts with this function.

System function _assignNameOfStationToDevice(). (Page 2282)

Automatic transfer of the F destination addresses (SIMOTION SCOUT / SIMOTION SCOUT TIA)

With function "Automatic transfer of the F destination addresses", you accept the F destination address configured in the hardware configuration for the Safety commissioning in the drive. The function is also available via scripting.

Transfer the F destination address (F_Dest_Add) (Page 2366)

Change of CRC-secured data of a PROFIsafe telegram is acquired (SIMOTION SCOUT TIA)

For plants and machines with comprehensive safety technology, it must be possible to detect changes to safety-related data quickly and precisely. As of V5.2, the F-Proxy Interconnection table is extended with the **CRC change** column. For each conversion action, changes are detected at the safety axes and displayed in the table.

Additional references

Detailed information and configuring examples specific to SIMOTION SCOUT TIA can be found in the *SIMOTION SCOUT TIA* Configuration Manual.

New in SIMOTION SCOUT as of V5.1

Configuration control for IO systems - machine tailoring (SIMOTION SCOUT TIA)

The configuration control for IO systems (machine tailoring) allows several specific versions or stages of a series machine to be generated from a single series machine project.

Configuration control for IO systems (Page 2287)

Additional references

Detailed information and configuring examples specific to SIMOTION SCOUT TIA can be found in the *SIMOTION SCOUT TIA* Configuration Manual.

New in SIMOTION SCOUT as of V4.5

VentorID, DeviceID for detecting wiring errors

Previously, if the wiring was incorrect, during the topology-based initialization of IO devices by the controller it was possible, for example, that the name of a SINAMICS drive was assigned to a SCALANCE switch. Before it initializes a device, the PN-IO controller now checks whether the device is of the expected type (VentorID/DeviceID).

Time-of-day synchronization

In order to synchronize the time of day of all the components in the system, the time of day can be synchronized via an NTP server.

NTP time-of-day synchronization (Page 2095)

SINAMCIS_Integrated cycle clock < 500µs

As of SIMOTION SCOUT / SIMOTION SCOUT TIA V4.5, SINAMICS_Integrated for drives on SIMOTION D4x5-2 can use bus cycles of less than 500 µs. It supports cycles of between 250 µs and 375 µs. SINAMICS_Integrated remains assigned to the servo cycle. Servo_fast is not supported.

SINAMICS_Integrated cycle clock (Page 2139)

Increasing the number of F-Proxy submodules to 128 safety axes

As of firmware version 4.5, a maximum of 128 F-Proxy submodules, and thus a maximum of 128 safety axes, can be configured for the SIMOTION D455-2. The maximum number of F-Proxy submodules **depends** on the configured PROFIsafe telegram.

Quantity structures I-Device-F-Proxy (Page 2342)

Increase in PN-IO address space

The IO address data (address space assigned to data) has been raised from 4K (in current version) to 6K for SIMOTION.

Quantity structures, address space (Page 2162)

Broadcast _udpReceive() and _udpSend()

A new checkbox "Permit broadcast _udpReceive/_udpSend" is provided, which allows you to configure whether the EDD network load filter should be deactivated if you want to perform Open User Communication with UPD broadcast (system functions _udpReceive() and _udpSend()).

Using broadcast communication (Page 2302)

Changes to PROFIenergy communication

PROFIenergy communication needs to be explicitly activated in SIMOTION SCOUT TIA for the SIMOTION I-Device in order to enable the PROFIenergy data sets. PROFIenergy communication is automatically activated in SIMOTION SCOUT and need not be set explicitly.

PROFIenergy communication (Page 2273)

Improvements in usability of isochronous tasks (SIMOTION SCOUT)

The Servo_fast cycle can be used as a reference cycle only if the Servo_fast box is checked. This selection is not available on SIMOTION controllers that do not support Servo_fast.

Overwriting of the existing PROFINET device name (SIMOTION SCOUT TIA)

Existing PROFINET device names are overwritten by the IO controller during the topology-based initialization routine. When IO devices were incorrectly initialized (due to a wiring error, for example) in previous software versions, it was necessary to reset the device names so that the devices could be reinitialized.

Address tailoring for series machines (SIMOTION SCOUT TIA)

The address tailoring function can be used to assign an IP address and a NameOfStation to IO controllers and IO devices by means of system functions at the commissioning stage. This feature allows a machine consisting of an IO controller and IO devices to be supplied with IP addresses and NameOfStation at the commissioning stage, and permits multiple instances of the machine to be commissioned on a single physical Ethernet without an engineering system.

Series machine projects - address tailoring (Page 2286)

Performance upgrade PN V2.3 (SIMOTION SCOUT TIA)

SIMOTION SCOUT TIA supports the performance upgrade PN V2.3. You can thus now configure send cycles of < 250 µs.

Optimized data transfer - performance upgrade (Page 2175)

Media redundancy MRPD (SIMOTION SCOUT TIA)

With this version and later, the MRPD functionality previously available only for SIMOTION SCOUT is now also available for SIMOTION SCOUT TIA.

Media redundancy (Page 2166)

SIMOTION controller as a shared I-Device (SIMOTION SCOUT TIA)

A SIMOTION I-Device can be operated as an IO device on two higher-level IO controllers. This functionality is known as shared I-Device.

Shared device on SIMOTION controller (SIMOTION SCOUT TIA)

It has been possible to configure a shared device for SIMOTION SCOUT as of version V4.2 of SIMOTION SCOUT. A shared device can now also be configured with SIMOTION SCOUT TIA. A typical application is a SIMOTION CPU as an automation CPU and an F-CPU for the safety application that is divided between the submodules of the SIMATIC ET200SP.

SINAMICS G120 as a shared device on the SIMOTION controller (SIMOTION SCOUT TIA)

SINAMICS G120 as a shared device only supports a fixed division of the subslots between the two sharing IO controllers. The shared device functionality is available only for the SINAMICS G120 GSD drives.

Update of GSD version to V2.32 (SIMOTION SCOUT TIA)

SIMOTION SCOUT TIA uses GSD version V2.32. A SIMOTION I-Device is exported as GSD V2.32 in SIMOTION SCOUT TIA. SIMOTION SCOUT utilizes GSD version V2.3.

Additional references

Detailed information and configuring examples specific to SIMOTION SCOUT TIA can be found in the *SIMOTION SCOUT TIA* Configuration Manual.

New in SIMOTION SCOUT as of V4.4

Shared iDevice

As of SIMOTION V4.4, an iDevice of the PN-IO interfaces can be operated with two higher-level IO controllers. This functionality is referred to as "shared iDevice" and is comparable with the shared device.

See also Shared iDevice and PROFIsafe (Page 2374)

PROFINET V2.3 Performance Upgrade

The Control Unit SIMOTION D455-2 DP/PN supports the PROFINET Performance Upgrade in accordance with the PROFINET specification V2.3. The PROFINET send clock can now be set to a minimum of 125 µs.

See also Performance Upgrade PN V2.3 (Page 2175)

Support for PROFIenergy

PROFIenergy is a data interface based on PROFINET. It allows loads to be shut down during idle time in a controlled, centralized manner, and irrespective of the manufacturer and device.

See also PROFIenergy (Page 2271)

Setting IP address and device name via user program

In the case of SIMOTION, the IP configuration can be set via the user program for the PN-IE and PN-IO interfaces.

See also Setting IP address and device name via UP (Page 2277)

Identification and maintenance (I&M) data

As of V4.4, SIMOTION supports the data sets 1-4 in addition to the I&M data set 0 (electronic rating plate) as a PROFINET iDevice.

See also I&M data (Page 2231)

PROFINET IO with IRT

As of SIMOTION V4.4, an additional configuration step has been added for the configuration of PROFINET IO with IRT. In the HW Config of the controller CPU, under **Object properties** in the **Isochronous tasks** tab, the setting is made as to whether the servo task is to be assigned to the PROFINET interface as the application level for the processing of isochronous data:

- If PROFINET IO devices with isochronous data are to be configured on PROFINET, the PROFINET interface must first be assigned to the servo under **Object properties** of the SIMOTION CPU in the **Isochronous tasks** tab (Use "..**I/O data isochronously to the servo**" checkbox). Only then can servo be assigned for the device in the **Object properties** of the PROFINET interface in the **I/O cycle** tab (**Assign I/O device isochronous** list box).
- If no devices are configured with isochronous data on the PROFINET interface, the PROFINET interface must also not be assigned to the SERVO.

As of SIMOTION V4.4, it is therefore also possible to configure the PROFINET interface of SIMOTION devices with IRT without the data being transferred isochronously. This means that the data is transferred synchronously via IRT on the bus, but the applications in SIMOTION and in the device do not accept the data isochronously. For example, an ET200HF can be configured with IRT, but the I/O data is not transferred isochronously to the physical input or output. Or a

SIMOTION I-device can be connected via IRT to a higher-level controller, but the input and output data of the I-device interface is not processed isochronously by the CPU.

Up to and including V4.3, SIMOTION always performed the communication with PROFINET IO with IRT isochronously to the servo, even if isochronous mode was not configured on any PROFINET IO device.

New in SIMOTION SCOUT as of V4.3

Media redundancy protocols MRP and MRPD

Via the so-called Media Redundancy Protocol (MRP or MRPD), it is possible to establish redundant networks, see Media redundancy for SIMOTION (Page 2166).

Second PROFINET I/O interface

With SIMOTION D4x5-2 DP/PN, is it possible to have two PROFINET I/O interfaces (integrated interface and CBE30-2) on one device, see Two PROFINET IO interfaces (Page 2144). If two interfaces are available, one interface can be operated in the servo cycle clock and the other interface in Servo_fast cycle clock (integrated interface).

Position control cycle clock and Servo_fast cycle clock

Position control cycle clock and Servo_fast both run via PROFINET (slow and fast bus cycle clock), see Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 2207).

Standard Ethernet interfaces

The standard Ethernet interfaces now include basic PROFINET services, Properties of the SIMOTION Ethernet interfaces (Page 2289).

More message frames for PROFIsafe

Additional message frames are provided for PROFIsafe, see also Message frames and signals in drive-based safety (Page 2332).

See also

Media redundancy for SIMOTION (Page 2166)

Two PROFINET IO interfaces (Page 2144)

Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface (Page 2207)

Properties of the SIMOTION Ethernet interfaces (Page 2289)

Message frames and signals in drive-based safety (Page 2332)

New with SIMOTION SCOUT V4.2 or higher

IRT synchronization process

SIMOTION with IRT High Performance. With PROFINET IRT, this synchronization process is set automatically.

Controllers with integrated PN interface

With SIMOTION D4x5-2 DP/PN, new SIMOTION D controllers with integrated PN interfaces are available.

Isochronous operation for applications has been simplified.

The typical isochronous applications for Motion Control have been made much more straightforward. The Ti/To time constants can be calculated automatically for all IO devices. The configuration in HW Config has been enhanced to allow the object properties of isochronous tasks to be set to automatic on the controller.

iDevice

The I device functionality has been improved with the release of Step 7 V5.5. If SIMOTION SCOUT projects use a lower version than V4.2, certain points must be considered when upgrading. You can find more information in the I device chapter.

iDevice F-Proxy

Using the I device F-Proxy, you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFINET with SIMOTION devices (SIMOTION D, SIMOTION P3xx, SIMOTION C240 PN) for the lower-level drives.

Second servo cycle clock (Servo_fast)

The second servo cycle clock enables you to operate two bus systems in different application cycle clocks in the case of SIMOTION D4x5-2 DP/PN. An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This allows you to divide your application into a slow part (SINAMICS_Integrated with servo and IPO) and a fast part (IO devices on the PROFINET interface with Servo_fast and IPO_fast).

Procedure for configuring PROFINET IO with IRT High Performance

Procedure

The following steps need to be performed when configuring PROFINET IO with IRT High Performance:

1. Insert the SIMOTION module.
   Select the **Device** followed by the **Device version**. A PROFINET interface is already integrated in devices labeled with PN.
   - With SIMOTION C and SIMOTION P, you only select the SIMOTION version, e.g. V4.4.
   - With SIMOTION D, the modules have up to two PROFINET IO interfaces (onboard and CBE30-2) depending on the version. Depending on the version, select SIMOTION Version V4.4, the option module, and the SINAMICS drive and version.
     You can select the CBE30-2 option module for PROFINET here or enter it later in HW Config from the hardware catalog under **SIMOTION Drive-based > SIMOTION D4x5 xxx > 6AUxxxxxx > Vx.x SINAMICS > CBE30-x-PN-IO**.
2. Insert IO devices:
   Insert IO devices from the hardware catalog in HW Config into the IO system. The IO devices can be found in the hardware catalog under **PROFINET IO**.
3. Configure sync domain and specify send clock:
   Configure the PROFINET IO node as sync master (clock generator) and define the associated sync slaves. Specify the send clock.
4. Generate the topology:
   Define the topology, i.e. specify how the individual ports of the PROFINET devices (IO controller and IO device) are interconnected with one another. The topology only needs to be configured for PROFINET IO with IRT High Performance. If topology-based initialization is to be used, the topology will need to be configured for all PROFINET devices.
5. As an option, you can still configure the controller-controller data exchange broadcast:
   Specify which address areas are to be used for sending and receiving.

Inserting and configuring SIMOTION D

General information on inserting and configuring SIMOTION D

General information

You have created a project and want to configure a SIMOTION D using a PROFINET interface.

With SIMOTION D, the control units feature an onboard PROFINET interface (D410 PN, D410-2 DP/PN, D4x5-2 DP/PN) or an option module CBE30 for PROFINET (D4x5), depending on the version concerned.

With V4.3 or higher, you can use a CBE30-2 as a second PROFINET interface with SIMOTION D4x5-2 DP/PN.

You can select the CBE30-2 when inserting the SIMOTION D controller in the SIMOTION SCOUT project or, alternatively, you can insert the relevant version for the SIMOTION D controller later

in HW Config from the HW catalog. You will find the CBE30-2, for example, under **SIMOTION Drive-based > SIMOTION D4x5 xxx > 6AUxxxxxx > Vx.x SINAMICS > CBE30-x-PN-IO**

Implementing the PROFINET interface:

- Already integrated in the case of D410 PN, D410-2 DP/PN, and D4x5-2 DP/PN
- With D4x5-2 DP/PN, a second PROFINET interface can be implemented with a CBE30-2.
- Available as an option in the case of D4x5 (option module CBE30)

The procedure is explained in the following sections.

Inserting and configuring SIMOTION D4x5-2 DP/PN/D410 PN/D410-2 DP/PN

General information

You have created a project and want to configure a SIMOTION D4x5-2 DP/PN, SIMOTION D410 PN, or SIMOTION D410(-2) DP/PN using the internal PROFINET interface. The procedure for this will be explained using the example of a SIMOTION D4x5-2 DP/PN.

Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Select SIMOTION D under **Device** and click on a **Device version** e.g. D455-2 DP/PN.

3. Select the **SIMOTION Version**, e.g. **V4.4** and **SINAMICS S120 Integrated.**

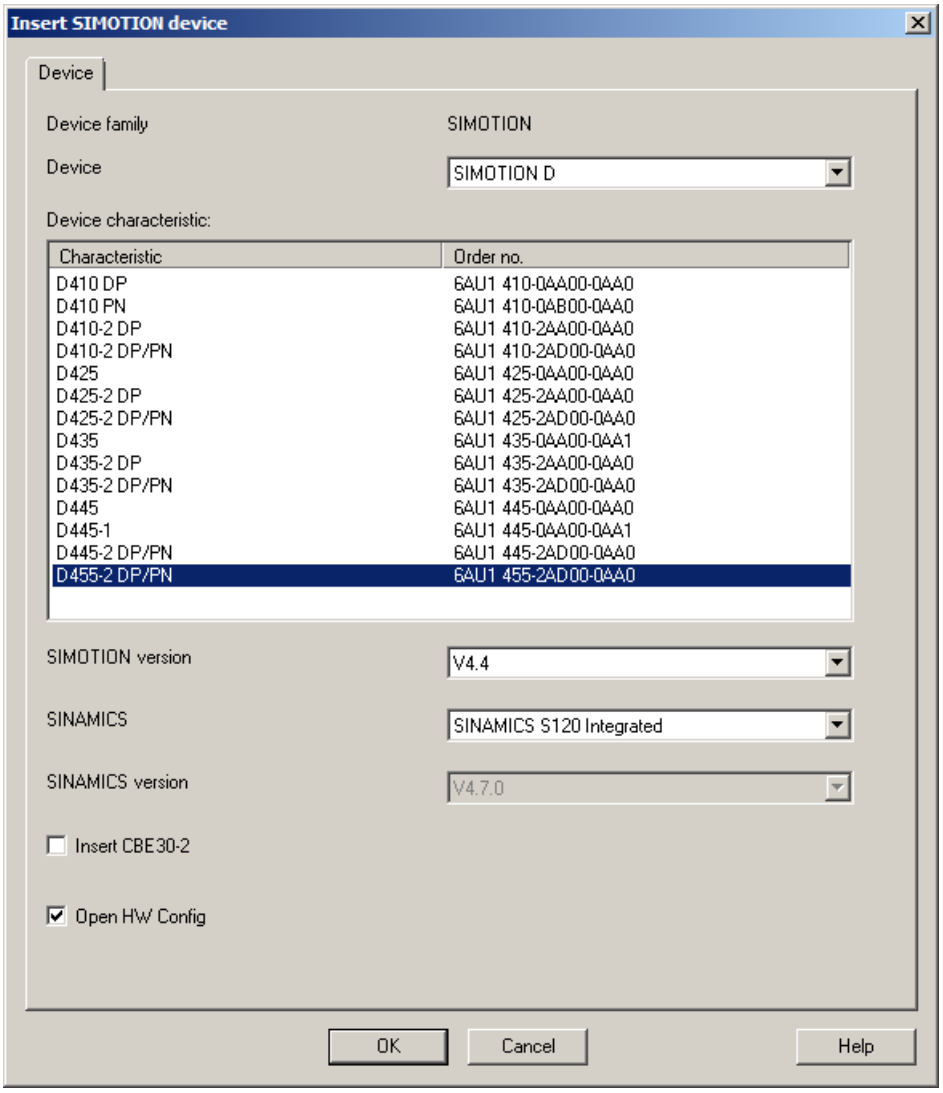


Figure 4-370 Inserting a SIMOTION device

4. Activate the **Open HW Config** check box to add e.g. an option module or an IO system.
5. Click **OK** to confirm.

6. The dialog box for creating a PROFINET subnet will be displayed. Create a new subnet using **New...** and enter the required **IP address** and **subnet mask**. Click **OK** to confirm.

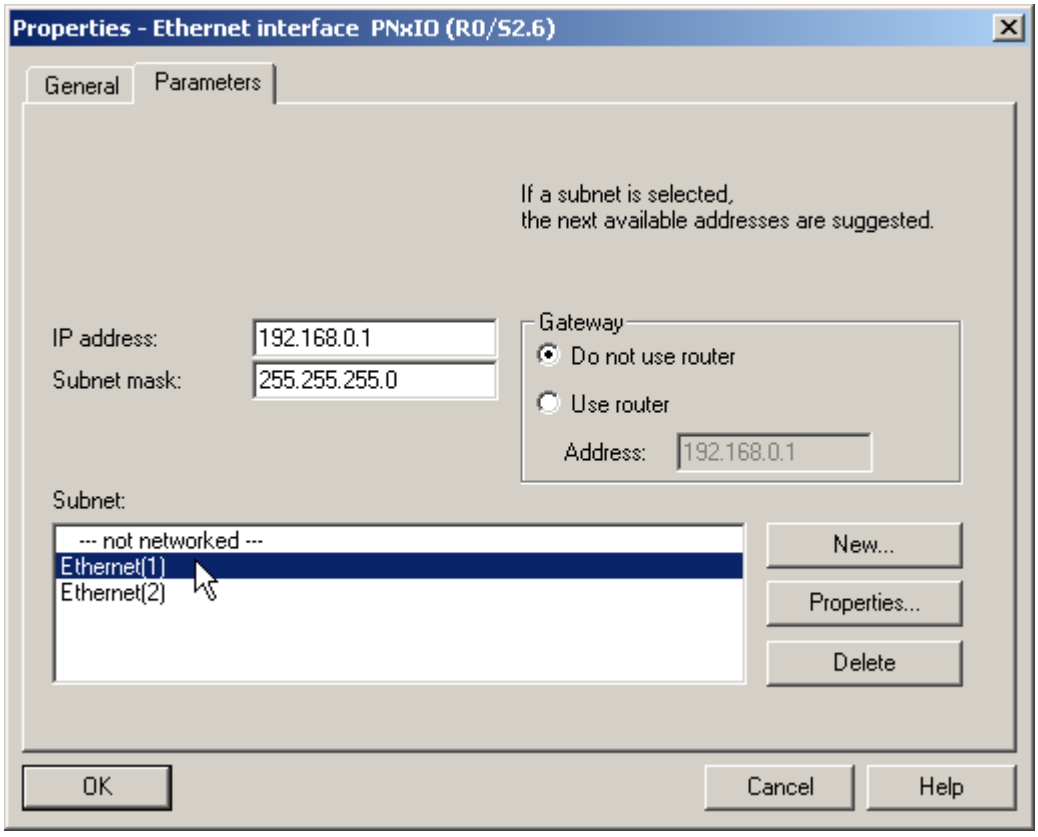


Figure 4-371 Creating a new Ethernet subnet

7. Select the PG/PC interface from the next dialog box and click OK to confirm. The device is inserted, HW Config opens, and the module with the configured PROFINET subnet is displayed.

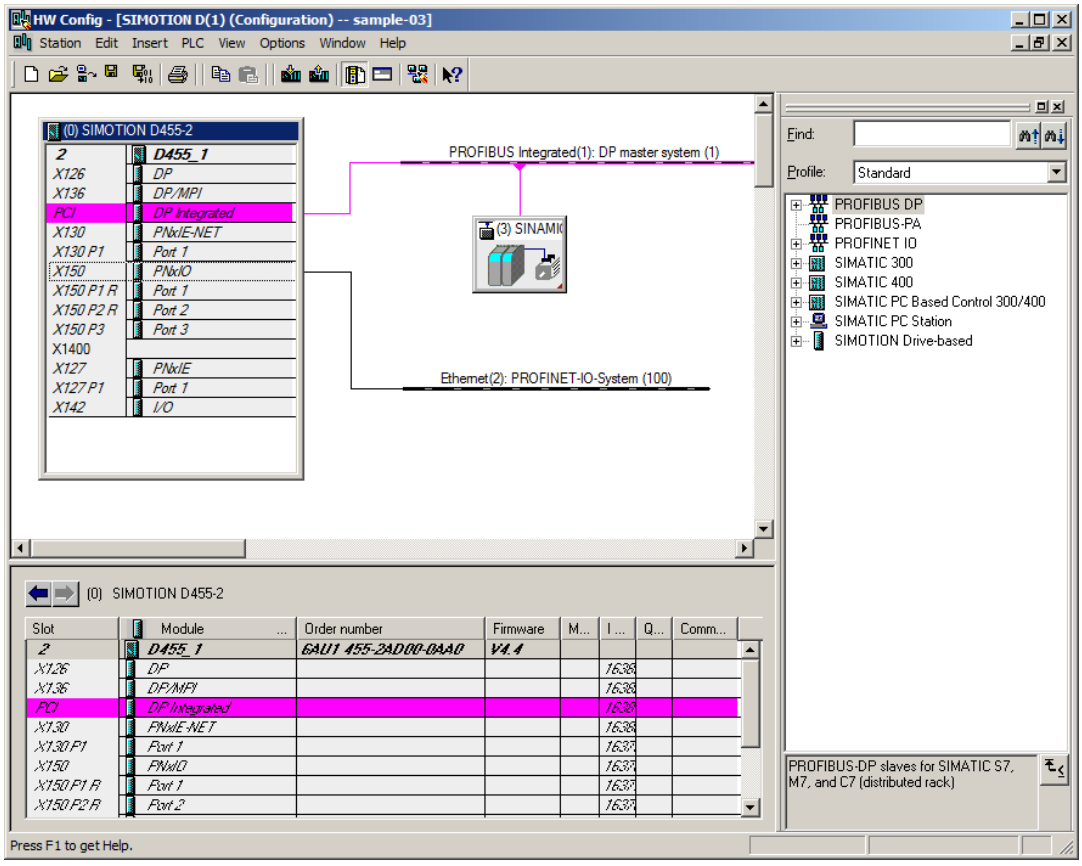


Figure 4-372 SIMOTION D455-2 DP/PN in HW Config

See also

Add and configure PROFINET interface CBE30-2 (Page 2191)

Insert and configure a SIMOTION D4x5-2 incl. CBE30-2

Requirement

You have already created a project and now want to insert a SIMOTION D4x5-2 with option module CBE30-2 for PROFINET.

Procedure:

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Select SIMOTION D under **Device** and click on the **Device version** e.g. D455-2 DP/PN.

3. Select the **SIMOTION Version**, e.g. **V4.4** and activate the option module **CBE30-2**.
   You can also insert the option module CBE30-2 in HW Config at a later point (see Add and configure PROFINET interface CBE30-2 (Page 2191)).

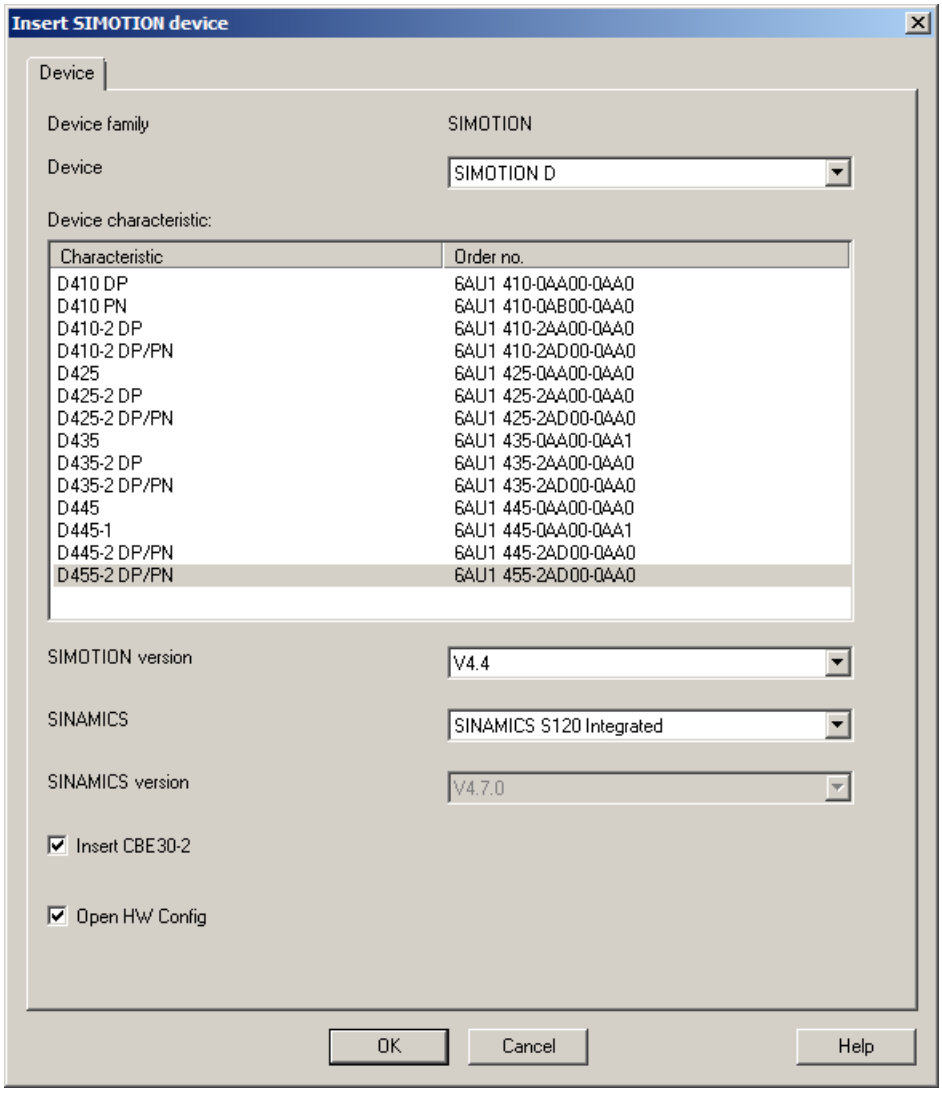


Figure 4-373 Creating a SIMOTION D4x5-2 with CBE30-2 for PROFINET

4. Activate the **Open HW Config** checkbox to insert an IO system, for example.

5. Click **OK** to confirm. The dialog box for creating a PROFINET subnet will be displayed. Create a new subnet using **New...** and enter the required **IP address** and **subnet mask**. Press **OK** to confirm.
6. Select the PG/PC interface from the next dialog box and click **OK** to confirm. The device is inserted, HW Config opens, and the module with the configured PROFINET subnet is displayed.

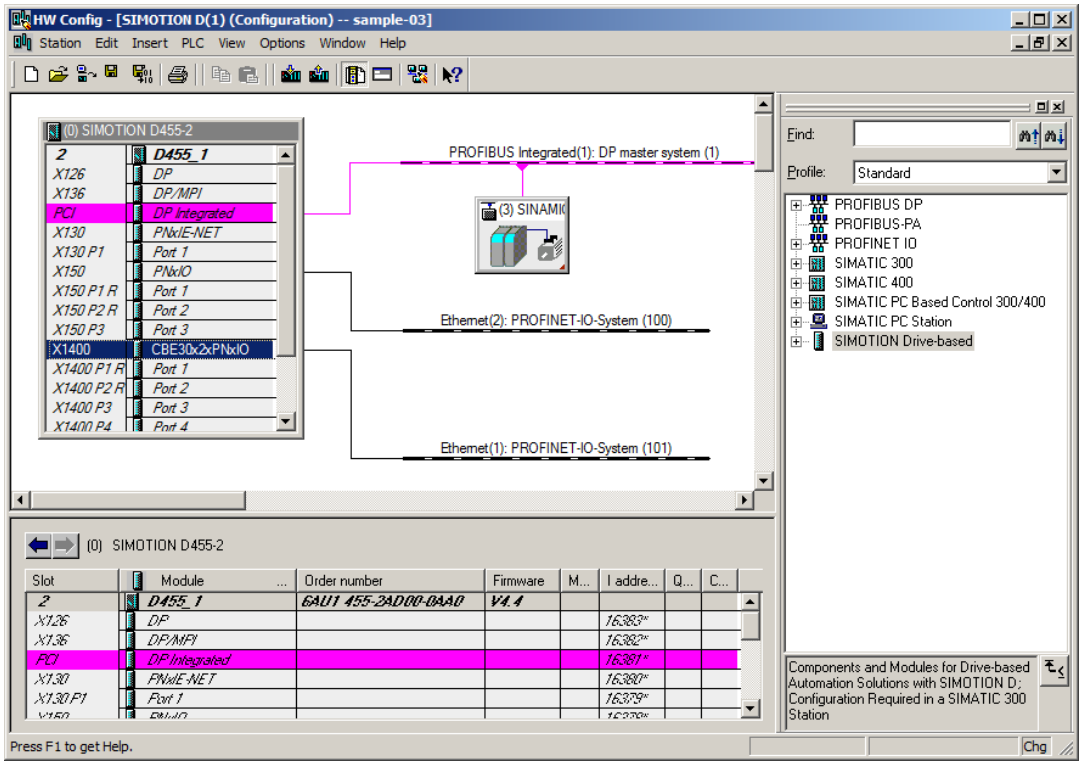


Figure 4-374 SIMOTION D4x5-2 in HW Config

Add and configure PROFINET interface CBE30-2

Requirement

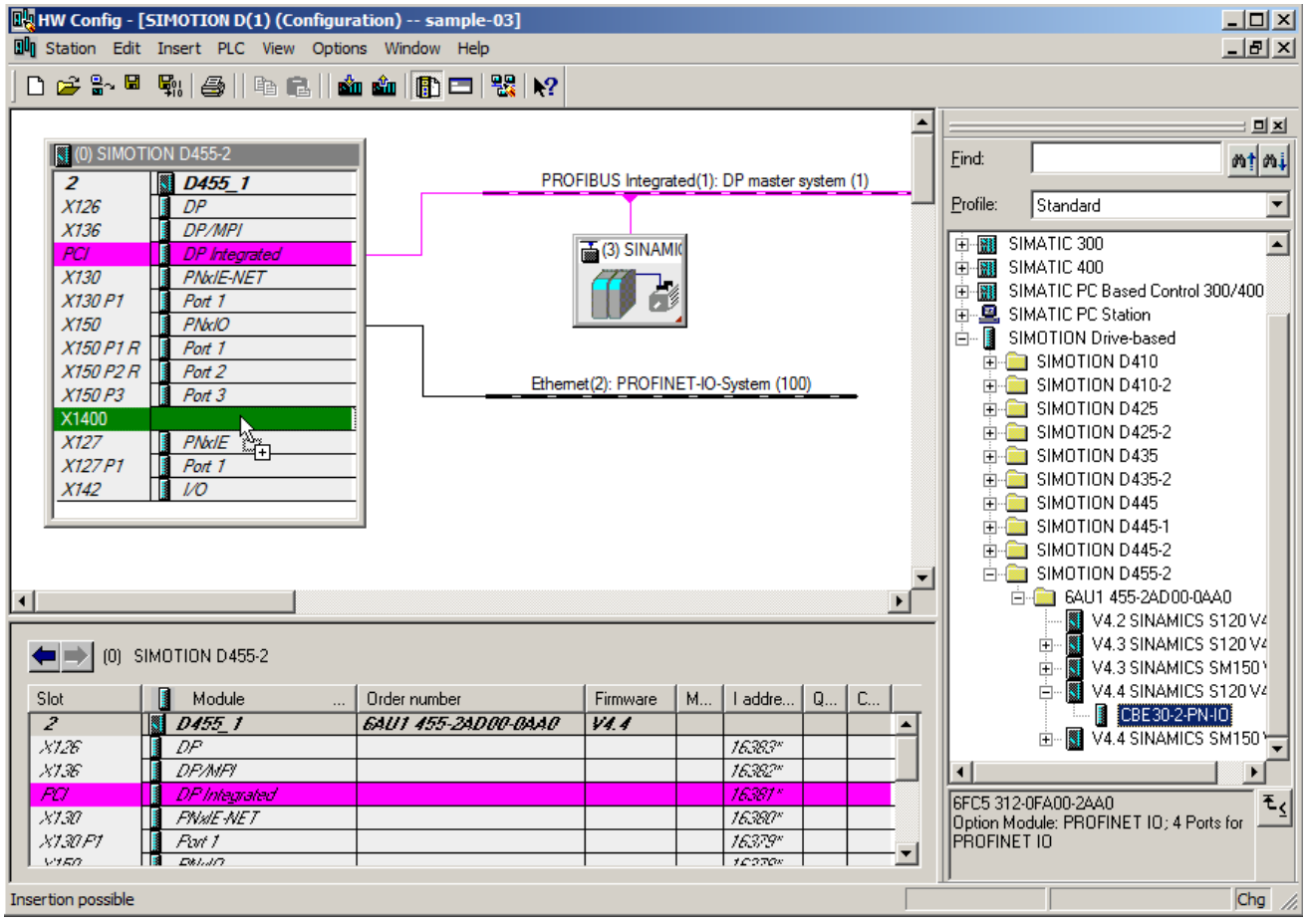
You have already created a project and inserted a SIMOTION D4x5-2 device.

When adding a device in SIMOTION SCOUT, you can add a CBE30-2 as standard. You can also insert the option module in HW Config at a later point.

Procedure

1. In the project navigator, double-click the module (in this case D455-2 DP/PN). HW Config is displayed with the corresponding module.
2. In the hardware catalog, select the appropriate option module under **SIMOTION Drive-based > SIMOTION D4x5-2 > 6AUxx > Vx.x SINAMICSxx > CBE30-x-PN-IO**. Note the CPU type and version.

3. Click PROFINET module CBE30-2-PN. As soon as the appropriate CBE30-PN is selected, the interface **X1400** in the rack turns green.

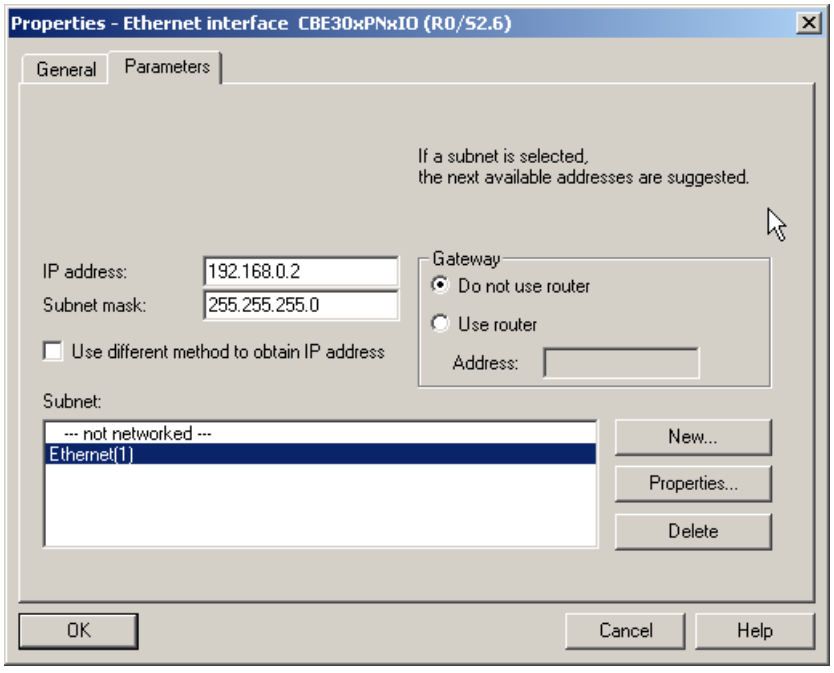


4. Drag the CBE30-2-PN to the corresponding interface of the SIMOTION module (X1400). The **Properties - Ethernet Interface CBE30-2-PN (R0/S2.6)** window opens.
5. Click **New** to create a new subnet. The **Properties – New subnet Industrial Ethernet** dialog box is displayed.

6. Click **OK** to confirm these entries. A new Ethernet subnet will be created, e.g. Ethernet(4).

Note

The IP addresses of the SIMOTION interfaces must be in different address areas.



7. Select the **subnet**.
8. Assign the required **IP address**.
9. Accept the settings by clicking **OK**.

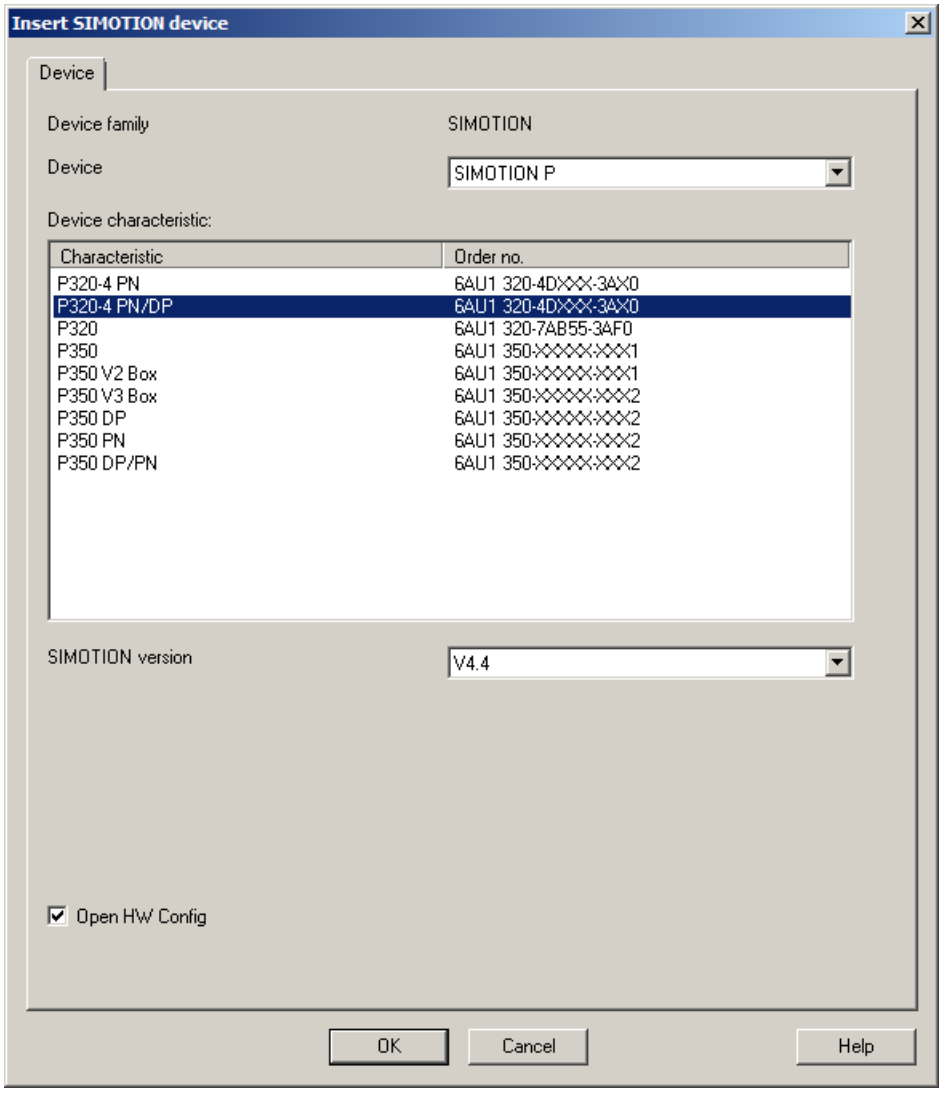
Adding and configuring SIMOTION P

Requirement

You have already created a project and want to insert a SIMOTION P with PROFINET. The procedure for this will be explained using the example of a SIMOTION P320.

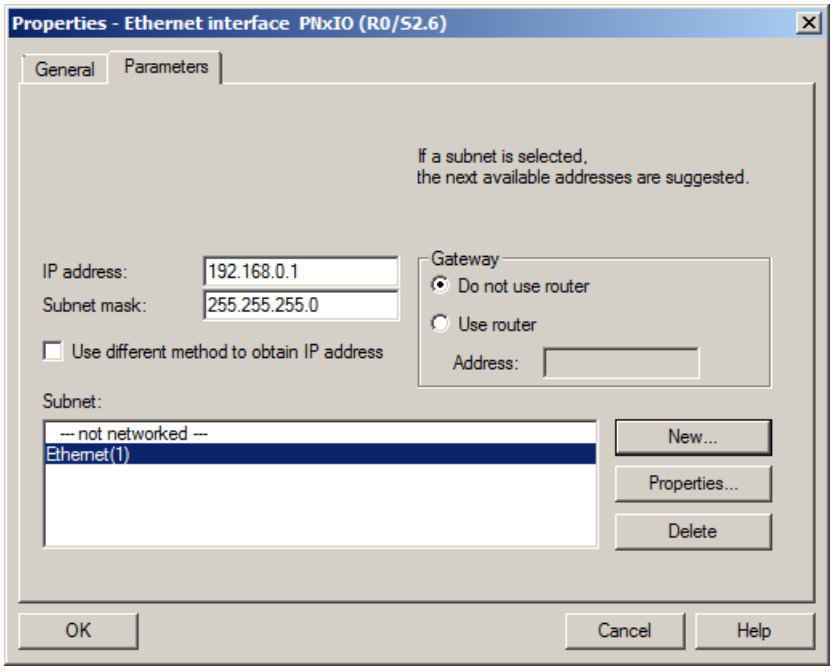
Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Under **Device**, select SIMOTION P, then click on a **Device version** (e.g. P320-4), and select the **SIMOTION Version**, e.g. V4.4.

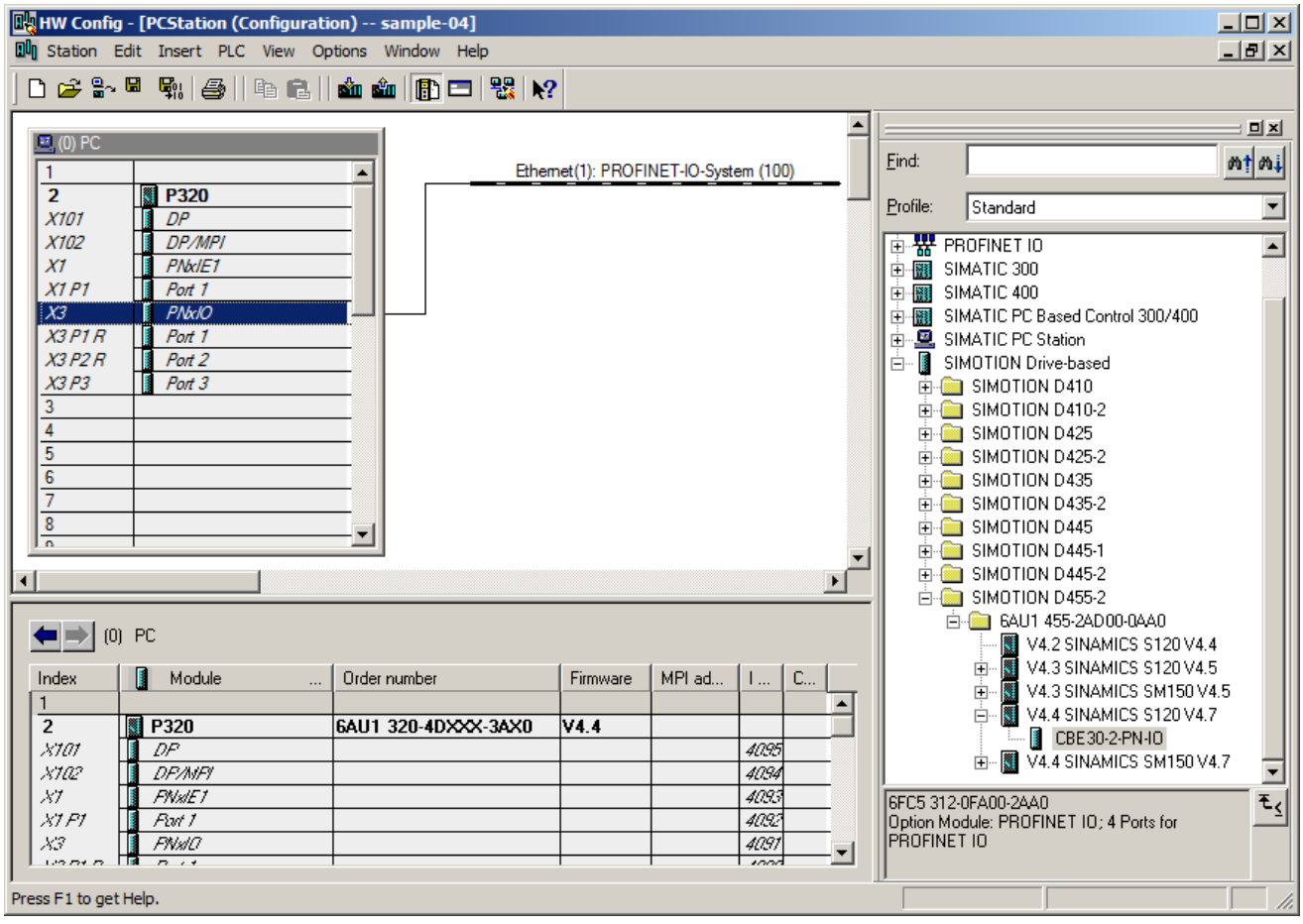


3. Activate the **Open HW Config** checkbox to insert an IO system, for example.
4. Click **OK** to confirm. The dialog box for creating a PROFINET subnet will be displayed.

5. Enter the required **IP address** and the **subnet mask** here. Click **OK** to confirm.



6. Select the PG/PC interface from the next dialog box and click **OK** to confirm.
   The HW Config opens and displays the module with the configured PROFINET subnet.



Adding and configuring SIMOTION C

Requirement

You have already created a project and now want to insert a SIMOTION C with PROFINET. The procedure for this will be explained using the example of a SIMOTION C240 PN.

Procedure

1. Click on **Insert SIMOTION device** in the project navigator to open the device selection dialog box.
2. Under Device, select **SIMOTION C**, then click on a **Device version** (e.g. **C240 PN**), and select the **SIMOTION version**, e.g. V4.4.

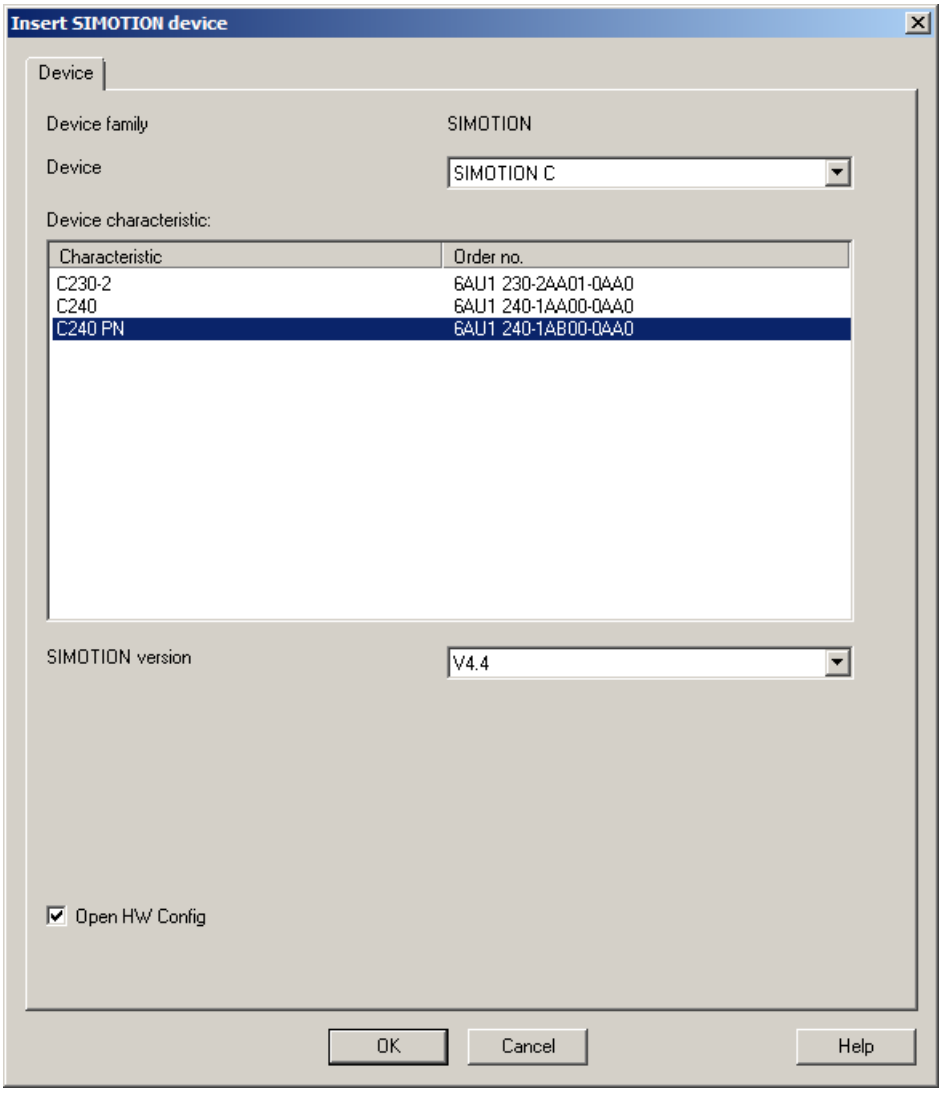


Figure 4-375 Creating a new C240 PN device

3. Activate the **Open HW Config** checkbox to insert an IO system, for example.

4. The dialog box for creating a PROFINET subnet will be displayed. Enter the required **IP address** and the **subnet mask** here. Click **OK** to confirm.

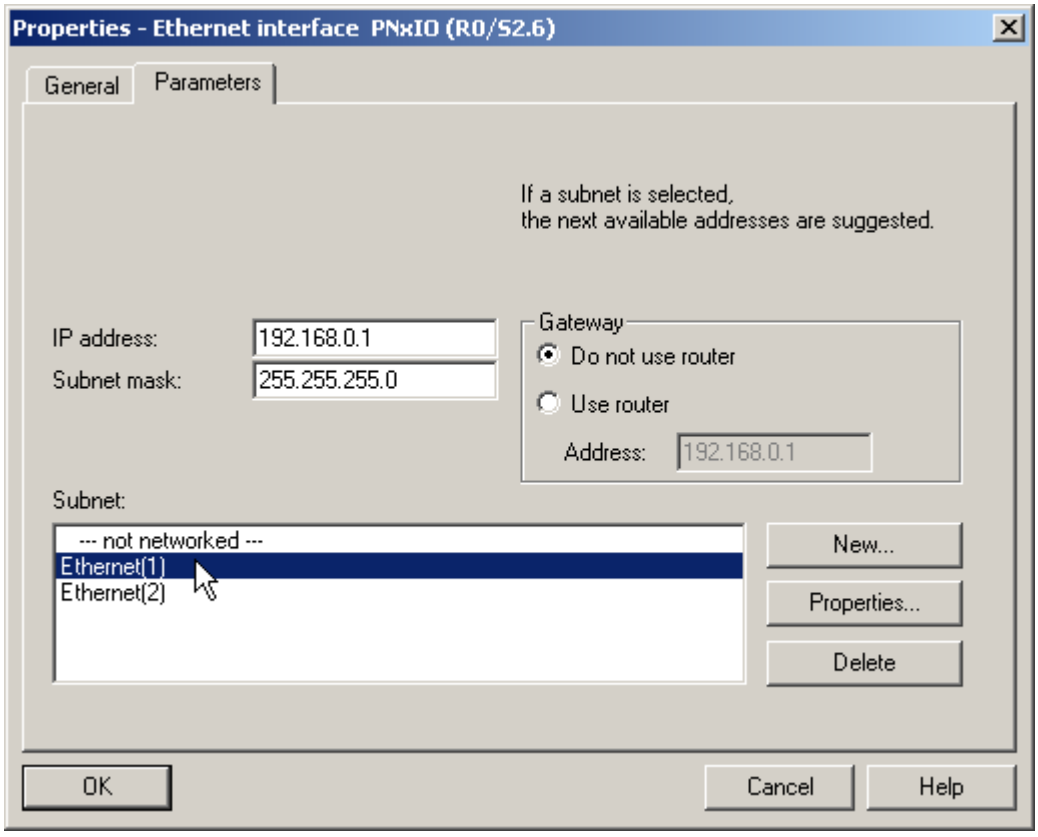


Figure 4-376 Creating a new Ethernet for C240 PN

5. Select the PG/PC interface from the next dialog box and click **OK** to confirm.
The HW Config opens and displays the module with the configured PROFINET subnet.

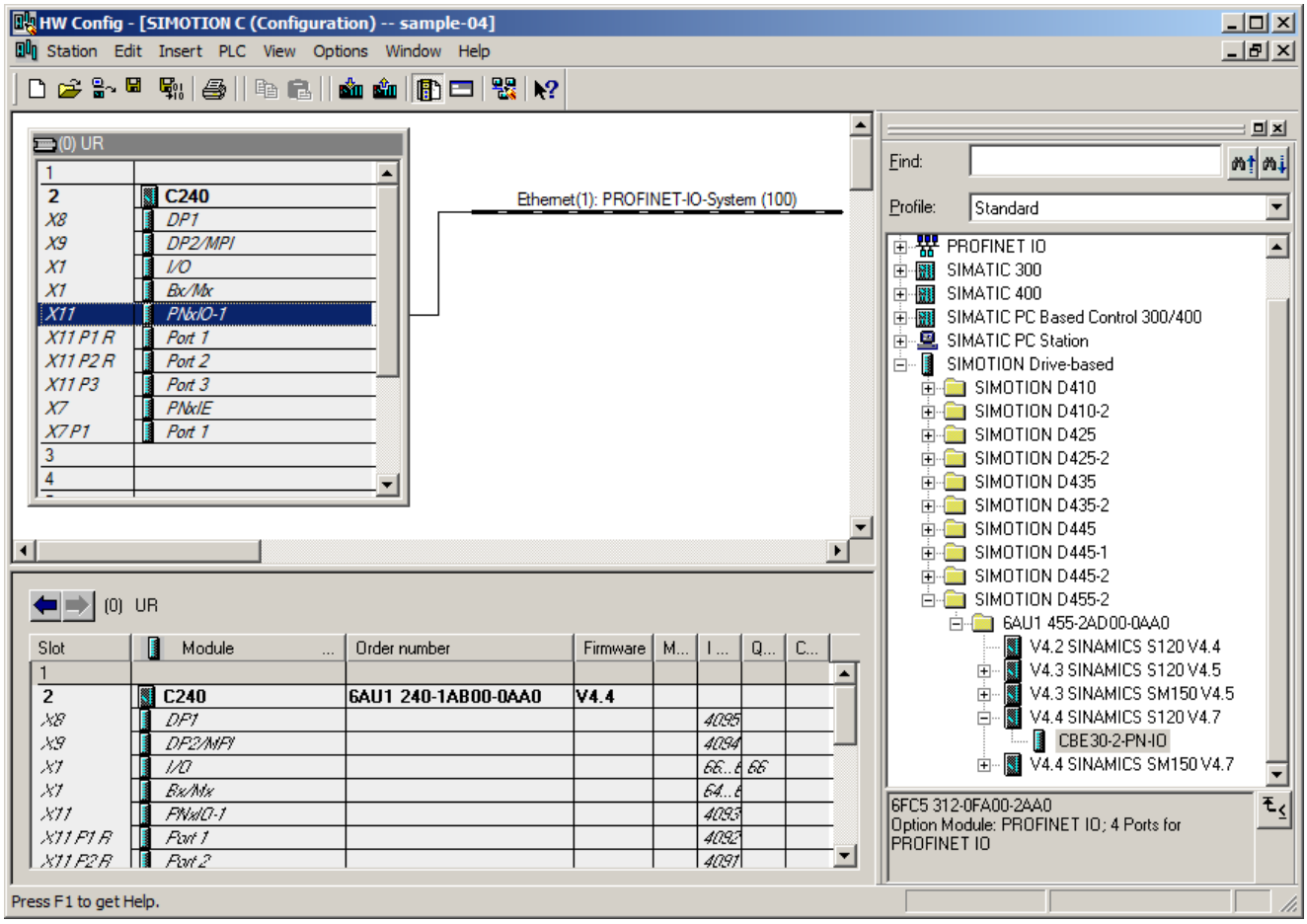


Figure 4-377 HW Config with PROFINET for C240 PN

Creating a sync domain

A sync domain is a group of PROFINET devices synchronized to a common cycle clock. One device has the role of the sync master (clock generator), all other devices have the role of sync slave.

Note

All devices that exchange data via PROFINET IRT must belong to a single sync domain and be directly connected to one another. The connection may not be interrupted by devices that do not support PROFINET IRT, because otherwise the synchronization information cannot be transferred.

Procedure

1. In HW Config, open the station with the PROFINET devices to be involved in IRT communication and select, for example, the PROFINET interface **PNxIO** in the case of a SIMOTION D455-2 DP/PN.
2. Select the **Edit > PROFINET IO > Domain Management** menu command. A dialog tab with the list of all the devices opens. A default sync domain is created and the devices are already assigned.
3. Select the station in the upper field and in the lower field double-click the device that is to be configured as the Sync-Master, e.g. SIMOTION D. This opens the Properties dialog box for the device.

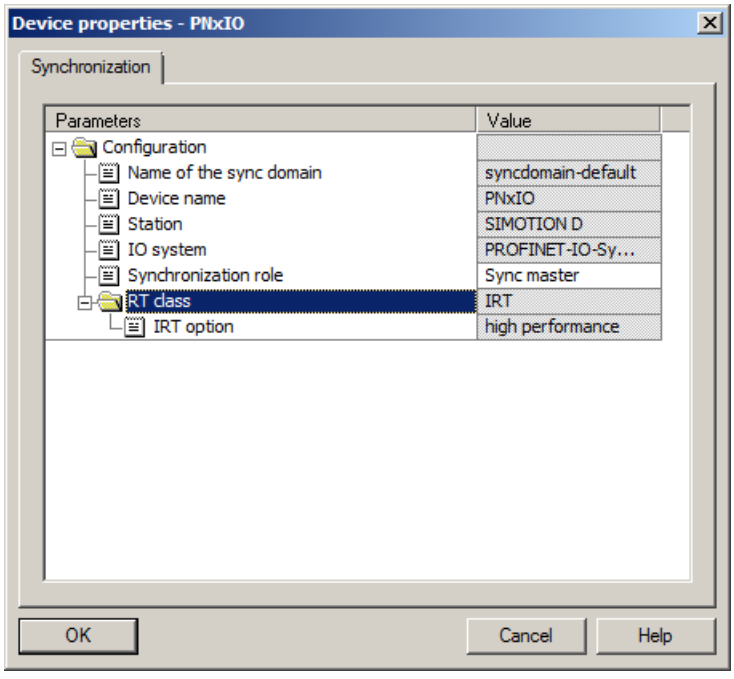


Figure 4-378 Selecting synchronization

4. Set the synchronization type to **Sync-Master**. IRT High Performance is the standard setting on SIMOTION controllers.
5. Confirm the settings by clicking **OK**.
6. Then, select all devices which are to be configured as sync slaves (keep the Ctrl key depressed and select the devices one after the other).
7. Then, click on the **Properties device** button.
8. Set the synchronization type to **Sync-Slave** in the dialog box.
9. Confirm the settings by clicking **OK**.

Note

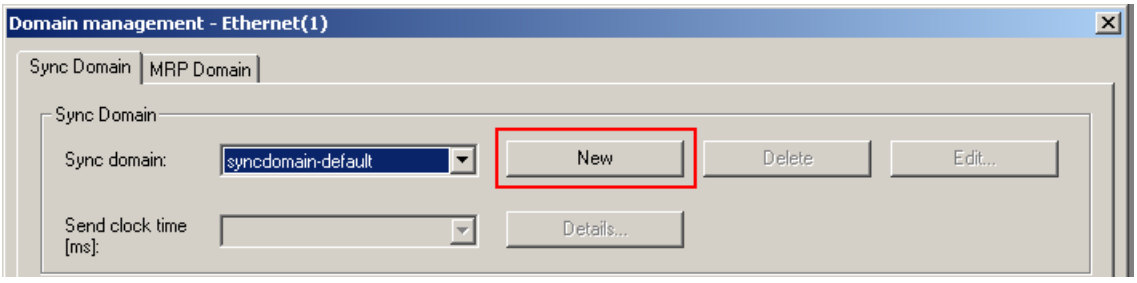
Any devices for which **not synchronized** is selected will not be involved in IRT communication, but will automatically take part in RT communication.

Creating a sync domain and changing the name of a sync domain

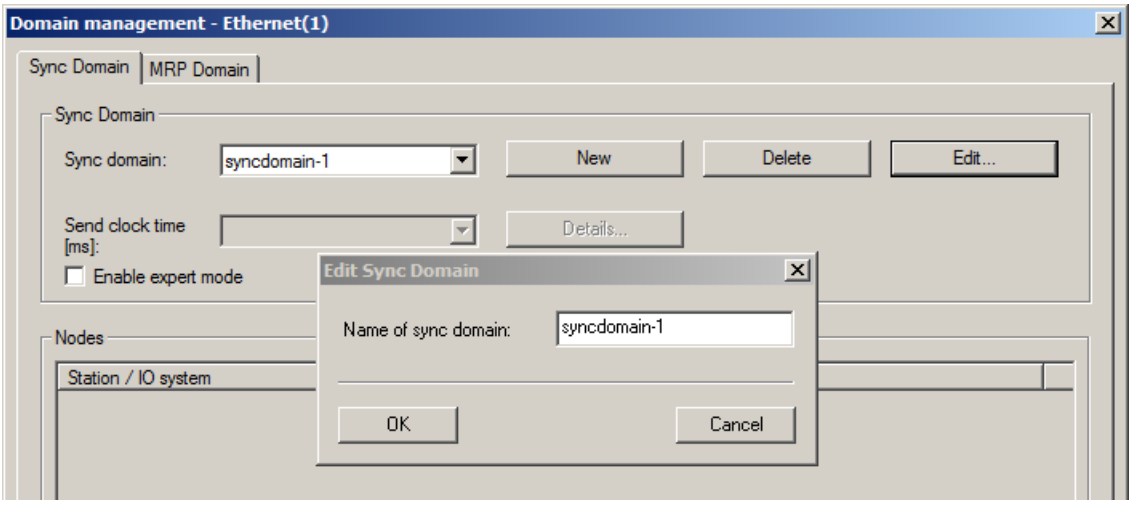
If you work with two PROFINET IO interfaces, they must be in different sync domains. The two sync domains must not have the same name.

To create a new sync domain and change the name of a sync domain, proceed as follows:

1. Open HW Config and right-click on a device of the sync domain.
2. In the shortcut menu, select **PROFINET IO Domain Management**.
   The dialog **Domain Management - PN-xxxx** opens.

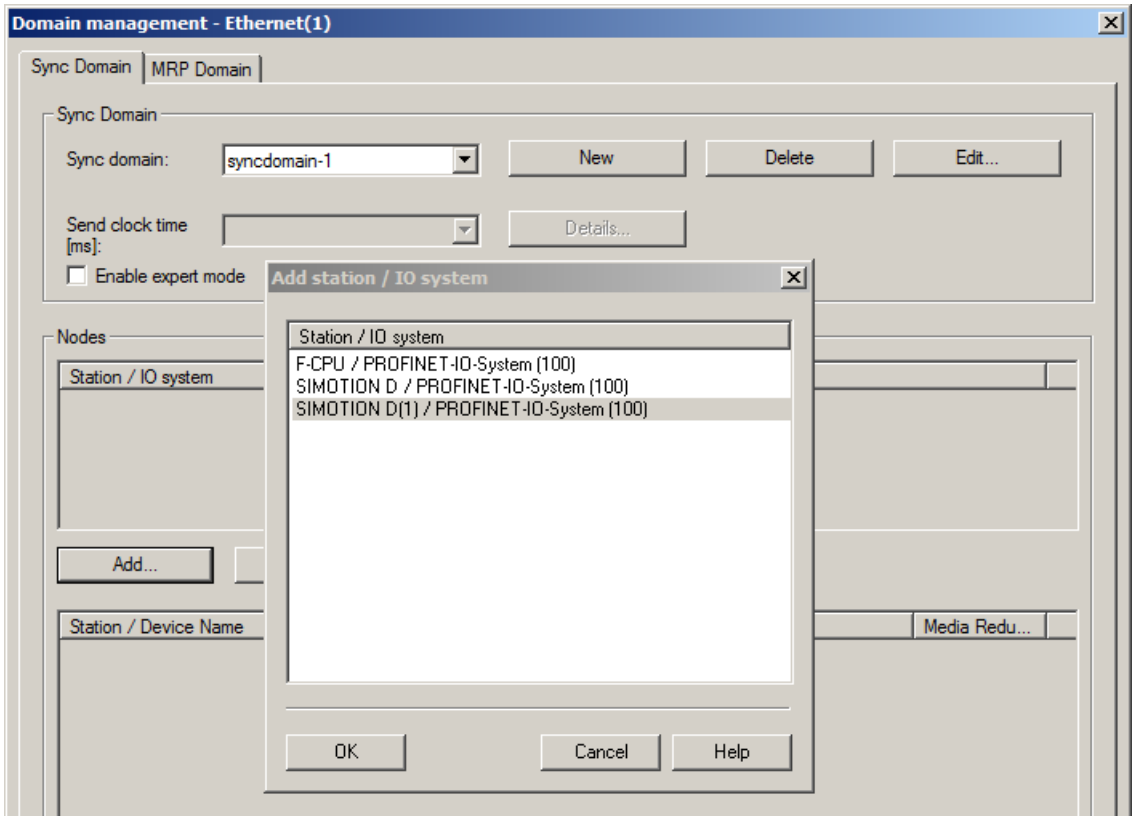


3. Click **New** to create a new sync domain.
4. Select the newly created sync domain in the **Sync Domain** drop-down list box.
5. Click the **Edit** button to open the **Edit Sync Domain** dialog box.



6. Enter a new name for the sync domain and click on **OK**.
   The name of the sync domain is changed.

7. Click on **Add** to add a station or an I/O system to the sync domain.



8. Click **OK** to add the station / IO system.

Defining send clock and refresh times

PROFINET RT and IRT are forms of cyclic communication; the basic clock is the send clock. The update time is a multiple (2^n) of the send clock and the devices are supplied with data in this cycle clock. Individual update times can be set for each device.

Note

With IRT High Performance, the devices are supplied with data during each sending cycle (sending cycle = update time). However, the data is only evaluated every nth cycle in the SIMOTION Servo cycle clock (Servo_fast runs scaled-down to the servo cycle clock) or SINAMICS_Integrated.

As far as SINAMICS_Integrated is concerned, the controller can be used to define an application cycle in such a way that it is only supplied with new data every n cycles.

Configuring a send clock for PROFINET IRT

1. In HW Config, open the **Domain Management** dialog box.

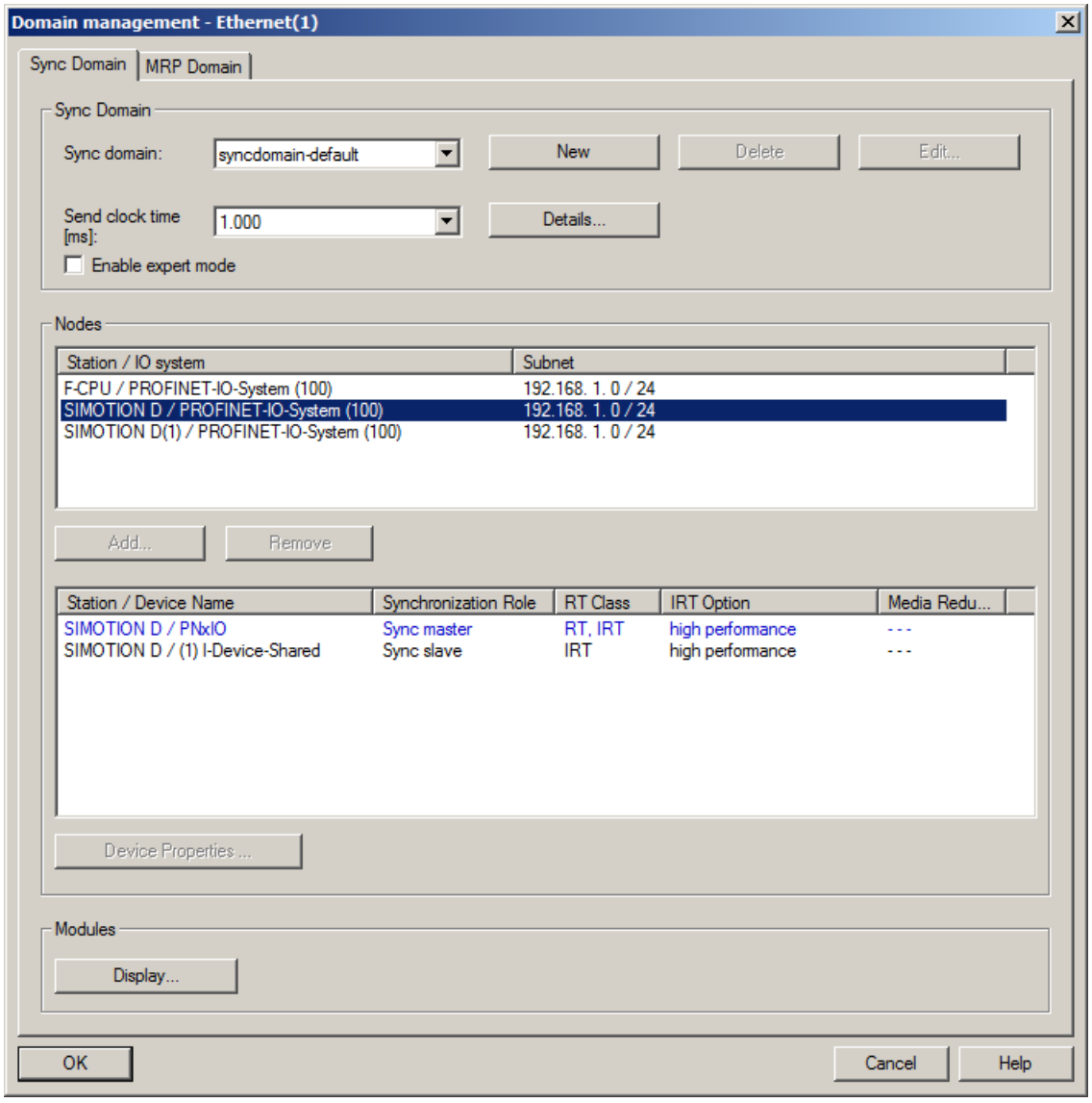


Figure 4-379 Domain Management

2. Select an appropriate send clock for the process. The transmission cycle clock is the smallest possible transmission interval. The send clock is preset to 1 ms.

Note

Communication should be no faster than required, irrespective of what the maximum communication speed may be. This will reduce the bandwidth requirement and the relieve the load that the devices need to support.

Configuring a send clock for PROFINET RT

1. Double-click on the PN interface. The **Properties** dialog box opens.
2. Switch to the PROFINET tab and select a **send clock**. The send clock can only be set if IO controllers and IO devices are not synchronized.

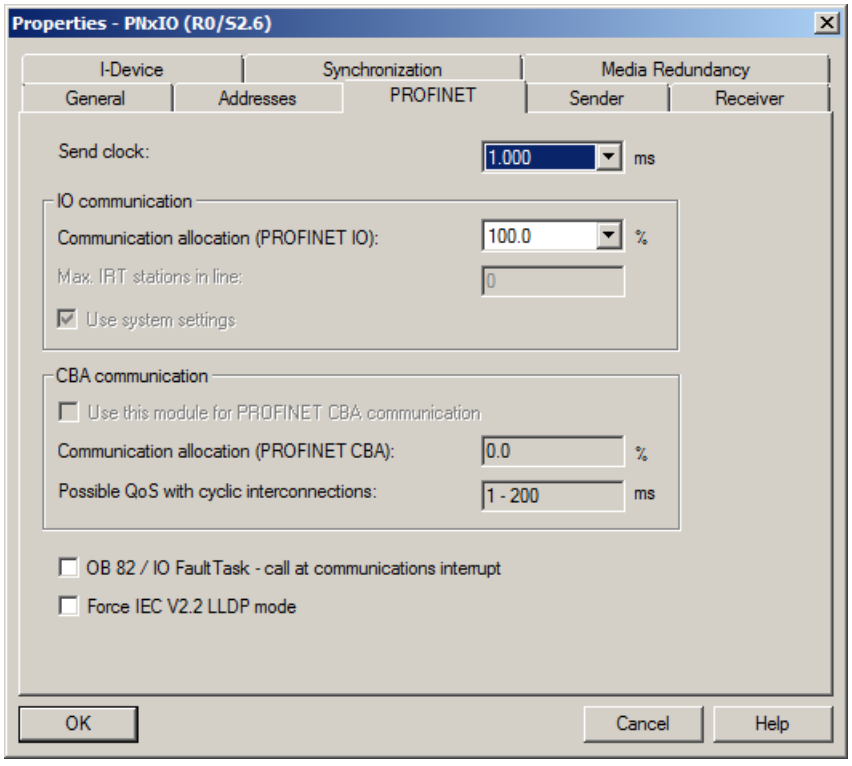


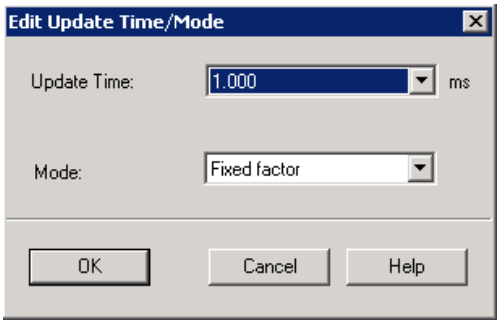
Figure 4-380 Configuring a send clock for PROFINET RT

Defining the update times

The update times for the data exchange of the IO devices is set in the **Properties PROFINET IO System** dialog box.

1. Click the path for the PROFINET IO system in HW Config and select **Object properties** from the context menu. The dialog is displayed.
2. Switch to the **Update time** tab and highlight the respective IO device for which you want to set the update time.

3. Click **Edit**. You can select the refresh time in the **Edit refresh time** dialog.



4. Click **OK** to confirm.

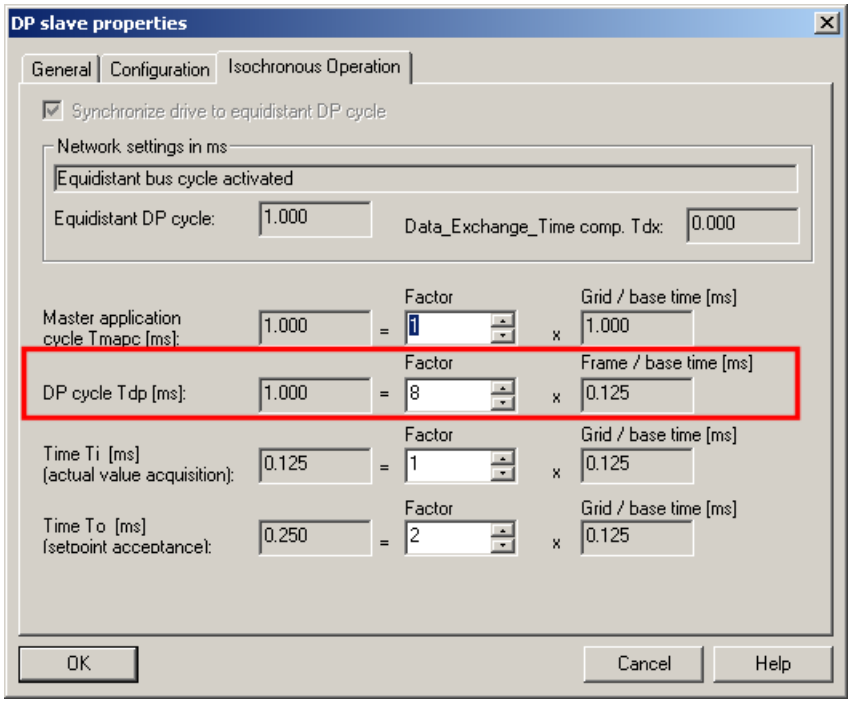
Ratio of PROFINET IRT send clock and DP cycle on SINAMICS_Integrated (SIMOTION D)

If a SINAMICS drive is being operated on a SIMOTION D via PROFINET IO IRT High Performance, the DP cycle of the SINAMICS_Integrated must be the same as the position control cycle clock. The DP cycle is set to 3 ms by default. In most cases, this will not be the same as the servo cycle clock selected for PROFINET applications. If the DP cycle does not correspond to the position control cycle clock, an error message is generated in SIMOTION SCOUT during the consistency check.

You set the DP cycle in the DP slave properties:

1. Double-click SINAMICS_Integrated in HW Config. The **DP slave properties** window appears.
2. Switch to the **Isochronous Operation** tab and activate the "Synchronize drive to equidistant DP cycle" checkbox.

3. Set the factor for **DP cycle Tdp**. The DP cycle must be the same as the position control cycle clock. For example, if the position control cycle clock is 1 ms, you will need to enter a factor of 8 (8 x [base time of 0.125 ms] = 1 ms).



4. Click **OK** to confirm.

When IO devices are operated on PROFINET the position control cycle clock must always correspond to the PROFIBUS cycle clock. The position control cycle clock and the PROFIBUS cycle clock can be scaled to the PROFINET cycle clock.

Example:

PROFINET send clock = 0.5 ms

PROFIBUS cycle clock = position control cycle clock = 1 ms

The PROFIBUS cycle clock can be operated relative to the PROFINET cycle clock at a ratio of 1:1 to 1:64 (max. 8 ms).

Servo_fast, scaling down of cycle clocks to the servo at the PROFINET interface

Cycle clock scaling to the Servo using Servo_fast

The second position control cycle clock enables you to operate two bus systems in different application cycle clocks. An assigned servo cycle clock and IPO cycle clock are available for each of the two application cycle clocks. This enables you to divide your application into slow and fast sections (Servo_fast and IPO_fast).

- The I/O on the fast bus system are used isochronously in the fast Servo_fast/IPO_fast.
- The I/O on the slow bus system are used isochronously in the slow Servo/IPO/IPO_2.
- Servo_fast is only intended to be used with fast IOs and drives.

Options available for isochronous use

For isochronous use, the application cycle is set in HW Config on the I/O module:

- The MAPC (Master Application Cycle) is set for DP slave. Only MAPC=1 is supported if you are also using PROFINET IO. All PROFIBUS devices run in the slow position control cycle clock. The master application cycle describes the reduction ratio between the DP cycle and the cycle of a higher-level controller, e.g. the servo clock of a SIMOTION controller.
- The CACF (Controller Application Cycle Factor) is set for I/O devices. In V4.2 or higher you can also select the position control cycle clock (Servo_fast) on I/O devices. If two servos are configured, only CACF = 1 is still allowed as a setting; i.e. send clock onboard PN interface = Servo_fast cycle clock, send clock CBE30-2 = servo clock. The servo must be scaled down to the Servo_fast at a factor of 2, 4, 8, 16, or 32 (as of SIMOTION V4.4).

The following settings are possible:

Product version	Property	Application cycle of the devices on		
		PROFINET IO Integrated (X150)	PROFINET IO (CBE30-2)	PROFIBUS DP
Before V4.2	1 position control cycle clock	--	--	Servo
Additionally for V4.2 or higher	2 servo clocks	Servo_fast	--	Servo
Additionally for V4.3 and higher	2 servo clocks	Servo_fast	Servo	Servo

Configuring the Servo_fast using the example of a D455-2 DP/PN

A precondition is that a D455-2 DP/PN must be configured with PROFINET IO system and PROFIBUS DP. In V4.3 or higher, you can also use a D455-2 DP/PN with two PN interfaces (onboard PN interface and CBE30-2).

1. In HW Config, select the SIMOTION device module, e.g. D455, and choose **Edit > Object properties** in the menu.
2. Activate the checkbox **Use Servo_fast/IPO_fast** on the **Isochronous Tasks** tab. Therefore the used PROFINET IO system is operated isochronously to the Servo_fast.

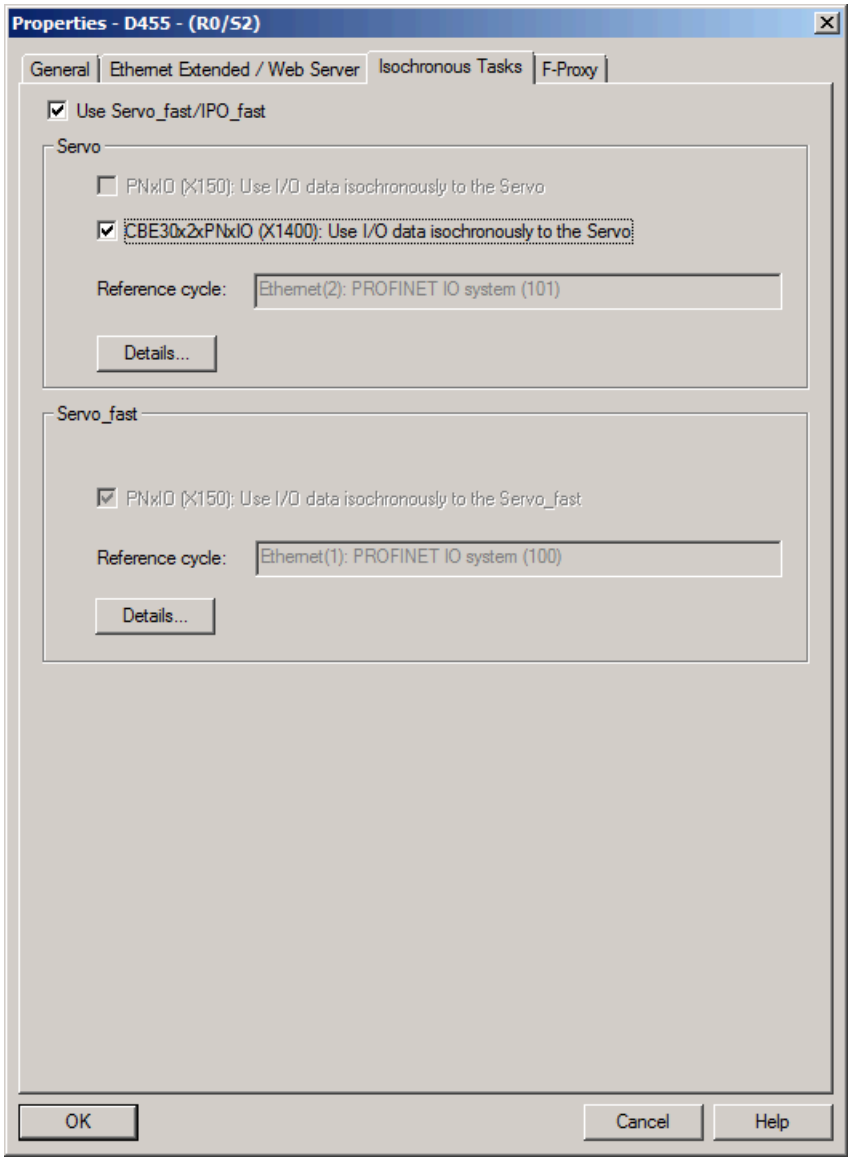


Figure 4-381 Configuring Servo_fast in HW Config

3. Click on the **Details...** button next to the **Servo** field.
4. On the **Isochronous operation** tab of the PROFIBUS DP, enter a factor for the DP cycle Tdp. The servo clock and DP cycle must always be configured the same when using PROFINET.

5. Confirm by selecting **OK** and save and compile the project in HW Config.
6. Switch to SIMOTION SCOUT and select **Set system cycle clocks** in the context menu of the SIMOTION CPU. This displays the values for the servo and for Servo_fast.

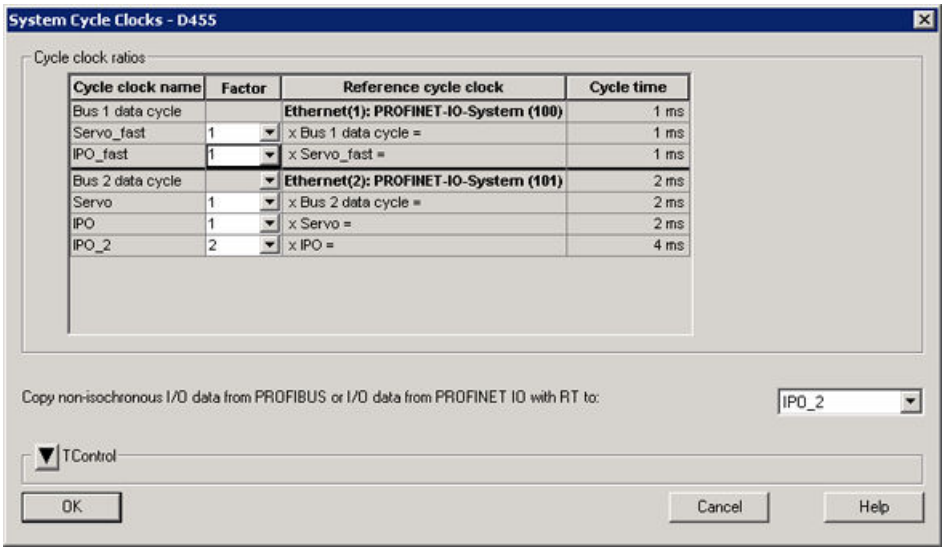


Figure 4-382 Servo and Servo_fast system cycle clocks

Note

For further information, refer to the *SIMOTION SCOUT Basic Functions* Function Manual.

Configuring a topology

Topology

Introduction

With IRT High Performance, the topology must be configured and settings made to determine which device is to be connected via which port to which other devices.

There are two options for defining the properties of the cables between the ports of the switches:

Using the topology editor (Page 2211)

Using the object properties (Page 2212)

Topology editor (graphical view)

Procedure

With the topology editor you have an overview of all ports in the project and can interconnect them centrally.

The topology editor is started with the **Edit > PROFINET IO > Topology** menu command in HW Config or NetPro (PROFINET device must be selected).

The topology editor offers the user two options for displaying the topology graphically (STEP7 V5.4 SP2 or higher) or in a tabular format. The graphic view is more suitable for situations involving interconnection.

Description

In the topology editor, you can:

- Interconnect ports
- Modify the properties of the interconnection
- Add passive components
- Arrange for an offline/online comparison to be displayed in online mode

Procedure

1. In SCOUT, double-click on the SIMOTION module in order to access HW Config.
2. Select the PROFINET module, e.g. PNxIO in the case of a D445-2 DP/PN.
3. Perform **Edit > PROFINET IO > Topology**. The topology editor opens.
4. Click **Graphical view** to bring the tab into the foreground.

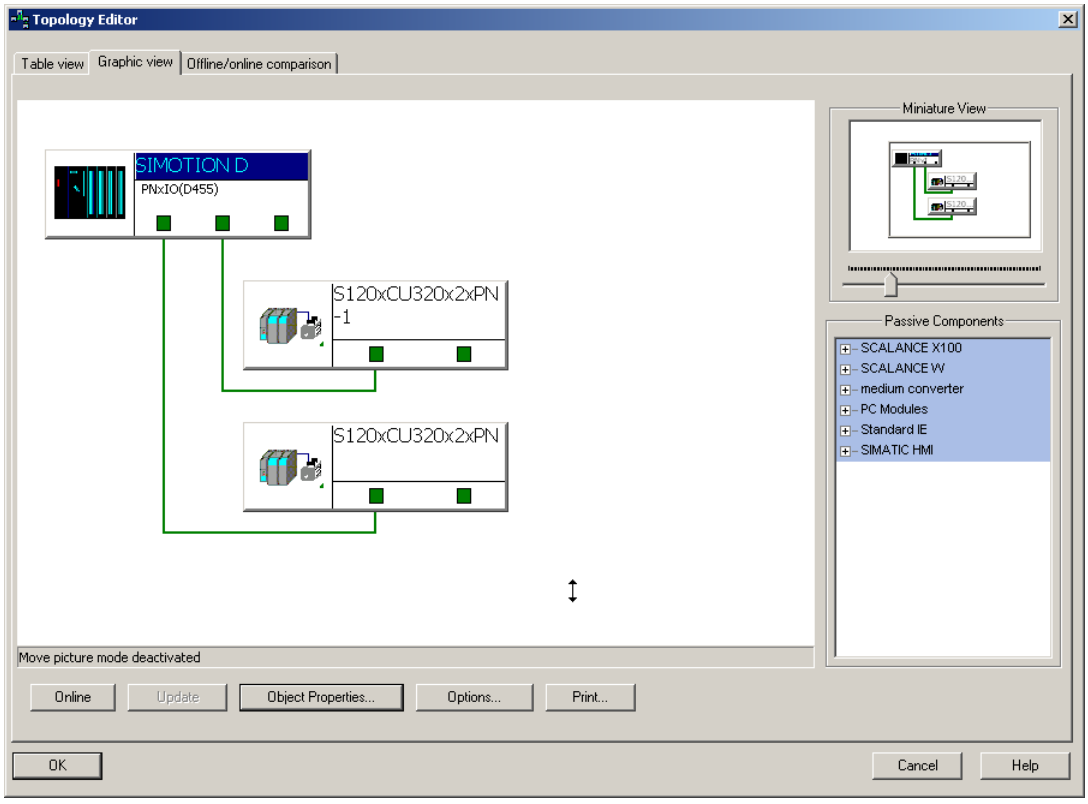
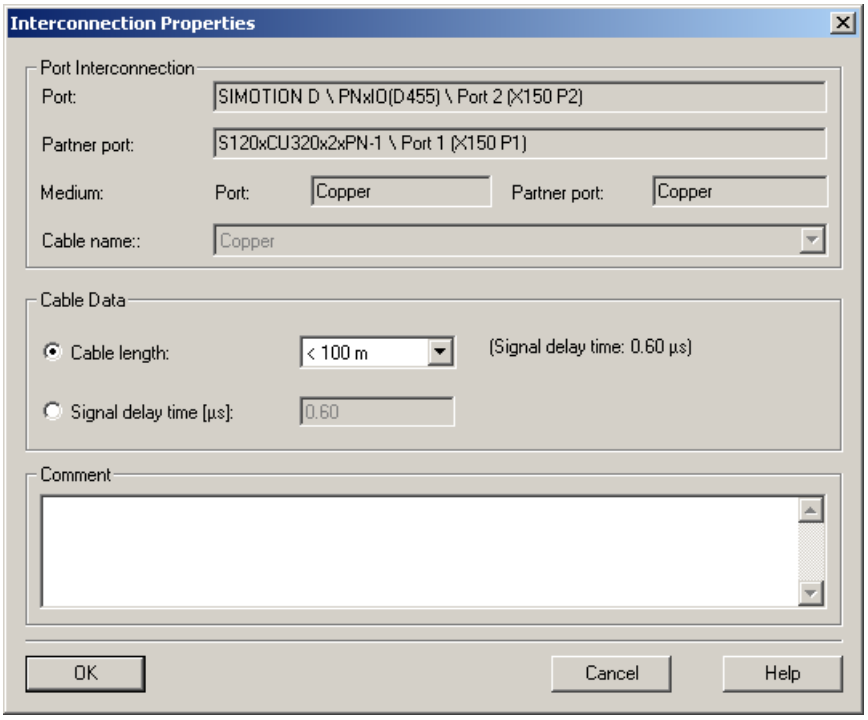


Figure 4-383 Topology editor (graphical view)

5. Establish connections between ports by pressing and holding down the left mouse button and drawing a line between the two ports to be connected. The **Interconnection Properties** window opens.
6. The port interconnection is displayed: You can configure the cable data: A cable length of <100 m is set as standard. This should not be modified under normal circumstances. Alternatively, you have the option of specifying a signal propagation delay; e.g. the latency time when using slip rings.



7. Click **OK** to confirm.

Offline/online comparison

When you switch to online mode, the topology in the editor is compared with the actual topology. Any components which are not recognized are highlighted by a question mark, while connections and components in RUN are highlighted in green.

Interconnecting ports via the topology editor (table view)

Procedure

Ports can be interconnected in the **Tabular view** of the topology editor.

The topology editor is started with the **Edit > PROFINET IO > Topology** menu command in HW Config or NetPro (PROFINET device must be selected).

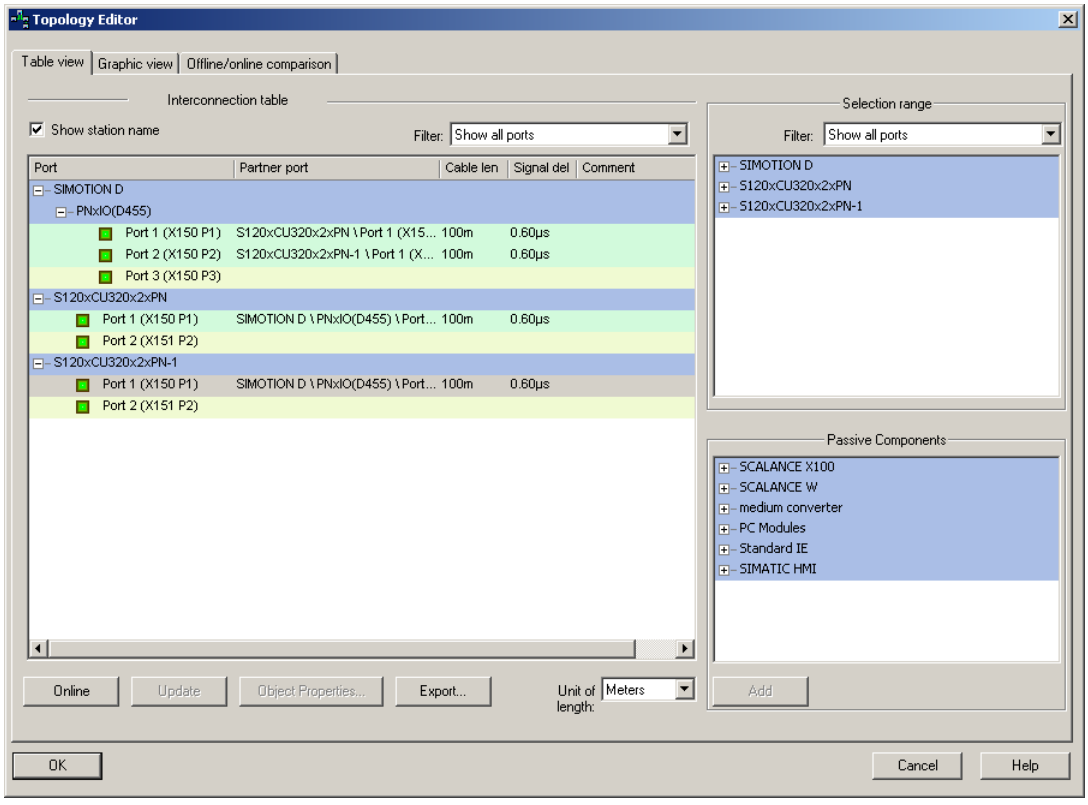


Figure 4-384 Topology editor

All configured PROFINET IO devices with their ports are listed in the interconnection table on the left-hand side. You can use the Filter dialog to select whether all ports, only the ports that have not yet been interconnected or only the ports that have already been interconnected are to be displayed.

Interconnecting ports in the tabular overview

1. To interconnect ports of different devices, select the port of a device that you want to interconnect in the right-hand field.
2. Drag this port to the desired port of a device in the interconnection table. The "Interconnection Properties" dialog opens.
3. Configure the cable data. A cable length of < 100 m should be set by default.
4. Confirm your entries with **OK**.

Interconnecting ports via object properties

Alternatively, a partner port can be selected via the properties of a port. Thus, the cable between two ports is defined and the properties of this cable can be edited.

Procedure

1. The dialog box is opened in HW Config by selecting a port on the module and selecting the **Edit > Object properties** menu command or double-clicking on the port.

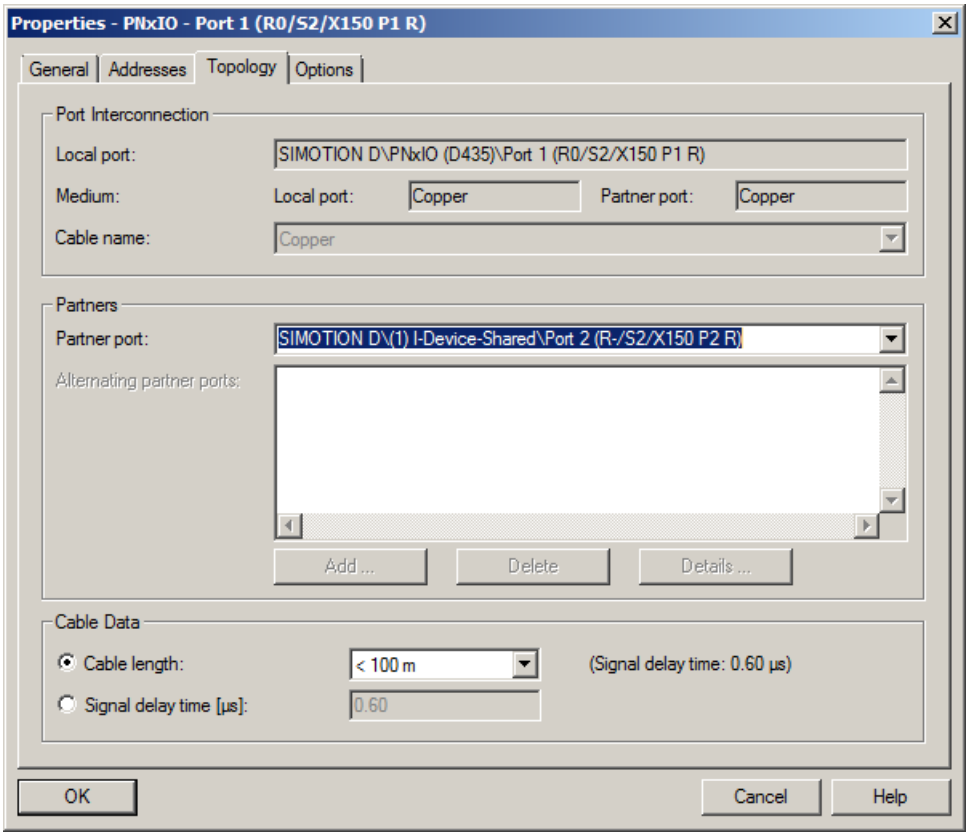


Figure 4-385 Object properties Topology

2. Then, select the **Topology** tab in the **Properties port...** dialog.
3. In the **Partner port** list, select the port with which you want to interconnect the currently selected port.
4. Confirm your entries with **OK**.

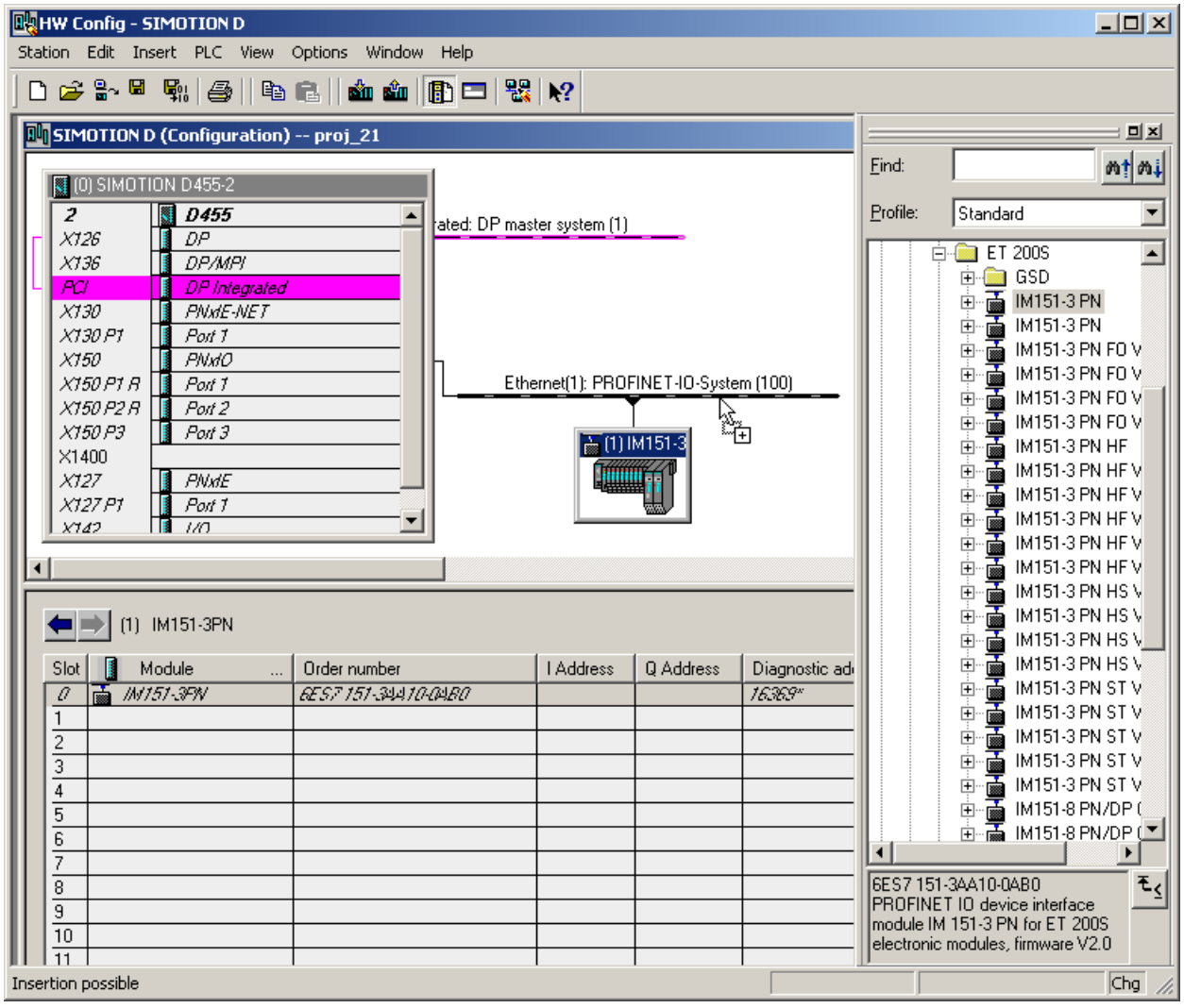
Creating an IO device

Requirement

You have already created a PROFINET IO system and configured a PROFINET IO module, e.g. SIMOTION D455-2 PN (see Inserting and configuring SIMOTION D4x5-2 DP/PN/D410 PN/D410-2 DP/PN (Page 2186)).

Procedure for PROFINET IO devices using the hardware catalog

1. Double-click the corresponding module in SIMOTION SCOUT to open HW Config.
2. Under PROFINET IO in the hardware catalog, select the module you wish to connect to the PROFINET IO system.
3. Drag the module to the path of the PROFINET IO system. The IO device is inserted.



4. Save and compile the settings in HW Config.

Procedure for third-party manufacturer PROFINET IO devices

1. Double-click the corresponding module to open HW Config.
2. Select the **Options > Install GSD files** menu command.
3. Select the GSD file to be installed in the **Install GSD Files** dialog box.
4. Click the **Install** button.
5. Close the dialog box by clicking the **Close** button.

6. Under PROFINET IO in the hardware catalog, select the module you wish to connect to the PROFINET IO system.
7. Drag the module to the path of the PROFINET IO system. The IO device is inserted.
8. Save and compile the settings in HW Config.

Inserting and configuring the SINAMICS S120

Requirement

You have inserted a SIMOTION D4x5-2 with integrated PROFINET interface in your project, a PROFINET IO subnet has already been created and the Sync-Master is configured.

Note

The configuration is almost identical for all SINAMICS S120 and SIMOTION D devices.

Procedure

1. Select **PROFINET IO > Drives > SINAMICS** in the HW Config hardware catalog. Then select the module, e.g. **SINAMICS S120 CU320-2 PN**.
2. Click the entry and drag the drive, e.g. **V4.4** , to the PROFINET IO subnet. The **Properties - Ethernet Interface SINAMICS-S120xCU320x2xPN** window opens.
   A suggested IP address will already be displayed here and the subnet will be selected.
3. Click **OK** to accept the setting.
4. In HW Config, select **Station > Save and Compile Changes**.

Setting a telegram in SIMOTION SCOUT

For V4.2 and higher, the telegrams are assigned automatically in SIMOTION SCOUT by means of symbolic assignment. For isochronous communication, a telegram must be configured (e.g. telegram 105) which enables the drive to be synchronized with PROFINET. The telegram is thus automatically set in SCOUT during the configuration of the drive unit and following the assignment of an axis.

1. If you have not yet configured a supply and drive, click **Configure drive unit** in the SCOUT project navigator and drive unit.
2. Run the drive unit configuration wizard.
3. After you have closed the wizard, click **Communication > Telegram configuration** under the drive unit in the project navigator. Tab **IF1: PROFIdrive PZD telegrams** contains a list of telegrams. A telegram is not yet configured following commissioning of the drive since the drive has not yet been symbolically assigned.
4. Therefore, insert a new axis TO and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding telegram will automatically be created (symbolic assignment).

5. Select **Project > Save and compile all** from the menu. This means that the addresses will be set up automatically with symbolic assignment. You can also trigger symbolic assignment of addresses using the menu item **Project > Set up addresses**.
6. Once the axis configuration process is complete, click **Communication > Telegram configuration** under the drive unit in the project navigator. Tab **IF1: PROFIdrive PZD telegrams** now lists the telegram for the drive (e.g. SIEMENS telegram 105).

Telegram in HW Config

Alternatively, you can assign the telegram in HW Config. This is only possible if the drive unit and the drive have already been configured in SCOUT. You will then be able to proceed as follows in HW Config.

Note

Symbolic assignment in SIMOTION SCOUT is recommended for telegram interconnection. For this purpose, make sure that a check mark is applied for **Project > Use symbolic assignment** in the menu.

If you are using the symbolic assignment, SIMOTION SCOUT is entirely responsible for managing the telegram between SIMOTION and the drive DOs. In other words, SIMOTION SCOUT independently creates telegrams that all contain the necessary signals according to the technological configuration. SIMOTION SCOUT automatically manages the positioning of signals in the telegram and its size; the user is no longer able to influence this process or make any changes.

1. Select the inserted SINAMICS drive and double-click the entry **SIEMENS / Standard telegram xx** in the lower table for the drive.
   The **Properties SIEMENS / Standard telegram xx** dialog box is called.
2. Select the corresponding telegram. After it is saved, the telegram can be also be selected in SCOUT, under **<"Drive_device_xx"> - Communication > Telegram configuration** in the project navigator. Alignment with HW Config is possible.

Settings for isochronous mode (V4.4 or higher)

For isochronous mode, the drive unit must be synchronized with the PROFINET cycle clock. You can make these settings at the PN interface of the drive unit and on the SIMOTION device (Sync-Master).

Settings on SIMOTION device (Sync-Master)

1. In HW Config, select the SIMOTION device module, e.g. D455, and choose **Edit > Object properties** in the menu.
2. Click the **Details...** button on the **Isochronous tasks** tab.
   This button only appears if you select the **...Assign I/O device isochronous** check box.

3. Select the **Automatic** setting in the **Details for servo** dialog box under **Ti/To mode**. The cycle clocks and time constants for all IO devices (Sync-Slaves) on the PROFINET are then calculated automatically by the system.

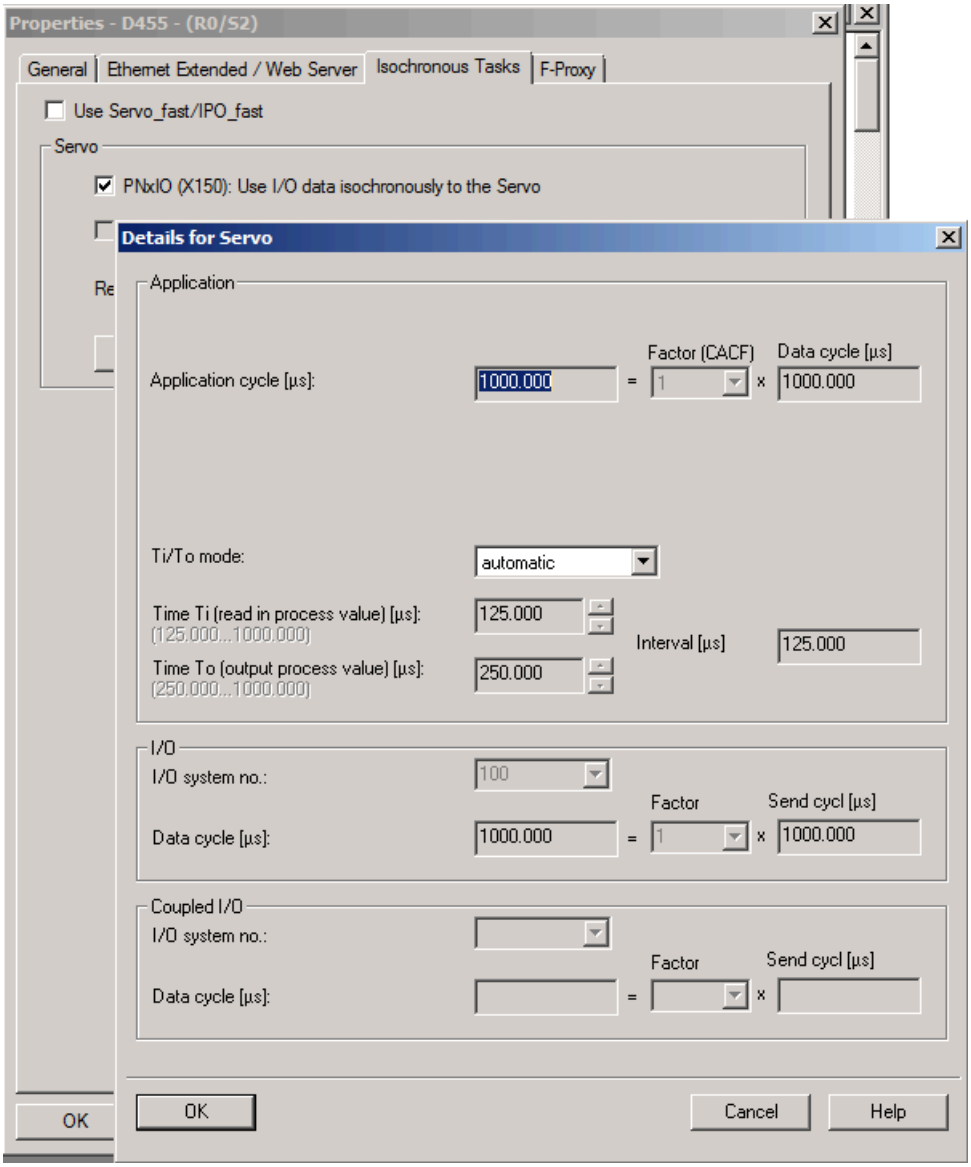


Figure 4-386 Configuring an isochronous task on the SIMOTION device.

4. Click **OK** to close both dialog boxes.

Settings on the SINAMICS drive (Sync-Slave)

1. Select the SINAMICS drive on the PROFINET IO system and double-click on the entry of the PROFINET interface in the lower table, e.g. PN-IO.
   The **Properties PN--IO** dialog box is displayed.
2. Select the **Servo** entry on the **IO cycle** tab under **Assign IO device isochronously**. The drive is operated isochronously. The time constants are calculated automatically.

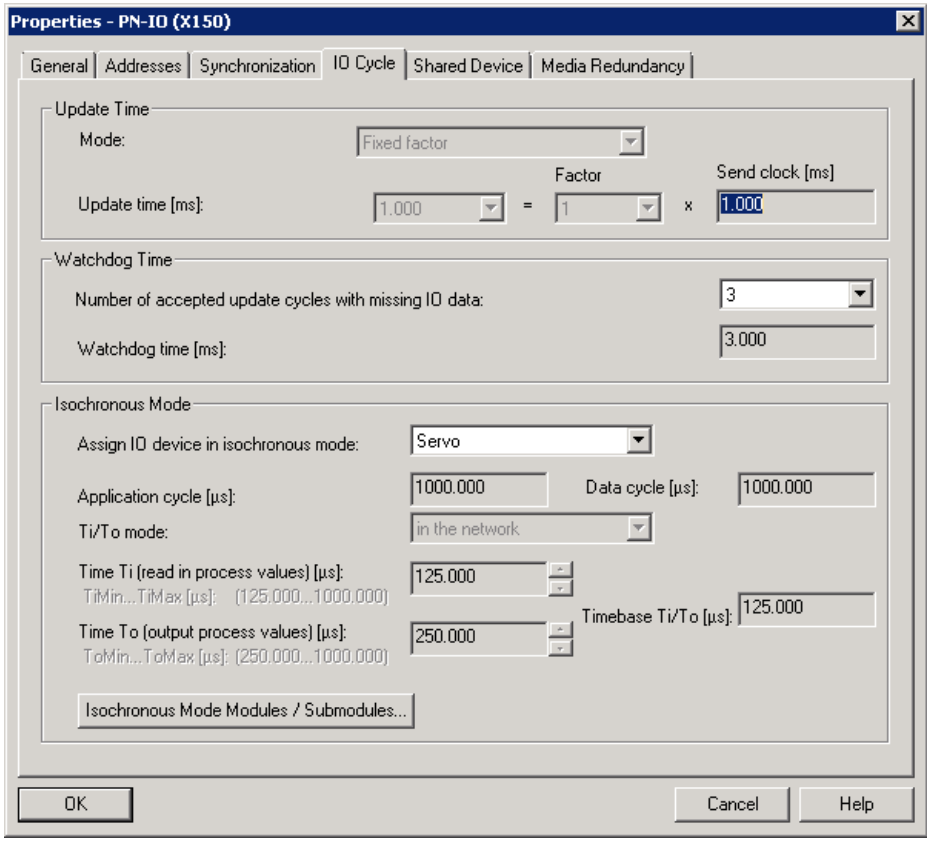


Figure 4-387 Configuring an isochronous task on the PROFINET IO device

3. You can see the current cycle clock in **Application cycle**. To configure cycle clock scaling, you need to configure down-scaling by selecting **Set system cycle clocks** in the **execution system**. This is required if the servo cycle clock should be slower than the PROFINET IRT send clock, for example. However, the servo cycle clock must already have been taken into account in the PROFINET IRT send clock.
4. If necessary, switch to the **Synchronization** tab to select the **Synchronization type**, **Sync slave** in this case. This can also be set under **Domain Management**.
5. Confirm the entries with **OK**.

Note

You must carry out further steps in order to include the drive in the sync domain (see Creating a sync domain (Page 2199)), load the IP address to the drive (see Assigning device names and IP addresses to IO devices (Page 2221)), and interconnect the ports (see Interconnecting ports via the topology editor (table view) (Page 2211)).

IP address and device name

Introduction

Devices and controllers have a fixed MAC address, a configurable IP address, and a device name. The IP address, subnet mask, and device name are defined within the Ethernet interface properties in the engineering software. This ensures the devices within the project have their own unique assignment. The device name is referred to as **NameOfStation** in accordance with the standards or, in the case of devices with multiple interfaces, **NameOfInterface**. The device name must be unique for the communication.

You can find the device name in the **Properties - Interface** dialog by double-clicking on the device or the controller. Click on **Properties** in the dialog to define the IP address. See also Add and configure PROFINET interface CBE30-2 (Page 2191).

Device name guidelines

In the case of an IO device, the device name is, for example, the drive name *Drive1*. In the case of controllers, the device name is the name of the PROFINET interface, e.g. *PNxIO* for an integrated interface of SIMOTION D. A controller can also have multiple interfaces and, therefore, multiple device names.

Device names are subject to certain syntax rules, i.e. they must always be DNS-compliant. For SIMOTION and SIMATIC, there are additional boundary conditions for the device names.

- Letters a-z and numbers 0-9 may be used.
- No German umlauts (ä, ö, ü)
- The special characters minus (-) and period (.) may be used
- No other special characters or spaces ! " § $ % & / ( ) = ? * ' _ : ; > < , # + | ~ \ }] [{
- The period is used for structuring, i.e. the device name can comprise several parts (labels)
  - Label.Label.Label.Label
  - Periods are used as separators.
  - A label must not begin with a hyphen (-) or a period (.)
  - A label must not exceed 63 characters
- The maximum total length of the name is 240 characters (including hyphen and period)
- The device name must start with a letter, but not with the character string "port-xyz-" (x, y, z = 0..9).
- **No** distinction is made between upper-case and lower-case letters. The configured device name can also contain upper-case letters but these will be saved on the device as lower-case letters.
- Up to SIMOTION SCOUT V4.3, the device name must not include hyphens (-), i.e. the name must comply with the ST naming convention. As of SIMOTION SCOUT V4.3, this restriction no longer applies.

Initialization of the controller and devices in online mode

As the controllers and devices do not have a device name or IP address when delivered, these will need to be assigned at the outset. When assigning addresses (i.e. the IP address and device name - a process also referred to as initialization), a distinction is made between controllers and devices. You can adopt different approaches when initializing devices and controllers.

Controller initialization

- Download the application
- Engineering software
  - HW Config, NetPro, SCOUT
  - Primary Setup Tool (PST)
  - PRONETA
- Via the application (the _setPnNameOfStation system function for SIMOTION (as of V4.4)

Note

When using engineering software for initialization, and particularly where larger systems are involved, you should establish direct connections with the device to ensure the device to be initialized is clearly identifiable. Alternatively, the Flash feature can be used, whereby devices can be identified by a flashing LED.

Device initialization

- Engineering software
  - HW Config, NetPro
  - SCOUT, STARTER
  - Primary Setup Tool (PST)
  - PRONETA
- Write to the MMC or CF card beforehand and then plug in.

Topology-based initialization for devices

Devices can also be initialized without an MMC or CF card. This method is known as topology-based initialization

Note

For topology-based initialization, the PN interface of the device must be in its factory setting.

See also

Creating an I device (Page 2245)

Assigning device names and IP addresses to IO devices

Introduction

You can only go online with the engineering software of a device (or controller) if it has been assigned an IP address in the engineering software. Devices must also be assigned a device name which is unique across the network. This enables the controller to identify the devices assigned to it.

You can specify the IP address in the **Properties - Ethernet interface....**dialog (double-click on the device to open this). Moreover, a default name is entered that you can modify. The default setting **Assign IP address through Controller** is active. In other words, when powering up, the controller identifies the devices assigned to it via the device names and then assigns the IP address defined in Engineering to them. We recommend leaving this function activated.

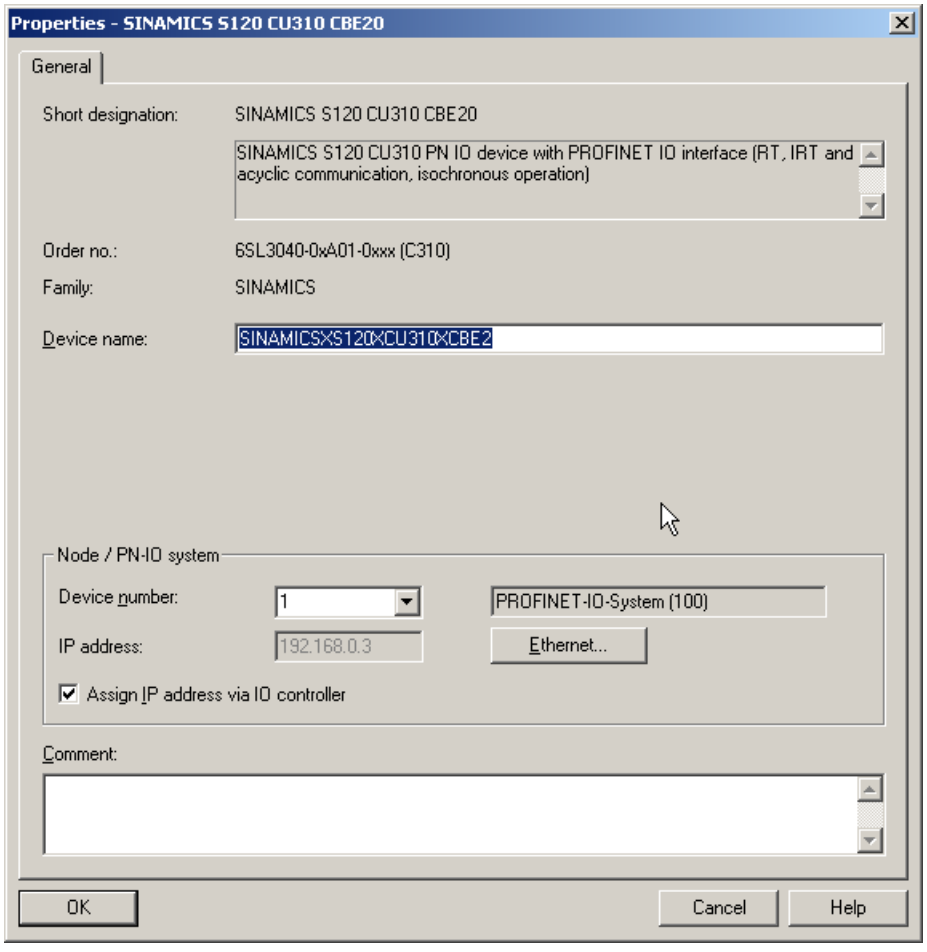


Figure 4-388 Properties SINAMICS S120

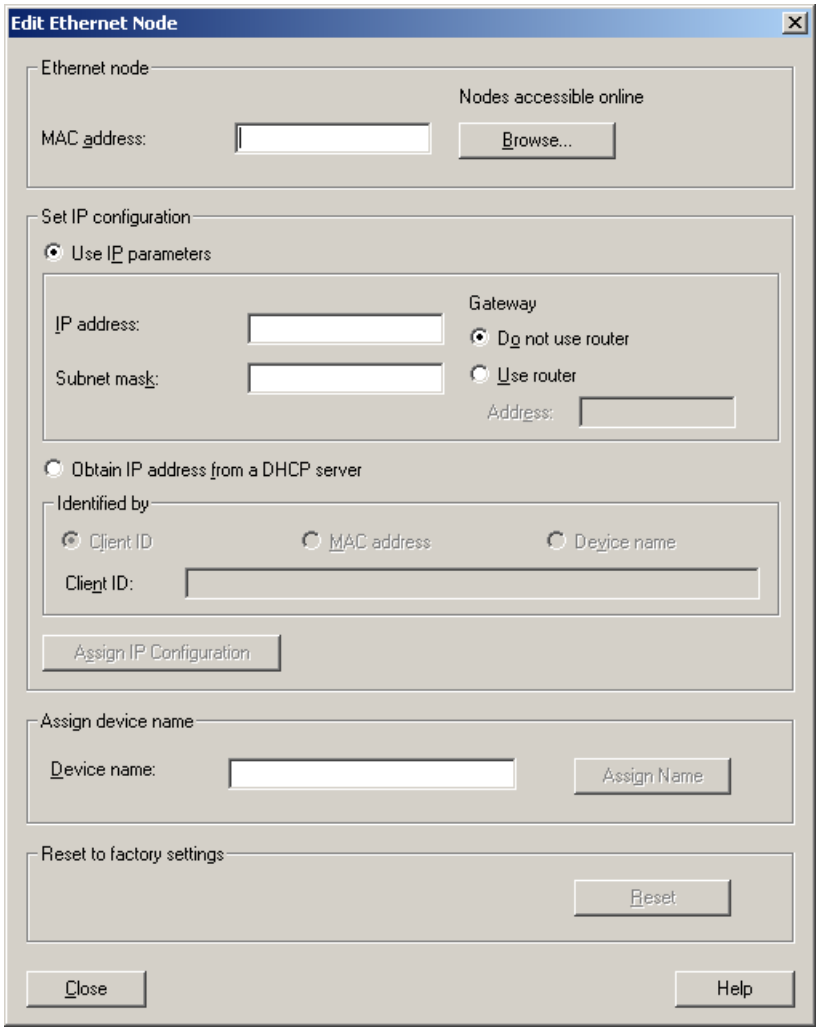
IO device initialization

Note

If you are connecting the programming device/PC directly to the device's PROFINET interface, you can use a patch or crossover cable. During commissioning, we recommend that the device to be initialized is connected directly to the programming device/PC.

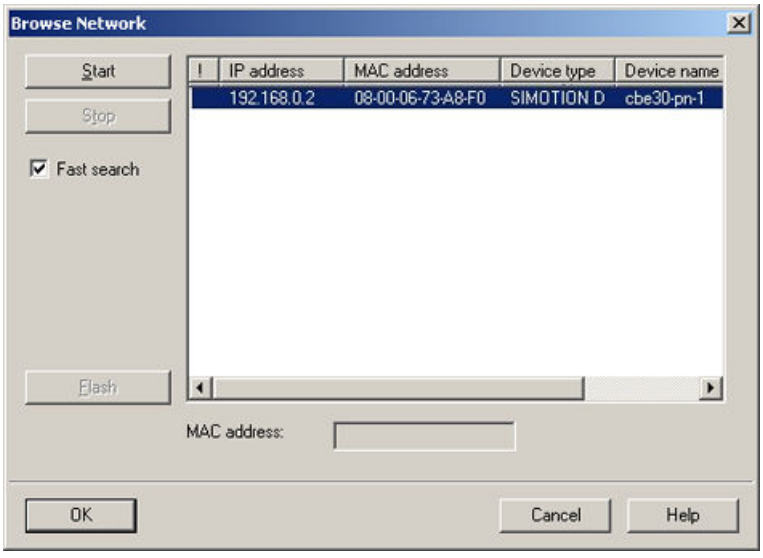
Alternatively, the Flash feature can be used, whereby the IO device can be identified by a flashing LED.

1. In HW Config or NetPro, select the **Target system - Ethernet - Edit Ethernet Node** menu item. The **Edit Ethernet node** dialog box is displayed.



2. Click the **Browse** button.

3. The **Browse Network** dialog box opens. The connected nodes are displayed.



4. Click the device to be initialized and confirm with **OK**.
5. Enter the IP address and subnet mask you specified in the **Properties – Ethernet interface ...** dialog.
6. The default setting (**Do not use router**) for the gateway remains unchanged.
7. Click on the **Assign IP configuration button**. The IP address is then assigned to the device online.
8. Enter the device name that you have defined in HW Config, see figure **Properties SINAMICS S120**.
9. Click on the **Assign Name** button. The device name is assigned to the device.

As an alternative, you can perform node initialization in SIMOTION SCOUT.

You can also perform the node initialization in SCOUT.

- In SCOUT, execute **Reachable nodes** and, in the dialog box displayed, right-click the device that you want to edit.
- Execute **Edit Ethernet nodes**. The corresponding dialog box is displayed.

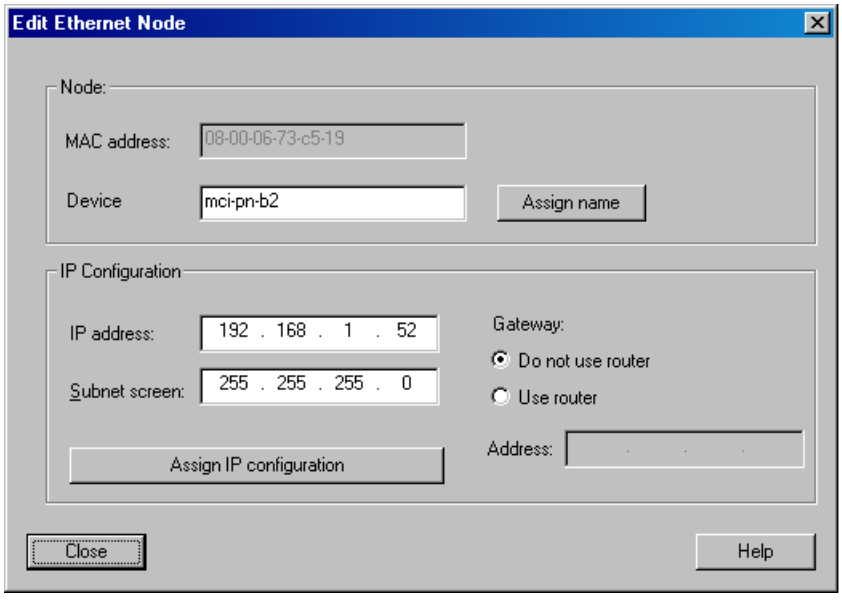


Figure 4-389 Edit Ethernet nodes

- Enter a device name, a subnet mask and an IP address.
- Confirm your entries.

The device name and IP address are transferred to the device and stored there.

Configuring media redundancy (V4.3 and higher)

Creating ring topology

Setting up ring topology

In order to use MRP, you must establish a ring topology. The ring nodes must support media/ system redundancy.

Note

Devices that support MRP

An overview of all devices that support media and/or system redundancy can be found in the Support area (https://support.industry.siemens.com/cs/ww/en/view/67364686).

To establish the ring, bring together the ends of the line topology into one device.

Requirement

You have created a project in SCOUT and added a SIMOTION module.

Setting up devices in HW Config

1. Open HW Config.
2. Set up a PROFINET IO system in the SIMOTION module.
3. Add a module, such as a SCALANCE switch, that can take the role of Redundancy Manager.
4. Then add the other required SIMOTION modules (or other modules). The modules assume the role of redundancy clients.

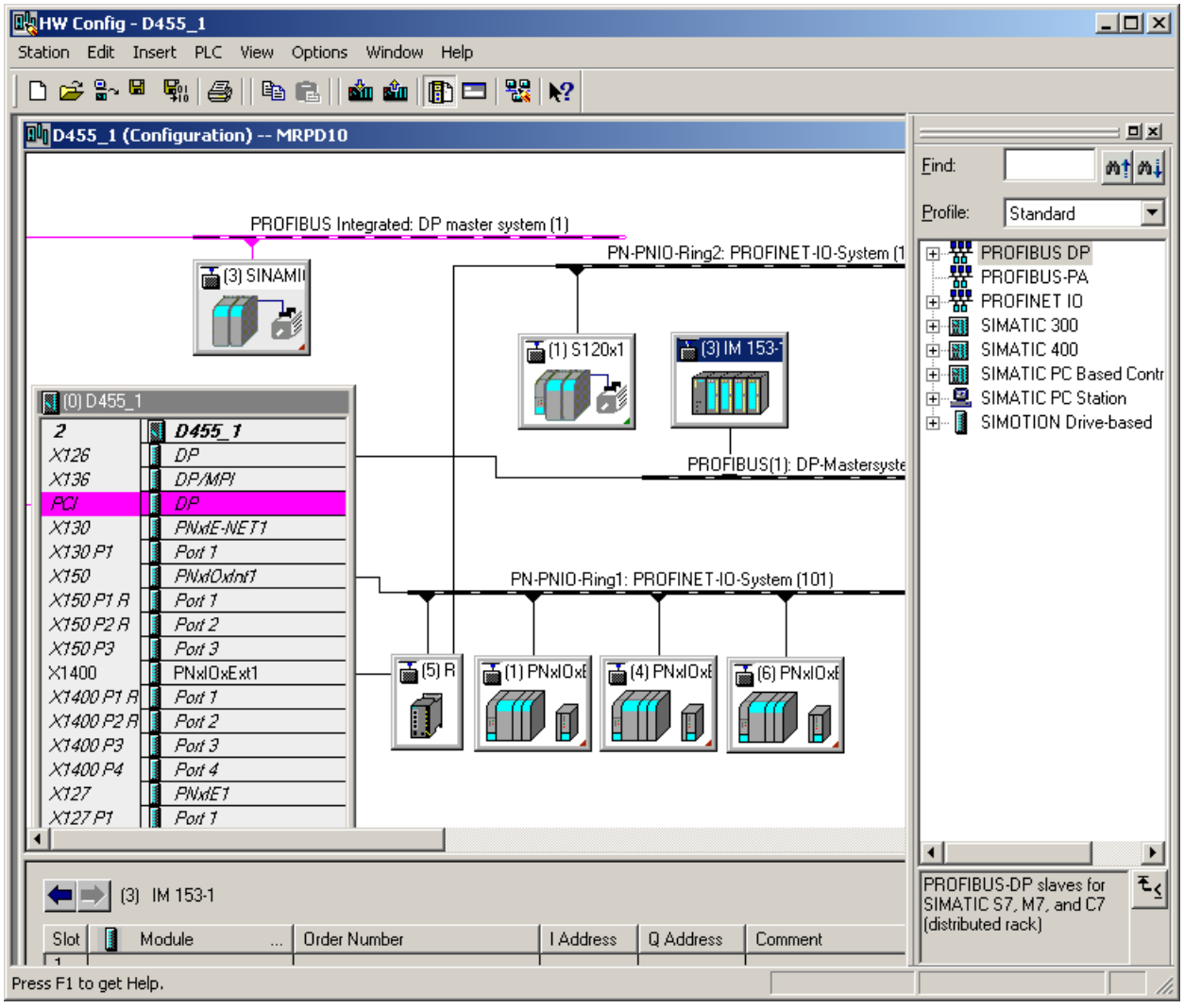


Figure 4-390 Ring project in HW Config

Establishing ring topology in the Topology Editor

1. Open the Topology Editor.
2. Interconnect the individual ports of the devices.
3. If necessary, change to graphic view to drag the connections with the mouse.
4. Connect the two end points of the line topology with each other in order to close the ring, e.g. the redundancy manager with a redundancy client.

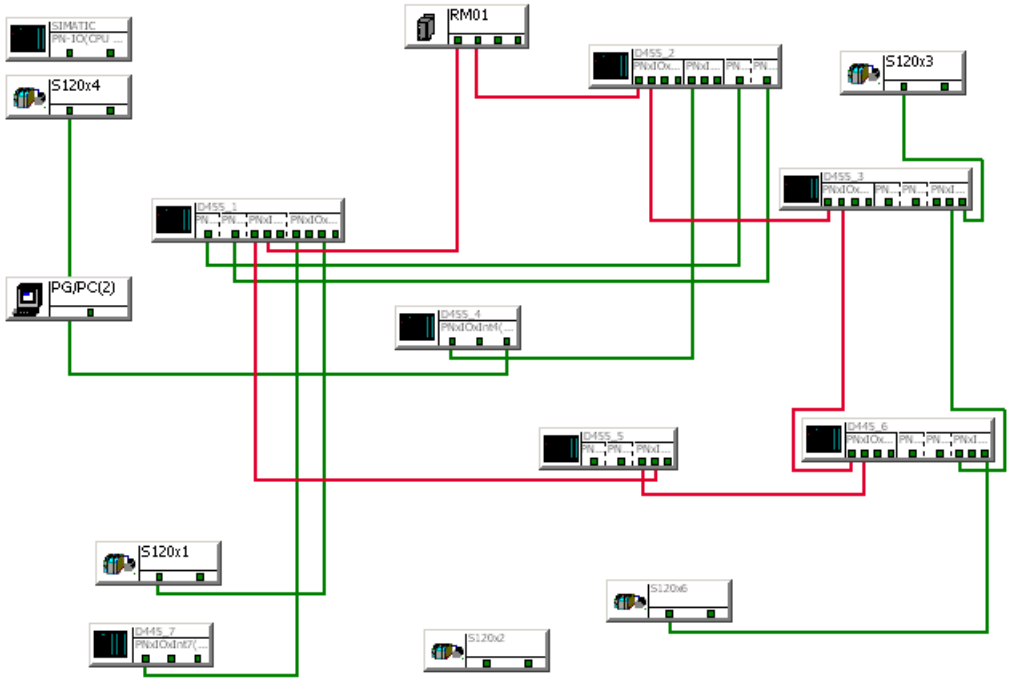


Figure 4-391 Ring topology in the Topology Editor

Setting up MRP domain

MRP domain

An MRP domain containing all ring nodes is required for media redundancy. If you are using several ring topologies, you must set up the corresponding number of MRP domains.

Setting up MRP domain

In order to set up an MRP domain, proceed as follows:

1. Select the PROFINET IO system in the working area of HW Config.
2. Go to **Edit > PROFINET IO > Domain Management**.
   The **Domain Management** dialog opens.

3. Switch to the **MRP domain** tab.

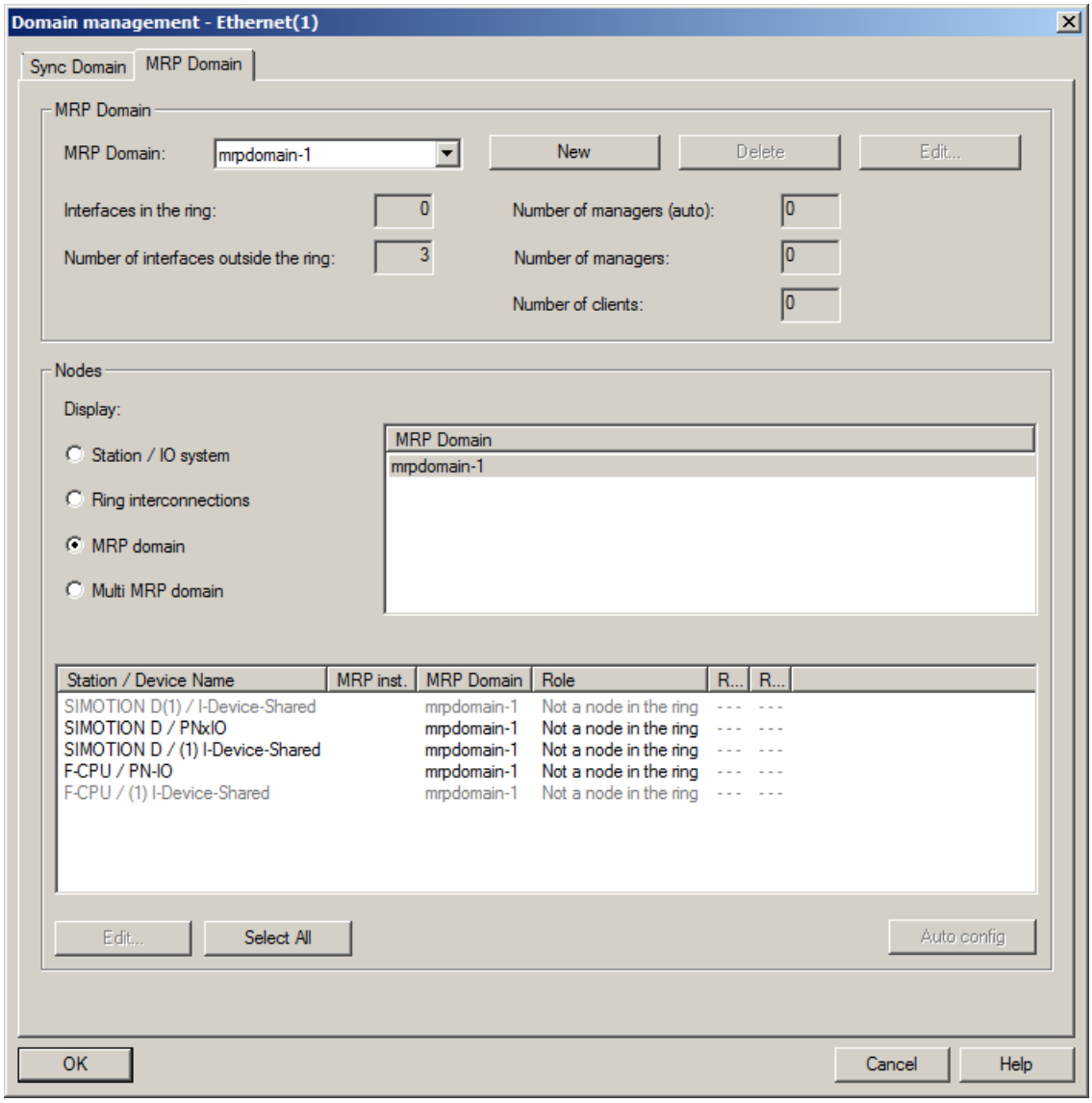


Figure 4-392 Domain Management MRP domain

4. Click **New** to create a new MRP domain. By default, the domain mrpdomain-1 will be created.

Assign the individual devices to the domain according to their roles.

1. Select the device under **Station/device name**and click on **Edit**.
   The "Edit Media Redundancy" dialog box opens.

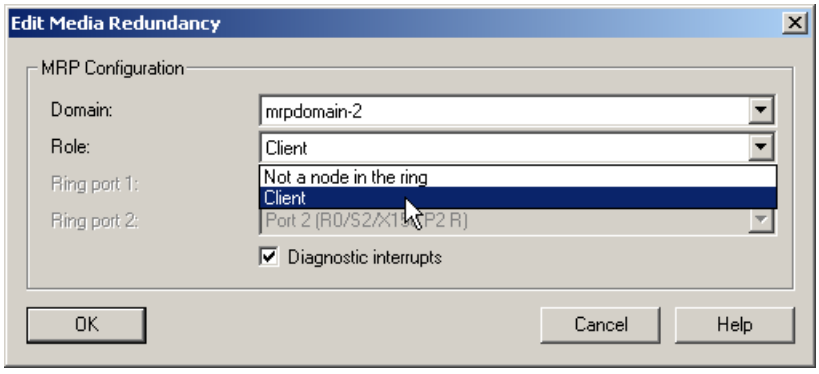


Figure 4-393 Edit media redundancy

2. Select the role intended for the device, e.g. Client, under **Role**.
3. Click OK to accept the settings.
4. Repeat the steps for all MRP domains and devices.

Result

The MRP domains have been created and the corresponding roles have been assigned to the members of the individual domains.

Configuring media redundancy

Entering settings for media redundancy

For media redundancy, you must assign the various roles and an MRP domain to the nodes.

Configure MRPD

If all participants within the MRP domain work in IRT mode (IRT with High Performance), MRPD is automatically activated for all nodes of the IRT domain that communicate via the ring.

Requirements

You have set up the devices in HW Config.

Procedure for the Redundancy Manager

To configure the Redundancy Manager, proceed as follows. Only SCALANCE switches can be Redundancy Managers.

1. Select the SCALANCE switch in the working area of HW Config.
2. Double-click on the PN interface in the detailed view; for example, on X1 in a SCALANCE switch.
   The **Properties - PN (xx)** dialog opens.

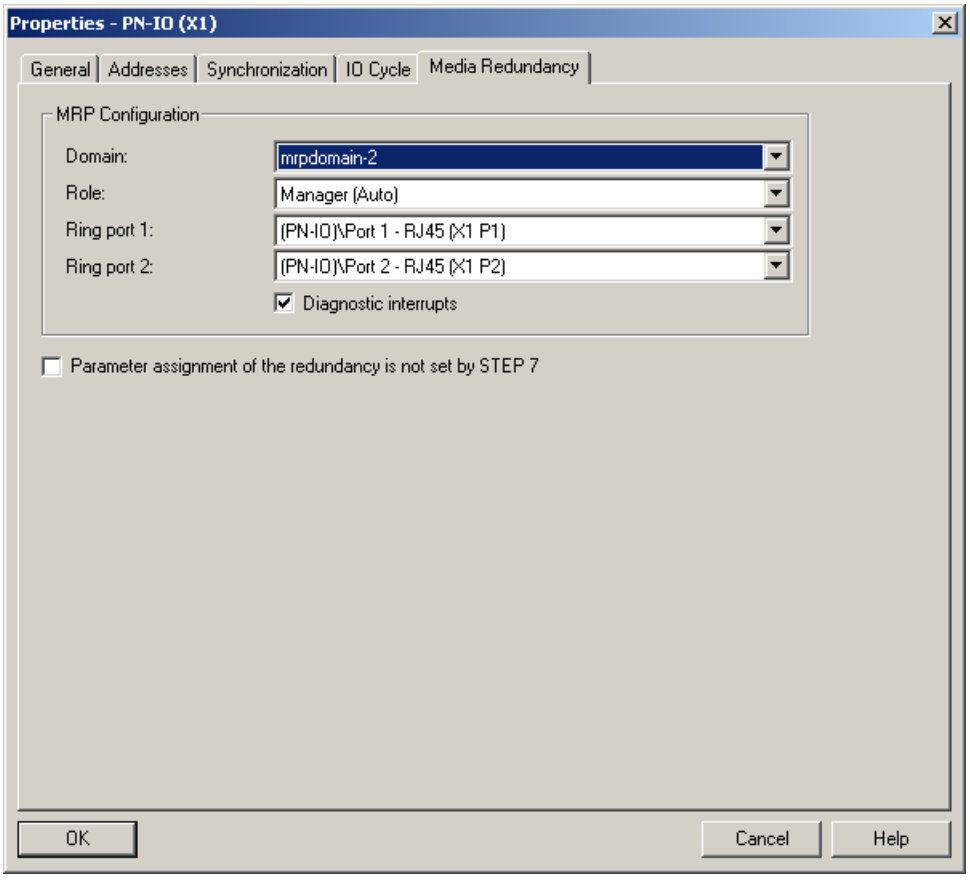


Figure 4-394 Properties Redundancy Manager

3. Switch to the **Media Redundancy** tab.
4. Select the MRP domain under **Domain**. If you are using only one domain, this is pre-assigned.
5. Under **Role** select the entry **Manager (Auto)**.
6. Under **Ring port 1** and **Ring port 2**, select the ports which are to be used in the ring (only with the SCALANCE switch).
7. Activate the **Diagnostic interrupts** option if you want to see ring communication faults in the diagnostics buffer and in the device diagnostics.

Procedure for the redundancy client

In order to configure devices as redundancy clients, proceed as follows:

1. Select the device in the work area of HW Config.
2. In the detailed view of the device, double-click on the interface line, e.g. on X1400 for a SIMOTION controller.
   The **Properties - Interface (xx)** dialog opens.

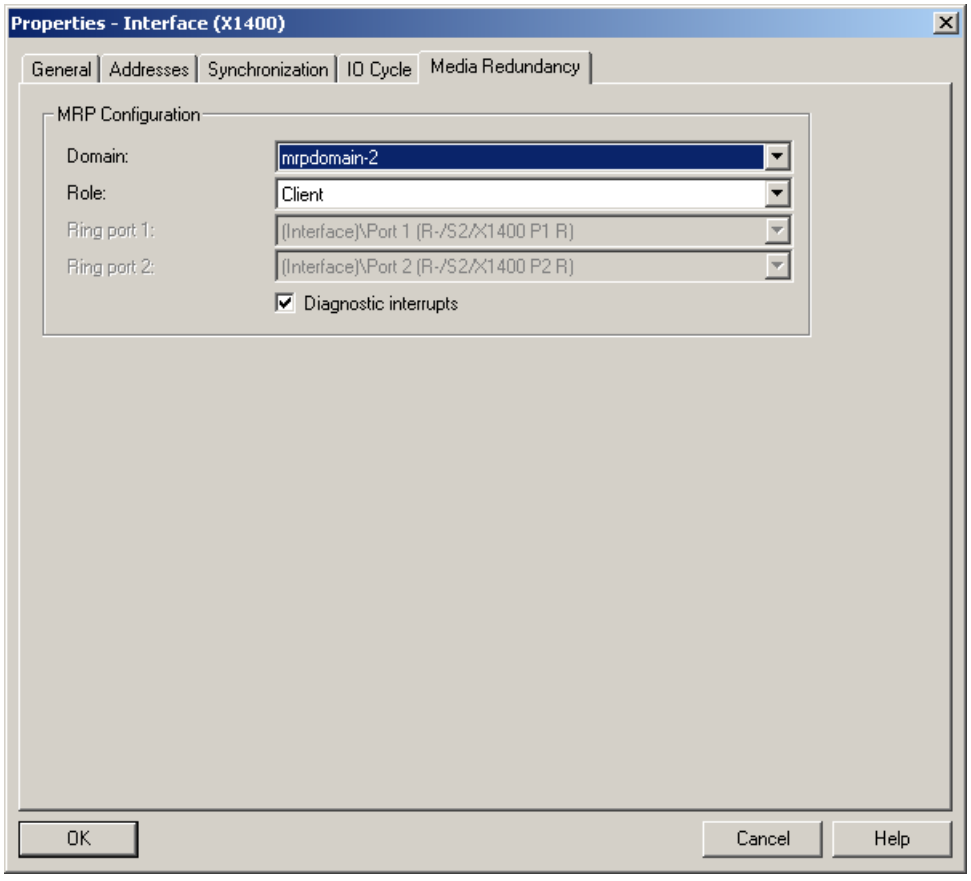


Figure 4-395 Properties of redundancy client

3. Switch to the **Media Redundancy** tab.
4. Select the MRP domain under **Domain**. If you are using only one domain, this is pre-assigned.
5. Activate the **Diagnostic interrupts** option if you want to see ring communication faults in the diagnostics buffer and in the device diagnostics.

Identification and maintenance (I&M) data

Introduction

For the identification and maintenance data (I&M), data structures are defined in PROFINET which can be implemented for all devices. These data structures are used to uniquely identify the IO device. The data set for the unique identification of an IO device, the identification and maintenance data set 0 (I&M0), is supported by all PROFINET IO devices. Further information can be stored in a standardized format as I&M 1-4 if required. As of V4.4, SIMOTION supports all I&M data.

PROFINET I-device I&M data

The I&M data is defined for a device (I-device in this case) and is readable and writable when the I-device is used by a higher-level controller. As of V4.4, SIMOTION supports the data sets 1-4 in addition to the I&M data set 0 (electronic rating plate) as a PROFINET I-device. Reading/writing via PROFINET data set services is supported. Access from the user program via the SIMOTION user interface is not possible.

I&M data sets:

- I&M0: Electronic rating plate
- I&M1: Location designation/Plant ID
- I&M2: Installation date
- I&M3: Description
- I&M4: Signature/additional information

Structure of the I&M data

The following description provides an overview of the I&M data structure. A detailed description can be found in the PROFINET specification.

- 0xAFF0 I&M0 (read) - electronic rating plate
  - Vendor ID (Unsigned16)
  - Order ID (VisibleString[20])
  - Serial Number (VisibleString[16])
  - Hardware Revision (Unsigned16)
  - Software Revision (“Vx.y.z”)
  - Revision Counter (Unsigned16)
  - Profile ID (Unsigned16)
  - Profile Specific Type (Unsigned16)
  - I&M Version (OctetString[2])
  - I&M Supported (Bit Sequence V2)
- 0xAFF1 I&M1 (read/write) - location designation/plant ID
  - Function (submodule’s function or task; “AKZ”; VisibleString[32])
  - Location (submodule’s location; “OKZ”; VisibleString[22])
- 0xAFF2 I&M2 (read/write) - installation date
  - Installation Date (date of installation or commissioning of a device or module; VisibleString[16])
- 0xAFF3 I&M3 (read/write) - description
  - Descriptor (allows storing any individual additional information and annotation; VisibleString[54])
- 0xAFF4 I&M4 (read/write) - signature disabled for PROFIsafe purposes, not changed by ResetToFactory
  - Signature ("security" code, e.g. for Safety; OctetString[54])

Constraints on the I-device I&M data

Please note the following constraints.

- The I&M 1-4 data can only be accessed if the option **Parameter assignment for the PN interface and its ports on the higher-level IO controller** has been activated on the **I-device** tab of the PN interface.
- In the case of a SIMOTION CPU with two PN-IO interfaces, each I-device has its own writable I&M data.

- The second SIMOTION PN-IO interface is a plug-in interface that has its own article number. Both I-devices, therefore, also have different I&M 0 data.
  - Onboard PN interface: SIMOTION CPU article number
  - CBE30-2 PN interface: CBE30-2 article number
- The rack for the I-device I&M data is the first subslot of the transfer areas (subslot 1000). The I&M 1-4 data can only be written via this subslot. The I&M data for the other subslots is also shown when reading.

Reading and writing I&M data

You can display the I&M data in SIMATIC Manager or in HW Config. If you have an online connection, you can also change the data.

Setting I&M data in SIMATIC Manager

The PG/PC is connected online with the SIMOTION I-devices. They are connected to the same physical Ethernet subnet.

1. Select the menu command **PLC > Display accessible nodes**.
2. Select the SIMOTION I-device that is found
3. Select menu command **PLC > Change module identification**. A window containing the I&M data appears.
4. Change the required I&M data. The **Take into account** checkbox must be activated for the change to be transferred to the module.
5. Click **OK** to confirm. The data is changed.

Changing I&M data in HW Config

1. In HW Config, double-click the SIMOTION I-device symbol in the working area.
   The **Properties window** is displayed.
2. You can view the I&M data in the **Identification** tab.
3. If you have an online connection, you can display a comparison of the offline and online data via the menu command **PLC > Load module identification**.
4. In order to change the data, you must activate the **Take into account** checkbox and confirm the new data with **OK**.

4.5.5.4 Configuring direct data exchange (data exchange broadcast) between IO controllers

Introduction

Introduction

I/O data areas can be exchanged cyclically between two or more SIMOTION controllers via IRT High Performance. This is also referred to as controller-controller data exchange broadcast. Controller-controller data exchange broadcast is only possible between SIMOTION controllers via PROFINET IO with IRT High Performance.

For data exchange to take place, the devices must be located in a common sync domain and configured accordingly as sync master and sync slaves.

Note

This function is not available for SIMATIC CPUs.

There are in fact two types of data exchange broadcast. One is automatically created by the system (e.g. distributed synchronous operation), while the other can be applied by the user in his or her application. You can configure this second type of data exchange broadcast.

Note

The user must not use the engineering tools to make changes (e.g. amending the address areas) to the data exchange broadcast which has been automatically configured by the system. Doing so will result in error states.

Note

The controller-controller data exchange broadcast can only be configured if all the SIMOTION controllers involved in slave-to-slave communication are together in one project. If this is not the case, alternatively the SIMOTION I-device can be used for cross-project data exchange broadcast.

Recommendation

We recommend, initially configure the send areas for all PROFINET devices and then the receive areas. Adopting this procedure will enable you to assign the previously defined send areas when defining the receive areas. This prevents invalid inputs.

Data volume

Approx. 3 KB (4 frames of 768 bytes each) can be transferred. 24 bytes are needed for each configured synchronous operation relationship. In other words, if 5 following axes were defined for one master axis, the system needs 5 * 24 bytes. The remaining data is available for application-specific data exchange broadcast. In controller-controller data exchange broadcast, the max. submodule size is limited to 254 bytes.

Note

An FAQ section on the subject of PROFINET configuration is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

This FAQ section deals with the subjects of distributed gearing and controller-controller data exchange broadcast (https://support.industry.siemens.com/cs/ww/en/view/38486079).

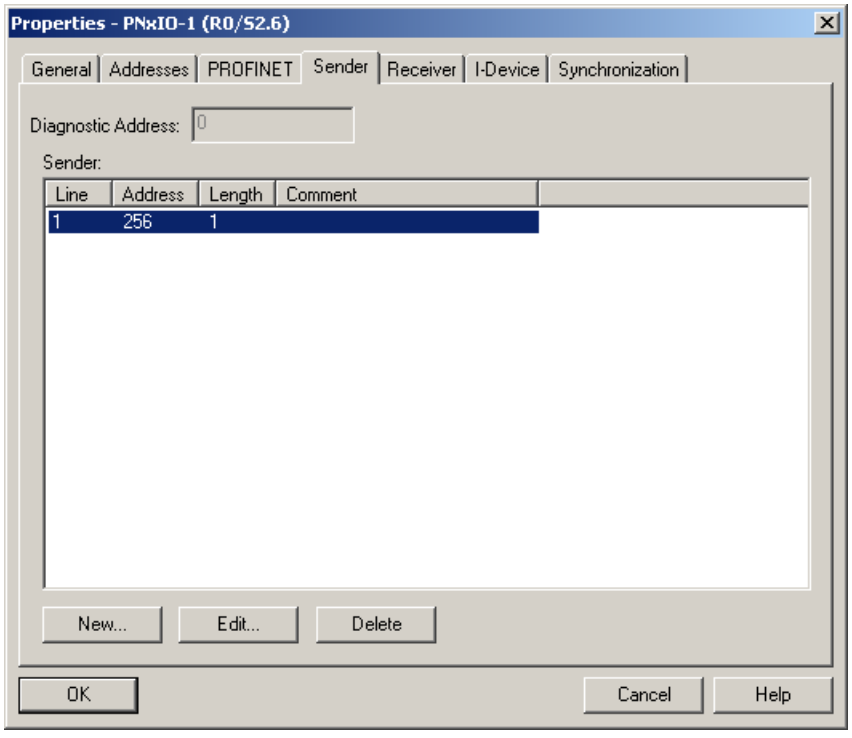
See also

Quantity structures (Page 2162)

Configuring the sender

Procedure

1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **Sender** tab.



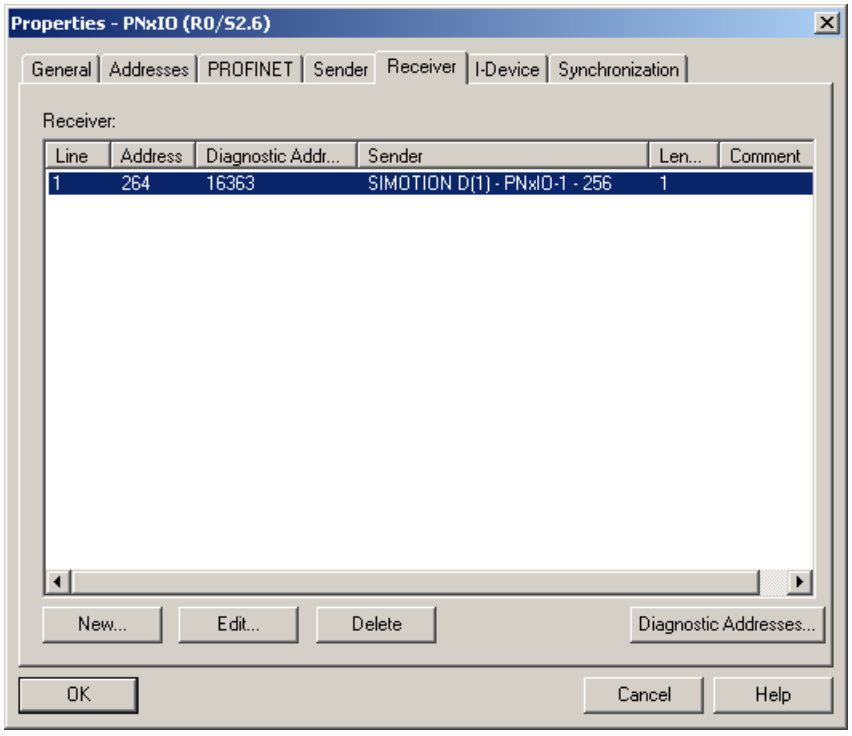
3. Click the **New** button.
4. Enter in the Properties dialog box of the sender, the start address from the I/O area and the length of the address area to be used for sending. Comment the data area so that you will be able to identify the data transmitted via this area later on. The maximum size of a variable is limited to 254 bytes (in the case of controller-controller data exchange broadcast).
5. Confirm the settings by clicking **OK**.
6. Repeat steps 3 to 5 for further send areas.
7. Change the preset diagnostics address for the send areas, if required.
8. Confirm your entries with **OK**.

Note

If a PROFINET interface of the SIMOTION controller is configured as a sender for controller-controller data exchange broadcast, this sender will have precisely one diagnostics address. This also applies if several slots are created for the data exchange broadcast.

Configuring the receiver

Procedure



1. Open the Properties dialog of the PROFINET interface (double-click the corresponding row in the configuration table of HW Config).
2. Select the **Receiver** tab.
3. Click the **New** button.
4. Click the **Assign sender** button in the **Properties receiver** dialog.
5. In the **Assign sender** dialog, select the data area of the desired node which is to be received by the local controller.
6. Confirm your selection with **OK**.
7. Enter in the Properties dialog of the receiver, the start address of the address area to be used for receiving. The length of the address area must not be changed as it is automatically adapted to the length of the send area. The configuration can only be compiled if the send and receive areas have identical lengths!

   Note

   Unlike the sender, each slot is assigned its own diagnostics address in the configuration of the receiver, through which the receiver can detect sender failure, for example. Click the Diagnostics addresses button if you want to edit this address.

8. Repeat steps 3 to 7 for further receive areas.
9. Confirm your entries with **OK**.

4.5.5.5 Configuring the iDevice

PROFINET IO and I device

Introduction

Up to SIMOTION 4.0, direct coupling (of SIMATIC and SIMOTION via PROFINET, for example) was only possible via TCP or UDP, or via additional hardware (PN/PN coupler, SIMATIC CP). With SIMOTION V 4.1.1.6 or higher, the direct coupling of controls - a familiar feature with PROFIBUS - has been introduced for PROFINET IO. It is possible, for example, to connect the SIMOTION as an I slave to the SIMATIC CPU via PROFIBUS. A similar function, referred to as "I-device", is also available for PROFINET IO. This supports data exchange between the controls via I/O areas, with the communication programming required for TCP or UDP being replaced by configuring and system functionality. In addition, the costs associated with the hardware solutions used previously no longer apply (PN/PN coupler, SIMATIC-CP).

The "I-device" (intelligent IO device) functionality of a controller enables exchange of data with an IO controller. Here the I-device is connected as an IO device to a higher-level IO controller. Communication can be established in both directions (bidirectional) via I/O areas using the I-device functionality. Further, this function allows both controls to operate in separate projects.

When operating as an I-device, a SIMOTION device can be used for data exchange with a SIMATIC station, for example. A SIMOTION device acting as an I-device may also be used as a feeder for a modular machine. Please see the *description of functions titled Motion Control basic functions for modular machines*.

Another application for a SIMOTION device acting as an I-device involves providing distributed synchronous operation beyond the limits of a project (see the *Motion Control Technology Objects Synchronous Operation, Cam* Function Manual).

Note

An I-device can only be created using SIMOTION V4.1.1.6 or higher.

Properties of an I-device

As well as performing the role of an IO device on a higher-level IO controller, an I-device can set up its own local PROFINET IO system with built-in local IO devices, thereby acting as an IO controller itself. Both these functions are implemented via the same PROFINET interface on the device.

With SIMOTION, the I-device is available for PROFINET IO with RT and IRT High Performance.

The following constraint applies to the options for combining functions:

Table 4-227 RT and IRT I-device combination options with SIMOTION

SIMOTION function	Possible additional functions			
	RT I-device	RT controller	IRT I-device	IRT controller
RT I-device		X	-	X
RT controller	X	-	X*	X*

SIMOTION function	Possible additional functions			
IRT I-device	-	X	-	-
IRT controller	X	X	-	

*Either an IRT I-device or an IRT controller

As with any other IO device, an I-device's PROFINET interface requires parameter assignment data in order to operate. With IO devices, this data is usually loaded via the associated IO controller in the form of parameter assignment data sets.

Two options are available for I-devices.

An I-device's interface and PROFINET interface ports can either be parameterized by the higher-level IO controller or by the I-device itself on a local level. The preferred option can be selected as part of the I-device's configuration.

- The higher-level IO controller does **not** need be used for parameterizing the PROFINET interface of the I-device.
  This option should be used if the I-device is to be operated as an RT I-device. The same interface can then additionally be operated as an RT controller or an IRT controller.
- The IO controller loads parameter assignment data sets for the PROFINET interface to the I-device.
  This option should be used if the I-device is to be operated as an RT I-device. The same interface can then also only be operated as an RT controller (IRT controller only possible on a second PROFINET interface).

If the I-device is being operated with IRT, the I-device's send clock must be set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system. If the I-device is being operated with RT, the I-device's update time must be either set to match the send clock for the sync domain of the higher-level IO controller's PROFINET IO system, or down-scaled by a multiple of this send clock.

The table below contains details of the send clocks and update times which must be set, along with their possible combinations.

Send clocks/update times of an I-device
Higher-level IO controller and I-device with IRT, no local PROFINET IO system or local PROFINET IO system with IO devices with RT • I-device send clock: – Must be the same as the send clock for the higher-level IO controller. – To be set on the I-device in <PROFINET Interface> properties using the **Send clock** drop-down list box on the **PROFINET** tab
Higher-level IO controller with IRT and I-device with RT, local PROFINET IO system with IRT • I-device update time: – Must be an integral multiple of the send clock for the higher-level IO controller and the send clock for the IO controller on the I-device – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab

Higher-level IO controller and I-device with RT, no local PROFINET IO system • I-device update time: – Any of the possible update times for the I-device may be set. – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab
Higher-level IO controller and I-device with RT, local PROFINET IO system with IRT: • I-device update time: – Must be greater than or equal to the send clock for the IO controller in the I-device. This means that, for example, if an IRT controller with a 2 msec send clock is configured on the CPU of the RT I-device, the following condition applies to the higher-level controller: Update time of the higher-level controller for access to the I-device >= send clock on the CPU of the I-device. This means that update times of 2, 4, 8, ...ms are possible. – To be set on the I-device proxy in <PROFINET Interface> properties, under **Update Time** on the **IO Cycle** tab

The figure below shows how an I-device can be configured on a higher-level IO controller. The higher-level IO controller sets up a PROFINET IO system containing the I-device.. The I-device is able to set up a local PROFINET IO system. Each of these PROFINET IO systems can belong to its own sync domain. However, the I-device must only be assigned to one of the possible sync domains, as a PROFINET interface can only belong to a single sync domain.

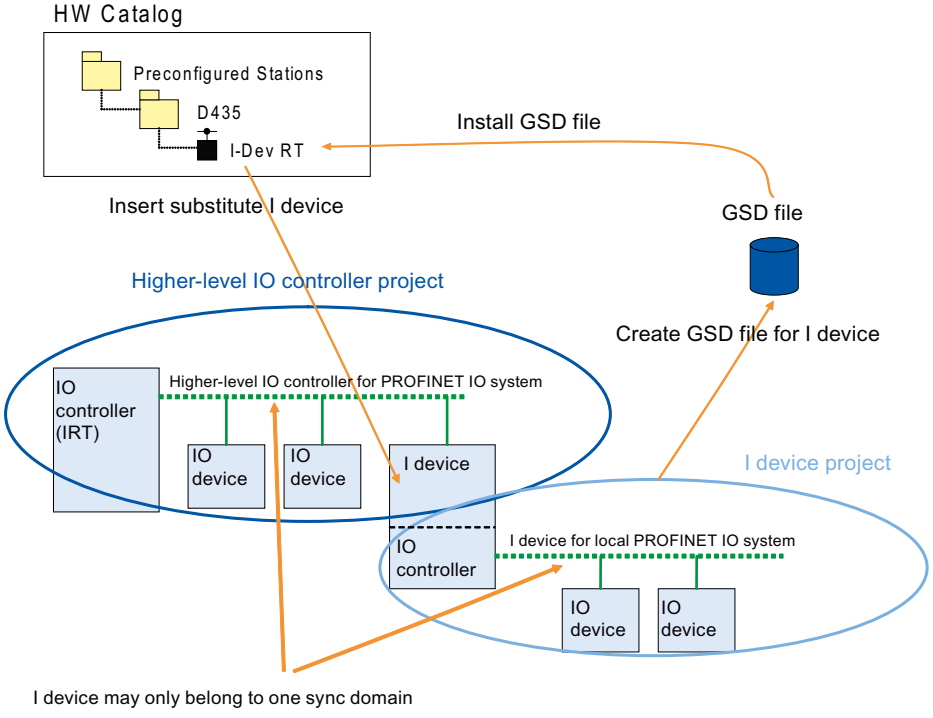


Figure 4-396 I-device configuration

Configuration procedure

- The I-device itself and the IO controller on which it is to be operated should be created in different projects.
- The PROFINET interface's I-device mode must be active for the I-device. In addition, the input and output ranges in the I-device must be configured for data exchange with the higher-level IO controller.
- After an I-device has been created and configured, a GSD file needs to be created and installed for its I-device proxy. The I-device proxy will then be available in the hardware catalog under "Preconfigured Stations."
- The next step involves inserting the I-device proxy from "Preconfigured Stations" in the hardware catalog into the higher-level IO controller's PROFINET IO system.

Since a manual process in the hardware catalog is required to create an I-device proxy, there is no automatic alignment between the project with the I-device and the corresponding I-device proxy in the GSD file. As a result, no subsequent modifications can be made to the I-device configuration. If a modification is made, however, a new GSD file will have to be created and installed. Where numerous modifications have subsequently been made to the configuration of a particular I-device, with multiple GSD files created and installed by the I-device, the version shown under "Preconfigured Stations" in the hardware catalog will always be the most recent one. The version will only be updated, however, if the identifier used for the I-device proxy when creating and installing the GSD file remains the same. Only the input and output addresses for data exchange may be modified in the project for the higher-level IO controller.

Since I-device connected to their higher-level IO controller and IO devices connected on the PROFINET IO system of a single I-device are connected via one and the same PROFINET interface, they will also be located in one and the same Ethernet subnet. This means that the device names and IP addresses of all these devices must be different from each other, and the subnet masks must be identical. It is particularly important to bear this in mind if the higher-level IO controller and the I-device are in different projects, as HW Config cannot check that device names, IP addresses, and subnet masks are consistent across different projects.

Device name (NameOfStation) for the I-device

As with all IO devices on PROFINET IO, a device name also has to be defined for the I-device in the configuration. As the device name (NameOfStation) for the I-device is set in the properties for its PROFINET interface, it is identical to the device name of the IO controller in the I-device. The name set is written to the GSD file for the I-device proxy when this file is created and installed.

When the I-device proxy is inserted into the PROFINET IO system of the higher-level IO controller, the device name previously assigned in the GSD file will be accepted into the configuration. The **device name will not be accepted** if you insert the GSD into the same project on a different IO controller. This is often the case with connections between a SIMOTION controller and a SIMATIC CPU. In this case the device name is automatically manipulated by the engineering system

(devicename"-1") and has to be manually adjusted to the original device name. Message 13:5144 ("There is already...") appears and must be confirmed.

Note

In all cases, it is important to ensure that the **device name in the configuration** of the higher-level **IO controller** is the **same** as the **device name** defined for the **I-device**. As a result, device names must not be amended after the higher-level IO controller has been added to the PROFINET IO system.

If the device names are different, the higher-level IO controller will be unable to identify the relevant I-device and, consequently, will not commence cyclic exchange of input/output data.

Applications of device names for the I-device

- Since a device name must not be used twice within an Ethernet subnet, any device name which already exists will be amended when an I-device proxy is inserted into the PROFINET IO system of its higher-level IO controller. In view of this, it is important to ensure that the device name previously assigned in the GSD file is not used at this stage.
- An I-device proxy can be used multiple times if multiple SIMOTION controllers with the same configuration are to be connected to a higher-level IO controller. You only have to ensure that each SIMOTION controller has a unique device name.

See also

Creating an I device (Page 2245)

Insert the iDevice substitute in the higher-level IO controller (Page 2250)

Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET (Page 2350)

iDevice (Page 2131)

I device functionality with SIMOTION SCOUT V4.2 or higher

Description

With Step 7 5.5 or higher, SIMATIC CPUs can also be configured as I devices. The I device functionality of SIMOTION CPUs and SIMATIC CPUs has been consistently standardized. In the context of this standardization process, the GSD version has been set as V2.25. With SIMOTION projects that use a lower version than V4.2, the iDevice must therefore be re-exported / reinstalled and reintegrated into the project of the higher-level IO controller when upgrading to V4.2. If you edit projects without re-exporting and reinstalling the GSD file, a diagnostics buffer entry will be created on the CPU of the I device when establishing a connection via PROFINET.

Note

If you import, restore or insert a project that contains a module configured via a GSD file onto a computer, and then open with SIMOTION SCOUT, the GSD file used within it is installed automatically, if this is no longer present on the computer. The hardware catalog is not updated automatically and must be manually updated. Execute **Tools > Install catalog** in HW Config to do this.

Note

You can scan the logical address of a proxy iDevice using the function _getLogDiagnosticAddressFromDpStationAddress. The function _getnextlogaddress is not intended for this purpose.

Note

Boundary conditions for I device V4.2 or higher

STEP 7 V5.5 must be used for configuration on the higher-level IO controller, as this is the minimum version required to import the GSD file V2.25 file.

When upgrading older versions of the SIMOTION controller to V4.2, the iDevice becomes "incompatible" and it is absolutely essential to export/import the GSD file.

Below you will find a list of various scenarios which illustrate I device compatibility.

Use case 1: Old project involving SIMOTION devices lower than V4.2 without modification to the I device.

1. Open an old project involving SCOUT V4.2/Step 7 5.5. The project contains RT I devices for communication with a SIMATIC CPU.
2. Change the project, but do not make **any** changes to the I device configuration.
3. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.1.
4. The project can be reloaded without any problems.

Use case 2: Old project involving SIMOTION devices lower than V4.2 with modification to the I device interface.

1. Open an old project involving SCOUT V4.2/Step 7 5.5. The project contains RT I devices for communication with a SIMATIC CPU.
2. Change the project and make changes to the I device configuration. Changes are automatically implemented on the I device interface and an IO slot is added.
3. Create a new GSD from the iDevice.
4. Install the exported iDevice and replace it on the higher-level SIMATIC CPU.
5. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.1.
6. The project can be reloaded without any problems.

Use case 3: Higher-level SIMATIC CPU and SIMOTION I device lower than V4.2 and upgrade to V4.2.

1. Open the project with a SIMOTION CPU and higher-level SIMATIC CPU. The SIMATIC CPU communicates as a PN controller with the I device of the SIMOTION CPU.
2. Upgrade the SIMOTION CPU to SIMOTION V4.2.
3. Export the I device of the SIMOTION CPU.
4. Diagnostics addresses and station numbers have changed compared to what they were in the previous I device. If these are used as absolute values in system calls in the SIMOTION application, the application program will need to be adapted accordingly.
5. Delete the previous I device of the SIMOTION CPU in the project of the higher-level SIMATIC CPU. Make sure that the input and output addresses of the new I device are identical to those of the old I device, so that the previous S7 application can be used.
6. Import the new GSD file into the project of the higher-level SIMATIC CPU.
7. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.2. The project has now been upgraded to V4.2.

Use case 4: Higher-level SIMOTION CPU and several SIMOTION I devices lower than V4.2 and upgrade to V4.2.

1. Open the project with several SIMOTION CPUs as I devices and a higher-level SIMATIC CPU. The higher-level SIMOTION CPU communicates as a PN controller with the I device of the SIMOTION CPU.
2. Upgrade the SIMOTION CPUs to SIMOTION V4.2.
3. Diagnostics addresses and station numbers have changed compared to what they were in the previous I device. If these are used as absolute values in system calls in the SIMOTION application, the application program will need to be adapted accordingly.
4. Export the I device of the SIMOTION CPUs and import the GSD file into the project of the higher-level SIMOTION CPU.
5. Compile the SIMOTION project, including HW Config, and save it as SIMOTION SCOUT V4.2. The project has now been upgraded to V4.2.

Creating an I device

Requirement

You have already created a project and created a station with rack or a SIMOTION controller in HW Config (SIMATIC Manager or SIMOTION SCOUT). You have configured the PROFINET IO system and now want to configure the I-device.

Note

When configuring the I-device, pay attention to the possible settings for the RT class, see PROFINET IO and I device (Page 2238).

Procedure

1. Double-click on the PROFINET interface of the SIMOTION controller that you want to configure as an I-device. The **Properties** dialog box opens.
2. Switch to the **General** tab and, if necessary, change the device name. This must comply with the DNS conventions (Page 2277).
3. Switch to the **I-Device** tab.
4. Select the **I-device mode** check box.

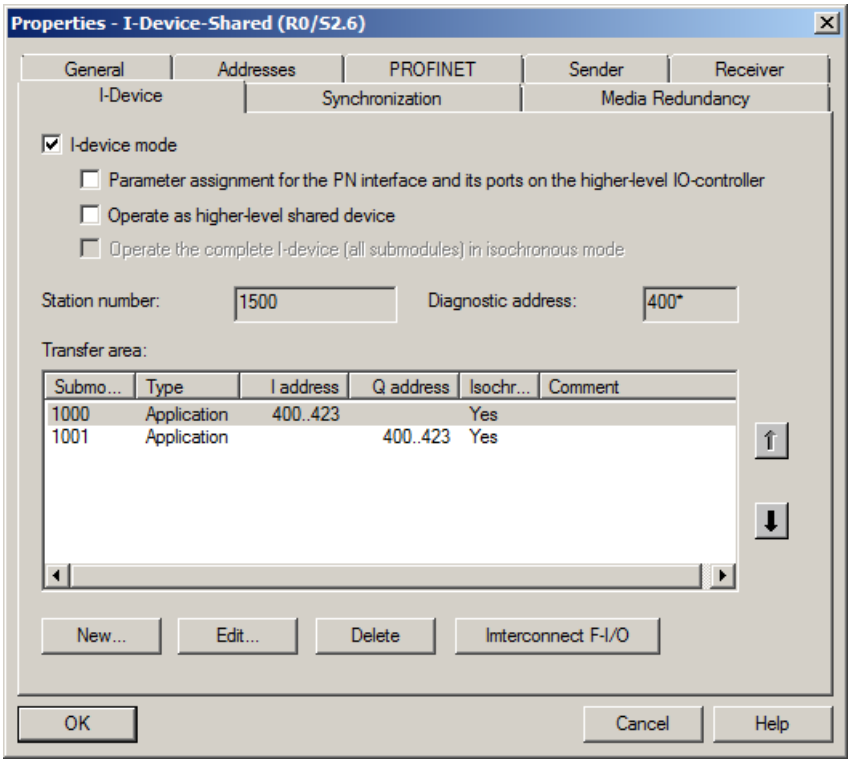


Figure 4-397 I-device properties dialog box

5. You have to activate different check boxes depending on whether I-device communication is to take place on the higher-level controller using RT or IRT.
   - **I-device using RT:**
     Select the **I-device mode** checkbox only.
   - **I-device using IRT**:
     If the I-device is to be operated on the higher-level IO controller isochronously, you also have to activate the **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and **Operate the complete I-device (all submodules) in isochronous mode** checkboxes. This will also result in ports being created in the GSD file and parameter assignment data sets being loaded to the I-device's controller on start-up. If you do not select this check box cyclic communication between the higher-level IO controller and the I-device can only take place via RT. On the I-device proxy, this selection causes the **IO Cycle** tab to appear in the **Properties for the PROFINET interface** dialog box. It will then be possible to select the **Servo** entry under **Assign I-device in isochronous mode** on this tab so that the I-device can be operated isochronously. If an I-device is to be operated with IRT, the PN interface and its ports have to be parameterized on the higher-level IO controller and **Assign I-device isochronously** must be set.
6. You configure the **transfer areas** in the next step. Click **New**... each time to create the virtual subslots (input and output address transfer area), and configure these according to your requirements. By doing this, you are configuring the I/O range for the I-device via which data is exchanged with the higher-level IO controller. You do not need to configure any more settings in the **Sender** and **Receiver** tabs.
   Depending on the CPU used, the following settings are available for the transfer area type:
   - **Application**
   - **I/O**
   - **F-I/O**
     Note:
     You will find more information on the transfer area in the next section.
7. For the **transfer areas**, you can specify whether each individual area is to be operated isochronously, e.g. the fail-safe I/O cannot be operated isochronously. In order to select isochronous mode separately for each transfer area, the check box **Operate the complete I-device (all submodules) in isochronous mode** must be deactivated (from SIMOTION V4.4).
8. Click **OK** to accept these settings and save the project.
9. Continue by creating the I-device proxy (Page 2248).

Creating transfer areas for an I-device

How to create an application transfer area:

1. Select **Application** for an application transfer area. STEP 7 automatically assigns the values of the transfer area on the higher-level IO controller (slot and subslot); the fields cannot be edited.
2. Specify whether the transfer area is to be a local input or output transfer area. To do this, select the corresponding **Address type**. STEP 7 automatically assigns the address type of the higher-level IO controller. If the transfer area is to appear as an output in the higher-level IO controller, it must be an input in the I-device, and vice versa.

3. For isochronous application areas, activate the **Isochronous** checkbox.
4. As with any other submodule, a transfer area requires an address space in order to be addressed by the user program; specify the start address, length, and the process image of the input/output.
5. Enter further information in the comment area as required and click **OK** to close the dialog box.

How to create an F-I/O transfer area (or I/O transfer area):

1. Select **F-I/O** (or **I/O**) for an I/O transfer area. STEP 7 automatically assigns the values of the transfer area on the higher-level IO controller (slot and subslot); the fields cannot be edited.

   Note

   The I/O transfer areas are established automatically for the entire configured fail-safe I/O using the **Interconnect fail-safe I/O** button. You no longer have to set up each fail-safe I/O individually.

2. Now specify which modules/submodules of the I-device are to be made available to the higher-level IO controller as I/O transfer areas. Click the **Select I/O** button: The **I/O transfer area - Select I/O** dialog box opens.
3. Select a module/submodule and click **OK** to close the dialog.
4. As with any other submodule, a transfer area requires an address space in order to be addressed by the user program. You must, therefore, specify the start address of the input/output. The length is determined automatically from the selected module/submodule.
5. Enter further information in the comment area as required and click **OK** to close the dialog box.

See also

Insert the iDevice substitute in the higher-level IO controller (Page 2250)

Exporting the GSD file for the I device

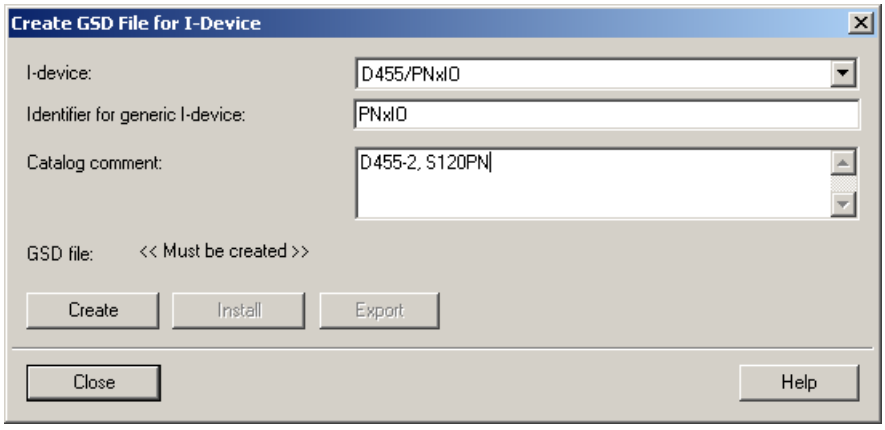
A GSD file must always be exported if an iDevice is to be used in a project on another PG/PC.

Requirement

You have already configured the module to be used as an I device.

1. First save the project.
2. Open HW Config and select **Extras > Create GSD file for iDevice...** from the menu. The **Create GSD file for I device** dialog opens.

3. Select the I device and enter a name for the substitute I device. The substitute I device will appear under this name in **Preconfigured Stations** in the HW catalog.



4. Click **Create** and then **Export**. The **Find folder** dialog opens.
5. Select the path in which the GSD file of the substitute I device is to be stored and click on **OK**.

Creating a substitute I device

There are two different ways of creating a substitute iDevice. The first, **Options > Install GSD files ...** , involves a previously exported GSD file. The second involves the **Create GSD file for I device** dialog.

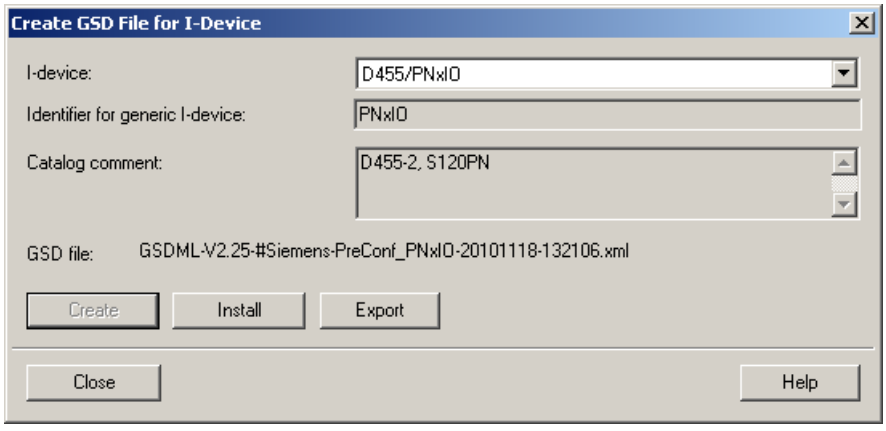
Requirement

You have already configured the module to be used as an I device and exported the GSD file from this.

Procedure 1:

1. First save the project.
2. Create an iDevice as described in Chapter Exporting the GSD file for the I device (Page 2247). There is no need to export the file; instead, it is installed immediately.

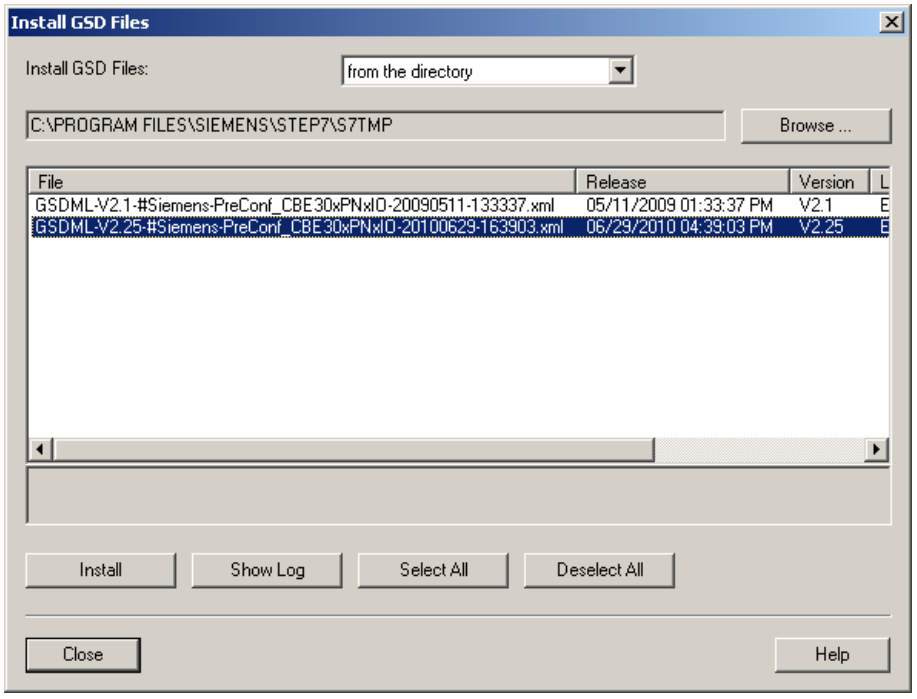
3. Click **Create** and then **Install**.



4. Click **Close**. The substitute I device will now be available under "Preconfigured Stations".

Procedure 2:

1. Select **Extras > Install GSD files....** . The "Install GSD files" dialog box opens.
2. Click **Browse...** . The **Find folder** dialog opens.
3. Select the path in which the GSD files of the substitute iDevice are stored and click on **OK.**



4. Select the required GSD files and click **Install**.
5. Click **Close**. The substitute I devices will now be available under "Preconfigured Stations".

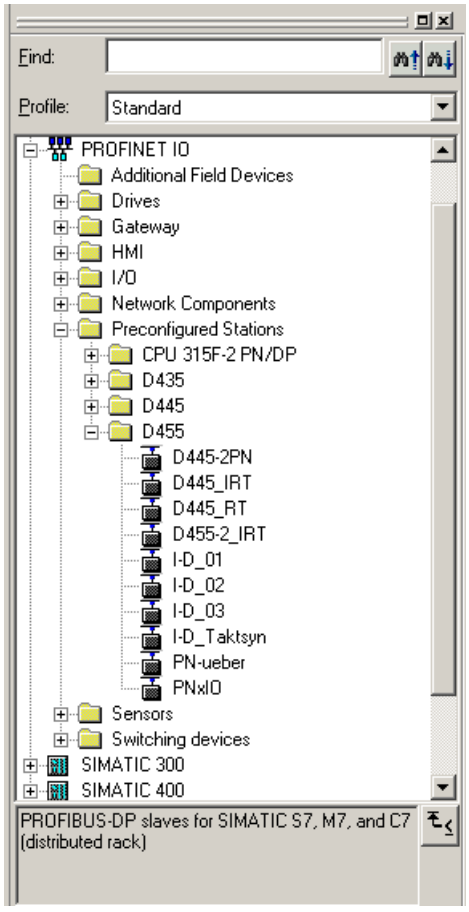


Figure 4-398 iDevice entry in the hardware catalog

Insert the iDevice substitute in the higher-level IO controller

Requirement

You have already created an I-device proxy. A project is open and an IO controller with a PROFINET IO system has already been configured.

Inserting an I-device proxy

1. Open the hardware catalog in HW Config.
2. Drag the relevant I-device proxy from the hardware catalog (**PROFINET IO > Preconfigured stations**) to the PROFINET IO system of the higher-level IO controller. The I-device proxy is displayed as a normal IO device on the PROFINET IO system. Depending on whether the option **Parameter assignment for the PN interface and its ports on the higher-level IO controller** has been selected for the I-device configuration, the PN interface and the individual ports of the I-device are also displayed here.

   Note

   Check whether the device name of the I-device proxy matches that of the PN interface of the corresponding controller. For more information, see the chapter PROFINET IO and I device (Page 2238).

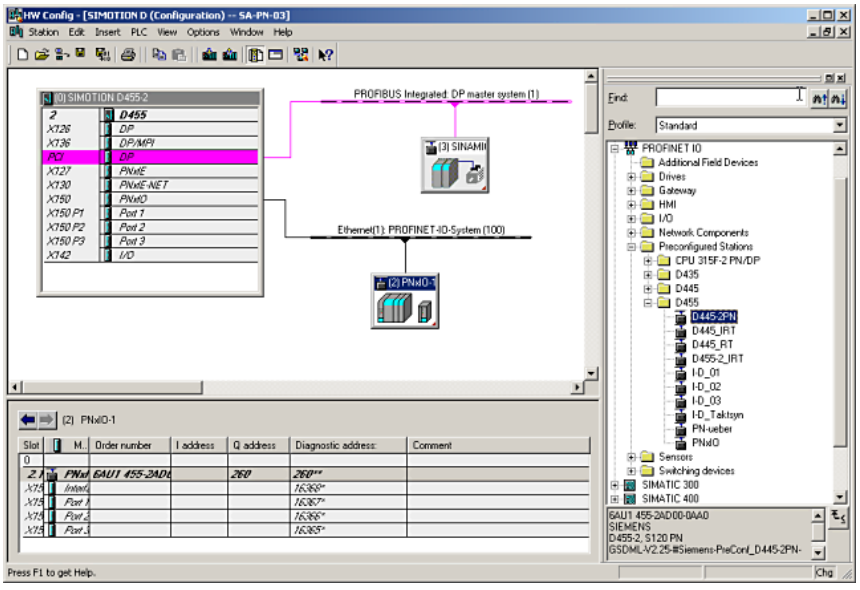


Figure 4-399 I-device on the IO controller

The number of submodules corresponds to the number of the configured submodules of the I-device in the GSD file. Modules and submodules (virtual subslots) cannot be deleted.

3. You can interconnect the I-device ports with the IO controller ports in the Topology Editor.

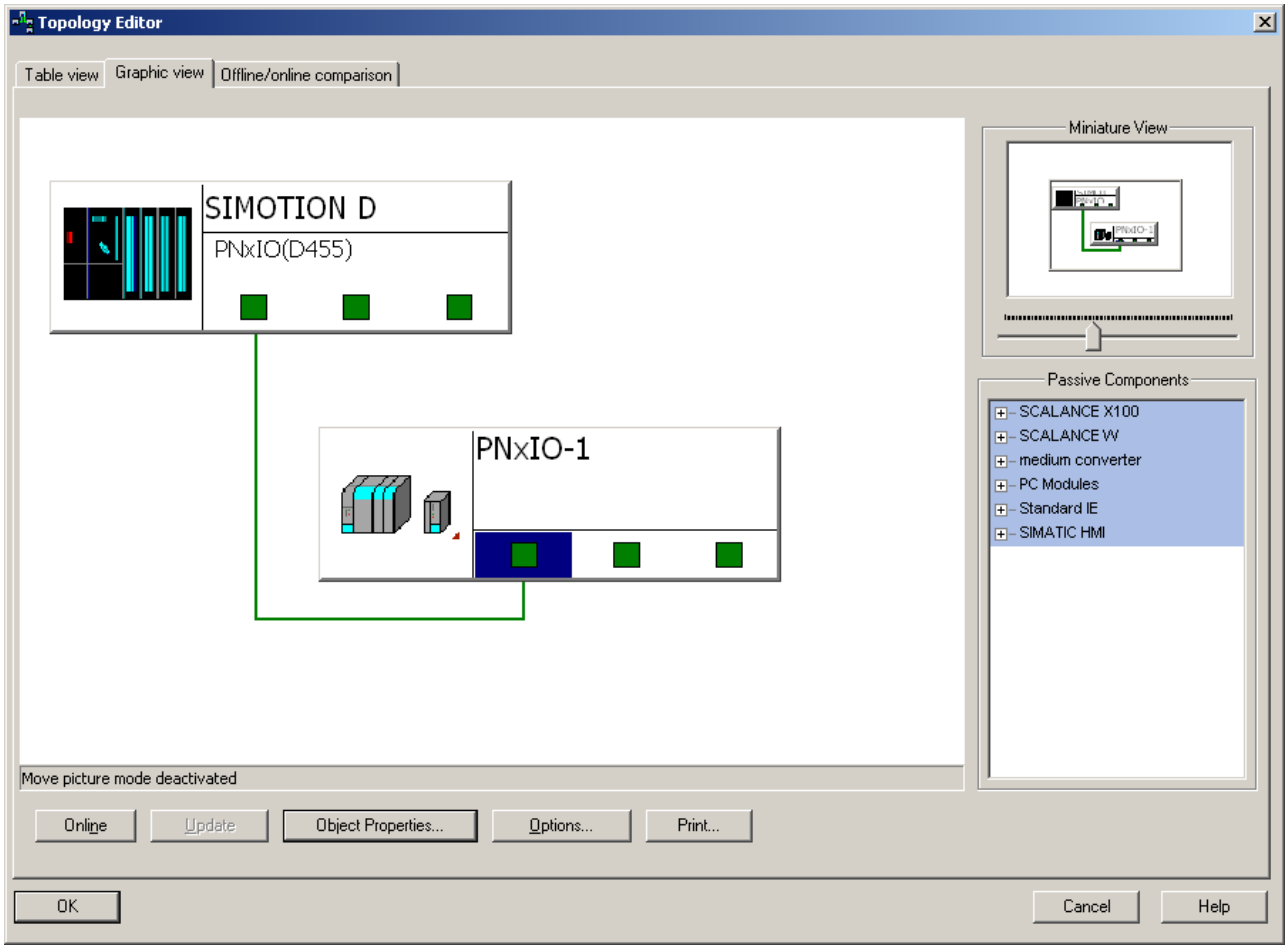


Figure 4-400 Interconnecting I-device ports

Assigning an IP address for the I-device proxy

1. Double-click the I-device to open the **Properties** dialog box. This displays how the IP address is assigned.
2. If the option **Obtain IP address using a different method** is activated for the SIMOTION controller that is configured as an I-device, **no** IP address will be saved in the project. Then the IP address must be assigned from the IO controller. If the option is **not** activated, then the IP address is already saved in the project.

Set synchronization role and isochronous mode for IRT I-device

1. Double-click on the PROFINET interface in the rack of the SIMOTION I-device to open its properties.
2. In the **Synchronization** tab, select **Sync-Slave** and **IRT** as synchronization role and RT class respectively.
3. Select Servo under **Assign IO device in isochronous mode** on the **IO Cycle** tab**.**

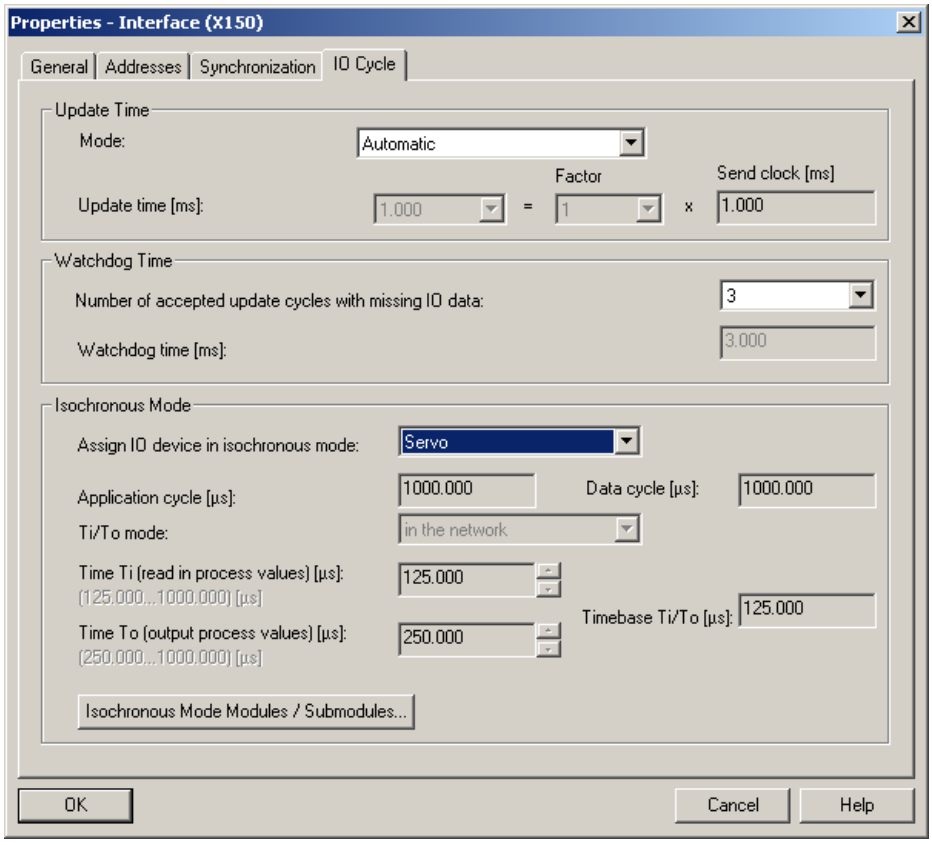


Figure 4-401 Setting isochronous mode on the I-device

Setting the update time and send clock

RT I-device

- The update time is to be set up for the RT I-devices. Double-click the PROFINET IO system and select the **Update time** tab in the **PROFINET subnet properties** dialog. Set the update time there.

IRT I-device

- The send clock needs to be set for IRT I-device. The setting made for the send clock in the I-device's project must be the same as for the send clock in the higher level IO controller's project. Set the send clock for the higher-level IO controller in HW Config using **Edit > PROFINET IO > Domain Management**.

See also

IP address and device name via UP/DCP (Mini-IP-Config) (Page 2277)

Deleting a substitute I device

The GSD files for the substitute I device can be found under "Preconfigured Stations" in the following directory:

<Program Files>\Siemens\Step7\S7DATA\GSD,
e.g. GSDML-V2.25-#Siemens-PreConf_**D455-2_IRT**-20100830-132044.xml.

Here, **D455-2_IRT** is the name of the substitute I device. You can delete the substitute I device by deleting the corresponding XML files. The substitute iDevices displayed under "Preconfigured Stations" are only updated when a new GSD file is created and installed or when the HW catalog is manually updated in HW Config.

Shared iDevice

Introduction

As of V4.4, the iDevice of the PN-IO interfaces can be operated with two higher-level IO controllers as a shared iDevice. The properties and configuration are comparable with a shared device. Configuration is similar to that of a shared device.

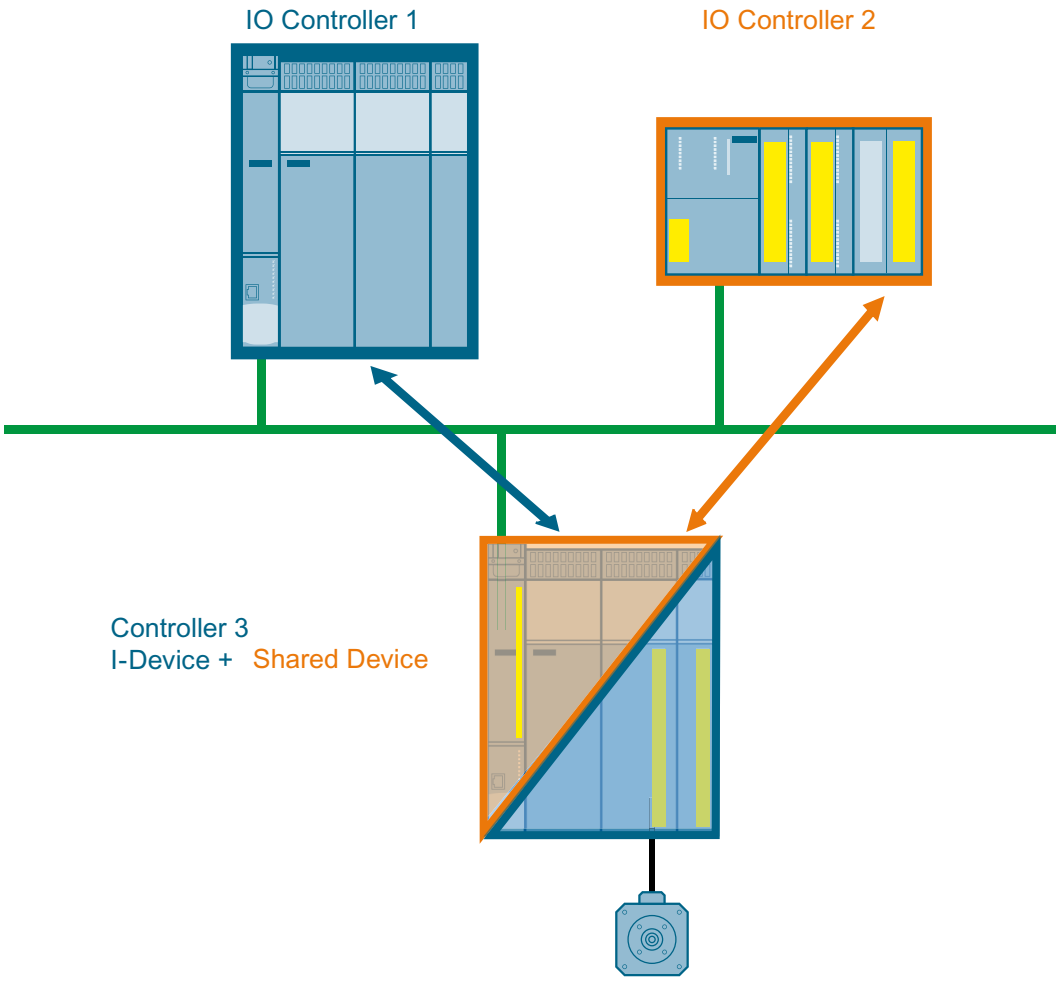


Figure 4-402 iDevice as a shared device

Boundary conditions

When using the shared iDevice, certain boundary conditions must be observed.

- Only one of the two higher-level IO controllers can communicate with the iDevice via PROFINET IRT. The interface module and the port submodule of the iDevice must be assigned to this IO controller.
- The second higher-level IO controller can only communicate with the iDevice via PROFINET RT.
- Each submodule of the iDevice can be assigned to one of the two IO controllers.

- Here a submodule can only be assigned to one of the two higher-level IO controllers.
- In the SIMOTION SCOUT the shared iDevice (shared master) appears only once in the project navigator.
  (The shared master: First device added directly to the network via drag&drop. The second device is created by copying the shared masters and clicking **Shared paste** in the context menu.)

Station failure and station reconnection

A shared iDevice communicates with two higher-level IO controllers which share the submodules of the iDevice. Using the diagnostic events "Partial station reconnection" and "Partial station failure", one can check on the CPU of the iDevice in an application-specific way whether only one or both higher-level IO controllers are connected to the iDevice. However, this is only possible if each submodule of the iDevice has also been assigned to one of the two IO controllers.

The establishment or removal of an application relationship between IO controller and iDevice is reported by opening the PeripheralFaultTask. A distinction can be thus made in an application-specific manner between the different cases using the reported event ID.

An overview of the evaluation with the PeripheralFaultTask can be found in the chapter Alarms on the IO controller (Page 2260).

Shared iDevice send clocks with IRT

If a shared iDevice is operated in connection with PROFINET IRT, then only "even" send clocks (125, 250, 500, 1,000, 2,000, 4,000 µsec) can be used on the higher-level IO controllers for PROFINET RT and IRT.

Background:
As only "even" send clocks are permitted for PROFINET RT and the shared iDevice physically represents a module, only "even" send clocks may also be set on the higher-level IO controller which communicates with the iDevice via PROFINET IRT.

Shared iDevice and PROFIsafe

The shared iDevice function can be used to implement a PROFIsafe configuration with the automation CPU and safety CPU. You can find more information on this subject from the chapter Shared iDevice and PROFIsafe (Page 2374) onward.

4.5.5.6 Loading the communication configuration

Loading the PROFINET IO configuration

Requirement

A PG/PC with which you can go ONLINE is connected.

Procedure

The configuration data must be loaded to all participating controllers once PROFINET IO configuration has been successfully completed.

1. In NetPro, select the Ethernet subnet and then select the **Target system > Loading in current project > Nodes on the subnet** menu command.

4.5.5.7 Communication connections between SIMOTION and SIMATIC

Communication connections overview

Description

The following options are available for establishing a communication connection between SIMOTION and SIMATIC:

- PROFINET RT (or IRT) via iDevice
- PROFINET RT (or IRT) via PN/PN coupler
- PROFINET RT via S7-300 CP as an IO device
- TCP/UDP user communication

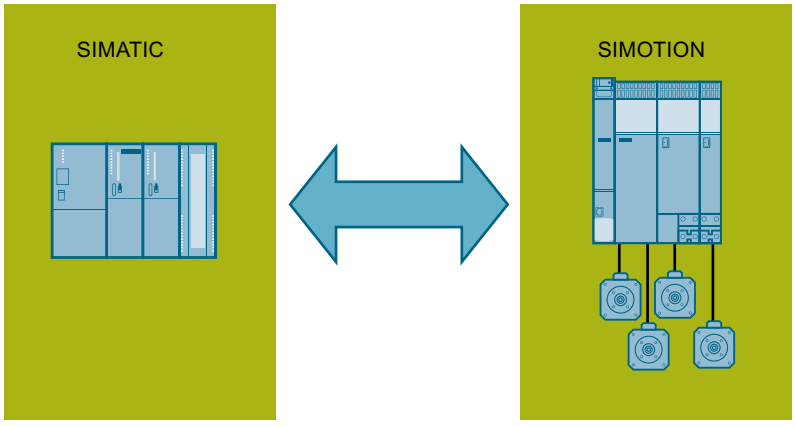


Figure 4-403 Data exchange between SIMOTION and SIMATIC

See also

Data exchange through the use of iDevices (Page 2258)

Data exchange through the use of iDevices

Description

With PROFINET IO/RT, data can be exchanged via an I-device.

Note

A PROFINET device connected to a SIMOTION CPU may have a maximum submodule size of 1,024 bytes.

This limit is especially important if a SIMATIC CPU is configured as an I-device for a SIMOTION CPU. Depending on the hardware and the PN interface, the submodule size may vary from 254 to 1024. Therefore, please note the possible quantity structures (Page 2162).

I-device FAQ

An FAQ section on the subject of coupling a PROFINET RT I-device between a SIMOTION control and a SIMATIC control is provided in SIMOTION Utilities & Applications. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

It looks at the following three application cases:

Case A: Two separate projects: SIMATIC and the SIMOTION as an I-device in a project; SIMOTION as a controller in a second, separate project

Case B: One project for all components

Case C: Multiple use of an I-device

For more detailed information on the configuration of I-devices, please refer to the chapter Configuring the iDevice (Page 2238).

Also note the FAQ for **Configuring RT communication between SIMATIC and SIMOTION (I-device)** I device FAQ (https://support.industry.siemens.com/cs/ww/en/view/38494882)

See also

Communication connections overview (Page 2257)

4.5.5.8 Diagnostic and alarm behavior

Device model for PROFINET

Definition of device model

The device model of PROFINET describes the structure of modular and compact field devices. Similar to a PROFIBUS DP slave, a PROFINET IO device is structured in a modular way. In this way, modules are attached to slots, and submodules to subslots. Channels are located on the modules and submodules, through which process signals can be read and output. In principle it is possible to divide one slot into further subslots, into which the submodules can be plugged.

Diagnostics levels for PROFINET

Diagnostics levels

Each occurring fault in an IO device is transmitted from it to the IO controller. The range and depth of information of a diagnosis can vary depending at which diagnostics level the diagnostics data is evaluated.

In principle there is the option of evaluating diagnostics information at the following levels of address.

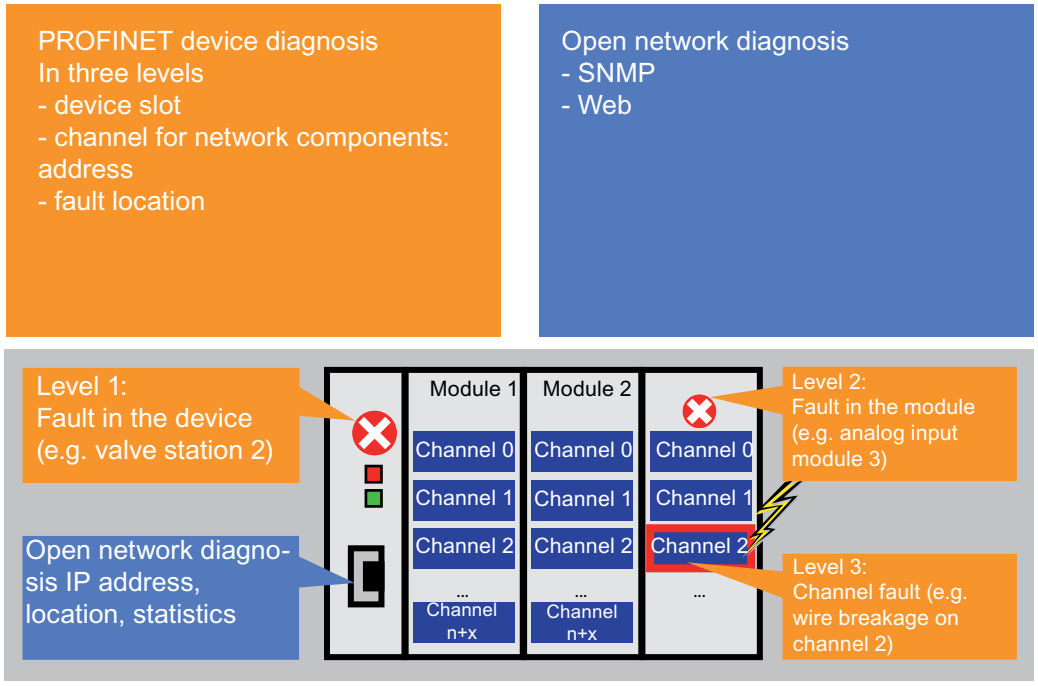


Figure 4-404 Diagnostics overview using the example of ET200

There is a group of standardized diagnostics data sets for PROFINET for each level of address with which the diagnostics information of the IO device can be read (e.g. via the SIMOTION system function _readRecord).

Note

You will find further information on the _readRecord system function in the online help or in the *"SIMOTION System Functions/Variables Devices List Manual"*.

You will find an overview of the data sets available with PROFINET, including the structure of data sets, from section 5 in the *Programming Manual "From PROFIBUS DP to PROFINET IO."* You can find the Programming Manual in the Service&Support portal as a download (https://support.industry.siemens.com/cs/ww/en/view/19289930).

Using TaskStartInfo

TaskStartInfo

In addition to the option of reading the diagnostics information of IO devices through standardized data sets, with SIMOTION the so-called "TaskStartInfo" can also be evaluated for fundamental fault diagnostics.

Important information on the start of each task in the execution system of the SIMOTION controller is saved in the "TaskStartInfo". The respective "TaskStartInfo" can be requested within a task (for diagnostics events that concern peripherals connected to the SIMOTION controller, this request must be made in the "PeripheralFaultTask").

In this process the event that occurs is reported through the system variable "TSI#interruptId" (e.g. CPU stop, process alarm, etc.). The system variable "TSI#eventClass" specifies whether the fault that occurs is appearing or disappearing. On this basis a more precise specification of the event is performed using the system variable "TSI#faultId" (e.g. channel diagnostics pending, station reconnection of an IO device with / without faults, etc.).

Note

Further information:

- Example application SIMOTION: Diagnose CPU stops and module statuses (https://support.industry.siemens.com/cs/ww/en/view/53705461).
- *Function Manual "SIMOTION Runtime Basic functions"* (section "Use Taskstartinfo").

Alarms on the IO controller

Description

A number of alarms are issued on the IO controller. Occurring alarms are listed with the corresponding EventID in the diagnostics buffer of SIMOTION. The following alarms are possible:

- Alarms for direct data exchange between IO controllers
- Station alarms reported by the PROFINET interface

The following table shows PROFINET IO alarms as they are represented in SIMOTION:

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Station failure (_SC_STATION_DISCONNECTED (= 202))	16#39	16#CA	PROFINET IO system error: In this case there is only an incoming event; an outgoing event is represented on 16#38 - 16#CB for each IO device present.
		16#CB	Station failure of an IO device. Shared I-device: No submodule of the I-device is currently subscribed to by the higher-level IO controller.
		16#CC	IO device fault present. Channel diagnostics or manufacturer-specific diagnostics pending. **Note:** 16#39:16#CB is always used for the station failure; 16#38:16#CC is used for incoming with fault.
		16#F8	Partial station failure Shared I-device: Submodules of the I-device are partially subscribed to by the higher-level IO controller.
Station reconnection (_SC_STATION_RECONNECTED (= 203))	16#38	16#CB	An IO device has been reconnected with an error or warning. Shared I-device: All submodules of the I-device are currently subscribed to by the higher-level IO controller.
		16#CC	An IO device has been reconnected with an error or warning.
		16#CD	An IO device has been reconnected, but with an error: Set configuration <> actual configuration.
		16#CE	An IO device has been reconnected, but error during module parameterization.
		16#F8	Partial station reconnection Shared I-device: Submodules of the I-device are partially subscribed to by the higher-level IO controller.

Use of TaskStartInfo

Information about using the TaskStartInfo for the PeripheralFaultTask can be found in the *"SIMOTION Runtime Basic Functions"* Function Manual.

Alarms from the IO device to the IO controller

Description

The alarms are transferred using the PROFINET alarm mechanism from the IO device to its associated IO controller. The alarms are entered in the diagnostic buffer and can be evaluated using the PeripheralFaultTask. The following table shows how alarms are represented as PeripheralFaultTask.

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Diagnosis (incoming)	16#39	16#42	Incoming diagnostic interrupt
Diagnosis disappears (outgoing) Multicast Communication Mismatch Port Data Change Notification Sync Data Changed Notification Isochronous Mode Problem Notification Network component problem notification (_SC_DIAGNOSTIC_INTERRUPT (=201))	16#38	16#42	Outgoing diagnostic interrupt
Process interrupt (_SC_PROCESS_INTERRUPT (= 200)	16#11	16#41	Process interrupt
Pull Alarm	16#39	16#51	PROFINET IO module has been removed or cannot be addressed.
		16#54	PROFINET IO submodule has been removed or cannot be addressed.
Plug Alarm Plug Wrong Submodule Alarm Return of Submodule Alarm (_SC_PULL_PLUG_INTERRUPT (=216))	16#38	16#54	PROFINET IO module or submodule has been inserted, module type OK (actual configuration = set configuration)
		16#55	PROFINET IO module or submodule has been inserted, but wrong module type (actual configuration <> preset configuration)
		16#56	PROFINET IO module or submodule has been inserted, but error during module parameterization
		16#58	IO status of a module has changed from BAD to GOOD
Status			Not Supported
Update			Not supported
Time data changed notification			Not supported
Upload and storage notification			Not supported
Pull module			Not supported
Manufacturer-specific			Not supported
Profile-specific			Not Supported

Alarm types indicated as "not supported" are acknowledged by the SIMOTION controller with "not supported" and not entered in the diagnostic buffer.

Use of TaskStartInfo

Information about using the TaskStartInfo for the PeripheralFaultTask is contained in the *SIMOTION Basic Functions* manual.

Transfer diagnostic data

The exact reason for the alarm is provided as diagnostic data. The _readDiagnosticData() function can be used to read out this data. The length is restricted to 240 bytes.

With V4.2 or higher, you can read out station/module diagnostic data of up to 65,535 bytes with the function block _readVariableDiagnosticData() and the user program.

Alarms for direct data exchange between IO controllers

Description

For PROFINET IO with IRT, communication monitoring takes place between IO controllers. If this establishes that IRT data is no longer being received (either there is no data arriving, or it is arriving too late) a station failure alarm is generated. If communication is re-established, a station reconnection alarm is generated. If IRT data arrives late on three occasions, a station failure alarm is reported.

Note

During communication monitoring, only the receiver slot is monitored.

The following table shows PROFINET IO alarms between IO controllers involved in direct data exchange as they are represented in SIMOTION:

Alarm (TSI#InterruptId)	TSI#eventClass	TSI#faultId	Meaning
Station failure (_SC_STATION_DISCONNECTED (= 202))	16#39	16#F3	The receiver in the direct data exchange is no longer receiving data.
Station reconnection (_SC_STATION_RECONNECTED (= 203)	16#38	16#F0	The transmitter in the direct data exchange has started up and is able to transmit.
		16#F1	The receiver in the direct data exchange has started up without errors and the receiver is receiving data again (all receiving areas are available).
		16#F2	The receiver in the direct data exchange has started up with errors and the receiver is receiving data again (at least one receiving area not available).

SINAMICS drives alarms

Description

In SINAMICS firmware Version 4.5 and higher, SINAMICS drives (S and G) have a PROFIBUS/ PROFINET diagnostics channel. Alarms can be forwarded to a higher-level control via this channel.

The drive is controlled via a TO

If the drive is controlled via a TO, drive alarms are issued via the alarm mechanisms of the technology object.

The drive is controlled directly by means of a user program

If you are controlling the drive directly by means of a user program, you must program the alarm response.

See also

Alarms on the IO controller (Page 2260)

System functions for the diagnostics for PROFINET or PROFIBUS

Overview of system and diagnostics functions

The following table provides an overview of the various system and diagnostics functions for PROFINET IO. Differences with PROFIBUS DP are also indicated.

You will find detailed information about each of the functions in the reference list for functions *List Manual, SIMOTION system functions/variables for devices*.

Function	Note	PROFIBUS	PROFINET
_getStateOfSingleDpSlave()	The function determines the status of communication with a cyclic communications partner (device - PROFINET, slave - PROFIBUS, transmitter or receiver controller-controller data exchange broadcast - PROFINET).	Logical diagnostic address of the DP slave	Logical diagnostic address of the IO device
_getStateOfDpSlave()	_getStateOfDpSlave provides information concerning whether the PROFIBUS DP slave or the PROFINET IO device is activated or deactivated.	Logical diagnostic address DP slave	Logical diagnostic address of the IO device

Function	Note	PROFIBUS	PROFINET
_getStateOfAllDPStations()	The system function determines the status of the communication with cyclic communications partners (device - PROFINET, slave - PROFIBUS, transmitter or receiver controller-controller data exchange broadcast - PROFINET). In addition to the activation status, information on availability is also provided.	Logical diagnostics address of the interfaces	Logical diagnostics address of the interfaces 4.2 and higher With PROFINET IO devices, this system function forms a group signal from the modules present in the device. This means that the function can also tell the user program when modules have been removed or are faulty by using the device feedback values for this purpose.
_getStateOfIO()	This function provides the user program with information on the status of the DP stations, modules, and submodules. It also provides this information in detail for modules and submodules. The following are supported: • Interfaces such as PROFIBUS DP or PROFINET • Stations (slaves, devices), modules, or submodules This function provides a logical diagnostics address or logical I/O address to identified modules, and a list of assigned submodules.		
_getNextLogAddress()	All configured logical addresses of a segment can be determined using this function.	Logical diagnostics address	Logical diagnostics address
_readDiagnosticData() _getStateOfDiagnosticDataCommand()	This function is used to output diagnostic data for a DP slave via the user program. For PROFIBUS, the diagnostics for the slave are read out, i.e. the slave supplies the complete diagnostic information. The structure of the diagnostic data is described in IEC 61158-6-3. For PROFINET, the subslot-specific diagnostics are read (i.e. the data set 0x800A). The diagnostics for a subslot are supplied. The structure of the diagnostic data is described in IEC 61158-6-10.	Logical diagnostic address DP slave	Logical diagnostics address of the IO device, of the module/slot, or IO address of the channel/subslot

Function	Note	PROFIBUS	PROFINET
_readVariableDiagnosticData()	This function block is used to read out diagnostic data of a station or module via the user program. The diagnostics format for PROFINET IO V2.2 and PROFIBUS DP is described in the IEC 61158-6 standard.	-	Data records with a total length up to 4 KB can be read with the FB _readVariableRecord. This function can be used for PROFIBUS and PROFINET. Data sets up to 4 KB are returned from the device only with PROFINET.
_readDriveFaults() SINAMICS only (ET200FC, S120, etc.)	This function is used to read the current fault buffer entry in the drive.	Logical start address of drive (slot).	Each valid logical I/O address of the subslot concerned or diagnostics address of the PAP (for modules without cyclic data). **Note:** In the case of PROFINET, parameter access takes place via the MAP (Module Access Point) of a DO. The MAP submodule is the substitute DO and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. MAP is displayed in the GSD file.

PROFINET device diagnosis in STEP 7

Device diagnosis in STEP 7

In SCOUT, HW Config can be used to perform an online device diagnosis via PROFINET. The diagnosis supplies not only the slot and the channel number, but also the error type. The diagnosis operates similar to that for PROFIBUS.

Procedure

1. Go online and open HW Config for the appropriate SIMOTION device.
2. Select **Target system > Diagnose, monitor/control Ethernet node**.
   HW Config searches for all network nodes. The **(Diagnosis) ONLINE** window opens and displays the network nodes.
3. Right-click the required node and select **Properties**. The detailed diagnosis is displayed. The associated fault is displayed here.

PROFINET IO and DS0 diagnostic interrupts

Diagnostic interrupt PROFINET IO maintenance concept

Description

A device or module of an automation system can essentially be in one of two states: either 'good' or 'failure.' With a view to increasing the availability of sensors/actuators, devices, and modules, these same components also supply information concerning necessary maintenance work in addition to these two states. This additional information includes details about the maintenance state, state of wear, and remaining life time, etc. This constitutes what is known as an 'extended maintenance concept'. The aim of the extended maintenance concept is to detect and eliminate potential faults early on - before they lead to production outages. For this purpose, the 'good' and 'failure' states are supplemented by states called 'maintenance required' and 'maintenance demanded'.

The above states can be depicted on the following maintenance state model.

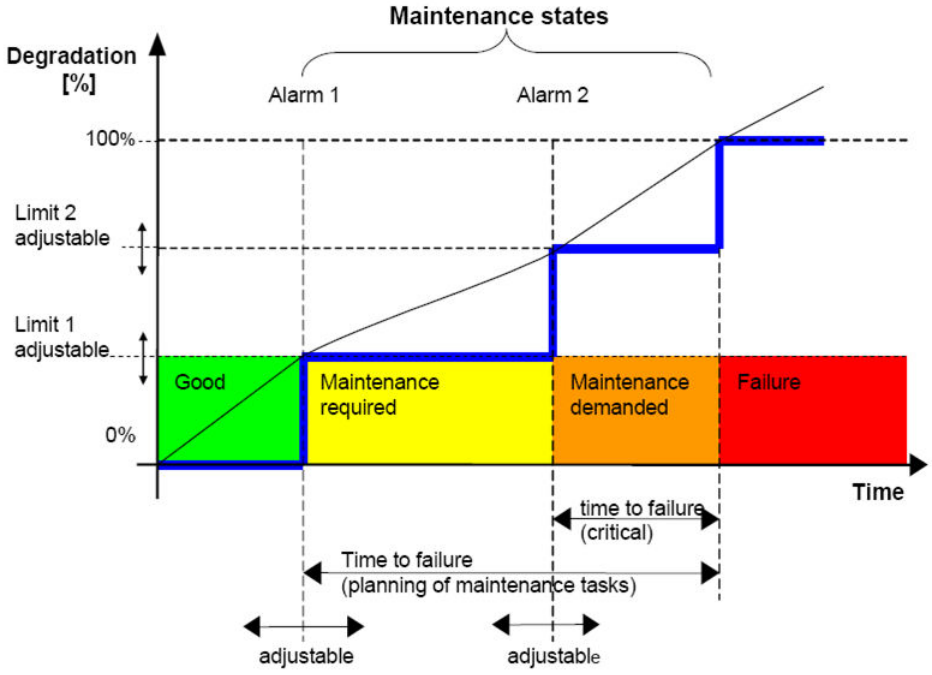


Figure 4-405 PROFINET IO maintenance state model

Maintenance state model

- Good (green): no action required
- Maintenance required (yellow): maintenance is to be scheduled
- Maintenance demanded (orange): maintenance is to be carried out
- Failure (red): fault

Device model for IO device

Description

The PROFINET IO device model stipulates that an IO device is to be divided into slots and subslots. A slot represents a module (= physical assembly) and a subslot represents a submodule (= physical subassembly).

Every submodule can be used for the following objects:

- Cyclic data (I/O interface useful data for process control)
- Alarms (e.g. diagnostic interrupt)
- Data sets
- Parameters
- Diagnostics

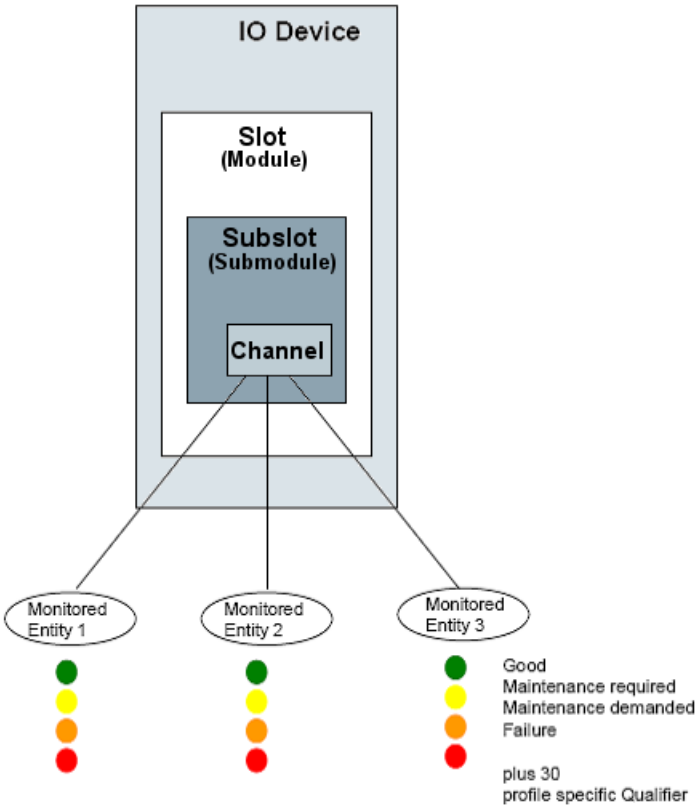


Figure 4-406 Classification of channel, diagnostic states and group information

In the PROFINET IO device model, channels are defined below submodules for the diagnostics. A channel is a logical substructure of a submodule. A submodule can contain up to 65,536 channels. Defined diagnostic functions (e.g. short-circuit, wire breakage, overtemperature) are monitored within the channels. A channel can also monitor several diagnostic functions in parallel.

This monitoring results in channel diagnostics. Several different sets of channel diagnostics can occur in a submodule at the same time. Channel diagnostics from different submodules can, of course, also occur. Channel diagnostics are signaled via a diagnostic interrupt. The diagnostic interrupt message is issued for each submodule in which channel diagnostics occur.

PROFINET IO and DS0 diagnostic interrupts

Description

If a diagnostics-related event occurs (e.g. failure or maintenance required) in an IO device, a diagnostic interrupt is generated and sent to the associated IO controller. The SIMOTION device acting as the IO controller then issues a diagnostic interrupt via the PeripheralFaultTask using TSI#interruptId = _SC_DIAGNOSTIC_INTERRUPT (= 201).

Data set 0 (DS0) for the diagnostic interrupt can be found under the TSI#InterruptId. Data set 0 (DS0) supplies the group states for the channels of a submodule.

Note

The DS0 contains only one selection of the possible diagnostics elements and is filled with all PROFINET-compliant I/O devices.

A diagnostic interrupt can occur as an incoming diagnostic interrupt with TSI#eventClass = 0x38, TSI#faultId = 0x42 and as an outgoing diagnostic interrupt with TSI#eventClass = 0x39, TSI#faultId = 0x42.

A diagnostic interrupt contains the following group states in data set 0 (DS0) in the form of the group bits below. The information relates to the sum total of all diagnostic functions of the channels within a submodule:

- Group failure: DS0.Byte0.Bit 0
- Group maintenance requirement: DS0.Byte1.Bit 7
- Group maintenance demand: DS0.Byte2.Bit7

The group bits are formed by the logical ORing of the respective individual bits of all diagnostic functions of the channels for a submodule. The three group bits are independent of each other and do not influence each other. The group bits are therefore set and/or reset independently of each other.

If just one modification is made to the maintenance states, individual maintenance requirements or maintenance demands are still pending in the submodule, and there are no faults, then an incoming diagnostic alarm is signaled and the following group bits are set accordingly in DS0: group maintenance requirement/group maintenance demand. If all maintenance states have disappeared and there is no failure, an outgoing diagnostic interrupt is signaled and the two group bits (group maintenance requirement/group maintenance demand) are set to 0. Normally, this means that the rule which applies is that an incoming diagnostic interrupt is signaled as soon as at least one of the group bits (group failure or group maintenance requirement or group maintenance demand) is set to 1. However, when all of the group bits (group failure or group maintenance requirement or group maintenance demand) are no longer set to 1, an outgoing diagnostic interrupt is signaled.

Explanation of the DS0 data set bits

Table 4-228 Table: DS0 byte 0

Bit	Description	Comment
Bit 0	Module Fault/OK	Group failure (diagnostics) for a submodule: • 0: No group failure (diagnostics) pending • 1: Group failure (diagnostics) pending
Bit 1	Internal error	Always 0
Bit 2	External error existent	Has the same content as bit 0
Bit 3	Channel error existent	Has the same content as bit 0
Bit 4	External auxiliary power missing	Always 0
Bit 5	Front connector missing	Always 0
Bit 6	Module not parameterized	Always 0
Bit 7	Wrong parameters in module	Always 0

Table 4-229 Table: DS0 byte 1

Bit	Description	Comment
Bit 0-3	Type class of module	3: Type class 3 is to be understood as a distributed I/O and as such also includes PROFINET IO.
Bit 4	Channel information existent	0: No readable channel information present 1: Readable channel information is present in the interrupt data of the diagnostic interrupt. The interrupt data can be read out using the _readDiagnosticData system function.
Bit 5	User information existent	0: No channel diagnostics or manufacturer-specific diagnostics available 1: At least one set of channel diagnostics and/or manufacturer-specific diagnostics is available
Bit 6	Diagnosis alarm from proxy	Always 0
Bit 7	Maintenance Required	Group maintenance requirement for a submodule: • 0: No group maintenance requirement pending • 1: Group maintenance requirement pending

Table 4-230 Table: DS0 byte 2

Bit	Description	Comment
Bit 0	-	Always 0
Bit 1	-	Always 0
Bit 2	-	Always 0
Bit 3	-	Always 0
Bit 4	-	Always 0

Bit	Description	Comment
Bit 5	-	Always 0
Bit 6	-	Always 0
Bit 7	Maintenance Demanded	Group maintenance demand for a submodule: • 0: No group maintenance demand pending • 1: Group maintenance demand pending

Table 4-231 Table: DS0 byte 3

Bit	Description	Comment
Bit 0	-	Always 0
Bit 1	-	Always 0
Bit 2	-	Always 0
Bit 3	-	Always 0
Bit 4	-	Always 0
Bit 5	-	Always 0
Bit 6	-	Always 0
Bit 7	-	Always 0

If detailed information on the channels of a submodule is required in the diagnostic interrupt, the interrupt data for the diagnostic interrupt must be read out and evaluated accordingly using the _readDiagnosticData system function.

Additional information

For more information, please refer to the FAQs in "SIMOTION Diagnostics Device Failure" and the SIMATIC "From PROFIBUS DP to PROFINET IO" Programming Manual.

4.5.5.9 PROFIenergy

Overview of PROFIenergy

Saving energy with PROFIenergy

PROFIenergy is a data interface based on PROFINET. It allows loads to be shut down during idle time in a controlled, centralized manner, and irrespective of the manufacturer and device. This means that the process is given only the energy it actually requires. As well as enabling non-operational states to be introduced, the interface also makes it possible to request consumption values and PROFIenergy statuses from the device. The majority of the energy is saved by the process itself; the PROFINET device only makes a small contribution to the saving potential.

Basic principles and functionality

The PROFINET devices are shut down using special commands in the user program of the PROFINET IO controller. No additional hardware is required; the PROFIenergy commands are interpreted directly by the PROFINET devices.

At the start and end of pauses, the plant operator activates or deactivates the pause function of the plant, after which the IO controller sends the PROFIenergy "START_Pause"/"END_Pause" command to the PROFINET devices. The device then interprets the content of the PROFIenergy command and switches to PROFIenergy mode PAUSE or back to READY TO RUN mode. Further PROFIenergy functions can be used to fetch device information.

PROFIenergy profile

Further information and the basic principles of the PROFIenergy profile can be found in the *Common Application Profile PROFIenergy; Technical Specification for PROFINET; Version 1.1; January 2012; Order No: 3.802*. This profile lists all possible measured values in table form, among other things.

Example of PROFIenergy with SIMOTION

PROFIenergy cell concept

As far as PROFIenergy is concerned, production plants are generally made up of independent cells. Each cell has a "head" controller, e.g. SIMOTION, which has a communication relationship with the higher-level controller. The higher-level energy management system (e.g. SIMATIC controller) communicates with the cell with the help of the PROFINET I-device functionality. In the example, cell 1 contains a SIMOTION controller which serves as a PROFIenergy controller for the lower-level devices (ET200, G120, and S120). At this point, SIMOTION is a PROFINET device for the higher-level, centralized energy management, but also a controller and PROFIenergy controller for its lower-level PN devices.

There may be several cells within a production plant. The higher-level controller records the state of the lower-level cells and shuts them down during idle time in a coordinated manner. A production plant can also consist of n cells.

In this example, the higher-level controller sends the PROFIenergy commands via the I-device interface to the SIMOTION controller. The SIMOTION controller evaluates these commands in the user program, responds with an appropriate PROFIenergy data set, and activates the relevant sensors and actuators. Please note that the interpretation and answering of the PE data sets must take place in the SIMOTION user program. A PROFIenergy library is available for SIMOTION for this purpose. This is described in more detail in the following chapters.

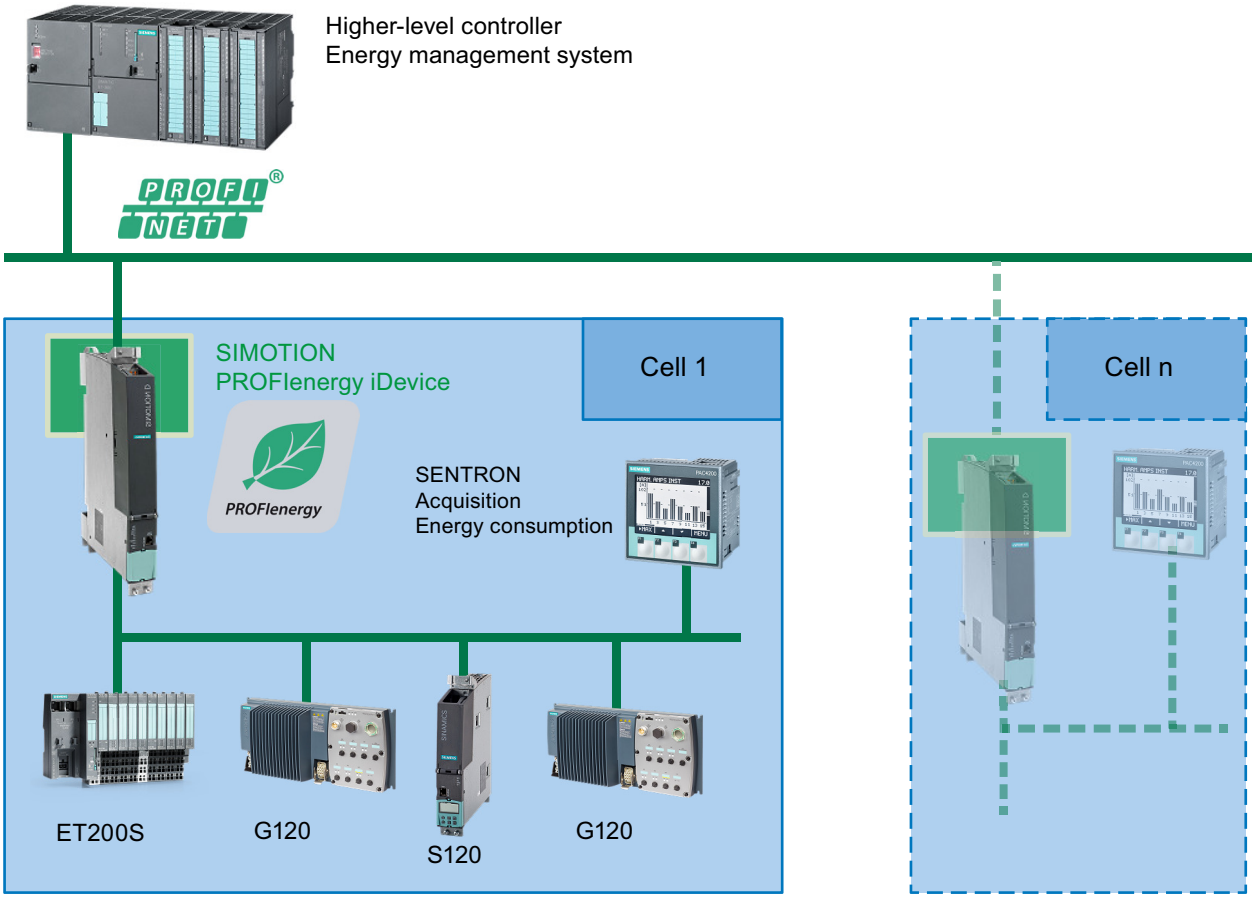


Figure 4-407 Schematic representation of the PROFIenergy cell concept

SIMOTION as PROFIenergy controller and PROFIenergy device

Basic principles of SIMOTION as PROFIenergy controller/device

SIMOTION controllers can control PROFIenergy states and read measured values when used with PROFIenergy-compatible devices (PROFINET devices, e.g. ET200S, S120, G120). SIMOTION controllers can also be operated via the I-Device interface as PROFIenergy devices, i.e. the SIMOTION controller can receive PROFIenergy data sets from the higher-level energy management system and respond to them (e.g. by supplying PROFIenergy measured values).

PROFIenergy telegrams are non-cyclic services and are further processed in data sets. SIMOTION uses the I-Device interface to provide the necessary non-cyclic services for the use of the PROFIenergy telegrams. In addition to the cyclic data, the higher-level PROFINET controller (e.g. S7 CPU) can also send non-cyclic data to SIMOTION in the form of data sets via the I-Device interface. The data set content from the non-cyclic I-device communication is made available for the SIMOTION applications. This data set content can be read by a user program as a receive buffer or written as a send buffer for the response. The processing and evaluation must take place in the user program.

Detailed description

As of SIMOTION V4.4, data sets can be received and sent via the I-device interface, which means that PROFIenergy data sets (commands) can be too. Two new SIMOTION system function blocks _receiveRecord() and _provideRecord() have been made available for this purpose. Data sets can be received by and sent to the higher-level IO controller by calling these system function blocks in the user program.

The interfaces, functionality, and handling of the new SIMOTION system functions are based on the existing SIMOTION system functions for reading/writing data sets, e.g. _readVariableRecord(), and on the corresponding system function blocks for the SIMATIC SFB73 and SFB74.

The system function blocks _receiveRecord() and _provideRecord() are used for non-cyclic communication between two IO controllers, whereby one of the two is functioning as an I-device. This means that non-cyclic data sets can be sent and received from the user program.

A short explanation of the function blocks can be found in the next chapter.

PROFIenergy communication - enabling data sets in the I-Device (as of V4.5)

PROFIenergy communication needs to be explicitly activated in SIMOTION SCOUT TIA for the SIMOTION I-Device in order to enable the PROFIenergy data sets. PROFIenergy data sets are enabled as standard for all "normal" data subslots as well as for F-Proxy subslots.

PROFIenergy communication is activated as standard in SIMOTION SCOUT and cannot be disabled. As a result, PROFIenergy data sets can always be used in SIMOTION SCOUT, but only in SIMOTION SCOUT TIA if the "Activate PROFIenergy communication" box is checked.

The following functions are active when PROFIenergy communication is activated:

- Reading and writing for PROFIenergy data sets via the application is enabled.
- The keywords for PROFIenergy are exported to the I-Device GSD.

Note

SIMOTION SCOUT TIA

The box can always be checked even if your application does not support PROFIenergy for the I-Device. For an I-Device that is to be certified, the box must be unchecked for all subslots of the I-Device for which a PROFIenergy application does not exist.

Activating PROFIenergy communication in the SIMOTION I-Device SIMOTION SCOUT TIA

The hardware is already configured and the operating mode PN interface of the SIMOTION controller is configured as an IO device. The transfer areas of the I-Device are already set up.

1. Click in the network view of TIA Portal on the SIMOTION controller.
2. In the Inspector window, select the "Properties > General" tab.
3. Select "PROFINET PN/IO interface > Operating mode > I-Device communication".
4. Click the entry for the transfer area, e.g. "Transfer area_1".

5. Check the "Activate PROFIenergy communication" box.
6. Look whether the box is checked for the other transfer areas that are to support PROFIenergy communication.

The I-Device can be exported and used for PROFIenergy communication.

Supported PROFIenergy functions and hardware

PROFIenergy is available as of SIMOTION V4.2 and SINAMICS V4.5.

SINAMICS S120 and SINAMICS G120 support the following PROFIenergy functionalities:

- PAUSE command
- Reading out two or three measured values
  - Active power P (MeasurementId = 34 in W)
  - Power factor cosφ (MeasurementId = 166, for G120 only)
  - Energy absorbed (MeasurementId = 200 in Wh)
- SINAMICS G120D-2 can also shut down encoders and I/Os.

Additional references

A detailed description of how to configure PROFIenergy communication in SIMOTION SCOUT TIA can be found in the *SIMOTION SCOUT TIA* Configuration Manual.

System function blocks _receiveRecord() and _provideRecord() for non-cyclic communication via the iDevice interface

Introduction

A SIMOTION controller uses these system function blocks to send and receive data sets via the iDevice interface. The user program must respond to a non-cyclic request. These system function blocks are used, among other things, to transfer PROFIenergy data sets.

Note

A detailed description of the system functions _provideRecord() and _receiveRecord() can be found in the online help and in the *SIMOTION System Functions/Variables Devices List Manual*.

_provideRecord()

The system function block _provideRecord() includes the following functionality:

- It checks cyclically (when mode=0) whether the iDevice has been sent a request to make a data set available.
- It makes the record data available to the higher-level controller.

_receiveRecord()

The system function block _receiveRecord() includes the following functionality:

- It checks cyclically whether the iDevice has been sent a request to receive a data set.
- It makes the data set available on the output parameters.
- It acknowledges receipt of the data by the higher-level controller.

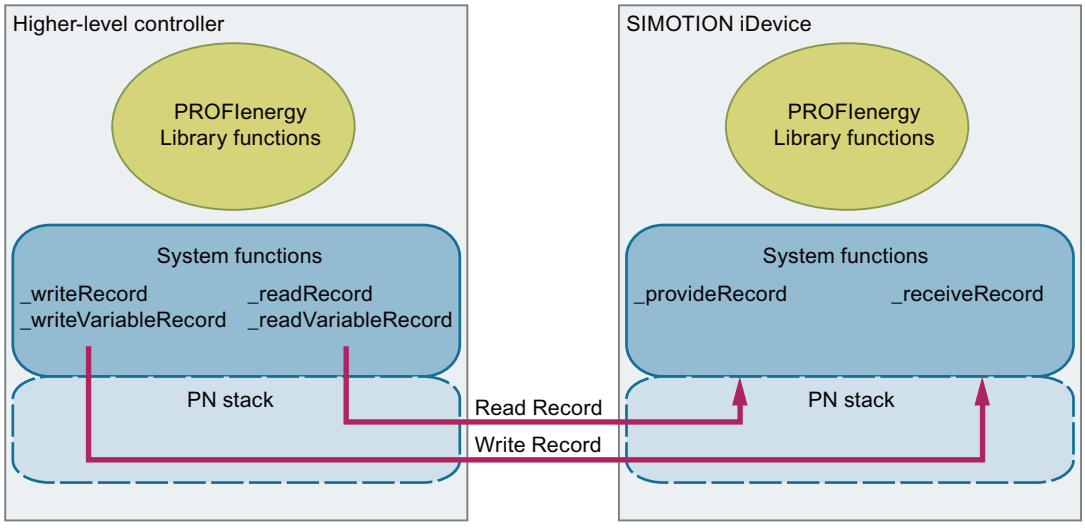


Figure 4-408 Read record and write record as iDevice

Function block for the SIMOTION PROFIenergy iDevice

PROFIenergy iDevice function block

A function block with an example application is available for decoding and coding the PROFIenergy data sets. This function block supports the following modes of operation:

- Forwarding the PROFIenergy commands from the higher-level controller to the lower-level PROFIenergy-compatible devices (e.g. ET200 or SINAMICS). Standard PROFIenergy function blocks that have already been implemented can be used in the user program (e.g. to read out the measured values).
- The PROFIenergy commands can be processed within the function blocks for a complete (production) cell; i.e. the PROFIenergy iDevice FB reflects the complete PROFIenergy functionality of a cell. As a prerequisite, there must be a SIMOTION CPU as the "head" controller. The shut-down and switch-on sequence for the individual PROFIenergy IO devices can be parameterized on the iDevice FB.

Download

The function block for the PROFIenergy iDevice with example application and comprehensive documentation can be found in the Support Portal (https://support.industry.siemens.com/cs/ww/en/view/58386840).

4.5.5.10 Series machine projects

IP address and device name via UP/DCP (Mini-IP-Config)

Description

It was previously only for I devices that SIMOTION supported the allocation of the IP address and device name (NameOfStation) from the user program or via the Discovery and Configuration Protocol (DCP). With SIMOTION V4.2 and Step 7 V5.5 or higher, this must be configured in HW Config and applies to all IO devices and IO controllers. To do this, you need to select the checkboxes for free allocation in the Properties of the Ethernet interface and the PROFINET interface. Previously in the case of SIMOTION, the IP configuration could only be set for the PN-IE interfaces via the user program; not for the PN-IO interfaces. As of V4.4, the IP address can be set from the user program for all PN interfaces.

A PN-IE/PN-IO interface is addressed in the system function via its diagnostics address (for the interface of the PN interface). You can also specify whether the IP configuration (device name, IP address) is to be set temporarily (until the power is next switched OFF/ON) or permanently (retained after the power is switched OFF/ON). If it is a temporary setting, the device name is deleted when the power is next switched OFF/ON and will contain a blank string.

With the new system functions, the IP addresses and the device names of the PN IE interfaces and the PN IO interfaces can be issued from the user program. An additional reboot is no longer necessary for the PN interfaces, as the set name does not also have to be activated via _activateNameOfStation.

This mechanism allows the IP addresses and device names to be adjusted without making changes to the project. The IP settings can be changed locally, especially for series machines.

New system functions as of V4.4

New system functions have been introduced as of SIMOTION V4.4 for assigning the IP address and device name. They expand the range of functions already available. The functionality of the existing system functions remains unchanged and they can still be used as before. You are advised to use the new system functions.

System functions up to V4.4	New system functions as of V4.4
_setNameOfStation/_activateNameOfStation	_setPnNameOfStation
_getActiveNameOfStation	_getPnNameOfStation
_getPnInterfacePortNeighbour	_getPnPortNeighbour (only PN IO interfaces)
_setIpConfig	_setPnIpConfig
_getIpConfig	_getPnIpConfig

Note

A detailed description of the system functions can be found in the online help and in the *SIMOTION System Functions/Variables Devices List Manual*.

Comparison of system functions for obtaining the IP address and device name via a different method

With the SIMOTION system functions, the following operations are possible:

	Assignment of device name via UP		Assignment of IP address via UP	
	(_setNameOfStation etc.)	(_setPnNameOfStation etc.) As of V4.4	(_setIPConfig/ _getIPConfig)	(_setPnIPConfig/ _getPnIPConfig) As of V4.4
PN IE interfaces	No	Yes	Yes	Yes
PN IO interfaces	Yes	Yes	No	Yes

Diagnostics when an error occurs

For the purpose of error diagnostics, a diagnostics buffer entry is created on the SIMOTION I device CPU if a DCP request to set the device name or IP address for the I device could not be carried out because the relevant checkboxes are not activated in HW Config.

Assigning the device name based on the MAC address (V5.2)

As of SIMOTION SCOUT V5.2, the device name can also be assigned based on the MAC address. The PN devices are named with the system function _assignNameOfStationToDevice() (Page 2282).

Configuring device names via system function

Introduction

The device name (NameOfStation or NameOfInterface) can be assigned via user program. This option must be explicitly enabled when configuring the hardware for this version.

Obtaining the device name using a different method

1. In HW Config, open the Properties dialog for the PROFINET interface.
2. Activate the **Obtain device name using a different method** checkbox on the General tab and confirm the selection by clicking **OK**.

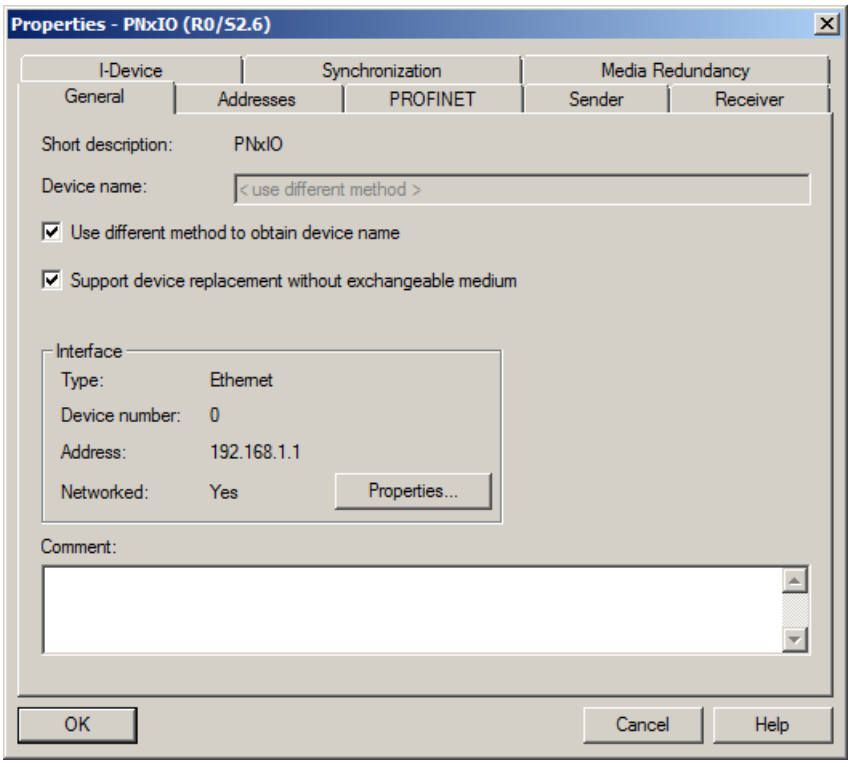


Figure 4-409 Obtaining the device name using a different method

Overview of the possible configuration methods

Configuration method		Response
Device name from project The option "Obtain device name using a different method" is **not** set.	System function **_setPnNameOfStation** or primary set-up tool via DCP	**The device name cannot be set.** The configured device name from the project is active. Following a restart, the device name from the project is still active.

Configuration method		Response
Device name not from project The option "Obtain device name using a different method" is set.	System function **_setPnNameOfStation** with parameter storeNameOfStationPermanent := YES	After fault-free performance of the system function, the set device name is active. Following a restart, the device name set by the system function is still active. **The device name from the project is no longer active.**
	System function **_setPnNameOfStation** with parameter storeNameOfStationPermanent := NO	After fault-free performance of the system function, the set device name is active. Following a restart, the device name set by the system function is no longer active. The device name contains an empty string. **The device name from the project is no longer active.**
	STEP7 / primary setup tool via DCP	The device name set via STEP7 / DCP is active. Following a restart, the device name set by STEP7 or the primary setup tool via DCP is still active. **The device name from the project is no longer active.**
	Through PROFINET controller via DCP (topology-based initialization)	**Before V4.5:** Following a restart, the device name set by the PROFINET controller is still active. Topology-based initialization only occurs when there is no existing device name (interface in factory setting). **The device name from the project is no longer active.** **As of V4.5 (only SIMOTION SCOUT TIA):** The device name set via DCP is active. The previous device name is overwritten.
	Through PRONETA via DCP with the "temporary" option (PRONETA supports the "temporary" option for tests)	Following a restart, the device name set by PRONETA is no longer active. The device name contains an empty string. **The device name from the project is no longer active.**
	System function **_assignNameOfStationToDevice**	**As of V5.2** The function block assigns the device name to an IO device with a specific MAC address. After error-free execution of the system function, the set device name is active. Following a restart, the device name set by the system function is still active.

Configuration method		Response
		The device name from the project is no longer active.

General information on device names

- If the setting "Use different method to obtain device name" is activated in the project, the last device name provided on the interface becomes active once the project is downloaded.
  This means:
  - The device name from the last project is active if a device name was configured in the previous project.
  - An empty string is set if the device name was in its factory setting (delivery state).
  - An empty string "" can be set via user program and DCP
  - A new device name can also be assigned via DCP and user program if a PROFINET connection (controller or I-Device) is active. Changing the name terminates the connection.
- In certain situations, the module is reset when downloading from SIMOTION SCOUT ("Target system reset" displayed in SCOUT log). This reset has the same effect on the NameOfStation as switching the power OFF/ON.

You will find an overview of the possible functions in the section IP address and device name via UP (Page 2277).

Note

If the option "Use different method to obtain device name" has been activated, the option "Use different method to obtain IP address" must also be activated.

Assigning the device name based on the MAC address

Initialization of PN devices with system function _assignNameOfStationToDevice()

With the system function _assignNameOfStationToDevice(), you assign a device name (NameOfStation) to a PN device with a specific MAC address. You can implement modular machine concepts with this function. During local commissioning, the technician assigns the device name directly for this plant extension based on the known MAC address, e.g. via the HMI. The HMI transfers the MAC address and the device name to the SIMOTION controller by means of variables.

Properties and supplementary conditions

- To set the IO device to a defined initial state, the system function resets the communication parameters of the IO device. Whereby, a permanent IP address is deleted, for example.
- The reset is performed via the ResetToFactory DCP service according to PN V2.3.
  If the IO device only supports PN V2.2 and rejects the ResetToFactory, only the PROFINET device name is assigned.

- No special characters such as umlauts, etc. are supported for the device name. **No** automatic conversion of the device name according to RFC 5890 is performed, for example, as in the TIA Portal.
- The device name is always set permanently. Temporary setting is not supported in contrast to the system function _setPnNameOfStation().

Configuring an IP address via system function

Introduction

The IP address can be assigned via user program. This option must be explicitly enabled when configuring the hardware for this version.

Obtaining the IP address using a different method

1. In HW Config, open the Properties dialog box for the PROFINET interface and click on the **Properties** button on the General tab.
2. Switch to the **Parameter** tab in the Properties dialog box that opens for the Ethernet interface.
3. Activate the **Use different method to obtain IP address** check box and confirm the selection by clicking **OK**.

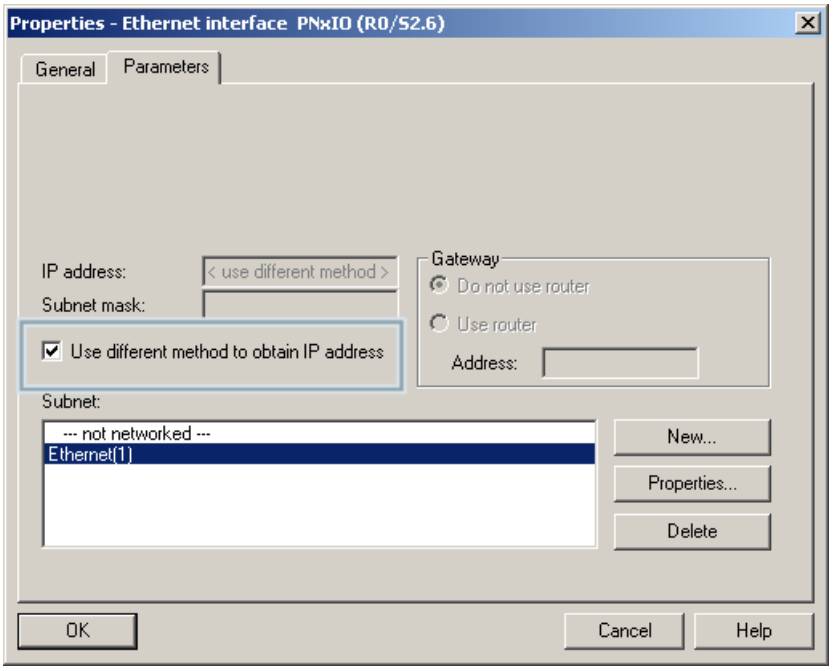


Figure 4-410 Obtaining the IP address using a different method

Overview of the possible configuration methods

Configuration method		Response
Project not present in the device	DCP (DCP BlockQualifier = permanent)	The IP address set via DCP is active. After a restart the IP address set via DCP is still active.
	DCP (DCP BlockQualifier = temporary)	The IP address set via DCP is active. After a restart the IP address 0.0.0.0 is active.
Project present in the device or not	Download project; in the project the "Obtain IP address using a different method" **option is not** set	The IP address loaded via the project is active. After a restart the IP address set via the project is still active.
	Download project; in the project the "Obtain IP address using a different method" option is set	The IP address last active on the interface is still valid. • The IP address from the last project, if a project with a valid IP address was previously loaded. • If a project with a valid IP address has not previously been loaded, then the IP address previously set via DCP / _setPnIPConfig or the default IP address as delivered.
Project present in the device The option "Obtain IP address using a different method" is **not** set	Opening of **_setPnIPConfig** to set an emergency IP address	The IP address loaded via the project is not temporarily overwritten by a so-called emergency IP address. Following a restart the IP address from the project is active again.
	DCP (DCP BlockQualifier = temporary or permanent) for setting an emergency IP address	
Project present in the device The option "Obtain IP address using a different method" is set	System function **_setPnIpConfig** with storeIp ConfigPermanent = YES	The IP address set via system function is active. After a restart the IP address set via the system function is still active.
	System function **_setPnIpConfig** with storeIp ConfigPermanent = NO	The IP address set via system function is active. After a restart the IP address 0.0.0.0 is active.
	STEP7 or primary setup tool via DCP (DCP BlockQualifier = permanent)	The IP address set via STEP7 / primary setup tool is active. After a restart the set IP address is still active.
	PROFINET controller via DCP (DCP BlockQualifier = temporary, only possible for PROFINET IO interfaces)	The IP address set via DCP is active. After a restart the IP address 0.0.0.0 is active.

General information on modifying the IP address

- If a PROFINET connection is active, DCP cannot be used to assign a new IP address. For safety reasons the setting of an IP address via DCP with a project present **in RUN and STOP operating state** is only permitted if **Obtain IP address using a different method** is set. If **Obtain IP address using a different method** is not set, the setting of an IP address is only permitted **in the STOP operating state**. This is intended to prevent an IP configuration set via a project, e.g. via the function "Edit Ethernet node," being changed easily from outside, by means of which a PROFINET communication of a SIMOTION device can be disrupted.
- An IP address can also be assigned via system function if a PROFINET connection (IO controller or I-device) is active. Changing the IP address via the system function leads to the breaking of the PROFINET connection.
- The IP address 0.0.0.0 can be set via the user program and DCP.
- In certain situations, the module is restarted when downloading with SIMOTION SCOUT ("Target system reset" displayed in SCOUT log). Such a restart has an impact on when IP addresses come into force.
- Emergency IP address:
  The emergency IP allows the user to temporarily overwrite the IP configured in the project via DCP or system function, e.g. in order to load a project with a corrected IP address if access is no longer possible via the IP address configured in the project. When downloading the new project, make sure that the emergency IP address assigned via "Edit Ethernet node" agrees with the IP address used in the project, because in certain cases SIMOTION performs restarts during downloading, and the IP address from the project subsequently becomes effective.

You will find an overview of the system functions in the chapter IP address and device name via UP (Page 2277).

ResetToFactorySettings via DCP

Description

A resetting of the device names or the IP address to the factory settings (ResetToFactorySettings) via DCP to one of the PN IO or PN IE interfaces sets the parameters as follows:

- IP address: 0.0.0.0
- NameOfStation: Blank string

Boundary condition

Due to security restrictions, DCP ResetToFactorySettings is only permitted when an iDevice is configured with **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and, at the same time, **Use different method to obtain device name** and **Use different method to obtain IP address** are both set. The interface must not be used as an IO controller, i.e. no IO device is configured.

For an interface with a default IP address, the response after a ResetToFactory is as follows:

- When a ResetToFactorySettings is performed, the IP address is immediately set to 0.0.0.0.
- If the power is then switched OFF/ON, the IP address remains 0.0.0.0.
- The default IP address is not reactivated until the NVRAM is erased.

Reusable IO systems - Addresstailoring

Introduction

The Addresstailoring function assigns the IP address and NameOfStation for IO controllers and IO devices via system functions at commissioning time. This feature allows a machine consisting of an IO controller and IO devices to be supplied with IP addresses and NameOfStation, and permits multiple instances of the machine to be commissioned on a single physical Ethernet without the engineering system.

There is only one project (configuration and programs). It can be loaded onto machines of the same type without change. To integrate the machine in an existing network infrastructure, only limited customization is required for the on-site commissioning (IP address, NameOfStation). The user program assigns the IP address and the NameOfStation during the first controller startup.

Note

Addresstailoring can be configured only for SIMOTION SCOUT TIA.

Example of an automation solution

The following figure shows how an automation solution with a reusable IO system is loaded into various automation systems and one of these is then adapted on-site to the existing network infrastructure.

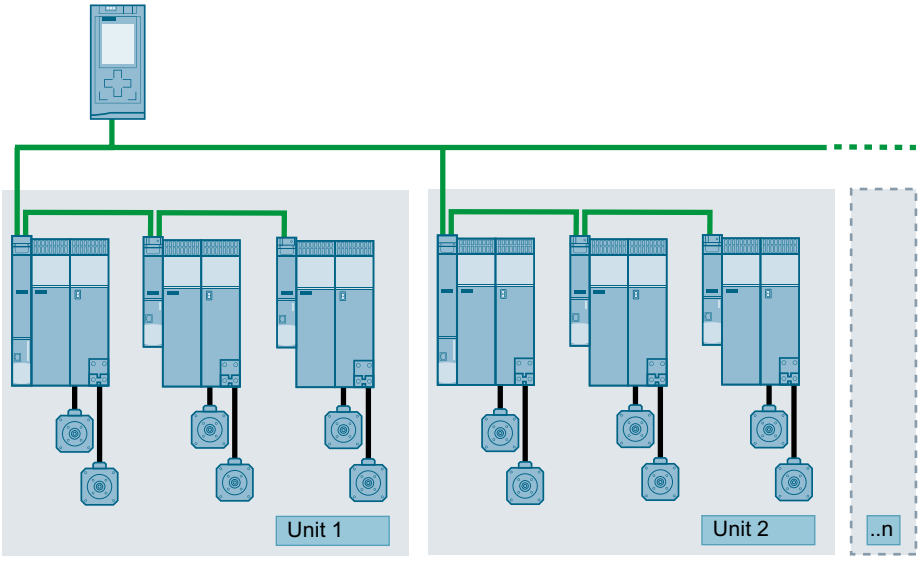


Figure 4-411 Addresstailoring: Series machine with the same units with a different name and IP address

Rules

The following rules apply to a reusable IO system:

- A series machine project consists of an IO controller and the associated IO devices. Consequently, configure only one CPU as IO controller and the associated IO devices in the series machine project.
- No IO device may be configured as shared device.
- If IRT is configured, all IO devices must belong to the same sync domain and the sync domain must not contain any further IO devices.

Alternatively to Addresstailoring via two PN interfaces

In the SIMOTION environment, under some circumstances, rather than using Addresstailoring, the same functionality can also be achieved by using the two PN interfaces. One PN interface is configured as I device and connected with the higher-level controller. The second PN interface is the IO controller for the IO devices. In this case, a second PN interface (e.g. CBE30-2) is required.

In the previously described configuration, the IO devices are located within a local network and so can have the same NameOfStation and IP address in all instances.

Additional references

Further information can be found in the *Basic Functions for Modular Machines Function Manual* and in the *online help for the TIA Portal*.

Configuration control for IO systems - Machinetailoring

Introduction

The configuration control for IO systems (Machinetailoring) allows several specific instances or configurations of the series machine to be generated from a single series machine project. The configuration of an IO system can vary flexibly for a specific application provided the real configuration can be derived from the configured configuration. The configured configuration (series machine project) so forms the superset of all derivable real configurations.

You can activate or deactivate a PROFINET IO system (IO devices, I devices) in a specific system. Different variants can be operated from a configured maximum configuration of an IO system. You prepare for a series machine project a modular system consisting of IO devices that can be adapted flexibly for the wide range of configurations via the configuration control.

Machinetailoring allows the following customizations to be made on-site (without engineering system):

- Variation of the number of associated modules (IO devices). This is also possible with restrictions with IRT (IRT node as leaf for the IRT node part in a line).

Note

Machinetailoring can be configured only for SIMOTION SCOUT TIA.

PROFINET IRT

Machinetailoring is possible for PROFINET IRT only when minor adaptations need to be made to the IRT devices of the master machine. Minor adaptation means that IRT IO devices are only bridged or individual branches and leaves are removed.

Example of an automation solution

The following figure shows an example from a series machine project how an instance can be created with a different number of IO devices. The IO devices in the middle and at the end can be optional.

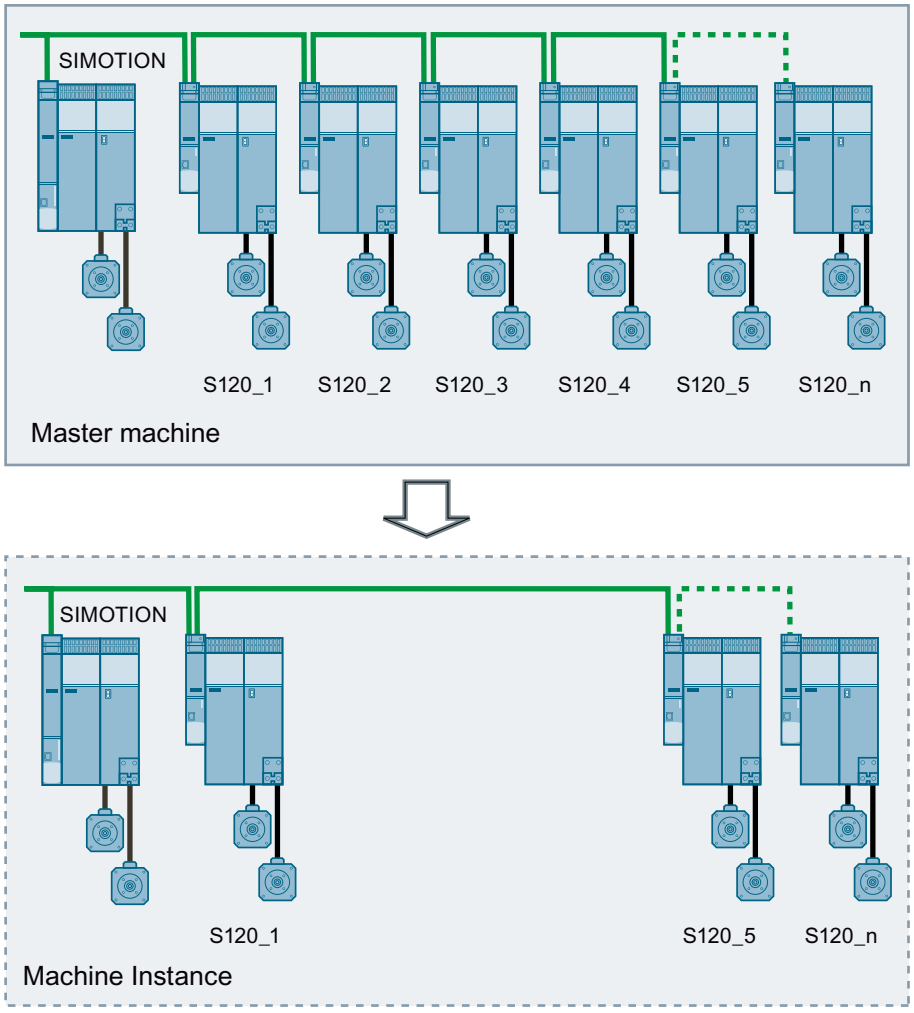


Figure 4-412 Instance of the machine with a different number of IO devices

Restrictions

- The number of optional IO devices in series is limited.
- The data exchange between IO controllers (controller-controller slave-to-slave communication) is possible only from the user program.
- Because it cannot be guaranteed that the optional IO devices actually exist in the machine instance, data exchange between IO devices is not meaningful.

Additional references

Further information can be found in the *Basic Functions for Modular Machines Function Manual* and in the *online help for the TIA Portal*.

4.5.6 Ethernet: General information (TCP and UDP connections)

4.5.6.1 Ethernet interfaces

Overview of Ethernet

Overview

Below is a description of how to configure TCP and UDP Ethernet connections between communications partners. TCP and UDP are based on Ethernet and the IP protocol.

Properties of the SIMOTION Ethernet interfaces

Ethernet interfaces

Depending on the device, SIMOTION has one or three onboard Ethernet interfaces, and one or two PROFINET IO interfaces. You can connect an Industrial Ethernet with a transmission rate of 10/100 Mbps or 1,000 Mbps to the RJ45 sockets with D4x5-2 DP/PN.

With several interfaces, TCP/IP timeout parameters can be set once for both interfaces. With several interfaces, the transmission rate/duplex can be set separately for the two interfaces. Utilities via TCP/IP and UDP are supported for both Ethernet interfaces, enabling S7 routing between all interfaces (including PROFIBUS DP). "Utilities via TCP" are not routed from one Ethernet interface to the other.

Alternatively, you can also connect an Industrial Ethernet through the onboard PROFINET interface of a SIMOTION D4x5-2 DP/PN (100 Mbit/s).

The MAC addresses of the Ethernet interfaces can be seen on the outside of the housing.

Note

Default gateway (network transition/router)

In a SIMOTION controller, only one of the existing Ethernet or PROFINET interfaces (onboard PN interfaces, CBE30-2...) can be configured as a default gateway (network transition/router). On the other Ethernet or PROFINET interfaces, the IP address and the subnet mask can be set. If more than one default gateway is configured on the SIMOTION controller, this will be output as an error message on the consistency check as of Step 7 V5.5 SP4. In older Step 7 versions, this check is is only made on the project download.

Converting standard Ethernet interfaces from V4.3 and higher

With SIMOTION V4.3, the standard Ethernet interfaces are marked as PN-IE interfaces:

- Determining neighborhood information via LLDP (Link Layer Discovery Protocol). To do so, use the system function _getPnPortNeighbour in SIMOTION.
- Via the Step7 diagnostic address, you can perform diagnostics from the user program for the Ethernet interface.
- You can assign the device name (NameOfStation) either via DCP or via the user program (see also IP address and device name (Page 2219)).
- You can assign the IP address either via DCP or via the user program.
- Provision of topology information via SNMP (Simple Network Management Protocol) (e.g. via the STEP7 Topology Editor, see also Topology editor (graphical view) (Page 2209)).
- You can configure a setpoint topology.
- Online diagnostics of the interface and ports via STEP7 device diagnostics.

The interfaces in HW Config are displayed as follows:

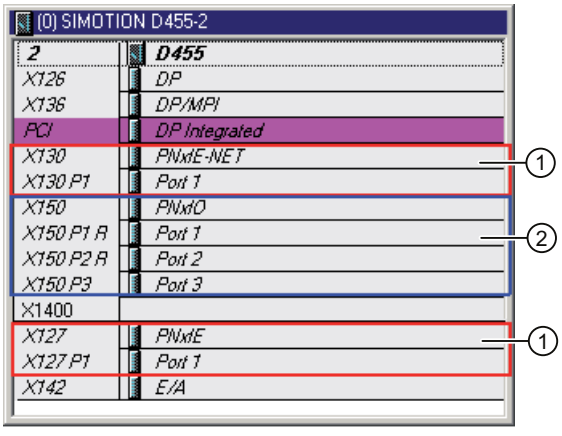


Figure 4-413 Ethernet interfaces in HW Config

①	Standard Ethernet interface with one port
②	PROFINET IO interface with three ports

See also

IP address and device name (Page 2219)

Topology editor (graphical view) (Page 2209)

Using the Ethernet interface

Using the Ethernet interface

- For communication with STEP 7, SIMOTION SCOUT, and SIMATIC NET OPC via a PG/PC
- For communication via UDP (User Datagram Protocol) with other components, e.g. other SIMOTION devices, SIMATIC devices, or PCs
- For communication via TCP (Transfer Control Protocol) with other components, e.g. other SIMOTION devices, SIMATIC S7 stations, or PCs
- For connecting SIMATIC HMI devices such as MP277, MP370, or PC-based HMIs
- For communication by means of SIMOTION IT Web server and SIMOTION IT OPC XML DA (no license required as of V4.2)
- For communication by means of SIMOTION VM (separate license required)

4.5.6.2 LCom communications library

LCom library

The **LCom** library is based on TCP and simplifies the use of both SIMOTION system functions and SIMATIC communication blocks in order to establish communication between multiple machines.

Note

You can find the LCom library on the SIMOTION SCOUT DVD "Documentation, Utilities & Applications". The DVD also contains an example project and comprehensive documentation for the library.

Types of control

The LCom library supports the following control types and combinations:

- SIMOTION ↔ SIMOTION
- SIMATIC ↔ SIMATIC
- SIMOTION ↔ SIMATIC

Functions

- The send and receive data must be of data type BYTE. Outside of the *FBLComMachineCom*, any user structures can be converted into an ARRAY OF BYTES via marshalling.
- Bi-directional operation
  - A logical point-to-point connection is established between two controls.
  - Each control can send and receive via a connection at the same time.
- Alignment of configuration settings between communications partners (e.g. send clock)
  - Assignment of communication parameters to the communications partner
  - Changing the configuration during operation
- Types of data transmission
  - Cyclic transmission (transmission at fixed periods of time)
  - Transmission when data is changed
  - One-off transmission
- Sending and receiving max. 64 kB useful data
- Acknowledgment and monitoring of received data
- Sign-of-life monitoring
- Time-of-day synchronization

4.5.6.3 TCP communication

Overview of TCP communication

Communication with TCP (Transfer Control Protocol)

Communication via TCP is connection-oriented, i.e. data can only be transferred once a connection to the communications partner has been established.

The main features of a communication connection are two end points. An end point is a controlled pair consisting of an IP address and a port. The port on the client may be the same as or different from the port on the server. Usually, one end point represents a server and the other end point a client.

The client-server relationship is only valid until the connection is established. After the connection is established, both communications partners are equivalent, i.e. each of the two can send or receive or close the connection at any point in time.

Principle of TCP communication (see figure)

- Server waits at port (1)
- Client announces connection request at this port (2). If a port is not announced on the server, there is a wait with TimeOut (system setting)

- Server creates internal communication port with connection announcement and releases server port for new connection. The internal communication port is identified via the connectionId (3)
- Possible to send/receive data via this connection not only from the client, but also from the server (4)
- Further connections can be established at the server port (5)
- An existing connection can be closed on the client or server side (6)
- Server port to establish connection is closed on the server side (7)

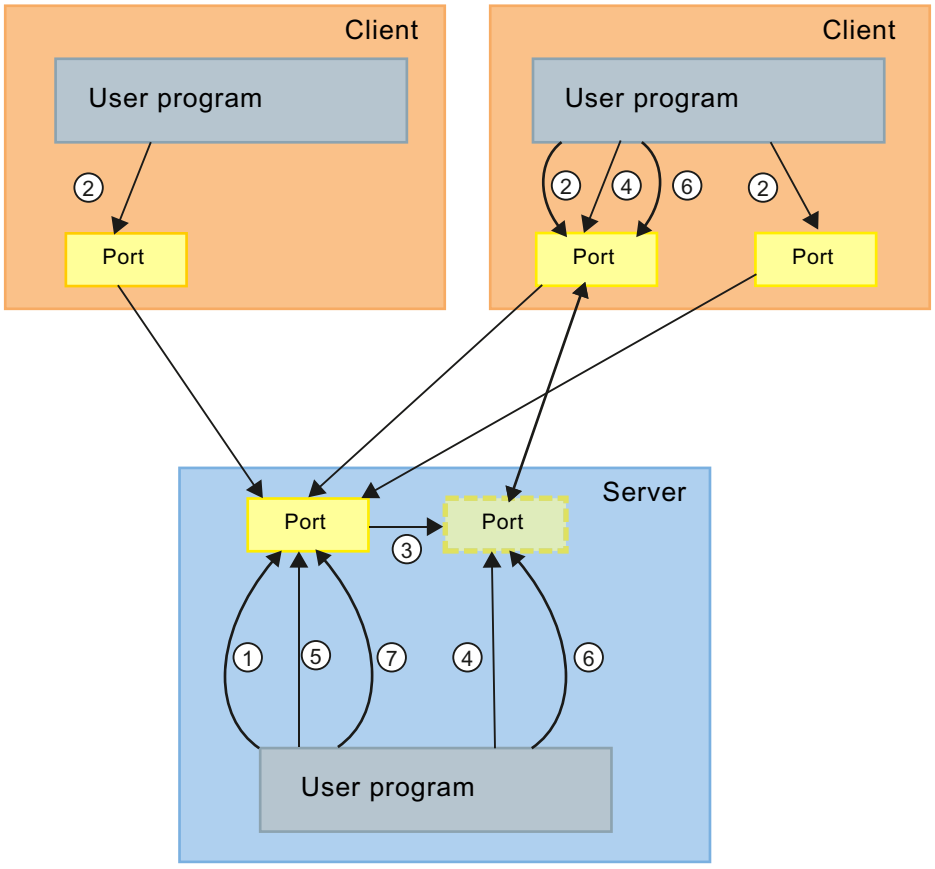


Figure 4-414 Schematic diagram showing TCP communication sequence

Principles of port assignment:

- The port number on the client can be the same as the port on the server.
- The port number on the client can be the different to the port on the server.

Data exchange between user program and TCP stack

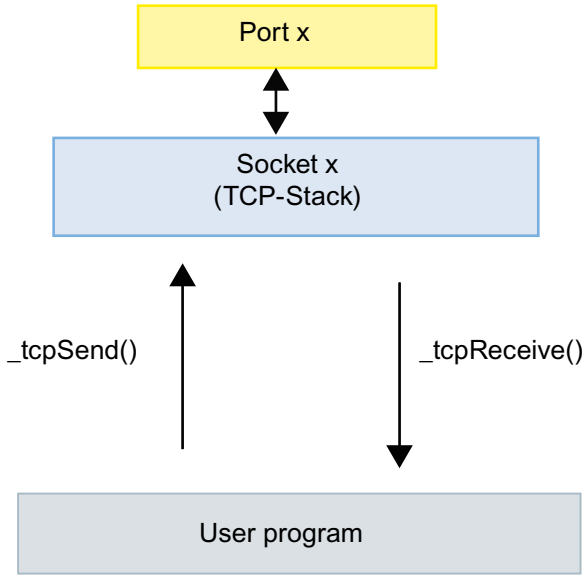


Figure 4-415 Data exchange between user program and TCP stack

Data packets

With TCP communication, the useful data received is buffered in the TCP stack and must be read out from here by the application. The size of the buffered useful data per connection depends on the control involved.

Table 4-232 Useful data and control

Control	Buffer size in bytes
SIMOTION	4096
SIMATIC	8192

The buffered useful data must be fetched promptly by the user program, otherwise it will not be possible for any other data to be received. However, by contrast none of the useful data is discarded, as TCP has a flow control facility. In this state, the communications partner does not send any additional data and alerts its application to this fact.

SIMOTION system functions for TCP communication

Overview of SIMOTION system functions

Maximum number of possible TCP connections

The table below contains examples of the number of possible communication connections for a SIMOTION CPU acting as a client. The values relate to a local network without any other external load sources, and a SIMOTION D435 acting as a server.

Table 4-233 Communication connections in relation to a SIMOTION CPU

SIMOTION CPU (client)	Number of communication connections
C2xx	45
D410	45
D410--2	45
D4x5	75
D4x5-2	75
P3x0	40

Function call

The SIMOTION system functions for TCP communication may only be called in the BackgroundTask or in a MotionTask. In the BackgroundTask, it should be noted that the system functions are executed asynchronously (nextCommand=IMMEDIATELY). In the MotionTask, however, the system functions can be executed synchronously (nextCommand=WHEN_COMMAND_DONE).

Note

To ensure reliable cyclic communication, you should use standard mechanisms such as those found in PROFIBUS DP or PROFINET IRT systems.

See also

_tcpOpenServer() system function (Page 2296)

_tcpOpenClient() system function (Page 2297)

_tcpSend() system function (Page 2298)

_tcpReceive() system function (Page 2298)

_tcpCloseConnection() system function (Page 2299)

_tcpCloseServer() system function (Page 2300)

_tcpOpenServer() system function

Overview

This system function is used in order to initiate passive establishment of a connection.

Table 4-234 Call example

```
sRetValueTcpOpenServer := _tcpOpenServer(  //StructRetTcpOpenServer
   port        := 3456                     //UINT, 1024-65535
  ,backLog     := 5                        //DINT
  ,nextCommand := IMMEDIATELY              //EnumTcpNextCommandMode
);
```

To parameterize the function, the locally assigned SIMOTION port is transferred for the **port** parameter.

The maximum number of parallel connection requests for this port that are to be permitted from other controllers is also specified as a further parameter for **backLog**.

The behavior of this function with respect to the advance when called is also parameterized with the **nextCommand** parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value the advance is immediate and with the second value it is after completion of the command.

When the _tcpOpenServer() function is called, the structure returned to the user program contains the following parameters.

The status of establishing the connection can be queried via the **functionResult** parameter.

The **connectionId** parameter is used as an (input) parameter for the call of the _tcpSend() and _tcpReceive() functions and assigns a unique TCP connection to these functions. This return value is referred to in the above call examples.

The **clientAddress** parameter returns the client IP address from which the connection is activated; this takes the form of an array.

The port number designated as the local port number of the client is specified in the **clientPort** parameter.

The port number is in the range 1024 to 65535.

_tcpOpenClient() system function

Overview

The _tcpOpenClient() system function is used to actively establish a connection.

Table 4-235 Call example

```
sRetValueTcpOpenClient :=_tcpOpenClient( // StructRetTcpOpenClient
   port           := 3456                // UINT, 1024-65535
  ,serverAddress  := au8ServerAddress    // ARRAY [0..3] OF USINT
  ,serverPort     := 3456                // UINT, 1024-65535
  ,nextCommand    := IMMEDIATELY         // EnumTcpNextCommandMode
  );
```

When called, the locally assigned SIMOTION port is transferred to the function for the **port** parameter.

The **serverAddress** parameter is the IP address of the communications partner, which is transferred in an array.

The port number designated as the local port number is transferred to the function in the **serverPort** parameter.

The port number is in the range 1024 to 65535.

The behavior of this function with respect to the advance when called is parameterized with the **nextCommand** parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediate and with the second value it is after completion of the command.

When the _tcpOpenClient() function is called, the structure returned to the user program contains the following parameters.

The status of establishing the connection can be queried via the **functionResult** parameter.

The **connectionId** parameter is used as an (input) parameter for the call of the _tcpSend(), _tcpReceive(), and _tcpCloseConnection() functions and assigns a unique TCP connection to these functions.

_tcpSend() system function

Overview

The _tcpSend() system function is used for sending data.

Table 4-236 Call example

```
i32RetValue := _tcpSend(                                    // DINT
   connectionId    := sRetValueTcpOpenClient.ConnectionId  // DINT
  ,nextCommand     := IMMEDIATELY                          // EnumTcpNextCommandMode
  ,dataLength      := 4096                                 // UINT, 0-4096
  ,data            := ab8SendData                          // ARRAY [0..4095] OF BYTE
 );
```

For the **connectionId** parameter, the **connectionId** return value of the _tcpOpenClient() or _tcpOpenServer() function is transferred, depending on whether the communication node that executes the function is a server or client.

The behavior of this function with respect to the advance when called is also parameterized with the **nextCommand** parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value the advance is immediate and with the second value it is after completion of the command.

The **dataLength** parameter informs the function of the useful data length in bytes that has to be transferred (max. 4,096 bytes per function call).

The **data** parameter specifies the useful data area in which the send data that is to be transferred with the function is located.

The return value of the function to the user program is of data type DINT. The various return values indicate any problems that occurred during execution of the function. There is also a confirmation when the data has been successfully sent. Negative values in **functionResult** indicate a data transmission error. In this case, the connection must be closed by calling _tcpCloseConnection().

_tcpReceive() system function

Overview

The _tcpReceive() system function is used for receiving data.

Table 4-237 Call example

```
sRetValueTcpReceive := _tcpReceive(                          // StructRetTcpReceive
   connectionId     := sRetValueTcpOpenClient.connectionId  // DINT
  ,nextCommand      := IMMEDIATELY                          // EnumTcpNextCommandMode
  ,receiveVariable  := ab8ReceiveData                       // ARRAY [0..4095] OF BYTE
  );
```

For the **connectionId** parameter, the **connectionId** return value of the _tcpOpenClient() or _tcpOpenServer() function is transferred, depending on whether the communication node that executes the function is a server or client.

The behavior of the __tcpReceive() function with respect to the advance when called is also parameterized with the **nextCommand** parameter. There are two setting options: IMMEDIATELY and WHEN_COMMAND_DONE. With the first value, the advance is immediate and with the second value it is after completion of the command.

For each function call, up to 4,096 bytes of receive data can be read out from the TCP stack. Please note that it will not be possible to predict the sizes of the packets received. On the receiver side, you must take care to ensure that all user data is present before the evaluation and further processing. For this purpose, the **nextCommand** parameter should be set to IMMEDIATELY.

The **receiveVariable** parameter is used to inform the function of the user data area in which the receive data is to be stored. The received data is available in the **receiveVariable** parameter in the length **dataLength**, if the return value in the functionresult = 16#0. At the next function call, the **ReceiveVariable** parameter is overwritten with new data in the length **dataLength**.

When the _tcpReceive() function is called, the structure returned to the user program contains the following parameters. The receive status can be queried via the **functionResult** parameter. The **dataLength** parameter signals the number of received user data bytes once the _tcpReceive() function has been successfully called.

Negative values in **functionResult** indicate a data transmission error. In this case, the connection must be closed by calling _tcpCloseConnection().

_tcpCloseConnection() system function

Overview

The _tcpCloseConnection() function is used for closing a connection that has been actively established by the communication node executing the function (client).

Table 4-238 Call example

```
sRetValueTcpCloseConnection := _tcpCloseConnection(      // DINT
   connectionId := sRetValueTcpOpenClient.ConnectionId   // DINT
  );
```

The return value of the _tcpOpenClient() function is transferred at the **connectionId** parameter in order to clearly identify which connection is to be closed.

The return value of the function to the user program is of data type DINT and indicates any problems that occurred during execution of the function or signals if the connection has been closed successfully.

_tcpCloseServer() system function

Overview

The _tcpCloseServer() function is used for closing a connection that has been passively established by the communication node executing the function (server).

Table 4-239 Call example

```
sRetValueTcpCloseServer := _tcpCloseServer(port := 3456); //1024 - 65535
```

The server port is transferred at the **port** parameter.

The return value of the function to the user program is of data type DINT and indicates any errors that occurred during parameterization of the function or signals if the port has been closed successfully.

4.5.6.4 UDP communication

Overview of UDP communication

Communication with UDP (User Datagram Protocol)

UDP offers a method of sending and receiving data over Ethernet from the user program with a minimum use of protocol mechanisms. No information concerning the transferred data is returned in the case of communication via UDP. Communication takes place via ports on both the send and receive sides. Unlike TCP, you do not need to program any connection establishment or closing.

Principle of UDP communication

- For reception, in the command you address the port that you want to use on your component for the communication job.
- When sending data, you specify the IP address and port number of the target system, as well as the port number of the local control.
- You can specify whether the port should remain reserved on your end after the communication job has been executed.
  UDP is not a secured model. Therefore, data may be lost during transmission if it is not read from the buffer in good time. You must program a secured method of data transmission via your application, e.g. by acknowledging receipt of the data.

- At the very least, the following data is required for sending:
  - IP address of communications partner
  - "Own" port number
  - Port number of communications partner
- UDP is not a secured transfer protocol. You must program feedback concerning the success of data transmission in the user program yourself.

Data exchange between user program and UDP stack

The following figure shows the UDP communication model at the SIMOTION end.

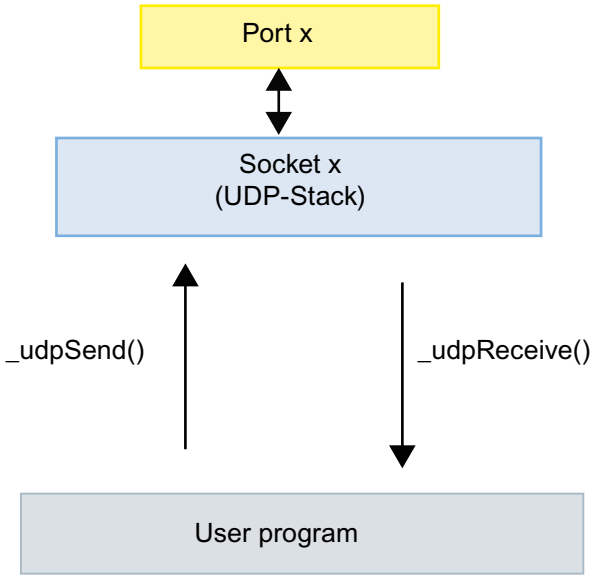


Figure 4-416 Data exchange between user program and UDP stack

With UDP communication, the useful data received is buffered in the UDP stack and must be read out from here by the application. The size of the buffered useful data per connection depends on the control involved. If the receive buffer is not read out on time, the received UDP telegrams will be lost once a new UDP telegram is received.

Table 4-240 Useful data and control

Control	Buffered useful data in bytes
SIMOTION	1,470 (SIMOTION V4.1 SP4 and higher) 1,400 (up to SIMOTION V4.1 SP3)
SIMATIC	Up to 2,048 (depending on CPU and communication via Ethernet CP or integrated Ethernet interface)

SIMOTION system functions for UDP communication

Overview of SIMOTION system functions

Function call

The SIMOTION UDP system functions may only be called in the BackgroundTask or in a MotionTask.

Data is sent via _udpSend(). If data is to be received on the SIMOTION side, the _udpReceive() function is used. The sections that follow describe the functions.

Note

To ensure reliable cyclic communication, you should use standard mechanisms such as those found in PROFIBUS DP or PROFINET IRT systems.

See also

_udpSend() system function (Page 2302)

_udpReceive() system function (Page 2304)

_udpAddMulticastGroupMembership() system function (Page 2305)

_udpDropMulticastGroupMembership() system function (Page 2306)

_udpSend() system function

Overview

The _udpSend() system function is used for sending data.

Table 4-241 Call example

```
i32RetValue :=_udpSend(
   sourcePort         := 3456                   //UINT, 1024 - 65535
  ,destinationAddress := au8DestinationAddress  //ARRAY[0..3] OF USINT
  ,destinationPort    := u16DestinationPort     //UINT, 1024 - 65535
  ,communicationMode  := CLOSE_ON_EXIT          //EnumUdpCommunicationMode
  ,dataLength         := u32DataLength          //UDINT
  ,data               := ab8SendData                //ARRAY[0..1469] OF BYTE
  );
```

When the _udpSend() function is called, the local port is transferred for the **sourcePort** parameter. The **destinationAddress** parameter is the IP address which is transferred in an array. The IP address can be configured and read out in HW Config.

The port of the communications partner is transferred as the **destinationPort**.

The user can use **communicationMode** to specify whether the communication resources are to be released after sending (CLOSE_ON_EXIT) or not released after sending (DO_NOT_CLOSE_ON_EXIT).

The **datalength** and **data** parameters specify the data length to be sent or the area where the send data is stored.

The status of the send job can be checked via the return value of the function.

Note

Up to SIMOTION 4.1 SP4, the length of the send data (data) was limited to 1,400 bytes.

Supplementary condition for UDP broadcast (as of V4.5)

If you want to use Open User Communication with UDP broadcast (_updSend(), _udpReceive()), you must configure this explicitly in SIMOTION SCOUT/SIMOTION SCOUT TIA. Broadcasts are filtered in the standard configuration in order to avoid potential network load problems. In other words, any broadcast that is not related to PROFINET services is discarded before it enters the TCP/IP stack. To enable UDP broadcasting, you must uncheck the box for the relevant network load filter.

You will be reminded during runtime by a corresponding error message that you need to check this box if you have programmed an _updSend() to a broadcast address.

Compatibility with older software versions

With older projects that you have upgraded to version V4.5 or set up as new projects in V4.5, the network load filter for UDP broadcasting is automatically active. If you want to communicate by UDP broadcast, you must check the box for use of UDP broadcast communication.

Activating communication with UDP broadcast in SIMOTION SCOUT

1. Click the SIMOTION controller in HW Config, e.g. D455.
2. Select "Object properties" from the context menu.
3. Check the box "Permit broadcast for _udpSend/_udpReceive" on the tab "Ethernet Extended / Web Server".

Activating communication with UDP broadcast in SIMOTION SCOUT TIA

1. Select the SIMOTION device in the network view in TIA Portal.
2. In the Inspector window, select the "Properties > General" tab and click "Web server/OPC/ Network protocols".
3. Check the box "Permit broadcast for _udpSend/_udpReceive".

Programming checkbox status (SIMOTION SCOUT only)

It is possible to set and read the checkbox value using the Step7 command interface and scripting functions. The attribute UDP_BROADCAST_ACTIVATE is used.

The following options are available for reading and setting the checkbox:

1. Directly using the Step 7 scripting mechanism for reading and writing.
2. External SCOUT scripting.
   If you are using a programming language that permits use of language-specific data types (e.g. LONG) (such as Visual Basic, MS Access), you can read and set the checkbox.
3. Internal SCOUT scripting.
   The VB script is used in this case. Only VARIANT data types are supported. In this case, the checkbox can only be set but not read if variables are not used.

Programming example for 1 and 2:

```
Private Sub CreateModule(Project As String, Station As String,
IOSystem As String, Value As String)
 Dim stationObj As S7Station
 Dim SimaticObj As New Simatic
 Dim Udp As String
 Dim io As Long
 Dim channel As Long

 Set stationObj = SimaticObj.Projects(Project).Stations(Station)
 Set cpuobj = stationObj.Racks(0).Modules(2)
 Set Udp = 1
 Call cpuobj.SetParameter("UDP_BROADCAST_ACTIVATE", 1, 0, 0)
‘ Writing
Call cpuobj.GetParameter("UDP_BROADCAST_ACTIVATE", Udp, io, channel)
‘ 'Reading
```

Programming example for 3:

```
Set oS7 = CreateObject("Simatic.Simatic")
oS7.Projects(PROJ.Name).Stations("SIMOTION
D").Racks(0).Modules(2).Putparameter "UDP_BROADCAST_ACTIVATE",1,0,0)
```

_udpReceive() system function

Overview

The _udpReceive() system function is called for receiving data.

Table 4-242 Call example

```
sRetValueUdpReceive :=_udpReceive(
   port               := 3456                  //UINT, 1024 - 65535
  ,communicationMode  := CLOSE_ON_EXIT         //EnumUdpCommunicationMode
  ,nextCommand        := IMMEDIATELY           //EnumNextCommandMode
  ,receiveVariable    := ab8ReceiveData               //ARRAY[0..1469] OF BYTE
  );
```

When the function is called, the local port is transferred for the **port** parameter.

In this case too, the user can use **communicationMode** to specify whether the communication resources are to be released after reception (CLOSE_ON_EXIT) or not released after reception (DO_NOT_CLOSE_ON_EXIT).

The behavior of this function with respect to the advance when called is parameterized with the **nextCommand** parameter. There are three setting options for this parameter: IMMEDIATELY, WHEN_COMMAND_DONE, and ABORT_CURRENT_COMMAND. With the first two values advance is either immediate or after completion of the command. With the third value, if the same port number as in the previous function call is transferred, the active function is aborted.

The **receiveVariable** parameter specifies the buffer in which the receive data is stored.

When the _udpReceive() system function is called, the structure returned to the user program contains the following parameters. The call status of the receive function can be queried in the **functionResult** parameter.

The **sourceAddress** parameter is an array that contains the IP address. The **sourcePort** parameter of the structure also contains the local port.

The number of received useful data bytes after a successful call of the _udpReceive() system function can be read out in the **dataLength** parameter.

Note

Up to SIMOTION 4.1 SP4, the length of the receive data (receiveVariable) was limited to 1,400 bytes.

Supplementary condition for UDP broadcast (as of V4.5)

If you want to use Open User Communication with UDP broadcast (_updSend(), _udpReceive()), you must configure this explicitly in SIMOTION SCOUT/SIMOTION SCOUT TIA. Please also read the section "Supplementary conditions for UPD broadcast" in chapter System function _udpSend() (Page 2302).

_udpAddMulticastGroupMembership() system function

Overview

This function is used to join a multicast group on a selected Ethernet interface.

A maximum of three multicast groups can be created on an Ethernet interface. These can occupy the same port or different ports.

```
myRetDINT :=
  _udpAddMulticastGroupMembership(
      multicastIPAddress :=
      ,interfaceIPAddress :=
      ,multicastPort :=
      // ,multicastTTL := 1
      // ,multicastLOOP := 1
      ,nextCommand :=
);
```

Note

In the case of UDP multicast communication, the _multicastPort must correspond to the destinationPort for function _udpSend.

Note

The UDP multicast functions (_updAddMulticastGroupMembership(), updDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.

_udpDropMulticastGroupMembership() system function

Overview

This system function is used to exit a multicast group on a selected Ethernet interface.

```
myRetDINT :=
  _udpDropMulticastGroupMembership(
      multicastIPAddress :=
      ,interfaceIPAddress :=
      ,multicastPort :=
      ,nextCommand :=
);
```

Note

The UDP multicast functions (_updAddMulticastGroupMembership(), updDropMulticastGroupMembership()) can only be used on Ethernet interfaces. PROFINET interfaces are not supported.

4.5.6.5 Services used

Communication services and port numbers used

Introduction

SIMOTION supports the protocols listed in the following table. The address parameters, the relevant communication layer as well as the communication role and the communication direction are specified for each protocol. This information allows you to match the security measures for the protection of the automation system to the protocols used.

Documentation of Ethernet services used

Protocol	Port number	Transport level	Role/direction	Function	Description
PROFINET protocols					
DCP Discovery and Configuration Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/ incoming Client/ outgoing	Accessible nodes, PROFINET discovery and configuration	DCP is used by PROFINET to determine PROFINET devices and to make basic settings.
LLDP Link Layer Discovery Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x88CC (PROFINET)	Server/ incoming Client/ outgoing	PROFINET Link Layer Discovery Protocol	LLDP is used by PROFINET to determine and manage neighborhood relationships between PROFINET devices.

MRP Media Redundancy Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x88E3 (PROFINET)	Server/ incoming Client/ outgoing	PROFINET media redundancy	MRP enables the control of redundant routes in a ring topology.
PTCP Precision Transparent Clock Protocol	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/ incoming Client/ outgoing	PROFINET send clock and time synchronization, based on IEEE 1588	PTCP enables a time delay measurement between RJ45 ports and, therefore, enables send clock synchronization and time synchronization.
PROFINET IO data	Not relevant	Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Server/ incoming Client/ outgoing	PROFINET cyclic IO data transfer	The PROFINET IO telegrams are used to transfer IO data cyclically between the PROFINET IO controller and IO devices via Ethernet.
Internal protocol	34962 34963	UPD	Reserved	Reserved	The ports are reserved for PROFINET extensions.
PROFINET Context Manager	34964	UDP	IO device/ incoming IO Client/ outgoing	PROFINET connectionless RPC	The PROFINET Context Manager provides an endpoint mapper in order to establish an application relationship.
Connection-oriented communication protocols					
FTP	21	TCP	Server/ incoming	File Transfer Protocol	FTP is used to transfer files to the flash card. Password protection via Web server user management. The port can be deactivated.
Telnet	23	TCP	Server/ incoming		Used for internal debugging. Password protection via Web server user management. The port can be deactivated.

HTTP Hypertext Transfer Protocol	80	TCP	Server/ incoming	Hypertext Transfer Protocol	HTTP is used for communication with the internal SIMOTION Web server. Password protection via Web server user management. The port can be deactivated. You can modify the port number if desired.
RFC1006	102	TCP	Server/ incoming Client/ outgoing	ISO-on-TCP protocol	The protocol extension RFC1006 is used for communication with ES, HMI, OPC server, etc.
SNMP Simple Network Management Protocol	161	UDP	Server/ incoming	Simple Network Management Protocol	SNMP enables the reading out of network management data (MIBs) by SNMP clients.
HTTPS Secure Hypertext Transfer Protocol	443	TCP	Server/ incoming	Secure Hypertext Transfer Protocol	HTTPS is used to communicate with the internal CU Web server via Secure Socket Layer (SSL). Password protection via Web server user management. The port can be deactivated. You can modify the port number if desired.
Internal protocol	3844	TCP	Server/ incoming	System diagnostics	Diagnostic access for traces within the system.
Internal protocol	5188	TCP	Server/ incoming	Project download	Communication with SCOUT for downloading project data.
Internal protocol	8412	UDP	Server/ incoming Client/ outgoing	Trace	Port is used for trace functionality that must be synchronized over several devices. Cannot be switched off.

Internal protocol	44550	UDP	Server/ incoming Client/ outgoing	Trace	Port is used for trace functionality that must be synchronized over several devices. Cannot be switched off.
Reserved	49152...65 535	TCP UDP	-	-	Dynamic port area that is used for the active connection endpoint if the application does not specify the local port number.

Documentation of Ethernet services used

Deactivating PN interface ports in SIMOTION

Introduction

On SIMOTION CPUs as of Version 4.4, PN interface ports can be set to **Disable** in the engineering system. This prevents devices being connected without permission and also increases security with regard to accessing the system. The engineering system and the PN stack ensure that at least one port on each interface is not set to **Disable** to prevent the user locking him/herself out. The default setting is **Automatic settings**.

How to deactivate ports

1. In HW Config, double-click in the station on the port that you want to deactivate.
2. In the dialog box that opens, switch to the **Options** tab.
3. Under **Transmission media/Duplex**, select the item **disable**. The port is deactivated.

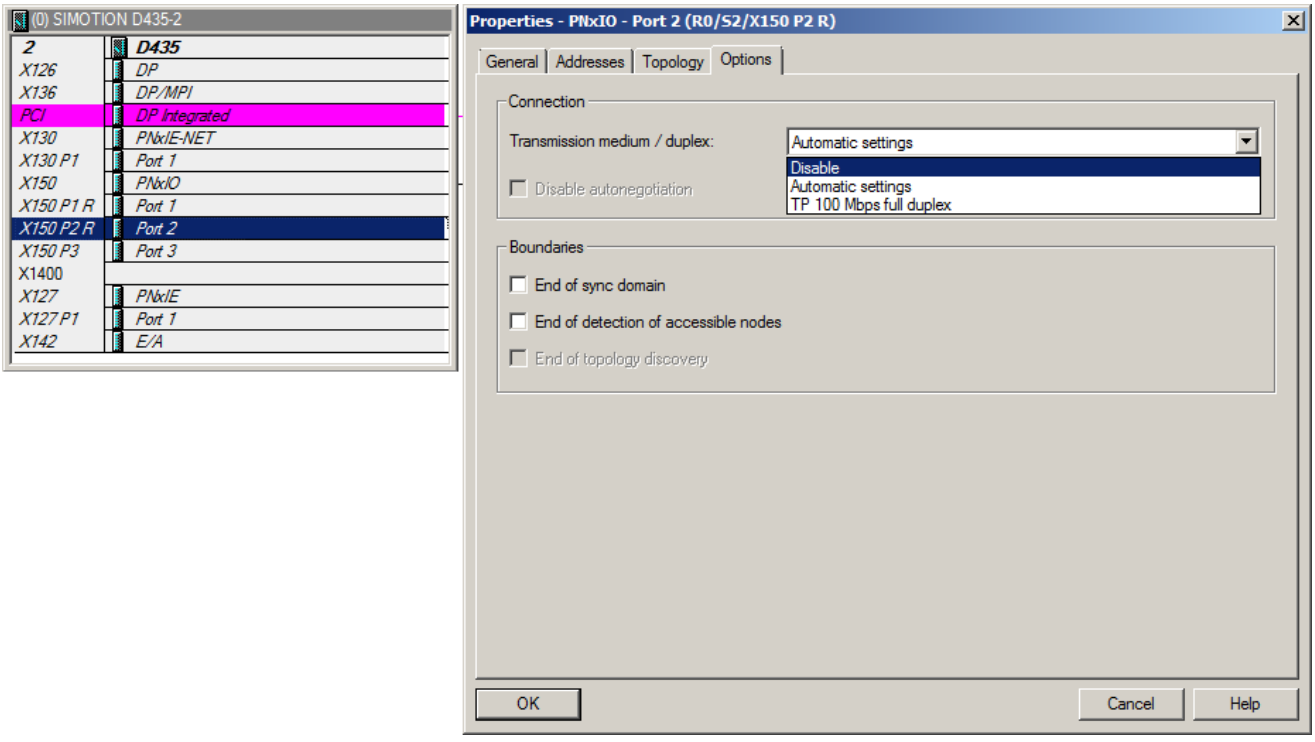


Figure 4-417 Deactivating PN ports on SIMOTION CPUs

4.5.7 Routing - communication across network boundaries

4.5.7.1 What does routing mean?

Introduction

Routing is the transfer of information from Network x to Network y.

There is a fundamental difference between intelligent, self-learning routing (e.g. IP routing in the Internet) and routing according to previously specified routing tables (e.g. S7 routing).

IP routing

IP routing is a self-learning routing procedure (which can also be performed manually), used exclusively in Ethernet communication networks which operate with the IP protocol, such as the Internet.

The function is performed by special routers that pass on the information to adjacent networks based on the IP address, when the IP address is not detected in the own network.

Note

IP routing is NOT supported by SIMOTION. Only S7 routing is possible between the interfaces of a SIMOTION controller.

S7 routing

S7 routing is a routing procedure based on previously configured routing tables, but which can also exchange information between different communication networks, e.g. between Ethernet, PROFIBUS and MPI. These routing tables can be created as interconnection tables in NetPro.

S7 routing does not work with the IP address, but with the so-called subnet IDs within the S7 protocol.

- Information transfer from Ethernet to MPI and vice versa
- Information transfer from Ethernet to PROFIBUS and vice versa
- Information transfer from MPI to PROFIBUS and vice versa
- Information transfer from Ethernet to Ethernet (SIMOTION V4.1.2 or higher, including PROFINET; CP343, CPU 315-2 PN/DP...)

CIDR Classless Inter-Domain Routing

Classless Inter-Domain Routing (CIDR) describes a process for a more efficient use of the existing 32-bit IP address space (IPv4). It was introduced to reduce the size of routing tables and utilize the available address areas more effectively. The CIDR (Classless Inter-Domain Routing) function includes subnetting and supernetting.

Subnetting refers to the splitting of an IP subnet into multiple subnetworks (by enlarging the subnet mask).

Supernetting refers to the compilation of multiple subnets into a joint IP subnet (by minimizing the subnet mask).

See also Relationships between subnet masks and IP addresses in relation to subnetting and supernetting (Classless Inter Domain Routing CIDR) (https://support.industry.siemens.com/cs/ww/en/view/2073614).

Note

CIDR is supported by all SIMOTION devices. In the case of SIMATIC devices, please refer to the notes in the manuals.

PG / PC assignment

Modification of the PG assignment may be required for S7 routing. You can do this now in the toolbar in SIMOTION SCOUT above the **Assign PG** button. This calls the properties window for PG assignment, where you modify the assignment and "activate" it (S7ONLINE access).

Note

You will find further notes on Ethernet/PROFINET and the settings required for routing in the *SIMOTION SCOUT Overview of Service and Diagnostics Options* product information as well as in the online help on this topic.

4.5.7.2 Configuration of S7 routing

S7 routing is configured in STEP 7 / SIMOTION SCOUT with the aid of the "NetPro" network configuration.

All stations contained in the network configuration can exchange information between one another. Connection tables must be created in NetPro for this purpose. The required routing tables are automatically generated during the compilation of the project, but must then be loaded to all the participating stations.

Establishing an online connection to the nodes in the project

Through S7 routing you can go to all the nodes in the project online.

This is how you establish an online connection from your PG/PC to the nodes in the project.

1. Insert a PG/PC station in NetPro.
2. Add a new interface.
3. Assign the new interface of the network card used to the PG/PC.
4. Make the assignment "active" (S7ONLINE access).
5. Compile NetPro and load the data into the routing devices by downloading it.

4.5.7.3 Routing for SIMOTION

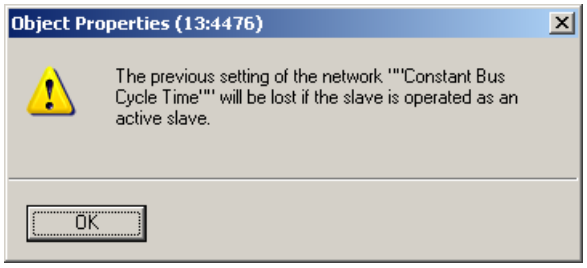
Definition

With routing you can, for example, access devices connected to subnets ONLINE via a PG/PC. S7 routing is supported by SIMOTION, i.e. information (engineering accesses) can be routed by a SIMOTION device from higher-level networks such as Ethernet and MPI to lower-level networks such as PROFIBUS or PROFINET/Ethernet (4.1.2 and higher).

Boundary conditions

The following boundary conditions must be taken into account in the "DP slave" mode when routing information on an isochronously operated PROFIBUS.

- The functions "Equidistant bus cycle" (requirement for isochronous applications) and "Active station" (requirement for routing to a lower-level network segment) mutually exclude each other.
- It is not possible to operate an active I slave on the isochronous bus.



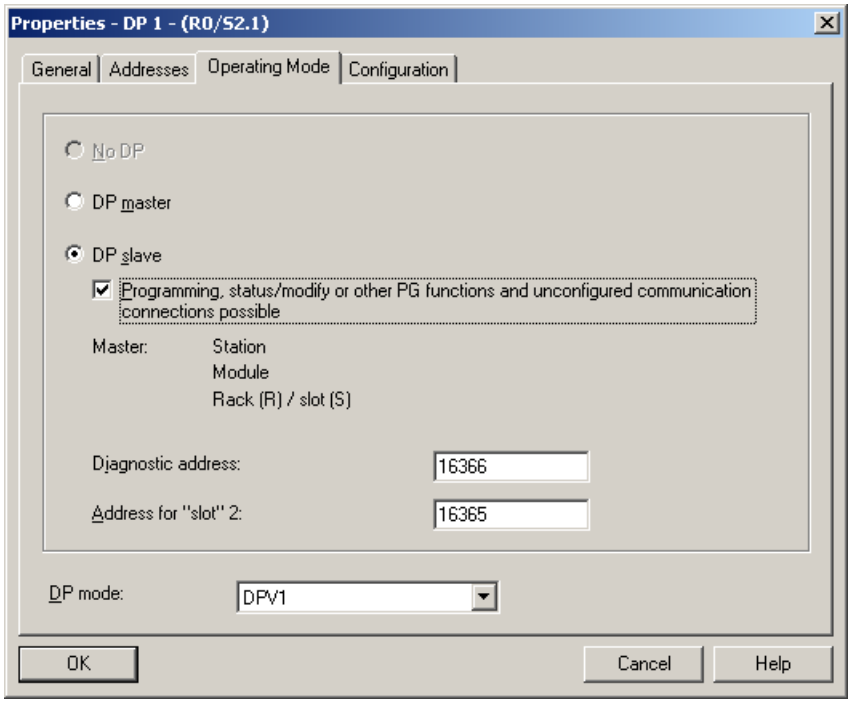


Figure 4-418 DP slave mode: Active station: Testing, commissioning, routing

The "Programming, status/control or other PG functions ..." check box must be activated if, for example, you frequently want to perform PG functions required for commissioning and testing via this interface, or if you want to access (S7 routing) SINAMICS drives on the cascaded, lower-level DP master interface of the SIMOTION controller with PG functions (e.g. STARTER).

If the "Programming, Status/Force or other PG functions..." option is activated, the interface becomes the active node on the PROFIBUS (i.e. the interface participates in the token rotation of the routing PROFIBUS). The following functions are then possible:

- Programming (e.g. loading)
- Test (status/control)
- S7 routing (I-slave as gateway)

The bus cycle time can be prolonged. Therefore, this option should not be activated for time-critical applications and when S7 routing and the client functionality are not required for the communication.

Note

When the "Programming, Status/Force or other PG functions ..." check box is not activated, the interface only operates as a server for cyclic data, i.e. S7 routing is not possible.

Note

If more than one gateway is used for the PG connection, all configured connection paths must also be physically connected. However, it is not possible to configure which of the configured connection paths is actually being used.

4.5.7.4 Routing with SIMOTION D (example of D4x5 with CBE30)

Routing between the different interfaces

The two standard Ethernet interfaces X120 and X130 of the SIMOTION D each form a separate subnet, all ports on the CBE30 also form a common subnet.

- S7 routing is possible to every interface.
  E. g. you can set up a connection from a PROFINET/Ethernet subnet to a PROFIBUS via S7 routing.
- IP routing from subnet to subnet is not supported. You can use an external IP router for this.

There are three options for connecting a PG/PC or HMI via S7 routing to a SIMOTION D with CBE30.

Note

The Ethernet interfaces X120 and X130 and the CBE30 must be in different subnets for S7 routing.

Engineering system to PROFINET (CBE30)

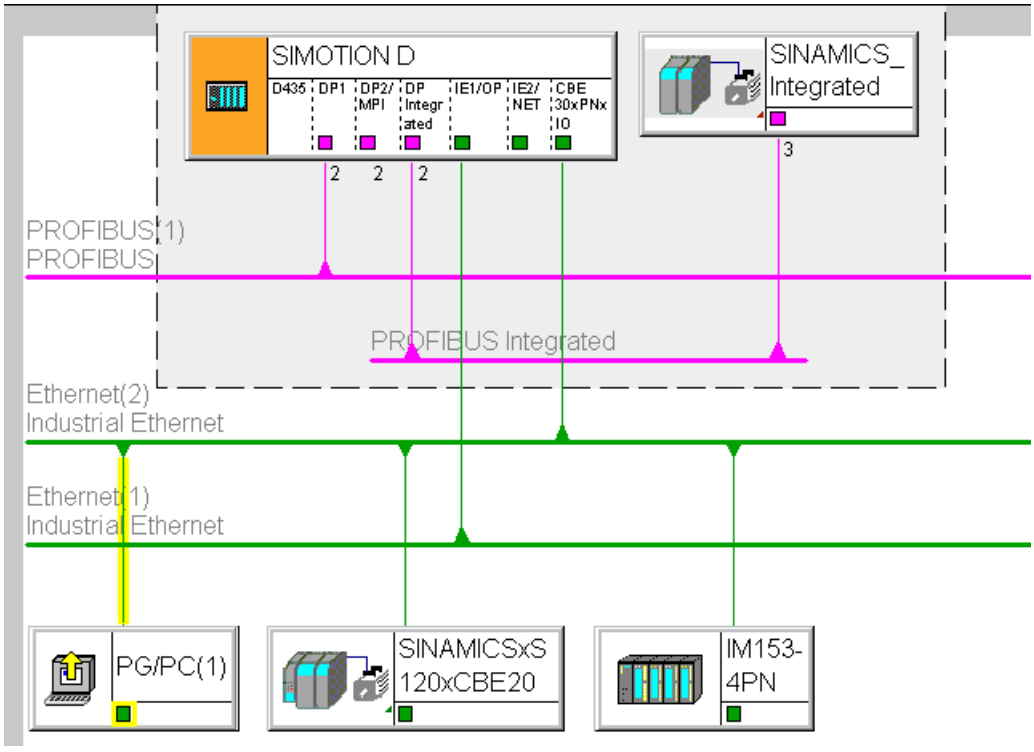


Figure 4-419 Example for PG/PC to CBE30

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the standard Ethernet interfaces ET1/ET2 (X120, X130) (V4.1.2 and higher)
- Access to the components on the same subnet (CBE30) via the switch functionality

Engineering system / HMI to PROFIBUS

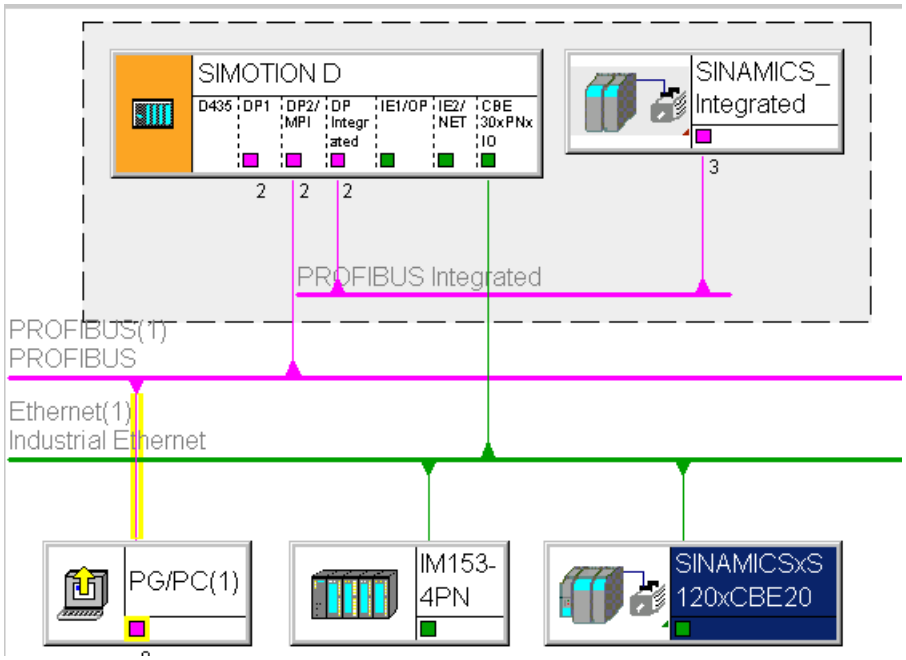


Figure 4-420 Example for PG/PC to PROFIBUS

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to X1400 on the CBE30
- S7 routing to the standard Ethernet interfaces (X120, X130) (V4.1.2 and higher)
- Access to nodes on the same network, e.g. HMI

Engineering system / HMI to Ethernet

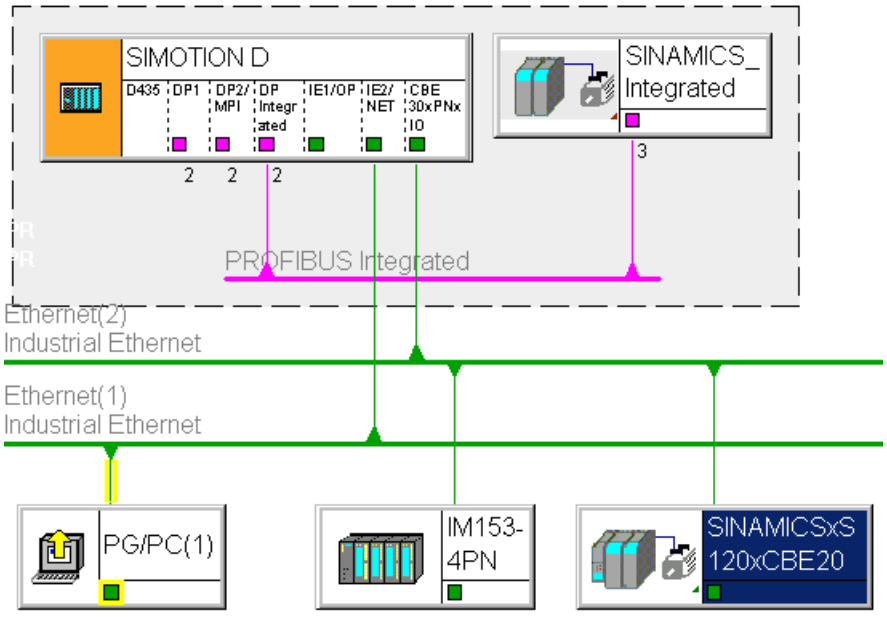


Figure 4-421 Example for PG/PC to Ethernet X120, X130

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to X1400 on the CBE30
- S7 routing between the Ethernet interfaces
- Access to nodes on the same network, e.g. HMI

4.5.7.5 Routing with SIMOTION D4x5-2 (example of D455-2 DP/PN)

Routing between the different interfaces

The two Ethernet interfaces of the D4x5-2 DP/PN (X127 P1 or X130 P1) each form a separate subnet.

The D4x5-2 DP/PN onboard PROFINET IO interface (X150, P1-P3) also forms a separate subnet. All ports of a PROFINET IO interface always belong to the same subnet.

- S7 routing is possible to every interface.
  E. g. you can set up a connection from a PROFINET/Ethernet subnet to a PROFIBUS via S7 routing.
- IP routing from subnet to subnet is not supported. You can use an external IP router for this.

There are therefore the following options for connecting a PG/PC or HMI device to a SIMOTION D using S7 routing.

Engineering system / HMI to PROFINET

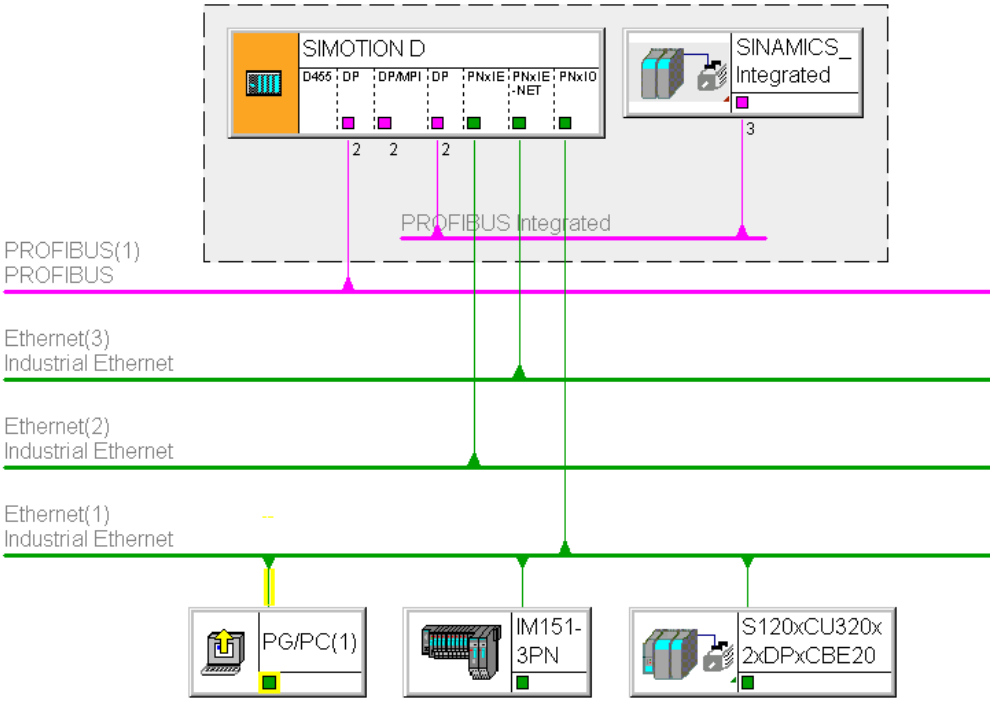


Figure 4-422 Example of PG/PC to PROFINET interface (PNxIO, X150)

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1)
- Access to the components on the same subnet via the switch functionality of the PROFINET IO interface

Engineering system / HMI to PROFIBUS

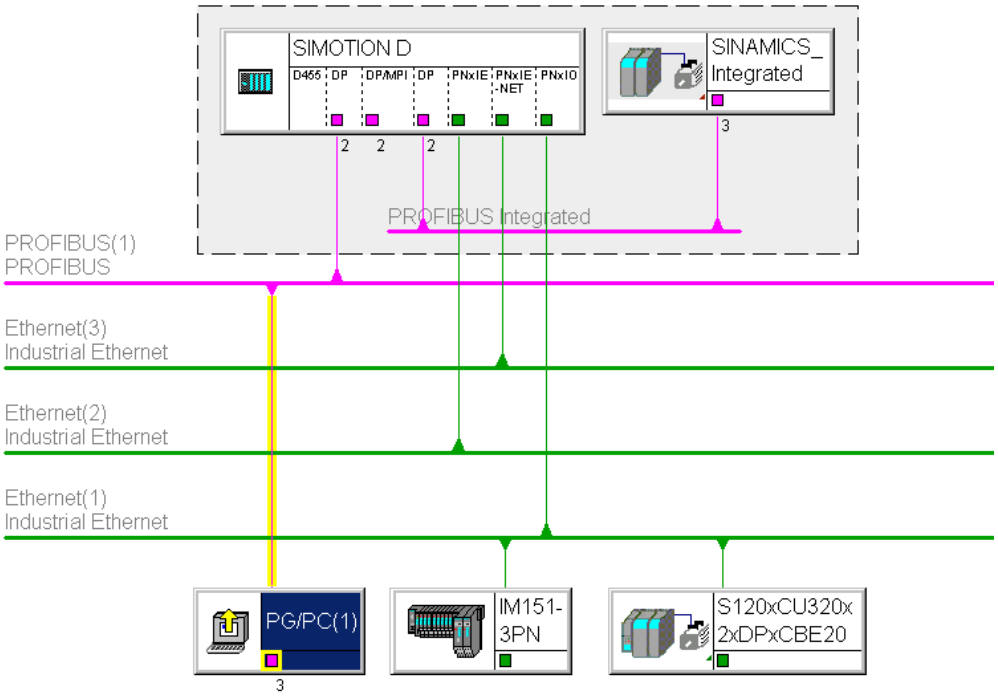


Figure 4-423 Example of PG/PC to PROFIBUS interface (DP, X126)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)
- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1)

Engineering system / HMI to Ethernet

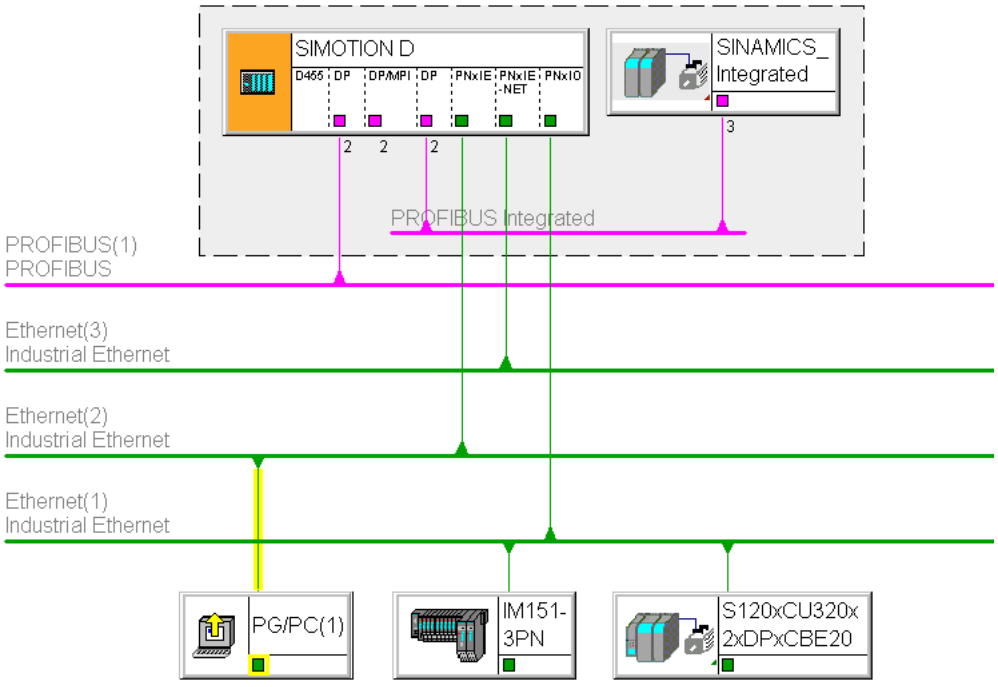


Figure 4-424 Example of PG/PC to Ethernet interface (PNxIE, X127)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the PROFINET interface (CBE30)
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)
- S7 routing between the Ethernet interfaces

4.5.7.6 Routing for SIMOTION D to the SINAMICS integrated

All SIMOTION D controllers have a SINAMICS drive integrated: the so-called SINAMICS Integrated. As this is connected internally to the SIMOTION controller via PROFIBUS DP, S7 routing is imperative here to be able to access it. Here the telegrams have to be routed from the external interfaces of the SIMOTION controller to the internal PROFIBUS DP. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.

4.5.7.7 Routing for SIMOTION P350

Description

S7 routing is possible:

- From PROFIBUS (IsoPROFIBUS board) on PROFINET subnet to MCI-PN board
- From PROFINET subnet to MCI-PN board on PROFIBUS (IsoPROFIBUS board)
- From SCOUT on SIMOTION P via softbus through the runtime on PN devices to the MCI-PN board and IsoPROFIBUS board
- From onboard Ethernet interfaces on PROFIBUS (IsoPROFIBUS board) and on PROFINET
- S7 routing between the Ethernet interfaces

IP routing is **not** possible via the P350 Ethernet interfaces.

Routing from PROFIBUS to PROFINET

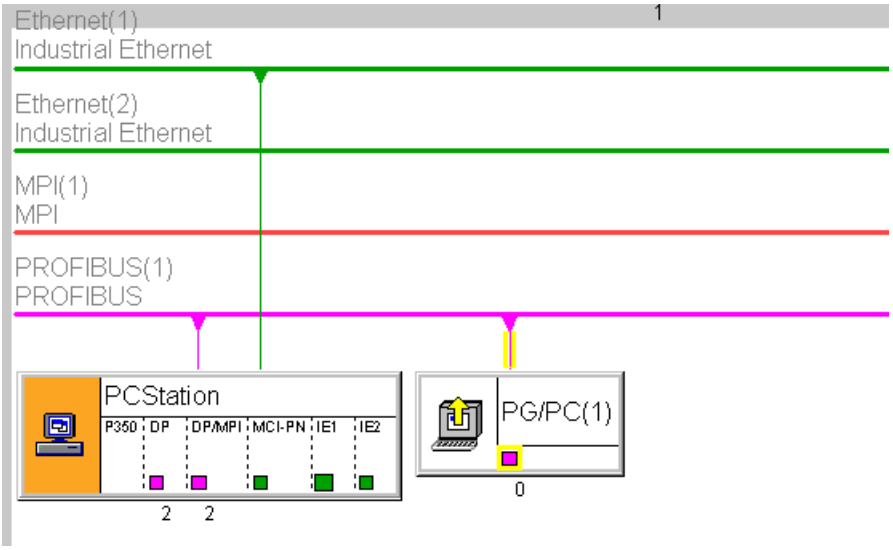


Figure 4-425 Example for P350 routing from PROFIBUS to PROFINET

Routing from PROFINET on PROFIBUS

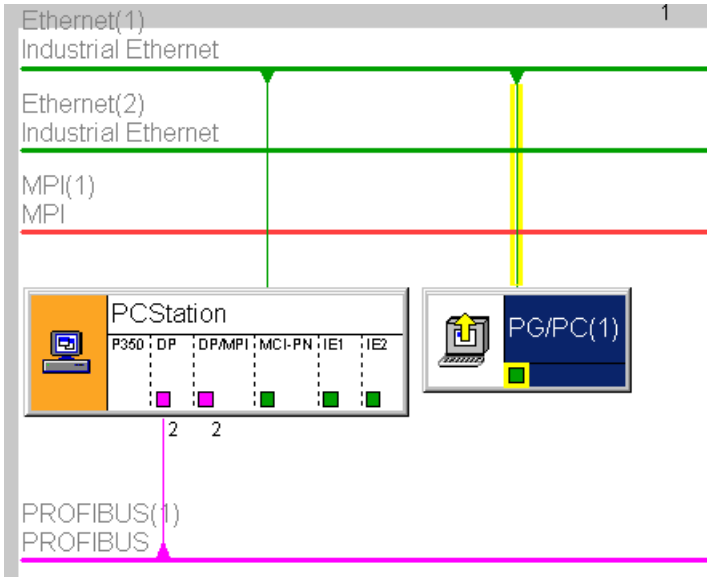


Figure 4-426 Example for P350 routing from PROFINET to PROFIBUS

4.5.7.8 Routing for SIMOTION P320

Description

S7 routing is possible:

- From the onboard Ethernet interface to the PROFINET subnet and the drive units or SIMOTION devices on the PROFINET subnet

Routing from Ethernet to PROFINET

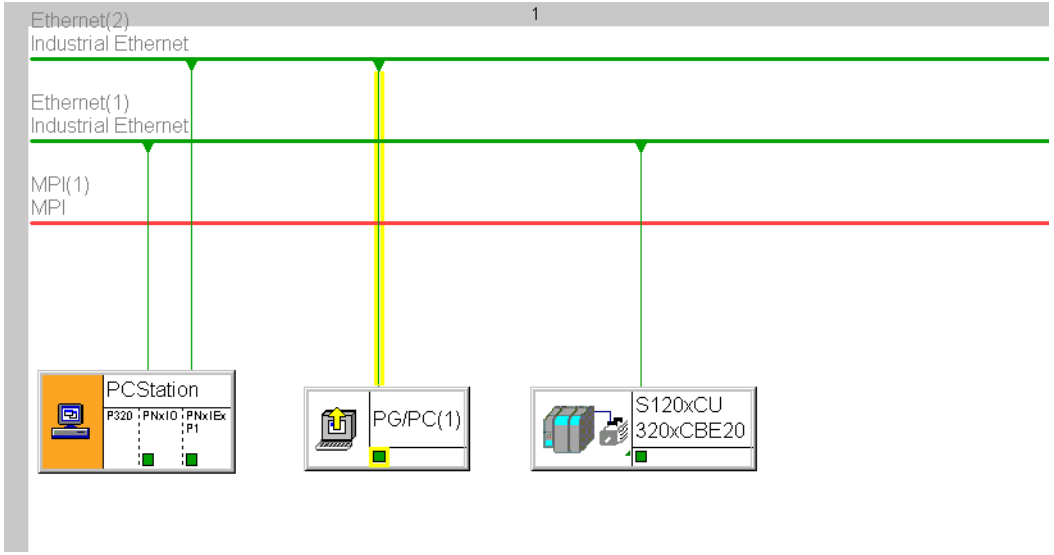


Figure 4-427 Routing for SIMOTION P320

4.5.8 SIMOTION IT

4.5.8.1 SIMOTION IT - overview

Description

SIMOTION IT provides the option of accessing this via standard web services (HTTP) and the integrated web server of the SIMOTION controllers.

Note

Appropriate protective measures (among other things, IT security, e.g. network segmentation) are to be taken in order to ensure safe operation of the system. You can find more information on Industrial Security on the Internet at:

www.siemens.de/industrialsecurity

This provides the following advantages.

- Location-independent open diagnosis / process monitoring
- Client device independent of the operating system (Windows, Linux, ...)
- Standardized communication interface for manufacturer-specific tools
- Independent of engineering system
- No version conflict between client application and SIMOTION RT (runtime)
- Series commissioning without engineering system

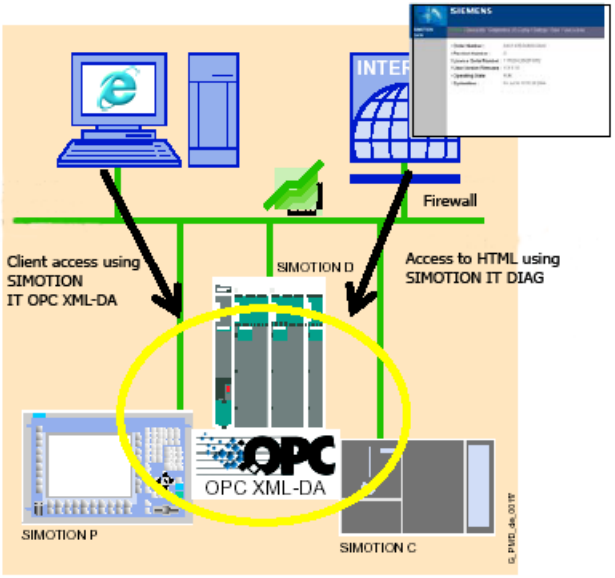


Figure 4-428 SIMOTION IT overview

SIMOTION IT provides the following service and diagnostics options via Internet technology:

- SIMOTION IT web server
- SIMOTION IT OPC XML DA
- Trace via SOAP
- File download using FTP (File Transfer Protocol)
- SIMOTION IT Virtual Machine

Further references

A detailed description of the SIMOTION IT products can be found in the *SIMOTION IT Ethernet-based HMI and Diagnostic Functions* product information on the SIMOTION SCOUT Documentation DVD.

Further information is available in the following manuals:

- *SIMOTION IT Diagnosis and Configuration, Diagnostics Manual*
- *SIMOTION – Jamaica*, Diagnostics Manual (Information on SIMOTION VM)

See also

Web access to SIMOTION (Page 2326)

SIMOTION IT web server (Page 2326)

SIMOTION IT OPC XML DA (Page 2329)

4.5.8.2 Web access to SIMOTION

Description

The following figure shows the various possibilities to access the data in a SIMOTION module.

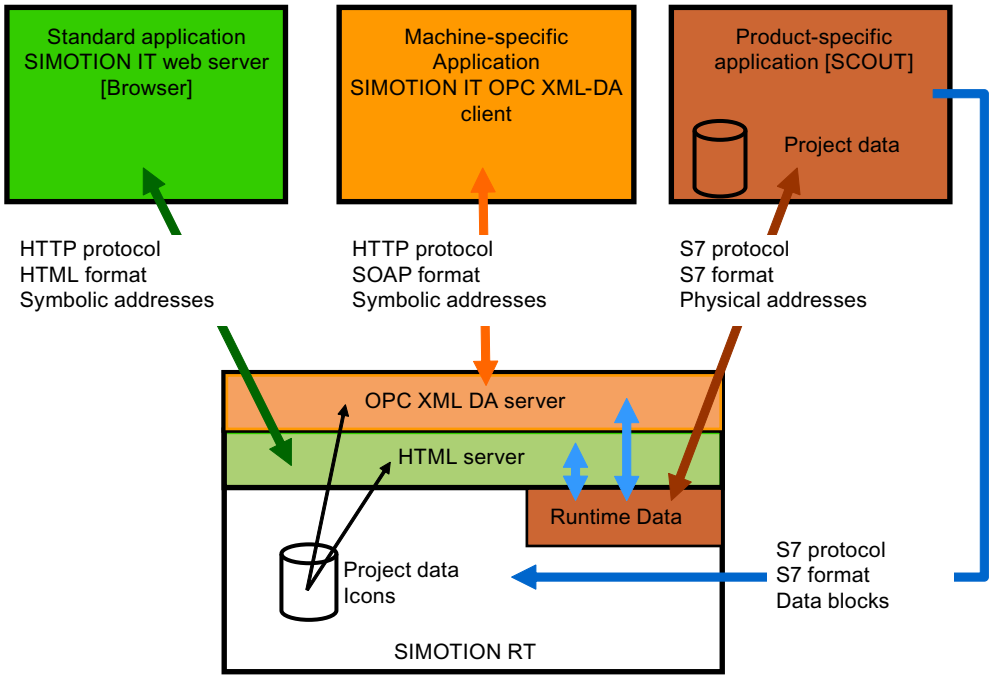


Figure 4-429 Access to SIMOTION

See also

SIMOTION IT web server (Page 2326)

SIMOTION IT OPC XML DA (Page 2329)

4.5.8.3 SIMOTION IT web server

Description

The SIMOTION IT web server allows a PC to use any Internet browser to access the the web pages of the integrated web server of the SIMOTION controllers.

You will find a detailed description in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Standard diagnostic pages

SIMOTION provides the following standard diagnostic pages:

- **Start page**
- **Device Info/IP-Config**
  (information about the firmware, devices, device components, technology packages, and data of the SIMOTION device Ethernet interface)
- **Diagnostics**
  (CPU utilization, memory use, operating mode, display of task runtimes, trace for devices and system, service overview)
- **Message&Logs**
  (diagnostics buffer, SIMOTION alarms and Drives, syslog and userlog)
- **Machine Overview**
  (modules and topology of a machine as well as hardware configuration information)
- **Manage Config**
  (loading SIMOTION IT web server configurations, device updates, saving device data)
- **Settings**
  (setting the time zone, switching operating modes, changing the display of user-defined pages)
- **Files**
  (accessing the SIMOTION file system, uploading and downloading files, creating folders, and storing additional data, e.g. documentation)

Simplified standard pages

To enable the best possible display of HTML pages on devices such as cell phones or PDAs, a set of special pages is provided for version 4.1.3 and higher. These contain a simplified representation of information from the standard pages. The start page of the simplified standard pages can be reached at the address http://<IPAddr>/BASIC.

Configuration via WebCfg.xml

There are two files which can be used to configure SIMOTION IT web server:

- WebCfg.xml
- WebCfgFrame.xml

The WebCfg.xml configuration file is used to configure user-relevant settings in the web server. The WebCfgFrame.xml file contains the manufacturer's settings.

User-defined pages

The SIMOTION IT web server offers the option of integrating individually designed web pages. Two mechanisms are available for accessing SIMOTION variables:

- JavaScript libraries opcxml.js and appl.js
- MiniWeb Server Language (MWSL)

User-defined pages can be displayed in the "User's Area". For this purpose, they must be stored in the FILES folder of the SIMOTION file system.

Trace via SOAP

The WebTrace is almost identical to the SIMOTION SCOUT trace. The only difference is in output of the data format and in the number of signals that can be parameterized.

The "Trace via SOAP" function package enables SIMOTION variables to be written to a buffer. The values are packed in a file and can be retrieved asynchronously via an HTTP request. This interface can only be used by client applications.

The trace can be parameterized via a web interface and the recording viewed immediately using a TraceViewer.

Variable access

The variable access for the SIMOTION IT applications is implemented using a variable provider. The following variable providers currently exist.

- MiniWeb
- SIMOTION
- SIMOTION diagnostics
- UserConfig
- IT Diag

These variable providers allow access to the following variables:

- Device system variables
- TO system variables
- Global unit variables from the interface section
- TO configuration data
- Also global device variables and I/O variables as of 4.2
- Drive parameters
- Setting of the operating mode, execute RamToRom, execute ActiveToRom
- Technological alarms
- Diagnostics buffer

Secure HTTPS connection

The Secure Socket Layer protocol (SSL) enables encrypted data transmission between a client and the SIMOTION device. The Secure Socket Layer protocol forms the basis for HTTPS access. Encrypted access can take place via both SIMOTION IT OPC XML DA and the SIMOTION IT web server.

4.5.8.4 SIMOTION IT OPC XML DA

Description

The SIMOTION IT OPC XML DA server enables access via Ethernet to data and operating modes of the SIMOTION controller. As of V4.2, a separate license is no longer required for SIMOTION IT OPC XML DA.

OPC stands for open connectivity and refers to standardized software interfaces which enables the exchange of data between applications from different manufacturers in automation technology. With OPC XML DA, it is possible to communicate with a control using Ethernet-based standard message frames. Commands are transmitted via the SOAP (Simple Object Access Protocol) communication protocol.

A customer-specific application created on a client PC, which, for example, is programmed with the C#, Visual Basic, or Java programming language, uses the SIMOTION IT OPC XML DA services and properties:

- Open communication using HTTP, SOAP, OPC XML between client and SIMOTION device
- Uses the OPC XML DA 1.0 specification of the OPC Foundation
- Access to SIMOTION variables
  - Reading/writing
  - Cyclical reading of so-called "subscriptions"
  - Browsing
- Trace using SOAP; this function is an extension of the OPC specification
- Clients on any hardware with various operating systems (Windows, Linux, etc.)
- Creating client applications using C#, Java, C++. You must implement the application that you want to access on the SIMOTION OPC XML DA server yourself.
- Access protection with user groups, user ID, and password

The following figure is a schematic representation of access to the OPC XML DA server.

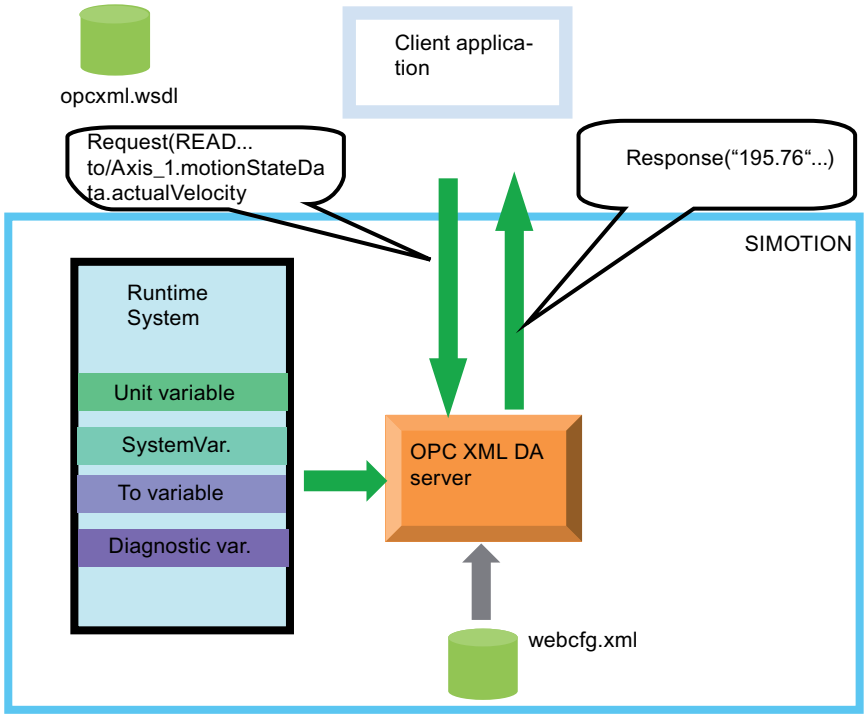


Figure 4-430 Access to the OPC XML DA server

4.5.9 PROFIsafe

4.5.9.1 Communication relationships for drive-based safety

Description

The drive-based safety functions in the drive can be controlled either by using safe terminals directly on the drive or from a fail-safe control (F control) via PROFIBUS/PROFINET.

The control signals for the drive-based safety functions, as well as the feedback relating to the safety function status, are safety-oriented and must be transmitted via a communication channel that is secured by means of the PROFIsafe protocol. The figure below contains a diagram providing a general overview of how interaction between the various control and drive processes works, as well as the communication relationships between them that are required for this purpose.

Signal flow for selecting and deselecting the drive-based safety functions and their signaling to the drive control process

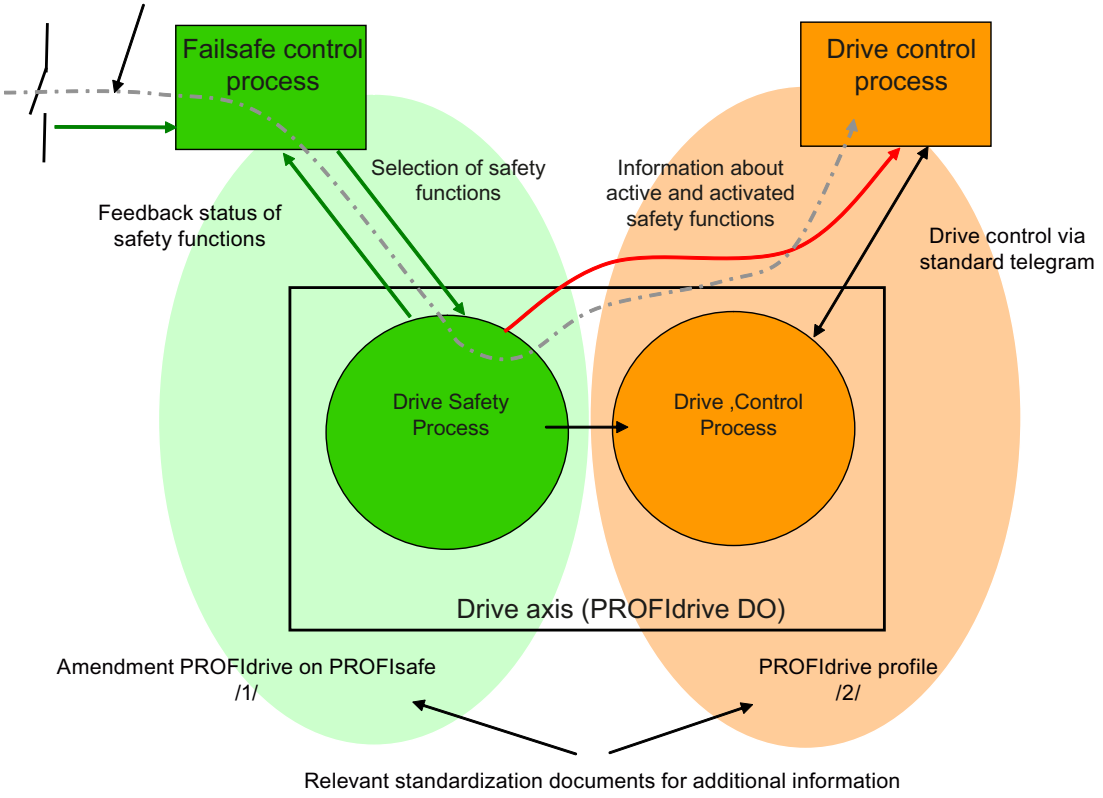


Figure 4-431 Communication relationships for drive-based safety

The "drive safety process" interfaces to the F control and drive control are PROFIdrive interfaces; their functions are defined in /1/ and /2/.

The F control introduces and monitors a drive-based safety function via the PROFIsafe-secured transmission channel between the F control and the drive (drive axis). The respective statuses of the drive-based safety functions in the drive also have repercussions on the drive interface to the drive control, since the drive priority switches between the drive control and drive safety process in the case of some drive-based safety functions.

In order for the F control and drive control to be coordinated effectively, therefore, an information channel from the "drive safety process" to the drive control is also required so that the drive control can respond to the required or activated drive-based safety functions accordingly.

4.5.9.2 Message frames and signals in drive-based safety

Description

Since SIMOTION does not have a safe logic function, it cannot attend to the F control process. Instead, this is carried out using a second controller featuring F functionality (usually a SIMATIC F-CPU). The image below shows how this setup works, using the example of a SINAMICS axis. A PROFIsafe-secured communication channel runs between the DO drive and the SIMATIC F-CPU. Standard telegram 30, consisting of a safety control word and status word (among other things), is normally available for this communication channel.

As of V4.3, the following PROFIsafe telegrams are available:

- Telegram 30
- Telegram 31
- Telegram 901
- Telegram 902 (from V4.4, can be used in SCOUT TIA in connection with the TIA Portal)
- Telegram 903 (as of V5.2)

With the user program on the F-CPU, the safety control word is used to select or deselect the configured drive-based safety functions in the drive (drive safety process). The feedback from the active safety functions is sent in the input data to the F-CPU, via the safety status word. A safe process in the drive is used to send the feedback from the active safety functions via a secured communication channel; therefore, the feedback may be used for activating protection zones and doors via the F-CPU.

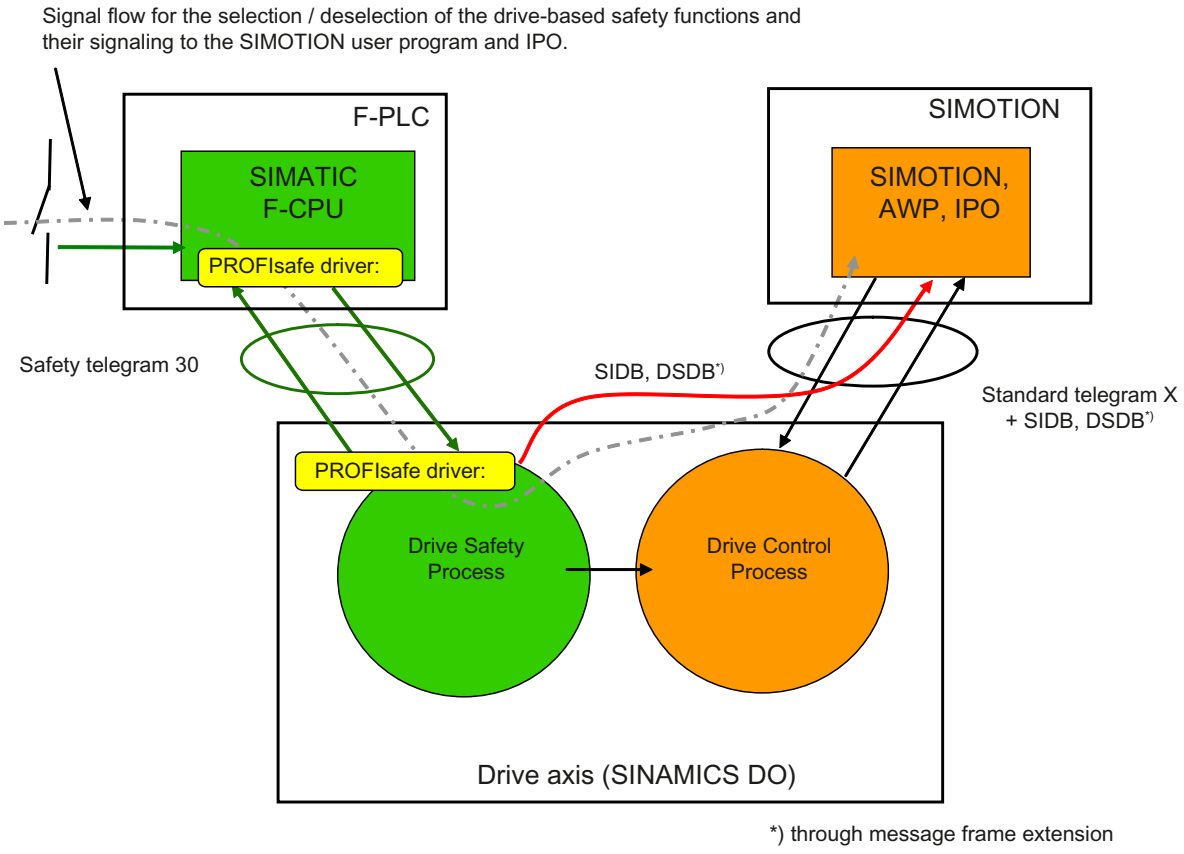


Figure 4-432 Messages and signals in drive-based safety

Safety information channel SIDB (up to V4.4) and DSDB (from SIMOTION V4.4)

In addition to controlling safety functions in the drive via the PROFIsafe channel, you have the option of transferring states and activation states of the drive-based safety functions from the drive to SIMOTION via a safety information channel (SIDB Safety Data Block). This has been expanded through the new standard DSDB (Drive Safety Data Blocks) from SIMOTION V4.4. The previous SIDB telegram extension block is replaced by two telegram extension blocks, DSDB1 and DSDB2. Instead of the previous single bico interconnections to bits in SINAMICS, interconnections are made to complete signals (bit blocks) with the new SC types. As a result, status information also becomes independent of the activated safety function (basic or extended functions). With DSDB the entire safety-oriented functionality configured on the drive is available on the TO. Configuration is automatic and uses the DSDB channel (Drive Safety Data Blocks).

Note

The Drive Safety Data Block is an extension of the PROFIdrive axis telegrams. Transfer according to PROFIdrive and is therefore not a secure safety telegram.

Note

Further information about SINAMICS Safety Integrated and the configuring with the TO axis can be found in Chapter **Support of the SINAMICS Safety Integrated functions** of the *TO Axis Function Manual*.

See also

New in SIMOTION SCOUT as of V4.3 (Page 2183)

4.5.9.3 SIMOTION F proxy functions

Description

SIMOTION features integrated F-Proxy functionality for the purpose of PROFIsafe connection of integrated SIMOTION D drives, as well as SINAMICS drives that are controlled by SIMOTION but are in a different communication domain from the F-CPU, for example. The F proxy functionality enables transparent routing of safety message frames from the SIMOTION I slave or I device interface to the respective SIMOTION master or controller interface on which the drive is configured. Despite the SIMOTION routing function, PROFIsafe communication between the F-CPU and drive is secured, as the PROFIsafe drivers in the end points (F-CPU, drive) monitor communication securely.

In order to use F proxy functionality, the two paths of communication - from the F-CPU to SIMOTION and from SIMOTION to the drive - need to be configured separately.

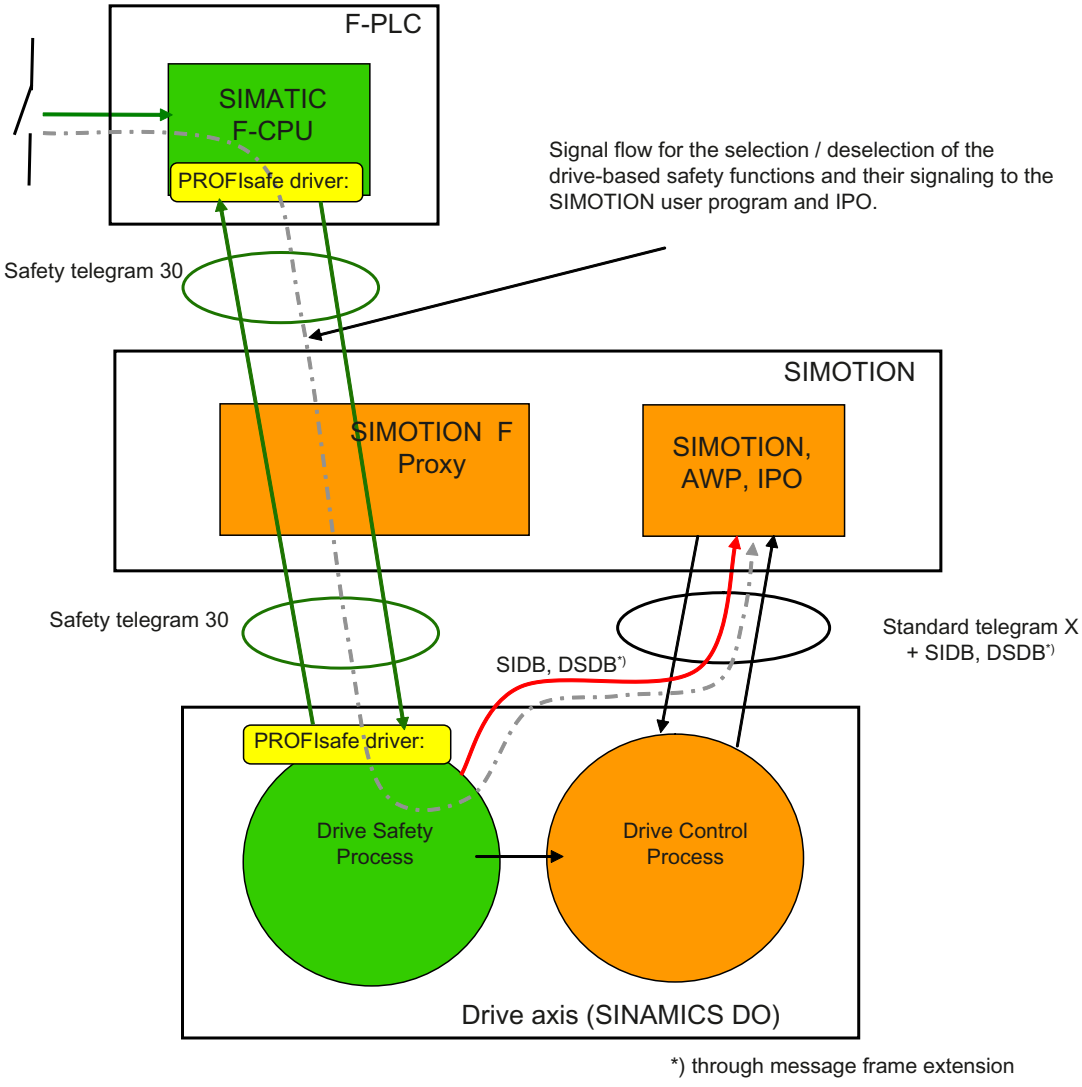


Figure 4-433 Routing the PROFIsafe channel with F proxy functionality

4.5.9.4 PROFIsafe properties for configuration

Mechanisms with PROFIsafe

Basically the PROFIsafe can communicate both via PROFINET and PROFIBUS. The I-device F-Proxy, the I-slave F-Proxy, the PROFINET shared device, and the PROFIBUS data exchange broadcast are available as mechanisms for PROFINET and PROFIBUS.

Note

Mixed operation is **not** supported for the PROFIsafe configuration. If, for example, in the case of a SINAMICS drive, the safety configuration for one axis is carried out via F-Proxy and those of the other axes are configured via shared device, this configuration will be rejected with a consistency error during compilation. The F-Proxy variants are recommended for the PROFIsafe configuration.

For all the mechanisms, the F data is managed by the F-CPU and the Motion Control data by the SIMOTION controller. The PROFIsafe communication properties required are set in the HW Config for the fail-safe parameters.

Note

For SIMOTION projects (RT version) < V4.2, secured PROFIsafe communication is only possible via PROFIBUS. Therefore, with SIMOTION devices < V4.2 with PROFIBUS and PROFINET interfaces, you can only implement PROFIsafe via the PROFIBUS interface and the Motion Control tasks (for example) via the PROFINET interface. Mixed operation (Motion Control via PROFINET and PROFIsafe via PROFIBUS) is, however, only possible with the integrated SIMOTION D drives. It is not permissible if you are using external drives (e.g.S120 CU320-2) on a SIMOTION controller.

Overview of failsafe parameters for PROFIsafe during configuration

To access the PROFIsafe dialog box, double-click the **PROFIsafe** entry in the drive rack in HW Config.

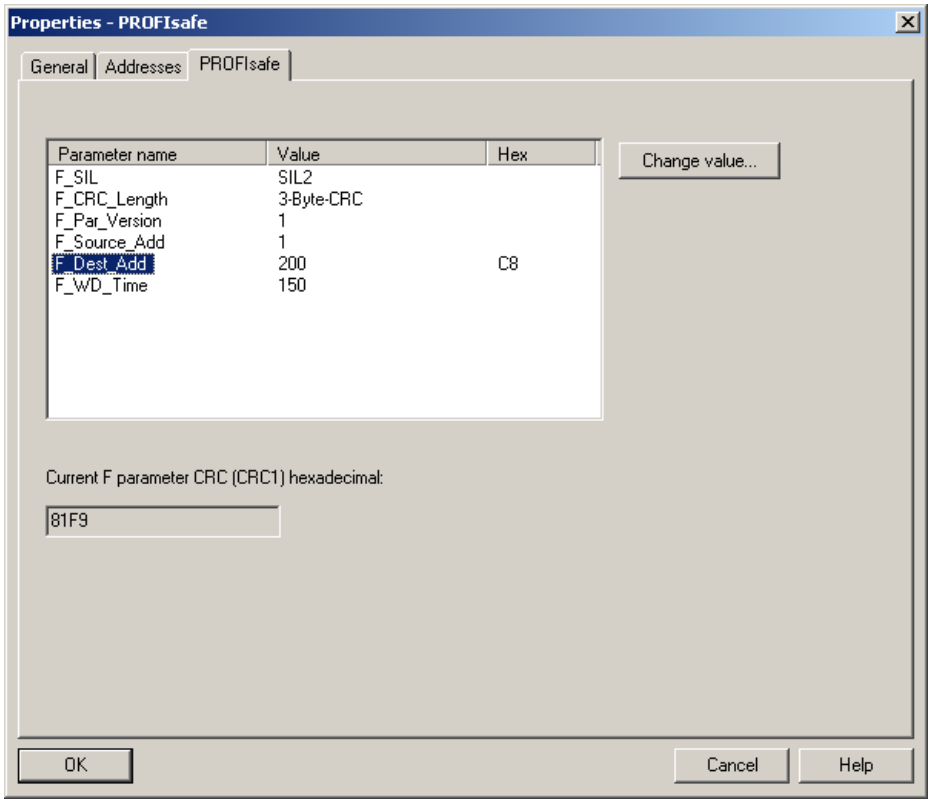


Figure 4-434 Properties of PROFIsafe using the example of I-device F-proxy

F_CRC_Length and F_Par_Version

Identifies not only the length of the failsafe useful data but also the PROFIsafe MODE.

- PROFIsafe V1-MODE
  F_CRC_Length = 2 to useful data length of 12 bytes
  F_CRC_Length = 3 from useful data length of 13 bytes
  F_Par_Version = 0
- PROFIsafe V2-MODE
  F_CRC_Length = 3 to useful data length of 12 bytes
  F_CRC_Length = 4 from useful data length of 13 bytes
  F_Par_Version = 1

Note

PROFIsafe V2 mode is not supported by all devices and/or firmware versions. Before configuration, find out which mode is available as of which version.

SINAMICS G120 CU240D DP F and SINAMICS G120 CU240S DP F only support V2 mode as of firmware version V3.2.

F_Dest_Add: 1-65534

F_Dest_Add determines the PROFIsafe destination address of the drive object.

Any value within the range is allowed, although it must be manually entered again in the Safety configuration of the drive in the SINAMICS drive unit and/or entered when configuring Safety Integrated (drive parameters p9610/p9810).

F_WD_Time: 10- 65535
A valid current safety telegram must be received from the F-CPU within the monitoring time. The drive will otherwise switch to the safe state.
The monitoring time should be of sufficient length to ensure not only that the communication functions tolerate telegram time delays, but also that the fault response is triggered as quickly as possible if a fault occurs (e.g. interruption of the communication connection).

For additional information on F Parameters, refer to the online help of the **PROFIsafe** dialog box.

4.5.9.5 PROFIsafe via PROFINET

Principles of I device failsafe proxy

Brief description

Using the I device F-Proxy you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFINET with SIMOTION devices (SIMOTION D, SIMOTION P, SIMOTION C) for the lower-level drives. The routing of cyclic PROFIsafe data to SINAMICS Integrated and SINAMICS drives on external PROFIBUS or PROFINET is therefore possible.

You can also use the functionality of a shared iDevice for the iDevice F Proxy from SIMOTION V4.4 and higher. The iDevice interface can be shared between an F CPU and a SIMOTION controller. This means that an iDevice can communicate with two higher-level controllers as an IO device. At the same time, higher-level controllers can access certain modules, e.g. an F-CPU can access the F-Proxy submodules, using the shared device. You can find a configuration example in the Shared iDevice section and PROFIsafe (Page 2374).

Note

F-Proxy modules can be set up either on the CBE30-2 or on the integrated PROFINET interface.

A failsafe host communicates with the drives via the I device interface and an F-Proxy of a SIMOTION CPU. These drives may be located on PROFIBUS DP external, PROFIBUS DP integrated and the PROFINET IO system of the SIMOTION CPU. The SIMOTION CPU's communication segments feature SINAMICS S120/S110, incl. SINAMICS Integrated/CX32/CX32-2 and G120.

The higher-level project with the F-CPUs is the master project. There may be several F-CPUs in the master project. The lower-level projects with SIMOTION CPUs are designated as subprojects. All F slaves of all segments (PROFIBUS/PROFINET) which originate from SIMOTION are mapped by the F-Proxy as n F submodules in an F-Proxy module.

During configuration, a failsafe I device is produced as a substitute of a SIMOTION device with drives and imported into the F-CPU master project.

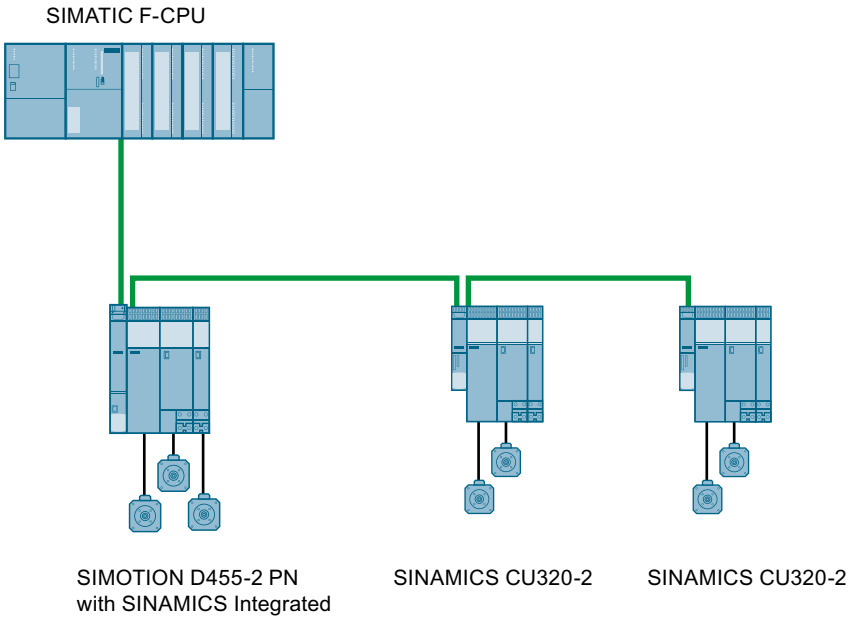


Figure 4-435 Example topology for F-CPU master with SIMOTION D and lower-level drives

Module/submodule structure

The submodules of the drive with the safety message frame are mapped to submodules of the PROFINET I device interface, regardless of whether the drive is connected to SIMOTION via PROFIBUS, integrated PROFIBUS, or PROFINET.

With an I device failsafe proxy, all failsafe proxy submodules of a SIMOTION device are depicted in one single module (module 2).

The diagram below shows an F-CPU as the master, to which a SIMOTION with integrated and lower-level drives is subordinate.

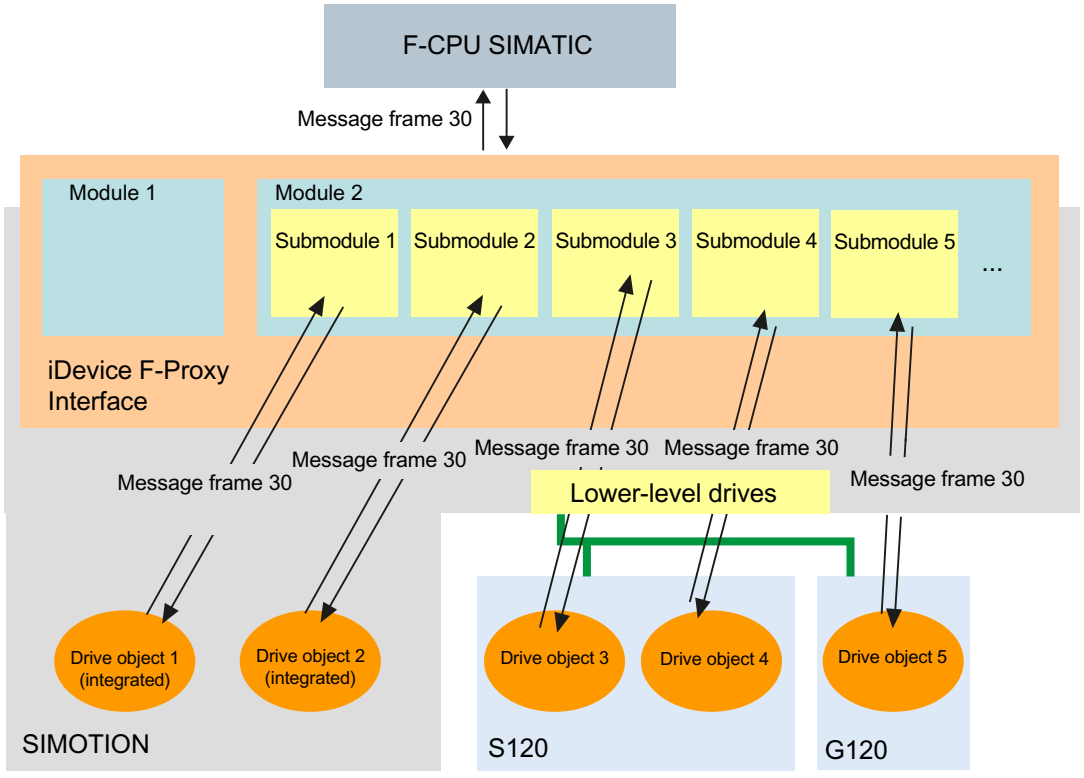


Figure 4-436 Master project with a sub-project as the I device failsafe proxy

See also

Use of Safety with MRRT (Page 2173)

Shared device via PROFINET (Page 2368)

Configuring I device failsafe proxy (Page 2348)

Supported devices and software requirements for I device failsafe proxy

Software requirement

- Step7 V5.5 or higher
- SINAMICS S120, V2.6 or higher
- SINAMICS G120, V3.2 or higher
- SIMOTION, V4.2 or higher
- S7 F configuration pack, V5.5 SP8 or higher (if using Safety/PROFIsafe)
- Valid from PROFIsafe V2

Supported devices

Drive units

- SINAMICS Integrated
- SINAMICS Controller Extension (CX32, CX32-2)
- SINAMICS devices on lower-level PROFIBUS
- SINAMICS devices on lower-level PROFINET

Devices with I-device F-proxy functionality

- SIMOTION D4xx, D4x5-2
- SIMOTION C240 PN
- SIMOTION P350-3, P320-3, P320-4
- F-CPU (List of supported F-CPUs (https://support.industry.siemens.com/cs/ww/en/view/44383954))

Table 4-243 The table shows which SIMOTION devices support the I-device F-Proxy and which SINAMICS devices can be accessed via the I-device F-Proxy.

Overview of devices capable of I-device F-proxy functions	
SIMOTION IO controller	
Controller Based	C240 PN
PC Based	P320-3 P350-3 P320-4
Drive Based (blocksize)	D410 PN D410-2 DP/PN
Drive Based (booksize)	D4x5 with CBE30 D445-1 with CBE30 D4x5-2 DP/PN D4x5-2 DP/PN with CBE30-2
SINAMICS IO devices and DP slaves	
S120 CX32	CX32 CX32-2
S120	CU320 DP CU320 CBE20 CU320-2 DP CU320-2 DP CBE20 CU310-2 DP CU310 DP CU310 PN CU310-2 PN CU320-2 PN

Overview of devices capable of I-device F-proxy functions	
S110	CU305 CU305 PN
G120	CU240S PN F CU240D PN F CU240E-2 CU240D-2 CU250D-2

I-Device F-Proxy quantity structure and properties

Specific properties of I-Device F-Proxy

Acyclic services on the I-Device F-Proxy

The current I-Device interface does not support acyclic data transfer. The I-Device F-Proxy submodules do not have parameter access (Parameter Access Point PAP) and cannot convey alarms.

If an alarm channel is to be used, the drive must be operated on PROFINET and directly incorporated in SIMOTION and the F-CPU using the Shared Device function. Here the F data of the F-CPU and the Motion Control data are managed by SIMOTION.

Supported PROFIsafe SINAMICS telegrams

Telegrams 30, 31, 901, 902 and 903 are supported.

Note

Telegram 902 is available from V4.4 and can only be used in SCOUT TIA in connection with the TIA Portal.

Isochronous mode

The F-Proxy submodules on the SIMOTION CPU are RT and can be operated non-isochronously.

Supported lower-level bus systems

Drives on PROFIBUS integrated, PROFIBUS external and PROFINET external are supported. All these bus interfaces can also be routed via the I-Device F-Proxy at the same time.

Supported I-Device interfaces

All PROFINET-capable SIMOTION devices are supported.

Two F-CPUs via shared I-Device

If the power of a single F-CPU does not suffice for the number of safety axes, the safety axes can be distributed via the shared I-Device functionality to two F-CPUs.

Note

Because all F-Proxy axes must be configured on the I-Device of a single interface, the use of the second PN interface as I-Device for an additional F-CPU is not possible.

F-Proxy on PN-IO CBE30-2

For the operation with two PN-IO interfaces, the PN-IO CBE30-2 interface is provided for the higher-level communication, whereas the PN-IO integrated interface is provided for lower-level communication with "local" drives or fast IO.

Only the PN-IO integrated can be used for Servo_fast. This means, you must use the PN-IO CBE30-2 for safety communication to a higher-level F-CPU.

Note

F-Proxy modules can be configured either on the PN-IO CBE30-2 or on the PN-IO integrated. It is not possible to configure F-Proxy modules concurrently on the PN-IO CBE30-2 and on the PN-IO integrated.

Response times - Transmission time for I-Device F-Proxy

If data is transferred from the F-CPU via the I-Device F-Proxy, this extends the runtime for the SINAMICS drives via the I-Device F-Proxy by a maximum of 2 servo cycles per send direction. Use the servo cycle of the lower-level system as the servo cycle.

Supported number of F-Proxy submodules

A maximum of 128 I-Device submodules are supported. Of these, up to 64 submodules can be used for Safety by default. The other 64 submodules are available for standard IOs. The maximum number of 64 F-Proxy submodules can be configured **independently** of the PROFIsafe telegram for all SIMOTION CPUs.

Up to 128 F-Proxy submodules are supported (as of SIMOTION V4.5)

As of firmware version 4.5, a maximum of 128 F-Proxy submodules, and thus a maximum of 128 safety axes, can be configured for the SIMOTION D455-2. The maximum number of F-Proxy submodules **depends** on the configured PROFIsafe telegram.

In the following tables, the input/output data as well as the maximum number of F-Proxy submodules are shown depending on the PROFIsafe telegram and the SIMOTION CPU.

SINAMICS PROFIsafe telegrams

PROFIsafe telegram	F-Proxy submodules SIMOTION D455-2	F-Proxy submodules SIMOTION CPU (other)	Single telegram Input	Single telegram Output	Single telegram Total
30	128	64	6	6	12
31	128	64	8	8	16
901	89	64	14	10	24
902[1]	79	64	16	10	26
903	89	64	14	10	24

1) Can only be used in SCOUT TIA in connection with the TIA Portal.

Configuring more than 64 F-Proxy submodules

The following must be taken into account when configuring more than 64 F-Proxy submodules:

- A SIMOTION I-Device supports a maximum of 128 subslots. If, for example, 128 subslots are assigned to F-Proxy axes, no further user data can be configured on the I-Device interface (see table above).
- A SIMOTION I-Device supports a maximum of 1440 bytes of user data (including IOPS/CS user data qualifier). Depending on the selected Safety telegram and the number of F-Proxy axes, only limited (or no) space is available for user data in the I-Device.

PROFIsafe via PROFINET with an F-CPU

Description

A detailed description of how you can configure PROFIsafe with PROFINET in conjunction with a SIMATIC F-CPU can be found described in an FAQ.

See Actuating internal drive safety functions via SIMOTION and PROFINET with PROFIsafe (https://support.industry.siemens.com/cs/ww/en/view/50207350).

See also

Use of Safety with MRRT (Page 2173)

Topology overviews I device F-Proxy

Topology for I device failsafe proxy for PROFIBUS drive units

Example of topology

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP.

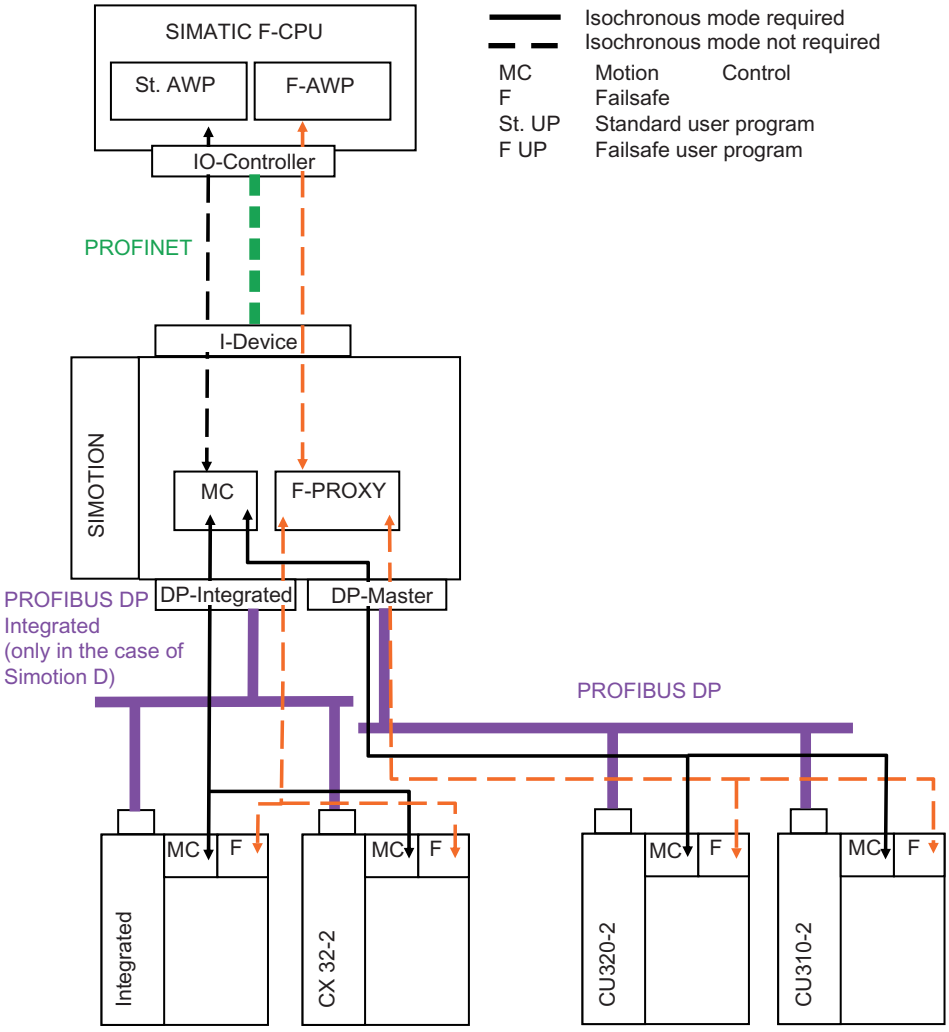


Figure 4-437 Topology for I device failsafe proxy for PROFIBUS drive units

Topology for I device failsafe proxy for PROFINET drive units

Example of topology

The diagram below shows a topology in schematic form in which the Safety drives are connected to the SIMOTION CPU via PROFINET and/or internally via PROFIBUS DP Integrated.

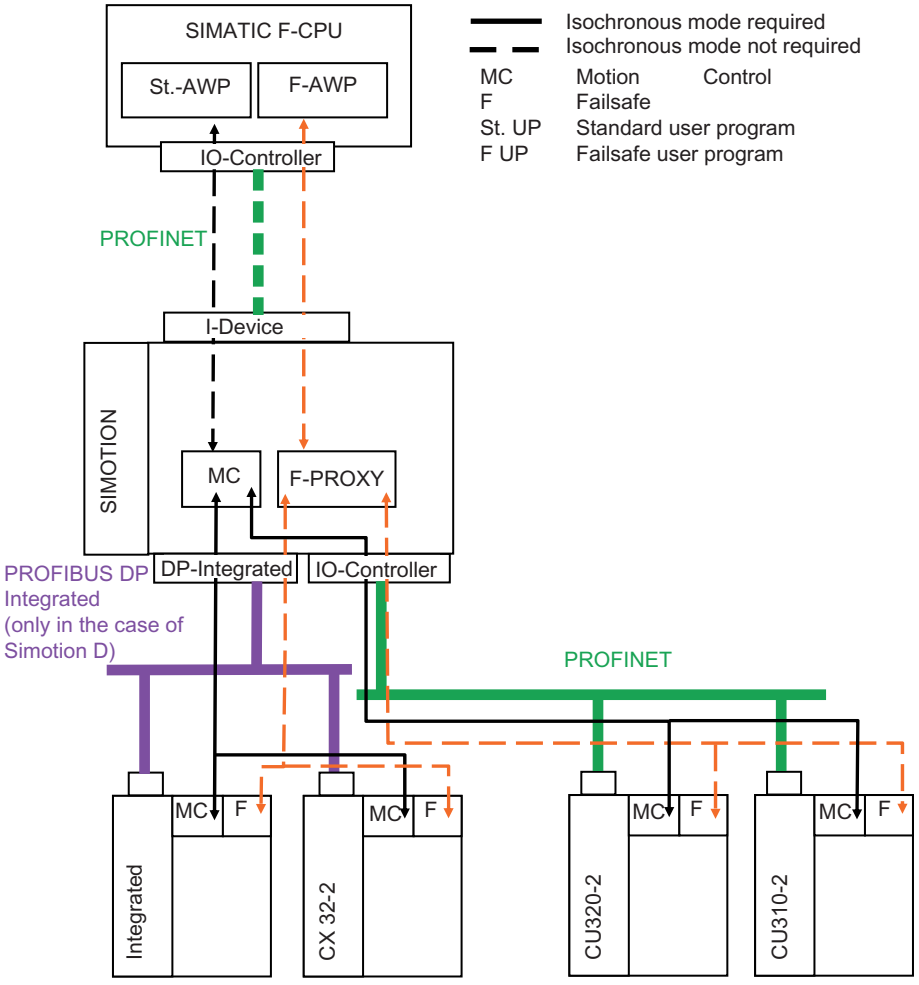


Figure 4-438 Topology for I device failsafe proxy for PROFINET drive units

Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units

Example of topology

The diagram below shows a topology in schematic form in which the Safety drives are connected to the SIMOTION CPU via PROFINET and PROFIBUS DP.

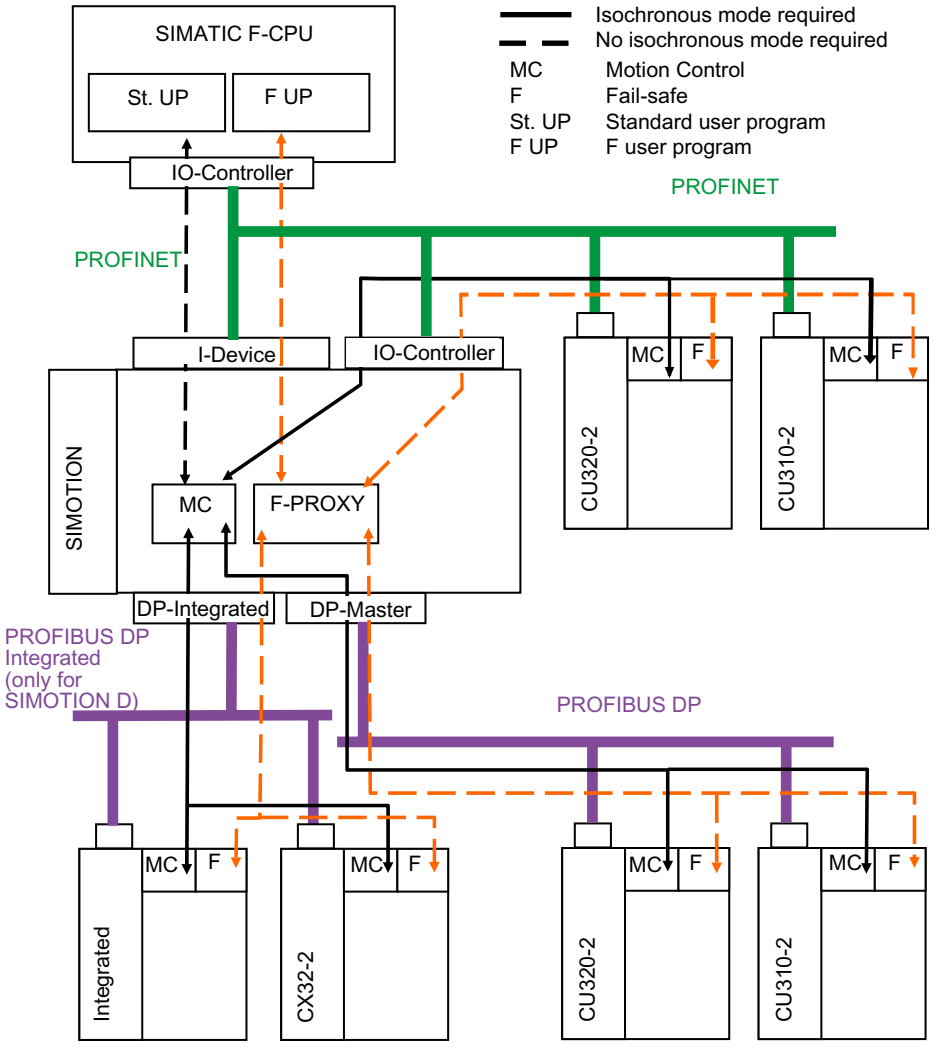


Figure 4-439 Topology for I device failsafe proxy for PROFIBUS and PROFINET drive units

Configuring I device failsafe proxy

Basic configuration process for I device failsafe proxy

Configuration requirements

A failsafe host communicates with the drives via the I device interface and an F-Proxy of a SIMOTION CPU. These drives may be located on PROFIBUS DP external, PROFIBUS DP integrated and the PROFINET IO system of the SIMOTION CPU.

Basic configuration process

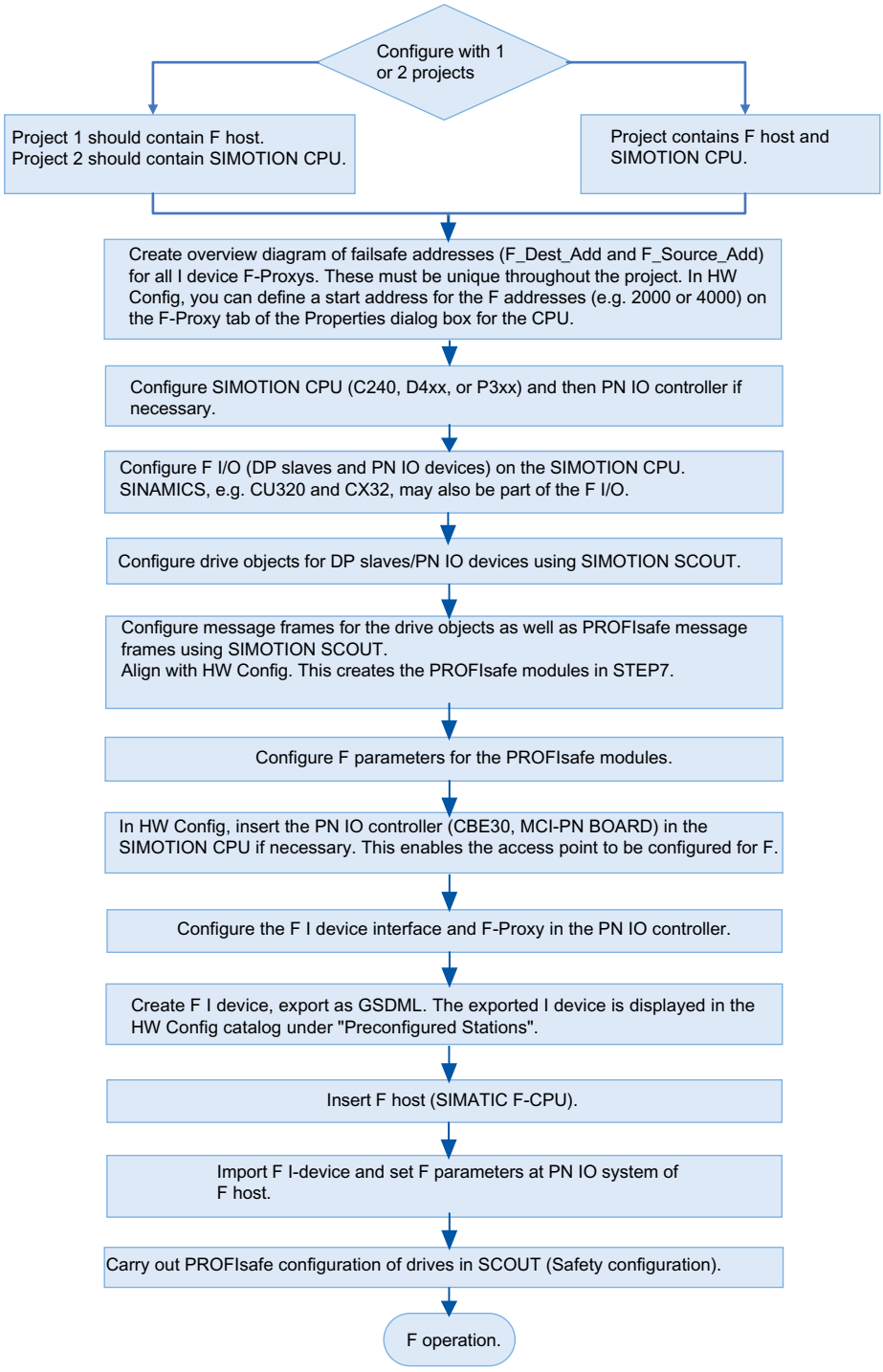


Figure 4-440 Basic configuration process for I device failsafe proxy

Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET

Configuration example for I device failsafe proxy

In the example below, you will configure a SIMATIC F-CPU 317F-2PN/DP V3.2. In addition, you will use a SIMOTION D435-2 DP/PN with a SINAMICS S120 CU320-2 PN, which is connected to the SIMOTION via the PN interface (CBE30-2). The exported I device F-Proxy of the lower-level SIMOTION CPU is then imported into the higher-level F-CPU. In the example, only one project is used in the configuration represented.

Configuring an I device F-Proxy

1. Create an overview diagram with the F-CPU, SIMOTION CPU and the drives intended to support PROFIsafe. In the example, there is just one SIMOTION CPU and a SINAMICS S120. Enter the start addresses of the CPU and the required drives into this overview diagram. The specified addresses are to be used later during configuration. The overview diagram is only required for larger projects.
2. Create a new project in SIMOTION SCOUT.
   You can find further information on configuration with PROFINET in the chapter Configuring PROFINET IO with SIMOTION (Page 2178).
3. Add a SIMOTION D435-2 DP/PN. The check box **Open HW Config** must be activated. Click **OK** to confirm. HW Config opens.
4. In HW Config, enter a start address for the F addresses of the F-Proxy under SIMOTION CPU. All F_Dest_Add for the lower-level drives then use this start address, making them easier to manage in the case of more extensive projects. If you use 4000 as a start address, for example, the first F_Dest_Add of the drive is allocated as 400, etc. Start address 2000 is used as standard.
5. Insert a CBE30 if necessary and configure the PROFINET network.
6. Add a SINAMICS S120 CU320-2 PN (e.g. V4.7) to HW Config and configure the interface.
7. Configure the drive unit in SIMOTION SCOUT with the help of the wizard. You should already be thinking here about selecting the appropriate safety functions in the drive; activate **Expanded Safety Functions via PROFIsafe**, for example.
8. Then insert a new TO axis and run through the axis wizard. In the wizard, you link the axis with the corresponding drive object of the S120, and a corresponding telegram (use symbolic assignment) is created automatically.
9. Save and compile the project.
10. After configuration, you have to select the PROFIsafe telegram. In the SIMOTION SCOUT project navigator, under the drive unit double-click on **<"Drive unit_xx"> - Communication > Telegram configuration**. The telegrams appear in the working area.

11. Mark the appropriate drive in the tab **IF1: PROFIdrive PZD telegram** of the telegram overview and in the bottom part of the window under **Adapt telegram configuration** select the entry **Add PROFIsafe**. The PROFIsafe telegram is added. Dependent on the drive, you can select from several telegrams.

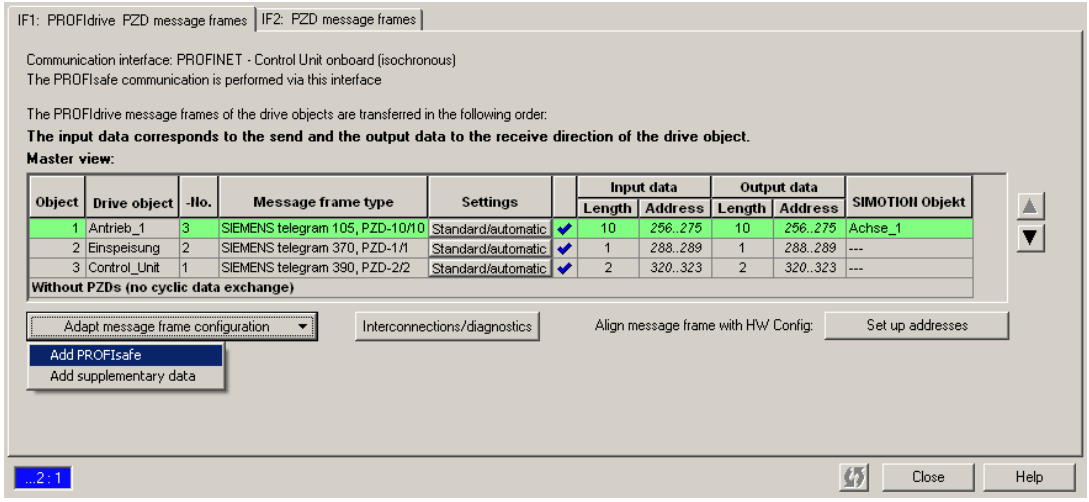


Figure 4-441 Adding PROFIsafe telegram

12. Save and compile the project.
13. Click on **Set up address** to run alignment between SIMOTION and HW Config. Configuration in SIMOTION SCOUT is completed by saving.
14. Move to HW Config and configure SYNC master and slave and an isochronous application.
15. Double-click in the SIMOTION D435 station overview on X1400 P1 and select the drive unit assignment in the **Topology** tab under **Partner port**. Click **OK** to confirm.

16. Double-click in the station on the PN IO interface to activate iDevice mode in the properties. In the **I device** tab, activate the check box **I device mode**. Click on **New...** and select the **Fail-safe I/O** entry in the dialog which opens under **Transfer area type**.

Note

In the **properties** dialog of the PN IO interface, you interconnect all the configured fail-safe I/Os directly (from STEP 7 V5.5 SP4). To do this, press the **Interconnect fail-safe I/O** button. The automatically generated transfer areas are displayed. You can then go straight on to Point 18.

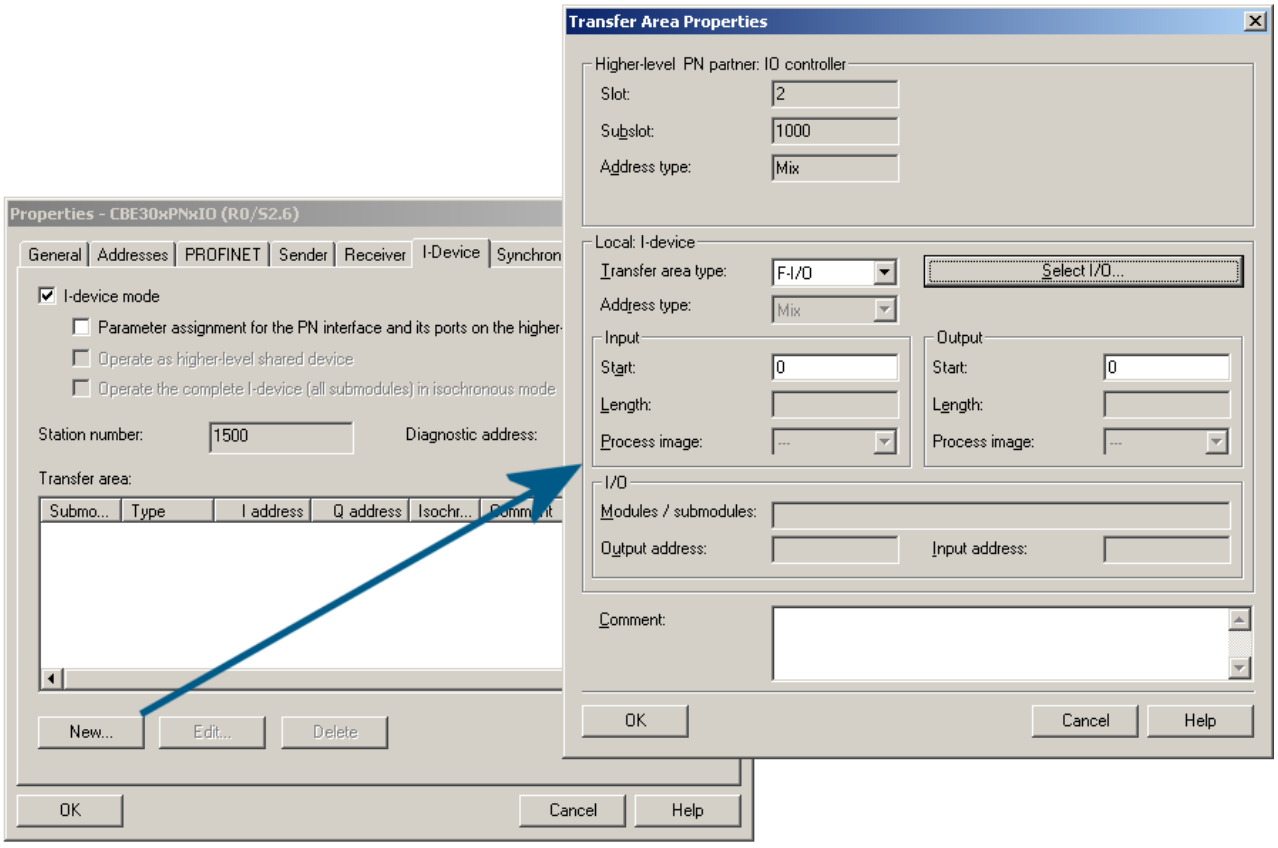


Figure 4-442 Properties of I device transfer area

17. Click on the **Select I/O** button and select the corresponding PROFIsafe channel in the dialog which opens. Confirm both dialogs with **OK**.
This completes configuration of the lower-level drive.

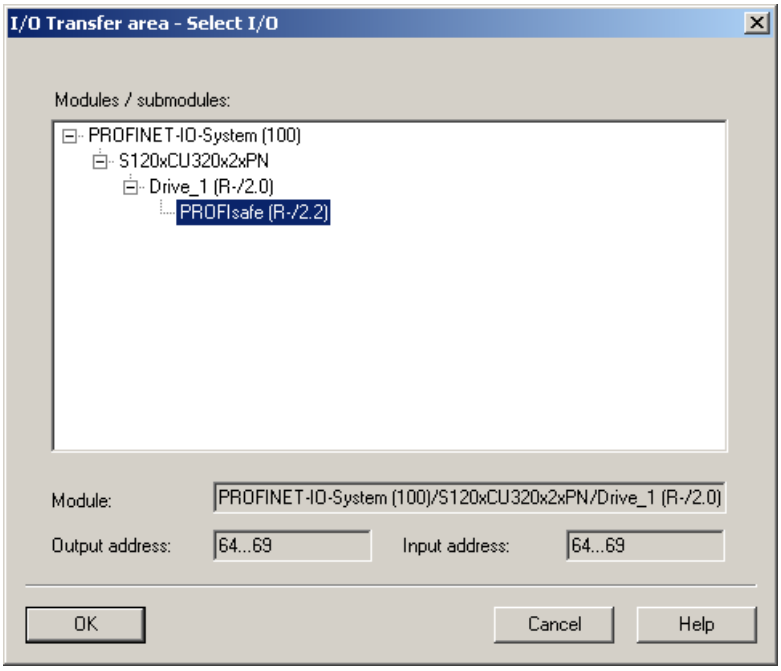


Figure 4-443 Selecting PROFIsafe channel of I/O

18. The transfer area must always contain two areas. You can either have two fail-safe I/O areas or you must define an application area for one fail-safe I/O area only. You should also refer to the iDevice chapter.

19. You can view the PROFIsafe settings for the drive unit. The detail view for the rack contains the entry **PROFIsafe**. Double-click on this to display the properties.
The **PROFIsafe** tab contains the F destination address under **F_Dest_Add**. This address must be unique within the entire project. If you are using several failsafe proxies, you must ensure that this address is only issued once. Change this value as required. The failsafe address is displayed in HW Config in the detail view in the station window under comments. This allows for clear assignment if using several participants.
The parameters F_CRC_Length=3-Byte-CRC and F_Par_Version=1 indicate PROFIsafe V2 mode. Please note these values because an I device F-Proxy configuration is only possible with this version and higher.
For more information about the fail-safe parameters, see PROFIsafe properties for configuration (Page 2336)

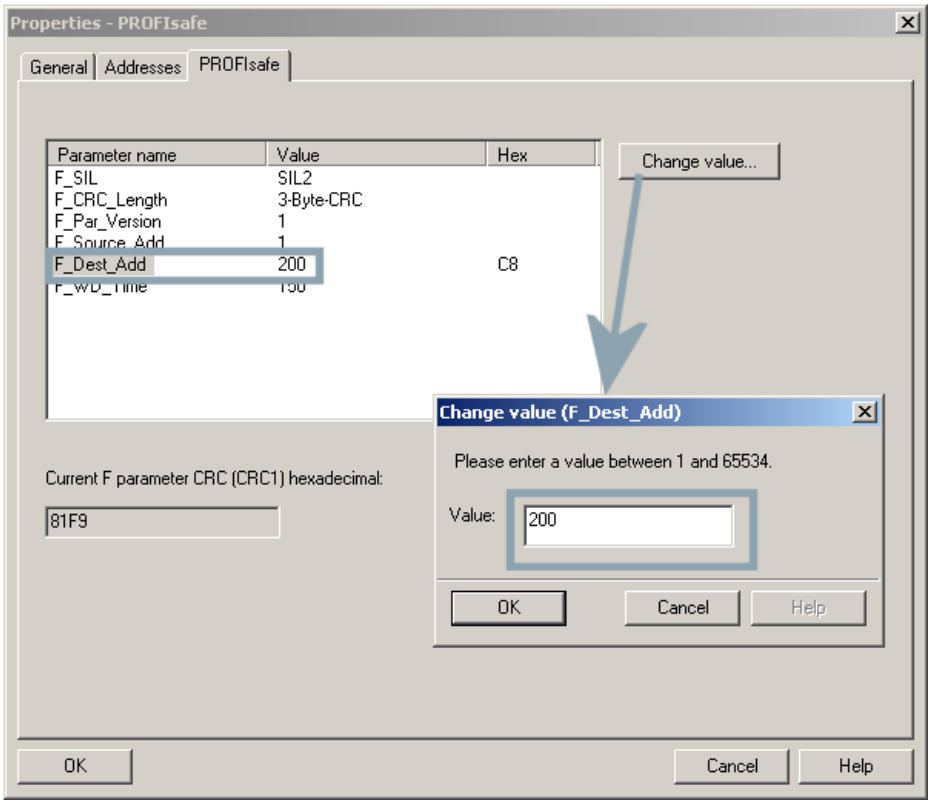


Figure 4-444 Setting the F destination address (F_Dest_Add)

20. The F destination address (F_Dest_Add) must match the PROFIsafe address during Safety configuration of the drive in SIMOTION SCOUT. In the example, this is address 200 dec or C8H. You enter the address in the Configuration window when configuring Safety Integrated. The value is stored in drive parameters p9610/p9810.

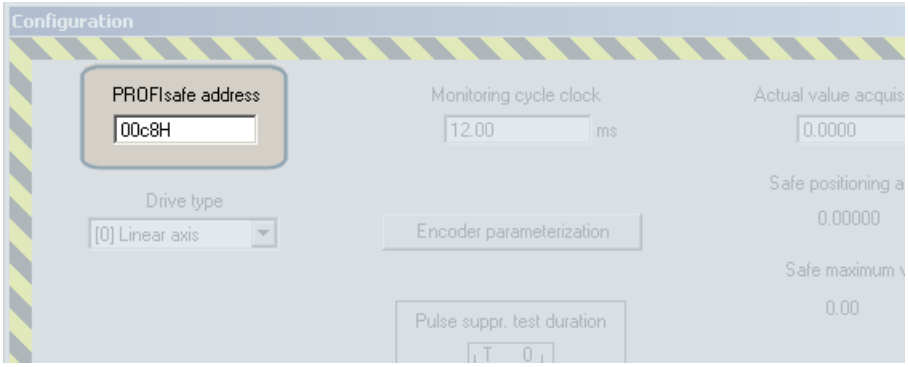


Figure 4-445 Entering F_Dest_Add as PROFIsafe address

21. Now produce the GSD file for the I device F-Proxy. In the menu, select **Options > Create GSD file for iDevice.** In the dialog which opens, click on **Create** and then **Install**. The I device is displayed under **Preconfigured Stations** in the hardware catalog.

22. You can now create a new project with F-CPU or open and use the existing project in the SIMATIC Manager. In our example, open the existing project in the SIMATIC Manager.

23. In the menu, select e.g. **Insert > Station > SIMATIC 300 Station.** Double-click on the station and then the entry **Hardware**. HW Config opens.

24. From the hardware catalog, insert e.g. an S7 300 rack if you want to select an F-CPU from the S7 300 series.

25. Insert the F-CPU, e.g. CPU317-2 PN/DP. This must at least be version V3.2.
26. Use drag&drop to move the I device previously created under **Preconfigured Stations** to the PROFINET IO network. Once saved and compiled, configuration is complete. Please observe the rules regarding device names in the chapter PROFINET IO and I device (Page 2238).

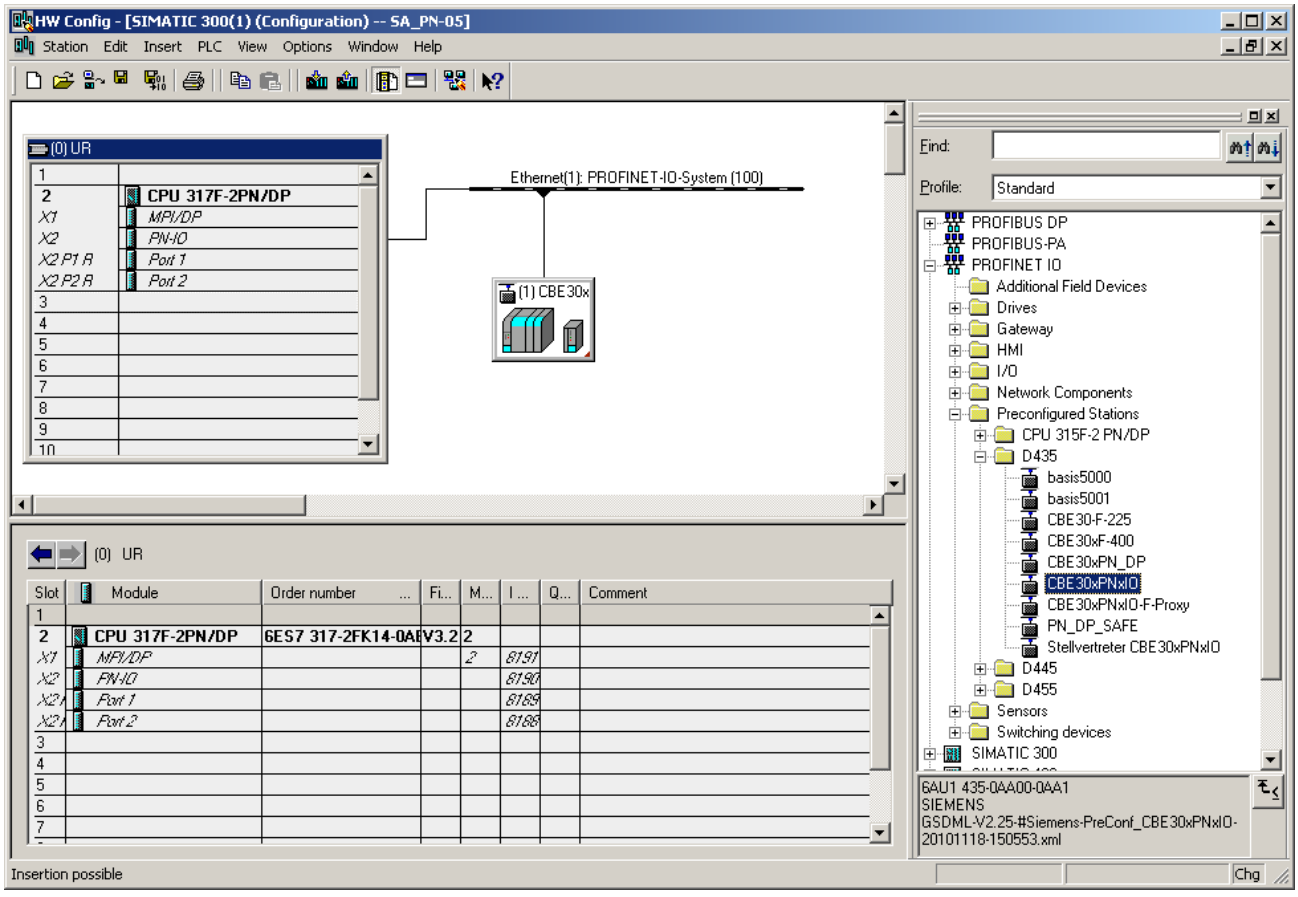


Figure 4-446 Master project with an I device failsafe proxy submodule

Adapting the F destination address (F_Dest_Add) in the existing project

Adapting the F destination address (F_Dest_Add) for the entire project

In an existing project, you can check whether the F destination address (F_Dest_Add) for the iDevice F-Proxy has been set correctly at a later point. The F destination address (F_Dest_Add) must be the same at the following locations:

- PROFIsafe slot of the drive on the SIMOTION CPU (HW Config)
- PROFIsafe slot of the I device for the SIMOTION CPU (HW Config)
- Safety configuration for drive in SIMOTION (SIMOTION SCOUT)

If you want to change the F destination address (F_Dest_Add) at a later point without having to create and install the iDevice again, you must make the change at the three locations referred to above.

In the example below, you set the F destination address (F_Dest_Add) to the value 300 (12CHex) for a SIMOTION CPU with a drive on PROFINET and an F-CPU in the same project.

How to check whether the F destination address (F_Dest_Add) is identical

1. Open the SIMOTION CPU project in HW Config.
2. In the detail view of the drive unit, double-click **PROFIsafe**. In the window that opens, switch to the **PROFIsafe** tab. The value 300 must be present next to **F_Dest_Add**. To change **F_Dest_Add**, click the **Change Value** button and enter 300 in the dialog box that appears.

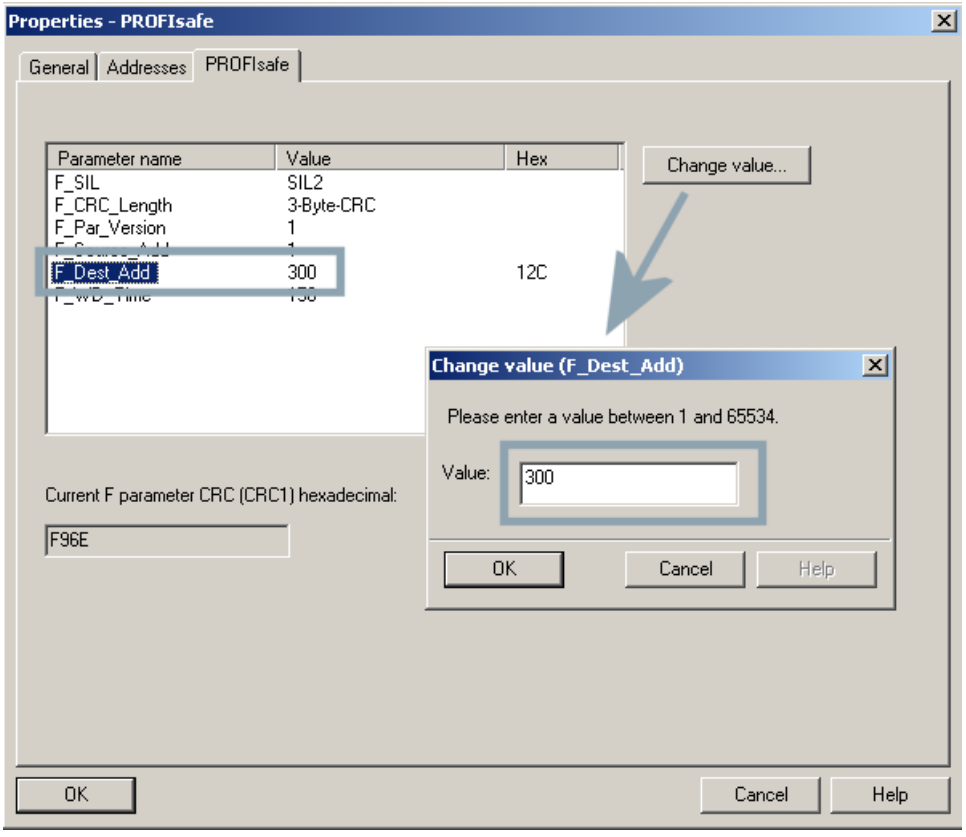


Figure 4-447 F destination address (F_Dest_Add) for PROFIsafe slot on drive unit

3. Confirm by selecting **OK** and save and compile the project.
4. Open the project with the F-CPU in HW Config.

5. In the detail view of the I device, double-click the **PROFIsafe I/O**, e.g. B. 6I/6O F-Periphery. In the window that opens, switch to the **PROFIsafe** tab. The value 300 must be present next to **F_Dest_Add**. To change **F_Dest_Add**, click the **Change Value** button and enter 300 in the dialog box that appears.

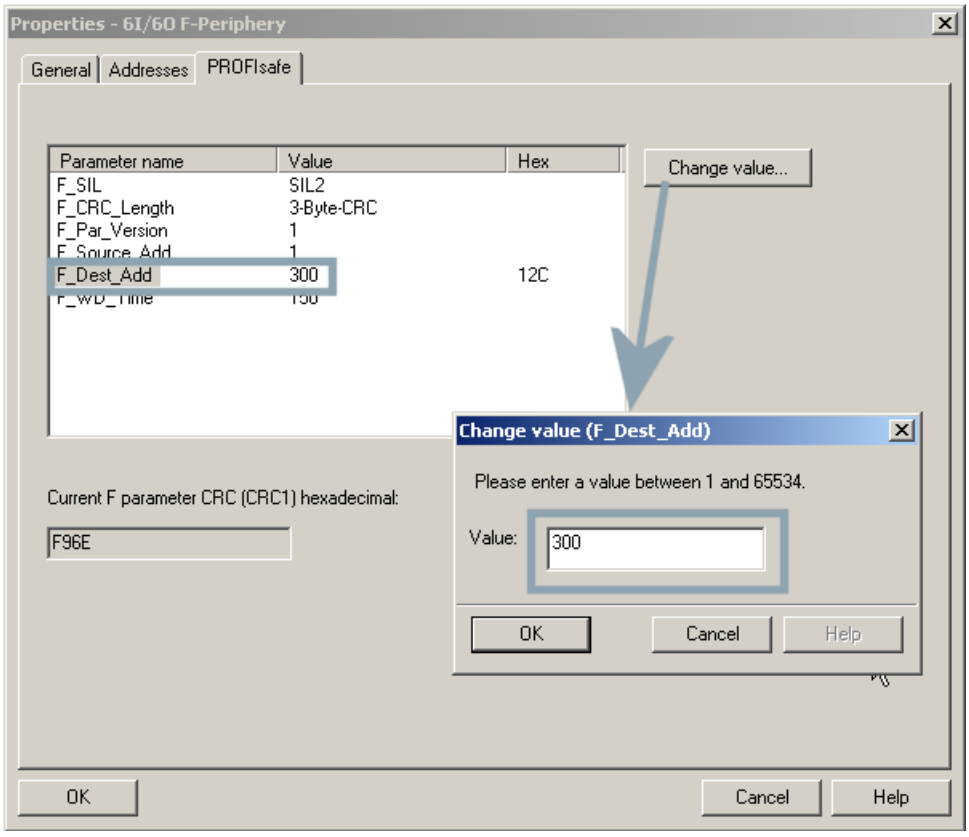


Figure 4-448 F destination address (F_Dest_Add) for iDevice on F-CPU

6. Confirm by selecting **OK** and save and compile the project.
7. Open the SIMOTION project in SIMOTION SCOUT.
8. In the project navigator, navigate to the drive (e.g. D435 > S120xCU320xCBE20 > Drives > Drive_1).
9. Under Functions in the project navigator, double-click **Safety Integrated**.
10. Click the **Configuration** button. The Configuration window appears.

11. Under **PROFIsafe address**, check the value 12CHex (300) and change it if necessary.

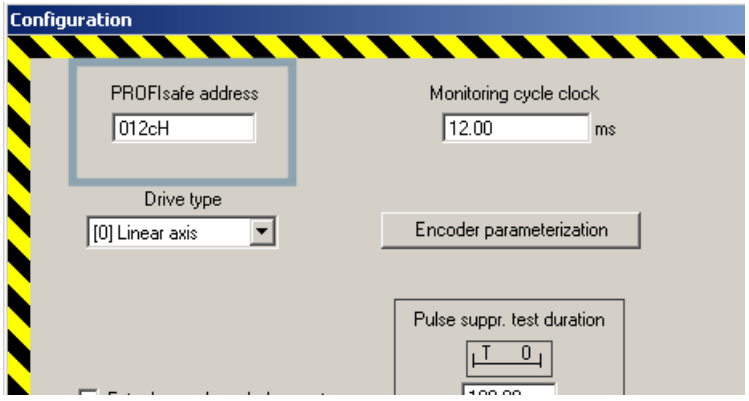


Figure 4-449 F destination address (F_Dest_Add) for configuring Safety Integrated

12. Confirm by selecting **Close** and save and compile the project.

Configuration of D435 with S120 on PROFINET and integrated PROFIBUS

Integrated drive on D435 for PROFIsafe

In the previous example you configured an S120 on D435 with PROFINET and imported it as an I-device. A SINAMICS_Integrated is now added to the project using the internal PROFIBUS DP.

This is how you configure a SINAMICS_Integrated

You have configured a project with D435 and SINAMICS S120 and imported it as an I-device to an F-CPU.

1. In SIMOTION SCOUT, configure the drive unit on SINAMICS_Integrated and insert a PROFIsafe telegram (see points 7-11 in the example Configuration example for SIMOTION D435 and SINAMICS S120 via PROFINET (Page 2350)).
2. In HW Config highlight SINAMICS_Integrated and double-click on the PROFIsafe module in the rack's detail view.
3. In the **Configuration** tab of the dialog which opens, highlight **PROFIsafe telegram** and click on **PROFIsafe...** . In the **PROFIsafe properties** dialog box, you can see and change the F Parameters.
   (If the **PROFIsafe...** button is not shown, you first have to click on the **Activate** button to make changes.)

4. Double-click on the PN IO interface of the SIMOTION CPU. In the **I-Device** tab in the dialog box which opens, select the **I-device mode** check box. Click on **New...** and select the **Fail-safe I/O** entry in the dialog which opens under **Transfer area type**. Click on the **Select I/O** button and select the corresponding PROFIsafe channel in the dialog which opens.

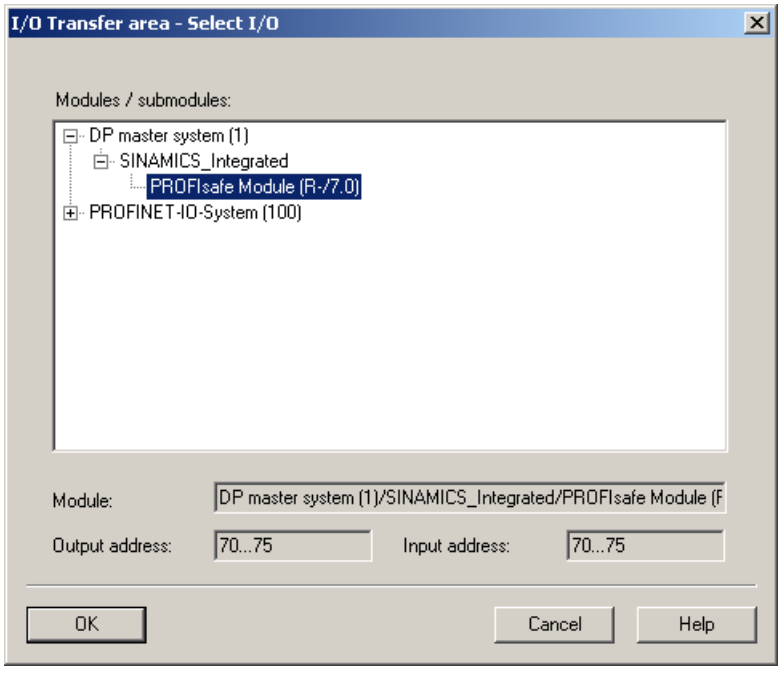


Figure 4-450 Releasing PROFIsafe module for SINAMICS_Integrated for transfer area

5. Click **OK** to confirm the dialog box. In the **Transfer Area Properties** dialog box, the inputs/outputs are assigned and an automatically generated comment is displayed. This comment includes, among other things, the subslot, the SIMOTION device name, the connection, and the device name of the drive. The F_Dest_Add is at the end. You can change the comment if necessary.

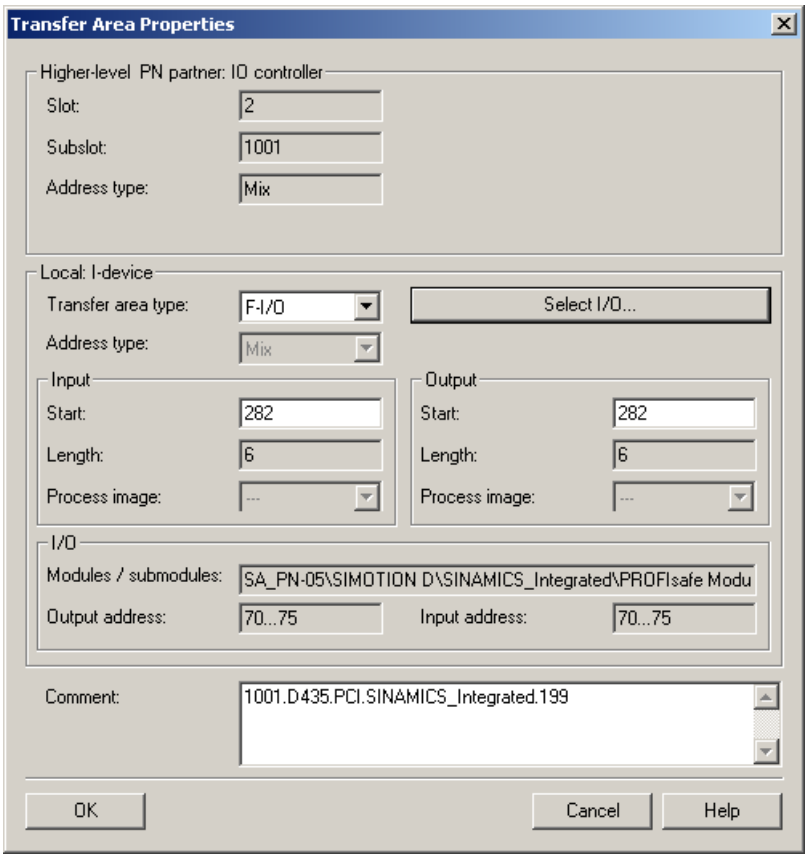


Figure 4-451 Properties of I-device transfer area, comment

6. Click **OK** to confirm this dialog box. The failsafe data for the two drives is displayed in the transfer area.
   This concludes configuration of the lower-level drive. All you need to do now is save and compile the station.

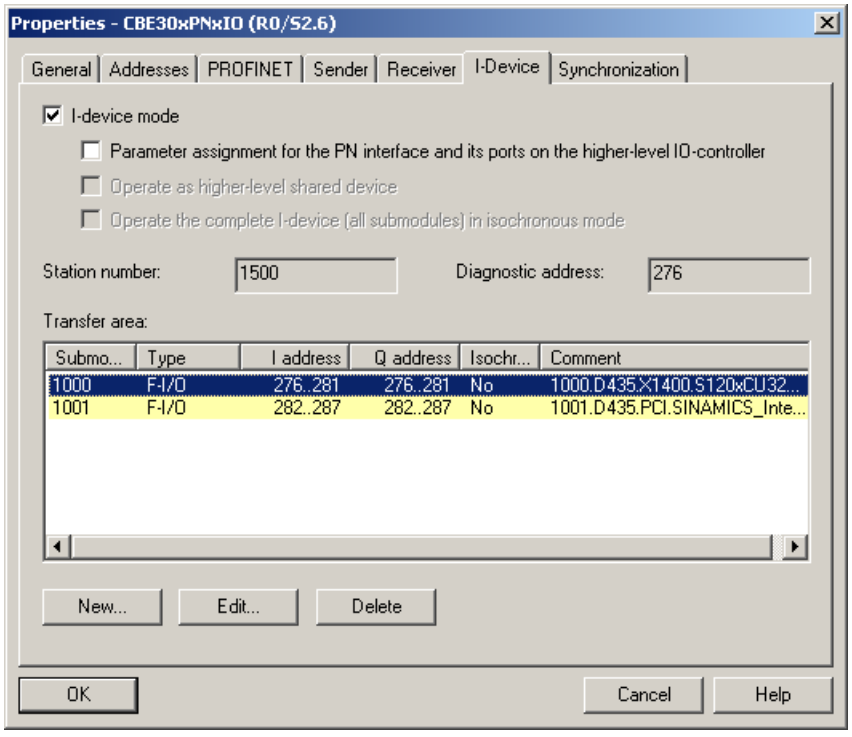


Figure 4-452 I-device transfer area

7. Now produce the GSD file for the I-device F-Proxy. In the menu, select **Options > Create GSD file for I-device**. In the dialog which opens, click on **Create** and then **Install**. The I-device is displayed under Preconfigured stations in the hardware catalog.
8. As in the previous example, create an F-CPU in the project and add the I-device of the SIMOTION module.
   The diagram shows the project with F-CPU and I-device F-Proxy with one drive on PROFINET and one drive on SINAMICS_Integrated on a D435.

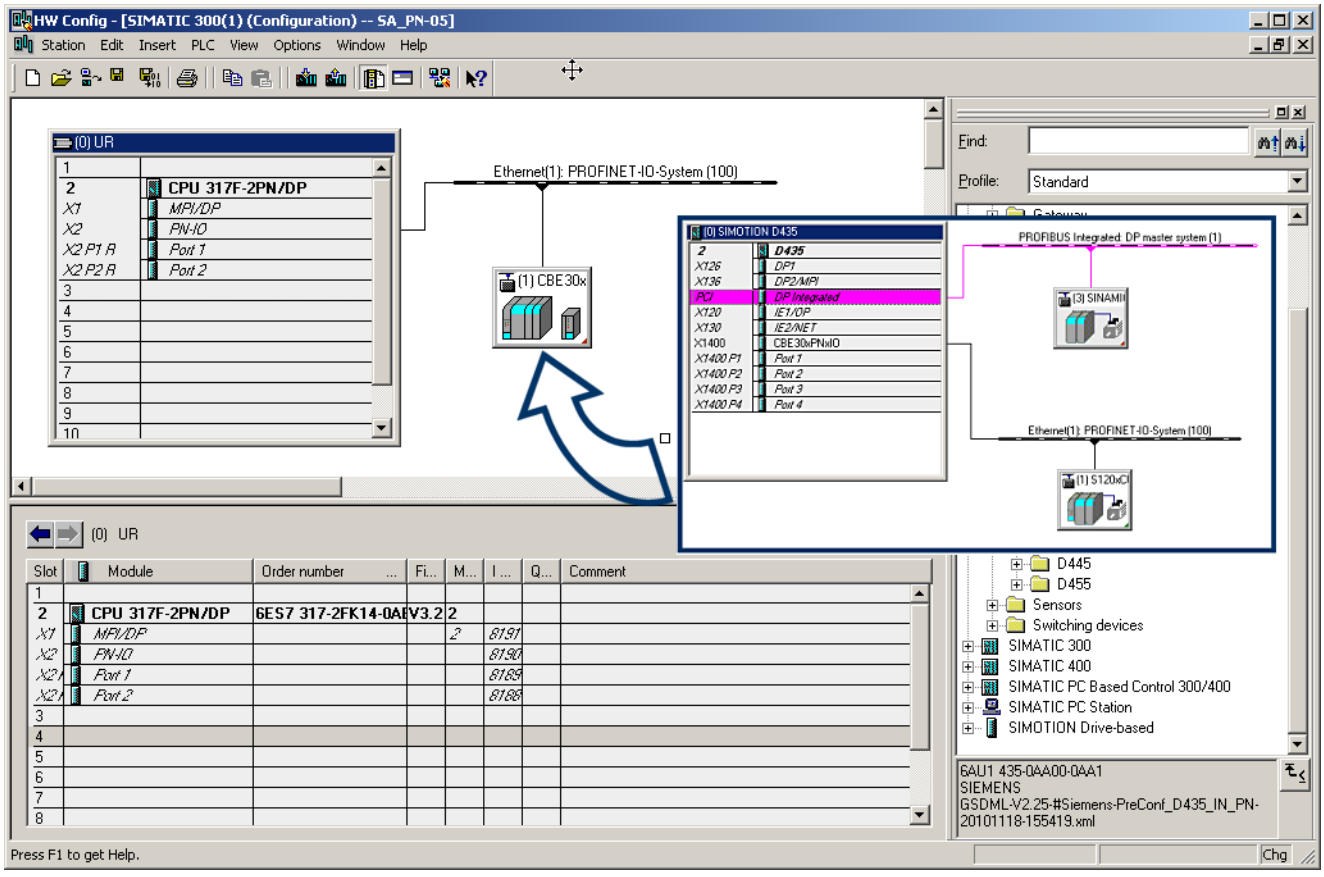


Figure 4-453 F-CPU with I-device F-proxy on PROFINET and integrated PROFIBUS

Upgrading an existing system with PROFIsafe via PROFIBUS to PROFIsafe via PROFINET

PROFIBUS to PROFINET

If PROFIsafe communication on an existing system has been operated via PROFIBUS up until now and you want to switch it to PROFINET, then you will need to upgrade the system.

How to carry out the upgrade

1. Delete the old I slave link.
2. If necessary, switch the DP interface from DP slave to DP master (SIMOTION CPU).
3. The PROFIsafe message frame configuration settings for the drives can remain unchanged.

4. If necessary, create the F parameters **F_Par_Version = 1** and **F_CRC_Length = 3-Byte-CRC** in order to use the PROFIsafe V2 standard. This will mean that a link cannot be established in the I device. V2 is automatically selected when the new PROFIsafe slots are created.

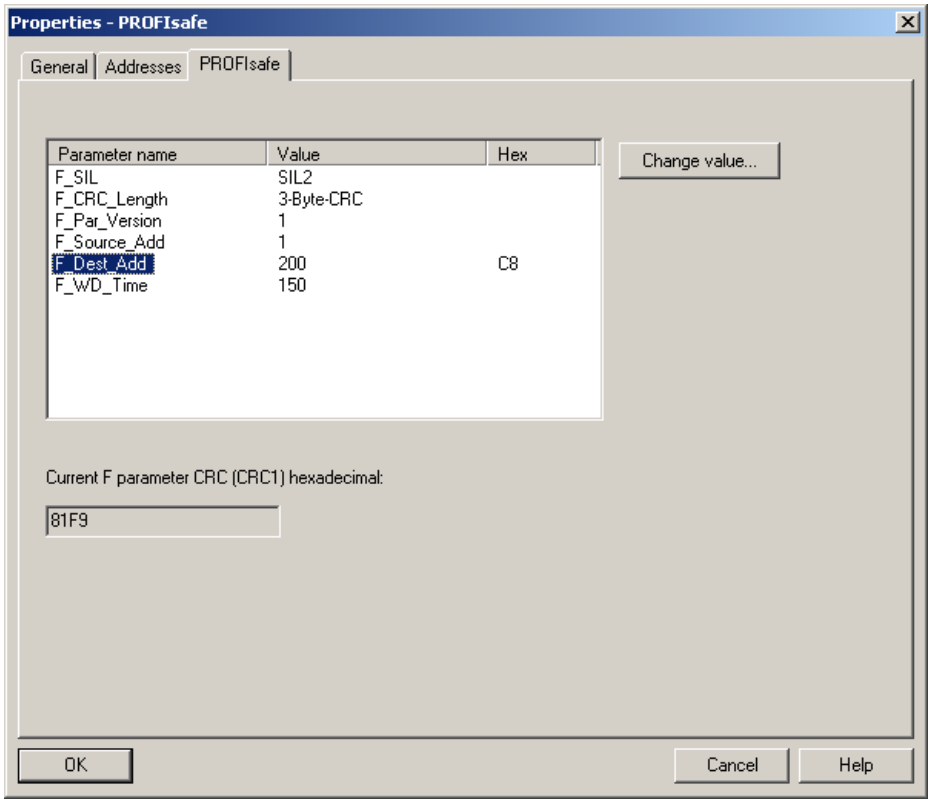


Figure 4-454 Properties of PROFIsafe taking the example of I device failsafe proxy

5. Select I device mode in HW Config on the CBE30.
6. Select **New** , followed by **failsafe periphery** under transfer area type to select the I/O and create a submodule.
7. Then create a new GSD via **Options > Create GSD file for I device...** and install it.
8. Link the GSD file to the S7 F-CPU.

General information on F destination addresses (F_Dest_Add) with iDevice F-Proxy

Communication addresses for F source (F_Source_Add) or F destination address (F_Dest_Add)

Create new F hosts, F modules/submodules in HW Config. HW Config then suggests the F source/ destination address (F_Dest_Add) as a default setting. You can change or overwrite this default setting. The default failsafe addressing is based on the failsafe start address parameter on the F-CPU and also on the SIMOTION CPU if applicable.

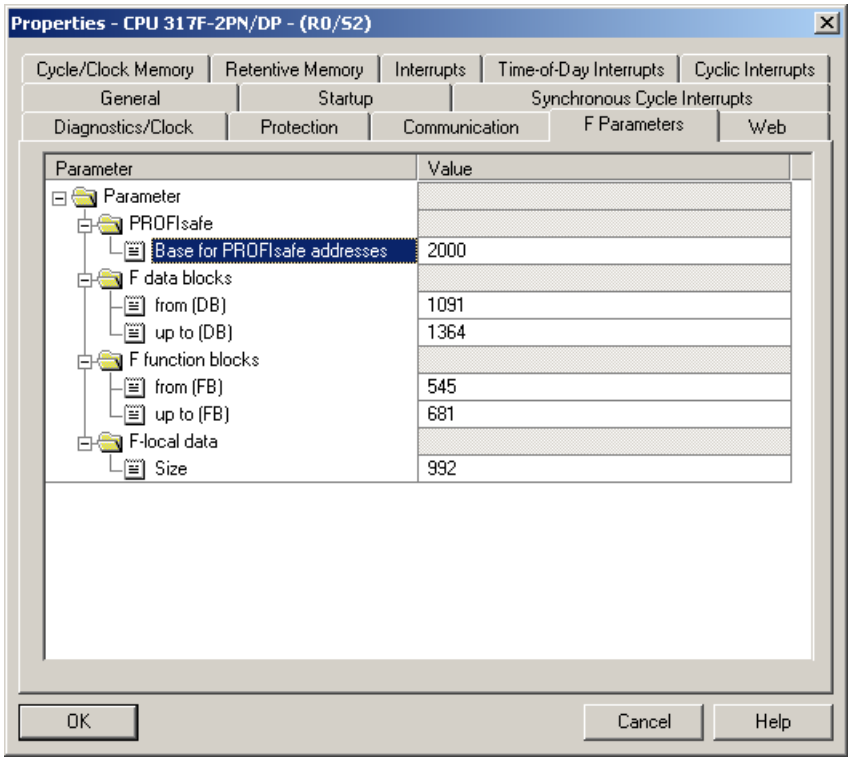


Figure 4-455 PROFIsafe start address

The F source address (F_Source_Add) is assigned in the same way as for Siemens F modules with OM:

- PROFIsafe start address of CPU + number of DP master system for PROFIBUS
- PROFIsafe start address of CPU for PROFINET
- 1, if the CPU does not have a PROFIsafe start address parameter (standard CPU and F-H-CPU)

Guidelines for addressing

- The full range of values between 1...65534 is used for addressing.
- The F destination address (F_Dest_Add) is assigned automatically.
- Automatic issuing takes place when plugging in as with the other failsafe modules in Step7: starting with an start value and working upwards, the system looks for the next free address.
- The start value is the PROFIsafe start address of the CPU/10 (or 1022, if there is no PROFIsafe start address or if it is greater than 10000).

- If the F start address is changed (CPU parameter), then F source and F destination addresses (F_Dest_Add) that have already been issued are not updated (remain in place). The change only affects the default address of newly created submodules.
- Several F-Source addresses may be issued in an F-CPU. The F source address and F destination address (F_Dest_Add) are included in the PROFIsafe CRC total.
- If addresses from an issued range of addresses are no longer used (when failsafe submodules are subsequently deleted), the gaps are filled first when new addresses are issued.

Note

If you want to use Safety to work beyond the project limits/proxies, you must ensure that the F source and F destination addresses are unique within the entire project. Before starting configuration, you should have/produce a failsafe addresses plan where unique ranges of failsafe addresses are assigned to particular sub-projects.

Automatic transfer of the F-destination addresses

Accept F-destination address (V5.2 and higher)

All axes are configured and commissioned during initial commissioning of a machine. Next the PROFIsafe telegrams are added in the telegram configuration of SIMOTION SCOUT or Starter. As soon as the telegram is used for the F-Proxy it can no longer be changed. The logical address is specified during "Save and compile" with symbolic assignment.

After the slot for the PROFIsafe telegram has been accepted in HW Config, the PROFIsafe telegram also receives its F-destination address. Up to V5.2, this address had to be entered manually for all drives with PROFIsafe while commissioning Safety Integrated. To avoid incorrect entries, the "Accept F-destination address" button is available during Safety commissioning as of V5.2 and higher.

F-destination address during Safety Integrated commissioning

You apply the F-destination address for PROFIsafe during drive commissioning.

1. In the SIMOTION SCOUT project navigator, click on "Functions > Safety Integrated" below the drive. The Safety configuration window opens in the work area. PROFIsafe is configured as Safety function.
2. Click the "PROFIsafe configuration" button.
3. You can view the "F_Dest_add" in the "PROFIsafe configuration" window.

4. Click "<" to accept the F-destination address as PROFIsafe address.

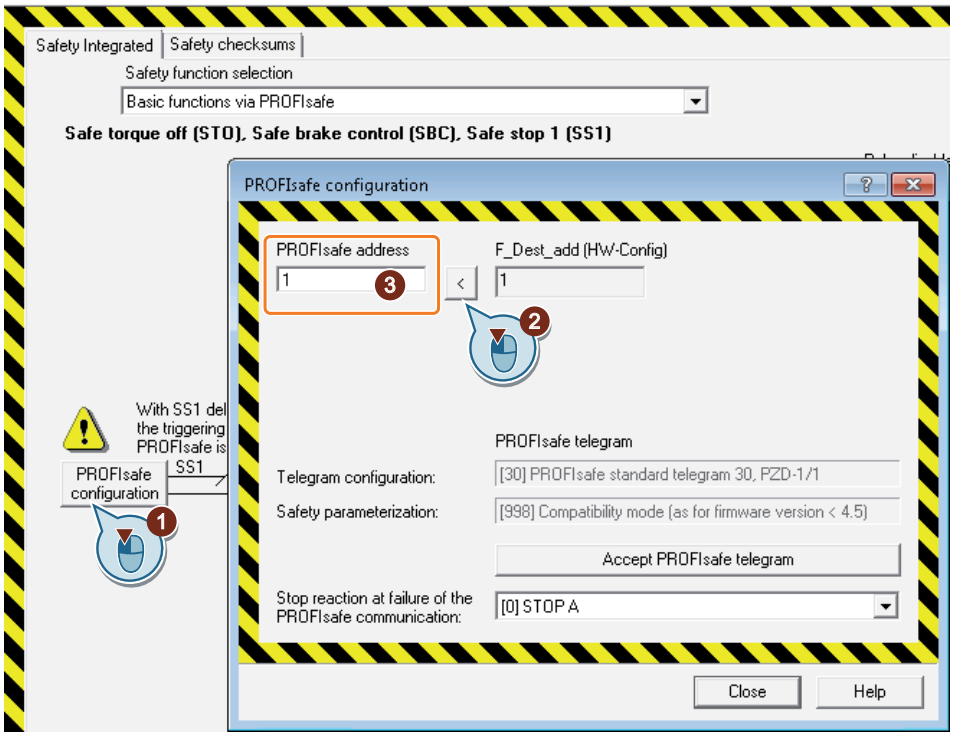


Figure 4-456 Accepting the F-destination address as PROFIsafe address

5. Click "Close" to confirm.

Scripting

The "Accept F-destination address" function can also be programmed via scripting. To do so, program the following scripting commands.

- GetFDestAdressFromHWCN
- SetUpAddresses

Shared device via PROFINET

General information on shared device

Description

With the Shared Device functionality, you can configure access to an IO device with several IO controllers using PROFINET. This enables channels/modules to be flexibly assigned to different IO controllers. This option is available for inputs and outputs. You can use this mechanism to access the fail-safe data of a drive configured below a SIMOTION CPU via the F-CPU, for example.

Note

When configuring PROFIsafe, I-Device F-Proxy functionality should be used rather than Shared Device.

Schematic diagram of Shared Device

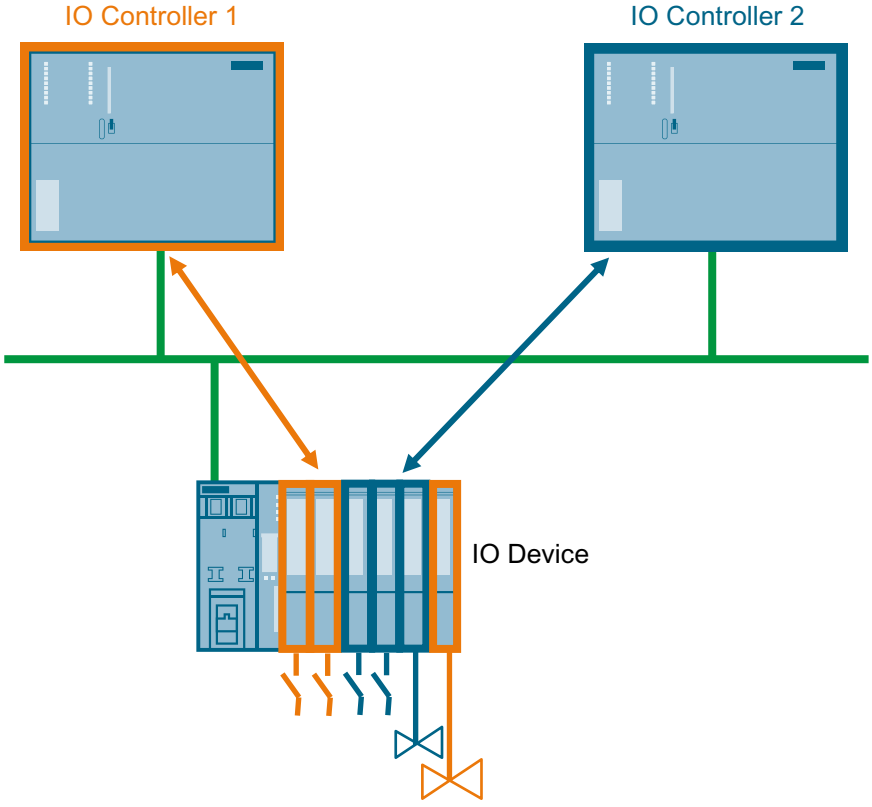


Figure 4-457 Schematic diagram of Shared Device

Software requirements

- SIMATIC Step 7 V5.5 or higher
- SINAMICS firmware and Support Packages V4.3.2 or higher

- SIMOTION SCOUT V4.2 or higher
- S7 F Configuration Pack, V5.5 SP7 or higher (if using Safety/PROFIsafe)
- GSDML file V2.25 or higher

Configuration principles

- I/O addresses can be assigned as usual for the submodules (that are assigned to the controller).
- A shared device must have the same IP parameters and the same device name in each station. There are two types of configuration case:
  - Shared device in the same project: STEP 7 takes important consistency check functions away from the user. STEP 7 checks that the IP parameters have been assigned correctly and monitors the IO controller access to the individual submodules to ensure that it is correct.
  - Shared device in different projects: The stations with the IO controllers that use the shared device are created in different projects. It is important to ensure in each project that the shared device is configured in exactly the same way in each station. Only one IO controller may have full access to a submodule in any case (see below). The IP parameters and device names must be identical. Inconsistencies in the configuration will cause the shared device to fail.

See also

Shared device (Page 2131)

Shared device in a STEP 7 project

Introduction

In the following example, the simplest configuration of a shared device is described: Two IO controllers (SIMOTION D445-2 DP/PN and CPU 317F-2 PN/DP) share the submodules of an IO device (ET200S HF). The two IO controllers are in the same STEP 7 project with the advantage that the consistency check is made automatically.

Procedure

To be able to use the shared device function, you need to take certain configuration steps in SIMOTION SCOUT, SIMATIC Manager, and HW Config.

Preparatory steps for SIMOTION CPU

1. In SIMOTION SCOUT, create a project called **Shared device project**.
2. Insert a SIMOTION D445-2 DP/PN and configure it.
3. Open the SIMOTION CPU in HW Config and configure the PROFINET interface.

4. Configure a PROFINET IO device ET 200S (IM151-3PN HF) with several submodules as shown in the figure below.

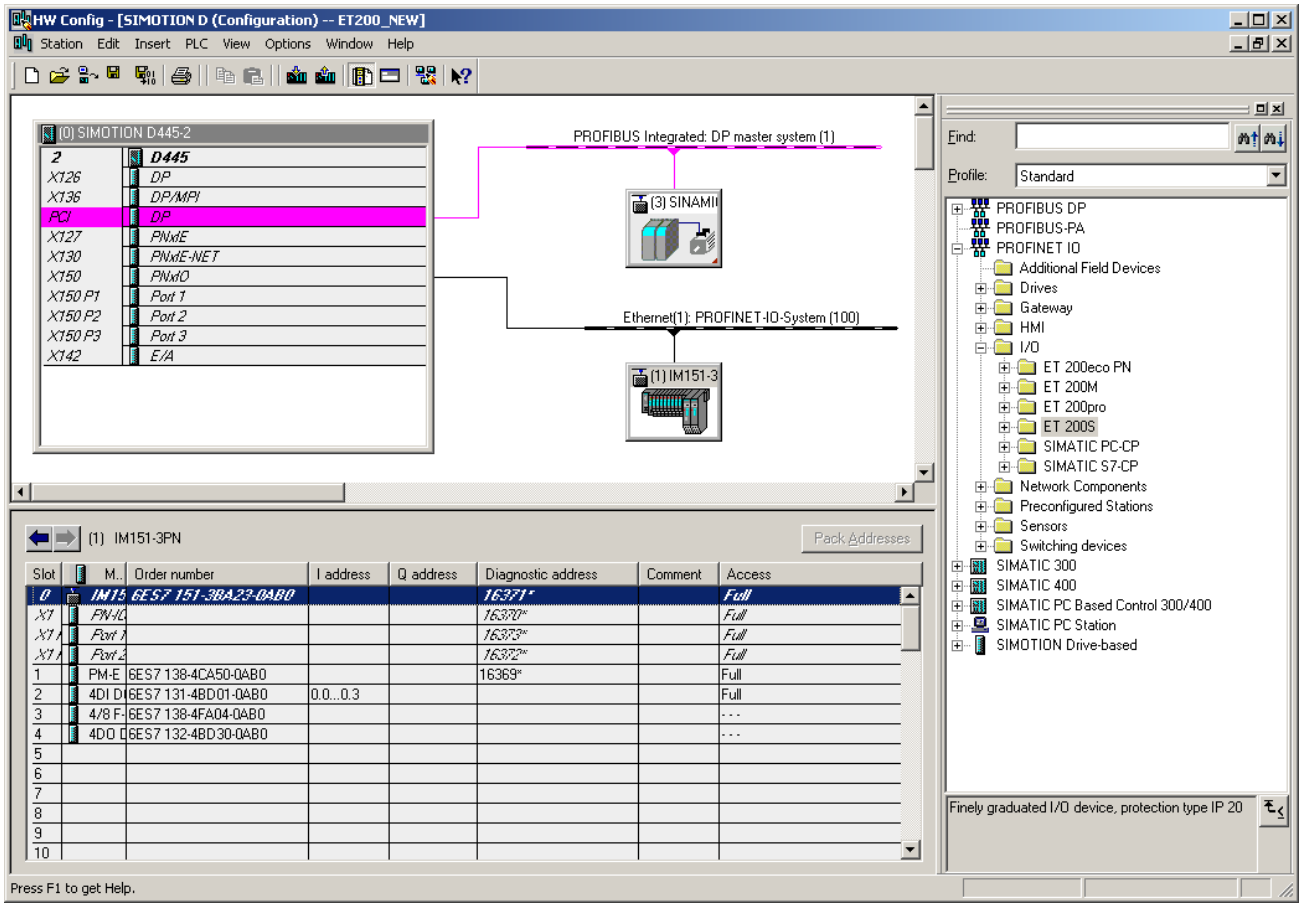


Figure 4-458 SIMOTION D445-2 DP/PN with IO device ET200S

5. **Save and compile** in HW Config.

Preparatory steps for SIMATIC CPU

1. Open the project you created in the SIMATIC Manager.
2. Insert a SIMATIC 300 station and open it in HW Config.
3. Insert a CPU 317F-2 PN/DP, for example, and configure the PROFINET interface.
4. **Save and compile** in HW Config.

Creating the shared device

1. Open one of the SIMOTION CPUs you created in HW Config.
2. **Copy** the IO device ET200S you created using the context menu (right mouse button).
3. **Save** the hardware configuration and close the configured station.
4. Open the station you created previously with the SIMATIC F-CPU in HW Config.

5. In order to add the IO device as a shared device, right click on the PROFINET IO system. Select the context menu command **Paste shared**.

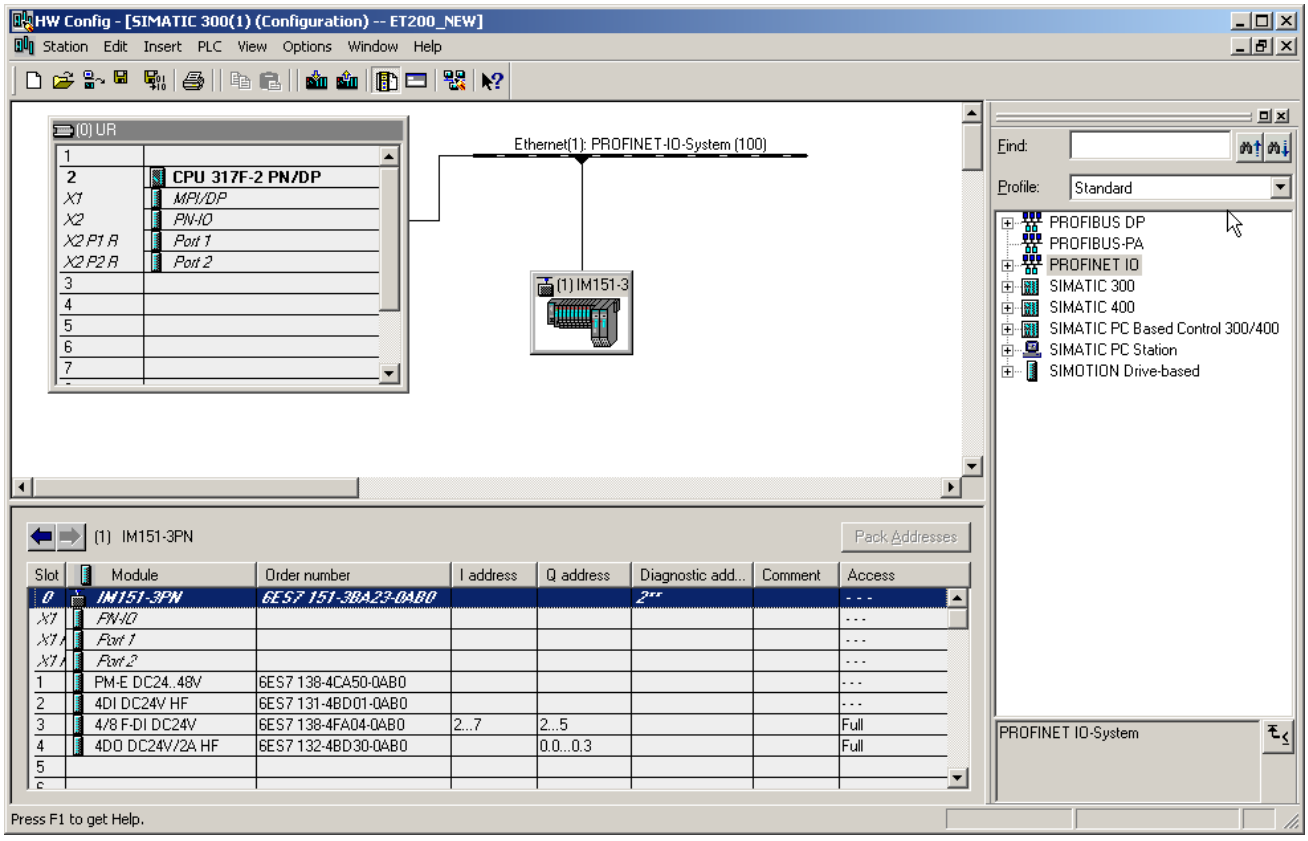


Figure 4-459 SIMATIC CPU 317F-2 PN/DP with shared device ET200S

6. Save the hardware configuration and close the configured station.

You have successfully created the shared device, now set the assignments of the submodules to the configured stations.

Assigning submodules

The submodules must be assigned separately for each station. Remember that changes to a station will also impact the other station(s)! A submodule can only ever be assigned to one station!

1. Open the Properties dialog box of the PROFINET IO device for the SIMOTION CPU.
2. Click the **Access** tab.

3. Configure the access to the individual submodules. To do this, select the type of access from the drop-down list in the **Value** column. You can select from the following:
   – No access to the submodule: "- - -"
   – Full access to the submodule: "full"

   Note that the setting "Full" automatically leads to the setting "- - -" in the other station(s).

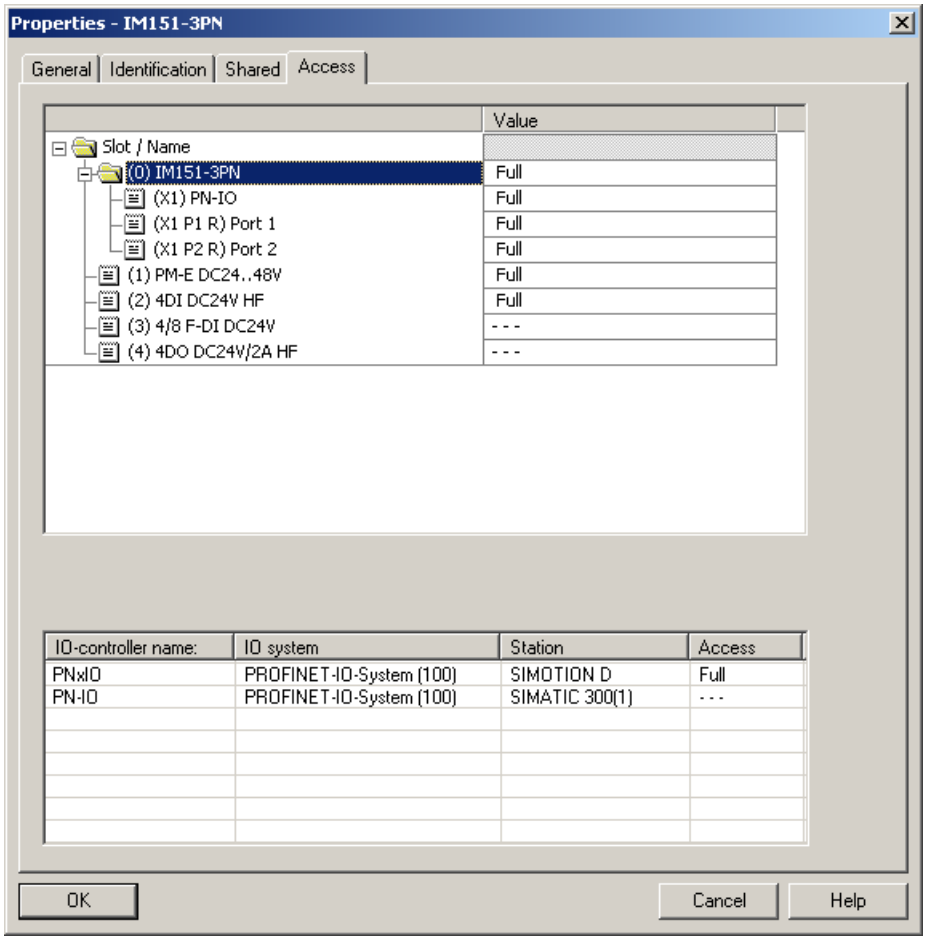


Figure 4-460 SIMOTION D445-2 DP/PN access to ET200S

4. Save and compile the station and close it.

5. Repeat steps 1 to 4 for the shared device on the SIMATIC F-CPU.

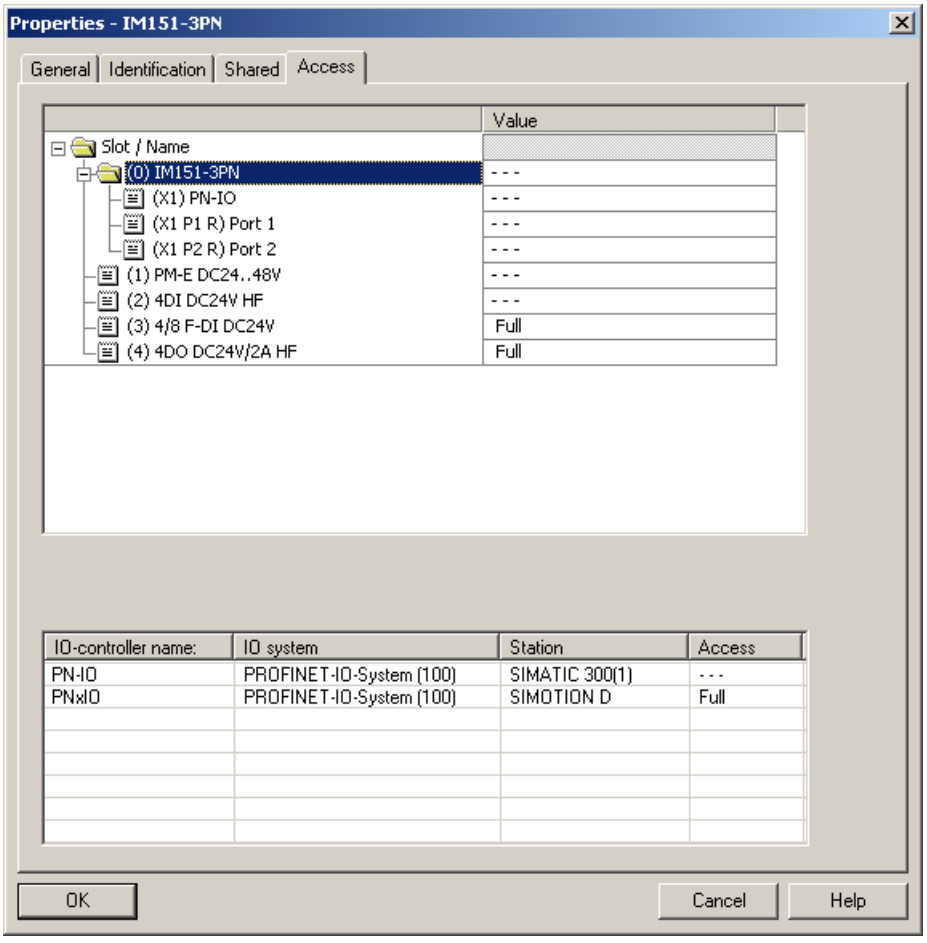


Figure 4-461 SIMATIC CPU 317F-2 PN/DP access to shared device ET200S

6. Then download the configuration to the stations.

Shared device in the user program

The shared device has no special role in the user program. The submodules assigned in the station are addressed as usual through your addresses, the other submodules do not receive addresses.

Shared iDevice and PROFIsafe

Using a shared I-Device

Description

An I-Device can be operated on two higher-level IO controllers as an IO device. This functionality is known as shared I-Device. It provides the following application areas:

- One of the higher-level controllers works as the "Automation CPU" and uses the SIMOTION I-Device as an isochronous I-Device in the IRT operating mode.
- The other higher-level IO controller works as the "Safety CPU" and uses the submodules configured as the F-Proxy in the RT operating mode.

This means that an I-Device can communicate with two higher-level controllers as an IO device. At the same time, higher-level controllers can access certain modules, e.g. an F-CPU can access the F-Proxy submodules, using the shared device.

Schematic diagram

The schematic diagram below shows two IO controllers which share the submodules on the I-Device of a third IO controller.

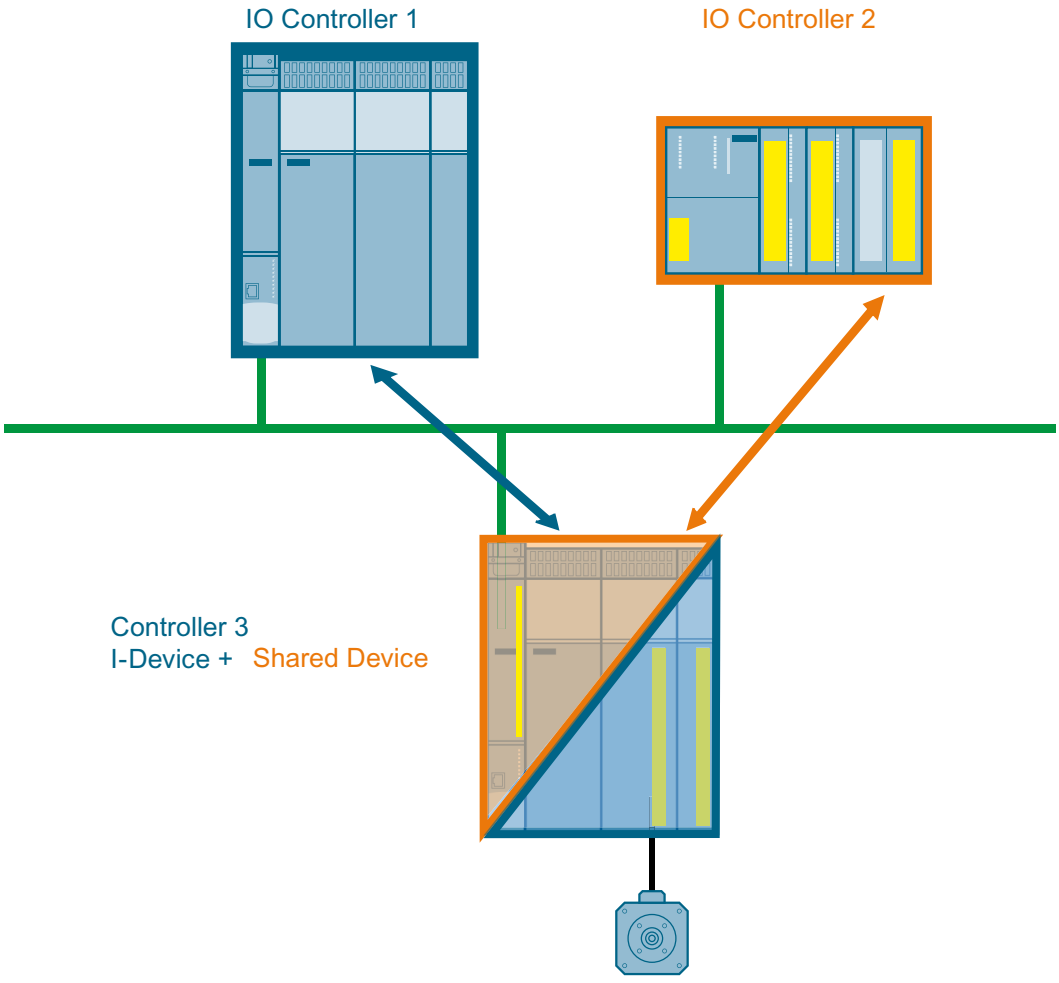


Figure 4-462 I-Device as Shared Device

Note

In this configuration, only the submodules that are used by the "Automation CPU" are configured as isochronous (IRT).

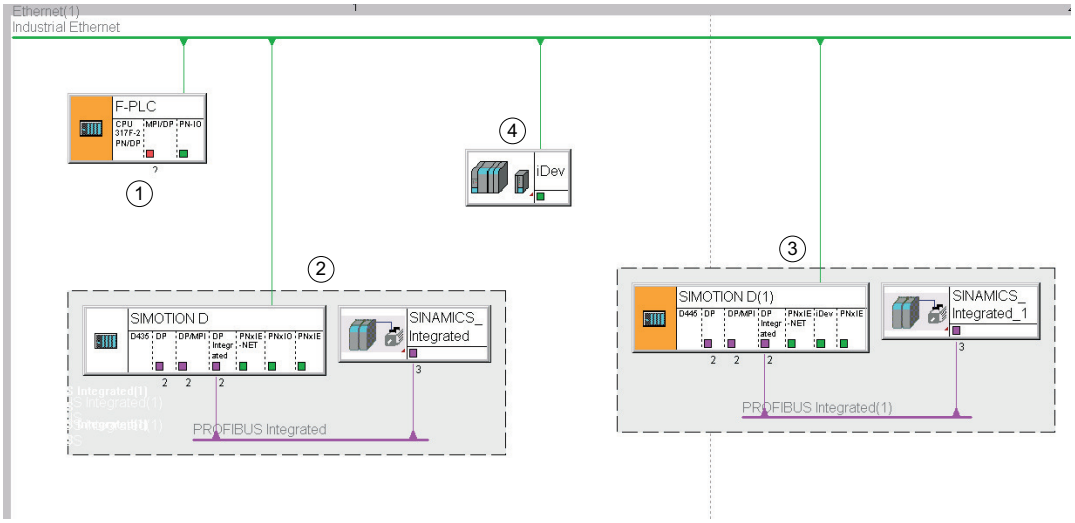
Basic information on this subject can be found in the section Shared I-Device (Page 2255).

Configuration example for shared iDevice and PROFIsafe

Configuration example

In the following example, a SIMOTION D445-2 DP/PN with two Safety drives on SINAMICS_Integrated is operated as a shared iDevice on a SIMOTION D435-2 DP/PN and a SIMATIC CPU317F-2. The SIMOTION D435-2 takes on the motion control tasks, while the SIMATIC CPU317F-2 is responsible for safety monitoring.

The overview diagram below shows the network view. The IO controllers are in the same subnet and are located in a project.



1 SIMATIC CPU 317F-2 as higher-level IO controller in RT mode (F-CPU)

2 SIMOTION D435-2 DP/PN as higher-level IO controller in IRT mode

3 SIMOTION D445-2 DP/PN with two Safety drives on SINAMICS_Integrated (PROFIBUS)

4 iDevice of SIMOTION D445-2 DP/PN which is connected to both higher-level IO controllers (shared iDevice).

Note:
In this example, the SIMOTION D445-2 DP/PN is shown twice in Netpro: Once as an iDevice proxy and once as a complete SIMOTION D device.

Figure 4-463 Configuration example for PROFIsafe via shared iDevice, representation in Netpro

Basic process for the example configuration

The following configuration steps must be observed for the example configuration:

1. Create a new project in SIMOTION SCOUT and configure a SIMOTION D435-2 DP/PN and a SIMOTION D445-2 DP/PN. These must be configured on the same subnet.
2. Configure two drives (power unit, supply, Safety functions, F destination address F_Dest_Add) on the SINAMICS_Integrated of the SIMOTION D445-2 DP/PN. (See also Configuring the iDevice F-Proxy (Page 2350))
3. Create two axes and connect each of them to one of the two drives in the axis configuration.

4. Insert the PROFIsafe telegrams in the telegram configuration for the drive unit and align the telegrams with HW Config.
   The diagram below shows an example project with the PROFIsafe telegram configuration open.

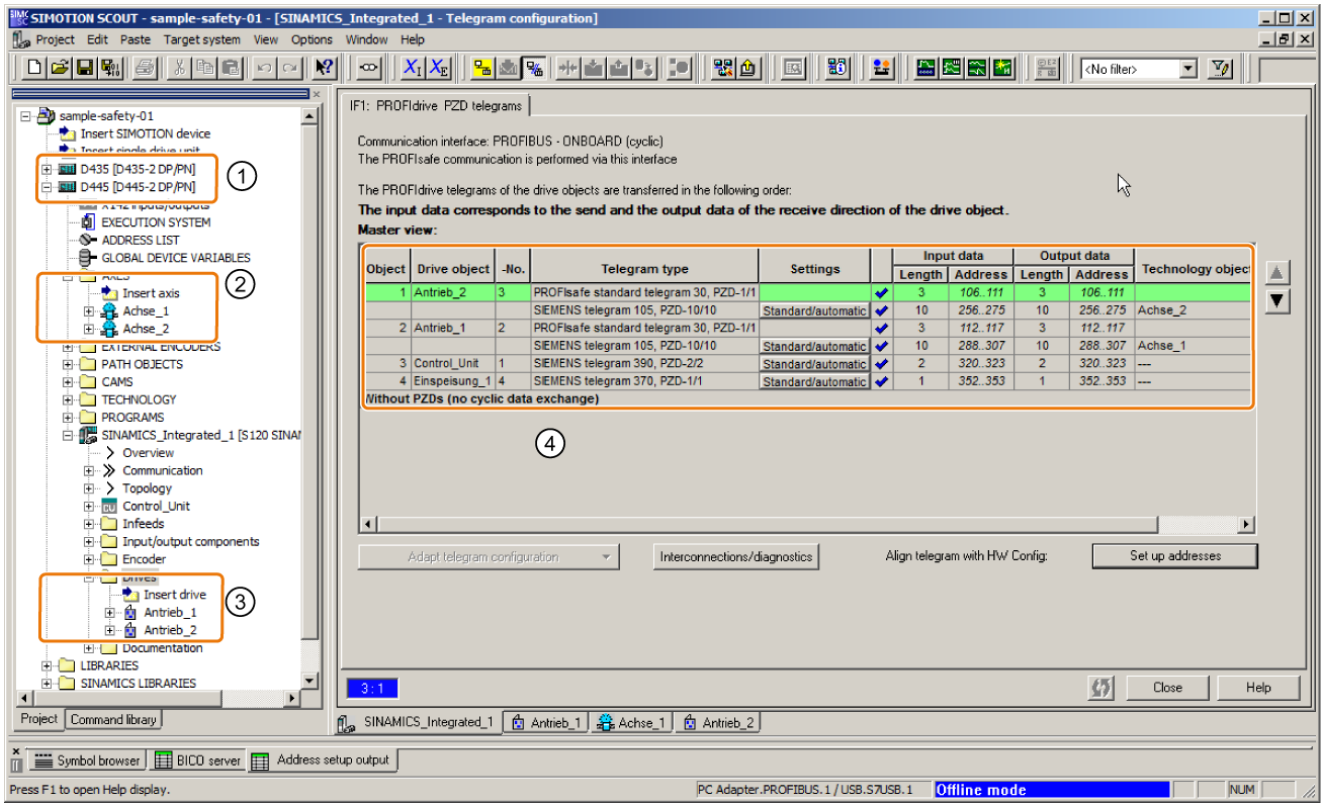


Figure 4-464 Example project for shared iDevice with two SIMOTION D

5. In the project navigator, double-click on a SIMOTION CPU, e.g. D445-2 DP/PN, to open HW Config. Insert a new SIMATIC 300 station with a SIMATIC CPU317F-2 in the project. The F-CPU is located in the same subnet as the SIMOTION CPUs.
6. In HW Config for the SIMOTION D445-2 DP/PN, configure the iDevice (i.e. the iDevice mode and transfer areas) and the isochronous mode. Export and install the iDevice (see also Configuring SIMOTION D for shared iDevice (Page 2378)).
7. When HW Config for the SIMATIC CPU317F-2 is open, add the iDevice to its PROFINET IO system. Copy the iDevice and use "Shared paste" to insert it into the PROFINET IO system of the SIMOTION D435-2 DP/PN. Configure the access to the submodules of the iDevice and the synchronization roles (see also Allocating an iDevice to two IO controllers as a shared device (Page 2380)).
8. **Save and compile** the stations/project.

More details about the configuration process can be found in the following chapters.

Configuring SIMOTION D for shared iDevice

Requirement

A SIMOTION D445-2 DP/PN with two drives on SINAMICS_Integrated and two axes with safety is configured in the example project. The SIMOTION D435-2 DP/PN and the SIMATIC CPU317F-2 are configured in the project as described in the previous step. In the next step, the SIMOTION D445-2 DP/PN will be configured as an I-device.

Configuring and creating a shared I-device

How to configure the SIMOTION D445-2 DP/PN for the creation of the I-device:

1. In the SIMOTION SCOUT project navigator, double-click on the SIMOTION D445-2 DP/PN to open HW Config.
2. Double-click on the **PN interface** (X150) and assign a new **device name** in the Properties dialog box in the **General** tab, e.g. I-Device-Shared.
3. Switch to the **I-Device** tab. Activate the **I-device** option and the additional options **Parameter assignment for the PN interface and its ports on the higher-level IO controller** and **Operate as higher-level shared device.**

4. In this step, you configure the transfer areas (submodules) which are to be shared with the higher-level controllers. Two fail-safe I/Os and two application areas are used in each case.
   - Click **Interconnect fail-safe I/O**. The configured transfer areas are automatically set up for the fail-safe I/O. In the example, two PROFIsafe drives are used. These are **not** operated **isochronously**.
   - Click **New...** and configure the transfer areas for the application area. Two application areas are configured which are operated **isochronously**. The addresses for the transfer areas are suggested by STEP 7. You can also define your own address areas.

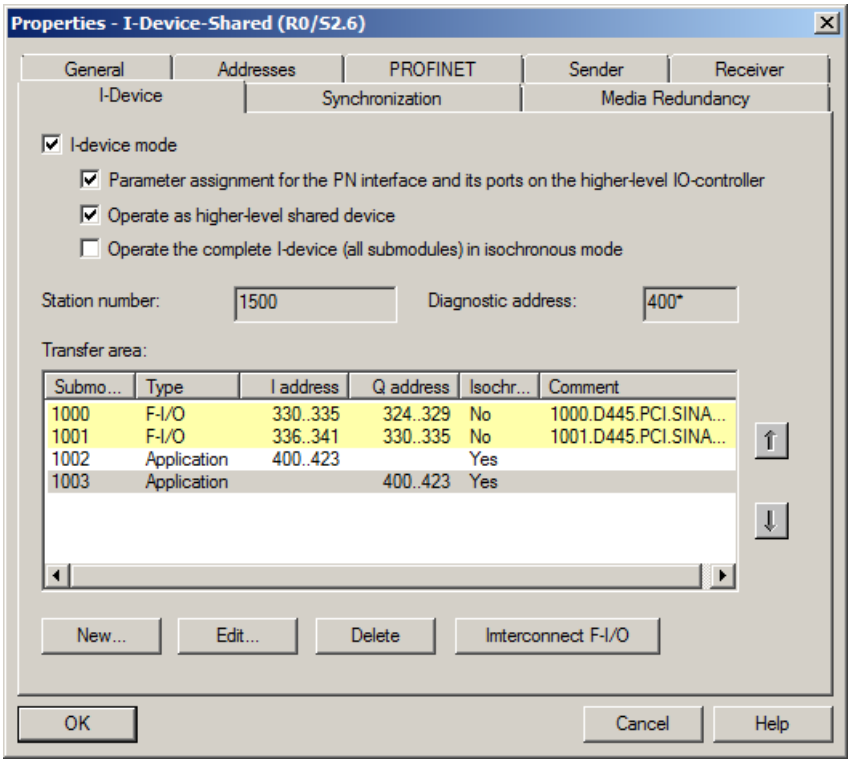


Figure 4-465 Shared I-device transfer areas

5. Activate the option **Use I/O data isochronously to the Servo** for the PN interface in the **Isochronous tasks** tab in the **Properties** of the SIMOTION D445-2 DP/PN and click **OK** to confirm. The I-device is used as a reference clock.
6. In the **PROFINET IO isochronous operation** of the SIMOTION D445-2 DP/PN, select servo as the **Isochronous application**.
7. Select the menu command **Station > Save and Compile**.
8. In the menu, select **Options > Create GSD file for I-device**.
9. Enter a **name** and **comment** for the I-device in the dialog box that appears.
10. Click **Create** and then **Install**. The I-device is displayed under **PROFINET IO > Preconfigured Stations > D445** in the hardware catalog.

In the next step, you will add the I-device to the higher-level IO controllers and configure the access to the transfer areas.

Allocating an iDevice to two IO controllers as a shared device

Requirement

The example project has been completely configured and the GSD file has been created for the iDevice. In the next step, the shared iDevice is configured on the two IO controllers, SIMOTION D435-2 DP/PN and SIMATIC CPU317F-2.

Inserting and configuring a shared iDevice

To configure the shared iDevice, proceed as follows:

1. Open the configured station SIMATIC CPU317F-2 in HW Config.
2. Drag&drop the created iDevice from the hardware catalog (**PROFINET IO > Preconfigured Stations > ...)** to the PN IO system of the SIMATIC CPU. The iDevice is inserted.
3. Double-click on the inserted iDevice. The Properties window is displayed.

Note

The first time you insert a device, you must set up the dialog for the safety password. Enter a password and confirm the dialog. This password will be requested whenever safety-relevant functions are changed.

4. In the **General** tab, enter the **device name** that you assigned earlier on the PN interface of the D445-2 DP/PN before exporting the iDevice (in the example, **iDevice shared**). The device names must be identical in order for communication to be possible.

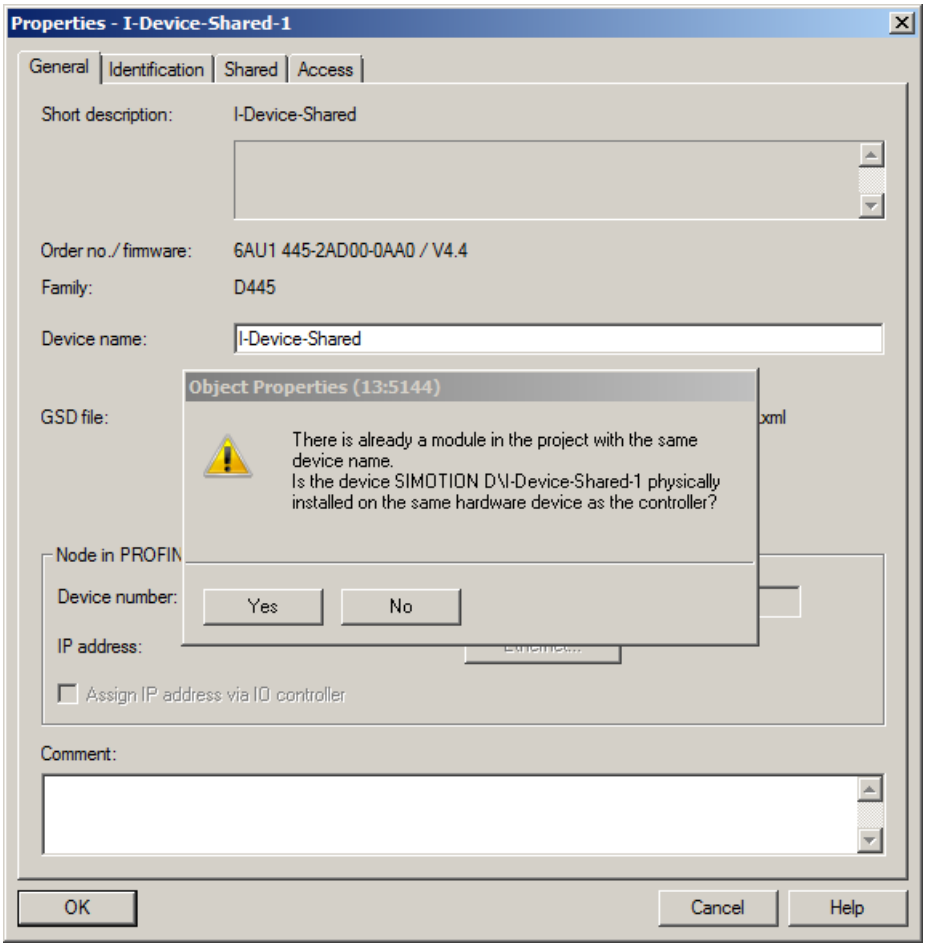


Figure 4-466 Entering a device name for the shared iDevice

Note

A notification informs you that there is already a module in the project with the same device name (IO controller of the iDevice). Click **Yes** to confirm this dialog box. This dialog may appear several times during configuration.

5. In the **Access** tab, enter the access to the submodules of the iDevice. On the F-CPU, the **fail-safe I/O** subslots are configured as **full**. Confirm by clicking **OK** and save and compile the station.

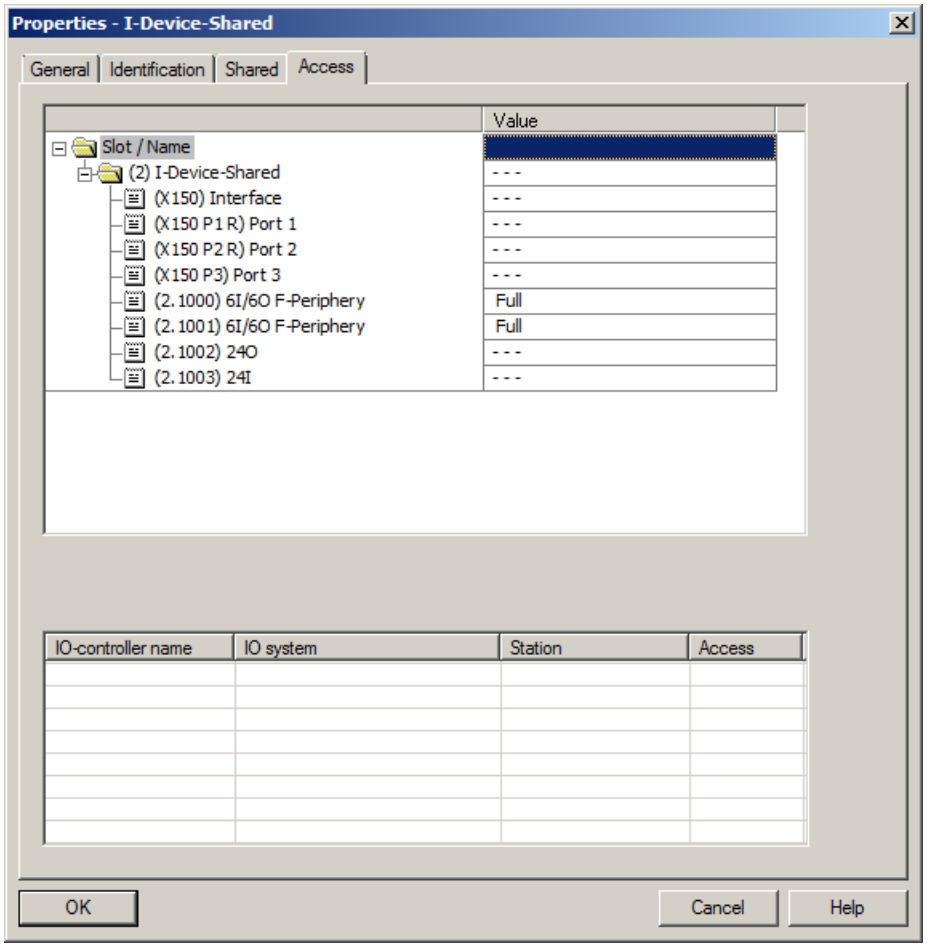


Figure 4-467 Setting access to the submodules for the F-CPU

6. Select the iDevice inserted in HW Config and select **Copy** in the context menu.

7. Switch to HW Config for the SIMOTION D435-2, select the PN IO system, and select **Shared paste** in the context menu. The iDevice is inserted.

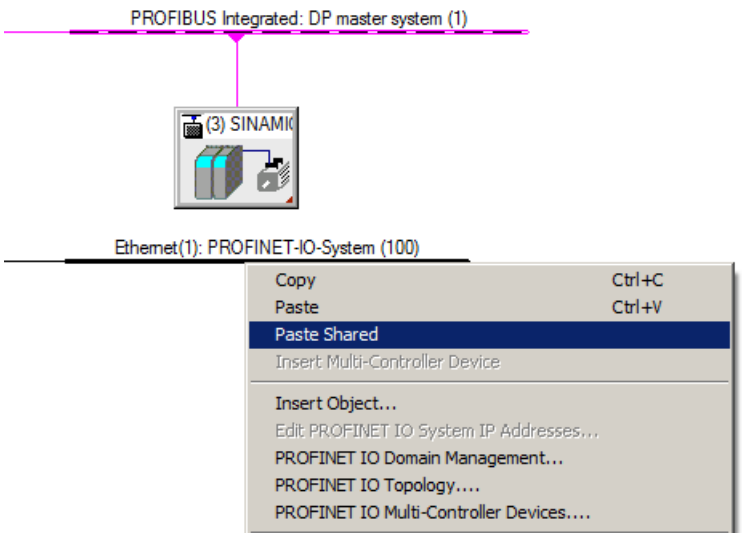


Figure 4-468 Shared paste for iDevice via context menu

8. In the **Access** tab, check the access to the submodules of the iDevice. These are automatically aligned with the HW Config of the F-CPU when the shared iDevice is inserted. The SIMOTION CPU must have full access to all subslots with the exception of the fail-safe I/O. The consistency of the subslots is checked automatically when you select "Save and compile", if the IO controllers are located in the same project.

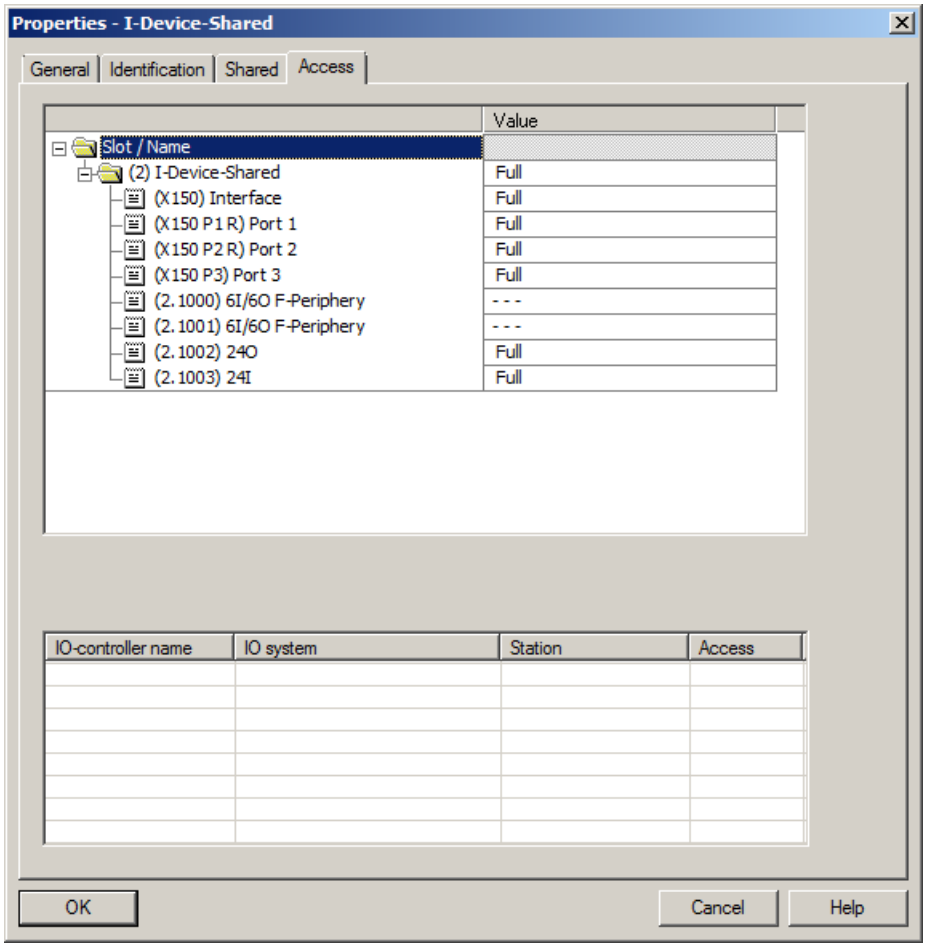


Figure 4-469 Setting access to the submodules for the SIMOTION D

9. On the SIMOTION D435-2, configure the **PN interface** (X150) as the **sync master** and as isochronous to the **servo**. (You should also refer to the chapter Isochronous application (Page 2126)).

10. In the SIMOTION D435-2 station, double-click on **Port 1 (X150 P1)** to interconnect the topology. In the dialog that appears, select a **partner port** of the shared iDevice, e.g. Port 1.

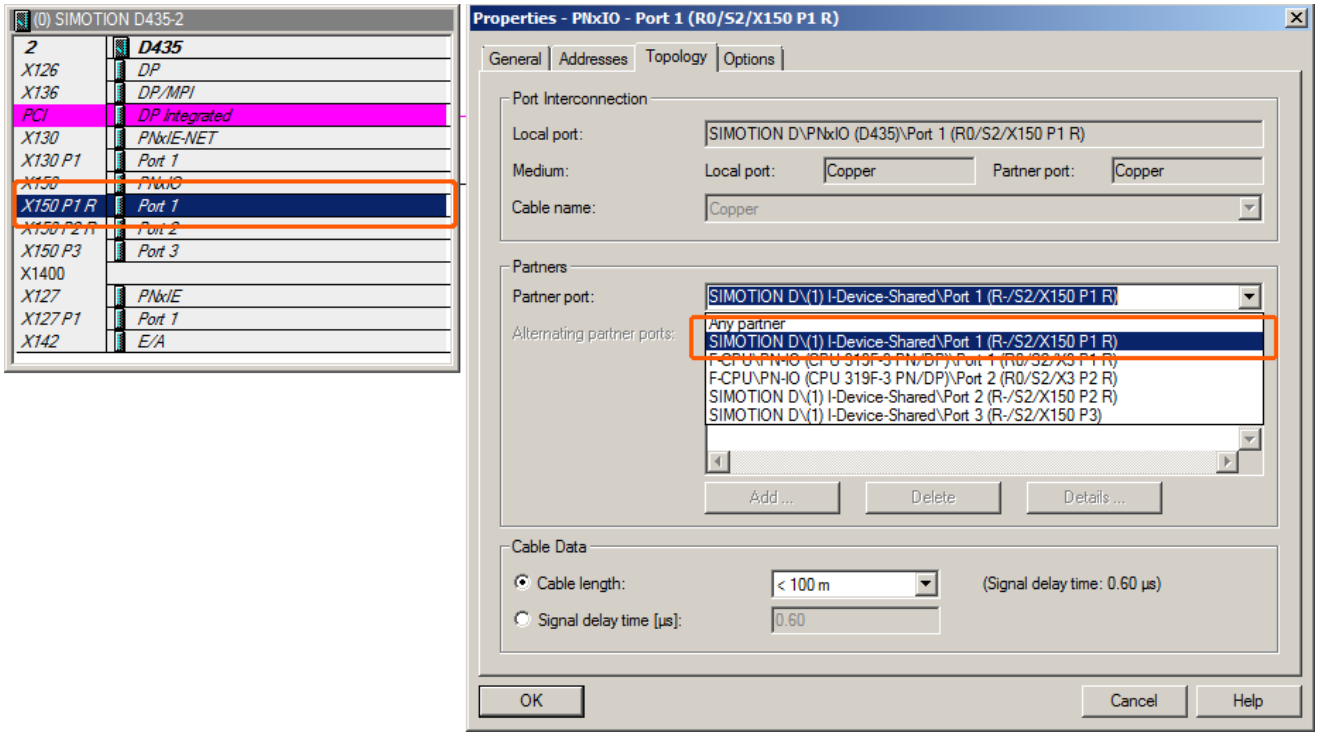


Figure 4-470 Port interconnection on the iDevice

Note

The ports of the SIMOTION D455-2 DP/PN are not displayed in the topology editor, as there is a shared iDevice with the same name in the project. For this reason, the ports must be interconnected using the Properties dialog.

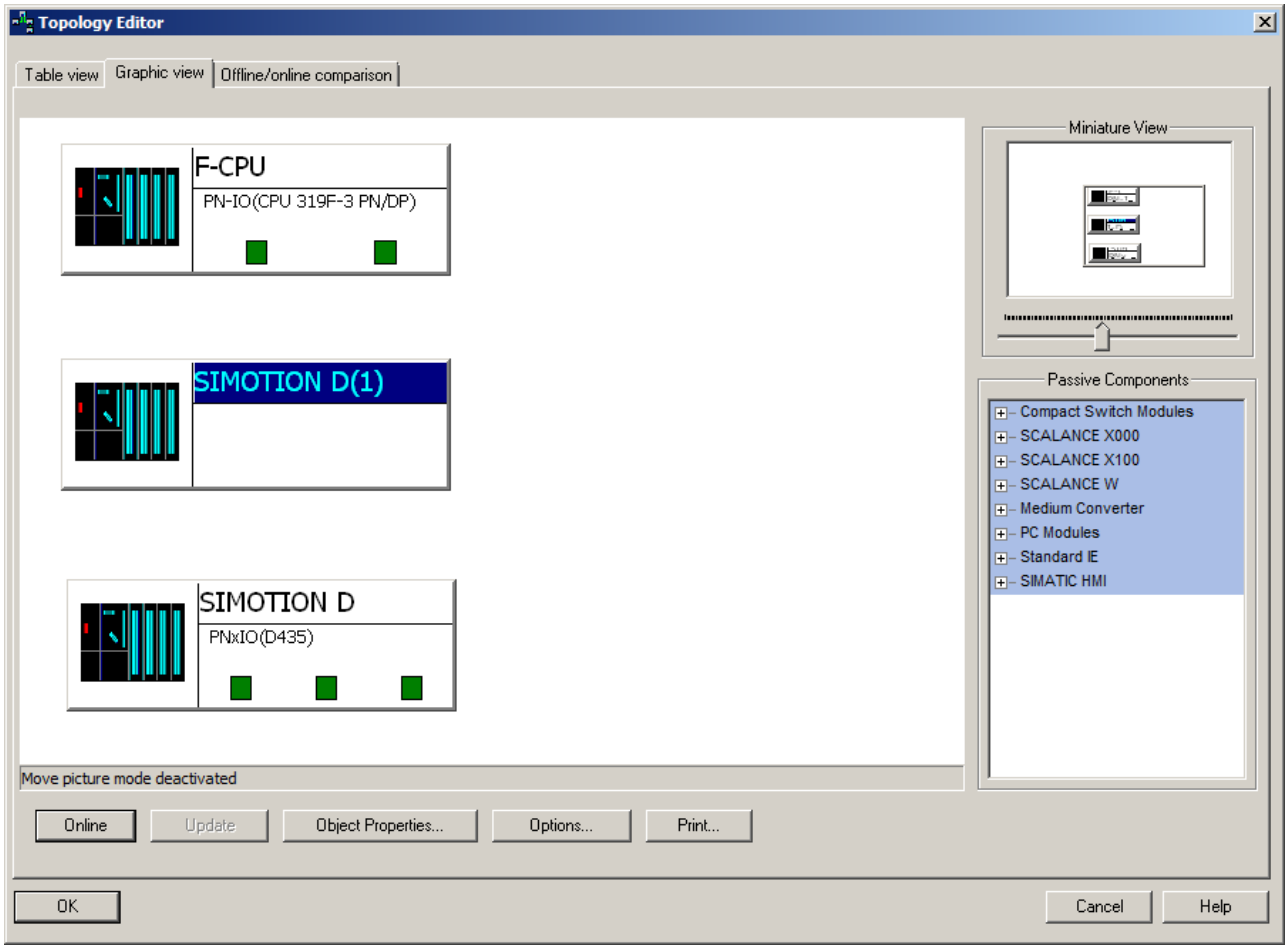


Figure 4-471 Shared iDevice in the topology editor

11. Select the iDevice in the PROFINET IO system of the D435-2 DP/PN and double-click on the PN interface of the iDevice (X150 interface). In the **Properties dialog** that appears, set sync slave and IRT in the **Synchronization** tab. On the **IO Cycle** tab, select **Servo** for the isochronous mode under **Assign IO device isochronously.**

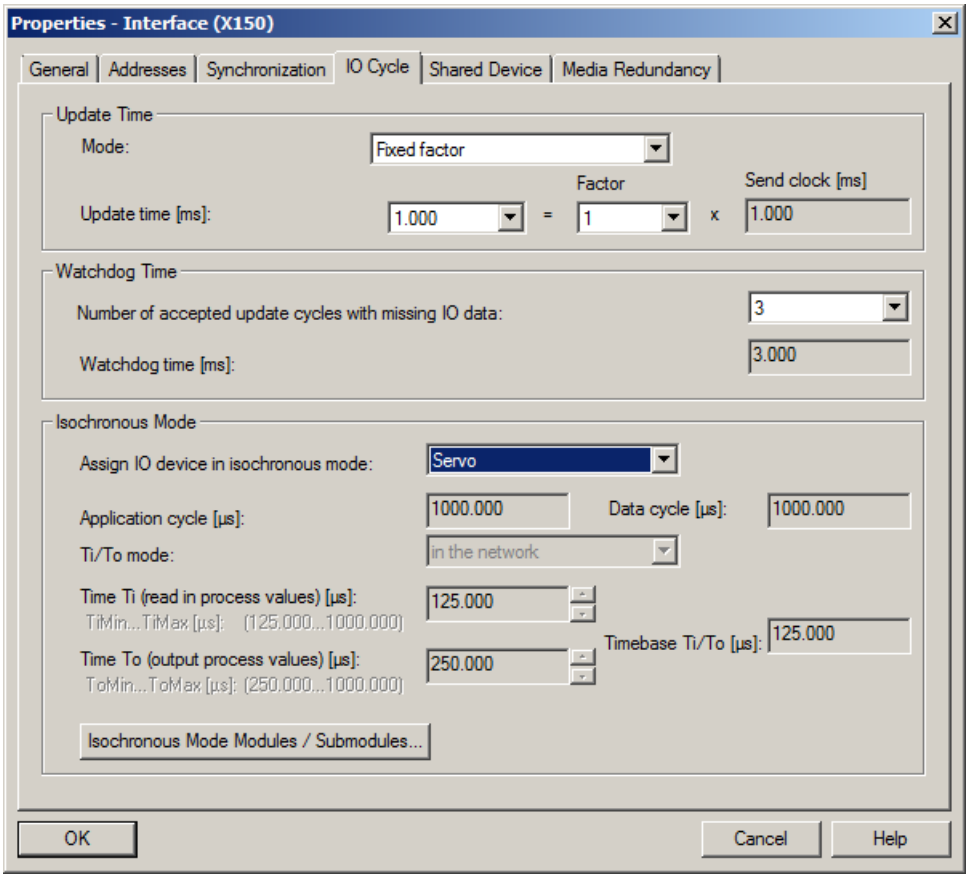


Figure 4-472 Configuring the iDevice as a slave IO device

12. **Save and compile** all stations. Once the stations have been compiled without any errors, configuration is complete.

4.5.9.6 PROFIsafe via PROFIBUS

General information about PROFIsafe on PROFIBUS

Two possibilities for PROFIsafe

There are basically two possible forms of PROFIsafe communication on PROFIBUS:

- **I-slave F-Proxy**
  F-CPU is the DP master in the project and monitors the drives on the lower-level SIMOTION I-slave
- **Failsafe data exchange broadcast**
  SIMOTION is the DP master in the project and the F-CPU monitors the drives as the I-slave.

The procedure for configuring PROFIsafe communication is virtually the same in both cases. The sections below each contain a brief example.

Supported devices and software requirements for PROFIsafe on PROFIBUS

Software packages to be installed on the programming device:

- SIMATIC Manager STEP7 version 5.4 SP2 or higher
- S7 F Configuration Pack Version 5.5 SP3 or higher
- S7 Distributed Safety Programming Version 5.4 SP3 or higher
- SIMOTION SCOUT Version 4.1.1 HF6 or higher
- SINAMICS firmware Version 2.5 or higher

Note

As of SIMOTION firmware 4.1.1 HF10 and SINAMICS firmware 2.5 SP1 HF10, five drives can be configured with a CX32. With earlier firmware versions, a maximum of 4 drives can be configured.

You will find the components suited to PROFIsafe in the *S120 Safety Integrated Function Manual*.

Supported devices

Table 4-244 Device overview

SIMOTION CPU	
Controller Based	C240 C240 PN
PC Based	P350-3 P320-4
Drive-based (blocksize)	D410 DP D410-2 DP D410-2 DP/PN
Drive-based (booksize)	D4x5 D445-1 D4x5-2 DP D4x5-2 DP/PN
SINAMICS drive units	
S120 CX32	CX32 CX32-2
S120	CU320 DP CU320-2 DP CU310 DP CU310-2 DP
S110	CU305 DP

Number of drive axes supported

With PROFIBUS, only 16 drive axes can be used per PROFIBUS interface.

I-slave failsafe proxy

Principles of I-slave failsafe proxy

Short description

Using the I slave F-Proxy you can produce a PROFIsafe configuration with an F host (F-CPU SIMATIC) on PROFIBUS with SIMOTION devices (SIMOTION D, SIMOTION P350, SIMOTION C) for the lower-level drives. Cyclic PROFIsafe data can then be routed to SINAMICS drives on SINAMICS_Integrated / PROFIBUS DP.

A failsafe host communicates with the drives via the I-slave interface and a failsafe proxy of a SIMOTION CPU. The drives may be located on the PROFIBUS DP of the SIMOTION CPU. The SIMOTION CPU's communication segments feature SINAMICS S120/S110 and SINAMICS Integrated/CX32/CX32-2.

Topology for I-slave failsafe proxy for PROFIBUS drive units

Example of topology for I-slave failsafe proxy

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP and this is connected to the F-CPU via PROFIBUS DP.

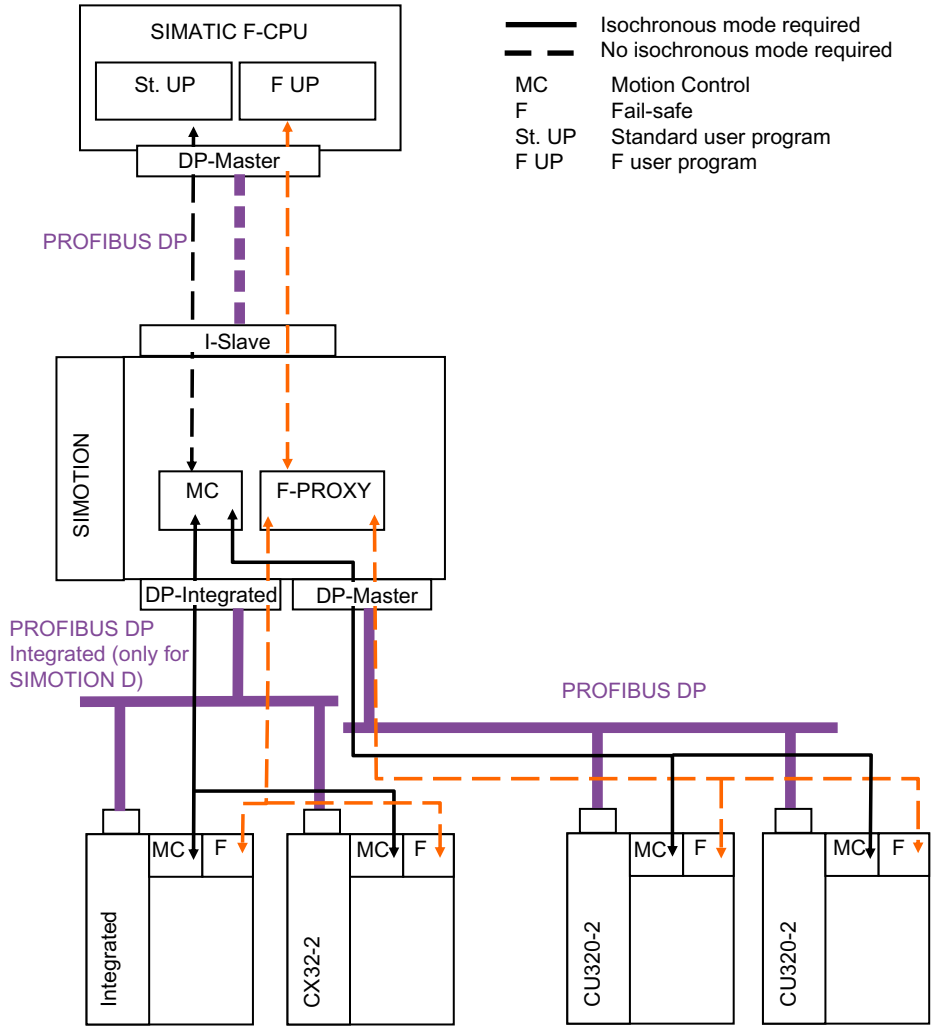


Figure 4-473 Topology for I-slave failsafe proxy for PROFIBUS drive units

PROFIsafe via PROFIBUS when SIMOTION D is used

The next sections deal with the configuration of PROFIsafe communication via PROFIBUS between the integrated drive unit SINAMICS S120 of a SIMOTION D or CX32 and a higher-level SIMATIC F-CPU.

Topology (network view of the project)

The basic topology of the components involved in PROFIsafe communication via PROFIBUS (SIMATIC F-CPU and D4x5 integrated with SINAMICS S120 or CU320) can be found in the previous section.

The drive unit (SINAMICS) and the SIMATIC F-CPU are located on different PROFIBUS subnets. In this case, a PROFIsafe router to SIMOTION D is configured so that the necessary data is copied from one network to the other.

Configuring PROFIsafe communication

The next sections describe the configuration of PROFIsafe communication between a SIMATIC F-CPU and a drive object of an integrated SINAMICS drive unit of a SIMOTION D. The procedure for configuring PROFIsafe communication between a drive unit of a CU320 and a SIMATIC F-CPU is basically the same and is not covered separately.

1. Create an F-CPU (e.g. CPU 317F-2) and a SIMOTION D4x5 controller (with integrated SINAMICS S120) in accordance with the hardware installed.
2. Define a SIMOTION CPU for operation as DP slave and the F-CPU as associated DP master.
3. Configure the SINAMICS drive unit in SIMOTION SCOUT in accordance with your hardware configuration.
4. Then insert a new TO axis and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding telegram will automatically be created (symbolic assignment).
5. Save and compile the project.

6. Create a PROFIsafe slot in the configuration of the SINAMICS drive unit.
   For this purpose, select in tab **IF1 the following: PROFIdrive PZD message frames** - the drive object which is to communicate with the SIMATIC F-CPU via PROFIsafe. Click on the **Adapt message frame configuration** button and select **Add PROFIsafe**.

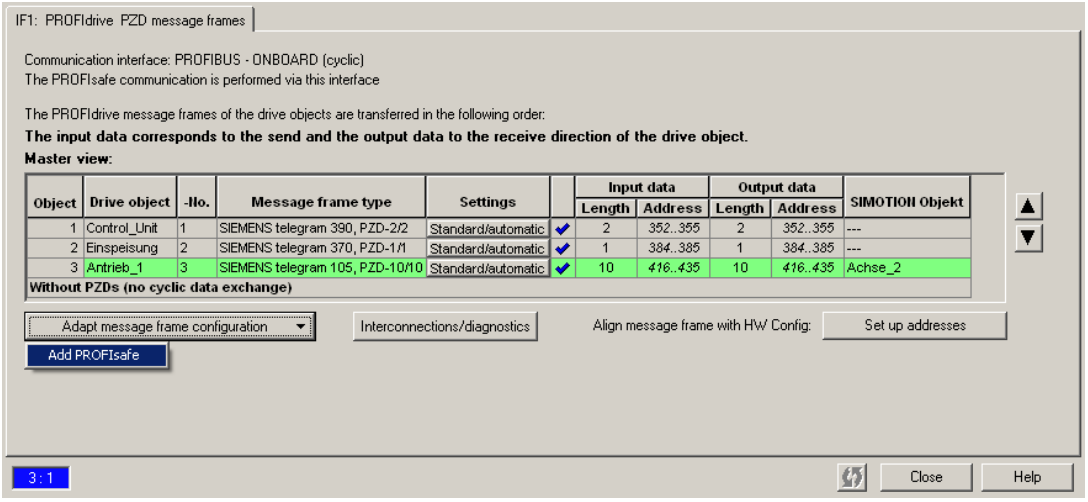


Figure 4-474 Inserting a PROFIsafe slot

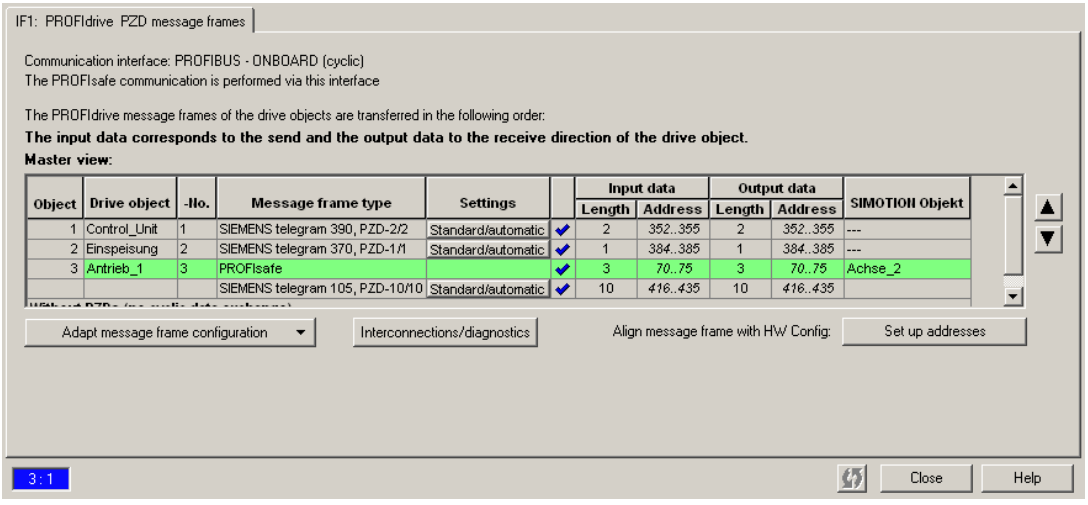


Figure 4-475 PROFIBUS message frame

7. Save and compile the project.
8. Transfer the new PROFIsafe slot to HW Config by clicking on the **Set up address** button.
9. In HW Config for the F-CPU, connect the preconfigured SIMOTION station to the F-CPU (hardware catalog: **PROFIBUS DP > Preconfigured stations** ...).

Note

To configure the SINAMICS Safety Integrated extended functions by means of SIMOTION, the message frames must be extended. To do so, a safety data block is appended to the PROFIdrive actual value message frame. Configuration and parameter assignment of this Safety data block are described in the Function Manual *SIMOTION Motion Control TO Axis electric/hydraulic, external encoder*.

10. You can display the F-communication parameters via the DP slave properties (double-click on SIMOTION I-slave). To do this, go to the **F configuration** tab and click on **New**.
Mode: Displays the communication relationship. F-MS module represents safety-related master-slave communication with SIMOTION.
DP partner (F I/O): SINAMICS drive properties.
Here you can select the relevant PROFIsafe drive via **DP address** or **Address**.
local: Properties of the SIMOTION CPU.
Enter the logical start address for F-communication of the SIMOTION CPU in the "Address" row.
The address space for sending and receiving the safety frames depends on the message frame used and must be located outside the process image of the SIMOTION CPU (>= 64).
Master (safety program): SIMATIC F-CPU properties.
The logical start address for F communication of the SIMATIC F-CPU must be entered here under "Address" (LADDR).
The address space for sending and receiving safety message frames depends on the message frame used and must lie within the process image of the SIMATIC F CPU.
In the SIMATIC F-CPU safety program, this address space can be used to access the PROFIsafe control or status words.
An overview of which message frames are possible for the individual drives can be found in the SIMOTION FAQs.

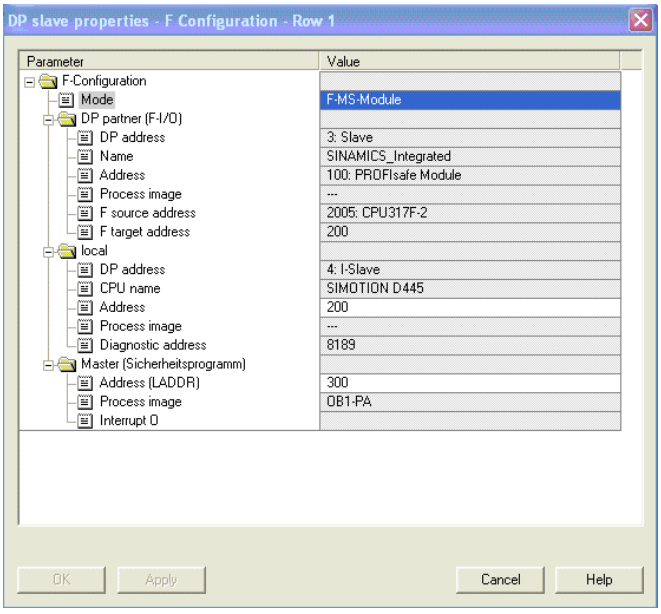


Figure 4-476 Master-slave coupling in PROFIsafe

11. Open the SIMOTION CPU project in HW Config.

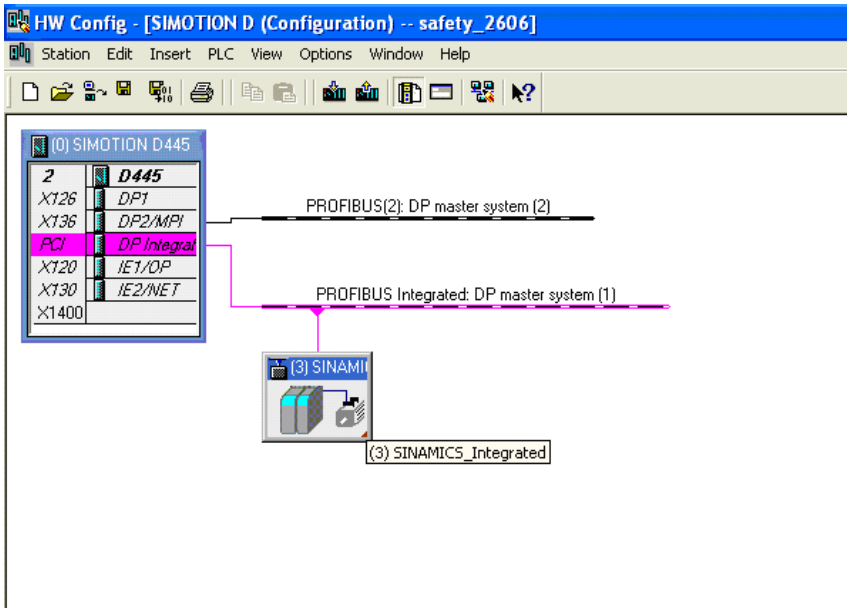


Figure 4-477 SIMOTION D configuration

12. Double-click on the icon of the SINAMICS drive unit and select the **Details** tab in the **Configuration** tab.

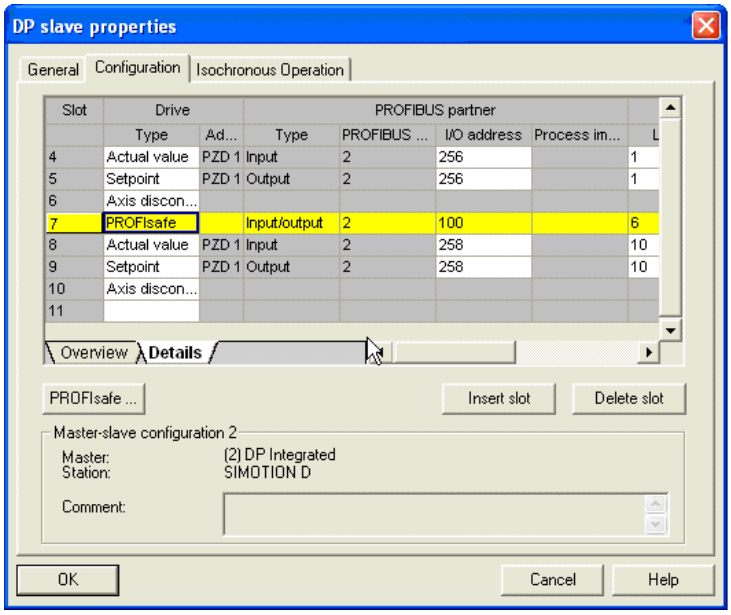


Figure 4-478 PROFIsafe configuration for SINAMICS drive unit

13. Click on the **PROFIsafe...** button and then define the F parameters which are important to F communication. As of STEP7 V5.5, PROFIsafe V2 is used by default.
(If the **PROFIsafe...** button cannot be used, you need to activate it using the **Activate...** button.)
For more information about the fail-safe parameters, see PROFIsafe properties for configuration (Page 2336)

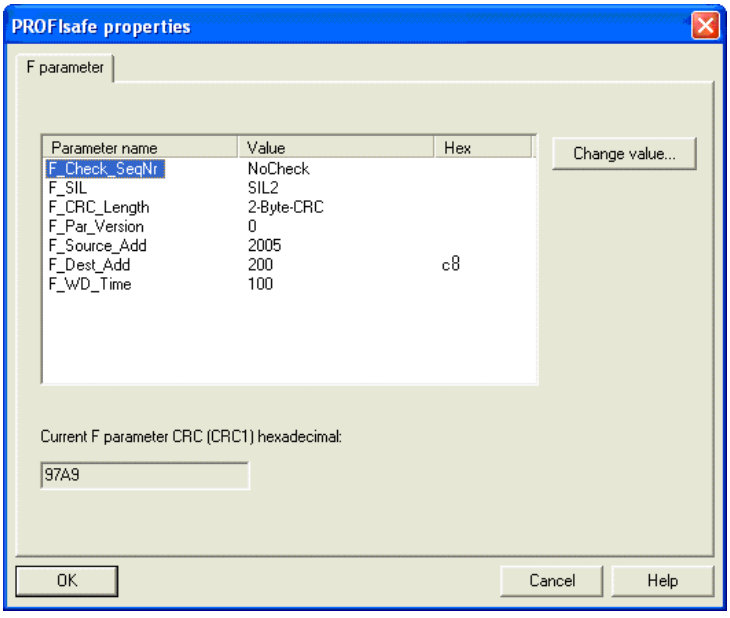


Figure 4-479 PROFIsafe properties (F-parameters)

14. Compile HW Config of the SIMOTION CPU. Compile the F-CPU configuration data in HW Config.

Note

For information about creating a safety program and accessing PROFIsafe useful data (e.g STW and ZSW) within the safety program, refer to the *SIMATIC, S7 Distributed Safety - Configuring and Programming* Programming and Operating Manual.

Safety configuration (online) in the SINAMICS drive

1. Call the configuration for Safety Integrated by selecting "Functions" at the SINAMICS drive entry in the tree structure.
2. Configure Safety Integrated and set to hex representation the F_Dest_Add parameter already defined under the drive's **PROFIsafe address** (p9610/p9810).
3. Finally, perform a POWER ON. The safety configuration is now active in the drive.

Note

For further information on safety configuration, see the SINAMICS S120 Safety Integrated Function Manual.

Failsafe data exchange broadcast

Principles of failsafe data exchange broadcast

Method of operation

SIMOTION CPU is the DP master for failsafe data exchange broadcast. The SIMATIC F-CPU is the DP slave on PROFIBUS DP and controls failsafe communication, e.g. with a CU320 of the SINAMICS S120.

Note

Control for the Safety Integrated functions cannot be routed to the SINAMICS Integrated of the SIMOTION D, Controller Extension CX32/CX32-2, or any other DP network in this constellation.

Topology of failsafe data exchange broadcast via PROFIBUS

Example of topology for failsafe data exchange broadcast for PROFIBUS

The diagram below shows a topology in diagrammatic form in which the Safety drives are connected to the SIMOTION CPU via PROFIBUS DP and the SIMATIC F-CPU is the PROFIBUS I-slave for failsafe communication.

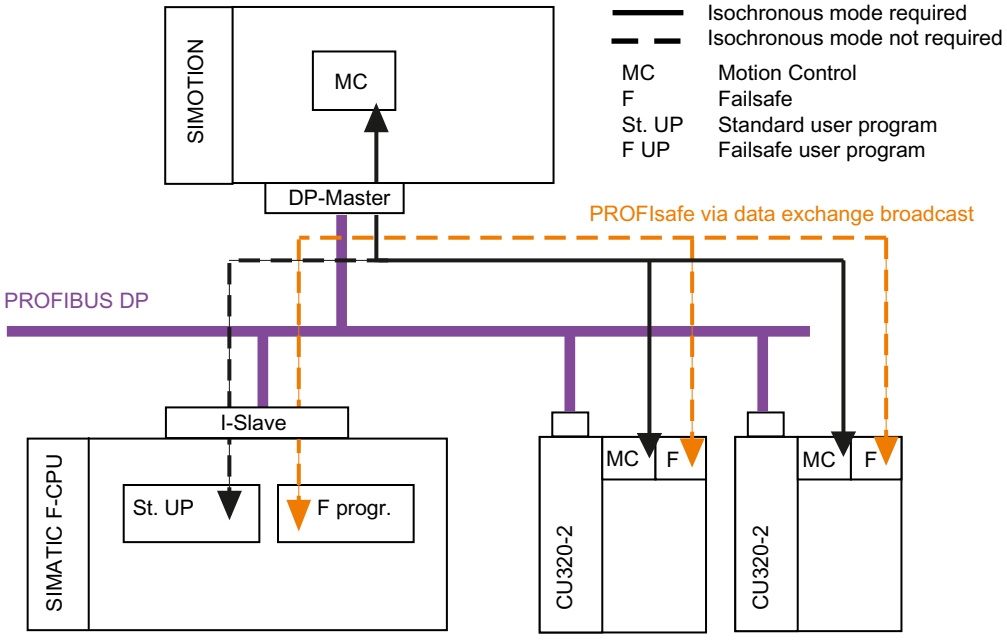


Figure 4-480 Topology of failsafe data exchange broadcast for PROFIBUS drive units

PROFIsafe via PROFIBUS with fail-safe internode data exchange taking the example of SIMOTION D

PROFIsafe communication is to be configured between a SINAMICS S120, SIMOTION D as DP master and SIMATIC F-CPU as I-slave via PROFIBUS.

Topology (network view of the project)

The basic topology of the components involved in PROFIsafe communication via PROFIBUS (SIMATIC F-CPU and D4x5 integrated with SINAMICS S120 or CX32) can be found in the following section:

The drive unit (SINAMICS) and the SIMATIC F-CPU are located in the same PROFIBUS subnet.

Configuring PROFIsafe communication via failsafe data exchange broadcast

1. In HW Config, create an F-CPU (e.g. CPU 317F-2), a SIMOTION D4x5 control and a SINAMICS S120 CU320 in accordance with the hardware installed.
2. Define the desired SIMOTION CPU as the DP master and the connected F-CPU as the associated DP slave.
3. Configure the SINAMICS drive unit in SIMOTION SCOUT in accordance with your hardware configuration.
4. Then insert a new TO axis and run through the axis wizard. In the wizard, interconnect the axis to the corresponding drive object of the S120 and a corresponding message frame will automatically be created (symbolic assignment).
5. Save and compile the project.
6. Create a PROFIsafe slot in the configuration of the SINAMICS drive unit.
   For this purpose, select in tab IF1 the following: PROFIdrive PZD message frames - the drive object which is to communicate with the SIMATIC F-CPU via PROFIsafe. Click on the **Adapt message frame configuration** button and select **Add PROFIsafe**.

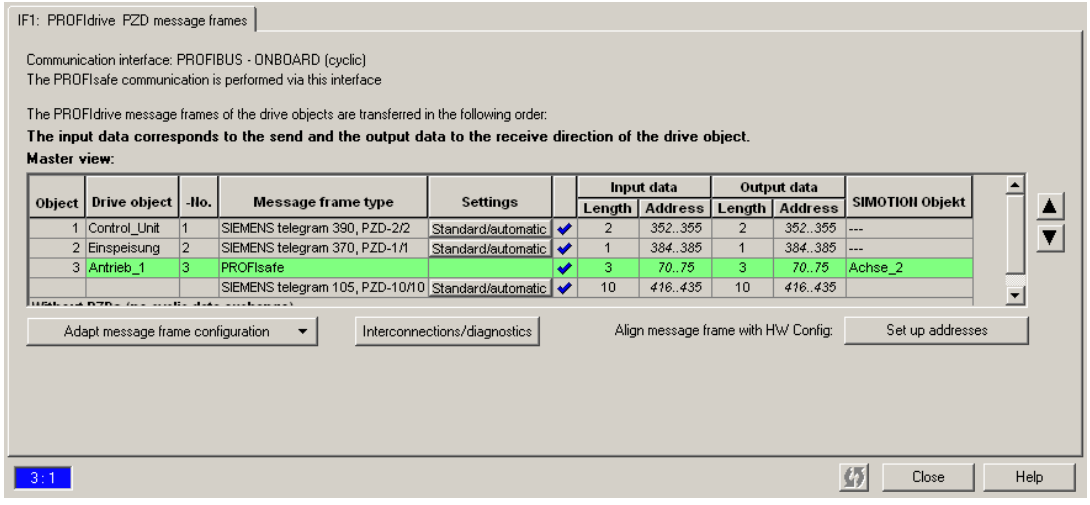


Figure 4-481 PROFIBUS message frame

7. Save and compile the project.
8. Transfer the new PROFIsafe slot to HW Config by clicking on the **Set up address** button.

9. In HW Config for the SIMOTION station, connect the preconfigured F-CPU to the SIMOTION station.(HW catalog: **PROFIBUS DP > Preconfigured stations...).**

10. The F-communication parameters are displayed in the DP slave (F-CPU) properties, tab **F Configuration**.
Mode: Displays the communication relationship. F-DX modules must be selected for data exchange broadcast. This represents a safety-related I-slave-slave relationship.
DP partner (F I/O): SINAMICS drive properties.
Here you can select the relevant PROFIsafe drive via DP address or Address.
local (safety program): SIMATIC F-CPU properties.
The logical start address for F communication of the SIMATIC F-CPU must be entered here under **Address (LADDR)**. The address space for sending and receiving safety message frames depends on the message frame used and must lie within the process image of the SIMATIC F CPU.
In the SIMATIC F-CPU safety program, this address space can be used to access the PROFIsafe control or status words.
Master address: Properties of the SIMOTION CPU.
Enter the logical start address for F communication of the SIMOTION CPU under **Input address** .
The address area for sending and receiving safety messages depends on the message frame used and must be located outside the process image of the process image of the SIMOTION CPU (>=64).
An overview of which message frames are possible for the individual drives can be found in the SIMOTION FAQs.

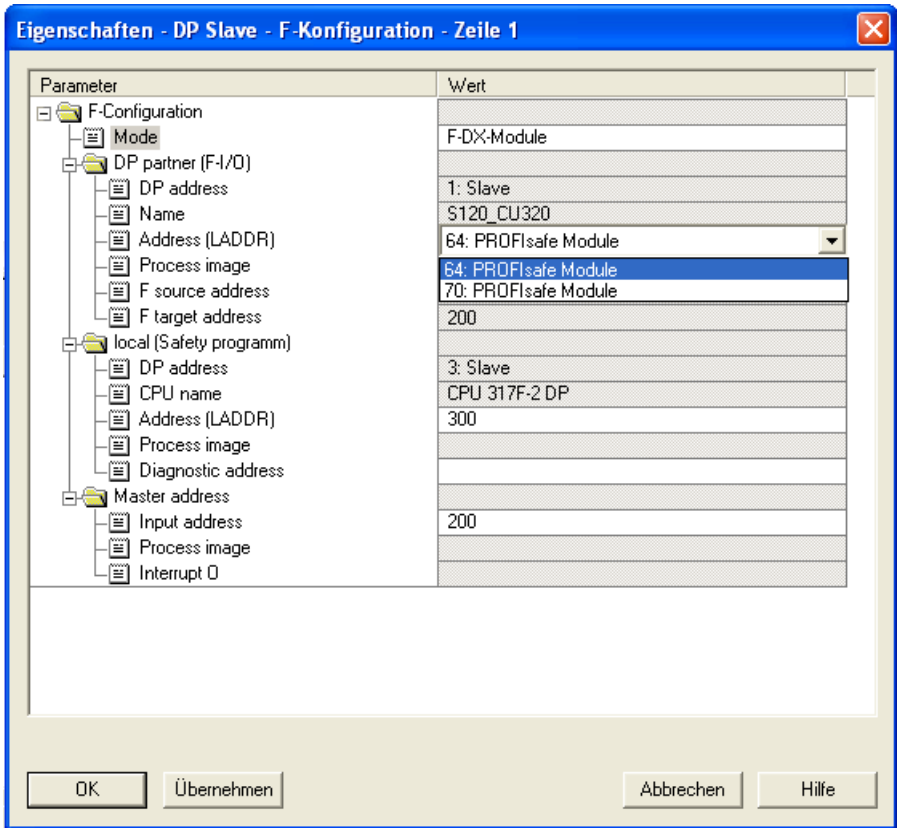


Figure 4-482 Properties of failsafe configuration for data exchange broadcast

11. Open the SIMOTION CPU project in HW Config.

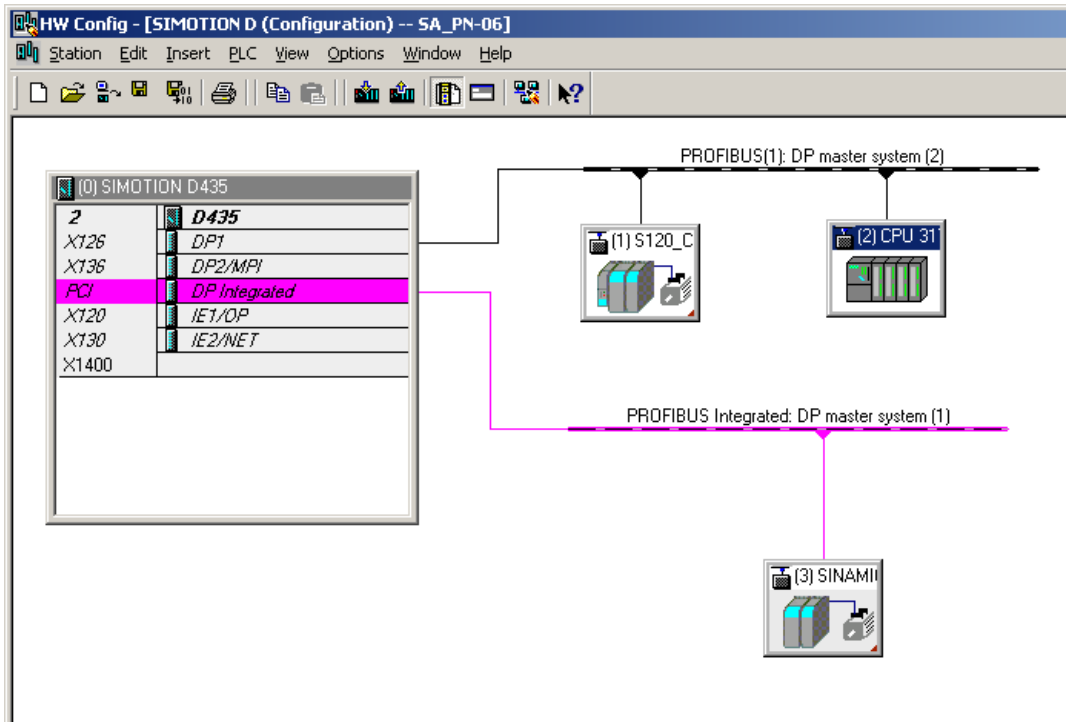


Figure 4-483 HW Config failsafe data exchange broadcast

12. Double-click on the icon of the SINAMICS drive unit and select the **Details** tab in the **Configuration** tab. Click the **PROFIsafe...** button to specify the relevant F Parameters for failsafe communication.
(If the **PROFIsafe...** button cannot be used, you need to activate it using the **Activate...** button.)
For more information about the failsafe parameters, see PROFIsafe properties for configuration (Page 2336)

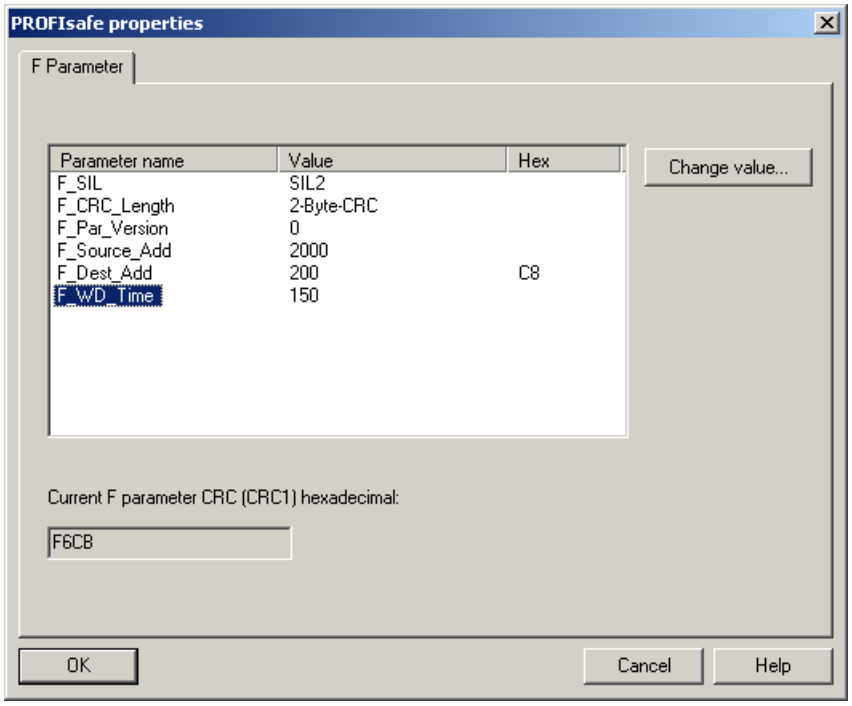


Figure 4-484 PROFIsafe properties failsafe data exchange broadcast mode V1

13. Compile HW Config of the SIMOTION CPU. Compile the F-CPU configuration data in HW Config.

Safety configuration (online) in the SINAMICS drive

1. Call the configuration for Safety Integrated by selecting the **Functions** entry in the structure tree where it says SINAMICS drive.
2. Configure Safety Integrated and set to hex representation the F_Dest_Add parameter already defined under the drive's PROFIsafe address (p9610/p9810).
3. Finally, perform a POWER ON. The safety configuration is now active in the drive.

Note

For further information on Safety configuration, see the *SINAMICS S120 Safety Integrated* Function Manual.

4.5.9.7 PROFIsafe configuration - acceptance test and reports

Acceptance tests and acceptance reports

Once configuring and commissioning has been successfully completed, an acceptance test of the drive safety functions must be carried out. This involves checking correct parameter assignment of the safety functions. The tests carried out are documented in reports.

Note

When carrying out the acceptance test for PROFIsafe communication, please note the information provided in the *SIMATIC, S7 Distributed Safety - Configuring and Programming* Programming and Operating Manual and the *SINAMICS S120 Safety Integrated* Function Manual of the drive used.

4.5.9.8 Additional information on SIMOTION and PROFIsafe

Description

Additional information on the subject of PROFIsafe is available in the following documents:

- For information on how to connect an axis to a SINAMICS drive with Safety Integrated, please refer to the TO Axis / External Encoder Function Manual.
- For information on how to configure a SINAMICS S120 drive or SINAMICS S110 drive with Safety Integrated, please refer to the following:
  - *SINAMICS S120 Function Manual*
  - *SINAMICS S120 Safety Integrated Function Manual*
  - *SINAMICS S110 Function Manual*.

4.5.9.9 Exporting/importing drive objects (DO)

Description

You can replace drive objects (DO) in an existing SIMOTION project. This means that you can replace deleted drive objects with new imported drive objects. You might need to do this, for example, if you want to replace motors with others. Whether replacement is successful or not depends on the configuration (for example, configured PROFIsafe, telegrams).

Requirement

Both the deleted drive object and the new imported drive object must meet the following conditions:

- The drive objects are the same version
- The number of deleted and the number of imported drive objects is identical
- The length of the telegrams is identical. This means, the activated technology data must be identical (TO axis) if they result in a telegram extension.
- F addresses are identical if PROFIsafe (F-Proxy) is used.

This is how you replace drive objects

1. Open the export list of the Control Unit whose drive you want to replace and navigate to parameter P978. This is where the sequence of drive objects is stored.
2. Make a note of the settings of parameter P978 or export the parameter.

   Note

   Alternatively, you can open the telegram configuration and make a mental or written note of the sequence of drive objects.

3. Delete the drive objects via the shortcut menu in the project navigator.
4. Import the new drive objects via the shortcut menu in the project navigator. The new drive objects are inserted automatically and are assigned new address ranges and positions in the telegram. You will have to change these in the next step.

NOTICE
Do not compile or align telegrams After completing this step, **do not** compile (store and compile) the project and **do not** align the telegrams.

5. Open the expert list of the CU and navigate to parameter P978.
6. Enter the sequence of drive objects you previously noted or import the previous values. Alternatively, you can open the telegram configuration and place the imported drive objects in the correct sequence using the arrow buttons on the right.
7. If symbolic assignment is active, go into the address list and open the shortcut menu in the top left field (next to the name) and run the "assignment setup" function.
8. Save and compile the project.

Note about replacing drive objects

If the above requirements for deleting and importing drive objects are not met, the project cannot be made consistent merely by adapting the sequence of drive objects. In this case, you will have to adapt the telegrams and addresses manually. If an F-Proxy is configured, the relevant drive, if it has a red background, must be deleted and created new. It may be necessary to adapt the PROFIsafe configuration on the drive.

4.5.10 PROFIdrive

4.5.10.1 Why profiles?

Profiles used in automation technology define certain characteristics, behaviors, and interfaces for devices, device groups or whole systems which specify their main and unique properties. Only devices with manufacturer-independent profiles can behave in exactly the same way on a fieldbus and thus fully exploit the advantages of a fieldbus for the user.

Profiles are specifications defined by manufacturers and users for certain characteristics, performance features, behaviors, and interfaces of devices and systems. They aim to ensure a certain degree of interoperability of devices and systems on a bus which are part of the same product family due to "profile-compliant" development.

Different types of profiles can be distinguished such as so-called application profiles (general or specific) and system profiles.

- Application profiles mainly refer to devices, in this case drives, and contain an agreed selection of bus communication methods as well as specific device applications.
- System profiles describe system classes and include the master functionality, program interfaces and integration methods.

PROFIdrive

PROFIdrive is a device-specific (Drives) application profile. PROFIdrive essentially describes a "standard interface" for drives to PROFIBUS and PROFINET. It contains a detailed description of how the communication functions "data exchange broadcast", "equidistance" and "isochronous operation" are used in drive applications. In addition, it specifies all device characteristics which influence interfaces connected to a controller over PROFIBUS or PROFINET. This also includes the State machine (sequential control), the encoder interface, the normalization of values, the definition of standard telegrams, the access to drive parameters, the drive diagnostics, etc.

The PROFIdrive profile supports both central as well as distributed motion control concepts.

The basic philosophy: – Keep it simple –

The PROFIdrive profile tries to keep the drive interface as simple as possible and free from technology functions. This philosophy ensures that reference models as well as the functionality and performance of the PROFIBUS/PROFIDRIVE masters have no or very little effect on the drive interface.

4.5.10.2 PROFIdrive overview

The PROFIdrive Profile

The PROFIdrive profile defines the device behavior and the access procedure to drive data for drives on PROFIBUS and on PROFINET, from simple frequency converters up to high performance servo controllers.

PROFIdrive is split into a general part and a bus-specific part. The following properties are defined in the general part.

- Base model
- Parameter model
- Application model

The following assignments are made in the bus-specific part:

- PROFIdrive to PROFIBUS
- PROFIdrive to PROFINET

Details of where to find a precise description of the PROFIdrive profile are given below.

Literature note

PROFIdrive profile

Profile Drive Technology – PROFIdrive Profile
Version V4.1, May 2006, Order Number 3.172
PROFIBUS User Organization e. V.
Haid-und-Neu-Straße 7, D-76131 Karlsruhe
http://www.profibus.com

Standards

Standard IEC 61800
IEC 61800-7 Part 203, Part 303

4.5.10.3 PROFIdrive base/parameter model

Description

The PROFIdrive base model describes an automation system in terms of a number of devices and their interrelationships (application interfaces, parameter access).

Table 4-245 The base model distinguishes between the following device classes:

PROFIdrive device class	PROFIBUS DP	PROFINET IO
Controller (higher-level controller or host of the automation system)	DP master Class 1	IO controller
Peripheral device (P device)	DP Slave (I-slaves)	IO device
Supervisor (engineering station)	DP master Class 2	IO supervisor

PROFIdrive device classes

In the picture below, the possible communication relationships of the PROFIdrive device classes are presented.

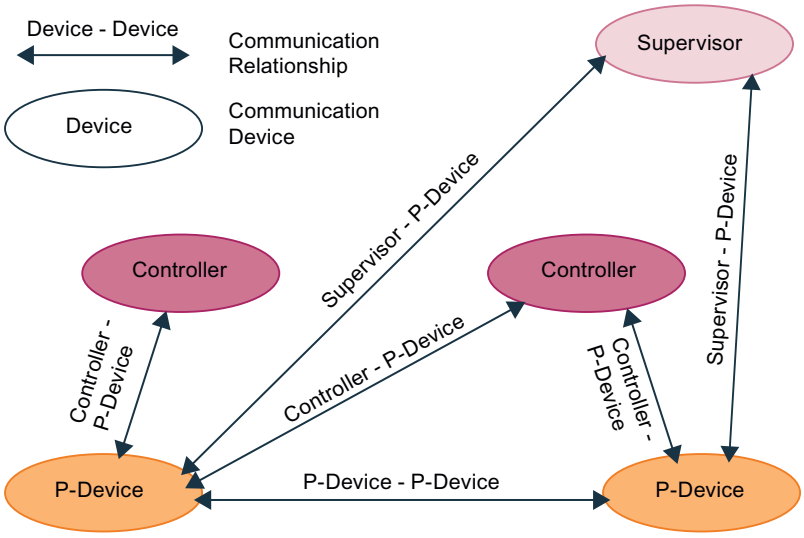


Figure 4-485 Communication between device classes according to PROFIdrive

Example of a PROFIdrive automation concept

The graphic below shows a typical automation concept.

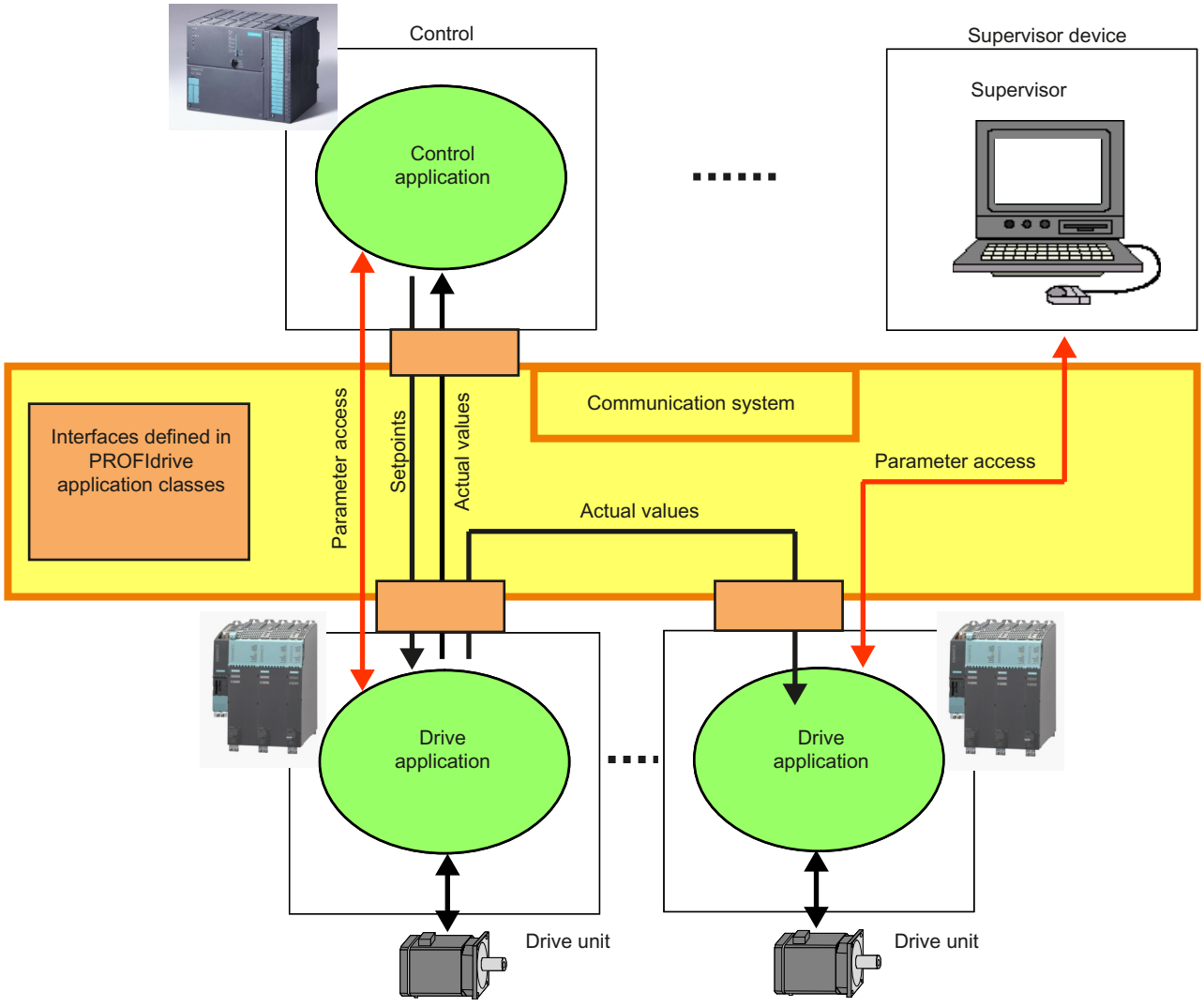


Figure 4-486 Automation concept

Communication services

Two communication services are defined in the PROFIdrive profile; namely, cyclic data exchange and acyclic data exchange.

- Cyclic data exchange via a cyclic data channel
  Motion control systems need cyclically updated data during operation for open- and closed-loop control purposes. This data must be sent to the drive units in the form of setpoints or transmitted from the drive units in the form of actual values, via the communications system. Transmission of this data is usually time-critical.
- Acyclic data exchange via an acyclic data channel
  In addition to cyclic data exchange, there is an acyclic parameter channel for exchanging parameters between the control/supervisor and drive units. Access to this data is not time-critical.

The graphic below shows the data model and data flow in the P device.

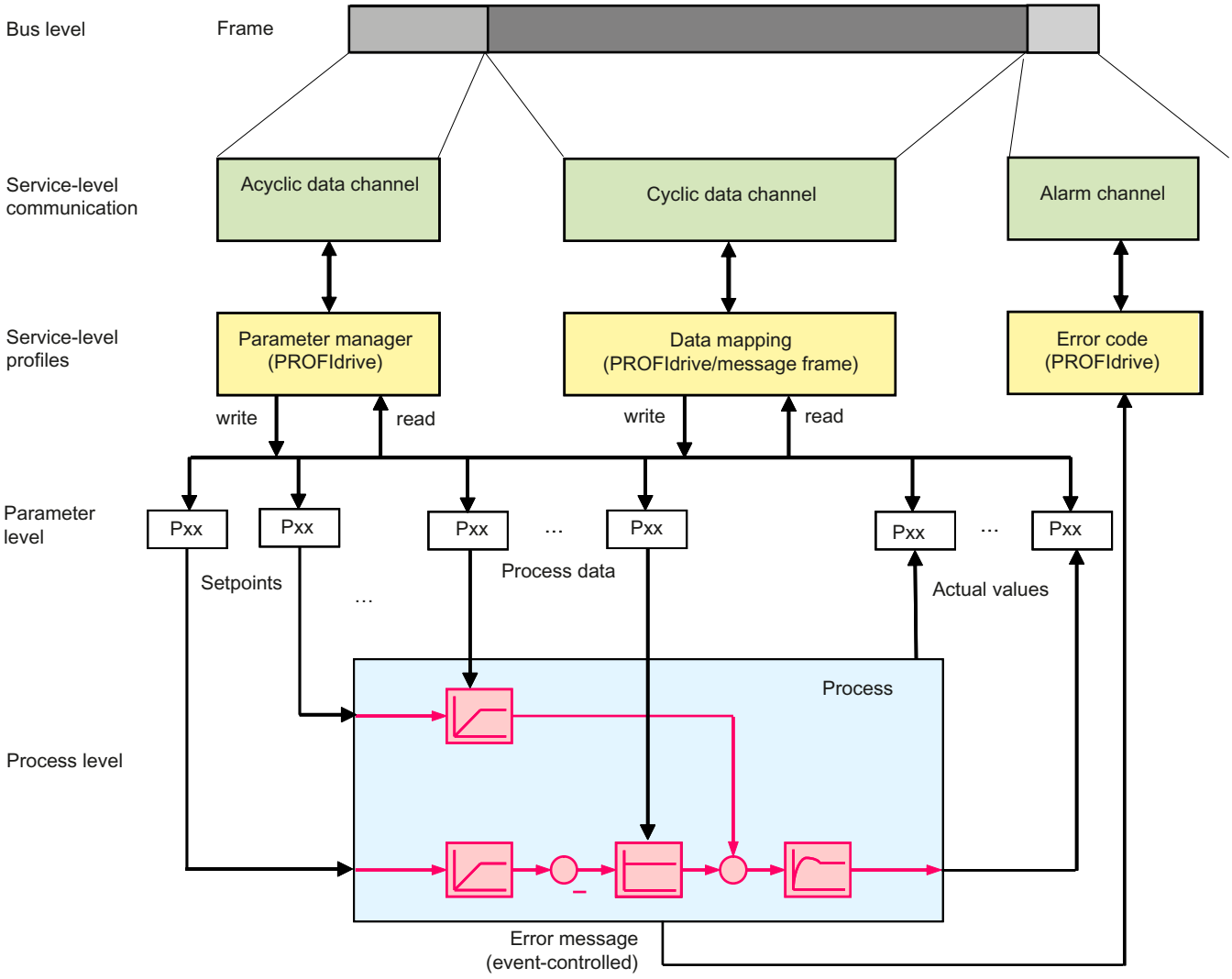


Figure 4-487 Data model and data flow in the P device

Alarms and error messages

Alarms are output on an event-driven basis, and show the occurrence and expiry of error states.

Parameter model

The parameter model in the PROFIdrive profile makes a distinction between profile parameters and manufacturer-specific parameters:

- Profile parameters are defined for objects derived from the device model of the PROFIdrive profile. These may include general functions such as drive identification, fault buffer, or drive control, for example. These parameters are the same for all drives. The profile-specific parameters are in the range 900–999 and 60,000–65,535.
- All other parameters are manufacturer-specific. The parameters are defined by the interface for the application process, rather than by the profile. The manufacturer-specific parameters are in the range 1–699 and 1,000–59,999.

Access to a parameter's elements (values, parameter descriptions, text elements) essentially works on an acyclic basis (with the exception of G120 drives, which can exchange data cyclically with PIV). An independent request/response data structure is defined for this purpose.

PROFIdrive defines a device model based on function modules which cooperate in the device and generate the intelligence of the drive system.

Objects are assigned to these modules that are described in the profile and defined in terms of their function. The overall functionality of a drive is therefore described through the sum of its parameters.

PROFIdrive defines the access mechanism to the parameter (PROFIdrive parameter channel) and approx. 70 profile-specific parameters; e.g. for fault buffer, drive control, and device identification.

All other parameters are manufacturer-specific which gives drive manufacturers great flexibility with respect to implementing control functions. The elements of a parameter are accessed acyclically using what is known as "Base Mode Parameter Access".

PROFIdrive uses DP V0, DP V1, and the DP V2 expansions for PROFIBUS, and the slave data exchange broadcast and isochronous operation functions contained within them as the communication protocol.

PROFIdrive for PROFINET contains the functions PROFINET RT, PROFINET (isochronous mode) and IO controller–IO device communication.

See also

Specifications for PROFIBUS and PROFINET IO (Page 2422)

4.5.10.4 Segmentation in application classes

Integration of drives in automation solutions

The integration of drives into automation solutions depends strongly upon the drive task. To cover the extensive range of drive applications from the most simple frequency converter up to highly dynamic, synchronized multi-axis systems with a single profile, PROFIdrive defines six application categories which can be applied to most drive applications.

Table 4-246 Application classes

Category	Drive
Category 1	Standard drives (such as pumps, fans, agitators, etc.); implemented in SIMOTION and SINAMICS
Category 2	Standard drives with technology functions
Category 3	Positioning drives; implemented in SIMOTION and SINAMICS
Category 4	Motion control drives with central, higher-level motion control intelligence and the "Dynamic Servo Control" position control concept; implemented in SIMOTION and SINAMICS
Category 5	Motion control drives with central, higher-level motion control intelligence and position setpoint interface
Category 6	Motion control drives with distributed, motion control intelligence integrated in the drives

Application classes

Application category 4 is the most important for highly dynamic and highly complex motion control tasks. This application category describes in detail the master/slave relationship between the controller and the drives which are connected to each other over PROFIBUS and PROFINET. The position control loop is closed via the bus. The synchronization of the position control cycles in the control and in the speed controllers in the drive requires a clock synchronization (PROFIBUS DP-V2 or PROFINET with IRT).

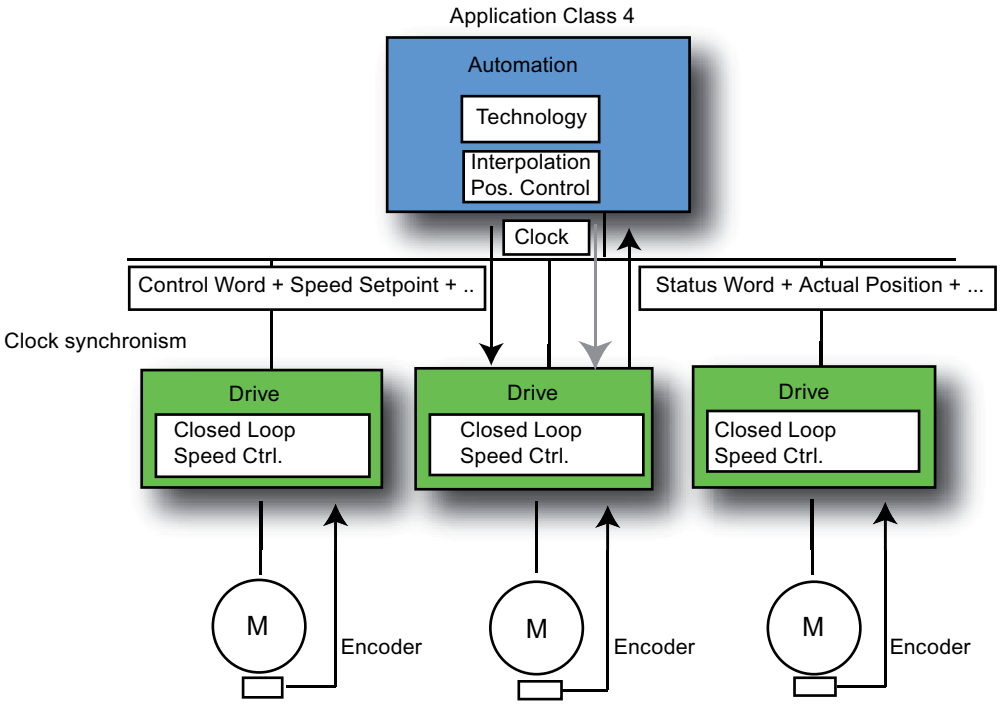


Figure 4-488 Application classes

Using the DSC function (Dynamic Servo Control), the dynamic interference immunity of the position control loop is increased considerably for mechanically rigid systems. This is achieved using an additional minimum dead time position feedback in the drive. As a result, short position control cycle clocks (e.g. for servo 125 µs in the SINAMICS S) are enabled and transmission dead time on the bus then only has an effect on the control behavior of the axis.

See also

Specifications for PROFIBUS and PROFINET IO (Page 2422)

4.5.10.5 PROFIdrive-specific data types

Description

A range of data types have been defined for the purpose of using communication that is compliant with PROFIdrive. You will find detailed information on this in the following standards:

- IEC 61800-7-203
- IEC 61800-7-303
- IEC 61158-5

These standards contain detailed descriptions of the data types. The most important data types are listed below. Data types are provided by the _readDriveParameterDescription function, for example.

Note

For the communication with SINAMICS, you will have to use the AnyType_to_BigByteArray() or BigByteArray_to_AnyType() SIMOTION system functions to perform a type conversion for different data types (normalized value N2, N4; normalized value X2, X4; fixed-point value E2, C4 and time constants T2 and T4).

PROFIdrive profile-specific and standard data types

Data types used in the PROFIdrive profile	Definition	Coding (dec.)
Standard data types		
Boolean	Boolean (IEC 61158-5)	1
Integer8	Integer8 (IEC 61158-5)	2
Integer16	Integer16 (IEC 61158-5)	3
Integer32	Integer32 (IEC 61158-5)	4
Unsigned8	Unsigned8 (IEC 61158-5)	5
Unsigned16	Unsigned16 (IEC 61158-5)	6
Unsigned32	Unsigned32 (IEC 61158-5)	7
FloatingPoint32	Float32 (IEC 61158-5)	8
FloatingPoint64	Float64 (IEC 61158-5)	15
VisibleString	VisibleString (IEC 61158-5)	9
OctetString	OctetString (IEC 61158-5)	10
TimeOfDay (with date indication)	TimeOfDay (IEC 61158-5)	11
TimeDifference	TimeDifference (IEC 61158-5)	12
Date	Date (IEC 61158-5)	13
TimeOfDay (without data indication)	TimeOfDay (IEC 61158-5)	52
TimeDifference (with data indication)	TimeDifference (IEC 61158-5)	53
TimeDifference (without data indication)	TimeDifference (IEC 61158-5)	54
Specific data types	**See below for description**	
N2 (normalized value (16-bit))		113
N4 (normalized value (32-bit))		114
V2 bit sequence		115
L2 nibble		116
R2 reciprocal time constant		117
T2 time constant (16-bit)		118
T4 time constant (32-bit)		119
D2 time constant		120
E2 fixed-point value (16-bit)		121
C4 fixed-point value (32-bit)		122

Data types used in the PROFIdrive profile	Definition	Coding (dec.)
X2 normalized value, variable (16-bit)		123
X4 normalized value, variable (32-bit)		124

Normalized value N2, N4

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2^{14} (0x4,000) for N2, or 2^{30} (0x40,000,000) for N4. The length is 2 or 4 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values N2, N4	Resolution N2, N4	Cod. N2, N4 (dec.)	Octet	Bit							
8	7	6	5	4	3	2	1				
-200% ≤ i ≤ (200-2^{-14})%	2^{-14} = 0,0061%	113	1	SN	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
			2	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
-200% ≤ i ≤ (200-2^{30})%	$2^{-30,}$ = 9,3 * 10^{-8}%	114	3	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}
			4	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}	2^{-29}	2^{-30}

Normalized value X2, X4 (example X = 12/28)

Linear normalized value, 0% corresponds to 0 (0x0), 100% corresponds to 2^{x}. These structures are identical to N2 and N4, except that normalization is variable. Normalization can be determined from the parameter descriptions. The length is 2 or 4 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values X2, X4	Resolution X2, X4	Cod. X2, X4 (dec.)	Octet	Bit							
8	7	6	5	4	3	2	1				
-800% ≤ i ≤ 800-2^{-12})%	2^{-12}	123	1	SN	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
			2	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
-800% ≤ i ≤ 800-2^{-28})%	2^{-28}	124	3	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
			4	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}

Fixed-point value E2

Linear fixed-point value with seven places after the decimal point. 0 corresponds to 0 (0x0), 128 corresponds to 2^{14} (0x4,000). The length is 2 octets.

Coding

Represented in two's complement; MSB (most significant bit) is the first bit after the sign bit (SN) of the first octet.

- SN = 0; positive numbers with 0
- SN = 1; negative numbers

Range of values E2	Resolution	Cod. (dec.)	Octet	Bit							
8	7	6	5	4	3	2	1				
$-256+2^{-7} \le i \le 256-2^{-7}$	$2^{-7} = 0{,}0078125$	121	1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
			2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}

Fixed-point value C4

Linear fixed-point value with four places after the decimal point. 0 corresponds to 0 (0x0), 0.0001 corresponds to 2^0 (0x0000 0001).

Coding

As with Integer32, the weighting of the bits has been reduced by a factor of 10,000.

Range of values	Resolution	Coding (dec.)	Length
$-214{,}748.3648 \le i \le 214{,}748.3648$	$10^{-4} = 0{,}0001$	122	4 octets

Bit sequence V2

Bit sequence for checking and representing application functions. 16 Boolean variables are combined to form 2 octets.

Range of values	Resolution	Cod. (dec.)	Octet	Bit							
8	7	6	5	4	3	2	1				
		115	1	15	14	13	12	11	10	9	8
			2	7	6	5	4	3	2	1	0

Nibble (half-byte) L2

Four associated bits make up a nibble. Four nibbles are represented by two octets.

Coding

Range of values	Resolution	Cod. (dec.)	Octet	Bit				
8	7	6	5	4	3	2	1	
-	-	116	1	Nibble 3				Nibble 2
			2	Nibble 1				Nibble 0

Overview of time constants T2, T4, D2, and R2

Note

The values for the time parameters of types D2, T2, T4, and R2 always relate to the specified, constant sampling time T_a. The associated sampling time (parameter p0962) is required to interpret the internal value.

Time constants T2 and T4

Time data as a multiple of sampling time T_a. Interpreted value = internal value * T_a

Coding

- T2: As with Unsigned16, with a restricted range of values of $0 \le x \le 32767$.
  When interpreted, internal values that fall outside this range of values are set to 0.
- T4: As with Unsigned32

Range of values	Resolution	Coding (dec.)	Length
$0 \le i \le 32{,}767 * T_a$	T_a	118	2 octets
$0 \le i \le 4{,}294{,}967{,}295 * T_a$	T_a	119	4 octets

Time constant D2

Time data as a fraction of the constant sampling time T_a. Interpreted value = internal value * T_a/16384

Coding

- T2: As with Unsigned16, with a restricted range of values of $0 \le x \le 32767$.
  When interpreted, internal values that fall outside this range of values are set to 0.

Range of values	Resolution	Coding (dec.)	Length
$0 \le i \le (2\text{-}2^{-14}) * T_a$	T_a	120	2 octets

Time constant R2

Time data as a reciprocal multiple of the constant sampling time T_a. Interpreted value = 16384 * T_a/internal value

Coding

- T2: As with Unsigned16, with a restricted range of values of $1 \leq x \leq 16,384$.
  When interpreted, internal values that fall outside this range of values are set to 16,384.

Range of values	Resolution	Coding (dec.)	Length
$1 * Ta \leq i \leq 16,384 * T_a$	T_a	117	2 octets

Note

Further data types:

Standard PROFIBUS/PROFINET data types (only available in English) (Page 2451)

Profile-specific PROFIBUS/PROFINET data types (only available in English) (Page 2461)

See also

Parameter request/response data set (Page 2419)

4.5.10.6 Acyclic communication (Base Mode Parameter Access)

Acyclic communication

Description

PROFIdrive drive devices are supplied with control signals and setpoints by the controller and return status signals and actual values.

These signals are transferred cyclically between the controller and the drive.

In addition, PROFIdrive drive devices recognize parameters in which additional, necessary data are held; such as error codes, warnings, controller parameters, motor data, etc. This data is not usually transmitted cyclically, but "acyclically" only when required. Commands for the drive can also be transferred using parameter accesses.

The reading/writing of parameters from PROFIdrive units is always performed acyclically, using what is known as "Base Mode Parameter Access". "Base Mode Parameter Access" can be used with both PROFIBUS and PROFINET. For differences between PROFIBUS and PROFINET, please refer to Specifications for PROFIBUS and PROFINET IO (Page 2422).

This service is defined and provided by PROFIdrive, and it can be used in parallel to the cyclic communication on the relevant bus. The PROFIdrive profile specifies precisely how this basic mechanism is used for read/write access to parameters of a PROFIdrive-compliant drive.

Note

The DPV1lib significantly simplifies configuration of acyclic communication. You can find it on the SIMOTION DVD U&A under **Applications > Cross-Sector Applications > Drive Communication**.

Reading and writing parameters with Base Mode Parameter Access

Description

Base Mode Parameter Access, whose structure is defined in the PROFIdrive profile, is always used for communicating the writing/reading parameters for PROFIdrive units such as SINAMICS S120. The structure is also contained, for example, in the **Acyclic communication** section of the SINAMICS S120 Function Manual.

Under this arrangement, parameter access always consists of:

- Write request ("Write data set")
- Read request ("Read data set")

This sequence must be observed, irrespective of whether read or write access is involved.

A "Write data record" is used to transfer the parameter job (request); for example, read parameter x. A "Read data record" is used to fetch the response for this parameter job (value of parameter x).

The following figure shows the method of operation of the parameter channel on reading/ writing the data set.

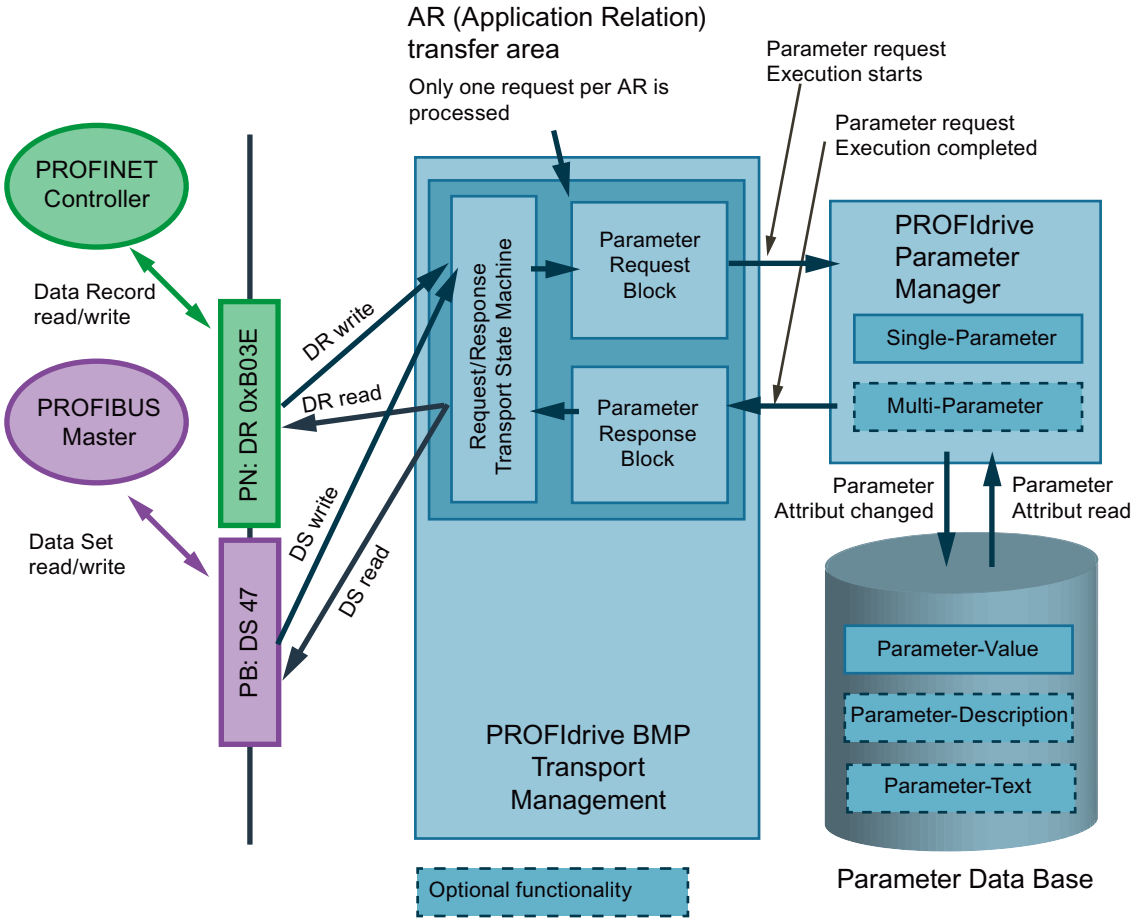


Figure 4-489 Function of the PROFIdrive parameter channel

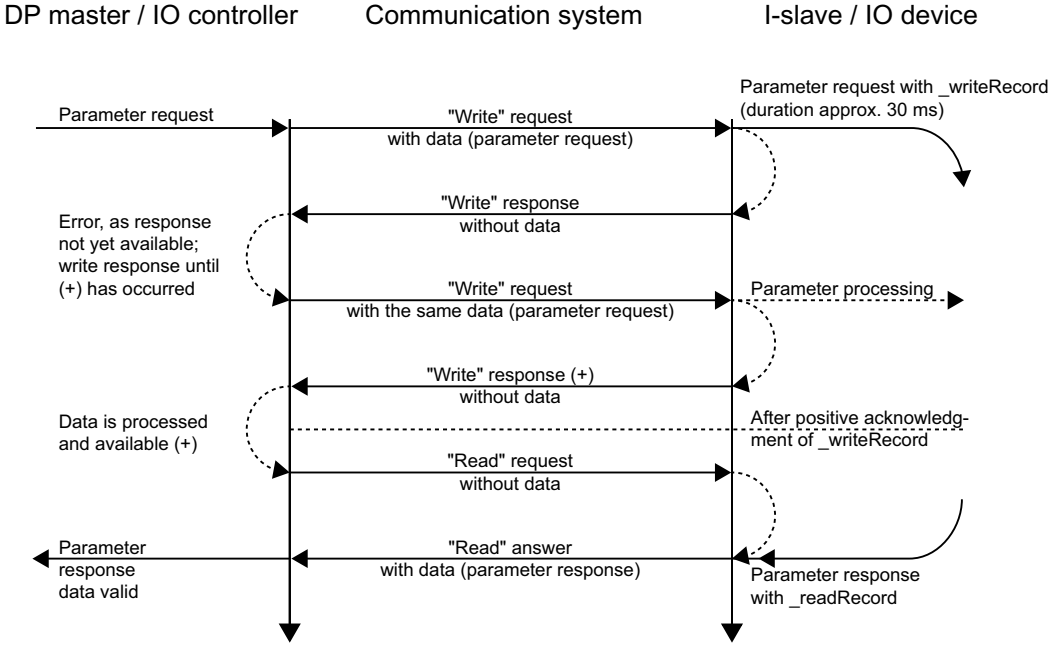


Figure 4-490 Reading and writing acyclically

The figure **Reading and writing acyclically** shows that both "Write data set" and "Read data set" consist of the following elements:

- Request
- Response

Writing parameter records

Initially, data (P request/response data set) is transmitted to the job structure for the purpose of writing (one or more) parameter values. The data is subsequently transmitted with "Write data set" using _writeRecord. Repeated instances of "Write data set" (_writeRecord without data) can be used to monitor the status until a positive acknowledgment is given. With a "Read data set" request (_readRecord), execution is requested in the target device.

Note

An instance of "Write data set" without data enables the status of "Write data set" with data to be determined until the positive acknowledgment is given.

If "Write data set" is successfully completed, this only signifies that the data set has been transmitted via the communication path without any errors; it does not signify that the action has been executed without any errors in the target device. This is checked by the subsequent "Read data set" request.

Reading parameter records

For the purpose of reading parameter values, the data block for determining which parameter(s) is/are to be read is created first. This data record is transferred to the drive as a request using "Write data record" (_writeRecord). A subsequent "Read data set" (_readRecord) as a request then returns the requested values one time.

The processes are also represented above in the form of a diagram.

The PROFIdrive profile specifies how data larger than one byte is to be transferred. The so-called "Big Endian" format, the highest value parts are transferred first, is used:

WORD		High Byte (Byte 1)	Low Byte (Byte 2)
DOUBLE WORD	High Word	High Byte (Byte 1)	Low Byte (Byte 2)
	Low Word	High Byte (Byte 3)	Low Byte (Byte 4)

WORD and DWORD representation in Big Endian format

Since the control has a different internal data representation in certain cases, an explicit conversion must be performed when grouping and evaluating the data in the P request/ response data block (data set 47).

A conversion may be required for SIMOTION, see Program example (Page 2446).

See also

Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 2434)

Parameter request/response data set (Page 2419)

Parameter request/response data set

Structure of the P request/response data set

This always consists of:

- Header (job identifier (request/change), job reference (job ID), number of parameters in the job, target axis / drive object ID)
- Parameter details (number of elements / indices, attribute, parameter number, subindex)
- Parameter values (only with job identifier "Change")

The data transmitted for a WRITE or READ request has the following structure.

	Job parameters	**Byte n+1**	**Byte**	**Offset**
Header	Request Header	Request Reference	RequestID	0
		Axis	Number of parameters	2
Details	1. parameter	Attribute	Number of elements	4
		Parameter number		6
		Subindex		8
	...			
	nth parameter	Attribute	Number of elements	
		Parameter number		
		Subindex		
Only write values for parameter (change)	1. parameter value(s)	Format	Number of values	
		Values		
	...			
	nth parameter value(s)	Format	Number of values	
		Values		
		...		

Structure after Base Mode Parameter Access - Parameter Request

The following structure is defined for the subsequent Parameter Response. This is received after a request with _readRecord (request).

	Parameter response	**Byte n+1**	**Byte**	**Offset**
	Request Header	Request Reference mirrored	RequestID mirrored or error	0
		Axis no./DO ID mirrored	Number of parameters	2

	Parameter response	Byte n+1	Byte	Offset
Only read values for parameter (request) / Error values for negative response only	1. parameter value(s)	Format	Number of values	4
		Values or error codes		6
		...		
	...			
Only read values for parameter (request) / Error values for negative response only	nth parameter value(s)	Format	Number of values	
		Values or error codes		
		...		

Structure after Base Mode Parameter Access - Parameter Response

The exact coding of the individual parts of the data structure can be obtained from the PROFIdrive profile or the SINAMICS S120 Function Manual. The assignment of "Request" and the related answer "Response" for "Write data record" or "Read data record" using the "Request Reference" job reference is important.

Request Reference

The "Request Reference" is used for the assignment of the write request to the following read request, and the response that follows thereafter, because the control can, in principle, process several actions (as many as 8) in parallel for different target devices using the same fieldbus.

Description of fields in parameter job and response

Field	Data type	Values	Comments
Job reference	Unsigned8	0x01 to 0xFF	
	Unique identification of the job/response pair for the master. The master changes the job reference with each new job. The slave mirrors the job reference in its response.		
Job identifier	Unsigned8	0x01 0x02	Read job Write job
	Specifies the type of job. In the case of a write job, the changes are made in a volatile memory (RAM). A save operation is needed in order to transfer the modified data to the non-volatile memory (p0971, p0977).		

Field	Data type	Values	Comments
Response ID	Unsigned8	0x01 0x02 0x81 0x82	Read job(+) Write job(+) Read job(-) Write job(-)
	Mirrors the job identifier and specifies whether job execution was positive or negative. Negative means: Cannot execute part or all of job. The error values are transferred instead of the values for each subresponse.		
Drive object number	Unsigned8	0x01 to 0xFE	Number
	Setting for the drive object number on a drive unit with more than one drive object. Different drive objects with separate parameter number ranges can be accessed over the same DPV1 connection.		
Number of parameters	Unsigned8	0x01 to 0x27	No. 1 to 39 Limited by DPV1 telegram length
	Defines the number of adjoining areas for the parameter address and/or parameter value for multi-parameter jobs. The number of parameters = 1 for single jobs.		
Attribute	Unsigned8	0x10 0x20 0x30	Value Description Text (not implemented in the case of SINAMICS)
	Type of parameter element accessed		
Number of elements	Unsigned8	0x00 0x01 ... 0xEA	Special function No. 1 to 234 Limited by DPV1 telegram length
	Number of array elements accessed		
Parameter number	Unsigned16	0x0001 to 0xFFFF	No. 1 to 65535
	Addresses the parameter accessed		
Subindex	Unsigned16	0x0000 to 0xFFFE	No. 0 to 65534
	Addresses the first array element of the parameter to be accessed		
Format	Unsigned8	0x02 0x03 0x04 0x05 0x06 0x07 0x08	Data type Integer8 Data type Integer16 Data type Integer32 Data type Unsigned8 Data type Unsigned16 Data type Unsigned32 Data type FloatingPoint
		Other values:	See PROFIdrive PROFILE v3.1
		0x40	Zero (without values as a positive subresponse of a write request)
		0x41 0x42 0x43 0x44	Byte Word Double word Error
	The format and number specify the adjoining space containing values in the telegram. Data types in conformity with PROFIdrive Profile shall be preferred for write access. Bytes, words, and double words are also possible as a substitute.		

Field	Data type	Values	Comments
Number of values	Unsigned8	0x00 to 0xEA	No. 0 to 234 Limited by DPV1 telegram length
	Specifies the number of subsequent values.		
Error values	Unsigned16	0x0000 to 0x00FF	Meaning of error values --> see table 4-29
	The error values in the event of a negative response. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the telegram.		
Values	Unsigned16	0x0000 to 0x00FF	
	The values of the parameter for read or write access. If the values make up an odd number of bytes, a zero byte is appended. This ensures the integrity of the word structure of the telegram.		

For more information on coding PROFIdrive data types, see PROFIdrive-specific data types (Page 2410).

See also

Programming example (Page 2446)

Reading and writing parameters with Base Mode Parameter Access (Page 2416)

Specifications for PROFIBUS and PROFINET IO

Global and local parameters

PROFIdrive makes a distinction between two parameter ranges:

- Global parameters; these are assigned to the drive unit as a whole. If you address different DOs on a drive unit, a global parameter will always show the same value.
- Local parameters; these parameters are specific to an axis or a DO. Axis-specific and DO-specific parameters can have different values for each axis/DO.

In view of this, there are two different types of access under Base Mode Parameter Access:

- Base Mode Parameter Access - local (BMPL)
- Base Mode Parameter Access - global (BMPG)

Note

The PROFIdrive standard states that "pipelining" of jobs on PROFIdrive drives (PROFINET and PROFIBUS) is not supported. For each drive unit/DO, only one "Write/read data set" per AR (Application Relationship = communication relationship = controller) is possible.

If, however, more than one PROFIdrive drive unit is connected to a controller via PROFIBUS or PROFINET, one job can be processed in parallel for each of these drive units. The maximum number then depends on the controller. The data for SIMOTION is specified in Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION (Page 2434).

Specific properties of acyclic communication with PROFIBUS

For communication via PROFIBUS, data set 47 (0x002F) is used to access parameters in PROFIdrive drives. Data set 47 is the parameter access point (PAP). Only the global (BMPG) access option is available on PROFIBUS devices. The desired DOs are addressed by supplying the DO ID in the parameter request job structure.

- Base Mode Parameter Access – global; the drive unit's parameters (all DOs, global and local parameters) can be addressed via all the drive unit's PAPs. The SINAMICS devices have a PAP on each PROFIBUS module. Strictly speaking, the PAP is a data record in the PROFIBUS module.

Specific properties of acyclic communication with PROFINET IO

When using PROFINET nothing changes in the fundamental processes for parameter access; however, there are two PAPs available per DO / module.

Note

The MAP submodule is the DO proxy and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. MAP is displayed in the GSD file.

In principle, with both BMPL (data set number 0xB02E) and BMPG (data set number 0xB02F) it is possible to access global parameters of a drive via any parameter access point (PAP/MAP) of the drive.

With **BMPL**, when accessing local parameters (DO), addressing automatically takes place at the local parameters of the DO to which the PAP/MAP is assigned. Therefore, there is no need to specify the DO ID in the parameter request structure, or this will be ignored by the drive.

Use case:
Each axis (DO) has its own parameter access point (this is necessarily the case in single-axis drives). The axis is selected using the logical address; the user does not need to worry about managing the DO IDs.
We recommend this type of access for new projects.

With **BMPG**, it is possible to access not only local parameters of a specific DO via the PAP/MAP of the DO, but also local parameters of other DOs. The required DO is addressed via the DO ID in the parameter request structure. This means that in the case of BMPG, a valid DO ID must also be transferred in all cases.

Use case:
If access to the parameters of a multi-axis drive is only to take place via a PAP/MAP (logical address), or if parameters are to be read to DOs that do not have a dedicated MAP/PAP (particularly PROFIBUS devices). In the case of BMPG, the user is responsible for managing the DO IDs for this purpose.
With this type of access, you remain compatible with old or existing user programs which previously ran on PROFIBUS.

Error assessment

Description

Two different types of errors can occur in conjunction with Base Mode Parameter Access services:

- Error in the communication (transfer of data)
  For example, the addressed device may not exist and is not switched on. This type of error is indicated with the return values of the system functions and is defined in the description of the system functions in the SIMOTION reference lists.
- Error during the processing of the jobs themselves
  For example, an attempt is made to write to a read-only parameter.
  Error codes for this second type of error are defined for PROFIdrive-compliant drives in the PROFIdrive standard and listed below.
  The ID 0x81 (hex) or 0x82 (hex) response indicates an error for the parameter access.
  Error codes are returned in the drive unit's response in the P response data block (see table below). The "Format" field in the parameter response can be used to distinguish whether the queried parameter represents an error code or a "true" value. See the Structure after Base Mode Parameter Access - parameter response (Page 2419) table, offset 4, "Format".
  This table also contains the coding for the "Format" field. Code 0x44 (hex) indicates an error code in the "Values" field. Other "Format" values specify the number format (e.g. Bool, Byte, Integer8, etc.) with which the value in the "Values" field was returned.

Note

The error codes up to 0x20 correspond to the PROFIdrive profile. The error codes from 0x65 onwards are manufacturer-specific and may, therefore, vary from drive to drive.

Error codes in Base Mode Parameter Access responses

Error code	Meaning	Comments	Additional info
0x00	Illegal parameter number	Access to a parameter which does not exist	–
0x01	Parameter value cannot be changed.	Modification access to a parameter value which cannot be changed	Subindex
0x02	Lower or upper value limit exceeded	Modification access with value outside value limits	Subindex
0x03	Invalid subindex	Access to a subindex which does not exist	Subindex
0x04	No array	Access with subindex to non-indexed parameter.	–
0x05	Wrong data type	Modification access with a value which does not match the data type of the parameter	–

Error code	Meaning	Comments	Additional info
0x06	Setting not allowed (only reset allowed)	Modification access with a value not equal to 0 in a case where this is not allowed	Subindex
0x07	Description element cannot be changed.	Modification access to a description element which cannot be changed	Subindex
0x09	No description data	Access to a description which does not exist (the parameter value exists)	–
0x0B	No operating priority	Modification access with no operating priority	–
0x0F	No text array exists	Access to a text array which does not exist (the parameter value exists)	–
0x11	Job cannot be executed due to operating mode.	Access is not possible temporarily for unspecified reasons.	–
0x14	Illegal value	Modification access with a value which is within the limits but which is illegal for other permanent reasons (parameter with defined individual values)	Subindex
0x15	Response too long	The length of the present response exceeds the maximum length that can be transferred.	–
0x16	Illegal parameter address	Impermissible or unsupported value for attribute, number of elements, parameter number, subindex, or a combination of these	–
0x17	Illegal format	Write job: illegal or unsupported parameter data format	–
0x18	No. of values inconsistent	Write job: a mismatch exists between the number of values in the parameter data and the number of elements in the parameter address.	–
0x19	Drive object does not exist.	You have attempted to access a drive object that does not exist.	–
0x65	Presently deactivated	You have tried to access a parameter that, although available, is currently inactive (e.g. n control set and access to parameter from V/f control).	–
0x6B	Parameter %s [%s]: no write access with enabled controller	–	–
0x6C	Parameter %s [%s]: unit unknown	–	–
0x6D	Parameter %s [%s]: Write access only in the commissioning state, encoder (p0010 = 4).	–	–
0x6E	Parameter %s [%s]: Write access only in the commissioning state, motor (p0010 = 3)	–	–
0x6F	Parameter %s [%s]: Write access only in the commissioning state, power unit (p0010 = 2)	–	–
0x70	Parameter %s [%s]: Write access only in the quick commissioning mode (p0010 = 1)	–	–
0x71	Parameter %s [%s]: Write access only in the ready state (p0010 = 0)	–	–
0x72	Parameter %s [%s]: Write access only in the commissioning state, parameter reset (p0010 = 30)	–	–

Error code	Meaning	Comments	Additional info
0x73	Parameter %s [%s]: Write access only in the commissioning state, safety (p0010 = 95)	–	–
0x74	Parameter %s [%s]: Write access only in the commissioning state, tech. application/units (p0010 = 5)	–	–
0x75	Parameter %s [%s]: Write access only in the commissioning state (p0010 not equal to 0)	–	–
0x76	Parameter %s [%s]: Write access only in the commissioning state, download (p0010 = 29)	–	–
0x77	Parameter %s [%s] may not be written in download.	–	–
0x78	Parameter %s [%s]: Write access only in the commissioning state, drive configuration (device: p0009 = 3)	–	–
0x79	Parameter %s [%s]: Write access only in the commissioning state, define drive type (device: p0009 = 2)	–	–
0x7A	Parameter %s [%s]: Write access only in the commissioning state, data set basis configuration (device: p0009 = 4)	–	–
0x7B	Parameter %s [%s]: Write access only in the commissioning state, device configuration (device: p0009 = 1)	–	–
0x7C	Parameter %s [%s]: Write access only in the commissioning state, device download (device: p0009 = 29)	–	–
0x7D	Parameter %s [%s]: Write access only in the commissioning state, device parameter reset (device: p0009 = 30)	–	–
0x7E	Parameter %s [%s]: Write access only in the commissioning state, device ready (device: p0009 = 0)	–	–
0x7F	Parameter %s [%s]: Write access only in the commissioning state, device (device: p0009 not equal to 0)	–	–
0x81	Parameter %s [%s] may not be written in download.	–	–
0x82	Transfer of the control authority (master) is inhibited by BI: p0806.	–	–
0x83	Parameter %s [%s]: requested BICO interconnection not possible	BICO output does not supply float values. The BICO input, however, requires a float value.	–
0x84	Parameter %s [%s]: parameter change inhibited (refer to p0300, p0400, p0922)	–	–
0x85	Parameter %s [%s]: access method not defined.	–	–

Error code	Meaning	Comments	Additional info
0xC8	Below the valid values	Modification job for a value that, although within "absolute" limits, is below the currently valid lower limit	–
0xC9	Above the valid values	Modification job for a value that, although within "absolute" limits, is above the currently valid upper limit (e.g. governed by the current inverter rating)	–
0xCC	Write access not permitted	Write access is not permitted because an access key is not available.	–

Additional information for the parameters of a PROFIdrive drive

Description

From a PROFIdrive drive device, not only the values of parameters, but also the descriptions of the parameters, can be read.

The P response/request data block in the "Attribute" field is used to express a preference when sending the "parameter request":

Attribute = 0x10 (hex)	Value
Attribute = 0x20 (hex)	"Parameter Description" parameter description
Attribute 0 0x30 (hex)	Parameter Text

If, rather than the value of a parameter, its "Parameter Description" is requested, the "Value" field in the "Parameter Response" contains the description (data type, possibly the number of indexes of the parameter, ...).

Note

Normally, parameter descriptions are read-only.

System commands in SIMOTION

_writeRecord/_readRecord SIMOTION system commands

Description

A "write data record" can be performed in SIMOTION using the _writeRecord() system command. A "read data record" can be performed in SIMOTION using the _readRecord() system command. This makes it also possible to read, write or fetch the description of parameters in a PROFIdrive drive.

The description of the system functions, their input parameters and return values can be found in the SIMOTION system documentation:

- C2xx reference list
- D4xx/D4xx-2 reference list
- P3xx reference list

The _write/_readRecord system commands can be used universally, not just for PROFIdrive drives, but, for example, also for intelligent sensors on the PROFIBUS or other peripheral modules that support the so-called DP V1 services for PROFIBUS.

Note

For SIMATIC, the corresponding system functions are

SFB52 WR_REC Write data record

SFB53 RD_REC Read data record

The following is required to be able to use the SIMOTION system commands _write/_readRecord:

- PROFIBUS: Access to the logical address of a telegram module or diagnostics address of the device (module 0)
- PROFINET: Data set 0xB02E for BMPG is accessed via the diagnostic address of the MAP submodule of the DOs . The MAP submodule is the DO proxy and hosts the alarm channel along with one or more PAPs (Parameter Access Point). The PAP is a data record in the MAP submodule. In the GSD file, the MAP is stated with submodule ID 0xFFFF and the submodule name "MAP."

Furthermore, the DO ID is only relevant for data set 47 (0x002F) and Global Access (PROFINET 0xB02F). Use data set 47 for compatibility with PROFIBUS user programs.

The diagnostics address of the corresponding PAP is relevant for Local Access (PROFINET IO 0xB02E), the DO ID is not analyzed. Use this type of access for new projects.

As a result, for example in connection with PROFIdrive units, the telegram start address of the PROFIdrive telegram exchanged cyclically with the device is required. Use of the diagnostic address of the MAP submodule is recommended. In SINAMICS drives, the log addresses of the telegram modules can also be used. In this way, an F-controller can also access PROFIsafe submodules.

If a drive has several axes (with a shared PROFIBUS interface connection) on a drive device, to differentiate the axes in the same device, the "Axis-No." or "DO-ID" in data set 47 is also required. SIMODRIVE 611universal and SINAMICS S120 are examples for such multi-axis drives. To determine the "DO-ID" for SINAMICS S120, refer to the **Acyclical Communication** section in the SINAMICS S120 Commissioning Manual.

"Axis-No." or "DO-ID" = 0 can be used to access the so-called "global parameters". Examples of such "global parameters" are:

- P0918: PROFIBUS address
- P0964: Device identification (manufacturer, version, number of axes, etc.)
- P0965: Profile number (the implemented PROFIdrive version)

- P0978: List of the DO Ids (the set "Axis-No." or "DO-ID")
- P61000: Name of the station (PROFINET devices only)

_writeDrive.../_readDrive... SIMOTION system commands

Description

Whereas the _readRecord and _writeRecord system functions can be used universally for all devices on PROFIBUS that support the so-called "read/write data record" DP V1 services, the following commands are specially tailored to PROFIdrive drives using the PROFIdrive profile:

- _read/writeDriveParameter (reads/writes a, possibly indexed, drive parameter)
- _read/writeDriveMultiParameter (reads/writes several, possibly indexed, drive parameters for a drive or drive object)
- _readDriveFaults (reads the current fault buffer entry of a drive or drive object)
- _readDriveParameterDescription (reads the descriptive data of a parameter from the drive or drive object)
- _readDriveParameterDescription (reads the descriptive data of several parameters from the drive or drive object)

The commands create internally the data set 47 required for the individual functions in accordance with PROFIdrive profile using the parameters transferred by the user when the system functions are called, and independently handle the communication to the PROFIdrive drive using "read/write data record".

The commands are described in the SIMOTION system documentation, refer to the reference lists for the associated platform.

See also

Scope for the rules (Page 2437)

Comparison of the system commands

Description

The following table shows the most important differences between the two groups of system commands:

Command group	Advantage	Disadvantages
_readRecord _writeRecord	• Generally usable, not just for DP V1 services for drives • Assumes only the knowledge of some I/O address *on the drive device* and the "DO-ID" or "Axis-No" on the drive device	• The user must create the data record • The user must program two calls for parameter accesses in a PROFIdrive drive • Users may need to perform the required data conversions themselves • "DO-ID" or "Axis-No" must be known
_readDrive... _writeDrive...	• Tailored for the typical communication with PROFIdrive drives • The user does not need to know the structure of data set 47 • Reduced programming effort for the user for communication to drives	• Assumes the presence or knowledge of an I/O address *of the associated drive object* • An I/O address for a drive object exists only for cyclical communication (with PROFIBUS) to the drive object, possibly, for example, not for TB30 and TMxx I/O expansion modules used exclusively in the drive • The user must make any required data conversions

Properties of the system commands

The use of the drive-specific _write/_readDrive... system commands on the one hand makes it easier for you than using general _write/_readRecord commands, since you do not need to know the structure of data set 47 and do not need to program the successive _writeRecord and _readRecord calls in sequencers. Because the general usability of these system functions means the structure of the transferred data records is not known to the system, you may need to perform the required conversion into the representation in accordance with the PROFIdrive profile for sending and receiving yourself, see Program example (Page 2446).

Up to SIMOTION V4.1, the use of the _write/_readDrive... commands is restricted to those cases for which there is cyclic data traffic to the associated drive object, because this is required as an input parameter. From this version up, it is possible to transfer the DO ID or axis no. via the **doId** parameter of the system functions. This means that it is possible to communicate with every DO, even if acyclic data traffic is involved.

In contrast, _write/_readRecord can also be used to access drive objects even when no cyclical data traffic exists (or when the I/O address is not known in the application). This succeeds with _write/_readRecord because the explicit knowledge of the "DO-ID" or "Axis-No." and the

knowledge of some I/O address on the device suffices to construct the data set 47. This can be advantageous, for example, when individual drive objects are used only drive-internal (namely, without cyclical telegram traffic for control) or they are not generally known for "generic programming".

Deleting _readDrive and _writeDrive jobs

Description

You can use the following functions to cancel or delete incorrect read or write jobs, which, for example, were called with the _readDriveParameter:

- _abortReadWriteRecordJobs, for the _readRecord or _writeRecord functions
- _abortAllReadWriteDriveParameterJobs, for the following functions:
  - _readDrive(Multi)ParameterDescription
  - _readDrive(Multi)Parameter
  - _writeDrive(Multi)Parameter
  - _readDriveFaults

You can call the functions without needing to know or read the CommandID.

Rules for using _readRecord and _writeRecord

Rule 1 - the job has its own job reference

Each job has its own job reference

This is required so that different jobs can be assigned. The job reference can be reused when the assignment is clear because of some other characteristic, such as the chronological sequence.

Rule 2 - system functions for asynchronous programming

Description

R2: For asynchronous programming, you must repeatedly call the system function with the same IDs until the function is terminated ("longrunner"). The correct use of the system functions _writeRecord and _readRecord based on communication with SINAMICS S120 is shown in the figure **Correct processing with the _readRecord and _writeRecord system functions**.

The communication for reading and writing parameters for the SINAMICS S120 is always performed using data set 47, whose structure is described in the documentation for the SINAMICS S120, refer to the Acyclical Communication section in the SINAMICS S120 Commissioning Manual.

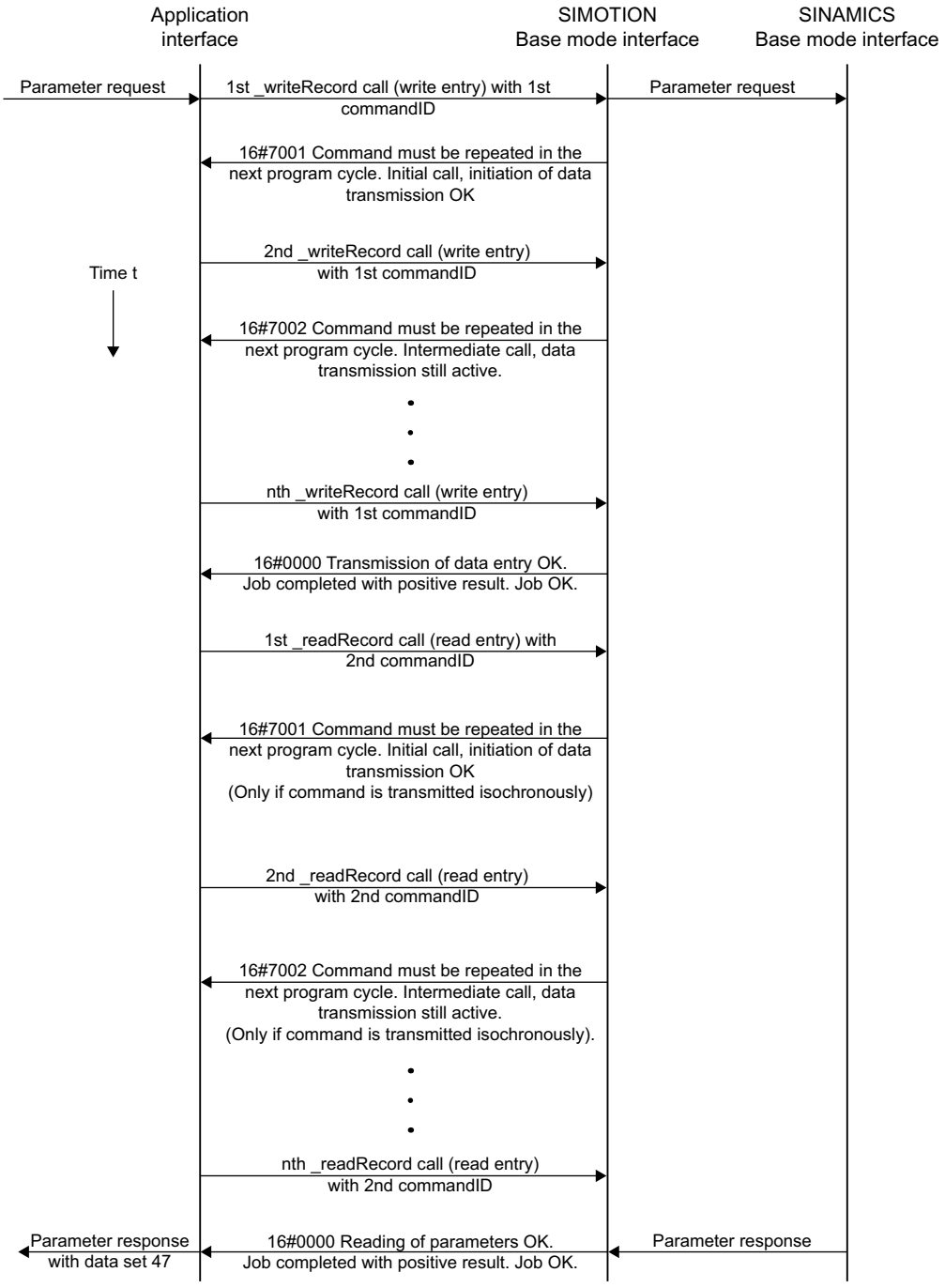


Figure 4-491 Correct processing with the _readRecord and _writeRecord system functions

Rule 3 - read/write data record per PROFIdrive drive unit

Only one read/write data record per PROFIdrive drive device concurrently

The PROFIdrive profile specifies that PROFIdrive drives do not perform any pipelining and consequently only one job will be processed at any one time. Consequently, this is also described for SINAMICS S120 in the Commissioning Manual.

Note

It does not matter which system functions are used for the transmission in the controller. A PROFIdrive drive can process only one job at any one time.

Note

It is certainly possible for other devices on the PROFIBUS that they support several "read/write data record" in parallel.

Note

Because the _write/_readRecord system functions can be used universally, *no* interlock is performed on the controller side to limit only one "read/write data record" per PROFIdrive drive to be initiated at any one time.

Consequence for the application on the controller:

An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive device, also refer to section Interlocking of several calls (Page 2438).

Rule 4 - the last call wins for SIMOTION

In case of doubt, the last call "wins" for SIMOTION

If Rule 3 "Only one read/write data record per PROFIdrive drive device concurrently" is violated by a second _writeRecord command being issued to the same drive in the meantime, the response of the first job can then no longer be read. The attempt to read the drive response to the first job can no longer be processed by the drive and will be acknowledged with an error and terminated. The chronological sequence is shown in the figure **The second _writeRecord call wins in case of doubt**.

To differentiate between the jobs at the controller, a separate commandID was used for each of the calls of the _writeRecord and _readRecord system functions.

To also differentiate between the jobs at the drive, unique job references for the first and second job were assigned in data set 47.

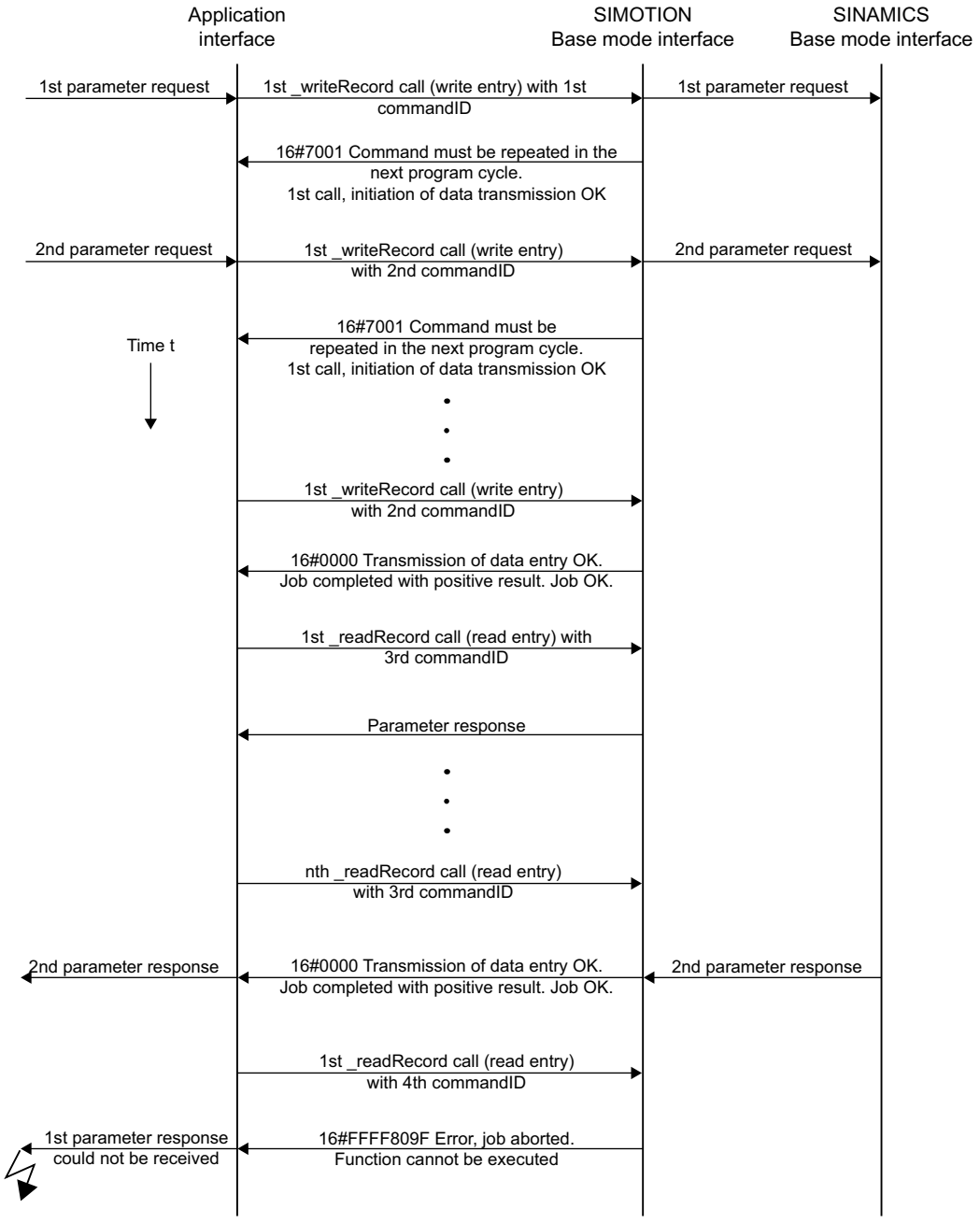


Figure 4-492 The second _writeRecord call wins in case of doubt

Rule 5 - a maximum of eight concurrent calls is possible in SIMOTION

SIMOTION can manage a maximum of eight _write/_readRecord calls concurrently

Although according to rule 3 (see Rule 3 (Page 2433)) only a single job can be processed at any given time for a *single* PROFIdrive drive device, it is still possible for the control program to issue several jobs in parallel.

Although this does not make any sense for a *single* PROFIdrive drive, it can be sensible for communication to *several* drives in parallel (or possibly for other devices that support this).

For SIMOTION, resources are reserved to permit a maximum of eight _write/_readRecord calls to be managed. The _write/_readRecord commandID is used to differentiate between the calls. If an attempt is made to issue a ninth concurrent call, this will be acknowledged by the controller with an error and suppressed.

The chronological sequence is shown in figure **Managing 8 jobs simultaneously**.

Initially seven _writeRecord jobs are initiated but not completed (no further _writeRecord calls to complete the jobs). The eighth _writeRecord job will be initiated and further processed until completion. It is then possible to issue a ninth call (which, however, is not further processed by the user program). The SIMOTION _writeRecord system function then acknowledges the attempt to issue the tenth job with error 16#80C3, because this would have been the ninth "open" job.

Note

The upper limit applies to each SIMOTION controller, not to each bus segment on the controller. This means it does not matter whether the addressed target devices operate on a single PROFIBUS segment or are assigned to several PROFIBUS segments.

Note

Because the _write/_readRecord system functions can be used universally, *no* interlock is performed on the controller side to limit only one "read/write data record" per PROFIdrive drive to be initiated at any one time.

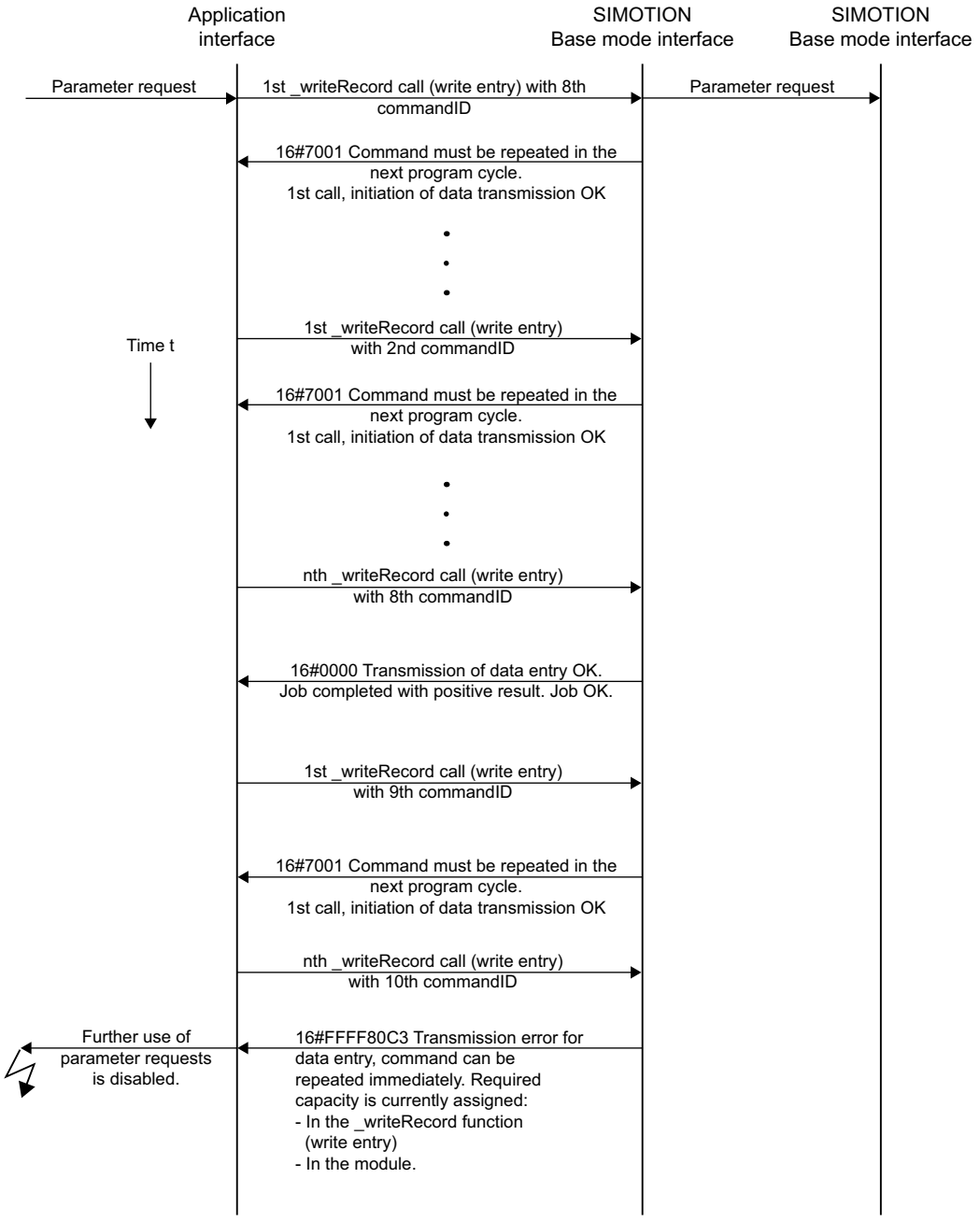


Figure 4-493 Managing 8 jobs simultaneously

Note

If the error 16#80C3 occurs, you must set the CPU to STOP and then back to RUN. This deletes the job buffer. In order to prevent the error, you should end the job with an abort command, if you are unable to end the job.

Rules for SIMOTION _writeDrive.../_readDrive... commands

Scope for the rules

Description

The following examples are shown using the _readDriveParameter system function . The descriptions also apply similarly for the previously mentioned _writeDrive.../_readDrive... system functions.

Rule 6 - repeated call of system function for asynchronous programming

Description

For asynchronous programming, the user must call repeatedly the system function with the same IDs until the function is terminated ("longrunner").

The following figure shows the correct use of the _readDriveParameter system function.

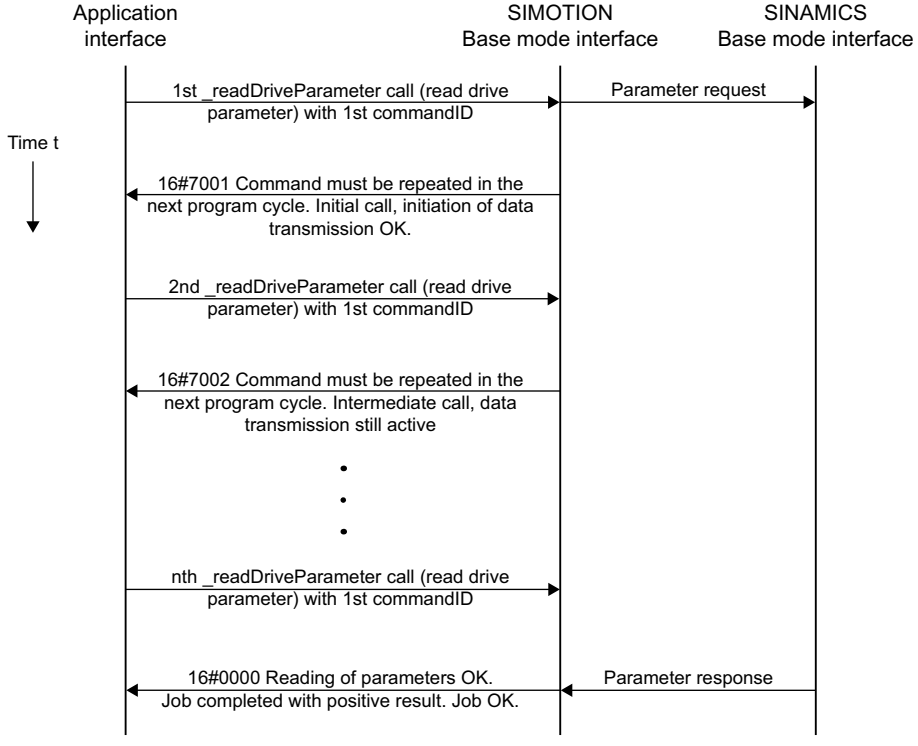


Figure 4-494 Correct processing with the _writeDriveParameter and _readDriveParameter system functions

Rule 7 - multiple concurrent calls per target device

Description

The PROFIdrive standard specifies that PROFIdrive units do not perform any pipelining and consequently only one job will be processed at any one time. Consequently, this is also documented for SINAMICS S120 in the SINAMICS S120 Commissioning Manual.

Because the SIMOTION _write/_readDrive... system commands have been created for the frequent use with PROFIdrive units, this is already handled by the controller.

Note

It does not matter which system functions are used for the transmission in the controller. A PROFIdrive drive can process only one job at any one time.

Consequence for the application on the controller:

An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive device.

The figure below shows the behavior when this is not handled. The attempt to issue a second job (with unique commandID) to the same target device will be acknowledged with an error. A further job to the same target device can then be issued only when the first job has completed or has been canceled, see Section Releasing the Interlocking (Page 2438).

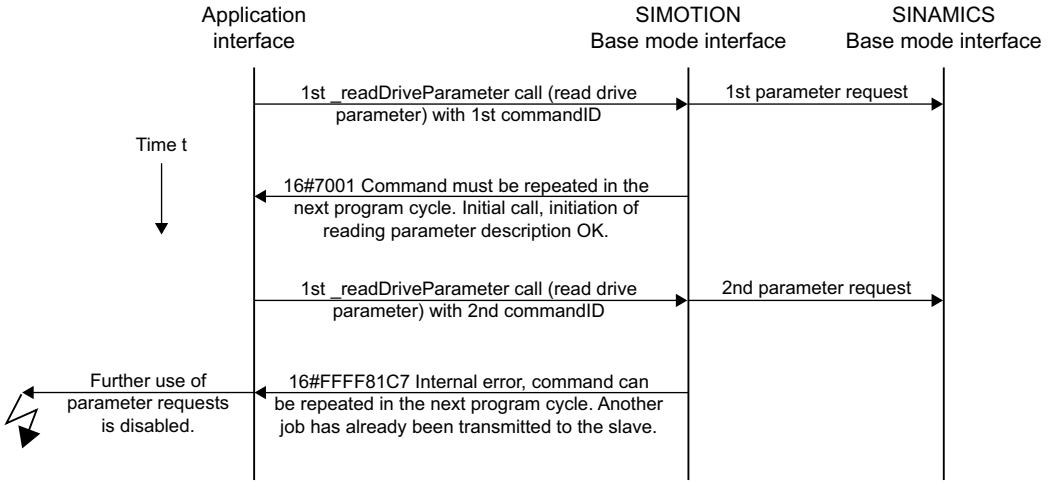


Figure 4-495 Interlocking of several _readDriveParameter jobs on a target device

Rule 8 - release the interlocking after the complete processing of a job

Enable the interlock only after the processing of a job has been completed

The following figure shows that it does not suffice to wait for "something", but rather the _read/_writeDrive... system functions must be called repeatedly until the job has been processed completely. The interlock will not be freed and the internal management resources released beforehand.

The number of calls has been selected so that the SIMOTION DP V1 interface answers each subsequent call for the first job with 16#7002 and thus is not processed completely. Depending on the loading of the bus and the drive, this can also be necessary very frequently (>25 times). This means an estimate cannot be given.

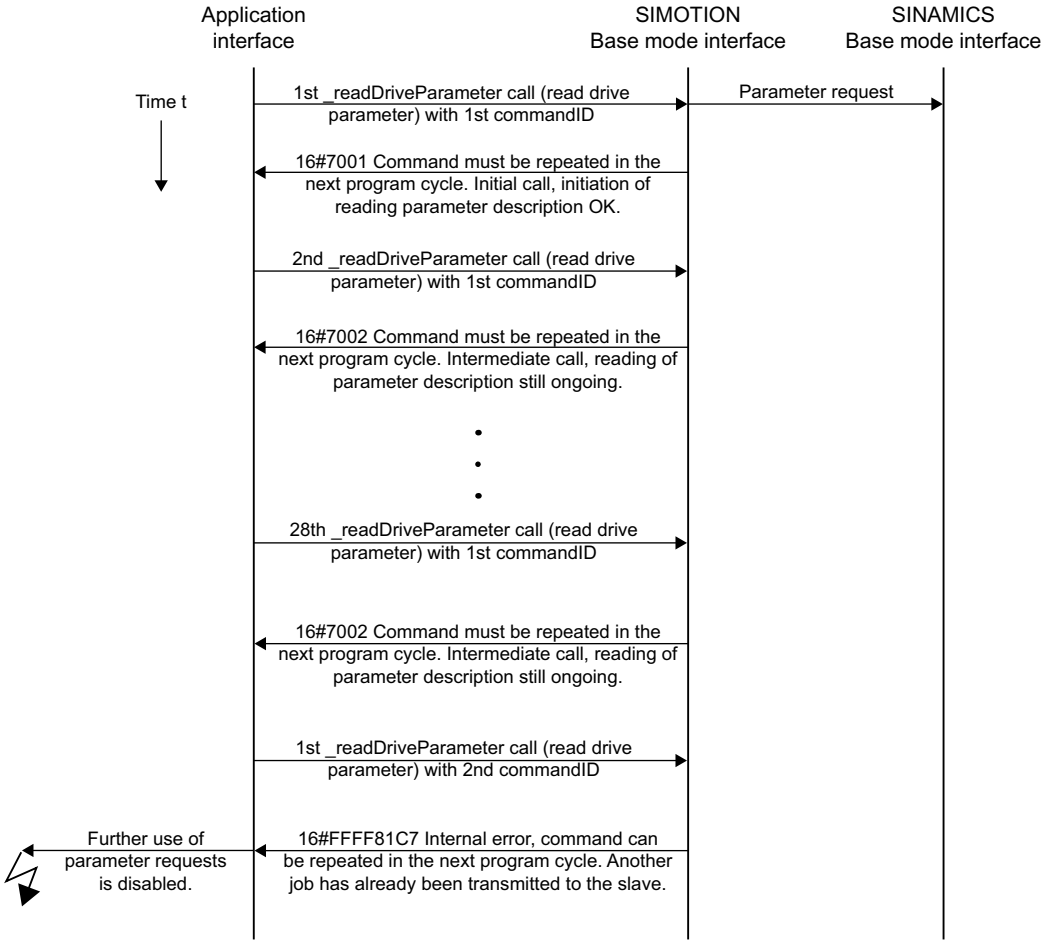


Figure 4-496 Complete processing of a _readDriveParameter required to release the interlock

Rule 9 - canceling jobs for an asynchronous call

CommandID is needed to cancel jobs for an asynchronous call

To re-enable the DP V1 service for the target device,

- either the first job must have completed (repeated calls with the commandID of the first job)
- or cancelled (again a call of the _readDriveParameter function with the same commandID as for the first initiation of the job. In addition, the nextCommand input parameter must have the ABORT_CURRENT_COMMAND value).

Note

From V4.1 and up, it is possible to cancel without knowing the commandID, see Deleting _readDrive and _writeDrive jobs (Page 2431) .

A sample call of the _readDriveParameter function with the first commandID (id1) and ABORTED_CURRENT_COMMAND has the following form:

```
Return_Par_read_delete :=
    readDriveParameter(
    ioId:=INPUT,
    logAddress := 256,
    parameterNumber := number,
    numberOfElements := 0,
    subIndex:= 0,
    nextCommand :=
    ABORT_CURRENT_COMMAND,
    commandId := id1);
```

The figure below shows the chronological sequence.

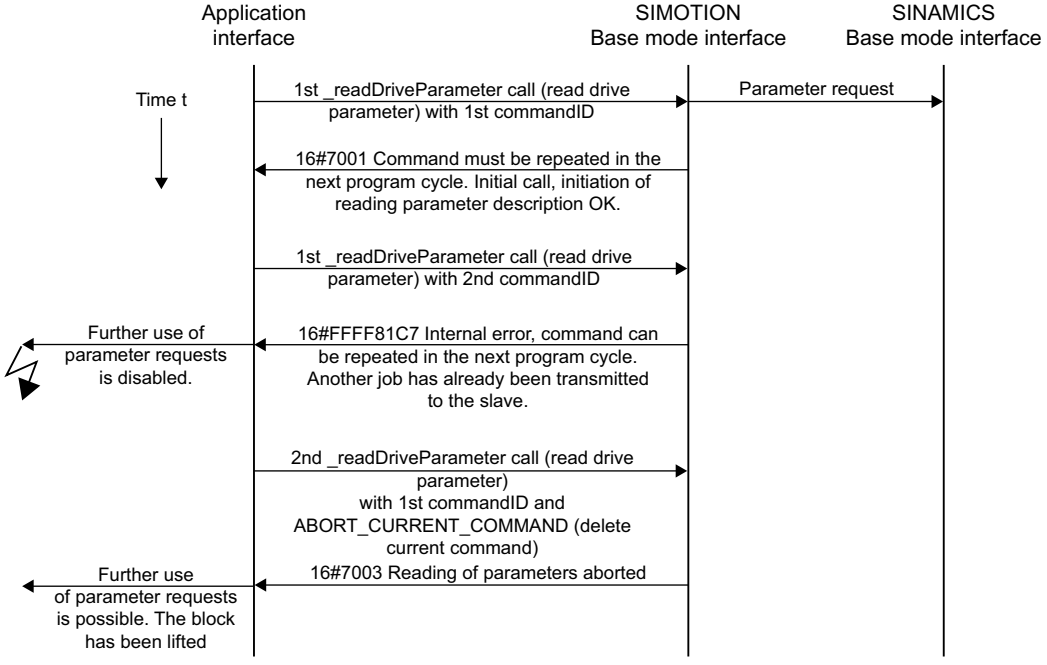


Figure 4-497 Canceling a _readDriveParameter job with known commandID

The process in the following figure shows that it is not possible to cancel a job without knowledge of the original commandID. Not the first job, but rather the cancel attempt will be canceled. The reason is that the commandID is used for managing the various jobs in the system.

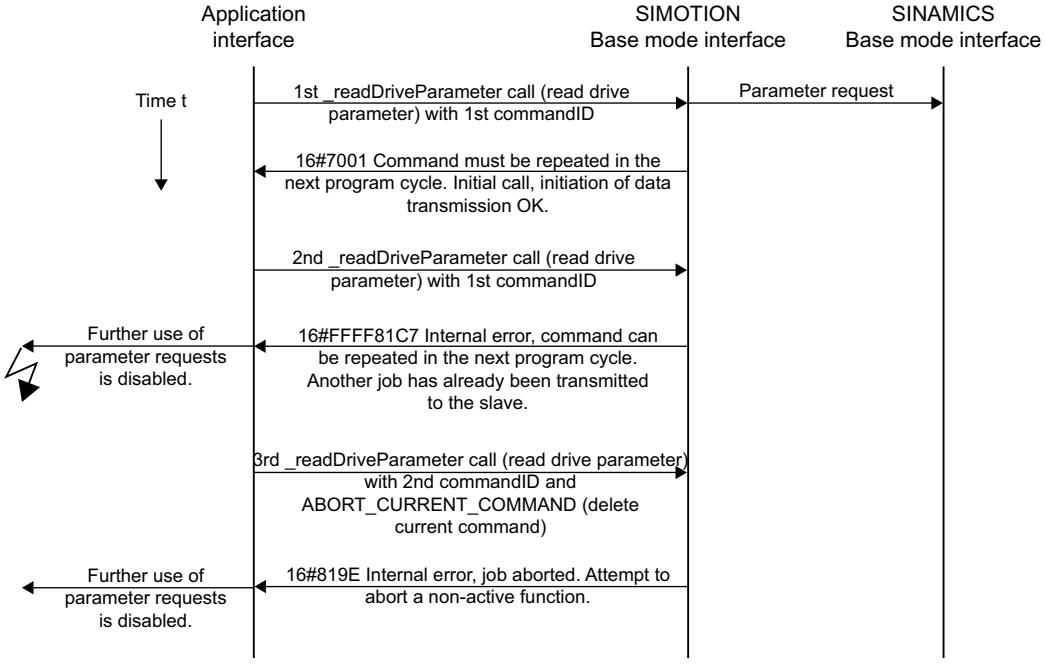


Figure 4-498 No cancelation of a _readDriveParameter job with new CommandID

Note

It is therefore important that the user program retains the commandID of the jobs until the job has completed or has been canceled.

Note

Take particular care, for example, through control by other conditions, that in the user program the processing of the _write/_readDrive... functions is not bypassed before they have completed.

Rule 10 - management of sixteen jobs

SIMOTION manages a maximum of 16 calls in parallel for different devices

The controller has limited resources (memory space) available for storing the management data for _write/_readDrive... system function calls. If too many calls are issued in parallel, an error message will be issued, similar to the limit for _read/_writeRecord in Section Maximum Number of Calls (Page 2434).

For SIMOTION, resources are reserved to permit a maximum of sixteen calls of _writeDrive.../_readDrive... system functions to be managed. The commandID is used to differentiate between

the calls. If an attempt is made to issue a seventeenth concurrent call, this will be acknowledged by the controller with an error and suppressed.

Rule 11 - parallel jobs for different drive devices

Parallel jobs to different drive units are possible

The figure **Parallel processing of _readDriveParameter jobs to different drive devices of a controller** shows that parallel jobs can be processed with different drive devices.. The SIMOTION D445-2 controller uses the SINAMICS Integrated of a D445-2 (for example) as first PROFIdrive drive unit and the CX32-2 expansion module as second PROFIdrive drive unit.

In the example, a total of three read jobs (two jobs to the first drive device (SINAMICS Integrated) and one job to the second drive device (CX32-2)) are issued with the _readDriveParameter system function.

- The first read job for the SINAMICS Integrated is intentionally called just once so that the interlock acts.
- The second read job is then issued to the second PROFIdrive drive device (CX32-2). This job is processed successfully.
- The third read job is addressed again to the first drive device (SINAMICS Integrated) and can no longer be executed successfully because the first job is still running.

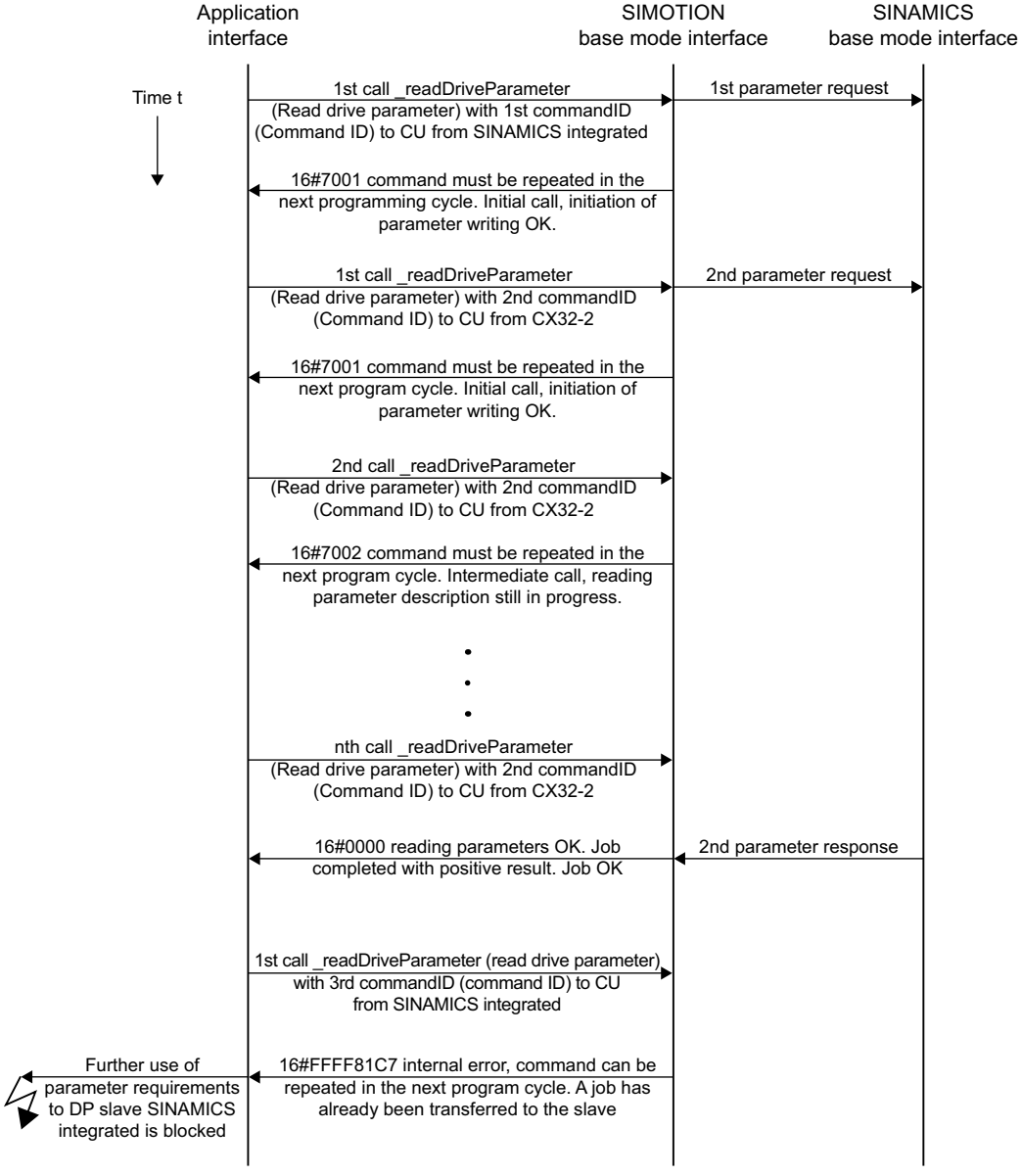


Figure 4-499 Parallel processing of _readDriveParameter jobs to different drive devices of a controller

Special features

Rule 12 - data buffering of up to 64 drive objects

SIMOTION buffers the data of up to 64 drive objects

The first call to the functions _write/_readDrive... after system power-up runs considerably longer than subsequent calls to the same drive object.

- The system must first set up internal management information that can be accessed faster in subsequent calls to the same drive object.

In SIMOTION, the data of up to 64 drive objects can be stored for use with _write/_readDrive... The distinction is made using the I/O address.

Rule 13 - a mix of system functions can be used

A mix of the _writeRecord/_readRecord and _writeDrive.../_readDrive... system functions can be used

A mixed use of the following system commands is generally possible:

- _writeRecord/_readRecord SIMOTION system commands
- _writeDrive.../_readDrive... SIMOTION system commands

Note

However, it is important to appreciate that a missing interlock of the system commands from the two command groups means several jobs could be issued to a PROFIdrive drive (see following section), which a PROFIdrive drive cannot process. Handling the system-internal interlocking for _write/_readDrive....

The figure **Mixed use of _readDrive... and _read/_writeRecord** shows that the _write/_readRecord functions, in particular, can be used for the same target device when, because of a running _readDriveParameter job, further jobs with the same command are suppressed by the system – this situation must be blocked by the user because it cannot be processed by a PROFIdrive.

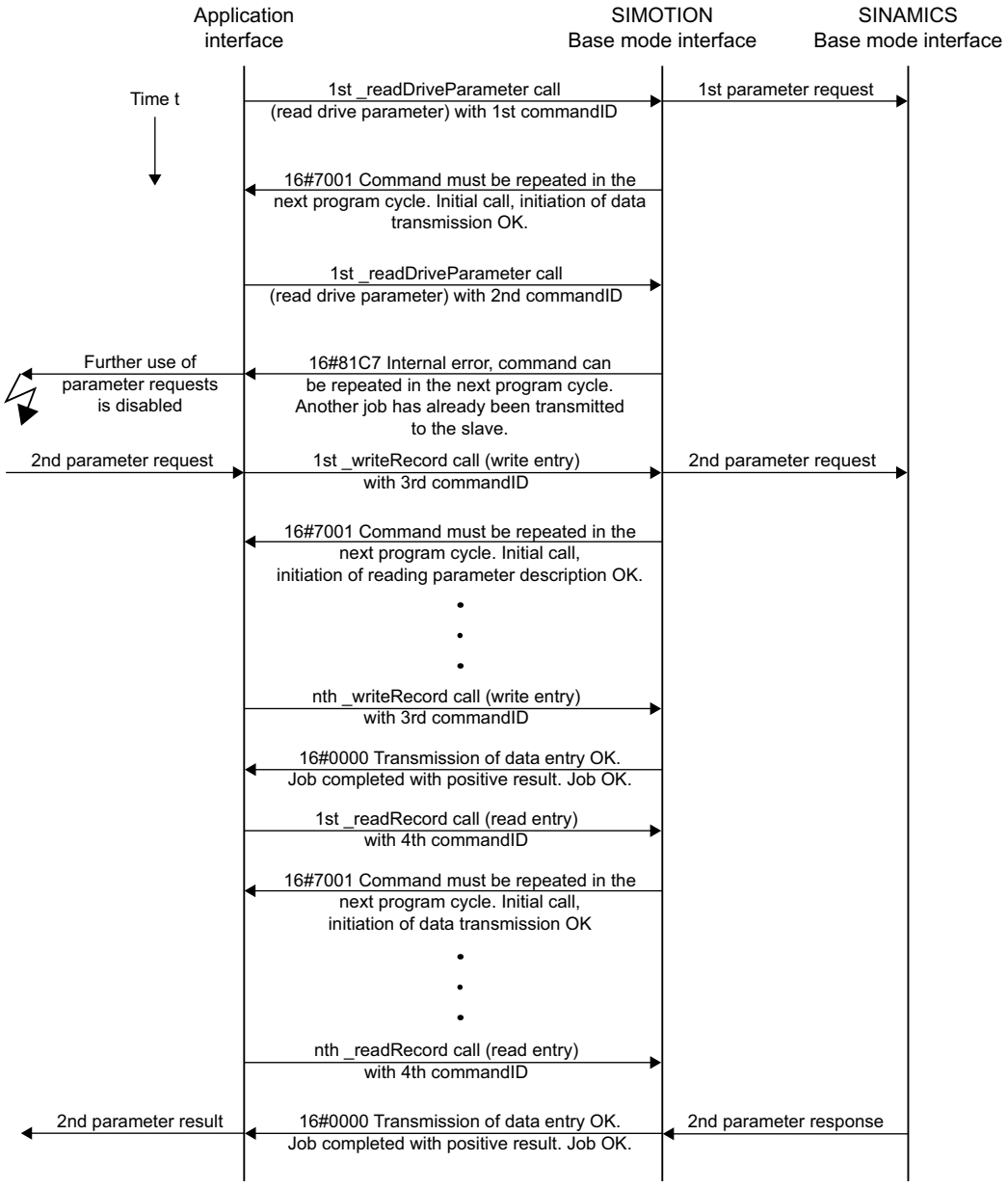


Figure 4-500 Mixed use of _readDrive... and _read/_writeRecord

Rule 14 - interlocking for the mixed use of commands

The user must interlock for the mixed use of the commands from the two command groups

When the following system commands are used together, it is possible that more than one "read/write data record" is issued concurrently to a single device because for SIMOTION interlocking and buffering is performed only within the command groups but not between command groups.

- _writeRecord/_readRecord SIMOTION system commands

and

- _writeDrive.../_readDrive... SIMOTION system commands

If necessary, this must be interlocked by the user to prevent data loss/overlapping because the PROFIdrive profile specifies that a PROFIdrive drive does not perform any pipelining and consequently can process only one job at any given time.

Program examples

Programming example

Description

The following example shows how the _writeRecord and _readRecord system commands can be used to fetch the error code from parameter p0945 of a SINAMICS drive (drive object DO3, I/O address 256).

Example

The sample program can be called, for example, in the BackgroundTask, because so-called "asynchronous programming" is used.

```
//=========================================================================
// demonstrate reading parameter 945 (fault code) via data set 47
// using SIMOTION system functions _write/_readRecord (asynchronous call)
// INPUT address 256 is assumed to address the SINAMICS
// drive is DO3 in SINAMICS S120
//=========================================================================
INTERFACE
 PROGRAM record;
 // declare request type
 TYPE
 // declare struct of header request
 Header_Type_Request : STRUCT
    Request_Reference : USINT;
    Request_Id : USINT;
    Axis : USINT;
    Number_Of_Parameter : USINT;
 END_STRUCT;

 // declare struct of parameter address request
 Parameter_Address_Request : STRUCT
    Attribute : USINT;
    Number_Of_Elements : USINT;
    Parameter_Number : UINT;
    SubIndex : UINT;
 END_STRUCT;

 // declare struct of request
 Request : STRUCT
    Header : Header_Type_Request;
    ParameterAddress : Parameter_Address_Request;
 END_STRUCT;
```

```
// declare struct of header response
 Header_Type_Response : STRUCT
 Response_Reference : USINT;
 Response_Id : USINT;
 Axis : USINT;
 Number_Of_Parameter : USINT;
 END_STRUCT;

 // declare struct of parameter address response
 Parameter_Address_Response : STRUCT
 Format : USINT;
 Number_Of_Elements : USINT;
 Value_Or_Error_Value : DWORD; // dependent on format
 END_STRUCT

 // declare struct of response
 Response : STRUCT
 Header : Header_Type_Response;
 ParameterAdress : Parameter_Address_Response;
 END_STRUCT;
 END_TYPE
 // declare global variables
 VAR_GLOBAL
 // declare variable, that represents the dataset 47 request
 myRequest : Request;
 // declare variable, that represents the dataset 47 response
 myResponse : Response;
 // declare variable, that returns a value after calling _writeRecord
 myRetDINT : DINT;
 // declare variable, that returns a struct after calling _readRecord
 myRetstructretreadrecord : StructRetReadRecord;
 // declare array of byte,
 // which helps to create the request/response
 // with marshalling function
 bytearray : ARRAY[0..239] OF BYTE;
 // declare array of USINT,
 // because the systemfunctions _writeRecord and _readRecord
 // use this array
 usintarray : ARRAY[0..239] OF USINT;
 // declare command ids
 id_write, id_read : commandidtype;
 // declare the variable, to control step by step execution
 // start cycle with setting to 0 by user
 program_step : USINT := 3; // initially idle;
 END_VAR
END_INTERFACE
```

Implementation

```
// ===============================================================
IMPLEMENTATION
PROGRAM record
CASE program_step OF
// initialize ------------------------------------------------------
0:
 // get command ids for calling system functions
 id_write := _getcommandid();
 id_read := _getcommandid();
 // header from the request
 // here: Axis-No / DO-ID is 3
 // read Parameter 945 (drive fault code)
 myRequest.Header.Request_Reference := 16#10; // arbitrary no.
 myRequest.Header.Request_Id := 16#1; // read request
 myRequest.Header.Axis := 16#3; // axis no 3
 myRequest.Header.Number_Of_Parameter := 16#1; // one parameter

 // parameter address from the request
 myRequest.ParameterAddress.Attribute := 16#10; // read value
 myRequest.ParameterAddress.Number_Of_Elements := 16#1; // one index
 myRequest.ParameterAddress.Parameter_Number := 945; // parameter no.
 myRequest.ParameterAddress.SubIndex := 0;

 // convert myRequest to a BIBBYTEARRAY to use the marshalling functions
 // two step conversion from user defined data type
 // to usintarray type required by system functions
 bytearray := ANYTYPE_TO_BIGBYTEARRAY(myRequest,0);
 usintarray := BIGBYTEARRAY_TO_ANYTYPE(bytearray,0);

 // next step
 program_step := 1;

// execute _writeRecord -------------------------------------------
```

```
1:
 // the systemfunctions _writeRecord and _readRecord
 // have to be called in sequence.
 // the functions occur always as pair.
 // call systemfunction _writeRecord to send the request
 myRetDINT := _writerecord(
    ioid := INPUT,
    logaddress := 256, // io address
    recordnumber := 47, // data set 47 for DPV1
    offset := 0,
    datalength := 240,
    data := usintarray, //
    nextcommand := IMMEDIATELY, // use asynchronous
    commandid := id_write // use known commandID
 );
 // check the return value
 // keep calling until _writeRecord ready
 IF(myRetDINT = 0)THEN
    // next step
    program_step := 2;
 END_IF;
// wait for requested data ------------------------------------------------
// execute _readRecord
2:
 // call systemfunction _readRecord to receive the data
 myRetstructretreadrecord := _readrecord(
    ioid := INPUT,
    logaddress := 256, // io address
    recordnumber := 47, // data set 47 for DPV1
    offset := 0,
    datalength := 240,
    nextcommand := IMMEDIATELY, // use asynchronous
    commandid := id_read // use known commandID
 );
 // check the return value
 // keep calling until _readRecord ready
    IF(myRetstructretreadrecord.functionresult = 0)THEN
        // next step
        program_step := 3; // --> done
        // get data
        // two step conversion into user defined data type
        // from usintarray type given by system functions
        bytearray :=ANYTYPE_TO_BIGBYTEARRAY(
        myRetstructretreadrecord.data,0);
        myResponse := BIGBYTEARRAY_TO_ANYTYPE(bytearray,0);
        // received data can now be read from myResponse...
    END_IF;
END_CASE;
END_PROGRAM
END_IMPLEMENTATION
```

4.5.11 Appendix

4.5.11.1 Standard PROFIBUS/PROFINET data types (only available in English)

Subset of IEC 61158-5 standard data types for use in PROFIBUS/PROFINET profiles

Coding of most of the data types in these profile guidelines is defined in IEC 61158-6:2003, clause 6.2. However, the usage in practice is different in some cases. Thus it is highly recommended to follow the definitions and hints within this annex. The next edition of IEC 61158 is supposed to comply with the content of this annex.

Boolean

A Boolean is representing a data type that only can have two different values i.e. TRUE and FALSE. Hint: for efficiency reasons this data type is not used in application profiles.

Numeric Identifier	Data type name	Value range	Resolution	Length
1	Boolean	True/false	-	1 Octet

Bits	7	6	5	4	3	2	1	0	
True	x	x	x	x	x	x	x	x	0x01 to 0xFF
False	0	0	0	0	0	0	0	0	0x00

Integer16

An Integer16 is representing a signed number depicted by 16 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
3	Integer16	$-32768 \leq i \leq 32767$	1	2 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Integer32

An Integer32 is representing a signed number depicted by 32 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
4	Integer32	$-2^{31} \leq i \leq 2^{31}-1$	1	4 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Integer64

An Integer64 is representing a signed number depicted by 64 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
55	Integer64	$-2^{62} \le i \le 2^{62}-1$	1	8 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octets 1	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octets 1	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{52}	2^{41}	2^{40}
Octets 1	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Unsigned8

An Unsigned8 is representing an unsigned number depicted by 8 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
5	Unsigned8	$0 \le i \le 255$	1	1 Octet

Table 4-247 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Table 4-248 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	7	6	5	4	3	2	1	0

Unsigned16

An Unsigned16 is representing an unsigned number depicted by 16 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
6	Unsigned16	0 ≤ i ≤ 65535	1	2 Octets

Table 4-249 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Table 4-250 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	15	14	13	12	11	10	9	8
Octets2	7	6	5	4	3	2	1	0

Unsigned32

An Unsigned32 is representing an unsigned number depicted by 32 bits ("enumerated"). Some application profiles (e.g. PROFIsafe) are using this data type for the coding of individual single bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
7	Unsigned32	0 ≤ i ≤ 4294967295	1	4 Octets

Table 4-251 Enumerated:

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Table 4-252 Single bits:

Bits	7	6	5	4	3	2	1	0
Octets 1	31	30	29	28	27	26	25	24
Octets 2	23	22	21	20	19	18	17	16
Octets 3	15	14	13	12	11	10	9	8
Octets 4	7	6	5	4	3	2	1	0

Unsigned64

An Unsigned64 is representing a signed number depicted by 64 bits.

Numeric Identifier	Data type name	Value range	Resolution	Length
56	Unsigned64	$0 \le i \le 2^{64}-1$	1	8 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

SN = 0: positive numbers and zero

SN = 1: negative numbers

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
Octets 2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
Octets 3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{52}	2^{41}	2^{40}
Octets 4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
Octets 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Octets 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Octets 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Octets 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Float32

A Float32 is representing a number defined by ANSI/IEEE 754 as single precision.

Numeric Identifier	Data type name	Value range	Resolution	Length
8	Float32	refer to ANSI/IEEE 754	refer to ANSI/IEEE 754	4 Octets

SN: sign 0 = positive, 1 = negative.

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
Octets 2	(E)	Fraction (F)						
	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Octets 3	Fraction (F)							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octets 4	Fraction (F)							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Float64

A Float64 is representing a number defined by ANSI/IEEE 754 as single precision.

Numeric Identifier	Data type name	Value range	Resolution	Length
15	Float64	refer to ANSI/IEEE 754	refer to ANSI/IEEE 754	8 Octets

SN: sign 0 = positive, 1 = negative.

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}
Octets 2	Exponent (E)				Fraction (F)			
	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Octets 3	Fraction (F)							
	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
Octets 4	Fraction (F)							
	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
Octets 5	Fraction (F)							
	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
Octets 6	Fraction (F)							
	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
Octets 7	Fraction (F)							
	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
Octets 8	Fraction (F)							
	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

Visible String

This data type is defined as the ISO 646 string type. Characters are based on 8 Bit ASCII.

Numeric Identifier	Data type name	Value range	Resolution	Length
9	VisibleString	refer to ISO 646	-	variable

Bits	7	6	5	4	3	2	1	0
Octets 1	1. Character							
Octets 2	2. Character							
...	...							
Octets n	n. Character							

OctetString

An OctetString is an ordered sequence of Bytes, numbered from 1 to n.

Numeric Identifier	Data type name	Value range	Resolution	Length
10	OctetString	-	-	variable

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Character							
Octet 2	2. Character							
...	...							
Octet n	n. Character							

BYTE

A Byte is 1 octet.

Numeric Identifier	Data type name	Value range	Resolution	Length
22	Byte	-	-	1 Octet

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							

WORD

A Word is an ordered sequence of 2 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
23	Word	-	-	2 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							

DWORD

A DWord is an ordered sequence of 4 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
24	DWord	-	-	4 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							
Octet 3	3. Byte							
Octet 4	4. Byte							

LWORD

A LWord is an ordered sequence of 8 octets.

Numeric Identifier	Data type name	Value range	Resolution	Length
25	LWord	-	-	8 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							
Octet 3	3. Byte							
Octet 4	4. Byte							
Octet 5	5. Byte							
Octet 6	6. Byte							
Octet 7	7. Byte							
Octet 8	8. Byte							

TimeOfDay

This data type is composed of two elements of unsigned values representing the time of day and the date. The first element is an Unsigned32 data type and contains the number of milliseconds since midnight, where midnight = 0.

The second element is of type Unsigned 16 containing the number of completed days since January 1, 1984.

Numeric Identifier	Data type name	Value range	Resolution	Length
12	TimeOfDay	$0 \le i \le (2^{28}-1)$ ms $0 \le i \le (2^{16}-1)$ days	-	6 Octets

The time is interpreted as 32 bit value. The first 4 (MSB) bits have the value zero. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octets 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds since midnight
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octets 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	days since January 1st, 1984
Octets 6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

TimeDifference

This data type is composed of two elements of unsigned values and expresses the difference in time. The first element is an Unsigned32 data type that contains the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that contains the difference in days.

Numeric Identifier	Data type name	Value range	Resolution	Length
13	TimeDifference	$0 \le i \le (2^{32}-1)$ ms $0 \le i \le (2^{16}-1)$ days	-	4 or 6 Octets

The time is interpreted as 32 bit value. The first 4 (MSB) bits have the value zero. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octets 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of millisec-onds (of one day)
Octets 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octets 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octets 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octets 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	Number of days (op-tional)
Octets 6	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Date

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 ... 50 correspond to the years 2000 to 2050; the values 51 ... 99 correspond to the years 1951 to 1999.

Numeric Identifier	Data type name	Value range	Resolution	Length
50	Date	0 ms ≤ i ≤ 99 years	-	7 Octets (Unsigned16 + 5 x Unsigned8)

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	0...59999 millisec-onds
Octet 2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 3	res	res	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0...59 mi-nutes
Octet 4	SU	res	res	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0...23 hours
Octet 5	day of week			day of month					1...7 day of week 1...31 day of month
	2^{2}	2^{1}	2^{0}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Bits	7	6	5	4	3	2	1	0	
Octet 6	res	res	2^5	2^4	2^3	2^2	2^1	2^0	1...12 month
Octet 7	res	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0...99 years

SU: Standard Time = 0; Daylight Saving Time = 1

day of week: 0 Monday = 1; Tuesday =2.....Sunday = 7

res = reserved

TimeOfDay without date indication

This data type consists of one element of an unsigned value and expresses the time of day without date indication. The element is an Unsigned32 data type and contains the time after midnight in milliseconds.

Numeric Identifier	Data type name	Value range	Resolution	Length
52	TimeOfDay without date indication	$0 \le i \le (2^{28}-1)$	ms	4 Octets (Unsigned32)

The time is interpreted as 32 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds since midnight
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

TimeDifference with Date indication

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the difference in number of days.

Numeric Identifier	Data type name	Value range	Resolution	Length
53	TimeDifference with date indication	$0 \le i \le (2^{32}-1)$ ms $0 \le i \le (2^{16}-1)$ days	ms, days	6 Octets (Unsigned32 + Unsigned16)

The time is interpreted as 32 bit value. The date indication is coded as 16 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds (of one day)
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
Octet 4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Octet 5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Number of days
Octet 6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	

TimeDifference without Date indication

This data type consists of one element of an unsigned value that expresses the difference in time. he element is an Unsigned32 data type providing the fractional portion of one day in milliseconds.

Numeric Identifier	Data type name	Value range	Resolution	Length
54	TimeDifference without date indication	$0 \le i \le (2^{32}-1)$ ms	ms, days	4 Octets (Unsigned32)

The time is interpreted as 32 bit value.

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds (of one day)
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

NetworkTime

This data type is based on the RFC 1305 standard and composed of two unsigned values that express the network time related to a particular date. Its semantic has changed in IEC 61158-6:2003.

The first element is an Unsigned32 data type that provides the network time in seconds since 1.1.1900 0.00,00(UTC) or since 7.2.2036 6.28,16(UTC) for time values less than 0x9DFF4400, which represents the 1.1.1984 0:00,00(UTC). The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s. Rollovers after 136 years are not automatically detectable and are to be maintained by the application.

Numeric Identifier	Data type name	Value range	Resolution	Length
58	NetworkTime	Byte 1 to 4: $0 \le i \le (232-1)$ Byte 5 to 8: $0 \le i \le (232-1)$	S $(1/2^{32})$ s	8 Octets (Unsigned32 + Unsigned32)

Bits	7	6	5	4	3	2	1	0	
Octet 1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Number of seconds since 1.1.1900. Rollover after 136 years. Thus, next would be 7.2.2036.
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional portion of seconds: $1/2^{32}$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

The Time Stamp mechanism in PROFIBUS DP is using this data type. However, the semantic is slightly different:

- The least significant Bit of the fractional portion (2^0) is device internally used to indicate a synchronized or unsynchronized state of the clock time.

NetworkTimeDifference

This data type is composed of an integer value and of an unsigned value that express the difference in network time. The first element is an Integer32 data type that provides the network time difference in seconds. The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s.

Numeric Identifier	Data type name	Value range	Resolution	Length
59	NetworkTimeDifference	Byte 1 to 4: $-2^{31} \le i \le (2^{31}-1)$ Byte 5 to 8: $0 \le i \le (2^{32}-1)$	S $(1/2^{32})$ s	8 Octets (Integer32 + Unsigned32)

Bits	7	6	5	4	3	2	1	0	
Octet 1	SN	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Signed number of seconds
Octet 2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Octet 5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fractional portion of seconds: $1/2^{32}$ s
Octet 6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Octet 7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Octet 8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

See also

PROFIdrive-specific data types (Page 2410)

4.5.11.2 Profile-specific PROFIBUS/PROFINET data types (only available in English)

Existing PROFIBUS/PROFINET profile specific data types

This topic contains the profile specific data types of PROFIBUS/PROFINET.

Float32+Unsigned8 (former "DS33")

This data structure consists of the value and the status of a Float32 parameter. The parameter can be an input or output..

Numeric Identifier	Data type name	Value range	Resolution	Length
101	Float32 +Unsigned8	See Float32 and Unsigned8	-	5 Octets

SN: sign 0 = positive, 1 = negative

Bits	7	6	5	4	3	2	1	0
Octets 1	Exponent (E)							
	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
Octets 2	(E)	Fraction (F)						
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
Octets 3	Fraction (F)							
	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
Octets 4	Fraction (F)							
	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}
Octet 5	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Unsigned8+Unsigned8 (former "DS34")

This data structure consists of the value and the status of the Unsigned8 parameter. The parameter can be an input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
102	Unsigned8 + Unsigned8	See Unsigned8	-	2 Octets

In two's complement; the most significant bit (MSB) is the bit after the sign (SN) in the first Byte.

Bits	7	6	5	4	3	2	1	0
Octet 1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Octet 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

OctetString2+Unsigned8 (former "DS35")

This data structure consists of the value and the status of the OctetString parameter. The parameter can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
103	OctetString2 + Unsigned8	see OctetString2 and Unsigned8	-	3 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	1. Byte							
Octet 2	2. Byte							
Octet 3	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Unsigned16_S

This data structure consists of the value and the status embedded in an unsigned 16 data type. The parameter using this data type can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
104	Unsigned16_S	0 to 2^{13} + 0 to 3	1	2 Octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
Octets 2	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

Integer16_S

This data structure consists of the value and the status embedded in an integer 16 data type. The parameter using this data type can be input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
105	Integer16_S	- 2^{12} to 2^{12} –1 + 0 to 3	1	2 Octets

SN: sign 0 = positive, 1 = negative

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
Octets 2	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

Unsigned8_S

This data structure consists of the value and the status embedded in an Unsigned8 data type. The parameter using this data type can be an input or output.

Numeric Identifier	Data type name	Value range	Resolution	Length
106	Unsigned8_S	0 to 2^5 + 0 to 3	1	1 Octet

Bits	7	6	5	4	3	2	1	0
Octets 1	2^5	2^4	2^3	2^2	2^1	2^0	St1	St0

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

OctetString_S

This data structure consists of the value and the status embedded in an octet string data type. The parameter using this data type can be an input or output.

In the scope of this profile, this is the preferred data type for digital values. It contains the value of a binary channel as a bit in a value field (ch(x)), and a status information in a status field (st1(x) and st0(x)) coded as shown below. If a channel is unused, its position in the value field and in the status field has to be set to 0.

The value field and the status field are packed into an OctetString with the bit coding as shown below. This data type is defined as OctetString_S.

Numeric Identifier	Data type name	Value range	Resolution	Length
107	OctetString_S	See OctetString and below	-	n Octets

Bits	bit7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Octet 1	ch(8)	ch(7)	ch(6)	ch(5)	ch(4)	ch(3)	ch(2)	ch(1)
***	***	***	***	***	***	***	***	***
Octet m	ch(n)	ch(n-1)	ch(n-2)	ch(n-3)	ch(n-4)	ch(n-5)	ch(n-6)	ch(n-7)
Octet m+1	st1(4)	st0(4)	st1(3)	st0(3)	st1(2)	st0(2)	st1(1)	st0(1)
***	***	***	***	***	***	***	***	***
Octet 3*m	st1(n)	st0(n)	st1(n-1)	st0(n-1)	st1(n-2)	st0(n-2)	st1(n-3)	st0(n-3)

ch(x) value for channel x; st(x) status information for channel x; 1<x≤n

st1 (bit1)	st0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

F message trailer with 4 octets

This data structure consists of the status/control byte, consecutive number, and 2 byte CRC parameters in PROFIsafe's V1-mode or status/control byte and 3 byte CRC parameters in PROFIsafe's V2- mode. This data type can be associated with input or output data up to 12 byte. So far the data type "OctetString" with numeric identifier "10" has been used for the coding of the F message trailer. For new product developments the new data type specification shall be considered.

Numeric Identifier	Data type name	Value range	Resolution	Length
110	F message trailer with 4 octets	PROFIsafe: V1-mode and V2-mode	-	4 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	Consecutive number (V1-mode) or High-octet CRC(V2-mode) *)							
Octet 3	CRC *)							
Octet 4	Low-byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

F message trailer with 5 octets

This data structure consists of the status/control byte and 4 byte CRC parameters in PROFIsafe's V2-mode. This data type can be associated with input or output data up to 122 byte.

Numeric Identifier	Data type name	Value range	Resolution	Length
111	F message trailer with 5 octets	PROFIsafe: V2-mode	-	5 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	1. byte CRC (most significant octet) *)							
Octet 3	2. byte CRC *)							

Bits	7	6	5	4	3	2	1	0
Octet 4	3. byte CRC *)							
Octet 5	4. byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

F message trailer with 6 octets

This data structure consists of the status/control byte, consecutive number and 4 byte CRC parameters in PROFIsafe's V1-mode. This data type can be associated with input or output data up to 122 byte. So far the data type "Octet String" with numeric identifier "10" has been used for the coding of the F message trailer. For new product developments the new data type specification shall be considered.

Numeric Identifier	Data type name	Value range	Resolution	Length
111	F message trailer with 6 octets	PROFIsafe: V1-mode	-	6 Octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	Consecutive number							
Octet 3	1. byte CRC (most significant octet) *)							
Octet 4	2. byte CRC *)							
Octet 5	3. byte CRC *)							
Octet 6	4. byte CRC (least significant octet) *)							

* Byte ordering according to IEC 61158-5 Type 3

See also

PROFIdrive-specific data types (Page 2410)

4.6 TO Path Object

Preface

Contents

This document is part of the **System and Function Descriptions documentation package**.

Scope

This manual is valid for SIMOTION SCOUT in conjunction with the SIMOTION Cam, Path or Cam_ext technology package for product version V5.3.

Chapters in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

Chapters in this manual

The following describes the purpose and objectives of the manual:

- **Overview of Path Interpolation**
  This chapter contains an overview of the TO functionality and a definition of the terms.
- **Basics of Path Interpolation**
  This chapter explains the basic setting options and functions of the Path Interpolation technology object.
- **Configuring the Path Object**
  This chapter explains the configuration procedure with reference to various tasks.
- **Sample Project for the Path Interpolation**
  This chapter contains a sample project for the path interpolation.
- **Programming/Homing Path Interpolation**
  This chapter explains the commands and functions in greater detail.
- **Appendix A**
  This chapter explains the specific kinematics with TrafoID 1001.
- **Index**
  Keyword index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.3:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C

- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.6.1 Fundamental safety instructions

4.6.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

4.6.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that can be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.6.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.6.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.6.2 Overview of Path Interpolation

4.6.2.1 Overview of Functions

As of Version V4.1, SIMOTION provides **path interpolation** functionality. This functionality enables up to three path axes to travel along paths. In addition, a position axis can be traversed synchronously with the path.

Paths can be combined from segments with linear, circular, and polynomial interpolation in 2D and 3D.

The path interpolation technology is provided by the **path object**, which represents an independent functionality.

The TO pathObject is interconnected to **path axes**, and can also be interconnected to a position axis.

The dynamic response parameters are predefined on the path motion.

The path motions of individual path commands can be blended together to form a complete path with no intermediate stop.

The machine kinematics are adapted to the Cartesian axes of the path coordinate system via the kinematic transformation.

As of V4.1.2, the functionality is available for the synchronization of the path motions with an externally specified position value, e.g. with the motion of a conveyor. This supports the system handling for the moved conveyor.

The **path interpolation** technology contains transformations for the following orthogonal kinematics:

- Cartesian linear aches
- SCARA
- 2D roller picker
- 3D roller picker
- 2D delta picker
- 3D delta picker
- 2D articulated arm
- 3D articulated arm
- 2D swivel arm
- 3D cylindrical robot

As of V4.4, it is also possible to create your own transformations with the aid of the 2D or 3D user function.

During a path motion, a position axis can be traversed synchronously with the path. The axis can approach a programmed, axis-specific target position synchronously or it can execute a motion according to the path length, thus enabling implementation of path-length-based output cams and measuring inputs.

Path interpolation functions are required for such applications as feeding or withdrawal of materials to or from a machine.

The application of commands for individual path segments requires a total path plan in the user program or application.

CNC programming according to DIN 66025 is not supported by SIMOTION.

4.6.2.2 Terminology

Axis coordinates

Coordinates of the path axes or the position axis with path-synchronous motion.

Path-axis interface

Interfaces for bidirectional data exchange between the path object and interconnected path axes.

Path axis

Axis that can execute a path motion along with other path axes via a path object.

Path motion

Motion resulting from the interpolation of a path motion command; output on path axes.

Path interpolation

Motion along a path with an assignable dynamic response.

Path interpolation generates the traversing profile for the path, calculates the path interpolation points in the interpolation cycle, and uses the kinematic transformation to derive the axis setpoints for the interpolation cycle points.

Path interpolation grouping

Several path and positioning axes connected by a path object or interpolation.

Path object

The path object provides the functionality for the path interpolation and for other tasks connected with the path interpolation. It also contains the kinematics transformations implemented in the system.

Path control panel

The path control panel enables the control and monitoring of path objects without a user program. It is mainly used to commission kinematic transformations.

Continuous-path control

Motion along a path at a definable velocity.

This can include velocity-based smoothing of the segment transitions by insertion of transition segments.

Basic coordinate system (BCS)

Coordinate system of path interpolation. A clockwise, rectangular coordinate system in accordance with DIN 66217 is used.

Motion sequence

Permits the coupling of the kinematic end point with a coupled OCS and so, for example, the coupling with the actual value of a conveyor. This means, for example, a product can be taken from a running conveyor or placed there.

As of SIMOTION V4.1.2, position details in the motion commands can be related optionally to the basic coordinate system or to an **object coordinate system** (**OCS**).

Motion sequence reference value (trackingInPosition)

The value made available to the **TrackingIn interface** of the path object by another technology object. This can be, for example, the actual value of an external encoder.

Motion sequence value (trackingPosition)

The current position of a **coupled OCS** with reference to the **OCS reference position**

Frame transformation

A frame transformation describes the position of a coordinate system relative to another coordinate system that defines, for example, the **OCS reference position** relative to the basic coordinate system of the path object. The frame transformation consists of **translations** along the X-, Y-, and Z-axes and **rotations** at the individual axes.

For the transformation, the displacements are performed first and then the rotations in the following order:

- **Roll** at the X axis
- **Pitch** at the (already turned) Y axis
- **Yaw** at the (already twice-turned) Z axis

Main plane

X-Y, Y-Z or Z-X plane or a parallel plane. The third coordinate is not evaluated.

Interface for path-synchronous motion

Interface for bidirectional data exchange between the path object and an interconnected position axis for path-synchronous motion.

Cartesian axes

Axes X, Y, and Z of the path object

Kinematics

The term "kinematics" in the context of robots and handling devices in motion control systems refers to the abstraction of a mechanical system onto the variables relevant for motion and motion control, i.e. the motion-capable elements (articulations) and their geometric positions relative to each other (arms).

Kinematic transformation, kinematic adaptation

Conversion of specifications in Cartesian coordinates to specifications for individual path axes, and vice versa.

Circular path

Path in 2D or 3D that describes a circle or an arc path.

Linear path

Path in 2D or 3D that describes a straight path.

Coupled OCS

An **object coordinate system (OCS)** coupled synchronously to the **trackingIn interface**.

Object coordinate system (OCS)

As of SIMOTION V4.1.2, in addition to the **base coordinate system (BCS)**, object coordinate systems (OCS) with the path object are also available. Path motions can be specified either in the BCS or in the OCS. The object coordinate systems are defined in their reference position using frame transformations for the BCS. They can be coupled with a specified motion value in the x direction of the OCS on the **TrackingIn interface**.

OCS reference position

Position of the OCS for the motion sequence value equal to zero. The OCS reference position for the BCS is defined using a **frame transformation**.

Polynomial path

Path in 2D or 3D that describes a polynomial segment.

Synchronous motion, path-synchronous motion

Synchronous coupling of an axis with a path motion; output on a position axis.

TrackingIn interface

The **trackingIn** input interconnection interface of the path object can be interconnected with another TO that provides an output interface with motion information. This can be, for example, the motion setpoint or actual value of an axis or the actual value of an external encoder.

4.6.3 Basics of Path Interpolation

4.6.3.1 Path interpolation

The **path interpolation** technology provides functionality for interpolating linear, circular, and polynomial paths in two dimensions (2D) and three dimensions (3D).

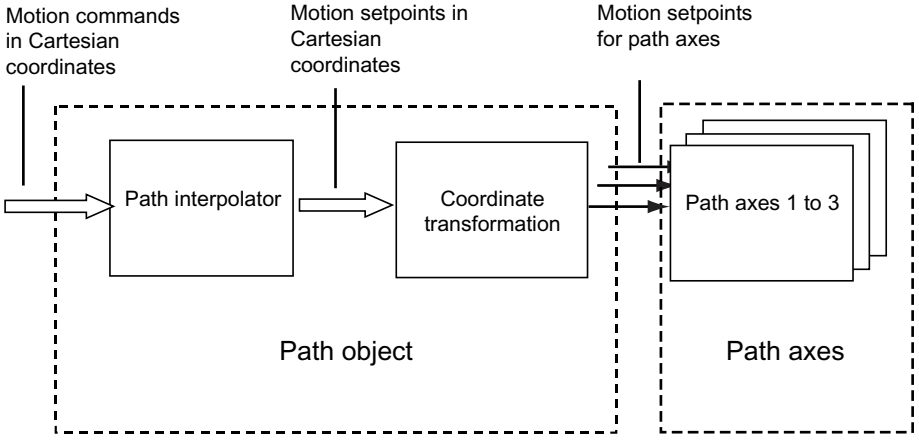


Figure 4-501 Role and basic principle of the path interpolator

Objects involved in path interpolation

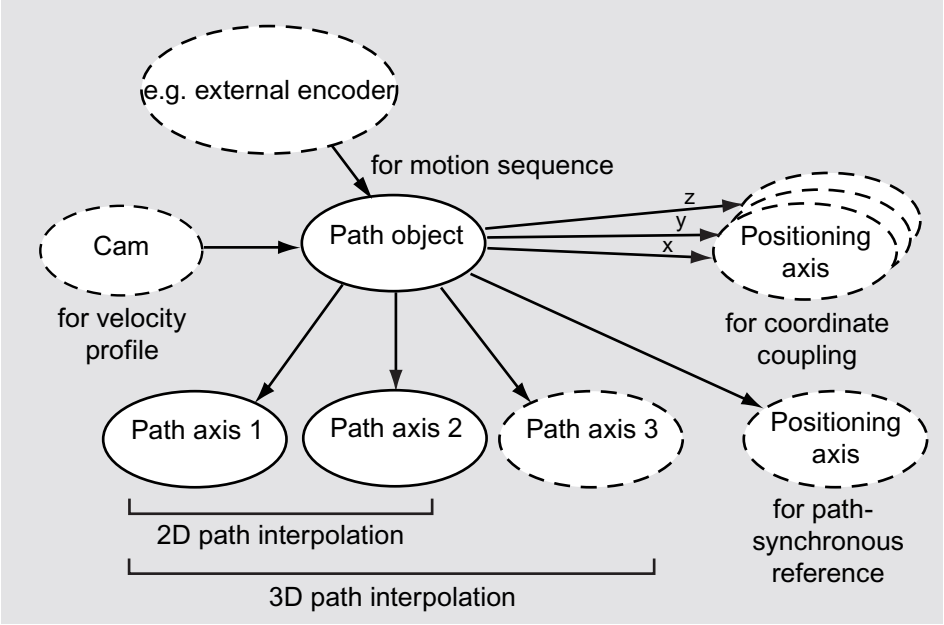


Figure 4-502 Objects involved in path interpolation

The path interpolation technology is made available in the Path Object technology object (TO Path Object).

The TO Path Object is interconnected with 2 or 3 path axes.

In addition, the TO Path Object can be interconnected with a positioning axis for path-synchronous motion and with positioning axes for connection to a coordinate. Likewise, it can be interconnected with a cam.

From V4.5 the TO Path Object can also be interconnected with an orientation axis.

The **TrackingIn** interface can be used to interconnect a technology object that provides motion information with a position (the motion sequence value), such as:

- External encoder
- Positioning axis

Role of the path axis

All single-axis functions can be executed on the path axis without limitations.

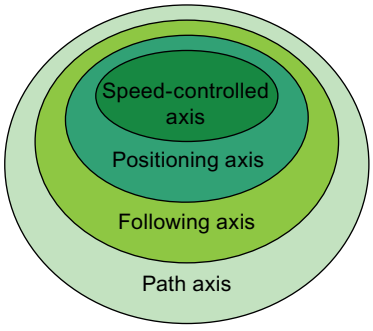


Figure 4-503 Role of the path axis

The path interpolation functionality is independent of the physical axis type. Path interpolation can be applied to electric axes, hydraulic axes, and stepper motor axes (real axes) as well as to virtual axes.

Inclusion of path interpolation in technology packages

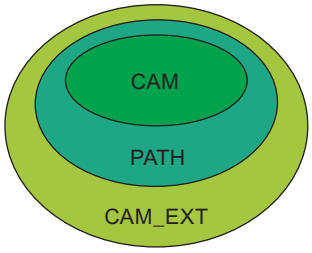


Figure 4-504 Inclusion of path interpolation in technology packages

Path functionality is made available in the **PATH** technology package, which also includes the functionality of the CAM technology package. The extensions include the TO Path Object and the TO Path Axis.

Thus, the **CAM_EXT** technology package also contains these object types.

For additional information, see **Motion Control Basic Functions**, "Available technology objects".

4.6.3.2 Coordinate system

The path interpolation functions require a Cartesian coordinate system. A clockwise, rectangular coordinate system in accordance with DIN 66217 is used.

The user programs in this right-handed system, irrespective of the real kinematics.

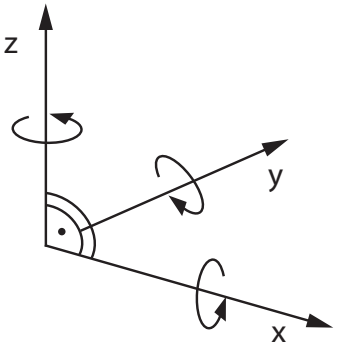


Figure 4-505 Cartesian coordinate system, right-handed system

Main planes

It is easy to program two-dimensional motions (2D) directly in one of the three main planes X-Y, Y-Z, or Z-X. In this case, the third coordinate remains constant and does not have to be programmed.

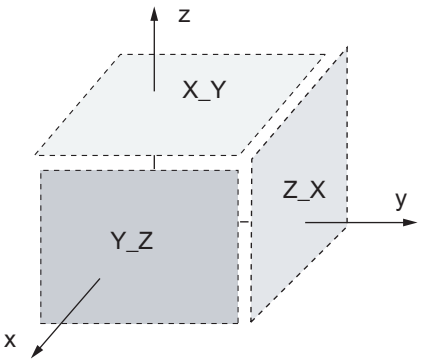


Figure 4-506 Main planes in 3D

4.6.3.3 Modulo properties

Both path axes and positioning axes can be used as modulo axes. However, no modulo range change for the path axis may occur in the path traversal area. The kinematic transformation does not take account of any modulo range change.

Consequently, only one modulo range of the path axis can be used for the traversal area on the path object. The activation of the path interpolation defines the modulo range for the path motion.

This means that the modulo transition of the axis must not be in the traversing range of the path motions. The modulo range and the modulo starting point as well as the position of the modulo range relative to the intended path travel range must be set appropriately, for example, using the settings for reference point and reference point offset during homing.

4.6.3.4 Units

All axis-related values are displayed in the quantity and unit of the assigned (interconnected) axes.

The Cartesian coordinates are indicated in a unit of length. The default setting for Cartesian values is [mm].

The default unit for rotary values, such as rotary angle, is [°] and calculated as degrees.

The transformation calculates directly with the numerical values. There is no unit conversion for transformations provided by the system. Thus, the same units must be used for the same base value, e.g, length specification.

Note

Use the same units for all objects associated with the path object that have the same reference quantities (e.g. linear axes in mm, rotary axes in °). Avoid, for example, the mixing of metric and non-metric units for the involved axes.

4.6.3.5 Path interpolation types

Path interpolation types

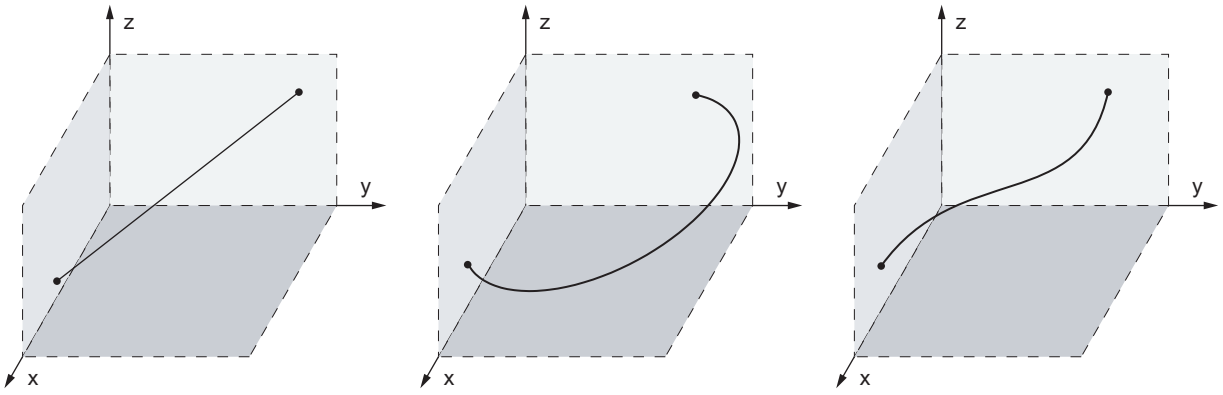


Figure 4-507 Examples of linear path in 3D, circular path in 3D, polynomial path in 3D

The following interpolation modes are available for the path object:

- Linear paths (Page 2480)
  - 2D in a main plane
  - 3D
- Circular paths (Page 2481)
  - 2D in a main plane with radius, end point, and orientation
  - 2D in a main plane with center point and angle
  - 2D with intermediate and end points
  - 3D with intermediate and end points
- Polynomial paths (Page 2485)
  - 2D in a main plane with explicit specification of geometric derivatives in the start point or with a geometrically continuous attachment
  - 3D with explicit specification of geometric derivatives in the start point or with a geometrically continuous attachment
  - 2D with explicit specification of polynomial parameters
  - 3D with explicit specification of polynomial parameters

The main plane (2D) or the 3D mode in which the path motion occurs can be specified with the **pathPlane** parameter of the interpolation command.

The third path coordinate perpendicular to the main plane is not changed in a 2D path.

Structure of commands for path interpolation

The following path interpolation commands are available:

- Linear interpolation: `_movePathLinear()`
- Circular interpolation: `_movePathCircular()`
- Polynomial interpolation: `_movePathPolynomial()`

These commands contain the following parameters:

- Specification of the object instance in `pathObjectType`
- Specification of the path plane in `pathPlane`
  This parameter is used to set the path plane. The main plane (2D) or the 3D mode in which the path motion should occur can be specified.
- Specification of the path mode in `pathMode`
  This parameter is used to set whether the value for the end point is specified as an absolute value or whether it is to be evaluated relative to the start point.
- Specification of the end point in `x, y, z`
- Specification of the blending mode in `blendingMode`
- Specification of the merge behavior in `mergeMode`
- Specification of the command transition in `nextCommand`
- Specification of the command ID in `commandId`

Specifications for the linear path only (_movePathLinear()):

(see Linear paths (Page 2480))

- None

Specifications for the circular path only (_movePathCircular())

(see Circular paths (Page 2481))

- Specification of the circle type in `circularType`
- Specification of the circle direction in `circleDirection`
- Specification of the intermediate point mode in `ijkMode`
- Specification of the intermediate point mode in `i, j, k`
- Specification of the arc angle in `arc`
- Specification of the circle radius in `radius`

Specifications for the polynomial path only (_movePathPolynomial())

(see Polynomial paths (Page 2485))

- Specification of polynomial mode in `polynomialMode`
- Specification of the vector components in `vector1x` to `vector4z`

Specifications for the dynamics

(see Path dynamics (Page 2491))

- Velocity profile in `velocityprofile`
- Velocity in `velocity`
- Acceleration in `positiveAccel`
- Deceleration in `negativeAccel`
- Jerk on start of acceleration in `positiveAccelStartJerk`
- Jerk at acceleration end in `positiveAccelEndJerk`
- Jerk on start of deceleration in `negativeAccelStartJerk`
- Jerk at deceleration end in `negativeAccelEndJerk`
- Selection of specific profile in `specificVelocityProfile`
- Specifies the velocity profile with a cam in `profileReference`
- Start point for specific profile in `profileStartPosition`
- End point for specific profile in `profileEndPosition`
- Adaptation to the axis dynamics in `dynamicAdaption`

Specifications for path-synchronous motion

(see Functionality of path-synchronous motion (Page 2505))

- Mode of path-synchronous motion in `wMode`
- Direction of path-synchronous motion in `wDirection`
- End point of path-synchronous motion in `w`

Details of the object coordinate system

(see Object coordinate system (OCS) on the path object (Page 2545))

- Specification of the coordinate system in `csType`
  This parameter is used to set whether the motion should be performed in the base coordinate system or in an object coordinate system.
- Specification of the object coordinate system in `csNumber`
  This parameter is used to set which object coordinate system should be used for the motion.

Linear paths

In the case of linear path interpolation, an end point is approached on a straight line starting from the current position.

Linear paths are traversed with the **_movePathLinear()** command.

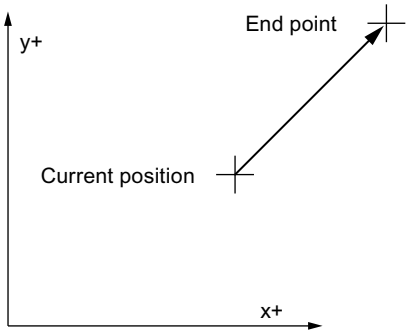


Figure 4-508 Example of a linear path

Example of a linear path in ST

In this example, the current position and the end point lie in the X-Y plane. Each end point is separated by 10 units in the positive direction from the current position along both axes.

```
myRetDINT :=
    _movePathLinear(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=relative,
        x:=10.0,
        y:=10.0
);
```

Circular paths

Circular paths

For a circular path, approach is made from the current position to a specified end point following an arc.

Circular paths are traversed with the **_movePathCircular()** command.

The arc can be specified using several modes. The **circularType** parameter specifies the mode to be used.

- Circular interpolation in a main plane with radius, end point, and orientation (Page 2482)
- Circular interpolation in a main plane with center point and angle (Page 2483)
- Circular interpolation with intermediate and end points (Page 2484)

If a circular path is not traversed because of the geometry, the **50002** error will be issued.

Circular path in a main plane with radius, end point, and orientation

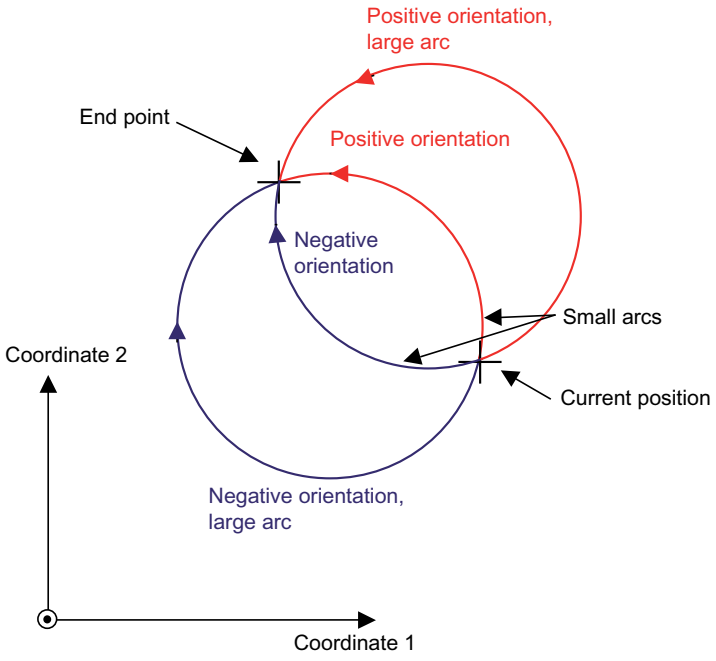


Figure 4-509 Circular path with radius, end point, and orientation

To perform circular interpolation in a main plane with specification of radius, end point, and orientation, you set **circularType:=WITH_RADIUS_AND_ENDPOSITION** in the **_movePathCircular()** command.

The end point is approached on a circular path starting from the current position. The current position and the end point lie in the same main plane. Circle radius, orientation (travel in the positive or negative direction of rotation), and travel on large or small arcs are specified in the command.

The end point position is entered in the x, y, and z parameters.

Example of a circular path with radius, end point, and orientation

In this example, the current position and the end point lie in the X-Y plane. The end point is separated from the current position by -10 units along the x-axis and 10 units along the y-axis. The large circle is traveled in the positive direction.

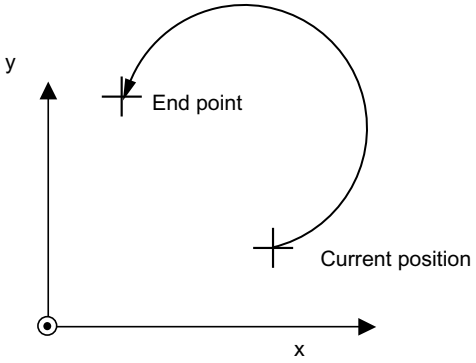


Figure 4-510 Example of circular path with radius, end point, and orientation

```
myRetDINT :=
    _movePathCircular(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        circularType:=WITH_RADIUS_AND_ENDPOSITION,
        circleDirection:=LONG_RUN_POSITIVE,
        pathMode:=RELATIVE,
        x:=-10.0,
        y:=10.0,
        radius:=12.0
);
```

Circle using center and angle

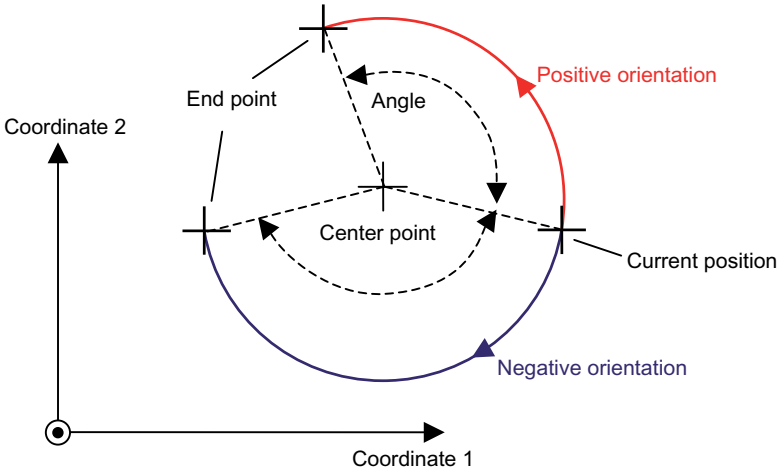


Figure 4-511 Circular path with center point and angle

To perform circular interpolation starting from the current position in a main plane with specification of center point and angle, you set **circularType:= BY_CENTER_AND_ARC** in the **_movePathCircular()** command.

The center point of the circle, the angle to be traveled, and the orientation (travel in the positive or negative direction of rotation) are specified in the command.

The position of the center point of the circle is entered in the **i**, **j**, and **k** parameters.

You use the **ijkMode** parameter to set whether the circle center point coordinates are entered absolutely or relative to the start point or whether the setting in the **pathMode** parameter should be used.

Example of a circular path with center point and angle

In this example, the center point is separated by -10 units from the current position along the X-axis. An angle of 90 degrees in the positive direction is traveled.

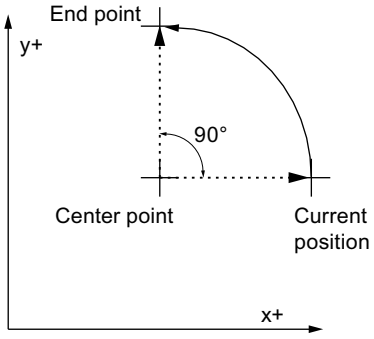


Figure 4-512 Example of a circular path with center point and angle

```
retval := _movePathCircular(
    pathObject := pathIpo,
    pathPlane := X_Y,
    circularType := BY_CENTER_AND_ARC,
    circleDirection := POSITIVE,
    ijkMode := RELATIVE,
    i := -10.0, j := 0.0,
    arc := 90.0
);
```

Circular path using intermediate point and end point

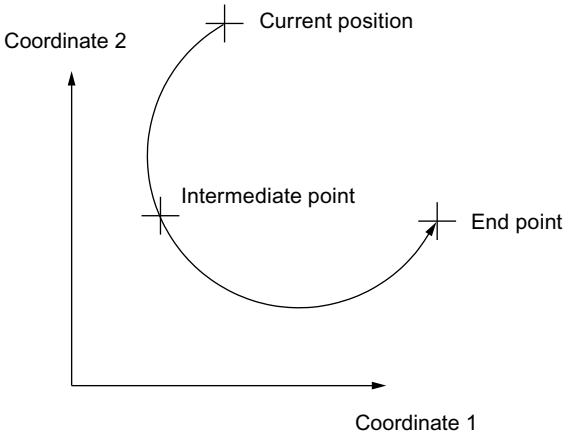


Figure 4-513 Circular path with intermediate and end points

To perform circular interpolation starting from the current position over an intermediate point to the end point, you set **circularType:=OVER_POSITION_TO_ENDPOSITION** in the **_movePathCircular()** command.

The current position, intermediate point, and end point specify the plane for the circular path.

The end point position is entered in the **x**, **y**, and **z** parameters.

The intermediate point is entered in the **i**, **j**, and **k** parameters.

You use the **ijkMode** parameter to set whether the intermediate point coordinates are to be evaluated absolutely or relative to the start point or according to the setting in **pathMode** of the end point.

Example of a circular path with intermediate point and end point

In this example, the end point of the circle is separated by 10 units from the current position in the X-direction. Each intermediate point is separated by 5 units in the X-, Y- and Z-direction from the current position.

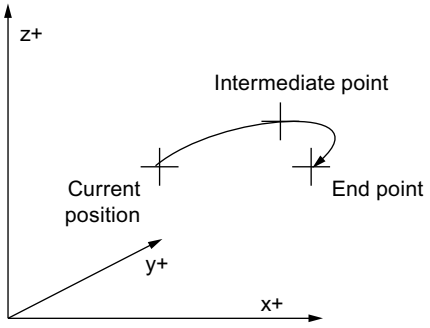


Figure 4-514 Example of a circular path with intermediate and end points

```
retval := _movepathcircular(
    pathObject := pathIpo,
    pathPlane := X_Y_Z,
    circularType := OVER_POSITION_TO_ENDPOSITION,
    pathMode := RELATIVE,
    x:=10.0, y:=0.0, z:=0.0,
    ijkMode := RELATIVE,
    i:=5.0, j:=5.0, k:=5.0
);
```

Polynomial paths

Polynomial paths

A polynomial segment enables you to achieve a constant-velocity and constant-acceleration transition between two geometry elements and to make use of user-programmable curve shapes, e.g. from a higher-level CAD system.

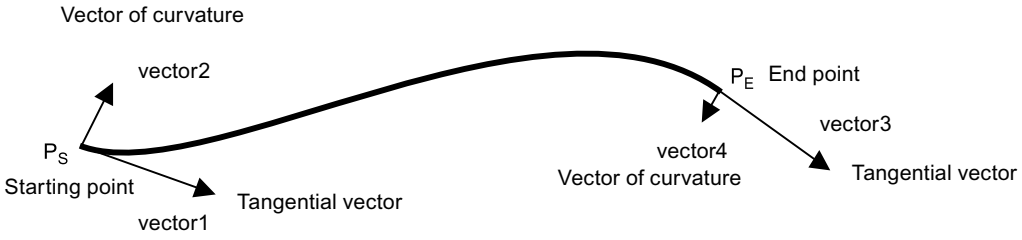
In addition to the implicit start point (P_S) of the polynomial, the end point (P_E) as well as four three-dimensional vectors for defining the polynomial coefficients are specified in the command parameters of the **_movePathPolynomial()** command

The vectors are entered in the command using their components. Thus, for example, **vector1** is entered with command parameters **vector1x**, **vector1y**, and **vector1z**.

The polynomial can be defined in three ways:

- Direct specification of the polynomial coefficients (Page 2487)
- Explicit specification of starting point data (Page 2487)
- Attach continuously (Page 2489)

For the two **explicit specification of the start point data** and **attach continuously** types, the derivatives at the start and end points of the polynomial are required. They can be determined using integrated functions.



First geometric derivative (tangential vector) $\dot{P} = \frac{dP}{ds}$; $|\dot{P}| = 1$

Second geometric derivative (vector of curvature) $\ddot{P} = \frac{d^2P}{ds^2}$

Figure 4-515 Polynomial description by specification of the geometric derivatives

Smooth-path transition between two linear paths

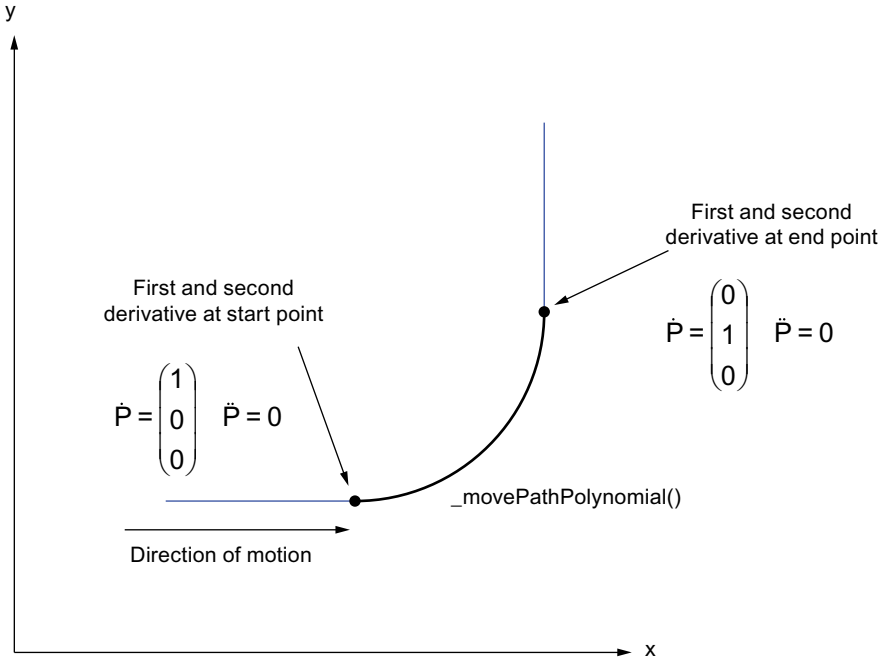


Figure 4-516 Specification of derivatives for polynomial transition between two linear paths

The derivatives at the end point of the previous geometry and at the start point of the following geometry can be calculated with the **_getLinearPathGeometricData()**, **_getCircularPathGeometricData()** and **_getPolynomialPathGeometricData()** commands.

If a polynomial path is not traversed because of the geometry, the **50002** error will be issued.

Effect of the start and end points

When polynomials are used, they must be linked smoothly to the previous and subsequent path segment. Depending on the choice of the start and end points, there are consequently different polynomial curves that can deviate significantly from a circular path.

The following graphic shows the curve of a polynomial path with different start points:

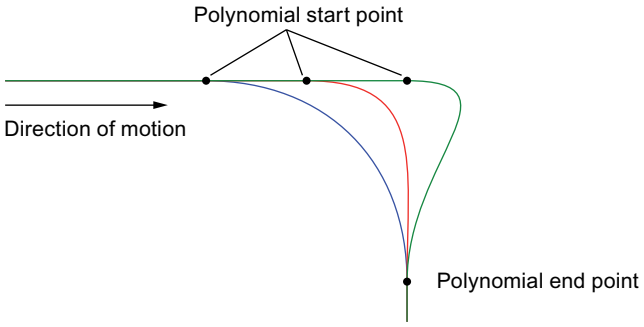


Figure 4-517 Behavior of a polynomial path with different start points

Polynomial path - direct specification of the polynomial coefficients

For the polynomial specification using (**polynomialMode:=SETTING_OF_COEFFICIENTS()**) polynomial coefficients, the polynomial path is determined using a function of the fifth degree:

$P = A_0 + A_1 \bullet p + A_2 \bullet p^2 + A_3 \bullet p^3 + A_4 \bullet p^4 + A_5 \bullet p^5$, $p \in [0,1]$

- **vector1**: A_2
- **vector2**: A_3
- **vector3**: A_4
- **vector4**: A_5
- A_0 and A_1 result from the start point and end point, and the predefined coefficients. For the parameter area indicated above, this means:
  - A_0 = start point
  - A_1 = end point - start point - A_2 - A_3 - A_4 - A_5

Polynomial paths - explicit specification of the starting point data

For the **polynomialMode:=SPECIFIC_START_DATA** setting and the explicit specification of the starting point data, the two geometric derivatives at the start point must also be specified for the derivatives at the end point of the polynomial.

The derivatives must be specified as follows:

- **vector1**: First geometric derivative/tangential vector in start point
- **vector2**: Second geometric derivative/curvature vector in start point
- **vector3**: First geometric derivative/tangential vector in end point
- **vector4**: Second geometric derivative/curvature vector in end point

Example of a polynomial path with explicit specification of the starting point data

This example connects a linear path and a circular path:

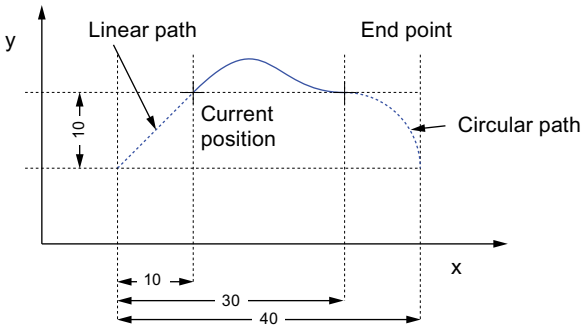


Figure 4-518 Example of a polynomial path with explicit specification of the starting point data

The two derivatives in the starting point of the polynomial must be calculated first. The **_getLinearPathGeometricData()** function used for this purpose calculates the two derivatives for the end point of the straight line (starting point of the polynomial) using the coordinates of the straight line.

The two derivatives of the polynomial end point are then determined. The **_getCircularPathGeometricData()** command used for the calculation uses the starting point of the arc (end point of the polynomial) as basis.

```
// StartPoly must be defined as
// StructRetGetLinearPathGeometricData
// EndPoly must be defined as
// StructRetGetCircularPathGeometricData
StartPoly :=
    _getLinearPathGeometricData(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=ABSOLUTE,
        xStart:=10.0,
        yStart:=10.0,
        xEnd:=20.0,
        yEnd:=20.0,
        pathPointType:=END_POINT
);
EndPoly :=
    _getCircularPathGeometricData(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        circularType:=WITH_RADIUS_AND_ENDPOSITION,
        circleDirection:=NEGATIVE,
        pathMode:=ABSOLUTE,
        xStart:=40.0,
        yStart:=20.0,
        xEnd:=50.0,
        yEnd:=10.0,
        radius:=10.0,
        pathPointType:=START_POINT
);
```

```
myRetDINT :=
    _movePathPolynomial(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=ABSOLUTE,
        polynomialMode:=SPECIFIC_START_DATA,
        x:=40.0,
        y:=20.0,
        vector1x:=StartPoly.firstGeometricDerivative.x,
        vector1y:=StartPoly.firstGeometricDerivative.y,
        vector2x:=StartPoly.secondGeometricDerivative.x,
        vector2y:=StartPoly.secondGeometricDerivative.y
        vector3x:=EndPoly.firstGeometricDerivative.x,
        vector3y:=EndPoly.firstGeometricDerivative.y,
        vector4x:=EndPoly.secondGeometricDerivative.x,
        vector4y:=EndPoly.secondGeometricDerivative.y
);
```

Polynomial paths - attach continuously

Polynomial paths can be attached continuously to a previous path segment using the **polynomialMode:=ATTACHED_STEADILY** setting. Because the geometric derivatives at the start point of the polynomial are taken from the predecessor geometry, only the first and the second derivative at the end point needs to be specified directly.

The two derivatives for the polynomial command are specified as following:

- **vector1**: First geometric derivative/tangential vector in end point
- **vector2**: Second geometric derivative/curvature vector in end point

If the geometric derivative cannot be determined in the start point (if no current motion is available), the command is not executed and error message 50002 "Calculation of the geometry element not possible, reason 3" is output.

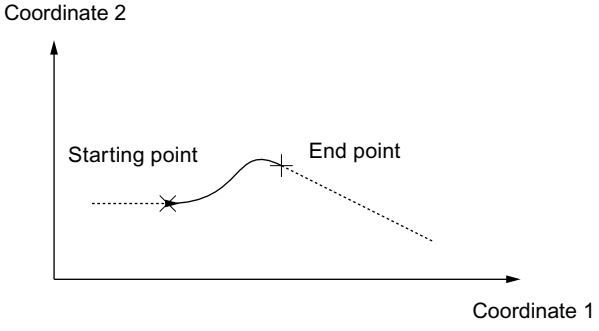


Figure 4-519 Polynomial path - attach continuously

Example of a polynomial path attached continuously

This example connects two straight lines using a polynomial.

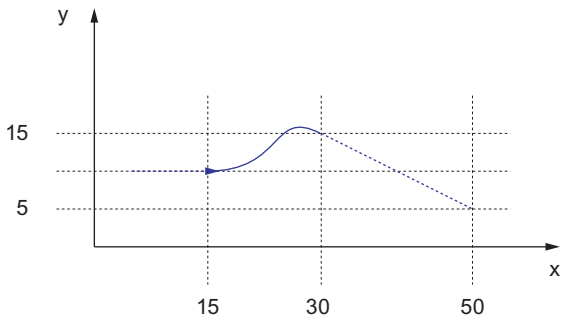


Figure 4-520 Example for the continuous attachment of polynomial paths

The two derivatives at the end point of the polynomial must be calculated first. The **_getLinearPathGeometricData()** function used here returns a structure with the derivatives. The function calculates the derivatives for the start point of the straight lines (end point of the polynomial) using the start and end point coordinates of the straight lines.

```
// EndPoly must be defined as
// StructRetGetLinearPathGeometricData
EndPoly :=
    _getLinearPathGeometricData(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=ABSOLUTE,
        xStart:=30.0,
        yStart:=15.0,
        xEnd:=50.0,
        yEnd:=5.0,
        pathPointType:=START_POINT
);
myRetDINT :=
    _movePathPolynomial(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=ABSOLUTE,
        polynomialMode:=ATTACHED_STEADILY,
        x:=30.0,
        y:=15.0,
        vector1x:=EndPoly.firstGeometricDerivative.x,
        vector1y:=EndPoly.firstGeometricDerivative.y,
        vector2x:=EndPoly.secondGeometricDerivative.x,
        vector2y:=EndPoly.secondGeometricDerivative.y
);
```

4.6.3.6 Path dynamics

Path dynamics

The path dynamics can be specified through preset dynamic values or a dynamic response profile.

The dynamic limits of the individual axes for motion along the path can also be taken into consideration.

An error message is output if the dynamic values are exceeded.

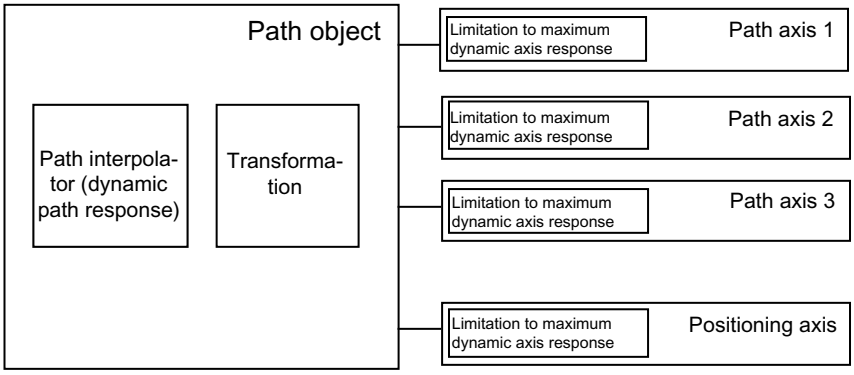


Figure 4-521 Path dynamics during path interpolation and dynamic limiting on the axis

Preset path dynamics

The path dynamics can be specified in three different ways in the respective motion command:

- Preset path dynamics via command parameters
- Preset path dynamics via velocity profile/cam
- Preset path dynamics via DynamicsIn

Preset path dynamics via command parameters

The dynamic values (velocity, acceleration, and, if applicable, jerk) are explicitly specified in the velocity profile type.

The path interpolator calculates the velocity profile for the path motion. Criteria for calculating the velocity profile include:

- The dynamic values for velocity, acceleration, and jerk specified in the path motion command
- The type of velocity profile set in **velocityProfile**:
  - **TRAPEZOIDAL**: Without jerk limitation; the path will be traveled with constant acceleration and deceleration.
  - **SMOOTH**: With jerk limitation; the path will be traveled with smooth acceleration and deceleration curve.

Preset path dynamics via velocity profile/cam

The path object can be interconnected with a cam for specifying a velocity profile.

Velocity as well as the derived values for acceleration and, if applicable, jerk, are taken from the velocity profile.

The base value (domain) is the path length. To rule out rounding errors in the path length calculation and to enable optimized calculation of profiles over more than one motion, parameters can be programmed simultaneously for the start and end points of the cam domain of the respective motion.

At the command end, the dynamics specified in the profile are also applied to the motion.

If additional follow-on motions are programmed, these dynamics are also applied to the transition to the new motion command. Possible settings for the path behavior at the motion end are ignored.

If no additional follow-on motions are programmed or if the motion is to stop at the command end, the dynamics in the profile should be selected such that a stop at the motion end is possible; a velocity of 0 with a braking dynamic that can be achieved with certainty.

In addition, the profile dynamics are limited by the dynamic values for the individual commands, taking into account the preassigned velocity profile type.

Preset path dynamics via DynamicsIn

From V4.3 and higher, the path dynamics can be specified via DynamicsIn. The position specified in the DynamicsIn vector refers to the path/path length. The position must be specified with the velocity and the acceleration at this path point. These values must be provided to the TO cyclically using system variables or TO interconnection.

The dynamic planning and dynamic response adaptation of the TO path is completely deactivated, i.e. there is no limit and no monitoring.

The dynamic limitations of the axes are still effective.

Note

The path movements are not blended with active DynamicsIn

The settings POLYNOMIAL and CIRCULAR have no effect when PathDynamicsIn := TO_Connection.

With deactivated DynamicsIn, the path movements are blended. With active DynamicsIn, programmed geometric blending is ignored and you can calculate the geometric blending segments yourself and specify them for the system. Circular and polynomial commands are available for this.

See also

Blending and substitution with insertion of intermediate segments (Page 2499)

Limiting the path dynamics

Technological limiting

The individual axis setpoints resulting from the path interpolation are limited to the dynamic limits specified for each path axis and positioning axis involved in path-synchronous motion.

The dynamic values of the axis for the path object are only taken into account if this has been programmed accordingly (command parameter **blendingMode := ACTIVE_WITH_DYNAMIC_ADAPTION** and/or **dynamicAdaption <> INACTIVE**).

Path velocity limiting, path acceleration limiting, and path jerk limiting can be specified in the **limitsOfPathDynamics** system variables. Changes in the system variables take effect immediately.

The maximum dynamic values over the path result from the lesser of the dynamic parameters set in the command, the dynamic limits on the path specified via the system variables (**limitsOfPathDynamics**), and, if programmed, the maximum dynamic values of the axes along the path.

Note that the path velocity for active dynamic adaptation, possibly also reduced, if the dynamic limits of the axes would not be violated even without active dynamic adaptation.

Because the reserve used for the active dynamic adaptation, the maximum possible dynamic path response will not always be attained.

The limitation of dynamic values to the individual axes can lead to dynamic and distance deviations on the path. Path dynamics and axis limits should be set so that the axis limits are not exceeded during path motion.

Allowance for dynamic limits of path axes

A reference to the dynamic limits of the axis can be established in the path object using the **dynamicAdaption** command parameter. The following settings are possible:

- No allowance for maximum dynamic values of path axes (**INACTIVE**)
  With this setting, the axial limits are not taken into account within the path interpolation. However, path axis limiting is still active and, if a violation occurs, a setpoint-side path error can result.
  The setting is useful if:
  - There are no transformed dynamic values
  - It can be ensured in advance (e.g. during commissioning) that the axial limit values are not exceeded
  - The axial limits have been taken into account through an application, e.g. through calculation of an optimized velocity profile
  - Superimposed axis motions occur
- Reduction in the maximum path dynamics according to the maximum dynamic values of path axes (**ACTIVE_WITH_CONSTANT_LIMITS**)
  The velocity and acceleration of the path is limited in the path interpolator to the maximum values in the Cartesian coordinates calculated from the maximum value settings of the individual path axes.
  Axis-specific jerk limits in the preliminary path plan are not taken into account. However, the jerk can be limited by specifying the pathMotion monitoring on the path axis accordingly. This can result in a setpoint-side path error.
  If the dynamic limits of an axis are reached, i.e. if the programmed path velocity/acceleration cannot be achieved due to these limits, an alarm will be issued.
  If the dynamic limits of the path axes are changed online, the changes take effect immediately but not for the currently active or decoded motion command.
- Segment-by-segment reduction in the maximum path dynamics according to the maximum dynamic values of path axes in these segments (**ACTIVE_WITH_VARIABLE_LIMITS**).
  This setting is equivalent to **ACTIVE_WITH_CONSTANT_LIMITS**, except that the path is segmented. Overall, the path is travelled faster; the velocity is not constant over the entire path.

From system variable **kinematicsData.transformationsOfDynamics** of the path object, you can read out whether the maximum dynamic values of the axis are transformed values. If not, the path dynamics are always limited with the path object dynamic limits, regardless of the setting in the **dynamicAdaption** command parameter.

Difference between ACTIVE_WITH_CONSTANT_LIMITS and ACTIVE_WITH_VARIABLE_LIMITS

The following trace shows the difference between **ACTIVE_WITH_CONSTANT_LIMITS** and **ACTIVE_WITH_VARIABLE_LIMITS**. Two circular paths are traveled with a 2D portal; the maximum velocities of the path axes follow:

- **Axis_X**: 500 mm/s
- **Axis_Y**: 200 mm/s

A path velocity of 400 mm/s is defined in the path commands.

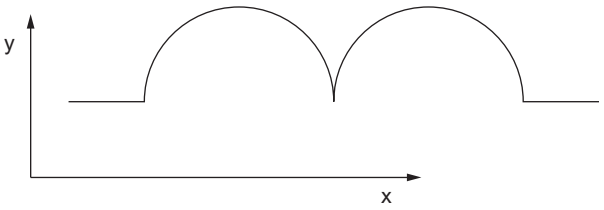


Figure 4-522 Example: Limiting the path dynamics

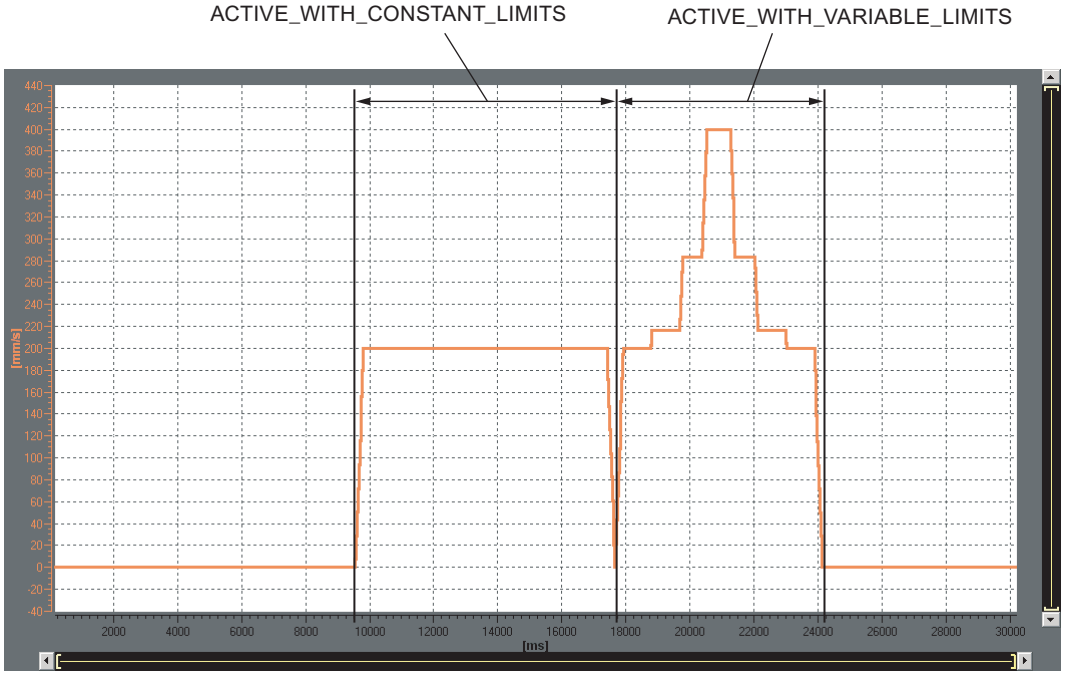


Figure 4-523 Trace: ACTIVE_WITH_CONSTANT_LIMITS, ACTIVE_WITH_VARIABLE_LIMITS

Override

A velocity override (system variable **override.velocity**) and an acceleration override (system variable **override.acceleration**) are available on the path object.

4.6.3.7 Stopping and resuming path motion

The **_stopPath()** command can be used to stop the current path motion. A stopped, but not canceled, path motion can be continued with the **_continuePath()** command.

When the path motion is resumed, the motion properties (velocity profile, acceleration, etc.) of the interrupted path command are applied. With SIMOTION V4.2 and higher, other dynamism parameters can be specified directly at the command _continuePath().

In the case of canceled path motions, if you want the application to start at the abort position, the last calculated setpoint position on the path is indicated in the **abortPosition** system variable.

Dynamic response for _stopPath()

The **_stopPath()** command can be used to define the dynamic response during deceleration. If the braking dynamic in the **_stopPath()** command is smaller than the braking dynamic in the active motion command, faults can occur in some situations.

If the dynamic response defined in the **_stopPath()** command in the previously defined path segments (i.e. in the path segment of the active command or in the path segment of the buffered commandPuffer) can be used to stop the path object, the path object will stop with error.

If the dynamic response defined in the **_stopPath()** command cannot stop the path object by the end of the previously defined path, the following can occur:

- The path interpolation is terminated.
- Each axis is delayed using the maximum dynamic response defined in the axis.
- The **50006** error message is generated.

As an example: The following path consists of three motion commands that blend in each other. This means, if the first command is active, the second command will be placed in the buffer. If the second command is active, the third command will be placed in the buffer. If the third command is active, it will be completed.

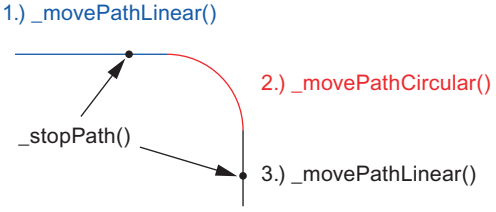


Figure 4-524 Dynamic response for _stopPath()

If **_stopPath()** with reduced dynamic response is called within the first linear path segment, the path object will be delayed using the dynamic response defined in the **_stopPath()** command. Under some circumstances, the path object stops in the circle section, it remains, however, on the path profile.

If **_stopPath()** with reduced dynamic response is called within the second linear path segment, the path object cannot stop before the end of the defined path. The axes are delayed with maximum dynamic response, the path profile may possibly by left, the **50006** error will be issued.

4.6.3.8 Path behavior at motion end

Path behavior at motion end

If the path dynamics are specified via a velocity profile, the behavior at the motion end is determined from the dynamics specified in the profile at the path end point.

If the path dynamics are specified via dynamic response parameters, the transition can be set. In addition to stopping at the end command, two sequential path segments can be dynamically linked together so that they do not have to be decelerated, see also *Axis Manual, Positioning with Blending section*.

No intermediate segments for the fillet are generated by the path interpolation for this blending.

Taking into account the axial limits, there are three transition types that can be set in the **blendingMode** parameter of the next command.

- Stopping at motion end (Page 2497) (**blendingMode:=INACTIVE**)
- Blending with dynamic adaptation (Page 2498) (**blendingMode:=ACTIVE_WITH_DYNAMIC_ADAPTION**)
- Blending without dynamic adaptation (Page 2498) (**blendingMode:=ACTIVE_WITHOUT_DYNAMIC_ADAPTION**)

The **blendingMode** parameter is only evaluated if the command is programmed with **mergeMode:=SEQUENTIAL** or **mergeMode:=NEXT_MOTION**.

The blending mode is specified in the motion command in which blending is to be performed.

Dynamic planning is executed by means of two motion commands. In SIMOTION V4.3 and higher, dynamic planning can be set by means of three motion commands (current, next and second motion command) (default setting when creating a new path object). This allows short intermediate commands to be blended without velocity reduction.

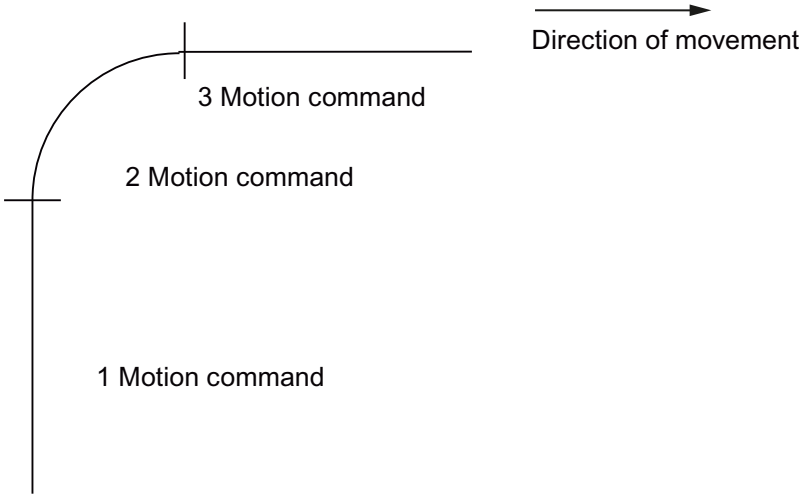


Figure 4-525 Dynamic planning via 3 motion commands

Stopping at motion end

The motion is ended in the target position of the path command. The path velocity and acceleration is zero. Any new path motion becomes active only after **END_OF_INTERPOLATION** (end of setpoint generation).

Blending with dynamic adaptation

During blending, the system supports a constant-velocity transition (with velocity profile type **TRAPEZOIDAL**) or a constant-velocity and constant-acceleration transition (with velocity profile type **SMOOTH**).

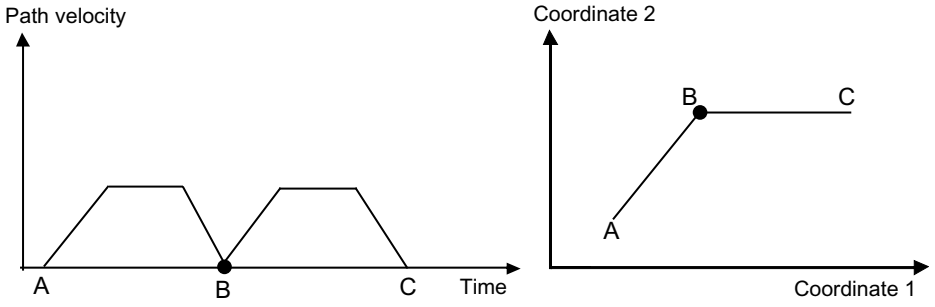


Figure 4-526 Example of blending with dynamic adaptation: Straight line - straight line

With this setting, the dynamic limits of the axis are taken into account directly when calculating the travel profile for path blending.

The axial limits for velocity and acceleration are also taken into account in the blending velocity.

For non-tangential path transitions (corners), the path velocity is reduced such that a velocity jump greater than the maximum acceleration does not occur for any of the participating axes. The result is a velocity-dependent smoothing of the path end point.

Note that with active dynamic adaptation, the dynamic axis response is set to the smaller value from axis acceleration and axis deceleration. Therefore, when an axis has a maximum acceleration of 1000 mm/s^2 and a maximum deceleration of 500 mm/s^2, the value for the deceleration is used for the calculation.

Blending without dynamic adaptation

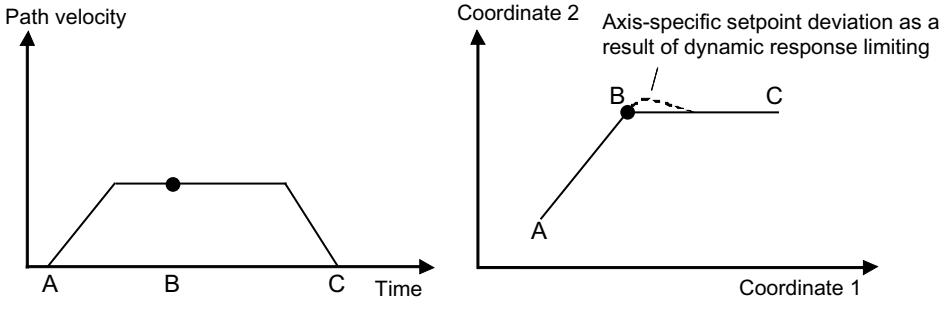


Figure 4-527 Example of blending without dynamic adaptation: Straight line - straight line

With this setting, the dynamic limits of the axis are not taken into account in path blending.

The path velocity is controlled as a scalar variable that is independent of direction and curvature.

A non-tangential attachment of path segments has no effect on the path velocity profile; for this reason, the velocity is not reduced during blending.

Because the setpoints that are generated for the individual axes are limited to the axis-specific dynamic limits for the axes, this can result in an axis setpoint error relative to the setpoint from

the path interpolation. This ultimately leads to an axis-specific deviation from the path in the blending range.

For example, this mode is applicable if the dynamic limits of the axes are to be adhered to on the path (when approaching positions, for example) but an axis-specific axis setpoint error relative to the path is acceptable at the segment transitions in the blending range.

Blending and substitution with insertion of intermediate segments

Blending with insertion of blending segments

When blending with insertion of transition segments, a blending segment is inserted between the two path segments. Either circular or polynomial segments can be inserted. As an alternative, an exact stop or blending can be carried out on the transitions without changing the path geometry.

Transition segments can even be inserted in substitute path motions.

Blending segment:

A polynomial segment can serve as a blending segment. A circular segment is also possible between two linear sets.

The command transition is the start of the blending. The path length of the blending segment and "remaining segment" is fully output on the 2nd motion command.

Blending with insertion of a polynomial segment:

Blend with insertion of a polynomial segment is possible between all path types. The transition between the segments and the polynomial path is consistent in terms of position, velocity and acceleration.

Blending with insertion of a circular segment:

Blending with insertion of a circular segment is only possible between 2 linear sets. The transition between the linear segments and the circuit is consistent in terms of position and velocity.

To determine the circular blending segment, the starting point for the blend segment (SU), the end point of blend segment (EU) and the blend radius, beginning with the common start and end point (SE) of segments 1 and 2, are required (see figure below). The distances between SE and SU, as well as SE and EU, correspond to the programmed blending clearance a. When blending, clearance is not the geometric distance between the points, but rather path lengths of segments 1 and 2 starting from SE (only relevant for non-linear segments). As in the past, the end point SE is programmed as target point of segment 1.

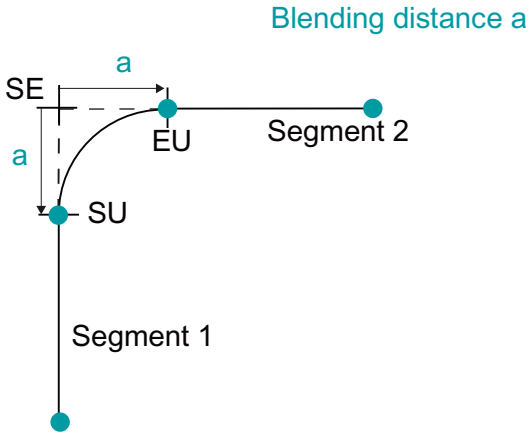


Figure 4-528 Inserting a circular blending segment between 2 linear path segments

The insertion of a circular segment is rejected by others as linear path commands and a technological alarm is issued:

"50013 blending segment not possible, reason 2: Circular blending segment can only be inserted between linear sets".

Substitution with insertion of transition segments

Behavior prior to V4.3

With **mergeMode:=IMMEDIATELY**, there is immediate replacement of the current path motion by the new path motion with the dynamic response parameters of new motion command, regardless of the setting in the **blendingMode** parameter.

Behavior as of V4.3

The system allows the insertion of transition segments even in substitute path motions. In this case, the settings for the replacement **in blendingMode** and **transitionType** are effective. With the setting **blendingMode:=INACTIVE** the substitute behavior as prior to V4.3 is effective. With the setting **blendingMode:=ACTIVE_WITHOUT_DYNAMIC_ADAPTION** or

blendingMode:=ACTIVE_WITH_DYNAMIC_ADAPTION the behavior depends on the setting in parameter **transitiontype**:

- **transitionType:=DIRECT:** (Compatibility mode, default setting)
  Substitute behavior as prior to V4.3, no transition segment, direct transition;
- **transitionType:=STOP:** Delay of the active path motion to standstill, start the new path motion; the dynamic response values of the new path motion are immediately effective, i.e., already also effective for the stop motion;
- **transitionType:=POLYNOMIAL | CIRCULAR**: A transition segment is inserted by the system, starting from the current position on the path; the blending distance to the current path point is applied.

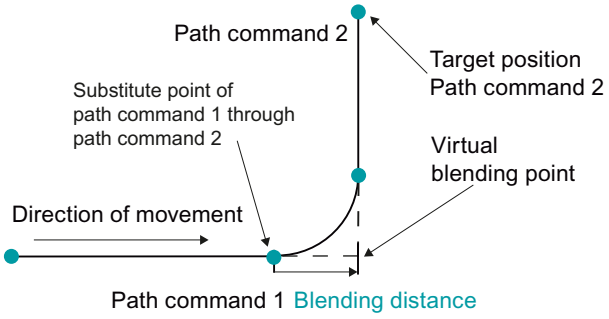


Figure 4-529 Blending when substituting commands

The virtual blending point formed is used as the end point of the current motion and as the starting point of the newly programmed motion. Blending with polynomial segment or circular segment is possible depending on the specifications.

The following applies here:

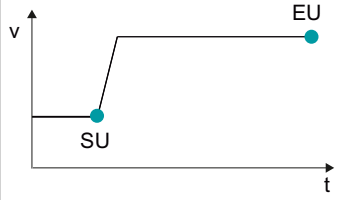
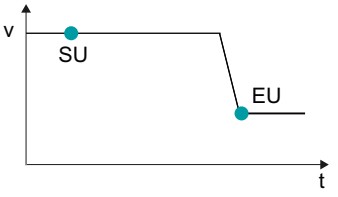
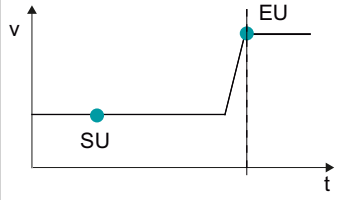
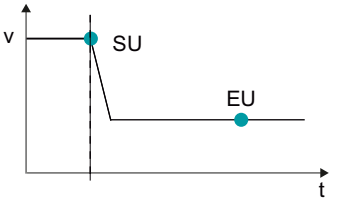
- Circular segment is possible if a linear set is blended,
- Polynomial segment is always possible

Blending dynamics

The blending velocity in the area of the blending geometry can be traversed with reference to the first or second path command. The selection is determined by setting the higher or the lower velocity of the two commands to run.

Acceleration and jerk for the velocity transition are assumed by the second path command.

Blending with mergeMode = SEQUENTIAL / NEXT_MOTION / IMMEDIATELY, transitionType:= DIRECT / STOP / POLYNOMIAL / CIRCULAR and transitionVelocityMode:= HIGH_VELOCITY / LOW_VELOCITY

HIGH_VELOCITY Running the blending geometry with the higher of the two velocities		
LOW_VELOCITY Running the blending geometry with the lower of the two velocities		

Extended behavior in substitute motions

An additional mode is available in substitute path motions. This mode first stops the substituted motion and then starts the new motion in the stop position. The result is a geometrically variable change point for the new substitute command.

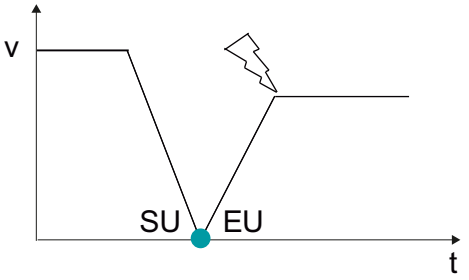


Figure 4-530 Behavior in the case of a stopped substitute motion

Overlap of blending distance

The behavior during detection of an overlap of blending distances can be configured as shown in the following diagram.

When blending several segments, overlapping blending distances may occur where necessary:

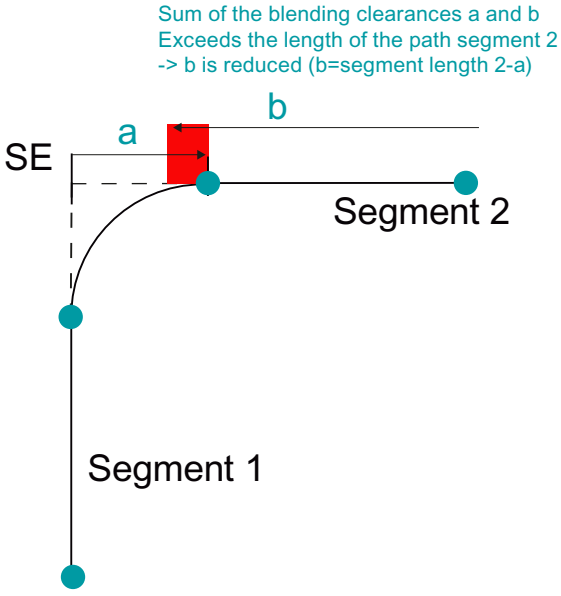


Figure 4-531 Overlapping blending distances

If an overlap of the blending distances is detected, the blending radius is reduced by the system to the maximum value (e.g. for b to MAX(s-a, s/2)), and the warning "50013 blending distance modified" is output in the alarm window. The alarm can be deactivated.

4.6.3.9 Display and monitoring options on the axis

Display and monitoring options for path motion on the axis

An active path motion is indicated on the path axis in system variable **pathMotion.state**.

Display of path-synchronous motion on the positioning axis

An active synchronous axis motion is indicated on the positioning axis in system variable **pathSyncMotion.state**.

Monitoring for setpoint error

The path axis or positioning axis can be monitored for setpoint errors (discrepancy between the setpoint specified by the path object and the setpoint output on the axis).

The difference between the setpoint and the actual value is not monitored.

Limiting and monitoring the setpoint error:

- With setting **enableCommandValue := NO_ACTIVATE**:
  - The dynamic limitation is performed without taking the jerk into account.
  - The resulting setpoint error is not monitored.
- With setting **enableCommandValue := WITHOUT_JERK**:
  - The dynamic limitation is performed without taking the jerk into account.
  - The resulting setpoint error is monitored.
- With setting **enableCommandValue := WITH_JERK**:
  - The dynamic limitation is performed taking the jerk into account.
  - The resulting setpoint error is monitored.

	Path motion on the path axis	Synchronous motion on the positioning axis
Activation of monitoring (configuration data)	**pathAxisPosTolerance. enableCommandValue**	**pathSyncAxisPosTolerance. enableCommandValue**
Tolerance value (configuration data)	**pathAxisPosTolerance. commandValueTolerance**	**pathSyncAxisPosTolerance. commandValueTolerance**
Alarm when violation occurs	**40401 Tolerance of the axis-specific path setpoints exceeded**	**40126 Tolerance of the axis-specific synchronous setpoints exceeded**
Setpoint errors exceeded (system variable)	**pathMotion. limitCommandValue**	**pathSyncMotion. limitCommandValue**
Setpoint discrepancy between path object specification and axis output value (system variable)	**pathMotion. differenceCommandValue**	**pathSyncMotion. differenceCommandValue**
Relevant path object (system variable)	**pathMotion.activePathObject**	**pathSyncMotion. activePathObject**

4.6.3.10 Allowance for axis-specific traversing range limits

The traversing range limits of the path and positioning axes, i.e. active software limit switches, are taken into account in the participating axes and not in the path object.

If a participating axis detects a possible violation of its axis-specific working area, an alarm is triggered along with an appropriate error response.

4.6.3.11 Behavior of path motion when an error occurs on a participating path axis or positioning axis

If an error occurs on a path axis or the positioning axis for path-synchronous motion causing the axis motion to stop and the command to be canceled, the path interpolation is canceled and the specified error response is performed.

See Local alarm response (Page 2613).

The other axes participating in the path motion travel to velocity 0.0 with the maximum dynamic values.

4.6.3.12 Functionality of path-synchronous motion

Functionality of path-synchronous motion

A path-synchronous motion on a positioning axis can be specified synchronous to the path motion with which it is specified. This causes the path-synchronous motion to start and end at the same time as the path motion. This enables a gripper to rotate in synchronism with the path motion, for example.

The path motion and the path-synchronous motion follow a common traversing profile. This also applies to the blending between two path segments.

The current path length can be output via w or w2. This is set with parameters in the path command.

Specification of path-synchronous motion

There are several options for path-synchronous motion, which are specified in the **wMode** parameter of the respective motion command:

- Movement to a defined end point of the path-synchronous axis w
  The target position of the path-synchronous motion is specified in the path command. This can be a relative (**RELATIVE**) or absolute (**ABSOLUTE**) position.
  As for the positioning command of the axis, the direction of the synchronous motion is specified using a parameter (**wDirection**).
  For further information, refer to the **Motion Control, TO axis, electrical / hydraulic, external encoder** Function Manual, "Positioning".
  The motion dynamics conform to the path, and the axis is "carried along". If the maximum dynamic values of the positioning axis are thereby violated, the dynamic parameters of the path are reduced accordingly.
  If the path length is zero and a path-synchronous motion is programmed, an error will be issued and the path-synchronous motion set to the programmed end position. The resulting setpoint jump is traversed axially with the maximum values.
  In this case, it is important to note that configured monitoring of the setpoint error of the path-synchronous axis also affects the setpoint jump.
- Movement according to the current path length via w or w2
  The current path distance is output. There are two ways of doing this:
  - Reference to the command (**OUTPUT_PATH_LENGTH**)
    The axis position is first set to 0.0 before the path distance is traveled.
    The reset of the axis position to zero is equivalent to a synchronized **_redefinePosition()** command.
  - Accumulated output without reset (**OUTPUT_PATH_LENGTH_ADDITIVE**)
    The path distance accumulated via the command limit is output.

Dynamics of path-synchronous motion

The path object does not keep its own dynamic response parameters for path-synchronous motion.

The following applies when calculating the path velocity profile for simultaneous traversing of a path-synchronous motion:

- Calculation of the path velocity profile without dynamic adaptation:
  - The velocity profile for the path is determined from the dynamic response parameters of the path, see Path dynamics (Page 2491).
  - The setpoints of the path interpolator for the path-synchronous motion are limited to the maximum dynamic values on the positioning axis.
  - The dynamic values (velocity, acceleration, and jerk) are adapted to the ratio of the path axis distance to the path-synchronous motion distance.

$$\frac{\text{Path (path motion)}}{\text{Path (path-synchronous motion)}} = \frac{\text{Velocity (path motion)}}{\text{Velocity (synchronous path motion)}} = \frac{\text{Acceleration (path motion)}}{\text{Acceleration (synchronous path motion)}} = \frac{\text{Jerk (path motion)}}{\text{Jerk (path-synchronous motion)}}$$

Use of this formula assumes that the unit settings for the path object and the participating axes are the same.

- Calculation of the path velocity profile with dynamic adaptation:
  The dynamics of the path-synchronous motion are incorporated into the path plan the same as an additional orthogonal coordinate, and, if necessary, the path velocity profile is adapted in such a way that the dynamic limits of the positioning axis are not violated by the path velocity profile.

Path blending with a path-synchronous motion

- Path blending with dynamic adaptation
  The dynamics of the path-synchronous motion are incorporated into the motion plan the same as an additional orthogonal coordinate, and, if necessary, the velocity profile in the blending range is adapted accordingly.
- Path blending without dynamic adaptation
  If the quotient of the distance length (path motion) / distance length (path-synchronous motion) is not equal over the individual path segments, the path segment transitions will be discontinuous with regard to the velocity setpoints of the path object for the path-synchronous motion.

The setpoints resulting from the path interpolation for the path-synchronous motion are limited on the positioning axis using the axis-specific dynamic limits of that axis.

For example, if the path object is limited over the path using just the dynamic limits available on the path object, this can result in a setpoint error on the positioning axis relative to the calculated setpoint on the path object for the path-synchronous motion. See Display and monitoring options on the axis (Page 2503).

Output of the path distance to the positioning axis

Alternatively, the traveled path distance, i.e. the current path length, can be output to the positioning axis. This distance can be relative to an individual path segment or added up over multiple path segments.

The setting is made in the path command.

For example, this can be used to output path distance-related output cams or measuring inputs.

Output of Cartesian coordinates using the MotionOut Interface

The **motionOut.x/y/z** interfaces can be used to interconnect the Cartesian coordinates directly with other technology objects, e.g. with the MotionIn interfaces of positioning axes.

For example, this functionality can be used in the application to implement output cams and measuring inputs on Cartesian axes.

4.6.3.13 Kinematic adaptation

Kinematic adaptation

The kinematic transformation or the kinematic adaptation is used to convert path axis values to the Cartesian axes, and vice versa.

Kinematic adaptation – fundamentals

Scope of the transformation functionality

During forward calculation of the kinematics (including direct kinematics, forward kinematics or forward transformation) for position and motion conversion, the position of the end point of the kinematics is determined in the basic coordinate system from the position of the articulation angle and its spatial arrangement.

During backward calculation (including backward transformation or inverse kinematics), the position of the individual articulation angle is determined from the position of the end point of the kinematics in the basic coordinate system. For path interpolation, the position of the end point of the kinematics in the basic coordinate system is calculated over time.

The position and the dynamic values are transformed.

If the axis settings are outside the transformation range, the values in the BCS and OCS will be set to zero.

The current modulo range is retained in path axes specified as modulo axes.

See Modulo properties (Page 2477).

Reference points

The following reference points are used in path interpolation:

- Cartesian zero point
- Kinematic zero point
- Kinematic end point
  (because a tool is not taken into account, this is equal to the path point)

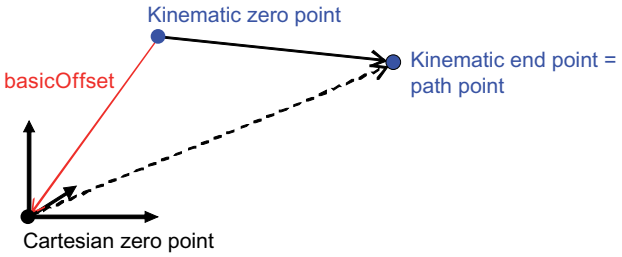


Figure 4-532 Reference points of the coordinate systems in path interpolation

The path object calculates the position on the path. This is also the kinematic end point.

System variables for path interpolation and transformation on the path object

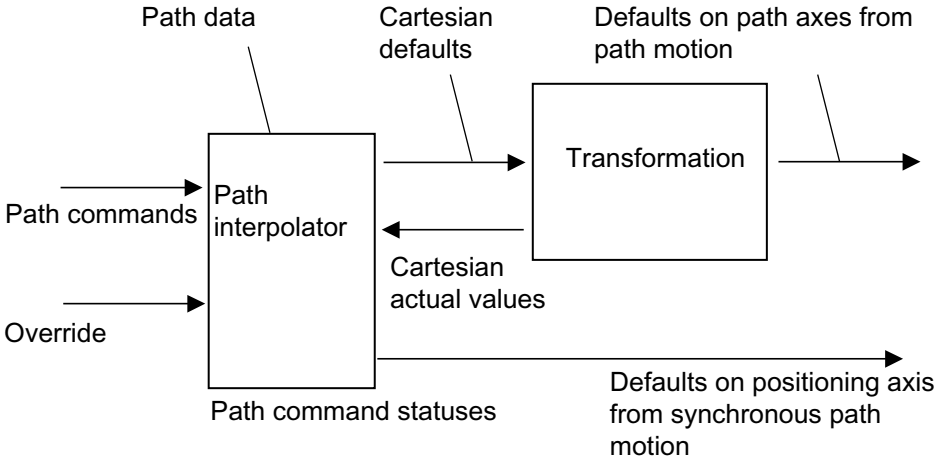


Figure 4-533 Overview of system variables of the path object

The position values and dynamic values can be accessed via a system variable:

Path data

System variables	Description
path.acceleration	Path acceleration
path.command	Status of a motion command
path.dynamicAdaption	Display showing whether adaptation of the path dynamics to the dynamic limit values of the path axes is active.
path.length	Length of the current path
path.motionState	Motion status of path motion

System variables	Description
path.position	Path position (within the path length)
path.velocity	Path velocity
path.csType	Type of coordinate system
path.csNumber	Number of the coordinate system (in OCS)

Cartesian specifications in the basic coordinate system / path-synchronous motion

System variables	Description
bcs.x/y/z/w.position	Set positions
bcs.x/y/z/w.velocity	Set velocities
bcs.x/y/z/w.acceleration	Set accelerations
bcs.linkConstellation	Set link constellation

Cartesian actual values

System variables	Description
bcs.x/y/zActual.position	Actual value of the Cartesian positions of the path axes
bcs.linkConstellationActual	Current articulated joint positioning space

Defaults on path axes from path motion

System variables	Description
mcs.a1/a2/a3.acceleration	Accelerations of the path axes
mcs.a1/a2/a3.position	Positions of path axes in the axis coordinates
mcs.a1/a2/a3.velocity	Velocities of the path axes

Object coordinate system

System variables	Description
ocs[1..3].trackingIn	Interface for motion sequence reference value with which the OCS is to be coupled (e.g. TO external encoder)
ocs[1..3].trackingInPosition	Current value of the motion sequence
ocs[1..3].trackingPosition	Position of the OCS relative to the reference position
ocs[1..3].trackingState	Synchronization status
ocs[1..3].x/y/z.accelaration	Acceleration
ocs[1..3].x/y/z.position	Item
ocs[1..3].x/y/z.velocity	Velocity

Override

System variables	Description
override.acceleration	Acceleration override
override.velocity	Velocity override

Path command statuses

System variables	Description
linearPathCommand.state	Status of linear interpolation
circularPathCommand.state	Status of circular interpolation
polynomialPathCommand.state	Status of polynomial interpolation

Velocity profile

System variables	Description
specificVelocityProfile.state	Information as to whether a velocity profile is in use
specificVelocityProfile.value	Profile value
specificVelocityProfile.activeProfile	Active profile reference
specificVelocityProfile.processingState	Status of the profile processing

Command queue

System variables	Description
motionBuffer.numberOfExistentEntries	Number of commands in the command buffer
motionBuffer.state	Status of the command buffer

Interconnections on the path object

System variables	Description
connection.a1	1. Path axis on the path object
connection.a2	2. Path axis on the path object
connection.a3	3. Path axis on the path object
connection.w	Path-synchronous axis on the path object

Transformation of the dynamic values

The **kinematicsData.transformationOfDynamics** system variable indicates whether a kinematic transformation supports the dynamics transformation functionality.

Differentiation of link constellations

If Cartesian kinematics end points can be reached via various articulation positions, articulation positioning spaces are defined for the corresponding kinematics.

All path motions take place in the same link constellation. For this reason, a change to another link constellation is not possible when a path is being executed. A change to another link constellation is possible through individual axis motions but not via a motion on the path object.

The current transformation-specific link constellation is indicated on the setpoint side in the **bcs.linkConstellation** variable and on the actual value side in the **bcs.linkConstellationActual** variable.

The link constellation is defined specifically for each transformation. See Supported kinematics (Page 2513).

Information commands for the kinematic transformation

In addition to the implicit conversion in the system, the transformation calculations can also be accessed directly via user commands.

- The **_getPathCartesianPosition()** command is used to calculate the Cartesian positions for the axis positions specified in the command.
- The **_getPathAxesPosition()** command is used to calculate the axis positions from the Cartesian positions.
- The **_getPathCartesianData()** command is used to calculate the Cartesian data for the position, velocity, and acceleration from the axis positions, axis velocities, and axis accelerations specified in the command.
- The **_getPathAxesData()** command is used to calculate the axis positions, axis velocities, and axis accelerations from the Cartesian data for the position, velocity, and acceleration specified in the command.

For the calculation of axis positions, the values are specified in the axis coordinate of the path axis, and not relative to the kinematic zero point of the axis.

The modulo range is taken into account.

For the transformation of Cartesian values to path axis values, a link constellation and not a reference position of the axes has to be specified in order to ensure uniqueness.

Axis-specific zero point offset in the transformation

It is possible to set an axis-specific offset of the zero position of the axis in the axis-specific coordinate system as well as the zero definition of the axis in the transformation.

The positive direction of the axis and of the axis in the transformation must be the same. These settings are made for the axis.

The offset of the kinematic zero point relative to the axis zero point is specified in the positive direction of the axis.

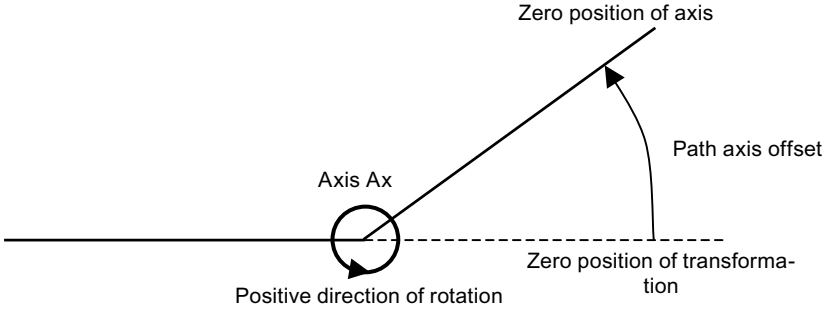


Figure 4-534 Path axis offset

When modulo axes are used for rotary links with a limited domain in kinematics, such as SCARA, the axis-specific zero point offset and the modulo property of the relevant path axis are defined such that the permissible modulo range of the path axis coincides with the domain of the relevant arm within the kinematics. Otherwise, this can cause an additional limitation in the traversing range of the kinematics.

Example: If a link is limited to [-180°; 180°) and a modulo range of 0° to 360° is defined on the path axis, the zero point offset to -180° should be specified.

Offset of the kinematic zero point relative to the Cartesian zero point

An offset of the kinematic zero point of the transformation relative to the Cartesian zero point can be set in the **basicOffset** configuration data.

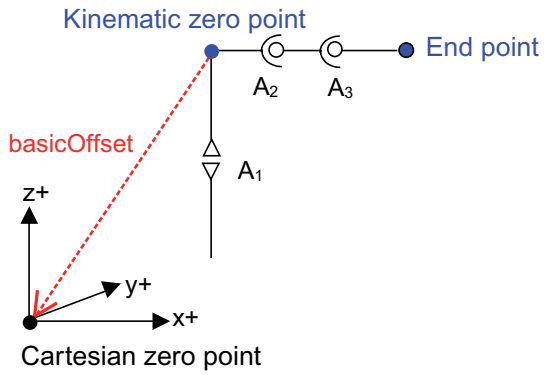


Figure 4-535 Example of kinematic offset

The above example produces negative values for the kinematic offsets.

Offset in example:
x: -100
y: -100
z: -200

With SIMOTION V4.2 and higher, not only can the BCS be offset but also rotated, allowing for any rotation of the coordinate system from the kinematics zero point. This allows flexible assignment of the BCS to the handling equipment's kinematics.

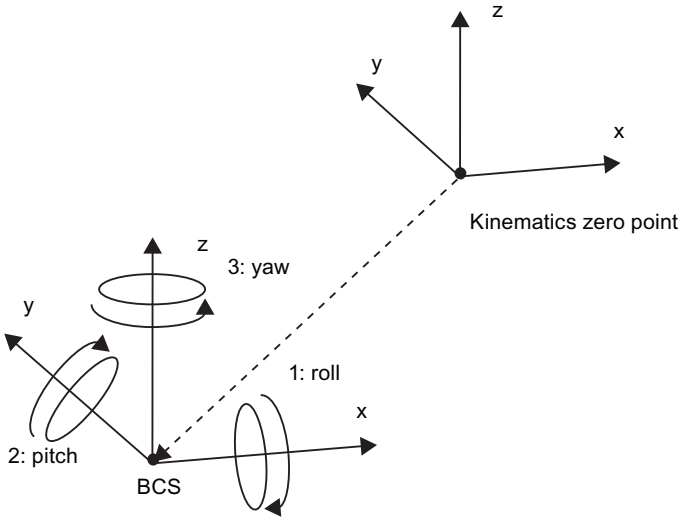


Figure 4-536 Coordinate system offset and rotation

The rotations are undertaken after the offset in the following order:

1. Roll around x axis
2. Pitch around (already rotated) y axis
3. Yaw around (already twice rotated) z axis

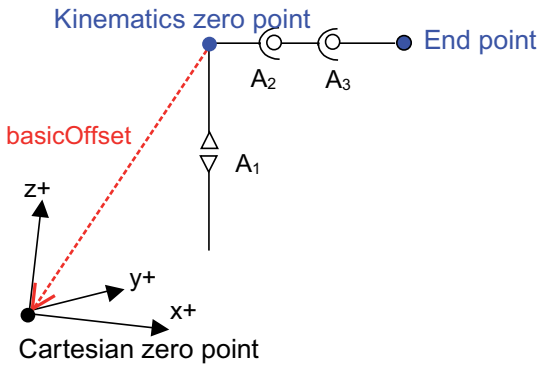


Figure 4-537 Example of kinematics offset with rotation

Offset and rotation (around the y axis) in the example:

x: -100
y: -100
z: -200
roll: 0°
pitch: +15°
jaw: 0°

Supported kinematics

Supported kinematics and their assignment

The following kinematics can be set using the **typeOfKinematics** configuration data:

- Cartesian 2D/3D gantries (Page 2517) (**CARTESIAN**):
- 2D roller picker (Page 2518) (**ROLL_PICKER**)
- 3D roller picker (Page 2520) (**ROLL_PICKER_3D**) available as of V4.4.
- 2D delta picker (Page 2521) (**DELTA_2D_PICKER**)
- 3D delta picker (Page 2523) (**DELTA_3D_PICKER**)
- SCARA kinematics (Page 2526) (**SCARA**)
- 2D articulated arm kinematics (Page 2530) (**ARTICULATED_ARM_2D**) available as of V4.2.
- 3D articulated arm kinematics (Page 2531) (**ARTICULATED_ARM**)
- 2D swivel arm kinematics (Page 2534) (**SWIVEL_ARM**) available as of V4.2.
- 3D cylindrical robot (Page 2536) (**CYLINDRICAL**) available as of V4.4.
- 2D/3D user function (Page 2538) (**USER_FUNCTION**) available as of V4.4.
- Other special kinematics (**SPECIFIC**)

A transformation can be selected for each path object.

Thus, multiple transformations can be configured/active in a SIMOTION system when multiple path objects are used.

Because a path axis can be interconnected with more than one path object, a path axis can theoretically be involved in multiple kinematic assemblies but obviously can only be active in one path group at a time.

See also

Specific kinematics (Page 2544)

Configuration screens

With SIMOTION version 4.2 and higher you can use parameterization screens to configure the kinematics. You can access the screens via the configuration menu.

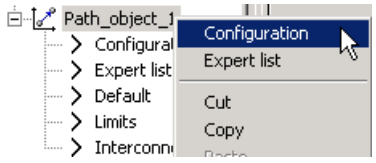


Figure 4-538 Open configuration

Depending on the kinematics, the screen will contain several tabs where you can enter mechanical data and offsets and rotations.

Example: Cartesian kinematics 2D

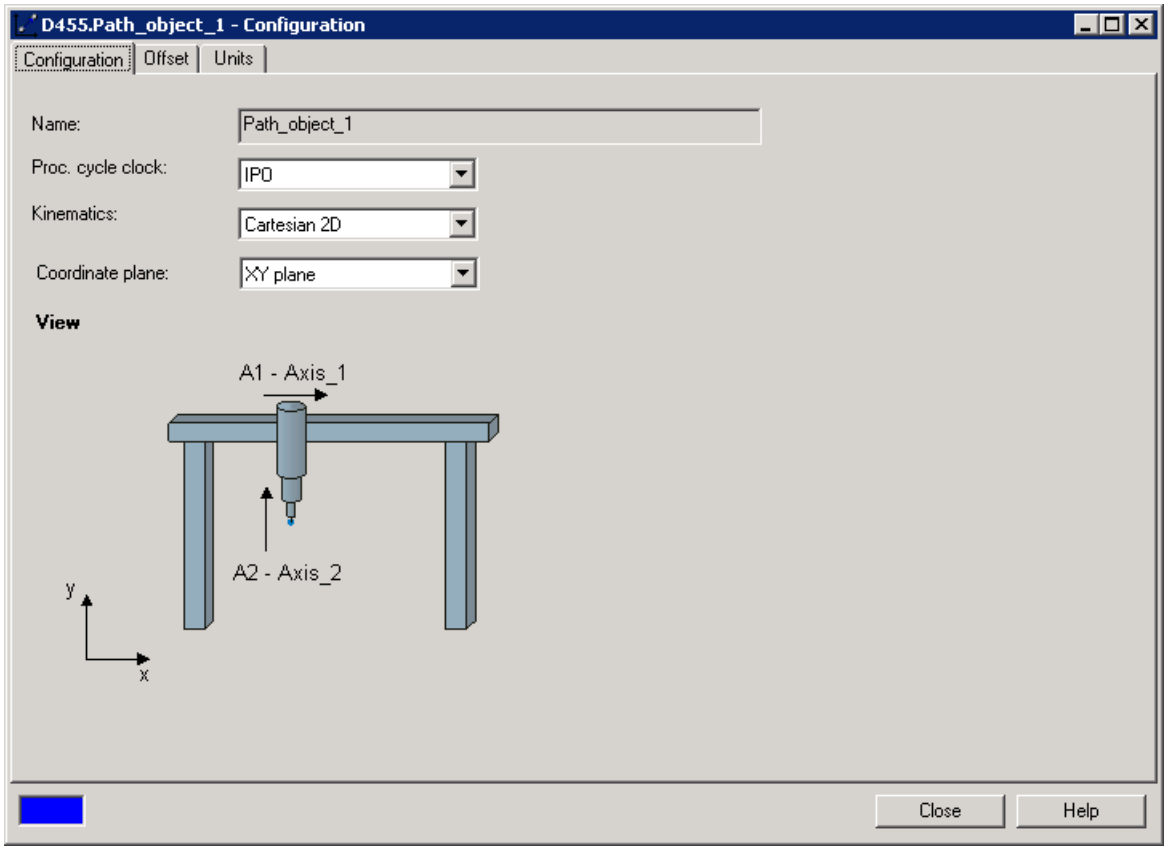


Figure 4-539 Cartesian kinematics 2D - configuration

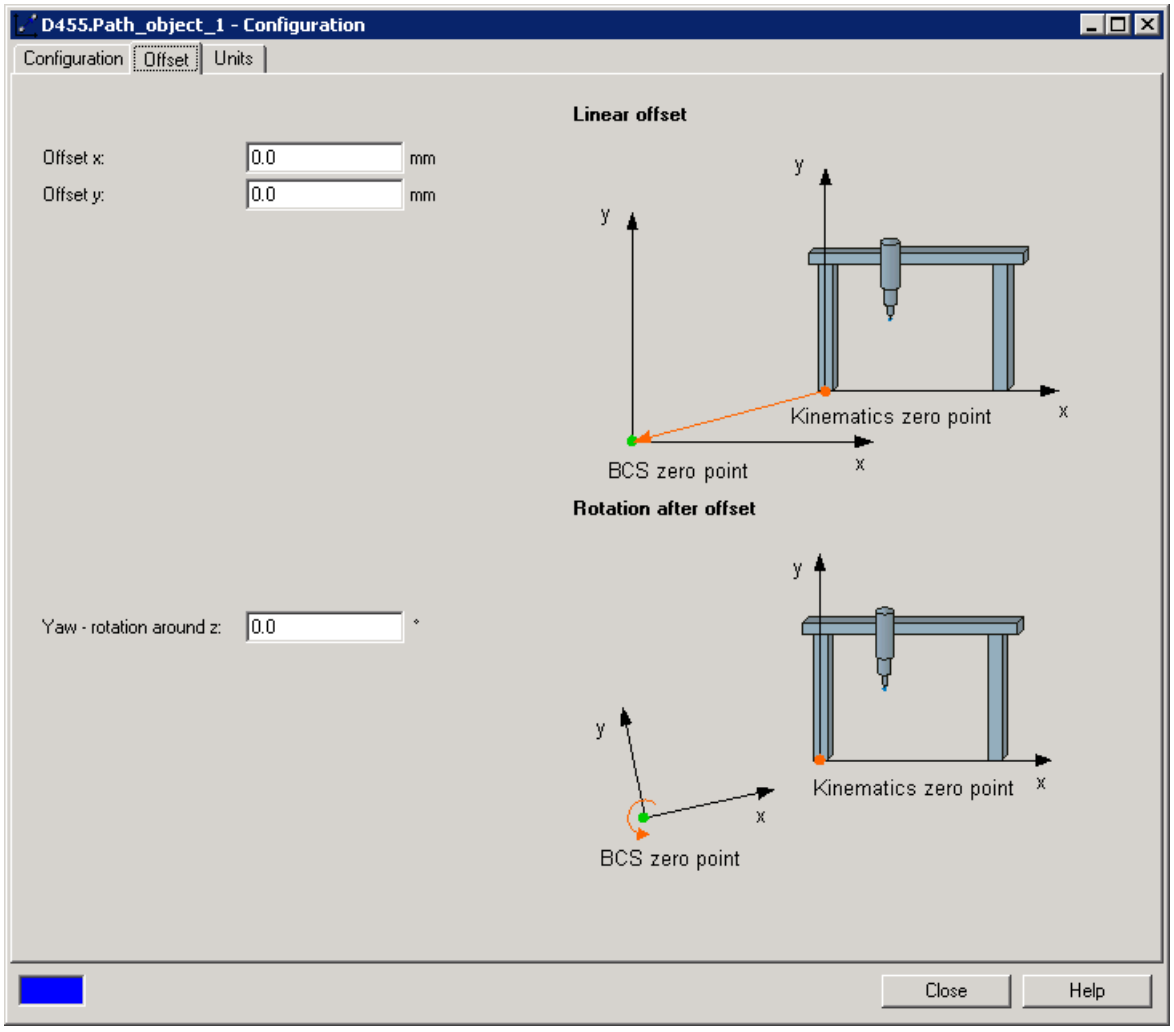


Figure 4-540 Cartesian kinematics 2D - offset

Cartesian 2D/3D gantries

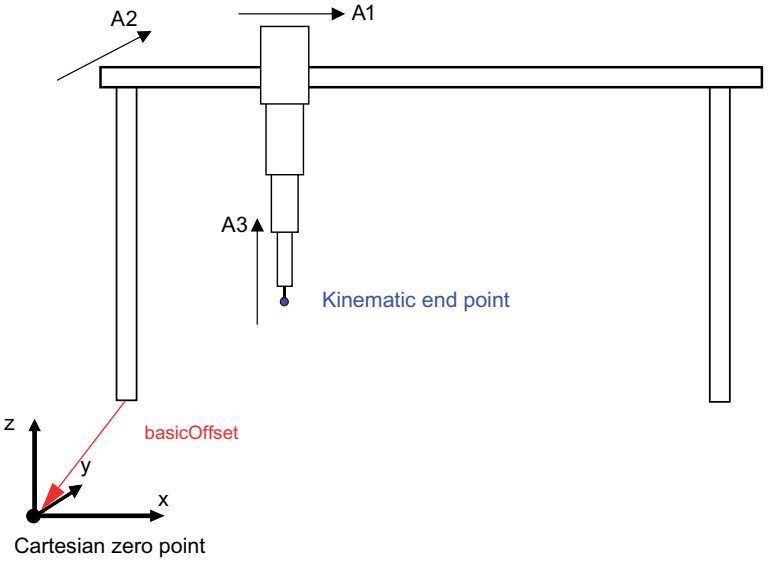


Figure 4-541 Kinematics example: 2D/3D gantry

Table 4-253 Configuration data for the Cartesian kinematics

typeOfKinematics: CARTESIAN	Cartesian gantry kinematic type
basicOffset.x	Offset of the zero point of Cartesian coordinate **x** relative to the zero point of axis coordinate A1
basicOffset.y	Offset of the zero point of Cartesian coordinate **y** relative to the zero point of axis coordinate A2
basicOffset.z	Offset of the zero point of Cartesian coordinate **z** relative to the zero point of axis coordinate A3
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
cartesianKinematicsType	Select 2D or 3D (determines the number of axes involved)
config2D	Main plane (only for 2D gantry)

Table 4-254 Possible articulated joint positioning spaces

linkConstellation	Irrelevant (always 1)

2D roller picker

General

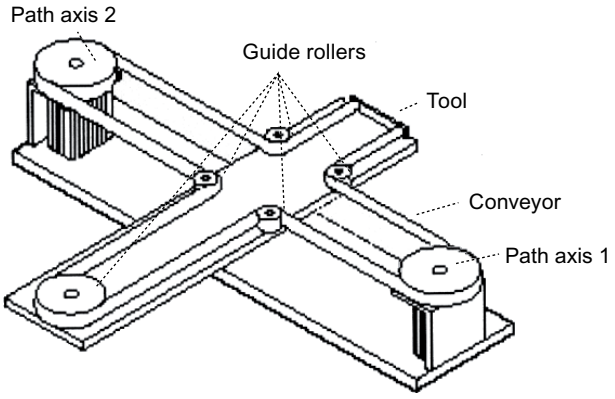


Figure 4-542 2D roller picker: Representation of the axis system

The 2D roller picker has two-dimensional kinematics. You can configure 2D roller pickers in all three main planes. This description is based on a configuration in the x-y plane.

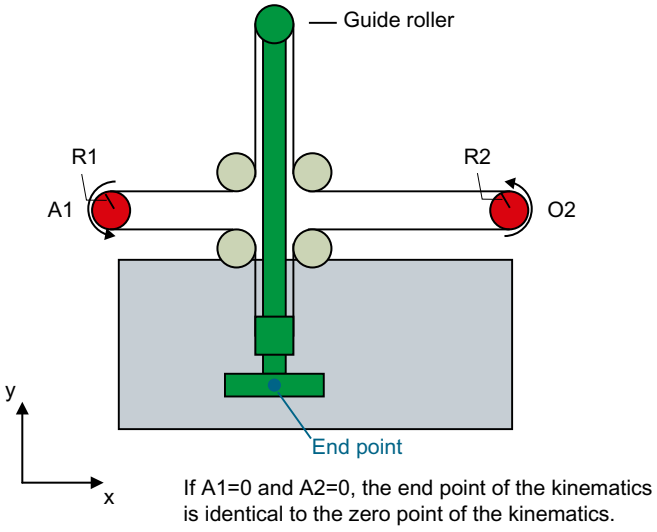


Figure 4-543 Kinematics of the 2D roller picker (guide pulley on the opposite side of the tool)

The guide pulley must be located on the opposite side of the tool.

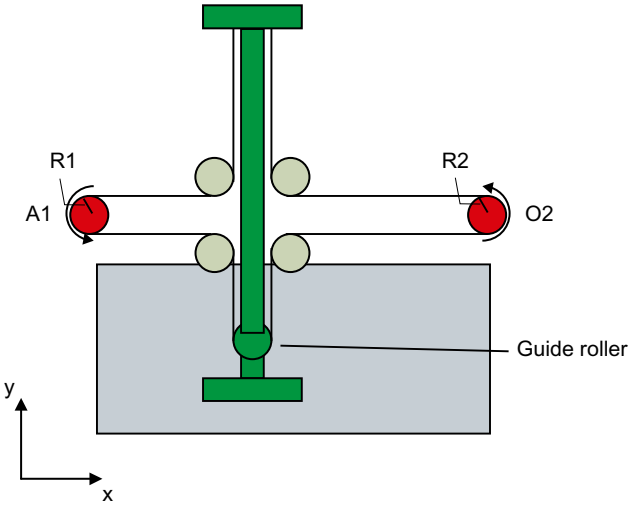


Figure 4-544 Kinematics of the 2D roller picker (guide pulley on the tool; this case is not considered)

The alternative variant with the guide pulley on the tool can be derived by converting the coordinates:

Guide pulley on the tool	Guide pulley on the opposite side of the tool
x	-x
y	-y
R1	R2
R2	R1

Note

The two path axes must be configured so that 360 axis-units (i.e. mm, degree, etc.) produce a disk revolution. The "modulo axis" setting should be prevented. See Units (Page 2477) and Modulo properties (Page 2477).

Table 4-255 Configuration data for the 2D roller picker kinematics

typeOfKinematics: ROLL_PICKER	2D roller picker kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
Axis 3 is not available for the 2D roller picker.	
config2D	Main plane of the path axes

Table 4-256 Specification of the radius of the disks on the motors in:

radius1	Disk radius for path axis 1
radius2	Disk radius for path axis 2

Table 4-257 Possible articulated joint positioning spaces

LinkConstellation	Irrelevant (always 1)

3D roller picker

General

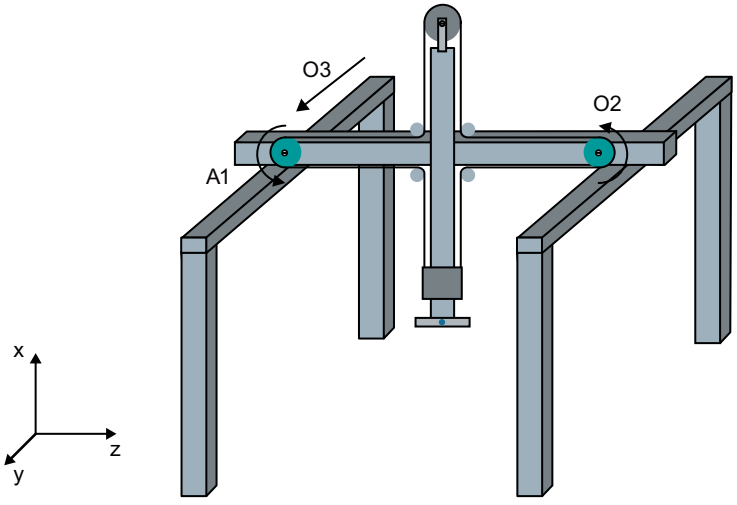


Figure 4-545 Kinematics of the 3D roll picker for a roll picker in the z-x plane

The 3D roller picker is a 2D roller picker that can be moved via an additional linear axis in a further Cartesian coordinate direction. You can set the 3D roller picker in all three main planes as with the 2D roller pickers. The third path axis then extends in the third Cartesian direction. In the example, the roll picker is set in the z-x plane.

A1 and A2 are the two axes of the roll picker in the example figure. The linear axis A3 is indicated by the arrow in the y direction.

Further information can be found in the documentation of the 2D roller picker (Page 2518).

Table 4-258 Configuration data for the 3D roller picker kinematics

typeOfKinematics: **ROLL_PICKER_3D**	3D roller picker kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate

basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
config2D	Main plane of the path axes
Specification of the radius of the disks on the motors in:	
radius1	Disk radius for path axis 1
radius2	Disk radius for path axis 2

Table 4-259 Possible articulated joint positioning spaces

LinkConstellation	Irrelevant (always 1)

2D delta picker

General

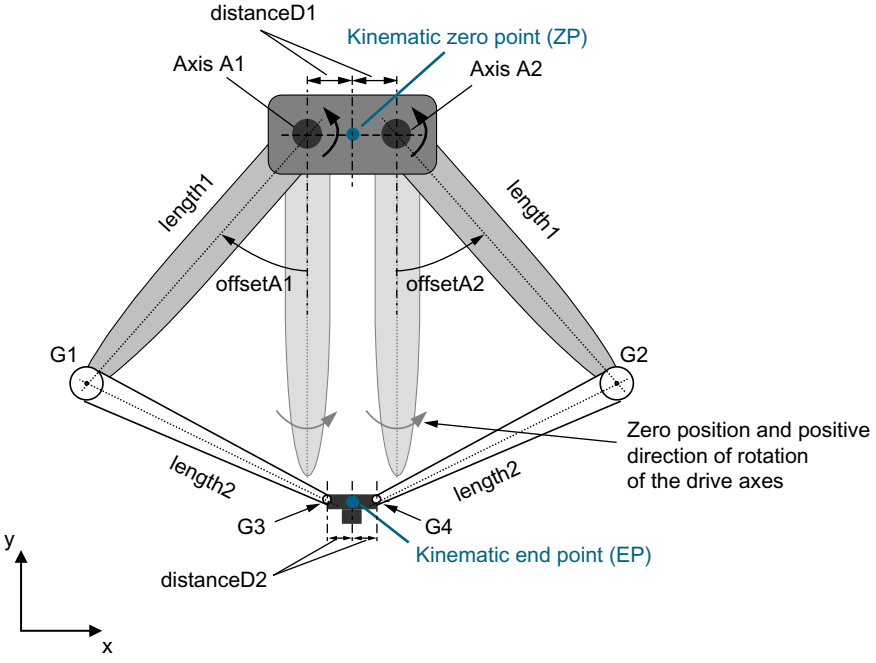


Figure 4-546 Kinematics of the 2D delta picker (x-y plane example)

Definitions

- The complete structure is contained in one of the two-dimensional main planes. The x-y plane is used as an example in the following description.
- A1 and A2 denote the two active drive axes of the kinematic structure. They lie on the straight line y = 0 and are separated from each other by the distance 2x **distanceD1**. Their zero position within the kinematic structure corresponds to the orientation of the upper arm segments (**length1**) in the direction of the negative y axis. Positive displacements occur as shown in the figure.

- It is assumed that the lower connection plate between G3 and G4 is always oriented horizontally.
  This results in $y_{G3} = y_{G4}$ and a horizontal distance of 2x **distanceD2**.
- If $x_{ZP} = y_{ZP} = 0$, the zero position of the kinematics is located midway between the drive axes A1 and A2.
- The target point of the direct transformation, with its coordinates x_{EP} and y_{EP} is defined as being midway between G3 and G4. This results in the position G3 = (x_{EP}-distanceD2; y_{EP}) and G4 = (x_{EP}+distanceD2; y_{EP}).

Table 4-260 Configuration data for the 2D delta picker kinematics

typeOfKinematics: DELTA_2D_PICKER	2D delta picker kinematics type
basicOffset	Offset of the kinematic zero point (ZP) relative to the Cartesian zero point
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
Axis 3 is not available for the 2D delta picker.	
config2D	Main plane of the path axes
length1	Length of the upper arm segment
length2	Length of the lower arm segment
distanceD1	Distance of the drive axes A1 and A2 from the kinematic zero point (ZP)
distanceD2	Distance of the joints G3 and G4 from the end point (EP)
offsetA1	Offset of the A1 drive axis
offsetA2	Offset of the A2 drive axis

Table 4-261 Possible articulated joint positioning spaces

The following angles are always in relation to the direct orientation φ_0 of the active axis to the end point.		
LinkConstellation	1	Angle of axis A1 in the range [-180°, 0°]. Angle of axis A2 in the range [0°, 180°].
	2	Angle of axis A1 in the range (0°, 180°). Angle of axis A2 in the range [0°, 180°].
	3	Angle of axis A1 in the range (0°, 180°). Angle of axis A2 in the range (-180°, 0°).
	4	Angle of axis A1 in the range [-180°, 0°]. Angle of axis A2 in the range (-180°, 0°).

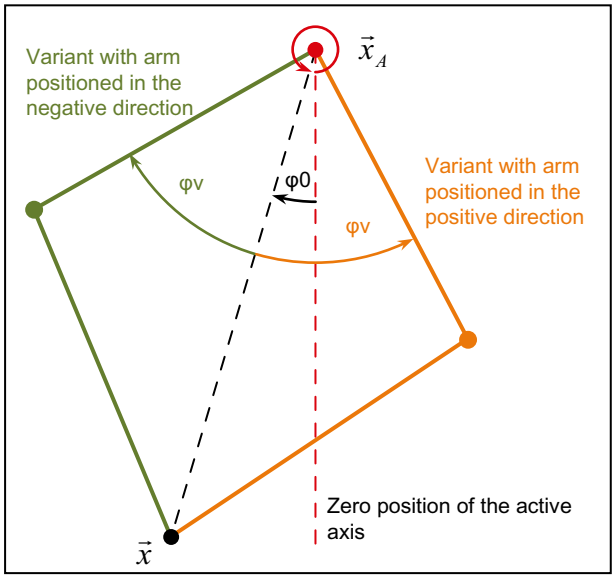


Figure 4-547 Illustration of the articulated joint positioning as exemplified by a single axis

3D delta picker

General

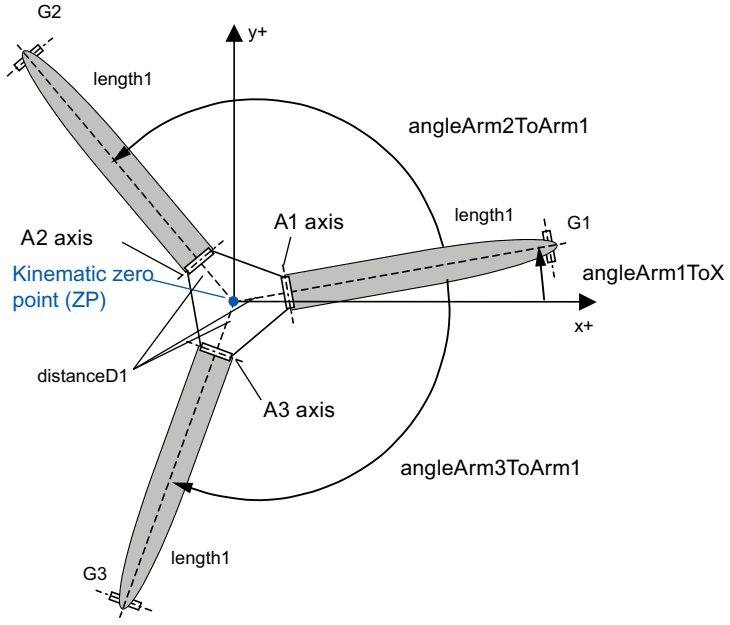


Figure 4-548 Kinematics of the 3D delta picker (top view)

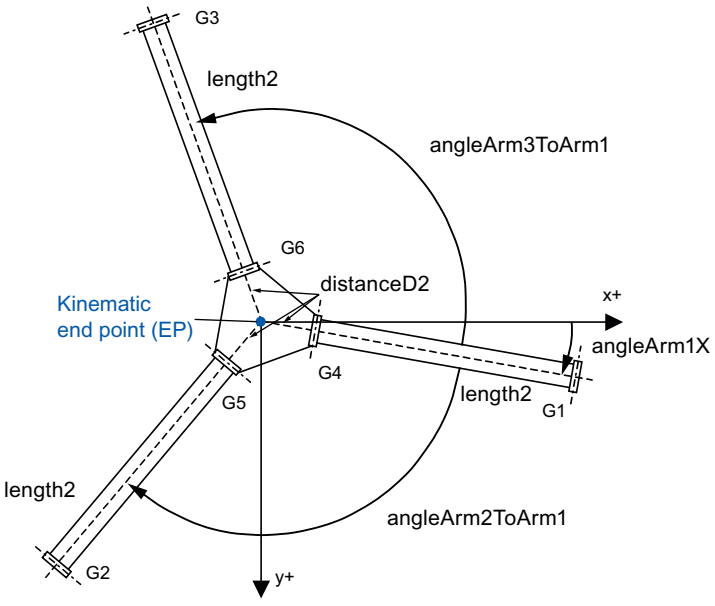


Figure 4-549 Kinematics of the 3D delta picker (bottom view)

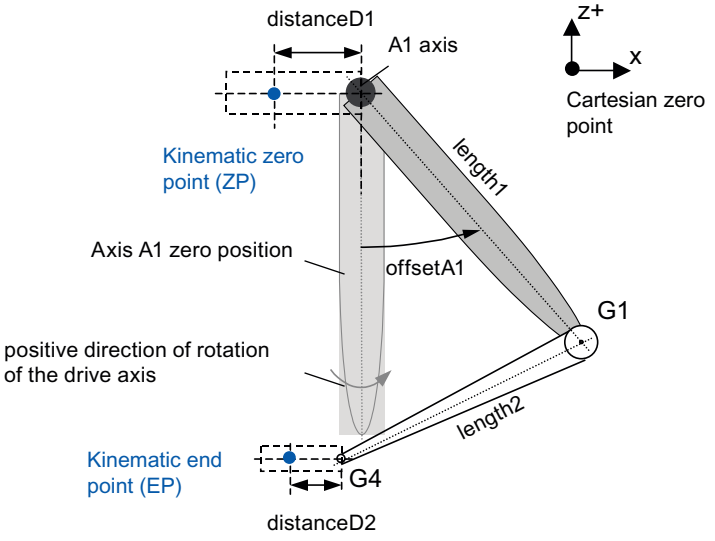


Figure 4-550 Kinematics of the 3D delta picker (single arm in the example, axis A1)

Definitions

- A1, A2 and A3 denote the three active drive axes of the kinematic structure. They are in the x-y plane where z = 0, and each is at distance **distanceD1** from the kinematic zero point (ZP). Their zero position within the kinematic structure corresponds to direct orientation of the upper arm segments (**length1**) in the direction of the negative Z axis. Positive displacements occur counterclockwise, as shown in the previous figure.
- G1 to G6 denote the freely movable joints.
- It is assumed that the connection of the joints at the end point (EP) has a horizontal orientation based on the parallel struts. This results in $y_{G4} = y_{G5} = y_{G6}$. G4 to G6 are all at the horizontal distance **distanceD2** from the end point (EP).

- If $x_{ZP} = y_{ZP} = z_{ZP} = 0$, the zero position of the kinematics is located midway between the drive axes A1 to A3.
- The target point of the transformation, with its coordinates x_{EP}, y_{EP} and z_{EP} is defined as being midway between G4 and G6.

Table 4-262 Configuration data for the 3D delta picker kinematics

typeOfKinematics: DELTA_3D_PICKER	3D delta picker kinematics type
basicOffset	Offset of the kinematic zero point (ZP) relative to the Cartesian zero point
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
length1	Length of the upper arm segment
length2	Length of the lower arm segment
distanceD1	Distance of the drive axes A1 to A3 from the kinematic zero point (ZP)
distanceD2	Distance of the articulations G4 to G6 from the end point (EP)
offsetA1	Offset of the A1 drive axis
offsetA2	Offset of the A2 drive axis
offsetA3	Offset of the A3 drive axis
angleArm1ToX	Angular offset of arm A1-G1-G4 with respect to the x axis for rotation around the positive z axis.
angleArm2ToArm1	Angular offset of arm A2-G2-G5 with respect to arm A1-G1-G4 for rotation around the positive z axis.
angleArm3ToArm1	Angular offset of arm A3-G3-G6 with respect to arm A1-G1-G4 for rotation around the positive z axis.

Table 4-263 Possible articulated joint positioning spaces

The following angles are always in relation to the direct orientation φ_0 of the active axis to the end point.		
LinkConstellation	1	Angle of axis A1 in the range [0°, 180°]. Angle of axis A2 in the range [0°, 180°]. Angle of axis A3 in the range [0°, 180°].
	2	Angle of axis A1 in the range (-180°, 0°). Angle of axis A2 in the range [0°, 180°]. Angle of axis A3 in the range [0°, 180°].

	3	Angle of axis A1 in the range [0°, 180°]. Angle of axis A2 in the range (-180°, 0°). Angle of axis A3 in the range [0°, 180°].
	4	Angle of axis A1 in the range [0°, 180°]. Angle of axis A2 in the range [0°, 180°]. Angle of axis A3 in the range (-180°, 0°).
	5	Angle of axis A1 in the range (-180°, 0°). Angle of axis A2 in the range (-180°, 0°). Angle of axis A3 in the range [0°, 180°].
	6	Angle of axis A1 in the range (-180°, 0°). Angle of axis A2 in the range [0°, 180°]. Angle of axis A3 in the range (-180°, 0°).
	7	Angle of axis A1 in the range [0°, 180°]. Angle of axis A2 in the range (-180°, 0°). Angle of axis A3 in the range (-180°, 0°).
	8	Angle of axis A1 in the range (-180°, 0°). Angle of axis A2 in the range (-180°, 0°). Angle of axis A3 in the range (-180°, 0°).

You will find an illustration of the articulated joint positioning options for a single axis in 2D delta picker (Page 2521).

SCARA kinematics

General

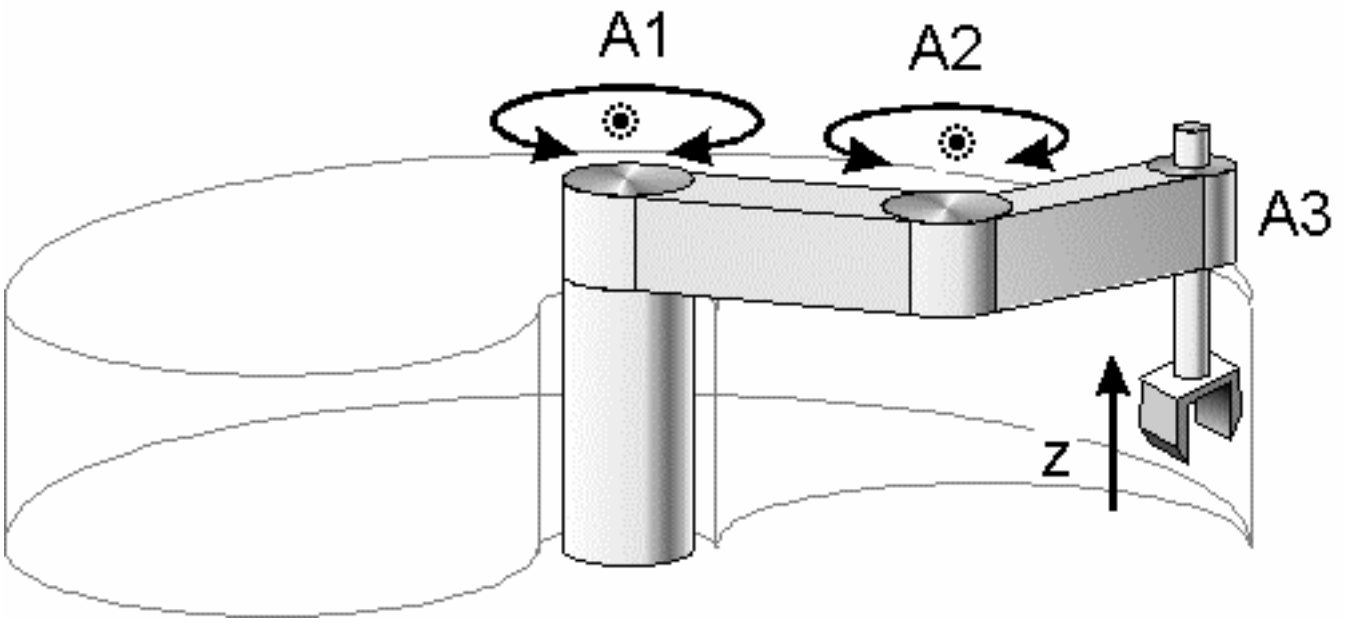


Figure 4-551 SCARA: Representation of the axis system

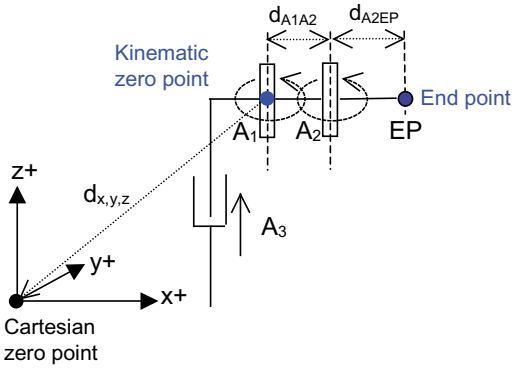


Figure 4-552 SCARA: Kinematics

The kinematic zero point is at point A1.

The zero positions of the A1 axis and A2 axis are as follows:

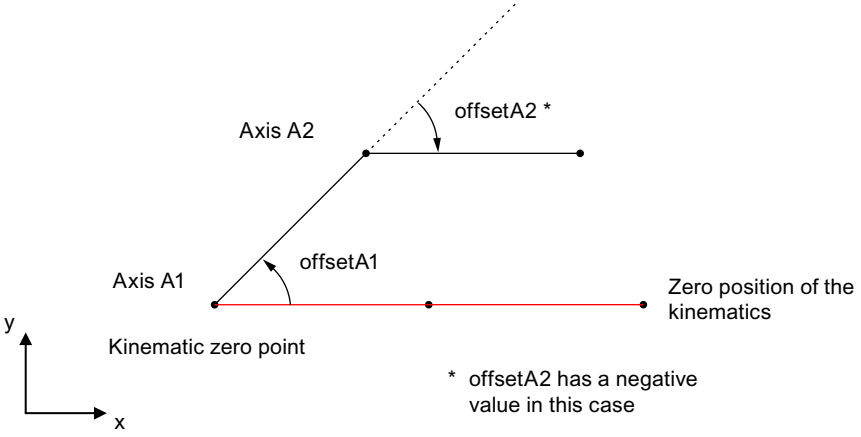


Figure 4-553 SCARA: Zero positions

The definition range of the single A1 and A2 axes is limited to [-180°; 180°).

Link compensations

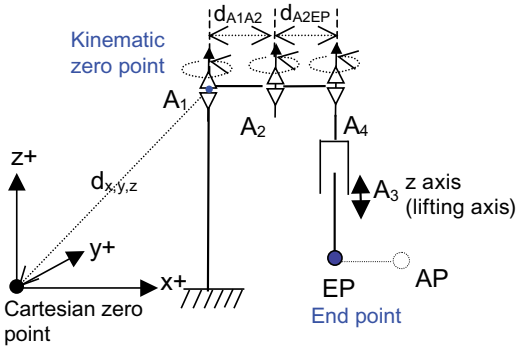


Figure 4-554 SCARA: Kinematics with link compensation

Coupled axes

Mechanical couplings are possible:

- between A1 and A2
- between A1, A2, and $A_{synchronous}$ (A4)
- between $A_{synchronous}$ (A4) and A3

The following axis couplings can be compensated via the system:

- A coupling from axis A1 to axis A2
- A coupling from axis A1 and axis A2 to the path-synchronous controlled axis A4
  That is, the setpoint of axis A4 is changed to the position axis in accordance with the changes of A1 and A2.
  If axis A1 and/or axis A2 is traversed and a path-synchronous motion in w is specified in parallel, the system superimposes/adds a path-synchronous motion specification and compensation onto the position axis.
- A coupling from axis A4 to axis A3 (lifting axis)
  If axis A3 is traversed via the path motion and a compensation from w to axis A3 is required simultaneously, the specifications are superimposed.

The compensation functionality and the specifications to the path-synchronous axis w via the path-synchronous motion are independent of one another and are executed simultaneously by the system.

Effective direction of the coupled axes

For the coupling of axis A1 to axis A2 and of axis A4 to axis A3, the following applies:

For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a negative motion on the second axis.

For the coupling of axis A1 to axis A4 and of axis A2 to axis A4, the following applies:

For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a positive motion on the second axis.

The coupling between axis A4 and axis A3 is implemented as a spindle pitch, i.e. for a coupling factor of +1.0, 360.0 degrees on axis A4 correspond to a path of -1.0 mm on axis A3.

Table 4-264 Configuration data for the SCARA kinematics

typeOfKinematics: SCARA	SCARA kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
offsetA1	Offset of the axis zero point of axis A1 relative to zero position of axis A1 in the transformation

distanceA1A2	Distance A1 - A2
offsetA2	Offset of the axis zero point of axis A2 relative to zero position of axis A2 in the transformation
distanceA2Endpoint	A2 - end point distance
linkCompensationA2.enableA1A2	Compensate articulated joint dependency of A1 to A2
linkCompensationA2.factorA1A2	Coupling factor for the compensation on axis A2
linkCompensationA4.enableA1A4	Compensate articulated joint dependency of A1 to A4
linkCompensationA4.factorA1A4	Coupling factor for the compensation on axis A4
linkCompensationA4.enableA2A4	Compensate articulated joint dependency of A2 to A4
linkCompensationA4.factorA2A4	Coupling factor for the compensation on axis A4
linkCompensationA3.enableA4A3	Compensate articulated joint dependency of A4 to A3
linkCompensationA3.factorA4A3	Coupling factor for the compensation on axis A3

Table 4-265 Possible articulated joint positioning spaces

LinkConstellation	1	Positive articulated joint positioning: Angle of axis A2 in the range [0°, 180°) relative to the kinematic zero point
	2	Negative articulated joint positioning: Angle of axis A2 in the range [-180°, 0°) relative to the kinematic zero point

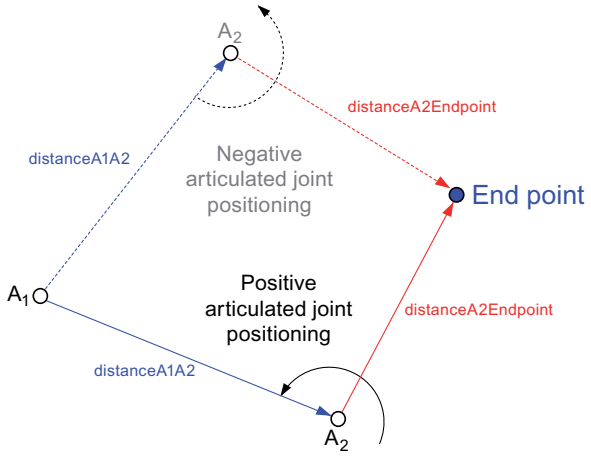


Figure 4-555 Possible articulated joint positionings

2D articulated arm kinematics

General

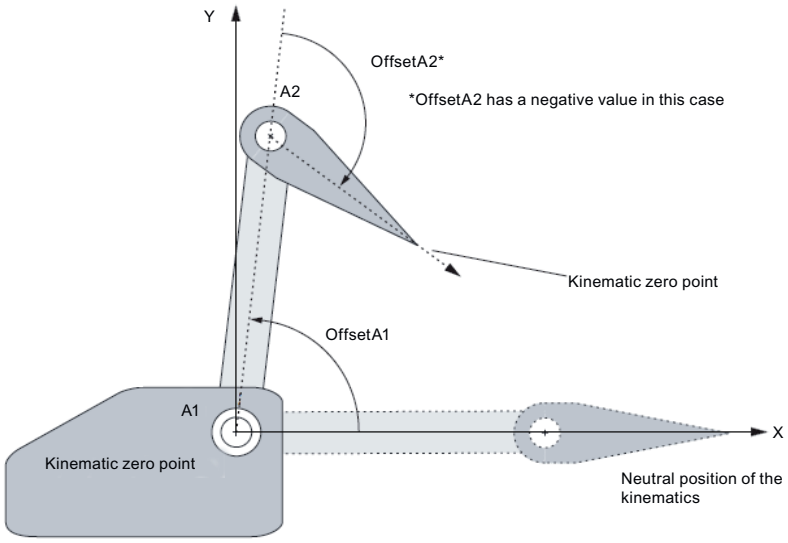


Figure 4-556 2D articulated arm: Display of the axes

Table 4-266 Configuration data for the 2D articulated arm kinematics

typeOfKinematics:ARTICULATED_ARM_2D	2D articulated arm kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
config2D	Main plane of the path axes
linkCompensationA2.enableA1A2	Compensate articulated joint dependency of A1 to A2
linkCompensationA2.factorA1A2	Coupling factor for the compensation on axis A2
distanceA1A2	Distance A1 - A2
offsetA1	Angle offset
offsetA2	Angle offset

Table 4-267 Possible articulated joint positioning spaces

LinkConstellation	1	Angle of axis A2 in the range [0°, 180°) relative to the kinematic zero point
	2	Angle of axis A2 in the range [-180°, 0°) relative to the kinematic zero point

3D articulated arm kinematics

General

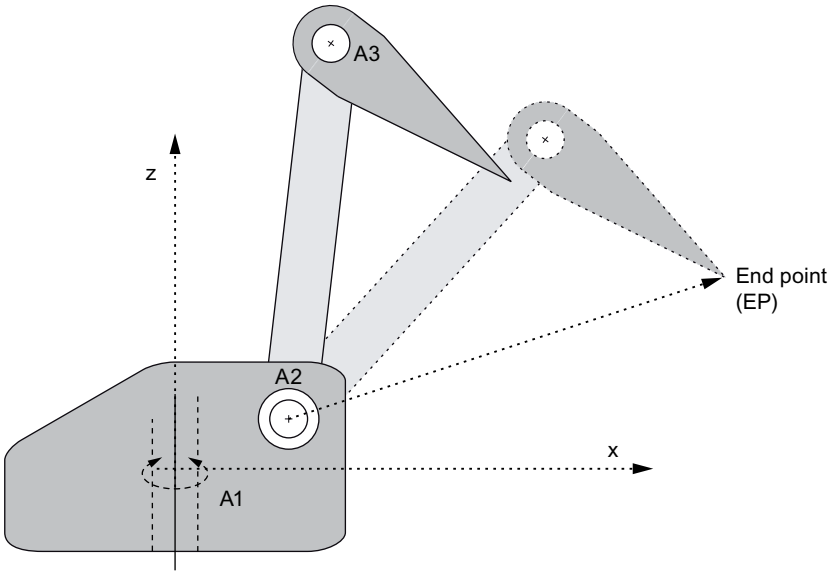


Figure 4-557 3D articulated arm: Display of the axes

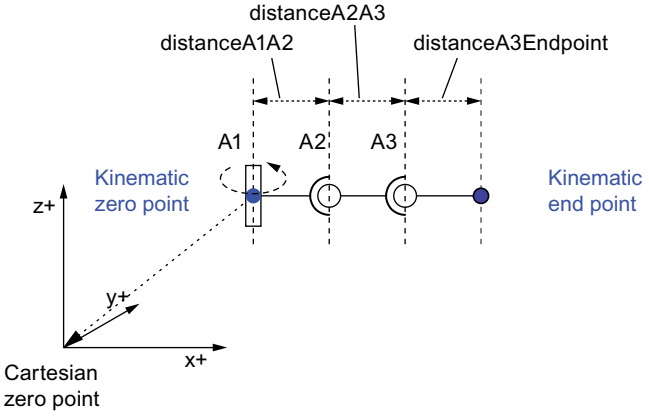


Figure 4-558 3D articulated arm: Kinematics

The kinematic zero point is at point A1.

The point is the zero position of the kinematics if **distanceA1A2**, **distanceA2A3**, and **distanceA3EP** are pointing in the Cartesian x-direction.

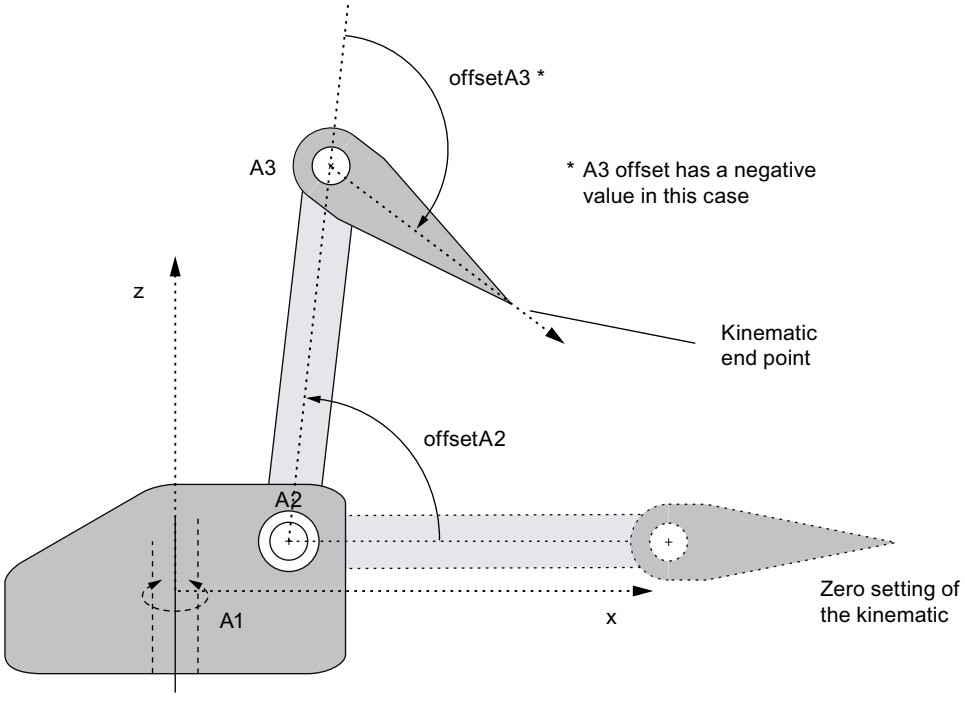


Figure 4-559 3D articulated arm: Zero positions of axes A2 and A3

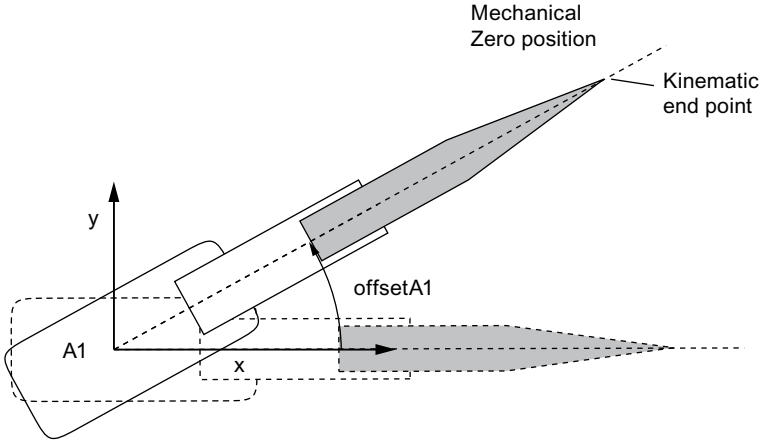


Figure 4-560 3D articulated arm: Zero position of axis A1

The definition range of the single A1 to A3 axes is limited to [-180°; 180°).

Coupled axes

If a positive coupling factor between two axes is specified, the transformation assumes that a positive motion on the first axes leads to a negative motion on the second axis.

Table 4-268 Configuration data for the 3D articulated arm kinematics

typeOfKinematics: ARTICULATED_ARM	3D articulated arm kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate

basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
offsetA1	Offset of axis zero point axis A1 relative to zero position of axis A1 in the transformation
distanceA1A2	Distance A1 - A2
offsetA2	Offset of axis zero point axis A2 relative to zero position of axis A2 in the transformation
distanceA2A3	A2 - A3 distance
offsetA3	Offset of axis zero point axis A3 relative to zero position of axis A3 in the transformation
distanceA3Endpoint	A3 - end point distance
linkCompensation.enableA2A3	Compensate articulated joint dependency of A3 to A2
linkCompensation.factorA2A3	Coupling factor for the compensation on axis A2

Table 4-269 Possible articulated joint positioning spaces

LinkConstellation	1	Angle of axis A3 in the range [0°, 180°) relative to the kinematic zero point Angle of axis A1 corresponds to atan(EPy/EPx)
	2	Angle of axis A3 in the range [-180°, 0°) relative to the kinematic zero point Angle of axis A1 corresponds to atan(EPy/EPx)
	3	Angle of axis A3 in the range [0°, 180°) relative to the kinematic zero point Angle of axis A1 corresponds to –atan(EPy/EPx)
	4	Angle of axis A3 in the range [-180°, 0°) relative to the kinematic zero point Angle of axis A1 corresponds to –atan(EPy/EPx)

2D swivel arm kinematics

2D swivel arm kinematics

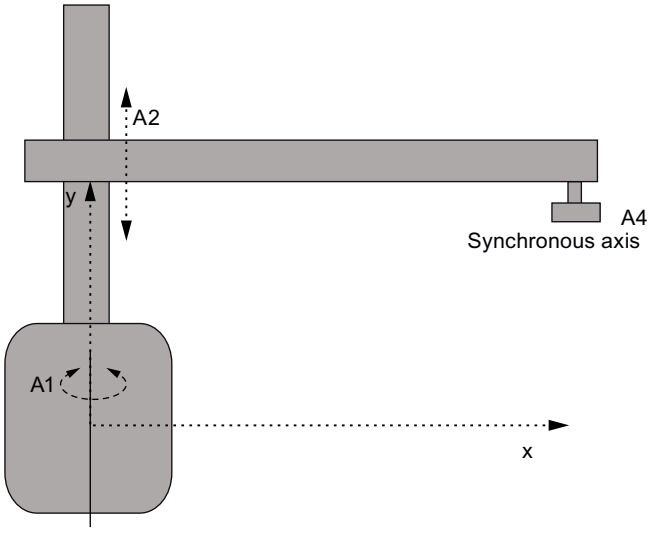


Figure 4-561 Display of the axes

2D swivel arm

In swivel arm kinetics, programming settings are made on the lateral surface that can be accessed by the kinematics.

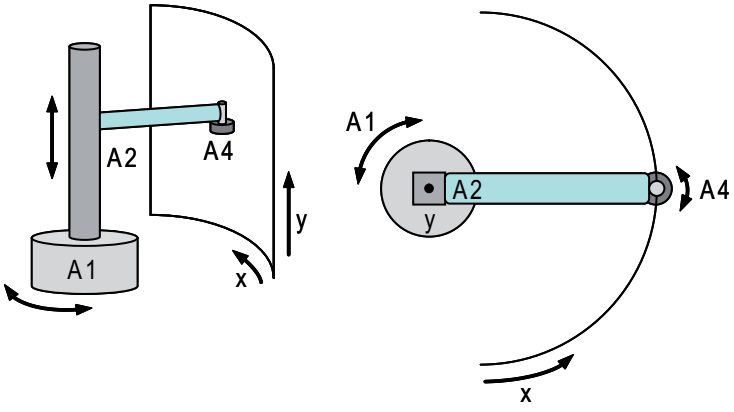


Figure 4-562 Kinematics working area

Unraveling the lateral surface results in a 2D plane, for which coordinate-plane and offset parameters can be assigned in the same way as with Cartesian 2D (Page 2517) kinematics. The offsets are applied to the set coordinate plane and rotation is about the axis that is perpendicular to the plane.

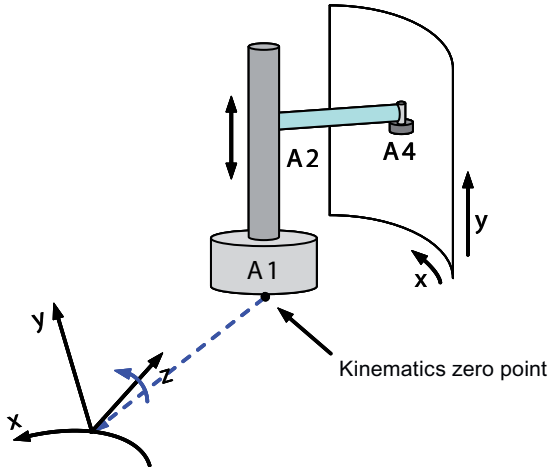


Figure 4-563 Kinematic offset and rotation

Depending on the plane used, the following parameters are effective:

X_Y plane:

Offset in x and y directions and rotation about the z axis

Y_Z plane:

Offset in y and z directions and rotation about the x axis

Z_X plane:

Offset in z and x directions and rotation about the y axis

With this type of kinematics, rotation does not serve any useful purpose.

The link compensations **LinkCompensationA1** and **LinkCompensationA2** and angular offset **offsetA1** at rotary joint A1 work in the same way as with SCARA kinematics (Page 2526).

With this type of kinematics, the conveyor tracking function (see Motion sequence at path object (Page 2545)) does not serve any useful purpose.

Table 4-270 Configuration data for the 2D swivel arm kinematics

typeOfKinematics:SWIVEL ARM	2D swivel arm kinematics type
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
config2D	Main plane of the path axes
linkCompensationA4.enableA1A4	Compensate articulated joint dependency of A1 to A4
linkCompensationA4.factorA1A4	Coupling factor for the compensation on axis A4

linkCompensationA2.enableA4A2	Compensate A4 articulated joint positioning dependence to A2
linkCompensationA2.factorA4A2	Coupling factor for the compensation on axis A2
distanceA1Endpoint	A1 - end point distance
offsetA1	Angle offset at rotary joint A1

3D cylindrical robot

General

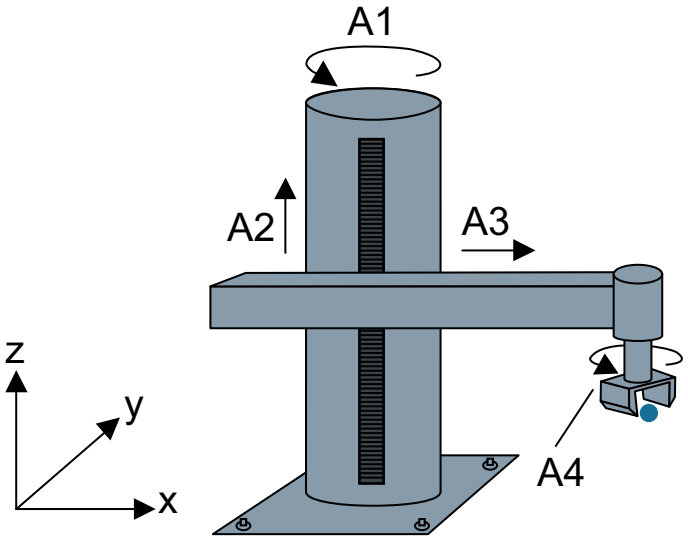


Figure 4-564 3D cylindrical robot: Overview of axes

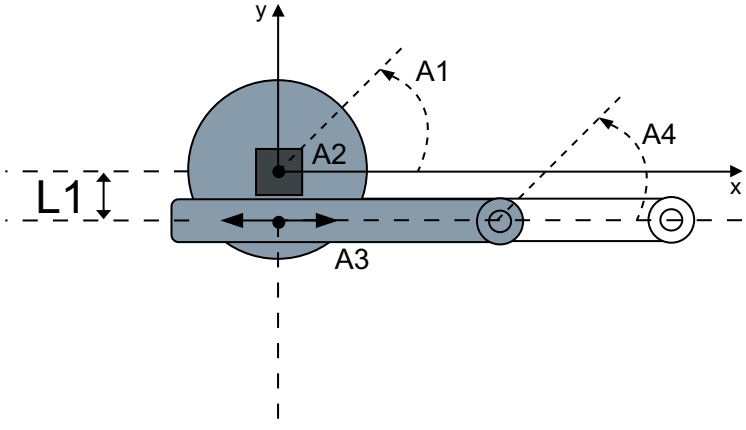


Figure 4-565 3D cylindrical robot: Plan view and display of the axes

In the 3D cylindrical robot, one of the Cartesian axes is replaced by a rotary axis. The working area is cylindrical.

Definitions

- A1, A2, A3 and A4 designate the four active drive axes of the kinematic structure. A1 and A4 are rotary axes and A2 and A3 linear axes.
- L1 designates the distance between axes A2 and A3. In the zero position, this corresponds to the distance between axis A3 to the x axis. This distance therefore also has an effect on the calculation of the end point of the transformation. Positive run in the direction of the y axis. In the above plan view, the value of L1 is therefore negative.
- The definition range of axis A1 is limited to [-180°; 180°).

Coupled axes

Mechanical couplings are possible:

- between A1 and A4
- between A4 and A2

The following axis couplings can be compensated via the system:

- a coupling from axis A1 to axis A4
- a coupling from axis A4 to axis A2

Effective direction of the coupled axes

For the coupling of axis A1 to axis A4, the following applies:

For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a positive motion on the second axis.

For the coupling of axis A4 to axis A2, the following applies:

For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a negative motion on the second axis.

The coupling between axis A4 and axis A2 is implemented as a spindle pitch, i.e. for a coupling factor of +1.0, 360.0 degrees on axis A4 correspond to a path of -1.0 mm on axis A2.

Table 4-271 Configuration data for the 3D cylindrical robot kinematics

typeOfKinematics: CYLINDRICAL	3D cylindrical robot kinematics type
basicOffset	Offset of the kinematic zero point (ZP) relative to the Cartesian zero point
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
distanceA2A3	Distance of axis A2 to axis A3
offsetA1	Angular offset of rotary axis A1
linkCompensation.enableA1A4	Compensate articulated joint dependency of axis A1 to axis A4

linkCompensation.factorA1A4	Coupling factor for the compensation on axis A4
linkCompensation.enableA4A2	Compensate articulated joint dependency of axis A4 to axis A2
linkCompensation.factorA4A2	Coupling factor for the compensation on axis A2

2D/3D user function

General

The kinematics 2D/3D user function allows the use of a transformation function provided by the user. This user function is created in ST notation and executed via a hook interface in the TO execution context. Compiled ST code in the TO execution context can be executed via the hook interface provided on the TO.

The function interface for the 2D/3D user function is permanently defined in ST. The input parameters are written by the TO and the calculated output values by the ST function.

Hook interface

Two different interfaces are defined for the forward and backward transformations. Each of the two transformation directions is therefore implemented by a function.

Table 4-272 FC input parameters for the forward transformation

Name	Type	Description
a1Position	LREAL	Position, axis 1
a1Velocity	LREAL	Velocity, axis 1
a1Acceleration	LREAL	Acceleration, axis 1
a2Position	LREAL	Position, axis 2
a2Velocity	LREAL	Velocity, axis 2
a2Acceleration	LREAL	Acceleration, axis 2
a3Position	LREAL	Position, axis 3
a3Velocity	LREAL	Velocity, axis 3
a3Acceleration	LREAL	Acceleration, axis 3
a4Position	LREAL	Position, axis 4
a4Velocity	LREAL	Velocity, axis 4
a4Acceleration	LREAL	Acceleration, axis 4
pathObject	_PathObjectType	Instance of the path object
executionContext	EnumPathObjectTransformationContext	Execution context `TO_INTERPOLATION_CYCLE`: FC call for calculation of the transformation in interpolation cycles, e.g. for path interpolation `NON_CYCLIC`: FC call for non-cyclic calculation of the transformation, e.g. for a query function
kinematicsConfigData	Array [1..32]	User-defined configuration data (corresponds to the configuration data item **parameter**)

Name	Type	Description
userTrafoID	UDINT	Identifier of the user-defined coordinate transformation (corresponds to the configuration data item **userTrafoId**)
kinematicsType	EnumPathUserKinematicsType	Setting of the kinematics for the user function (`_2D` or `_3D`)
config2D	EnumPathKinematicsConfig2D	Main plane of the path axes (is ignored for the 3D user function). Possible values are `X_Y`, `Y_Z` and `Z_X`)

Table 4-273 FC output parameters for the forward transformation

Name	Type	Description
functionResult	DINT	Function result 0: OK, velocity and acceleration have not been calculated 1: OK, velocity and acceleration have been calculated Other values: Error
xPosition	LREAL	X component of the position
xVelocity	LREAL	X component of the velocity
xAcceleration	LREAL	X component of the acceleration
yPosition	LREAL	Y component of the position
yVelocity	LREAL	Y component of the velocity
yAcceleration	LREAL	Y component of the acceleration
zPosition	LREAL	Z component of the position
zVelocity	LREAL	Z component of the velocity
zAcceleration	LREAL	Z component of the acceleration
wPosition	LREAL	W component of the position
wVelocity	LREAL	W component of the velocity
wAcceleration	LREAL	W component of the acceleration
linkConstellation	DINT	Articulated joint positioning

Table 4-274 FC input parameters for the backward transformation

Name	Type	Description
xPosition	LREAL	X component of the position
xVelocity	LREAL	X component of the velocity
xAcceleration	LREAL	X component of the acceleration
yPosition	LREAL	Y component of the position
yVelocity	LREAL	Y component of the velocity
yAcceleration	LREAL	Y component of the acceleration
zPosition	LREAL	Z component of the position
zVelocity	LREAL	Z component of the velocity
zAcceleration	LREAL	Z component of the acceleration

Name	Type	Description
wPosition	LREAL	W component of the position
wVelocity	LREAL	W component of the velocity
wAcceleration	LREAL	W component of the acceleration
w2Position	LREAL	W2 component of the position (path length)
w2Velocity	LREAL	W2 component of the velocity
w2Acceleration	LREAL	W2 component of the acceleration
linkConstellation	DINT	Articulated joint positioning
pathObject	_PathObjectType	Instance of the path object
executionContext	EnumPathObjectTransformationContext	Execution context TO_INTERPOLATION_CYCLE: FC call for calculation of the transformation in interpolation cycles, e.g. for path interpolation NON_CYCLIC: FC call for non-cyclic calculation of the transformation, e.g. for a query function
kinematicsConfigData	Array [1..32]	User-defined configuration data (corresponds to the configuration data item **parameter**)
userTrafoID	UDINT	Identifier of the user-defined coordinate transformation (corresponds to the configuration data item **userTrafoId**)
kinematicsType	EnumPathUserKinematicsType	Setting of the kinematics for the user function (_2D or _3D)
config2D	EnumPathKinematicsConfig2D	Main plane of the path axes (is ignored for the 3D user function). Possible values are X_Y, Y_Z and Z_X)

Table 4-275 FC output parameters for the backward transformation

Name	Type	Description
functionResult	DINT	Function result 0: OK, velocity and acceleration have not been calculated 1: OK, velocity and acceleration have been calculated Other values: Error
a1Position	LREAL	Position, axis 1
a1Velocity	LREAL	Velocity, axis 1
a1Acceleration	LREAL	Acceleration, axis 1
a2Position	LREAL	Position, axis 2
a2Velocity	LREAL	Velocity, axis 2
a2Acceleration	LREAL	Acceleration, axis 2
a3Position	LREAL	Position, axis 3
a3Velocity	LREAL	Velocity, axis 3

Name	Type	Description
a3Acceleration	LREAL	Acceleration, axis 3
a4Position	LREAL	Position, axis 4
a4Velocity	LREAL	Velocity, axis 4
a4Acceleration	LREAL	Acceleration, axis 4

Note

The functionResult output parameter

The user must set the functionResult output parameter to 1 if the velocities and the accelerations are to be calculated in the FCs in addition to the positions. If the parameter is not set explicitly, then it corresponds to 0. In this case, only the calculated position values of the FC are taken into account.

Note

Units for the interface variables

All variables at the interface are available in the set user units.

Function creation

The transformation functions can be created in the ST programming language. In order to be able to use the functions for the transformation, a pragma with attribute must be set.

```
ToHookApplicable := YES | NO
```

If the pragma is set, only certain operations are permitted.

Permitted operations

- Numeric and logic standard functions
- Access to bits in bit strings
- Standard functions for data type conversion
- Converting between any data types and byte arrays
- Combination of bit-string data types
- Functions for verification of floating-point numbers
- Functions for selection
- Access to global variables
- Consistent access to global variables of derived data types (UDT)
- Determination of the memory size of a variable or of a data type

Operations that are not permitted

- System functions on the TO
- Access functions to the TO (configuration data, system variables)
- Access functions to the I/O data and the I/O container
- System functions on devices

- Standard functions for controlling I/O modules and drive components
- System functions for controlling axes according to PLCopen
- Data backup and initialization from user program
- Communication functions
- System function blocks (for example counters, edge encoders or timers)
- Marshalling
- String machining
- Conversion of technology object data types

Note

In the functions of the user program, you must ensure that floating-point function calls (e.g ACOS) are called with valid values when working with the free transformation interface.

Example

Here is a minimal example of the FCs of the forward and backward transformation.

Forward transformation

```
FUNCTION FC_trans_Direct_Real  :StructPathDirectUserTransformationOut
{ToHookApplicable:= YES}
    VAR_INPUT
        ParIn : StructPathDirectUserTransformationIn;
    END_VAR
    FC_trans_Direct_Real.xPosition      := ParIn.a1Position;
    FC_trans_Direct_Real.yPosition      := ParIn.a2Position;
    FC_trans_Direct_Real.zPosition      := ParIn.a3Position;
    FC_trans_Direct_Real.wPosition      := ParIn.a4Position;
    FC_trans_Direct_Real.functionResult := 0;
END_FUNCTION
```

Backward transformation

```
FUNCTION FC_trans_Inverse_Real  :StructPathInverseUserTransformationOut
{ToHookApplicable:= YES}
    VAR_INPUT
        ParIn : StructPathInverseUserTransformationIn;
    END_VAR
    FC_trans_Inverse_Real.a1Position     := ParIn.xPosition;
    FC_trans_Inverse_Real.a2Position     := ParIn.yPosition;
    FC_trans_Inverse_Real.a3Position     := ParIn.zPosition;
    FC_trans_Inverse_Real.a4Position     := ParIn.wPosition;
    FC_trans_Inverse_Real.functionResult := 0;
END_FUNCTION
```

Execution time

The time for execution of the user transformation function is included in the execution time of the IPO system task. You will find more information on monitoring functions and settings of the IPO system task in the Basic functions Manual in Section AUTOHOTSPOT.

Breakpoints

For the FCs of the user transformation function, breakpoints are ignored.

Definitions

Table 4-276 Configuration data for the user-specific coordinate transformation

typeOfKinematics: USER_FUNCTION	2D/3D user function kinematics type
kinematicsType	Setting of the kinematics for the user function • **_2D**: 2D programming in a plane • **_3D**: 3D programming in space
basicOffset	Offset of the kinematic zero point (ZP) relative to the Cartesian zero point
basicOffset.x	Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate
basicOffset.y	Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate
basicOffset.z	Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate
basicOffset.roll	Rotation about x (roll)
basicOffset.pitch	Rotation about y (pitch)
basicOffset.yaw	Rotation about z (yaw)
config2D	Main plane of the path axes
parameter	Up to 32 user-defined parameters for the transformation
userTrafoID	Identifier of the user-defined coordinate transformation

Use of virtual axes

If you want to set up kinematics that cannot be mapped onto any existing kinematics but not all axes are used as real axes for these kinematics, you must create and interconnect the missing axes as virtual path axes.

As an example: The articulated arm kinematics provides three axes.

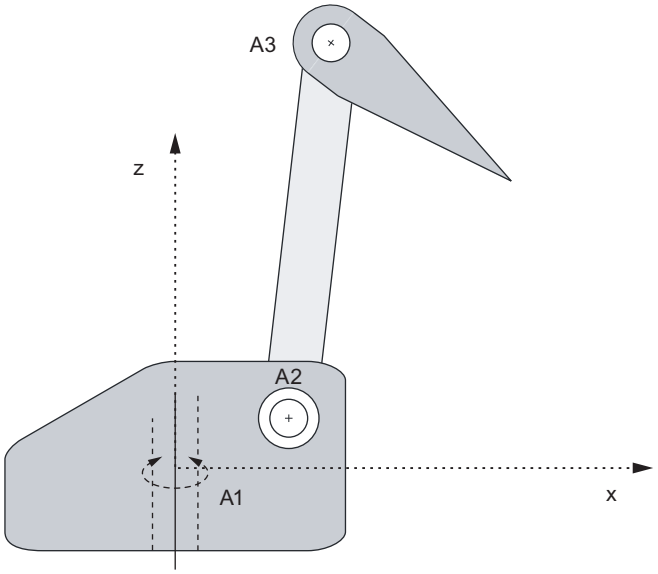


Figure 4-566 Articulated arm kinematics: Axes

- **A1** axis: 1. Path axis
- **A2** axis: 2. Path axis
- **A3** axis: 3. Path axis

If your kinematic system does not provide any axis for the first path axis, you must create a virtual axis for the first path axis and so interconnect the path object. The path object must then only be traversed and displayed in the corresponding main plane.

Specific kinematics

SPECIFIC kinematics type

The SPECIFIC kinematics type can be set using the typeOfKinematics configuration datum.

Example of settings on path object

Kinematics.typeOfKinematics = SPECIFIC (6)

The kinematics and parameters required can then be specified by the TrafoID and a parameter list.

See also

Appendix A (Page 2614)

4.6.3.14 Object coordinate systems and motion sequences on the path object

Object coordinate system (OCS) on the path object

As of SIMOTION V4.1.2, position details in the motion commands can be related optionally to the basic coordinate system (BCS, previously present functionality) or to an object coordinate system (OCS).

The motion sequence is available as from V4.2

An OCS can be permanent (static OCS) or coupled with a motion value supplied to the **trackingIn** interface of the path object.

A technology object that provides motion information with a position (the motion sequence reference value) can use the **TrackingIn** interface to interconnect with the path object. This can be, for example, an external encoder or a positioning axis.

If path motions relate to an OCS, then, for example, products can be taken from a moving belt or placed there.

Note

For an active motion sequence, a blending with dynamic response adaptation (`blendingMode:= ACTIVE_WITH_DYNAMIC_ADAPTION`) is not supported. Motions programmed with blending will then be performed without blending.

Three object coordinate systems are available on the path object and the positions of the working point are displayed in the relevant OCS (see also system variable `ocs`).

Activating OCS

The reference position of the OCS relative to the BCS is set again via the system function `_setPathObjectOcs`. The default values can be set in the SCOUT form **Object coordinate systems**. However, these values are only activated after `_setPathObjectOcs` has been called. The corresponding system variable for the default values is `userDefaultOcs`.

The default values can also be set by calibration (Page 2573). In this case, the new values are first written into `userDefaultOcs`. The reference position of the object coordinate system is itself not overwritten. For this, it is not necessary to call `_setPathObjectOcs`.

If the parameter `ocsSettingType` of the system function `_setPathObjectOcs` is set to the value `DIRECT`, values for offset and rotation can be entered directly.

Coupled OCS

A coupled OCS is an OCS coupled with a motion value of a technology object interconnected to the **trackingIn** interface.

Static OCS

A static OCS is an OCS not coupled with a motion value. The position of a static OCS is always the **OCS reference position**.

A static OCS can be used to perform motions in a coordinate system displaced relative to the BCS and has been rotated.

Motion sequence

The **motion sequence** functionality permits the synchronous coupling of a kinematic end point with a coupled OCS. It contains the functions for the synchronization and coupling with a moving product on a conveyor.

Motion sequence – fundamentals

Defining an OCS reference position

The reference position of the OCS is defined compared to the BCS in the OCS basic frame. The OCS basic frame contains the translation of the Cartesian X-, Y-, and Z-axes and the subsequent rotation at the individual axes.

To define the reference position of the OCS, the translation is performed first:

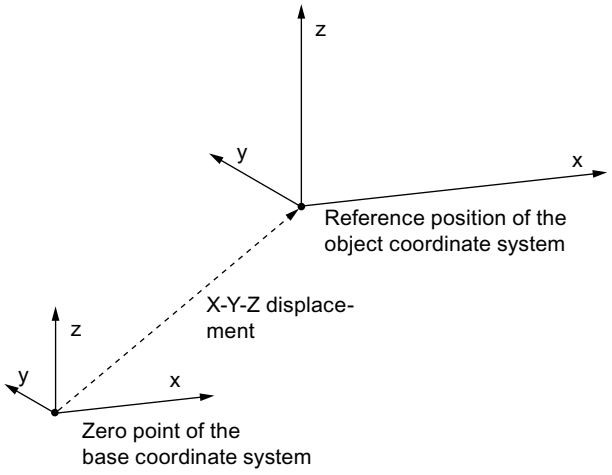


Figure 4-567 Translation of the OCS compared to the BCS

The rotations are then performed in the following sequence:

1. **Roll** at the X axis
2. **Pitch** at the (already turned) Y axis
3. **Yaw** at the (already twice-turned) Z axis

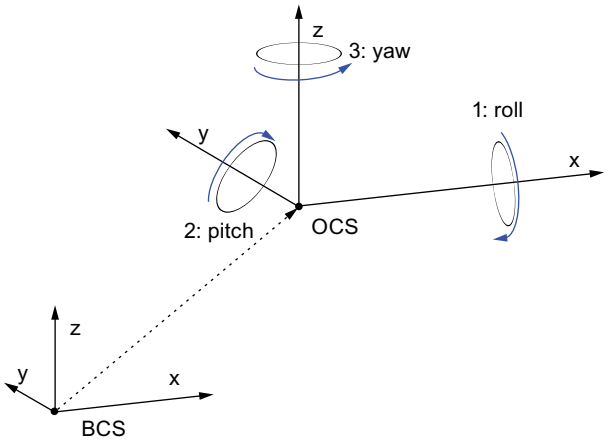


Figure 4-568 Rotations of the OCS

The **_setPathObjectOcs()** command can be used to set the basic frame for each OCS on the path object. Either default values in the system variables can be used or other values specified directly.

Definition of the terminology

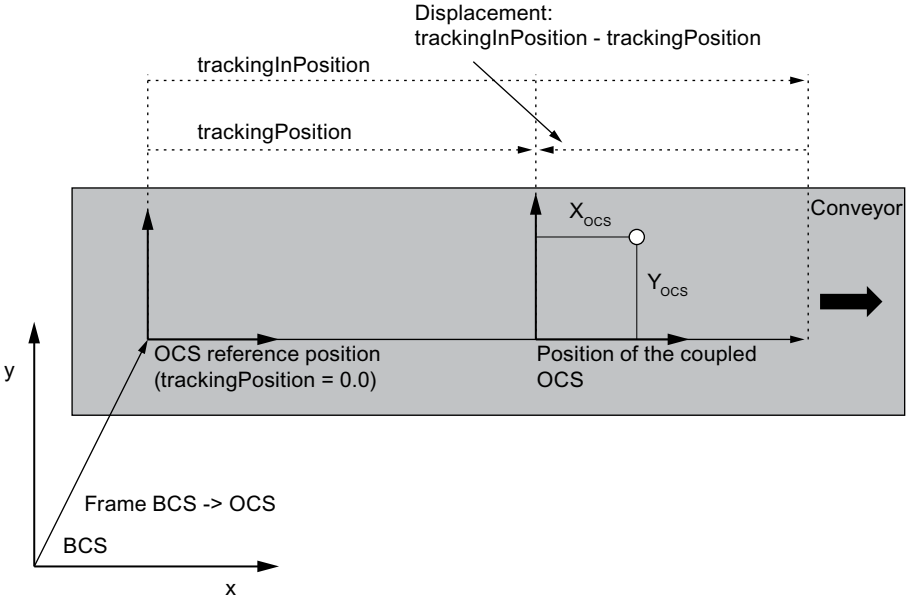


Figure 4-569 Schematic drawing of the motion sequence

OCS	Object coordinate system
BCS	Basic coordinate system
Frame	Translation and rotation of the OCS for the BCS

Reference position	Position of the OCS after translation/rotation in accordance with the basic frame
Motion sequence reference value	For example, actual value of an external encoder
Motion sequence value	Current position of the coupled OCS relative to the OCS reference position
X_{OCS} ,Y_{OCS}	Translation of the kinematic end point to the position of the coupled OCS

Assigning an OCS to a motion sequence reference value

The **trackingIn** input interconnection interface of the path object can be interconnected with another TO that provides an output interface with motion information. This can be, for example, the motion setpoint or actual value of an axis or the actual value of an external encoder.

The motion sequence value and the motion sequence reference value are assigned to the X-direction of the OCS. The OCS is coupled to the motion sequence reference value, the OCS coupling position is translated by the motion sequence value with regard to the OCS reference position in the X-direction of the OCS.

If the OCS is not interconnected or no TO is specified in the **_setPathObjectOcs()** command (**trackingIn:=TO#NIL**), the OCS then acts in its reference position.

If the kinematic end point is synchronized to a coupled OCS or is already synchronous with it (**trackingIn** <> **TO#NIL** and **trackingState** <> **INACTIVE**), the **_setPathObjectOCS()** command is not performed on this OCS and an error message issued. Before executing the command, the synchronized state ('**SYNCHRONIZED**' status) on this OCS must be ended.

See Terminate the coupling of the kinematic end point to a controlled OCS ('desynchronize') (Page 2552) for further information.

Defining the translation of the position of the coupled OCS

Because normally a product-based programming is performed, the position of the OCS on the conveyor must be modified appropriately for the product position, i.e. translated.

The **_redefinePathObjectOCS()** command is used to translate the position of the OCS in the X-direction and so in the direction of the value on the motion sequence input.

If the kinematic end point is synchronized to a coupled OCS or is already synchronous with it (**trackingIn <> TO#NIL** and **trackingState <> INACTIVE**), the **_redefinePathObjectOCS()** command is not performed on this OCS and the 30002 error message issued. Before executing the command, the synchronized state (**SYNCHRONIZED** status) on this OCS must be ended.

The current position values of the coupled OCS and the motion sequence input will be displayed in the following system variables:

- **Motion sequence reference value**
  The **trackingInPosition** system variable contains the position value present at the motion sequence input of the OCS, the motion sequence reference value (the conveyor position).
- **Motion sequence value**
  The **trackingPosition** system variable contains the position of the coupled OCS, the motion sequence value.
  trackingPosition = trackingInPosition + translation

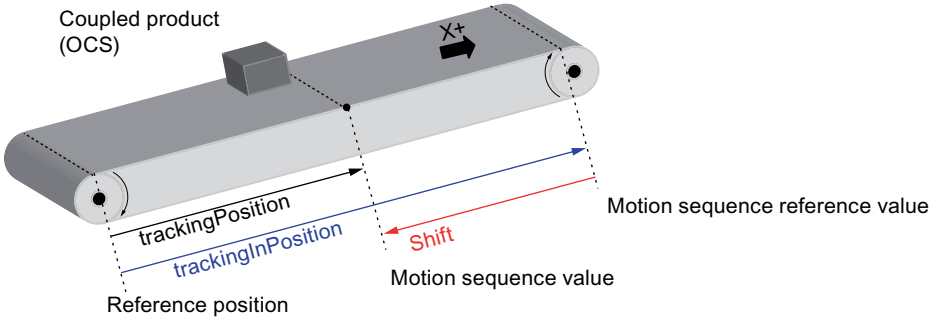


Figure 4-570 Current position of the OCS

Behavior of the OCS for modulo encoders

The motion sequence value indicates the continuing value on the motion sequence input without considering the modulo properties, i.e. the motion sequence value will not be reset when the motion sequence reference value on the modulo range end is reset, refer to the following figure.

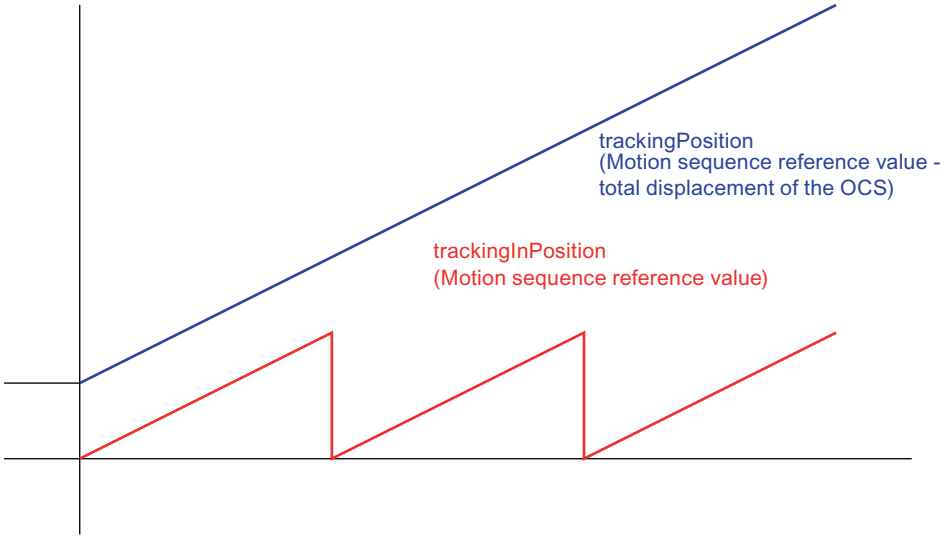


Figure 4-571 Behavior of trackingPosition for modulo-assigned value in trackingInPosition

Resetting trackingPosition

The **_redefinePathObjectOCS()** function can be used to set or translate **trackingPosition** only when the kinematic end point is not synchronous to this OCS or currently being synchronized (indicated using the **trackingState:=INACTIVE** variable).

There are 2 modes for translating the **trackingPosition** with the function **_redefinePathObjectOcs()** - absolute or relative.

For the respective mode the value for **trackingPosition** can be calculated as follows:

- For **mode:= RELATIVE**
  trackingPosition:=trackingPosition + value
- For **mode:= ABSOLUTE**
  trackingPosition:=trackingPosition + value

The use of **mode:=ABSOLUTE** has the advantage that translating always refers to trackingInPosition, which means that previous translations do not have to be buffered in the application. For **mode:= RELATIVE** it is necessary to buffer the translation in the application.

The **_redefinePathObjectOCS()** and **_setPathObjectOCS()** functions are not executed for the associated OCS when it is in the '**SYNCHRONIZED**' status.

See Terminate the coupling of the kinematic end point to a controlled OCS ('desynchronize') (Page 2552) for further information.

Synchronizing motion on the path object to the coupled OCS

A handling application for which, for example, a product is to be fetched from a moving conveyor, is realized with an OCS coupled with the conveyor. The motion commands are configured here so they act directly in the OCS.

This requires the motion calculated for the path object for the kinematic end point to be synchronized to the coupled OCS.

In the simplest case, after the synchronization, the kinematic end point moves with a defined point in the OCS and so with a point located on the conveyor. Furthermore, after the synchronization in the coupled OCS, linear, circular or polynomial paths can also be followed.

The **_enablePathObjectTrackingSuperimposed()** command is used to synchronize the kinematic end point with an OCS coupled with the motion sequence reference value (e.g. position of a conveyor). Some of the arguments specified with the command:

- Synchronization mode (**synchronizingMode**)
  The following synchronization modes are available:
  - Other coupling with the position in the OCS specified in the command
    (setting: **synchronizingMode:=IMMEDIATELY**)
    Synchronization is executed immediately and coupled with the OCS.
  - Synchronization and coupling in the OCS at the position of the OCS specified in the command, i.e. as soon as the **ocsTrackingPosition** (e.g. of a conveyor belt) has reached a specified value (the synchronization position)
    (setting: **synchronizingMode:=ON_POSITION**).
    For this synchronization mode, a preliminary synchronization is made to the specified synchronization position in the OCS.
- The synchronization position (**position**)
  The specification of a motion sequence value, above which travel synchronous with the OCS is to take place. This value is used only for **synchronizingMode:=ON_POSITION**.

The following applies for both synchronization modes:

- The synchronization on the conveyor belt occurs where necessary superjacent to a motion that is still active in the BCS.
- No further motion commands are possible in the BCS during synchronization.
- Motion commands in the OCS are only possible once the status **SYNCHRONIZED** has been reached.

The desired position of the product in the OCS can be approached after the synchronization via a path command in the OCS.

The following applies for the synchronization mode **ON_POSITION**:

The synchronization process will be aborted when the direction of the encoder value reverses during the synchronization.

This can occur when the external encoder of a conveyor belt delivers a fluctuating position value during standstill due to missing filters or an insufficient tolerance window (with Extrapolation) which results in a direction reversal of the acutal position.

Synchronization status

The synchronization status of the kinematic end point to a coupled OCS is indicated in the **ocs[i].trackingState** system variables on the OCS.

The synchronization status is **SYNCHRONIZED** when

- there is **no motion active in the BCS**, the speed of the kinematic end point is equal to the speed of the conveyor belt and the position misalignment of the synchronization motion resulting from the synchronization has been rectified.
- **a motion is active in the BCS**, the speed of the overlying synchronization motion is equal to the speed of the conveyor belt and the position misalignment of the synchronization motion resulting from the synchronization has been rectified.

A static OCS interconnected with **TO#NIL** always has the **SYNCHRONIZED** status. In addition to the static OCS, not more than one coupled OCS can have the **SYNCHRONIZED** status.

Dynamic values for the synchronization action

Dynamic values for the synchronization action can be specified in the **_enablePathObjectTrackingSuperimposed ()** command.

The default dynamic values of the path object can be used or the values specified explicitly.

The overall dynamics during the synchronization process result from the active path motion in the BCS (where necessary) and the overlying synchronization motion. This must be taken into consideration when specifying the dynamics, as otherwise this can cause the programmed or maximum dynamic values on the path object to be exceeded.

Performing path motions in the coupled OCS

Path motions can be related using the **csType** command parameter optionally to the BCS or an OCS.

Prerequisite for path commands in the OCS acting is that the OCS has the **SYNCHRONIZED** status. Otherwise the path motion command for the OCS will not be performed. Path motion commands for the OCS can only be issued after synchronization.

- **csType**
  This parameter specifies whether the coordinates apply to the OCS or the BCS.
- **csNumber**
  This parameter is the index of the OCS (1...3).

Terminate the coupling of the kinematic end point to a controlled OCS ('desynchronize')

The coupling of the kinematic end point to a controlled OCS is terminated by the coming into force of a path motion command related to the BCS or a static OCS.

Any existing path motion commands are discarded; the new path motion command will be executed immediately.

The system variables for **PATH** on the path object indicate the state of the path in the BCS or OCS coordinates system selected.

When revoking a synchronous motion sequence (conveyor synchronization) using **_stopPath** in the BCS, **no path active** and/or zero values are displayed for removal of the motion from the motion sequence.

Stopping in the OCS

The **_stopPath()** command can be stopped relative to the OCS. The **SYNCHRONIZED** status with the coupled OCS is retained. This means the motion can be continued using **_continuePath()** relative to the coupled OCS.

Motion sequence – sample application

Sample application of an OCS

The use of an OCS for the motion sequence is explained using a short example.

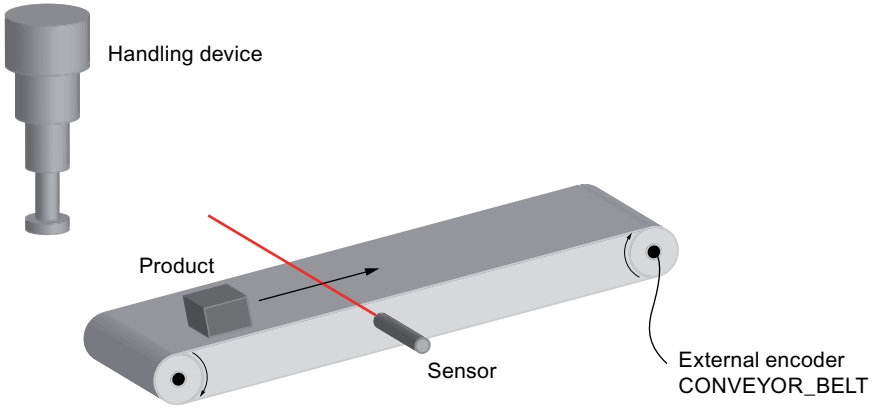


Figure 4-572 Overview of the sample application

In this example, products are placed on a conveyor. A sensor records the exact position of the products. The handling device should fetch products from the conveyor and place them at another location.

Defining the reference position of the OCS

The reference position of the OCS is defined in the system variables.

In this example, the OCS1 is used. The settings for this coordinate system are made in the **userdefaultocs[1]** structure.

userdefaultocs	User defaults of the object coordinate s		
userdefaultocs[1]	User defaults of the object coordinate s		
frame	Transformation frame		
pitch	Rotation at the Y vector after rotation at	-15.0	°
roll	Rotation at the X vector after the offset	0.0	°
x	Offset in X direction	100.0	mm
y	Offset in Y direction	0.0	mm
yaw	Rotation at the Z vector after rotation at	0.0	°
z	Offset in Z direction	15.0	mm
userdefaultocs[2]	User defaults of the object coordinate s		
userdefaultocs[3]	User defaults of the object coordinate s		

Figure 4-573 OCS-relevant system variables

Consequently, the OCS is displaced by 100 mm in the X-direction and 15 mm in the Z-direction:

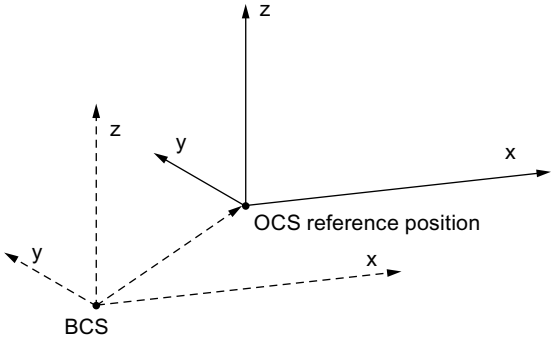


Figure 4-574 Displacement of the OCS

The OCS is rotated by -15° at the Y-axis.

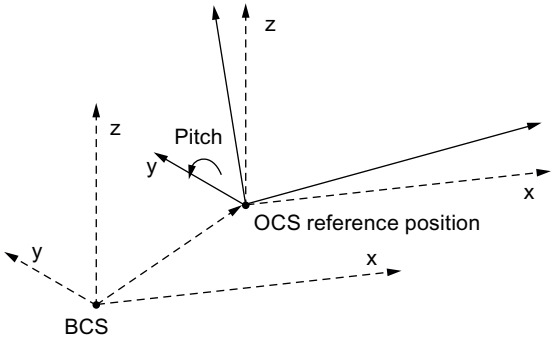


Figure 4-575 Rotation of the OCS

Determining the motion sequence reference value of the OCS

The position and motion data of the conveyor are acquired using the **CONVEYOR_BELT** external encoder. For the OCS, the base frame is set and activated with the **CONVEYOR_BELT** motion sequence reference value. The OCS is then coupled with the conveyor, in particular, at the position supplied by the **CONVEYOR_BELT** external encoder.

```
// Set OCS_1 to CONVEYOR_BELT,
// for the BCS base frame for the OCS,
// use the default settings.
myRetDINT :=
    _setPathObjectOcs(
        pathObject:=Portal_3D,
        ocsNumber:=1,
        trackingIn:=CONVEYOR_BELT,
        ocsSettingType:=USER_DEFAULT
);
```

Defining the position of the OCS relative to the motion sequence reference value

The sensor, for example, a light barrier, is triggered by the passing product. The current value of the **CONVEYOR_BELT** external encoder is stored in the **belt_position** variable.

Because the position of the sensor related to the reference position of the OCS is known, the position of the product with regard to the motion sequence reference value is known.

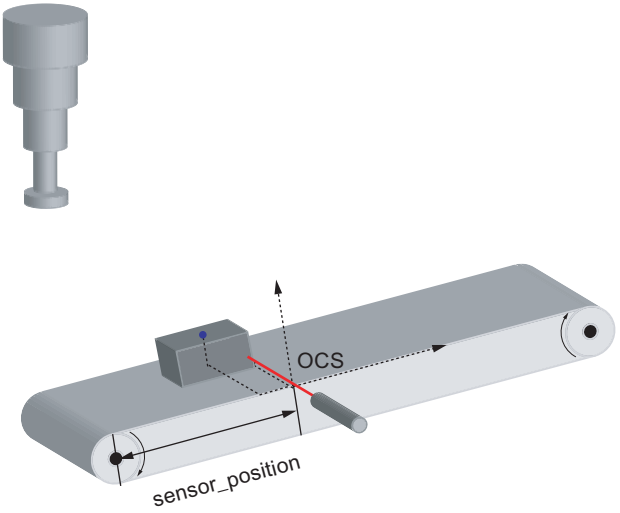


Figure 4-576 Defining the position of the OCS

```
myRetDINT :=
    _redefinePathObjectOcs(
        pathObject:=Portal_3D,
        ocsNumber:=1,
        mode:=RELATIVE,
        value:=belt_position - sensor_position
);
```

Synchronizing motion on the path object to the coupled OCS

The handling device should be coupled synchronous with the product after a **synch_space** travel length after the sensor.

The acting point position of the grabber is specified in the OCS. In the example, this is done using the **offset_x**, **offset_y** and **offset_z** variables. This displacement is then used for positioning after synchronization by means of a command in the OCS so that the gripper can hold the product above its center of gravity.

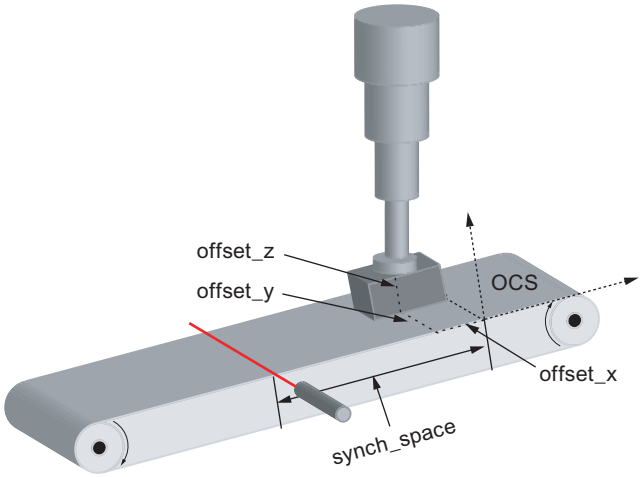


Figure 4-577 Synchronizing the handling device

```
myRetDINT :=
    _enablePathObjectTrackingSuperimposed(
        pathObject:=Portal_3D,
        ocsNumber:=1,
        synchronizingMode:=ON_POSITION,
        position:=sensor_position + synch_space
);
```

When the status "synchronous" has been reached (**ocs[1].trackingState = SYNCHRONIZED**), the command for positioning the gripper at the acting point of the product (**offset_x**, **offset_y**, **offset_z**) can be issued in the OCS.

```
myRetDINT :=
    _movePathLinear(
        pathObject:=Portal_3D,
        pathMode:=ABSOLUTE,
        x:=offset_x,
        y:=offset_y,
        z:=offset_z,
        csType:=OCS,
        csNumber:=1
);
```

Performing path motions in the coupled OCS

The handling device now moves itself 15 mm away from the conveyor and brings the product to the placement position (**dispose_x**, **dispose_y**, **dispose_z**). The first motion occurs in the OCS, the second in the BCS. The synchronization is terminated by calling the second command.

```
myRetDINT :=
    _movePathLinear(
        pathObject:=Portal_3D,
        pathMode:=RELATIVE,
        z:=15.0,
        csType:=OCS,
        csNumber:=1
);

myRetDINT :=
    _movePathLinear(
        pathObject:=Portal_3D,
        pathMode:=ABSOLUTE,
        x:=dispose_x,
        y:=dispose_y,
        z:=dispose_z,
        cstype:=BCS
);
```

4.6.3.15 Interconnection, interconnection rules

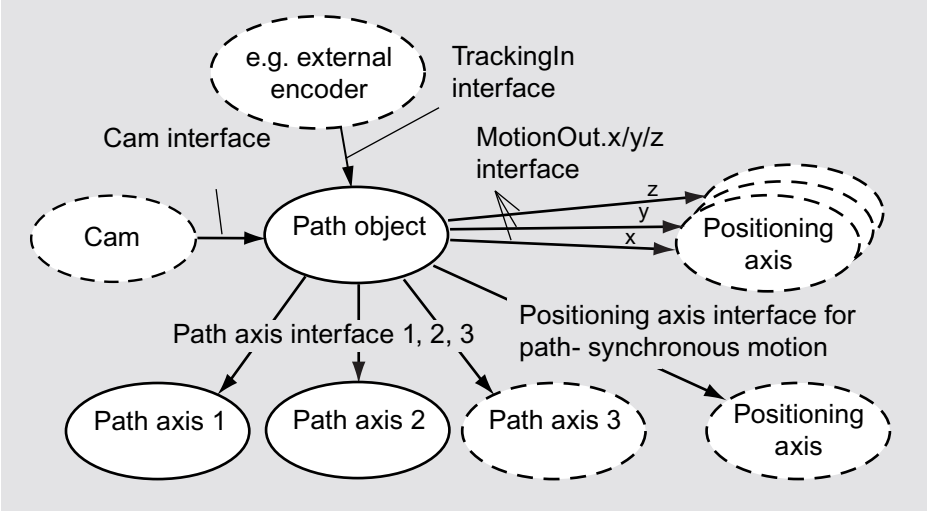


Figure 4-578 Interfaces on the path object

- Path axis interface 1 and path axis interface 2 of a path object must be interconnected with path axes.
- Path axis interface 3 of the path object can optionally be interconnected with a path axis, irrespective of the kinematic settings.
- The positioning axis interface for the path-synchronous motion can optionally be interconnected with a positioning axis.
- The MotionOut.x, MotionOut.y or MotionOut.z interface can optionally be interconnected with a positioning axis.

- The path object can be interconnected with a cam for specifying a velocity profile.
- To specify a motion sequence reference value, the TrackingIn interface can be interconnected with a TO that provides an output interface with position value.

For additional information, see **Motion Control Basic Functions** Function Manual, "Available technology objects".

Notes:

- The path interfaces cannot be distributed, i.e. all objects involved in the path group (path object, path axes, and positioning axes) must be on the same device.
- The objects involved in a path interpolation group (path object, path axes and, if applicable, a positioning axis) must be assigned to the same IPO or IPO_2 execution level. The SERVO setting is not possible.
- The currently effective interconnections are shown in the system variables **specificVelocityProfile**, **motionBuffer**, **connections**.

4.6.3.16 Simulation operation

A path interpolation can be switched to simulation, i.e. values are calculated on the path object but are not output to the slave axes/positioning axis.

It is possible to enable and disable the path simulation at any time, including while the relevant axes are in motion, provided there is no error response.

The **simulation [ACTIVE/INACTIVE]** system variable provides information about the simulation status of the path object.

Commands for the simulation operation

- The **_enablePathObjectSimulation()** command sets the path interpolation to simulation mode.
  Values are calculated but are not output to the path axes/positioning axis. This can be done at any time.
- The **_disablePathObjectSimulation()** command resets the path interpolation from simulation mode.
  Values are output to the path axes/positioning axis again.
  If there is a discrepancy between the axis setpoint calculated from the path interpolation and the current setpoint on the axis, the change in the axis setpoint on the axis is limited due to the maximum dynamic limits of the axis.

Maintaining the setpoint calculation on the path object even when the axis enables are canceled

The configuration data **decodingConfig.disablePathOperation** can be used to specify whether the setpoints on the path object will continue to be calculated even when the axis enables are canceled.

- If **NO** (default), the path interpolation is also canceled in simulation mode if the enables on the path axis/positioning axis have been canceled.
- If **YES**, the path interpolation is not canceled in simulation mode if the enables on the path axis/positioning axis have been canceled while the path object is in simulation mode.
  Any path commands that are undergoing execution are retained.

4.6.4 Configuring the Path Object

4.6.4.1 Selecting the path interpolation technology package

1. Select the device in the project navigator and select **Select technology packages** in the shortcut menu (right-click).
2. Select the **PATH** option and confirm with **OK**.

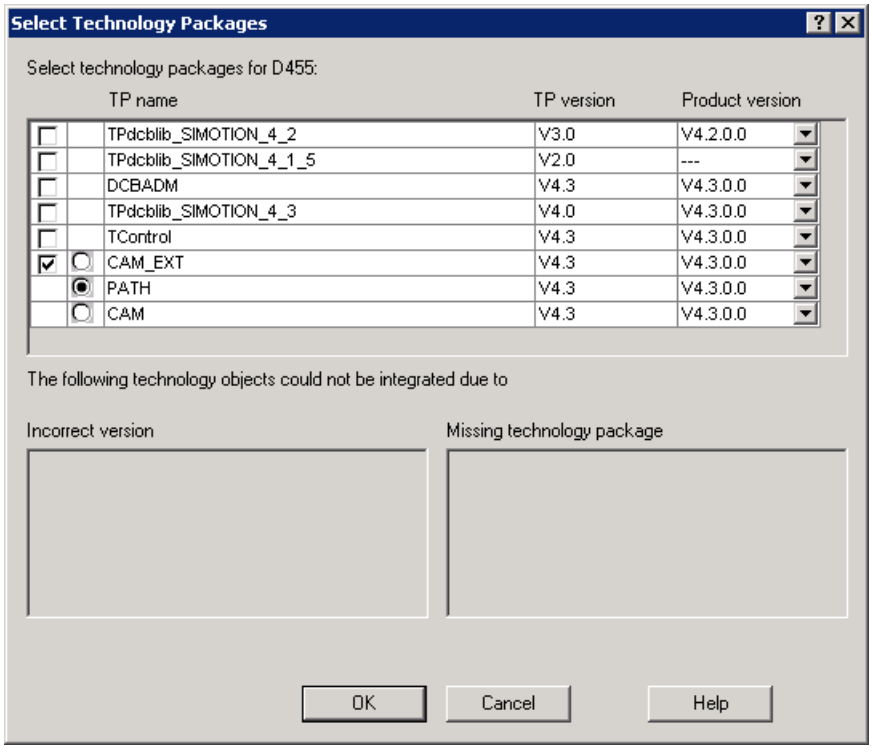


Figure 4-579 Selection of technology packages

4.6.4.2 Creating axes with path interpolation

- When creating an axis in **SCOUT**, enable the **path interpolation** technology.

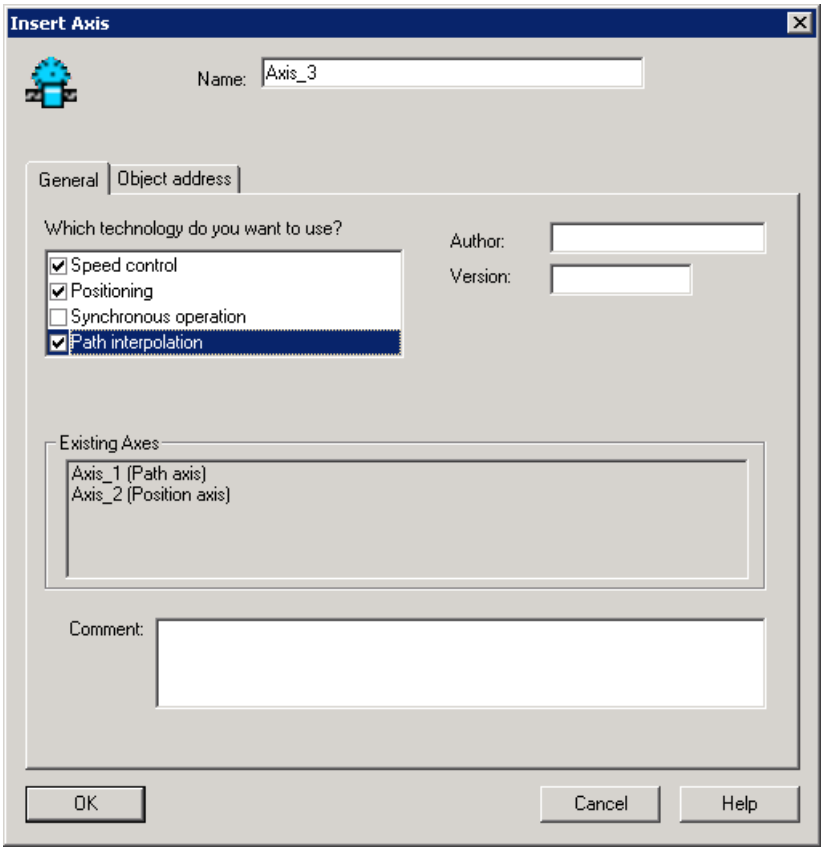


Figure 4-580 Inserting an axis with path interpolation

Notes

When you specify the path interpolation technology for an Axis technology object, a path object is not inserted automatically.

A positioning axis, for example, cannot be changed into a path axis at a later point.

Interconnections are made on the path object.
See Interconnecting a path object (Page 2570).

4.6.4.3 Creating a path object

In SIMOTION SCOUT, path objects are created at the same level as an Axis and a Cam technology object. These path objects can be assigned to all applicable axes of the device.

1. To create a new path object, double-click **Insert path object** under **PATH OBJECTS** in the project navigator.
   You can also copy an existing path object to the clipboard and then insert it under another name.

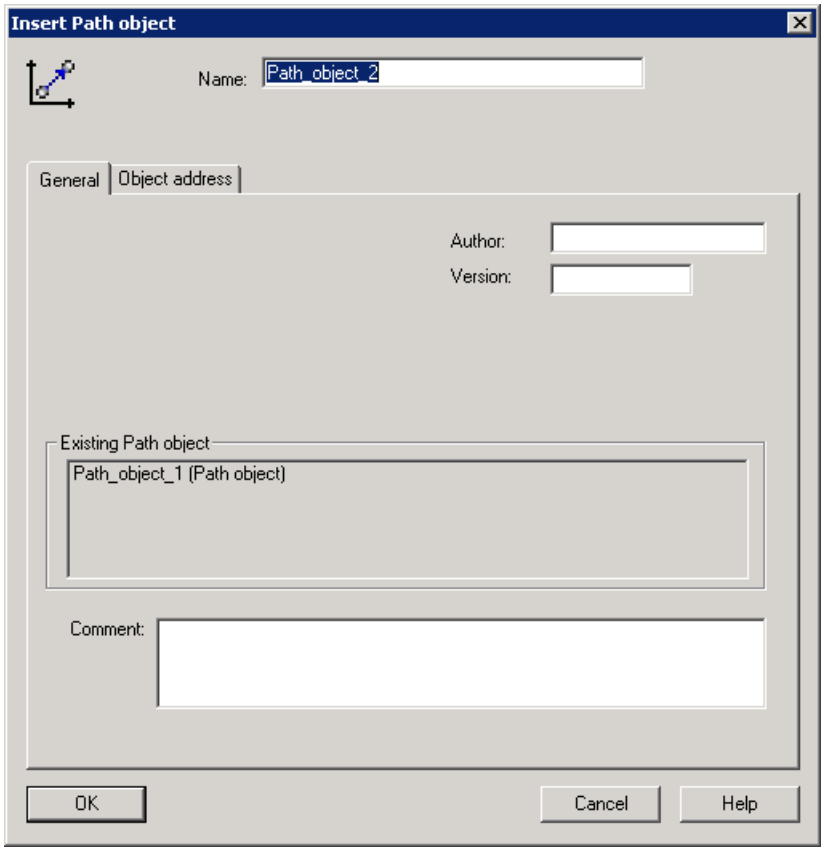


Figure 4-581 Inserting a path object

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm.

The new path object will be inserted under **TECHNOLOGY**.

4.6.4.4 Representation in the project navigator

The path object appears in the project navigator at the same level as the Axis and Cam technology objects. Links symbolize the connection to path axes or a positioning axis for path-synchronous motion.

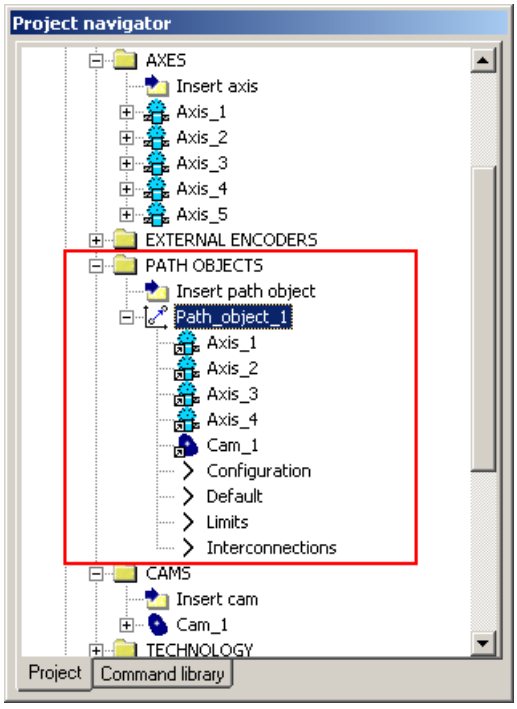


Figure 4-582 Project with path interpolation in the project navigator

4.6.4.5 Assigning path object parameters/default values

- In the project navigator, double-click **Default** under the path object.
  In this window, you define the substitute values (default) for calling the path object (**_movePath...()**, **_stopPath()**, **_setPathObjectOcs()**, etc.).

Default for object coordinate systems

On the **Object coordinate systems**, you can define the default values for the three available OCS.

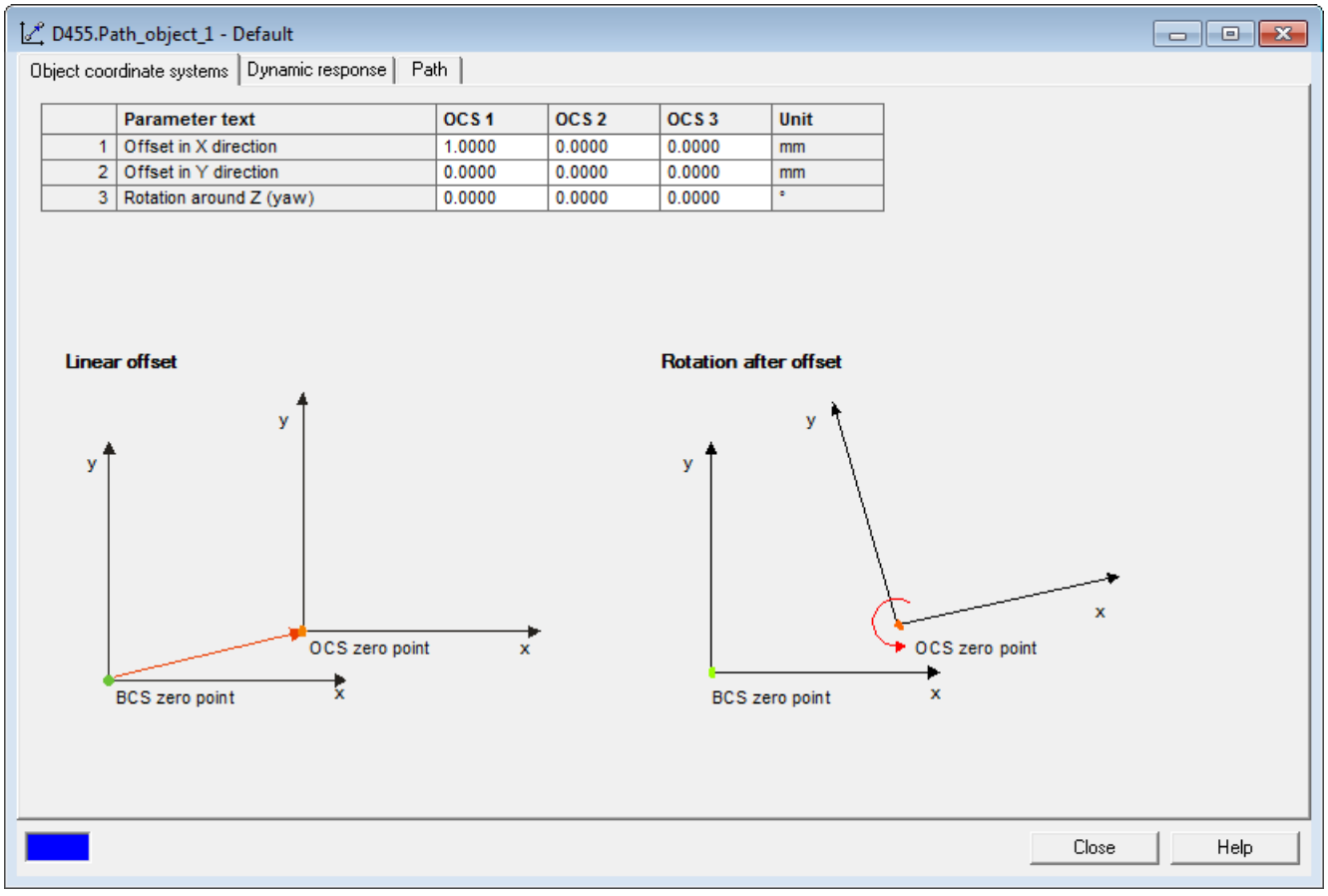


Figure 4-583 Path object: Default - Object coordinate systems tab

In this window, you can define the values for the object coordinate systems that are used by default.

For more information, refer to Object coordinate system (OCS) on the path object (Page 2545).

Dynamic response parameters

You specify the path velocity, the velocity profile, and the acceleration/deceleration and jerk on the **Dynamic response** tab.

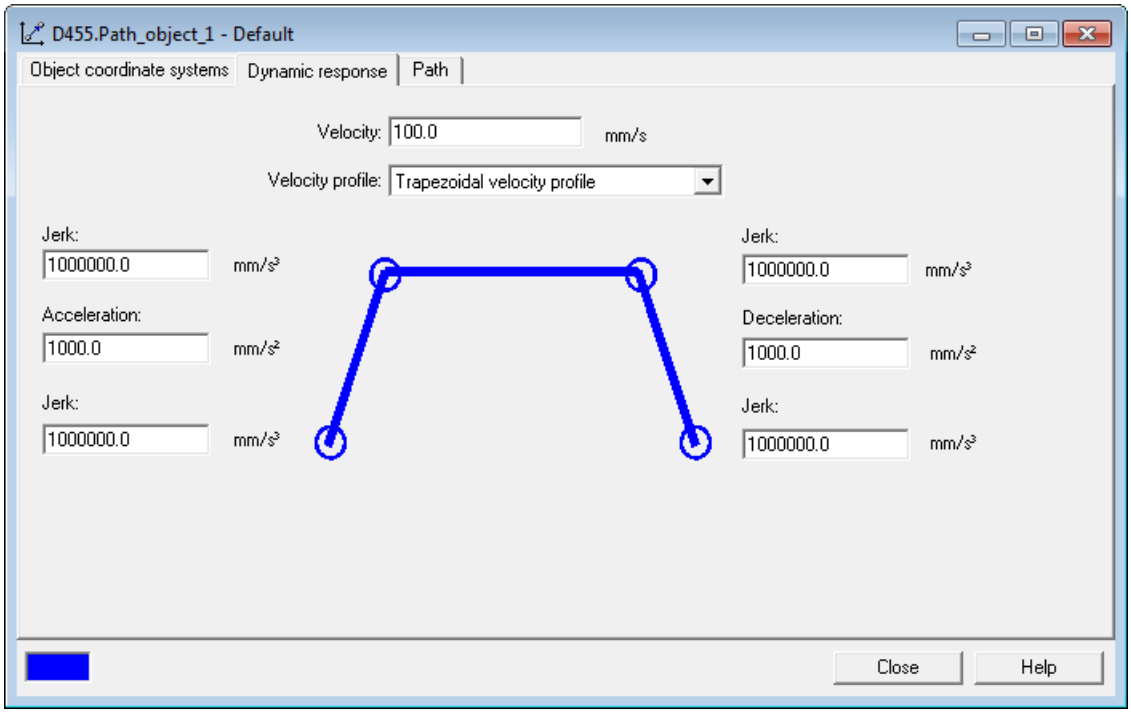


Figure 4-584 Path object: Default settings - Dynamic response tab

You define the substitute values (default settings) for the dynamic response in this window.

You can set the following parameters:

Field/button	Meaning/instruction
Velocity	Enter the substitute value for the path velocity here. (`userDefault.pathdynamics.velocity`)
Velocity profile	Select the velocity profile here. (`userDefault.pathdynamics.profile`)
Acceleration	Enter the substitute value for the path acceleration here. (`userDefault.pathdynamics.positiveAccel`)
Deceleration	Enter the substitute value for the path deceleration here. (`userDefault.pathdynamics.negativeAccel`)
Jerk	Enter the substitute value for the path jerk here. (`userDefault.pathdynamics.positiveAccelStartJerk` / `positiveAccelEndJerk` / `negativeAccelStartJerk` / `negativeAccelEndJerk` / `profile`)

For more information, refer to Path dynamics (Page 2491).

The meaning of the parameters in the dialog box and their permissible value ranges can be found in the SIMOTION reference lists.

Path parameters

You specify the default settings for the path on the **Path** tab.

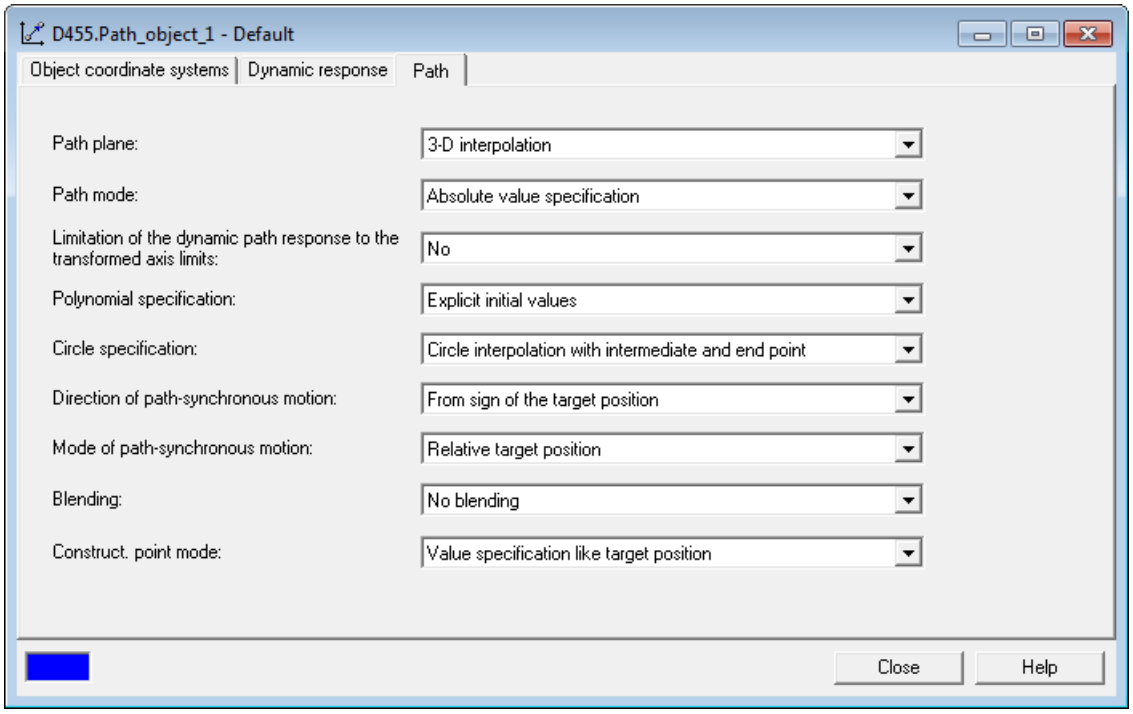


Figure 4-585 Path object: Default settings - Path tab

You define the substitute values (default settings) for the path in this window.

You can set the following parameters:

Field/button	Meaning/instruction
Path plane	Here, you specify the path plane: X_Y_Z / X_Y / Y_Z / Z_X (default: X_Y_Z, for 3D; for 2D kinematics, this is implicitly X_Y) (`userDefault.path.plane`)
Path mode	Specify the path mode: absolute or relative (userDefault.path.mode)
Limit dynamic path response to transformed axis limit values	Here, you specify whether the dynamic path response should be limited to the transformed axis limit values. (`userDefault.path.dynamicAdaption`) See Limiting the path dynamics (Page 2492).
Polynomial default	Here you can define the type of polynomial interpolation. (`userDefault.path.polynomialMode`) See Polynomial paths (Page 2485).
Circle default	Here you can define the type of circular interpolation. (`userDefault.path.circularType`) See Circular paths (Page 2481).
Direction of synchronous path motion	Here, you specify the direction of the positioning axis for path-synchronous motion. (`userDefault.w.direction`) See Functionality of path-synchronous motion (Page 2505).

Field/button	Meaning/instruction
Mode of path-synchronous motion	Here, you specify the synchronous axis mode: • Absolute • Relative • Output path lengths • Additive output of path lengths (`userDefault.w.mode`) See Functionality of path-synchronous motion (Page 2505).
Blending	Here, you specify whether and how blending is performed. (`userDefault.blendingMode`) See Path behavior at motion end (Page 2496).
Construction point mode	Center point or intermediate point: • Absolute • Relative • Same as target position mode (`userDefault.path.ijkMode`) See Circular paths (Page 2481).

Note:

Transformations are set via the Configuration (Page 2565) form or via the expert list.

For more information, refer to Basics of Path Interpolation (Page 2474).

The meaning of the parameters in the dialog box and their permissible value ranges can be found in the SIMOTION reference lists.

4.6.4.6 Configuring a path object

- In the project navigator, double-click **Configuration** under the path object.

Configuration of the path object

On the **Configuration** tab car, you define the type of kinematics, the processing cycles, and, depending on the kinematics, the coordinate plane.

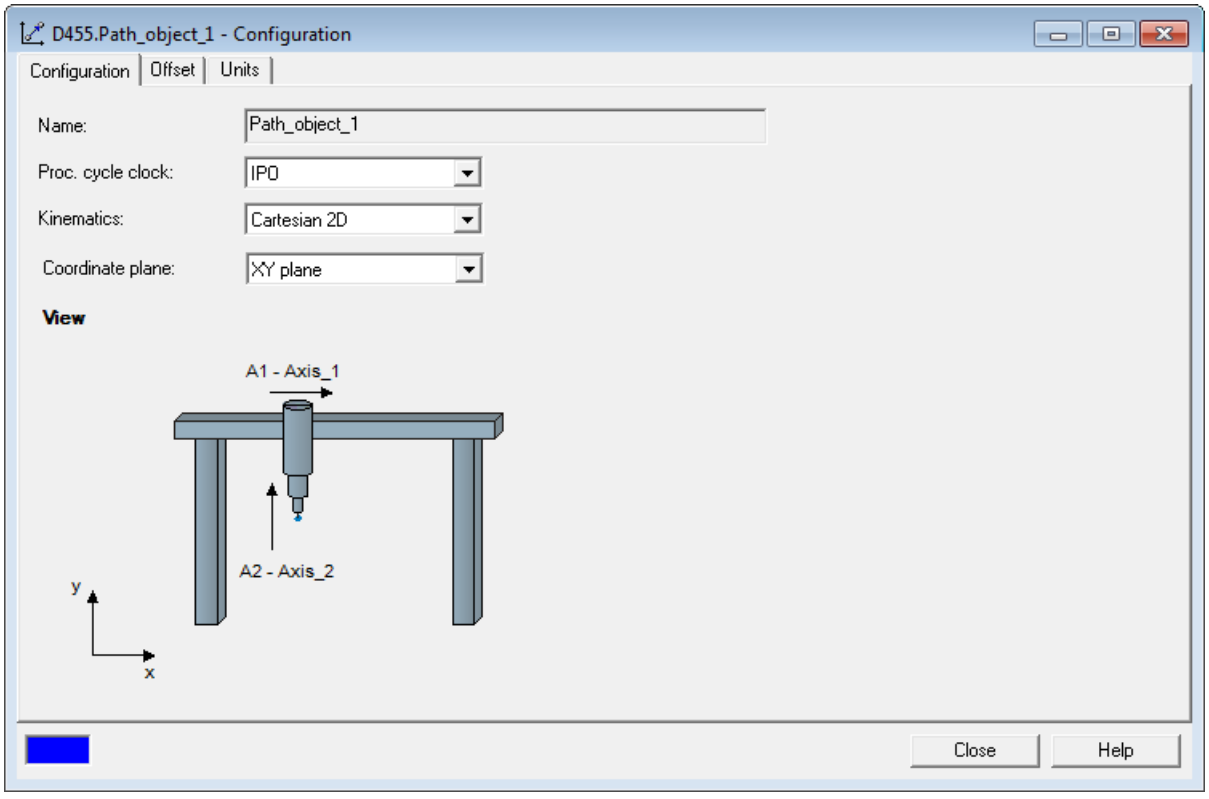


Figure 4-586 Path object: Configuration - Configuration tab

In this window, you can define the following parameters:

Field/button	Meaning/instruction
Processing cycle clock	You define here whether the path object is processed in the IPO or in the IPO_2. All TOs (path object, path axes, positioning axis for path-synchronous motion) involved in a path interpolation grouping must be assigned to the same interpolation cycle! (`Execution.executionlevel`)
Kinematics	This is where you set the desired kinematics. (`Kinematics.typeOfKinematics`)
Coordinate plane	This is where you set the coordinate plane. This setting is not available for all kinematics. (`config2D`)

Offset and rotation on the path object

On the **Offset** tab, you define the translation or rotation of the kinematic zero point relative to the BCS zero point.

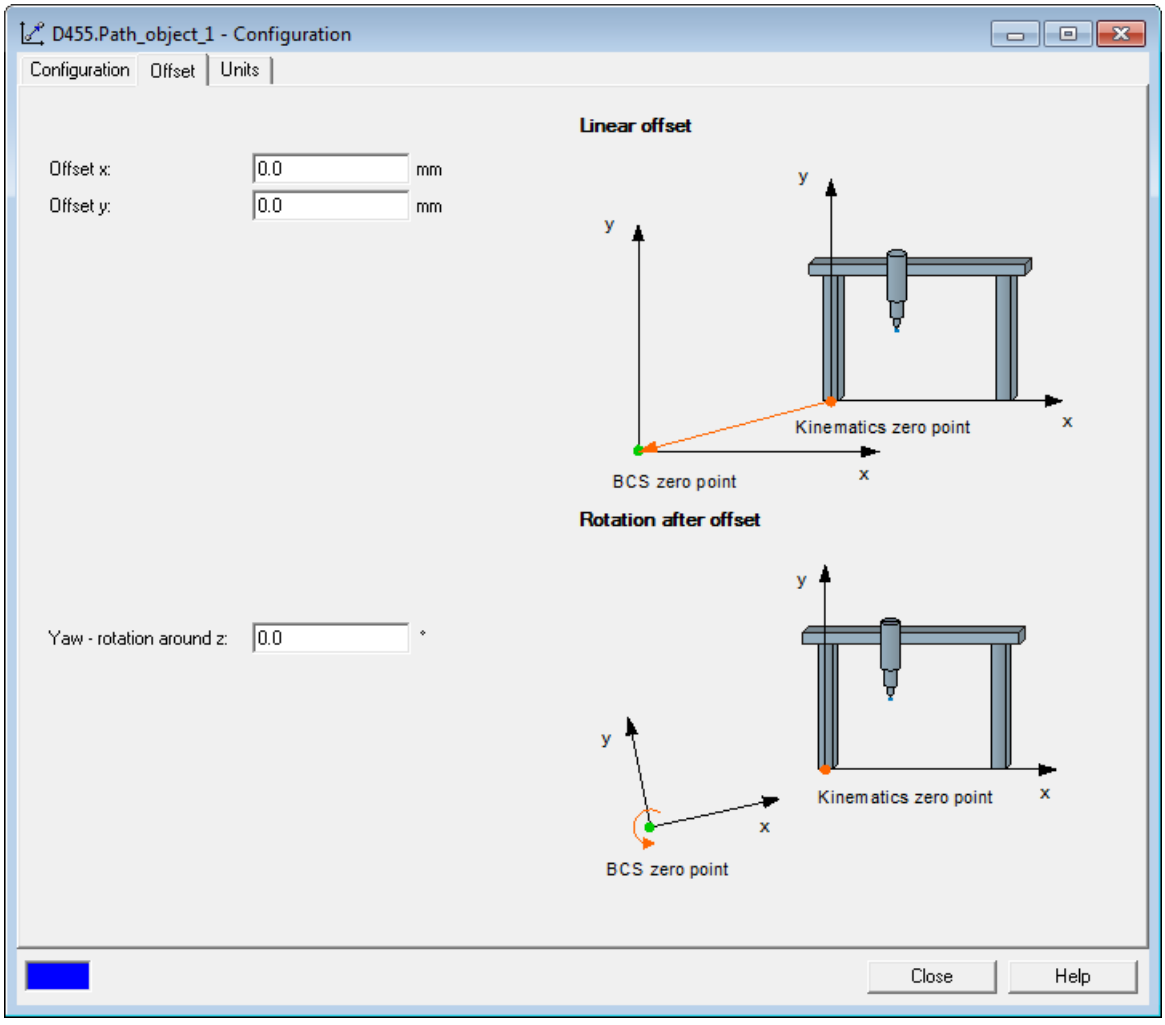


Figure 4-587 Path object: Configuration - Offset tab

In this window, you can define the following parameters:

Field/button	Meaning/instruction
Offset x Offset y Offset z	Here, you set the offset of the kinematic zero point. Depending on the kinematic type and/or set coordinate level, only certain fields exist. In this example, it is 2D kinematics or coordinate level XY. For this reason, the offset can only be set in the x and y direction. (`basicOffset.x`, `basicOffset.y`, `basicOffset.z`)
roll - rotation about x pitch - rotation about y yaw - rotation about z	Here, you set the rotation of the kinematic zero point. Depending on the kinematic type and/or set coordinate level, only certain fields exist. In this example, only rotation about the z axis is possible. (`basicOffset.roll`, `basicOffset.pitch`, `basicOffset.yaw`)

Units used for the path object

On the **Units** tab, you define the units to be used on the path object.

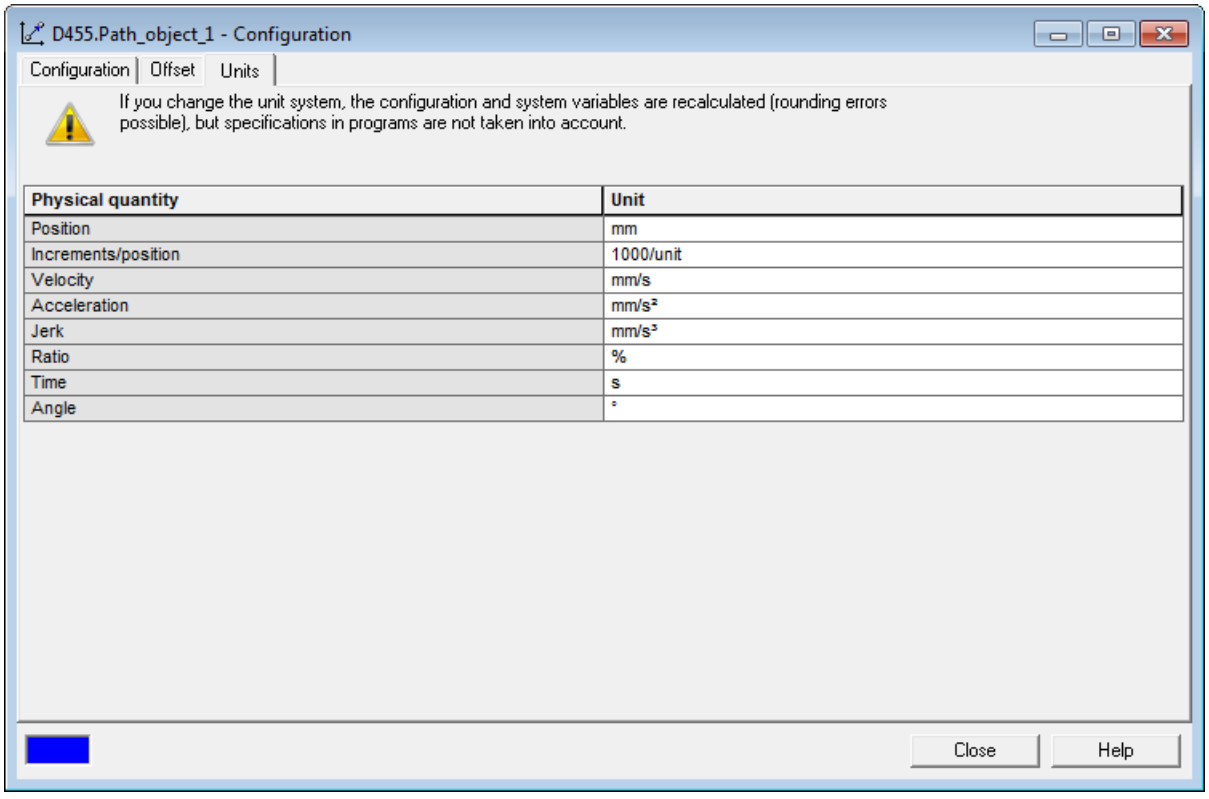


Figure 4-588 Path object: Configuration - Units tab

Further information

For an overview of functions, refer to Overview of Path Interpolation (Page 2470).

For a description of functions, refer to Basics of Path Interpolation (Page 2474).

The meaning of the configuration data and the permissible value ranges can be found in the SIMOTION reference lists.

4.6.4.7 Defining limits

- In the project navigator, double-click **Limits** under the object.

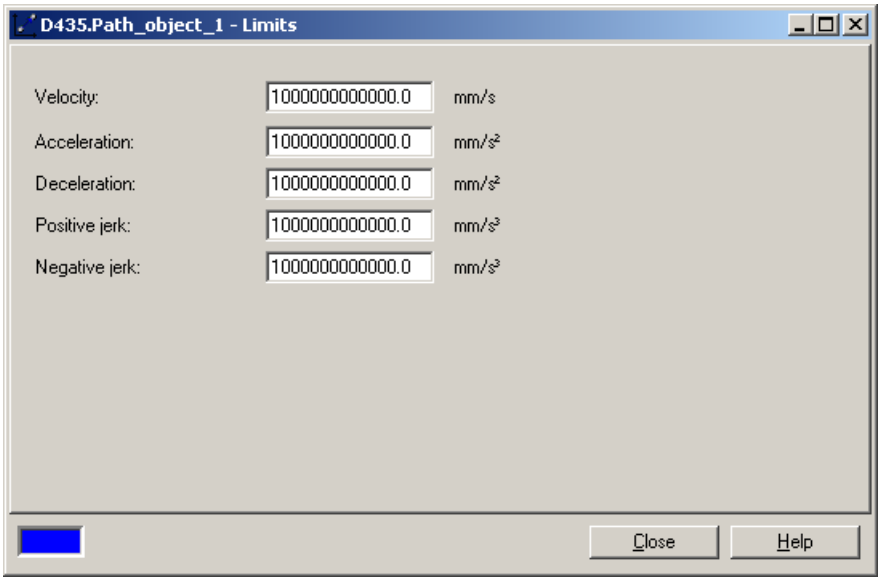


Figure 4-589 Limits on the path object

In this window, you specify the maximum dynamic path limit values:

Field/button	Meaning/instruction
Velocity	Enter the maximum velocity here. (`limitsOfPathDynamics.velocity`)
Acceleration	Enter the maximum acceleration here. (`limitsOfPathDynamics.positiveAccel`)
Deceleration	Enter the maximum deceleration here. (`limitsOfPathDynamics.negativeAccel`)
Positive jerk	Here, you enter the maximum jerk during acceleration build-up / deceleration reduction. (`limitsOfPathDynamics.positiveJerk`)
Negative jerk	Here, you enter the maximum jerk during acceleration reduction / deceleration build-up. (`limitsOfPathDynamics.negativeJerk`)

For more information, refer to Path dynamics (Page 2491).

The meaning of the configuration data and the permissible value ranges can be found in the SIMOTION reference lists.

4.6.4.8 Interconnecting a path object

- In the project navigator, double-click **Interconnections** under the object.

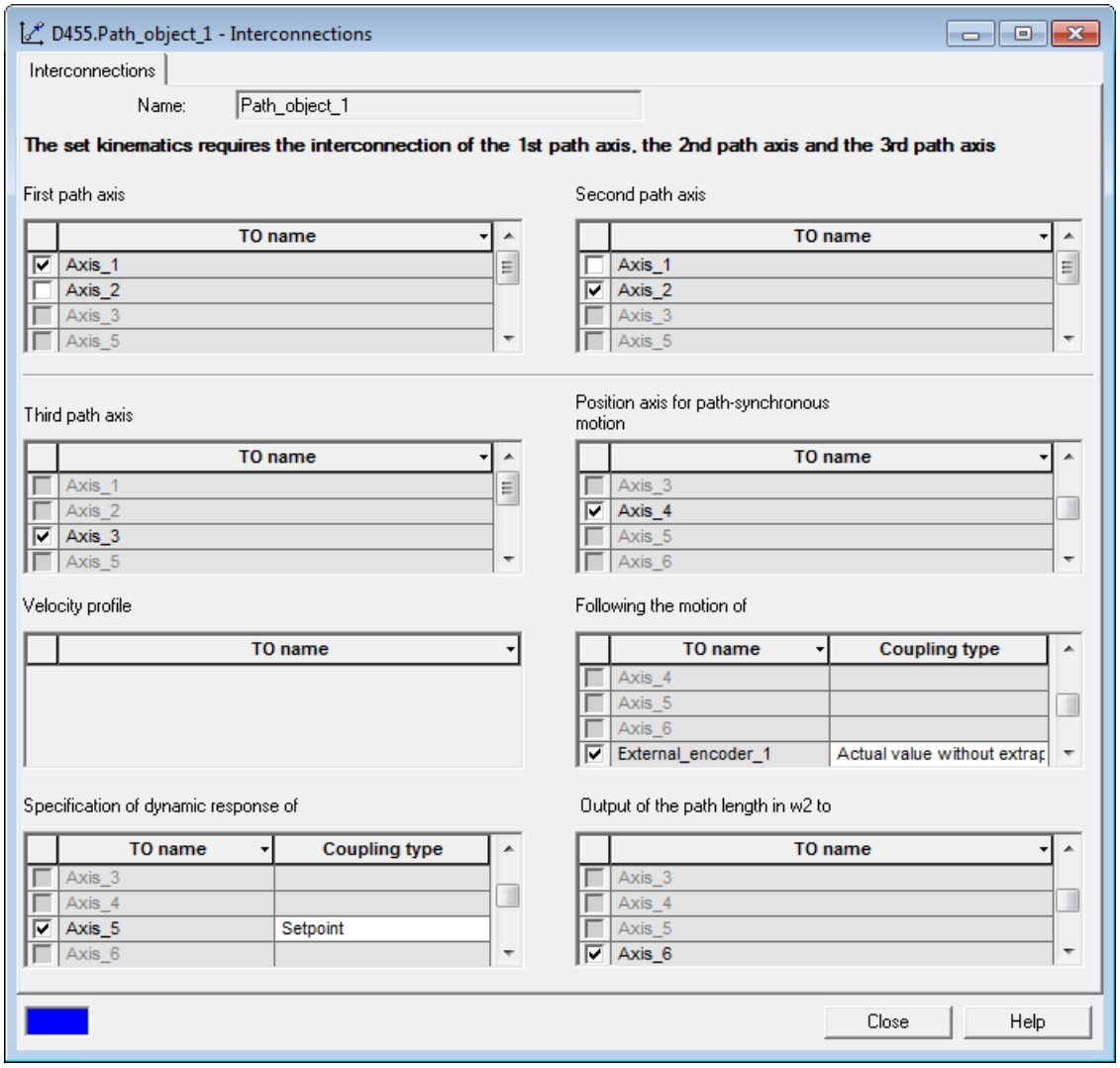


Figure 4-590 Interconnecting a path object

In this window, you interconnect the outputs of the path object with the path axes or with a position axis. (The objects must have already been created.)

Place a check mark beside the required objects.

A path object must be interconnected with at least two path axes.

The following connectors of the path object must be interconnected:

- **Path axis 1**: With a path axis
- **Path axis 2**: With a path axis

The following connectors of the path object can be interconnected:

- **Path axis 3**: With a path axis
- **Axis for path-synchronous motion**: With a position axis, synchronous axis, or path axis

- **Velocity profile**: With a cam
- **Output of the path length in w2** (as of V4.4): with a positioning axis

For more information, refer to Interconnection, interconnection rules (Page 2556).

Following the motion of

As from V4.2, an axis or an external encoder can be interconnected for the motion sequence. For more information, refer to Object coordinate systems and motion sequences on the path object (Page 2545).

Specification of the dynamics

As of V4.3, the path dynamics can be specified via DynamicsIn. For more information, refer to Preset path dynamics (Page 2491).

4.6.4.9 Path control panel

The path control panel is used to commission kinematics. It is therefore possible to control and monitor path objects without a user program. Applications include the following:

- Enabling and disabling all connected axes
- Function test of path objects and the connected axes
- Commissioning and setting up the kinematics
- Cartesian traversing in the BCS or in an OCS
- Traversing in machine axes
- Monitoring axis positions and Cartesian coordinates

control priority

The control priority for the path object and the connected axes can either be with the path control panel or with the user program. If the control priority is with the path control panel, motion commands on the path object and the axes involved are refused.

⚠ **WARNING**
Danger to life from unexpected machine movement Failure to observe the safety instructions can result in personal injury and material damage. • Note the safety instructions in the **Assume Control Priority** SCOUT dialog box.

Requirements

- An online connection to the SIMOTION device must be established.
- The current configurations of the axes and the path object must be on the target device. If necessary, download the project or upload the configuration data for alignment purposes (Target system > Load > Load configuration data to PG).

You can find additional information in the SCOUT online help (index: Path control panel).

4.6.4.10 Configuring kinematic adaptation in the expert list

All configuration data and system variables for the **Path object** technology object can be displayed and changed in the expert list.

Here, you define the kinematic type and adapt it for your requirements (see Kinematic adaptation (Page 2507)).

For additional information, see **Motion Control Basic Functions** Function Manual, "Expert list".

4.6.4.11 Configuring path monitoring

Path monitoring can be configured on the axis.

In the project navigator, double-click **Monitoring** under the axis object.

Set the required parameters on the **Path motion** or **Synchronous path motion** tab.

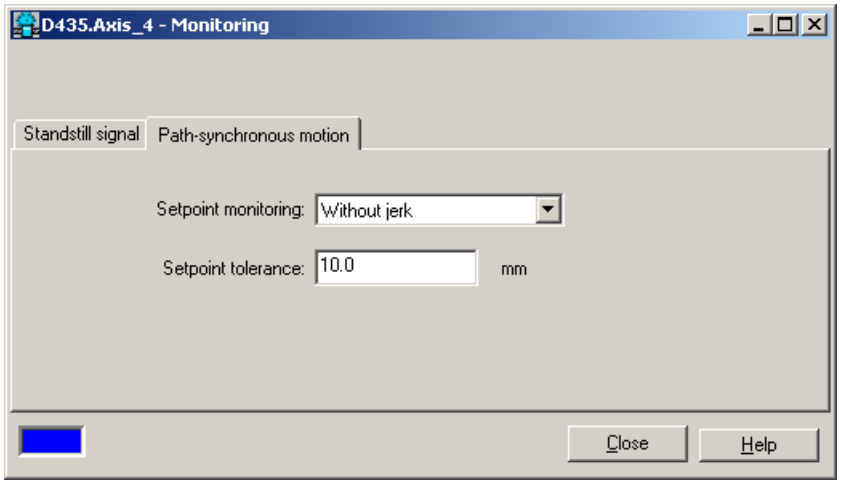


Figure 4-591 Monitoring on the axis with path interpolation

Path motions

Field/button	Meaning/instruction
Setpoint monitoring	Here, you activate the setpoint monitoring for the path axis. (pathAxisPosTolerance.enableCommandValue)
Setpoint tolerance	Here, you specify the permissible deviation of the setpoint on the axis calculated by the path object for the path axis, taking into account the limits of the executable setpoint. (pathAxisPosTolerance.commandValueTolerance) An alarm will be initiated if exceeded.

Path-synchronous axis - monitoring functions - path-synchronous movement

Field/button	Meaning/instruction
Setpoint monitoring	Here, you activate the setpoint monitoring for the positioning axis. (pathSyncAxisPosTolerance.enableCommandValue)
Setpoint tolerance	Here, you specify the permissible deviation of the setpoint on the axis calculated by the path object for the positioning axis, taking into account the limits of the executable set-point. (pathSyncAxisPosTolerance.commandValueTolerance) An alarm will be initiated if exceeded.

For additional information, see Display and monitoring options on the axis (Page 2503).

4.6.4.12 Calibrate path object

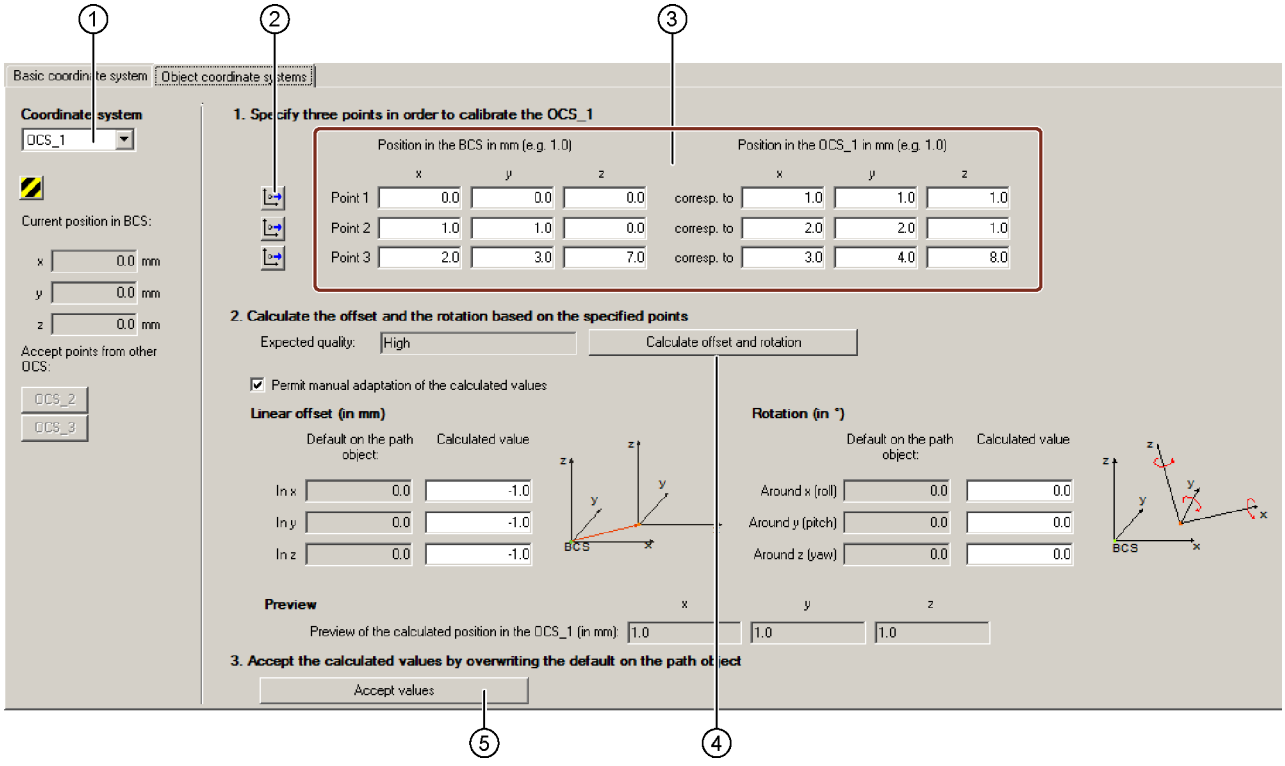


Figure 4-592 Calibration

You calibrate the path object in this window by re-aligning the coordinate systems. The basic coordinate system can be aligned with the kinematic zero point or an object coordinate system with the basic coordinate system.

The following five steps are required for the example:

① Select an OCS

② If required, accept the current position of the kinematics

③ Specify the actual positions and the position setpoints of the kinematics

④ Start the calculation of the offset and the rotation

⑤ If required, accept the calculated values as the new values for the kinematics

Step 1 is omitted when calibrating the basic coordinate system.

Calibrating the basic coordinate system (BCS)

You calibrate the BCS to the kinematic zero point on the **Basic coordinate system** tab. Depending on the dimensions of the selected kinematics, you must specify two or three points in the BCS and two or three corresponding new points from which the offset and the rotation then result. Input fields for at least two coordinates are available for each point. With 3D kinematics, there is also an additional input field for the third coordinate. Individual points in the BCS can also be taken over directly from the current position of the path object using the button to the left of the input fields.

After entering valid coordinates, a button appears to calculate the offset and rotation for the BCS. The calculated values are displayed and can be transferred to the offline data management via a further button for the path object. If the checkbox for manual adaptation is activated, the calculated values can be subsequently edited.

You can open the path control panel via the black and yellow button. This button is only available when the controller is in online mode.

Calibrating the object coordinate system (OCS)

You calibrate the OCS_1, OCS_2 and OCS_3 object coordinate systems on the **Object coordinate systems** tab. The respective OCS is selected in the area on the left. The calibration is similar to that of the BCS to the kinematics zero point. The difference is that the inputs refer to reference positions in the BCS and positions in the respective OCS.

It is also possible to take over the points of a previously calibrated OCS for another OCS. The appropriate buttons are on the left.

4.6.4.13 Path interpolation - context menu

You can select the following functions:

Function	Meaning/Description
Configuration	This function opens the configuration for the path object selected in the project navigator. Define the processing cycle clock in this window.
Expert list	This function opens the expert list for the path object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list.
Default	This function opens the defaults for the path object selected in the project navigator. You define the substitute values for the object coordinate systems, the dynamic response, and the path in this window.
Calibration	With this function, you open a window for calibrating the path object. In this window you can calibrate the values for the BCS or the three OCS.

Function	Meaning/Description
Limits	This function opens the limits for the path object selected in the project navigator. You define the dynamic limits for the path object in this window.
Interconnections	This function opens the interconnections for the path object selected in the project navigator. You can see the inputs of the axes in this window.
Expert	This function opens the submenu for the expert settings.
• Import object	Use **Import object** to open a window for the XML import. You can define the parameters for the XML import in this window.
• Save project and export object	Use **Save project and export object** to open a window for the XML export. You can define the parameters for the XML export in this window.

4.6.5 Sample Project for the Path Interpolation

4.6.5.1 Overview of the example

To illustrate how the path interpolation is configured, the following sections describe a sample project step-by-step.

The following descriptions assume that the project with the SIMOTION device and the drives have already been created in the HW Config.

In this example, the following 2D gantry is created:

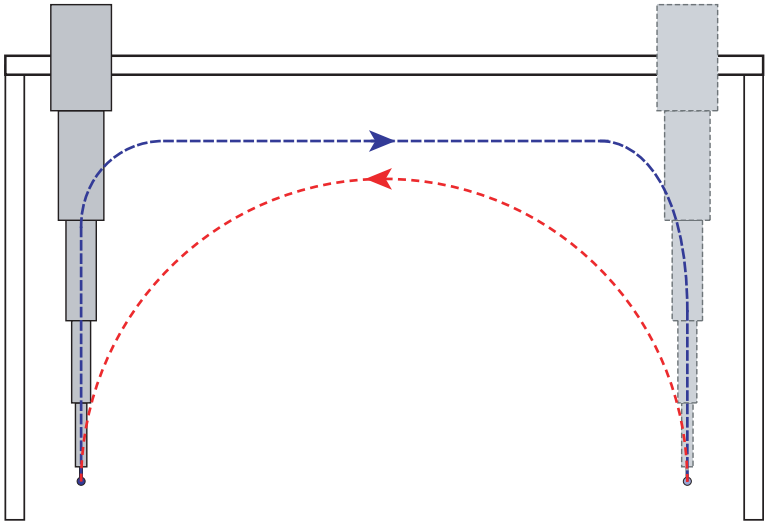


Figure 4-593 Overview

This 2D gantry comprises the following axes:

- Vertical axis: 1400 cm traversing length, axis zero point at the lowest position
- Horizontal axis: 2000 cm traversing length, axis zero point at the left-hand side

This means the zero point of the path object is at the lower left.

How to create and use the 2D gantry:

- Select technology package (Page 2576)
- Create axes. (Page 2577)
- Creating a path object (Page 2579)
- Defining the kinematics (Page 2580)
- Interconnecting a path object (Page 2580)
- Programming a path in MCC (Page 2584)
- Checking a motion with trace (Page 2602)

To show how a path-synchronous axis (Page 2603) operates, a gripper will also be created at the end of the example.

4.6.5.2 Select technology package

The **PATH** and **CAM_EXT** technology packages support path interpolation.

Right-click the device in the project navigator, select **Select technology package** in the context menu and use the **PATH** technology package.

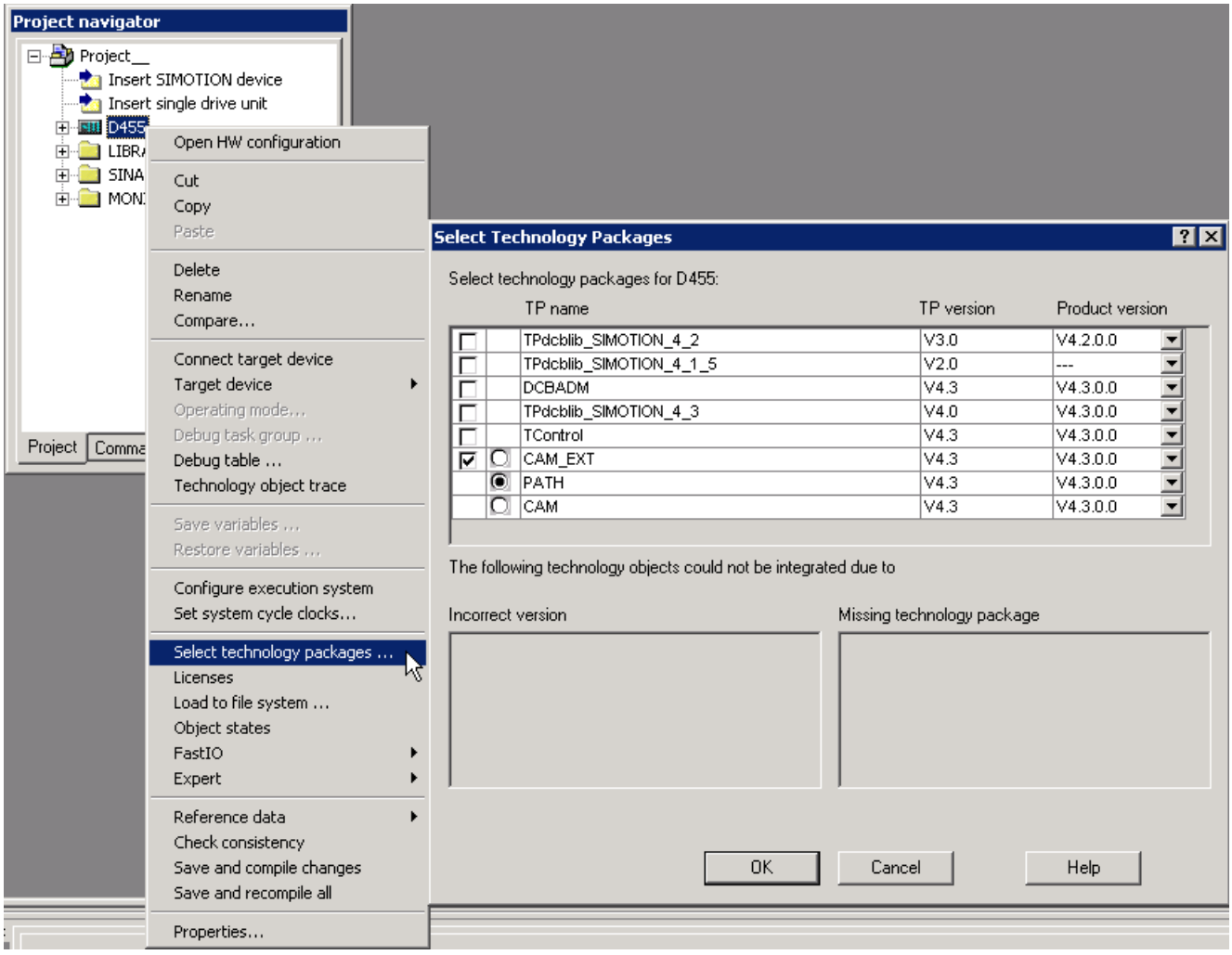


Figure 4-594 Select technology package

4.6.5.3 Create axes.

The example project requires two linear axes: **Axis_X** and **Axis_Z**. These axes are created with the **path interpolation** technology. Both axes are created as linear, virtual, non-modular axes.

Click **AXES -> Insert axis** to add an axis to the device. Name the first axis **Axis_X** and the second axis **Axis_Z**, and set up the two as follows:

1. **Path interpolation** technology selected

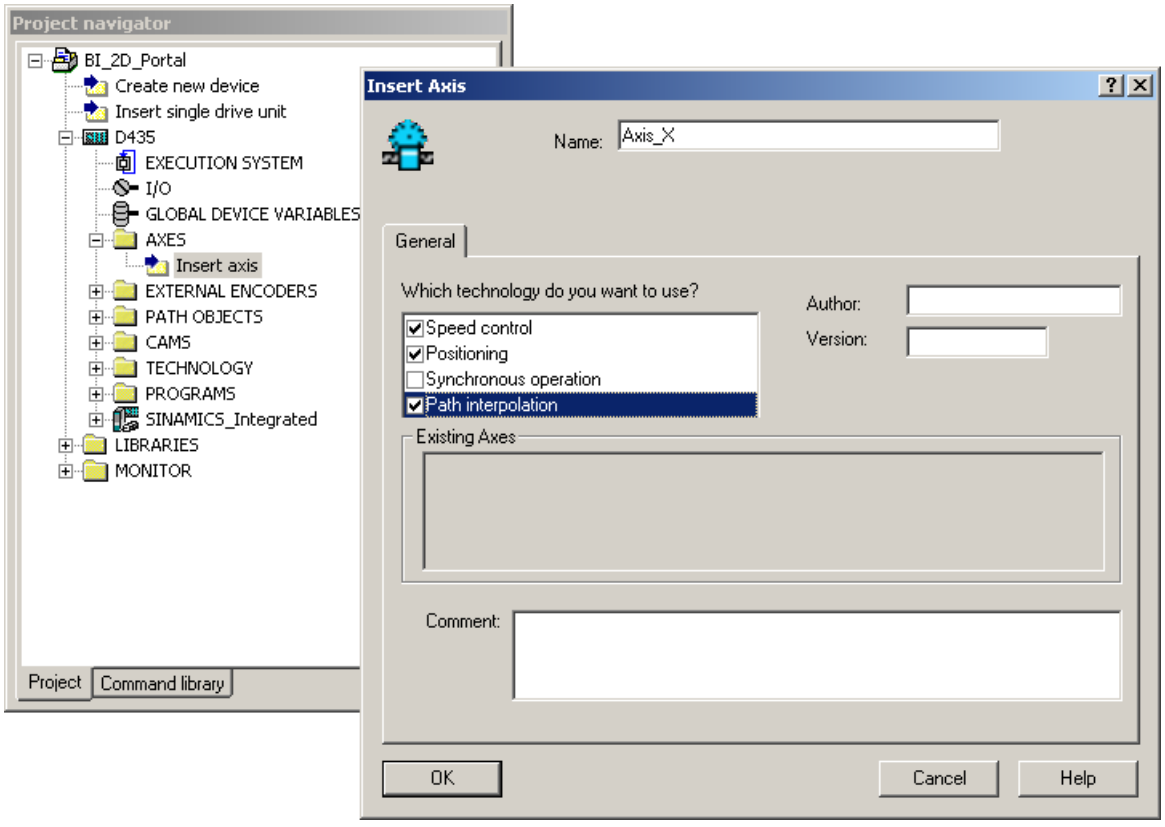


Figure 4-595 Creating an axis

2. Linear, virtual, non-modular axis
   The configuration of the two axes is as follows:

```
Configuration of this axis:
Name:
  - Axis_X
Technology:
  - Path axis
Axis type:
  - Linear axis
  - No modulo selected
Drive:
  - Axis is virtual
```

Figure 4-596 Sample summary of the configuration (Axis_X axis)

The project navigator should now look like this:

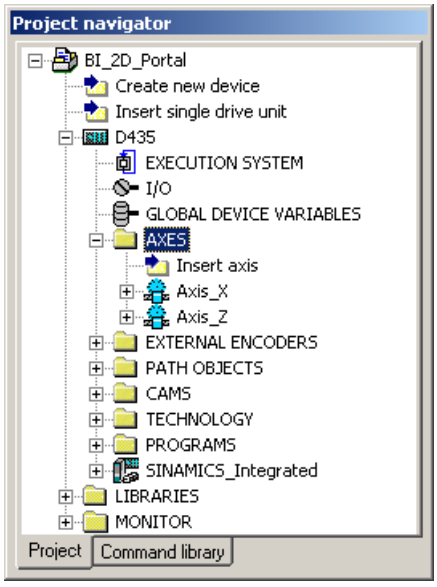


Figure 4-597 Project navigator with created axes

4.6.5.4 Creating a path object

Insert a new path object for the device under **PATH OBJECTS**. Name the path object **Portal_2D**.

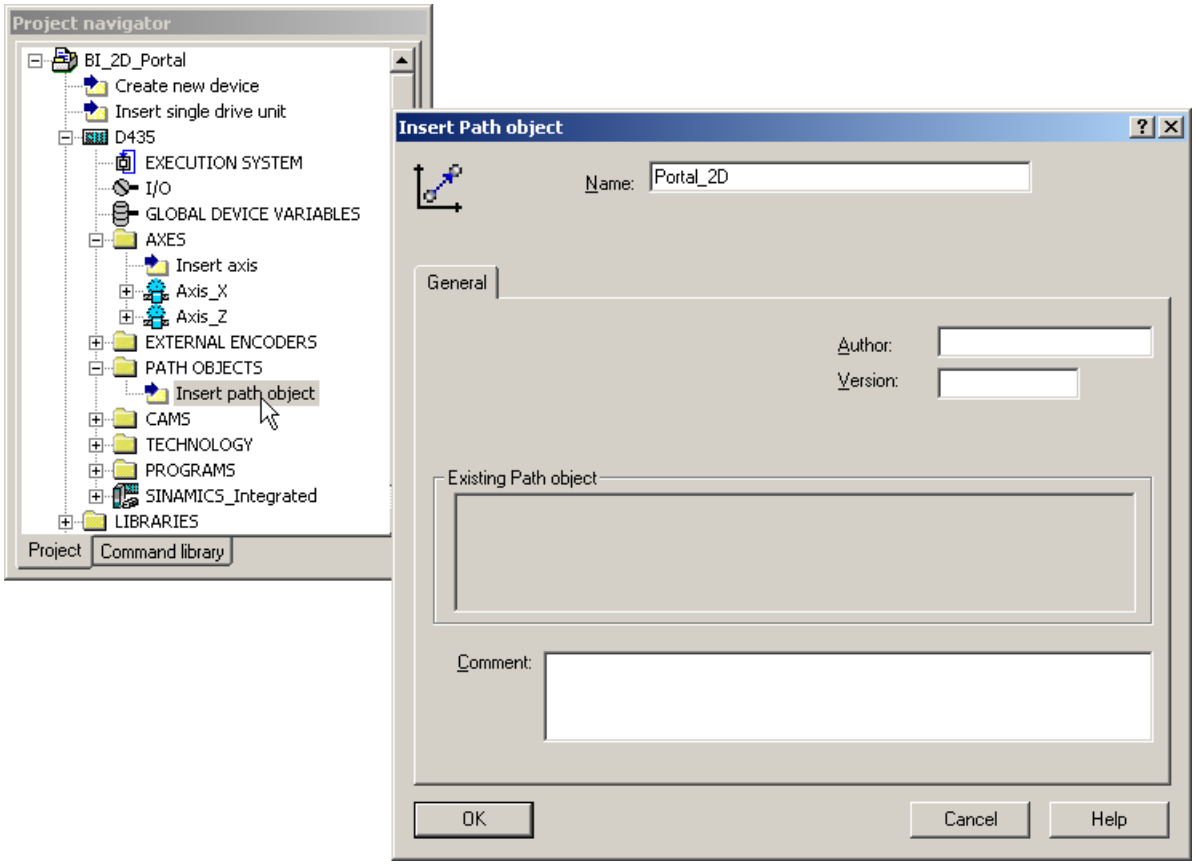


Figure 4-598 Creating a path object

4.6.5.5 Defining the kinematics

For the definition of the kinematics, the kinematics type with its mechanical data and the displacement of the coordinate system at the zero point of the base coordinate system are specified.

Open the Configuration window for the path object. In this form, youc an set the type of kinematics and the coordinate plane as follows:

- Kinematics: **Cartesian 2D**
- Coordinate plane: **ZX plane**

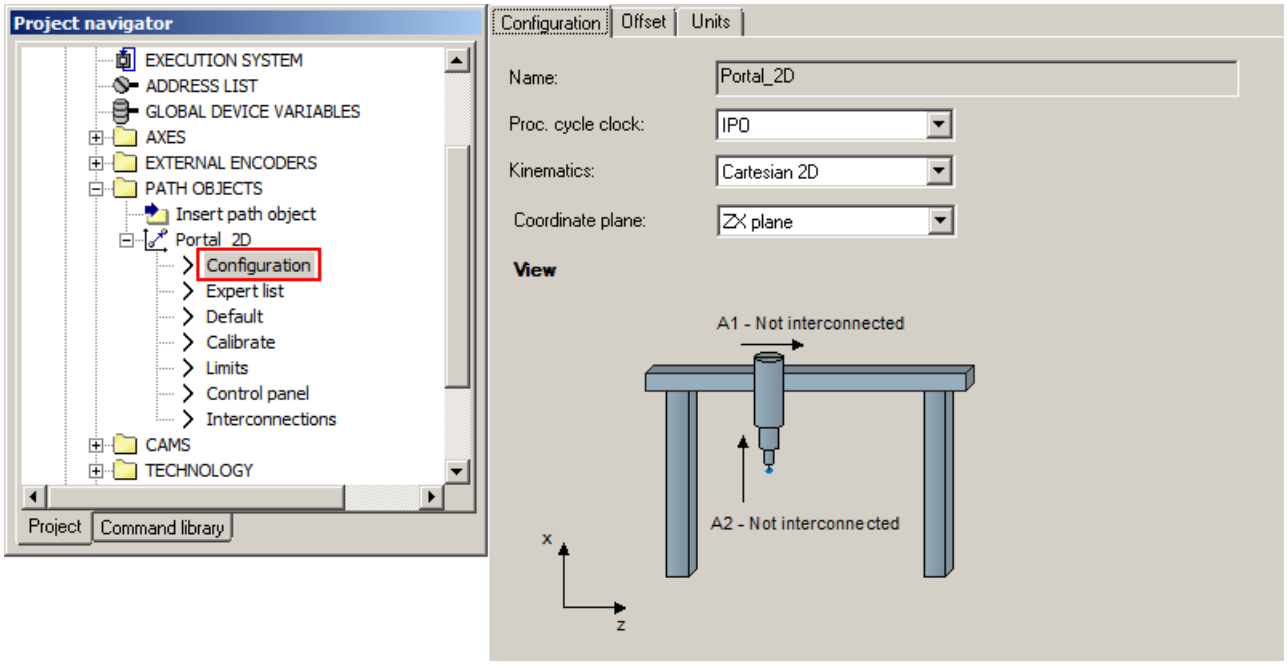


Figure 4-599 Configuration of the kinematics

4.6.5.6 Interconnecting a path object

Open the **Interconnections** window for the path object. In this screen form, assign the axes to the path object.

Because the kinematics operate in the Z-X plane, the Z-axis must be used as first path axis and the X-axis as second path axis.

Parameterize the **interconnections** of the path object as follows:

- 1. Path axis: **Axis_Z**
- 2. Path axis: **Axis_X**

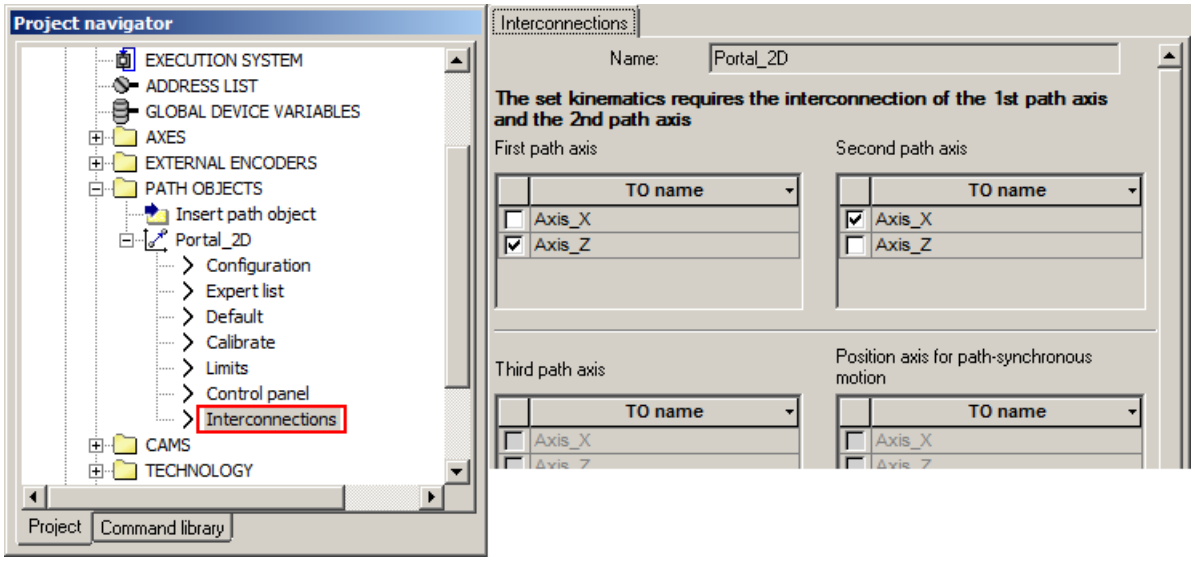


Figure 4-600 Interconnecting a path object

4.6.5.7 Setting the default settings of the path object

The settings described below must be made for the default of the path object.

How to set the defaults for the path object:

1. For the path object, click **Default**.
2. In the **Default** window, on the **Object coordinate systems** tab, you can set the object coordinate system:

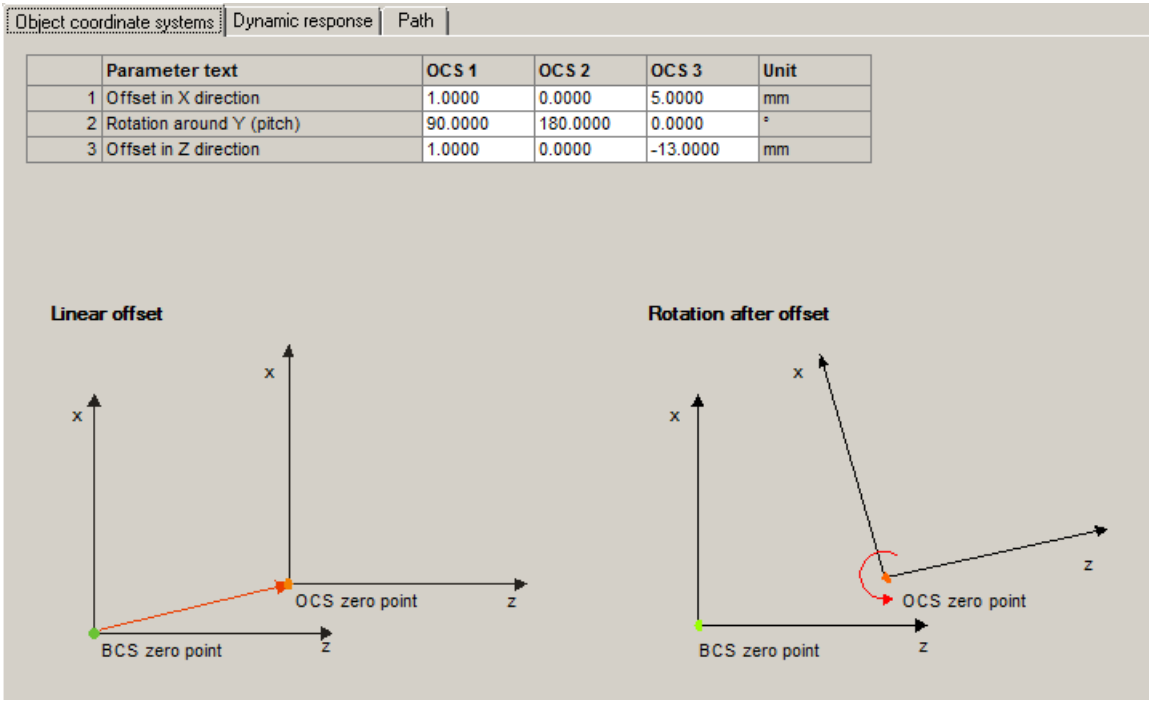


Figure 4-601 Default - Object coordinate systems

If you want to activate an object coordinate system, you must call the command `_setPathObjectOcs` (see Object coordinate system (OCS) on the path object (Page 2545)).

3. On the **Dynamic response** tab, you can set the path object, for example, as follows:

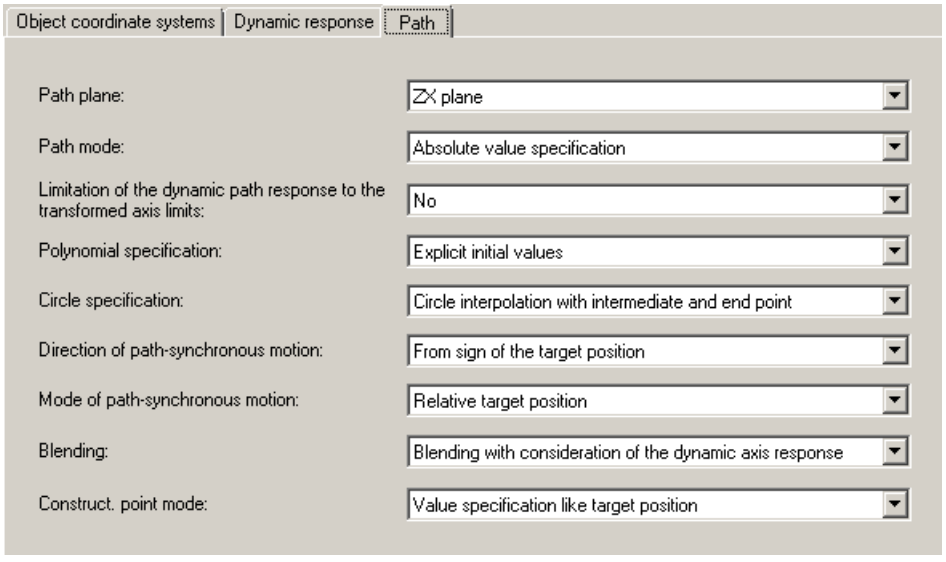


Figure 4-602 Default - dynamic response

4. Make the following settings on the **Path** tab:

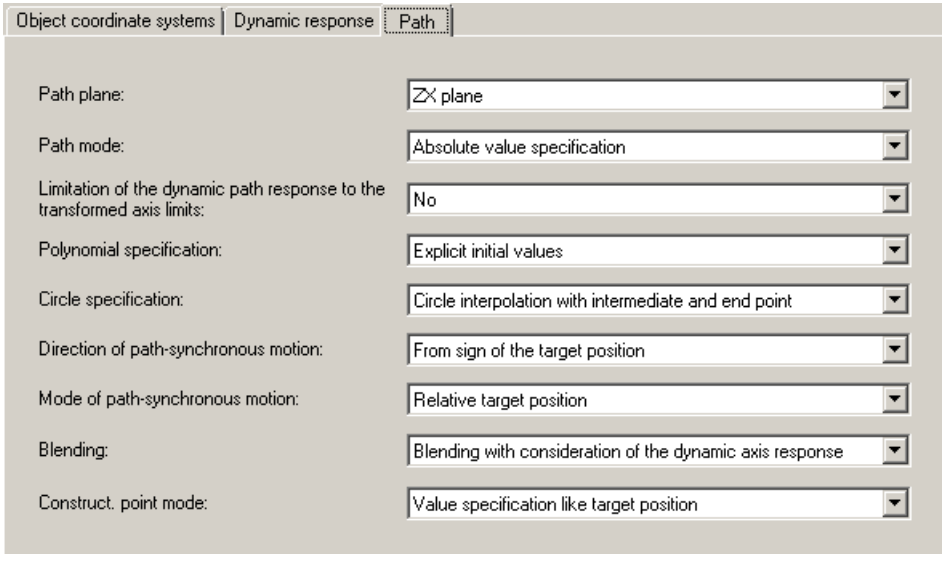


Figure 4-603 Default - path

4.6.5.8 Programming the path interpolation in MCC

Programming the travel commands in MCC

The following path should be created for this example:

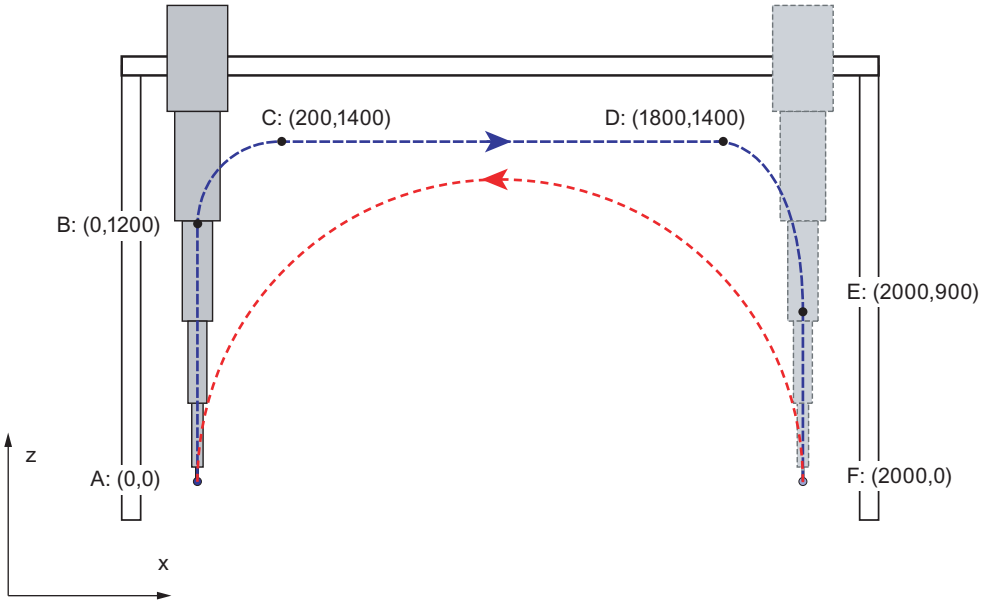


Figure 4-604 Path to be traversed

This path consists of the following path segments:

Segment	Path type	Start point (x,z)	End point (x,z)
A - B	Linear path	(0,0)	(0,1200)
B - C	Polynomial path	(0,1200)	(200,1400)
C - D	Linear path	(200,1400)	(1800,1400)
D - E	Polynomial path	(1800,1400)	(2000,900)
E - F	Linear path	(2000,900)	(2000,0)
F - A	Circular path	(2000,0)	(0,0)

Creating the program

1. In the project navigator, click **Insert MCC source file**. Name the MCC source file **MCC_Example**.

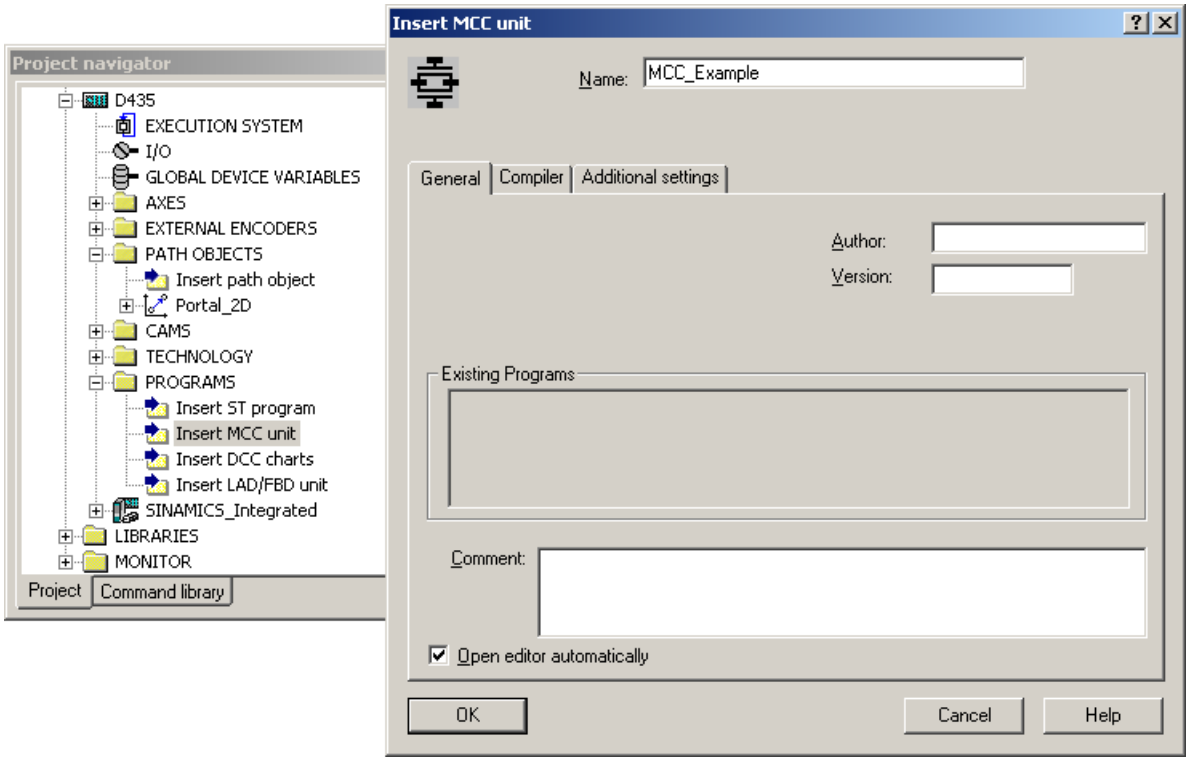


Figure 4-605 Creating an MCC source file

2. Define the following variables in the **Interface** of the MCC source file:
   - **start_move** (BOOL, true): The gantry should perform the motion when **start_move=true** is set and stop when **start_move=false**.

– **forw_back** (BOOL, false): The forw_back variable indicates whether the gantry moves forwards (true, A->F) or backwards (false, F->A).

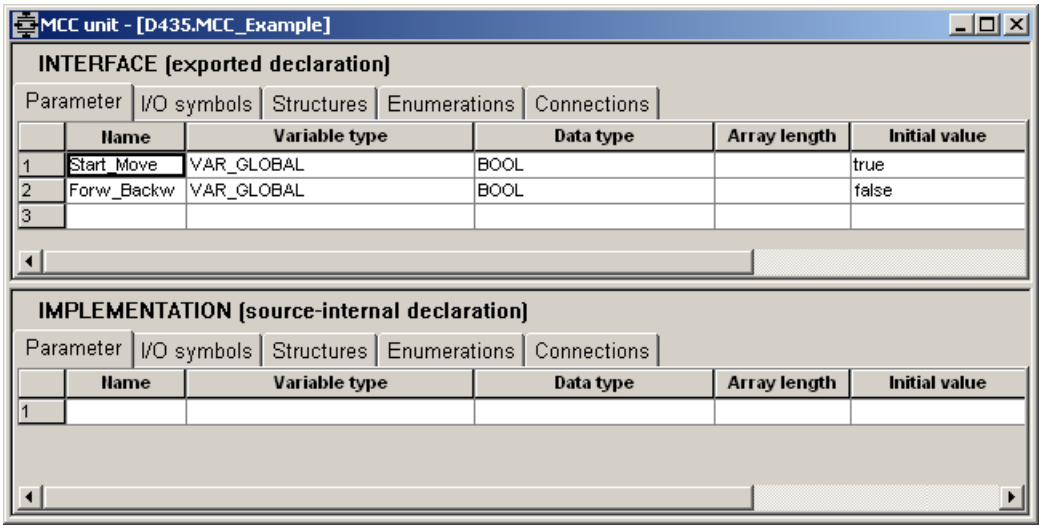


Figure 4-606 Defining variables in the MCC source file

3. In the project navigator, click **Insert MCC chart** for the new MCC source file. Name this chart **TopLoader**.

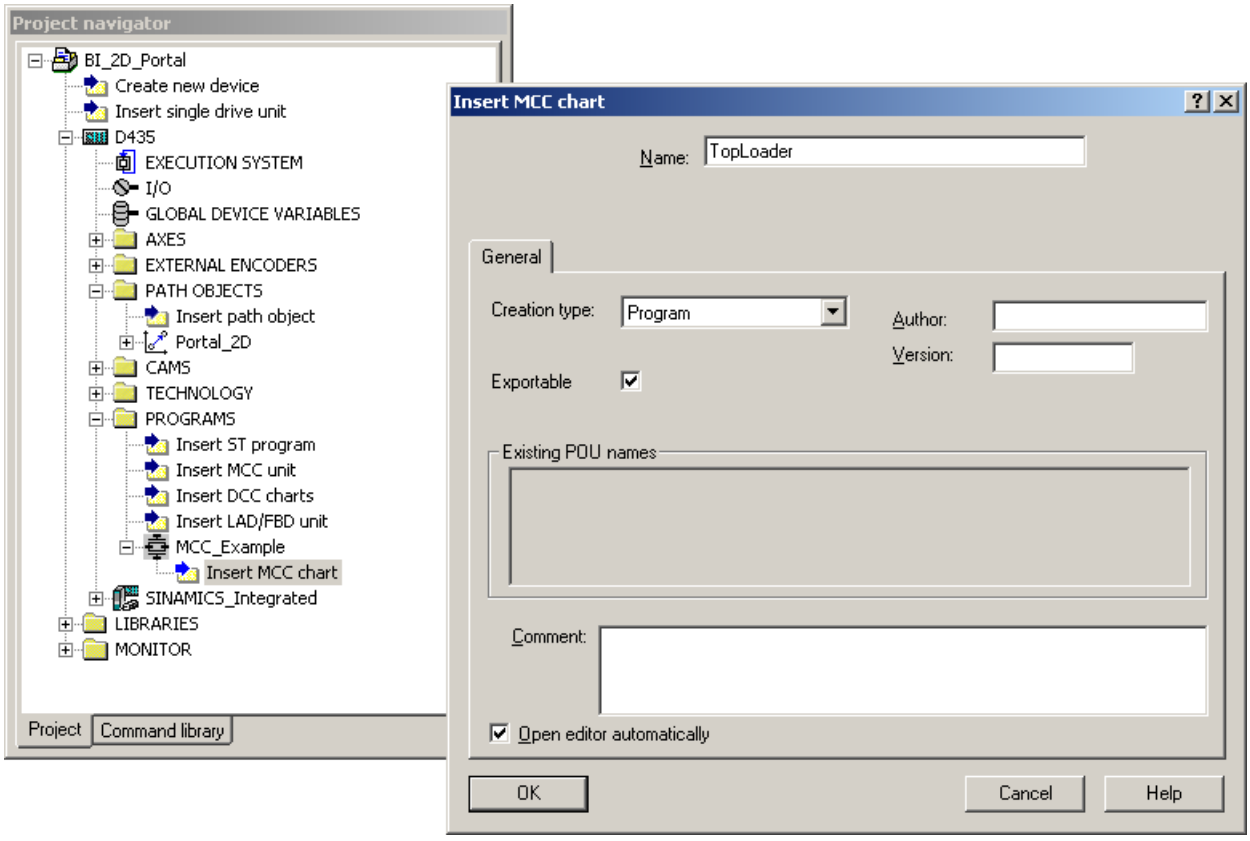
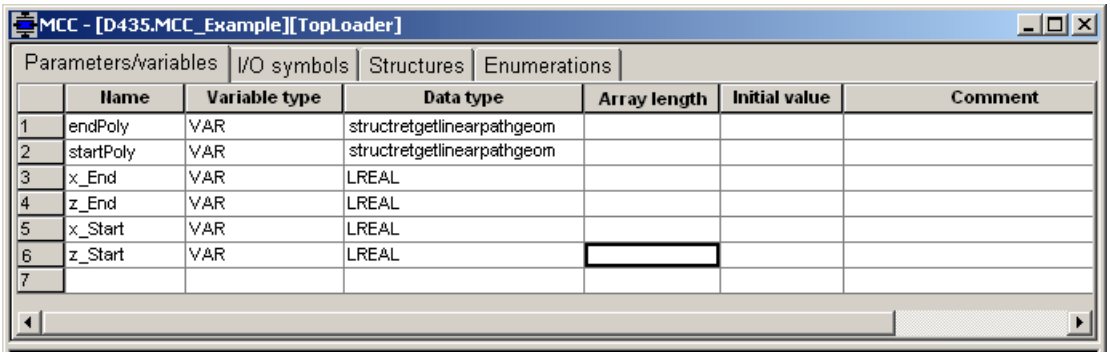


Figure 4-607 Inserting an MCC chart

4. Open the MCC chart and define the following variables.
   - **endPoly**, **startPoly**: structRetGetLinearPathGeometricData
   - **x_start**, **z_start**, **x_end**, **z_end**: LREAL

   These variables are used for calculating the data for the polynomial interpolation.



	Name	Variable type	Data type	Array length	Initial value	Comment
1	endPoly	VAR	structretgetlinearpathgeom			
2	startPoly	VAR	structretgetlinearpathgeom			
3	x_End	VAR	LREAL			
4	z_End	VAR	LREAL			
5	x_Start	VAR	LREAL			
6	z_Start	VAR	LREAL			
7						

Figure 4-608 Creating variables in the MCC source file

Programming the traversing program

Programming the traversing program

The following traversing program should be programmed:

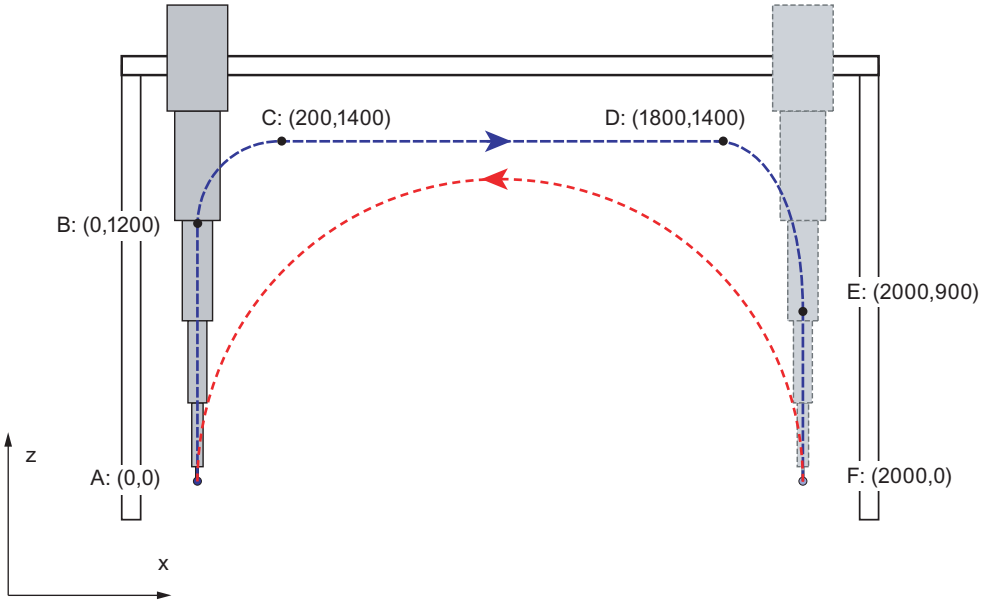


Figure 4-609 Path to be traversed

The traversing program should be performed within a **While** loop. It will be performed while **start_move** is set to **true**.

Creating a WHILE loop

For the example, a While loop is created that is run while **start_move** is set to true.

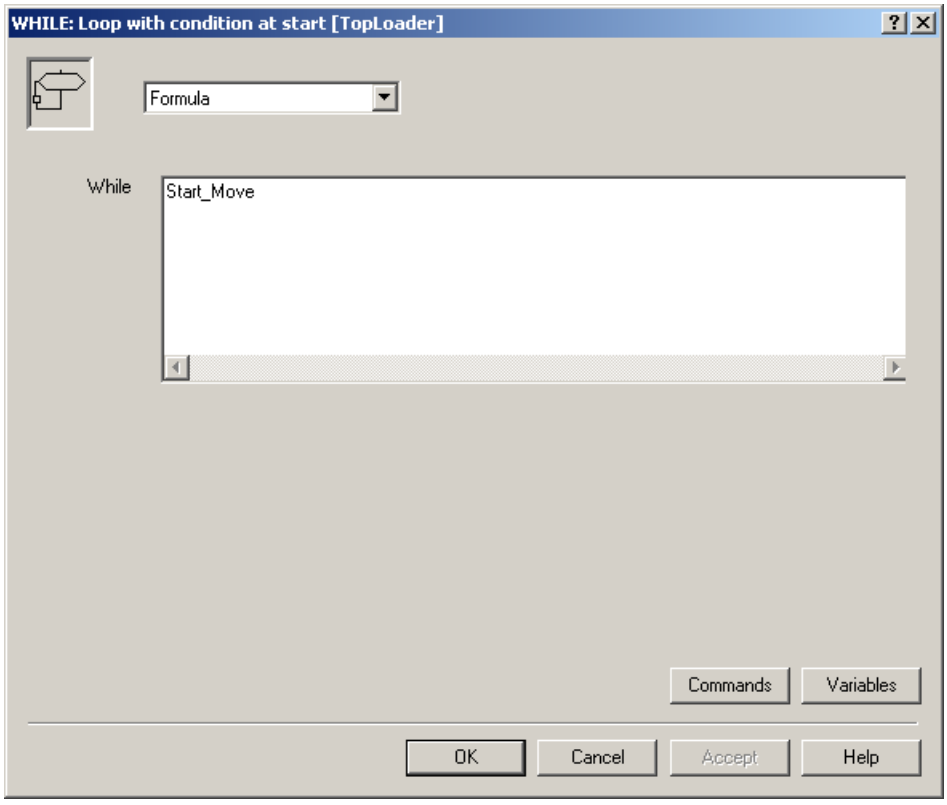


Figure 4-610 WHILE loop

Programming the A - B linear path

Before starting the forwards motion, the **forw_back** direction flag is set to **true**. This is performed using a variable assignment.

To do this, place an ST zoom-in command within the While loop and program it as follows:

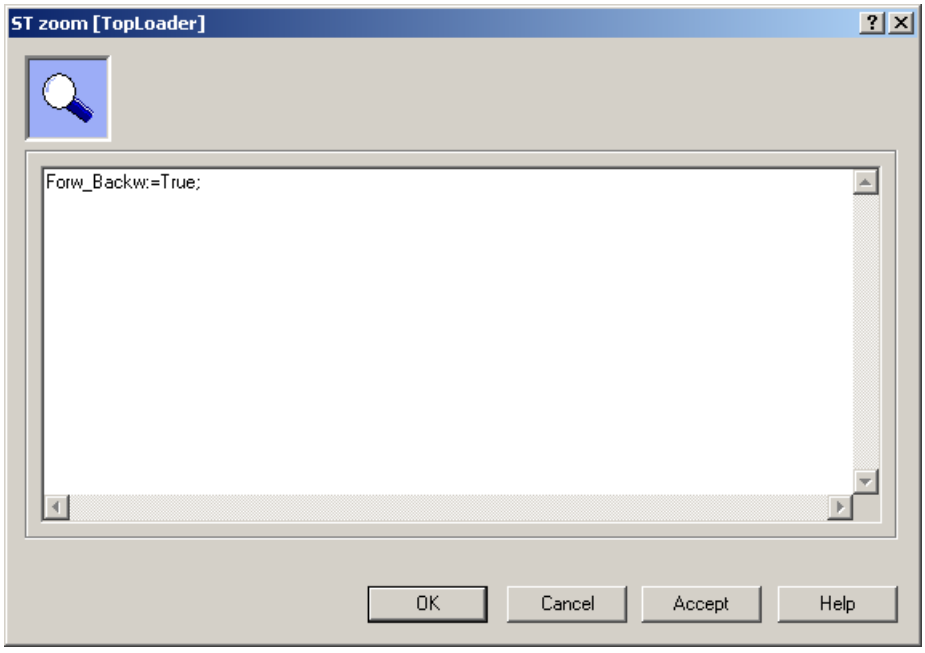


Figure 4-611 Set forw_back to true

Then add a **travel linear path** command to the While loop. Define the **A-B** linear path as follows:

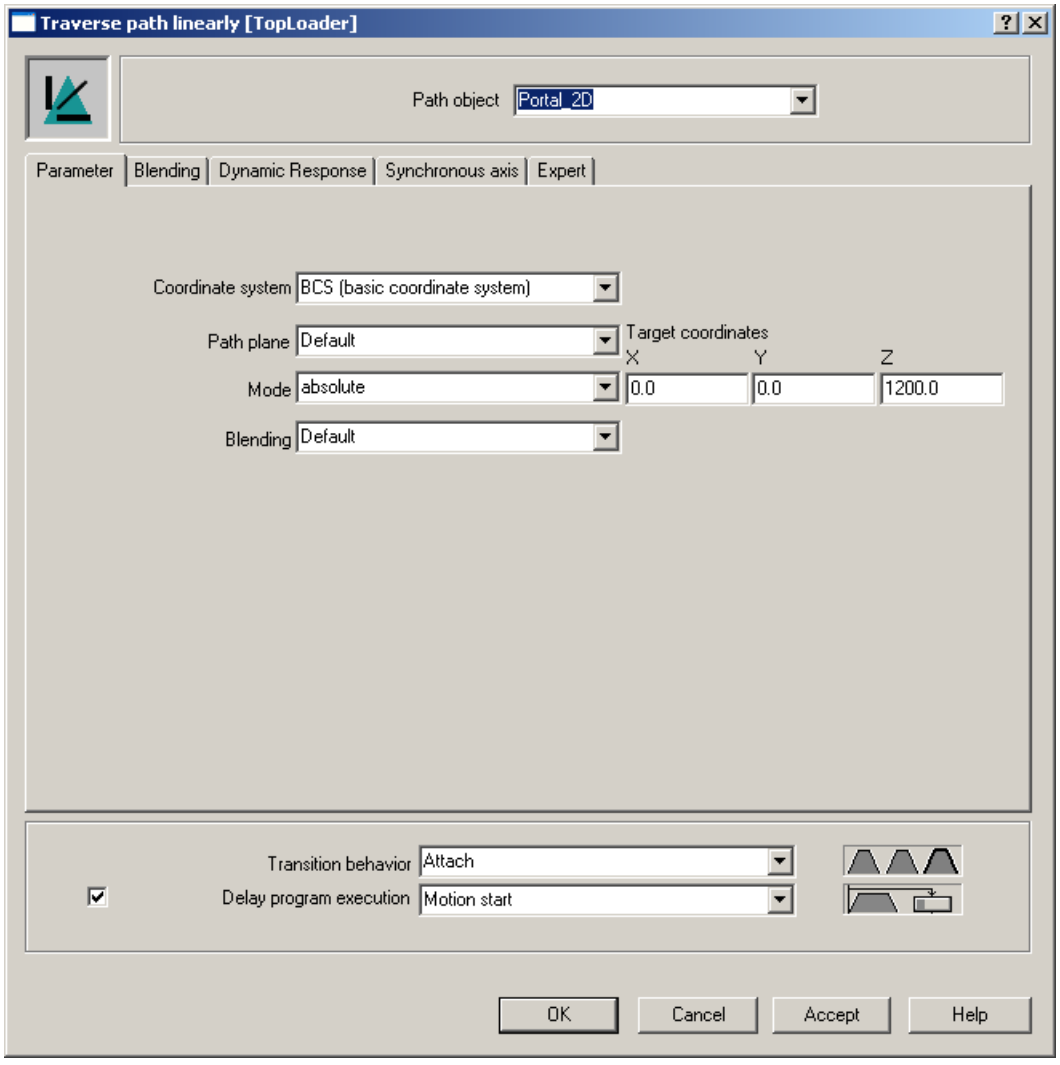


Figure 4-612 Programming the A - B linear path

Programming the B-C polynomial path

To program the **B-C** polynomial path, the geometric derivatives for the start and end points must be calculated in an ST zoom-in.

Add an ST zoom-in command for both the start point and the end point in the While loop, and program it as shown:

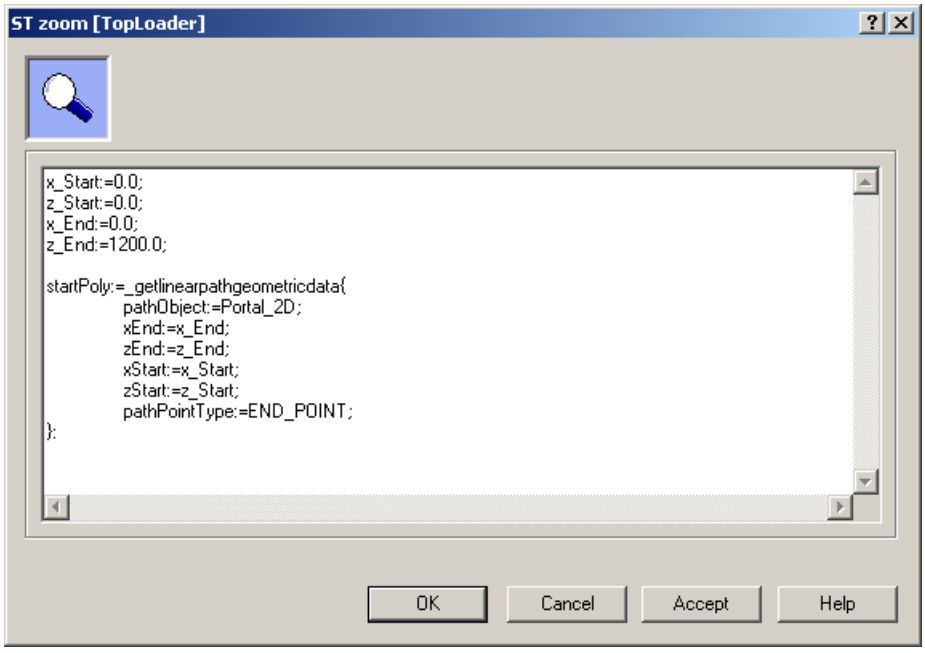


Figure 4-613 Calculating the derivatives at the start point of the first polynomial path

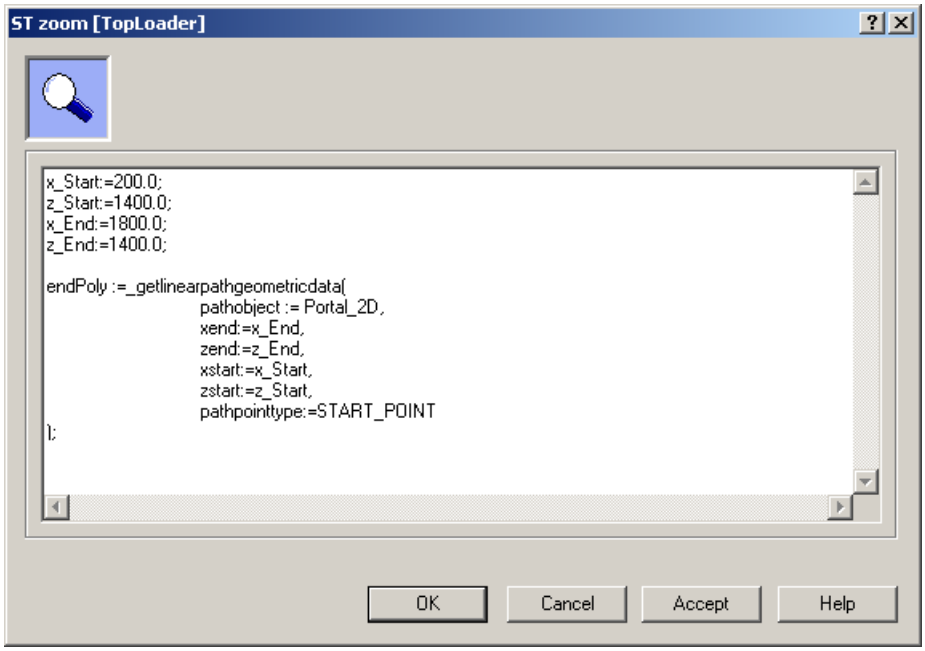


Figure 4-614 Calculating the derivatives at the end point of the first polynomial path

Now add a **travel polynomial path** command to the While loop and define the polynomial path as follows:

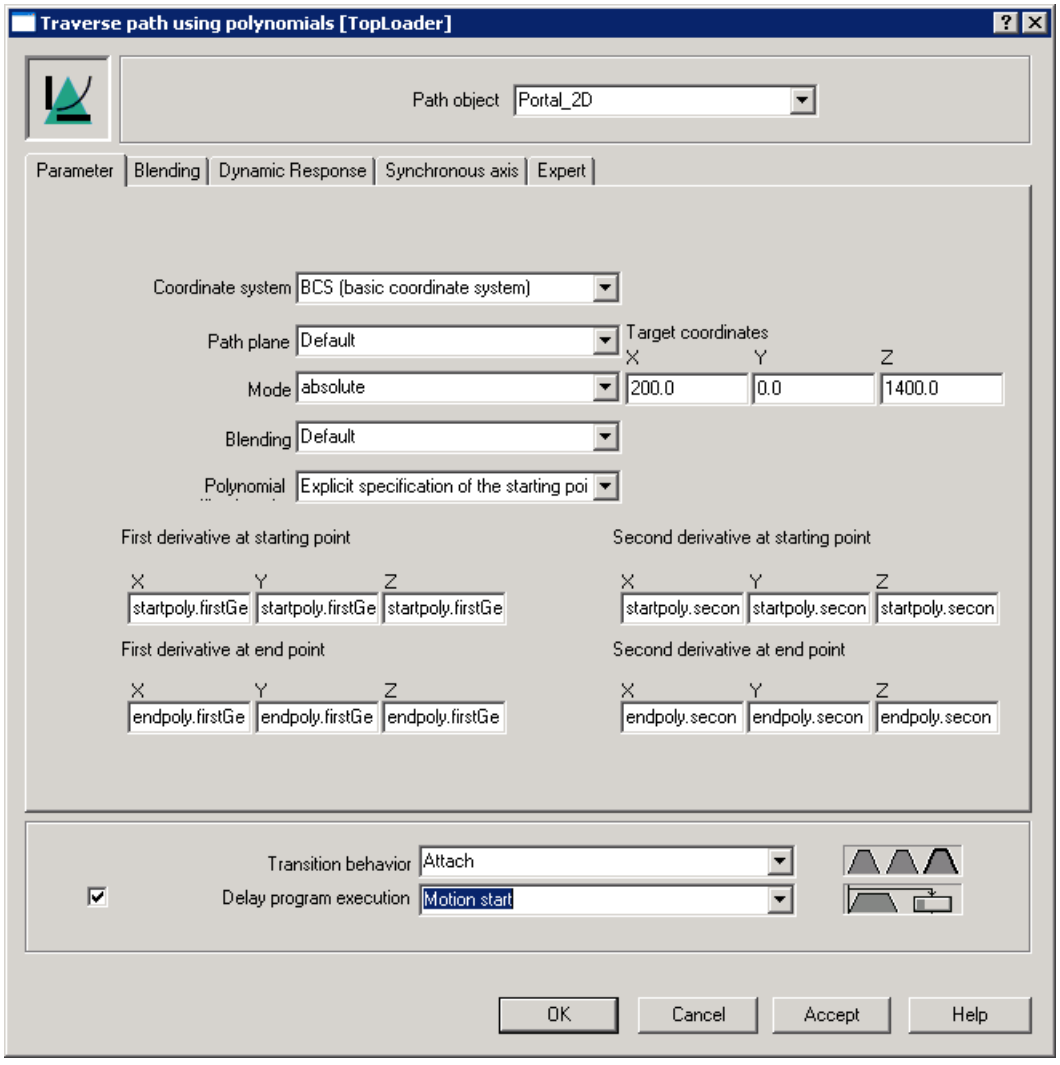


Figure 4-615 Programming the B-C polynomial path

The previously calculated derivatives are specified as follows in the command:

- First derivative at the start point: **startpoly.firstGeometricDerivative.x / .y / .z**
- Second derivative at the starting point: **startpoly.secondGeometricDerivative.x / .y / .z**
- First derivative at the end point: **endpoly.firstGeometricDerivative.x / .y / .z**
- Second derivative at the starting point: **endpoly.secondGeometricDerivative.x / .y / .z**

Note

An alternative polynomial assignment is described for the programming of the D-E polynomial path.

Programming the C-D linear path

For the **C-D** linear path, add the **travel linear path** command to the While loop and make the following settings:

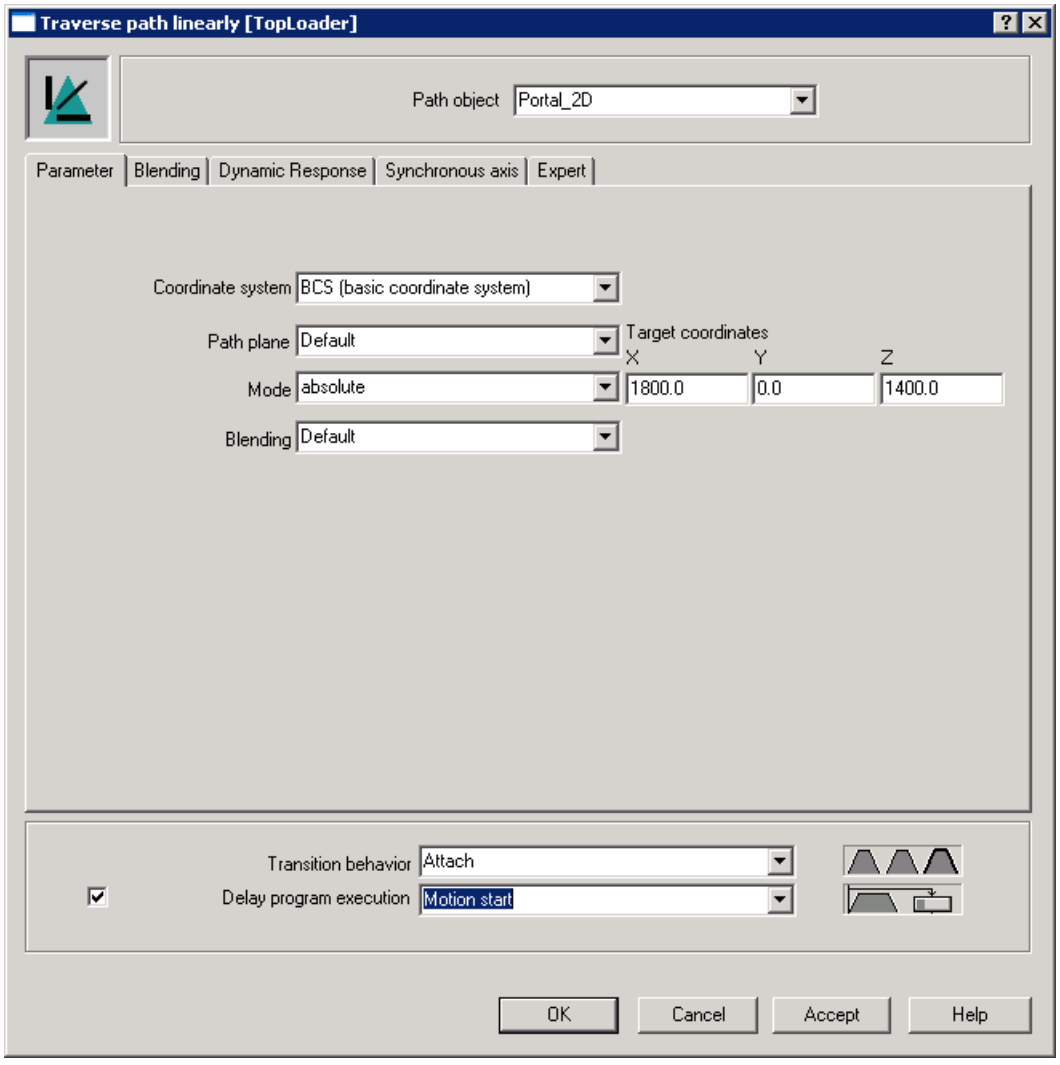


Figure 4-616 Programming the C-D linear path

Programming the D-E polynomial path

The **attach continuously** type of polynomial specification is used for the **D-E** polynomial path. The geometric deviations of the start point are calculated using the previous path segment. Only the deviations for the end point need to be calculated using an ST zoom-in.

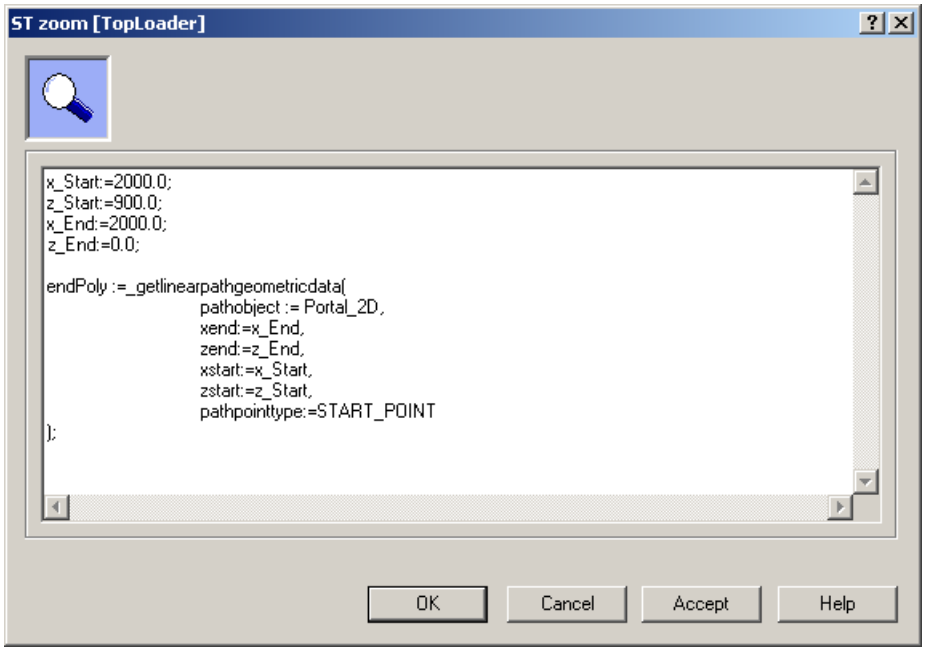


Figure 4-617 Deviations at the end point of the D-E polynomial path

Both the ST zoom-in command and the **travel polynomial path** command must be added successively to the While loop.

Parameterize the polynomial path as shown.

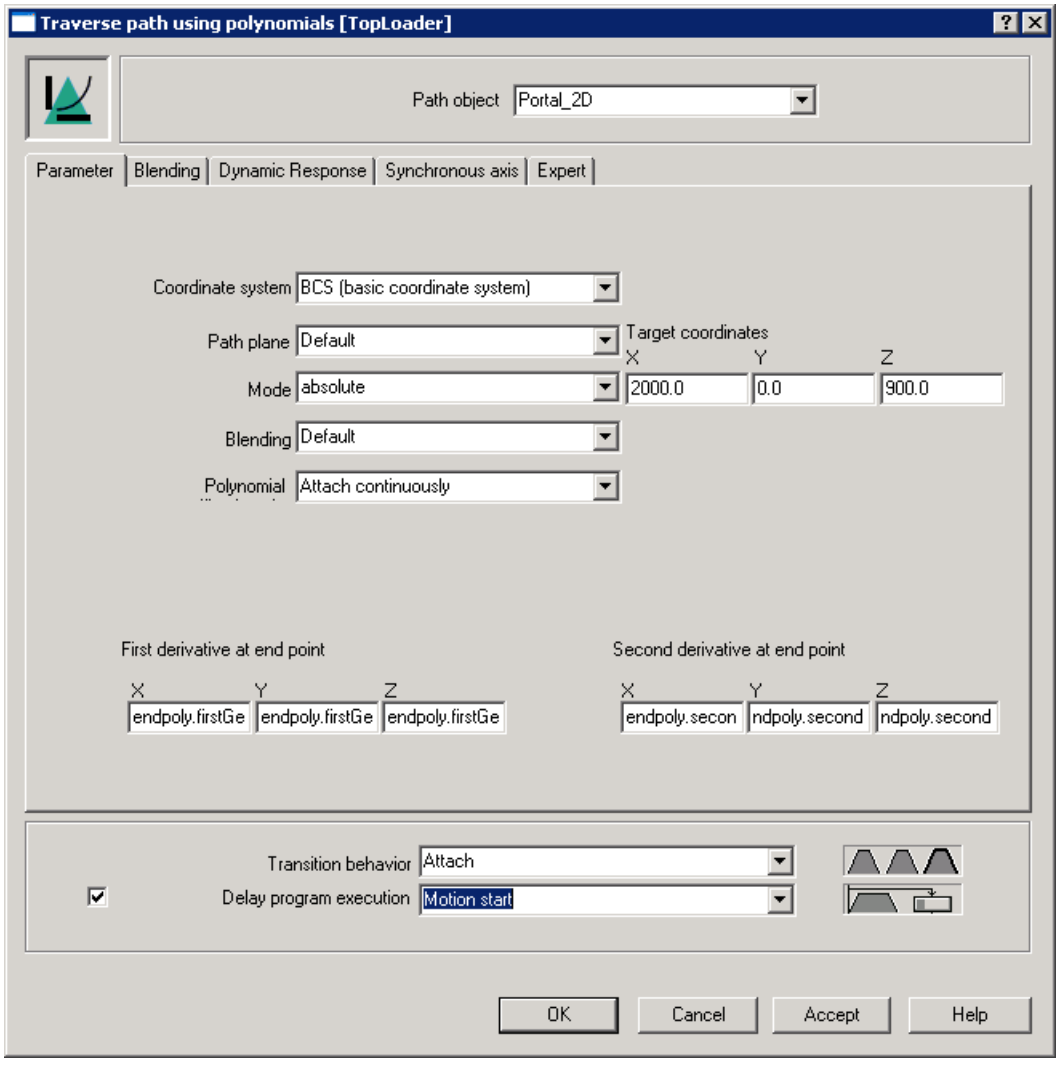


Figure 4-618 Programming the D-E polynomial path

Programming the E-F linear path

For the **E-F** linear path, add the **travel linear path** command and make the following settings:

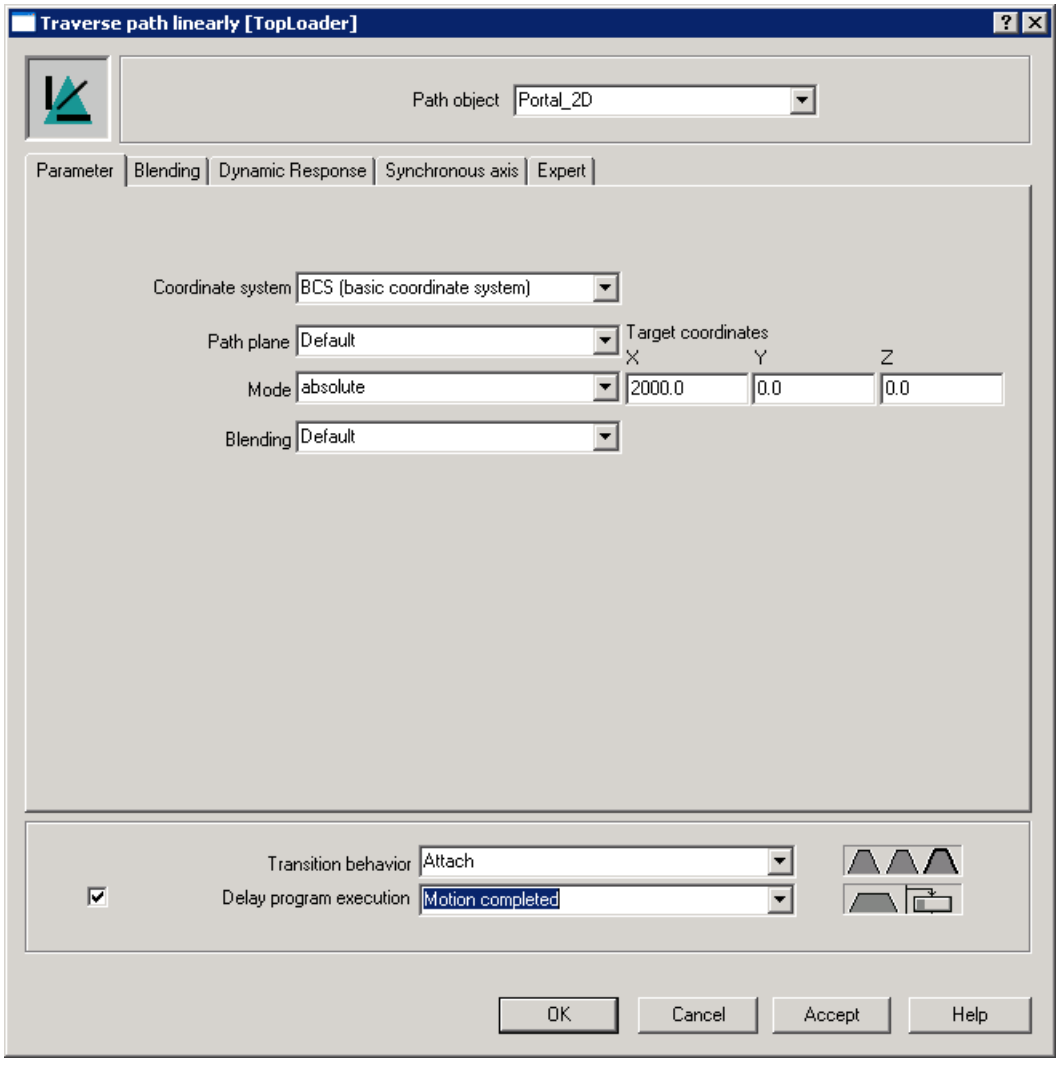


Figure 4-619 Programming the E-F linear path

Programming the F-A return travel

The gantry grabber should return to the initial position taking a circular path. The start point of the circular path is (2000, 0), the end point is (0, 0).

There are several ways of defining the circular path. For this example, the circular path should be defined using an intermediate point and the end point. The point (1000, 1000) is chosen as intermediate point.

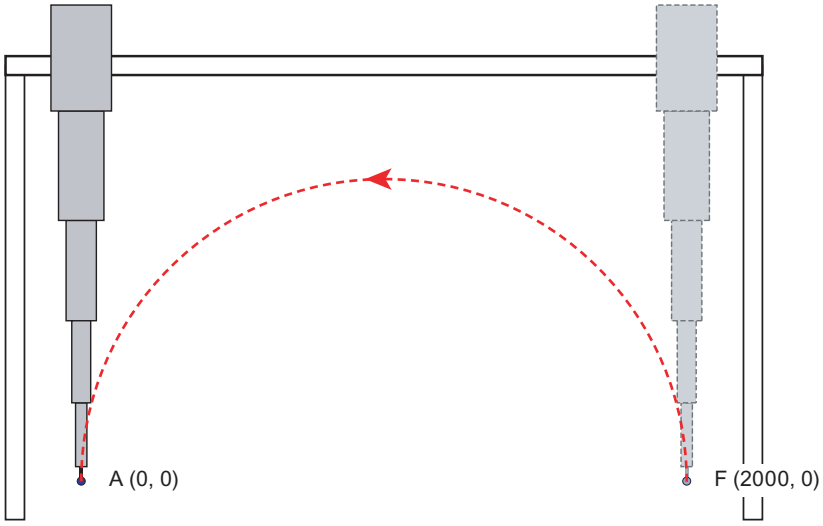


Figure 4-620 Defining the return circular path

To indicate the reverse motion, the **forw_back** variable is set to **false** in an ST zoom-in command.

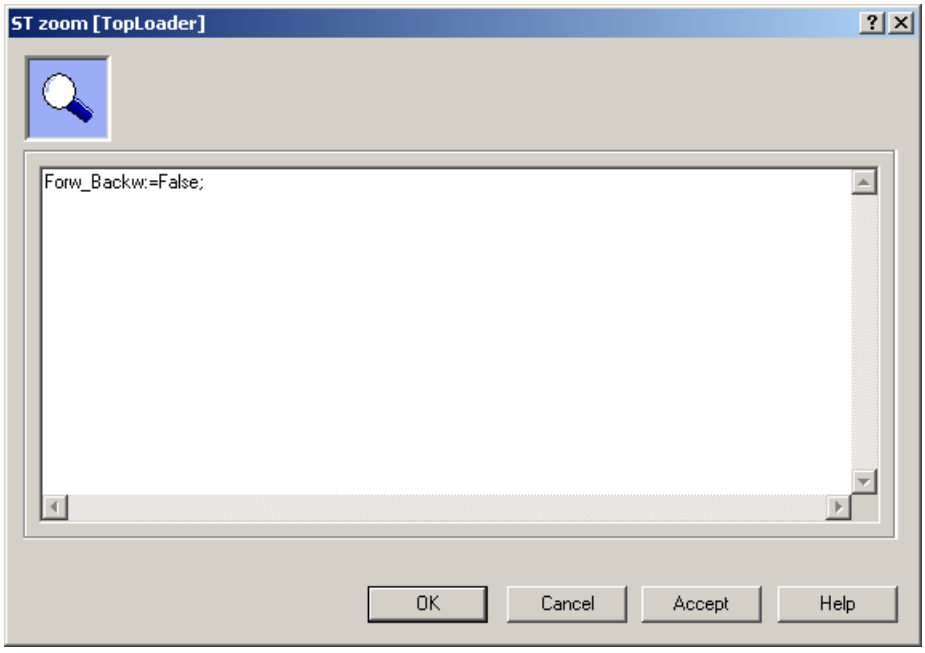


Figure 4-621 Set forw_back to false

The **travel circular path** command is then added and the following settings made:

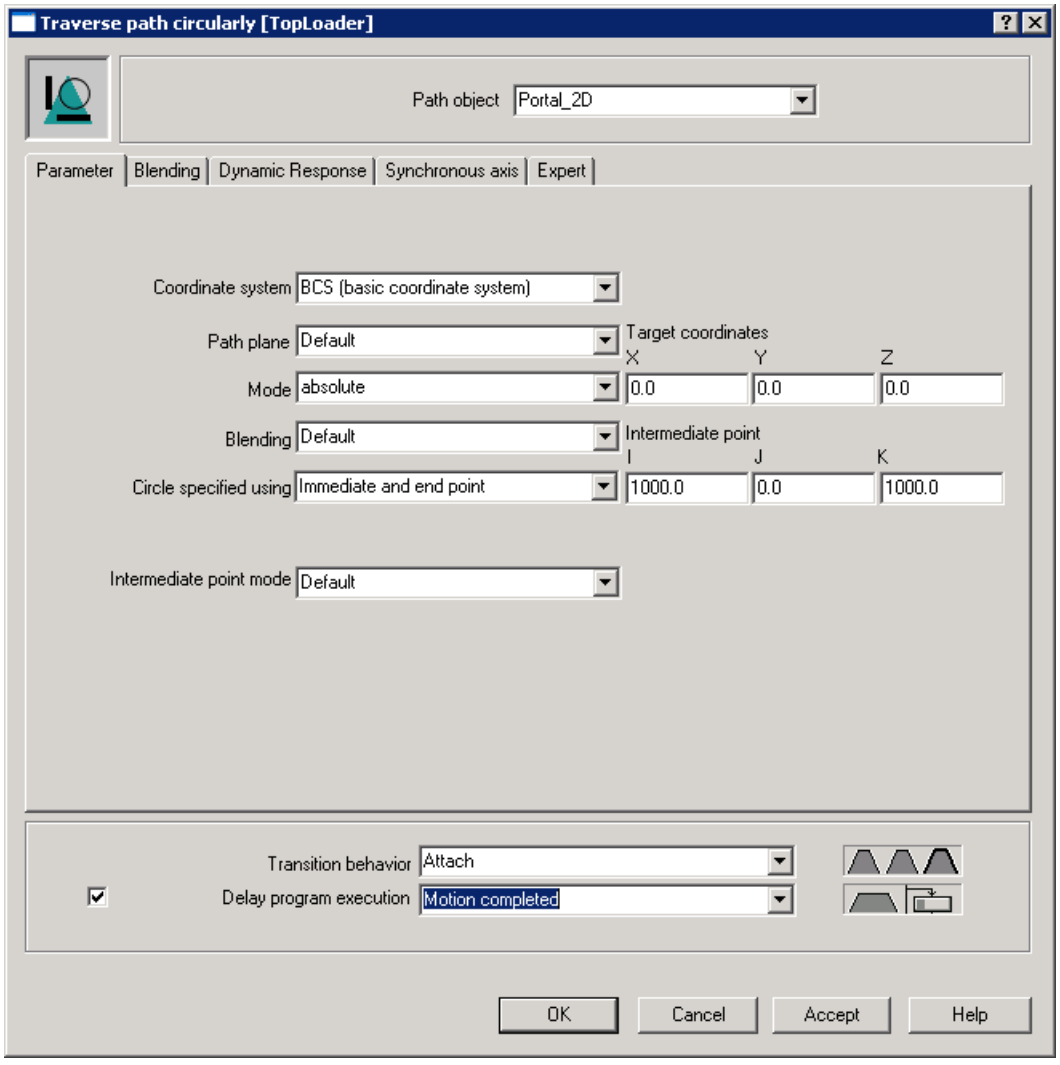


Figure 4-622 Programming the F-A circular path

Activating the axis enables and homing the axes

To move the axes, an enable must be made for each of them. The axes must also be homed. This requires that the the **enable axis** and **home axis** commands are added for each axis before the While loop.

You can use the default values for the **enable axis** commands.

For the **home axis** commands, set the **home coordinates** to **0 mm**. The other values do not need to be changed.

MCC diagram

The MCC chart now has the following form:

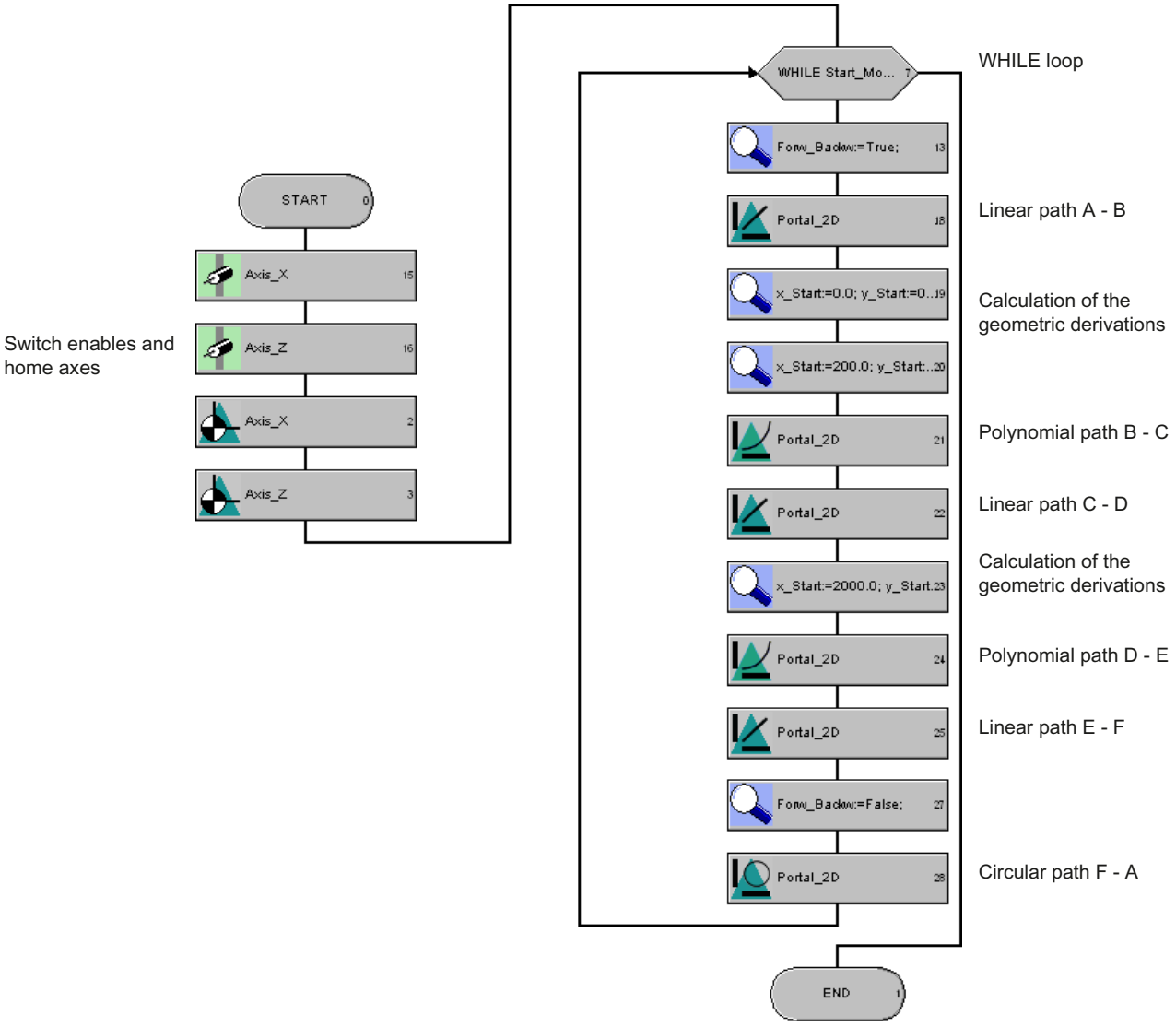


Figure 4-623 MCC chart

Assigning MCC chart in the execution system

The MCC chart must be assigned in the execution system to any MotionTask. The MotionTask must be activated after the StartupTask.

To assign the MCC chart to a MotionTask, proceed as follows:

1. In the project navigator, select any **MotionTask** and move the **MCC_Example.toploader** program to the **used programs**.

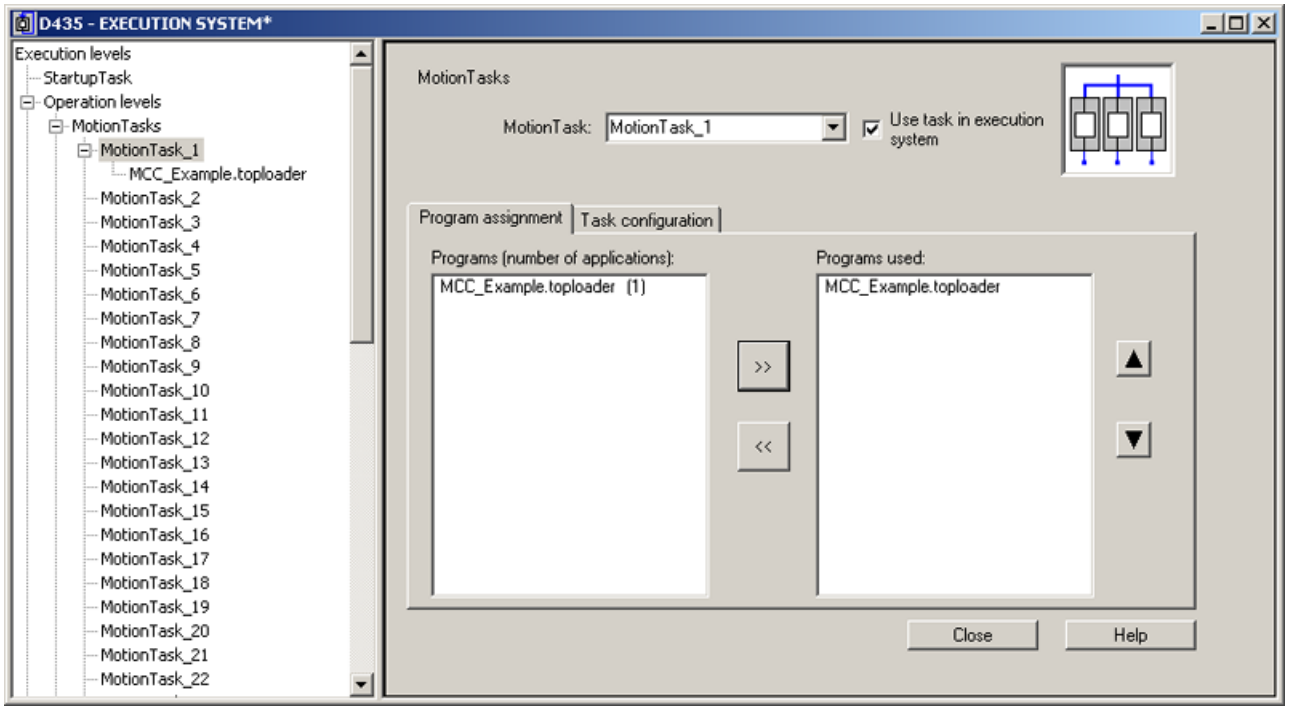


Figure 4-624 Assigning the MotionTask in the execution system

2. In the task configuration of the MotionTask, select **activation after StartupTask**.

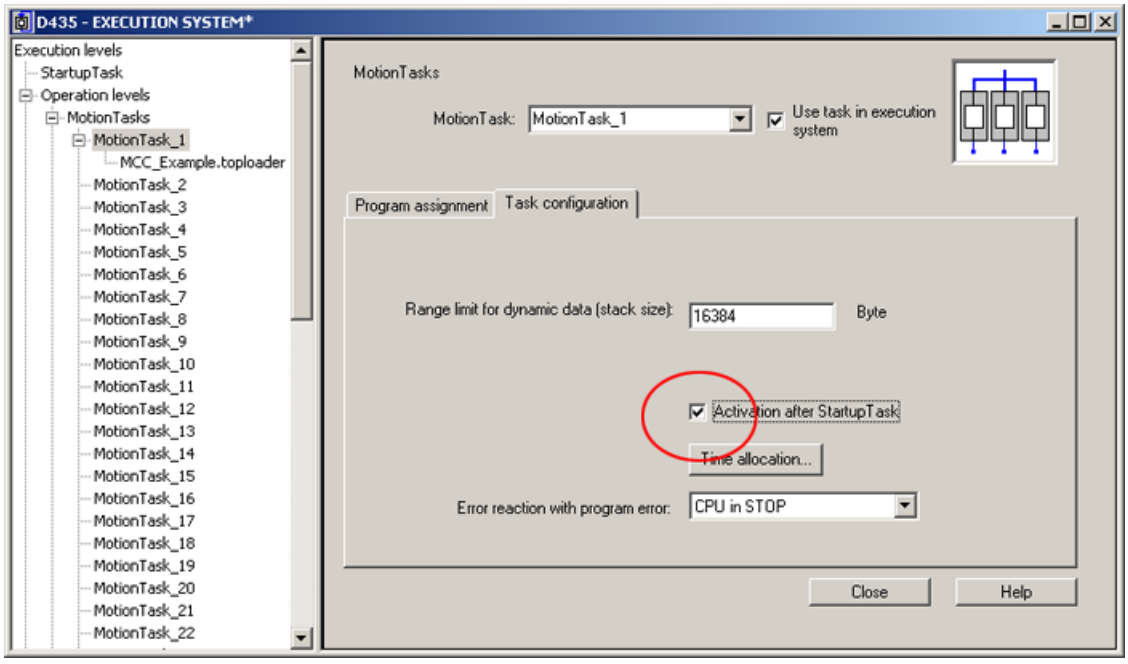


Figure 4-625 Configuring the MotionTask in the execution system

Checking a motion with trace

To see how the motion runs, a trace of the following variables is defined:

- **TO.Axis_X.positioningstate.actualposition**
- **TO.Axis_Z.positioningstate.actualposition**
- **Forw_back**

A log of the motion loop now has the following form:

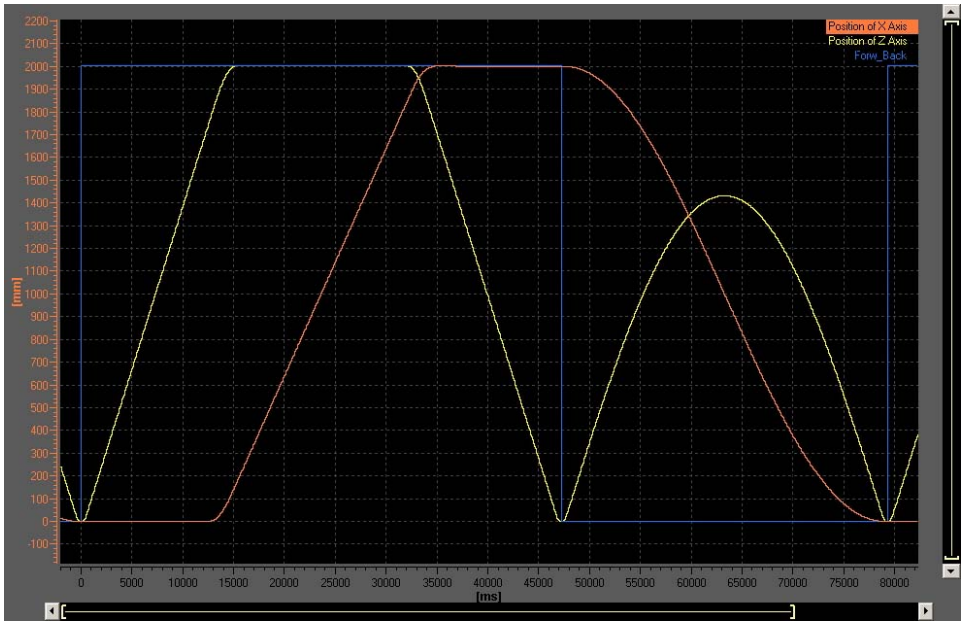


Figure 4-626 Result of the example as trace

4.6.5.9 Creating a path-synchronous axis

To show the functionality of the path-synchronous motion, a path-synchronous axis is added to the project. The path-synchronous axis is used, for example, to additionally rotate products during the motion. The path-synchronous axis is to rotate the gripper by 90° between the **C** and **D** points.

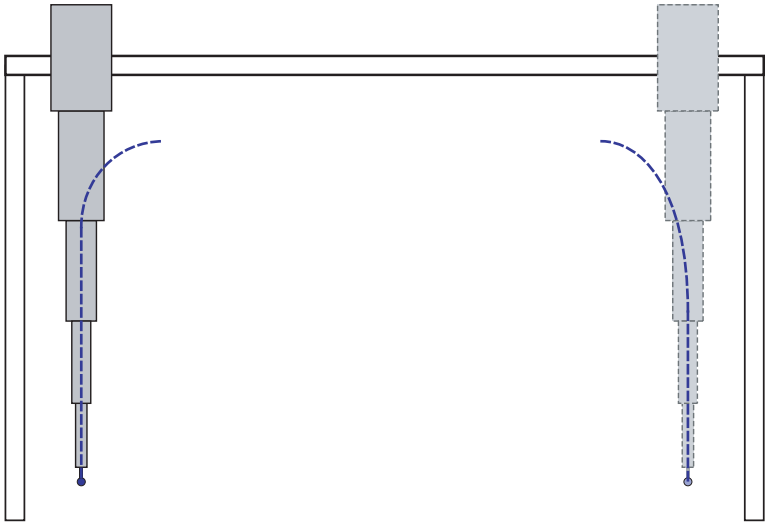


Figure 4-627 Path-synchronous axis

Creating an axis

Add a positioning axis with the name **Axis_Sync** to the project. This axis is created as linear, virtual, non-modular axis. Note that the path axis functionality is not used for this axis.

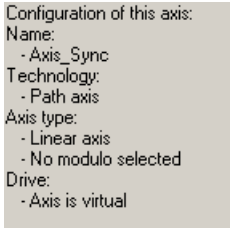


Figure 4-628 Creating the "Axis_Sync" axis

Creating a path object

For **interconnections** of the path object, select the **Axis_Sync** axis as **positioning axis for path-synchronous motions**.

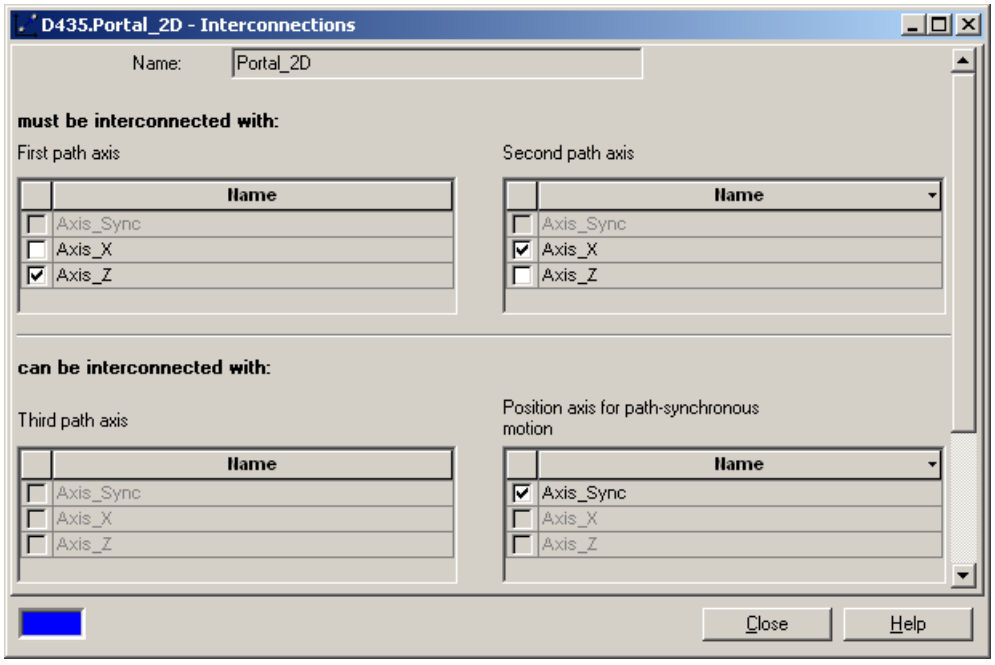


Figure 4-629 Interconnecting the "Axis_Sync" axis

Modifying MCC charts

Perform the following changes in the MCC chart:

- Before the While loop, add the **enable axis** and **home axis** commands for the **Axis_Sync** axis analog to those for the X- and Z-axis.
- The **Axis_Sync** axis should also rotate synchronously in the C-D linear path.
  To rotate the axis as required, open the travel command for the **C-D** linear path and select the **Synchronous axis** tab. Make the following settings there:

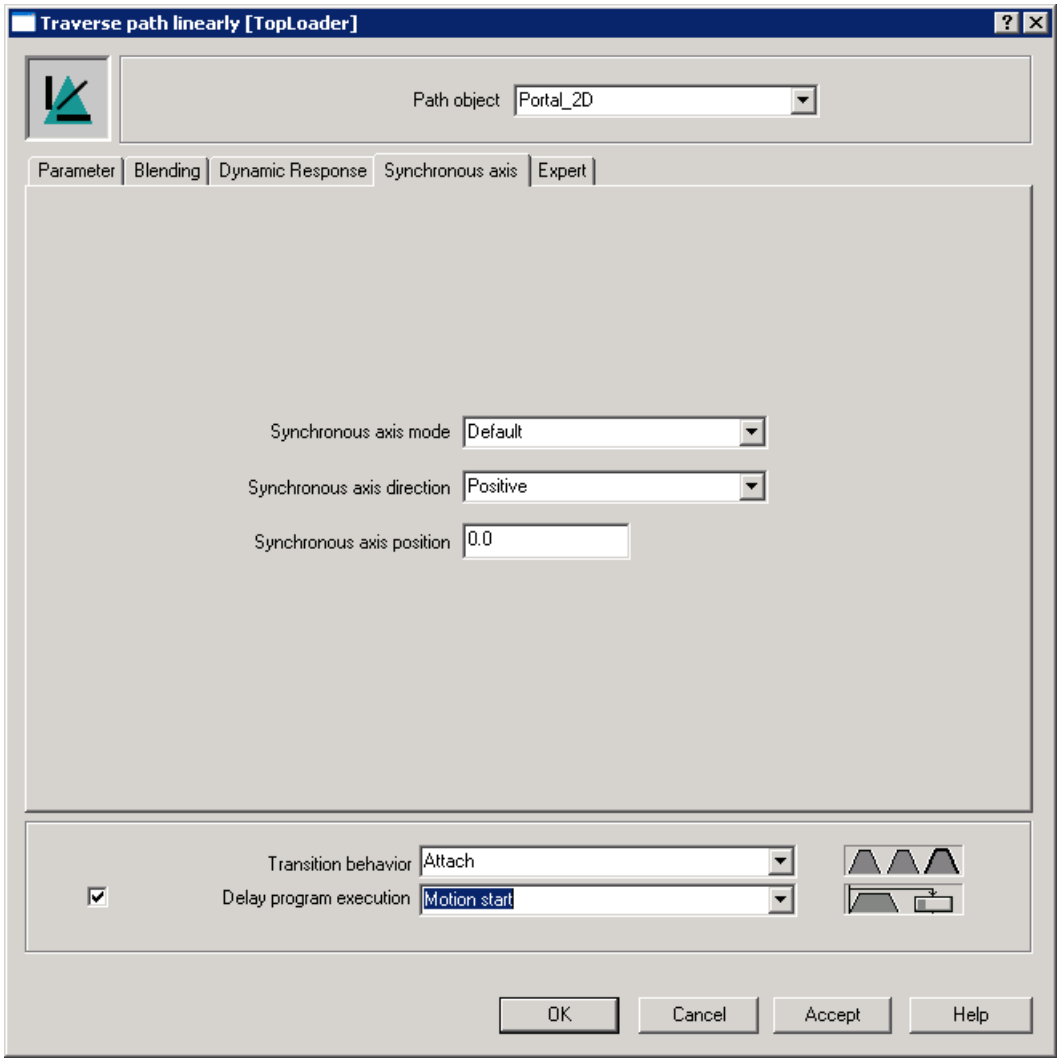


Figure 4-630 Rotating the "Axis_Sync" axis synchronously

- In the **F-A** circular path, the **Axis_Sync** axis should be rotated back to the zero position.

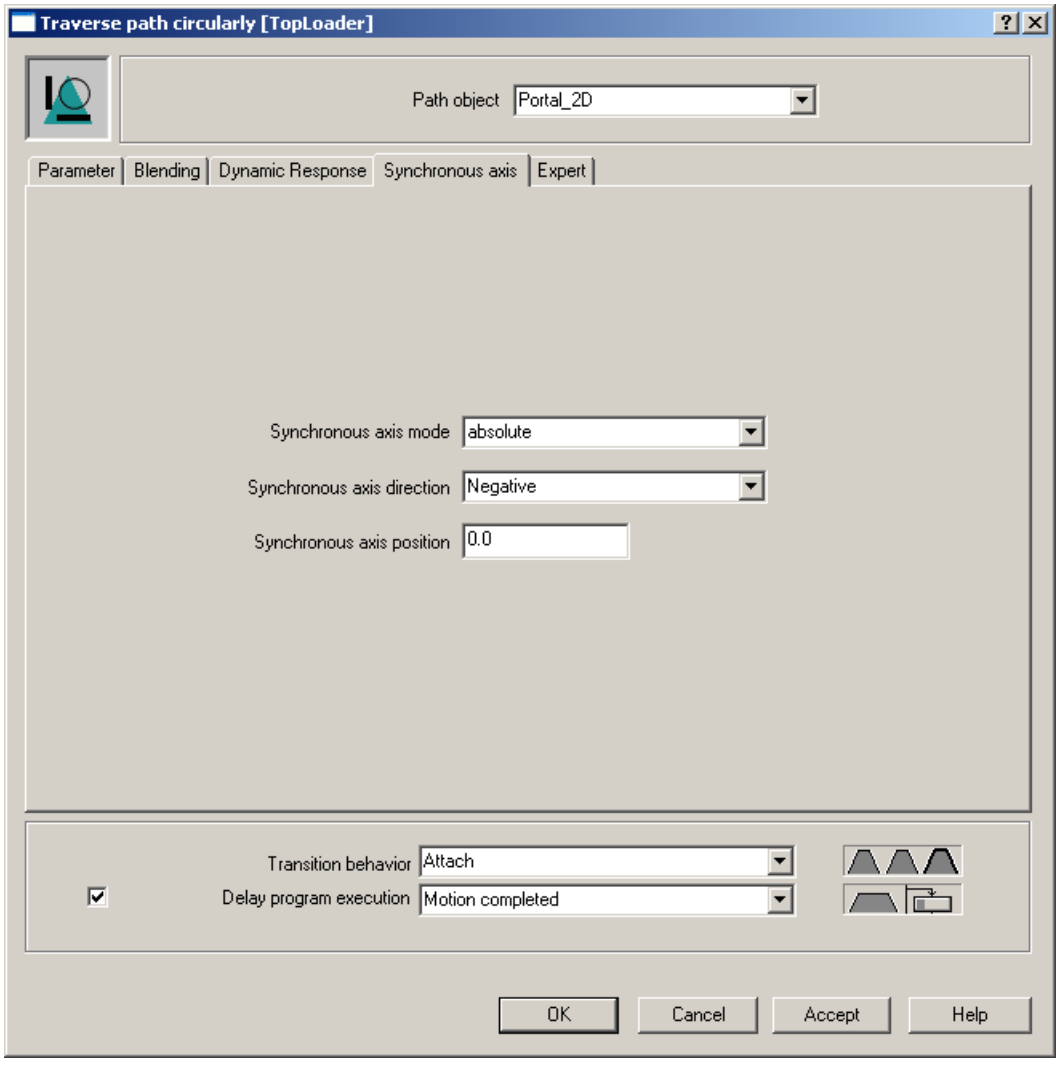


Figure 4-631 Moving the synchronous path back to the zero position

4.6.6 Programming/homing path interpolation

4.6.6.1 Programming

Programming: Overview

The following section provides information about the commands and the alarm responses of the path object technology package. Further information is contained in the reference lists of the technology packages.

The description of the functions for the "Toploading" standard library based on the TO path object is contained on the "Application Toploading" function description.

Overview of commands

Information and conversion

Calculating the path length:

The following commands calculate the path length without starting or executing the path motion.

The start and end points must be specified in the command.

- **_getLinearPathData()**
- **_getCircularPathData()**
- **_getPolynomialPathData()**

Geometric path analysis:

The commands listed below are used to calculate Cartesian path data, such as path direction and path curvature, at the start point, endpoint, and a specifiable point on the path without starting or executing the path motion.

The point on the path is specified by means of the default setting of the path length distance to the start point.

The start point is specified in the command, as is the position with reference to the path length where the information data is determined.

- **_getLinearPathGeometricData()**
- **_getCircularPathGeometricData()**
- **_getPolynomialPathGeometricData()**

With SIMOTION V4.2 and higher, the calculated geometry is stored in a geometry cache and a cacheID is provided as the return value. If the path segment is queried several times and only the queried point changes, this cacheID can be used to access the data already calculated. This can greatly improve the performance of the information functions.

Conversion commands

- **_getPathCartesianPosition()**
  Calculates the Cartesian positions from the axis positions.
- **_getPathAxesPosition()**
  Calculates the axis positions from the Cartesian positions.
- **_getPathCartesianData()**
  Calculates the Cartesian data for position, velocity, and acceleration from the axis positions, axis velocities, and axis accelerations.
- **_getPathAxesData()**
  Calculates the axis positions, axis velocity, and axis accelerations from the Cartesian data for position, velocity, and acceleration.

Command tracking

The current processing and motion status of motion commands can be tracked using the following commands. Permanent storage of the states associated with a **CommandId** makes it possible to evaluate them beyond the lifetime of the motion command.

- **_getStateOfPathObjectCommand()**
- **_getMotionStateOfPathObjectCommand()**
- **_bufferPathObjectCommandId()**
- **_removeBufferedPathObjectCommandId()**

Motion

- **_movePathLinear()**
  Interpolation of linear paths (Page 2480)
  - 2D in a main plane
  - 3D
- **_movePathCircular()**
  Interpolation of circular paths (Page 2481)
  - 2D in a main plane with radius, end point, and orientation
  - 2D in a main plane with center point and angle
  - 2D with intermediate and end points
  - 3D with intermediate and end points
- **_movePathPolynomial()**
  Interpolation of polynomial paths (Page 2485)
  - 2D in a main plane
  - 3D
- **_stopPath()**
  Stops the current motion without terminating.
- **_continuePath()**
  Continues a stopped motion.
  With V4.2 and higher, the motion properties can be stated to continue the movement.

Object and Alarm Handling

- **_cancelPathObjectCommand()**
  Cancels a command that is waiting or active in the Ipo. The command to be canceled is specified by the indication of its CommandId in the **commandToBeCancelled** parameter.
- **_enablePathObjectSimulation()**
  Places a path object in simulation mode, path values are calculated, but are not output on the axes.
- **_disablePathObjectSimulation()**
  Ends simulation mode.

- **_resetPathObject()**
  This command resets the path interpolator to its initial state. Active commands are stopped, commands are aborted, and errors are reset. If required, system variables can be reset to their default values or configuration data can be read in again.
- **_resetPathObjectError()**
  This command resets all errors or a specific error on the path object.
- **_getPathObjectErrorNumberState()**
  Reads out the status of a specific error.
- **_getPathObjectErrorState()**
  This command provides information on whether alarms are pending on the path object and if so, how many. It also outputs information about these errors.
- **_resetPathObjectConfigDataBuffer()**
  Resets the configuration data buffer.
- **_getStateOfPathObjectMotionBuffer()**
  Returns the status of the command queue of the path object.
- **_resetPathObjectMotionBuffer()**
  Clears all pending commands from the command queue.
  The **030002 Command aborted** alarm is issued for each of the deleted commands.

Object coordinates

- **_enablePathObjectTrackingSuperimposed()**
  Starts the synchronization action of a path object on an OCS.
- **_getPathObjectBCSFromOCSData()**
  Calculates a position (x, y, z) in the base coordinate system of the path object using the position in the object coordinate system.
- **_getPathObjectOCSFromBCSData()**
  Calculates a position (x, y, z) in the object coordinate system using the position in the base coordinate system of the path object.
- **_redefinePathObjectOCS()**
  Displaces the object coordinate system along the X-axis (travel direction).
- **_setPathObjectOCS()**
  Defines the translational and rotatory displacement of the object coordinate system compared with the base coordinate system of the path object.

Command execution

Command buffer

The path object has three command buffers for every command.

- One buffer for motion commands has an immediate (**IMMEDIATELY**) and sequential (**SEQUENTIAL**) effect
- A separate buffer for stopping path motion (**_stopPath()**) and continuing path motion (**_continuePath()**)
- A buffer for other (i.e. superimposed) instructions

Command	Function	Position*)
_movePathLinear()	Starts linear path motion	4
_movePathCircular()	Starts circular path motion	4
_movePathPolynomial()	Starts polynomial path motion	4
_stopPath()	Stops motion	1 for stop without command abort 4 for stop with command abort
_continuePath()	Resume motion	1
_getLinearPathData()	Linear path length	5
_getCircularPathData()	Circular path length	5
_getPolynomialPathData()	Polynomial path length	5
_getPathCartesianPosition()	Axis to path	5
_getPathAxesPosition()	Path to axis	5
_getPathCartesianData()	Axis to path with dynamic response data	5
_getPathAxesData()	Path to axis with dynamic response data	5
_getLinearPathGeometricData()	Geometric linear path data	5
_getCircularPathGeometricData()	Geometric circular path data	5
_getPolynomialPathGeometricData()	Geometric polynomial path data	5
_enablePathObjectSimulation()	Places path object in simulation mode	3
_disablePathObjectSimulation()	Resets the path object out of simulation	3
_resetPathObject()	Resets the path object	5
_resetPathObjectError()	Reset error	5
_getStateOfPathObjectCommand()	Read out command status	5
_getMotionStateOfPathObjectCommand()	Read out motion phase of a command	5
_bufferPathObjectCommandId()	Permanently stores the command ID	5
_removeBufferedPathObjectCommandId()	Terminates permanent storage	5
_getPathObjectErrorNumberState()	Reads out the status of a path object error	5
_getPathObjectErrorState()	Reads out the status and number of the pending path object error	5
_resetPathObjectConfigDataBuffer()	Deletes that configuration data collected in the buffer since the last activation	5
_getStateOfPathObjectMotionBuffer()	Returns the status of the command queue of the path object.	5

Command	Function	Position*)
_resetPathObjectMotionBuffer()	Clears all pending commands from the command queue.	5
_enablePathObjectTrackingSuperimposed()	Starts the synchronization action of the path object on an OCS.	3
_getPathObjectBCSFromOCSData()	Calculates a position in the BCS using a position in the OCS.	5
_getPathObjectOCSFromBCSData()	Calculates a position in the OCS using a position in the BCS.	5
_redefinePathObjectOCS()	Displaces the OCS along the X-axis.	3
_setPathObjectOCS()	Defines the displacement of the OCS compared with the BCS.	3
_cancelPathObjectCommand()	Cancels a command that is waiting or active in the Ipo.	1

***) Legend:**

1 Buffer for Stop-Continue commands
2 Not used
3 Buffers for superimposed commands
4 Sequential command buffer
5 Not assigned to the command buffers (commands can be executed in parallel)

Override behavior

The response to command insertion is defined in the **mergeMode** command parameter.

- Override current interpolator command
  (**mergeMode:=IMMEDIATELY**)
- Overwrite commands in the buffer
  (**mergeMode:=NEXT_MOTION**)
  This command is executed as soon as the current interpolator command has been executed.
- Append to commands already in the buffer
  (**mergeMode:=SEQUENTIAL**)
  If the buffer is full, the command can wait until an entry becomes available or the command execution can continue without a wait time.

Commands are decoded in the task context, i.e. in the execution level of the user task that issued the command (before it is entered in the command buffer).

mergeMode has the same settings and action as in the Axis technology object.

Interactions between the path object and the axis

Override behavior

The response to other active motions on the axis is defined with the **mergeMode** parameter of the path object motion command.

With the **substitute** setting, all active assigned axis motions are canceled (as a function of the **transferSuperimposedPosition** setting in the configuration data).

- When a path motion is substituted by an additional path command, an immediate transition takes place to the path that yields the new end point at the point of the substitution.
- When a path is substituted by a motion command, such as **_move...()**, the other axes involved in the path motion and, if applicable, the positioning axis for path-synchronous motion stop with the maximum dynamic values.
  The action of the other override responses is the same as for synchronous operation.
  The following applies when motion commands occur simultaneously on the respective object within one interpolation cycle, i.e. one on the synchronous object, one on the path object, and one on the axis:
  - When **mergeMode=SEQUENTIAL/NEXT_MOTION**, the synchronous/path command is executed.
  - When **mergeMode=IMMEDIATELY**, the command on the axis is executed.
- The **_stop()** command from an axis involved in the path motion or participating via the path-synchronous motion does not stop the path motion on the path object, i.e. it has no effect (this is the same behavior as for the synchronous motion on the Synchronous technology object).
  _stop() only stops motions that were initiated on the axis.
  The **_stopEmergency()** command is also effective on motions initiated on the Synchronous operation and Path object technology objects.
- Superimpositions can only take place on the axis (axis motion or synchronous operation), not on the path object.
- If a positioning axis for path-synchronous motion of the path group is overridden by means of an axis command on this axis, this has the same effect on the path as when a path axis is overridden.
- Path axes and positioning axes for path-synchronous motion have the same response.

With the other settings, the path motion is started after the end of all previous axis motions or an active path motion.

Sequence of effectiveness

Technology objects are processed in the sequence: Path object - Synchronous object - Axis object. In the case of simultaneous motion commands on several technology objects that are interconnected with an axis and have the same override response, the motion commands take effect according to the processing order and the setting in the **mergeMode** parameter:

- When **mergeMode:=IMMEDIATELY**, the motion command on the axis takes effect (last processed command).
- When **mergeMode:=SEQUENTIAL/NEXT_MOTION**, the motion command on the path object takes effect (first effective command).

Interaction with the axis

If the motion of an axis is stopped as a result of its local alarm response or if the interconnection of the path object to the axis becomes invalid, the path motion is canceled, the other axes are also traversed with the maximum dynamic values to velocity 0.0, and a technological alarm is issued.

If the remaining distance is smaller than the deceleration distance, the new target position is overshot, and the axis travels back to the target position (with a reversing motion). The other axes travel with their maximum dynamic values to velocity 0.0. However, these axes do not travel back to the cancellation point. Depending on the general conditions (number of participating axes, dynamic values), the path is no longer maintained.

See **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Motion transitions"

Interactions with other path motions

If a path axis is interconnected with several path objects, and if a path motion on a path axis substitutes another path motion on another path object, the other path axes are traversed with the maximum dynamic values to velocity 0.0, and a technological alarm is issued.

4.6.6.2 Local alarm response

Local alarm responses are specified by means of the system.

The following responses are possible:

- **NONE**: No response
- **DECODE_STOP**: Command decoding is canceled, but the current motion and command in the buffer remain active.
- **END_OF_MOTION_STOP**: Abort at end of error-causing command; motion on the path stops.
- **MOTION_STOP**: Controlled motion stop with programmed dynamic path values on the path. Motion can be continued by acknowledging the error.
- **MOTION_EMERGENCY_STOP**: Controlled motion stop with maximum dynamic path values/ limit values for the axis. Motion can be continued by acknowledging the error.
- **MOTION_EMERGENCY_ABORT**: Controlled motion stop with maximum dynamic path values on the path. Active commands in the path interpolator are canceled; read-in of new commands is prevented and is only possible following error acknowledgement. Active commands (IPO) are fed back to the user program with an error code.
- **DISABLE_MOTION**: Motion stop on path axes and positioning axis for path-synchronous motion. The path motion component is realized by means of a stop with the maximum dynamic values of the axis followed by cancellation of the path motion component. Thus, the path group is ungrouped. Active commands in the path interpolator are canceled; read-in of new commands is prevented and is only possible following error acknowledgement.

4.6.7 Appendix A

4.6.7.1 Specific kinematics with TrafoID 1001

Type of kinematics

The kinematics can be used with the TrafoID 1001. However, it can only be selected via the expert list by entering the corresponding ID there.

Kinematics 1001 are 2D kinematics (X-Y)

The following types of axes can be used:

- Axis A1 (X axis): Rotary or linear axis without modulo function
- Axis A2 (Y axis): Rotary axis without modulo function

Display of kinematics

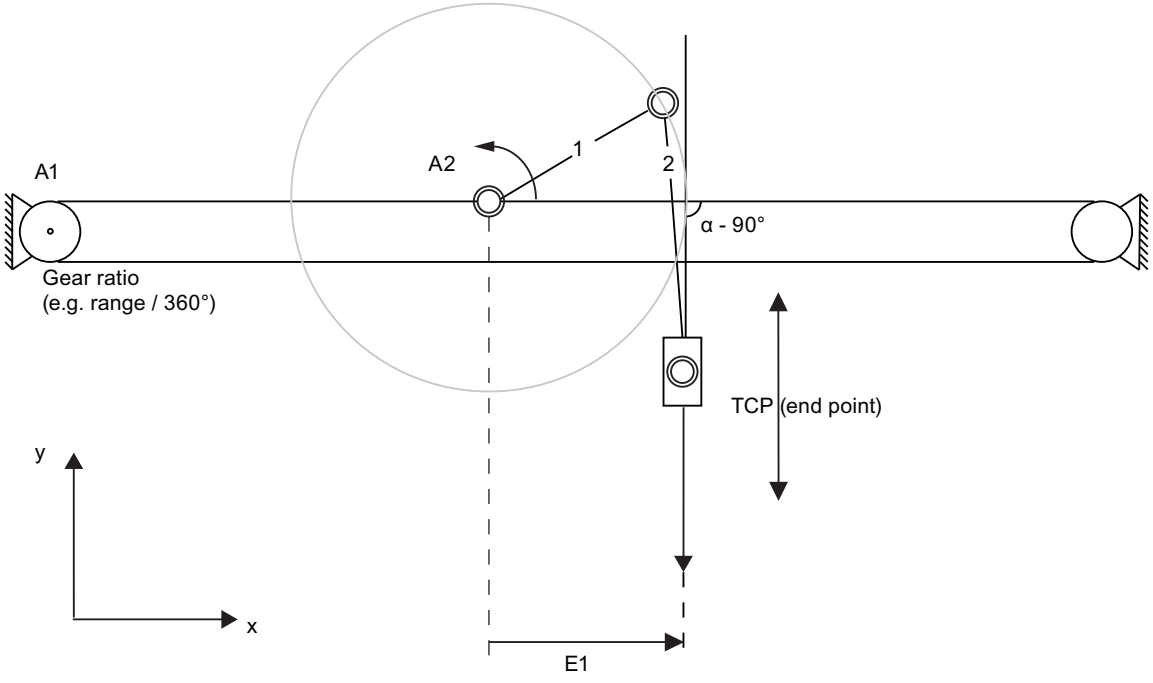


Figure 4-632 Kinematics display

Settings on path object

Kinematics.typeOfKinematics = SPECIFIC (6)

Kinematics.specTrafoId = 1001

Table 4-277 Kinematics parameter

Parameter	Use	Unit	Type	Min.	Max
Kinematics.parameter[1]	basicOffsetX	e.g. mm	LREAL	-1E+012	1E+012
Kinematics.parameter[2]	basicOffsetY	e.g. mm	LREAL	-1E+012	1E+012
Kinematics.parameter[3]	Reserved				
Kinematics.parameter[4]	ratioA1	e.g. mm/°	LREAL	-1E+012	1E+012
Kinematics.parameter[5]	length1	e.g. mm	LREAL	0	1E+012
Kinematics.parameter[6]	length2	e.g. mm	LREAL	0	1E+012
Kinematics.parameter[7]	distanceE1	e.g. mm	LREAL	-1E+012	1E+012
Kinematics.parameter[8]	verticalPosition2 lower / upper	0: lower, ≠0: upper	LREAL	-1E+012	1E+012
Kinematics.parameter[9]	offsetA2	e.g.	LREAL	-1E+012	1E+012

Kinematics reference

The position of the TCP/ end point depends on the orientation (see **Traversing range** section) of rod 2. If the **lower** orientation is selected (as in the kinematics reference diagram), the position of the TCP/ EP can be determined as follows if the kinematics are in a neutral position:

$$X_{(EP\_bottom)} = 0.0 - basicOffsetX$$

$$Y_{(EP\_bottom)} = -\sqrt{(length2)^2 - (length1 - distanceE1)^2} - basicOffsetY$$

If the **upper** orientation is selected:

$$X_{(EP\_top)} = 0.0 - basicOffsetX$$

$$Y_{(EP\_top)} = \sqrt{(length2)^2 - (length1 - distanceE1)^2} - basicOffsetY$$

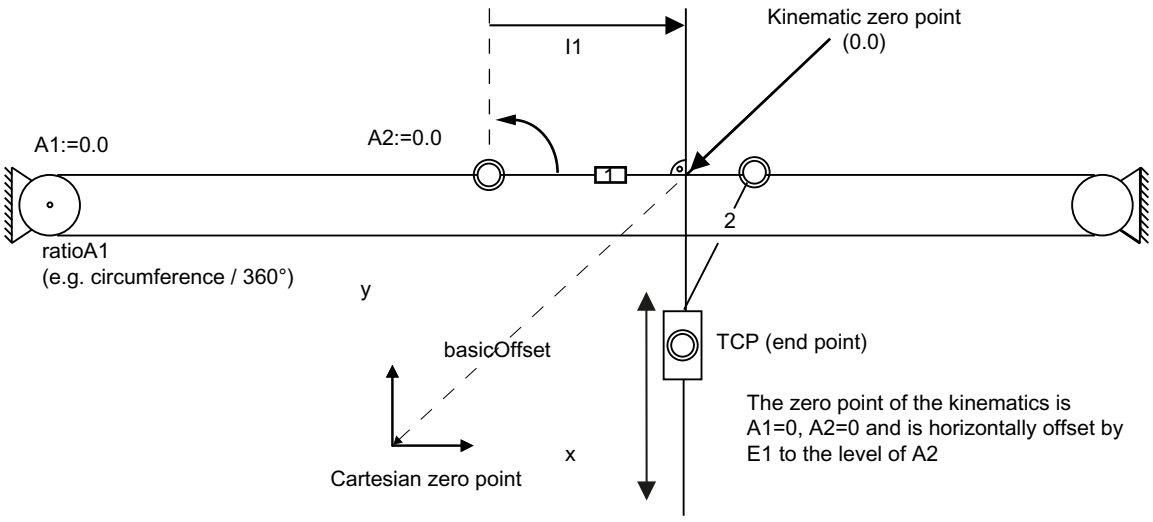


Figure 4-633 Kinematics reference

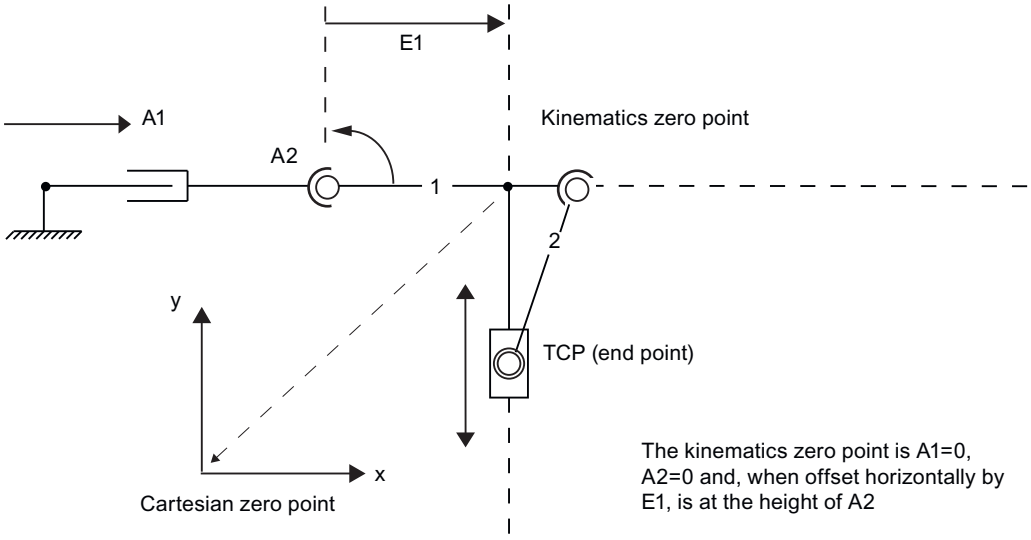


Figure 4-634 Schematic display - kinematics 1001

Transformation in X direction

Transformation in X direction can be used for linear and rotary axes (belt drive, spindle etc.).

BCS.x.position=(axis position A1 * parameter[4]) - parameter[1]

BCS.x.position=(axis position A1 * ratioA1) - basicOffsetX

Transformation in Y direction

Within the traversing range, transformation (MCS -> BCS) of the path object provides valid Cartesian values. If the axis is outside this range, 0.0 is output.

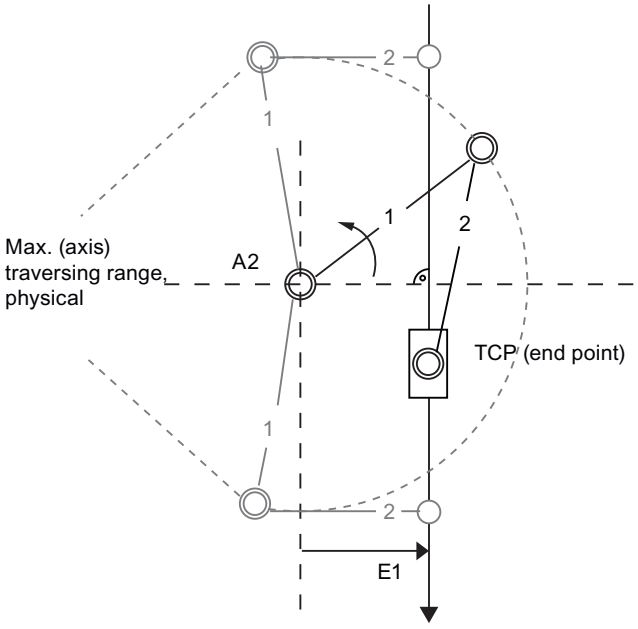


Figure 4-635 physical traversing range

Traversing range

The traversing range is determined by the orientation (position) of the sliding range. This can be determined using Kinematics.parameter[8].

Upper orientation

Kinematics.parameter[8] ≠ 0.0 (e.g. 1.0)

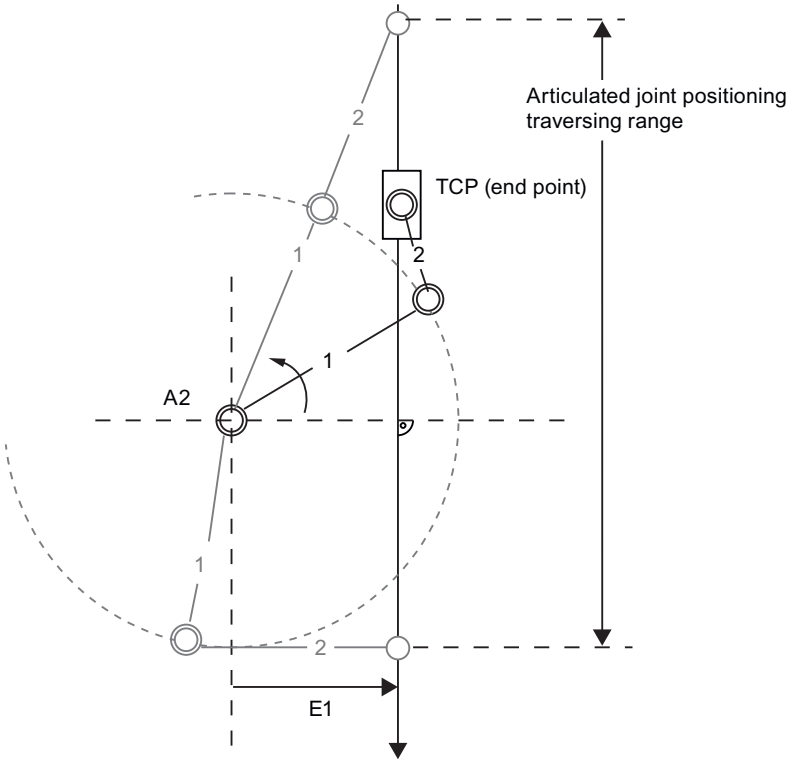


Figure 4-636 Traversing range with orientation selected as upper

Lower orientation

Kinematics.parameter[8] = 0.0

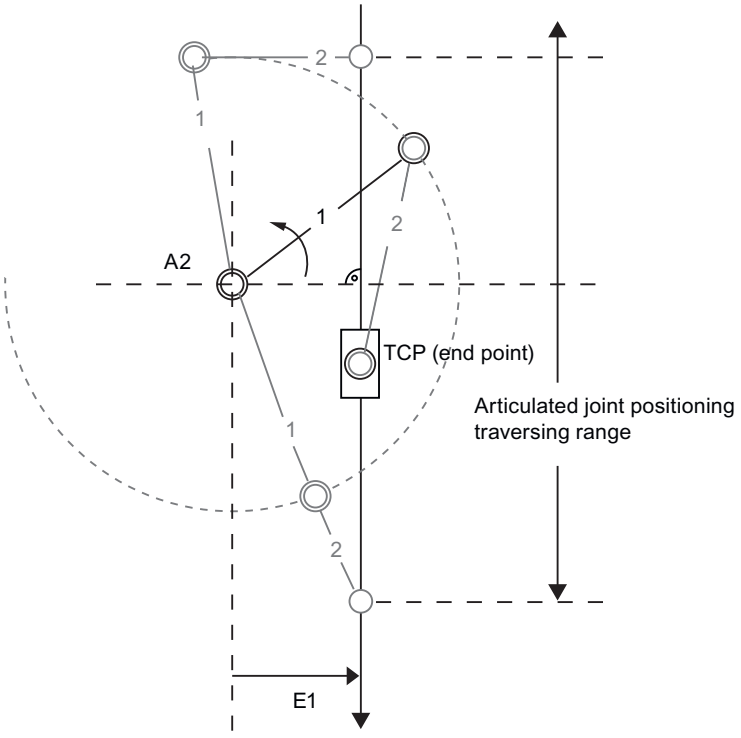


Figure 4-637 Traversing range with orientation selected as lower

4.7 Technology Objects Synchronous Operation, Cam

Preface

Preface

This **document** is part of the **System and Function Descriptions documentation package**.

Scope

This manual is valid for the following versions:

- SIMOTION SCOUT V4.5
- SIMOTION Kernel V4.5, V4.4, V4.3, V4.2, V4.1, V4.0, V3.2, V3.1 or V3.0
- SIMOTION technology packages Cam, Cam_ext/Path (as of Kernel V3.2) and TControl in the version for the respective kernel (including technology packages Gear, Position and Basic MC up to Kernel V3.0)

Chapters in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- Synchronous Operation (Page 2624) (Part I)
  Function of the synchronous operation, i.e. the grouping of a master object and a following axis
- Distributed Synchronous Operation (Page 2758) (Part II)
  Function of the distributed synchronous operation, i.e. synchronous operation across different controllers
- Synchronous Operation IPO - IPO_2 (Page 2824) (Part III)
  Function of synchronous operation with master object and following axis in different interpolator cycle clocks (IPO or IPO_2)
- Cam (Page 2830) (Part IV)
  Function of the Cam technology object
- Index
  Index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P

- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.7.1 Fundamental safety instructions

4.7.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.7.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.7.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.7.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.7.2 Part I - Synchronous Operation

4.7.2.1 Overview of synchronous operation

This part describes the function of the **synchronous operation** technology. It introduces you to the setting and configuration functions, as well as providing information about the supplementary conditions and the operating characteristics of synchronous objects.

What is synchronous operation used for?

Synchronous operation functions are taking on an increasingly significant role in automation engineering. The progress in open-loop and closed-loop control engineering and the availability of increasingly more powerful systems mean that solely mechanical solutions are more and more frequently being replaced with "electronic" variants.

The synchronous operation functions of the SIMOTION technology provide the option of replacing rigid mechanical connections with "control engineering", thus producing more flexible, maintenance-friendly solutions.

What is synchronous operation in SIMOTION?

The synchronous operation functionality of axes is provided by the synchronous object. A leading object (master) generates a master value, which is processed by the synchronous object according to specific criteria (gear ratio, scaling, offset, cam) and assigned to the following axis (slave) as a reference variable.

Mechanical model

The mechanical model for a synchronous operation relationship is, for example, a gear with a drive wheel and an output wheel. The model for camming could be a cam gear with a mechanical cam and sampling mechanism. A coupling used for enabling and disabling the following motion on-the-fly is also used as a model.

Synchronous operation functions

The following synchronous operation functions can be implemented:

- With gearing (Page 2631), a linear transmission function between a master value and a following axis can be achieved using control engineering, thus producing the same result as could be achieved mechanically using a gear. A gear ratio can be specified for use in linear mapping of the master axis position onto the following axis position.

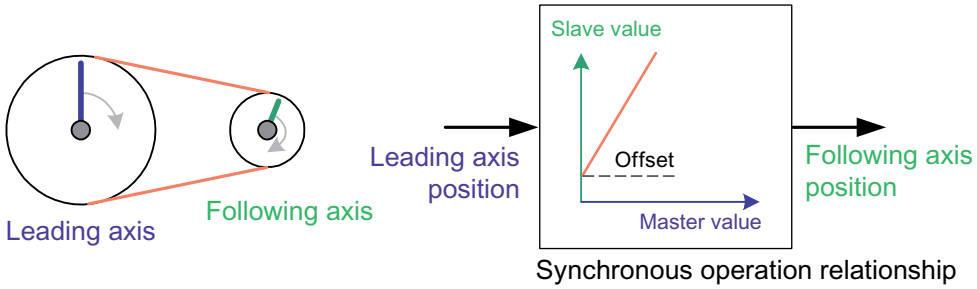


Figure 4-638 Gearing synchronous operation function (mechanical example)

- With velocity gearing (Page 2637), a constant velocity coupling is implemented (as of V3.1).
- With camming (Page 2638), a non-linear transmission function between a master value and following axis can be achieved. The slave value is generated from the master value using the transmission function defined in the cam. The cam is defined using interpolation points or mathematical functions and is interpolated between the specifications.

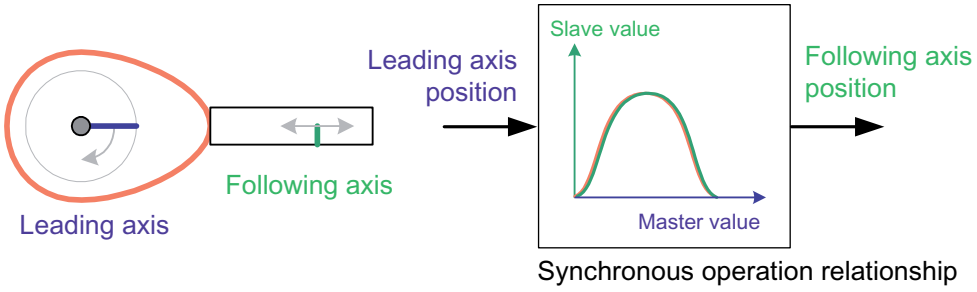


Figure 4-639 Camming synchronous operation function (mechanical example)

A synchronous operation sequence

Synchronous operation of a following axis to a master value using the SIMOTION synchronous operation functions is divided into three phases:

- Synchronization
- Synchronized traversing
- Desynchronization

Within these phases, there are several options for influencing the synchronous operation functions.

Synchronization/Desynchronization

The synchronous operation to the master value during synchronization or desynchronization can be defined differently depending on the application.

It is determined on the basis of:

- The synchronization criterion/synchronization position
- Synchronization direction
- The position of the synchronization range relative to the synchronization position
- Synchronization profile

See the section titled Synchronization (Page 2652).

Objects

A *synchronous operation relationship* exists between the following objects:

- At least one *master object* (master)
  The master object is a technology object that provides motion information with a position (the motion slave value). This can be, for example, a positioning axis or an external encoder.
- At least one *synchronized axis*, comprising:
  - A *following axis* (slave)
  - One or two *synchronous objects*
  - Possibly one or more *cams*

A synchronous object is automatically created as a separate object in SIMOTION SCOUT when an axis with synchronous operation technology is created.

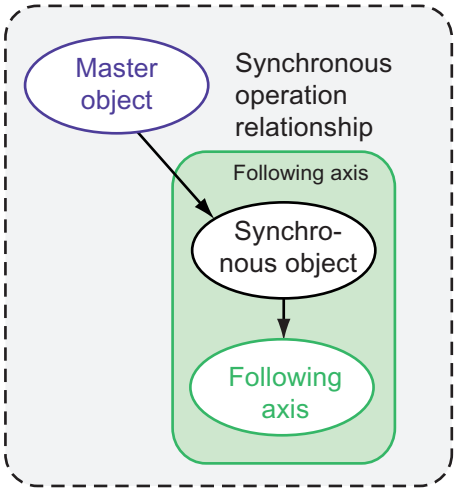


Figure 4-640 Objects in gearing

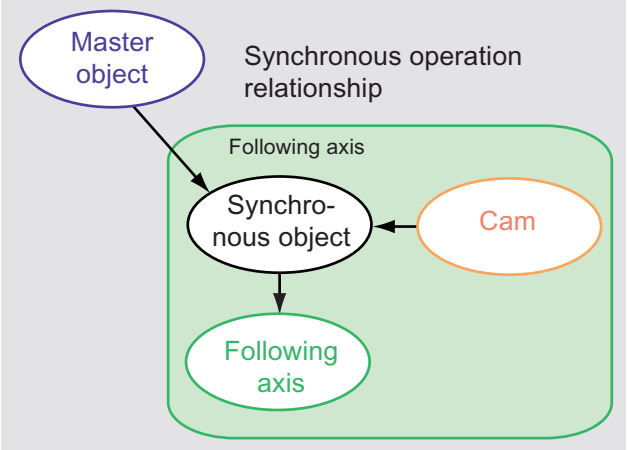


Figure 4-641 Objects in camming

Master values

The master value can be specified by the following technology objects:

- Axis
- External encoder

With restrictions (not for distributed synchronous operation or synchronous operation IPO-IPO_2), the following technology objects can also specify the master value:

- Fixed gear
- Addition object
- Formula object

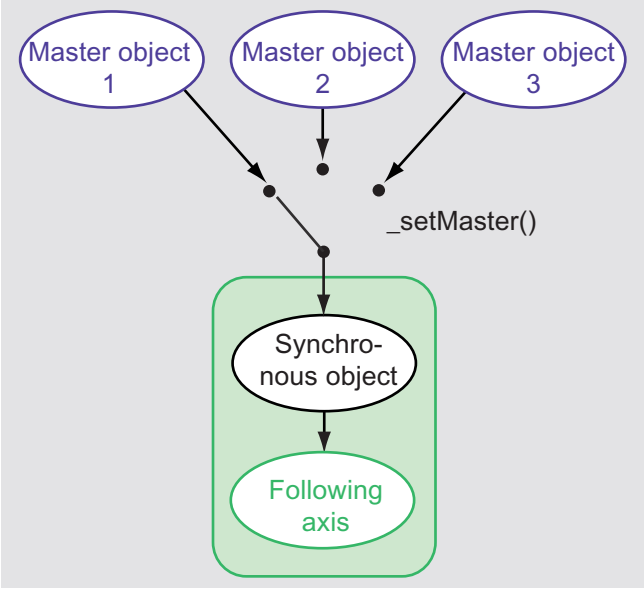


Figure 4-642 Example: Synchronous object with several master values

A following axis can be interconnected with more than one master value by means of the synchronous object. However, only one of these master values can be activated at any given

time. The process of switching over to a different master value is described in the section titled Switching over the master value source (Page 2680).

When axes serve as the master value source, the setpoint coupling or the actual value coupling with extrapolation can be selected. When external encoders serve as the master value source, actual value coupling/actual value coupling with extrapolation (V3.0 and higher) can be selected. See the section titled Setpoint/actual value coupling (Page 2647).

Note

A drive axis cannot be used as the master value source for a gearing.

Processing cycle clock of the synchronous object

The processing cycle clock of the synchronous object and the processing cycle clock of the synchronized axis must be identical.

Note

If you change the processing cycle clock of the synchronized axis using the configuration screen form, the processing cycle clock of the synchronous object is changed automatically. If you change the processing cycle clock of the synchronized axis or of the synchronous object using the expert list, the processing cycle clock of the synchronous object or synchronized axis is *not* changed.

Recursive synchronous operation interconnection

A recursive synchronous operation interconnection is present when, in a single synchronous operation relationship, a synchronized axis is interconnected directly or indirectly again as a master value via other technology objects. A synchronized axis cannot act as a following axis to a master value and as a master value for the same axis simultaneously. Recursive synchronous operation interconnections can result if, for example, a synchronous operation relationship is to be switched over in the event of an error. Also refer to the section titled Error handling in the user program (Page 2756).

Units

The master and slave values are coupled without physical conversion *in the relevant parameterized units*. If, for example, the master axis is a linear axis and the following axis is a rotary axis, a length unit corresponds to an angular unit (for a 1:1 conversion ratio).

Modulo behavior

Different modulo ranges on the master value object and the following axis are taken into account on the synchronous object.

Camming with several cams

Several cams can be used in one camming operation. You can switch over to another cam dynamically using the **_enableCamming()** command in the user program.

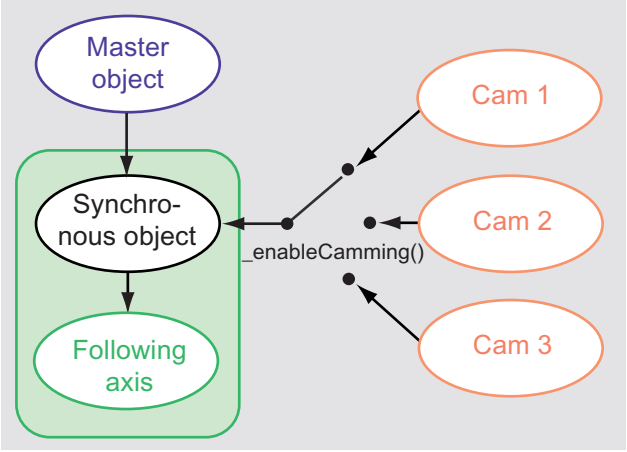


Figure 4-643 Example: Camming with several cams

Rules for interconnection

To recap, the following rules apply to synchronous operation:

- The synchronous object and following axis must be on the same runtime system (SIMOTION device).
- The master object and the following axis can be in different SIMOTION devices.
  In this case, we talk about distributed synchronous operation (Page 2758).
- The master object and synchronized axis may operate in different IPO cycles
  (see Overview of synchronous operation IPO - IPO_2 (Page 2824)).
- The synchronous object and the following axis are permanently assigned to each other during configuration.
- Up to two synchronous objects can be interconnected to one following axis.
- The master value object can be interconnected to several followings.
- The synchronous object can be interconnected to several master values and cams.
- A cam can be interconnected with several synchronous objects.

Superimposed synchronous operation

In superimposed synchronous operation, two synchronous objects can be connected to one following axis. The two synchronous operations superimpose one another (V3.0 and higher).

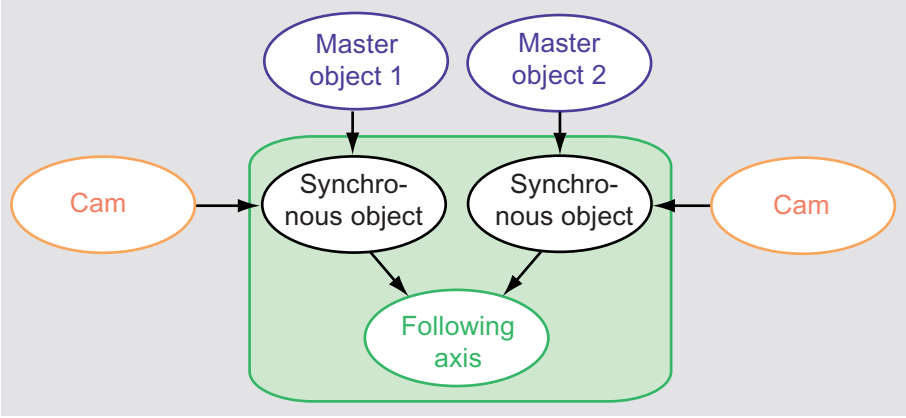


Figure 4-644 Example of superimposed synchronous operation

For additional information, see the section titled Superimposed synchronous operation (Page 2683).

See also

Synchronous Operation Configuration (Page 2721)

Synchronous Operation Programming/References (Page 2744)

4.7.2.2 Fundamentals of Synchronous Operation

Gearing

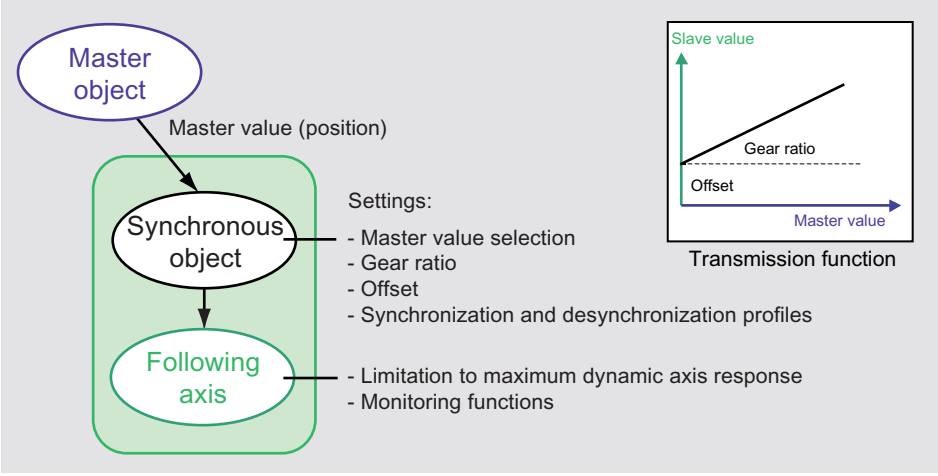


Figure 4-645 Gearing

Gearing is characterized by a linear transmission function between the master value source and the following axis/axes.

Slave value = Gear ratio x Master value + Offset

This gear ratio can be specified as the ratio of two decimal numbers (numerator/denominator) or as a rational number. A zero point offset can be included in the calculation. Absolute or relative gearing can be set using the gearingType parameter of the **_enableGearing()** command.

Absolute gearing

With absolute gearing (**gearingType=ABSOLUTE**), the synchronous operation occurs absolutely relative to the zero point of the master value and slave value, taking into account the gear ratio.

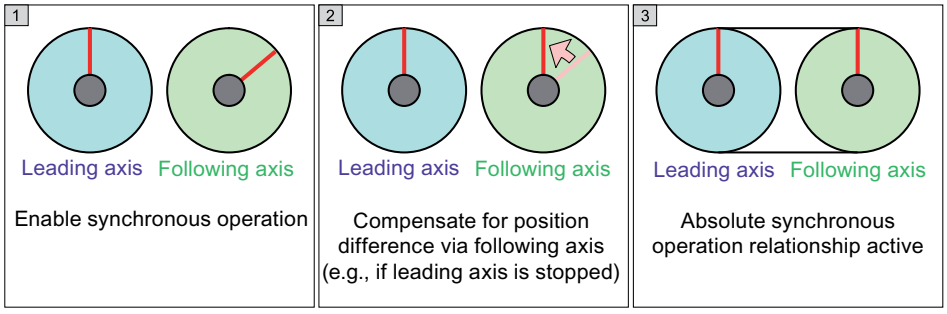


Figure 4-646 Sequence of the absolute gearing synchronization (simplified example)

A specified offset of the slave value is included. This offset is equal to zero, except when the synchronization criterion **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION** is set in the **syncPositionSlave** parameter, in which case an offset is specified.

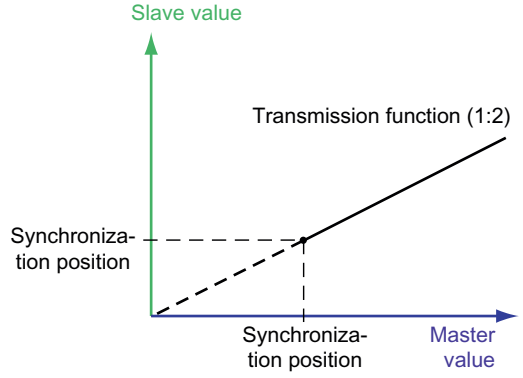


Figure 4-647 Absolute gearing without specification of the slave value position

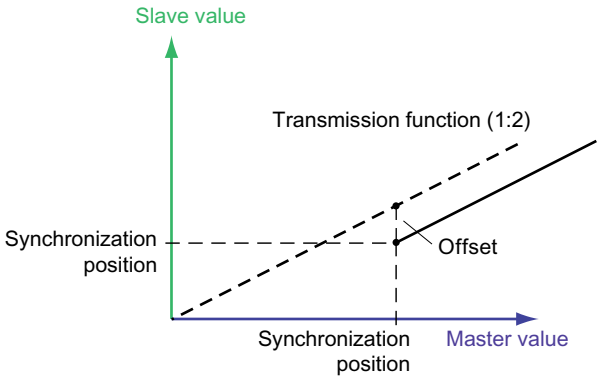


Figure 4-648 Absolute gearing with specification of the slave value position

Position differences on the slave value side are compensated for during synchronization. Modulo settings are taken into account.

Relative gearing

With relative gearing (**gearingType:=RELATIVE**), the synchronous operation occurs relative to the synchronization position on the master value and slave value sides.

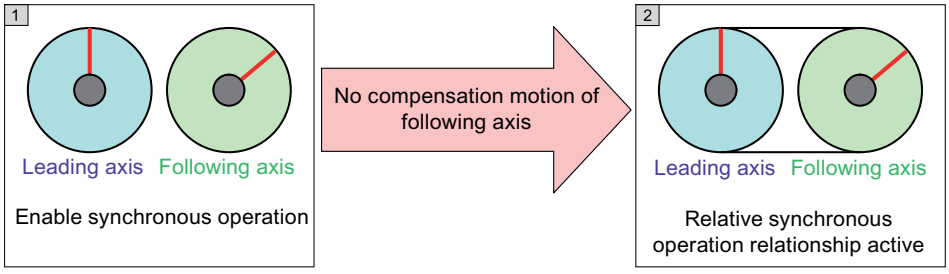


Figure 4-649 Sequence of the relative gearing synchronization (simplified example)

The offset is determined implicitly in the transmission function:

- If programmed without a specified offset: The offset is derived from the current position of the following axis at the start of synchronization and from an offset derived implicitly during synchronization if the axis is driven to a velocity (and acceleration) derived from the gear ratio.
- If programmed with a specified offset: Using the synchronization criterion setting **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION**, the offset is determined from the current following axis position at the start of synchronization and the offset programmed in **syncPositionSlave**.

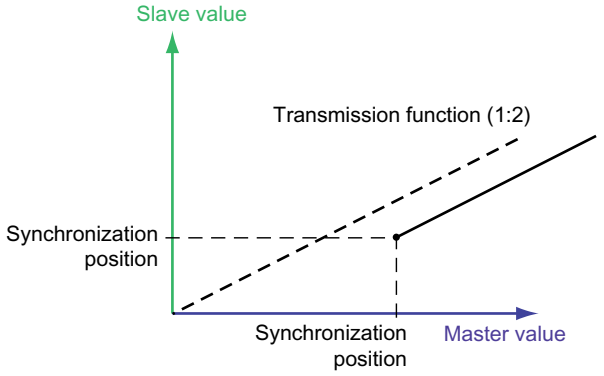


Figure 4-650 Relative gearing without offset

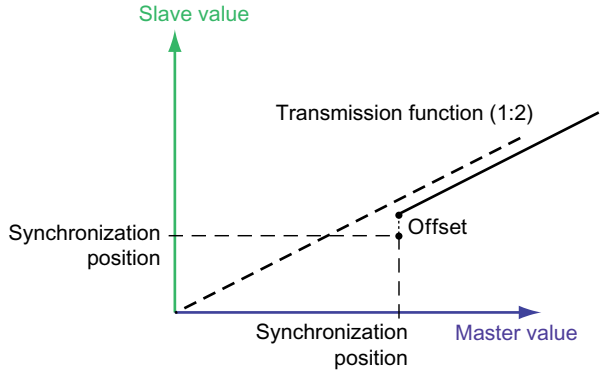


Figure 4-651 Relative gearing with offset

From the point at which the status becomes "synchronous", relative gearing is also in positional synchronism. In other words, the offset remains constant in the transmission function from this point onwards.

Gear ratio

The gear ratio is used to define the transmission function of the gearing between the master value and slave value. The gear ratio corresponds to the slope of the transmission function. It can entered as either a fraction or a floating-point number using the **gearingMode** parameter of the **_enableGearing()** command.

- As a fraction (**gearingMode:=GEARING_WITH_FRACTION**)
  The gear ratio is specified as a fraction (slave value difference/master value difference) using the following function parameters:
  - **gearingRatioType**: Type of gear ratio specification (directly or via replacement values)
  - **gearingNumerator**: Value for direct specification of the gear ratio numerator
  - **gearingDenominator**: Value for direct specification of the gear ratio denominator
- As a floating-point number (**gearingMode=GEARING_WITH_RATIO**)
  The gear ratio is specified as a floating-point number using the following function parameters:
  - **gearingRatioType**: Type of gear ratio specification (directly or via replacement values)
  - **gearingRatio**: Value for direct specification of the gear ratio as a floating-point number
    Disadvantage: Gear ratios such as 1/3 ≈ 0.333 are subject to rounding errors.

The long-term effect is to be taken into account with modulo axes. If master and slave axes are configured as modulo axes, to ensure the long-term stability of the gear, the gear ratio is preferably to be entered as a nominator/denominator ratio. If this is not possible, a LREAL value with corresponding decimal places should be used.

Direction of gearing

The gear ratio can be set in the same direction or in the opposite direction (corresponding to a negative gear ratio) using the **direction** parameter of the **_enableGearing** command.

- For **POSITIVE**, traversal is made in the same direction as the master values, this means that the axes run in the same direction.
- For **NEGATIVE**, traversal is made in the opposite to master values, this means that the axes run in the opposite direction.
- With **CURRENT**, the direction of the current slave values is retained; along with the direction of the master value; this results in coupling in the same or opposite direction, which is then maintained for the entire command execution time (that is, if the master value direction changes, then the slave direction changes as well).
- **REVERSE** means movement in the inverse direction of the slave values.

If the slave values are at a standstill at the point at which the command is activated, the following conversion is performed: **CURRENT** becomes **POSITIVE** and **REVERSE** becomes **NEGATIVE**.

Change in the offset

The **activationMode** parameter of the **_setGearingOffset()** command specifies when the offset takes effect (as of V3.1).

The changeover applies as follows:

- *For the next synchronous operation and all subsequent synchronous operations* if **DEFAULT_VALUE** is set
- *For the current synchronous operation only* if **ACTUAL_VALUE** is set
- *For the current synchronous operation and all subsequent synchronous operations* if **ACTUAL_AND_DEFAULT_VALUE** is set

Note the following:

- If the synchronization operation of the **_enableGearing()** command is not yet active, the current offset is carried out without compensation, that is, it is figured in directly.
- If the **_setGearingOffset()** command is programmed to current values during synchronization, the offset does not take effect until after synchronization. A compensating movement takes place.

Apply offset as superimposition

The **dynamicReference** parameter of the **_setGearingOffset** command can be used to specify whether the dynamic parameters refer to the total motion or the motion difference (V3.2 and higher).

- **TOTAL_MOVE**: Dynamic response parameters refer to the total motion. (Default setting)
  The transition is determined entirely on the basis of the offset values and the dynamic response parameters.
- **OFFSET_MOVE**: Dynamic response parameters refer to the motion difference.
  The transition is determined on the basis of the current synchronous operation definition as superimposed motion with the specified dynamic values.

Note

With a constant master value velocity, the dynamic transitions have a similar form and differ as a result of the dynamic response parameters that act differently.

See also

Example of applying offset as superimposition (Page 2712)

Velocity gearing

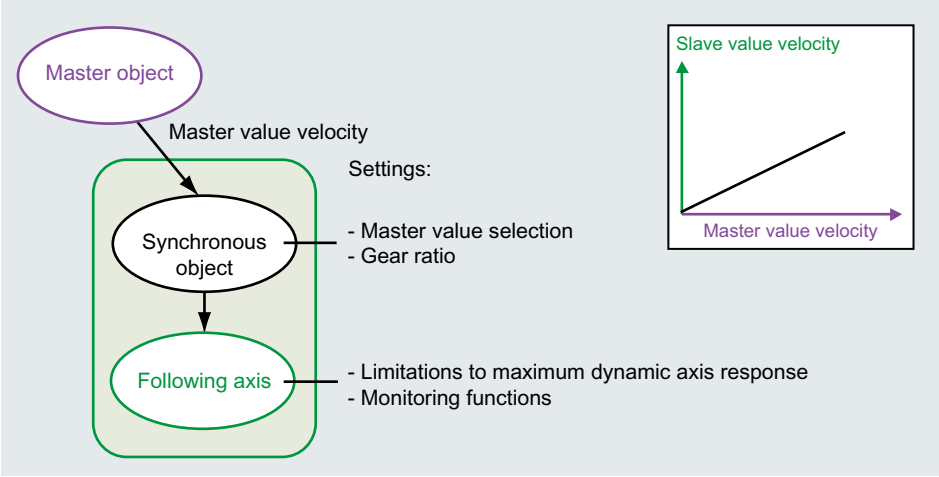


Figure 4-652 Velocity gearing

In contrast to gearing or camming, which relate to the position of an axis, **synchronous velocity operation** (V3.1 and higher) relates to the *velocity of an axis*. A velocity setpoint is calculated for the following axis. After activation, the axis travels immediately at the specified acceleration to the synchronous operation velocity. A linear transmission function is implemented.

The gear ratio can be specified as a positive floating-point number.

The gear ratio can be set in the same direction or in the opposite direction (corresponding to a negative gear ratio) using the **direction** parameter of the **_enableVelocityGearing()** command.

- **POSITIVE** means that the axes are running in the same direction.
- **NEGATIVE** means that the axes are running in opposite directions.

See also

Substitution of velocity gearing with absolute synchronous operation (Page 2717)

Camming

Description

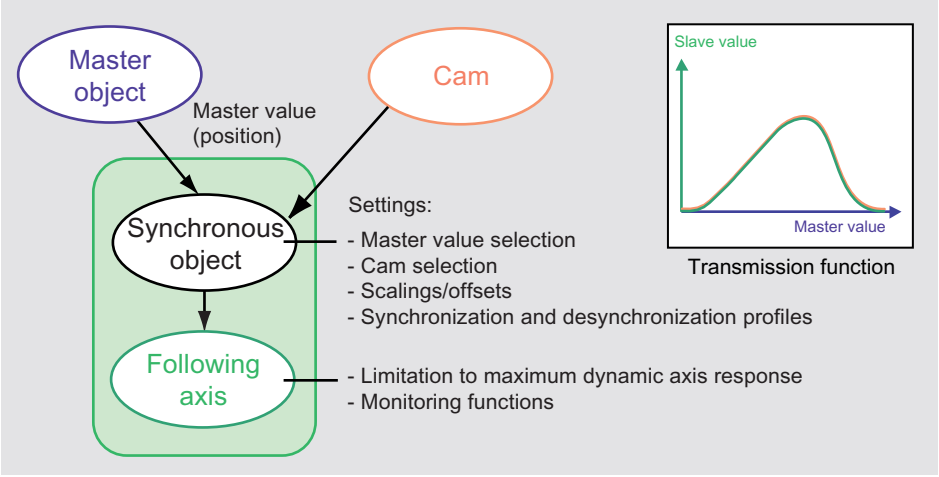


Figure 4-653 Camming

With **camming**, a non-linear transmission function between the master value position and following axis position is implemented using a **cam** .

See Cam, Definition (Page 2831)

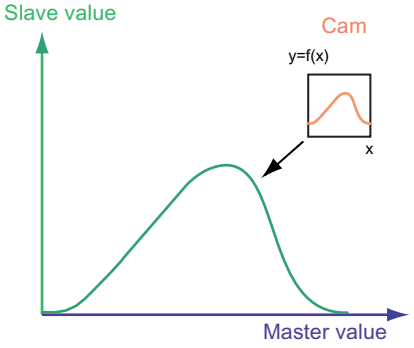
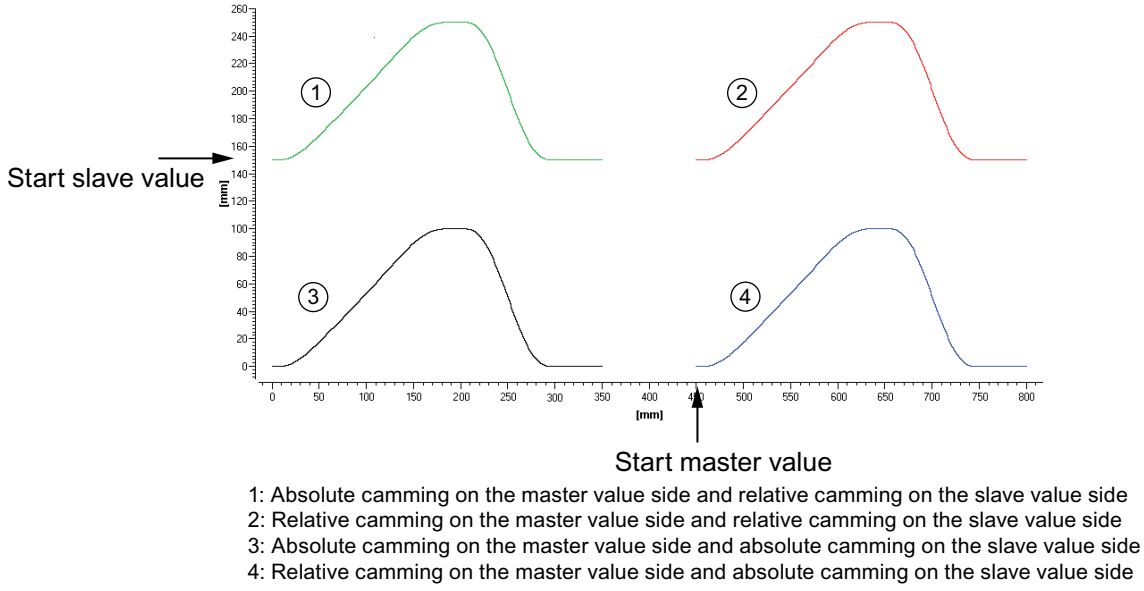


Figure 4-654 Example of transmission function for camming

The cam can be applied as an absolute cam or as a relative cam in both the definition range (master values) and the value range (slave values). The setting is made for the master values in the masterMode parameter and for the slave values in the slaveMode parameter of the _enableCamming() command.

The figure below compares the relationship between the master value and slave value with the following supplementary conditions:

- Same cam: here, with definition range {0.0, 300.0} and value range {0.0, 100.0}
- Same initial value of following axis: here, 150 mm
- Same initial value of master value: here, 450 mm (master value is modulo value 0 to 1000 mm)



1: Absolute camming on the master value side and relative camming on the slave value side
2: Relative camming on the master value side and relative camming on the slave value side
3: Absolute camming on the master value side and absolute camming on the slave value side
4: Relative camming on the master value side and absolute camming on the slave value side

Figure 4-655 Possible combinations of absolute/relative camming on the master/slave value side

Absolute camming on the slave value side

Absolute camming on the slave value side is set in **slave Mode:=ABSOLUTE**. With absolute camming on the slave value side, the slave values are taken directly from the value range of the cam. The offset on the slave value side is equal to zero, except when the synchronization criterion **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION** is set in the syncPositionSlave parameter, in which case an offset is specified. (Case 3 and 4 in Figure **Possible combinations of absolute/relative camming on the master/slave value side**).

Relative camming on the slave value side

Relative camming on the slave value side is set in **slaveMode:=RELATIVE**. With relative camming in slave mode, the initial value of the cam is offset to the slave value position at the start of synchronization (case 1 and 2 in Figure **Possible combinations of absolute/relative camming on the master/slave value side**).

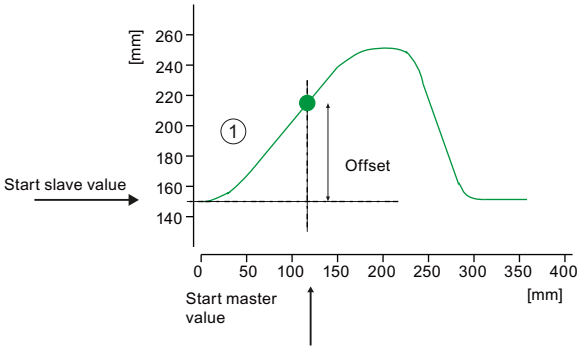
The offset on the slave value side is determined as follows:

- If programming is performed without a specified offset, the offset is determined from the offset of the cam initial value to the slave value position at the start of synchronization
- If programming is performed with a specified offset in the synchronization criterion setting **ON_MASTER_AND_SLAVE_POSITION** or **IMMEDIATELY_AND_SLAVE_POSITION**, the offset is determined from the offset of the cam initial value to the slave value position at the start of synchronization plus the offset programmed in the **syncPositionSlave** parameter.

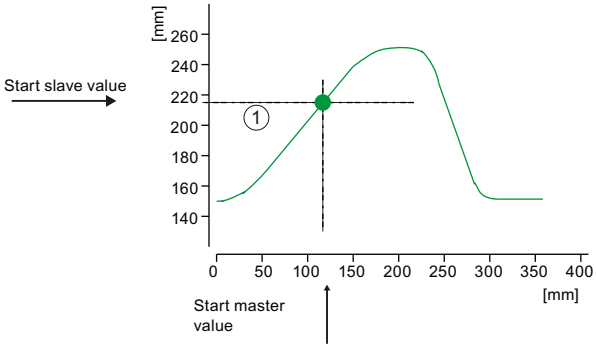
From the point at which the status becomes "synchronous", the offset on the slave value side remains constant in the transmission function.

In V4.2 and higher, there is new configuration data element **syncingMotion.camReferenceBySlaveModeRelative** with the following settings:

- Effectiveness in the case of relative camming on the slave value side and
  - on the absolute master value side with and without master value offset or
  - relative master value side with master value offset
- **BEGIN_OF_CAM** is based on the current following axis position to the initial cam value. During the synchronization, the following axis also covers the path differential between the start of the cam and the cam start value if this is not offset by a following value offset.



- **POSITION_AT_START_OF_CAMMING** is based on the current following axis position to the starting point of the synchronization within the cam. The difference between the slave-side starting point and the initial cam value does not have to be specified in the slave-value offset.



- In projects upgraded from < V4.2 to >= V4.2, the configuration data is set to **COMPATIBILITY_MODE**. This corresponds to **SYNCPOSITION_AT_START_OF_CAMMING** for synchronization at a standstill and **BEGIN_OF_CAM** with moving following axis or moving master value.
- When creating a new project as of V4.2, **syncingMotion.camReferenceBySlaveModeRelative:=BEGIN_OF_CAM** is set.

Absolute camming on the master value side

Absolute camming on the master value side is set in **masterMode:=ABSOLUTE**.
With **ABSOLUTE,** the master values are used as absolute values in the definition range of the cam.

Relative camming on the master value side

Relative camming on the master value side is set in **masterMode:=RELATIVE**. With relative camming on the master value side, the synchronization position on the master value side is assigned to the position within the cam definition range specified in the **camStartPositionMaster** parameter. The offset on the master value side is determined from the difference between the synchronization position on the master value side and the value specified in the **camStartPositionMaster** parameter.

If the position specified in **camStartPositionMaster** is not within the definition range of the cam, alarm "40017 Cam starting point is outside the definition range" is generated. From the point at which the status becomes "synchronous", the offset on the master value side remains constant in the transmission function.

Loading a new camming

A new cam is loaded by a recalling an **_enableCamming** command. Using the **synchronizingMode** parameter, the synchronization criterion is specified (e.g. at the end of the current camming with **AT_THE_END_OF_CAM_CYCLE**).

If the transition from the old cam to the new cam is already smooth, the synchronization parameters should be selected in such a way that either the synchronization length is very small for master value reference or the dynamics are very high for the forward time reference or, alternatively, use the synchronization profile with backward time reference.

Non-cyclic/cyclic cam application

The **cammingMode** parameter of the **_enableCamming()** command can be used to set the cam for either a non-cyclic application or a cyclic application.

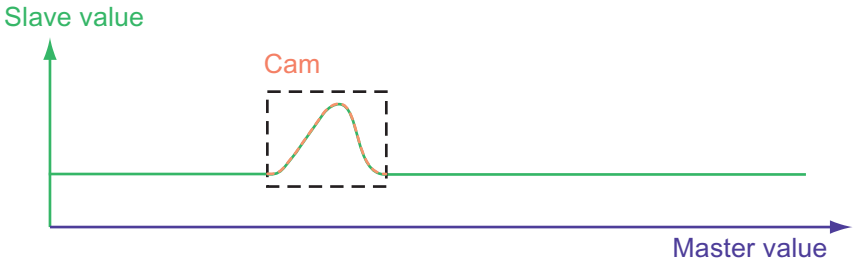


Figure 4-656 Non-cyclic cam application

- Non-cyclic (**NOCYCLIC**) means that the cam is applied exactly once in the defined master value range. When the end point or starting point of the cam is reached, the cam terminates itself.
  If the master value range is run through again either in the same direction or, after reversing, in the opposite direction, the cam is not applied again.

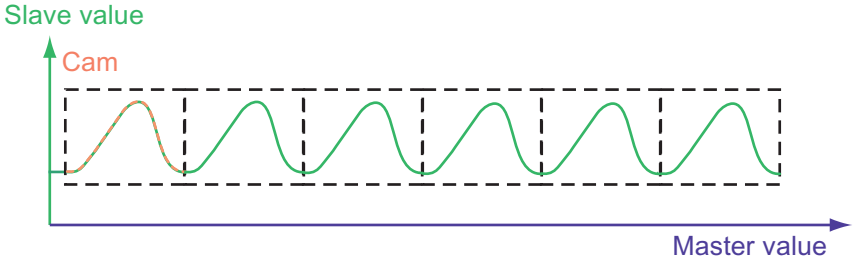


Figure 4-657 Cyclic cam application

- With the cyclic (**CYCLIC**) application of a cam, the definition range of the cam is mapped cyclically onto the master values.
  If the master values reverse, the cam is also continued cyclically beyond the original starting point.

Cyclic application of a cam with absolute synchronous operation on the slave value side

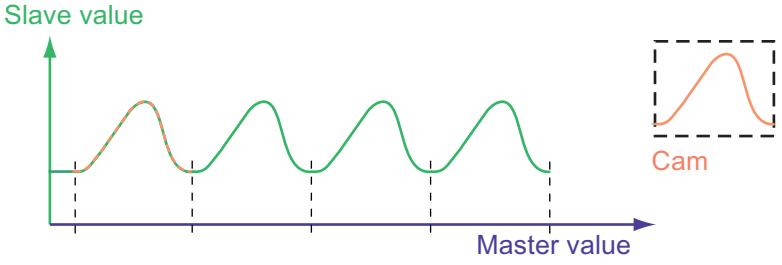


Figure 4-658 Cyclic absolute cam application with equal initial and end values on the slave value side

- If the function values of the cam are equal at the start and end of the definition range of the cam, the motion can be continued smoothly.
  This produces a periodic motion.

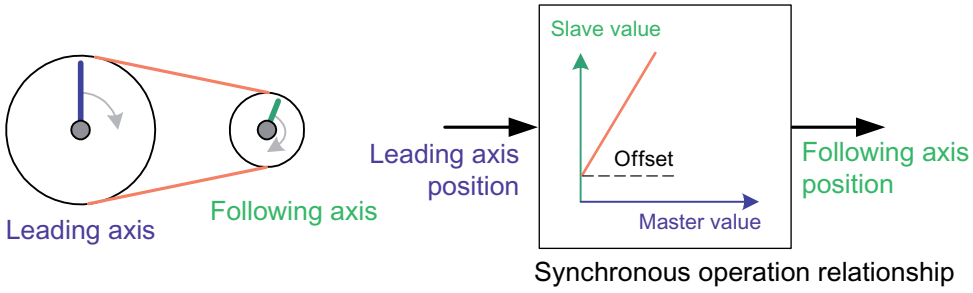


Figure 4-659 Cyclic absolute cam application with unequal start and end values on the slave value side

- If the function values of the cam are not equal at the start and end of its definition range, this results in a discontinuity in the position.
  This is limited on the following axis to the maximum dynamic values.

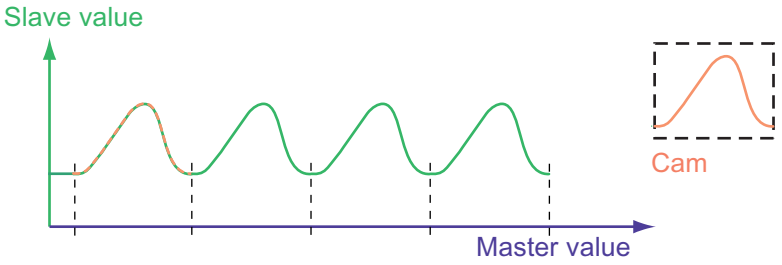


Figure 4-660 Example of cyclic cam application with identical start and end values

- If the function value of the cam is *not* identical, or equal in terms of a modulo relationship, at the start and end of its definition range, the new starting point of the cam is the end point of the executed cam.

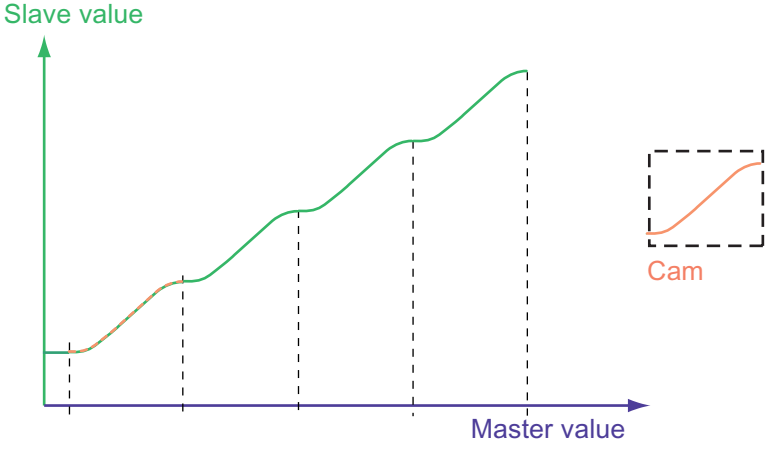


Figure 4-661 Example of cyclic cam application with different start and end values - relative

Cam direction

The **direction** parameter of the **_enableCamming** command can be used to set the cam in a positive or negative direction.

- **POSITIVE** means in the same direction. Increasing master values correspond to increasing values in the definition range of the cam, and vice versa.

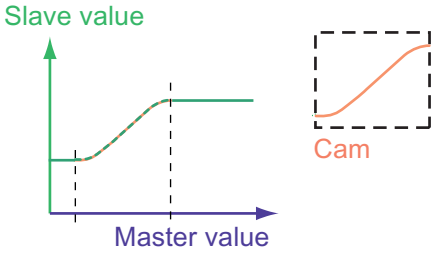


Figure 4-662 Positive cam application (POSITIVE)

- **NEGATIVE** means in the opposite direction. Decreasing master values correspond to increasing values in the cam definition range, and vice versa.
  The cam is mirrored at the center of its definition range.

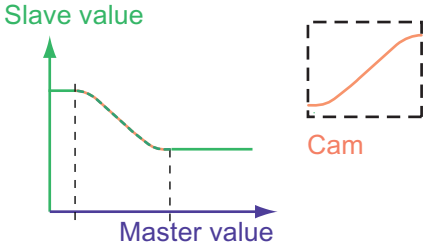


Figure 4-663 Negative cam application (NEGATIVE)

Example application:
The aim is to use the same cam for deceleration as for acceleration, but in the opposite direction.

Correction of camming motions

Synchronous motions can be corrected by changing the scaling and offset of the master value and the slave value.

Other options include:

- Offset and scaling on the cam itself
- Superimposed motions on the following axis
- On-the-fly setting of the reference point on the leading value source and the following axis

Scaling and offset

The *scaling* and *offset* can be specified on the *synchronous object* for camming on both the master value side and slave value side.

The slave value is determined from the master value using the following equation:

$$\text{Slave value} = F_{CAM}\left(\frac{\text{Master value} - \text{Offset}_{\text{master value}}}{\text{Scaling}_{\text{master value}}}\right) \bullet \text{Scaling}_{\text{slave value}} + \text{Offset}_{\text{slave value}}$$

Figure 4-664 Equation for scale and offset on the camming

See also Example of offset and scaling on the synchronous object (Page 2709)

Scaling/offset on the cam

In addition to the option of the scaling/offset on the synchronous object, a scaling/offset is also possible on the cam. This enables a cam to be custom-adjusted in its definition range and value range.

See Scaling and offset (Page 2833)

Changing the scaling and offset

The **_setCammingScale()** and **_setCammingOffset()** commands can be used to switch the scaling and offset within active, cyclic camming. The **activationMode** parameter determines when these take effect:

- For the next camming operation and all subsequent operations if **DEFAULT_VALUE** is set.
- For the current camming operation, only if **ACTUAL_VALUE** is set.
- For the current camming operation and all subsequent operations if **ACTUAL_AND_DEFAULT_VALUE** is set

Note the following:

- If synchronization of the **_enableCamming()** command is not yet active, the current scaling/offset is carried out without compensation; that is, it is included directly.
- If the **_setCammingScale()**/**_setCammingOffset()** command is programmed with new values during synchronization (using the **ACTUAL_VALUE** setting), the scaling/offset only becomes active after synchronization. A compensating movement takes place.

Effectiveness of scaling and offset

The **scaleSpecification**/**offsetSpecification** parameter of the **_setCammingScale()** or **_setCammingOffset()** command is used to program the effectiveness of a new scaling or offset procedure.

- With immediate effect (**IMMEDIATELY**)
- At the start of a new cycle for a cyclic cam application (**NEXT_CAM_CYCLE**)

Comment: If a **_setCammingScale()**/**_setCammingOffset()** command is canceled during the compensating movement due to another **_setCammingScale()**/**_setCammingOffset()** command with **NEXT_CAM_CYCLE**, compensation is canceled and a jump in the setpoints can occur. The new command is enabled at the beginning of the new cam cycle.

Examples

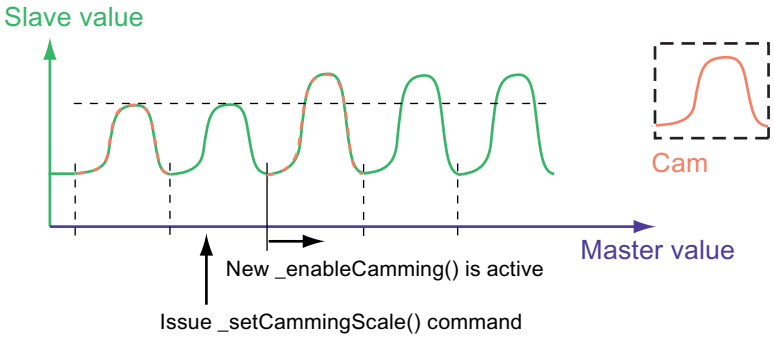


Figure 4-665 Example of switchover from scaling during cyclic synchronous operation; setting: activationMode:=DEFAULT_VALUE; effective: scaleSpecification:=NEXT_CAM_CYCLE

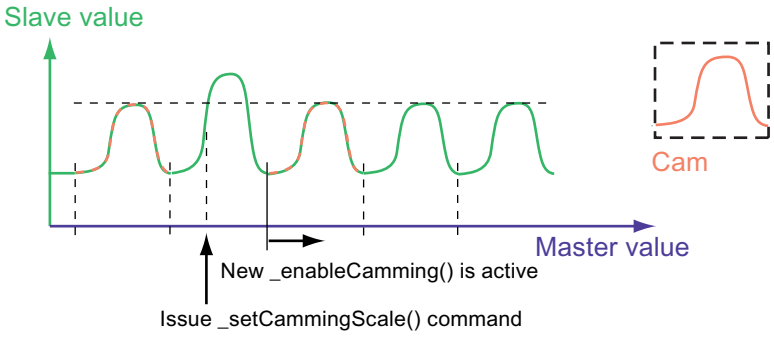


Figure 4-666 Example of switchover from scaling during cyclic synchronous operation; setting: activationMode:=ACTUAL_VALUE; effective: scaleSpecification:=IMMEDIATELY

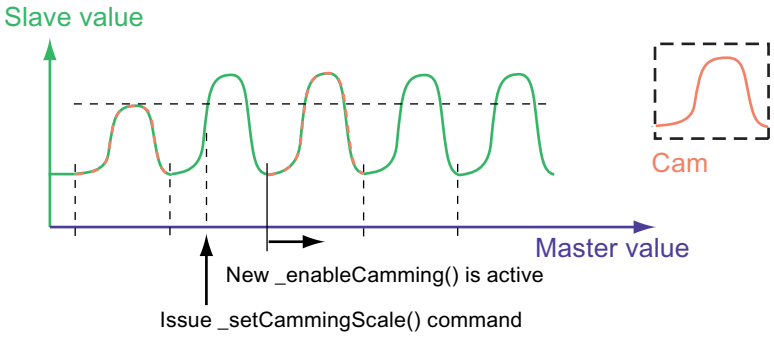


Figure 4-667 Example of switchover of scaling and offset during cyclic synchronous operation, ACTUAL_AND_DEFAULT_VALUE setting with effectiveness IMMEDIATELY

Applying scaling/offset as superimposition

The **dynamicReference** parameter of the **_setCammingScale()** or **_setCammingOffset()** command can be used to specify whether the dynamic parameters refer to the total motion or the motion difference (V3.2 and higher).

See **Apply offset as superimposition** at Gearing (Page 2631).

See also

Display of the synchronous position (Page 2673)

Setpoint/actual value coupling

Overview

When an axis is used as a master value object, the following can be configured for the synchronous operation:

- **Setpoint coupling**: The setpoint of the axis is used as the master value for the following axis. This is advantageous if the control specifies the setpoints for both the master axis and following axis, and the axes are to behave synchronously in relation to one another. In general, setpoint coupling is recommended for purposes of signal quality.
- **Actual value coupling with extrapolation** (V3.0 and higher): The actual value of an axis is used as the master value for the following axis.
  The actual value can be extrapolated in order to compensate for delays caused by actual value acquisition, actual/master value processing in the control, and the dynamic follow-up response of the following axis. Since the actual values are equal to the setpoints for the virtual axis, an extrapolated setpoint can be set.

When an external encoder is used as a master value object, the following can be configured for the synchronous operation:

- **Actual value coupling**: The actual value of an external encoder is used as the master value for the following axis.
- **Actual value coupling with extrapolation** (V3.0 and higher): The actual value can be extrapolated in order to compensate for delay times caused by actual value acquisition, actual/master value processing in the control, and the dynamic follow-up response of the following axis.

A tolerance window with respect to the actual value behavior can be specified for the actual value coupling.

Note

If the actual values/setpoints are required to be equal during a synchronous operation, the same Ti (actual value acquisition)/To (setpoint acceptance) times must be adopted for all drive units used (e.g. SINAMICS_Integrated, CU320).

See also

Actual value coupling with tolerance window (Page 2651)

Actual value coupling with extrapolation

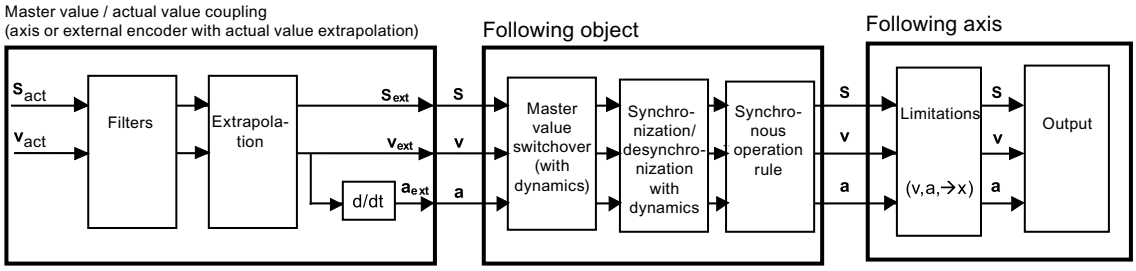


Figure 4-668 Principle of actual value coupling (overview)

For a synchronized group with actual value coupling (e.g. master value is the actual encoder value of an axis or an external encoder), the associated principle means delay times result because of bus communication, system cycle clocks and clock-pulse scaling, fine interpolation, position setpoint filters, and controller settings. These times can be compensated for using an extrapolation.

Extrapolation means you know the history and are looking into the future (extrapolation time).

The extrapolation time should be as short as possible for dynamic master value changes. An IPO : Servo pulse duty factor of 1:1 is good.

If an actual encoder value is assumed as the master value, it is useful to extrapolate the measured actual value for the synchronous operation in order to compensate for dead times that result within the system when acquiring actual values, e.g. due to bus communication and system processing times.

The extrapolation is set on the leading axis or on the external encoder.

Noisy sensor signals result in large velocity jumps which affect the extrapolation. These can be reduced or compensated by using suitable filter settings.

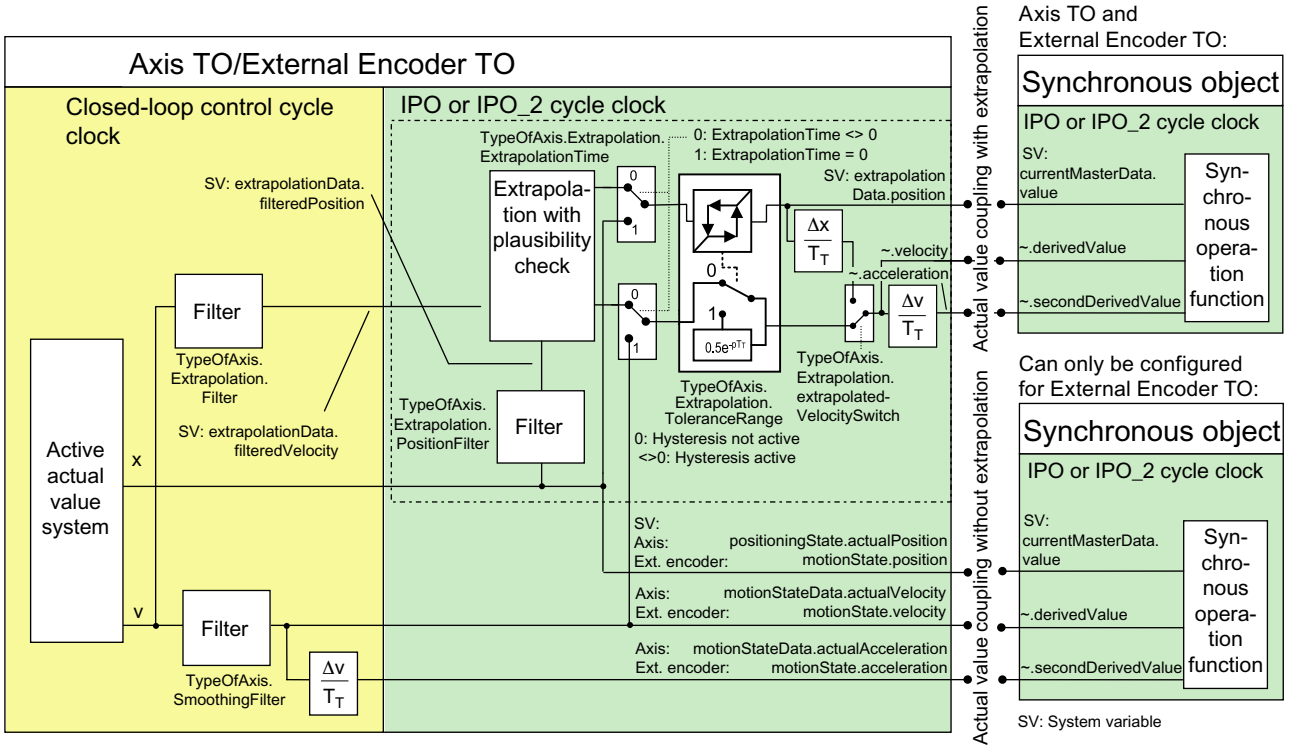


Figure 4-669 Actual value coupling with extrapolation for the Axis TO or External Encoder TO

You can find an overview of the actual value coupling with and without extrapolation in the axis configuration in the SCOUT signal flow dialog (project navigator, select Axis TO > **Signal flow > Extrapolation**).

Filtering of actual position

The actual position value for the synchronous operation can be filtered separately for the extrapolation using a PT2 filter (as of V4.1). The filter for the actual position value of the axis is set using the **typeOfAxis.extrapolation.positionFilter.T1** and **typeOfAxis.extrapolation.positionFilter.T2** configuration data. The filter acts on the actual position for the extrapolation. The velocity for the extrapolation is taken over from the actual values of the axis or External Encoder before application of the smoothing filter (**typeOfAxis.smoothingFilter**).

Filtering of actual velocity

The position is extrapolated based on the filtered or averaged velocity value.

We recommend setting the velocity filter (Extrapolation.Filter) first and then also using the position filter if the result is not satisfactory.

The position filter times are an additional factor to be taken into account in the extrapolation time.

The velocity filter (Extrapolation.Filter) does not affect the extrapolation time, but does have influence during dynamic changes to master values (due to delayed values). See also the *TO axis, Electric/hydraulic, External encoder* Manual.

- **TypeofAxis.Extrapolation.filter.timeConstant:** Time used for averaging or time constant for filtering

Extrapolation of actual velocity and actual position

These delay times can be compensated using an extrapolation.

- **TypeofAxis.Extrapolation.extrapolationTime:** Time definition for extrapolation

Extrapolation is not performed if 0.0 is specified. The extrapolated values (position and velocity) can be monitored (**extrapolationData** system variable). The extrapolation compensates for the local delays that result from use of the actual value instead of the setpoint.

Note

Extreme care must be taken when changing the extrapolation time to the runtime; otherwise knocking could result in the machine.

Note

SIMOTION contains utilities & applications which are included in the scope of delivery of SIMOTION SCOUT, a tool to assist in calculating the extrapolation times.

Switch for the velocity master value during master value extrapolation

The **TypeofAxis.Extrapolation.extrapolatedVelocitySwitch** configuration data element is used to select whether the velocity master value is generated from the extrapolated or filtered position master value using differentiation or if the extrapolated velocity master value is used.

Display

The extrapolated and filtered values are indicated in the following system variables:

- **sensorData[n].position**
- **sensorData[n].velocity**
- **sensorData[n].acceleration**

The system variables for **sensorData** are calculated in the position control cycle clock.

The actual axis value that is active for closed-loop control, the IPO cycle, and master value coupling

is displayed in the following variables:

- **positioningState.actualPosition**
- **motionStateData.actualVelocity**
- **motionStateData.actualAcceleration**

The system variables for **positioningState** and **motionState** are calculated in the IPO cycle.

These actual values are the reference for the output cam calculation in the IPO cycle, for the

actual value connection with the external encoder without extrapolation, for the actual value reference in the IPO cycle,

e.g. for profiles relating to actual position.

Reduction in reaction times/dead times

The **Execution.executionLevel:=SERVO** setting on the master value object, e.g. External Encoder technology object, can be configured in the Synchronous Object technology object and Following Axis technology object to enable execution of the IPO system component of the master value, synchronous operation, and axis in the Servo following actual value acquisition.

For further information, refer to the **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual, "Motion execution/interpolator".

Transferring the actual velocity from the drive (V4.2 and higher)

With the
setting
typeOfAxis.numberOfEncoders.encoder_n.encoderValueType:=POSITION_AND_PROFIDRIVE_NIST_B, you have the option of converting the speed of rotation transferred in **PROFIdrive NIST_B** to a velocity and applying this value as the actual velocity of the encoder/sensor. In this case, the actual position of the sensor does not need to be differentiated to derive the actual velocity.

With the setting **typeofAxis.numberOfEncoders.encoder_n.encoderValueType:= POSITION_AND_DIRECT_NIST**, a speed of rotation transferred in the I/O area and normalized as NIST_B is taken as the actual value and converted to an actual velocity. In this case, 4000H corresponds to 100%. The address is set in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.logAddress**, and the reference value is set in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.referenceValue**. With encoders with nact evaluation, the speed determined by the encoder and the resulting velocity can be accepted by the encoder. In this case, the actual position of the sensor does not need to be differentiated to derive the actual velocity. Two methods of transmission are available:

- Transmission in the PROFIdrive message frame
- Transmission in the I/O area

See also *section Actual value acquisition / Actual value system in the Manual TO Axis electric/ hydraulic, External Encoder*.

Actual value coupling with tolerance window

If the master value is superimposed with high-frequency noise signals that cannot be followed by the synchronous operation, this can cause the dynamic response boundaries to be exceeded or the master value to briefly change directions during synchronization.

In the **typeOfAxis.extrapolation.toleranceRange** configuration data element *on the master axis* or external encoder, a tolerance window can be set around the actual position (V3.1 and higher), to prevent, for example, the dynamic limits from being exceeded on the following axis in the case of a master value with high-frequency disturbances, or direction changes during synchronization.

Synchronization

In order for the following axis to follow the master value according to the transmission function, the following axis must first be synchronized to the master value.

The type of synchronization is determined from several assignable parameters/settings:

- The **synchronization criterion/synchronization position**, which corresponds to the setting specified in the **synchronizingMode** parameter; the synchronization position on the master value side and/or the synchronization position on the slave value side are directly specified here or are derived from the synchronization criterion and, if necessary, the transmission function
- The **synchronization direction**, the motion direction of the slave values during synchronization; can be set in the **synchronizingDirection** parameter
- **Position of synchronization range relative to synchronization position:** Leading, trailing, or symmetrical synchronization; can be set in the **syncPositionReference** parameter
- the **reference of the synchronization profile**; can be set in the **syncProfileReference**
  - **Synchronization over a specifiable master value distance**
    The synchronization length over the master value is specified in the synchronization command.
  - **Synchronization profile via specifiable dynamic response parameters (time reference)**
    The dynamic response parameters are specified in the synchronization command.

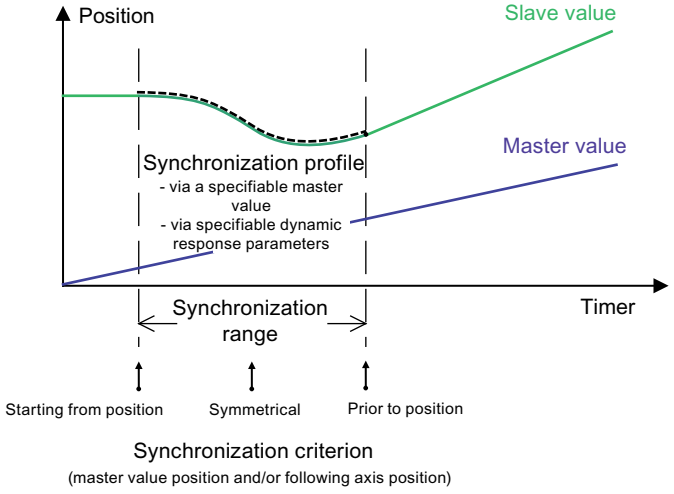


Figure 4-670 Parameters for synchronization

Properties	Synchronization via a specifiable master value distance	Synchronization profile based on specifiable dynamic response parameters, leading synchronization	Synchronization profile based on specifiable dynamic response parameters, trailing synchronization
Dynamic response properties			
• Constant velocity synchronization profile	Yes	Yes	Yes

Properties	Synchronization via a specifiable master value distance	Synchronization profile based on specifiable dynamic response parameters, leading synchronization	Synchronization profile based on specifiable dynamic response parameters, trailing synchronization
• Constant acceleration synchronization profile	No	With SMOOTH velocity profile setting	With SMOOTH velocity profile setting
Adherence to dynamic response parameters (without limiting functions on the following axis side)	No User can influence the dynamic response via the synchronization length	With master value and constant velocity, otherwise master value dynamic response is superimposed	Yes
Dynamic response can be adapted to the master value dynamic response	Indirectly	With dynamicAdaption setting	With dynamicAdaption setting
Applicability to stationary master value			
• If following axis is at a standstill	Conditional Following axis must already be at the synchronous position, for example, with relative gearing	Conditional Following axis must already be at the synchronous position, for example, with relative gearing	Yes
• With moved following axis	No	No	Yes
Applicability to master value with constant velocity			
• If following axis is at a standstill	Yes	Yes	Yes
• With moved following axis	Yes	Yes	Yes
Applicability to master value with non-constant velocity			
• Master value with constant acceleration / deceleration	Yes Superimposition of master value dynamic response	Yes Superimposition of master value dynamic response	Conditional With extended look-ahead or dynamic response of synchronization >> master value dynamic response
• Modified master value dynamic response or faulty/ noisy master value signal	Yes Superimposition of master value dynamic response	Yes Superimposition of master value dynamic response	No Exception: Dynamic response of synchronization >> master value dynamic response
Synchronization properties			

Properties	Synchronization via a specifiable master value distance	Synchronization profile based on specifiable dynamic response parameters, leading synchronization	Synchronization profile based on specifiable dynamic response parameters, trailing synchronization
• Synchronism reached after starting the synchronization	Yes Exception: master value changes motion direction	Yes Exception: master value changes motion direction	Conditional No, if master value dynamic response > resulting dynamic response of synchronization or varying master value dynamic response; see above
• Specification of synchronous position after starting the synchronization	Supported	Supported	No

Properties of different synchronization options

Synchronization criterion

Synchronization criterion/synchronization position

The **synchronization criterion** can be set for synchronization using the synchronizingMode parameter of the **_enableGearing()**/**_enableCamming()** or **_disableGearing()**/ **_disableCamming()** command as outlined below. Synchronization can be performed over several modulo ranges of the master value or slave value.

Synchronization on current master value position without specification of an offset on the slave value side

The current master value position is the synchronization criterion and the synchronization position on the master value side. The synchronization criterion is set with **synchronizingMode:=IMMEDIATELY**. The **syncPositionMaster** parameter is not active. An offset on the slave value side is not specified, and the **syncPositionSlave** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active. Synchronization starts immediately. Synchronization occurs subsequently.
The **syncPositionReference** parameter is not active.

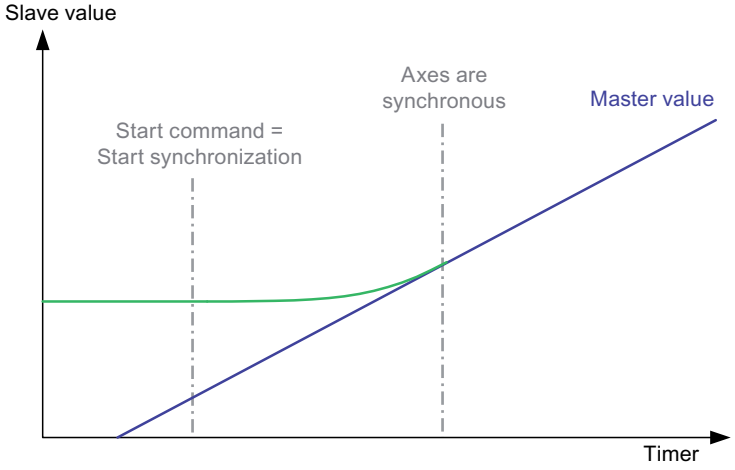


Figure 4-671 Example of synchronization - immediately active, trailing synchronization, absolute without offset, ratio 1:1

Synchronization on current master value position with specification of an offset on the slave value side

The current master value position is the synchronization criterion, and an offset on the slave value side is specified. The synchronization criterion is set with **synchronizingMode:=IMMEDIATELY_AND_SLAVE_POSITION**. The synchronization position on the master value side is the current master value position. The **syncPositionMaster** parameter is not active. The offset on the slave value side is specified in the **syncPositionSlave** parameter. With relative camming on the master value side, the **camStartPosition** parameter is active. Synchronization starts immediately. Synchronization occurs subsequently.
The **syncPositionReference** parameter is not active.

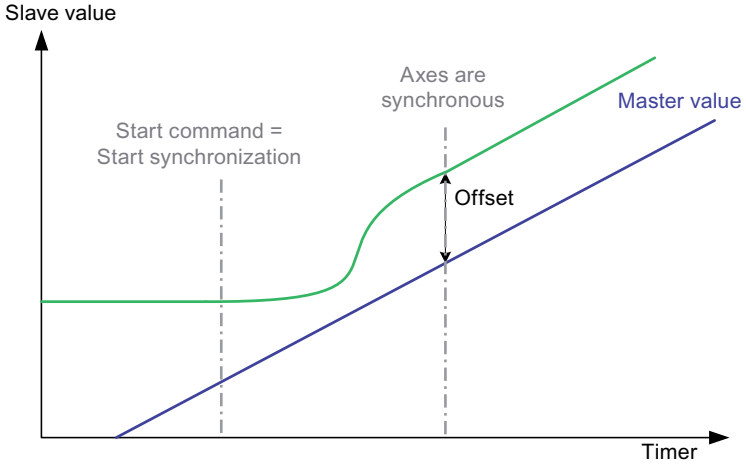


Figure 4-672 Example of synchronization - immediately active, trailing synchronization, absolute and offset on following axis position, ratio 1:1

Synchronization on specified master value position without specification of an offset on the slave value side

The specified master value position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_MASTER_POSITION**. The synchronization position on the master value side is set in the **syncPositionMaster** parameter. An offset on the slave value side is not specified, and the **syncPositionSlave** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active.
The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

Regarding the start of synchronization, see Position of synchronization range relative to synchronization position (Page 2660).

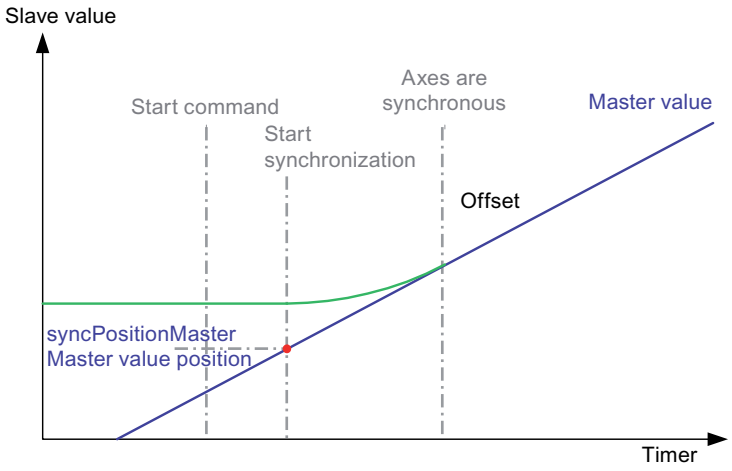


Figure 4-673 Example of synchronization - specification of master value synchronization position, trailing synchronization, absolute, ratio 1:1

Synchronization on specified master value position with specification of an offset on the slave value side

The specified master value position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_MASTER_AND_SLAVE_POSITION**. The synchronization position on the master value side is set in the **syncPositionMaster** parameter. With relative camming on the master value side, the **camStartPosition** parameter is active. The offset on the slave value side is specified in the **syncPositionSlave** parameter.
The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

Regarding the start of synchronization, see Position of synchronization range relative to synchronization position (Page 2660).

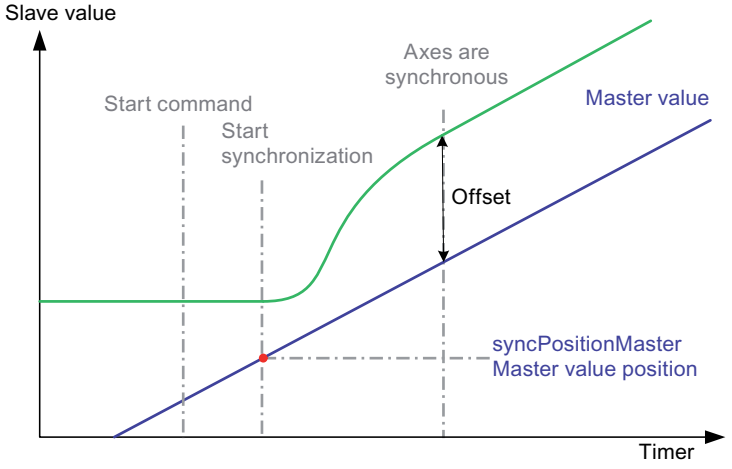


Figure 4-674 Example of synchronization - specification of master value synchronization position and following axis offset, trailing synchronization, absolute, ratio 1:1

Synchronization on the specified following axis position

The specified following axis position is the synchronization criterion. The synchronization criterion is set with **synchronizingMode:=ON_SLAVE_POSITION**. The synchronization position on the slave value side is specified in the **syncPositionSlave** parameter. An offset on the slave value side cannot be specified. The synchronization position on the master value side is determined from the application of the inverse transmission function to the synchronization position on the slave value side. With relative camming on the master value side, the **camStartPosition** parameter is active. The **syncPositionMaster** parameter is not active. Synchronization starts if the synchronization position specified in the **syncPositionSlave** parameter is reached on the following axis as a result of a motion initiated elsewhere.

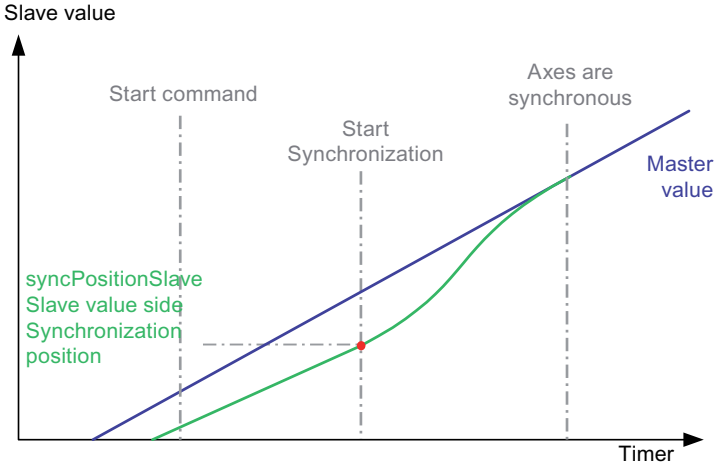


Figure 4-675 Example of synchronization - specification of following axis synchronization position, trailing synchronization, absolute, ratio 1:1

Synchronization at the end of the current camming cycle

The master value position at the end of the current camming cycle is the synchronization criterion. This setting can only be assigned in conjunction with relative camming on the master value side and already active camming. The synchronization criterion is set with **synchronizingMode:=AT_THE_END_OF_CAM_CYCLE**. The synchronization position on the master value side is the master value position at the end of the current camming cycle. The **syncPositionMaster** parameter is not active. With relative camming on the master value side, the **camStartPosition** parameter is active. An offset on the slave value side cannot be specified, and the **syncPositionSlave** parameter is not active. The **syncPositionReference** parameter specifies whether to activate leading, symmetrical (only for synchronization via a specifiable master value distance), or trailing synchronization.

Synchronization direction

The **synchronizingDirection** parameter of the synchronous operation commands can be used to specify the direction of the motion for synchronization. If a specific synchronization direction is specified, the synchronization motion is in this direction only.

The synchronization direction of the following axis in the synchronization phase can be specified with the **synchronizingDirection** parameter in the **_enableGearing(), _disableGearing(), _enableCamming()**, and **_disableCamming()** commands (V3.1 and

higher). This function is relevant, for example, for axes, for which synchronization is possible in both directions.

For information on axes with backstop, refer to the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Manipulated variable limitation (backstop)"

Synchronization with direction of specification can be set as follows:

- **Maintain present system behavior (SYSTEM_DEFINED):**
  This corresponds to the **shortest path** setting, however the direction of motion is maintained for the axis motion.
- **Maintain direction of the following axis (SAME_DIRECTION):**
  The current direction of following axis motion is maintained during the synchronization phase.
  - The direction of following axis motion is maintained during synchronization when the master value is at a standstill.
  - The synchronization occurs in the positive direction during synchronization at both the master value standstill and the following axis standstill.
- **POSITIVE_DIRECTION** setting: A **positive** synchronization direction is defined.
- **NEGATIVE_DIRECTION** setting: A **negative** synchronization direction is defined.
- **SHORTEST_WAY** setting: Synchronize to the **shortest way**, regardless of which direction of motion ensues in the synchronization phase.
  With this setting, however, a direction reversal could occur during synchronization.

Position of synchronization range relative to synchronization position

The position of the synchronization range relative to the synchronization position can be set with the **syncPositionReference** parameter of the **_enableGearing()** or **_enableCamming()** command:

- Synchronize before the specified synchronization position: **Leading synchronization (syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION)**
  - The end point of the synchronization is specified via the synchronization criterion.
  - The starting point of synchronization is determined for synchronization via a specifiable master value distance, on the basis of the specified synchronization length, and is calculated with reference to the dynamic response parameters of the system in accordance with the specified dynamic values and the master value behavior.
- Synchronize starting from the specified synchronization position: **Trailing synchronization (syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED)**
  - The starting point of synchronization is specified directly or implicitly (by means of the following axis position).
  - The end point of the synchronization is calculated for synchronization via a specifiable master value distance from the specified synchronization length and is calculated with reference to the dynamic response parameters of the system according to the dynamic response parameters and the master value behavior.
- Symmetrically relative to the specified synchronization position **(syncPositionReference:=SYNCHRONIZE_SYMMETRIC)**
  - With synchronization via a specifiable master value distance, the starting point and end point of synchronization are determined on the basis of the master value positions, in accordance with the specified synchronization length.
  - Synchronization with the SYNCHRONIZE_SYMMETRIC specification is not possible with a synchronization profile using specifiable dynamic response parameters (RELATE_SYNC_PROFILE_TO_TIME). This command is rejected with TO alarm "30001: illegal parameter".

Synchronization starts under the following conditions:

- With trailing synchronization: When the synchronization position on the master value side or slave value side is reached.
- With symmetrical synchronization: When the master value has reached the synchronization position, reduced by half the synchronization length in the master value direction of motion Master value>= synchronization position (synchronization length in direction of motion of master value/2).
- With leading synchronization: When the synchronization position on the master value side is reached, reduced by the synchronization length in the direction of motion.

Note the following:

- With trailing synchronization and absolute synchronous operation, the following axis distance to be traveled according to the transmission function based on the progress of the master value must be applied too. For this reason, leading synchronization should be used whenever possible.
- With synchronization criterion **synchronizingMode:=IMMEDIATELY** or **IMMEDIATELY_AND_SLAVE_POSITION**, trailing synchronization is implicitly present.

Synchronization via a specifiable master value distance

With synchronization via a specifiable master value distance, a synchronization profile is calculated from a specifiable path length of the master value and is applied relative to the master value.

The setting is made with the **syncProfileReference:= RELATE_SYNC_PROFILE_TO_LEADING_VALUE parameter.**

The path length of the master value is specified in the parameter **syncLength.** As a result, synchronism is always achieved in the setpoint specification. The dynamic response during synchronization is dependent on the calculated profile via the master value and on the change in the master value. The dynamic response values specified in the command are not active.

Synchronization length

The synchronization operation takes place as long as the master value is within this defined length. No dynamic response values are taken into account. The profile is calculated as a function of the master value velocity. (See **Synchronization profile type**)

The **synchronization range** is specified using the synchronization length of the master value defined in the **syncLength** parameter of the **_enableGearing()** or **_enableCamming()** commands.

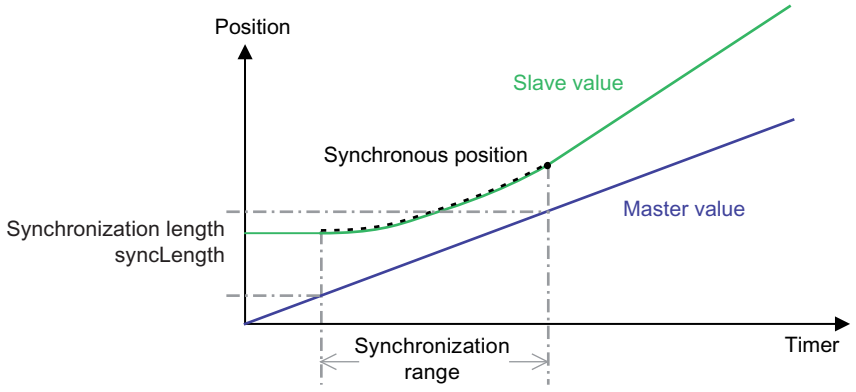


Figure 4-676 Synchronization length for synchronization via a specifiable master value distance

The "synchronous" status is set immediately if the master value source and following axis are at a standstill and the synchronization criterion has already been fulfilled. In this case, the message "50006 Activation/deactivation of synchronous operation executed directly" is output.

Synchronization profile type

The synchronization profile type for synchronization with master value reference is set using the **syncingMotion.velocityMode** configuration data element:

- In the **CONTINUOUS** setting (default), the synchronization profile is calculated with a constant position and constant velocity, but not with constant acceleration.
  The slave value is synchronized using a polynomial profile and the master value. Therefore, the resulting velocity and acceleration of the following axis for synchronization are dependent on the synchronization length and the dynamic response of the master value during synchronization.
- In the **NON_CONTINUOUS** setting, the synchronization profile is calculated using the specified master value length with a constant position only in the slave value behavior. The slave value is synchronized using a linear profile and the master value.

Synchronization profile

Synchronization profile based on specifiable dynamic response parameters (time reference)

When this synchronization profile is used, a synchronization profile is calculated according to the specified dynamic response parameters and the master value dynamic response that exists at the start of the profile. The profile is applied *according to the master value* for leading synchronization and *according to time* for trailing synchronization.

The setting is made with the **syncProfileReference:= RELATE_SYNC_PROFILE_TO_TIME parameter.**

The dynamic response for the synchronization is specified in the dynamic response parameters for the synchronous operation commands. A velocity profile with constant velocity (**TRAPEZOIDAL**) or constant acceleration (**SMOOTH**) can be specified using the **velocityProfile** parameter. For synchronization with a synchronization profile based on dynamic response parameters and with constant acceleration, any reversing that occurs during synchronization is at zero acceleration at the reversing point.

The synchronization profile based on dynamic response parameters can be applied:

- For leading synchronization
- For trailing synchronization

Symmetrical synchronization is not possible.

See also Position of synchronization range relative to synchronization position (Page 2660), as well as section Adapt the synchronization velocity to the master value velocity (Page 2718).

Application

A synchronization profile based on dynamic response parameters is especially suited for:

- Time-optimized synchronization, according to dynamic response specifications

Adaptation to the dynamic response of the master value (leading and trailing synchronization)

If the current dynamic response variables of the master value are larger than the dynamic response parameters of the synchronization command, the parameters can be automatically adapted to the dynamic response parameters (as of V3.1).

Parameters for adapting the synchronization dynamic response to the target dynamic response can be assigned on the synchronous object under **Settings (syncingMotion.synchronizing-Adaption)**.

If dynamic response adaptation is disabled, the synchronization dynamic response is no longer adapted to the required target dynamic response. This can result in the situation where, during trailing synchronization, the synchronized axis can no longer synchronize with the leading axis. During leading synchronization, the synchronization motion may not be started under certain circumstances.

Overdrive factor

The permissible magnification of the specified dynamic response values for making up a constant path difference is specified using the magnification factor **(syncingMotion.overdriveFactor)** under **Settings**. The magnification factor relates to the dynamic response of the master value. At 100% magnification, the dynamic response of the synchronization is adapted to the current dynamic response of the master value, taking into account the transmission. When **overDriveFactor** > 100% is set and has taken effect, the "synchronous" status can also be established if the master value is in the acceleration or deceleration phase during synchronization.

If an overshoot occurs, alarm 40012 "Dynamic limitations (type: ...) are violated" is output at the synchronous object.

Application

If a low synchronization velocity is selected on the command, the adaptation is set, and a corresponding overshoot factor is selected, the synchronization velocity of the following axis is adapted to the master value velocity.

Direction-dependent dynamic response

Direction-dependent or direction-independent effectiveness of the programmed dynamic response values can be set with **syncingMotion.directionDynamic**.

See **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits"

Leading synchronization with synchronization profile based on dynamic response parameters

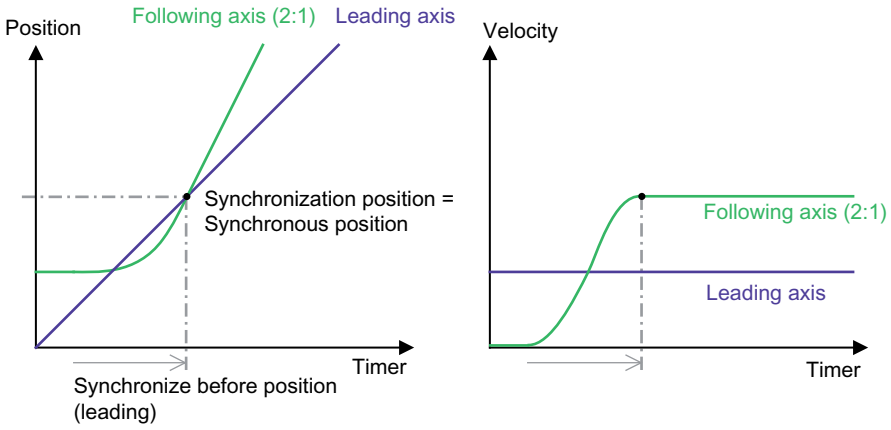


Figure 4-677 Example of leading synchronization (gearing with 2:1 gear ratio, synchronize following axis from standstill)

Only with a synchronization profile based on specifiable dynamic response parameters is a synchronization profile calculated for leading synchronization, taking into account the current master value velocity, the current position, and the dynamic response of the following axis, as well as the dynamic values for synchronization specified in the command. The synchronization profile is then traversed relative to the master value. In addition, if the master value dynamic response changes, the synchronization profile is not recalculated. For this reason, a change in the master value velocity can be seen superimposed in the synchronization operation.

In addition, with the **Extended Look ahead** setting, the acceleration of the master value in the synchronization profile is not taken into account.

Start of synchronization

The synchronization operation starts:

- At the time calculated by the system from which the specified dynamic response parameters can be optimally synchronized with respect to time at a constant master value velocity
- Immediately, if it is not possible to calculate an optimum synchronization time and the synchronization position can be reached (with static master value, for example)

Application

Leading synchronization is appropriate in the following cases:

- If there is to be synchronism at the specified synchronous position and the synchronous position can be easily specified from the application, taking into account the required synchronization operation.
- If dynamic response changes for the master value can be expected during synchronization and are taken into account in the synchronization profile but are not to be reinforced through extrapolation.

Comments

- The programmed velocity profile (SMOOTH, TRAPEZOID) is applied.
- The **syncingMotion.smoothAbsoluteSynchronization** configuration data element is not relevant for leading synchronization.
- Synchronization with extended look-ahead is not active with leading synchronization.
- With leading synchronization, the current master value velocity, and the resulting synchronization profile, if there is insufficient time for synchronization before the master value reaches the synchronization position, synchronization does not occur. The status can be read out via system variables.
  Exception: Modulo master value; in this case, the next possible synchronization position is awaited.

Trailing synchronization with synchronization profile based on dynamic response parameters

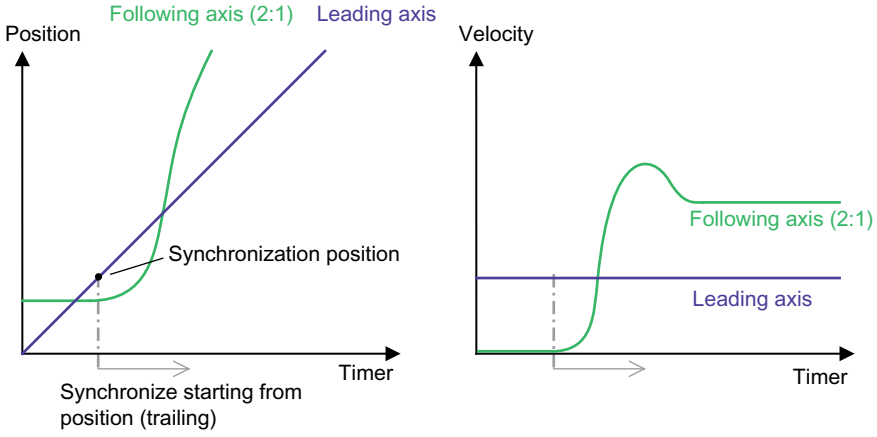


Figure 4-678 Example of trailing synchronization (gearing with 2:1 gear ratio, synchronize following axis from standstill)

With trailing synchronization, the synchronization operation starts when the synchronization criterion is reached. Taking into account the current master value velocity and the specified dynamic response values, the system calculates and executes a time-related synchronization profile such that synchronization is achieved as fast as possible. When calculating the synchronization profile, the current master value acceleration is only taken into account in cases where extended look-ahead is specified.

Changes of the master value for the synchronization are directly taken into account.

Application

Trailing synchronization is appropriate in the following cases:

- When the current master value position must be used directly as the synchronization position.
- When no dynamic response changes of the master value during synchronization are expected.
- When other reasons dictate that synchronization can take place only after the synchronization position.
  Depending on the position of the slave value at the synchronous position, large dynamic movements of the slave value may occur since, in order to comply with the dynamic limits and taking into account the master value dynamic response, the following must occur:
  - Synchronism must be achieved
  - If there are dynamic response changes to the master value, the (anticipated) position changes to the master value must be made up for.

Comments

Trailing synchronization is not suitable for non-constant velocity and acceleration of the master value (i.e. if the acceleration/deceleration changes continually).

Smooth synchronization

Absolute trailing synchronization with consideration of jerk

The **syncingMotion.smoothAbsoluteSynchronization** configuration data element can be used to specify whether a smooth velocity profile is supported during the succeeding synchronization (as of V3.2).

Synchronization is not calculated subject to changes in the master value velocity.

- Provision for jerk during absolute synchronization can be made by setting **syncingMotion.smoothAbsoluteSynchronization:= YES**.
- A setting of **NO** means that jerk will not be taken into account during absolute synchronization, even with velocity profile **SMOOTH**.

Trailing synchronization with extended look ahead

The synchronization with extended look-ahead allows a constant acceleration/deceleration of the master value to be used during the synchronization.

- With synchronization with **standard look ahead**, the position and the velocity are included in the synchronization calculation.
- For synchronization with **extended look-ahead**, the current values of the master axis for position, velocity and acceleration/deceleration are used to calculate the synchronization distance of the following axis (as of V3.2).
  Any changes to the acceleration of the master axis during the synchronization action are ignored and can cause a longer synchronization distance than for synchronization with standard look-ahead.
- During the braking phase of the master synchronization, it can occur that the synchronization process is terminated only after the braking phase of the master. In this case, synchronization without **extended look-ahead**, i.e. in **Standard** mode, can be executed. It can nevertheless occur that the following axis becomes synchronous only after the braking phase.

The function can be activated via the **synchronizingWithLookAhead** parameter of the **_enableGearing()** command.

Extended look-ahead can be preset on the synchronous object via system variable **.gearingSettings.synchronizingWithLookAhead** (as of V4.0).

See also

Position of synchronization range relative to synchronization position (Page 2660)

Settings for evaluation of the master value behavior during synchronization

Toleration of a master value reversal during synchronization:

During synchronization, when the direction of the master value is reversed, the synchronization process is canceled with error message 50007 output at the synchronous object.

If the direction reversal is based on tolerable and predominantly non-influenceable master value fluctuations during downtime, as they occur in the case of an actual value coupling (so-called actual value "noise") or primary extrapolation of the master value, cancelation of synchronization can be prevented by specifying a tolerance band for the master value reversal (master value hysteresis) (as of V4.0).

The maximum master value fluctuation to be tolerated by the system is to be parameterized in the configuration data **syncingMotion.masterReversionTolerance** in the master value positioning unit. An effective hysteresis at a value greater than 0 freezes the master value in the reversal point during direction reversal and thus simulates a still-standing master value for synchronization.

The master value tolerance continuously acts on the active or soon-to-be active master value (see master value switching), beginning with activation of the master value and the hysteresis.

Note

When a master value tolerance > 0 is specified, the system is permitted to immediately offset the following axis change corresponding to one of these master value variables, with maximum dynamics on the following axis, e.g. during use in connection with time-related synchronization. During this process, the current effective coupling factor must also be taken into account.

The variable of the master value tolerance should therefore be selected at as low a level as possible and should be immediately oriented toward the measured fluctuations of the actual value or the extrapolation error.

The effective hysteresis, and thus also an existing reversal point, can be reset by setting the master value tolerance to the value 0. The change, in the same way as setting the tolerance to a value greater than 0, immediately becomes active.

Effective direction of the hysteresis

The effective direction of the hysteresis is automatically determined by the system and is maintained by the permanent functionality that is also outside of synchronization.

As of V4.4, you can specify the effective direction of the master value via the configuration data item **syncingMotion.toleranceDirection**.

- **BOTH_DIRECTIONS** (default)
- **POSITIVE**. The effective direction of the tolerance band is positive. The tolerance band is effective in the positive direction for a master motion, i.e. the hysteresis function takes effect when the master moves from a positive to a negative direction, but not vice versa.
- **NEGATIVE**. The effective direction of the tolerance band is negative. The tolerance band is effective in the negative direction for a master motion, i.e. the hysteresis function takes effect when the master moves from a negative to a positive direction, but not vice versa.

No synchronized status while master value in the hysteresis

As of V4.4 there is a configuration data item **syncingMotion.enableSyncWithinTolerance** with which the reaching of the **synchronized** status is retained within the hysteresis of the master value, or the new response that the **synchronized** status is only reached by the master value when it has left the hysteresis range.

You can set that synchronous operation does not go into **synchronized** status and therefore refers to the current position of the master value using the synchronous operation rule as long as the master value is in the hysteresis as a result of a change of direction. In this way, the difference that results from the synchronous operation rule and the position of the master value at the start of the hysteresis and the current position, is no longer traversed with the maximum dynamic response of the following axis.

If a master value reversal tolerance greater than zero is set for the synchronization, with

- synchronization with master value reference: relative synchronization at rest
- synchronization trailing with time reference: synchronization ends within the tolerance

an unexpected traversing motion can occur when the synchronized status is reached, because the difference resulting from the synchronous operation slave value at the start of the hysteresis and from the synchronous operation slave value for the current master value position is traversed immediately with the maximum dynamic response of the following axis when the **synchronized** status is reached.

Example of switchover in the hysteresis:

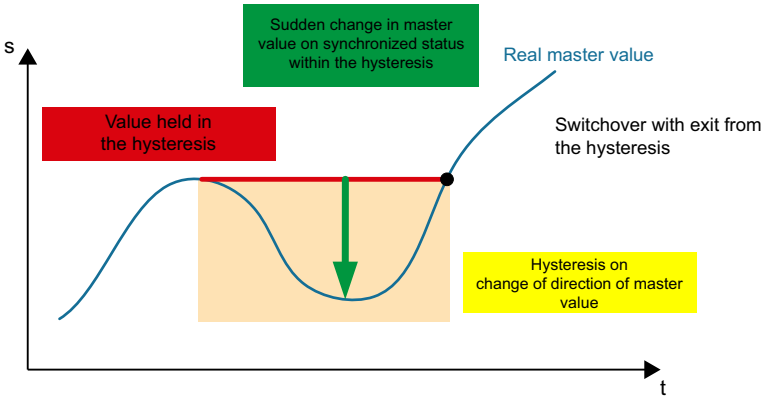


Figure 4-679 Switchover in the hysteresis

To prevent that, as of V4.4 there is a configuration data item **syncingMotion.enableSyncWithinTolerance**.

- **Synchronized status also in hysteresis**
  With the setting **syncingMotion.enableSyncWithinTolerance = YES** (system default setting, compatibility setting), the **synchronized** status can also be reached by the master value within the hysteresis.
- **No synchronized status in hysteresis**
  With the setting **syncingMotion.enableSyncWithinTolerance = NO** (system setting when creating a new synchronous axis), the **synchronized** status is not set for the master value in the hysteresis range.

Toleration of master value velocity changes during synchronization:

The tolerance of the velocity changes of the master value can be set in the configuration data **syncingMotion.maximumOfMasterChange** (default setting: 20%).
If, during synchronization with a synchronization profile based on the master value distance, there is a change to the master value velocity in excess of what has been parameterized in the configuration data element, an error message is output and the synchronization profile is recalculated.
If, during synchronization with a synchronization profile based on dynamic response parameters and with leading synchronization, there is a change to the master value velocity in excess of what has been set in the configuration data element, an error message is generated but the profile is not recalculated.
During synchronization with a synchronization profile based on dynamic response parameters and with trailing synchronization, the configuration data element is not active. A change to the master value velocity produces an immediate response.

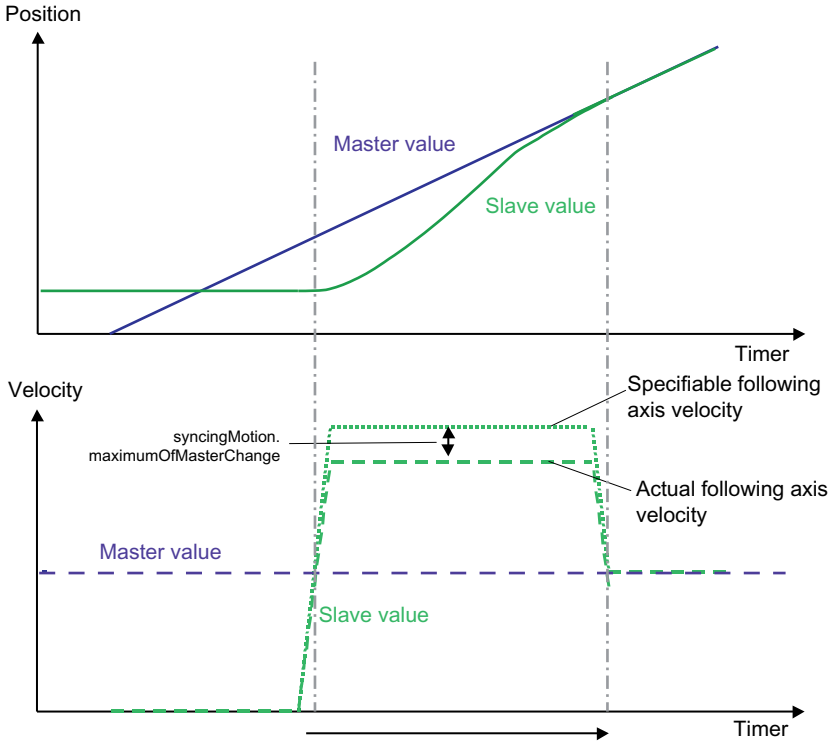


Figure 4-680 Example of syncingMotion.maximumOfMasterChange for leading synchronization with synchronization profile based on specifiable dynamic response parameters

The values of customizable dynamic parameters for synchronization profiles are initially reduced by the values defined in the **syncingMotion.maximumOfMasterChange** configuration data element. The following axis is then accelerated at reduced acceleration to the reduced velocity, in order to maintain reserves and to terminate synchronization safely at the specified synchronization position.

If the master value velocity changes, then the same changes are made to the dynamic response values of the synchronization procedure. Error message "50009 Changing the dynamic response of the master leads to a dynamic violation when synchronizing/desynchronizing" is issued if the parameterized tolerance is exceeded.

If the direction of the master value reverses during synchronization, the synchronization procedure is aborted with the "50007 Error occurred while activating/deactivating the synchronous operation" error. This does not apply for immediate synchronization or synchronization starting at a defined reference point when **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** with **synchronizingMode:=IMMEDIATELY** or **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** with **synchronizingMode:=SYNCHRONIZE_WHEN_POSITION_REACHED**.

With trailing synchronization, the **syncingMotion.maximumOfMasterChange** configuration data element is not active, i.e. responses are issued continuously.

Monitoring the synchronization

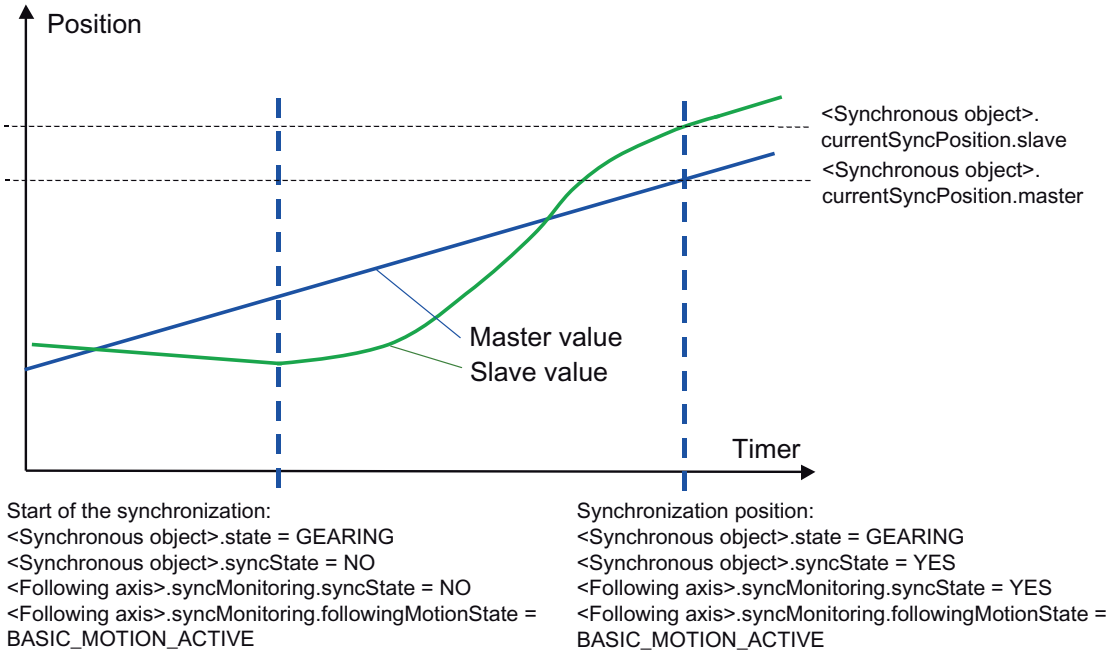


Figure 4-681 System variables for the synchronization

Synchronization state on synchronous object

- The **state** system variable on the synchronous object indicates whether a synchronous operation is active:
  - With state:=**CAMMING** a camming is active.
  - With state:=**GEARING** a gearing is active.
  - With state:=**VELOCITY_GEARING** a velocity gearing is active.
  - With state:=**INACTIVE**, no function is active on the synchronous object.
    When synchronization starts, the system variable is set to the appropriate value and reset again once synchronization is complete.
- The **syncState** system variable on the synchronous object indicates whether the slave value calculated on the synchronous object is synchronous with the master value.
  - If both the master value and slave value are synchronous, this variable will be set to the **YES** state.
    The master value pending on the synchronous object (**currentMasterData.value**) and the slave value output to the following axis (**currentSlaveData.value**) will then be synchronous.
  - The start of desynchronization or any other loss of synchronism causes the variable to be reset to the **NO** value.
    Any restrictions imposed on the slave value transferred by the following axis as a result of the limitation to the maximum dynamic values and the associated non-synchronization of the master and following axis are *not* reflected in the state of the **syncState** variable.
    For the dynamic response limits on the following axis, see Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder Function Manual, "Dynamic limits."
- The master value information on the synchronous operation object does not match the input values from the master value object 1:1, but is instead influenced by:
  - Extrapolation/interpolation for distributed synchronous operation
  - Extrapolation/interpolation through task-wide synchronous operation
  - Extrapolation/interpolation through setpoint output delays
  - Plausibility check at standstill of the master value

  Example:
  If the actual velocity adopts the value zero (because two position values are the same in succession), the filtered value is no longer taken for the actual velocity, but is instead set directly to zero. In the active synchronous operation object therefore 0 is taken for a cycle in currentMasterData.derivedValue. If the (position) setpoint change = 0, the dynamics for preventing reverse procedures are already adjusted to the position change with the setpoint calculation.
- The associated synchronization position of the master and the slave value, i.e. the position from where the master and the slave axis run synchronously are contained in the **currentSyncPosition** system variables on the synchronous object.
  See Display of the synchronization position (Page 2673).

- The state of the synchronization can be queried using the **synchronizingState** system variable on the synchronous object (as of V3.2).
  - **WAITING_FOR_SYNC_POSITION**: Waiting for master value synchronization position
  - **WAITING_FOR_CHANGE_OF_MASTER_DIRECTION**: Waiting for master value direction reversal
  - **SYNCHRONIZING_NOT_POSSIBLE**: Synchronization is not possible
  - **SYNCHRONIZING**: Synchronization in progress
  - **INACTIVE**: Synchronization phase is not active
  - **WAITING_FOR_MERGE**: Synchronization command issued, but not yet active
- The execution state of the active command for activation/deactivation is described in the **enableCommand** and **disableCommand** system variables.
  - **INACTIVE** means that no command is configured.
  - **WAITING_FOR_START** means that the command is executed during slave value generation, and that it is waiting for the start criterion for synchronization to be reached.
  - **ACTIVE** means that synchronization is active and that the operation is synchronous.
  - If there are two commands during slave value generation, both system variables can assume a value not equal to **INACTIVE**. If both are enable commands, the state of the current command is displayed (the state of the next command is always **WAITING_FOR_START**).
- The relevant active command parameter and the parameter for synchronization are grouped together and can be read out in the **effectiveData** system variable structure.

Synchronization state on following axis

- The **syncMonitoring.syncState** variable on the following axis indicates the synchronous operation state on the setpoint side.
  During synchronization and desynchronization, syncState:=NO.
- The **syncMonitoring.followingMotionState** variable on the following axis indicates the state of the synchronous motions:
  - **INACTIVE**: Synchronous motion is not active
  - **BASIC_MOTION_ACTIVE**: Synchronous operation is active as main motion
  - **SUPERIMPOSED_MOTION_ACTIVE**: Synchronous operation is active as superimposed motion
  - **BASIC_AND_SUPERIMPOSED_MOTION_ACTIVE**: Synchronous operation is active as main and superimposed motion

Display of the synchronous position

The **currentSyncPosition** system variables on the synchronous object indicate the last calculated synchronous position of a synchronous operation.

- **currentSyncPosition.master**: Synchronous position of the master value
- **currentSyncPosition.slave**: Synchronous position of the following axis

These values are valid only when "syncState = YES."

Start position of the cam on the axis

The master value and slave value at the cam start of the current camming operation are shown in system variables (V4.0 and higher). The values can also be shown when the starting point of the synchronous operation lies within the cam.

- **currentSyncPosition.camMasterMatchPosition**:
  Master value at the cam start
- **currentSyncPosition.camSlaveMatchPosition**:
  Slave value at the cam start
- **currentSyncPosition.distanceCamMasterMatchPostion**:
  Current relative position in the cam (distance to the cam start)
  Application: Calculation of the corresponding axis positions, also for camming, e.g. a desynchronization position. The axis position must be specified absolute in relation to the axis, even for relative synchronous operation.
  These system variables, can be used, for example, to determine the exact position of the cam for the axis, even for relative camming, and to specify the desynchronization position relative to the axis.

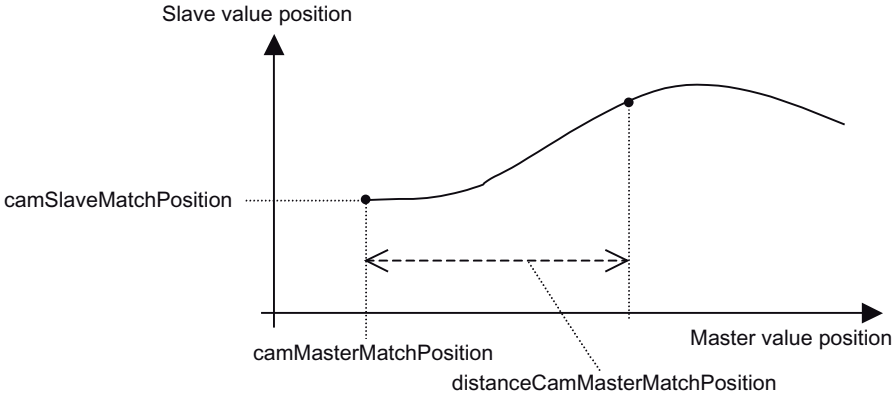


Figure 4-682 Display of the master value and slave value positions in the currentSyncPosition system variable

Position reproduction for modulo axes with gearing

The **currentSyncPosition.slavePositionAtMasterModuloStart** system variable can be used to read out the slave value position at the modulo starting point of the master value (V4.0 and higher).

- **currentSyncPosition.slavePositionAtMasterModuloStart**:
  Slave value position at the modulo starting point of the master value **currentSyncPosition.slave** will be indicated if no modulo is active at the master value.

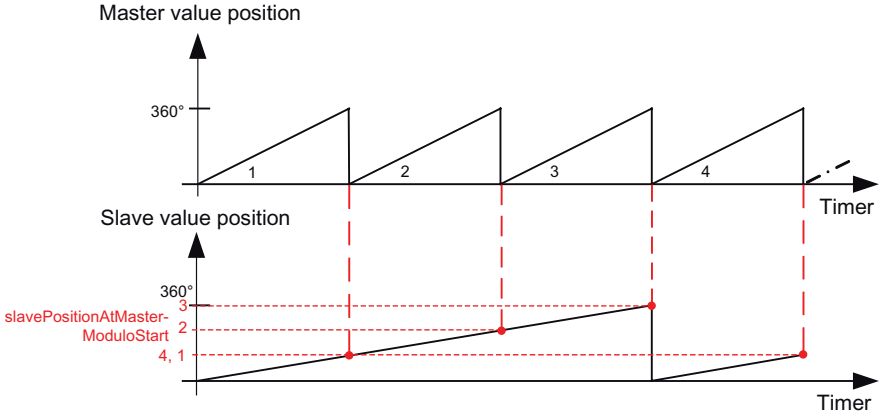


Figure 4-683 Position difference caused by a different modulo starting point

Application

For known gear ratio and known modulo lengths, the assignment of master value and slave value can be closed, even when the slave value modulo length does not correspond to the master value modulo length.

"Synchronous" status during synchronization

- With synchronization via a specifiable master value distance, the "synchronous" status is achieved at the end of the synchronization distance.
- The "synchronous" state of synchronization with a synchronization profile based on customizable dynamic response parameters and leading synchronization is achieved when the synchronization position is reached. (In this case, this is the same as the synchronous position.)
  At the synchronous point, synchronism is present in the position, velocity, and acceleration (only with **SMOOTH** velocity profile).
- With synchronization using a synchronization profile based on specifiable dynamic response parameters and with trailing synchronization, the "synchronous" status is achieved when synchronism exists in the position, velocity, and acceleration (only with profile **SMOOTH** and **syncingMotion.smoothAbsoluteSynchronization:=YES**) in accordance with the transmission function.

With relative gearing without offset, the position is not evaluated during synchronization.

Desynchronization

Desynchronization - Overview

Desynchronization refers to the termination of the synchronous operation.

The desynchronization is defined by several parameters/settings:

- **Desynchronization criterion/desynchronization position**
- **Position of synchronization range relative to desynchronization position**
- the **desynchronization profile**
  - **Desynchronization over a specifiable master value distance**
    The desynchronization length is specified in the desynchronization command.
  - **Desynchronization profile via specifiable dynamic response parameters**
    The dynamic response parameters are specified in the desynchronization command.

Desynchronization criterion/desynchronization position

The desynchronization criterion/desynchronization position is specified in **syncOffMode**.

- Desynchronization at the current master value position; immediate desynchronization
  Desynchronization at the current master value position is set in **syncOffMode:=IMMEDIATELY**. Only trailing desynchronization can occur. The setting in the **syncOffPositionReference** parameter is not active. The **syncOffPositionMaster** parameter is not active. The **syncOffPositionSlave** parameter is not active.
- Desynchronization at a specified master value position
  Desynchronization at a specifiable master value position is set in **syncOffMode:=ON_MASTER_POSITION**. The **syncOffPositionReference** parameter can be used to set leading, symmetrical (only with desynchronization via master value distance), and trailing desynchronization. The desynchronization position on the master value side is set in the **syncOffPositionMaster** parameter. The **syncOffPositionSlave** parameter is not active.
- Desynchronization at a specified slave value position
  Desynchronization at a specifiable slave value position is set in **syncOffMode:=ON_SLAVE_POSITION**. The **syncOffPositionReference** parameter can be used to set leading, symmetrical (only with desynchronization via master value distance), and trailing desynchronization. The desynchronization position on the slave value side is specified in the **syncOffPositionSlave** parameter. The **syncOffPositionMaster** parameter is not active.
- Desynchronization at the end of cam cycle
  Desynchronization at the end of the cam cycle is set in **syncOffMode:=AT_THE_END_OF_CAM_CYCLE**. The **syncOffPositionMaster** parameter is not active. The **syncOffPositionSlave** parameter is not active.

Desynchronization over a specifiable master value distance

Desynchronization via a specifiable master value distance is set in the **syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE** parameter. The slave values travel to zero velocity while the master value travels through the desynchronization length. The desynchronization length is specified in the **syncOffLength** parameter.

See also Synchronization over a specifiable master value distance (Page 2661).

Desynchronization profile via specifiable dynamic response parameters

Desynchronization based on specifiable dynamic response parameters is set in the **syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME** parameter. The slave values travel to zero velocity with the dynamic response values specified in the desynchronization command according to the specified desynchronization criterion.

See also Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 2664).

Position of synchronization range relative to desynchronization position

The position of the synchronization range relative to the desynchronization position can be specified more precisely with the **syncPositionReference** parameter of the **_disableGearing()** or **_disableCamming()** command:

- Desynchronization before the specified desynchronization position
  Desynchronization before the specified desynchronization position is set using the **syncOffPositionReference:=AXIS_STOPPED_AT_POSITION** parameter. The slave value travels to zero velocity to the specified desynchronization position.
- Desynchronization starting from the specified desynchronization position
  Desynchronization starting from the specified desynchronization position is set using the **syncOffPositionReference:=BEGIN_TO_STOP_WHEN_POSITION_REACHED** parameter. The slave value travels to zero velocity starting from the specified desynchronization position.
- Symmetrical desynchronization relative to the specified desynchronization position
  Symmetrical desynchronization relative to the specified desynchronization position is set with the **syncOffPositionreference:= STOP_SYMMETRIC_WITH_POSITION** parameter. The slave value travels at zero velocity symmetrically to the specified desynchronization position. The setting is not possible for the desynchronization profile based on dynamic response parameters.

Replacement of an active synchronous operation

If a synchronous operation is active, it can be replaced by another synchronous operation only when the synchronization criterion of the following synchronous operation can be maintained.

Example:

- For an active camming, the following axis travels only in the negative direction.
- A **_enableCamming()** command with **synchronizingDirection:=POSITIVE_DIRECTION** is issued at the end of the cam cycle (or the criterion, for example, the specified master distance ensures this).

In this case, the first active camming is not ended: the system waits until the following axis travels in the positive direction, which never occurs because of synchronous operation is active. To prevent the described behavior, change the synchronization criterion appropriately or end the active synchronous operation first (using a **_disable** command) and then activate the new synchronous operation.

Replacement of a synchronous operation with small synchronization length

If a synchronous operation group is ended using a **_disable** command with a very short desynchronization length, high dynamic response values result that must be replaced using discontinuous acceleration. If the **_disable** command is replaced before its ending by a new motion command with continuous velocity profile, the still-present high acceleration values are replaced prior to processing the added command. This results in a longer traversal distance of the axis that can also cause the reversing of the axis.

In cases in which an immediate replacement of the synchronous operation is necessary, for the described reason, no **_disable** command should be used, but rather an immediate switch made to the replacing motion command. In this case, the jerk for the following command may need to be increased.

If no **_disable** command is necessary, the subsequent move/pos command for a continuous velocity command should have high dynamic response values or traverse on a "trapezoidal profile."

Substituting motions

A motion that is in an acceleration or deceleration phase is replaced by a motion with jerk limitation (**smooth**). Correspondingly long times can result from the reduced jerk until the current acceleration is reduced.

For an actual value coupling, the result is essentially a noisy signal, depending on the encoder quality. In the case of substitution with jerk limiting (**smooth**), the slave movement is continued relative to this noise-prone signal (acceleration). The system interprets the signal as if the default value is in an acceleration or braking phase and reduces it via the jerk limitation.

With the optional **abortAcceleration** parameter, the functions **_stop** and **_stopEmergency** can be set so that any acceleration is reduced immediately and not via the jerk limitation.

Dynamic response effect on slave values

The dynamic response of the slave values is determined from:

- The dynamic response of the master value
- The dynamic response of any master value switchover occurring during the motion
- The dynamic response of the synchronization
- The dynamic response resulting from the transmission function
- If necessary, the dynamic response resulting from the application of offsets and scaling changes
- The limitation of the slave value dynamic response to the maximum values of the following axis

The dynamic response specifications on the synchronous object refer to the slave values calculated on the synchronous object during synchronization. The dynamic limits of the following axis are not taken into account by the synchronous object.

To avoid excessive dynamic response specifications on the slave values,

- The slave values calculated from the the master value based on the transmission function should not exceed the dynamic limits
- The dynamic response specifications for synchronization and master value switchover should not exceed the dynamic limits

The resulting dynamic values are limited to the maximum values on the following axis based on the axis configuration.

The individual dynamic response parameters that are active during synchronous operation are illustrated in the figure below.

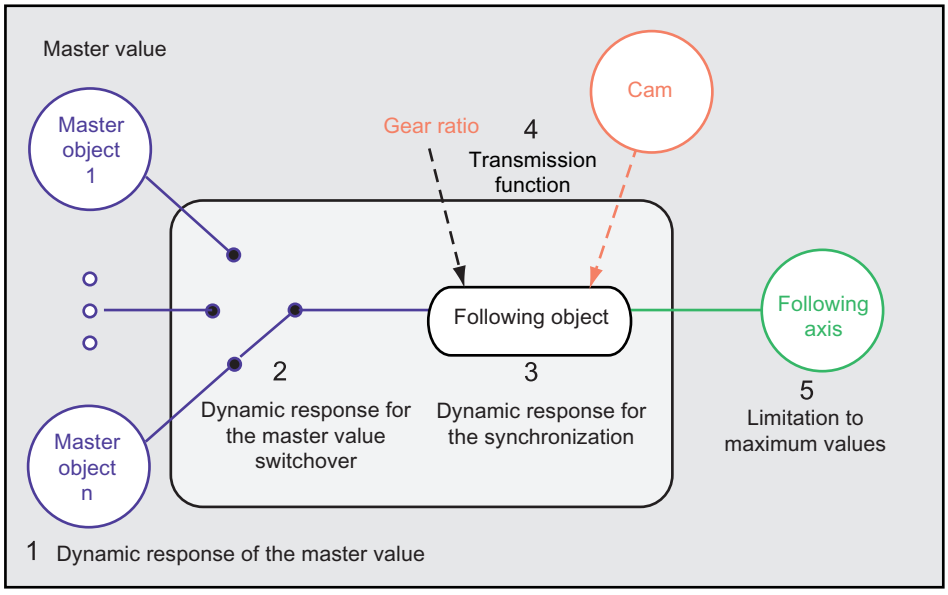


Figure 4-684 Dynamic response during synchronous operation

Legend:

1. The dynamic response of the master value is determined by the motion.
2. The dynamic response of the master value switchover can be specified using the **_setMaster()** command.
3. Synchronization/desynchronization and compensation:
   - *Without* dynamic specification during synchronization via a specifiable master value distance **(RELATE_SYNC_PROFILE_TO_LEADING_VALUE)**
   - Synchronization profile based on specifiable dynamic response parameters (time reference) **(RELATE_SYNC_PROFILE_TO_TIME)**
     The dynamic values set on the synchronous object are only valid for camming or gearing *during synchronization/desynchronization* and when applying corrections, but not in "synchronous" status. See Synchronization (Page 2652), Desynchronization (Page 2676).

4. The dynamic response specification for the following axis is determined by the synchronous object and the gear ratio. The synchronous object is *not* subject to any dynamic limitation in the 'synchronous' status.
5. The slave setpoints on the following axis are limited to the maximum axial dynamic response.
   Configuration data: **TypeOfAxis.MaxAcceleration/MaxVelocity/MaxJerk**
   System variables: **plusLimitsOfDynamics/minusLimitsOfDynamics**. The lowest limit is taken into account in each case.
   See the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits."
   The maximum axial jerk is only used to monitor and, when appropriate, limit the synchronous operation setpoints if synchronous operation monitoring has been activated with provision made for jerk. The setting for synchronous operation monitoring has no effect on the synchronous operation setpoints generated. This even applies for the synchronization procedure, for example.
   The alarm "40202 Dynamic response of the synchronous operation setpoint cannot be achieved" is output if the slave setpoints calculated by the synchronous operation are higher than the active axial limits for velocity and acceleration. If the slave values are limited as a result of this or due to dynamic discontinuity of slave values for synchronous operation (caused by master value jumps, for example), a setpoint error is generated in the slave values.
   See Synchronous operation monitoring (Page 2687).
   The maximum jerk on the axis can be exceeded during synchronization and desynchronization if the jerk setting in the synchronization parameters on the synchronous object is greater than the maximum jerk on the axis. To prevent this, an alarm response can be configured, for example.
6. If desynchronization occurs within one system clock cycle and the target dynamics are therefore zero in respect of velocity and acceleration when the next clock cycle starts, no alarm will be output.

Switching of the master value source

Switching over the master value source – Overview

If more than one master value is assigned to a synchronized axis, the master value source can be selected and switched over on the synchronous object using the **_setMaster** command.

If a synchronous object is assigned to multiple master values, a random master value source is selected internally following system startup. The correct master value source must be specified in the user program. The master value source can be switched "on-the-fly". When it is enabled, the master values are referenced to the units system of the current master value source. A relative or absolute coupling influences the transition process.

The master value transition can be set with and without dynamic response using the **transientBehavior** parameter of the **_setMaster** command (V3.2 and higher):

- **DIRECT**: Without dynamic response (default)
- **WITH_DYNAMICS**: With dynamic response
- **WITH_NEXT_SYNCHRONIZING**: With next synchronization (as of V4.1)

See also

Master value switchover without dynamic response (Page 2681)

Master value switchover with dynamic response (Page 2681)

Master value switchover with next synchronization (V4.1 and higher) (Page 2682)

Master value switchover without dynamic response

The transition behavior when the master value source is changed is different for absolute synchronous operation and relative synchronous operation.

- With relative synchronous operation, an additional slave value difference occurs only if the dynamic master values are different with regard to velocity and acceleration.
- With absolute synchronous operation, a non-continuous master value transition can occur. Discontinuities in the slave values are limited to the maximum dynamic axis parameters on the following axis. A compensation movement is generated in certain circumstances.

Different modulo settings of the master value sources are taken into account.

Master value switchover with dynamic response

The dynamic response parameters: The velocity profile, velocity, acceleration, and, if necessary, jerk can be specified directly in the **_setMaster()** command. These parameters refer to the dynamic response of the transition of the master value source. The **setMasterCommand** system variable indicates the status of the **_setMaster()** motion on the synchronous object.

Note

If the **_setMaster** command switches over the master value, the output of the synchronous object remains asynchronous to the new master value during the transition. The system variables for the synchronization remain unaffected. The transition behavior of the master value does not have any effect on the active gearing/camming.

Please note that a master value switchover does not constitute a new synchronization procedure, i.e. the syncState system variable (on the synchronous object) indicates YES.

To ensure setpoint synchronism, the **setMasterCommand** and **syncState** system variables must be monitored.

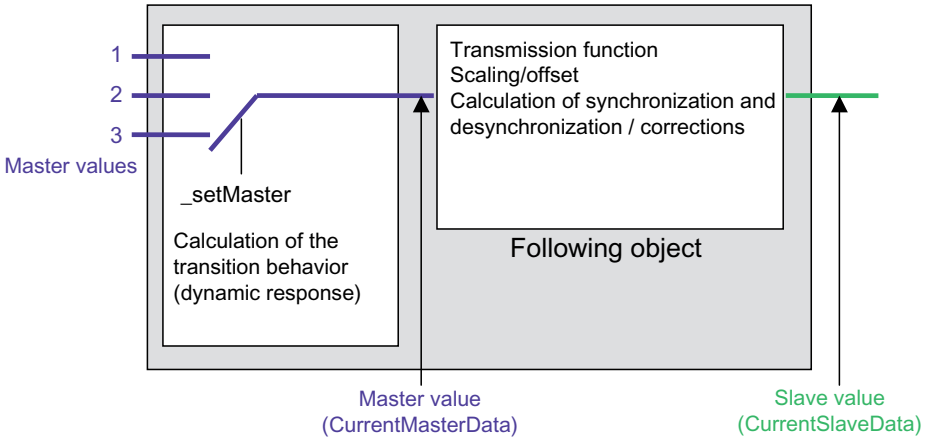


Figure 4-685 Transition behavior and master value for the master value switching with dynamic response

The transition behavior for the new master value is calculated separately from the synchronization and desynchronization, and until the end of the compensation is used as master value for the setpoint monitoring and the evaluation of the **syncState, synchronizingState** synchronization status.

The comparison of this value with the output value of the synchronous object sets the **syncState** and **synchronizingState** variables: **syncState=YES** and **synchronizingState=INACTIVE**. Despite the switching, the **differenceCommandValue** setpoint difference is zero.

Master value switchover with next synchronization (V4.1 and higher)

The master value switchover is active together with the next **_enableCamming()/ _enableGearing()** synchronization command, whereby all specifications refer to the new master value. The dynamic response values in the **setMaster()** command are not active because the dynamic response values of the synchronization command are active during synchronization.

System variable **stateSetMasterCommand** indicates the current status. Parameter **transientBehaviour** sets the master value switchover:

- Master value switchover: is not active: system variable **stateSetMasterCommand = INACTIVE ,** parameter **transientBehaviour = INACTIVE**
- Master value switchover is active, switchover occurs directly: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_DIRECT,** parameter **transientBehaviour = DIRECT**
- Master value switchover is active, switchover occurs with dynamic response values: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_WITH_DYNAMICS ,** parameter **transientBehaviour = WITH_DYNAMICS**
- Master value switchover is active; switchover occurs with next synchronization: system variable **stateSetMasterCommand = TRANSIENT_BAHAVIOR_WITH_NEXT_SYNC,** parameter **transientBehaviour = WITH_NEXT_SYNCRONIZING)**

Superimposed synchronous operation

Description

Two synchronous operations can be *superimposed* by creating another synchronous object on an axis (as of V3.0).

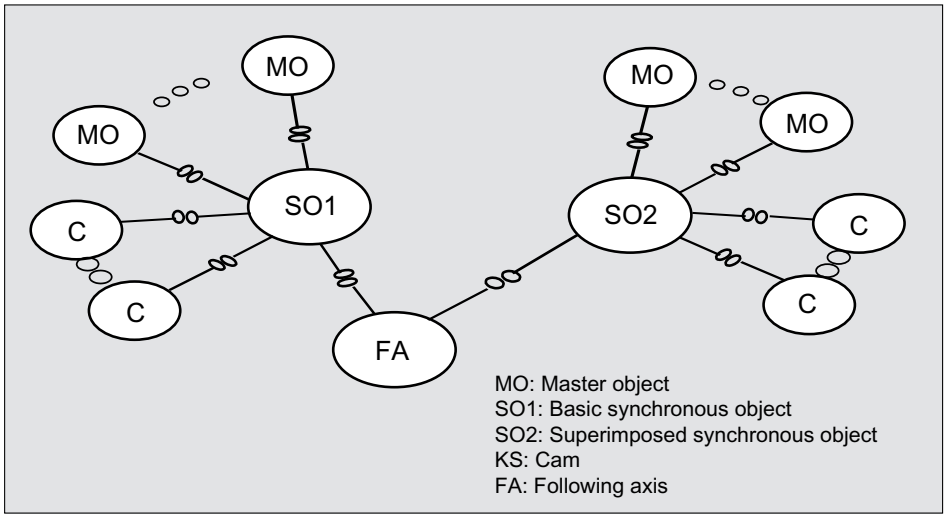


Figure 4-686 Schematic of a superimposed synchronous operation

To differentiate from the simple synchronous operation, the first synchronous operation is called a *basic synchronous operation* and the second is called a *superimposing synchronous operation*. The synchronous objects are called accordingly *basic synchronous object* and *superimposing synchronous object*.

The synchronous operation can be set on the synchronous object to act as the basic motion, main motion (has the same effect as non-superimposed synchronous operation), or superimposed (or secondary) motion (**syncingMotion**.**motionImpact:=STANDARD**/ **SUPERIMPOSED_MOTION** configuration data element).

A maximum of one basic synchronous object can be interconnected to an axis, plus one superimposed synchronous object on the same axis.

Creating an axis with superimposed synchronous operation

In the project navigator under <**Axis_n**>, it is possible to insert a maximum of one further synchronous object <**Axis_n_SYNCHRONOUS_OPERATION_1**>, which is then superimposed, i.e. configuration data **motionImpact** is initialized to **SUPERIMPOSED_MOTION**.

1. Select the axis in the project navigator.
2. Select **Expert > Superimposed synchronous object** from the shortcut menu.

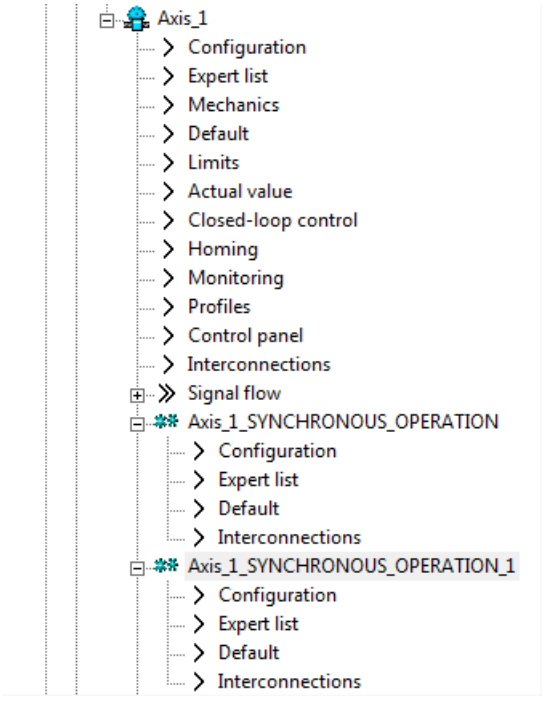


Figure 4-687 Representation of superimposed synchronous operation in the project navigator

The configuration and settings for superimposed synchronous operation are made in the same way as for the basic synchronous object.

Programming

All functions of the basic synchronous object (e.g. _enableGearing, _disableGearing, etc.) can be applied on the superimposed synchronous object. Cross-references between synchronous objects are not possible.

Absolute or relative superimposed synchronous operation

With superimposed synchronous operation, the properties for absolute or relative are the same as for basic synchronous operation, with the exception that the coordinates refer to the superimposed coordinate system on the following axis.

Coordinates

For the basic synchronous object, the synchronization parameters specified for the slave value position refer to:

- The total coordinate system with **mergeMode:=IMMEDIATELY** and **TypeOfAxis.decodingConfig.transferSuperimposedPosition <> TRANSFER_RESET**
- The basic coordinate system in all other cases

The superimposed synchronous object refers to the superimposed coordinates where slave value position specifications have been given.

$$\begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Cumulative} \\ \text{coordinates}}} = \begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Basic} \\ \text{coordinates}}} + \begin{pmatrix} s \\ v \\ a \end{pmatrix}_{\substack{\text{Superimposed} \\ \text{coordinates}}}$$

Figure 4-688 Coordinates for superimposed synchronous operation

Just as with a superimposed motion, each synchronous object has its own coordinate system. The "outputs" of the synchronous objects are added in the Slave Axis technology object. If, for example, the basic synchronous operation and superimposed synchronous operation have the same master value, and motion takes place in absolute gearing with a ratio of 1:1, the position value of the following axis will be twice that of the master axis after both synchronous objects have been synchronized.

Behavior of superimposed synchronous operation in relation to basic motion

Superimposed synchronous operation behaves like a superimposed positioning motion with regard to the basic motion on the axis (motion or synchronous operation). **TypeOfAxis.decodingConfig.transferSuperimposedPosition** configuration data on the synchronous axis set when superimposed motions will be applied to the basic motion, and when they will be substituted. This setting determines, for example, when **mergeMode=IMMEDIATELY** that the superimposed motion is to be substituted on the basic motion. These configuration data settings also apply to superimposed synchronous operations (see also here (Page 2749)).

There can be only *one* superimposed motion on the axis at one time, for example, a **superimposed positioning motion** *or* **superimposed synchronous operation**. A **superimposed synchronous operation** can be active without a basic motion being active at the same time.

See also superimposed motion for axis, **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual.

Monitoring

The output values of a synchronous object (and thus also the motion component of the superimposed synchronous operation for the axis) can be read out in the **currentSlaveData** system variable on the synchronous object.

Table 4-278 Coordinates on the following axis for the superimposed synchronous operation

	System variable	Description
All coordinates:		
positioningState.	**commandPosition**	Target position (total)
motionStateData.	**commandVelocity**	Target velocity (total)
	commandAcceleration	Target acceleration (total)
Basic coordinates:		
basicMotion.	**position**	Position in the basic coordinate system
	velocity	Velocity in the basic coordinate system
	acceleration	Acceleration in the basic coordinate system
Superimposed coordinates:		
superimposedMotion.	**position**	Position in the superimposed coordinate system
	velocity	Velocity in the superimposed coordinate system
	acceleration	Acceleration in the superimposed coordinate system

The **syncMonitoring** system variable on the following axis also displays the state of the synchronous operation motion (as of V3.0):

- **followingMotionState =**
  - **INACTIVE**: Synchronous motion is not active
  - **BASIC_MOTION_ACTIVE**: Standard synchronous operation is active
  - **SUPERIMPOSED_MOTION_ACTIVE**: Superimposed synchronous operation is active
  - **BASIC_AND_SUPERIMPOSED_MOTION_ACTIVE**: Standard synchronous operation and superimposed synchronous operation are active

Compensations during distributed synchronous operation

Compensation during distributed synchronous operation is also useful/effective during superimposed synchronous operation.

See Compensations for distributed synchronous operation (Page 2767).

Synchronous operation monitoring / state

- The following table describes the conditions that apply when the state is synchronous (**syncMonitoring.syncstate = YES**):

Basic synchronous operation	Superimposed synchronous operation
Not sychronous (**synchState = No**)	Synchronous (**synchState = Yes**)
Basic synchronous operation synchronous (**synchState = Yes**)	Not synchronous (**synchState = No**)

- The Not synchronous state (**syncMonitoring.syncstate = No**)only applies during the synchronization/desynchronization process (**synchronizingState <> Inactive**)

Example:
A synchronous operation is started. After synchronization, the synchronous operation is assigned the 'synchronized' state. Now another synchronous operation is started. The 'synchronized' state disappears for the duration of the synchronization and is not reset until the synchronization of the second synchronous operation is complete.

The variables and monitoring on the axis refer to the total synchronous operation. Error messages (synchronous operation errors on the synchronized axis) are issued on all interconnected synchronous objects.

Synchronous operation monitoring

The slave values calculated by the synchronous object (**currentSlaveData**) and any additional setpoints on the axis are limited on the following axis to the maximum dynamic response values. The deviation in the slave values resulting from this limitation is monitored. The current maximum limits for velocity and acceleration (and jerk) on the axis influence this monitoring process.

See the **Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder** Function Manual, "Dynamic limits".

The **syncMonitoring** system variables on the following axis indicate setpoint and actual value differences:

- **Setpoint monitoring**
  differenceCommandValue shows the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account. **limitCommandValue** shows that the difference between the calculated slave value and executable setpoint is above the permissible tolerance.
- **Actual value monitoring**
  differenceActualValue contains the sum from **differenceCommandValue** (the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account) and the position lag at the following axis. **limitCommandValue** shows that the difference between the calculated slave value and the actual value lies above the permissible tolerance.

Setpoint monitoring

On the axis, the slave values calculated by the synchronous operation are limited to the maximum dynamic response values of the axes. This can cause the setpoints on the axis to change.

Consequently, the setpoint error is the difference between the calculated setpoint of the synchronous object (**currentSlaveData**) and the associated setpoint at the following axis that results from observing the acting dynamic maximum limits of the following axis (**basicMotion.position**).

The setpoint error of one of the motions possibly superimposed to the synchronous operation is the difference between the calculated setpoint of the synchronous object (**currentSlaveData**)

and the superimposed setpoint at the following axis that results from observing the acting dynamic maximum limits of the following axis (**superimposedMotion.position**).

The setpoint error from the main and superimposed motion is displayed in **syncMonitoring.differenceCommandValue** on the following axis.

The **syncMonitoring.syncState** synchronization state on the following axis is set according to the **syncState** on the synchronous object. Exception: Superimposed synchronous operation Superimposed synchronous operation (Page 2683).

The following comparison must be performed to determine whether the following axis is synchronized on the setpoint value side:

(<following axis>.**syncMonitoring.syncState:=YES) AND**

(<following axis>.**syncMonitoring.differenceCommandValue = 0)**

Actual value monitoring

The **syncMonitoring.differenceActualValue** system variable allows the position lag present at the following axis, including any acting dynamic response limitation of the following axis (setpoint error in **syncMonitoring.differenceCommandValue**), to be queried. The position lag for DSC operation applies to the position lag present at the position controller in the drive.

For versions earlier than V4.2, the difference is formed between the setpoint calculated on the synchronous object and the actual value present in the interpolator.

Velocity gearing monitoring

The **syncMonitoring** system variable on the following axis also displays the status of the velocity gearing (as of V3.1):

- **differenceCommandVelocity**: velocity difference on the setpoint side between the velocity setpoint calculated on the synchronous object (**currentSlaveData**) and the executable velocity on the following axis (only applies to synchronous velocity operation).
- **differenceActualVelocity**: velocity difference on the actual value side between the velocity setpoint calculated on the synchronous object (**currentSlaveData**) and the executable velocity on the following axis (only applies to velocity gearing).

Configuration

The synchronous operation monitoring is set on the following axis under **Monitoring - Synchronous operation monitoring** (**GearingPosTolerance** configuration data element).

Limiting and monitoring the setpoint error:

- With setting **enableCommandValue := NO_ACTIVATE**:
  - the tolerance monitoring of the setpoints is not activated.
- With setting **enableCommandValue := WITHOUT_JERK**:
  - the tolerance monitoring of the setpoints is activated without inclusion of the jerk. Alarm 40201 is output if the setpoint tolerance is exceeded.
- With setting **enableCommandValue := WITH_JERK**:
  - the tolerance monitoring of the setpoints is activated with inclusion of the jerk. Alarm 40201 is output if the setpoint tolerance is exceeded. The jerk on the axis is also monitored.

  Note

  For distributed synchronous operation with extrapolation on the following axis, the setpoint monitoring with jerk setting is not appropriate.

The actual value deviation monitoring is set via **GearingPosTolerance.enableActualValue**. The synchronous operation monitoring setting only becomes active after synchronization is complete **(syncState = YES).**

For the dynamic response limits on the following axis, see the Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder Function Manual, "Dynamic limits".

Error handling

When the synchronous operation tolerance is exceeded, the following axis issues the technological alarm "40201 Synchronous operation tolerance exceeded on the following axis". An alarm message can also be sent to the master value source; this setting is made in the **TypeOfAxis.GearingPosTolerance.enableErrorReporting** configuration data element. Here, a distinction can be made between deviation tolerance violations of the calculated slave value setpoint and of the calculated slave value actual axis value. The leading axis then issues the error "40110 Error triggered on slave during synchronous operation (error number: ...)".

As of V4.2, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error.

Note

If the **ALL_ERRORS_WITH_ABORT_SYNCHRONIZATION** parameter is set, error message 40110 is triggered on the relevant master object (e.g. also with following error on following axis).

If the synchronous motion is interrupted by a substitutional motion, no error message is generated.

Only following axis errors are reported to the master object. Errors occurring on the synchronous object on command output and/or synchronization are not taken into account.

The troubleshooting can be set using the following configuration data element:

- <Following axis>.TypeOfAxis.GearingPosTolerance.enableErrorReporting
  - **NO_REPORTING**- (default) no message, present
  - **COMMAND_VALUE_TOLERANCE**- setpoint monitoring, present
  - **ACTUAL_VALUE_TOLERANCE**- actual value monitoring, present
  - **ALL_ERRORS_WITH_ABORT_SYNCHRONIZATION**- all errors that occurred

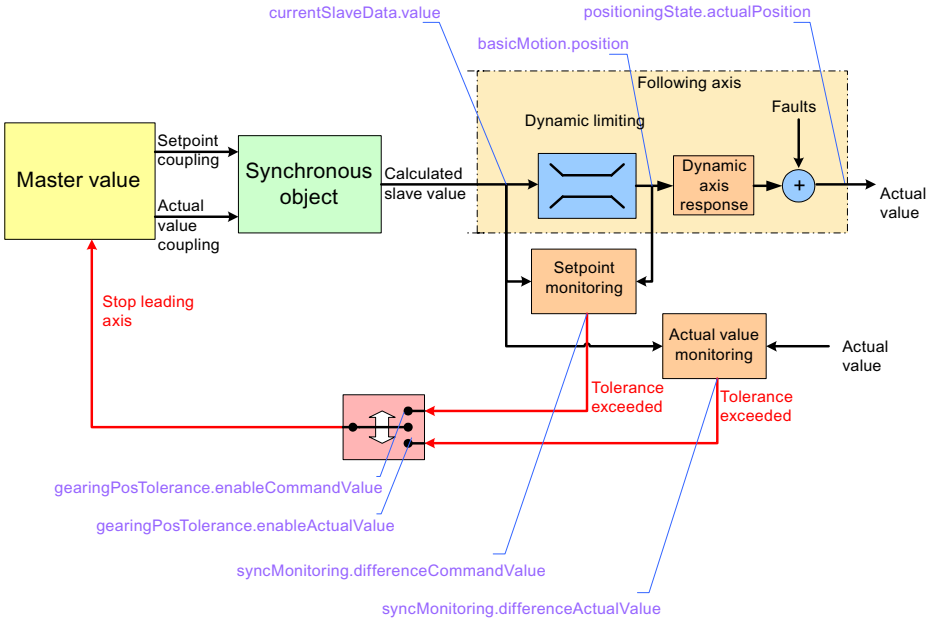


Figure 4-689 Interrelationship between synchronous operation monitoring functions

See Error handling in the user program (Page 2756).

See also

Monitoring the synchronization (Page 2671)

Commands for command tracking (Page 2747)

Simulation mode

Synchronous operation can be switched to *simulation*, in which the values at the synchronous object are calculated but not output to the following axis. The synchronous operation simulation can be activated and deactivated at any time provided there are no faults present.
The **simulation** [ACTIVE/INACTIVE] system variable provides information about the simulation status of the axis.

Application: Retaining a synchronous operation connection with _disableAxis() (Page 2716).

Commands for the simulation operation

- The **_enableFollowingObjectSimulation()** command sets a synchronous operation into simulation mode. The synchronous values are calculated, but are not output to the following axis. This can be done at any time. However, the axis states are taken into account in generating the slave values.
- The **_disableFollowingObjectSimulation()** command resets the synchronous operation relationship out of simulation mode. The synchronization values are output to the following axis again.
  If there is a difference between the setpoint calculated in synchronous operation and the setpoint present at the axis, or superimposed motion has occurred, only dynamic limitation to the maximum values of the following axis is performed.

Configuration data for the simulation operation

The **disableSynchronousOperation** configuration data can be used to set whether master values are to be forwarded to the slave axis.

- If **NO** (default), the synchronous operation is also canceled in simulation mode, provided that the enables on the following axis have been canceled.
- If **YES**, the synchronous operation is not canceled in simulation mode if the enables on the following axis have been canceled while synchronous operation is in simulation mode. Any synchronous operation commands that are undergoing execution are retained.

Configuring units

You can define the basic units for each technology object. The same physical variables can have different units in different technology objects. These are converted:

How to configure the units:

1. In the project navigator, open the context menu under the TO.
2. In the context menu, select Expert > Configure units. The Configure Units window appears in the working area.
3. Select the unit for the physical variables. These units are used for the technology object, for example, s for units of time.

or

1. In the project navigator, open the Configuration under the TO.
2. Select the "Units" tab.

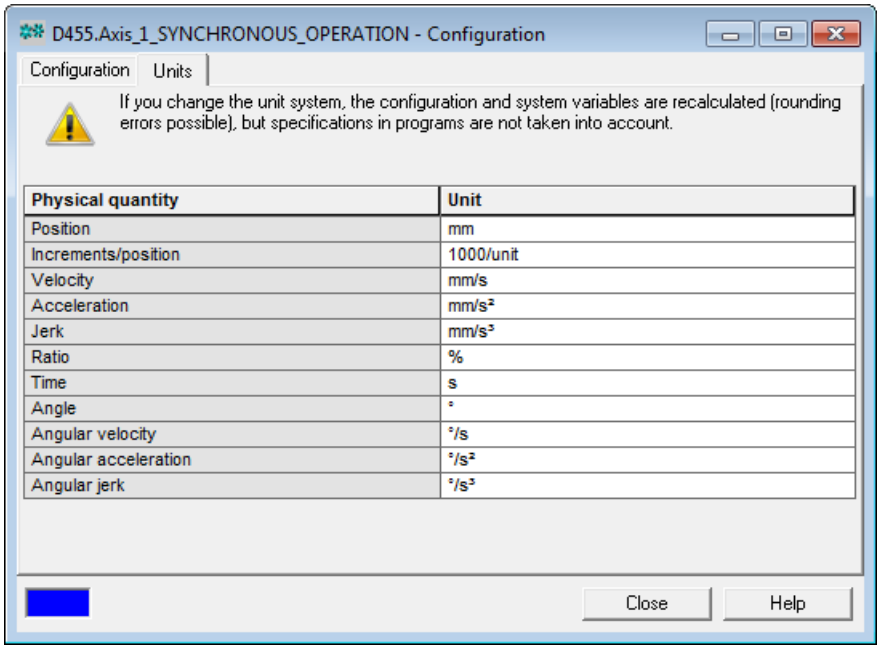


Figure 4-690 Synchronous operation units

You can set the following parameters:

Field/Button	Meaning/Instruction
Table with units	
Physical variable column	Shows the physical variable. The physical variables which are used by the TO are available for the configuration.
Unit column	Displays and configures the unit. Clicking on the cell opens up a combo box for selecting the unit.
Toolbar	
	• Displays whether offline data or online data is shown • Blue field = offline display • Yellow field = online display
Close	Button for closing the dialog.
Help	Button for opening the online help for the dialog.

Examples of synchronization operations as a function of the output position on the slave value side

Synchronization via a specifiable master value distance

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the synchronization profile.

Influence of starting position of the following axis

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Slave value at start of synchronization at standstill
- Synchronization over a master value distance
- Velocity profile type **CONTINUOUS** for synchronization

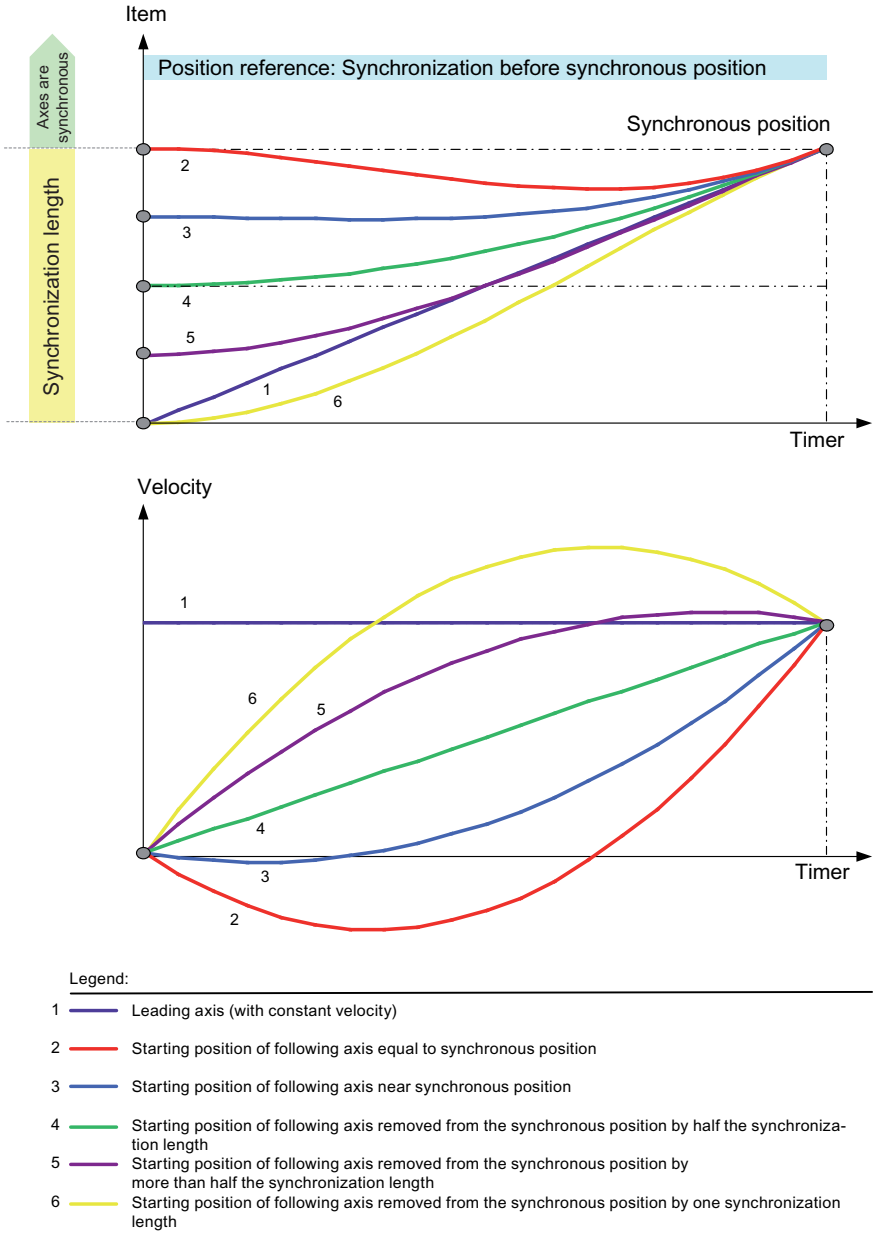


Figure 4-691 Synchronization via a specifiable master value distance

While the position characteristics of the individual synchronization operations do not differ very much, the velocity characteristics differ significantly:

- To accelerate to the synchronous velocity at the synchronous position, motion in the opposite direction following by reversal is required (2).
- It is possible to accelerate directly to the synchronous velocity and synchronous position; the position difference to be applied is minor (3).
- The acceleration to the synchronous velocity in the synchronous position is even (4).
- Direct acceleration to the synchronous velocity in the synchronous position is possible. While the position difference to be applied is greater, a velocity greater than the synchronous velocity and, thus, a velocity reversal is not required for synchronization (5).
- To apply the position difference, a velocity greater than the synchronous velocity and, thus, a velocity reversal is required for synchronization (6).

Recommendation for master value-related synchronization

Recommendation for synchronization via master value distance with 1:1 gear and synchronization from standstill:

- Starting position of the following axis removed from the synchronous position by half the synchronization length.
- Synchronization range symmetrical relative to the synchronous position.

Synchronization profile based on specifiable dynamic response parameters

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the synchronization profile.

Influence of starting position of the following axis with leading synchronization

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Following axis at start of synchronization at standstill

- Synchronization profile based on dynamic response parameters
- Velocity profile type TRAPEZOID for synchronization

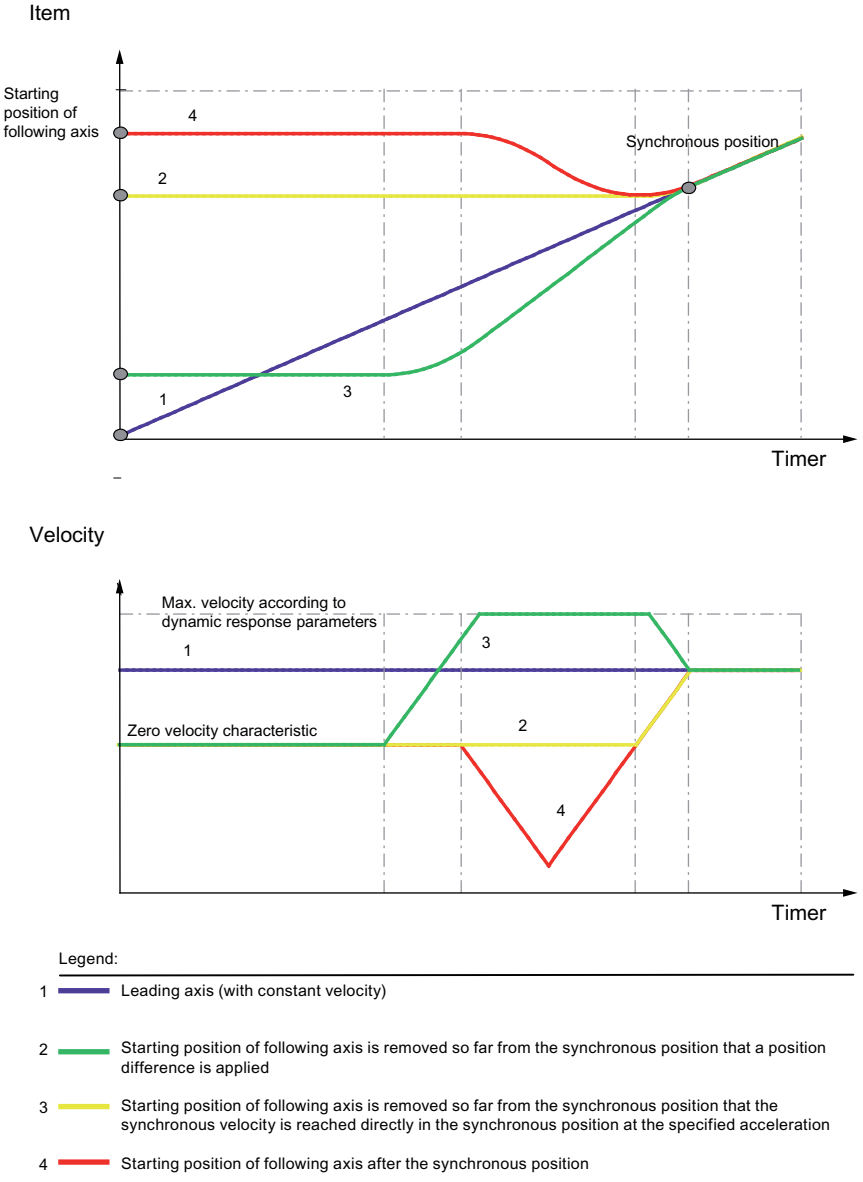


Figure 4-692 Synchronization profile based on specifiable dynamic response parameters and leading synchronization

In the case of leading synchronization based on dynamic response parameters, synchronization starts differently depending on the starting position of the following axis:

- If necessary, the following axis can be accelerated directly to the synchronous velocity and synchronous position (2).
- If the starting position of the following axis is below this point, a position difference must also be applied with the specified dynamic values (3).
- If the starting position is above this point, reversing (i.e. traveling in the opposite direction) is required in order to traverse to the synchronous point at the required synchronous velocity (4).

Influence of starting position of the following axis with trailing synchronization

Together with the master value behavior, the starting position of the following axis relative to the synchronous position on the following axis side largely determines the characteristic of the trailing synchronization profile.

Example:

- Absolute 1:1 gear without offset
- Constant master value velocity
- Following axis at start of synchronization at standstill
- Synchronization profile based on dynamic response parameters
- Velocity profile type **TRAPEZOID** for synchronization

With this type of dynamic response-related synchronization, a master value position is defined from which the synchronization procedure between the master value and following axis is initiated. Here, the desynchronization operation itself is performed based on the dynamic response value settings.

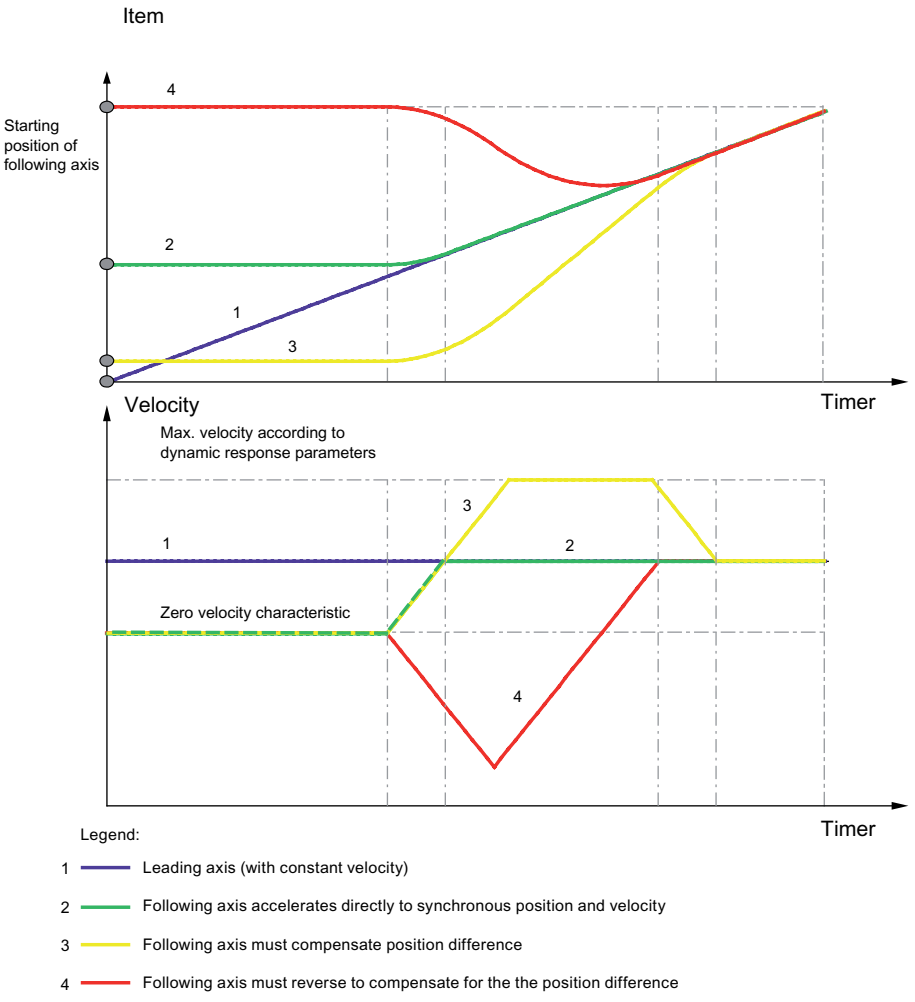


Figure 4-693 Synchronization profile based on specifiable dynamic response parameters and trailing synchronization

With trailing synchronization via dynamic response parameters, "synchronous" status is achieved at different times, depending on the starting position of the following axis.

Depending on the starting position of the following axis:

- If necessary, the following axis can be accelerated directly to the synchronous velocity and synchronous position (2).
- If the starting position of the following axis is below this point, a position difference must also be applied with the specified dynamic values (3).
- If the starting position of the following axis is above this point, reversing (i.e. traveling in the opposite direction) is required in order to traverse to the synchronous point at the required synchronous velocity (4).

Examples

Examples of typical synchronization operations

Several examples for synchronization operations of the gearing and their parameterization in MCC and ST commands are listed here.

Note

The function parameters that are not important to function calls are omitted from the examples. The required parameters are entered directly.

Relative synchronization with master value reference

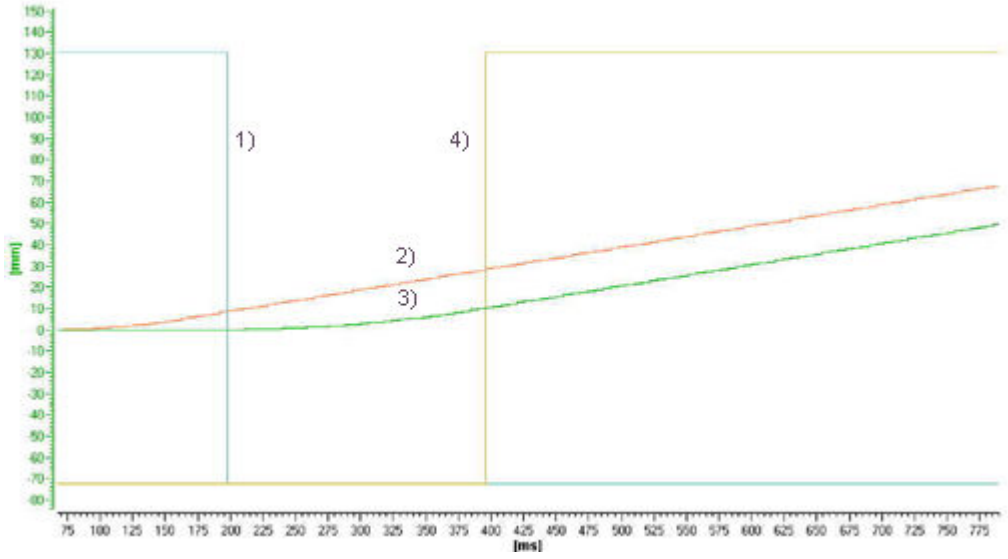
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 0 mm. The synchronization is started immediately; after 20 mm, this should result in relative synchronism between the master axis and following axis.

Table 4-279 ST programming

```
retval:=_enablegearing (
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    direction:=POSITIVE,
    gearingType:=RELATIVE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingDenominator:=1,
    synchronizingMode:=IMMEDIATELY,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    syncLength:=20.0);
```

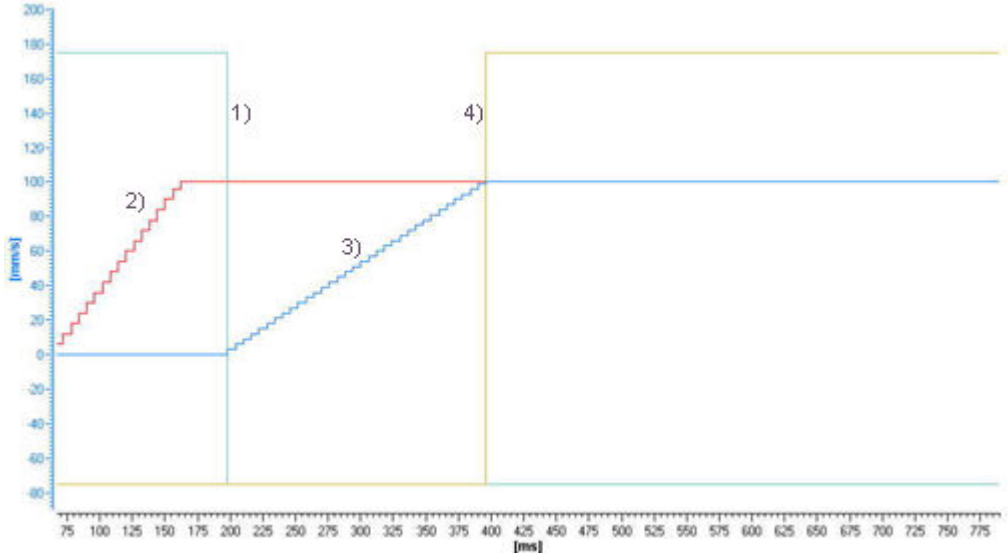
Table 4-280 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	Gearing relative to start position
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	Synchronize immediately
	Synchronization length:	20 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 4-694 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 4-695 Master and following axis velocity

Absolute synchronization with master value reference

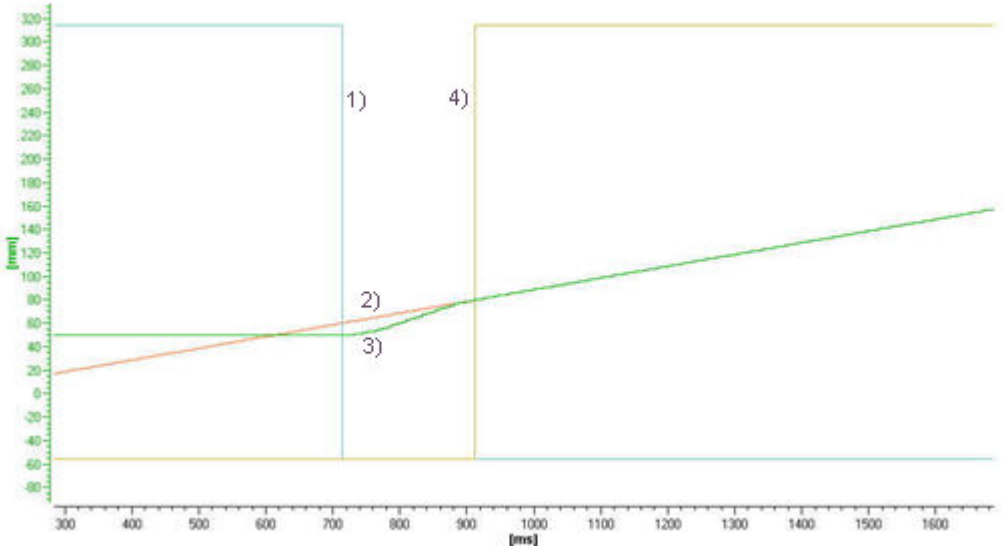
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. Synchronization is performed within 20 mm; this means that for a master axis position of 80 mm, absolute synchronism between the master and following axis occurs.

Table 4-281 ST programming

```
retval:=_enablegearing (
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingRatioType:=DIRECT,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    synclength:=20.0,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=80.0);
```

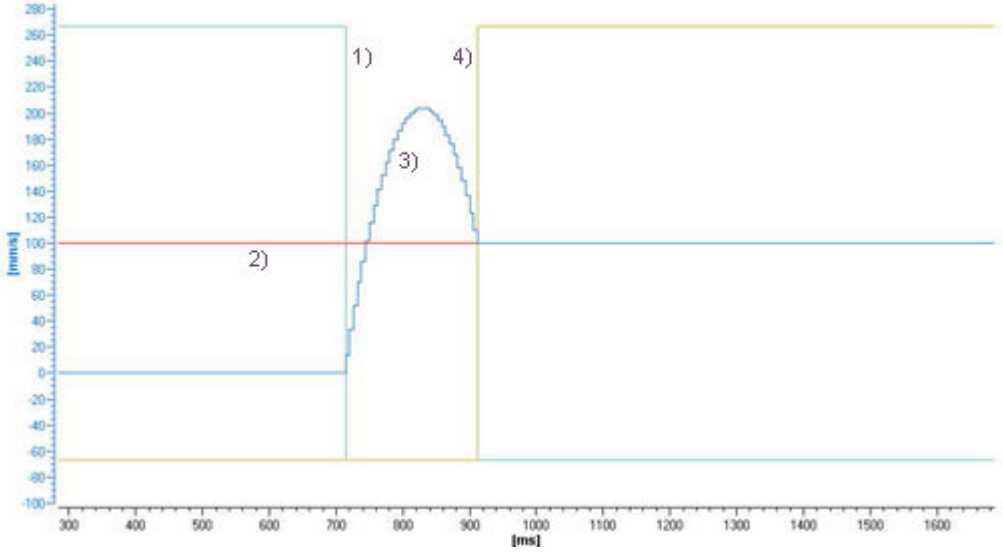
Table 4-282 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize before synchronization position
	Synchronization length:	20 mm
	Leading axis position:	80 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 4-696 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 4-697 Master and following axis velocity

Relative synchronization with master value reference and offset

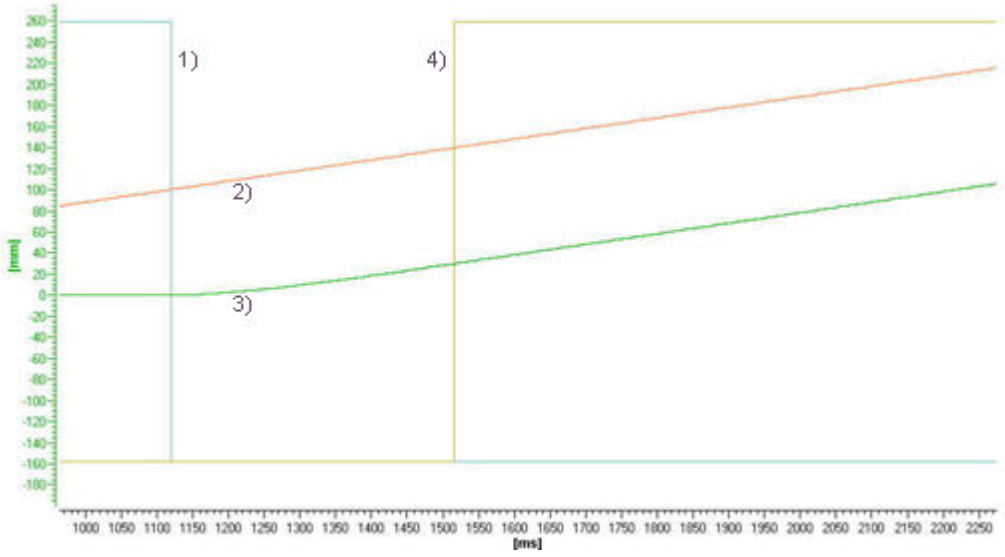
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 0 mm. Starting with a master axis position of 100 mm, the following axis will be synchronized relative to the master axis within 40 mm. When synchronism is achieved, this results in a following axis position based on the position at the start of synchronization and the offset of 30 mm.

Table 4-283 ST programming

```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=RELATIVE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_AND_SLAVE_POSITION,
    syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE,
    syncLengthType:=DIRECT,
    synclength:=40.0,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=100.0,
    syncPositionSlaveType:=DIRECT,
    syncPositionSlave:=30.0);
```

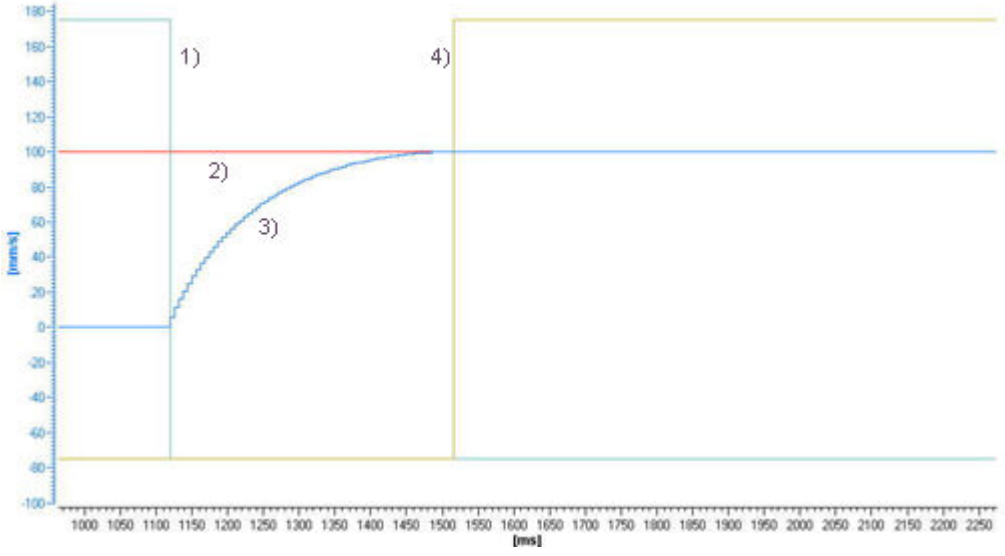
Table 4-284 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	Gearing relative to start position
Synchronization:		
	Synchronization reference:	Leading axis
	Start of synchronization:	at leading axis position with offset
	Offset:	30 mm
	Reference point of the leading axis position:	Synchronize from synchronization position
	Synchronization length:	40 mm
	Leading axis position:	100 mm



1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 4-698 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 4-699 Master and following axis velocity

Absolute synchronization with time reference, trailing synchronization

The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. The specified dynamic response parameters (velocity = 300 mm/s and acceleration = 1,000 mm/s^2) are used for synchronization starting from the master axis position 300 mm, in order to achieve absolute synchronism between the master and the following axis.

Note

Depending on the specified offset, the following axis may need to perform a reversing motion.

Table 4-285 ST programming

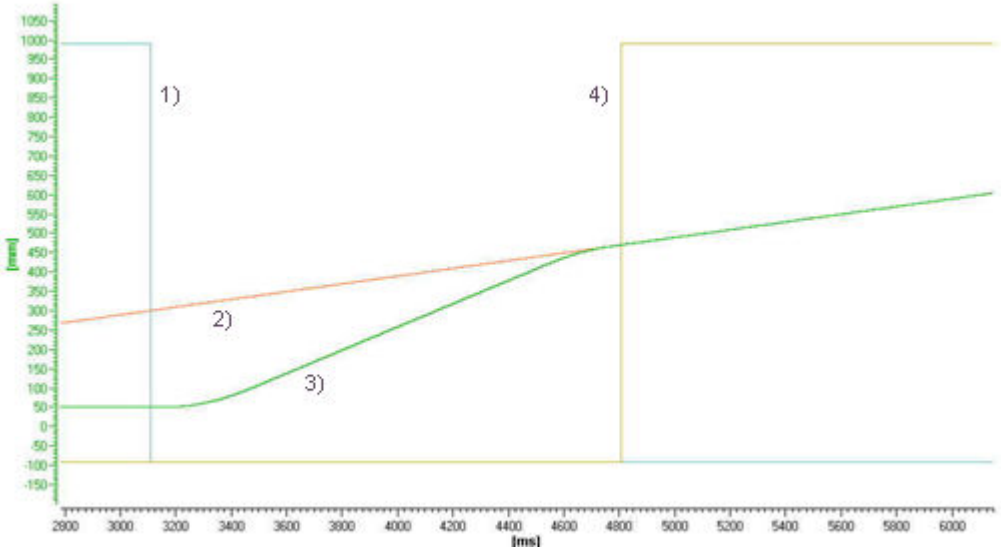
```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingdenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=SYNCHRONIZE_WHEN_POSITION_REACHED,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=300.0,
    velocityType:=DIRECT,
    velocity:=300.0,
    positiveAccelType:=DIRECT,
    positiveAccel:=1000.0,
    negativeAccelType:=DIRECT,
    negativeAccel:=1000.0,
    positiveAccelStartJerkType:=DIRECT,
    positiveAccelStartJerk:=10000.0,
    positiveAccelEndJerkType:=DIRECT,
    positiveAccelEndJerk:=10000.0,
    negativeAccelStartJerkType:=DIRECT,
    negativeAccelStartJerk:=10000.0,
    negativeAccelEndJerkType:=DIRECT,
    negativeAccelEndJerk:=10000.0,
    velocityprofile:=SMOOTH);
```

Table 4-286 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Timer
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize from synchronization position
	Leading axis position:	300 mm
Dynamic response:		
	Velocity:	300 mm/s
	Deceleration:	1000 mm/s²
	Jerk:	10000 mm/ s³
	Velocity profile:	Constant

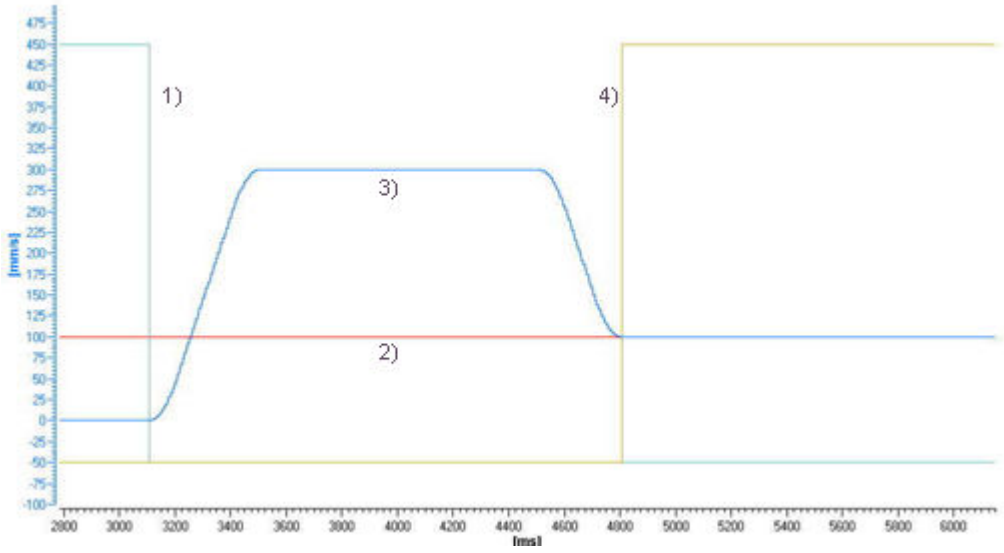
Note

For a motion with a continuous velocity profile, the enable of the jerk-limited synchronization **syncingMotion.smoothAbsoluteSynchronization:=YES** with absolute synchronous operation relationships should be set on the synchronous object.



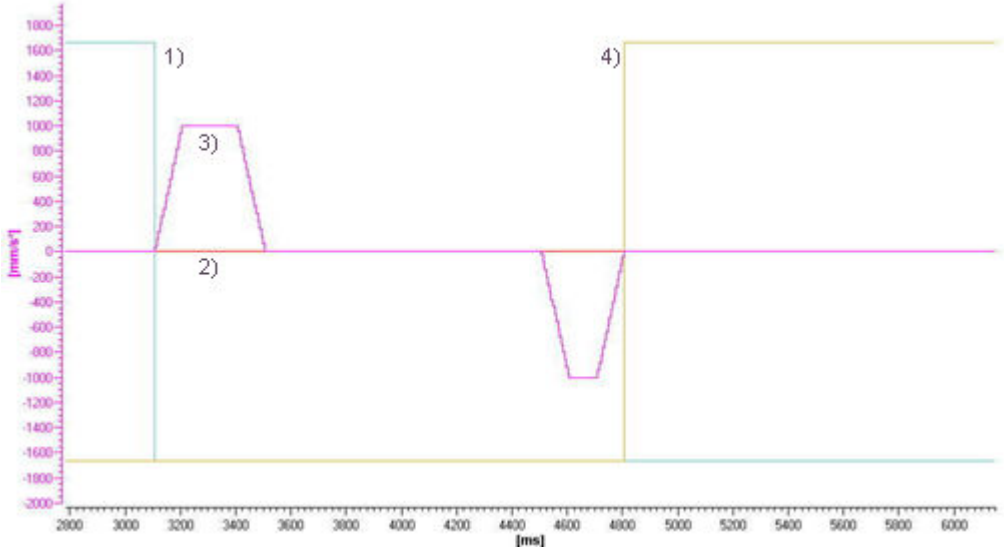
1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 4-700 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 4-701 Master and following axis velocity



1) Start of synchronization
2) Master axis acceleration
3) Following axis acceleration
4) Synchronization status

Figure 4-702 Master and following axis acceleration

Absolute synchronization with time reference, leading synchronization

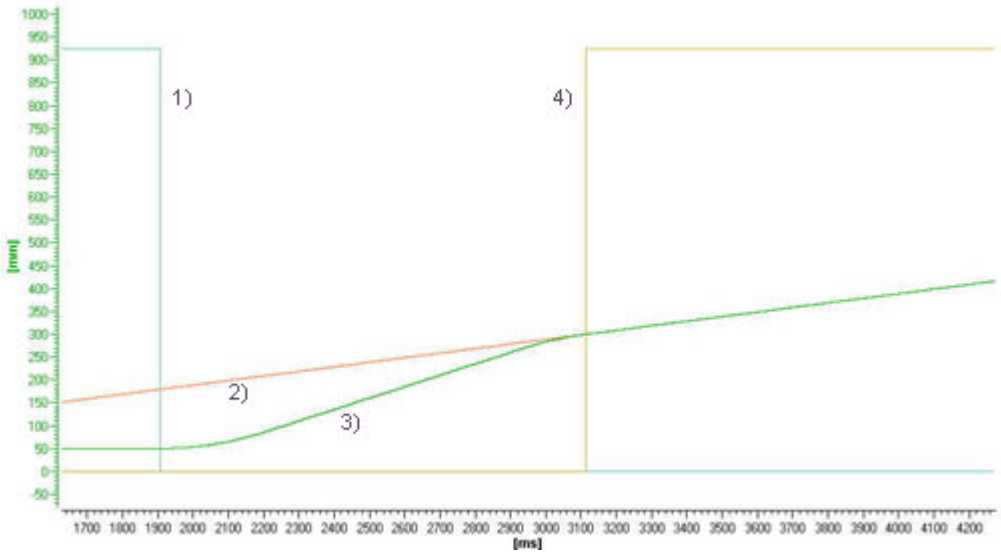
The master axis moves at a velocity of 100 mm/s. The following axis is stationary at position 50 mm. The specified dynamic response parameters (velocity = 300 mm/s and acceleration = 1,000 mm/s^2) are used for synchronization so that, with a master axis position of 300 mm, absolute synchronism exists between the master and following axes. For the synchronization procedure, a maximum change in the master value velocity of 20% is permitted on the synchronous object (**syncingMotion.maximumOfMasterChange**).

Table 4-287 ST programming

```
retval:= _enablegearing(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    direction:=POSITIVE,
    gearingType:=ABSOLUTE,
    gearingMode:=GEARING_WITH_FRACTION,
    gearingNumerator:=1,
    gearingDenominator:=1,
    synchronizingMode:=ON_MASTER_POSITION,
    syncPositionReference:=BE_SYNCHRONOUS_AT_POSITION,
    syncProfileReference:=RELATE_SYNC_PROFILE_TO_TIME,
    syncPositionMasterType:=DIRECT,
    syncPositionMaster:=300.0,
    velocityType:=DIRECT,
    velocity:=300.0,
    positiveAccelType:=DIRECT,
    positiveAccel:=1000.0,
    negativeAccelType:=DIRECT,
    negativeAccel:=1000.0,
    velocityProfile:=TRAPEZOIDAL);
```

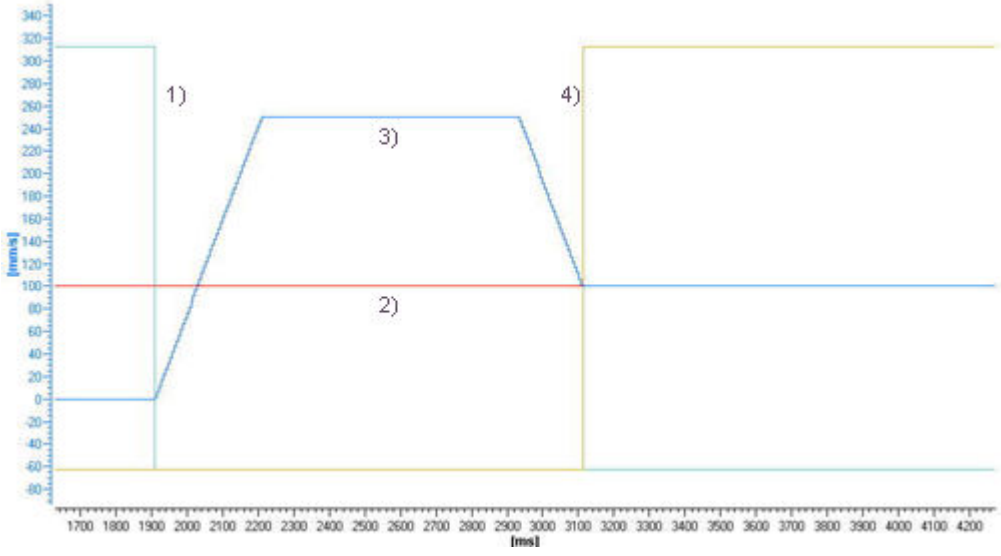
Table 4-288 MCC programming

Parameters:		
	Gear ratio:	1:1
	Reference point:	gearing is achieved based on the axis zero point
Synchronization:		
	Synchronization reference:	Timer
	Start of synchronization:	at leading axis position
	Reference point of the leading axis position:	Synchronize before synchronization position
	Leading axis position:	300 mm
Dynamic response:		
	Velocity:	300 mm/s
	Deceleration:	1000 mm/s^2
	Velocity profile:	Trapezoidal



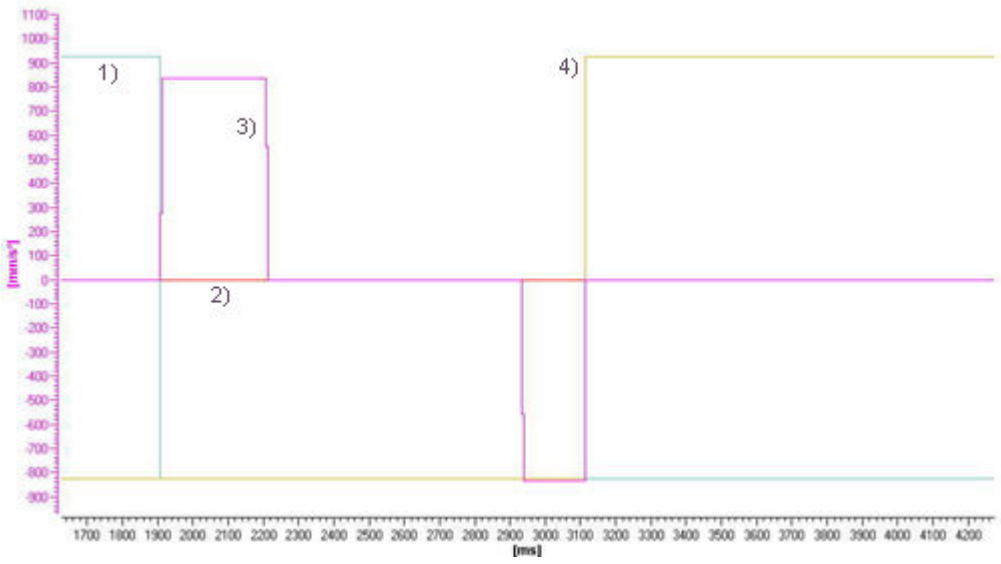
1) Start of synchronization
2) Master axis position
3) Following axis position
4) Synchronization status

Figure 4-703 Master and following axis position



1) Start of synchronization
2) Master axis velocity
3) Following axis velocity
4) Synchronization status

Figure 4-704 Master and following axis velocity



1) Start of synchronization
2) Master axis acceleration
3) Following axis acceleration
4) Synchronization status

Figure 4-705 Master and following axis acceleration

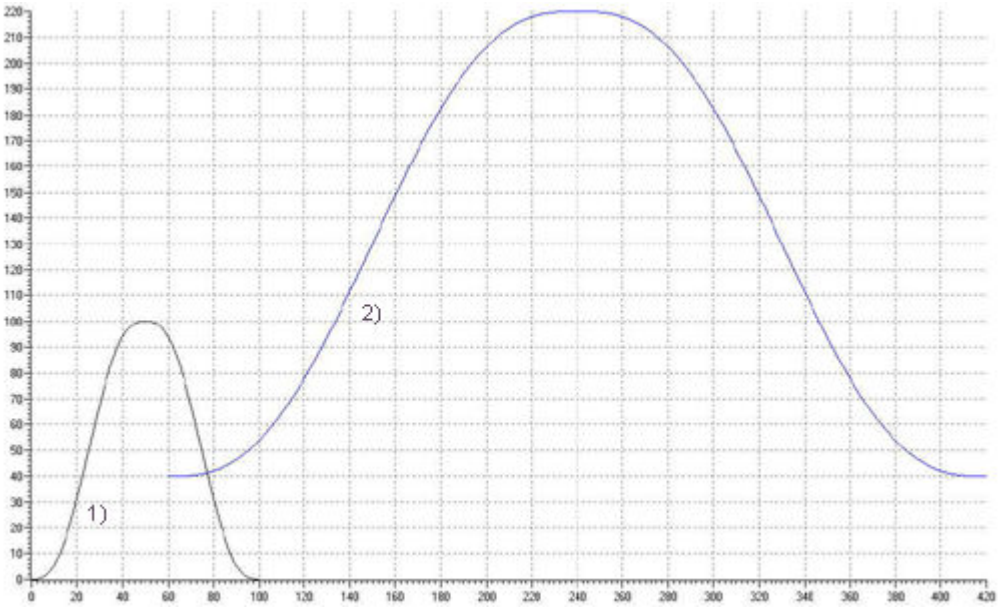
Example of offset and scale on the synchronous object

A leading axis provides position values over a range of 360° and 60° phase offset, i.e. from 60...420°. The following axis should move in the range 40...220°.

Definition range of cam: 0...100
Range of cam: 0...100

The definition and value range of the cam can be adapted to the required representation range for the camming using scaling and offset as follows (See also Figure **Equation for scale and offset on the camming** in Chapter **Camming**):

	Scaling	**Offset**
Master value	360 / 100 = 3,6	60
Slave value	(220 - 40) / 100 = 1,8	40



1) unscaled, non-offset function

2) scaled, offset function

Figure 4-706 Example of the scaling and the offset of a cam

In the following programming example, the offset and scale commands for the synchronous object take effect when camming is next activated:

Table 4-289 ST programming

```
(*scaling of the master value*)
    retval:= _setCammingScale(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    scalingRange:= MASTER_RANGE,
    scaleValue:= 3.6,
    activationMode:= DEFAULT_VALUE
);
(*scaling of the slave value*)
retval:= _setCammingScale(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    scalingRange:= SLAVE_RANGE,
    scaleValue:= 1.8,
    activationMode:= DEFAULT_VALUE
);
(*offset of the master value*)
retval:= _setCammingOffset(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    offsetRange:= MASTER_RANGE,
    offsetMode:= ABSOLUTE,
    offsetValue:= 60.0,
```

```
    activationMode:= DEFAULT_VALUE
);
(*offset of the slave value*)
retval:= _setCammingOffset(
    followingObject:= <SYNCHRONOUS_OBJECT>,
    offsetRange:= SLAVE_RANGE,
    offsetMode:= ABSOLUTE,
    offsetValue:= 40.0,
    activationMode:= DEFAULT_VALUE
);
```

MCC programming

Table 4-290 <Set scaling on camming>:master value side

Parameters:		
	Range:	Master Range
	Offset:	3.6
	Effect:	on following commands

Table 4-291 <Set scaling on camming>:slave value side

Parameters:		
	Range:	slave range
	Offset:	1.8
	Effect:	on following commands

Table 4-292 <Set offset on camming>:master value side

Parameters:		
	Range:	Master Range
	Offset:	60.0
	Mode:	Absolute
	Effect:	on following commands

Table 4-293 <Set offset on camming>:slave value side

Parameters:		
	Range:	slave range
	Offset:	40.0
	Mode:	Absolute
	Effect:	on following commands

See also

Camming (Page 2638)

Example of applying offset as superimposition

The following example shows the two dynamic corrections of the **dynamicReference** parameter in their different effects using an accelerating master value.

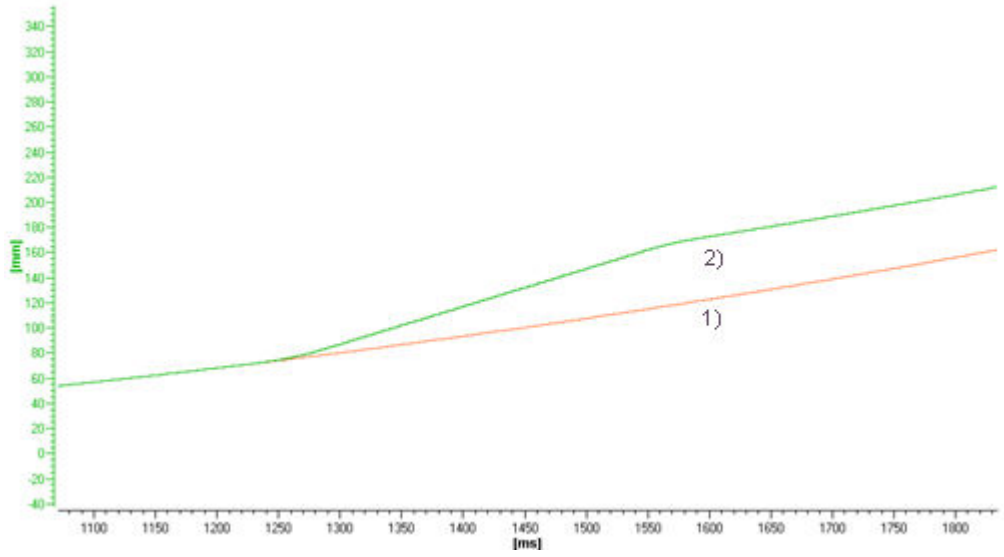
Example: Master and following axis for a synchronous operation of 1:1 are absolutely synchronous, accelerate at 100 mm/s². The offset of -50 mm on the master value side is implemented using _setGearingOffset for a programmed correction velocity of 300 mm/s and a correction acceleration of 3,000 mm/s², by means of a non-continuous-acceleration velocity profile.

Table 4-294 ST programming

```
retval := _setGearingOffset (
    followingObject:= <SYNCHRONOUSOBJECT>,
    offsetRange := MASTER_RANGE,
    offsetMode := RELATIVE,
    offsetValue := -50.0,
    velocityType := DIRECT,
    velocity := 300.0,
    positiveAccelType := DIRECT,
    positiveAccel := 3000.0,
    negativeAccelType := DIRECT,
    negativeAccel := 3000.0,
    velocityProfile := TRAPEZOIDAL,
    activationMode := ACTUAL_VALUE,
    dynamicReference := TOTAL_MOVE / OFFSET_MOVE
);
```

Note

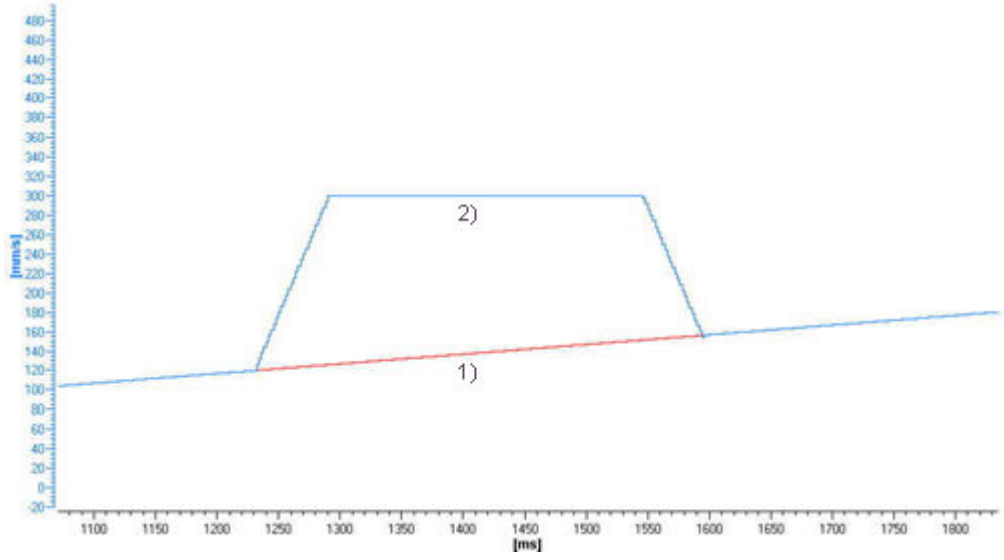
When the master value velocity is constant, the dynamic transitions have a generally identical form and differ only as a result of the dynamic response parameters that act differently.



1) Master axis position

2) Following axis position

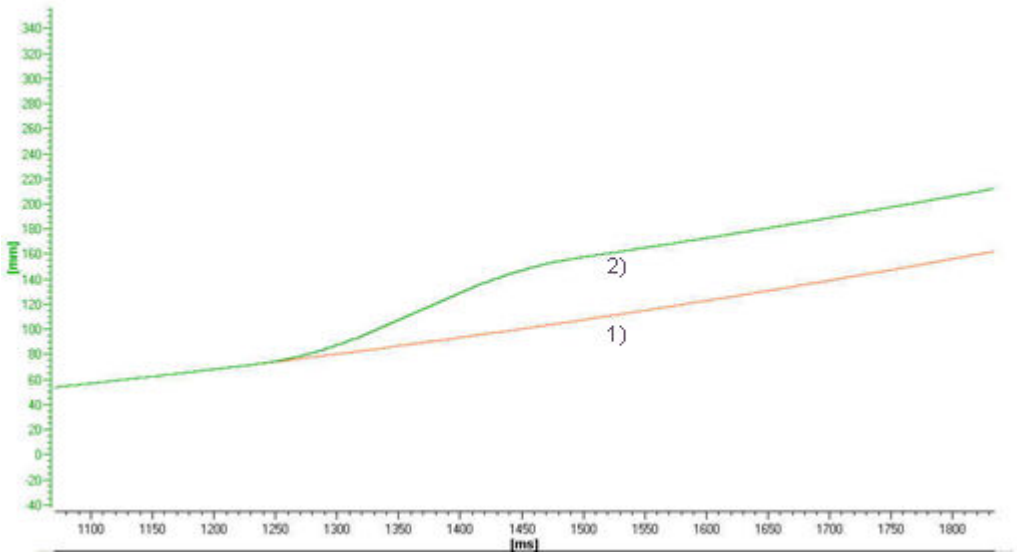
Figure 4-707 Leading and following axis position for dynamicReference:= TOTAL_MOVE



1) Master axis velocity

2) Following axis velocity

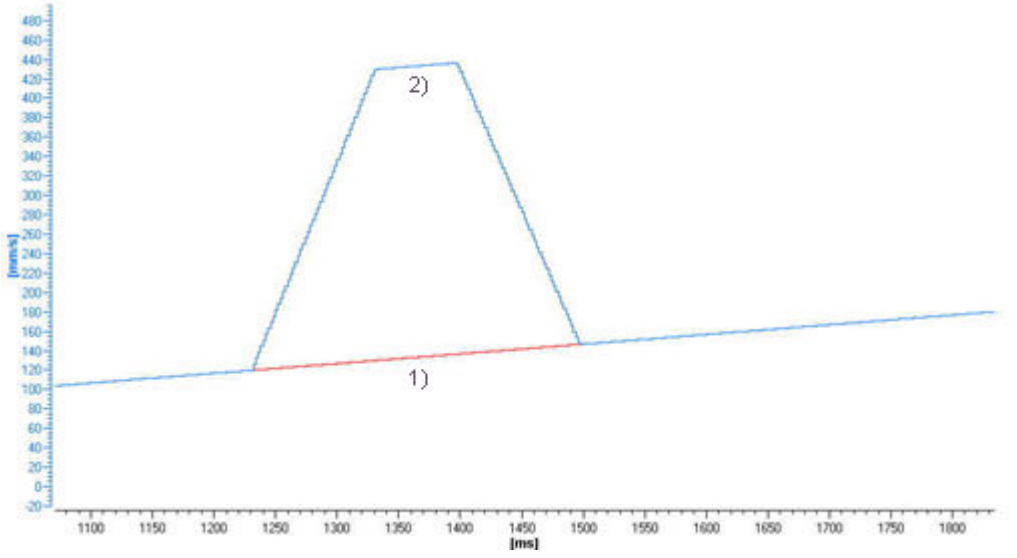
Figure 4-708 Leading and following axis velocity for dynamicReference:= TOTAL_MOVE



1) Master axis position

2) Following axis position

Figure 4-709 Leading and following axis position for dynamicReference:= OFFSET_MOVE



1) Master axis position

2) Following axis position

Figure 4-710 Leading and following axis velocity for dynamicReference:= OFFSET_MOVE

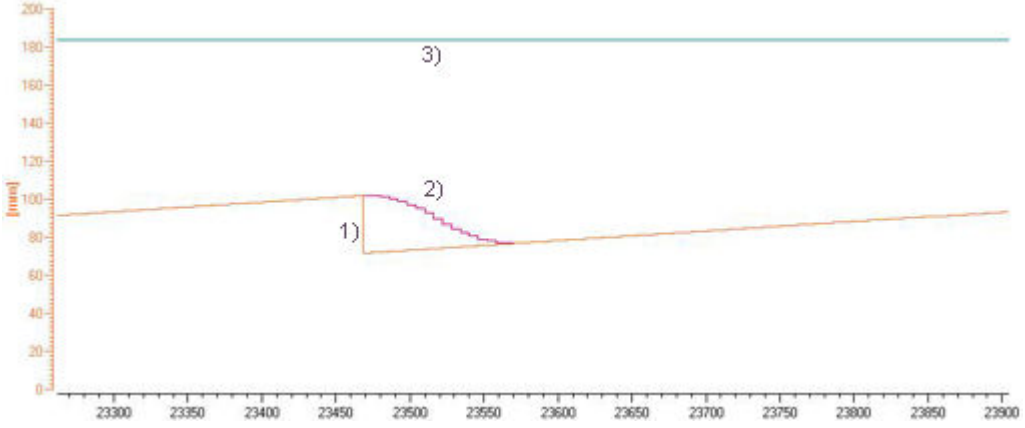
Special actions

Redefining the axis position during active synchronous operation

The following options are available for resetting an axis position, e.g. by means of **_redefinePosition()** or **_homing()**:

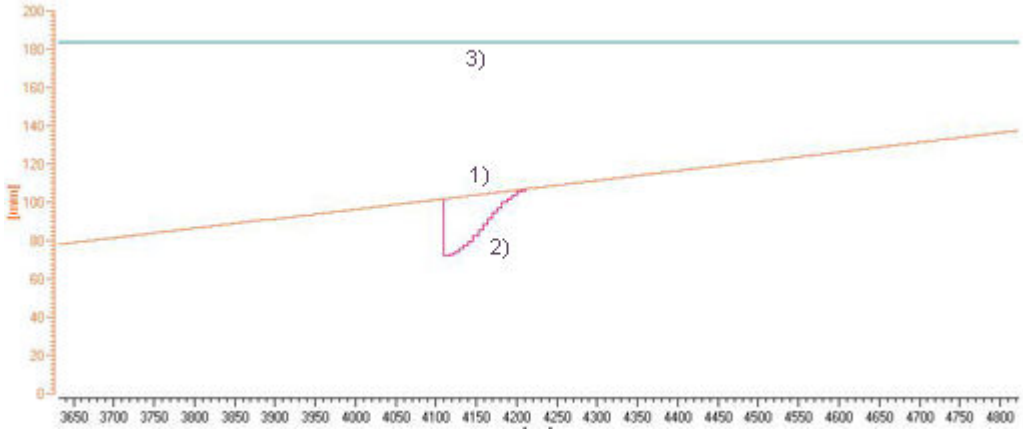
- Redefining the *master axis position* causes a jump in the master value.
  The following axis then performs a compensating movement and finally moves in synchronism again with the master axis. If the position tolerance > the synchronous operation tolerance, the error 40201 "Synchronous operation tolerance on gearing axis exceeded" will be output.
- Redefining the *following axis position* does *not* cause a jump on the master value.
  - For the *absolute synchronous operation*, the slave axis is no longer position-synchronous and so performs a compensating movement.
  - For the *relative synchronous operation*, the following axis does *not* perform a compensation motion, because position synchronization is not necessary.

Examples:



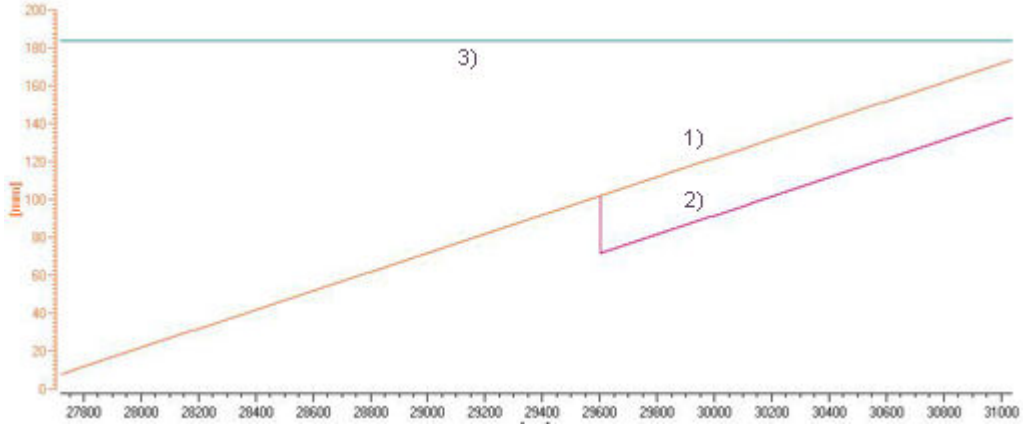
1) Master axis position
2) Following axis position
3) Synchronization status

Figure 4-711 Redefining the leading axis position (absolute or relative gearing) -> The following axis performs a compensation motion



1) Master axis position
2) Following axis position
3) Synchronization status

Figure 4-712 Redefining the following axis position for the absolute gearing -> The following axis performs a compensation motion



1) Master axis position
2) Following axis position
3) Synchronization status

Figure 4-713 Redefining the following axis position for the relative gearing -> The following axis does not perform a compensation motion

Retaining a synchronous connection for _disableAxis

If the slave axis is no longer able to apply the generated synchronous operation setpoints, e.g. due to the removal of enables or an error response, an active synchronous operation connection is closed. It is possible to retain the synchronous operation connection by using synchronous operation in simulation mode and setting the **DecodingConfig.disableSynchronousOperation=YES** configuration data element on the synchronous object.

See Simulation mode (Page 2690).

The existing synchronous connection remains active except in the following situations:

- Restart of following axis (**_restartAxis()**)
- Invalid actual values on the following axis

When simulation mode is terminated, the current synchronous operation setpoints are applied immediately as axis setpoints.

Remove the axis from the synchronous operation interconnection, and return it

Example:
Opening and closing protective doors

Removing an axis from the synchronized group

1. Stop the leading axis.
2. Switch the synchronous operation to simulation mode.
3. **DecodingConfig.disableSynchronousOperation** on the synchronous object must be set to **YES**. As a result, the synchronous operation connection is not disconnected on **_disableAxis**.
4. If a superimposed motion had been performed (e.g. for correcting):
   - Save the position of the superimposed coordinate system in a user variable.
   - Perform **_redefinePosition** in the coordinate system 2 to absolute position 0.
5. Cancel the controller enables (**_disableAxis()**). This will clear the drives. The axis goes into follow-up mode, "control" on the axes is deleted.

Returning the axis to the synchronized group

1. Reconnect the controller enables (**_enableAxis()**).
2. If necessary, use **_redefinePosition** to pass the saved actual position as absolute position to the coordinate system 2.
3. Switch synchronous operation back from simulation mode.

A compensation motion will be performed if the setpoints for the synchronous object and the setpoints on the axis do not match.

Substitution of velocity gearing with absolute synchronous operation

A synchronous velocity operation cannot be substituted directly with an absolute synchronous operation. Immediately afterwards, only a relative synchronous operation can be executed.

If an attempt is made to execute an absolute synchronous operation immediately after a velocity gearing, the following technological alarm is output:
50110 "Call-up of an absolute synchronous position operation after a synchronous velocity operation not permitted" (as of V3.2)

Procedure:

- Switch a relative position-controlled synchronous operation in between for one IPO cycle. (mergeMode = IMMEDIATELY and then a **waitTime** with Time = 0 sec)

Motion with absolute synchronous operation

Substitution of a position axis by an absolute synchronous operation (as of V4.4)

A position axis traversing in speed-controlled mode (_move(movingMode = SPEED_CONTROLLED)) can be substituted directly by an absolute synchronous operation. As a consequence of this, for example, with a stationary master value, the following axis positions itself at the absolute position of the master value.

Canceling active and pending synchronous operations

If gearing or camming is active and another gearing or camming operation is started, the first **_disableCamming()/_disableGearing()** command ends the synchronous operation to be synchronized, and the second **_disableCamming()/_disableGearing()** command ends the active synchronous operation.

Alternatively, in V4.1 and higher the synchronous operation commands can be canceled using the commandId associated with the command. The **_cancelFollowingObjectCommand()** command can be used to cancel the synchronous operation command by specifying the commandId in the **commandToBeCancelled** parameter. This removes the synchronous operation command from the command buffer. This means that a pending command for synchronization can be canceled (for example, if the application recognizes that resynchronization is not intended to take place).

Adapt the synchronization velocity to the master value velocity

The following axis cannot be synchronous with the master object if the velocity of the following axis for the synchronization process is lower than that of the master value during the synchronization process.

Example: The master axis is, for example, currently travelling at 200 mm/s. A gearing is started for which the synchronization velocity is limited by the default value of 100 mm/s. The following axis cannot overtake the master object.

To adapt the synchronization velocity of the following axis to the master value velocity, proceed as follows:

1. Create a **master axis**.
2. Create a **following axis** (synchronized axis).
   A **Following_Axis_SYNCHRONOUS_OPERATION** object is automatically created below the following axis.
   The project navigator should look like this:

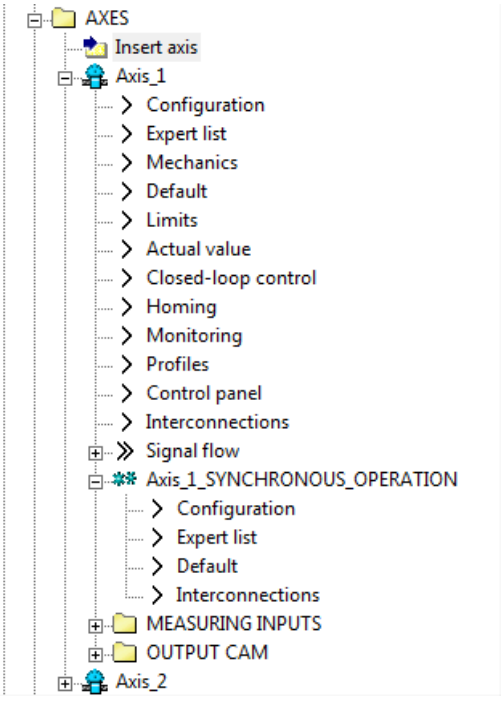


Figure 4-714 Axes for example

3. Make the following settings for **FollowingAxis_SYNCHRONOUSOPERATION -> Settings**:
   – **Overdrive factor for dynamic values**: 150 %

– **Adapting the dynamic values for synchronization**: activated

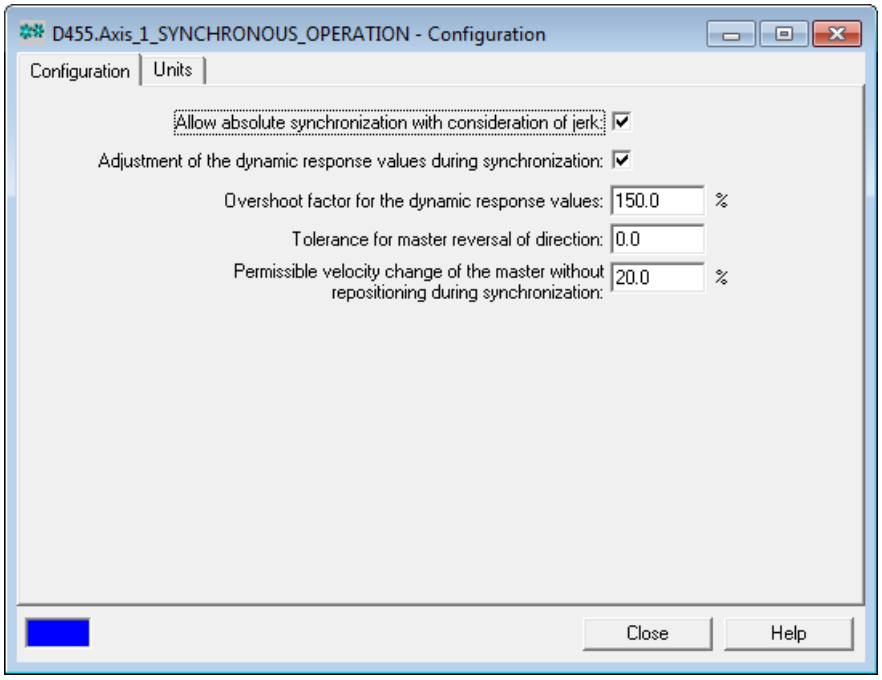


Figure 4-715 Settings for FollowingAxis_SYNCHRONOUSOPERATION

For the synchronization, the following axis uses the dynamic response values of the master axis (increased by the **overdrive factor**) and travels with the maximum velocity of 300 mm/s.

Another application scenario

The master object travels with the constant velocity of 80 mm/s. The dynamic response setting of the following axis allows a synchronization velocity of 100 mm/s. Under these conditions, the following axis can synchronize itself without adaptation of the dynamic response values. If the master object accelerates to 150 mm/s, but the **synchronous** status is not yet attained, the following axis can synchronize itself only with adaptation of the dynamic response values. During the synchronization action, the maximum velocity of the following axis is calculated using the current set velocity of the master object (150% of the current value).

Substitution of synchronous operation through _move, _pos, _stop with reset of an existing acceleration to zero

Description

When a motion is replaced with the **_stop()**, **_pos()** and **_move()** functions and a smooth velocity profile with jerk specification is applied during the axis acceleration phase, the velocity can continue to increase until the acceleration is reduced by means of the jerk that has been set. In extreme cases, the velocity can rise still further until it reaches the configured maximum velocity. The axis is only decelerated once acceleration has been reduced by the jerk.

With the **abortAcceleration** parameter, you can set that a possible acceleration does not exceed the jerk limitation, but is immediately cleared.

The **abortAcceleration** parameter is available for **_stop()** as of V4.2.1 and for **_pos** and **_move** as of V4.4.

You will find more information in the *TO Axis Electric/Hydraulic, External Encoder* Function Manual.

4.7.2.3 Synchronous Operation Configuration

This section shows you how to create and configure axes with **synchronous operation** in SIMOTION SCOUT.

It is assumed that you have already created cams, master axes, or external encoders.

Note

If the actual values/setpoints are required to be equal during a synchronous operation, the same Ti (actual value acquisition) and To (setpoint acceptance) times must be adopted for all drive units used (e.g. SINAMICS Integrated, CU320).

When configuring a synchronous operation, you must follow the steps below:

- Create an axis with synchronous operation functionality (Page 2722).
- Assign master values and cams to the synchronized axis (Page 2723).
- Assign synchronous operation parameters (Page 2725).
- Define the settings for the synchronization procedure (Page 2741).
- Define synchronous operation monitoring (Page 2743).

Creating an axis with synchronous operation

Create a synchronous axis as follows:

1. To create an Axis TO with the Synchronous operation technology in **SCOUT**, double-click **Insert axis** under **AXES** in the project navigator. You can also copy an existing Axis technology object using the clipboard and insert it with another name.
2. Activate for creating the axis the **Synchronous operation** technology. A synchronous object will automatically be created.

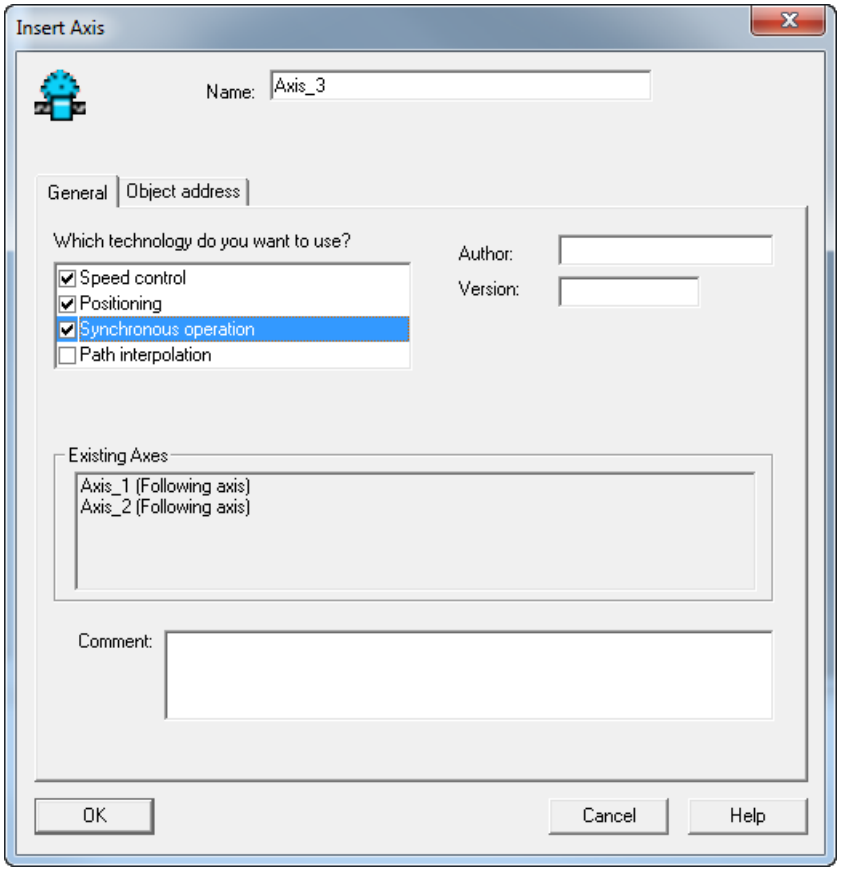


Figure 4-716 Inserting an axis with synchronous operation

Note

If synchronous operation technology is specified for an Axis technology object, the synchronous object will be inserted together with the Axis technology object. The synchronous object is permanently assigned to the Axis technology object. The name of the synchronous object is automatically defined and cannot be changed.

It is *not* possible to subsequently convert a positioning or speed axis to a following axis.

It is *not* possible to add a synchronous object. Only a superimposed synchronous operation can be added to the synchronous axis using expert (see Superimposed synchronous operation (Page 2683)).

Representation in the project navigator

The synchronous object is generated automatically when a synchronous axis is created and is displayed below this synchronous axis in the project navigator. The object is automatically given the name of the axis, followed by _SYNCHRONOUS_OPERATION. There the permitted synchronous operation relationships of the synchronous axis can be defined and the preassignments for the synchronous operation coupling to a leading axis assigned.

The assignment of the master values and cams is indicated in the project navigator using links:

- For the **Synchronous object**:
  Links to the master values (axes, external encoders, addition objects, formula objects, and fixed gears) as well as cams
- For the technology objects used:
  Link to the synchronous object
- Below the **master values** (axes, external encoders, addition objects, formula objects, and fixed gears): Link to the synchronous object

Superimposed synchronous operation

If a superimposed synchronous operation relationship to another master value is also to be established, an additional synchronous object can be placed below the following axis via the shortcut menu for the following axis (**Expert > Insert superimposed synchronous object**) (see Superimposed synchronous operation (Page 2683)).

The standard and superimposed synchronous operation relationships then affect the following axis additively by means of the two master values.

Assigning master values and cams

When an axis with synchronous operation is created, the synchronous operation configuration still has to be specified, i.e., the master values to be used must be selected and, if required, a cam must be assigned.

Note

Synchronous operation is not possible if a master value is not assigned. Camming is not possible if a cam is not assigned.

Defining the synchronous operation configuration

- In the project navigator, double-click on **Interconnections** under the object <Axis name>**_SYNCHRONOUS OPERATION**. The window below opens.

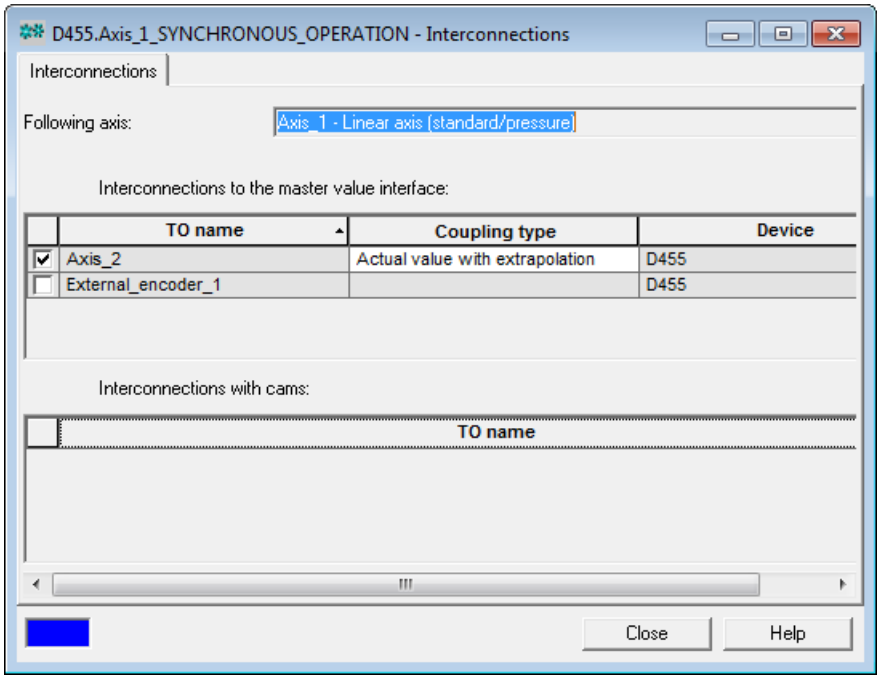


Figure 4-717 Selection of master values and cams

Assign the master values and cams to the following axis in this window.

You can set the following parameters:

Table 4-295 Parameters for configuring synchronous operation

Field/Button	Meaning/Instruction
Following axis	The name of the following axis is displayed here.
Possible master values (leading axis)	The master values available in the project, which you can assign to the following axis, are listed here. The master value can be specified by the following technology objects: • Axis (real or virtual axis) • External encoder • Fixed gear • Addition object • Formula object In accordance with the specified synchronous operation condition (e.g. camming), the slave value is calculated on the basis of the master value and assigned to the following axis as a leading value. When more than one master value is assigned, you must specify which master value is to be used by means of programming in SIMOTION SCOUT (e.g. with MCC).
Possible cams	Cams created in the project are listed here. You can assign cams to the synchronous object for camming. When more than one cam is assigned, you must specify which cam is to be used by means of programming in SIMOTION SCOUT (e.g. with MCC).

1. Assign the desired **master values** to the axis with synchronous operation. You select the current master value to be used in the user program (**_setMaster**).
2. Assign the desired **cams** to the axis with synchronous operation. You select the cam to be used in the user program (**_enableCamming**).
3. For real axes or external encoders, select **setpoint coupling** or **actual value coupling** for axes, or select **actual value coupling** or **actual value coupling with extrapolation** for external encoders.
   See Setpoint/actual value coupling (Page 2647).

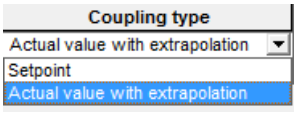


Figure 4-718 Selection of the coupling type

When the window is **closed**, the configuration is accepted and saved automatically.

Assigning parameters/defaults for synchronous operation

In the project navigator, double-click **Default** under the object <Axis name>**_SYNCHRONOUS OPERATION** to change the default settings.

Synchronous operation - Default

In this window, you define the replacement values (default) for calling the synchronous operation functions (**_enableGearing**, **_enableVelocityGearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGearing**, and **_disableCamming**).

These replacement values are only evaluated if no special settings are made in the associated function calls for synchronization/desynchronization (ST and LAD or MCC).

You can set parameters for the following functionality:

- Gearing (Page 2727)
- Velocity gearing (Page 2728)
- Camming (Page 2729)
- Gearing synchronization (Page 2731)
- Camming synchronization (Page 2735)
- Dynamic response (Page 2738)
- Master dynamic response (Page 2740)

Note

All tabs are always available for default settings. Only parameters used for the relevant functionality are evaluated.

Further information

- For information about this function, refer to Overview of synchronous operation and Fundamentals of synchronous operation.
- For information about programming, refer to Synchronous operation programming/ references (Page 2744).
- The meaning of the dialog window parameters and their permissible value ranges can be found in the **SIMOTION reference lists**.

Gearing

Gearing is characterized by a constant coupling between the master value source and following axis. This coupling (via the gear ratio) can be specified as the ratio of two decimal numbers (numerator/denominator) or as a floating-point number.

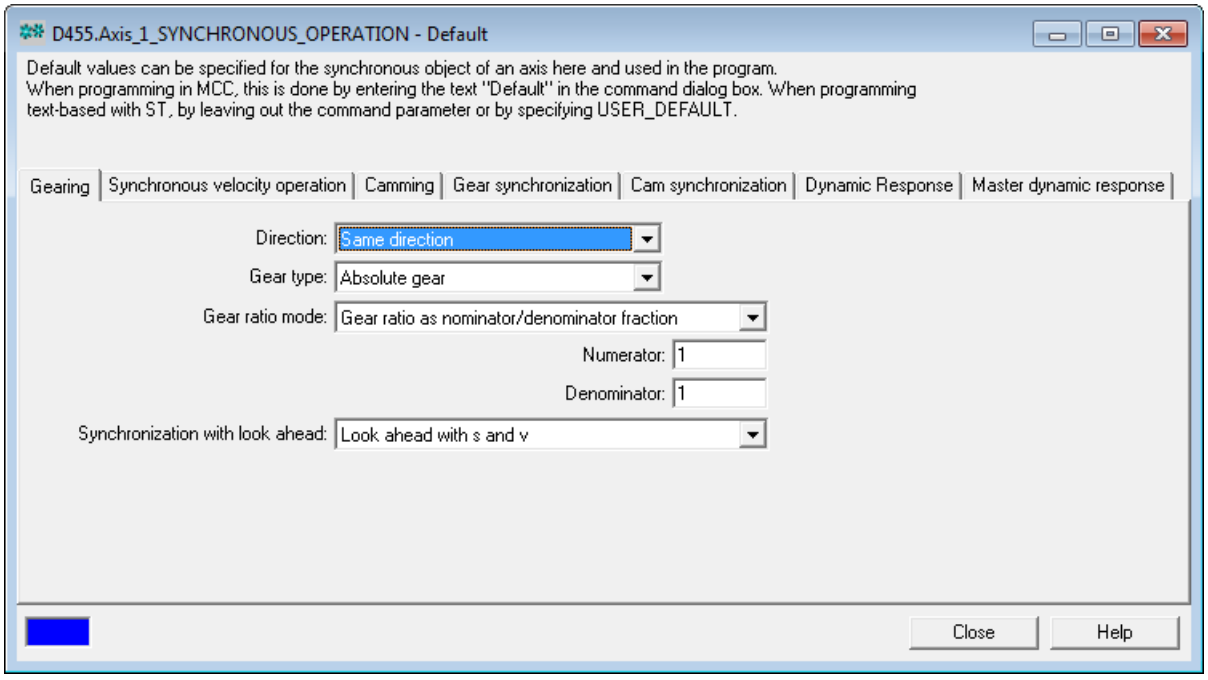


Figure 4-719 Synchronous object: Default setting for gearing

Gearing - default

In the **Gearing** tab, select the **direction**, **absolute** or **relative synchronous operation** and the **gear ratio**. These settings are only relevant when **gearing** mode is used.

You can set the following parameters:

Table 4-296 Parameters for gearing

Field/Button	Meaning/Instruction
Direction	Here, you specify the direction of the gearing.
Gear type	Here, you select the gear type (absolute or relative).
Gear ratio mode	Specify the gear ratio mode here. Further parameters are displayed depending on the selected mode (gear ratio as floating-point number or gear ratio as numerator/denominator ratio).
Gear ratio	You can enter the gear ratio as a floating-point number here.
Counter instructions	Here, you can enter the numerator of the gear ratio in the form of a numerator/denominator ratio.
Denominator	Here, you can enter the denominator of the gear ratio in the form of a numerator/denominator ratio.
Synchronization with look-ahead	You can set here whether a constant acceleration/deceleration of the master value is to be used for the synchronization.

See also

Gearing (Page 2631)

Velocity gearing

In contrast to gearing or camming, which relate to the position of an axis, velocity gearing relates to the velocity of an axis with a constant coupling between the master value source and the following axis. With velocity gearing, a non-position-controlled, i.e. speed-controlled synchronous operation with the velocity of a master axis, is activated on an axis that is set as a position-controlled axis (synchronized axis).

Possible master values are:

- Velocity of a master axis configured as a position-controlled axis
- Velocity of an external encoder

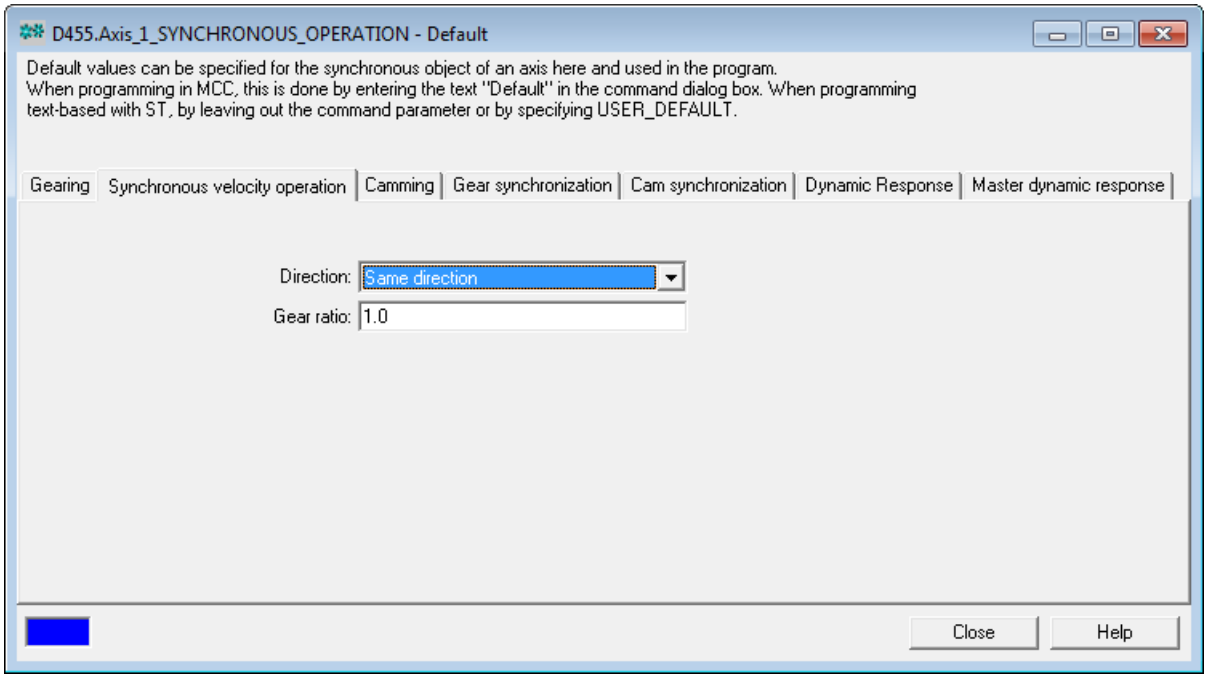


Figure 4-720 Synchronous object: Default setting for synchronous velocity operation

Velocity gearing - default

On the **Velocity gearing** tab, you set the **direction** and the **gear ratio**. These settings are only relevant when **velocity gearing** mode is used.

You can set the following parameters:

Table 4-297 Parameters for velocity gearing

Field/Button	Meaning/Instruction
Direction	Specify the direction of the synchronous velocity operation here.
Gear ratio	Enter here a coupling ratio for the synchronous velocity operation as floating-point number.

See also

Velocity gearing (Page 2637)

Camming

Camming is characterized by variable coupling between the master value source and the following axis. The coupling is described by a cam (transmission function). Camming can be scaled and offset on both the master value source side and on the following axis side. This enables a cam to be custom-adjusted in its definition range and value range.

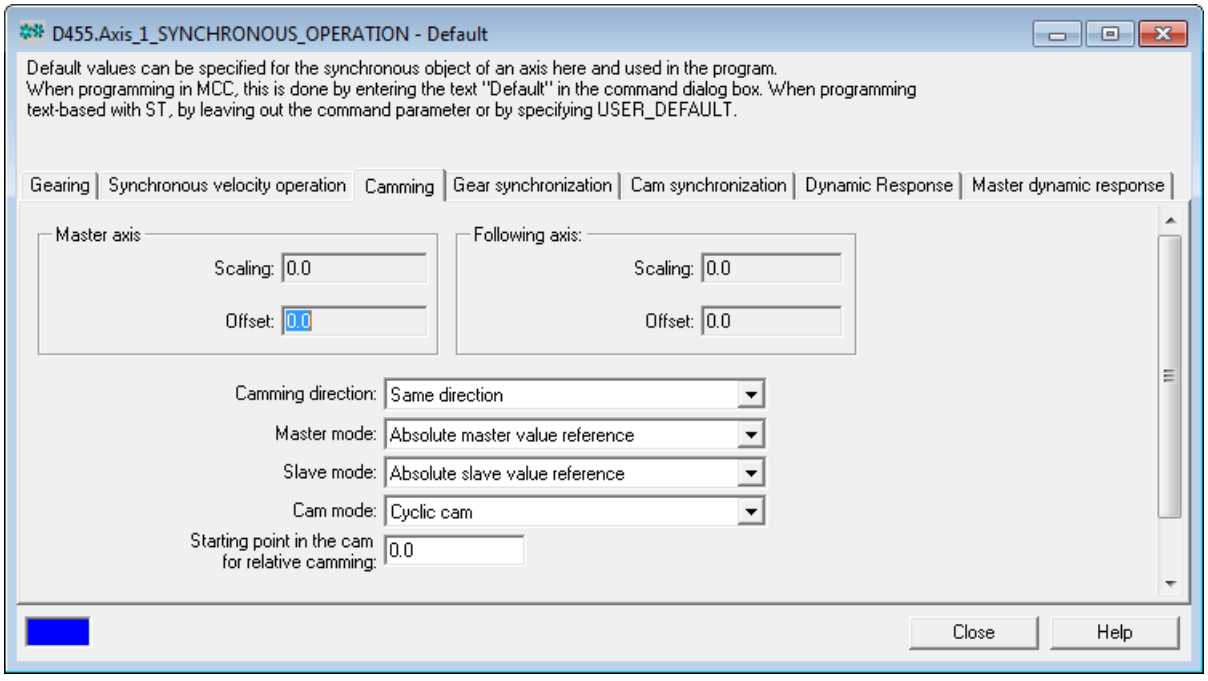


Figure 4-721 Synchronous object: Default setting for camming

Camming - Default

On the **Camming** tab, the **scaling** and **offset** for each master axis and following axis are shown. The **Scaling** and **Offset** setting can be made on the cam or via **_setCammingScale** or **_setCammingOffset**. You select the **direction**, **absolute** or **relative synchronous operation** for the master value and the slave value, as well as the **cam mode**. These settings are only relevant when **camming** mode is used.

You can set the following parameters:

Table 4-298 Parameters for camming

Field/Button	Meaning/Instruction
Leading axis	
Scaling	The scaling of the master value is displayed here.
Offset	The offset of the master value is displayed here.
Following axis	
Scaling	The scaling of the following axis is displayed here.
Offset	The offset of the slave axis is displayed here.
Camming direction	Specify the direction in which the cam is run.
Master mode	Specify the mode in which the master value runs through the cam (absolute or relative).
Slave mode	Specify the mode in which the following axis runs through the cam (absolute or relative).
Cam mode	Specify the execution type for the cam here (cyclic or non-cyclic).
Starting point in the cam for relative camming	Specify here for relative master value reference the reference of the master values by specifying a starting position (camStartPositionMaster) in the cam. The position is always an absolute reference in relation to the definition range of the cam.

See also

Camming (Page 2638)

Gearing synchronization

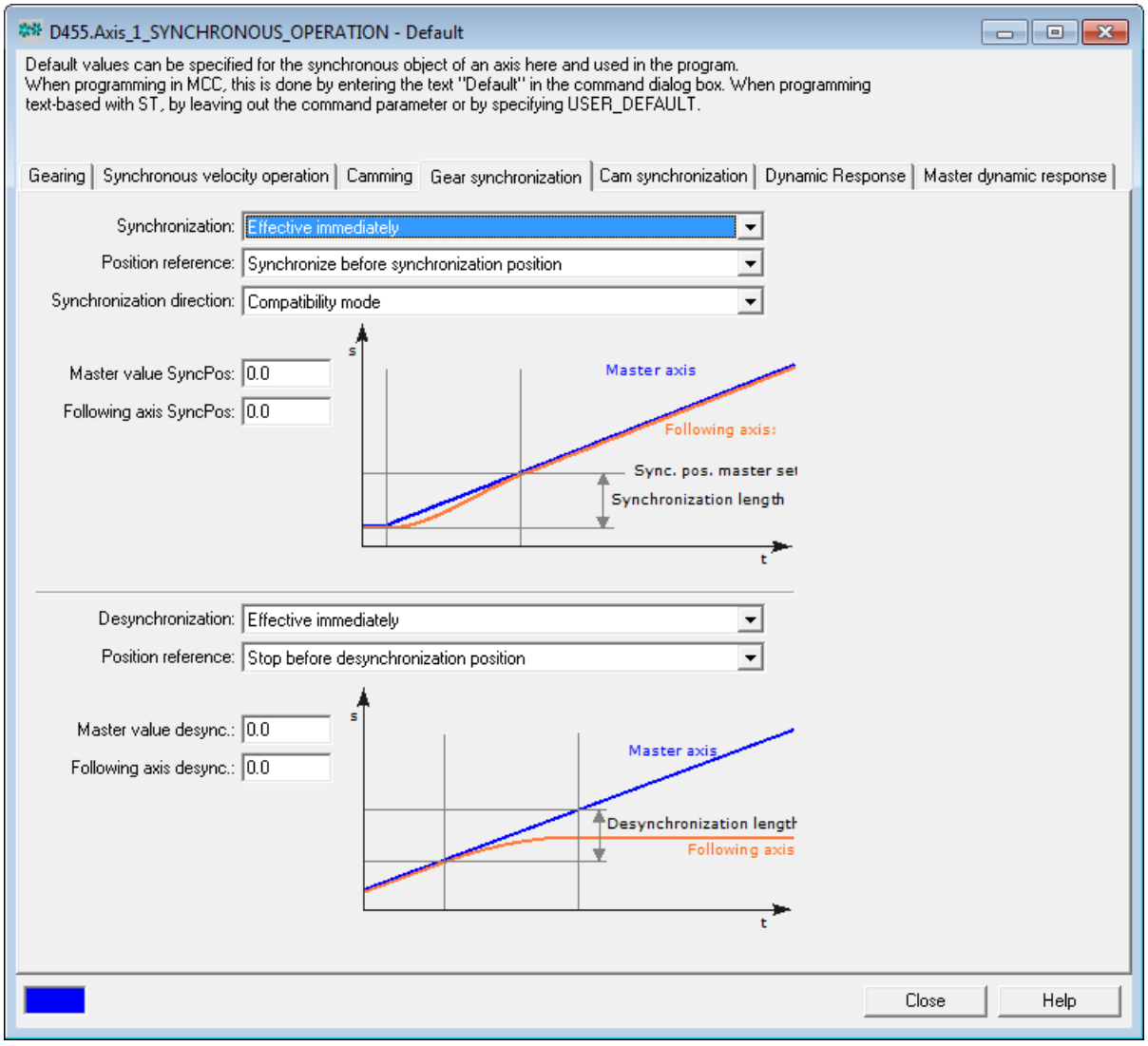


Figure 4-722 Synchronous object: Default setting for gear synchronization

Gear Synchronization - Default

You use the **Gear synchronization** tab to set the parameters for **synchronization and desynchronization**. These settings are only relevant when **gearing** mode is used.

You can set the following parameters:

Table 4-299 Parameters for gear synchronization

Field/Button	Meaning/Instruction
Synchronization	Specify when synchronization of the following axis with the master value is performed. For more information, see Synchronizing the gear (Page 2652)
Position reference	Here, you specify the position of the synchronization profile relative to the position of the synchronization point. For more information, see Position reference during synchronization (Page 2733)
Synchronization direction	Specify the synchronization direction here. See Synchronization direction (Page 2658).
Master value SyncPos	Here, you enter the position of the synchronization point of the master value.
SyncPos slave axis	Here, you enter the position of the synchronization point of the slave axis
Desynchronization	Specify when desynchronization of the following axis with the master value is performed. For more information, see Desynchronizing the gear (Page 2734).
Position reference	Here, you specify the position of the desynchronization profile relative to the position of the desynchronization point For more information, see Position reference during desynchronization (Page 2734).
Desync. Master value	Here, you enter the position of the desynchronization point of the master value.
Desync. Following axis	Here, you enter the position of the desynchronization point for the following axis

Synchronizing the gear

You can set the synchronization condition using the **Synchronization** drop-down list box on the **Gear synchronization** tab:

Table 4-300 Parameters for gear synchronization

Setting	Meaning
Effective immediately	Synchronization takes immediate effect. The start point is derived from the position of the current master value. The settings in **Master value SyncPos** and **Following axis SyncPos** are not evaluated.
Default synchronization position of the leading axis	This setting is only appropriate for an absolute master value coupling. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. The setting in **Following axis SyncPos** is not evaluated.
Specified by the synchronization position of the following axis	This setting is only appropriate for an absolute following axis. The synchronization criterion depends on the position of the following axis. The synchronization position is specified in **Following axis SyncPos**. The setting in **Master value SyncPos** is not evaluated.

Setting	Meaning
Default synchronization position of the leading axis and following axis	This setting is only appropriate for an absolute master and an absolute following axis. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. An offset is additionally created on the following axis by the setting in **Following axis SyncPos**. That is, the following axis is not synchronized to the programmed slave position but to the position **Following axis SyncPos** plus absolute position value of the following axis.
Last programmed setting	Setting of the last active command

See also

Synchronization criterion/synchronization position (Page 2654)

Position reference during synchronization

How synchronization is to performed is set via the **Position reference** picklist:

Table 4-301 Position reference parameters for synchronization

Setting	Meaning
Synchronize from synchronization position	Synchronization starts at the synchronization position. The following axis runs synchronously after the synchronization length resulting from the dynamic response data.
Synchronize before synchronization position	The synchronization is performed so that the following axis runs synchronously to the master value at the synchronization position. If the following axis is stopped, it is accelerated before the synchronization position so that it runs synchronously to the master value at the synchronization position. The position at which the axis is accelerated is the synchronization position minus the synchronization length.
Synchronize symmetrically to synchronization position	The synchronization is performed so that the synchronization position is exactly in the middle of the synchronization length. Therefore, a stopped following axis is already accelerated before the synchronization position and only runs synchronously to the master value after half the synchronization length.
Last programmed setting	Setting of the last active command

You will find further information in the section Position of synchronization range relative to synchronization position (Page 2660).

Desynchronizing the gear

You can set the position at which desynchronization is to start using the **Desynchronization** drop-down list box on the **Gear synchronization** tab:

Table 4-302 Parameters for desynchronization

Setting	Meaning
Effective immediately	Desynchronization takes immediate effect.
Specified by the desynchronization position of the leading axis	The desynchronization is performed as of the **Master value desync** value. The setting in **Following axis desync** is not evaluated.
Default desynchronization position of the following axis	The desynchronization is performed as of the **Following axis desync** value of the following axis. The setting in **Master value desync** is not evaluated.
Last programmed setting	Setting of the last active command

See also

Desynchronization criterion/desynchronization position (Page 2676)

Position reference during desynchronization

How desynchronization is to performed is set via the **Position reference** picklist:

Table 4-303 Parameters for the position reference

Setting	Meaning
Stop after desynchronization position	The desynchronization is started at the desynchronization position. The desynchronization is completed after the synchronization length resulting from the dynamic response data.
Stop before desynchronization position	The desynchronization is performed so that the desynchronization is completed at the desynchronization position. The position at which the desynchronization is started is the desynchronization position minus the desynchronization length.
Stop symmetrically to desynchronization position	The desynchronization is performed so that the desynchronization position is exactly in the middle of the desynchronization length.

See also

Position of synchronization range relative to desynchronization position (Page 2677)

Camming synchronization

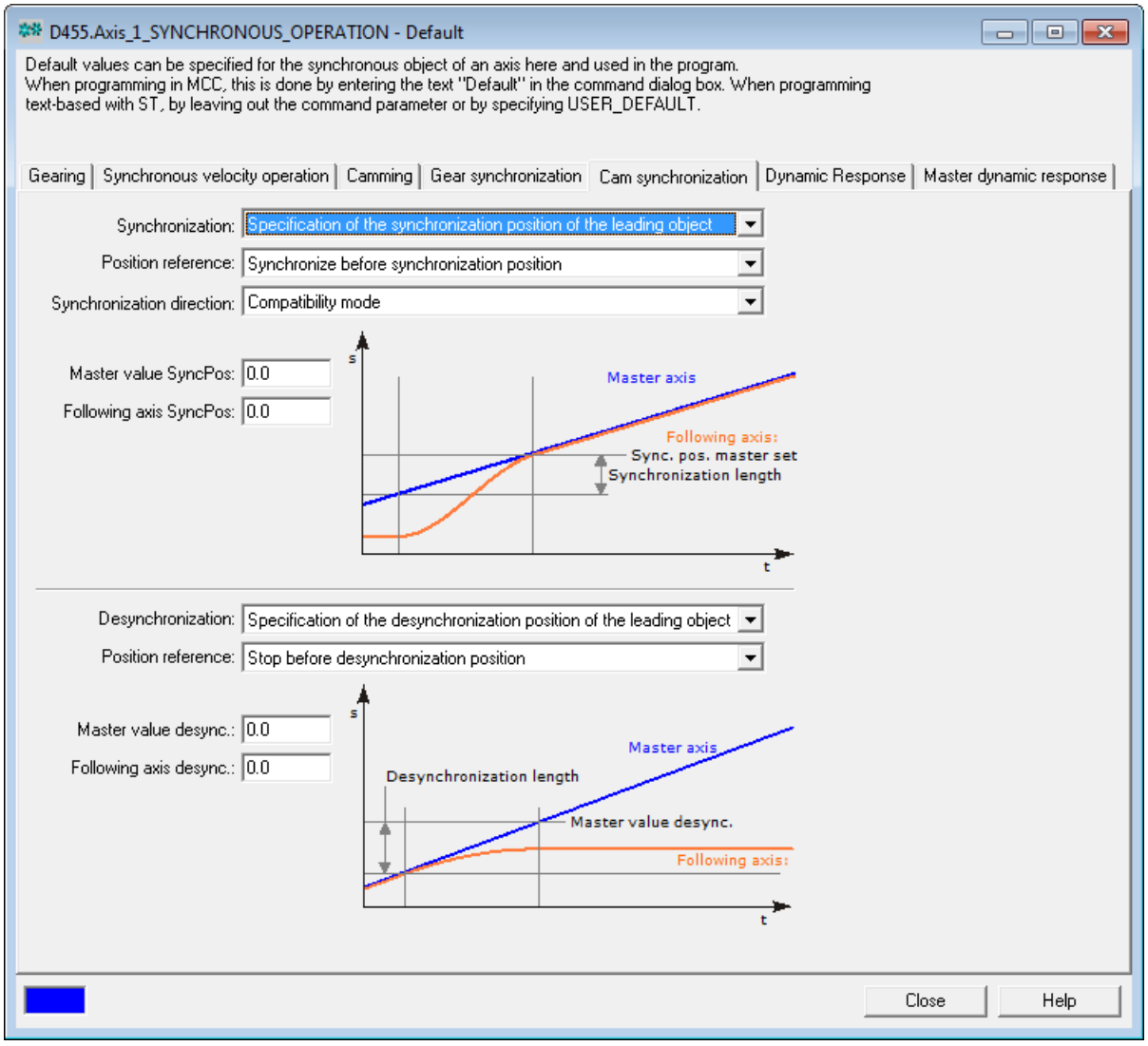


Figure 4-723 Synchronous object: Default setting for cam synchronization

Cam Synchronization - Default

You use the **Cam synchronization** tab to set the parameters for **synchronization and desynchronization**. These settings are only relevant when **camming** mode is used.

You can set the following parameters:

Table 4-304 Parameters for camming synchronization

Field/Button	Meaning/Instruction
Synchronization	Specify when synchronization of the following axis with the master value is performed. For more information, see Synchronizing the curve (Page 2736).
Position reference	Here, you specify the position of the synchronization profile relative to the position of the synchronization point. For more information, see Position reference during synchronization (Page 2733)
Synchronization direction	Specify the synchronization direction (Page 2658) here.
Master value SyncPos	Here, you enter the position of the synchronization point of the master value.
SyncPos slave axis	Here, you enter the position of the synchronization point of the slave axis
Desynchronization	Specify when desynchronization of the following axis with the master value is performed. For more information, see Desynchronizing the curve (Page 2737).
Position reference	Here, you specify the position of the desynchronization profile relative to the position of the desynchronization point For more information, see Position reference during desynchronization (Page 2734).
Desync. Master value	Here, you enter the position of the desynchronization point of the master value.
Desync. Following axis	Here, you enter the position of the desynchronization point for the following axis

See also

Synchronization (Page 2652)

Cam synchronization

You can set the synchronization condition using the **Synchronization** drop-down list box on the **Cam synchronization** tab:

Table 4-305 Parameters for synchronization

Setting	Meaning
Effective immediately	Synchronization takes immediate effect. The start point is derived from the position of the current master value. The settings in **Master value SyncPos** and **Following axis SyncPos** are not evaluated.
Default synchronization position of the leading axis	This setting is only appropriate for an absolute master value coupling. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. The setting in **Following axis SyncPos** is not evaluated.

Setting	Meaning
Transition at the end of the active cam	This setting is only possible for a relative master value coupling. The synchronization criterion is the master setpoint position at the end of the current cam disk cycle. The setting in **Following axis SyncPos** is not evaluated.
Default synchronization position of the leading axis and following axis	This setting is only appropriate for an absolute master and following axis. Synchronization depends on the position of the master value. The synchronization position is specified in **Master value SyncPos**. An offset is additionally created on the following axis by the setting in **Following axis SyncPos**. That is, the following axis is not synchronized to the programmed (e.g. via cam) slave position but to the position **Following axis SyncPos** plus absolute position value of the following axis in relation to the cam.
Last programmed setting	Setting of the last active command

See also

Synchronization criterion/synchronization position (Page 2654)

Cam desynchronization

You can set the position at which desynchronization is to start using the **Desynchronization** drop-down list box on the **Cam synchronization** tab:

Table 4-306 Parameters for desynchronization

Setting	Meaning
Effective immediately	Desynchronization takes immediate effect.
At position of the leading axis	The desynchronization is performed as of the **Master value desync** value. The setting in **Following axis desync** is not evaluated.
At position of the following axis	The desynchronization is performed as of the **Following axis desync** value of the following axis. The setting in **Master value desync** is not evaluated.
End of cam cycle	The desynchronization is performed at the end of the current cam cycle.
Last programmed setting	Setting of the last active command

See also

Desynchronization criterion/desynchronization position (Page 2676)

Dynamic response

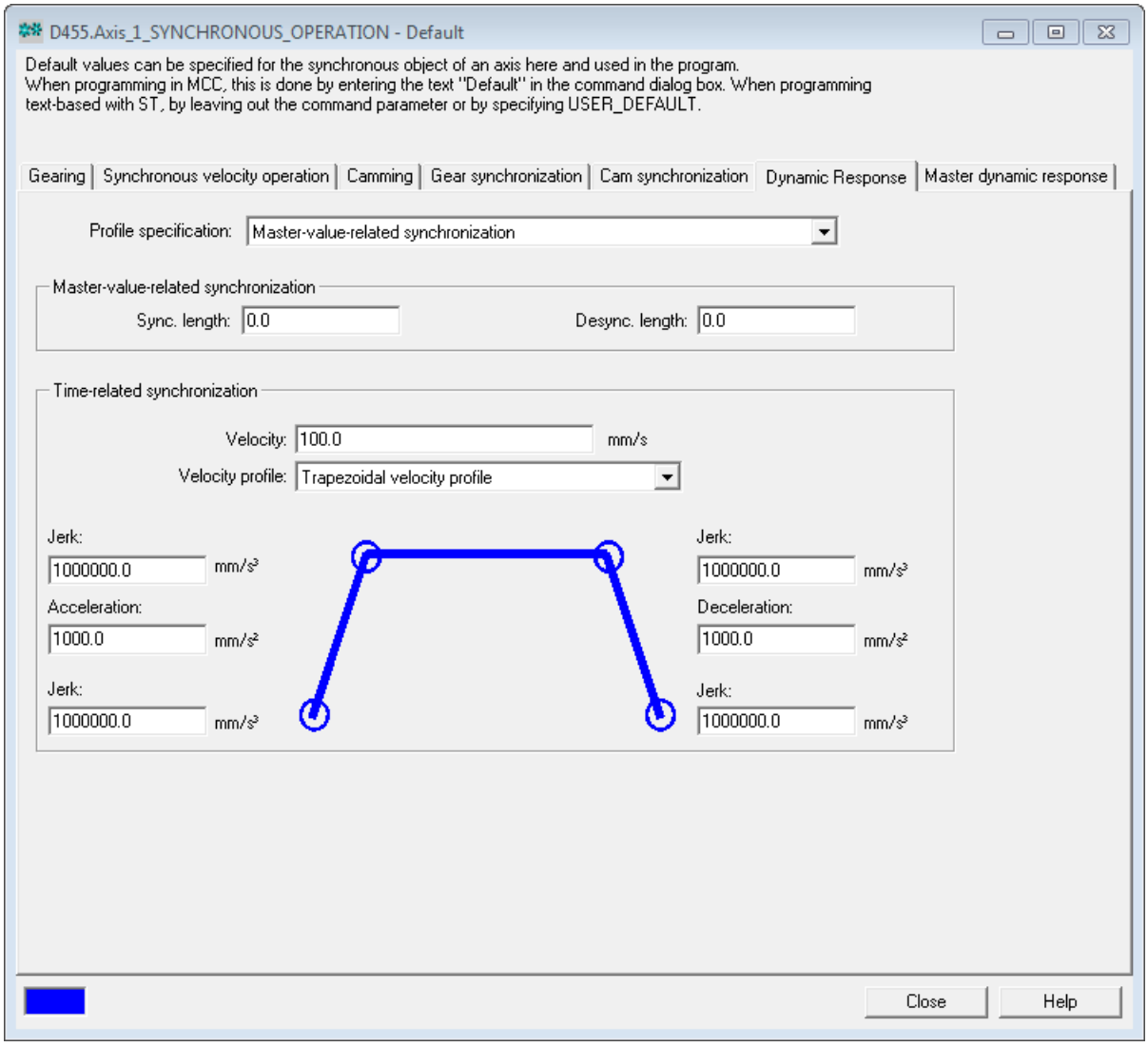


Figure 4-724 Synchronous object: Default setting for dynamic response

Dynamic response - Default

In the **Dynamics** tab, you make the basic settings for synchronization and desynchronization.

You can enter the following profile settings:

- **Length-related synchronization**; see Synchronization via a specifiable master value distance (Page 2692).
- **Time-related synchronization**; see Synchronization profile based on specifiable dynamic response parameters (Page 2694).

Dynamic response parameters are used for:

- Time-related synchronization
- Compensatory motions on the synchronous object

You can set the following parameters:

Table 4-307 Dynamic response parameters

Field/Button	Meaning/Instruction
Profile specification	Specify the reference for the synchronization profile here.
Leading axis-related synchronization	
Sync. Length	Here, you enter the path length for synchronization.
Desync. Length	Here, you enter the path length for desynchronization.
Time-related synchronization	
Velocity profile	Select the velocity profile here.
Velocity	Enter the maximum velocity here.
Acceleration	Enter the maximum acceleration here.
Deceleration	Enter the maximum deceleration here.
Jerk	Enter the maximum jerk here.

The synchronization length and desynchronization length are only evaluated for master-axis-related synchronization profiles. Velocity profile, velocity, acceleration, deceleration and jerk are only evaluated for time-related synchronization profiles.

See also

Synchronization (Page 2652)

Master dynamic response

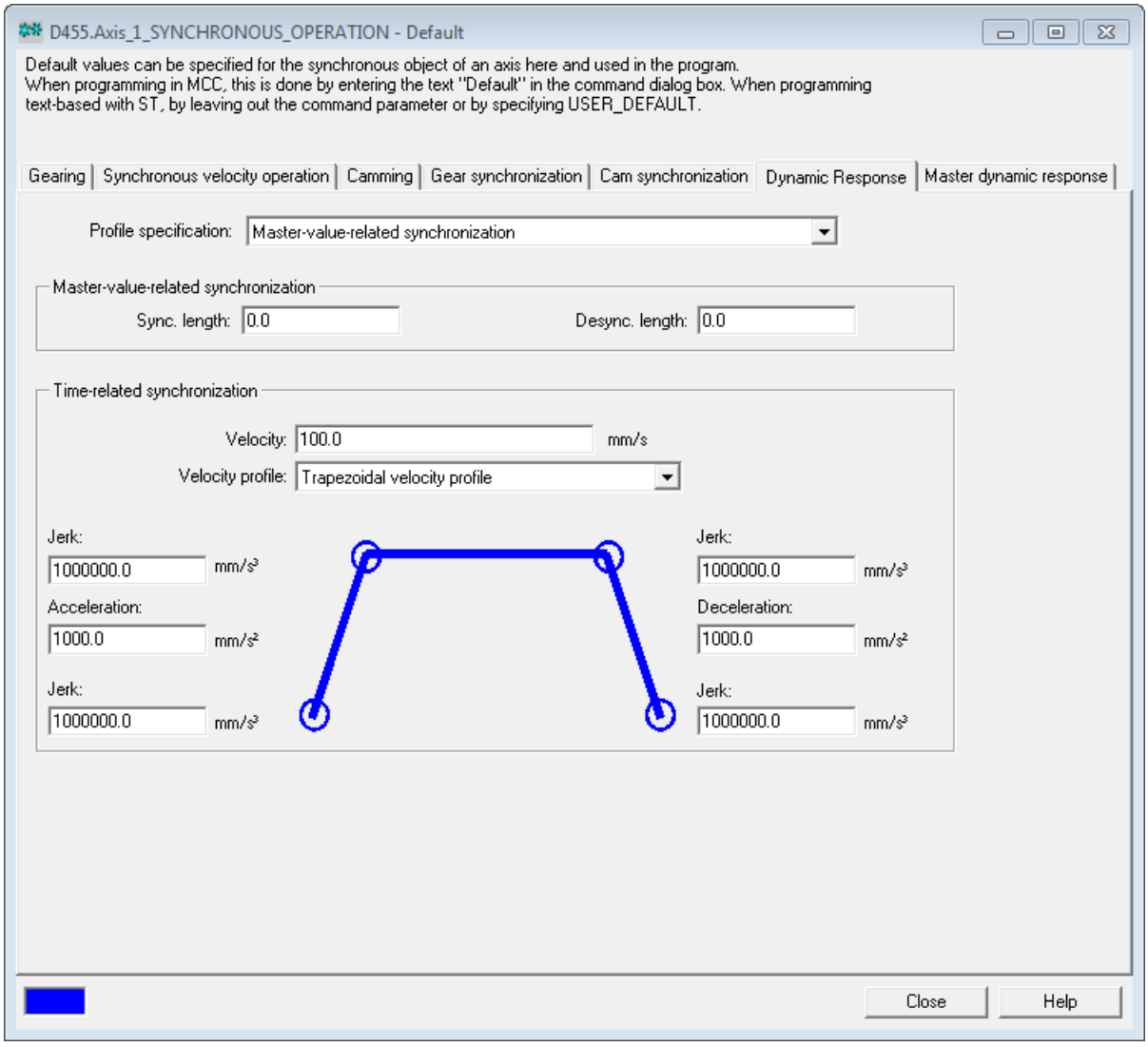


Figure 4-725 Synchronous object: Default setting for master dynamic response

Master dynamic response - default

In the **Master dynamic response** tab, you make the settings for the dynamic response for the master value switchover.

You can set the following parameters:

Table 4-308 Dynamic response parameters - Master

Field/Button	Meaning/Instruction
Master switchover with dynamic response values	
Velocity profile	Select the velocity profile here.
Velocity	Enter the maximum velocity here.

Field/Button	Meaning/Instruction
Acceleration	Enter the maximum acceleration here.
Deceleration	Enter the maximum deceleration here.
Jerk	Enter the maximum jerk here.

See also

Switching over the master value source – Overview (Page 2680)

Set synchronization

Certain settings for synchronization can be defined on the synchronous object.

- In the project navigator, double-click **Settings** under the synchronous object.

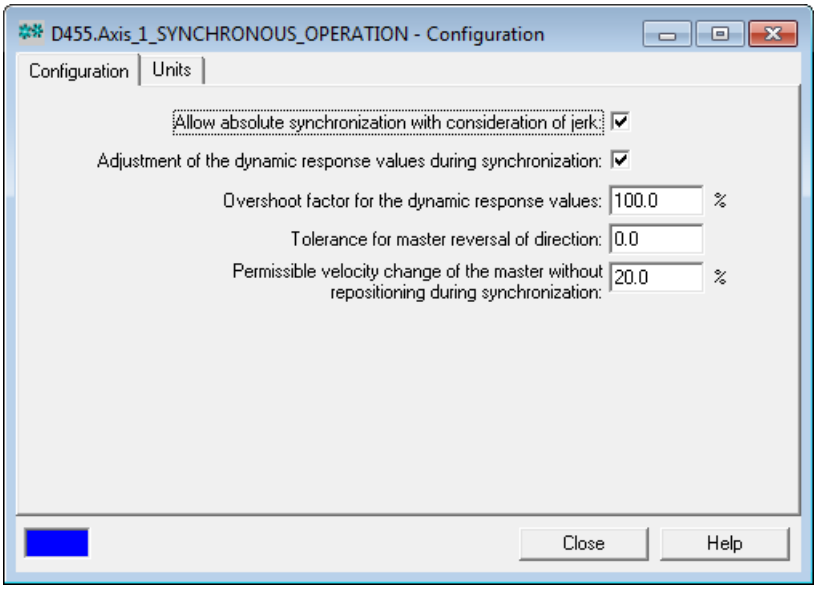


Figure 4-726 Settings on the synchronous object

Synchronous object - Settings

In this window, define the parameters for the synchronization.

Table 4-309 Synchronization parameters

Field/Button	Meaning/Instruction
Permit absolute synchronization with provision for jerk	For absolute synchronization, the "Yes" setting of the jerk is used. If "No" is set, the set jerk will not be used in spite of velocity profile = SMOOTH being selected. Travel follows a trapezoidal path. (syncingMotion.smoothAbsoluteSynchronization) See also section Trailing synchronization with synchronization profile based on dynamic response parameters (Page 2665).
Adapt the dynamic response values for the synchronization	Adaptation to the dynamic response in the synchronous position (syncingMotion.synchronizingAdaption) If "Yes" is set, the following parameter is available: See also section Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 2662).
Magnification factor for the dynamic response values	Overdrive factor for the adapted dynamic response values to compensate for a remaining path difference (syncingMotion.overdriveFactor) Value as percentage (%) Reference to the current master value velocity for the synchronization start. See also section Synchronization profile based on specifiable dynamic response parameters (time reference) (Page 2662) and section Adapt the synchronization velocity to the master value velocity (Page 2718).
Tolerance for master direction reversal	Tolerance window for canceling the synchronization for direction reversal of the master values (syncingMotion.masterReversionTolerance) Position value in the user unit of the master values. See also section Settings for evaluation of the master value behavior during synchronization (Page 2667).
Permitted velocity change of the master without restarting for synchronization	Maximum permitted change of the master value velocity (syncingMotion.maximumOfMasterChange) Refers to the current master value velocity for the synchronization start. Value as percentage (%). See also section Settings for evaluation of the master value behavior during synchronization (Page 2667).

See also

Dynamic response effect on slave values (Page 2678)

Synchronization (Page 2652)

Configuring synchronous operation monitoring

Synchronous operation monitoring between the master value/following object and the following axis can be configured on the synchronized axis.

1. In the project navigator, double-click **Monitoring** under the axis object.
2. Set the required parameters on the **Synchronous operation** tab.

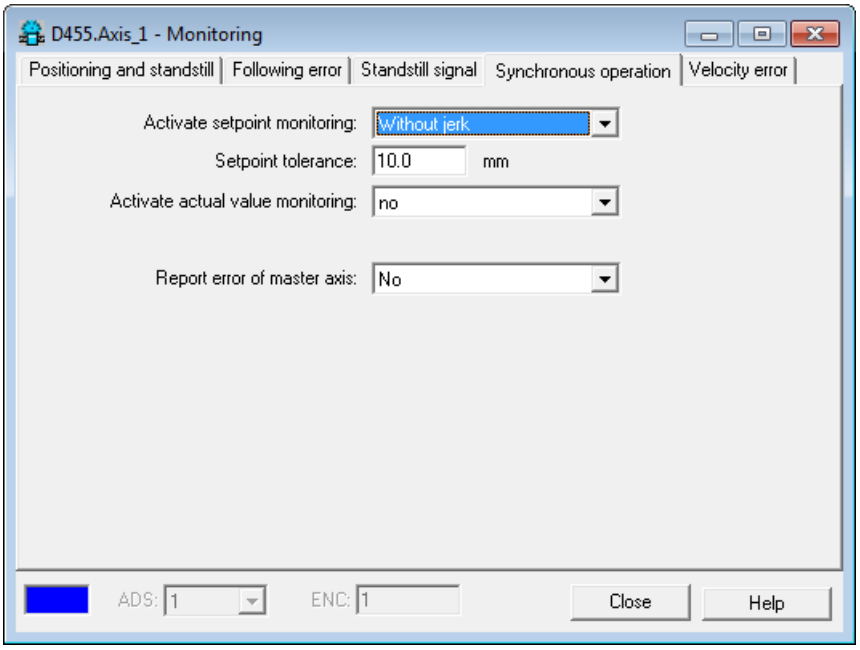


Figure 4-727 Monitoring of a following axis

Here, you specify the synchronous operation monitoring of the axis.

Table 4-310 Monitoring parameters

Field/Button	Meaning/Instruction
Activate setpoint monitoring	Activate the setpoint monitoring of the following axis here. (TypeOfAxis.GearingPosTolerance.enableCommandValue)
Setpoint tolerance	Specify the setpoint tolerance with activated setpoint monitoring. (TypeOfAxis.GearingPosTolerance.commandValueTolerance)
Activate actual value monitoring	Activate the actual value monitoring of the following axis here. (TypeOfAxis.GearingPosTolerance.enableActualValue)
Actual value tolerance	Specify the actual value tolerance with activated actual value monitoring. (TypeOfAxis.GearingPosTolerance.actualValueTolerance)
Signal error of the leading axis	Specify here which tolerance-exceeded messages (actual values / setpoints) are signaled to the leading axis. Take into account that the tolerance-exceeded messages are signaled to all higher-level leading axes (if, for example, the leading axis is also a following axis in a synchronous operation configuration). (TypeOfAxis.GearingPosTolerance.enableErrorReporting)

See also

Synchronous operation monitoring (Page 2687)

Error handling (Page 2754)

4.7.2.4 Synchronous Operation Programming/References

This chapter contains an overview of the **Synchronous Operation** technology object commands and information on the local alarm response. You will find a complete list of all commands and their syntax, the system variables, and error messages, in the SIMOTION Reference Lists.

See also

Overview of commands (Page 2744)

Command processing (Page 2749)

Error handling (Page 2754)

Menus (Page 2757)

Overview of commands

ST commands

Table 4-311 Commands on the synchronous object

Command type / command	Description
Information and conversion	See Commands for reading out function values (Page 2746).
_getMasterValue	The **_getMasterValue** command supplies the master value at a specified slave value position.
_getSlaveValue	The **_getSlaveValue** command supplies the slave value on a specified master value position.
_setMaster	The **_setMaster** command is used to assign a new master value source to a synchronous object. See Switching of the master value source (Page 2680).
Command tracking	See Commands for command tracking (Page 2747).
_getStateOfFollowingObjectCommand	The **_getStateOfFollowingObjectCommand** command returns the execution state of a synchronous operation command.
_getMotionStateOfFollowingObjectCommand	The **_getMotionStateOfFollowingObjectCommand** command returns a structure with the state of a synchronous operation command. The state of a synchronous operation command must then also be read out.

Command type / command	Description
_bufferFollowingObjectCommandId	The **_bufferFollowingObjectCommandId** command enables the **commandId** and the associated command status to be saved beyond the processing time of the command. The **commandId** parameter is used to define the command for which the respective status is to be saved. The maximum number of command states that can be saved is set in configuration data **decodingConfig.numberOfMaxBufferedCommandId**.
_removeBufferedFollowingObjectCommandId	The **_removeBufferedFollowingObjectCommandId** command discontinues saving of the **commandId** and the associated command status beyond the processing time of the command.
Motion	
_enableGearing	The **_enableGearing** command produces gearing with a constant transfer ratio. See Gearing (Page 2631).
_disableGearing	The **_disableGearing** command terminates gearing with a constant transfer ratio.
_enableVelocityGearing	The **_enableVelocityGearing** command produces velocity gearing with a constant coupling. See Velocity synchronous operation (Page 2637).
_disableVelocityGearing	The **_disableVelocityGearing** command terminates velocity gearing with a constant coupling.
_enableCamming	The **_enableCamming** command produces camming with a variable transfer ratio. See Camming (Page 2638).
_disableCamming	The **_disableCamming** command terminates camming with a variable transmission ratio.
Correction and superimposition	
_setGearingOffset	The **_setGearingOffset** command offsets the gearing with reference to the master value or the slave value (V3.1 and higher). See **Changing the offset** in chapter Gearing (Page 2631)
_setCammingScale	The **_setCammingScale** command scales the camming relative to the master value or the slave value. See **Scaling and offset** in Camming (Page 2638).
_setCammingOffset	The **_setCammingOffset** command offsets the camming with reference to the master values or the slave values. See **Scaling and offset** in Camming (Page 2638).
Object and Alarm Handling	See Commands for resetting of states and errors (Page 2749).
_resetFollowingObject	The **_resetFollowingObject** command resets the synchronous object, i.e. an active synchronous grouping will be cancelled, any pending errors will be cleared.
_resetFollowingObjectError	The **_resetFollowingObjectError** command acknowledges and resets errors on the synchronous object.
_resetFollowingObjectErrorState	The **_resetFollowingObjectErrorState** command acknowledges and resets the error status on the synchronous object.
_getFollowingObjectErrorNumberState	The **_getFollowingObjectErrorNumberState** command is used to fetch the status of a specific error
Simulation	See Simulation mode (Page 2690).

Command type / command	Description
_enableFollowingObjectSimulation	The **_enableFollowingObjectSimulation** command sets a synchronous operation into simulation mode.
_disableFollowingObjectSimulation	The **_disableFollowingObjectSimulation** command disables simulation mode of the synchronous grouping.

PLCopen commands

MultiAxis commands of PLCopen are relevant for synchronous operation. These commands are provided for use in cyclic programs/tasks and enable motion control programming in a view shaped by a PLC. They are used primarily in the LAD/FBD programming language. The commands are certified in accordance with "PLCopen Compliance Procedure for Motion Control Library V1.1".

Table 4-312 PLCopen commands for synchronous operation

Command	Description
_MC_CamIn	Synchronize and engage cam; included implicitly:
_MC_CamOut	Remove cam with desynchronization
_MC_GearIn	Synchronize synchronous operation
_MC_GearOut	Desynchronize synchronous operation
_MC_Phasing	Phase offset

For further information, refer to the **PLCopen Blocks** Function Manual.

MCC commands

MCC commands are available for synchronous operation. For information on this see the Programming and Operating Manual for the **SIMOTION MCC Motion Control Chart**.

Commands for reading out function values

Commands that calculate values and supply them as return values are available for reading out master value positions and slave value positions in pairs:

- The **_getSlaveValue** command supplies the slave value at a specified master value position. If several master values may have the same slave value, an approximate value can be specified for the master value.
  - **slavePositionType:=CURRENT** returns the current value.
  - **slavePositionType:=DIRECT** returns in **slavePosition** the specified approximation value.
- The **_getMasterValue** command supplies the master value at a specified slave value position. If several slave values may have the same master value, an approximate value can be specified for the slave value.
  - **masterPositionType:=CURRENT** returns the current value.
  - **masterPositionType:=DIRECT** returns the approximate value specified in **masterPosition**.

The commands only supply the correct position if a synchronous relationship is active and synchronized.

Commands for command tracking

The following commands are available for command tracking:

- The **_getStateOfFollowingObjectCommand** command returns the execution state of a synchronous operation command:
  - **ACTIVE**: The command is being executed.
  - **NON_EXISTENT**: The command is complete or the commandID is unknown.
  - **WAITING**: The command is decoded, but execution not yet started.
  - **WAITING_FOR_SYNC_START**: The command is waiting for synchronous start.
- The **_getMotionStateOfFollowingObjectCommand** command returns a structure with the execution state of a command.
  - **functionResult** specifies the error code.
  - **motionCommandIdState** returns the current phase of the motion of the queried command.
  - **abortId** specifies the reason for aborting the command.
    The reason for aborting is specified with alarm 30002 "Command aborted (reason: <abortId>, command type: ...)".
- With **_bufferFollowingObjectCommandId**, the command status can be queried after completion or an abort of the command.
- With **_removeBufferedFollowingObjectCommandId**, the command should be explicitly removed from the command management of the technology object after evaluation is complete.

The number of motion commands that the MotionBuffer can accept can be specified using the **followingObjectType.DecodingConfigInfo.numberOfMaxbufferedCommandId** configuration data.

System variables

The sequence of programmed commands can be read out by means of system variables; see Monitoring the synchronization (Page 2671).

System variable syncMonitoring:

The structure elements belonging to **syncMonitoring** indicate the monitoring functions and status information for synchronization.

The variable is used for the monitoring of the synchronous operation relationship to the synchronous operation setpoint, see also Synchronous operation monitoring (Page 2687).

- The **limitCommandValue** variable indicates whether the setpoint differences have exceeded the maximum limit for synchronous operation errors.
- The **differenceCommandValue** variable shows the difference between the setpoint generated at the synchronous object and the executable setpoint at the axis, with dynamic limits taken into account.
- The **limitActualValue** variable indicates whether the actual value differences have exceeded the maximum limit for synchronous operation errors.
- The **differenceActualVelocity** variable shows the pending position lag at the following axis, including the setpoint error, caused by any acting dynamic response of the following axis (**syncMonitoring.differenceCommandValue**). The dynamic response also acts on superimposed motions and can possibly trigger monitoring of the actual value. The position lag for DSC operation refers to the position lag present at the position controller in the drive. For versions prior to V4.2, the difference between the setpoint calculated on the synchronous object and the actual value present in the interpolator is indicated.
- The **syncErrorReported** variable indicates whether an alarm has already been issued to the master axis.
- The **syncState** variable indicates the synchronous operation status of the setpoint; it is syncState:=NO during synchronization and desynchronization. The setpoint synchronous operation error is the error that arises from limiting the setpoint to the maximum slave axis dynamic response values for the synchronous operation component.
  The **followingMotionState** indicates the state of the synchronous operation motions.
  The **differenceCommandVelocity** variable shows the difference, during velocity synchronous operation, between the set velocity calculated on the synchronous object and the set velocity that is executable on the axis taking into account the dynamic limits.

Canceling/deleting a synchronous operation command

The **_cancelFollowingObjectCommand** command can be used to cancel a synchronous operation command by specifying the **commandId** in the **commandToBeCancelled** parameter (as of V4.1). This removes the synchronous operation command from the command buffer and, if necessary, also from the IPO.

The synchronous motion is canceled with the local response **FOLLOWING_OBJECT_DISABLE**; see also **_cancelAxisCommand/_cancelExternalEncoderCommand** for the Axis/External Encoder technology object.

Additional references

You can find detailed information in both the **Motion Control Technology Objects Axis Electric/ Hydraulic, External Encoder** Function Manual and the **SIMOTION reference lists**.

Commands for resetting states and errors

The following commands are available for resetting states and errors:

- The **_resetFollowingObject** command resets the synchronous object, i.e. an active synchronous operation relationship is canceled and any pending errors cleared.
  The command can be executed when an error response (e.g. **DECODE_STOP**) is pending. All active commands and commands pending in the buffer are canceled. Changed default values can be reset using the **userDefaultData:=ACTIVATE_CONFIGURATION_DATA** parameter. Canceling commands by resetting the synchronous operation function does not generate any warnings.
  The **deleteSynchronizingCommandsOnly** parameter (V3.2 and higher) can be used to remove the waiting and active commands directly, without the need to reset the entire technology object.
- The **_resetFollowingObjectError** command acknowledges and resets errors on the synchronous object.
  The command can be executed when an error response (e.g. **DECODE_STOP**) is pending. One particular error or all errors can be ordered to be reset. The error response changes to the response with the highest priority of the still pending errors. It is terminated with a negative acknowledgment for any errors that cannot be acknowledged at this point. If all errors can be reset, the error response changes to **NONE**.
- The **_getFollowingObjectErrorNumberState** command is used to fetch the status of a specific error

Command processing

Command execution

Command advance

A criterion for the command advance is specified in the commands. If the condition for the command advance is satisfied, the next command in the user program is executed.
*****Specifying a command advance condition in a command influences the time at which the next programmed command will be executed.

Possible step enabling conditions on the synchronous object

The step enabling condition is indicated in the command parameter **nextCommand**.

Table 4-313 Step enabling conditions on synchronous object commands

Step enabling condition for next command	Time of advance
IMMEDIATELY	As soon as the command is issued
WHEN_BUFFER_READY	When the command enters the command buffer
AT_MOTION_START	When the command changes over to the interpolator
WHEN_ACCELERATION_DONE	When the acceleration phase is complete
AT_DECELERATION_START	When deceleration starts

Step enabling condition for next command	Time of advance
WHEN_INTERPOLATION_DONE	When setpoint interpolation for this command is complete
WHEN_AXIS_SYNCHRONIZED	When axis is synchronized with the master value
WHEN_MOTION_DONE	When motion generation is complete

Group classification of commands and command buffers

Each synchronous object has two dedicated command buffers for each command, which can be fetched for each IPO cycle. These command buffers are the buffer for synchronous operation commands and the buffer for *parallel effective commands*. The behavior of the buffers corresponds to the behavior of the axis.

Commands are classified in groups as follows:

- **Group 1**: *Synchronous operation commands*
  The following commands belong to this group: **_enableGearing**, **_disableGearing**, **_enableVelocityGearing**, **_disableVelocityGearing**, **_enableCamming**, and **_disableCamming**
  Depending on the transition behavior, new commands to be entered may behave in the following ways if the buffer is full:
  - Return with an error (**nextCommand** = **IMMEDIATELY** and **mergeMode** = **SEQUENTIAL**),
  - Wait for the buffer to become available (**nextCommand** ≠ **IMMEDIATELY** and **mergeMode** = **SEQUENTIAL**) or
  - Supersede the command pending in the buffer (**mergeMode** = **IMMEDIATELY**).
- **Group 2**: *Concurrently active commands*
  Commands that supersede each other in the command buffer. These commands are executed in the IPO. If there are more than one parallel effective command, only the last one is executed, because these commands overwrite each other. In this case, technology alarm 30002 "Command aborted (reason: ..., command type: ...)" is output.
  The following commands belong to this group: **_setGearingOffset**, **_setCammingScale**, **_setCammingOffset**, **_enableFollowingObjectSimulation**, **_disableFollowingObjectSimulation**, and **_setMaster**
- **Group 3**: Directly executed commands without entry in a command buffer
  Commands in this group do not abort each other.
  This group includes:
  _getSlaveValue, _getMasterValue,
  _resetFollowingObjectError,
  _getStateOfFollowingObjectCommand,
  _getMotionStateOfFollowingObjectCommand,
  _bufferFollowingObjectCommandID,
  _removeBufferFollowingObjectCommandId,
  _resetFollowingObject,
  _resetFollowingObjectConfigDataBuffer,
  _getFollowingObjectErrorNumberState

Command execution of sequential commands in the IPO cycle clock

If enabled to this effect, the commands are read out from the command buffer in every interpolation cycle. Within the slave value generation, up to two synchronous operation commands (_enableGearing(), _disableGearing(), _enableVelocityGearing(), _disableVelocityGearing(), _enableCamming(), _disableCamming()) are active at the same time and are executed in the IPO.

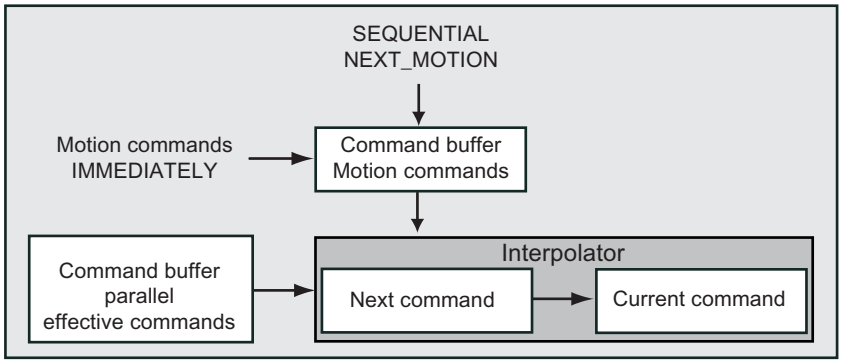


Figure 4-728 Command buffer and execution of sequential commands

Behavior regarding programming of a motion command:

- No motion command is active:
  An **_enable...** command is processed as the current command in the interpolator first as a waiting command, while a **_disable...** command is aborted immediately. If the synchronization criterion is satisfied, the waiting command changes its state to active (synchronizing/synchronous).
- If a motion command is waiting or active, the new motion command is processed as the next command in the interpolator first as a waiting command. A new **_disable...** command is immediately canceled if the current command is not a complementary **_enable...** command. If the next command is active, the current command is canceled. The next command is thereafter processed as the current command.
- If two motion commands are active or waiting and **mergeMode:=SEQUENTIAL** or **mergeMode:= NEXT_MOTION**, the new command is prevented from being read in until at least one command has been aborted or completed. With **mergeMode:= IMMEDIATELY**, the next command up to that point is aborted and replaced with the new command. Exception: If the new command is a complementary **_disable...** command for the next command, the new command is canceled immediately and only the current command is waiting/active.

See Command transition conditions (Page 2752) for further information.

Behavior regarding programming of a parallel effective command:

- **_setCammingScale()/_setCammingOffset()/_setGearingOffset()** act on the current (**_enable...**) command when **activationMode:= ACTUAL_VALUE** or **activationMode:= ACTUAL_AND_DEFAULT_VALUE**.
  If no command is active or the active command does not correspond to the correction being executed, the parallel effective command is canceled.
- In general, commands for simulation mode and resetting of current master value source are active.

Command transition conditions

The transition behavior is set in the **mergeMode** command parameter, on the commands of the synchronous object. The transition behavior affects the execution of the queued commands on the synchronous object. An active command can thus affect the execution of a command of another task.

The **mergeMode** on the synchronous object commands also determines the behavior of the following axis commands.

- A command issued with **mergeMode** = **IMMEDIATELY** clears the command buffer and overwrites the IPO (next command). The current command is executed. The next command is substituted.
- A command issued with **mergeMode** = **NEXT_MOTION** will be executed once the active command is complete and the pending commands have been deleted. It overwrites the command buffer.
- A command issued with **mergeMode** = **SEQUENTIAL** will be executed after the completion of the active command and the motion. The command will be entered in the command buffer when it is empty; if the command buffer is not empty, the command waits.

Influences between axis and synchronous object

Table 4-314 Transition behavior on synchronous object commands

Transition behavior for synchronous motion	Effect
IMMEDIATELY	The active command is aborted. The starting point for the slave value in the synchronous operation is the current axis setpoint. If the synchronous motion is active, the second synchronous function to be programmed or the non-active synchronous function is overwritten. If the synchronous function is already effective, then it remains effective. An active motion function in the following axis is overridden. It is therefore possible for motion, positioning, and synchronous operation to override each other another. If a synchronous operation command is issued with **mergeMode:=IMMEDIATELY**, the command waits in the command buffer until the synchronization criterion is fulfilled. Only then will the command enter the **active** state. A synchronous operation command will only be aborted by another motion command if the motion command is already active. In the same way, motion commands that are already active will only be aborted once a waiting synchronous operation command is activated.
SEQUENTIAL, NEXT_MOTION	If a main motion is active on the following axis, this main motion is first carried out completely. The starting point is the current axis setpoint for the main motion or, in the case of multiple synchronous commands, the internally generated slave value. If a synchronous motion is already active, the synchronous function is set to wait until the active synchronous function is completed or aborted. This requires the **nextCommand=WHEN_BUFFER_READY** setting. When **nextCommand=IMMEDIATELY** is set, the command is not executed and is returned with error information in the return value. The synchronous function is added to an active motion function in the following axis. A synchronous function that is in effect but not yet active can be deleted • With the **_disable...** command with **mergeMode=IMMEDIATELY** and **Synchronization criterion=IMMEDIATELY** or • With the **_resetFollowingObject** command

Table 4-315 Transition behavior on axis commands

Transition behavior for axis motion	Effect on the synchronous object
IMMEDIATELY	All active synchronous motions are aborted. None of the synchronous motions that are not active (i.e. they are neither in the synchronization motion nor in the "synchronous" status, but are instead waiting for the synchronization criterion) are aborted.
SEQUENTIAL, NEXT_MOTION	The synchronous operation commands/synchronous motions are not aborted and the axis motion only becomes active once the synchronous operation is complete.

The following special feature applies when motion commands are pending on the synchronous object and on the axis in the same interpolation cycle clock:

- If **mergeMode**(axis)=**SEQUENTIAL**, the synchronous command is executed
- If **mergeMode**(axis)=**IMMEDIATELY**, the axis command is executed

Error handling

Local alarm reaction on the synchronous object

Local **alarm responses** are specified by means of the system.

The following responses are possible:

- **NONE**
  No response
- **DECODE_STOP**
  Abort of the command preparation, the current synchronous operation function remains **active**.
- **FOLLOWING_OBJECT_DISABLE**
  Abort of the command preparation, abort of the current synchronous operation function.

An error can be reset with **_resetFollowingObject** or **_resetFollowingObjectError**

Note

The stop responses are listed in order of increasing priority. Global alarm responses can be set in the **alarm configuration** in the technology object; these can also be set to require **Power On**. For further information, see the **Motion Control Basic Functions** Function Manual, "Configuring technological alarms".

Local alarm reaction of the following axis and its mutual effects with the synchronous object

The synchronous object and the Axis TO have a reciprocal effect on each other depending on their respective operating modes and which commands are in effect. Thus, errors and alarms in the TO axis have a direct effect on the synchronous object functions. For example, if a technology alarm triggers a stop response in the following axis, the synchronous motion is also stopped. If an error is pending on the synchronous object only, the following axis can still position but can no longer perform synchronous operation.

The following responses by the Axis TO, which can be read in the **errorReaction** system variable, affect the synchronous object:

- **MOTION_STOP**
  Controlled motion stop with programmed ramp values.
- **MOTION_EMERGENCY_STOP**
  Controlled motion stop with maximum ramp values/limits for the axis.
- **MOTION_EMERGENCY_ABORT**
  Controlled motion stop with maximum axis dynamic values (starting from the current setpoint).

- **FEEDBACK_EMERGENCY_STOP**
  Motion stop with preassigned braking ramp.

  Note

  Problems with FEEDBACK_EMERGENCY_STOP on the leading axis

  Switchover to the actual value of the leading axis is performed. An existing setpoint-actual value different results in a step response of the speed or position on the synchronous object and therefore also on the synchronized axis.

  - Configure Motion_Emergency_Abort on the master axis

- **OPEN_POSITION_CONTROL**
  Motion stop with speed setpoint zero and abort.
- **RELEASE_DISABLE**
  Motion disable with controller disable and abort of all commands.

Synchronous operation is aborted for all responses indicated.

Note

Errors on the synchronous object do not have any effect on the enables/error response of the following axis.

Note

If the **disableSynchronousOperation** configuration data item is set to **YES** and simulation is active on the synchronous object, the synchronous operation function is *not* aborted on the stop response or when the **_disableAxis** command is executed.

The synchronous operation function is aborted with alarm "20005 Device type:..., log. address:... failed" if no valid actual value is present, e.g. if the axis is restarted.

Error handling in the user program

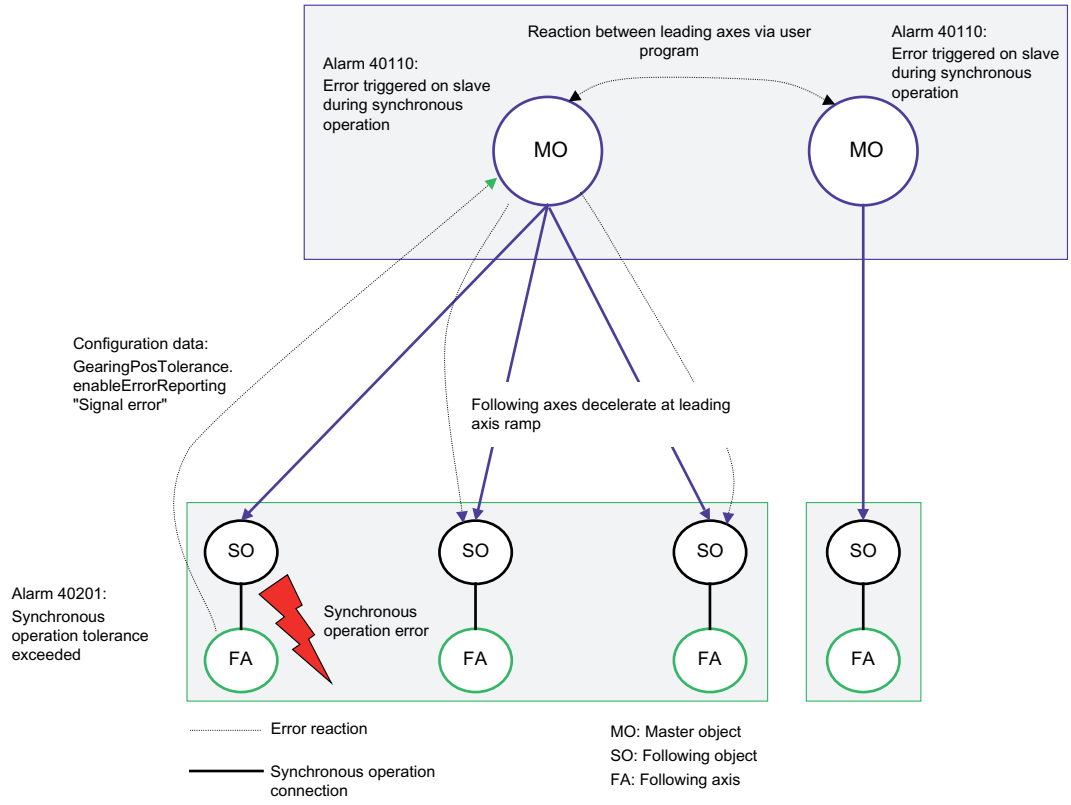


Figure 4-729 Error response in user program in distributed synchronous operation

The starting point is a synchronous operation error in a following axis (synchronous operation tolerance exceeded). Alarm 40201 "Synchronous operation tolerance of the gearing axis exceeded" is issued. The following axis changes to STOP mode.

The leading axis/external encoder responds with an error. The alarm 40110 "Error triggered on slave during synchronous operation (error number: ...) is signaled. The master object enters the STOP state.

- The local following axes also change to STOP mode.
- However, distributed synchronized axes continue to follow the master setpoint if measures are not taken in the user program to initiate an appropriate response to this error response.

Note

With V4.2 and higher, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error.

For additional information, see Synchronous operation monitoring (Page 2687).

Menus

Synchronous Operation - Menu

Grayed out functions cannot be selected.

Table 4-316 You can select the following functions:

Function	Meaning/Note
Close	Use this function to close the active window in the working area.
Properties	Properties displays the properties of the synchronous object selected in the project navigator. You can enter the object name plus author and version in this window.
Configuration	This function opens the configuration for the synchronous object selected in the project navigator. Assign the master values and cams to the following axis in this window.
Expert List	This function opens the expert list for the synchronous object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list; see Basic functions - Expert list.
Default setting	This function opens the default settings for the synchronous object selected in the project navigator. In this window, you define the replacement values for calling synchronous operation functions (**_enableGearing**, **_enableVelocityGearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGearing**, and **_disableCamming**).
Settings	This function opens the settings for the synchronous object selected in the project navigator. You can define some synchronization settings in this window.
Interconnections	This function opens the interconnections for the synchronous object selected in the project navigator. You can see the inputs of the axis in this window.
Expert	This function opens the submenu for the expert settings.
Configure units	This function opens the Configure units of the object window in the working area. You can configure the units used for the selected object here.

See also

Synchronous Operation Configuration (Page 2721)

Synchronous Operation - Context Menu

Grayed out functions cannot be selected.

Table 4-317 You can select the following functions:

Function	Meaning/Note
Open configuration	This function opens the configuration for the synchronous object selected in the project navigator. Assign the master values and cams to the following axis in this window.
Default setting	This function opens the default settings for the synchronous object selected in the project navigator. In this window, you define the replacement values for calling synchronous operation functions (**_enableGearing**, **_enableVelocityGearing**, and **_enableCamming** or **_disableGearing**, **_disableVelocityGearing**, and **_disableCamming**).
Settings	This function opens the settings for the synchronous object selected in the project navigator. You can define some synchronization settings in this window.
Interconnections	This function opens the interconnections for the synchronous object selected in the project navigator. You can see the inputs of the axis in this window.
Expert	This function opens the submenu for the expert settings.
Insert script folder	Insert script folder enables you to insert a folder for scripts.
Import object	Use **Import object** to open a window for the XML import. You can define the parameters for the XML import in this window.
Save project and export object	Use **Save project and export object** to open a window for the XML export. You can define the parameters for the XML export in this window.
Insert external master value	External master value (**ExternalMasterType**): Proxy object for an external master value.

See also

Synchronous Operation Configuration (Page 2721)

4.7.3 Part II - Distributed Synchronous Operation

4.7.3.1 Overview of distributed synchronous operation

This chapter describes the **Distributed Synchronous Operation** function (V3.0 and higher). It introduces you to the operating principle and provides information about the technological supplementary conditions as well as the operating characteristics of distributed synchronous operation. You are shown how to create and configure a distributed synchronous operation.

Function overview

The **Distributed synchronous operation** functionality allows you to create a master value source and a synchronized axis on different controls. In a project, it is possible to form function groups and thus a machine structured on a modular basis. Synchronized axes no longer need to be present in a single control, but can instead be distributed among several controls.

Isochronous (clock-synchronized) bus coupling

The coupling between the master axis (or the external encoder) and the following axis is performed using an isochronous bus coupling between the controls, via PROFIBUS DP or PROFINET IO with IRT.

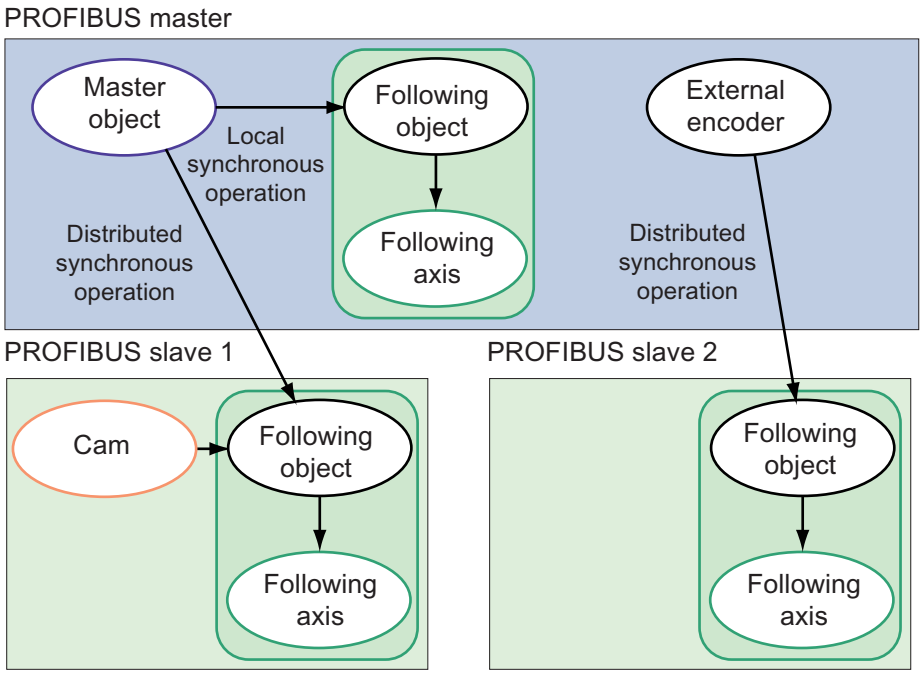


Figure 4-730 Distributed synchronous operation using PROFIBUS DP as an example

Synchronizing the bus interfaces

The DP/PN interfaces must be synchronized with each other for distributed applications using the isochronous PROFIBUS or PROFINET IO with IRT.

Information on this subject is available in the **Motion Control Basic Functions for Modular Machines** Function Manual and the **Communication** Configuration Manual.

Application

You can use distributed synchronous operation to create function groups in your project and set up a machine on a modular basis. Instead of having to use the same control system to control synchronously operating axes, you can now distribute the axes across a number of modules.

Operating principle/Compensation

With distributed synchronous operation, the interpolator cycle clocks of the master object and the following axis may be offset. As a result of the communication required, there is also an offset in the calculation of related signals (master value source and remote following axis).

The cycle clock offset can be compensated using the following measures:

- Compensation on the master value side by means of setpoint output delay
- Compensation on the slave value side by means of master value extrapolation

See Compensations for distributed synchronous operation (Page 2767).

See also

Fundamentals of Distributed Synchronous Operation (Page 2760)

Distributed Synchronous Operation Configuration (Page 2777)

Configuring distributed synchronous operation across projects (Page 2786)

4.7.3.2 Fundamentals of Distributed Synchronous Operation

Boundary Conditions

Rules for the communication/topology for distributed operation using PROFIBUS

The following rules apply for the PROFIBUS topology with distributed synchronous operation:

- Distributed synchronous operation is only possible via equidistant master/slave communication.
- The leading axis or external encoder must be located in the PROFIBUS master, and the distributed following axis must be located in the PROFIBUS slave.
  Further local synchronizations with the master control are possible.
- Distributed synchronous operation can only be created on one PROFIBUS level.
  Consequently, cascaded distributed synchronous operation is not possible.
- Different IPO cycle clocks and position control cycle clocks can be used in the SIMOTION devices involved.
- The same DP cycle clock must be used in the SIMOTION devices taking part in the distributed synchronous operation.

Exception: See **Cycle clock scaling for SIMOTION D** in this chapter

Data transmission for distributed synchronous operation using PROFIBUS

A total of 24 bytes are transmitted and received via the PROFIBUS interface for each synchronous operation connection and cycle clock (bi-directional connection for synchronous operation data). Only a certain amount of data can be transmitted in each DP cycle for each master-slave connection (a maximum of 244 bytes can be sent and received). This also enables a maximum of 10 connections with 24 bytes each. In addition, the amount of data in the PROFIBUS master is limited to 1 Kbyte for inputs and 1 Kbyte for outputs per PROFIBUS interface, irrespective of the number of devices connected; in other words, 40 connections are theoretically possible.

In addition to the transmitted data, the following must be noted:

- Drive connection
- I/O connection
- Application Data

This limits the number of possible connections with distributed synchronous operation.

The system can be optimized. Instead of several distributed connections, for example, a virtual axis on the slave can first be coupled to a (real or virtual) master axis which then serves several following axes.

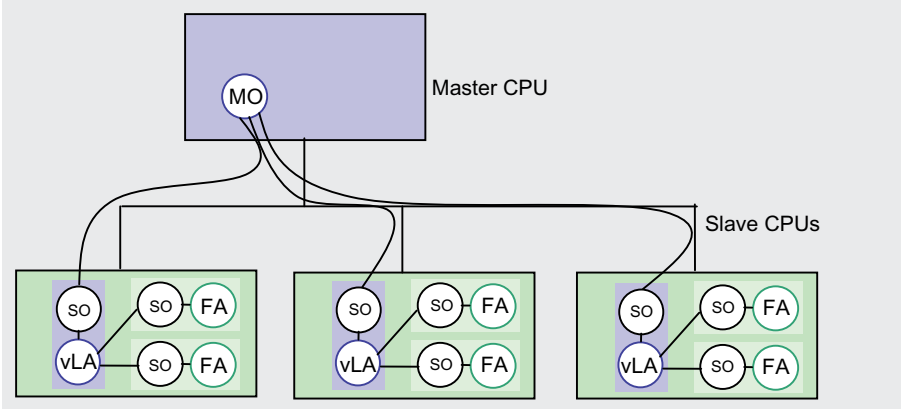


Figure 4-731 Example of optimizing connections using virtual master axes on slave devices

The virtual master axes on each slave allow axis groupings of the individual machine modules to also be operated "independently" (e.g. for the commissioning of individual modules).

Master-slave relationship

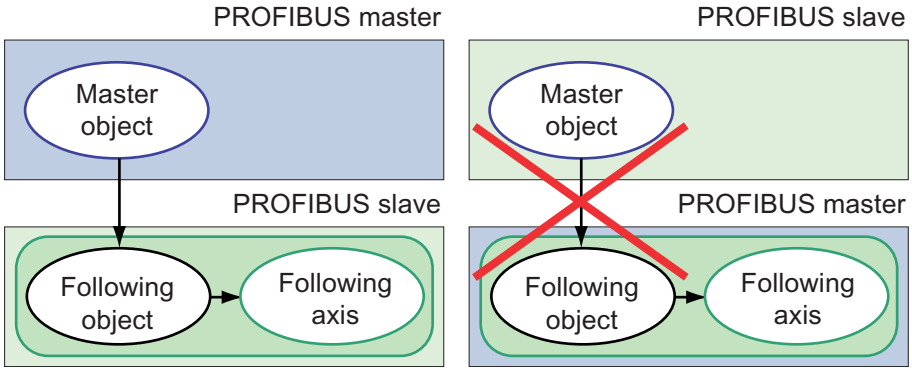


Figure 4-732 Master-slave relationship in distributed synchronous operation

Connection between axis and synchronous object

The synchronous object and, if appropriate, the cam must be located on the slave controller, together with the slave axis. The master value source (axis or external encoder) is always located on the master control.

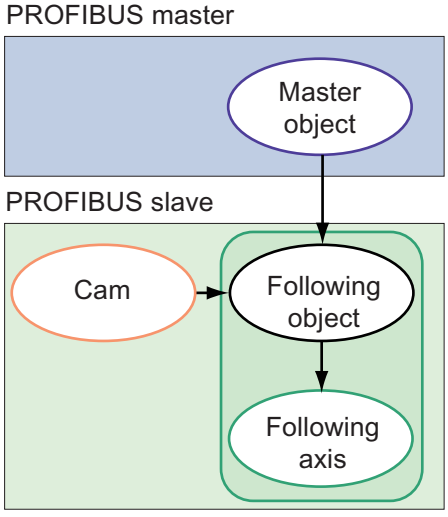


Figure 4-733 Following axis and synchronous object on the same control

Cascading

A distributed synchronous operation can be connected to following local synchronous operation on the slave controller.

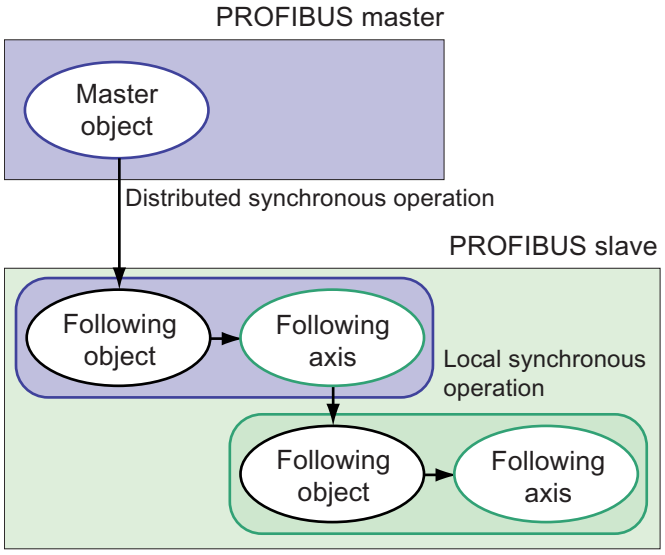


Figure 4-734 Cascading: distributed synchronous operation to following local synchronous operation

However, it is not possible to cascade two distributed synchronous operations one after the other, i.e. the following axis in distributed synchronous operation 1 cannot be used as the master axis in distributed synchronous operation 2. This is also true if the second PROFIBUS interface configured as a master is used.

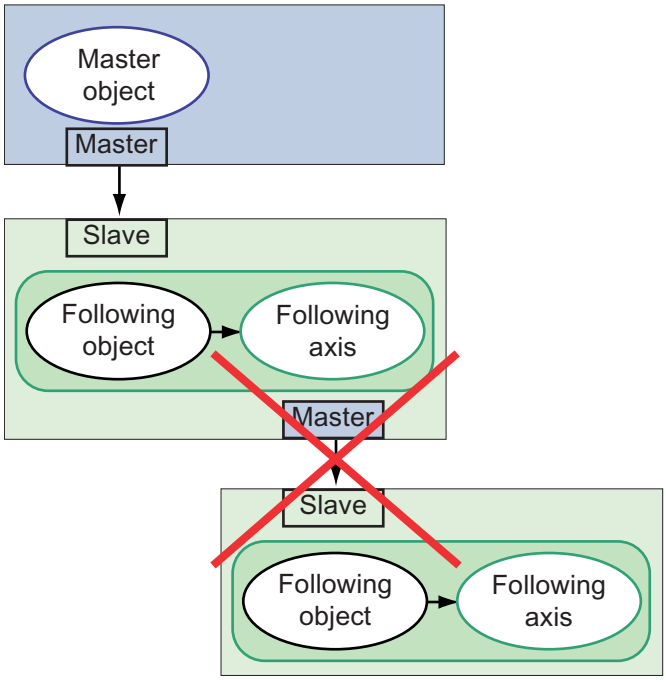


Figure 4-735 Distribution via one PROFIBUS level only

No feedback

It is not permitted to configure distributed synchronous operation from Device1 to Device2 and back again. This is true even if two appropriately configured PROFIBUS interfaces are used.

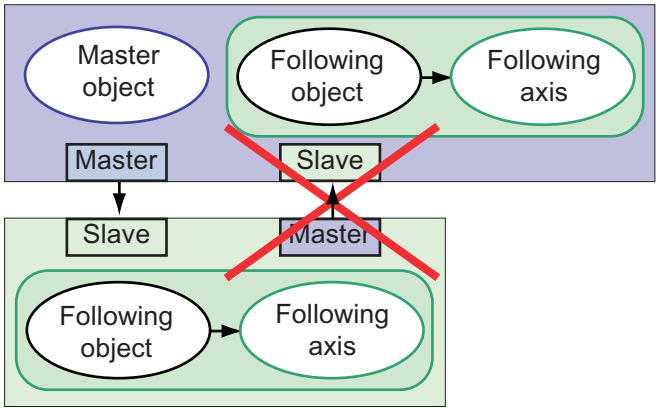


Figure 4-736 No feedback in distributed synchronous operation

Example hierarchy with synchronized isochronous PROFIBUS interfaces

The following is an overview of requirements for distributed synchronized operation:

All PROFIBUS connections:

- Must have the same DP cycle clock settings
  Exception: See **Cycle clock scaling for SIMOTION D** in this chapter
- Must have isochronous cycle clock settings
- Must have master and slave synchronization
  if a master and slave are used on the same device

- Distributed synchronous operation is only possible over one shared bus segment.
- Possible number of slaves: See **Data transmission for distributed synchronous operation using PROFIBUS** in this chapter

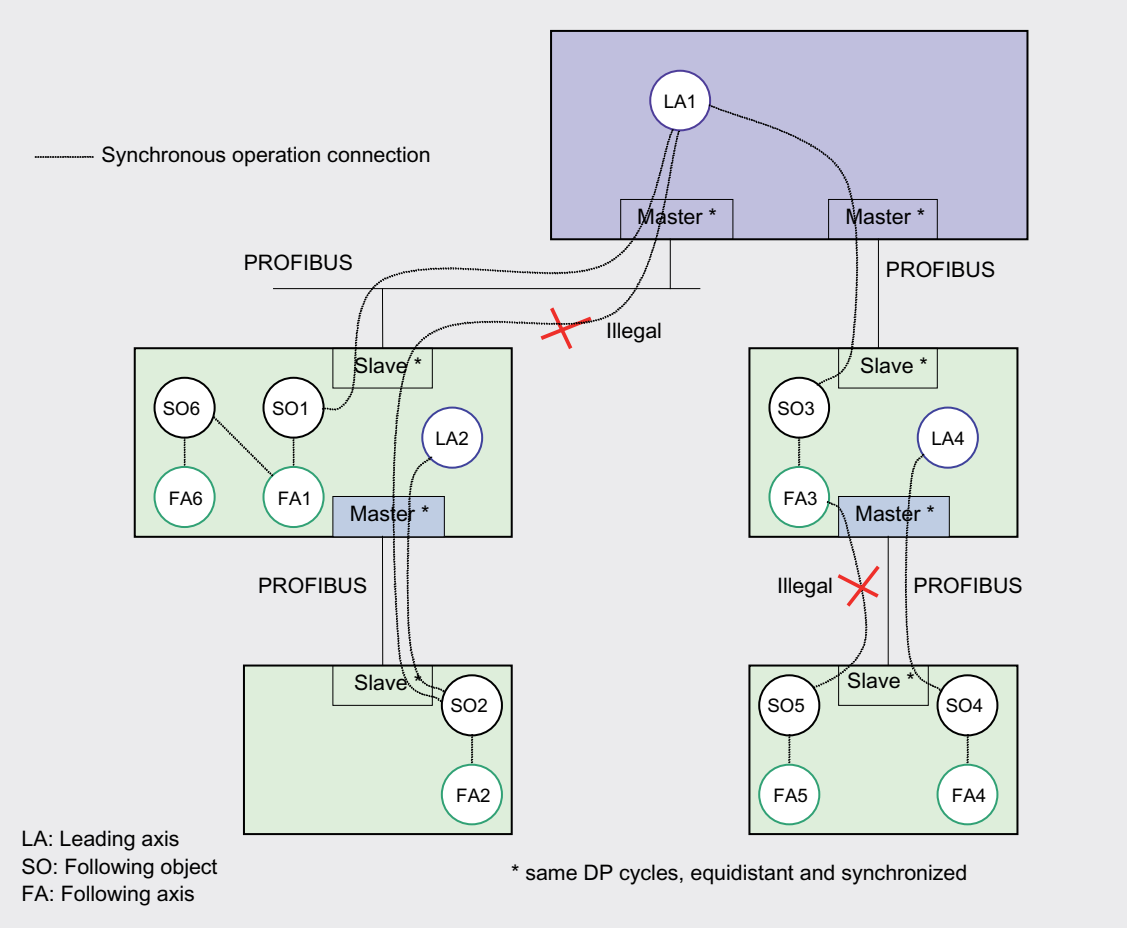


Figure 4-737 PROFIBUS topology: Hierarchy with synchronized isochronous PROFIBUS interfaces

Cycle clock scaling for SIMOTION D

With SIMOTION SCOUT V3.2 SP1 and higher, a distributed synchronous operation with cycle clock scaling between the two external DP interfaces (DP1/DP2) and the internal DP interface for SIMOTION D is possible.

If the master value changes only very slowly or the external DP interface requires a faster cycle time than the internal DP interface, a decoupling of the fast internal DP cycle from the slower external DP cycle is desirable.

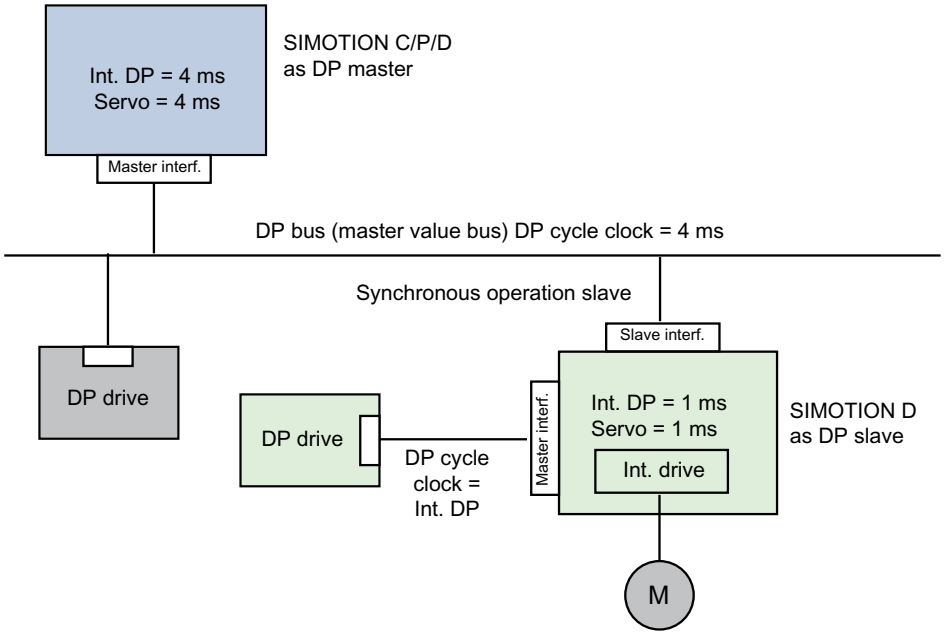


Figure 4-738 Cycle clock scaling for SIMOTION D

This is possible under the following boundary conditions:

- An external DP interface of SIMOTION D is used as an isochronous slave interface. Only in this case can an integer cycle clock scaling of isochronous external DP slave interface to internal interface be specified.
- For SERVO, IPO, and IPO_2, settings can be made for all permissible cycle clocks. The master axis and following axis can run on different IPO levels.
  The IPO cycle clock of the synchronous object, however, must be set equal to the cycle clock of the isochronous external DP slave interface (otherwise the error message "50205 - Offset cannot be determined" will be issued).
- In addition, the second external DP interface can be operated as isochronous master (the first is isochronous slave), for example, to operate external drives. In this case, the cycle clock must be the same as the cycle clock of the internal DP cycle.
- The second external DP interface can also be operated as a "non-isochronous, free-running interface". In this case, there is no effect on the cycle clock settings.
- The reduced cycle clocks of the external DP interfaces must be set in HW Config.
- For D4xx, DP to Servo reduction ratio is possible
- For D4xx-2, a DP to Servo reduction ratio is possible from V4.3 and higher
- A DP to Servo reduction ratio is generally only permissible if no PROFINET IO with IRT has been configured.

Rules for the communication/topology for the distribution using PROFINET IO with IRT (V4.0 or later)

Distributed synchronous operation between SIMOTION devices via PROFINET IO with IRT uses the controller-controller data exchange broadcast for PROFINET IO to exchange the synchronous operation data.

Differences to PROFIBUS

Regarding distributed synchronous operation with PROFIBUS, see **Rules for the communication/topology for distributed operation using PROFIBUS** in this chapter, the following differences exist:

- Master object and following axis / synchronous object can be located on any controllers. (PROFINET IO with IRT does not have any communications master and communications slave as for PROFIBUS.)
- Cascading of distributed synchronous operations is possible over more than one level.
- Recursive interconnection with PROFINET is possible (see Note)
  Note:
  slave value compensation via master value extrapolation is possible.
  Master value compensation by delaying setpoint output is not possible.

Note

For the following Control Units, a second Servo task is optionally available:
- SIMOTION D435-2 DP/PN (V4.3 and higher)
- SIMOTION D445-2 DP/PN (V4.2 and higher)
- SIMOTION D455-2 DP/PN (V4.2 and higher)
For further information, see *D4x5-2 Commissioning and Hardware Installation Manual and Basic Functions Function Manual*.

Compensations for distributed synchronous operation

In a distributed synchronous coupling, calculation of related signals between the master value source and the remote following axis is offset due to the distribution and the associated communication requirements. Compensation of this offset is supported by the system.

The following compensations are available in the system:

- Compensation on the master value side by means of setpoint output delay on the component that provides the master value for the distributed synchronous operation
- Compensation on the slave value side by means of master value extrapolation on the component containing the remote slave objects

Note

For distributed synchronous operation with extrapolation on the following axis, the **setpoint monitoring with jerk** setting is not appropriate.

The compensations are set and displayed via the system variable **distributedMotion**.

- The output delay is displayed on the leading axis.
- The master value delay is displayed on the synchronous object.
- The cycle clock offset is displayed on the synchronous object.

Sign-of-life monitoring is required for the compensation using master value extrapolation.

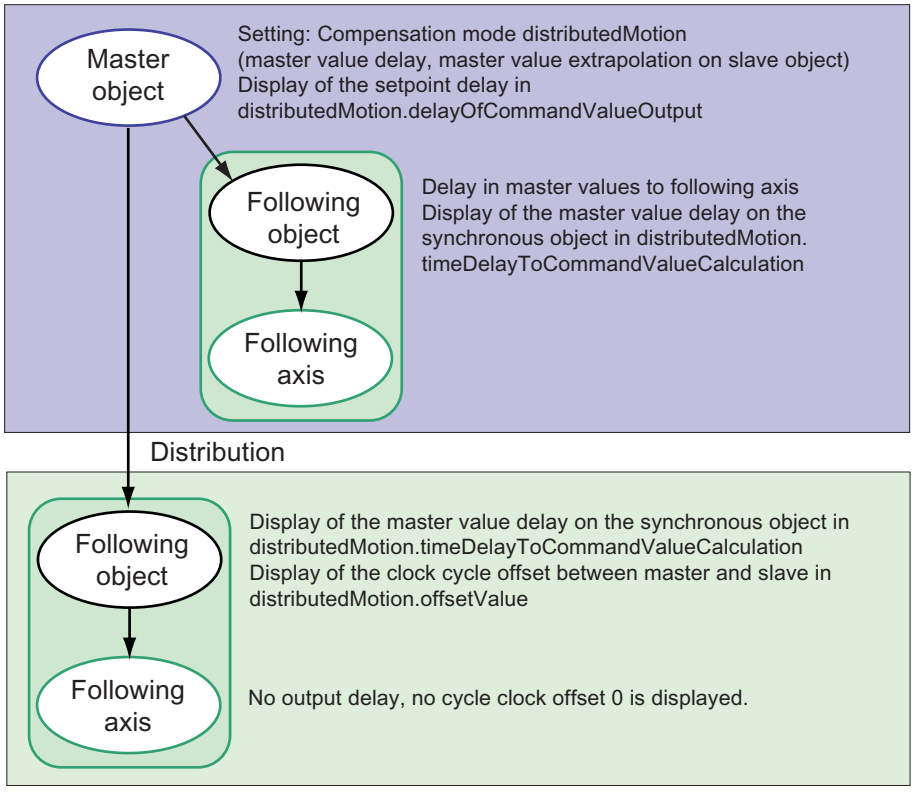


Figure 4-739 Overview: Compensations for distributed synchronous operation

Applications and results

- It is useful to activate a setpoint output delay on the master value side if, for example, synchronism of distributed synchronous operation is of primary importance and a rapid response on the master value side to local events is of lesser importance.
- It is useful to activate compensation on the slave value side by means of master value extrapolation without a setpoint output delay on the master value side if, on the master side, the master values and slave values need to be output without a delay due to a short response time, for example, and if, on the slave side, synchronism or a secondary error resulting from the larger extrapolation range is permissible.

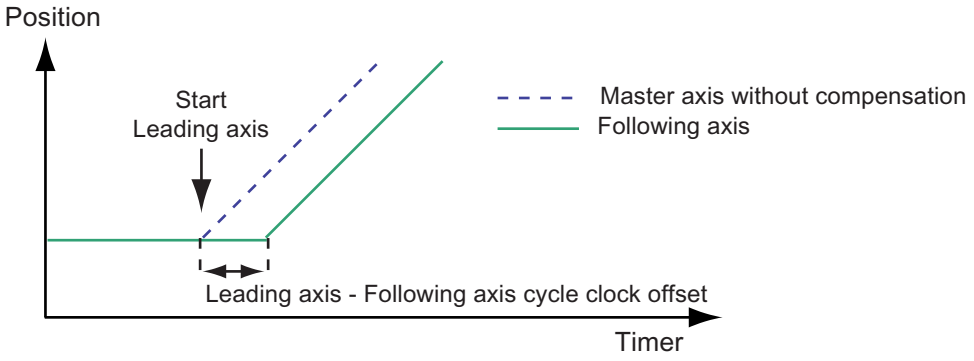


Figure 4-740 Distributed synchronous operation without setpoint output delay on the leading axis side and without compensation on the following axis side

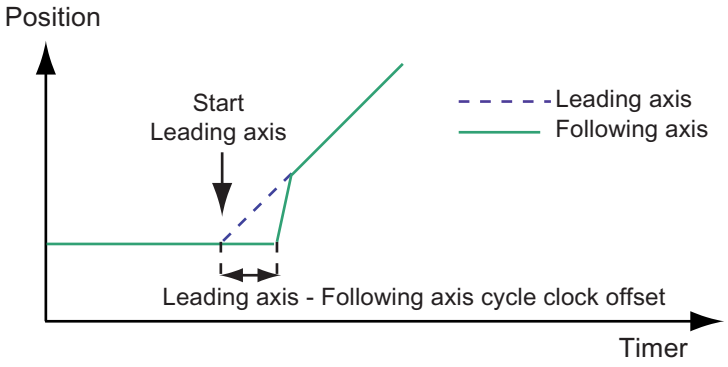


Figure 4-741 Master value extrapolation on the slave value side without setpoint output delay on the master value side

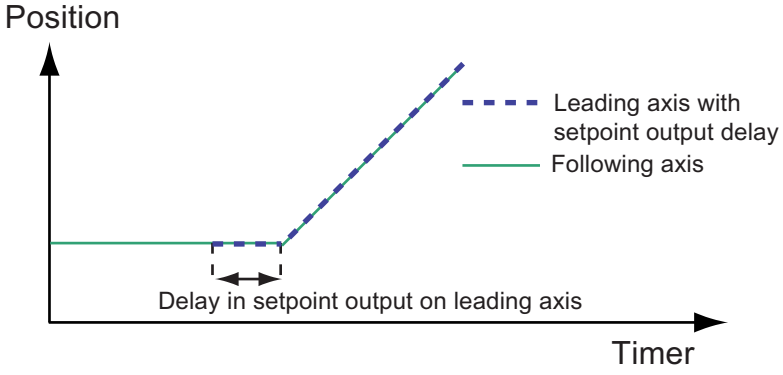


Figure 4-742 Setpoint output delay on the leading axis side

Activating

- When output delay on the master value side is activated, the signal output on the master value side is delayed by the calculated IPO clock cycles and any resulting IPO phase offset is compensated by means of interpolation on the slave value side.
- When master value extrapolation on the slave value side is activated without output delay on the master value side, the total cycle clock offset between the master value calculation and the slave value calculation is compensated for by means of the master value extrapolation on the slave value side.

Scope

- The setpoint output delay on the master value side is applicable to the following:
  - Axis setpoints calculated directly on the leading axis that are delayed on the axis prior to being passed on to the servo
  - Axis setpoints calculated by local synchronous objects; the synchronous object forwards the calculated setpoints to the axis
- Compensation on the slave value side by means of interpolation/extrapolation takes place on the master value of the remotely interconnected synchronous object.

See also

Compensation on master value side by means of setpoint output delay (Page 2770)

Compensation of the slave value side by means of master value extrapolation (Page 2772)

Permissible combinations for cycle clock offset compensation in distributed synchronous operation (Page 2773)

Cycle clock offset calculation using a command (Page 2774)

Compensation on master value side by means of setpoint output delay

Compensation on the master value side by means of setpoint output delay of the distributed synchronous operation will be activated for the master object.

Compensation on the master value side results in the following:

- The output of setpoints calculated for the axis, delayed by "n" IPO cycle clocks, to the servo/ position controller of the axis
- The output of setpoints from a local synchronous object interconnected with the master axis or external encoder, delayed by "n" IPO cycle clocks, to the interconnected synchronized axis

The number of IPO cycle clocks is calculated from the maximum cycle clock offset across all distributed synchronous relationships with the master value source. The number of IPO cycle clocks (integer), which contains the total delay, is calculated.

When compensation on the master value side by means of setpoint output delay is deactivated, setpoints are output to the axis and the master value is forwarded to and evaluated on local synchronous objects without delay.

Activating compensation on the master value side by means of setpoint output delay

Master value compensation by delaying setpoint output is activated using the following configuration data on the master axis and/or an external encoder:

- **(TypeOfAxis.)distributedMotion.enableOffsetCompensation:=YES**: Calculation of offset is activated
- **(TypeOfAxis.)distributedMotion.enableDelayOfCommandValueOutput:= YES**: Compensation on master value side is activated Delay to setpoint output on master value side, simultaneous start and no overshoots. The local following axis also has a delayed start. Disadvantage of "YES": There is less likely to be a response to an event on the master. When activated, a setpoint output delay on the axis is always effective.
- **enableDelayOfCommandValueOutput:= NO** is used to deactivate compensation on the master side. The master immediately outputs the setpoint value.

When activated, a setpoint output delay on the axis is always effective.

Cycle clock offset calculation

The system automatically calculates the maximum cycle clock offset after a transition from **STOP**/ **STOPU** to **RUN**. Cycle clock offset calculation also runs after one of the axes or an external encoder involved has been restarted and after disconnection/reconnection. The status of the cycle clock offset calculation is indicated in the **distributedMotion.stateOfOffsetCalculation** system variable on the master axis and on the remote distributed synchronous object. The cycle clock offset is not yet determined for status **INVALID**, the cycle clock offset cannot be determined. The master axis/external encoder issues the technological alarm "40304 Offset cannot be determined".

Compensation on the master value or slave value side requires that offset calculation be activated in configuration data element **(TypeOfAxis.)distributedMotion.enableOffsetCompensation**, on the master axis or external encoder.

setpoint output delay

The absolute value of the setpoint output delay can be read out by means of the **distributedMotion.delayOfCommandValueOutput** system variable on the master axis.

The time is indicated in the **distributedMotion.timeDelayToCommandValueCalculation** system variable on the synchronous object of the remote following axis.

The status of the setpoint output delay is indicated in the **distributedMotion.stateOfDelayValue** system variable on the master axis or external encoder and on the remote synchronous object.

- If the status is **INVALID**, the setpoint output delay is not activated.
- If the status is **VALID**, the setpoint output delay is active.

The maximum permissible delay for setpoint output is 10 interpolation cycle clocks. If the delay exceeds this, two alarms are output on the master value axis or external encoder: "40124 Offset cannot be compensated" and "40125 Setpoint output delay on the master side is deactivated."

If a local interconnected synchronized axis is acting as the master value for a distributed synchronous operation, the same setting in terms of the effective compensations on the master value side should be made on the first master value and on the local synchronized axis.

Compensation of the slave value side by means of master value extrapolation

When compensation on the master value side by means of setpoint output delay is not activated, the compensation on the slave value side performs a linear extrapolation using the two most recent master values received in order to compensate for the cycle clock offset.

In the event that master values are lost, the two most recent master values received are used for the extrapolation.

Activating compensation on the slave value side using master setpoint extrapolation

Slave value compensation (interpolation/extrapolation) is activated using the following configuration data on the master axis and/or an external encoder:

- **(TypeOfAxis.)distributedMotion.enableOffsetCompensation:=YES**: Calculation of offset is activated.
- **(TypeOfAxis.)distributedMotion.enableDelayOfCommandValueOutput:= NO**: Compensation on master value side is deactivated.

Display of setpoint output time delay on the master value side

The setpoint output delay time on the master value side is indicated in the **distributedMotion.timeDelayToCommandValueCalculation** system variable on the synchronous object. This delay time is generally greater than the offset calculated across all remote synchronous relationships.

The **distributedMotion.timeDelayToCommandValueCalculation** system variable on the remote synchronous object corresponds to the **distributedMotion.delayOfCommandValueOutput** system variable of the associated master value object.

Status of calculation of compensation on the slave value side

The status of the compensation calculation on the slave value side is indicated by means of the **distributedMotion.stateOfOffsetCalculation** system variable on the synchronous object as well as on the master value source/master axis.

Cycle clock offset calculation

The system automatically calculates the maximum cycle clock offset after a transition from **STOP**/ **STOPU** to **RUN**. Cycle clock offset calculation also runs after one of the axes or an external encoder involved has been restarted and after disconnection/reconnection. The status of the cycle clock offset calculation is indicated by means of the **distributedMotion.stateOfDelayValue** system variable on the synchronous object as well as on the master value source.

If a command is transmitted to the synchronous object before completion of the cycle clock offset calculation, a technological alarm is output ("50204 Offset calculation is active"). If the cycle clock offset cannot be calculated, the synchronous object outputs a technology alarm ("50205 Offset cannot be calculated").

Cycle clock offset between master value calculation and slave value calculation

The clock cycle offset between the master value calculation and slave value calculation is indicated on the synchronous object by means of the **distributedMotion.offsetValue** system variable. The cycle clock offset displayed does not depend on the output delay on the master value side.

Comment:

- All system variables on the master value source indicate the status or the respective value across all interconnected following axes.
- All system variables on the synchronous object indicate the status or the respective value for the interconnection with the current master value source.

Permissible combinations for cycle clock offset compensation in distributed synchronous operation

The compensation settings are made on the master value side (on the master control) using configuration data elements **(TypeOfAxis.)-distributedMotion.enableDelayOfCommandValueOutput** and **(TypeOfAxis.)distributedMotion.enableOffsetCompensation**.

The following combinations are possible:

Table 4-318 Permitted settings of compensation on the master value side and slave value side (setting on the leading axis or external encoder)

enableOffset Compensation	enableDelay OfCommand ValueOutput	
NO	**NO**	No compensation activated
NO	**YES**	Not permitted. An output delay on the master value side without interpolation on the remote following axis is not permitted.
YES	**NO**	Linear extrapolation in the following axis on the last two master setpoints using the cycle clock offset
YES	**YES**	Leading setpoint output delay in the local following axis, linear interpolation in the following axis of the master setpoints transmitted using the cycle clock and phase offset

Cycle clock offset calculation using a command

The cycle clock offset calculation can be initiated explicitly (V4.1 and higher), for example, when adding an axis for modular machine concepts.

The cycle clock offset can be calculated on the leading axis with the **_enableDistributedMotionDelayValueCalculation** command. This command acts on the master value and all cascades below it for which this master value applies in its cascade. Active synchronous operation commands are aborted when the offset calculation is started. If there are multiple master values at the top level, the command must be called on each object.

See also

Rules for the communication/topology for the distribution using PROFINET IO with IRT (V4.0 or later) (Page 2766)

Operating axes with distributed synchronous operation

Sign-of-life monitoring

The master axis/external encoder and the remote synchronous object exchange signs of life to provide confirmation that the application is running correctly on the other side. For example, a distributed synchronous operation connection can be adversely affected by a fault on the bus (such as, message frame repetition).

Life-sign monitoring is implemented in the form of one life-sign counter in the master axis/external encoder and the synchronous object for each distributed synchronous operation. The process could be described as a "clock comparison". The sign of life is sent from the synchronous object to the master axis/external encoder and vice versa (bidirectional sign-of-life monitoring).

The life-sign counters are incremented on the source side in each IPO cycle clock, and the current value is transmitted to the relevant partner, where it is compared to an expected value. If the life-sign counter values differ from the expected values, an error will be output. Life-sign monitoring is only active if both SIMOTION devices are in **RUN** mode.

Note

A life-sign failure is not the same as an interrupted connection. The life-sign failure occurs if the life-sign is not received by the communication partner. This is the case with IPO overflow, for example. The connection is interrupted when the two communication partners are physically separated from one another. Further information can be found under "Connection interruption" in this chapter.

Failure limit

A parameter can be assigned to permit the life-sign to fail "n" times before an error response is output. This is set using the **(TypeOfAxis.)DistributedMotion.numberOfLifeSignFailures** configuration data (default: **1**).

The failure limit can be set on the leading axis or external encoder and on the following axes.

Extrapolation in the event of a failure

If a sign of life (and therefore also the master value) is absent and no error response is triggered (n > 0), the last available master setpoint is extrapolated.

Error reaction

An error response is triggered if the number of failures exceeds the parameterized value "n". The life-sign error is indicated on the following axis and the leading axis/external encoder. Consequently, the error reaction is applied to both of the following:

- **Leading axis/external encoder**: **40301** Loss of slave connection to the distributed control in the distributed synchronous operation
- **Following axis**: **50201** Loss of connection on the assigned control to the master in the distributed synchronous operation

Activating/deactivating life-sign monitoring

Life-sign monitoring can be enabled/disabled using the **(TypeOfAxis.)distributedMotion.enableLifeSignMonitoring** configuration data element on the master axis/external encoder and the synchronous object (default: **YES**).

Status signal

Life-sign monitoring must be activated on all participating objects or deactivated on all participating objects. If this is not the case, a technological alarm is issued on the master axis/external encoder or synchronous object to indicate that life-sign monitoring has been deactivated:

- **Leading axis/external encoder**: **40302** Life-sign monitoring for slave deactivated in distributed synchronous operation
- **Following axis**: **50202** Life-sign monitoring for master deactivated in distributed synchronous operation

The **distributedMotion.lifeSignError** system variable on both the leading axis/external encoder and synchronous object can be used to check whether life-signs have failed. When life-sign monitoring is activated, the assigned tolerated life-sign monitoring failures in **(TypeOfAxis.)DistributedMotion.numberOfLifeSignFailures** must be identical.

Connection interruption

If there is no axis error as subsequent error of synchronous operation monitoring, the following steps must be complied with in the event of a connection interruption:

- The connection must be physically restored.
- Wait with the program execution until the offsets have been exchanged and the setpoint output delay has been calculated (see System variables **stateOfOffsetCalculation**, **stateOfDelayValue**).
- The synchronous operation must be enabled again.

Operating states

Please note that in order for distributed synchronous operation to function, the master CPU *and* the slave CPU must both be in **RUN** mode. The connection for distributed synchronous operation does not occur automatically unless both are in **RUN** mode.

Association with life-sign monitoring

If life-sign monitoring is *deactivated* (**(TypeOfAxis.)distributedMotion.enableLifeSignMonitoring** = **No**) and a transition occurs from **RUN** to **STOP**/**STOPU**, the master CPU outputs a technological alarm (life-sign error **50201**) to all connected slave CPUs.

Synchronization status

The following device-related system variables on the slave indicate the status of synchronization:

- **StateOfDpinterfaceSynchonization**: only relevant if DP of PROFIBUS is equidistant
- **StateOfDpSlaveSynchonization**: only relevant if DP of PROFIBUS is not equidistant

Direction reversal of the master value for the synchronization

A maximum master value reversal can be defined in the configuration data **syncingMotion.masterReversionTolerance** on the synchronous object (V4.0 and higher). Specifying a tolerance is particularly useful for a distributed synchronous operation, where master value noise for actual value coupling can cause a master value reversal due to extrapolation.

See Actual value coupling with tolerance window (Page 2651).

Error handling

During distributed synchronous operation, the following axis sends the "Synchronous operation tolerance exceeded" error to the master axis. With V4.2 and higher, an error message is also output on the master object when the following axis disconnects the synchronous coupling for any reason in the event of an error. The error transfer must be configured: By means of the synchronous operation monitoring wizard on the following axis of the slave or by means of the **TypeofAxis.GearingPosTolerance.enableErrorReporting** configuration data.

Other errors (e.g. if the following axis is unable to synchronize) are not communicated to the master object. It is recommended that the 4xxxx errors be sent by the application to the master via the bus.

In HW Config of the I-slave, for example, a byte can be added to the configuration created by the system. This means that the following axis can report every error to the master axis by means of an application.

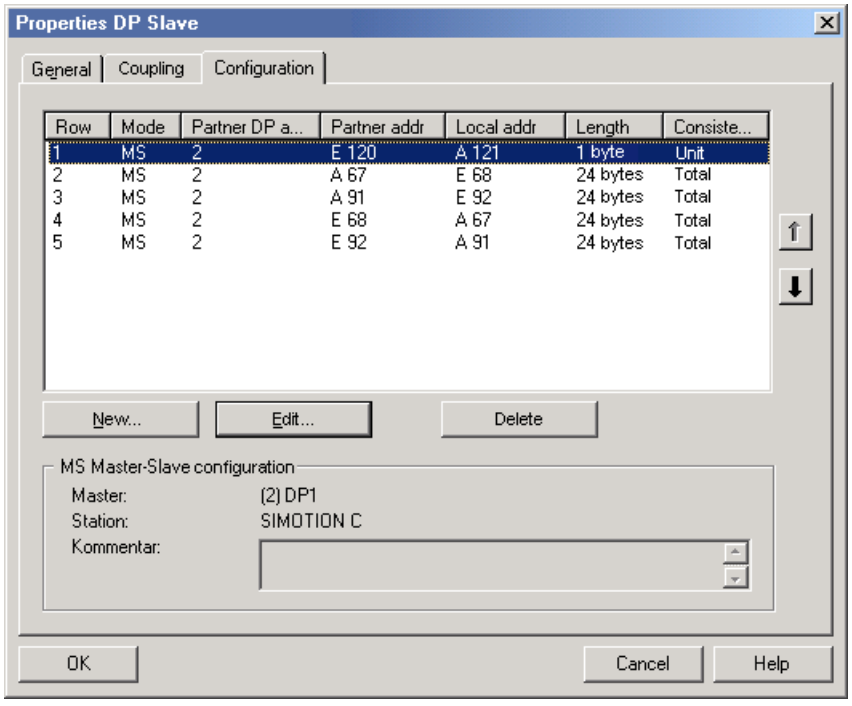


Figure 4-743 Configuration of an error byte for signaling the leading axis

See Error handling (Page 2754).

4.7.3.3 Distributed Synchronous Operation Configuration

This section describes how to create and configure SIMOTION devices and objects with distributed synchronous operation, and how to download them to the target system.

It is assumed that you have already created cams, master axes, or external encoders. A sample project for distributed synchronous operation can be found in the FAQ on the **SIMOTION Utilities & Applications DVD**.

To configure a distributed synchronous operation, you must carry out the following tasks:

- Creating SIMOTION devices with SCOUT (Page 2778)
- Creating connections with HW Config (Page 2778)
- Creating synchronous operation connections with SCOUT (Page 2780)
- Synchronizing the interfaces (Page 2784)
- Generating a synchronous operation configuration (Page 2783)
- Identify Possible error (Page 2783)

Creating SIMOTION devices with SCOUT

Creating a slave and master

In SIMOTION SCOUT, create two SIMOTION devices in your project:

- A SIMOTION device configured as a slave.
- A SIMOTION device configured as a master.
- Both SIMOTION devices must be connected and configured so that the HW Config can be compiled without errors.

Note

The SIMOTION devices must always be created with the same kernel version for the master and all slaves.

For exceptions, see information at Overview of distributed synchronous operation with different RT versions (Page 2822).

Note

Addresses entered by SIMOTION SCOUT for axis communication of distributed synchronous operations in HW Config may not be changed.

Creating connections with HW Config

Requirement

You have created a SIMOTION project containing two SIMOTION devices.

Creating connections using PROFIBUS

1. Set the "Operating mode" property for SIMOTION devices in HW Config. Open the "Properties" window at the relevant interface, e.g. DP/MPI. On the "Operating mode" tab, set one SIMOTION device to **DP Slave** and the second SIMOTION device to **DP Master**.
2. Set PROFIBUS addresses for both SIMOTION devices.
3. Set the DP mode to **DPV1** for both SIMOTION devices.
4. Insert a subnet master system on the PROFIBUS master.
5. Under **Properties**, set the transmission rate.
   For PROFIBUS communication, a transmission rate of **12 Mbit/s** is recommended.
6. Under **Options** in **Properties**, select **Equidistant bus cycle** and set the cycle time.
7. Check that the check box for **Programming, status/control or other PG functions and non-configured communication connection possible** under **Operating mode** in **PROFIBUS properties** for the slave has *not* been selected. Clear it if necessary.
   If the check box is selected, isochronous operation is *not* possible.
8. From the hardware catalog of **PROFIBUS DP**, **already configured stations**, add the **C23x**/**P3xx**/**D4xx ISlave** to the existing master system.

9. The DP slave which has already been configured is offered under Coupling. Use **Connect** to connect the slave to the master.

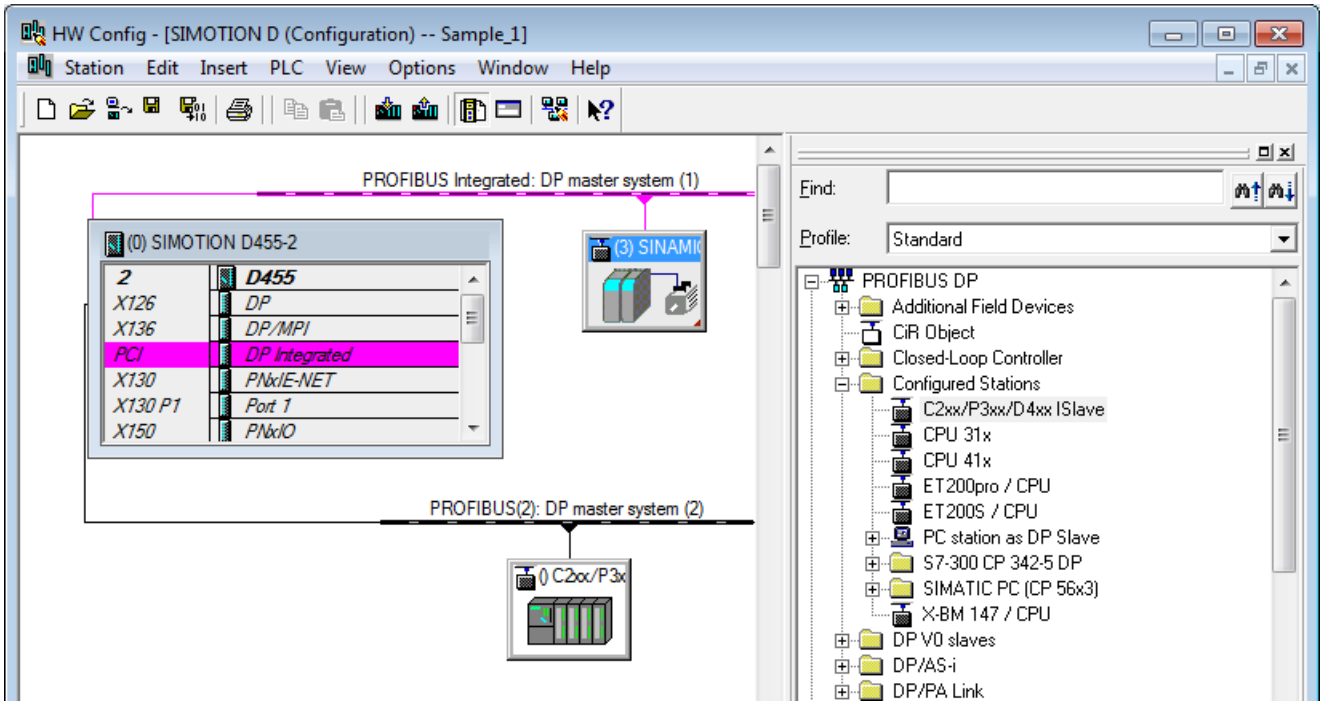


Figure 4-744 Device as PROFIBUS master

10. Establish a connection between the two devices (e.g. one byte for synchronization tasks between user programs on the devices); alternatively, you can ignore compilation errors in HW Config resulting from the absence of configured connections.
The connection is configured only once the axes on various devices in SIMOTION SCOUT have been interconnected.

Detailed information on cross-project distributed synchronous operation illustrated with an example configuration can be found in Section PROFIBUS communication configuration (Page 2788). You will find detailed information on communication configuration in SCOUT TIA in Section Communication between SIMOTION two controllers in the SIMOTION SCOUT TIA Configuration Manual.

Creating connections using PROFINET IO with IRT

The creation of a distributed synchronous operation connection requires that at least two IO controllers are connected with each other using IRT; a complete IRT configuration does not need to have been performed:

1. Set the IP addresses. Recommendation: fixed IP addresses
2. Create the two SIMOTION devices.
3. Configure the IRT operation by setting synchronization type **SyncSlave** for one of the devices and **SyncMaster** for the other device in HW Config.

This creates HW Config for distributed synchronous operation. The configuring of the axes and the interconnection of the axes is the same as for PROFIBUS.

Detailed information on cross-project distributed synchronous operation illustrated with an example configuration can be found in Section Communication configuration via PROFINET IO (Page 2799).

Saving and compiling in SIMOTION SCOUT automatically creates the data to be exchanged between the SIMOTION devices for the distributed synchronous operation.

Creating synchronous operation connections with SCOUT

Master axis/external encoder

1. Create the master axis or the external encoder (in the PROFIBUS master for the distribution using PROFIBUS).
2. Configure the master object (master axis/external encoder) using the wizard.

Following axis

1. Create a following axis (in the PROFIBUS slave if distribution is taking place via PROFIBUS). Make sure that **Synchronous operation** is activated as the technology when creating the following axis.

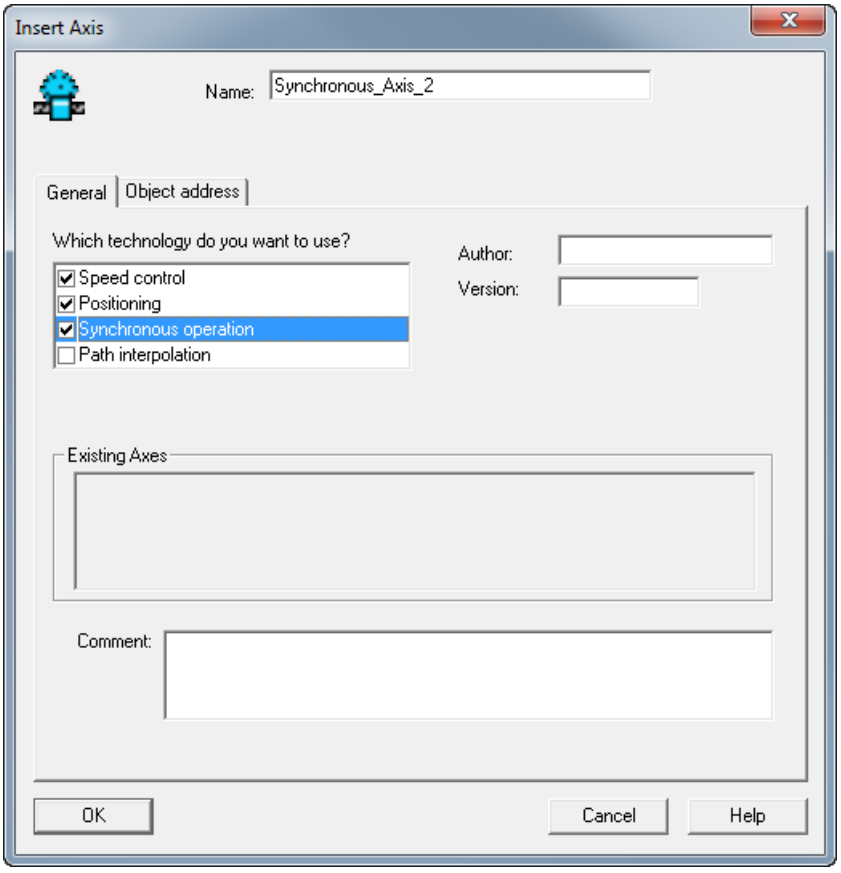


Figure 4-745 Inserting an axis with synchronous operation

2. Configure the following axis using the wizard.
3. Connect the synchronous object for the following axis to the master object (master axis/ external encoder) by selecting the master object under Interconnections.

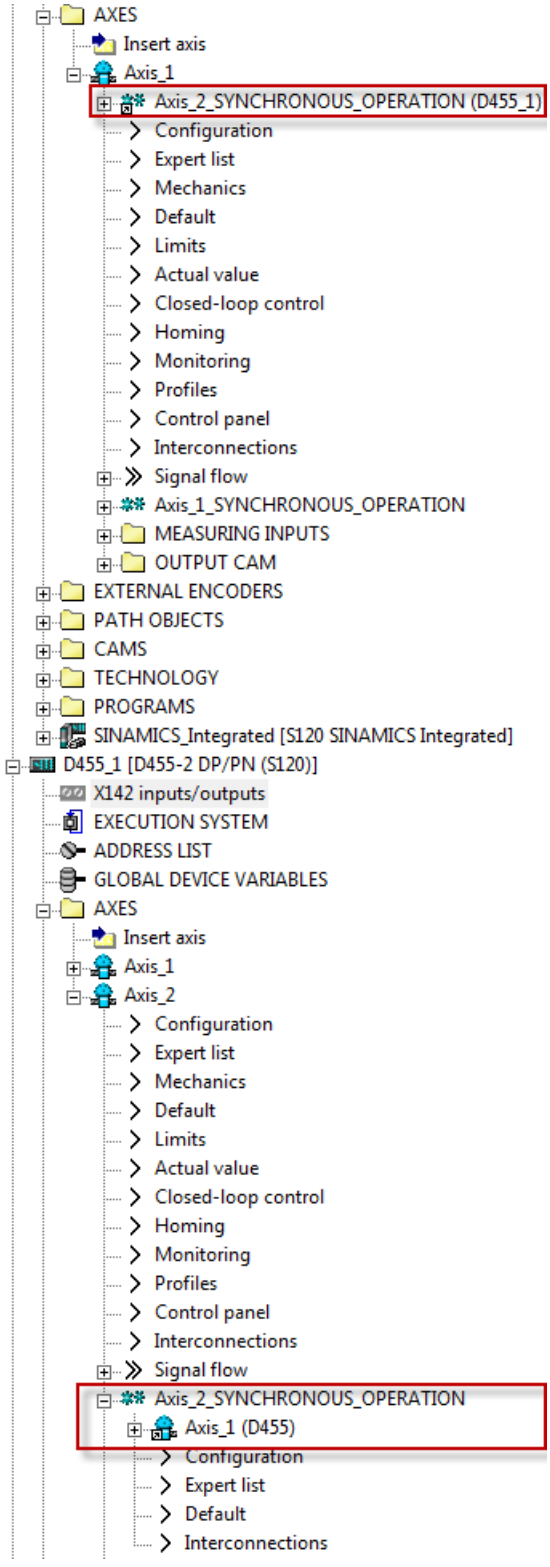


Figure 4-746 Distributed synchronous operation: Connecting a leading axis to a following axis

Generating a synchronous operation configuration

Compiling the project

Save the project via the menu **Project -> Save and compile changes**.

The system automatically generates the PROFIBUS IO configuration data in HW Config for the 24 bytes of send and receive info for the distributed synchronous operation. The configuration is generated and checked for consistency. Both SIMOTION devices must be connected and configured so that the HW Config can be compiled with no errors.

Note

Only at this point can it be determined whether the network resources are sufficient. If they are not, an error message will be generated to this effect. The compilation process must be free of errors; otherwise, it will not be possible to download the project to the target system.

Downloading the project to the target system

Load the project to both SIMOTION devices, so that these are displayed consistently. The configuration for the synchronous operation is also transferred to the SIMOTION devices during loading.

Note

There is no mechanism integrated into the SIMOTION Kernel for checking the consistency of downloaded projects among multiple devices, e.g. during ramp-up. Consistent downloading can only be achieved by means of "Download" in SIMOTION SCOUT.

Possible error

I/O resources unavailable (PROFIBUS)

If the system detects that the required resources (number of bytes) are no longer available in the PROFIBUS I/O data, SCOUT will generate an error message during the compilation process.

Check resources

- Call the PROFIBUS **DP Properties** for the relevant slave in HW Config.

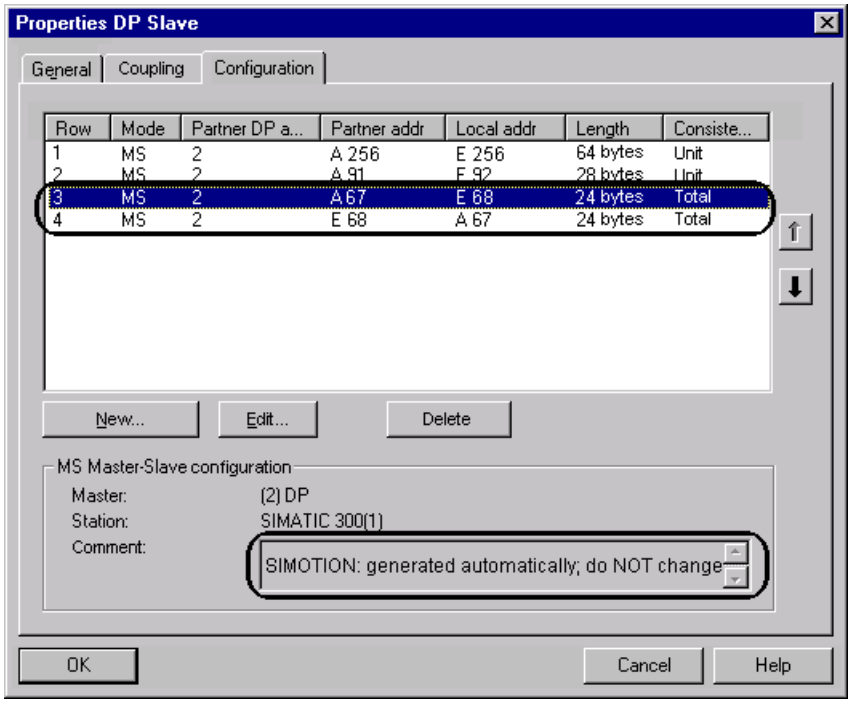


Figure 4-747 View of PROFIBUS resources with automatically generated entries

Note

Do *not* change or delete the entries in the I/O PROFIBUS configuration generated automatically by SIMOTION SCOUT. If you do, the distributed synchronous operation connection can no longer be used.

Faulty/incomplete configuration

If the system detects a fault in the configuration, SIMOTION SCOUT will output an error message during the compilation process.

4.7.3.4 Programming distributed synchronous operation

Synchronizing the interfaces

The interfaces are synchronized in the StartupTask for the slave by means of the **_enableDpInterfaceSynchronizuationMode** function call, either automatically by transferring the **dpInterfaceSyncMode** function parameter with the **AUTOMATIC_INTERFACE_SYNCHRONIZATION** option, or manually by transferring the **MASTER_SLAVE_ALARMMESSAGES_1** option.

If the **_enableDpInterfaceSynchronizationMode** function is called with the **MASTER_SLAVE_ALARMMESSAGES_1** option, the **_synchronizeDpInterface** function must be called at a later point in time to synchronize the interfaces.

If the "automatic" function is used and the connection is interrupted, the interface synchronizes itself automatically once the connection is restored.

If the "automatic" function is not used, the interface must be synchronized manually every time the connection is interrupted.

The **stateOfDpInterfaceSynchronization** = **DP_INTERFACES_SYNCHRONIZED** system variable on the slave indicates whether the two interfaces are synchronized.
Exception for SIMOTION C240:
If drives are connected to a SIMOTION C240 via the analog interface (onboard, e.g. for hydraulic applications), the synchronization is displayed via the **stateOfDpSlaveSynchronization = DP_INTERFACES_SYNCHRONIZED** system variable.
The interfaces must be synchronized to ensure error-free distributed synchronous operation.

Further information is available in the **Motion Control Basic Functions for Modular Machines** Function Manual.

Synchronous commands

If your commands include the object name of the master value source that is located on the other SIMOTION device, you must specify the name of this device as a prefix (e.g. D445.axis_1)

If you require data from technology objects that are located on the other device, you can use the application as a vehicle for this, by expanding synchronous operation communication between the devices.

See also the section titled *Error handling*.

Checking to determine whether the following axis is ready

When the distributed synchronous operation is to be started on the following axis (e.g. through **_enableGearing**), he application must ensure that the synchronized axis is ready.

Compensation on the following value side:

The following conditions must be met on the following axis if the following are activated on the master axis **TypeOfAxis.DistributedMotion.enableOffsetCompensation = YES** and

TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = NO:

- The **stateOfDpInterfaceSynchronization** system variable for the following axis must indicate **DP_INTERFACES_SYNCHRONIZED**.
- The **distributedmotion.stateOfOffsetCalculation** system variable for the synchronous object must indicate **valid**.

Compensation on the master value side:

The following conditions must be satisfied on the following axis if the following are activated on the master axis **TypeOfAxis.DistributedMotion.enableOffsetCompensation = YES** and **TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = YES**:

- The **stateOfDpInterfaceSynchronization** system variable for the following axis must indicate **DP_INTERFACES_SYNCHRONIZED**.
- The **distributedmotion.stateOfOffsetCalculation** system variable for the synchronous object must indicate **valid**.
- The **distributedMotion.stateOfDelayValue** system variable for the synchronous object must indicate **valid**.

No compensation:

When **TypeOfAxis.DistributedMotion.enableOffsetCompensation = NO** and **TypeOfAxis.DistributedMotion.enableDelayOfCommandValueOutput = NO** is activated on the master axis, the function is the same as for compensation on the following value side.

If this is not the case (for example, if **_enableGearing()** is executed on the following axis even though the master axis is not ready), the command is aborted with error "50102 Master is not assigned/configured or is faulty (reason: ..." aborted).

4.7.3.5 Configuring distributed synchronous operation across projects

Overview

As of V4.1, it is possible to configure a distributed synchronous operation across projects; this is carried out using proxy objects. The SIMOTION devices for the master object and the following axis are located in separate SIMOTION projects.

In each case, there is a proxy "External Synchronous Operation" technology object, which is assigned to the master axis, and a proxy "External Master Value" technology object, which is assigned to the slave axis. The external interface of the proxy technology objects is formed by means of coupling via I/O addresses. The network configuration must be set in HW Config. The send and receive data must be interconnected with one another. From the point of view of the runtime system, *distributed synchronous operation* and *cross-project distributed synchronous operation using proxy technology objects* are identical.

With regard to version requirements, see Overview of distributed synchronous operation with different RT versions (Page 2822)

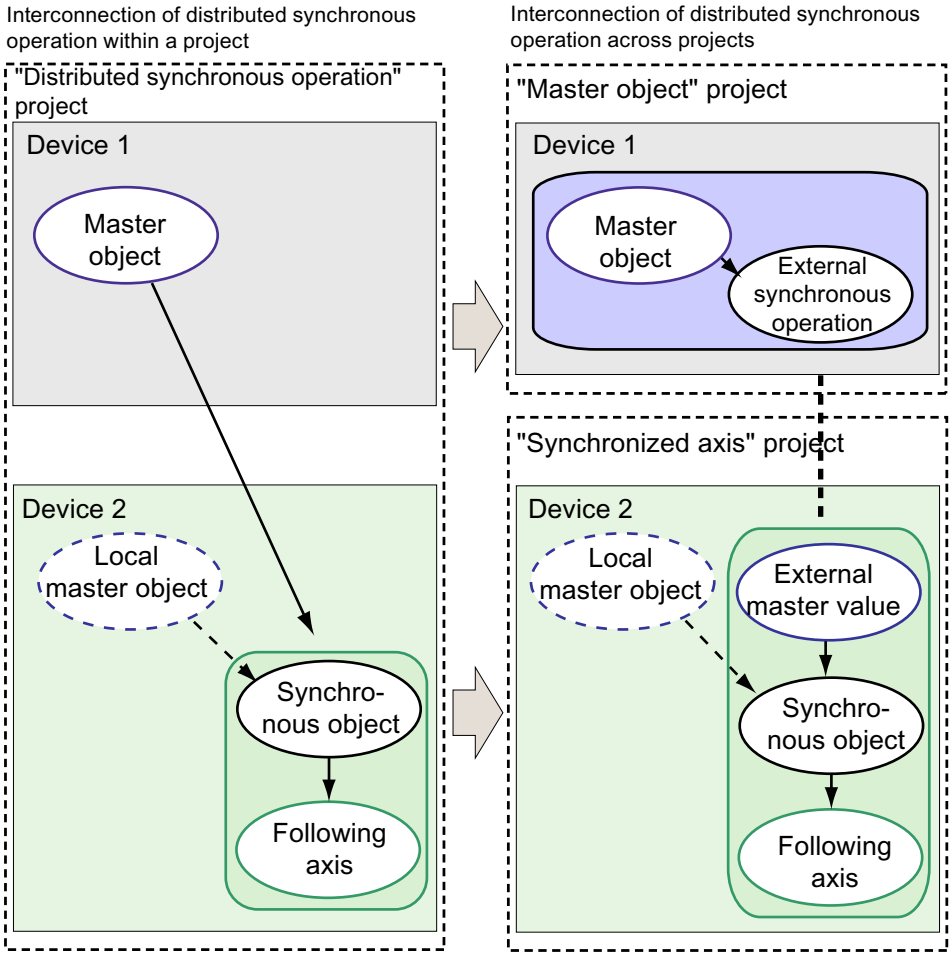


Figure 4-748 Configuration of a cross-project distributed synchronous operation

This figure shows the logical view of a synchronous operation interconnection. If the objects are located in the same project, the master object (master) and the synchronous object (slave) can be directly interconnected (see left-hand side of figure).

If the master object and synchronous object are in different SIMOTION devices that are not located in the same project, they must be interconnected using proxy objects. In each case, these represent the external object (see right-hand side of figure). External synchronous operation reflects the "transmitter master value," whereas the external master value reflects the "receiver master value."

This description uses the following terms:

Table 4-319 Terms used

Master/slave	PROFINET IO	PROFIBUS DP
Master object on	IO controller	DP master
Synchronous object on	I device	I slave

The description below uses an example to explain how communication is configured for cross-project distributed synchronous operation.

Communication configuration for PROFIBUS DP

The following steps need to be carried out for configuration using PROFIBUS DP:

- Create and configure a "synchronized axis" project; see Section Creating and configuring a "synchronized axis" project (Page 2789).
- Create and configure a "master object" project; see Section Creating and configuring a "master object" project (Page 2796).
- Create proxy objects; see Section Creating proxy objects (Page 2814).

Communication configuration for PROFINET

The following steps need to be carried out for configuration using PROFINET:

- Create and configure a "synchronized axis" project; see Section Creating and configuring "Synchronous axis" project on SIMOTION D455-2 (Page 2800).
- Create and configure a "master object" project; see Section Creating and configuring "Master Object" project on SIMOTION D455-2 (Page 2808).
- Create proxy objects; see Section Creating proxy objects (Page 2814).

See also

PROFIBUS communication configuration (Page 2788)

Communication configuration via PROFINET IO (Page 2799)

Proxy objects (Page 2814)

PROFIBUS communication configuration

Communication via PROFIBUS DP

This section uses an example to describe how cross-project distributed synchronous operation is configured using PROFIBUS DP. You can adapt the project properties to suit your system requirements. Please note that you must set the entire PROFIBUS configuration. This example only refers to configuring cross-project distributed synchronous operation.

Introduction

If the SIMOTION devices are connected via PROFIBUS, data exchange takes place between a DP master and a DP slave. The SIMOTION device for the "master object" project is configured as the DP master and the SIMOTION device for the "synchronized axis" project is configured as the DP slave. In the "master object" project, in order to configure data exchange with the DP slave in the "synchronized axis" project the DP slave on the PROFIBUS line must be configured with the "GSD file" of the SIMOTION device.

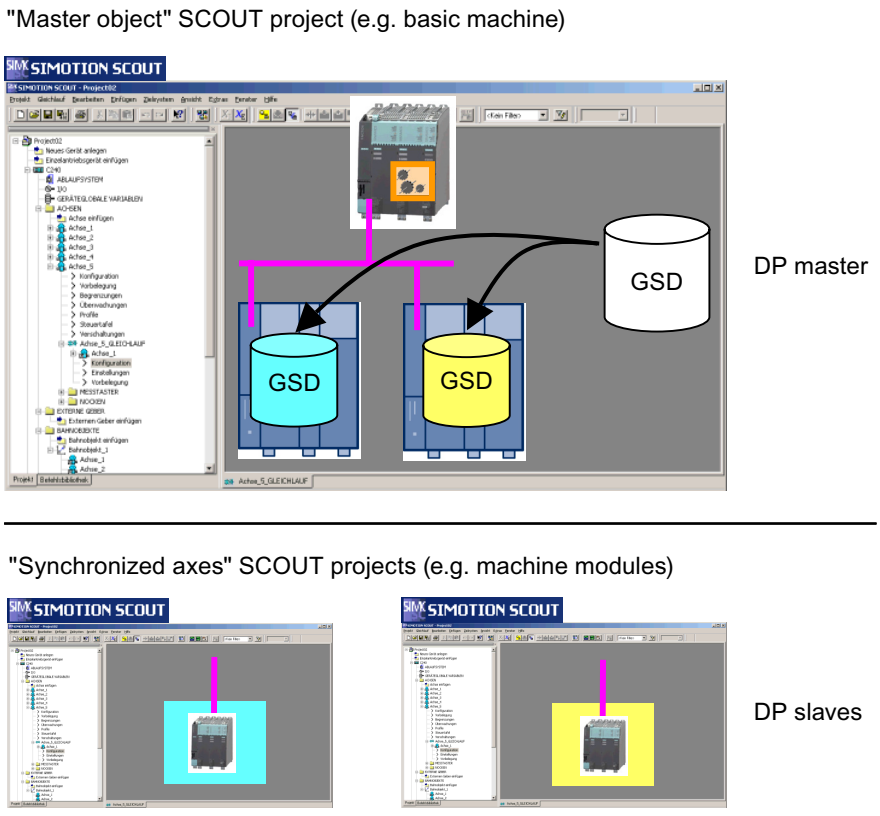


Figure 4-749 PROFIBUS DP: HW configurations of DP slaves via GSD

Creating and configuring a "synchronized axis" project

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, communication takes place via PROFIBUS.

Creating a DP slave (project creation in SIMOTION SCOUT)

1. Open SIMOTION SCOUT and create a new project named "synchronous axis".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. In the "Interface Selection - D455-2" window, select the "PROFIBUS DP1 (X126)" interface by clicking "OK" and click "OK" again to confirm.
   HW Config opens.

Configuration of the interface in HW Config

1. In the rack, double-click the "DP" interface.
   The "Properties - DP - (R0/S2.1)" window opens.
2. Click the "Properties..." button on the "General" tab.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens. Address 2 is displayed as the default PROFIBUS address. Leave this setting unchanged.
3. Click the "New..." button in the "Subnet" area to create a subnet.
   The "Properties - New PROFIBUS subnet" window opens.
4. On the "General" tab, enter the name "Synchronous operation" at "Name".
5. Click the "Network settings" tab and set the recommended value of 12 MBit/s as the transmission rate.
6. Click "OK" to confirm your settings. You will then be in the "Properties - DP1 - (R0/S2.1)" window.

Setting DP slave

1. Click the "Operating Mode" tab and activate "DP Slave."
   Ensure that the option button "Programming, status/control or other PG functions and non-configured communication connection possible" is not activated, deactivate it if necessary. When the option is activated, no isochronous operation is possible.

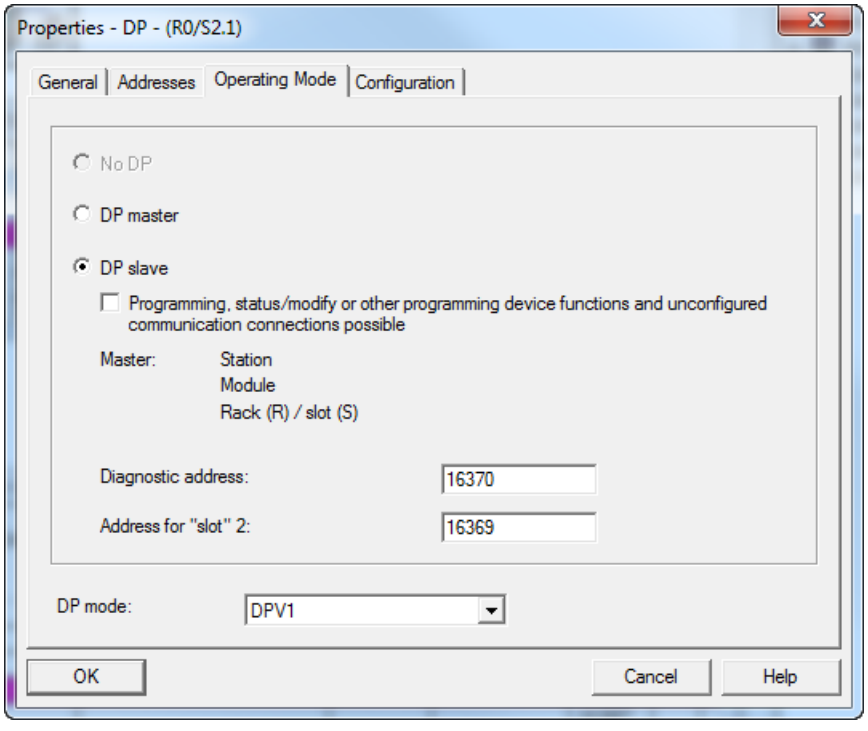


Figure 4-750 Setting the properties for the DP slave

2. Click the "Configuration" tab to create input and output addresses for communication.

3. Click the "New..." button.
   The "Properties - DP - (R0/S2.1) - Configuration - Line 1" window opens.
   Enter the following settings for the input data:
   - Address type: Input
   - Address: 256
   - Length: 12
   - Unit: Word
   - Consistency: Unit
   , and click "OK" to confirm your settings.

Note

In this example, you enter 256 as the first available address. If you have already configured other objects in the project, such as a drive, you will need to enter the first potentially available address.

Please note that you will need the set input and output addresses later when configuring the external master value in SIMOTION SCOUT.

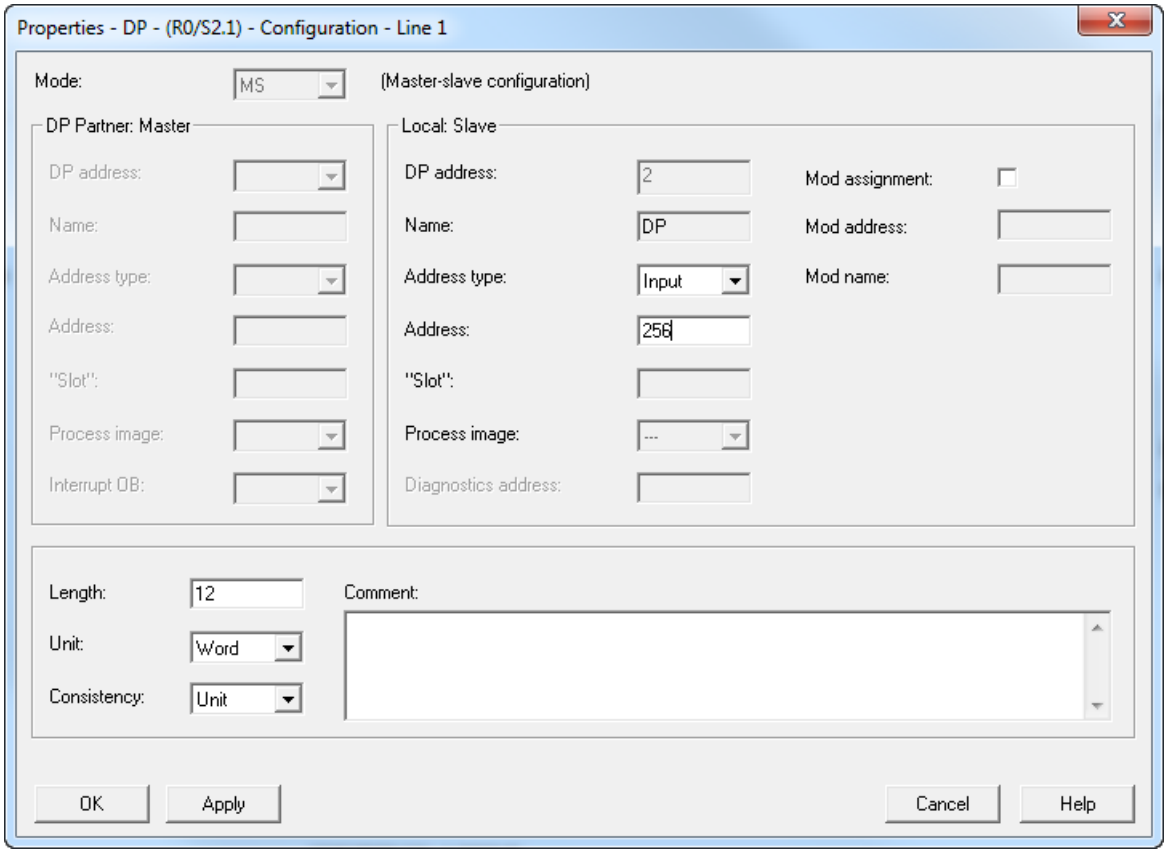


Figure 4-751 Configuring the input address on the slave

4. Click "New...".
   The "Properties - DP1 - (R0/S2.1) - Configuration - Line 2" window opens.
   Enter the following settings for the output data:
   - Address type: Output
   - Address: 256
   - Length: 12
   - Unit: Word
   - Consistency: Unit
   , and click "OK" to confirm your settings.

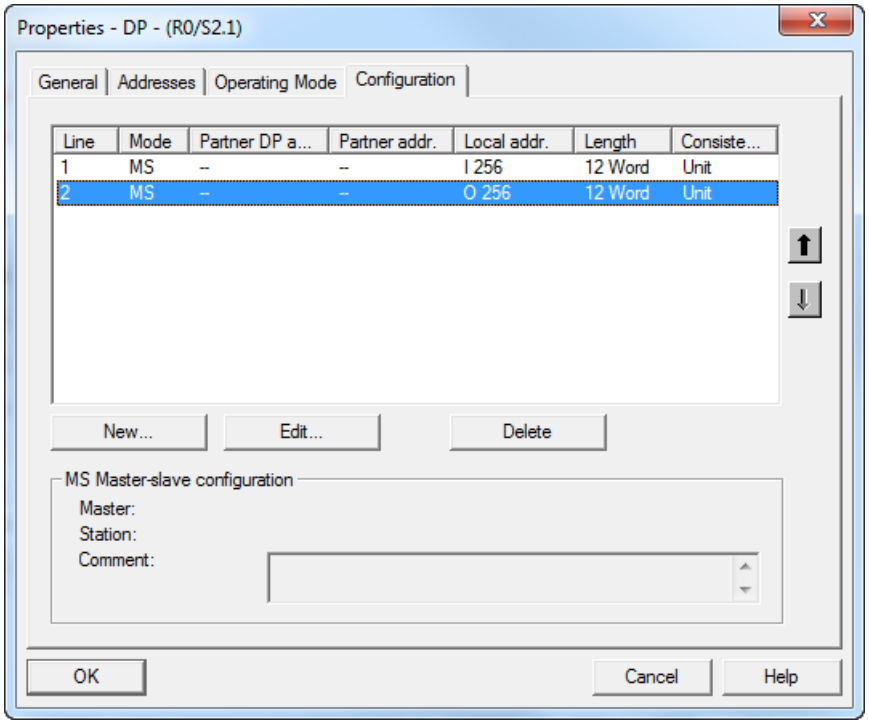


Figure 4-752 Configured input and output addresses on the DP slave

Note

Please also note that, in addition to the data for the synchronous operation, you may also want to transfer user data (communication between master and slave, such as project-specific control and status data for synchronous axes) and will, therefore, need to configure further lines with additional input and output addresses accordingly.

5. Click "OK" to confirm your settings, followed by the "Save" button.

In order to configure an isochronous PROFIBUS line, you must now use HW Config to create and configure a new station ("Dummy master") in the same project. This "dummy master" is only required to act as a proxy and does not have to be physically present.

Configuring the dummy master

It is recommended that you configure an S7 CPU as the master, so that the slave project remains clear and the dummy master is not visible in SIMOTION SCOUT.

1. Open or switch to HW Config of the DP slave.
2. In the HW Config menu bar, click "Station > New...".
   The "New" window opens. Enter "dummy master" as the object name and select the SIMATIC 300 station as the object type.

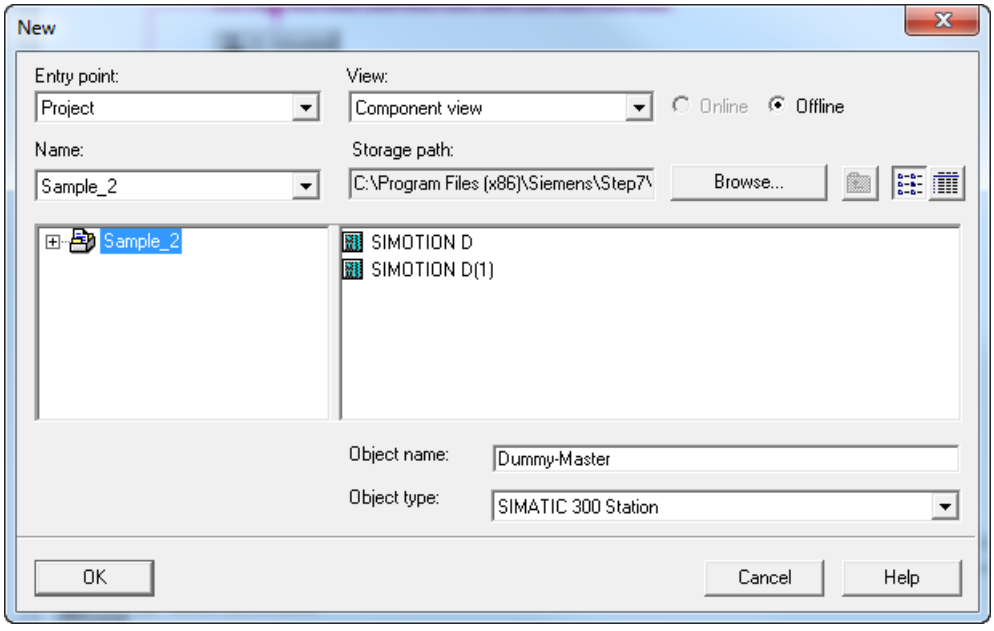


Figure 4-753 Creating a new station ("Dummy master") in HW Config

3. Click "OK" to confirm your settings. The new "Dummy master" station opens

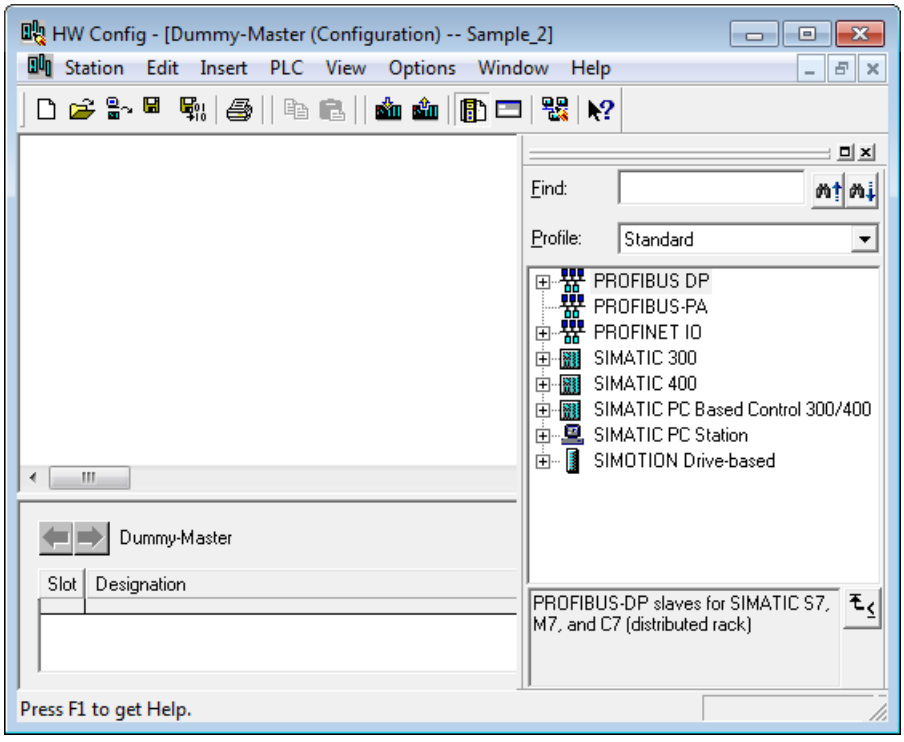


Figure 4-754 New station in HW Config

4. In the hardware catalog, open "SIMATIC 300 > RACK 300". Double-click the "Mounting rail". This is inserted in HW Config.

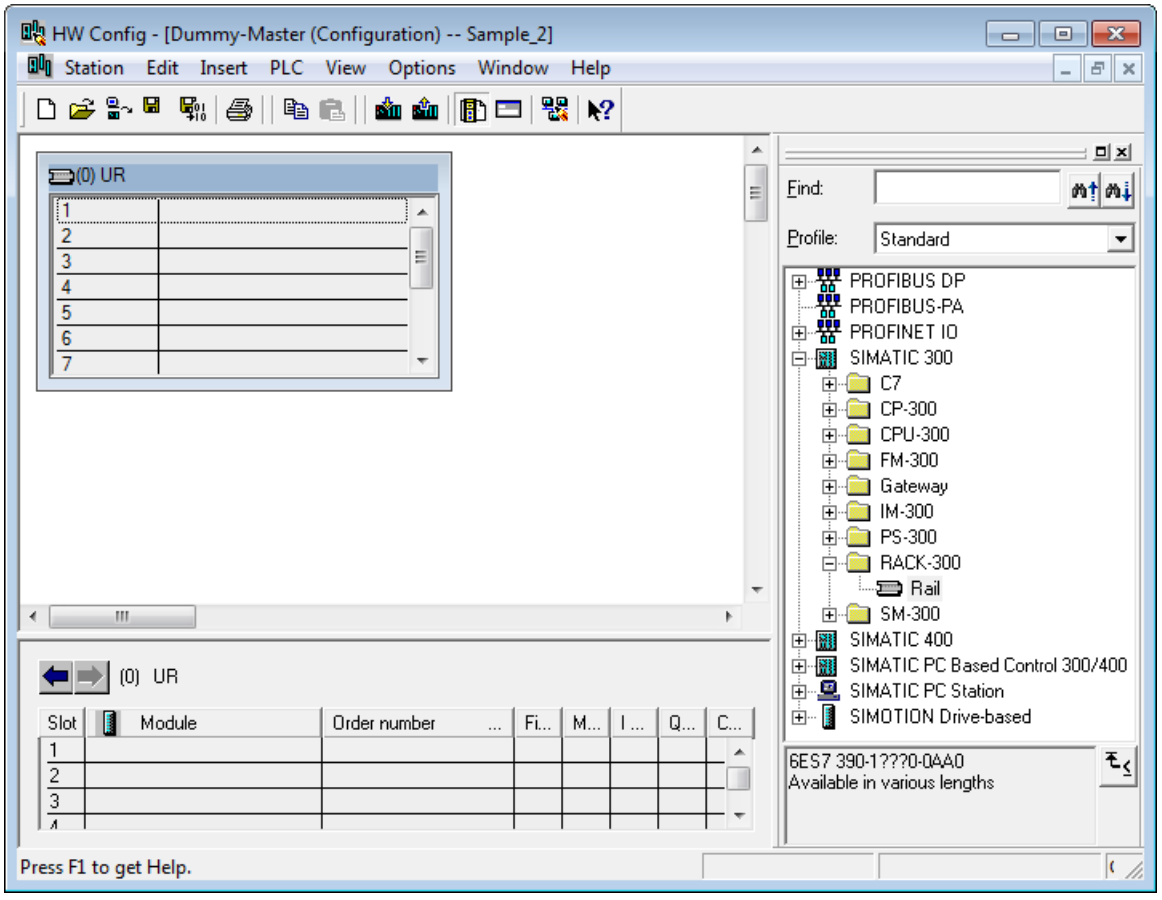


Figure 4-755 Inserting a mounting rail in HW Config

5. In the hardware catalog, open SIMATIC 300 > CPU 300 > CPU 316-2 DP > 6ES7 316-2AG00-0AB0. Select "V1.2" and move the module to its designated slot (highlighted in green) using drag-and-drop.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens. It is not necessary to set the operating mode for the dummy master.
6. Select PROFIBUS address 3 as the address. The PROFIBUS address of the dummy master must be identical to that of the DP master in the "master object" project. You will create the "master object" project later.
7. Click "New...". The "New" window opens.
8. Click the "Network settings" tab and set the recommended value of 12 MBit/s as the transmission rate.
9. Click the "Options..." button. The "Options" window opens.
10. Activate the "Activate equidistant bus cycle" function and enter 3 ms for the "Equidistant DP cycle." This value must also be set later in the "master object" project.

11. Click "OK" to confirm your settings in the open window, followed by the "Save" button.
12. Switch to SIMOTION SCOUT and save the "synchronous axis" project.
    This concludes configuration of the DP slave communication in HW Config. The next step involves creating the "master object" project in SIMOTION SCOUT, see section Creating and configuring a "master object" project (Page 2796).

Creating and configuring a "master object" project

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, communication takes place via PROFIBUS.

Creating a "Master object" project in SIMOTION SCOUT

1. Open SIMOTION SCOUT and create a new project named "Master Object."
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. In the "Interface selection - D435" window, select the "PROFIBUS DP1 (X126)" interface by clicking "OK" and click "OK" again to confirm.
   The HW Config opens.

Configuration of the interface in HW Config

1. In the rack, double-click the "DP" interface.
   The "Properties - DP - (R0/S2.1)" window opens.
2. Click the "Properties" button on the "General" tab.
   The "Properties - PROFIBUS interface DP - (R0/S2.1)" window opens.
3. Select the PROFIBUS address by putting 3 in the "Address" field. The address of the DP master must be identical to that of the dummy master in the "synchronous axis" project.
4. Click the "New..." button in the "Subnet" area to create a subnet.
   The "Properties - New PROFIBUS subnet" window opens.
5. On the "General" tab, enter the name "Synchronous operation" at "Name".
6. Click the "Network settings" tab and set the recommended value of 12 Mbit/s as the transmission rate, then click "Options".
   The "Options" window opens.
7. Activate the "Activate equidistant bus cycle" function and enter 3 ms for the "Equidistant DP cycle."
8. Click "OK" to confirm your settings in the open window.

Setting DP Slave

1. In the hardware catalog, open "PROFIBUS DP > Additional field devices > PLC > SIMOTION" and select "SIMOTION D4xx".
2. Using drag-and-drop, move "SIMOTION D4xx" (DP slave) to the "Synchronous operation: DP master system (2)" subnet. The "Properties – PROFIBUS interface SIMOTION D4xx" window opens.
3. Select PROFIBUS address 2 as the address and confirm the settings with "OK". This PROFIBUS address must be identical to that of the DP slave in the "synchronous axis" project.

4. Insert a 12-word module from the hardware catalog for the slave outputs first. Under "SIMOTION D4xx" in the hardware catalog, select the "Master_O Slave_I 12Wo/Unit" module and move it to its designated slot (highlighted in green) using drag-and-drop.

Note

Please note that the the inputs and outputs must always be configured in the opposite direction. An input slot must always go to an output slot (and vice-versa). That means that if the first slot is configured as an input in the "synchronous axis" project, then the first slot in the "master object" project has to be configured as an output. The length of configured inputs and outputs must always be identical.

The addresses for the inputs and outputs must lie above the first 64 bytes of the logic address area. You will need this address again later for configuring the proxy object.

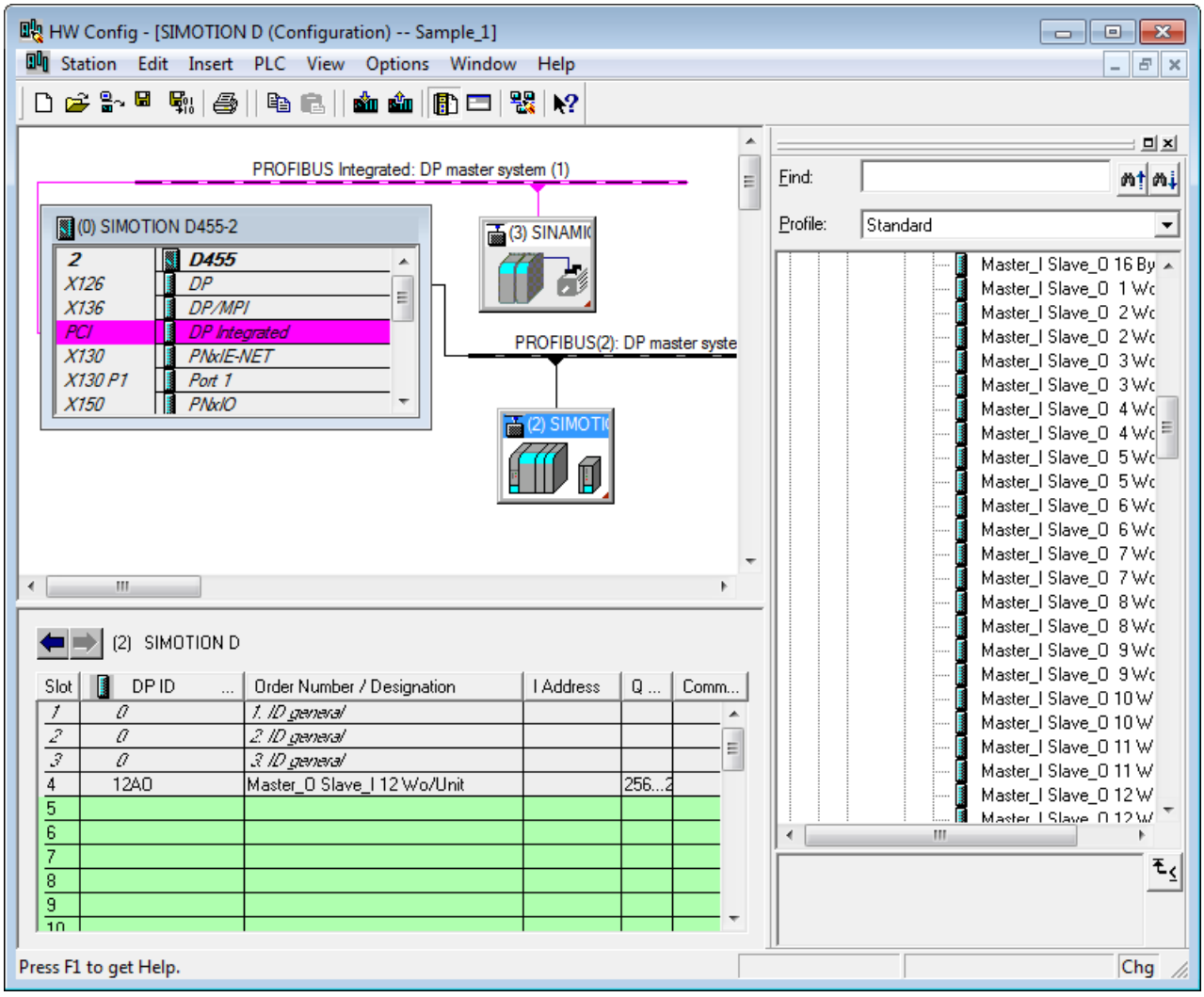


Figure 4-756 Creating an output module on the DP slave in HW Config

5. Under "SIMOTION D4xx" in the hardware catalog, select the "Master_I Slave_O 12Wo/Unit" module and move it to its designated slot (highlighted in green) using drag-and-drop.

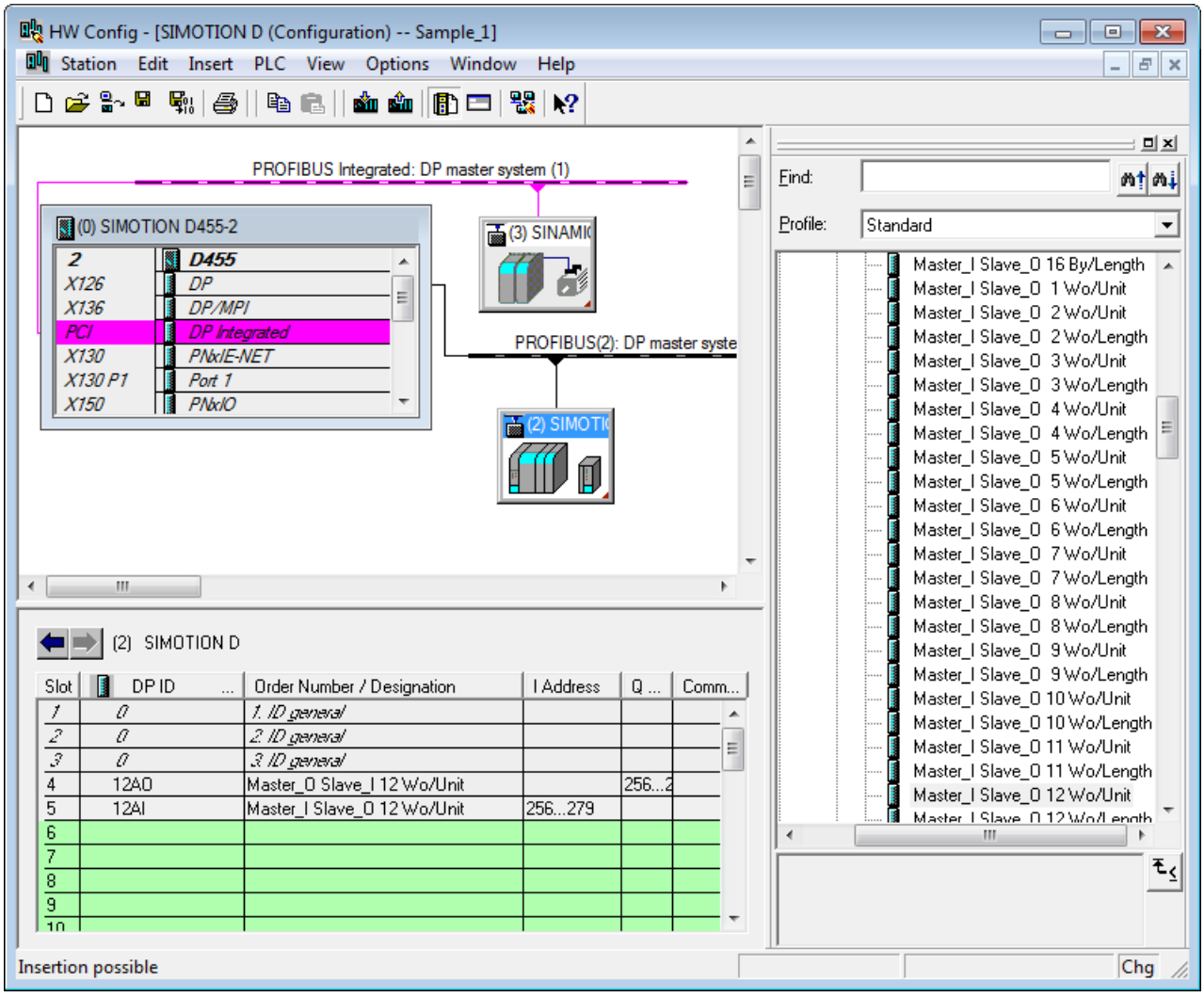


Figure 4-757 Creating an input module on the DP slave in HW Config

6. Click the "Save" button.
7. Switch to SIMOTION SCOUT and save the "master object" project.
   This concludes configuration of the DP master communication in HW Config. The next step involves creating proxy objects in SIMOTION SCOUT, see section Creating proxy objects (Page 2814).

Communication configuration via PROFINET IO

Communication via PROFINET IO

This section uses an example to describe how cross-project distributed synchronous operation is configured using PROFINET IO. You can adapt the project properties to suit your requirements.

Introduction

If the devices are connected via PROFINET, data exchange takes place between an IO controller and an I device. The I device must be taken into account during the configuration of the I/O controller by means of its GSDML file. The GSDML can be generated in HW Config of the I device.

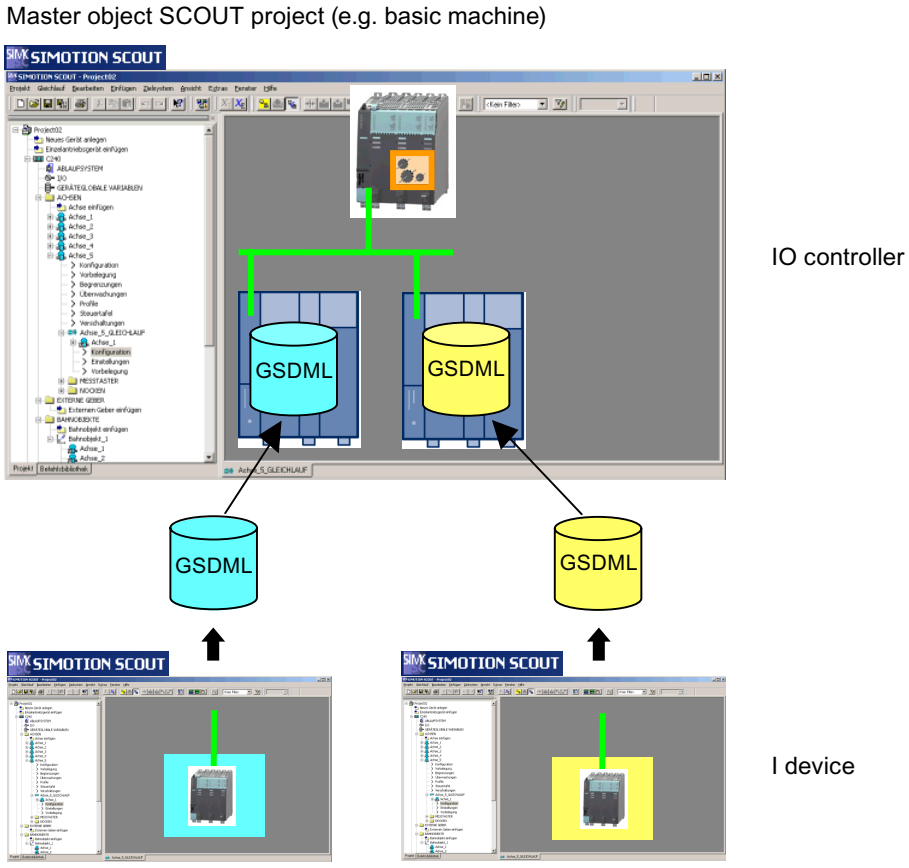


Figure 4-758 PROFINET distribution bus: HW configurations of the basic machine and modules

Note

The I device can be configured in STEP 7 as of V5.4 SP2.

Note

Distributed synchronous operation is also possible via two PN interfaces, see Section *Example configuration for controlled Sync Master* in the *Communication* System Manual.

Creating and configuring "Synchronous axis" project on SIMOTION D455-2

This description is just an example. You can adapt the project properties to suit your system requirements. In this example, configuration using PROFINET V2.2 is described. SIMATIC STEP 7 V5.5 SP1 must be installed for this.

The creation and configuration of a "synchronous axis" project on SIMOTION D455-2 consists of the following steps:

Project creation in SIMOTION SCOUT

1. Open SIMOTION SCOUT and create a new project named "synchronous axis".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".
4. Click on "New..." in the "Properties - Ethernet interface PNxIO" window.
5. In the following window, add a new Industrial Ethernet subnet with the name "Synchronous operation".
6. Make the interface selection for PG/PC connection.
7. Click "OK" to confirm your settings. HW Config opens.

Configuration of I device in HW Config

1. In the rack, double-click the PNxIO interface to change the device name for the PROFINET interface of the I device, and enter the device name "pn1a."
   This is necessary because you are working with 2 projects and the default name is the same for both. retaining these same names would result in consistency problems at a later point.
2. In this same window, click the "PROFINET" tab and set the required send cycle clock. In this example, this is set to 2 ms.

Note

You can set the cycle clocks in two ways:

- You can set the send cycle clock so it is equal to the DP cycle clock on the drive.
- You can set the send cycle clock so that it is lower than the DP cycle clock: By opting for this you will be working with cycle clock scaling. The position control cycle clock (master application cycle) and the DP cycle on the drive must then be of an equal length.

3. In this same window, click the "I device" tab.

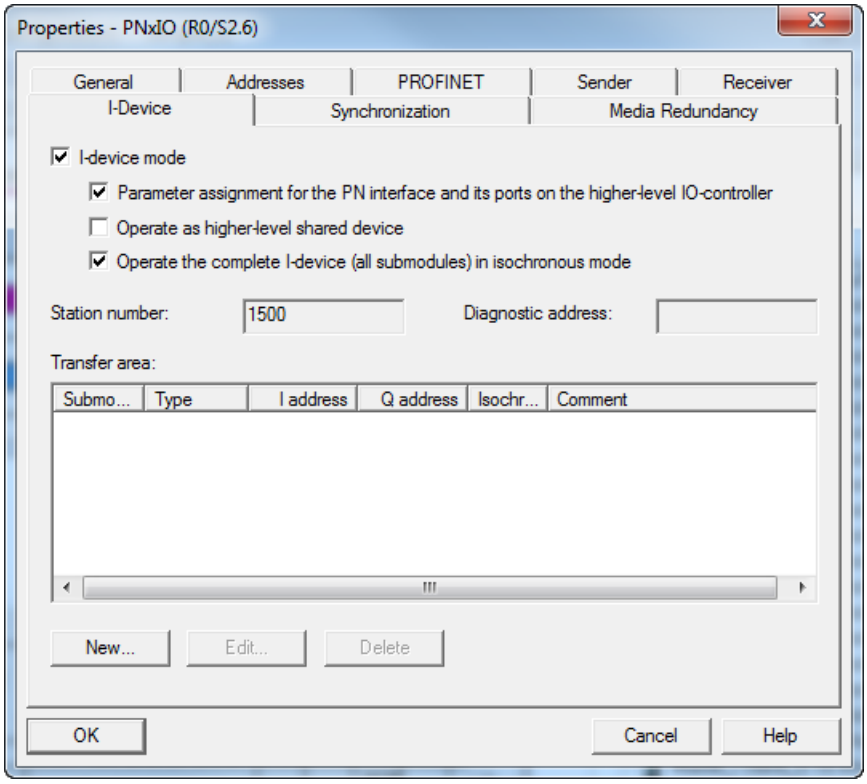


Figure 4-759 Settings on the I device

4. Activate the "I device mode", "Parameterization of the PN interface and its ports on the higher level IO controller" and "Operate complete I device (all submodules) isochronously" functions.
5. In the lower area of this same tab, click "New...".
The "Properties of virtual submodule" window opens.

6. Enter the following properties for the send and receive data:
   - Address type: Input
   - Start: 256
   - Length: 24
     and click "OK" to confirm your settings.

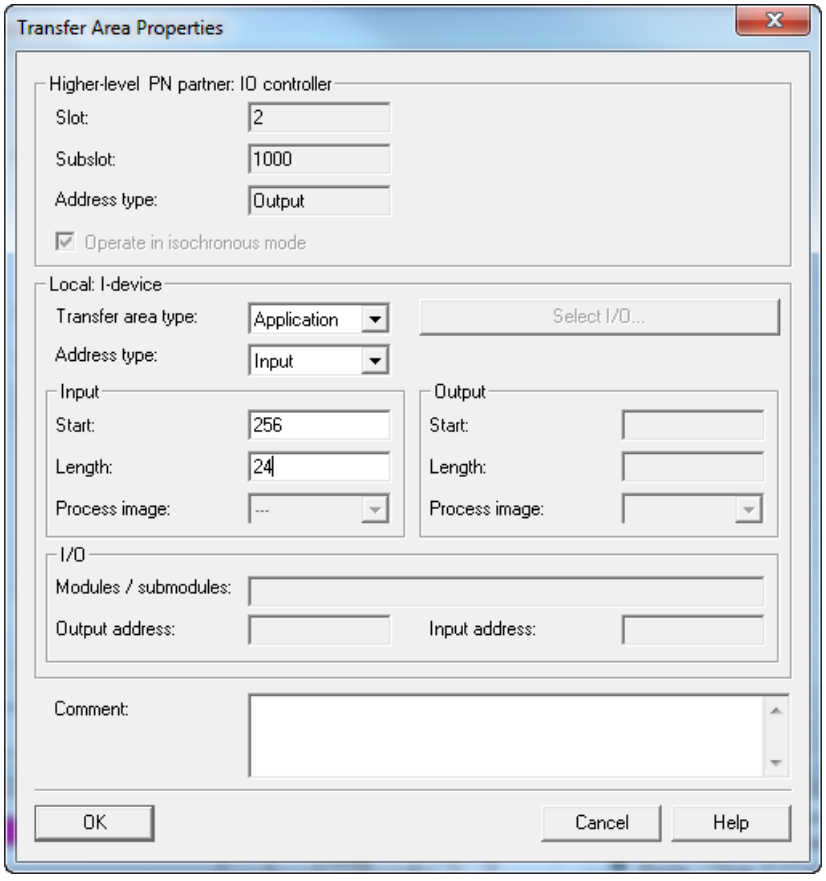


Figure 4-760 Configuring settings on the I device

7. Click the "New..." button again to configure the output address as described below:
   – Address type: Output
   – Start 256

– Length 24
and click "OK" to confirm your settings.
You will need the set input and output addresses later to configure the proxy object in SIMOTION SCOUT.

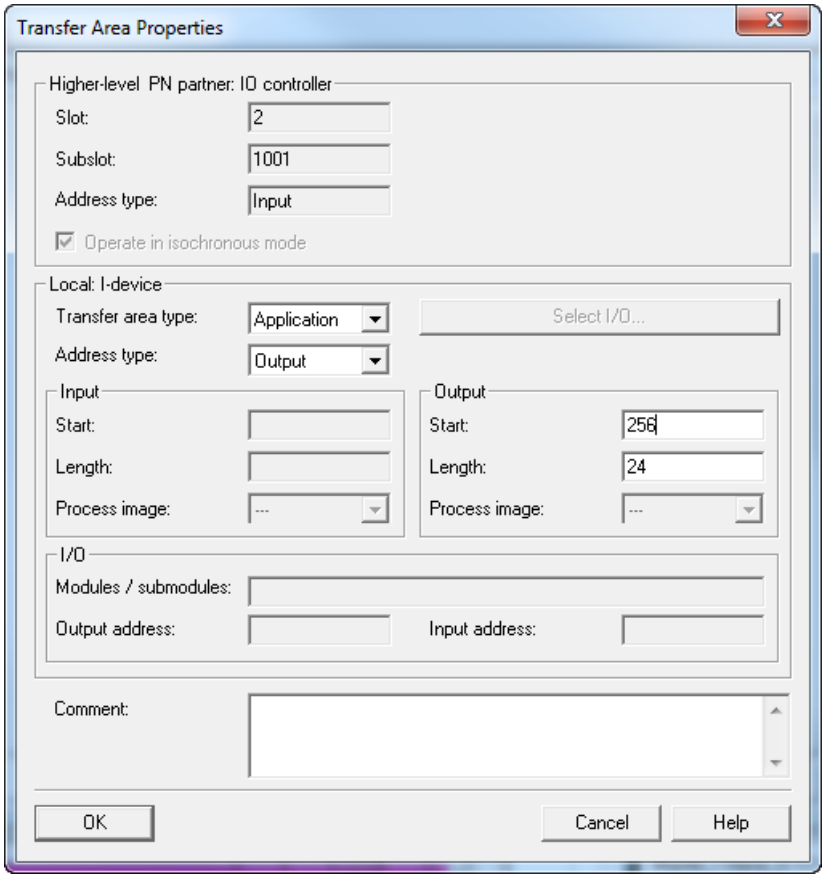


Figure 4-761 Configuring output addresses on the I device

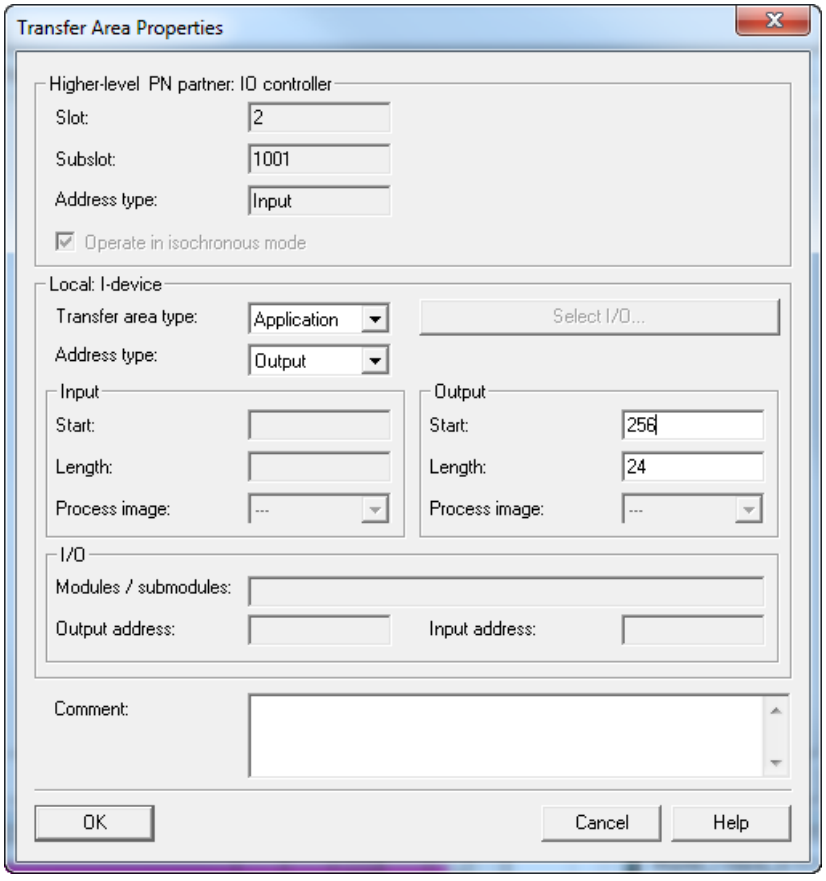


Figure 4-762 Settings on the I device - Addresses

Note

Please note that, in addition to the data for the synchronous operation, you may also want to transfer user data (communication between master and slave, such as project-specific control and status data for synchronous axes) and will, therefore, need to configure additional inputs and outputs accordingly.

8. Confirm your entries with "OK". The I device is created.

Setting DP cycle

1. Double-click on SINAMICS_Integrated.
   The "DP slave properties" window opens.
2. Click the "Isochronous operation" tab and set the "DP cycle" to factor 16. This corresponds to a DP cycle clock of 2 ms.
3. Confirm your entry with "OK".
4. Click the "Save" button.

Creating the GSD file for the I device

1. On the menu bar, select Options > Create GSD for I Device.
   The "Create GSD for I Device" window opens.

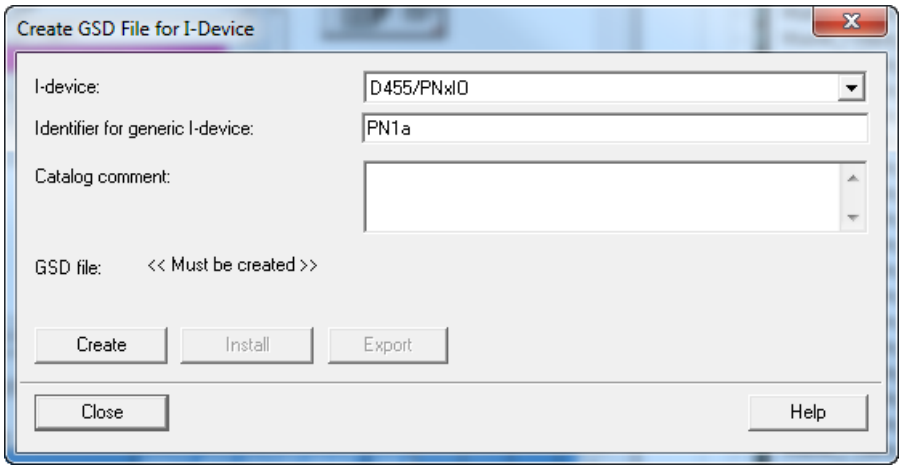


Figure 4-763 Creating the GSDML file

2. Click the "Create" button. This creates a GSDML file for the I device. You will need this file as the proxy for the I device in the "master object" project on the PROFINET IO controller. The name of the GSDML file is shown.
3. Click the "Install" button.

4. Click "OK" to confirm the message that appears, followed by the "Close" button. The GSDML file will now be entered in the HW Config hardware catalog in the PROFINET IO > Preconfigured Stations/D455 structure.

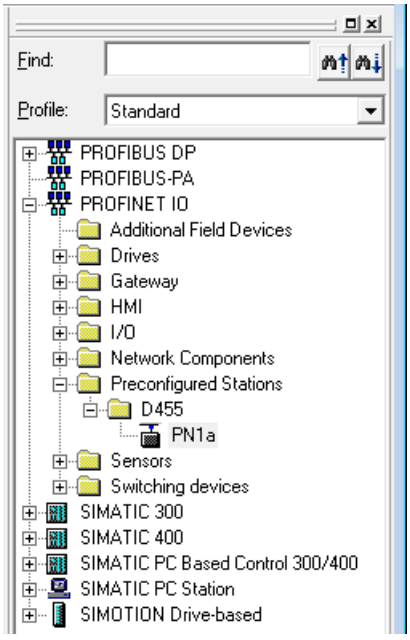


Figure 4-764 View of HW Config after installation of GSDML file

Note

If the project for the PROFINET IO controller is not to be executed with the same engineering system, you must export the GSDML file. The GSDML file must then be imported into HW Config of the other engineering system.

5. Click the "Save" button.
6. Switch to SIMOTION SCOUT and save the "Synchronous axis" project.
This concludes configuration of the I device communication in HW Config. The next step involves creating the "master object" project in SIMOTION SCOUT, see section Creating and configuring "Master Object" project on SIMOTION D455-2 (Page 2808).

Creating and configuring "Master Object" project on SIMOTION D455-2

This description is just an example. You can adapt the project properties to suit your system requirements.

The creation and configuration of a "master object" project for communication via PROFINET IO consists of the following steps:

Project creation in SIMOTION SCOUT

1. Create a new project in SIMOTION SCOUT and call it "Master object".
2. Double-click on "Add SIMOTION device" and add a SIMOTION D455-2 DP/PN V4.3.
3. Click "OK".

4. Click on "New..." in the "Properties - Ethernet interface PNxIO" window.
5. In the following window, add a new Industrial Ethernet subnet with the name "Synchronous operation".
6. Make the interface selection for PG/PC connection.
7. Click "OK" to confirm your settings. HW Config opens.

Setting synchronization type

1. In the rack, double-click on the PNxIO interface.
   The "Properties - PNxIO (R0/S2.10)" window opens.
2. Click the "Synchronization" tab and set Sync-Master as the synchronization type.

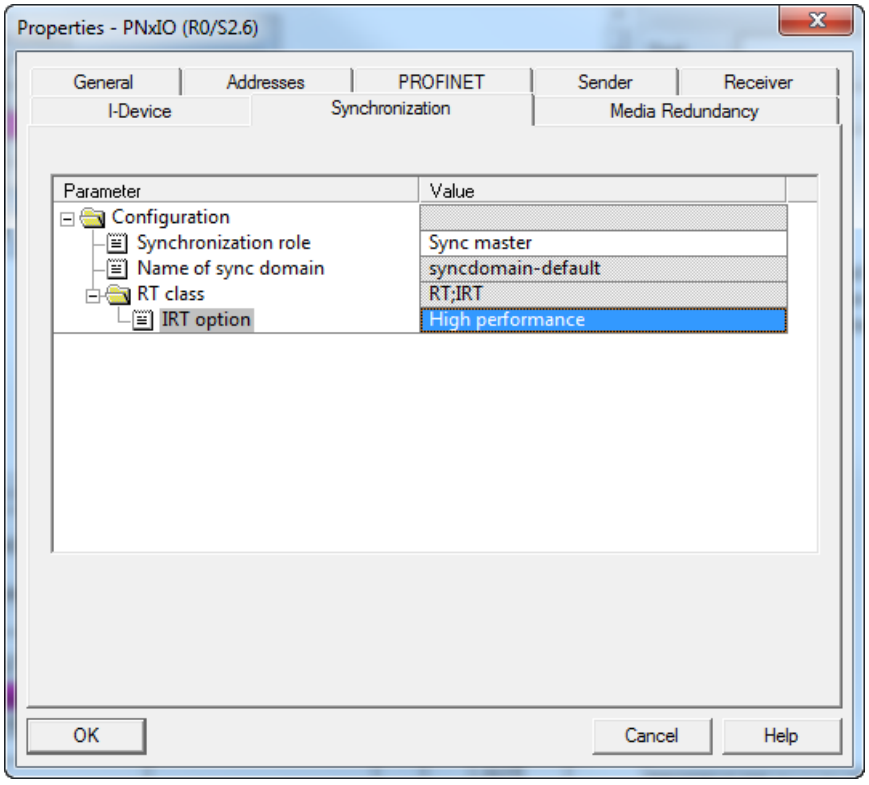


Figure 4-765 Setting Sync-Master on the IO controller

3. Click "OK" to confirm your settings.
4. On the menu bar, select "Edit > PROFINET IO > Domain management..."
   The "Domain management - Synchronous operation" window opens.

5. Set the required send cycle clock: In our example, this is 2 ms.

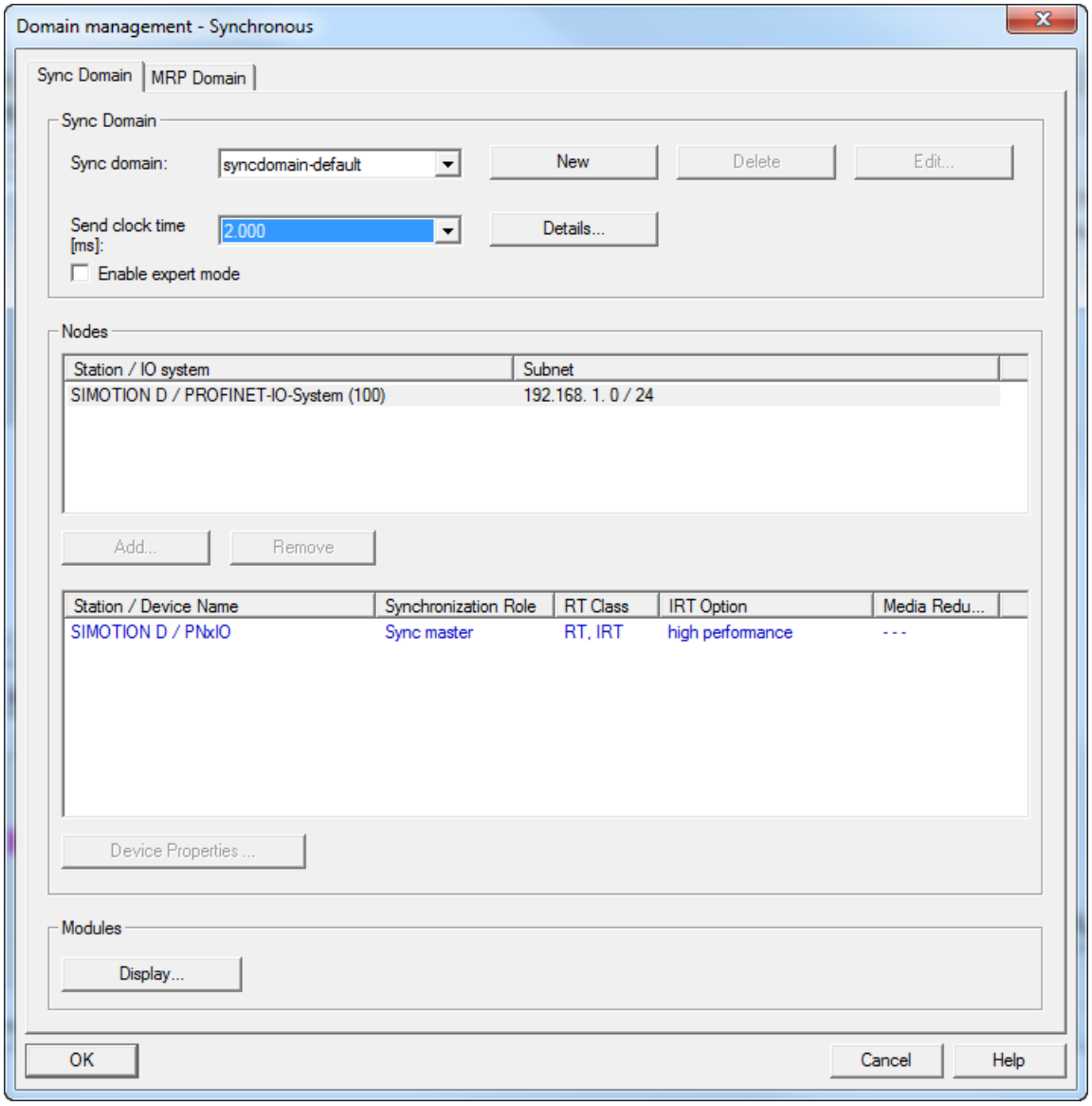


Figure 4-766 Setting the sending cycle on the IO controller

6. Click "OK" to confirm your settings.

Interconnect to PROFINET IO I device

1. Using drag-and-drop, move the I device from the hardware catalog to the "Synchronous operation: PROFINET IO system (100)" subnet. The proxy for the I device is located in the hardware catalog in the PROFINET IO > Preconfigured Stations > D455 structure.

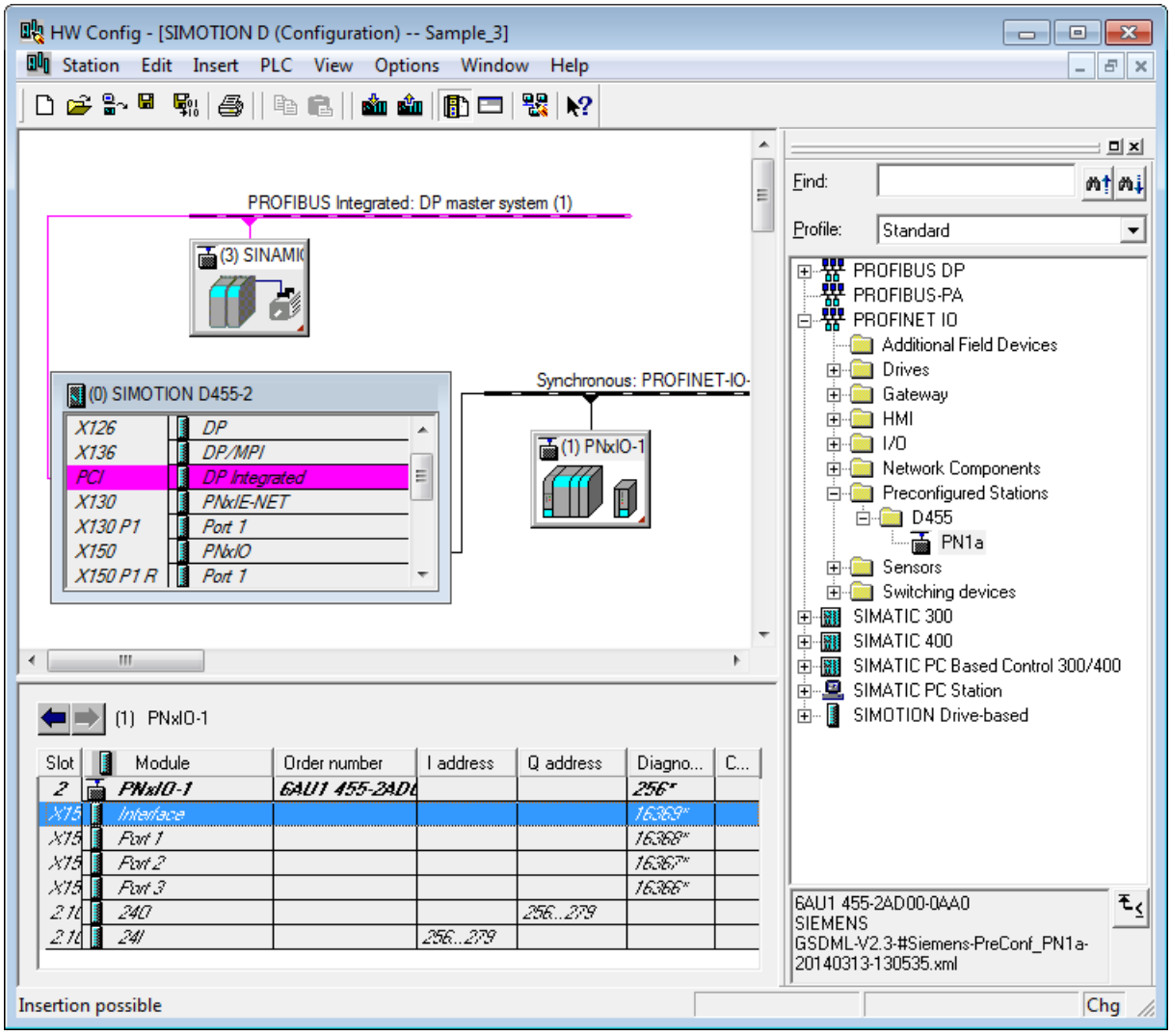


Figure 4-767 View of HW Config after insertion of the I device

Note

When you insert the proxy for the I device, the logic addresses for the cyclic input and output data of HW Config are preassigned. If necessary, correct these addresses before continuing with the configuration of the proxy objects in SIMOTION SCOUT.

2. Select the I device and double-click "Interface" in the detail view.
   The "Properties - Interface (X150)" window opens.
3. Click the "Synchronization" tab and set Sync-Slave as the synchronization type.

4. In this same window, click the "IO Cycle" tab and select the "Servo" entry in the "IO-Device/ assign isochronous application" drop-down list.

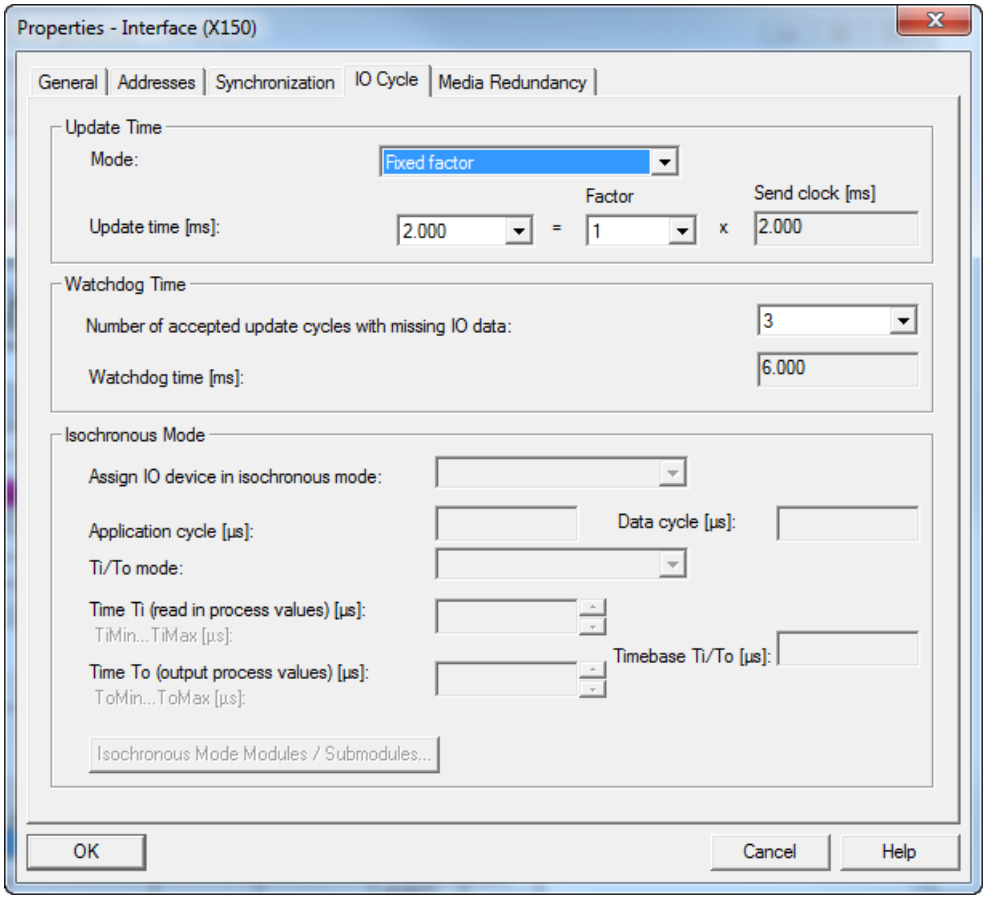


Figure 4-768 Setting isochronous mode

5. Click "OK" to confirm your settings.

Setting DP cycle

1. Double-click the SINAMICS_Integrated.
   The "DP slave properties" window opens.
2. Click the "Isochronous operation" tab and set the "DP cycle" to factor 16. This corresponds to a DP cycle clock of 2 ms.
3. Confirm your entry with "OK".

Interconnecting ports

1. Interconnect the ports by double-clicking on port 1 of the PNxIE net in the rack.
   The "Properties - PN-IO - Port 1 (R0/S2/X1400 P1 R)" window opens.
2. Click the "Topology" tab.

3. At "Partner", select the required port;
   In our example, this is "SIMOTION D\(1)pn1a\Port 1 (X1400 P1)."

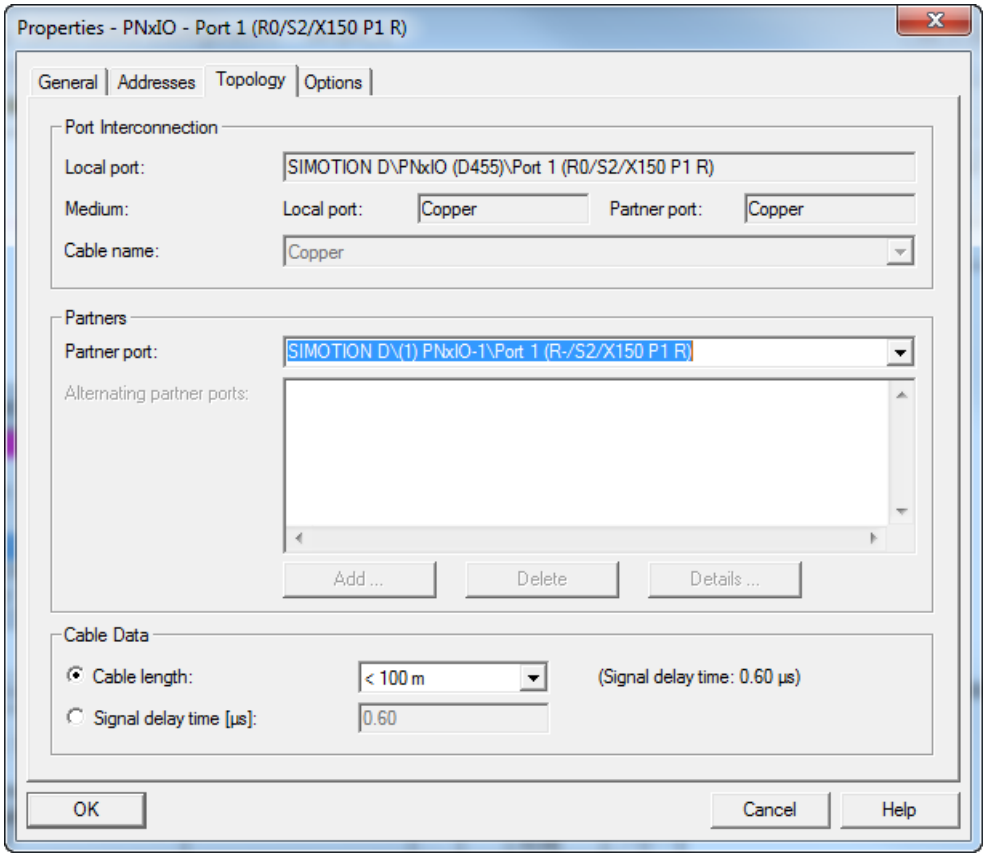


Figure 4-769 Interconnecting ports

4. Confirm your entry with "OK".
5. Click the "Save" button.
6. Switch to SIMOTION SCOUT and save the "master object" project.
   This concludes configuration of the IO controller communication in HW Config. The next step involves creating proxy objects in SIMOTION SCOUT, see section Creating and configuring "Synchronous axis" project on SIMOTION D455-2 (Page 2800).

Proxy objects

Proxy object types

Proxy object types

There are two different types of proxy objects:

- **External master value** (**ExternalMasterType**): Proxy object for an external master value
  A proxy object for an external master value can only be created under a synchronous object, i.e. interconnected with it.
- **External synchronous operation** (**ExternalFollowingObjectType**): Proxy object for an external synchronous operation
  A proxy object for external synchronous operation can be created under the following technology object types, i.e. interconnected with them.
  - External encoder
  - Synchronous axis
  - Positioning axis
  - Path axis

See also

Creating proxy objects (Page 2814)

Configuring proxy objects with SIMOTION scripting (Page 2818)

Creating proxy objects

In SIMOTION SCOUT, you now need to create the proxy objects. This section uses an example to describe the process involved in this.

Creating TOs with a master value for external synchronous operation

1. In the "master object" project, create a virtual position axis using the axis wizard and call it "master_axis_vir".
2. Select the axis in the project navigator, followed by "Expert > Insert external synchronous operation" (in the shortcut menu).
   An object called "Ext_sync_operation" is created below the selected axis and interconnected with the "Master_axis_vir."

3. In the working area, enter the input and output addresses configured in HW Config. 256 is set for the address in this example.

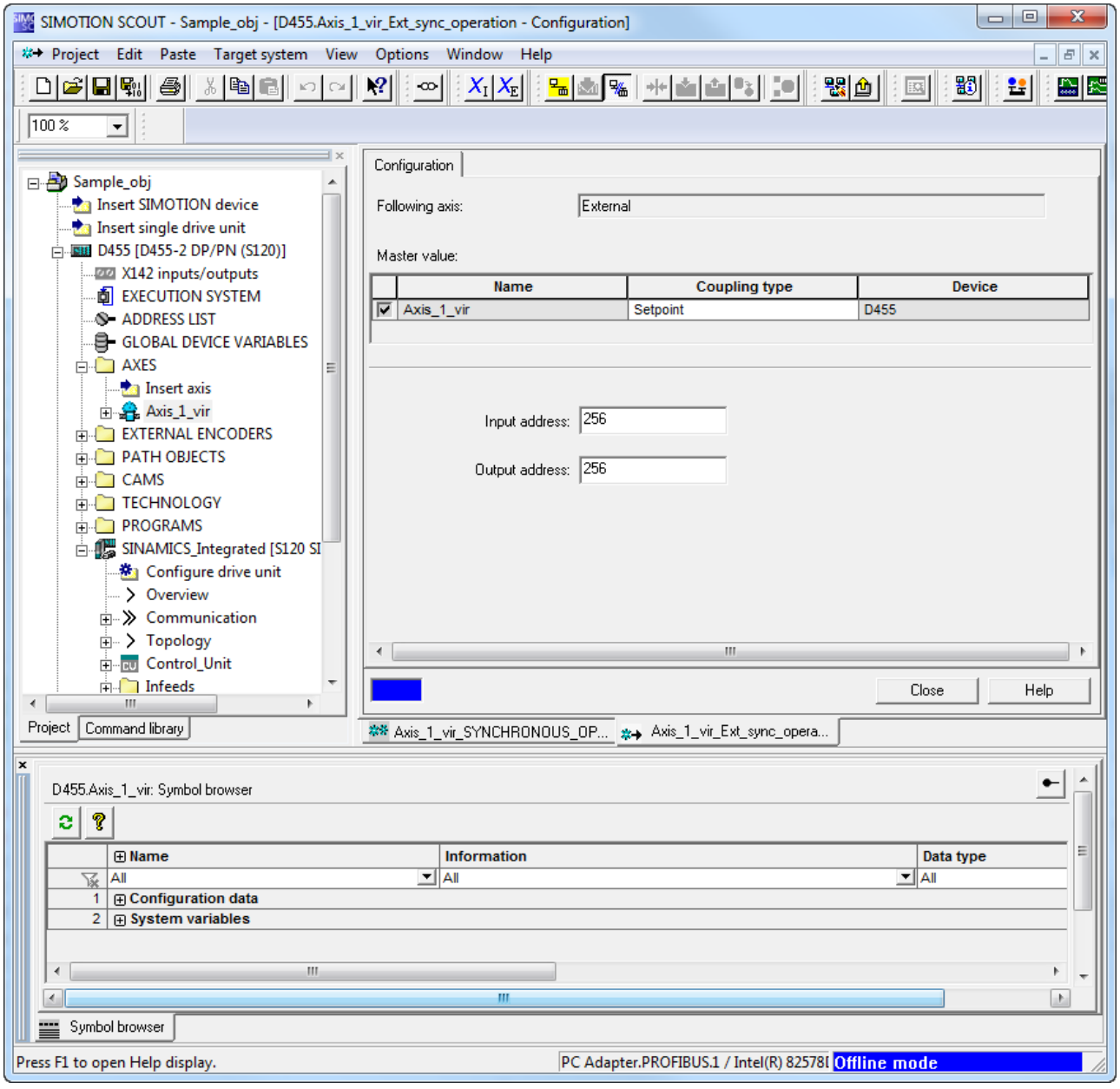


Figure 4-770 Entering input and output addresses

4. Save and close the project.

This concludes configuration of the "Ext_sync_operation" on the master object. The next step involves creating the "Ext_master_value" in the "synchronous axis" project.

Creating the synchronous axis TO with an external master value

1. Open the "synchronous axis" project.
2. Create a synchronous axis and call it "synchronous_axis_slave".

3. An "Ext_master_value" object is created below the selected synchronous object and interconnected with the synchronous object. In the project navigator, select the "synchronous_axis_slave_SYNCHRONOUS_OPERATION" synchronous object below the synchronous axis, followed by "Expert > Insert external master value" (in the shortcut menu). The "Ext_master_value" window opens in the working area.

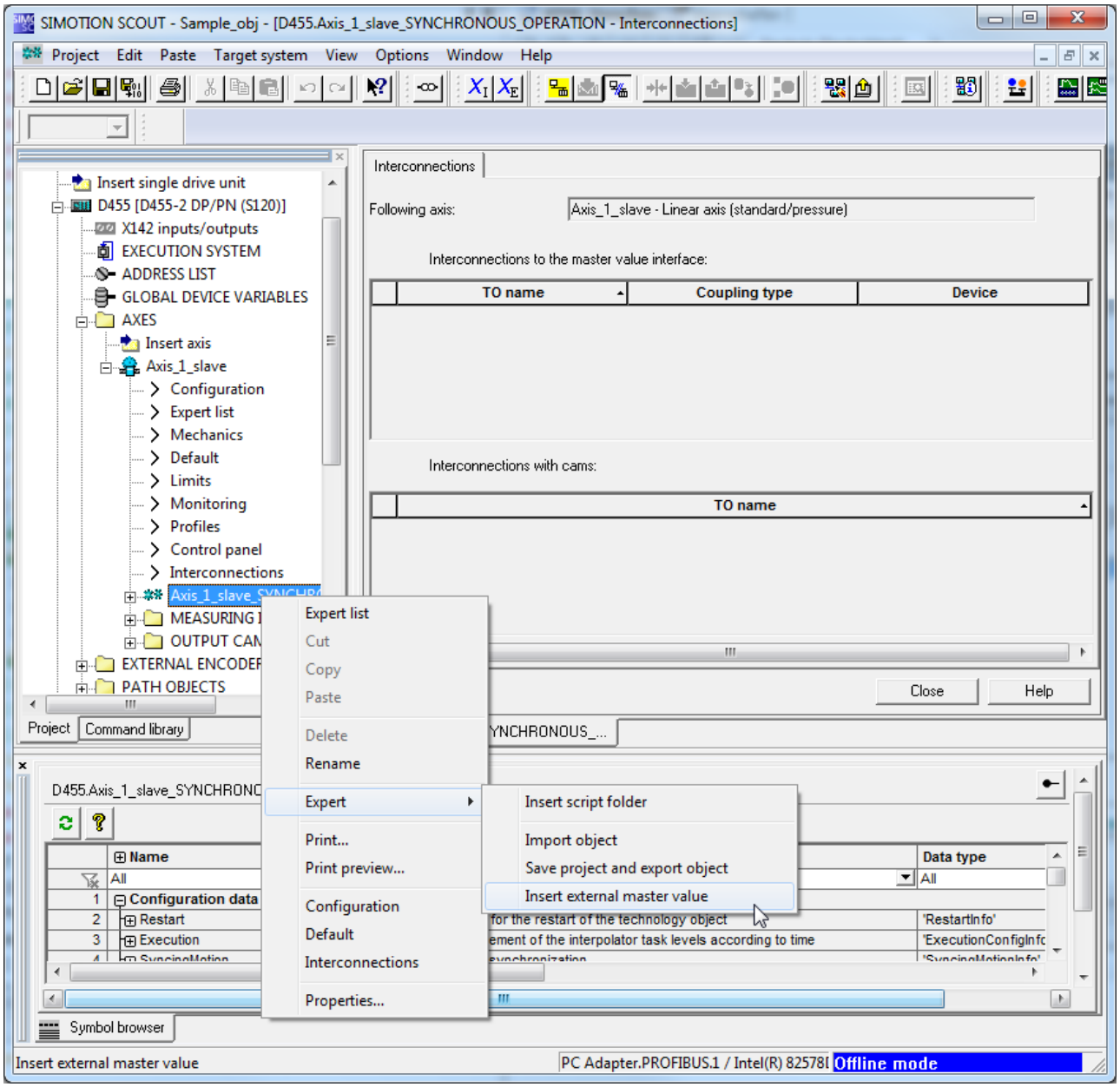


Figure 4-771 Proxy technology object - Setting the external master value

4. In the "Ext_master_value" window within the working area, enter the input and output addresses that you set as input and output addresses in HW Config. 256 is set for the address in this example.

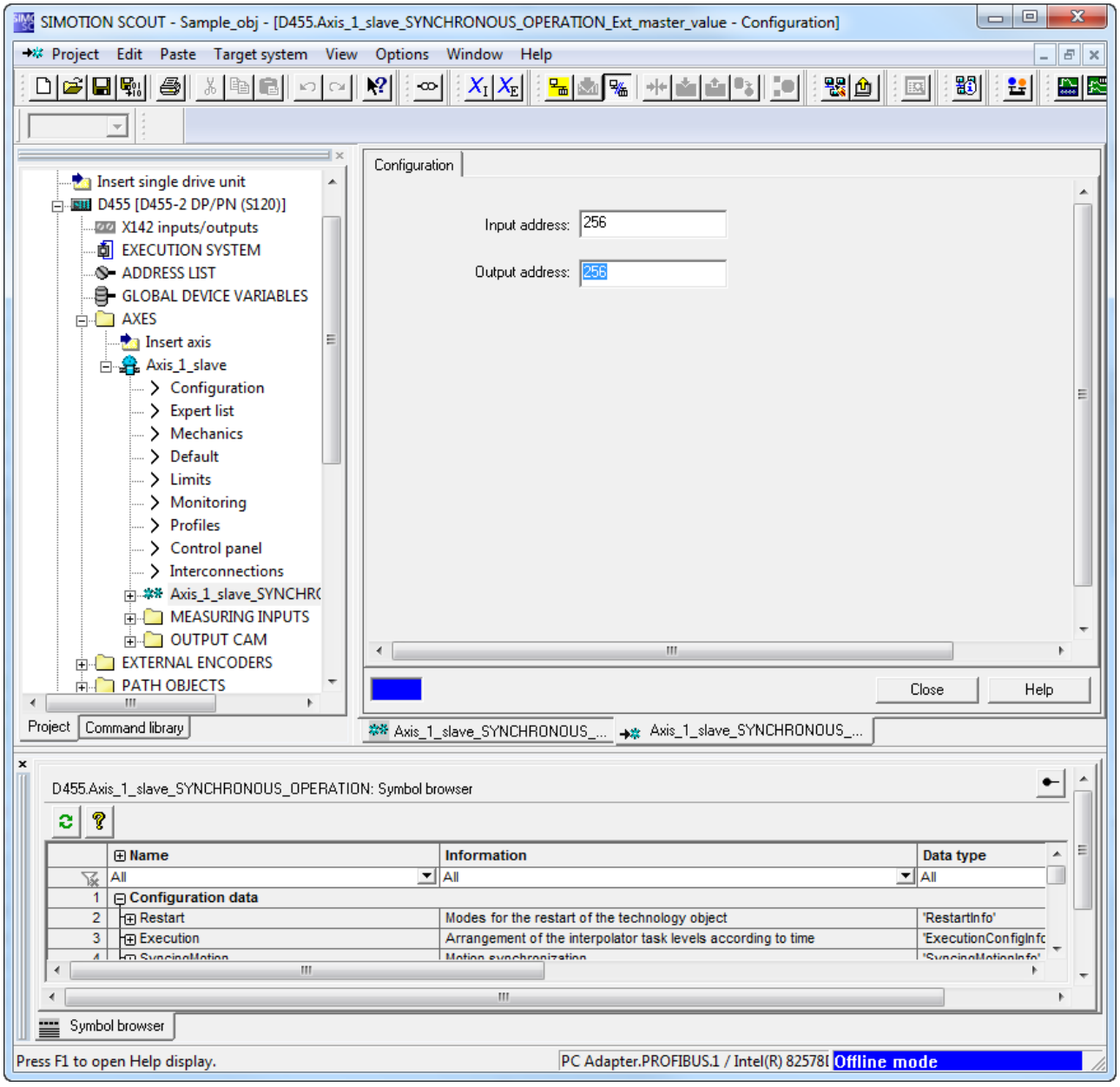


Figure 4-772 Entering input and output addresses on the external master value

This concludes configuration of the communication for cross-project distributed synchronous operation. The steps that follow this are the same as those involved in configuring a distributed synchronous operation, see section Distributed Synchronous Operation Configuration (Page 2777).

Configuring proxy objects with SIMOTION scripting

The logical input and output addresses for the proxy object can be accessed with SIMOTION Scripting. Only the offline data of the object can be accessed. Further information can be found in the online help and on the Utilities & Applications DVD.

Table 4-320 Configuration data for access via scripting

DriverInfo		
	logAddressIn	Base address of the 12-word input data from HW Config
	logAddressOut	Base address of the 12-word output data from HW Config

Changing the logical input and output addresses in the SIMOTION project

An example is described below. The logical input address is changed to address 257 and the output address to address 258.

Note

If the logical input and output addresses are changed with SIMOTION Scripting within the SIMOTION SCOUT project, the project will no longer be consistent, as there is no automatic alignment with HW Config. This means that you must automate the changes to the input and output addresses in HW Config, for example via SIMATIC STEP 7 Scripting.

Proceed as follows to change the logical input and output addresses with SIMOTION Scripting:

1. In SIMOTION SCOUT, create a folder called SCRIPTS below the project. Select the name of the project, open the shortcut menu and click on **Insert script folder**.
2. Create a script.
3. Enter the following text:
   - `PROJ.TOs ("Ext_sync_operation").Symbols("DriverInfo.LogAddressIn") = 257`
   - `PROJ.TOs ("Ext_sync_operation").Symbols("DriverInfo.LogAddressIn") = 258`

4. Save the project.
5. Execute the script by selecting it and clicking on **Accept and execute** in the context menu. The logical input and output addresses are then changed.

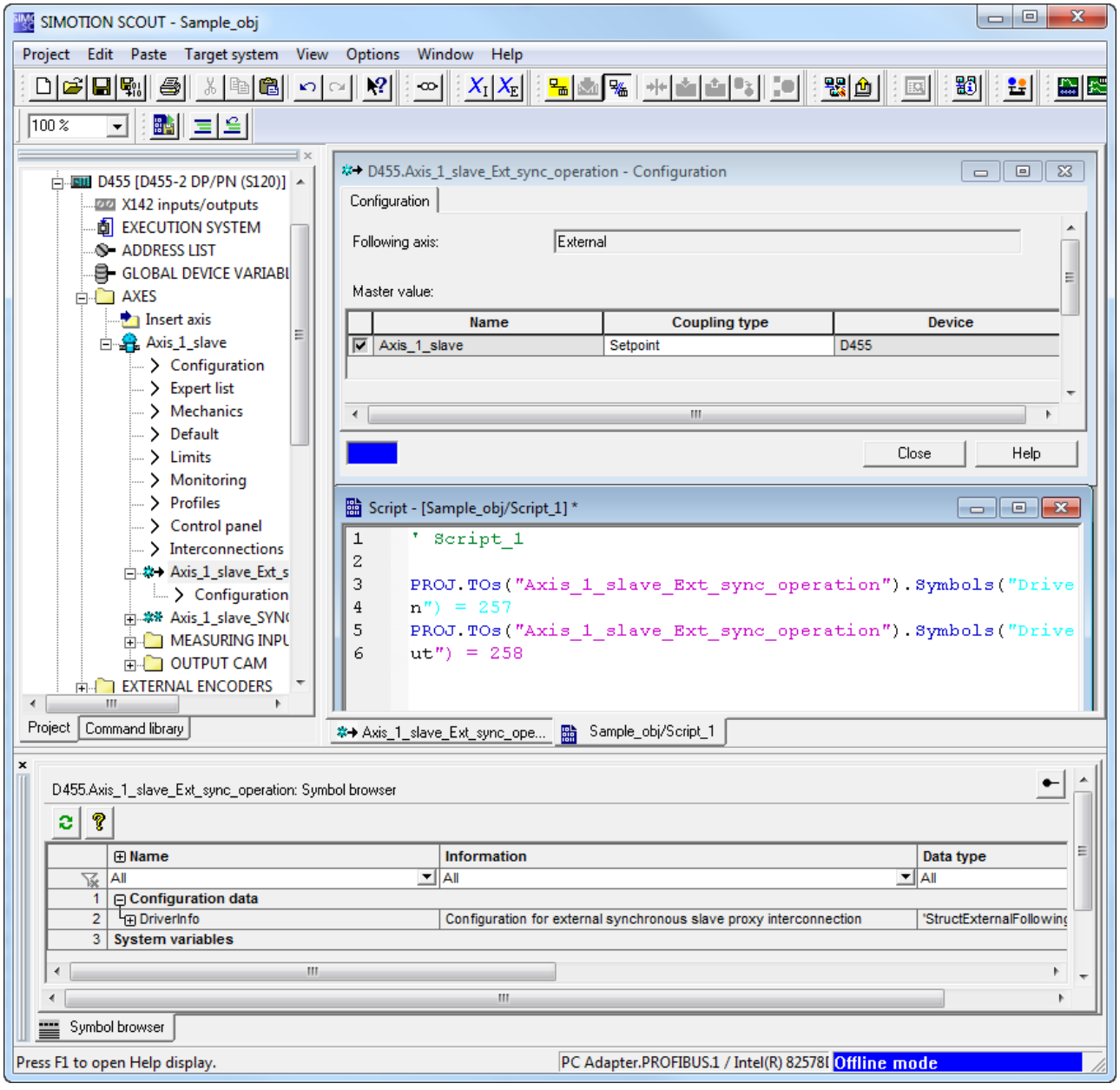


Figure 4-773 SIMOTION Scripting - example

Interconnection possibilities

In general, proxy objects can be interconnected with a maximum of one technology object. The following figure shows the interconnection options.

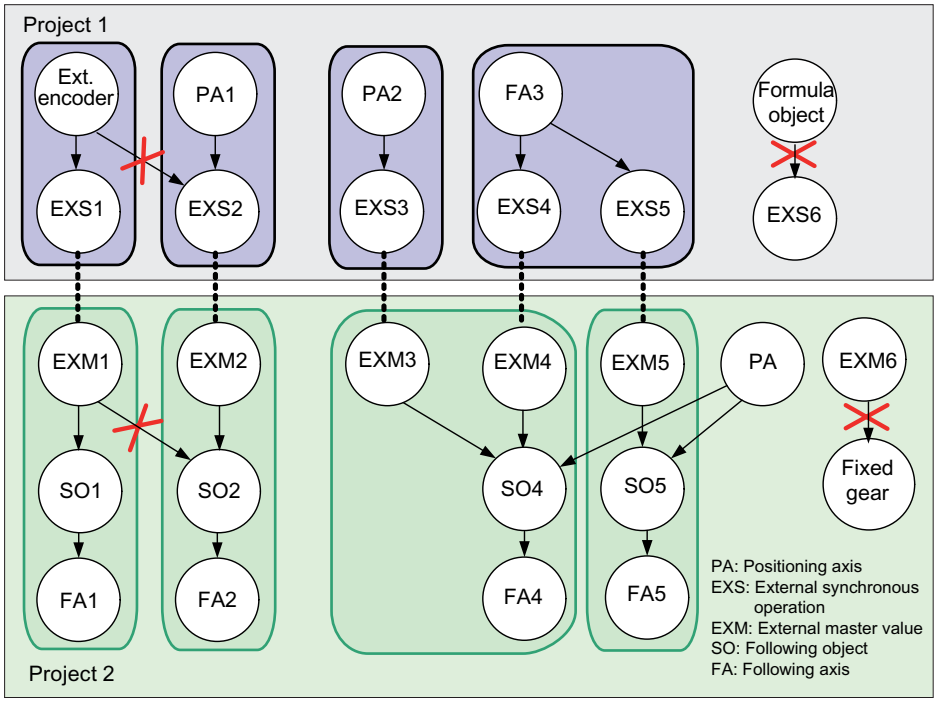


Figure 4-774 Interconnection options for proxy objects

In order to interconnect multiple synchronous objects to one external master value, you must create an additional synchronous operation with a virtual axis, to which the external master value proxy object will be assigned. In this case, the additional virtual axis acts as a master value source for multiple synchronous objects.

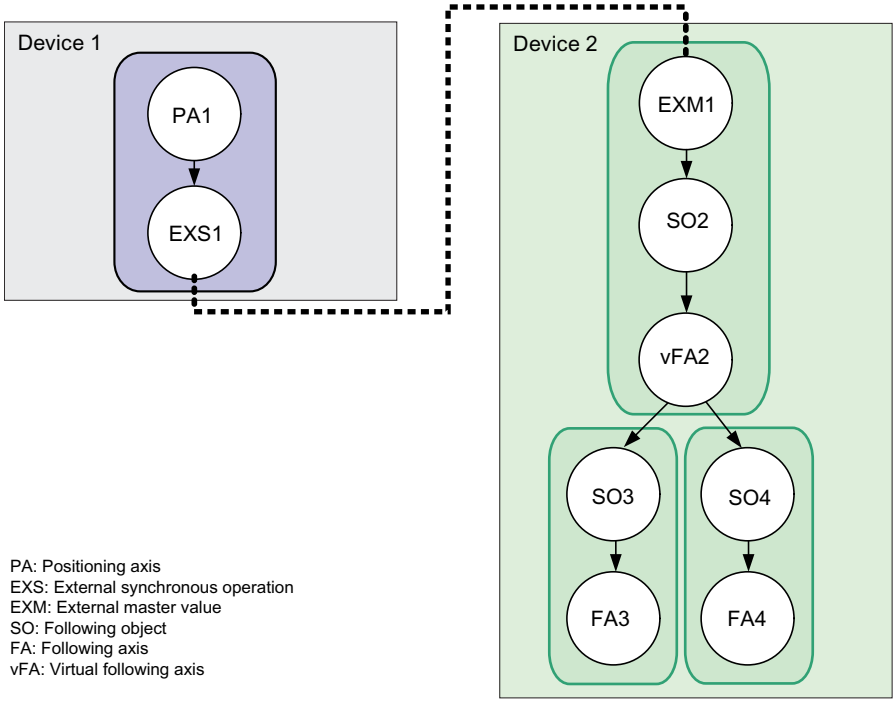


Figure 4-775 Interconnection of multiple synchronous objects to an external master value

The same also applies when different master objects can alternatively be the master value for an external synchronous operation. In this case, an additional synchronous object with a virtual axis must be provided to be interconnected with the master object. The external synchronous operation is assigned to the virtual axis.

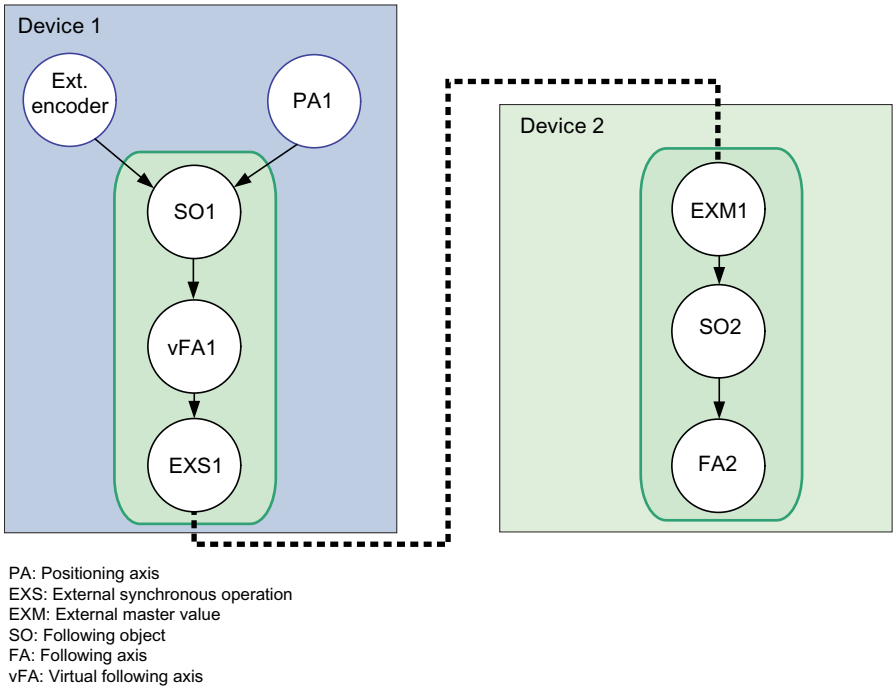


Figure 4-776 Interconnection of multiple master objects to an external synchronous operation

Synchronizing the interface

See Synchronizing the interfaces (Page 2784).

Switching over to an external master value source

If more than one master value is assigned to a synchronized axis, the master value source can be selected and switched over on the synchronous object using the **_setMaster** command (see **Fundamentals of synchronous operation**, "Switching over the master value source"). If you want to switch to an external master value source, you must specify the name of the external master value proxy object in the **_setMaster** command.

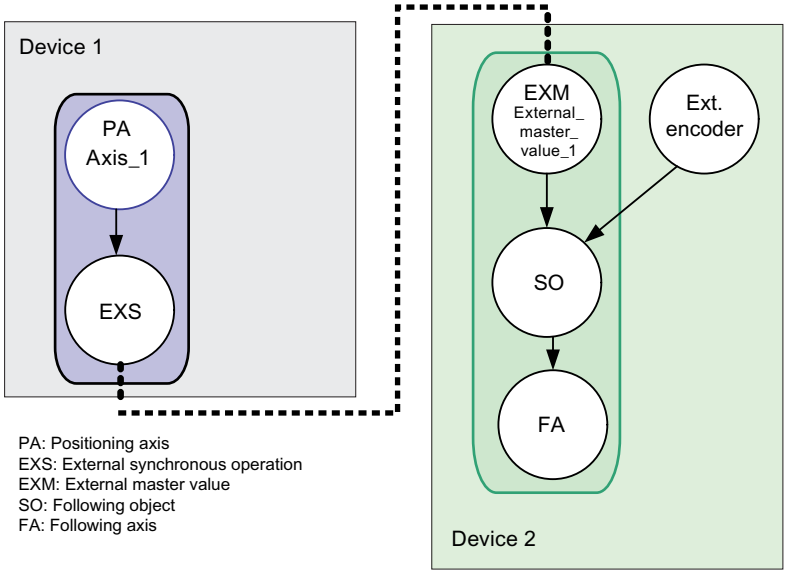


Figure 4-777 Synchronous operation with external and local master value source

This figure shows an example with interconnection to a local and external master value source. In order to switch to the master value of **Axis_1** with the **_setMaster** command, the name of the **Ext_master_value_1** proxy object must be specified in the command.

4.7.3.6 Configuration of distributed synchronous operation with different RT versions

Overview of distributed synchronous operation with different RT versions

It is useful to configure a distributed synchronous operation with different RT versions when existing plants/machines are to be extended partly, or functions are required that are not supported in the currently used RT version. This applies for an extension by one or more controllers and for the upgrade of an individual SIMOTION controller in a machine with several SIMOTION controllers.

Configuration of distributed synchronous operation with different RT versions across projects

This is possible as of RT version V4.3. For further information, see Configuring distributed synchronous operation across projects (Page 2786).

Configuration of distributed synchronous operation with different RT versions in one project

Prerequisite for the configuration is an engineering system version as of SCOUT V5.1 in conjunction with RT versions as of V4.3.

The following scenarios are considered in more detail:

- Extending an existing project by a SIMOTION controller with a different RT version than that used previously in the project.
- Upgrading a single controller to a different RT version in the project.

Project extension by a SIMOTION controller with a different RT version

To configure an additional controller that has a different RT version for a distributed synchronous operation in an existing project, proceed as follows:

1. Create the new controller with the appropriate RT version in a project.
2. Configure the controller as IRT slave (master axis available in the project) or as IRT master (slave axes available in the project) and integrate it in the PROFINET topology. Further information can be found in Section Distributed Synchronous Operation Configuration (Page 2777).
3. Create an external synchronous operation on the master axis. Further information can be found in Section Configuring distributed synchronous operation across projects (Page 2786).
4. Create an external master value on the synchronous object on the slave axis.
5. Create the input and output addresses with 24-byte length in the controller-controller direct data exchange. More detailed information can be found in the Communication System Manual, *in Section Configuring direct data exchange (data exchange broadcast) between IO controllers*.
6. Assign input and output addresses on the external synchronous operation (master) and external master value (slave) from item 5. corresponding to the start address in the controller-controller direct data exchange. Note that changes to the addresses in the controller-controller direct data exchange must also be made in the external synchronous operation / external master value.
7. Optionally: Perform the PROFIsafe configuration on the new controller (F-proxy). Note that when using the controller-controller direct data exchange with IRT, the controllers must not be used on the same interfaces as IRT I device. Only RT I device couplings are possible. Further information on F-proxy configuration can be found in the Communication System Manual.

Upgrading a single controller to a different RT version in the project

To upgrade a single controller to a different RT version in the project, proceed as follows:

1. Upgrade the controller in the existing project that requires functions of a later RT version.
2. Remove the existing distributed synchronous operation configuration on the controller.
3. Use the proxy TOs to implement the distributed synchronous operation. If the master axis is on the upgraded CPU, an external synchronous operation must be created. If the slave axis/ axes are on the CPU, create an external master value. Further information can be found in Section Configuring distributed synchronous operation across projects (Page 2786).

4. Depending on which axis/axes are on the upgraded CPU, the counterpart of the proxy TO (external synchronous operation / external master value) must be created on the other controllers.
5. Create the input and output addresses with 24-byte length in the controller-controller direct data exchange. More detailed information can be found in the Communication System Manual, *in Section Configuring direct data exchange (data exchange broadcast) between IO controllers*.
6. Assign input and output addresses on the external synchronous operation (master) and external master value (slave) from item 5. corresponding to the start address in the controller-controller direct data exchange. Note that changes to the addresses in the controller-controller direct data exchange must also be made in the external synchronous operation / external master value.
7. Adapt the user programming. Change the master values on the synchronous operation commands on the external master value.

4.7.4 Part III - Synchronous operation across multiple cycles

4.7.4.1 Overview of multiple cycle synchronous operation

This part describes the function of **synchronous operation across multiple cycles**. A synchronous operation can be configured such that high-priority axes are calculated in the SERVO or IPO processing cycle clock and low-priority axes in the IPO or IPO_2 processing cycle clock (see under processing cycle clock Real and virtual axes Runtime model and processing). It introduces you to the operating principle and technological boundary conditions of synchronous operation with the master object and following axis in different interpolator cycle clocks (IPO, IPO_2, SERVO). You are shown how to create and configure a synchronous operation in different interpolator cycle clocks.

Function overview

The **synchronous operation across multiple cycles** functionality allows you to operate a master value source and a synchronous axis in different IPO cycle clocks (IPO, IPO_2 and SERVO).

Example:

- Axis_1 is the master axis and is assigned to the IPO task.
- Axis_2 is the following axis and is assigned to the IPO_2 task.

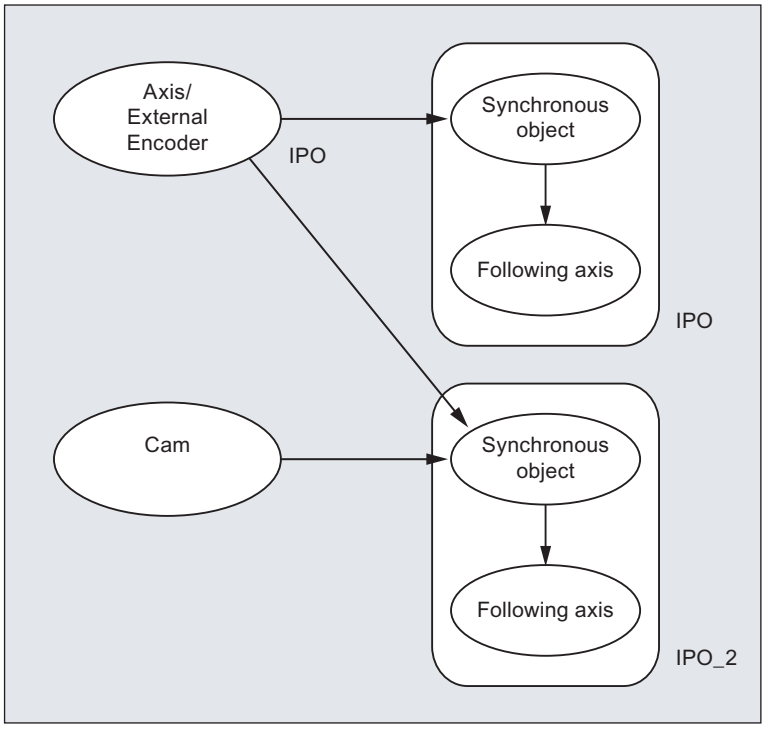


Figure 4-778 Example configuration of synchronous operation between IPO and IPO_2

Application

The **synchronous operation across multiple cycles** function enables the interpolator to be placed in a cyclical system task with a greater cycle time for axes that do not require a high time resolution for calculating the reference value. This reduces the required processor performance.

In this way, it is possible to:

- Operate coupled axes on one device in different IPO cycle clocks
- Operate coupled axes on different devices in different IPO cycle clocks (**synchronous operation across multiple cycles** can be combined with **distributed synchronous operation**).

Operating principle/Compensations

The phase error, which results from processing in a succession of different cycles, can be compensated for in the same way as with distributed synchronous operation:

- Compensation on the master value side by means of setpoint output delay
- Compensation on the slave value side by means of master value extrapolation

For details, see section Running synchronous operation across multiple cycles (Page 2827).

4.7.4.2 Boundary conditions

Objects with synchronous operation across multiple cycles cannot be created arbitrarily, but rather must satisfy the following rules:

- In a synchronous operation interconnection of multiple axes, more than one transition of axes is allowed in different IPO cycle clocks. Several compensations are thus required. These can be set independently at the transitions. The total compensation is determined by the system. The effective delay and offset are indicated at each axis and following object.

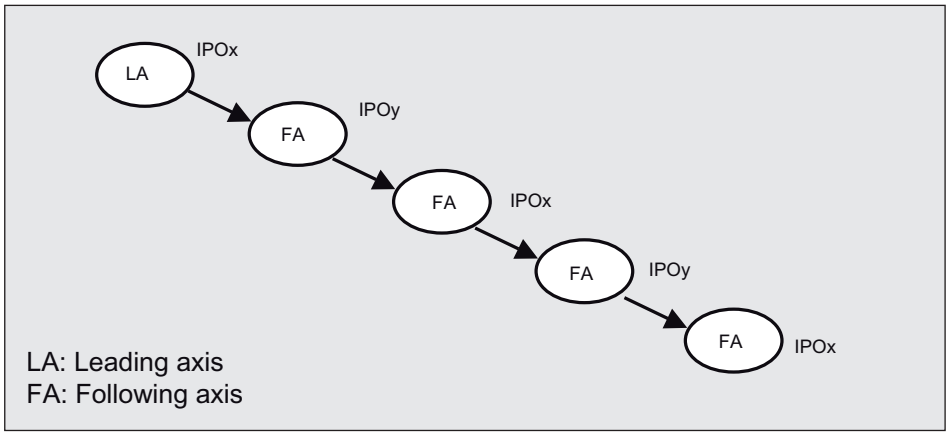


Figure 4-779 Inappropriate configuration with multiple change

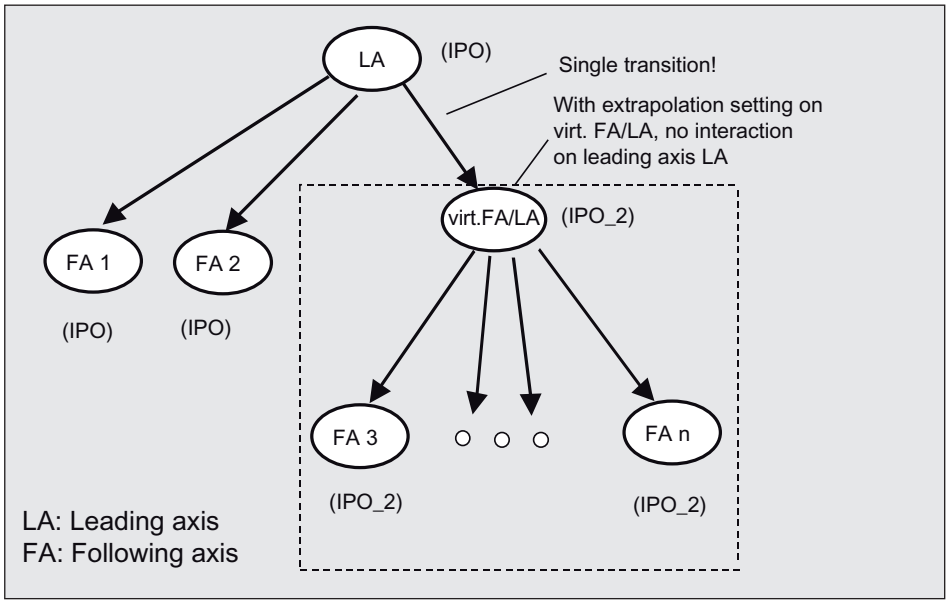


Figure 4-780 Configuration example with axis grouping

Since the calculation of compensation values needs processor resources, it is recommended to keep the number of transitions low and to form axis groups.

- Distributed synchronous operation and synchronous operation across multiple cycles can be combined.
  Synchronous operation across multiple cycles is permitted both in the master system as well as in the slave system.
  For further information, see Rules for the communication/topology for distribution using PROFIBUS. (Page 2760)
- A distributed synchronous operation and an IPOx - IPOy transition is only possible between a leading axis (Axis TO or External Encoder TO) and a synchronous object.
  - A Fixed Gear TO, Addition Object TO, Formula Object TO, Closed-loop controller TO cannot be at a transition, neither on the master value side nor on the slave value side.
  - These TOs may only be located in an effective sequence without transitions between distributed synchronous operation or IPO-IPO 2.
- Recursive interconnections which also contain IPOx - IPOy transitions are not prevented.
  Please note, however, that:
  When switching over during motion, the compensations will lead to a jump or a temporarily constant position setpoint, and a loss in velocity as a result. (Although the monitoring functions will remain active.)
  Therefore, the IPOx- IPOy transitions in the recursive synchronous operation interconnection should be switched *during standstill only*.

4.7.4.3 Running synchronous operation across multiple cycles

With synchronous operation across multiple cycles, an offset between the master value source and following axis results from the calculation of the synchronous operation in different cycles.

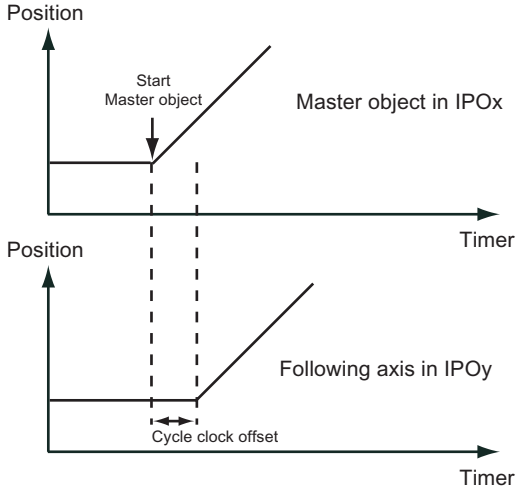


Figure 4-781 Schematic representation of the clock cycle offset due to different interpolator clock cycles

Compensation

The offset can be compensated for in the same way as for distributed synchronous operation:

- Compensation on the master value side by means of setpoint output delay in the IPO cycle clock that provides the master value for the distributed synchronous operation (master CPU).
- Compensation on the slave value side by means of master value extrapolation in the IPO cycle clock containing the slave objects (slave CPU).

The compensations are set and displayed via the system variable **distributedMotion**.

- The output delay is displayed on the leading axis.
- The master value delay is displayed on the synchronous object.
- The cycle clock offset is displayed on the synchronous object.

The details in Compensations for distributed synchronous operation apply accordingly.

Information on compensations

- Compensations can be set differently on different master objects.
- If "Extrapolation" compensation mode is selected, the output delay on the master is zero. In this way, there is no effect on other higher-level master values and their compensation calculation. This allows you to form axis groups.
- It is not possible to specify interpolation on the following axis without master value delay on the leading axis.
- With the "No compensation" setting, the master values are taken over as the exist.
  - Therefore a transition from the slower IPO to the faster IPO is possible but not technically practical.
  - A transition from the faster IPO to the slower IPO is possible, but is associated with a phase offset.

Life-sign monitoring

The life-sign monitoring is also active for synchronous operation across multiple cycles and takes effect, for example, with level overflows.

The details in Operating axes with distributed synchronous operation apply accordingly.

Note

If commands for the master value and the slave value are programmed in the same task, it must be ensured that the commands are effective in the respective processing cycle clock.

If the master value is in a slower execution level than the slave value (e.g. IPO_2 to IPO), it must be noted, for example, that a **_redefinePosition** to the master value for the slave value only becomes effective at the IPO_2 cycle limit. This must be taken into account before issuing an immediately following synchronize command.

4.7.4.4 Creating synchronous operation across multiple cycles in SCOUT

This section describes how to create and configure devices and objects with synchronous operation across multiple cycles.

Master object

1. Insert the leading axis or external encoder and configure it.
2. If required, insert cams and configure these.

Following axis

1. Create one or more following axes.
   Make sure that **Synchronous operation** is activated as the technology on the following axes.
2. Configure the synchronous object.
   Interconnect the synchronous object with the master object.

Processing cycle clock

The processing cycle clock is defined in the **executionConfigInfo.executionLevel** configuration data element of the axis and the synchronous object. The setting can also be made on the axis via the configuration mask.

- Set the desired IPO cycle clock for each object.
  When doing so, select the **IPO** or **IPO2** setting for the **Processing cycle clock**.

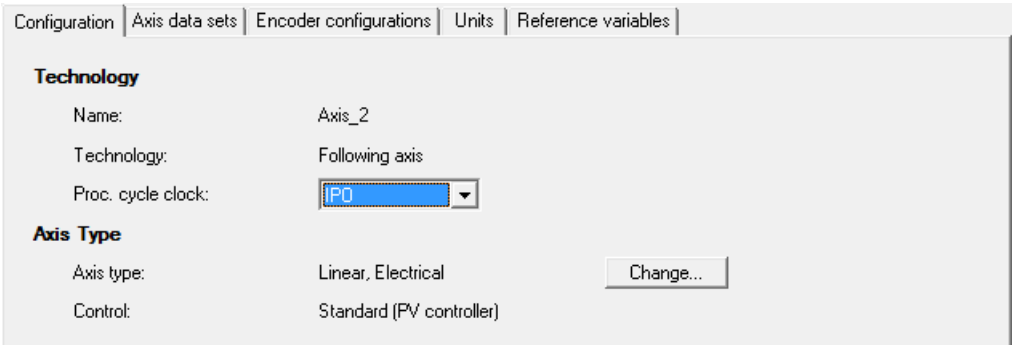


Figure 4-782 Defining the execution cycle in the axis configuration

Configuration

1. In the shortcut menu of the synchronous object, select **Expert > Expert list**; see **Motion Control Basic Functions**, "Expert list."
2. Specify the required configuration (see Operation of synchronous operation across multiple cycles (Page 2827) and Compensations of the distributed synchronous operation (Page 2767)).

4.7.5 Part IV - Cam

4.7.5.1 Overview of cam

This part describes the function of the **Cam** technology object. It introduces you to the creation and definition of cams, as well as providing information on supplementary conditions and the operating characteristics of cams.

Function overview

The **Cam technology object** can be used to define a *transmission function* and apply it with other technology objects. A cam describes the dependency of an output variable on an input variable.

- An *input variable* could be the actual position of a master axis, a virtual master value source, or the time.
- An *output variable* could be used as the set position of a following axis, the setpoint profile, or the pressure/force profile.

The TO cam is a *stand-alone* technology object, which can be interconnected with other technology objects.

Application

At present, a **TO cam** can be utilized with the following objects:

- With a **synchronous object** as a transmission function
- With a **TO axis**, e.g.
  - As a velocity, position, or pressure profile
  - As a valve characteristic curve in the hydraulic axis setting

Definition

A **TO cam** describes a function y = f(x) in sections. The sections can be defined by means of *interpolation points* or *segments* (using polynomials). Cams are *dimensionless*. No physical units are used to define them.

Creation and storage

Cams can be created by means of the SIMOTION parameter assignment tool in the engineering system (**CamEdit**) or using the **CamTool** add-on. Cam objects *cannot* be created by the user program.
In order to define a cam in the **user program**, the object must have been created previously in SIMOTION SCOUT.

Cams are stored *by device* and can be assigned to each applicable object of this device. Multi-device cams are not possible.

Exporting/importing cams

With CamEdit, you can export or import the geometry data of the cam.

Scaling and offset

Cams are scalable either in subranges or overall using commands from the user program, even if you have defined the cam using the parameter assignment system. For further information, see "Scaling and offset".

Interpolation

If a curve is defined using segments, gaps in the definition range can be filled by interpolation. If the cam is defined by interpolation points, the characteristic is interpolated. You can select from a variety of interpolation methods. For additional information, see "Interpolation".

Reset

Resetting a cam causes the contents of the cam to be reset. The reset command deletes previously defined interpolation points or segments. The reset command sets the scaling factor to 1 and the offset to 0. If the cam is interpolated, it must be reset before definitions are made in the user program.

Access protection

Only one write action can be performed on the cam at any one time. Any number of read actions can be performed on the cam at any one time. Several write or read actions cannot be performed at the same time.

4.7.5.2 Fundamentals of Cam

Definition

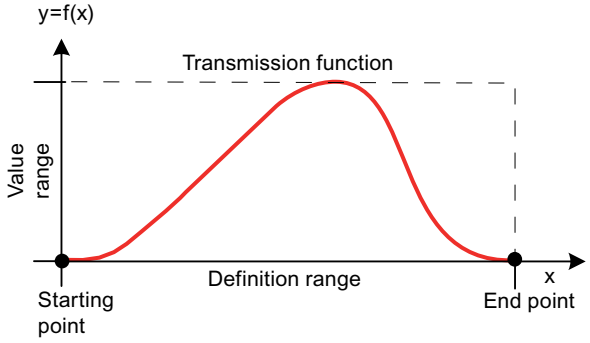


Figure 4-783 Definition of the cam

A cam is defined by the following:

- Definition range
- Starting point and end point of the function in the definition range
- Transmission function
- The value range,
  which is generated from the transmission function

SIMOTION provides the following two options for defining cams. As an option, on the cam in the **interpolatorSettings.dataMode** configuration data element you can set whether only interpolation points, only segments, or both are to be used:

- Definition by means of interpolation points and segments (default).
  Setting **interpolatorSettings.dataMode=SEGMENTS_AND_POINTS**
- Definition by means of segments.
  Setting **interpolatorSettings.dataMode=SEGMENTS_ONLY**
- Definition by means of interpolation points.
  Setting **interpolatorSettings.dataMode=POINTS_ONLY**

A cam can be non-normalized or normalized (with a unit interval of 0.0 ... 1.0) (see Normalization (Page 2833)).

Definition based on interpolation points

Interpolation points are represented in the form **P = P(x, y)** in the interpolation point tables. The order by which the value pairs are entered is irrelevant. They are automatically sorted in the definition range in ascending order. SIMOTION interpolates according to the assigned **interpolation type**.

Definition by means of segments

Individual segments are described in accordance with **VDI 2143**, Motion Laws for Cam Mechanisms. For additional information, see Motion laws in accordance with VDI (Page 2841).

Polynomials with a maximum polynomial degree of 6 and (as an option) a compound trigonometric function are used for this purpose.

Advantages and disadvantages of the definition modes

Table 4-321 Advantages and disadvantages of defining cams by means of interpolation points or segments

	Definition based on interpolation points	Definition by means of segments
advantages	• Simple definition • Any algorithms can be mapped by interpolation points • Curve creation assisted by teach-in • Simple interface to HMI	• Fewer data used for definition • Standard transitions in accordance with VDI ... • Contour is very precise, transitions are continuous
Disadvantages	• Large number of interpolation points required for the exact representation of the contour	• Complex arithmetic required for calculation of coefficients

Definition by means of interpolation points and segments (mixed)

With V4.2 and higher, on the cam in the **interpolatorSettings.dataMode** configuration data element you have the option of setting whether only interpolation points, only segments, or a mixture of both are to be used.

Normalization

When a cam is defined by means of segments, the individual cam segments can be in normal form (normalized to 1), i.e. both the definition range and value range are contained within the interval [0,1].

Alternatively, the segments can also be entered in the real range.

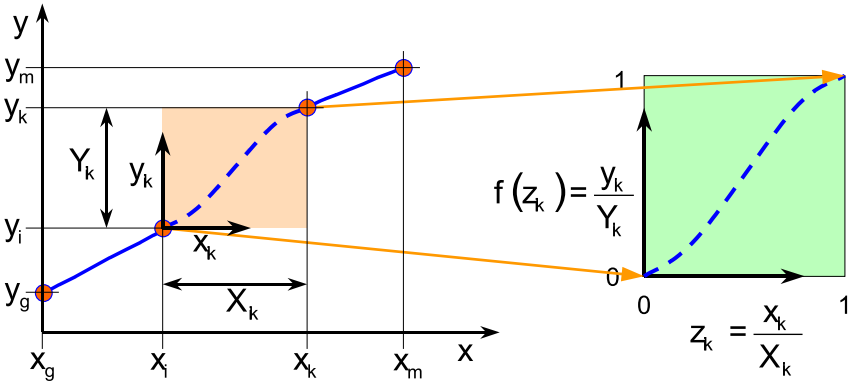


Figure 4-784 Mapping of a real cam segment to the normalized range

A normalization offers the following advantages:

- Motion is clearly defined for similar tasks
- Independent of the real units and value ranges

In addition to the function, the derivatives can also be normalized (normalized transmission function, NTF).

Scaling and offset

The definition range and value range of a cam can be adjusted according to the application, i.e. the function can be offset and stretched/compressed (scaled).

The function value for a scaled and offset cam is generated from the definition using the following formula:

$$y = F_{CAM}\left(\frac{\text{x = Offset of definition range}}{\text{Scaling of definition range}}\right) \bullet \text{Scaling of value range + Offset of value range}$$

Figure 4-785 Formula for scaling and offset of the cam

Scaling

The **_setCamScale** command scales a cam in the value range or definition range. Scaling is applied when the command is successfully issued. Scaling takes place over the complete cam or within a range defined by the starting and end points.

- With *basic scaling*, the entire cam can be scaled and offset.
- With *range scaling*, individual segments of a curve can be scaled and offset.

The zero point of the coordinate axes is used as the scaling point ("pivot point") for basic scaling, whereas the starting point of the specified scaling range is used for range scaling. The starting point of the range scaling can be greater than the end point. In this case, the larger value is the pivot point for scaling (thus the starting point).

The following are possible for the x-axis and y-axis, respectively:

- One complete scaling
- Two range scalings
- One offset

The range scalings can overlap.

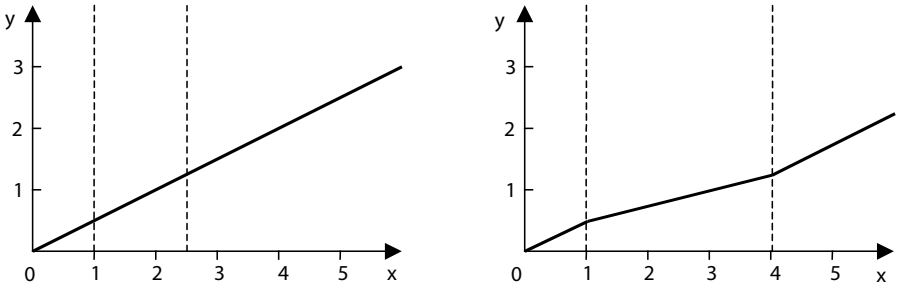


Figure 4-786 Example of scaling of definition range in the range of 1 to 2.5 using a factor of 2

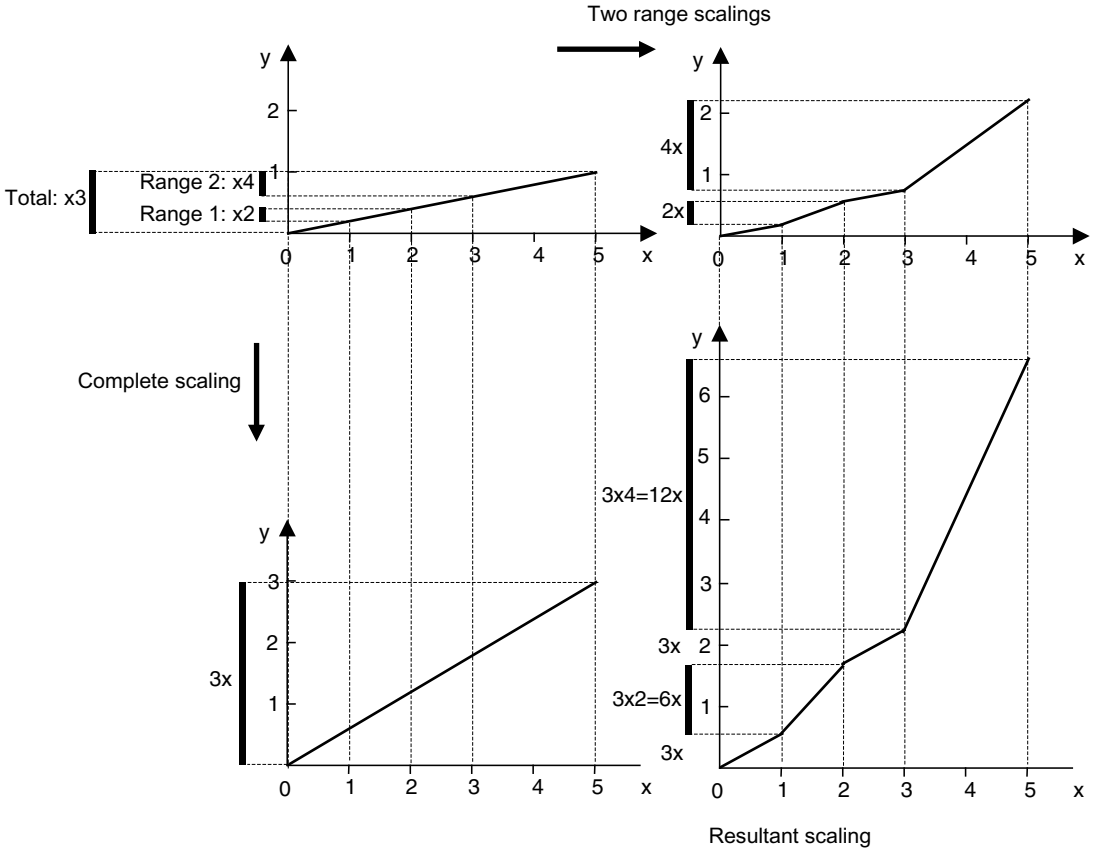


Figure 4-787 Example of two range scalings and a complete scaling in the value range

Scaling can be performed before or after segments and points are inserted or the interpolation is performed. If scaling is performed after interpolation, however, there will be a discontinuity in the first derivative of the cam (even if B-spline or C-spline interpolation is used).

Tip: To prevent this, scaling should begin and end in the dwell ranges.

Offset

The **_setCamOffset** command can be used to separately offset the domain and/or range of a cam.

An offset can be specified as *absolute* or *relative* with respect to the current offset.

- With **ABSOLUTE**, the offset value applies instead of the previous offset value.
- With **RELATIVE**, the offset value is added to the current offset value.

Interpolation

If a curve is defined using segments, gaps in the definition range can be filled by interpolation.

When the cam is defined using **interpolation**, the following checks are performed:

- A plausibility check, i.e., the cam definition is checked (for redundant values within the definition range, for example).
- Missing ranges are added (interpolated).
- Continuity and junction conditions in boundary points are checked.

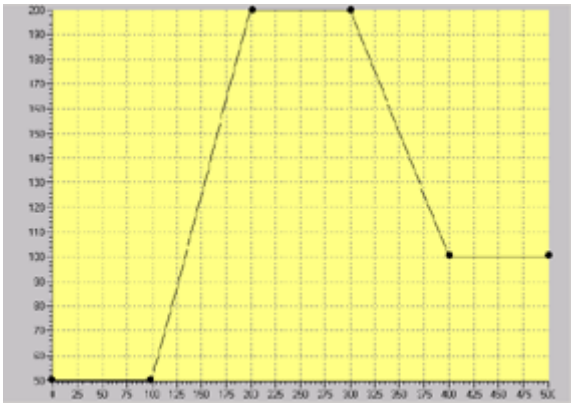
Note

Once interpolation has been performed, new segments or interpolation points can only be inserted after resetting the cam (see Commands for resetting the cam and errors (Page 2850)).

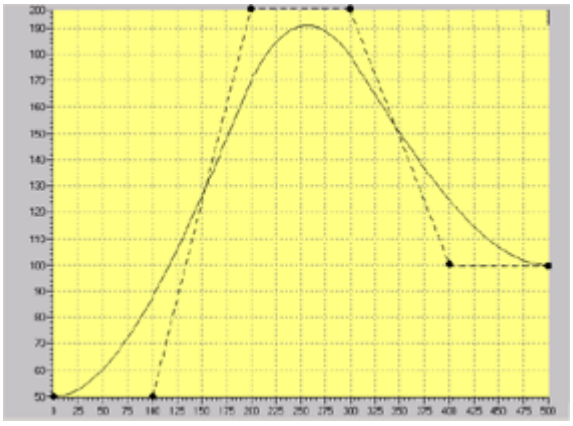
If an attempt is made to insert new segments or interpolation points without first resetting the cam, the attempt is rejected and the return value for the function provides error information. Previously defined interpolation points and segments are deleted when the cam is reset.

Interpolation types

SIMOTION offers the following interpolation types for the **Cam** technology object:

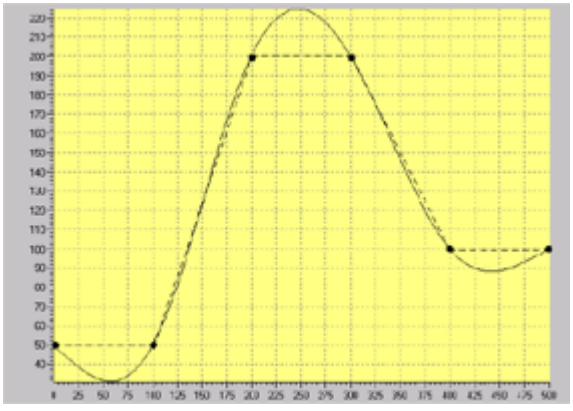
Interpolation	Description	Example
LINEAR	Linear interpolation	

B_SPLINE Approximation using *Bezier splines*, i.e., curve characteristic along the interpolation points



With V4.2 and higher, in the **interpolationSettings.bSplineInterpolation = WITHOUT_APPROXIMATION** configuration data element you have the option of setting whether the 2nd derivation should run continuously.

C_SPLINE Interpolation using *cubic splines*, i.e., curve characteristic through the interpolation points



With V4.2 and higher, in the **interpolationSettings.cSplineInterpolation = ADVANCED** configuration data element you have the option of setting whether the 2nd derivation should run continuously.

Special interpolation settings

- **B spline interpolation**: As an option, you may choose not to set approximation of the B splines in the **interpolationSettings.bSplineInterpolation** configuration data element. This is a good idea if the interpolation points are not at the same distances and at least continuity of the 2nd derivation is required. An accurate display in the ES is not supported in this mode.
- **C spline interpolation**: As an option, you may choose to set a global interpolation in the **interpolationSettings.cSplineInterpolation** configuration data element if problems sometimes arise in the continuity of the 2nd derivation. This results in additional memory and performance requirements.

Continuity check

A function with assigned parameters can be checked for **continuity** in the definition range and value range, and possible points of discontinuity can be corrected. During this process, the points of discontinuity are examined separately for the definition range and value range, and are rated for one of the following corrective actions:

- If the absolute value of the spacing between segments exceeds a maximum value, a correction is made by performing an interpolation between the two segments. This results in insertion of a new segment.

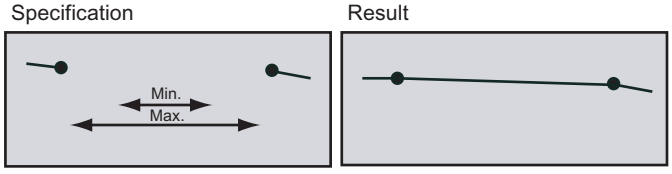


Figure 4-788 Interpolation by insertion a new segment

- If the absolute value of the spacing between segments is greater than the minimum value and less than the maximum value, correction is made by joining the segment end points. The mean value of the spacing of the function is used for the correction. The shape of the segments is affected as a result.

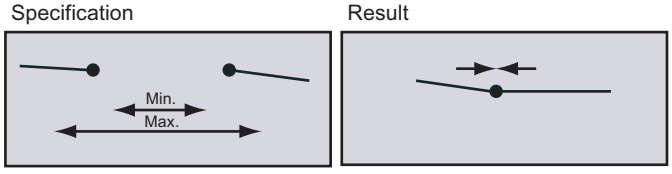


Figure 4-789 Correction by joining segment end points

- If the absolute value of the spacing between segments or interpolation points is less than the minimum value, a correction is not made. The discontinuity point is retained. When this discontinuity point is accessed, the right boundary point is output.

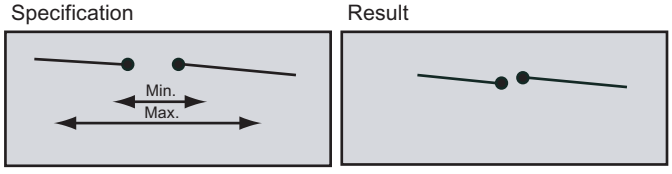


Figure 4-790 Allowing the discontinuity to remain unchanged

The point of discontinuity is corrected according to the evaluation for the definition range and value range.

Table 4-322 Boundary conditions for evaluation in the definition range or value range of the point of discontinuity

Condition	Result
Deviation < minimum	Discontinuity retained
Minimum < deviation < maximum	Join segment end points
Deviation > maximum	Interpolation (new segment)

The correction is controlled (separately for definition range and value range) by specifying the **minimum** and **maximum shape deviation**. This specification can be made on the **_interpolateCam** command for interpolation of the cam.

Depending on the combined evaluation in the definition range and value range, the point of discontinuity is corrected according to the following scheme:

Table 4-323 Combined evaluation for definition range and value range of point of discontinuity

		Correction for definition range		
Discontinuity retained	Join segment end points	Interpolation		
Correction for value range	Retain discontinuity	Retain discontinuity	Join segment end points	New segment
	Join segment end points	Join segment end points	Join segment end points	New segment
	Interpolation	Retain discontinuity	Join segment end points	New segment

- Function continuity can be achieved with **linear interpolation**.
- With **spline interpolation**, continuity is possible in the derivatives.

If the continuity condition cannot be adhered to because of the selected interpolation method or the programmed geometry, a message is provided to that effect.

If an **interpolation boundary point** lies within the programmed geometry, all geometry elements up to the boundary points are rejected. If an interpolation boundary point lies outside the programmed geometry, an end point is extrapolated according to the interpolation method used and taking into account the geometry characteristic.

Continuity at boundary points

When assigning parameters to a cam object, you can make three different settings in the **interpolation** tab.

These settings specify how the runtime system should handle discontinuity at the boundaries of the cam. The appearance of the cam in SIMOTION SCOUT may be different from when it is used later in the runtime system.

- **Non-cyclic**: Not constant at the boundary points
  The runtime system uses the cam as specified, including all discontinuities at the boundaries, even if it is applied cyclically. However, the acceleration limits and moment inertia of the mechanical system / drive are the governing factors.
- **Cyclic absolute**: Position-continuous in the boundary points
  The runtime system converts the cam in such a way that is position-continuous and velocity-continuous at the boundaries during cyclic operation, which can cause changes to occur in the characteristic.
- **Cyclic relative**: Constant velocity in the boundary points
  The runtime system calculates the cam in such a way that is has constant velocity at the boundaries during cyclic operation - within mathematical limits, which can cause changes to occur in the characteristic.

Overlapping segments

In the case of overlapping segments, segment validity can be defined using the options listed below.

- Segments *after* the *segment starting points* are valid
- Segments *up to* the *segment end points* are valid
- Valid segments are determined by the chronological order of insertion.

This behavior is set using the **_resetCam** command.

Inversion

Inverse mapping

For some applications, it is necessary to determine the master value for a defined slave value. The master value can be retrieved using **_getCamLeadingValue**. This inverse mapping is only unique in the case of strictly monotone output functions. In order to supply a master value for output functions that are not strictly monotone, an x-value is specified, and the nearest solution (in both directions) for a y-value is then sought for the specified x-value. If an x-value is not specified, the search will begin from the starting point of the function.

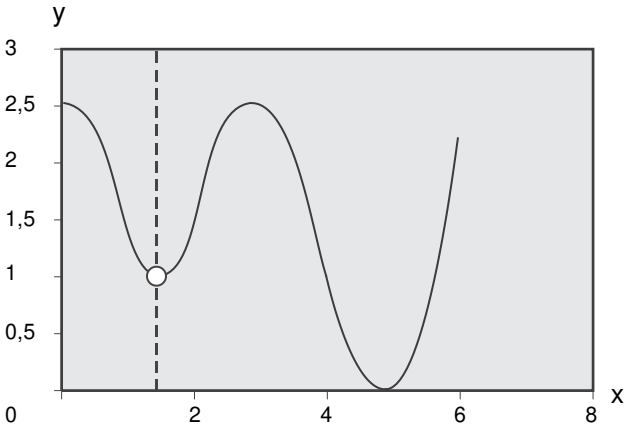


Figure 4-791 Output function

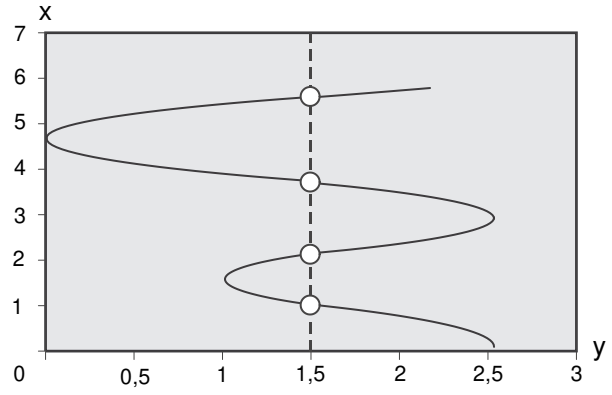


Figure 4-792 Inverse mapping

Inversion during interpolation

The option exists to invert a cam during preparation/interpolation and to store the cam in both the non-inverted and inverted shape (V3.0 and higher). Setting via configuration data **camRepresentation**

Advantage:
Data that has already been inverted can be accessed more quickly when the master value associated with the slave value is read out.

Disadvantage:
More memory is required in the runtime system.

Note

The inverted shape of the cam cannot be represented.

Motion laws in accordance with VDI

The VDI concept of *working ranges and motion transitions* is used to define a cam by means of segments. The **VDI Wizard** can also be used for assistance in the creation of cams.

References

- **VDI Guideline 2143**, 1: Motion Laws for Cam Mechanisms - Basic Theory Düsseldorf: published by VDI-Verlag, 1980
- Volmer, J. (edited.): Mechanism Design - Cam Mechanisms, 2. Ed. Berlin: Published by Technik Verlag, 1989

See also

Motion tasks (Page 2841)

Defining a cam for a motion task using segments (Page 2843)

Motion tasks

The VDI concept differentiates between working ranges and motion transitions.

- *Working ranges* correspond to sequences in a process. The VDI (Association of German Engineers) distinguishes between four different types of working ranges (see below).
- *Motion transitions* are transitions between working ranges, which are not directly relevant to the process but must satisfy certain boundary conditions (such as continuity in velocity and acceleration).

Working ranges in accordance with VDI

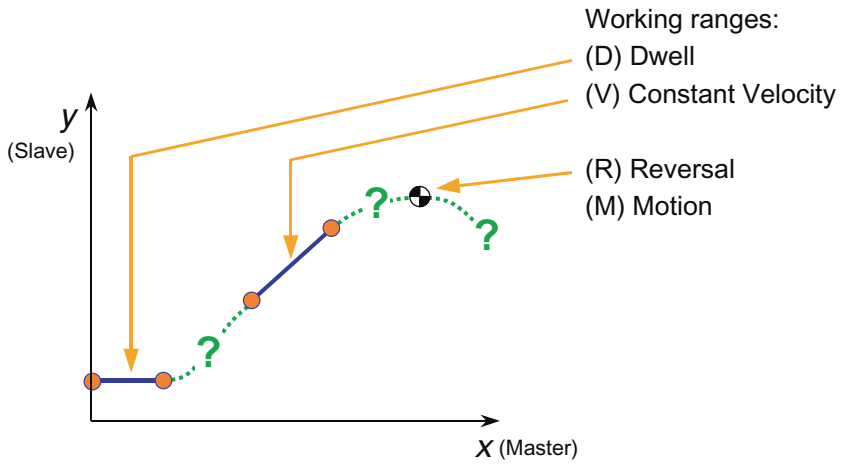


Figure 4-793 Working ranges in accordance with VDI

The VDI concept distinguishes between the following working ranges:

- **R: Dwell** (velocity = 0, acceleration = 0)
- **V: Constant velocity** (velocity <> 0, acceleration = 0)
- **A: Dwell** (velocity = 0, acceleration <> 0)
- **B: Motion** (velocity <> 0, acceleration <> 0)

Example

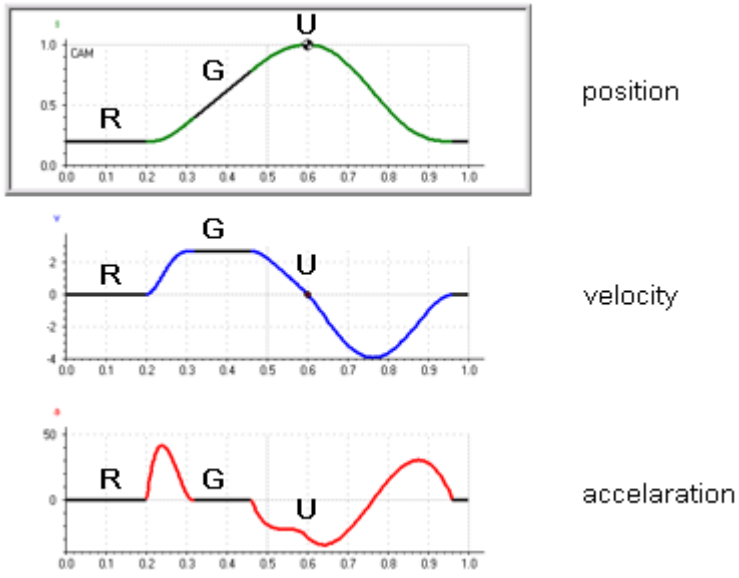


Figure 4-794 Example of a cam with three working ranges

Motion transitions in accordance with VDI

The motion transitions shown in the figure can occur between the individual working ranges.

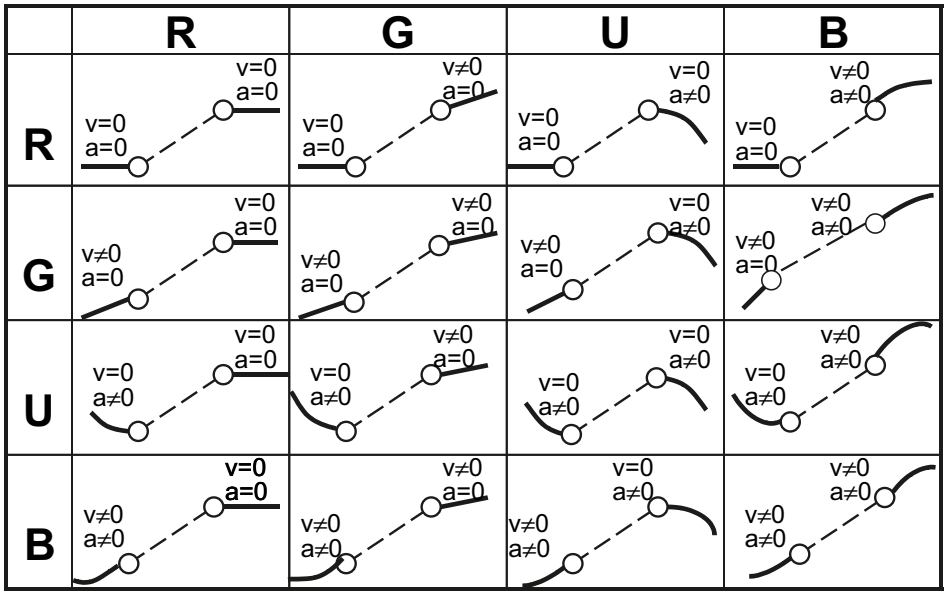


Figure 4-795 Motion transitions in accordance with VDI 2143

Defining a cam for a motion task using segments

Definition of working ranges

The **working ranges** of a motion task are usually specified by the process.

Example:

1. A tool waits on a production line for a piece to pass by (dwell).
2. The tool is synchronized to the work piece and performs an action on the work piece (constant velocity).
3. The tool then returns to the waiting position (reversal).

The process starts over from the beginning.

- In order to implement this sequence, the segments of a cam corresponding to the working ranges must first be created.

Creating a motion transition

Now the "only" things left to define are the **motion transitions** that satisfy certain conditions (e.g., jerk-free motion).

- Start by transforming the motion transition to the normalized range.
  For additional information, see Normalization (Page 2833).
- The **boundary conditions**, i.e., positions, velocities, and accelerations, must then be taken into account at the boundaries between segments.
- In order to apply a polynomial defined in such a way, it must be transformed back into the **real range**.

4.7.5.3 Cam Configuration

Cams can be created with SIMOTION SCOUT or the add-on SIMOTION CamTool add-on. In addition, the curve characteristic can also be defined by a user program during runtime.

You must perform the following tasks when you configure **cams** in SIMOTION SCOUT:

- Create a cam (Page 2844)
- Define the cam. (Page 2845)
- Exporting/importing cams (Page 2845)
- Interconnect the cam.
  Cams are assigned in the respective application. For related information, refer to the descriptions for the corresponding technology objects.

Creating a cam

Follow the steps described below to create a cam:

1. To create a Cam TO in **SCOUT**, double-click **Insert cam** below **CAMS** in the project navigator. You can also copy an existing Cam TO using the clipboard and insert it under another name.
2. Define the cam.

All cams are saved to the **CAMS** folder and are valid for the entire device. The cams can be assigned to all applicable objects of a device (e.g. synchronous objects). This assignment is symbolized in the project navigator, for example, as follows:

- A link to the cam is created below the synchronous object.
- A link to the synchronous object is created below the cam.

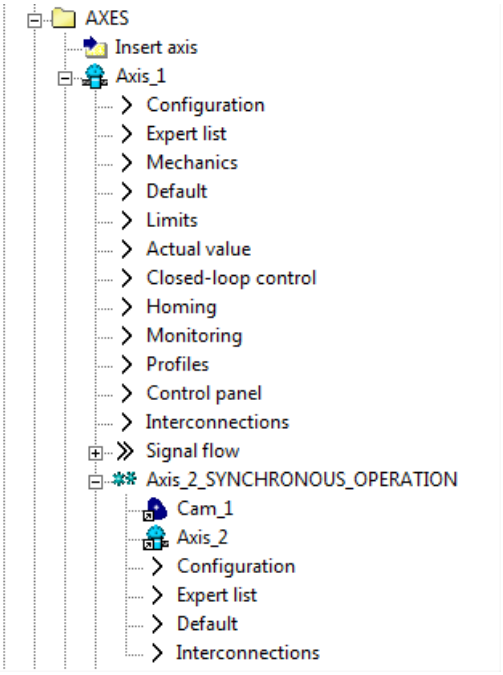


Figure 4-796 Representation of cams in the project navigator

Exporting/importing cams

Description

With CamEdit, you can export or import the geometry data of the cam. For further information, refer to the online help of the "CamEdit cam editor."

Defining and loading cams

Defining with CamEdit

CamEdit can be used to describe cams by means of either interpolation points or segments. These two methods cannot be combined. If the cam is to be created from segments using polynomials, **SIMOTION SCOUT** provides the **VDI Wizard** to assist in creation of the cam.

Defining with CamTool

CamTool is an add-on to the **SIMOTION SCOUT** engineering system providing a graphical option for creation of cams. The add-on is supplied with its own documentation.

Defining by means of program

SIMOTION provides various commands for creation of cams from the application. In ST, cams can be defined by specifying interpolation points, segments, interpolation type, and scaling.

For additional information, see Cam Programming/References (Page 2846).

Loading cams

With V4.2 and higher, an active, meshing cam can be loaded with CamEdit or by pressing the **Download** button.

Engaged cams can, therefore, be modified in the engineering system and replaced online. Neither the active camming motions nor active calculation of the valve characteristics are interrupted.

With versions lower than V4.2, it is not possible to download meshing cams to the device.

A cam is only calculated during runtime in the runtime system.

To facilitate diagnostics, the cam can be uploaded again from the runtime system following download so that any changes due to dynamic response adaptations can be detected.

If the cam is modified during runtime and the original cam is going to be downloaded to the runtime system again, the cam must be recompiled in SIMOTION SCOUT. (Otherwise, no change will be detected and the download will be skipped.)

See also

Commands for definition (Page 2847)

4.7.5.4 Cam Programming/References

Table 4-324 Commands for cam programming

Command	Description
Message functions	Commands for reading out function values (Page 2849)
_getCamFollowingValue **_getCamFollowingDerivative** **_getCamFollowingMinMax**	The **_getCamFollowingValue** command returns the cam value, **_getCamFollowingDerivative**, the nth derivation for a specified value in the definition range. The **_getCamFollowingMinMax** command returns (with V4.2 and higher) the minimums and maximums of the cam or of an area of the cam and the associated nth derivation in a specified section of the definition range.
_getCamLeadingValue	**_getCamLeadingValue** returns the value in the definition range for the specified value in the value range
Command tracking	Commands for command tracking (Page 2851)
_getStateOfCamCommand	The **_getStateOfCamCommand** command returns a structure with the processing state of a command.
_bufferCamCommandId	With **_bufferCamCommandId**, the command status can be queried after completion or an abort of the command.
_removeBufferedCamCommandId	With **_removeBufferedCamCommandId**, the command should be explicitly removed from the command management of the TO after evaluation is completed.

Command	Description
Geometry	Commands for definition (Page 2847)
_addSegmentToCam	The **_addSegmentToCam** command provides you with the option of defining a cam profile in the user program on the basis of polynomial segments f = f(t).
_addPointToCam	The **_addPointToCam** command provides you with the option of defining a cam profile in the user program on the basis of individual interpolation points.
_addPolynomialSegmentToCam	The **_addPolynomialSegmentToCam** command creates a segment f = f(t), consisting of a polynomial with a maximum of 6 degrees.
_interpolateCam	Before a cam is used, it must first be interpolated with the **_interpolateCam** cam.
_setCamScale	The **_setCamScale** command scales a cam in the range or domain. See Scaling and offset (Page 2833).
_setCamOffset	The **_setCamOffset** command can be used to separately offset the domain and/or range of a cam. See Scaling and offset (Page 2833).
Object and Alarm Handling	Commands for resetting the cam and errors (Page 2850)
_getCamErrorNumberState	The **_getCamErrorNumberState** command can be used to fetch the status of an error number.
_resetCam	The **_resetCam** cam resets the cam.
_resetCamError	The **_resetCamError** command provides you with the option of acknowledging a particular error or all pending errors on the cam.

Commands for definition

Interpolation points, segments, or a combination of the two can be used to describe cams.

Commands for the following actions are available for modifying the definition of a cam:

- Adding segments (**_addSegmentToCam**)
- Adding interpolation points (**_addPointToCam**)
- Adding a polynomial segment (**_addPolynomialSegmentToCam**)
- Interpolation (**_interpolateCam**)

Programming a cam

When the cam is created, the order in which the cam is edited plays a role. If the shape of the cam is dependent on parameters and undergoes a change, the cam must be reset before each redefinition using the **_resetCam** command. The cam can also be reset prior to the first calculation without triggering an error message. This command "erases" the cam. This means that the interpolation is cleared and the points and segments are removed. The technology object is retained but is empty.

The interpolation points and/or segments are then placed side by side in the appropriate order. As with CamEdit, the interpolation points can be in any order and will be sorted automatically.

- This is accomplished using the **_addPointToCam** command (addition of an interpolation point), the **_addSegmentToCam** command (addition of a segment) or **_addPolynomialSegmentToCam** (addition of a polynomial segment).
- Use the **_resetCam** command to define cam behavior for overlapping segments/ranges.
- If the shape of the cam is described completely, interpolation is performed using the **_interpolateCam** command.

Subsequent addition of one or more points or segments, or modification of points or segments is not possible. In this case, the cam must be reset and recreated.

Adding segments (_addSegmentToCam)

The **_addSegmentToCam** command provides you with the option of defining a cam profile in the user program on the basis of polynomial segments f = f(t). Individual segments consist of a polynomial with a maximum of 6 degrees and a trigonometric component.

The polynomial parameters, amplitude, period, and phase of a sine function must be entered in standard form. The transformation parameters must be specified in the basic cam representation (without scaling or offset) or in the actual cam representation (with scaling, offset). The definition range values of the cam must always be increasing, i.e. specified in the positive direction.

Adding interpolation points (_addPointToCam)

The **_addPointToCam** command provides you with the option of defining a cam profile in the user program on the basis of individual interpolation points. In so doing, you can choose to specify values either for a scaled and offset range or a range that is not scaled and offset. The definition range values of the cam must always be increasing, i.e. specified in the positive direction.

Adding a polynomial segment (_addPolynomialSegmentToCam)

The **_addPolynomialSegmentToCam** command creates a segment f = f(t), consisting of a polynomial with a maximum of 6 degrees. The polynomial parameters are input in the real range.

Interpolation (_interpolateCam)

Before a cam is used, it must be interpolated. The interpolation defines the connections between points, between segments and between a point and a segment.

For additional information, see Interpolation (Page 2835).

Commands for reading out function values

The following commands can be used to read out individual function values from cam characteristics.

- The **_getCamLeadingValue** command supplies the value in the domain (master value) for the specified value in the range (slave value).
  Since this relationship is not always unique, a reference value can be specified. See Inversion (Page 2840).
- The **_getCamFollowingValue** command supplies the cam with a specified value in the domain (master value).
- The **_getCamFollowingDerivative** command can be used to obtain the n-th derivative of the function for a specified value in the domain (V4.0 and higher).
  The **derivativeOrder** parameter can be used to select the n-th derivative.
  This allows, for example, the application to replace specific cams and so take account of the velocity, acceleration, etc.

The **leadingPositionMode** parameter can be selected for the commands to specify whether the scaling and offset is to be used (**ACTUAL**) or not (**BASIC**).

Reading out the minimums/maximums of a cam

The **_getCamFollowingMinMax** command returns (with V4.2 and higher) the minimums and maximums of the cam or of an area of the cam and the associated nth derivation in a specified section of the definition range.

The minimums/maximums are also available (in the case of V4.2 and higher) via the **followingRange** system variable.

System variables

The sequence of programmed commands can be read out by means of system variables; see Monitoring the synchronization (Page 2671).

The **activeCam** variable indicates the active cam on the synchronous object. The variable is a read-only variable.

The **activeMaster** variable indicates the active master on the synchronous object. The variable is a read-only variable.

Commands for resetting the cam and errors

- The **_resetCam** command has the following effect:
  - The cam is reset to the initial state.
  - Pending errors are deleted.
  - Depending on the command parameters, the geometry and corrections are deleted as applicable (see table below).
  - The system variables are reset according to parameters.
    With versions lower than V4.2, resetting an active cam that is interconnected with a synchronous operation function by means of the **_enableCamming** command always results in an error message that has to be acknowledged. The **_resetCam** command cannot be executed.
  - With V4.2 and higher, the **camModify = WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM** parameter can even be used to reset an active cam. This makes it easy to modify the cam subsequently.
- The **_resetCamError** command provides you with the option of acknowledging a particular error or all pending errors on the cam.
  The command is terminated with a negative acknowledgment if any errors that must not be acknowledged at this point are present.
- Parameter of the **_resetCam** command (for additional information, see TP system functions reference list)
- **insertMode** parameter (optional). Specifies the sequence interpretation with respect to subsequent entry of overlapping segments.
- **userDefaultData** parameter (optional). This parameter controls whether the user default values are reset to the configured values.
- **activateRestart** parameter (optional). Can be used to perform a technology object restart as well.
- **camData** parameter (optional). Specifies how the cam is to be reset.
- **camModify** parameter (optional, V4.2 and higher). This parameter controls the cam status during and after the **_resetCams** in relation to the geometry data.

Table 4-325 Modes of operation for the parameters associated with the **_resetCam** command for modifying a cam via the user program (UP)

	camModify parameter (V4.2 and higher)		
	WITH_INTERPOLATION	**WITHOUT_INTERPOLATION**	**WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM**
camData parameter			
CAM_DATA_RESET	Deletes the currently effective cam. The cam can be reprogrammed. *) (Also applies to < V4.2)	Deletes the currently effective cam. The cam can be reprogrammed. *)	The active cam is retained. The cam can be reprogrammed in the background in parallel. *) **_interpolateCam()** activates the cam programmed in the background.

LOAD_CONFIGURED_DATA	The currently effective cam is loaded and interpolated with the configured interpolation with the data specified by the ES. (Also applies to < V4.2)	The currently effective cam is loaded with the data specified by the ES without interpolation and can be modified via UP before interpolation is then performed. *)	The active cam is retained. The cam is reloaded in parallel in the background without interpolation with the data specified by the ES and can be modified via the UP. *) **_interpolateCam()** activates the cam programmed in the background.
MAINTAIN_PROGRAMMED_DATA (V4.2 and higher)	Regardless of whether it is loaded from the ES or programmed via the UP, the currently effective cam is interpolated with the current interpolation settings.	Regardless of whether it is loaded from the ES or programmed via the UP, the currently effective cam is loaded without interpolation. It can then be modified via the UP and subsequently interpolated. *)	The active cam is retained. Regardless of whether it is loaded from the ES or programmed via the UP, the cam is loaded in parallel in the background without interpolation. It can then be modified via the UP. *) **_interpolateCam()** activates the cam programmed in the background.

*) Alteration of geometry definition, e.g. with **_add**...**ToCam()**

Commands for command tracking

- The **_getStateOfCamCommand** command returns a structure with the processing state of a command.
  - **functionResult** specifies the error code.
  - **commandIdState** returns the current state of the cam.
  - **abortId** specifies the command abort reason. The abort reason is specified for the alarm 30002 "Command aborted (reason: <abortId>, command type ...)".
    See also **_getMotionStateOfAxisCommand** for the **Axis technology object**.
- With **_bufferCamCommandId**, the command status can be queried after completion or an abort of the command.
- With **_removeBufferedCamCommandId** , the command should be explicitly removed from the command management of the technology object after evaluation is complete.

The number of motion commands that the MotionBuffer can accept can be specified using the **camType.DecodingConfigInfo.numberOfMaxbufferedCommandId** configuration data.

Further information is available in the SIMOTION reference lists.

Programming and sequence model

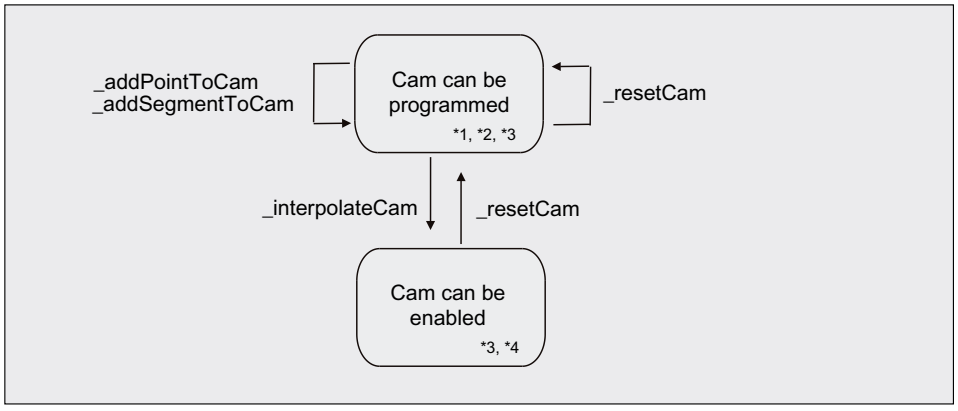


Figure 4-797 Programming and sequence model for a cam technology object

The following commands and functions are active in the particular technology object states:

*1 **_setCamScale**
_setCamOffset

*2 **_addPointToCam**
_addSegmentToCam

*3 **_getCamFollowingValue**, **_getCamFollowingDerivative**
_getCamLeadingValue

*4 **Error**
In general, illegal parameters and commands do not affect the states of the technology object, but they must be acknowledged using **_resetCamError**.

If commands that are not permitted in the technology object state are transmitted to the **Cam** technology object, an error message is triggered, which requires an acknowledgment. The technology object state, such as **programmable** or **cannot be activated**, is retained.

Local alarm response

Local **alarm responses** are set by means of the system.

The following responses are possible:

- **NONE**
  No response
- **DECODE_STOP**
  Command preparation aborted

Following a **_resetCam** or **_resetCamError**, editing of the Cam technology object can be resumed.

4.7.5.5 Graphic output

Reading out cams with CamEdit

Cams can be read out from the controller and displayed graphically using CamEdit.

A cam is only calculated during runtime in the runtime system. If it is read out from the controller, the calculated cam can be displayed graphically in CamEdit. If the cam is modified during runtime and the original cam is going to be downloaded to the runtime system again, the cam must be recompiled in SIMOTION SCOUT. (Otherwise, no change will be detected and the download will be skipped.)

By default, the cam is graphically displayed in the standard form in CamEdit. If scaling and/or offsetting of the cam is required, display in scaled form must be explicitly enabled.

If the command **_resetCam** with the parameter **camModify = WITHOUT_INTERPOLATION_AND_HOLD_ACTIVE_CAM** has been output in the user program, the active cam disk will be retained. On read-out, the data of the cam disk loaded in parallel in the background and modified via user program are fetched.

4.8 TO Axis Electric / Hydraulic, External Encoder

Preface

Content

This document is part of the **System and Function Descriptions** documentation package.

Scope of validity

This manual is valid for SIMOTION SCOUT product version V5.4:

- SIMOTION SCOUT V5.4 (engineering system of the SIMOTION product family)
- SIMOTION Kernel V5.4, V5.3, V5.2, V5.1, V4.5, V4.4, V4.3, V4.2, V4.1, V4.0
- SIMOTION technology packages CAM, CAM_ext and TControl in the version adapted to the respective kernel.

Sections in this manual

The following is a list of sections included in this manual along with a description of the information presented in each section.

Part I Axis

- **Overview**
  This section provides the user with an overview of the Axis technology object.
- **Axis Fundamentals**
  This section explains the basic setting options and functions of the Axis Technology Object.
- **Configuring an Axis**
  This section explains the configuration procedure with reference to various tasks.
- **Support for SINAMICS Safety Integrated Functions**
  This section describes the SIMOTION-side support for safety functions for SINAMICS drives.

Part II Hydraulic Functionality

- **Overview**
  This section provides the user with an overview of the hydraulic functionality of the Axis technology object.
- **Fundamentals of Hydraulic Functionality**
  This section explains the basic setting options and functions concerning the hydraulic functionality of the Axis technology object.

Part III Programming / Reference

- This section explains the commands and functions in greater detail.

Part IV External Encoder

- **Description**
  This section provides the user with an overview of the External Encoder technology object.
- **External Encoder Fundamentals**
  This section explains the basic setting options and functions of the External Encoder technology object.
- **Configuring an External Encoder (online help only)**
  This section explains the configuration procedure with reference to various tasks.
- **Programming External Encoders Reference**
  This section explains the commands and functions in greater detail.

Index

- Keyword index for locating information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics

- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

4.8.1 Fundamental safety instructions

4.8.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.8.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

4.8.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

4.8.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

4.8.2 Part I Axis - Overview

4.8.2.1 General information about axes

Overview of available axis types

The instantiable **Axis technology object (TO)** is downloaded to the SIMOTION device with a technology package, thus providing the function for activating and monitoring an actuator (drive, motor, valve, etc.). As of V3.2, the **Axis technology object (TO)** is included in the Cam, Path, and Cam_ext technology packages.

The functionality is set by configuration and parameter assignment / programming.

The Axis technology object can be applied to an axis with an electric drive (axis) or an axis with a hydraulic final controlling element / valve (hydraulic functionality).

Any number of axes may be generated, based on the CPU processing power.

When programming in SIMOTION SCOUT (e.g. with MCC), an Axis technology object can be accessed via system functions or system variables. For example, to traverse an axis at a specified velocity to a certain position, you specify the velocity and position via system functions. All other functions (e.g. monitoring of limit values) are specified via the configuration data and system variables of the Axis technology object.

The following axis technologies are distinguished during the configuration phase:

- **Speed-controlled axis**
  Motion control is performed using a speed specification without position control.
  Choosing this axis technology provides the minimum functionality for an axis.
  The speed-controlled axis is referred to by the data type **driveAxis** in reference lists and when programming.
- **Positioning axis**
  Motions are position-controlled.
  The position axis is referred to by the data type **posAxis** in reference lists and when programming.

- **Synchronous axis**
  The synchronous axis creates a grouping of the following axis and synchronous object. Functions such as master value coupling, synchronization and desynchronization, and gearing and camming are provided via the synchronous object. The synchronous object can be interconnected with different master values.
  See the Technology Objects Synchronous Operation, Cam manual for information about using the synchronous axis.
  The following axis is referred to by the data type **followingAxis** and the synchronous object by **followingObjectType** in reference lists and when programming.
- **Path axis**
  The path axis type can be interconnected with a path object.
  The path object can be used to calculate and traverse a linear, circular or polynomial path in the 2D/3D coordinate system for at least two path axes and up to three path axes. A synchronous axis can be traversed in parallel to this.
  The Technology Object Path Object manual describes how to use the path axis with the path object.
  The path axis is referred to by the data type **_pathAxis** and the path object by the data type **_pathObjectType** in reference lists and when programming.

All axis types can also be configured as **virtual axes**, i.e. they do not control a real drive but are used as auxiliary axes for calculations, e.g. as a virtual master axis for several slave axes (line shaft).

Axis operation in SIMOTION

- Any necessary settings can be made by means of configuration data on the axis.
- States are indicated, and standard values and settings can be read and entered by means of system variables on the axis.
- Axis motion sequences are specified by means of motion commands on the axis. The user program can be used to check the motion status at any time and to control specific aspects of the motion. Motions can be aborted, overridden, appended, or superimposed.
- Errors and technological alarms are indicated using alarms on the axis.

Advanced functions

Advanced functions on the axis include path interpolation, synchronous operation, measuring inputs, and output cams. For more information refer to the manuals: Technology Object Path Object, Technology Object Synchronous Operation, and Technology Objects for Output Cams and Measuring Inputs.

Functional interface to the drive

The functional interface to the drive is the speed setpoint interface.

The functional interface to a hydraulic valve is the analog flow rate setpoint and, if available, the analog force limiting or pressure limit value.

It is possible to connect both analog drives and digital drives as well as stepper drives.

Standardized protocols are used for setpoint specification for digital drives and for acknowledgments of encoder information.

Note

A SIMOTION axis can only execute functions that are supported by the connected drive. For further information, refer to the relevant product descriptions for the drives.

Automatic configuration as of V4.2 (Use symbolic assignment setting)

The only settings required during configuration are the preferred technology and functionality for the axis and encoder and the drive to be used by the axis. The PROFIdrive telegrams required for communication are determined and set by the system when the online connection is established.

Automatic adaptation uses add relevant drive data to the TO configuration without having to make these settings manually.

This functionality is available as of with SIMOTION V4.2 in conjunction with SINAMICS V2.6.2.

Programming commands / functions for the Axis technology object

The MCC and ST programming languages are available for programming axes.

Axis functions can also be addressed via the PLCopen blocks of the SIMOTION Function Library (up to V3.2) and the SCOUT command library (as of V4.0). This can also be accomplished in the LAD and FBD programming languages.

See the *SIMOTION MCC Motion Control Chart*, *SIMOTION ST Structured Text* and *PLCopen* Programming Manuals.

4.8.3 Axis fundamentals

4.8.3.1 Axis technologies

Overview of axis technologies

The Axis technology object can be configured as a speed-controlled axis, positioning axis, synchronous axis, or path axis. The various axis technologies differ according to the functionality provided on the axis.

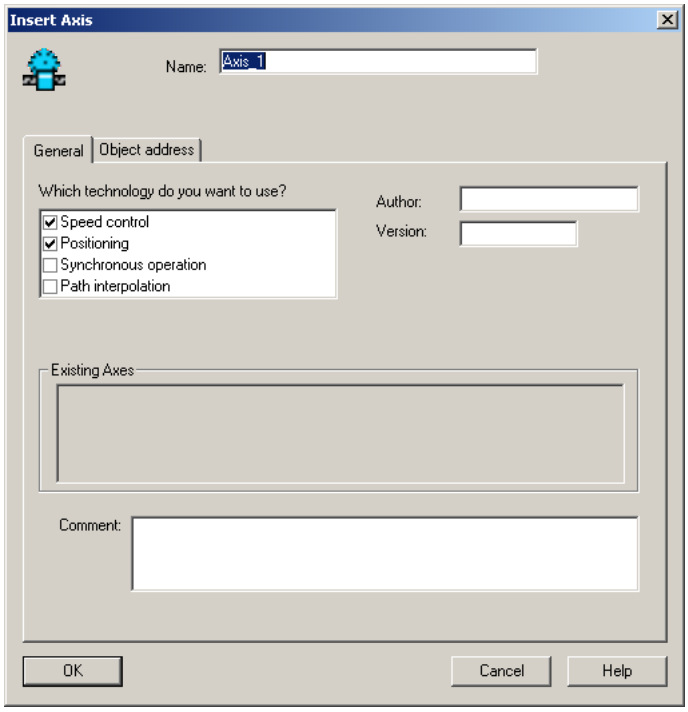


Figure 4-798 Axis technology setting in SIMOTION SCOUT

Possible axis functions in relation to the axis type

Table 4-326 Axis functions - Overview

Function	Speed-controlled axis	Positioning axis	Following axis	Path axis
Speed or velocity specification	X	X	X	X
Travel with torque limitation	X	X	X	X
Traversing according to motion vectors	X	X	X	X
Positioning	-	X	X	X
Travel to fixed stop	-	X	X	X
Homing	-	X	X	X

Function	Speed-controlled axis	Positioning axis	Following axis	Path axis
Advanced functions				
Measuring input	-	X	X	X
Output cam	-	X	X	X
Cam track	-	X	X	X
Gearing	-	-	X	X
Camming	-	-	X	X
Path interpolation	-	-	-	X

Operating modes

- Setpoint operation
  Setpoint mode is the "normal" mode of the axis, in which motion commands are accepted and executed.
- Follow-up mode
  The setpoint is corrected to the actual value in follow-up mode. The actual position value and actual speed value will be updated. The axis can then also be tracked if it is moved by external effects.
  Motion commands are not accepted/executed.
- Simulation mode
  The axis and the position controller are active in simulation mode. Simulation mode is used to test the programmed sequences in the control and the interaction between various axes, for example, on the basis of trace recordings without moving the axis.
  Simulation mode is only useful for real axes.
  Two variants of simulation mode are possible:
  - Program simulation mode
    The setpoints are calculated according to the programming, but are not sent to the position controller. The position controller setpoints remain set at the values they had prior to switching to simulation mode.
    See also Enabling/disabling program simulation (Page 3170).
  - Axis simulation mode
    Unlike program simulation, it is possible to switch a real axis to axis simulation status even if the drive is not connected. The position controller remains active and the drive is simulated.
    See also Setting as a real axis without drive (axis simulation) (Page 2899).

The operating modes are set via commands.

Speed-controlled axis

The speed-controlled axis type is used when the position of the axis is of no importance. Only the speed or velocity of an axis is specified, controlled, and monitored.

The speed is monitored if an encoder is configured on the axis; otherwise, the speed is not monitored.

The position of the speed-controlled axis is not monitored or controlled.

The speed of rotation is indicated as a unit of velocity, such as rpm.

Operating modes

- Speed controlled / speed specification (setpoint mode)
- Simulation
- Follow-up mode

Functions

- Speed or velocity specified via
  - Programmable value setting
  - Free velocity profile (time-related)
- Travel with torque limitation
- Force/pressure control, force/pressure limiting (for hydraulic functions only)

Positioning axis

The positioning axis type specifies, controls, and monitors the position of the axis.

The axis is moved to a programmed target position, which can be specified as a relative or an absolute value. The direction of motion or rotation can be specified for modulo axes.

For position-controlled positioning axes, position control is possible in the CPU or in the drive, provided the drive supports the dynamic servo control (DSC) method.

The positioning axis in SIMOTION does not have a velocity controller. The speed controller for an electric axis is in the drive.

The positioning axis can be interconnected with a path object for path-synchronous motion as a master axis with synchronous axis or as an axis.

Operating modes

As for speed-controlled axis, plus

- Position control

Functions

- Speed or velocity specified via
  - Programmable value setting
  - Free velocity profile (time- or position-related)
- Travel with torque limitation
- Position specified via
  - Programmable value setting
  - Free velocity profile (time-related)
- Travel to fixed stop
- Homing
- Force/pressure control, force/pressure limiting

- Synchronization of encoder values
- Positioning axis with path-synchronous motion (see path interpolation)

Synchronous axis

The synchronous axis type determines the axis setpoint from a master value according to conversion rules.

The synchronous object and following axis are separate objects, but together they form a synchronous axis.

The "Axis" and "Synchronous Operation" technology objects have a reciprocal effect on each other in terms of their operating states and the effectiveness of the commands used.

For example, errors in the "Axis" technology object will directly affect the synchronous operation functionality. When an axis stop response is triggered, the synchronous motion is stopped as well.

Operating modes

As for positioning axis

Functions

In addition to the positioning axis functions, other functions are available via the synchronous object:

- Gearing
- Camming
- Velocity gearing
- Dynamic synchronization/desynchronization

Other functions associated with the Synchronous Axis technology object can be found in the SIMOTION Technology Objects for Synchronous Operation, Cam Manual.

Path axis

The path axis type is used to travel along a path together with at least one additional path axis on the path object.

Via the path object, a path can be generated for at least two and up to three path axes.

The setpoints generated for the axis on the path object are limited on the axis to the maximum dynamic values of the axis.

The "Path Axis" and "Path Object" technology objects have a reciprocal effect on each other in terms of their operating states and the effectiveness of the commands used.

For example, errors in the "Path Axis" technology object will directly affect the generation of motion on the path object. When a stop response is triggered on the axis, the path motion is stopped as well.

Operating modes

As for positioning axis

Functions

In addition to the positioning axis functions, other functions are available via the path object:

- Linear path interpolation
- Circular path interpolation
- Polynomial path interpolation

The path axis contains the functionality of the synchronous axis.

Other functions associated with the Path Interpolation technology object can be found in the *SIMOTION Technology Object Path Object* Manual.

See also

Setting and canceling the axis enables (Page 3143)

Axis-drive relationship

Technology objects represent the applicable real objects (e.g. an axis) in the controller.
The Axis technology object provides the user with a technological view of the drive and encoder (actuator and sensor), provides technological functions for them and performs the specific hardware connection.

The Axis technology object includes comprehensive functionality, e.g. communication with the drive, actual value conditioning, position control and positioning functions. It executes control and motion commands and indicates statuses and actual values.

The drive contains the speed and current control for the motor.

Technological limiting functions and values for the axis and encoder mechanics (e.g. leadscrew pitch and gear) are set at the axis.
The user can then only work with technological parameters.

The Axis technology object communicates with an actuator (drive or hydraulic valve) via a field bus system (PROFIBUS or PROFINET via PROFIdrive protocol) or via a direct setpoint interface (analog ±10 V or pulse/direction).

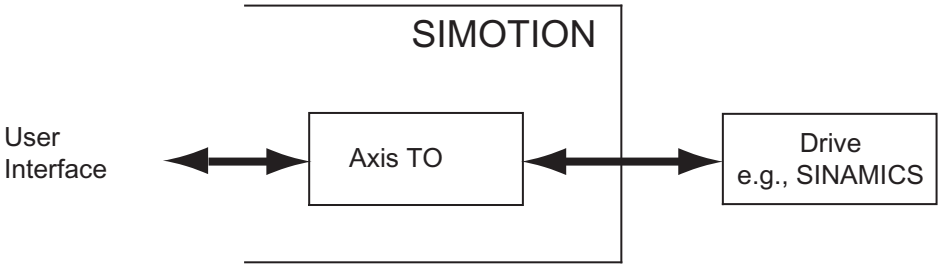


Figure 4-799 Axis-drive relationship

Functional interface to the drive

Various functional interfaces to the drive are available.

Analog drives, hydraulic valves, or stepper drives can be operated at the direct setpoint interface. See also Setting as a real axis with analog drive link (Page 2877), Linking analog drives to SIMOTION (Page 3062).

Setpoint specifications and feedback signals (incl. encoder information) for drives connected to a field bus are provided via standardized protocols (standard telegrams in accordance with the PROFIdrive profile). See also Coupling of digital drives (Page 2895).

See also

Axis with stepper motor connection (Page 3063)

Overview of axis settings / drive assignment (Page 3114)

Encoder interface using the PROFIdrive message frame (Page 2906)

4.8.3.2 Axis types

Overview of axis types

There are different axis types that differ according to their axis mechanism. The axis type also determines the units used to calculate axis variables such as position, velocity, etc.

- **Linear axes**
  Linear axes have coordinates that are specified in a unit of length. The position profile is continuous within the traversing range. Motion instructions are specified in units of length, for example in mm.
- **Rotary axes**
  Rotary axes have coordinates that are specified in a unit of rotation. The position profile is continuous within the traversing range. Motion instructions are specified in units of rotation, for example in degrees or rad.
- **Setting a linear axis or rotary axis as a modulo axis**
  Modulo axes have an unlimited traversing range and their position is mapped to a repeating modulo traversing range. The modulo range is defined by the start value and the modulo length.
  If the position value or axis position overshoots the modulo length, the position is reset to the modulo start value. If the position value undershoots the modulo start position, it is reset to the modulo start value plus the modulo length.
  Like rotary axes, linear axes can also be defined as modulo axes (modulo linear axis, modulo rotary axis).

Axis type setting

The following settings are available for the **axis type**:

- **Linear**
- **Rotary**

and

- **Electrical**
- **Hydraulic**
- **Virtual**

The **electrical**, **hydraulic**, or **virtual** setting affects the subsequent menu content.

See also

Actual value measurement / actual value system (Page 2969)

Setting for the electric axis type

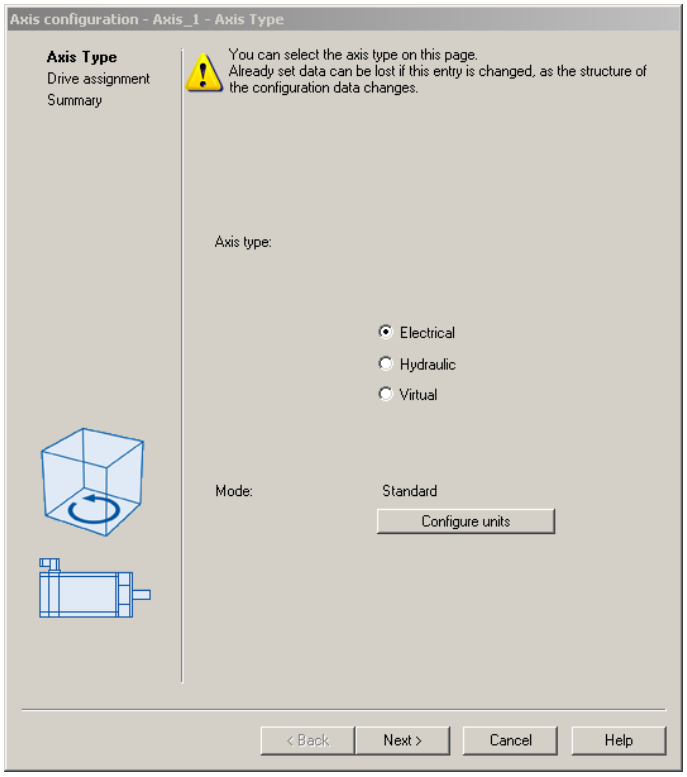


Figure 4-800 Axis type setting for an electric drive axis in SIMOTION SCOUT

Note

There is no force/pressure control for the electric drive axis. In this case, the mode is preset to Standard and cannot be changed.

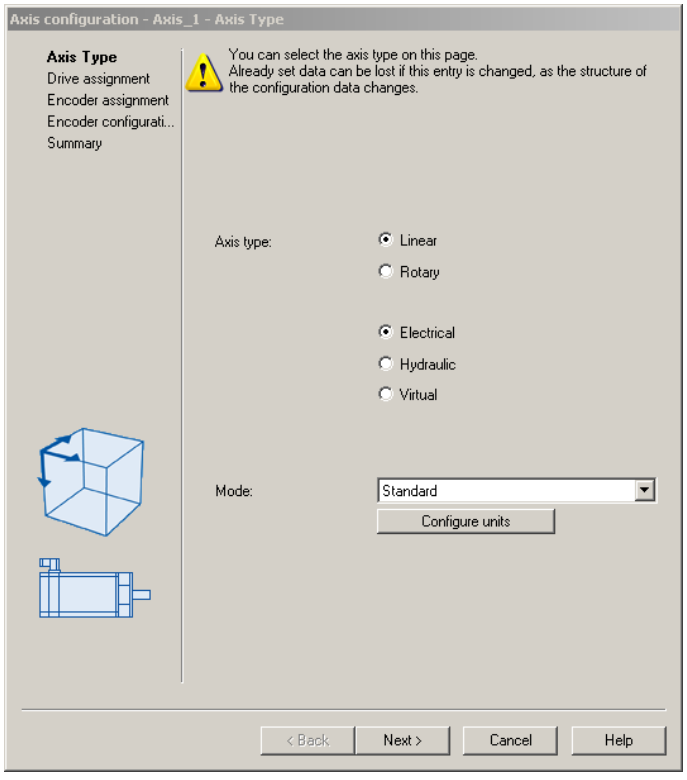


Figure 4-801 Axis type setting for an electric position or following axis in SIMOTION SCOUT

Table 4-327 Axis type options

Axis type	Description
Linear	Setting as linear axis
Rotary	Setting as rotary axis

Table 4-328 Mode options

Mode	Description
Standard	Position control
Standard + pressure	Position control and pressure control/pressure limitation
Standard + force	Position control and force control/force limitation

See also

Overview of force/pressure control (Page 3026)

Overview of force/pressure limiting (Page 3033)

Overview of axis types (Page 2866)

TypeOfAxis configuration data (Page 2870)

Setting for the hydraulic axis type

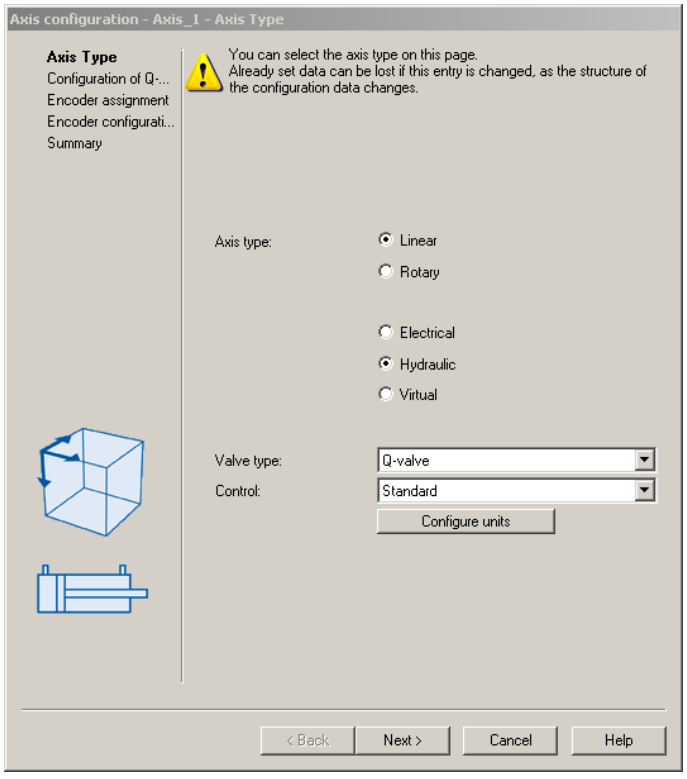


Figure 4-802 Axis type setting for a hydraulic position or following axis in SIMOTION SCOUT

Table 4-329 Valve type options

Valve type	Description
Q-valve	Axis with Q-valve (volumetric flow control)
P-valve [1]	Axis with P-valve (force/pressure control)
P+Q-valve	Axis with P+Q-valve

[1] Additional choice for drive axis

Table 4-330 Closed-loop control options

Closed-loop control	Description
Standard	Position control only
Standard + pressure	Position control and pressure control
Standard + force	Position control and force control

See also

Overview of force/pressure control (Page 3026)

Overview of force/pressure limiting (Page 3033)

Hydraulic functionality overview (Page 3113)

TypeOfAxis configuration data (Page 2870)

Setting as a real axis with Q valve only (Page 3116)

Setting as a real axis with Q valve + P valve/F output (Page 3118)

Setting an axis as a real axis with P valve only (V3.2 and higher) (Page 3120)

Setting for the virtual axis type

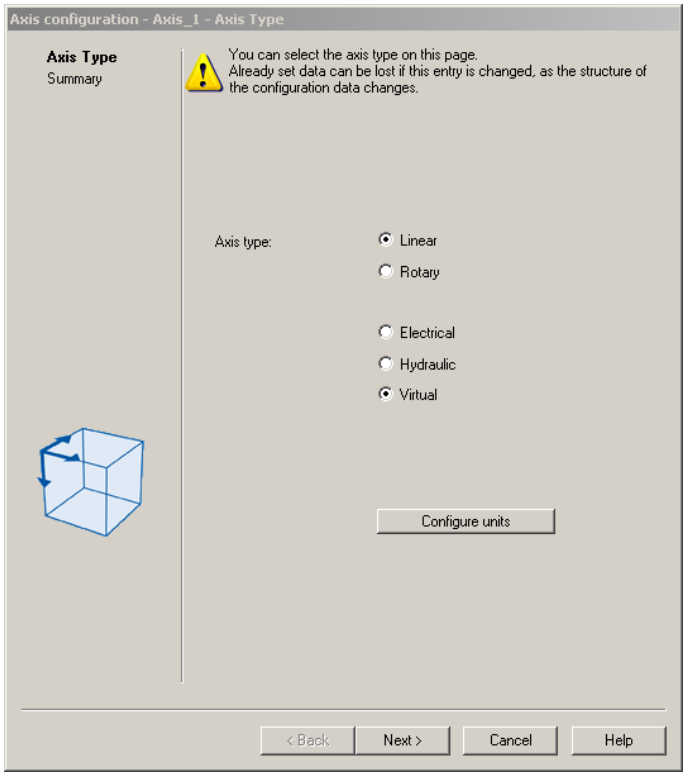


Figure 4-803 Axis type setting for a virtual position or following axis in SIMOTION SCOUT

TypeOfAxis configuration data

The axis type is entered according to the axis type parameter assignments under the **TypeOfAxis** configuration data in the axis wizard. The following table shows which axis wizard parameterization corresponds to which value under **TypeOfAxis**.

Assignment of **TypeOfAxis** based on the current axis wizard configuration:

	Axis type					Closed-loop control			Valve type d		
TypeOfAxis	Linear c	Rotary	Electrical	Hydraulic	Virtual	Standard	Standard+force	Standard+pressure	P valve	Q valve	P+Q-valve
VirtualAxis	x				x						
		x			x						
Real_Axis	x		x			x					
		x	x			x					
Real_Axis_With_Signal_Output	x										
		x	x								
Real_Axis_With_Force_Control [a]	x		x				x				
		x	x				x				
	x		x					x			
		x	x					x			
Real_QFAxis	x			x		x				x	
		x		x		x				x	
Real_QFAxis_With_Open_Loop_Force_Control	x			x			x				x
		x		x			x				x
	x			x				x			x
		x		x				x			x
Real_QFAxis_With_Closed_Loop_Force_Control	x			x			x			x	
		x		x			x			x	
	x			x				x		x	
		x		x				x		x	
Real_QFAxis_With_Open_Loop_Force_Control_only [b]		x		x			x		x		
		x		x				x	x		

a not possible for speed-controlled axis
b speed-controlled axis only
c not possible for speed-controlled axis
d hydraulic functionality only

4.8.3.3 Units and accuracies

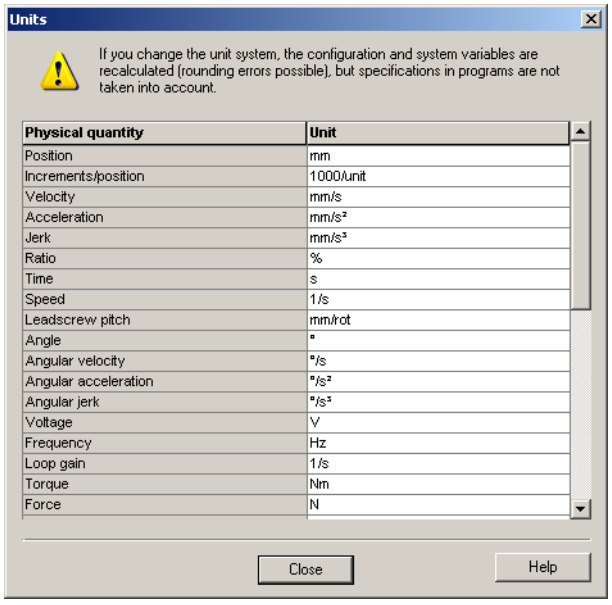


Figure 4-804 Setting the units and resolution in SIMOTION SCOUT

With the SIMOTION technology objects, such as the Axis technology object, physical variables such as position, velocity, acceleration, time, force, and torque are represented in the SI or US system of units (metric or imperial). The unit can be defined for all variables on the relevant technology object during configuration, for example, for:

- Unit of length
  - mm
  - m
  - km
  - inch
- Unit of force
  - N
  - kN
  - tfm
    ton force, metric unit
  - tfs
    ton force (short) or ton force (US), US unit

The defined units represent system variables and configuration data.

Changing the unit settings converts the current values of the system variables and configuration data from the engineering system to the new units.

Note

Values in the command parameters are interpreted in the defined unit on the technology object.

Numerical values in user programs (e.g. in the motion commands) are **not** converted to the new units when the unit settings are changed!

In compound variables, such as controller gains, the units may differ from the units of the individual variables, for example, force in [kN] and force/time in [N]/[sec].

The **internal computational accuracy and internal representation of the position** can also be defined in the unit configuration. It defines, among other things, the accuracy with which specifications in system variables, configuration data, and command parameters are accepted, processed, and displayed by the system.

Note

The positioning accuracy equates to one computational increment (increments/position), i.e. positions can only be defined at integer increment values. Intermediate values arising during interpolation and as a result of closed-loop control may also exist between the integer increments.

The display resolution and the resolution when entering parameter values are independent of the increments.

This definition refers to a specific basic unit of the position, depending on the axis type.

- Linear axis: Increments/mm
- Rotary axis / speed-controlled axis: Increments/degree

The control performs internal calculations in increments with reference to these basic units. Prior to processing, values are converted to the internal representation.

Example 1

The following configuration has been defined:

- Linear axis
- Position unit: m
- Increments/position: 1000/mm

Calculation of setpoint accuracy during positioning:

Position: 1,000/mm corresponds to 0.001 mm = 10^{-6} m

Example 2

The following configuration has been defined:

- Linear axis
- Position unit: mm
- Increments/position: 1000/mm

- Leadscrew pitch: 10.3334 mm
- Modulo length: 20.3335 mm

Determination of the effective leadscrew pitch and modulo length:

Position accuracy: 0.001 mm

- Effective leadscrew pitch on the TO: 10.333 mm
- Effective modulo length: 20.333 mm

If the leadscrew pitch and the modulo length have to be mapped precisely, the positioning accuracy on the TO has to be increased (incr/mm).

Fetch the units setting

The units setting can be fetched using the associated specially defined system variables. The listing of the variables is shown below:

- <speedaxis>.units : StructUnitsSpeedAxis
- <posaxis>.unitsPosAxis : StructUnitsPosAxis
- <externalencoder>.units : StructUnitsExternalEncoder
- <controller>.units : StructUnitsController
- <sensor>.units : StructUnitsSensor
- <synchronousoperations>.units : StructUnitsSynchronousOperations
- <temperaturechannel>.units : StructUnitsTemperatureChannel
- <pathobject>.units : StructUnitsPathObject
- <additionobject>.units : StructUnitsAdditionObject
- <fixedgear>.units : StructUnitsFixedGear
- <measuringinputs>.units : StructUnitsMeasuringInputs
- <outputcam>.units : StructUnitsOutputCam
- <camtrack>.units : StructUnitsCamTrack

Note

Fetching is possible only when the units have valid values, which is the case only in online mode.

For further information, see the *TP System Variables* List Manual.

4.8.3.4 Axis settings / drive assignment

Overview of how to create an axis

New axes are created using an axis wizard in which the axis parameters (configuration data and system variables) are queried or are configured automatically. You can specify other selected parameters via the axis parameter assignment dialog boxes (under Axis Object in the project navigator).

Real and virtual axes

Setting options

In SIMOTION, you can define the axes as:

- **Real axis**
  This axis features motion control, as well as drive and encoder interfaces.
- **Real axis with force/pressure control**
  This axis features motion control, drive and encoder interfaces, and an interface for force/pressure measurement and closed-loop control.
  For force/pressure control, the input for the force/pressure measurement must also be configured.
- **Virtual axis**
  This axis features reference variable generation, but does not have closed-loop control or a drive or encoder interface. The setpoints and actual values are always identical. A virtual axis is generally used as an auxiliary axis, in order to generate the setpoints for several real axes as the master axis, for example.
  The controller-specific enables are set by default.

See also Overview of axis types (Page 2866).

Axes with no drive assignment (as of V4.2)

A drive is assigned to the axis in the drive assignment dialog during axis configuration. It is also possible, however, to leave the drive assignment open. The real axis can be programmed with all functions and the system remains operational after axes with no assigned drive have been downloaded. No consistency error is output.

One advantage of axes with no assigned drive is that you can set up and test the configuration in advance while there is still no drive or if you do not have all the necessary drive know-how. The axis can be connected to a drive later.

Runtime model and processing

The integrated motion control functionality uses a deterministic real-time runtime model for motion control purposes. This includes, in particular:

- Isochronous system levels for
  - Bus task
    PROFIBUS/PROFINET communication for linking digital drives, data exchange with the I/O
  - Servo_fast (optional)
    Position control and monitoring of axes, drive communication, and I/O processing synchronized with a fast PROFINET cycle clock
  - Servo
    Position control and monitoring of axes, drive communication, and I/O processing
  - IPO_fast
    Motion control for the axes is calculated in the fast PROFINET bus system
  - IPO
    Interpolator = reference variable calculation/motion profile calculation of the axes
    The interpolator cycle clock is set during configuration of the execution system of the device. There are two interpolator levels in the system, IPO and IPO2.
- Adjustable transformation ratios between bus task, servo, and IPO for appropriate load distribution and optimum system utilization

The processing cycle clock (axis-specific interpolator cycle clock) of the axis technology object can be set to IPO or IPO2. This makes it possible to place an interpolator for axes that do not require a high time resolution to calculate reference variables in a cyclical system task with a longer cycle time, thereby requiring less processing power.

For fast responses in motion control, in exceptional cases, the processing cycle clock can even be set to servo or servo_fast, see also Motion execution/interpolator (Page 3045).

The processing cycle clock is set in the axis dialog configuration.

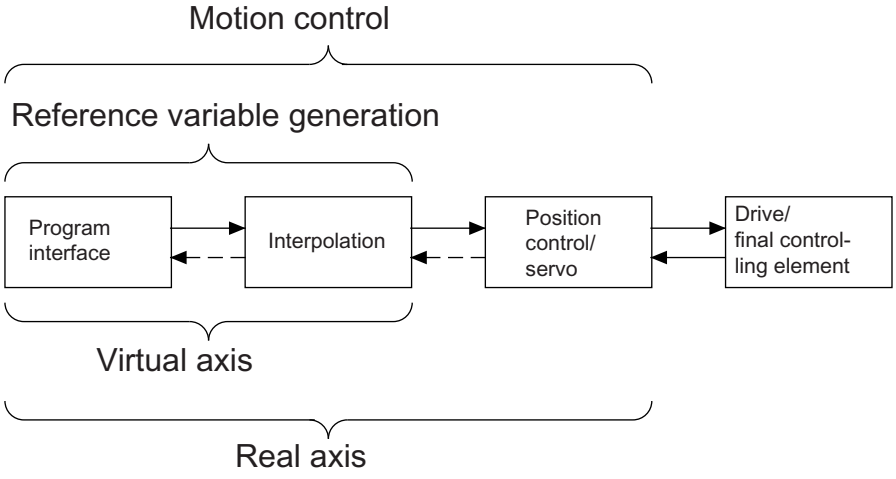


Figure 4-805 Difference between real and virtual axis (position axis example)

For real axes, the interface to the drives/final controlling elements is set.

For the hydraulic functionality, the analog output is set for manipulated variable value Q (flow rate) and, if applicable, for manipulated variable value F (force/pressure limiting). This makes it possible to connect valves with an analog manipulated variable.

If DSC (Dynamic Servo Control) is set for digital drives with a PROFIdrive interface, a position controller is executed in the drive (e.g. in the closed-loop speed control cycle clock).

If **symbolic assignment** is used (default setting as of V4.2), DSC is selected automatically.

Setting as a real axis with analog drive link

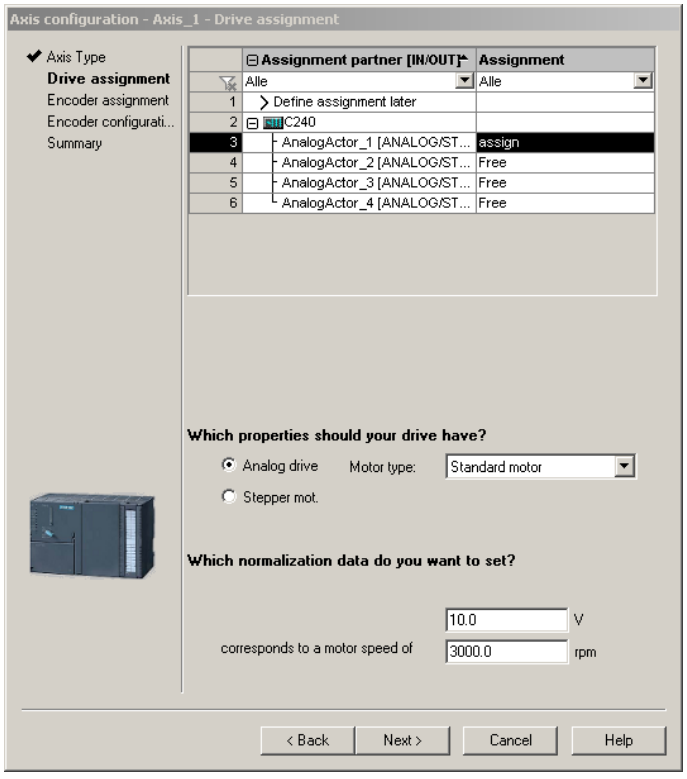


Figure 4-806 Setting for analog drive link to C2xx

For interconnection of the axis-specific inputs/outputs (encoder, analog output, enables), see the C2xx operating instructions.

In addition to the option of operating analog axes on the onboard inputs of the C2xx, the PROFIBUS ADI4 and IM174 modules are available for use in all platforms as interfaces for analog drive links. From the SIMOTION perspective, these modules behave like digital drive links; see Coupling of digital drives (Page 2895).

The enable signal to the drive is activated with **_enableAxis()** (**enableMode**=ALL); the enable is displayed by the system variables **actormonitoring.driveState** = ACTIVE and **actormonitoring.power** = ACTIVE.

See also

Coupling of digital drives (Page 2895)

Setting as a real axis with digital drive coupling

Symbolic assignment and adaptation (as of V4.2)

As of V4.2, SIMOTION supports the automatic adaptation of drive data as well as the configuration of technology objects (TOs) with symbolic assignment.
This makes it easier to configure the technical relationships between the SIMOTION technology objects (TOs) and the SINAMICS drive objects (DOs).

Adaptation and symbolic assignment enable:

- The required axis telegrams, as well as the addresses used, to be automatically specified by the engineering system
- Telegrams to be expanded and interconnections in the drive to be set up automatically dependent upon the selected TO technology (e.g. SINAMICS Safety Integrated)
- Axis configuration and drive configuration to be completed as two separate procedures
- Drive and encoder data, as well as reference variables, maximum variables, torque limits, and the selectivity associated with torque reduction on the SINAMICS S120 to be adapted automatically on system ramp-up

For more information, please refer to the sections of the **Motion Control Basic Functions** manual which deal with *symbolic assignment (as of V4.2)*.

Requirement

The following components support adaptation and symbolic assignment:

- SIMOTION C, P and D devices, with firmware as of SIMOTION V4.2
- SINAMICS S110 drives, firmware V4.x with SINAMICS as of V4.3
- SINAMICS S120 drives, firmware V2.x with SINAMICS as of V2.6.2
- SINAMICS S120 drives, firmware V4.x with SINAMICS as of V4.3

The SINAMICS versions indicated relate to:

- SINAMICS control units connected via PROFIBUS or PROFINET
- SINAMICS Integrated on a SIMOTION D
- Controller extensions CX32/CX32-2

Axis configuration: Drive assignment

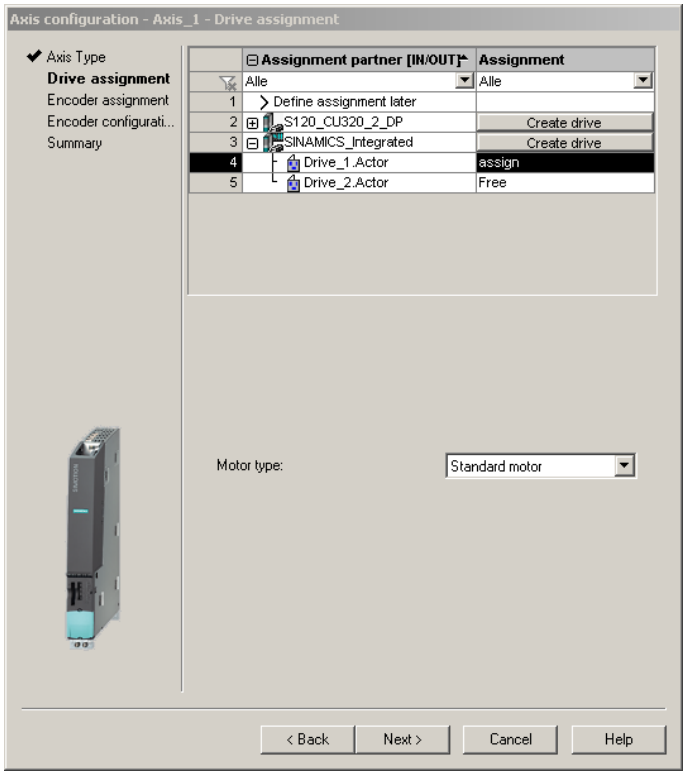


Figure 4-807 Drive assignment dialog

The following settings can be made in the **Drive assignment** dialog for the axis configuration:

- Assign drive
  All of the drives configured in SCOUT or HW Config are displayed as assignment partners. To interconnect a drive, click a drive object in the table in the **Drive assignment** dialog.
  The hardware addresses and telegrams are determined automatically by the system.
- Assign later
  Enables the axes to be configured independently on the SIMOTION side and the drives to be configured independently on the SINAMICS side.
  This means, for example, that:
  - The drives can be configured by a drive expert and tested and optimized using the drive control panel
  - The PLC and Motion Control functions can be configured by a programmer using technology objects (e.g. Axis technology object)
  - The technology objects can then be symbolically assigned to the drive objects at a later point in time via an assignment dialog
- Create drive
  From the Assign dialog, a new drive can be created on an existing drive unit directly (e.g. S120 CU320-2 or SINAMICS Integrated) and the axis assigned. As a result, it is not necessary to configure a drive before creating an axis.

The TO axis, TO external encoder, TO output cam, TO cam track, and TO measuring input support symbolic assignment. Furthermore, the onboard I/O of a SIMOTION D, a SINAMICS S120 control unit, and even selected terminal modules can be interconnected symbolically (e.g. for hardware-limit-switch inputs on the technology object).

The list of addresses in the **All addresses** view provides an overview of the assignments of all the Axis technology object interfaces. It is also possible to change the assignments from this view via the Assign dialog (**...** button).

Note

The previous methods of drive and axis configuration continue to be available. Symbolic assignment must be deactivated in such cases.

Symbolic assignment is used by default for newly created projects.

If projects < V4.2 are converted upwards, symbolic assignment is deactivated by default and must be activated if required.

Symbolic assignment can be activated/deactivated in SIMOTION SCOUT via the **Project > Use symbolic assignment** menu.

Adaptation

In addition to the benefits offered by symbolic assignment, configuration is also made easier as of SIMOTION V4.2 with the addition of automatic adaptation of SINAMICS S120 data.

Drive and encoder data as well as reference variables, maximum values, torque limits and granularity for SINAMICS S120 torque reduction, are taken over automatically for the configuration data of the SIMOTION Axis and External encoder technology objects. These data no longer have to be entered in SIMOTION.

Events and commands after which an adaptation is carried out:

- Switching the controller on (ramp-up)
- Download
- STOP-RUN transition, if no adaptation has been made yet
- Restarting a TO
- **_activateTO()** command
- **_enableAxisInterface()** command if the status "not adapted" is present at the interface with non-exclusive drive assignment
- **_adaptAxisConfigData()** and **_adaptExternalEncoderConfigData()** user commands
- Drive data set changeover (and encoder changeover does not cause adaptation)

The adaptation of relevant drive data is activated automatically as of SIMOTION V4.2 in conjunction with SINAMICS S120 as of V2.6.2. If projects < V4.2 are converted upwards, automatic adaptation is deactivated by default. You can activate automatic adaptation if required via the expert list in the configuration data element **TypeOfAxis.DriveControlConfig.dataAdaption**. You can activate automatic adaptation of encoder data in the configuration data element **TypeOfAxis.NumberOfEncoders.Encoder_n.dataAdaption**.

The configured data initially contains default values for the values to be adapted. Adaptation may result in a difference between the configured values and the adapted values. In the case of a *Copy current data to RAM*, the TO would, therefore, be displayed as inconsistent in the project navigator.

When executing the functions *Copy current data to RAM* and *Copy RAM to ROM*, the difference will be detected and an upload of the adapted data into the SCOUT project will be executed. However, this can be deselected in the dialog box.

Adapted data (SINAMICS S120 as of V2.6.2)

The adapted drive and encoder data is listed in the following.

Adapted drive data (motor and actuator)

Table 4-331 Reference and maximum velocity

Motor type	Parameter	Configuration data element
Round-frame motor	p2000	**driveData.nominalSpeed**
Linear motor	p2000	**linearMotorDriveData.nominalSpeed**
Round-frame motor	p1082	**driveData.maxSpeed**
Linear motor	p1082	**linearMotorDriveData.maxSpeed**

Table 4-332 Resolution of torque reduction

Motor type	Parameter	Configuration data element
Round-frame motor	p1544	**driveData.torqueReductionGranularity**
	p1520	**driveData.maxTorque** (and p1520 = |p1521|)
Linear motor	p1544	**linearMotorDriveData.forceReductionGranularity**
	p1520	**linearMotorDriveData.maxForce** (and p1520 = |p1521|)

Table 4-333 Resolution of the reference and maximum values for B+, B-, M_{add} (technology data block)

Motor type	Parameter	Configuration data element
Round-frame motor	p2003	**driveData.nominalTorque**
Linear motor	p2003	**linearMotorDriveData.nominalForce**

Table 4-334 Adaptation of the parameters for specific DO types and availability in the drive

Parameter	Semantics	DO Servo	DO Vector	DO TM41	DO DCM	DO Encoder
p2000	Reference speed	x	x	x	x	x
p2003	Reference torque	x	x	x	x	-
p1082	Maximum speed	x	x	-	-	-

Parameter	Semantics	DO Servo	DO Vector	DO TM41	DO DCM	DO Encoder
p1520	Maximum torque	x	x	-	-	-
p1544	Torque reduction granularity (only 100 telegrams)	x	-	-	-	-
p0979	Encoder information	x	x	x	x (only telegram 1)	x

The following conditions must be observed:

- When assigning an axis on a SINAMICS drive, the following is set implicitly:
  - **Adaptation**
    driveControlConfig.dataAdaption=YES
  - **speedReference** (speed, velocity) to nominalspeed, ~velocity
    driveDate.speedreference=NOMINAL_VALUE
    linearMotorDriveData.velocityReference=NOMINAL_VALUE
  - **torqueReference**, **forceReference** (torque, force) to nominaltorque, ~force
    driveDate.torqueReference=NOMINAL_VALUE
    linearMotorDriveData.forceReference=NOMINAL_VALUE
- Adaptation is not set generally for an interconnection to a SINAMICS DO, but specifically depending on the DO type.
  This means that the properties of the DO type are known when setting the adaptation.
- Parameter p1544:**driveData.torqueReductionGranularity** or **linearMotorDriveData.forceReductionGranularity** is only adapted when a 1xx telegram is set.
  Otherwise no adaptation, since it is not relevant or with DO vector, DO TM 41 and DO DCM also not available.
- If parameter p1082 (maximum speed) is not available on a DO, then the maximum speed is set equal to the reference speed in SIMOTION (**driveData.maxSpeed** = driveData.nominalSpeed or **linearMotorDriveData.maxSpeed** = linearMotorDriveData.nominalSpeed).
  This is the case, for example, for DO TM41 or TO DCM.
- If parameter p1520 (maximum torque) is not available on a DO (e.g. DO DCM), then the maximum torque/force is set equal to the reference torque/force on the TO (**driveData.maxTorque** = driveData.nominalTorque or **linearMotorDriveData.maxForce** = linearMotorDriveData.nominalForce). The torque values are not adapted when interconnecting to DO TM41.

Adapted encoder data

The encoder type set in the axis configuration, incremental or absolute, is checked during the adaptation.

The measuring system setting, linear or rotary, is adapted.

Table 4-335 Incremental rotary encoder

Encoder characteristic data	Parameter	Configuration data element
Increments/revolution	p0979[2]	**incEncoder.incResolution**
Fine resolution	p0979[3]	**incEncoder.incResolutionMultiplierCyclic**

Table 4-336 Incremental linear encoder

Encoder characteristic data	Parameter	Configuration data element
Grid spacing	p0979[2]	**resolution.distance**
Fine resolution	p0979[3]	**resolution.multiplierCyclic**

Table 4-337 Absolute rotary encoder

Encoder characteristic data	Parameter	Configuration data element
Increments/revolution	p0979[2]	**absEncoder.absResolution**
Fine resolution	p0979[3]	**absEncoder.absResolutionMultiplierCyclic**
Fine resolution for absolute value in Gn_XIST2	p0979[4]	**absEncoder.absResolutionMultiplierAbsolute**
Data width of absolute value without fine resolution	p0979[2] and p0979[5] *)	**absEncoder.absDataLength**

*) The data width of the absolute value (without fine resolution) is a result of the sum of the bits for representing the number of encoder pulses and the bits for representing the maximum number of revolutions that can be registered by the encoder according to the type plate.

Example:
4,096 pulses/rev (= 12 bit) and maximum of 4096 revolutions that can be registered results in a 12 + 12 = 24-bit data width of the absolute value.

Table 4-338 Absolute linear encoder

Encoder characteristic data	Parameter	Configuration data element
Grid spacing	p0979[2]	**resolution.distance** (**resolution.multiplierCyclic** has no significance for a linear absolute encoder)
Fine resolution	p0979[3]	**absEncoder.absResolutionMultiplierCyclic**

Encoder characteristic data	Parameter	Configuration data element
Fine resolution for absolute value in Gn_XIST2	p0979[4]	**absEncoder.absResolutionMultiplierAbsolute**
Data width of absolute value without fine resolution	p0979[2] and p0979[5] *)	**absEncoder.absDataLength**

*) The data width of the absolute value (without fine resolution) is a result of the sum of the bits for representing the number of encoder pulses and the bits for representing the maximum number of revolutions that can be registered by the encoder according to the type plate.

Example:
4,096 pulses/rev (= 12 bit) and maximum of 4096 revolutions that can be registered results in a 12 + 12 = 24-bit data width of the absolute value.

Communication and control word / status word

Communication with the drive via PROFIdrive

The communication with digital drives using PROFIBUS/PROFINET is made in accordance with the PROFIdrive V4 specification and the application classes 1 to 4 (class 4 both with and without DSC).

The ADI4 and IM 174 modules can be used to connect axes with analog interface to the system. The system uses the standard 3 telegram in accordance with the PROFIdrive profiles for the ADI4 and IM174 modules. Consequently, in this case, the axis TO sees a real axis with digital drive coupling.

For axes with digital drive coupling using the PROFIdrive telegram, the maximum velocity is twice the reference velocity.

Control word

Drive enables are set in conformity with PROFIdrive Profile V3.1 via control word STW1.

Control bits in control word STW1

The Axis technology object takes over control of the control word with the **_enableAxis()** and **_disableAxis()** commands.

The system uses **_enableAxis()** or **_disableAxis()** to set the control bits in STW1 bit 0 - STW1 bit 6. Also refer to the following table for the semantics of the STW1 bit 0 – STW1 bit 6 control word bits.

_enableAxis() (**enableMode**=DRIVE) is used to set Bit4 - Bit6 in STW1, while **_enableAxis()** (**enableMode**:=POWER) is used to set Bit0 - Bit3 in STW1.

enableMode=ALL is used to set the DRIVE and POWER bits.

Table 4-339 Meaning of the control word bits STW1 bit 0 – STW1 bit 6

STW1	Bit = 0 meaning	Bit = 1 meaning	Notes
Bit 0	OFF, Brake on ramp-function gen. (OFF 1)	ON	Brake along the ramp generator deceleration ramp followed by pulse suppression
Bit 1	Coast stop (OFF2)	No coast stop (no OFF2)	Pulse suppression
Bit 2	Quick stop (OFF3)	No quick stop (no OFF3)	Brake along the OFF3 deceleration ramp followed by pulse suppression
Bit 3	Disable operation	Enable operation	-
Bit 4	Disable ramp generator	Enable ramp generator	-
Bit 5	Freeze ramp generator	Unfreeze ramp generator	-
Bit 6	Disable setpoint	Enable setpoint	-

You can find a detailed description of the bits in status and control words in the SINAMICS List Manuals.

The control word from the drive protocol is displayed in the system variable **driveData.STW** (STW1 corresponds to STW[0]).

The states are also displayed in the SCOUT at **Drives - Drive_x - Diagnosis - Control/status words**.

Stop modes with PROFIdrive – Profile Drive Technology

- STW1 bit 0 = 0 (OFF1): **Ramp stop**
  - The drive travels to zero velocity with the deceleration that can be defined on the drive.
  - The stopping process can be interrupted and the drive reactivated.
  - After stopping, a pulse suppression is carried out and the status changes to ready to start.
- STW1 bit 1 = 0 (OFF2): **Coast stop**
  - The drive immediately goes to pulse suppression and the status changes to switch-on disable.
    The drive coasts to a standstill.
- STW1 bit 2 = 0 (OFF3): **Quick stop**
  - The drive travels to zero velocity with the torque limit that can be defined on the drive.
  - The stopping process cannot be interrupted.
  - After stopping, a pulse suppression is carried out and the status changes to switch-on disable.

Status word

Status word ZSW1 contains status information of the drive in accordance with PROFIdrive profile V3.1.

Table 4-340 Meaning of the status word bits ZSW1 bit0 – ZSW1 bit 15

ZSW1	Bit = 0 meaning	Bit = 1 meaning	Notes / parameters
Bit 0	Not ready to power up	Ready to start Power supply on, electronics initialized, line contactor released if applicable, pulses inhibited.	BO: r0899.0
Bit 1	Not ready for operation Reason: No ON command present	Ready for operation Voltage at Line Module, i.e. line contactor closed (if used), field is being built up.	BO: r0899.1
Bit 2	Operation inhibited	Operation enabled Enable electronics and pulses, then ramp up to active setpoint	BO: r0899.2
Bit 3	No fault active No active fault in the fault buffer.	Fault active The drive is faulty and, therefore, out of service. The drive switches to "switching on inhibited" once the fault has been acknowledged and the cause has been remedied. The active faults are stored in the fault buffer.	BO: r2139.3
Bit 4	Coasting down active (OFF2) An OFF2 command is active.	No OFF2 active	BO: r0899.4
Bit 5	Quick stop active (OFF3) An OFF3 command is active.	No OFF3 active	BO: r0899.5
Bit 6	No "Power-on inhibit" Switching on is possible	Power-on inhibit A restart is only possible by means of OFF1 and then ON.	BO: r0899.6
Bit 7	No alarm active No active alarm in the alarm buffer.	Alarm active The drive is operational again. No acknowledgment necessary. The active alarms are stored in the alarm buffer.	BO: r2139.7
Bit 8	Setpoint/actual speed monitoring not within tolerance band	Setpoint/actual speed monitoring within tolerance band Actual value within a tolerance band; dynamic overshoot or undershoot for t < t_{max} permissible, e.g. $n = n_{set}\pm$ $f = f_{set}\pm$, etc., t_{max} can be parameterized	BO: r2197.7

ZSW1	Bit = 0 meaning	Bit = 1 meaning	Notes / parameters
Bit 9	Local operation Control only possible on device	Control requested The PLC is requested to assume control. Condition for applications with isochronous mode: Drive synchronized with PLC system.	BO: r0899.9
Bit 10	f or n comparison value not reached.	f or n comparison value reached or exceeded.	BO: r2199.1 **Note:** The message is parameterized as follows: p2141 Threshold value p2142 Hysteresis
Bit 11	I, M or P limit reached or exceeded	I, M or P limit not reached	BO: r1407.7
Bit 12	Holding brake closed	Holding brake opened	BO: r0899.12
Bit 13	Motor overtemperature alarm active	Motor overtemperature alarm not active	BO: r2135.14
Bit 14	Actual speed < 0	Actual speed > = 0	BO: r2197.3
Bit 15	Alarm, converter thermal overload The overtemperature alarm for the converter is active.	No alarm active	BO: r2135.15

The status word from the drive protocol is displayed in the system variable **driveData.ZSW** (ZSW1 corresponds to ZSW[0]).

See also

Drive communication based on DPV1 services (Page 3059)

Alarm reactions (Page 3181)

Specific removal of drive enables in the event of a fault

In the event of a fault (local alarm reaction RELEASE_DISABLE), all drive enables are removed by default.
A setting can also be made to remove drive enables individually.
To access the **Functions - Response to alarm** dialog, select the *Axis - Configuration* menu and click the *Change* ... button under Functions.

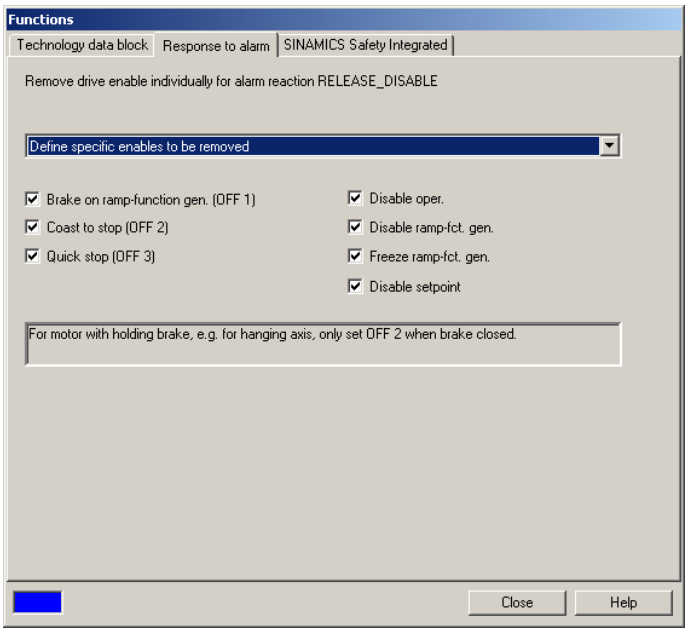


Figure 4-808 Settings for the drive in SCOUT

The bits selected in the screen form will be reset in the control word for the RELEASE_DISABLE local alarm response.

For the local RELEASE_DISABLE alarm response, all enables to the drive will be removed for system setting. Before the enables in STW1 bit 0 - STW1 bit3 are removed, status word ZSW1 bit 10 = 0 (n = 0 in the drive) will be awaited in accordance with the general state diagram in SIMOTION (see figure below).
This system setting can be changed in the **TypeOfAxis.driveControlConfig.releaseDisableMode** configuration data as well as in the dialog box.

Note

At a minimum, the canceling of one enable from bit 0 to bit 6 must be set.

If the value 0 is entered in **releaseDisableMode** in bit 0 to bit 6, all enables are canceled.

See also

Alarm reactions (Page 3181)

Brake control

The brake control is required for operating "hanging axes", for example. The integral brake control of the drives is used for this purpose.
The brake control must be taken into account when enabling the drive and revoking the enable signal.

System support (from SIMOTION V4.4)

The configuration data element **TypeOfAxis.DriveControlConfig.disableModeSpecification** can be used to switch off the drive via AUS3 by default if OFF1_AND_OFF3_MODE is set. OFF2 is therefore only set for the drive when e.g. the brake is closed.
This setting is also the standard setting for newly created axes.

When the enable signals are revoked (via **_disableAxis()** or the RELEASE_DISABLE alarm response), the system behaves as follows if OFF1_AND_OFF3_MODE is set:

1. OFF1 and OFF3 are canceled first when the axis is switched off.
   - The OFF3 ramp is therefore active.
   - STW1 bit0 is already 0 (power-on inhibit canceled)
2. After the drive has transitioned to S1 or S2 as per the PROFIdrive General State Diagram, the remaining enable signals for switching off are canceled.

This functionality is independent of the brake control configuration in the drive and is active even when the enable signals are canceled via the axis control panel.

Directly and exclusively setting OFF2 is always effective.

Extended brake control with SINAMICS

In the case of extended brake control with SINAMICS, the ready signal "Operation enabled" is sent to SIMOTION via status word 1 (system variable **driveData.ZSW[0]** Bit 2) after the brake opening time has expired (p1216). This means the step enabling condition following **_enableAxis()** is not met until the brake opening time has expired.

Please note that when using the extended brake control, the feedback signal "Brake released" is not sent to the ZSW[0] Bit2 (operation enabled), and instead only the brake opening time is taken into account.

When implementing a brake control in the drive, the enable signals must be specifically revoked in the case of **_disableAxis()** and the RELEASE_DISABLE local alarm response (see above), e.g.:

- First, revoke OFF3 (STW1 bit 2)
- When the brake is closed, disconnect the power (revoke STW1 bit 1)

For information on defining the reaction to technological alarms, refer also to Chapter Adjustable response to RELEASE_DISABLE (Page 3185).

Specify control bits via the application (without TO check) [expert]

As of V3.2, it is also possible to specify Bit0 - Bit6 in STW1 specifically via **_enableAxis()** (enableMode:=BY_STW_BIT) and **_disableAxis()** (disableMode:=BY_STW_BIT). Each bit to be set/canceled is then specified in the STWBitSet parameter. For this setting, the axis TO checks also for single bit assignment whether the specifications of the PROFIdrive state machine are observed.

If the axis is switched on from the S1 state, in accordance with PROFIdrive, bit 0 of the initial state 0 for STW1 is required (pulse suppression and ready-to-start status). If STW1 bit 0 in state S1 is not set to 0 (as result of enables being canceled using single-bit assignment or the cancelation of some enables with the RELEASE_DISABLE alarm reaction), STW1 bit 0 must be set to 0 by means of single-bit assignment in the **_disableAxis()** command or **_disableAxis()** with disableMode:=ALL. If this is not the case, a switch on of the axis will be rejected with alarm 20005: Type 1 reason 0x0100h (control signals to the PROFIdrive state machine specified incorrectly).

The status in **actorMonitoring.power** is displayed as specified in STW1 in versions up to and including V3.1. As of V3.2, it is displayed as per Bit0 - Bit2 in ZSW1.

The status indication in **actorMonitoring.driveState** is made in accordance with the specifications in STW1 bit 4 - STW1 bit 6, and will not be derived from the drive status.

The control word and the status word from the drive protocol are indicated in the **drivedata.stw** and **drivedata.zsw** system variables.

n-actual from the drive protocol is indicated in the **actorData.actualspeed** system variable (as of V4.0).

Specify control bits in STW1 via the application (without checking technology object) [expert]

As of V4.1 SP2, the **_disableAxis()** (**disableMode**:=STATE_MACHINE_CONTROL_BY_APPLICATION) setting can be used to directly specify the control bits in STW1 bit 0 - STW1 bit 6 using the **_setAxisSTW()** command without the TO testing correctness in accordance with the PROFIdrive profile state machine.

This makes it possible to switch a drive on and off that does not behave conform to the PROFIdrive profile state machine. A traversing of the drive using the TO or a motion variable generation using the TO for the drive is not possible in this status.

Another possible use is, for example, the stopping of large drives with closed brake in the magnetized state. If the speed controller enable is removed, after the re-setting of the speed controller enable, it is possible to continue motion again without switching the drive on again and so needing to pass through the S1 to S4 states.

The position controller must be switched to 'tracking' (**servoControlMode**=INACTIVE). This setting places the **control** indicator variable to INACTIVE, i.e. no movement commands can be performed. For independent drive transitions, e.g. S4->S5, no 20005 alarm will be generated in this state. The STATE_MACHINE_CONTROL_BY_APPLICATION mode is switched off with the **_enableAxis()** / **_disableAxis()** commands, with another setting of the mode and with **_resetAxis()**.

In the STATE_MACHINE_CONTROL_BY_APPLICATION mode,

- The **actorMonitoring.driveState** and **actorMonitoring.power** system variables indicate INACTIVE.
- The setting/resetting of the STW1 bit 0 - STW1 bit 6 bits is permitted directly using the **_setAxisSTW()** command.
- A fault message for the drive can be reset only using the TO control with **_resetAxisError()**. The control bit STW1 bit 7 (reset fault memory / fault acknowledge) remains in TO administration, a handling of STW1 bit 7 using **_setAxisSTW()** is not possible.
  This means a fault is reset uniformly using the TO, whereby the acknowledgement can be made using SCOUT/HMI or the user program.
- The TO axis no longer performs any monitoring for drive failure / independent disable of the drive. Other monitoring functions remain active, e.g. encoder monitoring, sign-of-life monitoring, drive error 20005: Reason 1 will be signaled and entered in the diagnostic buffer.
- All alarms are performed with RELEASE_DISABLE response as OPEN_POSITION_CONTROL response.

When the STATE_MACHINE_CONTROL_BY_APPLICATION mode is exited and so further handling of the state machine continues using the TO,

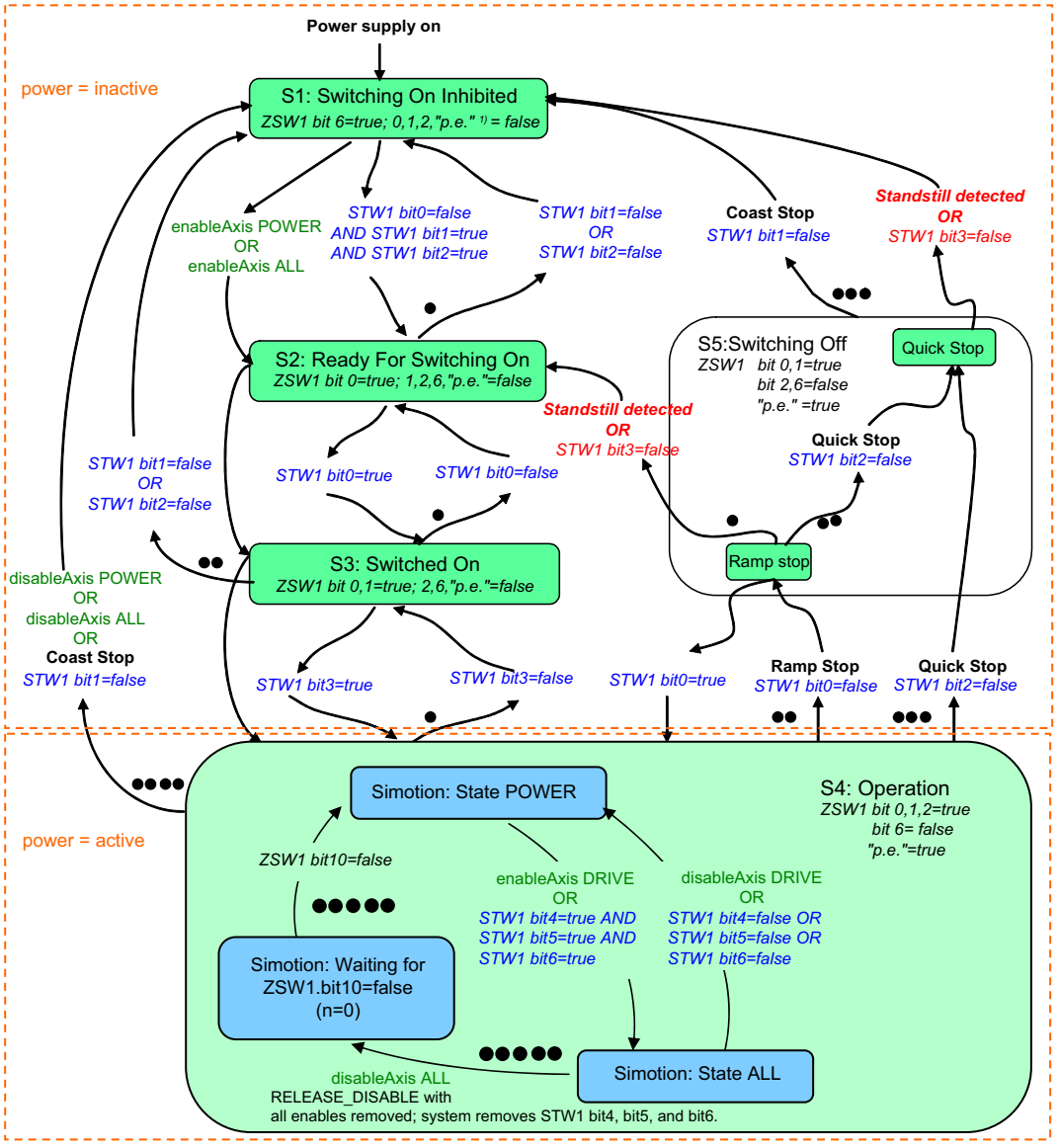
- The TO uses the control signals that were transferred last to the drive.
- When the mode is disabled, the current status of the state machine will be assumed. If the axis is activated in state S4, e.g. with **_enableAxis()**, no reactivation of the drive is possible. Consequently, the drive should be in a defined PROFIdrive S1-S4 state with regard to the drive, otherwise the TO will initiate the technological alarm 20005: type 1 reason 0x04 and the enables for the drive removed.

The status is indicated with the **actorMonitoring.stateMachineControl**=APPLICATION system variables.

See also

Alarm reactions (Page 3181)

Status diagram - PROFIdrive General State Diagram



STW1 bit x,y = These control word bits must be set by the controller.
ZSW1 bit x,y = These status word bits indicate the current status.
1) Abbreviation: "p.e." = "pulses enabled" optional, not supported by SIMOTION
Standstill detected: an internal result of a Stop operation
●●●…: The number of points at the illustrated transitions determines the priority.
The more points, the higher the priority.
- With SIMOTION V3.2 and higher, two variations are possible for the parameter assignment of the _enableAxis() command:
POWER/DRIVE/ALL: The StateMachine is controlled by the Axis technology object.
BY_STW_BIT: The StateMachine is controlled by the user program.
- The drivestate system variable is active in all states S1-S4 if STW1 bits 4 to 6 are set to true.

Figure 4-809 General state diagram in SIMOTION

SIMOTION system behavior with a drive-autonomous OFF3 ramp

When an OFF3 ramp is triggered (by the SS1 or SS2 safety functions, for example), the behavior with regard to the shutting down of the SINAMICS drives during the OFF3 ramp varies dependent upon certain settings on the Axis technology object:

A) Behavior as of SIMOTION V4.2:

As of SIMOTION V4.2, drive-autonomous motion (triggered by SS1, for example), can be clearly identified even if the *Safety Integrated Extended Functions* have not been set; in other words, even if **TypeOfAxis.TechnologicalData.driveSafetyExtendedFunctionsEnabled**=NO.
This requires that the configuration data item **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** is set to YES (system setting when new axes are set up).
This means that drive enables can be canceled during the drive-autonomous state.

If **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** is set to COMPATIBILITY_MODE (for axes from projects converted upwards), the behavior from V4.1 applies. If DSDB (Page 3087) is used, this setting is not allowed (**TypeOfAxis.TechnologicalData.numberOfDriveSafetyDataBlock**>0).

B) Behavior in SIMOTION V4.1:

If a drive-autonomous motion is triggered, it is not possible to cancel the drive enables during the autonomous state unless the Extended Safety Functions are set (**TypeOfAxis.TechnologicalData.driveSafetyExtendedFunctionsEnabled**=NO).

A drive-autonomous motion can be triggered by selecting SS1, for example. The drive enables cannot be removed during the drive-autonomous motion - neither through the user command **_disableAxis()** nor through an error on the axis with the local reaction RELEASE_DISABLE.

This is to be taken into account when designing software and hardware limit switches among other things.

If the Extended Functions are set (that is to say if **TypeOfAxis.TechnologicalData.driveSafetyExtendedFunctionsEnabled**=YES and **TypeOfAxis.DriveControlConfig.pulsesEnabled** are correctly configured), it is possible to cancel the drive enables during the autonomous state (e.g. by using the **_disableAxis()** command and the RELEASE_DISABLE error response).

For **TypeOfAxis.DriveControlConfig.pulsesEnabled** configuration, see Pulse enable evaluation (when standard settings are deactivated) (Page 3107).

This needs to be taken into account, for example, in cases where a control has already been configured for a V4.1 CPU and an additional control is then added via PROFIsafe or TM54F to the OFF3 ramp via a terminal or to the Safety Integrated Basic Functions via terminals (mixed operation).

To avoid following error limit 50102 during the OFF3 ramp in this case, the following procedure is recommended:

1. Cyclic evaluation of the system variables
   - driveData.driveSafetyExtendedFunctionsInfoData.state.1=TRUE
   - driveData.driveSafetyExtendedFunctionsInfoData.state.2=TRUE
   - driveData.ZSW[0].5=FALSE
2. If a condition is fulfilled switch to follow-up mode.
   (e.g. with the command **_move()** at velocity = 0 and movingMode=SPEED_CONTROLLED)

See also

Safe Stop 1 (SS1)As of SIMOTION V4.1 (Page 3098)

Safe Stop 2 (SS2)As of SIMOTION V4.1 (Page 3100)

Technologies and telegram types

Table 4-341 Technologies and telegram types supported for real axis with digital drive link

Drive	Technology	Telegram type
SIMODRIVE 611U universal SIMODRIVE 611U universal HR	All	1 to 6, 101, 102, 103, 105, 106 [1]
SIMODRIVE POSMO CA/CD	All	1 to 6, 101, 102, 103, 105, 106
SIMODRIVE POSMO SI	All	1, 2, 3, 5, 101, 102, 105
SIMODRIVE sensor isochronous	External encoder	81
SIMODRIVE sensor (PROFIBUS)	External encoder	81
SIMODRIVE sensor (PROFINET)	External encoder	81, 83
MASTERDRIVES MC	All	1 to 6 [2]
MASTERDRIVES VC	Speed-controlled axis	1, 2
MICROMASTER 4xx	Speed-controlled axis	1
SINAMICS S120	All	1 to 6, 83, 102, 103, 105, 106, 125 [3], 126 [3]
SINAMICS Integrated (SIMOTION D)	All	1 to 6, 83, 102, 103, 105, 106, 125 [3], 126 [3]
SINUMERIK ADI4, SIMATIC IM174	All	3

[1] For further details, see also the SIMOTION SCOUT Configuration Manual and the accompanying 611U product description.

[2] For further details, see also the SIMOTION SCOUT Configuration Manual and the accompanying MD MC product description.

[3] As of SIMOTION V4.4 with SINAMICS V4.7

Table 4-342 Telegram types

Telegram type	Brief description
Standard telegrams	
1	n-set interface, 16-bit, without encoder
2	n-set interface, 32-bit, without encoder

Telegram type	Brief description
3	n-set interface, 32-bit, with encoder 1
4	n-set interface, 32-bit, with encoder 1 and encoder 2
5	n-set interface, 32-bit, with DSC and encoder 1
6	n-set interface, 32-bit, with DSC, encoder 1 and encoder 2
81	Encoder telegram with 1 encoder channel
83	n-act. interface, 32-bit, extended encoder telegram with 1 encoder channel
Siemens telegrams	
101	n-set interface
102	n-set interface with encoder 1 and torque limiting
103	n-set interface with encoder 1, encoder 2 and torque limiting
105	n-set interface with DSC, encoder 1 and torque limiting
106	n-set interface with DSC, encoder 1, encoder 2 and torque limiting
125	n-set interface with DSC with spline, encoder 1 and torque limiting
126	n-set interface with DSC with spline, encoder 1, encoder 2 and torque limiting

Information on the activation of technology data blocks can be found in Chapter Technology Data.

See also

Technology data (Page 3018)

Coupling of digital drives

Separation of reference value and maximum value (as of V4.0 through < V4.2)

With V4.2 and higher, the reference/maximum velocity, reference/maximum torque, and reference/maximum force are adapted by the SINAMICS drive during runtime by default. The following is not relevant in this context.

For digital drive coupling, the reference value for sending the speed or velocity to the drive can be configured independently of the maximum value. Alternatively, the maximum speed/velocity can still be used as the reference value. This can be defined accordingly in the **driveData.speedReference** or **linearMotorDriveData.speedReference** configuration data element.

When a new axis is created and coupled with a digital drive, the normalization speed/velocity is pre-selected as the reference value by default in the axis wizard. The maximum speed/velocity can be entered independently.

With SIMODRIVE and MASTERDRIVE, this data must be entered manually. You must make sure that the reference values are the same on the drive and the controller.

In all cases, you still have the option of setting the maximum velocity as the reference velocity.

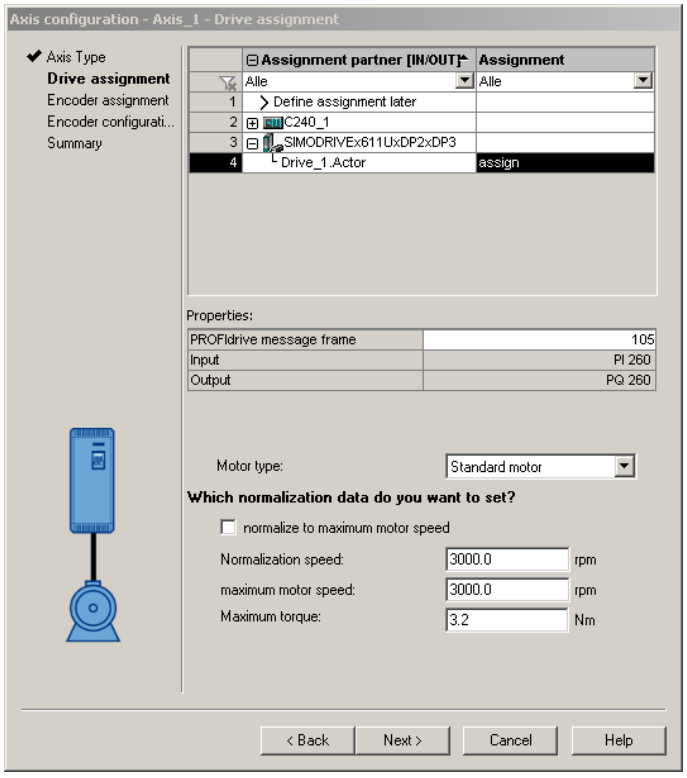


Figure 4-810 Setting the normalization speed and maximum motor speed

In the axis wizard, the drive settings are displayed as default settings.

For a coupling with SINAMICS, you can use the **Data transfer from the drive** button to transfer the **normalization speed**/velocity and the **maximum speed**/velocity from the offline configuration of the drive. If the relevant drive parameters are modified online, they must be uploaded and saved to the project prior to the data transfer.

SINAMICS Safety Integrated Extended Functions

See Overview - Support of SINAMICS Safety Integrated Functions on the Axis technology object (Page 3079).

Drive communication based on DPV1 services

Drive communication based on DPV1 services facilitates acyclic communication with the drive, e.g. making it possible to read and write drive parameters.

For more information, see Drive communication based on DPV1 services (Page 3059).

Time-of-day synchronization SIMOTION - SINAMICS

As of V4.2, SIMOTION features automatic time-of-day synchronization.

The time of day on the SINAMICS drive system is synchronized with the master SIMOTION system and the time ratio between SINAMICS messages and SIMOTION messages is correct.

For firmware versions < V4.2, see Drive communication based on DPV1 services (Page 3059).

The following control units support time synchronization:

- SINAMICS Integrated of the SIMOTION D
- CX32/CX32-2 controller extensions
- SINAMICS S120 CU310, CU310-2, CU320, CU320-2 Control Units connected via PROFIBUS or PROFINET

Requirement: SIMOTION V4.4 and higher

See also

Overview - Support of SINAMICS Safety Integrated Functions on the Axis technology object (Page 3079)

Adjustable response to RELEASE_DISABLE (Page 3185)

Drive communication based on DPV1 services (Page 3059)

Overview of torque limiting via torque reduction (Page 3010)

Setting as a real axis with stepper drive C2xx (V3.2 and higher)

Possible drive types for stepper motors:

- Stepper motor with encoder
- Stepper motor without encoder
  In this case, stepper motor is entered as the encoder mode and the encoder side is not used.

When you select stepper motor without encoder on drive 1, encoder input 1 is assigned automatically since this data is traced back to it within the system. Drive 1 cannot be selected if the encoder input is no longer free.

For stepper motors, you can enter the number of steps and the stepping rate. The speed of rotation is then calculated and displayed.

Stepper drives on IM174 and stepper drives with PROFIBUS interface

The IM174 module is available as the interface for stepper drives for all SIMOTION platforms. From the SIMOTION perspective, stepper drives linked via IM174 behave like digital drive links.

Alternatively, stepper drives can be linked with a PROFIBUS interface if they support the PROFIdrive profile.

You can find more information in the *Distributed I/Os IM 174 PROFIBUS Module* Manual.

See also

Coupling of digital drives (Page 2895)

Setting as a real axis with encoder signal simulation (V4.0 and higher)

With this setting, no drive/actuator is connected via the Axis technology object. An angle signal is output directly to a third-party control via the SINAMICS TM41 peripheral module (encoder simulation).

The output angle signal behaves just like the signal from an incremental encoder.

Enable signal

The enable signal to the drive is activated with **_enableAxis()** (**enableMode**=ALL); the enable is displayed by the system variables **actormonitoring.driveState** = ACTIVE and **actormonitoring.power** = ACTIVE.

See also

Encoder signal output (V4.0 or later) (Page 2984)

Setting a non-exclusive drive assignment (as of V4.1 SP1)

The **typeOfAxis.setpointDriverInfo.InterfaceAllocation** setting is used to assign the drive/actuator of the axis to the drive, either exclusively (in versions up to and including V4.0) or non-exclusively (as of V4.1 SP1). For non-exclusive assignment, you must specify whether this drive interface should be activated when the technology object is ramped up.

With non-exclusive drive assignment, you can interconnect several Axis technology objects with one drive.

The **_enableAxisInterface()** and **_disableAxisInterface()** commands or the actor=YES function parameter are used to activate and deactivate the actuator interface during operation.

The Activated/Deactivated drive interface status is displayed in the **actorMonitoring.output** system variable.

See also

Non-exclusive encoder assignment (as of V4.1 SP1) (Page 2917)

Setting as a real axis without drive (axis simulation)

It is possible to switch a real axis to simulation status even if the drive is not connected. The axis simulation status is displayed in the configuration dialog for the axis under drive and encoder assignment.

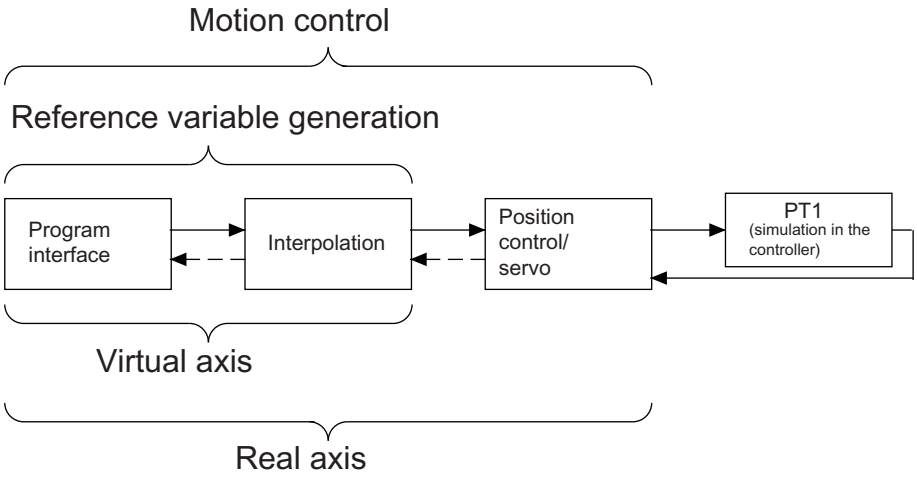


Figure 4-811 Axis simulation functional scheme

The dynamic behavior of the axis is simulated with the total time constant via an internal PT1 element.

The axis is thus available for the user program in the basic functions (e.g. setting of enables, execution of motion commands), but the manipulated variable is not output to the connected or missing drive.

There is no need for precise dynamic modeling of signals.

The simulation of an axis with an "analog output via I/O peripheral" drive assignment is not supported.

The complete axis configuration is kept when setting the simulation status and is still available without any changes after switching back in the expert list to the normal mode status.

Table 4-343 Settings in the configuration data

TypeOfAxis.SetPointDriverInfo.mode	= SIMULATION
TypeOfAxis.SetPointDriverInfo.outputNumberOnDevice	= 0
TypeOfAxis.NumberOfEncoders.Encoder_1.encoderIdentification	= SIMULATION
TypeOfAxis.NumberOfEncoders.Encoder_1.DriverInfo.encoderNumberOnDevice	= 0
TypeOfAxis.NumberOfDataSet.DataSet_x.ProcessModel.T1 (as of V4.0)	Time constant T1
TypeOfAxis.NumberOfDataSet.DataSet_x.ProcessModel.T2 (as of V4.0)	Time constant T2

Comments

- All real axis types can be set to SIMULATION.
- The process response/axis response is simulated from **processModel.T1** and **processModel.T2** via the PT1 element with the total time constant (T1+T2).

- Homing to the encoder zero mark and the external zero mark is not possible in Axis simulation mode.
- Output cam functions are executed.
- Only measuring input via time stamp is possible.
- If an axis with the DSC setting is set to SIMULATION, the position controller is calculated in the position control cycle clock in the control/simulation driver, i.e. it may be necessary to cancel the gain and adjust the following error monitoring.
- Unlike program simulation, the drive does not need to be present and the position controller remains active with simulated drive behavior.
- The simulation mode is provided for actuators and sensors. Technology data (such as Technology Data Block TDB, Safety Information Data Block SIDB, Drive Safety Data Block DSDB, etc.) of an axis cannot be simulated. Deactivate them before the axis is switched to simulation mode.
- The **_setAxisSTW()** command is not active in simulation mode.

As of SIMOTION V4.2

Axes with symbolic drive allocation

If symbolic allocation is set and if an axis allocated to a SINAMICS drive is to be removed from axis simulation, then this axis must be reallocated to the drive again.

Axes without drive assignment

If the axis is not yet assigned (see also Real and virtual axes (Page 2875)), the logical addresses for the actuator and encoder must be converted.
To do this, set the following configuration data in the expert list to a free and valid address: The address must not be equal to 65535 and the same one must be used for all four items of configuration data, e.g. 15000.

Configuration data:

- **TypeOfAxis.SetPointDriverInfo.logAdressIn**
- **TypeOfAxis.SetPointDriverInfo.logAdressOut**
- **TypeOfAxis.NumberOfEncoders.Encoder_n.DriverInfo.logAdressIn**
- **TypeOfAxis.NumberOfEncoders.Encoder_n.DriverInfo.logAdressOut**

When deactivating the simulation, set each of the values for this configuration data back to 65535.

Note

A script is available to switch the axis simulation on and off for axes with and without symbolic drive assignment. This script can be integrated into a project via the ProjectGenerator (accessible from SCOUT). You can find the script in *SIMOTION Utilities & Applications*, which is part of the scope of delivery of SIMOTION SCOUT.

See also

Enabling/disabling program simulation (Page 3170)

Coupling of digital drives (Page 2895)

4.8.3.5 Encoders and encoder parameters

Overview of encoders and encoder parameters

Sensing of the actual position value can be performed using:

- Motor measuring system (motor encoder)
- Additional direct measuring systems (external encoders), if necessary

The adaptation of relevant drive data is activated automatically as of SIMOTION V4.2 in conjunction with SINAMICS S120 as of V2.6.2. The encoder parameters are adapted automatically.

A direct measuring system measures the technological parameter directly, i.e. without the interference of influences such as torsion, backlash, slip, etc.
This may facilitate improved smoothing of mechanical influences by means of the closed-loop control.

Up to eight encoders can be generated on the axis. All generated encoders are internally active, and the measured values are updated cyclically. It is possible to switch between the encoders via data sets. Whether an encoder is active for closed-loop control is displayed in the **sensorData[n].control** system variable.

Several data sets can be assigned to one axis. These data sets can contain different encoders. A data set is active.
Encoders can be added and parameterized in the Axis technology object under **Configuration** on the **Encoder configuration** tab.

Via the
typeOfAxis.numberOfEncoders.Encoder_1.sensorControlConfig.tolerateSensorDefect setting (V4.0 and higher), it is possible to tolerate the failure of an encoder which is not selected or which is not involved in the closed-loop control. This setting is defined for individual encoders. It is output via **Alarm 20015**.

For more information on data set switchover and encoder configuration, see Data set switchover / encoder switchover (Page 3037).

See also

Data set overview (Page 3037)

Encoder for position

The controller uses an encoder to detect the axis position.

From a technological standpoint, the following encoder types can be differentiated:

- Incremental encoder
  On the controller side, only the difference between two read encoder values is evaluated. Values are read out in the servo cycle clock at equal intervals. In order to determine the mechanical axis position, the axis must be homed each time the system is switched on. Position zero is displayed after it is switched on.
- Absolute encoder
  This encoder supplies the absolute value or, in the case of an absolute value in the PROFIdrive message frame, the absolute value is read one time after the system is switched on. After this, the actual value is processed in the same way as with the incremental encoder.
  The absolute value supplied by the encoder is assigned to the associated mechanical axis position by means of the absolute encoder adjustment. The absolute encoder adjustment occurs only one time, and the controller remembers the compensation value/absolute encoder offset even after it is powered on/off.
  Certain situations, such as an encoder failure, a restart, etc., can require an absolute encoder re-adjustment. For more information, refer to the chapter titled *Absolute encoder homing/ Absolute encoder adjustment*.
  The following absolute encoder types are differentiated:
  - Absolute encoder with absolute encoder setting
    The measuring range of this encoder is greater than the axis traversing range.
    The axis position results directly from the current encoder value since this can be uniquely mapped.
    An offset may be entered - it is not necessary to hold overflows within the controller.
    There are no overflows of the absolute actual value stored when SIMOTION is switched off. The next time it is switched on, the actual position value is generated from the absolute actual value only.
  - Absolute encoder with cyclic absolute encoder setting
    The axis traversing range is greater than the range of values measured by the encoder, and the encoder returns an absolute value within its measuring range.
    The controller also counts the number of measuring ranges internally in order to hold a unique actual axis position beyond the range of measured values.
    When SIMOTION is switched off, the overflows of the absolute actual value are stored in the retentive memory area of SIMOTION. The next time it is switched on, the stored overflows are taken into account for the calculation of the actual position value.
    The actual position of the axis is passed internally in a 64-bit integer variable.
    Example of a single-turn encoder with 4,096 increments:
    The position of the encoder is mapped in bits 0 to 11, and the number of overflows of the encoder range in bits 12 to 63.
    Example of a multi-turn encoder with 4,096 x 4,096 increments:
    The position of the encoder is mapped in bits 0 to 23, and the number of overflows of the encoder range in bits 24 to 63.
    The overall position of the axis is retained after the controller is switched off. When the controller is switched on again, if the actual encoder value does not match the actual position stored in the controller, it will be corrected to a maximum of ± ½ the encoder measuring range.

Note

If the axis/encoder is moved by more than one half the encoder measuring range with the controller switched off, the actual value in the controller no longer matches the real axis.

See also

Overview of homing (Page 2923)

Encoder for velocity

Encoders for detection and display of the speed/velocity can only be applied to speed-controlled axes.

Options are:

- Incremental encoders/absolute encoders with number of increments or pulses/revolution (for electric axes)
- Interval counters (for hydraulic axes)
- Encoders providing the velocity as a direct value in the I/O area (for hydraulic axes)
- Read-out of the speed from the PROFIdrive message frame and provision for technological functionality, e.g. velocity monitoring

Encoder assignment and terminology

The encoder type is specified with the encoder mode.

Table 4-344 Definable encoder mode depending on the encoder type

Encoder mode	Encoder type		
	Absolute encoder	Cyclic absolute encoder	Incremental encoder
PROFIdrive (with adaptation [1]) (as of V4.2)	x	x	x
Endat (encoder data interface)	x	x	x
SSI (synchronous serial interface)	x	x	-
Sinusoidal	-	-	x
Rectangle	-	-	x
Resolver	-	x [2]	x
Analog encoder (value in the I/O area)	x	-	-

[1] The adaptation setting (set by the system by default with V4.2 or higher and SINAMICS) makes changes based on the encoder set in SINAMICS. The encoder resolution and fine resolution are adopted during runtime.

[2] Possible with single-pole-pair resolver only

When the encoder signals are evaluated in an encoder module or drive, the actual values are transferred to the controller in the normalized format according to the PROFIdrive profile. The type of measured value acquisition is dependent on the drive or encoder module, e.g. Sinus, SSI or Endat. The PROFIdrive encoder mode can therefore be set in the controller (as of V4.2).

As of SIMOTION V4.2, the system makes the settings for communication between SIMOTION and SINAMICS drive (encoder). The project setting **Use symbolic assignment** is activated by default for new projects as of V4.2. Telegrams are set automatically. Encoder characteristics such as fine resolution, grid spacing, and the absolute value for data width are adapted automatically when the system ramps up.

For encoder data, refer to the data sheet or type plate of the encoder. With SINAMICS, encoder data can be transferred from the drive.

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

Encoder pulses per revolution

The encoder pulses per revolution is specified on the encoder type plate as the number of signal periods per revolution (incremental encoder: Pulses/rev; absolute encoder: Pulses/rev; resolver: Number of pole pairs (for SINAMICS and MASTERDRIVES)).

Configuration data:

- **AbsEncoder.absResolution**
- **IncEncoder.incResolution**

Grid line spacing (linear encoder system)

The grid line spacing is specified on the type plate of the encoder as the distance between lines on the linear measuring system (linear scale).

Configuration data:

- **Resolution.distance**

Fine resolution

The fine resolution of the actual value is the interpolation result of a signal period of an encoder pulse.

The fine resolution steps are generated by the measuring electronics from the raw signal of the encoder pulses. Factors as a multiple of 2 are possible.

Example:

- A rectangle signal has a fine resolution of 1
- Two rectangle tracks (TTL signal) offset by 90° have a fine resolution of up to 4
- Depending on the measuring electronics, a sinusoidal signal can have any fine resolution, in principle, e.g. 2,048

Depending on the defined encoder type, the default value 0 is interpreted differently in SIMOTION (see table: *Default settings for fine resolution in SIMOTION*).

In SIMOTION, the multiplication factor is specified, rather than the shift factor/number of bits (x).

The actual value, including the fine resolution, is indicated in the **sensorData.incrementalPosition** system variable.

Configuration data:

- **AbsEncoder.absResolutionMultiplierCyclic**
- **IncEncoder.incResolutionMultiplierCyclic**

Data width of absolute value (without fine resolution) for absolute encoders

The data width of the absolute value (without fine resolution) is a result of the sum of the bits for representing the number of encoder pulses and the bits for representing the maximum number of revolutions that can be registered by the encoder according to the type plate.

Example:

4,096 pulses/rev (= 2^{12}) and a maximum of 4,096 revolutions that can be registered yields a 12 + 12 = 24-bit data width of the absolute value.

Configuration data:

- **AbsEncoder.absDataLength**

Fine resolution of absolute value in Gn_XIST2.

This parameter for the format of the Gn_XIST_2 is only relevant for encoders via PROFIdrive telegram (for more information, see *Encoder interconnection via PROFIdrive telegram*).

Configuration data:

- **AbsEncoder.absResolutionMultiplierAbsolute**

The fine resolution of the absolute value in Gn_XIST2 must be less than or equal to the fine resolution of the absolute value in Gn_XIST1.

Default setting for fine resolution.

Depending on the encoder mode, the default settings are evaluated by the system as described in the table below. The default settings are used if 0 is assigned to the value.

Table 4-345 Default settings for fine resolutions in SIMOTION

Encoder type	Encoder mode	Fine resolution (Gn_XIST1)	Fine resolution on the absolute value (Gn_XIST2)
Onboard C2xx			
Incremental encoder	Rectangle	4	-
	Stepper motor	1	-
Absolute encoder	SSI	1	1
Encoder in PROFIdrive axis telegram (applies to SINAMICS, SIMODRIVE 611U and MASTERDRIVES)			
Incremental encoder	Rectangle	2048	-
	Sinusoidal	2048	-
	Resolver	2048	-
	Endat	2048	-
Absolute encoder	Endat	2048	512
	SSI	1	1
...Cyclic absolute	Resolver (only pole pair number 1 possible)	2048	512
	Endat	2048	512
	SSI	1	1
PROFIBUS absolute encoder in PROFIdrive encoder telegram			
Absolute encoder	SSI	$2^{(32 - \text{Number of data bits})}$	1

These settings are aligned to the default parameter settings of the corresponding Siemens devices. If the behavior is different, alignment with the encoder should be performed by making a corresponding entry in the configuration data of the technology object or in the parameters of

the drive or encoder. Depending on the device it may be necessary to assign the corresponding parameters in the drive or encoder with the value of the exponent (shift factor).

Device-specific features for Masterdrives with Endat encoders:

For Masterdrives with Endat encoders, Endat or SSI can be selected in the encoder mode. However, the fine resolution must always be configured in the axis wizard of the Axis or External Encoder technology object differently from the default settings (see Encoder list (Page 2906)).

Default setting:

- Fine resolution (**~Encoder.~ResolutionMultiplierCyclic**) = 0
- Fine resolution of the absolute encoder in Gn_XIST2 (**AbsEncoder.absResolutionMultiplierAbsolute**) =0
- If encoderMode=PROFIDRIVE, the values are evaluated as available, since the correct values are adapted by the drive. A default setting of 0 is not permitted for this setting.

See also

Encoder interface as a direct value in the I/O area (Page 2911)

Encoder list (Page 2906)

Encoder list

For the most up-to-date list of encoders you can use with SIMOTION in conjunction with SINAMICS, SIMOVERT-MASTERDRIVES, and SIMODRIVE 611U, go to **http://support.automation.siemens.com/WW/view/de/18769911**.

The encoder list is also documented under *FAQs > Drives > Parameters of the connectable encoders* in the *SIMOTION Utilities & Applications* and in the online help (look for encoder parameter assignment in the index). *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

Onboard encoder interface on SIMOTION C2xx

Incremental encoders with TTL signal and absolute encoders with SSI protocol can be connected directly to C230-2 or C240. (See the C230-2 and C240 operating instructions and Encoder list (Page 2906))

See also

Encoder list (Page 2906)

Encoder interface using the PROFIdrive message frame

As of SIMOTION V4.2, the system sets up communication between SIMOTION and SINAMICS drive (encoder). For SINAMICS drives and encoders with V4.2 and higher, encoder resolution data is transferred directly from the drive during runtime. Telegrams are set automatically.

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

The encoder values are transmitted in the PROFIdrive telegram (see Table *Telegram types* in Section *Setting as a real axis with digital drive coupling*).

Encoder forced values, status values, and actual values are transmitted in the PROFIdrive telegram.

The encoder behavior on SIMOTION is set as represented in the PROFIdrive protocol.

The encoder parameters are defined via the drive wizard during drive configuration (either user-defined or by selecting the encoder).

Encoder parameters that are entered subsequently in the SIMOTION axis wizard must match the encoder parameters in the drive.

Note

For SINAMICS drives with earlier versions than SIMOTION V4.2, it is possible to transfer the encoder parameters from the drive. When assigning encoders in the axis wizard, click **Data transfer from the drive**.

If you are using a DRIVE-CLiQ component with electronic type plate (e.g., SMI motor, DRIVE-CLiQ encoder) you must first upload the parameters from the drive and save them in the project (online commissioning). If the online commissioning is carried out at a later point, you can work with the default settings of the axis wizard in the meantime during offline configuration. Once online commissioning is complete, upload the drive parameters, save them in the project, run the axis wizard again, and perform the **Data transfer from drive** function.

If you change the encoder data in the drive, you must perform an alignment again in the axis wizard.

Further sources of information:

- Encoder list (Page 2906)
- User Manual *SIMODRIVE sensor Absolute encoder with PROFIBUS-DP*
- Operating Instructions *Absolute encoder with PROFINET IO* (Chapter *Operating with SIMOTION*)

Encoder value via PROFIdrive axis telegrams

For further information, refer to the commissioning manuals for the drives.

The first and (if present) second encoder of the PROFIdrive axis telegram can be assigned freely to an External Encoder technology object or to the encoder of an axis.

Encoder value via PROFIdrive encoder telegram 8x

PROFINET/PROFIBUS encoders can be set in accordance with the current specifications of the encoder profile with telegram type 81 and as of V4.2, with telegram type 83. These encoders can be assigned freely. See also **PROFINET/PROFIBUS absolute encoder via PROFIdrive encoder telegram** in the next chapter.

Inconsistent configuration

In the event of errors or inconsistencies between the configuration data in SIMOTION and the parameter settings for the encoder in the drive, a technological alarm is triggered as soon as an online connection is established between the control and the drive/encoder.

As off SIMOTION Runtime V4.4: **Technological alarm 20025** with applicable reason

In SIMOTION Runtime < V4.4: **Technological alarm error 20005: Device type:2, log.address:1234 faulty. (Bit:0, reason: 0x80h)**

For PROFIdrive encoders and encoders on the axes according to PROFIdrive, a comparison of the parameter assignment takes place via the following drive/encoder parameters:

P979 (SensorFormat) according to PROFIdrive, which contains information about the type, resolution, and shift factors.

For drives or encoders that do not support parameter P979, the configuration data is evaluated as valid without alarm message.

Actual value Gn_XIST1

The incremental actual value is transferred cyclically with the defined fine resolution in Gn_XIST1. The incremental actual value in Gn_XIST1 is steadily continued according to the actual value change and reset when the data width of Gn_XIST1 is exceeded. If operating with incremental and absolute encoders, the control evaluates the incremental actual value in Gn_XIST1 according to the settings made for encoder pulses per revolution and fine resolution, or grid line spacing for linear scaling.

When the controller is switched on, the fine resolution value within one encoder signal period is indicated correctly in Gn_XIST1. The initial value for the number of signal periods is set by the drive/encoder, and the actual value can then be steadily continued from this initial value.

In the PROFIdrive profile, the fine resolution is given as "shift factor" (x).

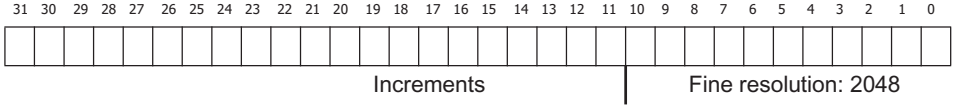


Figure 4-812 Example composition of the 32-bit encoder data of the cyclic actual value Gn_XIST1

Example of for an encoder with number of encoder pulses = 2048 (data width, 11 bits)

The fine resolution in SIMOTION in the **Inc/AbsResolutionMultiplierCyclic** configuration data element is set to the default setting 0 and is thus evaluated as a default fine resolution of 2048 (the default value depends on the encoder mode setting; see Table *Default settings for fine resolution in SIMOTION*).

SIMODRIVE 611U:

Table 4-346 Settings

SIMOTION		611U	
Encoder pulses per revolution [1]	=2048	P1007	=2048
Fine resolution [2]	=0 (≡ 2048)	P1042	=11

[1] Inc/AbsEncoder.Inc/AbsResolution

[2] Inc/AbsEncoder.Inc/AbsResolutionMultiplierCyclic

SINAMICS:

Table 4-347 Settings

SIMOTION		SINAMICS	
Encoder pulses per revolution [1]	=2048	P408	=2048
Fine resolution [2]	=0 (≡ 2048)	P418	=11

[1] Inc/AbsEncoder.Inc/AbsResolution

[2] Inc/AbsEncoder.Inc/AbsResolutionMultiplierCyclic

Note the information about the SINAMICS alignment.

Actual value Gn_XIST2

If the positions for the measuring input or homing functions are transferred to Gn_XIST_2 (n = 1 or 2, number of encoder), they are transferred with the fine resolution defined for the encoder. When the absolute value is read, the value in Gn_XIST_2 is evaluated based on the settings for the data width of the absolute value (without fine resolution) in **AbsEncoder.absDataLength** and the fine resolution absolute value in Gn_XIST2 is evaluated in **AbsEncoder.absResolutionMultiplierAbsolute**.

The fine resolution of the absolute value in Gn_XIST2 indicates the fine resolution factor included in the absolute value transfer. This can match the fine resolution of the actual value, but it can also be smaller, for example, if the 32-bit data width in Gn_XIST2 is not sufficient for the entire fine resolution factor as a result of the data width of the absolute value (without fine resolution).

Example:

Encoder pulses per revolution = 2048 (11 bits) and multi-turn resolution of 4,096 revolutions (12 bits)

Thus, the data width of the absolute value without fine resolution is 11 bits + 12 bits = 23 bits.

Therefore, 9 bits remain for the fine resolution in Gn_XIST2 (32 bits - 23 bits = 9 bits). The setting 0 for the fine resolution of the absolute value in Gn_XIST2 is thus evaluated by the system as 512 (= 9 bits).

Table 4-348 Setting the encoder data

Encoder pulses per revolution [1]	2048
Data width of absolute value (without fine resolution) [2]	23
Fine resolution of absolute value in Gn_XIST2 [3]	0 (= 512)
Fine resolution [4]	0 (= 2048)

[1] AbsEncoder.AbsResolution

[2] AbsEncoder.absDataLength

[3] AbsEncoder.absResolutionMultiplierAbsolute

[4] AbsEncoder.AbsResolutionMultiplierCyclic

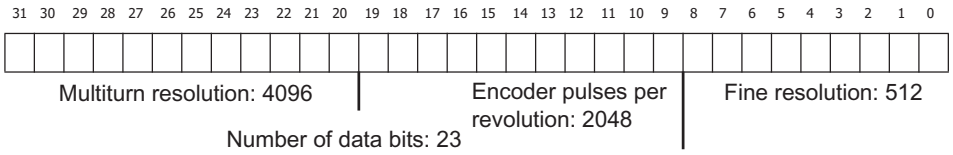


Figure 4-813 Example composition of the 32-bit encoder data of the absolute actual value Gn_XIST2

The number of bits resulting from the data width of the absolute value (without fine resolution) and the number of data bits for the fine resolution of the absolute actual value must not exceed 32. If it is less than 32, leading zeroes are added in Gn_XIST2.

Resolver in PROFIdrive axis telegram

For SINAMICS and MASTERDRIVES, parameters are assigned for the pole pair number of the resolver rather than the encoder pulses per revolution (example: 8-pole resolver = 4 pole pairs → input value = 4).

With SIMODRIVE, parameters are assigned for the encoder pulses per revolution based on parameter P1011.2

As of V4.1 SP1, the resolver with pole pair number 1 is supported as an absolute encoder with the cyclic absolute setting. (encoder pulses per revolution = 1, data width of the absolute value = 0, default value evaluation: fine resolution = 2048, fine resolution Gn_XIST2 = 512)

When a 1-pole resolver is used as Endat encoder, the p418(XIST1) and p419(XIST2) parameters must be set to "11" to prevent information loss of the absolute position. (Settings on the axis: absolute encoder, cyclic absolute, Endat, line count = 1, fine resolution = 2048, fine resolution absolute value = 2048, data width = 0)

See also the Encoder list (Page 2906).

PROFINET/PROFIBUS absolute encoder via PROFIdrive encoder telegram

The data width of the encoder value in the technology object configuration data in SIMOTION must match the parameter settings for the PROFINET/PROFIBUS absolute encoder in HW Config.

See also the Encoder list (Page 2906).

Example:

Parameter settings of a PROFIBUS absolute encoder in HW Config with 24-bit data width of the absolute value.

The 'SIMODRIVE isochronous sensor' PROFIBUS absolute encoder in HW Config is defined according to the default setting for 24-bit data width and encoder pulses per revolution of 4096: measuring steps per revolution = 4096
24-bit data width for the total resolution yields 0x01000000 (32-bit HEX number). This number, represented separately in HighWord and LowWord, equals 0x0100 in the HighWord and 0x0000 in the LowWord. The decimal values of these two parts (0x0100 = 256 decimal) are to be entered as follows:
total resolution (high) = 256
total resolution (low) = 0

This results in the following consistent configuration for the technology object:
the encoder value is transferred left-justified to Gn_XIST1; the unused bits of the fine resolution are set to 0 according to PROFIdrive, but must be specified in the fine resolution of the actual value. This results in a fine resolution of 8 bits (32 bits - 24 bits = 8 bits) (2^8 = 256 as a factor).

According to the setting above, the absolute value in Gn_XIST2 has a right-justified alignment and therefore a fine resolution of the absolute value in Gn_XIST2 of 0 bits ($2^0 = 1$ as a factor).

Encoder via PROFIdrive axis telegram on ADI4 and IM174

At least one electric or hydraulic axis must be configured on ADI4/IM174.

The defined update rate (BaudRate) for SSI encoders must be supported by the encoder.

See also the Encoder list (Page 2906).

For more information about configuration and operation, refer to the *ADI4 - Analog Drive Interface for 4 Axes Manual* and *Distributed I/Os IM 174 PROFIBUS Module Manual*. These documents are provided on the *SIMOTION Documentation* DVD.

Note

An encoder in a PROFIdrive axis telegram can only be used for an external encoder TO or for the encoder of a TO axis in cyclical operation if a TO axis is also created on the PROFIdrive telegram and is not deactivated.

See also

Coupling of digital drives (Page 2895)

Encoder list (Page 2906)

Encoder interface as a direct value in the I/O area

Encoders can be used that

- Provide actual value information directly as absolute value in the input/output area.
- Provide a counter value in the peripheral area (as of V4.0).
- Provide an actual velocity in the peripheral area.

Actual value information directly as absolute value

These encoders must be parameterized and operated as absolute encoders, for example with respect to homing.

The following settings are provided to set the **adaptation to the properties of the measured value**:

- The **justification of the measured value** in the **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.format** configuration data element
  - Signed left-justified (VALUE_LEFT_MARGIN)
  - Signed right-justified (VALUE_RIGHT_MARGIN)
  - Unsigned left-justified (VALUE_LEFT_MARGIN_WITHOUT_SIGN)
  - Unsigned right-justified (VALUE_RIGHT_MARGIN_WITHOUT_SIGN)
- The **data width of the measured value** without the sign bit in the **NumberOfEncoders.Encoder_n.analogSensor.DriverInfo.resolution** configuration data
- The upper limit and lower limit, the **maximum limits of the measured value** in the following configuration data
  - **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.maxValue**
  - **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.maxValue**

Example: Use of an ET 200S, SSI module, or an analog input.

Justification of the measured value

The measured value is mapped to an internal 32-bit wide signed data value of the DINT type in accordance with the justification specification. The mapped value is then tested with actual value limitation against the maximum limits and evaluated using the weighting factor for the **NumberOfEncoders.Encoder_n.AnalogSensor.ConversionData.factor** direct value that specifies the technological resolution or assignment of the LSB.

⚠ WARNING

Weighting factor for analog sensors

The scaling or conversion in the value of the configuration data **ConversionData.factor** for the position of the analog sensor has changed in SIMOTION V4.4.

As of V4.4, the following applies:

The values are entered directly in inches and no longer converted from mm to inches for the configured Inch unit system. This can result in a higher velocity being detected (by a factor of 25) if it is an axis that works in inches.

The following applies for SIMOTION up to V4.4:

The values were entered in mm and then converted to inches (1 inch corresponds to approx. 25 mm). Please check the values entered for **ConversionData.factor** in older projects and correct them when necessary.

Regardless of the sign, the encoder resolution/measured value width is limited to maximum 31-bit, because the configuration data of the maximum limits are signed data values of the DINT type.

For the **Left-justified/Right-justified without sign** setting, the measured value for the setting of the encoder resolution/measured value width is processed as follows:

- At ≤16 bits the measured value left-justified/right-justified is mapped to the least-significant byte 1 and byte 2 of an internal data value of the DINT type. In this process, the missing 16 bits minus the measured value width of the least-significant byte 1 and byte 2 are right/left-padded with zeros and the most-significant byte 3 and byte 4 are filled with zeros.
- > At 16 bits the measured value left-justified/right-justified is mapped to an internal data value of the DINT type. In this process, the missing 32 bits minus the measured value width are right/left-padded with zeros.
  As this mapped measured value is tested against the maximum limits of the data type DINT, the encoder resolution/measured value width is limited to maximum 31 bits. This means the measuring range is limited to maximum 50% of the measured value width.

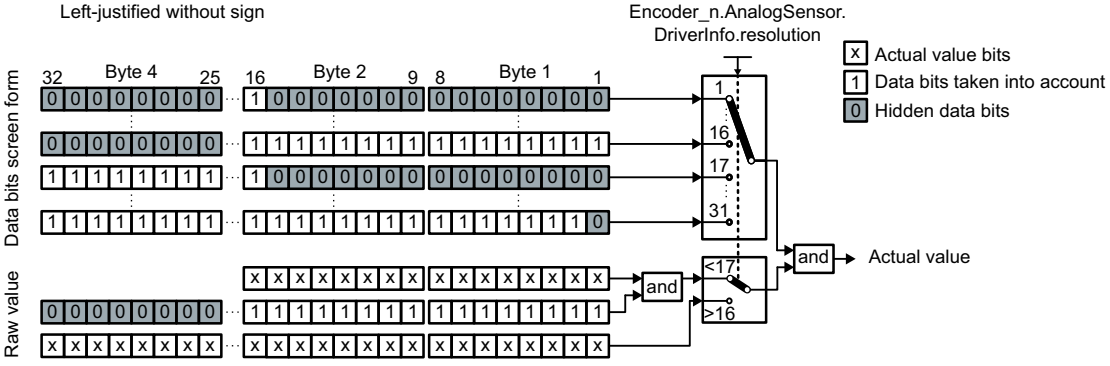


Figure 4-814 Masked reading of the analog IO value left-justified without sign

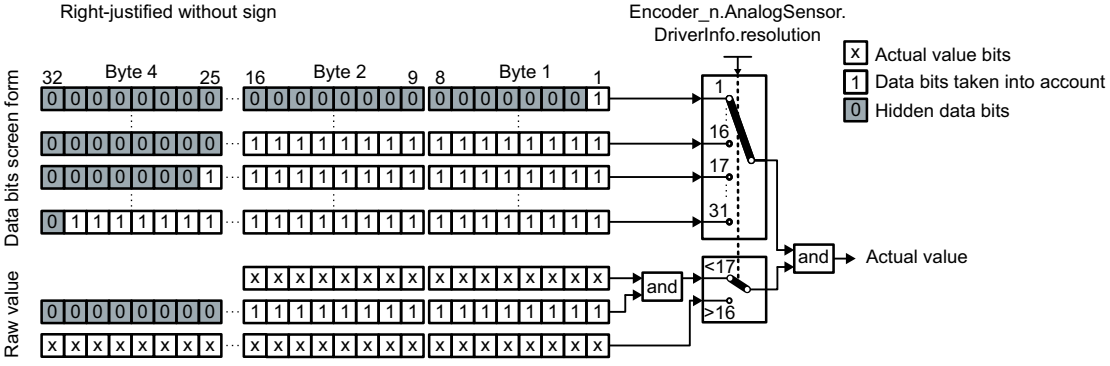


Figure 4-815 Masked reading of the analog IO value right-justified without sign

For the **Left-justified/Right-justified without sign** setting, the measured value for the setting of the encoder resolution/measured value width is processed as follows:

- At <16 bits the measured value left-justified/right-justified is mapped to the least-significant byte 1 and byte 2 of an internal data value of the DINT type. For left-justified, the missing 15 bits minus measured value width of the least-significant byte 1 and byte 2 are right-padded with zeros and the sign bit is extended left to bit 32. For right-justified, the sign bit is padded left to bit 32.
- At ≥16 bits the measured value left-justified/right-justified is mapped to an internal data value of the DINT type. For left-justified, the missing 31 bits minus measured value width are right-padded with zeros. For right-justified, the sign bit is padded left to bit 32.

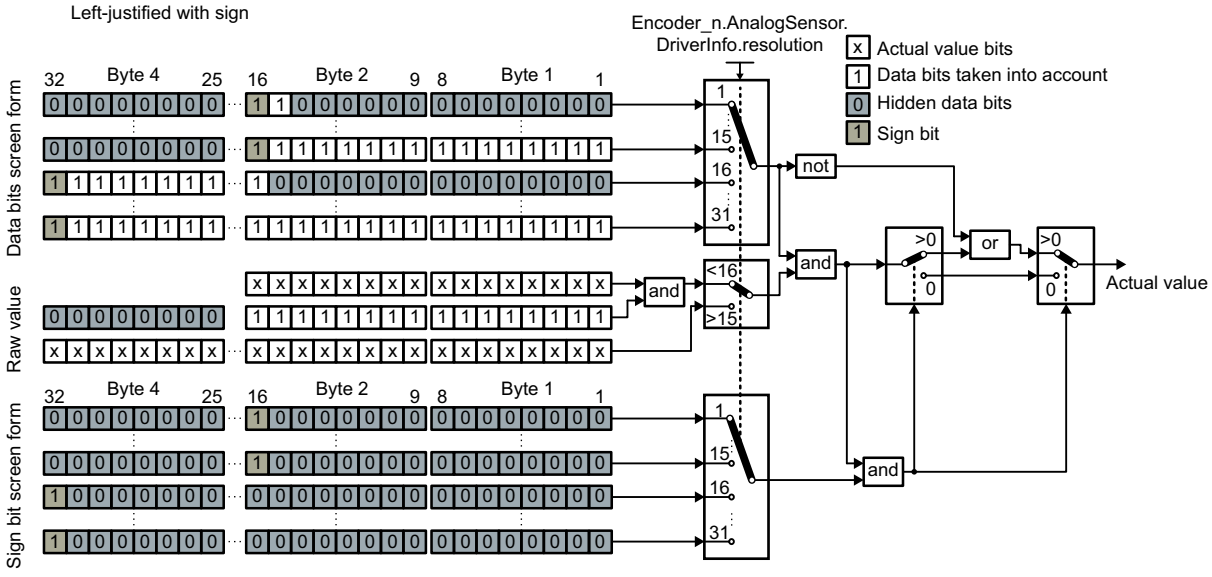


Figure 4-816 Masked reading of the analog IO value left-justified with sign

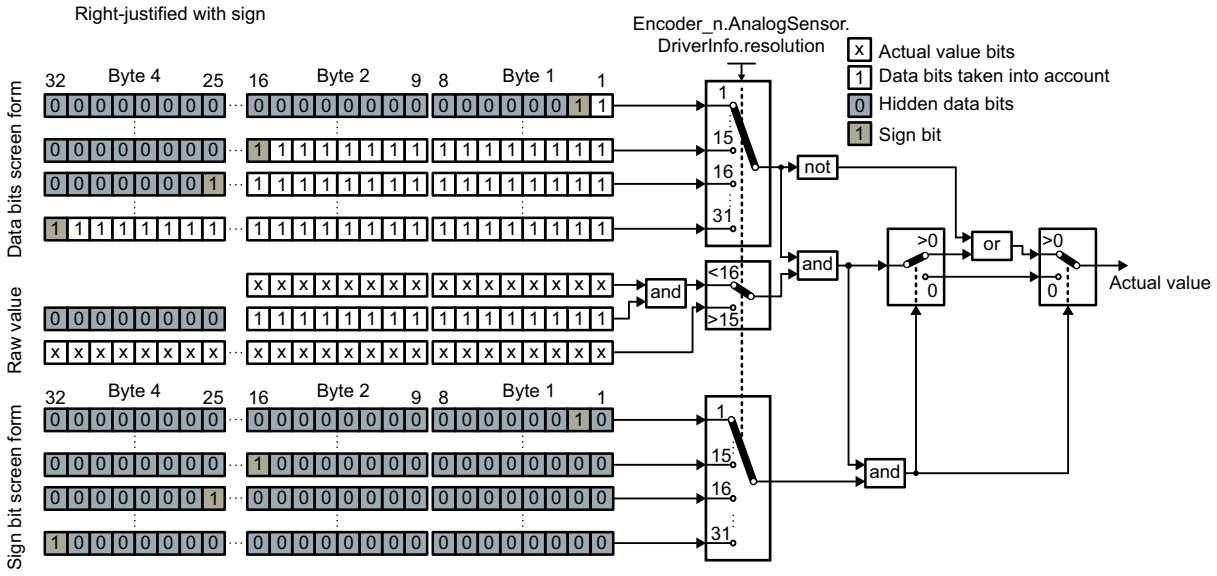


Figure 4-817 Masked reading of the analog IO value right-justified with sign

Counter value (as of V4.0)

The encoder is set as an incremental encoder. 16 bits or 32 bits can be set as counter value width.

Example: Use of an ET 200S, COUNT module

Actual velocity

The actual value information can include the number of pulses between two scans or, alternatively, the time interval between two sequential pulses. This type of encoder is used, for example, to measure actual velocities with the hydraulic axis functionality.

Example: Use of an ET 200S, COUNT module

Direct value as absolute value in I/O area (as of V4.1 SP1)

The following bits can be configured for the direct value as absolute value in the I/O area:

- **Ready bit** via the elements of the **analogSensor.readyStateMonitoring** configuration data element
- **Error bit** via the elements of the **analogSensor.errorStateMonitoring** configuration data element

These can be used for evaluation on the Axis technology object of a *ready identifier* and an *error identifier*, which are available from the peripheral module in addition to the measured value.

In SIMOTION V4.1 SP1, this configuration data is defined directly in the expert list.

If the *not ready* status or an error is displayed in these identifiers during operation, and the **ready bit** or **error bit** is configured, technology alarm 20005 is issued with the *sensor error* identifier.

If the Axis technology object is ready during ramp-up, but the direct value in the I/O area is not yet in ready status, the *WAIT_FOR_VALID Sensor* status is displayed on the encoder. (**sensorData[n].state** system variable)

As of V4.1 SP1, the direct value in the I/O area does not have to be updated in every equidistant communication cycle (for example, if an encoder connected to the peripheral module is not able to supply a new measured value in each cycle during a fast communication cycle clock due to measurement reasons or processing time reasons). In such cases, the actual value is extrapolated by the control.

The control supports the following options:

- The peripheral module displays the new measured value set in an update bit/update counter. The update bit or update counter is defined in the **analogSensor.UpdateCounter** configuration data element.
  UpdateCounter configuration: The update counter can be one (toggle) bit or several (counter) bits in width.
- The refresh cycle of the actual value in the peripheral module is known and is defined directly in the **analogSensor.UpdateCounter.updateCycle** configuration data element.
  Default setting with refresh cycle = 1 (default behavior for updating in every cycle clock)

Encoder value via system variable

A **position or velocity value** can also be defined directly in a system variable (sensorSettings.actualValue) from the user program, for example. With this value, it is possible, for example, to simulate the actual technological value for an axis or to provide the value from any encoder/I/O value (for which there does not exist a TO interface) via the user program and to apply it to the axis as an actual value.

This behavior is defined in
TypeOfAxis.numberOfEncoders.Encoder_x.encoderIdentification=SET_ACTUAL_VALUE.

This encoder type can be defined on the axis and the external encoder.

The update rate for the default setting (**updateCycle**) and the maximum number of servo clocks (**maxFailures**) in which the system variable is not written are specified in the **StructAxisSensorSetActualValueConfig** configuration data structure.

The **sensorSettings.actualValue** system variable must be provided in the cyclical level set in the **typeOfAxis.NumberOfEncoders.Encoder_x.SensorSetActualValue.updateCycle** configuration data. If, for example, in the user program of the IPOsynchronousTask, the actual position is written cyclically to **sensorSettings.actualValue**, **~.updateCycle** = IPO must be set.

The value specified in **sensorSettings.actualValue** is accepted using the set refresh rate (**SensorSetActualValue.updateCycle**) and extrapolated linearly in between.

SensorSetActualValue.maxFailures always refers to the servo clock. The **SensorSetActualValue.updateCycle** setting does not have any affect.

Example:
If the **sensorSettings.actualValue** system variable is updated in the IPO cycle clock and IPO:Servo is set 2:1, **SensorSetActualValue.maxFailures** = 1 must be set.

Non-exclusive encoder assignment (as of V4.1 SP1)

The sensor/encoder of the axis can be defined as exclusive or non-exclusive in **typeOfAxis.NumberOfEncoders.Encoder_x.interfaceAllocation**. If it is defined as non-exclusive, you must also specify whether the encoder interface should be activated when the technology object is ramped up.

The functionality supports the configuring of an encoder on several axes. However, the encoder can only be evaluated on one axis at a time.

The **_enableAxisInterface()** or **_disableAxisInterface()** commands and the sensor=<sensorNumber> function parameter are used to activate and deactivate the encoder interface during operation and the specification of the sensor. Because the sensor function parameter is encoded as bits, it is also possible to activate/deactivate several encoders concurrently. If the bit is not set, the encoder interface status remains unchanged.

The Activated/Deactivated encoder interface status is displayed in the **sensorData.sensorData[i].input** system variable.

Example:

sensor=5 Activation/deactivation of the encoder interface 1 and 3

Also refer to the parameters description in the reference lists.

These commands are not available for enabling/disabling the actuator/sensor interface on the hydraulic axis. For information on the hydraulic functionality, see Access to the same final controlling element from multiple axes (Page 3134).

See also

Setting a non-exclusive drive assignment (as of V4.1 SP1) (Page 2898)

Diagnostic features

The measuring system increments are displayed as 32-bit information in the **sensorData.incrementalPosition** system variable.

It can first be verified whether the change in this variable over one encoder revolution with allowance for the measuring gear corresponds to the increments of the encoder. If this is not the case, the configuration of the axis and the drive must be modified accordingly (e.g. measuring gear, (drive) parameters for setting the resolution: number of increments per revolution, fine resolution, etc.).

In addition, the path (angle) change/revolution (**sensorData.position**) can be checked. If this check reveals problems, the axis configuration must be checked (e.g., leadscrew pitch, gear ratios, etc.).

4.8.3.6 Input limits, technological limiting functions

System-side input limits

All parameters that can be entered have a lower and an upper limit derived either from the variable format range or from limits set by the system for each parameter. See the TP Cam System Variables Reference List.

Internally there are also limits for the position specifications resulting from the accuracy of the data format of the variables. When these limits are violated, an error message is output on the axis: **Internal traversing range limit reached**.

4.8.3.7 Setting for axis and encoder mechanics

Overview of setting options for axis and encoder mechanics

The following options are available on the drive side of a real axis:

- Consideration of load gear
- Consideration of spindle pitch on linear axis
- Inversion of manipulated variable/drive direction

The following options are available on the encoder side of a real axis:

- Setting of encoder mounting method
- Consideration of measuring gear
- Setting of distance per revolution

- Settings for load compensation
- Consideration of the count direction (inversion of actual position value)

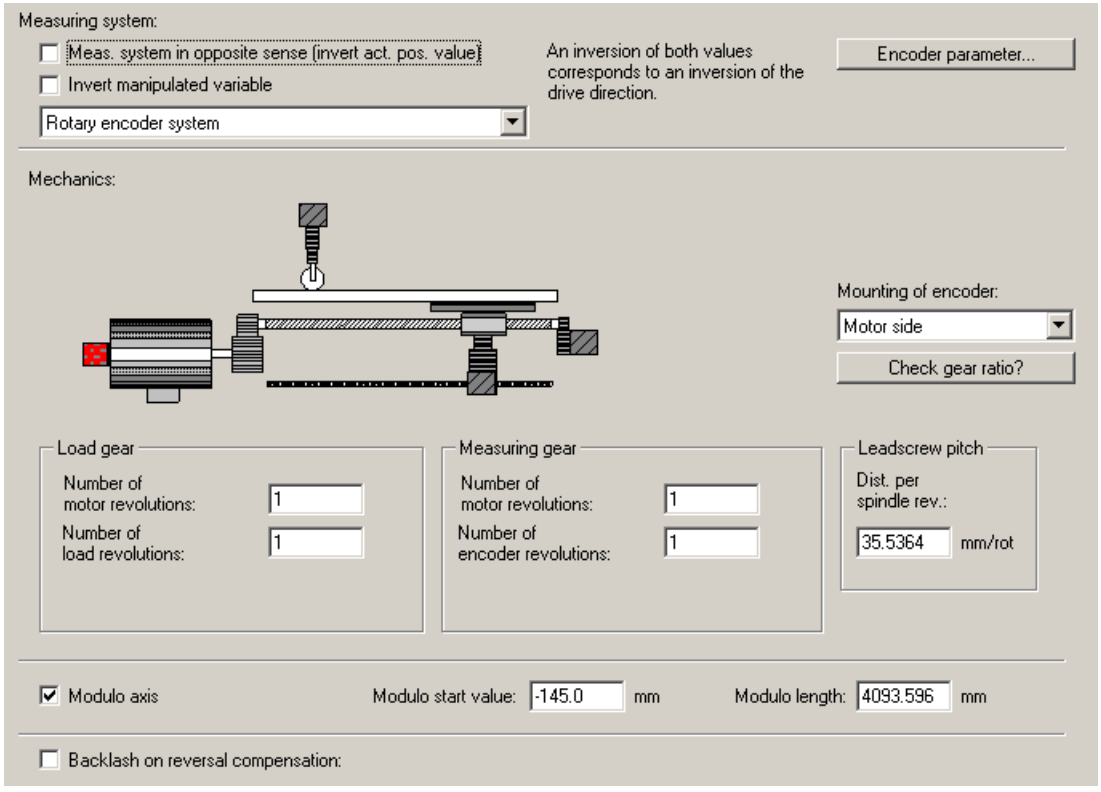


Figure 4-818 Overview of encoder/linear axis mechanics

The following options are available on the final control element side for the hydraulic functionality:

- Consideration of output characteristic
- Inversion of manipulated variables

The available encoders in SIMOTION can be used for the hydraulic functionality. They can be configured on the **Encoder configurations** tab in the **Configuration** dialog or via the expert list. (see SIMOTION SCOUT online help)

Inversion of actual position value

You can invert the count direction of an incremental or absolute encoder by selecting the **Meas. system in opposite sense** check box in the **Encoder/linear axis mechanics** screen form. This changes the count direction to match the technological view.

Once the Invert actual position value setting has been modified, the encoder should be adjusted once again, this time with the new absolute encoder offset which has been set accordingly, just as the home position offset needs to be redefined for an incremental encoder.

Special features of the absolute encoder

With an n-bit encoder (n-bit data length), the incremental position lies between 0 and 2^n-1. The position is then derived independently of the resolution, e.g. from 0 to 100 mm.

If the sign is changed by inverting the actual position value (measuring system is in opposite direction or encoder is mounted backwards), the incremental position lies between 2^n-1 and 0. The position range is inverted accordingly, e.g. from 100 to 0 mm. If the actual position in this case is 15 mm without sign change when the encoder is switched on, then the position with sign change is 85 mm.

Example:
A linear scale with a grid line spacing of 0.005 mm and a measuring range of 30 mm or 6,000 increments is specified. The maximum value for the incremental position can be displayed on a computer as a data value with a width of 13 bits (2^{13} = 8,192).
Therefore, the encoder's maximum position is in theory 40.96 mm (8,192 * 0.005 mm). If the axis is offset in the positive direction, the displayed encoder position increases its value, moving from 0.0 up toward 40.96 mm, and the incremental position increases from 0 up toward 8,191 increments. An incremental position of 1,000 increments, for example, corresponds to an axis position of 5.0 mm. By contrast, changes to the axis position in the positive direction on an encoder where the invert actual position value function has been activated will cause the displayed encoder position to decrease its value, moving from 46.96 down toward 0.0 mm. The inverted encoder position is derived from the difference between the encoder's theoretical maximum position and the encoder position determined by the current incremental position. This means that the encoder incremental position of 1,000 increments given in the example results in a position of 35.96 mm (40.96 mm - 5.0 mm = 35.96 mm).

Boundary conditions for mechanical settings for modulo axes (long-term stability / long-term accuracy)

For modulo axes, SIMOTION checks whether the long-term accuracy can be guaranteed. Testing takes place on initial download, each restart of the CPU and each restart of the technology objects, if relevant data have changed. The long-term accuracy ensures the following:

- That the axis position displayed by the controller always conforms to the real position, i.e. the axis position can be displayed uniquely from the accumulated encoder increments.
- That the calculated position in the controller does not differ from the real position. Without long-term accuracy, differences result, e.g. because of rounding errors or calculations with finite accuracy.

This enables positions to be approached exactly, even after any number of modulo overflows.

Note on non-modulo axes:

Non-modulo axes are “always accurate” because of an endless traversing range, even if the convertibility of gear ratios is not stable over the long term.

If the maximum internal position ($9x10^{12}$ at 1000 increments/unit) is reached and the convertibility of gear ratios is not stable over the long term, an alarm 50011 occurs (range limit of the incremental actual value exceeded) and the axis must be re-homed.

The long-term accuracy/stability of modulo axes is dependent on the following:

- Internal resolution, increments/position [units/mm]
  (can be set at **Configuration > Axis type > Configure units**)
- Encoder resolution: [incr/rev]

- Encoder fine resolution
- Spindle pitch [mm/rev] (for linear modulo axes)
- Measuring gearbox
- Load gearbox

Checking long-term accuracy as of V4.2

The long-term accuracy is only checked during runtime, as a result of the encoder data adaptation.

Long-term accuracy means that the technological position can be determined from the accumulated encoder increments (without rounding error) and is therefore always accurate.

With long-term accuracy, the encoder position is converted to the mechanical position via the numerator/denominator factor of the gear ratio. If there is no long-term accuracy, an LREAL factor is used as substitute for the conversion. Alarm 50024 (long-term stability of the actual values is not guaranteed) is then output. NONE is the default setting for the local response.

When using the LREAL factor to take into account the gear ratio, modulo axes are accurate as long as the total traversing distance of the axes without modular conversion does not exceed the maximum traversing range of the axis (representation range $9x10^{12}$ at 1000 increments/unit).

If the maximum value of the representation range is exceeded, this results in inaccuracy of the technological position. The unavoidable rounding error in the last available position of the LREAL factor applies in this case. However, for the generally used settings, this is only reached after years.

Determining the maximum traversing time of an endlessly traversing axis

The maximum traversing time can be determined based on the following estimate. The maximum position and the traversing velocity are decisive:

Maximum position = $9x10^{12}$mm at an internal resolution (default) of 1000 internal increments/ position unit

Traversing time = maximum position / velocity

Table 4-349 Example of a linear modulo axis

Velocity:	20 m/min = $2x10^4$ mm/min
Maximum position:	$9x10^{12}$mm
Traversing time = $9x10^{12}$ mm / ($2x10^4$ mm/min) = $4.5x10^8$ min = 856.16 years	

At a higher set internal computational accuracy, the traversing time is correspondingly lower.

If the determined traversing time does not meet your requirements, you can use the alternative procedure presented below or identify the required settings for long-term stability in accordance with the section “Checking long-term accuracy < SIMOTION V4.2”.

Alternative procedure

Reset the position by homing if the maximum traversing time does not meet your requirements.

Incremental encoder

Home the incremental encoder again before the position value $9x10^{12}$ is reached.

Absolute encoder (as of SIMOTION V4.3 SP1 HF17)

Perform absolute encoder adjustment with preset current known position via the **_homing()** function using homingMode=SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION.

This resets the position values to the current measurement range in a cyclic absolute encoder. The reset may be performed via a user program e.g. after a

Power ON.

Checking long-term accuracy < V4.2

The long-term accuracy of a modulo axis is a precondition for operating the axis.

If long-term accuracy cannot be guaranteed as a result of the mechanical settings, then one of the following error messages is output in the engineering system during the consistency check.

- Configured gear ratio cannot be represented
- Configured modulo length cannot be represented

A test of the long-term accuracy is also performed on the target device during runtime. If the long-term stability is not ensured due to the configuration, the following alarm is issued: **Alarm 20006 Configuration error, reason 3041**.

The reason for this error is the unsuitable selection of values in the configuration data. The Excel list available on the Siemens Service & Support (https://support.industry.siemens.com/cs/ww/en/view/40786053) website assists you when checking and calculating gearbox parameters.

See also

Encoder interface using the PROFIdrive message frame (Page 2906)

Overview of setting options for axis and encoder mechanics (Page 2918)

4.8.3.8 Defaults

The axis has system variables that can be set to default values for the axis. The most important default values, e.g., for the dynamic response of the axis, are compiled in the Dynamic Response screen form. These default values will be used if you use **Default** or **USER DEFAULT** in your programming.

Table 4-350 Other default values that can be set via the expert list

Default values for	System variable
Clamping values	**userDefaultClamping**
Dynamic values	**userDefaultDynamics**
Switchover to force control	**userDefaultForceControl**
Force/pressure limiting	**userDefaultForceLimiting**
Homing	**userDefaultHoming**

Default values for	System variable
Positioning	**userDefaultPositioning**
Hydraulics	**userDefaultQFaxis**
Torque limiting	**userDefaultTorqueLimiting**

See also

Default settings for dynamic response parameters (Page 2997)

4.8.3.9 Homing

Overview of homing

On the positioning axis, inputs and displays concerning the position refer to the coordinate system of the axis.

The coordinate system of the axis must be aligned with the real, physical position of the axis.

Absolute encoder

For absolute encoders, inclusion of the absolute encoder offset must be activated **once**.

For more information, refer to Section States that require re-adjustment of the absolute encoder (Page 2934).

Incremental encoder

For **incremental encoders**, synchronization is performed by homing when the home position coordinates, or the home position coordinate taking into account the home position offset, are set in active homing to a defined mechanical position of the axis. This defined mechanical position of the axis is signaled to the controller via the encoder zero mark of the measuring system or via the external zero mark.

For incremental encoders, if you want to establish a direct reference to the position, you must synchronize the actual value system of the axis after every activation.

Note

Traversing commands with **relative position specification** can always be executed. The axis configuration can be set to determine whether traversing commands with **absolute position specification** can also be executed on a non-homed axis. Configuration data: **referencingNecessary**.

See also

Homing (Page 3156)

Terminology

Home position

When the axis has been synchronized and the reference point offset has been applied, the axis is at the reference point and has the value specified in the reference point coordinate.

Set the home position or the home position coordinate in the **userDefaultHoming.homePosition** system variable. The setting is axis-specific.

The **homingCommand.state** system variable displays the homing status on the axis.

Synchronization point

In the synchronization point, the actual value of the axis due to an external or internal event is set to the value **home position coordinate minus home position offset**.

Reference point offset

The offset between the home position and the synchronization point is only effective during active homing. It is applied after synchronization of the axis by means of the homing command. In modulo axes, the **home position offset** is always applied with the **Shortest_Way** direction setting. (Exception: As of 4.1 SP1, the direction can be specified with the "Home in one direction only" setting.)

Specify the home position offset in the **TypeOfAxis.NumberOfEncoders.Encoder_1.IncHomingEncoder.proceedShiftPos** configuration data element.

Reference mark

A homing mark is a hardware signal that is used for homing.

- Encoder zero mark
  The encoder zero mark of an incremental encoder is used as a reference mark.
- External zero mark
  An external signal (zero mark substitute) is used as the reference mark.

Homing output cam

The homing output cam outputs an enable signal for the actual reference signal (encoder zero mark or external zero mark). The drive reduces the velocity on the basis of the switching edge returned by the homing output cam and waits for the next incoming homing signal in order to perform the homing operation.

Note

Device-specific properties

The homing output cam is connected to any control input for homing with homing output cam and encoder zero mark.

For more information about the device-specific boundary conditions and additional parameter setting requirements, see the supplementary information for drives and the product manuals.

Homing types

Open-loop control supports the following types of homing:

- **Active homing**
  A special traversing motion is carried out for this type of homing. The following homing modes can be selected via the configuration:
  - Homing with homing output cam and encoder zero mark
  - Homing with external zero mark only
  - Homing with encoder zero mark only.
- **Passive homing/flying homing**
  This type of homing occurs during motion that was not initiated by a homing command. The following homing modes can be selected via the configuration:
  - Homing with homing output cam and encoder zero mark
  - Homing with external zero mark only
  - Homing with encoder zero mark only.
- **Direct homing / Set home position**
  The axis position is set without a traversing motion having taken place.
- **Relative direct homing**
  The actual position value of the axis is shifted by a specified offset without a traversing motion having taken place.
- **Absolute encoder homing/absolute encoder adjustment**
  The zero point of the absolute encoder is adjusted.

Note

As of V4.2, the inputs for homing output cams and for the status of the external zero mark and of the reversing cam can be interconnected to SINAMICS inputs symbolically instead of via the intermediary of a logical address.

Active homing

During active homing, the homing operation is performed according to the specified mode by means of a motion initiated by the homing command.

A home position offset is in effect and is applied after synchronization with the reference mark. Once the home position offset has been applied, the axis position has the value specified in the home position coordinate.

In active homing, the velocity specifications for the homing approach velocity, the homing reduced velocity, and the homing entry velocity are effective.

The axis has **synchronized** or **homed** status once the reference mark is detected.

Note

The direction of approach to the reference cam or reference mark must be specified in the user program. As of V4.1, support is provided for using hardware limit switches as reversing cams.

Active homing with reference cam and encoder zero mark mode

The homing command causes the axis to move toward the reference cam. After the axis has detected and left the reference cam, the axis moves to the next encoder zero mark of the position measuring system. You can specify in a configuration data element whether the encoder zero mark is in the positive or negative traversing direction. The axis is synchronized with the first encoder zero mark detected after the reference cam. Then, the home position offset is traveled at the homing entry velocity. The axis position now has the value defined in the home position coordinate.

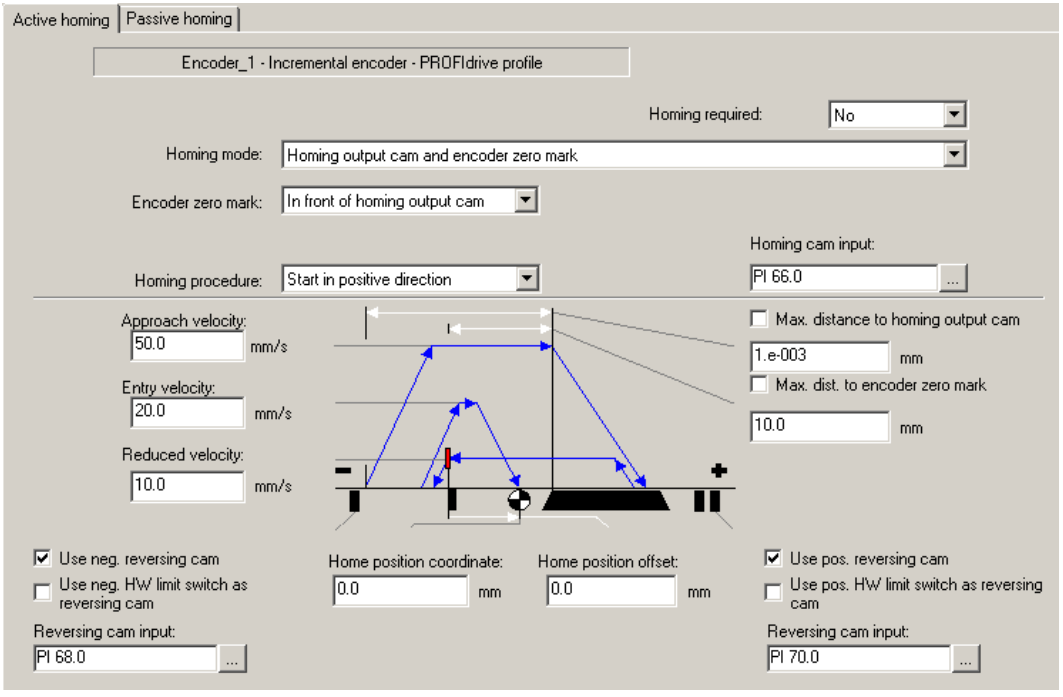


Figure 4-819 Parameters for active homing with reference cam and encoder zero mark

The sequence can be divided into three phases:

- **Phase 1**
  Travel to the reference cam
  The axis moves to the reference cam at the **approach velocity**. The approach direction is parameterized.
- **Phase 2**
  Synchronization with encoder zero mark
  The axis moves to the encoder zero mark of the incremental position encoder at the **reduced velocity**. The position of the encoder zero mark relative to the reference cam can be parameterized. Depending on this position, the movement is either continued in the same direction or reversed. The controller is synchronized to the first detected encoder zero mark which is detected according to the configuration setting. When the encoder zero mark is detected, the axis is considered to be synchronized, and the axis position is set to the value specified in the home position coordinate, minus the home position offset.
- **Phase 3**
  Travel to the home position
  When the encoder zero mark is detected, the axis moves to the home position at the **entry velocity**.

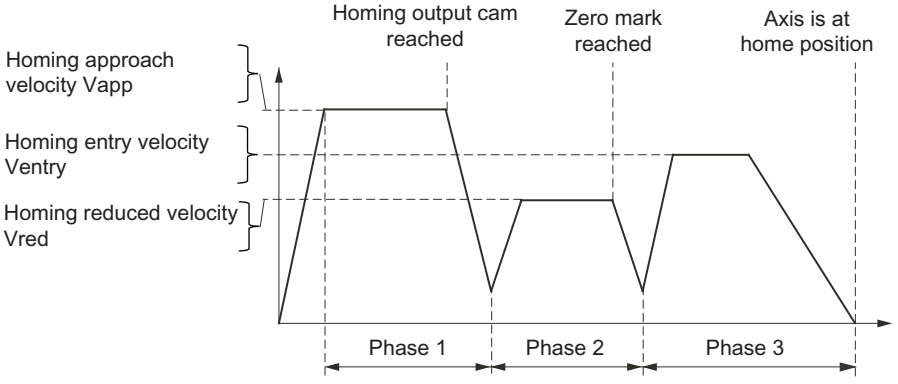


Figure 4-820 Homing with incremental measuring systems

Hardware limit switch as reference cam (as of V4.1 SP1)

The hardware limit switches can also be defined as a reference cam. The axis is homed to the first encoder zero mark after the direction is reversed as a consequence of approaching the hardware limit switch. The axis cannot continue to travel in the direction of the hardware limit switch.

This corresponds to homing with reference cam and encoder zero mark, whereby the encoder zero mark lies in front of the reference cam.

Alternatively, a left and right hardware limit switch can be used (positive or negative approach direction). In this case, the hardware limit switch is deactivated during homing.

Active homing with external zero mark only

Here, the homing command initiates motion toward the external zero mark. Once the axis reaches the configured edge of the external zero mark, the home position offset is applied at the homing entry velocity. The axis position now has the value defined in the home position coordinate.

Note

For homing to the external zero mark of axes with a digital drive link via PROFIdrive, the external zero mark must be configured as a digital input on the drive (configuration of zero mark substitute for SIMODRIVE 611U, SINAMICS). The external zero mark signal must be connected at the same location where the encoder is measured, e.g. on the drive, on the ADI4/IM174, or on the inputs of the C2xx intended for the external zero mark.

SINAMICS

With SINAMICS, a positive direction of motion is synchronized to a positive trigger edge and a negative direction of motion is synchronized to a negative trigger edge, i.e. on the left side of the external zero point signal in each case.

By inverting the signal (setting option on the drive: SINAMICS parameter via P490), synchronization is also possible on the right side of the external zero point signal.

In SINAMICS, the homing to the encoder zero mark or the external zero mark is set in P495.

To be able to detect for drives coupled using PROFIdrive that the drive is positioned at the external zero point, the state of the external zero point on the axis must be made available. Because the corresponding status bit is not provided in the PROFIdrive telegram of the drive DO, an additional input bit containing the status of the external zero mark can be configured on the axis in the **incHomingEncoder.stateDriveExternalZeroMark** configuration data element as of V4.1 SP1. For drives linked with SINAMICS, the corresponding status bit in PZD2 of telegram 390 or 391 of the CU is transferred to the controller.

If symbolic assignment is activated, the status of the external zero mark can be assigned to the corresponding onboard terminal on the control unit directly using the Assign dialog.

Refer to the SINAMICS documentation for the assignment of digital inputs of the drive to the status bits of the PZD2.

The system will detect whether the axis is already positioned at the external zero point. Travel from the external zero point is made opposite to the approach direction. This is followed by normal homing.

SIMODRIVE 611U

With SIMODRIVE 611U, only falling edges are permitted in the positive traversing direction, and only rising edges are permitted in the negative traversing direction, meaning that the same output cam side is always used.

Inverting is not possible with SIMODRIVE 611U. The edges can be inverted by using initiators of the inverted type or via an application (inaccurate).

MASTERDRIVES

With MASTERDRIVES Motion Control, only rising edges are permitted.

OnBoard C2x0

With OnBoard, only rising edges are permitted.

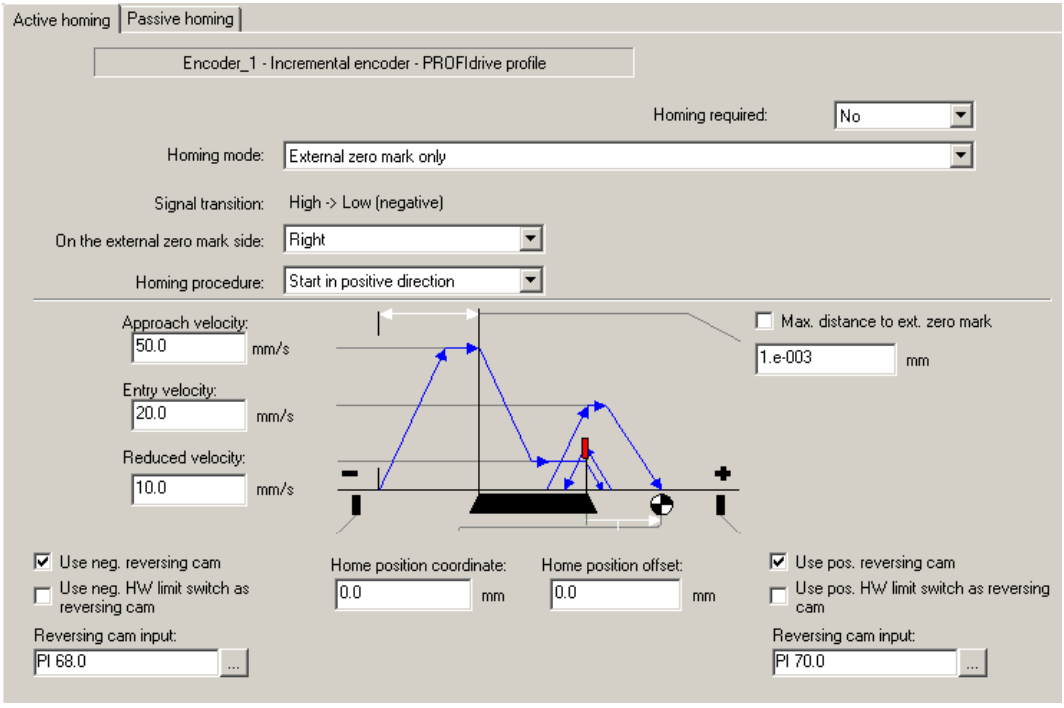


Figure 4-821 Parameters for active homing with external zero mark only

Note

- Only homing with a reference cam and an encoder zero mark should be considered for a resolver with a pole pair number > 1.

- With a digital link to SINAMICS S120, the settings for homing are read out from the drive on request. Changes are not written back to the drive.

Active homing with encoder zero mark only (without reference cam)

Homing without a reference cam is used, for example, in axes for which the encoder has only one encoder zero mark in the entire axis traversing range. This homing command causes the axis to travel to the encoder zero mark. After the encoder zero mark is detected, the axis approaches the shifted reference point at homing velocity. The axis position now has the value defined in the home position coordinate.

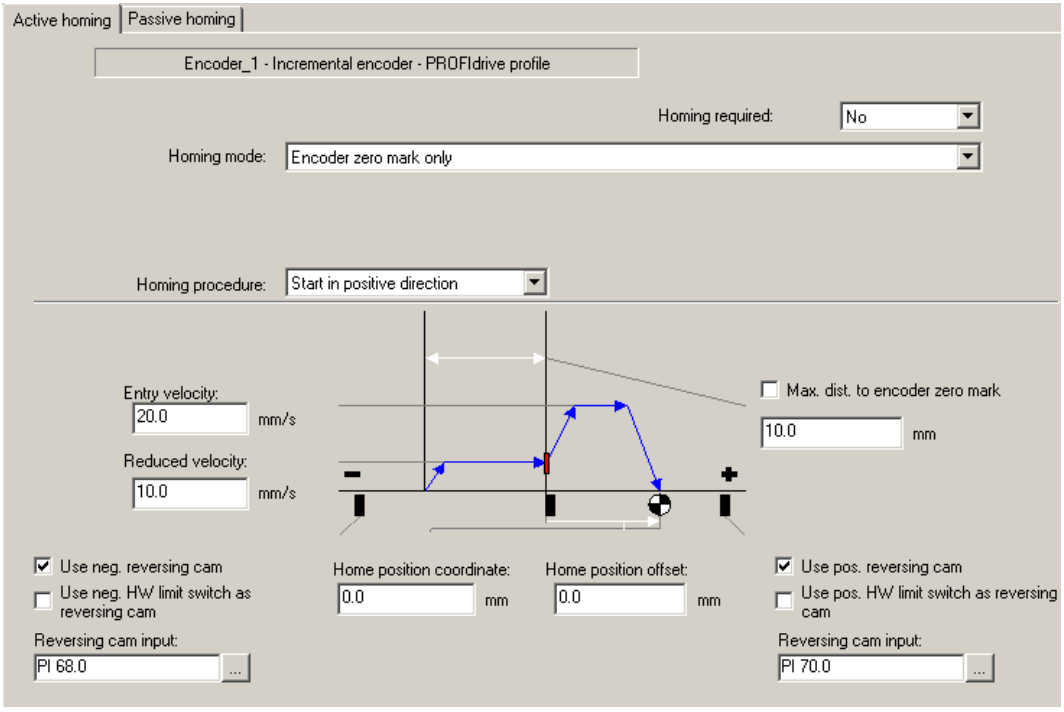


Figure 4-822 Parameters for active homing with encoder zero mark only

Note

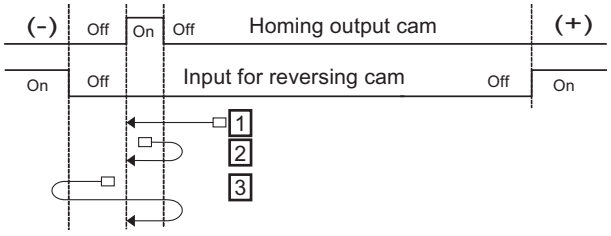
Only homing with a reference cam and an encoder zero mark should be considered for a resolver with a pole pair number > 1.

Reversing cams for homing (as of V4.1 SP1)

Reversing cams, which are effective only during active homing, are used to reverse the direction of the home position travel when approaching a reversing cam.

The reversing cams are configured as two additional input signals. The left reversing cam and the right reversing cam can be configured and activated separately. The reversing cams are defined once on the axis. They cannot be defined specifically for an encoder.

The following figure illustrates homing sequences based on the starting point position with respect to the reference cam.



All examples are programmed with the Left-hand approach direction (negative)

1 Starting point in front of reference cam

2 Axis is on reference cam
This is detected by the system, and the axis travels against the approach direction away from the reference cam. This is followed by normal homing.

3 Axis is behind the reference cam on the left
A normal homing operation with approach direction left is started. The axis reverses direction at the reversing cam and travels against the approach direction until it is over the reference cam. This is followed by normal homing.

Figure 4-823 Reversing cams for homing

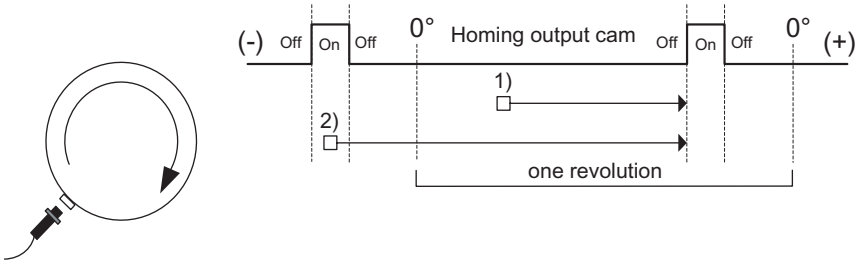
The hardware limit switches can also be defined as reversing cams. In this case, the hardware limit switch is deactivated during homing.

The reversing cams are defined via the configuration data elements **typeOfAxis.homing.reverseCamPositive** and **typeOfAxis.homing.reverseCamNegative**.

Homing in one direction only (as of V4.1 SP1)

As of V4.0, you can specify in the **typeOfAxis.homing.direction** configuration data element that a direction reversal during homing should be suppressed and the specified traversing direction should always be maintained. This setting is effective only during active homing.

The figure below shows an application example featuring a rotary axis that is not permitted to reverse direction.



All examples are programmed with the Right-hand approach direction

1) Starting point in front of reference cam
Reference cam is found in the modulo area.

2) Axis is on reference cam
This is detected by the system. The axis travels in the programmed approach direction to the next reference cam, even if the modulo area is overtraveled.

Figure 4-824 Example of homing in one direction only

Passive homing/on-the-fly homing

In passive homing, the homing is performed according to the mode setting using a motion not initiated by the homing command. Passive homing is possible in position-controlled mode in association with motion commands. A home position offset is not applied. The axis has synchronized or homed status once it has detected the homing mark. Specifications for the homing approach velocity, reduced velocity, and entry velocity are not applied.

Default homing mode (DEFAULT_PASSIVE)

With this setting, the homing mode is defined by the system based on the encoder type:

- With incremental sin/cos encoders, TTL encoders, or resolvers, homing is executed to an encoder zero mark.
- With Endat encoders defined as incremental encoders, homing is executed to an external zero mark.

Passive homing with homing output cam and encoder zero mark mode

Once the homing output cam has been detected, the next encoder zero mark takes effect according to the synchronization specified in the configuration. Synchronization is performed with the first encoder zero mark detected after the homing output cam. When the encoder zero mark is detected, the axis is set to the value specified in the home position coordinate. The axis is then given Synchronized and Homed status.

Passive homing with external zero mark only

Once the external zero mark has been detected, synchronization takes place according to the configuration. When the external zero mark is detected, the axis is set to the value specified in the home position coordinate. The axis is then in synchronized and homed status.

Passive homing with encoder zero mark only

Homing without a homing cam cam is used, for example, in axes for which the encoder has only one homing mark in the entire axis traversing range. When the homing mark is detected, synchronization takes place according to the configuration. Once the homing mark has been detected, the axis is set to the value specified in the home position coordinate and is in synchronized or homed status.

Note

Use the "homing output cam and encoder zero point" setting for homing for measuring systems with more than one encoder zero point in the traversing distance of the axis. This ensures that travel is made to an exact, reproducible homing position.

As an alternative, you can use the "only external zero point" setting. The external signal means, however, that the attainable homing position is less accurate.

Direct homing/setting the home position

The current position of the axis is set to the value specified in the home position coordinate. A home position offset is not applied. A traversing motion is not carried out. When the command is executed, the axis is in synchronized or homed status.

The setting of parameters for axis homing is irrelevant here. The home position coordinate is specified in the command.

Direct homing is not possible if the axis is traversed in speed control. Switch to position control for the homing.

See also

Positioning (Page 3158)

Synchronization/homing with incremental encoders (Page 3205)

Relative direct homing/relative setting of home position (V3.2 and higher)

The current position of the axis is offset by the value specified in the home position coordinate, so in this case the home position coordinate should be regarded as an offset.

This homing method can also be used during operation (while traversing).

The setting of parameters for axis homing is irrelevant here. The home position coordinate is specified in the command.

States that require a new homing procedure for incremental encoders

With incremental encoders, the status is reset to not homed in the following cases:

- Error in the sensor system / encoder failure
- New homing command
- Power off
- Termination of the communication to the drive/encoder
- SCOUT is downloaded with selection of the *Initialization of all retentive data* setting
- Changes requiring a download or restart are made to the axis configuration
- Technology object restart is performed on the Axis technology object
- When active homing is initiated

Absolute encoder homing / absolute encoder adjustment

The current axis position is set to equal the encoder value + absolute encoder offset using the **_homing()** command in ENABLE_OFFSET_OF_ABSOLUTE_ENCODER homing mode.

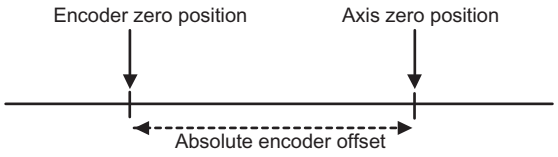


Figure 4-825 The axis zero position is the encoder zero position plus the absolute encoder offset

The axis must be in position-controlled mode while the absolute encoder is adjusted.

The absolute encoder offset (as of V3.2) may be set as an additive or absolute value.

This absolute encoder offset is stored in the NVRAM and remains in effect until the next absolute encoder adjustment. This function must therefore be executed once when the control is commissioned.

The offset value and its calculation are set during **configuration**.

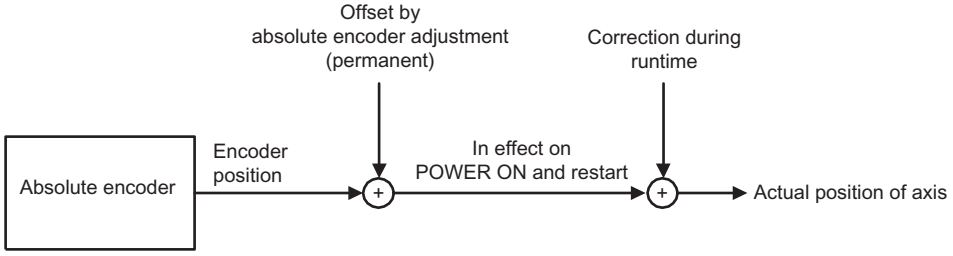


Figure 4-826 Incorporating the absolute encoder offset

A value may be set as a total offset using the **absHomingEncoder.setOffsetOfAbsoluteEncoder** and **absshift** configuration data.

Setting an additive offset

Setting **absHomingEncoder.setOffsetOfAbsoluteEncoder**=RELATIVE (default behavior):

- Actual axis value = actual encoder value + (previous offset already in effect + absshift)
- (new) offset = previous offset + absshift

absHomingEncoder.absshift is added to the existing absolute encoder offset whenever the **_homing()** function is called.

Setting an absolute offset (as of V3.2)

Setting **absHomingEncoder.setOffsetOfAbsoluteEncoder**=ABSOLUTE (as of V3.2):

absHomingEncoder.absshift is set as the absolute encoder offset whenever the **_homing()** function is called.

- Actual axis value = actual encoder value + **absshift**

Example of ABSOLUTE offset where actual encoder position value = 100.000:

absshift = 5.000		
_homing() command called for the first time	⑧	Position = 105.000
_homing() command called for the second time	⑧	Position = 105.000
absshift = 7.000		
_homing() command called for the third time	⑧	Position = 107.000

Setting the axis to a predefined position (as of V4.1 SP1)

When the function
parameter **homingMode**=SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION is set on the **_homing()** command, the value in the *homePosition* parameter is set as the current position. The resulting absolute encoder offset is calculated with this value, indicated in the **absoluteEncoder[n].totalOffsetValue** system variable, and stored as a retain variable in the system.

The value in the **absHomingEncoder.absshift** configuration data element is not changed.

Displaying the offset

The offset can be read out. (as of V3.1)

The total offset is indicated in the **absoluteEncoder[x].totalOffsetValue** system variable, while the activation status of the total offset is indicated in the **absoluteEncoder[x].activationState** variable.

In addition, the status indicates whether the **_homing()** function where **homingMode**= ENABLE_OFFSET_OF_ABSOLUTE_ENCODER was executed at least once after the project was downloaded.

Absolute encoder adjustment

Perform the following steps to adjust the absolute encoder:

1. Disable the software limit switches, because you cannot adjust the absolute encoder while these are active.
2. Perform the absolute encoder adjustment:
   - **_homing()** where **homingMode**= ENABLE_OFFSET_OF_ABSOLUTE_ENCODER
     The value of the **absHomingEncoder.absshift** configuration data element is included when the command is called once. (**absHomingEncoder.absshift** is a configuration data element that can be modified online, i.e. any changes take effect immediately.)
     or
   - **_homing()** where **homingMode**=SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION (as of V4.1 SP1)
     Move the axis to the desired reference position and call the command once.
     This sets the value in the *homePosition* parameter as the current position. The resulting absolute encoder offset is calculated using this value, indicated in the **absoluteEncoder[n].totalOffsetValue** system variable, and stored as a retain variable in the system.
     The value in the **absHomingEncoder.absshift** configuration data element is not changed.
3. Enable the software limit switches again (if necessary).

Note that the absolute encoder adjustment only acts as an offset to the absolute encoder value. The offset from the absolute encoder adjustment and the value of the absolute encoder are determining factors for the position after a power off or restart. During operation, the current actual position is also affected by the modulo settings for the axis and by position setting or position correction tasks.

States that require re-adjustment of the absolute encoder

- Once a new project has been downloaded to the control, the stored offset is no longer available.
  If the control already contains a project before the new project is downloaded, and if the technology object name is not changed, the stored offset is retained (as of V4.1 SP1). This behavior also applies for an upgrade, i.e. it is not version-dependent.
- After power is cycled off and on, the offset is deleted unless the project was saved to the ROM.
- After a memory reset.

See also

Encoder for position (Page 2901)

Homing mark monitoring

If the homing mark is not reached within the defined travel path, an alarm is triggered. In homing with homing output cam and homing mark, the path is monitored only after the axis leaves the homing output cam.

If reversing cams are present, monitoring is applied again when the direction reverses as a result of the revering cams.

When homing mark monitoring is enabled, both active and passive homing procedures are monitored.

Homing output monitoring

If the homing output cam is not reached within the defined travel path, an alarm is triggered.

If reversing cams are present, monitoring is applied again when the direction reverses as a result of the revering cams.

When homing output cam monitoring is enabled, both active and passive homing procedures are monitored.

Displaying actual value change during homing (V4.0 and higher)

A change of the actual value during homing is displayed in the **homingCommand.positionDifference** system variable.

Traversing with a non-homed axis

With the **referencingNecessary** configuration data element, you define whether absolute positions can be used with a non-homed axis.

Settings:

- referencingNecessary = NO
  - Relative and absolute motions are possible.
  - The software limit switches are monitored with the setting **swlimit.state** = YES.
- referencingNecessary = YES
  With a non-homed axis:
  - Only relative motion is possible
  - The software limit switches are not monitored even with the setting **swlimit.state** = YES.

Correcting the actual position/set position without homing

Position correction can also be used to manipulate the actual values and setpoints of individual coordinates (basic coordinates, superimposed coordinates).

The homing status of the axis (Homed/Not homed) does not change.

See also

Resetting the set and actual positions (Page 3168)

Differential position measurement (V3.2 and higher)

The axis position is not measured directly; it is determined as the difference between two encoders/actual values. The system adopts this differential position as the actual position and uses it as an absolute value.

The encoders for the individual positions may be either absolute encoders or incremental encoders.

Homing of the differential position is not permitted.

The offset relative to the differential position can be modified online using the **positionDifferenceMeasurement.Offset** configuration data element.

The differential position and the position sensors are used as encoders.

Differential position value =
Position (numberEncoderA) - Position (numberEncoderB)

Proceed as follows to specify the setting via the expert list:

- Configure at least two position sensors on the axis.
- In the expert list, increase the value of **TypeOfAxis.NumberOfEncoders.NumberOfEncoders** by 1.
- Set the other sensor as the "differential position sensor" via the **TypeOfAxis.NumberOfEncoders.Encoder_n.EncoderType** configuration data element with SENSOR_POSITION_DIFFERENCE_MEASUREMENT.
- The sensors whose values are used and the individual factors are set in the elements of the **TypeOfAxis.NumberOfEncoders.Encoder_n.PositionDifferenceMeasurement** structure.

See also

Using the expert list for an axis (Page 3065)

4.8.3.10 Monitoring/limiting functions

Overview of monitoring/limiting functions (block diagram)

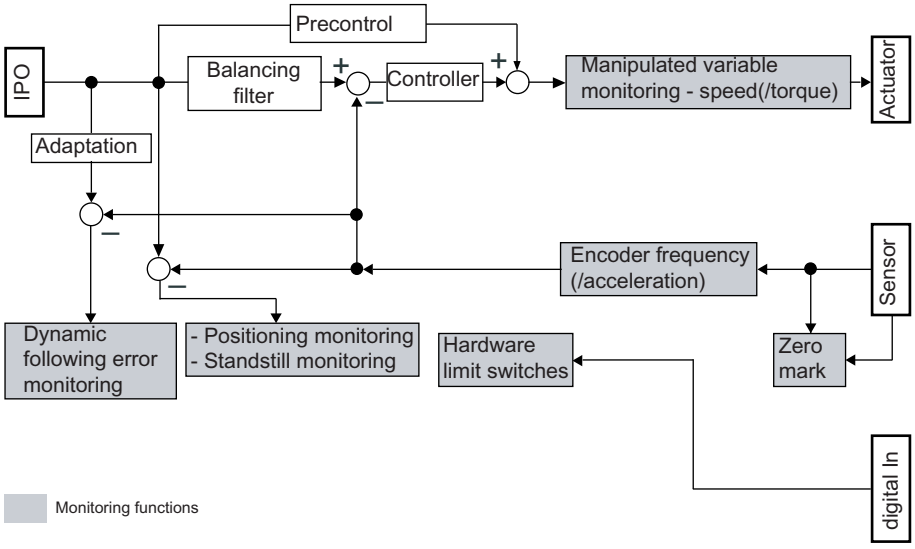


Figure 4-827 Block diagram of positioning axis monitoring functions

Dynamic monitoring of following errors

The following error monitoring on the position-controlled axis is performed on the basis of the velocity-controlled following error limit.

The maximum permissible following error is dependent on the set velocity.

For velocities less than a specified lower velocity, the maximum permissible following error is constant and set in **typeOfAxis.NumberOfDataSets.DataSet_N.DynamicFollowing.minPositionTolerance**. The lower velocity value for the characteristic of the maximum following error is set in **typeOfAxis.NumberOfDataSets.DataSet_N.DynamicFollowing.minVelocity**.

Above this lower velocity, the maximum permissible following error is proportional to the set velocity up to the maximum permissible velocity.

The maximum permissible following error for the maximum velocity of the axis is set in **typeOfAxis.NumberOfDataSets.DataSet_N.DynamicFollowing.maxPositionTolerance**. The gradient factor of the characteristic for the maximum following error is calculated from these settings.

The **dynamicFollowing.warningLimit** configuration data can be used to specify a threshold as percentage value based on the maximum permissible following error of the associated set velocity which, when exceeded, causes the issuance of a warning.

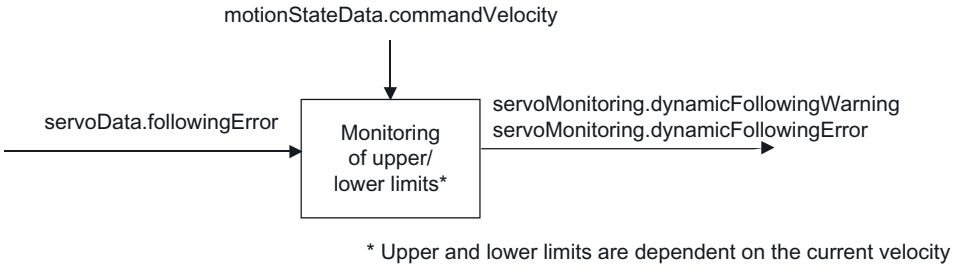


Figure 4-828 Dynamic monitoring of following errors

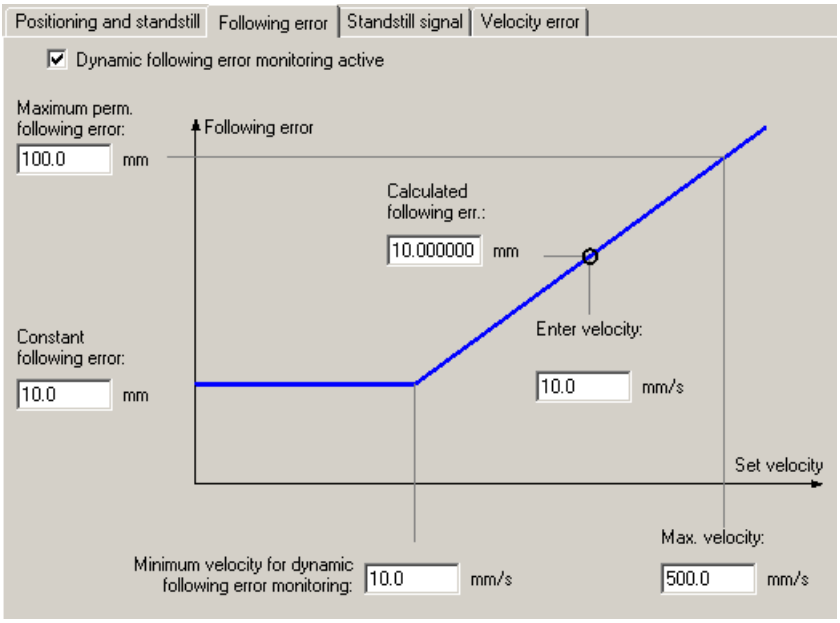


Figure 4-829 Function and parameters for following error monitoring

System variable on the TO with current position lag limit

The **Permissible following error** block is provided to monitor the stability of the axis. This contains the permissible dynamic following error as system variable (**servoData.permissibleFollowingError**).

Following error determination without DSC

The following error is determined from the difference between the non-symmetrical setpoint prior to inclusion of the dynamic response adjustment and the actual value present in the control.

Therefore, the transfer times of the setpoint to the drive and the actual value to the control are included in the following error.

Following error determination with DSC

The following error is determined from the difference between the non-symmetrical setpoint delayed by the transfer times (T_i+T_o+T_{dp}+T_{servo}) + **systemDeadTimeData.additionalTime** prior to inclusion of the dynamic response adjustment and the actual value present in the control.

The transfer times of the setpoint to the drive and the actual value to the control are calculated from the following error in order to refer to the following error present on the position controller in the drive.

Positioning and standstill monitoring

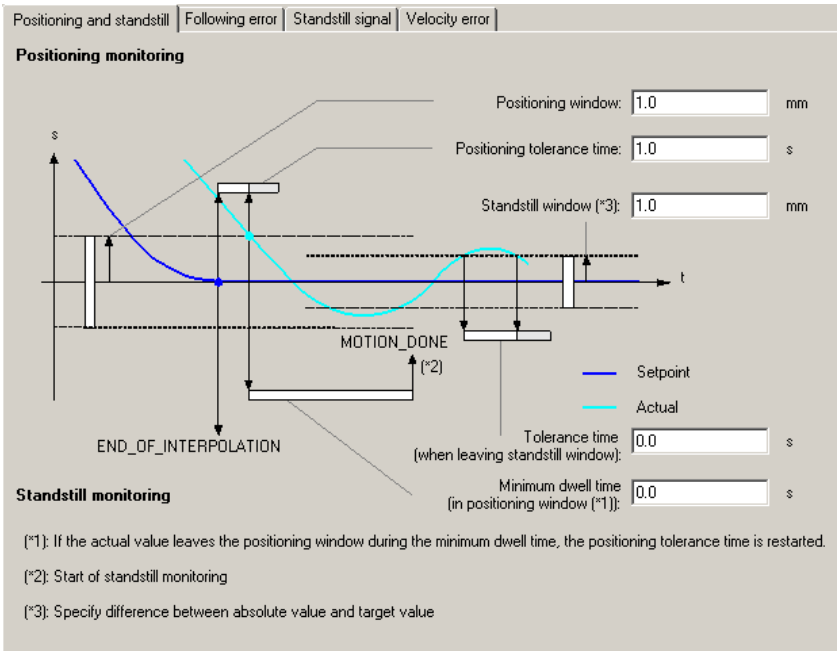


Figure 4-830 Function and parameters for positioning monitoring

Positioning monitoring

The behavior of the actual position at the end of the setpoint interpolation is monitored. (Positioning monitoring)

This positional monitoring does not distinguish whether the setpoint interpolation is ended as a result of reaching the target position from the setpoint side or due to a position-controlled stop during the motion performed by the interpolator (e.g. with a _stop() command). This monitoring is referred to as positioning monitoring, although the position does not have to be the same as the target position. The tolerance window specified for this monitoring is called the positioning window.

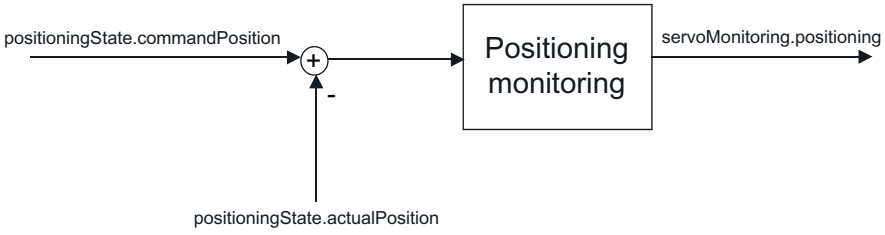


Figure 4-831 Positioning monitoring

Sequence:

- When the setpoint interpolation is complete, the **typeOfAxis.positionMonitoring.posWinTolTime** monitoring time is started.
- If, before this time expires, the actual value reaches a definable window **typeOfAxis.positionMonitoring.tolerance** around the existing position at the end of the setpoint interpolation, monitoring of the minimum delay time **typeOfAxis.positionMonitoring.posWinTolDelayTime** is initiated. This window is indicated as a deviation in **typeOfAxis.standStillMonitoring.stillStandTolerance**, meaning that half of the window width is set.
  If the actual value does not reach the window within the monitoring time **typeOfAxis.positionMonitoring.posWinTolTime**, alarm 50106 (positioning monitoring) is issued.
- If the actual value leaves this window again during the minimum dwell time **typeOfAxis.positionMonitoring.posWinTolDelayTime**, the monitoring time **typeOfAxis.positionMonitoring.posWinTolTime** is restarted; each time it re-enters this window, the minimum delay time **typeOfAxis.positionMonitoring.posWinTolDelayTime** is restarted.
- If the actual value remains in this window for the minimum delay time **typeOfAxis.positionMonitoring.posWinTolDelayTime**, the MOTION_DONE status is set in the system variable **motionStateData.motionCommand** and the standstill monitoring is started.

In addition, the individual monitoring phases are displayed in **servoMonitoring.positioningState** (as of V4.1 SP1):

- ACTUAL_VALUE_OUT_OF_POSITIONING_WINDOW = setpoint interpolation is complete; actual value has not yet reached the positioning window.
- ACTUAL_VALUE_INSIDE_POSITIONING_WINDOW = actual value is inside positioning window; standstill monitoring is not yet started.
  The ACTUAL_VALUE_OUT_OF_POSITIONING_WINDOW is displayed when the actual value has left the positioning window again.
- STANDSTILL_MONITORING_ACTIVE = standstill monitoring is active; positioning to the position reached at the end of the setpoint interpolation has taken place.
- INACTIVE

Note

Positional monitoring (e.g. following error monitoring or positioning monitoring) are disabled when the pressure control is enabled or via the pressure limiting or torque limiting command.

Standstill (zero-speed) monitoring

Standstill monitoring is defined by the standstill window and the tolerance time during which the standstill window may be exited without alarm 50107 (standstill monitoring) being triggered.

Standstill monitoring starts when the minimum dwell time in the positioning window has elapsed.

The standstill window is indicated as a deviation in **typeOfAxis.standStillMonitoring.stillStandTolerance**, meaning that half of the window width is set.

The status of the standstill monitoring is displayed in **servoMonitoring.stillstand**. Standstill monitoring is not available on the speed-controlled axis.

Standstill signal

The standstill signal **motionStateData.stillstandVelocity** is ACTIVE when the current velocity is less than a configured velocity threshold for at least the duration of the delay time.

Note

Below this velocity, the motion is stopped in response to **_stopEmergency()** at zero setpoint without a preconfigured deceleration ramp.

In speed-controlled motion, the criterion for completing the command is standstill monitoring. The completion of commands for position-controlled motions is described in **Positioning and standstill monitoring**.

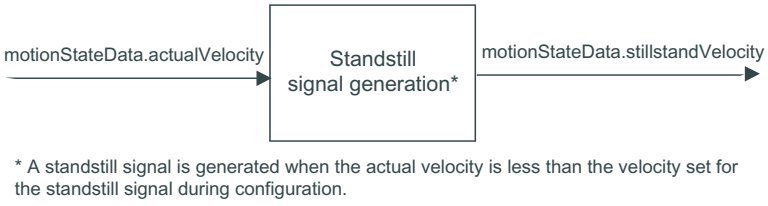


Figure 4-832 Standstill signal generation

The standstill signal is available on the positioning axis and the speed-controlled axis.

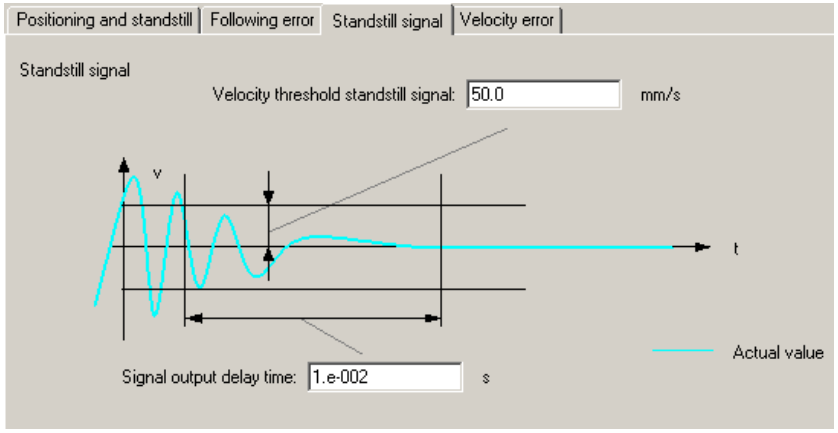


Figure 4-833 Function and adjustable parameters for the standstill signal

Standstill signal monitoring

motionStateData.ActualVelocity shows the actual velocity after the smoothing filter. The axis velocity before the smoothing filter is used for the standstill signal monitoring.

The unfiltered velocity used for the standstill window is not displayed specifically.

See also

Positioning and standstill monitoring (Page 2941)

Manipulated variable monitoring

The maximum manipulated variables are limited for monitoring of the assigned speed limits. If the manipulated variables exceed the maximum value configured in **MaxSpeed**, alarm "50005 speed setpoint monitoring" is triggered.

With the help of increase monitoring of the manipulated variable, the maximum possible acceleration and thus the maximum torque are monitored. If the monitoring function is switched on and the acceleration set in the configuration data **MaxAcceleration** is exceeded, alarm 50003 (limit of speed set acceleration is active) is output, and the parameter **SetPointChangeRate** is set to LIMIT_EXCEEDED.

Manipulated variable limiting (backstop) (V3.1 and higher)

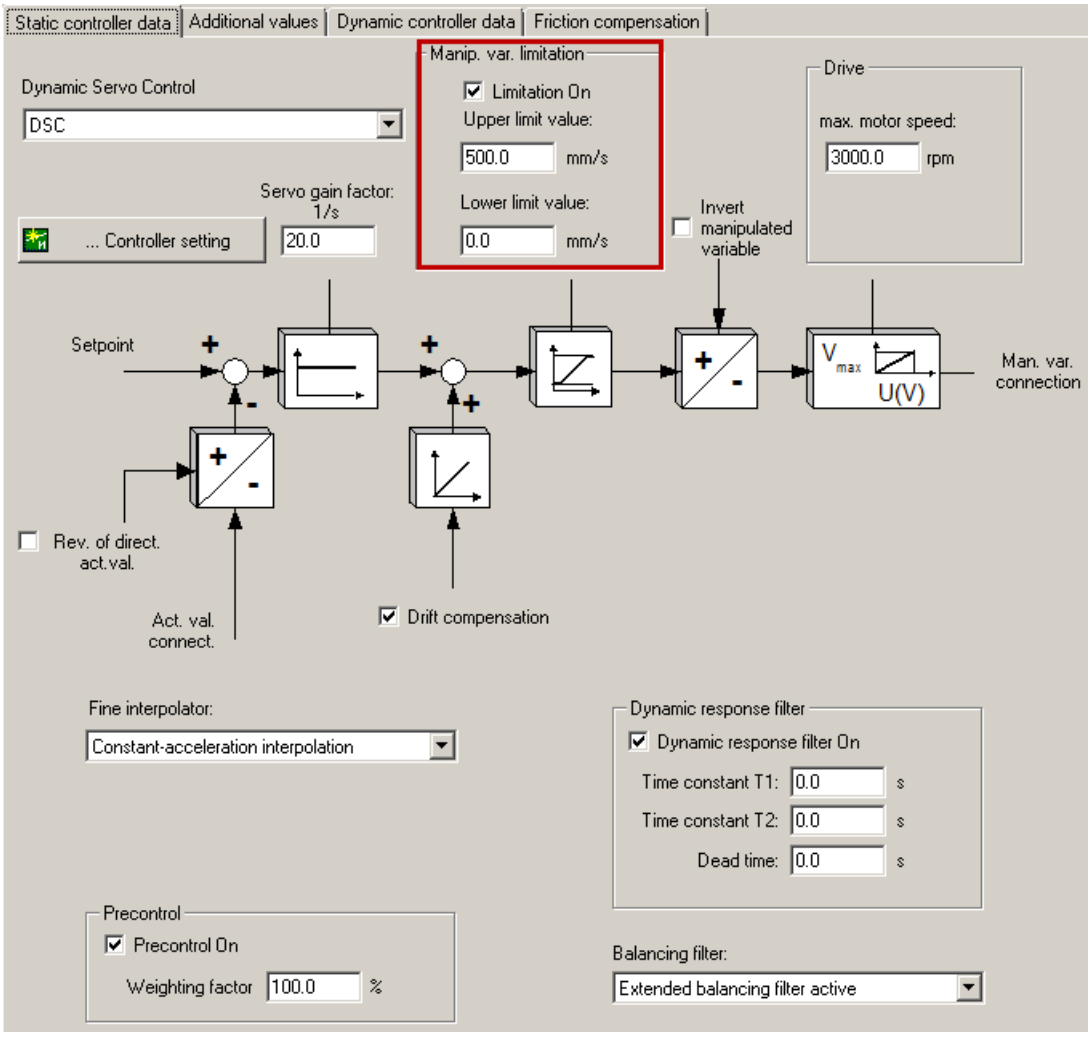


Figure 4-834 Manipulated variable limitation in static controller data

Manipulated variable limitation on an electrical positioning axis in the servo performs absolute limiting of the position value to an upper and a lower limit value.

This limitation is applied prior to inversion.

The values can be modified online (with immediate effect).

If the upper value is positive and the lower value is 0, the axis can, for example, be traversed in the positive direction only.

The manipulated variable limitation can be configured and activated by means of the **speedLimitation** configuration parameter. The zero velocity must lie within the permissible range to enable the axis standstill.

Only the setpoint is limited; reversing of the drive must also be prevented.

Note

When the **Dynamic Servo Control** (position controller in the drive) function is active, backstop (limiting of the manipulated variable for the drive) is only effective for the precontrol.

Therefore, when DSC is active, the backstop must be generated in the drive.

Hardware limit monitoring

Traversing range limits are monitored by means of digital inputs and limit switches. Hardware limit switches are **always normally closed switches** and should **always be active outside** the permissible traversing range.

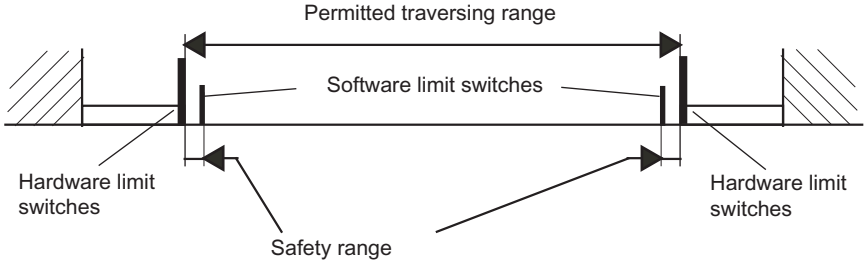


Figure 4-835 Monitoring of the permissible traversing range by limit switches

Approaching the limit switch triggers **technology alarm 50007**.

This state can be remedied in two ways:

- **Manual retraction (without drive)**
  The axis is returned manually to the permissible traversing range. The system variables remain in LIMIT_EXCEEDED state until the axis leaves the limit switch. Only then can the alarm be acknowledged.
- **Retraction with drive**
  The technology alarm must be acknowledged. **Warning 50009: Limit switch overtraveled** is retained. The axis can only travel in the direction of retraction as long as the limit switch is active. Travel in the opposite direction triggers a technology alarm and cancels the enables. When the axis has left the limit switch, the system variables **sensorMonitoring.hwLimitSwitchMinus** and **sensorMonitoring.hwLimitSwitchPlus** assume the **O_K** state, and the **Limit switch overtraveled** warning can now be acknowledged.

When a limit switch is approached, its position is stored. The limit switch is not considered to have been left until the axis moves back from the position.

NOTICE
Switching the controller off after the limit switch has been overtraveled
Once the limit switch has been overtraveled, the controller must not be switched off to avoid a conflict between the polarity monitoring of the limit switches and the overtravel monitoring of the limit switches in the direction of the permissible area.
If the controller is switched off, the information of the limit switch polarity is lost. The axis must then be moved into the permissible range by the user.
When the controller is **switched on**, the axis must be positioned within the permissible traversing range.
If the hardware limit switches are configured as reversing cams or reference cams for homing, these hardware limit switches can be overtraveled during homing, even if they are activated.

If the limit switch is overtraveled and the configuration reloaded, internal states are lost. Reloading without loss of the approach information is only possible within the valid range. Exception: Deactivation of limit monitoring after a polarity reversal error.

A cable break can only be reset with **Power On** or by a one-time deactivation of the function.

The hardware end position interface can be activated and deactivated on the axis using the **_enableAxisInterface()** and **_disableAxisInterface()** commands.

The Activated/Deactivated hardware end position interface status is displayed in the **sensorMonitoring.hwLimitSwitchInput** system variable.

Software limit monitoring

Software limit switches can be specified and monitored as soon as the actual values are valid. Monitoring is enabled/disabled using the **swLimit.state** system variable. The software limit switch positions are defined in the **swLimit** system variable. Software limit switches should be inside of the hardware limit switches.

If an axis motion, synchronous motion, or path motion is canceled by approaching the software limit switch, because continuing the motion would violate the software limit switch, movement to the software limit switch is performed at the braking ramp's maximum dynamic values. Technological alarm 40106, "Approach SW limit switch", is only output when the software limit switch is reached.

If the software limit switches are also to be monitored with non-homed axes, **homing.referencingNecessary** = NO must be set in the configuration data element.

With the setting **Homing required**, active limit monitoring is only effective if the axis is homed. If the setting **Homing required** is not selected, active limit monitoring is always effective.

The software limit position monitoring is effective also for isochronous movement and path movement. For a violation of the software limit position, the isochronous movement or path movement will be terminated and travel made to the software limit switch with maximum deceleration and maximum jerk.

If valid actual values are present for a speed-controlled procedure, these serve as limits for the software end position monitoring.

Monitoring the software limit switch at motion start (V3.2 and higher)

The controller checks for a violation of the limits before the motion starts. If the software limit switch is exceeded, the axis is limited to the software limit switch position and alarm 40105 is triggered.

If alarm 40105 is active, no more movement commands will be accepted and the limit switch will be approached at the programmed dynamic values. The error only needs to be acknowledged when, for example, using an application program, to stop before the limit switch or to travel in the reverse direction.

If, for example, a second motion is superimposed and acts in the opposite direction, it is possible for the software limit monitoring to signal an alarm when the first motion is activated, even though the software limit switch is not reached.

The **monitoringAtMotionStart** configuration data element can be used to enable/disable the check when the motion starts.

Note

The cyclical software limit monitoring check is always performed during the motion.

Passing a software limit switch

If the axis is positioned after the software limit switch due to reconfiguring of the software limit switch, for example, or the axis has passed the software limit switch in speed-controlled mode, alarm 40107 *Software limit switch passed* is output. The axis can be traversed out of the limit switch area in position-controlled mode depending on the local error response set in alarm 40107, e.g. when setting FEEDBACK_EMERGENCY_STOP.

For retraction in speed-controlled mode, a tolerance window with the configuration data **relieveWindow** can be defined. The tolerance window always refers to the current actual value. This means that jittering of the actual value will not cause the software limit switch to respond again during retraction. Traversing within the tolerance window is possible in both directions.

Encoder limit frequency monitoring

The system monitors the limit frequency of the encoders for compliance. The monitoring function triggers an alarm. It has no effect on axis motion.

Velocity error monitoring

An encoder must be connected and configured to monitor the velocity error (setpoint minus actual value) on the axis. The controlled system is simulated using a PT1 model. This model is supplied with the setpoint as the input value, and the difference of the output value is compared with the real actual value curve. The time constant for the PT1 model is set during axis configuration in the **dynamicData.velocityTimeConstant** or **dynamicQFData.velocityTimeConstant** configuration data element. Velocity error monitoring is of relevance for the drive axis and for the position axis in SPEED_CONTROLLED mode.

Measuring system differential/slip monitoring

The system can monitor a measuring system difference / slip between two encoders on the axis to a specified maximum value. The monitoring is activated with **_enableMonitoringOfEncoderDifference()** and deactivated with **_disableMonitoringOfEncoderDifference()**.

The maximum value is specified in the **_enableMonitoringOfEncoderDifference()** command in the *maximalEncoderDifference* function parameter and transferred with the execution of the command in the **sensorMonitoring.maximalSensorDifference** system variable or the existing value in the **sensorMonitoring.maximalSensorDifference** system variable is used optionally by setting the *maximalEncoderDifferenceType* function parameter.

If the maximum measuring system difference specified in **sensorMonitoring.maximalSensorDifference** for activated monitoring is exceeded, the alarm 20009, "the permissible difference between encode (/1/%d) and (/2/%d) has been exceeded" will be generated and signaled with LIMIT_EXCEEDED in the **sensorMonitoring.slippageTolerance** system variable.

Before activating the function (and if applicable while the function is active), check whether the two encoders involved in the measuring system difference are valid. The check can be performed using the system variables **sensormonitoring.cyclicinterface**=ACTIVE and **sensordata.state**=VALID, for example.
The absolute encoder is restarted, for example, on switchover by means of data set changeover and may then supply other values over the course of a small number of cycle clocks, so that in this case measuring system difference monitoring may respond.

4.8.3.11 Positioning axis with position control

Overview of positioning axis with position control

This figure shows the block diagram of the position axis with position control.

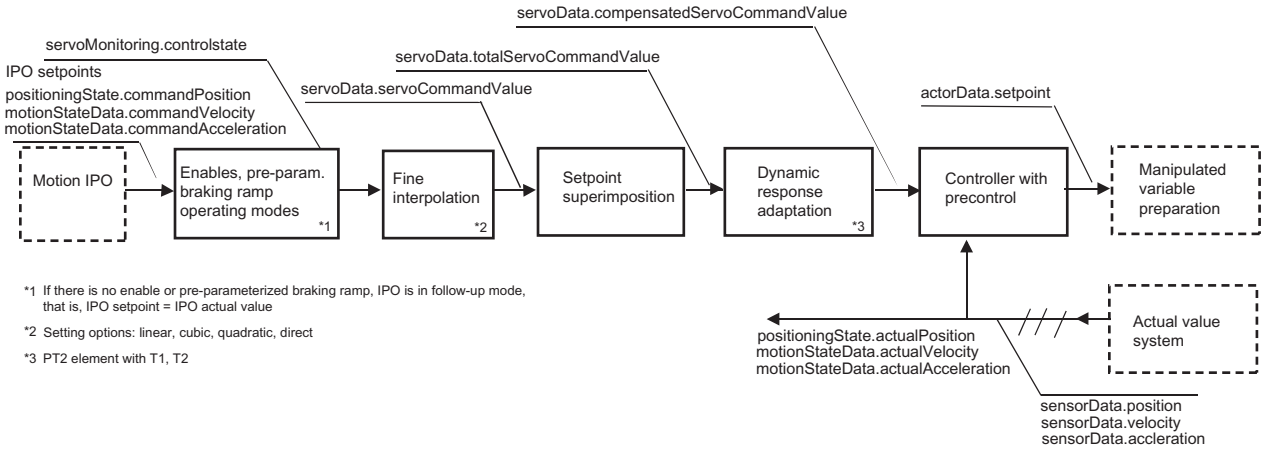


Figure 4-836 Overview of position axis with position control

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

You can call signal flow diagrams under the Axis object in the project navigator. The signal flow diagram illustrates the signal path and enables major items of configuration data and important system variables to be parameterized using the signal flow.

Position control

When position control is **active**, controllers, monitoring, and compensation are active. There are modes in which monitoring mechanisms are deactivated. They include torque limitation and the position-related monitoring functions for pressure limitation, for example.

All compensation functions can be activated/deactivated.

Encoder systems, actual value calculation, and monitoring are active on the actual value side when position control is not activated. Compensation functions are not taken into account.

The **servoMonitoring.controlState** system variable indicates whether the position controller is active.

SIMOTION provides a P-controller with or without precontrol and a PID controller.

Controller with precontrol

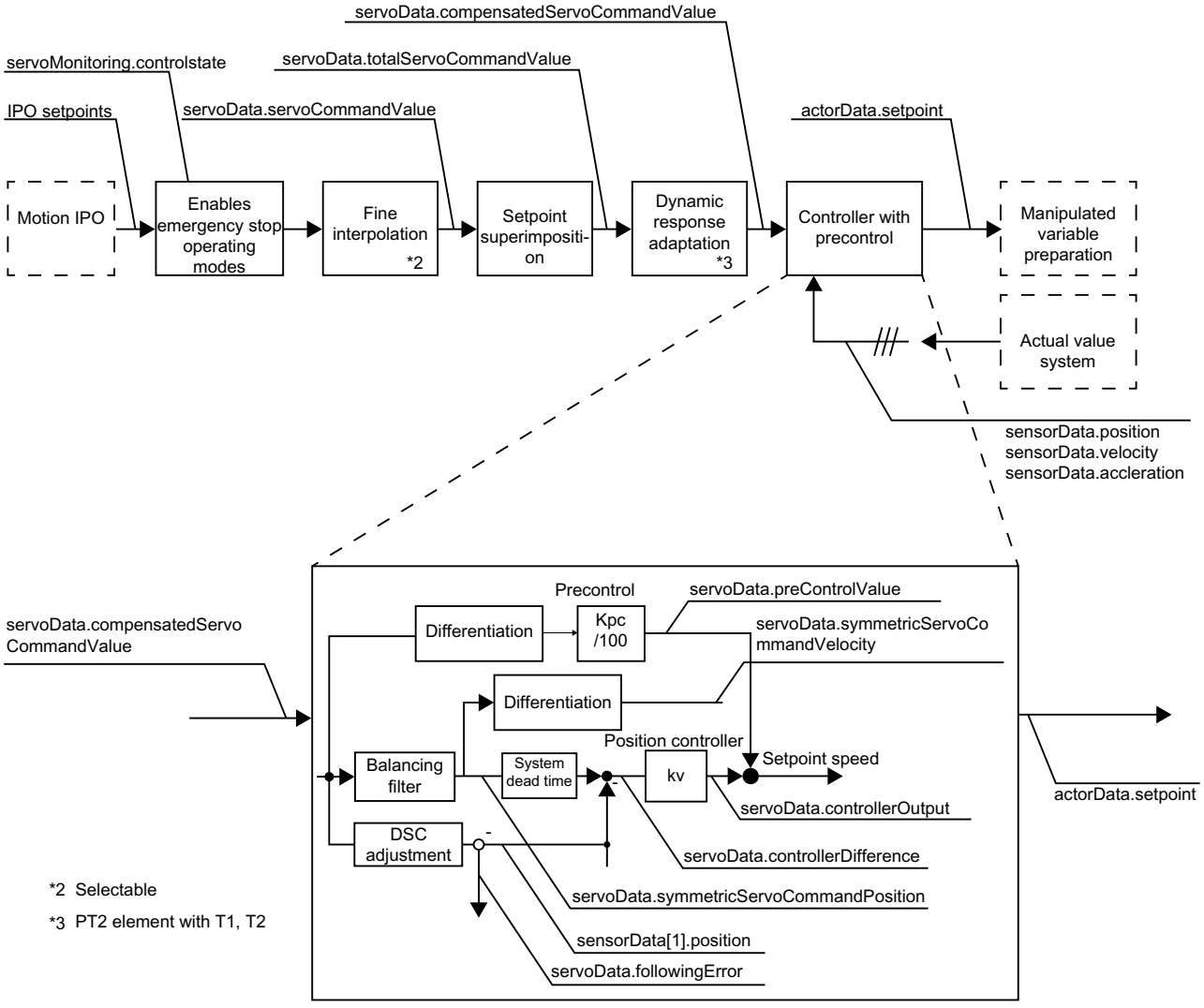


Figure 4-837 Proportional-action controller with precontrol

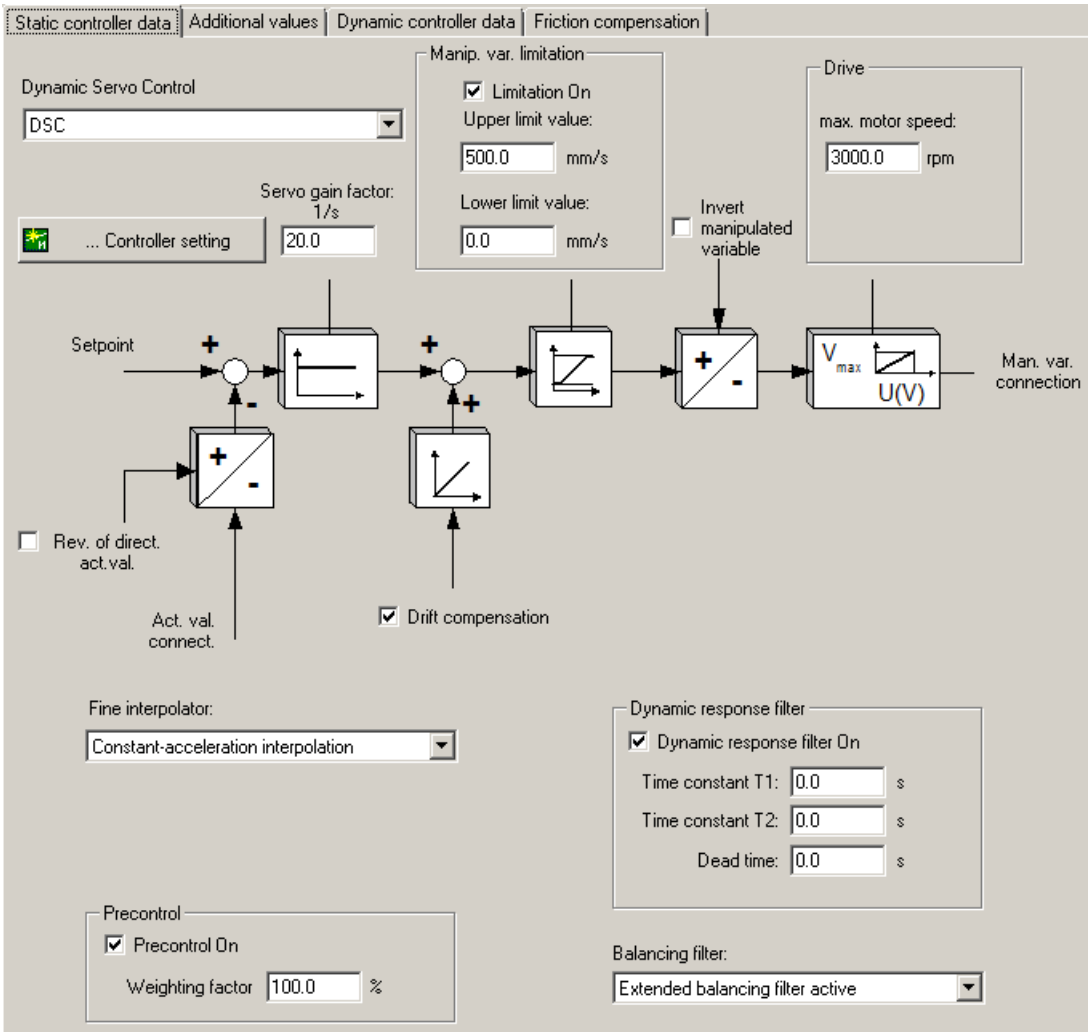


Figure 4-838 Static controller data

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

Axis wizard default settings

- P-action controller with precontrol for the electric axis
- DSC to improve the control quality (higher servo gain values) for digitally coupled drives (only with P controllers with precontrol)
- PID controller for hydraulic axes (allows you to switch the actual value directly to the D component).

See also Position control for setting a positioning axis with hydraulic functionality (Page 3124).

Position loop gain (servo gain factor)

For a P controller with or without precontrol, the gain of the P controller can be specified via the servo gain factor. The manipulated variable/traversing velocity component of the closed-loop control is generated from the control error via the servo gain factor Kv.

The mathematical (proportional) relationship is:
Servo gain factor Kv = (traversing velocity v / control error Δs) [1/s]

The servo gain factor Kv affects the following characteristic values:

- Positioning accuracy and holding control
- Uniformity of motion
- Positioning time

The better the axis design (high degree of stiffness), the higher the achievable servo gain factor Kv, and therefore the better the axis parameters are from a technological perspective (lower following error and higher dynamic response).

Precontrol

The velocity precontrol can be used to limit the velocity-related following error during position control. It can also help to achieve faster positioning.

With precontrol, the velocity setpoint is also switched additively to the position controller output. This additional setpoint can be weighed with a factor.

Balancing filter for feedforward control

The balancing filter is a simplified model of the speed control loop. It is used to prevent the position controller from overriding the manipulated velocity variable during the acceleration and deceleration phases. This is accomplished by delaying the position setpoint of the position controller by the balancing time with reference to the velocity precontrol.

Function of the balancing filter

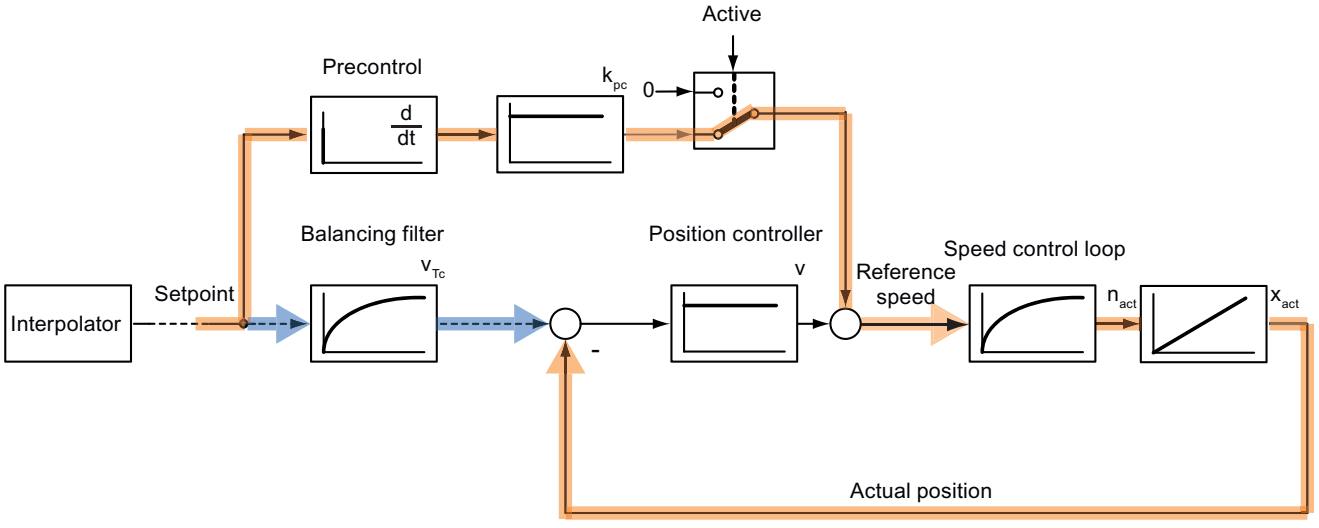


Figure 4-839 Balancing filter - Example of an electric axis without DSC

The speed is precontrolled; to this end, the position setpoint is adopted from the reference variable calculation directly or with differentiation applied, and the set velocity is defined on the speed controller directly. The position controller then simply has to compensate for any position errors which may be present in spite of the precontrol.

The position error, therefore, is generated from the position setpoint, delayed by the speed control loop equivalent time, and the existing actual position value.

Settings and configuration data

The setting can be made using the *Closed-loop control* dialog for *Dynamic controller data* (activate expert mode).

When precontrol is active, allowance can be made in the balancing filter for the response of the speed control loop prior to formation of the system deviation from the position setpoint and actual position.

- With the **balanceFilterMode=OFF** configuration setting, the balancing filter is switched off.
- With the **balanceFilterMode=MODE_1** configuration setting, a PT1 filter is used as the balancing filter.
  - For electric axis without DSC:
    The following must be set as the time constant for the balancing filter in **dynamicData.velocityTimeConstant**: $T_{Balancing\ filter}=T_{Speed\ control\ loop\ equivalent\ time}+T_i+T_o+T_{dp}+T_{servo}$
    The transmission dead times and process response (equivalent time of speed control loop) must be taken into account in the time constant.
    systemDeadTimeData.additionalTime is not included additively by the system in the balancing filter.
  - For electric axis with DSC:
    The following must be set as the time constant for the balancing filter in **dynamicData.velocityTimeConstant**: $T_{Balancing\ filter}=T_{Speed\ control\ loop\ equivalent\ time}$, because the transmission dead times for the position controller in the drive are not present in the position control loop.
    systemDeadTimeData.additionalTime is not included additively by the system in the balancing filter.
  - For the hydraulic axis:
    The following must be set as the time constant for the balancing filter in **dynamicQFData.qOutputvelocityTimeConstant**: $T_{Balancing\ filter}=T_{qOutputEquivalent\ time}$ + Communication times according to the setpoint output/actual value interface.
    ($T_{qOutputEquivalent\ time}$ as equivalent time for the QOutput process response)
- With the **balanceFilterMode=MODE_2** configuration setting (as of V3.1), the equivalent time of the speed control loop, the dead time determined by the system for the drive, and a dead time that can be input in additive increments by the user are taken into account in the balancing filter.
  Maximum time constant = 16 x T_{servo}
  - For electric axis without DSC:
    The following must be set as the time constant for the balancing filter in **dynamicData.velocityTimeConstant**: $T_{Balancing\ filter}=T_{Speed\ control\ loop\ equivalent\ time}$
    $T_i+T_o+T_{dp}+T_{servo}$ is taken into account by the system.
    The value set in **systemDeadTimeData.additionalTime** is included additively by the system in the balancing filter.
  - For electric axis with DSC:
    The following must be set as the time constant for the balancing filter in **dynamicData.velocityTimeConstant**: $T_{Balancing\ filter}=T_{Speed\ control\ loop\ equivalent\ time}$, because the transmission dead times for the position controller in the drive are not present in the position control loop.
    systemDeadTimeData.additionalTime is not included additively by the system in the balancing filter.

– For the hydraulic axis:
The following must be set as the time constant for the balancing filter in **dynamicQFData.qOutputvelocityTimeConstant**: $T_{Balancing\ filter}=T_{qOutputEquivalent\ time}$ + Communication times according to the setpoint output/actual value interface ($T_{qOutputEquivalent\ time}$ as equivalent time for the QOutput process response)

The setting **balanceFilterMode=MODE_2** on the balancing filter is recommended.

An overswing is apparent should the **dynamicData.velocityTimeConstant** value be too small. If the value is too large, the axis has limited dynamic behavior and creeps to the end position.

System deviation

The control deviation is the difference between the balanced setpoint and the actual value and is displayed in the **servoData.ControllerDifference** system variable.

Note

As of runtime version V4.1 SP1, the control deviation present on the position controller in the drive is displayed for DSC. This is calculated in the control using a model (see the Control Structure figure in the next chapter).

With versions prior to V4.1 SP1, the control deviation present in the control is displayed with DSC.

The **servoData.controlDifferenceCommandPosition** system variable (as of V4.4) displays the setpoint present in the drive. With reference to the current actual value, this setpoint is relevant to control deviation generation. The signal runtime and output times are taken into account.

Calculating the servoData.controlDifferenceCommandPosition system variable without balancing filter (configuration setting balanceFilterMode=OFF):

- Without DSC
servoData.controlDifferenceCommandPosition
= **servoData.symmetricServoCommandPosition**
- With DSC
servoData.controlDifferenceCommandPosition
= **servoData.symmetricServoCommandPosition** delayed by $T_i+T_o+T_{dp}+T_{servo}$

Calculating the servoData.controlDifferenceCommandPosition system variable with balancing filter (configuration setting balanceFilterMode<>OFF):

- Without DSC
servoData.controlDifferenceCommandPosition
= **servoData.symmetricServoCommandPosition** delayed by $T_i+T_o+T_{dp}+T_{servo}$
- With DSC or DSC with spline (requirement: position-control in the controller and in the drive is to the same encoder)
servoData.controlDifferenceCommandPosition
= **servoData.symmetricServoCommandPosition** delayed by $T_i+T_o+T_{dp}+T_{servo}+T_{additionalTime}$

Control loop structures

When the mode is changed from speed-controlled to position-controlled mode while the axis is in motion, the equivalent time of the position controller is required to apply the setpoint. The equivalent time is set during configuration in **dynamicData.positionTimeConstant** (for an electric axis) or in **dynamicQFData.positionTimeConstant** (for the hydraulic functionality).

Quantization of the control error for stepper motors or low-resolution encoders

The **commandValueQuantization.enable**=YES configuration data can be used to activate the quantization of the control deviation. A quantization of the control deviation is performed in accordance with the encoder resolution (distance per increment) or stepper motor increment.

This prevents, for example, the motor from oscillating between two increments while at a standstill.

With the setting **commandValueQuantization.mode**=DIRECT, the value for quantizing the control deviation can also be specified directly in **commandValueQuantization.value** (as of V4.1 SP1). This is sensible when for stepper motors, the encoder has a higher resolution than the increment of the stepper motor.

Note

Quantization of the control deviation should be activated for stepper motors.

Axis data sets

Like other items of axis configuration data, controller data can be assigned to multiple data sets. A data set is assigned to every axis when it is created.

Axis data sets are used as a means of activating multiple controller settings simultaneously.

Axis data sets are configured for axes on the **Axis data sets** tab under **Configuration**.

See also

Overview of commissioning the position controller of positioning axes (Page 2986)

Hydraulic axis with position control/velocity control (Page 3124)

Data sets (Page 3037)

Dynamic Servo Control (DSC)

With the **Dynamic Servo Control (DSC)** function, the position controller in the drive is executed in the cycle clock of the speed control loop. In the drive, intermediate setpoints are generated in the speed controller cycle clock from the position setpoints transferred in the position controller/communication cycle clock via linear fine interpolation.

It is thus possible to set a substantially greater position controller gain factor **K**v. This increases the dynamic response for the reference variable sequence and disturbance variable compensation for highly dynamic drives.

For a position axis with position control and an assigned SINAMICS drive, the system sets DSC by default provided this is available at the drive's DO type (e.g. when a servo drive is assigned).

Advantages of DSC (compared to a position controller in the controller)

- Higher servo gain factor Kv (position controller gain) possible
- Larger bandwidth -> higher dynamic response
- Shorter response times for disturbance characteristic

Detailed information

The position difference (XERR) and the gain factor for the position controller in the drive are transferred in the PROFIdrive telegram, in addition to the speed precontrol value.

To activate the **DSC** function, the position controller must be set as PV controller (P controller with precontrol). In addition, the encoder in **typeOfAxis.NumberOfEncoders.DSCEncoderNumber** on the axis must specify to which increments the position differential (XERR) is normalized during operation. In SINAMICS, this is by default the motor encoder. **typeOfAxis.NumberOfEncoders.DSCEncoderNumber** must be initialized to the first encoder of the axis TO.

DSC is supported by MASTERDRIVES (standard telegrams 5 and 6 according to PROFIdrive), SIMODRIVE 611U and SINAMICS S120 (additional SIEMENS telegrams 105 and 106).

A SCRIPT is available to support you when commissioning MASTERDRIVES.

In SINAMICS, SIMODRIVE 611U, and MASTERDRIVES, the motor measuring system is used for normalizing the position difference in the drive.

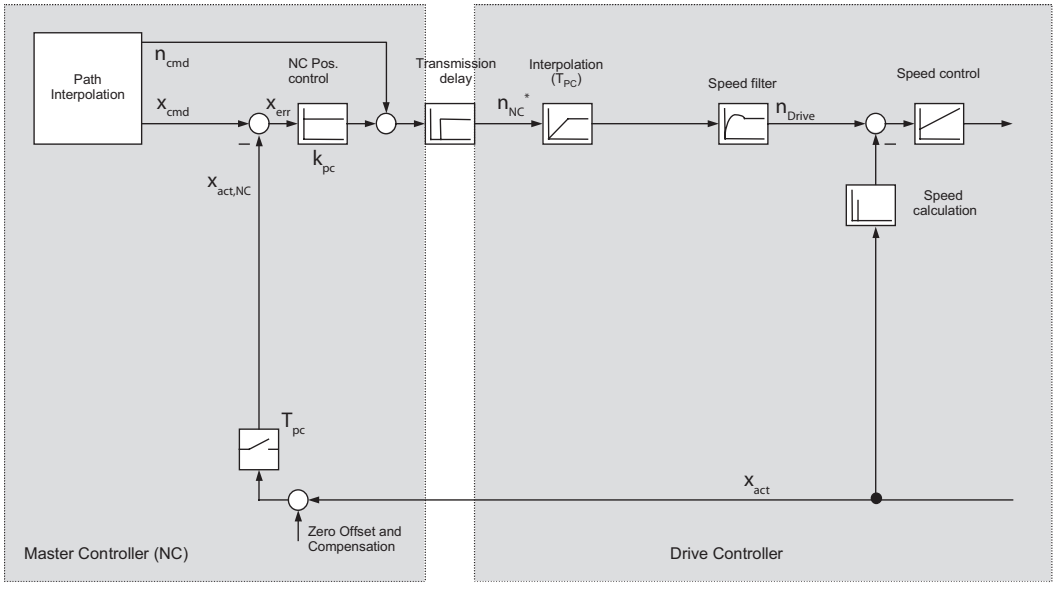
The following is to be taken into account for DSC:

- With DSC, XERR (position error) and Kpc (position control loop gain) are also transmitted, i.e., an 8-byte long setpoint telegram is required.
- With DSC, the communication times are taken into account in determining the following error of the setpoint since the actual comparison of setpoint and actual value is generated in the drive in the speed control cycle clock.
- The response time to a change in the actual position value is one speed controller cycle clock.
- With DSC and synchronous operation with actual value coupling, the extrapolation time must be increased by one position control cycle clock.

- If precontrol is deactivated for the DSC, zero is shown for the system variables with reference to the controller output or manipulated variable value.
  System variables:
  - **servodata.controllerOutput**
  - **actorData.setPoint**
  - **actorData.compensatedSetPoint**
  - **actorData.totalSetPoint**

  The axis is moved in accordance with the system deviation (XERR) given to the drive.
- If the direction in the drive is inverted for DSC p0410.0 and p0410.1 must be set the same, i.e. either the actual speed value and the actual position value are not inverted, or they are both inverted.

Structure



n_{cmd}: speed command x_{err}: position error command T_{pc}: position control sampling time (=T_{MAPC})
x_{cmd}: position command x_{act}: actual position k_{pc}: position control gain

Figure 4-840 Structure of the position-control loop with the velocity setpoint interface to the drive without DSC

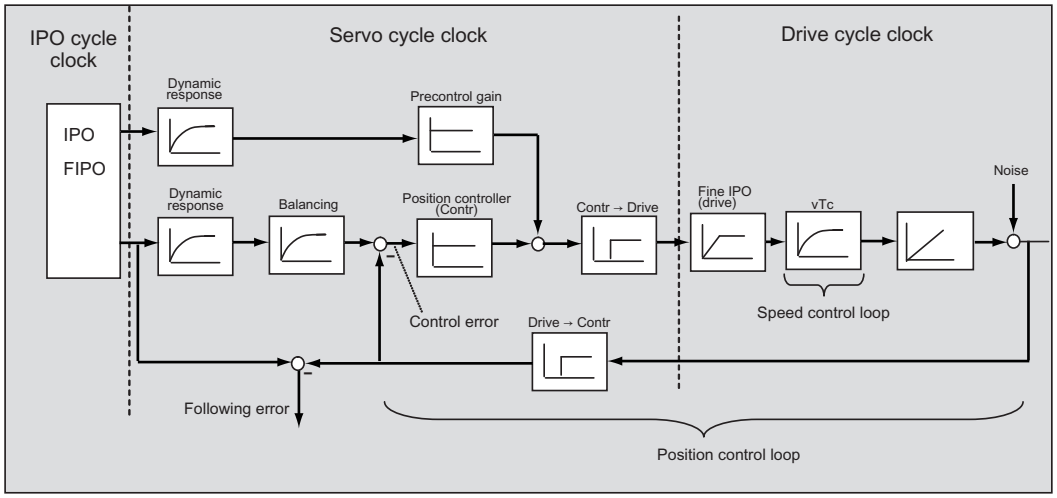
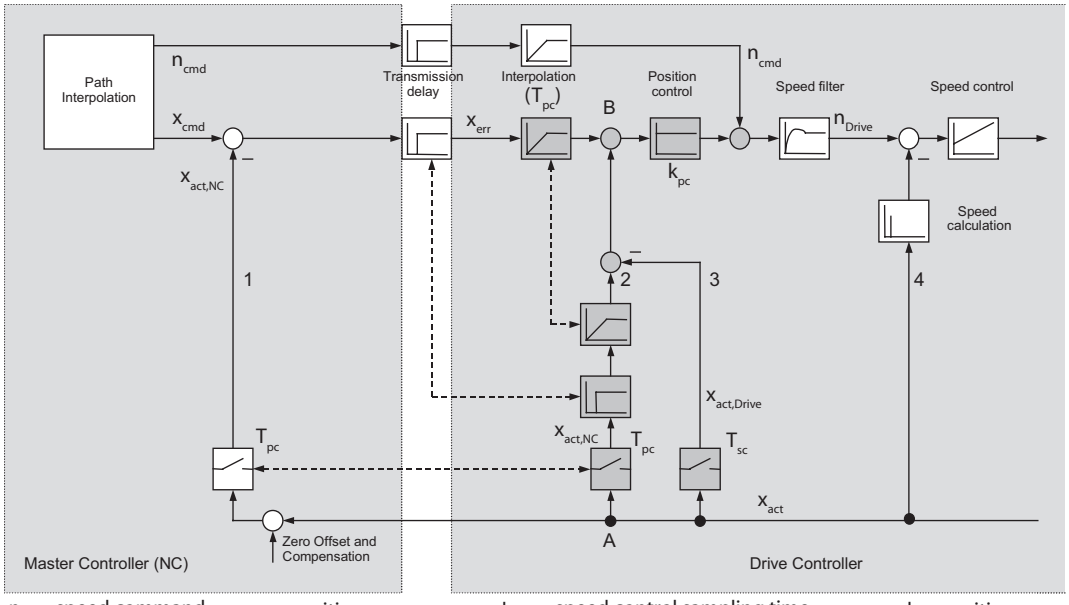


Figure 4-841 Control structure without DSC (simplified)



n_{cmd}: speed command x_{cmd}: position command x_{err}: position error command x_{act}: actual position $_{sc}$: speed control sampling time $_{pc}$: position control sampling time (=T_{MAPC}) k_{pc}: position control gain

Figure 4-842 Structure of the position-control loop with DSC functionality in the drive

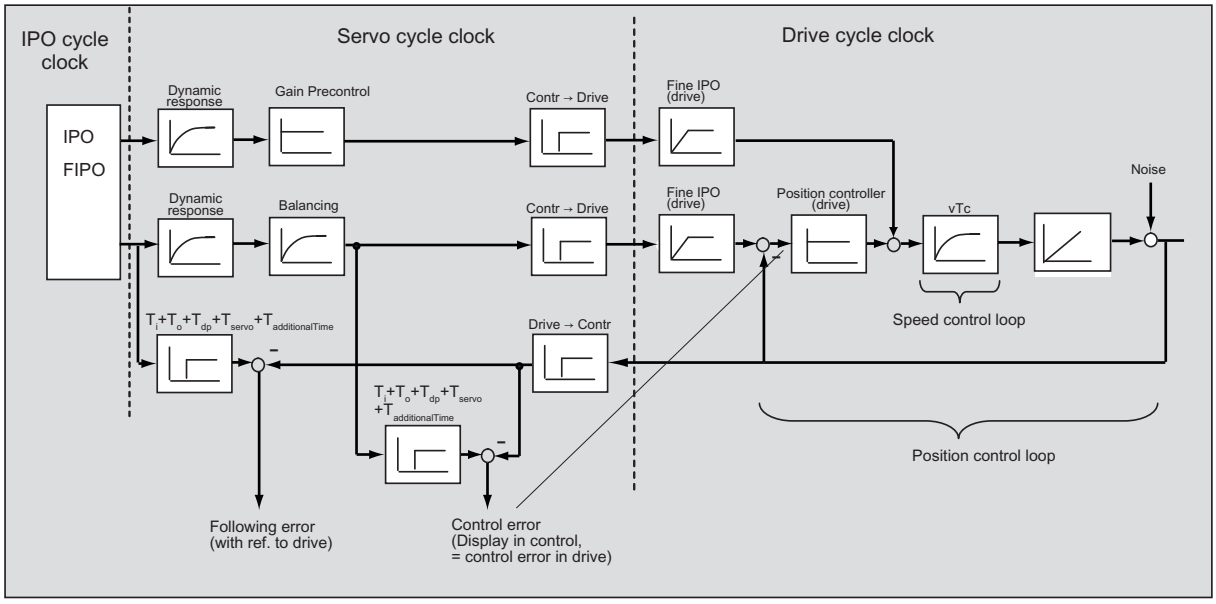


Figure 4-843 Control structure with DSC (simplified)

For more information about DSC, refer to the corresponding drive documentation, e.g. the SINAMICS S120 Function Manual for SINAMICS.

See also

Overview of commissioning the position controller of positioning axes (Page 2986)

DSC with spline (as of V4.4) [Expert]

With the **Dynamic Servo Control (DSC)** function, the position controller in the drive is executed in the cycle clock of the speed control loop. In the drive, intermediate setpoints are generated in the speed controller cycle clock from the position setpoints transferred in the position controller/communication cycle clock via linear fine interpolation.

With the **DSC with spline** function, the position controller is also executed in the drive in the cycle clock of the speed control loop. The intermediate values in the drive are generated via polynomial functions. Intermediate setpoints are also generated for speed and torque, in addition to position. Using the intermediate setpoints, highly dynamic motions down to the current controller cycle clock are simulated exactly and precontrol is possible right down to the torque.

To use DSC with spline, activate "DSC with spline" in the drive of the function module.

The setpoints are fine-interpolated in the following way:

- Position setpoint - cubic fine interpolation
- Speed setpoint - quadratic fine interpolation
- Acceleration setpoint/torque setpoint - linear fine interpolation

DSC with spline provides the following advantages and functionality:

- Linear fine interpolation and precontrol of the torque
- Extended support for highly dynamic motions
- Process response is determined by the lower current controller equivalent time

DSC with spline can be used or set as follows:

1. **DSC with spline and torque precontrol**
   - The torque is precontrolled.
   - Speed and position are controlled taking into account the equivalent time of the current controller in the balancing filter in the drive.
   - The torque is calculated in the drive from the acceleration and the total moment of inertia.
   - Balancing of the position setpoint and of the speed setpoint using the current controller equivalent time (ttc).

   The total moment of inertia J is used to calculate a torque from the acceleration.
   The following parameters are available in the drive for input:
   - p0341 Motor moment of inertia (in motors on the motor list this parameter is automatically preset)
   - p0342 Moment of inertia ratio Load+motor / motor (should be defined during speed controller optimization, reference model determination)
   - p1498 Load moment of inertia
   - p1497 Moment of inertia scaling factor
2. **DSC with spline and speed precontrol**
   - The speed is precontrolled.
   - The position is controlled taking into account the equivalent time of the speed controller (VTC) in the balancing filter in the drive.
   - The speed controller equivalent time is taken into account in the balancing filter.
3. **DSC with spline without precontrol**
   - Speed and torque are not precontrolled.
   - The position is controlled in the drive.
   - The balancing filter in the drive and in the controller is not included.

Use DSC with spline with torque precontrol for constant load torque and
DSC with spline with speed precontrol for non-constant load torque.

If the torque constant of the feed motor does not remain in the overload range, the function module **Extended torque control** functions can be activated. This improves the accuracy of the torque simulation.

Requirements and settings in SIMOTION

The following requirements must be satisfied for the use of DSC with spline:

- Axis type must be a real electrical axis (REAL_AXIS)
- Telegrams 125 and 126 must be supported by the drive
  (SINAMICS drives as of V4.7 that support telegrams 125 and 126)
- Activation of DSC with spline
  DSC with spline is first activated when the axis is powered up if the axis is at standstill (standstill signal active).
- Consistent setpoints
  Position and velocity setpoints must be consistent with each other, i.e. the setpoint pair must originate from the same cycle clock time of the motion control.
  In the setpoint calculation of the technology object, the position setpoints and the velocity are consistent with each other and are limited to the maximum values.
  If the setpoints are modified using compensation values or limits after the setpoint calculation, the consistency of position and velocity must be ensured by the user.
  If the setpoints are defined by the user, the user must ensure consistency.
  System variable with DSC spline status display: **servoData.dscSpline**
- Fine interpolation type
  Use the default setting (configuration data item **FineInterpolator._type**=CUBIC_MODE)
- High-resolution measuring system
  DSC with spline requires a high setpoint resolution. The setpoint resolution in the system is based on the actual value resolution. If the actual value only has a low resolution, you can increase the setpoint resolution internally.
  (Set p0418, e.g. to 18-bit)
- Recommendation: Set the equivalent time of the speed control loop and the current control loop to at least 3x current control cycle clock
  VTC configuration data item **dynamicData.velocityTimeConstant**
  TTC configuration data item **dynamicData.torqueTimeConstant**
- Switching the setting from DSC with spline to standard DSC and back
  controllerStruct.PV_Controller.enableDSCSpline configuration setting
  Restart required
- Speed control mode
  Performed via Kpc=0.
  The position for the generation of the position difference (XERR) is simulated internally.
- Speed precontrol = 100%
  Precontrol factor is not evaluated for the DSC with spline setting.
- Friction compensation
  This setting can cause irregular splines at the start of motion.
- Backlash compensation
  This setting can cause irregular splines at the direction reversal.

- Setpoint superimposition
  DSC without spline is used by the system, as long as superimposition is active, by activating system variable **servoSettings.additionalCommandValueSwitch**.
- Manipulated variable superimposition
  DSC without spline is used by the system, as long as superimposition is active, by activating system variable **servoSettings.additionalSetPointValueSwitch**.

Restrictions

The following restrictions apply in SIMOTION when using DSC with spline:

- Encoder switchover during operation or during a motion is not supported
- Pressure control is not possible
- Pressure limiting is not possible

Closed-loop control on a direct measuring system

A direct measuring system can be controlled on condition that the control is set to the same encoder system in the controller and in the drive.

The encoder for the position control in the drive is set in p1192.

The transmission factor between the motor measuring system (MMS) (motor encoder) and the direct measuring system (DMS) is set in p1193.

X_ERR is further normalized to the motor measuring system if not converted in the drive.

Encoder switchover during operation or during a motion is not supported.

Note

Position and velocity values must be consistent

Closed-loop control on a direct encoder in the controller in conjunction with closed-loop control on a motor encoder in the drive is not permitted. The position and velocity values would then no longer be consistent in the spline calculation.

The user must ensure the values are consistent with each other if the position setpoints and velocity setpoints change.

Balancing filter

When DSC with spline is activated, the balancing filter is calculated in the drive. The balancing time is transferred to the drive in the telegram. The setpoint to calculate X_ERR is therefore tapped before the balancing filter.

The controller calculates the actual control error in the drive taking into account the balancing filter.

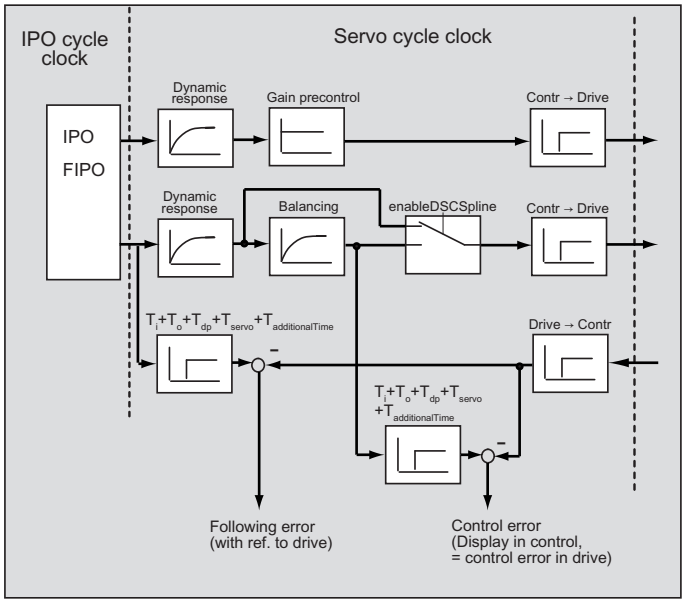


Figure 4-844 Effect with activated DSC with spline

Fine interpolation

The purpose of the fine interpolator (FIPO) is to generate interim setpoints for the position setpoints when the interpolator (IPO) and the controller (servo) have different cycle clock ratios.

Interpolation types

During configuration, the following interpolation types can be set by means of the **FineInterpolator._type** configuration data element:

- **DIRECT_MODE:** when no fine interpolation is required
- **LINEAR_MODE:** linear interpolation (continuous position for positioning axis)
  Use for discontinuous velocity setpoints (no precontrol)
- **QUADRATIC_MODE:** quadratic interpolation (continuous velocity for positioning axis)
  Use for continuous velocity setpoint curves
- **CUBIC_MODE (recommended, default setting):** cubic interpolation
  Use for continuous velocity or continuous acceleration setpoint curves

With the positioning-axis setting, the set position is interpolated.

With the speed-controlled axis setting, the set velocity is interpolated.

Dynamic controller data

The equivalent time of the current control loop is set in the **dynamicData.torqueTimeConstant** configuration data element. The equivalent time of the current control loop is not used at present.

The equivalent time of the speed control loop is set in the **dynamicData.velocityTimeConstant** configuration data element and used in the balancing filter. See also *Balancing filters for precontrol* in Position control (Page 2950).

The setting can be made using the *Closed-loop control* dialog for *Dynamic controller data* (activate expert mode).

The equivalent time of the position control loop is set in the **dynamicData.positionTimeConstant** configuration data element. The equivalent time of the position control loop is set in the following cases:

- Preassigned braking ramp
- Switchover from SPEED_CONTROLLED to POSITION_CONTROLLED
- Switchover from pressure-controlled to position-controlled operation
- A moving axis is enabled with **_enableAxis()**

If the equivalent time of the position control loop has not be set correctly, compensation movements can occur for switching tasks.

With DSC, the equivalent time of the position control loop can be set as follows:

- Without precontrol
  PTC = 1/Kv
- For 100% precontrol of the velocity setpoint:
  The equivalent time of the position control loop can be set as equal to the equivalent time of the speed control loop (PTC = VTC).
  (Control loop optimization for minimal overshoot)

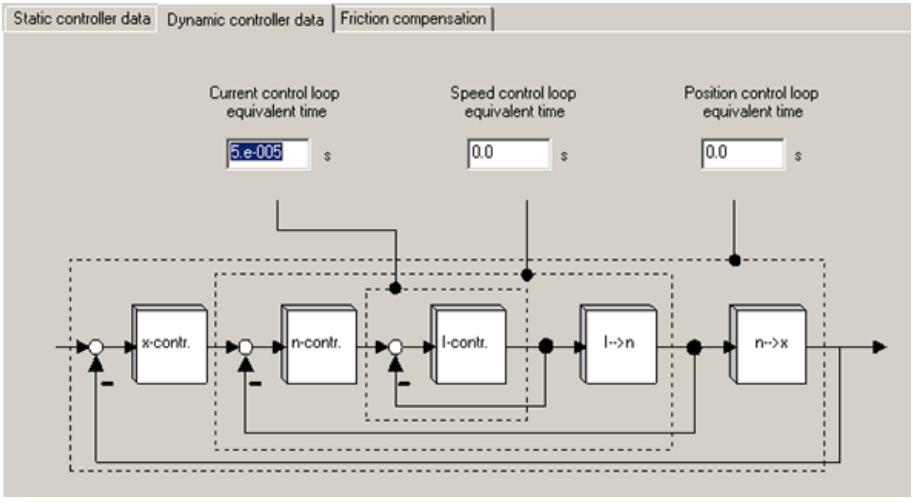
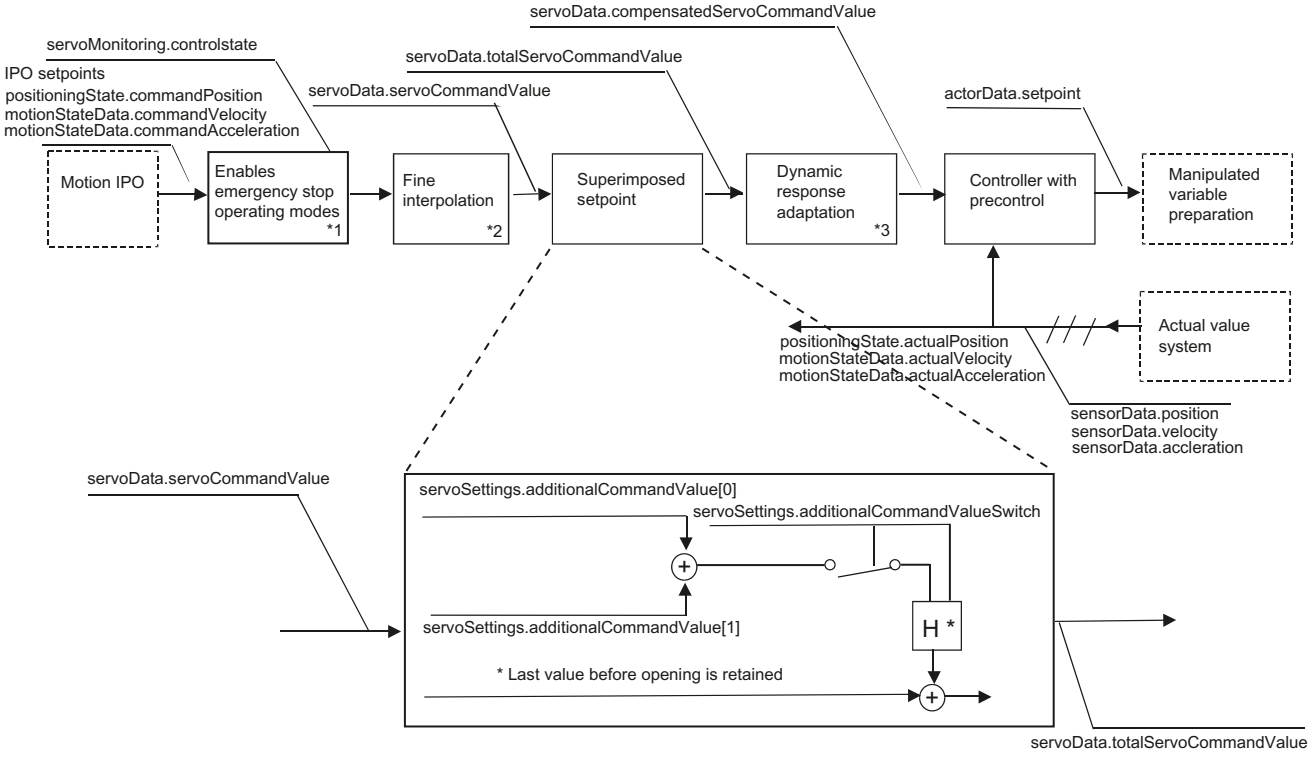


Figure 4-845 Dynamic controller data

Setpoint superimposition

The setpoint specified by the interpolator can be superimposed in the servo via the cyclically active system variables
(**servoSettings.additionalCommandValue**, **servoSettings.additionalCommandValueSwitch**)



*1 If there is no enable or emergency stop, IPO is in follow-up mode, i.e. IPO setpoint = IPO actual value

*2 Selectable: linear, cubic, quadratic, direct

*3 PT2 element with T1, T2

Figure 4-846 Setpoint superimposition

Superimposition has an effect on the position of the position axis. It is also effective during active position control and interpolator (IPO) in follow-up mode.

No setpoint superimposition occurs for traversing the axis in the SPEED_CONTROLLED mode and for active force/pressure control.

If the setpoint superimposition is disconnected, the last value of the setpoint superimposition is retained. If the value should no longer function, it must be explicitly set to zero.

Note

During execution of the alarm response FEEDBACK_EMERGENCY_STOP and the execution of **_stopEmergency()** with stopMode=STOP_WITH_COMMAND_VALUE_ZERO:

- The actual value (position and velocity) is accepted once and the preassigned deceleration ramp is applied
- The superimposed setpoint is unclamped (as of V4.0)
- The switch for the superimposed setpoint is opened (status displayed in system variable).
- A switch is prevented from closing or the switch is closed via the system variable and triggers Alarm 50021 "ServoSettings system variable (Element /1/%d) cannot be write-accessed due to a stop response".

With the FEEDBACK_EMERGENCY_STOP alarm response, there is no setpoint superimposition.

When a stop ramp is assigned (e.g. **_stopEmergency(**...WITH_COMMAND_VALUE_ZERO) **)**, the superimpositions are deleted and transferred to the setpoint generation. This triggers Alarm 50020 "Servosettings system variable (Element /1/%d) is reset due to a stop response".

For switching to SPEED_CONTROLLED (as of V4.1 SP1) and in pressure control (as of V4.1 SP1), the setpoint superimposition is not active.

Dynamic response adaptation

In order to adapt the dynamic behavior of axes, the setpoint branch contains a programmable PT2 setpoint filter with the time constants T_1, T_2, and T_t. This allows compensation for axes with higher dynamic behavior with the lowest dynamic behavior (axis with the largest equivalent time constant of the T_{LR} position controller).

The dynamic response of the axes is determined by T_{Res}, the resulting total time constant.

$T_{Res} = T_{LR\,1}$ (axis with the smallest dynamic behavior)

$T_{Res} = T_{da} + T_{LR\,2}$ (considered dynamic axis)

The dynamic adaptation, T_{da}, must be selected so that the resulting total time constants are equal for all axes to the adapted.

The dynamic adaptation consists of
$T_{da} = T_1 + T_2 + T_t$

T_1	additive time constant 1 (setting in the configuration data: **NumberOfDataSets.DataSet_1.DynamicComp.T1**)
T_2	additive time constant 2 (setting in the configuration data: **NumberOfDataSets.DataSet_1.DynamicComp.T2**)
T_t (V4.1 SP1 and higher)	Dead time (setting in the configuration data: **NumberOfDataSets.DataSet_1.DynamicComp.deadTime**)

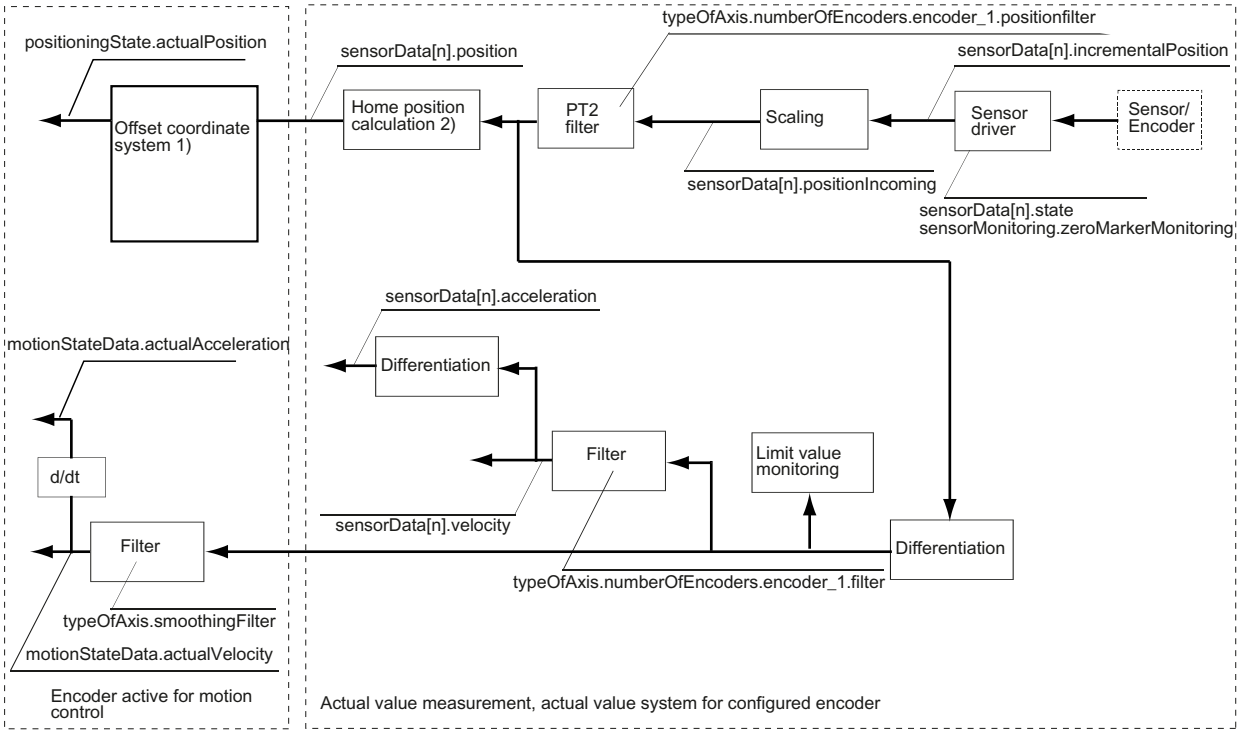
T_{Res}	(Desired) resulting total time constant of the axis
T_{LR}	Equivalent time constant of the closed position control loop of the axis (see **dynamic~Data.positionTimeConstant**)

The function is enabled/disabled via **dynamicComp.enable**.

Note

The used procedure means that a setpoint delay can be implemented exactly by specifying a dead time.

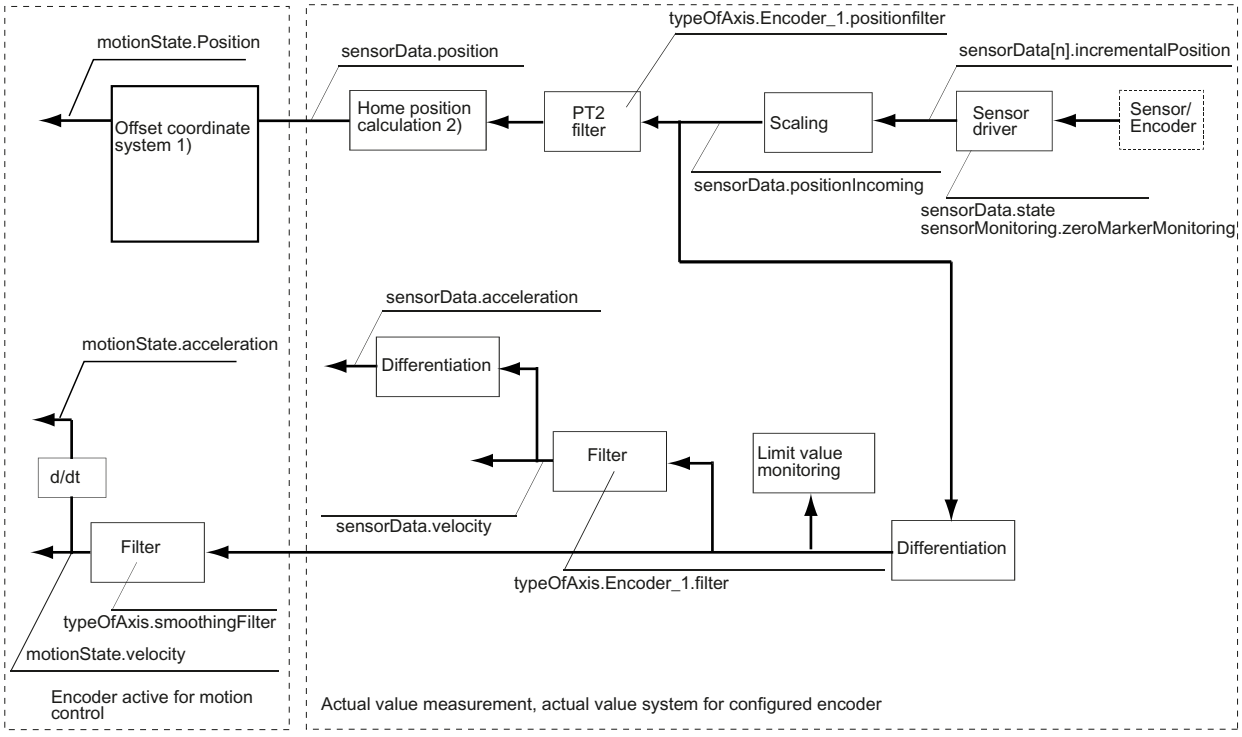
Actual value measurement / actual value system



1) Absolute and incremental encoder: Offset coordinate system e.g. using _redefinePosition() and _homing() with the DIRECT_HOMING setting
Incremental encoder: Home position userDefaultHoming.homePosition, home position offset incHomingEncoder.proceedshiftpos

2) Incremental encoder: Positions are zeroed at zero mark
Absolute encoder: Absolute encoder adjustment in absShift or absoluteEncoder.totalOffsetValue is included

Figure 4-847 Actual value system of technology object axis



1) Offset coordinate system e.g. using _redefineExternalEncoderPosition and _synchronizeExternalEncoder with the DIRECT_HOMING setting

2) Incremental encoder: userDefaultHoming.homePosition
Absolute encoder: absShift

Figure 4-848 Actual value system of technology object external encoder

The monitoring of the **actual velocity** and **actual acceleration** is used to identify errors in the control loop of the drives. If the rise in the actual value exceeds the **encoder limit frequency**, an **alarm** is triggered.

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

Display

The actual data for each sensor/encoder is displayed in the following variables:

- sensorData[n].position
- sensorData[n].velocity
- sensorData[n].acceleration

The system variables for **sensorData** are calculated in the servo cycle clock.

The actual axis value that is active for closed-loop control, the IPO cycle clock, and master value coupling is displayed in the following variables:

- positioningState.actualPosition
- motionStateData.actualVelocity
- motionStateData.actualAcceleration

The system variables for **positioningState** and **motionState** are calculated in the IPO cycle clock.

These actual values comprise the reference for the output cam calculation in the IPO cycle clock, the actual value coupling for external encoders without extrapolation, and the actual value reference in the IPO cycle clock, e.g. for actual-position-related profiles.

Noisy encoder signals lead to high velocity jumps. These can be reduced or compensated by using suitable filter settings.

Filtering of the velocity

The **smoothingFilter** configuration data element refers to the velocity calculated in the IPO cycle clock. Here, you can select whether a PT1 filter is to be applied to the data or whether the data is to be generated from the mean value. The mean value is determined from the ratio of the servo cycle clock to the IPO cycle clock.

The **numberOfEncoders.encoder_1.filter** configuration data element relates to the velocity calculated in the servo cycle clock. A PT1 filter is used.

During synchronous operation with master value reference to the actual axis values, these axis values are derived separately (refer to *Synchronous Operation description of functions, Actual value coupling*).

Actual position filtering (as of V4.1 SP1)

A sensor/encoder-specific actual position value filter is available.

This actual value filter is set in **typeofAxis.numberOfEncoders.encoder_1.positionfilter.T1** and **~.T2** and activated via **~.enable**.

The *positionFilter* has no effect with the velocity encoder setting **encoderValueType**=VELOCITY.

The actual velocity and actual acceleration are derived from the filtered position.

The *positionFilter* is calculated based on the actual values in SIMOTION and so does not act for active DSC for the actual values of the lower-level control loop in the drive.

The *positionFilter* present on the analog sensor input (for raw value filtering) is not dependent on the filter described here.

Extrapolation

For a synchronized group with actual value coupling (e.g. master value is the actual encoder value of an axis or an external encoder), the associated principle means delay times result because of bus communication, system cycle clocks and clock-pulse scaling, fine interpolation, position setpoint filters, and controller settings. These times can be compensated using an extrapolation (**Extrapolation.extrapolationTime**).

Extrapolation means knowing what has gone before and casting a glance at what is to come in the future (extrapolation time). In the case of dynamic changes to the master value, the extrapolation time should be as short as possible. An IPO cycle clock ratio: Servo pulse duty factor of 1:1 is good.

Note

Extreme care must be taken when changing the extrapolation time to the runtime; otherwise knocking could result in the machine.

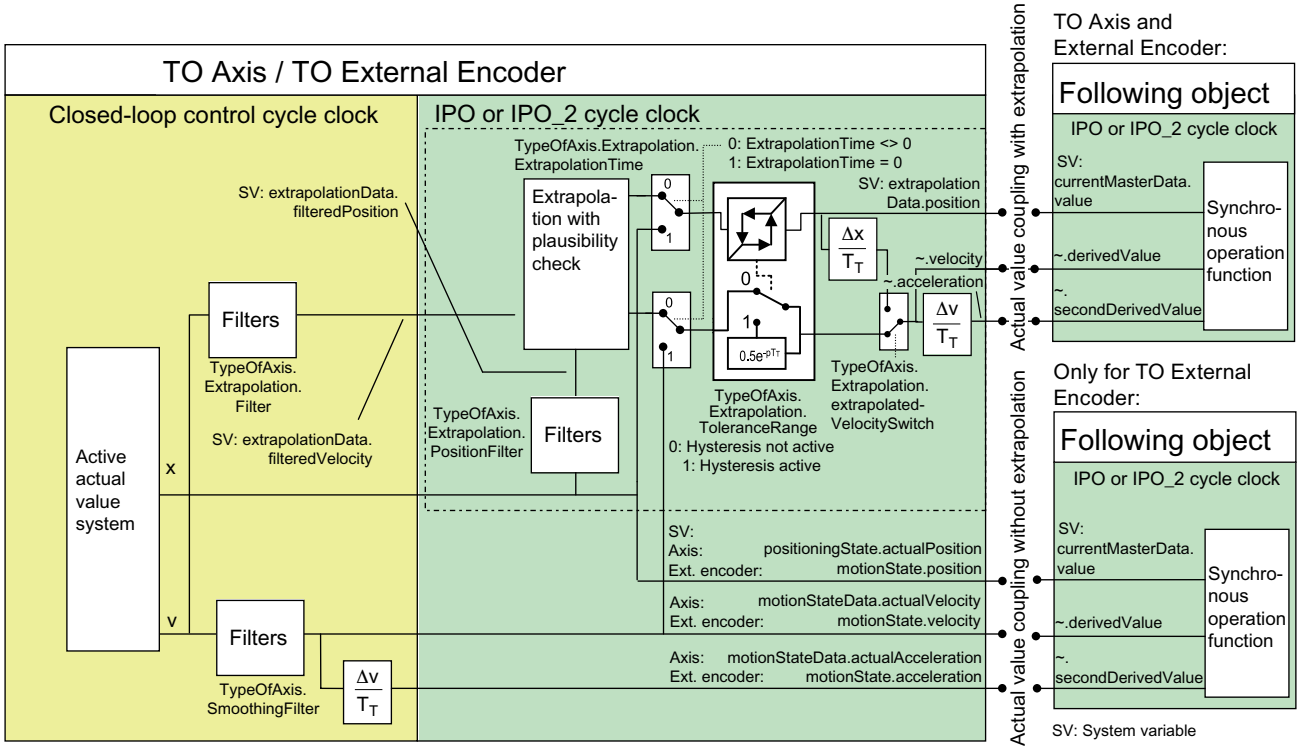


Figure 4-849 Actual value coupling with extrapolation for the Axis technology object or External Encoder technology object

During master value extrapolation, filtering on the actual velocity value is performed separately by means of a PT1 filter/mean value generation that is set with **typeOfAxis.extrapolation.Filter**.

The actual position value for synchronous operation can be filtered separately during extrapolation by means of a PT2 element. (as of V4.1 SP1)

As with extrapolation, there is only one filter for each axis rather than for each sensor/encoder. It is set in **typeOfAxis.extrapolation.positionFilter.T1** and **~.T2**. The filter acts on the actual position for the extrapolation, see also *Technology Objects Synchronous Operation, Cam* function manual, *Actual value coupling with extrapolation* section.

The velocity for the extrapolation is taken over from the actual values of the axis or External Encoder before application of the smoothing filter (**typeOfAxis.smoothingFilter**).

Although the velocity filter (**Extrapolation.Filter**) does not affect the extrapolation time, it is significant in the case of dynamic changes to the master value (due to the values being delayed).

The filter for the velocity during extrapolation is independent of this filter.

We recommend setting the velocity filter (**Extrapolation.Filter**) first; if the result is not sufficient, use the position filter as well. The position filter times are an additional factor to be taken into account in the extrapolation time.

For extrapolation, you can specify in **typeOfAxis.extrapolation.extrapolatedVelocitySwitch** whether the master value velocity should be recalculated from the extrapolated position or whether the velocity derived for the extrapolation should also be used as the master value velocity.

The extrapolated and filtered actual values can be checked in the following system variables:

- Filtered and extrapolated
  - **extrapolationData.position**
  - **extrapolationData.velocity**
  - **extrapolationData.acceleration**
- Filtered and not extrapolated
  - **extrapolationData.filteredposition**
  - **extrapolationData.filteredvelocity**

(This topic is presented in detail in the *Technology Objects Synchronous Operation, Cam* Function Manual under Actual value coupling with extrapolation.)

Note

You can find a tool to help you calculate extrapolation times under *Tools and Documentation > Calculating the extrapolation time for an actual-value-coupled drive* in the *SIMOTION Utilities & Applications*. *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

Transferring the actual velocity from the drive (as of V4.1 SP1)

With velocity precontrol, sudden changes in velocity directly affect the control behavior (quantization jumps primarily in the case of low-resolution encoders).

With the setting **typeOfAxis.numberOfEncoders.encoder_n.encoderValueType**=POSITION_AND_PROFIDRIVE_NIST_B, you have the option of converting the speed of rotation transferred in PROFIdrive NIST_B to a velocity and applying this value as the actual velocity of the encoder/sensor. In this case, the actual position of the sensor does not need to be differentiated to derive the actual velocity.

With the setting **typeofAxis.numberOfEncoders.encoder_n.encoderValueType**= POSITION_AND_DIRECT_NIST, a speed of rotation transferred in the I/O area and normalized as NIST_B is taken as the actual value and converted to an actual velocity. In this case, 4000H corresponds to 100%. The address is set in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.logAddress**, and the reference value is set in **typeofAxis.numberOfEncoders.encoder_n.sensorNist.referenceValue**.

With encoders with NAct evaluation, the speed determined by the encoder and the resulting velocity can be accepted by the encoder. In this case, the actual position of the sensor does not

need to be differentiated to derive the actual velocity.
Two transmission options are available:

- Transmission in the PROFIdrive telegram
- Transmission in the I/O area

Number of modulo revolutions (as of V3.2)

The number of modulo revolutions is displayed in the **positioningState.commandModuloCycles**, **positioningState.actualModuloCycles** and **sensorData.moduloCycles** system variables with the following constraints:

- The value is not initialized at first.
  The start value must be stored in the user program so that, for example, the number of revolutions since the start can be calculated subsequently.
- The value is reset when the system is switched on and in response to initialization tasks such as set actual value system (**_redefinePosition()**) or homing (**_homing()**).
  The modulo revolution can change via **_redefinePosition()**. For absolute position specifications and stationary axes, the principle of the shortest path is used; for moving axes, corrections are made in the direction of motion. For relative position specifications, the correction is made over the entire position difference.
- There is no special overflow handling for the count value.
  A counter overflow must be taken into account in the user program.

Table 4-351 System variables for determining the modulo revolutions

Variable	State	Meaning
positioningState.commandModuloCycles	See the **StructAxisPositioningState** data type in the System variables parameters manual	Setpoint for modulo revolutions
positioningState.actualModuloCycles	See the **StructAxisPositioningState** data type in the System variables parameters manual	Actual value for modulo revolutions
positioningState.moduloCycles	See the **StructAxisSensorData** data type in the System variables parameters manual	Actual value for modulo revolutions

Preparation of manipulated variables for electric axis

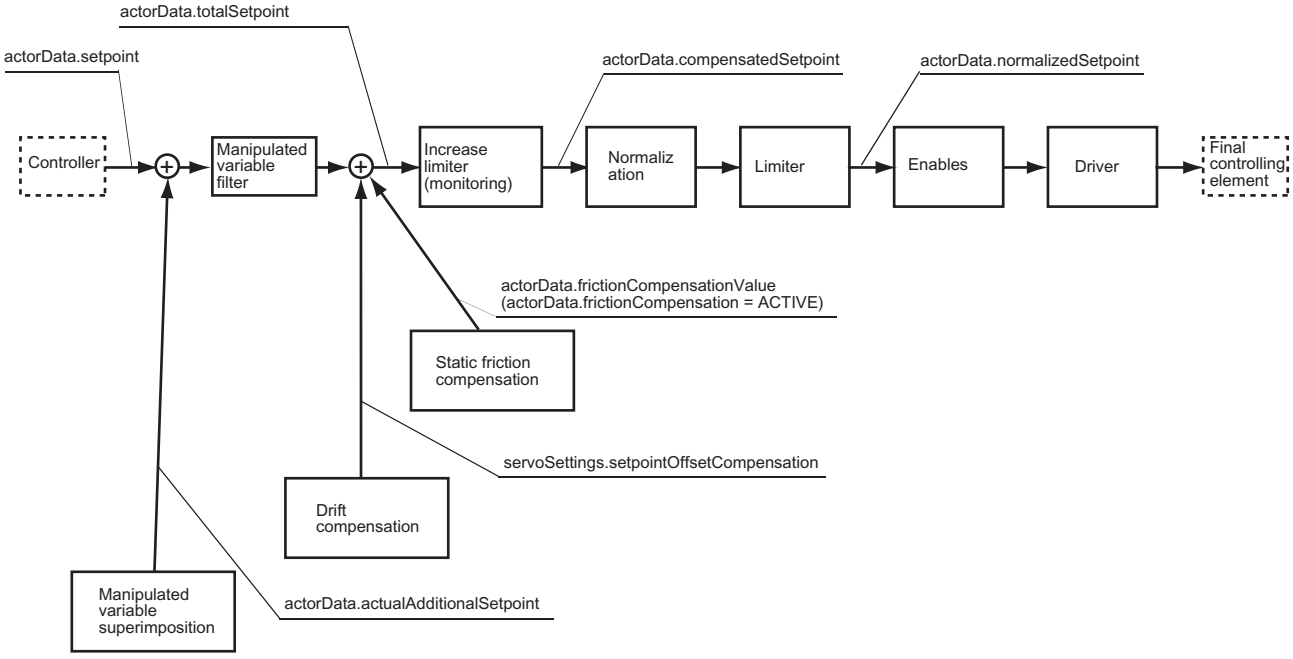


Figure 4-850 Manipulated variable preparation

Manipulated variable superimposition

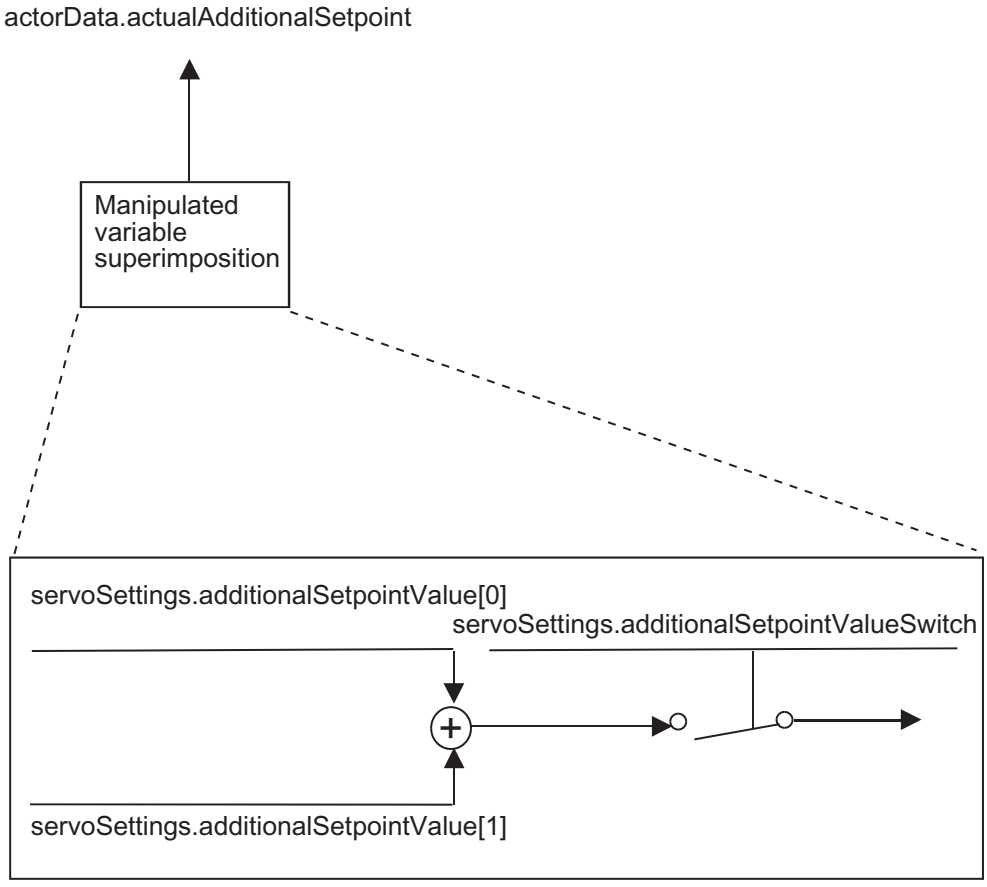


Figure 4-851 Manipulated variable superimposition

Superimposition of manipulated variables is enabled by means of a switch.

Manipulated variable superimpositions remain in effect when the drive is active. The user is responsible for handling the superimpositions.

As of V4.1 SP1, the superimposition of manipulated variables is not active for the FEEDBACK_EMERGENCY_STOP alarm reaction and the **_stopEmergency()** command with stopMode=STOP_WITH_COMMAND_VALUE_ZERO.

Manipulated variable filtering (as of V4.1 SP1)

A man. var. filter can be set as a PT1 filter in the **setpointFilter** configuration data element. This filter acts after the controller and after the precontrol value and the additive manipulated variable value (additionalSetpoint) have been added.

A change in the filter data takes effect immediately.

With the DSC setting, the man. var. filter is only effective for the precontrol.

Drift/offset compensation

The analog output signal of analog-coupled drives can include a **drift**. This can be compensated for by an offset in the axis.

Drift is enabled/disabled using the **DriftEnable** configuration data element. The value is specified in the system variable **servoSettings.setpointOffsetCompensation**.

The value of the system variable **servoSettings.setPointOffsetCompensation** is reset to 0.0 with **_disableAxis()** (or with the alarm response RELEASE_DISABLE). If a value not equal to 0.0 is effective when the axis is switched on, the value needs to be reset before switching on the axis (before issuing **_enableAxis()**).

Static friction compensation

A simple compensation is available for overcoming for the forces of static friction. During startup from a standstill, a DT1 element adds a static friction compensation signal to the manipulated variable.

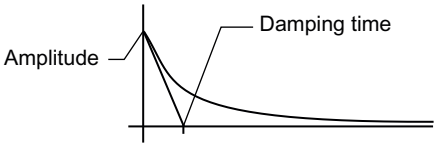


Figure 4-852 Static friction compensation

Static friction is added relative to the velocity setpoint. It only takes effect when processing motion specifications and does not affect force/pressure control.

The standstill identification for static friction compensation can be set separately, as is the case for the amplitude and the decay response. The amplitude and decay response are set in the configuration.

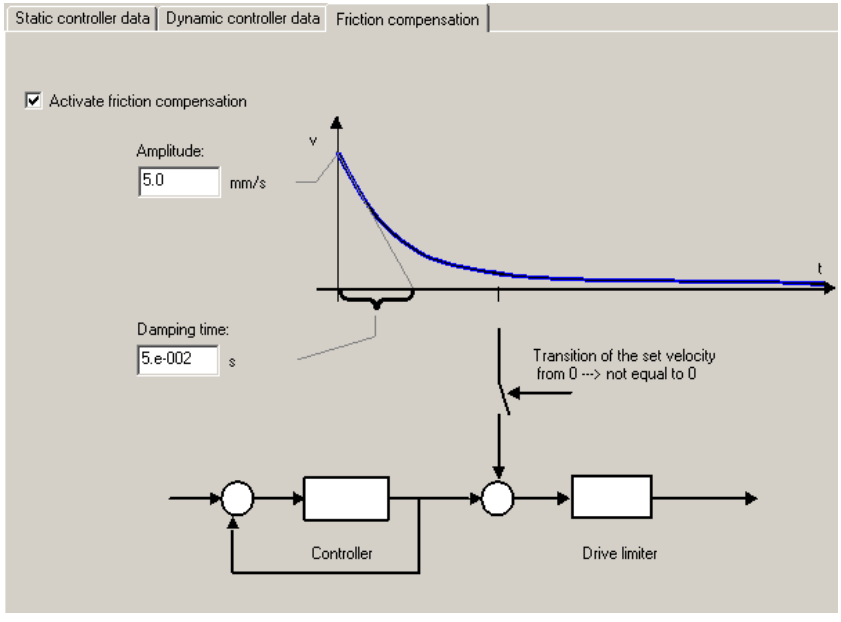


Figure 4-853 Static friction compensation

Backlash on reversal compensation

During the power transmission between a moving machine part and the corresponding drive, a backlash on reversal (play) usually occurs.

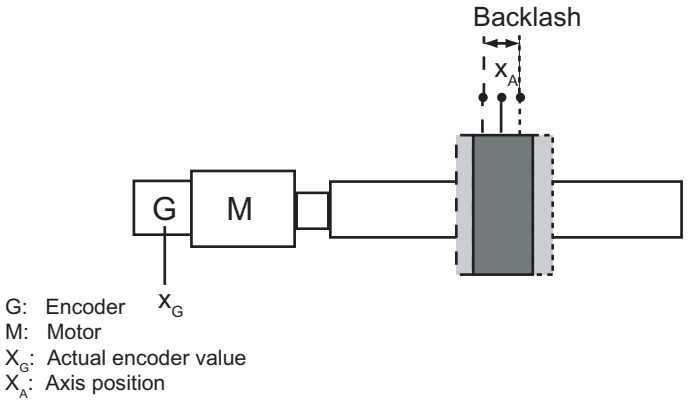


Figure 4-854 Backlash

Despite the backlash on reversal, a clear derivation of the axis position from the encoder position and an exact traversing, positioning and synchronous operation of the axis must be possible. SIMOTION provides the backlash on reversal compensation function for this purpose.

The backlash is specified in **absBacklash.length** or **incBacklash.length**, and the backlash compensation velocity is specified in **absBacklash.velocity** or **incBacklash.velocity**.

Backlash on reversal compensation is effective only for encoders mounted on the motor side and is set specifically for each encoder. A direct measuring system measures the variable directly without any intermediate backlash. Therefore, any existing backlash is compensated with closed-loop control to the direct measuring system.

Backlash type

- Positive backlash
  A positive backlash is set with **absBacklash._type** = POSITIVE or **incBacklash._type** = POSITIVE.
  Setting =POSITIVE means that the mechanical position lags behind the actual encoder value. This is the case, for example, if the ball screw has play and the encoder is attached to the motor (normal condition, default).
  When the direction is reversed, the backlash is applied by the system at the backlash compensation velocity.
  Unless stated otherwise, the POSITIVE setting is assumed in the following.
- Negative backlash
  The setting **incBacklash._type** = NEGATIVE is not supported.
  The setting **absBacklash._type** = NEGATIVE is not supported.

Homing with incremental encoders

In homing, the encoder value is assigned a unique (mechanical) axis position based on the reference signal of a homing mark.

If there is a backlash, the axis must always be traversed from the same side to the synchronization point during homing. The assignment of the actual control position to the mechanical position of the axis is thus unique.

This applies to:

- Homing on encoder zero mark
- Homing on external zero mark
- Homing via actual value setting (setting the actual axis value to a specified value)

Note

The fraction backlash that the axis lags when traversing in the homing direction is irrelevant. Homing yields a unique assignment of the mechanical and displayed axis position relative to the encoder value. When traversing in this direction, the same ratios always result.

Direction reversal and switch-on behavior of incremental encoders

In a direction reversal when **incBacklash._type**=POSITIVE, the motor runs through the backlash range. During this motor motion, the mechanical and thus the displayed actual position of the axis does not change, whereas the encoder value in **sensordata.incrementalPosition** does change. The axis is then traversed by the commanded distance or to the commanded position.

In case of direction reversal, the backlash compensation is independent on the 'homed' status. However, the first motion after the control unit is switched on is run through without backlash compensation.

When the backlash range has been run through completely one time (no matter which direction), the set backlash is then compensated when the direction is reversed - if backlash compensation is enabled. This is independent of the homing status and applies to the relative or, if applicable, absolute traversing of the axis in the non-homed state.

Homing with absolute encoders

The absolute encoder value is assigned a mechanical axis position by defining the absolute encoder offset for homing with absolute encoders or absolute encoder adjustment.

This results in a direction dependency for the absolute encoder, too, since the position of the backlash relative to the encoder value/axis position is relevant when setting the absolute encoder offset or the mechanical axis position.

After the absolute encoder adjustment, backlash on reversal is carried out if the direction is reversed, but not when traversing is continued in the same direction.

If the control unit is switched off/on, the mechanical axis position is assigned/indicated to the actual encoder value via the absolute encoder offset. In the same way as motion restart after absolute encoder adjustment, backlash compensation is not activated after switch-on for motion in the same direction as the absolute encoder adjustment; however, it is activated for motion in the opposite direction.

In the **absBacklash.startupDifference** configuration data element, the direction opposite to the reference direction must be entered for setting the absolute encoder offset. For example, **absBacklash.startupDifference**= NEGATIVE must be set for referencing the absolute encoder offset to a positive direction of motion, since the backlash must be compensated in this case

when immediate travel in the negative direction occurs after switch-on. This is independent of the position of the backlash relative to the actual axis position at the time of switch-on.

Note

Directly after switch-on, the mechanical axis position is displayed correctly only if the position of the backlash at the time of switch-on corresponds to the position of the backlash relative to the mechanical axis position when the absolute encoder offset is set. If this is not the case, the mechanical axis position may deviate from the displayed axis position up to the total amount of the backlash; that is because the control detects the actual encoder value at the time of switch-on but it is not able to establish the position of the backlash without moving the axis.

Direction reversal and switch-on behavior of absolute encoders

With direction reversal and the setting **absBacklash._type**=POSITIVE, the motor runs through the backlash range. During this motor motion, the mechanical axis position and, thus, the displayed actual position of the axis does not change, whereas the encoder value in **sensordata.incrementalPosition** does change (also with the absolute encoder). The axis is then traversed by the commanded distance or to the commanded position.

Backlash compensation for direction reversal is independent of the 'Homed' status (here, the absolute encoder adjustment).

Backlash compensation is not activated for the first motion in both directions once the control has been switched if no absolute encoder adjustment has yet been carried out. When the backlash range has been run through completely one time (no matter which direction), the set backlash is then compensated when the direction is reversed - if backlash compensation is enabled.

Status display

The **sensorMonitoring.passingBacklash** system variable indicates that the backlash is being run through without changing the actual axis value.

Since the travel specification resulting from the specified motion and backlash recovery is superimposed, this indication is not identical to the specification for backlash recovery.

Note

In order not to impair the response times with the backlash, the backlash compensation is started simultaneously with the motion.

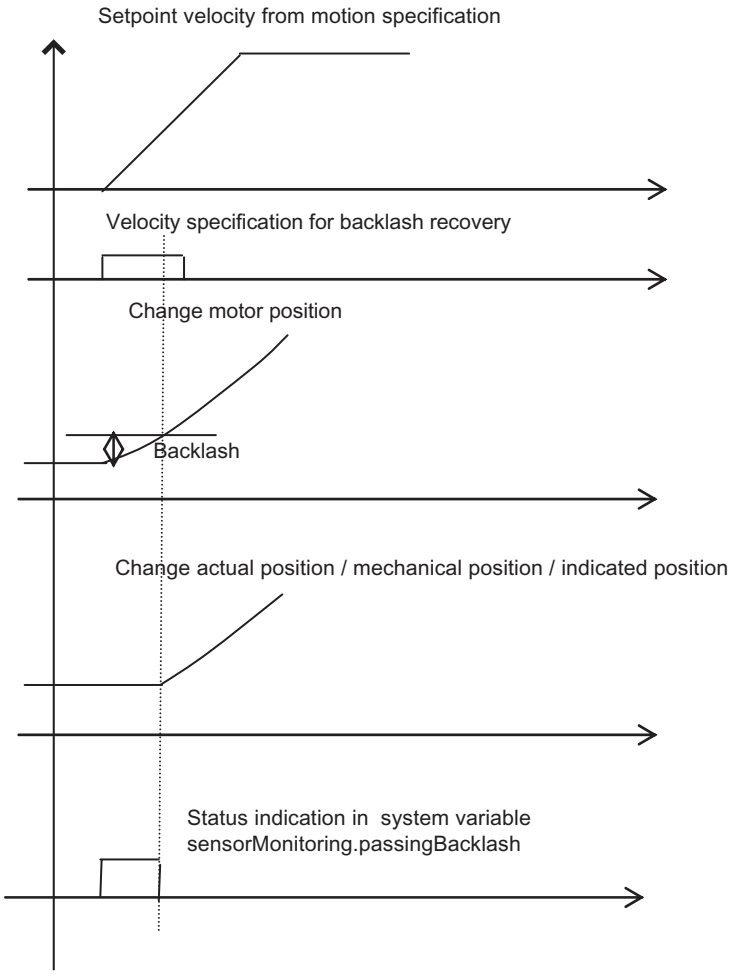


Figure 4-855 Backlash on reversal compensation sequence

Traversing of the positioning axis without position control

The positioning axis can also be traversed without active position control.

Traversing motions with **_move()** or as velocity profiles can be applied on the position axis/ synchronized axis with position control or, alternatively, with a velocity specification only.

Table 4-352 Setting via movingMode parameter in command

Commands	Function
_move()	For traversing with a programmable velocity profile
_runTimeLockedVelocityProfile()	For traversing with a user-definable velocity profile
_runPositionLockedVelocityProfile()	For traversing with a position-related velocity profile

The transition from a position-controlled or open-loop/closed-loop pressure-controlled motion to a motion solely with a velocity/speed specification can take place when the axis is at a standstill or in motion; this is also the case for the transition from a motion with speed specification to a position-controlled motion.

The **decodingConfig.speedModeSetpointZero** configuration data element can be used to specify whether the current velocity is maintained during the switchover from position-controlled or open-loop/closed-loop force-/pressure-controlled operation to operation with velocity/speed specification, or whether the velocity is set to zero at the time of switchover.

The dynamic response parameters and the maximum values for speed specification are derived from the settings for position-controlled axis operation.

The position-related monitoring functions are deactivated. The encoder limit frequency can be exceeded.

It is possible to activate the position-controlled axis with **_enableAxis()** for the velocity specification only.

See also

Setting and canceling the axis enables (Page 3143)

Moving (Page 3158)

Traversing the axis via velocity specifications (Page 3002)

Stepper drives

The special torque characteristics of a stepper motor and the response to overloading should be considered when assigning the stepper motor axis parameters.

Functionality of stepper motor axes

The Axis technology object can be used to implement the following functions:

- Positioning axis without additional encoders
  - Homing with edge of external zero mark
  - Rotation monitoring with external zero mark
- Position axis with additional incremental or absolute encoder
  - The axis works in the same way as with a servo motor with an analog interface +/-10 V.

Behavior of a stepper motor

From a given speed of rotation of the stepper motor (around 500 rpm), the torque produced by the motor drops logarithmically, and approaches zero at a maximum speed (around 3,000 rpm). The specific data can be found in the data sheet for the motor used.

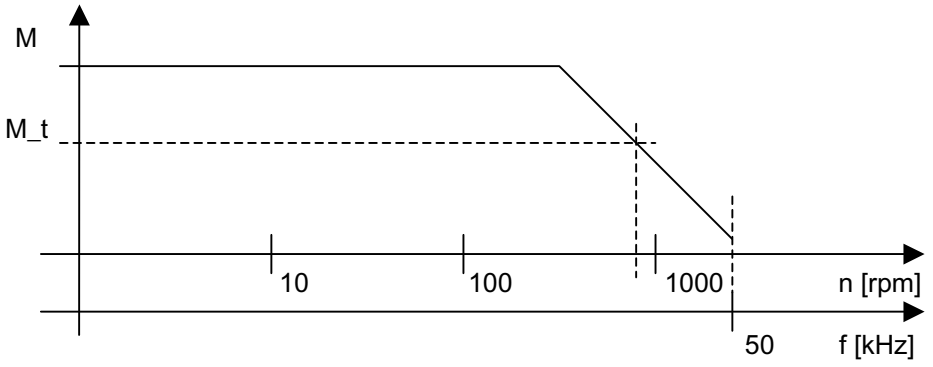


Figure 4-856 Example of a stepper motor torque characteristic

Consequences of operation in the overload range

If the stepper motor is ever unable to produce the requested torque, it loses synchronization with the predefined frequency and its speed drops suddenly. This can lead to standstill.

In this state, motion cannot be resumed unless a setpoint of 0 is entered in the meantime.

For a position axis without additional encoders, the traversing position and, as a result, the synchronization of the axis are lost.

Avoiding operation in the overload range

When the axis is designed, a maximum motor speed should be determined using the torque M_t required by the process. This maximum speed corresponds to the maximum velocity of the axis. The maximum frequency of the stepper motor must not be exceeded.

Encoder signal output (V4.0 or later)

The real axis with the setting **typeOfAxis**=REAL_AXIS_WITH_SIGNAL_OUTPUT can be used to output encoder signals via the SINAMICS TM41 module.

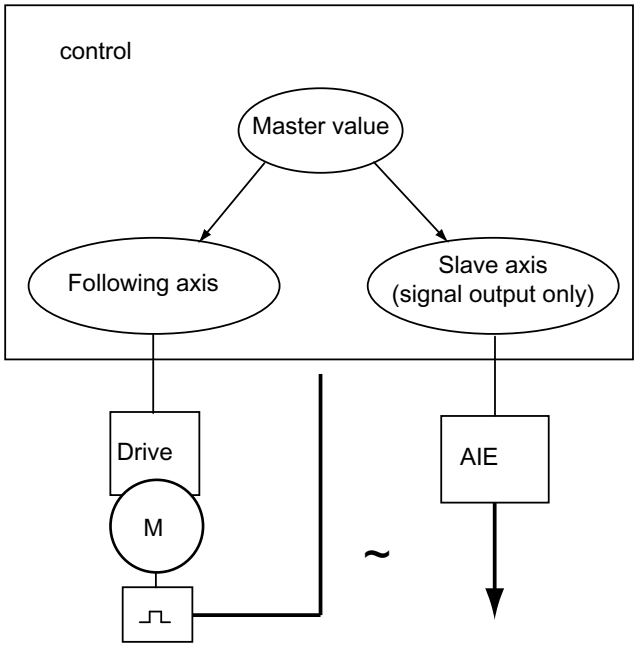


Figure 4-857 Using the axis for encoder signal output

Application

The axis position (a master value) is to be made available to a second control unit as an encoder signal via encoder signal simulation.

The standard message frame 3 is set between control unit and drive. An offset for the zero pulse output can be specified in the application.

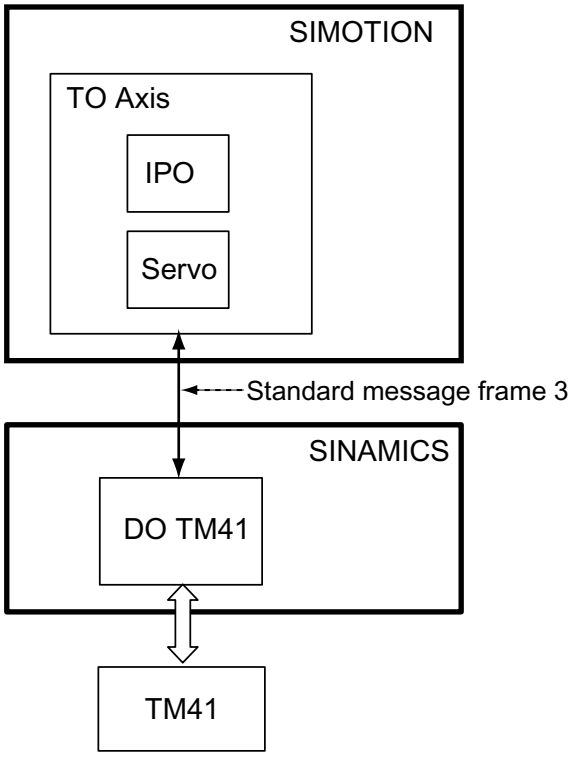


Figure 4-858 Interface of TM41 / DO41 to Axis technology object

In order to rule out lasting position deviations in the output signal, the axis can be provided with a PI controller that is applied to the actual position returned by TM41 / DO41.

Setting as an axis with signal output via the TM41 module only

Setting the axis as an axis with signal output only via the enumerator element **REAL_AXIS_WITH_SIGNAL_OUTPUT in TypeOfAxis**.

The following functions are not practicable or cannot be activated with this setting:

- Active homing is not permitted
- Measurement via digital drive is not permitted
- Compensations are disabled
- Following error monitoring functions are disabled
- Following error monitoring is disabled
- Positioning monitoring is deactivated
- Standstill monitoring is disabled
- Hardware limit monitoring is disabled

Note

You can also find further information under *FAQs* in the *SIMOTION Utilities & Applications* provided with SIMOTION SCOUT.

See also

Setting as a real axis with encoder signal simulation (V4.0 and higher) (Page 2898)

4.8.3.12 Commissioning the position controller of positioning axes

Overview of commissioning the position controller of positioning axes

Basic procedure

When commissioning the position controller, you should follow a basic procedure, which is outlined below. You can find detailed information in subsequent chapters.

- Enter/check the settings for the controller type, load gear, measuring gear, normalization of the speed setpoint interface, leadscrew pitch, encoders, and valve characteristic (for hydraulic axes)
- Set the speed controller on the drive
- On the SIMOTION control unit, preselect the axis data set (Page 3037) whose controller parameters are to be optimized (and, if necessary, preselect the appropriate data set on the drive)
- Set the controller parameters
  The SIMOTION measuring functions (Page 3073) for manual optimization and the automatic controller setting (Page 3066) are available for this.
- Set the equivalent time constant of the position controller
- Non-volatile storage of parameters on the target device and in the offline project

You can also find a commissioning guide under *FAQs* in the *SIMOTION Utilities & Applications* provided with SIMOTION SCOUT.

See also

Overview of automatic controller setting (as of V4.1 SP1) (Page 3066)

SIMOTION measuring functions (Page 3073)

Axis control panel (Page 3078)

Data sets (Page 3037)

Configuration data

Configuration data in the axis data set

The configuration data of a data set is contained in the structure **TypeOfAxis.NumberOfDataSets.DataSet_n** (n: data set number).

Table 4-353 Configuration data in the axis data set

Parameter	Configuration data element
General position control parameters	
Controller type	controllerStruct.conType
VTC Equivalent time of the speed control loop (time constant of the balancing filter). With the MODE_2 setting in the balancing filter, VTC is effective with a maximum value of 16 servo cycle clocks.	• Electric axis Axis data set: dynamicData.velocityTimeConstant • Hydraulic axis controllerStruct.DynamicQFData.qOutputTimeConstant
PTC Equivalent time of position control loop	• Electric axis dynamicData.positionTimeConstant • Hydraulic axis dynamicQFData.positionTimeConstant
Proportional-action position controller with precontrol for electric drives (controller type = PV)	
Balancing filter type	controllerStruct.PV_Controller.balanceFilterMode
Activation of Dynamic Servo Control (DSC)	controllerStruct.PV_Controller.enableDSC
Activation of DSC with spline	controllerStruct.PV_Controller.enableDSCSpline
Kpc Weighting of velocity precontrol	controllerStruct.PV_Controller.kpc
Kv P controller gain	controllerStruct.PV_Controller.kv
Activation of velocity precontrol	controllerStruct.PV_Controller.preCon
PID position controller with/without actual-value-dependent D component for hydraulic drives (controller type = PID/ PID_ACTUAL)	
Balancing filter type	controllerStruct.PID_Controller.balanceFilterMode
Activation of the integrator limitation.	controllerStruct.PID_Controller.enableAntiWindUp
Delay time of DT1 element	controllerStruct.PID_Controller.delayTime
Kd D-component gain	controllerStruct.PID_Controller.kd
Ki I-component gain	controllerStruct.PID_Controller.ki
Kp P-component gain	controllerStruct.PID_Controller.kp
Kpc Weighting of velocity precontrol	controllerStruct.PID_Controller.kpc
Activation of velocity precontrol	controllerStruct.PID_Controller.preCon

Parameter	Configuration data element
Mechanics	
Load gearbox	• Load revolutions Gear.denFactor • Motor revolutions Gear.numFactor

Table 4-354 Configuration data that is not included in the axis data set

Parameter	Configuration data element
Normalization of drive interface (electric drive)	
Maximum speed of rotation (rotary drive)	TypeOfAxis.setPointDriverInfo.DriveData.maxSpeed
Maximum traversing velocity (linear drive)	TypeOfAxis.setPointDriverInfo.LinearMotorDriveData.maxSpeed
Normalization speed (rotary drive)	TypeOfAxis.setPointDriverInfo.DriveData.nominalSpeed
Reference velocity (linear drive)	TypeOfAxis.setPointDriverInfo.LinearMotorDriveData.nominalSpeed
Reference for normalization speed (rotary drive)	TypeOfAxis.setPointDriverInfo.DriveData.speedReference
Reference for reference velocity (linear drive)	TypeOfAxis.setPointDriverInfo.LinearMotorDriveData.speedReference
Mechanics	
Leadscrew pitch	leadScrew.pitchValue
Fine interpolation	
Fine interpolator in the servo cycle clock	Fineinterpolator._type
Encoder	
Encoder parameters	The encoder settings are contained in the encoder data set: TypeOfAxis.NumberOfEncoder.encoder_x (x: Number of the encoder data set)

Example of commissioning a proportional-action controller with precontrol

The procedure for determining the control parameters of an electric axis manually is outlined below using a P controller with precontrol (PV controller) as an example.

This procedure is the same whether or not the DSC setting is used.

Requirements

- The settings have been made for the controller type (PV controller), velocity precontrol, load gear, measuring gear, normalization of the speed setpoint interface, leadscrew pitch, and encoders.
- The fine interpolator should be set to cubic (continuous acceleration) or quadratic (continuous velocity) interpolation in order for the velocity precontrol to provide continuous values in the acceleration and deceleration phases.

- The axis data set whose control parameters are to be determined is active (if necessary, the relevant data set on the drive must also be selected).
- Monitoring and compensation functions are deactivated
  - Drift and friction compensation
  - Dynamic monitoring of following errors
  - Positioning monitoring

Checking the interface to the drive

Before performing the actual axis optimization, you should check whether the interface between the SIMOTION control and the drive has been configured correctly.

The following check is useful for this purpose:

The axis is traversed at constant velocity, and the control errors are recorded with the SIMOTION trace function in SCOUT. For this check, the servo gain factor Kv should be set to a low value (e.g. default value: Kv = 10/s).

-> Control error: **servoData.controllerDifference**

If the axis is operated with 100-percent velocity precontrol, the control error in the steady-state condition must be zero on average during the constant motion phase.

If velocity precontrol is not in effect, the control error in the steady-state condition is calculated as follows:

Control error = set velocity/Kv

If another control error results, then the configuration of the interface between the SIMOTION control and the drive is faulty.

A common cause for the faulty behavior is a discrepancy in the normalization speed configuration between the SIMOTION control and the drive. Check the following setting:

Scenario 1: **TypeOfAxis.setPointDriverInfo.DriveData.speedReference** = MAX_VALUE

In this case: drive normalization speed =
TypeOfAxis.setPointDriverInfo.DriveData.maxSpeed

Scenario 2: **TypeOfAxis.setPointDriverInfo.DriveData.speedReference** = NOMINAL_VALUE

In this case: drive normalization speed =
TypeOfAxis.setPointDriverInfo.DriveData.nominalSpeed

Note

Drives are typically linked to the SIMOTION control with PROFIdrive (via PROFIBUS/PROFINET). The parameters for setting the normalization speed for the SINAMICS, MASTERDRIVE MC, and SIMODRIVE 611U drive systems are listed below.

For SINAMICS, the normalization speed can be accepted by the drive in the axis wizard; as of V4.2, it is adapted during runtime. See Coupling of digital drives (Page 2895).

Table 4-355 Parameters for setting the normalization speed

Drive system	Parameter
SINAMICS	P2000
MASTERDRIVES MC	P353
SIMODRIVE 611U	P880

Determining the controller parameters

The following controller parameters of the active axis data set should be determined:

Kv P controller gain	**controllerStruct.PV_Controller.kv**
VTC Equivalent time of the lower-level drive (time constant of the balancing filter)	**dynamicData.velocityTimeConstant**
PTC Equivalent time of the position control loop	**dynamicData.positionTimeConstant**

Two main optimization goals can be differentiated:

1. High dynamic response for continuous movements and synchronous operation groups. The axis is allowed a certain amount of overswing.
2. High dynamic response for discontinuous movements, i.e. fast, time-optimized positioning movements without overswings.

As first step, the automatic speed optimization and subsequently the automatic position optimization (if DSC possible) are performed.

This optimization provides very good results even for the above-mentioned strategy.

The optimization of movements without overswing is described next.

Prerequisite:
The speed controller of the drive has already been set as (nearly) overshoot-free. In principle, this can be achieved by using a low-pass filter as the speed setpoint filter or through the use of a system model (reference model) for the I-component of the speed controller. For more information, refer to the relevant drive documentation.

Determining the servo gain Kv of the P-controller

Disable the velocity precontrol and balancing filter in the configuration data of the axis.

controllerStruct.PV_Controller.kpc = 0
dynamicData.velocityTimeConstant = 0

Position the axis cyclically with a high deceleration rate and trapezoidal velocity profile and trace the system variables for the set and actual positions of the axis with SIMOTION trace in SCOUT. Make sure that the axis reaches the constant velocity phase during this operation. You must also select the deceleration such that the current or torque in the drive is not limited.

Set position: **servoData.symmetricCommandPosition**
Actual position: **servoData.actualPosition**

Starting with Kv = 10, continue increasing the servo gain factor in five-percent increments as long as the actual value does not overshoot or undershoot when entering the target position.

Deduct 10 percent from this maximum Kv value and make this setting on the target device.

Setting the balancing filter

When velocity precontrol is used, the entry behavior during positioning and the control errors in the acceleration and deceleration phases can be influenced by the time constant of the balancing filter (VTC).

VTC: **dynamicData.velocityTimeConstant**

The velocity precontrol must be activated to determine the time constant.

controllerStruct.PV_Controller.kpc = 100
controllerStruct.PV_Controller.preCon = YES

Position the axis cyclically with a high deceleration rate and record the control error with SIMOTION trace in SCOUT. Make sure that the axis reaches the constant velocity phase during this operation. You must also select the deceleration such that the current or torque in the drive is not limited.

Control error: **servoData.controllerDifference**

Continue increasing the time constant of the balancing filter (VTC) incrementally until the axis enters the target position without overshooting or undershooting.

If you have to set a very large time constant to avoid overshooting/undershooting, and if the axis "creeps" to its target position, you may be able to achieve a smoother setpoint curve by using the SMOOTH velocity profile. In this case, you must limit the jerk in the motion command for entry in the target position and then repeat the balancing filter optimization.

If this still does not yield the desired result, you should check the drive setting again. It could be that the drive setting has not yet been optimized to be overshoot-free.

You may want to allow a small overshoot/undershoot upon entry in the target position to enable a faster settling behavior.

For axes with different balancing filter settings in a synchronous operation group, a dynamic response adaptation is desirable, see Dynamic response adaptation (Page 2968).

Setting the equivalent time of the position controller

To apply the preassigned axis stop ramp (in the event of an error, for example), the system requires the equivalent time constant of the position controller (PTC).

PTC: **dynamicData.positionTimeConstant**

In this example, this time constant is derived from the time constant of the balancing filter (VTC), due to the use of velocity precontrol (at 100%). The following relation applies:

PTC = VTC

If the velocity precontrol is not activated, the following is true for PTC:

PTC = 1/Kv

Storing the control parameters

If the control parameters have been entered in the corresponding configuration data in the target device, they must now be stored as non-volatile data so that they will still be available the next time the control is powered up. The following online functions are available for this purpose under the target device in SCOUT.

Table 4-356 Online storage functions

Function	Comments
Copy ACTUAL to RAM	Copies the configuration data to the RAM memory on the target device
Copy from RAM to ROM	Non-volatile storage of configuration data on memory card

In order to transfer modified configuration data to the configuration, you must select the **Load configuration data to PG** function under the target device in SCOUT and then save the project.

After you have optimized the position controller, you must reset the compensations and monitoring functions listed under "Requirements", if applicable.

4.8.3.13 Command variable calculation

Velocity profiles

Specifications of velocity changes for axes (approaching, stopping, other velocity changes) are applied via **velocity profiles**.

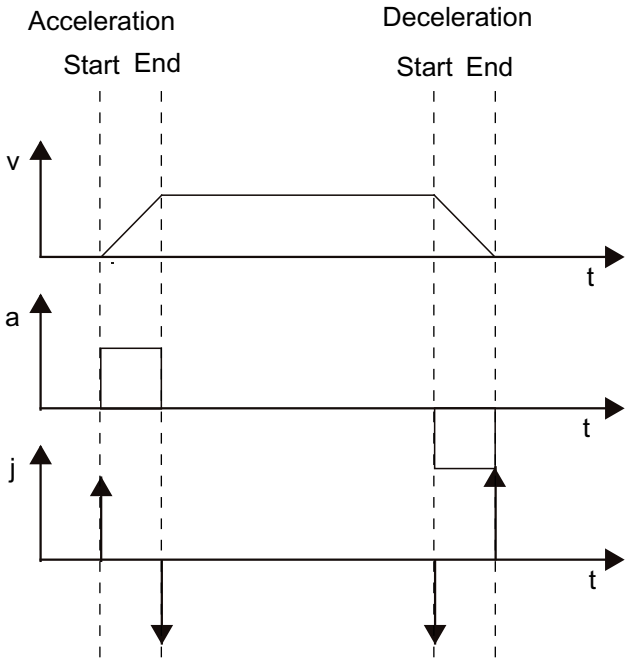
The velocity profile defines the behavior of the axis during startup and when braking, and for velocity changes.

The following velocity profiles are available:

- **Trapezoidal** velocity profile (**TRAPEZOIDAL**)
  The trapezoidal profile is set for constant acceleration and deceleration of motion in a positive and negative direction.
- **Smooth** velocity profile (**SMOOTH**)
  This profile is used for a continuous acceleration and deceleration. The jerk can be set.

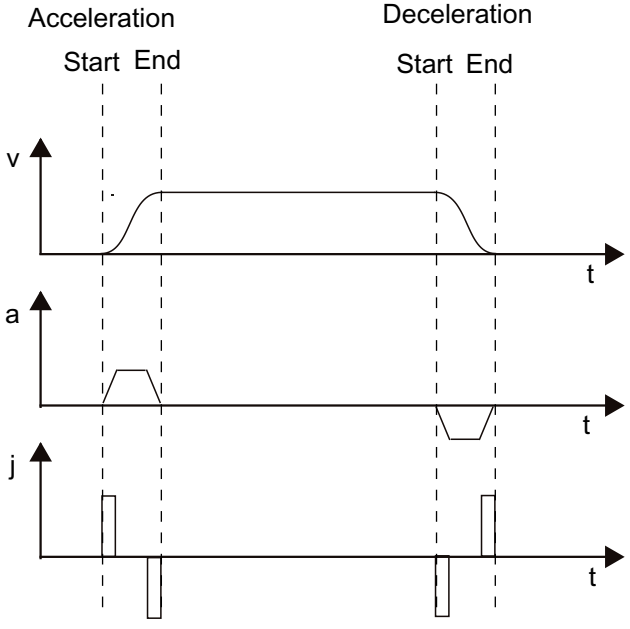
The parameters for profile, velocity, acceleration, and jerk can be assigned in the command, or the **userDefault** value settings are used.

The parameter for the velocity profile is specified via the motion command (velocityProfile) or stored as a default value in the userdefaultdynamics.profile system variable.



v - velocity
a - acceleration
j - jerk
t - time

Figure 4-859 Trapezoidal velocity profile



v - velocity
a - acceleration
j - jerk
t - time

Figure 4-860 Smooth velocity profile

Continuous acceleration curve for smoothing

Prior to SIMOTION V4.4., the following applies to single axis motions and path motions when a continuous acceleration curve is specified: When blending to the motion of the new command on transition, acceleration is zeroed, i.e. the deceleration is canceled via the jerk.

As of SIMOTION V4.4, it is possible to set that the acceleration is not zeroed. See also Positioning with blending (Page 3003).

Continuous acceleration curve for reversing

For single axis movements, for path movements, for the synchronization of synchronization movements and for the specification of a continuous acceleration curve, for any required reversing in the reversal point of the movement direction, the acceleration is made to zero. This means the acceleration/deceleration will be removed using the jerk and re-established after the reversal using the jerk.

This does not apply for motion specification via user-defined profiles or for the procedure based on motion vectors (MotionIn interface).

Setting the acceleration to zero when stopping or substituting a motion

When stopping with the **_stop()** and **_stopEmergency()** commands or substituting a motion with the **_pos()** and **_move()** commands, an existing acceleration or deceleration can be set immediately to zero. The existing acceleration or deceleration is not first reduced via the jerk.

The abortAcceleration function parameter is available for the stop commands as of V4.2.1 and the substitution commands as of V4.4.

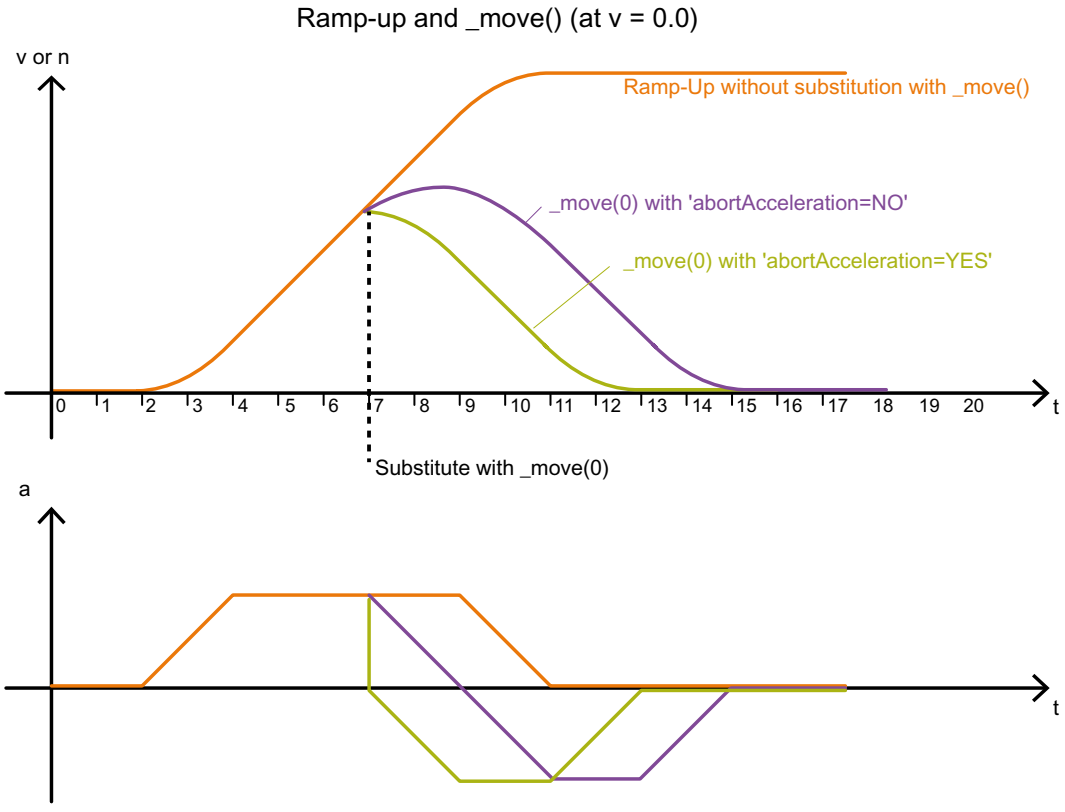


Figure 4-861 Velocity profile on ramp-up with and without abortAcceleration

Defining accelerations and decelerations

The axis acceleration and deceleration parameters can be set to take effect in relation to the direction or state using the **decodingConfig.directionDynamic** configuration data element.

Parameters for direction-related setting

- **positiveAccel:** Acceleration in the positive direction of motion and deceleration in the negative direction of motion
- **negativeAccel:** Acceleration in the negative direction of motion and deceleration in the positive direction of motion

The option to assign parameters for a direction-dependent dynamic response is relevant, for example, for applications such as vertical axes.

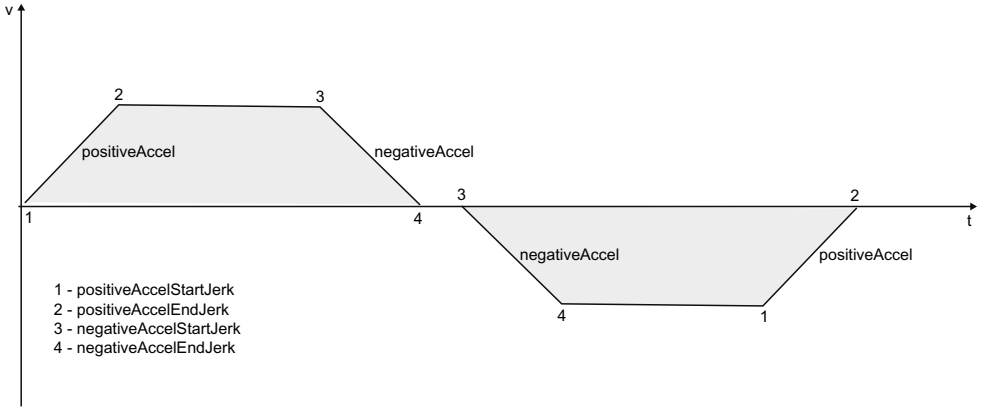


Figure 4-862 Operative points for acceleration and jerk parameters

Parameters for state-dependent setting

- **positiveAccel:** Acceleration of axis motion, irrespective of the direction of motion
- **negativeAccel:** Deceleration of axis motion, irrespective of the direction of motion

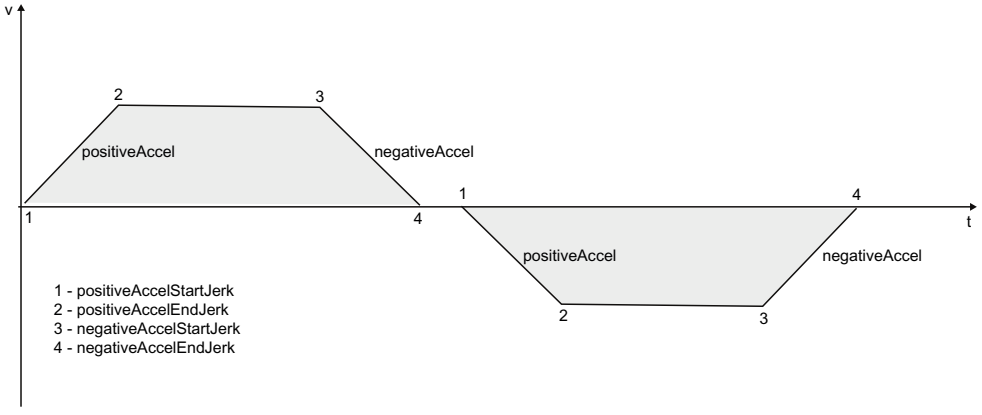


Figure 4-863 Operative points for acceleration and jerk parameters

Values set by the command or set by means of a system variable

- velocity
- positiveAccel
- negativeAccel
- positiveAccelStartJerk
- positiveAccelEndJerk
- negativeAccelStartJerk
- negativeAccelEndJerk

The dynamic response parameters are set with parameters in the motion command or stored as default values in the system variables of the userdefaultdynamics structure.

Override

Factors can be superimposed online on the current traversing velocity or acceleration/ deceleration. The velocity override is applied to the velocity, and the acceleration override is applied to the acceleration and deceleration.

The override values can be set and read via **system variables**.

The override for the **velocity** can be set from **0** to **200%**.

The override for the **acceleration/deceleration** can be set from **1%** to **1000%**.

System variables

- override.velocity
- override.acceleration

Default settings for dynamic response parameters

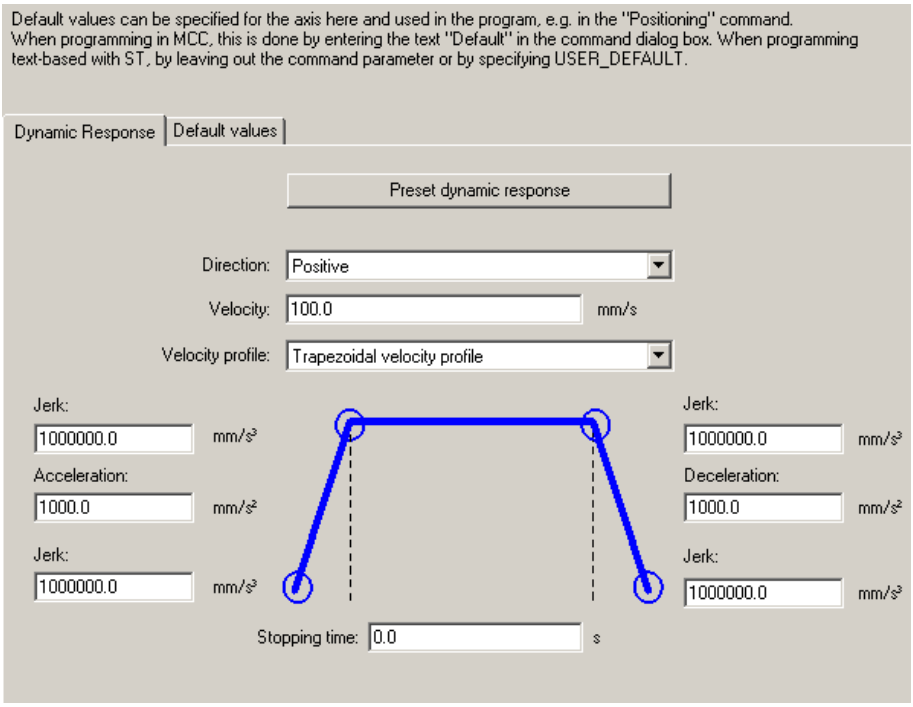


Figure 4-864 Default settings for dynamic response parameters

Stop time

This time becomes active if the default value is used for the time with **_stopEmergency()** and the **stopDriveMode=**STOP_IN_DEFINED_TIME setting.

Dynamic response default setting

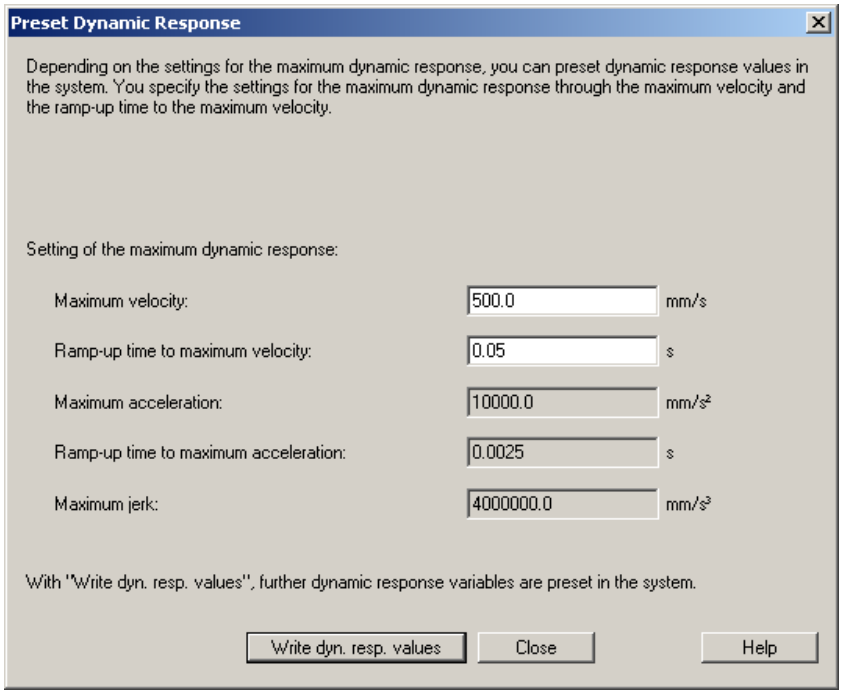


Figure 4-865 Dynamic response default setting

For the selection of the screen form, the **maximum velocity**, **maximum acceleration** and **maximum jerk** fields will be displayed using the setting in the associated configuration date. This provides the **startup time to the maximum velocity** and **startup time to the maximum acceleration**.

If the **maximum velocity** or the **startup time to the maximum velocity** is changed, the dependent values will be recalculated and displayed. The configuration data will be written directly and remain at the changed values also for **close** without **write dynamic values**.

The **startup time to the maximum velocity** will be used as startup time assuming the constant maximum acceleration over this time, i.e. without considering the time for building and removing the acceleration because of the maximum jerk.

Write dyn. resp. values writes corresponding values to the configuration data and adapts system variables, see Dynamic limiting functions (Page 2999).

See also

Velocity profiles (Page 2992)

Defining accelerations and decelerations (Page 2995)

Defaults (Page 2922)

Dynamic limiting functions

Maximum values

The properties of the drive and the mechanical system produce the maximum values for velocity, acceleration and jerk. The values are set in the **maxVelocity**, **maxAcceleration**, and **maxJerk** variables.

Technological maximum values

For programming, additional dynamic limit values are available in the **plusLimitsOfDynamics** and **minusLimitsOfDynamics** system variables. These can be read and written in the program.

The technological maximum values are state-dependent, i.e. are not a direction-dependent setting:

- **plusLimitsOfDynamics.velocity**
  Technological limit value for the velocity of the axis independent of the direction of motion
- **plusLimitsOfDynamics.positiveAccel**
  Technological limit value for the acceleration of the axis independent of the movement direction
- **plusLimitsOfDynamics.negativeAccel**
  Technological limit value for the deceleration of the axis independent of the movement direction
- **plusLimitsOfDynamics.positiveAccelJerk**
  Technological limit value for the jerk for establishing the acceleration and removing the deceleration independent of the movement direction
- **plusLimitsOfDynamics.negativeAccelJerk**
  Technological limit value for the jerk for removing the acceleration and establishing the deceleration independent of the movement direction

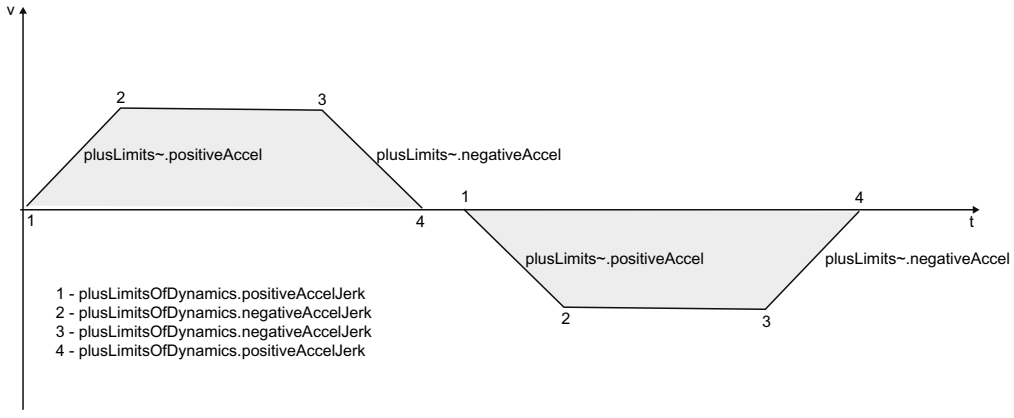


Figure 4-866 Technological maximum values for non-direction-dependent setting

Technological maximum values for direction-dependent setting:

- **plusLimitsOfDynamics.velocity**
  Technological limit value for the velocity of the axis independent of the direction of motion
- **minusLimitsOfDynamics.velocity**
  Technological limit value for the velocity of the axis independent of the direction of motion

- **plusLimitsOfDynamics.positiveAccel**
  Technological limit value for the acceleration of the axis in the positive movement direction
- **plusLimitsOfDynamics.negativeAccel**
  Technological limit value for the deceleration of the axis in the positive movement direction
- **minusLimitsOfDynamics.positiveAccel**
  Technological limit value for the acceleration of the axis in the negative movement direction
- **minusLimitsOfDynamics.negativeAccel**
  Technological limit value for the deceleration of the axis in the negative movement direction
- **plusLimitsOfDynamics.positiveAccelJerk**
  Technological limit for the jerk for establishing the acceleration and removing the deceleration in the positive movement direction
- **plusLimitsOfDynamics.negativeAccelJerk**
  Technological limit for the jerk for removing the acceleration and establishing the deceleration in the positive movement direction
- **minusLimitsOfDynamics.positiveAccelJerk**
  Technological limit for the jerk for establishing the acceleration and removing the deceleration in the negative movement direction
- **minusLimitsOfDynamics.negativeAccelJerk**
  Technological limit for the jerk for removing the acceleration and establishing the deceleration in the negative movement direction

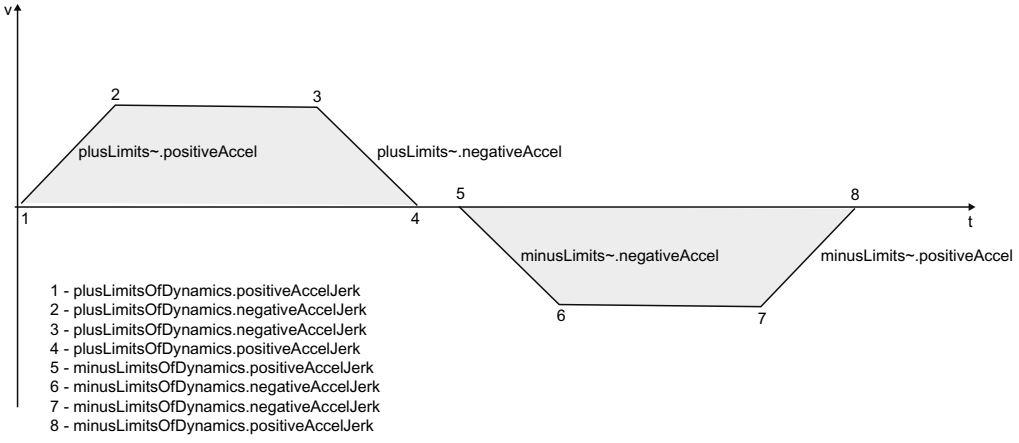


Figure 4-867 Technological maximum values for direction-dependent setting

The minimum from the maximum value of the axis and the set technological maximum value for the associated dynamic response quantity are used.

For system created movements with maximum dynamic response values, e.g. travel to the software limit switch, the minimum from the maximum value and the technological maximum value is used for the corresponding dynamic response quantity.

Active dynamic limitation

- Enter the maximum velocity in **maxVelocity**. This must always be less than or equal to the velocity that is physically possible. The greater the difference between the two, the larger the available control margin.
  Please note that the gear is data-set-dependent. Maximum velocity is calculated per data set. The maximum across all data sets is taken to be the maximum velocity.
- The effective limit is always the minimum of the maximum value and the technological maximum value.
- The **maxJerk** limiting is only monitored for motions in jerk-controlled mode or motions with continuous acceleration.

Dynamic limitation with synchronous operation relationships

If the axis provides a master value for a synchronous operation relationship, the maximum velocity is also limited to less than half the modulo range/IPO cycle clock (maximum possible master value velocity of the axis).

It is the minimum value from the following that applies when calculating the setpoint value for the following axis:

- Maximum velocity (**maxVelocity**) of the following axis
- Maximum technological velocity (**plusLimitsOfDynamics**/**minusLimitsOfDynamics**)
- Maximum master-value velocity of the axis

If a velocity faster than this maximum velocity is specified, the technological alarm 40002 "velocity will be limited" will be generated and the set velocity adapted appropriately.

Dynamic limitation with active ramp calculation in the drive

Please note that when linking with digital drives and with a ramp calculation for the speed active in the drive, additional dynamic limitation may occur, which is not taken account of separately in the motion control and motion monitoring functions performed in the control.

The operating mode is displayed in the drive's PROFIdrive parameter R0930 (PROFIdrive operating mode):

1. Speed-controlled operation with ramp-function generator
2. Position-controlled operation
3. Speed-controlled operation without ramp-function generator

SINAMICS drives:

- Drive object type vector
  Default setting *1. Speed-controlled operation with ramp-function generator*
  The motion dynamic response is, therefore, limited in the drive, if applicable.
- Drive object type servo
  Default setting *3. Speed-controlled operation without ramp-function generator*
  The motion dynamic response is not limited in the drive.

When using vector drives, therefore, you must check that the ramp-up/ramp-down time parameterized for the drive matches the configured axis acceleration/deceleration.

Stopping with preassigned braking ramp

The deceleration set in the **emergencyRampGenerator.maxDeceleration** configuration data element is used to stop an axis with a preconfigured braking ramp.

The current velocity is brought to zero at the assigned deceleration rate.

With position-controlled traversing, the position controller continues to be active during the stopping procedure.

For position-controlled, and thus position-related, traversing, the application point of the actual position value is extrapolated with the current velocity and the equivalent time of the position control loop set in the **dynamicData.positionTimeConstant** configuration data element.

Therefore, the application point can only be calculated correctly if the equivalent time of the position control loop is set correctly.

The preassigned braking ramp is effective in the following cases:

- During stopping with the **_stopEmergency()** command and the **STOP_WITH_COMMAND_VALUE_ZERO** setting
- During the **FEEDBACK_EMERGENCY_STOP** alarm response
- When program simulation is enabled during axis motion using the **_enableAxisSimulation ()** command

Traversing the axis via velocity specifications

The axis can be traversed via velocity specifications. A position axis can be traversed either via a velocity setpoint specification or in position-controlled mode.

The following is specified for the axis motion:

- Direction
- Dynamic response parameters
- Optionally, the duration of the constant motion phase

Parameters

- Direction
  The direction of motion is defined using either the sign of the velocity specification or a specific parameter.
- Velocity

- Duration of constant motion phase
  The duration of the constant motion phase is an optional parameter. If the constant motion phase is not specified, the **_move()** command triggers a continuous motion that must be overridden by a **_stop()** command or another motion command.
- Dynamic response parameters

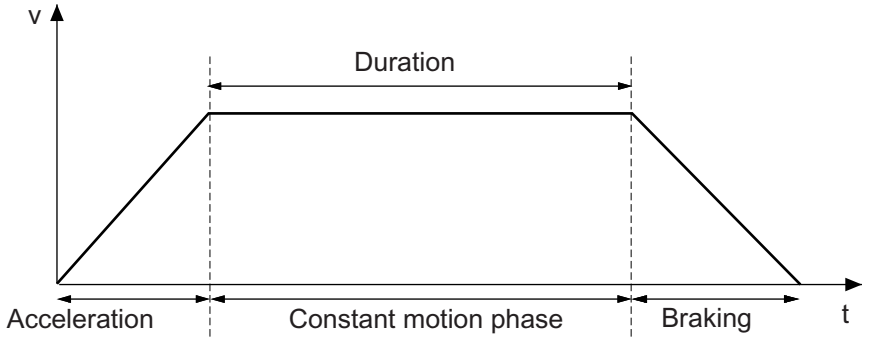


Figure 4-868 Phases for moving with _move()

An acceleration phase or a deceleration phase can be ended with jerk control using the **velocityType=RESULTING** parameter. The resulting velocity is then retained.

See also

Velocity profiles (Page 2992)

Defining accelerations and decelerations (Page 2995)

Positioning

The axis is moved to a target position via a parameterizable velocity profile. The entry in the target position is monitored.

See also

Positioning and standstill monitoring (Page 2941)

Positioning with blending

Blending is a special way of linking the positioning motion programmed in the command to a previous positioning motion. Unlike overriding, the previous motion command is completed up to the target position, where the transition occurs. Blending takes place between two positioning commands. The set velocity specified in the commands for each motion is never violated.

The previous positioning motion is performed at the velocity specified in the command up to the target position. **Exception:** If the velocity of the new positioning motion is less than the set velocity of the previous motion and both motions have the same sign, then the velocity of the previous positioning motion is reduced to the velocity of the new motion by the time the target position is reached.

In versions earlier than SIMOTION V4.4, the dynamic response is planned via two traversing blocks.

As of SIMOTION V4.4, the dynamic response is planned via three traversing blocks (current, next and next but one) for a newly created axis. It is therefore possible to plan a higher velocity if necessary for axis motion in the same direction. The setting is saved in the **DecodingConfig. commandsForAxisDynamics** configuration data element.

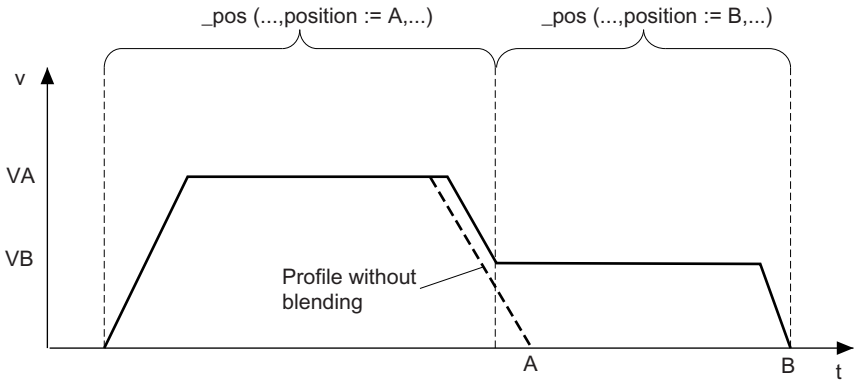


Figure 4-869 Blending when velocity of the subsequent motion is lower

If the direction is reversed, the previous positioning motion is slowed down at the target point, and transition to the new motion takes place immediately.

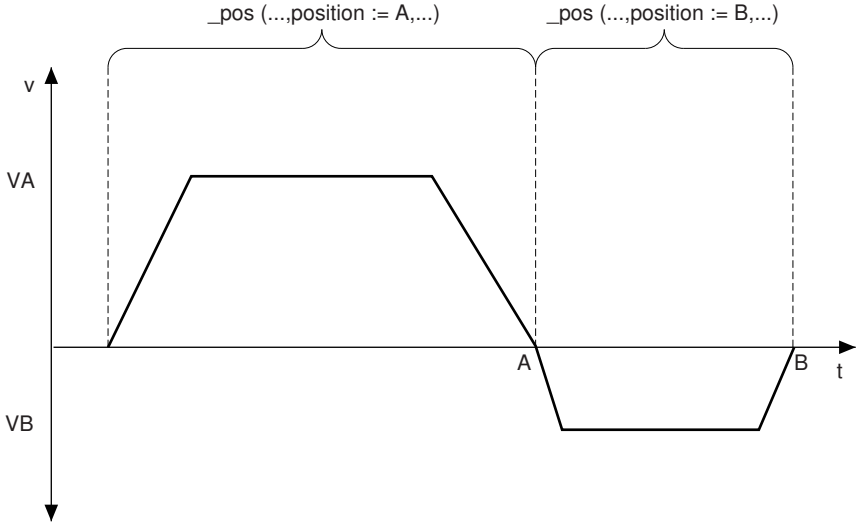


Figure 4-870 Blending when the subsequent motion reverses direction

If the absolute value of the velocity of the new command is greater than the velocity of the current motion, the velocity is increased after the transition to the new command, that is, after the previous target position is reached.

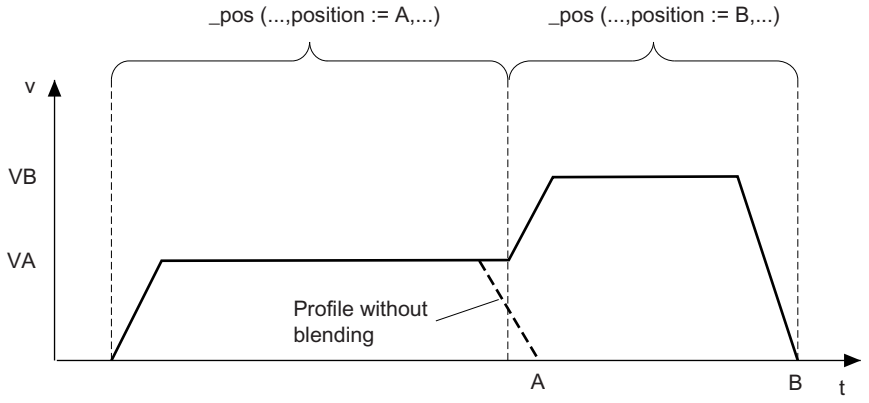


Figure 4-871 Blending when velocity of the subsequent motion is higher

Active blending requires **mergeMode**= NEXT_MOTION or SEQUENTIAL in the command where blending is to take place, plus timely decoding of the command. If the command in which blending is to be carried out is not known to the interpolator at the deceleration starting point of the current motion, the braking ramp is traveled.

The braking distance is always adhered to, even if the path length of the subsequent motion command is shorter than the required braking distance.

Acceleration at the command transitions

Versions before SIMOTION V4.4 are limited by the fact that acceleration is zeroed on blending with programmed, constant acceleration curve (SMOOTH) at the transition to the new command.

As of SIMOTION V4.4, it is possible to use the **DecodingConfig.blendingAcceleration** configuration data element to set that the acceleration at the transition to the new motion command is not zeroed if not necessary for the overall motion. In a newly created axis, WITHOUT_REDUCTION is the default.
When reversing, the acceleration is always zeroed regardless of this setting. Reversing here means that the direction of motion of the axis changes at the blending point.

Superimposed positioning

Superimposed positioning is performed in the superimposed coordinate system of the axis. The target position can be specified as an absolute or relative position in this coordinate system. (For feedback of the target position, see **Superimposed motion**)

The superimposed motion has its own programmable dynamic response parameters. The **userDefault** settings of the dynamic response parameters are identical to the settings of the main motion. Blending conditions for the superimposed motion are ignored.

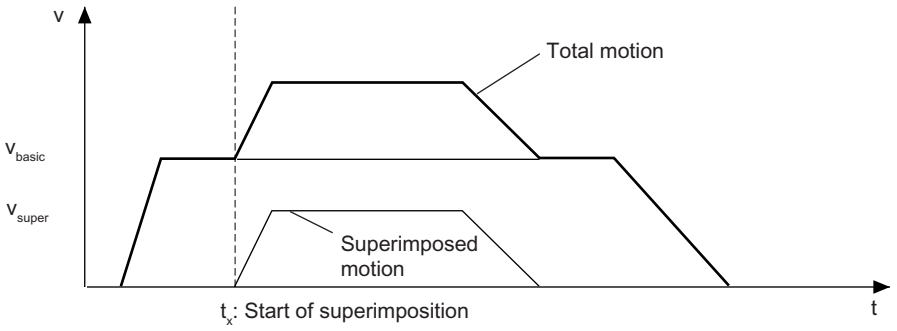


Figure 4-872 Superimposed positioning

See also

Superimposed motion (Page 3008)

Traversing with specific motion profiles

In addition to traversing/positioning via system functions, the axis can also be traversed via user-defined profiles/specific profiles.

See also

Overview of traversing with user-defined motion and force/pressure profiles (Page 3040)

Traversing according to motion vectors (V3.2 and higher)

The axis can be traversed directly in accordance with the specification on the MotionIn interface.

The MotionIn values can be directly picked up by another technology object, e.g., axis, external encoder, adding object, formula object, fixed gear. Alternatively, the MotionIn values can be specified via the default values from the user program.

The **_runPositionBasedMotionIn()** command activates the MotionIn interface with position-reference and **_runVelocityBasedMotionIn()** activates the MotionIn interface with velocity-reference. The MotionIn interface is replaced by motion commands, such as **_move()**, **_pos()**, **_stop()**, and **_stopEmergency()** or it can have an overriding effect itself.

Options for linking drive and position-controlled axes (position axes, following axes) to the motion vector:

- Linking position-controlled axis to motion vector (based on: position)
- Linking position-controlled axis to motion vector (based on: velocity) in position-controlled or velocity-controlled mode
  In this way, a position-controlled axis (position axis) is linked to a drive axis.
- Linking drive axis to motion vector (based on: velocity)
  This links a drive axis to a drive axis.

The MotionIn values can also be directly specified cyclically via the default variables.

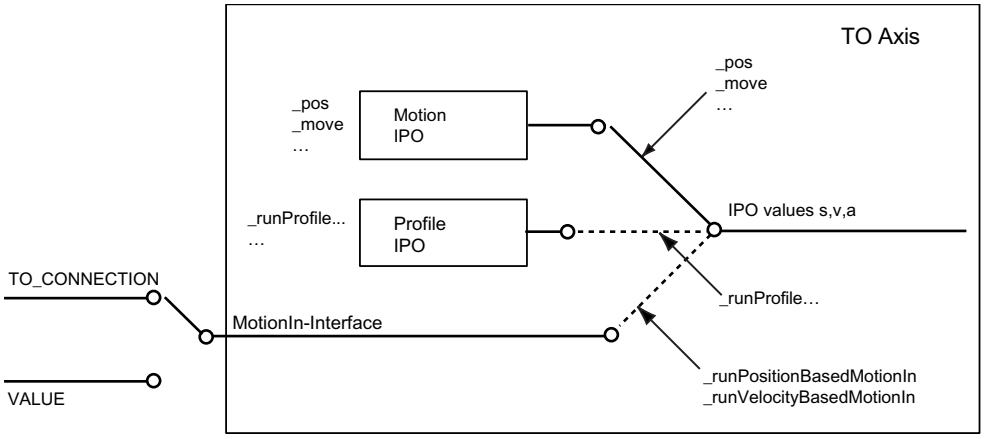


Figure 4-873 How the MotionIn interface works

Property

- Cyclical inclusion of MotionIn interface values
- Superimposition and overriding motion are possible
- Dynamic response parameters can be predefined in the commands for moving to the MotionIn values and for stopping
- The dynamic response can be limited via configuration data
- The MotionIn interface on the axis is multipoint-capable
- The reference TO is specified in the activation command
- It is then possible to switch to the reference TO

Note

Further information is available under Interconnection of technology objects in the Motion Control Basic Functions Function Manual.

Jerk limitation for local stop response (V3.2 or higher)

Local stop responses (stop responses triggered by alarm responses) may optionally be applied with or without rounding and jerk limiting.

The **decodingConfig.StopWithJerk** configuration data element can be used to specify whether the maximum jerk limitation of the Axis technology object is taken into account.

The jerk is only taken into account for IPO stops.

The jerk is determined from the minimum of the **plusLimitsOfDynamics**/ **minusLimitsOfDynamics** system variables and the **maxJerk** configuration data element.

4.8.3.14 Superimposed motion

Superimposed motion is defined with mergeMode=SUPERIMPOSED_MOTION_MERGE.

Superimposed motions are independent motions.

Superimposed motions cancel each other.

Superimposed motions can be stopped and continued separately.

Superimposed motions on a position axis are executed in the superimposed axis coordinate.

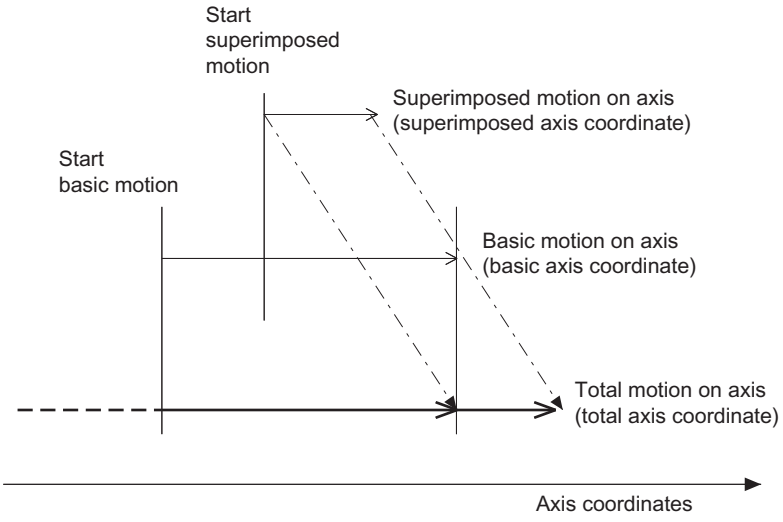


Figure 4-874 Functional diagram for motion transition SUPERIMPOSED_MOTION_MERGE

Commands that allow superimposed motions on the axis include:

_move()
_pos()
_runTimeLockedVelocityProfile()
_runTimeLockedPositionProfile()

Note

Only one superimposed motion can be active on the axis at a given time, e.g. superimposed positioning motion or superimposed synchronous operation.

Note

The dynamic limits always relate to the overall dynamic response. This may result in the superimposed motion having a dynamic response that is not as good as that defined in the specifications.

The superimposed motion is carried out in the superimposed axis coordinate as relative or absolute motion, depending on how it was programmed. Likewise, the basic motion of the axis is executed in the basic axis coordinate as an absolute motion or a relative motion, depending on the programming.

The timing of the feedback of the superimposed axis coordinate to the basic axis coordinate is set in the **decodingConfig.transferSuperimposedPosition** configuration data element; this

setting also specifies when superimposed motions are applied to the basic motion and, thus, when they are overridden. When the criterion becomes effective, the basic motion accepts the superimposed motion component.

Possible settings:

- Only on axis reset with _resetAxis()
- On axis reset and on each overriding command where mergeMode= IMMEDIATELY
- On axis reset, on each overriding command where mergeMode=IMMEDIATELY, and on each axis standstill motionStateData.motionStat=STANDSTILL

In addition, the superimposed axis coordinate/motion is fed back to the basic axis coordinate in the following cases:

- The axis switches into or out of follow-up mode.
- Switchover occurs between speed-controlled, position-controlled, and force-/pressure-controlled operation.
- For force-/pressure-controlled operation

Absolute and/or relative traversing is possible in the basic axis coordinate and in the superimposed coordinate. The total motion of the axis results from adding the position values, velocity values, and acceleration values of the basic motion and the superimposed motion together.

Notes on working with the superimposed axis coordinate

During active homing, a superimposed axis coordinate is always reset.

An active superimposed motion is aborted during active homing.

Display of current axis coordinates

The values/dynamic response vector of the basic axis coordinate and the superimposed coordinate are displayed on the axis.

These coordinates can be read via the following system variables:

- Total axis coordinates:
  positioningState.commandPosition: Set position (total)
  motionStateData.commandVelocity: Set velocity (total)
  motionStateData.commandAcceleration: Set acceleration (total)
- In basicMotion, the values of the basic axis coordinate are displayed on the axis.
  basicMotion.position
  basicMotion.velocity
  basicMotion.acceleration
- In superimposedMotion, the values of the superimposed axis coordinate are displayed on the axis.
  superimposedMotion.position
  superimposedMotion.velocity
  superimposedMotion.acceleration

4.8.3.15 Torque limiting via torque reduction

Overview of torque limiting via torque reduction

On the electric axis, torque limiting is available via a torque reduction. The limit value is specified in the command.

This function can be used with a drive axis, a position axis and a following axis. The accuracy depends on the drive used.

To be able to use the function, the PROFIdrive telegram type 10x (apart from telegram 101) must be set. It also requires the use of a drive that supports the function. As of V4.2, 10x telegrams are used with automatic telegram definition.

When the function is called, the desired torque (for rotary and linear axes) or the desired force (for linear axes only) is specified in the corresponding unit or as percentage relative to a reference value (**userDefaultTorqueLimiting.torqueLimit**). 0% means no torque on the drive and 100% means full torque on the drive.

The torque limiting is transferred to the drive as a torque reduction in the PROFIdrive telegram. If, for example, 80% is specified for the torque limiting, SIMOTION calculates the value 20(%) for the torque reduction for the drive and transfers this value to the drive via the PROFIBUS interface.

The limitation acts as an absolute value and thus has the same effect on both positive and negative torque.

If torque limiting is active, the following error monitoring and positioning monitoring functions are deactivated.

Active motion commands and synchronous operation relationships continue to be executed.

The limiting functions can be activated before a motion or simultaneously with a motion and can be switched by reissuing the command.

The position-related monitoring can remain active even with active torque limiting. For this purpose, there is the **TypeOfAxis.ServoMonitoring.motionMonitoringWhenTorqueLimiting** configuration data as of V5.1. When activated, this ensures that the position monitoring, positioning monitoring and/or following error monitoring that was active before the torque limiting remains active during the limiting.

As a result of torque limiting, for example, a greater difference between actual and target values can build up on the position-controlled axis, which can cause the axis to accelerate (in order to reduce the difference) even if the velocity calculated by the interpolator is now lower.

Please note that if an axis is ready as a result of active torque limits/torque reduction, the control difference to the position reference value is initially canceled and then the **_stop()** command is terminated, despite a position-controlled stop motion. Alternatively, you can enter a speed-controlled **_stop()** command.

If, for example, torque limiting is not required during the acceleration phase, the function must not be activated until after the acceleration phase; otherwise, the acceleration must be reduced.

Value specifications and transferring from the drive

The maximum values **SetpointDriverInfo.DriveData.maxTorque** and **SetpointDriverInfo.LinearMotorDriveData.MaxForce** are the reference values for the transmission of the obtained torque reduction to the drive and must be entered in the drive and in SIMOTION in accordance with the motor values.
With SINAMICS, the parameters are present in P1520, P1521, and P2003:
~.DriveData.maxTorque = P1520 = |P1521|
If the technology data block (Page 3018) is interconnected, the following applies: **~.DriveData.maxTorque** = P2003

As of V4.2

The above still applies for the setting **SetpointDriverInfo.DriveData.TorqueReference** = MAX_VALUE or **SetpointDriverInfo.LinearMotorDriveData.ForceReference** = MAX_VALUE.

For SINAMICS and round-frame motors, and for the setting **SetpointDriverInfo.DriveData.TorqueReference** = NOMINAL_VALUE and **SetpointDriverInfo.DriveData.SpeedReference** = NOMINAL_VALUE, the following applies:

- **~.DriveData.nominalSpeed** = P2000
- **~.DriveData.MaxSpeed** = P1082
- **~.DriveData.nominalTorque** = P2003
- **~.DriveData.MaxTorque** = P1520

For SINAMICS and linear motors, and for the setting **SetpointDriverInfo.LinearMotorDriveData.SpeedReference** = NOMINAL_VALUE and **SetpointDriverInfo.LinearMotorDriveData.ForceReference** = NOMINAL_VALUE, the following applies:

- **~.LinearMotorDriveData.nominalSpeed** = P2000
- **~.LinearMotorDriveData.MaxSpeed** = P1082
- **~.LinearMotorDriveData.nominalForce** = P2003
- **~.LinearMotorDriveData.MaxForce** = P1520.

By default, these settings are transferred by the drive during the assignment of the axis to a SINAMICS drive by the system as of V4.2.

By default, the settings for torque granularity (see below as well) are transferred by the system from the drive:

~.driveData.torqueReductionGranularity = P1544

~.LinearMotorDriveData.ForceReductionGranularity = P1544

Enabling/disabling

The limiting function is disabled by default and must be specifically activated.

The torque limiting is activated using the **_enableTorqueLimiting()** command and has the following effect:

- The reduced maximum torque limit is in effect immediately
- The following error and positioning monitoring are disabled

With the function parameter **torqueLimitType**=USER_DEFAULT (system default), the default torque limit value set in the **userDefaultTorqueLimiting.torqueLimit** system variable is used. This default can be modified by entering a value in the **torqueLimit** parameter. The value of the parameter is interpreted as a percentage.

With the setting **torqueLimitType**=DIRECT, the value is specified as torque or force based on the **torqueLimitUnit** parameter. Depending on the **torqueLimitUnit** parameter, this value refers to the load or the motor side. For more information, see Conversion of torque/force (Page 3014).

The torque reduction is not disabled with **_disableAxis()**; it must be explicitly disabled via a command.

It is disabled/canceled

- With **_disableTorqueLimiting()**
- In the case of **_resetAxis()**, **_restartAxis()**, etc.
- In the case of a transition to STOP

Suppression of the "Speed alignment" fault message of the drive

If torque reduction is enabled and the value is not equal to 0, the fault message of the drive (discrepancy between actual speed and the speed setpoint) is suppressed.

If the fault message is also to be suppressed when the torque reduction is disabled or when the torque reduction is enabled and the value is equal to 0, bit 8 in STW2 can be set for message suppression using the **_setAxisStw()** function.

The setting via the **_setAxisStw()** function has a storing effect, i.e. it is kept during and after the period that a torque reduction is activated.

Special features

- The commands for torque limiting (**_enableTorqueLimiting()**) and travel to fixed endstop (**_enableMovingToEndStop()**) cannot be active simultaneously.
  The transition from _enableTorqueLimiting() to **_enableMovingToEndStop()** is permitted (and has an override effect).
  The transition from **_enableMovingToEndStop()** to **_enableTorqueLimiting()** is not permissible since the setpoint must be clamped when retaining the torque in the fixed endstop.
- The **_stopEmergency()** command:
  If a motion is ended with position control (movingMode = POSITION_CONTROLLED) with the **_stopEmergency()** command, the stop ramp is generated from the current set position in stop modes STOP_IN_DEFINED_TIME, STOP_WITH_DYNAMIC_PARAMETER, and STOP_WITH_MAXIMAL_DECELERATION.
  If a following error has built up at this time due to active torque limiting, it is removed, and as a result, the axis does not come to an immediate standstill.
  For this reason, if you are using torque limiting, you should stop the motion with speed control (movingMode = SPEED_CONTROLLED). Alternatively, you can select the WHEN_COMMAND_VALUE_ZERO stop mode for stopping the motion with position control.
  An active torque reduction (including when moving to a fixed end stop) is retained.
  Exception:
  The **_stopEmergency()** command with stopDriveMode = STOP_WITH_COMMAND_VALUE_ZERO disables the torque reduction, and the travel to fixed endstop command is aborted.
- When torque limiting is active, the following error monitoring is disabled.
  As a result of torque limiting, for example, a greater difference between actual and desired values can build up on the position-controlled axis, which can cause the axis to accelerate (to reduce the difference) even if the velocity calculated by the interpolator is now lower.
  If, for example, torque limiting is not required during the acceleration phase, the function must not be activated until after the acceleration phase; otherwise, the acceleration must be reduced.
- The torque limits B+ / B- and the torque reduction must not be active simultaneously.

Status displays

Table 4-357 Status display variables

Variable	Meaning
TorqueLimitingCommand.State	
	Indicates the torque limiting state. ACTIVE: Torque limiting activated INACTIVE: Torque limiting deactivated
TorqueLimitingCommand.torqueLimitingState	
	Here, it is indicated if the drive is running at the torque limit. **Note:** This state is derived from the PROFIdrive profile status word 'MeldW' (PZD 5), bit 1 (M < Mx) of the drive.

Assignment of the resolution in the drive

Table 4-358 System data

SIMOTION system data	Meaning
userDefaultTorqueLimiting.torqueLimit	
	This data element specifies the user default setting of the torque limit value for the torquelimit function parameter in the _enableTorqueLimiting() command. This value is accessed when the command contains the USERDEFAULT setting. System default: 10.0

Setting the resolution in SIMOTION

The resolution of the torque reduction to the drive can be set, as of V4.0,
in **driveData.torqueReductionGranularity** or
in **linearMotorDriveData.forceReductionGranularity**.
In terms of value transfer, also consult the above section *Value specifications and transferring from the drive*.

Possible settings:

- BASIC (default, 1% resolution)
  64H in PZD (PROFIdrive telegram) is a torque reduction of 100%.
  - SINAMICS
    Setting required: P1544 = 16384.0
  - SIMODRIVE 611U
    Standard setting: P0881 = 16384.0
- STANDARD (~0.006% resolution)
  4000h in PZD (PROFIdrive telegram) is a torque reduction of 100%.
  - SINAMICS
    Standard setting: P1544 = 100.0
  - SIMODRIVE 611U
    Setting required: P0881 = 100.0

Note

Be sure to set the values in the drive consistently.

See also

Enabling and disabling torque limiting (Page 3162)

Conversion of torque/force

Speed-controlled axes and rotary axes

When the **_enableTorqueLimiting()** function is programmed, there is always a torque specified in Nm, kNm or MNm.

If the TORQUE setting is specified when calling the function with the **torqueLimitUnit** function parameter, the specified torque refers to the motor. The gear ratio is not taken into account.

If the DEFAULT_UNIT setting is selected in the function parameter **torqueLimitUnit**, the torque refers to the load side and the gear ratio is taken into account. The following conversion formula applies:

$M_{Load} = M_{Motor}$ x (motor revolution/load revolution)

Linear axes with standard motor

In the **_enableTorqueLimiting()** function, the following can be optionally specified: a torque in Nm, kNm or MNm in relation to the motor, or a force in N, kN or MN in relation to the load side.

If the TORQUE setting is specified when calling the function with the **torqueLimitUnit** function parameter, the programmed value is interpreted as torque in relation to the motor. The gear ratio, leadscrew pitch, and efficiency of the leadscrew are not taken into account.

If the DEFAULT_UNIT setting is selected in the function parameter **torqueLimitUnit**, the programmed value is interpreted as force in relation to the load side. The gear ratio, leadscrew pitch, and efficiency of the leadscrew are taken into account in this setting. The following conversion formula applies, whereby **S** is the leadscrew pitch (adjustable in **leadscrew.pitchval**) and η is the efficiency of the leadscrew (adjustable in **leadscrew.efficiency**):

$F = M_{Motor}$ x 2 x π x ($\eta_{Leadscrew}$ / S) x (motor revolution/load revolution)

Linear axes with linear motor

There is always a force in N, kN or MN specified when programming.

4.8.3.16 Travel to fixed endstop

The **_enableMovingToEndStop( )** function activates the monitoring of **travel to fixed endstop** in parallel with an axis motion activated by a motion command and the retention of a clamping torque once the fixed endstop has been reached. The operation is also referred to as **clamping**.

This function requires that torque reduction be supported by the digital coupled drive.

Positioning with an immediate step enabling condition must take place first (position-controlled traversing).

During configuration, a setting can be made in **clampingMonitoring.recognitionMode** to define which of the following methods is to be used for detecting that the fixed stop has been reached:

- The following error limit is exceeded
- The torque limit is exceeded (evaluation of the **| M | < M**x message bit in the PROFIdrive telegram / status word (message word bit1 for SIEMENS telegram 101, 102, 103, 104, 105, 106, 125 and 126))
  As of V4.2, 10x telegrams are used with automatic telegram definition.

If the criterion is met, the fixed endstop status is regarded as having been reached, the **movingToEndStop.clampingState**= ACTIVE system variable is enabled, and the clamping tolerance is activated. Following error monitoring is switched off as soon as the command

becomes effective. The setpoint at the input of the position controller is maintained. Provided that the position of the fixed endstop does not change, the difference between the setpoint and actual value remains the same.

Note

The position-related monitoring can remain active with TypeOfAxis.ServoMonitoring.motionMonitoringWhenTorqueLimiting even with active torque limiting.

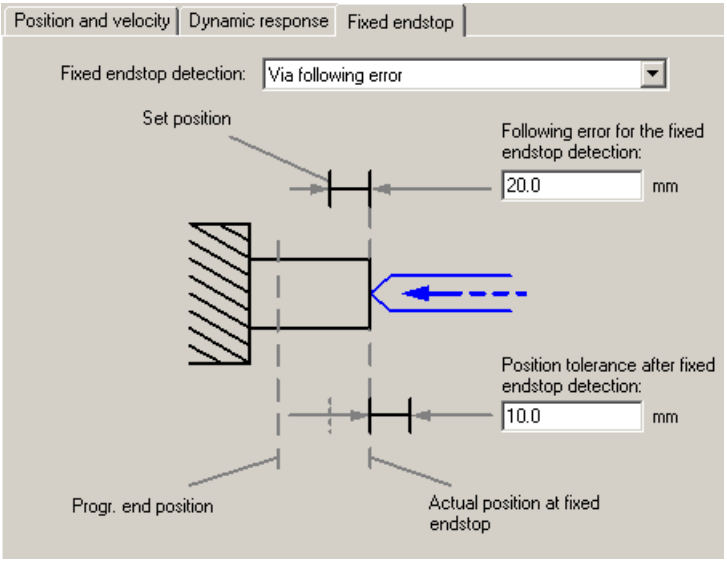


Figure 4-875 Function and parameters for travel to endstop with detection of fixed endstop by means of following error

The motion command is aborted with an error message; closed-loop control remains active. The setpoint at the input of the position controller is clamped. For new motion commands, the clamping set position is used as the start position. Motion commands in the direction of clamping are aborted. Motion commands in the opposite direction to the clamping direction are executed, thereby reducing the torque.

Travel to fixed stop is canceled when the clamping tolerance window is exited.

Issuing the **_enableMovingToEndStop()** command again can cause the torque to switch over, even if clamping is active.

The following commands cancel the function:

- **_resetAxis()**
- **_disableAxis()**
- **_disableMovingToEndStop()**
  - Up to and including V4.1 SP3: Cancel if outside the clamping window
  - As of V4.1 SP3 HF4: Cancel always
- **_cancelAxisCommand()** as of V4.1 SP3 HF4

Non-stepped torque transitions and torque retention over a defined time period can be implemented in the user program, as can specification of torque profiles.

Note

If the clamping torque is increased during clamping (by a new **_enableMovingToEndStop()** command), then this torque might not be achieved. Position control is active, but no more setpoints are generated for a motion in the direction of clamping.

The **_disableMovingToEndStop()** command deactivates the **Travel to fixed endstop** function. The **_disableMovingToEndStop()** command has no effect while clamping is active. The **_disableMovingToEndStop** command can be issued in parallel to position-controlled motion.

If a breakaway of the fixed endstop occurs, the position controller abruptly reduces this difference. This response is used for monitoring purposes. Clamping monitoring saves the current actual value that applies at the time of clamping. The clamping tolerance window then wraps around this value (see the figure above). This is why the clamping tolerance value should be lower than the setpoint/actual value difference at the point when clamping is detected. An error is only detected if the axis leaves the clamping tolerance window during abrupt reduction of the setpoint/actual value difference. In this case, a 50108 alarm is triggered.

If the clamping tolerance window is exited, **alarm 50108: Clamping monitoring error** is triggered and the **travel to fixed stop** is aborted.

In the command, the limiting force for the clamping value is set in [N] for linear axes, while for rotary axes, the limiting torque is set in [Nm].

If the **_enableMovingToEndStop()** command is active and the fixed stop has not been detected, the system behaves the same as with active torque limiting.

The **_enableMovingToEndStop()** and **_enableTorqueLimiting()** commands cannot be active simultaneously.

The transition from "torque limiting" (**_enableTorqueLimiting()**) to "travel to fixed stop" (**_enableMovingToEndStop()**) is permitted (and has an override effect), but the reverse is not permitted.

System variables

Table 4-359 System variables for travel to fixed stop

Variable	State	Meaning
MovingToEndStopCommand.state	ACTIVE / INACTIVE	Command status
MovingToEndStopCommand.clampingState	ACTIVE / INACTIVE	Clamping torque reached status

Note

In the case of travel to fixed stop with fixed stop detection by means of the following error, the entry for the position tolerance applicable to the fixed stop detection should be significantly less than the entry for the following error for the fixed stop detection.

Note

If the "travel to fixed stop" function is to be used in combination with the RecognitionMode=CLAMP_WHEN_TORQUE_LIMIT_REACHED setting, the following setting must also be made on the 611U and in SINAMICS:

- 611U
  Set parameter P1426 ("tolerance band for nset = nact message") to the maximum speed of the drive (P880).
- SINAMICS
  Set parameter P2163 ("Speed 4 threshold value") to the maximum possible value.
- Axis with digital drive link and no setting of a 1xx telegram (i.e. telegram without torque reduction and without actual torque identification)
  The "travel to fixed stop" functionality is only possible by means of the following error criterion.

You will also find further information about travel with torque limiting and travel to fixed stop in the corresponding drive documentation.

See also

Overview of torque limiting via torque reduction (Page 3010)

4.8.3.17 Technology data

Technology data can be specified cyclically from the controller to the drive or read cyclically by activating the technology data block.

The technology values are needed to implement winder functionality with SIMOTION, for example.

The setting of the motor type determines whether the physical quantity is force or torque (round-frame motor: torque, linear motor: force).

For more information, refer to *'LREAL' interconnection interface type* in the **Motion Control Basic Functions** manual.

The **additive data block 1** contains:

- Controller → drive:
  - Additive torque setpoint: 16 bits
    (display of the value in the **additiveTorque.value** system variable)
  - Positive torque limit (B+): 16 bits
    (display the value in the **torqueLimitPositive.value** system variable)
  - Negative torque limit (B-): 16 bits
    (display of the value in the **torqueLimitNegative.value** system variable)
- Drive → controller:
  - Actual torque value: 16 bits
    (display of the value in the **actualTorque.value** system variable)

As of SIMOTION V4.2, the technology data block is set up and connected automatically.

The technology data is appended as INT to the In/Out drive protocol. This is activated in the **technologicalData.enable** configuration data element.

The logical addresses for the technology data are displayed in the **technologicalDataOutInfo** and **technologicalDataInInfo** configuration data elements.

- They are set when the telegram is specified in the axis wizard. (When you set the axis telegram in HW Config, the relevant data must be created via the PZD area.)

The values B+, B- and additive torque setpoint

- Are indicated and entered according to their associated units.
- Are normalized to the specified maximum values (maxTorque, maxForce) before they are transferred to the drive and are sent to the drive as % values.
- Can be sent to the drive as values from 0 to +/- 200%.

The positive and negative torque limits (torqueLimitPositive (B+), torquelimitNegative (B-)) are sent as a weighting, rather than as a reduction.

The values are sent with reference to the 100% value:

- 4000H corresponds to 100%
- 7FFFH corresponds to 200%

Negative values are possible.

Note

The **SetpointDriverInfo.DriveData.maxTorque** or **SetpointDriverInfo.LinearMotorDriveData.MaxForce** configuration data and the P2003 parameter on the drive serve as normalization basis on the axis. The values must be equal.

See also Chapter Overview of torque limiting via torque reduction (Page 3010)
Section *Specification of the values and transfer from the drive*.

Creating technology data block 1 (e.g. with SINAMICS drive):

The drive has already been configured.

Telegram extension (as of V4.2)

As of V4.2, with SINAMICS drives and the **Use symbolic assignment** SCOUT setting (default setting) activated, telegram extension is set automatically by the system.

Telegram extension in HW Config (up to V4.1 SP1)

- Set standard telegram for the drive, then add to the PZD via Details:
  - Input: +1 (PZD)
  - Output: +3 (PZD)
- Save HW Config

Telegram extension in SCOUT (as of V4.1 SP1)

- In the project navigator, select the drive device (e.g. "SINAMICS_Integrated") and open the configuration:
  Project - CPU (e.g. D435) - drive device (e.g. SINAMICS_Integrated) - configuration
- Mark the line of the drive, e.g. "Drive_1"
- Click the "Insert line" button and select "Telegram extension"
- Set one word for the input data and three words for the output data
- Click the "Transfer to HW Config" button and confirm the comparison with HW Config
- Click the "Configure telegram" button
- For "PROFIBUS receive direction" and "PROFIBUS send direction", enter the telegram extension according to the table below at:
  Project - CPU (e.g. D435) - drive device (e.g. SINAMICS_Integrated) - Drives - Drive_x - Communication - PROFIBUS

Table 4-360 BICo interconnections

PROFIdrive telegram - receive direction		
	1st additional PZD (M_Add)	P1511
	2nd additional PZD (B+)	P1522
	3rd additional PZD (B-)	P1523
PROFIdrive telegram - send direction		
	1st additional PZD (M_act)	E.g. R0080 (actual torque value)

The values are normalized in the drive via P2003.

NOTICE
Scaling of additional torque 1 Parameter P1512 "Additional torque 1 scaling" must be interconnected so that additional torque (M_Add) is transferred correctly. Values from 0 to 100% can be set. A scaling factor of 100% corresponds to the transfer of the value configured in M_Add.

Activation in SIMOTION

The technology data block is created when the axis is configured.

The technology data block can only be assigned correctly by the system for active symbolic assignment, manual telegram setting and deactivated automatic telegram extension when the correct BICO interconnection for the technology data block is available on the drive side.

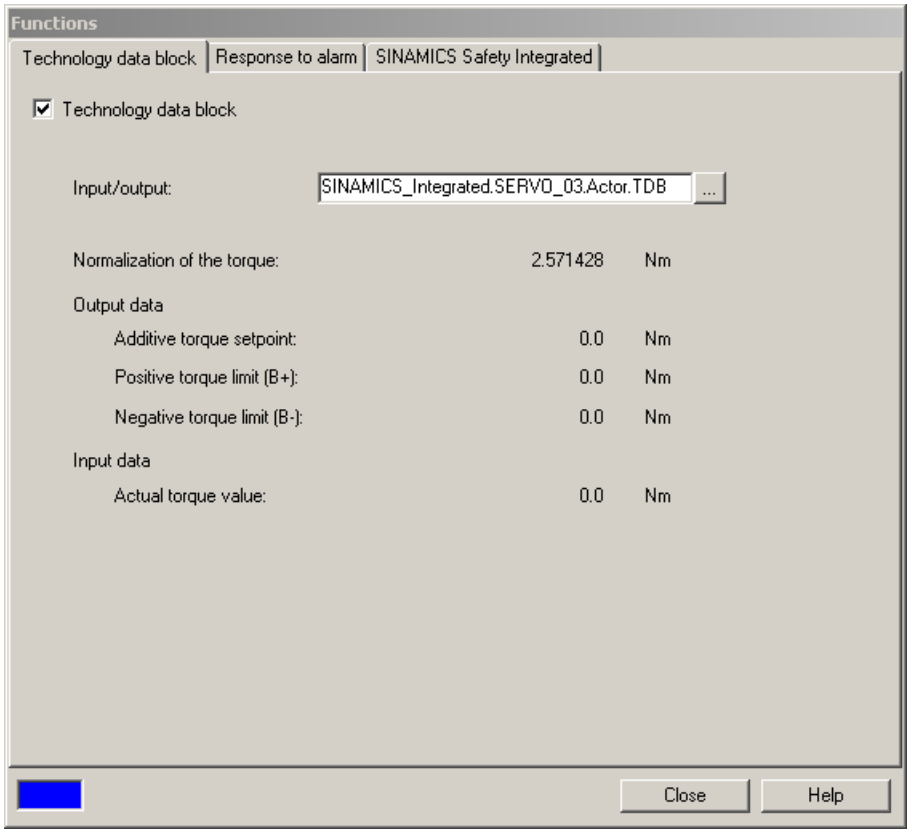


Figure 4-876 Activating a technology data block

Alarms

20005 "Logical address of technology data block driver not found"

The limiting values and additive values are initialized/transferred regardless of the activation and error status of the axis.

4.8.3.18 Torque limiting B+/B- (V3.2 and higher)

The torque limits B+/B- can be cyclically specified on the electric axis downstream of the speed controller using the expanded drive message frame to the drive.

As a prerequisite, the technology data block (Page 3018) must have been activated on the electric axis. The torque limits B+ and B- are set downstream of the speed controller.

By specifying the upper limit B+ and the lower limit B-, it is possible to enter a specific range that does not have to be symmetrical with respect to the zero position.

The limit values can be predefined on the axis directly by means of cyclically effective system variables. These values can be preset.

The limit values can also be defined on the axis by means of interconnection with other technology objects.

The B+ value and the B- value can be interconnected by means of an LREAL interface or an LREAL connector (using a formula object or a controller object, for example).

See also:

- **SIMOTION Supplementary Technology Objects** function description
- **Motion Control Basic Functions** Manual, Sections *Interconnection using general interconnection dialog box* and *'LREAL' interface type*

Note

If the manipulated variable is inverted on the Axis technology object, then B+/B- are also reversed on the Axis technology object. You must take this into account when specifying the torque limits B+/B-.

Units

The unit settings are used.

The physical quantity depends on the type of motor:

- For rotary motors: Torque [Nm]
- For linear motors: Force [N]

Commands and effectiveness

The limiting is deactivated by default and has to be explicitly activated.

The torque limits are activated using the **_enableAxisTorqueLimitPositive()** / **_enableAxisTorqueLimitNegative()** commands.

The command specifies whether the limits relate to the interconnected value, the system variable or a value defined on the command itself. If they relate to the system variable, then this is applied cyclically; in other words, a change in the system variables takes effect immediately.

If torque limitation B+/B- is activated using the above commands, this will deactivate the following monitoring and limiting functions:

- Following error monitoring
- Velocity error monitoring via reference model
- Time-outs for positioning monitoring and standstill monitoring

The torque limits are not disabled with **_disableAxis()**; they must be disabled explicitly using a command.

The torque limits are deactivated or aborted using the following commands:

- **_disableAxisTorqueLimitPositive()** / **_disableAxisTorqueLimitNegative()**
- **_resetAxis()**, **_restartAxis()**, etc.
- In the case of a transition to STOP

If the function is not activated, the maximum torque and maximum force are output via the interface in accordance with **maxTorque** and **maxForce**.

With activated torque limiting B+ / B- (**_enableAxisTorqueLimitPositive()** / **_enableAxisTorqueLimitNegative()**), as of V4.2, with setting **TypeOfAxis.SetPointDriverInfo.DriveData.torqueReference**, the = MAX_VALUE maximum of the set value in **~.DriveData.maxTorque** or **maxForce** is transferred to the drive.

In the versions <V4.2 up to 200% of this value are possible with activated torque limitation.

If the torque limiting function B+ / B- is not activated, the maximum value set in **~.maxTorque** is transferred in V4.2 as well as in versions <V4.2.

When using the technology data block and with the setting **~.torqueReference** = MAX_VALUE in **~.maxTorque** , the 100% reference value for torque transmission to the drive must be set, i.e., for the value corresponding to p2003 (reference torque), a greater maximum torque is possible in the drive, in accordance with the value in p1520 (maximum torque).

The separation of reference and maximum torque has been introduced in the reference variables with V4.2.

In projects with V4.2 or higher, the new configuration data **~.torqueReference** is set to MAX_VALUE. As such, only the reference torque value can be set as maximum torque limitation without changing projects.

Remedy:

If, in the case of activated torque limiting, a value greater than the reference value in p2003 is transferred to the drive**, ~.torqueReference** = NOMINAL_VALUE needs to be set, in **~.nominalTorque** the value corresponding to p2003 and in **~.maxTorque** the value corresponding to p1520 must be entered.

With the deactivated torque limiting function B+ / B, the set value in **maxTorque** is transferred to the drive.

For new TOs created in V4.2, **~.torqueReference** = NOMINAL_VALUE is set, as well as for the default set adaption the reference values for **~.nominalTorque** and **~.maxTorque** for runtime are transferred from the drive.

The position-related monitoring can remain active even with active torque limiting. For this purpose, there is the **TypeOfAxis.ServoMonitoring.motionMonitoringWhenTorqueLimiting** configuration data as of V5.1. When activated, this ensures that the position monitoring, positioning monitoring and/or following error monitoring that was active before the torque limiting remains active during the limiting.

System variables

Table 4-361 System variables for torque limitation

Variable	Meaning
torqueLimitPositiveIn	Value, state and validity of the interconnected positive limitation
torqueLimitPositiveInDefault	Default
torqueLimitNegativeIn	Value, state and validity of the interconnected negative limitation
torqueLimitNegativeInDefault	Default
torqueLimitPositive	Shows the programmed limit value
torqueLimitNegative	

Variable	Meaning
torqueLimitPositiveCommand	Command status / function status
torqueLimitNegativeCommand	
motionMonitoringWhenTorqueLimiting	The position-related monitoring is active during active torque limiting.

Suppression of the "Motor blocked" fault message of the drive

Message: **F07900 (N, A) drive: Motor blocked / speed controller at its endstop**

Cause: The motor has been operating at the torque limit for longer than the time specified in p2177 and below the speed threshold set in p2175.

If the fault message is to be suppressed, the **_setAxisStw()** function can be used once to set bit 8 in ctrl word2 for message suppression.

The setting via the **_setAxisStw()** function has a storing effect, i.e., it is kept during and after the period that a torque reduction is activated.

The p2175 (0.0) parameter to suppress the fault message on the drive.

See also

Coupling of digital drives (Page 2895)

Technology data (Page 3018)

4.8.3.19 Additive set torque (V3.2 and higher)

An additive set torque can be transmitted to the drive on the axis.

This requires that an axis with digital drive link is used and that the technology data block (Page 3018) is activated.

The additive setpoint torque can be set on the axis directly using a cyclically effective system variable (**defaultAdditiveTorque**). This value can be preset.

The additive setpoint torque can also be defined on the axis by means of interconnection with other technology objects.
The additive torque can be interconnected by means of an LREAL interface or an LREAL connector (using a formula object or a controller object, for example).

See also:

- **SIMOTION Supplementary Technology Objects** function description
- **Motion Control Basic Functions** manual, *Interconnection using general interconnection dialog box* and *'LREAL' interface type* chapters

The function is activated via the **_enableAxisAdditiveTorque()** command. Without an activation, the transmitted additive torque is = 0.

The command specifies whether the limits relate to the interconnected value, the system variable or a value defined on the command itself. If they relate to the system variable, then this is applied cyclically; in other words, a change in the system variables takes effect immediately.

The **additiveTorque** configuration data element can be used to determine whether the last valid value or the replacement value is used if an interconnection value is activated but is invalid. After system start-up, the last valid value is 0.

The additive set torque can also be specified directly via the replacement value.

The replacement value can be preset.

This value takes effect immediately after a change if the function is active.

For more information, refer to *'LREAL' interconnection interface type* in the **Motion Control Basic Functions** manual.

Unit

The unit setting is used.

The physical quantity depends on the type of motor:

- For rotary motors: Torque [Nm]
- For linear motors: Force [N]

Commands

_enableAxisAdditiveTorque() enables the additive torque.

The additive set torque is not disabled with **_disableAxis()**; instead it must be explicitly deactivated via a command.

It is disabled/canceled

- with **_disableAxisAdditiveTorque()**
- In the case of **_resetAxis()**, **_restartAxis()**, etc.
- In the case of a transition to STOP

System variables

Table 4-362 System variables for the additive set torque

Variable	Meaning
additiveTorqueIn	Value, state and validity of the additive set torque
defaultAdditiveTorque	Default
additiveTorque	Value that is sent to the drive (before normalization)

Suppression of the "Motor blocked" fault message of the drive

See Torque limiting B+/B- (V3.2 and higher) (Page 3021).

See also

Coupling of digital drives (Page 2895)

4.8.3.20 Force/pressure control

Overview of force/pressure control

Force/pressure control requires at least the **position axis technology**. If a hydraulic axis is used as a hydraulic drive axis, force/pressure control is also possible here.

Pressure control and force control are available on the position axis. The actual pressure or force value is measured using an external sensor. Several force/pressure sensors can be connected to the axis. The actual force/pressure values can be smoothed with a **PT1 filter**. A **PT2 filter** is available at the output of the pressure control.

The manipulated variable on the drive is still the speed or velocity setpoint.

Monitoring for maximum force, control deviation, force/pressure entry (see positioning monitoring) and force/pressure end value is performed on the actual value side. Limiting functions on the setpoint side include speed setpoint limiting (manipulated variable limitation) and anti-windup (I-component) with manipulated variable limitation.

Conditions can be specified in a specific activation command **_enableForceControlByCondition()** for switchover from position-controlled motion to force/pressure controlled status. The force/pressure setpoint or profile is specified using commands.

The commands for force/pressure control can also be programmed on the virtual axis. Commands that cannot be executed, such as activate force/pressure controller, position-related profile, or data set commands generate a technological alarm, **Technological alarm: 030015: Technology not configured**.

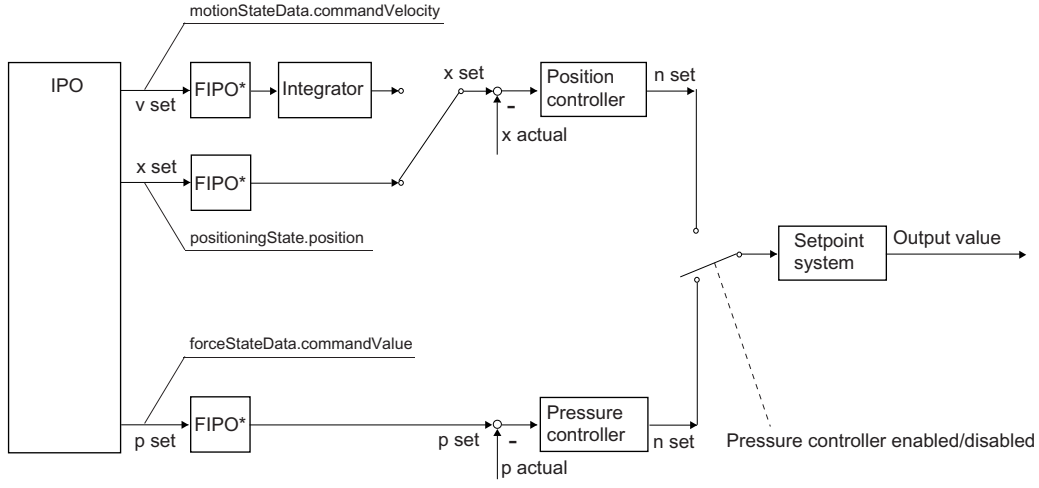


Figure 4-877 Overview of control structure for force/pressure control

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

You can call signal flow diagrams under the Axis object in the project navigator. The signal flow diagram illustrates the signal path and enables major items of configuration data and important system variables to be parameterized using the signal flow.

Possible operating modes with force/pressure control

- **Force/pressure control active, setpoint mode**
  Actual force/pressure value is controlled to force/pressure setpoint.
  Monitoring and limiting functions are active.
- **Force/pressure control active, follow-up mode**
  Force/pressure setpoint is controlled to actual force/pressure value.
  Force/pressure monitoring and limiting functions are not active in follow-up mode.
- **Program simulation**
  (analogous to positioning mode)
  Force/pressure setpoints are calculated but not output.
  Actual force/pressure value is set to force/pressure setpoint.
- **Velocity-limited mode**
  The closed-loop force/pressure control or open-loop force/pressure control is the master. The velocity is limited in parallel.

Remark

While the force/pressure controller is active, the position values of the IPO are corrected.

Switchover between position-controlled motion and force/pressure-controlled motion is performed by the application.

Fine interpolation

The force/pressure values undergo linear fine interpolation (V4.1 SP1 and later).

See also

Force/pressure control with hydraulic speed-controlled axes with Q valve only (Page 3136)

Technology data (Page 3018)

Configuration of the actual force/pressure value sensors

More than one pressure or force sensor can be configured.

The various sensors can be of different types and have different interfaces. The sensor for the transition criterion can be specified in the switchover command.

There can be up to eight (8) sensors for the force/pressure value on the axis and up to eight (8) binary inputs on the axis, for example, for switchover criteria.

Because the sensor can be accessed via the logical address, the same sensor can therefore be used on several axes. The sensor data must be set separately on each axis.

In V4.0 or higher, a calculated value from the user program can also be specified as the actual sensor value via the cyclically effective **forceActualValueSetting.value** system variable. This requires the setting **additionalSensorType**= SET_ACTUAL_VALUE.

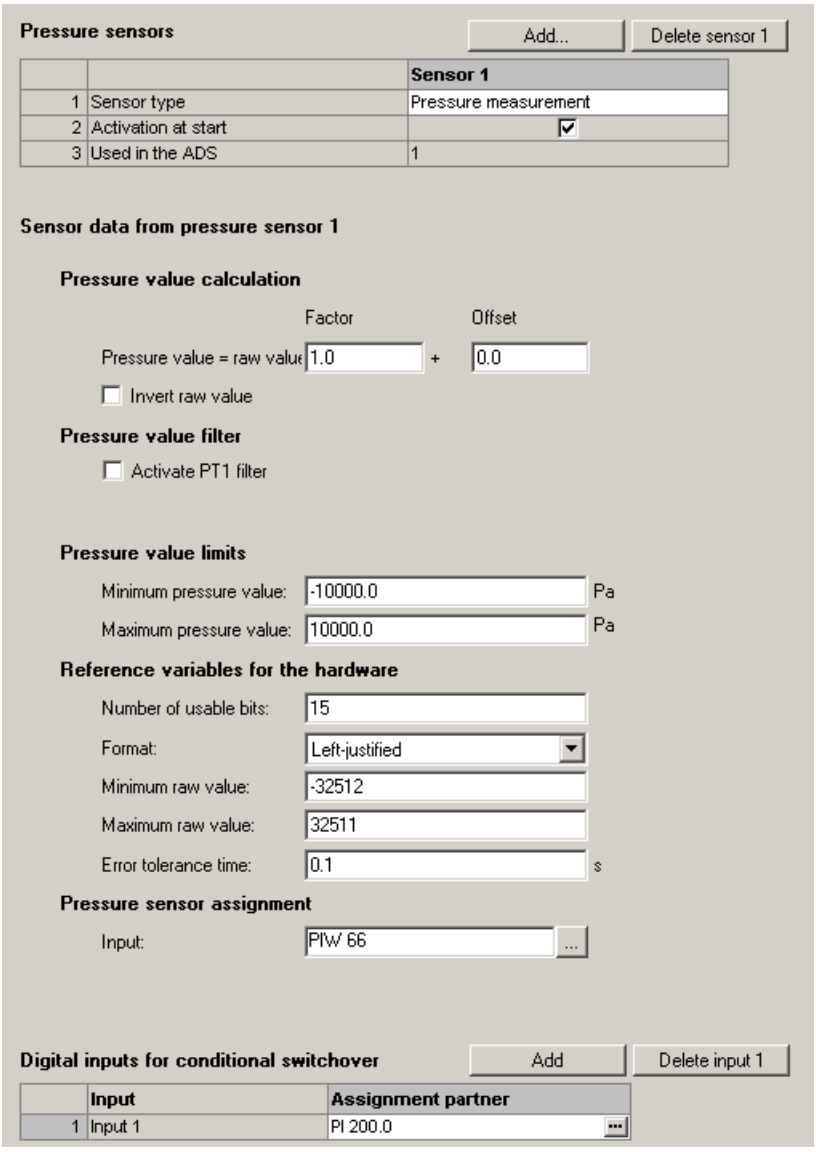


Figure 4-878 Setting the force/pressure sensor

The force/pressure sensor interface can be activated and deactivated on the axis using the **_enableAxisInterface()** and **_disableAxisInterface()** commands.

The Activated/Deactivated force/pressure sensor interface status is displayed in the **additionalSensorData.additionalSensorData[i].input** system variable.

Controller for force/pressure control

The force/pressure control is carried out via an adjustable PID controller.

The controller has the same structure for force/pressure control and for force/pressure limiting. One controller for both tasks is available; the controller parameters for each task can be accessed by switching the data set.

The I component of the controller is implicitly set during enabling. When the data set is changed over, the system sets the I component so that a continuous setpoint curve is obtained.

When pressure limiting is enabled, the most recent position setpoint is maintained during the switchover to pressure control; otherwise, the actual value is applied. The force/pressure setpoint is set the same as the actual force/pressure value at the switchover time.

The initial value of the force/pressure controller can be limited by an upper value (**forceControllerData.outputLimits._max**) and a lower value (**forceControllerData.outputLimits._min**). It is thus possible to react in one direction only, e.g. to maintain pressure, to move in one direction only in the event of overpressure, to not actively build up pressure.

If the interpolator cycle clock is not the same as the position control cycle clock, a linear fine interpolation is carried out for the force/pressure setpoint. The force/pressure setpoints are limited to a maximum value.

If the hydraulic functionality of the axis is used, specific factors can be predefined for the sliding friction and the additive offset for the force/pressure control.

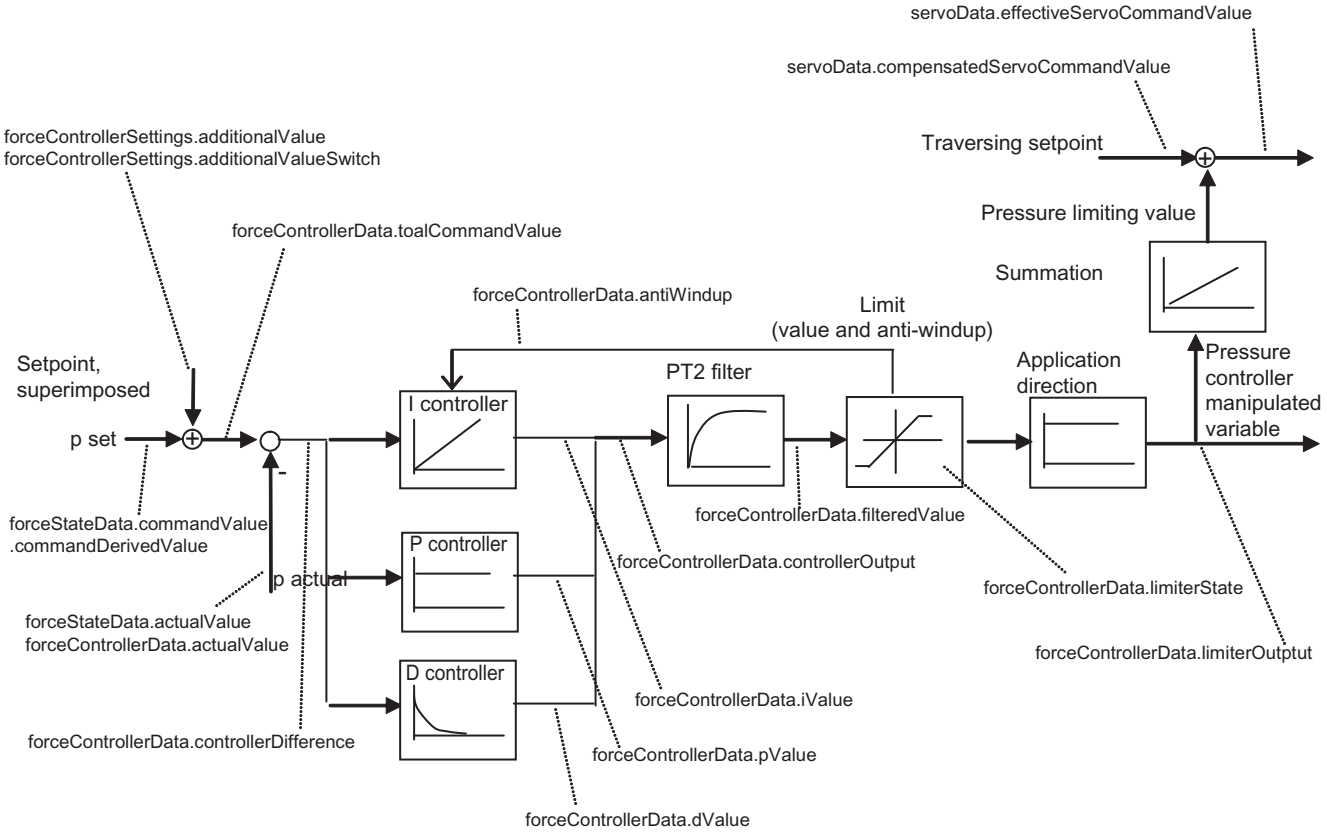


Figure 4-879 Overview of pressure controller/pressure limiting controller

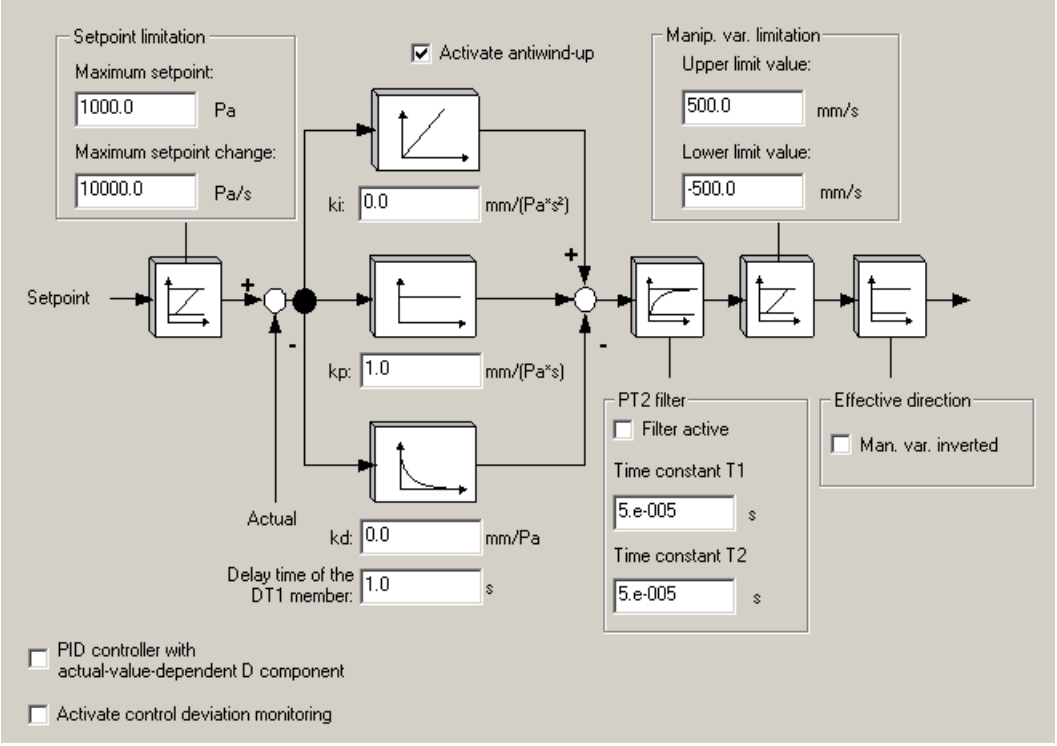


Figure 4-880 Controller setting

Reducing the I component via the user program (as of V4.1 SP1):

It is now possible to reduce the I component of the pressure controller with a PT1 filter (setting in **PID_Controller.iValueFeedbackTimeConstant** configuration data element) using the write-accessible **forceControllerSettings.integratorMode** system variable from the user program.

This enables faster reduction of the I component, for example, following the switchover from pressure limiting to pressure control.

See also

Compensations that are active only on the axis with hydraulic functionality (Page 3129)

Monitoring functions / limiting functions / emergency strategies with active force/pressure control

The actual force/pressure value is monitored for a maximum value or limit value. The control deviation is monitored for a maximum value.

Based on a force/pressure entry window, you can monitor correct termination of a force/pressure profile (force/pressure entry monitoring) – refer to positioning monitoring during positioning.

You can also monitor observance of a constant force/pressure setpoint within a tolerance band (force/pressure end value monitoring) – see standstill monitoring on a position-controlled axis.

Travel to software limit switch:

When traveling to the software limit switch, pressure limitation is deactivated.

You must ensure that the maximum deceleration values are set correctly so that the setpoints of the software limit switches are not overtraveled when the system switches automatically from pressure-limited operation to position-controlled operation.

Limiting functions are available for:

- Force/pressure setpoint
- Force/pressure controlled variable
- Actual force/pressure value

Note

To allow an immediate stop with active force/pressure control, e.g. via the **_stopEmergency()** command (stopDriveMode=WITH_COMMAND_VALUE_ZERO), the control switches to pressure limitation and, with pressure limitation active for the force/pressure setpoint, moves to velocity 0 using the specified velocity profile for active position control.

If the STOP_EMERGENCY alarm response occurs for active pressure control, a switch is made to position control using pressure limitation. The pressure setpoint is active at the switching point. If in this case, the actual pressure value is higher than the pressure setpoint or limit value, the actual pressure value will be removed appropriately.

See also

Dynamic limiting functions (Page 2999)

Activating the force/pressure control

Force/pressure control is activated by commands from the application.

SIMOTION supports different activation modes.

- **Direct activation**
  Direct activation is achieved using the axis-enable command (**_enableAxis()** or **_enableQFAxis()**) (**forceControlMode** parameter = ACTIVE). When enabled, the last actual force/pressure value is taken as the force/pressure setpoint. The position controller must also be enabled.
  The direct activation of the pressure control is possible in the stopped state (**motionStateData.stillStandVelocity** = ACTIVE).
- **Direct deactivation**
  Direct deactivation occurs via the command to reset the axis enable or the command for axis enable without the parameter for force/pressure control.
- **Automatic activation with condition**
  There is a separate command for automatic activation. The conditions are verified and the switchover takes place in the position control cycle clock. The axis is switched over to a pressure-time profile. The conditions (force, pressure, position, time, and input) are specified in the command and can be linked in the command in several stages. The conditions can also be changed before they occur by reissuing the command. The force, pressure, position, and time are stored at the time of switchover. The values are available through system variables.

Switching from position control to force/pressure control:

To obtain a continuous manipulated variable curve for switching from the position control to force/pressure control, the I component of the process controller will be initialized so that the new manipulated variable from the process controller corresponds to the estimated manipulated variable from the position controller.

The estimated manipulated variable is the last manipulated variable of the position controller plus the filtered actual acceleration multiplied with the clock time.

Switching from force/pressure control to position control:

To obtain a continuous manipulated variable curve for switching from force/pressure control to force/pressure control, the setpoint generation and the position controller will be restarted. The current actual value measured by the encoder modified with the estimated following error is used as starting value for the set position.

The estimated following error is calculated from the current manipulated variable multiplied by the replacement time constant of the position control loop PTC (**typeOfAxis.NumberOfDataSets.DataSet_1.DynamicData.positionTimeConstant** configuration data item).

The position controller then outputs in the first cycle clock after the switching the same manipulated variable as the force/pressure controller before the switching.

Note

In certain situations, the reference to the current manipulated variable can cause problems. This is the case, for example, when an offset compensation is active. The PTC for the switching time should then be set to zero. PTC can be switched with the data set.

Force/pressure setpoint specification

Force/pressure setpoints can be specified as follows:

- Directly as a value
- Via a time-related force/pressure profile
- Via a position-related force/pressure profile

The derivation value for the transition to the setpoint, or starting value can be specified in the command. The last force/pressure setpoint is maintained at the end of the profile.

Even with direct setpoint value specification, the command requires different runtimes for its execution on account of the possible transition to the pressure setpoint (pressure entry monitoring). Synchronous and asynchronous command execution can be set.

With direct force/pressure setpoint specification, the setpoint is always active immediately. **mergeMode** can be set in the force/pressure profile commands (sequential execution).

In position-related profiles the absolute position reference of the profile applies at the starting point. During ramp up to the profile and ramp down from the profile, the **additionalSensorData.derivedValue** refers to derivation of the setpoint over time.

An error message is output for starting at a position outside the profile.

Commissioning procedure for force/pressure control

Procedure for commissioning:

1. Commission the drive.
2. Commission the position controller for positioning mode.
3. Set the force/pressure controller. To do this, you can define setpoints for the force/pressure controller via the additive force/pressure setpoint using a function generator.

Force/pressure control with velocity limiting

On the position axis, velocity limiting (flow rate limiting for hydraulic functionality) can be activated in parallel with the force/pressure control. Velocity limitation is effective within pressure control.

It is activated in the same way, for example, as force limiting with limiting value setting or limiting profile enabling. Explicit deactivating is possible by means of a specific command.

If the velocity/flow rate limit is reached, the flow rate is limited, and the I component is stopped in the force/pressure controller.

Like torque limiting, velocity limiting can be activated in parallel with the motion commands.

Velocity limiting is enabled by means of the following commands:

- Activate position-related limiting profile
  _enablePositionLockedVelocityLimitingProfile()
- Activate time-related limiting profile
  _enableTimeLockedVelocityLimitingProfile( )
- Activate limiting value
  _enableVelocityLimitingValue()

Velocity limiting is disabled by means of the following command:

- **_disableVelocityLimitingValue()**

In the event of an error, the velocity limiting remains inactive, the force/pressure control is replaced by the force/pressure limiting and the velocity limiting is disabled.

The velocity limiting direction can be activated by means of the **velocityLimitingDirection** parameter (as of V4.1 SP1).

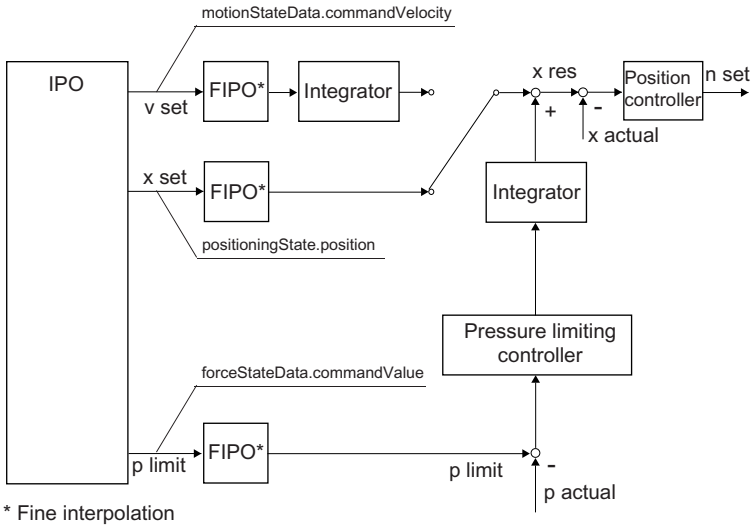
The set direction is indicated in **velocityLimitingCommand.velocityLimitingDirection**.

4.8.3.21 Force/pressure limiting

Overview of force/pressure limiting

The force/pressure limiting functionality requires **position axis technology**. If a hydraulic axis is used as a drive axis, force/pressure limiting is possible here, too.

The position controller is active as a lower-level controller if force/pressure limiting is active.



Overview of control structure for force/pressure limiting

With force/pressure limiting, the force/pressure controller is not activated until the force/pressure limit is exceeded ($\mathbf{p}_{act} > \mathbf{p}_{limit}$). It is limited to positive force/pressure values.

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

The force/pressure limiting is switched on using commands, e.g.:

- **_enableTimeLockedForceLimitingProfile()**
- **_enableMotionInPositionLockedForceLimitingProfile()**
- **_enablePositionLockedForceLimitingProfile()**
- **_enableForceLimitingValue()**
- **_enableForceLimitingByCondition()**

Time-related monitoring is deactivated.

The force/pressure limiting remains active in the event of an error, except for the RELEASE_DISABLE and OPEN_POSITION_CONTROL error reactions.

Position-related monitoring (e.g. following error monitoring or positioning monitoring) are deactivated when pressure control is activated or via the pressure limitation command.

You can configure the monitoring functions to remain active with external pressure/force limiting in the **servoMonitoring.motionMonitoringWhenExternalForceLimiting** configuration data element.

A PT2 filter on the force/pressure limiting controller output (as of V3.2) will prevent abrupt signal changes. The filter time constants can be modified online and take effect immediately.

The pressure limiting value is not subject to fine interpolation if **_enableForceLimitingValue()** or **_enable...LimitingProfile()** with derivativeLimitingMode=WITHOUT_LIMITING is set.

The status of the pressure limiting is displayed via the system variable **forceControllerData.limitingState**. The requirement for updating the status variable is that the pressure controller must be in operation (kp must not be equal to 0).

See also

Force/pressure limiting with hydraulic speed-controlled axes with Q valve only (V4.0 and higher) (Page 3137)

Positioning with active force/pressure limiting (V3.2 and higher)

With position-related motions and active pressure limitation, cyclic continuation of the interpolator position and velocity can be set.

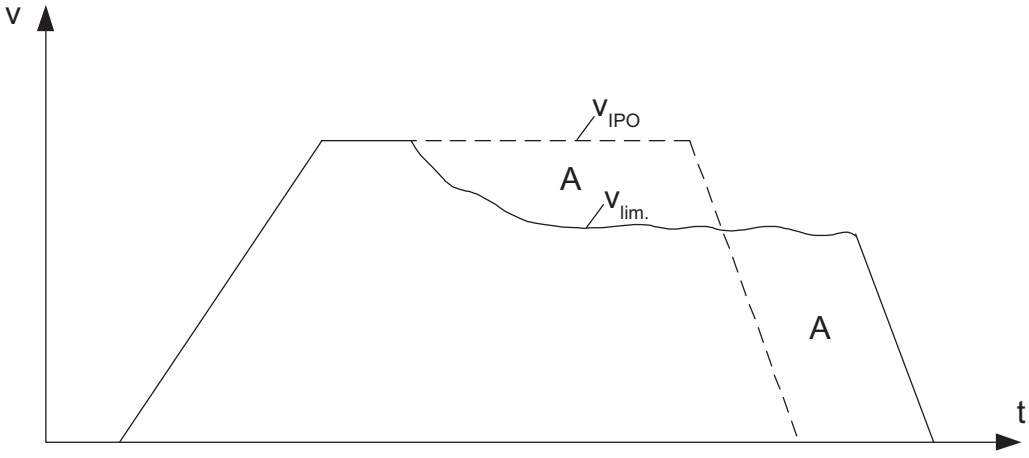


Figure 4-881 Position-related motion with pressure limitation and cyclic continuation

This example shows a v/t profile in which the velocity is reduced by the pressure limitation. Cyclic continuation allows continuous interpolation to the end point, starting from the current position and, optionally, the velocity.

The **decodingConfig.cyclicSetUpInForceLimiting** configuration data element defines how the continuation should take place.

With the setting **decodingConfig.cyclicSetUpInForceLimiting** = POSITION_BASED, cyclic continuation at the current set position and the non-reduced velocity of the interpolator takes place.

With the setting **decodingConfig.cyclicSetUpInForceLimiting** = POSITION_AND_DYNAMIC_BASED, cyclic continuation at the current position and velocity takes place.

Recommended setting: **decodingConfig.cyclicSetUpInForceLimiting** = POSITION_BASED

Braking phase initiated via software limit switch

Cyclic continuation stops from the time at which the braking phase is initiated because the software limit switch is reached. The drive shuts down at maximum deceleration. The pressure limitation remains active.

At standstill, the axis switches from position control to velocity control.

Behavior during positioning/moving to the target position

- Reaching the target position and actual position within the positioning window
  - The target point is reached with respect to the setpoint with cyclic continuation
  - The axis is positioned within the positioning window
  - MOTION_DONE is set and the command is ended as EXECUTED
- Reaching the target position within the positioning window
  - The target point is reached with respect to the setpoint with cyclic continuation
  - The axis is not positioned within the positioning window
  - MOTION_DONE is not set and the positioning monitoring alarm is suppressed due to active pressure limitation
  - Standstill monitoring is activated, and an alarm is triggered if the standstill window is overshot, which cancels the command if necessary
- As a result of pressure limitation, the target position is not reached even through cyclic continuation
  - The target point is not reached with respect to the setpoint even with cyclic continuation
  - No alarm is triggered
  - The command is not ended

Note

Cyclic continuation is not practicable with active torque limitation. In this case, the torque limitation takes place in the drive, and the setpoints limited by the drive are not known to the control.

Increase limiting for pressure profiles and pressure limiting (V3.2 and higher)

Increase limitation can be specified for **_enable...ForceLimiting()** commands.

Determining the output value for the increase limitation

When determining the output value, it is necessary to ascertain whether the pressure limitation is already active when the **_enable...ForceLimiting()** command is activated.

Note

If you prefer to specify the output value directly:

- Use **_enableForceLimitingValue()** to set the output value

- Wait until the value is reached

- Use a new command to ramp up to the target value

A programmed increase limitation can then be used to move to new values.

Initial activation of the pressure limitation

- Output value when pressure sensor not present is FValue=0.0
- If pressure sensor is activated, the output value is the measured actual pressure value
- When pressure control is triggered by means of pressure limitation, the output value is the pressure setpoint in the IPO

New pressure limitation command with pressure limitation already activated

The output value is the limit value in the IPO.

4.8.3.22 Data sets

Data set overview

The configuration data belonging to the axis data set (ADS) can be seen on the **Axis data sets** tab in the Configuration dialog for the axis.
Details of the parameters can be found in the TP Cam Configuration Data Reference List.

ADS contain axis configuration data for the axis, especially that pertaining to the servo (controller data, encoder data, process data, data for axis gearing, and data for force/pressure control, for example).

Data sets must be defined and activated as a group because some data, such as controller data, can only be activated simultaneously in groups to ensure consistency of the controller and function.

Data set switchover / encoder switchover

Data sets can be switched over in the runtime application. This switchover is also possible within one IPO cycle clock (e.g., to activate new controller data or to switch over to a second encoder).

One axis can use several encoders for the position control, but only one encoder at a time.

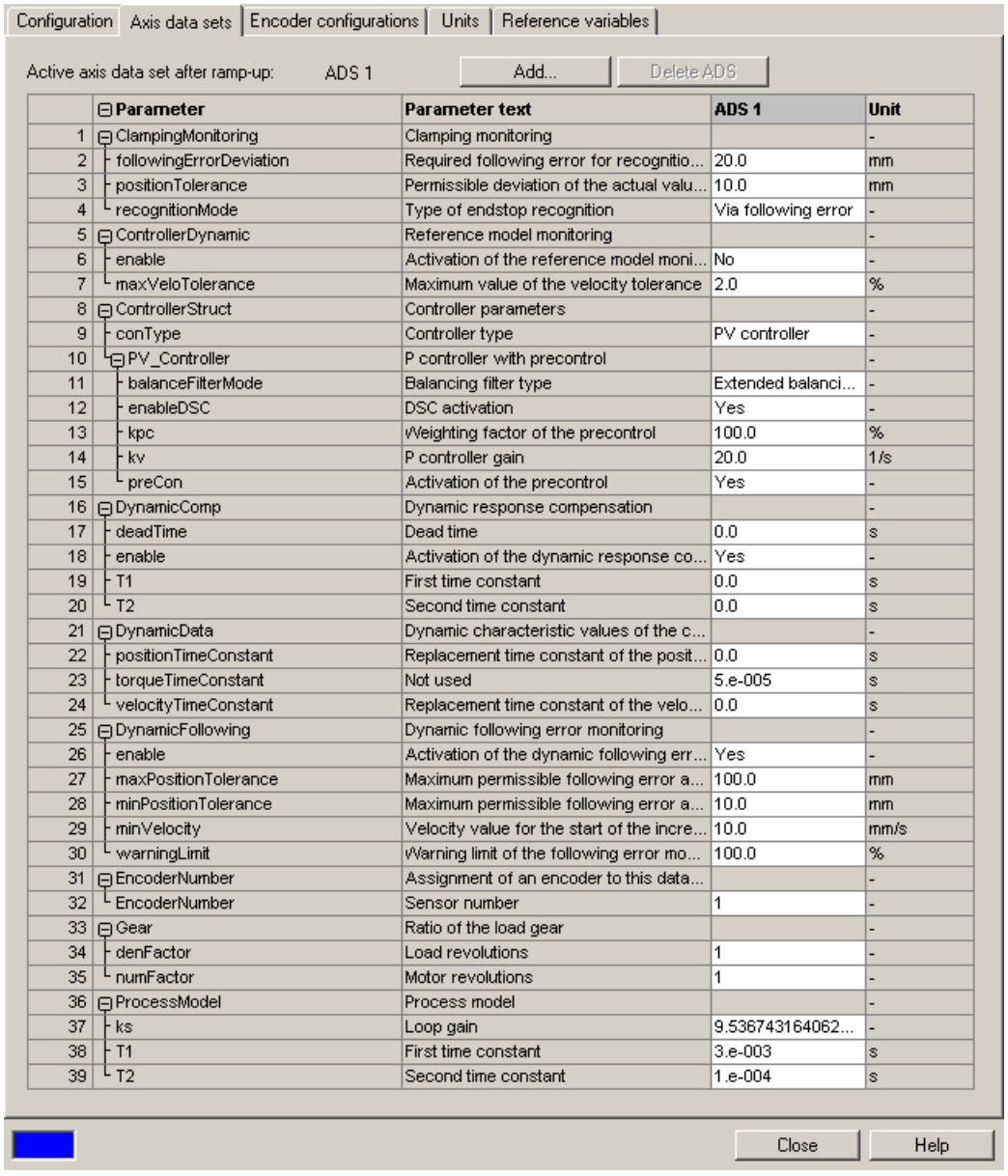
One data set and one encoder are assigned to each axis when it is set up. If you need additional data sets or encoders, they must be added. Data sets can be added in the Axis technology object under **Configuration** on the **Axis data sets** tab.

In V4.0 and higher, the **smoothingTimeByChangeDifference** configuration data element can be used to configure a switchover smoothing filter. This switchover smoothing filter is active for all status transitions/switchovers in which an offset in the manipulated variable can occur due to the switchover. Gear switchover in the data set is not smoothed.

The data sets must have the same control structure and must not contain any change that requires a restart. For example, if data set 1 has DSC, data set 2 must have DSC too, or if following error monitoring is disabled for data set 1, it must also be disabled for data set 2. Data sets with different structures cannot be set up.

Data sets are specified and selected using their numbers.

You can also define which data record is to be loaded for the technology object during CPU power up.



	Parameter	Parameter text	ADS 1	Unit
1	ClampingMonitoring	Clamping monitoring		-
2	followingErrorDeviation	Required following error for recognitio...	20.0	mm
3	positionTolerance	Permissible deviation of the actual valu...	10.0	mm
4	recognitionMode	Type of endstop recognition	Via following error	-
5	ControllerDynamic	Reference model monitoring		-
6	enable	Activation of the reference model moni...	No	-
7	maxVeloTolerance	Maximum value of the velocity tolerance	2.0	%
8	ControllerStruct	Controller parameters		-
9	conType	Controller type	PV controller	-
10	PV_Controller	P controller with precontrol		-
11	balanceFilterMode	Balancing filter type	Extended balanci...	-
12	enableDSC	DSC activation	Yes	-
13	kpc	Weighting factor of the precontrol	100.0	%
14	kv	P controller gain	20.0	1/s
15	preCon	Activation of the precontrol	Yes	-
16	DynamicComp	Dynamic response compensation		-
17	deadTime	Dead time	0.0	s
18	enable	Activation of the dynamic response co...	Yes	-
19	T1	First time constant	0.0	s
20	T2	Second time constant	0.0	s
21	DynamicData	Dynamic characteristic values of the c...		-
22	positionTimeConstant	Replacement time constant of the posit...	0.0	s
23	torqueTimeConstant	Not used	5.e-005	s
24	velocityTimeConstant	Replacement time constant of the velo...	0.0	s
25	DynamicFollowing	Dynamic following error monitoring		-
26	enable	Activation of the dynamic following err...	Yes	-
27	maxPositionTolerance	Maximum permissible following error a...	100.0	mm
28	minPositionTolerance	Maximum permissible following error a...	10.0	mm
29	minVelocity	Velocity value for the start of the incre...	10.0	mm/s
30	warningLimit	Warning limit of the following error mo...	100.0	%
31	EncoderNumber	Assignment of an encoder to this data...		-
32	EncoderNumber	Sensor number	1	-
33	Gear	Ratio of the load gear		-
34	denFactor	Load revolutions	1	-
35	numFactor	Motor revolutions	1	-
36	ProcessModel	Process model		-
37	ks	Loop gain	9.536743164062...	-
38	T1	First time constant	3.e-003	s
39	T2	Second time constant	1.e-004	s

Figure 4-882 Configuration of axis data sets in SIMOTION SCOUT

The individual data sets appear next to each other on the **Axis data sets** tab.

Configuration data which must be the same for all data sets can only be entered in ADS 1. A change in ADS 1 is only mapped in the other ADS if the configuration data has to be set identically for all ADS.

If parameters for axis data sets undergo changes in the expert list, consistency conditions may be violated (check during download). This may happen with controller settings, for example. In this case, the line and cell are highlighted on the **Axis data sets** tab.

Only variables which are effective immediately can be changed online.

The parameter assignment dialog boxes for the Axis technology object (e.g. Control) display the configuration data for one data set at a time. A function bar at the bottom of the dialogs features

a combo box  with the option to switch over the axis data set. This switches over the data set displayed (not the active data set).

Note

Please also note the tooltips above the parameter fields in the individual screen forms for the axis. These show the name of the variable, from which you can also tell whether the variable is assigned to a specific axis data set.

The number of data sets is specified with the structural elements for the **NumberOfDataSets** configuration data element. Up to 16 data sets are possible.

The data set changeover mode is specified in the **NumberOfDataSets.changeMode** parameter.

Table 4-363 Parameter NumberOfDataSets.changeMode

NEVER	No data set changeover takes place.
IN_STANDSTILL	Data set changeover takes place when the standstill signal is applied.
IMMEDIATELY	Data set changeover takes place immediately.
IN_POSITION	Data set changeover takes place when the positioning window is reached.

Switchover to another encoder via the data set switchover

The data set shows which encoder is used with this data set. You can define up to 8 different encoders. All configured measuring systems (encoders) are internally active, and the measured values are updated cyclically.

The encoder contained in the data set is specified and selected using its number (EncoderNumber).

The basic settings for encoders are made under **Configuration** on the **Encoder configuration** tab.

Note

Please also note the tooltips above the parameter fields in the individual screen forms for the axis. These show the name of the variable, from which you can also tell whether the variable is assigned to a specific encoder.

Activating data sets

Data set commands can be used to read, write, and activate data sets on the axis.

The configuration data of a data set are contained in the structure **TypeOfAxis.NumberOfDataSets.DataSet_n** (n: data set number). The display depends on the technology set on the axis.

Data sets must be defined and activated as a group because some data, such as controller data, can only be activated simultaneously in groups to ensure consistency of the controller and function.

- **_getAxisDataSetParameter()** reads a data set on the axis
- **_setAxisDataSetParameter()** overwrites a data set on the axis
- **_setAxisDataSetActive()** sets the data set specified in the function parameter to active

Note

If the active measuring system is switched via a data set changeover, the **_setAndGetEncoderValue()** system function should be used to synchronize the two measuring systems before the switchover takes place. This will prevent unwanted compensating movements of the position controller if differences in position are identified.

See also

Activating data sets (Page 3174)

Writing a data set (Page 3175)

Reading a data set (Page 3175)

Writing the force/pressure-specific data of a data set (Page 3175)

Reading the force/pressure-specific data of a data set (Page 3176)

Writing the data of the data set (hydraulic functionality only) (Page 3176)

Reading the force/pressure-specific data of the data set (hydraulic functionality only) (Page 3176)

4.8.3.23 Traversing with user-defined motion and force/pressure profiles

Overview of traversing with user-defined motion and force/pressure profiles

In addition to traversing/positioning via system functions, the axis can also be traversed directly via user-defined profiles.

The curve specification of a cam is used as profile function. The cam can be set using the SIMOTION SCOUT engineering system (CamEdit, CamTool) or, alternatively, by means of the application.

A system command is used to assign the appropriate technological variables of the axis to the domain and range.

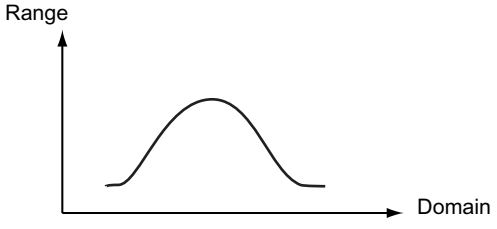


Figure 4-883 Application of cam/technological profiles

A defined start is also possible within a profile.

Application:

Specific adaptation and optimization of traversing specifications.

Additional information about cams can also be found in the SIMOTION CamTool Manual and in the SIMOTION SCOUT online help.

Table 4-364 Possible assignments

Designation	Definition range	Value range	Command
Position-related velocity profile	Position 1) 2)	Velocity specification	_runPositionLockedVelocityProfile()1) _runMotionInPositionLockedVelocityProfile() 2)
Time-related velocity profile	Time	Velocity specification	_runTimeLockedVelocityProfile()
Time-related position profile	Time	Position	_runTimeLockedPositionProfile()
Time-related force/pressure profile	Time	Force/pressure specification	_runTimeLockedForceProfile()
Position-related force/pressure profile	Position 1) 2)	Force/pressure specification	_runPositionLockedForceProfile() 1) _runMotionInPositionLockedForceProfile()2)
Position-related force/pressure limiting profile	Position 1) 2)	Force/pressure limiting	_enablePositionLockedForceLimitingProfile() 1) _enableMotionInPositionLockedForceLimitingProfile() 2)
Time-related force/pressure limiting profile	Time	Force/pressure limiting	_runTimeLockedForceLimitingProfile()
Time-related velocity limiting profile	Time	Velocity limitation	_runTimeLockedVelocityLimitingProfile()
Position-related velocity limiting profile	Position 1) 2)	Velocity limitation	_enablePositionLockedVelocityLimitingProfile() 1) _enableMotionInPositionLockedVelocityLimitingProfile() 2)

1) Actual position of axis – the axis position is assigned to the definition range.

2) Interconnected position – the actual position present at the MotionIn interface is assigned to the definition range.

Profile reference

Time-related profiles

The domain of the cam to be used in the command is interpreted as time in the time unit of the axis. The profile can be executed from start to finish or, alternatively, from a starting point that can be predefined.

Position-related profiles with respect to the axis' own position

The domain of the cam to be used in the command is interpreted as the position in the position unit of the axis.

The profile reference is equivalent to the absolute axis position. The profile is started from the current axis position.

The velocity traversing profiles are related to the resulting position setpoint.

The velocity limiting profiles and the force/pressure profiles are related to the actual position value.

Position-related profiles with respect to interconnected position (V3.2 and higher)

The profile is related to the position interconnected via MotionIn connectors.

Table 4-365 Possible commands for reference to a position interconnected via MotionIn

Command	Description
_runMotionInPositionLockedForceProfile()	Force profile related to an interconnected position (MotionIn)
_runMotionInPositionLockedVelocityProfile()	Velocity profile related to an interconnected position (MotionIn)
_enableMotionInPositionLockedForceLimitingProfile()	Force limiting profile related to an interconnected position (MotionIn)
_enableMotionInPositionLockedVelocityLimitingProfile()	Velocity limiting profile related to an interconnected position (MotionIn)

Separate system variables are available for status queries, e.g.:

- **forceMotionInPositionProfileCommand** for force/pressure profiles
- **velocityMotionInPositionProfileCommand** for velocity profiles

Profile types

Velocity profile/velocity limiting profile

The range of the cam to be used in the command is interpreted as the velocity in the velocity unit of the axis. The motion direction, acceleration, and jerk are calculated accordingly.

A transition profile is generated by the axis for discontinuous transitions. The dynamic response parameters of this profile are specified using the command parameters for acceleration and jerk.

Position profile

The range of the cam to be used in the command is interpreted as the position in the position unit of the axis. The motion direction, position, acceleration, and jerk are calculated from this relationship.

The position reference to the axis can be selected as either absolute or relative.

A transition profile is generated by the axis for discontinuous transitions. The dynamic response parameters of this profile are specified using the command profile parameters for acceleration, jerk, and velocity.

Force/pressure profile and force/pressure limiting profile

The range of the cam to be used in the command is interpreted as the force/pressure in the pressure unit of the axis. The derivation of force/pressure for any necessary transition motions (for entering the profile and exiting the profile, for example), can be programmed in the command. The behavior at the end of the profile is set during axis configuration.

Limitation of the acceleration and braking ramps (V3.2 and higher)

The **DecodingConfig.profileDynamicsLimiting** configuration data element is used to determine whether the permitted acceleration and braking ramps should be limited by the programmed values (**derivedCommandValue**) or the maximum values (configuration data).

- Limiting by programmed values:
  The axis is ramped up to the profile or to the values specified in the motion vector (MotionIn) using the minimum value from the dynamic values defined in the command and the maximum values set in MaxJerk, MaxAcceleration, and MaxVelocity.
  While traversing on the profile, the axis is limited to the lowest value of the programmed values and the set maximum values.
- Limiting by maximum values:
  The axis is ramped up to the profile or to the values specified in the motion vector (MotionIn) using the minimum value from the dynamic values defined in the command and the maximum values set in MaxJerk, MaxAcceleration, and MaxVelocity.
  While traversing on the profile, the axis is only limited to the set maximum values.

Status display during profile processing (V3.2 and higher)

During profile processing, extended states are displayed in the system variables for the profiles, e.g. in **positionTimeProfileCommand.processingState**.

The PROFILE_END status is set if the end of profile has already been reached, but the command is still active.

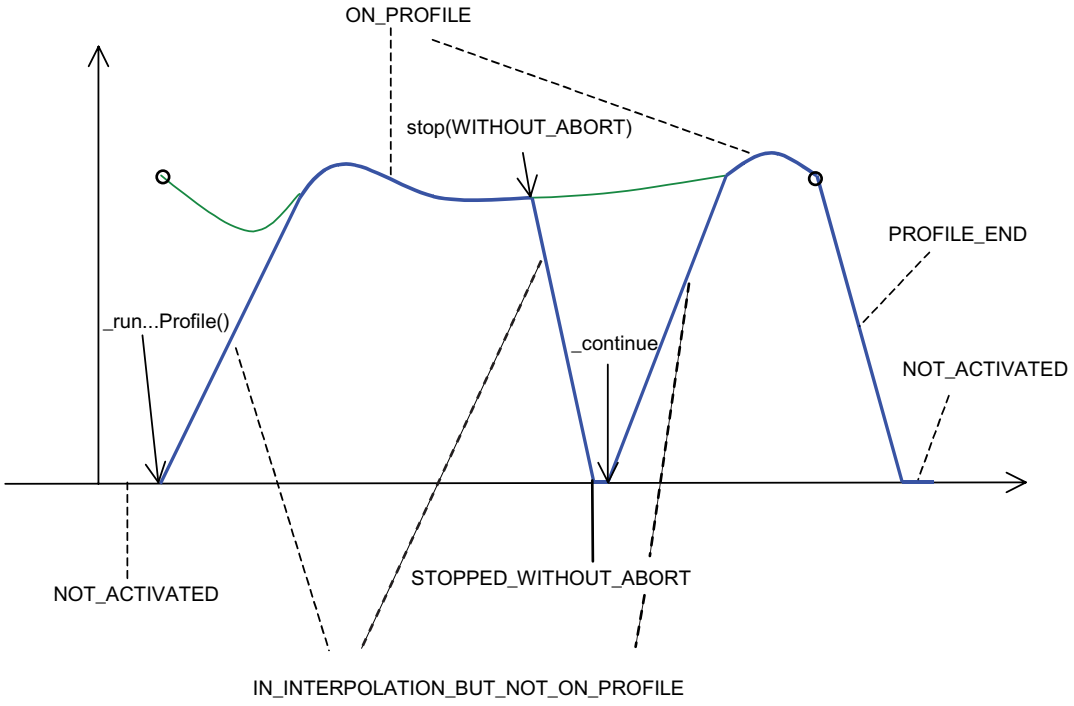


Figure 4-884 Status displays when traveling through a profile (example)

See also

Positioning with user-definable position profile (Page 3161)

Enable force/pressure limiting with position-related force/pressure limiting profile (Page 3163)

Enabling force/pressure limiting with time-related motion profile (Page 3163)

Starting the time-related force/pressure profile (Page 3164)

Starting the position-related force/pressure profile (Page 3165)

Starting a time-related velocity profile (Page 3160)

Starting a position-related velocity profile (Page 3160)

Enabling velocity limiting with position-related velocity limiting profile (Page 3167)

Enabling velocity limiting with time-related velocity limiting profile (Page 3168)

Behavior at the end of the profile (V3.2 and higher)

The **decodingConfig.behaviourAtTheEndOfProfile** configuration data element is used to configure various behaviors.

Table 4-366 Assignable behaviors

Setting	Meaning
MOVE_WITH_CONSTANT_SPEED	Constant continuous motion • Position/velocity: Hold value • Force/pressure: Hold value
STOP_IN_PROFILE_END	Decelerate to target point/velocity 0 • Position/velocity: Velocity 0 at the end of profile (via ramp) • Force/pressure: Value at the profile end point (= Hold value)
STOP_WHEN_PROFILE_END_REACHED	Decelerate after completed traverse • Position/velocity: Velocity 0 when end of profile reached (via ramp) • Force/pressure: Value at the profile end point (= Hold value)

Changes can be made online and take effect immediately.

Note

For position-related profiles with the setting STOP_IN_PROFILE_END, it should be remembered that the velocity calculated by the system can cause vibration.

4.8.3.24 Motion commands

Motion execution/interpolator

Axis motion is executed in the interpolator, in the servo, and in Servo_fast.

The servo for all axes is calculated in the position-control cycle clock.

The reference variables are calculated in the interpolator. The interpolator cycle clock of the device is set during the configuration of the execution system. There are two interpolator levels in the system, **IPO** and **IPO2**.

The processing cycle clock (axis-specific interpolator cycle clock) of the Axis technology object can be set to **IPO** or **IPO2**.
For the possible setting **IPO_fast**, see the chapter titled *Second position control cycle clock (Servo_fast)* in the *Motion Control Basic Functions* manual.

This makes it possible to place an interpolator for axes that do not require a high time resolution to calculate reference variables in a cyclical system task with a longer cycle time, thereby requiring less processing power.

Setting for shorter axis reaction times (e.g. faster start of motion) (as of V4.1 SP1)

The **Execution.executionLevel=SERVO** or **processing clock** = Servo setting in the configuration screen form can be used to execute the IPO system component of the axis in the servo following actual value measurement. After the actual value system in the servo, the IPO system functionality is calculated first, followed by the controller and the setpoint system.

This reduces the response time to the switching of external signals or synchronous operation in the control to one servo cycle clock.

Note

Due to the higher performance requirement, this setting should only be used for selected (i.e. a limited number of) axes.

Temporarily high IPO system components in the servo of the axis can result in a level overflow in the servo, thus causing the CPU to go to STOP mode.

For the possible setting **Servo_fast**, see the chapter titled *Second position control cycle clock (Servo_fast)* in the *Motion Control Basic Functions* manual.

System and user tasks

In addition to the technology object system tasks of ServoTask, IPOTask, and IPOTask_2, there are also synchronous user tasks called ServoSynchronousTask, IPOSynchronousTask, and IPOSynchronousTask_2.

How the system and user tasks interact with each other is demonstrated in the following example for the Axis technology object. At the end of cyclic data transfer, the ServoSynchronousTask and ServoTask are started, followed by the IPOSynchronousTask and IPOTask. Axis data which is active in the servo (e.g. superimposed setpoint or manipulated variable value) can be modified in the ServoSynchronousTask. In the IPOSynchronousTask, issued motion commands are processed directly in the subsequent IPOTask (e.g. switchover to superimposed motion or registration mark).

Because the ServoSynchronousTask is executed before the IPO system component in the servo level even if **execution.executionLevel=SERVO** is set, this enables a very rapid response with motion affected at user level as well.

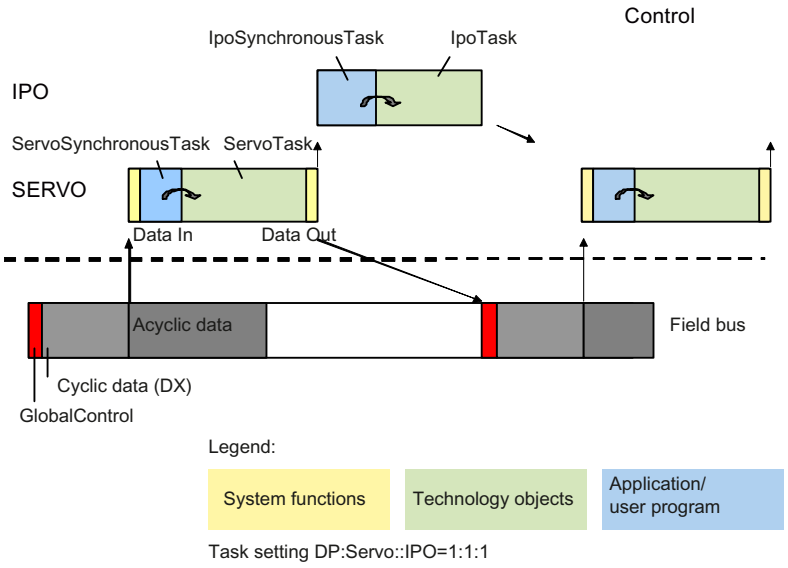


Figure 4-885 Example of cycle clock settings

See also the chapters titled *Execution system/Tasks/System cycle clocks* and *Dynamic response to data processing in the control* in the *Motion Control Basic Functions* Function Manual.

Command groups

Commands on the axis are divided into command groups to enable multiple commands to be active in one **IPO** cycle clock. Commands on the axis are read in and processed in the IPO cycle clock. If the user program issues more than one command for a command group within an interpolator cycle clock and the user program can run, for example, in another task, then a specific behavior is defined for the command group. The commands assigned to a command group are implicitly assigned to the corresponding command buffer.

The overwritten commands trigger technological alarm "030002 Command aborted".

Command buffers and their properties

Every axis has command buffers that can buffer one command each. These buffers are read out in every interpolator cycle clock.

- Buffer[1)] for **_stopEmergency()**, **_stop()**, and **_continue()** commands
  This buffer contains **_stopEmergency()**, **_stop()** without command abort, and **_continue()**. The procedure followed when motions stop depends on the command priorities. Higher priorities are indicated by larger numbers. A command with the same or higher priority displaces a command already in the buffer.
  - **Priority 1: _stop()** without command abort and **_continue()**
  - **Priority 2: _stopEmergency()** with time ramp
  - **Priority 3: _stopEmergency()** with maximum deceleration
  - **Priority 4: _stopEmergency()** with setpoint zero
- Buffer[2)] for **Enable**- and **Disable** commands
  This buffer contains the enable and disable commands. The commands displace each other from the command buffer.
- Buffer[3)] for **superimposed** commands
  Commands that are executed in parallel or superimposed to a main motion are, for example:
  - Motion commands with mergeMode=SUPERIMPOSED_MOTION_MERGE
  - **_redefinePosition()**
  - **_enableAxisAdditiveTorque()**
  - **_homing()** with homingMode=DIRECT_HOMING or PASSIVE_HOMING

  The following applies to SIMOTION < V4.4:
  The buffer has an entry. These commands overwrite each other if they are issued within one IPO cycle clock.
  As of SIMOTION V4.4, the following applies:
  A maximum of 10 command entries are possible in the command buffer for superimposed commands. The number can be set in
  the **DecodingConfig.lenghtOfBufferForSuperimposedCommands** configuration data element. When a new axis is created, five entries are preset by the system.
- Buffer[4)] for **overriding** and **sequential** commands
  Acceptance of all commands, in particular motion commands that have been programmed as sequential motion using the mergeMode=SEQUENTIAL function parameter or as overriding motion using the mergeMode=IMMEDIATELY function parameter.
  In the case of sequential motions with immediate command advance (nextCommand=IMMEDIATELY and mergeMode=SEQUENTIAL), new commands to be entered can return with errors if the buffer is full.
  In the case of sequential motions with immediate command advance, if the motion command can be accepted by the system (nextCommand=WHEN_BUFFER_READY and mergeMode=SEQUENTIAL), the command does not advance until the motion command has been accepted by the system.
  In the case of mergeMode=NEXT_MOTION or mergeMode=IMMEDIATELY, the command can always be accepted by the system, because the next motion command in the command buffer or the current motion command in the interpolator is overwritten.

The following table shows the assignment of axis commands to command buffers.

Table 4-367 Allocation of commands to the command buffers

Command	Position
_continue() Resume motion	1
_stop() Stop a motion without traversing command abort	1
_stop() Stop a motion with traversing command abort	4
_stopEmergency() Stop the motion with cancellation of the motion commands without further motion commands having any affect	1
_disableAxis() Remove axis enable	2
_disableQFAxis() Remove enable for axis with hydraulic functionality	2
_enableAxis() Set axis enable	2
_enableQFAxis() Set enable for axis with hydraulic functionality	2
_adaptAxisConfigData() Adapts the characteristic data for motors and/or encoders	3
_disableAxisAdditiveTorque() Deactivate additive set torque	3
_disableAxisSimulation() Disable simulation mode	3
_disableForceLimiting() Deactivate force/pressure limiting	3
_disableMovingToEndStop() Disabling command for travel to fixed endstop	3
_disableTorqueLimiting() Deactivate torque limiting	3
_disableAxisInterface() Deactivate actuator interface	3
_disableAxisTorqueLimitNegative() Activate negative torque limiting	3
_disableAxisTorqueLimitPositive() Activate positive torque limiting	3
_disableMonitoringOfEncoder() Deactivate differential encoder monitoring	3
_disableVelocityLimiting() Deactivate velocity limiting	3

Command	Position
_enableAxisAdditiveTorque() Activate additive set torque	3
_enableAxisInterface() Activate actuator interface	3
_enableAxisSimulation() Enable simulation mode	3
_enableAxisTorqueLimitNegative() Activate negative torque limiting	3
_enableAxisTorqueLimitPositive() Activate positive torque limiting	3
_enableForceControlByCondition() Switch over to p(t) profile with switchover criterion	3
_enableForceLimitingByCondition() Switchover to force/pressure limiting when the switchover criterion defined in the command is reached	3
_enableForceLimitingValue() Activate force/pressure limiting with constant force/pressure limit value	3
_enableMonitoringOfEncoder() Activate differential encoder monitoring	3
_enableMotionInPositionLockedForceLimitingProfile() Force limiting profile related to an interconnected position (MotionIn)	3
_enableMotionInPositionLockedVelocityLimitingProfile() Velocity limiting profile related to an interconnected position (MotionIn)	3
_enableMovingToEndStop() Enabling command for travel to fixed endstop	3
_enablePositionLockedForceLimitingProfile() Activate force/pressure limiting with p(s) profile	3
_enablePositionLockedVelocityLimitingProfile() Activate velocity limiting with position-related limiting profile	3
_enableTimeLockedForceLimitingProfile() Activate force/pressure limiting with time-related limiting profile	3
_enableTimeLockedVelocityLimitingProfile() Activate velocity limiting with time-related limiting profile	3
_enableTorqueLimiting() Activate torque limiting	3
_enableVelocityLimitingValue() Activate velocity limiting	3
_redefinePosition() Reset actual and set position	3
_setAndGetEncoderValue() Synchronize measuring systems	3
_setAxisDataSetActive() Switch over data set	3

Command	Position
_homing() Homing	3 4 homingMode = ACTIVE_HOMING
_move() Rotate	4 3 for mergeMode = SUPERIMPOSED
_pos() Positioning	4 3 for mergeMode = SUPERIMPOSED
_runMotionInPositionLockedForceProfile() Force profile related to an interconnected position (MotionIn)	4 3 for mergeMode = SUPERIMPOSED
_runMotionInPositionLockedVelocityProfile() Velocity profile related to an interconnected position (MotionIn)	4 3 for mergeMode = SUPERIMPOSED
_runPositionBasedMotionIn() Activate MotionIn interface with position reference	4 3 for mergeMode = SUPERIMPOSED
_runPositionLockedForceProfile() Run a p(s) profile	4
_runPositionLockedVelocityProfile() Apply a v(s) profile	4 3 for mergeMode = SUPERIMPOSED
_runTimeLockedForceProfile() Run a p(t) profile	4
_runTimeLockedPositionProfile() Apply an s(t) profile	4 3 for mergeMode = SUPERIMPOSED
_runTimeLockedVelocity() Apply a v(t) profile	4 3 for mergeMode = SUPERIMPOSED
_runVelocityBasedMotionIn() MotionIn interface with velocity reference activated	4 3 for mergeMode = SUPERIMPOSED
_setForceCommandValue() Set force/pressure setpoint	4
_bufferAxisCommandId() Store command status permanently	5
_cancelAxisCommand() Abort command with the specified CommandID	5
_getAxisDataSetParameter() Read a data set	5
_getAxisErrorNumberState() Read out error number status	5

Command	Position
_getAxisErrorState() Read out error status, alarm number and alarm parameter of up to eight pending alarms	5
_getAxisInternalPosition() Convert from axis coordinates to encoder position values	5
_getAxisStoppingData() Calculate braking distance depending on a specified actual velocity, actual acceleration, traversing profile and dynamic response parameters	5
_getAxisUserPosition() Convert from encoder positions to axis coordinate system	5
_getForceControlDataSetParameter() Read a data set	5
_getMotionStateOfAxisCommand() Read out motion phase of a command	5
_getProgrammedTargetPosition() Display programmed absolute end position including superimposition	5
_getStateOfAxisCommand() Read out command status	5
_getStateOfMotionBuffer() Shows whether the command queue can be filled	5
_getQFAxisDataSetParameter() Read specific parameters for the axis with hydraulic functionality in the data set	5
_removeBufferedAxisCommandId() End permanent storage of the command status	5
_resetAxis() Reset axis	5
_resetAxisConfigDataBuffer() Clear the configuration data in the buffer without activation	5
_resetAxisError() Reset error on the axis	5
_resetMotionBuffer() Reset command queue	5
_setAxisDataSetParameter() Overwrite data set	5
_setAxisSTW() Specify control bits in STW1 directly	5
_setForceControlDataSetParameter() Overwrite data set	5
_setQFAxisDataSetParameter() Overwrite the specific parameters for the axis with hydraulic functionality in an inactive data set	5

Command	Position
_setQFAxisPCharacteristics() Set characteristics for P value	5
_setQFAxisQCharacteristics() Set characteristics for Q value	5

1) Buffer for emergency stop and stop-continue commands

2) Buffer for enable and disable commands

3) Buffer for superimposed commands

4) Buffer for overriding and sequential commands

5) Not allocated to the command buffers; commands are executed synchronously as called

Changing motion commands into the interpolator

A **DecodingConfig.decodeSequentialMotionCommand** configuration data item can be used to set when the next **SEQUENTIAL** or **NEXT_MOTION** command is switched in or executed in the interpolator - immediately in the same cycle clock or in the next **IPO** cycle clock.

- If the setting is **IMMEDIATELY** (default), the next command is switched in immediately and started in the same cycle clock once interpolation/execution of the current command in the IPO cycle clock is complete.
- If the setting is **NEXT_IPO_CYCLE**, the next command is not changed into the interpolator until the previous command has been completely interpolated.
  This prevents execution of multiple motion commands in the interpolator, thus preventing a greater-than-average interpolator load from being called forth in one cycle clock.

Motion transitions

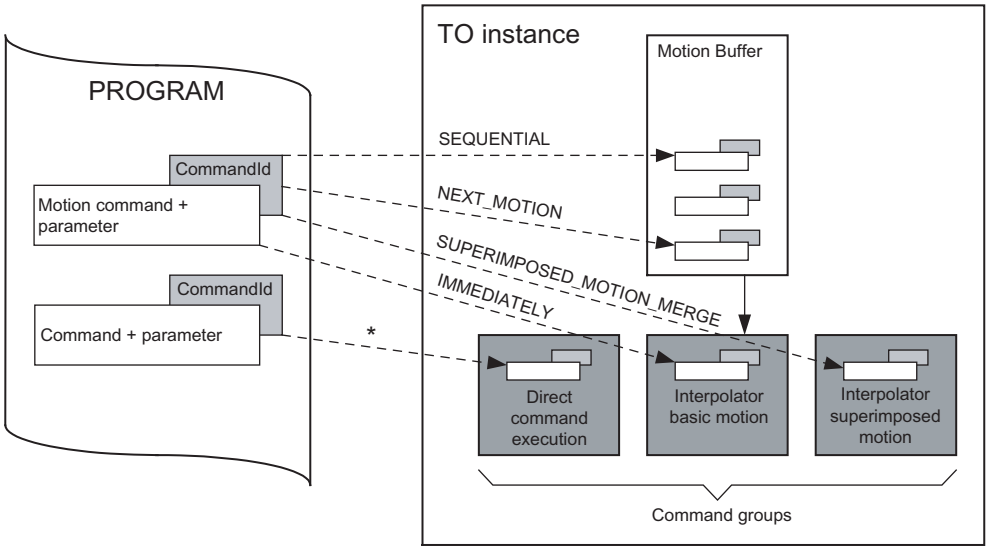
The behavior during a transition between two motions is defined by **mergeMode**.

The programmed motion transitions are decisive for the active motion. The priority of the task in which the motion command is issued does not affect the priority assignment of the command.

Definable motion transitions

- **IMMEDIATELY** (substitute)
  The motion specified with the command is activated immediately. Motions which are already active are overridden and pending commands/motions are aborted.
- **NEXT_MOTION** (attach, delete pending command)
  Execute after the active motion and delete further pending commands/motions.

- **SEQUENTIAL** (attach)
  Append to previous commands/motions.
- **SUPERIMPOSED_MOTION_MERGE** (superimpose)
  Superimposed motion on the axis is possible in addition to the basic motion



* Not allocated to the command buffers; commands are executed synchronously as called
(See also allocation 5) in the table titled "Allocation of commands to the command buffers)

Figure 4-886 Command reactions in the Axis technology object

Conditions for command advance

If the condition for the command advance is satisfied, the next command in the user program is executed. Specifying a step enabling condition affects the **execution time** of the next command in the same user task.

- **IMMEDIATELY**
  As soon as the command has been issued, irrespective of whether or not the commanded motion can be executed
- **WHEN_BUFFER_READY**
  After the command has been entered in the motion buffer
- **AT_MOTION_START** (motion start)
  After the command has been changed into the interpolator
- **WHEN_ACCELERATION_DONE** (end of acceleration)
  When the acceleration phase is completed
- **AT_DECELERATION_START** (start of deceleration phase)
  When the deceleration phase starts
- **WHEN_INTERPOLATION_DONE** (end of setpoint interpolation)
  When the setpoint interpolation for this command is completed
- **WHEN_MOTION_DONE** (positioning window reached)
  When the setpoint interpolation is complete and the configured positioning window has been reached

- **WHEN_COMMAND_DONE** (when command is completed or aborted)
  After completion of the command, for example, for commands that require time for their execution, but do not contain a motion element
- **WHEN_TORQUELIMIT_REACHED** (as soon as the torque is limited)
  When torque limiting is triggered
- **WHEN_TORQUELIMIT_GONE** (as soon as the torque limiting is switched off)
  When torque limiting is removed
- **WHEN_LIMITING_COMMAND_ACTIVATED** (when the velocity limiting command is activated)
  When velocity limiting is activated
- **WHEN_LIMIT_REACHED** (as soon as the velocity is limited)
  When velocity limiting is triggered
- **AT_PROFILE_START**
  Beginning of interpolation with profile
- **BY_PROFILE_END**
  End of setpoint interpolation with profile
- **WHEN_AXIS_HOMED** (when axis is homed)
  Axis was homed
- **WHEN_ENDSTOP_REACHED** (when clamping value is reached)
  Clamping value reached (travel to fixed endstop)
- **WHEN_FUNCTION_DISABLED** (when command is aborted or completed)
  Command is completed or aborted (travel to fixed endstop)

State model/axis status

Table 4-368 The axis status is indicated in:

• Axis inactive/can be activated:	control= INACTIVE
• Axis active:	control= ACTIVE
• Motion:	motionStateData.motionCommand= IN_MOTION
• Fault:	error= YES and ErrorReaction <> NONE
• StopEmergency:	stopEmergencyCommand.state= ACTIVE

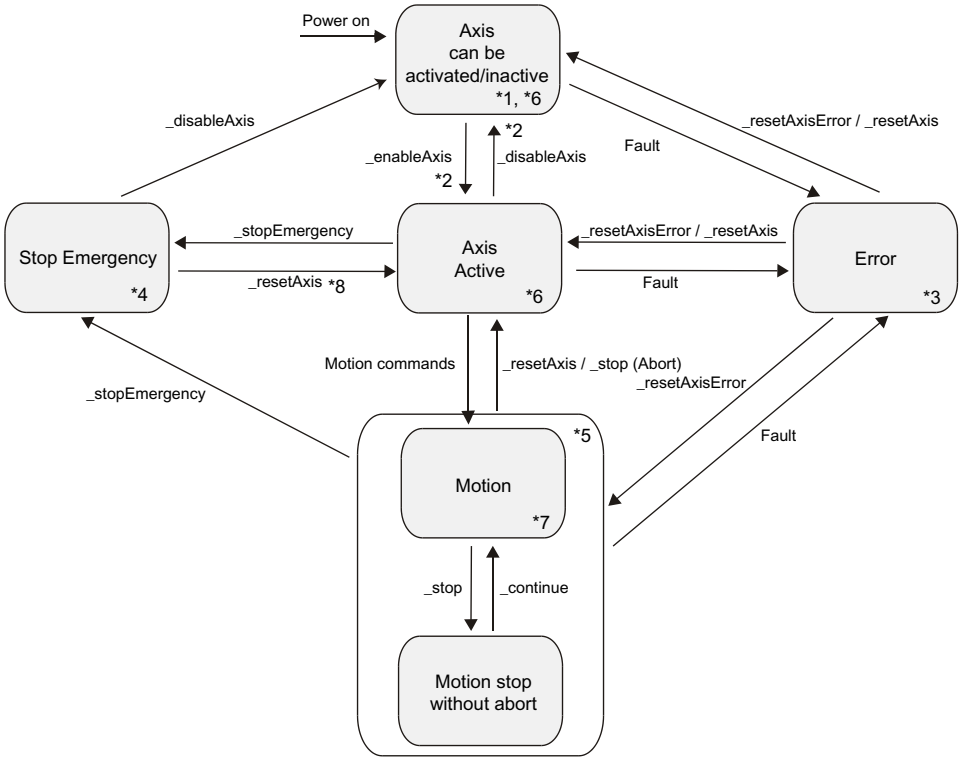


Figure 4-887 State model/axis status

When the axis is configured as an axis with hydraulic functionality (QFAxis) in **TypeOfAxis**, **_enableAxis()** must be replaced with **_enableQFAxis()** and **_disableAxis()** must be replaced with **_disableQFAxis()** in the state model for the axis.

The status of **_stopEmergency()** is displayed in the **stopEmergencyCommand** system variable. Following a reset, this variable must be scanned explicitly by means of **_disableAxis()**.

Table 4-369 The following commands and functions are active in the individual states of the axis:

***1** Axis can be enabled/is inactive with respect to motion control of the axis; also includes the disabled state

***2** For active drive, output and control:

- Switch from follow-up mode or to follow-up mode
- Enable/disable the control

***3** Depending on the stop or error states, **_disableAxis()** / **_disableQFAxis()** and **_stopEmergency()** are permitted

***4** Faults are handled according to existing criteria. Higher-priority local stop reactions can be handled.

***5** Superimposed motions and motions on the following axis are included.

***6** **resetAxis()** permitted

***7** Motion commands permitted

***8** If axis is in position control

Axis in inactive/can be activated state

When the control is switched on, the technology object goes into follow-up mode. In that case:

- All commands in the motion buffer are deleted
- The axis and controller are inactive
- System variables are set either to the configured values or to start values
- The motions commanded by motion commands are not executed; the axis is in follow-up mode
- The **_enableAxis()** or **_enableQFAxis()** command enables the controller and switches the technology object state to active
- Changing from STOP U operating state to RUN, and vice versa, has no effect on the Axis technology object
- During the transition from STOP to STOP U operating state, all alarms that can be reset are acknowledged and the motion buffer is cleared
- Encoders/actual value system is/are active in the STOP operating state
  The axis position is retained, unless an alarm is triggered.

Axis in active state

Motion commands can be issued and executed in this state.

Axis in motion state

- Motion jobs are executed and motion commands can be issued in this state.
- Motion can be stopped by sending the **_stop()** command, stopMode= STOP_WITHOUT_ABORT setting, and resumed by sending the **_continue()** command
- Motion is aborted with the **_stop()** command, stopMode= STOP_AND_ABORT

Axis in fault technology object state

In fault technology object state, the following actions are possible:

- Actions that reset the fault state
- Actions that cause a higher-priority stop reaction
- Actions that do not affect the state
- Actions that are generally permitted according to criteria for technological alarms

Motion jobs and the simulation command **_enableAxisSimulation()** are not executed.

The fault is remedied with the **_resetAxis()** or **_resetAxisError()** command if these commands are permitted for the fault in question.

The status commands **_getStateOfAxisCommand()** and **_getStateOfMotionBuffer()** are permitted, as are the reset commands **_resetMotionBuffer()** and **_disableAxis()** or **_disableQFAxis()**.

Aborted motion commands trigger a **technological alarm** and a notice in the alarm window.

The reactions on the axis can be configured in the technological alarm.

Note

Errors can be acknowledged in SIMOTION SCOUT, the program or the operating panel.

You can find additional information under Error handling in the *Motion Control Basic Functions* Function Manual.

Axis in StopEmergency state

The **_stopEmergency()** command

- Does *not* cause the alarm task to start immediately
- Does *not* disable the axis and drive
- Is *not* active if the axis is in follow-up mode
- Is *not immediately* active if a following error has built up during active torque limitation
- Is *not immediately* active if a pressure axis switches from closed-loop pressure control to pressure limitation

A **_stopEmergency()** command with a higher-priority stop reaction cancels a lower-priority reaction.

The **_stopEmergency()** command generates the **_stopEmergency_Status**. This status can be canceled with the **_disableAxis()** / **_disableQFAxis()** or **_resetAxis()** commands.

Commands active in the interpolator are aborted.

4.8.3.25 Data exchange between Axis technology object and DCC

For data exchange between the Axis technology object and DCC charts/DCC blocks, system variables of the technology object can be interconnected directly with block inputs and block outputs.

The update/effectiveness of the system variables is specified in the reference lists.

- Cyclically updated display data in the IPO include:
  - **motionStateData.actualVelocity**
  - **motionStateData.actualAcceleration**
  - **positioningState.actualPosition**
- Cyclically effective system variables in the IPO include:
  - **defaultMotionIn.x** (when activated by means of a command)
  - **defaultMotionIn.y** (when activated by means of a command)
  - **defaultMotionIn.z** (when activated by means of a command)
  - **override.velocity**
  - **override.acceleration**
  - **plusLimitsOfDynamics.velocity**
  - **plusLimitsOfDynamics.postiveAccel**
  - **plusLimitsOfDynamics.negativeAccel**
- Cyclically updated display data in the servo includes:
  - **sensorData[n].position**
  - **sensorData[n].velocity**
  - **sensorData[n].acceleration**
  - **sensorData[n].incrementalPosition**
- Cyclically effective system variables in the servo include:
  - **servoSettings.additionalCommandValue**
  - **servoSettings.additionalSetpointValue**

You can find additional information about DCC in the *Motion Control Basic Functions* Manual.

4.8.3.26 Drive communication based on DPV1 services

Drive communication based on PROFIdrive provides the basis for the digital SIMOTION/ SINAMICS drive system.
The focus is primarily on cyclic services. Acyclic DPV1 services also provide the option of on-demand data transfer, e.g. to set various parameters or options for slave devices during operation.

The LDPV1 library provides the basis for the quick and easy use of DPV1 services in applications.

Functions supported:

- Buffer management
- Time-of-day synchronization SIMOTION-SINAMICS
- Ramp-up coordination
- Assignment of SIMOTION technology objects to SINAMICS objects

- Reading of SINAMICS errors and warnings
- Activation and deactivation of technology objects and drive objects
- Reading and writing of SINAMICS parameters
- Functions for SINAMICS Safety Integrated

You can find the LDPV1 library in the *SIMOTION Utilities & Applications* included in the scope of delivery of SIMOTION SCOUT (under **Applications > Cross-sector applications > Drive communication**).

Each individual function supported is accompanied by documentation describing the actual application.

Note

As of SIMOTION firmware V4.2, time-of-day synchronization between SIMOTION and SINAMICS is automatic.

4.8.3.27 Links/interconnections to other technology objects

If you interconnect technology objects with one another, they must be configured in the same execution level - with the exception of synchronous axes.

If you need to connect a technology object in a different execution level with an axis, you can work with a virtual synchronous axis that is in the desired execution level and is connected to the actual axis. You can generally use this to evaluate cyclic data of an axis in other levels also (e.g. from the user program).

Note

Further information is available under Interconnection of technology objects in the Motion Control Basic Functions Function Manual.

4.8.4 Configuring an axis

4.8.4.1 Overview of axis configuration

Axes are configured using an axis wizard,
which is started automatically when a new axis is created. In order to change the axis settings, the configuration can be launched using *Configure displayed data set...* in the *Configuration* axis parameter assignment dialog box. Important axis parameters and drive connections are configured in the axis wizard.

Data sets are also managed in the *Configuration* axis parameter assignment dialog for the purpose of data set changeover and encoder configuration.

For more information, see Data sets (Page 3037).

You can specify other selected parameters via the axis parameter assignment dialog boxes, which can be accessed under Axis Object in the project navigator. You can also use the expert list to access all configuration data and system variables relating to the Axis technology object in list format.

System default settings of the configuration data

The system default information shows the settings of the configuration data, which are backward compatible (can also be used for older versions). These are documented in the reference lists for the configuration data.
When a technology object is created or when drives are assigned, the system may make other technological settings than the system defaults in order to support system optimizations or functions of the new versions for the respective context.

4.8.4.2 Linking digital drives

Overview of linking digital drives

The Assign dialog for the drive link is included in the axis wizard when setting up an axis. Once an axis has been set up, this dialog can also be called via the **...** button in the **Configuration** axis dialog or from the address list (**All addresses** view). Drives can also be assigned symbolically, see Coupling of digital drives (Page 2895).

Digital drives are linked to the SIMOTION Axis technology object via a PROFIdrive message frame.

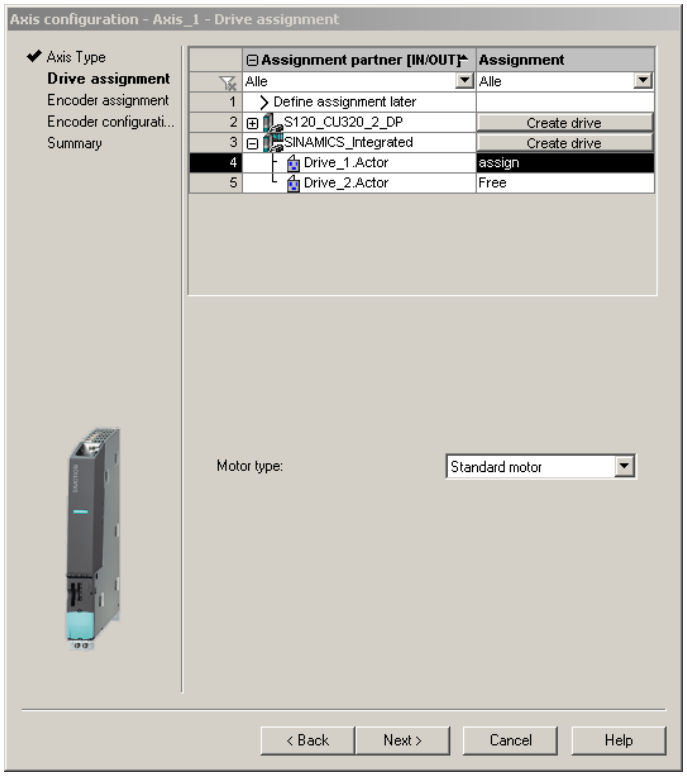


Figure 4-888 SINAMICS drive assignment

The dialog for drive assignment supports both immediate and retrospective drive assignment. Furthermore, a new drive can be created and assigned to the axis. See also Coupling of digital drives (Page 2895).

Configuring PROFIBUS DP in HW Config to optimize run-time

The configuration of PROFIBUS DP in HW Config to optimize run-time is described in the **Motion Control Basic Functions** manual under *Configuring PROFIBUS DP in HW Config to optimize run-time*.

4.8.4.3 Linking analog drives to SIMOTION

Connection of any drive with an analog setpoint interface to a SIMOTION device (e.g. C2xx).

Axis - analog drive relationship

For an axis with an analog drive, the speed setpoint is applied to the analog output as a voltage signal, and the actual value is read in via an encoder interface.

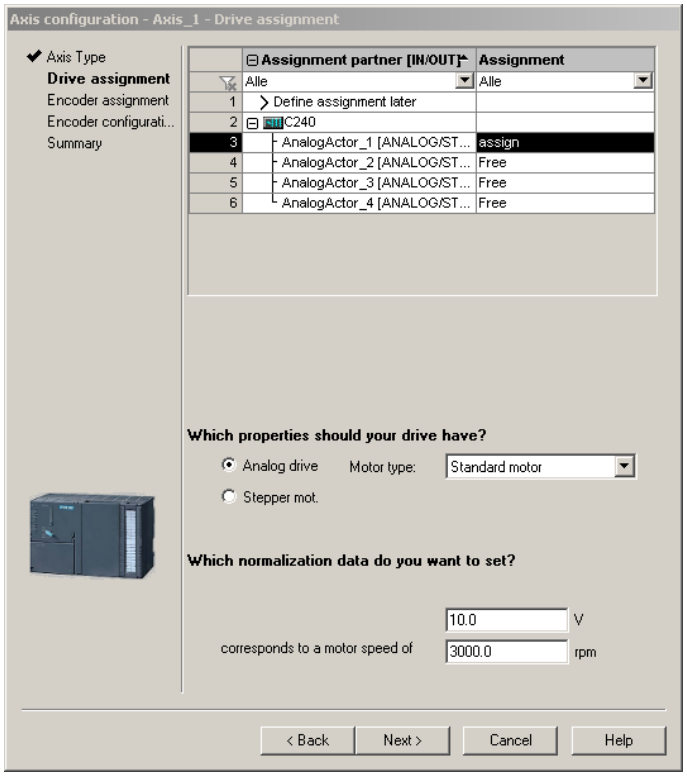


Figure 4-889 Drive assignment of analog drives

Drive

A drive is controlled with +/- 10 V via an **analog interface**.

Axis-drive connection

An axis is connected to a drive by entering the **logical address of the selected analog output** of the SIMOTION device when the axis is configured. Use the axis configuration wizard to do this.

If you change the hardware configuration in **HW Config** at a later time, the specified logical addresses may become incorrect. In this case, repeat the configuration in the hardware configuration wizard.

Make sure that the Volt >> Speed setting in the drive matches the setting in the axis.

ADI4/IM174

In addition to the option of operating analog axes on the onboard inputs of the C2xx, the ADI4 and IM 174 modules are available for use in all platforms as interfaces for analog drive connections. From the SIMOTION perspective, these modules behave like digital drive links. See also Coupling of digital drives (Page 2895) for more information.

You can find further information in the *ADI4 - Analog drive interface for 4 axes* and *Distributed I/O, PROFIBUS module IM174* manuals.

See also

Coupling of digital drives (Page 2895)

Setting as a real axis with analog drive link (Page 2877)

4.8.4.4 Axis with stepper motor connection

Connection of a stepper drive with a pulse/direction interface to a SIMOTION device (e.g. C2xx).

For an axis with stepper motor connection, there is one clock pulse, directional signal, and enable signal output per axis.

For a link via a bus system (e.g., to IM174 via PROFIBUS DP), the general information for configuring PROFIdrive drives is applicable, with special consideration of the behavior of a stepper motor (see Stepper drives (Page 2982)).

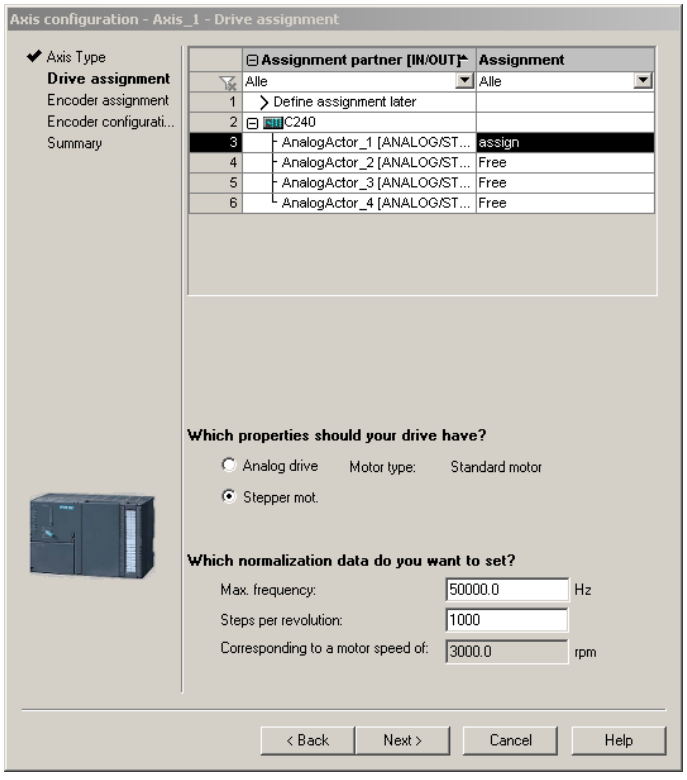


Figure 4-890 Drive assignment of the stepper motor

An axis with stepper motor connection can be moved either with or without an encoder. With an encoder, the actual value signals are read in via the encoder interface; without an encoder, the actual value information is generated from the output motor pulses.

Note

The product of the maximum frequency of the stepper motor and the reciprocal of the number of steps per motor revolution gives the maximum speed available to the control loop (corresponds to maxSpeed for conventional drives).

Note

When an encoderless stepper motor connection is configured on a position axis, an encoder input is still automatically reserved and cannot be used for an external encoder, for example.

With a drive axis, the encoder interface is available for and can be assigned to an Axis or External Encoder technology object.

For operation without an encoder, additional data can be specified in the configuration data for rotation monitoring.

- **NumberOfEncoders.Encoder_1.stepMotorMonitoring.enable** for enabling/disabling the monitoring function
- **NumberOfEncoders.Encoder_1.stepMotorMonitoring.beroCycleDistance** for setting the permitted deviation in terms of motor steps per revolution
- **NumberOfEncoders.Encoder_1.stepMotorMonitoring.beroCycleTolerance** for setting a tolerance range around the beroCycleDistance

Rotation monitoring can be activated for operation without an encoder:

An external zero mark is used and is connected to the associated external zero mark input of the axis channel.

For a linear axis, the external zero mark must monitor the rotation of the motor shaft.

If a signal is not received within the specified motor steps + tolerance, a technological alarm is issued.

IM174/PROFIBUS drives

In addition to the option of operating stepper drives on the onboard inputs of the C2xx, the IM174 module is available for use in all platforms as an interface for stepper drives. From the SIMOTION perspective, stepper drives linked via IM174 behave like digital drive links.

Alternatively, stepper drives can be linked with a PROFIBUS interface if they support the PROFIdrive profile. See Coupling of digital drives (Page 2895).

You can find more information in the *Distributed I/O IM 174 PROFIBUS Module Manual*.

See also

Position control (Page 2950)

Setting as a real axis with stepper drive C2xx (V3.2 and higher) (Page 2897)

Stepper drives (Page 2982)

4.8.4.5 Using the expert list for an axis

For the standard SIMOTION application, the required parameters (configuration data and system variables) are queried in the axis configuration wizard or are specified automatically. For the Axis technology object (TO), you can specify additional selected parameters via the axis parameter assignment dialog boxes (under Axis Object in the project navigator).

The expert list provides read/write access to all configuration data and system variables of the Axis technology object. The list includes data which cannot be set in the axis wizard or in axis parameter assignment dialog boxes.

With V4.1 SP1 and higher, the **Selected Parameters** tab in the expert list provides a default view of the most important configuration data and system variables.

This tab also shows the most important configuration data and system variables for programming and diagnostics of the virtual, drive, and position axis types, the following object of a following axis, and the external encoder.

Note

In individual SIMOTION applications, it may be necessary to change automatically specified parameters. These configuration data and system variables can only be displayed and changed in the expert list.

To open the expert list for the Axis technology object:

1. In the project navigator, double-click TO Axis. The configuration window appears in the working area and the **Axis** menu is displayed.
2. Double-click the **Expert list** entry under the Axis TO in the project navigator. The expert list is displayed in the working area.

Additional information on working with the expert list can be found in the **Motion Control Basic Functions** Function Manual.

4.8.4.6 Automatic controller setting

Overview of automatic controller setting (as of V4.1 SP1)

In the **Automatic Controller Setting** screen form, you can configure an automatic setting of the speed controller and the DSC position controller for SINAMICS drive units. The necessary steps for this calculation can be controlled from this screen form. The calculated parameter values for the speed controller or position controller are displayed and can then be transferred online to the drive or axis on the control.

Open the Automatic Controller Setting screen form

The following options are available for opening this screen form:

- Via the *Target System – Automatic Controller Setting* menu command from the main menu
- From the project navigator (under *Drive – Commissioning*)
- Via the main toolbar (in the Trace/Measuring Functions group)
- From the *Static Controller Data* dialog box for the axis via a button next to the Kv factor

Requirements

The conditions for the automatic speed controller setting and position controller setting are listed below. You can find additional requirements for the automatic position controller setting in the section about the automatic position controller setting (Page 3070).

- Drive is a SINAMICS drive
- Drive is operated in the "SERVO" operating type

- The control is performed with the motor encoder
- An online connection to the associated drive device exists

Note

The software limit switches for the axes do not function during automatic controller setting.

Note

In certain cases, automatic controller setting may not be able to determine the optimum controller settings for servo. This applies to the positioning of band-stops, for example.

Supported devices

An automatic controller setting is possible with the following devices: SINAMICS Integrated, CX32, CX32-2, S120 - CU320, S120 - CU320-2, S120 - CU310 and S120 - CU310-2.

Procedure

The sequence below must be followed for the automatic controller setting:

1. Set the speed controller
   (see also Automatic speed controller setting (as of V4.1 SP1) (Page 3067))
2. Set the DSC position controller
   (see also Automatic position controller setting (as of V4.1 SP1) (Page 3070))

Note

You can cancel automatic controller setting by pressing the SPACEBAR.

- The step currently being executed is canceled.
- The drive enable is inhibited.

See also

Overview of commissioning the position controller of positioning axes (Page 2986)

Assigning automatic controller optimization (Page 3177)

Automatic speed controller setting (as of V4.1 SP1)

In the **Automatic Controller Setting** screen form, you can select the SINAMICS drive unit and the drive for which you want to carry out an automatic speed controller setting.

The automatic setting is divided into individual steps in which the measuring functions on the drive can be initiated, among other things. Drive parameters are changed online to perform these steps. After an individual step has been executed or canceled, the parameter set is restored with its status before the step was performed.

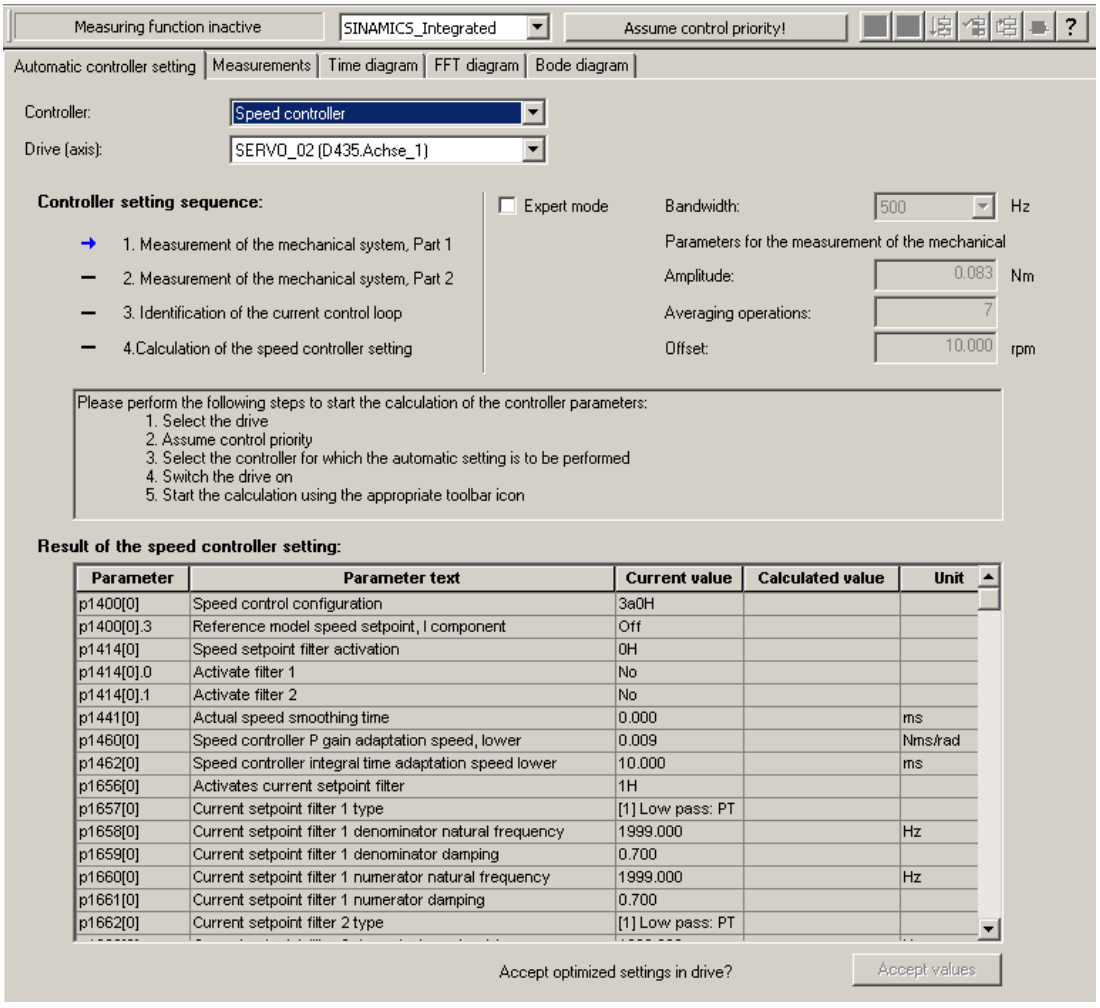


Figure 4-891 Automatic controller setting - Speed controller example

Features

The automatic speed controller setting has the following characteristics:

- Damping of resonances in the speed control section
- Setting of filters in the current setpoint branch
- Automatic setting of the Kp gain factor and the Tn reset time of the speed controller
- The speed setpoint filter and the reference model are not changed

The requirements for the speed controller setting are described in Overview of the automatic controller settings (Page 3066).

Save the current settings

Online drive parameters are changed during a step. It is recommended that the current settings are saved before setting the controller. If the online connection is canceled during the execution of a step, you can restore these saved settings.

Proceed as follows to make a backup:

1. In the project navigator, select the SINAMICS unit with the drive for which you want to perform the automatic setting
2. Load it to the programming device (*Target System - Load - Load to PG...*)

To restore the data, perform a download.

Procedure

Perform the following steps for the automatic setting of the speed controller:

1. Call the **Automatic Controller Setting** screen form (the automatic speed controller setting is already preset in the Controller field)
2. Select the drive unit and drive
3. Assume control priority
4. Enable the drive
5. Perform these four steps in automatic mode or in individual steps
6. Consider the calculation results for the relevant parameters
7. Transfer the calculated speed controller parameter values to the drive
8. Disable the drive
9. Return control priority
10. Save the online parameters

Note

Emergency cancelation of automatic setting with <space key>

The following actions are performed:

- The step currently being executed is canceled
- The drive is disabled

Transferring the calculated parameter values to the online parameters

You initiate the transfer of the calculated parameter values to the corresponding online parameters of the drive with the "Accept" button.

Backing up the automatically set parameters

Proceed as follows to save the parameters:

1. In the project navigator, select the SINAMICS unit with the drive for which you want to perform the automatic setting
2. Copy RAM to ROM (*Target System - Copy RAM to ROM*)
3. Load it to the programming device (*Target System - Load - Load to PG...*)

Note

If required, the automatic controller settings can be checked using the measuring functions.

See also

SIMOTION measuring functions (Page 3073)

Assigning automatic controller optimization (Page 3177)

Automatic position controller setting (as of V4.1 SP1)

In the **Automatic Controller Setting** screen form, you can select the SINAMICS drive unit and the drive for which you want to carry out an automatic DSC position controller setting. The necessary steps for this calculation can be performed from this screen form. The calculated Kv value is displayed and can then be accepted online in the configuration data of the axis that is assigned to the drive.

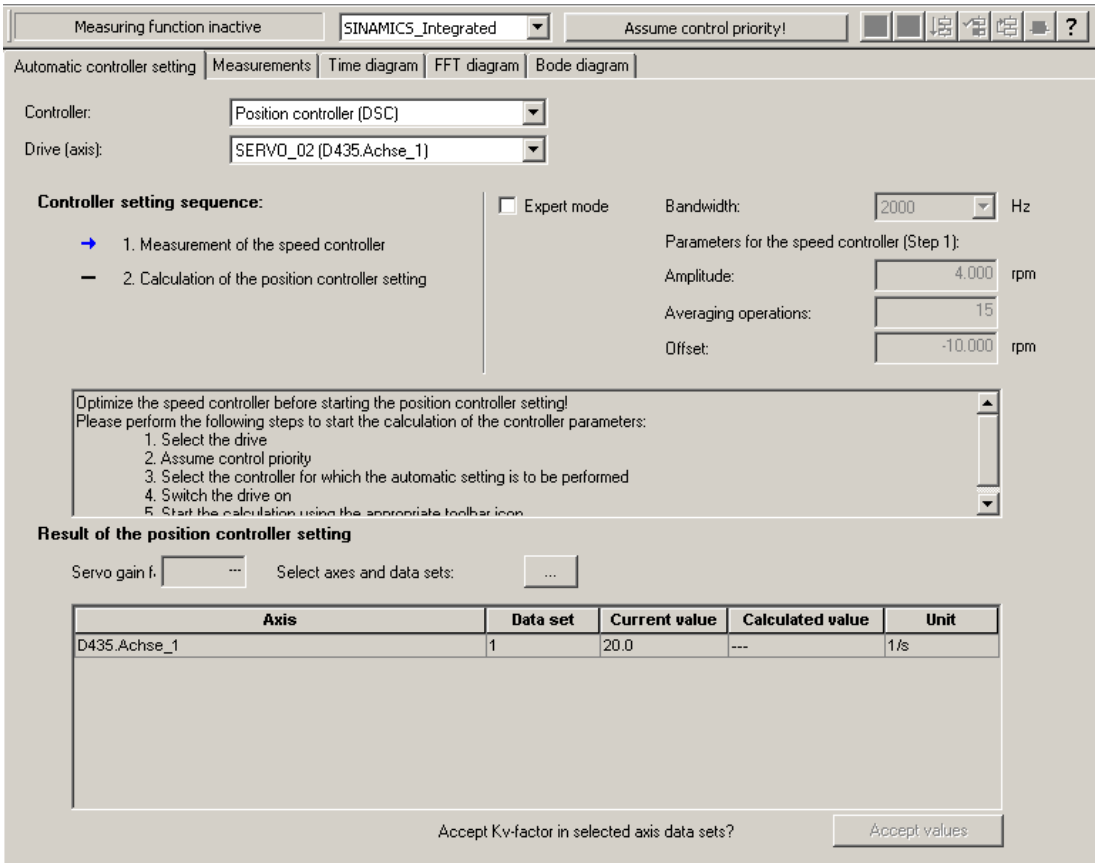


Figure 4-892 Automatic controller setting - Position controller example

Requirements and boundary conditions

In addition to the General requirements (Page 3066) for the automatic controller setting, the following requirements/boundary conditions apply:

- DSC or DSC with spline is required for the position controller setting.
  - You use a telegram that supports DSC (telegram 5, 6, 105, 106, 125 or 126)
  - If you have selected a telegram which does not support DSC, you must select one of the telegrams mentioned above instead.
    As various presettings are required for DSC that can only be set when running through the axis wizard the first time, you have to make these settings manually:
    - Static controller data: Precontrol with weighting factor 100%
      (**NumberOfDataSets.DataSet_1.ControllerStruct.PV_Controller.preCon** = YES and **NumberOfDataSets.DataSet_1.ControllerStruct.PV_Controller.kpc** = 100%)
    - Static controller data: Symmetry filter (extended symmetry filter active)
      (**NumberOfDataSets.DataSet_1.ControllerStruct.PV_Controller.balanceFilterMode** = MODE_2)
    - Static controller data: Fine interpolator (acceleration continuous interpolation)
      (**FineInterpolator._type** = CUBIC_MODE)
    - Dynamic controller data: Speed control loop 0.0 equivalent time and position control loop 0.0 equivalent time
      (**NumberOfDataSets.DataSet_1.DynamicData.positionTimeConstant** = 0.0 and **NumberOfDataSets.DataSet_1.DynamicData.velocityTimeConstant** = 0.0)
- The speed controller has been set previously (e.g. with the automatic speed controller setting).
- At least one axis is connected to the SINAMICS drive (servo).
- The actual values and the manipulated variables between the controller and the drive have been correctly scaled by the user. No validation is performed.
  Match the configurations of the SIMOTION controller with the drive, for example, using the "Transfer data from the drive" button in the axis wizard.
  The following parameters in the drive are used for SINAMICS:
  P2000: Normalization speed
  P1082: Maximum speed
  See also Coupling of digital drives (Page 2895).
- An online connection to the SIMOTION device must exist for the result of the automatic position controller to be transferred.
- The balancing filter is not changed.
- For operation without precontrol, the equivalent time constant of the position controller must be adjusted manually by the user (PostionTimeConstant = 1/Kv).
- Vibration on the load side is not taken into account for the position controller setting.

Procedure

Perform the following steps for the automatic setting of the position controller:

1. Open the **Automatic Controller Setting** screen form
2. Select the drive unit and drive (axis)
3. Specify the "Position controller (DSC)" controller preselection
4. Assume control priority

5. Enable the drive
6. Perform these two steps in automatic mode or in individual steps
7. Select the axis data sets to which the Kv factor are to be transferred
8. Transfer the calculated Kv factor to the axis data sets you have selected
9. Disable the drive
10. Return control priority
11. Back up the online parameters

Note

Emergency cancelation of the automatic setting with <spacebar>

The following actions are performed:

- The step currently being executed is canceled
- The drive is disabled

Transferring the calculated parameter values to the online parameters

You initiate the transfer of the calculated Kv factor to the axis data set of the target device with the "Accept" button.

Backing up the automatically set parameters

Proceed as follows to back up the parameters:

1. In the project navigator, select the SIMOTION unit with the axis for which you want to perform the automatic setting
2. Copy current data to RAM function (*Target System - Copy current data to RAM*)
3. Copy RAM to ROM (*Target System - Copy RAM to ROM*)
4. Load the configuration data to the programming device (*Target System - Load - Load Configuration Data to PG*)

Note

If required, the automatic controller settings can be checked using the measuring functions.

See also

SIMOTION measuring functions (Page 3073)

Assigning automatic controller optimization (Page 3177)

4.8.4.7 SIMOTION measuring functions

The SIMOTION measuring functions are used to commission the axis controller without requiring a user program. In SIMOTION SCOUT, the user can choose a predefined measuring function from a selection of functions. Based on this selection, parameters are then assigned in the target device for the SIMOTION Trace, the function generator, and the required axis enables and motion functions. The user can then launch these measuring functions via SIMOTION SCOUT and evaluate the resulting measurements in the SIMOTION Trace diagrams.

DSC with spline is deactivated when the measuring functions are executed.

The user can configure a user-defined measuring function in expert mode.

The following measuring functions are available:

- **Final controlling element jump (as of V4.0)**
- **Final controlling element frequency response (as of V4.0)**
- **Position control ramp (as of V4.0)**
- **Position control reference frequency response (as of V4.0)**
- **Expert mode (as of V4.0)**

In addition, the **circularity test** is also available for synchronous operation.

Table 4-370 Available measuring functions based on the axis type

	0	1	2	3	4	5	6	7
Positioning/following axis								
Final controlling element jump	X	-	X	X	X	X	-	-
Final controlling element frequency response	X	-	X	X	X	X	-	-
Position control ramp	X	-	X	X	X	X	-	-
Position control reference frequency response	X	-	X	X	X	X	-	-
Expert mode	X	-	X	X	X	X	-	X
Circularity test	X	-	X	X	X	X	-	X
Speed-controlled axis								
Final controlling element jump	X	-	X	X	X	X	-	-
Final controlling element frequency response	X	-	X	X	X	X	-	-
Expert mode	X	-	X	X	X	X	-	-

0 Real electric axis (REAL_AXIS)

1 Virtual axis (VIRTUAL_AXIS)

2 Real electric axis with force/pressure control (REAL_AXIS_WITH_FORCE_CONTROL)

3 Real hydraulic axis (Q valve) (REAL_QFAXIS)

4 Real hydraulic axis with force/pressure specification (Q-valve and P-valve) (REAL_QFAXIS_WITH_OPEN_LOOP_FORCE_CONTROL)

5 Hydraulic axis with pressure/force control (Q valve) (REAL_QFAXIS_WITH_CLOSED_LOOP_FORCE_CONTROL)

6 Real hydraulic axis with force/pressure specification (P-valve) (REAL_QFAX-IS_WITH_OPEN_LOOP_FORCE_CONTROL_ONLY)

7 Real electric axis with master value output via encoder signal simulation (REAL_AX-IS_WITH_SIGNAL_OUTPUT)

Note

Requirements

- An online connection to the SIMOTION device must be established.
- The target device must contain the current axis configuration. If necessary, download the project or upload the configuration data for alignment purposes (Target system > Load > Load configuration data to PG).
- It must be permissible to change the operating mode to STOP_U in the SIMOTION device. An automatic switchover to the STOP_U operating mode will be performed.
- No alarms may be pending on the axis. If necessary, acknowledge the pending alarms in the alarm window and restart the measuring function.

Final controlling element jump measuring function (as of V4.0)

This measuring function can be used for optimizing the lower-level control loop or process (e.g. speed control) in the time range.

A jump function is superimposed on the manipulated variable.

Signal form	Jump with velocity offset
Superimposition point	internalServoSettings.additionalSetpointValue[0]
Measured variables	• Superimposed manipulated variable internalServoSettings.additionalSetpointValue[0] • Actual velocity sensorData[x].velocity **1)**

1) The index is based upon the selected measuring system. The measuring system is selected by the user.

Final controlling element frequency response measuring function (as of V4.0)

This measuring function can be used for optimizing the lower-level control loop or process (e.g. speed control) in the frequency range.
This measuring function can also be used to determine the frequency response of an actuator (e.g. hydraulic process).

A PRBS signal (Pseudo-Random-Binary-Signal) generated by the signal generator is superimposed on the manipulated variable. The control loop is opened.

Signal form	PRBS signal
Superimposition point	internalServoSettings.additionalSetpointValue[0]
Measured variables	• Superimposed manipulated variable internalServoSettings.additionalSetpointValue[0] • Actual velocity sensorData[x].velocity **1)**

1) The index is based upon the selected measuring system. The measuring system is selected by the user.

Position control ramp measuring function (as of V4.0)

This measuring function can be used to optimize the position controller in the time range.

A ramp function is superimposed on the position setpoint. The position control loop is closed.

There are two variations of this measuring function:

- Position control ramp:
  The signal is injected before the dynamic response adaptation.
- Position control ramp without precontrol and setpoint filter:
  The setpoint is injected directly before the summation point of the position controller.

Signal form	Rectangle + rectangular integrator
Superimposition point	• A) Position control ramp internalServoSettings.additionalCommandValue[0] • B) Position control ramp without precontrol and setpoint filter internalServoSettings.additionalControllerCommandValue[0]
Measured variables	• Superimposed setpoint Superimposition point A) or B) • Manipulated variable actorData.totalSetpoint • Actual velocity sensorData[x].velocity **1)**

1) The index is based upon the selected measuring system. The measuring system is selected by the user.

Position control reference frequency response measuring function (as of V4.0)

This measuring function can be used to optimize the position controller in the frequency range.

A PRBS signal (pseudo random binary signal) is superimposed on the position setpoint. The position control loop is closed.

There are two variations of this measuring function:

- Position control reference frequency response:
  The signal is injected before the dynamic response adaptation.
- Position control reference frequency response without precontrol and filter:
  The setpoint is injected directly before the summation point of the position controller.

Function generator 1 for reference frequency response (ramp)		
	Signal form	Ramp (continuous in 1st derivation)
	Superimposition point	• A) Position control reference frequency response internalServoSettings.additionalCommandValue[0] • B) Position control reference frequency response and precontrol and setpoint filter internalServoSettings.additionalControllerCommandValue[0]
Function generator 2 for reference frequency response		
	Signal form	PRBS signal
	Superimposition point	• C) Position control reference frequency response internalServoSettings.additionalCommandValue[1] • D) Position control reference frequency response and precontrol and setpoint filter internalServoSettings.additionalControllerCommandValue[1]
Measured variables		• Superimposed position setpoint 1, ramp Application of function generator 1 A) or B) • Superimposed set position 2, PRBS Application of function generator 2 C) or D) • Actual position sensorData[x].position 1) • Modulo overflow counter, actual value sensorData[x].moduloCycles 1)

1) The index is based upon the selected measuring system. The measuring system is selected by the user.

Expert mode measuring function (as of V4.0)

Various settings can be changed individually in expert mode. Thus, the signal type and superimposition point can be selected explicitly.

Possible signal types:

- PRBS (pseudo random binary signal)
- Jump
- Triangular
- Sinusoidal

The **internalServoSettings.~** structure contains internal system variables for setpoint and manipulated variable override of the measuring functions.

Table 4-371 Possible superimposition points (internalServoSettings.~ structure)

Identifier	Variable	Comment/function
Setpoint	additionalCommandValue	Setpoint superimposition before dynamic response adaptation Same function as servoSettings.additionalCommandValue
Manipulated variable	additionalSetpointValue	Manipulated variable superimposition Same function as servoSettings.additionalSetpointValue
Setpoint on the controller	additionalControllerCommandValue **1)**	Setpoint superimposition on the controller Setpoint superimposition before the summation point of the position/speed controller

1) For measurements that are performed without precontrol or setpoint filter (dynamic response adaptation, balancing filter), the setpoint excitation must occur directly before the summation point of the position or speed controller.

Note

The system variables for superimposition are intended for exclusive use by the SIMOTION measuring functions. These variables cannot be modified by the user.

Circularity test - evaluation of the axis dynamics for synchronous operation with angular synchronism

Minor deviations of amplitude and phase can be seen clearly in a circularity diagram. Another advantage of the circularity test is that it enables exclusive observation of actual values. In contrast, a disadvantage of a following error observation is that the following error (followingError system variable) is corrected, for example, with DSC, and is thus subject to error in a sense.

Function generator 1 of the circularity test, Axis 1		
	Signal form	Sinusoidal
	Superimposition point	• internalServoSettings.additionalCommandValue[0]
Function generator 2 of the circularity test, Axis 2		

	Signal form	Sinusoidal
	Superimposition point	• internalServoSettings.additionalCommandValue[0]
Measured variables		• Position setpoint, Axis 1 internalServoSettings.additionalCommandValue[0] • Position setpoint, Axis 2 internalServoSettings.additionalCommandValue[0] • Actual position, Axis 1 sensorData[x].position **1)** • Actual position, Axis 2 sensorData[x].position **1)** • Actual value modulo overflows, Axis 1 sensordata[x].moduloCycles **1)** • Actual value modulo overflows, Axis 2 sensordata[x].moduloCycles **1)**

1) The index is based upon the selected measuring system. The measuring system is selected by the user.

See also

Setpoint superimposition (Page 2967)

Manipulated variable superimposition (Page 2976)

4.8.4.8 Axis control panel

The axis control panel is used to control and monitor individual axes without requiring a user program. You can use it to traverse axes along with the drive.

Application examples

- Function test on individual axes before the axes are traversed in a coordinated manner under the control of a program
- Traversing axes before the user program is available or if only parts of the user program are available
- Traversing the axes for optimization purposes (controller setting)
- Axis homing or absolute encoder adjustment
- Enabling and disabling an axis
- Acknowledging axis alarms
- Retracting **an axis**: Traverse the axis to another position quickly and independently of the program

Requirements

- An online connection to the SIMOTION device must be established.
- The target device must contain the current axis configuration. If necessary, download the project or upload the configuration data for alignment purposes (Target System - Load - Configuration Data to PG).

⚠ WARNING
Danger to life through unexpected machine movement
Failure to observe the safety instructions can result in personal injury and material damage.
Note the safety instructions in the **Assume Control Priority** SCOUT dialog box.

The system variable **controlPanel** indicates whether the control panel on the Axis technology object is active.

The control panel can be used in operating state STOPU and (from V4.4) in operating state RUN. Please refer to the additional information in the SCOUT Online Help (Axis control panel index).

4.8.5 Support for SINAMICS Safety Integrated Functions

4.8.5.1 Overview - Support of SINAMICS Safety Integrated Functions on the Axis technology object

SIMOTION provides support for SINAMICS drives that can perform safety-related functions. The safety status information of the drive is provided in the system variables.
As of V4.4, SIMOTION commands for triggering safety-related functions are possible on the drive (e.g. SBT).

The purpose of this support from SIMOTION is to prevent stop responses on the drive side by ensuring for the safety-related monitoring functions that the drive does not exit the monitored operating state.

As of SIMOTION V4.4, the SIMOTION-side support for *SINAMICS Safety Integrated Functions* has been simplified and adapted to a generic concept with Drive Safety Data Block (DSDB).

If symbolic assignment is activated (standard setting for new projects), communication is automatically adjusted to the safety-related functions enabled on the drive.

Always use the standard setting (DSDB) on the Axis technology object in conjunction with symbolic assignment for new projects as of V4.4.

For projects < V4.4, the compatibility mode with Safety Information Data Block (SIDB) is available.

The following figure provides decision guidance for managing projects < V4.4 with SCOUT as of V4.4.

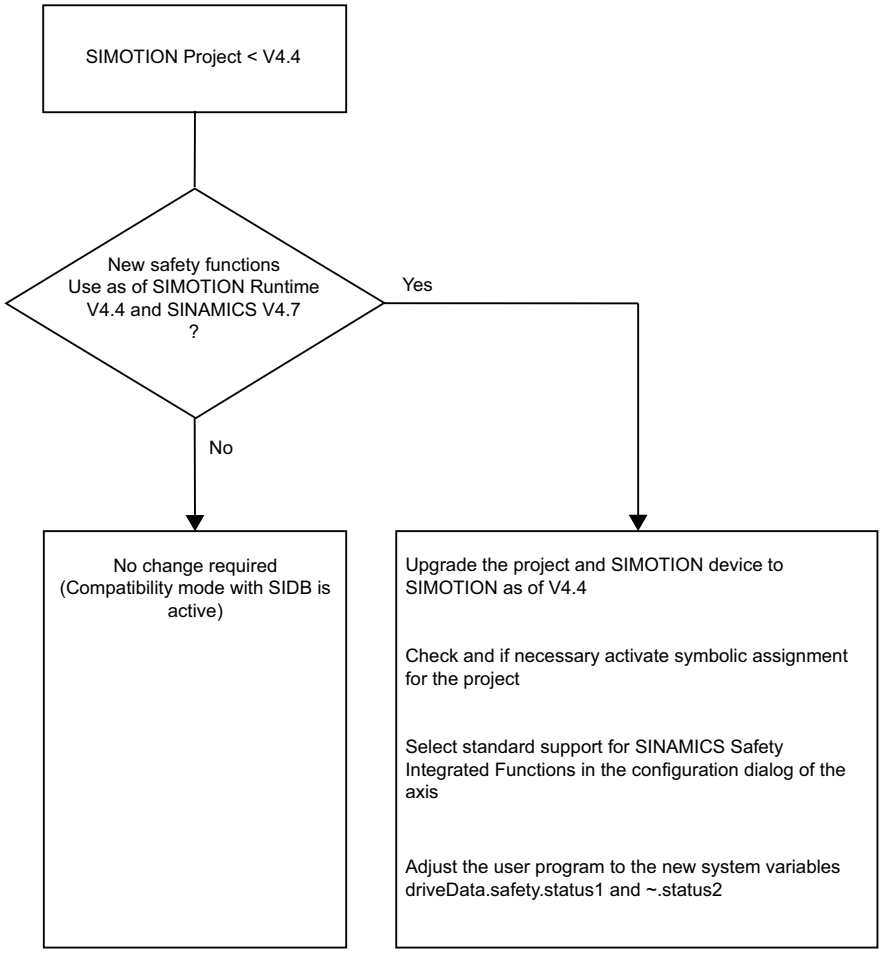


Figure 4-893 Using older SIMOTION projects with SIMOTION SCOUT as of V4.4

If the project was created with SCOUT < V4.2, please refer to Section Restrictions after switching from compatibility mode (SIDB) to standard (DSDB). (Page 3112).

The status information < V4.4 (Page 3093) is still provided for compatibility reasons, even with the standard setting (DSDB), with the exception of SSM.

Note

Further documentation

The safety-related functions of SINAMICS: are described comprehensively in the *SINAMICS Safety Integrated* Function Manual.

As of SIMOTION V5.3

- SLA - Safely-Limited Acceleration with fine resolution and acceleration filtering
- 2 ms monitoring cycle on a double-axis Motor Module

- SS2ESR (Safe Stop 2 Extended Stop and Retract) (The PROFIsafe telegram has been extended with the External Stop E)
- Reluctance motor can be used with Safety encoderless

As of SIMOTION V5.2

- Acceptance of the F destination address from HW Config (STEP 7) or HWCN (TIA Portal)
- Support of new safety-related functions SLA and SCA
- Support of PROFIsafe telegram 903
- SLP, SP, Safe Referencing, SCA. As of SINAMICS V5.1, Advanced Safety is required SP synchronous cannot be used with SIMOTION
- New safety licenses as of SINAMICS V5.1, see Safety licenses (Page 3087)

As of SIMOTION V4.5

- Selection of response to PROFIsafe failure (STOP B)

As of SIMOTION V4.4

Simplification and implementation of the SIMOTION-side support with DSDB

Extension of the safety-relevant functionality:

- SIMOTION-side support of the DSDB with automatic selection of telegram extension
- *SINAMICS Safety Integrated Basic Functions* support on the TO axis
- Support for new safety-related functions: SP, SLP, SBT
- Support for an additional PROFIsafe telegram (902 on the TIA Portal)

As of SIMOTION V4.3

Extension of the safety-relevant functionality:

- SDI function is available in SIDB
- SKS stage 1 can be evaluated with a factor transferred via PROFIsafe, whereby almost any SLS limit values can be specified online
- Support for new SINAMICS V4.5 functions
- Support for additional PROFIsafe telegrams (31 and 901)

As of SIMOTION V4.2

Extension of the safety-relevant functionality:

- SIMOTION-side support for SIDB with automatic telegram extension
- Additional generation of the technological alarm 50023 in the event of drive-autonomous changes in status
  Drive-autonomous changes in status occur e.g. when SS1 or SS2 is selected
- Read-out of the safety message buffer via system function

See also Coupling of digital drives (Page 2895).

As of SIMOTION V4.1 SP1

SIMOTION supports the *SINAMICS Safety Integrated Extended Functions* for the axis TO in the following way:

- Display of the status of safety-related functions in the drive in system variables of the technology object
- Message of a new safety event in the safety message buffer of the drive (alarm TO)
- Message of the status for SLS, SOS and SS2 (alarm TO)

See also

Technology data (Page 3018)

Safety status information (part of the DSDB) (Page 3088)

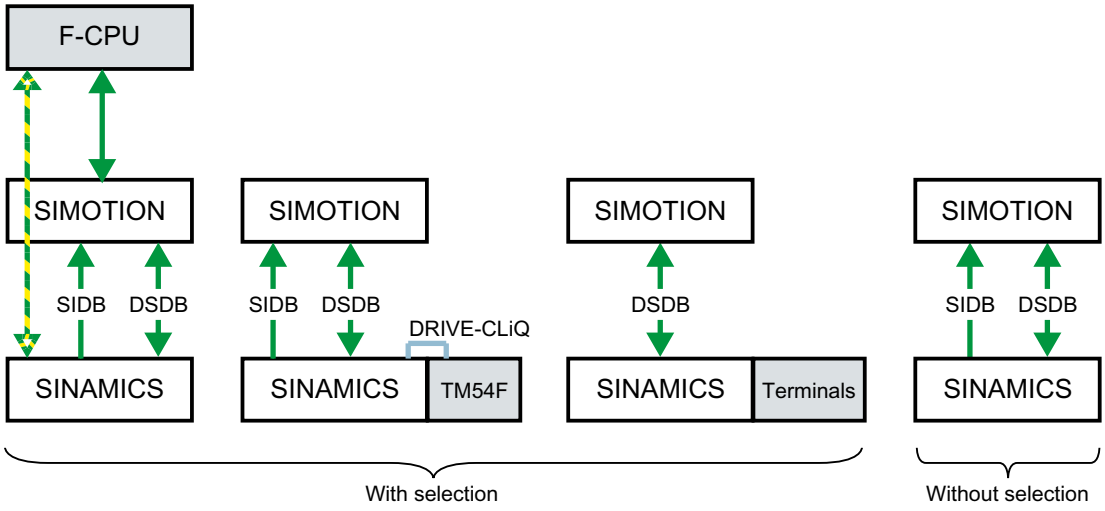
Safety channel and safety channel types (when symbolic assignment is deactivated) (Page 3085)

Standard (DSDB) as of SIMOTION V4.4 (Page 3087)

4.8.5.2 Communication

As of SIMOTION V4.4, the safety information is transferred between the drive and SIMOTION by default as a telegram extension via the DSDB. Both *SINAMICS Safety Integrated Basic Functions* and *SINAMICS Safety Integrated Extended Functions* are supported.

Prior to SIMOTION < V4.4, the safety information is transferred from the drive via the SIDB. *SINAMICS Safety Integrated Extended Functions* are supported.



Optional SIDB (compatibility mode) or DSDB (standard)
SIDB: Supports the SINAMICS Safety Integrated Extended Functions
DSDB (as of SIMOTION V4.4): Supports the SINAMICS Safety Integrated Basic & Extended Functions

Figure 4-894 Communication with DSDB or SIDB in different safety configurations

4.8.5.3 Main procedure for safety configuration with SIMOTION

To use the support for the *SINAMICS Safety Integrated Extended Functions* to the axis TO, proceed as follows:

- Start up the axis and the drive without safety functions
- Configure the safety-related functions in the drive (see *SINAMICS Safety Integrated* Function Manual)
- If the safety-related function is to be activated via PROFIsafe, configure PROFIsafe communication to a safety PLC (e.g. SIMATIC S7 CPU317F).
  See Section *PROFIsafe* in the *SIMOTION SCOUT Communication* System Manual.
- Activate the SIMOTION-side support for the safety-related functions:
  - Connect the drive symbolically in the axis configuration to the axis. To do so, click the ... button under drive assignment.
    or
  - Activate the safety-related functions in the safety dialog.
- Only if symbolic assignment is deactivated:
  Configure the DSDB or SIDB as extension in the telegram.
  As of SIMOTION V4.2, this is set up automatically by the system and interconnected in the drive if symbolic assignment is activated.

The safety status information transferred by DSDB (Page 3088) or SIDB (Page 3093) can be evaluated via system variables on the TO. The status is required for many safety functions to bring the axis into the permissible operating range via the user program.

The *SINAMICS Safety Integrated Functions* are displayed in the configuration dialog of the axis.

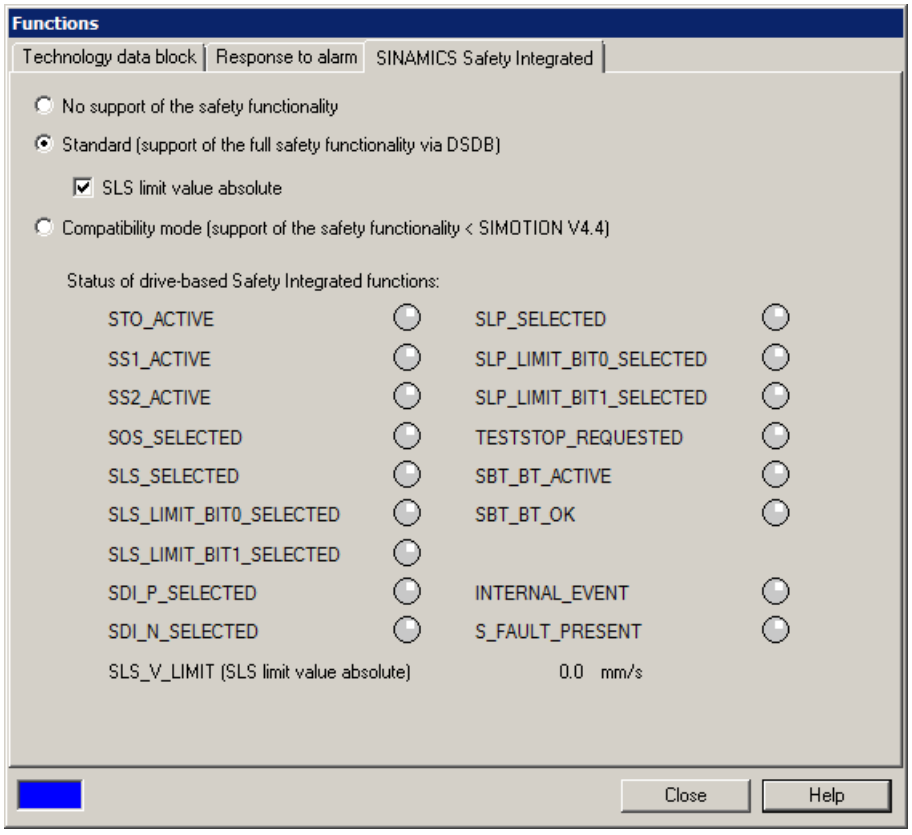


Figure 4-895 Safety dialog

Table 4-372 Support safety functions and their status information

	Safety-related function in the drive	**DSDB - standard (driveData.safety.status1 and driveData.safety.status2)**	**SIDB - compatibility mode (driveSafetyExtendedFunctionsInfoData.state)**
STO	Safe Torque Off	Status1 - bit 0 (STO_ACTIVE)	Bit 0 (STO_ACTIVE)
SS1	Safe Stop 1 (SS1)	Status1 - bit 1 (SS1_ACTIVE)	Bit 1 (SS1_ACTIVE)
SS2	Safe Stop 2 (SS2)	Status1 - bit 2 (SS2_ACTIVE)	Bit 2 (SS2_ACTIVE)
SOS	Safe operating stop	Status1 - bit 3 (SOS_SELECTED) Status2 - bit 4 (SOS_ACTIVE)	Bit 3 (SOS_SELECTED)
SLS	Safely Limited Speed	Status1 - bit 5 (SLS_SELECTED) Status1 - bit 6 (SLS_LIMIT_BIT0_SELECTED) Status1 - bit 7 (SLS_LIMIT_BIT1_SELECTED) Status2 - bit 5 (SLS_ACTIVE)	Bit 4 (SLS_SELECTED)
SSM	Safe Speed Monitor (SSM)	-	Bit 7 (SSM_ACTIVE)
SDI	Safe Direction (safe direction of motion)	Status1 - bit 8 (SDI_P_SELECTED) Status1 - bit 9 (SDI_N_SELECTED)	Bit 12 (SDI_P_SELECTED) Bit 13 (SDI_N_SELECTED)
SP	Safe Position SP synchronous cannot be used with SIMOTION	Status1 - bit 23 (REF_POS_LATCHED_OR_VALID) Status1 - bit 24 (REF_REQUESTED)	-

	Safety-related function in the drive	DSDB - standard (driveData.safety.status1 and driveData.safety.status2)	SIDB - compatibility mode (driveSafetyExtendedFunctionsInfoData.state)
SLP	Safely Limited Position (SLP)	Status1 - bit 11 (SLP_SELECTED) Status1 - bit 12 (SLP_LIMIT_BIT0_SELECTED) Status1 - bit 23 (REF_POS_LATCHED_OR_VALID) Status1 - bit 24 (REF_REQUESTED)	-
SBT	Safe Brake Test (SBT)	Status2 - bit 15 (SBT_SELECTED) Status2 - bit 16 (SBT_CONTROL_PRIO_DRIVE) Status2 - bit 17 (SBT_SIGN_LOAD) Status2 - bit 18 (SBT_BT_BRAKE_2) Status2 - bit 19 (SBT_BT_ACTIVE) Status2 - bit 20 (SBT_BT_OK) Status2 - bit 21 (SBT_BT_FINISHED) Status2 - bit 22 (SBT_BT_BRAKE_CLOSE_REQ)	-
SLA	Safely-Limited Acceleration	Status1 - bit 16 (SLA_SELECTED)	-
SCA	Safety-related function in the drive: Safe Cam	Standard: Status1 - bit 23 (REF_POS_LATCHED_OR_VALID) Status1 - bit 24 (REF_REQUESTED)	-

The stop responses in the drive (e.g. STOP A) for the applicable safety-related functions are described in the *SINAMICS Safety Integrated* Function Manual.

4.8.5.4 Safety channel and safety channel types (when symbolic assignment is deactivated)

This section is only applicable if the project standard setting **Use symbolic assignment** has been deliberately deactivated.

For communication, a distinction is made between various SC types (safety channel types). The various SC types can be used on the SC channels (SIDB, DSDB_1, DSDB_2).

In compatibility mode, the SIDB is the only SC type, so there is no need to distinguish between types.

Use of SIDB and DSDB is optional. Parallel use is not possible.

Table 4-373 List of SC types in SIDB

SC types of SIDB	Supported safety-relevant functions	Input parameter	Output parameter
SIDB	STO SS1 SS2 SOS SLS SDI	r9722.0 r9722.1 r9722.2 r9722.11 r9720.4, r9733[2] r9734.12, r9734.13	-

Note

Support for DSDB when symbolic assignment is deactivated

Note that the standard setting (DSDB) must not be used for SINAMICS drives < V4.7.

Table 4-374 List of SC types in DSDB

SC types of DSDB_x	Supported safety-relevant functions	Input parameter	Output parameter
SC1	STO SS1 SS2 SOS SLS SDI SLA	r9734	-
SC2	as SC1 SLP SP[1)] SCA	r9734 r9743	p10250
SC3	as SC2 SBT	r9734 r9743 r10234	p10250 p10235
SC100		r9733[2]	-

1) SC2 only when using SP with a safe absolute position

4.8.5.5 Safety licenses

SINAMICS makes a distinction between the following function groups:

- Safety Integrated Basic Functions
- Safety Integrated Extended Functions
- Safety Integrated Advanced Functions

As of SIMOTION V5.2 and SINAMICS V5.1, you require the Safety Integrated Advanced Functions license for SLP, SP and SCA.

4.8.5.6 Standard (DSDB) as of SIMOTION V4.4

DSDB is the standard setting for safety as of SIMOTION V4.4. The setting provide the complete safety functionality set up on the drive at the TO. Configuration is automatic and uses the DSDB channel (Drive Safety Data Blocks).

Note

The Drive Safety Data Block is an extension of the PROFIdrive axis telegrams. Transmission is according to PROFIdrive and is therefore not a secure safety telegram.

Display of SINAMICS Safety Integrated Functions in SIMOTION

If support for *SINAMICS Safety Integrated Functions is activated* (**technologicalData.numberOfDriveSafetyDataBlock** > 0), the following information is displayed via the system variable:

- **driveData.safety.status1**
  Contains safety status information to which the user program generally must respond immediately.
- **driveData.safety.status2**
  Contains safety status information to which no time-critical response is generally needed.
- **driveData.safety.slsSpeedLimitSelected**
  Effective absolute set velocity limitation in the drive from the DSDB.
  slsSpeedLimitSelected contains the maximum set velocity at which the Axis technology object may run.

  Note

  The value in **slsSpeedLimitSelected** is already weighted with drive parameter p9533. If p9533 < 100%, it does not represent the maximum possible actual velocity monitored by the drive (p9531[x]).

If **technologicalData.numberOfDriveSafetyDataBlock** (standard with DSDB) = 0 and **technologicalData.driveSafetyExtendedFunctionsEnabled** (compatibility mode with SIDB) = NO, there is no support for *SINAMICS Safety Integrated Functions* on the Axis technology object.

In the system variables, zero is displayed or the contents are irrelevant.

Axis configuration

If *SINAMICS Safety Integrated Functions* are configured in the drive, support is activated automatically on the Axis technology object if a drive is assigned.

The following configuration data settings are made:

- **technologicalData.numberOfDriveSafetyDataBlock** > 0
- **technologicalData.SafetyDataBlock_n** = type, logAddressIn, logAddressOut

Deactivated symbolic assignments

The rest of the chapter is only relevant if the project standard setting **Use symbolic assignments** is deliberately deactivated.

To configure the *SINAMICS Safety Integrated Functions*, proceed as follows:

- Select the SC type in the safety dialog for a maximum of two data blocks (DSDB_1 and DSDB_2) and enter the logical base addresses.
- Evaluate the status of the pulse enable of the drive on the Axis technology object (Page 3107).

If a safety commissioning of the *SINAMICS Safety Integrated Functions* in the drive has already been performed and the data has been loaded into the configuration (load into the PG), click the "Data transfer from the drive" button (drive assignment in the axis wizard) to make these settings.

If this is not required, *SINAMICS Safety Integrated Functions* support can be deactivated by setting the **technologicalData.numberOfDriveSafetyDataBlock**=0 configuration data element.

Safety status information (part of the DSDB)

The following tables show the structure of the safety status information in system variables on the TO axis.

Table 4-375 **Allocation of the system variable driveData.safety.status1**

Bit	State	Identifier	Interconnection in the drive
0	STO is active	STO_ACTIVE	r9734.0
1	SS1 is active	SS1_ACTIVE	r9734.1
2	SS2 is active	SS2_ACTIVE	r9734.2
3	SOS is selected	SOS_SELECTED	r9734.5
4	-	-	-
5	SLS is selected	SLS_SELECTED	r9734.6
6	Bit 0 for selected SLS limit value	SLS_LIMIT_BIT0_SELECTED	r9734.9
7	Bit 1 for selected SLS limit value	SLS_LIMIT_BIT1_SELECTED	r9734.10
8	SDI positive selected	SDI_P_SELECTED	r9734.12

Bit	State	Identifier	Interconnection in the drive
9	SDI negative selected	SDI_N_SELECTED	r9734.13
10	-	-	-
11	SLP selected	SLP_SELECTED	r9743.7
12	Bit 0 for selected SLP limit value	SLP_LIMIT_BIT0_SELECTED	r9743.4
13	-	-	-
14	SLP acceptance test on the drive is active	ACCEPTANCE_TEST_SLP_ACTIVE	r10234.14
15	-	-	-
16	SLA is selected (as of V5.2)	SLA_SELECTED	r9734.8
17	-	-	-
18	-	-	-
19	-	-	-
20	-	-	-
21	-	-	-
22	-	-	-
23	Pos latch command has been recognized by the drive	REF_POS_LATCHED_OR_VALID	r9743.15
24	Axis is not homed - SLP and SP function not available	REF_REQUESTED	r9743.14
25	-	-	-
26	-	-	-
27	Acceptance test on the drive is selected	ACCEPTANCE_TEST_SELECTED	r10234.15
28	Test stop sequence running on the drive	TESTSTOP_ACTIVE	r9743.12
29	Test stop timer has ended	TESTSTOP_REQUESTED	r9743.13
30	Entry present in the SINAMICS safety fault buffer	S_FAULT_PRESENT	r9734.15
31	Safety error has occurred	INTERNAL_EVENT	r9734.7

Table 4-376 **Allocation of the system variable driveData.safety.status2**

Bit	State	Identifier	Interconnection in the drive
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	Safe standstill monitoring is active	SOS_ACTIVE	r9734.3
5	Safe speed monitoring is active	SLS_ACTIVE	r9734.4
6	-	-	-
7	-	-	-
8	Request retract function	ESR_REQUESTED	r9734.14

Bit	State	Identifier	Interconnection in the drive
9	-	-	-
10	-	-	-
11	-	-	-
12	-	-	-
13	-	-	-
14	-	-	-
15	Safe brake test is selected	SBT_SELECTED	r10234.0
16	Setpoint specified during safe break test in the drive	SBT_CON-TROL_PRIO_DRIVE	r10234.1
17	Sign of the load torque is positive	SBT_SIGN_LOAD	r10234.7
18	Test being performed with brake 2	SBT_BT_BRAKE_2	r10234.2
19	Brake test sequence is active	SBT_BT_ACTIVE	r10234.3
20	Test of brake was OK	SBT_BT_OK	r10234.4
21	Brake test was completed	SBT_BT_FINISHED	r10234.5
22	Request close external brake	SBT_BT_BRAKE_CLOSE_REQ	r10234.6
23	-	-	-
24	-	-	-
25	-	-	-
26	-	-	-
27	-	-	-
28	-	-	-
29	-	-	-
30	-	-	-
31	-	-	-

4.8.5.7 Compatibility mode (SIDB) as of SIMOTION V4.1

The compatibility mode is available for the following projects:

- Projects < SIMOTION V4.4 with Safety Information Data Block (SIDB)
- Projects that have been converted upwards from < V4.4

This mode can only be activated if the *SINAMICS Safety Integrated Extended Functions* settings have been set on the drive.

Safety Information Data Block (SIDB)

The Safety Information Data Block (SIDB) shows the state of the activated *SINAMICS Safety Integrated Extended Functions* in SIMOTION.

Note

The Safety Information Data Block (SIDB) is an extension of the PROFIdrive axis telegrams. Transmission is according to PROFIdrive and is therefore not a secure safety telegram.

Table 4-377 Structure of the Safety Information Data Block (SIDB) in the actual value telegram

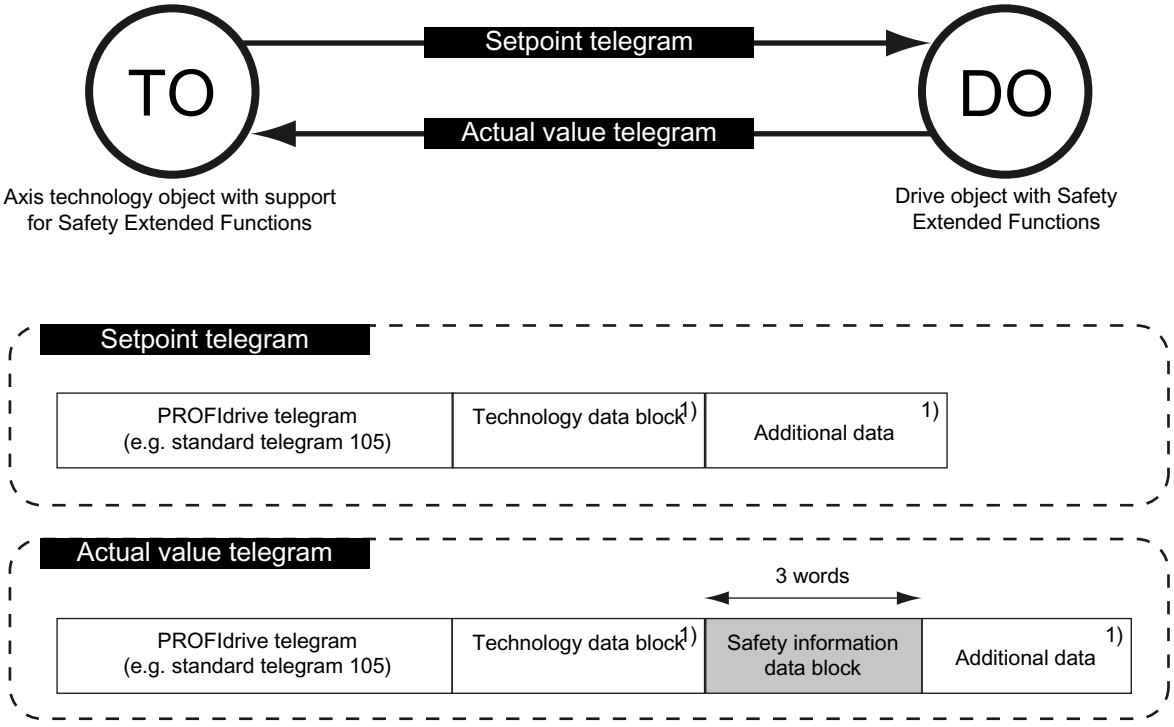
Word	Meaning
1	Safety status word (Page 3093)
2	Effective absolute set velocity limitation in the drive **Note:** This double word must be interconnected with r9733[0] (up to SINAMICS V4.4 only) or with r9733[2] (as of SINAMICS V4.4). For the *Interconnection of the Safety Information Data Block (SIDB) to the actual value telegram in the drive*, see figure below.
3	

As of SIMOTION V4.2, the Safety Information Data Block (SIDB) is set up by the system and automatically interconnected in the drive.

Telegram extension (V4.1 only or with symbolic assignment deactivated)

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deliberately deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

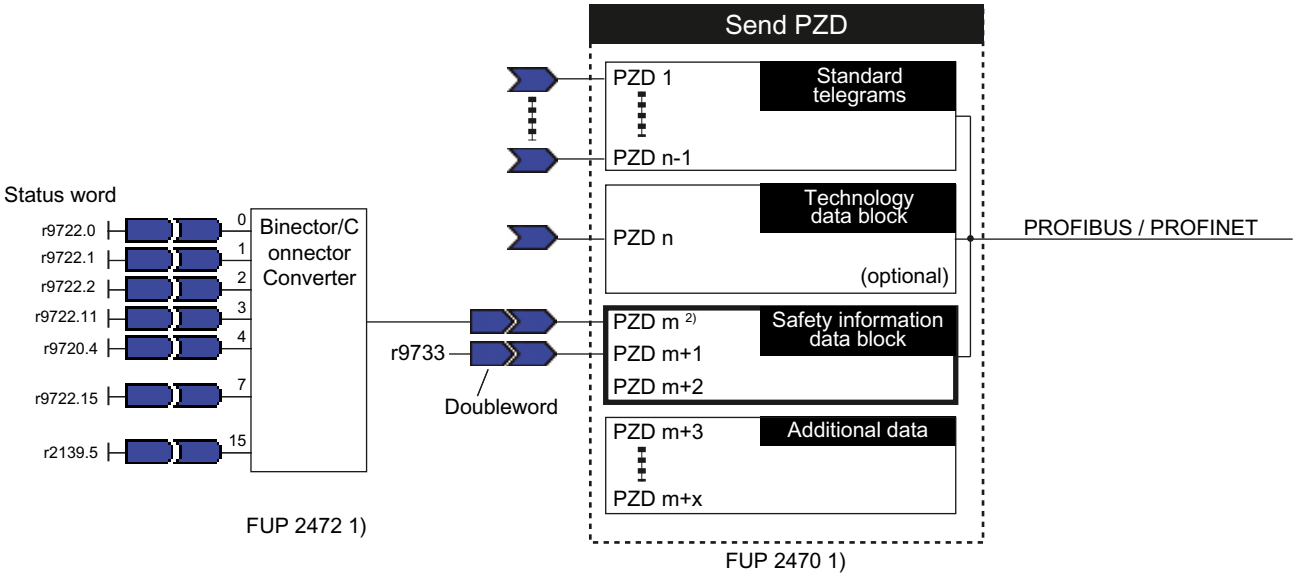
The SIMOTION support for **SINAMICS Safety Integrated Extended Functions** requires the actual value telegram to be extended by the Safety Information Data Block (SIDB) (three words).



1) Optional

Figure 4-896 Layout of a drive telegram with Safety Information Data Block

In order for the *SINAMICS Safety Integrated Extended Functions* to be displayed in SIMOTION, the corresponding SINAMICS parameters must be interconnected with the Safety Information Data Block (SIDB) using BICO.



1) Function charts can be accessed in online help via the “PDF documents” button
2) If technology data block is available: m = n + 1
If no technology data block is available: m = n

Figure 4-897 Interconnection of the Safety Information Data Block (SIDB) to the actual value telegram in the drive

The configuration of the telegram extension is configured using the SINAMICS device > Communication > Telegram configuration dialog with an extension of the input data of the axis telegram by three words.
Via *Configure telegram*, the interconnection of the user-defined PZDs can be carried out using a binector-connector converter.

The logical base address for the Safety Information Data Block (SIDB) **technologicalData.driveSafetyExtendedFunctionsInfoDataIn.logAddress** results as follows:
standard telegram base address + (m-1) * 2
(m is the number of the PZD where the Safety Information Data Block (SIDB) begins). (See also figure above).

Also refer to the information for creating the technology data block in Chapter Coupling of digital drives (Page 2895).

See also

Display of SINAMICS Safety Integrated Functions in SIMOTION (Page 3095)

Safety status information (part of the SIDB)

The following table shows the structure of the safety status word:

If standard SINAMICS Safety Integrated functionality with DSDB is activated, the displays in the safety status word of the SIDB are also updated for compatibility reasons. The content of the

driveSafetyExtendedFunctionsInfoData.state system variable then also displays the information for the *Safety Integrated Basic Function* in this case.

Table 4-378 Structure of the safety status word

Bit	State	Identifier		Interconnection in the drive
0	STO is active	STO_ACTIVE	1-active	r9722.0
1	SS1 is active	SS1_ACTIVE	1-active	r9722.1
2	SS2 is active	SS2_ACTIVE	1-active	r9722.2
3	SOS is selected	SOS_SELECTED	1-active	r9722.11
4	SLS is selected	SLS_SELECTED	0-active	r9720.4
5	Reserved	-	-	-
6	Reserved	-	-	-
7	SSM (n below limit) (See also Restrictions after switching from compatibility mode (SIDB) to standard (DSDB). (Page 3112))	SSM_ACTIVE	1-active	r9722.15
8	Reserved	-	-	-
9	Reserved	-	-	-
10	Reserved	-	-	-
11	Reserved	-	-	-
12	SDI positive selected (as of SINAMICS V4.4)	SDI_P_SELECTED	1-active	r9734.12
13	SDI negative selected (as of SINAMICS V4.4)	SDI_N_SELECTED	1-active	r9734.13
14	Reserved	-	-	-
15	Entry present in the SINAMICS safety fault buffer (See SINAMICS Safety Fault Buffer (Page 3110))	S_FAULT_PRESENT	1-active	r2139.5

Display of SINAMICS Safety Integrated Functions in SIMOTION

If support for *SINAMICS Safety Integrated Extended Functions* is activated (**technologicalData.driveSafetyExtendedFunctionsEnabled**=YES), the following information is displayed via the system variable:

- **driveData.driveSafetyExtendedFunctionsInfoData.state**
  Displays the safety status word (Page 3093).
  The status word can be used to evaluate the status of the safety functions in the drive in the controller. The status is required for many safety functions, in order to bring the axis from the user program to the permissible operating range. See also Overview (Page 3096).
- **driveData.driveSafetyExtendedFunctionsInfoData.safeSpeedLimit**
  Effective absolute set velocity limitation in the drive from the Safety Information Data Block (SIDB), see also the table for the Structure of the Safety Information Data Block (SIDB) (Page 3091).
  safeSpeedLimit contains the maximum set velocity at which the Axis technology object may run.

  Note

  The value in **safeSpeedLimit** is already weighted with drive parameter p9533. If p9533 < 100%, it does not represent the maximum possible actual velocity monitored by the drive (p9531[x]).

If **technologicalData.numberOfDriveSafetyDataBlock** = 0 (standard with DSDB) and **technologicalData.driveSafetyExtendedFunctionsEnabled** = NO (compatibility mode with SIDB), there is no support for *SINAMICS Safety Integrated Functions* on the Axis technology object.

In the system variables, zero is displayed or the contents are irrelevant.

Axis configuration

As of SIMOTION V4.2, if *SINAMICS Safety Integrated Extended Functions* are configured, support is activated automatically in SIMOTION.

The configuration data element **technologicalData.driveSafetyExtendedFunctionsEnabled** = YES is also set**.**

Symbolic assignments deactivated or SIMOTION < V4.2

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deliberately deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

To configure the *SINAMICS Safety Integrated Extended Functions*, proceed as follows:

- Set the **technologicalData.driveSafetyExtendedFunctionsEnabled** configuration data element to YES. This activates the Extended Functions support.
- Enter the logical base address for the Safety Information Data Block (SIDB) in **technologicalData.driveSafetyExtendedFunctionsInfoDataIn.logAddress**.
- Evaluate the status of the pulse enable of the drive on the axis technology object (Page 3107).

If a safety commissioning of the *SINAMICS Safety Integrated Extended Functions* in the drive has already been performed and the data has been loaded into the configuration (load to PG), click the **Data transfer from the drive** button (drive assignment in the axis wizard) to make these settings.

If this is not required, *SINAMICS Safety Integrated Extended Functions* support can be deactivated by setting the **technologicalData.driveSafetyExtendedFunctionsEnabled** configuration data item to NO.

4.8.5.8 Behavior and user responses

Overview

The safety functions in the drive are selected and deselected either from a Safety PLC (e.g. SIMATIC S7 CPU317F) using secure PROFIsafe communication, from a safety-related SINAMICS terminal module TM54F or using the fail-safe terminals on the CU (D410-2, CU310-2 or CU305). The safety functions can be selected and deselected independently of the Motion Control user program.

Many safety functions (e.g. SOS, SLS) contain monitoring functions and must be supported by the SIMOTION user program so that the drive can be set to or kept in the monitored operating state. If the monitored limits are exceeded, this causes a safety-related stop response in the drive.

The user can evaluate the safety status word via system variables. See also the status information for DSDB (Page 3088) and SIDB (Page 3093).

General recommendation for the user program when using SINAMICS Safety Integrated Functions as of SIMOTION V4.2

The system responds automatically to the selection of SS1 or SS2 by switching the axis to follow-up mode. The following requirements must be satisfied to do this:

- Default settings for symbolic assignment
- Automatic telegram selection
- Evaluation of the pulses enabled bits must not be changed

When STO is selected, the drive shuts down automatically.
Therefore, safety functions STO, SS1, and SS2 do not require any provision to be made in the user program in terms of a response.

For other functions such as SOS or SLS, it must be ensured via the user program that the monitored operating state is not violated.

See also

Overview - Support of SINAMICS Safety Integrated Functions on the Axis technology object (Page 3079)

Display of SINAMICS Safety Integrated Functions in SIMOTION (Page 3095)

Overview of safety functions

How the drive reacts and what user responses we recommend in SIMOTION are described below.

Note

Design your user program in accordance with the priority of the safety drive functions. Take into account possible consecutive responses of the drive, e.g. STOP_B in the event of SLS limit value violation.

Evaluate the status bits cyclically in your user program.

Also take into account user responses for connected axes.

Table 4-379 Responses of the drive and responses in the user program when a safety function is selected

Function	Drive response	Recommended response in the user program
STO	Drive switches off immediately	No response possible
SS1E	Drive switches off after expiration of a delay time	Bring axis to a standstill before expiration of the delay time and switch off the drive
SS1	Drive decelerates on the OFF3 ramp and then switches off	Set axis to follow-up mode (automatic as of SIMOTION V4.2)
SS2	Drive decelerates on the OFF3 ramp and then monitors the operating stop	Set axis to follow-up mode (automatic as of SIMOTION V4.2)
SOS	Drive monitors the operating stop after expiration of the delay time	Bring axis to a standstill and remain at a standstill
SLS	Drive monitors the maximum permissible velocity after expiration of a delay time	Limit the axis velocity accordingly
SDI	Drive monitors the direction of motion after expiration of a delay time	Maintain or stop the direction of motion of the axis accordingly
SP	Drive transfers actual position to the F-CPU	-
SLP	Drive monitors compliance with a defined position range after expiration of a delay time	Maintain position range of the axis accordingly
SLA	The drive monitors the maximum permissible acceleration after expiration of a delay time	Limit the axis accelerations accordingly
SCA	The drive transfers safe cam information to the F-CPU	-

Safe Torque Off (STO)As of SIMOTION V4.1

When the **STO** is selected, pulse suppression will be activated directly in the drive: A moving motor coasts to standstill. If there is a motor holding brake and activation via sequential control system, the brake closes.

If the drive enables have been canceled by **STO**, they can be reset by the SIMOTION user program with the **_enableAxis()** command. An enable, however, is possible only when **STO** has been deselected again (safe power-on disable).

In compatibility mode with SIDB, **STO** is only supported in conjunction with the *SINAMICS Safety Integrated Extended Functions* on the axis.

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safe Stop 1 (SS1)As of SIMOTION V4.1

SS1 with OFF3

When **SS1** is selected, the drive independently decelerates on the OFF3 ramp in a speed-controlled manner (n=0) and enters pulse suppression after a configured delay time.

The drive is shut down when pulse suppression is achieved. This is indicated on the Axis technology object by alarm 20005 reason 4.

NOTICE
Abort of the OFF3 ramp
For certain settings, the OFF3 ramp cannot be aborted on the control side!
See SIMOTION system behavior with a drive-autonomous OFF3 ramp (Page 2893).

User response as of SIMOTION V4.2:

The system responds automatically to SS1 selection if the configuration data item **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** is set to YES (set accordingly when creating an axis as of SIMOTION V4.2).
At the start of the OFF3 ramp, TO alarm 50023 (drive makes transition to autonomous state, default local response = OPEN_POSITION_CONTROL) is signaled, switching the axis to follow-up mode automatically.
Consequently, both the position controller enable and following error monitoring are inactive during the OFF3 ramp, and no more motion commands are effective. The drive enables can be canceled with the **_disableaxis()** user command, for example.
Automatic follow-up mode is only activated for the axis concerned in each case. For a possible synchronous operation line-up, the corresponding responses must be converted in the application.

If the drive enables have been canceled by **SS1**, they can be reset by the SIMOTION user program with the **_enableAxis()** command. An enable, however, is possible only when **SS1** has been deselected again (safe power-on disable).

User response with SIMOTION V4.1 or TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation=COMPATIBILITY_MODE:

When SS1 is selected (status information SS1_ACTIVE=TRUE), the current axis motion must be canceled and the position setpoint switched to follow-up mode (with the **_stop()** command at velocity = 0 and movingMode=SPEED_CONTROLLED, for example). This prevents the enables for the drive from having to be canceled (as a result of alarm 50102, following error monitoring, for example).

SS1E external stop (as of SINAMICS V4.5 and SIMOTION V4.3)

If SS1E is selected, the drive switches off after a delay time (p9652 Safety Integrated Basic Functions / p9556 Safety Integrated Extended Functions) without being autonomously decelerated along the OFF3 ramp. Within the delay time, synchronous axes can be stopped synchronously via the user program.

User response to use of the DSDB (SIMOTION V4.4 and higher) or the SIDB (for Safety Integrated Extended Functions):

If SS1E is selected (status information SS1_ACTIVE = TRUE), stop the axis before the delay time elapses and switch off the drive.

User response to Safety Integrated Basic Functions in the user program in SIMOTION V4.3:

The following section is only relevant to SIMOTION V4.3 or if the DSDB is not used.

Transfer parameters p9772.2 and p9872.2 and check them cyclically, for example, in the IPO-synchronous task.

As soon as at least one of the two parameters is TRUE, the axis must be stopped before the delay time elapses and the drive switched off.

Control via terminals

If controlling via terminals, please see the *SINAMICS Safety Integrated* Functional Manual, Chapter *Simultaneity and tolerance time of the two monitoring channels*.

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safe Stop 2 (SS2)As of SIMOTION V4.1

When Safe Stop 2 is selected, the drive decelerates on the speed-controlled OFF3 ramp (n=0) and then switches to the Safe Operating Stop (SOS). The drive then decouples itself from the SIMOTION setpoint interface.

NOTICE
Abort of the OFF3 ramp For certain settings, the OFF3 ramp cannot be aborted on the control side! See SIMOTION system behavior with a drive-autonomous OFF3 ramp (Page 2893).

Note

If the drive is not yet ready to run and SS2 has already been selected, all drive enables can only be set once SS2 has been deselected.
This may be the case following a further ramp up of the selected SS2, for example.

The drive enables can be set using the **_enableaxis()** user command, for example.

User response as of SIMOTION V4.2:

The system responds automatically to SS2 selection if the configuration data item **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** is set to YES (set accordingly when creating an axis as of SIMOTION V4.2):
As a result, the user does not have to carry out cyclic evaluation of whether SS2 is selected.
At the start of the OFF3 ramp, TO alarm 50023 (drive makes transition to autonomous state, default local response = OPEN_POSITION_CONTROL) is signaled, switching the axis to follow-up mode automatically.
As long as SS2 is selected, the position controller enable and the following error monitoring therefore remain inactive and no more motion commands are effective. The drive enables can be canceled with the **_disableaxis()** user command, for example.
Automatic follow-up mode is only activated for the axis concerned in each case. For a possible synchronous operation line-up, the corresponding responses must be converted in the application.

After SS2 has been deselected, the position controller enable has to be reset with the **_enableaxis()** user command, for example, so that the axis can be traversed again.

User response with SIMOTION V4.1 or TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation=COMPATIBILITY_MODE:

When SS2 is selected (status information SS2_ACTIVE=TRUE), cancel the current motion on the axis and switch the position setpoint to follow-up mode, e.g. using the **_stop()** command at velocity = 0 and movingMode=SPEED_CONTROLLED. This prevents the enables for the drive from having to be canceled (as a result of alarm 50102, following error monitoring, for example).

Deselecting SS2 (status information SS2_ACTIVE=FALSE) terminates standstill monitoring in the drive. If no other safety function is active, the axis can be traversed again without the drive enables having to be reset.

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safe Operating Stop (SOS)As of SIMOTION V4.1

When SOS is selected, the drive continues to follow the setpoint interface and activates the standstill monitoring after the delay time configured in the drive.

Response in the user program:

When SOS is selected (status information SOS_SELECTED=TRUE)), bring the drive to a standstill within the time parameterized in p9551/p9531 and keep it in this state.

When SOS is deselected, the standstill monitoring in the drive will be deactivated; the drive can traverse again without restriction provided no other safety function is active.

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safely-Limited Speed (SLS)As of SIMOTION V4.1

When SLS is selected, the drive continues to follow the setpoint interface and activates the maximum velocity monitoring after the delay time configured in the drive.

When using SLS, ensure that the checkbox **SLS limit value absolute** is selected in the safety dialog.

Response in the user program:

When SLS is selected (status information SLS_SELECTED), the velocity must be reduced to the defined maximum velocity (system variable **driveData.safety.slsSpeedLimitSelected**). For this purpose, ensure that the application observes the distance to the actual limit value monitored

in the drive. Reduction either on the drive side using the p9533 parameter or using a programming reduction factor.

Note

It should be noted that updating the system variables with the status information and the defined maximum velocity is not necessarily simultaneous. Two cases must therefore be allowed for in the user program:

1. SLS is selected in the drive
   The system variable with the status information reports this.
   The system variable **~.slsSpeedLimitSelected** still includes the maximum velocity of the axis and not the limited velocity.
2. SLS is deselected in the drive.
   The system variable with the status information still reports SLS selected.
   The system variable **~.slsSpeedLimitSelected** already contains the maximum velocity of the axis.

Please also see the note regarding the reduction factor for **~.slsSpeedLimitSelected** in Chapter Display of SINAMICS Safety Integrated Functions in SIMOTION (Page 3087)

The velocity limit value monitored in the drive can be changed by switching between the SLS levels. Therefore, the system variable **~.safeSpeedLimit** should be continuously monitored for changes while SLS is selected.

In the user program, the activation of velocity reduction or the change to the reduced speed must be cyclically checked and the current velocity setpoint must be compared with the safe maximum velocity.

A detailed procedure can be found in the table below.

Table 4-380 User responses for SLS

	SLS_{new} selected	**SLS active and new SLS speed activated**	
		SLS_{new} is greater than SLS_{old}	**SLS_{new} is less than SLS_{old}**
Production speed is less than the limited speed SLSnew	No response necessary	If production speed is greater than SLS_{old}, increase to production speed is possible	No response necessary
Production speed is greater than the limited speed SLSnew	Deceleration to limited speed SLS_{new} required	Increase to SLS_{new} is permissible	Deceleration to limited speed SLS_{new} required

Production speed = set velocity before selecting a safety function

Note

For active velocity monitoring on a following axis (synchronous coupling and with a fixed gear or coupling via MotionIn interface), the speed of the master axis is reduced or, if this is not possible, synchronous operation during the reduction is canceled.

Compatibility mode with SIDB

The defined maximum speed is saved in the system variable **driveData.driveSafetyExtendedFunctionsInfoData.safeSpeedLimit**.

Please also see the note regarding the reduction factor for **~.safeSpeedLimit** in Chapter Display of SINAMICS Safety Integrated Functions in SIMOTION (Page 3095)

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safe Direction (SDI)As of SIMOTION V4.3

After selecting SDI (status information SDI_P_SELECTED or SDI_N_SELECTED), the motion may only be performed in the enabled direction.

Response in the user program:

If the direction of motion is incorrect, the motion must be stopped via the SIMOTION user program and it must be ensured that the motion is only performed in the enabled direction.

See also

Safety status information (part of the DSDB) (Page 3088)

Safety status information (part of the SIDB) (Page 3093)

Safe Position (SP)

Function SP can be selected optionally with reference to an absolute position. Use of SLP with transfer of the safe position SP corresponds to using SP with reference to an absolute position.

To update the relevant status bits in DSDB, drive parameter p9501.27 must be set.

Using SP with a safe absolute position

As for SLP (Page 3104), safe homing of the drive is required.

The position transferred via PROFIsafe can be used in the F-CPU to evaluate the position.

Using SP without a safe absolute position

The position transferred via PROFIsafe can only be used in the F-CPU to ascertain the speed of the drive.

See also *SINAMICS S120 Safety Integrated* Function Manual.

Synchronous transfer of safe position values (SP)

An isochronous transfer of safe positions (isochronous PROFIsafe telegram) is not possible with SIMOTION.

Safely-Limited Position (SLP)As of SIMOTION V4.4

Monitor the absolute traversing range of an axis using SLP. You may define a maximum of two traversing ranges in the drive. The active traversing range is displayed in SLP_LIMIT_BIT0_SELECTED of the system variable **driveData.safety.status1**.

The motion limitation when SLP function is activated must be performed by the user program. SLP activation is displayed in SLP_SELECTED of system variable **driveData.safety.status1**.

To update the status bits relevant for SLP in the DSDB, drive parameter p9501.27 must be set.

Requirement

The safe limited position can only be monitored if the following conditions are met:

- The drive must be homed at standstill while the machine is being commissioned.
- User agreement must be set once during commissioning after homing.
  Exception: It is reset by the user or by a positional tolerance violation in the drive.
- The SLP acceptance test must be performed.

Status **REF_REQUESTED** of the system variable **driveData.safety.status1** indicates whether the drive has been homed and/or whether homing has been requested by the drive. Homing must be performed again, for example, after the CU has been started up.

Note

Status of user agreement

The DSDB does not indicate whether user agreement is lost after successful homing of the drive. Only the S_FAULT bit is set. Evaluation via acyclic read requests may take too long to recognize this state and trigger a response in the user program if SLP is selected or already active.

Make sure a suitable response is implemented in the F-CPU user program in this case. For example, evaluate the "safely homed" bit in the PROFIsafe telegram. For example, trigger SOS if the drive is not safely homed.

Homing and user agreement

Homing must be performed once and user agreement set during commissioning. This can be done in the expert list of the drive or via an HMI.

Therefore home the TO axis and set the user agreement via an HMI:

1. Home the TO axis to a position that you can check in situ and make the axis remain stationary at this position.
2. Transfer the current position of the axis to the drive and apply the home position at standstill (two acyclic **_writeDriveParameter()** requests).
3. Wait for at least five safety cycles, then read the actual position of the drive (r9708[0]) and transfer it to the HMI.
4. Check whether the position of the TO axis is the same as the drive position and that this position corresponds to the machine position.

5. If the positions in step 4. are equal, set the user agreement with a **_writeDriveMultiParameter()** command.

   Note

   If the command is executed and user agreement is active, user agreement will be reset.

6. Now perform the SLP acceptance test.

In the Internet, you will find a program example for homing and setting user agreement as an FAQ (https://support.industry.siemens.com/cs/ww/en/view/100778695).

Violation of the traversing range when the SLP Retraction function is activated

If an SLP limit is exceeded, a safety error is displayed on the drive. A safe stop response is triggered on the drive as error response.

The **INTERNAL_EVENT** in the system variable **driveData.safety.status1** displays the safety error on the Axis technology object.

Retract the axis after an SLP violation as follows:

1. Activate the SDI function in the opposite direction to the violated direction of travel. Depending on the application, implement further safety requirements. For example, additionally activate the lowest SLS level and/or ensure that the drive can only be retracted by the F-CPU when a button is permanently pressed.
2. Deactivate SLP as soon as SDI is active.
3. Acknowledge the safety errors.
4. Travel the axis manually back into the valid SLP traversing range.
5. Activate the SLP function.
6. Deactivate the SDI function as soon as SLP is active.

Safe Cam (SCA)

With SCA, you can define safe cams that are available for evaluation in the F-CPU.

To update the status bits relevant for SCA in the DSDB, drive parameter p9501.27 must be set. With safe cams, for example, other safety functions can be selected in the F-CPU for defined positions.

Requirement

- Use of PROFIsafe telegram 903
- The drive must be homed safely at standstill while the machine is being commissioned. This must be performed as for SLP (Page 3104).

SLA - Safely-Limited Acceleration (as of SIMOTION V5.2)

When SLA is selected, the drive continues to follow the setpoint interface and activates the maximum acceleration monitoring in the drive.

The activation of **SLA** is displayed in **SLA_SELECTED** of the **driveData.safety.status1** system variable.

Response in the user program

For selection, and so active SLA (**SLA_SELECTED** status information), the acceleration for traversal commands must be reduced to the defined maximum acceleration of the drive. For this purpose, ensure that the application observes the distance to the actual limit value monitored in the drive.

New functions as of SINAMICS firmware V5.2

As of SINAMICS FW V5.2, SLA has been extended with fine resolution and acceleration filtering. You can parameterize the acceleration limit value and the stop reaction on SLA, and the filter time as of V5.2.

If the determination of the acceleration results in very noisy signals, the drive cannot reasonably monitor the acceleration. Remedy: In this case, increase the **"SLA filter time" (p9576)**. Note that **SLA** reacts with a delay when you increase the filter time.

Transmission via SIC

Once **SLA** has been parameterized and selected, the monitoring results are also transmitted in **SIC** in status word **S_ZSW1B.8** (Section "Safety Info Channel and Safety Control Channel"). You will find this status word in telegrams 700 and 701.

4.8.5.9 Safety brake test as of SIMOTION V4.4

As of V4.4, SIMOTION support the SINAMICS safe brake test. It is triggered by a SIMOTION command from the Axis technology object. During the SBT, the drive no longer follows the controller setpoint.

During the test, the load torque applied at the motor is measured once. The test torque is then increased n times depending on the direction and the number of brakes, maintained for a period of time and then reduced again.

The brake test can be performed for a maximum of two brakes in one test cycle. One internal and one external or two external brakes are possible if there are two brakes. Each brake can be tested against two different holding torques.

Requirements for the safe brake test

- The axis must be enabled with the **_enableAxis()** command in mode **enableMode**=POWER with **servoControlMode**=INACTIVE.
- All brakes present on the motor must be open.

Perform the safe brake test for an internal brake as follows:

1. Start the test by calling the **_executeAxisSafetySbt()** function in the user program.
2. Monitor the status if necessary via the **driveData.safety.sbtState** system variable.
3. Wait until the function has finished and then check the return value of the function.

Perform the safe brake test as follows if there is at least one external brake:

1. Start the test by calling the **_executeAxisSafetySbt()** function in the user program.
2. If there are two external brakes:
   Check the status **SBT_BT_BRAKE2** of the system variable **driveData.safety.status2** to see whether brake 1 or 2 is tested.
3. Check the status **SBT_BT_BRAKE_CLOSE_REQ** of system variable **driveData.safety.status2** cyclically in the user program. Open or close the external brake, depending on the status.
4. Report the current status of the external brake to the system using the **_setAxisSafetySbtExtBrakeState()** function.
   After a change of status of **SBT_BT_BRAKE_CLOSE_REQ**, the feedback must be provided within 11 seconds.
5. Monitor the status if necessary via the **driveData.safety.sbtState** system variable.
6. Repeat steps 2 to 5 until the **_executeAxisSafetySbt()** function has ended and then check the return value of the function.

Note

External brakes

Take into account the brake opening and closing times for status feedback.

4.8.5.10 Pulse enable evaluation (when standard settings are deactivated)

This chapter is only applicable if standard settings such as **Use symbolic assignment** have been deactivated.

For *SINAMICS Safety Integrated Functions* support, the status of the pulse enable of the drive on the Axis technology object must be evaluated. This status information is transferred to various positions in the telegram depending on the telegram type.
This means the **driveControlConfig.pulsesEnabled.pzdNumber** and

driveControlConfig.pulsesEnabled.bitNumber configuration data as appropriate for the telegram configuration must be set as follows:

- Standard telegram 2-6
  pzd = 4; bit = 10 (ZSW2; "Enable pulses")
  (possible as of SINAMICS V2.6)
- Ten SIEMENS telegrams
  pzd = 5; bit = 13 (message word; "Enable pulses")
- Non-assigned telegrams
  The user has sole responsibility for ensuring that both of the configuration data settings referred to are set according to the position of the pulse enable status bit within the telegram.

Note

- Standard telegram 1 cannot be used to support safety functions SS1 and SS2, since it does not transmit the status of the pulse enable.
- Depending on the telegram used, set p2038 as follows:
  - For standard telegrams in the drive p2038=0
  - For SIEMENS telegrams in the drive p2038=1

4.8.5.11 Messages and alarms

The messages are displayed as technological alarms of the *SINAMICS Safety Integrated Extended Functions* on the SIMOTION Axis technology object.

Technological alarms:

- **Alarm 50023: Drive makes transition to autonomous state (as of SIMOTION V4.2)**
  If the drive behaves autonomously, e.g. braking on the OFF3 ramp in response to an SS2 selection, alarm 50023 is signaled.
- **Alarm 50201: Safety alarm in the drive (see SINAMICS Safety Message Buffer)**
  A new event is present in the safety message buffer of the drive.

Note

Acknowledgment of alarm 50201

Earlier than SIMOTION V4.4:
Then alarm can only be acknowledged permanently if there are no longer any entries in the safety message buffer.

As of SIMOTION V4.4 (with DSDB):
Then alarm can only be acknowledged permanently if neither S_FAULT_PRESENT nor INTERNAL_EVENT is set.

- **Alarm 50202: SINAMICS Safety Integrated Extended Function is selected**
  Alarm 50202 is signaled if the safety functions SS2/SOS/SLS are selected and, as of V4.4, SDI/SLP as well.
  Exceptions:
  - On transition to SOS with SS2 active (automatic transition or explicit selection), no new 50202 alarm is activated.
  - If the *SINAMICS Safety Integrated Extended Functions* are activated using PROFIsafe, alarm 50202 is generated after a SINAMICS reset without a safety function being selected.
  - If the *SINAMICS Safety Integrated Extended Functions* are activated using PROFIsafe, alarm 50202 is generated if the SIMOTION CPU or F-CPU switches to STOP.

  The alarm is used only for display in the SCOUT or HMI.
  User responses should reference system variable **driveData.safety.status1/ driveData.safety.status2** or **driveData.driveSafetyExtendedFunctionsInfoData.state**.
- **Alarm 50203: SINAMICS Safety Integrated Extended Function is deselected**
  Alarm 50203 is signaled if the last active safety function SS2/SOS/SLS is deselected and, as of V4.4, SDI/SLP as well.
  Exception:
  When controlling the *SINAMICS Safety Integrated Extended Functions* via PROFIsafe, the PROFIsafe driver is acknowledged (user action on the F-CPU side as an additional prerequisite for full drive enable (drive parameter r46.8)), alarm 50203 is generated without a safety function being deselected.
  The alarm is used only for display in the SCOUT or HMI.
  User responses should reference system variable **driveData.safety.status1/ driveData.safety.status2** or **driveData.driveSafetyExtendedFunctionsInfoData.state**.
- **Alarm 50209: Error on brake test**
  Alarm 50209 is signaled if an error occurs when the SBT brake test is performed.

Note

Alarm 50023 only becomes active on the Axis technology object if configuration data **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** is set to YES.

Alarms 50201, 50202 and 50203 are only displayed on the TO axis when DSDB and SIDB are activated.

No safety alarm is triggered when STO and SS1 are selected. Alarm 20005 reason 4 is signaled if the pulses are deleted by the drive. There is no alarm to signal that the functions have been deselected.

Illustration of safety alarm behavior (50202, 50203) based on a pulse diagram

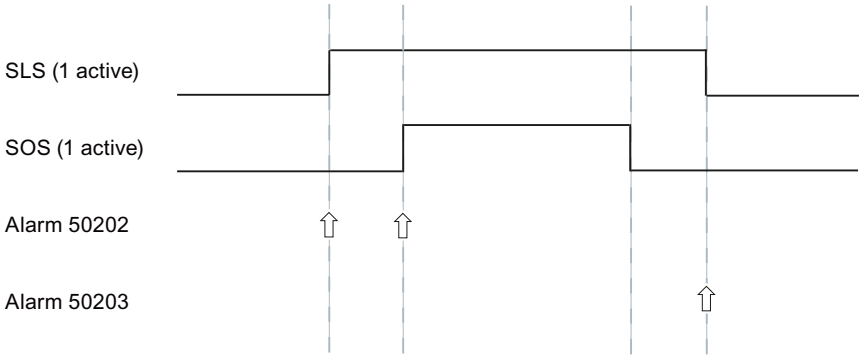


Figure 4-898 Pulse diagram illustrating safety alarm behavior

4.8.5.12 SINAMICS Safety Message Buffer

Errors in the safety function in the drive or stop responses that are triggered by the monitoring functions initiate messages.

Messages for the *SINAMICS Safety Integrated Basic Functions* are displayed in the drive's message buffer (r947).

Messages for the *SINAMICS Safety Integrated Extended Functions* are displayed in the safety message buffer (r9747).

Table 4-381 Basic structure of the message buffer

Parameter	Meaning
r944	Message buffer change count
r947 [0...63]	Message code
r949 [0...63]	Additional information
r952	Messages counter

Table 4-382 Basic structure of the safety message buffer

Parameter	Meaning
r9744	Message buffer change count
r9747 [0...63]	Message code
r9749 [0...63]	Additional information
r9752	Messages counter

Acknowledge

Errors in the safety functions or stop responses may only be acknowledged on a fail-safe basis. A distinction must be made here between the *Basic Functions* and *Extended Functions*.

Basic Functions: Acknowledgment via two-channel selection/deselection by STO.

Extended Functions: Acknowledgment of safety PLC via the PROFIsafe signal Internal_Event_Acknowledge - Bit 7 of the PROFIsafe control word or via a terminal on TM54F or F-DI OnBoard for D410-2.

With extended alarm acknowledgment, the messages can be acknowledged via the acknowledgment mechanism of the *Basic Functions*.

Note

Limitation for SINAMICS V2.5

Fail-safe acknowledgment via PROFIsafe bit or a fail-safe digital input following violation of SLS limit value is only possible if STOP C (p9563=2) has been configured as a fault response. For other stop responses, an acknowledgment is only possible by means of Power Off/Power On.

For further information, see the *SINAMICS Safety Integrated* Function Manual.

Display and evaluate an error message

An entry in the safety message buffer is displayed by S_FAULT_PRESENT in the system variable **driveData.safety.status1** (Bit 30).

The occurrence of a safety STOP response is displayed by INTERNAL_EVENT in the system variable **driveData.safety.status1** (Bit 31).

If this bit is set, the **50201 Safety Alarm in the drive** alarm will be triggered in SIMOTION.

The **_readDriveFaults()** system function can be used to read out the message code.

As of SIMOTION V4.2 it is possible to fetch data from the safety message buffer directly by setting faultType=SAFETY (drive parameter p9747). The additional information for the message code can be read out by the user program using the system function **_readDriveParameter()** from drive parameter r9749.

Compatibility mode with SIDB

An entry in the safety message buffer is displayed by S_FAULT_PRESENT in the system variable **driveData.driveSafetyExtendedFunctionsInfoData.state** (Bit 15). If this bit is set, the **50201 Safety Alarm in the drive** alarm will be triggered in SIMOTION.

4.8.5.13 Isochronous PROFIBUS operation

PROFIsafe data must be transferred isochronously.
When PROFIsafe data is transferred isochronously, additional configuration and programming steps are required or are useful.

Essential settings for isochronous PROFIsafe operation

1. Synchronization of the SIMOTION CPU with the F-CPU
   In the I-slave F-proxy and I-device F-proxy topologies, synchronize the connection from the SIMOTION user program to the **_enableDpInterfaceSynchronizationMode()** system function.
2. Consistent process image partition for the safety program
   Make sure that the update of the process image partition for isochronous operation is not performed during the processing of the safety program.
   An example can be found in Chapter *F-CPU: Isochronous operation and safety operation* of the FAQ Actuating internal drive safety functions via SIMOTION and PROFINET with PROFIsafe (https://support.industry.siemens.com/cs/ww/en/view/50207350).

Additional recommended settings for isochronous PROFIsafe operation

1. Check the "synchronous" status on the SIMOTION controller
   Check for error-free synchronization through an I/O status query of the PROFIsafe PZD ("synchronous").
2. Call the safety program on the F-CPU in a multiple of the SI monitoring cycle
   Select the cycle of the cyclic interrupt in which the safety program is called (typically OB35) as a multiple of the SI cycle (drive parameter p9500).
   In this way, you avoid problems during startup (SINAMICS fault 1611/1711 with fault value 6165).

4.8.5.14 Restrictions after switching from compatibility mode (SIDB) to standard (DSDB).

The following points should be noted when switching from compatibility mode (SIDB) to standard (DSDB):

- Safe Speed Monitor, **SSM**
  Safe speed monitoring via the SSM bit is no longer available after switching to standard (DSDB).
- System variable **safeSpeedLimit**
  The system variable is only provided after switching to standard (DSDB) if the following conditions are met:
  - The *SINAMICS Safety Integrated Extended Functions* are configured.
  - The **SLS limit value absolute** option is selected.
- Alarm 50201
  The alarm is only triggered by S_FAULT_PRESENT if the SIMOTION Runtime is less than V4.4.
  As of SIMOTION Runtime V4.4, the alarm can also be triggered by INTERNAL_EVENT.
  See also SINAMICS Safety Message Buffer (Page 3110).
- Always evaluate the pulses enabled bit
  The configuration data item **TypeOfAxis.DriveControlConfig.pulsesEnabledEvaluation** must be set to YES if the DSDB is used and correct transfer of the pulses-enabled bits must be ensured.

See also

Axis configuration (Page 3088)

Safety channel and safety channel types (when symbolic assignment is deactivated) (Page 3085)

Overview - Support of SINAMICS Safety Integrated Functions on the Axis technology object (Page 3079)

Safety status information (part of the DSDB) (Page 3088)

4.8.6 Part II Hydraulic Functionality

4.8.6.1 Hydraulic functionality overview

The hydraulic functionality is contained in the **Axis technology object (TO)**.

Like the electric axis, the hydraulic axis can be configured with the following technologies:

- Speed-controlled axis
- Positioning axis
- Synchronous axis

The user view of the electric and hydraulic axes is maintained together, to the extent possible.

Example:

- Motion commands
- Axis settings

Important differences compared to the electric axis include:

- Allowance for a valve characteristic curve
- Special compensation
- A valve can be switched between more than one axis
- No lower-level velocity limiting or torque control in the actuator / drive
- Separate actuators/valves for flow (Q valve) and force/pressure (P valve), if necessary

The hydraulic functionality is explained in the following sections, building on the description of the axis.

See also

Overview of axis technologies (Page 2861)

General information about axes (Page 2858)

4.8.7 Fundamentals of hydraulic functionality

4.8.7.1 Axis settings / drive assignment

Overview of axis settings / drive assignment

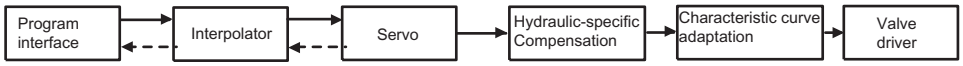


Figure 4-899 Overview of motion control for axes with hydraulic functionality

Valve types:

- **Q valve** (valve for closed-loop control of volumetric flow, path valve)
  Hydraulic valve for control of direction and volume of a material flow, suitable for closed-loop velocity control
- **PQ valve**
  Special Q valve suitable for force, position, velocity, and pressure control
- **P valve** (pressure limiting valve)
  Valve used for limiting the system pressure, suitable for protecting a hydraulic system against excessive pressure (overpressure protection)

Setting as a real axis with hydraulic functionality

The manipulated variable values for the final controlling elements (valves) are defined as analog or direct values.

Options are:

- Analog outputs on the C2xx
- Analog outputs in the I/O area
- Analog outputs on a PROFIBUS module, e.g. SINUMERIK ADI4, SIMATIC IM174
  With SIMODRIVE ADI4 and SIMATIC IM174, PROFIdrive profile standard message frame 3 must be used.

A specific enable signal is available for each final controlling element (see the drive enable signal for electric axes).

The enable signal for the Q valve is set/reset via the qOutputEnable parameter in the **_enableQFAxis()/_disableQFAxis()** command. The status of the enable signal is indicated in the **actorMonitoring.driveState** and **actorMonitoring.power** system variables. When the hydraulic output is connected to the IM174, these system variables are formed using the bits returned in the status word. The status of the Q output is displayed in **actorMonitoring.qOutputState**.

The enable signal for the P valve is set/reset via the fOutputEnable parameter in the **_enableQFAxis()/_disableQFAxis()** command. The status of the enable signal for the P valve is displayed with the **actorMonitoring.fOutputEnable** system variable. The status of the P output is displayed in **actorMonitoring.fOutputState**.

When defining an axis as an axis with hydraulic functionality, it is possible to assign several axes to a final controlling element during configuration. During runtime, they are assigned using the **_enableQFAxis()/_disable QFAxis()** enable/disable commands.

A real axis with hydraulic functionality has manipulated variables for the flow rate (Q, Q valve) and, if required, a separate manipulated variable for direct force/pressure limiting or force/pressure control (F, P valve).

On axes with a P valve, the technological commands and system variables for force/pressure limiting can be used to specify the force and pressure.

The addresses in the I/O area for outputting the Q value and, if applicable, for outputting the F/P value can be set on the axis.

Note

You can find application examples for hydraulic axes under *FAQs > Technology* in the *SIMOTION Utilities & Applications* provided with SIMOTION SCOUT.

See also

Access to the same final controlling element from multiple axes (Page 3134)

Setting as a real axis with Q valve only

With hydraulic axes with Q valve only, the functions for traversing the axis, for velocity limitation, for force/pressure control, and for force/pressure limiting are available as with the electric axis.

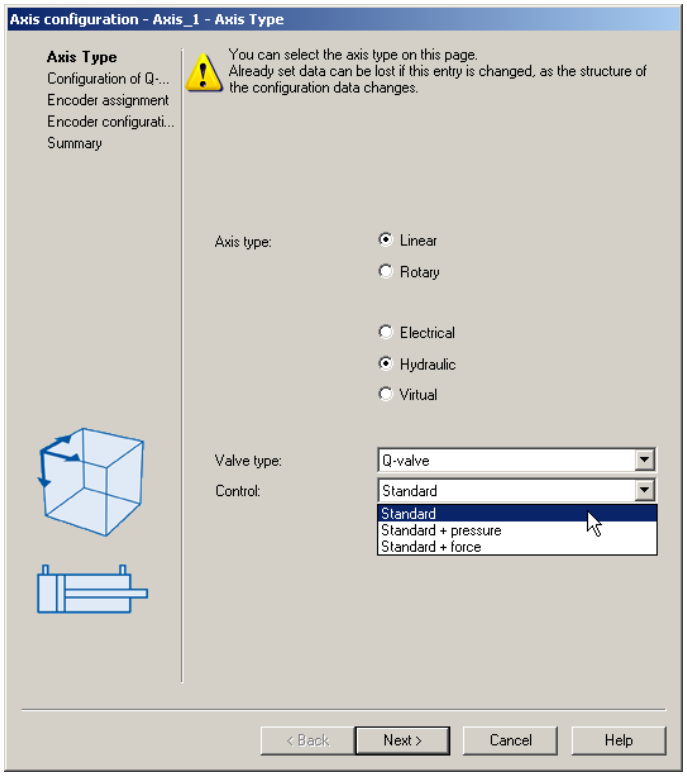


Figure 4-900 Setting the axis type

Table 4-383 Setting the control

Standard	Motion via Q valve
Standard+pressure	Motion and force/pressure control via Q valve Unit: Pascal, bar
Standard+force	Motion and force/pressure control via Q valve Unit: Newton

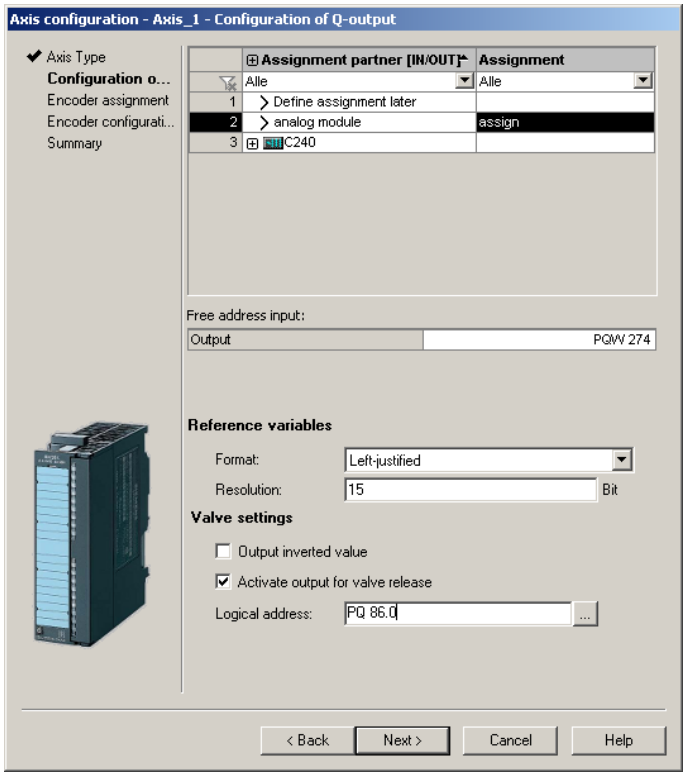


Figure 4-901 Example configuration of the Q output using analog output module

The bit resolution of the analog output module is set under Resolution.

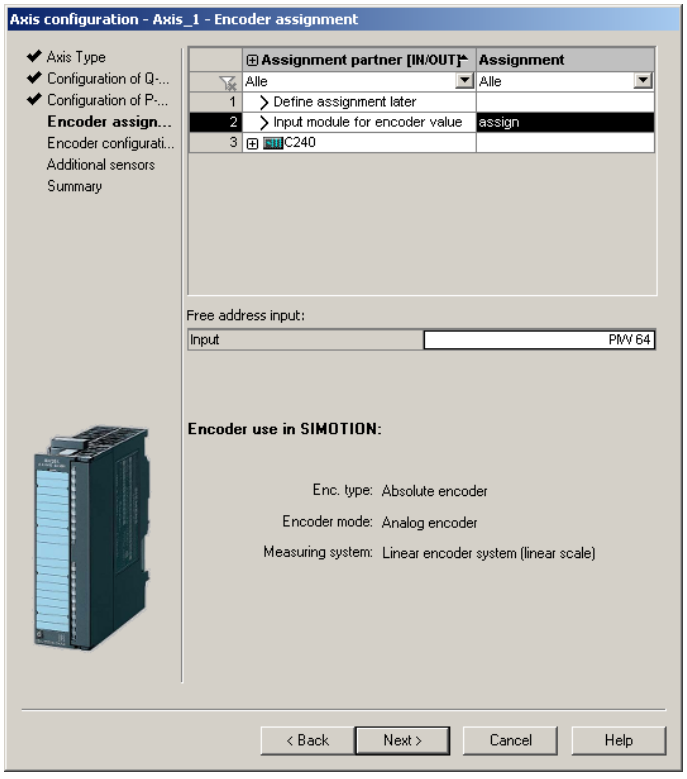


Figure 4-902 Example of configuration for the encoder assignment

The logical hardware address of the input module can be found in the HW Config in SCOUT.

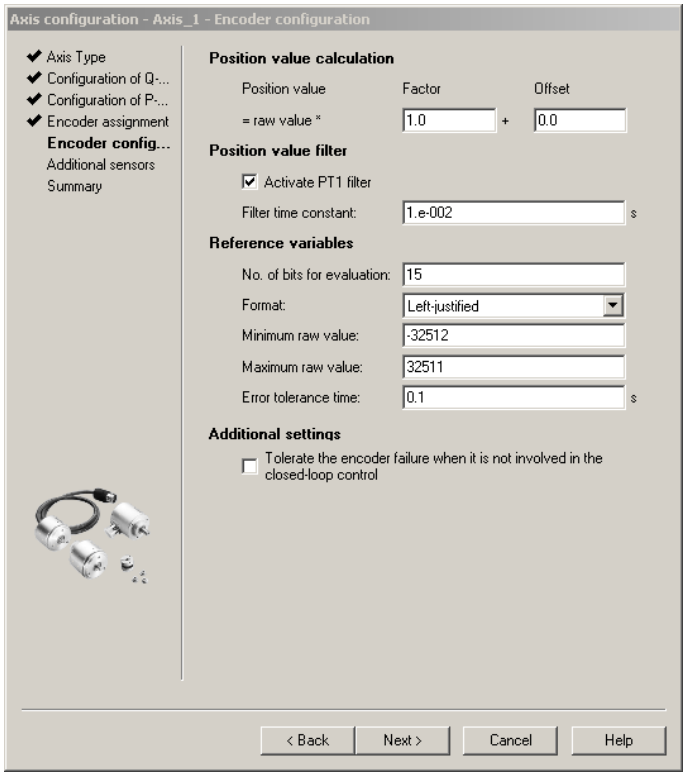


Figure 4-903 Example of configuration for the position value

The **Factor** (scaling factor) and **Offset** are used to convert the internal value into a physical variable that can be represented.

The value can be output as right-justified or left-justified within the word width (16 bits).

The minimum and maximum raw values are the limitations. If the measured actual value is outside these limits, technological alarm **50013: The permissible range limit has been exceeded** is issued and the internal actual value is set to the limit value.

The error tolerance time states how long an error can be pending before a technological alarm is set.

See also

Setting for the hydraulic axis type (Page 2869)

TypeOfAxis configuration data (Page 2870)

Setting as a real axis with Q valve + P valve/F output

With hydraulic axes with Q valve + P valve/F output, the functions for traversing the axis (Q valve) are available. In addition, a manipulated variable can be controlled and output on the P valve/F output.

The following variants are available:

- Two valves (P valve and Q valve)
- One valve with two connections (analog)
- One variable-capacity pump

On the Axis technology object, one analog controller output is configured and controlled for the Q valve (flow, velocity) and one for the P valve (force/pressure limiting).

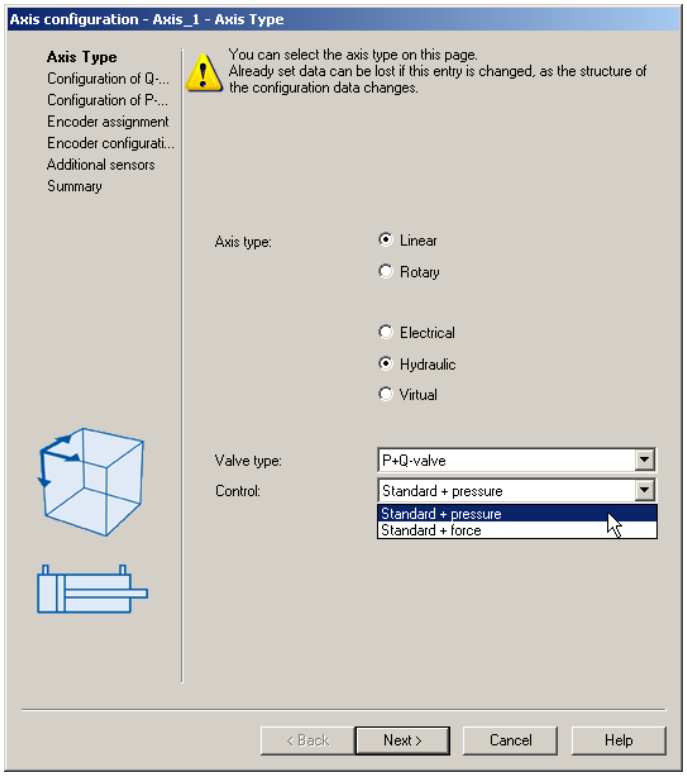


Figure 4-904 Setting the axis type

Table 4-384 Setting the control

Standard+pressure	Motion at the Q output and pressure limiting at the P valve/F output Unit: Pascal, bar
Standard+force	Motion at the Q output and force limiting at the P valve/F output Unit: Newton

Hydraulic axes with Q valve and P valve/F output do not have any pressure control (pressure control commands are rejected); in this case, the pressure limiting commands act on the P valve/F output. The pressure limiting value in the command is issued as a manipulated variable at the P valve/F output.

See also

Setting for the hydraulic axis type (Page 2869)

TypeOfAxis configuration data (Page 2870)

Setting an axis as a real axis with P valve only (V3.2 and higher)

At the P valve (F output of the axis), a time-related or actual-position-related profile or a manipulated variable can be output. This means that no position control, velocity control or force/pressure control takes place. There are no force/pressure sensors required, but they can be configured.

Position encoders cannot be configured.

You must configure a speed-controlled axis.

The following commands are possible on this axis:

- _resetAxis()
- _resetAxisError()
- _getStateOfAxisCommand()
- _bufferAxisCommandId()
- _removeBufferedAxisCommandId()
- _enableQFAxis()
- _disableQFAxis()

Commands to output a force/pressure limiting value or a force/pressure limiting profile can also be used:

- _enableForceLimitingValue()
- _enableTimeLockedForceLimitingProfile()
- _enableMotionInPositionLockedForceLimitingProfile()
- _disableForceLimiting()

Position encoders cannot be configured or activated.

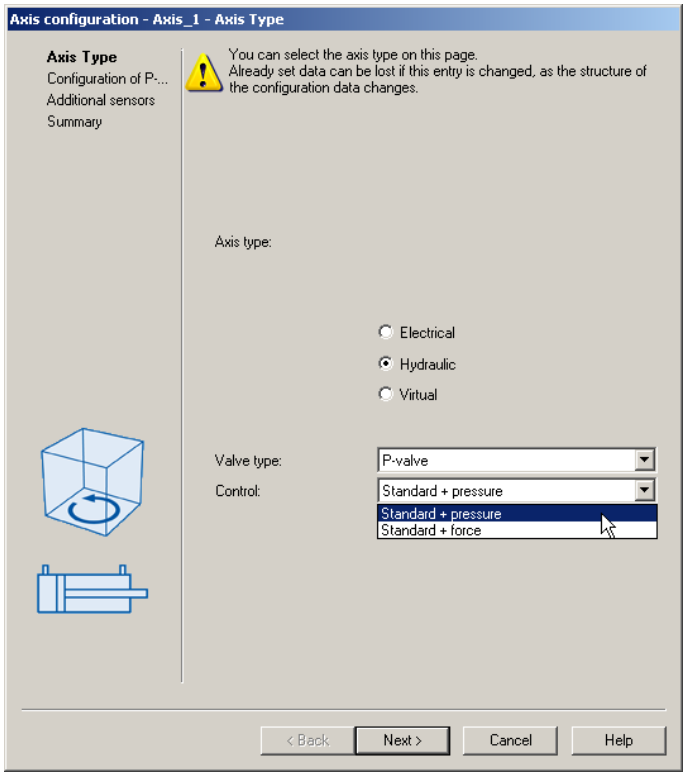


Figure 4-905 Creating an axis with P valve only

Table 4-385 Setting the control

Standard+pressure	Force/pressure control at the P valve/F output Unit: Pascal, bar
Standard+force	Force/pressure control at the P valve/F output Unit: Newton

See also

Overview of axis technologies (Page 2861)

Setting for the hydraulic axis type (Page 2869)

TypeOfAxis configuration data (Page 2870)

Setting an axis as a real hydraulic axis without a valve (axis simulation)

Hydraulic axes with manipulated variable output directly in the I/O area cannot be defined for axis simulation.

Axis simulation with hydraulic axes is only possible with a Q valve and with a digital drive interface or onboard C2xx.

For more information, see Setting as a real axis without drive (axis simulation) (Page 2899).

4.8.7.2 Input limits, technological limiting functions

Information about the system input limits and technological limitations is contained in Input limits, technological limiting functions (Page 2918).

4.8.7.3 Settings for axis and encoder mechanics

Note

With the hydraulic axis, not all of the setting options for the electric axis are required or displayed.

See also

Overview of setting options for axis and encoder mechanics (Page 2918)

4.8.7.4 Defaults

Information about default settings for system variables can be found in Defaults (Page 2922).

4.8.7.5 Homing

A short summary for homing, absolute encoder and incremental encoder is contained in Overview of homing (Page 2923).

4.8.7.6 Differential pressure measurement (V3.2 and higher)

The actual pressure value can be set as a pressure difference.

The pressure difference is implemented as a separate sensor type that determines the resulting differential pressure from two sensor-measured values using the following formula:

$F_{act} = (p_A \times A_A - p_B \times A_B) \times F_{factor}$
(F_{factor}: force factor)

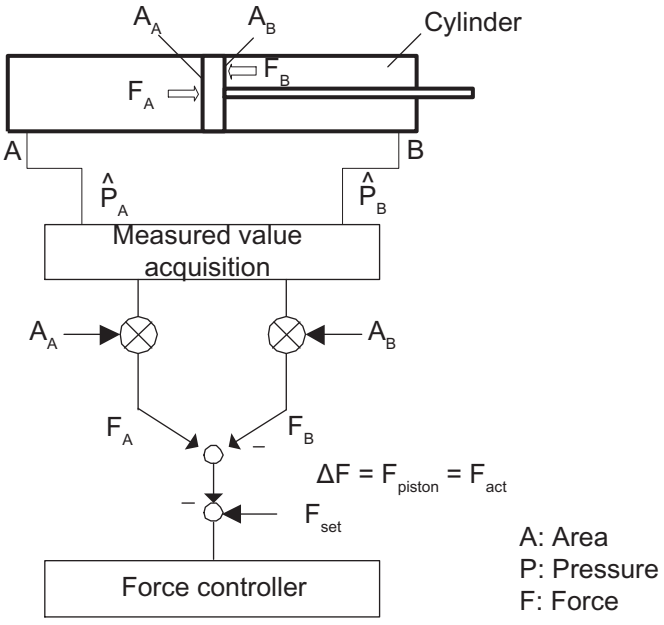


Figure 4-906 Example of a pressure difference measurement

Like the measuring sensors, the pressure difference is defined as force/pressure sensors.

Proceed as follows to specify the setting via the expert list:

- Configure at least two pressure sensors on the axis.
- In the expert list, increase the value of **TypeOfAxis.NumberOfAdditionalSensors.number** by 1.
- Set the other sensor as a "differential pressure sensor" with PRESSURE_DIFFERENCE_MEASUREMENT via the **TypeOfAxis.NumberOfAdditionalSensors.AdditionalSensor_n.additionalSensorType** configuration data element.
- The sensors whose values are being used and the individual factors are set in the elements of the **TypeOfAxis.NumberOfAdditionalSensors.AdditionalSensor_n.pressureDifferenceMeasurement** structure.

The pressure difference may be the pressure difference at a cylinder or another pressure difference. All sensors used for the pressure difference measurement must be configured on the same technology object.

The pressure difference measurement may also be applied to the electric axis.

If pressure difference measurement is used, force control is available in accordance with the function definition. In this case, the pressure sensors are also displayed in the force unit, dependent upon the system.

See also

Using the expert list for an axis (Page 3065)

4.8.7.7 Differential position measurement (V3.2 and higher)

Information on differential position measurement is contained in Differential position measurement (V3.2 and higher) (Page 2937)

4.8.7.8 Monitoring/limiting functions

The monitoring/limiting functions of the electrical axis are applied.

For the hydraulic axis, the pressure control and limiting functions can also be applied to the drive axis.

See also

Overview of monitoring/limiting functions (block diagram) (Page 2938)

4.8.7.9 Motion profiles

The motion profiles that can be applied are the same as for the electrical axis.

See also

Overview of traversing with user-defined motion and force/pressure profiles (Page 3040)

4.8.7.10 Hydraulic axis with position control/velocity control

Position control for setting a positioning axis with hydraulic functionality

Block diagram of the hydraulic axis with closed-loop control:

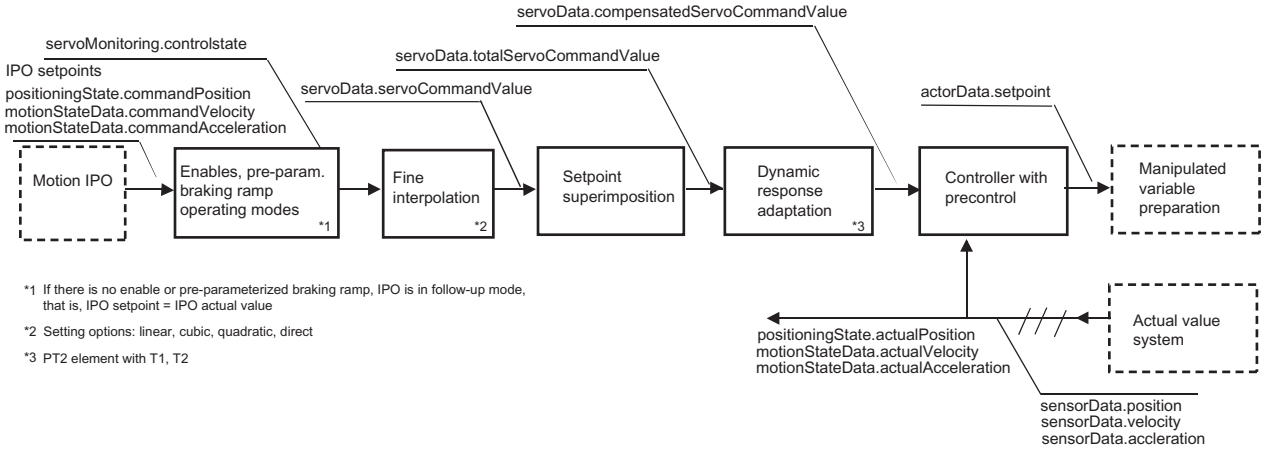


Figure 4-907 Overview of hydraulic axis with closed-loop control

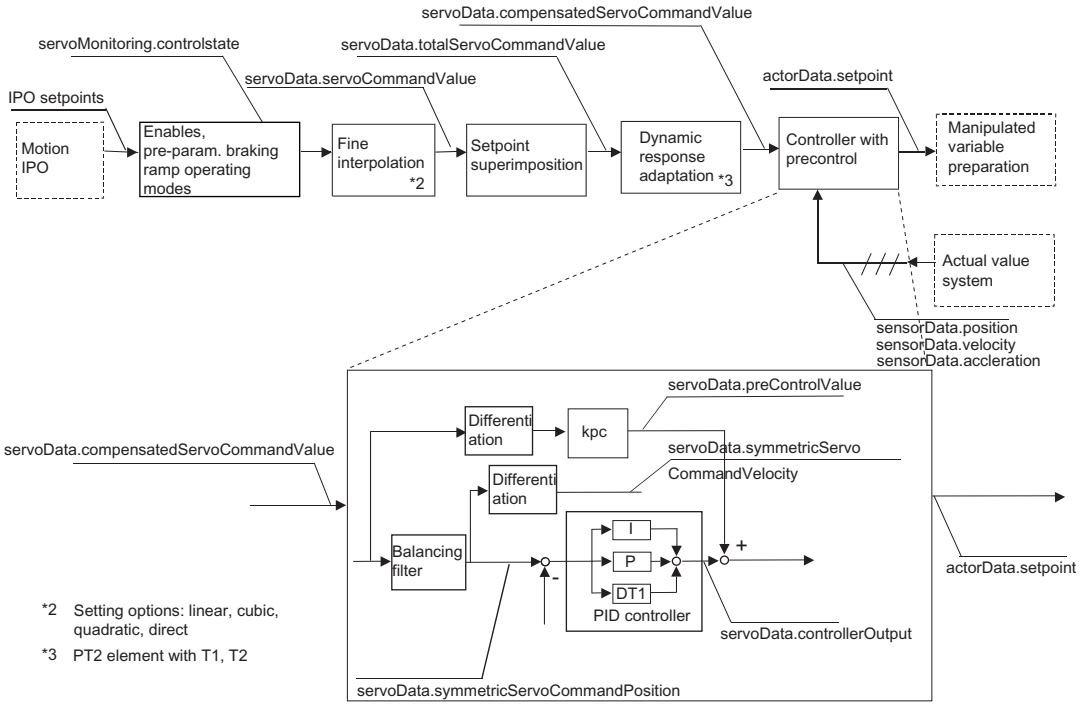


Figure 4-908 PID controller with precontrol

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

Remark

You can specify during configuration whether the D component of the controller refers to the control deviation or the actual value (in the **ControllerStruct.conType** configuration data element).

The dynamic behavior of the process is taken into account in the balancing filter.

With the hydraulic positioning axis, the dynamic response of the position control loop is specified in the **dynamicData.positionTimeConstant** configuration data element. The dynamic response of the process is specified in **dynamicQFData.qOutputTimeConstant**.

The hydraulic position axis can now be activated in non-position-controlled mode with the **_enableQFAxis()** command via the parameter **movingMode**= SPEED-CONTROLLED (as of V4.1 SP1).

See also

Overview of positioning axis with position control (Page 2949)

Position control (Page 2950)

Velocity controller when setting speed-controlled axis with hydraulic functionality

When an axis is set as a drive axis with hydraulic functionality, the velocity controller is calculated in SIMOTION.

With the electric drive axis, the velocity controller is included in the drive and the control specifies the speed setpoint.

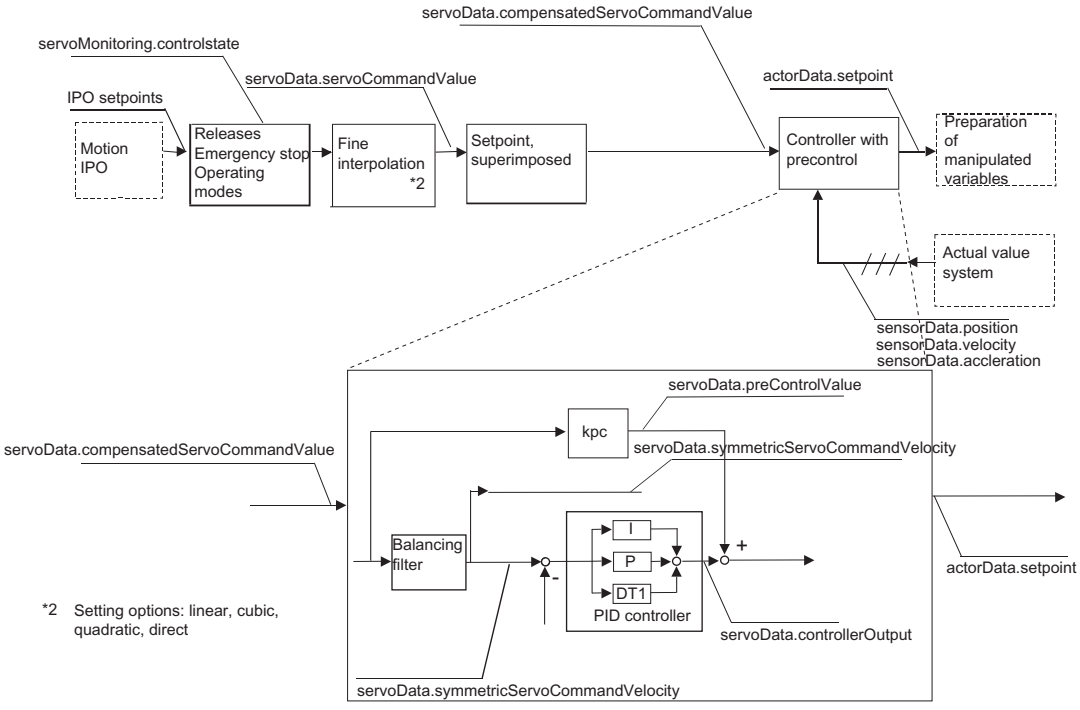


Figure 4-909 Block diagram of the velocity controller

Setting an axis as a drive axis with hydraulic functionality

Functionality:

- PID controller available as velocity controller
- Precontrol of the manipulated variable available
- Velocity monitoring available

If a controller is applied, the axis is traversed with closed-loop velocity control. If no controller is applied, the axis is traversed with open-loop velocity control.

Note

The closed-loop velocity-controlled hydraulic axis cannot be moved with open-loop velocity control via movingMode=SPEED_CONTROLLED.

The status of the velocity controller is displayed in the **servoControl.controlState** system variable.

Controller error monitoring is displayed in **servoMonitoring.controllerDifferenceError**.

The components **dynamicFollowingError**, **dynamicFollowingWarning**, **positioning**, and **standstill** in **servoMonitoring** are irrelevant.

With setting **ControllerStruct.conType** = DIRECT, the controller can be disconnected.

Monitoring

Data and statuses for the velocity controller on the axis are displayed in the **servoData** components.

See the list manuals for details.

For the velocity controller, the setpoints, actual values, and superimpositions refer to the velocity. Position-related information such as **actualPosition** or **symmetricServoCommandPosition** are irrelevant.

preControlValue displays the precontrol value.

In **servosettings**, the **additionalCommandValue** setting refers to the velocity.

Standstill monitoring is not available on the drive axis.

The standstill signal is available.

Remarks

Dynamic response adaptation is not active on the axis with hydraulic functionality.

Superimpositions are in effect.

With the hydraulic drive axis, the dynamic response of the velocity control loop is specified in the **dynamicQFData.velocityTimeConstant** configuration data element; the dynamic response of the process is specified in the **dynamicQFData.qOutputTimeConstant** configuration data element.

See also

Overview of positioning axis with position control (Page 2949)

Preparation of manipulated variables for axis with hydraulic functionality

Preparation of manipulated variables for axis with hydraulic functionality, Q output

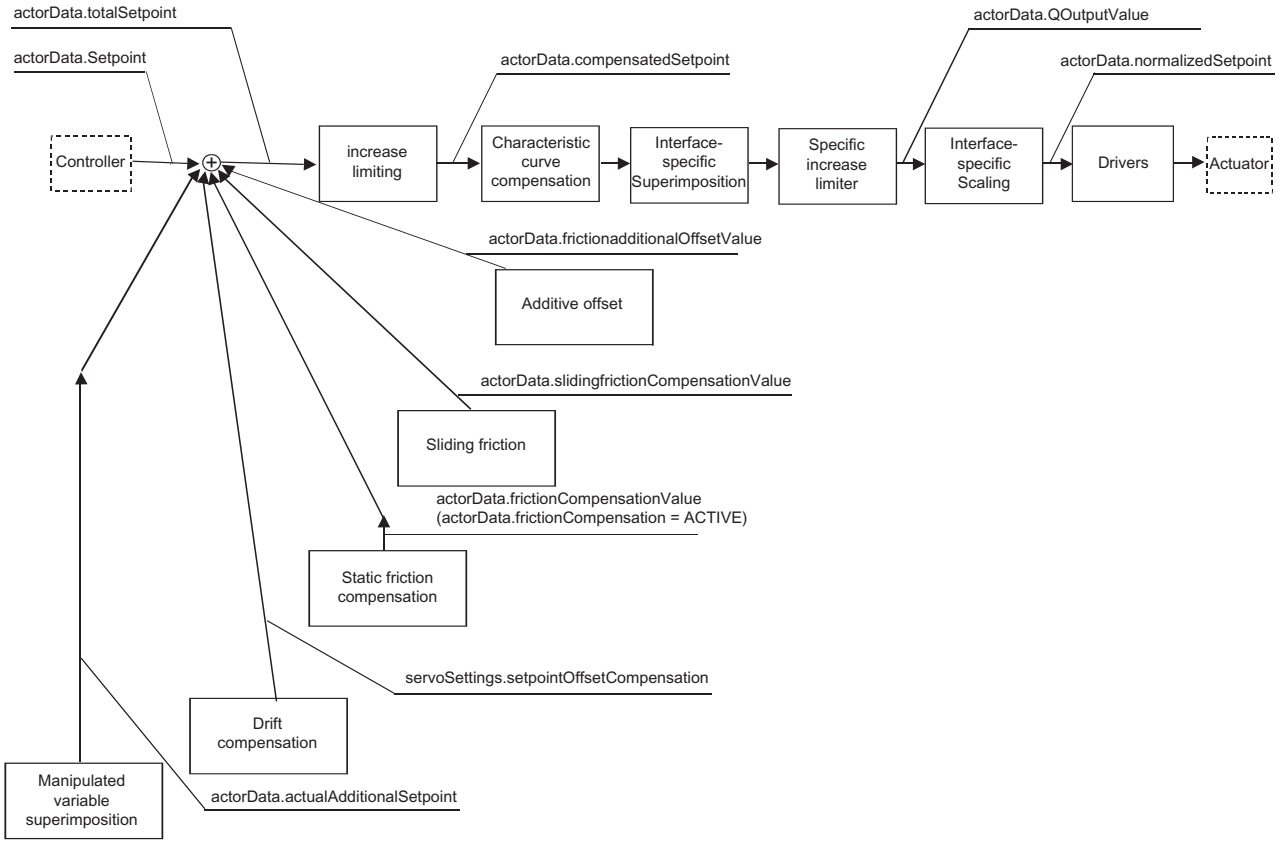


Figure 4-910 Preparation of manipulated variables for axis with hydraulic functionality, Q output

Preparation of manipulated variables for axis with hydraulic functionality, F output

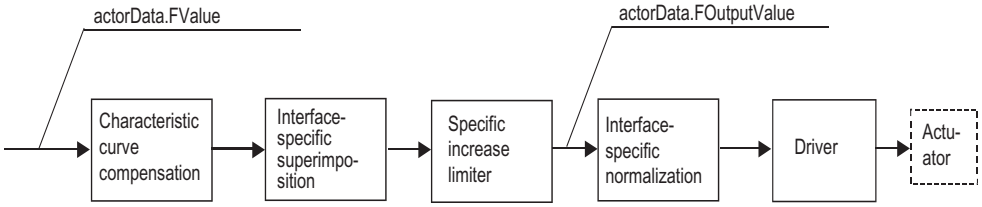


Figure 4-911 Preparation of manipulated variables for axis with hydraulic functionality, F output

Note

For hydraulic axes, a valve characteristic can be assigned to each output (P or Q). If this is not the case, a linear characteristic is used. Here, 100% corresponds to the limit value of the axis (**TypeOfAxis.MaxVelocity**, **TypeOfAxis.MaxForceCommandData**).

The interface-specific superimposition is specified as a percentage (%), e.g. for specifying manipulated variables when recording of the characteristic curve.

An increase limiting is specified in the command for output activation and for activation of the characteristic curve because the manipulated variable must be continuos on the valve, e.g. the increase limiting (ramp function) is used to prevent a step change. If the increase limiting is active, the I component is retained in the position controller or force/pressure controller.

The specific increase limiting after allowing for the valve characteristic is only in effect for the transition at **_setQFAxisFCharacteristics()**, **_setQFAxisQCharacteristics()**, and **_disableQFAxis()**, **_enableQFAxis()**. The increase limiting is specified in the command (default in **userDefaultQFAxis.maxDerivative**).

Note

If the RELEASE_DISABLE or OPEN_POSITION_CONTROL alarm responses occur, the setpoint or manipulated variable = 0 will be output. For hydraulic axes, the value 0 will be converted into an appropriate output value using the valve characteristic.

A velocity encoder must be present for speed-controlled operation. The control supports the following velocity encoders:

- Pulse encoder via SM335 or E510
- Analog velocity encoder via analog input modules

See also

Overview of positioning axis with position control (Page 2949)

Manipulated variable filtering (as of V4.1 SP1)

A man. var. filter can be set as a PT1 filter in the **setpointFilter** configuration data element. This filter acts after the controller and after the precontrol value and the additive manipulated variable value (additionalSetpoint) have been added.

A change in the filter data takes effect immediately.

Compensations that are active only on the axis with hydraulic functionality

A static compensation component (additive sliding friction) and a compensation component proportional to velocity (sliding friction) can be set on the axis with hydraulic functionality.

These settings are made in the Closed-Loop Control view in the project navigator under the axis. The Further Compensations tab appears when Expert mode is selected.

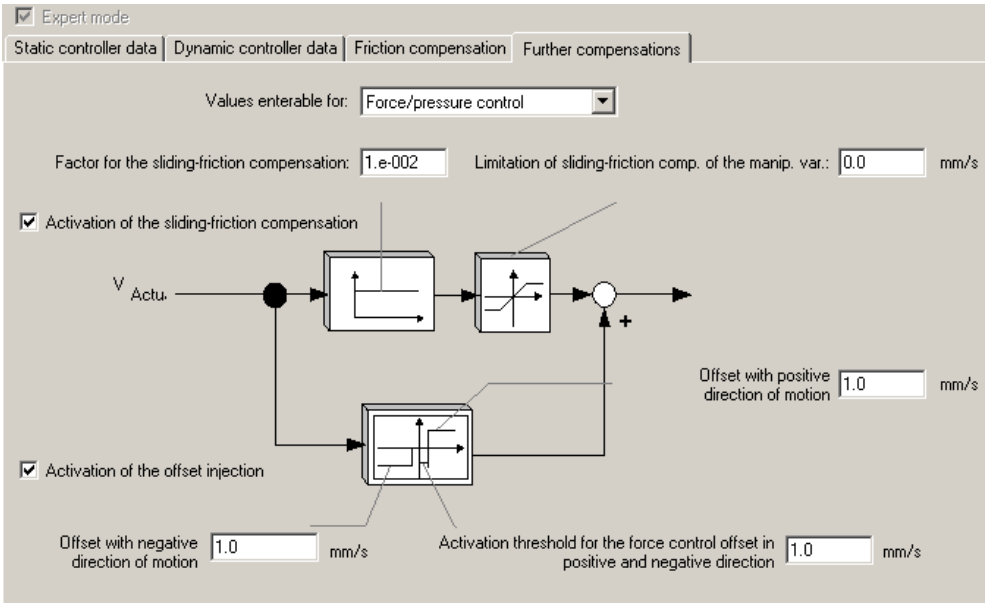


Figure 4-912 Sliding-friction compensation/additive sliding-friction compensation on an axis

Sliding-friction compensation

Sliding friction is applied relative to the velocity setpoint in axis motion via motion specification, and relative to the actual velocity value in axis motion via force/pressure specification.

The factor for sliding-friction compensation can be set specifically for axis motion via motion specification (**factorMotionControl**) and for axis motion via force/pressure specification (**factorForceControl**) in the structure elements for **SlidingFriction**.

The current value of the sliding-friction compensation is displayed in the **actorDataSlidingFrictionCompensationValue** system variable.

Additive sliding-friction compensation (offset application)

Additive sliding friction is applied relative to the velocity setpoint in axis motion via motion specification, and relative to the actual velocity value in axis motion via force/pressure specification.

The factor for **additive sliding friction** can be set specifically for axis motion via motion specification, and separately for each velocity direction, in factorMotionControlPositive and factorMotionControlNegative; it can be set for axis motion via force/pressure specification in factorForceControlPositive and factorForceControlNegative.

These factors are set in the structure elements for AdditionalOffset. The current value is set in **frictionAdditionalOffsetValue** (direction-dependent sliding-friction compensation value).

The current value for additive sliding friction is indicated in the **actorDataFrictionAdditionalOffsetvalue** system variable.

See also

Overview of positioning axis with position control (Page 2949)

Consideration of valve characteristic when specifying a hydraulic axis

The existing non-linearity between the technological manipulated variable, for example, an oil flow rate (Q) or a velocity or force/pressure value (F), and the manipulated variable value of the valve is mapped by means of a characteristic curve in the open-loop control system and is taken into account in calculating the manipulated variable value.

The valve characteristic is set using a cam. See the Technology Objects Synchronous Operation, Cam description of functions.

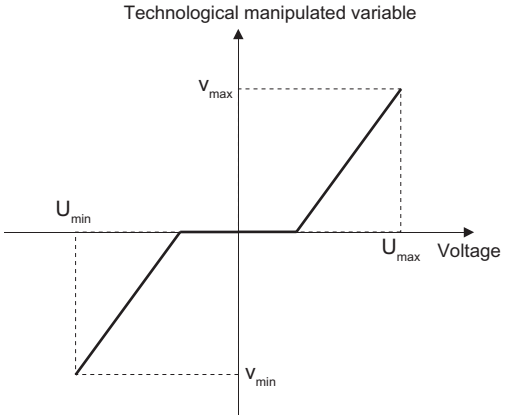


Figure 4-913 Characteristic curve setting for hydraulic axis

In the cams for the characteristic curves, the technological manipulated variable (velocity, pressure/force) is specified by means of the manipulated variable value of the final controlling element.

The valve characteristic curves are selected/switched using **_setQFAxisQCharacteristics()** or **_setQFAxisFCharacteristics()**. They can also be switched during the motion.

If several characteristic curves are interconnected with an axis, one characteristic curve must be explicitly selected with the command.

If only one valve characteristic curve is available for an axis, then a selection command is not required.

If there is no valve characteristic curve selected for the hydraulic axis under the Profiles screen form, the value in the **TypeOfAxis.MaxVelocity** configuration data element is determinative for the manipulated variable normalization. In the valve characteristic curve for the P output, the **TypeOfAxis.MaxForceCommandData** configuration data element is determinative.

Assignment of range limits

- The valve position is specified as a percentage (%).
  - Cam - master: manipulated variable -100% to +100%
  - Cam - slave: velocity or pressure -min% to +max%

  See the figure below for an example of how to assign characteristic curve parameters in SCOUT.
- The value specified in **maxSetpointVoltage** or **maxOutputVoltage** is assigned to 100%.
- A valve position of zero is mapped onto an output voltage of zero; thus, the mapping is defined explicitly.
- The interface-specific superimposition is specified as a percentage (%).
- The technology variable is specified in the application unit.

Procedure for recording the characteristic curve for a Q valve

- Enable the axis via **_enableQFAxis()** without controller
- Specify an output value between -100% and +100% via the **servosettings.additionalQOutputValue** (**servosettings.additionalFOutputValue** for P valve) system variable

  Note

  Only traverse the machine within valid range!

- Measure the actual velocity (QOutputValue and velocity can be recorded using the trace function)
- Read out the resulting technological variables
- Enter the values and the technological variables in the characteristic curve
  Each technological variable received for a specified output voltage is entered in the characteristic curve
- Assign the cam to the axis and activate it under Profiles/Valves

  Note

  With F output, the procedure applies for P valves accordingly.

Example of how to set the characteristic curve in SCOUT

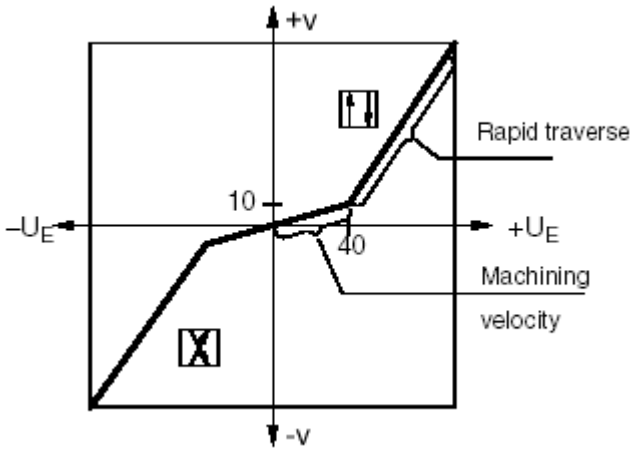


Figure 4-914 Valve characteristic curve from the manufacturer's catalog

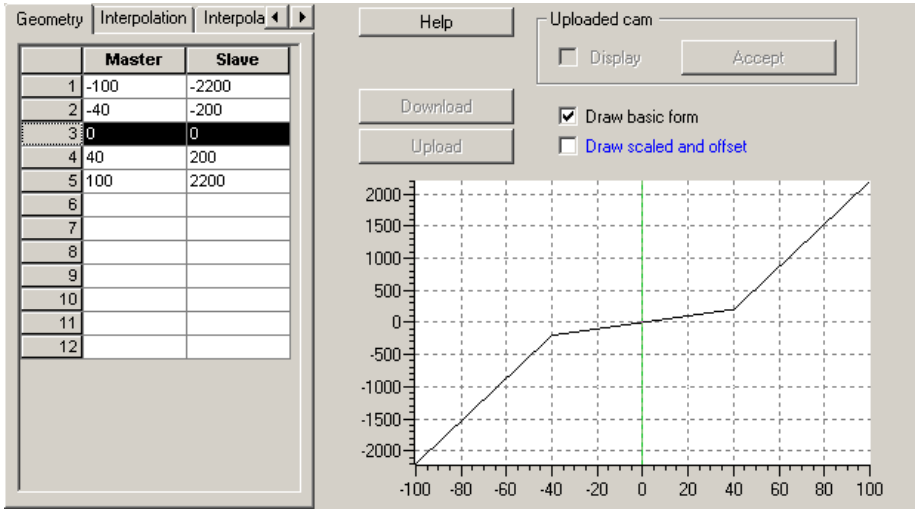


Figure 4-915 Assigning characteristic curve parameters in SCOUT

At 100% manipulated variable, a velocity of 2,200 mm/s is reached in the example (position axis).

See also

Overview of positioning axis with position control (Page 2949)

Access to the same final controlling element from multiple axes

The axis with hydraulic functionality has its own activation/deactivation command **_enableQFAxis()/_disableQFAxis()**.

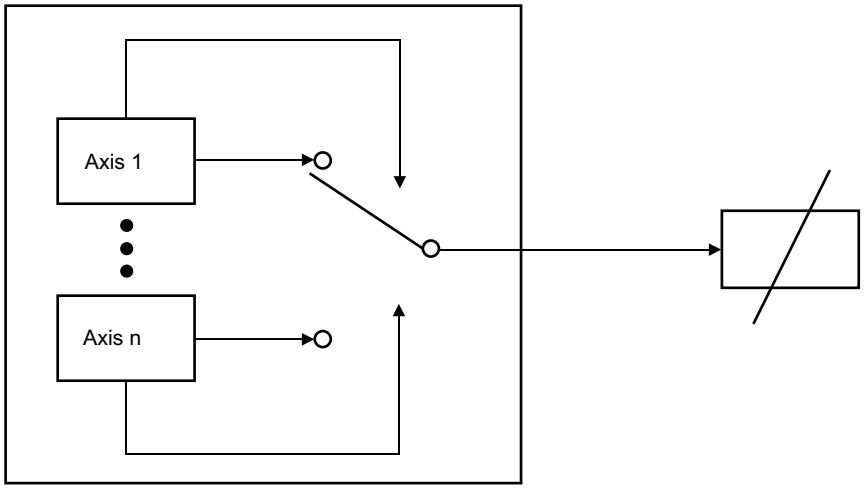


Figure 4-916 Axis with hydraulic functionality: one valve for several axes

When an axis is specified as an axis with hydraulic functionality, you can assign a final controlling element to more than one axis in the configuration. Only one axis can specify the setpoint for a final controlling element at a given time.

Application example: one variable-capacity pump for several axes. The axes are traversed sequentially. After the motion is ended, the axis is disabled (**_disableQFAxis()** with valve enable). Digital valves switch to the next axis. The next axis can be enabled.

This assignment is made during runtime using the **qOutput** or **fOutput** parameter in the **_enableQFAxis()**/**_disableQFAxis()** activation and deactivation commands and is displayed in the **actorMonitoring.qOutputState** or **actorMonitoring.fOutputState** system variable. A final controlling element that has been activated, and thus occupied, by an axis must be re-enabled by this axis before another axis can activate and occupy this final controlling element for itself.

Replacement values (stop values) can be predefined for enabling a final controlling element. This is achieved by enabling the final controlling element (command **_disableQFAxis()**, **qOutput/fOutput** = DISABLE) but without canceling the enabling signal (command **_disableQFAxis()**, **qOutputEnable/fOutputEnable** =. DO_NOT_CHANGE). The default replacement value 0% is output at the final controlling element. With **qOutputValueSetMode**/**fOutputValueSetMode** = set and **qOutputValue/fOutputValue** = 5 in the **_disableQFAxis()** command, a manipulated variable of 5% is output. The replacement value is output until a technology object occupies the final controlling element again (**_enableQFAxis()**).

The manipulated variable change is limited when the final controlling element with replacement value is applied or when changing to a different characteristic curve on the axis. These limits are specified in the **userDefaultQFAxis.maxDerivative** system variable.

The enabling signal for a final controlling element can be configured and set on multiple axes. The axis does not specifically occupy the enabling signal; rather, the enable is performed directly.

See also

Overview of positioning axis with position control (Page 2949)

4.8.7.11 Travel to fixed endstop

The travel to fixed endstop functionality is not available for hydraulic axes.

See also

Travel to fixed endstop (Page 3015)

4.8.7.12 Force/pressure control with hydraulic axes with Q valve only

On the hydraulic axis with Q valve only, the force/pressure control acts on the velocity setpoint (Q valve manipulated variable value) analogous to how the force/pressure control acts on the speed setpoint on the electric axis.

The handling requirements and functionality are therefore the same as for the electric axis.

The **_enableForceControlByCondition()** switchover command is active.

The **_enableForceLimitingByCondition()** switchover command is not active.

For the local RELEASE_DISABLE or OPEN_POSITION_CONTROL alarm response, the manipulated value 0 will be output using the valve characteristic value.

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

Note

Special Q valves (so-called PQ valves) also support pressure control.

See also

Overview of force/pressure control (Page 3026)

4.8.7.13 Force/pressure limiting with hydraulic axes with Q valve only

With hydraulic axes with Q valve only, the force/pressure limiting control is available the same as with the electric axis.

The handling requirements and functionality are therefore the same as for the electric axis.

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

See also

Overview of force/pressure limiting (Page 3033)

4.8.7.14 Force/pressure limiting with hydraulic axes with P valve

The specifications for force/pressure limiting are switched to the P valve/F output. The pressure limitation value in the commands is output on the P-valve/F output as a manipulated variable. The **_enableForceLimitingByCondition()** switchover command with condition is active. The pressure limiting value is not subject to fine interpolation if **_enableForceLimitingValue()** or **_enable...LimitingProfile()** with derivativeLimitingMode=WITHOUT_LIMITING is set.

When the event occurs, the motion is not aborted; the motion specifications continue to be output to the Q-valve. An application can be used to switch back to the motion.

No force/pressure limiting controller is in effect.

No force/pressure control is in effect.

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

4.8.7.15 Force/pressure control with hydraulic speed-controlled axes with Q valve only

Force/pressure control is available for hydraulic speed-controlled axes (V3.2 and higher). (**typeOfAxis**=REAL_QFAXIS_WITH_CLOSED_LOOP_FORCE_CONTROL)

On-the-fly switching between pressure control and speed control is possible (V4.0 and higher).

Note

Note the following points for the transitions from speed control to pressure control and vice versa:

- Apply the force/pressure control during motion (as of V4.0)
- Switch on pressure control at standstill (as of V3.2)
- Switch off pressure control at standstill (as of V3.2)
- Transition to speed control from force/pressure control (as of V4.0)
- If pressure control is active, only the RELEASE_DISABLE stop reaction is in effect. (V3.2)
  - No applying / overriding of the pressure control through motion
  - No preassigned braking ramp
- Switchover with condition not available

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

4.8.7.16 Force/pressure limiting with hydraulic speed-controlled axes with Q valve only (V4.0 and higher)

Force/pressure limiting on the hydraulic drive axis is available (V4.0 and higher).

With the hydraulic drive axis, the pressure limiting component is added to the velocity setpoint. The pressure limiting value is not subject to fine interpolation if **_enableForceLimitingValue()** or **_enable...LimitingProfile()** with derivativeLimitingMode=WITHOUT_LIMITING is set.

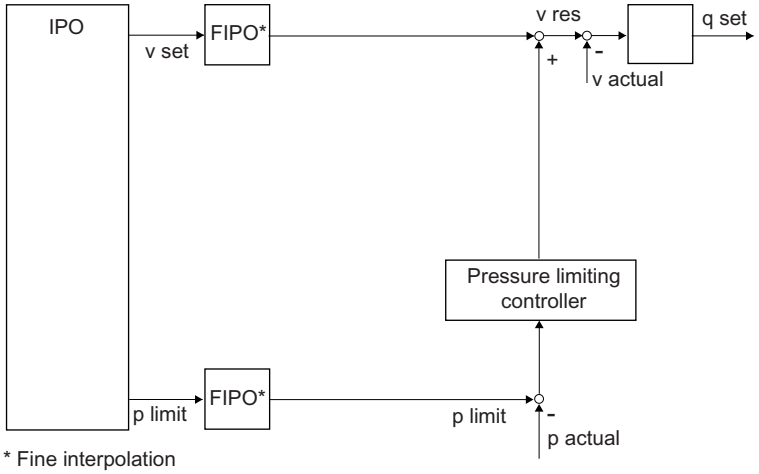


Figure 4-917 Overview of control structure for force/pressure limiting on hydraulic drive axis

Note

The *PDF documents* button can be used in the online help to locate the function charts with signal paths.

Features:

- Parallel pressure limiting
- Transition from pressure-limited to pressure-controlled
- Transition from pressure-controlled to speed-controlled (with/without pressure limiting)

The same controller is used as the pressure controller on the drive axis and as the force/pressure limiting controller.

On-the-fly switching between pressure control and speed control (open-loop and closed-loop) and vice versa is possible (as of V4.0).

Error responses / StopEmergency

If the FEEDBACK_EMERGENCY_STOP error response occurs in combination with the **_stopEmergency()** command with the STOP_WITH_COMMAND_VALUE_ZERO setting in active pressure control, the preassigned braking ramp is executed and the pressure limiting remains active. However, the pressure limiting can be deactivated via a command.

4.8.7.17 Velocity limiting with hydraulic axes

Velocity limiting on the hydraulic axis with Q valve only

This velocity limiting is available:

- On the positioning axis (V3.2 and higher)
- On the speed-controlled axis (V4.0 and higher)

Velocity limiting on the hydraulic axis with P valve and Q valve

The velocity limiting is output to the Q valve as velocity setpoints.

4.8.8 Part III Programming/Reference

4.8.8.1 Overview of commands

Note

You can also find information about the system functions in the *SIMOTION Cam Technology Package list manual*.

Overview of commands

Programming (execution of commands)

The axis is controlled by means of commands. These commands are used to enable and disable functions, specify and influence motions, specify data, and read out status information.

Commands for specification and control of motion on the axis

Table 4-386 Overview of commands for specification and control of motion on the axis

Command	Meaning	D	P	G
Enables, stop and continue commands, resets				
_enableAxis()	Enable axis	X	X	X
_enableQFAxis( )*	Enable hydraulic axis	X	X	X
_disableAxis()	Disable axis	X	X	X
_disableQFAxis( )*	Disable hydraulic axis	X	X	X
_enableForceControlByCondition()	Activating force/pressure control depending on switch-over conditions	-	X	X
_setForceCommandValue( )	Set force/pressure setpoint	-	X	X
_setQFAxisQCharacteristics*( )	Set characteristics for Q value	X	X	X
_setQFAxisPCharacteristics*( )	Set characteristics for P value	X	X	X
_stopEmergency()	Rapid stop with motion abort	X	X	X

Command	Meaning	D	P	G
_stop()	Stop axis motion with/without abort	X	X	X
_continue()	Resume an interrupted motion	X	X	X
_resetAxis()	Reset axis	X	X	X
_getAxisErrorNumberState()	Read out error number status	X	X	X
_resetAxisError()	Acknowledge axis error	X	X	X
_cancelAxisCommand( ), as of V4.1 SP1	Abort command with the specified CommandID	X	X	X
* Active only when the **TypeOfAxis** setting is **QFAxis**.				
Axis motions				
_homing()	Homing	-	X	X
_move()	Continuous turning (speed- or position-controlled)	X	X	X
_pos()	Positioning	-	X	X
_runTimeLockedVelocityProfile()	Rotational speed profile	X	X	X
_runTimeLockedPositionProfile()	Velocity profile	-	X	X
_runPositionLockedVelocityProfile()	Position-related velocity profile	-	X	X
_enableTorqueLimiting()	Switch on torque limitation	X	X	X
_disableTorqueLimiting()	Deactivate torque limitation	X	X	X
_enableAxisAdditiveTorque( )	Activate input interconnection for additive torque	X	X	X
_disableAxisAdditiveTorque( )	Deactivate input interconnection for additive torque	X	X	X
_enableAxisTorqueLimitPositive( )	Activate input interconnection B+	X	X	X
_disableAxisTorqueLimitPositive( )	Deactivate input interconnection B+	X	X	X
_enableAxisTorqueLimitNegative( )	Activate input interconnection B-	X	X	X
_disableAxisTorqueLimitNegative( )	Deactivate input interconnection B-	X	X	X
_enableVelocityLimitingValue()	Activate velocity limiting	X	X	X
_disableVelocityLimiting()	Deactivate velocity limiting	X	X	X
_enablePositionLockedForceLimitingProfile()	Activate force/pressure limiting with position-related force/pressure limiting profile	-	X	X
_enablePositionLockedVelocityLimitingProfile()	Activate velocity limiting with position-related limiting profile	-	X	X
_runTimeLockedForceProfile()	Starting the time-related force/pressure profile	-	X	X
_runPositionLockedForceProfile()	Starting the position-related force/pressure profile	-	X	X
_enableForceLimiting()	Enable force limiting	-	X	X
_disableForceLimiting()	Disable force limiting	-	X	X
_enableMovingToEndStop( )	Activate travel to fixed endstop	-	X	X
_disableMovingToEndStop()	Deactivate travel to fixed endstop	-	X	X
_enableTimeLockedVelocityLimitingProfile( )	Activate velocity limiting with time-related limiting profile	X	X	X
_enableTimeLockedForceLimitingProfile()	Activate force/pressure limiting with time-related limiting profile	-	X	X
Coordinate system				
_redefinePosition( )	Set actual value system	-	X	X
_setAndGetEncoderValue( )	Synchronize measuring systems	-	X	X
_enableMonitoringOfEncoderDifference()	Activate encoder difference monitoring	-	X	X
_disableMonitoringOfEncoderDifference()	Deactivate encoder difference monitoring	-	X	X
_getAxisUserPosition( )	Return user position for specified encoder value	-	X	X

Command	Meaning	D	P	G
_getAxisInternalPosition( )	Return encoder value for specified user position	-	X	X
Simulation				
_enableAxisSimulation()	Activate program simulation	X	X	X
_disableAxisSimulation()	Deactivate program simulation	X	X	X
Information functions / command buffers				
_getStateOfAxisCommand()	Reading the execution status of a motion command	X	X	X
_getMotionStateOfAxisCommand()	Reading current phase of motion	X	X	X
_bufferAxisCommandId()	Save CommandId	X	X	X
_removeBufferedAxisCommandId()	Remove buffered CommandId	X	X	X
_getStateOfMotionBuffer()	Read status of motion buffer	X	X	X
_resetMotionBuffer()	Reset motion buffer	X	X	X
_getProgrammedTargetPosition( )	Read the programmed absolute end position	-	X	X
_getAxisErrorNumberState()	Reading the status of a specific error on the axis	X	X	X
Data set commands				
_setAxisDataSetActive()	Activate data set	X	X	X
_setAxisDataSetParameter()	Write data set	X	X	X
_getAxisDataSetParameter( )	Read a data set	X	X	X
_setForceControlDataSetParameter()	Writing the force/pressure-specific data of a data set	-	X	X
_getForceControlDataSetParameter()	Reading the force/pressure-specific data of a data set	-	X	X
_resetAxisConfigDataBuffer( )	Clear the configuration data in the buffer without activation	X	X	X

D = drive axis

P = positioning axis

G = synchronized axis

PLCopen commands

The blocks listed below can be used in cyclic programs/tasks in SIMOTION.

They are used primarily in the LAD/FBD programming language. PLCopen blocks are available as standard functions (directly from the command library).

Table 4-387 SingleAxis functions for the axis (as of V4.0)

Function	Description
_MC_Power()	Enabling an axis
_MC_Stop()	Stopping the axes
_MC_Reset()	Resetting the axis
_MC_Home()	Homing axes
_MC_MoveAbsolute()	Absolute positioning axes
_MC_MoveRelative()	Relatively positioning axes
_MC_MoveVelocity()	Traversing axes at defined velocity (position axis travels position-controlled)
_MC_MoveAdditive()	Relative traversing of axes by a defined path additively to the remaining path
_MC_MoveSuperimposed()	Relative superimposing of a new motion on an existing motion

Function	Description
_MC_PositionProfile()	Traversing axis by a predefined and specified position/time profile
_MC_VelocityProfile()	Traversing axis by a predefined and specified velocity/time profile
_MC_ReadActualPosition()	Reading the actual position of axis
_MC_ReadStatus()	Reading the status of an axis
_MC_ReadAxisError()	Reading the error of an axis
_MC_ReadParameter()	Reading the axis parameter data type LREAL
_MC_ReadBoolParameter()	Reading the axis parameter data type BOOL
_MC_WriteParameter()	Writing the axis parameter data type LREAL
_MC_WriteBoolParameter()	Writing the axis parameter data type BOOL
Apart from the standard PLCopen functions, the following additional standard axis function is included:	
_MC_Jog()	Continuous or incremental jogging

Table 4-388 MultiAxis functions for the axis (as of V4.0)

Function	Description
_MC_CamIn()	Start camming between a master and a slave
_MC_CamOut()	End camming
_MC_GearIn()	Start gearing between a master and a slave
_MC_GearOut()	End gearing
_MC_Phasing()	Offsets the position of the slave axis relative to the master

Table 4-389 Functions for external encoder (as of V4.1 SP1)

Function	Description
_MC_Home()	Home external encoder
_MC_Power()	Enable external encoder
_MC_ReadStatus()	Read status of external encoder
_MC_ReadAxisError()	Read error of external encoder
_MC_Reset()	Reset external encoder
_MC_ReadParameter()	Read external encoder parameters of data type LREAL
_MC_ReadBoolParameter()	Read external encoder parameters of data type BOOL
_MC_ReadActualPosition()	Read actual position of external encoder

You can find additional information in the *PLCopen blocks* Function Manual and in the online help.

Command properties

Function parameters

Motion commands have the following function parameters:

- Information on the type of motion (**_pos()**, **_move()**, etc.) and any function parameters for specification (e.g. direction specification)
- Information regarding the response to active motions/commands (mergeMode)
- Parameters for the motion (dynamic response parameters)
- Parameters for advancing in the program (nextCommand)
- In the case of profile commands, information regarding the starting point within the profile, if required

The **dynamic response parameters** for motions and motion transitions (velocity, acceleration, and jerk) can be specified in the commands.

Additional properties

The following is applicable for profiles:

- The specified quantity corresponding to a cam-defined function is traversed
- Dynamic response parameters can be specified for travel to the start value in the cam or for travel from the end value
- If the profile is specified with a user-definable cam function, a value within the definition range of the cam must also be specified as the starting point from which the profile is to be applied
- The function in the cam is traversed to the end
- The definition range of the cam is evaluated in the user-defined time unit for time-related profiles and in the user-defined position unit for position-related profiles; the range is evaluated in the user-defined unit of the quantity to be traversed
- The direction of motion, velocity, and acceleration are calculated accordingly
- Behavior at the end of the profile can be configured
- The axis override function has no effect on the positioning profiles. The velocity override and acceleration override are included in the velocity profiles.

The status of motion commands can be seen from the **system variable** for the command; if necessary, additional information such as the braking distance and remaining distance for the positioning command is provided in the **posCommand** system variable.

Commands that contain a CommandId parameter allow the CommandId to be initialized to its defaults (value 0.0) (V3.1 and higher). This does not apply to **_getStateOf...CommandId()** commands, since initialization to defaults would not be practicable in this case. The CommandId should be explicitly specified here.

4.8.8.2 Enables, stop and continue commands, resets

Setting and canceling the axis enables

The commands for enabling and disabling the axis are:

- **_enableAxis(), _enableQFAxis()**
- **_disableAxis(), _disableQFAxis()**

The following enables can be set specifically for the electric axis using **_enableAxis()** / **_disableAxis()**:

- Controller enable
- Drive enable
- Power enable
- Follow-up mode (motion commands are not accepted/executed)
- Force/pressure enable

The following enables can be set specifically for the hydraulic axis using **_enableQFAxis()** / **_disableQFAxis()**:

- Controller enable
- Q output enable
- F output enable
- Follow-up mode (motion commands are not accepted/executed)
- Force/pressure enable

The axis enables for the virtual axis have exactly the same effect as for the electric axis.

Note

Drive and encoder error messages can be acknowledged, even when the error is still present. The same error is not to be signaled again.
The status of the actuator or sensor can be tested in the status indication of the corresponding system variable (**state**).

Activate/deactivate axis

- The **_enableAxis()** / **_disableAxis()** commands are used to enable/disable an axis as an electric axis where the **TypeOfAxis** is set to REAL_AXIS....
- The **_enableQFAxis()** / **_disableQFAxis()** commands are used to enable/disable an axis as a hydraulic axis where the **TypeOfAxis** is set to REAL_QFAXIS....

These commands can be issued while the axis is in motion. Canceling enables results in a technological alarm; the configured alarm response is executed.

Note

If, for example, **_disableAxis()** is called immediately after **_enableAxis()**, the commands can displace one another from the command buffer.

- With **_enableAxis()**, **_enableQFAxis()** the position controller in SIMOTION is enabled immediately, provided nothing different is set in the command parameters.
- The parameter **movingMode**=POSITION_CONTROLLED enables speed and position-controlled operation.
  With SPEED_CONTROLLED, speed-controlled operation is enabled. When activating in speed mode, command parameter **servoControlMode=**ACTIVE must be set so that the setpoint path is activated in the servo.
  In SPEED_CONTROLLED mode, the axis can be traversed if the encoder fails and there are no pending error responses (see section *Switching on the speed specification mode with _enableAxis (V4.0 and higher)*).
- As of V4.1 SP1, it is also possible to activate the hydraulic positioning axis in non-position-controlled mode using the **movingMode** = SPEED_CONTROLLED parameter with command **_enableQFAxis()**
  The closed-loop velocity-controlled hydraulic axis cannot be moved with open-loop velocity control via movingMode=SPEED_CONTROLLED.
- If, when activating the drive with **_enableAxis()**, the position controller is not enabled in SIMOTION or the axis remains in follow-up mode because user-defined functions such as motor identification, brake release, etc. are executed in the drive, settings options are available with **servoControlMode** = INACTIVE or **servoCommandToActualMode** = ACTIVE. Thus, the position controller does not position on the setpoint when **_enableAxis()** is activated.
  After the transition to the Operation or S4 drive state of the PROFIdrive state machine (displayed in the **actorMonitoring.power** = ACTIVE system variable), the position controller can then be enabled in SIMOTION with the **_enableAxis()** command and the **servoControlMode** = ACTIVE parameter setting or switched from follow-up mode with **servoCommandToActualMode** = INACTIVE.
- Axes can also be activated when the actual velocity is not equal to zero. You can use the configuration data item **decodingConfig.speedModeSetPointZero** to set the mode for getting the axis to zero velocity when the enables are set.
  - **decodingConfig.speedModeSetPointZero**=YES
    The velocity setpoint (and acceleration) of the axis are set directly to zero without a braking ramp.
    As of SCOUT V4.2, this is the standard setting when creating a new axis TO.
  - **decodingConfig.speedModeSetPointZero**=NO
    The axis is stopped via a braking ramp with the maximum dynamic values.
    See also Manipulated variable superimposition (Page 2976).
    Please note that braking may be superseded by other motions which become effective immediately after the axis is activated.

Axis enable and follow-up mode

In the following tables, the behavior and displays with different configurations of **_enableAxis()** / **_enableQFAxis()** are described.

Table 4-390 Behavior and displays for axis enable with **traversing mode "position and speed controlled"**

	Follow-up mode ACTIVE (Follow-up setpoint)	**Follow-up mode INACTIVE (Do not follow-up setpoint)**
Position controller ACTIVE (**servoControlMode**=ACTIVE)	Special case	**STANDARD SETTING**
	• Traverse of the axis using setpoint superimposition is possible (see Setpoint superimposition (Page 2967)) using **servosettings.additionalCommandValue** (e.g., for commissioning the position controller) • Axis is in position control (**servoMonitoring.controlState**= ACTIVE) • Motion commands CANNOT be performed (due to follow-up mode, **control**=INACTIVE) • Set position IPO is tracked • Servo set position is not tracked • Monitoring functions are not active	• Axis is in position control (**servoMonitoring.controlState**=ACTIVE) • Motion commands can be executed (**control**=ACTIVE)
Position controller INACTIVE (**servoControlMode**=INACTIVE)	Special case	Special case
	• The drive is only enabled for drive-autonomous motions and functions • Axis is NOT in position control (**~.controlState**=INACTIVE) • Motion commands CANNOT be executed • IPO set position and servo is tracked • Monitoring functions are not active	In real axes, "Do not follow up setpoint" is only effective if all the other enables have been assigned.

Table 4-391 Behavior and display for axis enable with traversing mode "**speed controlled**"

	Follow-up mode ACTIVE (Follow-up setpoint)	**Follow-up mode INACTIVE (Do not follow-up setpoint)**
Position controller ACTIVE (**servoControlMode**=ACTIVE)	Special case	**STANDARD SETTING**
	• Traversing the axis without position control using manipulated variable superimposition is possible (see Manipulated variable superimposition (Page 2976) using **servosettings.additionalSetpointValue** • Axis is NOT in position control (but servo setpoint enable is active, so **~.controlState**=ACTIVE) • Motion commands CANNOT be performed (due to follow-up mode, **control**=INACTIVE) • IPO set position and servo are tracked using following error model (set and actual position are offset by following error) • Monitoring functions are not active	• Axis is NOT in position control (but servo setpoint enable is active, so **~.controlState**=ACTIVE) • Motion commands for speed-controlled procedures are possible (**control**=ACTIVE) • Set position in IPO and servo are tracked using the following error model (set and actual position are offset by the following error) • Standstill monitoring is not active (internal follow-up mode, as speed-controlled mode)
Position controller INACTIVE (**servoControlMode**=INACTIVE)	Special case	Special case
	• The drive is only enabled for drive-autonomous motions and functions • Axis is NOT in position control (**~.controlState**=INACTIVE) • Motion commands CANNOT be executed • IPO set position and servo is tracked • Monitoring functions are not active	In real axes, "Do not follow up setpoint" is only effective if all the other enables have been assigned.

Remove axis enable

In the command **_disableAxis()** or **_disableQFAxis()**, (Remove axis enable) the position controller, drive and power enable withdrawn, "Do not follow up setpoint" has no effect.

Enabling line infeed

Infeed must be enabled before activating the axis.

Infeed can be activated and deactivated, and basic diagnostics can be carried out, with the **_LineModule_control** function block (as of V4.2).

The following infeeds are supported by the **_LineModule_control** function block:

- Basic Line Modules (BLM)
- Smart Line Modules (SLM)
- Active Line Modules (ALM)

The **_LineModule_control** function block is part of the command library of the SIMOTION SCOUT engineering system. You can find the function block under **Drives > SINAMICS**.

For more detailed information about the **_LineModule_control** function block, see the SIMOTION SCOUT online help (standard function for Line Modules) or the Function Manual *Standard Function for SINAMICS S120 Line Modules*.

The **LineModule_control** function block can be used for firmware versions < V4.2. You can find the block under *Applications > Cross-sector applications > Function block for controlling Line Modules* in the *SIMOTION Utilities & Applications*.

Monitoring

The status of the current drive and power enables for an electric axis is displayed in the following system variables:

- **actormonitoring.drivestate** (drive enable)
- **actormonitoring.power** (power enable)

In a hydraulic axis, the occupation status of the Q final controlling element and F controlling element is displayed in the following system variables:

- **actorMonitoring.QOutputState** (Q output)
- **actorMonitoring.FOutputState** (F output)
  The Q valve enable is displayed in **actormonitoring.drivestate**.

If the axis is switched from follow-up mode, the **Control** system variable switches to ACTIVE. The **Control** system variable indicates whether the axis is able to generate motions.

The axis goes into follow-up mode after control startup. Enables are not mandatory in follow-up mode.

For position-controlled axes, disabling an electric drive results in automatic disabling of the position controller. The position control enable signal is ignored for speed-controlled axes. For a hydraulic axis, enabling or disabling of the Q output results in disabling of the controller.

Behavior of electric axis should a digital drive be missing or switched off

If a digital drive is missing or switched off, the **_enableAxis()** command and motion commands behave as follows:

- If a drive is missing, **_enableAxis()** is aborted with an error message
- If the drive is switched on but is not yet in cyclic operation, if commands are issued synchronously, the system waits for the command; the command is not aborted and the command job is executed even if it is an asynchronous command; the command remains active even with asynchronous programming
- The **cyclicInterface** system variable on the axis indicates whether the drive is in cyclic operation, i.e. **cyclicInterface**= ACTIVE; cyclic operation is determined using the life-sign
  Comment:
  - On analog onboard axes, **cyclic operation active** is always indicated.
  - If a drive error is present (e.g. temperature error), the system variable displays INACTIVE.

Force/pressure control

Force/pressure control can be explicitly enabled using the **forceControlMode** function parameter in the enable/disable command. This requires that the axis is at a standstill (standstill signal **motionStateData.motionState**=STANDSTILL).

When enabled, the last actual force/pressure value is taken as the force/pressure setpoint.

Force/pressure control is deactivated using the **_disableAxis()** or **_disableQFAxis()** command, or by deselecting force/pressure control in the **_enableAxis()** or **_enableQFAxis()** command.

For switching over during motion, the conditional switchover with switchover criterion with active pressure limitation is available.

System variables

Table 4-392 System variables

Variable	State	Meaning
.control	Active / Inactive	Axis active or inactive
.servomonitoring.controlstate	Active / Inactive	Position controller active or inactive
.actormonitoring.drivestate	Active / Inactive	Drive enable (drive ON) for electric axis Q valve enable for hydraulic axis
.actormonitoring.power	Active / Inactive	Power enable (pulse enable) (for electric axis only)
.actorMonitoring.qOutputState	Active / Inactive	Q output active (for axis with hydraulic functionality only)
.actorMonitoring.fOutputState	Active / Inactive	F output active (for axis with hydraulic functionality only)
.actorMonitoring.fOutputEnable	Enabled / Disabled	Status of F output enable (for axis with hydraulic functionality only)

Variable	State	Meaning
.cyclicInterface	Active / Inactive	Communication active (only for DP, this is always active for analog or onboard axes; if a drive error is present (e.g. temperature error), the system variable displays INACTIVE.)
.velocityControllerServo Monitoring.controlstate	Active / Inactive	Velocity controller on the hydraulic drive axis active or inactive

Setting enables individually

For PROFIdrive-coupled drives, the BY_STW_BIT parameter of the **_enableAxis()** and **_disableAxis()** functions provide an interface for setting and canceling of individual enables. This allows the users to implement their own state machine transitions.

Bits 0 to 6 of the ctrl word1 control word can be set and reset using the BY_STW_BIT parameter:

_enableAxis() sets the bits transferred in the parameter in the control word

_disableAxis() clears the bits transferred in the parameter in the control word

For non-PROFIdrive-coupled drives, these bits have no significance and are rejected with 'illegal command parameters'.

The statuses can be read from status word STW via the **driveData** system variable. The state display in the **actorMonitoring.driveState** and **actorMonitoring.power** system variable remains unchanged.

Entering control bits 0 to 6 directly using the **_enableAxis()** and **_disableAxis()** functions can be combined with specifying the command mode (DRIVE or POWER).

For detailed information about linking drives using PROFIdrive, refer to Coupling of digital drives (Page 2895).

For information on defining the reaction to technological alarms, refer also to Section Adjustable response to RELEASE_DISABLE (Page 3185).

Note

STW1 is initialized by the drive and the **actorMonitoring.driveState** and **actorMonitoring.power** system variables are generated in STW1 according to the drive status. (as of V3.2)

The status display in these system variables is thus delayed by one cycle clock relative to the control word specifications. (as of V3.2)

Enabling the speed specification mode with _enableAxis (as of V4.0)

By enabling speed specification mode (parameter **movingMode**=SPEED_CONTROLLED of **_enableAxis()**), it is possible to continue motion in the event of the failure of a selected encoder that is not involved in closed-loop control. When activating in speed mode, command parameter **servoControlMode=**ACTIVE must be set so that the setpoint path is activated in the servo.

In position-controlled operation, the axis is brought to a standstill if a selected encoder involved in the position control fails. The drive enable to be canceled can be specified (**driveControlConfig.releaseDisableMode** configuration data element).

After the axis is stopped and the error is acknowledged, the axis can be switched to activation mode with speed specification. This allows non-position-controlled axis motion in the case of an encoder failure.

Note

- Failure of an encoder involved in the closed-loop control for position-controlled positioning still causes the axis to be brought to a standstill.
- In differential position measurement, all involved encoders are in the active state.
- If a command is issued to an encoder which has an error, alarm 20005 is issued as previously. This can be the case, for example, with an encoder that has an error in the data set.
- In SIMOTION, the encoder selected for closed-loop control is specified in the data set.

See also

Command groups (Page 3047)

Enabling force/pressure control depending on switchover conditions (Page 3150)

Coupling of digital drives (Page 2895)

Dynamic limiting functions (Page 2999)

Positioning and standstill monitoring (Page 2941)

Enabling force/pressure control depending on switchover conditions

The **_enableForceControlByCondition()** command causes a switch to force/pressure control when the switchover criterion defined in the command is satisfied.

The switchover criterion is checked in the servo. The switchover criterion can be a position, pressure, time, or digital input. The conditions are specified in the command and can be logically combined in the command in multiple stages. Reissuing the command before the conditions are met enables you to switch between conditions.

After the switchover to force/pressure control, the axis executes the function specified in the cam according to the profile specification. The rise in pressure for any necessary transitional motions, e.g. for entering and exiting the profile, can be programmed in the command.

Force/pressure, position, and time are stored at the time of switchover and are available in the system variables.

Velocity limiting can also be activated as a dependent condition in the switchover command, and a velocity limit value can be set.

The velocity limit is set directly, or the current set velocity (manipulated variable) or actual velocity can be retained as the velocity limit (as of V4.1 SP1).

A pressure/time profile or a pressure/position profile (V4.1 SP1 and higher) is enabled or the pressure value can be directly specified (as of V4.1 SP1) when the switchover occurs.

Axis with Q valve

The conditional switchover is triggered by the **_enableForceControlByCondition()** command.

Axis with P valve

Only the force limiting commands have any effect; the conditional switchover is triggered by the **_enableForceLimitingByCondition()** command.

Stopping motions with _stopEmergency()

The **_stopEmergency()** function stops the axis with a programmable stop mode. If a motion command is active, it is aborted and cannot be continued with a **_continue** command. The axis does not switch to follow-up mode. The axis is prevented from receiving additional motion commands. This status can be reset with **_resetAxis()** or **_disableAxis ()**. The command becomes active immediately.

The motion on the axis is stopped according to the behavior set in the command; the position control is not affected. Torque limiting, any active torque reduction (including when traveling to fixed stop) and force/pressure limiting are retained.

Active and pending motion commands on the axis are aborted and cannot be resumed.

Note

When a motion is stopped using **_stopEmergency()** and a smooth velocity profile with jerk specification is present during the axis acceleration phase, the velocity can continue to increase until the acceleration is reduced by means of the jerk which has been set.

In extreme cases, the velocity can rise still further, until it reaches the configured maximum velocity. The axis is only decelerated once acceleration has been reduced by the jerk.

With the parameter **abortAcceleration** (as of V4.2.1), you can set that a possible acceleration does not exceed the jerk limitation, but is immediately cleared.

Status

The **stopEmergencyCommand.state= ACTIVE** status is set. Motion commands on the axis are not active in this status.

Table 4-393 System variable for _stopEmergency command

Variable	State	Meaning
stopEmergencyCommand.state	ACTIVE	The **_stopEmergency()** command was triggered.
	INACTIVE	No **_stopEmergency()** command was triggered, or the command was acknowledged with **_resetAxis()** or **_disableAxis()**.

stopEmergencyCommand=ACTIVE status is canceled with **_disableAxis()** or **_resetAxis()**.

Note

The **stopEmergencyCommand**=ACTIVE status may last longer than the **_disableAxis()** command is active, so the **stopEmergencyCommand**=INACTIVE status should be checked explicitly before the enables are reset.

The axis enables are not deactivated.

The command has no effect if the axis is in follow-up mode; the status is not set.

A **_stopEmergency()** command with a higher-priority stop response overrides a lower-priority response.

The stop behavior can be set using the **stopDriveMode** function parameter.

The priority setting is defined in the stop behavior as follows:

- Specification of timing for stop motion (STOP_IN_DEFINED_TIME), irrespective of current velocity
- Stopping with maximum dynamic response values on the axis (STOP_WITH_MAXIMAL_DECELERATION)
- Stopping with dynamic response parameters (STOP_WITH_DYNAMIC_PARAMETER) This is set during configuration.
- Stop axis with setpoint zero (STOP_WITH_COMMAND_VALUE_ZERO)

StopEmergency with dynamic response parameters (as of V3.2)

The stopDriveMode=STOP_WITH_DYNAMIC_PARAMETER setting allows dynamic values to be entered and enabled directly in the **_stopEmergency()** command.

The defaults defined in the **userDefaultDynamics** system variable are used as default parameters.

See also

Stopping with preassigned braking ramp (Page 3002)

Stopping motions with _stop()

The **_stop()** and **_continue()** commands can be used to stop and resume motions.

The **_stop** command is effective in the following technology object states: motion, motion stop without abort. It stops the entire motion or a portion of the motion of the axis using a programmed braking ramp.

The motion to be stopped can be interrupted or terminated. The portion of the motion to be stopped is specified using the **Command ID** or the type of motion.

An interrupted motion (entire motion or portion of motion) that was stopped with **_stop()**, stopMode=STOP_WITHOUT_ABORT and was not completely interpolated before the stop command was issued can be resumed using the **_continue()** command. When a motion is

resumed, the dynamic response parameters of the interrupted command, such as velocity profile, acceleration, etc., are used.

Motions interrupted by **_stop()** commands in **stopMode**=STOP_WITHOUT_ABORT can be overridden by motion commands with **mergeMode** IMMEDIATELY. The NEXT or SEQUENTIAL merge modes do not cause the motion to be executed immediately.

stopMode=STOP_AND_ABORT aborts the commands selected in the stop command. The aborted commands are returned with errors. When motion commands are aborted, the **Command Aborted** technology alarm is generated.

A command containing a position-controlled stop motion changes from ACTIVE to EXECUTED if the interpolation is completed, and the actual value is in the positioning window and the minimum dwell time has elapsed (MOTION_DONE). If the axis does not achieve this status, e.g., due to active torque limiting for deactivated following error monitoring and deactivated position monitoring (and therefore no canceling of a command due to a local alarm response), the command status remains ACTIVE.

Note

When a motion is stopped using **_stop()** and a smooth velocity profile with jerk specification is present during the axis acceleration phase, the velocity can continue to increase until the acceleration is reduced by means of the jerk which has been set.

In extreme cases, the velocity can rise still further, until it reaches the configured maximum velocity. The axis is only decelerated once acceleration has been reduced by the jerk.

With the parameter **abortAcceleration** (from V4.2.1), you can set that a possible acceleration does not exceed the jerk limitation, but is immediately cleared.

Stopping position-controlled axes in speed-controlled mode (V3.1 and higher)

- The **movingMode=SPEED_CONTROLLED** parameter of the **_stop()** and **_stopEmergency()** commands can be used to stop position-controlled axes in speed-controlled mode.
  The speed ramp takes effect immediately, and any pending following error is not removed. This also means that in force-controlled mode, force-limited mode, torque-limited mode, and velocity-limited mode, the axis is stopped immediately with the ramp via **_stopEmergency()** on switching over to speed-controlled mode.
- Axes can be stopped with position control in the **movingMode=POSITION_CONTROLLED** setting.
  With a drive axis, the **POSITION_CONTROLLED** setting is ignored.
- In the **movingMode=CURRENT_MODE** setting (default setting, compatibility mode), the axis is stopped in the last traversing mode set on the axis.
  In force-/pressure-controlled operation as well, the axis is stopped in the last traversing status (position-controlled or speed-controlled) set on the axis.

This enables a transition on the position-controlled axis with **_stopEmergency()** from each operating mode to speed-controlled mode and to position-controlled mode.

Comment:

Position-controlled commands cannot be resumed after the **_stop()** command is set with **movingMode= SPEED_CONTROLLED**; the motion command is aborted.

Resuming motions

The **_continue()** function resumes the complete motion or portion of a motion of the specified axis if it was stopped with **stop()** and STOP_WITHOUT_ABORT in the **stopMode** parameter.

The portion of the motion to be resumed is specified with the **Command ID** or by the type of motion.

When a motion is resumed, the dynamic response parameters of the interrupted command, such as velocity profile and acceleration, are used.

Resuming motions after _disableAxis() (V3.2 or higher)

With setting **TypeOfAxis.DecodingConfig.disableMotionOperation**=No, motion commands that are stopped with **_stop()** (stopMode=STOP_WITHOUT_ABORT) are not aborted by **_disableAxis()**, and may be resumed after **_enableAxis()** with **_continue()**.

The following restrictions apply:

- Only the main motion (basic motion) can be resumed.
- If motions are superimposed, continuation is only possible if the superimposed coordinate system is returned to while in the stop state.
  To do this, the **decodingConfig.transferSuperimposedPosition** configuration data element must be configured with TRANSFER_STANDSTILL.
- Relative positioning is started again, i.e. it is run through again in full after **_continue()**.

Note

_stopEmergency() is used to abort all motion commands; it is not possible to continue in this case.

Reset axis

The **_resetAxis()** command places the axis in a defined initial state.

- All active motions are stopped with a preassigned braking ramp, and the axis enables are retained.
- Commands in the motion buffer and commands pending at the motion buffer are deleted; synchronous commands are aborted. The **Command aborted** technology alarm is suppressed.
- Optionally, the axis can be restarted using a function parameter in the command.
- The command is executed synchronously.
- The command can also be executed asynchronously (V3.1 and higher).
- Pending errors on the axis are reset. The command is terminated with a negative acknowledgment if any errors occur that must not be acknowledged at this point.
- Any system variables changed by the program are reset to the configured values upon request. Actual values are retained.
- The homing status is retained.

Restart

When restart is activated on the technology object with the **activateRestart** parameter, the technology object performs a restart.

- An active motion on the axis is aborted without error messages or the like.
- The actual value system of the axis and, thus, the homing status is reset.
- All commands on the axis are aborted.
- All connections to other technology objects are removed and reestablished.

The **_resetAxis()** command stops any synchronous operation command that is in effect on this axis. The Synchronous Operation technology object on the synchronous axis is not reset.

Note

Setting **restartActivation** initiates a restart that is performed asynchronously. For this reason, it can take some time for the TO restart to begin.

If synchronous processing is required, **_resetAxis()** with the **ACTIVATE_RESTART** parameter must be used.

System variables

Table 4-394 System variables for the _resetAxis() command

Variable	State	Meaning
.reset	ACTIVE/INACTIVE	Reset command is active or not active
.restartActivation	ACTIVATE_RESTART	Setting the **restartActivation** variable to ACTIVATE_RESTART activates a restart of the technology object.
	NO_RESTART_ACTIVATION	Once the restart of the technology object is completed, the system resets the variable to NO_RESTART_ACTIVATION.

See also

Saving the Command ID (Page 3173)

Resetting an axis error

The **_resetAxisError()** function resets a specified error or all errors on the axis. The command is terminated with a negative acknowledgment if any errors occur that must not be acknowledged at this point.

The command is asynchronous. The command can also be issued synchronously (V3.1 and higher). If necessary, the error is not reset until the local reaction triggered by the error has been completed.

Please also note the information in the **Motion Control Basic Functions**manual under *Display and acknowledge technological alarms*.

System variables

Table 4-395 System variables for the _resetAxisError() command

Variable	State	Meaning
.error	NO/YES	No errors/warnings are pending on the axis, or one or more errors/warnings are pending on the axis.
.errorReaction	See the **EnumAxisErrorReaction** data type as defined in the reference list for the system variables	Indicates the response that occurs based on one or more technological alarms

Canceling/deleting an axis command (as of V4.1 SP1)

The **_cancelAxisCommand()** function cancels the execution of the command or function with the CommandId specified in the **_cancelAxisCommand()** command and removes the command from the job list.

With **_cancelAxisCommand()**, commands such as **_homing()** with setting **homingMode**=PASSIVE_HOMING can be canceled too, as long as they are not the active motion.

The motion is cancelled with the maximum dynamic values. This also applies to superimposed motions, whereby the dynamic values from either motion cannot exceed the maximum values.

4.8.8.3 Commands for axis motions

Homing

The **_homing()** command homes the axis.

Various homing modes can be set by means of the **homingMode** parameter:

- Active homing (**ACTIVE_HOMING**)
  The axis is homed according to the sequence defined in the configuration.
- Setting of current position value (**DIRECT_HOMING**)
  The axis coordinate system is set to the value of the home position coordinate. The axis does not move.
- Offsets the actual position value (**DIRECT_HOMING_RELATIVE**)
  The axis coordinate system is offset by the value of the home position coordinate. The axis does not move.

- Absolute encoder adjustment
  - **homingMode**=**ENABLE_OFFSET_OF_ABSOLUTE_ENCODER**
    The configured value for the absolute encoder adjustment is compensated additively with the retentively stored offset. The total offset is stored in the NVRAM and is available after the control is switched off.
  - **homingMode**=**SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION (as of V4.1 SP1)**
    The value in the *homePosition* parameter is set as the current position, and the resulting absolute encoder offset is calculated based on this. The total offset is stored retentively and is available after the control is switched off.

    Note

    Once a new project has been downloaded to the control, the stored offset is no longer available.

- Passive homing (**PASSIVE_HOMING**)
  The **_homing()** command with the PASSIVE_HOMING setting does not itself trigger an active axis motion; rather, homing occurs during the next axis motion. The axis is homed according to the sequence defined in the configuration.
  The motion command can be triggered before or after the **_homing()** command.
  The **_homing()** command is active in parallel until the axis is homed.
  Disabling the command in SIMOTION versions earlier than V3.2
  - **_resetAxis()**
  - **_disableAxis()**

  Disabling the command as of SIMOTION V3.2 with
  - **_resetAxis()**
  - **_stopEmergency()**

The sequences for active homing, **homingMode**=ACTIVE_HOMING, and the criteria for passive homing, **homingMode**=PASSIVE_HOMING, are set in the configuration.

The dynamic parameters for homing are programmable and refer to all phases of the homing procedure.

An axis has **homed** status when the axis coordinate system has been aligned with the homing signal. The status can be read out in the **positioningstate.homed** system variable.

System variables

Table 4-396 System variables for the _homing() command

Variable	State	Meaning
.userDefaultHoming	See the **StructAxisHomingDefault** data type as defined in the reference list of the system variables	User defaults for homing
.homingCommand	See the **StructAxisHomingCommand** data type as defined in the reference list of the system variables	Status of the homing sequence
.positioningstate.homed	YES/NO	Axis is/is not homed

See also

Homing (Page 2923)

Moving

The **_move()** command can be applied to all axis types.

With the speed-controlled axis, the motion is implemented with speed specification.

With the positioning axis, the motion is implemented as velocity specification. In this case, the motion can be implemented with position-control (signal-integrated velocity) or only as a velocity specification without position control.

This selection is made using the **movingMode** parameter in the command.

- Position-controlled motion (POSITION_CONTROLLED)
- Speed-controlled motion (SPEED_CONTROLLED)

The **movingMode** parameter has no effect on the speed-controlled axis.

The **_move()** command can be applied as a superimposed command.

AS of SIMOTION V4.4, an existing acceleration or deceleration can be set to zero immediately when a motion is initiated by the **_move()** command. The existing acceleration or deceleration is not first reduced via the jerk. The **abortAcceleration** function parameter is available for this purpose.

Table 4-397 System variables for moving with _move()

Variable	State	Meaning
moveCommand.state	INACTIVE / ACTIVE_ABSOLUTE /ACTIVE_RELATIVE	Indicates that a **_move()** command is active on the axis. A distinction is made according to whether the **moveCommand.TargetVelocity** variable indicates an absolute or relative value.
moveCommand.TargetVelocity		Indicates the absolute or relative velocity.
moveCommand.relativeActual VelocityToTargetVelocity		Indicates the current velocity relative to the target velocity.

See also

Traversing the axis via velocity specifications (Page 3002)

Positioning

- The **_pos()** motion command moves the axis to the programmed target position using an assignable velocity profile.
- The position can be specified as an absolute or relative position.
- The direction of motion can be specified for modulo axes because the target position can be attained in various directions with such axes. The various modes must be entered using the **direction** parameter (see table).

- **_pos()** commands have a special **blending** mode that links the _pos command to the previous _pos command.
- The **_pos** commands can be applied as superimposing commands.

Note

While the positioning motion is active, if the setpoint system is reset using **_redefinePosition()** or **_homing()** (DIRECT_HOMING / PASSIVE_HOMING), the motion is resumed as follows:

- For absolute positioning, the target position is approached in the newly-defined coordinate system.
- For relative motion, the relative path is traveled. In this case, offsetting the logical coordinate system has no effect.

AS of SIMOTION V4.4, an existing acceleration or deceleration can be set to zero immediately when a motion is initiated by the **_move()** command. The existing acceleration or deceleration is not first reduced via the jerk. The **abortAcceleration** function parameter is available for this purpose.

The following table provides an overview of the possible combinations of direction specification, axis type, and positioning mode.

Table 4-398 Possible specifications for direction in the _pos( ) command

	Non-modulo axis / absolute positioning	**Non-modulo axis / relative positioning**	**Modulo axis / absolute positioning**	**Modulo axis / relative positioning**
POSITIVE	Direction determined from the target position	Positive direction	Positive direction	Positive direction
NEGATIVE	Direction determined from the target position	Negative direction	Negative direction	Negative direction
BY_VALUE	Direction determined from the target position	Direction determined from the sign of the distance specification	Direction determined from the sign of the position specification	Direction determined from the sign of the distance specification
SHORTEST_WAY	Direction determined from the target position	Direction determined from the sign of the distance specification	Direction is selected to achieve shortest distance to target	Direction determined from the sign of the distance specification

System variables

Table 4-399 System variables for superimposed positioning

Variable	**State**	**Meaning**
posCommand	See the **StructAxisPosCommand** data type as defined in the reference list for system variables	Status of the positioning motion contains status, current target position, remaining distance, and target braking distance

See also

Positioning (Page 3003)

Positioning with blending (Page 3003)

Starting a time-related velocity profile

With the **_runTimeLockedVelocityProfile()** command, the axis is traversed via a user-definable time-related velocity profile.

The velocity profile is stored in a cam.

The profile is applied from a definable starting point to the end. The dynamic response parameters for any necessary transition motions, e.g. for entering the profile and exiting the profile, can be programmed on the command.

The traversing behavior at the end of the profile is set during configuration in **decodingConfig.behaviourAtTheEndOfProfile**.

For position-controlled axes, the **movingMode** parameter in the command can be used to specify whether the axis is to move with position control or at a defined velocity.

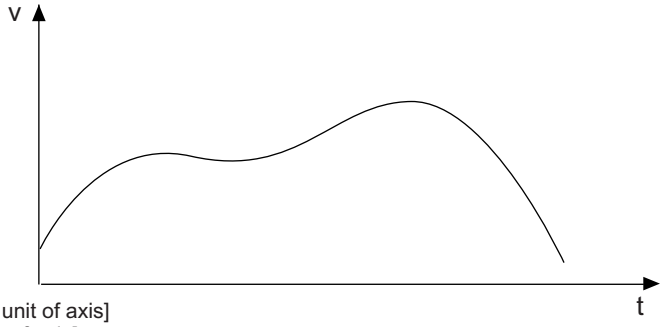


Figure 4-918 Example of a time-related velocity profile

The status and values are displayed in the elements of the **velocityTimeProfileCommand** system variable.

See also

Behavior at the end of the profile (V3.2 and higher) (Page 3045)

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Starting a position-related velocity profile

With the **_runPositionLockedVelocityProfile()** and **_runMotionInPositionLockedVelocityProfile()** commands, the axis is traversed via a user-definable position-related velocity profile.

The domain / the x coordinate of the cam corresponds to the absolute axis position (setpoint).

The profile is started at the current axis position. This must fall within the domain of the profile, otherwise the command is aborted and an alarm is triggered.

Note

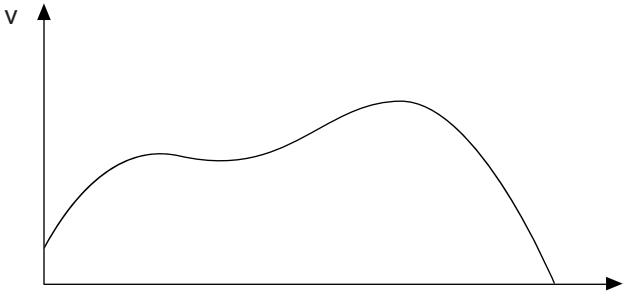
The set velocity must be a value other than zero at the starting point.

The dynamic response parameters for any necessary transition motions, e.g. for entering the profile and exiting the profile, can be programmed on the command.

The traversing behavior at the end of the profile is set during configuration in **decodingConfig.behaviourAtTheEndOfProfile**.

In addition, a tolerance for the detection of the end of the profile can be specified via the **TypeOfAxis.VelocityPositionProfile.endPositionTolerance** configuration data element. The tolerance is required if the above command has been preassigned with the WHEN_MOTION_DONE command transition.

The movingMode parameter in the command can be used to specify whether the axis is to move with position control or only at a defined velocity.



v [velocity unit of axis]
s [position unit of axis]

Figure 4-919 Example of a position-related velocity profile

The status and values are displayed in the elements of the **velocityPositionProfileCommand** system variable.

See also

Behavior at the end of the profile (V3.2 and higher) (Page 3045)

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Positioning with user-definable position profile

With the **_runTimeLockedPositionProfile()** and **_runMotionInPositionLockedVelocityProfile()** commands, the axis is traversed via a user-definable time-related position profile. A starting point can be specified in the cam.

The profile is applied from a definable starting point to the end. An absolute or relative position reference can be selected in the command.

The traversing behavior at the end of the profile is set during configuration in **decodingConfig.behaviourAtTheEndOfProfile**.

A transition profile is generated by the axis for discontinuous transitions. The transition profile is specified in a dynamic response parameter using the acceleration, jerk, and velocity profile parameters of the command.

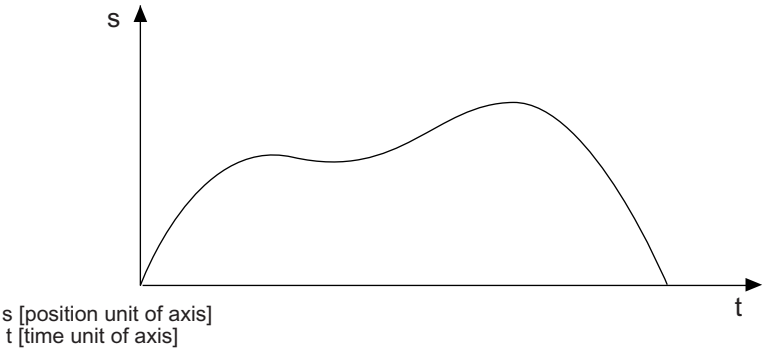


Figure 4-920 Example of a time-related position profile

The status and values are displayed in the elements of the **positionTimeProfileCommand** system variable.

See also

Behavior at the end of the profile (V3.2 and higher) (Page 3045)

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Enabling and disabling torque limiting

The **_enableTorqueLimiting()** command enables torque limiting in the drive, which takes effect immediately. The limiting value is specified in the command.

Active motion commands and synchronous operation relationships continue to be executed.

The limiting functions can be enabled before a motion or simultaneously with a motion and can be switched by reissuing the command.

The commands for travel to fixed endstop (**_enableMovingToEndStop()**) and torque limiting (**_enableTorqueLimiting()**) cannot be active at the same time.

The **_disableTorqueLimiting()** command cancels the torque limiting.

System variables

Table 4-400 System variables for torque limiting

Variable	State	Meaning
TorqueLimitingCommand.State	ACTIVE/INACTIVE	Indicates the torque limiting status
UserDefaultTorqueLimiting.TorqueLimit		Default for torque limiting
actualTorque		Actual torque on the axis See also **Technology data**.

See also

Overview of torque limiting via torque reduction (Page 3010)

Technology data (Page 3018)

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Enable force/pressure limiting with position-related force/pressure limiting profile

With the **_enablePositionLockedForceLimitingProfile()** and **_enableMotionInPositionLockedForceLimitingProfile()** commands, the pressure limitation is activated with a position-related force/pressure limiting profile. The profile is set by means of a cam.

The domain of the cam to be used in the command is interpreted as a position, and the range as a force/pressure limiting value in the respective position and force/pressure units of the axis.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

The profile is started at the current axis position. This must fall within the domain of the profile, otherwise the command is aborted and an alarm is triggered.

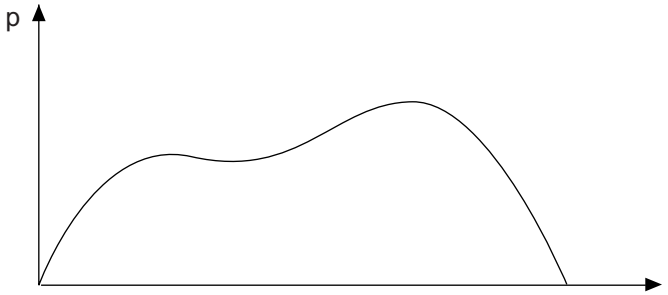


Figure 4-921 Example of a position-related force/pressure limiting profile

The status and values are displayed in the elements of the **forceLimitingCommand** system variable.

See also

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Enabling force/pressure limiting with time-related motion profile

The **_enableTimeLockedForceLimitingProfile()** command activates pressure limiting with a time-related limiting profile. The profile is set by means of a cam.

The starting point can be specified in the cam.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

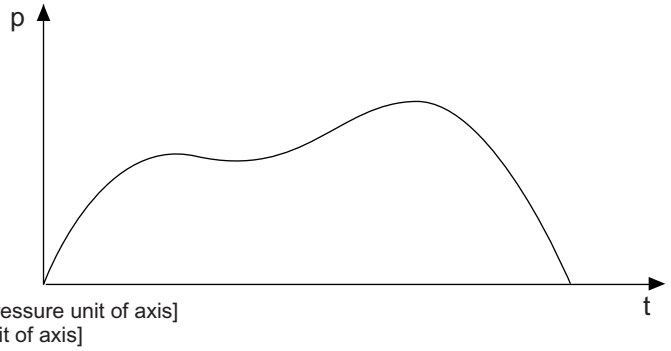


Figure 4-922 Example of a time-related limiting profile

With the axis setting **TypeOfAxis**=REAL_QFAXIS_WITH_OPEN_LOOP_FORCE_CONTROL, the profile is output as a manipulated variable on the F output.

See also

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Starting the time-related force/pressure profile

The **_runTimeLockedForceProfile()** command starts the time-related force/pressure profile.

The axis executes the function specified in the cam as a force/pressure profile.

The domain of the cam to be used in the command is interpreted as time, and the range as a force/pressure in the respective time and force/pressure units of the axis.

The profile is applied from a definable starting point to the end.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

The derivation of pressure for any necessary transition motions, e.g. for entering and exiting the profile, can be programmed in the command. The behavior at the end of the profile is set during axis configuration.

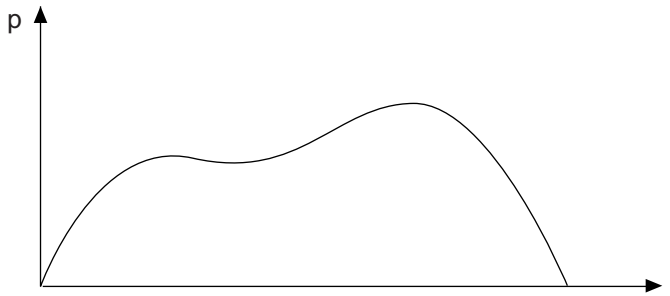


Figure 4-923 Example of a time-related force/pressure profile

The status and values are displayed in the elements of the **forceTimeProfileCommand** system variable.

If the step enabling condition is satisfied or the status is WHEN_INTERPOLATION_DONE, a pressure/time profile can switch from pressure control to position control.

See also

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Starting the position-related force/pressure profile

Possible start commands:

- **_runPositionLockedForceProfile()**
- **_runMotionInPositionLockedForceProfile()**

The axis executes the function specified in the cam as a force/pressure profile.

The definition range of the cam to be used in the command is interpreted as a position and the value range as a force/pressure in the relevant position and force/pressure units of the axis.

The profile is started at the current axis position. This must fall within the definition range of the profile, otherwise the command is aborted and an alarm is triggered.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

The derivation of pressure for any necessary transitions, e.g. for entering and exiting the profile, can be programmed in the command. The behavior at the end of the profile is set during axis configuration.

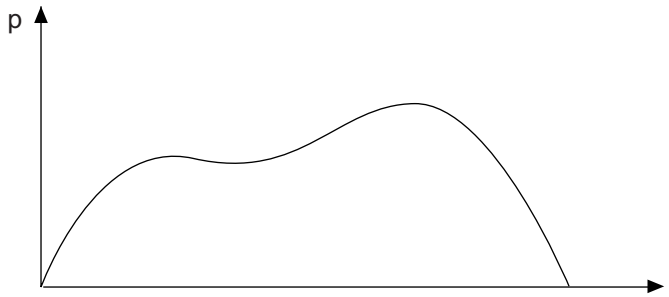


Figure 4-924 Example of a position-related force/pressure profile

The status and values are displayed in the elements of the **forcePositionProfileCommand** system variable.

See also

Traversing with user-defined motion and force/pressure profiles (Page 3040)

Enabling/disabling force/pressure limiting

The **_enableForceLimitingValue()** command enables force/pressure limiting on the axis. The limit value is specified in the command.

Active motion commands and synchronous operation relationships continue to be executed.

The limiting functions can be enabled before a motion or simultaneously with a motion. They can be switched by reissuing the command.

Active limiting disables the following error and positioning monitoring.

The **_disableForceLimiting()** command cancels the force/pressure limiting.

With the axis setting **TypeOfAxis**=REAL_QFAXIS_WITH_OPEN_LOOP_FORCE_CONTROL, a manipulated variable is output or canceled on the F output with these commands.

System variables

Table 4-401 System variables for force/pressure limiting

Variable	State	Meaning
forceLimitingCommand.state	ACTIVE / INACTIVE	Activation status of a command for the force limitation on the axis

See also

Overview of force/pressure limiting (Page 3033)

Enabling/disabling velocity limiting

The **_enableVelocityLimitingValue()** command enables velocity limiting on the axis. The limiting value is specified in the command.

If the velocity limit is reached, the **Anti-windup** functionality in the pressure controller is activated, i.e., the I component is stopped.

Velocity limiting can be enabled in parallel to motion commands.

In addition to a force/pressure specification, the velocity is limited in the case of dwell pressure, so that if an error occurs in the process, the velocity no longer increases, as is otherwise permitted.

In the event of an error, velocity limiting remains active except in the case of the RELEASE_DISABLE error reaction.

The **_disableVelocityLimitingValue()** command cancels velocity limiting.

System variables

Table 4-402 System variables for velocity limiting

Variable	Status	Meaning
velocityLimitingCommand.state	ACTIVE/INACTIVE	Status of the velocity limiting command on the axis

See also

Force/pressure control with velocity limiting (Page 3033)

Enabling velocity limiting with position-related velocity limiting profile

Possible enabling commands:

- **_enablePositionLockedVelocityLimitingProfile()**
- **_enableMotionInPositionLockedVelocityLimitingProfile()**

This command activates velocity limiting with a position-related velocity profile. The profile is set by means of a cam. The velocity limit value is not implicitly deactivated at the end of the profile.

For position-related profiles, the absolute position reference of the profile is the starting point. This also applies when modulo axes are used, thus allowing the profile to remain valid beyond the end of the modulo range.

The profile is started at the current axis position. This must fall within the definition range of the profile, otherwise the command is aborted and an alarm is triggered.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

See also

Force/pressure control with velocity limiting (Page 3033)

Enabling velocity limiting with time-related velocity limiting profile

The **_enableTimeLockedVelocityLimitingProfile()** command activates velocity limiting with a time-related limiting profile. The profile is set by means of a cam. Force/pressure limiting is not implicitly disabled at the end of the profile.

The starting point can be specified in the cam.

Behavior at the end of profile:

- Last value is still in effect
- Force/pressure limiting remains active

See also

Force/pressure control with velocity limiting (Page 3033)

Travel to fixed endstop

The **_enableMovingToEndStop( )** and **_disableMovingToEndStop( )** commands are available for the travel to fixed endstop function.

See also

Travel to fixed endstop (Page 3015)

4.8.8.4 Commands for defining the coordinate system

Resetting the set and actual positions

The **_redefinePosition()** command is used to set/offset the axis coordinate system. Either the setpoint or actual value is specified. The value that is not specified is aligned to keep the following error constant. The servo values (actual values, setpoints) are not modified.

The **_redefinePosition** command is effective in the following technology object states: Active, Motion, and Motion stop without abort.

In the case of a virtual axis, the velocity and acceleration can also be set.

The setpoint and actual values of an axis can be changed using the **_redefinePosition()** command. The position value is specified as an absolute value or as a relative position offset. The behavior can be specified by means of the **RedefineSpecification** parameter:

- **COMMAND_VALUE** setting: The setpoint of the axis position is modified.
- **ACTUAL_VALUE** setting: The actual value of the axis position is modified.
- **COMMAND_VALUE_BASIC_MOTION** setting: Like COMMAND_VALUE, but with reference to the basic coordinate system
- **COMMAND_VALUE_SUPERIMPOSED_MOTION** setting: Like COMMAND_VALUE, but with reference to the superimposed coordinate system

The other value in each case is aligned.

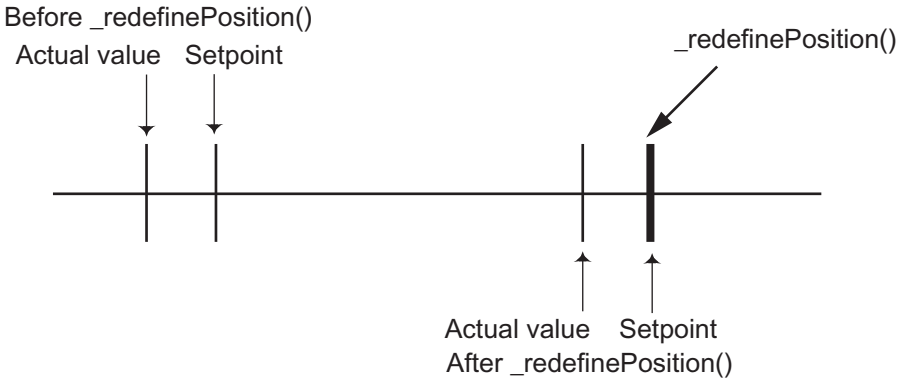


Figure 4-925 Absolute offset of axis position relative to setpoint

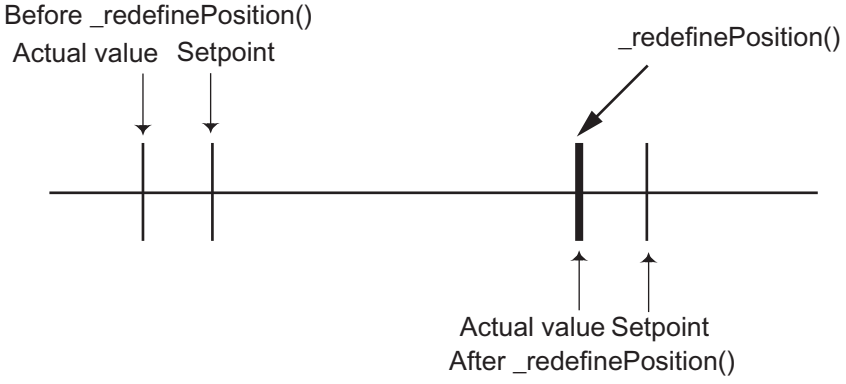


Figure 4-926 Absolute offset of axis position relative to actual value

If the set positions and/or actual positions of several axes must be re-set concurrently, this can be realized as follows:

- Programming of the **_redefinePosition()** commands for the various axes within the IPO-synchronous user task
  Requirement: For the associated axes, the appropriate IPO cycle clock must be set for this IPO-synchronous user task.
- Programming of the **_redefinePosition()** commands within a block for the synchronous processing
  (See **Motion Control Base Functions** manual, *Synchronous Start* section).

Basic coordinate system or superimposed coordinate system (as of V3.1)

_redefinePosition() can act on the basic coordinate system or the superimposed coordinate system. The behavior can be specified by means of the **RedefineSpecification** parameter.

See also

Superimposed motion (Page 3008)

4.8.8.5 Simulation commands

Enabling/disabling program simulation

The **_enableAxisSimulation()** command sets the axis to program simulation. If the axis is in motion, it is stopped using the preassigned braking ramp. As for a virtual axis, the actual values are taken from the setpoints. The axis enables and setpoint generation remain active, as do the active motion commands.

While simulation is active, the command variable is calculated and the actual value of the interpolator is set to the setpoint of the interpolator, irrespective of the real actual value of the axis. The real actual value of the axis is not affected by the program simulation.

If the axis is operated in simulation mode, a program with command variable calculation can be executed; in this case, motions are not executed by the axis, but are only simulated.

Terminate simulation.

The **_disableAxisSimulation()** command resets the axis out of program simulation. The real actual values then continue to be accepted. All other states remain unchanged. Any existing following error is removed immediately.

The following examples describe how to avoid the following error being removed immediately.

Example: Travel onwards from the actual position of axis

1. Switch the module off and on again.
2. Deactivate the simulation.

Before deactivation, the setpoint has been set to the actual value.

Example: Travel onwards from the new setpoint

1. Determine the different between the setpoint and actual value.
2. Set the setpoint to the actual value using **_redefinePosition()**.
3. Deactivate the simulation.
4. Travel the difference between the setpoint and actual value.

The **_resetAxis()** command also terminates the program simulation of the axis.

Effectiveness and status

The **_enableAxisSimulation()** and **_disableAxisSimulation()** commands are effective in the following TO states: Inactive, Active, Motion, and Motion stop without abort.

The current simulation mode can be scanned by means of the **simulation** system variable (program simulation).

Note

The outputting of setpoint values may be blocked by the simulation.

Axis errors, such as "Drive not connected" or "Encoder fault" cannot be avoided in this way; the axis must be functional same as a non-simulated axis.

System variables

Table 4-403 System variables for program simulation

Variable	State	Meaning
Simulation	ACTIVE/INACTIVE	Simulation status

See also

Setting as a real axis without drive (axis simulation) (Page 2899)

4.8.8.6 Information functions / command buffers

Overview of information functions/command buffer

The information commands on the axis can be used to read out the command and motion status of axis commands. It is possible to store the command status for longer than the period during which the command is in effect on the axis. The commands are identified with the **CommandId**. To scan the status, you need to specify the axis as well as the **CommandId** assigned to the command.

Reading the execution status of a motion command

The **_getStateOfAxisCommand()** command returns the execution status of a motion command. The status can be scanned while the command is being executed. Upon completion, the **CommandId** is deleted.

If a **CommandId** scan is required beyond this time period, it can be stored in a buffer and then cleared at any time, see **_bufferAxisCommandId** and **_removeBufferedAxisCommandId**.

The following statuses are reported:

- Command is being executed
- **CommandId** not known or command already completed
- Command is decoded, execution has not yet started
- Command is decoded, waiting for a synchronous start
- Execution of the command has been completed
- Execution of the command has been aborted

The "Execution of the command has been completed" and "Execution of the command has been aborted" statuses are only returned when the command status is stored. The **_bufferAxisCommandId( )** command can be used to store the **CommandId** and command status in the system beyond the end of the command.

For the "Execution of the command has been aborted" status (ABORTED), the reason for aborting is also given (V3.2 and higher)

System variables

None

See also

Saving the Command ID (Page 3173)

Reading current phase of motion

The **_getMotionStateOfAxisCommand()** command returns the current phase of the motion.

The following statuses are reported:

- **NOT_EXISTENT**
  CommandId not known or command already completed
  See also Saving the Command ID (Page 3173)
- **BUFFERED**
  Command is in the command queue
- **IN_EXECUTION**
  Command is being executed
- **IN_ACCELERATION**
  The motion generated by the command is in the acceleration phase
- **IN_CONSTANT_MOTION**
  The motion generated by the command is in the constant motion phase
- **IN_DECELERATION**
  The motion generated by the command is in the deceleration phase
- **AXIS_HOMED**
  Axis synchronized
- **INTERPOLATION_DONE**
  The setpoint interpolation of the command is completed
- **SYNCHRONIZING**
  Synchronizing
- **DESYNCHRONIZING**
  Desynchronizing
- **SYNCHRONIZED**
  Synchronous operation
- **MODIFICATION_ACTIVE**
  Compensating movement is active with scaling or offset in synchronous operation

- **EXECUTED**
  Execution of the command has been completed
  See also Saving the Command ID (Page 3173)
- **ABORTED**
  Execution of the command has been aborted

System variables

None

See also

Saving the Command ID (Page 3173)

Saving the Command ID

The **_bufferAxisCommandId()** command causes the command status and the **CommandId** to be stored beyond the command execution time. This permits a command status to be queried even if the command has already been completed.

In V3.2 and higher, the command includes an additional parameter that can suppress deletion of the **CommandId** on resetting.

Note

The maximum number of **CommandIds** with command status that can be stored is specified in the **decodingConfig.numberOfMaxBufferedComandId** configuration data element. This must be managed by the user program. If the buffer overflows, the command will be rejected with a return value.

Canceling saving of Command ID

The **_removeBufferedAxisCommandId()** command clears the specified **CommandId** and the command status from the buffer. Alternately, all stored **CommandIds** can be deleted.

Reading the status of a specific error on the axis

The **_getAxisErrorNumberState()** command is used to read the status of a specific error on the axis.

Reading out pending alarms (V4.0 and higher)

The error status, alarm number, and alarm parameters of up to eight pending alarms can be read out with the **_getAxisErrorState()** command.

Reading the status of the motion buffer on the axis

The information command **_getStateOfMotionBuffer()** on the **MotionBuffer** can be used to manage the buffer on the axis. This can be used to check whether the axis is ready to accept motion commands before those commands are issued. All **sequentially** effective commands are stored in the motion buffer.

The **_getStateOfMotionBuffer()** command returns the status of the motion buffer on the axis: EMPTY, FULL, or WRITEABLE.

Clearing the motion buffer on the axis

The information command and the reset command on the motion buffer can be used to manage the motion buffer on the axis. This can be used to check whether the axis is ready to accept motion commands before those commands are issued.

The **_resetMotionBuffer()** command deletes all commands in the motion buffer and the commands pending for the motion buffer. The current command is not deleted. The **_resetMotionBuffer()** command is executed synchronously. The command is active in all axis states.

Activating data sets

Data set commands can be used to read, write, and activate data sets on the axis.

The **_setAxisDataSetActive()** command sets the data set specified in the function parameter to active.

System variables

Table 4-404 System variables for activating data sets

Variable	State/type	Meaning
dataSetMonitoring	See the **StructDataSetMonitoring** data type as defined in the reference list of the system variables	Information for data set changeover • State • Requested data set • Current data set

Note

If the active measuring system is switched via a data set changeover, the **_setAndGetEncoderValue()** system function should be used to synchronize the two measuring systems before the switchover takes place. This will prevent unwanted compensating motions of the position controller if differences in position are identified.

The data sets must have the same control loop structure and must not contain a change that requires a restart. This means that when DS_1 is with DSC, DS_2 must also be with DSC.

Settings that require a restart cannot be activated via a data set changeover.

See also

Data sets (Page 3037)

Writing a data set

The **_setAxisDataSetParameter()** command can be used to overwrite an inactive data set on the axis. The data set number is specified in the command.

See also

Data set overview (Page 3037)

Reading a data set

The **_getAxisDataSetParameter()** command can be used to read any data set on the axis. The data set number is specified in the command.

The command reads the data that can be written with the **_setAxisDataSetParameter()** command.

See also

Data set overview (Page 3037)

Writing the force/pressure-specific data of a data set

The **_setForceControlDataSetParameter()** command can be used to overwrite the force/pressure-related data of an inactive data set. The data set number is specified in the command.

The following force-/pressure-related data is contained in the data set:

- Monitoring of the control deviation of the force controller
- Force controller data
  Force controller setting
  Limitations of the manipulated variable of the force controller
  PID controller setting
  Controller type
  Effective direction of the manipulated variable (can be switched with V4.2 and higher)
  Inversion of the force controller manipulated variable

Detailed information about the parameters can be found in the TP Cam System Functions Reference List.

Note

The force/pressure-specific data of the data set cannot be written on the speed-controlled axis.

Reading the force/pressure-specific data of a data set

The **_getForceControlDataSetParameter()** command can be used to read the force/pressure-related data in the data set. The data set number is specified in the command.

The command reads the data that can be written with the **_setForceControlDataSetParameter()** command.

Note

The force-/pressure-specific data of the data set cannot be read on the drive axis.

Writing the data of the data set (hydraulic functionality only)

The **_setQFAxisDataSetParameter()** command can be used to overwrite the specific parameters for the axis with hydraulic functionality in an inactive data set. The data set number is specified in the command.

The following specific parameters of the axis with hydraulic functionality are contained in the data set:

- Inversion of F output
- Dynamic response adaptation to the axis with hydraulic functionality
- Data for the speed controller on the drive axis with hydraulic functionality

Detailed information about the parameters can be found in the TP Cam System Functions Reference List.

Note

The specific data for the axis with hydraulic functionality in the data set cannot be written on the drive axis.

Reading the force/pressure-specific data of the data set (hydraulic functionality only)

The **_getQFAxisDataSetParameter()** command can be used to read the specific parameters for the axis with hydraulic functionality in the data set. The data set number is specified in the command.

The command reads the data that can be written with the **_setQFAxisDataSetParameter()** command.

Note

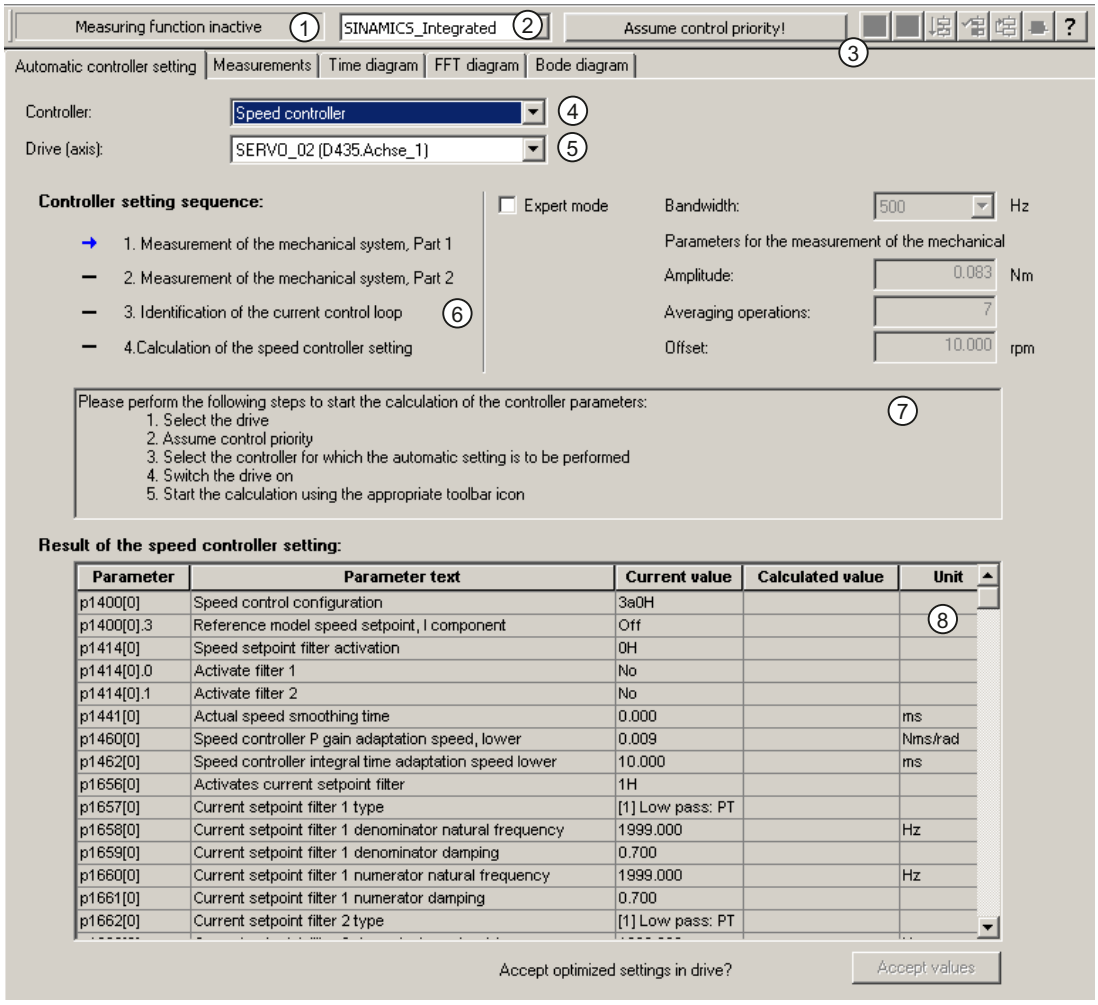
The specific data for the axis with hydraulic functionality in the data set cannot be read on the speed-controlled axis.

Command for calculating a braking distance

With the **_getAxisStoppingData()** command, the system calculates the braking distance as a function of a specified actual velocity, actual acceleration, motion profile, and dynamic response parameters.

4.8.8.7 Assigning automatic controller optimization

Here, you operate the automatic controller optimization. To perform the automatic controller setting, you must have an ONLINE connection to the drive device of the relevant drive.



1 Measuring functions status display
2 Drive unit
3 Operator buttons
4 Controller selection
5 Drive selection
6 Progress message box
7 Description of current step
8 Result display

Figure 4-927 Automatic optimization - Example: speed controller

The following operator input options and displays are available:

Field/Button	Meaning/Instruction
Measuring functions status display	This area shows whether or not the drive is active in the step. The following information is displayed: • Measuring function active Drive/axis is moving • Measuring function inactive Drive/axis is not moving
Drive unit	This combo box offers all the SINAMICS drive units in the open project that contain drives operated in the "SERVO" closed-loop control type and that the motor measuring system uses as encoders. Selection is possible provided the master control has not yet been assumed.
Assume control priority/ **Give up control priority**	This allows you to assume master control in order to perform automatic controller setting. Master control can be released only once the drive has been switched off.
Operator buttons	

Field/Button		Meaning/Instruction
		**Drive ON** This button can be pressed if master control has already been assumed. This enables the infeed and assigns the drive enable. This is not the same as a POWER ON of the hardware. Afterwards, the controller setting can be started. To cancel and then restart the function, perform the following operating input sequence: Switch off drive -> Switch on drive
		**Drive OFF** You can use this button to reset the functions initiated by *Drive ON*. This enables the infeed and inhibits the drive enable. This is not the same as a POWER OFF of the hardware. If it is detected that calculated parameters are present (the last step has been completed) that have not yet been transferred to the drive/axis, a confirmation prompt will be issued, asking whether the controller setting is now complete and, as a result, the calculated parameters should be discarded.
		**Automatic execution** Automatically performs all steps of the controller setting starting with the current selection on the step display. This button will be deactivated once the last possible step has been performed.
		**Execute step** The single step of the controller optimization selected on the step display will be performed and the step display then incremented by one step. This button will be deactivated once the last possible step has been performed.
		**Step back** Sets the step display back one step. If the step display is at step 1, the button is deactivated.
		**Cancel step** Cancels the execution of the current step; the step display remains at the canceled step.
		**Help** Opens the help for this screen form.
Controller selection		In **Controller selection**, you can select which controller to set. The view on the screen form will then show the appropriate controller. The combo box contains the following selection options: • "Speed controller" (always available) • "Position controller (DSC)" (offered only if SCOUT is installed on the PC). Before the setting is applied, a few call conditions are checked and any unfulfilled conditions are displayed. The selection can be changed at any time, provided no step is currently being executed.

Field/Button	Meaning/Instruction
Drive selection	In **Drive selection**, a drive corresponding to the setting in the **Drive unit** field can be selected. The following rules apply for the "Drive" input field: • The field is labeled "Drive". If SCOUT is installed on the PC, it is called "Drive (axis)". • If SCOUT is installed and the drive is connected to at least one axis, the connected axis is appended to the drive name in brackets, in the form "(<device>.<axis name>)". If the drive is to be connected to multiple axes, "..." is appended to the first axis name. • The tool tip for the Drive field contains all names in the syntax shown above for the axes connected with the drive. The field can only be operated provided the master control has not yet been assumed. As soon as the "Drive selection" field registers a change, the parameters of the newly selected drive are registered in the "Current value" column in the result display for a cyclic update.
Progress message box	The step control presents the status overview of the steps of the "automatic controller setting" on the left.
	The step has been executed.
	Selection of the step that will be executed next or is currently being executed.
	Step has not yet been executed.
Expert mode	Checkbox If the checkbox for **Expert mode** is selected, the **Amplitude**, **Bandwidth**, **Averaging operations**, and **Offset** input fields are displayed. The user can enter special settings for the next optimization step in these fields. The checkbox is deselected as the default setting. If the checkbox is deactivated, the parameter settings determined by the optimization tool are used for the measurement.
Kv factor (controller = position controller)	As a result of step 2 of the position controller optimization ("Calculation of the position controller setting"), the determined **Kv factor** is indicated. Provided no new value has been calculated, the display shows "–". This is also the standard display when the screen form starts.
Select axes and data sets button "..." (controller = position controller)	This button can be used to call a dialog that shows the possible axes (in accordance with the relevant requirements and supplementary conditions) and their axis data sets in which the Kv factor can be accepted.
Result display	This table displays the parameters whose values were determined during the controller optimization. The **Current value** column displays the current value of the parameter read out cyclically from the target device. The **Calculated value** column displays the value of the parameter determined from the controller setting. Provided no new value has been calculated, the "–" character is displayed instead of the value. This is also the standard display when the screen form starts.

Field/Button	Meaning/Instruction
Accept (controller = speed controller)	The **Accept** button is active only when step 4 of the speed controller optimization has been completed successfully. The calculated values are applied to the drive.
Accept (controller = position controller)	The **Accept** button is active only under the following conditions: • Step 2 for the position controller optimization must have completed successfully. • An online connection to the associated SIMOTION device must exist. • The axis is online-consistent. • The axis data set table contains at least one data set. The calculated values are applied to the drive.

Note

Emergency cancelation of automatic setting with <spacebar>

The following actions are performed:

- The step currently being executed is canceled
- The drive is disabled

See also

Overview of automatic controller setting (as of V4.1 SP1) (Page 3066)

4.8.8.8 Technological alarms

Alarm reactions

The reactions of the axis to **technological alarms** (local reaction) are implemented by the system.

The local reactions are preset on an alarm-specific basis or they can be set in the alarm configuration.

The following local reactions on the axis are available:

- **NONE**
  - No reaction
  - The responses can be specified by the user in the TechnologicalFaultTask.
- **DECODE_STOP**
  - Command processing aborted
  - The current motion (thus, for example, also the MotionIn command) and commands in the motion buffer are still executed.
  - New motion commands are rejected.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
- **END_OF_MOTION_STOP**
  - Abort at the end of the command causing the error
  - Stop of the motion at the end of the command
  - The current motion command (thus also the MotionIn command) is still executed.
  - New motion commands are rejected.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
- **MOTION_STOP**
  - Controlled motion stop with programmed ramp values.
  - The current motion command (thus also the MotionIn command) is stopped.
  - New motion commands are rejected.
  - Motion commands are not canceled and are resumed when the error reaction is acknowledged.
  - No more new motion commands are accepted.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
- **MOTION_EMERGENCY_STOP**
  - Controlled motion stop with maximum ramp values/limits for the axis.
  - The current motion command (thus also the MotionIn command) is stopped.
  - New motion commands are rejected.
  - Motion commands are not canceled and are resumed when the error reaction is acknowledged.
  - No more new motion commands are accepted.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.

- **MOTION_EMERGENCY_ABORT**
  - Controlled motion stop with maximum axis dynamic values (starting from the current setpoint).
  - Active commands (IPO) are aborted and signaled back to the user program with an error code.
  - The drive remains active.
  - The position controller remains active.
  - The IPO remains active.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
  - After the error is acknowledged with **_resetError()**, the axis can travel again. (provided that the cause of the error no longer exists)
- **FEEDBACK_EMERGENCY_STOP**
  - Motion stop with preassigned braking ramp.
    See Stopping with preassigned braking ramp (Page 3002).
  - Manipulated variable superimposition not effective.
  - Active commands (IPO) are aborted and signaled back to the user program with an error code.
  - The drive remains active.
  - The position controller remains active.
  - The IPO goes into follow-up mode, preassigned braking ramp servo (time specification in the servo for preassigned braking ramp)
    IPO and servo are then active again.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
  - After the error is acknowledged with **_resetError()** and the axis is reset with **_resetAxis()**, the axis can be traversed further (provided the cause of the problem no longer exists).

- **OPEN_POSITION_CONTROL**
  - Motion stop with speed setpoint zero and abort
  - The position control loop is decoupled from the setpoint circuit.
  - Active commands (IPO) are aborted and signaled back to the user program with an error code.
  - The position controller enable is canceled.
  - The control unit transmits the setpoint to the drive.
  - The drive stops (because of setpoint 0).
  - The drive remains active.
  - The Ipo goes into follow-up mode.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
  - After the error is acknowledged with **_resetError()** and the drive and position controller are activated with **_enableAxis()**, the axis can travel again (provided that the cause of the problem no longer exists).
- **RELEASE_DISABLE**
  - Motion disable with controller disable and abort of all commands
  - The drive enables are canceled.
    See also Coupling of digital drives (Page 2895) and Adjustable response to RELEASE_DISABLE (Page 3185).
  - The controller enables are canceled.
  - Active commands (IPO) are aborted and signaled back to the user program with an error code.
  - Position controller and Ipo go into follow-up mode.
  - Further reactions can be specified by the user in the TechnologicalFaultTask.
  - After the error is acknowledged with **_resetError()** and the drive and position controller are activated with **_enableAxis()**, the axis can travel again (provided that the cause of the problem no longer exists).

    Note

    Active commands are commands in the IPO or commands that are being executed; these commands are no longer in the motion buffer.

    When you acknowledge an error with **_resetAxisError()**, you must delete any commands in the motion buffer with **_resetMotionBuffer()**.

    The preset responses of the technology alarms can be overwritten by higher-priority stop responses.

    Global alarm reactions can be set on the technology object (execution system, FaultTasks).

See also

Jerk limitation for local stop response (V3.2 or higher) (Page 3007)

Adjustable response to RELEASE_DISABLE

The drive response to the RELEASE_DISABLE alarm response can be set (with a digital drive link).

See Coupling of digital drives (Page 2895).

See also

Alarm reactions (Page 3181)

Toleration of the failure of an encoder not involved in closed-loop control (V4.0 and higher)

If an encoder that is not involved in closed-loop control fails, the system should neither stop the axis nor disable the drive.

In the following cases, an encoder is not involved in closed-loop control:

- If the encoder is not selected.
  That is, one of the 8 possible encoders of the axis is configured and active (running) but it is not the selected encoder at the moment.
- If the encoder is selected but not involved in closed-loop control.

This behavior is set using the
typeOfAxis.numberOfEncoders.Encoder_1.sensorControlConfig.tolerateSensorDefect
configuration data element.

As of SINAMICS V2.6.2, the failure of an encoder not involved in closed-loop control is indicated by alarm 20015.

4.8.9 Part IV External Encoder - Description

4.8.9.1 External encoder overview

The External Encoder technology object ("ExternalEncoder") records a position and provides the position to the control. The determined position can be evaluated in the user program.

The reference of the encoder position to a defined position is produced by parameterization of the mechanical properties and a homing procedure.

The position is stated independently of the encoder type and depending on the selected unit system:

- **Linear unit system**
  The position is given as a length dimension, e.g. millimeters (mm).
- **Rotary unit system**
  The position is given as an angle dimension, e.g. degrees (°).

Information from the External Encoder technology object can be used as input variables for other technology objects, such as the Synchronous Operation technology object, Output Cam technology object, or Measuring Input technology object, or to display the actual value of an external motion.

The External Encoder technology object can be used with

- Incremental encoders
- Absolute encoders
- Resolvers
- Encoders on direct value/analog value
- Encoders on count value
- ...

An external encoder can be used only on one position-related encoder.

The external encoder is referred to by the data type **externalEncoderType** in reference lists and when programming.

From V3.2 onwards, the External Encoder technology object is included in the Cam and Cam_ext technology packages.

Interconnections

The External Encoder technology object can be interconnected with the following technology objects:

- **Synchronous Operation technology object**
  The External Encoder technology object is used to provide a master value.
- **TO output cam**
  The External Encoder technology object is used to provide a reference value.
- **TO cam track**
  The External Encoder technology object is used to provide a reference value.
- **TO measuring input**
  The External Encoder technology object is used to provide a reference value.

Programming commands/functions for the External Encoder technology object

The MCC and ST programming languages are available for programming external encoders.

See the *SIMOTION MCC* Programming Manual or the *MCC programming language* in the online help.

4.8.10 External Encoder Fundamentals

4.8.10.1 Actual values for the external encoder technology object

Each position actual value possible on the technology object type can be used as the actual value of the external encoder technology object.

These are, for example:

- Actual value (encoder1 or encoder2) from a PROFIdrive axis telegram
  - Encoder from a PROFIdrive axis telegram to a SINAMICS drive object
    With SINAMICS it is possible to configure multiple encoders, from which a maximum of two encoder values can be transmitted in each axis telegram.

    Note

    An encoder in a PROFIdrive axis telegram can only be used for an external encoder TO or for the encoder of a TO axis in cyclical operation if a TO axis is also created on the PROFIdrive telegram and is not deactivated.

  - Encoder from a PROFIdrive axis telegram to ADI4 (at least one axis must be created on the ADI4)
  - Encoder from a PROFIdrive axis telegram to IM174 (at least one axis must be created on the IM174)
  - Encoder from a PROFIdrive axis telegram to a non-SINAMICS drive
- Actual value from a PROFIdrive encoder telegram
  - Encoder from a PROFIdrive encoder telegram to a SINAMICS encoder object, e.g. to an SMC module
  - Encoder from a PROFIdrive encoder telegram to a PROFIBUS encoder module, e.g. SIMODRIVE sensor
  - Encoder from a PROFIdrive encoder telegram to a PROFINET encoder module, e.g. MC encoder
  - Encoder from a PROFIdrive encoder telegram to any other encoder module
- Actual value as a direct value in the I/O area
- Only C2xx: Actual value via encoder signal evaluation directly at the SIMOTION CPU

For further information, refer to the relevant hardware manuals.

4.8.10.2 Encoder mounting type

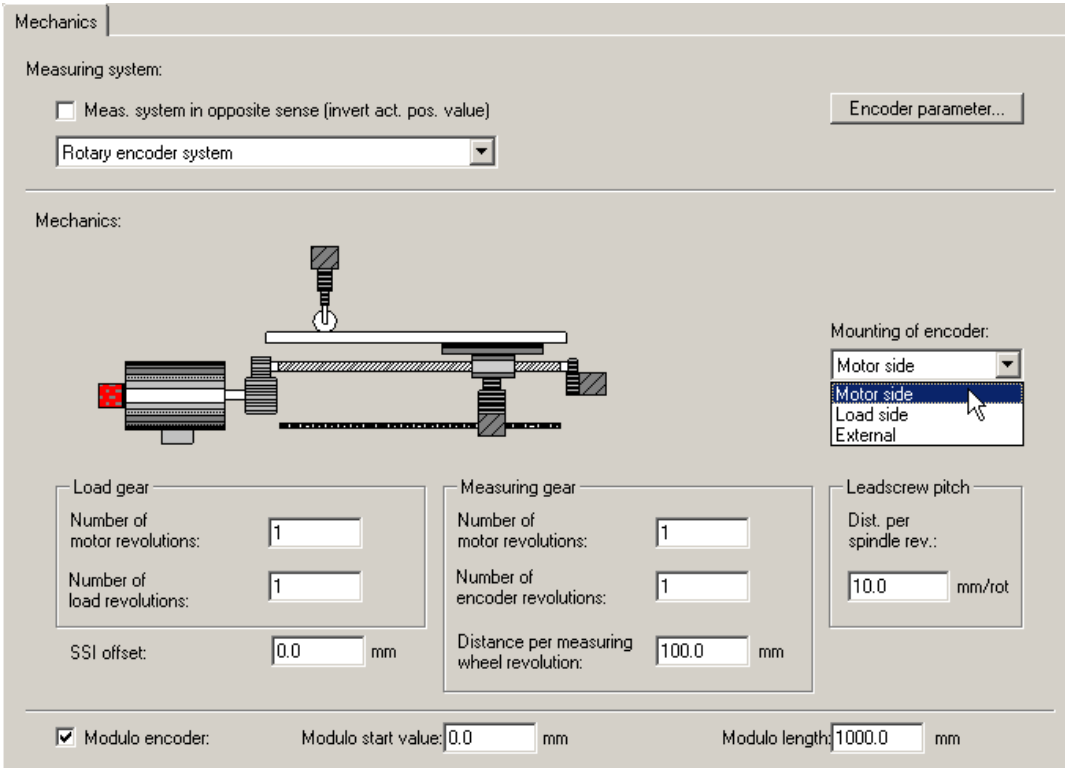


Figure 4-928 Definition of the mounting type of an external encoder in SIMOTION SCOUT

Mounting types for the external encoder

- On the drive side
- On the load side
- External

The transmission ratios must be entered via configuration data:

- For "drive side" mounting using the **adaptdrive** configuration data element
- For "load side" mounting using the **adaptload** configuration data element
- For "external" mounting using the **adaptextern** configuration data element
- For linear encoder system mounting, the transmission ratio is always 1.

4.8.10.3 Encoder for position

The controller uses an encoder to detect the axis position.

From a technological standpoint, the following encoder types can be differentiated:

- Incremental encoder
  On the controller side, only the difference between two read encoder values is evaluated. Values are read out in the servo cycle clock at equal intervals. In order to determine the mechanical axis position, the axis must be homed each time the system is switched on. Position zero is displayed after it is switched on.
- Absolute encoder
  This encoder supplies the absolute value or, in the case of an absolute value in the PROFIdrive message frame, the absolute value is read one time after the system is switched on. After this, the actual value is processed in the same way as with the incremental encoder.
  The absolute value supplied by the encoder is assigned to the associated mechanical axis position by means of the absolute encoder adjustment. The absolute encoder adjustment occurs only one time, and the controller remembers the compensation value/absolute encoder offset even after it is powered on/off.
  Certain situations, such as an encoder failure, a restart, etc., can require an absolute encoder re-adjustment. For more information, refer to the chapter titled *Absolute encoder homing/ Absolute encoder adjustment*.
  The following absolute encoder types are differentiated:
  - Absolute encoder with absolute encoder setting
    The measuring range of this encoder is greater than the axis traversing range.
    The axis position results directly from the current encoder value since this can be uniquely mapped.
    An offset may be entered - it is not necessary to hold overflows within the controller.
    There are no overflows of the absolute actual value stored when SIMOTION is switched off. The next time it is switched on, the actual position value is generated from the absolute actual value only.
  - Absolute encoder with cyclic absolute encoder setting
    The axis traversing range is greater than the range of values measured by the encoder, and the encoder returns an absolute value within its measuring range.
    The controller also counts the number of measuring ranges internally in order to hold a unique actual axis position beyond the range of measured values.
    When SIMOTION is switched off, the overflows of the absolute actual value are stored in the retentive memory area of SIMOTION. The next time it is switched on, the stored overflows are taken into account for the calculation of the actual position value.
    The actual position of the axis is passed internally in a 64-bit integer variable.
    Example of a single-turn encoder with 4,096 increments:
    The position of the encoder is mapped in bits 0 to 11, and the number of overflows of the encoder range in bits 12 to 63.
    Example of a multi-turn encoder with 4,096 x 4,096 increments:
    The position of the encoder is mapped in bits 0 to 23, and the number of overflows of the encoder range in bits 24 to 63.
    The overall position of the axis is retained after the controller is switched off. When the controller is switched on again, if the actual encoder value does not match the actual position stored in the controller, it will be corrected to a maximum of ± ½ the encoder measuring range.

    Note

    If the axis/encoder is moved by more than one half the encoder measuring range with the controller switched off, the actual value in the controller no longer matches the real axis.

See also

Absolute encoder homing / absolute encoder adjustment (Page 2934)

4.8.10.4 Encoder for velocity

Encoders for detection and display of the speed/velocity can only be applied to speed-controlled axes.

Options are:

- Incremental encoders/absolute encoders with number of increments or pulses/revolution (for electric axes)
- Interval counters (for hydraulic axes)
- Encoders providing the velocity as a direct value in the I/O area (for hydraulic axes)
- Read-out of the speed from the PROFIdrive message frame and provision for technological functionality, e.g. velocity monitoring

4.8.10.5 Encoder assignment and terminology

The encoder type is specified with the encoder mode.

Table 4-405 Definable encoder mode depending on the encoder type

Encoder mode	Encoder type		
	Absolute encoder	Cyclic absolute encoder	Incremental encoder
PROFIdrive (with adaptation [1]) (as of V4.2)	x	x	x
Endat (encoder data interface)	x	x	x
SSI (synchronous serial interface)	x	x	-
Sinusoidal	-	-	x
Rectangle	-	-	x
Resolver	-	x [2]	x
Analog encoder (value in the I/O area)	x	-	-

[1] The adaptation setting (set by the system by default with V4.2 or higher and SINAMICS) makes changes based on the encoder set in SINAMICS. The encoder resolution and fine resolution are adopted during runtime.

[2] Possible with single-pole-pair resolver only

When the encoder signals are evaluated in an encoder module or drive, the actual values are transferred to the controller in the normalized format according to the PROFIdrive profile. The type of measured value acquisition is dependent on the drive or encoder module, e.g. Sinus, SSI or Endat. The PROFIdrive encoder mode can therefore be set in the controller (as of V4.2).

As of SIMOTION V4.2, the system makes the settings for communication between SIMOTION and SINAMICS drive (encoder). The project setting **Use symbolic assignment** is activated by default for new projects as of V4.2. Telegrams are set automatically. Encoder characteristics such

as fine resolution, grid spacing, and the absolute value for data width are adapted automatically when the system ramps up.

For encoder data, refer to the data sheet or type plate of the encoder. With SINAMICS, encoder data can be transferred from the drive.

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

Encoder pulses per revolution

The encoder pulses per revolution is specified on the encoder type plate as the number of signal periods per revolution (incremental encoder: Pulses/rev; absolute encoder: Pulses/rev; resolver: Number of pole pairs (for SINAMICS and MASTERDRIVES)).

Configuration data:

- **AbsEncoder.absResolution**
- **IncEncoder.incResolution**

Grid line spacing (linear encoder system)

The grid line spacing is specified on the type plate of the encoder as the distance between lines on the linear measuring system (linear scale).

Configuration data:

- **Resolution.distance**

Fine resolution

The fine resolution of the actual value is the interpolation result of a signal period of an encoder pulse.

The fine resolution steps are generated by the measuring electronics from the raw signal of the encoder pulses. Factors as a multiple of 2 are possible.

Example:

- A rectangle signal has a fine resolution of 1
- Two rectangle tracks (TTL signal) offset by 90° have a fine resolution of up to 4
- Depending on the measuring electronics, a sinusoidal signal can have any fine resolution, in principle, e.g. 2,048

Depending on the defined encoder type, the default value 0 is interpreted differently in SIMOTION (see table: *Default settings for fine resolution in SIMOTION*).

In SIMOTION, the multiplication factor is specified, rather than the shift factor/number of bits (x).

The actual value, including the fine resolution, is indicated in the **sensorData.incrementalPosition** system variable.

Configuration data:

- **AbsEncoder.absResolutionMultiplierCyclic**
- **IncEncoder.incResolutionMultiplierCyclic**

Data width of absolute value (without fine resolution) for absolute encoders

The data width of the absolute value (without fine resolution) is a result of the sum of the bits for representing the number of encoder pulses and the bits for representing the maximum number of revolutions that can be registered by the encoder according to the type plate.

Example:

4,096 pulses/rev (= 2^{12}) and a maximum of 4,096 revolutions that can be registered yields a 12 + 12 = 24-bit data width of the absolute value.

Configuration data:

- **AbsEncoder.absDataLength**

Fine resolution of absolute value in Gn_XIST2.

This parameter for the format of the Gn_XIST_2 is only relevant for encoders via PROFIdrive telegram (for more information, see *Encoder interconnection via PROFIdrive telegram*).

Configuration data:

- **AbsEncoder.absResolutionMultiplierAbsolute**

The fine resolution of the absolute value in Gn_XIST2 must be less than or equal to the fine resolution of the absolute value in Gn_XIST1.

Default setting for fine resolution.

Depending on the encoder mode, the default settings are evaluated by the system as described in the table below. The default settings are used if 0 is assigned to the value.

Table 4-406 Default settings for fine resolutions in SIMOTION

Encoder type	Encoder mode	Fine resolution (Gn_XIST1)	Fine resolution on the absolute value (Gn_XIST2)
Onboard C2xx			
Incremental encoder	Rectangle	4	-
	Stepper motor	1	-
Absolute encoder	SSI	1	1
Encoder in PROFIdrive axis telegram (applies to SINAMICS, SIMODRIVE 611U and MASTERDRIVES)			
Incremental encoder	Rectangle	2048	-
	Sinusoidal	2048	-
	Resolver	2048	-
	Endat	2048	-
Absolute encoder	Endat	2048	512
	SSI	1	1
...Cyclic absolute	Resolver (only pole pair number 1 possible)	2048	512
	Endat	2048	512
	SSI	1	1
PROFIBUS absolute encoder in PROFIdrive encoder telegram			
Absolute encoder	SSI	$2^{(32 - \text{Number of data bits})}$	1

These settings are aligned to the default parameter settings of the corresponding Siemens devices. If the behavior is different, alignment with the encoder should be performed by making a corresponding entry in the configuration data of the technology object or in the parameters of the drive or encoder. Depending on the device it may be necessary to assign the corresponding parameters in the drive or encoder with the value of the exponent (shift factor).

Device-specific features for Masterdrives with Endat encoders:

For Masterdrives with Endat encoders, Endat or SSI can be selected in the encoder mode. However, the fine resolution must always be configured in the axis wizard of the Axis or External Encoder technology object differently from the default settings (see Encoder list (Page 3193)).

Default setting:

- Fine resolution (**~Encoder.~ResolutionMultiplierCyclic**) = 0
- Fine resolution of the absolute encoder in Gn_XIST2 (**AbsEncoder.absResolutionMultiplierAbsolute**) =0
- If encoderMode=PROFIDRIVE, the values are evaluated as available, since the correct values are adapted by the drive. A default setting of 0 is not permitted for this setting.

See also

Encoder list (Page 3193)

4.8.10.6 Encoder list

For the most up-to-date list of encoders you can use with SIMOTION in conjunction with SINAMICS, SIMOVERT-MASTERDRIVES, and SIMODRIVE 611U, go to **http://support.automation.siemens.com/WW/view/de/18769911**.

The encoder list is also documented under *FAQs > Drives > Parameters of the connectable encoders* in the *SIMOTION Utilities & Applications* and in the online help (look for encoder parameter assignment in the index). *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

4.8.10.7 Onboard encoder interface on SIMOTION C2xx

Incremental encoders with TTL signal and absolute encoders with SSI protocol can be connected directly to C230-2 or C240. (See the C230-2 and C240 operating instructions and Encoder list (Page 3193))

See also

Encoder assignment and terminology (Page 3190)

Encoder list (Page 3193)

4.8.10.8 Encoder interface using the PROFIdrive message frame

As of SIMOTION V4.2, the system sets up communication between SIMOTION and SINAMICS drive (encoder). For SINAMICS drives and encoders with V4.2 and higher, encoder resolution data is transferred directly from the drive during runtime. Telegrams are set automatically.

The rest of this chapter is only relevant if the project default setting **Use symbolic assignment** is deactivated as of V4.2, or if you are working with a project which uses SIMOTION V4.1.

The encoder values are transmitted in the PROFIdrive telegram (see Table *Telegram types* in Section *Setting as a real axis with digital drive coupling*).

Encoder forced values, status values, and actual values are transmitted in the PROFIdrive telegram.

The encoder behavior on SIMOTION is set as represented in the PROFIdrive protocol.

The encoder parameters are defined via the drive wizard during drive configuration (either user-defined or by selecting the encoder).

Encoder parameters that are entered subsequently in the SIMOTION axis wizard must match the encoder parameters in the drive.

Note

For SINAMICS drives with earlier versions than SIMOTION V4.2, it is possible to transfer the encoder parameters from the drive. When assigning encoders in the axis wizard, click **Data transfer from the drive**.

If you are using a DRIVE-CLiQ component with electronic type plate (e.g., SMI motor, DRIVE-CLiQ encoder) you must first upload the parameters from the drive and save them in the project (online commissioning). If the online commissioning is carried out at a later point, you can work with the default settings of the axis wizard in the meantime during offline configuration. Once online commissioning is complete, upload the drive parameters, save them in the project, run the axis wizard again, and perform the **Data transfer from drive** function.

If you change the encoder data in the drive, you must perform an alignment again in the axis wizard.

Further sources of information:

- Encoder list (Page 3193)
- User Manual *SIMODRIVE sensor Absolute encoder with PROFIBUS-DP*
- Operating Instructions *Absolute encoder with PROFINET IO* (Chapter *Operating with SIMOTION*)

Encoder value via PROFIdrive axis telegrams

For further information, refer to the commissioning manuals for the drives.

The first and (if present) second encoder of the PROFIdrive axis telegram can be assigned freely to an External Encoder technology object or to the encoder of an axis.

Encoder value via PROFIdrive encoder telegram 8x

PROFINET/PROFIBUS encoders can be set in accordance with the current specifications of the encoder profile with telegram type 81 and as of V4.2, with telegram type 83. These encoders can be assigned freely. See also **PROFINET/PROFIBUS absolute encoder via PROFIdrive encoder telegram** in the next chapter.

Inconsistent configuration

In the event of errors or inconsistencies between the configuration data in SIMOTION and the parameter settings for the encoder in the drive, a technological alarm is triggered as soon as an online connection is established between the control and the drive/encoder.

As off SIMOTION Runtime V4.4: **Technological alarm 20025** with applicable reason

In SIMOTION Runtime < V4.4: **Technological alarm error 20005: Device type:2, log.address:1234 faulty. (Bit:0, reason: 0x80h)**

For PROFIdrive encoders and encoders on the axes according to PROFIdrive, a comparison of the parameter assignment takes place via the following drive/encoder parameters:

P979 (SensorFormat) according to PROFIdrive, which contains information about the type, resolution, and shift factors.

For drives or encoders that do not support parameter P979, the configuration data is evaluated as valid without alarm message.

Actual value Gn_XIST1

The incremental actual value is transferred cyclically with the defined fine resolution in Gn_XIST1. The incremental actual value in Gn_XIST1 is steadily continued according to the actual value change and reset when the data width of Gn_XIST1 is exceeded. If operating with incremental and absolute encoders, the control evaluates the incremental actual value in Gn_XIST1 according to the settings made for encoder pulses per revolution and fine resolution, or grid line spacing for linear scaling.

When the controller is switched on, the fine resolution value within one encoder signal period is indicated correctly in Gn_XIST1. The initial value for the number of signal periods is set by the drive/encoder, and the actual value can then be steadily continued from this initial value.

In the PROFIdrive profile, the fine resolution is given as "shift factor" (x).

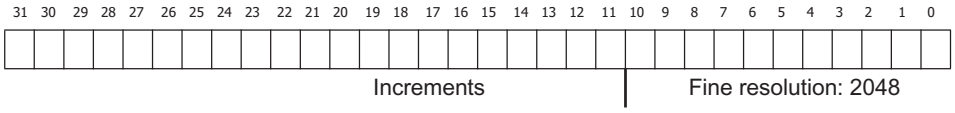


Figure 4-929 Example composition of the 32-bit encoder data of the cyclic actual value Gn_XIST1

Example of for an encoder with number of encoder pulses = 2048 (data width, 11 bits)

The fine resolution in SIMOTION in the **Inc/AbsResolutionMultiplierCyclic** configuration data element is set to the default setting 0 and is thus evaluated as a default fine resolution of 2048 (the default value depends on the encoder mode setting; see Table *Default settings for fine resolution in SIMOTION*).

SIMODRIVE 611U:

Table 4-407 Settings

SIMOTION		611U	
Encoder pulses per revolution [1]	=2048	P1007	=2048
Fine resolution [2]	=0 (≡ 2048)	P1042	=11

[1] Inc/AbsEncoder.Inc/AbsResolution

[2] Inc/AbsEncoder.Inc/AbsResolutionMultiplierCyclic

SINAMICS:

Table 4-408 Settings

SIMOTION		SINAMICS	
Encoder pulses per revolution [1]	=2048	P408	=2048
Fine resolution [2]	=0 (≡ 2048)	P418	=11

[1] Inc/AbsEncoder.Inc/AbsResolution

[2] Inc/AbsEncoder.Inc/AbsResolutionMultiplierCyclic

Note the information about the SINAMICS alignment.

Actual value Gn_XIST2

If the positions for the measuring input or homing functions are transferred to Gn_XIST_2 (n = 1 or 2, number of encoder), they are transferred with the fine resolution defined for the encoder. When the absolute value is read, the value in Gn_XIST_2 is evaluated based on the settings for the data width of the absolute value (without fine resolution) in **AbsEncoder.absDataLength** and the fine resolution absolute value in Gn_XIST2 is evaluated in **AbsEncoder.absResolutionMultiplierAbsolute**.

The fine resolution of the absolute value in Gn_XIST2 indicates the fine resolution factor included in the absolute value transfer. This can match the fine resolution of the actual value, but it can also be smaller, for example, if the 32-bit data width in Gn_XIST2 is not sufficient for the entire fine resolution factor as a result of the data width of the absolute value (without fine resolution).

Example:

Encoder pulses per revolution = 2048 (11 bits) and multi-turn resolution of 4,096 revolutions (12 bits)

Thus, the data width of the absolute value without fine resolution is 11 bits + 12 bits = 23 bits.

Therefore, 9 bits remain for the fine resolution in Gn_XIST2 (32 bits - 23 bits = 9 bits). The setting 0 for the fine resolution of the absolute value in Gn_XIST2 is thus evaluated by the system as 512 (= 9 bits).

Table 4-409 Setting the encoder data

Encoder pulses per revolution [1]	2048
Data width of absolute value (without fine resolution) [2]	23

Fine resolution of absolute value in Gn_XIST2 [3]	0 (= 512)
Fine resolution [4]	0 (= 2048)

[1] AbsEncoder.AbsResolution

[2] AbsEncoder.absDataLength

[3] AbsEncoder.absResolutionMultiplierAbsolute

[4] AbsEncoder.AbsResolutionMultiplierCyclic

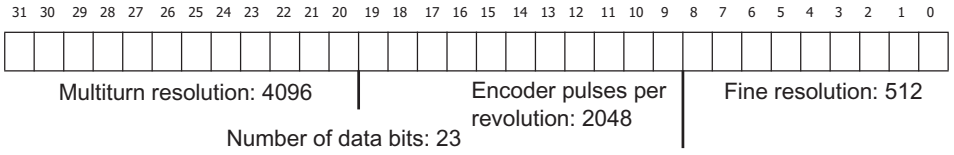


Figure 4-930 Example composition of the 32-bit encoder data of the absolute actual value Gn_XIST2

The number of bits resulting from the data width of the absolute value (without fine resolution) and the number of data bits for the fine resolution of the absolute actual value must not exceed 32. If it is less than 32, leading zeroes are added in Gn_XIST2.

Resolver in PROFIdrive axis telegram

For SINAMICS and MASTERDRIVES, parameters are assigned for the pole pair number of the resolver rather than the encoder pulses per revolution (example: 8-pole resolver = 4 pole pairs → input value = 4).

With SIMODRIVE, parameters are assigned for the encoder pulses per revolution based on parameter P1011.2

As of V4.1 SP1, the resolver with pole pair number 1 is supported as an absolute encoder with the cyclic absolute setting. (encoder pulses per revolution = 1, data width of the absolute value = 0, default value evaluation: fine resolution = 2048, fine resolution Gn_XIST2 = 512)

When a 1-pole resolver is used as Endat encoder, the p418(XIST1) and p419(XIST2) parameters must be set to "11" to prevent information loss of the absolute position. (Settings on the axis: absolute encoder, cyclic absolute, Endat, line count = 1, fine resolution = 2048, fine resolution absolute value = 2048, data width = 0)

See also the Encoder list (Page 3190).

PROFINET/PROFIBUS absolute encoder via PROFIdrive encoder telegram

The data width of the encoder value in the technology object configuration data in SIMOTION must match the parameter settings for the PROFINET/PROFIBUS absolute encoder in HW Config.

See also the Encoder list (Page 3190).

Example:

Parameter settings of a PROFIBUS absolute encoder in HW Config with 24-bit data width of the absolute value.

The 'SIMODRIVE isochronous sensor' PROFIBUS absolute encoder in HW Config is defined according to the default setting for 24-bit data width and encoder pulses per revolution of 4096: measuring steps per revolution = 4096
24-bit data width for the total resolution yields 0x01000000 (32-bit HEX number). This number, represented separately in HighWord and LowWord, equals 0x0100 in the HighWord and 0x0000

in the LowWord. The decimal values of these two parts (0x0100 = 256 decimal) are to be entered as follows:
total resolution (high) = 256
total resolution (low) = 0

This results in the following consistent configuration for the technology object:
the encoder value is transferred left-justified to Gn_XIST1; the unused bits of the fine resolution are set to 0 according to PROFIdrive, but must be specified in the fine resolution of the actual value. This results in a fine resolution of 8 bits (32 bits - 24 bits = 8 bits) (2^8 = 256 as a factor).

According to the setting above, the absolute value in Gn_XIST2 has a right-justified alignment and therefore a fine resolution of the absolute value in Gn_XIST2 of 0 bits (2^0 = 1 as a factor).

Encoder via PROFIdrive axis telegram on ADI4 and IM174

At least one electric or hydraulic axis must be configured on ADI4/IM174.

The defined update rate (BaudRate) for SSI encoders must be supported by the encoder.

See also the Encoder list (Page 3190).

For more information about configuration and operation, refer to the *ADI4 - Analog Drive Interface for 4 Axes Manual* and *Distributed I/Os IM 174 PROFIBUS Module Manual*. These documents are provided on the *SIMOTION Documentation* DVD.

Note

An encoder in a PROFIdrive axis telegram can only be used for an external encoder TO or for the encoder of a TO axis in cyclical operation if a TO axis is also created on the PROFIdrive telegram and is not deactivated.

See also

Encoder assignment and terminology (Page 3190)

Encoder list (Page 3193)

4.8.10.9 Encoder interface as a direct value in the I/O area

Encoders can be used that

- Provide actual value information directly as absolute value in the input/output area.
- Provide a counter value in the peripheral area (as of V4.0).
- Provide an actual velocity in the peripheral area.

Actual value information directly as absolute value

These encoders must be parameterized and operated as absolute encoders, for example with respect to homing.

The following settings are provided to set the **adaptation to the properties of the measured value**:

- The **justification of the measured value** in the **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.format** configuration data element
  - Signed left-justified (VALUE_LEFT_MARGIN)
  - Signed right-justified (VALUE_RIGHT_MARGIN)
  - Unsigned left-justified (VALUE_LEFT_MARGIN_WITHOUT_SIGN)
  - Unsigned right-justified (VALUE_RIGHT_MARGIN_WITHOUT_SIGN)
- The **data width of the measured value** without the sign bit in the **NumberOfEncoders.Encoder_n.analogSensor.DriverInfo.resolution** configuration data
- The upper limit and lower limit, the **maximum limits of the measured value** in the following configuration data
  - **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.maxValue**
  - **NumberOfEncoders.Encoder_n.AnalogSensor.DriverInfo.maxValue**

Example: Use of an ET 200S, SSI module, or an analog input.

Justification of the measured value

The measured value is mapped to an internal 32-bit wide signed data value of the DINT type in accordance with the justification specification. The mapped value is then tested with actual value limitation against the maximum limits and evaluated using the weighting factor for the **NumberOfEncoders.Encoder_n.AnalogSensor.ConversionData.factor** direct value that specifies the technological resolution or assignment of the LSB.

⚠ WARNING

Weighting factor for analog sensors

The scaling or conversion in the value of the configuration data **ConversionData.factor** for the position of the analog sensor has changed in SIMOTION V4.4.

As of V4.4, the following applies:

The values are entered directly in inches and no longer converted from mm to inches for the configured Inch unit system. This can result in a higher velocity being detected (by a factor of 25) if it is an axis that works in inches.

The following applies for SIMOTION up to V4.4:

The values were entered in mm and then converted to inches (1 inch corresponds to approx. 25 mm). Please check the values entered for **ConversionData.factor** in older projects and correct them when necessary.

Regardless of the sign, the encoder resolution/measured value width is limited to maximum 31-bit, because the configuration data of the maximum limits are signed data values of the DINT type.

For the **Left-justified/Right-justified without sign** setting, the measured value for the setting of the encoder resolution/measured value width is processed as follows:

- At ≤16 bits the measured value left-justified/right-justified is mapped to the least-significant byte 1 and byte 2 of an internal data value of the DINT type. In this process, the missing 16 bits minus the measured value width of the least-significant byte 1 and byte 2 are right/left-padded with zeros and the most-significant byte 3 and byte 4 are filled with zeros.
- > At 16 bits the measured value left-justified/right-justified is mapped to an internal data value of the DINT type. In this process, the missing 32 bits minus the measured value width are right/left-padded with zeros.
  As this mapped measured value is tested against the maximum limits of the data type DINT, the encoder resolution/measured value width is limited to maximum 31 bits. This means the measuring range is limited to maximum 50% of the measured value width.

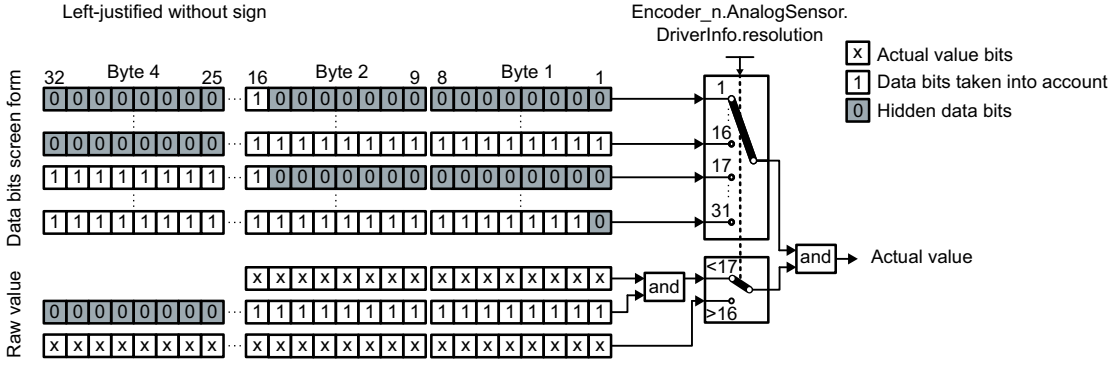


Figure 4-931 Masked reading of the analog IO value left-justified without sign

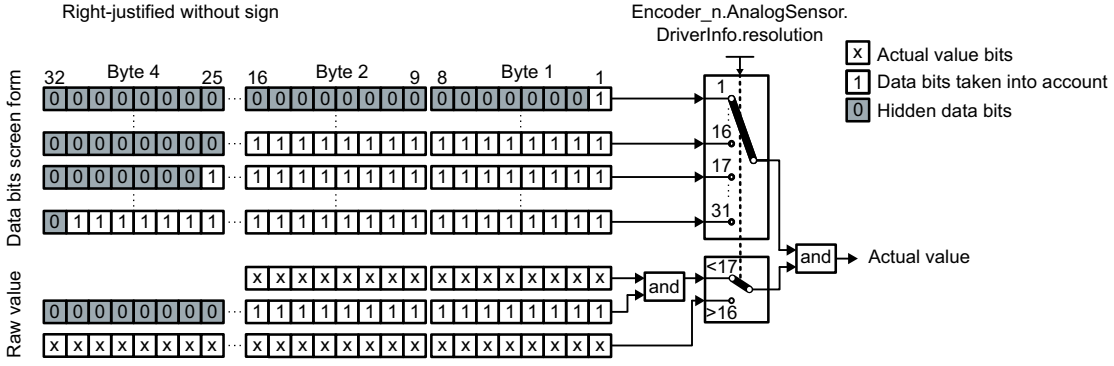


Figure 4-932 Masked reading of the analog IO value right-justified without sign

For the **Left-justified/Right-justified without sign** setting, the measured value for the setting of the encoder resolution/measured value width is processed as follows:

- At <16 bits the measured value left-justified/right-justified is mapped to the least-significant byte 1 and byte 2 of an internal data value of the DINT type. For left-justified, the missing 15 bits minus measured value width of the least-significant byte 1 and byte 2 are right-padded with zeros and the sign bit is extended left to bit 32. For right-justified, the sign bit is padded left to bit 32.
- At ≥16 bits the measured value left-justified/right-justified is mapped to an internal data value of the DINT type. For left-justified, the missing 31 bits minus measured value width are right-padded with zeros. For right-justified, the sign bit is padded left to bit 32.

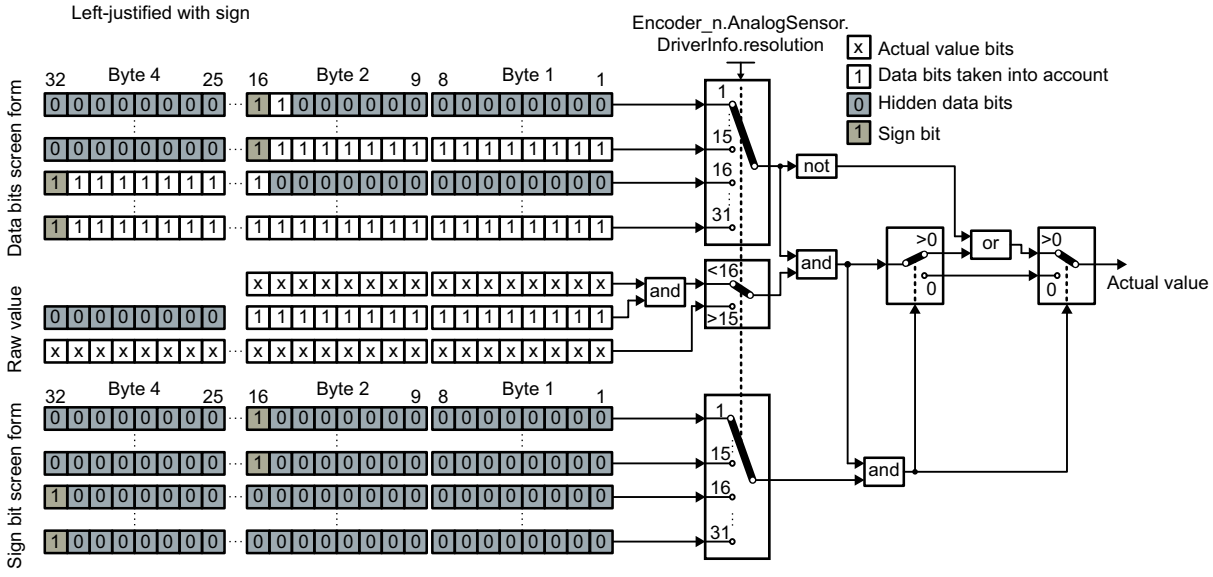


Figure 4-933 Masked reading of the analog IO value left-justified with sign

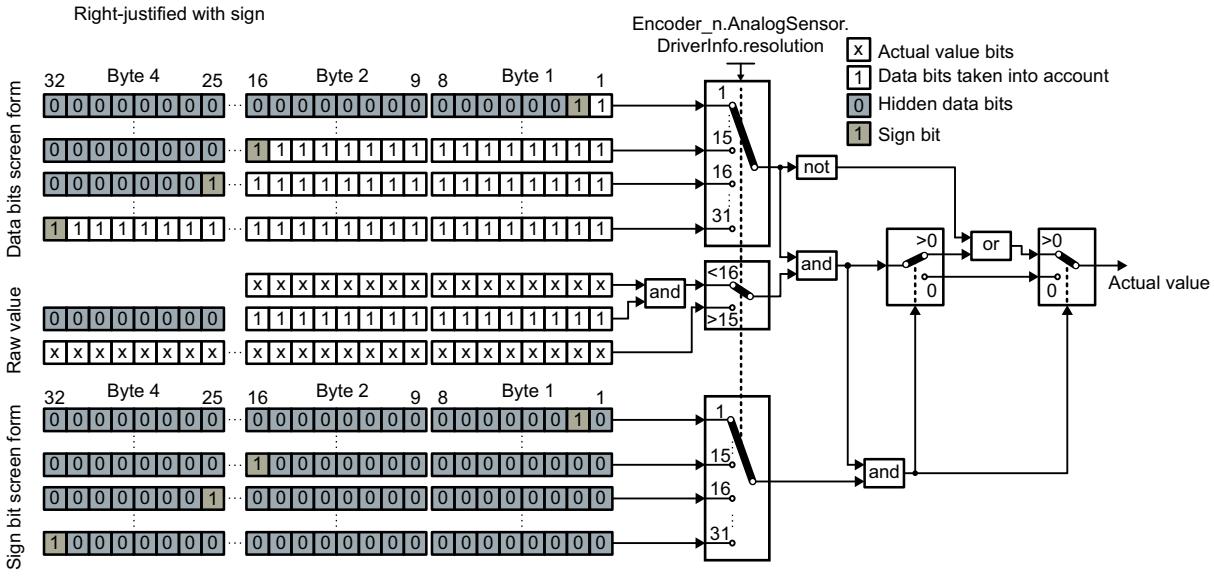


Figure 4-934 Masked reading of the analog IO value right-justified with sign

Counter value (as of V4.0)

The encoder is set as an incremental encoder. 16 bits or 32 bits can be set as counter value width.

Example: Use of an ET 200S, COUNT module

Actual velocity

The actual value information can include the number of pulses between two scans or, alternatively, the time interval between two sequential pulses. This type of encoder is used, for example, to measure actual velocities with the hydraulic axis functionality.

Example: Use of an ET 200S, COUNT module

Direct value as absolute value in I/O area (as of V4.1 SP1)

The following bits can be configured for the direct value as absolute value in the I/O area:

- **Ready bit** via the elements of the **analogSensor.readyStateMonitoring** configuration data element
- **Error bit** via the elements of the **analogSensor.errorStateMonitoring** configuration data element

These can be used for evaluation on the Axis technology object of a *ready identifier* and an *error identifier*, which are available from the peripheral module in addition to the measured value.

In SIMOTION V4.1 SP1, this configuration data is defined directly in the expert list.

If the *not ready* status or an error is displayed in these identifiers during operation, and the **ready bit** or **error bit** is configured, technology alarm 20005 is issued with the *sensor error* identifier.

If the Axis technology object is ready during ramp-up, but the direct value in the I/O area is not yet in ready status, the *WAIT_FOR_VALID Sensor* status is displayed on the encoder. (**sensorData[n].state** system variable)

As of V4.1 SP1, the direct value in the I/O area does not have to be updated in every equidistant communication cycle (for example, if an encoder connected to the peripheral module is not able to supply a new measured value in each cycle during a fast communication cycle clock due to measurement reasons or processing time reasons). In such cases, the actual value is extrapolated by the control.

The control supports the following options:

- The peripheral module displays the new measured value set in an update bit/update counter. The update bit or update counter is defined in the **analogSensor.UpdateCounter** configuration data element.
  UpdateCounter configuration: The update counter can be one (toggle) bit or several (counter) bits in width.
- The refresh cycle of the actual value in the peripheral module is known and is defined directly in the **analogSensor.UpdateCounter.updateCycle** configuration data element.
  Default setting with refresh cycle = 1 (default behavior for updating in every cycle clock)

4.8.10.10 Actual value system

Actual value processing for the external encoder is described using the actual value processing on the axis.

4.8.10.11 Overflow bit for modulo counting

The number of modulo revolutions (V3.2 and higher) is described in Chapter Actual value measurement / Actual value system.

See also

Actual value measurement / actual value system (Page 2969)

4.8.10.12 Actual value smoothing

The actual values are read in the position control cycle clock.

Other quantities, such as the velocity, are calculated from these data. The system variables for **SensorData** are calculated in the position control cycle clock, whereas the system variables for **MotionState** are calculated in the IPO cycle clock.

The filter for the velocity in the position control cycle clock is set in **typeOfAxis.Encoder_1.filter**.

The filter for the velocity in the interpolator cycle clock is set in **typeOfAxis.SmoothFilter**.

See also

Actual value measurement / actual value system (Page 2969)

4.8.10.13 Actual value extrapolation

In the case of synchronous operation with master value reference to external encoder position values, a setting can be made during configuration to specify whether the actual values are to be generated using extrapolation (see "Actual value coupling" in the Synchronous Operation description of functions).

The velocity is filtered / averaged separately using a filter that is set via **extrapolation.Filter**.

The extrapolated actual values are displayed in **extrapolationData.position** and **extrapolationData.velocity**.

See also

Actual value measurement / actual value system (Page 2969)

4.8.10.14 Standstill signal

The standstill signal **motionState.stillstandVelocity** is ACTIVE when the current velocity is less than a configured velocity threshold for at least the duration of the delay time.

The **StandStillSignal** configuration data element can be used to specify when the standstill signal is to be output.

4.8.10.15 Monitoring functions

- **Zero mark monitoring** for incremental encoders
  You can activate the function for monitoring the number of increments between two encoder zero marks.
- Permissible **changes to the actual value** of an absolute encoder
  The user can activate the function for monitoring permissible changes to the actual value for an absolute encoder.
- **Limiting frequency**
  The encoder limit frequency can be monitored.
- **Current velocity**
  The permissible maximum value of the actual velocity can be monitored. If the maximum value is exceeded, the system variable **sensordata.sensorMonitoring.velocity** is output as **limitexceeded**. The velocity is not limited to this value.
- **Cyclical data exchange** (as of V3.1)
  The system variable **sensorMonitoring.cyclicInterface** indicates whether cyclic data exchange is active on the active encoder. The cyclical data exchange is determined using the sign-of-life.
  Application: The encoder may be located on a different drive than the actuator of the axis, or it may even have its own protocol (PROFIBUS/PROFINET encoder).
- **Number of the active encoder** (as of V3.1)
  The number of the active encoder is output directly via system variable **sensorMonitoring.actualSensor**.

4.8.10.16 Synchronization / Homing

Overview of synchronization / homing

SIMOTION supports different synchronization/homing modes for the external encoders.

You can set the reference position of the external encoder using the **_synchronizeExternalEncoder()** function.

Synchronization/homing with incremental encoders

- **Direct homing / Set home position**
  Direct homing is set with the function parameter synchronizingMode=DIRECT_HOMING.
  The current actual position of the encoder is set to the value of the home position coordinate (**syncPosition** system variable) without a traversing motion having taken place.
- **Relative direct homing**
  Relative direct homing is set with the function parameter synchronizingMode=DIRECT_HOMING_RELATIVE.
  The current actual position of the encoder is offset by the value of the home position coordinate
  (**syncPosition** system variable), without a traversing motion having occurred.
- **Passive homing/flying homing**
  Passive homing is set with the function parameter synchronizingMode=PASSIVE_HOMING.
  At the synchronization point, the external encoder is set to the value of the home position coordinate.
  IncHomingEncoder.passiveHomingMode can be set as synchronization event in the configuration data:
- Encoder zero mark (ZM_PASSIVE)
- External zero mark (CAM_PASSIVE)
- Next encoder zero mark after homing output cam (CAM_AND_ZM_PASSIVE)
  The path traveled after the homing output cam is exited up until the encoder zero mark is reached can be monitored.
- Encoder zero mark or external zero mark system setting dependent on the encoder type (DEFAULT_PASSIVE).

See also

Passive homing/on-the-fly homing (Page 2932)

Direct homing/setting the home position (Page 2933)

Relative direct homing/relative setting of home position (V3.2 and higher) (Page 2933)

Synchronization/homing with absolute encoders

Direct homing/Set home position

Direct homing is set with the function parameter synchronizingMode=DIRECT_HOMING.

The current actual position of the encoder is set to the value of the home position coordinate (**syncPosition** system variable) without a traversing motion having taken place.

Relative direct homing

Relative direct homing is set with the function parameter synchronizingMode=DIRECT_HOMING_RELATIVE.
The current actual position of the encoder is offset by the value of the home position coordinate.

(**syncPosition** system variable), without a traversing motion having occurred.

Absolute value conversion

The value of the external encoder is set equal to the encoder value + absolute encoder offset using the **_synchronizeExternalEncoder()** command and the setting of the function parameter synchronizingMode=ENABLE_OFFSET_OF_ABSOLUTE_ENCODER. The absolute encoder offset is set in the **absHomingEncoder.absshift** configuration data.

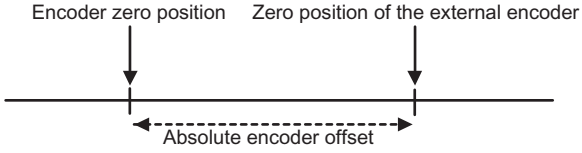


Figure 4-935 The external encoder value is the encoder zero position plus the absolute encoder offset.

The absolute encoder offset (as of V3.2) may be set as an additive or absolute value.

This absolute encoder offset is stored in the NVRAM and remains in effect until the next absolute encoder adjustment. This function must therefore be executed once when the control is commissioned.

The offset value and its calculation are set during **configuration**.

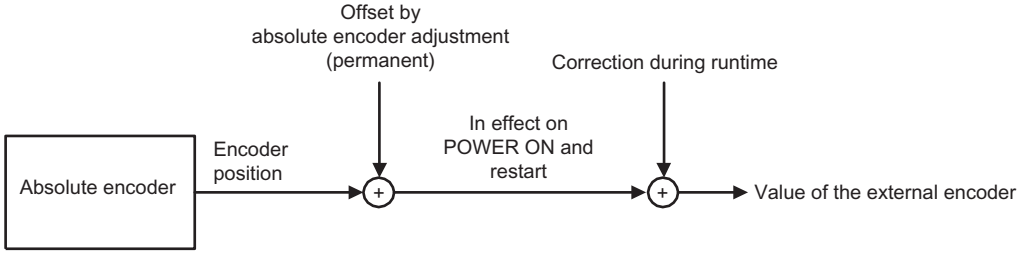


Figure 4-936 Incorporating the absolute encoder offset

A value may be set as a total offset using the **absHomingEncoder.setOffsetOfAbsoluteEncoder** and **absshift** configuration data.

Setting an additive offset

Setting **absHomingEncoder.setOffsetOfAbsoluteEncoder**=RELATIVE (default behavior):

- Actual external encoder value = actual encoder value + (previous offset already in effect + absshift)
- (new) offset = previous offset + absshift

absHomingEncoder.absshift is added to the existing absolute encoder offset whenever the **_synchronizeExternalEncoder()** function is called.

Setting an absolute offset (as of V3.2)

Setting **absHomingEncoder.setOffsetOfAbsoluteEncoder**=ABSOLUTE (as of V3.2):

absHomingEncoder.absshift is set as the absolute encoder offset whenever the **_synchronizeExternalEncoder()** function is called.

- Actual external encoder value = actual encoder value + **absshift**

Setting the value of the external encoder to a predefined position (as of V4.1 SP1)

When the synchronizingMode=SET_OFFSET_OF_ABSOLUTE_ENCODER_BY_POSITION function parameter is set on the **_synchronizeExternalEncoder()** command, the value in the syncPosition parameter will be set as the current position. The resulting absolute encoder offset is calculated with this value, indicated in the **absoluteEncoder[n].totalOffsetValue** system variable, and stored as a retain variable in the system.

The value in the **absHomingEncoder.absshift** configuration data element is not changed.

Displaying the offset

The offset can be read out. (as of V3.1)

The complete offset is indicated in the **absoluteEncoder.totalOffsetValue** system variable and the activation status of the complete offset indicated in the **absoluteEncoder.activationState** variable.

In addition, the status indicates whether the **_synchronizeExternalEncoder()** function with synchronizingMode= ENABLE_OFFSET_OF_ABSOLUTE_ENCODER has been executed at least once after the project was downloaded.

States that require a new setting of the external encoder value

- Once a new project has been downloaded to the control, the stored offset is no longer available.
  If the control already contains a project before the new project is downloaded, and if the technology object name is not changed, the stored offset is retained (as of V4.1 SP1). This behavior also applies for an upgrade, i.e. it is not version-dependent.
- After power is cycled off and on, the offset is deleted unless the project was saved to the ROM.
- After a memory reset.

See also

Absolute encoder homing / absolute encoder adjustment (Page 2934)

Direct homing/setting the home position (Page 2933)

Relative direct homing/relative setting of home position (V3.2 and higher) (Page 2933)

4.8.11 Programming External Encoders/Reference

4.8.11.1 Commands

The External Encoder technology object can be addressed in the user program using the following commands:

- **_enableExternalEncoder()**
  Activate external measuring system; enables updating of actual values in the IPO.
- **_disableExternalEncoder()**
  Deactivate external measuring system; disables updating of actual values in the IPO. Actual values are frozen in the IPO, they remain unchanged until the next activation of the measuring system.
- **_resetExternalEncoder()**
  Reset External Encoder technology object.
- **_resetExternalEncoderError()**
  Reset an error on the External Encoder technology object.
- **_synchronizeExternalEncoder()**
  Homing of the measuring system by setting the synchronous position directly or by enabling the synchronous position with the next external zero mark / encoder zero mark / homing output cam and encoder zero mark or absolute encoder adjustment.
  The command is only performed in the **active** TO state.
- **_resetExternalEncoderConfigDataBuffer()**
  This function clears the configuration data collected in the buffer since the last activation without activating them.
- **_bufferExternalEncoderCommandID()**
  Enables the saving of commandId and the associated command status beyond the execution period of the command.
- **_removeBufferedExternalEncoderCommandId()**
  Stops the saving of commandId and the associated command status beyond the execution period of the command.
- **_enableMonitoringOfEncoderDifference()**
  Activates the monitoring of the maximum permissible difference between the measuring systems specified in the command.
- **_disableMonitoringOfEncoderDifference()**
  Deactivates the encoder monitoring.
- **_getStateOfExternalEncoderCommand()** (as of V3.1)
  Read out command status
- **_getExternalEncoderErrorNumberState()** (as of V3.1)
  Readout of error number status

- **_redefineExternalEncoderPosition()** (as of V3.2)
  Sets the encoder coordinate system
- **_cancelExternalEncoderCommand()** (as of V4.1 SP1)
  Cancels the command with the specified CommandID.

Note

You can also find information about the system functions in the *SIMOTION Cam Technology Package parameters manual*.

4.8.11.2 Technological alarms

Possible alarm reactions

- **NONE**
  No response.
- **DECODE_STOP**
  Command preparation aborted. The current function remains active.
  Processing on the technology object can be continued after **_resetExternalEncoder()** or **_resetExternalEncoderError()**.
- **SIMULATION_STOP**
  Calculation of the simulation value by a function is stopped. The simulation function is not aborted. Simulation can be continued with **_resetExternalEncoderError()**.
- **SIMULATION_ABORT**
  Calculation of the simulation value by a function is stopped. The simulation function is aborted.
- **ENCODER_DISABLE**
  Stops and aborts all commands.

Note

Local alarm responses are specified by means of the system.

4.9 Industrial Security

Preface

You can find information on the following topics at the following address (https://support.industry.siemens.com/cs/de/en/view/108464614):

- Ordering documentation/overview of documentation
- Additional links to download documents
- Using documentation online (find and search in manuals/information)

If you have any questions regarding the technical documentation (e.g. suggestions, corrections), please send an e-mail to the following address (mailto:docu.motioncontrol@siemens.com).

mySupport/Documentation

At the following address (https://support.industry.siemens.com/My/ww/en/documentation), you can find information on how to create your own individual documentation based on Siemens' content, and adapt it for your own machine documentation.

Training

At the following address (https://www.siemens.com/sitrain), you can find information about SITRAIN (Siemens training on products, systems and solutions for automation and drives).

FAQs

You can find Frequently Asked Questions in the Service&Support pages under Product Support (https://support.industry.siemens.com/cs/de/en/ps/faq).

SINUMERIK

You can find information about SINUMERIK at the following address (https://www.siemens.com/sinumerik).

SIMOTION

You can find information about SIMOTION at the following address (https://www.siemens.com/simotion).

SINAMICS

You can find information about SINAMICS at the following address (https://www.siemens.com/sinamics).

Target group

This documentation is intended for manufacturers of machine tools / production machines, particularly:

- Planners and project engineers
- IT departments of end users and OEMs

The following knowledge is a prerequisite for implementing the described security concepts:

- Administration of the IT technologies familiar from the office environment
- Configuration of the SINUMERIK/SIMOTION/SINAMICS products used

Benefits

The "Industrial Security" documentation contains the necessary measures and information for planning and configuring systems or plants. The documentation serves as a reference manual and guideline. This documentation cannot and does not want to suggest that there is 100% security because the current threat scenario is much too diverse and complex.

This documentation includes all of the necessary measures that should be taken into account for configuring systems in a secure environment. This documentation is intended to support machine manufacturers in safely operating their controls or plants. You, as operator, are responsible for implementing the security measures.

Note regarding the General Data Protection Regulation

Siemens observes standard data protection principles, in particular the principle of privacy by design. That means that

this product does not process / store any personal data, only technical functional data (e.g. time stamps). If a user links this data with other data (e.g. a shift schedule) or stores personal data on the same storage medium (e.g. hard drive) and thus establishes a link to a person or persons, then the user is responsible for ensuring compliance with the relevant data protection regulations.

Technical Support

Country-specific telephone numbers for technical support are provided on the Internet at the following address (https://support.industry.siemens.com/sc/ww/en/sc/2090) in the "Contact" area.

If you have any technical questions, use the online form in the "Support Request" area.

4.9.1 Fundamental safety instructions

4.9.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

4.9.1.2 Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. Application examples do not represent specific customer solutions, but are only intended to provide support for typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

4.9.1.3 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit
https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under
https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions).

Further information is provided on the Internet:

Industrial Security Configuration Manual (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ WARNING

Unsafe operating states resulting from software manipulation

Software manipulations, e.g. viruses, Trojans, or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
- Make sure that you include all installed products into the holistic industrial security concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- On completion of commissioning, check all security-related settings.
- Protect the drive against unauthorized changes by activating the "Know-how protection" converter function.

4.9.2 What is industrial security?

Definition of industrial security

Generally, industrial security is understood to be all of the measures for protecting against the following:

- Loss of confidentiality due to unauthorized access to data
- Loss of integrity due to data manipulation
- Loss of availability (e.g. due to destruction of data or Denial-of-Service (DoS))

Objectives of industrial security

The objectives of industrial security encompass:

- Fault-free operation and guaranteeing of availability of industrial plants and production processes
- Preventing hazards to people and production due to cyber security attacks
- Protection of industrial communication from espionage and manipulation
- Protection of industrial automation systems and components from unauthorized access and loss of data
- Practicable and cost-effective concept for securing existing systems and devices that do not have their own security functions
- Utilization of existing, open, and proven industrial security standards
- Fulfillment of legal requirements

An optimized and adapted security concept applies for automation and drive technology. The security measures must not hamper or endanger production.

4.9.3 Why is industrial security so important?

4.9.3.1 Networking and wireless technology

Overview

There are many new trends which affect industrial security:

- Cloud computing in general
  The number of network connections across the world is constantly increasing. This enables innovations such as cloud computing and the applications that go hand in hand with it. In conjunction with cloud computing, there has been a massive increase in the number of mobile devices, such as cell phones and tablet PCs.
- Wireless technology
  On the other hand, the increasing use of mobile devices has only become possible thanks to the ubiquitous availability of mobile networks. Wireless LAN is also becoming increasingly available.
- Worldwide remote access to plants, machines and mobile applications
- The Internet of Things (IoT)
  Millions of electronic devices are becoming network-capable and are communicating via the Internet, such as onboard computers in cars, which transmit warranty information to dealers, or water meter sensors that transmit water consumption data to municipal water suppliers via radio.

However, in order for everything from cloud computing to the Internet of Things to function disturbance-free, you require a reliable network infrastructure and applications that are well protected against attacks from malware and hackers.

4.9.3.2 Possible corporate security holes

Possible security holes or weak points

The security chain of a company is only as strong as its weakest link. Security holes can exist at numerous points. The following list gives only a few examples:

- Employees / external companies
- Production plants
- Network infrastructure
- Data centers / PC workstations
- Laptops/tablets
- Printers
- Smartphones/smartwatches
- Mobile data storage media

For this reason, a holistic approach is required to deal with the issue of security. Coordinated guidelines and regulations are required that cover all areas: Devices, systems, processes and employees.

The topic of data security and access protection (security) is becoming more and more important in industrial environments. The following technologies results in higher requirements placed on protecting and securing industrial plants and systems:

- The ongoing networking of complete industrial plants and systems
- The vertical integration and networking of various company levels
- New techniques, e.g. remote maintenance and/or remote access

The threats are diverse and the consequences are far-reaching:

Possible threats:

Potential threats come from the industry environment and involve the topic of confidentiality, integrity and availability. Examples of threats include the following:

- Espionage of data, recipes, etc.
- Sabotage of production plants
- System stoppage, e.g. due to virus infection and malware
- Manipulation of data or application software
- Unauthorized use of system functions

Possible effects of a security incident

- Loss of intellectual property
- Loss of production or reduced product quality
- Negative company image and economic damage

- Catastrophic environmental influences
- Danger to people and machines

4.9.4 Security measures in automation and drive technology

Siemens automation and drive technology concerns itself with security aspects at the following levels:

- **Application security** refers to products and functions that take into consideration the needs of industrial security in the field of automation. This involves particular consideration of the application and task at hand, as well as the people performing the actions in an automated plant. This allows industrial security to be easily implemented in production processes.
- **Security support** provides support during the analysis, planning, implementation, testing and optimization of industrial security - by means of specialists with special knowledge of networks and the industry. These services lead to the highest possible level of industrial security and operating capacity of the production plant.
  Siemens offers comprehensive customer support based on the "Implement Security" service: With this service you can implement protective measures to increase the security level of plants and production facilities. You can find more information about the entire "Implement Security" portfolio on the Internet.

See also

Implementing security (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/implementation.html)

4.9.4.1 Security measures

With increasing digitalization, comprehensive security in the automation system is becoming ever more important. For this reason, industrial security is a core element of every product that can be networked.

Integration of security into the products

As manufacturer of automation and drive products, Siemens supports secure operation for its customers by **integrating security into its products**:

- All of the measures involving automation and drive technology are stored in the **Product Lifecycle Management (PLM) process**, which is certified by the German Technical Inspectorate (TÜV) based on IEC 62443-4-1.
- Analytically potential attack threats are detected and evaluated using **Threat and Risk Analyses (TRA)**. Identified critical threats are implemented in the product as necessary basic functions, based on the motto "Security by Design".
- Siemens regularly performs **code analyses** in order to identify and correct possible errors at an early stage during the formal check.

- In its products and its manufacturing process, Siemens has implemented **measures to secure integrity** to indicate any changes to the integrity.
- Siemens constantly checks the measures relating to **hardening**:
  - Operating systems are configured in such a way that **points of attack** (e.g. via ports, unneeded services) are **minimized**.
  - Siemens **tests its products** to detect weak points at an early stage.
  - Siemens offers a specific **hotfix/patch management**.

Protection of the development infrastructure and supply chain

As manufacturer of automation and drive products, Siemens supports secure operation for its customers by **securing the development infrastructure and supply chain**:

- The Siemens ProductCERT (https://www.siemens.com/cert/en/cert-security-advisories.htm) (**C**yber **E**mergency **R**eadiness **T**eam) is the central department for security-related incidents in the Siemens product and solution environment. Siemens ProductCERT supports development work with consulting and other services. **ProductCERT** provides information about current threats and vulnerabilities as well as the appropriate countermeasures.
- Industrial security is a dynamic and complex subject that requires continuous monitoring and adaptation of new security measures. Information on how Siemens protects its products and solutions against cyber attacks and how industry profits from the competence of Siemens can be found on the Internet (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/always-active.html).

Provision of patches, security components, and appropriate services

As manufacturer of automation and drive products, Siemens supports secure operation for its customers through direct support of integrators and operating companies **by providing patches, security components and the appropriate services**:

- SIEMENS offers monitoring through a **SIEM system** to monitor residual risk. SIEM stands for Security Information and Event Management and has become an established term in IT security. Such systems are able to identify and evaluate security-relevant events and notify the administrator.

4.9.4.2 Siemens Industrial Holistic Security Concept

Siemens places great emphasis on protecting the integrity and guaranteeing the confidentiality of the processed data for its own products. Intellectual property and know-how of the Siemens products are also in focus.

To achieve this, the Siemens Industrial Holistic Security Concept (SI HSC) is applied which protects development departments and production plants (see the following diagram). Multi-level security systems and basic security improvements of the IT infrastructure are implemented. In parallel, process improvements have been introduced and training in security awareness provided in the development and production. These measures are being performed continuously by Siemens and clearly demonstrated by the security levels reached.

SI HSC also benefits the customers who Siemens has selected as partners for their industrial solutions, or who want to orientate themselves on the concept. Siemens suppliers are also

considered with regard to security so that Siemens already applies the same security standards when purchasing as for the manufacture of its own products.

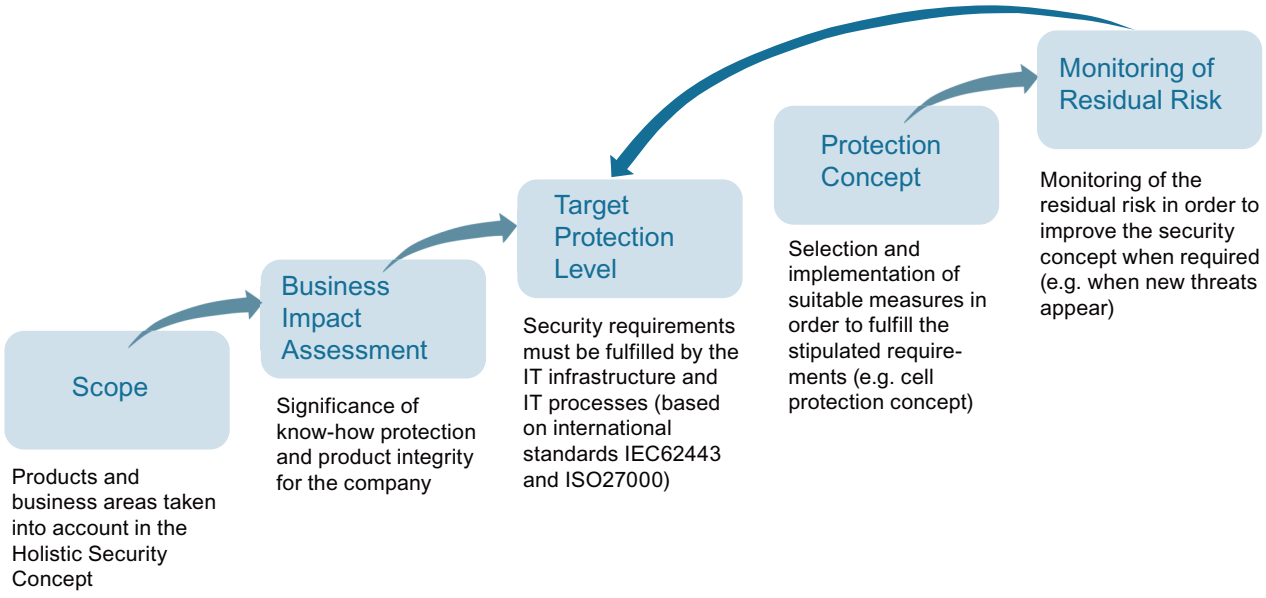


Figure 4-937 SI HSC security management process

Further information

Further information on the Siemens Industrial Holistic Security Concept is available on the Internet on page Always active (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/always-active.html).

4.9.4.3 Standards and regulations

Siemens takes the applicable Industrial Security standards and regulations into consideration throughout the entire development process:

- ISO 2700X: Management of information security risks
- IEC 62443: IT security for industrial higher-level control systems – network and system protection

Further information on certifications and standards in the Industrial Security field can be found on the Internet (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/certification-standards.html).

4.9.5 Security management

The security management process as a basis

Protect your system and your company. Security management according to IEC 62443 and ISO 27001 forms the basis for the successful implementation of industrial security.

The security management process is shown in the following:

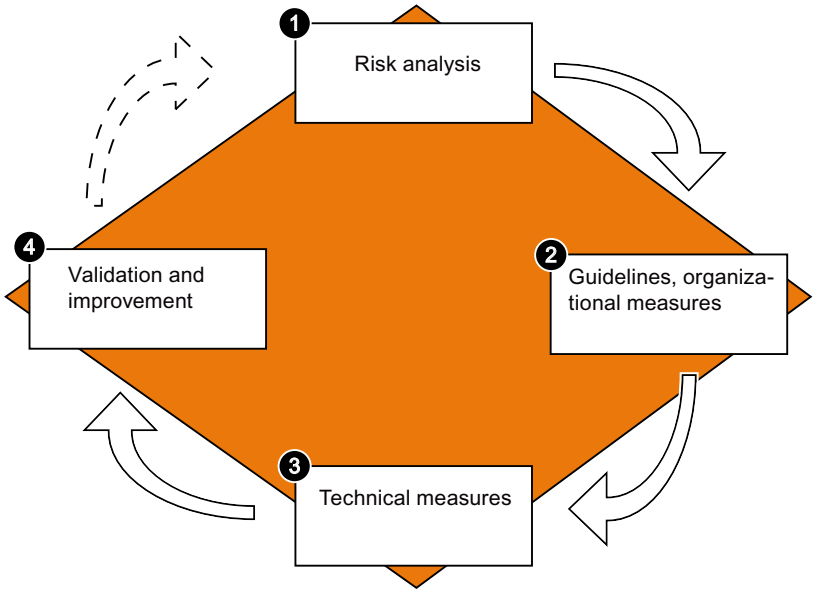


Figure 4-938 Security management process

1. Carry out a risk analysis. Determine all possible risks and define countermeasures for reducing the risk to an acceptable level. In detail, a risk analysis encompasses the following steps:
   - Identification of threatened objects
   - Analysis of value and potential for damage
   - Threat and weak point analysis
   - Identification of existing security measures
   - Risk evaluation
   - Evaluation of the effects with regard to the protection goals of confidentiality, integrity, and availability
2. Define guidelines and introduce coordinated, organizational measures. To this end, the awareness of the importance of industrial security must be borne by all levels of the company. In addition, define guidelines and processes in order to achieve a uniform procedure and to support compliance with the defined industrial security concept.

3. Introduce coordinated technical measures. You can find a list of general measures that help to protect your plant against threats in Section General security measures (Page 3220). You can find measures to be considered for SINUMERIK, SIMOTION and SINAMICS environments in Chapter Product-specific security measures (Page 3234).
4. A security audit must ensure that all of the measures have been implemented and that they have also eliminated or reduced the identified risks.

Note

Continuous process

Due to constantly changing security threats, **this process must be continuously repeated** in order to guarantee the security of your plant. For this reason, the security management process must be seen as a continuous process.

4.9.6 General security measures

In this chapter you will learn about the general security measures you must take in order to protect your system from threats.

Additional specific security measures for SINUMERIK, SIMOTION and SINAMICS products can be found in Section Product-specific security measures (Page 3234).

4.9.6.1 Defense in depth concept

To protect industrial plants and systems comprehensively against cyber attacks, measures must be applied simultaneously at all levels. From the operational up to the field level – from access control to copy protection. For this purpose, we use "Defense in Depth" as a general protection concept, according to the recommendations of ISA99 / IEC 62443, the leading standard for security in industrial automation.

Figure 4-939 Defense in depth strategy

Further information on the defense in depth concept and the planning of a protection concept for industrial plants can be found on the Internet (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/planning.html).

Protection levels

A defense in depth model has a three level structure:

- **Plant security**
  Plant security represents the outermost protective ring. Plant security includes comprehensive physical security measures, e.g. entry checks, which should be closely coordinated with protective measures for IT security.
- **Network security**
  The measures, grouped under the keyword "Network security", form the core of the protective measures. This refers to the segmentation of the plant network with limited and secure communication between subnetworks ("secure islands") and the interface check with the use of firewalls.
- **System integrity**
  "System integrity" represents the combination of two essential protection aspects. PC-based systems and the control level must be protected against attacks. Steps include the following measures:
  - Integrated access protection mechanisms in the automation components to prevent unauthorized changes via the engineering system or during maintenance
  - The use of antivirus and whitelisting software to protect PC systems against malware
  - Maintenance and update processes to keep the automation systems up-to-date (e.g. patch management, firmware updates, etc.)

4.9.6.2 Plant safety

Unauthorized persons may be able to enter the production site/building and damage or alter production equipment as a result of gaps in a company's physical security. Confidential information can also be lost. This can be prevented if both the company's site and the production areas are protected accordingly.

Physical protection of critical production areas

Company security

The company's physical security can be ensured via the following measures:

- Closed off and monitored company premises
- Entry control, keys / card readers and/or security personnel

- Escorting of external personnel by company employees
- Security processes in the company are taught and followed by all employees

Physical production security

The physical security of a production location can also be ensured via the following measures:

- Separate access control for critical areas, such as production areas
- Installation of critical components in lockable control cabinets / switching rooms including monitoring and alarm signaling options. The control cabinets/contact chambers must be secured by a cylinder lock. Do not use simple locks, such as universal, triangular/square or double-bit locks.
- Configuration of the radio field to restrict the WLAN range so that it is not available outside the defined areas (e.g. factory building).
- Guidelines that prevent the use of third-party data storage media (e.g. USB sticks) and IT devices (e.g. notebooks) classified as insecure on systems.

Further information

Further information on integrated Siemens security solutions can be found on the Siveillance page (https://new.siemens.com/global/en/products/buildings/security/security-management.html).

4.9.6.3 Network security

Network security includes all measures taken to plan, implement and monitor security in networks. This includes the control of all interfaces, e.g. between the office network and plant network, or remote maintenance access via the Internet.

Network segmentation

Separation between production and office networks

One important protective measure for your automation or drive system is the strict separation of the production networks and the other company networks. This separation creates protection zones for your production networks.

Note

The products described in this manual must only be operated in defined protection zones.

Separation by means of a firewall system

In the simplest scenario, separation is achieved by means of an individual firewall system which controls and regulates communication between networks.

See also Network segmentation with SCALANCE S (Page 3224)

Separation via a DMZ network

In the more secure variant, the coupling is established via a separate DMZ network. In this case, direct communication between the production network and the company network is completely prevented by firewalls and only takes place indirectly via servers in the DMZ network.

Note

The production networks should also be divided into separate automation cells in order to protect critical communication mechanisms.

General security measures

Observe the general security measures even within protection zones, for example as listed in System hardening (Page 3227):

Network segmentation with SCALANCE S

Siemens provides SCALANCE S security modules to meet network protection and network segmentation requirements. Further information on SIEMENS SCALANCE S can be found on the Internet (https://siemens.com/scalance-s).

SCALANCE S security module

SCALANCE S security modules with Security Integrated provide:

- Stateful inspection firewall
  In order to implement user-specific control and logging, firewall rules can also be specified that only apply to certain users.
- VPN via IPsec (data encryption and authentication)
  This establishes a secure tunnel between authenticated users whose data cannot be intercepted or manipulated. The most important aspect is the protection against external access via the Internet.
- NAT/NATP (address translation)
- Router functionality (PPPoE, DDNS) for broadband Internet access (DSL, cable)
- SCALANCE S623 with additional VPN port (DMZ) enables the secure connection of an additional network for service and remote maintenance purposes. S623 also permits the secure, redundant connection of subordinate networks by means of routers and firewall redundancy.
- SCALANCE S615 has five Ethernet ports with which different network topologies can be protected by means of a firewall or Virtual Private Network VPN (IPsec and OpenVPN), and security concepts implemented flexibly.

Requirement

NOTICE
Data misuse Long distances between the device to be protected and the upstream security modules represent an invitation for data misuse. • Note that upstream security modules, such as SCALANCE S, must be installed close to the device to be protected in a locked control cabinet. This ensures that data cannot be manipulated here without notice.

Principle

The following application example shows cell segmentation by several SCALANCE S modules, each of which is upstream of the automation cells. The data traffic to and from the devices within automation cells can be filtered and controlled with the SCALANCE S firewall. If required, the traffic between the cells can be encrypted and authenticated. Secure channels and client access from the PCs to the cells can be established via SOFTNET Security Client, VPN client software for PCs.

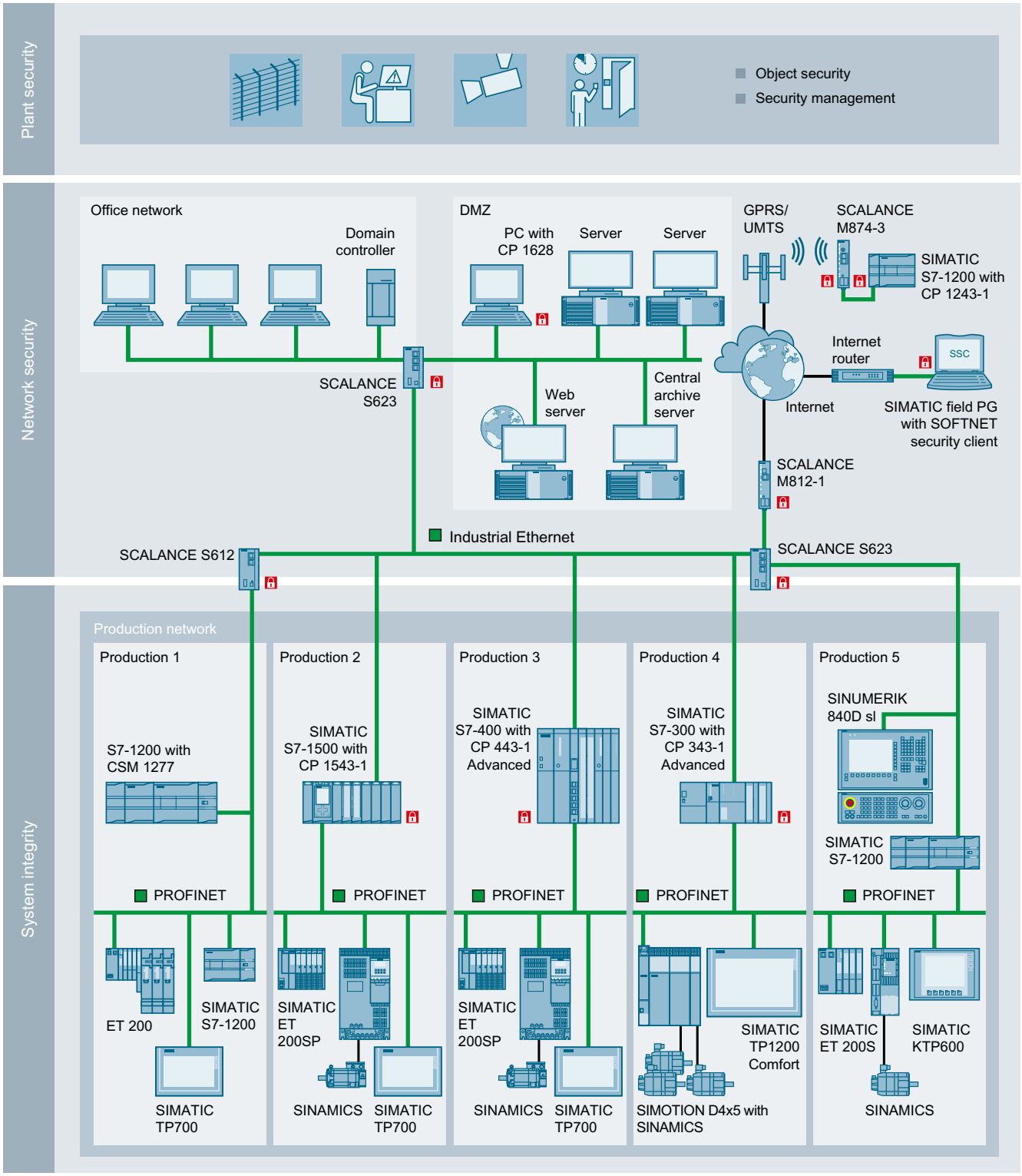


Figure 4-940 SCALANCE S application example

VPN access

Note

Note that a SCALANCE S security module must always be used for VPN access.

Cloud Security

Since cloud applications are becoming increasingly common and the cloud continues to grow in its significance, the issue of security in the cloud environment is also becoming more and more important.

The following addresses are designed to give you an idea of how you can make your system safe in the cloud environment. Inform yourself about the applicable requirements and tried and tested solutions/best practices.

Further information

- Companion Guide for Cloud - CIS Organization (https://www.cisecurity.org/press-release/cis-controls-companion-guide-for-cloud-now-available/)
- Questionnaire Cloud Security - CSA Organization (https://cloudsecurityalliance.org/artifacts/consensus-assessments-initiative-questionnaire-v3-0-1/)

4.9.6.4 System integrity

System integrity is understood to mean the "integrity" or "correctness" of the data or the correct response of the system. Thus, the following measures for protecting the system integrity should ensure that the data/functionality of the system cannot be manipulated by unauthorized persons or that manipulations can be detected.

System hardening

Services and ports

Activated services and ports represent a risk. To minimize the risk, only the necessary services for all of the automation components should be activated. Ensure that all activated services are taken into account (especially Web servers, FTP, remote maintenance, etc.) in the security concept.

A description of all of the ports used can be found in the Equipment Manuals/Function Manuals of the respective products.

User accounts

Any active user account that allows access to the system is thus a potential risk. Therefore, take the following security measures:

- Reduction of configured/activated user accounts to the actually needed minimum
- Use of secure access data for existing accounts. This also involves assigning a secure password.
- Regular checks, especially of the locally configured user accounts
- Regular change of passwords

PC/notebooks and mobile devices in an industrial environment

The terminal devices used in industrial environments (PCs, notebooks and mobile devices) must meet the generally applicable security requirements. Therefore, take the following measures:

- The terminal device that is used is set up, administered, regularly checked and patched by the appropriate departments. This ensures it is always kept up-to-date. This also means that software and operating systems which are supported and maintained by the manufacturer are always installed.
- A current virus scanner, which is adapted to the operating system used, must be installed on the terminal device that is used. Both the virus patterns and the software itself must be regularly updated. Or, alternatively, work with the Whitelisting (Page 3231) method.
- Activate a firewall with appropriate settings on the terminal device that is used.
- Use a configuration without admin. rights on the terminal device that is used.
- Encrypt all of the hard drives or mass storage units (e.g. eMMC or SSD) of the terminal device that is used to protect sensitive data against unauthorized access.
- Do not use the terminal device for other tasks, e.g. in the office network. This is part of the separation of networks dealt with in Chapter "Separation between production and office networks (Page 3224)".
- Secure terminal devices from data theft using a physical lock (e.g. Kensington lock) or do not leave them unmonitored.
- If you leave protected terminal devices at the workstation, always activate the lock mode of the operating system. This prevents access to the terminal device and the contents of the screen can no longer be read.
- Set up user accounts (Page 3228) for the access rights accordingly.
- Take appropriate measures to protect unneeded interfaces (e.g. USB, network, etc.) to prevent unauthorized access. This can be done physically using commercially available USB port locks or via corresponding software measures.

Data storage

Note

When you store security-relevant data on your PC, you are responsible for secure data storage.

These include, for example, the following measures:

- Consequent marking of your documents according to confidentiality levels by introducing a document classification.
- Protection of your encrypted storage locations, such as sharepoints, against manipulation.
- If absolutely necessary, only store your confidential or security-relevant data encrypted on your PC / systems or the network.
  Security-relevant data includes sensitive data, such as archives, passwords, or executable files (*.exe).
- Regularly back up your security-relevant data and carefully protect it against loss and manipulation.

Transporting data

Note

Apply the following measures when transporting data:

- Always encrypt your emails if you send confidential and/or security-relevant data by email.
- If you wish to transport confidential and/or security-relevant data on a data storage medium (USB flash drive, hard disk, etc.), carefully investigate as to which data storage media are considered secure. A regular virus check must be carried out for these data storage media. Always save your data on local data storage media so that the data is encrypted.

These measures are especially important for sensitive data, such as archives, passwords, or executable files (*.exe).

Passwords

NOTICE

Data misuse caused by using passwords that are not secure enough

Data can be easily misused by using passwords that are not secure enough. Insecure passwords can easily be guessed or decoded.

- Therefore, change the default passwords during the commissioning and adapt them at regularly defined intervals.
- Also change passwords for functions that you yourself do not use to ensure that such unused functions are not misused.
- Always keep your passwords secure, and ensure that only authorized persons have access to these passwords.

Note

Assigning secure passwords

Observe the following rules when creating new passwords:

- When assigning new passwords, make sure that you do not assign passwords that can be guessed, e.g. simple words, key combinations that can be easily guessed, etc.
- Passwords must always contain a combination of upper-case and lower-case letters as well as numbers and special characters. PINS must comprise an arbitrary sequence of digits.
- When assigning a password, always ensure you are adhering to the applicable company specifications, e.g. special password policy of the respective company.
- Observe that, in accordance with the applicable company specifications, passwords with the maximum required minimum length must be assigned.
- Wherever possible and where it is supported by the IT systems, a password must always have a character sequence as complex as possible.

Programs are available that can help you to manage your passwords. Using these programs, you can encrypt, save and manage your passwords and secret numbers – and also create secure passwords.

Further information on assigning secure passwords can be found in Chapter References (Page 3287).

Product security notifications

Note

Complying with product security notifications

Threats are extremely diverse in nature and are continually changing. As a consequence, always keep yourself up-to-date on a regular basis through the Industry Online Support (https://support.industry.siemens.com/sc/ww/en/sc/2090) regarding whether there are new and relevant product security notifications for your particular products. Comply with the instructions provided in the product security notifications.

Virus scanner

An anti-virus program, virus scanner or virus protection program is a software that can detect, block and, if required, eliminate computer viruses, computer worms or Trojans horses.

In principle, virus scanners can only detect known malware (viruses, worms, Trojans, etc.) or harmful logic and therefore cannot provide protection against all viruses or worms. For this reason, virus scanners can only be considered as a complement to general precautionary measures.

The use of a virus scanner must not impact the production operations of a plant. As the last consequence, this will lead to even a virus-infected computer not being permitted to immediately shut down if this would cause the control of the production process to be lost.

NOTICE
Data misuse when using online virus scanners If you use an online virus scanner, then security-relevant or confidential data can get into the wrong hands and be misused. • Therefore, do not check any security-relevant or confidential data via an online virus scanner.

Note

Keep virus scanners up-to-date

Always ensure that the virus scanner database is always up-to-date.

Note

Do not install several virus scanners together.

You must always avoid installing several virus scanners together in one system.

Note

Operation in a local network

Always use a virus scanner when locally connecting with the plant or system network.

Whitelisting

The basic philosophy of whitelisting is that all applications are mistrusted, unless they have been classified as trustworthy after an appropriate check. This means that a whitelist is maintained in the system. This whitelist therefore contains all applications that have been classified as trustworthy and consequently can be run on your PC systems.

Whitelisting mechanisms provide additional/alternative protection against undesired applications or malware and unauthorized changes to installed applications or executable files (.exe, .dll).

Heed the corresponding product-specific information (Page 3234) to determine whether the use of virus scanners and/or whitelisting is recommended.

Patch management

WSUS

The **WSUS** (Windows Server Update Service) system functionality provided by Microsoft is available for current Windows systems. WSUS supports administrators by providing Microsoft updates in large local networks. WSUS automatically downloads update packages (Microsoft update) from the Internet and offers them to the Windows clients for installation.

The fully automatic update process ensures that Microsoft security updates are always available on Siemens clients.

NOTICE
Security gaps for out-of-date operating systems Note that security updates, hotfixes, etc. are no longer supplied by Microsoft for obsolete operating systems < Windows 10. As a consequence, dangerous security gaps can occur with your operating system. • Therefore always upgrade your operating system - if possible - to the latest version. • If you work with an older operating system, take appropriate additional measures (e.g. whitelisting) to protect your system.

Note

Before installing Microsoft Updates, note the following important points:

- **Prior to the update**, back up the system status for a fallback, if necessary. Ensuring the compatibility of the update with the individual system configuration is the responsibility of the customer.
- Never establish a direct connection to the WSUS server in the Internet! Ensure that the environment is secure and install an intermediate layer (e.g. DMZ network, firewall, SCALANCE S modules, etc.).

Product software

Note

Out-of-date product software also represents a potential security gap for attacks.

- As a consequence, always install the latest product software versions.

Data integrity

Data integrity is understood to mean the correctness (integrity) of data and the correct functioning of systems. Ensuring the integrity of the data is thus an essential goal of information security. Integrity protection should not be confused with protecting the confidentiality.

NOTICE

Corruption of data and the resulting malfunctioning of the system

For automation and drive systems as well as controller components, data such as archives and programs can be imported from external sources. This data influences the behavior of these systems and should therefore be protected against unauthorized changes.

Data such as archives, programs, and OA applications can also be saved and archived. The systems currently do not provide the capability of ensuring the integrity of programs, archives, and OA applications.

Therefore take your own measures for ensuring integrity to guarantee the data integrity of your archives, OA applications, or other saved data:

- Apply the Siemens Industrial Holistic Security Concept.
- Use digital signatures to protect data.
- Ensure there is sufficient access protection:
  - Restrict access rights such as to data archives/Sharepoints accordingly.
  - Do not send any unencrypted/unsigned emails.

Disposal

The products are to be disposed of in accordance with the respectively valid national regulations. The products described in this manual are extensively recyclable on account of the low-toxic composition of the materials used. To recycle and dispose of your old equipment in an environmentally friendly way, please contact an appropriate disposal company.

NOTICE

Misuse of data resulting from insecure methods of deleting data

Incomplete or insecure deletion of data from memory cards or hard drives can lead to misuse of the data of the part programs, archives, etc. by third parties.

- Therefore ensure that all storage media are securely deleted **before disposing of the product** :
- There are programs that support you in securely deleting/formatting storage media. Alternatively, contact a certified data destruction specialist to take care of this task.

Also observe the special disposal information of the products in Chapter Product-specific security measures (Page 3234).

4.9.7 Product-specific security measures

In this chapter, you will find additional product-specific security measures for the SINUMERIK products and the CNC Shopfloor Management Software, SIMOTION, SINAMICS and SIMOCRANE.

4.9.7.1 SINUMERIK

The following chapter provides you with an overview of the security-related measures that you must take to protect your SINUMERIK control from threats. Detailed descriptions and procedures can be found in the corresponding SINUMERIK documentation.

Many products (SINUMERIK, SIMOTION, SINAMICS) contain OpenSSL. The following applies to these products:

- This product contains software (https://www.openssl.org/) that has been developed by the OpenSSL project for use in the OpenSSL toolkit.
- This product contains cryptographic software (mailto:eay@cryptsoft.com) created by Eric Young.
- This product contains software (mailto:eay@cryptsoft.com) developed by Eric Young.

Firewall and networking

NCU/PCU networking structure

The following graphic shows the networking of the control (NCU) and the IPC. X130 at the NCU and eth 1 at the IPC are used to establish a connection to the company network. A firewall protects these two interfaces against unauthorized access.

The NCU contains a packet filter functionality (firewall) that filters the connection to the factory network. This integrated firewall is preconfigured with optimum settings for the incoming and outgoing communication. The firewall is configured in such a way that access to the networks behind the firewall is blocked, and when several logon attempts are made from a certain IP address, this is identified, blocked and prevented. In this way, the control is protected against brute-force attacks.

The IPC has the firewall function via the Windows functionality.

NOTICE
Data misuse via an unprotected interface Since the X120 interface of the NCU or the eth2 port of the IPC are not protected by a firewall, there is a risk of misuse of data. The interface only provides the option of establishing a connection to the local plant/system network. • As a consequence, never connect this local network with the Internet/company network.

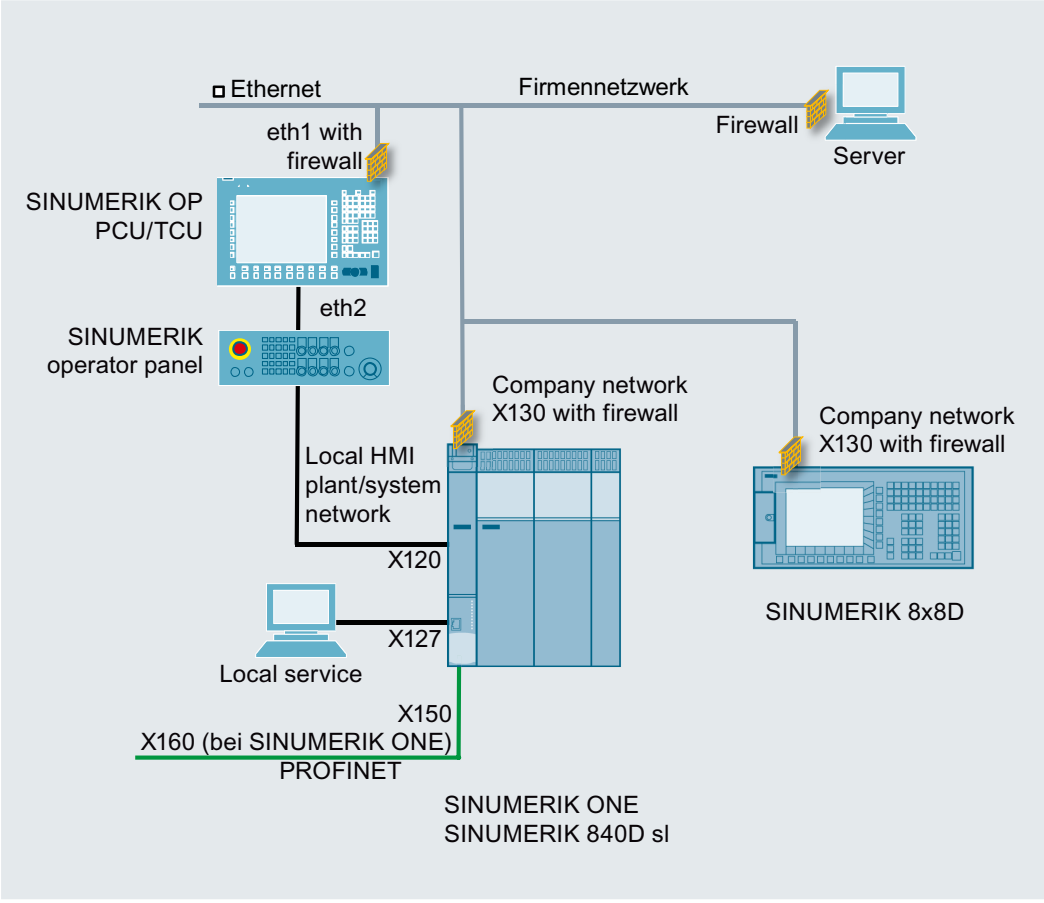


Figure 4-941 Networking of NCU/IPC

Firewall settings

Ethernet interface X130 of the NCU and the eth1 interface of the IPC are protected by a firewall for security reasons. If individual programs require access to a communication port for communication purposes, you can activate or deactivate the firewall via SINUMERIK Operate. Additional ports can be separately released.

Alternatively, you can configure the firewall via the "basesys.ini".

Further information

Further information on the configuration of the firewall and default settings can be found in the following manuals:

- SINUMERIK Operate (IM9) (https://support.industry.siemens.com/cs/de/en/view/109769186)
- Diagnostics Manual (808D) (https://support.industry.siemens.com/cs/de/en/view/109763685)

Physical protection of the NCU

NOTICE

Misuse, manipulation and theft

Modules, such as the NCU, are open equipment. If not protected, there is the risk of misuse by unauthorized personnel, manipulation or theft of data (e.g. CompactFlash Card).

- As a consequence, always install NCUs in housings and locked control cabinets or in electrical rooms. Only appropriately trained and authorized personnel may access these housings, electrical cabinets and electrical rooms. Information on the permitted locks can be found in Chapter Physical protection of critical production areas (Page 3222).
- **Further information** on the control cabinet installation of the NCU can be found in the SINUMERIK 840D sl NCU 7x0.3 PN Manual (https://support.industry.siemens.com/cs/de/en/view/99922219) or in the corresponding Equipment Manuals for SINUMERIK ONE: "NCU1750" and "NCU1760".

Machine control panels - SINUMERIK (MCP/MPP)

Machine control panels (**M**achine **C**ontrol **P**anels and **M**achine **P**ush Button **P**anels) are available for user-friendly operation of SINUMERIK machine functions.

Note

Only operate the machine control panels (MCP/MPP) on an internal, local machine network and secure them against any possible external access.

Note

Firmware update

For MCP/MPP/PP72-48 firmware updates or module diagnostics (Port 3845) contact Siemens Service&Support (https://support.industry.siemens.com/sc/ww/en/sc/2090).

See also

Passwords (Page 3229)

System hardening

Deactivating hardware interfaces

Deactivating interfaces

Measure	Description
Deactivate/activate Ethernet interfaces in the BIOS of the PCU	You can activate or deactivate the Ethernet interfaces in the BIOS of the PCU. You can find detailed information on this in PCU-Basesoftware (IM8) Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109748542), Chapter "BIOS settings".
Deactivating/activating USB interfaces	To prevent malware entering the control or the plant network via the USB interfaces, you can disable the USB interfaces of the NCU. Use the service command "sc_usb disable". Enter the relevant command on the Service Desktop in the "Run" dialog box or at the prompt. Use this function to make your system more secure and protect it from unwanted manipulation and malware. **Further information** can be found in the PCU-Basesoftware (IM8) Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109748542), Chapter "How to disconnect the USB interfaces".

Deactivating ports

Measure	Description
Deactivating the PROFINET port for SINUMERIK 840D sl PLC	In STEP 7 HW Config, a PROFINET interface port of a SINUMERIK PLC can be deactivated (X150). It is activated by default. The SINUMERIK PLC cannot be accessed via a deactivated PROFINET interface port. **Further information** can be found in the SIMATIC S7-300 CPU 31xC and CPU 31x Equipment Manual: Technical specifications (https://support.industry.siemens.com/cs/de/en/view/12996906), Chapter "Configuration of the port properties". **No communication function.** Note that no communication functions, such as PG/OP functions, open IE communication or S7 communication (PROFINET IO), are possible via a deactivated port.
Deactivating the PROFINET port for SINUMERIK ONE PLC	The PROFINET port can be deactivated in the TIA Portal under "Device configuration". Select the PLC and then switch to the menu "General > PROFINET interface> Advanced options > Port > Port options". Uncheck the box for "Activate this port for use". **Further information** can be found in the TIA Portal online help.
Deactivating a PROFINET port of SCALANCE X switch (possible as of the X200 series)	For secure operation, only one defined access point should be available to the network for diagnostics/maintenance. All of the other ports to the controls, devices, or switches (Scalance X) should be deactivated. This prevents unauthorized access. **Further information** can be found in the SIMATIC NET Configuration Manual: SCALANCE X-200 Industrial Ethernet switches (https://support.industry.siemens.com/cs/de/en/view/109757352), Chapter "Ports".

Communication services and used port numbers

SINUMERIK supports certain communication protocols. The address parameters, the relevant communication layer as well as the communication role and the communication direction are decisive for each protocol.

This information allows you to match the security measures for the protection of the automation system to the used protocols (e.g. firewall).

Further information can be found in the product information for "SINUMERIK port lists" (available soon).

Note

System hardening for software solutions

When using SINUMERIK Integrate software and other PC applications (e.g. Create MyConfig (CMC) or Access MyMachine (AMM)), make sure that the PC on which the software is used, always fulfills the requirements of industrial security.

These include, for example:

- Current Microsoft security updates
- Current virus scanner software
- Activated firewall, etc.

Further information can be found in Section System integrity (Page 3227).

Secure Boot with SINUMERIK ONE

Note

Only signed software

The SINUMERIK ONE NCUs have a Secure Boot feature that ensures that only software that is signed by Siemens can be loaded onto the NCU. This concerns both GIV software versions of the controller and any other software (e.g. SINAMICS TEC). Once a *.tgz file is imported and there is no accompanying *.sig file, the NCU will no longer ramp up.

In this situation, the controller can no longer be accessed via any interface. The previously installed software can no longer be deleted.

Virus protection

In the context of the length of the service life of a machine tool, the use of antivirus software does not make sense.

The reasons for this are as follows:

- **Continuous pattern updates necessary**
  The protection of an antivirus program is only as good as the up-to-datedness of its virus patterns. Providing this in day-to-day production of the machine without a direct Internet connection is not possible without further action.
- **Changed patterns and background scans affect the runtime behavior of the controller**
  Scans that run in the background can influence the system load and thus the system behavior of the machine. The longer the pattern list, the more time and resource-consuming the scan, which means the influence increases as the machine ages.
- **Short support times**
  As a rule, updates and patterns for antivirus software are provided by the end of the operating system support, at the latest. However, a machine tool generally has a significantly longer service life than a Windows operating system. Therefore the protection becomes ineffective against new threats over time.
- **Unforeseeable backlash effects on machine functionality**
  In certain circumstances, a pattern update can result in a desired system function being detected by the antivirus software as a suspicious operation and prevented from executing, which can have unforeseeable consequences for the operation of the machine.

Reasons for whitelisting

The use of whitelisting makes sense for protecting a Windows-based IPC in the SINUMERIK system for the following reasons:

- In a normal scenario, whitelisting does not require updates to continuously protect the machine.
- Whitelisting does not lose its protective effect over the service life of the machine (apart from technical progress).
- Whitelisting also protects against malware that may not yet be known to an antivirus program.

Note

Measures for protecting against viruses in a CNC environment

Take all the necessary measures for virus protection in the CNC environment. This also includes the proper handling of data storage media, USB sticks and network connections, precautionary measures when copying data and during software installations, etc.

See also

Virus signatures (https://support.automation.siemens.com/WW/view/en/19577116)

Whitelisting

SINUMERIK application example

Using the McAfee Application Control Software as example, a description as to how SINUMERIK PCU 50 with Windows XP can be "hardened" is provided. The licensed software can be used with the PCU 50 as a standalone version (Solidifier/Solidcore). The whitelisting software is directly purchased from the manufacturer.

A detailed description of this application can be found on the Internet (https://support.industry.siemens.com/cs/ww/en/view/89027076).

Virus protection / memory card

The memory card must be handled with particular care for all SINUMERIK devices that use a memory card so that no malicious software is loaded to the system.

⚠ WARNING

Risk of death due to software manipulation when using exchangeable storage media

Storing files on exchangeable storage media poses an increased risk of infection, e.g. with viruses and malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect files stored on exchangeable storage media from malicious software using appropriate protection measures, e.g. virus scanners.

Security updates / patch management

For organizational reasons, it is not possible to provide the latest Windows security patch when the **PCU /** the **IPC** is shipped.

Windows 10-based SINUMERIK IPCs are basically delivered with variant LTSB 2016. This version of Windows not only provides extended support, but also gives you the capability of installing patches for specific problems and there are no compulsory updates.

Provide your Windows-based devices with the current security updates in a timely manner. These updates should not take place during machine operation. Therefore, use a local WSUS server (see Chapter Defense in depth concept (Page 3221)).

Note

Before installing Microsoft Updates, note the following important points:

- **Prior to the update**, back up the system status for a fallback, if necessary. Ensuring the compatibility of the update with the individual system configuration is the responsibility of the customer.
- Never establish a direct connection to the WSUS server in the Internet! Ensure that the environment is secure and install an intermediate layer (e.g. DMZ network, firewall, SCALANCE S modules, etc.).

Since it is usually difficult to regularly update the Windows operating system of an IPC for a machine that is in use, and no more updates are available after the end of the support period, you should ensure the IPC is protected in accordance with the Defense in Depth concept (Page 3221), e.g. by means of a security router and the use of a whitelisting solution.

Possibly identified security weak points of the **NCU** are taken into account or corrected in the current CNC software version.

Note

Availability

The availability of Microsoft security updates is published via Microsoft Security Bulletins. The use of security updates is entirely up to the customer and is their sole responsibility. This can be realized based on the "evaluation of maximum severity" provided in the Microsoft Security Bulletin. Microsoft publishes information on security updates for the PCU and download links on the Internet (https://technet.microsoft.com/en-us/security/bulletins).

See also

Patch management (Page 3231)

Account management

Definition of access levels

Access to functions and machine data

The access concept controls access to functions and data areas. Access levels 1 to 7 are available, where 1 represents the highest level and 7 the lowest level. Access levels 1 to 3 are locked using a password and 4 to 7 using the appropriate key-operated switch.

Access level	Locked by	Area	Data class
1	Password: SUNRISE	Machine manufacturer	Manufacturer (M)
2	Password: EVENING	Service	Individual (I)
3	Password: CUSTOMER	User	User (U)
4	Key-operated switch setting 3	Programmer, machine setter	User (U)
5	Key-operated switch setting 2	Qualified operator	User (U)
6	Key-operated switch setting 1	Trained operator	User (U)
7	Key-operated switch setting 0	Semi-skilled operator	User (U)

Linux passwords / NCK

Level	User name	UID
1	manufact	102
2	service	103
3	user	104
4	operator3	105
5	operator2	106
6	operator1	107
7	operator	108

Corresponding to the named user, there is always a Unix group with the same name (also with the GID to the UIDs). As user, you are always member of your own group and also in all "lower-level" groups. For example, "operator2" is a member of the "operator2", "operator1" and "operator" groups. The file access rights are mainly controlled via these groups.

NOTICE

Data misuse caused by using passwords that are not secure enough

Data can be easily misused when passwords that are not secure enough are created. Passwords that are not secure enough can be easily hacked into.

The default passwords for the basic commissioning procedure are listed in the documentation.

- Therefore, always change the preassigned default passwords during commissioning
- Change the passwords at regularly defined intervals.
- For CNC software <V4.8: During commissioning, change the Linux password in addition to the SINUMERIK Operate passwords. You can find additional information in the Commissioning Manual "NCU operating system".
- A continuous warning appears on the SINUMERIK ONE if the default passwords are not changed.

Further information on assigning secure passwords can be found in Chapter Passwords (Page 3229).

Note

Changing passwords between SINUMERIK Operate and Linux

The access levels for SINUMERIK Operate and Linux are merged as of software version 4.8 SP3 (840D sl/828D) and 6.13 (SINUMERIK ONE). Changing a password for SINUMERIK Operate simultaneously changes the relevant password in Linux and vice versa. It is important to note the following behavior:

- When a general NC reset is performed, no passwords are reset to the default passwords.
- Following a software upgrade, the SINUMERIK Operate passwords apply to the NC unchanged.
- Once a password has been changed, it cannot be reset to its original state.
- When recommissioning the system with Restore [-full] (menu item in the Emergency Boot System "Recover system from USB memory stick (reformat CF card)"), the CF card is formatted and restored to a system in the delivery state. The passwords are not included in the SINUMERIK archive. Therefore, always change the default passwords after a Restore [-full] to individual passwords.

Safety Integrated password

In SINUMERIK Operate, commissioning data is generally protected via various access levels. You protect the safety-related drive parameterization additionally with the Safety Integrated password. This password is stored in the drive data so that it can be changed only by authorized persons who know the password.

Note

The SINAMICS Safety password must be used for SINUMERIK Safety

The assignment of the Safety Integrated password using the SINUMERIK Operate screen is supported as of V4.8 SP2 HF1. The assignment of the Safety Integrated password is also supported by a screen form of the SINUMERIK ONE Commissioning Tool.

- Always set a Safety password to prevent parameters from being changed using the external configuration software Starter or the commissioning software SINAMICS Startdrive.

Further information can be found in the Safety Integrated plus Commissioning Manual.

See also

SINUMERIK 840Dsl Safety Integrated plus Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109777982)

Further information

You can find **further information** on how you can change the passwords of the access levels along with other information on access levels for programs and softkeys and access rights for files in the SINUMERIK Operate (IM9) Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109769186), Chapter "General settings > Access levels".

CNC lock function

You can use the "CNC lock function" and the encrypted file that was created with SINUMERIK Integrate Access MyMachine (AMM) application to activate a lock date in the control. This allows the use of the machine to be limited to the time until the lock date is reached. The NC Start function of the control is locked when the lock date is exceeded.

The CNC lock function supports the business model with time-limited use. This protects against unauthorized use beyond the set interval.

Note

Only for SINUMERIK 828D

Note that the CNC lock function is only available on the SINUMERIK 828D controller.

Further information on the CNC lock function and on the creation of a lockset file can be found at:

- "SINUMERIK Integrate Access MyMachine /P2P (PC)" Operating Manual (https://support.industry.siemens.com/cs/de/en/view/109770206)
- "PLC" Function Manual, Chapter "P4: PLC for SINUMERIK 828D > CNC lock function"

Deleting the preinstalled SSH key

Application

Removing the SSH key preinstalled by Siemens reduces the risk of data misuse. However, in order to ensure sufficient access to the system, you can define and install your own SSH key.

Service command

The service command 'sc' is a tool used for performing a range of service tasks on a SINUMERIK NCU:

Syntax:	sc clear preinstalled-keys
Alternative names:	---
Authorization level	service

This command deletes all of the SSH keys preinstalled by Siemens on the control. When called from the service system, the keys on the CompactFlash card are affected, and not the SSH keys on the service system itself.

Further information

You can find additional information in the Base Software and Operating Software Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109763236).

PLC web server

In the delivered state, the PLC has no password and the Web server of the PLC is not activated.

Note

- If you activate the PLC Web server in the S7 project, you must define an appropriate user and an associated password for it. Create a secure password. When creating a new password, carefully follow the information provided in Chapter Passwords (Page 3229).
- Only use the HTTPS protocol to establish communication confidentiality and integrity.

Further information

Further information on the PLC web server can be found in the Function Manual S7-1500, ET 200SP, ET200pro web server (https://support.industry.siemens.com/cs/ww/en/view/59193560).

Access levels for softkeys

The display and operation of softkeys can be suppressed by both the OEM and the user. This allows the operating software to be specifically adapted to the required functional scope and therefore be configured as transparently as possible. To prevent access to functions in the operating software, or to restrict the possibility of operator errors, restricts the functional system scope.

Note

Applicability of modified access levels for softkeys

The setting of specific access levels for softkeys on a PCU only affects the respective PCU softkeys themselves. To implement access rights on the NCU, both the manufacturer and the user must use the appropriate mechanisms and set the rights accordingly.

Further information can be found in the SINUMERIK Operate (IM9) Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109769186), Chapter "Access levels for programs".

BIOS and AMT access protection

In order to prevent unauthorized access to the BIOS of the PCU 50 and the SIMATIC IPCs, make sure that you use a very secure BIOS password (see Section Passwords (Page 3229))

Further information

Further information on BIOS settings of the PCU 50 can be found in the PCU-Basesoftware (IM8) Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109748542).

Setting the password for AMT (Intel® Active Management Technology)

The Active Management Technology (AMT) function is used for the remote management of the PCU. For remote management, generally suitable protective measures must be taken (such as network segmenting) in order to guarantee secure operation of the plant.

For security reasons, AMT is deactivated when a PCU is delivered. When you activate AMT the first time in the BIOS setup, assign a strong password to prevent misuse of the remote management.

Further information on the AMT function of the PCU as well as the procedure for changing the password can be found on the Internet (https://support.industry.siemens.com/cs/de/en/view/52310936).

Password protection for Create MyConfig (CMC)

NOTICE

Data misuse due to incorrect assignment of rights

Access data, such as the pre-configured passwords for access to the control system, can be stolen and misused.

- For that reason, set up organizational measures to ensure that only authorized persons are given access to these files.

Note

Password protection for linked external files

The protection mechanisms integrated into CMC (password protection) are ineffectual for linked external files that are integrated into the CMC context.

Note

Protecting CMC packages from reimporting

Note that CMC packages have to be protected by password against being reimported.

- For that reason, always set up a password against reimporting when you assign a password for a new project.

Know-how protection

The following functions provide protection of technological know-how and prevent unauthorized access to the SINUMERIK controllers:

SINUMERIK Integrate Lock MyCycles

Using the "SINUMERIK Integrate Lock MyCycles" (cycle protection) function, cycles can be encrypted and then stored protected in the control. The cycles are encrypted outside the control using the SINUMERIK Integrate Access MyMachine/P2P program.

For cycles with cycle protection, execution in the NC is possible without any restrictions.

In order to protect the manufacturer's know-how, any type of view is inhibited for cycles with cycle protection.

This software option is available for the SINUMERIK 808D, 828D and 840D sl controllers.

You can find an application example for cycle protection for SINUMERIK on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109474775).

Further information

Further information on cycle protection can be found in the SINUMERIK Access MyMachine / P2P (PC) Operating Manual (https://support.industry.siemens.com/cs/ww/en/view/109770206).

SINUMERIK Integrate Lock MyPLC

With the aid of block properties, you can protect the blocks created in the SIMATIC PLC from unauthorized changes, for example.

The block properties should be edited when the block is open. In addition to properties that can be edited, data that is only displayed for your information is also displayed in the respective dialog field: It cannot be edited.

A block that has been compiled using this option does not allow you to view the instruction section. The interface of the block can be viewed, but not changed.

Further information

You can find further information on block protection in the SIMATIC Programming with STEP 7 Programming and Operating Manual (https://support.industry.siemens.com/cs/de/en/view/109751825), Chapter "Block properties".

Note

The integrated CP in the SINUMERIK 840D sl does not support the "Module access protection / protection level" option.

Encryption of blocks

As of STEP 7 Version 5.5 SP3 and the CNC system software V4.5 SP2 for 840D sl/ 840D sl or V6.13 for SINUMERIK ONE, you can create encrypted block protection for functions and function blocks in the offline and online view. You can use this function to encrypt your blocks and protect the block code against external access.

The option "SINUMERIK" and, if required, "SIMATIC" must be selected for the encryption with SINUMERIK.

A detailed procedure of how to encrypt your blocks can be found on the Internet (https://support.automation.siemens.com/WW/view/en/45632073).

OPC UA

OPC UA (Unified Architecture) is a standardized, industrial communication protocol for access to control data, e.g. by higher-level control systems. Variables of a SINUMERIK 840D sl, SINUMERIK 828D or SINUMERIK ONE can be read and written to via this communication protocol using the SINUMERIK Integrate Access MyMachine /OPC UA software option.

NOTICE

Date misuse resulting from an insecure connection to the client

There is a danger of data misuse due to an unencrypted connection to the OPC UA client.

- Therefore, always encrypt your connection to the OPC UA client.
- Information on the encryption of the data connection can be found in the SINUMERIK Access MyMachine /OPC UA Configuration Manual (https://support.industry.siemens.com/cs/de/en/view/109777871).

NOTICE

Data misuse due to incorrect user administration / rights assignment

A significant security risk can ensue through incorrect user administration and faulty right assignment. Users can access data or actions for which they have not been authorized.

- As a consequence, always very carefully consider which users are assigned which rights. As administrator, you are responsible for professional user administration and assignment of rights.

Note

Selecting a secure password

Always set a secure password for your connection to the OPC UA client! Further information on selecting a secure password can be found in Section Passwords (Page 3229).

User administration in the TIA Portal

The TIA Portal gives you the capability of using user administration for projects. In this way, for example, a project can be protected against unintentional or unauthorized modification. A user sets up the project protection to activate user management. This user is created as a project administrator. Once the project protection has been activated, the project can only be opened and edited by authorized users. Note that project protection cannot be revoked.

User administration via the TIA Portal is available for the SINUMERIK 840D sl and SINUMERIK ONE controllers.

UMC - User Management Component

In addition, you can install the "User Management Component UMC" software package, which provides central user administration, on one or more computers.

Further information

Further information on the topic of secure user administration can be found in the TIA Portal online help.

SIMATIC Logon

User administration and traceability

The SIMATIC Logon option package is used to set up access rights for products and libraries in STEP 7. These projects can therefore only be accessed by an authorized group of people. SIMATIC Logon can be used in conjunction with SINUMERIK STEP 7.

More detailed information can be found in Chapter Secure access control with SIMATIC Logon (Page 3261).

Data backup

There are different archive forms and archiving methods in the SINUMERIK for the different components.

Time of the data backup

Perform a data backup at the following times:

- After commissioning
- After changing machine-specific settings
- After the replacement of a hardware component
- Before and after a software upgrade
- Before the activation of memory-configuring machine data

You can find **further information** in the following manuals:

- SINUMERIK 840Dsl Commissioning CNC Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109777906)
- Installation Manual SINUMERIK ONE New installation and upgrade

Note the general information on secure data storage with regard to archives in Section Data storage (Page 3229).

NOTICE
Misuse of confidential data on the control system On the control system, there is a risk of confidential data being misused. • As a consequence, it is not permissible to load confidential data to the control (e.g. using the "SINUMERIK Integrate Access MyMachine/P2P" software). • Always store confidential data in an encrypted form locally on an encrypted storage location in the network.

Disposal

NOTICE
Misuse of data resulting from insecure methods of deleting data Incomplete or insecure deletion of data from memory cards or hard drives can lead to misuse of the data of the part programs, archives, etc. by third parties. • Therefore ensure that all storage media are securely deleted **before disposing of the product** : • There are programs that support you in securely deleting/formatting storage media. Alternatively, contact a certified data destruction specialist to take care of this task.

4.9.7.2 CNC Shopfloor Management Software

The following chapter provides you with an overview of the security-related measures you can take to protect your CNC Shopfloor Management software products from threats. Detailed descriptions and procedures can be found in the corresponding documentation of the CNC Shopfloor Management software.

System overview

A leading-edge IT architecture is created based on the CNC Shopfloor Management Software, more specifically, at three levels – "In Cloud", "In Line" and "In Machine". These levels correspond to the 3 platforms "MindSphere", "SINUMERIK Integrate" and "SINUMERIK/SINUMERIK Edge", with their numerous tailored functions from the field to the cloud.

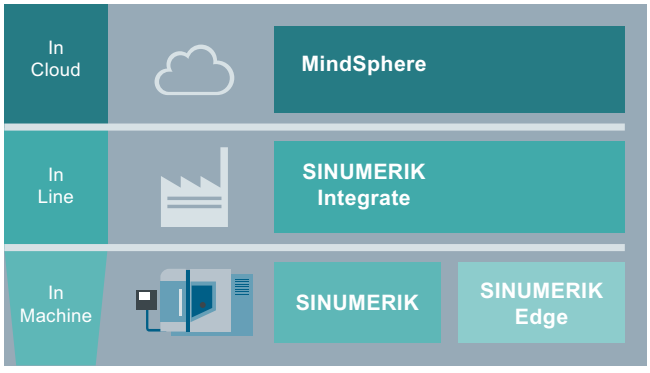


Figure 4-942 CNC Shopfloor Management Software

Cloud applications (In Cloud)

MindSphere provides state-of-the-art security during on-site data collection and when transferring and saving to the Cloud.

The security framework is oriented on the principles of industry standards, e.g. IEC 62443, International Organization for Standardization (ISO)/IEC 27001, and the Federal Office for Information Security (BSI) and recommendations from authorities on working with data in Cloud environments.

In accordance with proven communication practices of the industry, all communication between the client and MindSphere is protected by TLS V1.2 via public end points. Reliable x509 certificates from the Siemens Trust Center are used. These correspond to the requirements of the European Telecommunications Standards Institute (ETSI) and of the Certification Authority Browser Forum (CA/B Forum).

Further information on encryption for MindSphere can be found in the MindSphere Whitepaper (https://www.plm.automation.siemens.com/global/en/topic/mindsphere-whitepaper/28842).

Saved data is always saved by Siemens on high-performance servers in the computer centers of the infrastructure providers. All of the infrastructure centers meet the highest standards for data security and are protected against cyber threats. As commercial providers of a Cloud IaaS (Infrastructure as a Service), they provide higher security standards than typical private, local facilities for data storage. The computer centers are operated in accordance with the Best Practices of the industry.

As an additional layer of security, all of the Cloud infrastructure partners must ensure on-site security measures such as electronic photo ID badges, card owner access control, biometrics, digital video monitoring with recording and alarm monitoring.

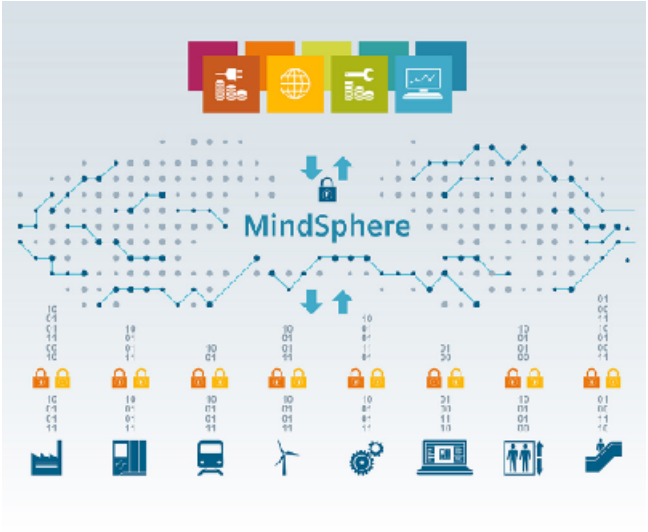


Figure 4-943 MindSphere

Manage MyMachines /Remote

Security measures and system hardening

Note

As a user of Manage MyMachines /Remote, always ensure that you operate the product with the latest versions of SINUMERIK Integrate client and Manage MyMachines /Remote Service Engineer and Machine Operator Clients. Also follow the guidelines for industrial security contained in Chapters 1.3 and 2.3 of the Manage MyMachines /Remote Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109759394).

The connection of SINUMERIK controls to MindSphere via "TLS 1.2 /HTTPS" meets the highest security standards.

The automatic confirmation of the machine identity, used in conjunction with token provided in MindSphere for the onboarding of the machine, ensures that the correct machine is accessed during a remote session.

Disposal

To completely delete an installation of Manage MyMachines /Remote, you should ensure that all of the software and certificates have been properly deleted from your Windows device or SINUMERIK control system, including backup systems. The data will continue to be available in MindSphere, unless the tenant is closed.

Additional information on the topic of disposal can be found in the Manage MyMachines / Remote Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109759394).

Further information on general concepts for secure remote access to industrial plants can be found in the following documents:

- cRSP IT Security Concept (https://support.industry.siemens.com/cs/ww/en/view/109759394)
- Siemens common Remote Service Platform (cRSP) (https://www.downloads.siemens.com/download-center/Download.aspx?pos=download&fct=getasset&id1=A6V11272777)

See also

cRSP IT security concept (https://www.downloads.siemens.com/download-center/Download.aspx?pos=download&fct=getasset&id1=A6V11272775)

PC/Server/Desktop applications (In Line)

SINUMERIK Integrate 4.1

Access to the resources of the SINUMERIK Integrate server

Note

Access to the resources of the SINUMERIK Integrate server

Read and write access to the file system and resources of the operating system (in particular to the Windows Registry) of the SINUMERIK Integrate server is only enabled for users with administrator rights. Make sure that these administrator IDs have sufficiently strong passwords.

NOTICE

Data manipulation possible

Within the production/machine network (Intranet), there is a risk that a hacker can access the file system of the SINUMERIK Integrate Servers or the variance SINUMERIK Integrate clients. There, the hacker can manipulate various system components (e.g. the content of databases). As a consequence, the attacker can change tool data, NC programs, machine archives or the system structure itself, for example. SINUMERIK Integrate cannot prevent this type of attack.

- As a consequence, as the person responsible for the machine network, it is imperative that you take the appropriate industrial security measures for the production/machine network.

Use of open programming interfaces

NOTICE

Data misuse by using open programming interfaces

There is a potential risk of data misuse when using open programming interfaces.

- Therefore, when using open programming interfaces, only use clients that at least communicate with the SINUMERIK Integrate server via "TLS 1.2 /https" communication paths.
- Note that the corresponding TLS version is not supported by every operating system.

System hardening

System hardening is the removal of all software components and functions that are not absolutely required by the desired application to fulfill the intended task.

System hardening of third-party software: Microsoft™ Internet Information Server, Microsoft™ SQL Server, browsers (Microsoft™ Internet Explorer, Mozilla Firefox)

SINUMERIK Integrate requires software from third parties: For example, the Microsoft™ Internet Information Server, Microsoft™ SQL Server products and various browsers. They must be hardened according to the latest technology. In particular, restrict access to the Microsoft™ SQL Server to the local host and protect access with a password. Encrypt access to the database. In addition, you should only use current browsers for communication with the SINUMERIK Integrate server. Secure communication cannot be guaranteed when using outdated browsers (SSL instead of TLS).

System hardening of neighboring products to SINUMERIK Integrate: For example, tool setting station, Teamcenter, AUVESY™ VersionDog

SINUMERIK Integrate communicates with neighboring software products, e.g. tool setting stations, Teamcenter and VersionDog. They must be hardened according to the latest technology so that there are no negative effects via this communication path.

Network file exchange via common drives

Note

Network file exchange via common drives (Server Message Block, SMB)

If you use SMBs for exchanging files with SINUMERIK Integrate functions, only use standard authentication mechanisms (user name / password). Also restrict the accesses for each user accordingly. Data storage on shared drives should be kept to a minimum.

Data backup

For data backup on machine tools, see Section "Data backup (Page 3249)". Also back up the server side. Relevant are the configuration and contents of the Microsoft SQL Server database, the SINUMERIK Integrate server and the following SINUMERIK Integrate applications:

- MMT
- SFT RM
- MMP
- AMC
- AMP

Perform a data backup at the following times:

- After commissioning
- When changing the hardware configuration
- After replacing the hardware
- Before and after upgrading the software

Communication security for SINUMERIK Integrate applications

The communication between the SINUMERIK Integrate server and various client forms (such as MMT or browsers) is preconfigured with HTTP. The required configuration should provide support for service technicians trained specially for SINUMERIK Integrate. If possible, switch the communication to HTTPS, which requires a separate certificate. It should be noted that HTTPS is not possible for all client/server combinations.

Cloud mode

If the AMM and AMC applications are used in cloud operation, the Siemens AG as operator ensures the security of the SINUMERIK Integrate server. Customers only have to ensure the security of the infrastructure on the machine side.

Firewall of the machine network

In contrast to standalone operation, a connection to the cloud server outside the machine and company networks is required when using the SINUMERIK Integrate AMM and AMC applications. The associated firewalls must enable the required ports. However, only the required ports should be opened. Further information on the required firewall settings can be found in the SINUMERIK Integrate Installation Manual, Section "System requirements".

Phishing for passwords

"Phishers" could attempt to obtain login data, which would allow them to carry out actions within SINUMERIK Integrate in the name of the user. Hackers could, for example, falsify e-mails and websites from SIEMENS and thus gain access to confidential information from SINUMERIK Integrate users. The users are then prompted, for example, to enter the access data in a form and then send it to their SINUMERIK Integrate organization.

Note

You should observe the following when you encounter suspicious e-mails:

- Be on your guard when you receive e-mails from someone you do not know, especially if the e-mails include links and attachments. Never open suspicious attachments and do not click on any links in the e-mail.
- Carefully check the sender’s complete e-mail address.
- Check the integrity of the links embedded in the e-mail (e.g. by moving the mouse over the link). Tell-tale signs are spelling mistakes or where links contain a confusing company name.
- Use digital signatures in emails.
- If in doubt, never divulge any confidential information.

Handling of confidential information

The communication route between the company's firewall and the SINUMERIK Integrate server is protected against malicious attacks by secure communication protocols (TLS). Third parties cannot eavesdrop on the transferred information or change it. Also note the company-specific guidelines for the transfer of confidential information via AMM and AMC.

Controller-related applications (In Machine, SINUMERIK Edge)

SINUMERIK Edge

Overview

The SINUMERIK Edge is a remote-controlled edge device that functions as a field gateway as well as a computation node for any user workload within an extended IoT/OT architecture. Thus, SINUMERIK Edge allows a vertical flow of information and data processing between all layers:

- In Machine
- In Line
- In Cloud

This also contains the temporary or permanent saving of process data. SINUMERIK Edge thus has the task, through its security architecture, of not allowing any regression/erosion of the present network security and of the data protection level. In order to not cancel the individual security mechanisms of the SINUMERIK Edge, organizational support is also needed here.

Security features and security measures

The SINUMERIK Edge is equipped with 2 physical network connections (RJ45), which, according to the manual, are to be used for connecting to the In Machine and In Line level. Ensure that the port assignment is correct due to the following reasons:

- The communication for the "In Machine" network is assumed to be unprotected for the most part.
- No uncontrolled connectivity to higher-level networks (In Line, In Cloud) is possible for the "In Machine" network.

Through a multi-level network architecture, the SINUMERIK Edge ensures the isolation of both networks, which is only overcome by an application-defined flow of data. Due to the use of container technology, there are additional mechanisms for isolating the workload (Edge application) with regard to network, memory and CPU resources.

Communication of the SINUMERIK Edge in the direction of "In Cloud" and "In Line" always takes place via an encrypted end-to-end channel (TLS 1.3). Supplemental to this, the integration into a PKI-based trust chain is supported. This ensures both its restriction to only permitted communication partners and a trustworthy transmission. For the In Line exchange of data in environments with special security requirements, client-based authorization via client certificates is also possible.

The initial exchange of the certificates needed for secure communication between the Edge Management System (MindSphere / In Cloud) and SINUMERIK Edge (In Line) takes place during the so-called onboarding procedure. Onboarding includes the exchange of a "shared secret", which connects a logical device (MindSphere asset) with a physical device (SINUMERIK Edge). Since this exchange does not take place via the same communication infrastructure, a compromise can be ruled out as early as the onboarding procedure. A second aspect of the onboarding is the linking/integration of the MindSphere IoT services (Timeseries Store, FileStore, Fleetmanager, etc.) in the correct MindSphere tenants. The SINUMERIK Edge platform also

ensures that no data flow into a tenant or asset that is not defined for this purpose can be established at any time.

Note

The requirement for using SINUMERIK Edge is a MindSphere tenant, including a valid MindAccess account (at least IoT value plan S).

The SINUMERIK Edge only communicates via "outgoing" connections. This means that no exposition of the SINUMERIK Edge on the In Line or In Cloud level is needed. Rather, this scenario is discouraged. Regardless of this configuration, the reachability of the MindSphere end points from the SINUMERIK Edge must be temporarily guaranteed. This concerns onboarding, the firmware update, or the (de)-installation of Edge applications. The SINUMERIK Edge allows applications (Industrial App) to not only provide data via a controlled path In Cloud, but these applications also provide user interfaces and/or interfaces (APIs), to allow new workflows (In Line) or to supplement existing ones. To this end, applications may provide separate user and access management options. The associated security information can be found in the relevant documentation.

Communication of the SINUMERIK Edge with the SINUMERIK only takes place via the "In Machine" network and is encrypted in accordance with the respective protocols. The authorization mechanisms vary depending on the protocol used, however. Since some protocols are protected using weak protection mechanisms, it is important in such cases to adhere to adequate password guidelines and to ensure within an organization that passwords are never saved or are only saved in urgent cases.

The SINUMERIK Edge is also protected against unwanted manipulation or weakening of the security features on both the firmware and application levels by the following features:

- Measured / Secured Boot
- Full Disk Encryption
- Rootless Access

To ensure a high level of security of the SINUMERIK Edge over a long period of time, the firmware is continuously being further developed and hardened. This is required to adapt to the ever intensifying cyber security threat situation. For this purpose, an update mechanism is available as part of the SINUMERIK Edge firmware, which is integrated into the corresponding IT process as part of a continuous security strategy.

4.9.7.3 SIMOTION

SIMOTION security measures

The following section provides an overview of the Industrial Security features available for SIMOTION (Motion Control) in order to protect your plant against threats.

Security functions

- There is only compiled code on the controller by default. For this reason, no upload and consequently no re-engineering is possible.
- No modifications can be made to the configuration without the matching engineering project.
- Know-how protection for source programs with password and encryption.
- Applicative copy protection for the configuration on the control system
- Detection of source code manipulation with the SIMOTION SCOUT engineering system.
- Activating/deactivating unused functions (web server, OPC UA server, ports, etc.)
- Use of the SIMATIC Logon for access to a project only with the appropriate rights.
- Virus scan and security updates for SIMOTION PC-based controllers (SIMOTION P).

A production plant is typically divided into several different network segments. These "segments" are components that have the required security functions connected upstream. They are shown with a padlock symbol in the overview graphic.

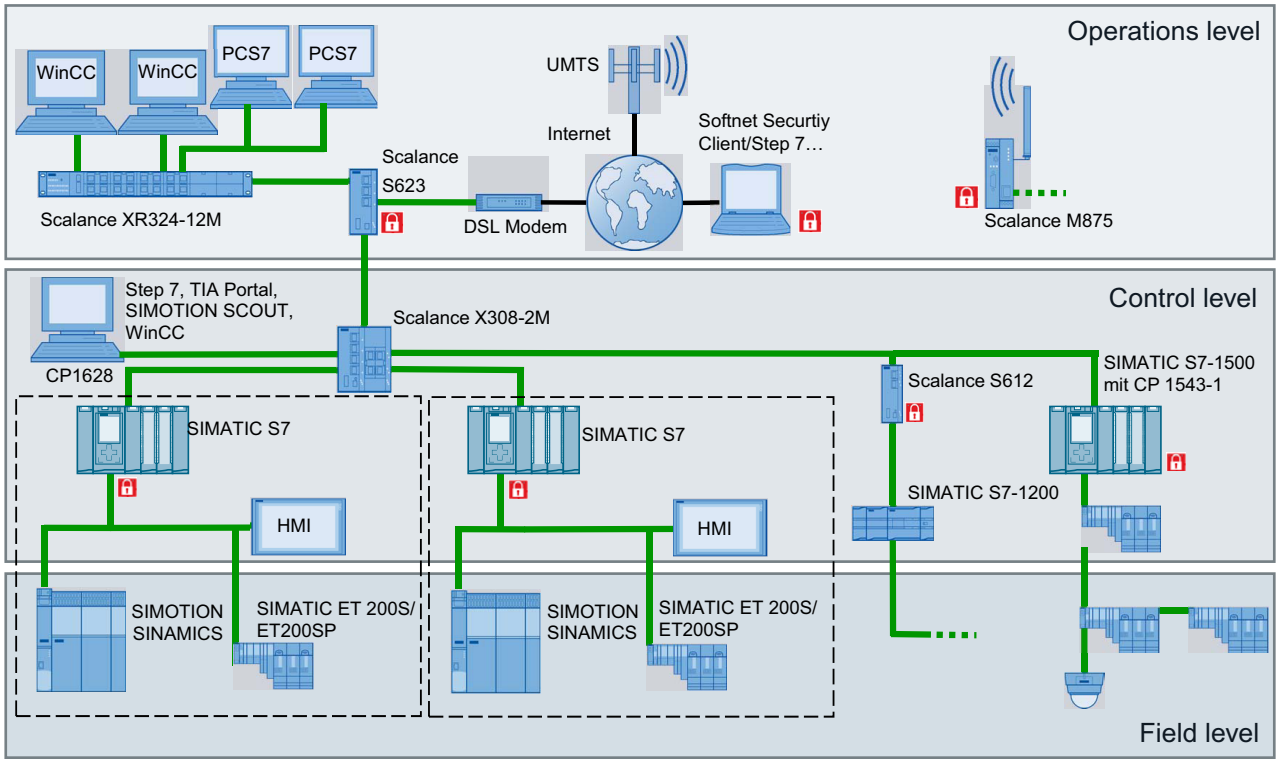


Figure 4-944 Display of a typical production plant with protected areas

Reference

Detailed descriptions and further procedures can be found in the corresponding SIMOTION documentation.

Many products (SINUMERIK, SIMOTION, SINAMICS) contain OpenSSL. The following applies to these products:

- This product contains software (https://www.openssl.org/) that has been developed by the OpenSSL project for use in the OpenSSL toolkit.
- This product contains cryptographic software (mailto:eay@cryptsoft.com) created by Eric Young.
- This product contains software (mailto:eay@cryptsoft.com) developed by Eric Young.

System hardening

Port security

Deactivating hardware ports

As of version 4.4, individual hardware ports of PROFINET interfaces (e.g. X150 interface ports) can be set to **Disable** in the engineering system (HW Config) for SIMOTION devices. This prevents devices being connected without permission and also increases security in terms of third-party access to the system. You should therefore deactivate unused ports.

Note

A SIMOTION device can no longer be accessed via a deactivated PROFINET interface hardware port.

The engineering system and the PN stack ensure that at least one port on each interface is not set to **Disable** to prevent users locking themselves out. The default setting is **Automatic settings**.

Further information

Further information on the logical Ethernet ports and protocols used for SIMOTION can be found in the Communication with SIMOTION System Manual (https://support.industry.siemens.com/cs/de/en/view/109767623), Chapter "Services used".

Virus scan, Windows security patches, SIMOTION P

General information on virus scanners

Once an industrial PC system is connected to the Internet, either directly or via an internal company network, there is a danger that it can become infected with a virus. However, malicious software is not only able to reach the system via the Intranet/Internet, but also, for example, via a removable storage device (such as a USB memory stick) attached to the system for backing up data.

SIMOTION P320 virus scanner

A virus scanner that runs on Microsoft Windows, as used in office or home computers, has a deep impact on a system's processes. There are, for example, processes such as real-time scans or regular system scans. Such interventions can cause performance issues for the system, and as a result, for the SIMOTION Runtime software. Although the SIMOTION Runtime software runs in a real-time environment, it still depends on the available system resources.

Note

Because of the resulting performance impairments, the installation and use of a standard virus scanner on a SIMOTION P320 during system runtime is not permitted.

Using a virus scanner

As a standard virus scanner cannot be used for SIMOTION P320, an alternative procedure is followed. The virus scanner is installed to a separately bootable Windows PE operating system. It is started, for example, from a CD or a USB storage device and then performs a virus scan.

Note

FAQ Service & Support portal

More information on using a virus scanner on a SIMOTION P320 can be found in the FAQ "How can a virus scanner be used on a SIMOTION P3x0?" (https://support.automation.siemens.com/WW/view/en/59381507) which is available as a download from the Service & Support portal.

Secure project storage

Project data storage in SIMOTION SCOUT

All relevant data, configurations and programs are stored in the project. Only the programs and libraries encrypted via the know-how protection can be stored in a project. To protect the entire project, you should protect the project data with conventional office solutions, e.g. password-protected archives or encrypted hard disks.

File structure

The SIMOTION SCOUT project data can come in the following formats:

Engineering data (ES)

- Standard storage: File structure in the project tree
  STEP 7/TIA Portal and SIMOTION SCOUT objects in the project directory. These objects are not secure and can be edited by anyone if there is no know-how protection for programs and libraries or external file encryption is used. Programs in this context programs are synonymous with units, which can contain the programs, function blocks and functions.
- XML data
  Project data created via an XML export/import. The know-how protection is retained.

Runtime data (RT) - data on the CF card

- ZIP archive of the SIMOTION project (not binary).
  The project archive is stored on the memory card of the respective SIMOTION controller (CFAST, CF card, MMC). The archive can be transferred, e.g. via SIMOTION SCOUT or using standard methods (FTP transfer).
- Binaries (zipped, unzipped)
  Binaries contain the compiled, executable project with the configurations and applications. Changes cannot be made during runtime without the SIMOTION SCOUT project because the project is stored as binary data on the SIMOTION controller.

The following figure shows an example of possible project data storage with display of the protected data.

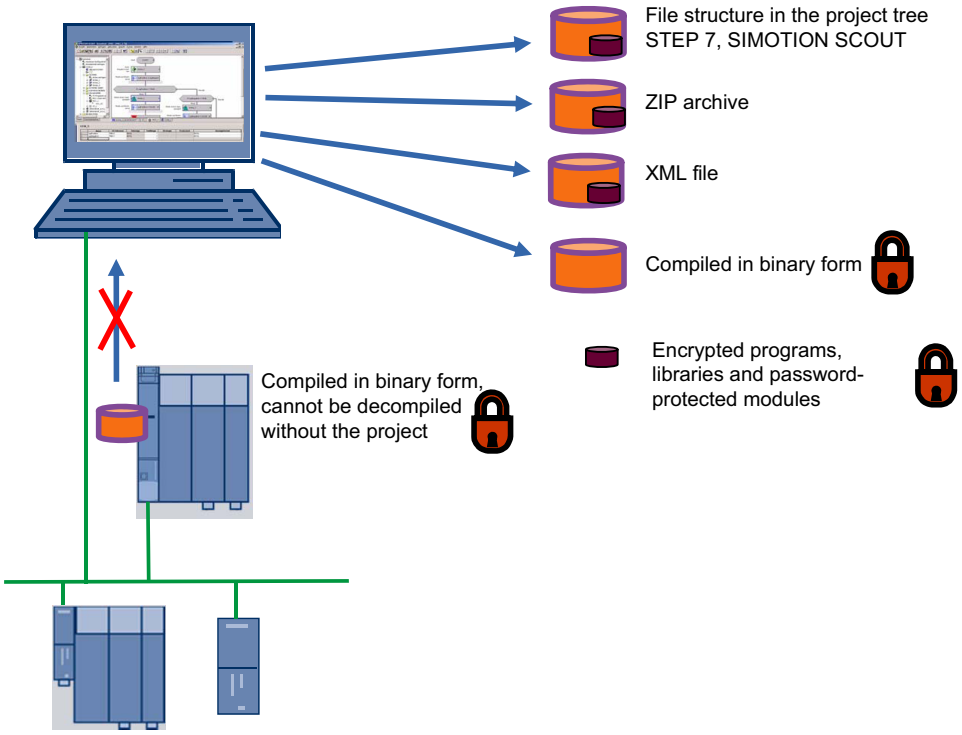


Figure 4-945 SIMOTION SCOUT project data storage

Know-how protection

Secure access control with SIMATIC Logon

User administration and traceability

The SIMATIC Logon option package is used to set up access rights for products and libraries in STEP 7. These projects can therefore only be accessed by an authorized group of people. SIMATIC Logon can be used in conjunction with SIMOTION SCOUT.

SIMATIC Logon supports the following functions:

- Assignment of individual authorization levels to users or user groups for the execution of specific actions (e.g. read, write, transfer blocks).
- Logging of online activities and logon actions on the computer. Access and changes in the project are reproducible.
- Assignment of authorization to users / user groups only for a limited time.
- Password aging strategies

Change log

A change log can be recorded when the access protection is activated. This includes, for example:

- Activation
- Deactivation
- Configuration of access protection and the change log
- Opening and closing of projects and libraries including their download to the target system as well as activities to change the operating state

Know-how protection in engineering

Know-how protection types

The know-how protection in SIMOTION SCOUT prevents unauthorized viewing and editing of your programs. Multiple logins are possible. The standard login can be set for the engineering session.

A distinction is made between two types of know-how protection:

- Know-how protection for programs and libraries
- Know-how protection for drive units (as of SINAMICS V4.5)

You set a login and a password under **Project -> Know-how protection**. The know-how protection for the program is activated via the **Set** menu command. The programs contained in the project are still visible to the user in this session, but the program names are displayed with a padlock symbol.

Programs and libraries

The know-how protection protects the programs and libraries in your project. Unauthorized viewing and editing of your programs is prevented when the know-how protection is activated. You can set the know-how protection for individual programs or for all programs in a project.

Access protection and encryption can be set in several levels for the following types of data:

- Programs (units in Structured Text (ST), Motion Control Chart (MCC) and LAD/FBD that contain programs, function blocks, and functions)
- Drive Control Charts (DCC)
- Libraries

You can select three different security levels for the encryption:

- **Standard**
  Access only with user login and password (backward compatible with versions before V4.2).
- **Medium**
  Improved coding of the password (due to a new procedure, no backward compatibility without knowledge of the password).
  Programs and libraries can be recompiled at any time even without knowledge of the password.
- **High** (only for ST source files in libraries)
  Compilation is only possible after the password has been entered.
  Protected libraries can also be used after an export without knowledge of the password, because in this case the compilation result is also exported.
  - **An export without source texts is also possible when exporting libraries**
    Highest protection. Complete removal of the source texts in the engineering upon export. The export only contains the compilation result (recompilation no longer possible).

The block interfaces are always visible.

Drive units in SIMOTION SCOUT

The know-how protection for drive units only applies online and is used to protect intellectual property, in particular, the know-how of machine manufacturers, against unauthorized use or reproduction of their products.

A detailed description can be found in Chapter Know-how protection (Page 3271).

Copy protection for the configuration on the control system.

Copy protection for SIMOTION projects

Measures can be taken to tie the configuration to the memory card or the controller. This prevents illegal duplication of the configuration.

The serial numbers of the CPU, memory card and DRIVE-CLiQ components in the application can be queried via system functions. This enables the machine manufacturer to create a block with an encryption algorithm which generates a key from the currently installed serial numbers during runtime and compares it with a machine key. Each machine configuration has a specific machine key which is generated by the machine manufacturer and stored in the application, and which can be entered by the end customer, for example, via the HMI, particularly during maintenance work.

In addition, special agreements can also be made regarding extended know-how protection and copy protection through the use of a SIMOTION Open Architecture technology package.

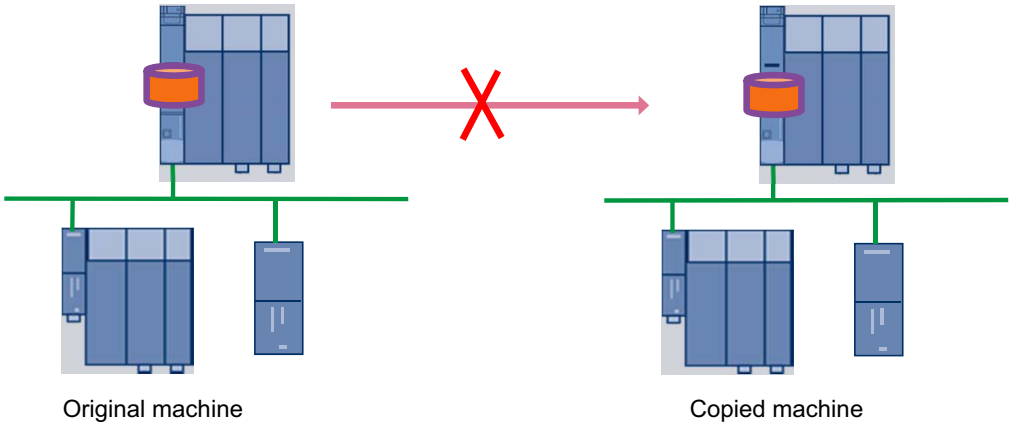


Figure 4-946 Copy protection of binary SIMOTION SCOUT projects

Offline/online comparison

Project comparison

You can use the SIMOTION SCOUT/STARTER **Project comparison** function (start this via the **Start object comparison** button) to compare objects within the same project and/or objects from different projects (online or offline).

The offline/online comparison is used to detect in detail any subsequent manipulations of the project data on the plant in comparison to your secured engineering data. Thus you check if any unauthorized third parties accessed the system.

The following comparisons are possible:

- Offline object with offline object from the same project
- Offline object with offline object from a different project
- Offline object with online object

The project comparison in SIMOTION SCOUT contains all objects in a project, such as SIMOTION devices, drive units, libraries, programs (units), technology objects, I/Os as well as the configuration of the execution system.

The offline/online comparison provides support for service jobs or for detecting changes to the project data.

It may, for example, be the case that inconsistencies are indicated when you switch to online mode in the project navigator, i.e. there are deviations between your project in SIMOTION SCOUT and the project loaded into the target system.

Possible causes can include, for example:

- A program has been changed
- The result of compiling a program is different
- There is a deviation on the global device variables
- The execution system has been changed

- The hardware configuration has been changed
- A library has been changed
- A configuration data item for an axis has been changed

The object comparison allows you to establish these differences and, if necessary, run a data transfer to rectify the differences.

Detailed offline/online comparison

You can determine specific differences between the offline and the online project by performing a complete project comparison. If there are discrepancies, you can determine the changes/ manipulation to the source code down to the program line level, when the additional information (source information) has also been stored on the target system during the download. This is also possible with the LAD/FBD and MCC graphical programming languages.

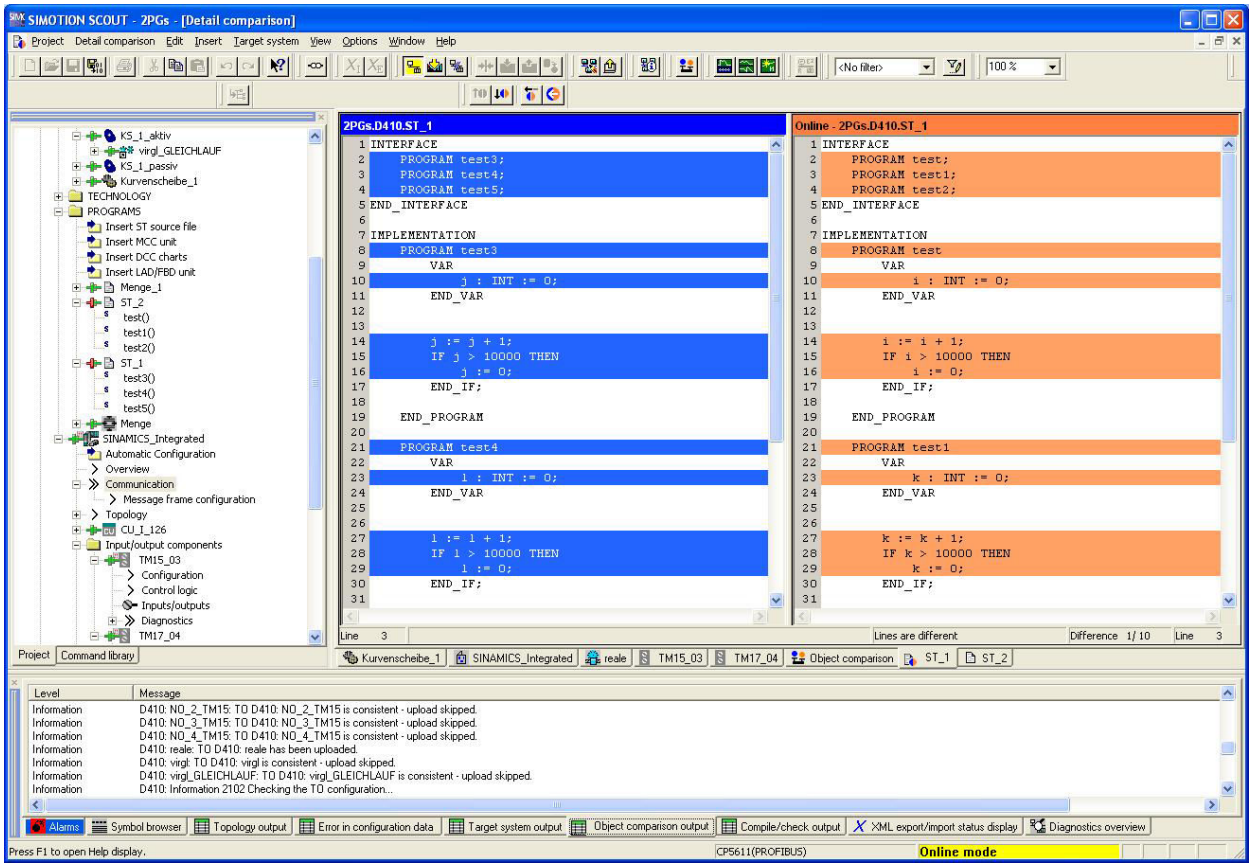


Figure 4-947 Example of ST detail comparison

SIMOTION IT Web server

Introduction

SIMOTION devices provide a Web server with preprepared standard websites. These websites can be displayed via Ethernet using a commercially available browser. Additionally, you have the option of creating your own HTML websites and incorporating service and diagnostic information. The web server can be deactivated. If the Web server is active, secure operation of the plant can be ensured via the integrated security concept and the user administration.

Deactivating/activating the Web server

The Web server with all functions and services can be activated or deactivated in the SIMOTION SCOUT or SIMOTION SCOUT TIA project under the hardware configuration of the controller. You can activate or deactivate individual functions.

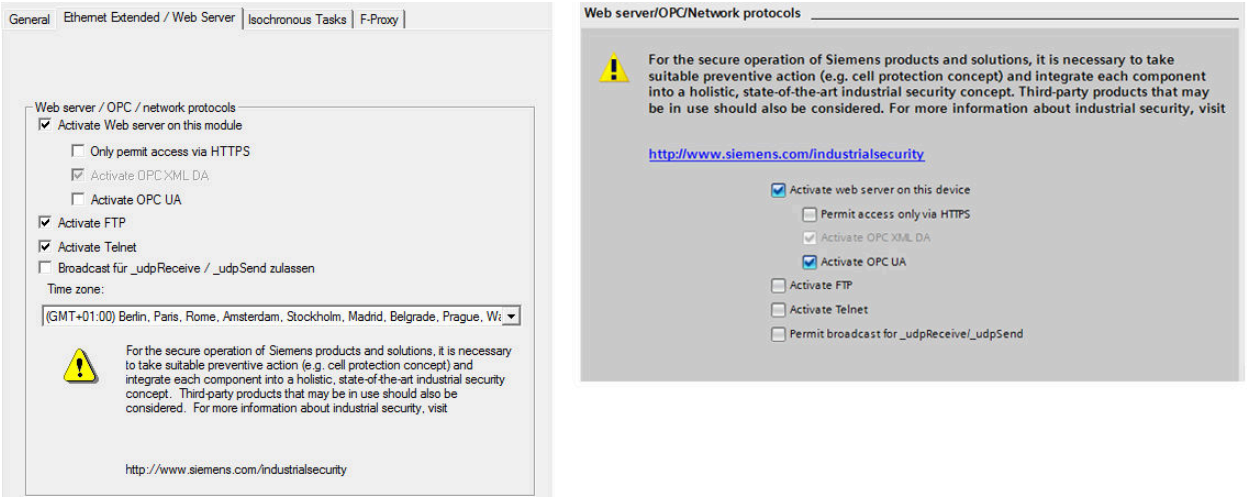


Figure 4-948 **Activating the SIMOTION IT Web server functions in SIMOTION SCOUT or SIMOTION SCOUT TIA**

Note

To activate the Web server, you must establish a user administration scheme with password-protected user access.

Security concept of HTTP/S, FTP and Telnet access on the Web server

As of version V4.4, access to the SIMOTION IT Web server is protected by a multi-level security concept.

The security status of the Web server is indicated by the security level on the website. This security level can have three different levels: Low, Normal, High

Security Level Low

The device is supplied with an empty user database. No projects exist yet. The security level is low to allow configuration of the device.

- In this state, access to the Web server as an anonymous user is possible to enable use of functions such as the project and firmware update or OPC XML.
- FTP and Telnet access are also possible.
- New users can be entered in the empty user database.

Security Level Normal

The controller has a user database. A project exists on the controller and HTTP, HTTPS, FTP, and Telnet are activated in the hardware configuration.

- User password authentication is required for access to websites with sensitive content (e.g. firmware update, watch table, etc.), FTP and Telnet.

As soon as a project has been loaded to the controller, Security Level Normal is active, with an empty user database as required. Standard websites are still visible. All other websites can only be accessed with the necessary authentication.

Security Level High

High security with maximum access protection:

- HTTP, HTTPS, FTP and Telnet have been deactivated via the project in the hardware configuration. Access to the Ethernet via the various ports of the services is then no longer possible. The Web server cannot be used.

User management

SIMOTION IT uses a user database to safeguard access to a device. The groups are stored in the user database along with their assigned users. The defined user groups can be assigned access rights to the individual Web server websites. The Web server is accessed after the authentication.

Authentication

- There are users (USERs).
- Each user has a password. This is encrypted.
- Users belong to groups (GROUP).
- Websites, directories, and applications are protected by secure areas defined for each group.
- Only users that belong to the secure area can access the protected website.
- Each secure area has a group of users who have access authorization.
- A user can belong to different groups.

Encrypted data transfer (HTTPS)

The Web server can be accessed via an HTTP as well as an HTTPS connection. The Secure Socket Layer protocol (SSL) in HTTPS enables encrypted data transmission between a client (browser) and the SIMOTION controller (Web server). Secure transmission can be forced by deactivation of the HTTP port for security reasons.

Certificates must be generated and installed for encrypted communication between the browser and the Web server. A device comes supplied with a standard root certificate and a private key of the Web server as a file. These files should be replaced with your own to increase the security of HTTPS access to the device.

Key files

- Delivery state
  In order for you to be able to access the SIMOTION controller via the SIMOTION IT diagnostics standard websites (in their delivery state) via HTTPS, a root certificate and a private key are supplied as a file on the device.
- Creating the SSL certificate yourself
  With the help of the Perl Tool and the supplied Perl Script (cert.pl), the certificates required for customer facilities (sites) can be generated and tied together to form loading packages.

There are two ways of acquiring your own server certificate (SSL certificate):

- Create a root certificate (self-signed) and a private key using a certificate software.
- Purchase a server certificate from a certificate authority.

Importing the SSL certificate into the browser

If you use SSL with your own certification authority, you will need to prepare your PCs for communication with the SIMOTION controller. To do this, the root certificate must be added to the list of certificates in your browser.

Further information

Further information on the SIMOTION IT web server can be found in the following documentation:

- SIMOTION IT Diagnostics and Configuration Diagnostics Manual (https://support.industry.siemens.com/cs/de/en/view/109767636)
- SIMOTION IT OPC UA Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767638)
- AUTOHOTSPOT
- SIMOTION IT Virtual Machine and Servlets Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767639)

See also

SIMOTION IT Programming and Web Services Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767637)

OPC UA server

Introduction

SIMOTION has implemented an OPC UA server with DA (Data Access).

OPC UA binary encoding is supported. Access to an arbitrary OPC UA client can be protected via authentication and encrypted data transfer.

Configuration

Note

Before connecting to the OPC UA server, ensure that the environment is secure and install a hardware-based intermediate layer (e.g. DMZ network, firewall, SCALANCE S modules, etc.).

The OPC UA server can be activated or deactivated via HW Config from TIA Portal or STEP 7.

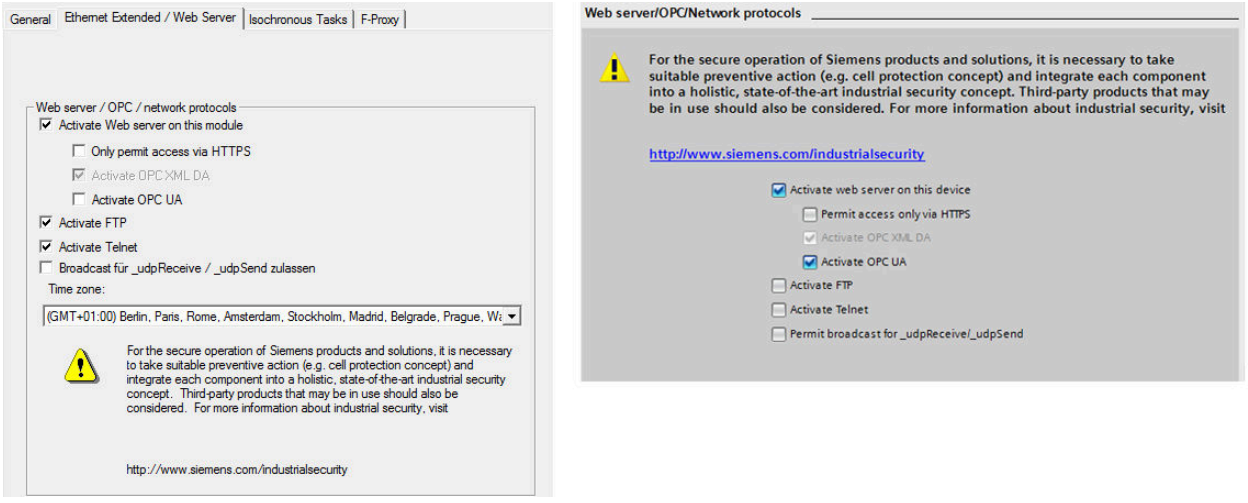


Figure 4-949 Activating the SIMOTION IT Web server functions in SIMOTION SCOUT or SIMOTION SCOUT TIA

Further settings are made via the SIMOTION IT Web server configuration masks:

- Enabling of the Ethernet interface and associated port of SIMOTION for the OPC UA access.
- Definition of the user name, password and user group as part of the user administration of the SIMOTION IT Web server.
- Handling of the certificates for the encryption of the data transfer.

Further information

Further information on the OPC UA server can be found in the SIMOTION IT OPC UA Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767638).

Disposal

NOTICE
Data misuse Unsafe disposal of the storage media (CF card/CFast/SSD) can lead to misuse of the data of the part programs, archives, etc. by third parties. • Therefore, ensure that the data on the storage media that is used is securely deleted before disposing of the product. Use programs that support you in securely deleting/formatting storage media.

4.9.7.4 SINAMICS

The following chapter provides you with an overview of the Industrial Security features available for SINAMICS to protect your converters from threats. In the following, you will find topics which you should pay special attention to regarding Industrial Security:

- Write and know-how protection
- Parameters: Access levels
- Using the memory card
- Note on Safety Integrated
- Communication services and used port numbers
- Web server
- Information about individual interfaces
- SINAMICS Startdrive and STARTER
- SINAMICS Drive Control Chart (DCC)

Detailed descriptions and procedures can be found in the corresponding SINAMICS documentation.

Many products (SINUMERIK, SIMOTION, SINAMICS) contain OpenSSL. The following applies to these products:

- This product contains software (https://www.openssl.org/) that has been developed by the OpenSSL project for use in the OpenSSL toolkit.
- This product contains cryptographic software (mailto:eay@cryptsoft.com) created by Eric Young.
- This product contains software (mailto:eay@cryptsoft.com) developed by Eric Young.

Network security

Note

SINAMICS products may only be used in a secure and trusted network. Observe the information on this topic in Chapter "Network segmentation (Page 3224)".

Know-how protection

Some SINAMICS converters provide you with a "Know-how protection" function: This function offers you protection of your intellectual property, especially the know-how of machine manufacturers against unauthorized use, modification or reproduction of their products.

Effect

Adjustable parameters which are not recorded in an exception list can neither be read nor written.

Exceptions

- The know-how protection does not affect parameters that are provided with the following attributes:
  - KHP_WRITE_NO_LOCK
    - These parameters are excepted from the know-how protection and can therefore be written to despite the know-how protection.
    - For a list of these parameters, see the List Manual of the respective product.
    - These parameters are not included in the exception list.
  - KHP_ACTIVE_READ
    - These parameters can also be read, but not written, with activated know-how protection.
    - For a list of these parameters, see the List Manual of the respective product.
    - These parameters are not included in the exception list.
- Know-how protection does not prevent the execution of certain functions:
  - In particular, the "Restore factory settings" function is still possible despite know-how protection.
  - For a full list of executable functions, please refer to the following references.

Further information

For more information on this topic, see the following references:

- SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109763287)
  Chapter "Know-how protection"
- SINAMICS G110M operating instructions (https://support.industry.siemens.com/cs/de/en/view/109757594)
  Chapter "Know-how protection"
- SINAMICS G120 Operating Instructions
  Chapter "Know-how protection"

- SINAMICS S and SINAMICS G List Manuals
  Section "Parameters for write protection and know-how protection"
- SINAMICS G130, G150 and S150 Operating Instructions
  Chapter "Know-how protection"

Parameters: Access levels and password

For the G110M, G120, G130, G150, S110, S120 and S150 series devices, the SINAMICS parameters are divided into the access levels 0 to 4. With the aid of the access levels, you can specify which parameters can be modified by which user or input/output device:

- With the aid of parameter p0003, you can specify which access levels you can select with the BOP or IOP.
- Parameters of access level 4 are password-protected and only visible for experts up to SINAMICS RT V4.9.

The SINAMICS S and SINAMICS G List Manuals specify in which access level the parameter can be displayed and changed.

Further information

For detailed information on this topic, see the following references:

- SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771805)
  Chapter "Parameters"
- SINAMICS S120 Safety Integrated Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771806)
  Section "Handling the safety password"
- SINAMICS G110M List Manual
  Chapter "Overview of parameters"
- SINAMICS G120 Operating Instructions
  Chapter "Parameters"
- SINAMICS S and SINAMICS G List Manuals
  Section "Explanation of the list of parameters"
- SINAMICS G130, G150 and S150 Operating Instructions
  Chapter "Parameters"

Using the memory card

The memory card must be handled with particular care for all SINAMICS devices that use a memory card so that no malicious software or erroneous parameterizations are spread between different commissioning PCs or inverters.

⚠ WARNING

Risk of death due to software manipulation when using exchangeable storage media

Storing files onto exchangeable storage media amounts to an increased risk of infection of the commissioning PCs, e.g. with viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect files stored on exchangeable storage media from malicious software using appropriate protection measures, e.g. virus scanners.

⚠ WARNING

Risk of death due to software manipulation when using exchangeable storage media

Storing the parameterization (incl. Safety Integrated parameterization) on exchangeable storage media carries the risk that the original parameterization (with Safety Integrated) will be overwritten, for example, by the memory card of another drive without Safety Integrated. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Ensure that only the memory card that belongs to the respective inverter is used.
- Ensure that only trained or authorized personnel have access to the enclosures, cabinets or electrical equipment rooms.

Safety Integrated

To actually reduce the risk for machines and plants through the use of Safety Integrated functions, working with Safety Integrated functions requires special care for all SINAMICS devices that have it.

⚠ DANGER

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety instructions in the Safety Integrated manuals.

Communication services and used port numbers

SINAMICS converters support specific communication protocols. The address parameters, the relevant communication layer, as well as the communication role and the communication direction are decisive for each protocol. You require this information to match the security measures for the protection of the automation system to the used protocols (e.g. firewall). Some of the security measures described here are restricted to Ethernet and PROFINET networks.

For detailed information on this topic, see the following references:

- SINAMICS S120
  - Starting from firmware Version 5.2
    SINAMICS S120 Communication Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109771803)
  - Older firmware versions
    SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771805)
  - Including: Section "Communication services and used port numbers"
- SINAMICS G Function Manual Fieldbuses (https://support.industry.siemens.com/cs/ww/de/view/109757336/en)
  Section "Ethernet and PROFINET protocols that are used"
- SINAMICS G130, G150 and S150 Operating Instructions
  Chapter "Communication services and used port numbers"

Integrated web server

The SINAMICS web server provides information on a SINAMICS device via its websites. This is accessed via an Internet browser.

Data transfer

In addition to the normal (unsecured) transmission (http), the Web server also supports secure transmission (HTTPS). Secure transmission (HTTPS) is the recommended setting.

Note

Smart Access Module

The web server of the Smart Access Module does not support secure transmission (HTTPS). Alternatively, you can use an encrypted WLAN transmission.

By entering "HTTP://" or "HTTPS://" in front of the address of the drive, you can decide yourself whether normal or secure transmission is used to access the data.

For safety reasons, secure transmission can be forced by deactivation of the http port.

Access rights

The normal protection mechanisms of SINAMICS also apply for access via the web server, including password protection. Further protective mechanisms have been implemented especially for the Web server. Different access options have been set for different users, depending on the function. The parameter lists are protected so that only users with the appropriate rights can access or change the data.

Further information

For detailed information on this topic (e.g. the supported Internet browsers), see the following references:

- SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109763287)
  Section "Web server"
- SINAMICS S210 Operating Instructions (https://support.industry.siemens.com/cs/ww/en/view/109771824)
- SINAMICS V20 Operating Instructions (https://support.industry.siemens.com/cs/de/de/view/109768394/en)
- SINAMICS G120 Smart Access Operating Instructions (https://support.industry.siemens.com/cs/ww/en/view/109771299)
- SINAMICS G130, G150 and S150 Operating Instructions

Certificates for the secure data transfer

Protecting the HTTPS access

The "Transport Layer Security" (TLS) protocol enables encrypted data transfer between a client and the SINAMICS drive. HTTPS access between the browser and the drive is based on Transport Layer Security.

The encrypted variant of communication between the browser and the Web server using HTTPS requires the creation and installation of certificates (default configuration, self-created certificates or server certificates from a certification authority).

TLS

Transport Layer Security (TLS V1.2 or higher) is a hybrid encryption protocol for secure transfer of data in the Internet.

Key files

You need 2 key files (a public certificate and a private key) for the encryption method used by the Transport Layer Security.

Certificate handling

The necessary certificate and key is generated on the drive so that you can access the drive via HTTPS in the SINAMICS as delivered. For this reason, the firmware certificate should only be used in secure networks (e.g. PROFINET below a PLC) or for direct point-to-point connections on the service interface X127.

Instead, use a certificate confirmed by an external certification center. The references cited in the following contain a detailed description of the procedure.

Further information

For detailed information on this topic, see the following references:

- SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771805)
  Section "Certificates for the secure data transfer"
- SINAMICS G130, G150 and S150 Operating Instructions

Information about individual interfaces

X127 LAN (Ethernet)

Ethernet interface X127 is intended for commissioning and diagnostics, which means that it must always be accessible. Note the following restrictions for the X127 interface:

- Only local access is possible
- No networking or only local networking in a locked control cabinet permissible

If you require remote access to the control cabinet, then you must apply additional security measures so that misuse through sabotage, data manipulation by unqualified persons and intercepting confidential data are completely ruled out.

X140 serial interface (RS232)

You connect an external HMI device for operator control/parameter assignment via the serial interface X140.

NOTICE
Access to the inverters only for authorized personnel Unauthorized persons may be able to damage or alter production equipment as a result of gaps in a company's physical security. Confidential information can also be lost or altered as a result of this. You can prevent this if you protect the company site and the production areas accordingly. • You can find information on suitable protective measures in Section "Physical protection of critical production areas (Page 3222)".

X150 LAN (Ethernet)

The network with which interface X150 is connected must be separated from the rest of the plant network in accordance with the Defense in Depth concept (see Chapter "General security measures (Page 3220)"). Access to cables and possibly open connections must be implemented in a protected fashion, as in a control cabinet.

X1400 LAN (Ethernet)

The network with which interface X1400 is connected must be separated from the rest of the plant network in accordance with the Defense in Depth concept (see Chapter "General security measures (Page 3220)"). Access to cables and possibly open connections must be implemented in a protected fashion, as in a control cabinet.

Further information

For detailed information on this topic, see the following references:

- SINAMICS S120
  - SINAMICS S120 Control Units and Additional System Components Equipment Manual (https://support.industry.siemens.com/cs/de/en/view/109763286)
    Section on the respective interfaces
  - Starting from firmware Version 5.2: SINAMICS S120 Communication Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109771803)
  - Older firmware versions:
    SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109763287)
- SINAMICS G and SINAMICS S Operating Instructions
- SINAMICS V90 Operating Instructions

Disposal

NOTICE

Data misuse resulting from unsafe disposal of the product

Unsafe disposal of the product can lead to misuse of the parameter data by third parties.

- Therefore, before disposal, restore all the parameters to the factory settings.

You can find further information on restoring the factory settings in the Function Manual or Operating Instructions of the respective product.

NOTICE

Data misuse resulting from unsafe disposal of the memory card

Unsafe disposal of the memory card can lead to misuse of the data etc. by third parties. Among other things, the data backups required for operating the converter are located on the memory card.

- Therefore, ensure that the data on the memory card is securely deleted before disposing of the product. There are programs that support you in securely deleting/formatting the memory card.
- This concerns all products that have a memory card.

Note

Deleting user-defined certificates

Make sure you securely remove all user-defined certificates before disposing of a SINAMICS product. A hacker can use your certificates to gain access to your protected data transmission.

- Products with memory card
  - Delete the files SINAMICS.key and SINAMICS.crt from the directory OEM\SINAMICS\WEB\WEBCONF\CERT on the memory card.
- Products with optional memory card (e.g. SINAMICS S210)
  - Create empty files ("SINAMICS.key" and "SINAMICS.crt") with the corresponding file names.
  - Copy these files to the memory card.
  - Insert the memory card into the converter.
  - Restart the converter.
  - Alternatively, when you no longer need data from the memory card: reformat the memory card.

You can find further information in the Function Manual or Operating Instructions of the respective product.

SINAMICS Startdrive and TIA Portal

Malfunctions of the machine as a result of incorrect or changed parameterization

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameter settings against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

SINAMICS Startdrive

Startdrive in the TIA Portal

SINAMICS Startdrive is an option package in the TIA Portal with which SINAMICS drives are commissioned. With regard to Industrial Security, you must consider the corresponding specifications for SINAMICS drives and for the TIA Portal.

In addition to the commissioning of single drives, you can also use Startdrive to configure drives on SIMATIC control systems such as the S7-1500. Information on how to proceed with SIMATIC controllers can be found in the TIA Portal online help at "Configuring networks".

Commissioning computer

Ensure the security of the commissioning computer. Follow the general security measures (Page 3220) for this purpose.

Device know-how protection

- You can protect the parameterization of your drive from unauthorized access via the "know-how protection" function.
- The function is only available online.
- Device know-how protection is supported as of Startdrive V16.

Security functions

- Activation/deactivation of unused functions (web server, ports)
- Write protection for the parameter assignment, p-parameters are readable, but not writeable, protects against unintentional changes to the parameter assignment (only available online).

Protecting backup files in the Windows file system

If you create backup files of charts or projects with Windows tools, also protect these files with Windows tools against unauthorized access using secure passwords. The Startdrive project itself is protected for integrity.

Scripting (Openness)

Scripts (Openness) are used for automating sequences in Startdrive. You must therefore test the scripts before using them on machines.

⚠ WARNING

Risk due to incorrect configurations for automated operating actions

Scripting provides the extensive automation options that are required to be able to automate manual operator actions in the Startdrive tools and therefore to optimize the time required for the recurring configuration of projects and tasks.

The script programmer and the script user are responsible for the operator actions implemented in scripting.

Incorrect configurations that are not discovered in tests can result in serious physical injury or death.

- Run systematic tests on new and modified scripts to verify and validate them.
- Before running a script, make sure it has the correct content. Verify and validate the results of script execution by tests on the machine.

As for DCCs, scripts can also be protected via know-how protection.

SINAMICS STARTER

Commissioning drives with STARTER

Drives of the MICROMASTER and SINAMICS families can be commissioned with STARTER. An integrated version of STARTER is contained in SIMOTION SCOUT. For information on SIMOTION SCOUT, see "SIMOTION (Page 3257)".

Commissioning computer

Ensure the security of the commissioning computer. Follow the general security measures (Page 3220) for this purpose.

Protecting backup files in the Windows file system

If you create backup files of charts or projects with Windows tools, also protect these files with Windows tools against unauthorized access using secure passwords.

Security functions

- Know-how protection for the parameter assignment, scripts and DCCs and DCC libraries with password and encryption
- Copy protection for the configuration on the drive unit. The project can only be opened together with the original card.
- Detection of parameter manipulation with STARTER via the project comparison, see also "Offline/online comparison (Page 3264)"
- Activation/deactivation of unused functions (web server, ports), see also "Integrated web server (Page 3274)"
- Write protection for the parameter assignment, p-parameters are readable, but not writeable, protects against unintentional changes to the parameter assignment (only available online).

Know-how protection for drive units

In addition to the know-how protection for DCCs, DCC libraries and scripts, you can also protect the parameter assignment of your drive against unauthorized access via the know-how protection for the drive. The function is only available online. See also "Know-how protection (Page 3271)".

Scripting

Scripts are used for automated execution in STARTER. You must therefore test the scripts before using them on machines.

⚠ WARNING

Risk due to incorrect configurations for automated operating actions

Scripting provides the extensive automation options that are required to be able to automate manual operator actions in the STARTER/SCOUT tools and therefore to optimize the time required for the recurring configuration of projects and tasks.

The script programmer and the script user are responsible for the operator actions implemented in scripting.

Incorrect configurations that are not discovered in tests can result in serious physical injury or death.

- Run systematic tests on new and modified scripts to verify and validate them.
- Before running a script, make sure it has the correct content. Verify and validate the results of script execution by tests on the machine.

As for DCC charts, scripts can also be protected via know-how protection.

SINAMICS Drive Control Chart (DCC)

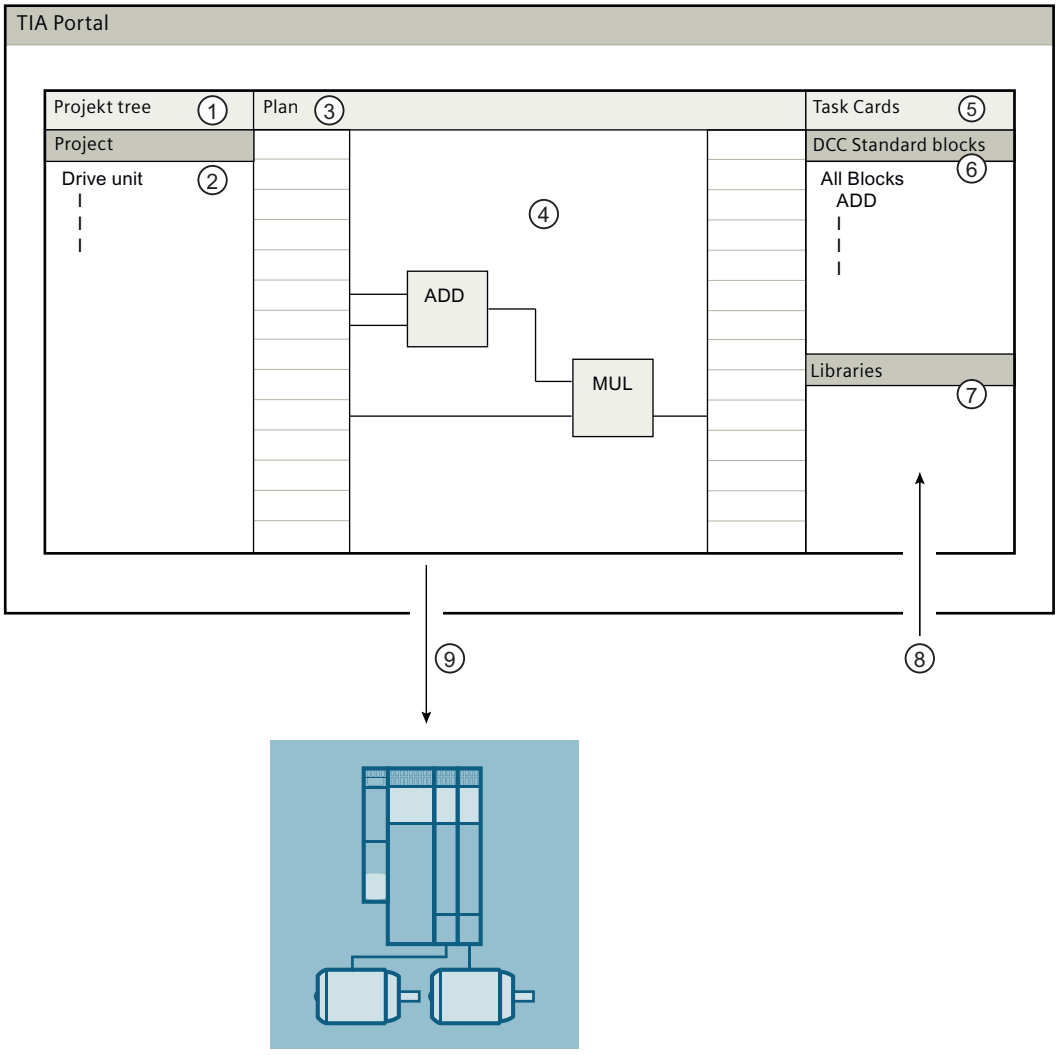
Industrial security with SINAMICS DCC

Overview

SINAMICS Drive Control Chart (DCC) offers a modular, scalable technology option, which has chiefly been developed for drive-related, continuous open-loop and closed-loop control engineering tasks within the drive.

With the Drive Control Chart Editor based on CFC, you configure the technology functions with DCC for SINAMICS drives graphically.

- Startdrive
  The following figure shows the data flow of the configuration data when configuring with SINAMICS DCC:



① Loading
② Import of DCB libraries

Figure 4-950 Flow of configuration data: TIA-DCC

- STARTER
  The following figure illustrates the data flow of the configuration data when configuring with SINAMICS DCC and the ways to protect the configured/programmed DCC sources:

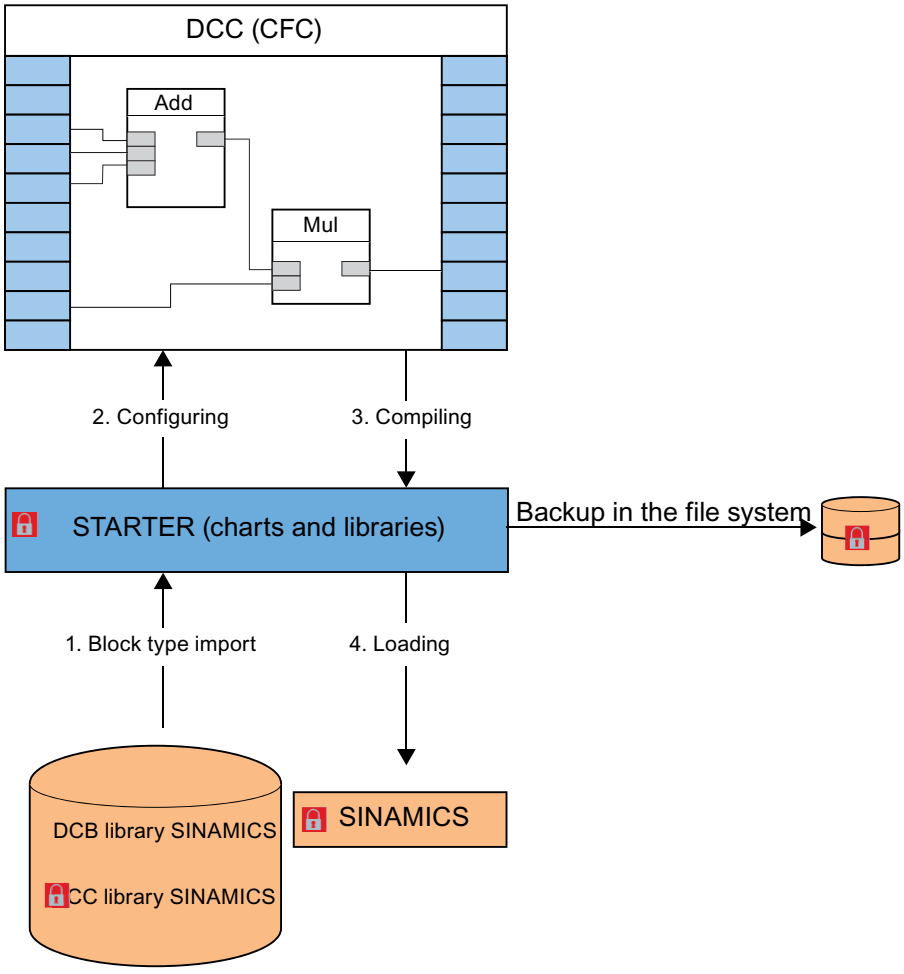


Figure 4-951 Flow of configuration data: Example for DCC Classic V2.1 ... V3.4 (STARTER)

Startdrive: Option package DCC

Note the following special features regarding Startdrive:

- DCC does not provide a backup in the file system.
- The DCB libraries used are loaded implicitly with the project into the target device
- DCC does not offer any chart know-how protection

Commissioning computer

Ensure the security of the commissioning computer. Follow the general security measures (Page 3220) for this purpose.

Using know-how protection

DCCs, DCC libraries, programs and backup files are subject to an increased risk of manipulation. Therefore, use the know-how protection, the write protection for drive units and the know-how protection for DCC charts and DCC libraries in STARTER, see also Use write and know-how protection (Page 3285).

Information on know-how protection can also be found in the "Motion Control SINAMICS/ SIMOTION Editor Description DCC" Programming and Operating Manual.

Protecting backup files in the Windows file system

If you create backup files of charts or projects with Windows tools, also protect these files with Windows tools against unauthorized access using secure passwords.

Note the information on SINAMICS and on the engineering systems

Also note the Industrial Security information for SINAMICS drives and engineering systems with which SINAMICS drives are commissioned. Particularly the information on network security is important, see also Network security (Page 3223).

Use write and know-how protection

Prevent unauthorized changes by means of know-how protection

⚠ WARNING

Danger to life through manipulation of DCC charts and DCC libraries

The use of unprotected DCCs and DCC libraries entails a higher risk of manipulation of DCCs, DCC libraries and backup files.

- Protect important DCCs and DCC libraries via "Know-how protection programs" or "Know-how protection drive units" in SCOUT or STARTER. Assign a strong password to prevent manipulation.
- Therefore, for "Know-how protection programs" or "Know-how protection drive units", use passwords which include at least eight characters, upper and lower case letters, numbers, and special characters.
- Make sure that only authorized personnel can access the passwords.
- Protect the backup files on your file system using a write protection.

SINAMICS Smart Access Module

The optional Smart Access Module offers you an intelligent solution for commissioning the SINAMICS V20 or G120 converter.

The Smart Access Module is a web server module with integrated WLAN connectivity. It allows web-based access to the converter from a connected device (conventional PC with WLAN

adapter, tablet or smartphone). This module is only intended for commissioning and therefore cannot be used with the converter for the long term.

NOTICE
WLAN: Changing a default password The misuse of passwords can also represent a considerable security risk. As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death. • After logging on to the Smart Access Module for the first time, change the module's default password. • Assign a secure password. Information on this can be found in the Operating Instructions of the respective converter.

NOTICE
Unauthorized access to the converter via the SINAMICS Smart Access Module Unauthorized access to the converter via the Smart Access Module as a result of cyber attacks could lead to interruptions in the process and thus to property damage or personal injury. • Before you log in to the web pages, check the status LED on the Smart Access Module. If the status LED is green or flashing, unauthorized access could have occurred: Switch the SINAMICS Smart Access Module off and then on again using the on-off switch to re-establish the WLAN connection.

For detailed information on this topic, see the following references:

- SINAMICS V20 Converter Operating Instructions (https://support.industry.siemens.com/cs/de/en/view/109773836)
- SINAMICS G120 Smart Access Operating Instructions (https://support.industry.siemens.com/cs/ww/en/view/109771299)

4.9.7.5 SIMOCRANE

Availability, productivity, and safety are the decisive factors in crane applications. Since SIMOCRANE products are based on products from SIMOTION and SINAMICS, observe the product-specific hardening measures in Chapters SIMOTION (Page 3257) and SINAMICS (Page 3270) for hardening.

4.9.8 References

General information

Additional **general information** about Industrial Security is available on the Internet:

- Industrial security (https://www.siemens.com/industrialsecurity)
- Cyber security (https://new.siemens.com/global/en/company/topic-areas/cybersecurity.html)
- Planning security (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/planning.html)
- Implementing security (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/implementation.html)
- Always active (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/always-active.html)
- Certifications and standards (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/certification-standards.html)
- Whitepapers and downloads (https://new.siemens.com/global/en/products/automation/topic-areas/industrial-security/downloads.html)

Secure passwords

Further information on assigning secure passwords can be found in the chapter under the following addresses:

- Federal Agency for Security in Information Technology (BSI) (https://support.industry.siemens.com/cs/de/en/view/109757594)(in German only)
- European Network and Information Security Agency (enisa). (https://www.enisa.europa.eu/media/news-items/basic-security-practices-regarding-passwords-and-online-identities)
- National Institute of Standards and Technology (NIST) (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf)

Web pages of the products

Additional product-specific information about Industrial Security is available on the **individual product websites**:

- SINUMERIK: SINUMERIK homepage (https://www.siemens.com/sinumerik)
- SIMOTION: SIMOTION homepage (https://www.siemens.com/simotion)
- SINAMICS: SINAMICS homepage (https://www.siemens.com/sinamics)

Product-specific manuals

Product-specific manuals for the individual products can be found on the Internet:

- "SINUMERIK 840D sl NCU 7x0.3 PN" Manual (https://support.industry.siemens.com/cs/de/en/view/99922219)
- "Commissioning CNC: NC, PLC, Drive" Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109777906)
- "SINUMERIK 840D sl Operating System NCU (IM7)" Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109763236)
- "SINUMERIK 840D sl Base Software and HMI sl" Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109254363)
- SINUMERIK 840Dsl Safety Integrated plus Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109777982)
- "SINUMERIK Operate (IM9)" Commissioning Manual (https://support.industry.siemens.com/cs/ww/en/view/109769186)
- PCU-Basesoftware (IM8) Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109748542)
- "PCU Base Software (IM10)" Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109769185)
- SINUMERIK Access MyMachine /P2P (PC) Operating Manual (https://support.industry.siemens.com/cs/ww/en/view/109770206)
- SINUMERIK Access MyMachine /OPC UA Configuration Manual (https://support.industry.siemens.com/cs/de/en/view/109777871)
- Manage MyMachines /Remote Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109759394)
- Diagnostics Manual 808D (https://support.industry.siemens.com/cs/de/en/view/109763685)
- SIMOTION IT Programming and Web Services Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767637)
- SIMOTION IT Diagnostics and Configuration Diagnostics Manual (https://support.industry.siemens.com/cs/de/en/view/109767636)
- SIMOTION IT Virtual Machine and Servlets Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767639)
- SIMOTION IT OPC UA Programming Manual (https://support.industry.siemens.com/cs/de/en/view/109767638)
- BA_G110M (https://support.industry.siemens.com/cs/de/en/view/109757594)
- Communication with SIMOTION System Manual (https://support.industry.siemens.com/cs/de/en/view/109767623)
- SINAMICS S120 Drive Functions Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771805)
- SINAMICS S120 Control Units and Additional System Components Equipment Manual (https://support.industry.siemens.com/cs/de/en/view/109771804)

- SINAMICS S120 Safety Integrated Function Manual (https://support.industry.siemens.com/cs/de/en/view/109771806)
- SINAMICS S120 Communication Function Manual (https://support.industry.siemens.com/cs/ww/en/view/109771803)
- SINAMICS S120/S150 List Manual (https://support.industry.siemens.com/cs/de/en/view/109763271)
- G110M Operating Instructions (https://support.industry.siemens.com/cs/de/en/view/109757594)
- G120 Operating Instructions (https://support.industry.siemens.com/cs/ww/en/view/109771299)
- SINAMICS G Fieldbuses Function Manual (https://support.industry.siemens.com/cs/ww/de/view/109757336/en)
- S210 Operating Instructions (https://support.industry.siemens.com/cs/ww/de/view/109771824)
- V20 Operating Instructions (https://support.industry.siemens.com/cs/de/en/view/109773836)
- Function Manual S7-1500, ET 200SP, ET200pro web server (https://support.industry.siemens.com/cs/ww/en/view/59193560)
- SIMATIC Programming with STEP 7 Programming and Operating Manual (https://support.industry.siemens.com/cs/de/en/view/109751825)
- SIMATIC S7-300 CPU 31xC and CPU 31x Equipment Manual: Technical specifications (https://support.industry.siemens.com/cs/de/en/view/12996906)
- SIMATIC NET Configuration Manual: SCALANCE X-200 Industrial Ethernet switches (https://support.industry.siemens.com/cs/de/en/view/109757352)

See also

Implement security (https://www.industry.siemens.com/services/global/en/portfolio/plant-data-services/industrial_security)

Safety Integrated plus Commissioning Manual (https://support.industry.siemens.com/cs/de/en/view/109763246)

Recommendations for creating secure passwords provided by BSI [German Federal Office for IT Security] (https://www.bsi-fuer-buerger.de/BSIFB/DE/Empfehlungen/Passwoerter/passwoerter_node.html)

Service and Diagnostics

5

5.1 Task Trace

5.1.1 Preface

5.1.1.1 SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

5.1.1.2 Hotline and Internet addresses

Additional information

Click the following link to find information on the the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following e-mail address:
docu.motioncontrol@siemens.com

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

5.1.2 Overview

Field of application

The **SIMOTION Task Trace** supports you when troubleshooting in the SIMOTION multitasking environment. The SIMOTION Task Trace records the sequence of individual tasks, identifies user events that you can generate via a program command, and displays these graphically. This makes the interaction between programs and tasks easily traceable, thereby affording better control over them.

Structure of the Task Trace

The SIMOTION Task Trace includes two main components:

- The **SIMOTION Task Tracer**, which writes the task change and events to a buffer on the target device
- The **SIMOTION Task Profiler**, an application for displaying the recorded data

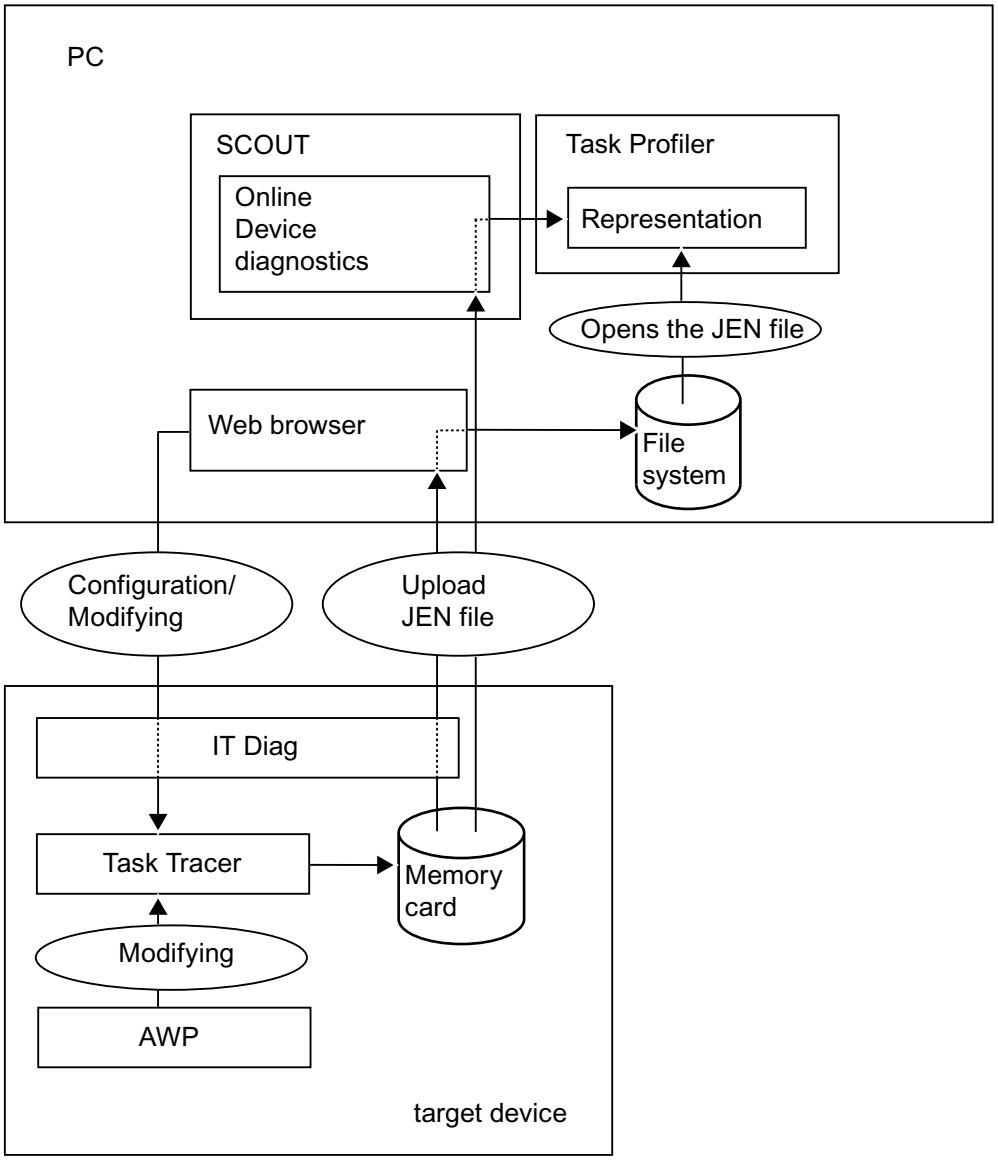


Figure 5-1 Schematic representation of SIMOTION Task Trace

SIMOTION IT DIAG (standard web pages) is used to configure the SIMOTION Task Tracer. The recording data is saved as a JEN file on the memory card of the target device.

If you start the Task Profiler via the device diagnostics, the trace data is automatically read from the memory card via the online connection and displayed. Alternatively, you can also save the JEN file from the target device to the file system of the PC using the SIMOTION standard web pages and then open the file in the SIMOTION Task Profiler.

5.1.3 Configuring

Overview

Configuring the SIMOTION Task Tracer is only necessary in exceptional circumstances, as the default configuration is used for recording. If you require a different configuration than the default, with Version V4.1.2 and higher this can be carried out on the target device via the standard web pages (SIMOTION IT DIAG).

Note

The functionality must be activated in the SIMOTION SCOUT project in the hardware configuration of the CPU. You can activate the required services in the hardware configuration on the "Ethernet extended/Webserver" tab in the object properties of the CPU.

Note

The changes you make to the configuration using the standard web pages remain in effect until the next time the target device is powered up. Following power-up, the default settings and the settings in the SIMOTION.ini file apply.

SIMOTION IT DIAG (standard web pages)

Configuration and control functionality of the Task Tracer are available on the standard web pages of SIMOTION IT DIAG.

Use the **Open IT Diag** button to start the standard web pages in the SCOUT device diagnostics.

Note

If there is no Ethernet connection from SCOUT to the target device, the button is deactivated (grayed out).

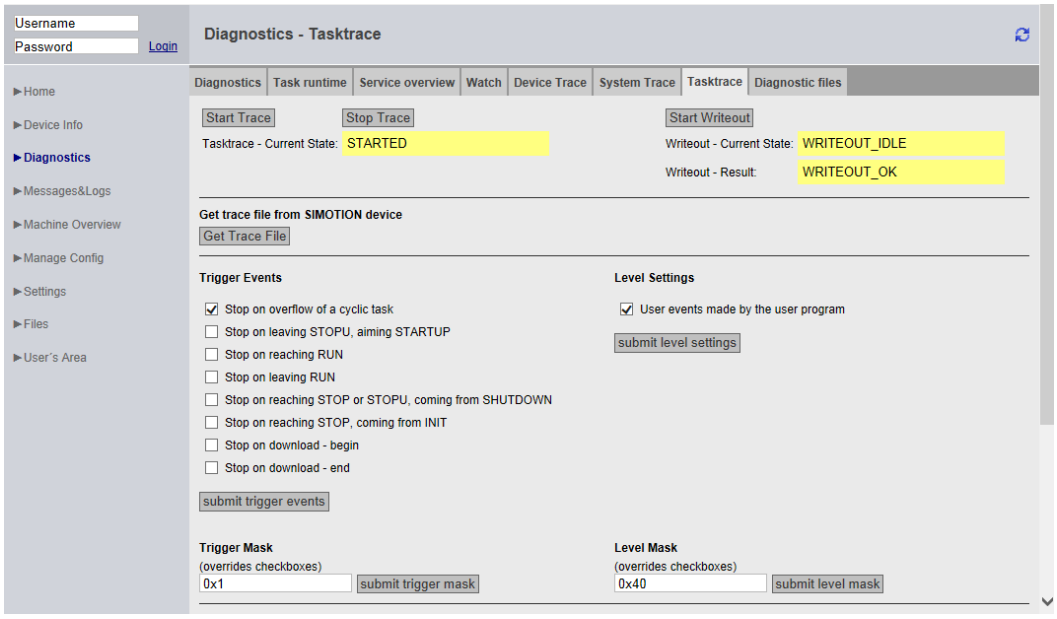


Figure 5-2 SIMOTION IT DIAG standard web pages - Task Trace (part 1)

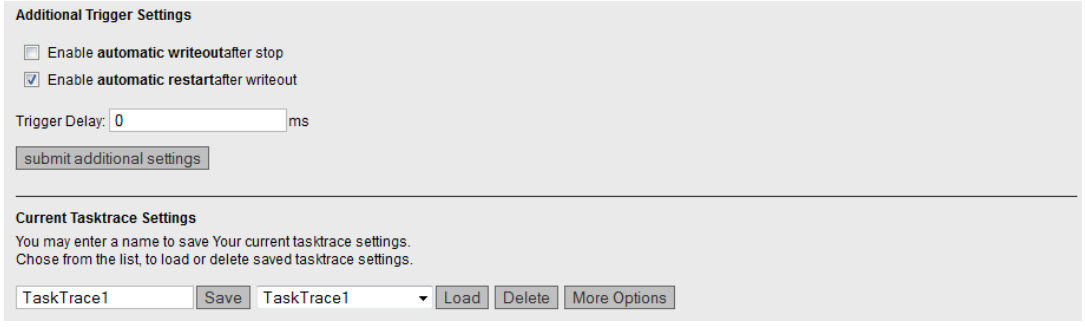


Figure 5-3 SIMOTION IT DIAG standard web pages - Task Trace (part 1)

Default configuration

The following table summarizes the default configuration of the Task Tracer and the setting options (configuration and control) using the standard web pages of the target device or the SIMOTION.ini file.

Note

The SIMOTION.ini is not created automatically and may need to be created and stored in the root directory of the memory card.

Note

For additional information regarding the configuration and status displays using standard web pages, refer to the "SIMOTION SCOUT Overview about Service and Diagnostic Options" documentation.

Table 5-1 Default configuration of the Task Tracer

Variable name in SIMOTION.ini	Meaning	Default setting	Can be changed via IT-Diag standard web pages
TT_BUFSIZE	Size of the trace buffer in KB	512,000 bytes (the maximum setting depends on the available RAM)	No
TT_WROUT	Activate automatic writing of the trace data to the memory card after the trace has stopped TT_WROUT =1	The trace data is not written automatically to the memory card (Because this is a default setting, no entry in the SIMOTION.ini)	Additional Trigger Settings: Enable **automatic writeout** after triggered stop
TT_RSTRTOFF	Trace is not started automatically after the trace data is written to the memory card TT_RSTRTOFF=1	The trace is started automatically once the trace data has been written to the memory card (Because this is a default setting, no entry in the SIMOTION.ini)	Additional Trigger Settings: Enable **automatic restart** after writeout
TT_LEVEL	Mask for specifying the events that are to be recorded in the Task Trace (see "Masking of events to be recorded") TT_LEVEL=0x0040	0x0040 (User_Events)	Level Settings
TT_TRIGGER	Mask for specifying the stop trigger of the trace (see "Masking of STOP trigger") TT_TRIGGER=0x0001	0x0001	Trigger Events or Trigger Mask
TT_TRGDELAY	Delay time [ms] after the trigger event until Task Trace is stopped	0 ms	Trigger Delay

Masking of events to be recorded

```
Level masks:
0x0040 user events made by the user program
```

Masking of STOP trigger

```
Trigger masks:
0x0001 stop on overflow of a cyclic task
0x0002 stop on leaving BZ STOPU, aiming STARTUP
0x0004 stop on reaching BZ RUN
0x0008 stop on leaving BZ RUN
0x0010 stop on reaching BZ STOP or STOPU, coming from SHUTDOWN
```

0x0020 stop on reaching BZ STOP, coming from INIT

0x0040 stop on leaving BZ INIT

0x0100 stop on Download-Begin

0x0200 stop on Download-End

You can also combine masking instances if required, for example 0x0009 from

0X0001 stop on overflow of a cyclic task and 0x0008 stop on leaving BZ RUN.

Creating the SIMOTION.ini file

The following example shows how you can modify the configuration using the SIMOTION.ini file. In this process, it should be specified that the CPU mode will be recorded until it changes to STOP mode. The buffer size is specified as 1,000,000 bytes. The automatic writing of the trace data to the memory card is activated once the trace has stopped.

Example:

TT_BUFSIZE = 1000000

TT_TRIGGER= 0x0010

TT_WROUT=1

Note

The maximum buffer size depends on the available memory capacity of the RAM and the memory card.

5.1.4 Working with the SIMOTION Task Profiler

5.1.4.1 Starting the SIMOTION Task Profiler

You can open the SIMOTION Task Profiler

- as an application from the STEP7 program folder or
- from the SCOUT device diagnostics.

Start via STEP7 program folder

You can open the SIMOTION Task Profiler as an application in the STEP 7 program folder (see figure below).

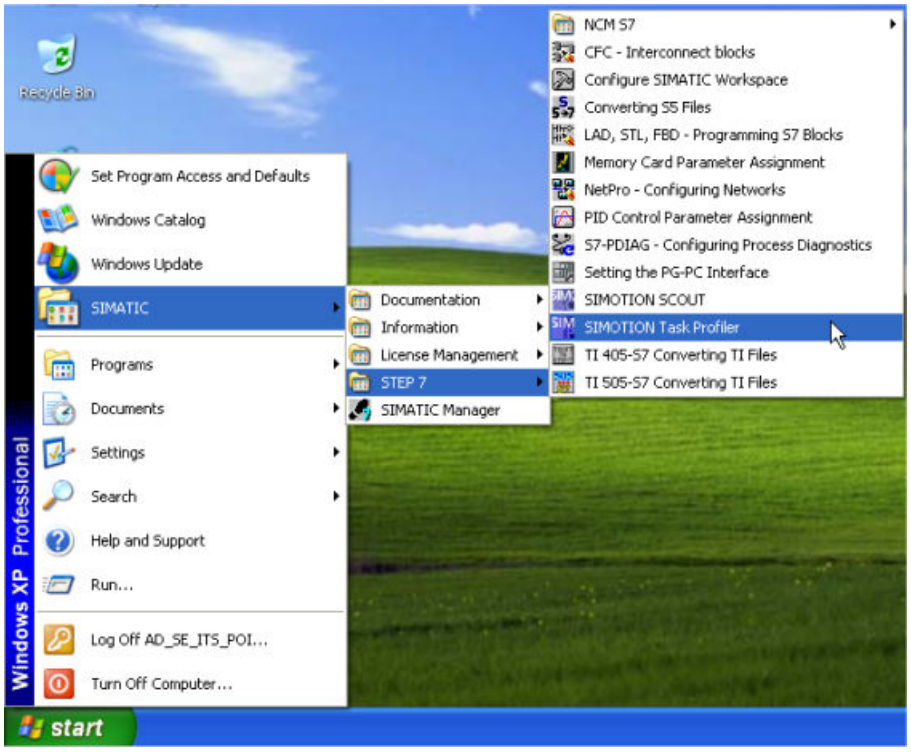


Figure 5-4 Opening the Task Profiler in the STEP7 program folder

When the Task Profiler is closed, the settings (e.g. most recently loaded Task Trace recording) are not saved. On reopening, the window for the display is empty (see figure below).

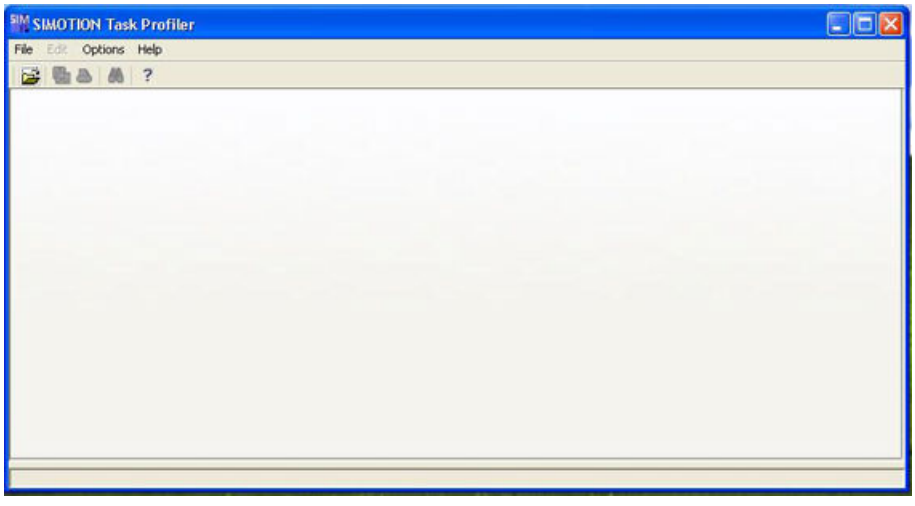


Figure 5-5 SIMOTION Task Profiler upon opening via the STEP7 program folder

Start using SCOUT device diagnostics

Note

In order to start the device diagnostics, an online connection to the target device must exist.

Start the Task Profiler in the device diagnostics as described below:

1. Switch to the **Task Manager** tab.
2. The following options are available:
   - Click the **Display** button. The Task Profiler starts. In addition, the trace file is loaded from the memory card of the target device and opened in a new tab in the Task Profiler. If you click the button again, the same file is read again.
   - Click the **Create snapshot** button. The setting that you have made using IT-Diag is reset to the default state. The settings made using the SIMOTION.ini file are retained. The Task Trace is started, written to the memory card, and then the Task Profiler started. If you click the button again, a new recording is started.

Note

If a trace file is not stored on the CF card of the target device, a new tab for the display is not shown.

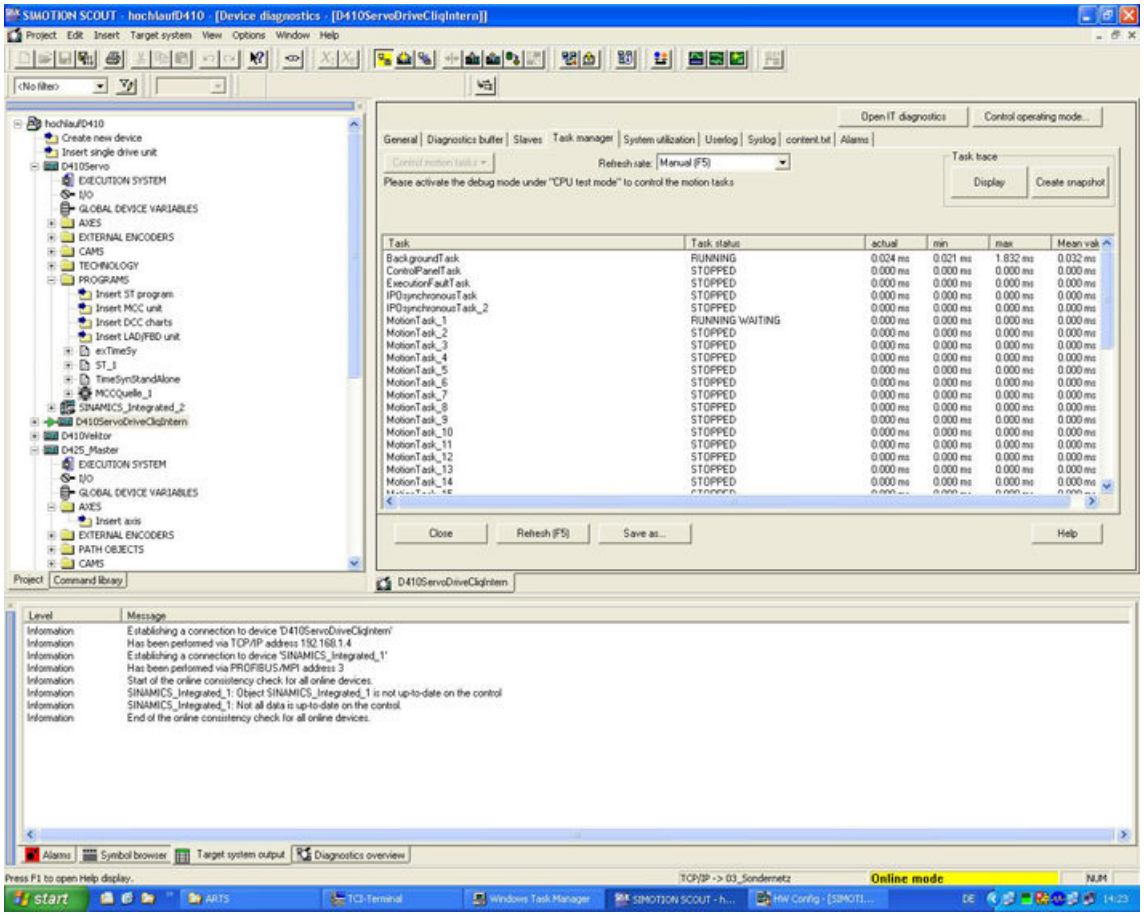


Figure 5-6 Start of Task Profiler via the device diagnostics

5.1.4.2 Structure of the SIMOTION Task Profiler

Structure of the interface

The structure of the SIMOTION Task Profiler is described below.

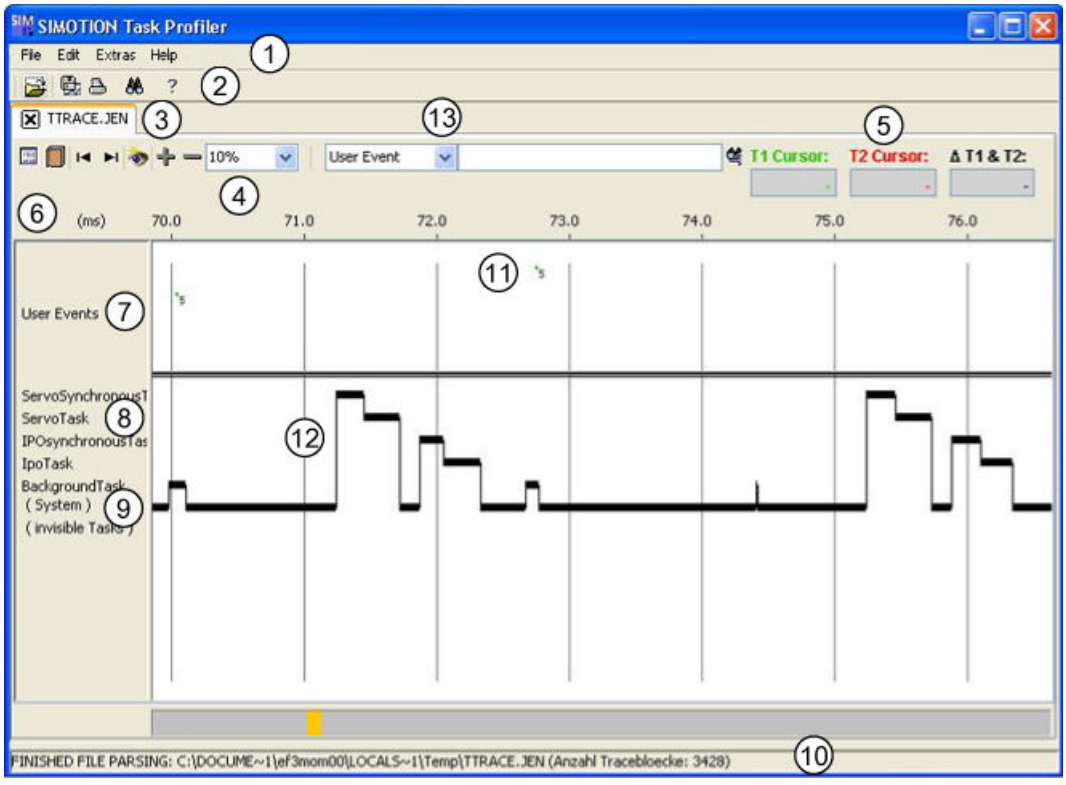


Figure 5-7 Structure of the user interface of the Task Profiler

No.	Description
1	Menu bar: File, Edit, Options, Help
2	Global toolbar
3	Tab with current trace. Several traces can be opened simultaneously.
4	Local toolbar
5	Display of T-measurement cursor
6	Time markings of the timing diagram
7	Labeling of user events
8	Display of task names
9	System tasks not displayed individually
10	Status bar
11	Marking of user events in the timing diagram
12	Timing diagram of tasks
13	Quick search

Structure of the menus

The menus are explained below. For more detailed information, refer to "Working with the SIMOTION Task Profiler".

File menu

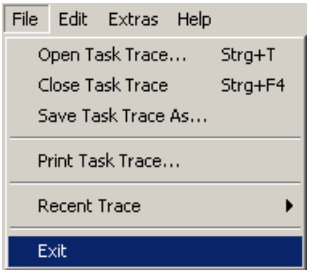


Figure 5-8 File menu

Table 5-2 Entries in the File menu

Entry	Description
Open Task Trace ...	Opens the JEN file
Close Task Trace ...	Closes the current Trace tab
Save Task Trace As ...	Saves the current trace
Print Task Trace ...	Prints the current trace
Recent trace	Selection of the most recently opened trace recordings
Exit	Exits the Task Profiler

Edit menu

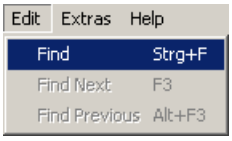


Figure 5-9 Edit menu

Table 5-3 Entries in the Edit menu

Entry	Description
Find	Opens the Find Options dialog box
Find Next	Searches for the next event corresponding to the Find Options
Find Previous	Searches for the previous event corresponding to the Find Options

Options - Menu

Figure 5-10 Options - Menu

Table 5-4 Entries in the Options menu

Entry	Description
User Event Configuration	Opens the dialog for assigning names to the user events of the current recording. You can export the configuration as an XML file and, if required, import it.

Help menu

Figure 5-11 Help menu

Table 5-5 Entries in the Help menu

Entry	Description
Help	Opens the Help system
About	Opens the About dialog box

Global toolbar

The following functions are available in the global toolbar:

Table 5-6 Global toolbar

Icon	Function
	Open Task Trace
	Saves Task Trace under a different name
	Print Task Trace
	Browses Task Trace for events
	Open help

For a description of the functions of the local toolbar, refer to "Analyzing a Task Trace".

5.1.4.3 Opening a Task Trace

Proceed as follows to open a trace file (trace stored on the PG/PC):

1. Open the **Open Task Trace...** dialog from the **File** > **Open Task Trace** menu

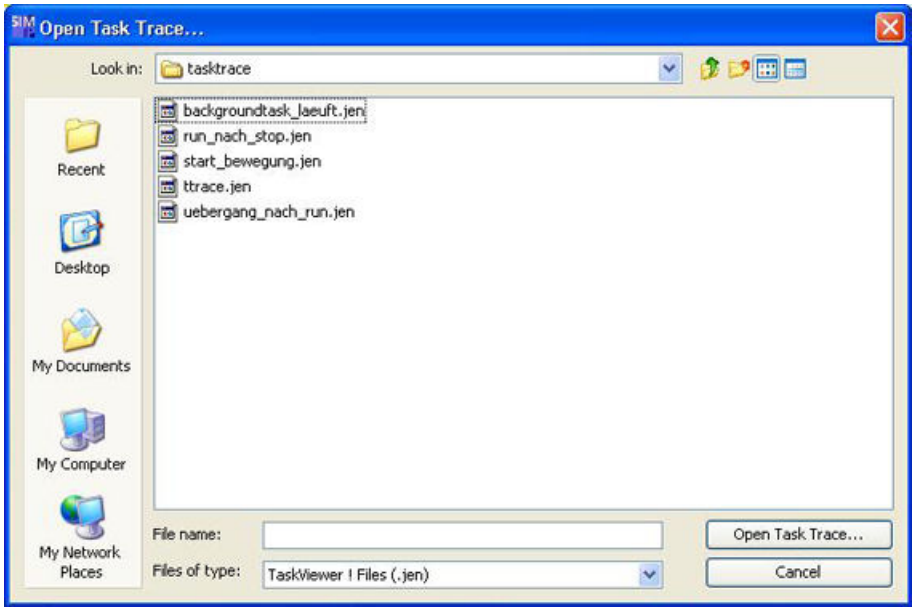


Figure 5-12 Open Task Trace

2. Select the JEN file you want to open in the Task Profiler, and confirm with **Open Task Trace...**

5.1.4.4 Control of Task Tracer

Control

You can control the Task Tracer using SIMOTION IT DIAG (standard web pages) of the target device or with system functions in the user program.

Note

For additional information, refer to the "Technology Packages System Functions" and "SIMOTION SCOUT Overview about Service and Diagnostic Options" documentation.

System functions in the user program

The following table provides an overview of the available system functions:

System function	Description
_taskTraceUserEvent	This function enters an event with a user-defined ID from 0 to 255 in the buffer of the SIMOTION Task Tracer.
_taskTraceStop	This function stops an active Task Tracer recording.

System function	Description
_taskTraceStart	This function starts the Task Trace recording. **Note:** When the function is called, whether or nor the Task Tracer has already been started is irrelevant. If the Task Tracer is active after the call, the function supplies 0 as return value.
_taskTraceWriteOut	The function stops a Task Trace recording that is already in progress and saves the Trace data as a JEN file (ttrace.jen) on the target device memory card under the following path: /USER/SIMOTION/HMI/SYSLOG/TASKTRACE/DIAG

5.1.4.5 Analyzing a Task Trace

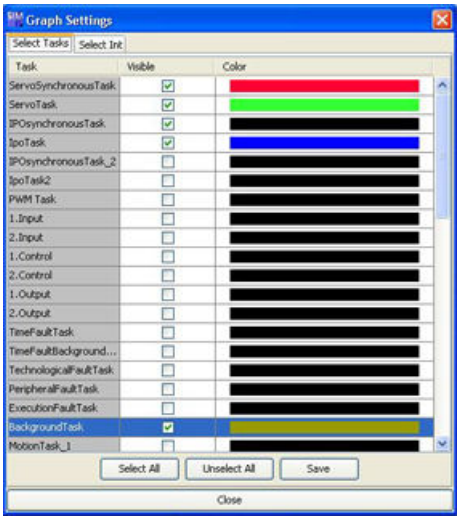
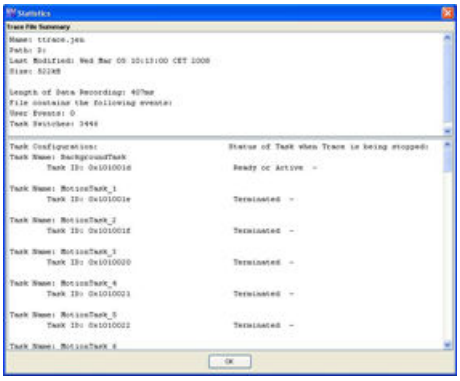
Note

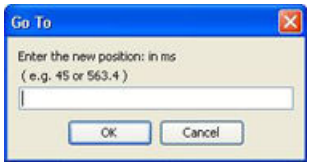
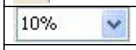
Peaks can occur during the recording, which indicate whether they can be re-started by means of background task testing. These peaks are irrelevant to the analysis.

The following options are available for analyzing the trace recording and for modifying the view:

Control via the local toolbar

Table 5-7 Functions of the local toolbar

Icon	Description
	Open the **Graph Settings** dialog. You can show or hide tasks and specify the color for each task in this dialog. 
	Open the statistics. This dialog provides an overview of the tasks available in the system and their status at the end of the trace. 
	Go to start of trace.
	Go to end of trace.

Icon	Description
	Open the **Go To** dialog. You can use this dialog to jump to the entered time value. 
	Zoom In
	Zoom Out
	Zoom factor can be entered (between 1 and 5,000).
	Fast search using User Event or Task Switch

Control via the mouse

Table 5-8 Functions via mouse or shortcut

Click button or shortcut	Description
Right-click event icon	Info box on selected event.
Left- or right-click	Generates a red or green vertical guide line, at whose upper end the time is displayed in ms. The values are displayed in the following information panel. 
Shift + left- or right-click	Removes the red or green guide line.
Ctrl + left-click	Generates a horizontal guide marking. This can also be generated by directly left-clicking the task name.
Ctrl + right-click	Removes the horizontal marking. Clicking the currently selected task name also removes the horizontal marking.
Right-click task name	Info box for corresponding task.

5.1.4.6 Saving the trace data

The recorded trace data is stored as a JEN file (ttrace.jen) on the memory card of the target device under the following path: /USER/SIMOTION/HMI/SYSLOG/TASKTRACE/DIAG

To save trace data under a different name, follow the procedure described below:

1. Open the **Save Task Trace As** dialog from the **File** > **Save Task Trace As ..** menu.
2. Enter the desired file name and save it. If a file of the same name already exists, it is overwritten.

If required, you can use a card reader to read out the JEN files on the memory card.

5.1.4.7 Find

You can use the find function (can be invoked in the main menu using **Edit** > **Find**) to search for the next user event or the next task in the trace recording.

If you are searching for a specific event or task, you must also specify the event ID or the task designator. You can use the **Find Previous** or **Find Next** button to search backwards or forwards, respectively.

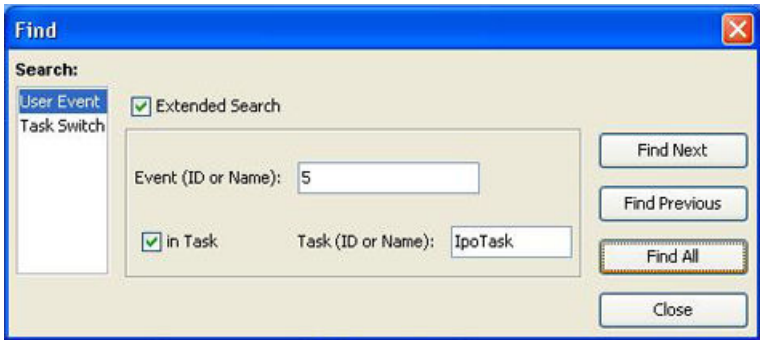


Figure 5-13 Find Event

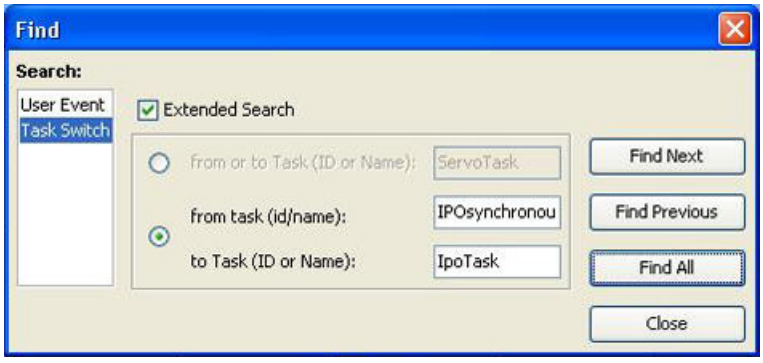


Figure 5-14 Find Task

5.2 Overview of service and diagnostics options

5.2.1 Preface

5.2.1.1 Preface

Contents

This document is part of the **SIMOTION Service and Diagnostics** documentation package.

Area of application

This manual is valid for SIMOTION V5.2.

Information blocks in this manual

The following is a list of sections included in this manual along with a description of the information presented in each section.

- **Part I: Service on the device**
  This section outlines the diagnostics options available on the device and contains references to additional descriptions as well as descriptions relating to specific devices.
- **Part II: Service without SCOUT Engineering System (PC-based, SIMOTION IT)**
  This section outlines the diagnostics options available with PC-based systems and with the SIMOTION IT web server, and contains references to additional descriptions.
- **Part III: Service with SCOUT Engineering System**
  This section outlines the diagnostics options available with the SCOUT Engineering System and contains references to additional descriptions.
- **Appendix**
  The appendix contains detailed descriptions and guidelines relating to the individual sections.
- **Index**
  Index for locating information.

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

5.2.1.2 Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

5.2.2 Fundamental safety instructions

5.2.2.1 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

5.2.2.2 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

5.2.2.3 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to

an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

5.2.2.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

5.2.2.5 Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

5.2.3 Introduction

5.2.3.1 Overview of service and diagnostics options

This manual lists the system diagnostics options available for SIMOTION devices. It also contains references to additional manuals and online help texts featuring the detailed information that is applicable in each case. The Manuals and Commissioning Manuals for the individual platforms describe device-specific diagnostics options in detail.

Note

The full version of this manual is available as an online help containing links.

This manual (i.e. the PDF version) provides an overview of how this help text is organized and structured. It does not include links to subjects that are only covered in the online help. The relevant manual is referred to in each case.

Note

In the event of an fault (e.g. CPU STOP), analyze the entries in the diagnostic buffer. You can access this in any of the scenarios covered here (Parts I to III). An overview of the scenarios is provided later.

Comprehensive diagnostic data (Page 3359) can be generated to enable the machine manufacturer or SIEMENS to carry out a more in-depth analysis.

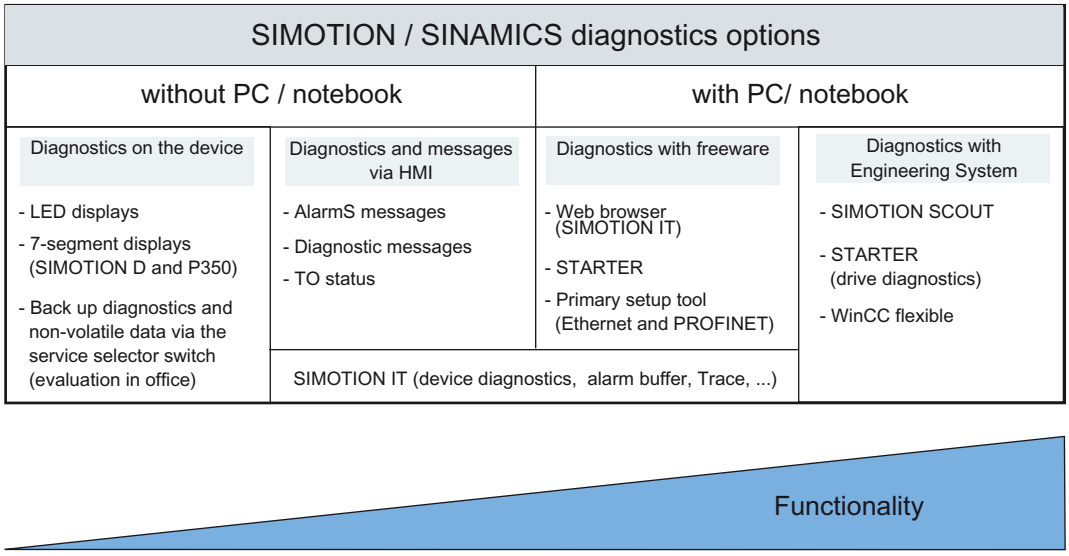


Figure 5-15 Overview of functions associated with diagnostics options

Note

If you contact the hotline for support, you should provide a description of the error/fault scenario that is as precise and detailed as possible.
Depending on the nature of the fault, the following information may be useful:

- Diagnostic buffer backup data (see also the descriptions of scenarios without the SCOUT Engineering System (Page 3360) and with the SCOUT Engineering System (Page 3382))
- Full alarm messages with numbers and all additional information
- State of LEDs/7-segment displays
- Installed software versions (SIMOTION SCOUT > Help > Information > System info...)
- Screenshots of error/fault messages and/or dialogs

Part I

Service on the device (Page 3319)

Part I describes the diagnostics and service options that can be implemented directly on the device.

Figure 5-16 Example: Module D445-2 DP/PN

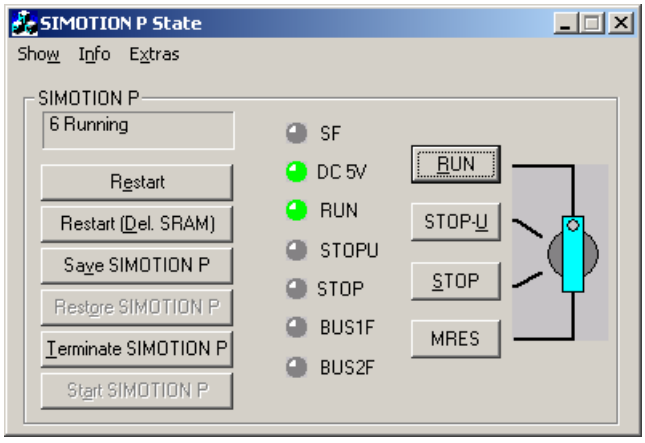


Figure 5-17 Example - SIMOTION P350 - P state

Part II

Service without SCOUT Engineering System (PC-based, SIMOTION IT) (Page 3352)

Part II describes the service options that can be implemented without the SCOUT Engineering System, via the Ethernet interface of SIMOTION devices. Diagnostic functions can be viewed using an Internet browser.

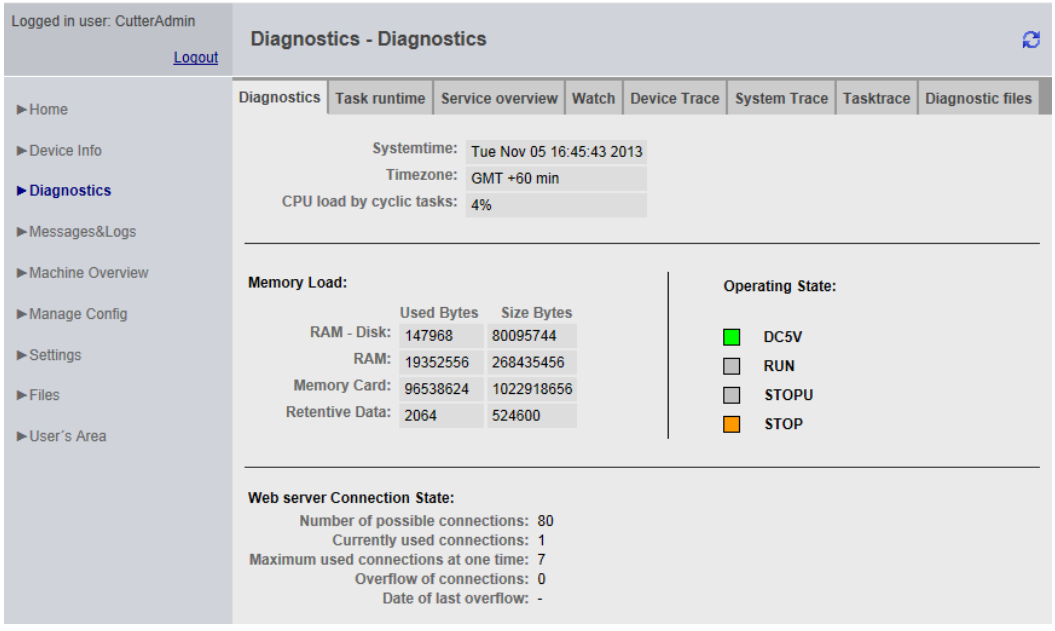


Figure 5-18 Example: SIMOTION IT web server - diagnostics in browser

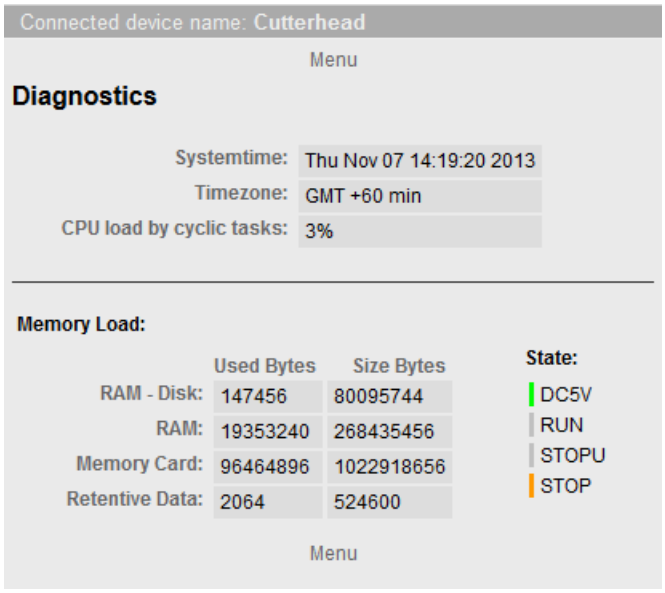


Figure 5-19 Example: simplified HTML pages, e.g. for PDA

Part III

Service with SCOUT Engineering System (Page 3367)

Part III presents the comprehensive range of SIMOTION SCOUT Engineering System functions that are available for error/fault diagnostics.

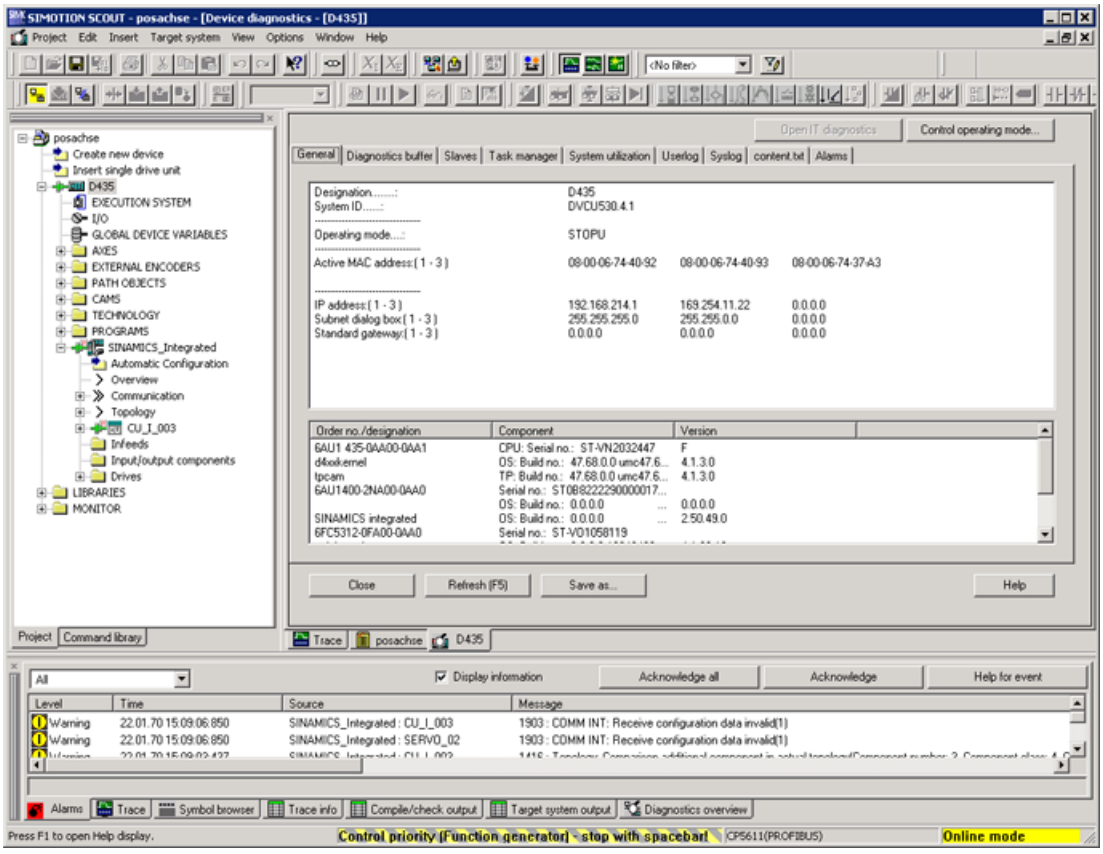


Figure 5-20 Example: SIMOTION SCOUT, Device diagnostics

5.2.4 Part I: Service on the device

5.2.4.1 Overview

The device hardware offers various displays and interfaces that may be used for diagnostics purposes. Diagnostic information can be displayed either directly on the device (e.g. by means of LEDs) or indirectly (e.g. HMI via Ethernet interface). In the case of SIMOTION D, a service selector switch can be used to initiate a backup of diagnostic data on the device and write this to the memory card.

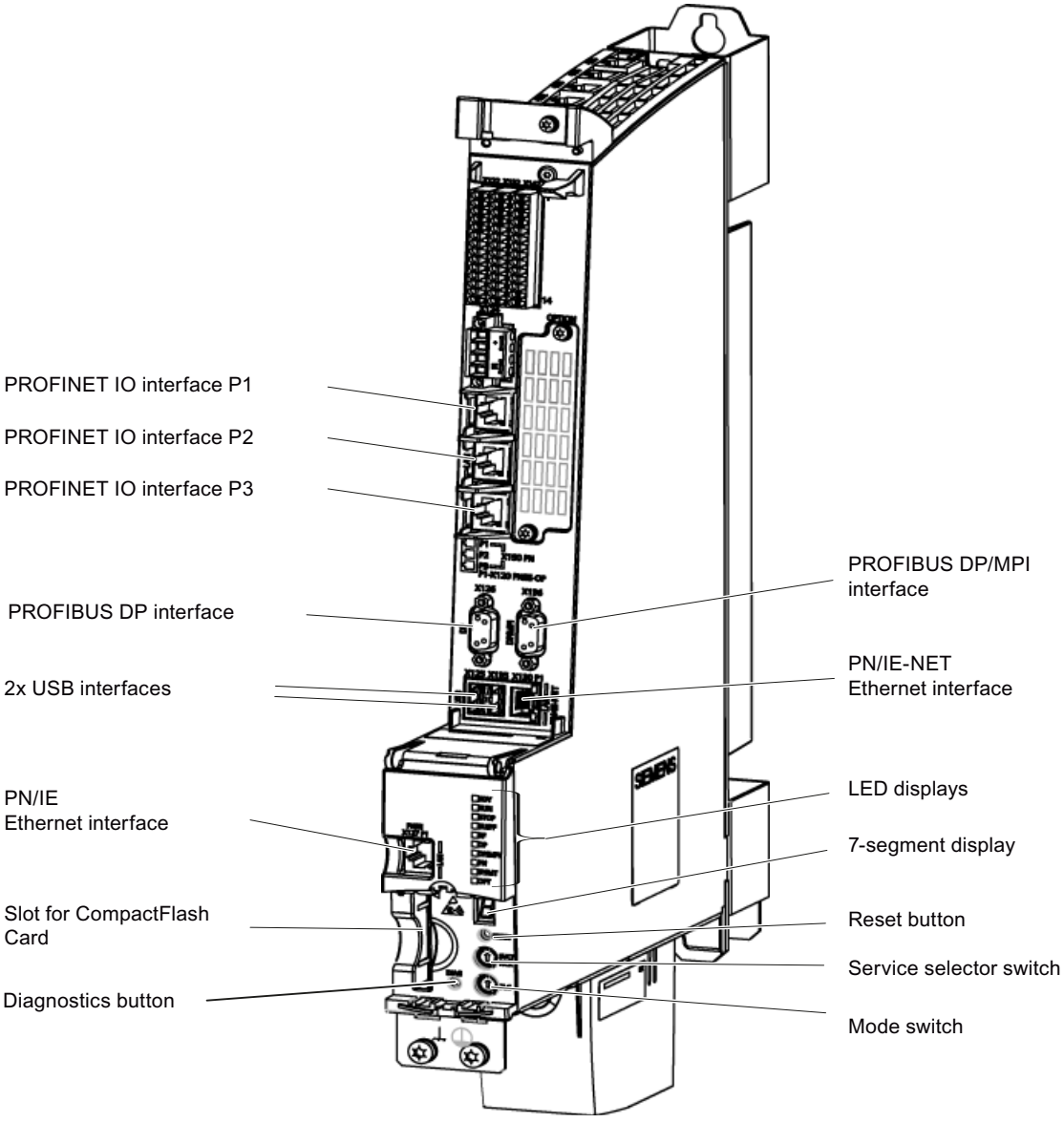


Figure 5-21 Interfaces and front panel elements - example of SIMOTION D445-2 DP/PN

Service selector switch/button

The service selector switch enables diagnostic data backed up in a SIMOTION device in the event of an error/fault to be stored offline without SIMOTION SCOUT. For details, see Backing up diagnostic data and non-volatile data (Page 3348).

SIMOTION D modules have a service selector switch that is important for service and diagnostic functions. In "normal" operation, this switch must remain in the 0 position. See also Backing up during operation using a service selector switch (Page 3396) and Backing up during ramp-up using a service selector switch or INI file (Page 3399).

With SIMOTION P320-3/P350-3, virtual versions of the service selector and mode switches and the LEDs appear in the *SIMOTION P state* application. See also Backing up during operation using a service selector switch (Page 3396) and Backing up during ramp-up using a service selector switch or INI file (Page 3399).

SIMOTION C does not provide any switches on the device for diagnostic functions. Errors/faults and states are displayed by means of LEDs.

Part II (Page 3359) contains a detailed description of how diagnostic data and non-volatile data should be handled.

Mode switch

You can use the mode switch to set the operating mode on the device. With SIMOTION P, the function of this switch is implemented by means of the SIMOTION P state application. It is also possible to change the operating mode via SIMOTION SCOUT.

Operating modes, based on the example of SIMOTION D4x5-2:

- RUN
  SIMOTION D4x5-2 is processing the user program.
  The technology packages are active in this state.
  They can execute commands from the user program.
- STOPU
  SIMOTION D4x5-2 is not processing a user program.
  The technology packages are active. Test and commissioning functions can be executed.
  The I/O modules are in a safe state. This means, for example, that digital outputs are "LOW" and analog outputs are de-energized (no current, no voltage).
- STOP
  SIMOTION D4x5-2 is not processing a user program.
  It is possible to load an entire user program.
  All system services (communication, ...) are active.
  The I/O modules are in a safe state. This means, for example, that digital outputs are "LOW" and analog outputs are de-energized (no current, no voltage).
  The technology packages are not active.
- MRES
  Switch position for overall reset of the module.

Note

Please take note of the information in the Manuals for the switches.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Manual
- *SIMOTION D410* Commissioning Manual
- SIMOTION D410-2 Manual
- SIMOTION D410-2 Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Manual
- *SIMOTION P350-3 and Panels* Manual

5.2.4.2 LEDs

Overview

Every SIMOTION device features LEDs that are used for device diagnostics. The LED displays indicate the different operating modes and any faults that occur. They do so by lighting up, flashing, or flickering in different colors.

With SIMOTION P320-3/P350, representations of the LEDs appear in the *SIMOTION P state* application.

Note

A STOPU LED may also light up if an axis control panel is activated.
The STOPU LED flickers when a CF/MMC card is being formatted. (D4x5-2: SU/PF-LED)

This does not indicate a fault.

See also

Overview (Page 3359)

Backing up diagnostic data and non-volatile data (Page 3348)

Backing up during operation using a service selector switch (Page 3396)

Backing up during ramp-up using a service selector switch or INI file (Page 3399)

Part II: Service without SCOUT Engineering System (PC-based, SIMOTION IT) (Page 3352)

SIMOTION C

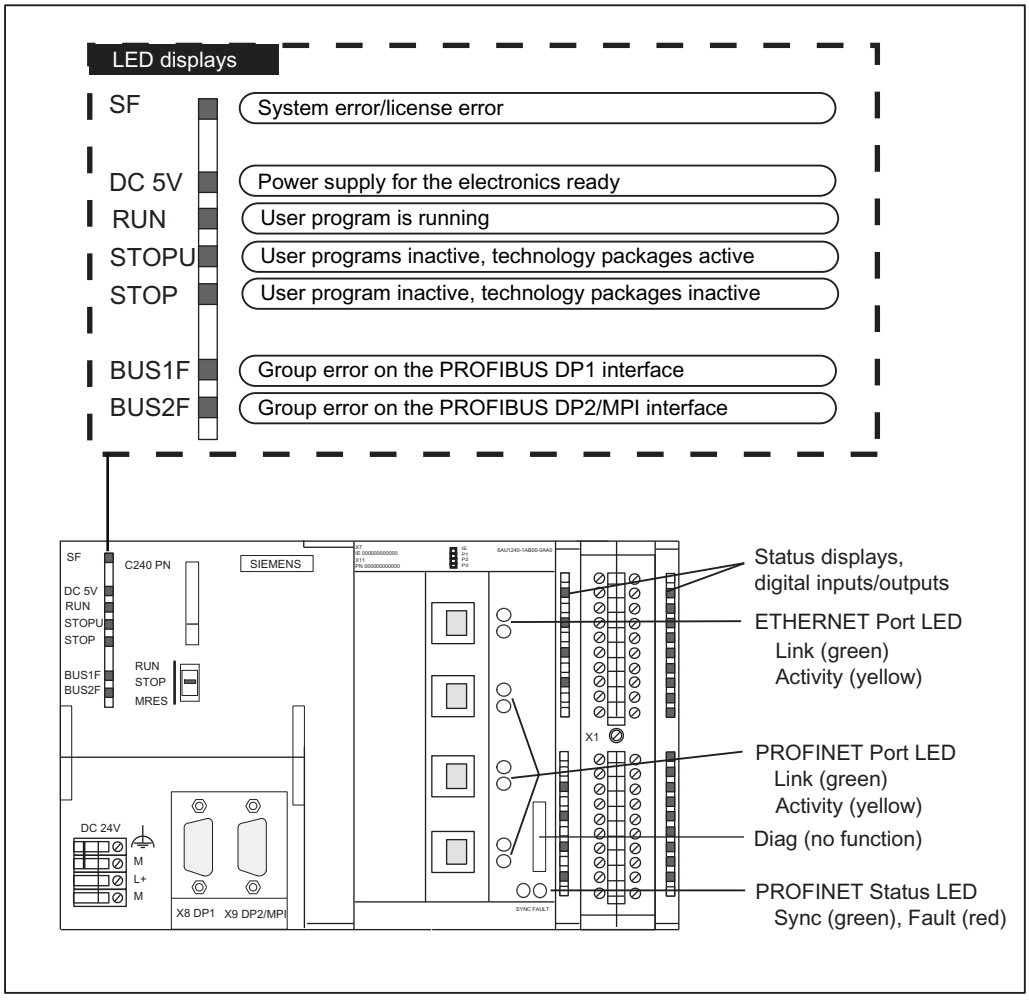


Figure 5-22 LED displays on the device - example of SIMOTION C240 PN

Note

The SIMOTION C Operating Instructions contain a detailed description of the LED displays, including all the possible combinations and flashing frequencies.

Typical faults

Table 5-9 Typical faults

Fault	Frequent causes	Remedy
SF LED lights up red (error/fault state on the SIMOTION C)		
	• An event that can be acknowledged is pending (alarm, message, note) • A fault has occurred to which the user program cannot respond.	Acknowledge the event. Switch the SIMOTION device off and then on again.
SF LED flashes red (error/fault state on the SIMOTION C)		
	• License missing for licensed technology/optional objects	Check the licenses.
5 VDC does not light up (supply voltage for electronics)		
	• No line supply connected or switched on • No specified load current supply connected • Module not connected correctly • Defective module	Check the line supply connection and the module
STOP LED lights up yellow (control is in STOP mode - diagnostics with Engineering System or web server)		
	• I/O access error • Program error (e.g. floating point exception) • Technological alarms with CPU STOP response	Correct I/O access **Localize error:** Check entries in the device diagnostics Example: Entry "Operating mode transition blocked" -> check error before this -> entry "I/O access error"
STOP LED flashes yellow (overall reset request)		
	• Memory card has been removed • The non-volatile data does not match up with the project on the memory card	Memory reset
STOPU LED lights up or flashes yellow (control in STOP mode of user program)		
	• The technology packages are active. • The user program is not active or is faulty • Device stuck at startup task	Check the entries in the diagnostic buffer and the user program. Switch the control to RUN mode.

Fault	Frequent causes	Remedy
BUS1F, BUS2F LED lights up red (fault on PROFIBUS DP interface)		
	• Terminating resistor missing or in the wrong place • Not all of the connected devices are switched on • Cabling fault • Incorrect baud rate configured or incorrect baud rate set on a bus node • Configuration error • Parameter assignment error	Check terminating resistor, bus nodes, cabling, baud rate, configuration settings, and configuration in HW Config

Table 5-10 Typical C240 PN errors

Fault	Frequent causes	Remedy
PROFINET Link LED does not light up		
	• There is no physical connection • The connected device is not switched on	Check the cabling, connectors, and device.
PROFINET Activity LED does not light up continuously		
	• There is no telegram traffic.	If the Link LED is green, use a ping command to check that the system is ready for communication
PROFINET Fault LED lights up (bus fault)		
	• No physical connection to a subnet/switch • Incorrect transmission rate • Full duplex transmission is not activated	Check configuration settings, check IO device, check connection
PROFINET Fault LED flashes red (bus fault)		
	• Failure of a connected I/O device. • At least one of the assigned I/O devices cannot be addressed • Incorrect or no configuration settings	Check configuration settings, check IO device
PROFINET Sync LED does not light up or flashes		
	• SIMOTION C task system is not synchronized with the send cycle of PROFINET IO IRT	Check configuration settings, check fault message in SINAMICS diagnostic buffer

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions

SIMOTION P320-3/P350-3

The LEDs and mode switch, which are implemented as hardware on other SIMOTION platforms, are displayed in virtual form on the SIMOTION P system screen.

This is achieved via the *SIMOTION P state* application.

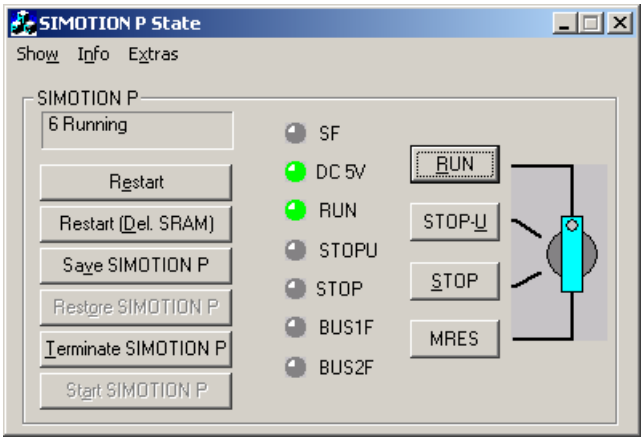


Figure 5-23 SIMOTION P state

Note

The relevant Manual/Commissioning Manual contains a detailed description of the LED displays, including all the possible combinations and flashing frequencies.

Typical errors

Table 5-11 Typical errors

Error	Frequent causes	Remedy
SF LED lights up red (error/fault state on the SIMOTION P)		
	• An event which can be acknowledged is pending (alarm, message, note) • A fault has occurred that does not allow a response from the user program	Acknowledge the event. Switch the SIMOTION device off and then on again.
SF LED flashes red (error/fault state on the SIMOTION P)		
	• License missing for licensed technology/optional objects	Check the licenses.
STOP LED lights up yellow (SIMOTION P in STOP mode)		
	• I/O access error • Program error (e.g. floating point exception) • Technological alarms with CPU STOP response	Correct I/O access **Localize error:** Check entries in the device diagnostics Example: Entry "Operating mode transition blocked" -> check error before this -> entry "I/O access error"
STOP LED flashes yellow (overall reset request)		

Error	Frequent causes	Remedy
	• Memory card has been removed • The non-volatile data does not match up with the project on the memory card	Overall reset
STOPU LED lights up or flashes yellow (SIMOTION P in STOP user program mode)		
	• The technology packages are active • The user program is not active or is faulty • Device stuck at startup task	Check the entries in the diagnostics buffer and the user program. Switch the control to RUN mode.
BUS1F, BUS2F LED (error/fault state on PROFIBUS DP interface, P350-3 only)		
	• Terminating resistor missing or in the wrong place • Not all of the connected devices are switched on • Cabling fault • Incorrect baud rate configured or incorrect baud rate set on a bus node • Configuration error • Parameter assignment error	Check terminating resistor, bus nodes, cabling, baud rate, configuration settings, and configuration in HW Config

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION P320-3 and Panels* Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

SIMOTION D4x5-2/D4x5

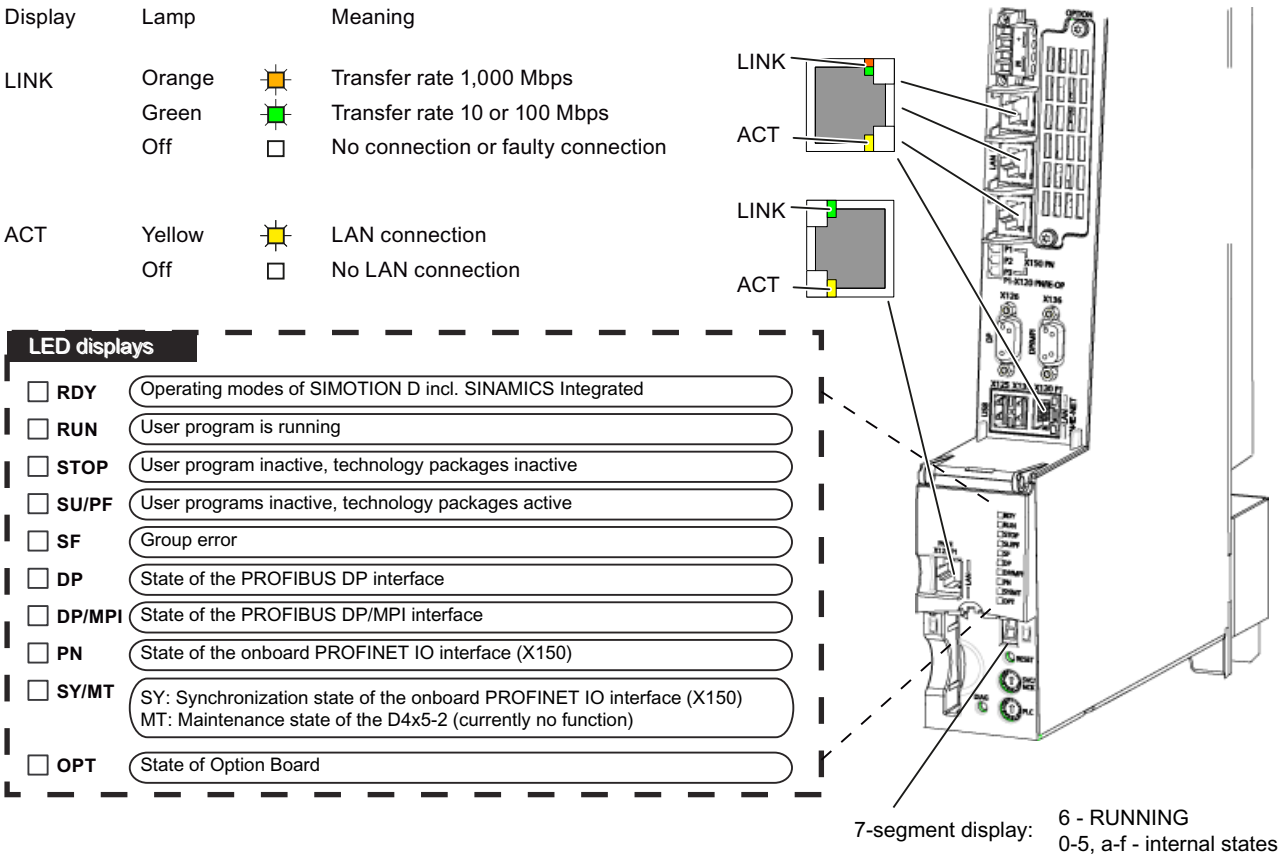


Figure 5-24 LED displays on the device - example of SIMOTION D4x5-2

In addition to the SIMOTION devices themselves, optional modules and interfaces such the PROFINET interface CBE30-2 for D4x5-2 DP/PN also feature LEDs for displaying states and fault diagnostics.

Note

The relevant Commissioning Manuals contain a detailed description of the LED displays, including all the possible combinations and flashing frequencies.

Typical faults

Table 5-12 Typical D4x5-2/D4x5 errors

Fault	Frequent causes	Remedy
RDY LED lights up red or flashes red, flashes quickly 2 Hz (SIMOTION D or SINAMICS Integrated fault)		
	• SINAMICS Integrated has not ramped up or is faulty	Correct and acknowledge fault Acknowledgment carried out with e.g. operator panel, engineering tool, etc. connected
STOP LED lights up yellow (control is in STOP mode - diagnostics with Engineering System or web server)		
	• I/O access error • Program error (e.g. floating point exception) • Technological alarms with CPU STOP response	Correct I/O access **Localize error:** Check entries in the device diagnostics Example: Entry "Operating mode transition blocked" -> check error before this -> entry "I/O access error"
STOP LED flashes yellow, slow flashing (0.5 Hz) (overall reset request)		
	• Memory card has been removed • The non-volatile data does not match up with the project on the memory card	Memory reset
SU/PF LED lights up or flashes yellow, fast flashing (2 Hz) (control in STOP mode of user program)		
	• The technology packages are active. • The user program is not active or is faulty • Device stuck at startup task	Check the entries in the diagnostic buffer and the user program. Switch the control to RUN mode.
SF LED lights up red (error/fault state on the SIMOTION D)		
	• An event that can be acknowledged is pending (alarm, message, note)	Acknowledge the event.
SF LED flashes red, slow flashing (0.5 Hz) (error/fault state on the SIMOTION D)		
	• License missing for licensed technology/optional objects	Check the licenses.
DP, DP/MPI LED lights up red (bus fault)		
	• Terminating resistor missing or in the wrong place • Not all of the connected devices are switched on • Cabling fault • Incorrect baud rate configured or incorrect baud rate set on a bus node	Check terminating resistor, bus node, cabling, and configuration settings
PN LED flashes red (bus fault)		

Fault	Frequent causes	Remedy
	• Failure of a connected I/O device. • At least one of the assigned I/O devices cannot be addressed • Incorrect or no configuration settings	Check configuration settings, check IO device
SY/MT LED does not light up or flashes		
	• SIMOTION D task system is not synchronized with the send cycle of PROFINET IO IRT • SINAMICS Integrated and ext. DP interfaces are not yet synchronized with the PROFINET IO IRT cycle	Check configuration settings, check fault message in SINAMICS diagnostic buffer
OPT LED lights up/flashes red, slow flashing (2 Hz)		
	• Bus fault (CBE30-2 in the case of D4x5-2 or CBE30 in the case of D4x5): – Failure of a connected I/O device – At least one of the assigned I/O devices cannot be addressed – Incorrect or no configuration settings • Firmware download faulty	Check devices (still set to factory settings?), check connection, check cycle clock
Link LED does not light up		
	• There is no physical connection • The connected device is not switched on	Check the cabling, connectors, and device.
Activity LED does not light up		
	• There is no telegram traffic.	If the Link LED is green, use a ping command to check that the system is ready for communication

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *S120 Control Units and Supplementary System Components* Manual

See also

CBE30-2/CBE30 Communication Board (Page 3333)

SIMOTION D410-2/D410

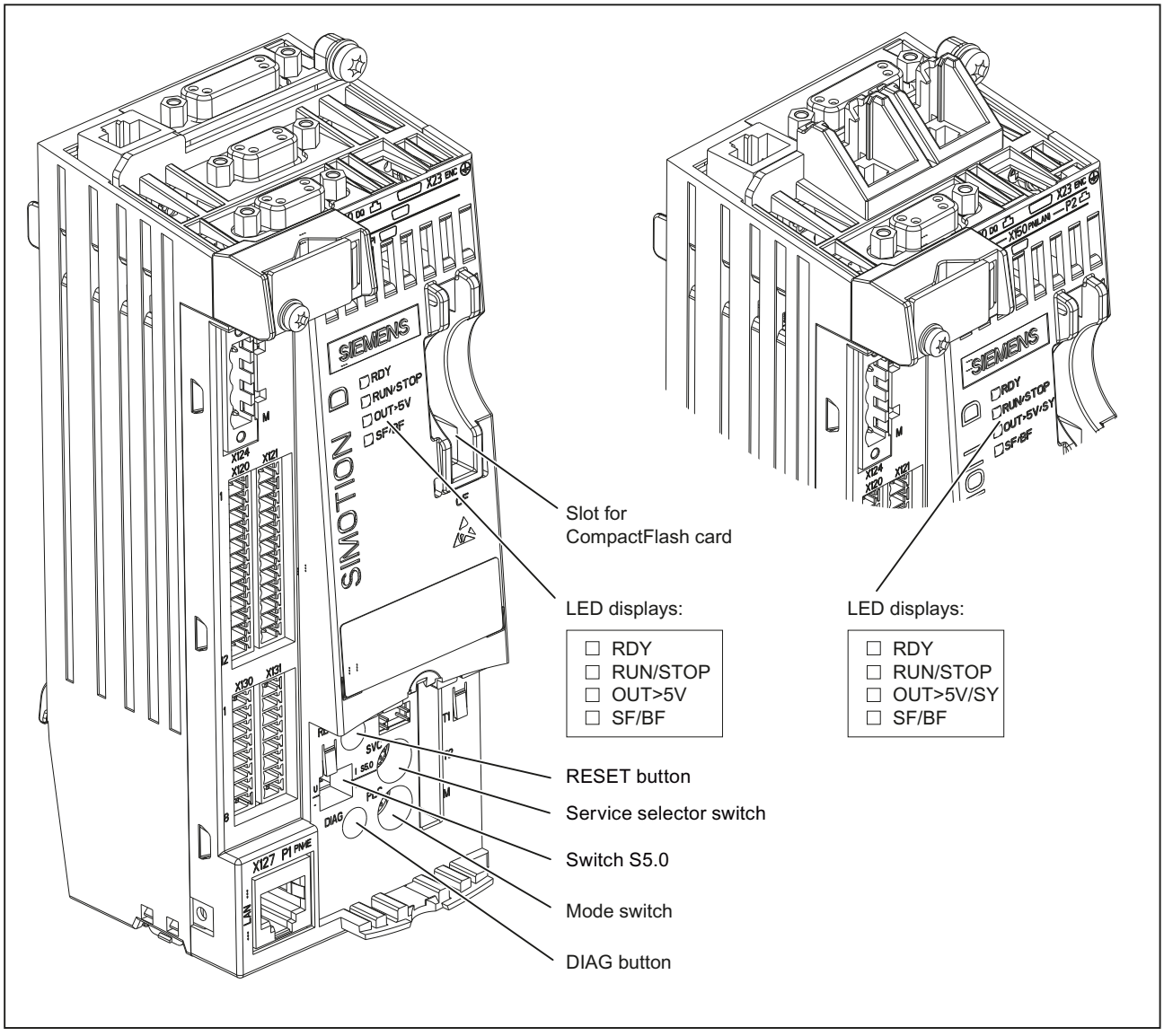


Figure 5-25 LED displays on the D410 device

Note

The Commissioning Manual contains a detailed description of the LED displays, including all the possible combinations and flashing frequencies.

Typical faults

Table 5-13 Typical D410-2/D410 faults

Fault	Frequent causes	Remedy
RDY LED lights up red or flashes red, flashes quickly 2 Hz (SIMOTION D or SINAMICS Integrated fault)		
	• SINAMICS Integrated has not ramped up or is faulty	Correct and acknowledge fault Acknowledgment carried out with e.g. operator panel, engineering tool, etc. connected
SF/BF LED lights up red (error/fault state on the SIMOTION D)		
	• An event that can be acknowledged is pending (alarm, message, note)	Acknowledge the event.
SF/BF LED flashes red, slow flashing (0.5 Hz) (error/fault state on the SIMOTION D)		
	• License missing for licensed technology/optional objects	Check the licenses.
SF/BF LED flashes red, fast flashing (2 Hz) (bus fault)		
	• A bus fault is pending – PROFIBUS master At least 1 slave is missing – PROFIBUS slave No parameter assignment master found – PROFINET Failure of a connected I/O device; at least one of the assigned I/O devices cannot be addressed; incorrect or no configuration settings	Check the bus nodes, cabling, configuration settings, and configuration in HW Config.
all LEDs light up yellow (D410-2) **RUN/STOP lights up red** (D410)		
	• Power-up of the SIMOTION D without CF card or with CF card without valid operating system (the boot loader may be defective).	Check the CompactFlash Card.
RUN/STOP LED lights up yellow		
	Control is in stop mode - diagnostics with Engineering System or web server	
	• I/O access error • Program error (e.g. floating point exception) • Technological alarms with CPU STOP response	Correct I/O access **Localize error:** Check entries in the device diagnostics Example: Entry "Operating mode transition blocked" -> check error before this -> entry "I/O access error"
	Control in STOP mode of user program	

Fault	Frequent causes	Remedy
	• The technology packages are active. • The user program is not active or is faulty • Device stuck at startup task	Check the entries in the diagnostic buffer and the user program. Switch the control to RUN mode.
RUN/STOP LED flashes yellow, slow flashing (0.5 Hz) (overall reset request)		
	• Memory card has been removed • The non-volatile data does not match up with the project on the memory card	Memory reset
OUT > 5V lights yellow or flashes yellow (electronic power supply for measuring system)		
	• Electronic power supply for measuring system > 5 V	The electronic power supply is configured for 24 V encoders. If you wish to connect a 5 V encoder, please check the parameter settings.
OUT > 5 V / SY off or flashes ("SY" only for D410-2 DP/PN)		
	• SIMOTION D task system is not synchronized with the send cycle of PROFINET IO IRT • SINAMICS Integrated and ext. DP interfaces are not yet synchronized with the PROFINET IO IRT cycle	Check configuration settings, check fault message in SINAMICS diagnostic buffer

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D410* Manual
- *SIMOTION D410* Commissioning Manual
- *S120 Control Units and Supplementary System Components* Manual
- *SIMOTION D410-2* Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual

CBE30-2/CBE30 Communication Board

The device is connected to PROFINET IO using the Communication Board Ethernet CBE30-2 interface module for the SIMOTION D4x5-2 DP/PN and the CBE30 interface module for the SIMOTION D4x5. The module supports PROFINET IO with isochronous Realtime Ethernet (IRT), PROFINET IO with RT, and standard TCP/IP communication. The Option Board has an X1400 interface with four ports and integrated switch functionality.

Note

A CBE30 can only be inserted into a SIMOTION D4x5. It cannot be inserted into a SIMOTION D4x5-2.

A CBE30-2 can only be inserted into a SIMOTION D4x5-2 DP/PN used as a second PROFINET interface. It cannot be inserted into a SIMOTION D4x5 or D4x5-2 DP.

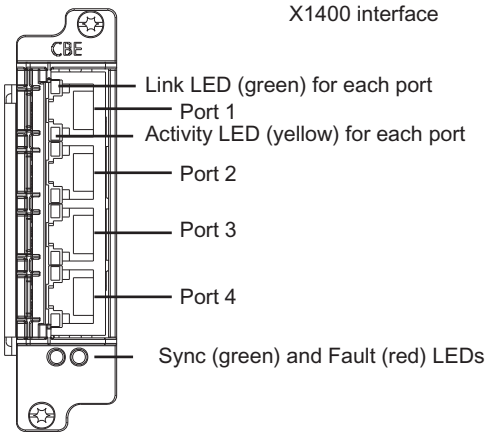


Figure 5-26 Ethernet CBE30-2/CBE30 Communication Board

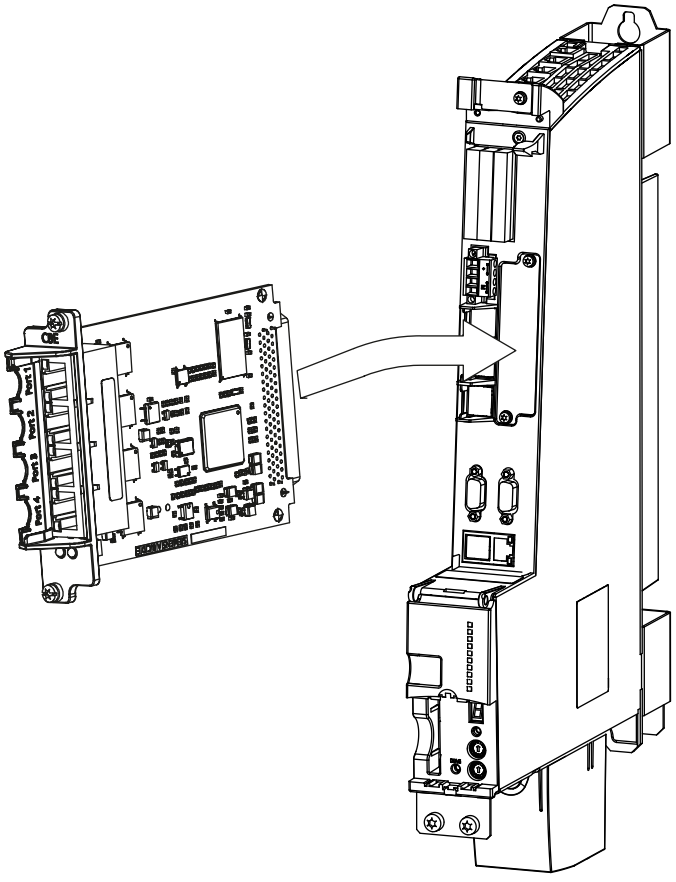


Figure 5-27 Example of D4x5-2: Inserting a CBE30-2

Typical faults

Table 5-14 Typical CBE30-2/CBE30 faults

Fault	Frequent causes	Remedy
Link LED does not light up		
	• There is no physical connection • The connected device is not switched on	Check the cabling, connectors, and device.
Activity LED does not light up		
	• There is no telegram traffic.	If the Link LED is green, use a ping command to check that the system is ready for communication
Fault LED lights up red (bus fault)		
	• No physical connection to a subnet/ switch • Incorrect transmission rate • Full duplex transmission is not activated	Check configuration settings, check IO device, check connection
Fault LED flashes red, fast flashing (2 Hz) (bus fault)		
	• Failure of a connected I/O device. • At least one of the assigned I/O devices cannot be addressed • Incorrect or no configuration settings	Check configuration settings, check IO device
Fault LED flashes red, slow flashing (0.5 Hz)		
	CBE30-2 startup stopped	CBE30-2 is plugged into wrong Control Unit. (CBE30-2 is only supported by SIMOTION D4x5-2 DP/PN, not by SIMOTION D4x5-2 DP)
Sync LED does not light up or flashes		
	• SIMOTION D task system is not synchronized with the send cycle of PROFINET IO IRT • SINAMICS Integrated and ext. DP interfaces are not yet synchronized with the PROFINET IO IRT cycle	Check configuration settings, check fault message in SINAMICS diagnostic buffer

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual

CX32-2/CX32 Controller Extension

The Controller Extension enables scaling of the drive-side computing power of products within the SIMOTION D range. This allows additional drives to be connected, for example.

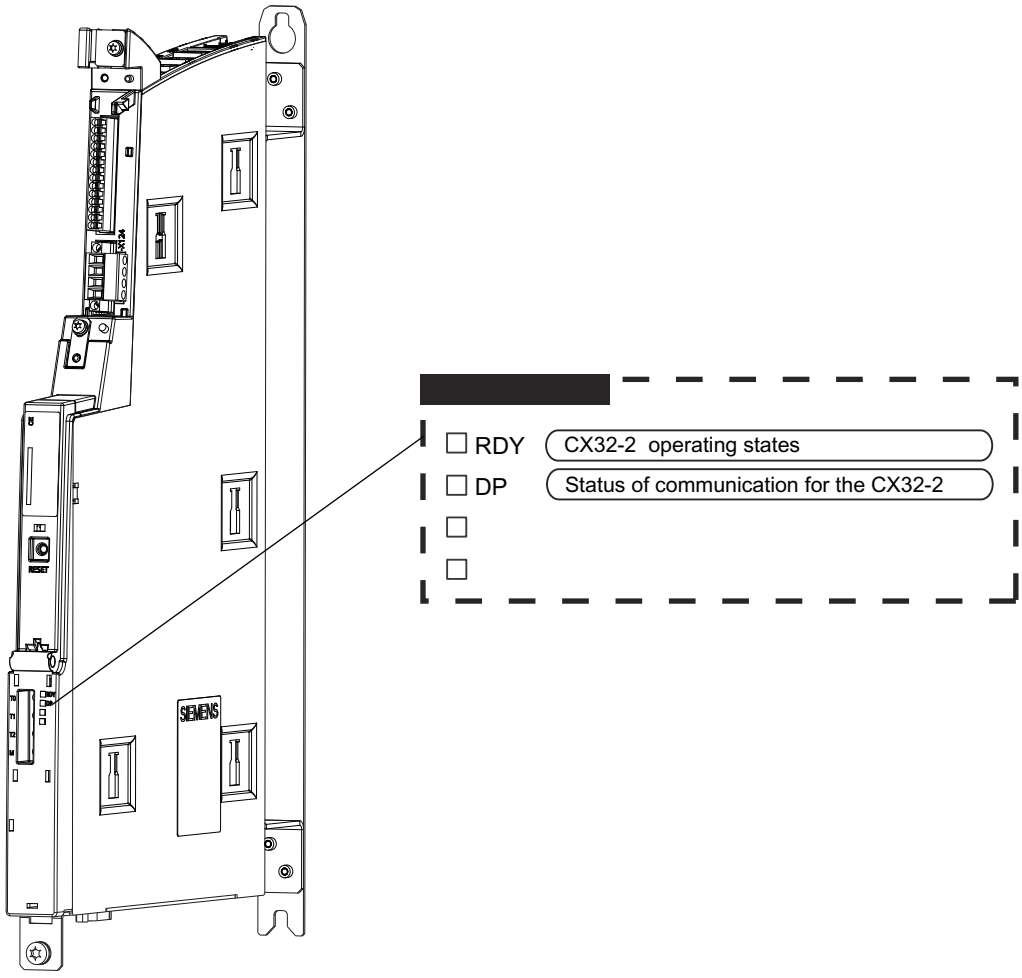


Figure 5-28 LED displays - example of CX32-2

Note

The *SIMOTION D4x5* Commissioning and Hardware Installation Manual and the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual contain detailed descriptions of the LED displays, including all the possible combinations and flashing frequencies.

Typical faults

Table 5-15 Typical CX32-2/CX32 faults

Fault	Frequent causes	Remedy
RDY LED does not light up (power supply)		
	• The electronic power supply is missing or outside the permissible tolerance range	Check power supply
RDY LED lights up red or flashes red, fast flashing 2 Hz (error on startup)		
	• At least one fault is pending (e.g. RESET, watchdog monitoring, basic system error) • CX32/CX32-2 is ramping up	Check messages in diagnostic buffer
RDY LED flashes red (boot error, CX32 only)		
	• Boot error (e.g. firmware cannot be loaded into the RAM)	Check connection to SIMOTION, check data on CF card, reinstall firmware if necessary
RDY LED flashes yellow, slow flashing (0.5 Hz)		
	• Updating the firmware of the connected DRIVE-CLiQ components	-
RDY LED flashes yellow, fast flashing (2 Hz)		
	• The DRIVE-CLiQ component firmware has been updated; wait for POWER ON of the affected component	Perform a POWER ON of the affected component
DP LED flashes red (CX32-2) **DP1 LED lights up red** (CX32)		
	• Not all of the connected devices are switched on • Cabling fault • CX32/CX32-2 not ready to run (e.g. after switch-on)	Check cabling and configuration settings Check messages in diagnostic buffer

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual

Control Unit CU320-2/CU320

The Control Unit is a central control module in which closed-loop and open-loop functions are implemented for Line Modules and/or Motor Modules.

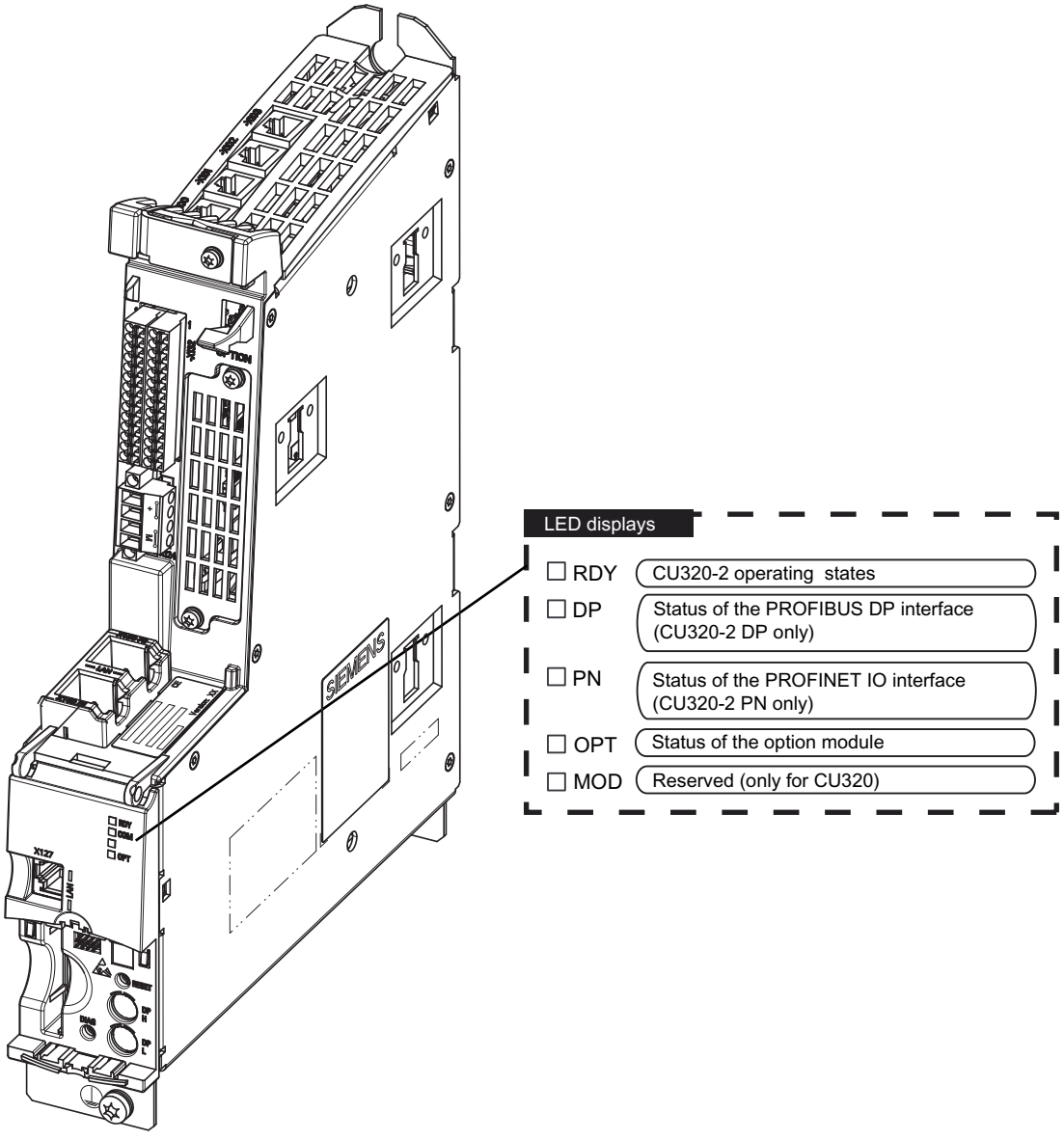


Figure 5-29 LED displays - example of CU320-2 PN

Note

The *SINAMICS S120 Control Units and Supplementary System Components* Manual contains a detailed description of the LED displays, including all possible combinations and flashing frequencies.

Typical faults

Table 5-16 Typical CU320-2/CU320 faults

Fault	Frequent causes	Remedy
RDY LED does not light up (power supply)		
	• The electronic power supply is missing or outside the permissible tolerance range	Check power supply
RDY LED lights up red or flashes red, fast flashing 2 Hz (error on startup)		
	• At least one fault is pending (e.g. RESET, basic system error) • CU320/CU320-2 is ramping up	Check messages in diagnostic buffer
RDY LED flashes red (CU320 only)		
	• Boot error (e.g. firmware cannot be loaded into the RAM)	Check data on CF card, reinstall firmware if necessary
RDY LED flashes yellow, slow flashing (0.5 Hz)		
	• Updating the firmware of the connected DRIVE-CLiQ components	-
RDY LED flashes yellow, fast flashing (2 Hz)		
	• The DRIVE-CLiQ component firmware has been updated; wait for POWER ON of the affected component	Perform a POWER ON of the affected component
DP LED flashes red (CU320-2 DP) **DP1 LED lights up red** (CU320)		
	• Terminating resistor missing or in the wrong place • Not all of the connected devices are switched on • Cabling fault • Incorrect baud rate configured or incorrect baud rate set on a bus node	Check terminating resistor, bus node, cabling, and configuration settings
PN-LED does not light up (CU320-2 PN only)		
	• Cyclic communication has not (yet) taken place.	-
PN-LED flashes green, slow flashing (0.5 Hz) (CU320-2 PN only)		
	• Full cyclic communication has not yet taken place.	-
PN-LED flashes red, slow flashing (0.5 Hz) (CU320-2 PN only)		
	• Bus error, incorrect parameter assignment/configuration	Adapt configuration between controller and devices.
PN-LED flashes red, fast flashing (2 Hz) (CU320-2 PN only)		
	• Cyclic bus communication has been interrupted or could not be established.	Remedy the fault
OPT LED does not light up		

Fault	Frequent causes	Remedy
	• The electronic power supply is missing or outside the permissible tolerance range • Option Board not present • No corresponding drive object created	Check power supply Check Option Board Create drive object
OPT LED lights up red		
	• At least one fault is pending • Option Board not ready to run (e.g. after switch-on)	Check messages in diagnostic buffer

Additional information

- Links in the *Guides* menu of the online help
- *SINAMICS S120 Control Units and Supplementary System Components* Manual

Communication Board CBE20

The device is connected to PROFINET IO using the Communication Board Ethernet CBE20 interface module for SINAMICS S120. The module supports PROFINET IO with isochronous Realtime Ethernet (IRT), PROFINET IO with RT, and standard TCP/IP communication. The Option Board has an X1400 interface with four ports and integrated switch functionality.

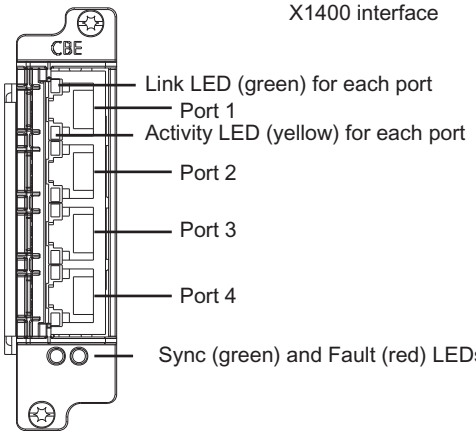


Figure 5-30 Communication Board Ethernet CBE20

Typical errors

Table 5-17 Typical errors

Error	Frequent causes	Remedy
Link LED does not light up		
	• There is no physical connection • The connected device is not switched on	Check the cabling, connectors, and device.
Activity LED does not light up		
	• There is no message frame traffic	If the Link LED is green, use a ping command to check that the system is ready for communication
Fault LED lights up red (bus fault)		
	• No physical connection to a subnet/ switch • Incorrect transmission rate • Full duplex transmission is not activated	Check configuration settings, check IO device, check connection
Fault LED flashes red (bus fault)		
	• Failure of a connected I/O device • At least one of the assigned I/O devices cannot be addressed • Incorrect or no configuration settings	Check configuration settings, check IO device
Sync LED does not light up		
	• If the Link LED is green: Control Unit task system is not synchronized with the IRT cycle clock. An internal substitute cycle clock is generated.	Check configuration settings, check error message in diagnostics buffer

Additional information

- *SINAMICS S120 Control Units and Supplementary System Components* Manual

5.2.4.3 7-segment display

In the case of SIMOTION D4x5-2/D4x5 and P350, a 7-segment display is used to indicate the SIMOTION state.

Meanings of the displays

- 6 - RUNNING
  SIMOTION has ramped up and the cyclic tasks are activated
- 0-5, a-f - internal states (displayed during start-up)
  If a state ≠ 6 is displayed continuously during start-up (i.e. for more than 3 minutes), please analyze the diagnostic buffer. If necessary, contact the hotline to clarify the meaning of the display.
- Flashing point
  Communication is taking place between SIMOTION and SINAMICS Integrated

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Manual

5.2.4.4 Interfaces

SIMOTION devices feature integrated interfaces, to which an HMI device or engineering PC, for example, can be connected for the purpose of reading out diagnostic information. This equipment is connected via either a PROFIBUS or PROFINET/Ethernet interface.

Use the following tools or software for reading out the information:

- SIMOTION SCOUT Engineering System (see Part III: Service with SCOUT Engineering System (Page 3367))
- SIMOTION IT web interface via web browser (see Part II: Service without SCOUT Engineering System (PC-based, SIMOTION IT) (Page 3352))
- HMI devices, e.g. WINCC flexible

Bus analyzer for diagnosing bus or device faults:

- PROFIBUS DP, e.g.
  - SIMATIC ET200 diagnostic repeater
  - SIMATIC diagnostic repeater for PROFIBUS DP
  - BT 200 physical bus test device for PROFIBUS DP (https://support.industry.siemens.com/cs/ww/de/ps/15683/man)
- PROFINET / Ethernet, e.g.
  - BANY PNIO (http://www.industry.siemens.com/industrial-services/it/de/PRODUCTS/DIAGNOSTICS/BANY_PNIO.HTM) for PROFINET
  - Standard network protocol analyzer for Ethernet, e.g. Wireshark (http://www.wireshark.org/)

You will find information on establishing an online connection via an interface in Establishing a connection to the device (without SCOUT Engineering System) (Page 3353) or the individual sections in Going online (Page 3368) (with SCOUT Engineering System).

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Manual
- *SIMOTION D4x5-2* Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Manual
- *SIMOTION D410*-2 Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

5.2.4.5 HMI

An HMI can be connected via the interfaces on the SIMOTION device not only for operator process control purposes, but also to enable diagnostics and display states. See also Interfaces (Page 3342).

In principle, the following message classes are available:

- User error messages
  These are usually messages from the process, containing information on how to resolve the error.
  Irrespective of the HMI system, they are transmitted by a method called bit messaging. The user him/herself assigns the texts to the bit numbers.
  The user can choose to acknowledge the messages via the HMI system or via the application.
  The message texts and their compilation data are located in the HMI.
- Technological alarms and drive messages, technical state alarms (info, warning), and fault messages (alarms) for devices/functions
  These are messages relating to malfuncions or faults, for example, in components such as CPUs or drives. The user can use the ALARM_S procedure from SIMATIC for these messages. ALARM_S is a message number procedure. The message numbers are automatically assigned during configuration in SIMOTION SCOUT. Message texts are uniquely assigned according to their name.
  The messages are called and acknowledged during runtime by means of appropriate system commands.
  The message texts and their compilation data are located in the SIMOTION SCOUT project.
- System messages (e.g. diagnostic buffer)
  These messages cannot be displayed directly in the WinCC flexible message window.
  If you wish to display the messages, you need to send them individually via the ALARM_S message procedure or read them out using the SIMOTION IT web server or the SCOUT engineering tool.

Warning and error messages relating to technology objects are usually output on the HMI and can be acknowledged by the operator.

Examples of Siemens HMI systems:

- Operator panels
- WinCC flexible
- WinCC

Note

Diagnostic buffer entries can also be displayed in a web browser via Ethernet (a PC-based method); see Part II: Service without SCOUT Engineering System (PC-based, SIMOTION IT) (Page 3352).

WinCC flexible

The discrete message procedure can be used to display messages stored on the device (even if these are in more than one language), e.g. information on machine states.

Additionally, S7-compliant message procedures can be used with Alarm_S messages (e.g. information on system errors).

Alarm_S messages are also available in SIMOTION. These messages are configured using the "Message configuration" editor in SIMOTION SCOUT. Alarm_S messages from SIMOTION are processed in WinCC flexible, in the same way as Alarm_S messages from STEP 7.

Special settings need to be made in WinCC flexible and the SIMOTION project in order to use this procedure.

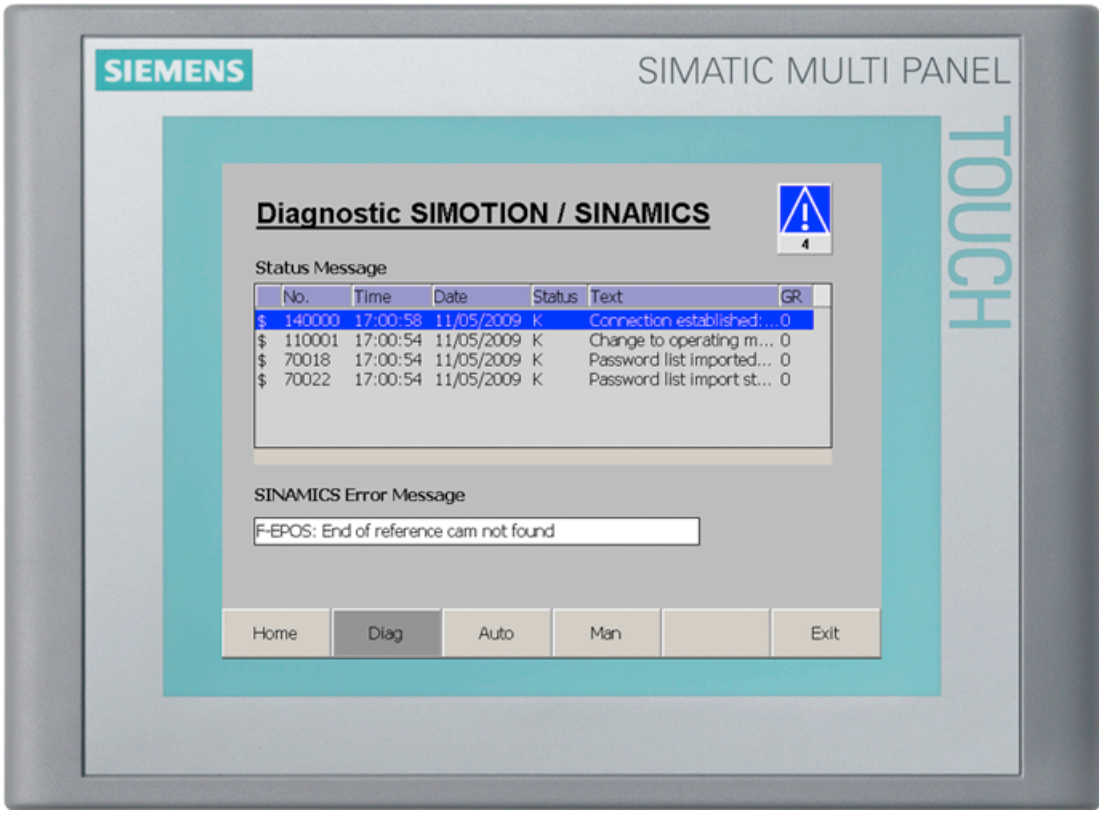


Figure 5-31 Message displays on the operator panel with WinCC flexible

How to activate Alarm_S and SIMOTION messages:

1. In WinCC flexible, open the **Message Settings** tab under **Messages > Settings**.

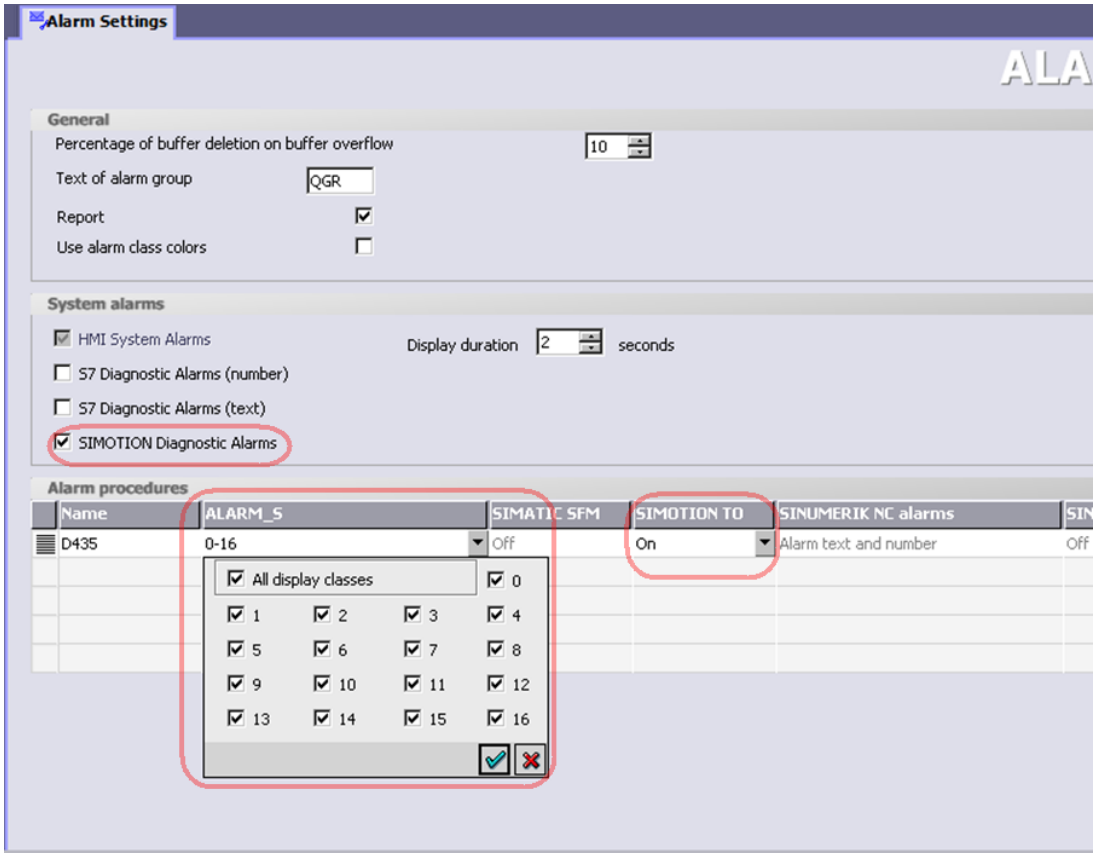


Figure 5-32 Message Settings in WinCC flexible

2. In the **Message procedures** table, activate the Alarm_S messages by assigning them to the message classes (activate the **All display classes** checkbox).
3. In the **Message procedures** table, activate the SIMOTION TO messages using the combo box.

How to activate the display for the Alarm_S and SIMOTION messages:

1. In WinCC flexible, open the **Message View** tab under **Messages > Settings**.

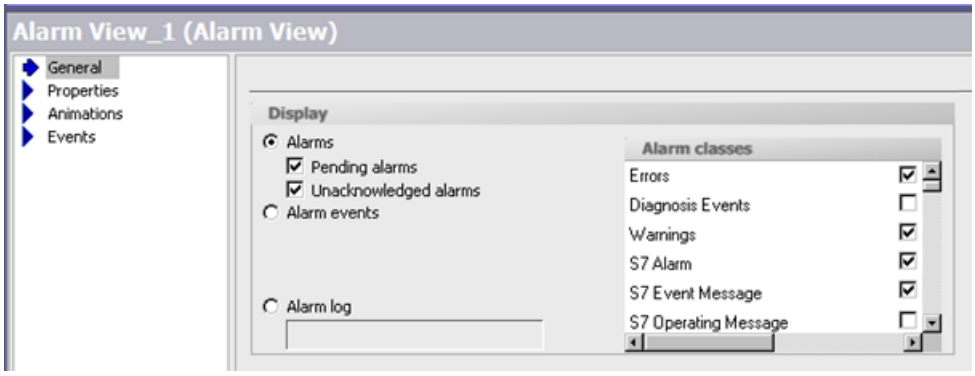


Figure 5-33 Message View dialog in WinCC flexible

2. Under **General**, activate the following message classes:
   - Faults
   - Alarms
   - S7 Message
   - S7 Event Message

How to activate CPU messages in SIMATIC Manager:

- In SIMATIC Manager, open the **CPU Messages** dialog by selecting **Target system > CPU messages...** for the SIMOTION device.
  Select the checkboxes in the **W** (Warning) and **A** (Alarm) columns for the SIMOTION module (see the image below).

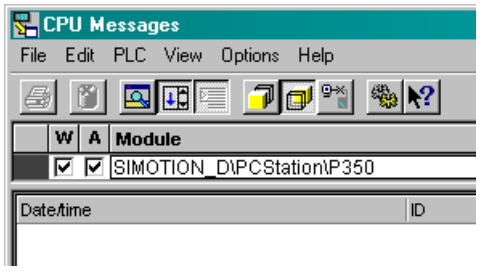


Figure 5-34 CPU Messages dialog in SIMATIC Manager

Additional information

- Links in the *Guides* menu of the online help

See also

Manuals for SIMATIC HMI (WinCC flexible) (https://support.industry.siemens.com/cs/ww/de/ps/14859/man)

5.2.4.6 Backing up diagnostic data and non-volatile data

Following a fault on a SIMOTION device, diagnostic data (e.g. diagnostics buffer content, up-to-date content of websites, etc.) can provide important information on the cause of the fault. With V4.1 SP2 and higher, the user has the option of backing up this data using a simple operation (e.g. a switch position). The data is stored on the CompactFlash Card/MMC in the case of SIMOTION D/C and on the hard disk in the case of SIMOTION P.

The process of backing up and restoring diagnostic data differs according to which SIMOTION platform is being used (C/P/D). SIMOTION D is the only SIMOTION device to feature both a service selector switch and a DIAG button.

This subject is covered in detail for all platforms in Part II, Diagnostic data and non-volatile data (retain data) (Page 3359).

5.2.4.7 Updating devices using the Device Update tool

In SIMOTION SCOUT, the Device Update tool can be used to create update data on the basis of one or more SIMOTION projects (menu **Project > Start Device Update tool**).

The update data created in this way can be uploaded to the SIMOTION device using the CF card/MMC, for example, or also using a USB stick in the case of SIMOTION D4x5-2/D4x5. An Engineering System is not required to do this. You can also update SIMOTION devices via the SIMOTION IT web server Backing up, updating, and restoring device data (Page 3365).

License keys are either retained or may need to be relicensed in the case of function extensions.

Updating does not only mean updating to a higher firmware version, but also involves a project update (for example).

The following update data can be selected and created:

- SIMOTION project (without technology packages)
- Technology packages
- Archive (ZIP file containing SIMOTION SCOUT project)
- User data
- Firmware

The update data contains all the information required for updating or restoring the data on a SIMOTION device.

Note

An up-to-date Firmware Support Package (FWSP) needs to be installed in order to update firmware (V4.1 SP2 and higher). The FWSP is provided with the SIMOTION SCOUT DVD2 under Add-ons and can also be obtained via the Internet at SIMOTION Firmware Support Package (https://support.industry.siemens.com/cs/ww/en/view/33119786).

Restoring refers to the process of reinstating the configuration that was backed up in the SIMOTION device when updating was last performed. If an attempt to update a SIMOTION device fails, for example, the device can be restored to the previous configuration. Restoring is performed via the SIMOTION IT web server and, in the case of SIMOTION D, optionally via the service selector switch.

Saving update data

When creating update data using the Device Update tool, you can select the following:

- Update medium (selection dependent on SIMOTION device)
  - USB stick
  - CF/MMC card
- SIMOTION IT file
- File system (update archive)

If you select the file system option, an update archive containing the **upd_tool.bat** batch file (amongst other things) will be created. By executing the **upd_tool.bat** file, you have the option of installing an update medium on a PC at a later date or creating a SIMOTION IT file. A SIMOTION SCOUT installation is not required on the PC for this purpose.

Transferring update data to the SIMOTION device

The starting point for this is the update media to which the the update data has already been written or a SIMOTION IT file.
An update medium (USB stick or CF/MMC card) is either inserted into the SIMOTION device to be updated, or the update data is transferred to the SIMOTION device via a communication connection (SIMOTION IT).

The update process is triggered by restarting or switching on the SIMOTION device to be updated. Once the update data have been transferred to the memory card in the device for the first time, the data are backed up in the SIMOTION device. (This applies to the USB memory stick and the SIMOTION IT Web server.) The update data is activated as the current configuration after the SIMOTION device has been restarted.

Behavior of the retain data during updating

After the update data has been transferred to the SIMOTION device, the device is restarted and the update data is applied.

During start-up, the non-volatile data that is saved in the SIMOTION device are checked to see if they match the current configuration.

For additional information, please refer to the chapter titled *Behavior of the retain data during updating* in the *Updating SIMOTION Devices* Operating Instructions.

Additional information

- Links in the *Guides* menu of the online help
- *Updating SIMOTION Devices* Operating Instructions

5.2.4.8 Licensing/License key

Overview

Depending on the type and number of RT components used in the project, licenses must be acquired as part of the licensing procedure for SIMOTION. The licenses required for a SIMOTION device are assigned to a hardware component. With SIMOTION C and D they are assigned to the memory cards, and with SIMOTION P to the Communication Board. One or more license keys are generated when the licenses are assigned to the hardware. The license key(s) are saved on the storage medium of the SIMOTION device or in file format during the licensing process.

When replacing the memory card or Communication Board (in the case of P350) or modifying the RT components (e.g. a new technology object), the license key(s) must be redetermined and stored on the virtual memory card.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION SCOUT* Configuration Manual
- PM 21 Catalog: Chapter 8, SIMOTION Runtime Software
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual
- Configurator for SIMOTION runtime licenses in the Mall http://mall.automation.siemens.com (http://mall.automation.siemens.com)

See also

FAQ: Runtime licensing (http://support.automation.siemens.com/WW/view/en/42014324)

FAQ: Displaying the license key (https://support.industry.siemens.com/cs/ww/en/view/42614521)

Checking and amending licensing information

If the system is under-licensed, the SF LED will flash as a warning and an entry will be made in the diagnostics buffer. You can resolve this by licensing the system correctly.

A wizard for the licensing procedure is available in SIMOTION SCOUT. To open the wizard, select the device in the project navigator followed by **Edit > Licenses**.

Note

Before performing licensing, the configuration process should be completed. Once assigned to a hardware component, a license can no longer be used for any purpose other than that for which it has been issued.

Note

The chapter titled "Licensing" in the SIMOTION SCOUT Configuration Manual contains a detailed description of licensing.

You can also find information on licensing under *FAQs > Updating > Licensing SIMOTION runtime functions* in *SIMOTION Utilities & Applications*, which is part of the scope of delivery of SIMOTION SCOUT.

See also

Remedial measures in the event of loss (Page 3351)

Remedial measures in the event of loss

What to do if you lose the license key due to a faulty card:

1. Order a new card through your Siemens contact.
2. Contact the hotline with the serial numbers of the old and new card.
   The hotline will assign the licenses to the new card and give you the new license key, which you can then transfer to the new card.

The same procedure applies to a Communication Board in the context of SIMOTION P.

What to do if you lose the license key:

1. Connect to the Web License Manager (http://www.siemens.com/automation/license) via a browser.
2. Click **Show License Key**.
   The **Show License Key** screen is displayed.
3. Select **Hardware serial number** and enter the serial number of the memory card (taking care to ensure you enter it correctly).
4. Click the **Get License Key** button.
   The license key is displayed.

How to transfer the license key to the memory card:

1. Choose either of the options below for transferring the license key.
   - Copy the license key and enter it via the **Licenses** dialog in SIMOTION SCOUT (CPU context menu).
   - Save the license key as a text file and copy it to the "KEYS\SIMOTION" directory of the memory card using a card reader and the Explorer. This directory is created when the SIMOTION control is ramped up. If it does not yet exist on the memory card, however, you will need to create it using the Explorer.
2. If necessary, generate a license report on all the licenses assigned to the memory card.

Note

The license key is saved in the "KEYS\SIMOTION" directory on the SIMOTION memory card. With SIMOTION V4.1 SP1 and higher, the license key is copied to the boot sector of the card when the control is ramped up. From then on, it is safely backed up (so that it cannot be lost). If the license key is no longer present on the card, it will be written from the boot sector to the "KEYS\SIMOTION" directory again during ramp-up. This means that the system can rectify any instances of the key being deleted accidentally.

Licensing during hardware replacement

When replacing licensed SIMOTION components (MMC, CF, IsoPROFIBUS board, or MCI-PN board), the associated license key must be assigned to the new SIMOTION component. In this case, please contact customer support for assistance.

5.2.5 Part II: Service without SCOUT Engineering System (PC-based, SIMOTION IT)

5.2.5.1 Overview

It is possible to display diagnostic functions in an Internet browser via the Ethernet interface of SIMOTION devices. The connection may be local (direct) or remote (Internet connection).

The SIMOTION IT web server enables direct diagnosis of the SIMOTION devices. Access takes place using a standard browser (e.g. Internet Explorer) via the IP address of the SIMOTION device (e.g. `http://169.254.11.22`). You can use the preconfigured standard diagnostics pages or your own HTML pages for access.

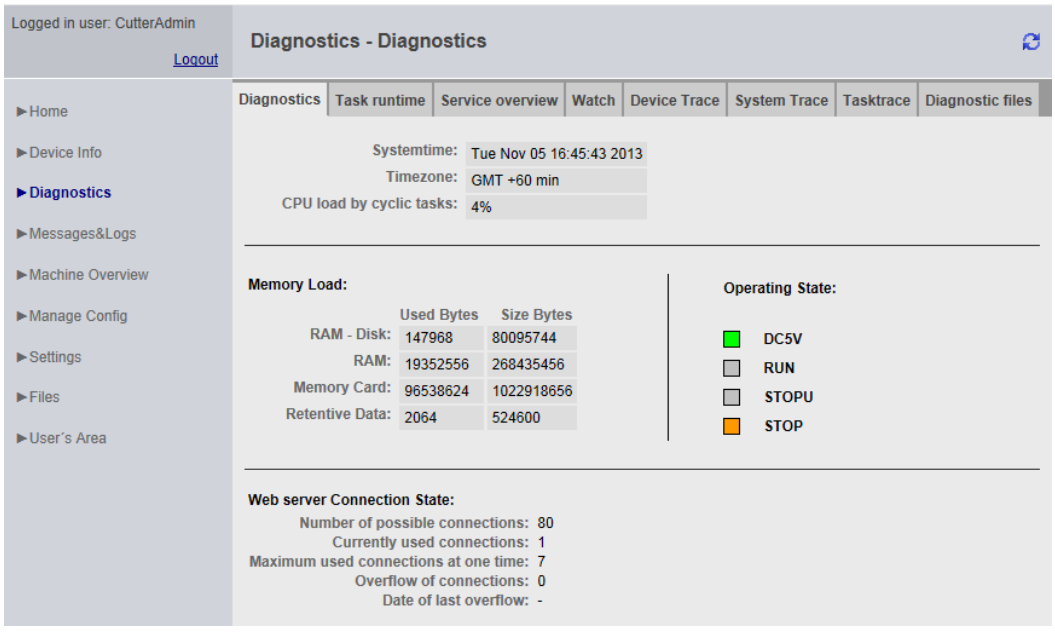


Figure 5-35 SIMOTION IT web server diagnostics

To enable the best possible display of SIMOTION IT web pages on devices such as cell phones or PDAs, a set of special pages is provided for version 4.1 SP4 and higher. You can access these pages via the address `http://<IPAddr>/BASIC`.

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

5.2.5.2 Establishing a connection to the device

An online connection can be established via the Ethernet interface of the SIMOTION device. You can log on to the device directly using a local network or externally via a remote connection (modem, VPN, WLAN, etc.).

If necessary, establish what local settings are in place on your network (firewall, router, etc.) using your network administration facility.

To establish a connection via the Ethernet interface, you need the IP addresses of the nodes. If you do not know these addresses, you can find out what they are by following the instructions below.

How to determine the IP address of the PC:

1. Open the status dialog box of your local area connection by selecting **Start > Settings > Network Connections** and double-clicking the required network connection (e.g. **Local Area Connection**).
   The status dialog box opens.
2. Select the **Support** tab.
   The active IP address and subnet mask are displayed.

How to determine the IP addresses of SIMOTION devices:

You can determine the IP address of a SIMOTION device using the Primary Setup Tool (PST) (https://support.industry.siemens.com/cs/ww/de/view/19440762). Any additional information can be accessed via this Internet link.

In principle, you can use a ping command to check whether a system is ready for communication.

Typical faults

Table 5-18 Typical faults

Fault	Frequent causes	Remedy
Connection not established		
	• IP address of PG does not match SIMOTION IP address	Adapt the PG's Ethernet address so that the PG is in the same subnet as the connected SIMOTION Ethernet interface.
	• SIMOTION devices with multiple Ethernet interfaces: IP addresses of interfaces are in the same subnets, e.g. Ethernet 1: 192.168.214.1 (255.255.255.0) Ethernet 2: 192.168.214.2 (255.255.255.0)	The Ethernet interfaces of a SIMOTION device must be in different subnets
	• With version V4.4 and higher, access to the SIMOTION IT web server is protected by a multi-level security concept. Depending on the security level, you will either have no access or only limited access to the web pages. Authentication may be required for unlimited access.	If web server has not been deactivated: authentication required Access is also possible bypassing the user administration when the service selector switch is in position "8". Access is possible for 120 minutes. After this time has expired, the position of the service selector switch must be changed. This state is also indicated by the LED flashing. (SIMOTION D4xx-2 only)
	• Cabling	Check cabling (crosslink cable!) Patch cables can also be used on SIMOTION PROFINET ports, D4x5-2 Ethernet interfaces, and PCs/switches featuring autocrossing functionality.

Fault	Frequent causes	Remedy
	• Router	If you are using routers, do not select "TCP/IP Auto" as the communication protocol. Reason: Selecting "TCP/IP Auto" automatically assigns an address in the SIMOTION subnet, which means the input port on the router can no longer be accessed.
	• Ports on PG disabled (e.g. in the case of PCs configured by an IT department)	Check default ports (80, 102, etc.)
	• Proxy entered for Internet services In the case of PCs configured by an IT department, the entry for a configuration script can prevent the online connection from being established	Deactivate the checkbox for the **Use automatic configuration script** setting You can find this setting in Windows under **Internet Properties > Connections > LAN settings > Settings...** button.
	• Other	Voltage Off/On on the device

5.2.5.3 Device diagnostics

Overview

You can use SIMOTION IT to display SIMOTION device data on the preconfigured standard diagnostics pages or via variables (Variable Provider).

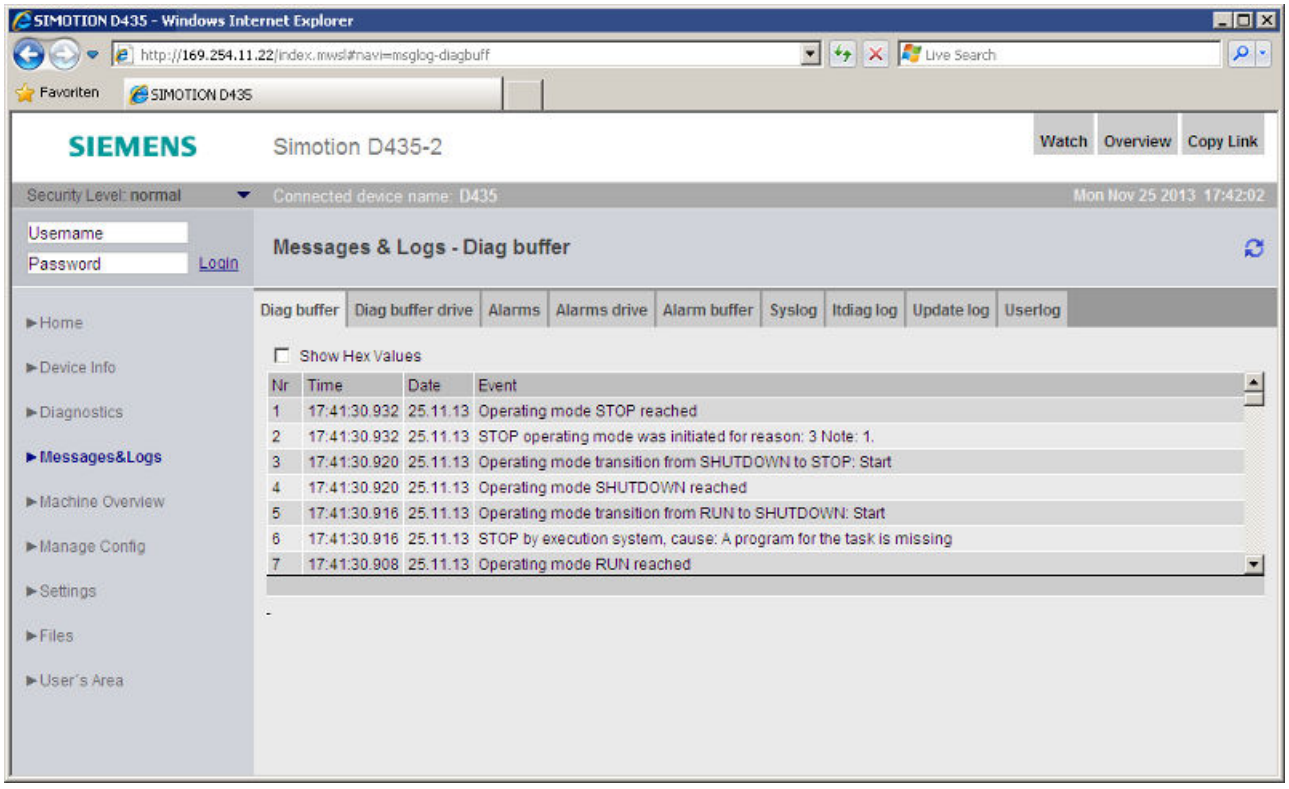


Figure 5-36 Example: SIMOTION IT web server - diagnostic buffer

The information that is available includes the following:

- Operating mode
- Component versions (I&M data)
- Firmware
- MAC and IP address
- Article numbers
- SIMOTION diagnostic buffer
- SINAMICS diagnostic buffer
- Status and runtimes of various tasks
- Memory size and assignment
- Processor utilization
- Current CPU utilization
- SIMOTION alarms
- SINAMICS alarms
- Drive parameters
- TO, I/O, and user variables
- Service overview with axis states

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*
- *Technology Packages Alarms* Diagnostics Manual
- SINAMICS List Manuals

See also

I&M (identification & maintenance) data (Page 3356)

I&M (identification & maintenance) data

I&M (identification & maintenance) information for most configurable PROFIBUS or PROFINET components is available in the module (e.g. SIMATIC ET200 or SINAMICS S120). This information enables service technicians to identify the module immediately in the event of a component failure and order the right version of the spare part required.

The PROFIBUS user organization standardizes the structure of this data and how it is meant to be interpreted (regardless of the component manufacturer) in itsI&M profile (http://www.profibus.com/nc/downloads/downloads/profile-guidelines-part-1-identification-maintenance-functions/display/). The data can be displayed on an HMI system, for example.

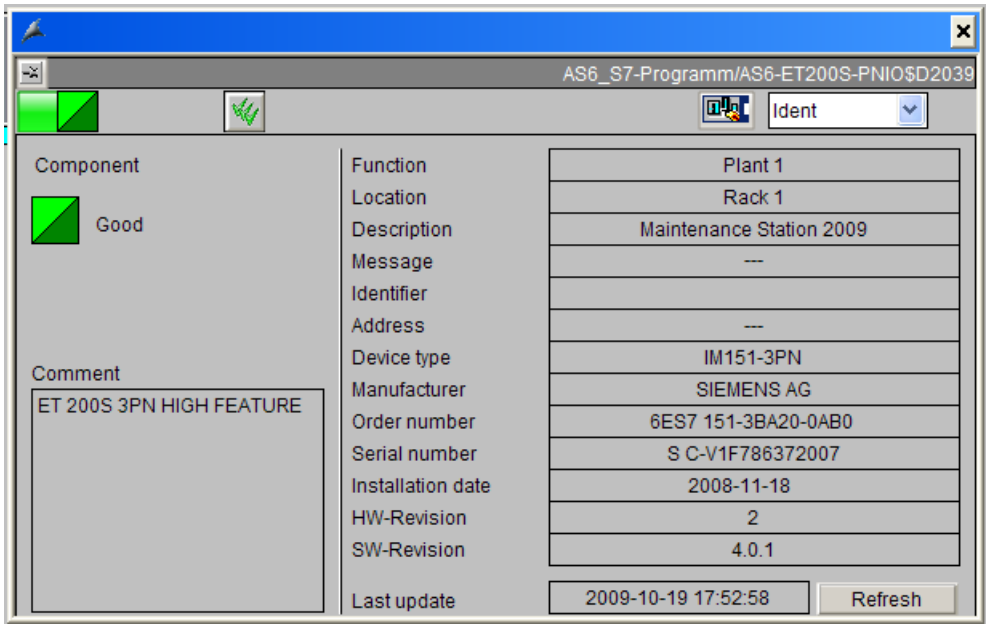


Figure 5-37 Example of I&M data

For SIMOTION, this data is available as a structure (under IM0-Data code) in the list of variables for the device.

Diagnostics buffer

The module states are logged in the diagnostics buffer. The standard site *Diag buffer* shows a list of the diagnostic events that have occurred, in chronological order.

Examples of possible diagnostic events include:

- Faults in a module
- Faults in the process wiring
- System errors in the CPU
- CPU operating mode transitions
- User-defined diagnostic events
- Technology object alarms
- Alarm_S messages
- Errors in the user program
- User-defined entries with the _writeAndSendMessage() function
- Compatibility errors, e.g. between the drive software and SIMOTION (SIMOTION D)

Note

Order of diagnostics buffer entries

The firmware of the devices is based on a real-time capable operating system that is controlled by interrupts and priorities. For this reason, an event can have several diagnostics buffer entries whose time stamps can vary by a few milliseconds. These entries might not be arranged in strict chronological order.

Therefore, when evaluating the diagnostics buffer, the adjacent previous and subsequent entries should also be taken into account.

System utilization

System utilization

You can locate and display information on system utilization using the diagnostics variables of the *SIMOTION diagnostics* variable provider. You will find a detailed description of this in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Tasktrace

The Tasktrace page of the SIMOTION IT web server enables you to set up and control the SIMOTION task trace. The SIMOTION Task Trace supports you when troubleshooting in the SIMOTION multitasking environment. The SIMOTION Task Trace records the sequence of individual tasks, identifies user events that you can generate via a program command, and displays all this information graphically.

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

Analyzing user data

Watch table

To enable variable monitoring, the SIMOTION IT web server provides a watch table with a symbol browser The SIMOTION and drive parameters are displayed in a tree structure. The selected parameters are displayed in a table alongside this.

Trace

To enable variable monitoring, the SIMOTION IT web server provides a variable trace via a web service. Variables can be selected here.

Records can be displayed graphically in the WebTraceViewer for evaluation purposes.

The WebTraceViewer is included with *SIMOTION Utilities & Applications* in SIMOTION V4.1 SP5 and higher, and can also be obtained via account managers (PridaNet); see WebTraceViewer - information (https://support.industry.siemens.com/cs/ww/de/view/31716712). *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

Note

With V4.1 SP5 and higher, the WebTraceViewer is included in the firmware of SIMOTION devices. The standard diagnostics page "Trace" on the SIMOTION IT web server also provides a download link for setting up the WebTraceViewer.

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

5.2.5.4 Diagnostic data and non-volatile data (retain data)

Overview

With version 4.0 and higher, you have the option of backing up diagnostic data. This can be done during operation or start-up. The settings required for this depend on what platform you are using and are described in detail in the following sections and the manuals referred to below.

With SIMOTION D, for example, you can use the service selector switch (V4.1 SP2 and higher) to write diagnostic data and non-volatile data to the CF card. SIMOTION P320-3/P350-3 provides a *diagnostic switch* as part of the *SIMOTION P state* application. The diagnostic data can then be sent to the technical support department (of the machine manufacturer or Siemens) for evaluation.

The information that is backed up includes retain data, diagnostic buffers, alarms, and HTML pages with up-to-date content (snapshots).

The data that is backed up can provide important information after a fault has occurred.

You can use non-volatile data (retain data) in situations where this has not been saved on the data medium using the *_savePersistentMemoryData()* system function and you wish to restore the non-volatile data after a CPU has been replaced.

Table 5-19 Overview of diagnostics options

SIMOTION device	Backing up diagnostic data		Storing diagnostic data	Replacing a CPU without SIMOTION SCOUT
	During start-up	During operation		
C2xx	simotion.ini	Web server	MMC [2)]	Yes
D4x5	Service selector switch simotion.ini	Service selector switch Web server	CompactFlash Card [2)]	Yes
D410-2/D4x5-2	Service selector switch DIAG button simotion.ini	Service selector switch DIAG button Web server	CompactFlash Card [2)]	Yes

SIMOTION device	Backing up diagnostic data		Storing diagnostic data	Replacing a CPU without SIMOTION SCOUT
	During start-up	During operation		
D410 [1]	Service selector switch simotion.ini	Service selector switch Web server[1]	CompactFlash Card [2]	Yes
P320-3/P350	Diagnostic switch simotion.ini	P state Web server	Hard disk	Yes

[1] D410 PN only; not D410 DP (an Ethernet or PROFINET interface is required for the SIMOTION IT web server)

[2] Read out with card reader

With the SIMOTION IT web server, the diagnostic data can be read out as a ZIP file in the *Diagnostic files* dialog box.

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Manual
- *SIMOTION D410*-2 Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

Backing up diagnostic data and non-volatile data

During operation

The advantage of backing up diagnostic data and non-volatile data during operation is that enhanced diagnostic information via HTML pages and TO alarm information are also available.

The HTML pages contain the contents of the standard pages including Watch tables and of user-defined pages with the current variable values.

Initiating the backup process

- **SIMOTION P**
  Via the *SIMOTION P state* application, in menu **Options** > **Set Diagnostic Switch**
- **SIMOTION D**
  Via the service selector switch and, in the case of D410-2/D4x5-2, via the DIAG button too, see Backing up during operation using a service selector switch (Page 3396)
- **SIMOTION IT web server** (all platforms)
  Via the *Diagnostic files* dialog box, see Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*, AUTOHOTSPOT and Backing up during operation using the SIMOTION IT web server (Page 3401)

During start-up

Backing up diagnostic data and non-volatile data during start-up provides you with diagnostic information, but without HTML pages/TO alarm information.

Possible applications for backing up during start-up

- Backing up data of a SIMOTION device that is unable to run/has crashed
- **SIMOTION P**
  If it is not possible to back up data via the SIMOTION IT web server or P state, e.g. in the case of P350 without an operator panel.

Initiating the backup process

- **SIMOTION P**
  By activating the diagnostic switch; see description in appendix
- **SIMOTION D**
  Via the service selector switch and, in the case of D410-2/D4x5-2, via the DIAG button too; see description in appendix
- **INI file** (all platforms)
  An INI file stored on the data medium is detected during start-up and initiates the backup process.

See also Backing up during start-up using a service selector switch (SIMOTION D) or INI file (Page 3399).

See also

Restoring non-volatile data (Page 3365)

Storing diagnostic data and non-volatile data

Diagnostic data and non-volatile data are stored identically for the **SIMOTION C**, **SIMOTION D** and **SIMOTION P** hardware platforms in the following directory:

- `\USER\SIMOTION\HMI\SYSLOG\DIAG`
  On the CompactFlash card or MMC (SIMOTION C / SIMOTION D), CFast card (SIMOTION P)

If necessary, you should provide the relevant technical support department with this information. With a CF card/MMC, diagnostic data is read out via a standard card reader or the standard SIMOTION IT web server pages.

Table 5-20 Diagnostic data and non-volatile data

File	Application
DIAGBUF.TXT	Diagnostic buffer in a simple text format: Numerical values; no specific plain text. A text editor is used for evaluation purposes.
PMEMORY.XML	Non-volatile data (retain data) An operator action can restore the non-volatile data backed up after a CPU has been replaced, see Restoring non-volatile data (Page 3365).
TOALARMS.TXT	Text file containing the pending TO alarms. Only TO IDs, alarm numbers, and auxiliary HEX values. **Note** The TO alarms are only generated if diagnostic data has been created during operation (STOP / STOPU / RUN).
HTML page	If the diagnostic data is backed up, the URLs are requested from the text file (DIAGURLS.TXT) and stored as HTML pages together with their contents, see also Displaying diagnostic data via websites (Page 3362). **Note** The HTML pages are only stored if diagnostic data is created during operation (STOP / STOPU / RUN).
Other files	All other files stored in the directory are only of relevance to technical support.

Note

Use HTML pages if you wish to back up diagnostic data in text format. HTML pages enable user-friendly diagnostics. In addition to the standard SIMOTION IT diagnostics pages, you have the option of creating your own HTML pages (e.g. for the axis status or for machine diagnostics). Customized diagnostics pages or even watch tables stored in the device are particularly suitable for application problems, as you can define the content yourself.

Displaying diagnostic data via websites

In the `DIAGURLS.TXT` text file found in the `...\USER\SIMOTION\HMI\SYSLOG\DIAG` directory, you can specify HTML files whose status is to be stored on the data carrier when diagnostic data is created during operation (e.g. `devinfo.mwsl` must be entered for the `devinfo.htm` HTML page)

Since the pages in question are stored together with their most up-to-date content, this enables archiving of the latest state information regarding the SIMOTION device, as well as the machine/system, from the point at which the diagnostic data was created (e.g. when the service selector switch or DIAG button was activated).

In addition to the standard SIMOTION IT diagnostics pages, it is possible to store customized pages. You can find out how to create pages of this type in, for example, the article *Creating WEB pages for SIMOTION IT* under *SIMOTION IT* in *SIMOTION Utilities & Applications*. *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

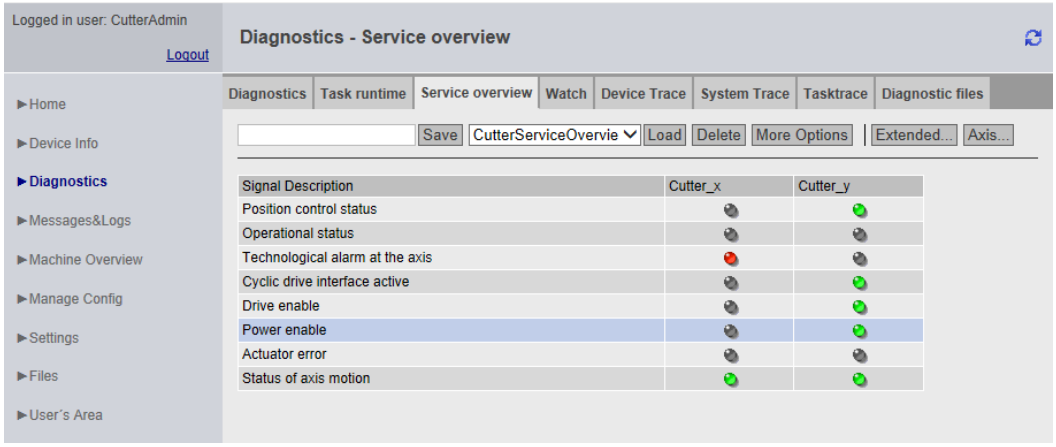


Figure 5-38 Example of service overview

DIAGURLS.TXT

The following points must be noted for the `DIAGURLS.TXT` file:

- A `DIAGURLS.TXT` file containing the standard SIMOTION IT pages is created automatically if you have not saved your own `DIAGURLS.TXT` file.
- Standard SIMOTION IT pages are entered "without" a path specification (e.g. "devinfo.mwsl" for the standard SIMOTION IT page "devinfo.htm").
- Customized SIMOTION IT pages (such as "user.htm") in the `...\USER\SIMOTION\HMI\FILES` directory on the CF card must contain the `FILES/` path specification.
- If you have created subfolders (e.g. "myfolder" in the FILES directory), these must also appear in the path.
- Only 1 file name may be used per line.
- Empty lines are not permitted (an empty line will be interpreted as the end of the list).
- No distinction is made between upper-case and lower-case letters.
- It does not matter whether you use "\" or "/" in the path name.

Backing up diagnostic data and non-volatile data via the SIMOTION IT web server

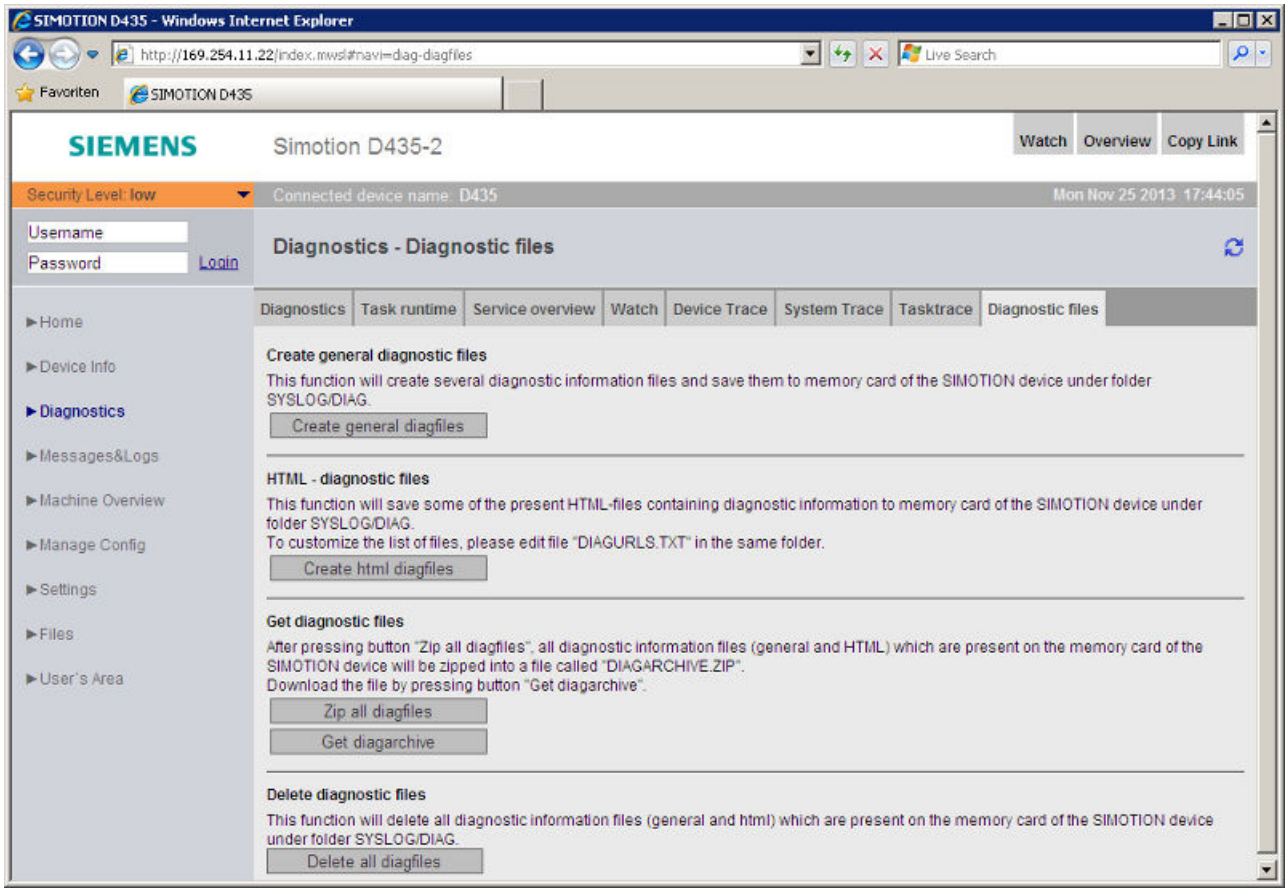


Figure 5-39 Example of diagnostics via an HTML page

Table 5-21 Diagnostic files dialog

Button	Description
Create general diagfiles	Saves diagnostic data and non-volatile data in the `...\USER\SIMOTION\HMI\SYSLOG\DIAG` directory. HTML files used for diagnostics purposes are not saved.
Create html diagfiles	Diagnostics HTML pages are saved on the data medium. It should be noted that only those pages that are listed in the `DIAGURLS.TXT` file in directory `...\USER\SIMOTION\HMI\SYSLOG\DIAG` are saved.
Zip all diagfiles	Stores all files and folders in a ZIP file in directory `...\USER\SIMOTION\HMI\SYSLOG\DIAG`, while retaining the folder structure.
Get diagarchive	The ZIP file is saved on a connected PG/PC.
Delete all diagfiles	Deletes all data stored in directory `...\USER\SIMOTION\HMI\SYSLOG\DIAG`. The directory itself is not removed, however.

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

Restoring non-volatile data

After a CPU has been replaced, the non-volatile data is restored automatically (provided it was saved beforehand).

The data can be saved on the data medium using the **_savePersistentMemoryData()** system function (for example, see the Commissioning Manual AUTOHOTSPOT)

With V4.1 SP2 and higher, it is possible to create non-volatile data on all SIMOTION platforms using a specific operation. The process of backing up this data can be initiated, for example, via the SIMOTION IT web server or, for SIMOTION D, using a service selector switch or DIAG button. See also Backing up diagnostic data and non-volatile data (Page 3360).

The process of restoring data is different for SIMOTION P and SIMOTION C/D. With SIMOTION P, the non-volatile data is restored on the basis of a backup copy on a PC data medium, and with SIMOTION C/D it is restored on the basis of a backup copy on the MMC or CF card.

The appendix contains a detailed description of Restoring non-volatile data (Page 3402) created in this way.

See also

Storing diagnostic data and non-volatile data (Page 3361)

5.2.5.5 Backing up, updating, and restoring device data

Some of the options the *Manage Config* standard page of the SIMOTION IT web server offers are: importing device updates, backing up selected items of data from the device, and restoring the most recent update that was imported.

The **Get selected data** button transfers the currently active device data that is selected to the PC. You can choose from the following data items:

- Firmware
- Technology packages
- Project
- Archive (archived SIMOTION SCOUT project)
- User data
- SIMOTION IT files

The data that was backed up can be reimported into the device.

When you press the **Send update data** button, the update data created with the Device Update tool is loaded onto the memory card in the SIMOTION device. The data for the existing configuration is renamed and backed up automatically, and can be retrieved again at any time. This data is accessed if a restore process is carried out.

Following this, the SIMOTION device is restarted. When the device is booted, the data which has recently been imported is applied as the current configuration and activated.

Note

If no structural change has been made, the retain data are always kept, regardless of the firmware version.

If structural changes are made during the update, the retain data can be saved before updating begins and then transferred again (see SIMOTION SCOUT function "Save and restore variables").

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

See also

Updating devices using the Device Update tool (Page 3348)

5.2.5.6 User-defined service and diagnostics information

SIMOTION IT also offers the option of creating user-defined web pages. Among other things, you can use these additional, individually created web pages to display and change device data that is relevant to your application as well as machine-specific data.

In addition, you can back up these web pages together with their most up-to-date content (snapshots); see also Displaying diagnostic data via websites (Page 3362).

For information on how to create user-defined pages see, for example, the article in *SIMOTION Utilities & Applications* under the category of SIMOTION IT. *SIMOTION Utilities & Applications* is part of the scope of delivery of SIMOTION SCOUT.

Additional information

- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

5.2.6 Part III: Service with SCOUT Engineering System

5.2.6.1 Overview

The SIMOTION SCOUT Engineering System offers a comprehensive range of functions for fault/error diagnostics. These functions are outlined in the sections that follow.

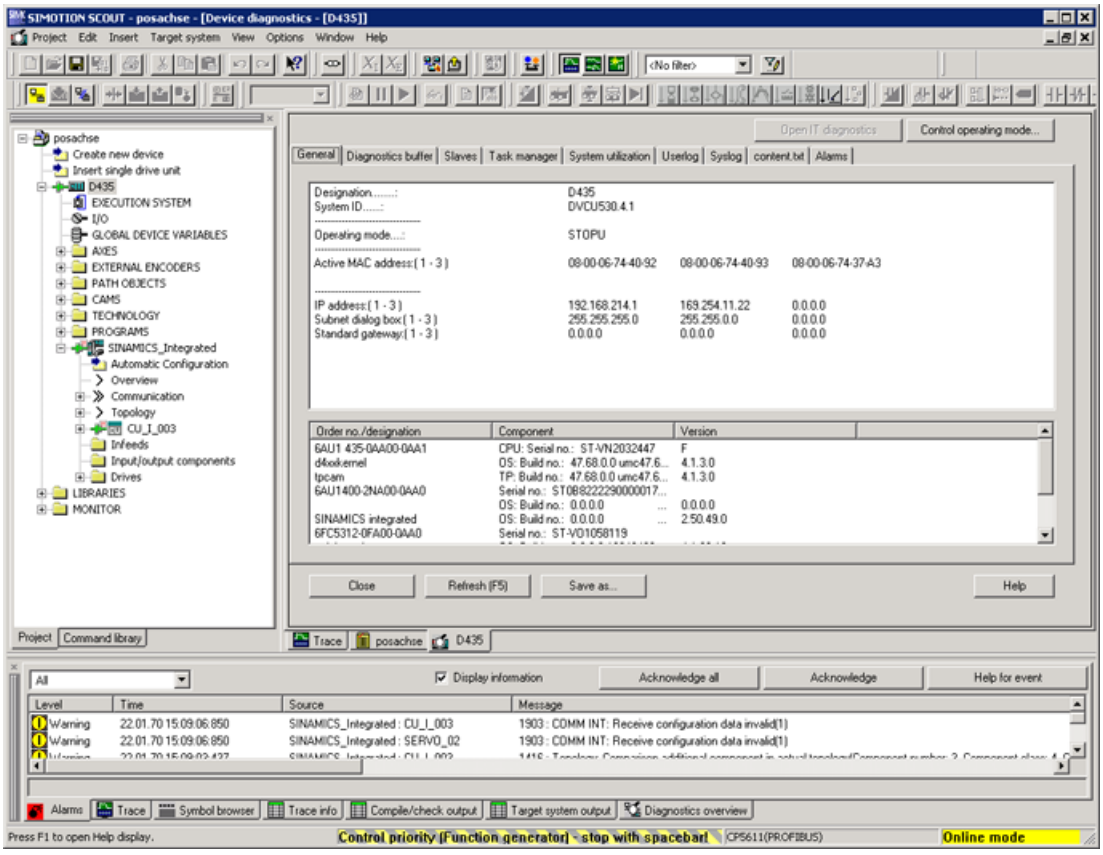


Figure 5-40 Example: SIMOTION SCOUT, Device diagnostics

Use the current project on the machine and the SIMOTION SCOUT software version that is compatible with this. See also Compatibility list (https://support.industry.siemens.com/cs/ww/de/view/36955304).

SIMOTION networks

SIMOTION supports PROFIBUS and PROFINET/Ethernet (TCP/IP). With these network solutions, you can connect the SIMOTION devices in accordance with the requirements of your application.

The sections below draw a distinction between using PROFIBUS and Ethernet/PROFINET for the connection to the SIMOTION device, and between using SIMOTION SCOUT with and without an existing project.

See also

SIMOTION -- Diagnostics -- Analyzing error messages (https://support.industry.siemens.com/cs/ww/de/view/68425516)

5.2.6.2 Going online

Overview

You have the choice of communicating with the control via PROFIBUS DP, PROFINET, or Industrial Ethernet. The S7ONLINE option allows you to communicate with all directly connected controls via the system network specified in the project (e.g. via PROFINET/PROFIBUS/MPI). This access method, which is specified once, enables you to use routing to reach other controls and drives connected to the SIMOTION control via PROFIBUS/PROFINET. This assumes that the nodes support routing.

These options require an online access method to be specified during initial commissioning. To do this, you need to connect the PG/PC interface of the device to the communication interface concerned in NetPro (yellow line in NetPro to PG/PC; see also figures in PROFIBUS (Page 3369) and Ethernet/PROFINET (Page 3372)).

Note

In NetPro, the actual computer being used to create the configuration settings is entered as the PG/PC. If the project is edited using another computer, then the PG/PC needs to be changed to this one. It is recommended that you enter multiple PGs in NetPro if several different persons are taking turns at working on the same project with their PGs.

Note

In addition to the S7Online connection option for the PG/PC interface, with SIMOTION V4.2 and higher you can also set up a second, direct connection to the drive. This enables you to connect STARTER/SIMOTION SCOUT to the device at various locations (e.g. from the control room and directly (locally) on the drive in the workshop) via the interfaces provided, without the need to reconfigure the PG/PC interface.

Note

You will also find a detailed explanation of the procedure in the Internet as FAQ Establishing an online connection to SIMOTION and STEP7 V5.5 SP3 (https://support.industry.siemens.com/cs/ww/en/view/79161896).

PROFIBUS

The PG/PC must be equipped with a PROFIBUS interface for connection purposes.

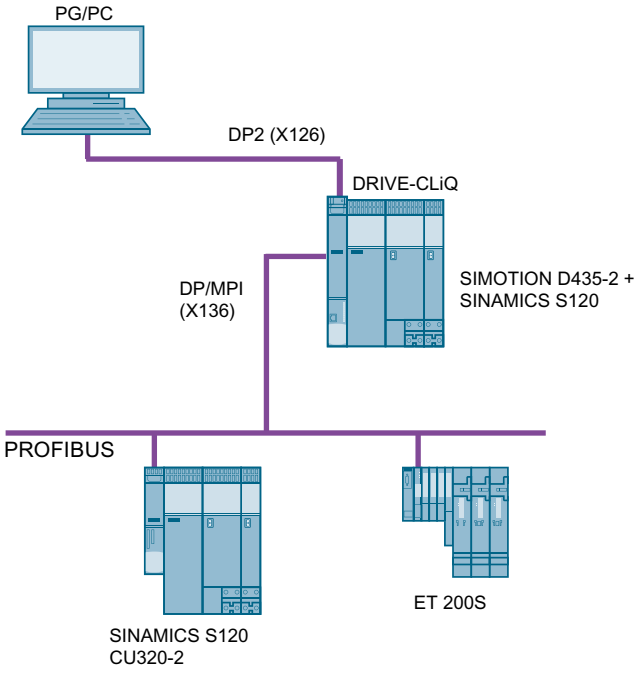


Figure 5-41 Example of a PROFIBUS application

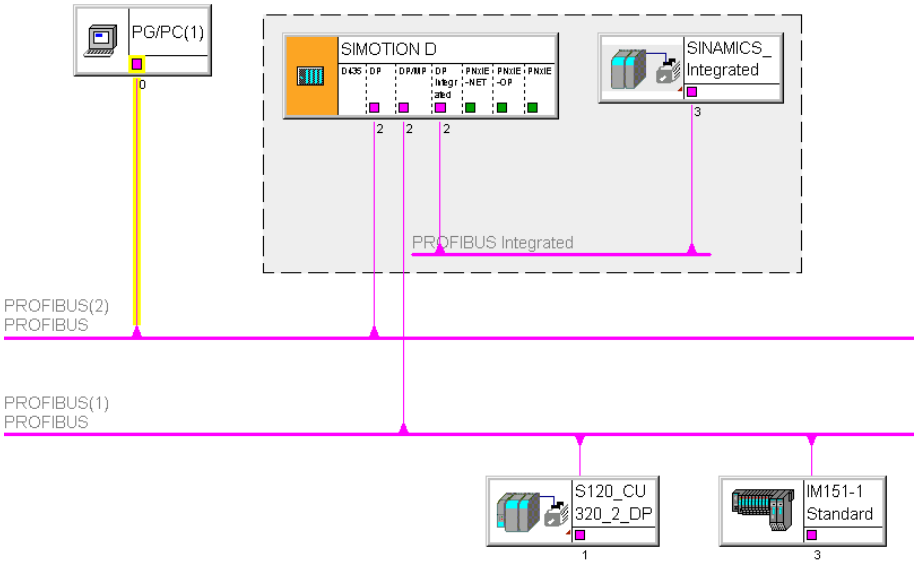


Figure 5-42 Example of a PROFIBUS application - configuration settings in SIMATIC NetPro for PG/PC

Check whether the nodes can be accessed.

In SIMOTION SCOUT, the network settings in menu **Options > Set PG/PC interface...** must match the existing network/interface on the SIMOTION device in order to establish the online connection.

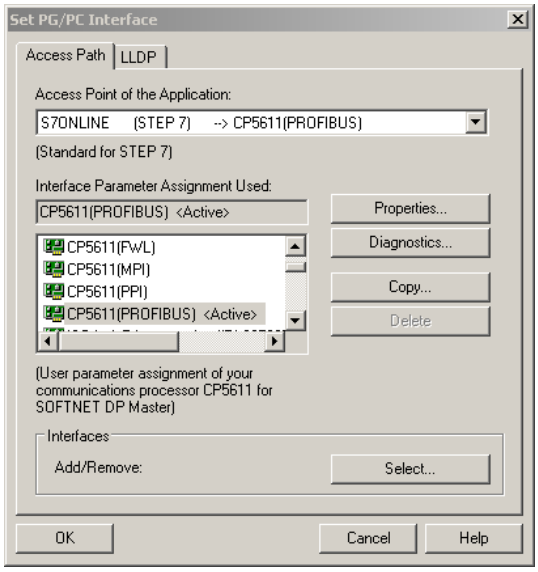


Figure 5-43 Setting the PG/PC interface to S7Online -> PROFIBUS

Pressing the **Diagnostics...** button in the **Set PG/PC Interface** dialog opens a diagnostics dialog, which you can use to check whether the PG/PC interface is functional and whether the right settings have been made for it.

For PROFIBUS, there is also an extended diagnostics option which opens in a separate dialog.

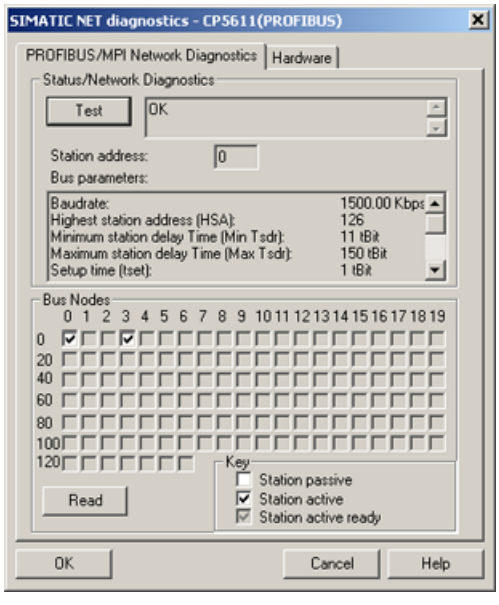


Figure 5-44 SIMATIC NET diagnostics

Table 5-22 Using the SIMATIC NET diagnostics dialog

Button	Description
Test	To check the operating status, click **Test**. If the module is ready to run, **OK** will appear in the field to the right of the button. In this case, the station address is displayed along with the additional current bus parameters and version data.
Read	To display the bus nodes, click **Read**. If the module is ready to run, a list of all the active nodes on the bus will be created. If the module is already communicating and the module supports this function, the list of bus parameters will be created using local information from the module. If the module is not currently communicating or it does not support the creation of a local list, the individual station addresses will be queried via the network. This creates a load on the bus and can take several seconds.

How to identify the bus nodes

1. Use the  button to execute the Accessible nodes function.
   The interfaces, device type, firmware, and addresses that can be used to access the nodes are displayed.

Typical faults

Table 5-23 Typical faults

Fault	Frequent causes	Remedy
Connection via PROFIBUS not possible		
	• Terminating resistor missing or in the wrong place • Not all of the connected devices are switched on • Cabling fault • Incorrect baud rate configured or incorrect baud rate set on a bus node	Check terminating resistor, bus nodes, cabling, and configuration settings, and check whether any PROFIBUS addresses have been assigned more than once. A diagnostic repeater installed in the network may provide additional diagnostic information.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION SCOUT Communication* System Manual

See also

PG/PCs and CPs - Manuals and Operating Instructions (https://support.industry.siemens.com/cs/ww/de/ps/15683/man)

Ethernet/PROFINET

The PG/PC must be equipped with a standard Ethernet interface.
Since PROFINET is based on Industrial Ethernet, a PROFINET network is accessed using the same mechanisms as those used for an Ethernet network.

Note

If the integrated ports on the SIMOTION devices are not enough for your requirements, you can use an external SCALANCE switch. If PROFINET IO with IRT communication is to be enabled downstream of the switch, a suitable switch (which supports IRT) must be used.

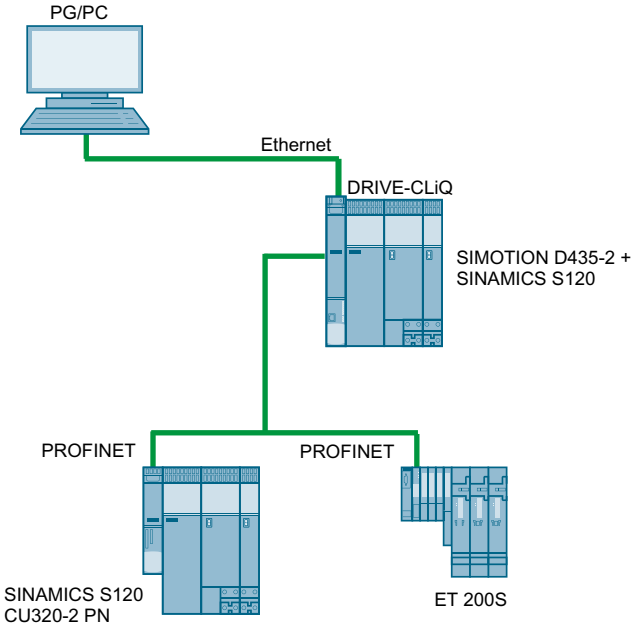


Figure 5-45 Example of an Ethernet/PROFINET application

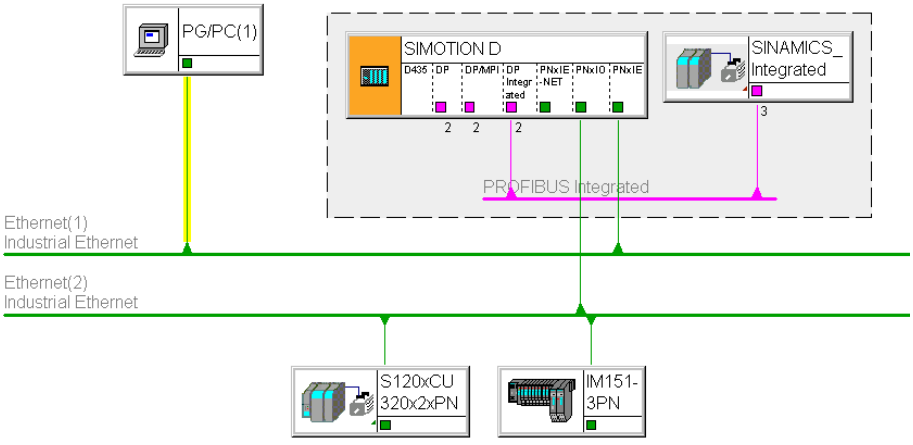


Figure 5-46 Example of an Ethernet/PROFINET application - configuration settings in NetPro

Note

SIMOTION supports internal routing as of V4.1 SP2. This, for example, makes connections from Ethernet (e.g. PN/IE X127) to PROFINET (PN-IO X150) possible.

Check whether the nodes can be accessed.

In SIMOTION SCOUT, the network settings in menu **Options > Set PG/PC interface...** must match the existing network/interface on the SIMOTION device in order to establish the online connection.

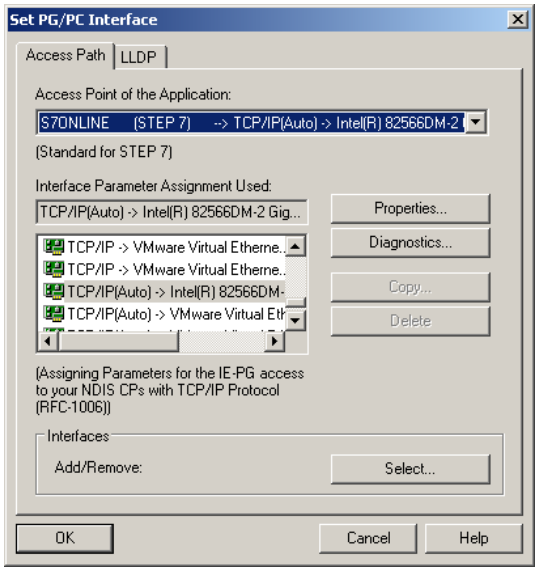


Figure 5-47 Setting the PG/PC interface to S7Online -> TCP/IP

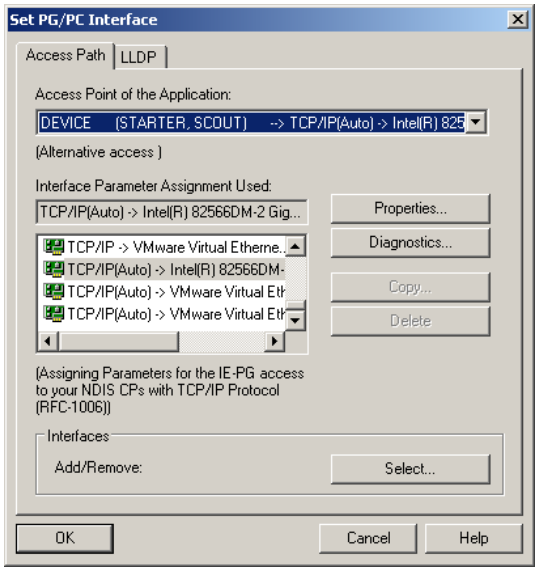


Figure 5-48 Setting the PG/PC interface to DEVICE -> TCP/IP (SIMOTION V4.2 and higher)

Setting the interface to DEVICE (SIMOTION V4.2 and higher) enables different PGs to be used, for example, without the need to reconfigure the PG/PC interface. Additional information can also be found via the list of links for this section in the relevant online help menu.

Pressing the **Diagnostics...** button in the **Set PG/PC Interface** dialog opens a diagnostics dialog, which you can use to check whether the PG/PC interface is functional and whether the right settings have been made for it. The Diagnosis Hardware function or SR protocol does not check the connection to the SIMOTION device, only whether the interface is suitable for establishing an S7Online connection.

To establish a connection via the Ethernet interface, you need the IP addresses of the nodes. If you do not know these addresses, you can find out what they are by following the instructions below.

How to determine the IP address of the PC:

1. Open the status dialog of your local area connection by selecting **Start > Settings > Network Connections** and double-clicking **Local Area Connection**.
   The **Local Area Connection Status** dialog opens.
2. Select the **Support** tab.
   The active IP address and subnet mask are displayed.

How to identify the IP addresses of the SIMOTION devices (example without a project):

1. Use the button to execute the Accessible nodes function.
   The interfaces, device type, firmware, and addresses that can be used to access the nodes are displayed. If nodes are present in other subnets, only the IP address will be displayed and a relevant message will be output.

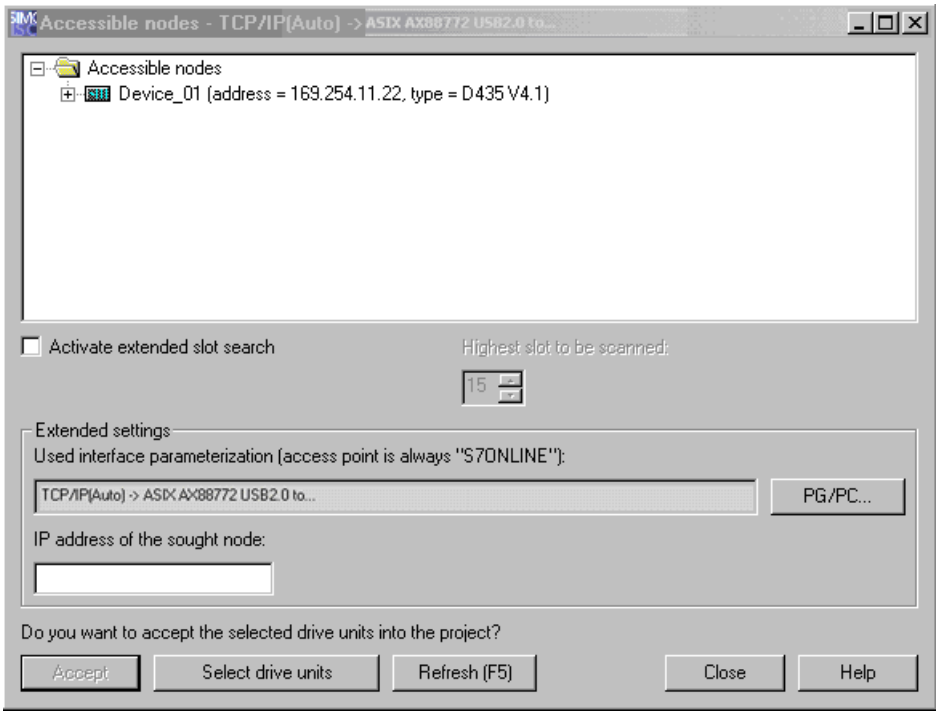


Figure 5-49 Accessible nodes dialog - example of Ethernet node

How to determine the IP addresses of the SIMOTION devices (example with a project):

1. Open HW Config by double-clicking the SIMOTION device in SIMOTION SCOUT.
2. In HW Config, select **Target system > Ethernet > Edit Ethernet node...** .
   The **Edit Ethernet node** dialog opens.
3. Start the search using the **Browse...** button.
   The devices which can be accessed via the unique MAC addresses are identified and displayed.
4. Pressing the **Flashing** button causes a flash telegram to be transmitted for the selected module. In turn, this causes the module to display a signal on the interface being addressed. Please refer to the device documentation to find out which display element is used for signaling (e.g. the SF LED flashes at a fast frequency with SIMOTION D4x5-2).

Note

Make sure the IP addresses are unique on the network before you go online with your PG/PC. Identical IP addresses on the network may cause communication and operating faults.

Contact your network administrator if you do not have all the information you need for network operation.

Typical faults

Table 5-24 Typical faults

Fault	Frequent causes	Remedy
Connection via Ethernet /PROFINET not possible		
	• IP address of PG does not match SIMOTION address	The PG/PC must be in the same subnet as the SIMOTION device
	• IP address assigned twice	Adapt the Ethernet address of a node (PG)
	• Cabling	Check cabling (crosslink cable) Patch cables can also be used on SIMOTION PROFINET ports, D4x5-2 Ethernet interfaces, and PCs/switches featuring autocrossing functionality.
	• Other	• Voltage Off/On on the device • If the Link LED is green, use a ping command to check that the system is ready for communication
See also typical errors/faults for the LED displays (Page 3322)		

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION SCOUT Communication* System Manual

Going online without a user project

Configuration settings cannot be changed or extended unless a user project matching the project in the SIMOTION device is available on the PG/PC. However, the section below describes how to establish an online connection to the device even if you do not have this, so that you can read device diagnostics or determine the firmware version of the devices, for example.

The process used to establish the connection depends on the interface being used and is described in the following sections:

- PROFIBUS (Page 3369)
- Ethernet/PROFINET (Page 3372)

Once you have successfully identified the nodes using the Accessible nodes function, in this same dialog you can right-click the required device to select it and then call up the device diagnostics (for example) via the context menu.

The diagnostics options of the SIMOTION IT web server (Page 3352) are also available.

See also

Device diagnostics (Page 3382)

Going online with a user project

Checking the PG/PC assignment

A project contains a PG/PC assignment configured in NetPro. This PG/PC assignment specifies which PG/PC interface is used to access a specific interface of the SIMOTION device.

You also need to take into account the interface settings in the SIMOTION device.

How to check whether a PG/PC assignment is active in the project:

1. Open the SIMOTION project.
2. Open NetPro using the  button.
   The network configuration settings for the project are displayed.
3. A PG/PC assignment is active if the following apply:
   - A PG/PC block is present
   - A connection to the SIMOTION device has been created
   - The connection from the PG/PC to the network is marked in yellow

Going online

1. Open HW Config by double-clicking the SIMOTION device in SIMOTION SCOUT.
2. In HW Config, select **Target system > Ethernet > Edit Ethernet node...** .
   The **Edit Ethernet node** dialog opens.
3. Start the search using the **Browse...** button.
   The devices which can be accessed via the unique MAC addresses are identified and displayed.
4. Pressing the Flashing button causes a flash telegram to be transmitted for the selected module. In turn, this causes the module to display a signal on the interface being addressed. Please refer to the device documentation to find out which display element is used for signaling (e.g. the SF LED flashes at a fast frequency with SIMOTION D4x5-2).

Note

Make sure the IP addresses are unique on the network before you go online with your PG/PC. Identical IP addresses on the network may cause communication and operating faults.

Contact your network administrator if you do not have all the information you need for network operation.

Typical faults

Table 5-25 Typical faults

Fault	Frequent causes	Remedy
Not possible to connect using the accessible nodes		
	• NetPro settings not made or incorrect	Check the NetPro settings. Multiple PG/PC stations can also be created. The connection from the assigned, active PG/PC to the network is shown in yellow.
	• Configured firmware version in the project is different to the firmware version in the device	Check the entries in the SIMOTION SCOUT message window (incorrect firmware message) Check the versions of SIMOTION SCOUT and the device firmware. You can find out which device firmware version requires which SIMOTION SCOUT version by consulting the Compatibility list (https://support.industry.siemens.com/cs/ww/de/view/36955304) or contacting the hotline.
	• Not possible to go online even though NetPro settings have been configured and firmware is correct	Check whether the cable is also connected to the interface configured in NetPro (e.g. PN/IE X127). Notice: The SIMOTION device can still be accessed via Accessible nodes.
Connection has been established, but project is not consistent with online data (red symbols in project navigator)		
	• SIMOTION inconsistent	Determine where exactly the differences lie using the Project comparison (Page 3389) function, and align the data
	• SINAMICS inconsistent	Determine where exactly the differences lie using the Project comparison (Page 3389) function, and align the data
Connection has been established, but downloading is not possible		
	• Causes are displayed on the Target system output tab in SIMOTION SCOUT • Too little memory with P320-3/ P350-3	Check the information on the Target system output tab Allocate more memory space in the SIMOTION P control manager

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION SCOUT Communication* System Manual
- *Project comparison* Function Manual

See also

Testing programs (Page 3394)

Device diagnostics (Page 3382)

Factory settings for SIMOTION devices

Factory settings for interfaces

The factory settings for the interfaces are outlined in the Commissioning Manuals. In the online help, you can access the relevant sections directly via the links for this section (*Guides* menu).

Table 5-26 Factory settings for SIMOTION devices

Interface	Setting
C2xx	
Ethernet PNxIE X7	169.254.11.22 (subnet 255.255.0.0)
PROFINET PNxIO X11 (only C240 PN)	192.168.0.1 (subnet 255.255.255.0)
PROFIBUS DP1 X8	2 (baud rate 1.5 Mbit/s)
PROFIBUS DP2/MPI X9	2 (baud rate 1.5 Mbit/s)
D4x5	
Ethernet IE1-OP X120	192.168.214.1 (subnet 255.255.255.0)
Ethernet IE2-NET X130	169.254.11.22 (subnet 255.255.0.0)
PROFIBUS DP1 X126	2 (baud rate 1.5 Mbit/s)
PROFIBUS DP2/MPI X136	2 (baud rate 1.5 Mbit/s)
D4x5-2	
Ethernet PNxIE X127	169.254.11.22 (subnet 255.255.0.0)
Ethernet PNxIE-NET X130	192.168.2.1 (subnet 255.255.255.0)
Ethernet PNxIE-OP X120 (only D4x5-2 DP)	192.168.213.1 (subnet 255.255.255.0)
PROFINET PNxIO X150 (only D4x5-2 DP/PN)	192.168.1.1 (subnet 255.255.255.0)
PROFIBUS DP X126	2 (baud rate 1.5 Mbit/s)
PROFIBUS DP/MPI X136	2 (baud rate 1.5 Mbit/s)
D410 DP	
PROFIBUS DP/MPI X21	2 (baud rate 1.5 Mbit/s)
D410 PN	
PROFINET PN-IO X200	192.168.0.1 (subnet 255.255.255.0)
D410-2 DP	
PROFIBUS X21 DP/MPI	2 (baud rate 1.5 Mbit/s)
PROFIBUS X24 DP	2 (baud rate 1.5 Mbit/s)
Ethernet PNxIE X127	169.254.11.22 (subnet 255.255.0.0)
D410-2 PN	
Ethernet PN/IE X127	169.254.11.22 (subnet 255.255.0.0)
PROFIBUS DP/MPI X21	2 (baud rate: 1.5 Mbits/s)
PROFINET PNxIO X150	192.168.1.1 (subnet 255.255.255.0)
P320-3	
Ethernet PNxIE X2	169.254.11.21 (subnet 255.255.0.0)
PROFINET PNxIO X3	192.168.0.1 (subnet 255.255.255.0)

Interface	Setting
P350-3	
Ethernet IE1 ETH1	169.254.11.21 (subnet 255.255.0.0)
Ethernet IE2 ETH2	192.168.214.1 (subnet 255.255.255.0)
PROFIBUS X101 DP (IsoPROFIBUS board)	2 (baud rate 1.5 Mbit/s)
PROFIBUS X102 DP/MPI (IsoPROFIBUS board)	2 (baud rate 1.5 Mbit/s)
PROFINET X21 (MCI-PN board)	192.168.0.1 (subnet 255.255.255.0)
CBE30-2/CBE30	
PROFINET X1400	192.168.0.1 (subnet 255.255.255.0)

Note

PROFINET addresses are only assigned by means of a project download; see also Ethernet/ PROFINET (Page 3372).

Restoring the factory settings

If there is definitely nothing wrong with the connection and it is not possible to download anything, you can restore the factory settings on the device.

NOTICE

Data loss

This will cause all data to be lost, with the exception of the SIMOTION kernel and licenses. For this reason, you should back up the non-volatile data before you do this. See also Backing up diagnostic data and non-volatile data (Page 3360).

First, the device should be restored to its delivery state (factory settings). The method for establishing the delivery state differs for the SIMOTION devices.

The Commissioning Manuals and Operating Instructions describe the process of restoring the factory settings. In the online help, you can access the relevant sections directly via the links for this section (*Guides* menu).

Note

The existing IP settings for the CBE30-2/CBE30 are restored to the factory settings.

Once the factory settings have been established, transfer the project and the previously backed-up non-volatile data to the device (see also Restoring non-volatile data (Page 3365)).

Further information

- Links in the *Guides* menu of the online help
- *SIMOTION C* Operating Instructions

- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

See also

Establishing a connection to the device (Page 3353)

5.2.6.3 Device diagnostics

Overview

The device diagnostics provide an overview containing operating information about the system and the individual devices in the project. The device diagnostics are only available in ONLINE mode and, in the case of SINAMICS devices, as of SINAMICS firmware version V2.5 SP1. SINAMICS device messages can be found in the diagnostic buffer, starting from the entry `>>>>>> SINAMICS Integrated: Start diagnostic buffer, station address = x <<<<<<`.

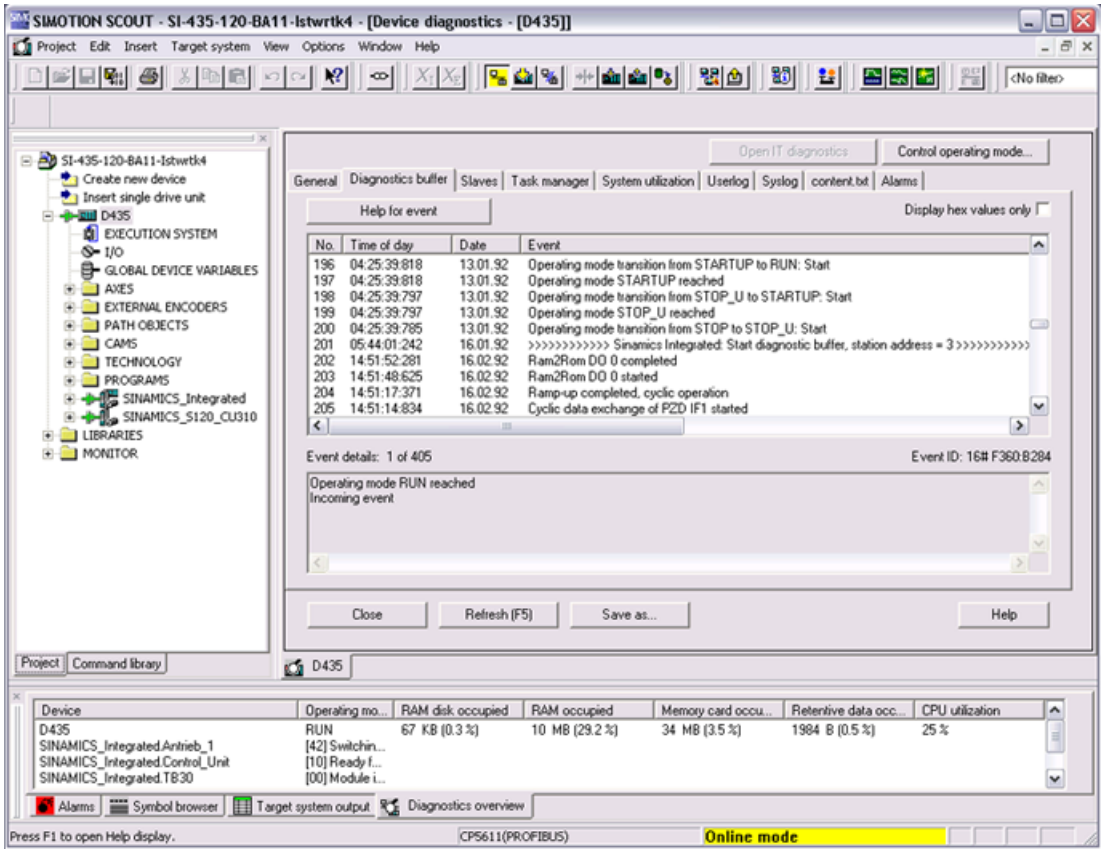


Figure 5-50 Example: D435 diagnostic buffer

In addition to the individual tabs, there are also buttons for accessing other functions, such as **Open IT diagnostics**.

Note

As well as the fault and alarm buffer, there is also a buffer for safety messages in the case of SINAMICS S120 (firmware version V2.5 SP1 and higher). These are displayed on the **Alarms** tab in the detail view.

Table 5-27 Functions that can be called up using buttons

Button	Function
Open IT diagnostics	When an Ethernet connection is present, this opens the Ethernet-based HMI and diagnostic function in an Internet browser. In addition to customized web pages and comprehensive device/diagnostic information, one of the options that the SIMOTION IT web server offers is the ability to perform firmware and project updates using an Internet browser.
Control operating mode...	The current operating mode of the SIMOTION device is displayed in the dialog. You can change the operating mode in ONLINE mode depending on the position of the mode switch.

To open the device diagnostics:

1. Select **Project > Connect to target system**.
   The PC/PG is connected to the target system.
2. Select the device in the project navigator or on the **Diagnostics overview** tab in the detail view.
3. Select **Target system > Device diagnostics**.
   The **device diagnostics** are displayed in the working area.

Note

When evaluating messages, it is beneficial if the SIMOTION and SINAMICS times of day are synchronized. With V4.2 and lower, times of day must be synchronized via the application. With V4.2 and higher, however, they can be synchronized in the SIMOTION SCOUT Engineering System, via the CPU context menu: **Setting on the device > Perform time synchronization with SINAMICS drive units**.

For application-based time-of-day synchronization, you need to use the **Time-of-day Synchronization (SIMOTION -> SINAMICS)** block from the LDPV1 program library. You can find this program library in *SIMOTION Utilities & Applications*, which is part of the scope of delivery of SIMOTION SCOUT. If necessary, get in touch with your Siemens contact.

You can use the **Save as...** button to save the data as a text file so that you can evaluate it offline.

Information and options provided by device diagnostics

- **General**
  General information on the SIMOTION device and versions
  - Device designation
  - Operating mode
  - MAC address and IP address
  - Components with article number/designation and version (I&M data)
  - SIMOTION version
  - BIOS version
  - Versions of SINAMICS Control Unit
  - Internal version/stamp
- **Diagnostic buffer**
  Shows the logged module states together with the diagnostic events that have occurred, in list format
- **Slaves**
  Shows the devices configured as slaves in HW Config, together with addresses and states
- **Task manager**
  Shows status and runtime of tasks created in the project; MotionTasks can also be controlled
- **System utilization**
  Information on memory and CPU
- **Userlog**
  Option of entering user information that is stored in the target system. This enables information about changes to the SIMOTION system to be documented, for example. Since the text is stored on the device, this information is always available regardless of which PG/PC or offline project is being used
- **Syslog**
  Shows logged ROM actions such as Copy RAM to ROM
- **content.txt**
  Shows the SIMOTION and SINAMICS firmware components on the card, as well as the card version
- **Alarms**
  Shows pending alarms (TOs and DOs). The alarms can be acknowledged.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION SCOUT* Configuration Manual

See also

I&M (identification & maintenance) data (Page 3356)

Diagnostics buffer

The module states are logged in the diagnostics buffer. The Diagnostics buffer tab shows a list of the diagnostic events that have occurred, in chronological order.

Examples of possible diagnostic events include:

- Faults in a module
- Faults in the process wiring
- System errors in the CPU
- CPU operating mode transitions
- User-defined diagnostic events
- Technology object alarms
- Alarm_S messages
- Errors in the user program
- User-defined entries with the _writeAndSendMessage() function
- Compatibility errors, e.g. between the drive software and SIMOTION (SIMOTION D)

Note

Sequence of the diagnostic buffer entries

The firmware of the devices is based on a real-time capable operating system that is controlled by interrupts and priorities. For this reason, an event can have several diagnostics buffer entries whose time stamps can vary by a few milliseconds. These entries might not be arranged in strict chronological order.

Therefore, when evaluating the diagnostics buffer, the adjacent previous and subsequent entries should also be taken into account.

Note

Number of diagnostic buffer entries

An event can result in several diagnostic buffer entries. For this reason, identical entries can occur in the diagnostic buffer for one event. This can be the case particularly for errors in the communication (e.g. failure of a device of connection interrupted). Note therefore that conclusions cannot be drawn on the number of diagnostic events through the number of identical diagnostic buffer entries. As a rule, only one error is the trigger for diagnostic events in this case.

System utilization

The System utilization tab displays the current memory assignment, free memory, and CPU utilization.

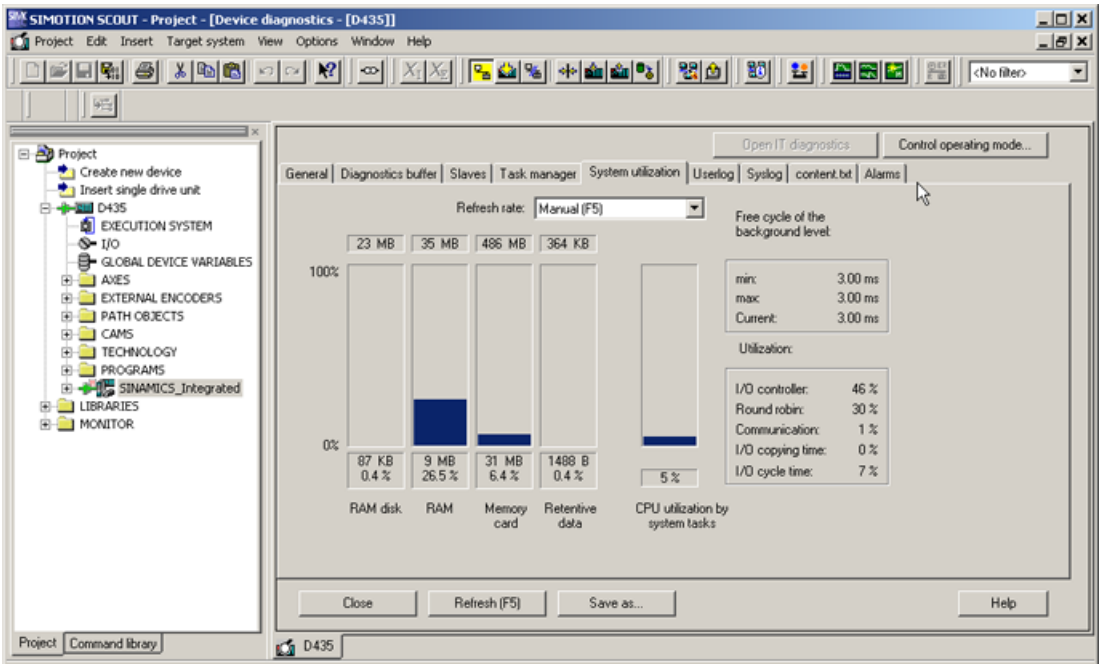


Figure 5-51 System utilization

Here, you can check the system utilization and adjust the system settings in line with the individual software configuration settings and user programs in the SIMOTION tasks. Adjustable gear ratios between the bus task, servo, and IPO support optimum load distribution and system utilization.

With SIMOTION P320-3/P350-3, you can check the CPU utilization in the *SIMOTION P state* application.

Userlog/Syslog

Userlog

The user log offers the option of entering user information that is stored in the target system. This enables information about changes to the SIMOTION system to be documented, for example. Since the text is stored on the device, this information is always available regardless of which PG/PC or offline project is being used

Syslog

For the SIMOTION CPU, the Syslog tab provides a syslog file for device diagnostics. This file contains all the actions that change a non-volatile configuration on the memory card.

The following actions are entered in the syslog file:

- Copy RAM to ROM
- Memory reset
- Formatting the card using the mode switch

content.txt

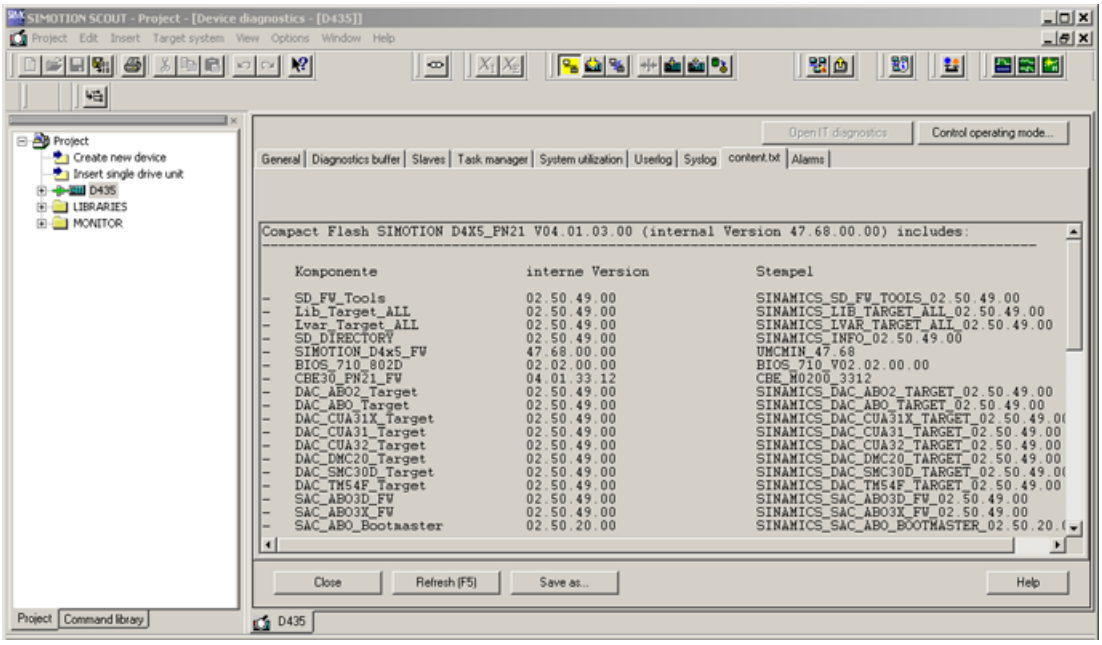


Figure 5-52 content.txt - example SIMOTION D

All the SIMOTION and SINAMICS firmware components available on the card are displayed on the content.txt tab of the device diagnostics. In addition to the card version, each individual firmware component is displayed together with its internal version code; this information can be saved in file format using the **Save as...** button. This file can be sent via e-mail for support purposes, for example.

This information will be required if the hotline needs to be contacted.

Note

It is available with SIMOTION V4.1 SP2 and higher, but not with older device types such as C230-2 or P350-2.

Service overview

In online mode, the service overview shows a complete overview of all configured axes in the project, in tabular format. The current state (including values from system variables) is displayed along with error/fault states.

The service overview is called up via the **Target system > Service overview** menu.

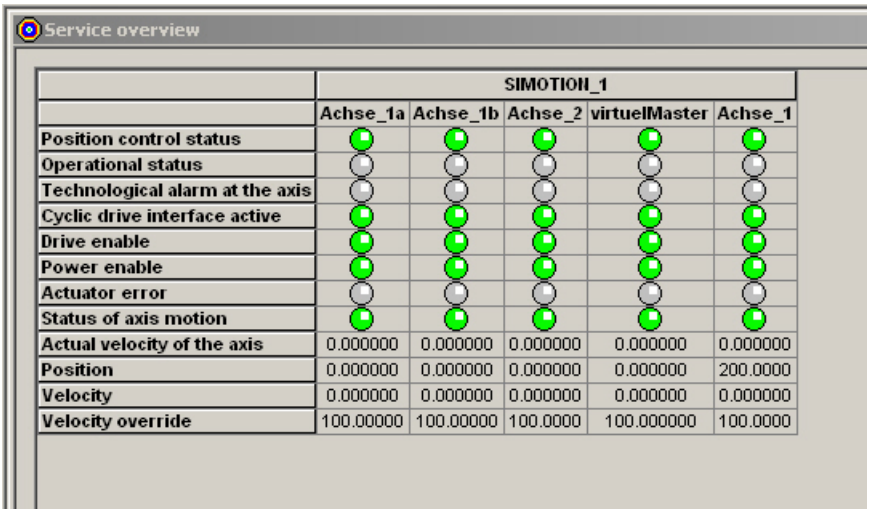


Figure 5-53 Service overview

Meanings of lamps

- Green
  Axis (system variable) is active/on or axis is stopped
- Red
  An error/fault is present
- Yellow
  Axis is in motion (constant velocity, acceleration, deceleration) or a warning is pending
- Gray
  Axis (system variable) is not active

Example

Position control status (**servomonitoring.controlstate**) has the Enum EnumActiveInactive

- Green lamp: ACTIVE, position control active
- Gray (no lamp): INACTIVE, position control not active

5.2.6.4 Ethernet/PROFINET topology

For PROFINET IO systems, the topology editor in HW Config features an integrated error/fault diagnostics facility. Start the topology editor with the **Edit > PROFINET IO > Topology...** menu command in HW Config or NetPro.

How the error/fault diagnostics facility works:

- Graphically displays the error/fault situation
- Online/offline comparison
- Displays faulty/missing components, e.g. CPU, I/O, switch, etc.

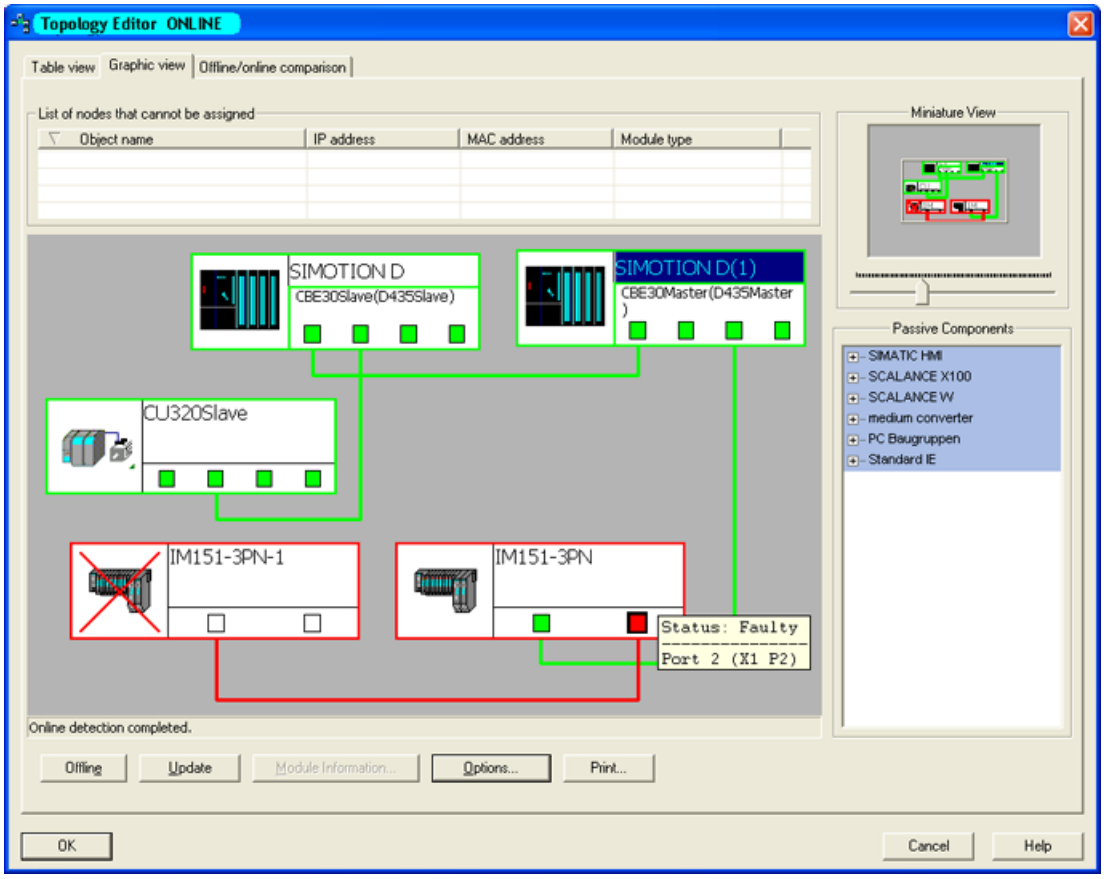


Figure 5-54 Example of topology editor - connection error

Additional information

- Links in the *Guides* menu of the online help
- *Communication* System Manual

5.2.6.5 Comparing projects

You can use the Project comparison function in SIMOTION SCOUT/STARTER (which is started via the button) to compare objects within the same project (offline/offline and offline/online) and/or objects from different projects (offline/offline). Objects are devices, programs, technology objects (TOs) or drive objects (DOs), and libraries. It is not possible to use an empty project in a comparison.

If inconsistencies are displayed in the project navigator when you switch to online mode, for example, you can carry out a project comparison to locate the differences between the SIMOTION SCOUT project and the target system project.

To enable you to analyze differences when comparing objects, there is a range of comparison attributes for each object type, which break an object down into smaller units (comparison features) and, as a whole, contain all the object data that are of relevance to execution.

Information relating to the SINAMICS devices used in the project is also displayed on the same screen. This includes DOs or individual parameters, for example.

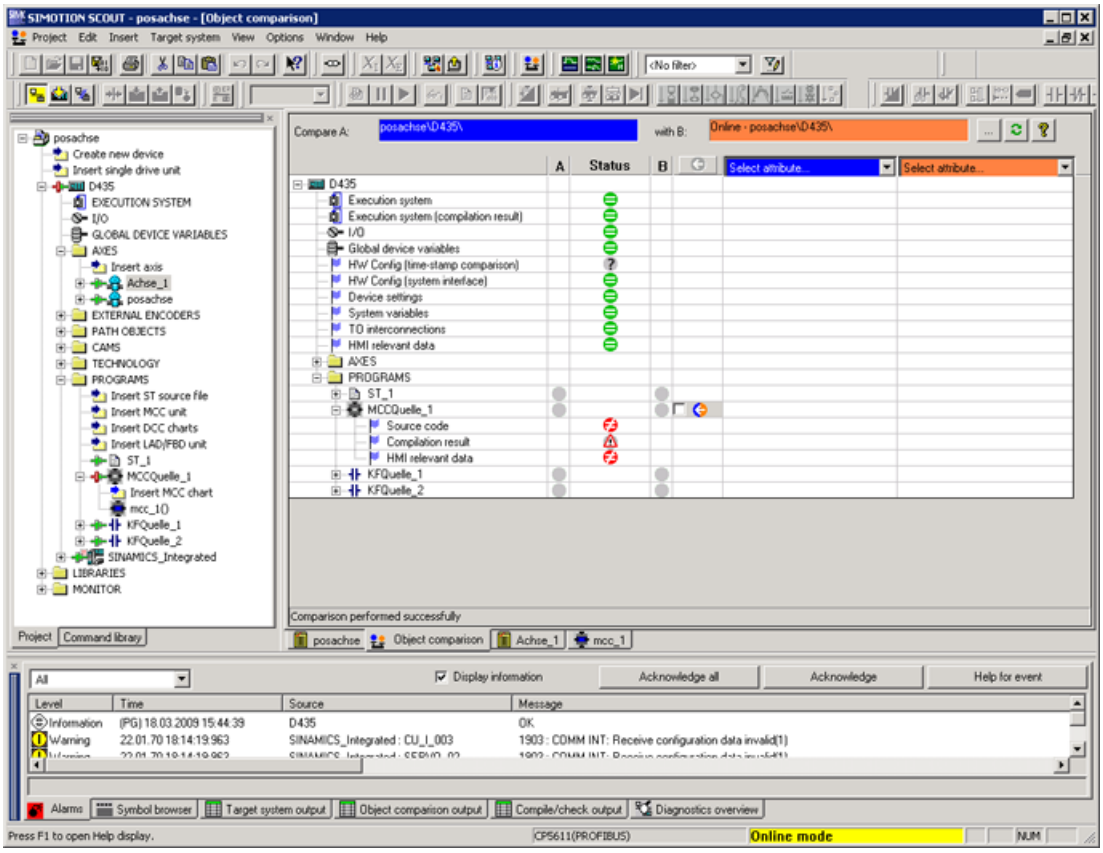


Figure 5-55 SIMOTION object comparison user interface

Possible reasons for inconsistencies in the project navigator (online mode):

- A program has been changed
- The result of compiling a program is different
- There is a deviation on the global device variables
- The execution system has been changed
- The hardware configuration has changed
- An ST program in a library has been changed
- A piece of configuration data for an axis has been changed

The object comparison allows you to establish these differences and, if necessary, run a data transfer in order to rectify them. The comparison can be carried out on an object-by-object or line-by-line basis.

The following comparisons are possible:

- Offline object with offline object from the same project
- Offline object with offline object from a different project
- Offline object with online object

Note

A comparison with online objects is only possible if there is already an online connection.

To enable detailed comparisons, the *Store additional data and sources on the target device* option must have been activated during the download carried out beforehand.

No implicit connection is established when the comparison is started.

Additional information

- Links in the *Guides* menu of the online help
- *Project comparison* Function Manual

5.2.6.6 Error handling in technology objects

Possible errors in technology objects

The following basic errors are possible when programming technology objects:

- The technology object itself cannot execute the function required by the application or reports certain events or states:
  A **technological alarm** is output.
  You can find information on the individual alarms in the SIMOTION Reference Lists
- The command sent to a technology object cannot be executed:
  The **return value** of the command provides information about the cause.
  You can find information on the return values of the commands in the SIMOTION Reference Lists
- Error while accessing configuration data, system variables, or I/O variables
  The ExecutionFaultTask is called in the event of errors when configuration data or variables are being read or written

The Trace technology object is a very effective way of analyzing processing in the technology objects.

Technological alarms

If an event (error, note) occurs on a technology object, the object issues a **technological alarm**.

Configuring technological alarms

Individual responses are preset for each alarm.

You can change this default presetting as follows:
In the project navigator, select the **Execution system** path. The Execution System window opens. Select the **SystemInterruptTasks** > **TechnologicalFaultTask** path.
Then click the **Alarm configuration...** button in the window.

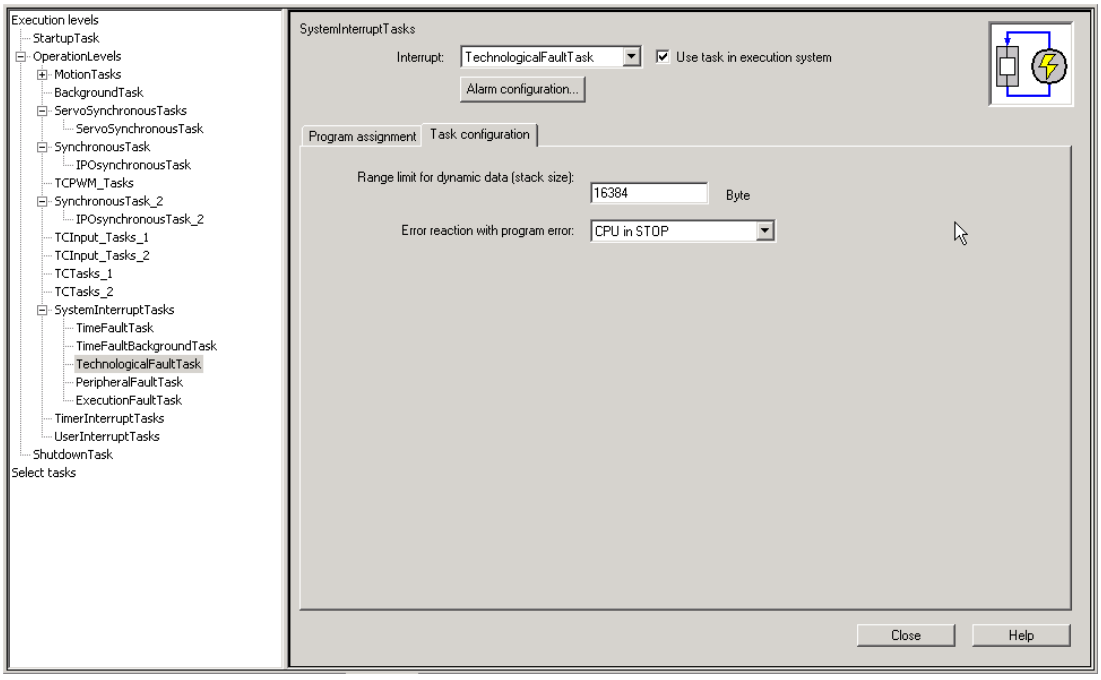


Figure 5-56 Configuring a technological alarm

Displaying and acknowledging technological alarms

Technological alarms can be evaluated and acknowledged in different ways:

- When SIMOTION SCOUT is in **online mode**, alarms and messages are displayed on the **Alarms** tab in the detail view of the workbench.
  Clicking **Acknowledge** deletes all alarms of the associated type.
- The alarms can be output, displayed, and acknowledged via the Human Machine Interface (HMI). See also HMI (Page 3343).
- All pending or individually selected alarms of a technology object can also be queried, evaluated, and acknowledged via the user program.

Acknowledging via SIMOTION SCOUT

1. Select the alarm on the **Alarms** tab of the detail view
2. Click **Acknowledge**.

All alarms of the associated type are then deleted.

Note

Because drive alarms usually generate technology object alarms as well, you can try using the Acknowledge (TO) switch to delete the drive alarms too. If, however, the cause of a drive alarm still exists, then a new TO alarm will be triggered immediately. In this case, first correct the cause of the drive alarm.

5.2.6.7 Advanced functions in the address list

With SIMOTION V4.2, advanced I/O diagnostic functions are available in the address list. This information is displayed in the **Availability** column in online mode. If you position the cursor over the relevant cell, detailed information is displayed in a tooltip.

How to open the address list

1. In the project navigator, navigate to the folder of your device.
2. Double-click the **ADDRESS LIST** entry.
   The address list is opened in the detail area.

The following diagnostic information is recorded

- I/O stations that have completely failed
- Modules that have been removed (e.g. with ET200S)
- Deactivated I/O stations
- I/O variables working with replacement values
- I/O stations whose set topology is different from the actual topology
- I/O stations that have been configured as isochronous, but are not operating isochronously
  - Distributed synchronous operation
  - Drive units
  - Isochronous I/O
- Partner device is in stop mode (e.g. I device, I slave)
- For PROFINET devices: Provider state/consumer state shows errors
  - Controller
  - I/O device
  - Modules
  - Submodules

See also the description of the **_quality()** system function in the section titled *Detailed status of the I/O variables (as of kernel V4.2)* in the *SIMOTION ST Structured Text* Programming and Operating Manual.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION ST Structured Text* Programming and Operating Manual

5.2.6.8 Testing programs

To facilitate program testing, a range of debugging and error handling options are available for the individual programming languages:

- Variable status
- Variable status in source code (online mode)
- Program status
- Single step
- Breakpoints
- Variable trace (see Commissioning functions (Page 3394))

In the watch tables, assigned variables for all SIMOTION devices and parameters of the SINAMICS drives can be monitored and controlled in the project.

Additional information

- Links in the *Guides* menu of the online help
- *SINAMICS/SIMOTION Editor description DCC* Programming and Operating Manual.
- *SIMOTION LAD/FBD* Programming and Operating Manual
- *SIMOTION MCC Motion Control Chart* Programming and Operating Manual
- *SIMOTION ST Structured Text* Programming and Operating Manual

5.2.6.9 Commissioning functions

SIMOTION SCOUT provides the following support functions for the commissioning and optimization of technology objects and user programs:

- Device trace/function generator
- System trace (as of V4.2)
- Measuring function
- Automatic controller setting
- Task Trace
- Technology object trace (as of V4.2)

Device trace/function generator

You can use the device trace to record and evaluate parameters, system variables, and program variables.

The device trace for system variables is mainly used to analyze time-synchronous sequences in the real-time system.

Program variables can be traced in order to find logical errors in the execution system or in user programs. For this purpose, an event-triggered measuring task can be used in the runtime system rather than a time-triggered measuring task. The event that causes the measurement to be recorded is the execution of a specific code position in the user program. In addition, a trigger event based on the variable can be selected on the Device trace tab in order to start the recording (for example, on a positive edge, a tolerance band, or a bit pattern).

The function generator can be used for test purposes to dynamically generate setpoints with defined shapes (e.g. rectangle, sine) for various system variables. With the aid of the device trace, the system response can then be recorded in order to optimize the controllers, for example.

System trace (as of V4.2)

You can use the system trace to record and evaluate parameters, system variables, and program variables from multiple CPUs at the same time. It is essential that the CPUs communicate via PROFINET. There must be an isochronous connection between the CPUs.

The function generator, mathematics functions, and bit tracks are not available for recording with the system trace.

Task Trace

The Task Trace is a tool for troubleshooting in the SIMOTION multitasking environment. The Task Trace offers the following options:

- Graphic display of the sequence of individual tasks and user events (generated using a program command)
- Trace of individual user tasks
- Option of configuring the Task Trace using the SIMOTION IT web server or via the user program (system functions).
- Storage of the trace file on a memory card
- Starting the SIMOTION Task Profiler as a separate application using the SIMOTION IT web server or SIMOTION SCOUT device diagnostics.

Measuring function

The measuring function is used for controller optimization.

The SIMOTION measuring functions are used to commission the axis controller without requiring a user program.

With the SINAMICS measuring function, you can directly inhibit the influence of higher-level control loops by means of simple parameterization, and analyze the dynamic response of individual drives.

Automatic controller setting

The automatic controller setting function can be used to configure the speed controller in the drive and the DSC position controller in the control (SIMOTION SCOUT only) for SINAMICS drive units.

Additional information

- Links in the *Guides* menu of the online help
- *Task Trace* Function Manual

5.2.7 Appendix

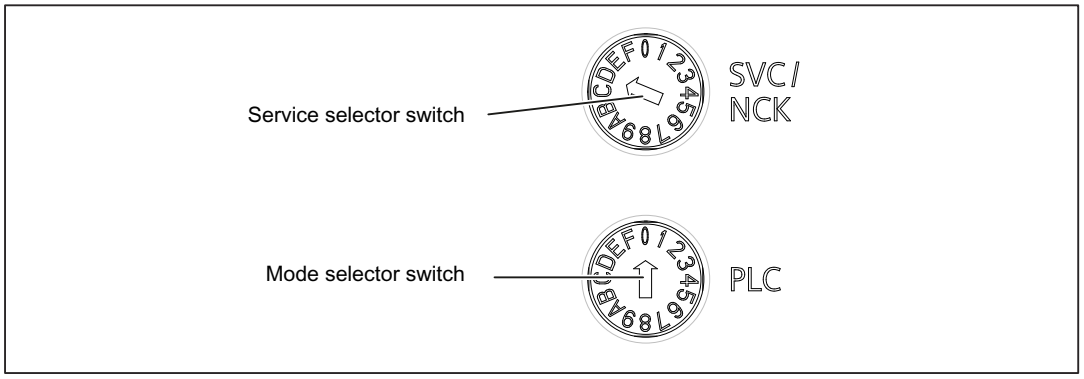
5.2.7.1 Diagnostic data and non-volatile data (retain data)

Backing up during operation using a service selector switch

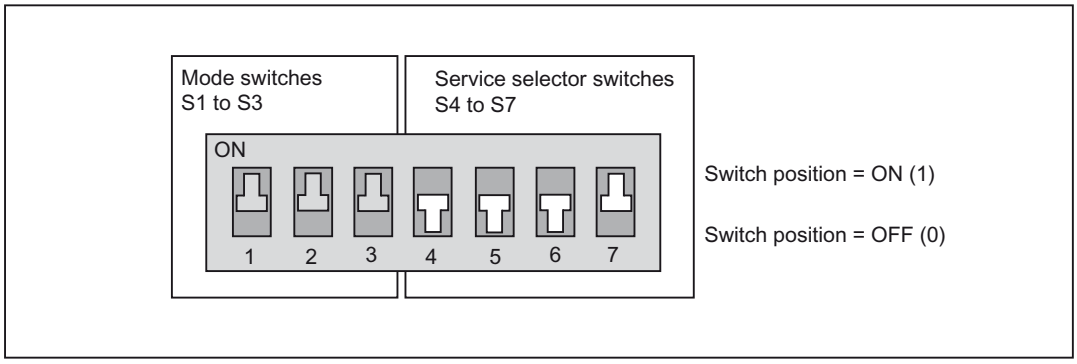
The advantage of backing up diagnostic data and non-volatile data during operation is that enhanced diagnostic information via HTML pages and TO alarm information are available.

Backing up diagnostic data and non-volatile data during operation using the service selector switch (SIMOTION D):

1. Set the service selector switch to "Diagnostics".
   – For D4x5-2: upper rotary switch (SVC/NCK) to position "13" (hex: D)



   – For D4x5: left rotary switch (SIM/NCK) to position "13" (hex: D)
   – for D410-2: upper rotary switch (SVC) to position "13" (hex: D)
   – For D410: DIP switch "7" to ON



   It does not matter what position the D410 mode switch is in.
   The diagnostic data and non-volatile data can be created in STOP, STOPU, and RUN modes.

2. The diagnostic data and non-volatile data are backed up to the CF card.
   The status LEDs display the backup process as follows:

Status	D410/D410-2	D4x5/D4x5-2
Backup in progress	RUN/STOP LED flashes yellow (2 Hz)	STOP LED and SU/PF LED flash yellow (2 Hz)
Backup complete	RUN/STOP LED flashes green (2 Hz) and SF LED lights up continuously	RUN LED flashes green (2 Hz) and SF LED lights up continuously

3. Once the backup is complete, switch the SIMOTION D off.
4. Remove the CF card and reset the service selector switch to its original setting.

Note

SIMOTION D410-2/D4x5-2 modules feature a DIAG button. As an alternative to setting the service selector switch to the "D" position, you have the option of backing up diagnostic data and non-volatile data by briefly pressing the DIAG button.

This method is preferable for D410-2/D4x5-2.

Backing up diagnostic data and non-volatile data during operation using SIMOTION P state (SIMOTION P):

1. Make sure that SIMOTION P has completed startup, e.g. with SIMOTION P state.
2. Activate the diagnostic switch using the following option:
   SIMOTION P state menu command **Options > Set Diagnostic Switch**
   The diagnostic data is recorded during operation and the **Options > Set Diagnostic Switch** switch is automatically reset after the data has been written.

Please contact Product Support for the evaluation of the recorded data.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

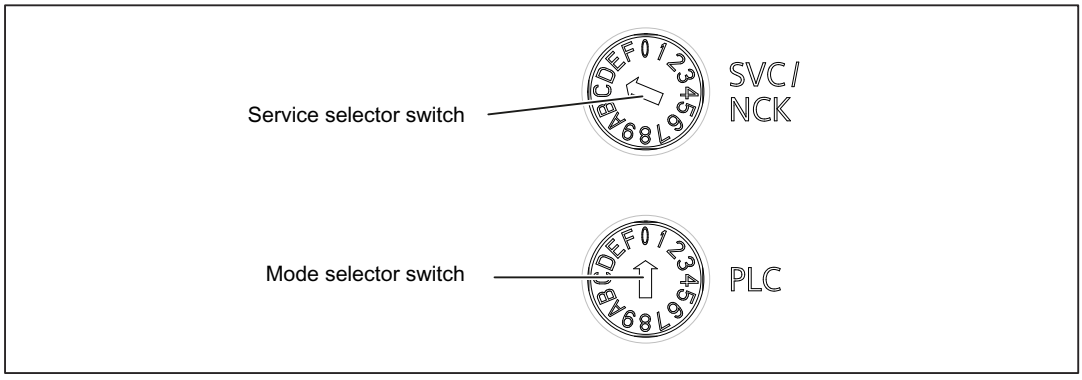
See also

Backing up diagnostic data and non-volatile data (Page 3360)

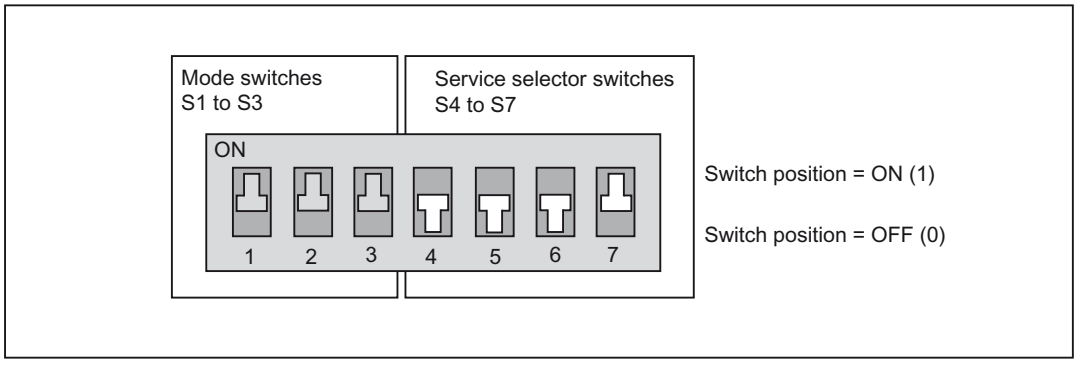
Backing up during ramp-up using a service selector switch or INI file

Backing up diagnostic data and non-volatile data during start-up using the service selector switch (SIMOTION D):

1. Set the service selector switch to "Diagnostics".
   - For D4x5-2: upper rotary switch (SVC/NCK) to position "13" (hex: D)



   - For D4x5: left rotary switch (SIM/NCK) to position "13" (hex: D)
   - for D410-2: upper rotary switch (SVC) to position "13" (hex: D)
   - For D410: DIP switch "7" to ON



     It does not matter what position the D410 mode switch is in.

2. Switch the SIMOTION D off and on again.
3. Wait for the device to start up.
   The diagnostic data and non-volatile data is backed up to the CF card during startup, provided that this is still possible and is not prevented by hardware defects, for example.
4. Once the backup is complete, switch the SIMOTION D off.
5. Remove the CF card and reset the service selector switch to its original setting.

Note

SIMOTION D410-2/D4x5-2 modules feature a DIAG button. As an alternative to setting the service selector switch to the "D" position, you have the option of backing up diagnostic data and non-volatile data by pressing the DIAG button. When backing up data during start-up, the DIAG button must be held down until the backup process is complete. As this can easily take 20-30 seconds, it is better to use position "D" in this case.

Backing up diagnostic data and non-volatile data during start-up using the SIMOTION P state application (SIMOTION P):

1. Terminate the SIMOTION P with **Terminate SIMOTION P** from SIMOTION P state.
2. Activate the diagnostic switch using the following option:
   SIMOTION P state menu command **Options > Set Diagnostic Switch**
   The diagnostic data is recorded during start-up and the **Options > Set Diagnostic Switch** switch is automatically reset once the data has been written.

Backing up diagnostic data and non-volatile data during start-up using an INI file (SIMOTION C/P/D):

1. Use a text editor (such as Notepad) to create a file called `simotion.ini`.
2. Add the following text to it: `DIAG_FILES=1`
   You must use a text editor and may not use any formatting in the text.
3. Copy `simotion.ini` to the main directory of the data medium (`D:\Card` with SIMOTION P320 or `F:\Simotion\user\Card` with SIMOTION P350).
4. With SIMOTION C/D: Insert the data medium into the module while it is switched off.
5. Switch the module on and allow the SIMOTION device to start up.
   The diagnostic data and non-volatile data is backed up to the data medium during start-up, provided that this is still possible and is not prevented by hardware defects, for example.
6. Once the backup is complete, switch off the SIMOTION device.
7. With SIMOTION C/D: Remove the data medium.

NOTICE
Diagnostics mode To suppress start-up in diagnostics mode again, you must delete the `simotion.ini` file from the data medium.

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Manual
- *SIMOTION D4x5-2* Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Manual
- *SIMOTION D410* Commissioning and Hardware Installation Manual
- *SIMOTION D410*-2 Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

See also

Backing up diagnostic data and non-volatile data (Page 3360)

Backing up during operation using the SIMOTION IT web server

SIMOTION devices feature pre-configured, standard SIMOTION IT diagnostic pages. These pages can be displayed via Ethernet using a commercially available browser. Additionally, you have the option of creating your own HTML pages and incorporating service and diagnostic information.

You can also use the SIMOTION IT Web server to back up diagnostic data and non-volatile data. The web page is opened by entering the IP address of the SIMOTION device in the address line of the browser; e.g. http://192.168.0.22

This opens the start screen of the web server. To back up diagnostic data and non-volatile data, call the "Diagnostic files" page from the "Diagnostics" menu.

Backing up and saving diagnostic data and non-volatile data on the PG/PC

1. If any diagnostic data is present, delete it using the **Delete all diagfiles** button.
2. Initiate the backup process using the **Create general diagfiles** button.
3. Compress the files using the **Zip all diagfiles** button.
4. Transfer the compressed diagnostic data to the PG/PC using the **Get diagarchive** button.

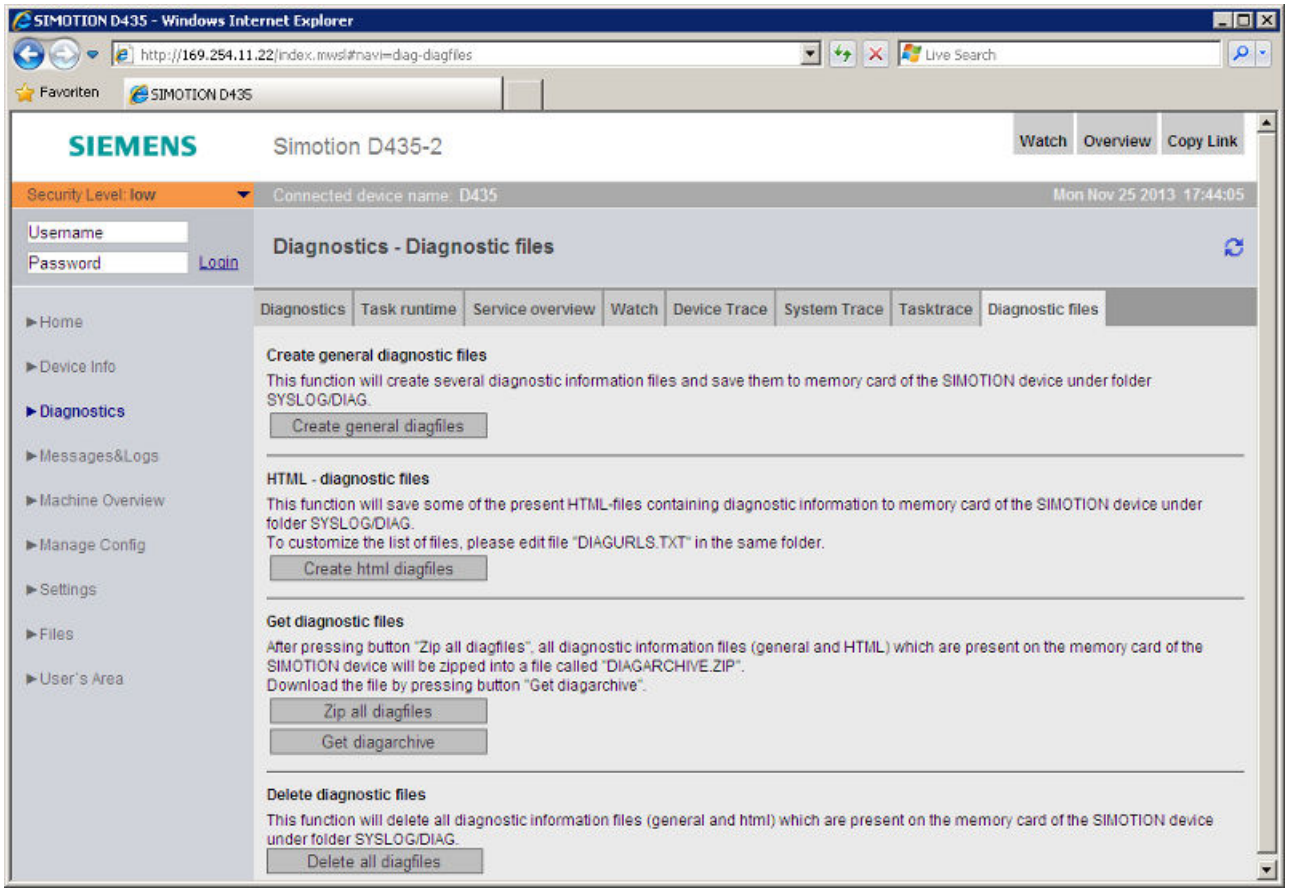


Figure 5-57 SIMOTION IT web server

Table 5-28 Functions on the "Diagnostic files" HTML page

Button	Function
Create general diagfiles	This button saves diagnostic data and non-volatile data in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory. HTML files used for diagnostics purposes are not saved.
Create html diagfiles	This button is used to save diagnostics HTML pages on the data medium. It should be noted that only those pages that are listed in the DIAGURLS.TXT file in directory ...\USER\SIMOTION\HMI\SYSLOG\DIAG will be saved (see Displaying diagnostic data via websites (Page 3362)).
Zip all diagfiles	The "zip all Diagfiles" button enables you to compress diagnostics files. This stores all files and folders in a ZIP file in directory ...\USER\SIMOTION\HMI\SYSLOG\DIAG, while retaining the folder structure.
Get diagarchive	This button is used to save the ZIP archive to connected PGs/PCs.
Delete all diagfiles	This button is used to delete all data stored in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory; the directory itself is not removed, however.

See also Storing diagnostic data and non-volatile data (Page 3361)

Additional information

- Links in the *Guides* menu of the online help
- Diagnostics Manual *SIMOTION IT Diagnostics and Configuration*

Restoring non-volatile data

SIMOTION C/P

Prerequisite

The diagnostic data and non-volatile data have been backed up to the MMC/data medium.

Procedure

1. With SIMOTION C and SIMOTION P320/P350, copy file `PMEMORY.XML` from directory `...\USER\SIMOTION\HMI\SYSLOG\DIAG` to directory `...\USER\SIMOTION`.
2. Following this, carry out an overall reset to force the non-volatile data to be reimported.

In principle, you can use the procedure that has just been described for SIMOTION D too, but we recommend using the method described below instead.

SIMOTION D

Prerequisite

The diagnostic data and non-volatile data have been backed up to the CF card.

Procedure

As of V4.3, the non-volatile SIMOTION data are restored automatically when a module is replaced. The non-volatile data can also be restored manually (by manual operation).

The CF card may contain backups of non-volatile SIMOTION data in various storage locations:

- data backed up with the system function _savePersistentMemoryData
  Storage location on CF card:
  - /USER/SIMOTION/PMEMORY.XML
  - /USER/SIMOTION/PMEMORY.BAK (backup file)
- manually by service selector switch / web server / DIAG button, backed-up data
  Storage location on CF card:
  - /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML

On manual restoration, the position of the service selector switch defines which of these data will be preferably restored.

During restoration, the non-volatile SIMOTION data are first deleted and then the non-volatile SIMOTION data are restored via the PMEMORY backup file.

If restoration is not possible (e.g. file does not exist or corrupt), the next file in the priority list is accessed.

Table 5-29 Restoration of the non-volatile SIMOTION data

Position of the service selector switch	Use case	Priority sequence for use of the data backups
D410-2/D4x5-2/D4x5: Rotary switch position "1" D410: DIP switch "5" to ON	The data backed up with the system function _savePersistentMemoryData are preferably restored	1. /USER/SIMOTION/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.BAK 3. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML
D410-2/D4x5-2 (V4.4 and higher): Rotary switch position "A"	The data backed up by service selector switch position "D" / web server / DIAG pushbutton are preferably restored	1. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.XML 3. /USER/SIMOTION/PMEMORY.BAK

Note

Firmware / kernel < V4.4

Because switch position "A" is only supported as of V4.4, for < V4.4, restoration must be performed with switch position "1." To force restoration of the data backed up by service selector switch position "D" / web server / DIAG button, it may be necessary to delete existing files PMEMORY.XML and PMEMORY.BAK in the directory /USER/SIMOTION/ on the CF card.

Restoring data with switch position "1" or "A"

Procedure

To restore the non-volatile SIMOTION data, proceed as follows:

Step	Switch position "1"	Switch position "A" (as of V4.4)
1.	Insert the CF card into the new SIMOTION D. The SIMOTION D must be switched off!	
2.	Set the service selector switch to "**1**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service switch position 1	Set the service selector switch to "**A**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service switch position A
3.	Switch on the SIMOTION D.	Switch on the SIMOTION D. The flickering green SF LED indicates that restoration is requested via position "A."
4.	Restoration is started automatically.	Turn the service selector switch to "1" to start restoration.
5.	Once restoration has been completed, the module will start up automatically.	
6.	Switch the module off and turn the service selector switch back to "0."	
7.	Switch the SIMOTION D on again.	

Note

For the SIMOTION D410, the data are restored by setting DIP switch "5" to ON

Additional information

- Links in the *Guides* menu of the online help
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION D410* Commissioning Manual
- *SIMOTION D410-2* Commissioning and Hardware Installation Manual
- *SIMOTION P320-3 and Panels* Commissioning and Hardware Installation Manual
- *SIMOTION P350-3 and Panels* Commissioning and Hardware Installation Manual

See also

Restoring non-volatile data (Page 3365)

5.3 Project comparison

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4.3:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

5.3.1 Fundamental safety instructions

5.3.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

5.3.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

5.3.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

5.3.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

5.3.2 Overview of project comparison

You can use the SIMOTION SCOUT/STARTER **Project comparison** function (start this via the **Start object comparison** button) to compare objects within the same project and/or objects from different projects (online or offline). Objects in the project comparison are SIMOTION devices, drive units, libraries. The project comparison is available with SIMOTION SCOUT and STARTER. Comparing projects is useful if you need to carry out service work on the system.

It may, for example, be the case that inconsistencies are indicated when you switch to online mode in the project navigator, i.e. there are deviations between your project in SIMOTION SCOUT/STARTER and the project loaded to the target system.

Examples of possible causes for this are:

- A program has been changed
- The result of compiling a program is different
- There is a deviation on the global device variables
- The execution system has been changed
- The hardware configuration has been changed
- An ST program in a library has been changed
- A piece of configuration data for an axis has been changed

The object comparison allows you to establish these differences and, if necessary, run a data transfer to rectify the differences. The following comparisons are possible:

- Offline object with offline object from the same project
- Offline object with offline object from a different project
- Offline object with online object (not for libraries)

The first comparison partner, the reference object (A), is in the open project, while the second partner, the comparison object (B), can be a comparison object in the open project, in a different project, or the online partner.

Note

When comparing to a project that has been created with a SIMOTION SCOUT/STARTER Version < 4.3, you must first convert the project to SIMOTION SCOUT/STARTER Version 4.3. For the comparison attributes, aspects and subaspects to be displayed, you must recompile the project after it has been upgraded.

The project comparison function is also available when using devices with older firmware versions if you have edited or converted the project with SIMOTION SCOUT or STARTER V4.3.

The highest level for a comparison is a device, not the project. The comparison can be started via a

- SIMOTION device,
- Drive unit, or
- Library.

A detail comparison of the individual objects is available for the programming languages ST (Structured Text), LAD/FBD (ladder logic/function block diagram), and MCC (Motion Control Chart) as well as for DOs (drive objects) and TOs (technology objects) and simply structured data such as CPU system variables, TO interconnections, among others.

Note

Reliable statements on the comparison status of the creation data (e.g. compilation result of a program) can only be made when the project has been compiled. For example, changing an interface in source A has an effect on the compilation result of source B, which uses this interface.

Note

The displayed values are of generic nature, for example, as calculated in the CamEdit configuration, and are not rounded off.

Differences between the project comparison and the online/offline comparison

In terms of drive objects, the project comparison produces precise results compared to the online/offline method. This is because the project comparison compares the contents of projects whereas the online/offline method only evaluates on the basis of a change counter. The consistency information for DOs, displayed in the project navigator, is also based on the change counter.

Example: For a particular DO, the project navigator shows an inconsistency between the project data in the PG/PC and the project data in the target system. The online/offline comparison may highlight a difference, while the results of the project comparison show everything to be the same.

5.3.3 General information on object comparison

You can use the SIMOTION SCOUT/STARTER Object comparison function (start this via the **Start object comparison** button) to compare objects within the same project and/or objects from different projects (online or offline), see also Project comparison overview. Objects are devices, programs, technology objects (TOs) or drive objects (DOs), and libraries. Below you will find an overview of the elements and structure of the **Object Comparison** dialog box. You can start the detail comparisons from this dialog box, e.g. for LAD/FBD, ST or also for CPU system variables.

- The user interface
- Comparison icons
- Comparison tree
- Comparison attributes, aspects, and sub-aspects
- Object attributes

5.3.3.1 The user interface

A user interface in which the differences in the various objects have been detected has been selected as an example for the screenshot below. In the case of programs, all the data from this user interface could be transferred and it would be possible to carry out a detailed comparison for two programs.

Table 5-30 The user interface elements of the project comparison

No.	Description
1	Comparison partner A and comparison partner B (online/offline)
2	Comparison tree made up of comparison partner elements
3	Object present (column A, column B): The columns indicate whether or not a tree object is present for comparison partner A and/or comparison partner B.
4	Result of comparison: The comparison result is shown in this column (same/different). (See Comparison icons) **Note:** If the comparison is no longer up-to-date or when offline during online comparisons, the status column heading is displayed hatched.
5	Select/change comparison partner
6	Update comparison
7	Call up online help
8	Display object attributes (see Object attributes) The attributes of the comparison partners are compared in two columns.
9	Start detailed comparison
10	Comparison attributes of an object (see Comparison attributes, aspects and sub-aspects)
11	Aspects of a comparison attribute (see Comparison attributes, aspects and sub-aspects)
12	Aspect, created data
13	Data transfer: Transfer of data from comparison partner B into the open project.
14	Status bar
15	Object comparison tab
16	Settings for transferring objects

Figure 5-58 SIMOTION object comparison user interface

See also

Comparison icons (Page 3412)

Object attributes (Page 3414)

Comparison attributes, aspects, and sub-aspects (Page 3413)

5.3.3.2 Comparison icons

Below you will find an overview of the icons used in columns A and B, and their status.

Table 5-31 Overview of icons used

Icon	Description
	Object is present (displayed in columns A and/or B).
	Object is not present (displayed in column A and/or B).
	Difference in at least one subordinate object or comparison attribute (only displayed if the folder or object is collapsed).
	The comparison attribute is the same.
	The comparison attribute is different.
	The compilation result does not match the source status of comparison partner A or B.
	A same/different statement cannot be made. An example of a possible cause of this is that the hardware configuration is being compared using the time stamp. If the time stamp deviates, a same/different statement cannot be made.

Please note the following:

- Columns A and B are not completed for comparison attributes. The same objects have the same attributes, i.e. they are present. However, the characteristic of the attributes may be the same or different; this same or different characteristic is indicated accordingly.
- If an object does not have a comparison partner, the comparison attributes cannot be seen (the objet cannot be expanded in the tree).
- Folders which do not contain any objects are not shown in the comparison tree.

5.3.3.3 Comparison tree

The way in which the comparison tree is displayed reflects how trees are displayed in the project navigator. The comparison tree is extended by displaying the comparison attributes (see the chapter titled Overview of comparison attributes (Page 3452)).

A single tree is produced from the two trees/subtrees of the comparison partners. If it has been possible to assign two of the comparison partners' objects (see the Assigning objects in object comparison (Page 3419) section), there will be **one** entry in the tree, and columns **A** and **B** will contain the **Object available** symbol. If it has not been possible to assign any comparison partners, the **Object present** or **Object not present** icon is displayed in the column for comparison partner A or comparison partner B, depending on which it belongs to (see example below).

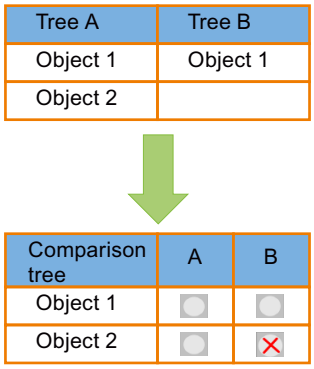


Figure 5-59 Example of merging comparison partners

If an object is only present for one comparison partner, no comparison attributes are shown in the tree. Subordinate folders and objects are shown.

If the comparison partners are different , the tree is shown expanded. This gives you a quick overview of the differences.

The comparison tree shows a snapshot. As a result of changes in the open project, the comparison status may no longer be up to date. You will be informed of this status by means of

- the shaded column heading  and
- the message **The comparison is no longer up-to-date** in the status bar.

Since the comparison tree is not automatically updated, you have to explicitly start the update by clicking the **Update** button.

Note

The comparison only displays parameter differences which are relevant to the download; differences relating to current data are not displayed. For the DOs, the download parameters are compared (please also refer to the Display filter for DO detail comparison) section; the configured values are compared for the TOs.

5.3.3.4 Comparison attributes, aspects, and sub-aspects

In order to be able to analyze differences when comparing objects, there is a range of comparison attributes for each object type, which structure an object into smaller units (aspects and sub-aspects) and which, as a whole, contain all the object data that is of relevance to execution.

You will already be familiar with some of the comparison attributes from the project navigator (e.g. global device variables). In addition to this, there are comparison attributes into which various items of data can be incorporated, such as the compilation result for programs, where the global and local compiler settings are also included.

Categorization of comparison features

Comparison features are divided into attributes, aspects, and sub-aspects on the basis of object type (sub-aspects are only available for MCC and LAD/FBD POUs). This enables you to determine the causes of differences as precisely as possible.

Comparison features are available in singular or multiple format (such as OEs) and their content can be empty, e.g. exported variables of a program. The comparison features are displayed in the object comparison, regardless of whether or not they are empty.

You will find an overview of the specific comparison attributes, plus their associated aspects and sub-aspects, for SIMOTION, SINAMICS, and MICROMASTER objects in the section titled Overview of comparison attributes (Page 3452).

To enable the attributes to be compared as efficiently as possible, information is calculated (e.g. a checksum) and is then stored online along with the project or objects. If this information is not available (e.g. in the case of the hardware configuration), a statement on the comparison cannot be made (indicated in the comparison tree by **?**).

5.3.3.5 Object attributes

You can select the following object attributes in the two right-hand columns in the object comparison.

Table 5-32 Overview of object attributes

Name	Description
Name	Name of object[1)]
Type	Type of object Objects with the same name but of a different type are not compared in the case of automatic assignment.
Object address	Unique object address[1)] You need the object address to address the objects via the HMI, for example.
Author	Name of the author[1)]
Version external	External version identifier[1)]
Version internal	Internal version identifier of object, e.g. the technology package version in the case of TOs
Creation version	SIMOTION SCOUT version with which the object was created
Know-how protection for programs	Login for all know-how-protected objects
Time stamp of last change	Date and time of last change
Comment	Comment[1)]
[1)] from the Object properties dialog.	

5.3.4 Functions

5.3.4.1 Downloading additional data

In order that can carry out a detailed comparison and, if necessary, a complete data transfer into the PG/PC during an offline/online comparison, when downloading the project to the target device you must also save the additional data, such as sources, icon information etc. This is a setting in SIMOTION SCOUT or STARTER.

If you only want to accept selected data from the additional data, you can do this via the detailed comparison.

Note

It is also possible to load supplementary data when using SIMOTION devices with older firmware versions if you have edited or converted the project with SIMOTION SCOUT V4.1 SP2. The project then has to be recompiled and loaded.

Further information on loading additional data can be found in the basic functions, Loading data from the target device to the PG/PC.

Procedure

Proceed as follows:

1. Select **Tools > Settings**.
   The **Settings** window appears.
2. Switch to the **Download** tab.
3. Activate **Save additional data on the target device** and confirm using **OK**.

The next time the project is downloaded to the target device, the additional data will be saved on the target device. If deactivated, the additional data will be deleted the next time downloading to the target device takes place. In addition, this option can be selected during each download (CPU-related).

5.3.4.2 Starting an object comparison

Object comparisons are only performed using saved data. Changes made after the object comparison was performed may mean that the object comparison and the detailed comparison have different results.

You must note the following during an online comparison and a comparison with a different project:

The comparison only uses the saved data in the open project.

During an **online comparison**, the device being used for comparison purposes is implicitly uploaded. Therefore, the comparison is made with a copy of the online data.

Note

We recommend compiling the project prior to starting the project comparison. This ensures that you obtain a reliable comparison result (e.g. compilation result of a program).

Note

A comparison with online objects is only possible if there is already an online connection.

Note

If you delete a comparison partner and then update the comparison, an error message is displayed.

Note

The object comparison does not display differences of current data (current data values). Only download-relevant parameter differences are displayed.

Note

In an offline/online comparison, the comparison result is still marked as current after restoring the factory settings. The status only changes when you have changed the project in SCOUT or have performed an online comparison and have then gone offline.

Starting the object comparison and selecting comparison partners

How to start the object comparison:

1. Click on the device with which you want to run the comparison in the project navigator.
2. Either start the object comparison via
   - The menu bar, using **Options > Compare...**,
   - Using the  button or
   - The context menu of the comparable objects (**Compare...**).
     The following dialog appears:

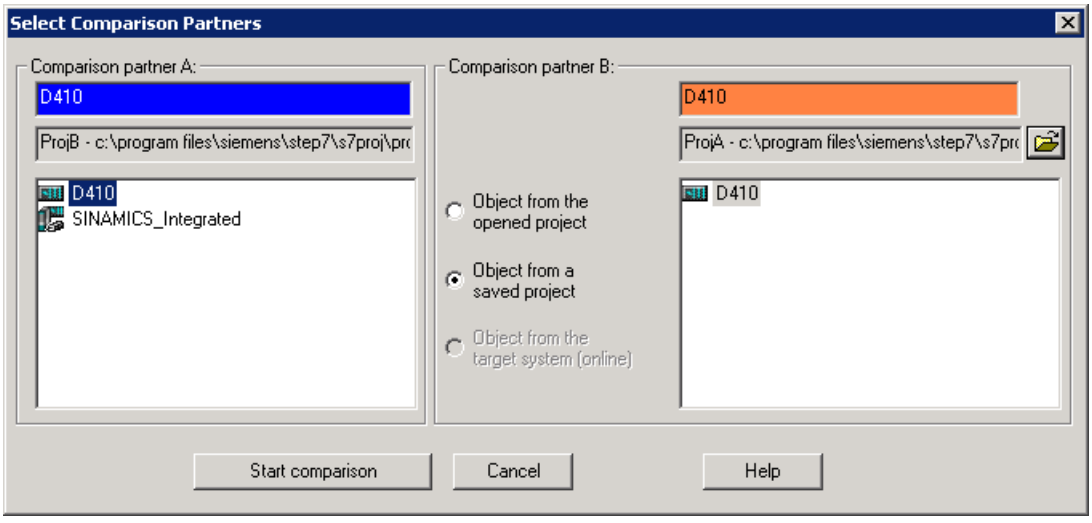


Figure 5-60 Select comparison partner dialog

The object you selected previously in the project navigator is highlighted as comparison partner A. If you did not select an object, the first object in the list is highlighted.

3. Select comparison partner B. The following options are available:
   - Object from the open project:
     You can choose from the objects in the open project.
   - Object from a saved project:
     First open a saved project using the button: 
     You can choose from the objects in this project.
   - Object from the target system (online): (only possible in ONLINE mode and if the project is present in the target system).
     The object from SCOUT or STARTER is compared with the object on the target system.

   Note

   Only comparable devices are displayed as comparison partners.

4. The **Start comparison** button is activated once you have selected the object with which the comparison can be run. Open the object comparison using the **Start comparison** button.

Different versions detected (offline comparison with saved project)

Note

If you are carrying out a comparison with a different project, the project concerned must be available in SIMOTION SCOUT/STARTER Version 4.2; otherwise, you will have to convert the project first.

If the comparison partner is in a different project created using an older version of SIMOTION SCOUT, a comparison cannot be carried out. The following note is displayed:

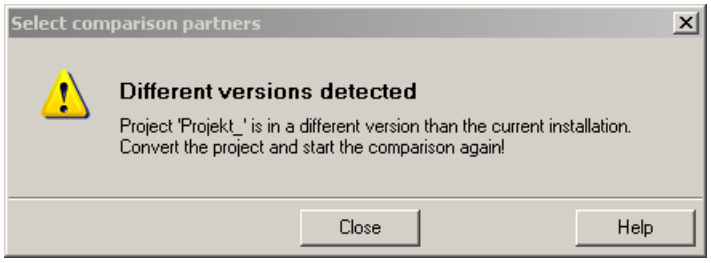


Figure 5-61 Different comparison partner versions

In this case, first convert the project to the current version of SIMOTION SCOUT. Then restart the comparison.

Object comparison with projects converted to a higher version

Requirement

To be able to perform an object comparison, where projects have been created with SIMOTION SCOUT/STARTER versions < 4.2, you first need to convert these projects to a higher version.

Display of new comparison attributes

As part of SIMOTION SCOUT/STARTER version 4.2, new comparison attributes and aspects have been introduced for object comparison purposes. This results in the following scenarios when carrying out comparisons:

- You convert a project to V4.2, go online, and perform a comparison with an object on the target system:
  The new comparison attributes are not displayed, as they are not available on the target system.
- You convert two projects to V4.2 and compare these offline:
  The new comparison attributes are not displayed.
- You convert two projects to V4.2, compile one of the two projects, and compare these offline:
  The new comparison attributes are displayed. However, no comparison is possible because the attributes in the non-compiled project are not available.

Note

The new comparison attributes are calculated when the projects are compiled with SIMOTION SCOUT/STARTER V4.2. When projects are recompiled, the previous comparison attributes are also recalculated.

Display of aspects

Aspects are updated when the comparison attribute is opened, making them available for comparison.

Going offline after online comparison (comparison is no longer up to date)

Comparison no longer up-to-date

Certain pieces of status information depend on the online connection. If you go offline after performing an online comparison, it means that

- A comparison attribute cannot be expanded (unless it was previously opened)
- A detail comparison can no longer be started
- It is no longer possible for entire objects to be transferred

If you attempt to execute one of the functions, the message **Target device is offline** is displayed.

Column header display is hatched

The hatching of column headers means that the comparison is no longer up-to-date. Update the comparison.

Hatching occurs in the following situations:

- If you have made a change on an open project (comparison partner A).
  If a change has been made with a comparison partner B, e.g. with another SCOUT, there is no hatching.
- If you have performed an online comparison and then go offline.

5.3.4.3 Assignment of objects in object comparison

Object assignment

The object comparison only allows you to compare objects which are sufficiently similar, for example, SIMOTION C240 can be compared with SIMOTION D445-2, but SIMOTION D445-2 cannot be compared with an ST source file. After you have decided which objects are to be compared (see the Starting the object comparison and selecting comparison partners section), the lower-level Simotion SCOUT/STARTER objects will be assigned automatically. The rules described in the next section apply within this context.

Automatic assignment of the subordinate comparison objects

During automatic assignment, all objects below the selected comparison object are assigned, taking their position in the tree into account. For example, axis_1 and axis_2 are assigned in the comparison. Similarly, the automatic assignment of lower-level objects will attempt to assign the measuring inputs under axis_1 to the measuring inputs under axis_2.

During a specific object-to-object assignment, a distinction is made between TOs/DOs and other objects:

Assignment of TOs/DOs using object addresses

TOs/DOs are assigned using their object address and name. In this case, the object address appears after the name: <Name>(<Object address>). Since the name of the object may differ in this case, both names are shown in the event of deviation.

When assignment uses the object address, it may be, for example,

1. that a measuring input under axis **A1** has the same object address as the measuring input under axis **A1** in the comparison project, although the object addresses of the axes are different.

Example before object address change

Tree A	Tree B
Axis A1 (17) └ Measuring input M1 (40)	Axis A1 (10) └ Measuring input M1 (40)
Axis A2 (25) └ Measuring input M2 (20)	Axis A2 (25) └ Measuring input M2 (20)

Comparison tree	A	B
Axis A1 (17) └ Measuring input M1 (40)	○ ○	✕ ✕
Axis A2 (25) └ Measuring input M2 (20)	○ ○	○ ○
Axis A1 (10) └ Measuring input M1 (40)	✕ ✕	○ ○

Example after object address change

Tree A	Tree B
Axis A1 (17) └ Measuring input M1 (40)	Axis A1 (17) └ Measuring input M1 (40)
Axis A2 (25) └ Measuring input M2 (20)	Axis A2 (25) └ Measuring input M2 (20)

Comparison tree	A	B
Axis A1 (17) └ Measuring input M1 (40)	○ ○	○ ○
Axis A2 (25) └ Measuring input M2 (20)	○ ○	○ ○

Figure 5-62 Example of assignment where axes have different object addresses

2. A measuring input under axis **A1** has the same object address as the measuring input under axis **A2** in the comparison project:

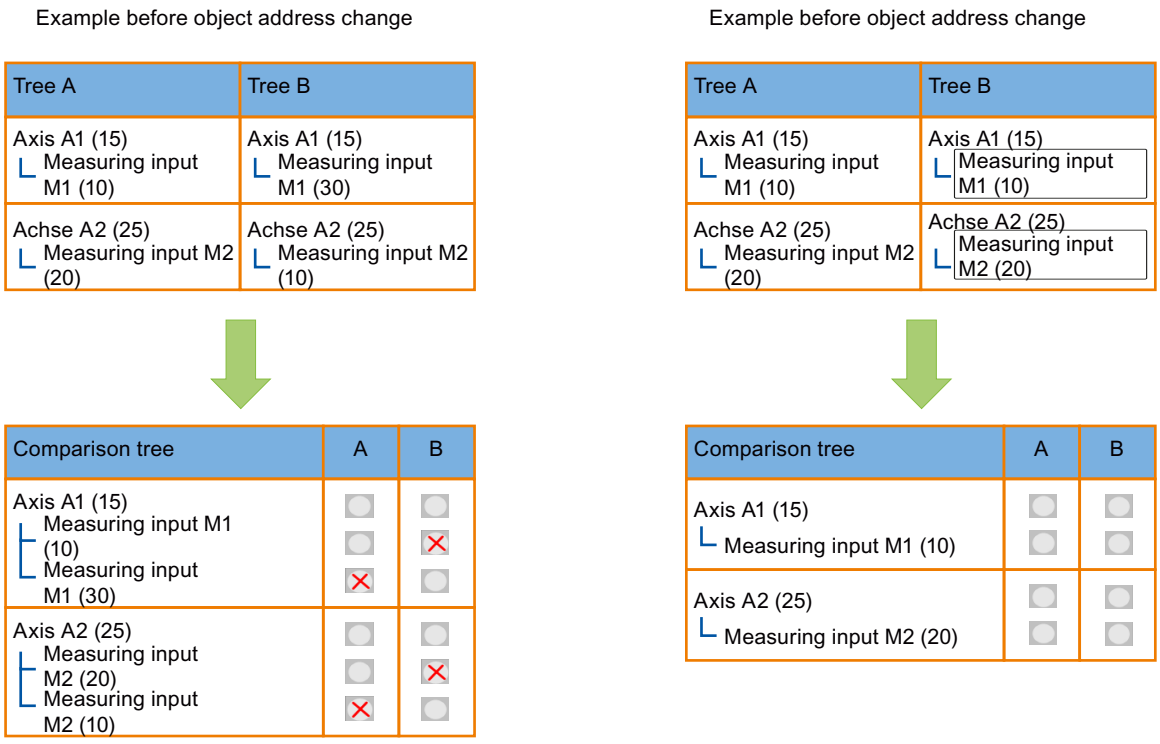


Figure 5-63 Example of assignment where measuring inputs have different object addresses

Since the object addresses are assigned automatically, this may occur if the objects are created in a different order or using script, for example.
However, this assignment is not desirable as it would result in incorrect object assignment. Therefore, the object address is only taken into account within the same hierarchy level during assignment; not across hierarchy levels.

If necessary, you can change the object addresses of TOs yourself. This allows you to enforce correct assignment of the objects later on.

Note

It is recommended that you explicitly assign the object address for TOs in all cases (see Changing the object address (Page 3422)). You **cannot** change the address of DOs.

Assignment of other objects using names

Other objects, i.e. all objects outside the TOs/DOs, are assigned using their names.

Note

If objects are not automatically assigned despite having the same name and are therefore positioned one above the other in the comparison tree, this is because the type is different.

Changing the object addresses of TOs

Procedure

To change a TO object address:

1. Select the appropriate TO in the project navigator.
2. In the context menu, open **Properties...**.
3. Click the **Object address** tab.
4. Change the object address in the **New address** field.
5. Click the **Assign** button and confirm with **OK**.

5.3.4.4 Detailed comparison

Introduction

The following section provides you with an overview of which detail comparisons are available and how you can make use of the respective types.

Detail comparison

You can compare and partially transfer the content of data in the detail comparison. If differences are detected during an object comparison, you can start a detail comparison for

- ST programs
- LAD/FBD (ladder logic/function block diagram)
- MCC (Motion Control Chart)
- TOs (configuration data, system variables), and

- DOs (parameters).
- Simply structured data:
  - I/O variables
  - Global device variables
  - Alarm configuration
  - Execution system
  - Compiler settings
  - CPU system variables
  - Technology object interconnections
  - Used technology packages
  - HW Config (system interface)
  - Cam configuration
  - Topology serial numbers
  - Message configuration (used symbols/IDs)

Note

If you have compared individual data items, such as the device name, then this is called "scalar" in this context. Differences are then displayed in a dialog box.

Know-how protection for programs

In the case of know-how-protected sources, differences are only displayed in the object comparison / detail comparison to a limited extent. Know-how protection for programs protects the source data, but not the attributes or properties. If you want to expand the source code comparison attributes, you will need to enter the password for the protected source. The source code aspects are only displayed when you are logged in.

Starting the detailed comparison

Requirement

The detail comparison can be started from the comparison tree by clicking the  button.

You can only start the detail comparison if

- you have found differences and
- if the additional data required is available for the online comparison.

In the detail comparison, the corresponding contents are shown next to each other, the differences marked, and the transfer of changes from the comparison object to the reference object made possible if required.

Note

You can only start the detail comparison provided the relevant program of comparison partner A is not already open in an editor. The editor is automatically closed when a detail comparison is started.

The detail comparison cannot process projects that are opened with write protection. You must first cancel the write protection.

Procedure

1. First, carry out an object comparison on the desired device (see Starting the object comparison).
2. The **Start detail comparison** button is shown in the Status column for objects for which a detail comparison can be run.

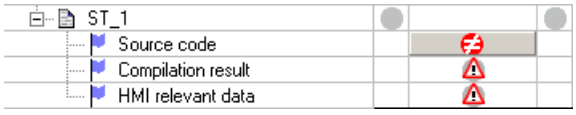


Figure 5-64 Example of a detail comparison for two ST programs

3. Click the **Start detail comparison** button. The corresponding editor in the working area will open and the differences will be highlighted and displayed next to each other.

Settings for transferring object addresses

With the setting, you can transfer the object addresses from comparison partner B to comparison partner A.

Note

For a successful transfer, the comparison objects must match in their fundamental parameters

For example, axes must match in their name, type and object address.

Introduction: ST, LAD/FBD, MCC detail comparison

The following section provides an overview of the function buttons and the menu commands for ST, MCC, and LAD/FBD detail comparisons.

Detail comparison function buttons

The following section provides an overview of the function buttons for ST, MCC, and LAD/FBD detail comparisons.

Table 5-33 Detail comparison function buttons

Button	Description
	Jump to next difference
	Jump to previous difference
	Transfer highlighted line(s) to source (from comparison partner B to A)
	Complete transfer of differences to the source (from comparison partner B to A)

Detail comparison menu

The following section provides an overview of the menu commands for ST, MCC, and LAD/FBD detail comparisons.

Table 5-34 Detail comparison menu

Menu	Description
Jump to next difference	Jumps to next difference
Jump to previous difference	Jumps to previous difference
Accept current line / Accept current network (LAD/FBD networks only)	Transfers highlighted line(s)/highlighted network to source (from comparison partner B to A)
Accept all differences	Transfers all differences (as regards current tab) to the source (from comparison partner B to A)
Display line numbers (ST only)	Displays the line numbers of comparison partners A and B
Ignore indents (ST only)	Indents in the source are ignored; you need to define this before accepting changes for the first time, as you cannot change the setting for the current comparison after you have done this.
Accept changes into project	Changes are accepted in the project and saved
Close and open editor	Opens comparison partner A in the editor. The detail comparison closes.
Close	Closes the detail comparison.

ST detail comparison

Introduction

The ST detail comparison allows you to run a textual comparison on ST sources. The compared sources are shown next to one another and the differences highlighted in color.

Missing lines (highlighted in gray) are added so that the display of identical lines can be synchronized.

In addition, an overview tab appears in the project navigator. In this case, all lines of the source code are compressed and any differences are highlighted. Simply click on the differences on the overview tab to jump directly to each difference in the source code.

The differences in the source code can be transferred either on a line-by-line basis or simultaneously by marking several lines at once. You have to open the ST editor for editing. You cannot edit in the detail comparison.

Note

Note the information provided in the status bar at the end of the working area.

Note

In isolated cases, a difference may be detected as a result of a control character such as a line break despite the fact that this is of no relevance to the source text.

Example: ST detail comparison

Example of an ST detail comparison

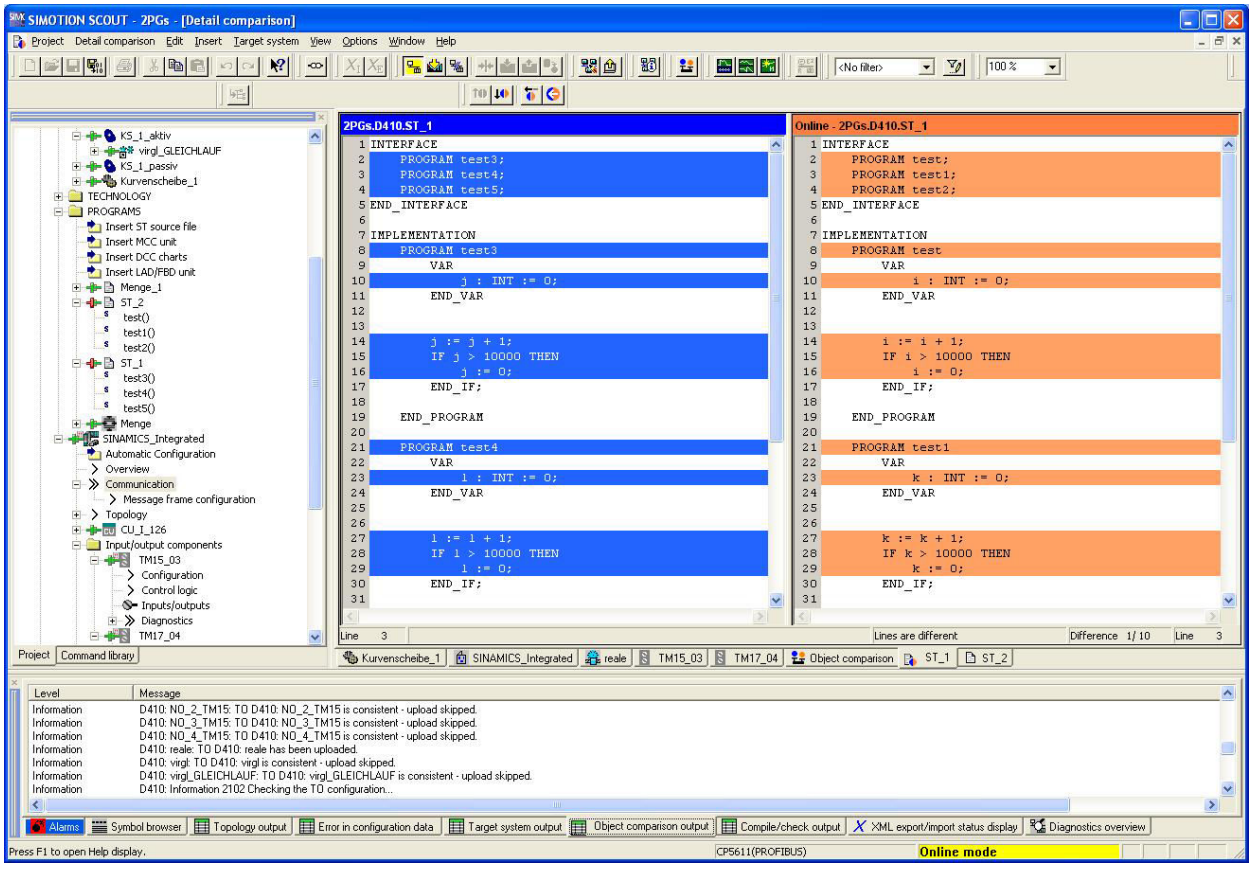


Figure 5-65 Example of ST detail comparison

MCC and LAD/FBD detail comparison of declaration tables

Introduction

The detail comparison helps you to identify differences between comparison partner A and B in the declarations. This concerns:

- MCC units and LAD/FBD units: Global unit declarations, "INTERFACE" section
- MCC units and LAD/FBD units: Global unit declarations, "IMPLEMENTATION" section
- MCC charts and LAD/FBD programs: POU local declarations

If differences have been detected within a tab during a detail comparison, this tab is underlined in the respective color for both comparison partners and appears in the foreground. If differences have been detected within a number of tabs, all the relevant tabs are underlined in the respective colors.

The reference order for the declarations is provided by comparison partner A. The order for comparison partner B is resorted accordingly, in order to see at a glance whether declarations

are missing or the extent to which declarations differ in terms of content. Color highlighting in the column with the line numbering indicates differences in the order.

Note

If the order differs for comparison partners A and B, you can only align them by accepting all the differences.

If there are differences between the declarations for comparison partners A and B, you can correct the differences identified line by line.

Special features of enumerations, structures, and pragmas

In future, pragmas will be used to mark the beginning of a new block both in the context of MCC and of LAD/FBD.

When the declarations of the two comparison partners are matched up, variables have to be located below a pragma for both comparison partners in order to be assigned to one another. If there is a pragma for one of the comparison partners, but not for the other one, the appropriate number of empty lines will be inserted for the comparison partner that does not have a pragma. Pragmas are assigned on the basis of names.

Note

If there are matching pragmas, you can accept subordinate variables to suit your specific requirements.

If the comparison partner has no matching assignment for a pragma, you must accept the pragma along with the variable declarations.

Example: Detail comparison of declaration tables

Example of a detail comparison for declaration tables

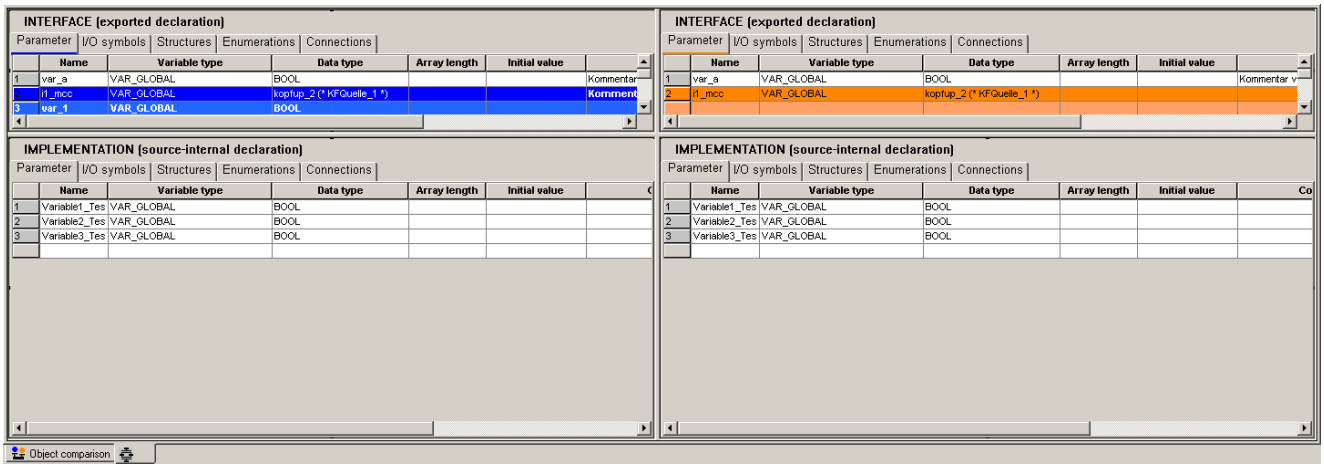


Figure 5-66 Example of a detail comparison for declaration tables

MCC detail comparison of charts

Introduction

The detail comparison is subdivided into the following areas: The upper area, which contains the local variable declarations for the comparison partner, and the lower area, which contains the two charts whose command blocks have been synchronized.

You can navigate between all the differences identified and align individual differences in accordance with your specific requirements.

In addition, an overview tab appears in the project navigator. This displays the differences in compact form. Simply click on the differences on the overview tab to jump directly to each difference in the chart.

Modules

In general, it is possible to zoom in on/zoom out from the module in the case of both comparison partner A and comparison partner B.

If two modules have been matched up with one another, the commands within each zoom level are compared with one another (as in the case of a chart without modules).

Note

At every zoom level, you have the option of viewing the differences in terms of command parameterization and accepting individual commands in accordance with your specific requirements.

If there is a module for only one comparison partner or if a module has been assigned to comparison partner A, but not to comparison partner B, although you can zoom in on/zoom out from the module, you can only accept the module in its entirety.

Command blocks

In order to analyze and align differences in terms of command block parameterization, you can open the generic detail comparison once a discrepancy has been identified.

Note

You can align differences in terms of parameterization by selecting checkboxes in accordance with your requirements. If a particular parameter cannot be aligned, no checkbox is made available.

Examples: MCC detail comparison of charts

Possible differences in the detail comparison

The following table provides you with an overview of the possible differences and how they are identified. This is illustrated on the basis of specific examples (see figure below).

Nature of difference	Identification
Differences in the local variable declarations	Differences are highlighted using the object comparison colors (1).
A command or several related commands are missing for comparison partner A or B	There is a corresponding gap in the chart for comparison partner A or B. Any subsequent elements that are assigned to one another become synchronized again after the difference. As regards the comparison partner, where the relevant command or set of related commands is/are present, this/these are displayed using the object comparison color of orange (2).
Differences in structures	In general, only structures of the same type are assigned to one another. Structures of different types are shown staggered. If there is a structure with subordinate commands for one comparison partner but the other comparison partner only features commands with a superordinate structure, the commands are still assigned.
There is a discrepancy between the parameterizations for a command or for several commands of the same type	The command blocks of the same type that are assigned to one another are displayed on one level and are color-coded using the object comparison colors (blue on the left and orange on the right) (3). When you double-click on a command where there is a discrepancy (whether for A or B), a dialog opens showing the parameterization differences, matched up and highlighted in color (see *Detail comparison command block*)
Differences in commented-out commands	The same rules apply as for commands. You also need to be aware of the following points: If the only difference between two commands that have been assigned to one another is that the command is commented out for one comparison partner but not for the other one, both commands are color-coded using the object comparison colors (blue on the left and orange on the right). Where commented-out commands have been assigned to one another but are subject to differences in parameterization, when you double-click on the command where there is a discrepancy, a dialog opens showing the parameterization differences, matched up and highlighted in color.
Differences in the brief comments	Background highlighted using object comparison colors
Parameterization differences affecting an ST zoom	When you double-click either of the ST zooms, the detail comparison window opens (only if there are parameterization differences).

Nature of difference	Identification
Parameterization differences affecting a comment block	When you double-click either of the comment blocks, the textual detail comparison window opens (only if there are parameterization differences).
Modules	Modules are always matched up for the comparison partners, if the majority of command blocks within the module are of the same command type. If this criterion is not met, the modules for both comparison partners are highlighted in color and shown staggered. If a module is only present for one of the two comparison partners, this is color-coded accordingly. There is a gap in the chart for the comparison partner that does not have this module. In general, it is possible to zoom in on/zoom out from the module in the case of both comparison partner A and comparison partner B. See description below.

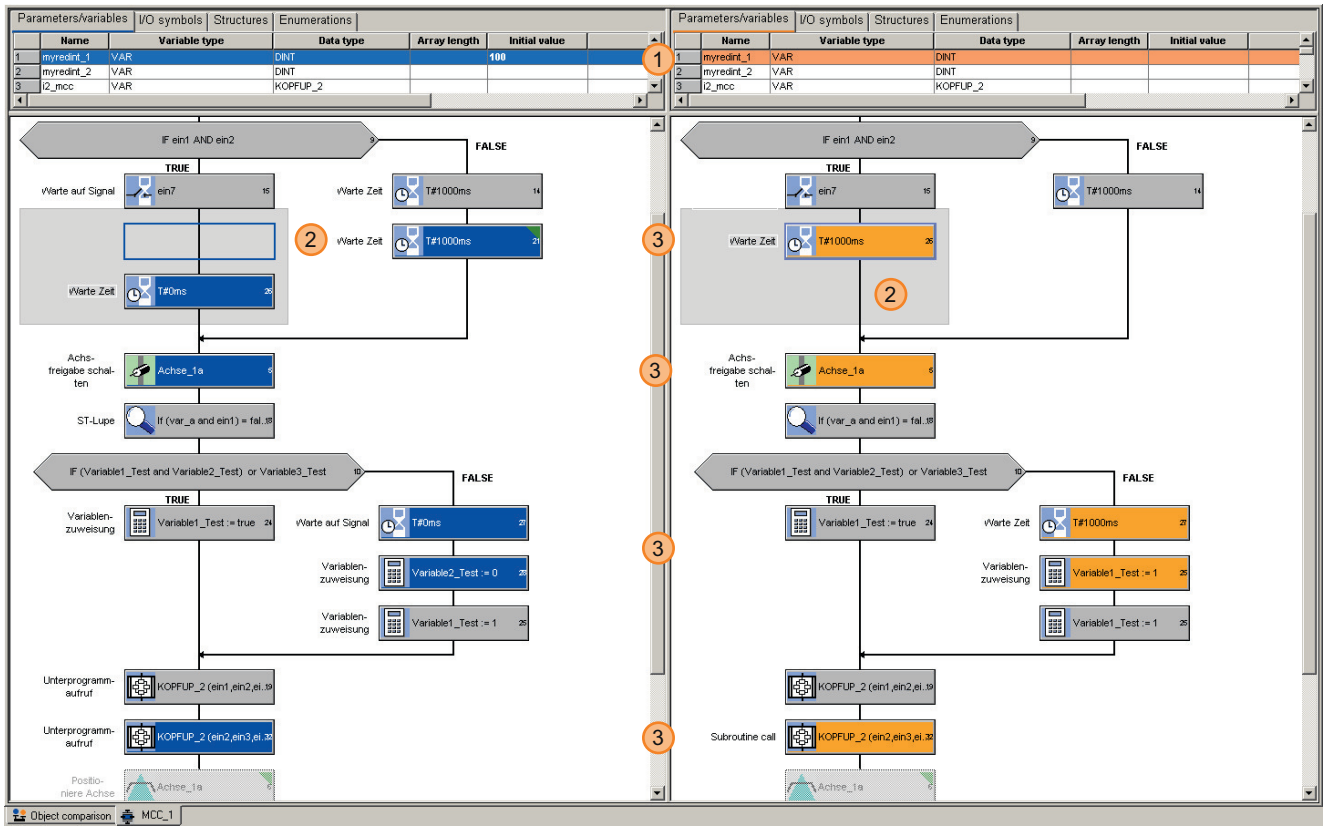


Figure 5-67 MCC detail comparison of charts

Detail comparison command block

The following example shows the detail comparison of a command block.

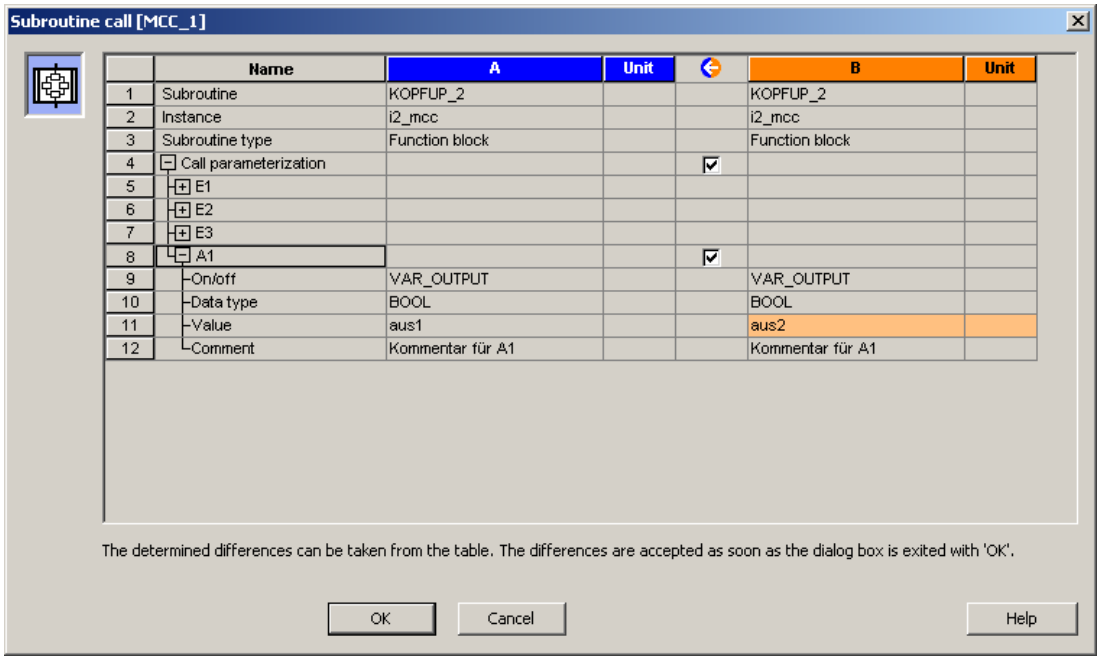


Figure 5-68 Detail comparison command block

Detail comparison overview tab

On the overview tab, you will find the following information in compact form:

- Differences in the numbers of commands (1)
- Differences in the parameterization of the commands (2)
- Missing elements for A or B (3)

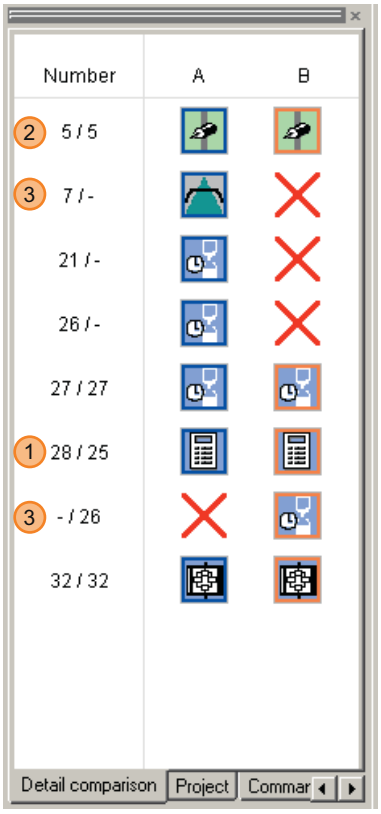


Figure 5-69 MCC detail comparison overview tab

LAD/FBD detail comparison of networks

Introduction

The detailed comparison is subdivided into the following areas: The upper area, which contains the local variable declarations for the comparison partner, and the lower area, which contains the two networks that have been synchronized accordingly.

You can navigate between all the differences identified and align individual differences in accordance with your specific requirements.

In addition, an overview tab appears in the project navigator. This displays the differences between the networks in compact form. Simply click the differences on the overview tab to jump directly to each difference in the network.

Assignment of networks

The networks are matched up on the basis of the comparison result content. Two networks are assigned to each other, if:

- They are 100% identical
- The structure of the two networks is absolutely identical
  (i.e. the number and position of blocks/elements are identical, all of these have a matching counterpart in the comparison partner, the connections are the same, the only differences relate to parameterization (different variables or constant values on the same pins) or instantiation (different instance names))
- Or if a relatively large number (for example, more than 50%) of elements can be assigned.

Detailed comparison of networks of different types (LAD vs. FBD or FBD vs. LAD)

LAD/FBD programs within an LAD/FBD unit are assigned on the basis of names. This makes it possible to assign two LAD/FBD POUs with the same name to one another even though the network type differs. If an LAD POU is assigned to an FBD POU, these POUs are marked as *non-identical* (i.e. aspect, comments, and formats).

When you open the detailed comparison, the differences can only ever be displayed in the same format. An attempt is made to switch POU B to the format used for POU A. If this is not possible, an attempt is made to switch POU A to the format used for POU B.

If the switch is not possible, the following error message is displayed:

- The detailed comparison is not possible.
- The network type of comparison partner B is switched over.

Note

The automatic switching process described above is only used for display purposes and is not saved automatically.

Examples: LAD/FBD detail comparison of networks

Possible differences in the detail comparison

The following table provides you with an overview of the possible differences and how they are identified. This is illustrated on the basis of specific examples (see figure below).

Nature of difference	Identification
Different PINs	Differences are highlighted using the comparison partner color.
Operations without a matching counterpart in the respective comparison partner	Operations are highlighted using the comparison partner color (1).
Differences in the network comments	Differences are highlighted using the comparison partner color (2).

Nature of difference	Identification
Network is missing completely for one comparison partner	The network that is present is color-coded (3).
Differences in the variables	Differences are highlighted using the comparison partner color (4).

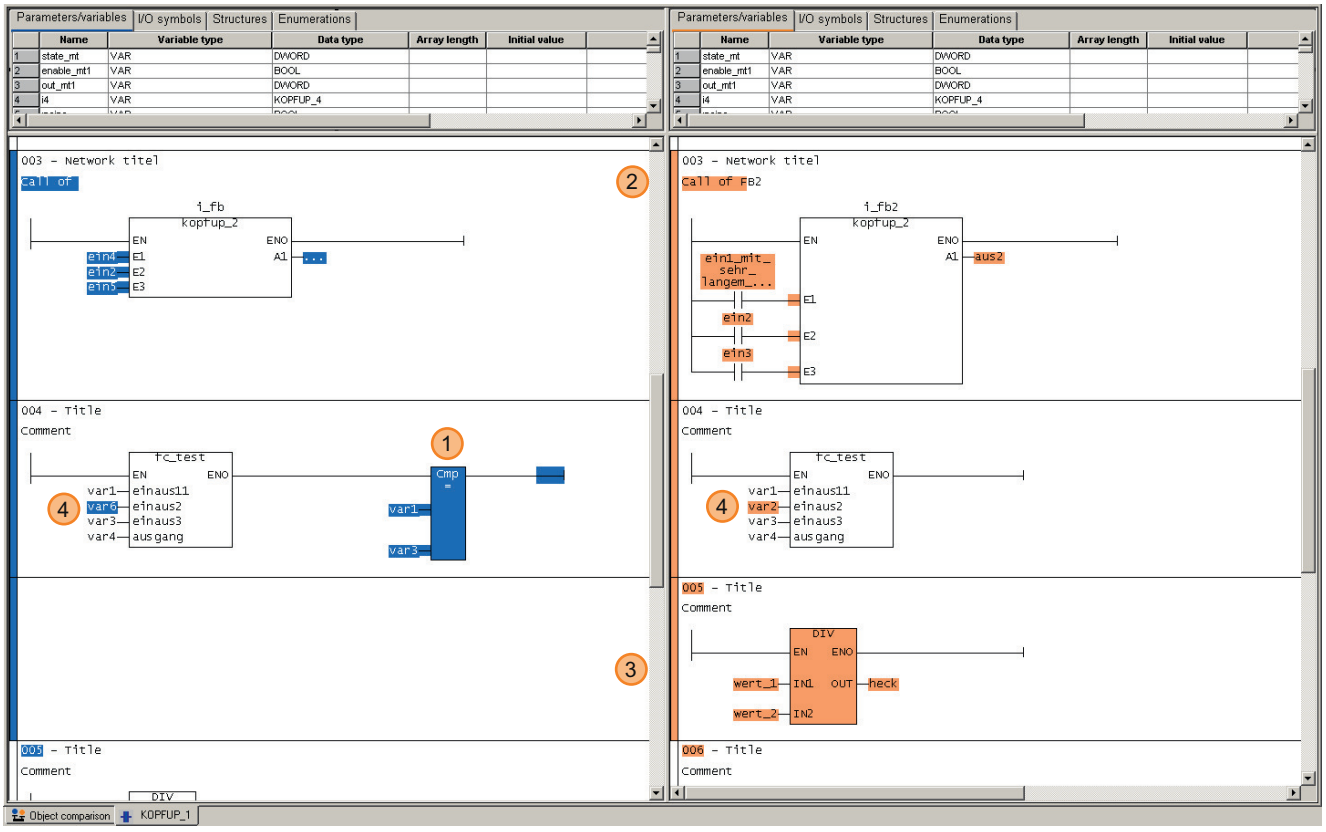


Figure 5-70 LAD/FBD detail comparison of networks

Detail comparison overview tab

The overview tab displays the following information in compact form for both LAD and FBD:

- Differences in the number of networks (1)
- Number of different networks (2)

- Missing networks for A or B (3)
- Areas with identical networks (4)

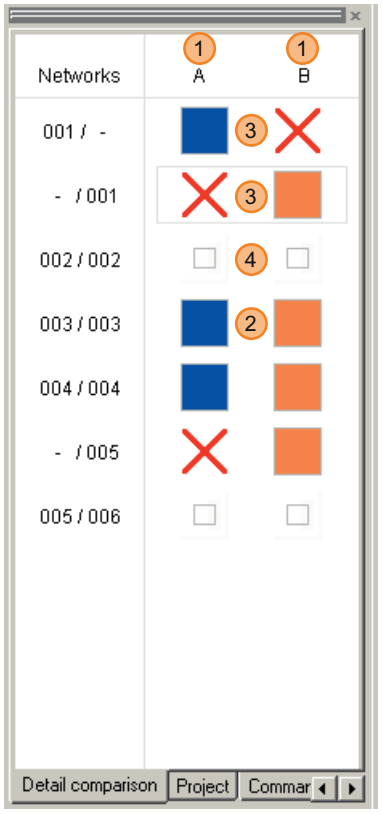


Figure 5-71 Detail comparison of LAD/FBD networks overview tab

DCC detailed comparison

Introduction

For SINAMICS DCC charts on the drive object (DO) and in the DCC library, a detailed comparison of two DCC charts can be started in the project comparison as of STARTER V4.4 / SCOUT V4.4. The detailed comparison of the DCC charts can be started from the object DCC chart in the project comparison.

The chart contents of two charts are compared on the basis of chart data, also known as chart sources, or compilations. The comparison result is displayed in tabular form. This feature is only available for SINAMICS DCC charts in SCOUT, not for SIMOTION DCC charts.

For comparison attributes which display a question mark or not equal sign in the object comparison, it is possible to open the detailed comparison.

The detailed comparison can be started for the following comparison attributes:

- Source code (time stamp comparison)
- Blocks
- Interconnections

- Execution groups
- @ parameters and external references

Requirements for the detailed comparison

- A CFC license and a CFC editor installation are required to start the detailed comparison for the "Source code (time stamp comparison)" comparison attribute. For this comparison attribute, comparisons are made on the level of the DCC chart data. The project must be converted to a suitable CFC data management when opening. If the DCC trial license is still available for activation, you can open the licensing box to activate the license.
  If no valid license is available, the following message is output and the operation is aborted: "DCC chart sources cannot be compared because a valid license has not been found".
- A CFC editor installation is required to start the detailed comparison for the "Execution groups", "Blocks" and "Interconnections" comparison attributes. For these comparison attributes, the compilation result, the so-called compilation, is compared with the DCC chart. This means that the result can differ from the "Source code (time stamp comparison)" comparison attribute. A CFC license is not required. The project must be converted to a suitable CFC data management when opening.
  If a CFC editor is not installed, the following message is output and the operation is aborted: "A CFC editor is not installed. DCC chart sources cannot be compared".
- Neither a DCC editor license nor a CFC editor installation are required to start the detailed comparison for the "@ parameter and external references" comparison attribute

Comparison attribute / requirement	CFC editor required	CFC license required	Project should be converted to CFC data management
Source code (time stamp comparison)	Yes	Yes	Yes
Blocks	Yes	No	Yes
Interconnections	Yes	No	Yes
Execution groups	Yes	No	Yes
@ parameters and external references	No	No	No

- The detailed comparison for DCC SINAMICS libraries is only possible in the offline-offline comparison.
- For calling up the detailed comparison in the offline-online comparison, the DCC chart data are required at comparison partner A. In the offline-offline comparison, the DCC chart data are only required at one of the two comparison partners.
- If the comparison results of the two comparison partners differ excessively, a meaningful comparison is not possible. A detailed comparison cannot be carried out.
- If conversion is required for comparison, note that reconversion is not possible. If a conversion is not wanted, you can create a copy of the projects, convert these projects and then run the comparison.

Note

If the CFC data management of a comparison project is not converted, an error message is output.

Note that the conversion cannot be undone. You can perform the conversion with a copy of the project and start the detailed comparison again.

Method of operation of the detailed comparison

The detailed comparison is offered when the  and  icons are display in the project comparison. Start the detailed comparison by clicking the respective icon.

The detailed comparison enables possible inconsistencies between the DCC charts to be precisely identified when going online or when activating the test mode in the CFC editor.

If the test mode is not switched on in the CFC editor or can only be switched on with limited functionality, then change to STARTER/SCOUT. You can then have SINAMICS display the differences in the DCC charts in the project comparison.

The detailed comparison is characterized by the following:

- If the comparison attributes (source code, blocks, interconnections, execution groups, @parameters) are not consistent, a comparison result is provided with detailed information.
- Tabular list of the comparison result.
- With an inconsistent comparison result, you can jump to the appropriate position in the DCC by double-clicking a difference in the table. This avoids unnecessary searching in the DCC chart. For this, you require the source of the DCC chart.
- As with the project comparison, the detailed comparison of DCC charts can be performed with the same comparison partners. This means that the detailed comparison can be performed between DCC charts and the project. The DCC charts and the project can be in the offline-offline or offline-online states respectively.
- If one of the comparison partners has a know-how-protected DCC chart, the detailed comparison of this DCC chart is only possible after specification of the login and password. The know-how protection of charts is only guaranteed when the chart sources are explicitly protected with know-how protection. This know-how protection is effective on the level of the chart sources, of the loaded execution code, for exported chart sources and in the detailed comparison.
  Even if the chart sources have been deleted in the project, the know-how protection must also be set for the protection of the export, compilation and detailed comparison.
- The DCC libraries are also contained implicitly as "DCC chart" comparison object.

The following figure illustrates the relationships between the comparison attributes and the data sources:

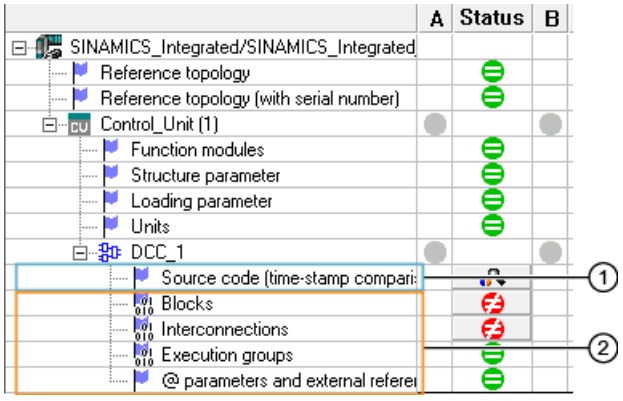


Figure 5-72 Relationships between comparison attributes and data sources

① Chart data

② DCC compilation + DCC parameter

Overview of comparison values

The values that are taken into account in the DCC detailed comparison are shown in the following table.

Value	Description
Chart properties	
Name	DCC name
Chart version in the CFC editor	Version of the chart that has been set in the DCC chart (default value: 0.0001)
Parameter number base	Base value for parameter numbers which has been set for the DCC chart
Libraries	
<Libraries>	List of the libraries used and their version numbers
Blocks	
<Blocks>	List of all blocks in the DCC chart with their properties (e.g. position, inputs, outputs or interconnections)
Block sequence	
<Execution groups>	List of the execution groups and the sequence of the blocks in them
Execution group sequence	
<Execution groups>	List of the execution groups in their set sequence
Chart structure	
<Subcharts>	List of the subcharts and their position in the DCC chart

Example: DCC detailed comparison

Example

You can only start the detailed comparison from the project comparison by clicking the or button. You cannot start the detailed comparison from the CFC editor.

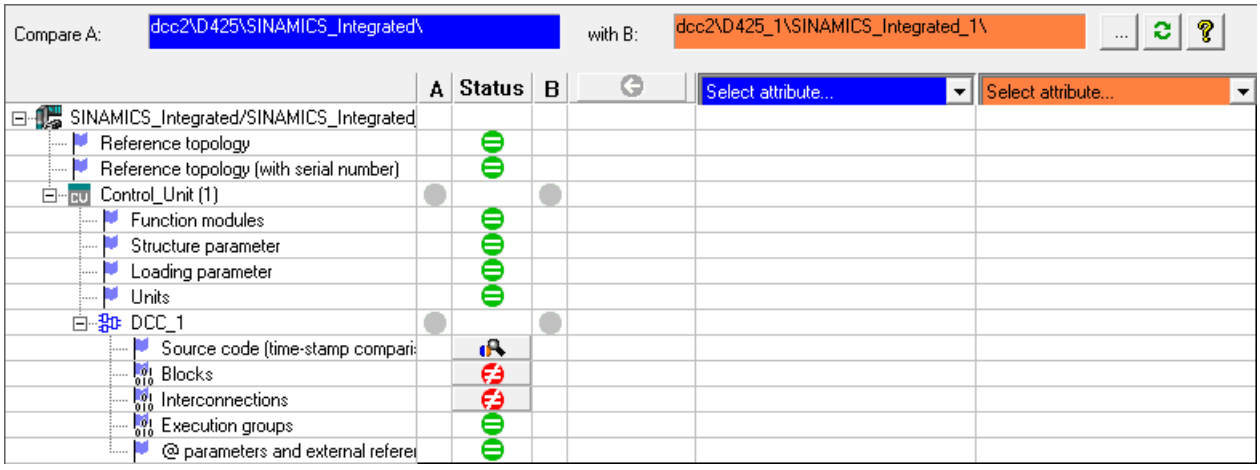


Figure 5-73 Example: Project Comparison

The detailed comparison that opens provides a tabular display of all differences.

		dcc2\D425\SINAMICS_Integrated\	dcc2\D425_1\SINAMICS_Integrated_1\
1	**Chart properties**		
2	**Libraries**		
3	**Blocks**		
4	ADD	1	1
5	Position	A(55:168)	A(55:168)
6	X1	2.0	1.0
7	X2	4.0	2.0
8	X3	6.0	3.0
9	X4	8.0	4.0
10	Y	0.0	0.0
11	Interconnection to	2.X1	
12			
13	SUB	2	
14	Position	A(65:156)	
15	X1	0.0	
16	Interconnection to	1.Y	
17	X2	0.0	
18	Y	0.0	
19			
20	**Block sequence**		
21	Execution group 1	DCC_1	DCC_1
22		1 (ADD)	1 (ADD)
23		2 (SUB)	
24			
25	**Execution group sequence**		
26	Execution group 1	DCC_1	DCC_1

Figure 5-74 Example: Display of the differences in the detailed comparison

Double-click a difference in the table to jump to the appropriate position in the DCC chart.

Detail comparison of simply structured data

Introduction

The detail comparison can help you to analyze and align simply structured data, for example, for CPU system variables and TO interconnections.

User interface

The following figure shows the structure of the detail comparison.

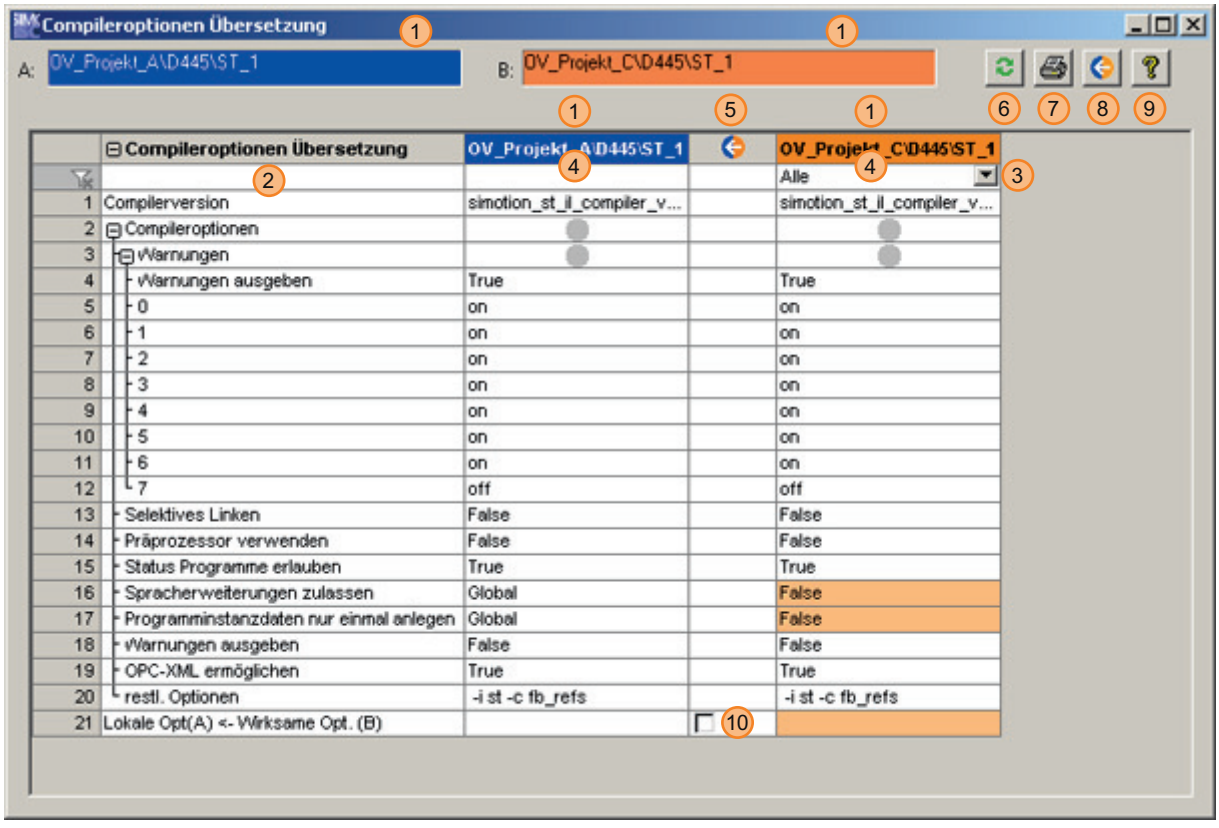


Figure 5-75 User interface of the detail comparison

Table 5-35 The user interface elements of the detail comparison

No.	Description
1	Comparison partner A and comparison partner B
2	Comparison tree made up of comparison partner elements
3	Filter The filter can be used to determine whether the entire data volume should be displayed or just the differences. The selection is set by default to **Differences only**.
4	Result of comparison
5	Data transfer of the selected differences See note below.

No.	Description
6	Update comparison
7	Print comparison
8	Data transfer of all differences See note below.
9	Call up online help
10	Checkbox for data transfer (if no transfer is possible, the checkbox is not displayed) See note below.

Note

Data transfers are not always possible, for example, during TO interconnections or generally for write-protected projects. In this case, checkboxes are not displayed for the transfer

Rules for data transfer

- Checkboxes for the data transfer are only displayed once a difference has occurred. If there are not any differences on one level, checkboxes are also not displayed on the higher level.
- As a general rule, data is structured in hierarchical order. As such, you can either fully transfer a hierarchy level with subordinate hierarchy levels/objects or just individual subordinate hierarchies/objects.
- Please note the following when changing the transfer selection:
  - Activate or deactivate the superordinate level:
    The checkboxes of the hierarchically subordinate elements assume the same value.
  - Activate or deactivate subordinate elements:
    The checkbox of the subordinate hierarchy level is disabled, as long as activated and deactivated checkboxes are located below the level.

- The **Data transfer** button (see figure below user interface, No. 5) is only activated once at least one checkbox has been activated for the transfer.
- The data transfer is only activated by clicking the **Data transfer** button.
  - If you only wish to transfer the selected differences, click on the **Data transfer** button in the column header (see figure below user interface, No. 5).
  - If you wish to transfer all differences, click on the **Data transfer** button on the toolbar (see figure below user interface, No. 8).

Note

In certain cases, a difference may still be displayed following successful transfer, for example, different CPUs may have a different number of MotionTasks.

Note

Differences in the source properties

A unit that does not exist can be transferred from comparison partner B to A. In this case, the new unit is created in the project and also the compile options. If the compilation result is now updated, the comparison still shows a difference in the compilation settings. This can be resolved easily by recompiling the CPU.

Data transfer failure

Errors can occur during the data transfer. For example, the transfer of a non-existent A1 IO variable can fail if an A1 global device variable already exists. If the transfer fails, the following message is output: **Errors occurred during data transfer!**.

The comparison is updated following the transfer. In the case of transfer failure, the differences continue to be displayed.

Examples: Detail comparison of simply structured data

The following section shows an example of a detail comparison of the HW Config system interface and global device variables.

HW Config (system interface)

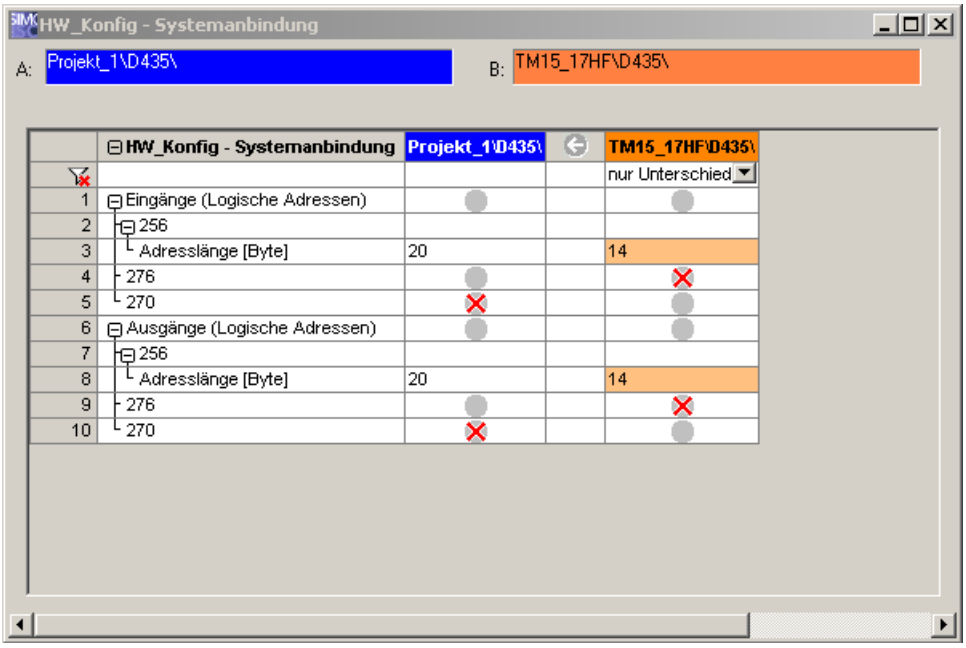


Figure 5-76 Detail comparison of HW Config system interface

Global device variables

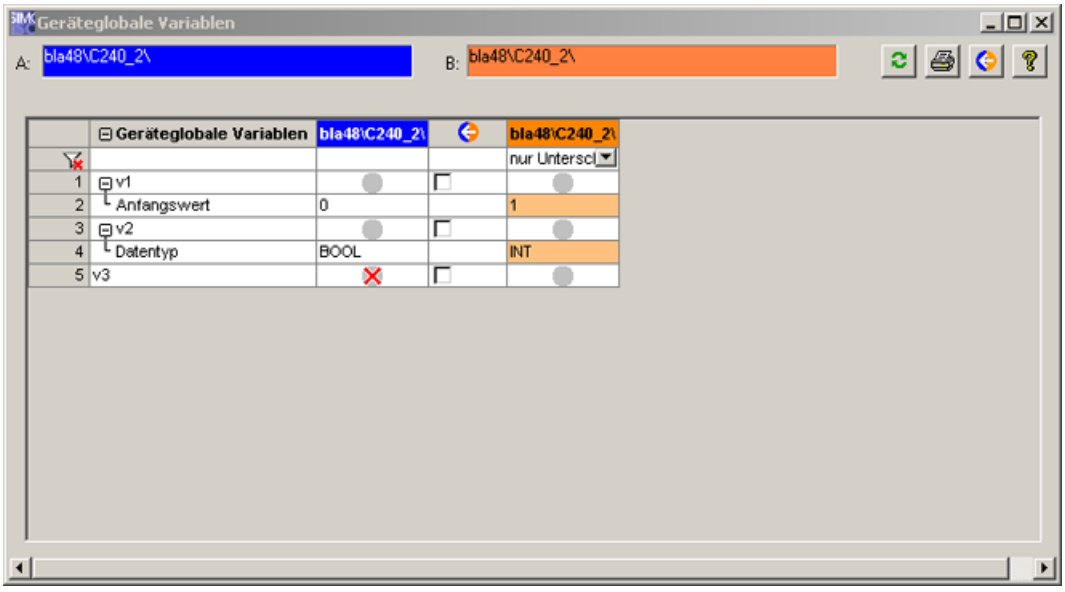


Figure 5-77 Detail comparison of global device variables

TO/DO detailed comparison

Differences between TO/DO detailed comparison and experts list comparison

The differences between the TO/DO detailed comparison and expert list comparison are summarized in the table below.

Table 5-36 Comparison of detailed comparison and Expert list comparison

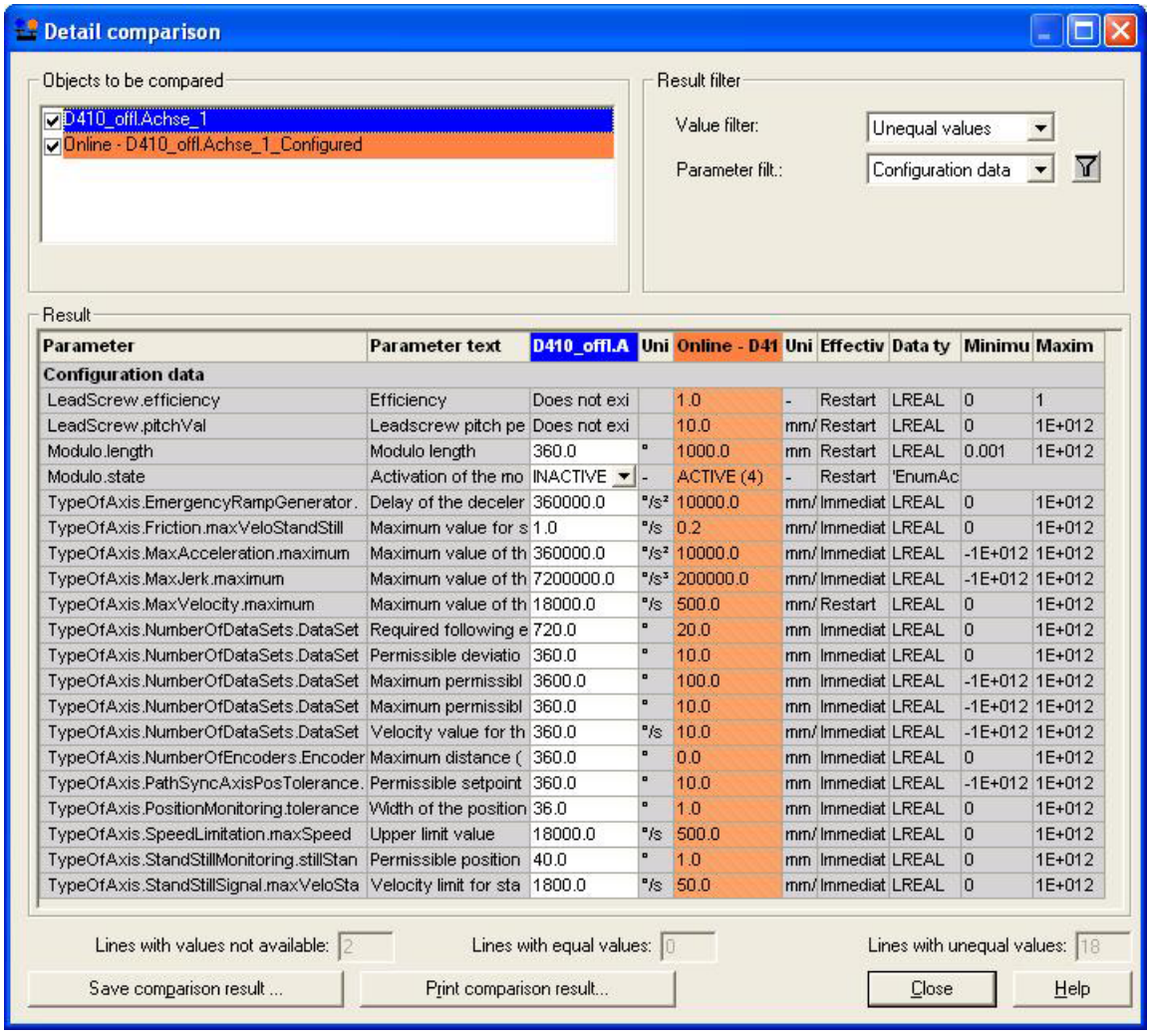
Feature	Detailed comparison	Expert list comparison
Number of comparison objects	1	1 ... n
Origin of comparison objects	Objects from different projects	Objects from the open project only
Values to be compared[1)]	Configured values; actual values are not included in the comparison	Actual values
Indication of differences (structural differences)	Full indication of all parameters (DOs) or system variables and configuration data (TOs), regardless of whether they occur in comparison partner A, B, or both.	Only the parameters and system variables and configuration data of comparison partner A are displayed. Therefore, you only see whether or not the parameters contained in the reference object are present in the comparison objects. Parameters that only the comparison objects contain (not the reference object) are not checked.
Access to online values	Always possible	Only if the structure is consistent with the offline object
Options for changing data in the comparison objects[2)]	Change to data in comparison partner A (reference object) only during offline comparison **Note**: You can use the context menu for the table cells of comparison partner B to transfer the value from B to A.	Change to data in all objects shown in the comparison
Topicality of online data	Snapshot from execution time of object comparison	Snapshot from execution time of expert list comparison

[1)] Differences between configured values and actual values are only of any consequence if the comparison partner is an online TO. Offline, parameters only ever have one (configured) value.
There is only one value for online objects of DOs, and the following applies:
actual value = configured value.

[2)] Fields hatched out in the detailed comparison cannot be edited.

TO detailed comparison

Example of a TO detailed comparison:



Parameter	Parameter text	D410_offl.A	Uni	Online - D41	Uni	Effectiv	Data ty	Minimu	Maxim
Configuration data									
LeadScrew.efficiency	Efficiency	Does not exi		1.0	-	Restart	LREAL	0	1
LeadScrew.pitchVal	Leadscrew pitch pe	Does not exi		10.0	mm/	Restart	LREAL	0	1E+012
Modulo.length	Modulo length	360.0	°	1000.0	mm	Restart	LREAL	0.001	1E+012
Modulo.state	Activation of the mo	INACTIVE	-	ACTIVE (4)	-	Restart	'EnumAc		
TypeOfAxis.EmergencyRampGenerator.	Delay of the deceler	360000.0	°/s²	10000.0	mm/	Immediat	LREAL	0	1E+012
TypeOfAxis.Friction.maxVeloStandStill	Maximum value for s	1.0	°/s	0.2	mm/	Immediat	LREAL	0	1E+012
TypeOfAxis.MaxAcceleration.maximum	Maximum value of th	360000.0	°/s²	10000.0	mm/	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.MaxJerk.maximum	Maximum value of th	7200000.0	°/s³	200000.0	mm/	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.MaxVelocity.maximum	Maximum value of th	18000.0	°/s	500.0	mm/	Restart	LREAL	0	1E+012
TypeOfAxis.NumberOfDataSets.DataSet	Required following e	720.0	°	20.0	mm	Immediat	LREAL	0	1E+012
TypeOfAxis.NumberOfDataSets.DataSet	Permissible deviatio	360.0	°	10.0	mm	Immediat	LREAL	0	1E+012
TypeOfAxis.NumberOfDataSets.DataSet	Maximum permissibl	3600.0	°	100.0	mm	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.NumberOfDataSets.DataSet	Maximum permissibl	360.0	°	10.0	mm	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.NumberOfDataSets.DataSet	Velocity value for th	360.0	°/s	10.0	mm/	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.NumberOfEncoders.Encoder	Maximum distance (	360.0	°	0.0	mm	Immediat	LREAL	0	1E+012
TypeOfAxis.PathSyncAxisPosTolerance.	Permissible setpoint	360.0	°	10.0	mm	Immediat	LREAL	-1E+012	1E+012
TypeOfAxis.PositionMonitoring.tolerance	Width of the position	36.0	°	1.0	mm	Immediat	LREAL	0	1E+012
TypeOfAxis.SpeedLimitation.maxSpeed	Upper limit value	18000.0	°/s	500.0	mm/	Immediat	LREAL	0	1E+012
TypeOfAxis.StandStillMonitoring.stillStan	Permissible position	40.0	°	1.0	mm	Immediat	LREAL	0	1E+012
TypeOfAxis.StandStillSignal.maxVeloSta	Velocity limit for sta	1800.0	°/s	50.0	mm/	Immediat	LREAL	0	1E+012

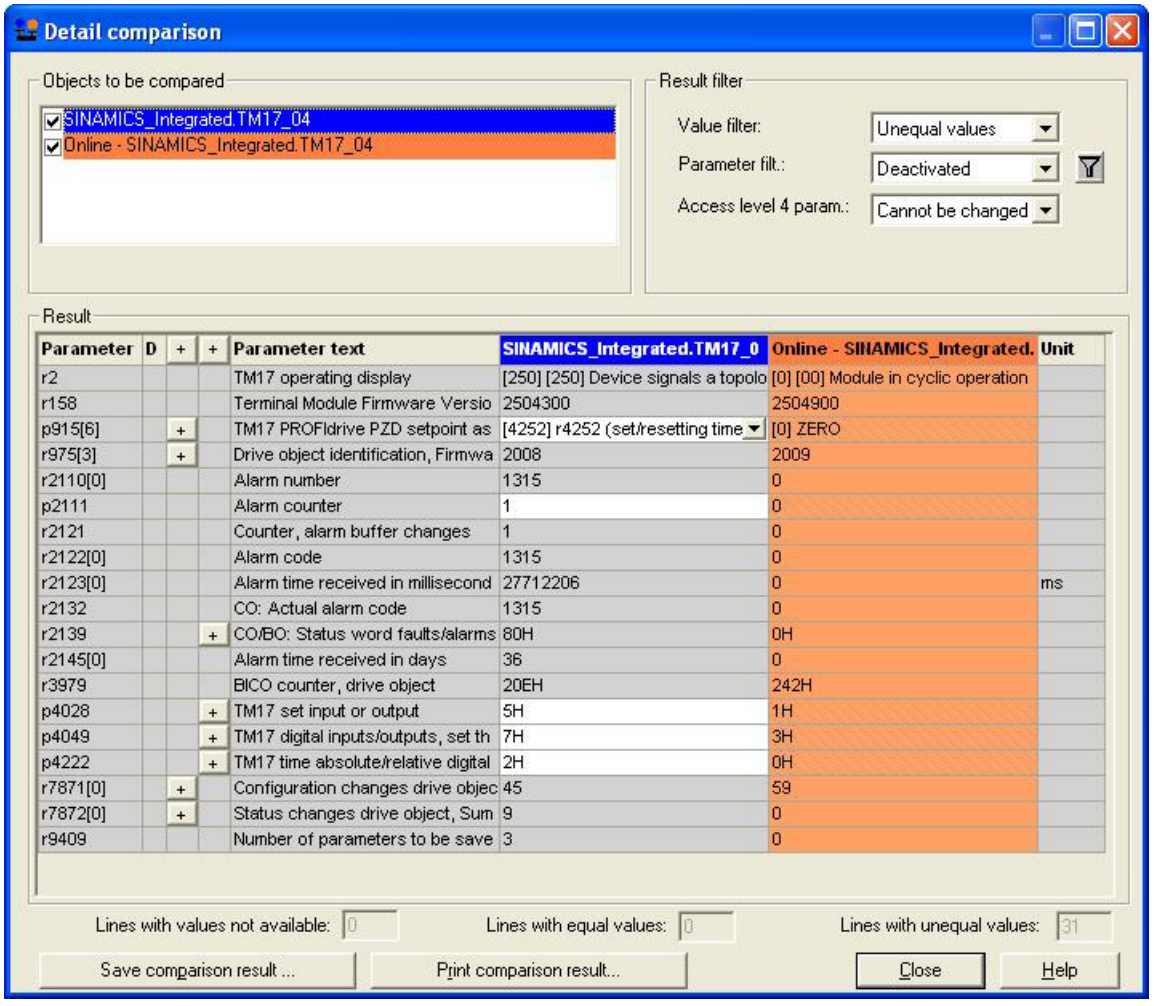
Example of a TO detailed comparison

Note

For more detailed information, refer to the section titled Differences between TO/DO detail comparison and expert list comparison and, in the basic functions, Comparing expert lists.

DO detail comparison

Example of a DO detailed comparison:



Parameter	D	+	+	Parameter text	SINAMICS_Integrated.TM17_0	Online - SINAMICS_Integrated.	Unit
r2				TM17 operating display	[250] [250] Device signals a topolo	[0] [00] Module in cyclic operation	
r158				Terminal Module Firmware Versio	2504300	2504900	
p915[6]		+		TM17 PROFIdrive PZD setpoint as	[4252] r4252 (set/resetting time	[0] ZERO	
r975[3]		+		Drive object identification, Firmwa	2008	2009	
r2110[0]				Alarm number	1315	0	
p2111				Alarm counter	1	0	
r2121				Counter, alarm buffer changes	1	0	
r2122[0]				Alarm code	1315	0	
r2123[0]				Alarm time received in millisecond	27712206	0	ms
r2132				CO: Actual alarm code	1315	0	
r2139			+	CO/BO: Status word faults/alarms	80H	0H	
r2145[0]				Alarm time received in days	36	0	
r3979				BICO counter, drive object	20EH	242H	
p4028			+	TM17 set input or output	5H	1H	
p4049			+	TM17 digital inputs/outputs, set th	7H	3H	
p4222			+	TM17 time absolute/relative digital	2H	0H	
r7871[0]		+		Configuration changes drive objec	45	59	
r7872[0]		+		Status changes drive object, Sum	9	0	
r9409				Number of parameters to be save	3	0	

Figure 5-78 Example of a DO detailed comparison

Note

For more detailed information, refer to the sections titled Differences between TO/DO detail comparison and expert list comparison and Comparison - Expert list.

Display filter for DO detail comparison

The display filter for the DO detail comparison is the same as for the expert list, with the following exception:

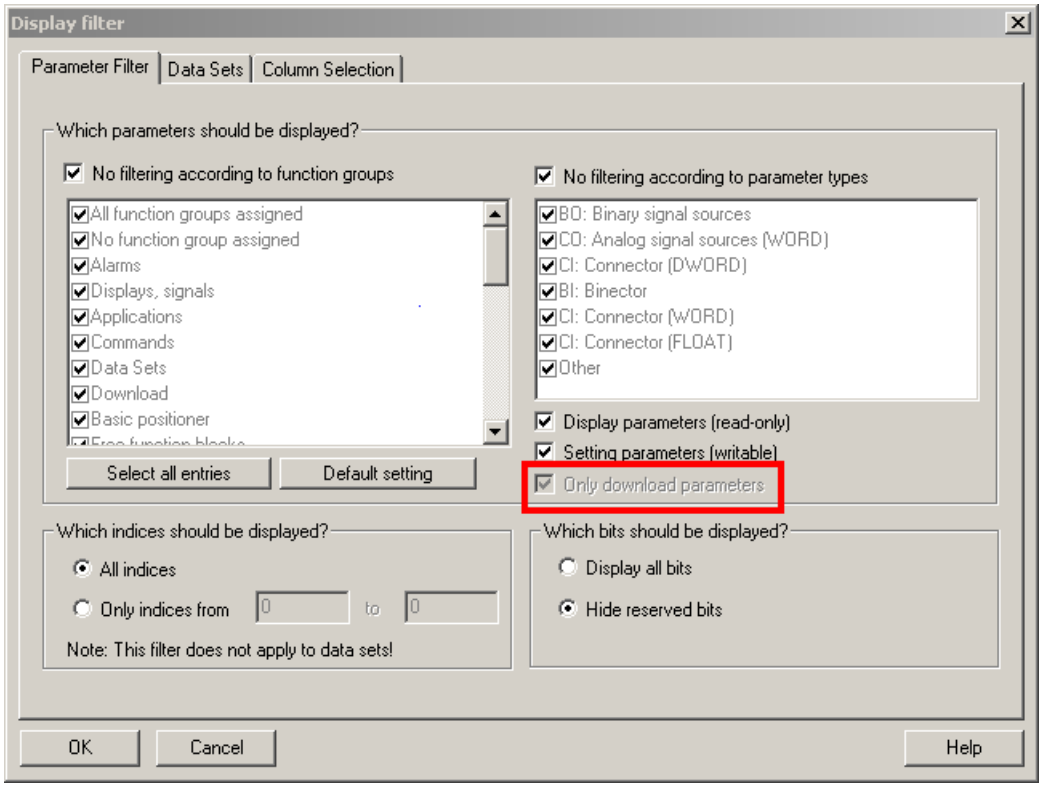


Figure 5-79 Display filter for DO detail comparison

The **Only download parameters** setting on the **Parameter filter** tab is always activated and cannot be changed. This ensures the detail comparison only includes parameters involved with the download. These are those parameters whose values will actually be loaded into the target device for a download from the PG. In contrast, adjustable parameters can only be changed online.

Note

For more information, refer to AUTOHOTSPOT.

5.3.4.5 Data transfer

Function

You can use the **Data transfer** function in the object comparison to transfer the data of the comparison object (B) to the open project (A). You can transfer individual data items from the detailed comparison. You can run a complete online calibration from B to A using the function **Download into PG/PC** or from A to B using **Download to target device**.

Note

In the event of errors when downloading to the PG, note the information in the detailed view on the **Object comparison** tab.

Note

You have to save in order to achieve online consistency after the data transfer of TOs. Programs are compiled after the data transfer, whereby these can become implicitly consistent.

Rules for data transfer

General rules for data transfer

The following rules apply to transfer procedures:

- A transfer is only possible from B to A.
- When you transfer an object, the complete content is transferred (including compiler settings, for example) and overwrites the data in the reference project (A). This means changes that you made in your project may be lost!
- If the comparison partner is a target system, the corresponding supplementary data must be downloaded. If this is not the case, the transfer cannot be performed.
- DOs cannot be transferred.
- A TO/program can be transferred to an existing TO/program. TOs/programs can be created in cases where there is no TO in A, but there is one in B, for example. TOs or a program entry in the execution system then still need to be linked manually.
- TOs/program can also be deleted by transferring them, e.g. if there is a TO/program in A, but not in B.
- When entire objects are transferred, the object address is also transferred (see Duplicate assignment of an object address (Page 3452)).
- If you only wish to transfer individual pieces of data from an object, rather than an entire object, it is only possible to do this in the detail comparison. Here, you can transfer one particular change to an ST program, for example (see Detailed comparison (Page 3422)).

Data transfer of POUs

MCC and LAD/FBD POUs are elements that you can accept individually in the object comparison. All the other elements (e.g. variables) can only be accepted in the specific detailed comparisons.

Rules for transferring:

- When POUs are transferred, the data is transferred in its entirety, including for example the properties (creation type, author, comment, etc.).
- If an object is present only for B, the POU is created for A during transfer. (A transfer is only possible if the higher-level source for A already exists.)
- If an object is present only for A, the POU is deleted for A during transfer.
- If the comparison partner is a target system, the corresponding information must be available online (see above: General rules for data transfer).
- If the source containing the POU is know-how protected (for A and/or B), then special rules apply (see below: Data transfer with know-how protected projects).

Data transfer for write-protected projects

If projects have been created in older SCOUT versions, you can open them as write-protected projects following an upgrade. The write-protected project allows you to perform an object comparison or detail comparison, but no data transfer is possible.

Data transfer with know-how protected projects

You can transfer know-how protected objects. You must be logged in to ensure that aspects are displayed and that you are able to transfer individual POUs.

Procedure for data transfer

To transfer data, proceed as follows:

1. Place a check mark in the check box (see diagram below) for the object you want to transfer.

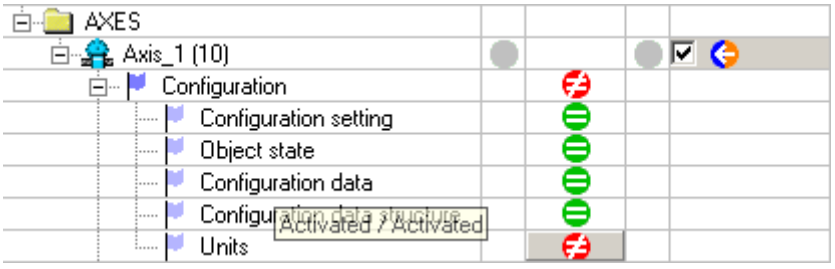


Figure 5-80 Selection and transfer of objects

2. Start the transfer by clicking the **Data transfer** button, and confirm the dialog that follows with **OK**.

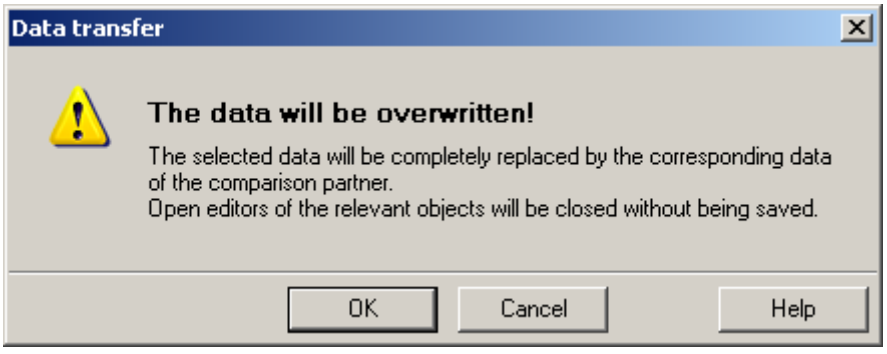


Figure 5-81 Data transfer dialog

3. Once the data have been transferred successfully, the check box is replaced with a check mark. The result of the comparison in the object comparison is then no longer up to date. This is indicated in the status bar at the end of the view.
You can, however, perform addition detail comparisons and data transfers without updating the comparison.

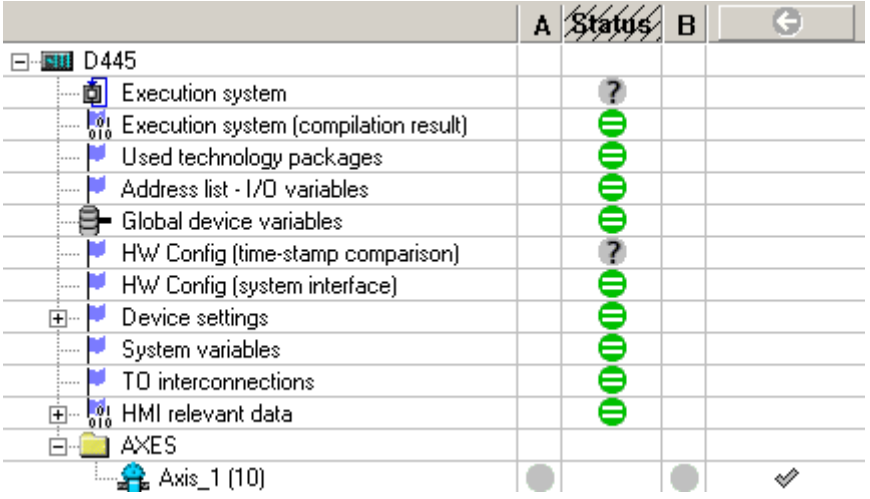


Figure 5-82 View following transfer

Duplicate assignment of an object address

When transferring entire objects, the object address is also transferred. Because the object address in the address space of a SIMOTION device must be unique, address conflicts may arise. Transfers which would result in duplicate assignment are not carried out. The object affected by the address conflict is entered in the log in the detailed view.

As of V5.1, it is possible with an occupied object address to perform a transfer.

To do so, click on the "Settings for transferring objects" check box and select the "Transfer without object address (recommended)" option or change the object address (see Chapter Changing the object address (Page 3422)).

5.3.5 Overview of comparison attributes

Below you will find an overview of the comparison attributes, along with their associated aspects/sub-aspects, for SIMOTION, SINAMICS, and MICROMASTER objects.

The comparison attributes are made up of several comparison features. There are various reasons for differences in comparison attributes. A difference in the compilation result may result from program changes (source code), for example, and/or a change to the compiler settings. The tables below describe the comparison attributes for the individual object types. These should help you to quickly establish the causes of a difference if a detailed comparison is not available.

We recommend the following procedure for efficiently finding the reason for a difference to a comparison attribute:

- If necessary, compile projects (A + B) if the compilation results are no longer up-to-date
- Start the detailed comparison, if available
- Search for the description of the comparison attribute in the object table
- Check the comparison features

Note

In conjunction with the detailed comparison, the term "scalar" means that only a single data item, such as the device name, is compared. Differences are then displayed in a dialog box.

Note

Online/offline comparison

- Please note that no current parameter values are compared during an online/offline comparison. Only parameter differences which are relevant to the download are displayed.
- A statement is not possible for the HW Config time stamp and the system display in an online/offline comparison. Therefore, a "?" is displayed in this case.

5.3.5.1 SIMOTION objects

SIMOTION device

Table 5-37 Comparison attributes of a SIMOTION device

Comparison attributes of a SIMOTION device	Aspects/Sub-aspects	Detailed comparison / scalar
Execution system	Configuration of the execution system: • The assignment of programs to the tasks • Configuration of tasks in the **Task configuration** tab • System cycle settings using the **Set system cycles...** function (Servo, Ipo, Ipo_2, TControl) **Note:** Compile the project prior to starting the project comparison. In this way, you will obtain a meaningful comparison result.	Execution system
Execution system - compilation result	The part which results from the project's compilation process. The compilation result includes: • The configuration of the execution system (see above) • The technology packages used	No detailed comparison
Selected technology packages	The technology packages selected in SCOUT with associated version.	Used technology packages
Address list of I/O variables	• Configuration of I/O variables **Note**: Structural changes to the I/O variables are also included in the **HMI-relevant data** comparison attribute.	Address list of I/O variables
Global device variables	• Configuration of global device variables **Note**: Structural changes to the global device variables are also included in the **HMI-relevant data** comparison attribute.	Global device variables
HW Config (time stamp-comparison)	Comparison of the hardware configuration is based solely on a time-stamp comparison. The time stamp of the most recent change is used. If a comparison is carried out on this basis, a statement on the equivalence is only possible when the time stamp is the same. If the time stamp is different, the hardware configuration may be the same or different. This will be the case if the most recent change has been undone, for example. STEP7 SIMATIC Manager offers a block comparison for such cases. **Offline comparison** The comparison is undertaken as described above. **Online comparison** A statement is not possible for the HW Config time stamp comparison. Therefore, a "?" is displayed in this case. The statement is that the hardware configuration is not included in the comparison. Displaying this attribute prevents the SIMOTION device from being displayed as inconsistent in the project navigator; however, everything is the same in the object comparison (the only difference is in HW Config).	No detailed comparison

Comparison attributes of a SIMOTION device	**Aspects/Sub-aspects**	**Detailed comparison / scalar**
HW Config (system integration)	The data from the hardware configuration which is included in the SIMOTION configuration: • I/O addressing (in HW Config) • DP cycle (bus cycle) • Equidistant bus cycle **Online comparison** A statement is not possible for the HW Config system display. Therefore, a "?" is always displayed in this case. The comparison attribute is only available offline.	HW Config (system interface)
Device settings		No detailed comparison
	• Device name	Scalar
	• Device address Required for distributed synchronous operation; see Object properties (Object address tab)	Scalar
	• Device type	Scalar
	• Behavior of dynamic data during STOP/RUN transition	Scalar
	• Fixed process image	Scalar
	• Time-of-day synchronization	Scalar
	• Fast I/O configuration (onboard I/Os, TM15/TM17 High Feature) [1)]	No detailed comparison
	Devices loaded with a SCOUT/STARTER version <= V4.0: This comparison attribute is not available.	
System variables	• Configuration of system variables in the SIMOTION device	Detailed comparison, CPU system variables
TO interconnections	• Interconnections of TOs on the device	Detailed comparison, technology object interconnections
HMI-relevant data	Records the structural setup (data type of variables, number of variables) of the data which an HMI system can access. If the setup changes, the HMI configuration must be updated. • Structure information of I/O variables • Structure information of global device variables	No detailed comparison
For every library used: Compilation result <Libname>	• Compilation result of the specific device version This comparison attribute is only shown if a library is used. A library comparison can be carried out for a detailed comparison.	No detailed comparison
1) Refers to the appropriate telegram of the fast I/O configuration. A difference is due to a discrepancy in terms of the fast I/O configuration (see drive units), or the failure to carry out HW Config alignment.		

Technology object (TO)

Table 5-38 Comparison attributes of a technology object

Comparison attributes of a technology object (TO)	Aspects/Sub-aspects	Detail comparison/ Scalar
Configuration	Contains the configuration of a technology object.	No detail comparison
	• Configuration setting For example, linear/rotary axis; technology object activated/deactivated	Scalar
	• Object state Values: Activated/deactivated	Scalar
	• Configuration data (see Comparing expert lists) Records the values and units	Detail comparison, expert list
	• Configuration data structure Records the data volume, for example the number of data sets or encoders	Detail comparison, expert list
	• Units (Indirect influence of unit configuration[1)])	Detail comparison, unit configuration
System variables	• The system variables of the TO Records the values and units (Indirect influence of unit configuration and configuration settings[1)])	Detail comparison, expert list
Alarm configuration	• Configuration of TO alarms (see TechnologicalFaultTask)	Detail comparison, alarm configuration
HMI-relevant data	Records the structural setup (data type of variables, number of variables) of the data which an HMI system can access. If the setup changes, the HMI configuration must be updated. Data included:	No detail comparison
	• The configuration data	Detail comparison, expert list
	• The object address (see TO properties)	Scalar
	Indirect influence of unit configuration and configuration settings[1)]	
Configuration data For TO cam only!	Configuration data of cam disc (e.g. support points, interpolation, scaling)	Detail comparison, configuration data
[1)] The unit and value of a technological parameter are always viewed separately (i.e. not the resultant parameter). This means: If a unit configuration is changed, the configuration data and system variables will be different to the original setting, even if the values in the new unit have been converted and the resultant variable is the same. A linear axis/pressure uses different physical variables to a linear axis/force; even if the numerical values are the same, these different variables will result in a difference. Not applicable to cams.		

Note

Modified data is only considered in the comparison once **Copy current data to RAM** has been performed.

Source

The following overview applies to ST, LAD, FBD, and MCC.

Note

Differences in the source after recompilation

If you recompile the source code of projects in version V4.0 or V4.1 in SCOUT in a newer version, differences for the sources can be displayed in a project comparison (old with new). This is because the signatures after the compilation are different for versions V4.0 and V4.1 and newer versions. In newer SCOUT versions, the signature is formed via the graphic programming languages LAD/FBD/MCC and not via ST.

Note

Sources with larger version jumps

With larger version jumps (e.g. from 3.2 to 4.1), the signatures of functions can change (for example for system functions, new parameters are added that are assigned useful default values). Therefore, the code generated by MCC changes during the compilation; the generated code however is still functionally identical.

Simultaneously, the difference is detected in the parameterization and also when editing the source, and can be adapted there.

As long as the source is not touched, it is identical in the project comparison.

If the source is edited on the V4.4 side, the project comparison and the detailed comparison find differences. If, however, the source is compiled without editing, only the project comparison finds differences, not the detailed comparison.

Table 5-39 Comparison attributes of a source

Comparison attributes of a source	Aspects/Sub-aspects	Detailed comparison / scalar
Source code (ST)	Source text (based on ASCII)	Detailed comparison
	• Execution-relevant source code Everything apart from comments and formats such as spaces, empty lines	No detailed comparison
	• Comments and formats	No detailed comparison

Comparison attributes of a source	**Aspects/Sub-aspects**	**Detailed comparison / scalar**
Source code (MCC/LAD/FBD)	Source text (based on ASCII)	No detailed comparison
	• INTERFACE (exported declaration) Declarations and connections (the detailed comparison shows the first interface difference)	Detailed comparison unit [1]
	• IMPLEMENTATION (source-internal declaration) Declarations and connections (the detailed comparison shows the first difference in implementation)	Detailed comparison unit [1]
	• Comment language	Scalar
	• Comments Only displayed if the comment language has the same value (the detailed comparison shows the first comment difference)	Detailed comparison unit [1]
	• POU (n times, n>= 0)	No detailed comparison
	Declarations	Detailed comparison
	Execution-relevant source code	Detailed comparison
	Formats Only displayed if the comment language is not the same	
	Comments and formats Only displayed if the comment language is the same Examples of formatting: spaces, empty lines, network or block numbers	Detailed comparison
Properties [2]	• Compiler options Corresponds to: Properties tab, Compiler and Additional settings	Detailed comparison for local options incl. processor settings
	• Object address	Scalar
Effective compiler options	Corresponds to: Properties tab, Compilation The "effective" compiler options are developed from the combination of "global" and "local" computer settings. Any transfer of the global compiler options is impossible as these options are not saved at the source. In order to achieve an identical compilation result, the "effective" compiler option can be transferred to the source properties as a "local" compiler option via the setting "Local Opt(A) <- Effective Opt. (B)" in the project comparison. In this case, click this option in the object comparison. This setting cannot be transferred for libraries.	Detailed comparison

Comparison attributes of a source	**Aspects/Sub-aspects**	**Detailed comparison / scalar**
Compilation result	The compilation result for a source includes:	
	• The source text of relevance to execution, i.e. everything apart from comments and formats	No detailed comparison
	• The compiler settings (local and global)	
	• The compiler version (e.g. compilation with new SCOUT)	
	• External references – Interfaces of other sources used – Interfaces of libraries used – Global device variables used – I/O variables used – TOs used – Tasks used in the task control commands	No detailed comparison
	• Message configuration (used symbols/IDs) If a different number has been assigned to an alarmS configuration icon, and this icon has been used in this program, then the source code is the same in both cases but the compilation result is different. If you need to work out online which number has been used for an icon, you upload a device into an empty project. Here, you can look in the message configuration (the icons and the assigned numbers are entered in this during the upload).	Detailed comparison
	• Data structure of retain data	No detailed comparison
	• Data structure of non-retain data	No detailed comparison
	• Exported interface (effective) Exported data and data types used [3)] Exported POUs	No detailed comparison
HMI-relevant data	Records the structural setup (data type of variables, number of variables) of the data which an HMI system can access. If the setup changes, the HMI configuration must be updated. Data included:	No detailed comparison
	Offline comparison • VAR_GLOBAL variables of the source	For ST: Detailed comparison, Var block displayed Otherwise: Detailed comparison unit
	• The object address (see source properties)	Scalar
	Online comparison Currently, the comparison attribute does not produce a meaningful result.	

1) First difference is displayed

2) Not relevant for online consistency

3) "Data types used" means that they are used in the exported global variables or in the exported POUs. Constants plus default and initialization values are not taken into account.

DCC chart

Table 5-40 Comparison attributes of a DCC chart

Comparison attributes of a DCC chart	Aspects/Sub-aspects
Source code Time stamp comparison - for time stamp "Last modified on (STEP7)"	**Offline only** • Changes in the DCC editor based on a time stamp (Last modified on (STEP7)). The comparison only relates to a statement about an offline/offline comparison, as the time stamp is not stored online. **Note**: The comparison is based on a time-stamp comparison only. The time stamp of the most recent change is used. If a comparison is carried out on this basis, a statement on the equivalence is only possible when the time stamp is the same. If the time stamp is different, the source code may be the same or different (e.g. if the most recent change was reversed). In such cases, it is not possible to make any statements regarding this in the object comparison. This is shown by a **?**.
Blocks	Block instances and their properties, such as • Type • Name • Address • Constant values (initial values) of inputs and outputs • Assignment to execution group • Change the predecessor of the execution group by inserting or deleting execution groups • Change to the run sequence
Interconnections	The comparison attribute for interconnections indicates differences in terms of chart interconnections. This includes: • Interconnections of all the inputs within the chart • Interconnection/assignment of @ parameters to inputs and outputs • SIMOTION: Interconnections with execution groups • SIMOTION: Interconnections with operands (TOs/sources, global device variables, IO) • External references for the chart (referenced operands, DCB and DCC libraries used) **Notes**: When interconnecting a chart's output with another chart's input, the interconnection is always incorporated in the chart's comparison attribute along with the input. If you have defined a BICO @ parameter for an input with SINAMICS while making another interconnection at the same time, the interconnection will be executed as a BICO interconnection and, as such, will be incorporated in the "@ parameters and external references" attribute rather than the "interconnections" attribute. Interconnections with operands are only incorporated in the interconnections attribute with SIMOTION. With SINAMICS, all interconnections with operands are executed as BICO interconnections and, as such, contained within the "@ parameters and external references" attribute. Interconnections depend on external references. The external references include not only non-DCC objects, but also the used DCB/DCC libraries.

Comparison attributes of a DCC chart	**Aspects/Sub-aspects**
Execution groups	Chart execution groups and their properties • Name • Task assignment (only applies to SIMOTION) • Predecessor execution group • Initial state of execution group (active/inactive)
@ parameters and external references	**SIMOTION:** • Published parameters • Compiler settings • External references – Interfaces of referenced sources (programs: ST, MCC, LAD/FBD) – Reference parameters of other DCC charts – Interfaces of DCC libraries used – Global device variables used – I/O variables used – TO system variables used **SINAMICS**: • Parameter values of the @ parameters (p parameters: Values of the adjustable parameters and BICO interconnections on BICO sinks) • Adjustable parameter for DCC (p21000[0 to 9])
HMI-relevant data	Records the structural setup (data type of variables, number of variables) of the data which an HMI system can access. If the setup changes, the HMI configuration must be updated. Data included: **SIMOTION**: **Offline/offline comparison**: • Data that can be addressed by the HMI • The object address (see chart properties) **Offline/online comparison <= device version V4.1.2:** No aspects **Offline/online comparison with device versions as of V4.2:** • Data that can be addressed by the HMI • The object address (see chart properties) **SINAMICS:** • Not relevant

Note

No detailed comparison is available for the comparison features of the DCC charts!

Note

If a published connection is changed in offline mode, the Online/Offline Comparison window is not opened when the online mode is opened.

The difference between the target device and the project in the DCC chart can only be seen at **Target system output** and in the project navigator.

If a connection that has not been published is changed in offline mode, the **Online/Offline Comparison** window is opened when the online mode is opened.

Library

Table 5-41 Comparison attributes of a library

Comparison attributes of a library	Aspects/Sub-aspects	Detail comparison/Scalar
Properties	• Compiler options Corresponds to: Properties tab, Compiler	Detail comparison for local options
	• Used technology packages <device type> See compilation result, dynamic aspect	Detail comparison, used technology packages
Effective compiler options	Corresponds to: Properties tab, Compilation	Detail comparison
Compilation result <Device type> Compilation result	Depending on the setting in the properties dialog (technology packages), the compilation result is shown for each device type or as unrelated to the device type. The following data is included in the comparison: • The source code of relevance to execution, i.e. everything apart from comments and formats • The compiler settings • The compiler version (e.g. compilation with new SCOUT) • External references (sources, libraries, TPs, TOs, I/Os, global device variables, tasks) • As with the compilation result <Device type> Compilation result unrelated to device type	No detail comparison

Source in a library

Table 5-42 Comparison attributes of a source in a library

Comparison attributes of a source in a library	Aspects/Sub-aspects	Detail comparison/ Scalar
Source code (ST)	Source text (based on ASCII)	Detail comparison
	• Execution-relevant source code Everything apart from comments and formats such as spaces, empty lines	No detail comparison
	• Comments and formats	No detail comparison
Source code (MCC/LAD/FBD)	Source text (based on ASCII)	No detail comparison
	• INTERFACE (exported declaration) Declarations and connections (the detail comparison shows the first interface difference)	Detail comparison unit [1]
	• IMPLEMENTATION (source-internal declaration) Declarations and connections (the detail comparison shows the first difference in implementation)	Detail comparison unit [1]
	• Comment language	Scalar
	• Comments Only displayed if the comment language has the same value (the detail comparison shows the first comment difference)	Detail comparison unit [1]
	• POU (n times, n>= 0)	No detail comparison
	Declarations	Detail comparison
	Execution-relevant source code	Detail comparison
	Formats Only displayed if the comment language is not the same	
	Comments and formats Only displayed if the comment language is the same Examples of formatting: spaces, empty lines, network or block numbers	Detail comparison
Properties [2]	• Compiler options Corresponds to: Properties tab, Compiler and Additional settings	Detail comparison for local options incl. processor settings
	• Object address	Scalar

DCC chart in a library

Table 5-43 Comparison attributes of a source in a DCC library

Comparison attributes of a DCC library	Aspects/Sub-aspects
Source code (time stamp comparison)	**Offline only** • Changes in the DCC editor based on a time stamp (most recent change). If values are changed in online mode, the changes and, therefore, the time stamp are detected offline. **Note**: The comparison is based on a time stamp comparison only. The time stamp of the most recent change is used. If a comparison is carried out on this basis, no statement can be made concerning whether the source code is the same unless the time stamp is the same. If the time stamp is different, the source code may be the same or different (e.g. if the most recent change was reversed). In such cases, it is not possible to make any statements regarding this in the object comparison. This is shown by a **?**.
Blocks	Block instances and their properties, such as • Type • Name • Address • Constant values (initial values) of inputs and outputs • Assignment to runtime group • Predecessor in runtime group
Interconnections	The following interconnections are included in the comparison: • Chart interconnections • Interconnection of all the chart's inputs • Interconnections with runtime groups • Interconnections with @ parameters • Interconnections with non-DCC objects within SIMOTION (TOs/sources, global device variables, I/Os)
Runtime groups	Chart runtime groups and their properties • Name • Task assignment • Predecessor execution group • Initial state of runtime group (active/inactive)

Note

No detail comparison is available for the comparison features of the DCC charts in a library!

5.3.5.2 SINAMICS objects

Drive unit

Table 5-44 Comparison attributes of a drive unit

Comparison attributes of a drive unit	Aspects/Sub-aspects	Detail comparison/Scalar
Reference topology	• The reference topology With this comparison attribute, the serial number is not taken into account in the comparison.	No detail comparison (see note below)
Topology (with serial number)	• Reference topology With this comparison attribute, the serial number is taken into account in the comparison.	Detail comparison, reference topology serial number

Note

The "Reference topology (with serial number)" comparison attribute acquires the same data as the "Reference topology" comparison attribute. The serial numbers are also acquired. In detailed comparison to the "Reference topology (with serial number)", only the differences to the serial numbers are displayed. No differences are displayed in the detailed comparison for differences in the reference topology, although the serial numbers are the same.

If differences in the reference topology emerge during an online/offline comparison, you can use the topology editor to determine the differences. Please ensure the reference topology is selected for both comparison partners.

Drive object (DO)

Table 5-45 Comparison attributes of a DO

Comparison attributes of a drive object (DO)	Aspects/Sub-aspects	Detail comparison/Scalar
Function modules	• Function modules (see Parameter r108 in the DO expert list or the properties dialog box of the DO)	Detail comparison, expert list
Structure parameters	• Parameters that determine the structure, e.g. r51 (drive data set DDS effective). These parameters are indexed, i.e. they form enums and bit arrays (arrays of a different dimension), in order to offer several selection options/settings or display values for a parameter. These parameters are identified in the expert list by a "+".	Detail comparison, expert list
Download parameters	• Download parameters (see DO Expert list). These are the parameters whose values will be downloaded to the device.	Detail comparison, expert list
Units	• Units Values: Name of the unit system, e.g. "SI unit system" This comparison feature does not determine the structure.	Scalar

DCC chart

Description of comparison attributes: See DCC chart (Page 3459) in the chapter titled "SIMOTION objects"

5.3.5.3 MICROMASTER and SINAMICS G120 objects

Drive unit

There are no comparison attributes for these drive units.

Drive object (DO)

Table 5-46 Drive object comparison attributes (DO)

Comparison attributes	Aspects/Sub-aspects	Detail comparison/ Scalar
Device version	• Power unit/infeed (configuration vector specifies the device version of a device type) (Only available with G120, not MM4 or Combimaster)	No detail comparison (see note below)
	•	
Download parameters	• Download parameters (see Comparing experts lists)	Detail comparison, expert list
Units	**Device version as of V4.4:** Values: Name of the unit system, e.g. "SI unit system"	Scalar

Note

If you use the settings on the Workbench tab to activate the option **Display Online/offline comparison for the connection**, a comparison will be displayed when you go online. This is not the same as a project comparison.

5.3.5.4 Display warnings

The "Display Warnings" dialog box opens when the "Warning(s)" selection box is activated.

You require a comparison project for the online comparison. This is generated based on the open project. The upload of the device to be compared is performed in this shadow project. This does

not alter the open project. Restoration of the offline data through an upload is not always complete, but this is not relevant for the comparison result.

- Missing supplementary data means that a detailed comparison or a data transfer is not possible.
- Changed know-how protection data means that a detailed comparison may not be possible because you cannot log on.

5.3.6 Appendix A

5.3.6.1 Frequently Asked Questions (FAQs)

Note

As a general rule, the project comparison can only be performed based on saved data. For this reason, data is automatically saved prior to performing the comparison.

We recommend compiling the project prior to starting the project comparison.

Differences are detected in the project comparison, yet the detail comparison does not display any differences.

Possible causes:

- The project compilation is not up-to-date.
- The project comparison is no longer up-to-date, changes have been made to the project following the start of the project comparison (such as transfers in the detail comparison) which directly affect the detail comparison.
- Parts of the project were created with a SCOUT version earlier than 4.0 and therefore can no longer be changed (I/O variables). In this case, the hash codes required for the project comparison have not yet been calculated correctly.
- In the online-offline comparison, the execution system is displayed as different.
  This can occur for a comparison with SCOUT as of V4.4 and a Runtime version up to V4.4. Namespaces have also been introduced for the execution system with SIMOTION SCOUT V4.4. A difference is detected because of the changed data format of the execution system (different hash codes). However, the detailed comparison does not show any differences because this is based on the actual data.

In the online/offline comparison, a question mark is always displayed next to the HW Config system interface.

For the online/offline comparison, only SIMOTION data is downloaded to the offline PG project, in the same way as uploading to the PG. As a result, only this data can be used for the comparison. This is the reason why no statements can be made with regard to differences/ similarities.

The DCC source code time-stamp comparison displays a question mark.

In the case of DCCs, comparison results can only be calculated correctly if the chart compilations are up-to-date.

This applies to both the online/offline comparison (if necessary, update the offline project compilation) and the offline/offline comparison (both project compilations must be up-to-date here).

Following transfer to the online/offline comparison, the object in question is still inconsistently online.

Possible causes:

Following the transfer, the project must be saved to ensure that it is consistently online.

When transferring individual programs, global settings such as compiler settings are not applied. The compilation result is therefore not always consistent. In this case, consistency can only be achieved by means of a download.

The configuration data of an axis has been changed by the system, yet the project comparison still does not display any differences.

Possible causes:

The project comparison compares only configured values (offline and online). Modifications to configuration data initially only change current data, i.e. configured data remains unchanged. To be able to compare the adapted data with the project comparison, you must firstly perform **Copy current data to RAM**.

The online/offline comparison displays differences, yet there are no buttons available for the detail comparison.

Possible causes:

It is likely that no supplementary data is available in the target system.

Program differences are highlighted in the online/offline comparison, but the source text folder can no longer be opened in the overview.

- If no supplementary data has been downloaded to the target system, statements cannot be made regarding the program details. The folder can therefore not be opened
- If the programs in the target system are know-how protected, details can only be displayed once the password has been entered.

There are program differences in the online/offline comparison, so why is there no data transfer button?

Possible causes:

- If there are no differences in source code, then there is no transfer button.
- If there is no supplementary data, then no transfer button is available.

In the detail comparison (online/offline comparison) for the execution system, no values are displayed for the cycles, only a question mark is displayed.

The cycle settings, which are displayed in the detail comparison of the execution system, are only partially based on SCOUT settings. Under the basic cycle clock, for example, the settings made in the HW Config are displayed. Due to the fact that HW Config cannot be used in the online

comparison, it is not possible to upload HW Config via SCOUT and, as a result, no statements can be made regarding differences/similarities.

Unknown values are displayed in the detail comparison (online-offline comparison) for the execution system

This can be TCMRM_MeasureSyncFuncs and TCMRM_BlockSyncFuncs, for example. These values show the internal representation of the configuration.

The displayed differences describe the existing differences in the configuration. The following applies:

- TCMRM_MeasureSyncFuncs = leave time monitoring active
- TCMRM_BlockSyncFuncs = discontinue time monitoring and interrupt the task

At the time of configuration, these values are indirectly set via the drop-down list box in the execution system of the IPO/ServoSynchronousTasks at "Monitoring during the execution of synchronous functions".

In the programs, a difference is displayed in the source text, yet the comparison of all POUs does not show any differences.

The project comparison enables fast comparison of online and offline projects based on hash calculations. In the case of detail comparisons, the original data is then applied.

Occasionally (such as following an upload), this may result in the calculated hash values no longer corresponding.

Save and compile the project and then perform a download.

The comparison tool displays numerous question marks.

To correctly display the comparison values, the projects to be compared must both be converted to SCOUT version 4.2 and preferably also compiled using this version. If necessary, also open the comparison project and re-compile it.

Objects are recreated during the import.

The project that has been exported and then imported again is subsequently no longer the same as the original project because the objects are recreated during the import. For example, the FastIO displays differences in the comparison.

Online-offline comparison refers to the device

The online-offline comparison always only refers to one device Therefore, cross-device TO interconnections are always displayed as non-existent online.

Converting DCC libraries

As of Version 4.3 it is possible to easily convert DCC libraries from SIMOTION libraries to SINAMICS libraries and vice versa. Complex conversion mechanisms are used in both directions. Therefore, a library that is copied back to the original location in this way is not identical to the original library.

5.4 Updating SIMOTION Devices

Preface

Content

This document is part of the **SIMOTION Service and Diagnostics** documentation package.

Scope of validity

This document is valid for SIMOTION SCOUT product version V5.4:

- SIMOTION SCOUT V5.4 (engineering system of the SIMOTION product family)
- SIMOTION Kernel V5.4, V5.3, V5.2, V5.1, V4.5, V4.4, V4.3, V4.2, V4.1 SP2
- SIMOTION technology packages CAM, PATH, Cam_ext and TControl in the relevant version for the kernel.

Sections in this documentation

This documentation describes a simple way of exchanging the configuration or firmware of one or more SIMOTION devices. This process is referred to as upgrading.

- Overview
  General overview of upgrade data and media, of upgrade scenarios, and of the people involved.
- Upgrading
  Description of the individual upgrading steps.
- Downgrading
  Information about downgrading

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT

- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

5.4.1 Fundamental safety instructions

5.4.1.1 General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

5.4.1.2 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

5.4.1.3 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

5.4.1.4 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

5.4.1.5 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

5.4.1.6 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

5.4.1.7 Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

5.4.2 Overview

SIMOTION offers a convenient solution for upgrading SIMOTION devices or SIMOTION projects for machine manufacturers and machine operators.

Upgrading does not simply refer to an update to a higher version of firmware; rather, in general terms it refers to switching to a defined configuration, e.g. a project update. It is also possible to return (downgrade) to a previous configuration.

Upgrade or downgrade procedures can easily be performed on SIMOTION devices locally or remotely. The data can be imported to a SIMOTION device via a portable, easy-to-use storage medium or a communication connection.

Options and advantages

An extensive range of options is available for upgrading SIMOTION devices. These include, for example:

- The user-friendly creation of upgrade data is via SIMOTION SCOUT / SIMOTION SCOUT TIA with the aid of the Device Update Tool (at the machine manufacturer's site).
- Time- and location-independent upgrade of the SIMOTION devices (without SIMOTION SCOUT for the machine operator).
- The upgrade data can be sent from the machine manufacturer to the machine operator via e-mail or post.
- There is no need to use license keys, as licenses are either retained or upgraded at the same time.
- Retain data is retained.
- An update which has been imported can be discarded again, and the previous configuration downgraded.
- You can upgrade either a single SIMOTION device or multiple devices of one or more SIMOTION projects.
- You can limit the upgrade to configuration subsets, e.g. only project data.

Single or multiple upgrades

Various upgrade scenarios are possible: When performing a single upgrade, only one SIMOTION device is upgraded, whereas one or more SIMOTION devices from one or more projects can be upgraded when performing a multiple upgrade.

Upgrade data

Upgrade data is the data created via SIMOTION SCOUT based on one or more SIMOTION projects. The upgrade data contains all the information required for upgrading or downgrading the data on a SIMOTION device. This includes:

- SIMOTION project
- Technology packages

- User data (unit data, SIMOTION IT data)
- Firmware

Handling upgrade data

The upgrade data is created by the machine manufacturer's application developer via SIMOTION SCOUT. The upgrade data can then be handled flexibly depending on the SIMOTION device (SIMOTION C, D, or P) and the customer requirements:

- Creating upgrade data and then copying it to a storage or upgrade medium:
  - CF / MMC card
  - USB memory stick
  - SIMOTION IT file or
  - Creating an upgrade archive
- Alternatively, the upgrade data can be created and stored in an archive on the computer, so that it can be imported to an upgrade medium that is compatible with the SIMOTION device later.
- The data can be transferred to an upgrade medium at either the machine manufacturer's site or the machine operator's site. For this purpose, the machine manufacturer transfers the upgrade archive to the user's at the machine operator's site.
- The user is prompted to import the upgrade data into the SIMOTION device(s), without any involvement by the application developer, thereby upgrading the SIMOTION device(s). SIMOTION SCOUT does not need to be available on site.

5.4.2.1 Data flow and handling update data

The machine manufacturer's application developer creates the update data using the SIMOTION SCOUT engineering system. He or she writes the SIMOTION applications and puts them into operation.

The user at the machine operator's site carries out service work on the machine. The user receives the update data from the application developer and transfers them to the machine. There is no need for SIMOTION SCOUT to be available on-site in order to transfer the update data to the SIMOTION device or to update a SIMOTION project.

The user at the machine operator's site can import the previously created update data, which are located on an update medium, to the SIMOTION device.

The figure below provides an overview of how the update data are handled. Update media include CF/MMC card, USB memory stick and SIMOTION IT file.

Alternatively, the user has the option of creating and storing update data in an update archive, and then creating the update medium at a later point. This can be carried out by either the application developer at the machine manufacturer's site or by the user at the machine operator's site.

The update data can be imported to an update medium immediately after they have been created by the application developer. The update media CF / MMC card or USB memory stick can be sent by post, or electronically by SIMOTION IT file and update archive to the machine operator.

The update data are generally transferred to the SIMOTION devices by the user at the machine operator's site.

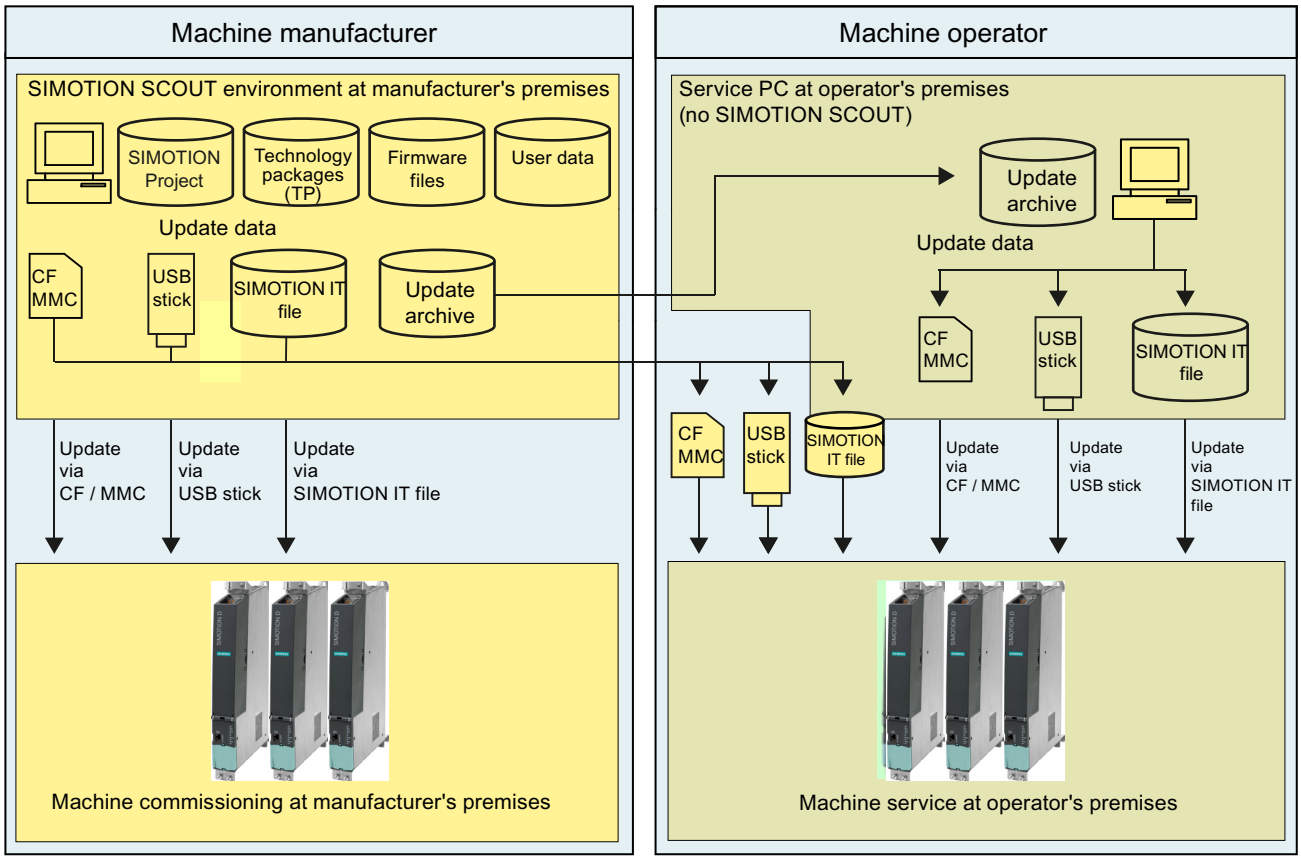


Figure 5-83 Data flow and handling of the update data at the machine manufacturer's and machine operator's sites

5.4.2.2 Update data, update archive, and update media

Update data is required to update a SIMOTION device or SIMOTION project. The newly created update data can be stored in an update archive or saved onto an update medium immediately.

The advantage of the update archive or the update media is that the update data created at the machine manufacturer's site can be used by the machine operator anywhere and at any time. SIMOTION SCOUT is not required in order to transfer the update data on-site.

Update data

The configuration is determined on the basis of the composition of the update data.

Update data is all the data required for an update procedure. The update data is created using SIMOTION SCOUT.

The update data contains the following data:

- Firmware (FW):
  The firmware comprises all the firmware components that are required to operate the respective SIMOTION device (e.g. drive FW, CBE FW, TM FW, BIOS, etc.)
- Project data:
  The project data contains all the data required to operate the respective SIMOTION device. SIMOTION D also contains the drive configuration for the SINAMICS Integrated and the CX32/ CX32-2.
- Technology packages (TP)
- User data (e.g. HTML pages, SSL key, SIMOTION project for storing data on memory cards, SIMOTION IT data, JAVA programs and unit data sets)
- SCOUT project archive (for data storage on the memory card)

It is assumed, in principle, that all of the update data components referred to above are to be updated. However, only individual update data components can be updated, e.g. only the technology packages. This is at the discretion of the user and is defined when the update data is created.

Note

For SIMOTION P, there are currently only plans to update the project data, technology packages and user data.
The firmware **cannot** be updated because of its dependence on other Windows components.

Update archive

The update data created using SIMOTION SCOUT is stored in an update archive if it is not imported to an update medium.
The update archive is a directory tree in which all of the update data is saved.

Update media

The term "update media" is used to refer to storage media on which the update data is saved before being imported to the SIMOTION devices to be updated.
Update media used for SIMOTION are USB memory sticks, CF / MMC cards and SIMOTION IT files.

The choice of update media depends on the SIMOTION devices to be updated.

For updating an individual device in the case of SIMOTION C or SIMOTION D, a CF/MMC card can be loaded or a SIMOTION IT file can be generated (also for several devices).

SIMOTION D4x5-2 also supports the option of using a USB memory stick as the update medium. The USB memory stick can be used to update one or more SIMOTION devices.

In the case of SIMOTION P, you can select either a USB memory stick or a SIMOTION IT file as an update medium for either an individual device or for several devices.

To find an overview of which update medium is suitable for which SIMOTION device, please see Device-specific update media (Page 3487).

5.4.2.3 Update data behavior within a SIMOTION device

Each SIMOTION device has a memory card containing the most important data for the device, such as firmware, technology packages, project data, and backed up retain data.

In the case of SIMOTION D this is the CF card (Compact Flash Card), and with SIMOTION C the MMC (Micro Memory Card).

With SIMOTION P, it depends on the hardware version of the SIMOTION P320-4 and is either the CFast card or the SSD (Solid State Disk); both variants function as an operational data carrier and are not intended for use as update media.

If a SIMOTION device is updated, the data on the memory card in the SIMOTION device behaves as follows:

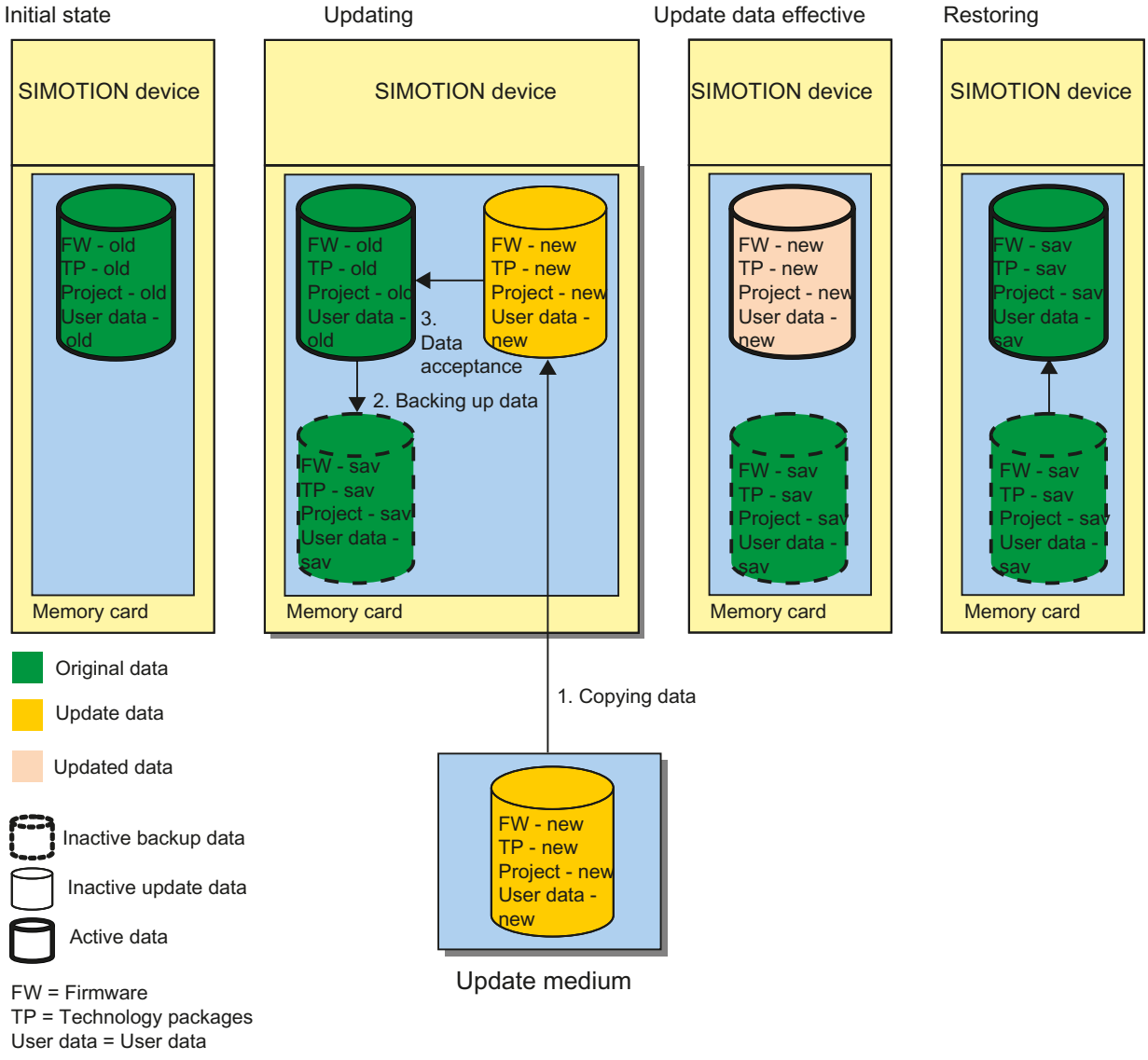


Figure 5-84 Update data behavior within a SIMOTION device

1. The update data is transferred from the update medium to the memory card in the SIMOTION device. (USB memory stick, SIMOTION IT)
   With the CF / MMC card, the equivalent mechanism is a CF / MMC adapter.
2. The existing data is then saved on the memory card.
3. The update data takes effect after restarting the device.
4. The memory card now contains both the update data and, concurrently, the saved data from the previous configuration.
5. In the event of an error, or if the new application does not meet your requirements, you can revert back to the saved data.
   This data is activated when you perform a restore function, overwriting the update data.

Note

Note the following when backing up restore data in the SIMOTION device

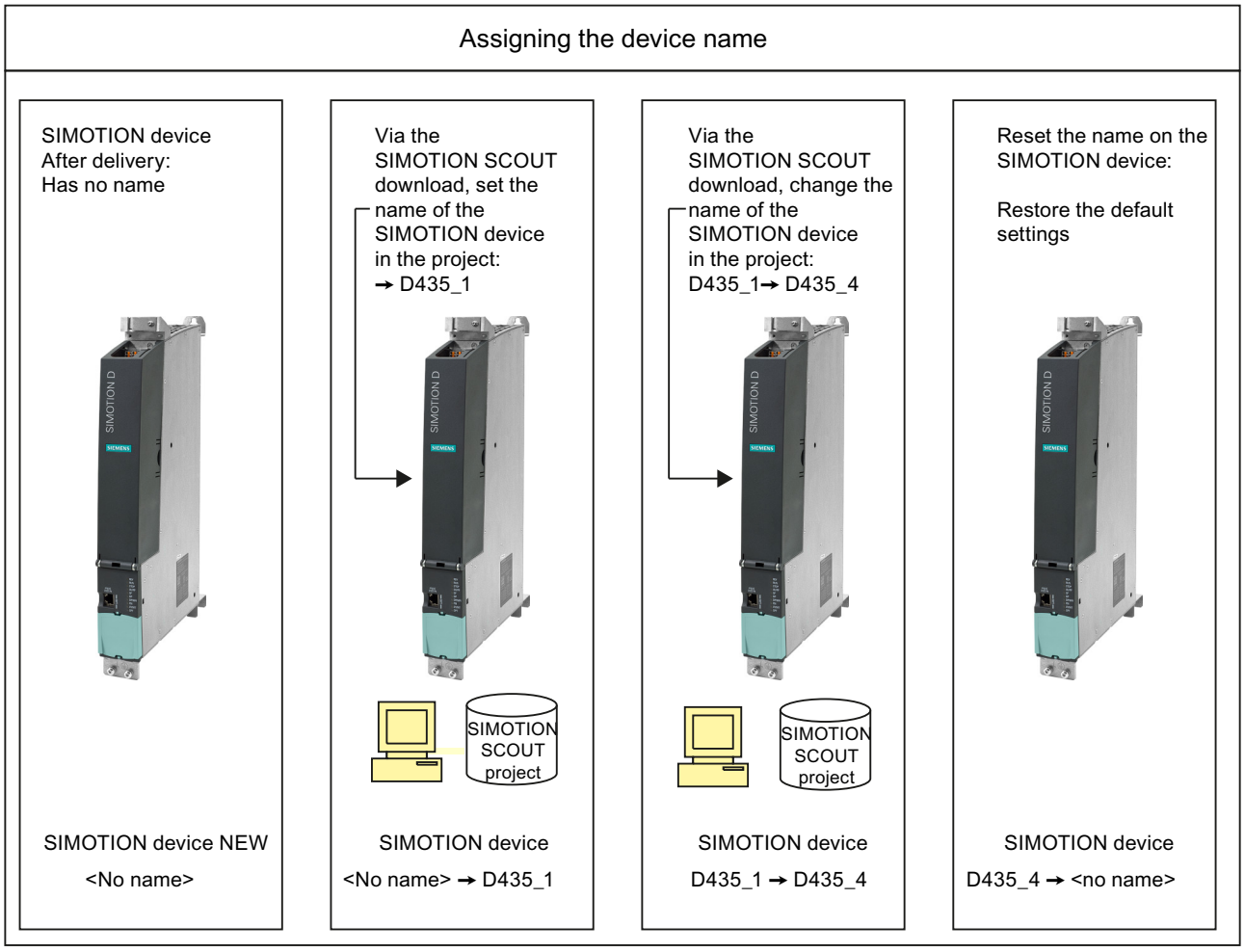
The current configuration data is saved as restore data when updating. The SIMOTION device has only one backup.

If the same update data is imported twice by mistake, the backed up restore data will also be overwritten.

In order to back up data over a longer period of time, the update archive can be backed up separately (onto a DVD, for example).

5.4.2.4 Assigning update data to devices

The device name which was saved in the project is contained in the update data. This name is used for assigning the update data to the relevant device. The name is saved persistently on the device. Since the name is saved persistently within the SIMOTION device, a corresponding device assignment takes place during the update.



5.4.2.5 Single and multiple updates

Upgrade data can be created for a single SIMOTION device or multiple SIMOTION devices relating to one or more SIMOTION projects.

Note

Creating upgrade data in SIMOTION SCOUT TIA

Note that you can create upgrade data in SIMOTION SCOUT TIA only for the currently opened project.

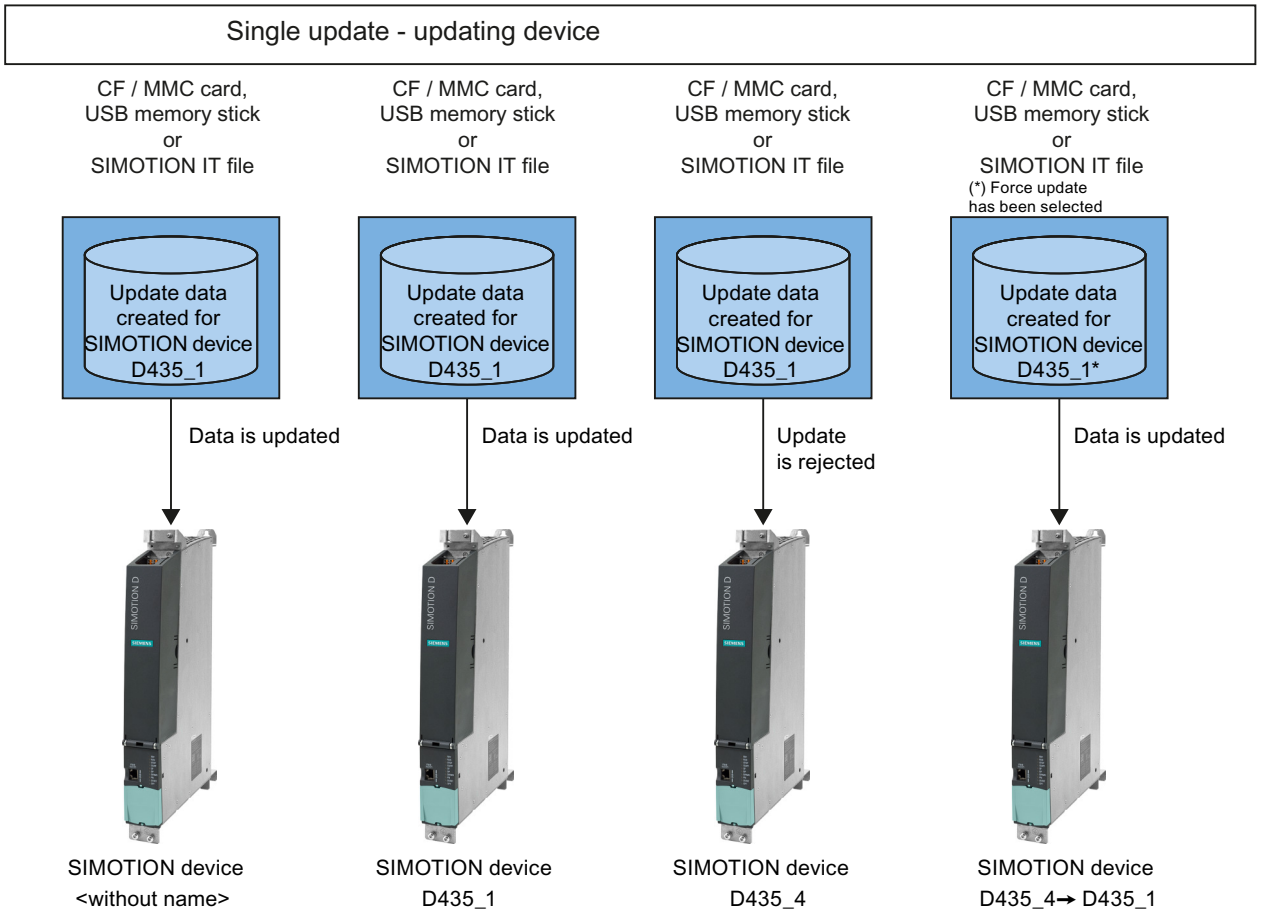
During **multiple upgrades**, i.e. when upgrading at least two SIMOTION devices, it is imperative that device names are assigned. If the device names in the upgrade datado not match the persistently saved names of the devices to be upgraded, the upgrade cannot take place.

An exception to this is the **single upgrade** of a device that has not yet been assigned a persistently saved name. In this case, the device receives its name from the upgrade data.

In a **single upgrade** scenario, it is also possible to force an upgrade if the device names do not match.

Single upgrade

Upgrade data for a single SIMOTION device is created during a single upgrade. This scenario is recommended in the case of large projects, for example, in which only one SIMOTION device from a project is to be upgraded.



The following scenarios are depicted in the figure:

- The SIMOTION device to be upgraded does not have a name.
  The upgrade is performed and the SIMOTION device to be upgraded is assigned the SIMOTION device name D435_1 from the upgrade data.
- The SIMOTION device to be upgraded and the SIMOTION device from which the upgrade data has been created have the same name. The data is upgraded without any prompt.

- The SIMOTION device to be upgraded and the SIMOTION device from which the upgrade data originates do not have the same name.
  It is assumed that they are two different SIMOTION devices, and that the data is not upgraded.
- If there is an explicit request for the data from SIMOTION device D435_1 to be applied by SIMOTION device D435_4, the **Force update** option, i.e. force upgrade can be selected. This means that the SIMOTION device is assigned the complete set of new data, including the device name.
  The **Force upgrade** option is available only when upgrading a single SIMOTION device.
  Further information can be found at Assigning upgrade data to devices (Page 3484)

Multiple upgrade

When multiple upgrades are concerned, the upgrade data is created for at least two or all SIMOTION devices of one or more SIMOTION projects.

Note

Multiple upgrade for SIMOTION SCOUT TIA

Note that you can create upgrade data in SIMOTION SCOUT TIA for just one SIMOTION project.

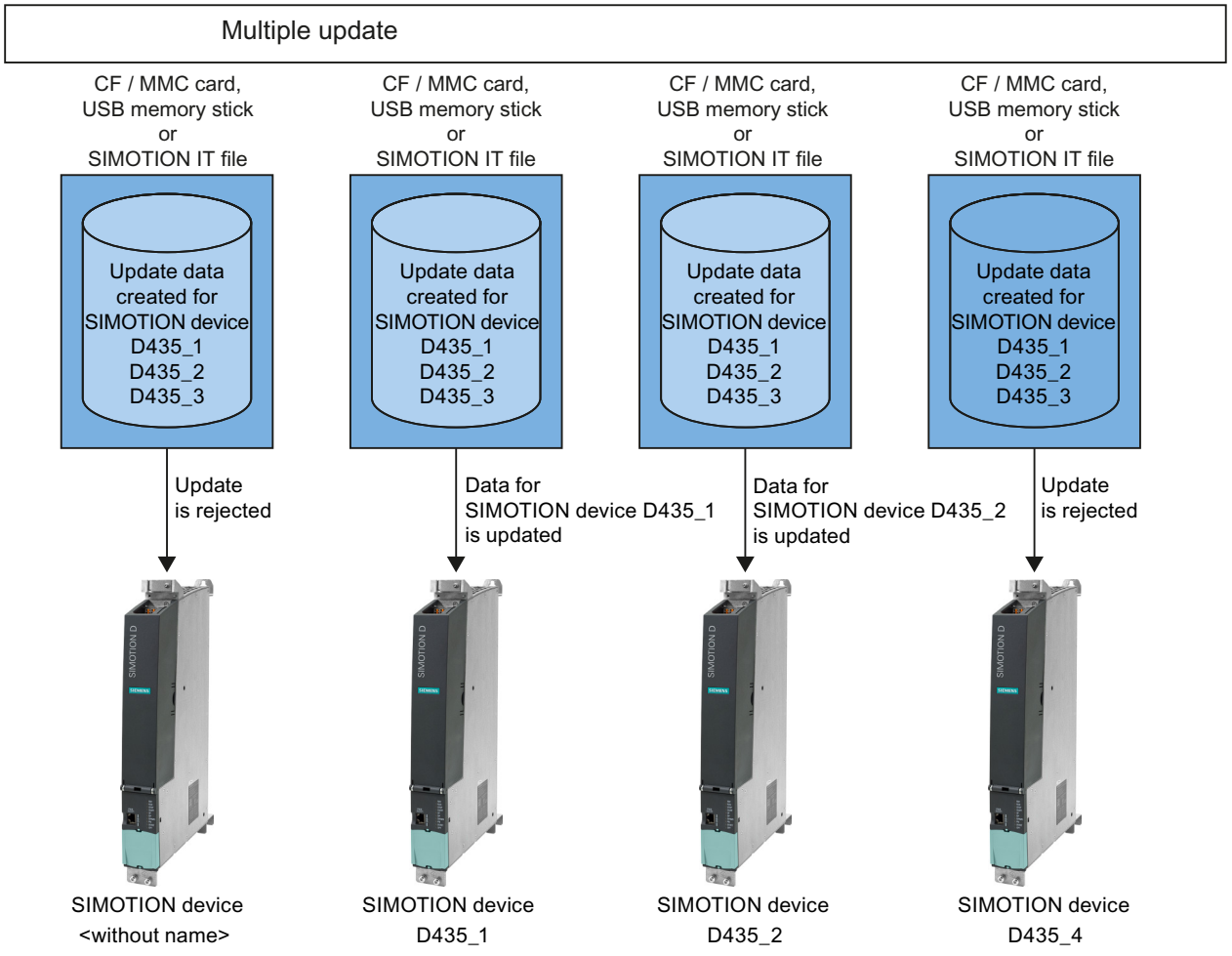
Either components of the upgrade data of a project or the entire project can be upgraded. This is at the discretion of the user.

However, it is imperative that the devices from the upgrade data and the devices to be upgraded have the same names. Upgrade data from two different SIMOTION devices with the same name (e.g. from different projects) may not be created at the same time because the data cannot be assigned.

Further information can be found at Assigning upgrade data to devices (Page 3484)

The graphic below shows the pool of upgrade data that has been created for the SIMOTION D435_1, D435_2, and D435_3 devices, and which is still located on an upgrade medium.

When multiple upgrades are concerned, the device names from the upgrade data must match the persistently saved device names from the SIMOTION devices to be upgraded to guarantee that the upgrade data is assigned correctly. If this is not the case, the upgrade **cannot** be performed.



5.4.2.6 Device-specific update media

The table below provides an overview of which update media can be used for which SIMOTION devices, and in which update scenarios:

Table 5-47 Overview of the update scenarios in combination with SIMOTION devices and update media

SIMOTION device	USB memory stick	SIMOTION IT	CF / MMC card
D4x5-2	Single/multiple update	Single/multiple update	Single update
D410-2	-	Single/multiple update	Single update
P320-4 E / P320-4 S	Single/multiple update	Single/multiple update	-

Note

Requirement for updating or restoring:

- SIMOTION SCOUT V4.1 SP2 or higher
- Firmware on the SIMOTION device to be updated, V4.1 SP2 or higher
- Firmware contained in the update data V4.1 SP2 or higher if firmware is selected.

Note

Upgrading using a USB memory stick

The SIMOTION device is started from the USB memory stick when updating devices via a USB memory stick. For this reason, a bootable USB memory stick must be used.

We recommend SIMATIC memory sticks. Due to the rapid developments within the market for USB memory sticks, it is not possible to make any additional recommendations.

Information on this is available on the Internet at:

https://support.industry.siemens.com/cs/ww/de/view/32580863 (https://support.industry.siemens.com/cs/ww/en/view/32580863)

Note

Because of the size of the memory card, the update or restore function is not possible for the SIMOTION C230-2.

With the SIMOTION C240, a 64 MB MMC must be used for the update or restore function because a 32 MB MMC (supplied with V4.0 HF2) does not have sufficient memory space.

For SIMOTION C240 / C240 PN >= V5.2, the function is not possible because of the available memory space on the MMC.

Information on this is available on the Internet at:

http://support.automation.siemens.com/WW/view/en/42349674

5.4.2.7 Transferring device-specific user data to the update data

Directory structure of the user data

The user data must be stored in a defined directory structure, as shown in the figure below.

The directory structure is created using a new folder in Windows Explorer. The structure must be accessible via Windows Explorer. The data can be saved on the hard disk, a USB memory stick, or in a network folder, for example.

The ID of the user-specific folder is defined by the user.

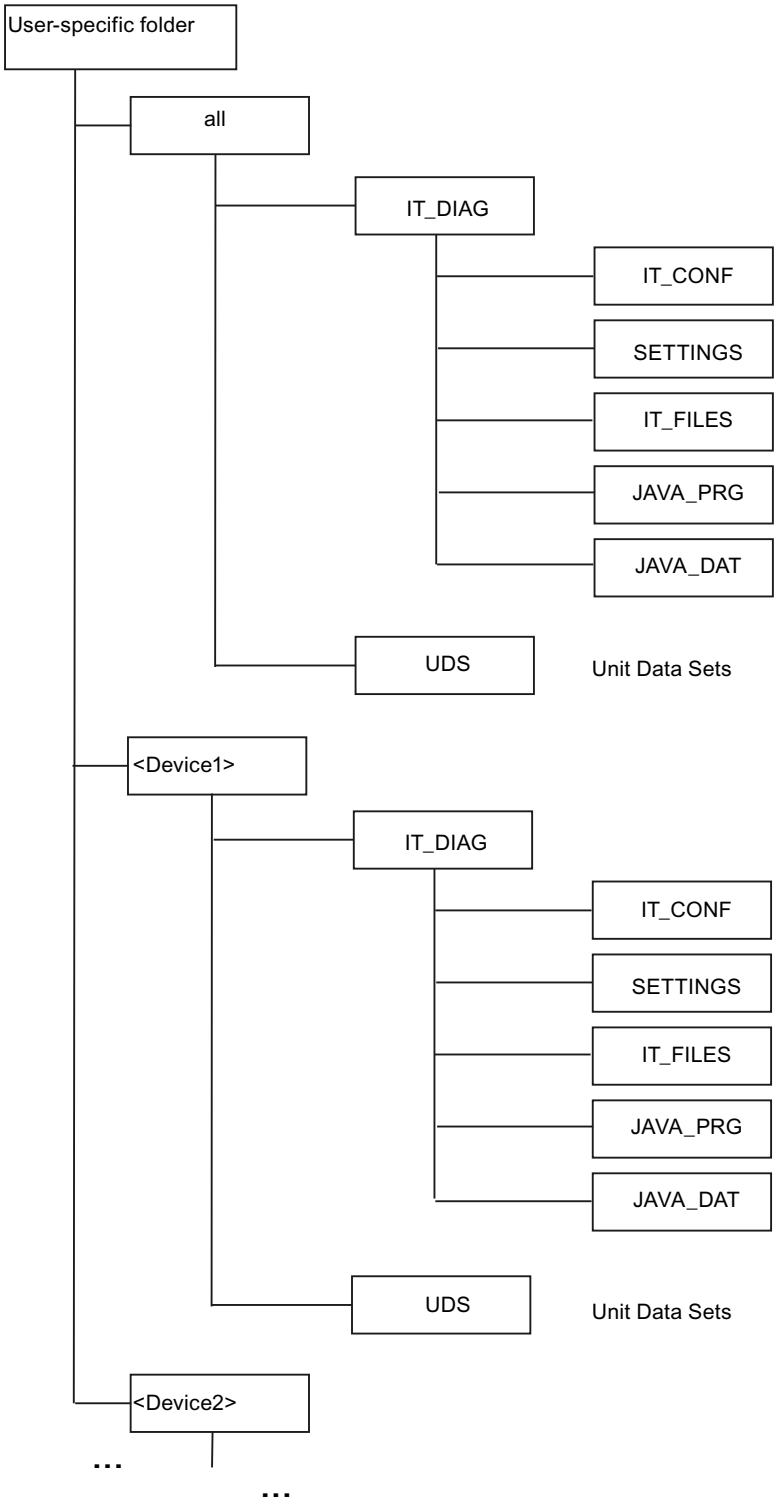


Figure 5-85 Directory structure for saving user data

The entire directory **UPP** \ **UNITDS** with the unit data sets it contains must be stored in the UDS folder.

The path to a user-specific file must be specified when importing user data via the Device Update Tool.
See also: Creating upgrade data with the Device Update Tool (Page 3492), dialog **Select additional options**.

The user data for all the devices from the projects are located in the user-specific folder. Data saved in the subdirectory **all** is imported to all the devices to be upgraded.

Data saved in device-specific folders, e.g. <**Device1**>, is only imported to the SIMOTION device whose name matches the folder name.

In the event of a conflict, if data is available for a device from both the **all** subdirectory and a device-specific folder (e.g. <**Device1**>), the user data from the device-specific folder is used.

Definition of user data

The following data is **not** part of the SIMOTION project within the context of upgrading and must, therefore, be provided by the user:

For example, at the following path:
E: \ update \ userdata \ all \ IT_DIAG

The following structure **must** be created in the IT_DIAG folder:

• **IT_DIAG**		
	IT_CONF Data for configuring the Web server (SIMOTION IT) and user rights	
		UserDataBase \ UserDataBase.XML User administration (file UserDataBase.XML in folder UserDataBase)
		WebCfg.XML Configuration data The data can be found on the CF card at: \USER\SIMOTION \HMICFG\ For further information, see: SIMOTION IT Diagnostics and Configuration Diagnostics Manual
	SETTINGS Settings for watch tables and traces from SIMOTION IT	
	IT_FILES User websites	
	JAVA_PRG Java programs (*.class, *.jar)	
	JAVA_DAT Files which are used from Java	

- **Unit data sets (UDS)**
  In the unit data sets, values from unit variables (e.g. machine data from commissioning) can be saved during runtime in data sets on the memory card using _save / _exportUnitDataSet. These are uniquely identified by the unit name and data set number (refer also to the Basic Functions Function Manual, section "Backing up data from the user program").
  In order to be able to use the data sets in the user data when upgrading, they must first be transferred via a Card-Reader or, for example, FTP, from the memory card to the PC.
  The data sets are available at **\ USER \ SIMOTION \ USER_DIR \ UPP \ UNITDS ** after they have been saved to the card.
  Alternatively, the SIMOTION SCOUT function **Save variables** can be used.

Assigning user data to the SIMOTION device

Only the user knows that user data has been assigned to the SIMOTION device to be upgraded. The following points must be clarified in order to enable user upgrade data to be transferred:

- Which user data is involved here? (Content)
- Where is the user data located? (Source)
- Where should the data be imported to? (SIMOTION device)

To ensure that all information is available, the user data must be provided in a directory structure that has already been defined.

5.4.3 Updating

5.4.3.1 Overview

Updating procedure

The following section provides an overview of the procedure for updating a SIMOTION device:

- Creating update data (machine manufacturer)
  Update data is created on an operator-guided basis via the Device Update Tool at the machine manufacturer's site, using the SIMOTION SCOUT engineering system. This is conducted by the application developer as part of commissioning or service.
- Copying the update media (machine manufacturer or machine operator)
  Once the update media has been created, it can either be saved directly to an update medium or stored as an update archive.
  If data is to be copied to the update medium at a later date, the update data that has been created is saved in an update archive (File system). This can be called up at any time and imported to an update medium.
  Data are copied to an update medium without SIMOTION SCOUT: This can be performed by either the application developer at the machine manufacturer's site or by the user at the machine operator's site.
- Transferring data from the update medium to the SIMOTION device (machine operator)
  Update data can also be imported from the update medium to the SIMOTION device, irrespective of time and place, using the Device Update tool on an operator-guided basis.

Note

If the parameterization of the SINAMICS Safety Integrated functions is contained in the update data, these functions must be activated after the update. See also *SINAMICS S120 Safety Integrated Function Manual*.

Saving update data to the update media or as an update archive

The special features for saving on the update media USB memory stick, CF / MMC card or SIMOTION IT file are described according to the procedure using the Update Tool device. The creation of the update archive is also examined.

- Creating update data with SIMOTION SCOUT and transferring data to an update medium
  - Creating update data, including copying to a USB memory stick (Page 3509)
  - Creating update data, including copying to a CF / MMC card (Page 3510)
  - Creating update data including generating a SIMOTION IT file (Page 3511)
  - Creating update data as an update archive (Page 3511)
- Copying to the update media on the basis of an update archive
  - Copying data to a USB memory stick (Page 3518)
  - Copying data to a CF / MMC card (Page 3519)
  - Creating a SIMOTION IT file and copying the update data (Page 3519)

Transferring update data from the update medium to SIMOTION devices

- Updating a module with a USB memory stick (Page 3530)
- Updating a module with a CF / MMC card (Page 3533)
- Updating a module with a SIMOTION IT file (Page 3534)

5.4.3.2 Creating upgrade data via SIMOTION SCOUT / SIMOTION SCOUT TIA

The section below describes how upgrade data is created.

The upgrade data is created user-prompted using SIMOTION SCOUT or SIMOTION SCOUT TIA. The individual data items are selected step-by-step using a wizard.

Creating update data with the Device Update tool

Requirement

The following data is required in order to create upgrade data:

- Installed SIMOTION SCOUT as of V4.1 SP2 or
  SIMOTION SCOUT TIA as of V4.5
- SIMOTION project(s)
- One or more SIMOTION devices (firmware as of V4.1 SP2)
- Current SIMOTION FWSP (Firmware Support Package) is installed (applies as of > V4.1 SP2)
- User data

Note

SIMOTION SCOUT TIA

Upgrade data is created only for the currently opened project.

Upgrade data creation procedure

Upgrade data is essentially created via SIMOTION SCOUT/ SIMOTION SCOUT TIA as follows:

- Call the Device Update Tool from the menu
- Choose whether to create upgrade data or downgrade data
- Select a project
- Select a device
- Select data
- Select further options (e.g. user data)
- Enter the version details
- Display the summary
- Copy the upgrade data to an upgrade medium or save it in an archive (to be copied to an upgrade medium at a later date)

How to create upgrade data

Call the Device Update Tool via **SIMOTION SCOUT** or **SIMOTION SCOUT TIA**. You are prompted step-by-step for creating the required upgrade data that is then either saved in an upgrade archive or imported directly to an upgrade medium.

Note

Creating upgrade data in SIMOTION SCOUT

If you call up the Device Update Tool via an open project, the project is automatically selected in the dialog.

If you also wish to save an archived SIMOTION SCOUT project in the upgrade data, you must close the project before creating the upgrade data.

1. Select the **Project > Start Device Update Tool** menu directly in **SIMOTION SCOUT**.

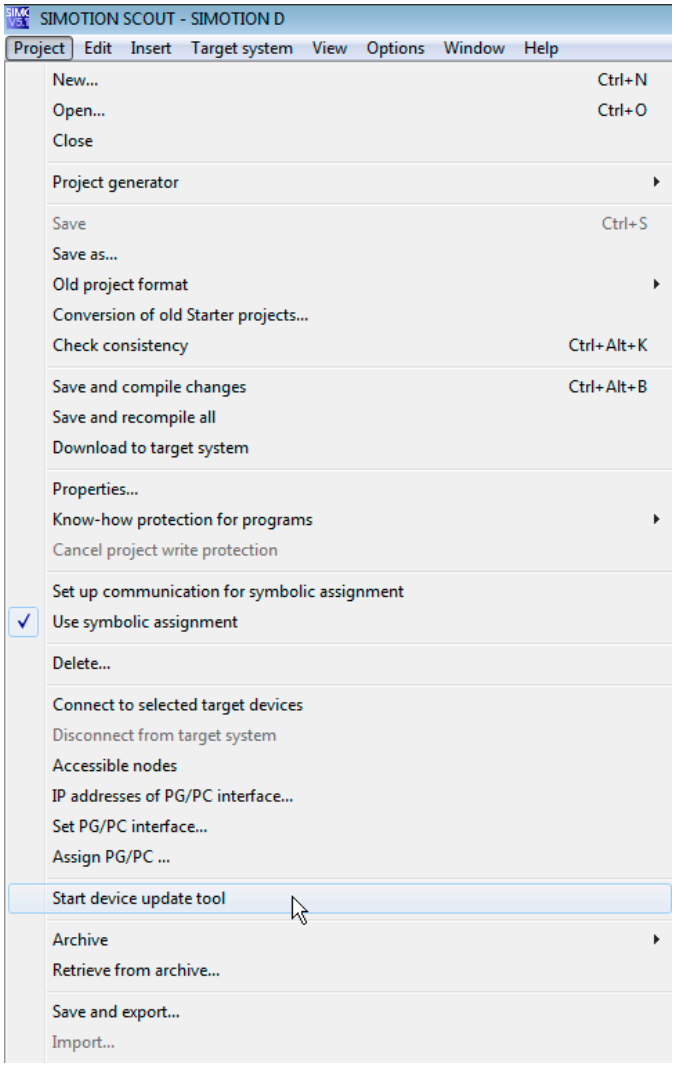


Figure 5-86 SIMOTION SCOUT: Start Device Update Tool

SIMOTION SCOUT TIA is also called via the **Project > Start Device Update Tool** menu.

Note

Creating upgrade data in SIMOTION SCOUT TIA

Note that you can create upgrade data in SIMOTION SCOUT TIA only for the currently opened project.

The processing in SIMOTION SCOUT TIA Step 3 **Select SIMOTION projects** is so omitted from the following description.

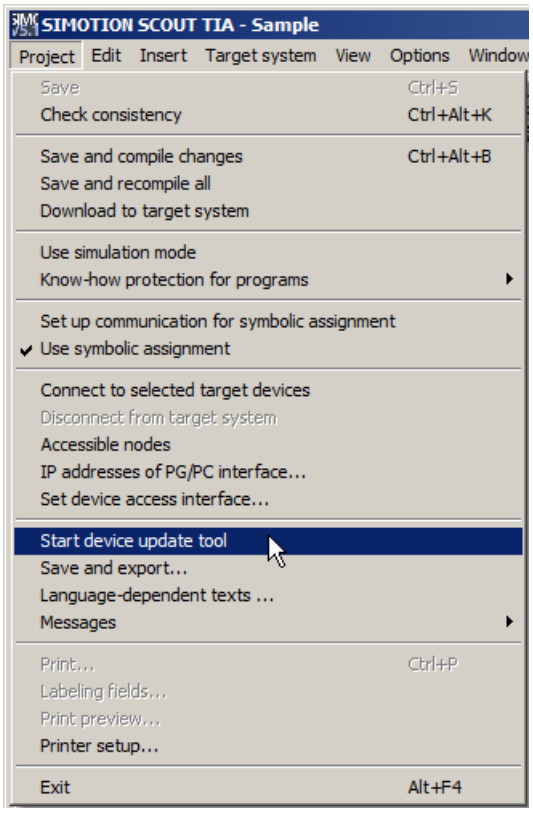


Figure 5-87 Call Device Update Tool via SIMOTION SCOUT TIA

2. The Device Update Tool opens at the start screen **Welcome to the SIMOTION Device Update tool**. (The Device Update Tool is in English.)
   Here, you can decide whether data is to be created for the purpose of upgrading or downgrading a SIMOTION device or a SIMOTION project.

   – Select **Create update data** if you wish to create data for upgrading a device or project. "Upgrading" means changing to a specific, defined configuration from upgrade data components; this may even be an older version than the current SIMOTION device.

   – Select the **Select configuration of data which were formerly stored** option if you are already upgrading a device/devices or a project/projects for a second time and wish to restore the default settings for projects, devices, and data from the last upgrade. This information can be obtained either from an upgrade archive or directly from an upgrade medium.
   Enter the path to the target folder of the upgrade archive or SIMOTION IT file or the drive for the upgrade medium plugged into the PC (USB memory stick, CF / MMC).
   The Device Update Tool will execute step-by-step. The default settings can also be changed if required.
   If you have selected this option, the **Version information** dialog is also displayed and write-protected (refer to point 7).

– Select **Create restore trigger** if you wish to create data for downgrading a device or project.
When downgrading data, activate the configuration saved on the memory card for the SIMOTION device during the last upgrade.
Further information can be found in the section Downgrading (Page 3538).

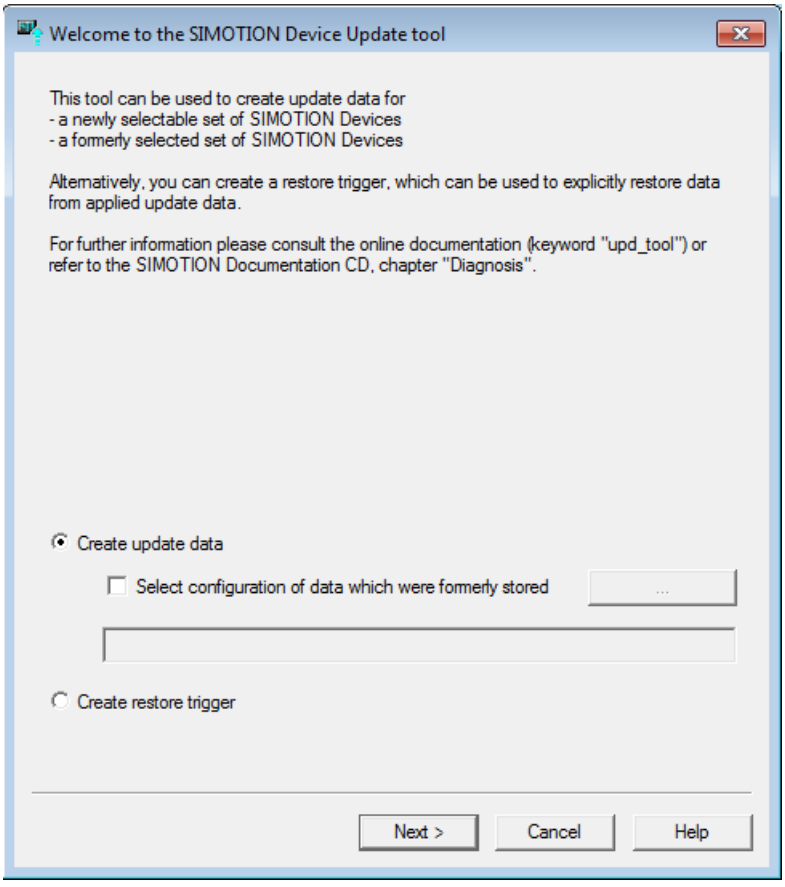


Figure 5-88 Start screen for Device Update Tool

Always click button **Next** to proceed to the next dialog box.

3. The **Select SIMOTION projects** dialog then opens. All the projects that have been saved for SIMOTION SCOUT in the default path or have already been opened on this computer using SIMOTION SCOUT appear here.

Note

Only SIMOTION SCOUT

The **Select SIMOTION projects** dialog applies only to SIMOTION SCOUT.

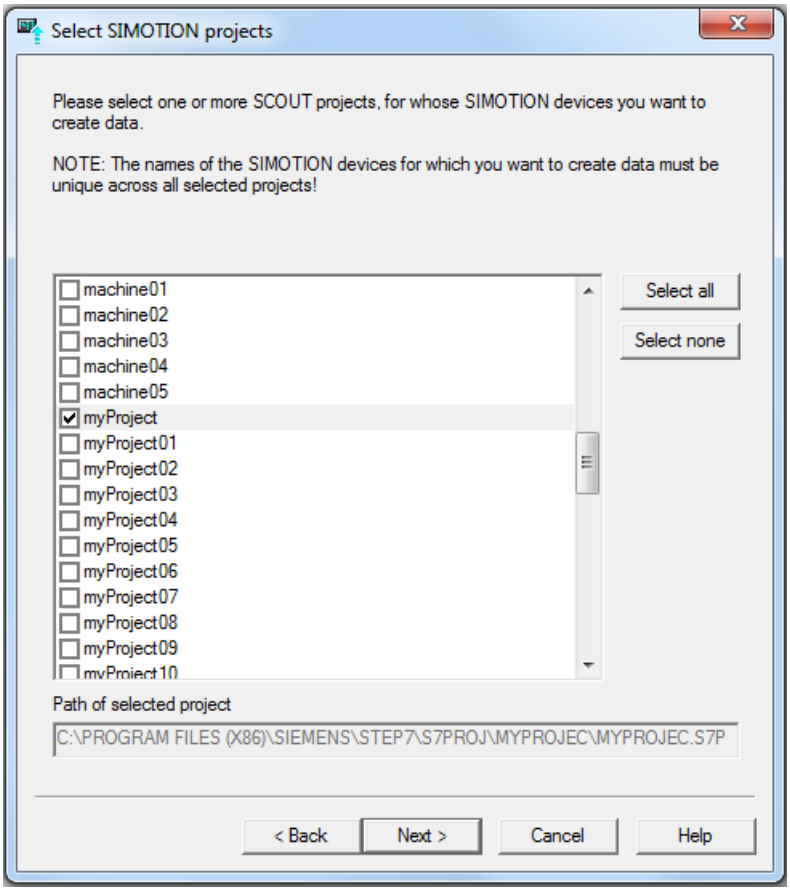


Figure 5-89 Select SIMOTION projects dialog

Please select one or more SIMOTION projects for which you wish to generate upgrade data.
The currently open project is automatically selected.
Via button **Select all** you can select all displayed projects.
Via **Select none** you can undo the project selection.
Under **Path of selected project** the path of the selected project is displayed.

4. In the next dialog (**Select Devices**), all devices assigned to the previously selected projects are displayed.
   Select the device(s) for which you wish to create upgrade data.
   Using the **Select all** button, select all devices shown that were selected in the previous dialog.
   You can deselect the devices by clicking **Select none**.
   The path for the project containing the selected device is displayed at **Project path of selected device**.

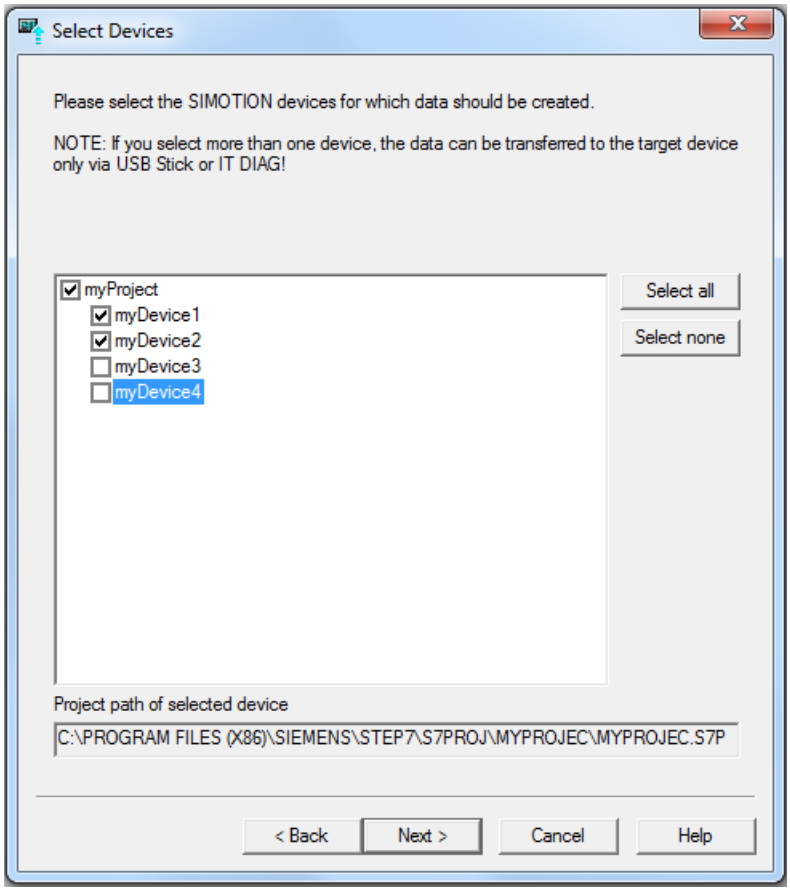


Figure 5-90 Select Devices dialog

Note

The selected devices from the selected projects must possess different device names. If there are two devices with the same name, only one of them can be selected.

See also: Assigning upgrade data to devices (Page 3484)

5. In the **Select data** dialog, select the data to be stored in the upgrade archive for each selected device. The options for choosing the upgrade data to be created can be selected separately for each SIMOTION device.

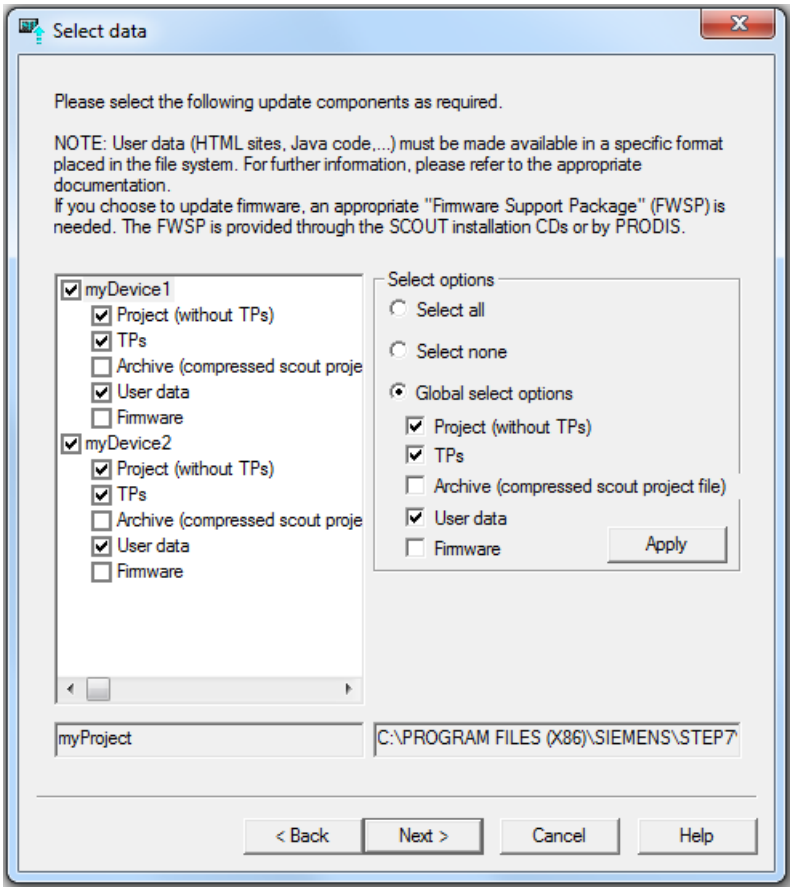


Figure 5-91 Select data dialog

The following options can be selected on the **left-hand side** of the dialog in the device tree:

- **Project (without TPs)**:
  Project data without technology packages
- **TPs** (technology packages)
  The technology packages selected in the **Select Technology Packages** dialog box in SIMOTION SCOUT are added to the upgrade data.

- **Archive (compressed scout project file)**:
  The archive data contains the entire archived SIMOTION SCOUT project belonging to this device (ZIP file).
  If the upgrade data is created via an **open** SIMOTION project, the archive cannot be created.

  Note

  If SCOUT projects already archived on the memory card are to be deleted, this can be performed via a card reader or FTP.

  Note

  Upgrading archive data in SIMOTION SCOUT TIA

  To upgrade archive data in SIMOTION SCOUT TIA, you must archive the project in the TIA Portal beforehand.

- **User data**:
  Your user data
  Further information about user data can be found at:
  Transferring device-specific user data to the update data (Page 3488)
- **Firmware**:
  In order to be able to add the firmware to the upgrade data, it is necessary to install a corresponding SIMOTION FWSP (Firmware Support Package).

  Note

  SIMOTION Firmware Support Package

  The compatible **SIMOTION FWSP** (SIMOTION Firmware Support Package) is automatically supplied as of SIMOTION SCOUT installation V4.1 SP4.

  Subsequent hotfixes (HFs) are also provided as SIMOTION FWSPs via the SIMOTION Product Support pages on the Internet at:

  SIMOTION Firmware Support Package (https://support.industry.siemens.com/cs/ww/en/view/33119786)

  Only one FWSP version can be installed within a main version.
  Check whether the installed FWSP corresponds to the desired firmware version. If this is not the case, install the correct FWSP.

You can define the selection options for all devices using the checkboxes on the **right-hand side of the dialog**:

- Using **Select options** , you can select all data for all devices with the **Select all** option.
- Selecting **Select none** deselects all devices.
- Using **Global select options** , you can select certain options which are assigned to all devices.
  For example, in this case **Project** (project data), **TPs** (technology packages), and **User data** have been selected on a global basis for the selected devices as upgrade data to be created.
- By clicking the **Apply** button, the selection made on the right-hand side of the dialog is assigned to the left-hand side, thereby validating the selection for the devices.

– The settings on the right-hand side of the dialog are not saved for the next time the Device Update Tool is run.

Note

SIMOTION P320-4

Provision is made for upgrading the project data, technology packages and user data for SIMOTION P320-4.
The firmware cannot be upgraded because of its dependence on other Windows components.

6. In the **Select additional options** dialog, you can select additional options and specify whether the user data is to be replaced or extended.

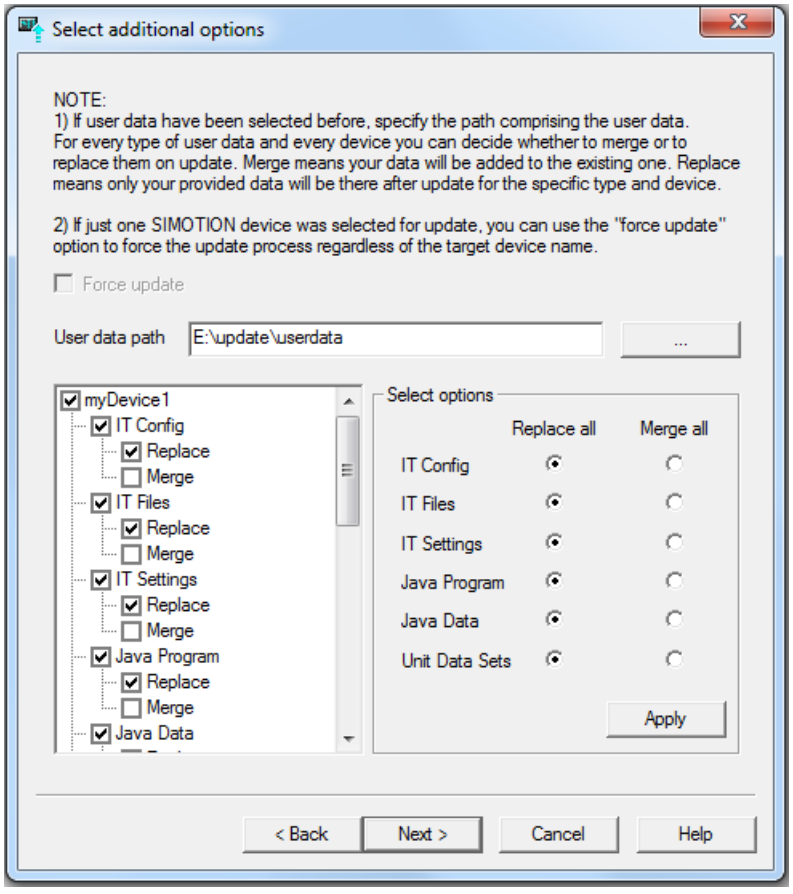


Figure 5-92 Select additional options dialog

– Using the **Force update** option, you can perform an upgrade even if the name of the device from the upgrade data is not identical to the persistently saved device name of the SIMOTION device to be upgraded.
  The device to be upgraded receives the name of the device from the upgrade data if you click **Force update**. This option is only available if precisely one device has been selected (single upgrade).
  The option is deactivated in the example shown because two devices have been selected.
– Enter the path to the user data at **User data path**. This option can only be selected if the User data is selected as the upgrade data to be created in the **Select data** dialog box.

– The data provided from the **User data path** is displayed in the folder structure on the **left-hand side of the dialog box**. The option **Merge** or **Replace** must be selected separately for each SIMOTION device.
The selection of the user data is already defined in the user-specific folder.

Note

Replace / Merge options

The options **Replace** and **Merge** refer to the directory level, i.e. file contents are not changed.

Further information about the behavior for Replace and Merge is at the end of this section.

– On the **right-hand side of the dialog** at **Select options** you can specify whether the user data is to be split and replaced according to **IT Config**, **IT Files, IT Settings, Java Program, Java Data** and **Unit Data Sets** (**Replace all**), or whether the existing data is to be merged with new information (**Merge all**).
For the significance of user data, refer to Acceptance of device-specific user data into upgrade data (Page 3488)
For example, if Unit Data Sets and Replace all are selected, and no data is contained within the relevant user-specific folder, the Unit Data Sets (unit data) on the memory card are deleted.
By clicking the **Apply** button, the selection on the right-hand side of the dialog is assigned to the left-hand side, thereby validating the selection for all devices.
The settings on the **right-hand side of the dialog** are not saved for the next time the Device Update Tool is run.

– The options can also be selected separately for each SIMOTION device on the **left-hand side of the device dialog** in the device tree.

– If the button **Next** is grayed after the option has been selected, this indicates that the selection is not complete (the option selection is checked automatically).
Please check if in the directory at **User data path** no user data has been saved for a device or if the selection on the left-hand side in the device tree or on the right-hand side in the dialog at **Select options** is incomplete.

7. Enter the version information for the upgrade data in the **Add version information** dialog:
   - Author
   - Version
   - Date
   - Comment
     The upgrade information is attached to the upgrade data and can be displayed throughout the entire upgrade procedure.

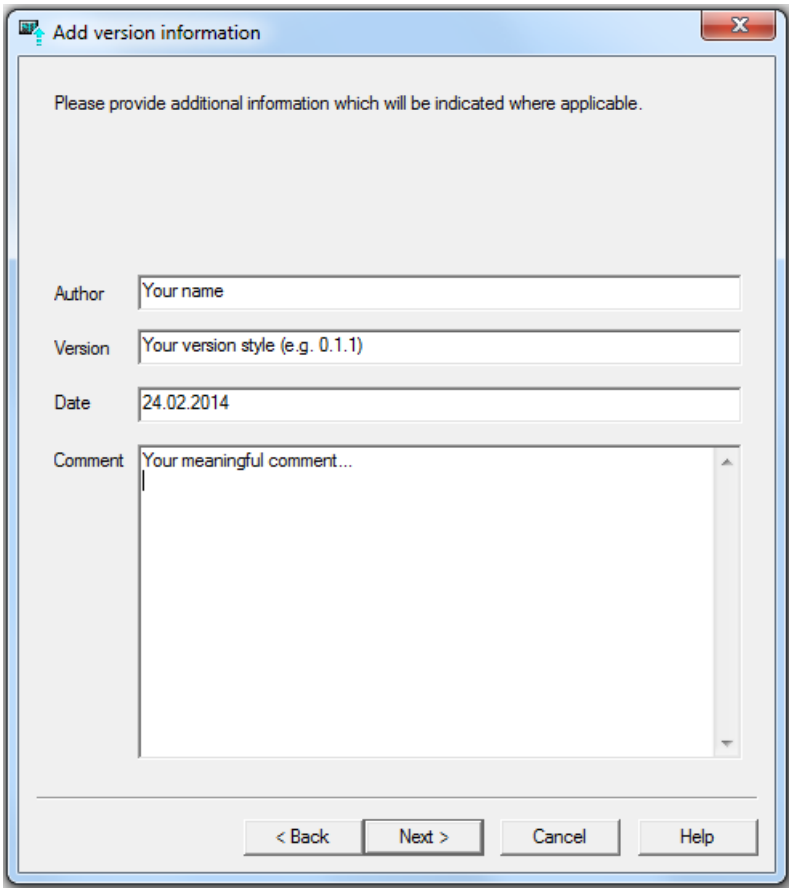


Figure 5-93 Add version information dialog

8. You can call up a summary of the projects and devices you have selected, as well as their associated components, in the **Summary** dialog box.

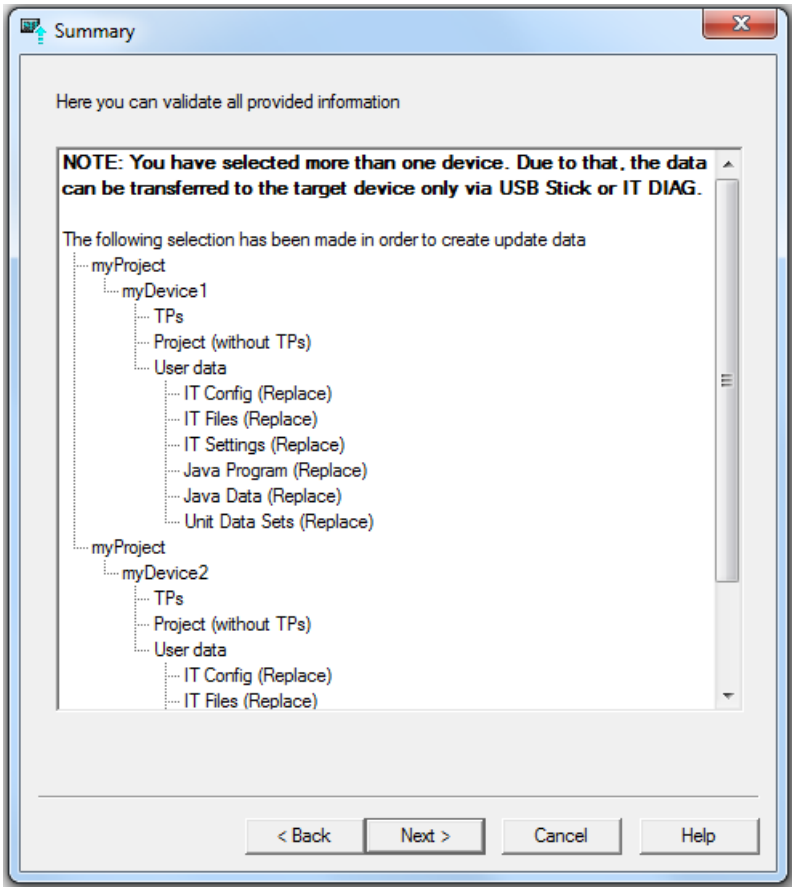


Figure 5-94 Summary dialog

9. In the **Select storage location/media** dialog, you can specify whether the upgrade data created is to be copied to a specific upgrade medium, or is to be saved in an upgrade archive in order to be written to an upgrade medium at a later date.

   – **File system** (upgrade archive)
   The data is exported to a **File system** in order to be written to an upgrade medium at a later date. An upgrade archive is created which can be called up at a later date outside the SIMOTION SCOUT context in order to load an upgrade medium (USB memory stick, CF / MMC card) or to transfer the data through a communication link (SIMOTION IT file).
   See also Creating upgrade data as an upgrade archive (Page 3511)
   To call up the upgrade archive, refer to Transferring the upgrade data (Page 3512)

   – **USB memory stick**
   If you select **USB memory stick**, the upgrade data is written directly to a USB memory stick inserted previously.
   Check the drive setting.
   This USB memory stick will have to be formatted before it can be used for upgrading purposes for the first time. Select the **Format as update stick** option.
   See also Creating upgrade data, including copying to a USB memory stick (Page 3509)
   Should you wish to undo the formatting of the USB memory stick, refer to Restoring the original status of a USB memory stick (Page 3546)

   – **CF/MMC**
   If you select **CF/MMC**, the upgrade data is written directly to a CF / MMC card.
   See also Creating upgrade data, including copying to a CF / MMC card (Page 3510)

– **SIMOTION IT file**
If you select **SIMOTION IT file**, the upgrade data will be written to a SIMOTION IT file. If upgrade data is created for several SIMOTION devices, one SIMOTION IT file will always be created **per** device with the name of the device (<devicename>.zip).
See also Creating upgrade data including creating a SIMOTION IT file (Page 3511)

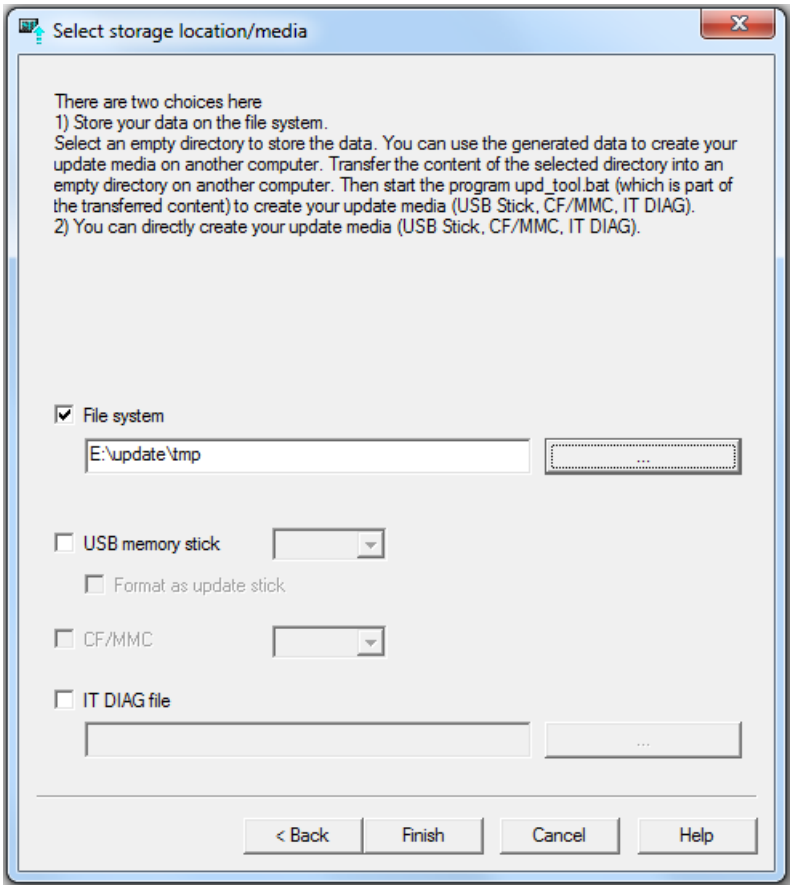


Figure 5-95 Select storage location/media dialog

Once you have selected the upgrade medium, click button **Finish** to activate the creation of the upgrade data.

10. After the upgrade data has been created, the **Succeeded** dialog appears. All successfully created upgrade data is listed here and the storage location recorded.
Close the Device Update Tool by clicking the **Close** button.

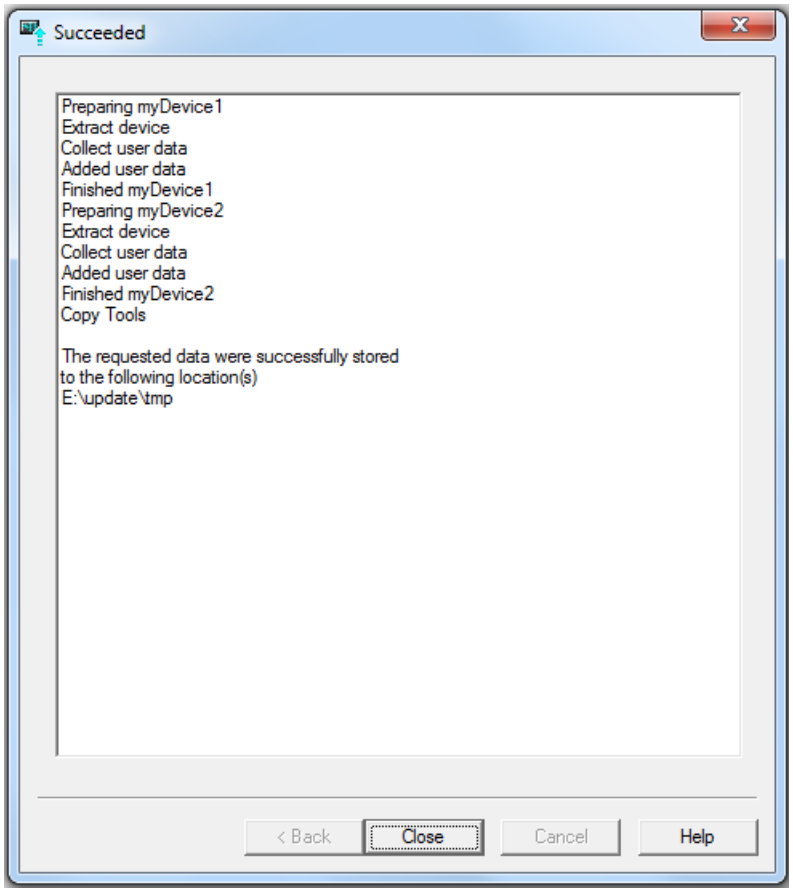


Figure 5-96 Closing window for the Device Update Tool

Canceling the Device Update Tool

As a rule, the Device Update Tool can be canceled after every step by clicking the **Cancel** button. If the write operation to the upgrade medium has already commenced, you cannot cancel this action until the procedure is complete.

Note

Exiting SIMOTION SCOUT TIA terminates the device update tool functionality

Note that when exiting SIMOTION SCOUT TIA or closing the project in SIMOTION SCOUT TIA, the functionality of the device update tool is also terminated.

- If required, close the open "Device Update Tool" dialog box with "Cancel".

Behavior after Merge or Replace

The following table shows the behavior of original and new files after **Merge** or **Replace** .

Original files	Upgrade files	After Merge	After Replace	Backup files
File 1 (original)	-	File 1 (original)	-	File 1 (original)
* File 2 (original)	File 2 (new)	File 2 (new)	File 2 (new)	File 2 (original)
File 3 (original)	-	File 3 (original)	-	File 3 (original)
--	* File 4 (new)	File 4 (new)	File 4 (new)	--
* In the example, File 2 is replaced and File 4 is added.				

With the **Merge** option, the original files are accepted and only replaced by new files with the same name or merged with new files.

With the **Replace** option, only the new files are accepted.

The original files are backed up with **Merge** and **Replace** and can be downgraded.

Creating update data, including copying to a USB memory stick

The USB memory stick can be used for single as well as multiple updates with SIMOTION D4x5-2 and SIMOTION P320-4 devices, i.e. you can update either a single device or multiple devices.
See also Device-specific update media (Page 3487)

Particular aspects of saving update data on a USB memory stick

- The USB memory stick, with at least a 512 MB capacity, appears as a logical drive when plugged into the PC.
- The USB memory stick must be formatted when it is used for the first time (**Format as update stick** option). You require administrator rights to do this.

The following hardware is required in order to update a SIMOTION D4x5-2 using a USB memory stick:

Table 5-48 Hardware required for USB memory stick updates

Module	Article no.	Minimum required HW version [1]
SIMOTION D425-2 DP	6AU1 425-2AA00-0AA0	from delivery
SIMOTION D425-2 DP/PN	6AU1 425-2AD00-0AA0	from delivery
SIMOTION D435-2 DP	6AU1 435-2AA00-0AA0	from delivery
SIMOTION D435-2 DP/PN	6AU1 435-2AD00-0AA0	from delivery
SIMOTION D445-2 DP/PN	6AU1 445-2AD00-0AA0	from delivery
SIMOTION D455-2 DP/PN	6AU1 455-2AD00-0AA0	from delivery
[1] The hardware release is listed on the SIMOTION D nameplates		

Note

Please note the following information about the USB memory stick

- The SIMOTION device is started from the USB memory stick when updating devices via a USB memory stick. For this reason, a bootable USB memory stick must be used.
  Due to the rapid developments within the market for USB memory sticks it is not possible to give any concrete recommendations except for SIMATIC USB memory sticks.
  Further information can be obtained on the Internet under Industry Online Support, SIMOTION: Which USB memory sticks are recommended for updating a module using a USB? (https://support.industry.siemens.com/cs/ww/en/view/32580863)
- If you are using the USB memory stick to update SIMOTION devices for the first time, it is **imperative** that you format the stick first. Unless it is formatted, the USB memory stick will not work, nor will the update process.
  Once the USB memory stick has been formatted, the amount of space which can be used in Windows is limited to 450 MB, irrespective of how much memory capacity is actually available.
- During formatting, all the existing data on the USB memory stick is permanently deleted.
- The system checks the size of the USB memory stick. If this is less than 512 MB, this option is not available in the **Store data** dialog box.
- The USB memory stick has to be restored to its original formatting if you wish to use it for any other purpose.
  Refer to: Restoring the original status of a USB memory stick (Page 3546).

Creating update data, including copying to a CF / MMC card

The CF / MMC card can only be used to perform single updates for SIMOTION C and SIMOTION D devices, i.e. only one device can be updated.
See also Device-specific update media (Page 3487)

Particular aspects of saving update data on a CF / MMC card

- The CF / MMC card is inserted into the PC or CF / MMC card adapter and is visible as a logical drive

  Note

  Note the following information regarding the CF / MMC card

  - You can only perform updates using a CF / MMC card if there already is a bootable SIMOTION system present on the target card, i.e. at the very least, executable firmware ≥ V4.1 SP2 must be present.
    A blank CF / MMC card cannot be updated.
  - If the **Device Update Tool** has not finished or the copying procedure has been interrupted, updating with the CF / MMC card will not be possible.
  - If an error such as PowerFail occurs while data are being written to the CF / MMC card, all the data on the card may be lost.
  - In the case of MMC (SIMOTION C), restoring can only be accomplished using a restore archive or SIMOTION IT.

Creating update data including generating a SIMOTION IT file

With SIMOTION IT, you can carry out single and multiple updates, i.e. both single and multiple SIMOTION devices can be updated.

SIMOTION IT is suitable as an update medium for **all** update scenarios and **all** SIMOTION devices.

If update data are created for several SIMOTION devices, essentially one SIMOTION IT file will be created **per** device with the name of the device (<devicename>.zip).
See also Device-specific update media (Page 3487)

Specifics for saving update data as a SIMOTION IT file

Note

Please note the following information about SIMOTION IT file

- If there is already a SIMOTION IT file in the selected directory, an error message will appear and the procedure will be aborted.
- First delete the existing SIMOTION IT file from the directory before storing the new SIMOTION IT file.

Creating update data as an update archive

The update data are created by an operator-guided process using the Device Update Tool in SIMOTION SCOUT, and then saved as an update archive in a directory.

Select the **File system** option in the second-to-last dialog box of the Device Update Tool **Select storage location/media** . Enter the path to a blank directory here. As soon as the Device Update Tool is closed, the update data selected and created previously are written to the directory and stored there for later use.

The update archive can be called up independently of SIMOTION SCOUT at a later point in time, so that it can be copied to an update medium.

Particular aspects of saving the update data as an update archive

Note

Note the following information about the update archive

- If data are already present in the directory selected, an error message is output and the process is stopped. No data is written.
  Select another blank directory in the **Select storage location/media** dialog. The update data are then written to the newly selected directory.
- If a CF / MMC card has been selected as an update medium, the update medium may only contain one device.

5.4.3.3 Copying to the update media on the basis of an update archive

Copying to an update medium without SIMOTION SCOUT

Requirements

- The update data has already been created and is located in an update archive
- SIMOTION Device Update Tool (update archive component)
- SIMOTION SCOUT does **not** need to be installed
- Released for the Windows XP SP3, Windows 7 Ultimate and Windows 7 Professional operating systems

Sequence

The data is copied to the update media independently of SIMOTION SCOUT; this can therefore be carried out at either the machine manufacturer's site or the machine operator's site.

All that is required is the update archive in which all the update data and the SIMOTION Device Update Tool are saved.

Transferring the update data

The SIMOTION Device Update Tool is part of the update archive that is created within the context of SIMOTION SCOUT.

You call up the Device Update Tool for the purpose of transferring update data from the update archive to an update medium. This involves starting the SIMOTION Device Update Tool via a batch file independently of SIMOTION SCOUT.

Proceed as follows:

1. Open the update archive directory.
2. Start the **upd_tool.bat** batch file from the directory. The SIMOTION Device Update Tool is started.

3. The **Welcome to the SIMOTION Device Update tool** dialog appears as the start screen.
   The Device Update Tool is in English.

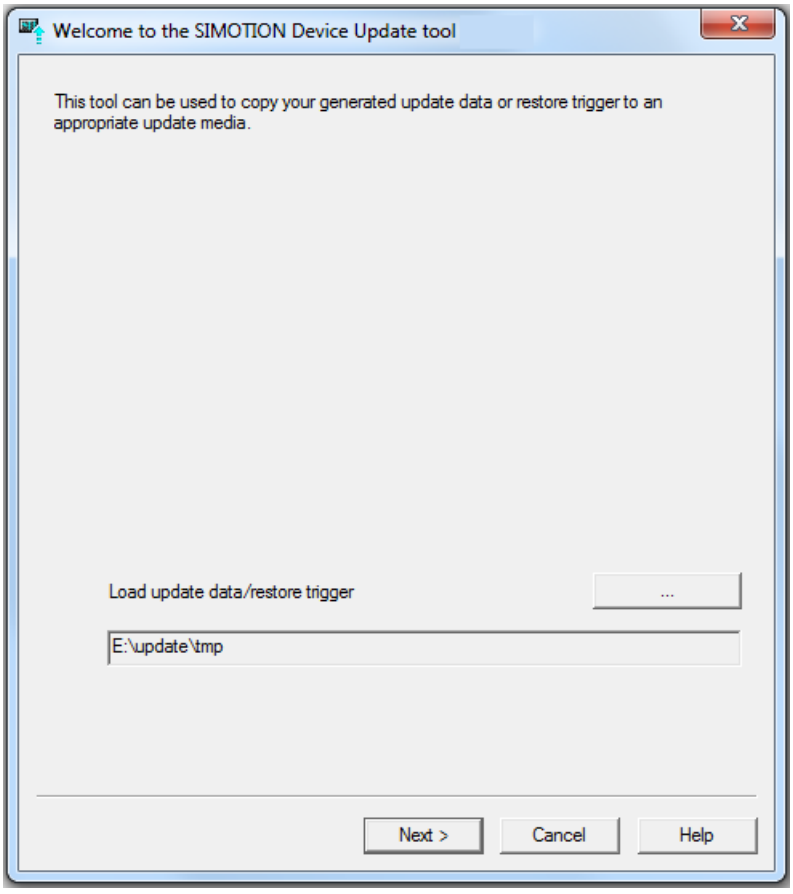


Figure 5-97 Device Update Tool start screen for transferring update data to an update medium

4. Click **Load update data/restore trigger** to call up either an update archive created using SIMOTION SCOUT or a restore archive.
   Select the path in which the update archive has been stored.
   Click button **Next** to proceed to the next dialog box.

5. The **Version Information** dialog box contains the stored identification data that was entered as the update data was created.

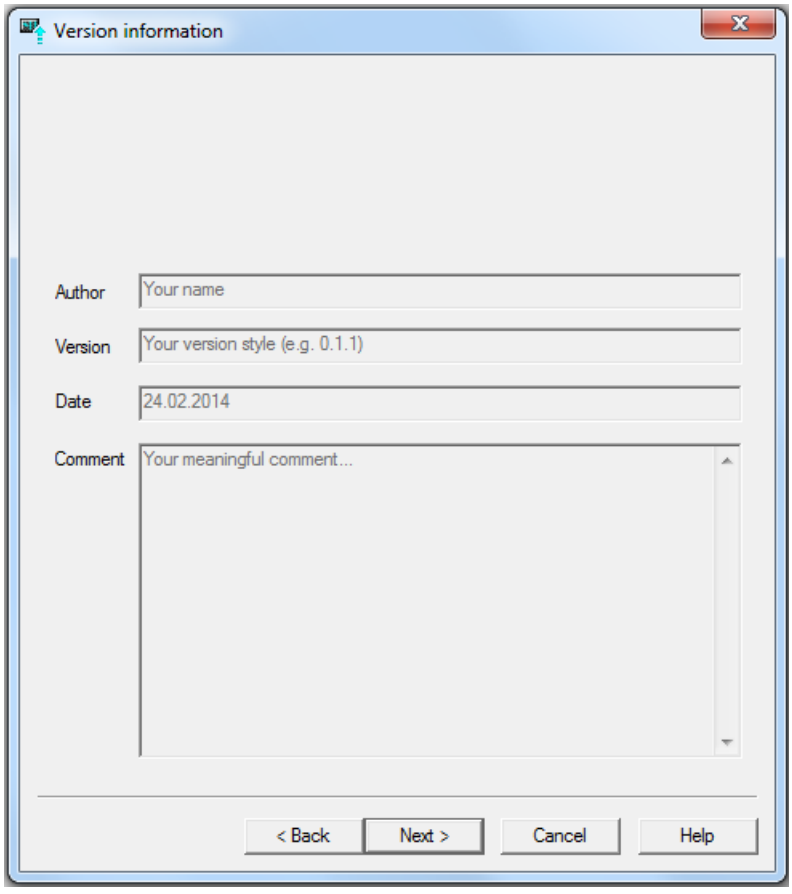


Figure 5-98 Displaying the identification data for the update archive

6. Select an update medium in the **Select storage location/media** dialog box. Data will then be written directly to this medium. You can select from the following:
   - USB memory stick
   - CF / MMC card
   - SIMOTION IT
     The update medium you can use depends on the SIMOTION device or on whether a single or a multiple update is taking place.
     You can find an overview under: Device-specific update media (Page 3487)

   Click button **Finish** to activate and execute the copying of data to the update medium.
   The following data can be created during this procedure:

   - **USB memory stick**
     Update medium for SIMOTION D4x5-2 and SIMOTION P320-4
     Update data for one or more SIMOTION devices is saved on this update medium.
     If you are using the USB stick as an update medium for the first time, you must format it. To do this, select **Format as update stick**.
     See also Copying data to a USB memory stick (Page 3518)
   - **CF- / MMC**
     Memory cards are an update medium for single updates on the SIMOTION C and SIMOTION D.
     Update data for a single SIMOTION device is saved on this update medium.
     See also Copying data to a CF / MMC card (Page 3519)

– **SIMOTION IT file**
SIMOTION IT is an update medium suitable for all update scenarios (single and multiple updates) and for all SIMOTION hardware platforms (SIMOTION C, D, P).
If update data is generated for several SIMOTION devices, essentially one SIMOTION IT file is created **per** device with the name of the device (<devicename>.zip) is created.
See also Creating a SIMOTION IT file and copying the update data (Page 3519)

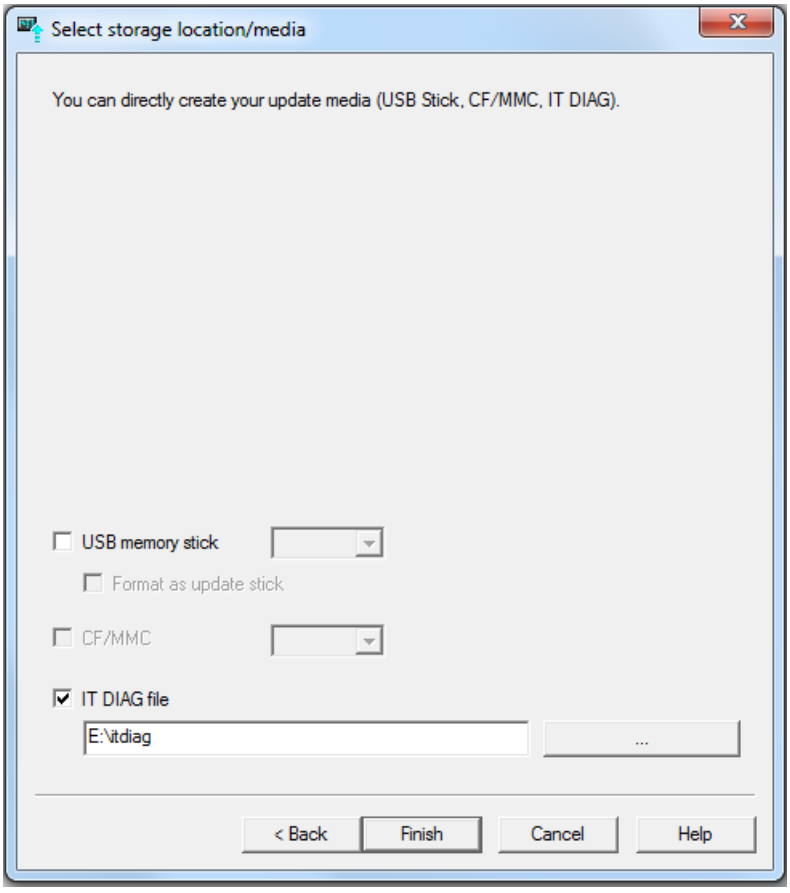


Figure 5-99 Selecting the update medium

7. The final **Succeeded** dialog in the Device Update Tool shows the devices for which update data from the update archive has been stored, and at what location.

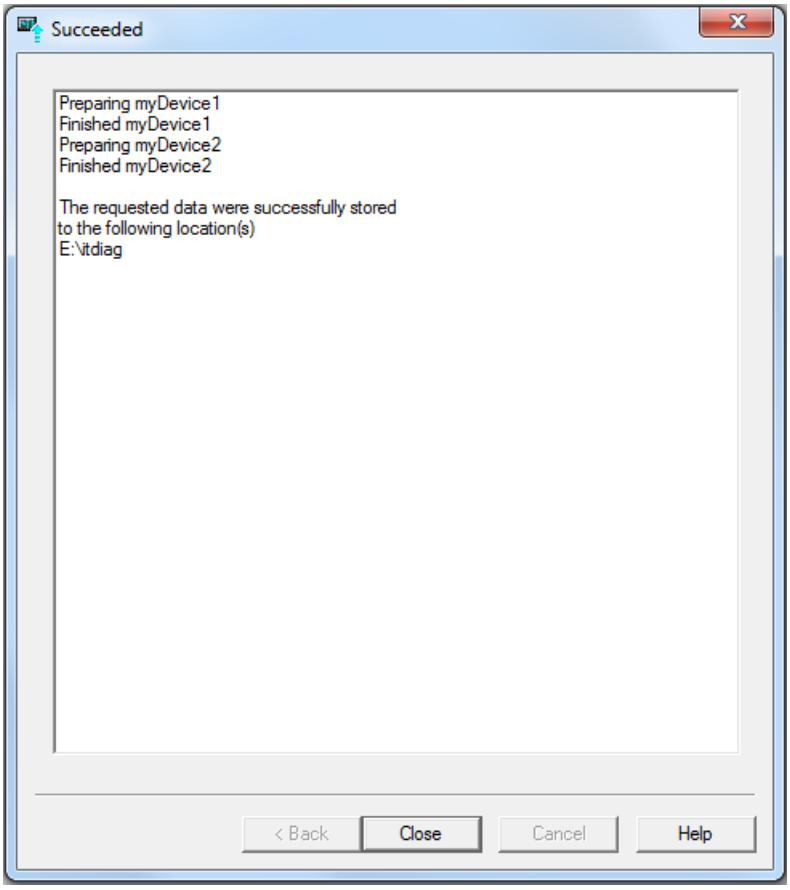


Figure 5-100 Final dialog Succeeded

Canceling the Device Update Tool

The Device Update Tool can be canceled after every step with the **Cancel** button. However, if the data write operation to the update medium has already started, you cannot exit from the tool until the write operation is complete.

Copying data to a USB memory stick

The USB memory stick can be used for single as well as multiple updates with SIMOTION D4x5-2 and SIMOTION P320-4 devices, i.e. you can update either a single device or multiple devices.
See also Device-specific update media (Page 3487)

Note

The SIMOTION device is started from the USB memory stick when updating devices via a USB memory stick. For this reason, a bootable USB memory stick must be used.

With the exception of SIMATIC USB memory sticks, due to the rapid developments within the market for USB memory sticks it is not possible to make any other concrete recommendations.

For information on this, go to:

https://support.industry.siemens.com/cs/ww/de/view/32580863

Particular aspects of transferring update data from the update archive to a USB memory stick

- The USB memory stick, with at least a 512 MB capacity, appears as a logical drive when plugged into the PC.
- The USB memory stick must be formatted when it is used for the first time (**Format as update stick** option). You require administrator rights to do this.
- The following hardware is required in order to update a SIMOTION D4x5-2 using a USB memory stick:

Table 5-49 Hardware required for USB memory stick updates

Module	Article no.	Minimum required HW version [1]
SIMOTION D425-2 DP	6AU1 425-2AA00-0AA0	from delivery
SIMOTION D425-2 DP/PN	6AU1 425-2AD00-0AA0	from delivery
SIMOTION D435-2 DP	6AU1 435-2AA00-0AA0	from delivery
SIMOTION D435-2 DP/PN	6AU1 435-2AD00-0AA0	from delivery
SIMOTION D445-2 DP/PN	6AU1 445-2AD00-0AA0	from delivery
SIMOTION D455-2 DP/PN	6AU1 455-2AD00-0AA0	from delivery
[1] The hardware release is listed on the SIMOTION D nameplates		

Note

Please note the following information about the USB memory stick

- If you are using the USB memory stick to update SIMOTION devices for the first time, it is **imperative** that you format the stick first. Unless it is formatted, the USB memory stick will not work, nor will the update process.
  Once the USB memory stick has been formatted, the amount of space which can be used in Windows is limited to 450 MB, irrespective of how much memory capacity is actually available.
- During formatting, all the existing data on the USB memory stick is permanently deleted.
- The system checks the size of the USB memory stick. If this is less than 512 MB, this option is not available in the **Store data** dialog box.
- The USB memory stick has to be restored to its original formatting if you wish to use it for any other purpose.
  Refer to: Restoring the original status of a USB memory stick (Page 3546).

Copying data to a CF / MMC card

The CF / MMC card can only be used to perform single updates for SIMOTION C and SIMOTION D devices, i.e. only one device can be updated.
See also Device-specific update media (Page 3487)

Particular aspects of transferring update data from the update archive to a CF / MMC card

- The CF / MMC card is inserted into the PC or CF / MMC card adapter and is visible as a logical drive.

Note

Note the following information about the CF / MMC card

- You can only perform updates using a CF / MMC card if there already is a bootable SIMOTION system present on the target card, i.e. at the very least, executable firmware ≥ version V4.1 SP2 must be present.
  A blank CF / MMC card cannot be updated.
- If the Device Update Tool has not finished or the copying procedure has been interrupted, updating with the CF / MMC card will not be possible.
- If an error such as PowerFail occurs while data are being written to the CF / MMC card, all the data on the card may be lost.
- If the update fails using MMC (in the case of C2xx) or if the previous configuration has to be accessed for other reasons, the restore data must be created through SIMOTION SCOUT at the machine manufacturer's site or the data can be restored using SIMOTION IT.

Creating a SIMOTION IT file and copying the update data

With SIMOTION IT, you can carry out single and multiple updates, i.e. both single and multiple SIMOTION devices can be updated. SIMOTION IT is suitable as an update medium for all update scenarios and all SIMOTION devices.

If update data are created for several SIMOTION devices, essentially one SIMOTION IT file will be created **per** device with the name of the device (<devicename>.zip).
See also Device-specific update media (Page 3487)

Specifics when transferring update data from the update archive to the SIMOTION IT file

Note

Note the following information

- If there is already a SIMOTION IT file in the selected directory, an error message will appear and the procedure will be aborted.
- First delete the existing SIMOTION IT file from the directory before storing the new SIMOTION IT file.

5.4.3.4 Transferring update data to the SIMOTION device

After update data has been created, it must be transferred to the relevant SIMOTION device.

This operation is based on the update media to which the update data has already been written. An update medium is either plugged into the SIMOTION device being updated (USB memory stick or CF / MMC card) or the update data can be transferred through a communication link to the relevant SIMOTION device (SIMOTION IT).

The update procedure is triggered by restarting the SIMOTION device to be updated. After the update data have first been transferred to the memory card in the device (this applies to both the USB memory stick and the SIMOTION IT), the data is backed up. The update data is activated as the current configuration after the SIMOTION device has been restarted.

Note

If the parameterization of the SINAMICS Safety Integrated functions is contained in the update data, these functions must be activated after the update.

See also *SINAMICS S120 Safety Integrated Function Manual*.

Requirements for updating:

- Firmware on the SIMOTION device to be updated, V4.1 SP2 or higher
- Update data must contain firmware V4.1 SP2 or higher (if firmware is selected).

Updating devices

- Updating a module with a USB memory stick (Page 3530)
- Updating a module with a CF / MMC card (Page 3533)
- Updating a module with SIMOTION IT (Page 3534)

Behavior of the retain data on updating (and restoration)

Overview

After the update data has been transferred to the SIMOTION device, the device is restarted and the update data is accepted.

During start-up, the non-volatile data stored in the SIMOTION device is checked to see if it is compatible with the current configuration.

Compatibility checks are conducted in relation to:

- Retain kernel data
- Device-global retain variables
- Unit retain variables
- TO retain data

Behavior of SIMOTION retain data during updating / restoring (as of V4.4)

During updating, existing retain data are retained under the following conditions:

- If no changes have been made to the retain data.
- In the case of TO retain data, if the TO name is unchanged.
- In the case of structural changes to unit-global retain data segments of program units, if
  - the name and type of a variable is unchanged.
  - the name of a variable is unchanged but its data type has changed and the value to be restored fits the value range of the new data type.

Otherwise the retain data will be initialized.

If no retain data is contained in the new project, the entire retain memory will be initialized in the control.

Behavior of the SINAMICS retain data

The SINAMICS retain data are kept during updates. All SINAMICS retain data can be located in the retain memory and interpreted by a new version of the firmware.

Receiving unit data

Behavior of SIMOTION unit data during updating

After the update data has been transferred to the SIMOTION device, the device is restarted and the update data is applied.

When loading a unit data set, a check is executed for compatibility with the current unit. Data saved with _saveUnitDataSet is only compatible if the data structure within the unit has not changed.

Data backed up using _exportUnitDataSet() also remains compatible after structure changes of the unit. This means that the data is retained even after a version or project change (when the unit and the data have the same name).

Whether existing unit data sets should be kept (Merge) or deleted (Replace) can be specified using the **Device Update Tool**.

New unit data sets can replace the existing sets (Replace) or be merged with them. **Replace** means that all existing unit data sets are deleted and the new ones are accepted. **Merge** means that all existing unit data sets are retained and the new unit data sets are merged with them. The only exception are the unit data sets with identical names. In this case, the existing unit data sets will be replaced by the new ones.

For more information about the Replace and Merge options, see:

Creating update data with the Device Update tool (Page 3492) in the **Select additonal options** dialog box.

Additional references

Additional information on the functions can be found in:

- SIMOTION Basic Functions Function Manual
- SIMOTION System Functions/Variables Devices List Manual
- Configuration Manual SIMOTION SCOUT
- SIMOTION SCOUT Online help

SIMOTION device status during updating

Status of SIMOTION devices

The table below provides an overview of the possible statuses, the corresponding LEDs, and their significance for SIMOTION devices during the update procedure.

Table 5-50 LED display on SIMOTION D for "Copying update data via USB memory stick to the memory card"

LED display	Status	Meaning	Description
RDY LED flashes yellow/ green at 0.5 Hz; later longer green and yellow phases (< 10 s)	USB in Progress	Update running, data is being copied from the USB memory stick to the memory card.	Update in progress. Do **not** switch off the SIMOTION device.
RDY LED constantly red	USB ERROR 1	Error copying from the USB memory stick.	Error occurred while copying. SIMOTION device is waiting. No communication possible with the SIMOTION device. **Required operator actions:** 1. Switch off the device. 2. Remove the USB memory stick. 3. Switch on the device. The device boots up in the starting configuration.
RDY LED must be constantly green for > 10 s so that the procedure is completed.	USB Done	Data has been successfully copied from the USB memory stick to the memory card.	SIMOTION device is waiting. No communication possible with the device. **Required operator actions:** 1. Switch off the device. 2. Remove the USB memory stick. 3. Switch on to start the actual update procedure (see following table: LED display when updating the devices).
RDY LED flashes yellow at 0.5 Hz (longer than 10 s)	USB BOOT ERROR	Booting is not possible from the used USB memory stick.	See Device-specific update media (Page 3487).

Table 5-51 LED display when updating the devices

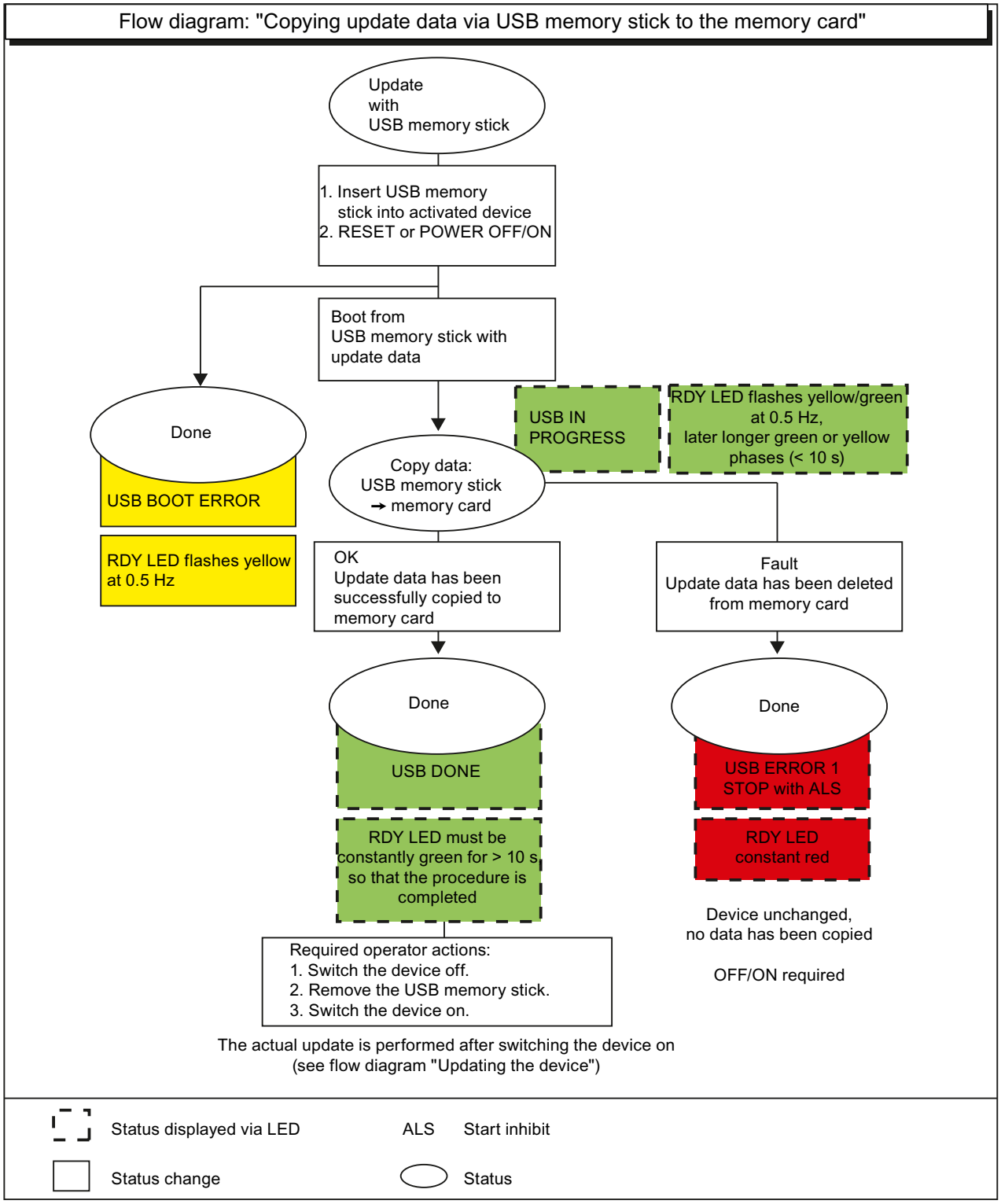
LED display	Status	Meaning	Description
SIMOTION D/P: SF LED flashes green at 0.5 Hz SIMOTION C: SF LED flashes red at 2 Hz	IN PROGRESS	Update running	Update in progress. Do not switch off the SIMOTION device.
SF LED flickers red (10 Hz)	ERROR 2	**Possible errors are**: • Error backing up the configuration; update has not been accepted. • Defective update; update data has been copied, but is defective.	SIMOTION device is booting up in **STOP with start inhibit** After OFF/ON, the device boots up in the starting configuration. If required, a restore must be activated.
SF LED off [1]	DONE 1	Update successfully completed	SIMOTION device boots up with the updated configuration.

[1] The update has been successfully completed when the SF LED goes out; the device then boots up automatically in the updated configuration. (SF LED display then depends on the respective mode of the device.)

The update status is shown not only via the LEDs, but also in the device diagnostics on the **Syslog** tab. **Syslog** can also be read via SIMOTION IT.

Flow diagram: "Copying update data via USB memory stick to the memory card"

The figure below illustrates the processes that run during updating within the SIMOTION device, using the USB memory stick as an example of an update medium.
The status and corresponding LED are displayed with different colors by way of analogy.


Flow diagram: "Copying update data via USB memory stick to the memory card"
Update with USB memory stick
1. Insert USB memory stick into activated device
2. RESET or POWER OFF/ON
Boot from USB memory stick with update data
Done
USB BOOT ERROR
RDY LED flashes yellow at 0.5 Hz
USB IN PROGRESS
RDY LED flashes yellow/green at 0.5 Hz, later longer green or yellow phases (< 10 s)
Copy data: USB memory stick → memory card
OK
Update data has been successfully copied to memory card
Fault
Update data has been deleted from memory card
Done
USB DONE
RDY LED must be constantly green for > 10 s so that the procedure is completed
Done
USB ERROR 1
STOP with ALS
RDY LED constant red
Device unchanged, no data has been copied
OFF/ON required
Required operator actions:
1. Switch the device off.
2. Remove the USB memory stick.
3. Switch the device on.
The actual update is performed after switching the device on (see flow diagram "Updating the device")
Status displayed via LED
ALS Start inhibit
Status change
Status

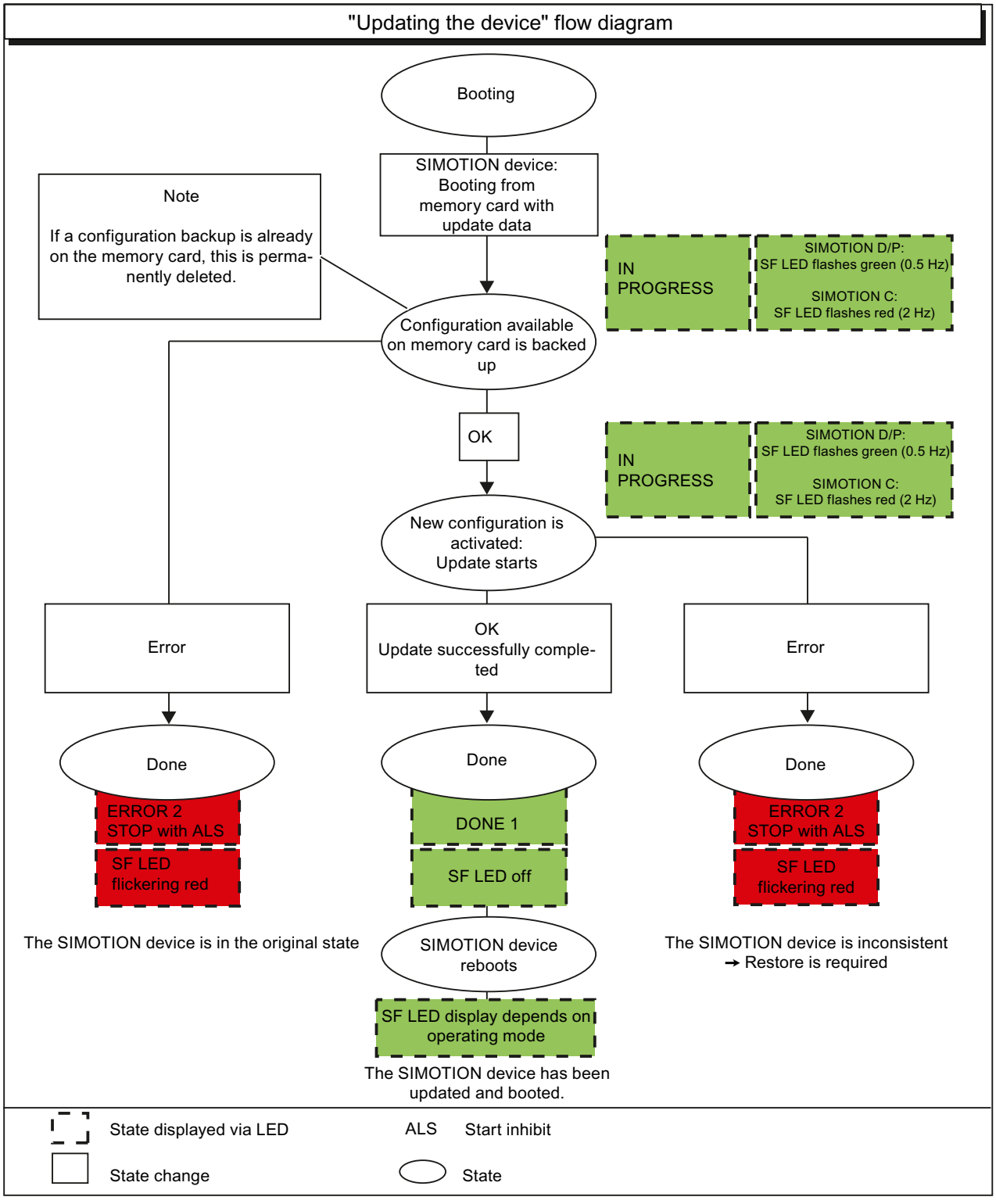
Note

In this procedure, only the update data is copied from the USB memory stick to the memory card, e.g. for SIMOTION D from the USB memory stick to the CF card.

The actual update procedure must be initiated in a further step (see flow diagram: "Updating the device").

Flow diagram: "Updating a SIMOTION device"

The figure below shows how a SIMOTION device can be updated once the update data has been copied to the device's memory card.
The status and corresponding LED are displayed with different colors by way of analogy.


"Updating the device" flow diagram
Booting
SIMOTION device: Booting from memory card with update data
Note
If a configuration backup is already on the memory card, this is permanently deleted.
IN PROGRESS
SIMOTION D/P: SF LED flashes green (0.5 Hz)
SIMOTION C: SF LED flashes red (2 Hz)
Configuration available on memory card is backed up
OK
IN PROGRESS
SIMOTION D/P: SF LED flashes green (0.5 Hz)
SIMOTION C: SF LED flashes red (2 Hz)
New configuration is activated: Update starts
Error
OK
Update successfully completed
Error
Done
ERROR 2
STOP with ALS
SF LED flickering red
Done
DONE 1
SF LED off
Done
ERROR 2
STOP with ALS
SF LED flickering red
The SIMOTION device is in the original state
SIMOTION device reboots
SF LED display depends on operating mode
The SIMOTION device has been updated and booted.
The SIMOTION device is inconsistent
→ Restore is required
State displayed via LED
ALS
Start inhibit
State change
State

Transferring update data from the update medium to SIMOTION devices

Updating a module with a USB memory stick

Requirement

USB memory stick updates are supported by the SIMOTION D4x5-2 and SIMOTION P320-4 devices.

Requirements for updating with a USB memory stick:

- USB memory stick to which the required update data has already been copied.
- SIMOTION device or devices that need to be updated.
- A CF card with sufficient memory capacity for the data to be copied must be inserted into the SIMOTION device(s) to be updated.

The following hardware is required in order to update a SIMOTION D4x5-2 using a USB memory stick:

Table 5-52 Hardware required for USB memory stick updates

Module	Article no.	Minimum required HW version [1]
SIMOTION D425-2 DP	6AU1 425-2AA00-0AA0	from delivery
SIMOTION D425-2 DP/PN	6AU1 425-2AD00-0AA0	from delivery
SIMOTION D435-2 DP	6AU1 435-2AA00-0AA0	from delivery
SIMOTION D435-2 DP/PN	6AU1 435-2AD00-0AA0	from delivery
SIMOTION D445-2 DP/PN	6AU1 445-2AD00-0AA0	from delivery
SIMOTION D455-2 DP/PN	6AU1 455-2AD00-0AA0	from delivery
[1] The hardware release is listed on the SIMOTION D nameplates		

Note

SIMOTION D4x5-2 - upgrading using a USB memory stick

When upgrading using a USB memory stick, the points listed below must be noted.

The version information refers to the firmware version on the CF card (that is, the original version from which upgrading is to take place).

- Version V4.2: Update via USB memory stick is not supported.
- Version < V4.3 SP1 HF12: The USB memory stick may only be inserted when the Control Unit is switched off.
- Version ≥ V4.3 SP1 HF2: The USB memory stick can be inserted when the Control Unit is switched on or off.

Sequence of operations for updating SIMOTION D4x5-2

When using a USB memory stick to perform an update, proceed as follows:

1. Check the position of the service selector switch
   (with D4x5-2: **SVC/NCK**).
   The switch position must be at **0**.
2. Insert the USB memory stick into one of the two USB interfaces on the SIMOTION D4x5-2 after it has been switched on. (Only one USB memory stick may be inserted.)
3. Switch the device OFF/ON or reset it using the RESET button.
4. SIMOTION D4x5-2 will now begin copying the data from the USB memory stick to the CF card.
   The copying procedure starts with some LED state changes and is finally displayed by a yellow/green flashing RDY LED (0.5 Hz).
   In the end phase of the update, the RDY LED has longer green and yellow phases (< 10 s).
   After completion of the copying procedure the RDY LED changes to:
   - **Constant green**, only when the RDY LED is constantly green for > 10 s, has the copying procedure been completed successfully.
   - **Constant red**, if copying was not successful.

Meaning of the 7-segment display	Description
0	No memory card available.
1	Internal error.
2	Internal error.
3	No or incomplete update data available on the USB memory stick.
4	No update data can be copied because there is no project available in the SIMOTION device, but update data for several SIMOTION devices is available on the USB memory stick. Because of the missing project on the SIMOTION device, assignment to the data on the USB memory stick is not possible.
5	The update data cannot be copied, for example, because there is not sufficient memory space.
6	Internal error.
7	Unknown update data available on the USB memory stick.

5. Switch off the SIMOTION D4x5-2 and remove the USB memory stick.
6. Switch the SIMOTION D4x5-2 on again. The system now starts the update procedure.
   The **SF LED** flashes green (0.5 Hz) during the update. The procedure can take several minutes.
7. Observe the green flashing of the **SF LED**:

- As soon as the update procedure has been completed successfully, the **SF LED** goes out. The device then boots up automatically in the updated configuration.
  (The **SF LED display** then depends on the respective operating state of the device.)
- If the update procedure was not successful, the **SF LED** flickers red (10 Hz).

Note

If the update has not been successful in terms of the application (for example, the machine is not behaving as desired), it can be undone using an operator action.

Note

If the firmware is being updated for SIMOTION D, depending on the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, PM340, etc.), the component firmware will also be automatically updated. This procedure can take up to several minutes and is indicated by LED displays.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY-LED flashes slowly (0.5 Hz)
- FW update complete: RDY-LED flashes rapidly (2 Hz), **POWER ON** required

These flashing patterns are also displayed on the yellow RDY LED on the SIMOTION D/CX32/CX32-2, and indicate that the firmware update has been carried out on components connected to the SIMOTION D/CX32/CX32-2 or that all components have completed the firmware update.

Components that require a POWER ON after the firmware update will signal this through a rapidly flashing RDY LED.
Switch the 24 V supply for initializing Off/On (POWER ON) for the relevant components.

Sequence of operations for updating SIMOTION P320-4

1. The USB memory stick to which the update data has been copied is inserted into one of the available USB ports on the SIMOTION P320-4. Updating or restoring is started automatically.
2. If the SIMOTION P320-4 is running at that point, you will be prompted to terminate the SIMOTION P.

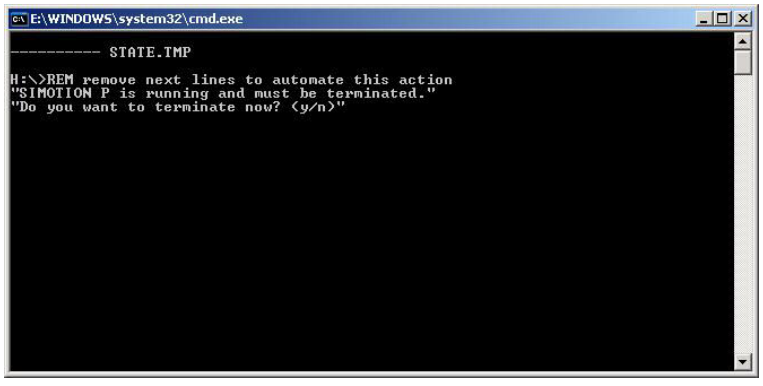


Figure 5-101 SIMOTION P: Query after the USB memory stick is inserted

3. After you click **y** to confirm, the device is switched off, and the update is started automatically. The data is copied from the USB memory stick to the SIMOTION P's hard disk. While this process is running, you will be kept informed of progress via SIMOTION P Update.
4. Once the copying procedure is complete, you will be prompted whether you wish to restart the SIMOTION P. If you click **y** to confirm, the SIMOTION P is restarted.

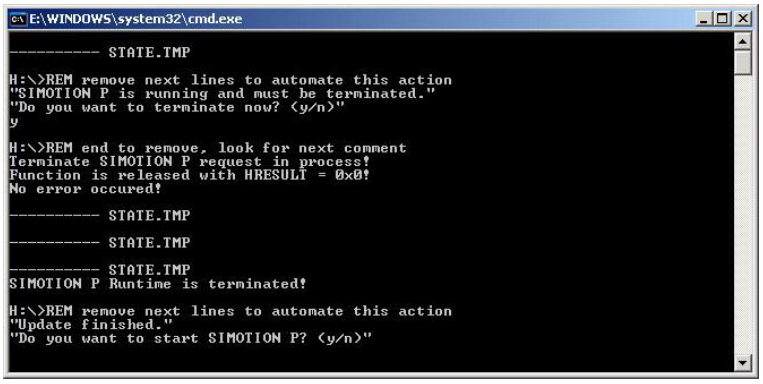


Figure 5-102 SIMOTION P: Device restart enquiry

5. The update or restore is activated when the SIMOTION P is restarted.

Updating a module with a CF / MMC card

Precondition

- CF / MMC card to which the required update data has already been copied.
- SIMOTION device which is to be updated.
- Sufficient storage space on the memory card for the active configuration, update data and backup.

Note

Only update data for exactly one SIMOTION device can be stored on the CF / MMC card

Procedure

1. Switch off the SIMOTION device to be updated.
2. Insert the CF / MMC card into the SIMOTION device.

3. Switch the SIMOTION device on again. The system now starts the update procedure. During the update, the **SF LED** flashes:
   - Green for SIMOTION D (0.5 Hz)
   - Red for SIMOTION C (2 Hz)
     The update procedure can take several minutes.
4. Observe the flashing of the **SF LED**:
   - When the update procedure has been completed successfully, the SF LED goes out. The device then boots up automatically in the updated configuration.
     (The **SF LED display** then depends on the respective operating state of the device.)
   - If the update procedure was not successful, the **SF LED** flickers red (10 Hz).

Note

If the firmware is being updated for SIMOTION D, depending on the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, PM340, etc.), the component firmware will also be automatically updated. This procedure can take up to several minutes and is indicated by LED displays.

An FW update on DRIVE-CLiQ components is indicated by the RDY LED flashing red/green:

- FW update running: RDY LED flashing slowly (0.5 Hz)
- FW update complete: RDY-LED flashes rapidly (2 Hz), **POWER ON** required

These flashing patterns are also displayed on the yellow RDY LED on the SIMOTION D/CX32/CX32-2, and indicate that the firmware update has been carried out on components connected to the SIMOTION D/CX32/CX32-2 or that all components have completed the firmware update.

Components that require a POWER ON after the firmware update will signal this through the rapidly flashing RDY-LED.
Switch on the 24 V supply to initialize the OFF/On (Power ON) for the respective components.

Updating a module with a SIMOTION IT file

Requirement

- The update or restore data have already been created and are available as *.zip in a SIMOTION IT file.
- There is a browser installed on the computer that is in use.
- The computer is connected via TCP/IP to the SIMOTION device to be updated.
- A memory card with sufficient memory capacity for the data to be copied must be inserted into the SIMOTION device(s) to be updated.

- It is **imperative** that updating / downgrading a SIMOTION device using SIMOTION IT takes place when the operational status is **STOP** .
- Sufficient memory capacity on the CF / MMC card: Memory capacity is required for the current configuration, the backup copy and the ZIP file.

Note

Switching off the control during data transfer may lead to loss of data on the CF/MMC card.

Procedure

1. Establish a connection via the browser to the SIMOTION device to be updated. The system sends the HTML start screen to the browser.
2. Select the **Manage Config** page and then the **Device Update** tab.
   The system sends the authentication page to the browser.
3. Log in on the **Manage Config** page using the password.
4. In the **Device Update** tab under **Send new update data**, select the stored SIMOTION IT file (the update data) using the  button.
5. Click button **Send update data** to copy the update data to the memory card in the SIMOTION device.
   The data from the existing configuration is renamed and backed up, and can be retrieved again at any time. In the event of a data restore, this data is activated again.
6. The SIMOTION device is restarted. When the device is booted, the data which has recently been imported is applied as the current configuration and activated.

Note

If the firmware is being updated for SIMOTION D, depending on the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, PM340, etc.), the component firmware will also be automatically updated. This procedure can take up to several minutes and is indicated by LED displays.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red/ green:

- FW update running: RDY-LED flashes slowly (0.5 Hz)
- FW update complete: RDY-LED flashes rapidly (2 Hz), **POWER ON** required

These flashing patterns are also displayed on the yellow RDY LED on the SIMOTION D/CX32, and indicate that the firmware update has been carried out on components connected to the SIMOTION D/CX32/CX32-2 or that all components have completed the firmware update.

Components that require a POWER ON after the firmware update will signal this through the rapidly flashing RDY-LED.
Switch on the 24 V supply to initialize the Off/On (POWER ON) for the respective components.

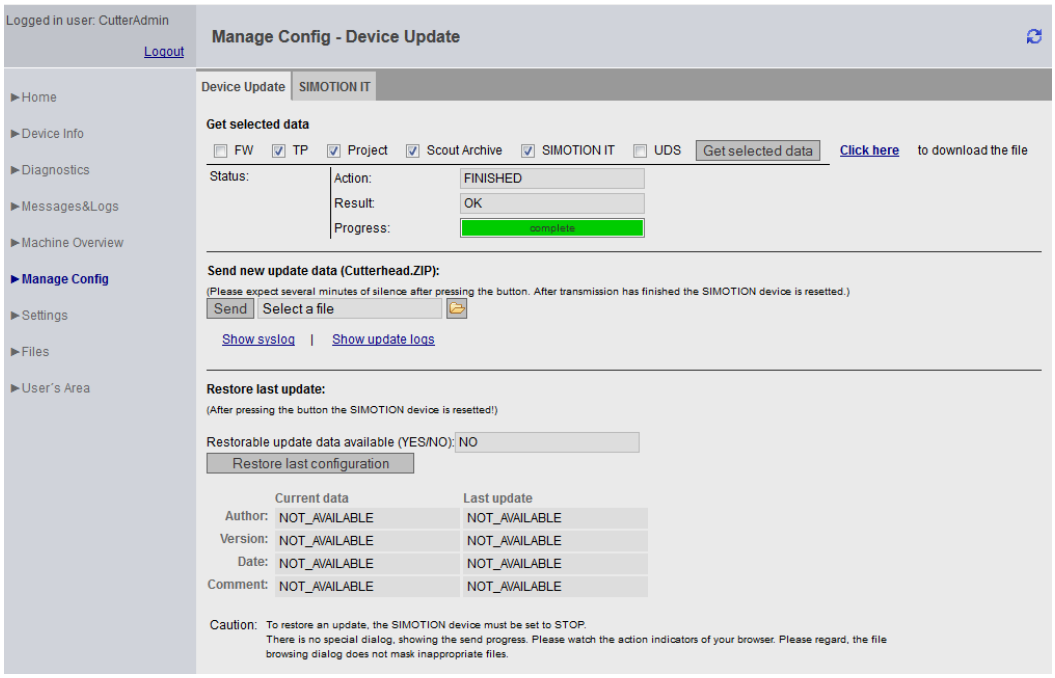


Figure 5-103 SIMOTION IT - Manage Config, tab Device Update

Saving and archiving the current configuration

Button **Get selected data**:
Select this button if you wish to save and archive the current configuration of the SIMOTION device externally. The data is saved on the PC as a *.zip file.
The configuration saved in this way can be reloaded and activated at a later time using Send update data (e.g. in the case of a memory card defect).
You can choose whether you want to archive the entire configuration or just specific subsets, e.g. firmware (FW), technology packages (TP), project and/or user data (SCOUT Archive, SIMOTION IT and UDS (Unit Data Sets)).

Further information about button **Restore last update** can be found in section: Restoring a SIMOTION device (Page 3538).

Literature

More information on SIMOTION IT can be found in the following documents:

- SIMOTION IT Diagnostics and Configuration
- SIMOTION IT, Programming and Web Services
- SIMOTION IT Virtual Machine and Servlets
- SIMOTION IT OPC UA

5.4.3.5 Possible error messages for the Device Update tool

Error messages and possible solutions

The error messages in the following list can occur when the **Device Update Tool** is being run. The error messages are in English.

Possible causes and remedies are described in the table below.

Error message	Cause	Solution
To add firmware to the updata data, the firmware must be provided by installing a **FW Support Package (FWSP)**. Currently, there is no appropriate FWSP installed. Please refer to the SCOUT documentation to get more information about the FWSP.	The firmware must be updated for at least one device. The corresponding SIMOTION FWSP (Firmware Support Package) is not available.	Please install the SIMOTION FWSP (Firmware Support Package). FWSP for versions, service packs (SPs) and hotfixes (HFs) are provided on the SIMOTION Product Support pages on the Internet at: https://support.industry.siemens.com/cs/ww/de/view/33119786 (https://support.industry.siemens.com/cs/ww/en/view/33119786)
The following selected projects are not consistent: ... In order to generate update data out of those projects, you must recompile them first.	The selected projects are not compiled in SIMOTION SCOUT. Compiled and consistent projects are required for updating.	Projects that you wish to update must first be compiled in SIMOTION SCOUT and checked for consistency. With the project open, select: • Project > Save and compile changes • Project > Check consistency
To add the selected archive data, the corresponding SCOUT project must not be opened by SCOUT. Please close the corresponding project within SCOUT or deselect the project within the SIMOTION Device Update tool.	Projects opened in SIMOTION SCOUT cannot be archived.	Close the project in SIMOTION SCOUT before you call and run the **Geräte Update Tool** .
The selected projects contain devices with identical names. To generate update data, all device names must be unique. Please deselect appropriate projects or rename the corresponding devices names to become unique. This applies to the following devices	Several projects were selected as the update data were being created and devices with identical names exist within this project.	Create the update data individually for each project or assign unique names to the devices.
The selected directory already contains data. Please select an empty directory, or delete all the data in the choosen directory.	The target directory for the update data must be empty.	Select an empty directory or delete the data from an existing one.

Error message	Cause	Solution
No Access to scout.	Error occurred while accessing the project.	If SIMOTION SCOUT is online, go offline. Contact the hotline in other cases.
<Device name> not found on card.	An attempt has been made to copy update data to a CF / MMC card. A project is already stored on the card. The existing device name does not match the device name in the update data.	If you want to overwrite the project on the card, select the option **Select additional options** in the **Force update** dialog.

5.4.4 Restoring

In the following cases it is recommended that you perform a restore on a SIMOTION project or device:

- Restoring to a previous configuration whenever an error occurs during updating
  Since a SIMOTION device update may fail, this ensures that communication capability is maintained, even after an update failure. If, for instance, the updated firmware is not recognized, the SIMOTION device may be restored to the firmware that was previously valid.
- Restoring in situations where a SIMOTION device or project has been successful but the new configuration has failed to perform as required; option of returning to the previous configuration.

5.4.4.1 Restoring a SIMOTION device

Overview

During restore procedures, it makes no difference whether the previous update was successful or unsuccessful. In both cases, the device is restored to the configuration backed up most recently.

If the update could not be carried out, the communication capability of the device is restored. The restore operation is initiated by an explicit user action.

When a restore is carried out, the SIMOTION device reverts to the configuration saved at the last update.
As long as the initial state of the SIMOTION device is not restored after an unsuccessful update attempt, the device remains in the state STOP with start inhibit. This means that, at the very least, communication capability will be ensured.
A restore procedure is necessary in order to resume work on a SIMOTION project following an unsuccessful update.

Even after a successful update, the device can be restored specifically to the data that had been backed up previously. This is the case if, for example, the new configuration has failed to perform as required and you wish to return to the previous configuration.

A backup is created as the first step in every update procedure. The device can only be restored to this backup.

Characteristics of a failed update

A project failure may manifest itself as follows:

- The SIMOTION device will not switch to RUN.
- The SIMOTION device switches to RUN but performs incorrectly.
- The new firmware has not been updated. The previous firmware state is restored, as a result of which the new project or the new technology packages cannot be loaded.
- The causes of this behavior are displayed in the diagnostics buffer entries. The stated information can be determined using the diagnostic pages in SIMOTION IT, for example.
- The SIMOTION device displays a request for an overall reset.

Requirements for performing a restore

A backup of the previous configuration was created during the update procedure which ensured that a backup of the data exists.

The configuration includes firmware (FW), technology package(s) (TP), project(s), and any user data which may exist.

Triggering a restore

A restore operation is triggered by saving the restore archive to the memory card, by operator control actions at the SIMOTION device or by use of the button **Restore last update** in the **Manage Config** dialog (SIMOTION IT).

In the case of SIMOTION D, restore can be triggered by means of switch settings on the device. Alternatively, restore can be initiated in all SIMOTION devices by transferring the restore archive. See Particular aspects of restoring SIMOTION devices (Page 3543).

Backup data

The backed up data from a SIMOTION device is retained as long as the backup exists, i.e. until it is overwritten by another update. Should errors occur during the update procedure, the device can be restored to the backup.

SIMOTION data behavior during restoring

- Current retain data
  The retain data will be restored only if the firmware is restored or if the data structures are not compatible. Behavior is described under: Behavior of the retain data during updating (Page 3521).
- Current diagnostics buffer
  The content at the time of backup is activated again.

You can find additional information in the section Particular aspects of restoring SIMOTION devices (Page 3543)

Behavior of the SINAMICS retain data during restoring

After a firmware restore has been carried out, the SINAMICS retain data is in the same state as when the data was backed up.

5.4.4.2 SIMOTION device status during restoring

Status of SIMOTION devices

The table below provides an overview of the possible statuses, the corresponding LEDs, and their meanings for SIMOTION devices during the restore process.

Table 5-53 Overview of SIMOTION device status during restoring

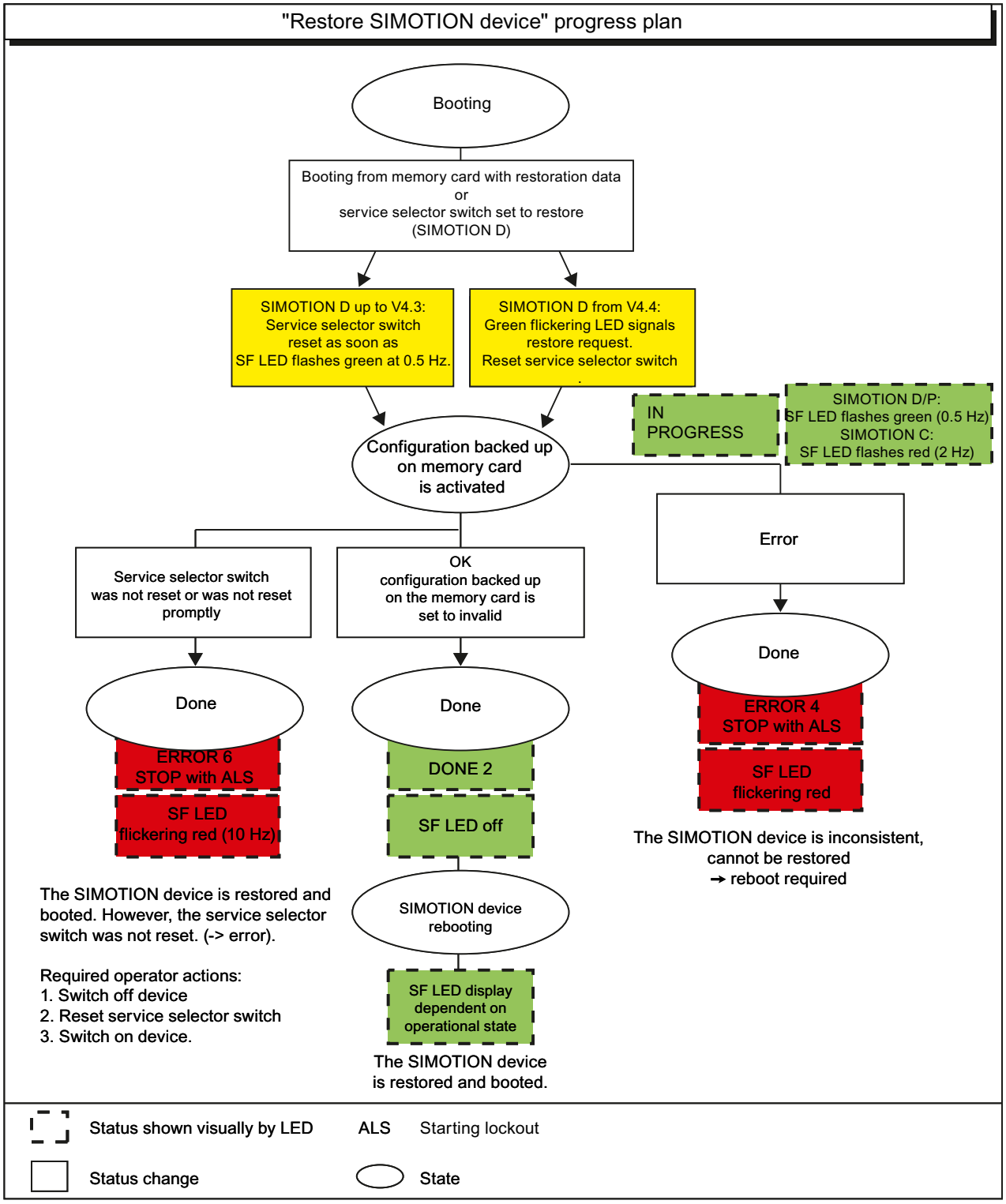
LED display	Status	Meaning	Description
SF LED flickers green (from V4.4)	IN PROGRESS	Restore requested.	The flickering green SF LED indicates that a restore is requested. The restore procedure is started by resetting the service selector switch.
SIMOTION D/P: SF LED flashes green at 0.5 Hz **SIMOTION C**: SF LED flashes red at 2 Hz	IN PROGRESS	Restore active.	Restore in progress. **Note** Do not switch off the SIMOTION device.
SF LED flickers red (10 Hz)	ERROR 4	Restore error.	SIMOTION device is booting up in **STOP with start inhibit**. After OFF/ON, the device boots up in STOP with start inhibit.
SF LED off [1]	DONE 2	Restore successfully completed.	SIMOTION device boots up with the restored configuration
SF LED flickers red (10 Hz)	ERROR 6	Error **SIMOTION D only:** Service selector switch is still set at Restore.	SIMOTION device boots up in **STOP with start inhibit**. Switch off the device, reset the service selector switch and switch the device back on. If the restore operation was otherwise successful, the devices boots up with the restored configuration.

[1] The restore has been successfully completed when the SF LED goes out; the device then boots up automatically in the restored configuration.
The SF LED display depends on the relevant operating state of the device.

The restore status is shown not only via the LEDs, but also in the device diagnostics on the tabs "Diagnostics buffer" and Syslog.

Flow chart: Restoring a SIMOTION device

The chart below shows how a SIMOTION device restore may proceed once the restore archive has been transferred to the memory card of the SIMOTION device.
The status displays appear in different colors by way of analogy.


"Restore SIMOTION device" progress plan
Booting
Booting from memory card with restoration data or service selector switch set to restore (SIMOTION D)
SIMOTION D up to V4.3: Service selector switch reset as soon as SF LED flashes green at 0.5 Hz.
SIMOTION D from V4.4: Green flickering LED signals restore request. Reset service selector switch
IN PROGRESS
SIMOTION D/P: SF LED flashes green (0.5 Hz) SIMOTION C: SF LED flashes red (2 Hz)
Configuration backed up on memory card is activated
Service selector switch was not reset or was not reset promptly
OK configuration backed up on the memory card is set to invalid
Error
Done
ERROR 6 STOP with ALS
SF LED flickering red (10 Hz)
Done
DONE 2
SF LED off
Done
ERROR 4 STOP with ALS
SF LED flickering red
The SIMOTION device is restored and booted. However, the service selector switch was not reset. (-> error).
Required operator actions:
1. Switch off device
2. Reset service selector switch
3. Switch on device.
SIMOTION device rebooting
SF LED display dependent on operational state
The SIMOTION device is restored and booted.
The SIMOTION device is inconsistent, cannot be restored
→ reboot required
Status shown visually by LED
ALS Starting lockout
Status change
State

Note

Restore request from Version V4.4

From Version V4.4 the restore request is indicated by a green flickering SF LED.

The restore procedure starts once the service selector switch is reset.

In the case of SIMOTION < V4.4, the restoration starts once the module is switched on. Here, the service selector switch must be reset immediately once the flash code is displayed (SF LED flashes green at 0.5 Hz).

If the service selector switch is not reset or not reset promptly, this will lead to the error status "Service selector switch is still at restore" (SF LED flickers red).

5.4.4.3 Particular aspects of restoring SIMOTION devices

Precondition

During restoring procedures, it makes no difference whether the previous update was successful or unsuccessful. The procedure is identical for every SIMOTION device.

A distinction is made between SIMOTION devices where a restore can be requested via an operator control. In the case of SIMOTION devices with Ethernet / PROFINET interfaces, the user can also request a restoration using SIMOTION IT. For all SIMOTION devices, restore data can be created as a restore archive on an operator-guided basis via the **Device Update Tool**. These can then be saved to the memory card of the SIMOTION device.

Table 5-54 Assignment of restore options to SIMOTION devices

SIMOTION device	Restore options	Method
C2xx	Create the restore archive by means of the Device Update Tool.	Machine manufacturer
D4x5-2	**Preferred solution**: Via service selector switch on the device **Alternative**: Create the restore archive by means of the Device Update Tool.	Machine operator
D410-2	**Preferred solution**: Via service selector switch on the device **Alternative**: Create the restore archive by means of the Device Update Tool.	Machine operator
P320-4	Create the restore archive by means of the Device Update Tool.	Machine manufacturer

Restoration using SIMOTION IT is essentially possible for all SIMOTION devices with Ethernet interfaces for all restore scenarios and can be carried out by the machine operator.

Restoring SIMOTION D410-2/D4x5-2 using the operating element on the device

The user at the machine operator's premises can restore using the operating element independently of the machine manufacturer because no SIMOTION SCOUT is required.

1. Switch off the SIMOTION D410-2/D4x5-2 device.
2. Set the service selector switch to position **B** (in the case of D410-2: SVC, in the case of D4x5-2: SVC/NCK).
3. Switch the SIMOTION D410-2/D4x5-2 device back on again.
   The flickering green SF LED indicates that a restore is requested.
4. Reset the service selector switch to position **0**.
   The system restores the data saved during the update. The data from the update procedure will be deleted.
   The restore will be signaled by a green flashing SF LED (0.5 Hz) and this may take several minutes.
   After a successful restore operation, the module will start up automatically. (SF LED display will then depend on the respective operating status of the device).
   If the restore procedure was not successful, the SF LED will flicker red.

Note

Starting the restore procedure (SIMOTION < V4.4)

In the case of SIMOTION < V4.4, the restore procedure starts when the module is switched on (see Step 3).
The service selector switch must be turned to position **0** immediately the flash code is displayed (SF LED flashes green at 0.5 Hz).

If the service selector switch is not reset to position **0** or not reset there promptly, this will lead to the error status "Service selector switch is still at restore" (SF LED flickers red).

In this case, switch off the SIMOTION D Control Unit, reset the service selector switch and switch the Control Unit back on again. If the restore was otherwise successful, the Control Unit will start up with the restored configuration.

Restoring firmware for SIMOTION D

Note

If the firmware is being restored for SIMOTION D, depending on the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, PM340, etc.), the component firmware will also be automatically restored. This procedure can take up to several minutes and is indicated by LED displays.

A FW update on DRIVE-CLiQ components is signaled by red-green flashing of the LED RDY:

- FW update running: LED **RDY** flashes slowly (0.5 Hz)
- FW update complete: LED **RDY** flashes rapidly (2 Hz), **POWER ON** required

This flash pattern is additionally displayed at the yellow LED **RDY** of the SIMOTION D/CX32/CX32-2 and signals that components connected to the SIMOTION D/CX32/CX32-2 are carrying out a FW update or that all components have completed the FW update.

After the firmware update has been completed on all components, a POWER ON must be carried out for all restored components.

Restoring via the restore archive

A restore archive is created at the machine manufacturer's site using the Device Update Tool function in SIMOTION SCOUT.

When restoring, you are guided by the Device Update Tool (in exactly the same way as with updating).

1. Select **Project > Start Device Update Tool** in the open SIMOTION project.
2. The **Device Update Tool** opens with the start screen **Welcome to the SIMOTION Device Update tool**.
   In this screen you can decide whether data is to be created for updating or restoring a SIMOTION device.
   Select **Create restore trigger** since you wish to create data for restoring a device.
3. The subsequent steps in this procedure are identical to those involved in updating.
   Refer to: Creating update data with the Device Update Tool (Page 3492)

Restoring using SIMOTION IT

The user at the machine operator's premises can perform a restore using the SIMOTION IT web server independently of the machine manufacturer because no SIMOTION SCOUT is required.

To do this, select **Manage Config** , tab Device Update and button **Restore last update** in order to activate the backup in the device.

The system restores the data backed up during the update. The data from the update will be deleted. Once this is complete, the SIMOTION device restarts. After rebooting, the configuration is restored to its pre-update status.

See also: Updating a module with SIMOTION IT (Page 3534)

5.4.5 Appendix A

5.4.5.1 Restoring the original status of a USB memory stick

Requirement

A USB memory stick has to be formatted once before it can be used as an update medium for the first time. A bootable update medium is created during formatting. The USB stick's memory is limited to a fixed value of 450 MB, irrespective of how much memory capacity is actually available. Various partitions, which are required for updating, are created.

To restore the USB memory stick to its original status, it is not enough to simply format it as the partitions have to be deleted first.

Note

A USB memory stick formatted using the **Device Update Tool** can **only** be restored under **Windows XP** and can then be used again.

Procedure

Before formatting the USB memory stick, the existing partitions must be deleted and a partition encompassing the entire USB memory stick must be created.

The procedure is as follows:

1. Insert the USB memory stick that has been formatted as an update medium into a USB port on the computer.
2. Right-click the **My Computer** icon on your desktop and select **Manage** from the context menu.
   Alternatively, you can also open the Windows Explorer. Right-click **My Computer** in the Start menu window and select **Manage** from the context menu.
   The **Computer Management** window opens.

3. Scroll down the Computer Management window until you reach **Disk Management**.
   In this example the USB memory stick appears under drive letter (F:). The total memory capacity of the USB memory stick is shown at the top right-hand side of the window under Capacity.
   The different partitions and the memory capacity that is currently available for use appear in the bottom right of the window under **Disk 2**.
   The USB memory stick is divided into three partitions:

   – VOLUME (F:) - 456 MB FAT32 - Error-free:
     The area on the left is the partition which is visible in Windows; this contains the update data.

   – 30 MB - Error-free:
     The area in the middle is the partition containing the LINUX system.

   – 1.45 GB - Not allocated:
     The area on the right is the partition containing the part of the USB memory stick which is of no relevance to the update data. The size varies depending on the memory capacity of the USB memory stick. This partition is not allocated and cannot, therefore, be read by Windows either.

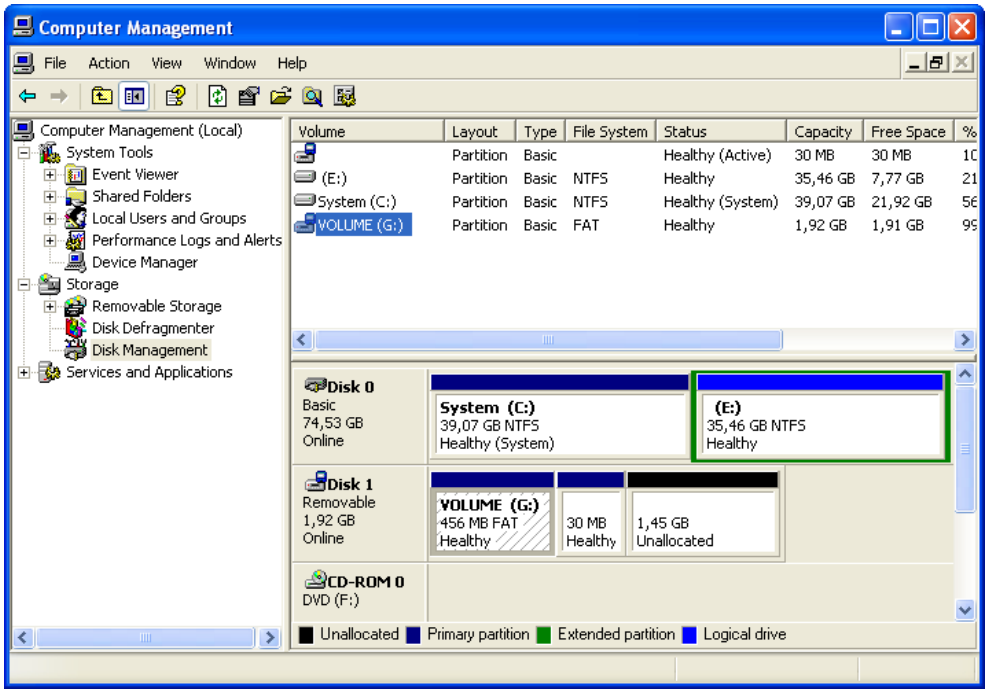


Figure 5-104 Computer Management window

4. In order to create a complete partition again, the LINUX partition which is not visible in Windows must be deleted.
   Right-click the middle partition (in this case 30 MB) and select **Delete Partition...** from the context menu.

5. Click **Yes** in the dialog box that appears to confirm that you wish to delete the partition.

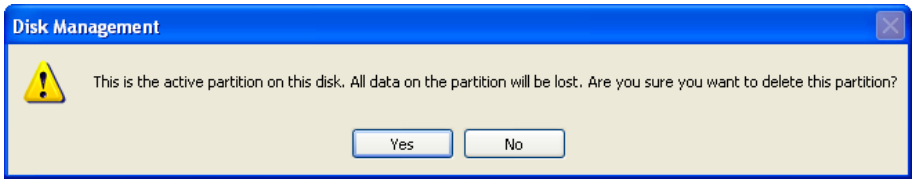


Figure 5-105 Disk Management

6. There is now only a single **unallocated** partition available under **Disk 2** in the Computer Management window.
Selecting the middle partition also automatically deletes the partition on the left-hand side containing the update data.

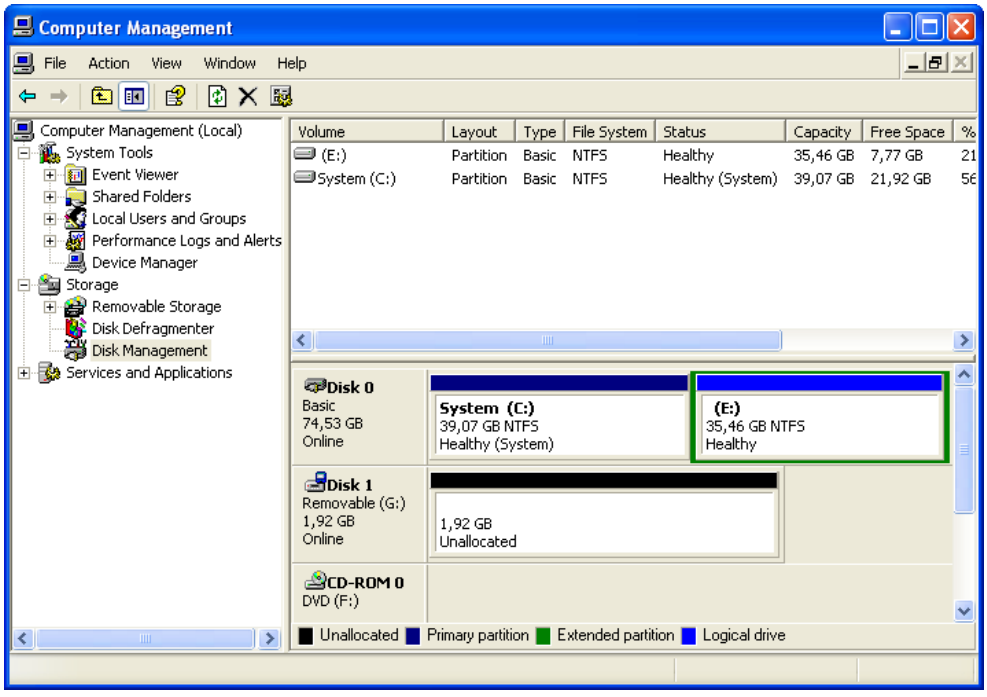


Figure 5-106 Computer Management window

In order to create a new partition, right-click the **unallocated** partition and select **Activate new partition** in the context menu.

7. The **New Partition Wizard** appears.
   Run the wizard, as indicated in the steps below.
   Click **Next** to proceed to the next window.

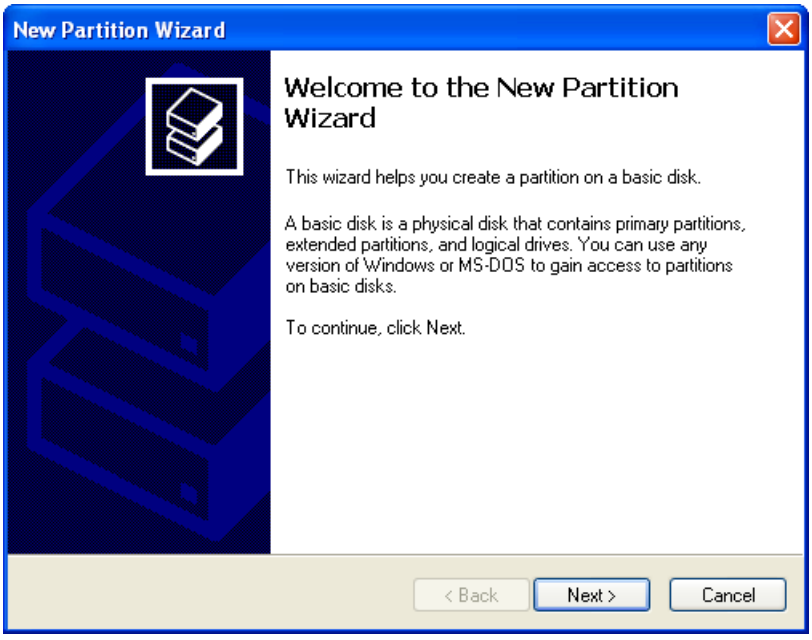


Figure 5-107 "New Partition Wizard" start screen

8. Confirm **Primary partition** as the partition type.

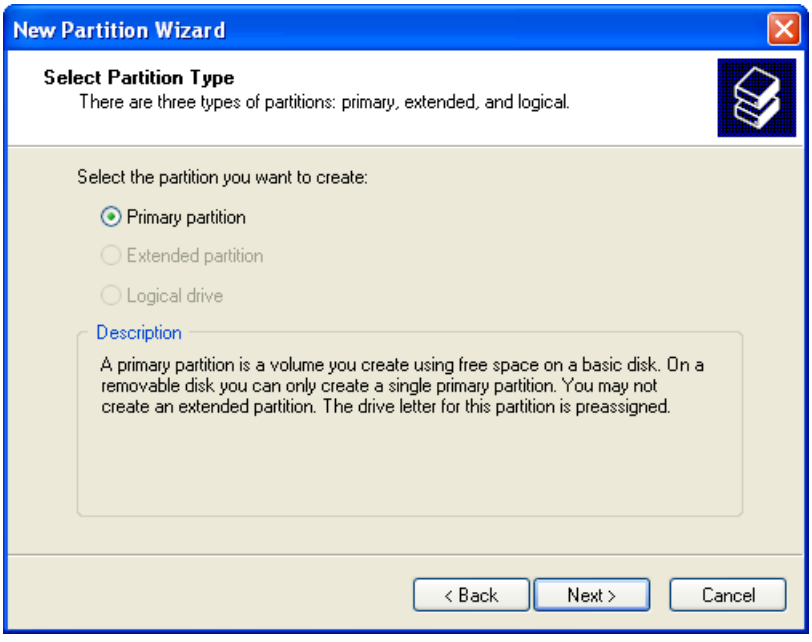


Figure 5-108 Wizard: Defining the partition type

9. Confirm the partition size.

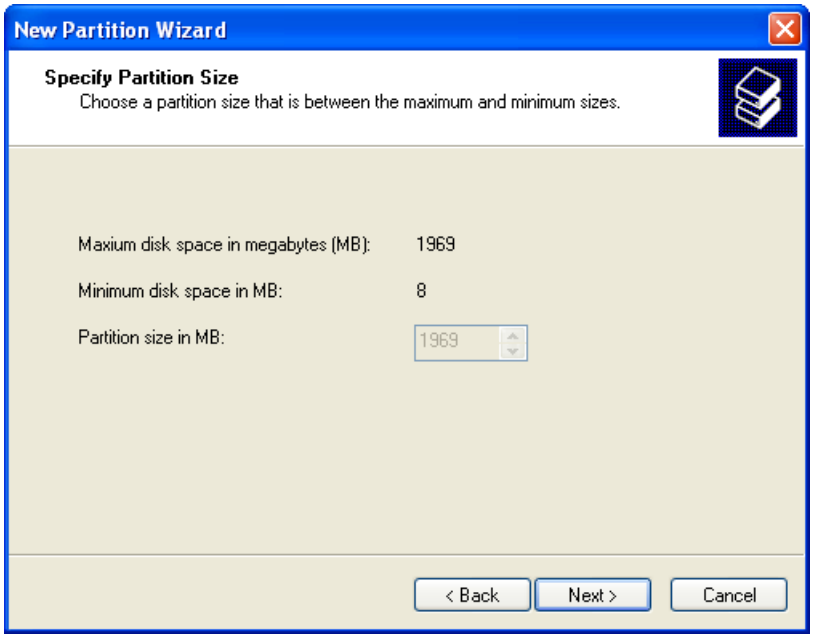


Figure 5-109 Wizard: Defining the partition size

10. Assign the drive into which the USB memory stick is to be inserted.

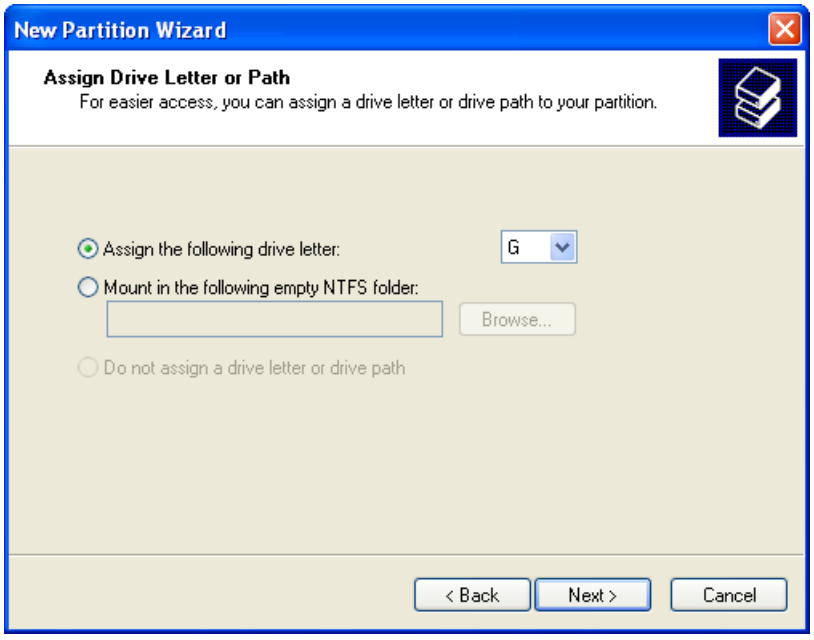


Figure 5-110 Wizard: Assign the drive letter or path.

11. Choose the following settings in order to format the partition:
   - File system to be used: FAT
   - Allocation unit size: Standard
   - Volume identifier: Volume

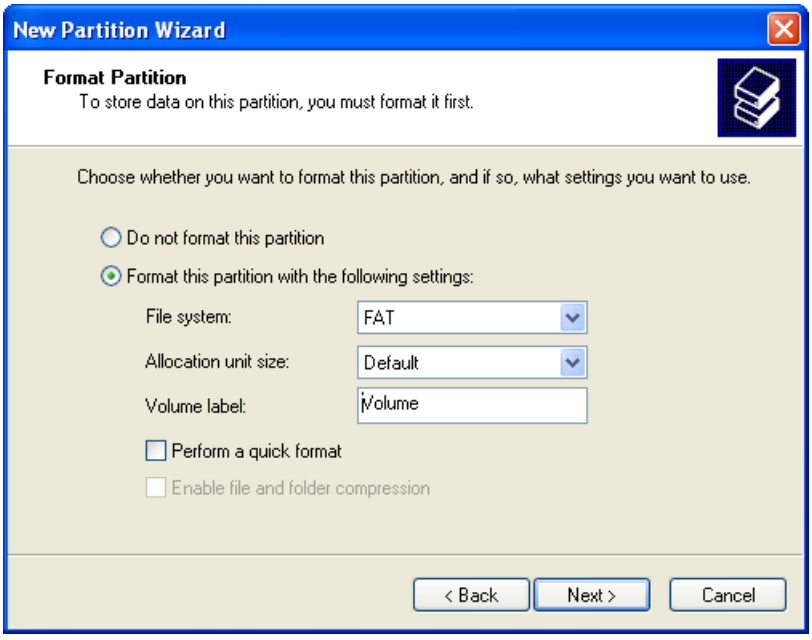


Figure 5-111 Wizard: Format a partition

12. The settings selected are displayed in the closing window for the wizard.
    Click **Finish** to close the wizard.

13. The changed partition appears in the **Computer Management** window under **Disk 2**. The USB memory stick is restored to its original status. The entire memory capacity will now be available again in a partition and displayed in Windows.

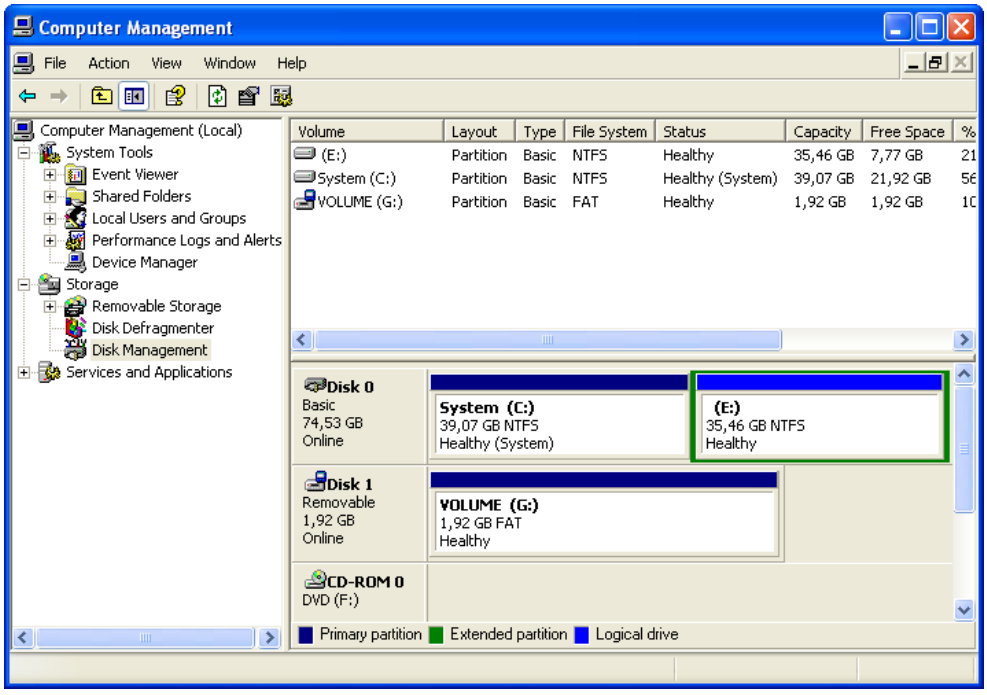


Figure 5-112 Computer Management - View of the newly created partition

SIMOTION IT 6

6.1 SIMOTION IT Diagnostics and Configuration

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

6.1.1 Fundamental safety instructions

6.1.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

6.1.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

6.1.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

6.1.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

6.1.2 Introduction

6.1.2.1 Overview of SIMOTION IT

Overview of SIMOTION IT manuals

The "SIMOTION IT Ethernet-based HMI and diagnostic functions" are described in four manuals (IT = Information Technology):

- **SIMOTION IT Diagnostics and Configuration**
  This present manual describes the direct diagnosis of SIMOTION devices. Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.
- **SIMOTION IT Programming and Web Services**
  This manual describes the creation of user-defined Web pages and access to the diagnostic functions via the two Web services provided by SIMOTION IT.
  A Web service enables users to create their own client applications in any programming language. These applications then communicate with the SIMOTION device using Web technologies. The SOAP (Simple Object Access Protocol) communication protocol is used for transferring commands.
  The manual includes information on programming such clients, as well as a description of the SIMOTION IT Web services (OPC XML-DA, Trace via SOAP TVS) via which data and operating states of the controller can be accessed and the variable trace functions can be used.
  See the SIMOTION IT Programming and Web Services manual.
- **SIMOTION IT Virtual Machine and Servlets**
  This manual describes the Java-based function packages. The Jamaica Virtual Machine (JamaicaVM) is a runtime environment for Java applications on the SIMOTION device. It is an implementation of the "Java Virtual Machine Specification."
  The Servlets section of the manual describes the use of servlets in a SIMOTION device.
  See the SIMOTION IT Virtual Machine and Servlets manual.
- **SIMOTION IT OPC UA**
  The manual describes access to SIMOTION devices via OPC UA.

See also

PDF in the Internet: SIMOTION IT Programming and Web Services (https://support.industry.siemens.com/cs/ww/en/view/109476528)

PDF in the Internet: SIMOTION IT Virtual Machine and Services (https://support.industry.siemens.com/cs/ww/en/view/109476529)

6.1.2.2 Schematic diagram of the function packages in the SIMOTION device

Access to a device with SIMOTION IT

SIMOTION IT allows HTTP/S access to a device by several means, which are shown in the diagram.

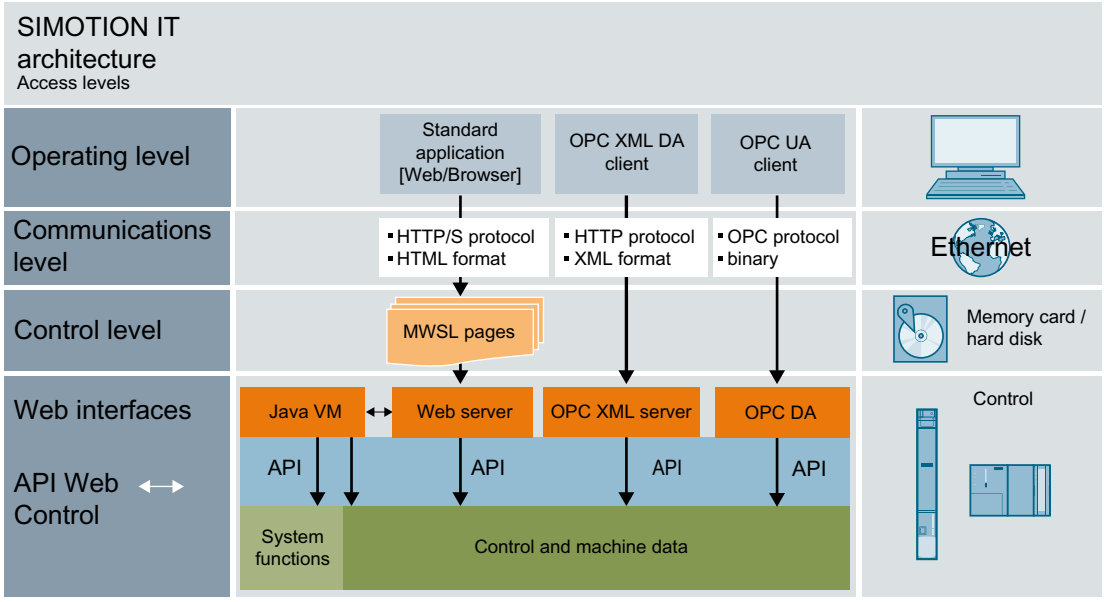


Figure 6-1 SIMOTION IT architecture of the HTTP/S access levels

The Web server of the SIMOTION controller is called Miniweb-Server.

Representation of the function packages

The following figure is a schematic diagram of the function packages in the SIMOTION device.

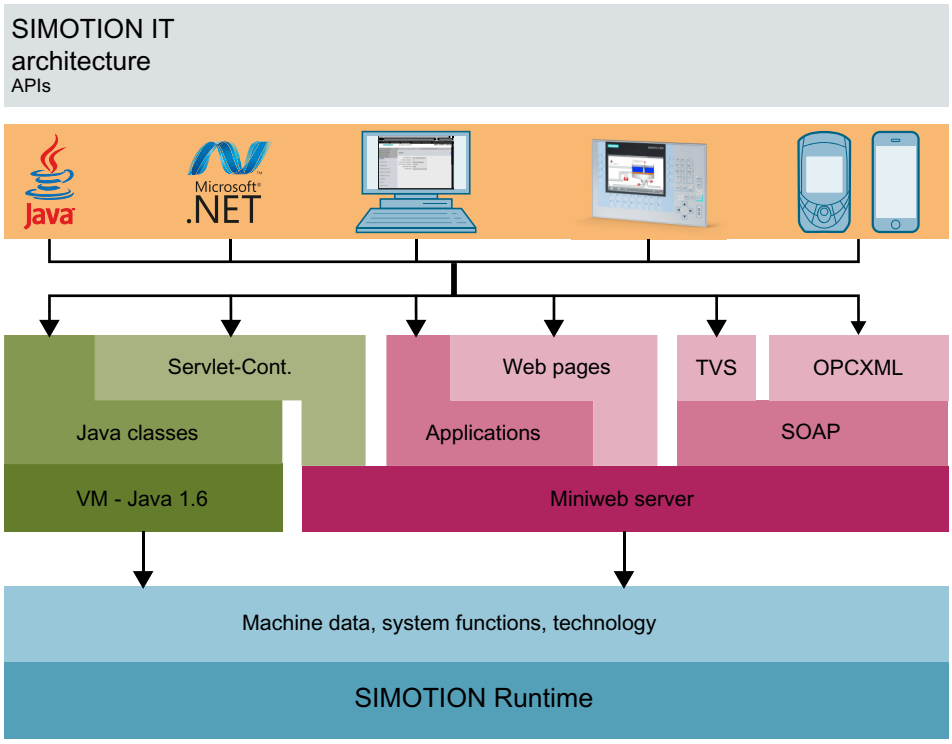


Figure 6-2 SIMOTION IT architecture of the APIs

As further access possibility to the SIMOTION controller, OPC UA access has been added as of version 4.5.

6.1.2.3 Form of delivery

Form of delivery

"SIMOTION IT Ethernet-based HMI and Diagnostic Functions" are included in the controller firmware.

Note

The functionality must be activated in the SIMOTION SCOUT project in the hardware configuration of the controller. You can activate the Web server services in the hardware configuration on the "Ethernet extended / Web server" tab in the object properties of the controller. Further information about project settings is contained in the "Activating communications services in SCOUT Classic" section.

The "Activating communications services in SCOUT TIA" section describes how the Web server can be activated in the TIA Portal.

Documentation, tools, examples, and configuration files

The documentation, tools, examples, configuration files and other supplementary features are available on the "Documentation, Utilities & Applications" DVD.

Runtime licenses before Version 4.2

The older versions require an OPC XML-DA single-user license for access to the Watch page, for example.

When any of these pages is opened, the following is displayed:

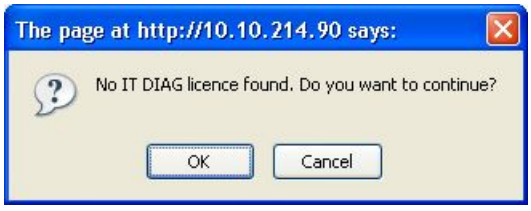


Figure 6-3 Warning - Missing license

Clicking the **OK** button opens the requested page. You can thus continue even without a license. However, an entry is made in the diagnostic buffer and the error LED on the controller starts to flash.

6.1.2.4 Possible applications

Standard information

Application of diagnostic pages

The Web pages from SIMOTION IT provide information on a SIMOTION device. The information is accessed via the Web browser and the Ethernet.

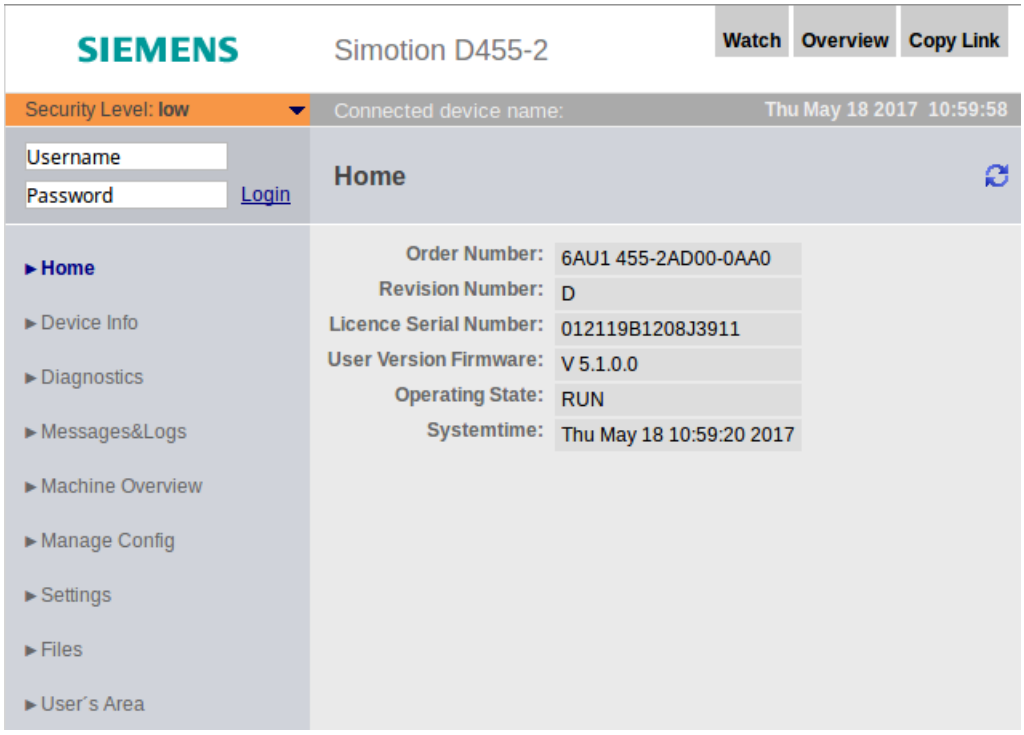


Figure 6-4 Home

The SIMOTION device is connected to the local Ethernet for this purpose. Diagnostic pages can then be accessed from any computer in the network using the IP address of the device.

HTTPS connections are also supported. The HTTPS connection should be used where possible because, with HTTP, the login and passwords are not encrypted for transmission.

The use of SIMOTION IT standard pages does not require a special installation. The device is already appropriately set up.

See also

Security concept (Page 3569)

Log-in administration (Page 3658)

TLS/SSL certificates (Page 3574)

Secure Socket Layer (Page 3702)

User-defined information

Displaying information in user-defined pages

In addition to displaying the standard pages, SIMOTION IT allows you to create your own Web pages. The *SIMOTION IT Programming and Web Services* manual describes the methods for creating your own Web pages.

With the aid of a JavaScript library, device data can be queried and displayed in a Web page.

A further option is the use of the MiniWeb Server Language (MWSL). This is a language based on an ECMA script that is executed on the server side.

The "variable providers" can be used to read and write the following information on a Web page:

- System variables of the SIMOTION device
- System variables and configuration data of the technology objects
- Global unit variables
- Drive parameters
- I/O variables
- Global device variables
- Connection monitoring

User-defined pages provide numerous options for displaying device information.

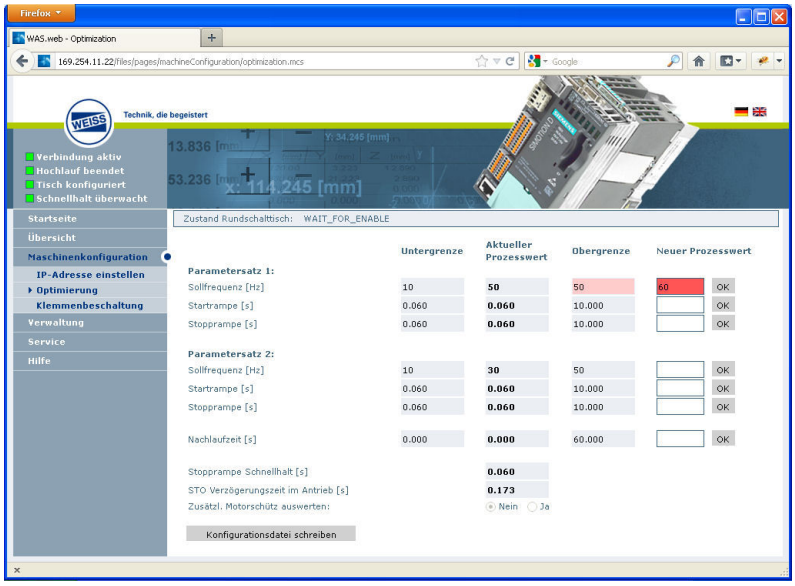


Figure 6-5 Example of a user-defined SIMOTION IT Web page of WEISS GmbH

MWSL

The MWSL is executed on the server side. It enables the creation of dynamic HTML code on Web pages. You can also use the MWSL if the created pages are displayed on devices that do not support JavaScript. Variable functions can be executed faster and more directly (closer to the system) than when using JavaScript.

Note, however, that the MWSL code evaluation requires server resources. For loaded controllers, this code evaluation can take a long time, and delay other Web processes and Web queries.

JavaScript

SIMOTION IT supports you in creating dynamic and flexible Web pages thanks to an extensive JavaScript library. Unlike MWSL, JavaScript runs in the browser. The use of JavaScript relieves the load on the controller and provides considerably more options than the MWSL. For display purposes, however, a modern browser with corresponding JavaScript support is required; this is something that cannot be guaranteed in all automation environments.

6.1.2.5 New features

Overview of the innovations

Version 5.2

- Cross Site Request Forgery (CSFR) Protection. Prevention of non-authorized HTTP requests.Cross Site Request Forgery (CSFR) (Page 3673)
  The activation of the CSFR protection has major effects and requires customizations for some accesses to the Web server. CSFR protection affects:
  - User-created Web pages
  - OpcXml DA in Web pages and external clients
  - POST/GET requests in Java applications and servlets
- Service selector switch position "8" receives a time-controlled disconnection. Resetting the security level from normal to low (Page 3569)

Version 5.1

- Encrypted connections are possible only as of TLS 1.2. Encryption (Page 3574)
- For security reasons, access to the file system is possible only with Javascript. The **Files** Web page uses a new File Manager for displaying the file system.

Version 4.5

- OPC UA server. The OPC UA server is described in the SIMOTION IT OPC UA manual.
- Separate authentication for read and write accesses for OPC XML.
- Revised WebCfg.xml.

Version 4.4

- Security concept (Security Level)
- Revised login administration (Page 3658). Separate storage of user data in file UserDataBase.xml. New page Users & Passwords (Page 3630) for editing user data.
- New version of the MiniWeb
- Output of messages by the messaging system on the SIMOTION IT pages without an adverse impact on processing.
- New display formats for floating-point numbers in the Watch (Page 3589) table
- New variables provider ITDiag
- Traces (WTRC files) can now be loaded and displayed in SIMOTION SCOUT.

6.1.3 Commissioning

6.1.3.1 Hardware and software requirements

Hardware requirements

- SIMOTION device
- Web-enabled device such as PC, notebook, smartphone with a minimum resolution of 320x240 pixels.

Software requirements

- Browser: Firefox as of version 3, Microsoft Internet Explorer as of version 11 and Microsoft Edge.

6.1.3.2 Activating communications services in SCOUT Classic

Activating the SIMOTION IT Web server in HW Config

The Web server of the SIMOTION controller can be activated in the HW Config. To do that, navigate via **Device object properties** to the **Ethernet extended / Web server** tab.

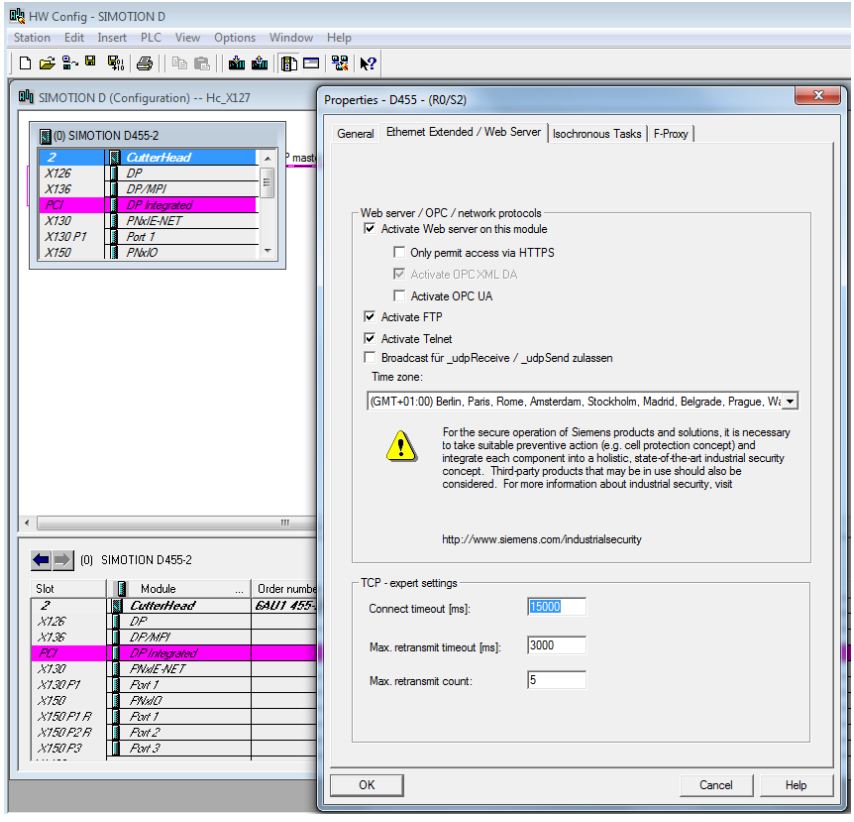


Figure 6-6 HW Config settings

The screenshot shows the default Web server settings. When you deactivate a service, the corresponding communications port is closed. If no service is activated, the Web server of the controller is also deactivated.

The Web server is addressed via HTTP/S. FTP and Telnet are connected only to the user administration. OPC UA activates the OPC UA access to the Web server.

Service	Port
HTTP (Browser, OPC XML)	Setting in the WebCfg.xml – default 80
HTTPS (Browser, OPC XML)	Setting in the WebCfg.xml – default 443
OPC UA	Setting in WebCfg.xml – in the URL attribute of the <END-POINTDESCRIPTION> tag.
FTP	21
Telnet	23

Setting the time zone

The applicable time zone of the Web server can be set in two ways. One possible way is the setting shown in the screenshot made via the drop-down list in the HW Config dialog.

The second way is to make the setting in the Web page **Settings**. In this case, the value from the HW Config dialog will be ignored.

Calling up HW Config from SCOUT

In SCOUT, you can access the settings in HW Config via **Gerät > Eigenschaften > Einstellungen** with the link **Web server settings in HW Config**.

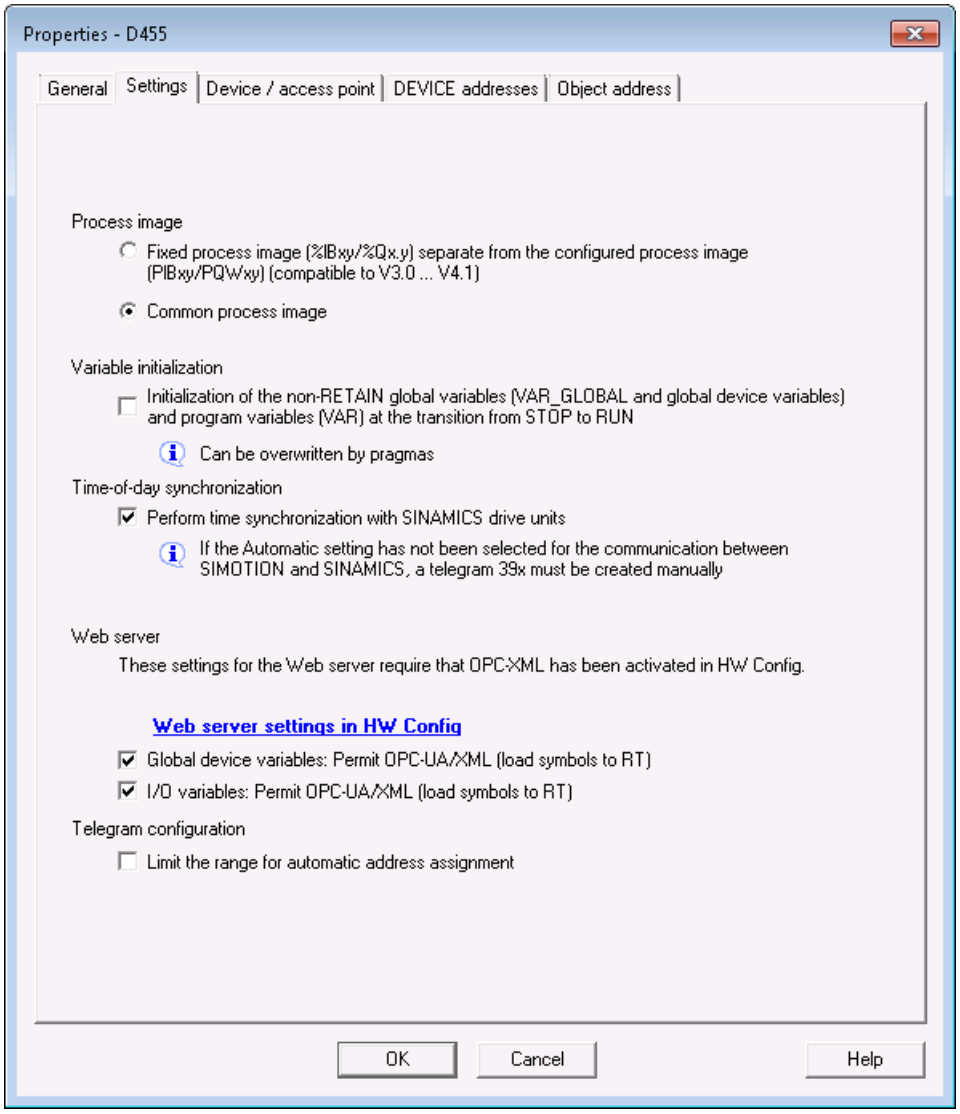


Figure 6-7 SCOUT connection to HW Config

See also

Settings (Page 3639)

6.1.3.3 Activating communications services in SCOUT TIA

Procedure

To activate the Web server in the TIA Portal, proceed as follows:

1. Select the SIMOTION device in the network view / device view.
2. In the Inspector window, select the "Properties" tab and click the "General" tab.
3. Select "Web server / OPC / Network protocols".
   The Web server is deactivated in the basic setting. You must activate the relevant checkboxes so that the CPU displays the Web pages.

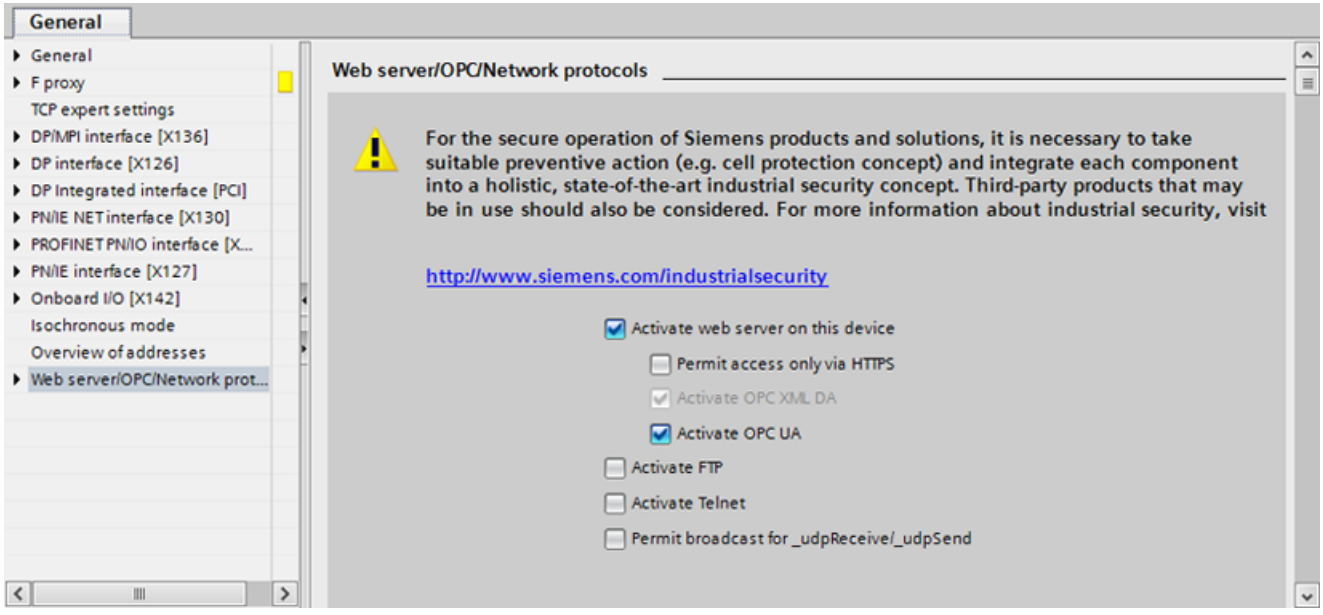


Figure 6-8 Activating the Web server

The Web server is addressed via HTTP/S. FTP and Telnet are connected only to the user administration. OPC UA activates the OPC UA access to the Web server.

Displaying SIMOTION websites in TIA Portal WinCC CA

To use the WebBrowserControl (HTML web browser), activate "ScrollViewer" mode in TIA Portal in the runtime settings of the HMI device. To do this, follow these steps:
Under "Runtime settings > Screens > Scrolling in Controls", select the "ScrollViewer" scrolling mode.

Note

Only the HTML web browser based on the WebKit engine works in connection with SIMOTION Web server. Error messages may occur in the HTML web browser with ActiveX support.

- Navigate to the HTML web browser type and choose the setting "Web browser based on a WebKit engine".

Calling the HW Config from SIMOTION SCOUT TIA

You can switch directly to the appropriate tab of the Inspector window in the TIA Portal via SIMOTION SCOUT TIA.

Proceed as follows:

1. Select the SIMOTION device in the project navigator.
2. Select the "Properties" entry in the context menu.
   The "Properties" dialog box opens.
3. Switch to the "Settings" tab and click the "Web server settings in HW Config" link.
   You can now make the web server settings in the TIA Portal.

See also

Activating communications services in SCOUT Classic (Page 3564)

6.1.3.4 Configuring the SIMOTION device interface

Configuration of the Ethernet interface

SIMOTION IT can be accessed via any Ethernet interface used with SIMOTION, including the PROFINET IO interface.

To establish a connection between the standard diagnostics pages and a SIMOTION device via a browser, the following steps for configuring the Ethernet interface must be performed:

Table 6-1 Configuring the interface

Step	Procedure
1	The functionality must be activated in the SIMOTION SCOUT project in the hardware configuration of the CPU. You can activate the relevant services in the hardware configuration on the "Ethernet extended / Web server" tab in the object properties of the CPU. As of V4.1.2, HTTP/S, FTP and Telnet are activated in the delivery state. In TIA Portal, the services are disabled by default.
2	SIMOTION IT uses a user database called UserDataBase.xml to control access to the device. If no user database is found on the device, an empty user database is created when the controller starts up. You cannot login until a user has been created. See Log-in administration (Page 3658)
3	To display the standard diagnostics pages in the browser, you must enter the IP address of the SIMOTION device, e.g. http://169.254.11.22. The preset IP addresses are documented in the manuals for the respective controls. This factory setting can be changed in the HW Config and then loaded to the SIMOTION device.

Note

This requires suitable protective measures (e.g. network segmentation for IT security) to ensure safe system operation. You can find more information on Industrial Security on the Internet at:

www.siemens.de/industrialsecurity.

6.1.3.5 Security concept

Safety concept HTTP/S, FTP and Telnet access

As of version V4.4, access to the SIMOTION IT Web server is protected by a multi-level security concept.

The security state of the Web server is indicated by the Security Level on the Web page. This Security Level can have three different levels: Low, Normal, High.

Security Level Low

The device is supplied with an empty user database. No project exists yet on the device. The security level is low to allow configuration of the device.

Note

As of V5.3 SP1 HF4, the Telnet access has been changed by default.

1. Telnet remains activated after loading the firmware.
2. When a new project is created, Telnet is deactivated in the hardware configuration by default. This means that Telnet is deactivated on the CPU after first loading a new project.

This change only affects SIMOTION SCOUT. In SCOUT TIA, Telnet is deactivated by default as previously.

- In this state, access to the Web server as an anonymous user is possible. Functions, such as project and firmware update or OPC XML, are consequently available.
- FTP access is open.
- New users can be entered in the empty user database.

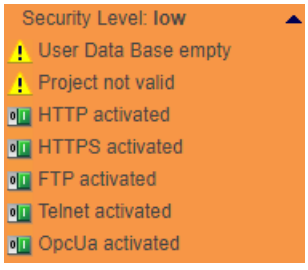


Figure 6-9 Security Level Low

In this state, series commissioning is possible via the Web server.

NOTICE
Protecting the device Security Level Low security level should only be used for commissioning and service as otherwise the device is not adequately access protected.

Security Level Normal

The controller has a user database. A project exists on the controller and HTTP, HTTPS, FTP, and Telnet are activated in HW Config.

- User password authentication is mandatory for access to Web pages with sensitive content (e.g. firmware update, watch table, ...), FTP and Telnet.

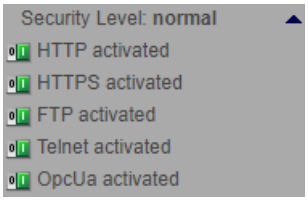


Figure 6-10 Security Level Normal

Security Level High

High security with maximum access protection:

- HTTP, HTTPS, FTP and Telnet were activated via the project in HW Config. Access to the Ethernet via the various ports of the services is then no longer possible. The Web server cannot be used.

Authentication

Many different access scenarios are made possible by the various security levels.

Table 6-2 Access control Security Level Low

	HTTP/S Web pages without authorization	HTTP/S Web pages with authorization	FTP	Telnet
No project exists on the controller and service selector switch not in position "8"				
No user in UserDataBase.xml	Access permitted	Access permitted	Access permitted	Access permitted
Whether or not the project exists on the controller and service selector switch in position "8"				
No user in UserDataBase.xml	Access permitted	Access permitted	Access permitted	Access permitted
User exists in UserDataBase.xml	Access permitted	Access permitted	Access permitted	Access permitted

Table 6-3 Access control Security Level Normal

	HTTP/S Web pages without authorization	HTTP/S Web pages with authorization	FTP	Telnet
No project exists on the controller and service selector switch not in position "8"				
User exists in UserDataBase.xml	Access permitted	Password	Password	Password
Project exists on the controller, the appropriate checkboxes are activated in HW Config and service selector switch not in position "8" If a checkbox has not been activated in HW Config, access to the port of the respective service is denied.				
No user in UserDataBase.xml	Access permitted	Password*	Password*	Password*
User exists in UserDataBase.xml	Access permitted	Password	Password	Password

Password = access only after authentication

Password* = login is not possible because there is no entry in UserDataBase.xml.

Table 6-4 Access control Security Level High

	HTTP/S Web pages without authorization	HTTP/S Web pages with authorization	FTP	Telnet
Project exists, but checkbox for HTTP/S, FTP and Telnet not activated in HW Config. Access to the controller via HTTP/SS, FTP and Telnet is locked.				
	Access blocked	Access blocked	Access blocked	Access blocked

State transition from Security Level Low to Normal

On receipt of the device on delivery, the user creates a project and loads it onto the device. This can be done by using the download functions of the SCOUT, by loading it directly onto the memory card, or via the Web page Manage Config.

Whichever method is used, a project download to the device from the point of view of the Web server corresponds to a transition from **Security Level Low** to **Security Level Normal**.

Resetting the security level from Normal to Low

If the user forgets to edit the UserDataBase.xml during initial commissioning, it will no longer be possible to access FTP, Web services, or access-protected pages during use.

In order to be able to subsequently configure the Web server, **Security Level Low** must be restored. Various methods are available for this purpose:

If there is no mechanical access to the memory card or the device, this can be achieved with the SCOUT function "Delete user data on card". After setting up the user administration, the project must be downloaded again.

Alternatives without SCOUT

Setting the service selector switch to position "8" restores **Security Level Low**. Using this method, the device can always be reset to **Security Level low** by hardware means.

This function of the switch is envisaged only for commissioning purposes and should not be used permanently in normal operation. As of V5.2, the switch position is handled as follows:

- If the switch is already set to "8" at ramp-up, it is ignored.
- The service mode stops immediately when position "8" is exited.
- For safety reasons, a time-controlled disconnection of the service selector switch is performed in position "8". Service mode ends after 120 minutes.
- It is possible to retrigger the disconnection at any time by turning the switch briefly from "8" to "7" and back, for example.
- An LED signal indicates the service mode. For SIMOTION D4X5-2, the service mode causes a slow red flashing of the SF LED.

Because only SIMOTION D modules are fitted with a service selector switch, this functionality is implemented on other SIMOTION modules by making an entry in the simotion.ini file. This requires that the `SERVICE_SELECTOR_MODE=8` entry is set.

For SIMOTION P modules, the PSTATE program is provided for this purpose.

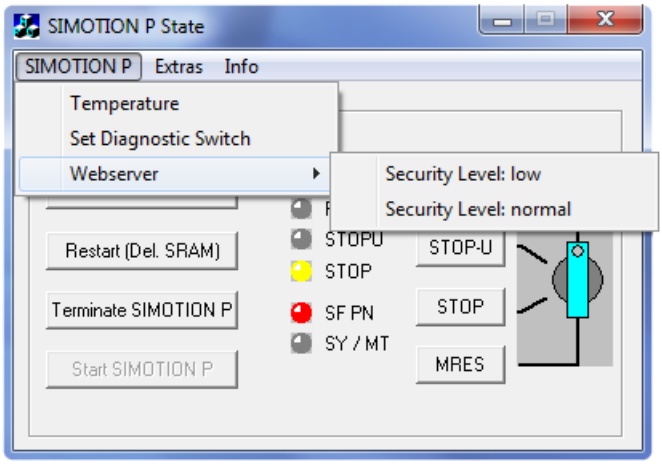


Figure 6-11 SIMOTION P state

Changing the simotion.ini or the setting of the status via the SIMOTION P menu as described above is interpreted as being an intentional action during the commissioning. The service mode is indicated on the Web page.

Note

SSL certificate

Replace the server certificate of the controller with your own to protect HTTPS access.

Note

The availability of the OPC UA server

The availability of the OPC UA server depends on the security level. This means that (similar to security level "normal") the OPC UA server is only accessible even in the "low" security level if:

- The OPC UA is activated in the HW configuration and
- OPC UA is activated in SIMOTION IT (Manage Config -> SIMOTION IT -> OPC UA)

See also

Activating communications services in SCOUT Classic (Page 3564)

TLS/SSL certificates (Page 3574)

Creating key files with the script cert.pl (V4.1 and higher) (Page 3704)

6.1.3.6 User administration

User database UserDataBase.xml

For secure access to the SIMOTION IT pages, users must be created in the user database. Users and groups are stored in file UserDataBase.xml.

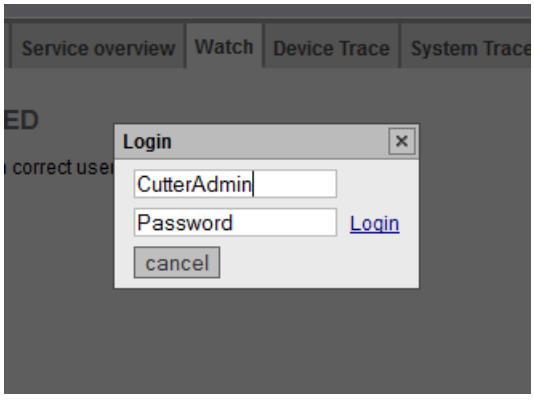


Figure 6-12 Login dialog

The web page Mange Config > SIMOTION IT > Users & Passwords allows user data to be edited in the browser. Alternatively, the file can also be edited offline and then sent to the control.

Chapter Login administration (Page 3658) describes how a user database is set up and edited.

6.1.3.7 Encryption

HTTPS connection with TLS 1.2 and SHA-2

Since the beginning of year 2017, all popular browsers issue an error message when an attempt is made to establish a connection from SHA-1. Consequently, the Web server permits only connections with encryption protocol as of TLS 1.2. SHA-2 functions are used to generate the hash code.

The RSA encryption is replaced by methods of the elliptic curve cryptography. SHA-512 or later is used as hash function.

If the required server certificates are not available, they are generated by the Web server.

Users are provided with Default Root certificates that are also created with SHA-512.

6.1.3.8 TLS/SSL certificates

Securing HTTPS access

Certificates

Certificates must be generated and installed to perform encrypted communication between the browser and Web server.

The as-delivered state includes a device with a standard server certificate and a private key of the Web server provided as a file. These files should be replaced with your own to increase the security of HTTPS access to the device.

There are two ways of acquiring your own server certificate:

- Create a server certificate (self-signed) and a private key using certificate software (e.g. OpenSSL)
- Purchase a server certificate from a certificate authority

On establishing a connection to the Web server, the firmware creates a new server certificate from the root certificate and the private key, if none exists. The server certificate is individualized for the IP address of the interface used for the communication.

Self-signed certificate

When the user makes a connection via HTTPS with the SIMOTION on which the self-signed certificate was stored, the server sends the server certificate belonging to that interface using the SSL protocol.

Browsers will now display a warning that an attempt is being made to communicate via an untrustworthy certificate.

The user can load and install the server certificate via a link to the browser. From now on, the browser is known to the signing certificate authority and no more warnings appear.

Server certificate of a certificate authority

If a certificate of a certificate authority is preinstalled in the browser, the connection is established without a warning message because the certificates are preinstalled in the browser.

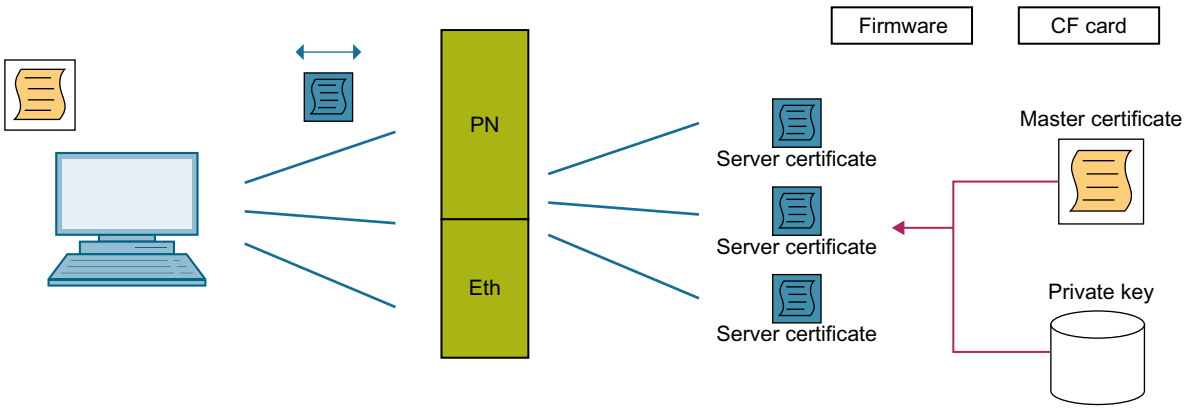


Figure 6-13 Certificate handling concept

See also

Secure Socket Layer (Page 3702)

6.1.3.9 Setting the language for AlarmS and user-defined diagnostics buffer messages

Any of the SIMOTION SCOUT languages can be used when setting the language for AlarmS and user-defined diagnostics buffer messages.

Language localization

SIMOTION IT uses four rules for language selection. The first rule that applies is used:

1. Configuration constant ForceUserMsgLanguageID

The language can be set with the configuration constant `ForceUserMsgLanguageID`. This variable is set to the corresponding country code (decimal value) for this purpose. The selected language must exist. If it does not, the HEX display is used.

To change the AlarmS language setting, change the configuration constant "ForceUserMsgLanguageID".

- Activate the set language selection. To do this, upload the user-defined AlarmS and diagnostic buffer messages exported from SCOUT to the CPU again.

You will find more information about the configuration constants in section *Configuration constants* in the *SIMOTION IT Programming and Web Services* Manual. The LCID country codes (Page 3735) are listed in the appendix.

2. SIMOTION SCOUT-Export

Performing a SCOUT export of user-defined AlarmS and diagnostic buffer messages and then uploading (Page 3636) this data, sets the SIMOTION IT language to the same language as is set in SCOUT.

3. Language of system diagnostics buffer texts

An attempt is made to find the language that matches the installed system diagnostics buffer texts.

4. Other language settings

If no matching language is found among the system diagnostics buffer texts, the system default language is selected instead.

The language which has been selected is documented in the syslog file.

6.1.3.10 Access protection

Description

The TO configuration contains essential machine configurations (e.g. axis settings, cams, coupling types, controller settings, etc.).

With access protection enabled it is possible to prevent read back of the TO configuration, the cams, the SIMOTION device configuration and the SINAMICS drive data as follows:

- Data can only be uploaded from the controller with a password.

- Data can only be read out with the web server with a password

- Data cannot be read/analyzed from a CF card without a password

Note

The extended functionality results in improved access protection for the configuration of the TO configuration.

As a further security measure, the user should ensure that unauthorized persons have no access to the memory card.

Management

- Access protection is based on SIMOTION IT and is managed via SIMOTION IT.
  If access protection is not yet active, a new password can be defined on the "Access protection" page.
  Defining the password sets up read protection for the project data on the card in the background. The current page is then re-loaded.
- If access protection has already been set up and is active (i.e. you have not yet successfully authorized yourself against the access protection), a login page is loaded.
- If access protection has already been set up and you have successfully authorized yourself against the access protection, you have the options to remove the access protection (read protection of project data on the card is removed), change the password, or activate the access protection (lock SIMOTION IT and project). The previous password does not have to be entered again.

  Note

  If you want to remove the access protection and the access protection password is not known, you must delete the entire card and reload the Siemens card image. It is not sufficient to delete the USER directory.

Scope of access protection

The following are locked without authorization with the correct password:

- IT Diag
- FTP (even when enabled in HW Config)
- OPC UA (even when enabled in HW Config)
- Upload from SCOUT
- Access protection is retained with switch position 8

Read protection of project data

To protect the TO data (axes, cams, ...), the project data on the card is protected against being read out. As a result, the following use cases are not supported when access protection is active:

- Upgrade using device update tool
- System function _activateConfiguration() (modular machine)

Authorization

The authorization is lost during a reset if you authorize yourself at the access protection and then perform a download with reset in this state.

Authorize yourself again after the controller has restarted. Only then can the SCOUT establish a new online connection and continue the download.

6.1.4 Operation (software)

6.1.4.1 SIMOTION IT Diagnostics overview and general functions

Overview

The SIMOTION device manages predefined diagnostic standard pages. These pages can be displayed using a generally available browser via Ethernet.

SIMOTION IT can be accessed via all available Ethernet interfaces of SIMOTION, including the PROFINET IO interface.

You can also create your own HTML pages and integrate servicing and diagnostics information.

Purpose and benefits

The purpose and benefits of HTML diagnostics pages are as follows:

- Preconfigured diagnostics pages are available to the user for the direct diagnosis of the SIMOTION device.
- Service and diagnostics information of the device can be accessed without manufacturer-specific programs to assist in production monitoring or diagnostics.
- User-defined HTML pages can be integrated.

6.1.4.2 SIMOTION IT log-on and log-off

Log on

If the control is in security level **Normal**, it is necessary to log on to access the protected pages of the control.

Figure 6-14 Login without registration

Login will only be successful if the associated password has been created in the User administration (Page 3658).

Figure 6-15 Login with registration

See also

Security concept (Page 3569)

Logging off

Logout from SIMOTION IT is performed via the **Logout** link in the login area.

Note

Exiting the browser without logout

Exiting the browser without logout results in the session remaining active on the server for another 5 minutes before it is closed. The technical reason for this behavior is the FormBased Authentication.

This behavior can be improved by deleting the cookies after closing the browser.

6.1.4.3 Standard pages

General links

Each SIMOTION IT page includes three general links:

Figure 6-16 General links

- "Watch" gives you access to the watch function (Page 3589).
- "Overview" displays in the service overview (Page 3586).
- "Copy Link" copies the URL of the current page to the clipboard.

Watch link

The Watch link provides fast access to the Watch page in a separate browser window.

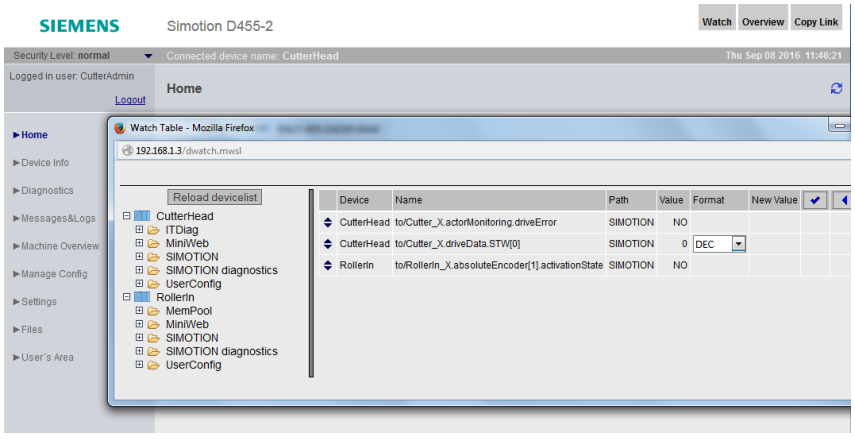


Figure 6-17 Watch link

Overview link

The Overview link calls the Overview page in a separate browser window.

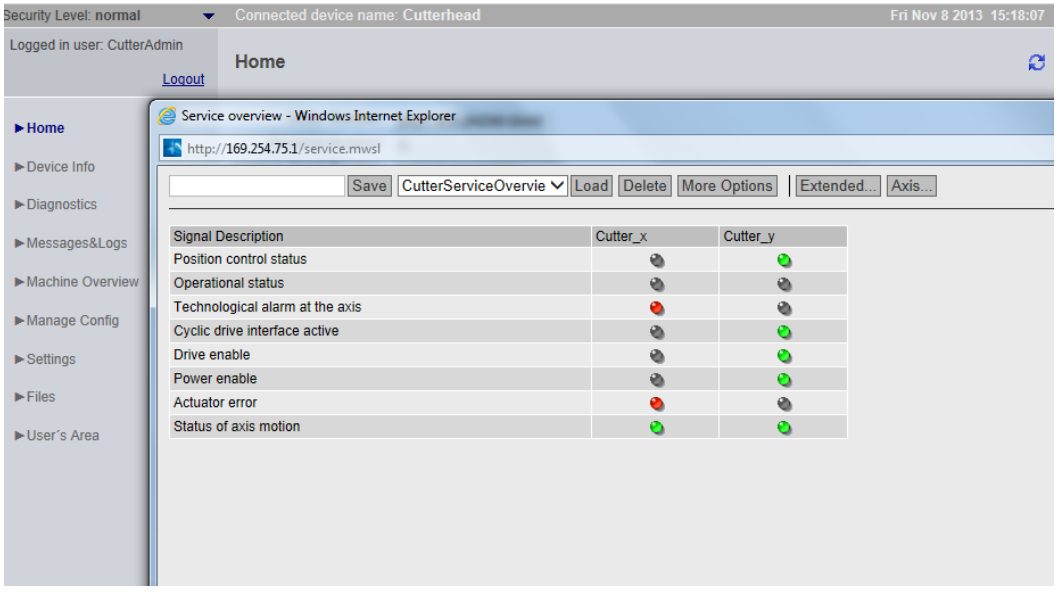


Figure 6-18 Overview link

Copy Link

The **Copy Link** copies the URL of the current page to the clipboard.

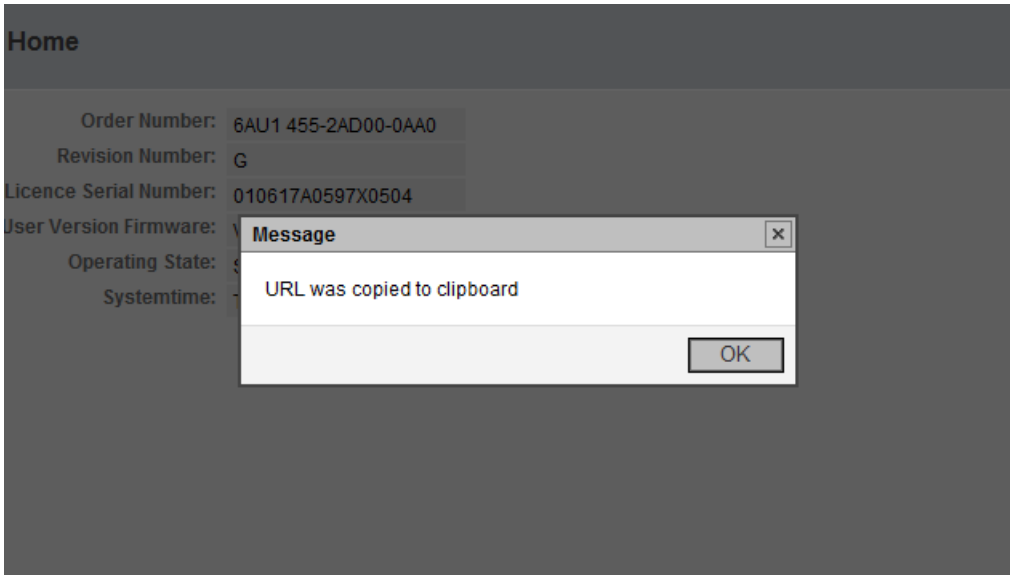


Figure 6-19 Copy link Internet Explorer

The screenshot shows the Internet Explorer message. In Firefox, another dialog opens that allows the link to be copied manually with **Ctrl+C**.

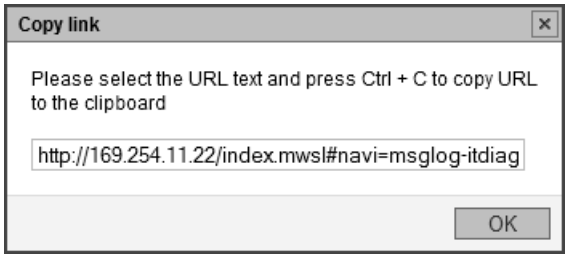


Figure 6-20 Copy link Firefox

Message system

The message system of SIMOTION IT shows additional information as pop-up messages at the bottom right-hand edge of a page.

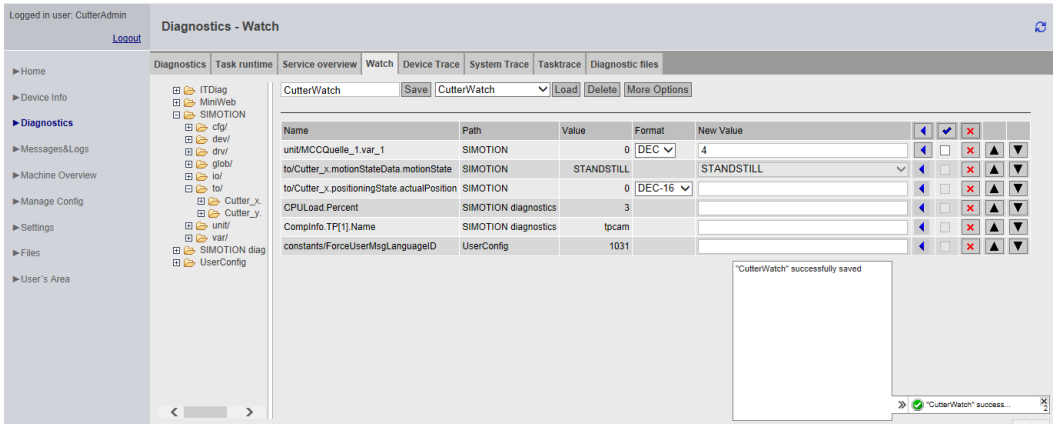


Figure 6-21 Message system example

The message system displays additional information. In this example, successful storage of the Watch settings is displayed as "CutterWatch".

Processing is not interrupted when a message is displayed. The numeric value **2** in the above example shows the time in seconds that the message remains visible.

Home

SIMOTION device data

The following current data of the SIMOTION device is displayed on the home page:

Order Number	Article number of the device
Revision Number	Hardware version
Licence Serial Number	The license key is tied to this serial number
User Version Firmware	SIMOTION kernel user version
Operating State	Operating state of the SIMOTION device RUN, STOP, STOPU
Systemtime	Current time-of-day of the SIMOTION device

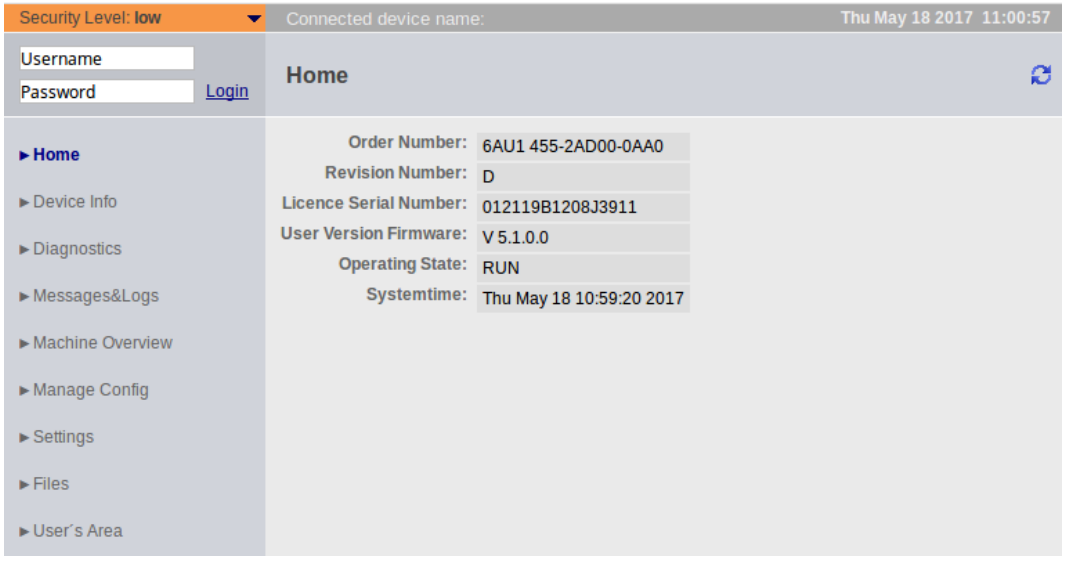


Figure 6-22 Home page

The screenshot shows the appearance of the Home page before a user or password has been created in the `UserDataBase.xml` user database.

An empty user database causes security level **Security Level low** if no project exists already on the controller. You can access the page where you create the user and passwords via the **User & Passwords** link. All subsequent screenshots show the SIMOTION IT pages after login of the user CutterAdmin and security level **Security Level normal**. The user CutterAdmin is used in the manual as an example and accordingly must exist the user database.

For more information regarding the current device data, refer to the "Device Info (Page 3582)" page.

See also

Watch (Page 3589)

Service overview (Page 3586)

Device Info

Hardware and firmware information

The following current hardware and firmware information of the SIMOTION device is displayed on the **Device Info** page:

Manufacturer Name	Siemens AG
Order Number	Article number of the device
Revision Number	Hardware version
Serial Number	Serial number of the SIMOTION device

User Version Firmware	SIMOTION Kernel user version
Build Number	Internal version number
Additional Hardware	Installed components of the SIMOTION device including: Article number, serial number, revision number, firmware name, user version number, internal version number
Technological Packages	Loaded technology packages including: Package name, user version number, internal version number

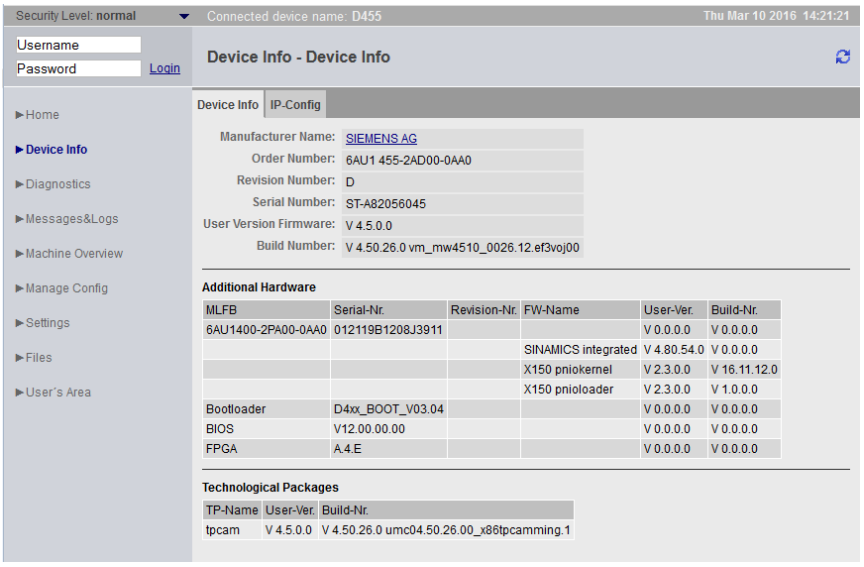


Figure 6-23 Device Info

Here, the Device Info page is shown after the example user has successfully logged in CutterAdmin.

IP Config

Data of the Ethernet and PROFINET interfaces of the SIMOTION device

The following current Ethernet and PROFINET interfaces data of the SIMOTION device is displayed on the **IP-Config** page:

IP Address	Address of the interface
Subnet Mask	Subnet mask of the interface
MAC Address	Physical address of the network card

Gateway	Default gateway of the interface The corresponding information is always displayed in the first column. It is not necessarily directly related to the IP address of the column and may even have been configured for the other interfaces.
Ethernet-port status:	Overview of Ethernet ports. The port speed and communication type are output for active ports.

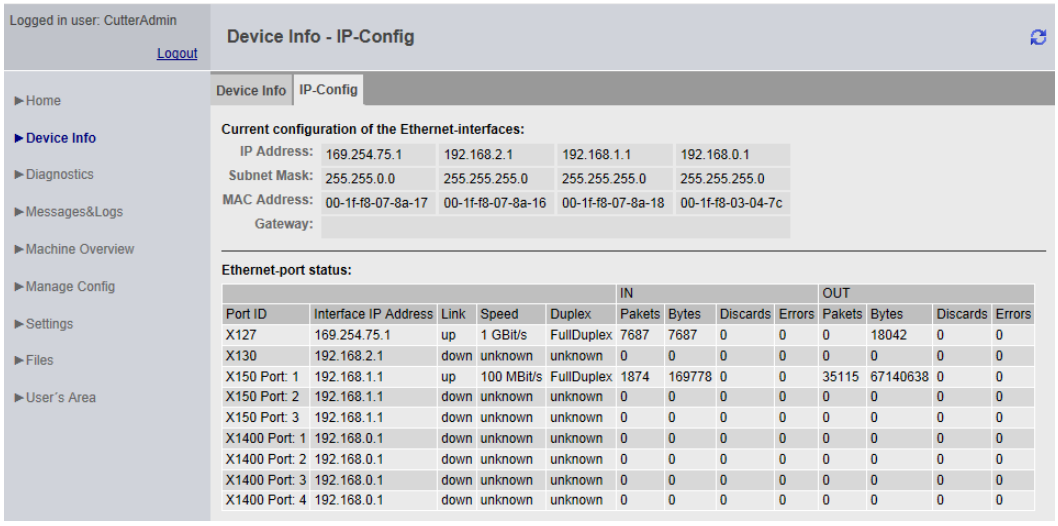


Figure 6-24 IP Config

Port ID	Name of the Ethernet or PROFINET port as stated on the hardware housing.
Interface IP Address	IP address of the interface
Link	Switching property of the port
Speed	Communications speed of the port
Duplex	Communications type of the port
Pakets - IN	Number of packets received at this port.
Bytes - IN	Number of octets received at this port.
Discards - IN	Number of received packets rejected for internal system reasons (e.g. due to system overload).
Errors - IN	Number of received packets not processed by higher protocol layers because of a detected error. For example, transmission/reception faults of the block and collisions.
Pakets - OUT	Number of packets sent at this port.
Bytes - OUT	Number of octets sent at this port.
Discards - OUT	Number of transmission requests for packets that were rejected. Packets that were rejected even though no errors that would have prevented transmission were detected are also counted.
Errors - OUT	Number of packets that were not sent due to an error.

Diagnostics

Overview of the general state of the SIMOTION device

The following states of the SIMOTION device are displayed on the **Diagnostics** page:

Systemtime	Current time-of-day of the SIMOTION device
Timezone	Current difference between the Systemtime and GMT in minutes
CPU Load by cyclic Tasks	Computation time of servo and IPO levels as a percentage of the total computation time
Memory Load	Size and allocation of the RAM disk, memory, memory card, and non-volatile memory
Operating State	Current operating state of the SIMOTION device
Web server Connection State	Information about the current connection status of the Web server.

Select the tabs on the page to access more detailed information.

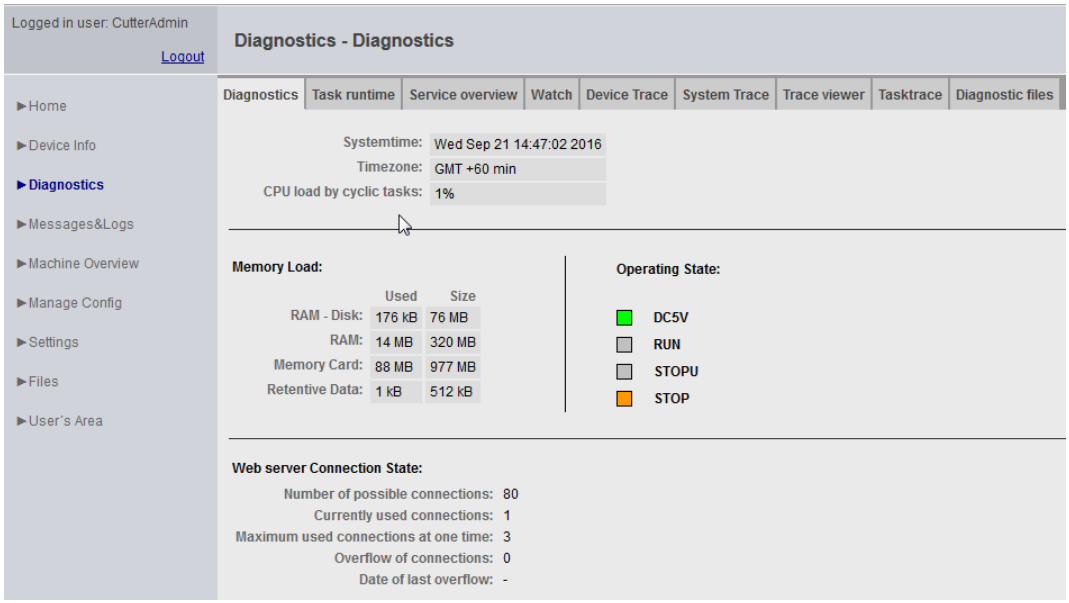


Figure 6-25 Diagnostics

Task runtime

Information on task runtimes and states

On the **Task runtime** page (opened via **Diagnostics > Task runtime**), you can view the following information:

Taskname	Name of the task
Status	Current status of the task

Actual	Current runtime of the task in ms
Min	Minimum runtime of the task in ms
Max	Maximum runtime of the task in ms
Average	Average runtime of the task in ms

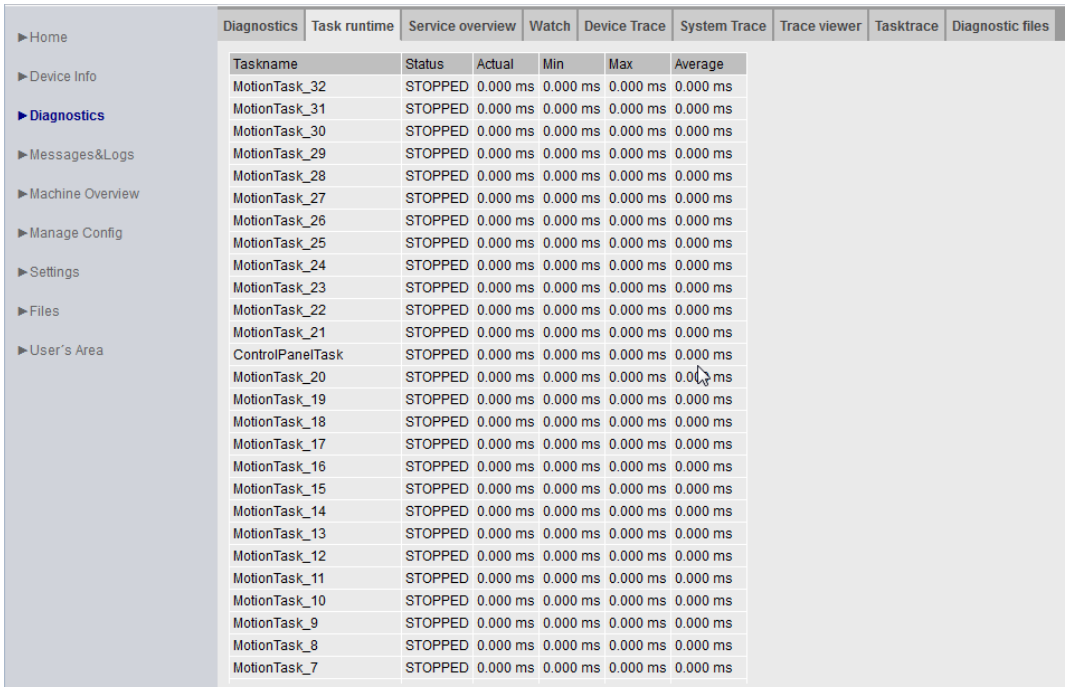


Figure 6-26 Task Runtime

Service overview

Service overview

SIMOTION SCOUT provides an overview screen that displays the state of the axes available in the project. The Web server provides a corresponding page.

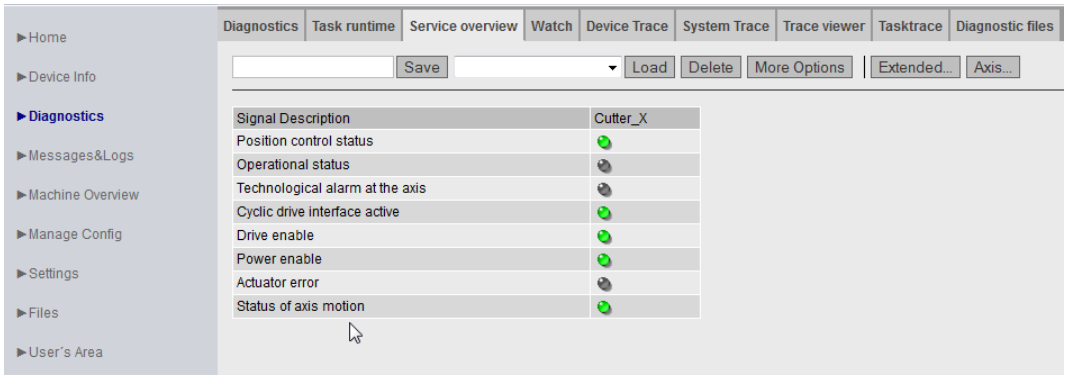


Figure 6-27 Service overview

The columns in the table represent each of the axes. The **Axis** button displays a selection of all the available axes from which the required axes can be selected.

The **Save** button saves the current setting in the device. A name for the setting must be entered in the input field to the left of the **Save** button.

The **Load** button loads a setting. The **Delete** button deletes a setting.

The **Extended...** button opens a window in which the required system variables can be selected.

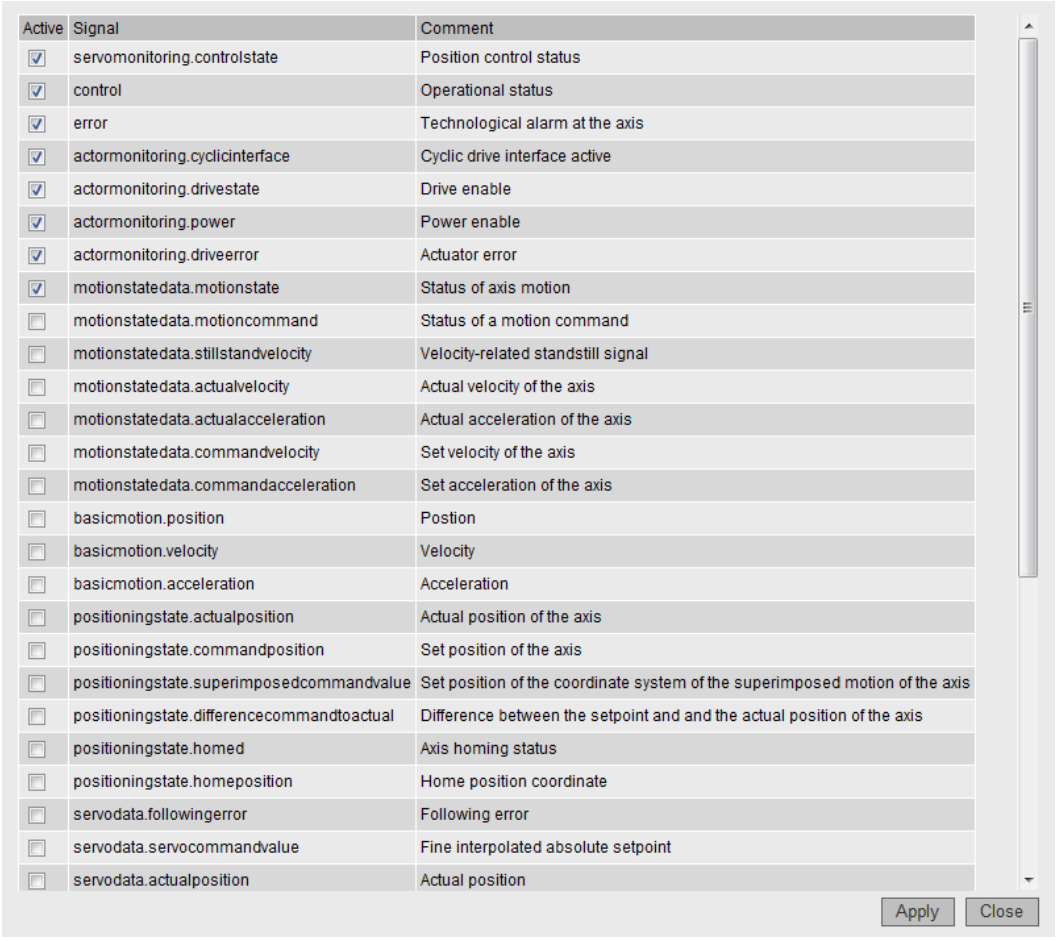


Figure 6-28 **Extended... button**: Selection of variables

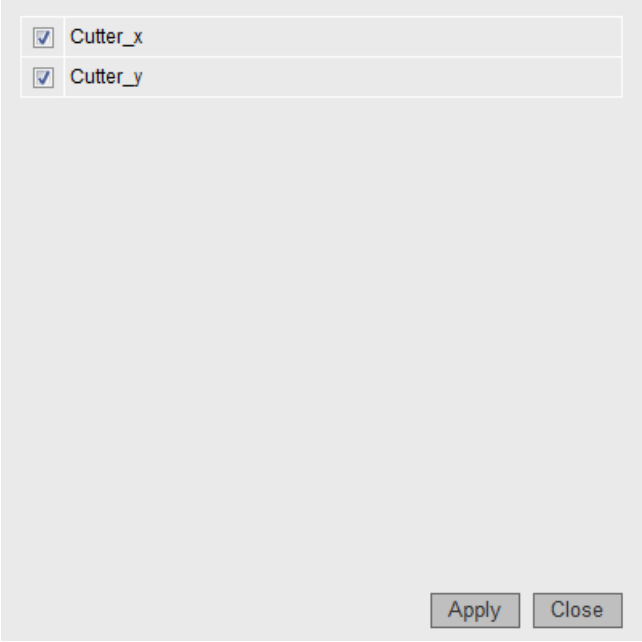


Figure 6-29 **Axis... button**: Selecting the axes

More Options

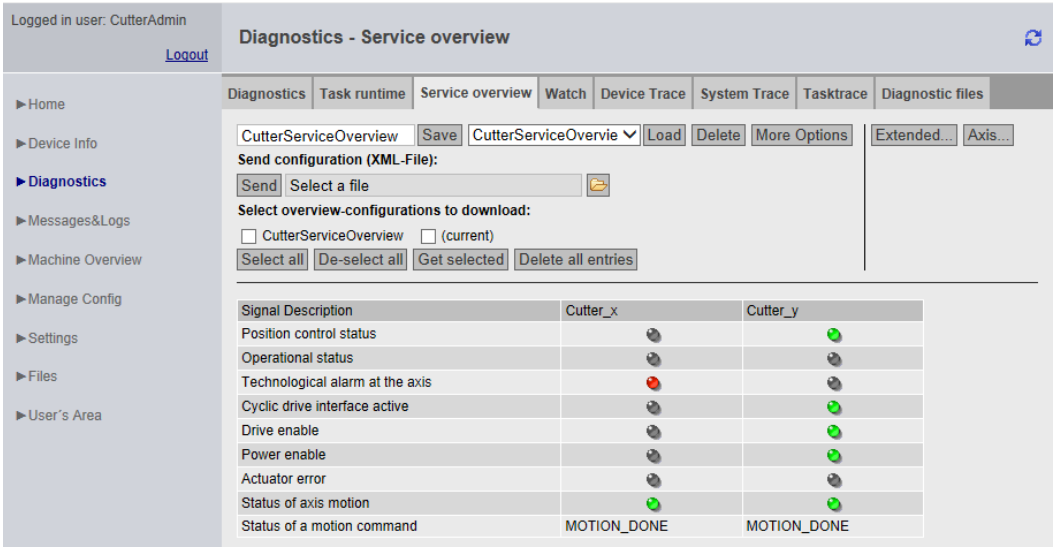


Figure 6-30 Service overview More Options

The **More Options** button extends the upper screen area to display additional functions. Additional buttons for selecting signals are displayed on the Service Overview page.

If multiple configurations are saved on the device, these configurations are offered for selection, can be selected and saved. The **Send** button loads a previously saved configuration to the device.

Watch

Watch table

This page combines a variable browser and a watch table. The variables are entered in the watch table with the aid of the browser. Variables of several devices are combined in the watch table.

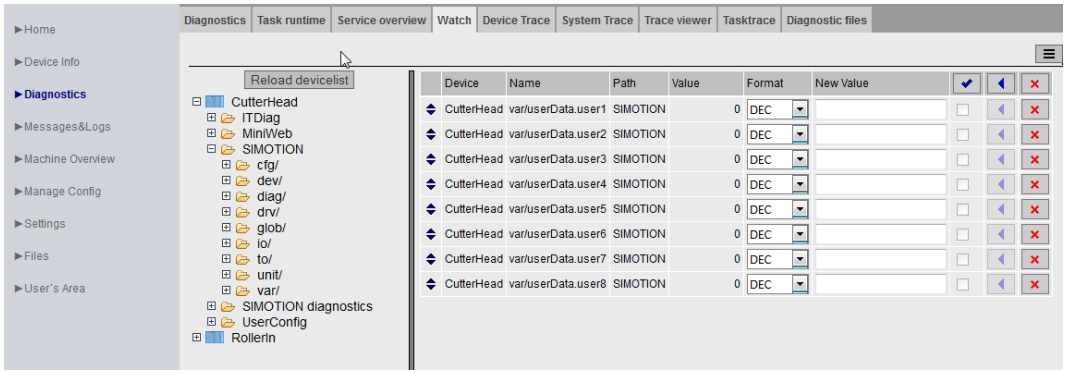


Figure 6-31 Watch table

For monitoring variables, the Web server provides a watch table and a symbol browser. The symbol browser allows browsing the variable management area of the CPUs connected with the SIMOTION controller. The variables are displayed in a tree topology on the left-hand side. The selected variables are displayed on the right-hand side and can be edited for the Watch. A maximum of 600 variables and array elements can be represented in the watch table.

In order to monitor unit variables, the "Permit OPC-UA/-XML" option must have been activated in the compiler settings for the associated unit. See Making unit variables available (Page 3701)

Only users who have logged in can access this page. See Log-in administration (Page 3658)

Device monitoring

Previously detected devices are monitored for sign-of-life. The **Reload devicelist** button causes all active devices to be detected and displayed next to the variables table.

If it is determined that a device is no longer available, the affected device and the associated entries in the variables list are displayed red.

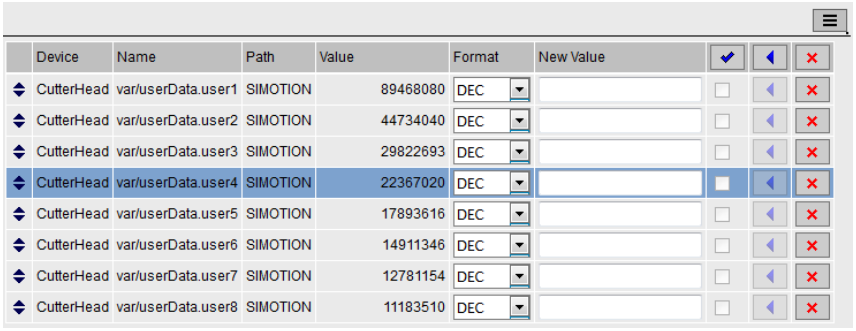
The variable monitoring is resumed after restoring an interrupted connection. All affected displays are then displayed in the normal color again.

Operating the watch table

The watch table can be operated not only with the mouse, but also with the keyboard. The focus within the table can be moved to any input field with the arrow keys.

The first column serves for marking and moving table cells.

Rows can be marked, and with pressed Shift key, by moving the cursor up or down (the blue-marked field) with either the mouse or the arrow keys on the keyboard.

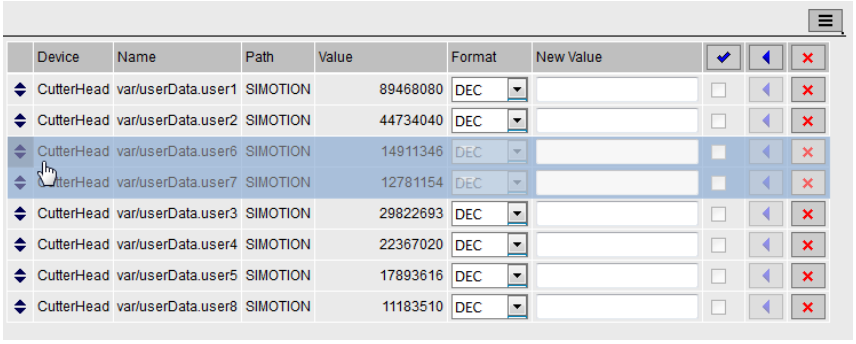


Device	Name	Path	Value	Format	New Value
CutterHead	var/userData.user1	SIMOTION	89468080	DEC	
CutterHead	var/userData.user2	SIMOTION	44734040	DEC	
CutterHead	var/userData.user3	SIMOTION	29822693	DEC	
CutterHead	var/userData.user4	SIMOTION	22367020	DEC	
CutterHead	var/userData.user5	SIMOTION	17893616	DEC	
CutterHead	var/userData.user6	SIMOTION	14911346	DEC	
CutterHead	var/userData.user7	SIMOTION	12781154	DEC	
CutterHead	var/userData.user8	SIMOTION	11183510	DEC	

Figure 6-32 Marking watch table entries

Moving a selection

The cursor can be moved without deleting the marked area by pressing the Ctrl key and the arrow keys.



Device	Name	Path	Value	Format	New Value
CutterHead	var/userData.user1	SIMOTION	89468080	DEC	
CutterHead	var/userData.user2	SIMOTION	44734040	DEC	
CutterHead	var/userData.user6	SIMOTION	14911346	DEC	
CutterHead	var/userData.user7	SIMOTION	12781154	DEC	
CutterHead	var/userData.user3	SIMOTION	29822693	DEC	
CutterHead	var/userData.user4	SIMOTION	22367020	DEC	
CutterHead	var/userData.user5	SIMOTION	17893616	DEC	
CutterHead	var/userData.user8	SIMOTION	11183510	DEC	

Figure 6-33 Moving watch table entries

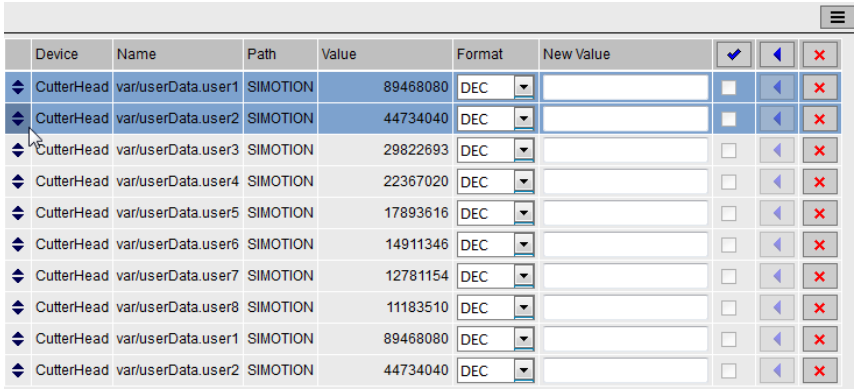
The row in which the cursor is positioned can also be marked with pressed Ctrl key and pressing the spacebar or also with pressed Ctrl key and left-click without deleting any other existing marking.

Marked rows can be grabbed with the mouse in the first column within the marking and then moved up or down. Alternatively, you can move using the keyboard by pressing the Alt key and the arrow keys, Home or End.

Marked rows can be deleted by pressing the <Del> key.

Copying a selection

Marked rows can be copied to an internal clipboard by pressing <Ctrl>-<C> and appended at the end of the watch table by pressing the <Ctrl>-<V>.

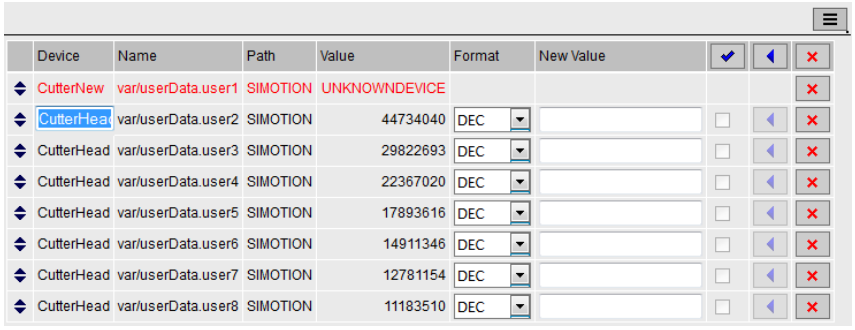


	Device	Name	Path	Value	Format	New Value			
	CutterHead	var/userData.user1	SIMOTION	89468080	DEC				
	CutterHead	var/userData.user2	SIMOTION	44734040	DEC				
	CutterHead	var/userData.user3	SIMOTION	29822693	DEC				
	CutterHead	var/userData.user4	SIMOTION	22367020	DEC				
	CutterHead	var/userData.user5	SIMOTION	17893616	DEC				
	CutterHead	var/userData.user6	SIMOTION	14911346	DEC				
	CutterHead	var/userData.user7	SIMOTION	12781154	DEC				
	CutterHead	var/userData.user8	SIMOTION	11183510	DEC				
	CutterHead	var/userData.user1	SIMOTION	89468080	DEC				
	CutterHead	var/userData.user2	SIMOTION	44734040	DEC				

Figure 6-34 Copy-and-Paste watch table

Editing an entry

The example shows the situation after copying and pasting the first two rows.



	Device	Name	Path	Value	Format	New Value			
	CutterNew	var/userData.user1	SIMOTION	UNKNOWNDEVICE					
	CutterHead	var/userData.user2	SIMOTION	44734040	DEC				
	CutterHead	var/userData.user3	SIMOTION	29822693	DEC				
	CutterHead	var/userData.user4	SIMOTION	22367020	DEC				
	CutterHead	var/userData.user5	SIMOTION	17893616	DEC				
	CutterHead	var/userData.user6	SIMOTION	14911346	DEC				
	CutterHead	var/userData.user7	SIMOTION	12781154	DEC				
	CutterHead	var/userData.user8	SIMOTION	11183510	DEC				

Figure 6-35 Editing a watch table

The associated entry can be edited by clicking in one of the Device, Name or Path fields.

This screenshot shows the situation after entering a non-connected device. Consequently, the first row is shown red.

Loading and saving of watch tables with the Menu button

The Menu button at the right above the watch table bundles the functions for saving and loading the watch tables.

In addition, a selection of saved watch tables can be exported to and reimported from a connected computer.

Similarly, watch tables exported from SCOUT can be imported.

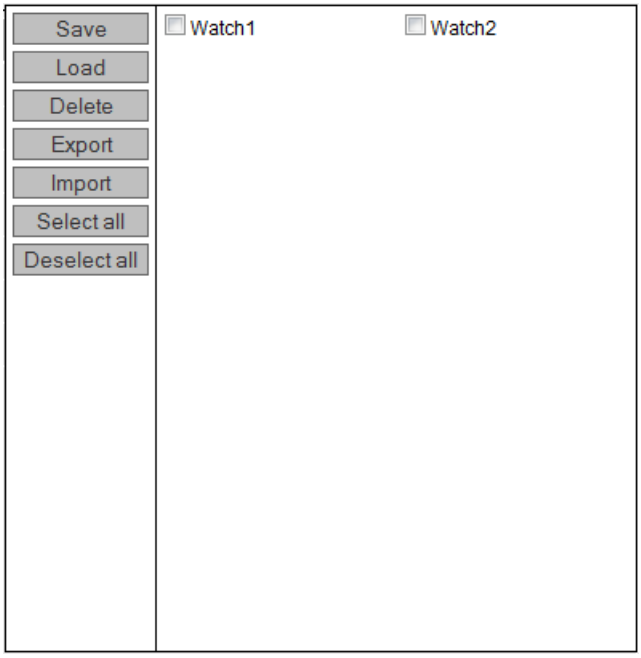


Figure 6-36 Menu button dialog box

- Save - Saves the selected watch tables after the specification of a name (Watch1, Watch2, ...).
- Load - Loads the selected watch tables.
- Delete - Deletes the selected watch tables.
- Export - Exports the selected watch tables to a connected computer as an XML file.
- Import - Imports watch tables from a connected computer. This function imports watch tables exported from SCOUT. Not only individual variables, but also structures and arrays can be imported. Before being entered in the watch table, they are resolved into individual variables. The variables that belong to a structure or array are determined online by browsing. This operation can take quite some time, depending on the number of variables and the CPU loading. The renewed saving of the watch table resolved into individual variables makes future imports much faster.
- Select all - Deletes all entries selected in the dialog box.
- Deselect all - Revokes the selection for all entries of the dialog box.

Note

After loading a watch table and adding new drive parameters, it is possible that the display no longer responds.

Display formats

The **Format** column allows you to change the display format for integer and floating-point variables.

- DEC for decimal display (default).
- HEX for hexadecimal display.
- BIN for binary display.

All control values are interpreted according to this setting.

Table 6-5 Display formats for floating-point numbers

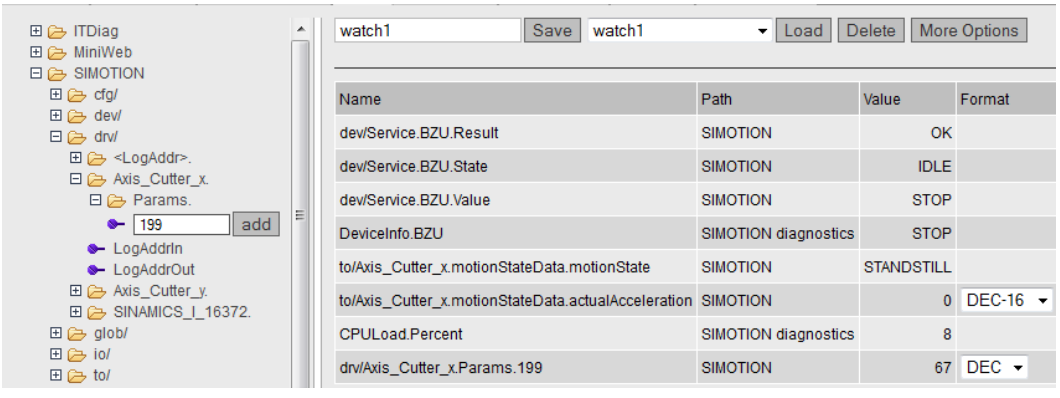
Format	Lowest value	Highest value	EXP notation
DEC-10	0.000000001	9999999999	*.*********E+-*
DEC-16	0.000000000000001	9999999999999999	*.**************E+-*
DEC-20	0.00000000000000000001	99999999999999999999	*.**************E+-*
DEC n.3	Three decimal places are displayed or EXP format if the value < 0.001 or > 1e+21		
EXP	*.**********E+-*		

Accessing the drive parameters

The drive parameters are accessed via a tree topology. The parameters are selected using the same method as when accessing variables via the variable provider. See Variable providers (Page 3676)

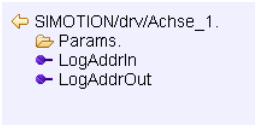
Simply click a variable in the tree overview to select it. Depending on the variable type, additional values, such as the parameter numbers, are also queried.

Parameters are displayed as a number without a preceding 'p' or 'r'. For example, parameter r0002 becomes 0002.

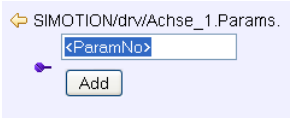


Three options are provided for accessing the drive parameters:

1. Axis technology object

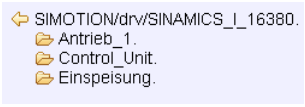


Selecting a technology object.

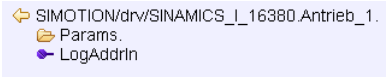


Selecting a drive parameter.

2. Drive object addressing



Selecting a drive object. The name is generated from the diagnostic address.



Selecting a drive.

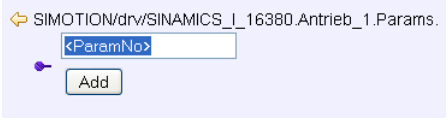


Figure 6-37 Selecting a DO parameter

3. Logical address

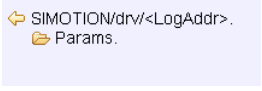
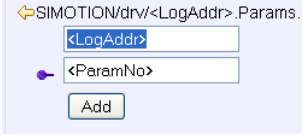


Figure 6-38 Selecting a logical address



Selecting a drive parameter and a logical address.

Device Trace

Setting up a Device Trace

The SIMOTION controller provides the user with the option of setting up a device trace via a Web service.

As of Version 4.2 not only the device trace described in this section is provided, but also a distributed trace (Page 3598) (System Trace).

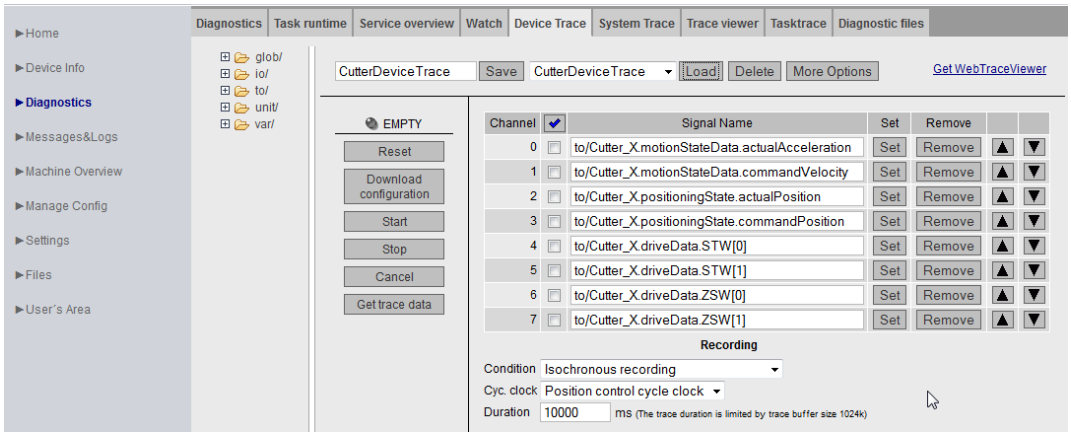


Figure 6-39 Device trace

Procedure for creating and executing a device trace:

- Select the **Device Trace** radio button
- Select the required signal from the provider list (glob, io, to, unit or var)
- The marked symbol is placed on the required signal with the **Set** button, double-clicking, or by drag-and-drop.
- Set the recording and trigger conditions
- **Download configuration** – Load the settings to the controller
- **Start** – Start the trace
- **Stop** – Stop the trace (only required for a manual trace)
- **Cancel** – Delete the settings from the controller
- **Get trace data** – Load the trace results to the device memory or to a file on the PC.
  - View the trace data on the Trace Viewer (Page 3709) page.
  - View a trace file stored in WTRC format with the WebTraceViewer or SCOUT.

In order to monitor unit variables, the "Permit OPC-UA/-XML" option must have been activated in the compiler settings for the associated unit. See Making unit variables available (Page 3701).

Note

Data types

TIME and STRING data types cannot be recorded

- Not all elementary data types can be tracked in the trace.
- Only the bit data types and numeric data types can be tracked in the trace.
- However, it is not possible to track TIME and STRING data types.

Trace display

Up to 128 signals can be displayed simultaneously in the WebTraceViewer. By contrast, only 8 signals can be displayed simultaneously in SCOUT.

Once the signals have been selected, the desired recording and trigger conditions must then be assigned.

The WebtraceViewer can be downloaded and installed from the **Get WebTraceViewer** link.

The **More Options** button expands the upper area of the screen to include options for saving the device trace settings on a PC and subsequently reloading them to the controller.

Only users who have logged in can access this page. See Login administration (Page 3658)

Note

Only a limited amount of memory, arranged as a ring buffer, is available for the Trace. 512 KB is available for SIMOTION C, SIMOTION D410-2, and 1024 KB is available for all other SIMOTION modules.

Trace modes

The device trace can be run in three modes:

1. Isochronous recording (recording immediately)
   The trace starts immediately and runs until the recording time set at Duration is reached.
2. Isochronous recording - manual trace (recording immediately)
   The trace starts immediately and runs until it is stopped by the operator. The trace buffer then contains data which was recorded for the time set in Duration before stop was triggered.
3. Isochronous recording – triggered (recording triggered)
   The trace starts when a trigger event occurs and stops when a parameterizable time expires or when the trace buffer is full.
4. EndlessIsochronous recording – endless
   The trace starts immediately and runs until it is ended with Stop. The trace data can be monitored during recording in the Tab Trace Viewer. You can save recordings made by endless trace in CSV format.

Trigger

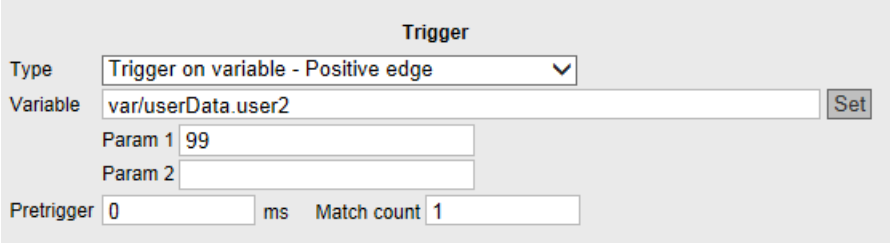


Figure 6-40 Device trace trigger

You can find a description of the recording settings and trigger conditions in the System trace (Page 3598) section.

Saving and loading a trace configuration

You can save a configuration with a name on the device with the **Save** button and load it again with the **Load** button. You can find a more detailed description of the **More Options** functionality in the Service overview (Page 3586) section.

Drag-and-drop

The drag-and-drop functionality enables variables to be dragged to the table rows of the signals.

Moving table rows

Using drag-and-drop you can move the table rows containing the signals. This functionality is also available in similar tables on the Watch and System Trace page.

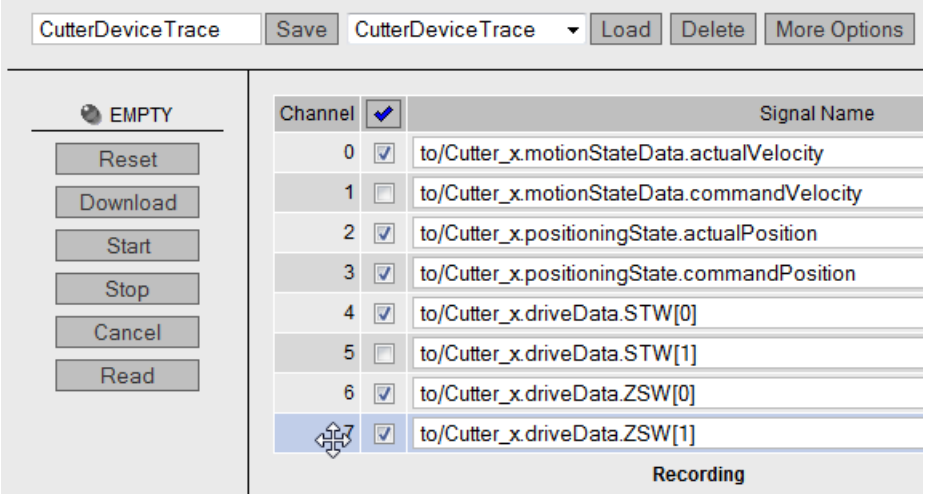


Figure 6-41 Drag-and-drop table rows

Select the required table row. Keep the left mouse button pressed and move the row to the desired position.

System Trace

Setting up and executing a system trace

The system trace is available as of SIMOTION Version 4.2. The system trace records a trace involving multiple devices.

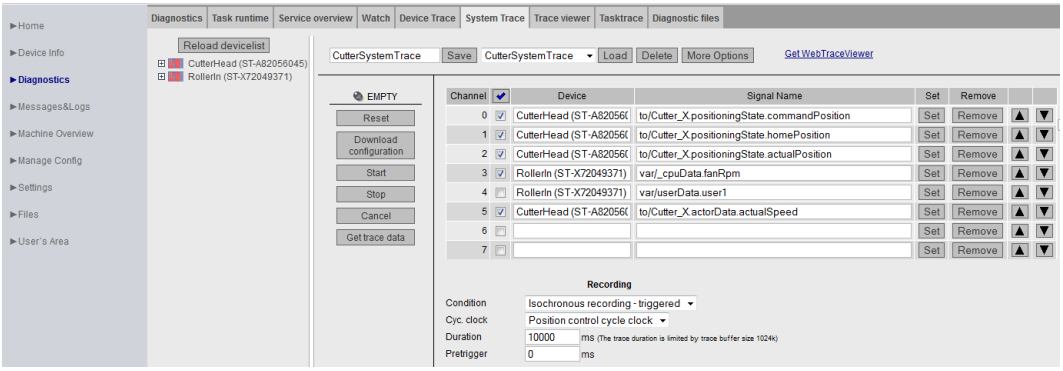


Figure 6-42 System trace (partial view)

Requirements for the system trace:

- It is essential that the CPUs communicate via PROFINET.
- There must be an isochronous connection between the CPUs.
- Direct data exchange (peer-to-peer communication) must be configured.
- The PROFINET Sync Master must be a SIMOTION device.

Procedure for creating and executing a system trace:

- Select the **System Trace** radio button
- Select the required signal from the provider device list (glob, io, to, unit or var)
- The marked symbol is placed on the required signal with the Set button, double-clicking, or by drag-and-drop.
- Set the recording and trigger conditions
- **Download configuration** – Load the settings to the controller
- **Start** – Start the system trace
- **Stop** – Stop the system trace (necessary only for manual trace)
- **Get trace data** – Load the trace results to the device memory or to a file on the PC.
  - View the trace data on the Trace Viewer (Page 3709) page.
  - View a trace file stored in WTRC format with the WebTraceViewer or SCOUT.

In order to monitor unit variables, the "Permit OPC-UA/-XML" option must have been activated in the compiler settings for the associated unit. See Making unit variables available. (Page 3701)

Note

Data types

TIME and STRING data types cannot be recorded

- Not all elementary data types can be tracked in the trace.
- Only the bit data types and numeric data types can be tracked in the trace.
- However, it is not possible to track TIME and STRING data types.

Requirements

The devices must be connected and synchronized via PROFINET IO for time synchronization of the distributed trace to function correctly.

Note

Error message: Error while synchronizing timestamp

If multiple controllers with the same IP address are involved on a system trace over sync domains, this error message is issued. The trace is then not recorded.

Remedy: Unique IP addresses must be used.

Quantity structures

Number of devices

- 128 signals on 128 CPUs are possible. 32 signals per CPU are possible.

Number of triggers

- Just one trigger is possible for each device. A total of four triggers are possible for the entire configuration. Depending on the utilization of the devices, the number of different possible devices can vary. As a recommendation, no more than 10 different devices should be used at the same time.

Trace display

Up to 128 signals are displayed simultaneously in the WebTraceViewer. By contrast, only 8 signals can be displayed simultaneously in SCOUT.

Once the signals have been selected, the desired recording and trigger conditions must then be assigned.

The WebtraceViewer can then be downloaded and installed from the **Get WebTraceViewer** link.

Trace modes

The system trace can only be run in 'triggered' mode. The trace starts when a trigger event occurs and stops when a parameterizable time expires or when the trace buffer is full.

Recording settings

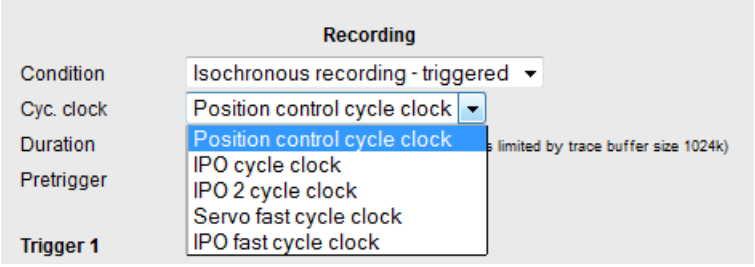


Figure 6-43 Recording basic cycle clock setting

- Condition: Measured value acquisition
- Cyc. Clock: Basic cycle clock
- Duration: Recording in a ring buffer. 512 KB memory (SIMOTION C, SIMOTION D410-2) or 1024 KB (all others)

- Pretrigger = Time in ms when the trigger is activated, this "run-in" is included in the recording

Trigger conditions

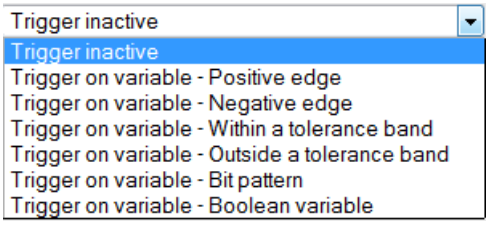


Figure 6-44 Trigger setting

Designation	Description	Operand 1	Operand 2
Positive Edge	Positive edge Triggers when the variable was below the threshold and then overshoots it.	Threshold value	-
Negative Edge	Negative edge Triggers when the variable was above the threshold and then undershoots it.	Threshold value	-
Within a tolerance band	Within a value range Triggered if variable is within the specified interval.	Lower limit of the interval	Upper limit of the interval
Outside a tolerance band	Outside a value range Triggered if variable is outside the specified interval.	Lower limit of the interval	Upper limit of the interval

Designation	Description	Operand 1	Operand 2
Bit pattern	The bit pattern triggers if the relevant bit is 1 both in the variable and in the bit pattern.	Bit pattern	-
Boolean Variable	Boolean variable Triggers depending on operand 1.	0 = trace triggered after a 1 →0 transition 1 = trace triggered after a 0 →1 transition	

Overview of trigger conditions

All mentioned operands must be specified for the download to function. In this case, "Trigger expression invalid" is displayed as error message.

Initialization

The trace variables and trigger conditions are transferred to the devices concerned in order to initialize the trace. If the initialization has been completed without errors on at least one device, the trace can start.

Note

Deleting a trace

A SCOUT trace is not deleted by SIMOTION IT diagnostics.

A SIMOTION IT Diagnostics trace can be deleted by SIMOTION SCOUT. In this case, a dialog appears in SIMOTION SCOUT in which it can be specified whether the available trace parameterization should be overwritten.

Deleting a trace up to Version 4.3

A SIMOTION SCOUT trace is not deleted by the SIMOTION IT Diagnostics.

A SIMOTION IT Diagnostics trace is not deleted by SIMOTION SCOUT.

Note

Downloading a trace

If a SIMOTION SCOUT trace exists on the device, a SIMOTION IT Diagnostics trace cannot be loaded.

Viewing the trace

The trace data can be displayed with the WebTraceViewer PC program or SIMOTION SCOUT as of V4.4.

Drag-and-drop

Variables can easily be moved into the trigger conditions using drag-and-drop.

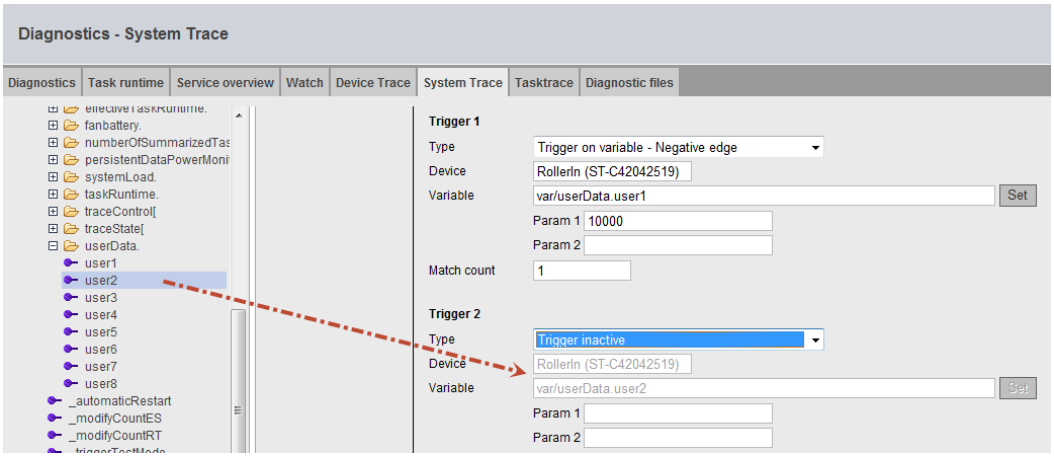


Figure 6-45 System Trace drag-and-drop

See also

Device Trace (Page 3595)

Trace Viewer

Trace data display

The Trace viewer page displays previously recorded trace data as a curve diagram.

To use this page you require a current browser version (Internet Explorer as of Version 10, Firefox, Chrome).

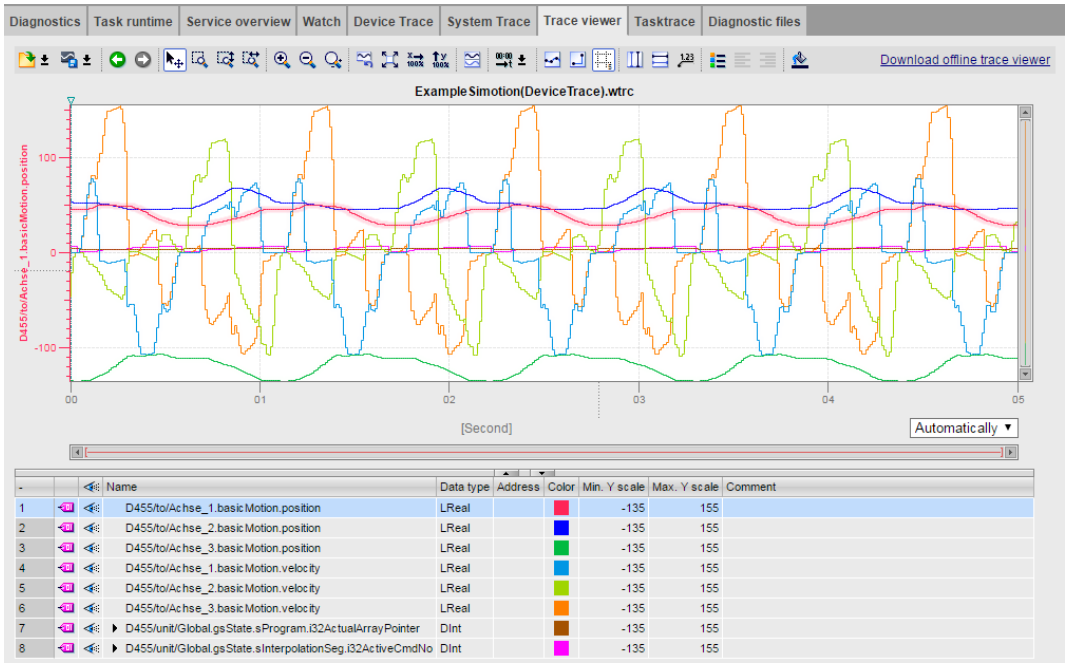


Figure 6-46 Trace viewer

The Trace Viewer shows only completed measurements. The measurements can be loaded from the device or a stored file.

The Trace viewer is described in detail in the Trace Viewer (Page 3709) chapter.

Tasktrace

Tasktrace

This page enables you to configure and control the SIMOTION Tasktrace (including trigger conditions).

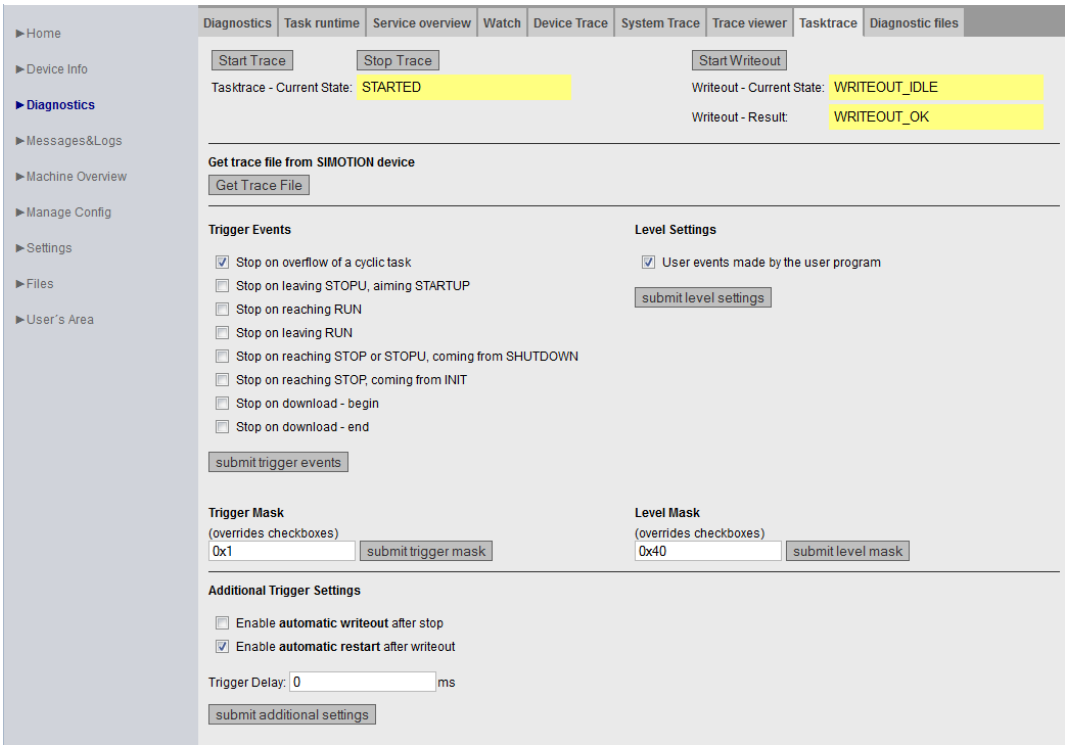


Figure 6-47 Tasktrace

The Tasktrace provides a diagnostics option during runtime which can be used to obtain reliable information about the processes in the individual tasks (e.g. task change).

The trace recording is continuously written to a ring buffer.

Once underway, a trace recording can be stopped manually or held conditionally by a trigger event. The recording can then be loaded to the PC and displayed with the Task Profiler with the **Get Trace File** button.

Start Trace

The **Start Trace**button starts the Tasktrace with the settings that have been made previously and have been transferred to the device using **Submit**.

Stop Trace

The trace is stopped manually with the **Stop Trace** button.

The state of the trace is displayed in the **Tasktrace - Current State:** field.

Start Writeout

The **Start Writeout** button writes the content of the trace buffer to the "/USER/SIMOTION/HMI/SYSLOG/TASKTRACE/DIAG/TTRACE.JEN" file on the device.

The state of the write process is displayed in the **Writeout - Current State:** and **Writeout - Result:** fields.

Get Trace File

The **Get Trace File** button loads the TTrace.jen file to the PC and displays with the TaskProfiler program. The setup of the TaskProfiler can be found in the tasktrace_viewer.zip file in the SCOUT setup directory \scout\release\VOL1\InstData\SCOUT\Media\.

Trigger Events

The **Trigger Events** can be selected and combined as you wish using various checkboxes. The **submit trigger events** button transfers the selection to the device.

Trigger Mask

The **Trigger Mask** input field enables the expert to input **Trigger Events** as coded number. The **submit trigger mask** button transfers the input to the device and overwrites all previous inputs.

Level Settings / Level Mask

You can use these settings to determine which events are entered in the Tasktrace.

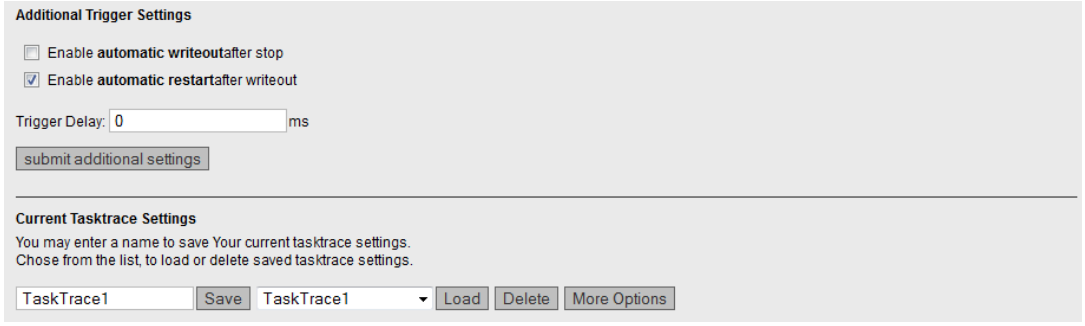


Figure 6-48 Tasktrace Additional Settings

Additional Trigger Settings

These settings enable you to back up a trace automatically.

- **Enable automatic writeout after stop**: The trace data is automatically backed up after the occurrence of a trigger event.
- **Enable automatic restart after writeout**: The trace is restarted after backing up the trace data.

Trigger Delay sets the time during which the trace remains active after a trigger condition occurs.

Current Tasktrace Settings

You can back up, load or delete a setting here.

Saving the trace settings

The current trace settings can be saved in the XML file "/USER/SIMOTION/HMI/FILES/PERSIST/TTRACE.XML" on the storage medium of the controller. This file is evaluated during startup. As a result, it is also possible to activate the trace of system function calls from the Web user interface. In addition, the Web server allows you to delete this file.

Diagnostic files

Backing up diagnostic pages of the Web server

You can use this page to back up general diagnostic data and individual HTML pages of SIMOTION IT .

The standard HTML pages of the Web server contain valuable information for analyzing problems that can occur during operation of the SIMOTION controller.

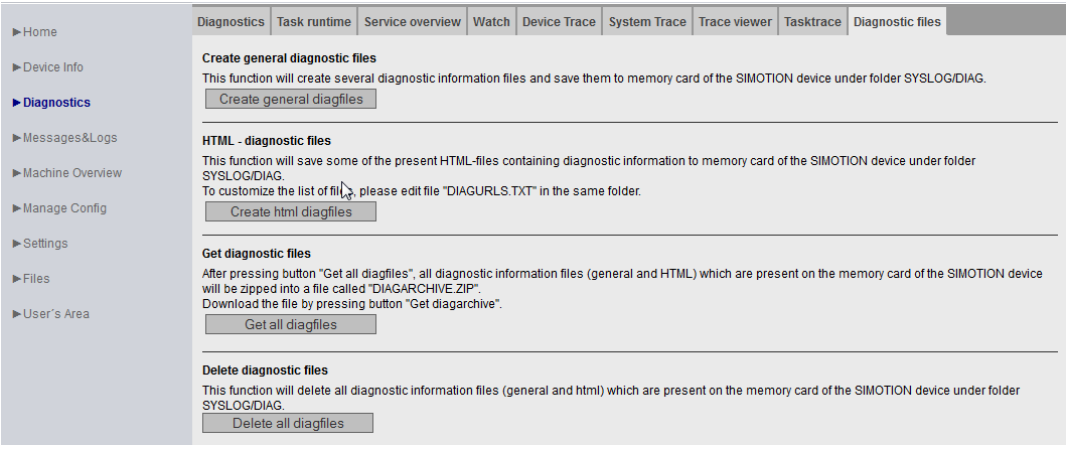


Figure 6-49 Diagnostic files

Create general diagnostic files
This function saves diagnostic data for Support.

SIMOTION device	Storage medium	Path
D, C	CF card / MMC	\USER\SIMOTION\HMI\SYSLOG\DIAG
P350	Hard disk	F:\Simotion\user\Card\USER\SIMOTION\HMI\SYSLOG\DIAG
P320	CF card	D:\Card\USER\SIMOTION\HMI\SYSLOG\DIAG
P320-4 E P320-4 S	External CFast card	D:\USER\SIMOTION\HMI\SYSLOG\DIAG

Directory paths for saving the diagnostic data

This function corresponds, for example, to actuating the service selector switch on the SIMOTION D controller.
HTML files used for diagnostics purposes are not saved.

HTML - diagnostic files
A selection of relevant diagnostic pages are saved on the data medium as HTML pages. You can use the DIAGURLS.TXT file to control which HTML pages will be saved.

Get diagnostic files
A selection of relevant diagnostic pages are saved on and loaded from the DIAGARCHIVE.ZIP data medium as HTML pages.

Delete all diagfiles
Deletes all diagnostic files present in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory. The directory itself is retained.

Messages&Logs

Diag buffer

Diagnostic buffer information

On the **Diag buffer** page (opened via **Messages&Logs > Diag buffer**), you can view the latest content of the controller diagnostic buffer.

Time	Time of the event
Date	Date of the event
Event	Displays the event as text. If the DGBUFTXT.EDB language file is missing, the texts are displayed in English. English texts are pre-installed on the device.

Note

The text is displayed in English by default. To display the Event text in a different language, you must transfer the relevant language versions of the DGBUFTXT-XX.EDB, DGEXTTXT.EDB and TOALARM.ADB files to the .../USER/SIMOTION/HMICFG directory on the SIMOTION controller memory card. A detailed description of this topic is available in the DiagBuffer group (Page 3690) and Alarms (Page 3609) sections.

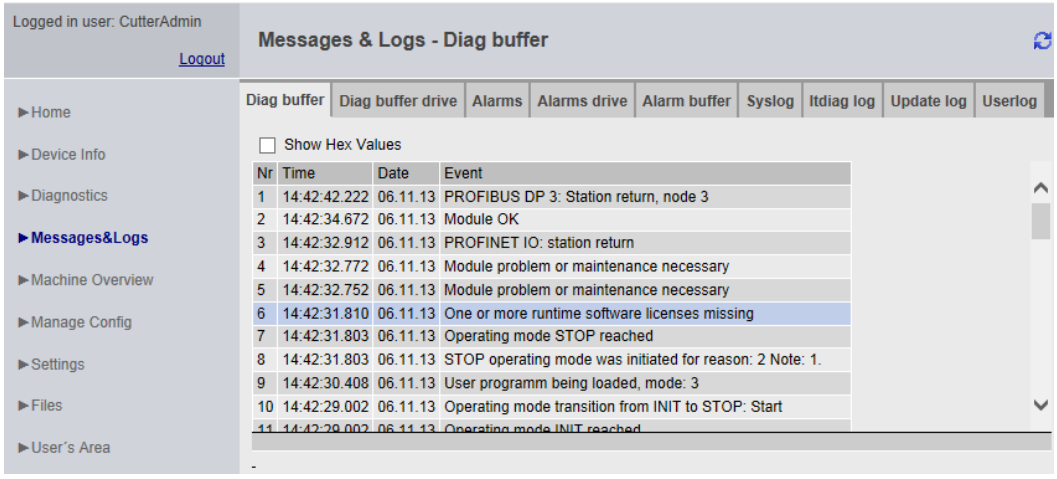


Figure 6-50 Diag buffer

Diag buffer drive

Representation of the drive diagnostic buffer

Like the SIMOTION diagnostic buffer, there is also a diagnostic buffer for the integrated drives only for the SIMOTION D.

Time	Time of the event
Date	Date of the event
Event	Displays the event as text. If the DGEXTTXT.EDB language file is missing, the display is in English. English texts are pre-installed on the device.

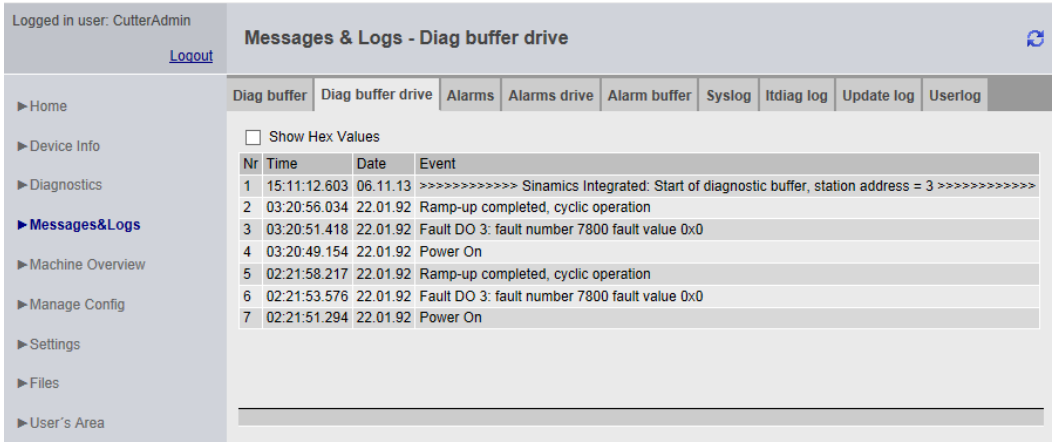


Figure 6-51 Display of the diagnostics buffer for the integrated drives

The diagnostics buffer of a CX (Controller Extension) module is displayed in this way.

Alarms

Information about alarms

The alarm and AlarmS/SQ messages of the device are displayed on the **Alarms** page.

Technological Alarms

Level	Category of the alarm
Time	Time of the alarm
TO	Technology object that triggered the alarm
Alarm	Alarm number
Text	Message displayed as text

Process Alarms (AlarmS/SQ)

AlarmNo	Number of the AlarmS/SQ
State	Status of the AlarmS/SQ
Time	The time when the AlarmS/SQ occurred.
Type	Type of the AlarmS/SQ
Text	Message displayed as text (message text)
More Info	Additional information (Infotext)

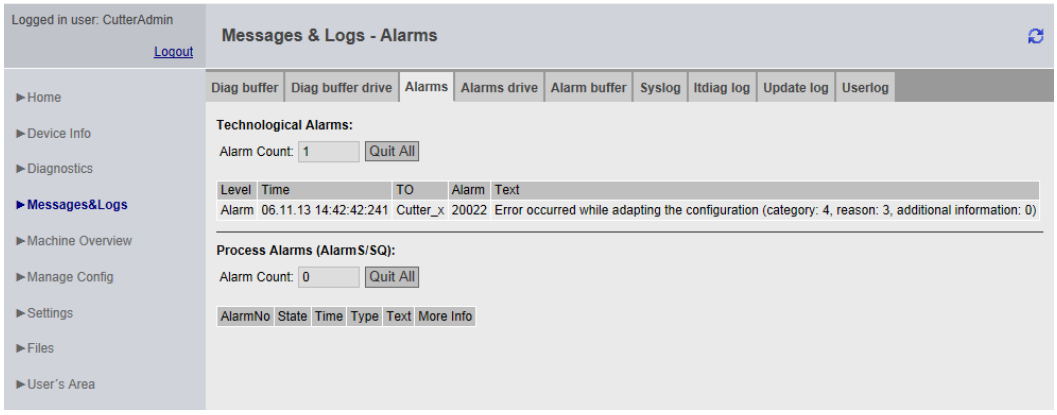


Figure 6-52 Alarms

The **Quit All** button allows you to close all alarms requiring acknowledgment.

Language setting of the technological alarms text

Alarm texts are displayed in English by default. To display the technological alarm texts in a different language, you must transfer the TOALARM.ADB file in the relevant language to the SIMOTION controller memory card.

Only one language can be saved in SIMOTION at a time.

Procedure SIMOTION D,C

1. Open the AddOn\4_Accessories\SIMOTION_IT\4_Alarm_Messages\< directory version >\ on the SIMOTION SCOUT Add-Ons DVD. For the language, you can choose between ger (German), eng (English), ita (Italian) and fra (French). The TOALARM.ADB file is located in the corresponding directory.
2. Insert the SIMOTION memory card in a reader/writer.
3. Copy the TOALARM.ADB file to the \USER\SIMOTION\HMICFG directory. You must create the directory if it does not yet exist.
4. Insert the memory card in the SIMOTION device again.

Procedure SIMOTION P350, P320

1. Shut down the SIMOTION P.
2. Open the \AddOn\4_Accessories\SIMOTION_IT\4_Alarm_Messages\< directory version >\ on the SIMOTION SCOUT Add-Ons DVD. For the language, you can choose between ger (German), eng (English), ita (Italian) and fra (French). The TOALARM.ADB file is located in the corresponding directory.
3. Copy the TOALARM.ADB file to the F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG directory (for the P350 default installation) or theD:\CARD\USER\SIMOTION\HMICFG directory (for the P320 default installation).
4. Start the SIMOTION P.

Procedure SIMOTION P320-4 E, P320-4 S

1. Shut down the SIMOTION P.
2. Open the \AddOn\4_Accessories\SIMOTION_IT\4_Alarm_Messages\< directory version >\ on the SIMOTION SCOUT Add-Ons DVD. For the language, you can choose between ger (German), eng (English), ita (Italian) and fra (French). The TOALARM.ADB file is located in the corresponding directory.
3. Copy the TOALARM.ADB file to the D:\USER\SIMOTION\HMICFG directory.
4. Start the SIMOTION P.

To have the correct path, replace the <version> with the SIMOTION version, such as V4.5.

Note

Data type and output format

When using the associated values (process values) of the alarms, ensure that the output format matches the data type of the value. Otherwise, errors may occur when displaying the values on the Web pages. Unlike SIMOTION SCOUT or TIA Portal, the Web server does not have any information on the data type of the variables used.

Additional notes are provided in the 'Syntax for process values in message texts' section in the SCOUT documentation.

Example

"My Alarm Message with LREAL Value: @1O%10d@" => incorrect display on the Web pages, as the LREAL value is to be output as decimal number (%d).

"My Alarm Message with LREAL Value: @1O%10.2f@" => correct display on the Web pages, as the LREAL value is to be output as floating-point number (%f).

See also

SIMOTION IT Text Databases (Page 3636)

Alarms drive

Drive faults and warnings

Similar to the technological alarms of the controller, a page containing fault and warning messages of the drive is also available. As of V5.1, the alarm texts for the drive alarms are available.

The following are displayed:

Time	Fault time
Type	Error type
Source	DO name
No.	Fault code
Value	Fault value

If DOs (Drive Objects) are present in the device by name, they are also output by name.

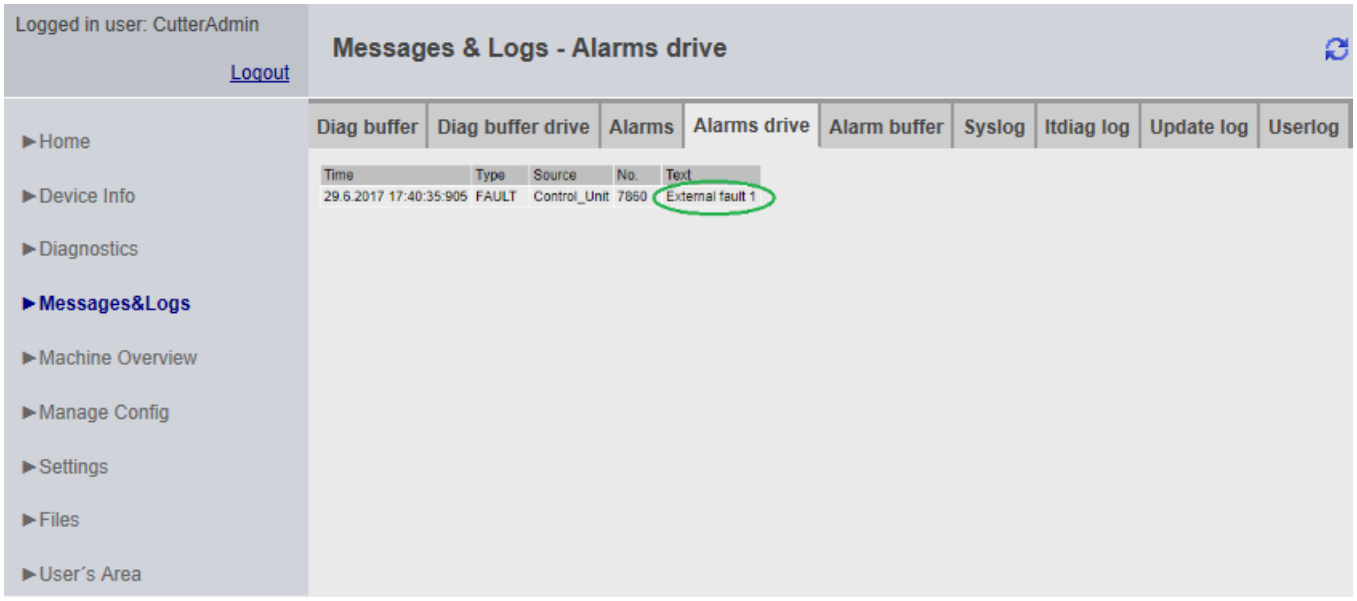


Figure 6-53 DriveAlarms

The drive alarms for the CX32/CX32-2 controller extension and external CUs can also be displayed.

Alarm buffer

Contents of the alarm buffer

On the **Alarm buffer** page, you can view the following information:

Index	Numbering of entry
Time	Time of the alarm
TO	Instance of the technology object
Alarm	Alarm number
Text	

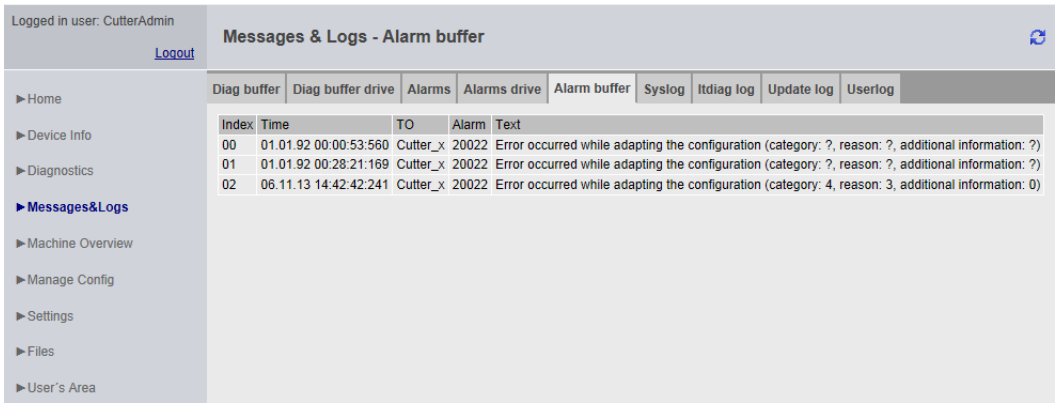


Figure 6-54 Display of the alarm buffer

In contrast to the **Alarms** page, which shows the alarms that are currently pending, the **Alarm buffer** page shows a history of all the alarms.

Syslog

Syslog

The **Syslog** page displays the syslog file for the relevant device.

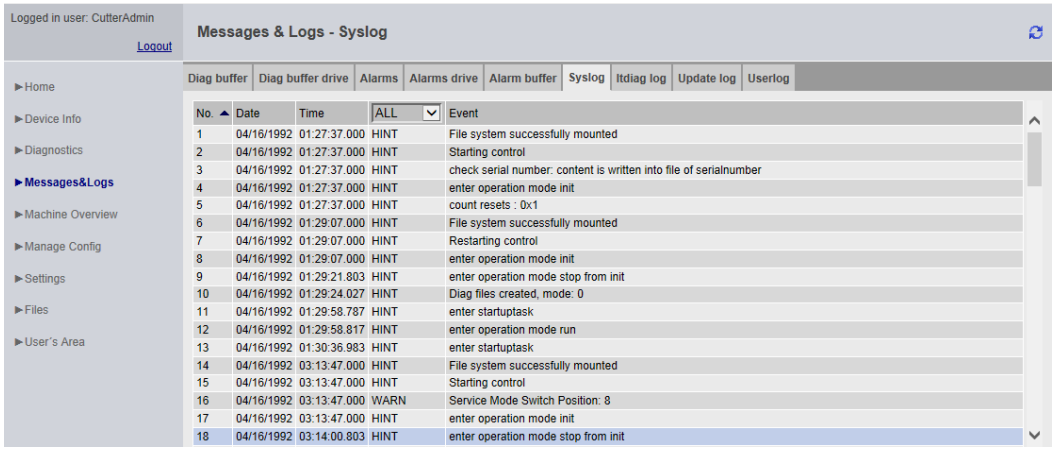


Figure 6-55 Syslog

This file is maintained by the system. Events that are important for diagnostic purposes are documented, such as RAM2ROM. When you start the page, all events are displayed. On the title page of the table, you can limit the display by deselecting **ALL**.

Itdiag log

Itdiag log

The messages from SIMOTION IT are output on the **Itdiag log** page.

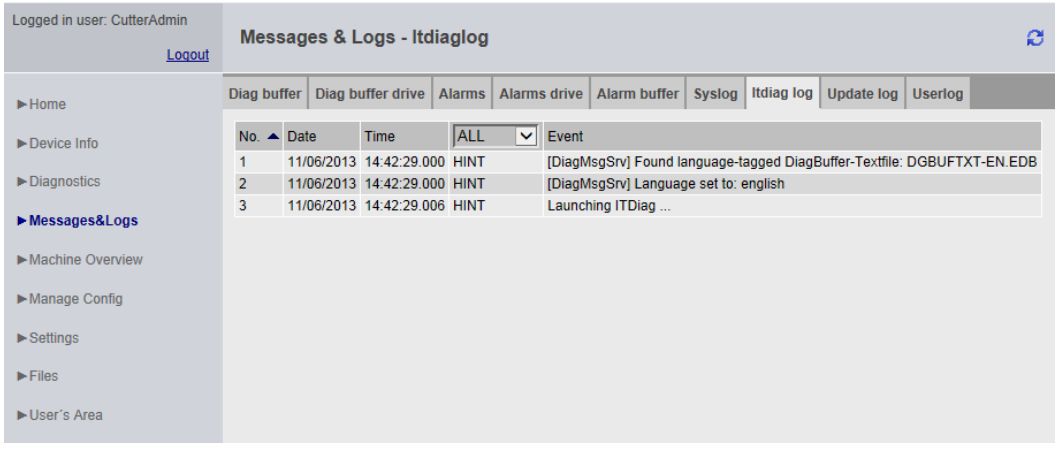


Figure 6-56 Itdiag log

SIMOTION IT-specific log outputs are displayed on this page.

Update log

Update log

The download and upload messages are displayed on the **Update log** page.

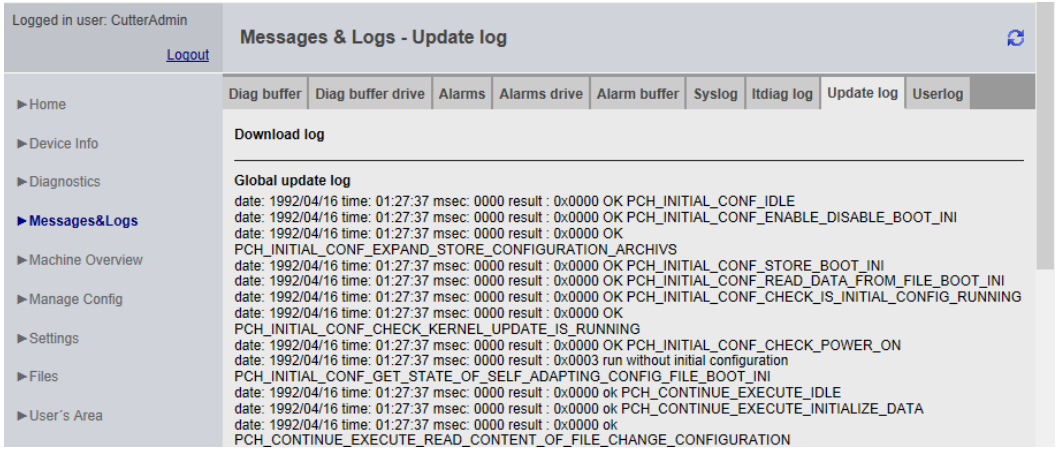


Figure 6-57 Update log

Messages that are generated during the project update are displayed on the Update log page.

Userlog

Userlog

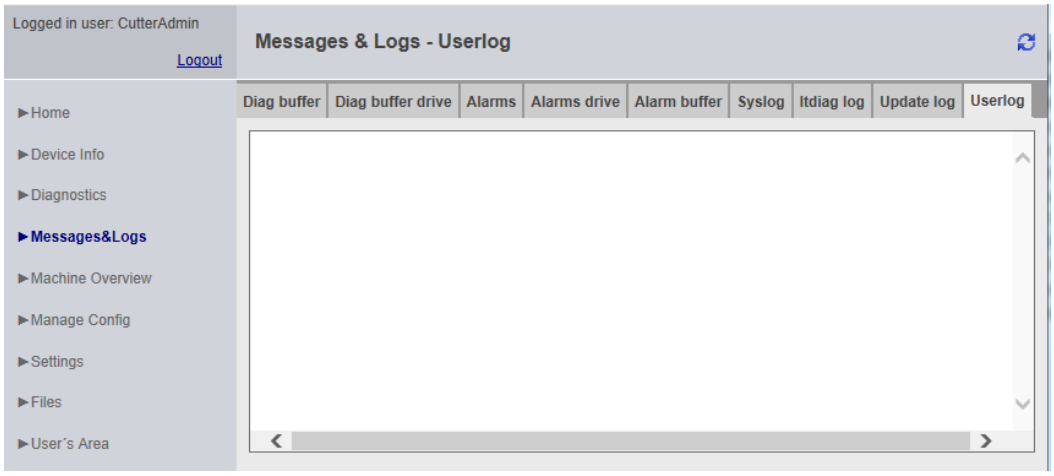


Figure 6-58 Userlog

The Userlog shows free texts entered by users in SIMOTION SCOUT (**Device Diagnostics** > **Userlog**). The texts are saved in a file on the memory medium of the control and displayed on the web page (in read-only format).

Machine Overview

Module information

Overview of configured modules

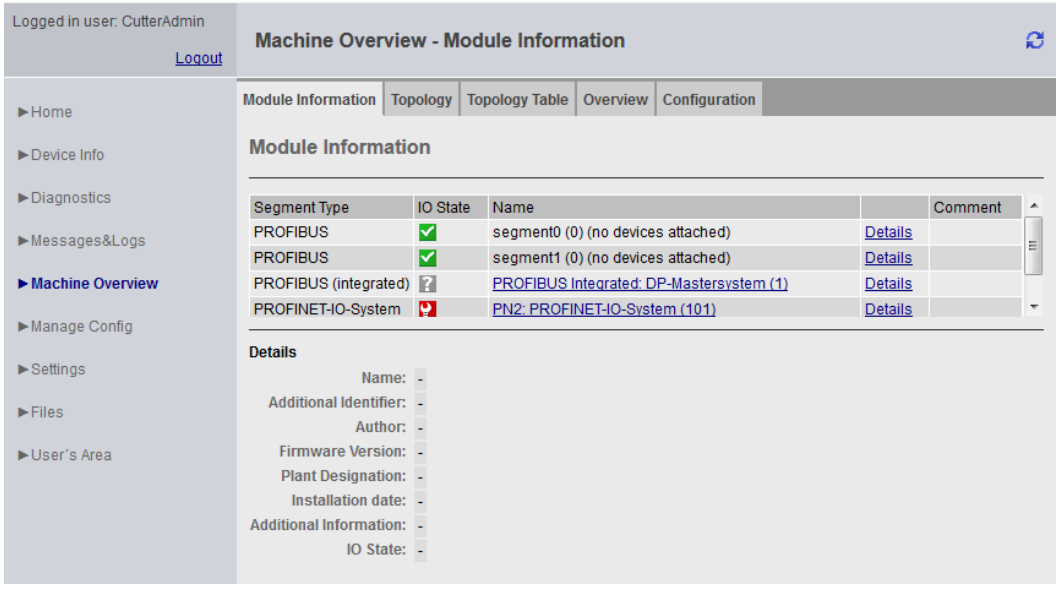


Figure 6-59 Module information

Overview of all modules configured on the machinery. Starting from the segment, you can navigate hierarchically to the element and call up information about it.

Note

For a correct representation of the information contained in the pages of the **Machine Overview**, it is necessary to load an HW Config in SIMOTION IT. The loaded HW Config must match the loaded SCOUT project, otherwise incorrect information is displayed.
See Configuration (Page 3620)

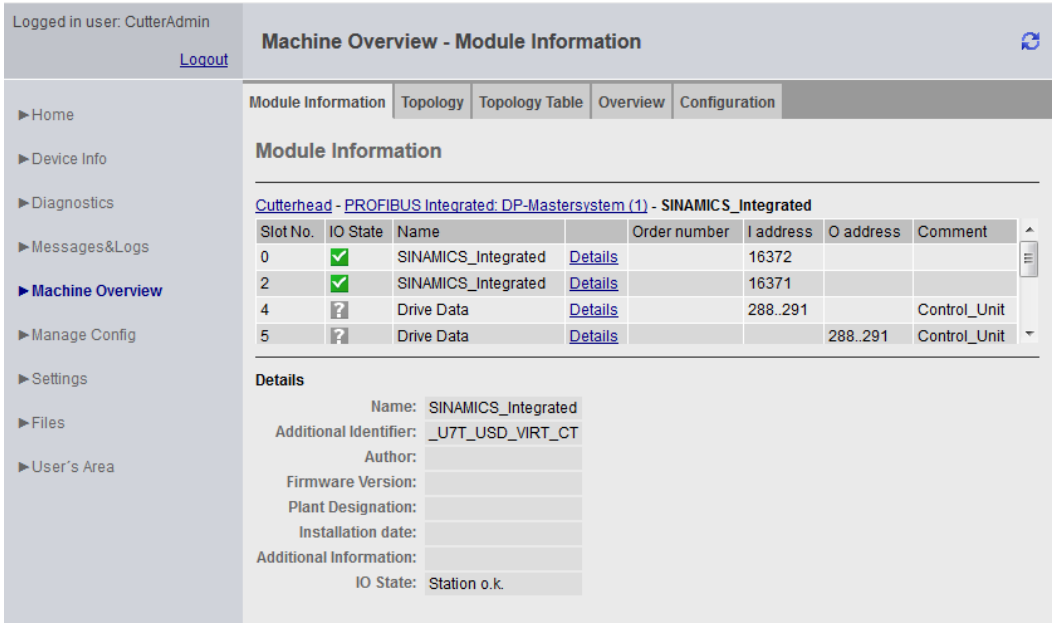


Figure 6-60 Module information - detail information

The hierarchy is always as follows: Segment > Device > Slot > Subslot (if present). Elements without subelements are not clickable.

Clicking on the segment displays all of the devices in the segment (PROFIBUS Integrated: DP-Mastersystem (1)).

Clicking on **Details** displays further information at the bottom (SINAMICS_Integrated).

Links allow you to jump back to the previously selected elements (breadcrumbs).

Topology

Overview of the configured topology

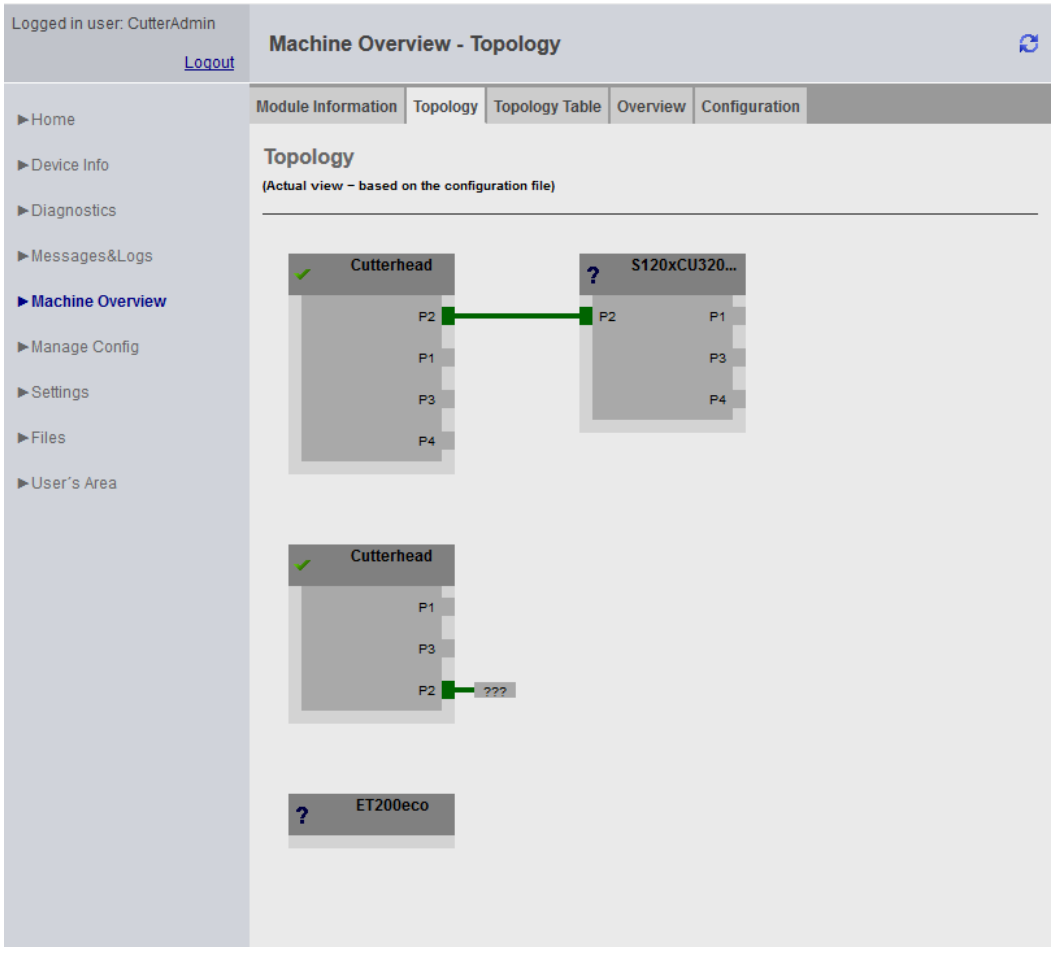


Figure 6-61 Topology of the device

The configured topology of a device is depicted on this page. Inaccessible nodes are highlighted in red.

The topology display shows how the nodes must be wired.

Topology table

Tabular overview of the configured topology

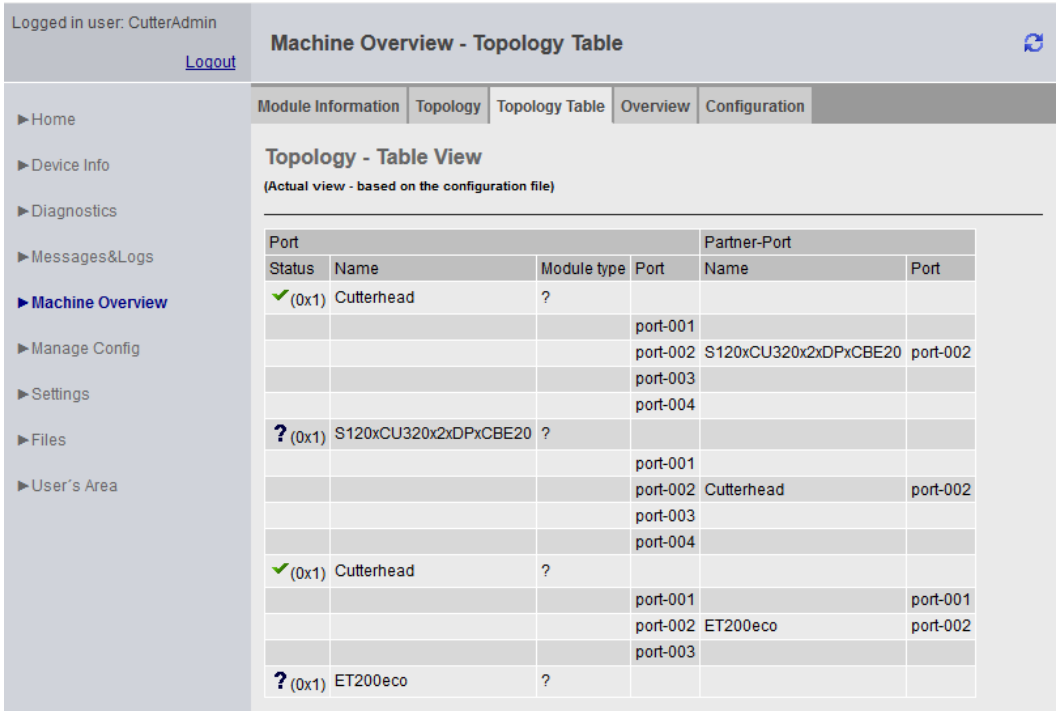


Figure 6-62 Tabular topology table

This page offers a quick overview of the wiring in text form.

The information displayed corresponds to that of the topology (Page 3618) page.

Overview

Overview of all modules configured on the network

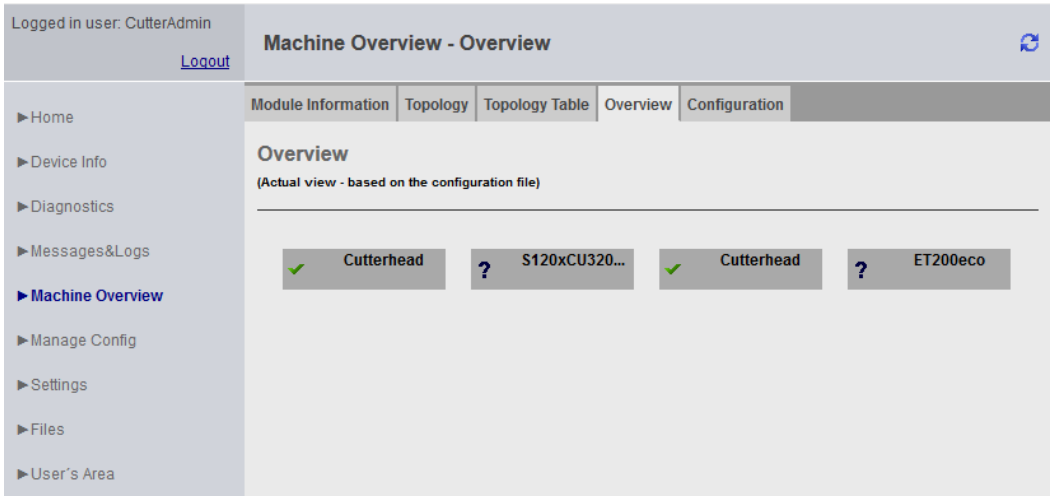


Figure 6-63 Overview

This overview displays all modules configured on the network without topology information. This overview is primarily intended for very large projects.

Inaccessible or failed nodes are shown in red.

Configuration

Downloading a configuration

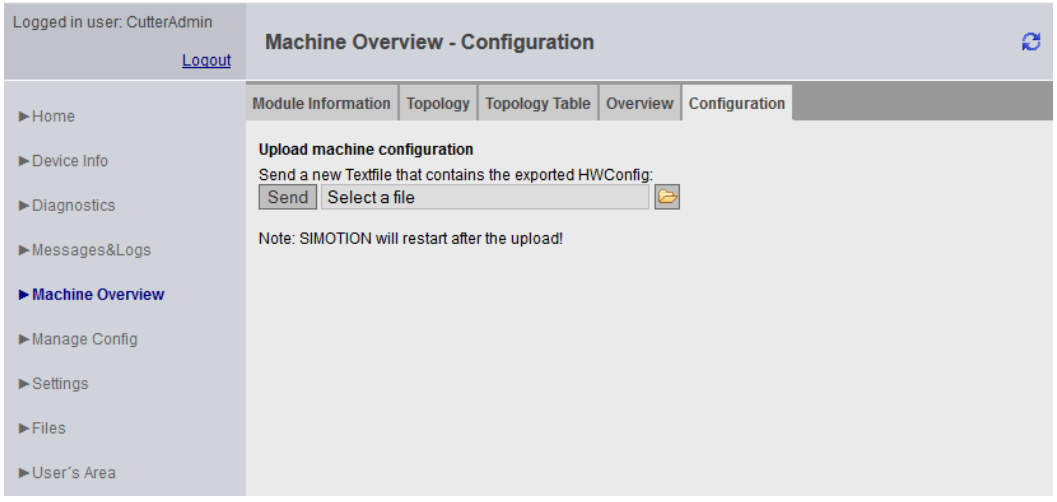


Figure 6-64 Configuration

Downloading HW Config information to SIMOTION IT

An HW Config export file must be loaded to SIMOTION IT. The texts and designations of the installed modules are only present once this has been done. The controller must be in STOP operating state for this purpose.

The HW Config export file and the loaded SCOUT project must match, otherwise incorrect information is displayed.

Note

TIA Portal

The TIA Portal does not provide any possibility to load the HW Config data.

Exporting in HW Config

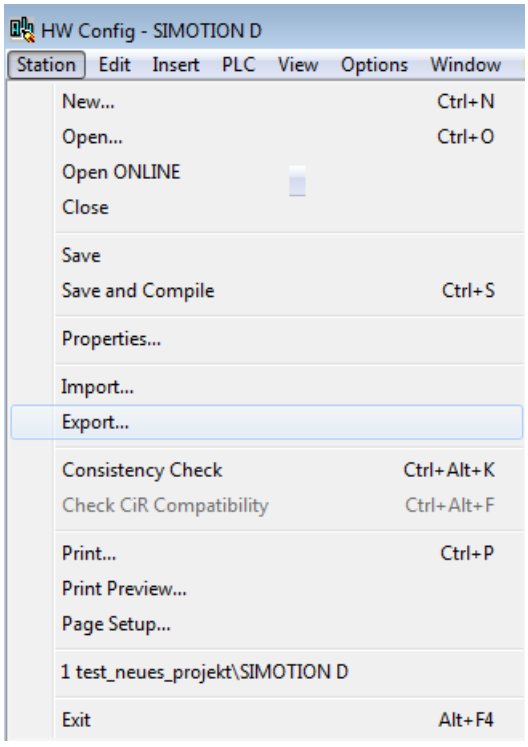


Figure 6-65 HW Config export

- Open HW Config
- Menu **Station Export**
- Save the file.
- The controller must be in STOP operating state.
- Load the resulting file using the form on the SIMOTION IT page.
- The SIMOTION controller subsequently performs a restart.

The file can then be found on the card in the directory /USER/SIMOTION/HMICFG/HWCONFIG.CFG

Alternatively, you can also directly copy the file to the card using a card reader.

⚠ WARNING

HW Config export file and SCOUT project

The HW Config export file and the loaded SCOUT project must match, otherwise incorrect information is displayed.

- If the HW Config is changed, the file must be reloaded.

Manage Config

Device update

Device update of the device

This page enables you to load a device update. The device update allows selected data to be saved to the PC from the device.

If several update archives have been written to the controller successively, you have the option of restoring a previous configuration.

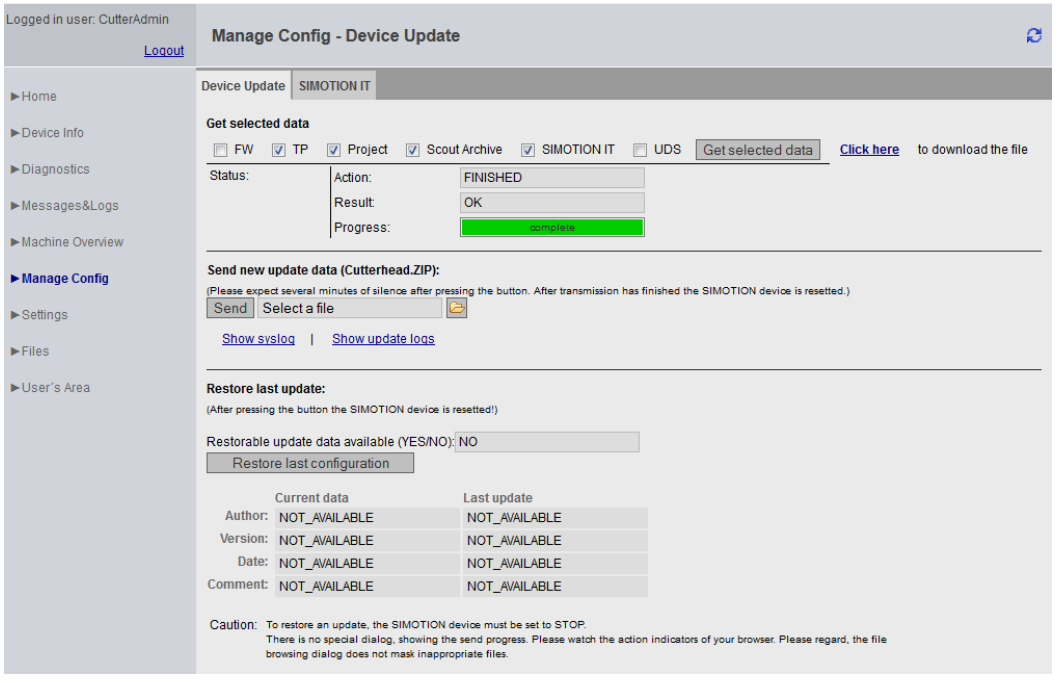


Figure 6-66 Manage Config - Device Update

- **Get selected data** transfers the currently active device data to the PC.
  The saved data has a format that allows it to be reimported to the device. To correctly save the project and the technology packages, a RAM2ROM must be performed in the SCOUT. Alternatively, the data can be saved directly in the file system on the card in the `\USER \SIMOTION\RT_DIR` directory.
  - **FW** (firmware)
  - **TP** (technology packages)
  - **Project** (current project)
  - **Scout Archive** (including the SCOUT backup)
  - **SIMOTION IT** (SIMOTION IT configuration)
  - **UDS** (including the **Unit Data Sets**)

  Note

  Transmission duration

  If the capacity utilization of the controller is very high at the cyclic levels, this operation may take some time. In individual cases, transmission times may be longer than 30 minutes.

- **Send new update data** transfers a file generated with the device update tool or SIMOTION IT **Get selected data** to the device.
  This process can take several minutes and restarts the device.

  Note

  No other SIMOTION IT pages may be called during the update. A progress bar shows how the update is progressing. Cancellation of the update is logged in the system log.

- **Restore last update** reactivates the last version of the device data of the preceding software update.

You will find more information on this topic in the "Updating SIMOTION Devices" Operating Instructions.

⚠ DANGER

The controller must be put into the STOP state.

To send or download a project or firmware, the controller must be switched to STOP state.

Type and contents of the file are not checked during transmission.

If an invalid configuration is used, the USER directory must be deleted from the memory card.

Note

SIMOTION P

The SIMOTION P controller does not support firmware download.

Note

Memory

If low-capacity cards (32 MB/64 MB) are used, problems may be encountered during the update due to insufficient memory space.

The amount of memory space required is determined by the size of the existing configuration plus that of the update.

Only users who have logged in can access this page. See Log-in administration (Page 3658)

Use of existing configuration data

Existing configuration data that was created with the SIMOTION SCOUT **Load to File System** function can continue to be loaded using SIMOTION IT.

The ZIP file generated by SCOUT as part of this process can be transferred to the device using **Send update data**.

Although the installation of an older version is generally possible via **Send update data**, the function is not envisaged for this purpose. If a previous version is installed, the operability of the device is not guaranteed. A manual customization of the configuration, the user database, etc. is required in this case.

Updating the firmware to V4.5

The update of the firmware to version 4.5 causes an existing WebCfg.xml to be converted to the new format. If a problem occurs during the migration, the error cause is noted in a diagnostic buffer entry.

Further information about the migration strategy is provided in the Structure of the webcfg.xml configuration data (Page 3657) section.

Updating the firmware to V4.4

When updating the firmware to Version 4.4, an attempt is made to convert the configuration file WebCfg.xml to the new format. A UserDataBase.xml is created and filled with the old WebCfg.xml.

The original file is renamed to WebCfg.xml.deprecated.

This conversion might fail for the following reasons:

1. The version of the WebCfg.xml is for firmware before V4.2, which cannot be updated.
2. An error occurred during an attempt to apply individual user settings.
3. The user administration UserDataBase.xml contains an invalid entry. If the user 'simotion' and the password 'simotion' are found, conversion will be canceled. An error message indicating this will then be placed in the diagnostic buffer.

If this error occurs, the configuration files have to be corrected manually.

Converting firmware from V4.4 to V4.3

When converting a module of SIMOTION V4.4 back down to firmware V4.3, the configuration file WebCfg.xml has to be converted because the formats are incompatible.

Control behavior

If the conversion archive does not contain WebCfg.xml, a check is made to see whether directory USER/SIMOTION/HMICFG contains a file WebCfg.xml.deprecated. The file is then restored.

If the file WebCfg.xml.deprecated does not exist, the WebCfg.xml file for SIMOTION V4.4 is deleted. The first time the module starts up with firmware SIMOTION V4.3, the associated Default WebCfg.xml is created.

Upgrading firmware prior to V4.2

A firmware update involving versions lower than Version 4.2 can result in the following: An old WebCfg.xml is retained on the device and and causes empty diagnostic pages to be displayed.

Option for avoiding this problem:

- Explicit deleting of WebCfg.xml in the /USER/SIMOTION/HMICFG directory.

After the next reset, a new WebCfg.xml is generated by the device. The old WebCfg.xml should be backed up first so that settings can be transferred from the old configuration to the new WebCfg.xml .

See also

Device update (Page 3622)

Upgrading firmware from V4.1 to V4.2

When upgrading the firmware from Version 4.1 to Version 4.2 or higher, WebCfg.xml must always be deleted. If WebCfg.xml is not deleted, the web pages will be incorrectly displayed.

Note

When upgrading from V4.2 to V4.3 or higher, this restriction is no longer valid. WebCfg.xml then no longer has to be deleted.

Editing function

Editing functions of the SIMOTION IT Standard pages

The WebCfg.xml and UserDataBase.xml configuration files can be edited on some standard pages via the browser. The editing functions are always structured in the same way and are explained in this section.

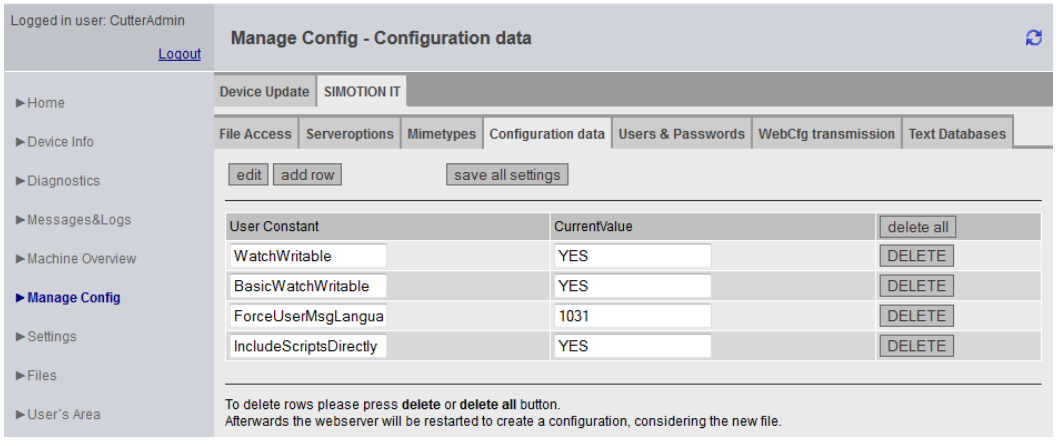


Figure 6-67 Editing functions

The **add row** button inserts one line.

To change the line, first click the **EDIT** button. The input fields can then be edited.

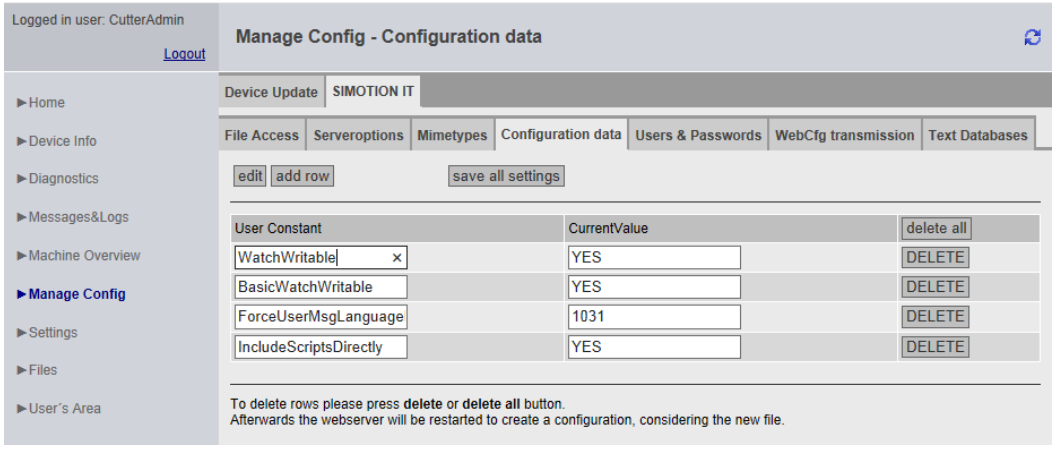


Figure 6-68 Editing functions input field active

The **DELETE** button deletes the inputs in the relevant line. The change is stored and the Web server is restarted immediately.

The **delete all** button deletes all lines.

The **save all settings** button saves all changes made on the controller.

SIMOTION IT tab

Web pages for making changes to the configuration

The SIMOTION IT tab summarizes the web pages that are used for configuring SIMOTION IT pages.

All changes under**Users & Passwords** are written to file UserDataBase.xml. All other tabs cause changes to the WebCfg.xml. As an alternative to editing using the web pages, changes can be made directly in these XML files.

SIMOTION IT File Access

Editing file and directory accesses

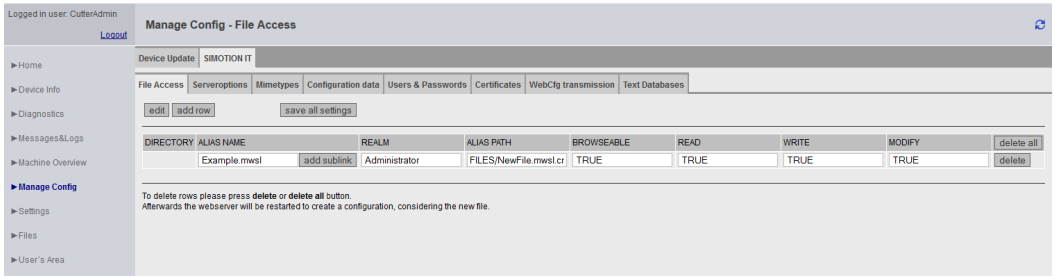


Figure 6-69 SIMOTION IT File Access

The tab File Access allows editing of file and directory accesses.

Attributes	Type	Example
ALIAS NAME	String	Example.mwsl
REALM	String	A group name: Administrator
ALIAS PATH	String	ALIAS="FILES/NewFile.mwsl.cms"
BROWSEABLE	true/false	
READ	String	One or more group names: Administrator,Servicegroup
WRITE	String	One or more group names: Administrator,Servicegroup
MODIFY	String	One or more group names: Anyone

Attribute overview tab File Access

See also

Configuration of the file system (Page 3665)

SIMOTION IT Serveroptions

Basic settings

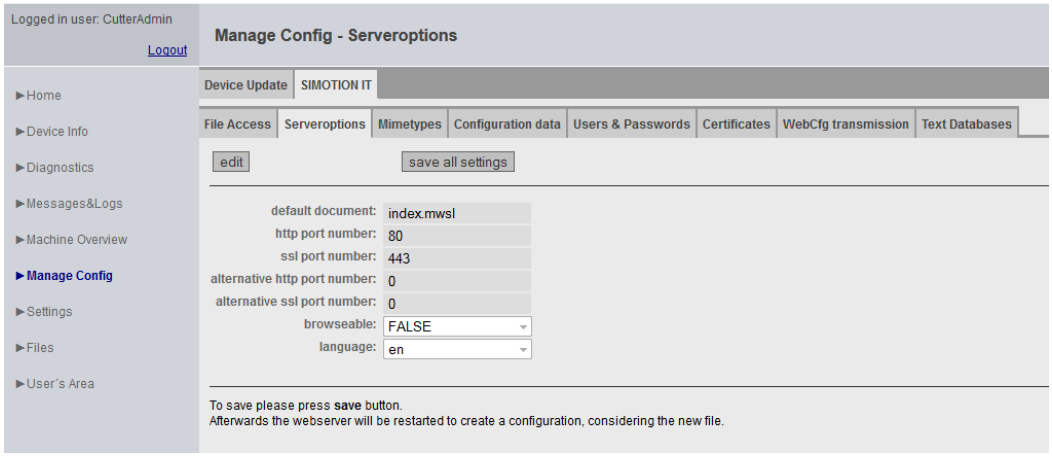


Figure 6-70 SIMOTION IT Serveroptions

This tab enables you to set basic parameters for the Web server.

Various settings for the `<SERVEROPTIONS>` tag in WebCfg.xml can be made on this page.

- <DEFAULTDOCUMENT> (Page 3728) enables you to change the home page. The default setting is `index.mwsl`.
- Default Port (Page 3730) defines the TCP/IP port for the output of the Web server pages. The default setting is port 80 (http).
- SSL Port (Page 3732) defines the TCP/IP port for the encrypted output of the Web server pages. The default setting is port 443 (https).
- Alternative Default Port (Page 3731) defines an additional TCP/IP port for the output of the Web server pages.
- Alternative SSL Port (Page 3733) defines an additional TCP/IP port for the encrypted output of the Web server pages.
- As of version 4.4, <BROWSEABLE> no longer has any effect.
- <LANGUAGE> saves the selected language in the WebCfg.xml in the `<LANGUAGE>` tag in the `VALUE` attribute. English language setting: `<LANGUAGE VALUE="en"/>`.

SIMOTION IT Mimetypes

MIME types

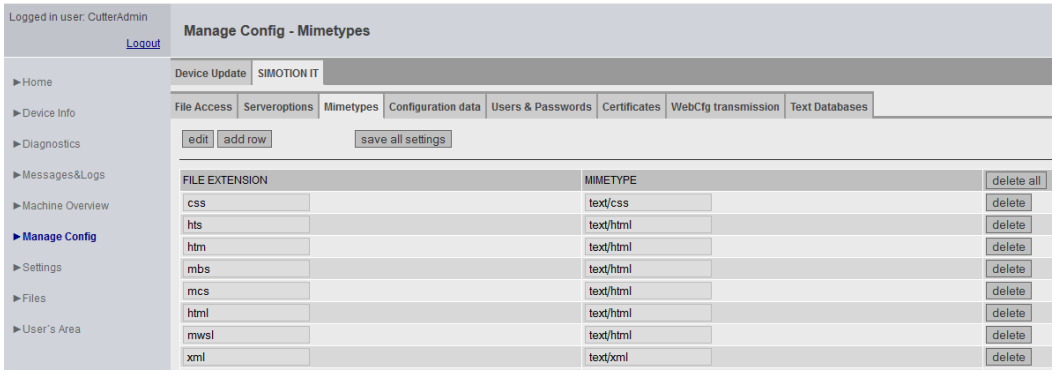


Figure 6-71 SIMOTION IT Mimetypes

A MIME type can be linked to a file extension on this tab.

The MIME type is used to signal to the browser, by means of the HTTP header, what type of data is being transmitted.

See also

<HEADER> (Page 3729)

SIMOTION IT Configuration data

Configuration of user-defined constants

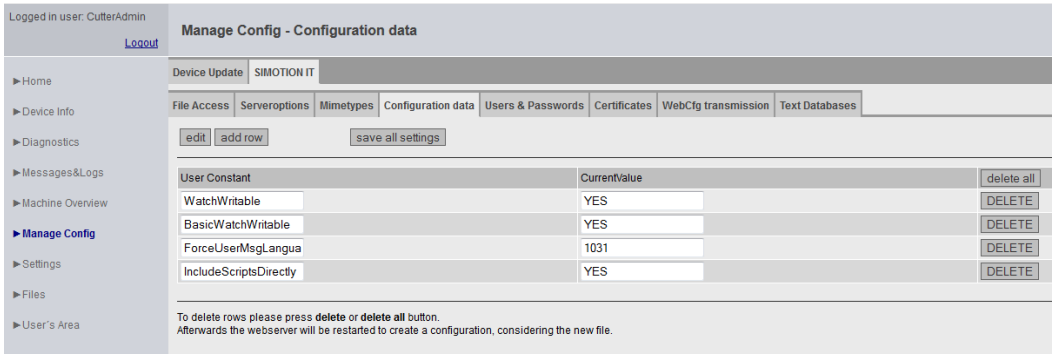


Figure 6-72 SIMOTION IT Configuration data

This page enables you to create and edit configuration constants.

See also

User-defined variables (Page 3698)

SIMOTION IT Users & Passwords

User database

The Users & Passwords page enables the user administration. Passwords, group rights, and access rights can be assigned to users here.

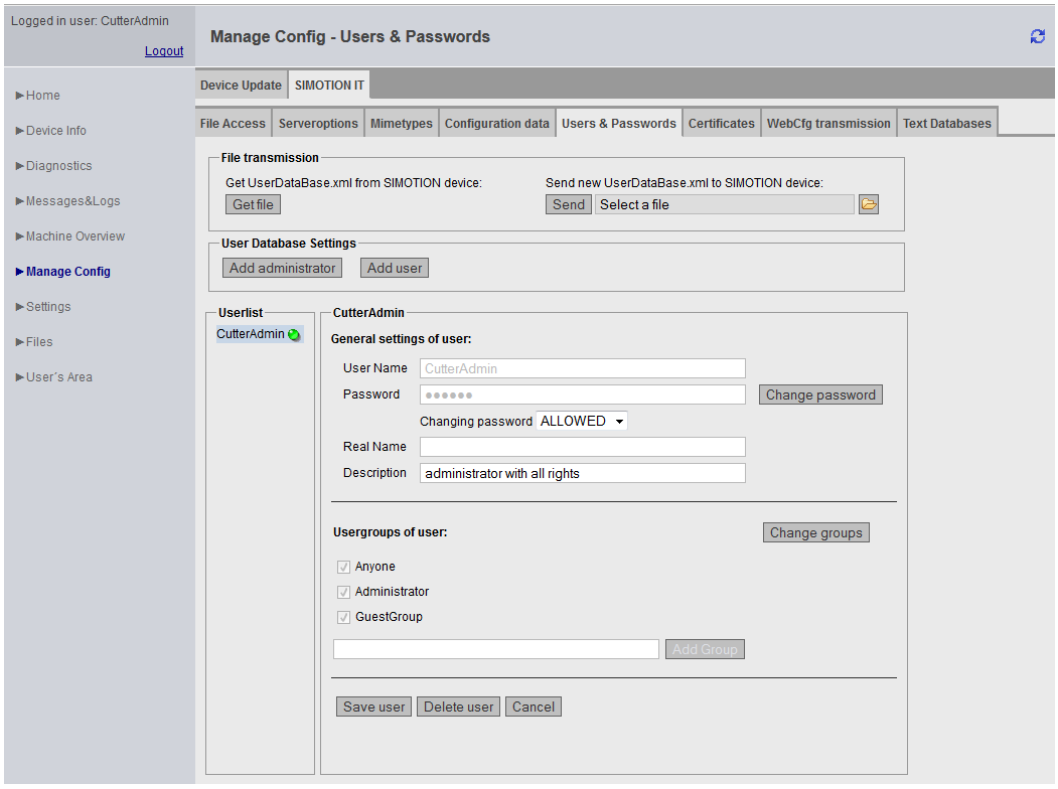


Figure 6-73 User database

File transmission

You can make a local backup of UserDataBase.xml of the controller with **Get file**. You can load a UserDataBase.xml onto the controller with **Send**.

Adding users

The **Add administrator** button creates administrators; the **Add user** button creates users.

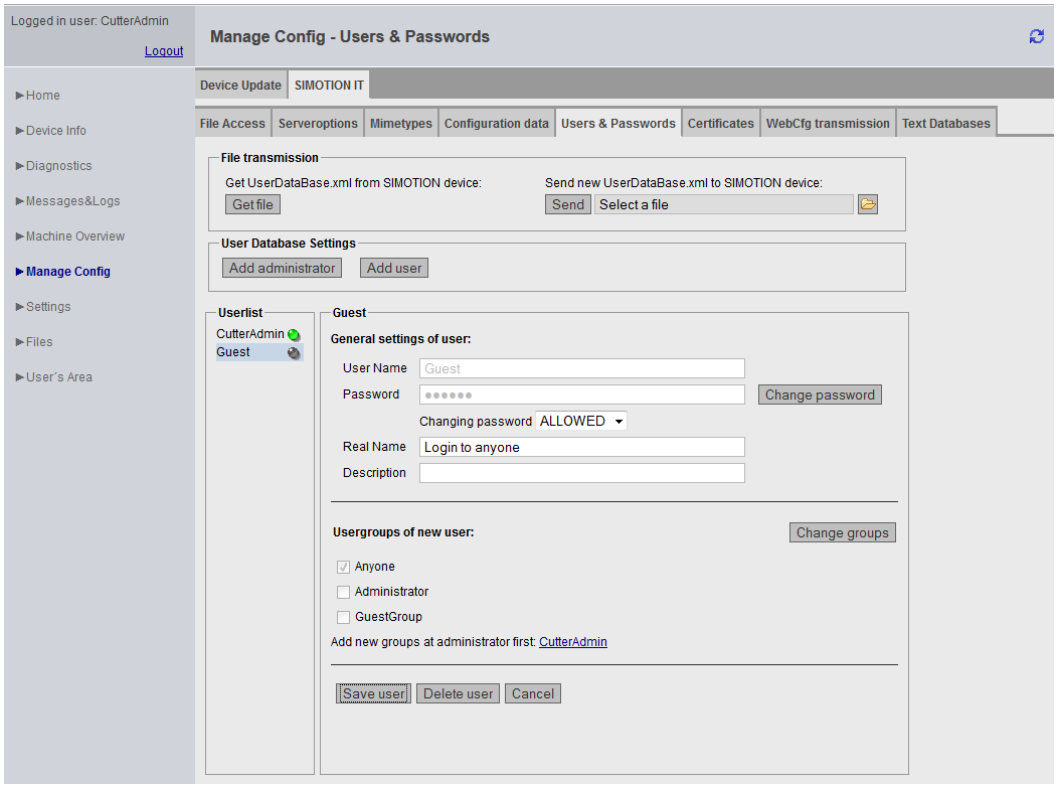


Figure 6-74 Benutzer Guest

This screenshot shows the situation after adding a user Guest who only belongs to the Anyone group.

Setting up a new group

A new group is only set up once the administrator has been assigned membership of that group. The **CutterAdmin** link in the above example opens the dialog box with the settings of the administrator CutterAdmin.

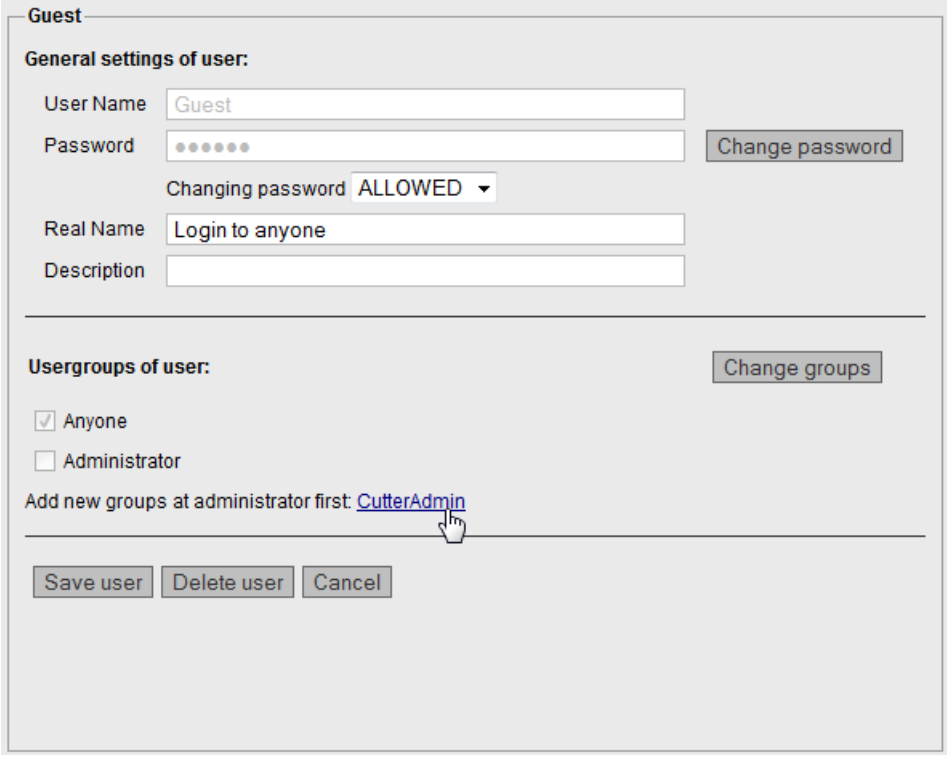


Figure 6-75 Creating a new group: Opening the administrator

The administrator can now create a new group with the **Add Group** button.

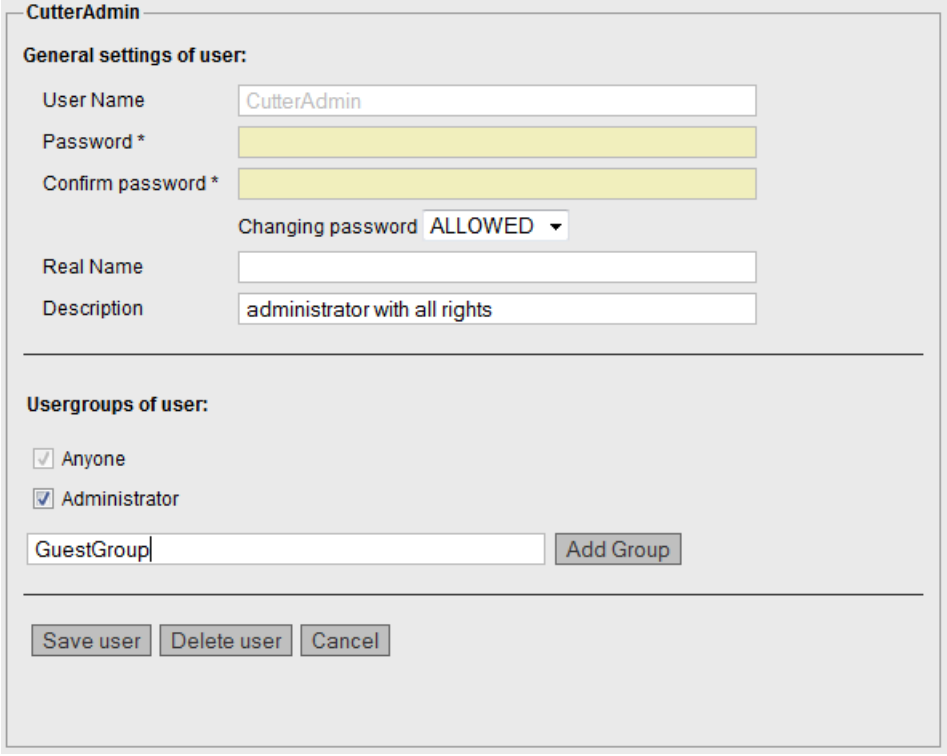


Figure 6-76 Creating a new group: Administrator creates the group

The new group GuestGroup can now be entered.

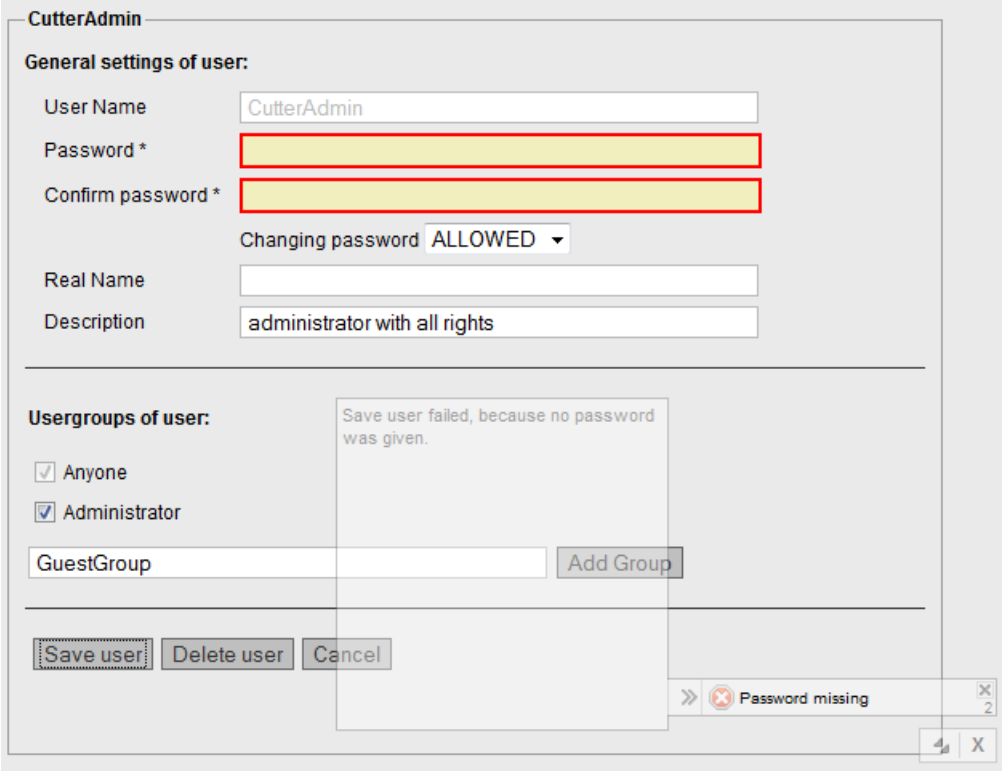


Figure 6-77 Creating a new group: Administrator password required

A new group can only be saved if the user is logged in as an administrator.

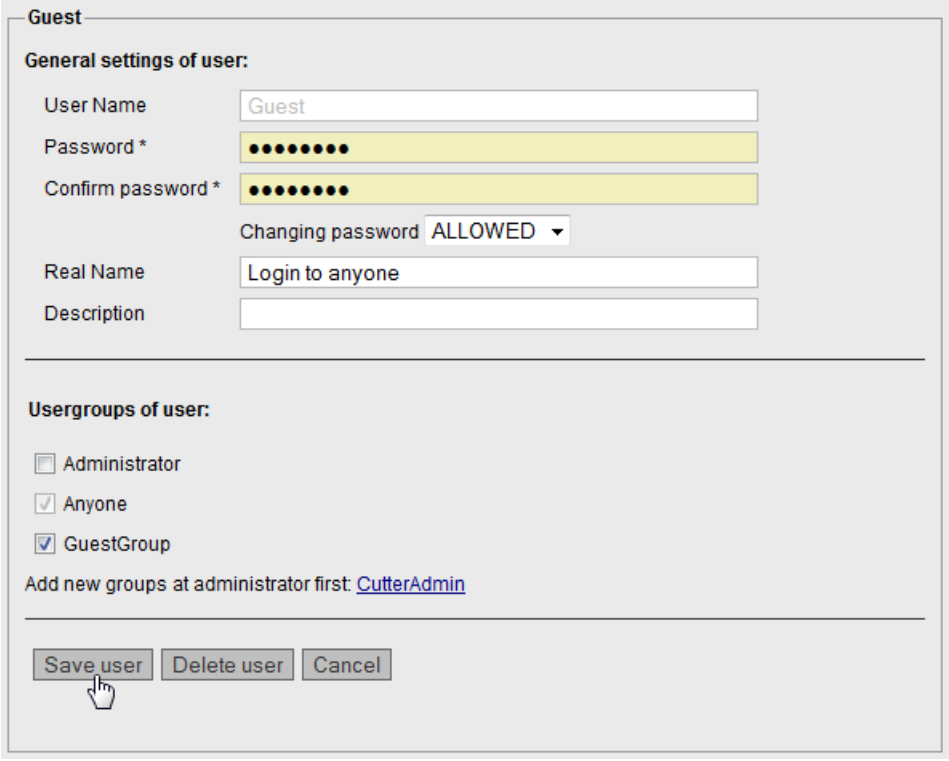


Figure 6-78 Creating a new group: Assigning a new group

Once the new group has been created, the group GuestGroup can be assigned to the user.

Note

Strong password

Only a strong password guarantees access to the device. The complexity and the length of the password must be appropriate for the type of data to be protected.

See also

Log-in administration (Page 3658)

SIMOTION IT Certificates

Uploading and downloading certificates

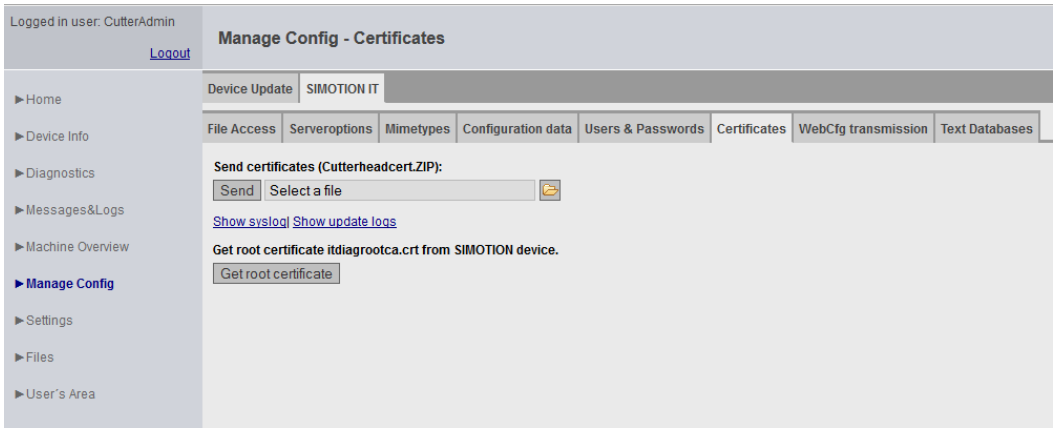


Figure 6-79 Certificates

The Certificates page enables certificates to be transferred to the controller. The ZIP file must have the same directory structure as is created when generating certificates with OpenSSL.

The **Get root certificate** button fetches the server certificate from the controller.

See also

Encryption methods (Page 3703)

SIMOTION IT WebCfg Transmission

Transferring configurations to the device

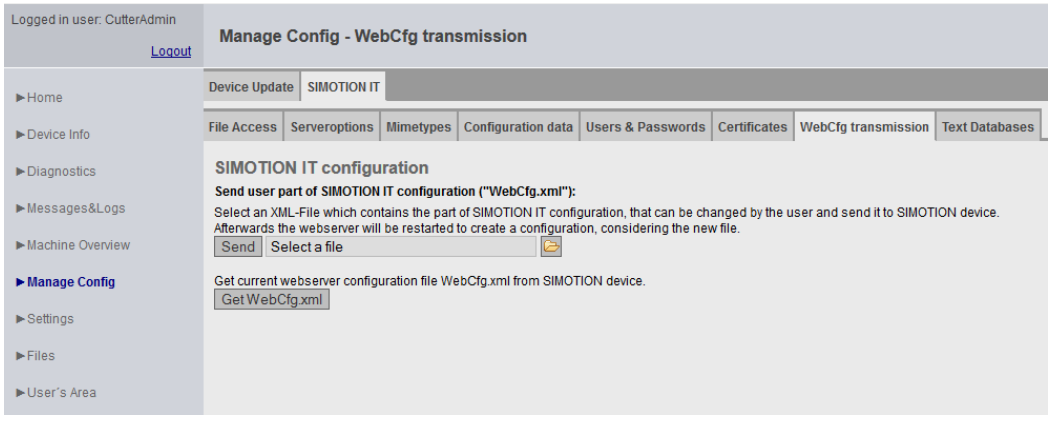


Figure 6-80 WebCfg transmission

The configuration data can be sent to or received by the device via this page.

The **Send** button transfers a locally edited WebCfg.xml to the device. As soon as the new WebCfg.xml has been sent, the Web server reboots and takes account of the new file.

SIMOTION IT Text Databases

Transmission of user-defined messages from SIMOTION SCOUT to the device

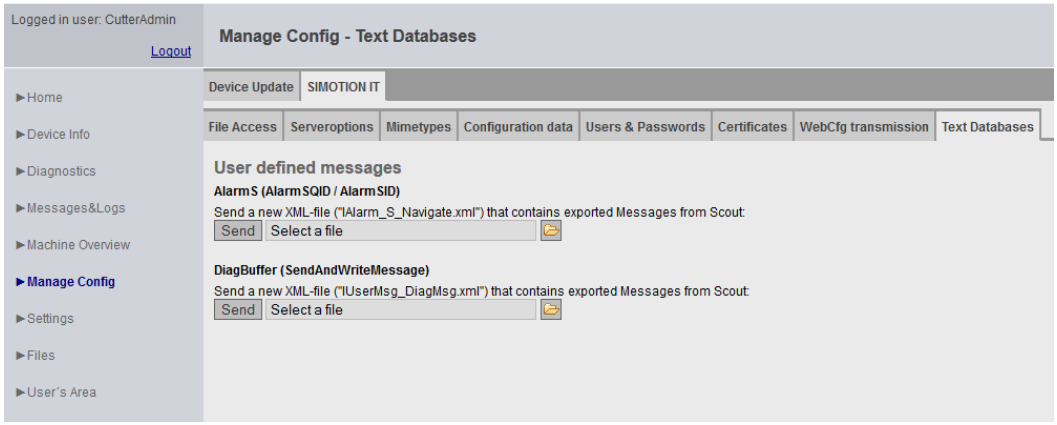


Figure 6-81 Text Databases

On this page, SIMOTION IT provides the option to transfer user-defined AlarmS and DiagBuffer messages, which have previously been exported into SIMOTION SCOUT, to the device.

For AlarmS, select the IAlarm_S_Navigate.xml file, and for DiagBuffer, select the IUserMsg_Navigate.xml file of a SIMOTION SCOUT language export. It is possible to select different languages for AlarmS and DiagBuffer messages.

Once the files have been transferred to the device, the messages exist in two files:

- dgusralarm.edb
- dgusrtxt.edb

in the /USER/SIMOTION/HMICFG directory. These files can be transferred to other controllers.

Language export from SIMOTION SCOUT

In SIMOTION SCOUT, the **Project > Language-dependent texts** and **Project > Messages** menu items enable export of user-defined messages.

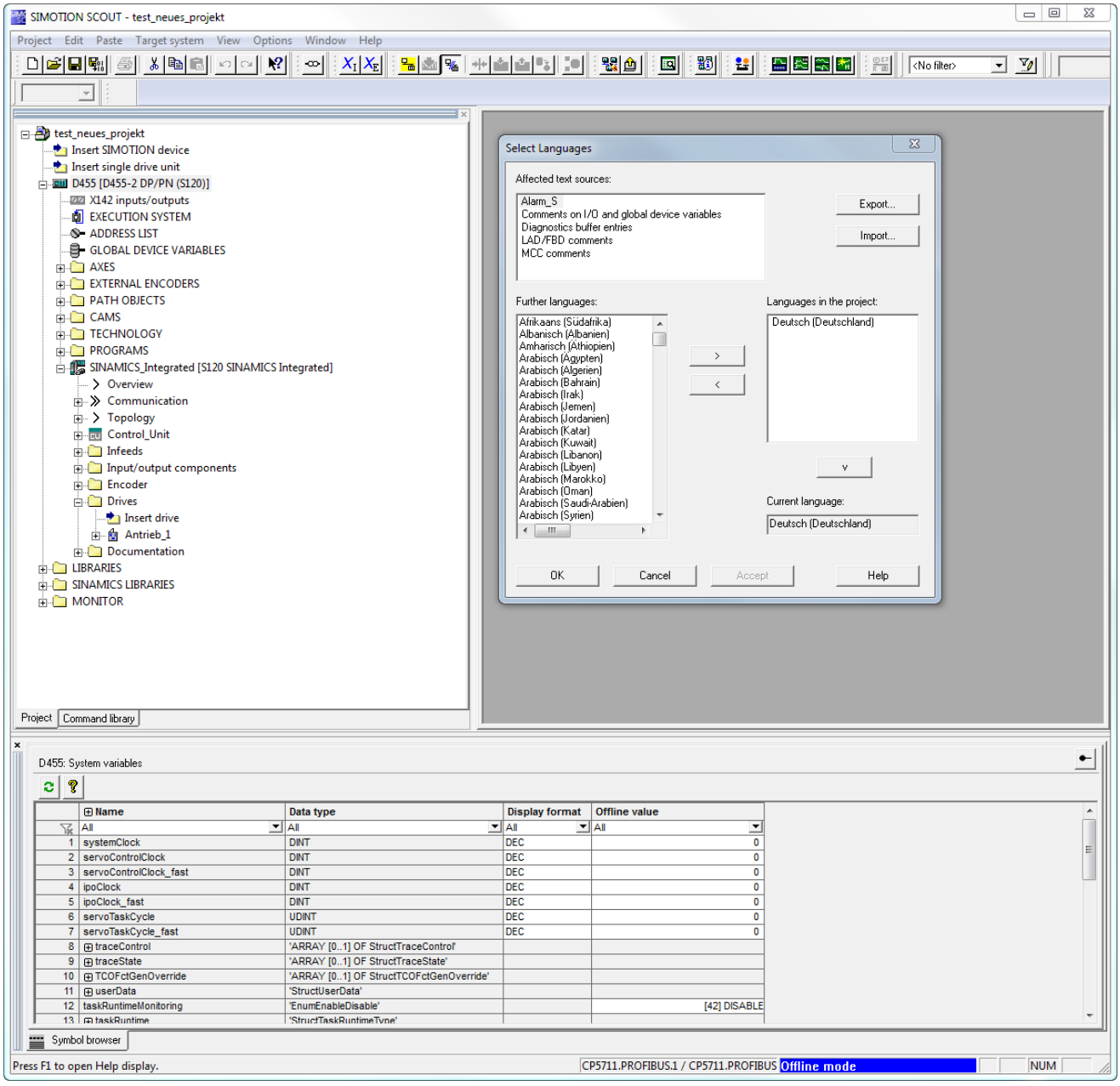


Figure 6-82 SIMOTION SCOUT language export language selection

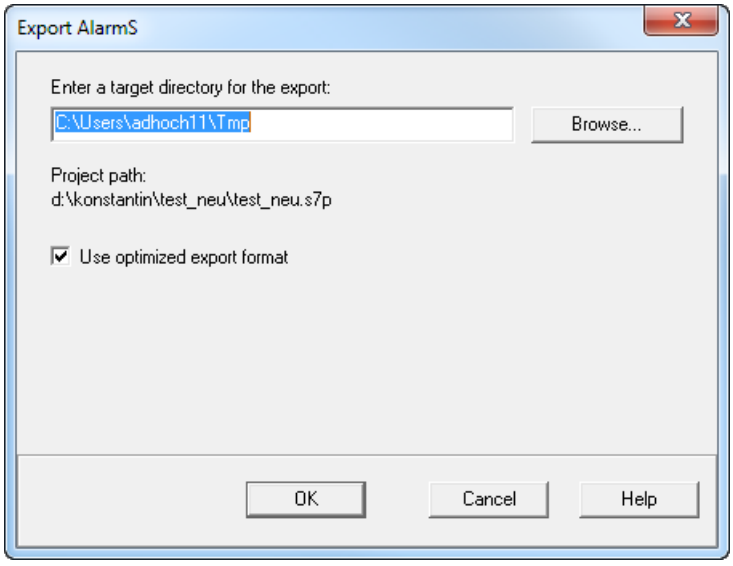


Figure 6-83 SIMOTION SCOUT language export, specification of the target directory

The **Project > Messages > Export AlarmS** menu item exports all user-defined texts in all available languages as XML files. During the upload to the device, only the language preselected in SIMOTION SCOUTis saved.

Every change made in SIMOTION SCOUT requires the texts to be exported and uploaded again.

Note

Special characters in AlarmS messages

Characters that cannot be represented are shown as question marks in messages.

The @ character is a reserved character for SCOUT. The $ character is a reserved character for SIMOTION IT. These characters must not be present in a AlarmS message.

OPC UA

For access to OPC UA, the interface and the TCP/IP port are activated on this page.

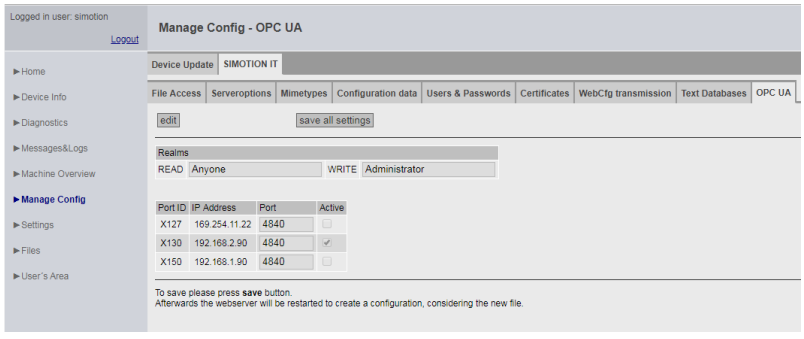


Figure 6-84 Activating OPC UA interfaces

You can find additional information about OPC UA in the **SIMOTION IT OPC UA** Manual.

Settings

This page enables you to change various settings.

Settings for the SIMOTION device can be changed in the **Control Operation state** and **Time Settings** areas.

In the **User Pages** area, you can change how user-defined pages and the **SIMOTION IT** menu editor appear.

⚠ WARNING

Danger to life as a result of incorrect or modified parameterization

As a result of incorrect parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- The Settings page is password-protected. See Login administration (Page 3658)

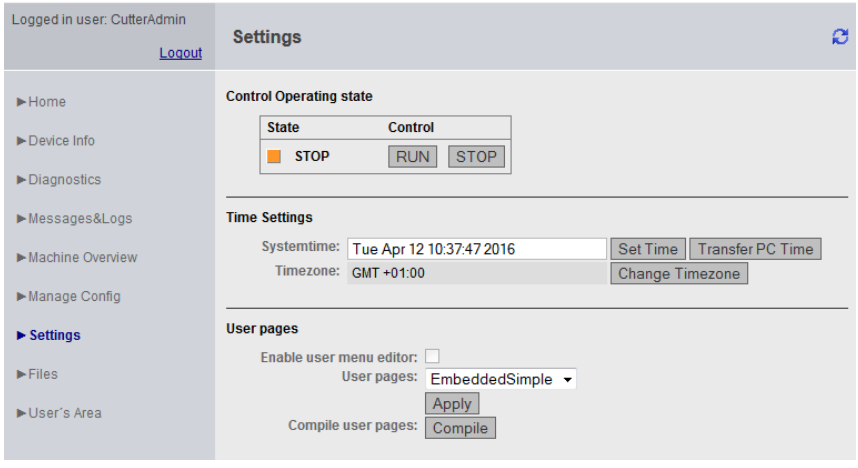


Figure 6-85 Settings

Changing the state of the SIMOTION device

Control Operation state

In the field for the operating state of the SIMOTION device, the request to change the operating state can be triggered by pressing the appropriate **RUN** or **STOP** button.

The switch on the controller has a higher priority than this input, i.e. if this switch is set to STOP , then RUN is not possible.

Note: For the purpose of transferring a project or firmware, the current operating state must be set to STOP.

⚠ DANGER

Danger to life posed by uncontrolled changeover between operating states

Uncontrolled changeover between operating states can cause machines to malfunction, which in turn can lead to injuries or death.

- Include the effects of changeover between operating states in the risk analysis

Time Settings

The system time and the time zone for the SIMOTION device are set in hours, including sign, in the field for the time settings.

Systemtime	Local time-of-day of the SIMOTION device
Timezone	Difference between the Systemtime on site (i.e. local time) and GMT

The **Transfer PC Time** button transfers the current time of the device on which the Web browser runs with SIMOTION IT to the Systemtime input field. The **Set Time** button sets the system time of the controller.

The system time and the time zone are relevant for the OPC XML DA access.

The OPC XML DA client expects all times sent by the SIMOTION device to be in GMT. However, a SIMOTION device is set to local time (GMT + X); therefore, a time zone must be set for the SIMOTION device.

The **Change Timezone** button opens a list of time zones, from which one time zone can be selected.

The time zone can also be set under **Hardware configuration > Object properties of the CPU > "Ethernet Extended" > OPC XML / diagnostic pages** and then applied by running a download. These settings are possible only when **Time Settings** and the <TIMEZONE> in the WebCfg.xml have never been changed.

User Pages

The **Enable user menu editor** checkbox enables you to activate the menu editor link on the user-defined pages. This option will only take effect once **Embedded** has been selected from the **User Pages** drop-down box.

The **User Pages** drop-down box affects how the user-defined pages are displayed. See the SIMOTION IT Programming and Web Services Manual , Section Embedded user-defined pages .

All MWSL pages on the controller can be compiled explicitly with the **Compile** button. This action is required, for example, whenever new MWSL pages are loaded onto the controller by FTP.

See also

<TIMEZONE> (Page 3734)

Files

Files

The subdirectories and files on the memory card of the SIMOTION device can be deleted on the **Files** page.

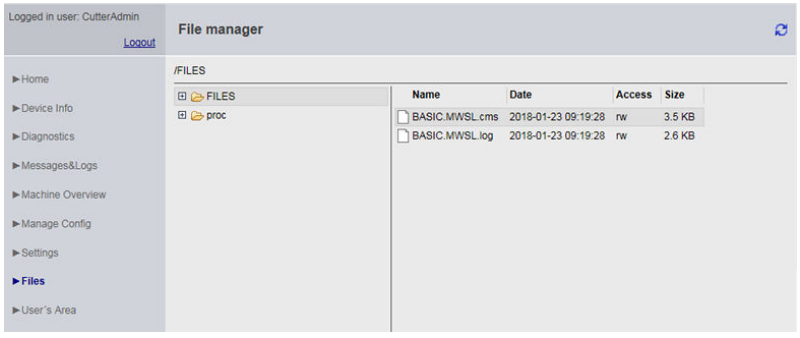


Figure 6-86 Files - User with administrator rights: rw

The File Manager consists of three areas:

- The path line at the top
- The directory tree on the left-hand side
- The directory and file list on the right-hand side

The path line shows which directory content is currently loaded on the right-hand side of the File Manager.

The directory tree shows only directories.

The directory and file list on the right-hand side is shown as a table. The width of the columns in this table is automatically adapted to the content. The unit of the file size in the **Size** column is shown as Bytes, KB, MB or GB depending on the file size.

Access rights

Users with administrator rights have full access to the directories and files of the File Manager.

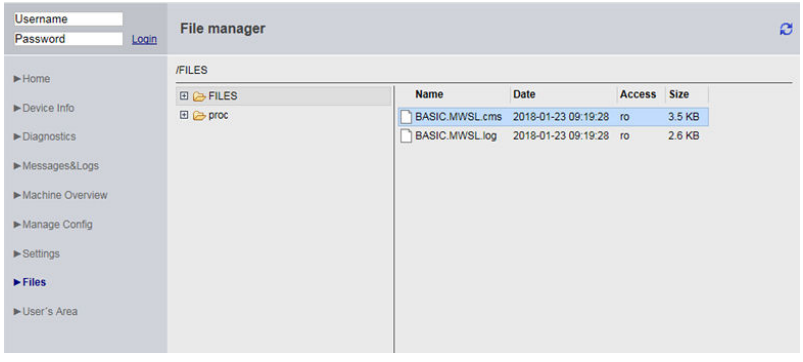


Figure 6-87 Files - User with restricted access rights: ro

The access rights displayed in the File Manager reflect the rights granted to the user. The `ro` access attribute in the above figure shows that only read rights to the files exist.

File Manager functions

The File Manager can be operated with mouse, keyboard and the context menu.

Directories can be created, renamed and deleted. In directories, files can be uploaded, downloaded and deleted.

Note

The following characters are permitted in directory and file names:

- (A-z, -, _)

Context menus

The context menus change depending on whether a directory or a file is selected.

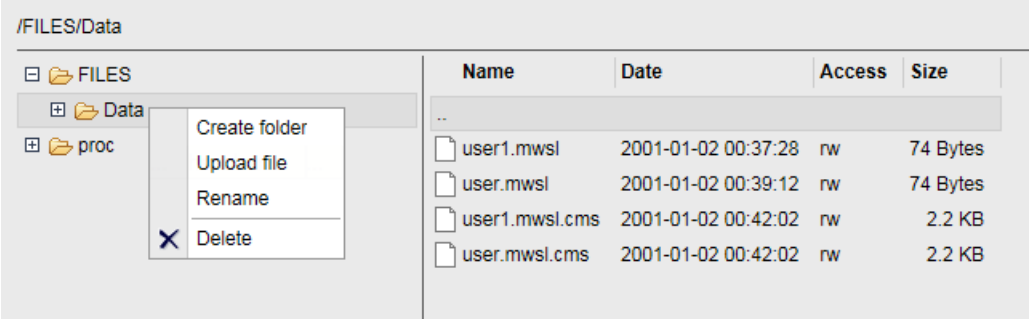


Figure 6-88 Context menu - Directory

When deleting directories, ensure that they do not contain any files.

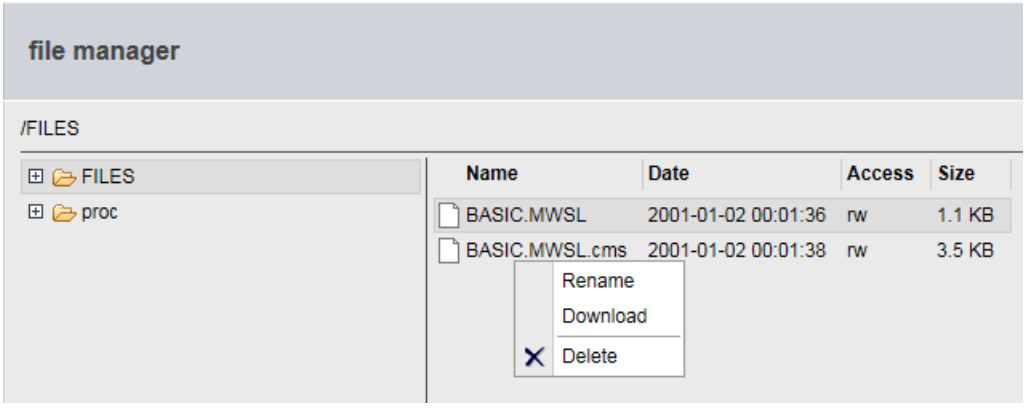


Figure 6-89 Context menu - File

The context menu of a selected file allows this file to be downloaded and deleted. The upload of a file opens the context menu of the directory.

Uploading files to the SIMOTION controller

The **Upload file** button transfers one or more files from the local file system to the SIMOTION controller. The **Download** button of the context menu transfers a file from the controller to the local file system.

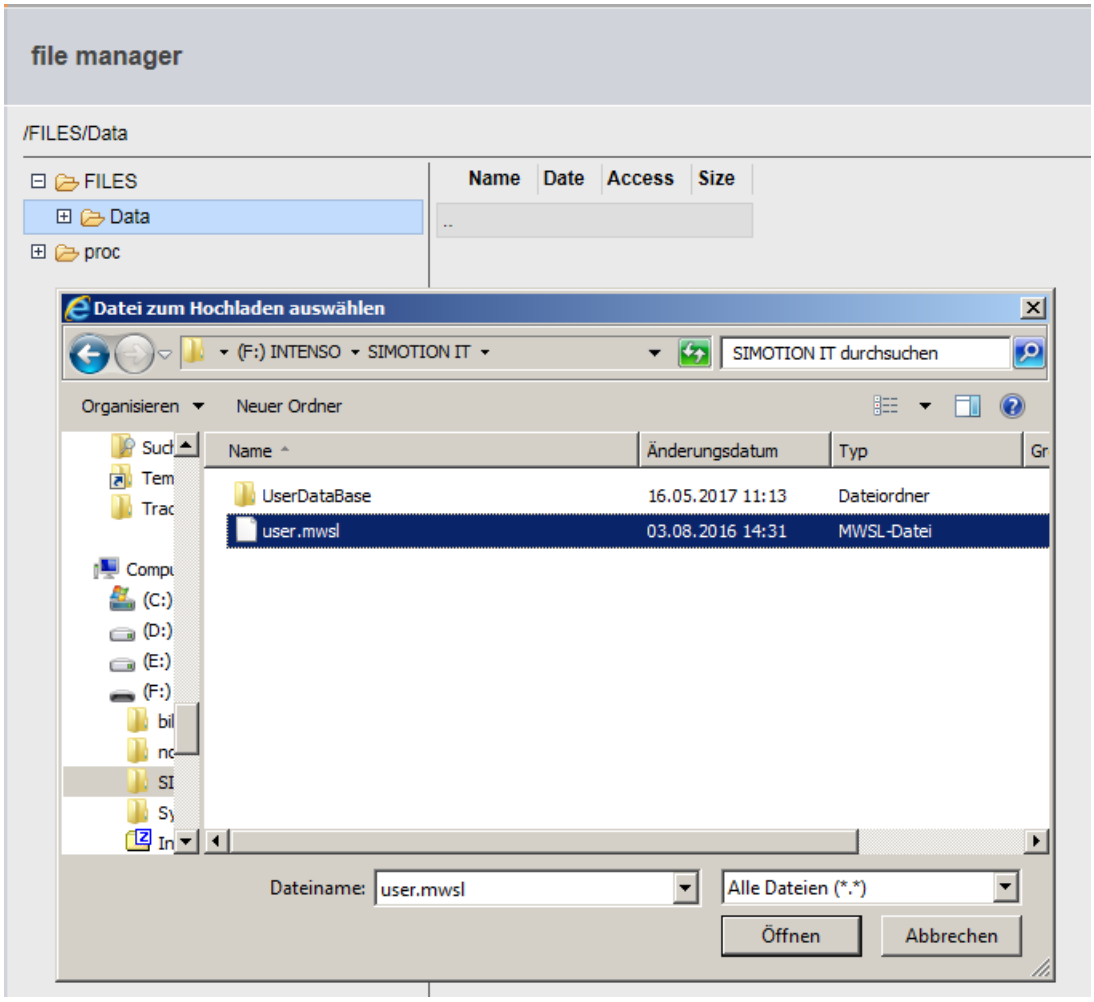


Figure 6-90 Context menu - "Upload file" file dialog

Note

Overwriting existing files

If you upload a file with the same name as one already saved in the SIMOTION controller, the existing file will be overwritten.

Note

Large files

If files that are larger than the remaining space on the memory card are transferred, a different error message will be displayed depending on the browser used.

Browsers do not check prior to transfer whether there will be sufficient memory space on the card for the file. The server cannot compensate for this browser response.

Uploading files with drag-and-drop

If one or more files have been selected, for example in the Explorer, they can be dragged to the desired directory of the File Manager. Because multiple files cannot be transferred concurrently, a queue is created that is processed in the background.

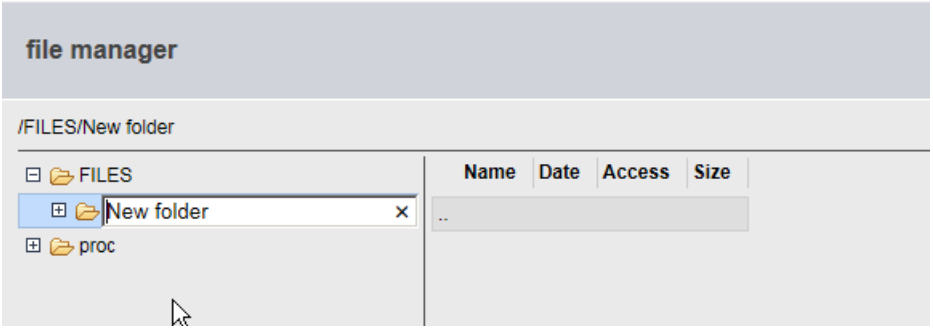


Figure 6-91 Renaming a directory

Appropriate confirmations or messages depending on the action are shown. For example, a confirmation is required when a file or directory is deleted.

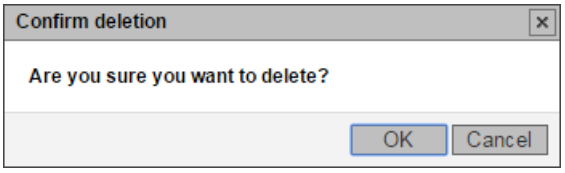


Figure 6-92 Delete file confirmation

File Manager keyboard assignment

Key	Description
Up	Select the previous (above) element.
Down	Select the next (below) element.
Left	Close the directory selected in the directory tree or select a higher-level element in the directory hierarchy.
Right	Open the directory selected in the directory tree or select the first lower-level element, if present.
F2	Activate the editing of a file or directory name.
Del	Delete a file or directory.
Input	Display the directory content in the right-hand area or download a file.
Esc	Cancel the editing of a file or directory name.
Tab	Switch between the directory tree and the list.

Device-specific directory paths

The user-specific directories and files are stored in a separate directory. These directories differ depending on the SIMOTION devices. The information in the table refers to a default installation.

SIMOTION device	Path
C, D	\USER\SIMOTION\HMI\FILES
P350	F:\SIMOTION\USER\CARD\USER\SIMOTION\HMI\FILES
P320	D:\Card\USER\SIMOTION\HMI\FILES
P320-4 E P320-4 S	D:\USER\SIMOTION\HMI\FILES

Note

Available memory space on the card

The memory available on the card is shown on the Diagnostics page in the "Memory Card" line. (Diagnostics (Page 3585)).

Proc

Accessing the device variables using the Proc file system

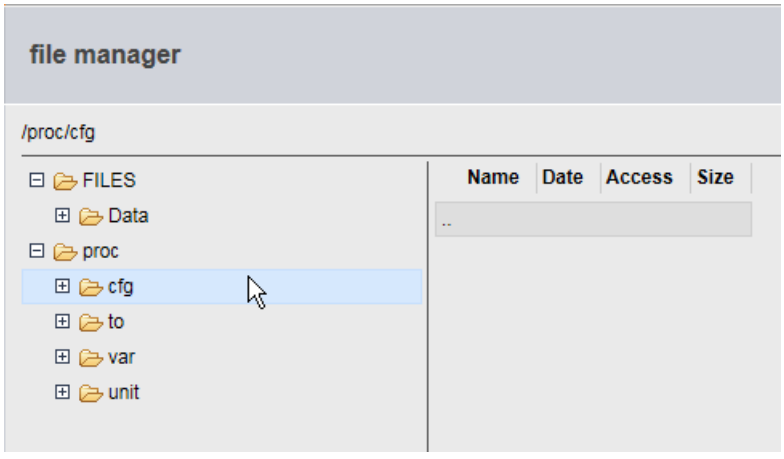


Figure 6-93 Proc file system

The Proc file system shows the device variables as a drive in the browser. This enables device variables to be read out via FTP, for example.

To give an FTP client access to the Proc drive of the controller, reassign the target drive to drive P:

Variables are accessed via a path specification and the addition of the extension "bin" to the name of the variable.

Variables	Path
TO configuration data	/cfg/<toname>/<varname>.bin
TO system variables	/to/<toname>/<varname>.bin
Device system variables	/var/<varname>.bin
Program variables	/unit/<unitname>/<varname>.bin

Arrays are also accessed via a path.

- Variable: unit/UnitName.StructName.StructCompSimple
- Path: /unit/UnitName/StructName/StructCompSimple.bin

Access to arrays and structures

- Variable: unit/UnitName.Array[5].StructName.StructCompSimple
- Path: /unit/UnitName/Array/5/StructName/StructCompSimple.bin

The files in the Proc file system comprise the contents of variables in binary format in the representation (Endianess) of the associated controller.

User's Area

The User's Area displays user-defined pages. The manual *SIMOTION IT Programming and Web Services* describes the creation of user-defined pages.

Figure 6-94 User's Area

6.1.4.4 Simplified standard pages

BASIC pages

Showing SIMOTION IT Diagnostics pages on devices with small displays

As of version 4.1.3, special pages are provided for the optimum display of SIMOTION IT Diagnostics pages on devices such as smartphones.

The following minimum configuration is recommended for the display of the basic SIMOTION IT Diagnostics pages:

- Mobile operating system with installed Web browser that supports the HTML 4 standard
- Minimum screen resolution of 320 x 240 pixels and color display
- Touch screen or stylus-operated device
- JavaScript (ECMA-262) is required if the full scope of functions is required.

You can access these pages via the http://<address>/BASIC/ address or https://<IP address>/BASIC/.

Figure 6-95 Start screen for simplified HTML pages

Device Info

Hardware and firmware information

The following up-to-date hardware and firmware information for the SIMOTION device is displayed on the **Device Info** page:

Manufacturer Name	Siemens AG
Order Number	Article number of the device
Revision Number	Hardware version
Serial Number	Serial number of the SIMOTION device
User Version Firmware	SIMOTION kernel user version
Build Number	Internal version number
Licence Serial Number	License serial number of the device
Operating State	Operating mode of the SIMOTION device (RUN, STOP, STOPU)
Systemtime	Current time-of-day of the SIMOTION device
Additional Hardware	Installed components of the SIMOTION device including: Article number, serial number, revision number, firmware name, user version number, internal version number
Technological Packages	Loaded technology packages including: Package name, user version number, internal version number

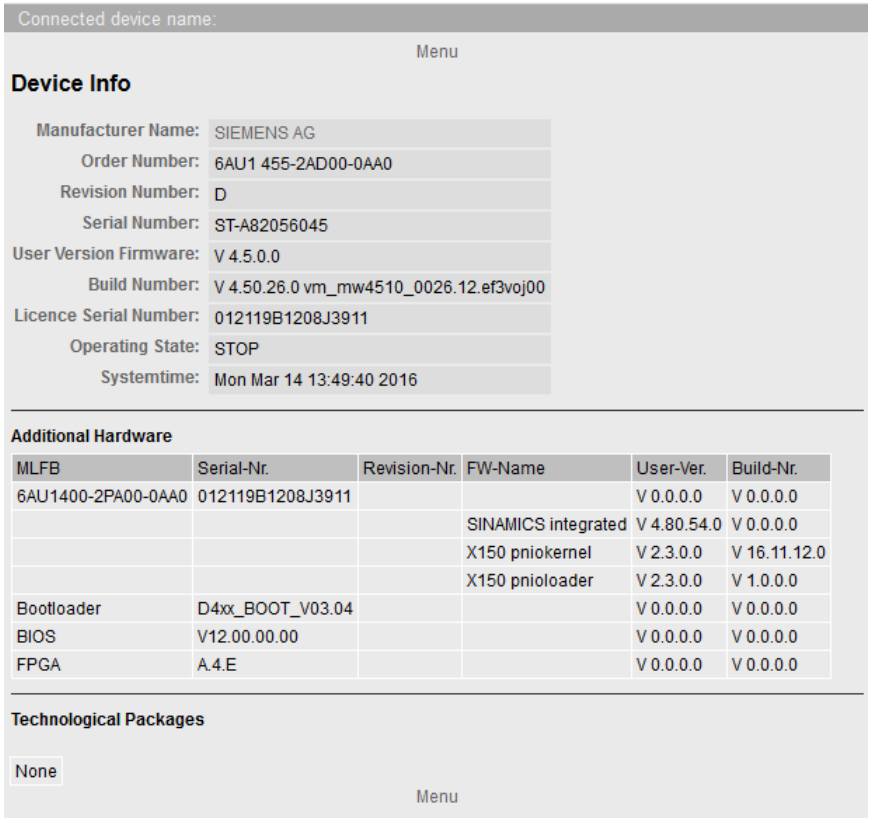


Figure 6-96 Device info on simplified HTML pages

Diagnostics

Overview of the general state of the SIMOTION controller

The **Diagnostics** page displays the following states of the SIMOTION controller:

Systemtime	Current time-of-day of the SIMOTION controller
Timezone	Current difference between the Systemtime and GMT in minutes
CPU Load by cyclic Tasks	Computation time of servo and IPO levels as a percentage of the total computation time
Memory Load	Size and allocation of the memory (RAM), RAM disk, memory card, and non-volatile memory. The memory space details are adapted dynamically to the size (B, kB, MB, etc.).
State	Current operating mode of the SIMOTION controller

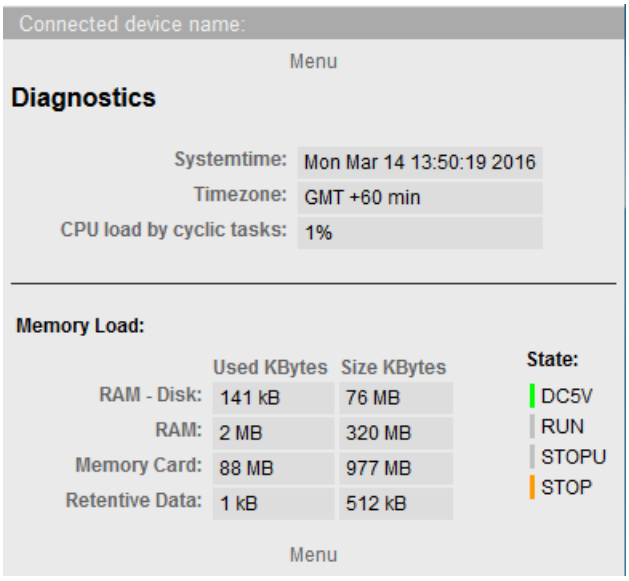


Figure 6-97 Diagnostics shown on simplified HTML pages

See also

Diagnostic files (Page 3606)

Diag buffer

Diag buffer information

The **Diag buffer** page shows the events in the diagnostics buffer.

Time	Time of the event
Date	Date of the event
Event	Displays the event as text. If the DGBUFTXT.EDB language file is missing, the display is in English. English texts are pre-installed on the device.

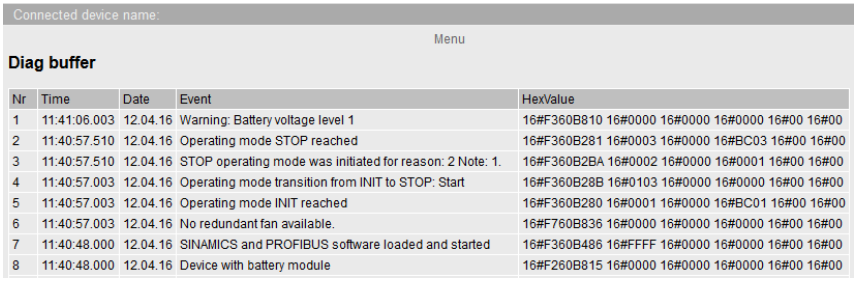


Figure 6-98 Diagnostics buffer shown in simplified format

See also

Diag buffer (Page 3607)

Diag buffer drive

Diag buffer drive information

The **Diag buffer drive** page shows the events in the drive diagnostics buffer for the integrated drives.

Time	Time of the event
Date	Date of the event
Event	Displays the event as text. If the DGEXTTXT.EDB language file is missing, the display is in English. English texts are pre-installed on the device.

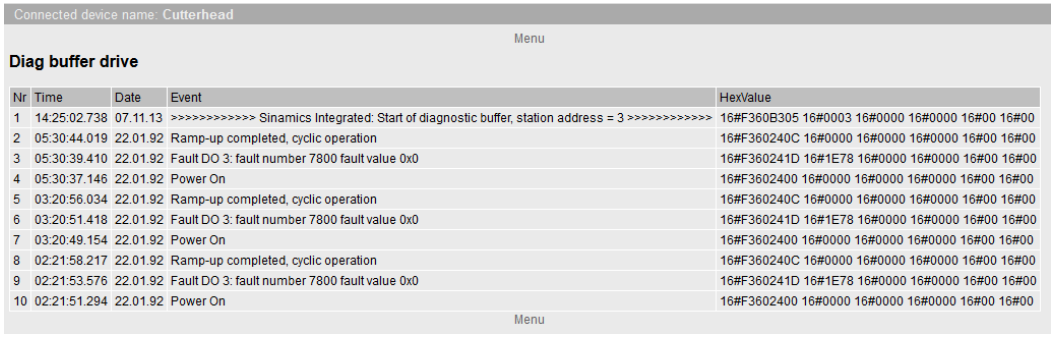


Figure 6-99 Diag buffer drive

Alarms

Information about alarms

Level	Category of the alarm
Time	Time of the alarm
TO	Technology object that triggered the alarm
Nr	Alarm number
Text	Displays the alarm message as text

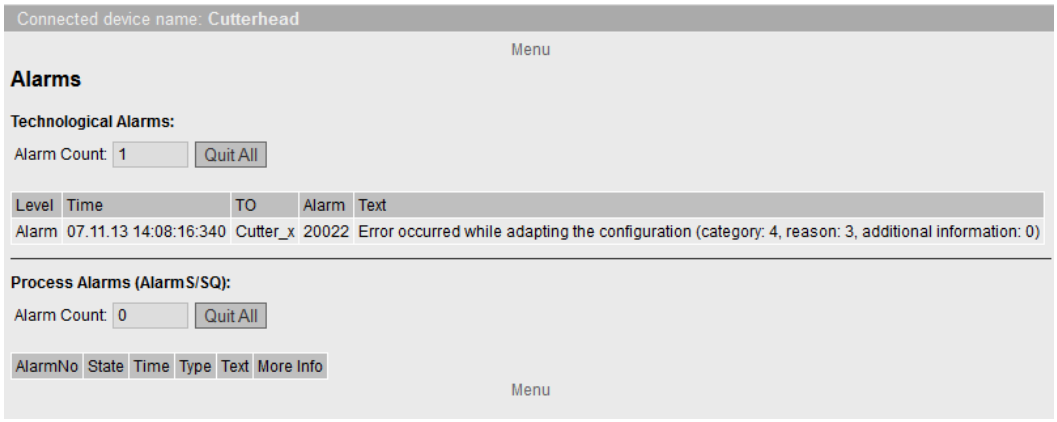


Figure 6-100 Alarms shown in simplified format

IP Config

Data of the SIMOTION controller Ethernet interface

IP Address	Address of the interface
Subnet Mask	Subnet mask of the interface
MAC Address	Subnet mask of the network card
Gateway	Default gateway of the interface
	The corresponding information is always displayed in the first column. It is not necessarily directly related to the IP address of the column and may even have been configured for the other interfaces.
Ethernet-port status:	Overview of the Ethernet ports. The port speed and communication type are output for active ports.

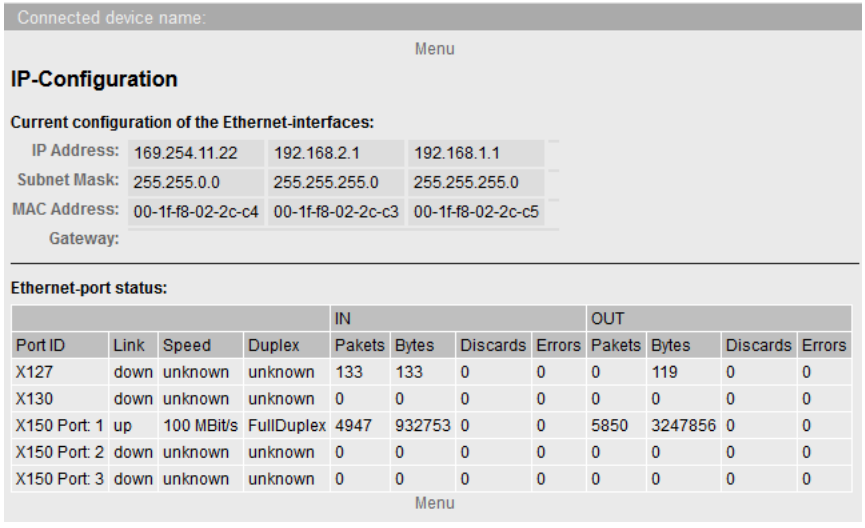


Figure 6-101 IP Config

Port ID	Designation of the Ethernet or PROFINET port as stated on the hardware housing.
Link	Switching property of the port
Speed	Communications speed of the port
Duplex	Communications type of the port
Pakets - IN	Number of packets received at this port.
Bytes - IN	Number of octets received at this port.
Discards - IN	Number of received packets rejected for internal system reasons (e.g. due to system overload).
Errors - IN	Number of received packets not processed by higher protocol layers because of a detected error. For example, transmission/reception faults of the block and collisions.
Pakets - OUT	Number of packets sent at this port.
Bytes - OUT	Number of octets sent at this port.
Discards - OUT	Number of transmission requests for packets that were rejected. Packets that were rejected even though no errors that would have prevented transmission were detected are also counted.
Errors - OUT	Number of packets that were not sent due to an error.

Diagnostic files

Backing up diagnostic pages of the web server

You can use this page to back up general diagnostic data and individual SIMOTION IT Diagnostics HTML pages.

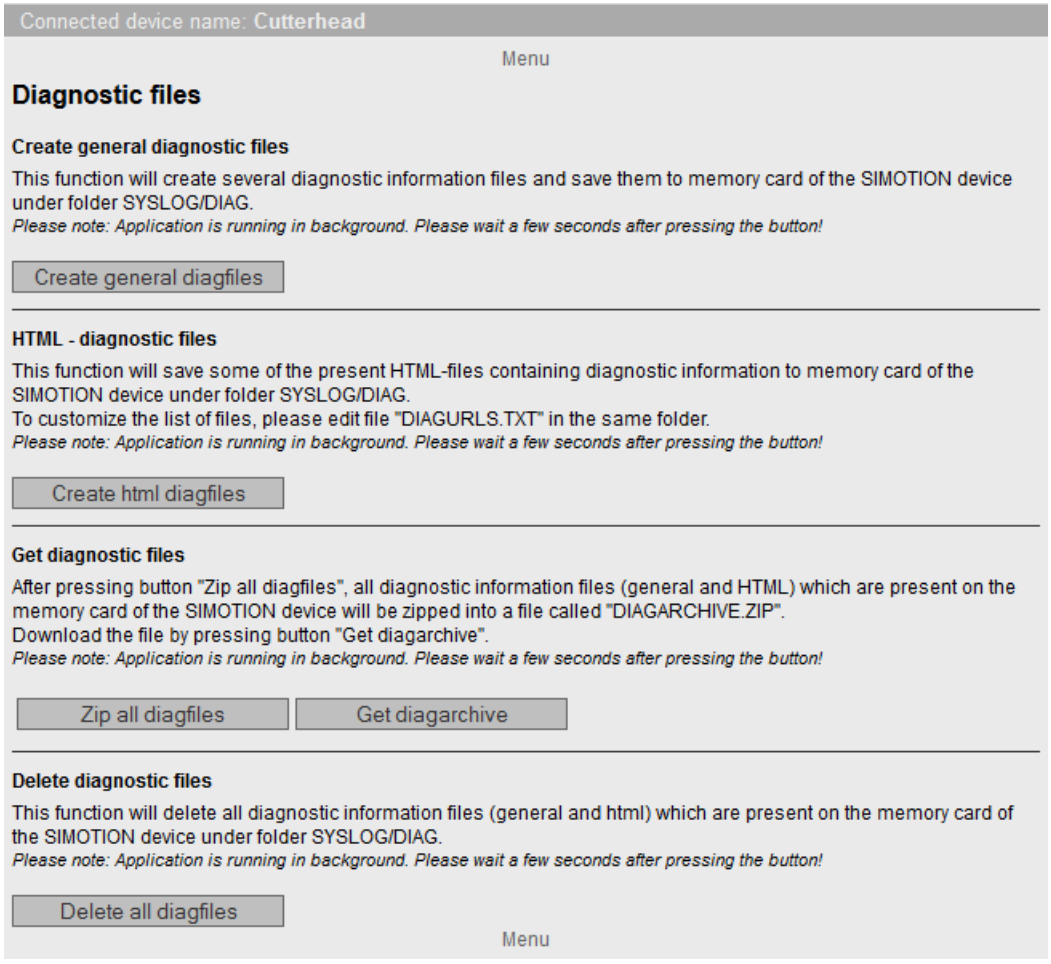


Figure 6-102 Diagnostic files

Watch tables

Watchtables

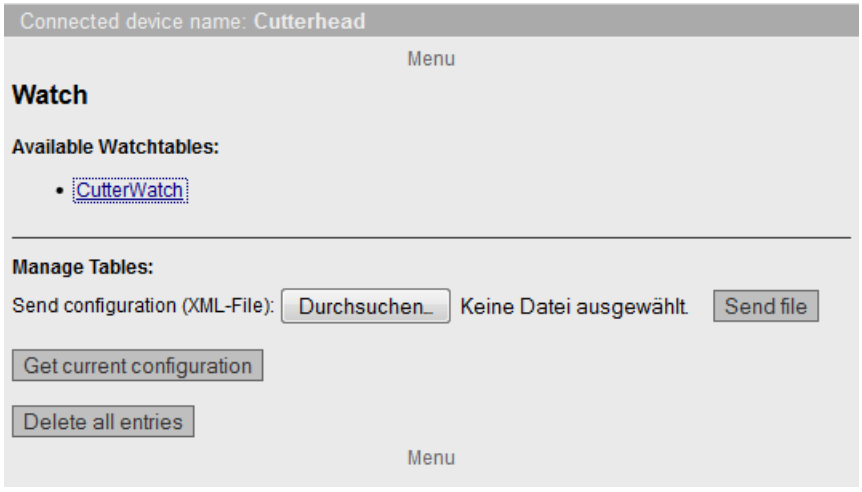


Figure 6-103 Watchtables

This page shows all created watch tables. These watch tables are identical with those on the standard SIMOTION IT Diagnostics page. They can be saved, deleted, and uploaded.

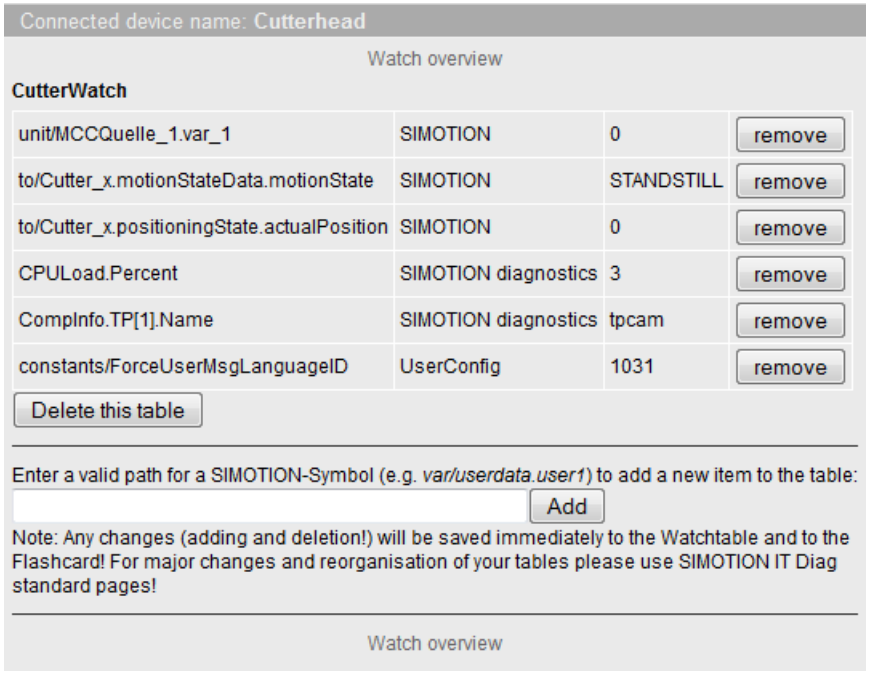


Figure 6-104 Display of a Watchtable

See also

Watch (Page 3589)

User's Area

User’s Area

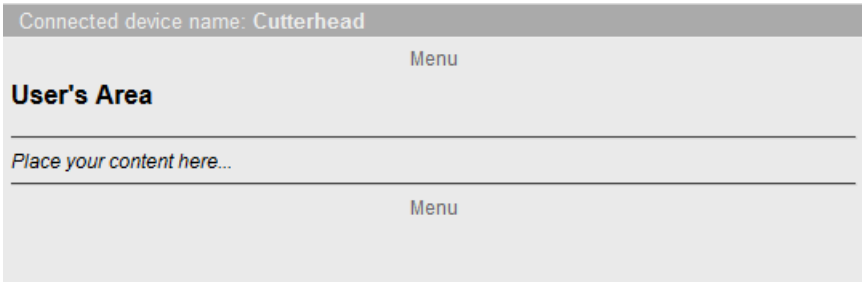


Figure 6-105 User's Area

User-defined pages are displayed in the User's Area .

6.1.4.5 SIMOTION IT configuration

Introduction

The UserDataBase.xml and WebCfg.xml configuration files are used to make user-relevant settings in the Web server.

UserDataBase.xml

File UserDataBase.xml contains user data of the controller. Access to the controller is controlled by the user administration. To back up a device, an administrator must be set up who can set up all other users and groups. See Section Login administration (Page 3658).

WebCfg.xml

All non-user-specific settings are made in the WebCfg.xml file. The file is subdivided into several different sections, e.g. server options and virtual file system.

The WebCfg.xml can be reloaded during runtime that causes a restart of the Web server. The modified settings are available after the restart. The restart of the Web server also causes restart of the OPC UA and OPC XML-DA server. The Java VM is not restarted.

The **Manage Config > SIMOTION IT** standard pages can be used to safely modify entries in WebCfg.xml . See SIMOTION IT File Access (Page 3627)

Structure of the webcfg.xml configuration data

The configuration file is divided into various areas:

- Virtual file system: Representation of the physical file system of the memory card in XML format.
- Server options: Replace the home page of the standard diagnostic pages with your own home page (see the *SIMOTION IT Programming and Web Services* Manual, *User-defined home page* section), port settings.
- Configuration area: Module-specific configuration data
- File types: Specification of the Mime type in the HTTP header.

The WebCfg.xml file can be found either on the SIMOTION controller memory card in the USER \SIMOTION\HMICFG\ directory or on the supplied DVD in the 3_Configuration directory (in the default state).

The individual tags of the WebCfg.xml (Page 3721) are listed with examples in the appendix.

MiniWeb versions and WebCfg.xml

The WebCfg.xml has been changed with SIMOTION Version 4.5. If this file exists in an older format, a migration strategy is deployed and any error messages will be entered in the diagnostic buffer.

- If a WebCfg.xml file in the older format with incompatible content is found, the diagnostic buffer entry has the form:
  "Web server: The WebCfg.xml configuration data exists in an older format and will be updated".
- If problems occur during the conversion that cause cancellation, the diagnostic buffer entry is, for example:
  "Webserver: The WebCfg.xml configuration data text could not be updated! Reason: Insufficient memory space available".
- In earlier versions of the MiniWeb, missing `READ`-, `WRITE`- and `REALM` tags resulted in complete write and read authorization of the file and the directory. In the current MiniWeb version, all rights are taken for missing tags.
- To ensure compatibility during the upgrade from an earlier version and on absence of all tags, `READ="Anyone"` and `WRITE="Anyone"` are added. Consequently, it is possible that the file system is open after a migration. The administrator should restrict access to the file system.
- The migration strategy is deployed only for configuration files before version 4.5.
- If the READ and WRITE tags are missing in WebCfg.xml, no further access is possible as of Version 4.5.

Further information about the differences in the various versions of WebCfg.xml is contained in section Updating the firmware to V4.4 (Page 3624).

See also

<HEADER> (Page 3729)

Authentication and login administration

Log-in administration

Structure of the login administration

SIMOTION IT uses a user database to safeguard access to a device. The UserDataBase.xml file contains this user data.

If the controller is started without a user database, a user database is automatically created when the controller starts up. This user database contains no users and is therefore empty.

If Web pages are called in this condition, the anonymous user `Anonymous` is active. This user has no special access rights.

The Web pages can only be used if the Web server has been activated via SCOUT or HW Config. If the Web server has not been activated, communication with the device is not possible. The services are activated by default when a new device is set up, and they must be explicitly deactivated to prevent access. In TIA Portal, the services are disabled by default.

The user administration is based on the Administrator user group. If no user who belongs to the Administrator user group exists in the UserDataBase.xml, no users can be set up, edited, or deleted via the User's & Passwords Web page.

There are many application cases related to the Web server and UserDataBase.xml that differ in terms of the individual files on the memory card.

Empty memory card, no SCOUT project exists on the memory card and empty UserDataBase.xml

The memory card only contains the firmware and the licenses.

In the as-delivered state, file UserDataBase.xml contains no users and is "empty" as far as the system is concerned.

In this case, the status of the controller is **Security Level Low**. To make it possible to perform commissioning via the Web server, all the Web pages can be used without login in this status. The FTP and Telnet can be accessed with any user name and password.

Users can be set up in the following ways to create a valid user database.

1. Call page **Manage Config > SIMOTION IT > Users & Passwords**. Add a user with Administrator group. As soon as the user is saved, the Web server switches to the **Security Level normal** condition because the user database now contains a valid entry.
2. Create a UserDataBase.xml file with content as described below. Upload via the Web page **Manage Config > SIMOTION IT > Users & Passwords**.
3. Create a **UserDataBase.xml** file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
4. Create a UserDataBase.xml file with content as described below. Use the device update tool and save the UserDataBase.xml file to the USERDATABASE folder in directory IT Config.

SCOUT project exists on the memory card and empty UserDataBase.xml

If a valid project exists on the card, the **Security Level normal** status applies to the Web server, in which the Web pages, FTP and Telnet are protected by a login. However, if the

UserDataBase.xml file is in the as-delivered state, it contains no users. In this case, login will not be possible.
The user database can be edited in any of the following ways:

1. Delete project with **Delete user data on card** in SCOUT. The Web server assumes the **Security Level Low** status. The user database can be edited as described above.
2. Create a UserDataBase.xml file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
3. Create a UserDataBase.xml file with content as described below. Use the device update tool (but not via the Web pages) and store file UserDataBase.xml in a folder USERDATABASE in directory IT Config.
4. Service switch 8, simotion.ini or the PSTATE program can be used to reset to Security Level low and thus change the password.

SCOUT project exists on the memory card and UserDataBase.xml contains valid users

If a valid project exists on the card, the **Security Level normal** status applies to the Web server, in which the Web pages, FTP and Telnet are protected by a login.

The user database can be edited in any of the following ways:

1. Call page **Manage Config > SIMOTION IT > Users & Passwords**. After logging in successfully with administrator rights, new users can be created and existing users edited. This however assumes that at least one user who belongs to the Administrator group has already been set up.
2. Create a UserDataBase.xml file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
3. Create a UserDataBase.xml file with content as described below. Use of the device update tool (but not via the Web pages) and storage of the file UserDataBase.xml in a folder USERDATABASE in directory IT Config.

Authentication

The authentication is constructed as follows:

- There are USERs.
- Every USER has a password that can be entered as plain text before startup. After startup, the password exists as A1 Hash .
- Users belong to groups (GROUP).
- Web pages, directories, and applications are protected by secure realms (REALM) defined for each group.
- Only users that belong to the realm can access the protected page.
- Each realm has a group of users who are authorized for access.
- A user can belong to multiple groups.

If a login attempt fails, this is logged in the system log.

Note

Editing the UserDataBase.xml file

- If the UserDataBase.xml file is not adapted, after the SCOUT project has been downloaded, it will no longer be possible to login to the Web pages or access with FTP and Telnet, because no valid user exists.
- The user database UserDataBase.xml must contain at least one user who is a member of group Administrator. Group Administrator is the REALM that the system expects for accessing protected applications, whose access rights cannot be set via WebCfg.xml.
- The editor used for editing UserDataBase.xml must be set to UTF-8 encoding.
- If file UserDataBase.xml contains illegal characters or the XML syntax contains errors, the file cannot be evaluated by the system. This makes login impossible.
- After startup, all plain text passwords are deleted and only exist in encrypted form. Neither can they be ascertained by the administrator. However, the administrator can assign a new password without knowing the old password.
- Since the password is no longer available in plain text in UserDataBase.xml, the password must be entered again each time a change is made to the groups of an existing user, otherwise the A1-Hash cannot be calculated.
- After loading via FTP, the controller must be restarted to transfer the UserDataBase.xml file. A restart the Web server does not suffice.

Structure of the UserDataBase.xml file

The user data is stored in UserDataBase.xml. UserDataBase.xml is located in directory /USER/SIMOTION/HMICFG/USERDATABASE

Sample configuration

UserDataBase.xml before startup

```
<?xml version="1.0" encoding="UTF-8"?>
<UserDataBase>
  <USER NAME="service"
    PASSWORD="a67_YjH"
    ChangePassword="never"
    DESCRIPTION="Administrator with all rights"
    REAL_NAME="">
      <GROUP NAME="Anyone"/>
      <GROUP NAME="Administrator"/>
  </USER>
  <USER NAME="user1"
    PASSWORD="93!ujEa"
    ChangePassword="allowed"
    DESCRIPTION="Normal user"
    REAL_NAME="">
      <GROUP NAME="Anyone"/>
  </USER>
</UserDataBase>
```

UserDataBase after startup

```
<?xml version="1.0" encoding="UTF-8"?>
<UserDataBase>
  <USER NAME="service"
    ChangePassword="never"
    DESCRIPTION="Administrator with all rights"
    REAL_NAME="">
      <GROUP NAME="Anyone"         A1="0302831a41b222c5f5bfc22e5ff80620"/>
      <GROUP NAME="Administrator"
A1="fa712df9294b40baa1e7504f8dd2b0d5" />
  </USER>
  <USER NAME="user1"
    ChangePassword="allowed"
    DESCRIPTION="Normal user"
    REAL_NAME="">
      <GROUP NAME="Anyone" A1="c5a15667e4d0cadff85d35354ea0fbb6"/>
  </USER>
</UserDataBase>
```

Table 6-6 Attributes of the USER node

Attribute	Permissible values	Description
NAME	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Login name
PASSWORD	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Password in plain text
CHANGEPASSWORD	ALLOWED ⇒ The password can be changed by the user in the Web page (default setting). NEVER ⇒ The password cannot be changed by the user in the Web page.	Behavior when user logs in via Web pages. No effect when opening the file in the file system
DESCRIPTION	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Description of the user
REAL_NAME	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Actual name of the user

Table 6-7 Attributes of the GROUP node

Attribute	Permissible values	Description
NAME	Numerals, letters, special characters but not: : =, " , <,>, %, &, \, `,'	Name of the group.
A1	Valid hash value (numerals, letters)	Hash value that is expressed as a MD5 checksum via USER NAME, USER PASSWORD and GROUP NAME. If none exists, generated after the controller starts up.

NOTICE
Invalid XML file
Using impermissible values may invalidate the XML file so that it cannot be read.

See also

SIMOTION IT Users & Passwords (Page 3630)

Login and WebCfg.xml

Differentiated protection of Web pages, directories, and applications with WebCfg.xml

The realms are assigned for individual Web pages, directories, and applications in configuration file WebCfg.xml. Content requiring protection is labeled REALM Administrator. The users belonging to this group are specified in file UserDataBase.xml.

Besides REALM Administrator used by the system, the user can create and apply his own realms to protect Web pages, etc.

Example

Excerpt UserDataBase.xml:

```
…
<USER NAME="user1"
    PASSWORD=""
    ChangePassword="allowed"
    DESCRIPTION="Service with restricted rights"
    REAL_NAME="John Smith">
<GROUP NAME="Anyone"
        A1="c5a15667e4d0cadff85d35354ea0fbb6"/>
<GROUP NAME="Servicegroup"
        A1="45735fdcee4d0cdfafde825354ea0aa17"/>
</USER>
…
```

Excerpt WebCfg.xml:

```
…
<settings.mwsl.cms ALIAS="html/standard/settings.mwsl.cms"
REALM="Servicegroup" READ="Servicegroup" WRITE="Servicegroup"
MODIFY="Servicegroup"/>
…
```

`user1` has been inserted. This user belongs to the new group `Servicegroup` and has access to the `settings.mwsl` page. However, any user who wants to open the Settings page must belong to the `Servicegroup` group. It is therefore recommended that administrators belong to all the groups that exist in the user database.

Realms for applications

Besides the realms for individual MWSL pages and directories, the REALM of some of the applications of the Web server are also defined in the configuration file WebCfg.xml.

These realms can be adapted if necessary.

⚠ CAUTION

Deleting a REALM

If you delete a REALM, the associated pages can be accessed without a login. So carefully check which pages were protected by the REALM.

- Web service for OPC-XML DA and therefore reading, writing, monitoring of the variables of all providers
  ```
  <WEBSERVICE NAME="OpcXml" URL="/SOAP/OPCXML"
  REALM="Administrator" />
  ```

  Note

  As-delivered state without REALM

  In the as-delivered state, this value has no REALM to ensure downward compatibility reasons! It is therefore recommended to prepare the OPC-XML DA client currently being used for password and user name use and to set the REALM here.

- Application for writing variables in all providers on the HTML diagnostics pages:
  <VarApp REALM="Administrator" />
- Application for updating project and firmware:
  <FWUpdtApp REALM="Administrator" />
- Application for reading and writing the user database UserDataBase.xml
  <UserDataBaseApp REALM="Administrator" />
- Application of Jamaica VM for calling servlets
  <JApp REALM="Administrator" />

In addition, there are system applications that require login of a user who belongs to the Administrator group.

A1 hash

Composition of the A1 hash

The A1 hash is formed by generating an MD5 hash value from a combination of user name, password, and REALM.

MD5 (Message-Digest Algorithm 5) is a cryptographic hash algorithm, which saves a character string requiring protection in the configuration but not as plain text.

Saving the password in plain text would have the disadvantage that a hacker could read it and use it to gain unauthorized access to the system. Instead, the password is saved as what is known as a Hash. The Hash is a fingerprint of the password.

To authenticate a user, the client (in this case the Web browser) sends the password to the server, which then generates the hash and the Hash. This Hash can be compared with the one saved in the configuration and the system can respond accordingly. This procedure is considered one of the most secure of its type. More information is available on the Internet, for example at: http://de.wikipedia.org/wiki/Message-Digest_Algorithm_5

Delete password

Deletion of a password in the user database depends on whether the user has administration rights.

Deleting user passwords

The Administrator can always overwrite user passwords. See SIMOTION IT Users & Passwords (Page 3630).

Deleting Administrator passwords

If the Administrator's password is no longer available, one of the methods described below can be used to modify the user database:

- Deleting UserDataBase.xml from the memory card. An empty UserDataBase.xml is created at startup.
- A password can be entered in plain text in UserDataBase.xml on the memory card. Example: `<USER NAME="CutterAdmin" PASSWORD="New password" ....>` The controller overwrites existing A1-Hashes if a `PASSWORD` attribute is found. A new A1-Hash is formed from the found `PASSWORD`.
- By setting the service selector switch to position "8", it is possible to send a UserDataBase.xml to the controller.

Configuration of the file system

Links to the physical file system (ALIAS)

Access to the physical file system of the memory card via the Web server is limited for security reasons.

To access a file via a URL, this file must be located in the so-called WWWRoot. In addition, the Web server recognizes the memory card area SystemRoot. The SystemRoot cannot be accessed via URLs and is used to store configuration files.

Table 6-8 Paths of the Web server areas

WWWRoot	/USER/SIMOTION/HMI
SystemRoot	/USER/SIMOTION/HMICFG

The URL of a file in the file system is always relative to the WWWRoot.

Example

The file mypage.mwsl is located in directory /USER/SIMOTION/HMI/FILES.

The URL for calling the file is: `http://<IP-Address>/Files/mypage.mwsl`

By making settings in file WebCfg.xml, it is possible to create references to individual files or directories in the physical file system. In addition, by assigning REALMS (Page 3668), the access rights to resources can also be assigned.

For that, the physical file system is mapped on XML data nodes. The node `<BASE>` corresponds to the WWWRoot - /USER/SIMOTION/HMI.

Each child node of <BASE> is a reference to a file or directory. Using these references, a direct call without specifying the entire path is possible. The tag name corresponds to the name of the file.

The optional ALIAS attribute creates a reference to a file that can be located in a different directory. The path specification of the ALIAS attribute is relative to WWWRoot. This allows the file to be accessed by several URLs. Each of these URLs must be saved individually.

Table 6-9 Example URL, physical file system and WebCfg.xml

URL	Target in the physical file system	Entry in WebCfg.xml	Remark
<ip address>/ myfile.mwsl	/USER/SIMOTION/HMI/FILES/ myfile.mwsl.cms	<BASE> <myfile.mwsl.cms ALIAS="/FILES/ myfile.mwsl.cms" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone" /> </BASE>	ALIAS to a file.
<ip address>/mydir	/USER/SIMOTION/HMI/FILES/mydir	<BASE> <mydir ALIAS="/FILES/mydir" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone" /> </BASE>	ALIAS to a directory.

MWSL Files are located in the physical file system in a compiled format with file name extension .cms and must be referenced accordingly.

See also

ALIAS attribute (Page 3723)

Browsing of directories

Note

Changed behavior as of Version 4.4

As of version 4.4, the BROWSEABLE and MODIFY attributes no longer have any effect.

Browsing of directories can be activated or deactivated. This is controlled using the `BROWSEABLE` attribute. If the attribute is `TRUE`, a directory view is permitted.

Setting the value of the BROWSEABLE global tag to true enables the browsing of directories by default.

Table 6-10 Examples of paths

```
/
 /Datei1

 /Directory1/
 /Directory1/Datei2.mwsl
 /Directory1/Datei3.mwsl
 /Directory1/Directory2

 /Datei4
```

The root directory / is the same as the FILES directory.

Table 6-11 WebCfg.xml:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE ALIAS="/">
    <www ALIAS="/" BROWSEABLE="true" .../>
  </BASE>
  [...]
</SERVERPAGES>
```

The client requests the URL http://<IP-Address>/www/Directory1.

In the XML file system, the parser searches for www in the root directory and finds ALIAS="/".

In the physical file system, the parser searches for /Directory1.. The "/" forward slash in this path is retained, because this was specified in the ALIAS="/" tag. Directory1 refers to the path.

The Directory1 directory exists in the physical file system. Because Browseable = true and no default HTML page has been specified, the browse view of the directory is returned.

See also

<DEFAULTDOCUMENT> (Page 3728)

Security concept of the file system

Permission information in the form of attributes can be stored at each XML node of the XML file system:

- REALM (secure area)
- READ (reading rights)
- WRITE (writing rights)
- MODIFY (modification rights)

REALM may only contain one group name, while READ, WRITE, and MODIFY may contain a list of group names separated by "," characters. No spaces or other Whitespace characters may be used.

A set of user groups is assigned to each user.

If a file is requested by a user, the XML file system is searched through for this file. The XML tree is run through corresponding to the file path. If several XML nodes are run through, the logged-in user must have rights for all of the "touched" nodes.

Example:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE ALIAS="/">
    <FILES ALIAS="FILES/" BROWSEABLE="true" REALM="Anyone"
           READ="Anyone" WRITE="Anyone" MODIFY="Anyone">
      <www ALIAS="/WebPages/"
           BROWSEABLE="true"
           READ="Administrator"
           WRITE="FileAdministrator" />
    </FILES>
    <Test.mwsl.cms ALIAS="/Tests/Test.mwsl.cms/"/>
    <XMLDir>
    </XMLDir>
  </BASE>
[...]
</SERVERPAGES>
```

Table 6-12 Types of file permissions

URL	Access	Groups	Comment
/<File>.mwsl	Read	None	
/<File>.mwsl	Write	None	Access not permitted
/MainDir/<File>.mwsl	Read	USER	Login mask if USER group is not present

REALM

Setting up a security area

Realm is used to designate a secure area in the WWW environment. If a directory is entered and the user is not a member of the specified Realm (or the user has not yet logged in), a login prompt appears (Authentication required).

If a file protected by REALM is accessed, the client must be authenticated. Web browsers usually display a login screen requiring users to enter their user name and password.

The REALM attribute can be used to enable or force a user login.

Note

Only one REALM can be specified for a directory. In a directory hierarchy, different REALM must be separate, not overlap.

Because the file objects are accessed on a hierarchical basis, different hierarchy levels may well have different security groups. In this case, no user can access the relevant files because it is not possible to change the realm during a request. An access is always connected to a maximum of one realm even if the user is a member of several security groups.

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE>
    <Motion REALM="Operator">
      [...]
    </Motion>
    <Tests REALM="Tester" >
      [...]
    </Tests >
  </BASE>
  [...]
</SERVERPAGES>
```

In this example, a user with the "Operator" and "Tester" security groups has access to Motion and Tests as well their subordinate objects.

NOTICE

ALIAS and XML file system

If you have linked a file or directory with an ALIAS and set the user rights, you must do the same for the XML file system!

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE ALIAS="/">
  <Test.mwsl.cms ALIAS="/Files/Test.mwsl.cms/"
            BROWSEABLE="true"
            READ="Administrator"
            WRITE="Administrator"
            MODIFY="Administrator" />
  [...]
  </BASE>
[...]
</SERVERPAGES>
```

With this configuration, the login window appears when you call

```
http://<IP-Adresse>/Test.mwsl
```

However, access to this page is still possible via:

```
http://<IP-Adresse/Files/Test.mwsl
```

To prevent this, the configuration must be made as follows:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE ALIAS="/">
    <FILES ALIAS="FILES/"
           BROWSEABLE="true"
           READ="Anyone"
           WRITE="Anyone"
           MODIFY="Anyone">

      <Test.mwsl.cms
                BROWSEABLE="true"
                READ="Administrator"
                WRITE="Administrator"
                MODIFY="Administrator" />
    </FILES>

    <Test.mwsl.cms ALIAS="/Files/Test.mwsl/"
              BROWSEABLE="true"
              READ="Administrator"
              WRITE="Administrator"
              MODIFY="Administrator" />

      [...]
  </BASE>
  [...]
</SERVERPAGES>
```

Special features for the administrator

A user with the `Administrator` group/realm can access all pages. The rules of other Realms that have been set up for security reasons do not apply for this user. A directory that has been set to `BROWSEABLE="false"`, is also visible for a user with administrator rights.

Note

Administrator rights prior to V4.4

Prior to V4.4, the behavior for users with administrator rights was different. A user assigned to the `Administrator` user group could only access pages that had been released for this user via the group/Realm. The pages of the CPU were not visible for the user.

See also

REALM attribute (Page 3726)

READ

Creating read authorization with the READ attribute

If the `READ` attribute is specified for a file or directory, the user must be a member of one of the groups specified for the `READ`–attribute. Several groups can be specified for `READ`; they must be separated with commas. Whitespace characters may not be used.

Example

```
<MyDir READ="User,Administrator" />
```

Users that belong to the User or Administrator group (or both) may read the content of the directory.

If users do not have read rights, i.e. they do not belong to any of the groups that are specified with `READ` , a FORBIDDEN message is generated. A login for the client is not initiated.

If no `READ` attribute is present for a directory, read access is always permitted.

See also

READ attribute (Page 3725)

WRITE

Setting write authorizations with the WRITE attribute

If a file or directory has the `WRITE` attribute and the logged-in user is a member of one of the specified groups, this user may create new files only in this directory.

The user may:

- Not create any new directories
- Not overwrite any files
- Not delete any files
- Create new files

Note

To create files, the user also needs READ rights!

See also

WRITE attribute (Page 3727)

Creating directories and files

If directories or files are created, they inherit the authorizations of the directory that contains them.

Rights cannot be changed via the directory browser. Rather, they can only be changed directly by modifying the WebCfg.xml file.

Browsing the file system

The web server allows you to visualize a (physical) directory in the client.

For this purpose, the BROWSEABLE attribute for the ALIAS tag or the global <BROWSEABLE>-tag must be set to true.

If a client accesses this link, a directory view of the directory is created. Navigation from this directory to subdirectories is also possible (also to higher-level directories if browsing is allowed for them).

Provided you have sufficient permissions, you can send, receive and delete files as well as create and delete directories. The appearance of the directory in the client can be freely configured.

If there is no authentication mechanism on the web server, write access is generally not permitted (see Security concept).

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE>
      <www ALIAS="/UserData" BROWSEABLE="true"
           REALM="GuestUser"/>
      <Test.mwsl.cms LINK="/Tests/Test.mwsl.cms/"/>
   </BASE>
   [...]
</SERVERPAGES>
```

In this example, a directory view of the local directory "/UserData" (relative to WWWRoot!) would be returned to the client if it requests the URL /www and has been authenticated as a user of the REALM "GuestUser."

Write access to the directory is not possible because a WRITE or MODIFY attribute has not been specified for the directory entry.

File access via FTP/Telnet

Access via FTP/Telnet

The configuration of the interfaces can be found in Section Configuring the SIMOTION device interface (Page 3568).

Note

As of V4.1.2, HTTP/S, FTP and Telnet are activated in the delivery state. In TIA Portal, these services are disabled by default.

Security concept of HTTP/S, FTP, and Telnet access on the Web server

As of version V4.4, access to the SIMOTION IT Web server is protected by a multi-level security concept.

The security state of the Web server is indicated by the security level on the website. This security level can have three different levels: Low, normal, high.

Further information on the safety concept of the connection via FTP/Telnet can be found Section Security concept (Page 3569).

Access within the user program

Access within the user program can influence the FTP transfer. For example, write access is only possible when the user program has closed the relevant files.

Currently set state of FTP/Telnet

You can see the currently set state of FTP/Telnet with a browser by clicking the arrow in the "Security Level" field when the Web server is active.

Securing FTP access

In the UserDataBase.xml file, a user must be in the Administrator group in order to log in to the FTP. During the FTP login, the user name and password entered there must be authenticated.

⚠ WARNING
FTP access with security level low If the security level is low, the user name and password will not be checked. Any values can be entered.

CSRF protection

Cross Site Request Forgery (CSFR)

Cross Site Request Forgery is an attack scenario with which the attacker without knowledge of the authorized user reloads or forwards an HTTP request (GET/POST). The HTTP request can initiate damaging actions on the Web server or even the complete SIMOTION controller. To protect the SIMOTION IT Web pages and the Web server, proven techniques have been implemented that prevent unauthorized access to the data.

Example of a CSRF attack

A logged in user is tempted to click a URL that executes an unwanted call in its user context.

```
http://<mydevice>/stop
```

The reason for this behavior is that the HTTP is a state-less protocol. After a successful authentication, the browser stores the session data in a Cookie. For every subsequent request, the session data is also sent, which means the Web server always receives a valid call of a logged in user. Consequently, a manipulated link is not detected.

CSFR protection in the MiniWeb

CSRF protects all applications or Web services for which it is potentially possible to generally impair the status of the Web server and/or of the controller. Examples are the provocation of STOP/RUN transitions, the manipulation of project data or the changing of Web server settings. Furthermore, unauthorized fetching of data is prevented. To achieve this, applications that only display information, but do not manipulate the status of the system, are also included in the CSFR protection.

Note

CSRF tokens and login

After a login, CSRF tokens are available, because they are linked with the login data.

A token generated by MiniWeb is placed in Web pages that issue an endangered HTTP request (GET/POST). This token is sent as additional request parameter to the target application, and validated there. If the token is valid, the application is performed.

Where necessary, the standard pages have been extended with the token technology. No further processing of these pages is required.

CSRF protection settings

To activate and deactivate the CSRF protection, the tag `<CSRFProtection disabled="false|true" />` is added to the WebCfg.xml for the device ramp-up. The setting is displayed on the **Manage Config SIMOTION IT Serveroptions** Web page, and can be changed.

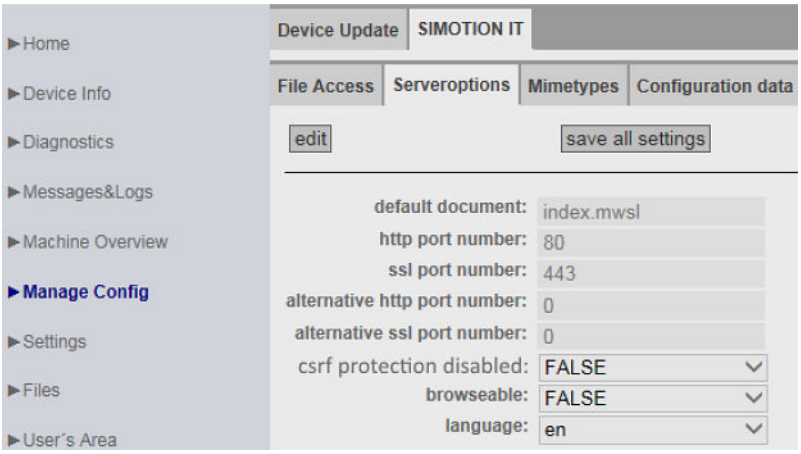


Figure 6-106 CSRF Serveroptions

CSRF protection is activated in the delivered state. The CSRF protection is deactivated for ramp-up with Security Level Low.

Activation of the CSRF protection in the WebCfg.xml: `<CSRFProtection disabled="false" />`

If CSRF protection is activated, all applications and services of the MiniWeb that receive GET/POST requests, and are considered to be protection-worthy with regard to a CSRF attack, respond correctly only when the appropriate tokens were also sent.

Note

CSRF protection and individually-created pages

Users who have created and programmed their own pages must customize these pages or deactivate the CSRF protection.

The following table provides an overview of which applications relevant for the user must be customized for the CSRF protection.

Table 6-13 **List CSRF protection in MiniWeb applications**

Application	URL	Further information	Request type	Token type
JavaApp	`http://<host>/JApp/jvmRestart`	Manual SIMOTION IT Virtual Machine and Servlets	GET	SUT
			POST	-
ServletApp	`http://<host>/servlet/myServlet`	Manual SIMOTION IT Virtual Machine and Servlets	GET	SUT
			POST	MUT
VarApp	`/VarApp`	Manual SIMOTION IT Programming and Web Services	GET	-
			POST	MUT

MUT = Multi Use Token
SUT = Single Use Token

Users must change any user-implemented Web pages in which the above mentioned URLs are called. The affected Web pages must be extended by passing the appropriate token.

Multi Use Tokens can be used more than once. Single Use Tokens are valid just once.

The aspect that an attacker "eavesdrops" the connection, and so also determines the multiple-valid tokens, is not considered. Because a Man-In-The-Middle attack is not a protection goal of the CSRF protection. This must be guaranteed, for example, by using HTTPS.

Single Use Tokens are recommended for deployment with all GET requests to be protected, because the token as parameter, is part of the URL. Because popular Web servers or appropriate company-internal security guidelines store this URL in log files, the use of tokens that lose validity immediately after deployment is recommended. Although the MiniWeb server does not maintain such logs, this criterion is adopted for the selection of the tokens.

6.1.4.6 Variable providers

Overview

Variable providers

The data of the SIMOTION device can be accessed via the "variable providers". Each provider enables access to certain variables.

At present there are five variable providers; these are described in the section below.

- SIMOTION
- SIMOTION diagnostics
- UserConfig
- MiniWeb
- ITDiag

You can access the data supplied by the variable providers from SIMOTION IT OPC XML-DA, OPC UA, SIMOTION IT Diagnostics standard pages, or, if necessary, from user-defined HTML pages.

SIMOTION

You can access SIMOTION process variables via the "SIMOTION" provider. As of V4.1, you can also change the operating status, initiate backups with RamToRom and ActiveToRam, and access drive parameters and technological alarms.

Note

You will find a description of the storage concept in the online help of SIMOTION SCOUT in section "SIMOTION storage concept (in the target device)".

Variables syntax of the "SIMOTION" provider

With OPC XML DA V1.0, access to the variables of the SIMOTION device is via the terms "ItemPath" and "ItemName". In MWSL functions, they are accessed via the "ItemName".

ItemPath

The name for "ItemPath" is always "SIMOTION" for SIMOTION process variables for use in the MWSL and SSI. It is not necessary to specify the ItemPath with MWSL and SSI.

```
ItemPath="SIMOTION"
```

Note

The "ItemPath" is only required for accessing via OPC XML-DA. None of the other SIMOTION IT accesses to the variable provider "SIMOTION" use "ItemPath".

Overview of variable access

Table 6-14 OPC XML-DA variable access

Variables	Variable declaration	Availability	Access syntax	Requirements for access
Global device variables (Page 3682)	*Variable type*			The appropriate checkmark in the properties dialog of the CPU must be set (CPU > Properties > Settings)
	retain	x	glob/<var name>	
	not retain	x	glob/<var name>	
I/O variables (Page 3684)	*Access modes*			
Addresses 0..63	"PI../PQ.. (without assignment to a process image)"		io/_direct.<var name> io/_image.<var name> io/_quality.<var name>	
	"PI../PQ.. (with assignment to a process image)"	x	io/_direct.<var name> io/_image.<var name> io/_quality.<var name>	
	%I../%Q..	-	-	
Addresses >63	"PI../PQ.. (without assignment to a process image)"	x	io/_direct.<var name> io/_quality.<var name>	
	"PI../PQ.. (with assignment to a process image)"	x	io/_direct.<var name> io/_image.<var name> io/_quality.<var name>	
Unit (MCC/ST/LAD-FBD) (Page 3678)	*Variable type*			The compiler option "Permit OPC-UA/-XML (load symbols to RT)" must be set at the source
Interface	(VAR_GLOBAL)	x	unit/<unit name>.<var name>	
	(VAR_GLOBAL RETAIN)	x	unit/<unit name>.<var name>	
Implementation	(VAR_GLOBAL)	-	-	
	(VAR_GLOBAL RETAIN)	-	-	
	(VAR)	-	-	
Unit DCC		x	unit/<unit name>.<var name>	

Internal functionality

The SIMOTION variables provider contains variables that can be used to implement internal functionality.

- diag/ (komplett)
- dev/Device

- dev/DiagIfc
- dev/Licence
- dev/PNDiag
- dev/Service/HTTPDiag
- dev/Service/ManageConfig
- dev/Service/SecLev
- dev/Service/Tasktrace
- dev/Special
- dev/Trace
- dev/dTrace

Accessing system variables/technology object system variables

For **system variables**, the **ItemName** syntax is:

ItemName="var/name"
Example: ItemName="var/userData.user3"

For **TO system variables**, the **ItemName** syntax is:

ItemName="to/name.variable"
Example: ItemName="to/Achse_1.positioningState.actualPosition"

Note

The names of the system variables and technology object system variables to be used can be found in the online help for SIMOTION SCOUT in "System Functions, System Variables and Configuration Data".

Access to unit variables (as of V4.1)

For **unit variables** in the interface, the **ItemName** syntax is:

ItemName="unit/name.variable"
Example: ItemName="unit/prog_1.var_1"

Note

The names to be used for the unit variables in the interface correspond to the program and variable names **in lower case characters**.

Accessing technology object configuration data (V4.1 and higher)

For **technology object configuration data**, the **ItemName** syntax is:

ItemName="cfg/TOName.activeConfigData|setConfigData.variable"

activeConfigData:	Currently valid configuration files, read-only
setConfigData:	Data set image, write access possible The data can be write-accessed if the "effectiveness" property has the "CHANGEABLE_WITH_RESTART" or "CHANGEABLE_WITHOUT_RESTART" value. In the case of "CHANGEABLE_WITH_RESTART," the change does not take effect until the respective technology object has been restarted.

Example: ItemName="cfg/Axis_0.setConfigData.Restart.restartActivationSetting"

Note

The names of the TO configuration data to be used can be found in the online help for SIMOTION SCOUT in "System Functions, System Variables and Configuration Data".

Accessing drive parameters (V4.1 and higher)

For **drive parameters**, the **ItemName** syntax is:

ItemName="drv/TOName|LogAddr.Params.ParamNo"

TOName:	Specifies the technology object name (possible if an Axis technology object exists for the drive object)
LogAddr:	Specifies the logical drive address
ParamNo:	Parameter number If an attempt is made to write-access a read-only drive variable, the drive issues a feedback message (error code) to this effect.

Example 1: ItemName="drv/Axis_0.Params.105"

Example 2: ItemName="drv/256.Params.5"

Accessing technological alarms (V4.1 and higher)

For **technological alarms**, the **ItemName** syntax is:

ItemName="dev/Alarm.Variable|Values-Array

Variable:
- State
  Status of query:
  READY
  BUSY
  ERROR
- Version
  Incremented each time the alarm buffer is modified. By entering this variable in a subscription, you can be notified each time a change is made to the alarm buffer.
- EventCount
  Number of currently pending alarms
- QuitAll
  Acknowledges all pending alarms

Values array: Array with the currently pending alarms

This array contains as many elements as are entered in EventCount.

Example: ItemName="dev/Alarm.Version"

For a currently pending alarm, the **ItemName** syntax is:

ItemName="dev/Alarm.Values[ValueNumber].ArrayElement"

ValueNumber: Index of an alarm in the list of currently pending technological alarms

ArrayElement:
- AlarmNo
  Alarm number
- To
  Name the technology object that generated the alarm
- Time
  Time of the alarm entry
- Text
  Alarm text
- Quit
  Acknowledges the alarm
- Type
  Classification of the technological alarm:
  ALARM
  WARNING
  INFORMATION

Example: ItemName="dev/Alarm.Values[0].AlarmNo"

Changing the operating mode (V4.1 and higher)

For setting the operating mode, the **ItemName** syntax is:

ItemName="dev/Service.BZU.Variable"

Variable:
- Value
  Writing one of the following values changes the operating mode accordingly:
  - STOP
  - STOPU
  - RUN
- State
  Displays the execution states during an operating mode change
  The states change from IDLE to ACTIVE to READY.
- Result
  Shows the result of the operating mode change (when State = READY)
  Result = OK if the operating mode has been changed successfully. Otherwise, Result = Error ID

Example: ItemName="dev/Service.BZU.Value"

RamToRom (V4.1 and higher)

For execution of **RamToRom**, the **ItemName** syntax is:

ItemName="dev/Service.RamToRom.Variable"

Variable:
- Value
  Save operation starts with Value = 0
- State
  Displays the status of the save operation
  The display starts with 0% and continues to 100%.
- Result
  Shows the result of the save operation (when State = 100%)
  Result = OK if the save operation has been completed successfully. Otherwise, Result = Error ID

Example: ItemName=" dev/Service.RamToRom.Value"

Note

Ram ToRom only works with the configuration data. System variables have their download value again after a 'Power on/off.'

ActiveToRam (V4.1 and higher)

For execution of **ActiveToRam** (after changing the configuration data), the **ItemName** syntax is:

ItemName="dev/Service.ActToRam.Variable"

Variable:
- Value
  Save operation starts with Value = 0
- State
  Displays the status of the save operation
  The display starts with 0% and continues to 100%.
- Result
  Shows the result of the save operation (when State = 100%)
  Result = OK if the save operation has been completed successfully. Otherwise, Result = Error ID

Example: ItemName=" dev/Service.ActToRam.Value"

Accessing the global variables (V4.2 and higher)

The way to access the control's "global device variables" created by the user in SCOUT is via /glo/.

For the **global device variables**, the **ItemName** syntax is:

ItemName="glob/name"

To make these variables visible, the symbol information must be

downloaded to the control. For this purpose, the relevant checkmark must be made under **Device > Properties > Settings** in SCOUT.

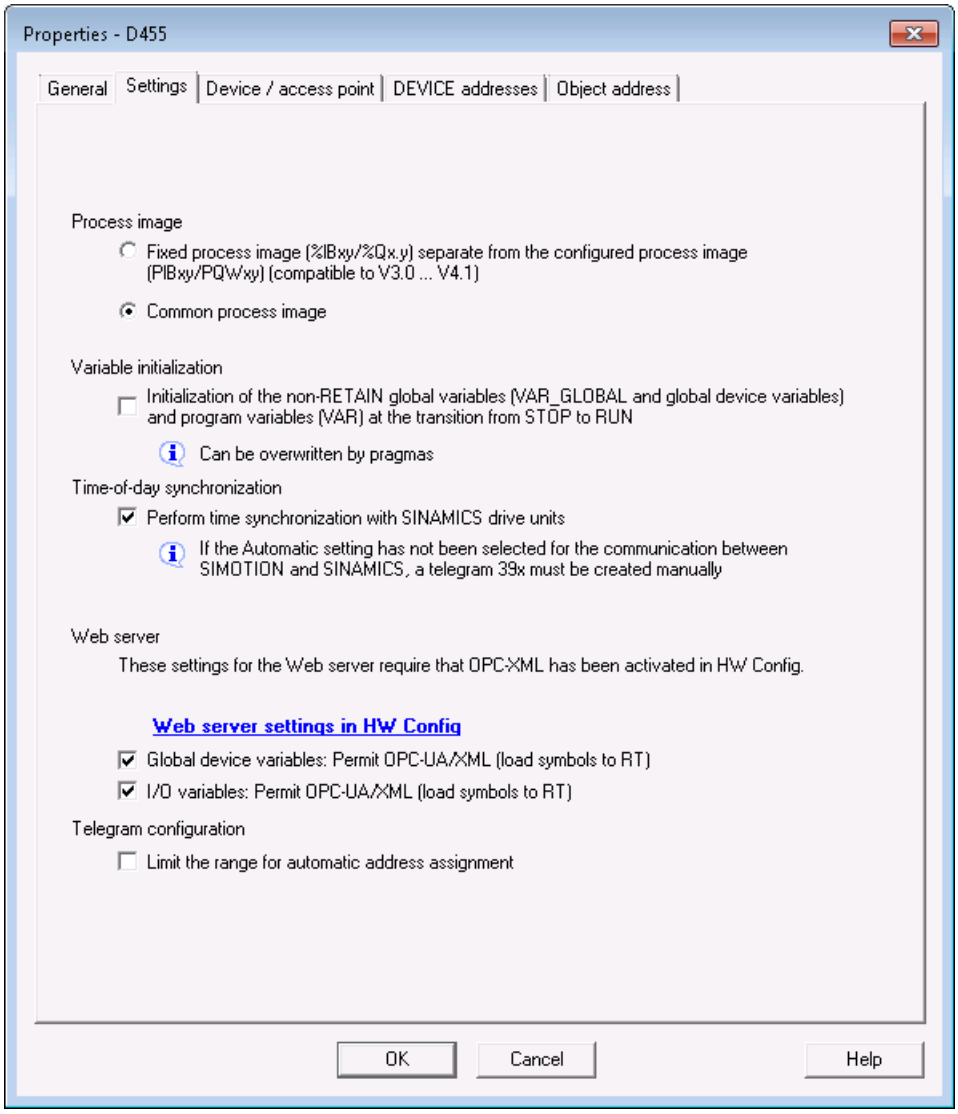


Figure 6-107 SCOUT setting global variables

Accessing the IO variables (V4.2 and higher)

There are three different ways to access the address list of the control's I/O variables that have been created in SCOUT:

- /io/_direct/
  addresses the direct I/O access (current value) of the I/O variables.
  This form of access is offered for all I/O variables.
- /io/_image/
  Addresses the process image of I/O variables.
  Only the I/O variables assigned to a process image are displayed. This applies for I/O variables in the address range 0 to 63 that are accessed via PI... /PQ... I/O variables in this address range that are accessed with %I... /%Q... cannot be displayed via /io/_image.
  All I/O variables outside the address range of 0-63 that are explicitly assigned to a process image in the address list are also displayed.
- /io/_quality/
  addresses the quality of I/O variables, i.e. the I/O status of the subslot (from HW Config) which contains this I/O variable.
  This is a 32-bit pattern. An overview of the possible bit pattern values can be found in the *SIMOTION ST Structured Text* manual, in the section entitled 'Access to I/O variables (as of V4.2)'.
  The quality is the same for all I/O variables in a subslot. The quality is given as an integer for the individual I/O variables of the basic data types (BIT, BYTE, WORD, DWORD) and for arrays. It is not given for array elements (i.e. arrays cannot be expanded).

For the IO **variables**, the **ItemName** syntax is:

ItemName="io/_direct|_image|_quality/name"

The symbol information must be loaded to the control in order to make these variables visible. For this purpose, the relevant checkmark must be made under **Device > Properties > Settings** in SCOUT.

Accessing the AlarmS messages (V4.2 and higher)

Access to the AlarmS messages created by the user in SCOUT and triggered by the controller.

For **AlarmS messages**, the **ItemName** syntax is:

ItemName="dev/AlarmS.Values[ValueNumber].ArrayElement"

ValueNumber: Index of an AlarmS in the list of currently pending technological alarms

ArrayElement:
- AlarmNo
  Alarm number
- AddInfo
  Additional information
- EventCount
- Time
  Time of the AlarmS entry
- Text
  AlarmS text
- Quit
  Acknowledge the AlarmS
- QuitAll
  Acknowledge all AlarmS messages
- Type
  S / SQ
- State
  Status of the AlarmS
- Version

Example: ItemName="dev/AlarmS.Values[0].AlarmNo"

SIMOTION diagnostics

Introduction

Access to diagnostics variables

The diagnostics variables of a SIMOTION control can be accessed via the "SIMOTION diagnostics" provider.

Most of the variables have read-only access and a few (e.g. operating mode) also have write access. All variables are of the string type. Therefore, numerical values are converted into strings by the provider.

The variable management area is dynamic and depends on the current configuration of the SIMOTION control. The provider supports browsing via OPC XML DA V1.0, meaning that the current variable management area can be viewed.

Variables groups of the "SIMOTION diagnostics" provider

The diagnostics variables of the "SIMOTION diagnostics" provider are combined into groups.

A variable name is made up of the group name and variable name:

For example: Group.Variable

DeviceInfo group

General information about the SIMOTION device

The DeviceInfo group contains general information about the SIMOTION device. The 10 variables of this group are always available.

Table 6-15 Variables of the DeviceInfo group

Variable	Description
DeviceInfo.Board	Specifies the system being used, read only
DeviceInfo.Licence-Serial-Nr	License serial number for this device, read-only
DeviceInfo.BZU	Access to the operating state, read and write, valid values for writing: STOP, STOPU, RUN
DeviceInfo.Systemtime	Access to the system time, read and write, the time must always be specified as in the following example: "Tue Aug 05 17:00:00 2003", any other format is not accepted.
DeviceInfo.Timezone	Time offset in minutes, read and write, valid values are -720 to +720
DeviceInfo.Active-MAC-0, ...-1, -2, -3	Active MAC address, read-only
DeviceInfo.Remanent-MAC-0, ...-1, -2, -3	Retentive MAC address, read-only
DeviceInfo.IP-Address-0, ...-1, -2, -3	IP configuration data (address, subnet mask and gateway), read-only
DeviceInfo.Subnet-Mask-0, ...-1, -2, -3	
DeviceInfo.Gateway	

CompInfo group

This group supplies information about the components of the device. The number of variables varies in this group depending on the number of technology packages or additional hardware components.

All variables are read-only.

Information about the CPU

The following variables of the CompInfo group supply information about the CPU.

Table 6-16 Variables of the CompInfo group

Variable	Description
CompInfo.Cpu.MLFB	CPU MLFB / article number
CompInfo.Cpu.Serial-Nr	CPU serial number
CompInfo.Cpu.Revision-Nr	Revision number
CompInfo.Cpu.Kernelname	Kernel name
CompInfo.Cpu.Build-Nr	Build number
CompInfo.Cpu.User-Version	User version (firmware)

Information about the technology packages (TPs) and hardware

The number of available TPs or hardware components can be determined with the following variables.

Table 6-17 Variables of the CompInfo group

Variable	Description
CompInfo.TP-Count	Number of available technology packages
CompInfo.HW-Count	Number of components from HW Config without TPs and CPU itself, => quantity of Additional Hardware on DeviceInfo.mwsl

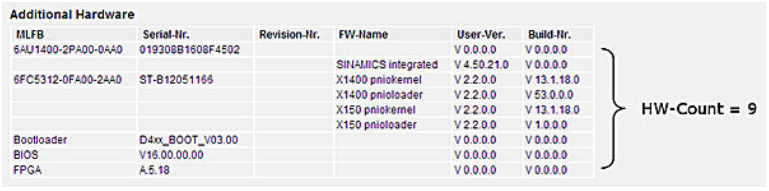


Figure 6-108 Example of CompInfo.HW-Count

When TPs are available, information about the individual TPs can be obtained with CompInfo.TP[x].<variable> (whereby x stands for the TP number).

The first TP is allocated the number 1 (not 0), for example: CompInfo.TP1.Name

The following information is available:

Table 6-18 Variables of the CompInfo group

Variable	Description
CompInfo.TP[x].Name	Name of the TP
CompInfo.TP[x].User-Version	User version of the TP
CompInfo.TP[x].Build-Nr	Build number of the TP

If additional hardware components are available, information about the individual hardware components can be obtained with CompInfo.HW[x].<variable> (whereby x stands for the HW number).

The first hardware component is allocated the number 1 (not 0), for example: CompInfo.HW1.Firmwarename

The following information is available:

Table 6-19 Variables of the CompInfo group

Variable	Description
CompInfo.HW[x].MLFB	MLFB / article number
CompInfo.HW[x].Serial-Nr	Serial number
CompInfo.HW[x].Revision-Nr	Revision number
CompInfo.HW[x].Firmwarename	Firmware name
CompInfo.HW[x].Build-Nr	Build number
CompInfo.HW[x].User-Version	User version

As the information is dynamic and the scope is not known beforehand, the following variables also exist to simplify the display of hardware components and TPs in HTML:

Table 6-20 Variables of the CompInfo group

Variable	Description
CompInfo.TableHead.TP	Supplies the header of an HTML table with all information about the TPs, e. g. "<tr><th>TP name</th><th>User ver.</th> <th>Build no.</th></tr>"
CompInfo.Table.TP	Supplies an HTML table with all the information about all the available TPs
CompInfo.TableHead.HW	Supplies the header of an HTML table with all the information about the hardware components, e. g. " <tr><th>MLFB</th><th>Serial no.</th> <th>Revision no.</th><th>FW name</th> <th>User ver.</th><th>Build no.</th></tr> "
CompInfo.Table.HW	Supplies an HTML table with all the information about all the available hardware components

Note

Separate access to the table and the table header enables separate formatting.

CPULoad group

Information on CPU load

The CPULoad group supplies information on the load of the CPU. Access to all variables is read-only.

Table 6-21 Variables of the CPULoad group

Variable	Description
CPULoad.Percent	CPU load in percent
CPULoad.Mintime	Minimum runtime of the BackgroundTask (free cycle) in ms with 5 decimal places
CPULoad.Acttime	Actual runtime of the BackgroundTask (free cycle) in ms with 5 decimal places
CPULoad.Maxtime	Maximum runtime of the BackgroundTask (free cycle) in ms with 5 decimal places

MemoryLoad group

Information about memory load

The MemoryLoad group supplies information on the load of the storage media. The unit of the memory details is variable and determined by the _sunit value.

Access to all variables is read-only.

Table 6-22 Variables of the MemoryLoad group

Variable	Description
MemoryLoad.Flash-Size	Size of the Flash memory.
MemoryLoad.Flash-Size_sunit	Unit of memory value.
MemoryLoad.Flash-Used	Currently occupied Flash memory.
MemoryLoad.Flash-Used_sunit	Unit of memory value.
MemoryLoad.RAM-Size	Size of the RAM.
MemoryLoad.RAM-Size_sunit	Unit of memory value.
MemoryLoad.RAM-Used	Currently occupied RAM.
MemoryLoad.RAM-Used	Unit of memory value.
MemoryLoad.RAMDisk-Size	Size of the RAM disk.
MemoryLoad.RAMDisk-Size_sunit	Unit of memory value.
MemoryLoad.RAMDisk-Used	Currently occupied RAM disk memory.
MemoryLoad.RAMDisk-Used_sunit	Unit of memory value.
MemoryLoad.Remanent-Size	Size of the retentive memory.
MemoryLoad.Remanent-Size_sunit	Unit of memory value.
MemoryLoad.Remanent-Used	Currently occupied retentive memory.
MemoryLoad.Remanent-Used_sunit	Unit of memory value.

TaskRT group

Variables of the TaskRT group

The TaskRT group supplies information about the task runtimes and the task states of the SIMOTION device. The same values are supplied as in the SIMOTION SCOUT under device diagnostics, task runtimes. Access to all values is read-only. The number of variables varies and depends on the current configuration of the execution system in SIMOTION SCOUT.

Table 6-23 Variables of the TaskRT group

Variable	Description
TaskRT.TaskCnt	Supplies the number of currently available tasks

Task names

The following information can be obtained for the individual tasks via TaskRT.Task-name.Variable-Name. The tasks have the same name in SIMOTION IT and SCOUT .

The same information can be obtained for every task; here is an example of the first MotionTask.

Example:

TaskRT.MotionTask_1.Status

Current task status, can be an appropriate combination of the following values:
STOP_PENDING, STOPPED, RUNNING, STOP_UNCOND, WAITING, SUSPENDED,

WAITING_FOR_NEXT_CYCLE, WAITING_FOR_NEXT_INTERRUPT, LOCKED, SUSPENDED_BY_DEBUG_MODE

Additional variables of the TaskRT group

Table 6-24 Variables of the TaskRT group

Variable	Description
TaskRT.MotionTask_1.Actual	Current runtime of the task in ms, with 5 decimal places
TaskRT.MotionTask_1.Min	Minimum runtime of the task in ms, with 5 decimal places
TaskRT.MotionTask_1.Max	Maximum runtime of the task in ms, with 5 decimal places
TaskRT.MotionTask_1.Average	Average runtime of the task in ms, with 5 decimal places

As the information is dynamic and the scope is not known beforehand, the following variables also exist to simplify the display of task information in HTML:

Table 6-25 Variables of the TaskRT group

Variable	Description
TaskRT.TableHead	Supplies the header of an HTML table with all the information about the tasks, e.g. " <tr><th>Taskname</th><th>Status</th> <th>Actual</th><th>Min</th><th>Max</th> <th>Average</th></tr> "
TaskRT.Table	Supplies an HTML table with all the information about the available tasks; all runtime values are entered with the unit as, unlike the individual value query, they can vary between s and ms. Three decimal places are displayed.

DiagBuffer group

The DiagBuffer group supplies information about the events in the DiagBuffer . Access to all variables is read-only.

Events can be output in English, French, German, Italian, and Spanish text.

Requirement

Text is output in English by default. To display event text in a different language, a file in the relevant language must be downloaded to the SIMOTION controller memory card.

Language	File name
English	DGBUFTXT-EN.EDB
German	DGBUFTXT-DE.EDB
French	DGBUFTXT-FR.EDB
Italian	DGBUFTXT-IT.EDB
Spanish	DGBUFTXT-ES.EDB

Language-specific file names of the DiagBuffer texts

Procedure SIMOTION D,C

1. Open the \3_Diag_Buf_Messages\Diag_Buf_Messages directory on the SIMOTION IT DVD.
2. Insert the SIMOTION controller memory card in a reader/writer.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the \USER\SIMOTION\HMICFG directory. You must create the directory if it does not already exist.
4. Insert the memory card in the SIMOTION device again.

Procedure for SIMOTION P350

1. Shut down the SIMOTION P controller.
2. Open the AddOn\4_Accessories\SIMOTION_IT\3_Diag_Buf_Messages \Diag_Buf_Messages directory on the SIMOTION SCOUT Add-Ons DVD.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the F:\SIMOTION\USER\CARD \USER\SIMOTION\HMICFG directory (for the default installation).
4. Start the SIMOTION P controller.

Procedure for SIMOTION P320

1. Shut down the SIMOTION P controller.
2. Open the AddOn\4_Accessories\SIMOTION_IT\3_Diag_Buf_Messages \Diag_Buf_Messages directory on the SIMOTION SCOUT Add-Ons DVD.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the D:\Card\USER\SIMOTION \HMICFG directory (for the default installation).
4. Start the SIMOTION P controller.

Procedure for SIMOTION P320-4 E, P320-4 S

1. Shut down the SIMOTION P controller.
2. Open the AddOn\4_Accessories\SIMOTION_IT\3_Diag_Buf_Messages \Diag_Buf_Messages directory on the SIMOTION SCOUT Add-Ons DVD.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the D:\USER\SIMOTION\HMICFG directory (for the default installation).
4. Start the SIMOTION P controller.

Note

On delivery and following a firmware update, the English version will be present on the device in all cases.

If the English language version is deleted and a different language version stored on the memory card, the English language version will be recreated at the next startup. The stored (non-English) language version is activated.

For reasons of compatibility, a DGBUFTXT.EDB file is recognized, even if no DGBUFTXT-XX.EDB file is found. If both files are present, priority is given to DGBUFTXT-XX.EDB.

Variables of the DiagBuffer group

The following variables are available for enhancing the display:

Table 6-26 Variables of the DiagBuffer group

Variable	Description
DiagBuffer.TableHead	Supplies the header of an HTML table with all events. The contents are: <tr><th>Nr</th><th>Time</th><th>Date</th><th>Event</th></tr>
DiagBuffer.Table	Supplies the contents of an HTML table with all events. The structure of each row is as follows: <tr><td>NUMBER</td><td>TIME</td><td>DATE</td><td>EVENT</td></tr> **Note:** The NUMBER, TIME, DATE , and EVENT texts specified in this format are replaced with the corresponding value of each event.
DiagBuffer.ExtendedTable	Supplies the contents of the HTML table with all events, including the extended entries displayed via the Info button.
DiagBuffer.ExtendedTableStatic	
DiagBuffer.HexValue[]	
DiagBuffer.HexValuesCloak	
DiagBuffer.PlainDataJScript	
DiagBuffer.ExtendedBufferJScript	Supplies the dynamically generated JavaScript fragment required to display the table.
DiagBuffer.LText[]	Supplies an array that enables access to the entire text of the diagnostics buffer entry. The index matches the index of the diagnostics buffer entry. The individual elements of a diagnostics buffer entry (time, date, text, extended entry text) are separated by "/@@/".

The following variables can be used for direct access to the data of certain events in the diagnostics buffer:

Table 6-27 Variables of the DiagBuffer group - direct access

Variable	Description
DiagBuffer.EventCnt	Number of events currently in the diagnostics buffer
DiagBuffer.CplEventCnt	Event counter beyond the circular buffer limit During startup, the buffer is initialized with the current number of diagnostics buffer entries. Each time an entry is made, the value is incremented, even beyond the maximum number of diagnostics buffer entries.
DiagBuffer.Time_1 bis DiagBuffer.Time[]	Array with the times of the associated events

Variable	Description
DiagBuffer.Date_1 bis DiagBuffer.Date[]	Array with the date details of the associated events
DiagBuffer. Text_1 bis DiagBuffer.Text[]	Array with the texts of the associated events **Note:** If the event text number and its parameters cannot be resolved, the number and parameters are output in HEX format. The variable in HEX format is a string of 20 hexadecimal characters (without separators).

Example of an HTML page

```
<html>
 <head>
  <title>SIMOTION <%=DeviceInfo.Board%> - Diagnostics</title>
  <script type="text/javascript">
   <%=DiagBuffer.ExtendedBufferJScript%>
  </script>
 </head>
 <body style="font-family: Arial">
  <h2>Diag Buffer (extended)</h2>
  <table border="2" cellspacing="1" cellpadding="5">
   <font size="4">
    <%=DiagBuffer.TableHead%>
    <%=DiagBuffer.ExtendedTable%>
   </font>
  </table>
 </body>
</html>
```

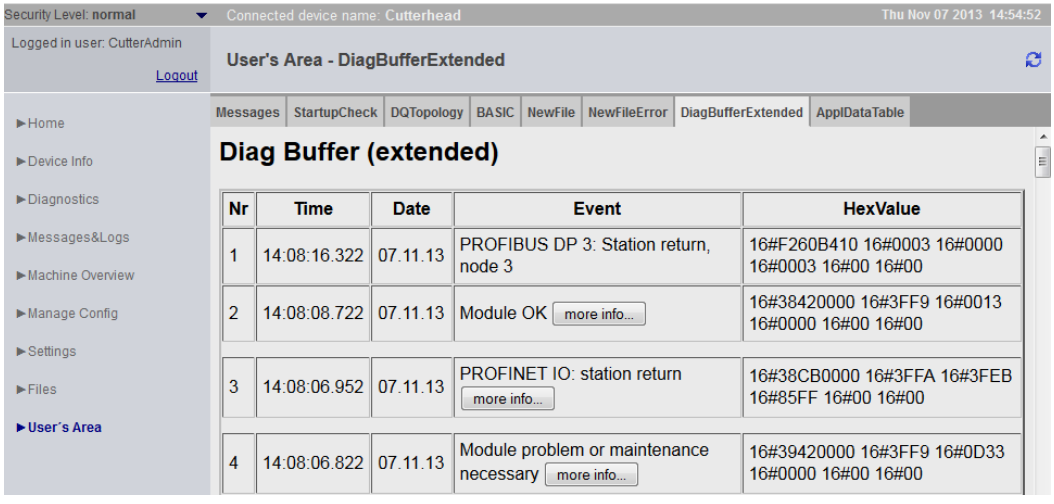


Figure 6-109 Example code result

DiagBufferDrv group

The DiagBufferDrv group provides information about the drive diagnostics buffer. Access to all variables is read-only.

Variables of the DiagBufferDrv group

Variable	Description
DiagBufferDrv.TableHead	Supplies the header of an HTML table with all events. The contents are: <tr><th>Nr</th><th>Time</th><th>Date</th><th>Event</th></tr>
DiagBufferDrv.Table	Supplies the contents of an HTML table with all events. The structure of each row is as follows: <tr><td>NUMBER</td><td>TIME</td><td>DATE</td><td>EVENT</td></tr> **Note:** The NUMBER, TIME, DATE , and EVENT texts specified in this format are replaced with the corresponding value of each event.
DiagBufferDrv.ExtendedTable	Supplies the contents of the HTML table with all events, including the extended entries displayed via the Info button.
DiagBufferDrv.ExtendedBufferJScript	Supplies the dynamically generated JavaScript fragment required to display the table.
DiagBufferDrv.LText[]	Supplies an array that enables access to the entire text of the diagnostics buffer entry. The index matches the index of the diagnostics buffer entry. The individual elements of a diagnostics buffer entry (time, date, text, extended entry text) are separated by "/@@/".

The following variables can be used for direct access to the data of certain events in the drive diagnostics buffer:

Table 6-28 Variables of the DiagBufferDrv group - direct access

Variable	Description
DiagBufferDrv.EventCnt	Number of events currently in the drive diagnostics buffer
DiagBufferDrv.CplEventCnt	Event counter beyond the circular buffer limit During ramp-up, the counter is initialized with the current number of drive diagnostics buffer entries. Each time an entry is made, the value is incremented, even beyond the maximum number of drive diagnostics buffer entries.
DiagBufferDrv.Time[1] bis DiagBufferDrv.Time[n]	Time of each event
DiagBufferDrv.Date[1] bis DiagBufferDrv.Date[n]	Date of each event
DiagBufferDrv. Text[1] bis DiagBufferDrv.Text[n]	Text of each event **Note:** If the event text number and its parameters cannot be resolved, the number and parameters are output in HEX format. The variable in HEX format is a string of 20 hexadecimal characters (without separators).

Alarms group

Information about alarm table

The Alarms group provides information about the pending alarms. Access to all variables is read-only.

Table 6-29 Variables of the Alarms group

Variable	Description
Alarms.AlarmCnt	Number of alarms
Alarms.Table	HTML table with all pending alarms
Alarms.TableHead	Table header for the HTML table of pending alarms
Alarms.TableHeadBuffer	HTML table (header only) of the alarm buffer
Alarms.TableHeadUser	HTML table (header only) of the AlarmS
Alarms.TableBodyBuffer	HTML table (content only) of the alarm buffer
Alarms.TableBodyUser	HTML table (content only) of the AlarmS
Alarms.TableBuffer	HTML table of the alarm buffer
Alarms.UserAlarmCnt	Number of AlarmS

AlarmsDrv group

Information about drive alarm table

The AlarmsDrv group provides information about the active drive alarms. Access to all variables is read-only.

Table 6-30 Variables of the AlarmsDrv group

Variable	Description
AlarmsDrv.AlarmCnt	Number of drive alarms
AlarmsDrv.AlarmDsc	JavaScript code for the standard page **Alarms drive**
AlarmsDrv.Table	HTML table with all active drive alarms
AlarmsDrv.TableHead	Table header for the HTML table of active drive alarms

The SIMOTION C and P do not support the AlarmsDrv group.

Comparison with the device diagnostics of SIMOTION SCOUT

Comparison with device diagnostics in SIMOTION SCOUT

The variables described in this section are based on the view of the device diagnostics in SIMOTION SCOUT. The following figures show the connection between the "SIMOTION diagnostics" variables and the device diagnostics in SIMOTION SCOUT.

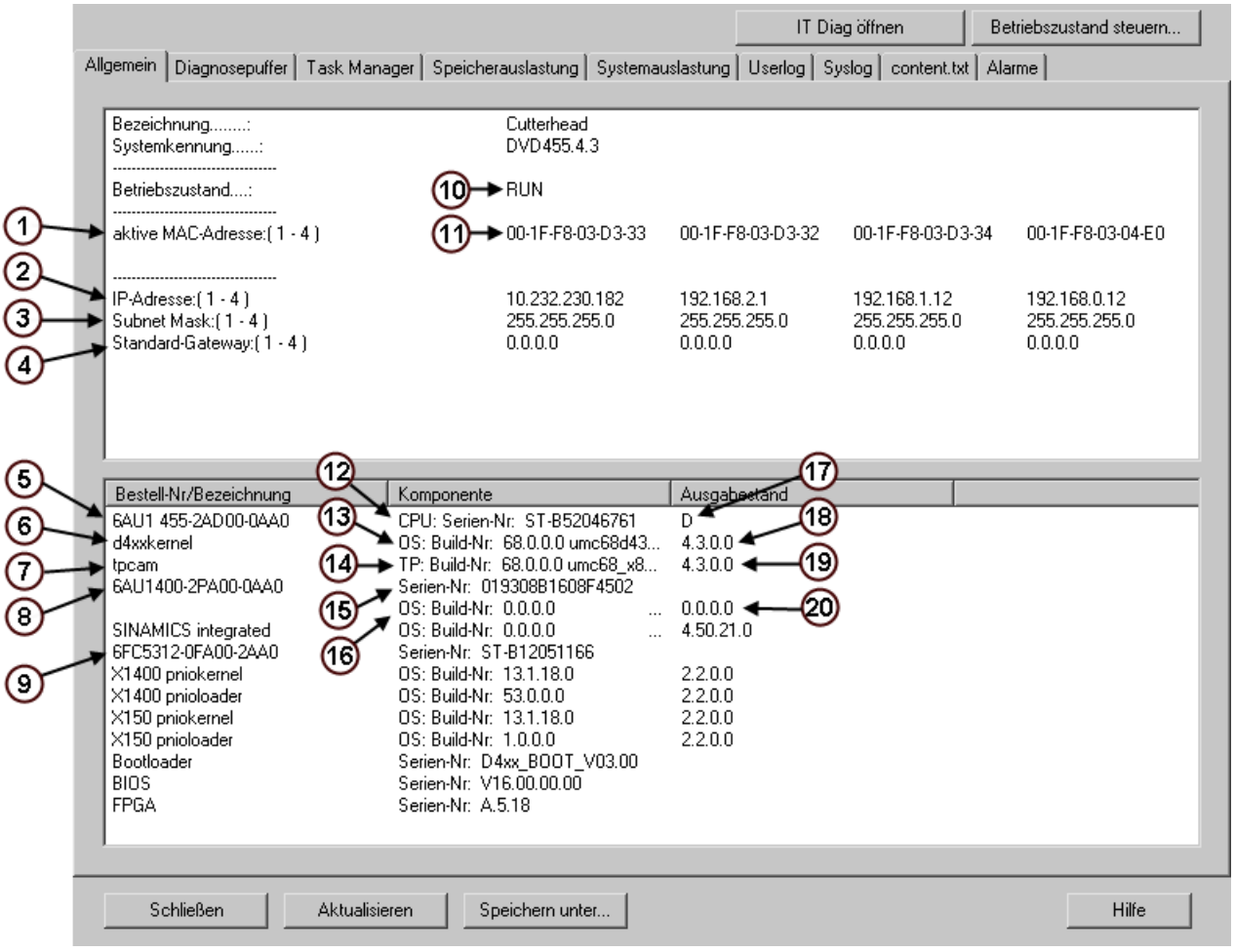


Figure 6-110 "General" device diagnostics

Table 6-31 Explanations

1	DeviceInfo.Active-MAC-0, ...-1, -2, -3	11	The active DeviceInfo MAC addresses 1-4
2	DeviceInfo.IP-Address-0, ...-1, -2, -3	12	CompInfo.Cpu.Serial-Nr
3	DeviceInfo.Subnet-Mask	13	CompInfo.Cpu.Build-Nr
4	DeviceInfo.Gateway	14	CompInfo.TP[1].Build-Nr
5	CompInfo.Cpu.MLFB	15	CompInfo.HW[1].Serial-Nr
6	CompInfo.Cpu.Kernelname	16	CompInfo.HW[1].Build-Nr
7	CompInfo.TP[1].Name	17	CompInfo.Cpu.Revision-Nr
8	CompInfo.HW[1].Firmwarename	18	CompInfo.Cpu.User-Version

9	CompInfo.HW[2].Firmwarename	19	CompInfo.TP[1].User-Version
10	DeviceInfo.BZU	20	CompInfo.Cpu.HW[1].User-Version

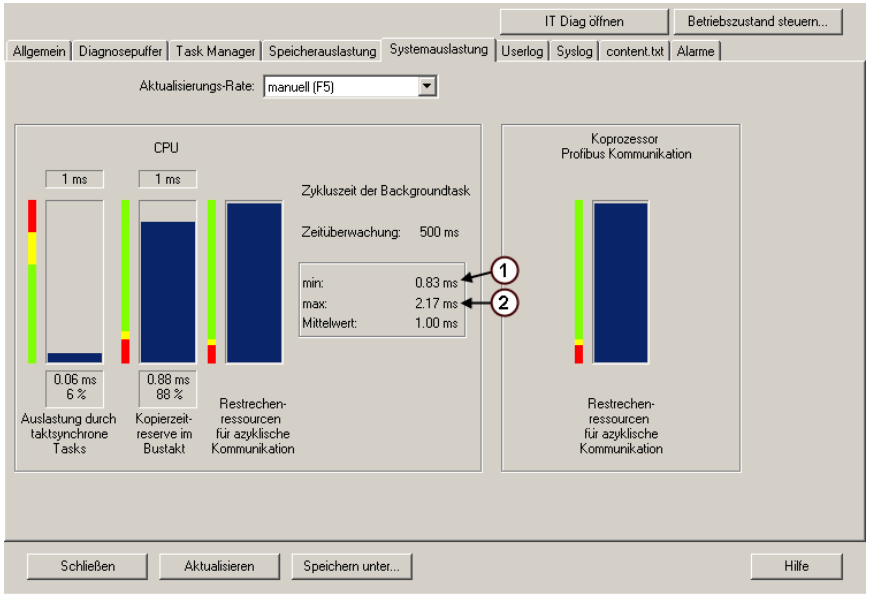


Figure 6-111 "System load" device diagnostics

Table 6-32 Explanations

1	CPULoad.Mintime
2	CPULoad.Maxtime

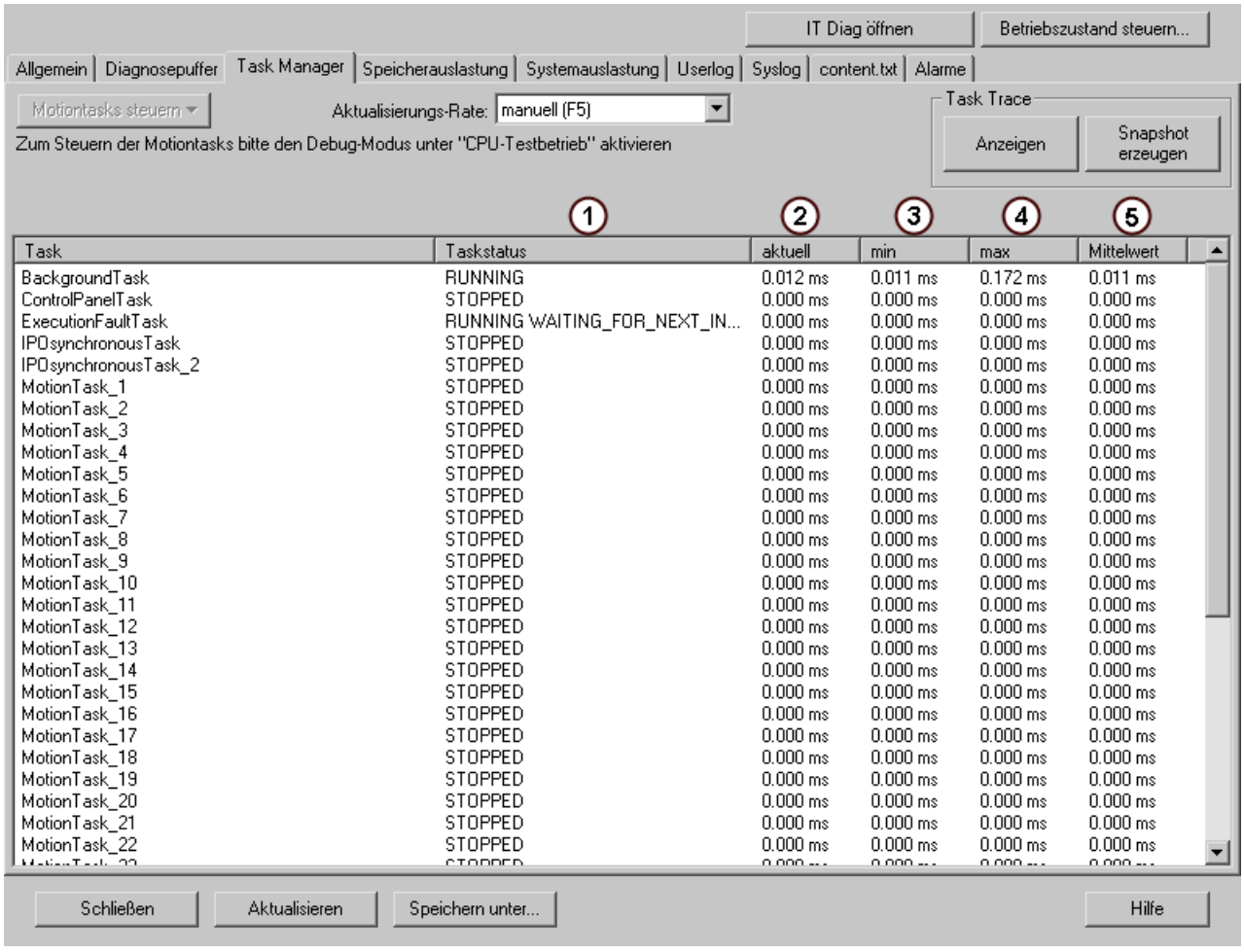


Figure 6-112 "Task runtimes" device diagnostics

Table 6-33 Explanations

1	TaskRT.MotionTask_11.Status
2	TaskRT.MotionTask_11.Actual
3	TaskRT.MotionTask_11.Min
4	TaskRT.MotionTask_11.Max
5	TaskRT.MotionTask_11.Average

UserConfig

User-defined variables

User-defined variables are declared in the UserConfig tag in the `<USERCONFIG>` file and can be read with the variable provider WebCfg.xml.

Some constant variables are predefined in the <USERCONFIG> tag. These variables are accessed via the **variable provider** UserConfig/constants.

Table 6-34 Overview of the predefined constant variables

Name	Type	Description
ForceUserMsgLanguageID	Integer (LCID)	Specifies the language to be used when importing user-defined messages (diagnostics buffer or AlarmS).Setting the language for AlarmS and user-defined diagnostics buffer messages (Page 3575)
WatchWritable	YES / NO Default setting: YES	Specifies whether watch tables may be edited and deleted on the standard pages.
BasicWatchWritable	YES / NO Default setting: YES	Specifies whether watch tables may be edited and deleted on the basic pages.
IncludeScriptsDirectly	YES / NO Default setting: YES	

See also

SIMOTION IT Configuration data (Page 3629)

<CONFIGURATION_DATA> (Page 3728)

MiniWeb

Variable provider MiniWeb

The variable provider MiniWeb contains variables of the basic settings of the Web server.

Cannot be configured by the user:

- MiniWeb_Build
- MiniWeb_Version
- Plattform
- SystemRoot
- Time
- UpTime
- WWWRoot

Can be configured in WebCfg.xml and via **Manage Config > SIMOTION IT > Serveroptions**:

- HTTP_PORT
- ALTERNATIVE_HTTP_PORT
- SSL_PORT
- ALTERNATIVE_SSL_PORT

Configurable in the HW Config dialog: **Device > Object properties > Ethernet extended / Web server** or **Settings**:

- SystemTime
- Date
- TIMEZONE

ITDiag

Variable provider ITDiag

Representation of Web server contents

The ITDiag provider represents the connection data of the Web server. The variables have mostly a diagnostic function, and are used by software developers and service personnel for performance and fault analysis.

Table 6-35 Overview of the variables provided by ITDiag

Name	Description
ActiveConnections	The number of connections on which data is currently being transferred (Request or Response).
MaxConnections	Maximum total number of possible connections for clients and servers. Remark: Client and server connections are each limited to a separate maximum number.
MaxConnectionsUsed	Maximum number of connections that have been open since the controller was switched on.
MaxIndisposableConnectionsUsed	Maximum number of simultaneously open connections without "SleepingConnections".
MaxSimultaneousConnections	Maximum number of connections that can be managed in the Select mechanism of the protocol stack.
OpenConnections	Number of connections that are currently open.
Overflows	Number of failed connection attempts since controller power-on.
SimultaneusConnections	Number of connections that are currently being managed in the Select mechanism of the protocol stack.
SleepingConnections	Number of connections that are still open because of a connection marked as "Keep-Alive". They are closed by the Web server as required.
WaitingConnections	The number of connections through which a complete Request, but not yet any response, have been transferred.
resetMaxUsedConnections	By writing "true" to this variable, the statistics variables can be reset.
MaxIndisposableConnectionsUsed-Time	The time when the MaxIndisposableConnectionsUsed occurred.
OverflowTime	Instant at which the last overflow occurred.

The information relevant to the user is shown on the Diagnostics (Page 3585) Web page.

Making unit variables available

To make variables available on the SIMOTION IT OPC XML DA server, you must declare them as VAR_GLOBAL.

Declaring unit variables in the interface

In the declaration table, you define the data type for each variable. Only variables declared as VAR_GLOBAL are available for OPC XML-DA.

The following figure shows an example of unit variable declarations in an MCC program.

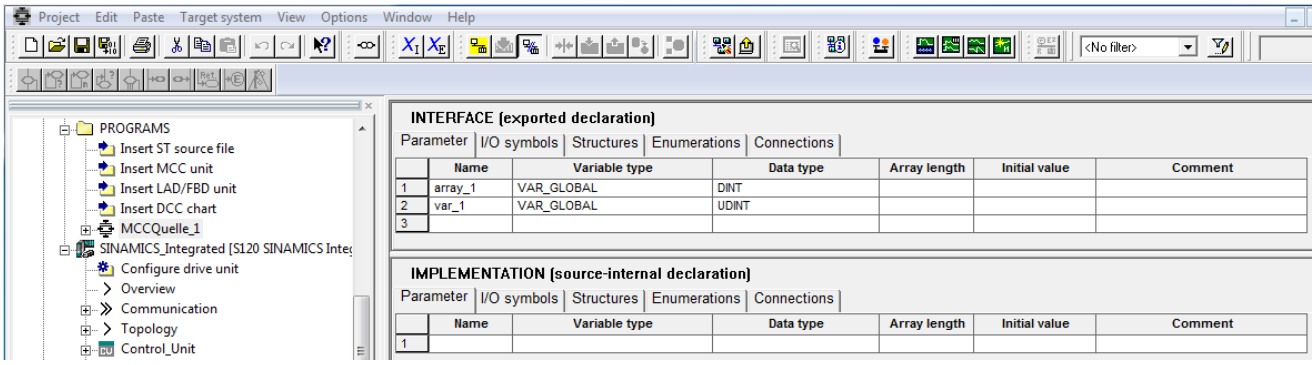


Figure 6-113 Declaring global variables

Permit OPC XML

To activate the variables for OPC XML DA, proceed as follows:

1. Open the **Properties** of the unit/source.
2. Select the **Compiler** tab.
3. Activate **Permit OPC-UA/-XML**, if it is not already activated (standard setting).

The following figure shows how to activate the unit variables from an MCC source.

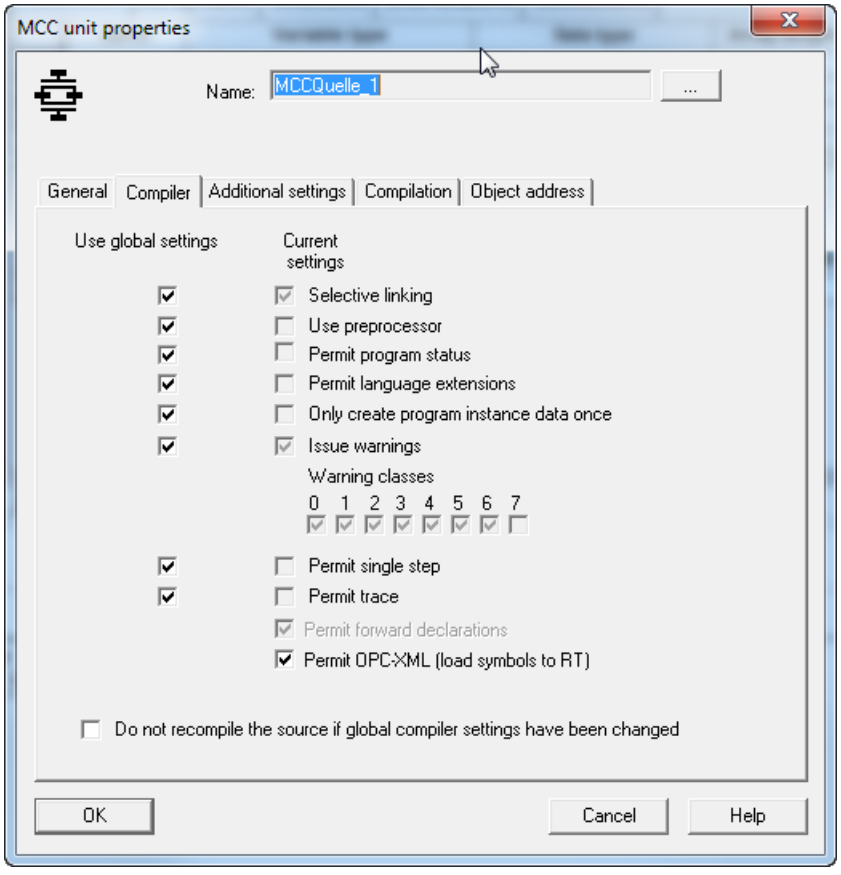


Figure 6-114 Making variables available for OPC XML DA

Note

The OPC XML activation applies also to variables in LAD/FBD and ST programs. To make variables available for OPC XML-DA in an ST program, they must be defined in a global variable block (VAR_GLOBAL and VAR_GLOBAL_RETAIN). This must be located in the interface section.

6.1.4.7 Secure Socket Layer

Introduction

The Secure Socket Layer protocol (SSL) enables encrypted data transmission between a client and SIMOTION. HTTPS access between the browser and the SIMOTION control is based on the Secure Socket Layer protocol.

Encrypted access to SIMOTION can take place via both SIMOTION IT OPC XML-DA and SIMOTION IT user-defined pages.

This section tells you which steps you need to follow to enable encrypted data communication between a client and SIMOTION. The possibilities are as follows:

1. You use the default configuration of the as-delivered condition.
2. You have a Certification Authority (CA) in your organization and the necessary key files are available.
3. You do not have a CA in your organization. In this case, you will need to create the key files yourself.

Note

HTTPS connections are supported in SIMOTION V3.2 and higher.

See also

Key files (V4.1 and higher) (Page 3703)

Encryption methods

You need two key files for the encryption method used by the Secure Socket Layer protocol. You need a public certificate and a private key. The pair of keys is created individually for each SIMOTION control. This ensures that the address requested matches the SIMOTION control accessed during HTTPS communication.

Note

Encrypted access to the SIMOTION control is only possible with the control identifier (name/IP address) specified when the key was created.

You can find further information about Secure Socket Layer certificates at http://www.verisign.de (http://www.verisign.de).

Key files (V4.1 and higher)

As delivered

So that you can access the SIMOTION controller via HTTPS in the delivery state of the SIMOTION IT diagnostics standard pages, a server certificate and a private key are supplied as a file on the device.

When you attempt HTTPS access using the key files supplied with the system, you will be warned that the certificate is unknown and that the current address of the controller does not match the name of the controller in the certificate.

Note

Secure data transmission

A HTTPS connection via the preinstalled certificate is not the most secure way of accessing the controller. The preinstalled certificate should therefore only be used if no self-created or purchased certificate can be used.

Creating key files with the script cert.pl (V4.1 and higher)

Overview

Note

HTTPS connections are supported as of SIMOTION V3.2.

If no Certification Authority (CA) is available in your organization, we recommend that you follow the steps described in the following section. The certificate and the key files are created with the OpenSSL tool and a Perl script cert.pl

Carry out the following steps:

No.	Working step	Remark
1.	Install a Perl runtime environment	If Perl is not installed
2.	Install OpenSSL	
3.	Create the certificate and key files with Perl script	
4.	Import the created certificate to the PC browser	This step must be performed once for each PC.

HTTPS access is available after the SIMOTION controller ramps up.

Installation of a Perl runtime environment

Install Perl if the Perl runtime environment is not present on your PC. You can download a free setup for Windows from the following websites, for example:

- http://www.activestate.com (http://www.activestate.com)
- http://www.perl.org (http://www.perl.org)

Installation of OpenSSL

You can download a free OpenSSL setup for Windows, for example, from the following website:

- http://slproweb.com/products/Win32OpenSSL.html (http://slproweb.com/products/Win32OpenSSL.html)

Application cert.pl

The cert.pl Perl script generates certificates for the controller. The script is on the AddOn-DVD in the \Addon\4_Accessories\SIMOTION_IT\6_Tools directory.

First, create a new directory <CertDir> (e.g. C:/cert) on your PC and copy the cert.pl file into it.

Call syntax cert.pl

```
Usage: perl cert.pl [-h][-?][-cert CertPath][-site <Site name>][-cpu
<CPU name>]
                  [-ip <IPAddr>[,<IPAddr>,...]][-ossl <path>][-tools
<path>]
                    [-d <duration>][-img <path>][-wcfg <WebCfgPath>]
                    [-ca][-srvn][-srvu][-ksize size]

Options:

  -cert <certpath>: Directory used for the creation of certificates;
must be given as absolute path e.G. C:\cert (default: current
directory)

  -site <site name>: Name of the site the cpu is belonging to

  -cpu <CPU name>: Name of the cpu

  -ip <IPAddr>[,<IPAddr>,...]: List of IP addresses belonging to 1
cpu (no spaces allowed)

  -ca: Create new root CA

  -srvrn: Create new server certificate

  -srvru: update existing server certificate

  -d <duration>: Duration of validity (in days)

  -tools <path>: Absolute Path to the tools dir containg eg. 7z.exe

  -img <path>: Path to the output dir (default: <certpath>)

  -e: Export the certificates of 1 cpu to the path specified by the -
img option

  -ossl <path>: Absolute Path to an openssl installation (eg. C:/
OpenSSL-Win64)

  -ksize <size>: Key size (default: 2048)

  -h: Print this help

  -?: Print this help

  -wcfg WebCfgFile: Use <WebCfgFile> as a template
```

The path to the OpenSSL installation is determined via the "OPENSSL_CONF" environment variable from the program. This environment variable is created during the installation of OpenSSL with a setup program. If the environment variable is not set, then the "-ossl" option must be used.

Creating a ZIP file for upload

If the created certificates are to be loaded into the controller afterwards via the standard page "Manage Config -> SIMOTION IT -> Certificates" , the ZIP tool `7-Zip` must be installed in addition. Download the program from the Internet (http://www.7-zip.org/download.html). After unpacking, copy the `7z.exe` program (`7za.exe` in older versions) to the `<certpath>` directory. Alternatively, when you call `cert.pl` , you can transfer the installation directory containing the `7z.exe` file with the option `-tools <toolpath>`.

See also

Importing the SSL certificate into the browser (Page 3709)

Creating a SSL certificate yourself

The cert.pl Perl tool can be used to generate the certificates required for customer systems (sites) and combine them into packages for loading.

Generation of root and server certificates

As of SIMOTION Version 4.4, there are two applications for which the tool can be used:

Automatic generation of the certificate

In the first application, the required server certificates and their private keys are generated automatically on first access to the controller via HTTPS. A root certificate and the associated private key are required for this purpose.
The root certificate and the associated private key are generated using the Perl tool.

Call: `perl cert.pl -ca -ossl C:/OpenSSL-Win64`

Name of the server certificate:	ITDiagRootCA.crt
Name of the private key:	ITDiagRootCA.key
Storage location in the file system:	<certpath>/CA

Note

UaExpert

Version 1.4.4 of the UaExpert functions only with a binary certificate in DER format. The certificate in DER format is stored in the /USER/SIMOTION/HMICFG/CERTSTORE/CA path in the "ITDiagRootCA.zip" ZIP file on the memory card of the controller.

The data of the certification institution is queried:

- Country (2-character code, e.g. DE)
- State (e.g. Bavaria)
- City (e.g. Erlangen)
- Company (e.g. MyCompany Corp.)
- Department (e.g. IT Development)

- Common name (e.g. ITDiagRootCA)
- E-mail (e.g. sepp@MyCompany.com)

Self-generated certificates

In this case, the required server certificates must be generated in addition to the root certificate.

Call:

```
perl cert.pl [-ca] [-cert <certpath>] [-site <sitename>] -cpu
<cpuname> -ip <IP-Addr1>,<IP-Addr2>,.... -srvn -ossl <opensslpath>
```
or
```
perl cert.pl [-ca] [-cert <certpath>] [-site <sitename>] -cpu
<cpuname> -ip <IP-Addr1>,<IP-Addr2>,.... -opcuacert -ossl
<opensslpath>
```

Name of the generated root certificate	ITDiagRootCA.crt, bzw. ITDiagRootCA.zip
Name of the private key	ITDiagRootCA.key
Storage location in the file system	<certpath>/CA

Name of the generated server certificates	<IP-Addr>.SSL.crt (z.B. 192.168.2.90.SSL.crt)
Name of the private key	<IP-Addr>.SSL.key (z.B. 192.168.2.90.SSL.key)

or

Name of the generated server certificates	<IP-Addr>_OPCUA.crt (z.B. 192.168.2.90_OPCUA.crt)
Name of the private key	<IP-Addr>_OPCUA.key (z.B. 192.168.2.90_OPCUA.key)
Storage location in the file system	<certpath>/<sitename>/<cpuname>/<IP-Addr>

The root certificate will only be generated if none already exists. For all subsequent calls, the existing root certificate is used to sign the newly generated server certificates. The generation of a new root certificate can be forced with the -ca option.

The list of IP addresses (<IP-Addr1>, <IP-Addr2>) must not contain any blanks. This also applies to all other parameters.

The data of the applicant is queried when creating the first server certificate of a site. If `-site` was not specified, of a CPU, the applicant data is queried:

- Country (2-character code, e.g. DE)
- State (e.g. Bavaria)
- City (e.g. Erlangen)
- Company (e.g. MyCompany Corp.)
- Department (e.g. IT Development)
- E-mail (e.g. sepp@MyCompany.com)

Note

Validity duration of the certificates

The default validity is 30 years (effectively infinite).

The d option generates certificates with a shorter validity. In this case, HTTPS communication will no longer function after the validity has expired.

The user is responsible for installing the new valid certificates on all affected controllers.

Update of existing server certificates

If one of the parameters essential for the generation of the server certificates (e.g. the server certificate, the lifetime or the configuration) changes, an update can be started for the server certificates.

Call: `perl cert.pl [-cert <certpath>] [-site <sitename>] [-cpu <cpuname>] -srvu -ossl <opensslpath>`

This call also affects the certificates for OPC UA.

If the `-cpu` parameter is missing, all certificates of the CPUs belonging to the site are renewed.

If the `-site` parameter is also missing, all certificates are renewed.

Deletion of existing server certificates

Server certificates can be deleted:

Call: `perl cert.pl [-cert <certpath>] [-site <sitename>] [-cpu <cpuname>] [-ip <IP-Addr1>,<IP-Addr2>,....] -svrr -ossl <opensslpath>`

Export of existing server certificates

The generated certificates can be exported for each CPU.

Call: `perl cert.pl [-cert <certpath>] [-img <path>] [-site <sitename>] -cpu <cpuname> [-ip <IP-Addr1>,<IP-Addr2>,....] -e -ossl <opensslpath>`

The path to the exported images can be specified with the `-img` option.

Storage location in the file system: `<path>/images/<sitename>/<cpuname>`

A directory structure can be found at `<imgpath>/images/<sitename>/<cpuname>/image` which can be copied to the `/USER/SIMOTION/HMICFG` directory of the CF card. An upload-capable ZIP archive (<cpuname>.zip) is also generated under `<imgpath>/images/<sitename>/<cpuname>` if the zipper 7z.exe (7za.exe in older versions) is in `<toolspath>` (option `-tools <toolspath>`).

- The archive can be unpacked in the HMICFG directory. Any existing server certificates have to be removed. To do this, the complete `/USER/SIMOTION/HMICFG/certstore/servercerts` directory is deleted. The controller then has to be restarted.
- The server certificates can also be loaded to the CPU via the **Certificates** website at **Manage Config**. Unnecessary files and directories are deleted and a restart of the Web server triggered.

SIMOTION versions prior to Version 4.4

For SIMOTION versions prior to Version 4.4, the functionality of the tool is retained.

Generated server certificates are entered in a copy of a template of the WebCfg.xml file.

The template is sought in one of the following directories in the specified order:

```
- –wcfg Option
- <certpath>/<sitename>/<cpuname>/<ipaddr>
-<certpath>/<sitename>/<cpuname>
- <certpath>/<sitename>
-<certpath>
```

See also

7-Zip Download (http://www.7-zip.org/download.html)

Importing the SSL certificate into the browser

If you use SSL with your own certification authority, you will need to prepare your PCs for communication with the SIMOTION controller. To do this, the "ITDiagRootCA.crt" root certificate must be included in the list of certificates in your browser.

Please follow the instructions of your browser when importing the certificate.

Various types of certificate use:

1. Browser import of the "ITDiagRootCA.crt" root certificate (e.g. from the "<certpath>\images \<site>\<cpu>\image\certstore\CA" directory).
2. If an HTTP connection is established to the device, the root certificate can be saved via the **Manage Config > Certificates** page with the **Get root certificate** button.
3. During HTTPS access to a device without previous import of the root certificate, a prompt appears in the browser as to whether the associated server certificate is to be imported. This import enables the secure connection to **one** device and must be repeated for all other devices. For this reason, import of the root certificate is always preferred.

6.1.4.8 Trace Viewer

Trace display

SIMOTION IT allows traces to be displayed on the Trace Viewer (Page 3602) Web page. Alternatively, you can use the WebTraceViewer (Page 3719) external program to display traces.

The Trace Viewer shows only completed measurements. Measurements are imported directly from the device or a file.

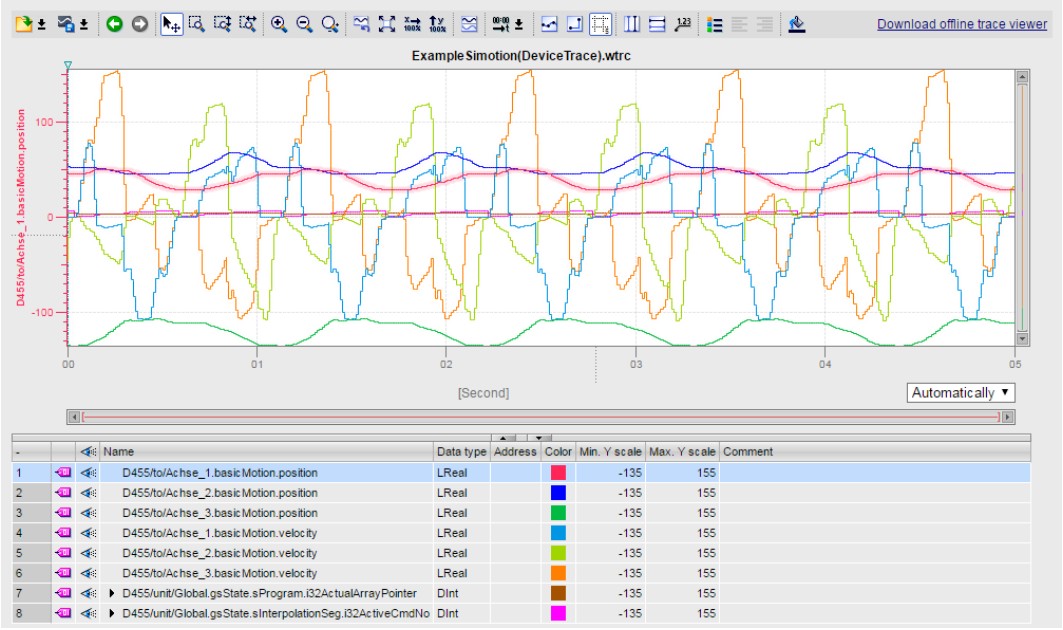


Figure 6-115 Trace Viewer working area

The working area of the Trace Viewer is divided into two areas. The upper area shows the curve diagram (Page 3711); the lower area shows the signals table (Page 3717).

Data import

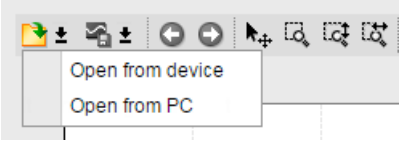


Figure 6-116 Trace Viewer data import selection

Open from device imports data directly from the device in WTRC format.

Open from PC imports data from a file in WTRC or CSV format. The Device Trace and System Trace Web pages provide with the **Get trace data** button the possibility to save trace data in WTRC format.

Trace files exported in CSV format from the TIA Portal can also be imported.

Conversion of trace data

1. WTRC data loaded in Trace viewer can be saved locally as CSV and WTRC.
2. CSV data loaded in Trace viewer can be saved only as CSV (copy).

The WTRC format is identical with that of the WebTraceViewer (Page 3719). The CSV format can be displayed only with the Trace Viewer.

Files can be imported by drag-and-drop to the working area of the Trace Viewer.

Curve diagram

The curve diagram displays the selected signals of a recording. Bits are shown in the lower diagram as a bit track. You can adjust the display of the signals in the signal table and with the toolbar of the curve diagram.

Setting options and displays in the curve diagram

The following figure shows a sample representation in SIMOTION IT.

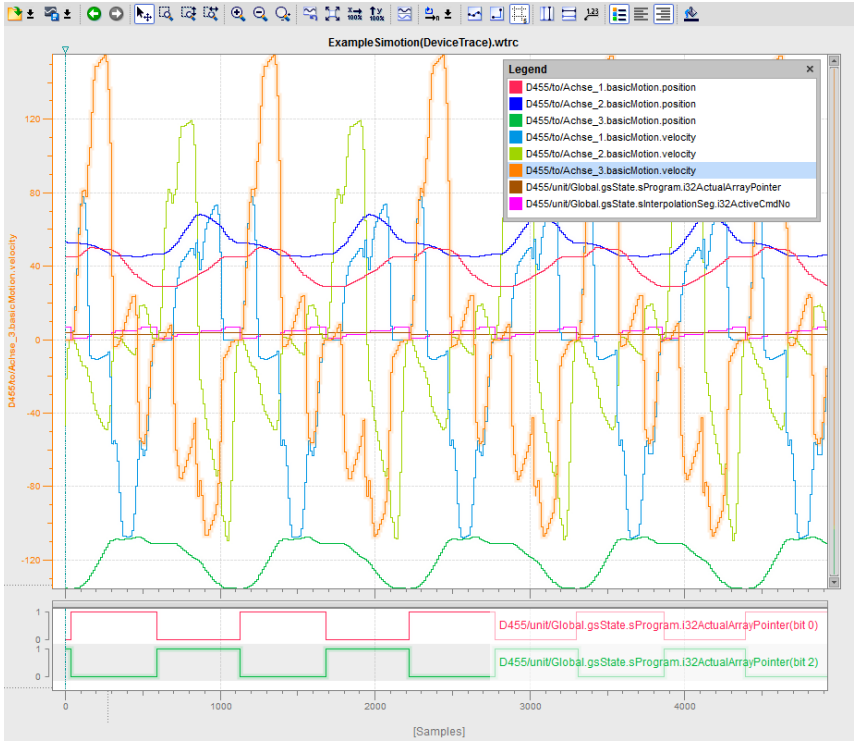


Figure 6-117 Trace Viewer legend

The scale in the diagram applies to the selected (orange background) signal in the legend. The vertical scale is also displayed in the color of the selected signal. The legend can be moved and its size can be adjusted with the mouse. Only signals, but no bits, are displayed in the legend.

The ▽ symbol shows with a vertical line the trigger instant with the trigger time of the device. The trigger time is displayed in a tooltip when hovering with the mouse.

Shortcut menus

A signal in the curve diagram can be selected with the cursor when **Move view** is activated in the toolbar. A right-click opens the shortcut menu that allows the curve to be zoomed or hidden automatically for the Y axis.

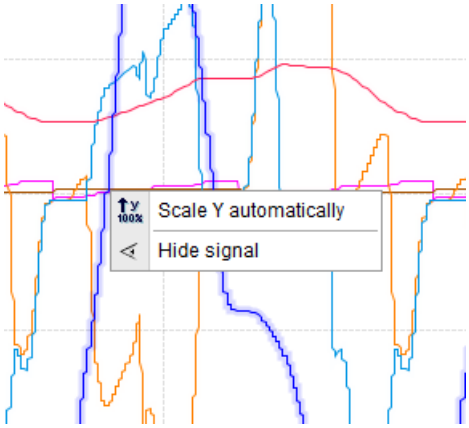


Figure 6-118 Signal shortcut menu

Further shortcut menus are provided in legends, in the signal table (depending on the signal type) and in the curve diagram.

Functions using the mouse wheel

The following table shows which functions are possible in the curve diagram using the mouse wheel:

Function of the mouse wheel	Description
Move the curve diagram vertically	Turning the mouse wheel moves the display in the upper curve diagram up or down. If the signals are arranged in tracks, the display of the group located below the cursor is moved. The cursor must be positioned above the upper curve diagram. Keyboard: Up and down arrow keys. Pg Up and Pg Dn.
Move the curve diagram horizontally	Turning the mouse wheel with the <Shift> key pressed down moves the display in the curve diagram to the left or the right. The cursor must be positioned above the curve diagram. Keyboard: Right and left arrow keys Home and End.
Zoom in and Zoom out of the curve diagram	Turning the mouse wheel with pressed <Ctrl> key zooms the display in the curve diagram and the display of the bit tracks. The cursor position is the starting point for zooming in or out. The cursor must be positioned above the curve diagram. Keyboard: Press the <+> or <-> key.

Functions in the vertical scale

The following table shows which mouse actions are possible in the vertical scale:

Function	Description
Scroll mouse wheel	Turning the mouse wheel scrolls the display. Turning the mouse wheel with pressed <Ctrl> key zooms the display in the curve diagram.
Zoom mouse wheel	Turning the mouse wheel with pressed <Ctrl> key zooms the display of the selected signal in the curve diagram.

Function	Description
Zoom left mouse button	Moving the mouse up or down with pressed left mouse button zooms the signal.
Zoom <Shift> left mouse button	Moving the mouse up or down with pressed left mouse button and pressed <Shift> key zooms the signal, whereby the upper and lower limits are synchronized.

Toolbar of the curve diagram

Tools are available for adapting the display via buttons.

The following table shows the functions of the buttons:

Symbol	Function	Description
	Load measurements	Measurements can be imported from the device memory (WTRC format) or a file (WTRC, CSV format).
	Save measurements	The displayed measurement can be saved in a file. Measurements that were imported in WTRC format can also be saved in WTRC and in CSV format. Measurements that were imported in CSV format can be saved only in this format.
	Undo the last action.	Undo the last action that was performed. Actions that can be undone: Zoom, colors, X axis unit, signal selection, display and hide. If several zoom functions have been executed, they can be undone step-by-step.
	Redo the last action	Redo the last undone zoom function. If several zoom functions have been undone, they can be redone step-by-step.
	Move view	Move the display of the curve diagram Keyboard: <M>, <Esc> Pressing these keys deactivates the keys of the zoom toolbar which causes the started zoom actions to be cancelled. If "move view" is active and the <Ctrl> key is pressed, the "zoom selection" is then activated.
	Zoom selection	Select an arbitrary range with the mouse button pressed. The display is scaled to the range selection. Keyboard: <Z>
	Vertical zoom selection	Select a vertical range with the mouse button pressed. The display is scaled to the range selection. Keyboard: <V>
	Horizontal zoom selection	Select a horizontal range with the mouse button pressed. The display is scaled to the range selection. Keyboard: <H>
	Zoom in	Enlarge the display. The ranges of the time axis and value axis are reduced every time the button is clicked. The curves are displayed larger. Keyboard numeric keypad: <+>

Symbol	Function	Description
	Zoom out	Reduce the display. The ranges of the time axis and value axis are increased every time the button is clicked. The curves are displayed smaller. Keyboard numeric keypad: <->
	Zoom within the graphic	Moving the pressed cursor within the graphic changes the zoom area. The zoom effects differ depending on the position of the cursor and the direction in which the cursor is moved: • To zoom the graphic from the left-hand side, click in the left-hand half of the graphic and move the mouse in the desired direction. • To zoom the graphic from the upper side, click in the upper half of the graphic and move the mouse in the desired direction. • The action for the lower and right-hand sides is identical. Pressing the Shift key synchronizes these actions.
	Standard view of the zoom settings	The standard view of the zoom settings is restored. All displayed signals are scaled in the visible area, whereby relative positionings (vertical separations) are retained.
	Display all	Scale the display of the available data so that the entire time range and all values are displayed.
	Display the complete time range	Scale the display so that the values for the complete time range are displayed in the currently displayed value range. All signal points are adapted (or scaled) to the visible width irrespective of whether the signal was selected. This is equivalent to the state after the original loading of a file.
	Automatic scaling of the value axis	Scale the display so that all values are displayed for the currently displayed time range. The relative scaling ratio between the signals changes.
	Arrange in tracks	Display the signals in separate tracks. The signals are arranged below each other with the associated value axes. Signal groups are displayed in the same track. This setting does not affect the display of the bit tracks.
	Unit change of the time axis	Change the unit of the time axis. The following units are adjustable: • Measurement points • Time (relative time related to the trigger time) • Absolute time Only the relative time can be displayed for system trace recordings. A change is not possible in this case.
	Display measurement points	The measurement points are displayed as small circles on the curves.

Symbol	Function	Description
	Interpolation mode	Change the interpolation mode.
	Display grid	Change the grid display. The brightness setting of the grid can be changed in the shortcut menu or with the <B> key.
	Display vertical measurement cursors	Display the vertical measurement cursors. The vertical position of the two measurement cursors can be moved with the mouse. The associated measured values and the difference of the measurement cursors corresponding to the position are shown in the signal table. Selecting a measurement line with the mouse causes the line to be displayed solid. A selected measurement line can be moved from measurement point to measurement point with the keyboard arrow keys. If the <Ctrl> key is pressed in addition, the line is moved pixel-by-pixel.
	Display horizontal measurement cursors	Display the horizontal measurement cursors. The horizontal position of the two measurement cursors can be moved with the mouse. The selected measurement line can also be moved pixel-by-pixel with the arrow keys.
	Display the measured cursor values	Change the display of the measured cursor values in the curve diagram.
	Display chart legend	Show or hide the legend in the curve diagram and the bit track labels.
	Display the bit-track designation left-justified	Display the bit-track designations on the left-hand side of the curve diagram.
	Display the bit-track designation right-justified	Display the bit-track designations on the right-hand side of the curve diagram.
	Change background color	Switch between three different background colors (white, dark blue, black).

Measurement cursor pane

The "Measurement cursor" pane shows the position of the measurement cursor in the curve diagram and the values at the intersection points.

Setting options and displays of the "Measurement cursor" pane

The figure below shows the "Measurement cursor" pane:

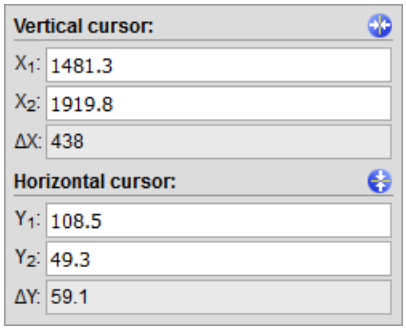


Figure 6-119 "Measurement cursor" pane

The blue buttons at the top right return the measurement cursor to the visible area.

The following table describes the settings and displays:

Setting/display		Description
Horizontal measurement cursor		
	Y1	Position of first measurement cursor The value states the position in relation to the scale of the signal currently selected. You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	Y2	Position of the second measurement cursor The value states the position in relation to the scale of the signal currently selected. You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	ΔY	Display of the position difference between the first and the second measurement cursor
Vertical measurement cursor		
	X1	Position of first measurement cursor You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	X2	Position of the second measurement cursor You also have the option of specifying a new position for the measurement cursor in this entry field for moving with the mouse.
	ΔX	Display of the position difference between the first and the second measurement cursor

Measurement cursor and curve diagram example

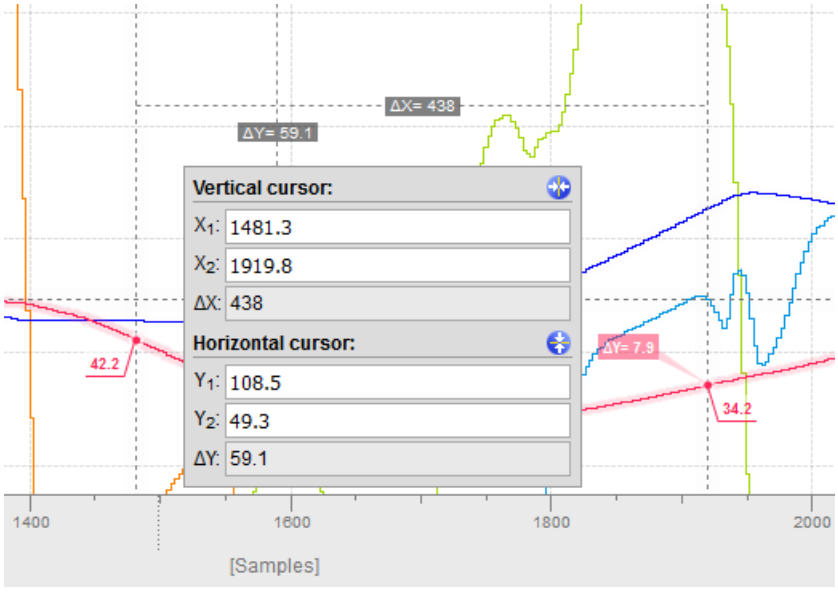


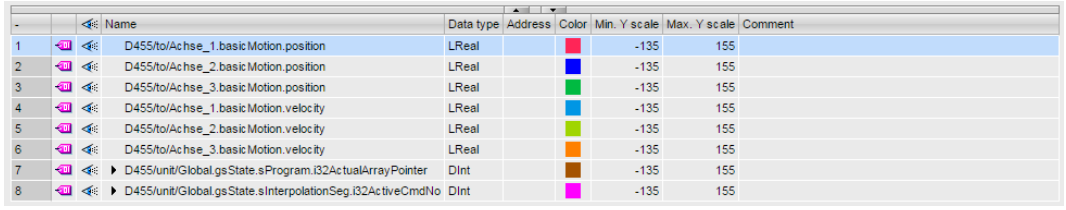
Figure 6-120 Measurement cursor and curve diagram pane

Signal table

The signal table lists the signals of the selected measurement and provides setting options for some properties.

Setting options and displays in the signal table

The following figure shows a sample representation of the signal table:



-		Name	Data type	Address	Color	Min. Y scale	Max. Y scale	Comment
1		D455/to/Achse_1.basicMotion.position	LReal			-135	155	
2		D455/to/Achse_2.basicMotion.position	LReal			-135	155	
3		D455/to/Achse_3.basicMotion.position	LReal			-135	155	
4		D455/to/Achse_1.basicMotion.velocity	LReal			-135	155	
5		D455/to/Achse_2.basicMotion.velocity	LReal			-135	155	
6		D455/to/Achse_3.basicMotion.velocity	LReal			-135	155	
7		D455/unit/Global.gsState.sProgram.i32ActualArrayPointer	DInt			-135	155	
8		D455/unit/Global.gsState.sInterpolationSeg.i32ActiveCmdNo	DInt			-135	155	

Figure 6-121 Trace viewer Signal table

Clicking the symbol in the table header shows or hides the signals and bit tracks.

The following table shows the settings and displays of the recorded signals:

Column	Description
	Static display of the signal symbol
	Selection for the display in the curve diagram The point indicates that at least one bit has been selected for display as bit track for the signal in the bit selection.

Column	Description
"Name"	Signal name display A click on the name of a displayed signal updates the scale in the curve diagram. The scale assumes the set color. The name is formed from the device name and the signal name.
▶	Open bit selection For simple integer data types, individual bits can be selected for display as bit track in the lower curve diagram: Example of an opened bit selection for the DInt data type:  Trace viewer Signal table bit display Select or deselect the associated bit for display by clicking the symbol or from the shortcut menu.
"Data type"	Display the data type
"Address"	Display the address of the signal The address is empty for a SIMOTION trace. This column can be filled for a CSV import from the TIA Portal.
"Color"	Display and setting option for the signal color. Clicking the color box opens a dialog for the color selection in which any color can be set.
"Min. Y-Scale"	Display the minimum value for scaling the signal
"Max. Y-Scale"	Display the maximum value for scaling the signal
"Y(t1)"	Display only for activated vertical measurement cursor
"Y(t2)"	Display only for activated vertical measurement cursor
"ΔY"	Display only for activated vertical measurement cursor
"Comment"	Display a comment for the signal

Shortcut menu commands

The following table shows the shortcut menu commands of the signal table:

Shortcut menu command	Description
"Scale Y automatically"	Scales the selected signal on the Y axis.
"Show signal"	Displays the selected signals in the curve diagram.
"Hide signal"	Hides the selected signals in the curve diagram.
Bit signals	
"Show bit"	Displays the selected bit signals in the curve diagram.
"Hide bit"	Hides the selected bit signals in the curve diagram.

WebTraceViewer

The WebTraceViewer PC program enables the trace data to be displayed.
The **GetWebTraceViewer** link can be used to save the WebTraceViewer on the PC. This link is not available with SIMOTION C modules. Alternatively, you can copy the WebTraceViewer from the Addon DVD.

This program is able to graphically display the data saved in a WTRC file.

As of SIMOTION V4.4, you can also load and display WTRC files in SIMOTION SCOUT.

Functions

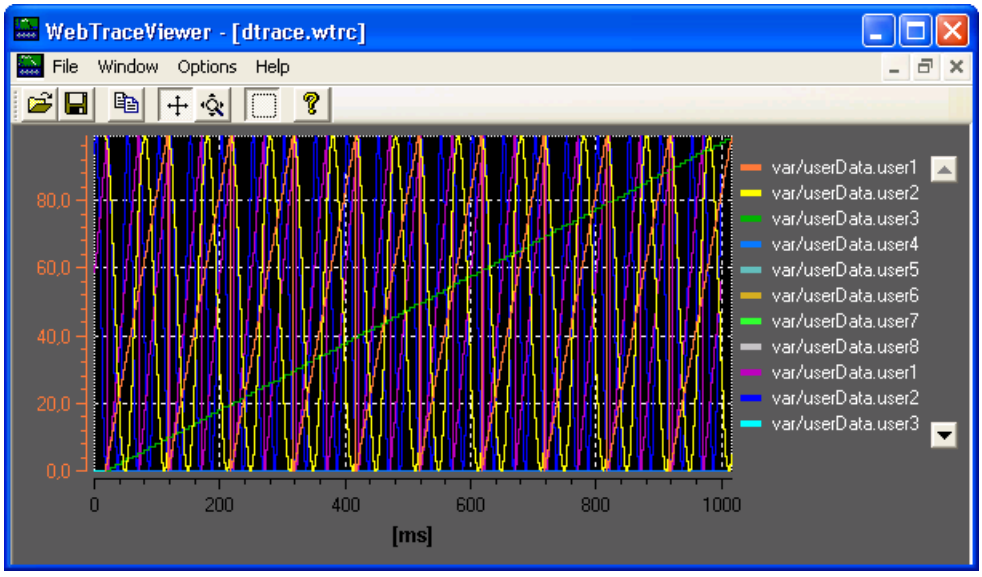


Figure 6-122 WebTraceViewer

Functions of the buttons

1. Open file: Enables you to open WTRC files.
2. Save file: Enables you to save WTRC files.
3. Copy: Copies the content of the current WTRC window to the clipboard in bitmap format. This enables the graphic to be copied to a word processing program, for example.
4. Scroll mode: Enables you to shift the visible area of the graphic using the mouse.
5. Zoom mode: Enables you to expand and compress the graphic using the mouse.
6. Selection mode: If this button is pressed, only a rectangular area of the graphic can be selected. Buttons 4 and 5 can then no longer be used.

Files

The **File Export** menu item allows you to save the trace data in CSV format so you can import it into a spreadsheet, for example.

Defective WTRC files

If WebTraceViewer imports a defective file, it provides information about the error.

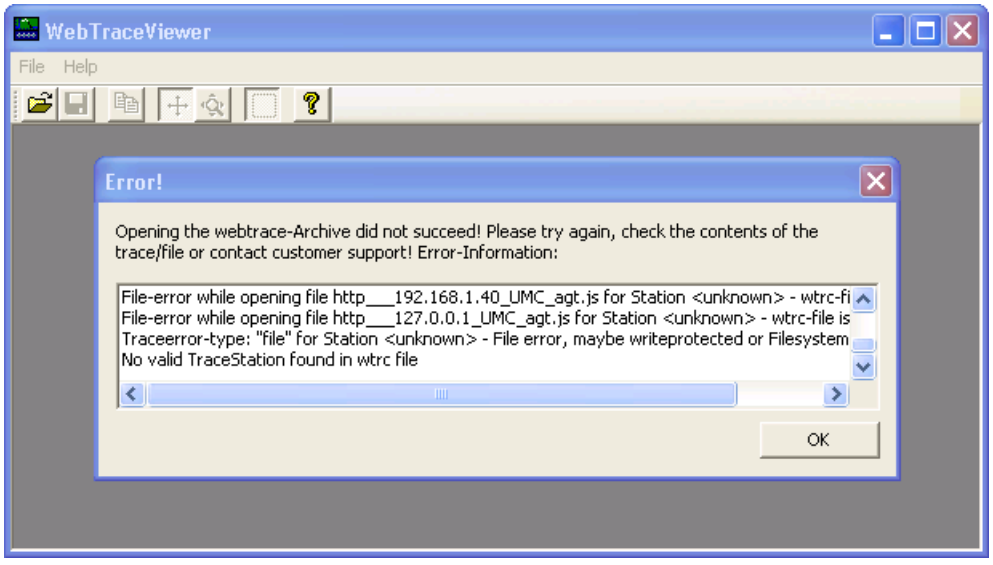


Figure 6-123 WebTraceViewer with faulty WTRC file

Note

The WebTraceViewer requires the "MS Visual C++ 2008 Redistributable Package" or an installed MS Visual Studio 2008 for program execution.

The "MS Visual C++ 2008 Redistributable Package" can be downloaded from the Microsoft Web page. It can also be found on the SIMOTION SCOUT Installation DVD "VOL1\Disk1\Setup \vcredist_2008".

6.1.5 List of abbreviations/acronyms

Abbreviations

CA	Certification Authority
CSS	Cascading Style Sheets
CSV	Character Separated Values
DO	Drive Object (Drive object)
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
FTP	File Transfer Protocol
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

HTTPS	Secure HTTP
JS	JavaScript
MWSL	MiniWeb Server Language
OPC	Denotes a standard interface for communication in automation technology. See OPC Foundation (https://opcfoundation.org/)
OPC XML-DA	OPC XML Data Access
SSI	Server Side Includes
SSL	Secure Socket Layer
TO	Technology Object (Technology object)
TVS	Trace Via SOAP
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation

6.1.6 Appendix

6.1.6.1 WebCfg.xml tags and attributes

SIMOTION IT diagnostics files

DIAGURLS.TXT

Structure of the DIAGURLS.TXT file

DIAGURLS.TXT contains the names of the SIMOTION IT diagnostics pages that are backed up when the diagnostics button is pressed or the pages are requested via **Diagnostics > Diagnostics files > Create general diagfiles**. The file is in directory /HMI/SYSLOG/DIAG and can be expanded with further URLs if necessary.

Here is an example of how this file might look like in the delivery state:

```
alarms.mwsl
alarmsdrvifrm.mwsl
alarmbufifrm.mwsl
devinfo.mwsl
basic/b_extdiag.mwsl
basic/b_diagbufdrv.mwsl
diagnost.mwsl
ipconfig.mwsl
mempool.mwsl
start.mwsl
taskrunt.mwsl
timezone.mwsl
```

Content of the file DIAGURLS.TXT

See also

Diagnostic files (Page 3606)

BASE

<BASE>

Tag	<BASE>
	The link lists for user-defined HTML pages are stored in the <BASE> tag of WebCfg.xml .
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE ALIAS="/"> [...] <myIndex.mwsl.cms ALIAS="mydir/myIndex.mwsl.cms" /> [...] </BASE> [...] </SERVERPAGES>

ALIAS attribute

Tag	Any node: <BASE> and all child nodes	
Attribute	ALIAS	The ALIAS attribute is a link to the physical file system, relative to the WWWRoot path /USER/SIMOTION/HMI. The file name must be identical to the file name in the ALIAS; otherwise, the file will not be found. Each data node of the XML file system can have a ALIAS attribute, including the <BASE> node. The <BASE> node corresponds to the WWWRoot of the file system.
Example	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <SERVERPAGES version="04.50"> [...] <BASE> <myfile.mwsl.cms ALIAS="/FILES/myfile.mwsl.cms" REALM="Administrator" READ="Administrator" WRITE="Administrator" MODIFY="Administrator" /> </BASE> [...] </SERVERPAGES> In this example, the myfile.mwsl.cms file can now be called via the following URL: http://<IP-Address>/myfile.mwsl	

BROWSEABLE attribute

Note

Changed behavior as of Version 4.4

As of version 4.4, the BROWSEABLE attribute no longer has any effect.

Tag	Any node: <BASE> and all child nodes or as a global switch, via the <BROWSEABLE> tag.	
Attribute	BROWSEABLE	BROWSEABLE can have the value "true" or "false". When a client accesses this link, a directory view of the directory is created. Navigation from this directory to subdirectories is also possible. Other higher-level directories can also be navigated to if browsing is also permitted for them. Provided you have sufficient rights, you can send, receive, and delete files as well as create and delete directories.
Example	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <SERVERPAGES version="04.50"> [...] <BASE> <FILES ALIAS="FILES/" BROWSEABLE="true" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone"> <myFile ALIAS="/FILES/myfile.mwsl.cms" BROWSABLE="true" REALM="Administrator" READ="Administrator" WRITE="Administrator" MODIFY="Administrator" /> </FILES> </BASE> [...] </SERVERPAGES>	

MODIFY attribute

Note

Changed behavior as of Version 4.4

As of version 4.4, the MODIFY attribute no longer has any effect.

Tag	Any node: <BASE> and all child nodes	
Attribute	MODIFY	If a directory has a MODIFY attribute and the logged-in users are members of one of the specified groups, they may perform all write operations in this directory. They may • Create new directories • Overwrite files • Delete files • Create new files Users must, of course, have READ rights as well (otherwise, they would not have access to the directory anyway).

READ attribute

Tag	Any node: <BASE> and all child nodes	
Attribute	READ	If a READ attribute is specified for a directory, the user must be a member of one of the groups specified for the READ attribute. With READ , several groups can be specified. These must be separated with commas and no Whitespace characters may be used.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE ALIAS="/"> <FILES ALIAS="FILES/" BROWSEABLE="true" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone"> <www ALIAS="/WebPages/" BROWSEABLE="true" READ="Administrator" WRITE="FileAdministrator" /> </FILES> <Test.mwsl.cms ALIAS="/Tests/Test.mwsl.cms/"/> <XMLDir> </XMLDir> </BASE> [...] </SERVERPAGES>	

REALM attribute

Tag	Any node: <BASE> and all child nodes	
Attribute	REALM	The REALM attribute is used to set up a secure area. REALM may only contain one group name. The REALM attribute enables one login for all users of a group. For all users that do not belong to this group, access is blocked.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE ALIAS="/"> <FILES ALIAS="FILES/" BROWSEABLE="true" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone"> <www ALIAS="/WebPages/" REALM="Anyone" BROWSEABLE="true" READ="Administrator" WRITE="FileAdministrator" /> </FILES> <Test.mwsl.cms ALIAS="/Tests/Test.mwsl.cms/" /> <XMLDir> </XMLDir> </BASE> [...] </SERVERPAGES>	

WRITE attribute

Tag	Any node: <BASE> and all child nodes	
Attribute	WRITE	If a directory has a WRITE attribute and the logged-in users are members of one of the specified groups, they may only create new files in this directory. They may • Not create any new directories • Not overwrite any files • Not delete any files • Create new files Users must, of course, have READ rights for the directory as well (otherwise, they would not have access to the directory anyway).
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE ALIAS="/"> <FILES ALIAS="FILES/" BROWSEABLE="true" REALM="Anyone" READ="Anyone" WRITE="Anyone" MODIFY="Anyone"> <www ALIAS="/WebPages/" BROWSEABLE="true" READ="Administrator" WRITE="FileAdministrator" /> </FILES> <Test.mwsl.cms ALIAS="/Tests/Test.mwsl.cms/"/> <XMLDir> </XMLDir> </BASE> [...] </SERVERPAGES>	

<CONFIGURATION_DATA>

Tag	<CONFIGURATION_DATA>
	Each module provides the option of defining module-specific configuration data within this tag. The format of the individual items of configuration data depends exclusively on the modules. Therefore, it cannot be described in general terms.
Example	<SERVERPAGES> [...] <CONFIGURATION_DATA> <USERCONFIG> [...] <IncludeScriptsDirectly>NO</IncludeScriptsDirectly> <!-- Add your constants here --> <ForceUserMsgLanguageID>1031</ForceUserMsgLanguageID> </USERCONFIG> </CONFIGURATION_DATA> [...] </SERVERPAGES>

<DEFAULTDOCUMENT>

Tag	<DEFAULTDOCUMENT>
	Specification of the document that is to be displayed if the URL received from the browser does not contain explicit page information. This is often called Default.mwsl or Index.mwsl There can be only one default document. If no default document is found and file browsing is permitted, the directory itself is returned.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE> [...] </BASE> <SERVEROPTIONS> <DEFAULTDOCUMENT VALUE="Default.mwsl" /> [...] </SERVEROPTIONS> [...] </SERVERPAGES> If, for example, the URL http://<IP-Address>/MyDir is used to query a directory, the Web server appends the file name "Default.mcs" to the URL (http://<IP address>/MyDir/Default.mwsl) and then attempts to resolve the URL: • If this succeeds, Default.mwsl is returned to the client. • If this is not successful, either a directory view is returned or an HTTP 404 "Not Found" error message is issued (depending on configuration).

<HEADER>

Tag	<HEADER>
	The Web server offers with HEADER elements within the HEADERS element the option of mapping the file extensions of a file to an associated Mime type.
Explanation	The content of a file is designated in the file system by its file extension (e.g. "txt" for text files). An assignment is not mandatory in a transport protocol such as HTTP. For this reason, an HTTP header that contains this information about the content type is inserted. **Caution**: If there are more than 60 HEADER entries, the controller is not started and cannot be used.

Example	`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
	`<SERVERPAGES version='05.10.15'>`
	`[...]`
	`<HEADERS>`
	`<!-- 60 entries allowed, otherwise MiniWeb will not start up -->`
	`[...]`
	`<EXTENSION Value="dvi">`
	`<HEADER Name="Content-Type" Value="application/x-dvi" />`
	`</EXTENSION>`
	`[...]`
	`</HEADERS>`
	`[...]`
	`</SERVERPAGES>`
	
	The Mime type `application/x-dvi` is specified for the `dvi` file extensions.
	For more information about MIME types, refer to the RFCs 2045 ff.

<LANGUAGE>

Tag	<LANGUAGE>
	Setting the language.

<MWSL2>

Tag	<MWSL2>
	This tag is used for internal purposes only and should not be changed.

<LISTEN> Primary HTTP port

Tag	<LISTEN ... Id="primary-http >
	Every TCP/IP server (or service) has a so-called well-known port number that can be used by a client to address it. For Web servers, this port is normally port number 80. The attributes of the <LISTEN> tag permit the setting of the port number. If no port number has been set, 5001 is set automatically as port. This prevents an address collision with an existing Web server.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <SERVEROPTIONS> [...] <SERVERS> <LISTEN Address="any" Port="80" Family="HTTP" Id="primary-http" > <HOST IgnoreCase="true" FQDN="*" /> </LISTEN> [...] </SERVEROPTIONS> [...] </SERVERPAGES> In this example, the port number of the Web server HTTP access is set to 80.

<LISTEN> Alternative HTTP port

Tag	`<LISTEN ... Id="alternate-http" IsAlternate="true" >`
	Additional port for requests for the Web server. Every TCP/IP server (or service) has a so-called well-known port number that can be used by a client to address it. For Web servers, this port is normally port number 80. The Web server can also "listen" to a second port number. For example, by adding a firewall you can establish a firewall-controlled security concept. If the value is set to 0, no alternative port is available. This is the default setting.

Example	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<SERVERS>`
	`[...]`
	`<LISTEN Address="any" Port="81" Family="HTTP" Id="alternate-http" IsAlternate="true" >`
	`<HOST IgnoreCase="true" FQDN="*" />`
	`</LISTEN>`
	`[...]`
	`</SERVERS>`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In this example, the alternative port number of the Web server is set to 81.

<LISTEN> Primary HTTPS port

Tag	`<LISTEN ... Family="HTTPS" Id="primary-https" IsSslServer="true" >`
	For the SSL protocol (Secure Socket Layer), an additional well-known port number is needed. This is normally port number 443. If SSL is used in the Web server, the port number for SSL can be set. If nothing is set, the number 5443 is set automatically in order to prevent a collision with any existing Web server.

Example	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<SERVERS>`
	`[...]`
	`<LISTEN Address="any" Port="443" Family="HTTPS" Id="primary-https" IsSslServer="true" >`
	`<HOST IgnoreCase="true" FQDN="*" />`
	`</LISTEN>`
	`[...]`
	`</SERVERS>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In this example, the port for the HTTPS call is set to 443.

<LISTEN> Alternative HTTPS port

Tag	`<LISTEN ... Family="HTTPS" Id="alternate-https" IsSslServer="true" IsAlternate="true" >`
	For the SSL protocol (Secure Socket Layer), an additional well-known port number is needed. The standard port number of the SSL protocol is 443. The Web server can also "listen" to a second port number. For example, by adding a firewall you can establish a firewall-controlled security concept. Another application of this alternative port uses the DAV module. The DAV module detects with the used port whether a DAV request or a Web request is involved.

Example	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<SERVERS>`
	`[...]`
	`<LISTEN Address="any" Port="5443" Family="HTTPS" Id="alternate-https" IsSslServer="true" IsAlternate="true" >`
	`<HOST IgnoreCase="true" FQDN="*" />`
	`</LISTEN>`
	`[...]`
	`</SERVERS>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In this example, the alternative port number for SSL is set to 5443.

<SERVEROPTIONS>

Tag	<SERVEROPTIONS>
	The "Server Options" tag includes all basic parameters of the web server. The settings made within the tag affect the core of the web server.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE> [...] </BASE> <SERVEROPTIONS> [...] </SERVEROPTIONS> [...] </SERVERPAGES>

<TIMEZONE>

Tag	<TIMEZONE>
	Sets the time zone of the Web server. To enable time zones to be synchronized with other partners (in other words, to enable the local time-of-day setting of the Web server to be converted to UTC), the Web server must know which time zone has been set for the control's local clock. The value specified here represents the deviation from UTC +/- minutes. In the as-delivered state, this entry is missing and either the default value "UTC +60" (if the project is missing) or the time zone set in HW Config for the Web server will be valid. If the TIMEZONE node is added, the value from the HW Config will not be considered.
Example	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE> [...] </BASE> <SERVEROPTIONS> <TIMEZONE VALUE="+60" /> [...] </SERVEROPTIONS> [...] </SERVERPAGES> In this example, the time zone is set to "UTC + 60 minutes". This corresponds to MET winter time.

6.1.6.2 SIMOTION IT diagnostics files

DIAGURLS.TXT

Structure of the DIAGURLS.TXT file

DIAGURLS.TXT contains the names of the SIMOTION IT diagnostics pages that are backed up when the diagnostics button is pressed or the pages are requested via **Diagnostics > Diagnostics files > Create general diagfiles**. The file is in directory /HMI/SYSLOG/DIAG and can be expanded with further URLs if necessary.

Here is an example of how this file might look like in the delivery state:

```
alarms.mwsl
alarmsdrvifrm.mwsl
alarmbufifrm.mwsl
devinfo.mwsl
basic/b_extdiag.mwsl
basic/b_diagbufdrv.mwsl
diagnost.mwsl
ipconfig.mwsl
mempool.mwsl
start.mwsl
taskrunt.mwsl
timezone.mwsl
```

Content of the file DIAGURLS.TXT

See also

Diagnostic files (Page 3606)

6.1.6.3 LCID country codes

LCID table

Country-specific codes

Table 6-36 English LCID

Decimal value	Country	UMC abbreviation	Priority
1033	United States	B	1
2057	Great Britain	B	2
3081	Australia	B	10
10249	Belize	B	10
4105	Canada	B	10

Decimal value	Country	UMC abbreviation	Priority
9225	Caribbean	B	10
6153	Ireland	B	10
8201	Jamaica	B	10
5129	New Zealand	B	10
13321	Philippines	B	10
7177	Southern Africa	B	10
11273	Trinidad	B	10

Table 6-37 German LCID

Decimal value	Country	UMC abbreviation	Priority
1031	Germany	A	3
3079	Austria	A	20
5127	Liechtenstein	A	20
4103	Luxembourg	A	20
2055	Switzerland	A	20

Table 6-38 French LCID

Decimal value	Country	UMC abbreviation	Priority
1036	France	C	4
2060	Belgium	C	30
3084	Canada	C	30
5132	Luxembourg	C	30
4108	Switzerland	C	30

Table 6-39 Spanish LCID

Decimal value	Country	UMC abbreviation	Priority
1034	Spain (trad.)	D	5
11274	Argentina	D	40
16394	Bolivia	D	40
13322	Chile	D	40
9226	Colombia	D	40
5130	Costa Rica	D	40
7178	Dominican Rep.	D	40
12298	Ecuador	D	40
17418	El Salvador	D	40
4106	Guatemala	D	40

Decimal value	Country	UMC abbreviation	Priority
18442	Honduras	D	40
2058	Mexico	D	40
19466	Nicaragua	D	40
6154	Panama	D	40
15370	Paraguay	D	40
10250	Peru	D	40
20490	Puerto Rico	D	40
14346	Uruguay	D	40
8202	Venezuela	D	40

Table 6-40 Italian LCID

Decimal value	Country	UMC abbreviation	Priority
1040	Italy	E	6
2064	Switzerland	E	50

FurtherLCID

```
Decimal value of country

=========================

1078 Afrikaans
1052 Albanian
14337 Arabic - United Arab Emirates
15361 Arabic - Bahrain
5121 Arabic - Algeria
3073 Arabic - Egypt
2049 Arabic - Iraq
11265 Arabic - Jordan
13313 Arabic - Kuwait
12289 Arabic - Lebanon
4097 Arabic - Libya
6145 Arabic - Morocco
8193 Arabic - Oman
16385 Arabic - Qatar
1025 Arabic - Saudi Arabia
10241 Arabic - Syria
7169 Arabic - Tunisia
9217 Arabic - Yemen
1067 Armenian
1068 Azeri - Latin
2092 Azeri - Cyrillic
1069 Basque
1059 Belarusian
1026 Bulgarian
1027 Catalan
2052 Chinese - China
3076 Chinese - Hong Kong SAR
```

```
5124 Chinese - Macau SAR
4100 Chinese - Singapore
1028 Chinese - Taiwan
1050 Croatian
1029 Czech
1030 Danish
1043 Dutch - Netherlands
2067 Dutch - Belgium
1061 Estonian
1065 Farsi
1035 Finnish
1080 Faroese
2108 Gaelic - Ireland
1084 Gaelic - Scotland
1032 Greek
1037 Hebrew
1081 Hindi
1038 Hungarian
1039 Icelandic
1057 Indonesian
1041 Japanese
1042 Korean
1062 Latvian
1063 Lithuanian
1071 F.Y.R.O. Macedonia
1086 Malay - Malaysia
2110 Malay – Brunei
1082 Maltese
1102 Marathi
1044 Norwegian - Bokml
2068 Norwegian - Nynorsk
1045 Polish
2070 Portuguese - Portugal
1046 Portuguese - Brazil
1047 Raeto-Romance
1048 Romanian - Romania
2072 Romanian - Republic of Moldova
1049 Russian
2073 Russian - Republic of Moldova
1103 Sanskrit
3098 Serbian - Cyrillic
2074 Serbian - Latin
1074 Setsuana
1060 Slovenian
1051 Slovak
1070 Sorbian
1072 Southern Sotho
1089 Swahili
1053 Swedish - Sweden
2077 Swedish - Finland
1097 Tamil
1092 Tatar
```

```
1054 Thai
1055 Turkish
1073 Tsonga
1058 Ukrainian
1056 Urdu
2115 Uzbek - Cyrillic
1091 Uzbek – Latin
1066 Vietnamese
1076 Xhosa
1085 Yiddish
1077 Zulu
```

6.2 SIMOTION IT Programming and Web Services

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

6.2.1 Fundamental safety instructions

6.2.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

6.2.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

6.2.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

6.2.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

6.2.2 Introduction

6.2.2.1 Overview of SIMOTION IT

Overview of SIMOTION IT manuals

The "SIMOTION IT Ethernet-based HMI and diagnostic functions" are described in four manuals (IT = Information Technology):

- **SIMOTION IT Diagnostics and Configuration**
  This manual describes the direct diagnosis of SIMOTION devices. Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.
  See the SIMOTION IT Diagnostics and Configuration Manual.
- **SIMOTION IT Programming and Web Services**
  This manual describes the creation of user-defined Web pages and access to the diagnostic functions via the two Web services provided by SIMOTION IT.
  A Web service enables users to create their own client applications in any programming language. These applications then communicate with the SIMOTION device using Web technologies. The SOAP (Simple Object Access Protocol) communication protocol is used for transferring commands.
  The manual includes information on programming such clients, as well as a description of the SIMOTION IT Web services (OPC XML-DA, Trace via SOAP TVS) via which data and operating states of the controller can be accessed and the variable trace functions can be used.
- **SIMOTION IT Virtual Machine and Servlets**
  This manual describes the Java-based function packages. The Jamaica Virtual Machine (JamaicaVM) is a runtime environment for Java applications on the SIMOTION device. It is an implementation of the "Java Virtual Machine Specification".
  The Servlets section of the manual describes the use of servlets in a SIMOTION device.
  See the SIMOTION IT Virtual Machine and Servlets manual.
- **SIMOTION IT OPC UA**
  The manual describes access to SIMOTION devices via OPC UA.

See also

PDF in the Internet: SIMOTION IT Diagnostics and Configuration (https://support.industry.siemens.com/cs/ww/en/view/109476533)

PDF in the Internet: SIMOTION IT Virtual Machine and Servlets (https://support.industry.siemens.com/cs/ww/en/view/109476529)

6.2.2.2 New features

Overview of the innovations

Version 5.2

- Cross Site Request Forgery (CSFR) Protection. Prevention of non-authorized HTTP requests.CSFR protection in MWSL pages (Page 3749) Example of a client application (Page 3847)

Version 5.1

- For security reasons, access to the file system is possible only with Javascript.

Version 4.5

- OPC UA server. The OPC UA server is described in the SIMOTION IT OPC UA manual.

Version 4.4

- New way of creating MWSL pages (Page 3746). The HTML files stored in MWSL format on the controller are compiled online. An offline compilation as for .mcs is no longer necessary.
- New MWSL functions (Page 3815)
  createGUID, die, DecodeString, EncodeString, ExistFile, GetLanguage, IsAuthAlgo, isFinite, isNaN, IsSSL, parseFloat, parseInt, ReadFile, ReplaceString, ShareRealm
- New version of the MWSL server language
- New MWSL operators (Page 3809)
- File extensions for Web pages have been changed from .mcs/.mbs to .mwsl/.mwsl.cms.

6.2.3 Software programming

6.2.3.1 User-defined pages

User-defined Home page

You can create your own home page and display it instead of the home page for the standard diagnostic pages of the control. To do this, you need to change the default page of the web server in the WebCfg.xml file.

Procedure

1. Creating your own home page. Save this home page, for example, as `MyIndex.mwsl`.
2. Transmit the home page to the memory card of the SIMOTION device using the **Files** page.
3. Open the WebCfg.xml file in an available editor. The file can be found either on the supplied DVD in the 3_Configuration directory (in the default state) or on the SIMOTION device memory card (possibly in a modified state) in directory \USER\SIMOTION\HMICFG.

4. Replace the file name index.mwsl in the <SERVEROPTIONS> in element <DEFAULTDOCUMENT VALUE="index.mwsl" /> with the name of your home page, including the path name "files" (all user-defined HTML pages are stored in the FILES directory).
   Example: <DEFAULTDOCUMENT VALUE="files/MyIndex.mwsl" />
5. Save the changed WebCfg.xml on the memory card via the **Manage Config > WebCfg transmission** page.

Introduction

Individually designed web pages with access to device data

SIMOTION IT DIAG offers the option of storing machine-specific pages on the SIMOTION control. They are stored in a separate binary format (*.mwsl.cms). The following resources can be used to integrate the process values of the SIMOTION control.

- **Server Side Includes (SSI):** Server-side, static, simplest, unformatted display (Page 3839). In the HTML code of the page, process values can be integrated simply and without a representation option (e.g. number of decimal points) at any point.
- **MiniWeb Server Language (MWSL):** Server-side, static, formatted display using scripts in the SIMOTION control (Page 3796).
  The generation of HTML code and the formatted integration of process values (e.g. in HEX or decimal value representation) can be selectively controlled with this script language.
- **OPC XML-DA web service:** Dynamic, formatted display using JavaScript in the browser (Page 3755).
  The OPCXML.JS library offers elegant use of process values for JavaScript when dynamizing HTML pages.

The creation of user-defined pages requires knowledge of HTML and JavaScript programming.

Key words for advanced programming: XML, HTTP Request, Ajax, and web services.

The following figures shows an example of a simple user-defined page.

Figure 6-124 Example of a simple user-defined page

Loading of MWSL pages into the controller

The source code of an MWSL page must be created as a file with the extension .mwsl. An MWSL file is an HTML file with MWSL extensions to be able to copy the control information.

The source code files is converted to the internal format of the controller in different ways:

- Load the files via the **Files > Files** page**.**
- Copy the files onto the card using a card reader.
- Load the files via FTP.

When loading via a card reader or FTP, the files must be stored in directory /USER/SIMOTION/HMI/FILES.

See also

Structure of a MWSL file (Page 3798)

Embedded, user-defined pages (Page 3750)

Compiling MWSL files

The .mwsl files have the same basic structure as standard HTML pages. The source code file should be coded with UTF-8 if special characters are to be displayed correctly.

1. Create the MWSL pages with a tool of your choice. The pages are assigned the file extension .mwsl
2. Compile the page. The converted files will appear in the target directory with the same name as the associated source files having the file extension .mwsl.cms
3. Check and test the page in the **User's Area**.
4. Errors that occur when compiling the MWSL file are output in the log file. Errors that occur when the Web page is called are written in the source text of the MWSL page as error messages.

The /FILES directory and all subdirectories are examined for the corresponding file types. The compiler replaces the original file with the compiled code. The compiled files have the extension *.cms

Compilation of the MWSL pages

There are three ways of compiling MWSL pages:

1. Click the **Compile** button. All files of the FILES directory are compiled. This option is recommended, for example, after an FTP upload.
2. When the controller boots. All files of the FILES directory are compiled.
3. Compile the first time the page is called via its URL. This is also the case when the users area is called.

Note

Backing up the originals

Because the .mwsl files are deleted during conversion, the originals should always be backed up on the PC and only a copy put on the card.

Successfully compiled pages with the .cms extension

The following example shows compilation via the **Files** page. On the **Files** page, you find the Directory Operations with which the MWSL pages can be loaded into the controller via the **Send** button and compiled.

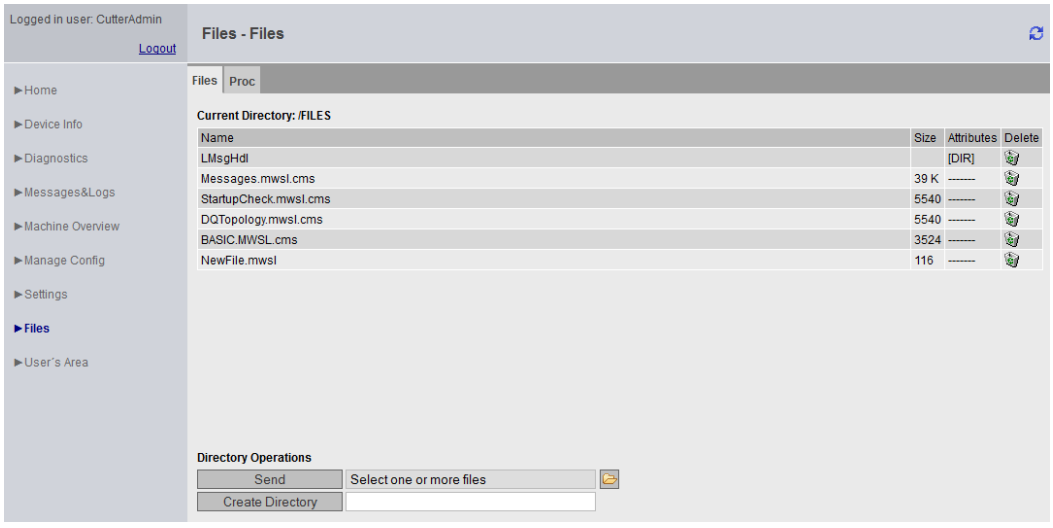


Figure 6-125 Loading of MWSL files

If a file has been loaded without error, it is given the extension .cms

Note

Files with errors do not have an .cms extension. In the above example, there is a problem with the file NewFile.mwsl

Example of compilation of a user-defined page

An MWSL page can be created using any text editor.

```
<html>
<head/>
<body>
<br>  State: <MWSL>WriteVar("DeviceInfo.BZU");</MWSL>
</body>
</html>
```

NewFile.mwsl

In this example, the source text is saved as a file with the name NewFile.mwsl.

The MWSL is used in the example to output device data. Structure of a MWSL file (Page 3798)

The device data is accessed in the MWSL-Ausdrücken by means of the variables provider. See Section Variables Provider (Page 3676) in the SIMOTION IT Diagnostics and Configuration Manual.

The page can only be displayed if the User's Area settings are correct. In this example, the EmbeddedSimple version has been selected. Embedded, user-defined pages (Page 3750)

Figure 6-126 User's Area Edition NewFile.mwsl.cms

The page outputs can be checked in the User's Area. The screenshot shows the output of the sample file. If there are errors in compilation, the file is named NewFile.mwsl.cms

Error analysis

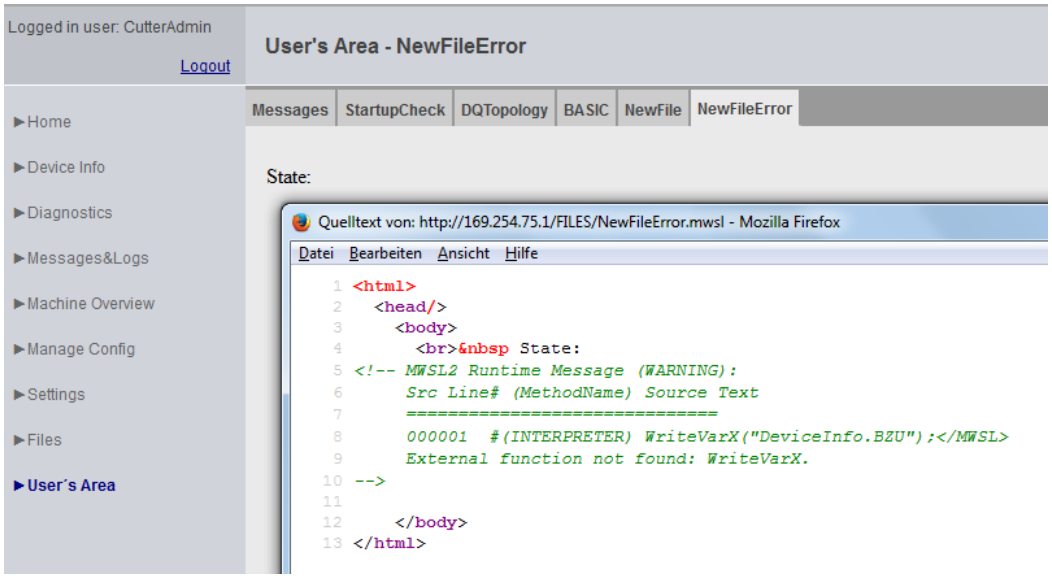


Figure 6-127 User's Area MWSL error message

Error messages that occur when the page is executing are inserted in the source text of the MWSL page as a comment. You can access the source text in the browser by displaying the source text of the current frame. In the example, an attempt is made to access the function `WriteVarX()`, which does not exist.

Errors occurring during compilation are output in a log file formed by appending ".log" to the file name.

Files in the old MBS and MCS format

Interpretation of MCS and MBS files (e.g. user-defined files) is still supported in Version 4.4 to ensure downward compatibility. A diagnostic buffer entry indicates that the format is obsolete. For future versions, this downward compatibility is not ensured. For this reason, old pages should be stored in the new format.

CSFR protection in MWSL pages

To protect MWSL pages against Cross-Site-Request-Forgery (CSFR) attacks, as of V5.2, a CSRF token must be set in the program. If the CSRF token is not set, this causes an error in the browser.

Example program:

```
<html>
<head>
    <title>SIMOTION Demopage CSRF Token Passing</title>
</head>
<body text="#000000" bgcolor="#FFFFFF" link="#FF0000"
alink="#FF0000" vlink="#FF0000">
    <h1>Demopage - CSRF Token Passing</h1>
    <p>
        Demovariable: <b>dev/Service.BZU.Value</b><br>
        current value: <b><%= dev/Service.BZU.Value %></b><br>
        RawMultiUseToken: <b><%=RawMultiUseToken%></b>
    </p>
    <p>
        <form method="post" action="/VarApp">
            write test value: <b>dev/Service.BZU.Value</b>:
           <!-- CSRF Token input _must_ be at 1. place in form !!! -->
            <input type="hidden" name="MultiUseToken" value="<
%=RawMultiUseToken%>" />
          <input type="text" name="dev/Service.BZU.Value" value="<%=
dev/Service.BZU.Value %>" />
            <input type="submit" value="Wert schreiben" />
        </form>
    </p>
</body>
</html>
```

The tag for CSFR protection is shown bold. This tag must be the first `<input>` tag within the form.

`MultiUseToken` designates a CSFR token that permits the same token to be used for the repeated call of a page.

The value of the CSFR token is set with the Server Side Include <%=RawMultiUseToken%> by the server.

Figure 6-128 Representation of the example program in the browser

Embedded, user-defined pages

Integrating user-defined pages

As of Version 4.1.3, user-defined pages can be embedded in the framework of the SIMOTION IT standard pages.

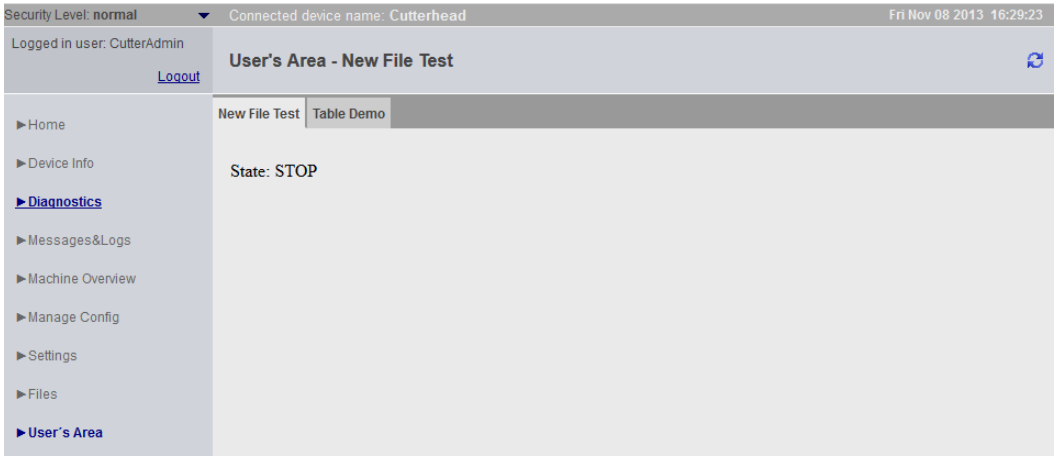


Figure 6-129 User's Area with embedded page

The menu in the **User`s Area** includes the files in the FILES folder in two different ways:

- EmbeddedSimple: The **User`s Area** page loads all Web pages contained in the FILES folder as tabs. The file name is shown without file extension.
- Embedded: The page loads a user-definable tab.

You can switch between EmbeddedSimple and Embedded on the**Settings** page. See the Manual SIMOTION IT Diagnostics and Configuration, Section Standard pages.

Settings in WebCfg.xml

In WebCfg.xml , the appearance of the User's Area can be set with the configuration constants <UserArea> and <UserDir> .

The <UserArea> tag can be used to set how the tab is displayed (the default setting is shown in bold):

```
<UserArea>( StandAlone | Embedded | EmbeddedSimple )</UserArea>
```

<UserDir> denotes the directory for the tab files relative to the FILE directory.

```
<UserDir></UserDir>
```

StandAlone

```
<UserArea>StandAlone</UserArea>
```

Access to the **User's Area** is permanently linked to the user.mwsl file. To display the **User's Area** , this file must exist and be available for calling.

Example user.mwsl

```
<html>
  <head/>
  <body>
    <br>  user.mwsl content ...
  </body>
</html>
```

This file must be stored in the FILES directory of the controller. The file can be copied from the **Files** page or directly to the memory card.

On the Settings page, a change can then be made at **User pages** to **StandAlone**.

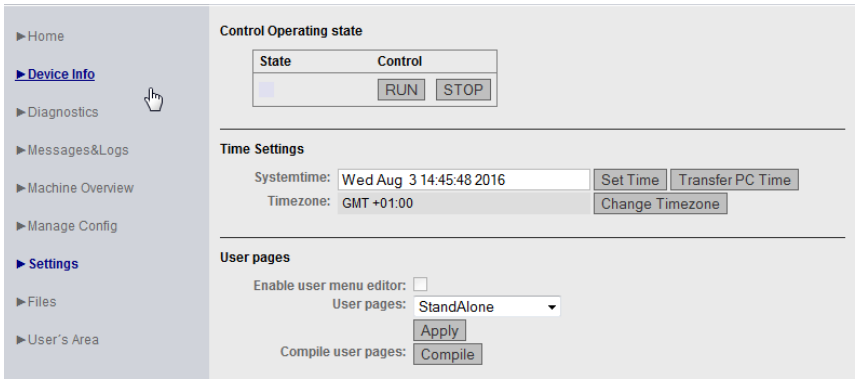


Figure 6-130 User pages: Stand alone

The page is activated by clicking the **Apply** button. Clicking User's Area displays the contents of the page.

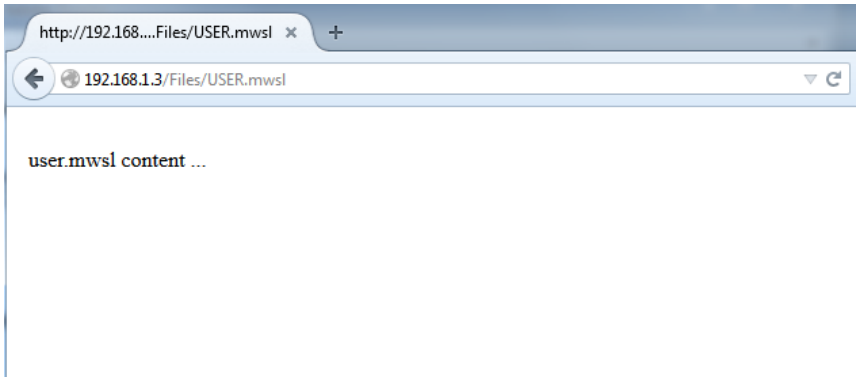


Figure 6-131 User pages: Stand alone Display user.mwsl

EmbeddedSimple

```
<UserArea>EmbeddedSimple</UserArea>
```

Select this option to use all the files found in the directory labeled `<UserDir>` to create the tab.

The respective file name (without the extension) is used as the title, and the corresponding link refers to this file.

Embedded - Using the menu editor

```
<UserArea>Embedded</UserArea>
```

If the **Enable user editor** checkbox has been activated on the **Settings** page, the menu editor (Page 3752) in which menus can be customized is displayed in the **User's Area**.

Example WebCfg.xml:

```
<SERVERPAGES version="78.00">
  [...]
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <UserArea>Embedded</UserArea>
      <UserDir/>
    </USERCONFIG>
  </CONFIGURATION_DATA>
  [...]
</SERVERPAGES>
```

Menu editor

Creating individual menus using the menu editor

As of Version 4.1.3, the menu editor is called with the **User's Area** link. The menu editor can be used to customize the layout of the menus displayed in the User's Area.

Prerequisites for using the menu editor

To use the menu editor, in WebCfg.xml, the configuration constant <UserArea> must be set to Embedded. Alternately, on the **Settings** page, **User pages** can be set to **Embedded**.

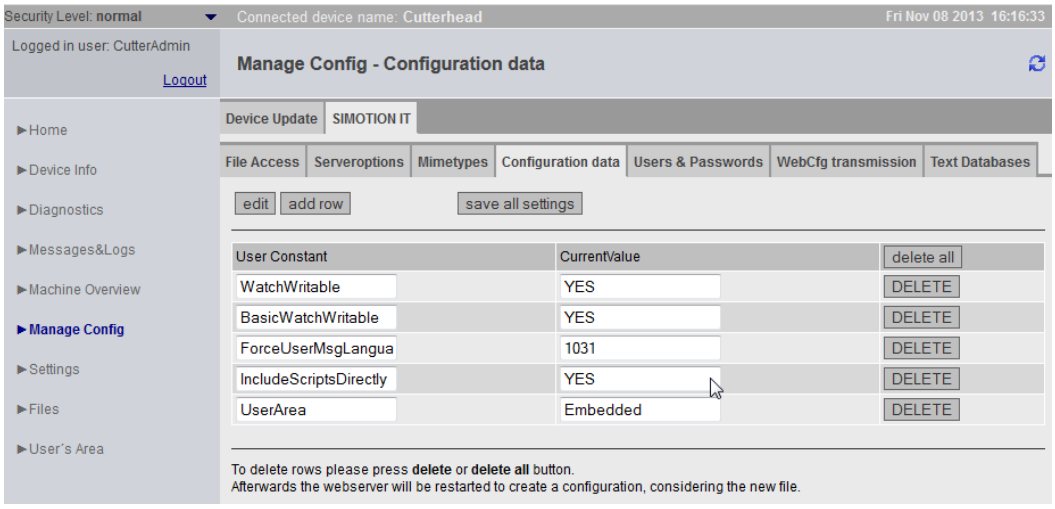


Figure 6-132 Configuration data menu editor

The **Enable user menu editor** option must then be selected on the **Settings** page.

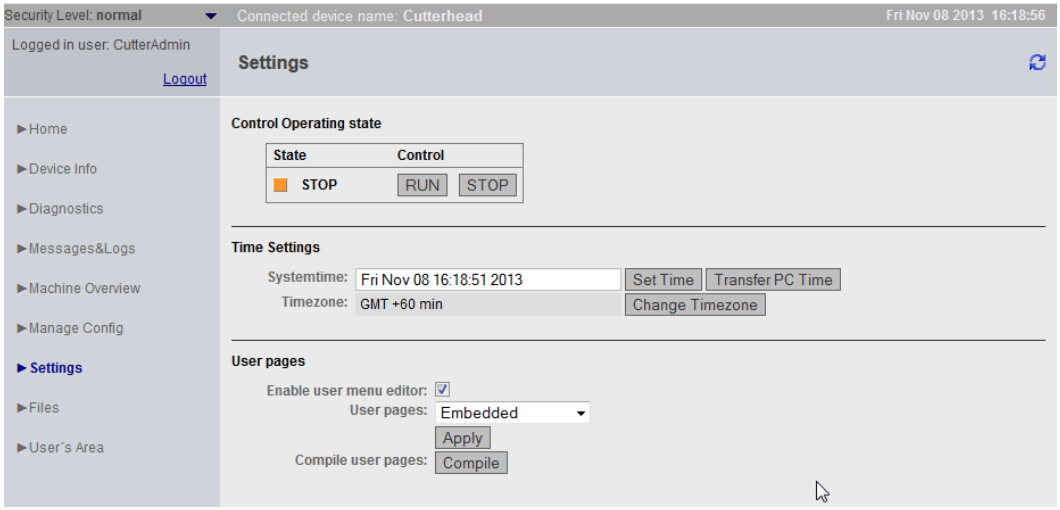


Figure 6-133 Settings menu editor

Working with the menu editor

The **User's Area** page now contains the **Menu editor** tab.

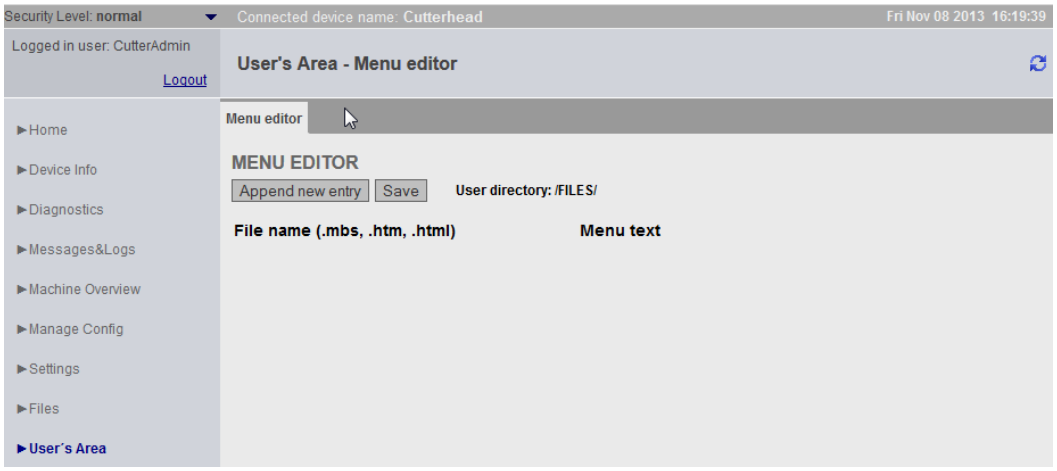


Figure 6-134 Starting the menu editor for the first time

The first time the menu editor is launched, a largely blank page appears.

New menu items can be created with the **Append new entry** button.

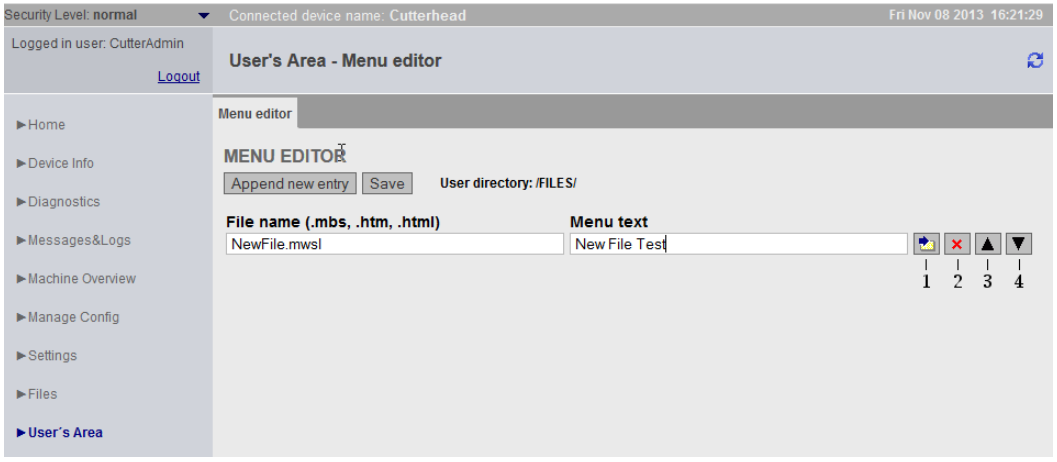


Figure 6-135 Menu editor with several items

In the screenshot above, the NewFile.mwsl file has been added. The buttons can be used to add or delete files, or change their position.

The names of the files to be displayed for the corresponding menu commands are entered in the **File name** column.

The **Menu text** column contains the name of the menu item.

The ① button creates new menu items that are inserted before each current item.

The ② button deletes the associated menu item.

The ③ button moves the menu item up.

The ④ button moves the menu item down.

Example

Two files are entered in the menu editor.

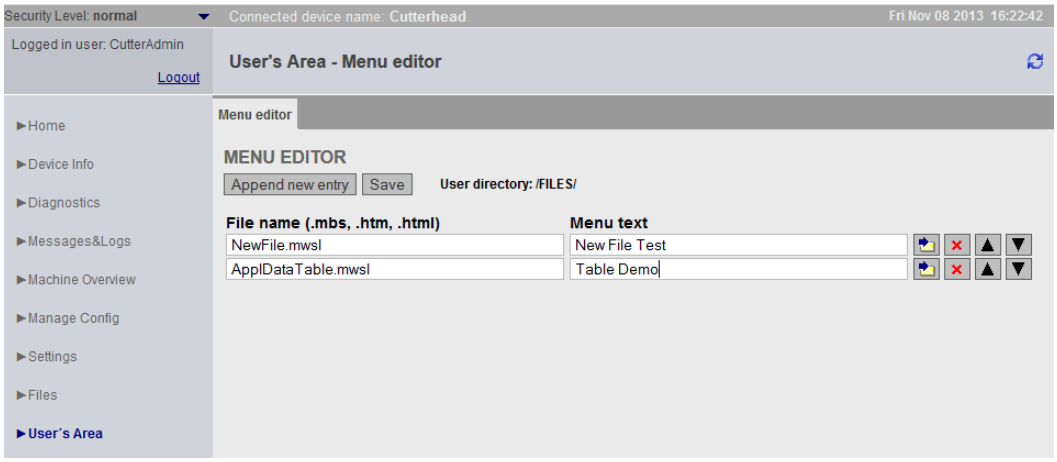


Figure 6-136 Example menu editor

The **User's Area** now only shows the selected pages.

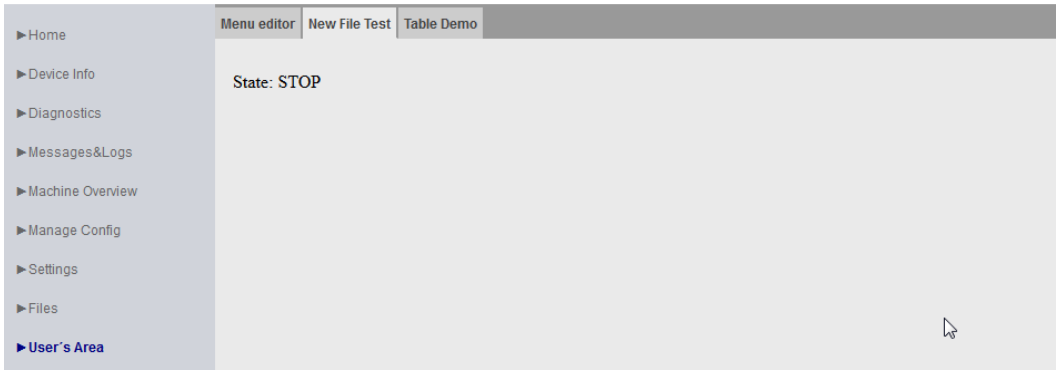


Figure 6-137 Example Menu Editor - display of Users's Area

To hide the tab **Menu editor** on the page, on the **Settings** page, the option **Enable user menu editor** must be deactivated.

JavaScript and web services

Variable access with JavaScript and web services

Access to a device with the JavaScript library

Using the DOM functionality of JavaScript , it is possible to implement simple Web service clients. This opens up a multitude of new options within a browser, e.g.:

- Reading and cyclic updating of variable contents using an OPC XML-DA Read command
- Writing of variables using an OPC XML-DA Write command

- Browsing of the entire SIMOTION variable management area
- Setting up and querying an OPC XML-DA Subscription

The functionality is provided by several JavaScript files:

- opcxml.js: Contains functions for the structure of the necessary XML documents and for communication with an OPC XML-DA server.
- appl.js: Based on opcxml.js and implements the following objects:
  - Variable browser. Representation of the SIMOTION variable management area in a tree topology within the browser
  - Property viewer: Representation of variable properties (value, data type, access rights, Enums) in the form of a table within a browser. For writable variables, the table contains an entry field for changing the variable content.
  - Watch table: Representation of a watch table in the browser

CSFR protection

As of V5.2, the firmware contains new versions of the appl.js and opcxml.js that are prepared for the CSFR protection. Older versions of the files may cause errors for execution with V5.2. Replacing the files rectifies the problem.

Communication with the OPC XML DA server (opcxml.js)

OPCReadRequest

The `OPCReadRequest` class can be used to read the values for a list of variables.

```
function OPCReadRequest(parLocaleId,parResultCB)
```

Transfer parameters:

- `parLocaleId`: Language identifier ("DE", "EN")
- `parResultCB`: Callback function that must be provided by the caller
  This function is called by OPCReadRequest when a response has arrived from the OPC XML DA server. The OPCReadRequest object is disposed of automatically (by calling the "destructor" method) if the callback function returns the value "true".

  ```
  function OPCReadRequestCB(parResponse)
  ```

  Transfer parameters:

  – `parResponse`: Array of ItemValues with the result of the read request.

  ```
  function OPCItemValue()
  {
     this.mItemPath;
     this.mItemName;
     this.mItemHandle;
     this.mItemValue;
     this.mItemResultId;
  }
  ```

User interface:

- `addItem(parItemPath,parItemName)` adds a variable to the variables list
- `removeItem(parItemHandle)` deletes a variable from the variables list
- `sendReadRequest()` sends the read request
- `destructor()` releases the entire request object

If `mItemResultId` is defined, an error occurred during reading. In this case, `mItemResultId` is assigned the OPC XML DA error ID.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
         function read()
         {
             var tmpReadCB = function(parResponse)
             {
                 tmpResultStr = "";
                 for (var tmpIndex = 0; tmpIndex < parResponse.length;
                      tmpIndex++)
                 {
                    var tmpItemValue = parResponse[tmpIndex];
                    var tmpValue = (tmpItemValue.mItemValue) ?
                                    tmpItemValue.mItemValue :
                                    tmpItemValue.mItemResultId;
                    tmpResultStr += tmpItemValue.mItemPath
                                    + "::"
                                    + tmpItemValue.mItemName
                                    + " = "
                                    + tmpValue
                                    + "\n";
                 }
                 alert(tmpResultStr);
                 return true;
              }
              var tmpReadRequest = new OPCReadRequest("DE",tmpReadCB);
              tmpReadRequest.addItem("SIMOTION","var/userdata.user1");
              tmpReadRequest.addItem("SIMOTION","var/userdata.user2");
              tmpReadRequest.addItem("SIMOTION","var/userdata.user10");
              tmpReadRequest.sendReadRequest();
         }
      </script>
      <title>Insert title here</title>
   </head>
   <body>
      <input type="button" onclick="read();" value="Read"/>
   </body>
</html>
```

OPCGetPropertiesRequest

The `OPCGetPropertiesRequest` class can be used to read the properties of variables:

- Values
- Data types

- Access rights
- Enum components

```
function OPCGetPropertiesRequest(parLocaleId,parResultCB)
```

Transfer parameters:

- `parLocaleId`: Language identifier ("DE", "EN")
- `parResultCB`: Callback function that must be provided by the caller.
  This function is called by OPCGetPropertiesRequest when a response has arrived from the OPC XML DA server. The OPCGetPropertiesRequest object is disposed of automatically (by calling the "destructor" method) if the callback function supplies "true" as a return value.

  `function OPCGetPropertiesRequestCB(parResponse)`
  parResponse: Array of PropertyResults that contain the properties of variables:

```
function OPCPropertyResult()
{
   this.mItemPath;
   this.mItemName;
   this.mName;
   this.mItemHandle;
   this.mIsItem;
   this.mHasChildren;
   this.mResultId;
   this.mValue;
   this.mType;
   this.mAccessRights;
   this.mEffectiveness;
   this.mEnums;
   this.mIsEnum;
}
```

User interface:

- `addItem(parItemPath,parItemName)` adds a variable to the variables list
- `removeItem(parItemHandle)` deletes a variable from the variables list
- `sendGetPropertiesRequest()` sends the read request
- `destructor()` releases the entire request object

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <title>GetProperties</title>
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
      function getProperties()
      {
         var tmpGetPropertiesCB = function(parResponse)
         {
             tmpResultStr = "";
             for (var tmpIndex = 0; tmpIndex < parResponse.length;
                   tmpIndex++)
             {
                 var tmpPropertyResult = parResponse[tmpIndex];
                 var tmpEnums = "";
                 if (tmpPropertyResult.mEnums &&
                     (tmpPropertyResult.mEnums.length > 0))
                {
                    for (var tmpIndex = 0;
                         tmpIndex < tmpPropertyResult.mEnums.length;
                         tmpIndex++)
                    {
                        tmpEnums += " " +
                           tmpPropertyResult.mEnums[tmpIndex] + "\n";
                    }
                 }
                 if (!tmpPropertyResult.mResultId)
                 {
                   tmpResultStr += tmpPropertyResult.mItemPath +
                                   "::" +
                                   tmpPropertyResult.mItemName +
                                   ":\n Type = " +
                                   tmpPropertyResult.mType +
                                   "\n value = " +
                                   tmpPropertyResult.mValue +
                                   "\n AccessRights = " +
                                   tmpPropertyResult.mAccessRights +
                                   "\n Enums = \n" + tmpEnums +"\n";
                 }
                 else
                 {
                     tmpResultStr += tmpPropertyResult.mItemPath +
                                     "::" + tmpPropertyResult.mItemName
                                     + ":\n ResultId = " +
                                     tmpPropertyResult.mResultId +
                                     "\n\n";
                 }
              }
              alert(tmpResultStr);
```

```
            return true;
          }
          var tmpGetPropertiesRequest =
           new OPCGetPropertiesRequest("DE",tmpGetPropertiesCB);
          tmpGetPropertiesRequest.addItem("SIMOTION",
                                          "var/userdata.user1");
          tmpGetPropertiesRequest.addItem("SIMOTION",
                                          "var/userdata.user20");
          tmpGetPropertiesRequest.addItem("SIMOTION",
                                          "dev/Service.BZU.value");
          tmpGetPropertiesRequest.sendGetPropertiesRequest();
        }
      </script>
   </head>
   <body>
      <input type="button" onclick="getProperties();"
             value="getProperties"/>
   </body>
</html>
```

OPCWriteRequest

`OPCWriteRequest` writes three values of one or more variables.

```
function OPCWriteRequest(parLocaleId,parResultCB)
```

Transfer parameters:

- `parLocaleId`: Language identifier ("DE", "EN")
- `parResultCB`: The callback function that must be provided by the caller.
  The function is called on conclusion of the send request.

  `function OPCWriteRequestCB(parResultList)`
  parResultList is an array of ItemValues that contains the results of the write request.

  ```
  function OPCItemValue()
  {
     mItemPath
     mItemName
     mItemHandle
     mItemValue
     mItemResultId
  }
  ```

  The OPCWriteRequest object is discarded automatically (by calling the "destructor" method) if the callback function returns the value "true".

User interface:

- `addItem(parItemPath,parItemName,parType)` adds a variable to the variables list and returns a variable handle, which can be used to reference the variable within the request. ParType designates the OPC XML DA data type to be used for writing. If parType is not passed, the "xsi::string" data type is applied.
- `removeItem(parItemHandle)` removes a variable from the variables list
- `setItemValue(parItemHandle,parValue)` sets the value that is to be written for a variable
- `sendWriteRequest()` sends the write request
- `destructor()` releases all resources occupied by the write object

Note

Variable count for the write request

A maximum of 100 variables per write request is recommended.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
      function writeValues()
      {
          var tmpWriteCB = function(parWriteResult)
          {
              var tmpString = "";
              for (var tmpIndex = 0; tmpIndex < parWriteResult.length;
                   tmpIndex++)
              {
                var tmpItemValue = parWriteResult[tmpIndex];
                var tmpValue = (tmpItemValue.mItemResultId) ?
                 tmpItemValue.mItemResultId : tmpItemValue.mItemValue;
                tmpString += tmpItemValue.mItemPath + "::" +
                  tmpItemValue.mItemName + " = " + tmpValue + "\n";
              }
              alert(tmpString);
          }
          var tmpWrite = new OPCWriteRequest("DE",tmpWriteCB);
          var tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                               "var/userdata.user1");
          tmpWrite.setItemValue(tmpItemHandle,"123");
          tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                           "var/userdata.user2");
          tmpWrite.setItemValue(tmpItemHandle,"234");
          tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                           "var/userdata.user10");
          tmpWrite.setItemValue(tmpItemHandle,"345");
          tmpWrite.sendWriteRequest();
      }
      </script>
      <title>Write</title>
   </head>
<body>
<input type="button" value="Write" onclick="writeValues()"/>
</body>
</html>
```

OPCBrowseRequest

The `OPCBrowseRequest` class can be used to browse the variable management area of a control.

```
function OPCBrowseRequest(parclaaLocaleId,parResultCB)
```

Transfer parameters:

- `parLocaleId`: Language identifier ("DE", "EN")
- `parResultCB`: Callback function that must be provided by the caller
  The function is called on conclusion of the send request.

  ```
  function OPCBrowseRequestCB(parResult,parItemPath,parItemName)
  ```

  `parResult` is an array of type `BrowseResult` and contains the Browse information.

  ```
  function BrowseResult()
  {
     mItemPath;
     mItemName;
     mName;
     mIsItem;
     mHasChildren;
  }
  ```

  `parItemPath` and `parItemName` are the path and name of the directory whose content is displayed in `BrowseResult`.

User interface:

- `sendBrowseRequest(parItemPath,parItemName)` sends the Browse request.
  `parItemPath` and `parItemName` are the path and name of the directory to be browsed.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
       <meta http-equiv="Content-Type" content="text/html;
             charset=ISO-8859-1">
       <script type="text/javascript" src="/common.js"></script>
       <script type="text/javascript" src="/opcxml.js"></script>
       <script type="text/javascript">
       function browse()
       {
           var tmpBrowseRequestCB = function(parBrowseResult,
                                             parItemPath,
                                             parItemName)
           {
             var tmpString = parItemPath + "::" + parItemName + "\n";
             for (var tmpIndex = 0; tmpIndex < parBrowseResult.length;
                  tmpIndex++)
             {
                 var tmpBrowseResult = parBrowseResult[tmpIndex];
                 tmpString += tmpBrowseResult.mItemName + "\n";
             }
             alert(tmpString);
           }
           var tmpBrowseRequest =
             new OPCBrowseRequest("DE",tmpBrowseRequestCB);
           tmpBrowseRequest.sendBrowseRequest("SIMOTION","var/");
        }
        </script>
        <title>Browse</title>
   </head>
   <body>
       <input type="button" value="Browse" onclick="browse();"/>
   </body>
</html>
```

OPCSubscriptionRequest

The `OPCSubscriptionRequest` class can be used to set up, poll, and delete an OPC XML DA subscription.

```
function OPCSubscriptionRequest(parLocaleId,parResultCB,parCancelCB)
```

Transfer parameters:

- `parLocaleId` Language identifier ("DE", "EN").call
- `parResultCB` Callback function that must be provided by the user. This callback function is called following setup and a refresh action.

  `function OPCSubscriptionRequestCB(parResultList,parResult)`
  `parResultList` is an array of type `OPCItemValue`, which contains the variable values which have been determined.

```
function OPCItemValue()
{
   mItemPath
   mItemName
   mItemHandle
   mItemValue
   mItemResultId
}
```

- `parCancelCB` Bback function that must be provided by the user. This callback function is called after a subscription is released.

  `function OPCSubscriptionCancelCB()`
  The function has no transfer parameters to be passed.

User interface:

- `addItem(parItemPath,parItemName)` adds the passed variable to the internal list of subscription variables
- `removeItem(parItemHandle)` deletes the passed variable from the list of subscription variables.
- `cancel()` logs out an active subscription on the server
- `refresh()` reads the current variable values.
  Only variables whose values have changed since the preceding query are passed. The first call of refresh after generating the OPCSubcription object or the first call after a cancel causes the subscription to log in with the current internal variable list on the server. Refresh must be called cyclically. The hold time mechanism of the OPC XML DA subscription is not supported, i.e. a wait time must be programmed between each 2 refresh cycles by means of a JavaScript timer. However, the wait time of the OPC XML DA subscription is active, i.e. a response to a refresh call is sent only after the wait time elapses, provided a variable value has not changed in the meantime.
- `destructor()`
  Logs out the subscription on the server and releases the resources occupied by the subscription object. Destructor must be called before releasing the object.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
          var gloSubscription;
          var gloItemHandle_1;
          var gloItemHandle_2;
          function subscription()
          {
              if (!gloSubscription)
              {
                  var tmpSubscriptionCB = function(parValues)
                  {
                      for (var tmpIndex = 0;
                           tmpIndex < parValues.length;
                           tmpIndex++)
                      {
                          var tmpItemHandle =
                              parValues[tmpIndex].mItemHandle;
                          var tmpItemValue =
                              parValues[tmpIndex].mItemValue;
                          if (tmpItemHandle == gloItemHandle_1)
                          {
                              var tmpValueNode =
                                 document.getElementById("user1");
                              tmpValueNode.firstChild.nodeValue =
                                 tmpItemValue;
                          }
                          else if (tmpItemHandle == gloItemHandle_2)
                          {
                              var tmpValueNode =
                                 document.getElementById("user2");
                              tmpValueNode.firstChild.nodeValue =
                                  tmpItemValue;
                          }
                      }
                    var tmpTimerCB = function()
                    {
                        gloSubscription.refresh();
                    }
                    setTimeout(tmpTimerCB,300);
                }
                var tmpCancelCB = function()
                {
                    if (gloSubscription)
                    {
                        gloSubscription.destructor();
                        gloSubscription = null;
                    }
                }
```

```
                gloSubscription =
                   new OPCSubscriptionRequest("DE",
                        tmpSubscriptionCB,tmpCancelCB);
                gloItemHandle_1 =
                   gloSubscription.addItem("SIMOTION",
                                           "var/userdata.user1");
                gloItemHandle_2 =
                   gloSubscription.addItem("SIMOTION",
                                           "var/userdata.user2");
                gloSubscription.refresh();
             }
          }
          function cancel()
          {
             if (gloSubscription)
                gloSubscription.cancel();
          }
       </script>
          <title>Subscription</title>
       </head>
       <body>
          <div>
             <input type="button" value="Start"
                    onclick="subscription();"/>
             <input type="button" value="Cancel" onclick="cancel();"/>
          </div>
          <table>
             <tr>
                 <td>user1</td>
                 <td id="user1">user1</td>
             </tr>
             <tr>
                 <td>user2</td>
                 <td id="user2">user2</td>
             </tr>
          </table>
       </body>
</html>
```

OPCSubscriptionAutoRefresh

`OPCSubscriptionAutoRefresh` provides the same function as `OPCSubscriptionRequest`, with the exception that the timer-controlled call of the refresh function occurs automatically.

```
function
OPCSubscriptionAutoRefresh(parLocaleId,parResultCB,parCancelCB,
 parCycleTime)
```

Transfer parameters:

- `parLocaleId`
- `parResultCB`

- `parCancelCB`
  See OPCSubscriptionRequest
- `parCycleTime`
  Cycle time in which the refresh function is called, in ms

User interface:

- `startRefresh()` starts the query cycle
- `cancel()` See OPCSubscriptionRequest
- `addItem(parItemPath,parItemName,parItemHandle)` See OPCSubscriptionRequest
  After the variable is added, the refresh cycle is restarted automatically.
- `removeItem(parItemHandle)`
- `destructor()`

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
         var gloSubscription;
         var gloItemHandle_1;
         var gloItemHandle_2;
         var gloItemHandle_3;
         function subscription()
         {
             if (!gloSubscription)
             {
                 var tmpSubscriptionCB = function(parValues)
                 {
                    for (var tmpIndex = 0; tmpIndex < parValues.length;
                         tmpIndex++)
                    {
                        var tmpItemHandle =
                            parValues[tmpIndex].mItemHandle;
                        var tmpItemValue =
                            parValues[tmpIndex].mItemValue;
                        if (tmpItemHandle == gloItemHandle_1)
                        {
                            var tmpValueNode =
                               document.getElementById("user1");
                            tmpValueNode.firstChild.nodeValue =
                               tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_2)
                        {
                            var tmpValueNode =
                                document.getElementById("user2");
                            tmpValueNode.firstChild.nodeValue =
                                tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_3)
                        {
                            var tmpValueNode =
                                document.getElementById("user3");
                            tmpValueNode.firstChild.nodeValue =
                                tmpItemValue;
                        }
                     }
                 }
                 var tmpCancelCB = function()
                 {
                     if (gloSubscription)
                     {
                         gloSubscription.destructor();
                         gloSubscription = null;
```

```
                    }
                }
                gloSubscription =
                  new OPCSubscriptionAutoRefresh("DE",
                              tmpSubscriptionCB,tmpCancelCB,300);
                gloItemHandle_1 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user1");
                gloItemHandle_2 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user2");
                gloSubscription.startRefresh();
             }
          }
          function addVar()
          {
             if (gloSubscription)
             {
                 gloItemHandle_3 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user3");
             }
          }
          function removeVar()
          {
             if (gloSubscription)
             {
                 gloSubscription.removeItem(gloItemHandle_3);
             }
          }
          function cancel()
          {
              if (gloSubscription)
                  gloSubscription.cancel();
          }
   </script>
   <title>Auto refresh</title>
</head>
<body>
   <div>
       <input type="button" value="Start"
              onclick="subscription();"/>
       <input type="button" value="Add variable"
              onclick="addVar();"/>
       <input type="button" value="Remove variable"
              onclick="removeVar();"/>
       <input type="button" value="Cancel" onclick="cancel();"/>
   </div>
   <table>
      <tr>
         <td>user1</td>
         <td id="user1">user1</td>
     </tr>
     <tr>
         <td>user2</td>
         <td id="user2">user2</td>
    </tr>
```

```
        <tr>
           <td>user3</td>
           <td id="user3">user3</td>
        </tr>
       </table>
    </body>
</html>
```

Representation of OPC XML-DA data in the browser (appl.js)

appl.js

The JavaScript library **appl.js** assembles classes for the representation of the data determined with OPC XML-DA requests.

ApplDataTable

`ApplDataTable` implements a dynamic table in which process variables can be displayed. The variable values are updated cyclically with an OPC XML-DA subscription.

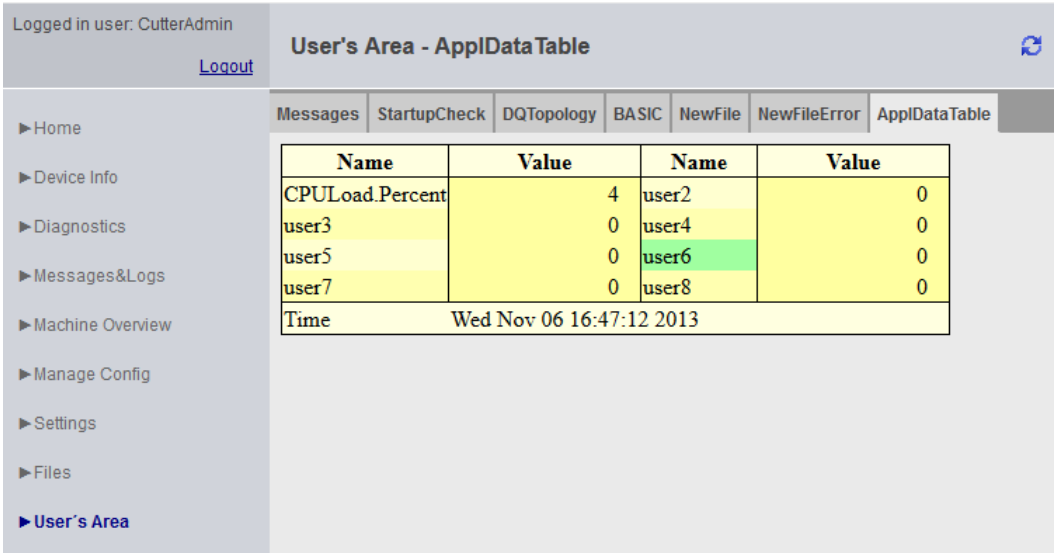


Figure 6-138 Example of implementing a dynamic table (ApplDataTable)

```
function
ApplDataTable(parDocument,parClassName,parColumnClasses,parColumnIds
,parHeader)
```

Transfer parameters:

- `parDocument`: JavaScript document used to generate elements
- `parClassName`: Entered as "class" attribute in the "Table" tag of the HTML table

- parColumnClasses: Array whose "length" attribute specifies the number of table columns. The values of the array are used as a "class" attribute for the table columns (<td class="...">).
- parColumnIds
  : Array whose values are used together with the parRowId parameter (described below) for the "id" attributes of the table columns. The value of the "id" attribute is produced by chaining together the ColumnId and RowId (in that order).
- parHeader: Array containing the column headings of the table

The transfer parameters passed are used to set the "class" and "id" attributes such that the table display can be specified using style sheets.

User interface:

The user interface of class ApplDataTable consists of a series of functions that are useful for general use in the web site and their passed and returned parameters.

- addRow(parRowId,parRowClass) appends a new row to the table
  - parRowId: Used as a value for the id attribute of row (<tr id="...">)
  - parRowClass: Used as a value for the "class" attribute of row (<tr class="...">)
- addElement(parElement,parDestructor,parColSpan,parColClass)
  Inserts the HTML element passed using parElement in the table.
  - parElement: HTML element to be inserted in the table
  - parDestructor (optional)
    Function that is called if the HTML element is deleted from the table (used in Internet Explorer to prevent memory leaks).
  - parColSpan (optional)
    : Specifies the number of columns the element is to span.
  - parColClass (optional)
    parColClass can be used to overwrite the ColClass assigned when the ApplDataTable object is created, if special formatting is to be applied for the current element.
- addVariable(parPath,parName,parColSpan,parColClass)
  : Inserts a variable in the table. The value of the variables is updated cyclically.
  - parPath
    Variable path (e.g. "SIMOTION" or "SIMOTION diagnostics")
  - parName
    Variable name (e.g. "var/userdata.user1")
  - parColSpan (optional)
    See addElement
  - parColClass (optional)
    See addElement

- `addText(parText,parColSpan,parColClass)` : Inserts text in the table.
  - `parText`
    : Text to be inserted.
  - `parColSpan` (optional)
    See `addElement`
  - `parColClass` (optional)
    See `addElement`
- `addRemoveButton (parRemoveCB,parRemoveData,parImage)`
  Inserts a button in the table for removing the current row.
  - `parRemoveCB(parData)` (optional)
    Transfer parameter passed to the Callback function. Data passed when addRemoveButton is called
  - `parRemoveData` (optional)
    Data passed as a parameter when parRemoveCB is called.
  - `parImage`: (optional)
    Optional parameters: DOM object of an HTML img element (e.g. created with document.createElement("img")), which is to appear on the button
- `addImage(parImage)`
  Inserts an image in the table.
  - `parImage`: URL of the image to be inserted
- `getVariables()` supplies an array of all variables of the table
  Each entry of the array consists of an object with the elements `mItemPath`, `mItemName`, and `mItemHandle`.
- `addRefreshCB(parRefreshCB)` registers a Callback function on the table object, which is called on completion of a refresh cycle; an array of `OPCItemValue`-Objekten is passed to the Callback function.

```
OPCItemValue
{
   mItemPath
   mItemName
   mItemHandle
   mItemValue
   mItemResultId
}destructor() must be called if the table is no longer required.
```

This function releases all the resources occupied by the table. In particular, the subscription used by the table is logged off on the OPC XML DA server. The function must also be called when the page is exited (onunload).

Example:

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript" src="/appl.js"></script>
      <script type="text/javascript">
      var gloApplDataTable = null;
      function init()
      {
          if (gloApplDataTable == null)
          {
              var tmpColumnIds     = new Array("Name_0","Value_0",
                                               "Name_1","Value_1");
              var tmpColumnClasses = new Array("Name","Value",
                                               "Name","Value")
              var tmpHeader        = new Array("Name","Value",
                                               "Name","Value");
              gloApplDataTable = new ApplDataTable(document,
                                                   "ReadTableClass",
                                                   tmpColumnClasses,
                                                   tmpColumnIds,
                                                   tmpHeader);
              gloApplDataTable.addRow("Row_0","RowClass_0");
              gloApplDataTable.addText("CPULoad.Percent");
              gloApplDataTable.addVariable("SIMOTION diagnostics",
                                           "CPULoad.Percent");
              gloApplDataTable.addText("user2");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user2");
              gloApplDataTable.addRow("Row_1","RowClass_1");
              gloApplDataTable.addText("user3");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user3");
              gloApplDataTable.addText("user4");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user4");
              gloApplDataTable.addRow("Row_2","RowClass_0");
              gloApplDataTable.addText("user5");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user5");
              gloApplDataTable.addText("user6");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user6");
              gloApplDataTable.addRow("Row_3","RowClass_1");
              gloApplDataTable.addText("user7");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user7");
              gloApplDataTable.addText("user8");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user8");
              gloApplDataTable.addRow("Time","Time");
              gloApplDataTable.addText("Time",1,"Time");
              gloApplDataTable.addVariable("SIMOTION diagnostics",
```

```
                                                "DeviceInfo.Systemtime",
                                                 3,"Time");

                var tmpTableRoot =
                              document.getElementById("TableRoot");
                if (tmpTableRoot)
                {
                   tmpTableRoot.appendChild(gloApplDataTable.mTable);
                }
             }
          }
          function close()
          {
             if (gloApplDataTable != null)
             {
                 gloApplDataTable.destructor();
             }
         }
         </script>
         <style type="text/css">
           table.ReadTableClass {background-color:#FFFFE0;
                                 border:1px solid black;
                                 border-collapse:collapse;}
           th.Name {border:1px solid black;}
           th.Value {border:1px solid black;}
           td.Name {width:80px;border-right:1px solid black;}
           td.Value {background-color:#FFFFA0;width:120px;
                     text-align:right;
                     border-right:1px solid black;padding-right:15px;}
           td.Input {width:200px;}
           td.Time {border-top:1px solid black;}
           tr.RowClass_0 {background-color:#FFFFD0;border:0px;}
           tr.RowClass_1 {background-color:#FFFFB0;border:0px;}
           #Name_1Row_2 {background-color:#A0FFA0;}
         </style>
         <title>Table</title>
      </head>
      <body onload="init();" onunload="close();">
         <div id="TableRoot"></div>
      </body>
   </html>
```

In the final section of the source code, the <style> tag demonstrates how the colors of table rows and individual table cells can be changed using CSS formatting.

This means that the tr.RowClass_0 declaration, for example, can ensure that a yellow background is produced when gloApplDataTable.addRow("Row_0","RowClass_0"). is called.

ApplBrowser

`ApplBrowser` enables the variable management area of the control to be queried.

After you have created an object of this type, the 1st browse operation is started automatically. If the browse information of a subtree has been received, the callback function `parNewTreeFct` is first called. Afterwards, the callback function `parNewNodeFct` or `parNewLeafFct` is called, depending on the type of information (node or leaf). All callback functions receive an object of type `ApplBrowseElement` as the first parameter.

User interface ApplBrowseElement:

- `getElement()` supplies an HTML anchor object (<a ...>) for displaying information
- `setCB(parCB)`
  : Registers a Callback function at the ApplBrowseElement-Objekt; this function is called if a user clicks the anchor object This function also receives an `ApplBrowseElement-Objekt` as a passed transfer parameter.
- `destructor()` must be called if the object is no longer required. This also applies to exiting the page (`onunload`).

```
ApplBrowser(parDocument,
            parItemPath,
            parItemName,
            parNewTreeFct,
            parNewNodeFct,
            parNewLeafFct)
```

Transfer parameters:

- `parDocument`: JavaScript document used to generate elements
- `parItemPath`, `parItemName` specifies the starting point for browsing the variable management area
- `parNewTreeFct(parBackElement,parItemPath,parItemName)` : Callback function that is called when the structure of a new subtree begins.
  - `parBackElement` object of type `ApplBrowseElement`
    The content describes the start node. If a user clicks the HTML anchor of this object, the elements of the preceding subtree (if present) are determined.
  - `parItemPath,parItemName`: Path and name of the current subtree
- `parNewNodeFct(parBrowseElement)` : Callback function that is called for a node element
  - `parBrowseElement` object of type `ApplBrowseElement`
    The content describes a node. If a user clicks the HTML anchor of this object, a browse operation is started for the subtree connected to this node.
- `parNewLeafFct(parBrowseElement)` Callback function that is called for a leaf element
  - `parBrowseElement` object of type `ApplBrowseElement`
    The content describes a leaf.

User interface:

- `destructor` releases all resources occupied by the browser

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<html>
   <head>
    <title>Browser demo</title>
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript" src="/appl.js"></script>
    <script type="text/javascript">
      var gloBrowseTable = null;
      var gloBrowser = null;
      function addItem(parBrowseElement)
      {
        var tmpBodyElement = gloBrowseTable.firstChild;
        var tmpTableRowElement = document.createElement("tr");
        var tmpTableDataElement = document.createElement("td");
        var tmpLinkElement = parBrowseElement.getElement();
        tmpTableDataElement.appendChild(parBrowseElement.getElement());
        tmpTableRowElement.appendChild(tmpTableDataElement);
                       tmpBodyElement.appendChild(tmpTableRowElement);
      }
      function browse()
      {
        var tmpNewTreeFct =
         function(parBrowseElement,parItemPath,parItemName)
         {
            var tmpBrowseTableHook =
                      document.getElementById("BrowseTable");
            if (tmpBrowseTableHook.firstChild)
               tmpBrowseTableHook.removeChild(
                                  tmpBrowseTableHook.firstChild);
            gloBrowseTable = document.createElement("table");
            var tmpBodyElement = document.createElement("tbody");
            gloBrowseTable.appendChild(tmpBodyElement);
            tmpBrowseTableHook.appendChild(gloBrowseTable);
            if (parBrowseElement)
            {
               parBrowseElement.getElement().firstChild.nodeValue =
                  "< " +
                  parBrowseElement.getElement().firstChild.nodeValue;
               addItem(parBrowseElement);
            }
            alert("Path: " + parItemPath + "\nName: " + parItemName);
         };
         var tmpNewNodeFct = function(parBrowseElement)
         {
             parBrowseElement.getElement().firstChild.nodeValue =
                 "+ " +
                 parBrowseElement.getElement().firstChild.nodeValue;
             addItem(parBrowseElement);
         };
         var tmpNewLeafFct = function(parBrowseElement)
         {
             var tmpCB = function(parBrowseElement)
             {
```

```
                    var tmpText = "Path: " +
                                  parBrowseElement.mItemPath + "\n" +
                                  "Name: " +
                                   parBrowseElement.mItemName + "\n";
                    alert(tmpText);
                }
                parBrowseElement.setCB(tmpCB);
                addItem(parBrowseElement);
            };
            gloBrowser = new ApplBrowser(document, "", "",
                                         tmpNewTreeFct,
                                         tmpNewNodeFct,
                                         tmpNewLeafFct);
        }
        function leave()
        {
            if (gloBrowser)
            gloBrowser.destructor();
        }
     </script>
    </head>
    <body onload="browse();" onunload="leave();">
        <div id="BrowseTable"></div>
    </body>
 </html>
```

ApplBrowseTree

`ApplBrowseTree` builds upon the `ApplBrowser` and displays the browse result in tree format.

```
ApplBrowseTree(parDocument,
               parItemPath,
               parItemName,
               parTablePrefix,
               parLeafCB,
               parNodeCB,
               parBackCB,
               parFilterCB,
               parLeafImg,
               parNodeImg,
               parBackImg)
```

Transfer parameters:

- `parDocument`
  : JavaScript document used to generate elements
- `parItemPath`, `parItemName`
  : Specifies the starting point for browsing the variable management.
- `parTablePrefix`
  Prefix for referencing elements in CSS
- `parLeafCB`
  Callback function that is called if a leaf is clicked
- `parNodeCB`
  Callback function that is called if a node is clicked. Each of the callback functions receives an `ApplBrowseElement` as a first parameter.
- `parBackCB`
  Callback function that is called if the first element of the tree is clicked
- `parLeafImg`, `parNodeImg`, `parBackImg`
  : Symbols that are displayed by a back, node, or leaf element.

User interface:

- `getElement()`
  Supplies a table element that contains the browse results.
- `destructor()`
  Releases the table and all resources connected to it

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<html>
  <head>
     <title>Browse table demo</title>
     <style type="text/css">
       table.BrowseTree {table-layout:fixed;}
       td.BrowseTreeImg0 {width:20px;}
       td.BrowseTreeImg1 {width:20px;}
       td.BrowseTreeBrowse {overflow:visible;text-align:left;}
       a.refLeaf {cursor:pointer;}
       a.refNode {cursor:pointer;}
     </style>
     <script type="text/javascript" src="/common.js"></script>
     <script type="text/javascript" src="/opcxml.js"></script>
     <script type="text/javascript" src="/appl.js"></script>
     <script type="text/javascript">
     var gloBrowseTree = null;
     function browse()
     {
        var tmpLeafCB = function(parBrowseElement)
        {
            var tmpText = "Path: " + parBrowseElement.mItemPath + "\n"
                          + "Name: " + parBrowseElement.mItemName +
                          "\n";
            alert(tmpText);
        }
        var tmpFilterCB = function(parBrowseElement)
        {
            var tmpItemName = parBrowseElement.mItemName;
            var tmpPos = tmpItemName.indexOf("unit/");
            if (tmpPos != 0)
            {
                tmpPos = tmpItemName.indexOf("to/");
                if (tmpPos != 0)
                   tmpPos = tmpItemName.indexOf("var/");
            }
            return (tmpPos == 0);
         }
         gloBrowseTree = new ApplBrowseTree(document,
                                            "SIMOTION",
                                            "",
                                            "BrowseTree",
                                            tmpLeafCB,
                                            undefined,
                                            undefined,
                                            tmpFilterCB
                                            );
        var tmpBrowseTreeHook =
                             document.getElementById("BrowseTree");
        tmpBrowseTreeHook.appendChild(gloBrowseTree.getElement());
      }
      function leave()
      {
          if (gloBrowseTree)
```

```
            gloBrowseTree.destructor();
        }
      </script>
    </head>
    <body onload="browse();" onunload="leave();">
      <div id="BrowseTree"></div>
    </body>
  </html>
```

JavaScript and file accesses

File access with JavaScript

The SIMOTION Web server provides a JavaScript API with which file operations can be performed.

Delete	Delete a file.
Rename	Rename a file.
Read	Read a file.
Write	Write a file.
Stat	Read the file properties (creation date, access rights, ...).
Browse	Browse the content of a directory.
MakePath	Create a directory path.
DeleteDir	Delete a directory.

Transactions

To perform consistent multiple file operations, a transaction mechanism is provided for the file accesses. All function calls (transaction API and the File Manager API) are made asynchronously.

A transaction is initiated by calling the `Begin` function. The transaction object is provided for successive operations that must be performed consistent. After the operation sequence has been processed, the transaction is completed by calling the `Commit` function.

To abort an operation sequence, the `Abort` function of the transaction must be called. In this case, all operations performed previously as transaction are rolled back.

Transactions are time-monitored in the runtime system. If a transaction is inactive for longer than 5 seconds, it is terminated automatically and all previous operations are rolled back.

Each function can be passed a data object that is returned in the Callback function.

```
function Transaction
{
 Begin: function(parData, parCB) {...};
 Commit: function(parData, parCB) {...};
 Abort: function(parData, parCB) {...};
```

```
 // private
 Run: function(parParams, parData, parCB) {...};
}
```

Begin

The `Begin` function initiates a transaction. The time monitoring is started and a transaction ID is generated.

`Begin: function(parData, parCB)`

Transfer parameters:

- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

`function(parRetVal)`

Transfer parameters:

- `parRetVal::`
  ```
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  ```
  `"Result":`
  - `"RC_OK"`: The function completed successfully.
  - `"RC_UNREACHABLE"`: The target controller cannot be reached.
  - `"RC_TA_ABORTED"`: The transaction was aborted by the user.
  - `"RC_TA_FINISHED"`: The transaction has already been completed with `Commit`.
  - `"RC_TA_IDLE"`: The transaction has not yet been activated with `Begin`.
  - `"RC_TA_TIMEOUT"`: The transaction has been completed by a timeout.

  `"Data":`
  - Variables defined by the user passed as `parData` parameters for the function call.

Commit

The `Commit` function completes a transaction.

`Commit: function(parData, parCB)`

Transfer parameters:

- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- `parRetVal::`
  ```
  {
   "Result": <ReturnCode>
  }
  "Result":
  ```
  - "`RC_OK`": The function completed successfully.
  - "`RC_UNREACHABLE`": The target controller cannot be reached.
  - "`RC_TA_ABORTED`": The transaction was aborted by the user.
  - "`RC_TA_FINISHED`": The transaction has already been completed with `Commit`.
  - "`RC_TA_IDLE`": The transaction has not yet been activated with `Begin`.
  - "`RC_TA_TIMEOUT`": The transaction has been completed by a timeout.

Abort

The `Abort` function aborts a transaction. All performed operations that belong to this transaction are rolled back.

```
Abort: function(parData, parCB)
```

Transfer parameters:

- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- parRetVal::
  {
   "Result": <ReturnCode>
  }
  "Result":
  - "RC_OK": The function completed successfully.
  - "RC_UNREACHABLE": The target controller cannot be reached.
  - "RC_TA_ABORTED": The transaction was aborted by the user.
  - "RC_TA_FINISHED": The transaction has already been completed with Commit.
  - "RC_TA_IDLE": The transaction has not yet been activated with Begin.
  - "RC_TA_TIMEOUT": The transaction has been completed by a timeout.

File Manager object

The File Manager API uses JavaScript file library to perform file operations that lie within the valid range of the Web server.

```
function FileManager(parTransaction)
{
 Delete: function(parParams, parData, parCB) {...};
 Rename: function(parParams, parData, parCB) {...};
 Read: function(parParams, parData, parCB) {...};
 Write: function(parParams, parData, parCB) {...};
 Stat: function(parParams, parData, parCB) {...};
 Browse: function(parParams, parData, parCB) {...};
 MakePath: function(parParams, parData, parCB) {...};
 DeleteDir: function(parParams, parData, parCB) {...};
}
```

Delete

The Delete function deletes the passed file.

```
Delete: function(parParams, parData, parCB)
```

Transfer parameters:

- `parParams`
  ```
  {
   "Path": <Path>
  }
  ```
  `"Path"`: The absolute path name of the file to be deleted.
- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- `parRetVal::`
  ```
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  ```
  `"Result":`
  - "`RC_OK`": The function completed successfully.
  - "`RC_UNREACHABLE`": The target controller cannot be reached.
  - "`RC_TA_ABORTED`": The transaction was aborted by the user.
  - "`RC_TA_FINISHED`": The transaction has already been completed with `Commit`.
  - `"RC_TA_IDLE"`: The transaction has not yet been activated with `Begin`.
  - `"RC_TA_TIMEOUT"`: The transaction has been completed by a timeout.
  - `"RC_FS_PATH_NOT_FOUND"`: The desired file does not exist.
  - `"RC_FS_ACCESS_DENIED"`: The current access rights do not suffice to perform the desired action.

  `"Data":`
  - Variables defined by the user passed as `parData` parameters for the function call.

Rename

The `Rename` function renames a file or a directory.

```
Rename: function(parParams, parData, parCB)
```

Transfer parameters:

- parParams
  {
   "SrcPath": <Path>
   "DestPath": <Path>
  }
  "SrcPath": The absolute path name of the file to be renamed.
  "DestPath": New file name without path details.
- parData: User-defined data mirrored by the Callback function.
- parCB: The Callback function that is called after completion of the action.

parCB: Callback function

function(parRetVal)

Transfer parameters:

- parRetVal::
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  "Result":
  - "RC_OK": The function completed successfully.
  - "RC_UNREACHABLE": The target controller cannot be reached.
  - "RC_TA_ABORTED": The transaction was aborted by the user.
  - "RC_TA_FINISHED": The transaction has already been completed with Commit.
  - "RC_TA_IDLE": The transaction has not yet been activated with Begin.
  - "RC_TA_TIMEOUT": The transaction has been completed by a timeout.
  - "RC_FS_PATH_NOT_FOUND": The desired file does not exist.
  - "RC_FS_ACCESS_DENIED": The current access rights do not suffice to perform the desired action.
  - "RC_FS_EXISTS": The new file name exists already.

  "Data":
  - Variables defined by the user passed as parData parameters for the function call.

Read

The Read function reads a file.

Read: function(parParams, parData, parCB)

Transfer parameters:

- `parParams`
  ```
  {
   "Path": <Path>
   "Type": <FileType>
  }
  ```
  `"Path"`: The absolute path of the file to be read.
  `"Type"`: Describes the content of a file.
  - `"xml"`: The file contains an XML document.
  - `"json"`: The file contains a JSON object.
  - `"text"`: The file contains a UTF8 document.
  - `"binary"`: The file contains binary data with Base64 coding.
- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- parRetVal::
  {
   "Result": <ReturnCode>
   "Data": <UserData>
   "File": <FileDsc>
  }
  "Result":
  - "RC_OK": The function completed successfully.
  - "RC_UNREACHABLE": The target controller cannot be reached.
  - "RC_TA_ABORTED": The transaction was aborted by the user.
  - "RC_TA_FINISHED": The transaction has already been completed with Commit.
  - "RC_TA_IDLE": The transaction has not yet been activated with Begin.
  - "RC_TA_TIMEOUT": The transaction has been completed by a timeout.
  - "RC_FS_PATH_NOT_FOUND": The desired file does not exist.
  - "RC_FS_ACCESS_DENIED": The current access rights do not suffice to perform the desired action.
  - "RC_FS_EXISTS": The new file name exists already.

  "Data":
  - Variables defined by the user passed as parData parameters for the function call.

  "File":
  - File type and content
    {
     "Type": <FileType>,
     "Content": <FileData>
    }
    "Type": Describes the content of a file (see above).
    "Content": The content of the file in the appropriate "Type" format.

Write

The Write function writes a file.

```
Write: function(parParams, parData, parCB)
```

Transfer parameters:

- `parParams`
  ```
  {
   "Path": <Path>
   "File": <FileDsc>
  }
  ```
  `"Path"`: The absolute path of the file to be written.
  `"File"`: File type and content.
  ```
    {
      "Type": <FileType>,
      "Content": <FileData>
    }
  ```
  `"Type"`: Describes the content of a file.
  - `"xml"`: The file contains an XML document.
  - `"json"`: The file contains a JSON object.
  - `"text"`: The file contains a UTF8 document.
  - `"binary"`: The file contains binary data with Base64 coding.

  `"Content"`: The content of the file in the appropriate `"Type"` format.
- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- parRetVal::
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  "Result":
  - "RC_OK": The function completed successfully.
  - "RC_UNREACHABLE": The target controller cannot be reached.
  - "RC_TA_ABORTED": The transaction was aborted by the user.
  - "RC_TA_FINISHED": The transaction has already been completed with Commit.
  - "RC_TA_IDLE": The transaction has not yet been activated with Begin.
  - "RC_TA_TIMEOUT": The transaction has been completed by a timeout.
  - "RC_FS_PATH_NOT_FOUND": The desired file does not exist.
  - "RC_FS_ACCESS_DENIED": The current access rights do not suffice to perform the desired action.
  - "RC_FS_EXISTS": The new file name exists already.
  - "RC_FS_DISK_FULL": The storage medium is full.

  "Data":
  - Variables defined by the user passed as parData parameters for the function call.

Stat

The Stat function returns information about the element (file or directory) specified with Path.

```
Stat: function(parParams, parData, parCB)
```

Transfer parameters:

- parParams
  {
   "Path": <Path>
  }
  "Path": The absolute path of the file or directory.
- parData: User-defined data mirrored by the Callback function.
- parCB: The Callback function that is called after completion of the action.

parCB: Callback function

```
function(parRetVal)
```

Transfer parameters:

- `parRetVal::`
  `{`
  `"Result": <ReturnCode>`
  `"Data": <UserData>`
  `"Element": <Properties>`
  `}`
  `"Result":`
  - "`RC_OK`": The function completed successfully.
  - "`RC_UNREACHABLE`": The target controller cannot be reached.
  - "`RC_TA_ABORTED`": The transaction was aborted by the user.
  - "`RC_TA_FINISHED`": The transaction has already been completed with `Commit`.
  - "`RC_TA_IDLE`": The transaction has not yet been activated with `Begin`.
  - "`RC_TA_TIMEOUT`": The transaction has been completed by a timeout.
  - "`RC_FS_PATH_NOT_FOUND`": The desired file does not exist.

  `"Data":`
  - Variables defined by the user passed as `parData` parameters for the function call.
- `"Element":`
  - Contents
    `{`
    `"Name":` <ElementName>,
    `"Type":` <ElementType>
    `"Access":` <AccessAuthorization>,
    `"Size":` <FileSize>,
    `"CTime":` <DateAndTime>
    `}`
    `"Name":` File name without path details.
    `"Type":` Unit type.
    - "`DIR`": It refers to a directory.
    - "`FILE`": It refers to regular file.

    `"Access":` Current access authorization.
    - "`rw`": Read and write.
    - "`ro`": Read-only.
    - "`wo`": Write-only.

    `"Size":` The size of the file in bytes (present only for regular files).
    `"CTime":` The date and time when created.

Browse

The `Browse` function reads the content of a directory.

```
Browse: function(parParams, parData, parCB)
```

Transfer parameters:

- `parParams`
  ```
  {
   "Path": <Path>
  }
  ```
  `"Path"`: The absolute path of the directory.
- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- `parRetVal::`
  `{`
  `"Result": <ReturnCode>`
  `"Data": <UserData>`
  `"Elements": <ElementList>`
  `}`
  `"Result":`
  - "`RC_OK`": The function completed successfully.
  - "`RC_UNREACHABLE`": The target controller cannot be reached.
  - "`RC_TA_ABORTED`": The transaction was aborted by the user.
  - "`RC_TA_FINISHED`": The transaction has already been completed with `Commit`.
  - "`RC_TA_IDLE`": The transaction has not yet been activated with `Begin`.
  - "`RC_TA_TIMEOUT`": The transaction has been completed by a timeout.
  - "`RC_FS_PATH_NOT_FOUND`": The desired file does not exist.
  - "`RC_FS_ACCESS_DENIED`": The current access rights do not suffice to perform the desired action.
  - "`RC_FS_NO_DIRECTORY`": The passed path is not a directory.

  `"Data":`
  - Variables defined by the user passed as `parData` parameters for the function call.
- `"Elements"::` The array elements of a directory.
  - `Elements`
    `[`
    `{`
    `"Name":` <ElementName>,
    `"Type":` <ElementType>
    `"Access":` <AccessAuthorization>,
    `"Size":` <FileSize>,
    `"CTime":` <DateAndTime>
    `}`
    `"Name":` File name without path details.
    `"Type":` Unit type.
    - `"DIR"`: It refers to a directory.
    - `"FILE"`: It refers to a regular file.

    `"Access":` Current access authorization.
    - `"rw"`: Read and write.
    - `"ro"`: Read-only.
    - `"wo"`: Write-only.

    `"Size":` The size of the file in bytes (present only for regular files).
    `"CTime":` The date and time when created.
    `]`

MakePath

The MakePath function creates the passed path recursively.

```
MakePath: function(parParams, parData, parCB)
```

Transfer parameters:

- `parParams`
  ```
  {
   "Path": <Path>
  }
  ```
  `"Path"`: The path to be created.
- `parData`: User-defined data mirrored by the Callback function.
- `parCB`: The Callback function that is called after completion of the action.

`parCB`: Callback function

```
function(parRetVal)
```

Transfer parameters:

- `parRetVal::`
  ```
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  ```
  `"Result":`
  - `"RC_OK"`: The function completed successfully.
  - `"RC_UNREACHABLE"`: The target controller cannot be reached.
  - `"RC_TA_ABORTED"`: The transaction was aborted by the user.
  - `"RC_TA_FINISHED"`: The transaction has already been completed with `Commit`.
  - `"RC_TA_IDLE"`: The transaction has not yet been activated with `Begin`.
  - `"RC_TA_TIMEOUT"`: The transaction has been completed by a timeout.
  - `"RC_FS_ACCESS_DENIED":` The current access rights do not suffice to perform the desired action.
  - `"RC_FS_DISK_FULL":` The storage medium is full.

  `"Data":`
  - Variables defined by the user passed as `parData` parameters for the function call.

DeleteDir

The `DeleteDir` function deletes the passed directory. The directory can be deleted only when it is empty.

```
DeleteDir: function(parParams, parData, parCB)
```

Transfer parameters:

- parParams
  ```
  {
   "Path": <Path>
  }
  ```
  "Path": The absolute path of the directory to be deleted.
- parData: User-defined data mirrored by the Callback function.
- parCB: The Callback function that is called after completion of the action.

parCB: Callback function

```
function(parRetVal)
```

Transfer parameters:

- parRetVal::
  ```
  {
   "Result": <ReturnCode>
   "Data": <UserData>
  }
  ```
  "Result":
  - "RC_OK": The function completed successfully.
  - "RC_UNREACHABLE": The target controller cannot be reached.
  - "RC_TA_ABORTED": The transaction was aborted by the user.
  - "RC_TA_FINISHED": The transaction has already been completed with Commit.
  - "RC_TA_IDLE": The transaction has not yet been activated with Begin.
  - "RC_TA_TIMEOUT": The transaction has been completed by a timeout.
  - "RC_FS_ACCESS_DENIED": The current access rights do not suffice to perform the desired action.
  - "RC_FS_NO_DIRECTORY": The passed path is not a directory.
  - "RC_FS_NOT_EMPTY": The directory is not empty.

  "Data":
  - Variables defined by the user passed as parData parameters for the function call.

MiniWeb Server Language (MWSL)

Mode of operation of the MWSL

The web server language is a scripting language that is interpreted on the web server. This scripting language is fairly similar to JavaScript, but is only a small subset of the complete language.

The MWSL enables the client to be operated with a simple browser without scripting, because the web server generates the pages dynamically.

MWSL enables access and processing of variables. Among other things, MWSL allows access to process variables that are present on the web server system. MWSL and the integrated template mechanism can then be used very effectively to process and evaluate these variables.

The template mechanism used to produce the dynamic pages is similar to a very simplified XSLT process. See W3C XSL transformation (http://www.w3.org/standards/xml/transformation)

The client requests a URL on the web server. The address is converted to the access to an MWSL file in the web server. From this file, the MWSL service generates a temporary HTML file on the web server. This file is then sent to the client and displayed there.

Note

Enabling access to the process variables

Access to the process variables with the MWSL functions GetVar and SetVar requires activation of the setting **Enable OPC_XML (load symbols to RT)** in SCOUT.

You can find more information in the manual SIMOTION IT Diagnostics and Configuration, in the chapters 'Accessing the global variables (V4.2 and higher)' and 'Making unit variables available'.

Structure of a MWSL file

An MWSL file is basically an HTML file which also contains MWSL tags. To distinguish them from HTML files, MWSL files have the extension ".mwsl." Loading of MWSL pages into the controller (Page 3746)

Example:

```
<HTML>
   <HEAD>
      …
   </HEAD>
   <BODY>
      <table>
         <tr>
            [...]
            <td>
               <MWSL>
               //MWSL code to be executed
               </MWSL>
            </td>
            [...]
            <td>
               <MWSL>
               //MWSL code to be executed
               </MWSL>
            </td>
            [...]
         </tr>
      </table>
   </BODY>
</HTML>
```

If the MWSL functionality is required, the following tags must be added:

- The <MWSL>–tag introduces an MWSL script
- The </MWSL>-tag marks the end of the script

For reasons of clarity, the examples below do not always include the HTML code and begin directly with the <MWSL>" tag.

Error messages

MWSL error messages

Errors that occur while an MWSL page is being used are output in two different ways:

1. Errors that occur when compiling the MWSL file are output in the logfile. The name of the log file is formed by appending ".log" to the file name.
2. Errors that occur when executing a web page, that is, when the page is called from the server, are written into the source text of the MWSL page as error messages.

Error output in the MWSL page

The source text of a page can be loaded to the editor in the Internet Explorer by right-clicking and selecting the menu command **Show Source Text**.

Example

This example shows the comment associated with the query relating to an unavailable variable.

exec.mwsl

```
<html>
  <head>
    <title>SIMOTION
      <MWSL>
        WriteVar("DeviceInfo.Board");
      </MWSL>
    </title>
    <meta name="DC.Subject" content="SIMOTION">
    <meta name="DC.Publisher" content="Siemens AG">
    <meta name="DC.Format" content="text/html">
    <meta name="DC.Language" content="en">
    <meta name="DC.Rights" content="Copyrights Siemens AG 2003">
  </head>
  <body style="font-family: Arial">
    <p>
    <MWSL>WriteVar("var/userData.user8");</MWSL>
    </p>
    <p>
    <MWSL>WriteVar("var/userData.user9");</MWSL>
    </p>
  </body>
</html>
```

In exec.mwsl file, the statement `WriteVar("var/userData.user9");` is used to query a non-existent variable.

Source text for the output page:

```
<html>
 <head>
  <title>SIMOTION D435</title>
  <meta name="DC.Subject" content="SIMOTION">
  <meta name="DC.Publisher" content="Siemens AG">
  <meta name="DC.Format" content="text/html">
  <meta name="DC.Language" content="en">
  <meta name="DC.Rights" content="Copyrights Siemens AG 2003">
 </head>
 <body style="font-family: Arial">
  <p>
  495399
  </p>
  <p>

<!-- MWSL2 Runtime Message (WARNING):
     Src Line# (MethodName) Source Text
     ============================
     000003  #(INTERPRETER) WriteVar("var/userData.user9");</MWSL>
     External function returned an error: E6B20014.
-->

  </p>
 </body>
</html>
```

The MWSL2 Runtime Message comment contains a description of the error cause.

Note

Unambiguity of error messages

Error messages are not always unambiguous. This can be with regard to the specification of the function as well as the error line.

Variable types

MWSL distinguishes between script variables and global variables:

- Script variables are defined within the script
- Global variables are provided by variable sources

Note

Global variables are not part of the script engine. Rather, they represent information from the web server environment. Variables are accessed exclusively via access functions. The global variables are grouped into variable sources according to their origin.

Script variables

Script variables are variables that are only valid on the current page.

The variables apply beyond MWSL tags, i.e. they can be created in one MWSL tag and be used starting in the next MWSL tag.

For these variables, no distinction is made between variable types, i.e. there is no Int, Char, etc.

A variable is created as follows:

```
var <Variablenname> = <Wert>;
```

The variable type is determined internally by the variable assignment.

Example:

```
<MWSL>
   var string1 = "Hello";
   var string2 = "World";
   write(string1 + " " + string2);
</MWSL>
```

Two variables are created in the example shown above: `string1` and `string2`.

The two strings are strung together (with spaces).

The `write` command outputs the result. See write (Page 3878)

Output: Hello World

Example:

```
<MWSL>
   var num1 = 5;
   var num2 = 7;
   var Result;
   Result = num1 + num2;
</MWSL>
```

Two variables are created in the example shown above: `num1` and `num2`.

The two numbers are added, and the result is stored in the `Result` variable.

`Result` contains the value 12.

The data type is converted in the same way as ECMA Script 262. See ECMA script (http://www.ecma-international.org/publications/standards/Ecma-262.htm)

Keyword var

The keyword var introduces a variable declaration. In ECMA script, variables do not have to be explicitly declared.

Syntax:

```
var VarName = InitalValue, VarName2 = InitialValue2, …;
```

Multiple variables are declared and (optionally) initialized with initial values.

You can specify several declarations, separated by commas.

For further information, please refer to the ECMA Script Definitions.

Ranges of visibility and validity

The visibility and validity of variables is analogous to ECMA script. (However, MWSL currently does not recognize any functions.)

Example:

```
<MWSL>
   var MyVar = 10;
   {
      MyVar = 20;
      write ("Inner:" + MyVar + "," );
   }
   write ("Outer:" + MyVar + "\n" );
</MWSL>
```

Output: Inner: 20, Outer: 20

In this example, the variable `MyVar` of the outer level will be accessed in the operation block, because a variable named `MyVar` was not declared on the operation block level.

Therefore, the `MyVar = 20` operation changes the value of the variable on the outer level.

```
<MWSL>
   var MyVar = 10;
   {
       var MyVar = 20;
       write ("Inner:" + MyVar + "," );
   }
   write ("Outer:" + MyVar + "\n" );
</MWSL>
```

Output: Inner: 20, Outer: 10

In this example, the variable `MyVar` is created new at the inner level, which leaves the value of the variable `MyVar` having the same name at the outer level unchanged.

Global variables

Definition

Global variables enable access to the variable management area of the web server. There are different types of global variables:

- PROCESS variables enable access to normal variables of the web server. This is known as standard access.
- DEFAULT variables are identical to the PROCESS variables.
- URL variables provide access to variables contained in a URL.
- HTTP variables return the content of variables in the HTTP header.
- COOKIE variables allow access to cookies.

A variable can be accessed using the following command:

```
GetVar("var/userData.user1", "PROCESS");
```

The variable source `PROCESS` must be written in upper case. If the `var/userData.user1` variable is not present, "null" is returned.

`PROCESS` is the standard variable source. Therefore, `PROCESS` can also be omitted.

```
GetVar("var/userData.user1");
```

If a variable provider is to be addressed directly, the name of the desired provider can be specified instead of the variable source `PROCESS`.

Format string for the GetVar and WriteVar functions

The format string always starts with a % sign, followed by the specification of the number of characters. This is then followed by the type information.

The following type information is available:

- %d for Integer values
- %f for Float values
- %e for representing the value with exponent
- %s for Strings

Examples

```
GetVar("var/systemClock", "PROCESS", "%d");
```

```
GetVar("var/modeOfOperation", "PROCESS", "%s");
```

Table 6-41 Format strings

Example	Meaning
"%.6s"	Outputs the first 6 characters of the specified variable (as a string).
"%.3s"	Outputs the first 3 characters of the specified variable (as a string).
"%.s"	Outputs the complete variable (as a string).
"%3.2f"	Outputs the variable interpreted as Float. The 3 indicates that 3 total places are output. The 2 indicates that, of the 3 places, 2 places after the decimal point will be displayed.
"%4d"	Outputs the variable interpreted as Integer. Four places are output. This parameter can only be passed if the variable source "PROCESS" has also been passed.

If the format string is omitted, the complete variable content is returned.

Float numbers are rounded at output. For example, the number pi is output with the formatting "`%4.3f`" as `3,142`

See also

GetVar (Page 3869)

WriteVar (Page 3879)

Special variables

Via the variable source HTTP, the value of the special variable `Username` can be queried.

```
GetVar("Username", "HTTP")
```

This call of `GetVar` returns the user currently logged on to the Web server.

The special variable `HTTP` supplies information on the HTTP connection.

```
GetVar( "HTTP", "HTTP")
```

Output:

```
Host: 192.168.1.1User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:28.0) Gecko/20100101 Firefox/28.0Accept: text/html,application/
xhtml+xml,application/xml;q=0.9,*/*;q=0.8Accept-Language: de,en-
US;q=0.7,en;q=0.3Accept-Encoding: gzip, deflateReferer: http://
192.168.1.1/index.mwslConnection: keep-alive
```

Configuration constants

Accessing constants of the WebCfg.xml configuration file

It is possible to create constant variables in the WebCfg.xml file. Accessing these constants enables more flexible programming, meaning that user-defined pages can be controlled using relevant configuration parameters.

A constant is defined in the following section of WebCfg.xml:
/SERVERPAGES/CONFIGURATION_DATA/UserConfig

```
<SERVERPAGES>
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <MyParam>MyParamValue</MyParam>
    </USERCONFIG>
  </CONFIGURATION_DATA>
</SERVERPAGES>
```

It is possible to access the `MyParam` constant in an HTML page by specifying the "`constants/MyParam`" path.

```
<html>
  <head>
  </head>
  <body style="font-family: Arial">
    <table width="100%" height="100%" bgcolor="#00349A">
      <tr>
        <td style="color: #FFFFFF">
          <b><MWSL>WriteVar("constants/MyParam");</MWSL></b>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Result:

```
<html>
  <head>
  </head>
  <body style="font-family: Arial">
    <table width="100%" height="100%" bgcolor="#00349A">
      <tr>
        <td style="color: #FFFFFF">
          <b>MyParamValue</b>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Variables and URL parameters

MWSL offers the option of editing URL parameter values using the `WriteVar`, `GetVar`, `SetVar`, and `ExistVariable` functions.

Example of a URL with appended parameters. The line break has been added for better readability:

```
http://localhost/MWSL/StringOperationtest.mwsl?
Parameter1=Hallo&Parameter2=du!&StartValue=2&EndValue=5
```

The URL points to the page StringOperationtest.mwsl and transfers the parameters `Parameter1`, `Parameter2`, `StartValue` and `EndValue`.

The following command outputs the URL variable Parameter1:

```
WriteVar("Parameter1", "URL");
```

Note that "URL" must be written in upper case.

If a URL variable that is not present in the URL is requested, an empty string ("") is always returned. This return is not a script error.

Parameters in URLs

In a URL, parameter passing begins after the "?" character. Individual parameters are separated by "&" characters. The value is assigned after the "=" character.

Certain characters require a coding in order to be passed correctly. The following table provides an overview of the most commonly used escape codes.

Table 6-42 URL escape codes

Character	Escape code
Blank	%20
<	%3C
>	%3E
#	%23
%	%25

Character	Escape code
{	%7B
}	%7D
|	%7C
\	%5C
^	%5E
~	%7E
[	%5B
]	%5D
`	%60
;	%3B
/	%2F
?	%3F
:	%3A
@	%40
=	%3D
&	%26
$	%24

The coding consists of the '%' character followed by the ASCII hexadecimal value of the desired character.

You can find more information on the Internet, e.g. at http://de.selfhtml.org/.

COOKIES

The following example shows how a cookie can be created in an MWSL file.

For this purpose, insert the <META> tag into the <HEAD> tag:

```
http-equiv="SET-COOKIE" content="siemens_automation_language=de;"
```

Example:

```
<HTML>
   <HEAD>
      <META http-equiv="SET-COOKIE"
            content="siemens_automation_language=de;">
   </HEAD>
   <BODY>
      [...]
   </BODY>
</HTML>
```

For more information on cookies: http://de.selfhtml.org/

Setting a cookie as an HTTP header

It is possible to set HTTP header for the HTTP response from MWSL. For example, this allows a cookie to be set via the HTTP header and not as a <META> tag.

Example:

```
<MWSL>
   var strCookie;
   strCookie = "Set-cookie: siemens_automation_language=";
   strCookie = strCookie + GetVar( "Language", "URL");
   strCookie = strCookie + ", path=/\r\n";
   AddHTTPHeader( strCookie );

   write("HTTP lang: " + GetVar("Cookie", "HTTP"));
</MWSL>
```

In this example, a cookie for detection of the language setting is set.

Variables and access to COOKIES

The access occurs similarly as for the URL parameters. The single difference is that the variable type is now `COOKIE` instead of `URL`.

The variable specified in the example can be accessed with the following command, for example:

```
GetVar("siemens_automation_language", "COOKIE");
```

Note that `COOKIE` must be written in upper case.

If the `COOKIE siemens_automation_language` is set with `en`, for example, the above call would return this value.

For more information on cookies: http://de.selfhtml.org/

If the variable is not present, there is no output.

Variables and HTTP header information

Using the variable source `HTTP`, the values are read out of the HTTP query header fields. These are, for example, `Accept`, `Content-Type`, or `User-Agent`.

Table 6-43 Example output of an HTTP header field

```
WriteVar("User-Agent", "HTTP");

Output of the Mozilla Firefox browser:
  Mozilla/5.0 (Windows NT 6.1; WOW64; rv:25.0) Gecko/20100101 Firefox/25.0
```

Table 6-44 Example of generation of an HTTP HEADER with the HTML META tag

Example:

```
<HTML>
   <HEAD>
      [...]
      <META http-equiv="Accept-Language" content="de">
      [...]
   </HEAD>
   <BODY>
      [...]
   </BODY>
</HTML>
```

In this example, the HTTP variable `Accept-Language` is defined in the META tag using the "`http-equiv="Accept-Language"` command. It is initialized by the `content` attribute with the value `de`.

Additional information on META information: http://de.selfhtml.org/html/kopfdaten/meta.htm#allgemeines

This variable is accessed similarly as for the URL parameters. The difference is that the variable source is `HTTP` and not `URL`. The variable specified in the example can be accessed with the following command:

```
GetVar("Accept-Language", "HTTP");
```

The variable source `HTTP` must be written in upper case.

Format string for the GetVar and WriteVar functions

As the third parameter of the functions `GetVar` and `WriteVar`, a format string can be specified. The format string defines from which position how many characters of the variable source will be returned.

```
GetVar("Accept-Language", "HTTP", "[3,5]")
```

In the above example, 5 characters of the HTTP variables `Accept-Language` are returned starting from the third character.

The format string can only be passed if a variable source, such as `HTTP` in the above example, has also been passed.

If the format string is omitted, the complete variable content is returned.

See also

Global variables (Page 3802)

GetVar (Page 3869)

WriteVar (Page 3879)

Operators

MWSL operators

All operators presented here behave as defined in ECMA 262. For more information, refer to the ECMA 262 Specification.

Boolean values are converted to numerical values 0 (for false) and 1 (for true) where applicable.

Table 6-45 Relational operators

Operator	Comment
`x < y`	This operator returns true if the `x` variable is less than the `y` variable. Otherwise, the value returned is false.
`x <= y`	This operator returns true if the `x` variable is less than or equal to the right `y` variable. Otherwise, the value returned is false.
`x > y`	This operator returns true if the `x` variable is greater than the `y` variable. Otherwise, the value returned is false.
`x >= y`	This operator returns true if the `x` variable is greater than or equal to the `y` variable. Otherwise, the value returned is false.
`x == y`	This operator returns true if the `x` variable is equal to the `y` variable. Otherwise, the value returned is false.
`x != y`	This operator returns true if the `x` variable is not equal to the `y` variable. Otherwise, the value returned is false.

Table 6-46 Logic operators

Operator	Remark
`!x`	Logical NOT This operator returns a false if `x` is true. if `x` is false, the value returned by the operator will be true.
`x && y`	Logical AND This operator returns true if `x` and `y` have the value true. Otherwise, the value returned is false.
`x || y`	Logical OR This operator returns a false if x and y have the value false. Otherwise, the value returned is true.

Table 6-47 Object operators

Operator	Remark
`new`	Creating an object This operator creates on object. The object can be of a pre-integrated or a user-defined type.
`delete`	Deleting an object This operator deletes an object created with `new`.

Table 6-48 Bit-by-bite operators

Operator	Remark
x & y	Bit-by-bit AND If x and y are one at a bit position, this bit position will have the value one. All bit positions are checked.
x | y	Bit-by-bit OR If x or y are one at a bit position, this bit position will have the value one. All bit positions are checked.
x ^ y	Bit-by-bit XOR If x or y is one at a bit position, but they are not both one, this bit position will have the value one. All bit positions are checked.
x >> y	Right shift considering the sign In x, the bits are shifted right by y digits. If x is positive, zeroes are filled in from the left; if x is negative, ones are filled in from the left.
x >>> y	Right shift without considering the sign In x, the bits are shifted right by y digits. Zeroes are filled in from the left.
x << y	Left shift In x, the bits are shifted left by y digits. Zeroes are filled in from the right.
~x	Bit-by-bit NOT Inverted bits of x.

Table 6-49 Arithmetic operators

Operator	Remark
x + y	Plus operator This operator adds x and y.
++x	Increment Increments the value of x by one. x++ or ++x is possible.
x − y	Minus operator This operator subtracts the value y from x.
−−x	Decrement Decrements the value of x by one. x−− or x is possible.
x / y	This operator divides x by y. The return value is a float.
x % y	Modulo This operator returns the integer remainder of a division of x and y.
x * y	This operator multiplies x by y.

Table 6-50 Assignment operators

Operator	Remark
`x = y`	Assignment operator This operator assigns the value of the right-hand expression to `x`, in this case the variable `y`.
`x += y`	Addition Assigns `x` the value of `x + y`.
`x -= y`	Subtraction Assigns `x` the value of `x - y`.
`x *= y`	Multiplication Assigns `x` the value of `x * y`.
`x /= y`	Division Assigns `x` the value of `x / y`.
`x ^= y`	Bit-by-bit OR Assigns `x` the value of `x ^ y`.
`x |= y`	Bit-by-bit OR Assigns `x` the value of `x | y`.
`x &= y`	Bit-by-bit AND Assigns `x` the value of `x | y`.
`x %= y`	Modulo Assigns to `x` the integer remainder of a division with `y`.
`x >>= y`	Bit-by-bit right shift Assigns `x` the value of `x >> y`.
`x <<= y`	Bit-by-bit left shift Assigns `x` the value of `x << y`.

Table 6-51 Conditional operator

Operator	Remark
`Condition ? x : y`	Conditional operator This operator has the value `x`, if the `condition` is true. Otherwise, the value of the operator is `y`.

Conditional operations

if Conditions

The `if` condition is familiar from Ecma 262.

Syntax
```
if(<condition>)
   Operation1
 else
   Operation2;
```

If the condition is true, instruction 1 is run.

If the condition is false, instruction 2 is run. If no `else` part is present, processing continues in accordance with the `if` operation.

Example:
```
<MWSL>
  [...]
  if (ExistVariable("Parameter", "PROCESS"))
  {
    WriteVar("Parameter");
  }
  [...]
</MWSL>
```

If the `Parameter` process variable exists, its content is output.

If not, the instruction is skipped and the program execution is then resumed. In the example above, this would be the code that follows after the closing curly bracket.

If an `else` branch is present, it is run provided that the condition has not been fulfilled.

Example:
```
<MWSL>
  [...]
  if (ExistVariable("Parameter") && GetVar("Parameter")>=3)
  {
    write("The parameter value is:");
    WriteVar("Parameter");
    write("This is a valid value!");
  }
  else
  {
   write("Parameter process variable is not permitted or is not available");
  }
  [...]
</MWSL>
```

If the Parameter variable is not present, the `else` part is executed. A message is then output, indicating that no variable with the specified name exists.

As the examples show, an operation can be replaced by an operation block.

An operation block is a list of operations that is enclosed in curly brackets.

Example:
```
<MWSL>
  [...]
  if (ExistVariable("Parameter") && GetVar("Parameter")>=3)
  {
    WriteVar("Parameter");
  }
  else
  {
    write("Parameter process variable is not permitted or is not
available");
  }
  [...]
</MWSL>
```

If the Parameter process variable exists and the content is greater than or equal to 3, it is output. Otherwise, a corresponding output is made.

switch Condition

The `switch` operation makes it possible to compare an <expression> with many conditions <value1>, <valuet2>, etc.

```
switch (<expression>)
{
  case <value1>:
    //Program block
    break;
  case <value2>:
    //Program block
  break;
    //etc.
 default:
    //Program block
}
```

Loops

for Loop

The MWSL provides a loop mechanism, such as is already familiar from JavaScript.

For a detailed description, refer to the ECMA 262 Specification.

Syntax
```
for( Start statement; End condition; Run statement)
{
    Loop body, code to be executed
}
```

Sequence:

1. The start instruction is executed
2. The loop body is executed
3. The run operation is executed
4. As long as the end condition is true, the processing is repeated starting from the loop body (2.).

Example:

```
for(i=1; i<5; i++)
{
    write(i);
}
```

```
Output: 1234
```

`do while` Loop

In the `do while` loop, the body of the loop is first run then the while <condition> is tested.

```
do
{
 //Operation block
} while (<condition>);
```

break continue

To break or resume loop execution, two operations `break` and `continue` are available.

With the `break` operation, the loop is always exited immediately without testing the <condition> again.

The `continue` operation jumps immediately to the head of the loop.

Functions

A user-defined function consists of the function operation and a program block of operations. No or multiple parameters can be passed.

A <Wert> can be returned. If no return value is specified, the function will return `undefined`.

```
function Function name([<parameter1>,
<parameter2>, ...])
{
   //Program block
   return <Wert>;
}
```

Comments

Comments can comprise one or more lines in the MWSL.

```
// One-line comment. All characters are
ignored up to the line break.

/*
   Multi-line comment starts with '/*'
and
   must be ended with '*/'.
*/
```

Overview of MWSL functions

The MWSL provides a variety of functions, which are presented in the following overview table. For detailed descriptions of the functions, refer to the appendix.

Table 6-52 MWSL functions

Function name	Explanation
AddHTTPHeader(<Http header>) (Page 3865)	Insert <Http Header> in a page.
createGUID() (Page 3865)	Generates a unique alphanumeric ID in the system.
DecodeString(<string>) (Page 3866)	Converts a string encoded with EncodeString back to its original.
die(<Param0>,<Param1>,...) (Page 3866)	Abort program execution.
EncodeString(<string>) (Page 3866)	Replaces special characters by their URL-coded hex value (%hh).
ExistFile(<file name>) (Page 3867)	Checks whether a file with the name <parFile-Name> exists. The function returns the file length as returned value.
ExistVariable(<variable name>, <variable source>) (Page 3868)	Query of the existence of a variable.
GetLanguage() (Page 3868)	Returns the currently set language in English.
GetVar(<variable name>, <variable source>, <format string>) (Page 3869)	Return the value of a variable of the corresponding variable source
InsertFile(<text file>) (Page 3871)	Import of a <Test File>. A path can be specified.
IsAuthAlgo(<parAuthMethod>) (Page 3872)	Returns true if the user logged in at the time of calling could be identified by the authentication method specified in parAuthMethod.
isFinite(<value>) (Page 3872)	Returns false if the passed value is NaN or infinite.
isNaN(<value>) (Page 3872)	Checks whether the passed value is an invalid Double.
IsSSL() (Page 3873)	Returns true if the client is connected to the server via an SSL connection.
parseFloat(<string>) (Page 3873)	Conversion of a string to a double value.
parseInt(<value>,[<basis>]) (Page 3873)	Conversion of a string to an integer value.
ProcessXMLData(<DATA>, <TEMPLATE>) (Page 3875)	Generation of dynamic HTML files with special XML files.
string ReadFile(<file name>) (Page 3876)	Returns the content of the file as the return value.

Function name	Explanation
ReplaceString<variable name>,<search pattern>,<replacement string>) (Page 3876)	Replacement of strings matching the search pattern.
SetVar(<variable name>, <value>) (Page 3876)	Sets values of variables.
ShareRealm(<group name>) (Page 3877)	Indicates whether the current user is a member of the group that is passed as a parameter. The return value can be true or false.
write(<text>) (Page 3878)	Writes <text> strings to the HTML page. <Text> can also be the return value of functions.
WriteVar(<variable name>, <variable source>, <format string>) (Page 3879)	Output of a variable value. The syntax is identical to the GetVar() function.
WriteXMLData (<DATA>, <TEMPLATE>) (Page 3881)	Outputs the data directly in contrast to ProcessXMLData().

Table 6-53 MWSL process variables

Process variable	Explanation
NodeIndex (Page 3882)	Parse process variable for the template. This variable outputs the number of nodes that have already been run.
NodeLevel (Page 3882)	Parse process variable for the template. This variable outputs the hierarchy level of the current node.

Mode of operation of the template mechanism

A Template is applied to data elements of a data source. This mechanism allows separate implementation of data and processing.

The Template-mechanism is started by the `ProcessXMLData()` and `WriteXMLData()` commands.

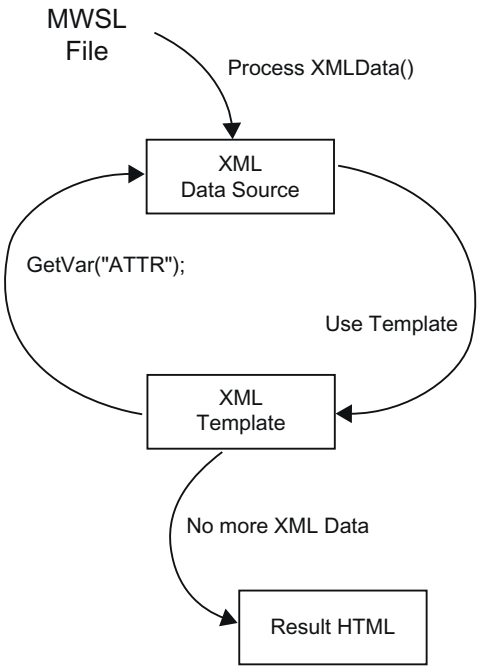


Figure 6-139 Basic overview of the template mechanism

The data file contains structured data that can be output as transformed data using the Template-mechanism.

An XML file consists of a series of XML nodes. A name as well as a set of attributes are assigned to each XML node. An attribute, in turn, consists of a name and a value.

The Template file contains a set of transformation operations.

A transformation operation can be assigned to an XML node (with a certain name).

During the transformation process, the data fragment is read node-by-node from top to bottom. If a Template can be assigned to a node, this Template is executed and the attributes of the current node are available to the Template as variables.

Structure of the template file

The Template file is an XML file whose data nodes contain the transformation operations for processing a data file.

Example

```
<?xml version="1.0" ?>
<TEMPLATES>
   [...]
    <TEMPLATE NAME="Variable">
       [...]
       <POSITION NAME="LINE">
       <![CDATA[
          [...]
          <MWSL>Name()</MWSL>
          [...]
       ]]>
        </POSITION>
       [...]
    </TEMPLATE>
    [...]
</TEMPLATES>
```

The Template file evaluates the data nodes of the data file.

The individual Template tags are defined in the <TEMPLATES> tag (in the example above, only one template tag is defined).

The line <TEMPLATE NAME="Variable"> defines that this Template (the subsequent code) is only run for data notes of the "Variable" type. The type of a data node is specified either through the name of the data node or through a special attribute named "Template".

```
<POSITION NAME="LINE">
   [...]
</POSITION>
```

This tag is used to specify the area of the web page in which the content of the template is to be output.

The positions HEAD, LINE, and FOOT are available and are run in that order during processing.

The content of the POSITION is encapsulated in a CDATA–block in order to protect it from the XML parser:

```
<![CDATA[   [...]]]>
```

MWSL can now be used to access the attributes of the current data node and to output them or use them in other operations.

Example:

```
<MWSL> WriteVar("Name")</MWSL>
```

Structure of a data source

A data source is an XML fragment, in which the nodes of the XML fragment form the data elements.

Note

If the XML fragment does not contain a root node, MWSL itself generates a root node so that the data source conforms to XML.

The XML fragment can also be a complete XML document.

Example:

```
<?xml version="1.0" standalone="yes"?>
   [...]
   <Variable Name="ZUFUEHRUNG.STATE"
             Type="String"
             InitialValue="good"
             Behavior="Manual"
             Description="Status of part infeed."
   />
   [...]
   <Variable Name="Language"
             Type="String"
             InitialValue="de_DE"
             Behavior="Manual"
             Description="Language setting of web page"
   />
   [...]
```

The example defines data nodes (in the example: `Variable`) with the associated attributes.

The different nodes can be evaluated using a corresponding template file.

A further option is the creation of a data structure (hierarchy). That is, data nodes can be created that, in turn, contain other data nodes.

Example:

```
<?xml version="1.0" standalone="yes"?>
   [...]
   <StructVariable Name="Farbe"
                   Type="String"
                   InitialValue="gelb"
                   Behavior="Manual"
                   Description="Fictitious value">
   [...]
   <Subvar Name="Rotteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="The red proportion of the color"
   />
   <Subvar Name="Blauteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="The blue proportion of the color"
   />
   <Subvar Name="Gruenteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="The green proportion of the color"
   />
   [...]
   </StructVariable>
   [...]
```

This example uses the data node of type `StructVariable`. This data node contains several data nodes of type `Subvar`.

Different templates can be used for the different types of data nodes.

Template transformation

The `ProcessXMLData()` command dynamically generates an HTML file (or just a text fragment) from an XML data file and an XML template file.

For this purpose, the parser runs through the data file step by step, from top to bottom.

The parser reads in a data node. Following this, a search is performed in the template file for a matching template for this data node. If a template is found, this template will be applied to the data node. The template is an MWSL fragment. The sole difference is that the attributes of the XML data node in the template are available as standard process variables if a variable source is not specified. If there are identical names, the XML attributes overlay the corresponding process

variables. If the variable source PROCESS is specified explicitly, the process variables are always used.

Example:
```
Prozessvariable
Color Value "Green"
```

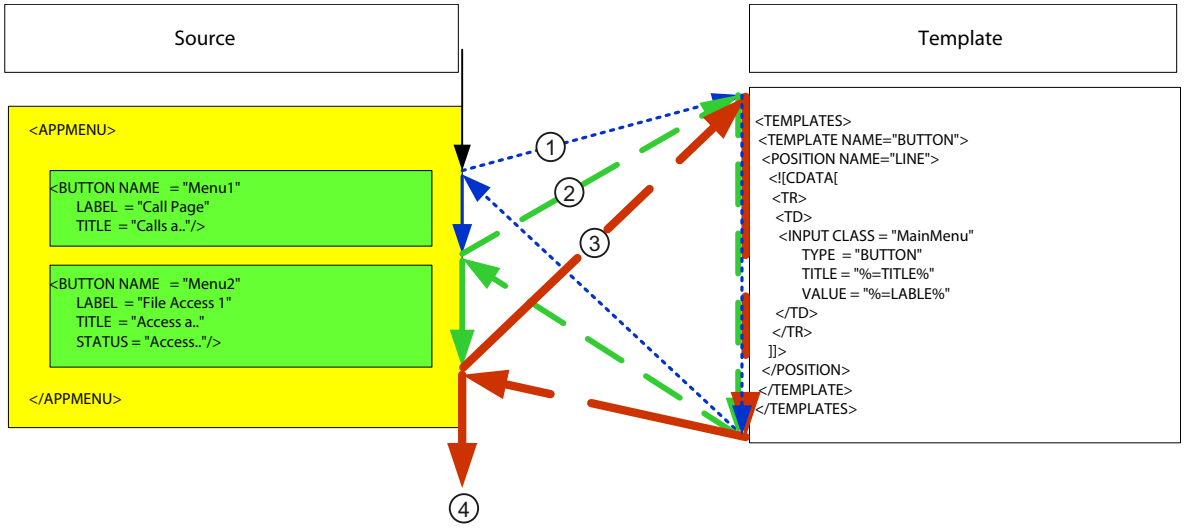
Data file:
```
<?xml version="1.0" standalone="yes"?>
[...]
<Variable Name="ZUFUEHRUNG.STATE"
          Farbe="Red"
/>
[...]
<Variable Name="Language"
/>
[...]
```

Template file:
```
<?xml version="1.0" ?>
<TEMPLATES>
   <TEMPLATE NAME="Variable">
      <POSITION NAME="LINE">
      <![CDATA[
         <MWSL>write (GetVar("Name")+":");</MWSL>
         <MWSL>write (GetVar("Color")+"\r\n");</MWSL>
      ]]>
      </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

Output:

```
ZUFUEHRUNG.STATE: Red
Language: Green
```

Sequence of a Template-process



The figure shows the sequence of the Template Parse-process.

The Template is run through once for each data element (yellow and green blocks). The attributes of the current data element can thereby be accessed as variables.

Chronological sequence

1. The <APPMENU> tag is found in the Source and the Template is run. No processing operation is found for this tag in the Template, therefore, no data element is written to the output.
2. The first <BUTTON> tag is found and processed by the Template .
3. The second <BUTTON> tag is found and processed by the Template .
4. No other tags are found; processing is complete.

Another example:

Calling MWSL file

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
  <HTML>
    <HEAD>
      <TITLE> MWSL Template Test Page</TITLE>
    </HEAD>
    <BODY>
    <MWSL>
      write(ProcessXMLData(
              "<EXTERNAL SRC=\"/FILES/MWSL/variables.xml\"/>",
                 "<TEMPLATES>
                    <EXTERNAL SRC=\"/FILES/MWSL/variablesTemplate.xml\"/>
                 </TEMPLATES>")
           );
    </MWSL>
  </BODY>
</HTML>
```

variables.xml

```
<?xml version="1.0" standalone="yes"?>

<Provider Name="SIMOTION">
  <Variable Name="dev/Service.BZU.Value"
    Type="STRING"
    InitialValue="RUN"
    Behavior="Automatic"
    Description="Aktueller Betriebszustand"
    />

  <Variable Name="var/userData.user1"
    Type="DINT"
    InitialValue="0"
    Behavior="Manual"
    Description="Variable die vom Anwender frei belegt werden kann"
    />
</Provider>
```

variablesTemplate.xml

```
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Provider">
    <POSITION NAME="HEAD">
    <![CDATA[
      <TABLE BORDER="1">
        <TR>
          <TH>
            Varname
          </TH>
          <TH>
            Actual Value
          </TH>
          <TH>
            Type
          </TH>
          <TH>
            Description
          </TH>
          <TH>
            InitialValue
          </TH>
          <TH>
            NI/NL
          </TH>
        </TR>
    ]]>
    </POSITION>

    <POSITION NAME="FOOT">
    <![CDATA[
      <TR>
        <TD COLSPAN="6">
        </TD>
      </TR>
    </TABLE>
    ]]>
    </POSITION>

    <TEMPLATE NAME="Variable">
      <POSITION NAME="LINE">
      <![CDATA[
        <TR>
          <TD align=center>
            <MWSL>WriteVar("Name")</MWSL>
          </TD>
          <TD align=center>
            <MWSL>WriteVar(GetVar("Name"))</MWSL>
          </TD>
          <TD>
            <MWSL>WriteVar("Type")</MWSL>
          </TD>
          <TD>
            <MWSL>WriteVar("Description")</MWSL>
          </TD>
```

```
          <TD align=right>
            <MWSL>WriteVar("InitialValue")</MWSL>
          </TD>
          <TD>
            <MWSL>WriteVar("NodeIndex") </MWSL> / <MWSL>
WriteVar("NodeLevel") </MWSL>
          </TD>
        </TR>
      ]]>
      </POSITION>
    </TEMPLATE>
  </TEMPLATE>
</TEMPLATES>
```

Call the `ProcessXMLData` command to start the parser with the data file variables.xml. The `<Provider>` tag is found first.

In the variablesTemplate.xml template file, a search is performed in order to ascertain whether a template has been defined for this type.

The `Position="HEAD"` area is executed.

The parser reads the next tag from the data file and generates the additional lines of the HTML file to be output based on the appropriate template in the template file.

The same happens with the next tag.

The footer part of the template provider is executed with the end tag `</Provider>`.

In the following expression, the generated parts are shown bold.

Generated file:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>

  <BODY>

    <TABLE BORDER="1">

        <TR>
          <TH>Varname</TH>
          <TH>Actual Value</TH>
          <TH>Type</TH>
          <TH>Description</TH>
          <TH>InitialValue</TH>
          <TH>NI/NL</TH>
        </TR>

        <TR>
          <TD align=center>dev/Service.BZU.Value</TD>
          <TD align=center>STOP</TD>
          <TD>STRING</TD>
          <TD>Aktueller Betriebszustand</TD>
          <TD align=right>RUN</TD>
          <TD>1 / 1</TD>
        </TR>

        <TR>
          <TD align=center>var/userData.user1</TD>
          <TD align=center>0</TD>
          <TD>DINT</TD>
          <TD>Variable die vom Anwender frei belegt werden kann</TD>
          <TD align=right>0</TD>
          <TD>3 / 1</TD>
        </TR>
        <TR>
          <TD COLSPAN="6"></TD>
        </TR>

    </TABLE>

  </BODY>
</HTML>
```

Varname	Actual Value	Type	Description	InitialValue	NI/NL
dev/Service.BZU.Value	STOP	STRING	Aktueller Betriebszustand	RUN	1 / 1
var/userData.user1	0	DINT	Variable die vom Anwender frei belegt werden kann	0	3 / 1

Figure 6-140 Generated file in the browser display

MWSL in XML attributes

As part of template parsing, it is also useful to write MWSL-statements to XML attributes of the data file, which are then evaluated at the time of parsing.

Example:

```
<?xml version="1.0" standalone="yes"?>
<Motor Name="M1"
       Nummer="1"
       Type="Dreh"
       Nennleistung = "7"
       Drehzahl="3"
       Alter="2"
       Farbe="RED"
       Prozess="&MWSL;WriteVar(&quot;CPULoad.Percent&quot;,&quot;PROCESS&qu
ot;,
                                &quot;%e&quot;); &END_MWSL;"
/>
```

The example uses the following MWSL command:

```
"&MWSL;WriteVar(&quot;CPULoad.Percent&quot;,&quot;PROCESS&quot;,&quot;%e&q
uot;);
&END_MWSL;"
```

The command would appear as follows in a template file or an MWSL file:

```
<MWSL>WriteVar("CPULoad.Percent", "PROCESS", "%e");</MWSL>
```

This command outputs the value of the `CPULoad.Percent` variables from the `PROCESS` variable source.

Please note that `<MWSL>` must be replaced with `&MWSL;`, and `</MWSL>` with `&END_MWSL;`. In addition, double quotation marks `"` must be replaced with `&quot;` .

The rest conforms to the MWSL syntax.

Examples

Examples of how MWSL can be used

The examples shown here outline the options for using MWSL.

Setting variable values using the SetVar function

```
<MWSL>
   SetVar(GetVar("VARNAME"), GetVar(GetVar("VARNAME"), "URL"));
</MWSL>
```

In this example, the variable whose name is saved in the `VARNAME` process variable is initialized with the value of the `VARNAME` URL variable.

For illustration

`GetVar("VARNAME")` supplies the content of the `VARNAME` process variable. This value will be viewed, in turn, as a variable name.

If we assume that the content of the `VARNAME` process variable is "Jack", the complete call already looks much simpler:

Complete call: `SetVar("Jack", GetVar("Jack", "URL"));`

If we were to assume that the URL variable "Jack" had the content "is a great guy", this would be equivalent to the following expression:

```
SetVar("Jack", "is a great guy");
```

TestTemplate.mwsl

This example will be used to briefly explain the template mechanism again.

For illustrative purposes, a few passages in the respective files are marked.

The template produces a table.

The called file

TestTemplate.mwsl

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <TITLE> MWSL Template Test Page </TITLE>
  </HEAD>
  <BODY>
    <MWSL>
      write(ProcessXMLData(
              "<EXTERNAL SRC=\"/FILES/MWSL/variables.xml\"/>",
             "<TEMPLATES> <EXTERNAL SRC=\"/FILES/MWSL/variablesTemplate.xml
\"/>
               </TEMPLATES>")
           );
    </MWSL>
  </BODY>
</HTML>
```

VariablesTemplate.xml

```
<?xml version="1.0" ?>
<TEMPLATES>
 <TEMPLATE NAME="Provider">
  <POSITION NAME="HEAD">
  <![CDATA[
   <TABLE BORDER="1">
    <TR>
     <TH>Varname</TH>
     <TH>Actual Value</TH>
     <TH>Type</TH>
      <TH>Description</TH>
     <TH>InitialValue</TH>
     <TH>NI/NL</TH>
    </TR>
  ]]>
  </POSITION>

  <POSITION NAME="FOOT">
  <![CDATA[
    <TR>
     <TD COLSPAN="6"> </TD>
    </TR>
   </TABLE>
  ]]>
  </POSITION>

  <TEMPLATE NAME="Variable">
   <POSITION NAME="LINE">
   <![CDATA[
    <TR>
     <TD align=center><MWSL>WriteVar("Name")</MWSL></TD>
     <TD align=center> <MWSL>WriteVar(GetVar("Name"))</MWSL> </TD>
     <TD><MWSL>WriteVar("Type")</MWSL></TD>
     <TD><MWSL>WriteVar("Description")</MWSL></TD>
     <TD align=right><MWSL>WriteVar("InitialValue")</MWSL></TD>
     <TD><MWSL>WriteVar("NodeIndex") </MWSL> / <MWSL> WriteVar("NodeLevel") </MWSL></TD>
    </TR>
   ]]>
   </POSITION>
  </TEMPLATE>
 </TEMPLATE>
</TEMPLATES>
```

In this case, the template file consists of multiple templates. The `Provider` template provides the header and footer parts for the data (table header and footer).

The `Variable` template is appointed for the lines of data output (one table row per variable entry).

The data is inserted after the corresponding attributes.

The data file `variables.xml`

```
<?xml version="1.0" standalone="yes"?>
<Provider Name="SIMOTION">
  <Variable Name="dev/Service.BZU.Value"
    Type="STRING"
    InitialValue="INIT"
    Behavior="Automatic"
    Description="Current operating state"
  />

  <Variable Name="var/userData.user1"
    Type="DINT"
    InitialValue="0"
    Behavior="Manual"
    Description="Variable that can be assigned freely by the user"
  />
</Provider>
```

Generated HTML file

```
<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER="1">
      <TR>
        <TH>
          Varname
        </TH>
        <TH>
          Type
        </TH>
        <TH>
          Description
        </TH>
        <TH>
          Value
        </TH>
        <TH>
          NI/NL
        </TH>
      </TR>
      <TR>
        <TD align=center>
          <A HREF="">ZUFUEHRUNG.STATE</A>
        </TD>
        <TD>
          String
        </TD>
        <TD>
          Status of part infeed.
        </TD>
        <TD align=right>
          good
        </TD>
        <TD>
          2 / 2
        </TD>
      </TR>
      <TR>
```

```
        <TD align=center>
          <A HREF="">Language</A>
        </TD>
        <TD>
          String
        </TD>
        <TD>
          Language setting of the Web page
        </TD>
        <TD align=right>
          de_DE
        </TD>
        <TD>
          3 / 2
        </TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

MainNavigation.mwsl

```
<html>
   <head>
      <title>
         MiniWeb Main Navigation
      </title>
   </head>
   [...]
   <body>
      <table>
         <tr>
         <MWSL>
             write(ProcessXMLData(
               "<EXTERNAL SRC=\"/XML/MainNavigation.xml\"/>",
               "<TEMPLATES><EXTERNAL
                      SRC=\"/Templates/MainNavigation.xml\"/>
               </TEMPLATES>") );
         </MWSL>
         </tr>
      </table>
   </body>
</html>
```

This file contains the actual body of the HTML page to be generated.

The dynamic part is generated with the `ProcessXMLData()` command and inserted in `<table>`.

/XML/MainNavigation.xml

MainNavigation.xml contains the data part for the generation.

```
<?xml version="1.0" standalone="yes"?>
<MAINNAVIGATION>
   <APPLICATION NAME = "Entrance"
                CLIENTAREA = "/Portal/Entrance.mwsl"
                TITLE = "Back to Entrance Page." />
   <APPLICATION NAME = "MWSL Test"
                CLIENTAREA = "/MWSL/Start.mwsl"
                TITLE = "Test environment for MWSL."/>
   <APPLICATION NAME = "File Browser"
                       REALM = "Administrator"
                       CLIENTAREA = "/www"
                       TITLE = "Browse the Filesystem" />
   [...]
   <APPLICATION NAME = "CSSA"
                REALM = "User"
                CLIENTAREA = "/CSSA/Main.mwsl"
                TITLE = "PKI Interface."/>
                <APPLICATION NAME = "VarSimulator"
                CLIENTAREA = "/Simulator/Simulator_index.mwsl"
                TITLE = "Simulate several variables."/>
</MAINNAVIGATION>
```

/Templates/MainNavigation.xml

```
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
   <TEMPLATE NAME="APPLICATION">
   <POSITION NAME="LINE">
   <![CDATA[
      <td>
         <input class = "MainMenu"
                type = "BUTTON"
                title = "<MWSL> WriteVar("TITLE")</MWSL>"
                value = "<MWSL> WriteVar("NAME")</MWSL>"
                OnClick =
                 "NavigateApp('<MWSL>WriteVar("CLIENTAREA")</MWSL>')"
         />
      </td>
   ]]>
   </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

This file is run for each data node and the appropriate variables are inserted.

Generated HTML file

```
<html>
   <head>
      <title>
         MiniWeb Main Navigation
      </title>
   </head>
   <body>
      <table>
          <tr>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Back to Entrance Page."
                       value = "Entrance"
                       OnClick = "NavigateApp('/Portal/Entrance.mwsl')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Test environment for MWSL."
                       value = "MWSL Test"
                       OnClick = "NavigateApp('/MWSL/Start.mwsl')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Browse the Filesystem"
                       value = "File Browser"
                       OnClick = "NavigateApp('/www')"
                />
             </td>
             [...]
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "PKI Interface."
                       value = "CSSA"
                       OnClick = "NavigateApp('/CSSA/Main.mwsl')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Simulate several variables."
                       value = "VarSimulator"
                       OnClick = "NavigateApp
                                   ('/Simulator/Simulator_index.mwsl')"
                />
             </td>
          </tr>
      </table>
   </body>
</html>
```

AppNavigation.mwsl

Call:

```
<MWSL>
   WriteXMLData("<EXTERNAL SRC=\"" + GetVar("XML", "URL") + "\"/>",
                  "<TEMPLATES><EXTERNAL SRC=\"" +
                  GetVar("TEMPLATE", "URL") + "\"/></TEMPLATES>");
</MWSL>
```

When the call is made, the data and the template file are transferred from the URL.

In this example, the transferred template file is assumed to be AppNavigation.xml and the transferred data file MWSLTestMenu.xml.

MWSLTestMenu.xml

```
<?xml version="1.0" ?>
<APPMENU>
   <MENU>
      <BUTTON NAME = "Menu1"
              LABEL = "Variablentest"
              TITLE = "Tooltip"
              STATUS = "Statusline"
              CLIENTAREA = "/MWSL/Variablentest.mwsl?Parameter=4711"
      />
      [...]
      <BUTTON NAME = "Menu9"
              LABEL = "Versuch"
              TITLE = "Tooltip"
              STATUS = "Statusline"
              CLIENTAREA = "/MWSL/Versuch.mwsl" />
   </MENU>
</APPMENU>
```

In the following template file, some attributes are queried as to their existence and the corresponding lines executed.

In this case, only the condition for `CLIENTAREA` is run since no other queried attribute is present.

AppNavigation.xml

```
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
   <TEMPLATE NAME="BUTTON">
      <POSITION NAME="LINE">
      <![CDATA[
         <TR>
            <TD>
               <INPUT CLASS = "MainMenu"
                      TYPE = "BUTTON"
                      TITLE = "<MWSL>TITLE()</MWSL>"
                      VALUE = "<MWSL>LABEL()</MWSL>"
                  <MWSL>
                  if ( ExistVariable( "CLIENTAREA" ))
                  {
                     write("OnClick=
                       \"top.ClientArea.window.navigate('"+
                           GetVar("CLIENTAREA") + "')\"");
                   }
                   if ( ExistVariable ( "TOP" ))
                   {
                       write("OnClick = \"top.window.navigate('" +
                        GetVar("TOP") + "')\"");
                    }
                    if ( ExistVariable ( "WIN" ))
                    {
                        write("OnClick = \"window.open('" +
                                 GetVar("WIN") + "')\"");
                    }
                    if ( ExistVariable ( "ACTION" ))
                    {
                       write("OnClick = \"top.ClientArea.window." +
                         GetVar("ACTION") + "\"");
                    }
                    </MWSL>
                 />
              </TD>
          </TR>
      ]]>
      </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

The template file contains `if` conditions in which attributes from the data file are queried. This results in different statements for each data node during runtime.

The generated HTML file will then only contain the line corresponding to the respective data node.

generierte Ausgabe

```
<TR>
   <TD>
      <INPUT CLASS = "MainMenu"
             TYPE = "BUTTON"
             TITLE = "Tooltip"
             VALUE = "Variablentest"
             OnClick = "top.ClientArea.window.navigate(
                         '/MWSL/Variablentest.mwsl?Parameter=4711')"
      />
   </TD>
</TR>
[...]
<TR>
   <TD>
      <INPUT CLASS = "MainMenu"
             TYPE = "BUTTON"
             TITLE = "Tooltip"
             VALUE = "Versuch"
             OnClick =
               "top.ClientArea.window.navigate('/MWSL/Versuch.mwsl')"
      />
   </TD>
</TR>
```

Server Side Includes (SSI)

Integration of process values

You can include process values in the user-defined HTML pages using Server Side Includes (SSI).

Note

In SIMOTION controller V4.1 and higher, HTML pages with SSI must be available as a binary file to display process values. A standard HTML page can be converted to a binary file using the supplied conversion tool (Page 3746).

HTML pages with static content only do not have to be converted.

Integration of process values

The variables are integrated in the HTML page using the `<%=IDENTIFIER %>` character string. `IDENTIFIER` is a placeholder, which you must replace with variables from the variable providers. For example, the variable `<%=DeviceInfo.Board%>` returns the name of the controller. On a D435, for example, the value is "D435".

Details of the variables and syntax can be found in the *Variable providers* chapter of the *SIMOTION IT Diagnostics and Configuration* manual.

The source text below shows an example for integrating the variable `userData.user1`. The value of the variable is first output (system variable `userData.user1`: `<%=var/`

userData.user1 %>). The value of the variable is used as a default in the input field and can be overwritten by a user input.

Up to V5.2

```
<html>
   <head>
      <title>Demo Seite</title>
   </head>
   <body text="#000000" bgcolor="#FFFFFF" link="#FF0000"
   alink="#FF0000" vlink="#FF0000">
      Demoseite<br>
      Systemvariable userData.user1 : <%= var/userData.user1 %> <br>
      <form method="post" action="/VarApp">
            SIMOTION C: userData.user1:
            <input type="TEXT" name="var/userData.user1" value="<
            %= var/userData.user1 %>" />
            <input type="submit" value="Wert schreiben" />
      </form>
   </body>
</html>
```

As of V5.2

```
<html>
    <head>
        <title>Demo Seite</title>
    </head>
    <body text="#000000" bgcolor="#FFFFFF" link="#FF0000" alink="#FF0000"
    vlink="#FF0000">
        Demoseite<br>
        Systemvariable userData.user1 : <%= var/userData.user1 %> <br>
        <form method="post" action="/VarApp">
                SIMOTION C: userData.user1:
                <!-- CSRF Token input _must_ be at 1. place in form !!! -->
                <input type="hidden" name="MultiUseToken" value="<
                %=RawMultiUseToken%>"/>
                <input type="TEXT" name="var/userData.user1" value="<%= var/
                userData.user1 %>" />
                <input type="submit" value="Wert schreiben" />
        </form>
    </body>
</html>
```

As of V5.2, the `MultiUseToken` attribute is required. This attribute provides Cross Site Request Forgery protection. Further information can be found in the *CSRF protection* chapter in the *SIMOTION IT Diagnostics and Configuration* manual.

The value of the `MultiUseToken` attribute is set by the Server Side Include `<%=RawMultiUseToken%>` by the Web server.

6.2.3.2 OPC XML-DA web service

Web services introduction

A SIMOTION IT web service supports application programs in text-based access to process values. The application program performs strictly symbolic access independently of the programming language, the operating system, and the time of programming. Details of the international standard of web services can be researched on the Internet or in technical literature.

The SIMOTION control offers the OPC XML-DA web service for data access in accordance with the OPC XML-DA definition V1.01 of the OPC Foundation and the Trace Via SOAP web service (TVS) for the use of the SIMOTION Runtime Trace.

- Addon\4_Accessories\SIMOTION_IT\7_Webservices\WSDL\OPC XMLDA 1.01.wsdl
- Addon\4_Accessories\SIMOTION_IT\7_Webservices\WSDL\TVS.wsdl

An example of a client using the OPC XML-DA service is supplied.

- Addon\4_Accessories\SIMOTION_IT\7_Webservices\Example

Web services can also be used with JavaScript in HTML pages. The standard diagnostics pages of SIMOTION IT use these. See Variable access with JavaScript and web services (Page 3755)

Required knowledge for programming: OPC XML-DA, web services, XML, SOAP, Javascript/AJAX.

Key words for advanced programming: Structuring of applications with model, view, controller architecture, if, for example, dynamic variable values are to be displayed and updated in the background.

Overview

The SIMOTION IT OPC XML DA server enables access via Ethernet to data and operating modes of the SIMOTION device.

What is OPC XML DA?

OPC stands for Open Connectivity and denotes a standard interface for communication in automation technology

With OPC XML DA, it is possible to communicate with a controller using Ethernet-based standard telegrams.

Commands are transmitted via the SOAP (Simple Object Access Protocol) communication protocol.

The interface is defined in a configuration file using a description language (WSDL) based on XML vocabulary. It describes the format of the HTTP request and response telegrams with which

function calls are output (see OPC XML-DA R1.0 Specification: OPC Foundation Download (http://www.opcfoundation.org/Downloads.aspx)).

This interface can only be used by client applications.

The figure below shows an example client of the OPC Foundation . The client enables browsing via the system, interface, IO, and global device variables.

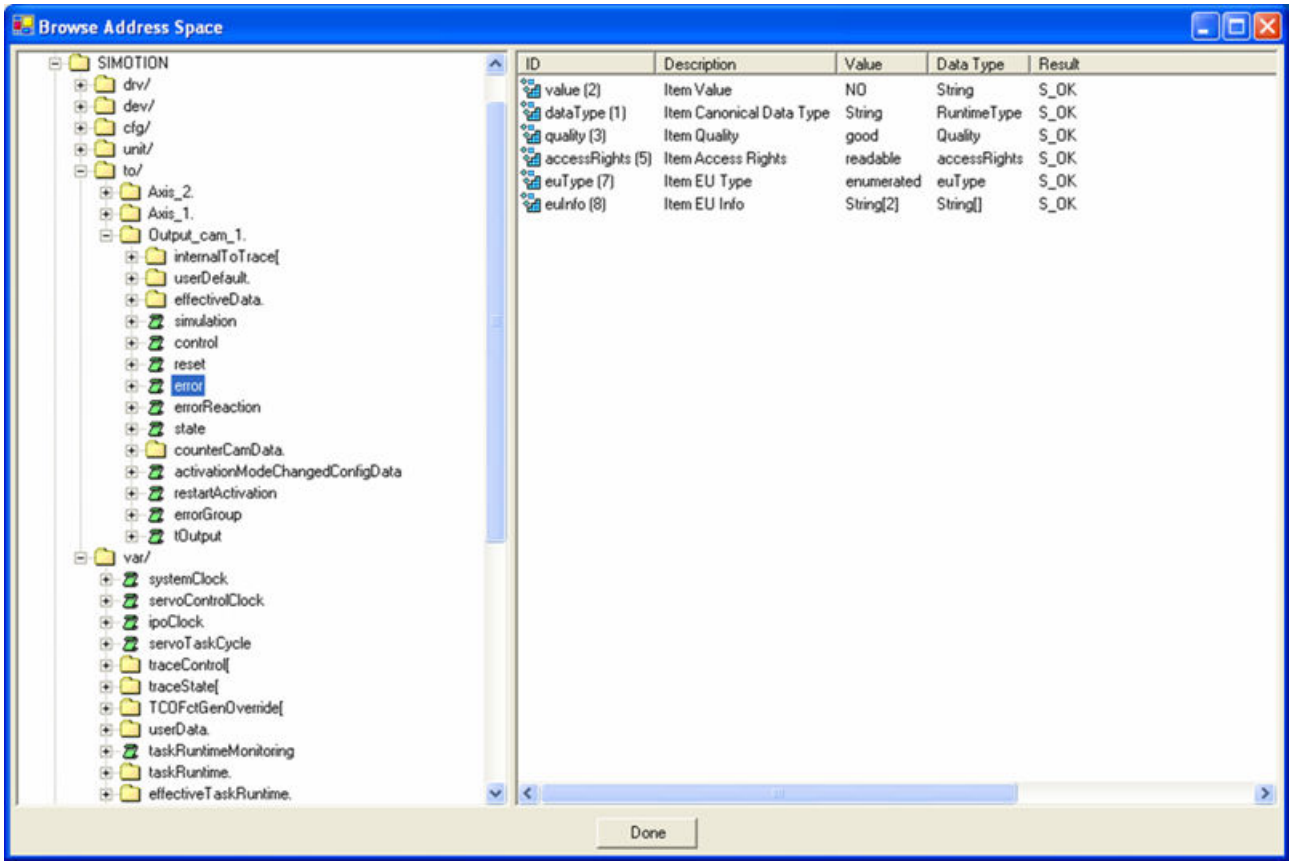


Figure 6-141 OPC Foundation Client

Purpose and benefits

The purpose and benefits of SIMOTION IT OPC XML-DA server are as follows:

- Symbolic access (without project information) to the data of the controller. Only knowledge of the variable names is required.
- Non-dependence on the engineering and project versions. The client can still access the data even if the control program has been modified.
- The server can be addressed by any client application which conforms to the OPC XML-DA V1.0 standard, regardless of its operating system (e.g. Linux).

What previous knowledge is required?

For the user to understand the SIMOTION IT OPC XML-DA server described in this chapter, prior knowledge of the terms associated with OPC XML-DA (see OPC XML-DA R1.0 Specification) is necessary.

Comparison of OPC XML DA / SIMATIC NET OPC DA

Comparison

The "SIMATIC NET OPC Server for SIMOTION" product exists in addition to the SIMOTION IT OPC XML DA server. This package also allows access to data and operating modes of the SIMOTION device via SIMATIC NET OPC DA.

The following table compares the two packages and describes the basic procedure:

Table 6-54 Basic procedure for accessing data

SIMOTION IT OPC XML DA	SIMATIC NET OPC DA
No configuration (OPC export) necessary with SIMOTION SCOUT.	OPC export with SIMOTION SCOUT required, which has to be repeated for every project change.
Symbols are resolved in the SIMOTION device, communication by means of text format (XML).	Symbols are resolved during OPC export and stored in the OPC server on the Windows system in binary format; binary communication -> higher data throughput.
At present only SIMOTION with OPC XML DA. Access to S7 devices not possible at present.	Simultaneous access to SIMOTION and S7 devices.
Client can run on any operating system.	Based on Windows COM/DCOM technology; client and server can only run on Windows operating systems.
Communication using standard protocols (TCP/IP, XML, SOAP) does not require vendor-specific (SIEMENS) tools or drivers on the client system.	S7 protocol used for communication, appropriate manufacturer-specific drivers required on the client.
Communication is only possible via Ethernet (e.g. PROFINET).	Communication is possible via PROFIBUS/MPI and Ethernet (e.g. PROFINET).
Direct addressing via firewalls is possible.	Generally, DCOM communication is not released for firewalls.

Schematic representation of creating the client application

Example arrangement

The figure below shows an arrangement example of the relevant software for the creation of a client application on a PC. The PC and the SIMOTION device are networked via Ethernet.

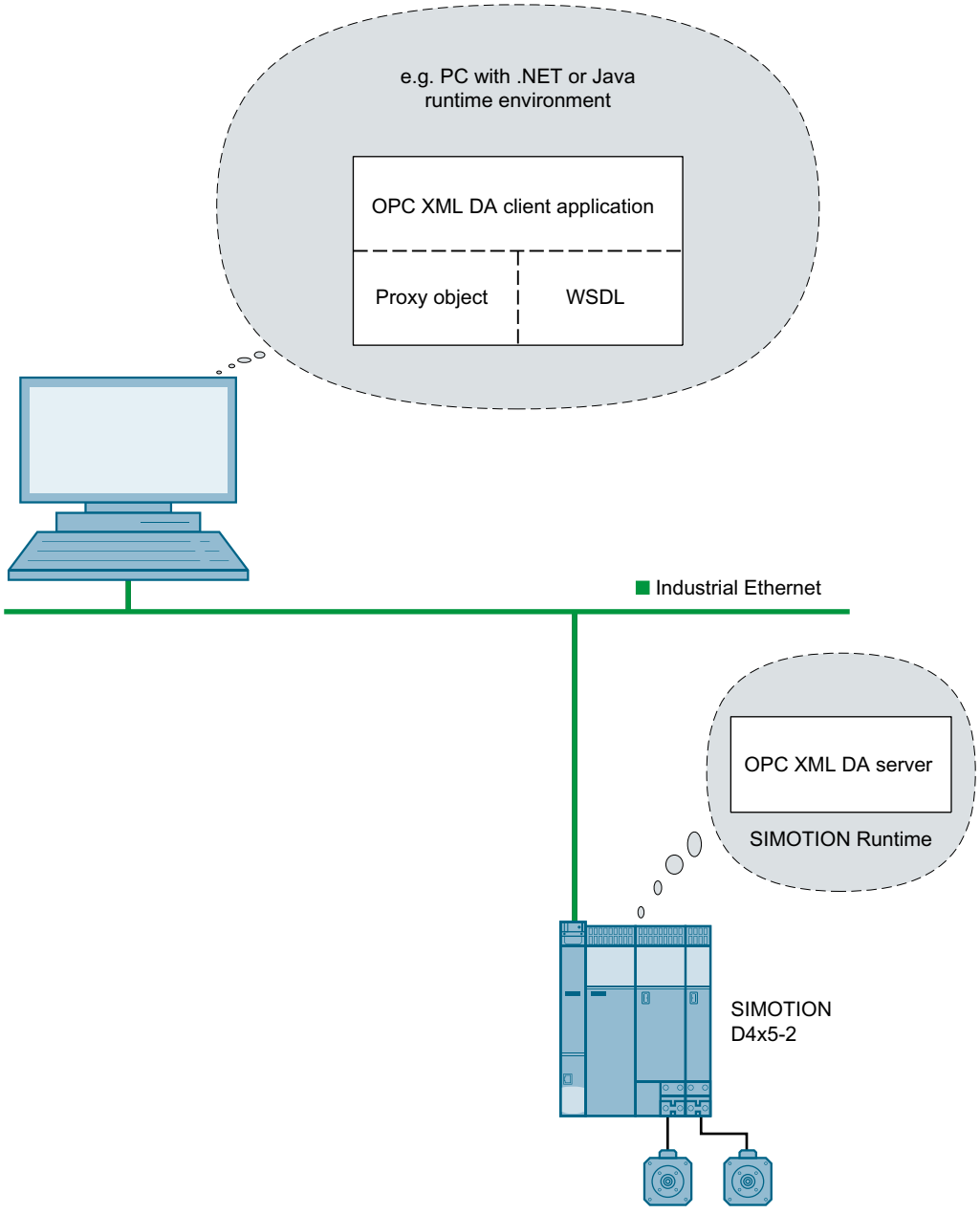


Figure 6-142 Design stage overview (example)

Schematic representation at runtime of the client application

Example arrangement

The figure below shows an example of accessing the OPC XML-DA server of a SIMOTION device via Ethernet during runtime. The example shows other devices connected to the SIMOTION device via PROFIBUS.

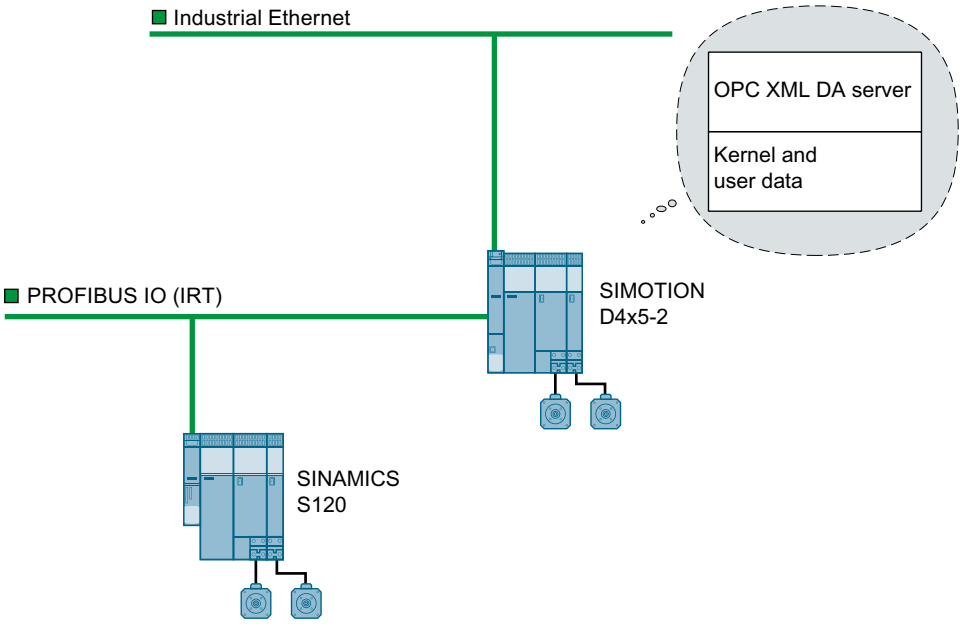


Figure 6-143 Overview at runtime (example)

Installation

Hardware and software requirements for creating the client application

Hardware requirements at the design stage

Note

You can freely select the programming environment. The following requirements are examples of Microsoft Visual Studio .NET, but are not mandatory.

Table 6-55 Hardware requirements at the design stage

Feature	Minimum requirement
Processor	Intel Pentium III or compatible, 800 MHz
Main memory	128 MB RAM

Software requirements at the design stage

Note

You can freely select the programming environment. The following requirements are an example for Microsoft Visual Studio .NET, but they are not binding.

- Microsoft Visual Studio .NET:
  http://msdn.microsoft.com/vstudio/ (http://msdn.microsoft.com/en-us)
  http://www.microsoft.com/net/ (http://www.microsoft.com/net/)
- Configuration file (WSDL), according to OPC XML- R1.0 Specification.

Configuring the SIMOTION device interface for using the client application

Configuring the interface

In order to establish a connection between a PC and a SIMOTION device when the system is running, you must carry out the following steps for the configuration of the Ethernet interface:

Table 6-56 Configuring the interface

Step	Procedure
1	You must activate the functionality in the SIMOTION project when configuring the control hardware via the "Ethernet Extended / Webserver" properties in the "Simotion IT - Web server settings" function.
2	The IP address of the SIMOTION control via which the OPC XML-DA server is accessed must be known. The IP address can be configured using the interface settings in HW Config.

OPC XML DA access protection

Settings for the OPC XML-DA access protection

WebCfg.xml provides a way of defining the access protection for OPC XML-DA via a `REALM`.

```
<CONFIGURATION_DATA>
...
  <SOAPAPP>
    <STATIC>
      <!--
        place all statically linked WebServices here
      -->
      <WEBSERVICE NAME="OpcXml" URL="/SOAP/OPCXML"
                  REALM="OPC_XML_USER_GROUP" />
      <WEBSERVICE NAME="TVS" URL="/SOAP/TVS" />
```

```
    </STATIC>
  </SOAPAPP>
...
</CONFIGURATION_DATA>
```

Authentication for read and write accesses

WebCfg.xml also allows read and write accesses to the OPC-XML server to be enabled or disabled depending on the logged in user.

The `<OpcXml>` entry in WebCfg.xml controls the associated access via a `REALM`.

```
<CONFIGURATION_DATA>
  <OPCBASESERVICE>
    <OpcXml READ="<ReadRealm>" WRITE="<WriteRealm>" />
    <OpcXmlWeb READ="<ReadRealm>" WRITE="<WriteRealm>" />
  </OPCBASESERVICE>
</CONFIGURATION_DATA>
```

OPC XML-DA variable access

Access to the OPC XML-DA data uses the same access syntax as that described in the Variable providers section of the SIMOTION IT Diagnostics and Configuration Manual.

Table 6-57 Examples of accessing variables

Variables	Access syntax
Global device variables	glob/<var name>
I/O variables	io/_direct.<var name> io/_image.<var name> io/_quality.<var name>
Unit (MCC/ST/LAD-FBD/DCC)	unit/<unit name>.<var name>

Access to the I/O variables (access syntax)

"_direct" addresses the direct I/O access (current value) to the I/O variables.

"_image": addresses the process image of the I/O variables

"_quality": addresses the Quality, i.e. the detailed status of the I/O variables

Example of a client application

Example

An example showing what a minimal client application for an OPC client can look like is included on the AddOn-DVD in the \VOL2\Addon\4_Accessories\SIMOTION_IT\7_Webservices\Example directory. The example explains the most important programming steps for the "Read" method with the Microsoft Visual Studio development environment.

The application example displays a **Read** button in a dialog box. When the button is pressed, the client connects to the SIMOTION IT OPC XML DA server and reads a variable. The result is displayed in the output field of the dialog box.

The dialog box of the application example is shown in the following figure:

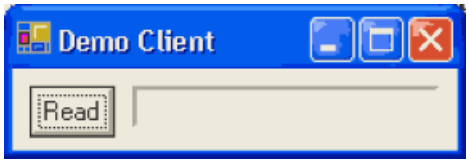


Figure 6-144 Demo client

Programming steps

The following programming steps are needed:

1. Create a new project with Microsoft Visual Studio .NET and import the WSDL file as interface description ("Add Web Reference" menu).
2. Create a dialog box with a text field and a **Read** button.
3. Enter the name assigned for the reference, e.g. "OPCXMLServer", in the program (`using DemoClient.OPCXMLServer`).
4. Declare the server URL in the program as follows:
   http://<IP-Adresse>/soap/opcxml
   Enter the IP address of your SIMOTION controller at <IP address>.
5. Instantiate the server proxy object in accordance with the code example and provide the call-up with the required parameters.
6. The required data is returned.

SIMOTION IT OPC XML–DA server interface

Overview

Introduction

This section describes the methods you can run across the interface to the OPC XML-DA V1.0 server. The server itself is integrated in the SIMOTION device.

For firmware version V4.2 and higher, a license is no longer required for using the OPC XML-DA server.

This is just a brief overview. These methods are described in detail in the document "OPC XML-DA Specification R1.0" of OPC Foundation.

You can find an up-to-date and detailed interface description on the home page of the OPC Foundation: http://www.opcfoundation.org (http://www.opcfoundation.org)

Methods which can be called synchronously

The SIMOTION IT OPC XML DA server provides the following methods, which can be called synchronously, under the "OpcXmlDaService" type:

Description of methods

Browse

The "Browse" method allows you to navigate through the available variables.

GetProperties

The "GetProperties" method can query the settings for a specific variable (e.g. access rights, time stamp, data type).

GetStatus

The "GetStatus" method supplies information about the server status, the program version and the supported interface version.

Read

The "Read" method reads out variable lists.

Subscribe

The "Subscribe" method passes a list of variable names and receives a handle for the subscription. This handle can be used in the SubscriptionPolledRefresh method to poll the values of the previously defined variables again. See Basics of subscriptions (Page 3853).

The buffering property, which corresponds to the "EnableBuffering" attribute, is not supported. This has an effect on the "SubscribeRequestItem" and "SubscribeRequestItemList" methods.

SubscriptionPolledRefresh

The "SubscriptionPolledRefresh" method returns the values of the variables written beforehand using the Subscribe method. The handle specifying the subscription is used as a parameter.

The "Holdtime" parameter defines the earliest possible response time. This limits the frequency of data transmission.

The "ReturnAllterms" parameter determines how the "WaitTime" parameter is used.

- True
  "WaitTime" is ignored, all requested values are returned immediately.
- False
  For the period set in the "WaitTime" parameter, the server checks whether one of the requested values has changed since the last call.
  If the specified time expires without a value having been changed, an empty response is returned.
  If values change during the specified time, the changed values are returned immediately and the polling ended.

SubscriptionCancel

The "SubscriptionCancel" method cancels the subscription and returns the subscription handle.

Which subscription is to be canceled, must be specified at the call.

If an asynchronous call form is used, the client is informed later of which subscription has been canceled, via a client handle.

Note

Once the subscription has been canceled, the subscription handle ceases to be valid for the client.

Write

The "Write" method writes variable lists.

Access to variables

Variable access using methods

Variables can be accessed via the methods which can be called synchronously and asynchronously.

Note

Information on accessing the variables of the different providers can be found in the *Variable providers* chapter of the *SIMOTION IT Diagnostics and Configuration* manual.

To make variables available on the SIMOTION IT OPC XML DA server, you have to declare them as VAR_GLOBAL. See the *Making unit variables available* chapter of the *SIMOTION IT Diagnostics and Configuration* manual.

6.2.3.3 Trace Interface via SOAP (TVS) web service

Trace overview

Introduction

The SOAP-based service provides a trace service option.

WebTrace uses the same runtime mechanisms as SIMOTION SCOUT Trace. The useful options are described in the chapters *Trace (device trace)* and *Trace (system trace)* of the *SIMOTION IT Diagnostics and Configuration* manual.

Trace-Service

The "Trace Interface via SOAP" web service enables variable values to be written to a buffer. The values are packed in files and can be retrieved asynchronously via an HTTP request.

This interface can only be used by client applications. The client enables the time characteristic of variables to be traced.

A WSDL file is available for creating the application.

Trace sequence

Introduction

When working with a trace, the trace can assume various states. The following graphic shows the possible states and transitions. The methods named are described in chapter "Trace interface".

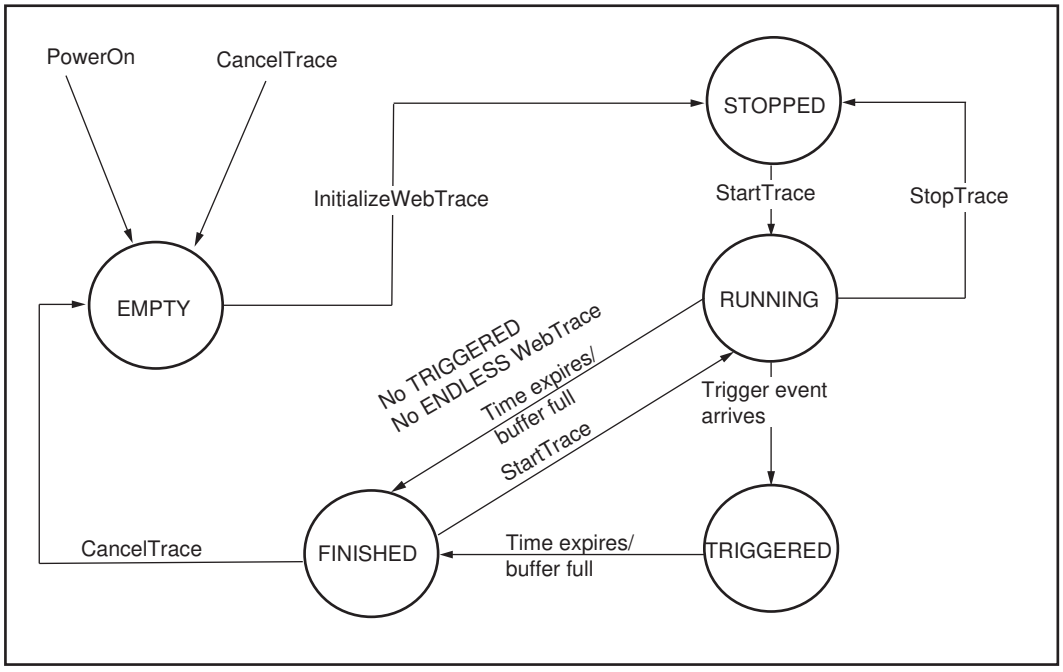


Figure 6-145 WebTrace

States

After a trace has been created with "InitializeWebTrace", this trace is in the STOPPED state. With "StartTrace", a trace starts up and writes the desired data to the buffer. Accordingly, a trace can be stopped again with "StopTrace". After the start, a trace switches to the RUNNING state. If the time specified in the call has expired, a trace assumes the FINISHED state. A trace can be deleted at any time with "CancelTrace" in order to create a new trace, for example.

Procedure/terms

HTTP methods - data exchange

The trace data are stored as compiled data on the RAM disk using the ReadData method. This data must be retrieved via ordinary HTTP requests.

Note

The data are not deleted by retrieval alone! To prevent the RAM disk from overflowing, an HTTP DELETE call to this URL must follow an HTTP GET call. (Reason: The use case under consideration is one in which a client may have to request the same trace data more than once, e.g. to compare traces that have already been executed.) These temporary data are completely deleted only after a CancelTrace operation, regardless of whether they have already been retrieved or not.

TRIGGERED

The trace offers a triggering option. Depending on the trigger method, different constants or variable symbols must be specified for this. The trace starts with:

- A rising edge (RE),
  if the variable exceeds the value of a constant.
- A falling edge (FE),
  if the variable falls below the value of a constant.
- Within a tolerance band (WIB),
  if the variable lies between two constants.
- Outside of a tolerance band (OOB),
  if the variable lies outside of a tolerance band.
- Bit mask has value (BHV),
  if the variable has a specified value after masking with a constant.

If the trace is set up in TRIGGERED mode, a trigger condition as described below must be specified. This trigger acts as a SingleShot. However, the MatchCountTriggerPoint parameter can be used to set the trigger for repeated occurrences (e.g. five: start trace/data recording only on the fifth time appearance of the trigger condition).

In this case, the trace takes place only after the trigger. The data are recorded for the duration specified during setup.

IMMEDIATE / ENDLESS

The counterpart to the TRIGGERED trace is the IMMEDIATE trace, which begins the trace immediately after the "StartTrace" call has occurred. In this case as well, the data are recorded for the duration specified during setup.

The ENDLESS Trace uses a ring buffer trace. Trigger conditions are not evaluated. ENDLESS Trace starts as soon as the StartTrace event arrives. However, it is terminated only when StopTrace is called explicitly. The size of the ring buffer must also be specified using the duration for the initialization call. Thus, an appropriate value must be found that uses fewer resources, but is sufficient to retrieve data in a timely manner via HTTP.

The size of the ring buffer (B) is determined from the number of variables (N), their size (S), the transferred time duration (t) and the cycle clock (T) in which they are traced.

$$B = t/T * \sum_{i=0}^{n-1} S^i$$

Within the transferred time duration, the buffer must be discharged at least once by calling the "readData" function in order to prevent the oldest trace data from being overwritten each time.

If the parameters require a larger memory area, the recording duration is reduced such that no more than the available memory space is occupied.

512 KB is available for SIMOTION C, SIMOTION D410-2, and 1024 KB is available as ring buffer for all other SIMOTION modules.

Error handling

All implemented methods of the TVS (trace via SOAP) supply either the requested data or status information, or an SOAP_FAULT. This behavior enables the use of the SoapFaultError in the .NET framework. The Try-Catch mechanism enables convenient error handling.

Basics of subscriptions

Introduction

"GetStatus" must be called in order to query the status of a trace. The fastest possible detection of a status change requires extremely frequent polling, which places an unnecessary load on the CPU in the control and causes heavy traffic on the network.

To optimize this operation, OPC XML DA provides "subscriptions". With subscriptions, a query does not receive a response until the required variable changes or a timeout occurs. Thus, the connection is kept open without causing traffic. As soon as relevant data are available for the client, these data are sent to the client.

The trace supports this mechanism via the SOAP web service also. However, in this case, only the status of the trace object is checked, as this is the only valuable information in this environment.

As soon as the status changes (e.g. RUNNING -> FINISHED), the clients that issued the query receive a response accordingly. In essence, any number of clients is possible (as long as there are sufficient resources).

Operational sequence

The operational sequence of a subscription is as follows:

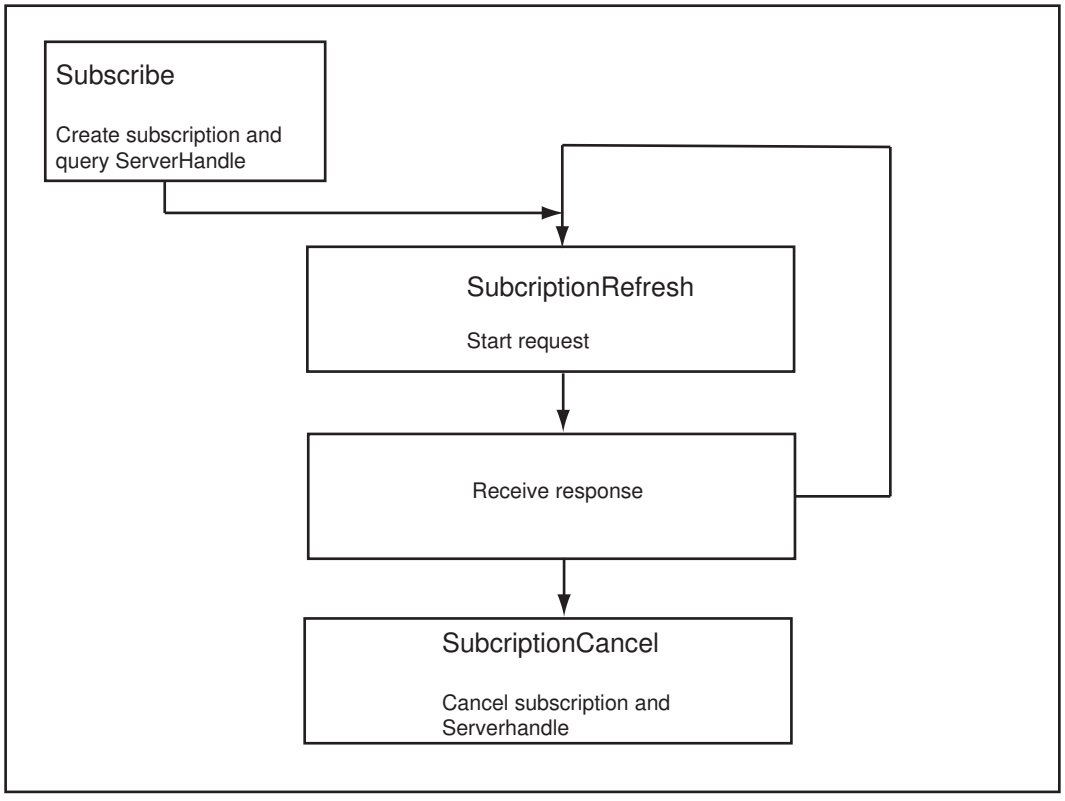


Figure 6-146 Subscription

First, a subscription must be created. It is answered with a unique ServerHandle, which is required for further communication.

Note

Up to 8 subscriptions per server are available

- With larger data requests, reduce the number of subscriptions and request more variables in fewer subscriptions.

SubscriptionRefresh can be called as often as necessary to start a new query. This request receives two time specifications in milliseconds as parameters:

- HoldTime:
  This time indicates the minimum hold time for the response, irrespective of whether the status has changed.
- WaitTime:
  The WaitTime begins after the HoldTime has expired. If the trace status has changed, the response to the current status is sent immediately. If there is no change, the response is sent once the WaitTime has expired.

The exact method calls are explained in the next section.

Interface

Global definitions

TraceStateEnum

Enumerator that indicates the status of the trace object.

Declaration:

```
public enum TraceStateEnum
{
  RUNNING,
  STOPPED,
  ERROR,
  EMPTY,
  FINISHED,
  TRIGGERED
}
```

TraceDataCycleEnum

Enumerator that specifies the cycle clock in which the data are to be traced. It must be noted here that large traces may cause a level overflow.

Declaration:

```
public enum TraceDataCycleEnum
{
  IPO1,
  IPO2,
  SERVO
}
```

Structure VDSC

Structure that contains information about the traced variables. These are:

- Variable name `VarName`
- Variable type `VarType` in S7 notation (e.g. DINT or BYTE)
- `VarOffset` specifies the offset of the variable within the data stream (relative to the start of the I/O container)
- Variable length `VarLen` (optional)

Declaration:

```
public class VDSC
{
  public string VarName;
  public string VarType;
  public System.UInt32 VarOffset;
  public System.UInt32 VarLen;
}
```

TriggerCondition

Structure indicating the trigger of a trace. The TriggerCondition contains the variable to be compared in symbolic names according to VarProvider notation.

- `Variable` refers to the variable for comparison.
- `Constant1` is a constant which is set according to the trigger type.
- `Constant2` is a constant which is set according to the trigger type.
- `Operation` indicates the comparison type as enumerator `TriggerOperationType`.
- `MatchCountTriggerPoint` indicates how many times the trigger condition must apply before the trigger is activated and data tracing starts.
- `GlobalTriggerID` is optional and contains the unique ID, which is generated by the browser when a distributed trace is set up.

Declaration:

```
public class TriggerCondition
{
  public string Variable;
  public string Constant1;
  public string Constant2;
  public TriggerOperationType Operation;
  public System.UInt32 MatchCountTriggerPoint;
  public System.UInt32 GlobalTriggerID;
}
```

For this purpose, the comparison type:

Call:

```
public enum TriggerOperationType {
 RE,
 FE,
 WIB,
 OOB,
 BHV
 }
```

The following table gives an overview of various possible combinations of trigger types and constants.

Trigger type	Symbol	Description	Constant1	Constant2
Rising Edge	RE	Triggered if variable exceeds the value of Constant1 in positive direction	Limit exceeded	-Not used-
Falling Edge	FE	Triggered if variable exceeds the value of Constant1 in negative direction	Limit exceeded	-Not used-
Within a tolerance Band	WIB	Triggered if variable is located within the interval spanned by Constant1 and Constant2	Lower limit	Upper limit
Out of tolerance Band	OOB	Triggered if variable is located outside the interval spanned by Constant1 and Constant2	Lower limit	Upper limit
Bit pattern	BHV	The bit pattern triggers if the relevant bit is 1 both in the bit mask and in the comparison result. ((v&c1)==c2)	Bit mask	Result of comparison

Overview of trigger types and constants

Enumerator that determines the type of trace.

Call:

```
public enum TraceStartTypeEnum
{
  IMMEDIATE,
  ENDLESS,
  TRIGGERED
}
```

Methods

StartTrace

The `StartTrace` method starts an initialized trace. The SoapFault "No Trace available" is returned if a trace has not yet been initialized. `StartTrace` is ignored (with a positive result) if the trace is already in progress.

The `GlobalTriggerID` must be transferred for a distributed trace. The trigger ID is a unique ID, which unambiguously identifies the station currently responsible for triggering; the same ID is used for all stations.

When a trace is restarted, the trigger ID must be reassigned so that the trigger events can be distinguished from one another.

Call:

```
public TVS_Client.TVS.StartTrace_Response StartTrace
(
  uint GlobalTriggerID
)

public class StartTrace_Response
{
  public TraceStateEnum TraceState;
}
```

StopTrace

The `StopTrace` method stops a trace in progress. The "No Trace available" SoapFault is returned when a trace has not yet been initialized. This is ignored (with a positive result) if the trace has already stopped.

Call:

```
public TVS_Client1.TVS.StopTrace_Response StopTrace ( )

public class StopTrace_Response
{
  public TraceStateEnum TraceState;
}
```

CancelTrace

The `CancelTrace` method deletes an active trace. The traces switches to EMPTY status, and all trace data are deleted. (Note: Data blocks of the WebTrace that have been requested but have not yet been retrieved are also deleted (see WebTrace::ReadData())

The "No Trace available" SoapFault is returned when a trace has not yet been initialized.

Call:

```
public TVS_Client.TVSIO.CancelTrace_Response CancelTrace ()

public class CancelTrace_Response
{
  public TraceStateEnum TraceState;
}
```

GetStatus

The `GetStatus` method returns the current status of the trace. When a trace object is deleted or has become invalid, then TraceIsValid will contain "false". In this case, the trace must be deleted via `CancelTrace`.

Call:

```
public TVS_Client.TVSIO.GetStatus_Response GetStatus ( )

public class GetStatus_Response
{
  public bool TraceIsValid;
  public TraceStateEnum TraceState;
}
```

ReadData

The `readData` method saves trace data on the RAM disk and supplies the URLs of the files in the return value. These data can be retrieved from the client with an HTTP-GET request.

The "No Tracedata available" SoapFault is returned if no trace data are available.

Call:

```
public ReadData_Response ReadData()

public class ReadData_Response
{
  public TraceStateEnum TraceState;
  public string[] URL;
}
```

InitializeWebTrace

A trace is created with `InitializeWebTrace`. `VariablesToTrace` is the list of symbolic names in accordance with VarProvider notation. `TraceDataCycle` determines the cycle clock in which the data is to be recorded. `TraceStartType` determines the type of trace. `Duration` specifies the duration of the recording in milliseconds. With an endless trace, this parameter specifies the size of the ring buffer in milliseconds.

`MatchCountTriggerPoint` in `TriggerInformation` determines how often the trigger must occur before it actually performs a trigger operation and, as a result, starts the recording. `Pretrigger` specifies the number of values which are to be recorded prior to triggering ("history").

Call:

```
public InitializeWebTrace_Response InitializeWebTrace
(
  string[] VariablesToTrace,
  TraceDataCycleEnum TraceDataCycle,
  TraceStartTypeEnum TraceStartType,
  uint Pretrigger,
  uint Duration,
  TriggerCondition TriggerInformation,
  string[] DevicesInvolved
)
```

```
public class InitializeWebTrace_Response
{
  public VDSC[] CurrentlyTracedVariables;
  public TraceStateEnum TraceState;
  public string UID;
  public string[]DevicesInvolved;
}
```

InitializeWebTraceEx

InitializeWebTraceEx is identical to InitializeWebTrace apart from the return value, where the variables are sorted according to their particular offset.

GetTraceParameters

`GetTraceParameters` can be used to read out an existing trace configuration.

Only one WebTrace is returned (if one actually exists).

Call:

```
public GetTraceParameters_Response GetTraceParameters
(
  string UID
)


public class GetTraceParameters_Response
{
  public TraceTypeEnum TraceType;
  public VDSC[] CurrentlyTracedVariables;
  public TraceStateEnum TraceState;
  public TraceDataCycleEnum TraceDataCycle;
  public string UID;
  public TraceStartTypeEnum TraceStartType;
  public unsignedInt Pretrigger;
  public unsignedInt Duration;
  public TriggerCondition TriggerInformation;
  public unsignedInt IOContainerOffset;
  public unsignedInt IOContainerLength;
  public hexBinary ClientHandle;
  public string[]DevicesInvolved;
}
```

EnableTrigger

Only for the distributed trace.

When a distributed trace has been set up and started, this activates the triggers. The `TriggerID` must be unique so that the stations can differentiate between them and do not lose their way on the network. The sequence is important: All traces should be running before the trigger is activated, to avoid loss of trigger events.

Return value: TriggerState activated or not activated.

Call:

```
public EnableTrigger_Response EnableTrigger
(
    uint TriggerID
)


public class EnableTrigger_Response
{
  public TraceStateEnum TraceState;
}
```

ReadData

With ReadData, the TVS service is requested to read out the trace buffer and pack the data in temporary files. These can then be accessed via HTTP under the relative paths specified in URL. If the buffer is empty, a response is made to the request with the "No Tracedata available" SoapFault. Currently, a maximum of 8 compiled files with a maximum of 8,192 recording points are provided for each request.

ReadDataArchive

If a trace is in the STOPPED state, this function can be used to request the recorded data.

The function supplies a URL, from which a WTRC data archive can be downloaded and then displayed in the WebTraceViewer.

Note

The WTRC file is deleted as soon as it has been downloaded.

Call:

```
public ReadDataArchive_Response ReadDataArchive ()


public class ReadDataArchive_Response
{
  public TraceStateEnum TraceState;
  public string URL;
}
```

ReadDataArchives

Only for the distributed trace.

ReadDataArchives automatically retrieves the trace data for all stations involved in a distributed trace and combines them in a WTRC file. The `URLField` array is used to transfer a list of the URLs for all the stations involved in the trace. The trace must be in the STOPPED state in order to use this method.

The return value is identical to that for ReadDataArchive.

Call:

```
public ReadDataArchives_Response ReadDataArchives
(
  public string[] URLField;
)


public class ReadDataArchives_Response
{
  public TraceStateEnum TraceState;
  public string URL
}
```

Subscriptions

Introduction

The subscription methods are listed below.

Subscribe

A subscription is created using the `Subscribe` method. The response is a `ServerHandle` that can be used to uniquely identify a subscription operation. In addition, the current TraceStatus is supplied.

Call:

```
public TVS_Client.TVS.Subscribe_Response Subscribe ( )

public class Subscribe_Response {
  public System.UInt32 ServerHandle;
  public TraceStateEnum TraceState;
}
```

SubscriptionRefresh

SubscriptionRefresh is used to query the status of the trace again; the trace was previously queried with the Subscribe method. The server response is received after HoldTime (milliseconds) + WaitTime (milliseconds), if the status has not changed during this time, or the response is received (at the earliest) after the HoldTime has expired and before the WaitTime has expired, if the status of the trace changes during the WaitTime . As such, the response is never expected before the HoldTime has elapsed.

In the response, StateChanged indicates whether the status has changed between request and response (true) or whether the TraceState status matches the status during the request (false = WaitTime expired).

Call:

```
public TVS_Client.TVS.SubscriptionRefresh_Response SubscriptionRefresh (
  System.UInt32 ServerHandle ,
  System.UInt32 WaitTime ,
  System.UInt32 HoldTime
)

public class SubscriptionRefresh_Response {
 public bool StateChanged;
 public TraceStateEnum TraceState;
}
```

SubscriptionCancel

With `SubscriptionCancel`, a subscription is canceled and the resource is enabled. The response indicates whether the Cancel operation was successful. Any current SubscriptionRefreshs are cancelled and responses sent immediately.

Call:

```
public TVS_Client.TVS.SubscriptionCancel_Response SubscriptionCancel (
  System.UInt32 ServerHandle )

public class SubscriptionCancel_Response {
  public bool SubscriptionCanceled;
 }
```

6.2.4 Appendix

6.2.4.1 MWSL functions

AddHTTPHeader

Syntax	`AddHTTPHeader(<Http-Header>)` This command can be used to add HTTP headers from MWSL . These are then not transmitted as part of the document but rather in the protocol portion of HTTP. The `AddHTTPHeader` command must therefore come before the HTML tag of a page. However, it is important to make sure that no MWSL functions that result in output into the page are used before the HTML tag.	
Parameters	`<Http-Header>`	Character string that ends with \r\n. If multiple HTTP headers are to be entered (only possible with `Set-cookie`), the individual headers must be separated by \r\n.

MWSL example	`<MWSL>`
	`var strCookie;`
	`strCookie = "Set-cookie: siemens_automation_language=de";`
	`AddHTTPHeader( strCookie );`
	</MWSL>
	`<html>`
	`<head>`
	`<title>`
	`MWSL Function AddHTTPHeader`
	`</title>`
	`</head>`
	`<body style="background-color:#DCDCDC">`
	`<h2>Testpage</h2>`
	`</body>`
	`</html>`	

createGUID

Syntax	`String createGUID()` Generates a unique alphanumeric ID in the system.	
Parameters		

Example	`<MWSL>`
	`write(createGUID());`
	`</MWSL>`	

Output	5022420B-02A7-0000-B362-3B7F4E87148D	
Valid as of	V4.4	

DecodeString

Syntax	`string DecodeString(<string>)` Converts a string encoded with EncodeString back to its original.	
Parameters	`<string>`	String in which URL-coded special characters are converted back to normal characters.
Example	`<MWSL>` `  var tmpString = "Straße Flüsse Gelände Vögel";` `  write("Original: " + tmpString + " ");` `  var tmpEncodedString = EncodeString(tmpString);` `  write("Encoded: " + tmpEncodedString + " ");` `  var tmpDecodedString = DecodeString(tmpEncodedString);` `  write("Decoded: " + tmpDecodedString);` `</MWSL>`	
Output	`Original: Straße Flüsse Gelände Vögel ` `Encoded:` `Stra&#xdf;e&#x20;Fl&#xfc;sse&#x20;Gel&#xe4;nde&#x20;V&#xf6;gel ` `Decoded: Straße Flüsse Gelände Vögel`	
Valid as of	V4.4	

die

Syntax	`die(<Param0>,<Param1>,...)` Break program execution.	
Parameters	<Param0>,<Param1>,...	Concatenation and output of the parameters.

Example	`<MWSL>`
	`  function dieTest()`
	`  {`
	`  write("Is there a life after die?");`
	`  die("--- ",123," ---");`
	`  };`
	`  dieTest();`
	`  write("There is a life after die");`
	`</MWSL>`	

Output	Is there a life after die?--- 123 ---	
Valid as of	V4.4	

EncodeString

Syntax	`string EncodeString(<string>)` Replaces special characters by their URL-coded hex value (%hh).	
Parameters	`<string>`	String in which the replacement will be performed.

Example	<MWSL>
	var tmpString = "Straße Flüsse Gelände Vögel";
	write("Original: " + tmpString + "
	");
	var tmpEncodedString = EncodeString(tmpString);
	write("Encoded: " + tmpEncodedString + "
	");
	var tmpDecodedString = DecodeString(tmpEncodedString);
	write("Decoded: " + tmpDecodedString);
	</MWSL>

Output	Original: Straße Flüsse Gelände Vögel Encoded: Straße Flüsse Gelände Vögel Decoded: Straße Flüsse Gelände Vögel
Valid as of	V4.4

ExistFile

Syntax	long ExistFile(<file name>) Checks whether a file with the name <file name> exists. The function returns the file length as the returned value. Special aspect due to the MWSL Compiler: Because all files that are associated with the MWSL Compiler (*.mwsl, *.msl, *.xsl, *.js, *.xmlf, *.css) are only present on the memory card in a compiled form, a ExistFile call must always be made to the compiled file (*.cms) in this case.	
Parameters	<file name>	Name of the sought file. The file path refers to the root directory of the user USER/SIMOTION/HMI.

Example	<MWSL>
	var tmpLength = ExistFile("/files/test.mwsl.cms");
	write("File length:"+ tmpLength);
	</MWSL>	

Output	File length: 38	
Example	<MWSL> var tmpLength = ExistFile("/files/mydata.txt"); write("File length:"+ tmpLength); </MWSL>	
Output	File length: 22	
Valid as of	V4.4	

ExistVariable

Syntax	`bool ExistVariable(<Variable Name>, <Variable Source>)` This function queries the presence of a variable. It returns `true` or `false`.	
Parameters	`<variable name>`	Variable name
	`<variable source>`	See GetVar (Page 3869)
Example	`ExistVariable("DeviceInfo.BZU", "PROCESS")`. If the process variable "DeviceInfo.BZU " existis, `true` is returned, otherwise `false`.	

See also

Conditional operations (Page 3812)

GetLanguage

Syntax	`String GetLanguage()` Returns the currently set language. The return value depends on whether one of the following values is found in the displayed sequence: 1. Content of the cookie `siemens_automation_language` if the cookie exists. 2. Value of the HTTP header `Accept-Language` if the value exists. 3. Return of the `SERVEROPTIONS` value `language` set in WebCfg.xml.	
Parameters		

Example	`<MWSL>`
	`  write("The currently set language is'" + GetLanguage() + "'");`
	`</MWSL>`	

Output	The currently set language is 'de'	
Valid as of	V4.4	

GetVar

Syntax	`GetVar(<Variable Name>, <Variable Source>, <Format String>);` This function returns the value of a variable from a variable source. If a parameter does not exist, "null" will be returned.	
Parameters	`<variable name>`	Variable name
	`<variable source>`	Name of variable source Valid values: • `URL` • `HTTP` • `PROCESS` Read variables from the providers. • `COOKIE` Read variables from the HTTP header of the cookie • `XML` • `SENDPAGE` • `DEFAULT` The default setting is `PROCESS`. If not source is stated, `DEFAULT` is selected, that is, the variable provider. The name of the variable providers, such as `SIMOTION`, `MINIWEB`, etc., are not designations for variable sources. The suitable provider is searched for in the web server based on the variable name.
	`<Format String>`	The handling of the format string depends on the variable source. Thus, this property is not possible for the variable sources `COOKIE` and `URL`. Syntax of an HTTP variable: Variables and HTTP header information (Page 3807) Syntax of a process variable: Global variables (Page 3802)

Example	`GetVar("var/userData.user1");` Returns the content of the variable `var/userData.user1`. Because `PROCESS` is the default variable source, the result corresponds to that of the following call. `GetVar("var/userData.user1", "PROCESS");` Returns the content of the variable `var/userData.user1`, the variable source `PROCESS`. `GetVar("Parameter", "URL");` Returns the content of the `Parameter` variable from the URL. `GetVar("Accept-Language", "HTTP", "?-" )` Returns the content of the HTTP variable `Accept-Language`. The format string "`?-`" indicates that all characters up to the first occurrence of the "–" character will be returned. `GetVar("var/userData.user1", "PROCESS", "[2,3]");` Returns three characters, starting from position 2 of the process variable `var/userData.user1`. The result is characters 2-5 of the process variable. `GetVar("Accept-Language", "HTTP", "[3,0]" )` Returns the content of the HTTP variable `Accept-Language` starting from the third character.

InsertFile

<table>
<tr><td>Syntax</td><td colspan="2">InsertFile(<File name>)

This command allows an existing text file to be imported individually.
The text file is interpreted before insertion with MWSL and embedded into the existing source text at the insertion point in the target file.
If the file has an ending (*.mwsl, *.msl, *.xsl, *.js, *.xmlf, *.css) associated with the MWSL compiler, the MWSL scripts it contains will be run.
URL parameters can be passed with usual syntax (<file name>?<parameter>=<value>).</td></tr>
<tr><td>Parameters</td><td><file name></td><td>Name of text file, including path.</td></tr>
<tr><td>Example</td><td colspan="2">
<pre>
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
   [...]
   <table>
     [...]
     <tr>
        <td>
          An HTML file is now displayed on the right-hand page.
        </td>
        <td>

        </td>
        <td>
          <MWSL>
          if(ExistFile("/FILES/TMPL/Output.mwsl.cms") > 0)
          {
             InsertFile("/FILES/TMPL/Output.mwsl?myparam=123");
          }
          </MWSL>
        </td>
     </tr>
     [...]
   </table>
   [...]
  </BODY>
</HTML>
</pre>
In the right-hand column of the table, the content of the file Output.mwsl is inserted and displayed in HTML format.
</td></tr>
</table>

IsAuthAlgo

Syntax	`bool IsAuthAlgo(<parAuthMethod>)` Returns true if the user logged in at the time of calling was successfully identified by the authentication method specified in `parAuthMethod`.	
Parameter	`<parAuthMethod>`	Possible specifications: `FormulaAuthentication`, `DigestAuthentication`, `BasicAuthentication`, `CertificateAuthentication`

Example	`<MWSL>`
	`  write(IsAuthAlgo("FormulaAuthentication"));`
	`</MWSL>`	

Output	1	
Note	`IsAuthMethod` returns the value '1' if no user is logged in. This behavior depends on the system because the 'Anonymous' is always logged in even when nobody else is logged in.	
Valid as of	V4.4	

isFinite

Syntax	`bool isFinite(<value>)` Returns false if the passed value is `NaN` or infinite.	
Parameters	`<value>`	Value for the check.
Example	`<MWSL>` `  write("Test of the number 123456 - isFinite: ");` `  write(isFinite(123456) + " ");` `  write("Test of NaN - isFinite: ");` `  write(isFinite(parseInt("ABC", 2)));` `</MWSL>`	
Output	`Test of the number 123456 - isFinite: 1` `Test of NaN - isFinite: 0`	
Valid as of	V4.4	

isNaN

Syntax	`bool isNaN(<value>)` Checks whether the passed value is an invalid double.	
Parameters	`<value>`	Value for the check.
Example	`<MWSL>` `  write("Test of 123456 - isNaN: ");` `  write(isNaN(123456) + " ");` `  write("Test of NaN - isNaN: ");` `  write(isNaN(parseInt("ABC", 2)));` `</MWSL>`	
Output	`Test of 123456 - isNaN: 0` `Test of NaN - isNaN: 1`	
Valid as of	V4.4	

IsSSL

Syntax	`bool IsSSL()` Returns true if the client is connected to the server via an SSL connection.	
Parameters		

Example	`<MWSL>`
	`write("If the client is connected to the server via an SSL connection:");`
	`write(IsSSL());`
	`</MWSL>`	

Output	`If the client is connected to the server via an SSL connection: 1`	
Valid as of	V4.4	

parseFloat

Syntax	`double parseFloat(<string>)` Conversion of a string to a double value.	
Parameters	`<string>`	String to be converted
Example		
Output		
Valid as of	V4.4	

parseInt

Syntax	`int parseInt(<value>,[<base>])` Conversion of a string to an integer value.	
Parameters	`<value>`	String to be converted If a value starts with `0x`, it will be interpreted as hexadecimal. Values starting `0` will be interpreted as octal. All other values are shown in decimal format. Maximum value: `0x7FFFFFFF`. If values exceed the upper limit, the maximum value `2147483647` is returned. If the value shall be shown as a negative number, put "–" in front.
	<base>	Basis to which the string shall be converted. Values: "2" = binary, "8" = octal, "16" = hexadecimal. No value = decimal interpretation.

Example	<MWSL>
	var tmpVar0 = "0x1";
	var tmpVar1 = "0x2";
	var tmpSum = tmpVar0 + tmpVar1;
	write(tmpSum + "
	");
	var tmpVarInt0 = parseInt(tmpVar0);
	var tmpVarInt1 = parseInt(tmpVar1);
	tmpSum = tmpVarInt0 + tmpVarInt1;
	write(tmpSum + "
	");
	tmpVar0 = "101";
	tmpVar1 = "100";
	tmpSum = tmpVar0 + tmpVar1;
	write(tmpSum + "
	");
	tmpVarInt0 = parseInt(tmpVar0,"2");
	tmpVarInt1 = parseInt(tmpVar1,"2");
	tmpSum = tmpVarInt0 + tmpVarInt1;
	write(tmpSum + "
	");
	tmpVar0 = "A";
	tmpVar1 = "B";
	tmpSum = tmpVar0 + tmpVar1;
	write(tmpSum + "
	");
	tmpVarInt0 = parseInt(tmpVar0,"16");
	tmpVarInt1 = parseInt(tmpVar1,"16");
	tmpSum = tmpVarInt0 + tmpVarInt1;
	write(tmpSum + "
	");
	
	</MWSL>

Output	0x10x2 3 201 9 AB 21
Valid as of	V4.4

ProcessXMLData

Syntax	`ProcessXMLData( <DATA>, <TEMPLATE>)` With this command, dynamic HTML files can be generated based on a data and template file. The parameter `<DATA>` contains the data that is interpreted with the template in parameter `<TEMPLATE>`. `ProcessXMLData` combines the two files into one HTML file. The data nodes of the data file are evaluated by the template file to be displayed. This produces a separation of the data from the content. With a subsequent change to the template file, the appearance of the pages can be altered without changing the data. This makes it easier to add to data. Using different templates, it is possible to generate pages with the same data but completely different appearances. Additional information about the template mechanism: Mode of operation of the template mechanism (Page 3816)	
Parameter	`<DATA>`	Data for the dynamic HTML file A file or a variable containing the data can be passed as a parameter. File: "`<EXTERNAL SRC=\"/datafile.xml \"/>`", in which `datafile.xml` is the file containing the data. Variable: `<variable name>` Specifies the variable name.
	`<TEMPLATE>`	Template (how the data is displayed) A file or a variable containing the templates can be passed as a parameter. File: "`<TEMPLATES><EXTERNAL SRC=\"/Template.xml\"/> </TEMPLATES>`", in which "Template.xml" is the file that contains the templates. Variable: `<variable name>` Specifies the variable name.

Example	`ProcessXMLData("<EXTERNAL SRC=\"/FILES/MWSL/variables.xml \"/>",`
	`"<TEMPLATES><EXTERNAL`
	`SRC=\"/FILES/MWSL/variablesTemplate.xml\"/></TEMPLATES>");`	

Example MWSL	An example can be found in Chapter 'MiniWeb Server Language (MWSL)' in the 'Examples' (Page 3827) section in the TestTemplate.mwsl program.	

ReadFile

Syntax	`string ReadFile(<file name>)` This function is similar to the function `InsertFile`, except that the content of the file is not written, but only returned as a return value.	
Parameters	`<file name>`	Name of text file, including path.

Example	`<MWSL>`
	`  var tmpFile = ReadFile("/files/include.mwsl");`
	`  write(tmpFile);`
	`</MWSL>`	

Output	The content of file `include.mwsl` is written to variable `tmpFile` and then written into the output.	
Valid as of	V4.4	

ReplaceString

Syntax	`ReplaceString(<variable name>,<search pattern>,<replacement string>)` Replacing strings.	
Parameters	`<variable name>`	Variable in which the characters shall be replaced.
	<search pattern>	Search pattern for replacing the characters.
	<replacement string>	string that is inserted.

Example	`<MWSL>`
	`  var tmpString = "Ein String";`
	`  var tmpOutString;`
	`  tmpOutString = ReplaceString(tmpString,"n","N");`
	`  write("Result: " + tmpOutString);`
	`</MWSL>`	

Output	Result: EiN StriNg	
Valid as of	V4.4	

SetVar

Syntax	`SetVar(<Variable Name>, <Value>)` This function is used to place process variables.	
Parameters	`<variable name>`	Variable name
	<value>	The new variable value.
Example	`SetVar("var/userData.user1", "NewUserData");` Sets the process variable `var/userData.user1` to the value `NewUserData`.	

ShareRealm

Syntax	`ShareRealm(<Group>)` Indicates whether the current user is a member of the group that is passed as a parameter. The return value can be `true` or `false`.	
Parameters	`<Group>`	The following groups are currently allowed as parameters: • `NO_REALM` No group association • `ANY_REALM` Any group association • `[group name]` Member of group `[group name]` Groups depend on configuration.
Example	`write(ShareRealm ("ANY_REALM"));` If the current user is in any defined group, 1 is output. Otherwise, 0 is output.	

MWSL example	`<MWSL>`
	`  if (ShareRealm("ANY_REALM"))`
	`  {`
	`  write ("<tr valign=\"baseline\">\r\n");`
	`  write ("<td><H2>Hello " + GetVar( "Username",  "HTTP") +`
	`  ", you're successfully logged in.</H2></td>\r\n");`
	`  write ("</tr>\r\n");`
	`  }`
	`</MWSL>`
	If the user is a member of any group, the instructions in the curly brackets are carried out.	

write

Syntax	`write(<Text>)` The `write` function writes text to the output of an HTML page.	
Parameters	`<Text>`	Text, return values of functions, or variable contents can be passed.

Example	`<MWSL>`
	` write("Hello World");`
	` // Output: Hello World`
	
	` write("Hello" + " " + "World");`
	` // Output: Hello World`
	
	` write(GetVar("Parameter", "URL"));`
	` // Output of the content of variable Parameter in the URL.`
	
	` write(5+6);`
	` // Output: 11`
	
	` var zahl1 = 5;`
	` var zahl2 = 7;`
	` var string = "Hello";`
	` write(string + ": " + zahl1 + zahl2);`
	` // Output: Hello: 57`
	
	` write(zahl1 + zahl2);`
	` // Output: 12`
	
	` write("Content of Parameter: " + GetVar("Parameter", "URL"));`
	` // Ausgabe: Content of Parameter: Hello`
	` // if Parameter contains the string "Hello".`
	`</MWSL>`	

WriteVar

Syntax	`WriteVar(<Variable Name>, <Variable Source>, <Format String>);`
	
	This command writes the content of a variable to the output.
	
	The functionality of the function `WriteVar` is similar to that of the function `GetVar`.
	`WriteVar` is equivalent to the call: `WriteVar(…) === write( GetVar (…) )`
	
	The sole difference is that `WriteVar` outputs the content of the specified variable, while `GetVar()` returns the content as a return value.	

Parameters	`<variable name>`	Variable name
	`<variable source>`	Name of variable source Valid values: • `URL` • `HTTP` • `PROCESS` Read variables from the providers. • `COOKIE` Read variables from the HTTP header of the cookie • `XML` • `SENDPAGE` • `DEFAULT` The default setting is `PROCESS`. If not source is stated, `DEFAULT` is selected, that is, the variable provider. The name of the variable providers, such as `SIMOTION`, `MINIWEB`, etc., are not designations for variable sources. The suitable provider is searched for in the web server based on the variable name.
	`<Format String>`	The handling of the format string depends on the variable source. Thus, this property is not possible for the variable sources `COOKIE` and `URL`. The call syntax is equivalent to that of the GetVar() (Page 3869) function.

Example	<MWSL>
	write(GetVar("Parameter", "URL"));
	// The content of the variable Parameter is output here.
	//GetVar returns the value of the variable,
	// which is then written to the output with write.
	// (See also GetVar() and write().)
	// The same output can also be achieved with the following
	// command.
	WriteVar("Parameter", "URL");
	//WriteVar writes directly to the output and supplies
	// no return value
	
	WriteVar("var/userData.user1");
	// Outputs the content of var/userData.user1, which is a
	// process variable.
	
	WriteVar("Accept-Language", "HTTP", "?-")
	// Outputs the content of the HTTP variable "Accept-Language"
	// up to the "-" character.
	
	WriteVar("var/userData.user1", "PROCESS", "[2,3]");
	// Outputs characters 2 - 5 of the process variable var/userData.user1.
	
	WriteVar("Accept-Language", "HTTP", "[3,0]")
	// Outputs the content of the HTTP variable "Accept-Language"
	// starting from the third character.
	</MWSL>

Example	<MWSL>
	// Identical calls
	WriteVar("var/modeOfOperation");
	WriteVar("var/modeOfOperation", "PROCESS");
	WriteVar("var/modeOfOperation", "DEFAULT");
	</MWSL>

See also

Global variables (Page 3802)

WriteXMLData

Syntax	`WriteXMLData ( <DATA>, <TEMPLATE>)` WriteXMLData outputs the data in contrast to ProcessXMLData. Instead of `write( ProcessXMLData(…));` , you can also write `WriteXMLData(…);` . `WriteXMLData();` is assigned the same parameters as `ProcessXMLData();`.	
Parameter	`<DATA>`	Data for the dynamic HTML file A file or a variable containing the data can be passed as a parameter. File: "`<EXTERNAL SRC=\"/Datendatei.xml \"/>`", in which "data file.xml" is the file containing the data. Variable: `<variable name>` Specifies the variable name.
	`<TEMPLATE>`	Template (how the data is displayed) A file or a variable containing the templates can be passed as a parameter. File: "`<TEMPLATES><EXTERNAL SRC=\"/Template.xml\"/></TEMPLATES>`", , in which "Template.xml" is the file containing the templates. Variable: `<variable name>`

Example	`WriteXMLData("<EXTERNAL SRC=\"/FILES/MWSL/variables.xml \"/>",`
	`"<TEMPLATES><EXTERNAL`
	`SRC=\"/FILES/MWSL/variablesTemplate.xml\"/></TEMPLATES>");`
	
	Additional information about the template mechanism: Mode of operation of the template mechanism (Page 3816)	

NodeIndex

Variable	NodeIndex	NodeIndex is a process variable that is available when a template is parsed. This variable outputs the number of nodes that have already been run through. The access is exactly the same as for other variables of the PROCESS variable source. Additional information about the template mechanism: Mode of operation of the template mechanism (Page 3816)
Example	<?xml version="1.0" ?> <TEMPLATES> <TEMPLATE NAME="Variable"> <POSITION NAME="LINE"> <![CDATA[<MWSL> WriteVar("NodeIndex ") </MWSL>]]> </POSITION> </TEMPLATE> </TEMPLATES>	

NodeLevel

Variable	NodeLevel	NodeLevel is a process variable that is available when a template is parsed. This variable outputs the hierarchy level of the current node. The access is exactly the same as for other variables of the PROCESS variable source. Additional information about the template mechanism: Mode of operation of the template mechanism (Page 3816)
Example	<?xml version="1.0" ?> <TEMPLATES> <TEMPLATE NAME="Variable"> <POSITION NAME="LINE"> <![CDATA[<MWSL> WriteVar("NodeLevel") </MWSL>]]> </POSITION> </TEMPLATE> </TEMPLATES>	

6.3 SIMOTION IT OPC UA

6.3.1 Preface

6.3.1.1 Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

6.3.1.2 SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4.3:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

6.3.2 Fundamental safety instructions

6.3.2.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

6.3.2.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

6.3.2.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

6.3.2.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

6.3.3 Introduction

6.3.3.1 SIMOTION IT Overview

SIMOTION IT Overview Manuals

The "SIMOTION IT Ethernet-based HMI and diagnostic functions" are described in four manuals (IT = Information Technology):

- **SIMOTION IT Diagnostics and Configuration**
  This manual describes the direct diagnosis of SIMOTION devices. Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.
  See the SIMOTION IT Diagnostics and Configuration Manual.
- **SIMOTION IT Programming and Web Services**
  This manual describes the creation of user-defined Web pages and access to the diagnostic functions via the two Web services provided by SIMOTION IT.
  A Web service enables users to create their own client applications in any programming language. These applications then communicate with the SIMOTION device using Web technologies. The SOAP (Simple Object Access Protocol) communication protocol is used for transferring commands.
  The manual includes information on programming such clients, as well as a description of the SIMOTION IT Web services (OPC XML-DA, Trace via SOAP TVS) via which data and operating states of the controller can be accessed and the variable trace functions can be used.
  See the SIMOTION IT Programming and Web Services Manual.
- **SIMOTION IT Virtual Machine and Servlets**
  This manual describes the Java-based function packages. The Jamaica Virtual Machine (JamaicaVM) is a runtime environment for Java applications on the SIMOTION device. It is an implementation of the "Java Virtual Machine Specification".
  The Servlets section of the manual describes the use of servlets in a SIMOTION device.
  See the SIMOTION IT Virtual Machine and Servlets Manual.
- **SIMOTION IT OPC UA**
  This manual describes access to SIMOTION devices via OPC UA.

See also

PDF in the Internet: SIMOTION IT Programming and Web Services (https://support.industry.siemens.com/cs/ww/en/view/109476528)

PDF in the Internet: SIMOTION IT Diagnostics and Configuration (https://support.industry.siemens.com/cs/ww/en/view/109476533)

PDF in the Internet: SIMOTION IT Virtual Machine and Services (https://support.industry.siemens.com/cs/ww/en/view/109476529)

6.3.3.2 Introduction

OPC Unified Architecture (OPC UA)

For OPC UA, this is an extension to the OPC industry standard. Functions such as platform-independence, scalability, high availability and Internet capability have been added to the standard.

Standardized communication via the Internet and firewalls

For data exchange, OPC UA uses a TCP-based, optimized, binary protocol. Whereby, port 4840 entered for the IANA is used for data transfer. As option, HTTP and Web service are also supported.

Just enabling port 4840 suffices to ensure data transfer.

Service-oriented architecture

OPC-UA defines generic services based on the Service Oriented Architecture (SOA) design paradigm. This architecture provides service providers with requests that can be edited and whose results are returned as response.

A WDSL for describing the services, as required for classic Web services, is no longer necessary. Generic services are already defined and standardized for OPC UA.

Security concept

To protect against unauthorized access, the data is encrypted using recognized Internet standards (SSL, TLS, AES). The security mechanisms belong to the standard and are mandatory for manufacturers.

Accessibility and reliability

A robust architecture, with reliable communications mechanisms, configurable timeouts and automatic error detection, permits fail-safe systems with maximum reliability.

Simplification thanks to standardization

OPC-UA defines an integrated address space and an information model. Process data, alarms and historical data, together with function calls, can be represented in the information model. Whereby, OPC UA combines all classic OPC functionality, and describes complex procedures and systems in standardized object-oriented components.

Information consumers that only support the base rules can also process data without needing to know the relationships of the complex server structures.

6.3.4 Software description

6.3.4.1 OPC UA implementation

OPC UA implementation

The Data Access (DA) of the OPC UA server functionality has been implemented.

OPC UA binary encoding is supported. Encrypted and non-encrypted data transfer are possible.

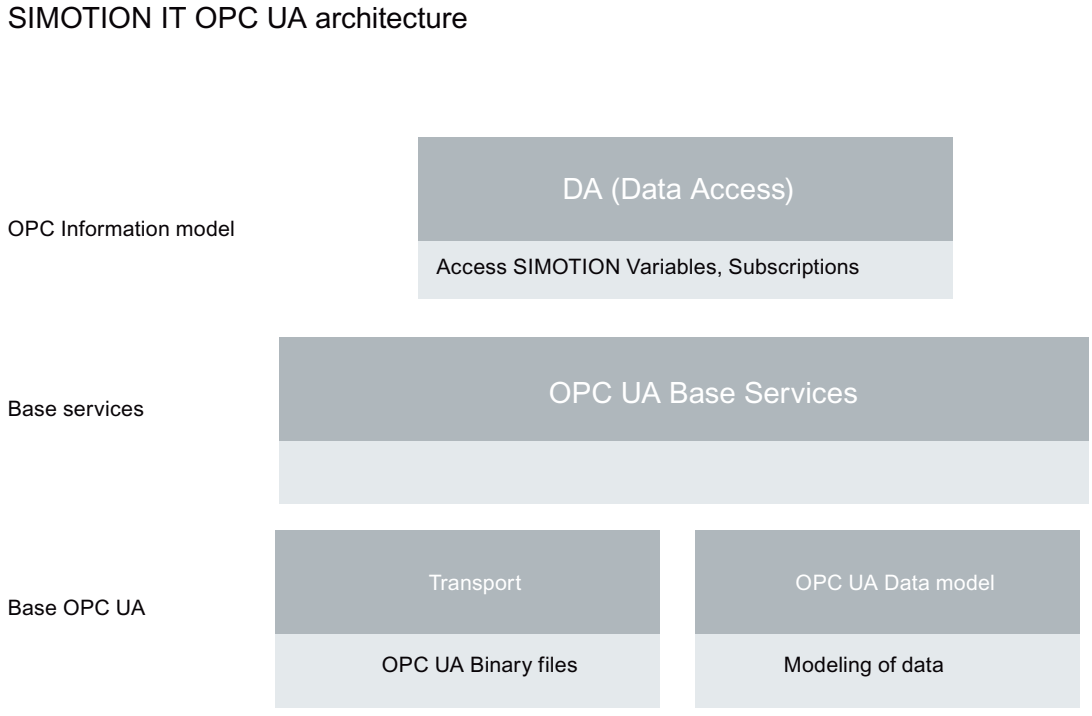


Figure 6-147 Excerpt from the OPC UA layer model

6.3.5 Commissioning (software)

6.3.5.1 Commissioning

Commissioning

To access OPC UA , an interface of the SIMOTION device for access must be activated and a TCP/IP port must be assigned on the OPC UA. page. A Realm must be set for the read and write rights.

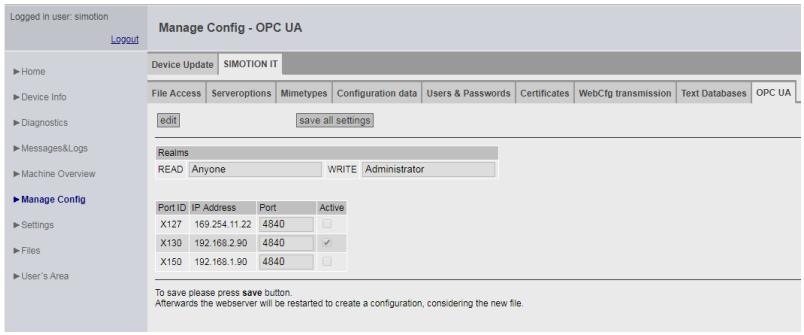


Figure 6-148 OPC UA Activate interfaces

Inputs on the OPC UA page saved with **save all settings** cause a change to be made in WebCfg.xml.

If the IP address of an interface is changed, the Web server is restarted; this disconnects all connections, including the OPC UA connections.

Activation of OPC UA in HW Config

The Web server of the SIMOTION controller is activated in the HW Config. To do that, navigate via **Device object properties** to the **Ethernet extended / Web server** tab.

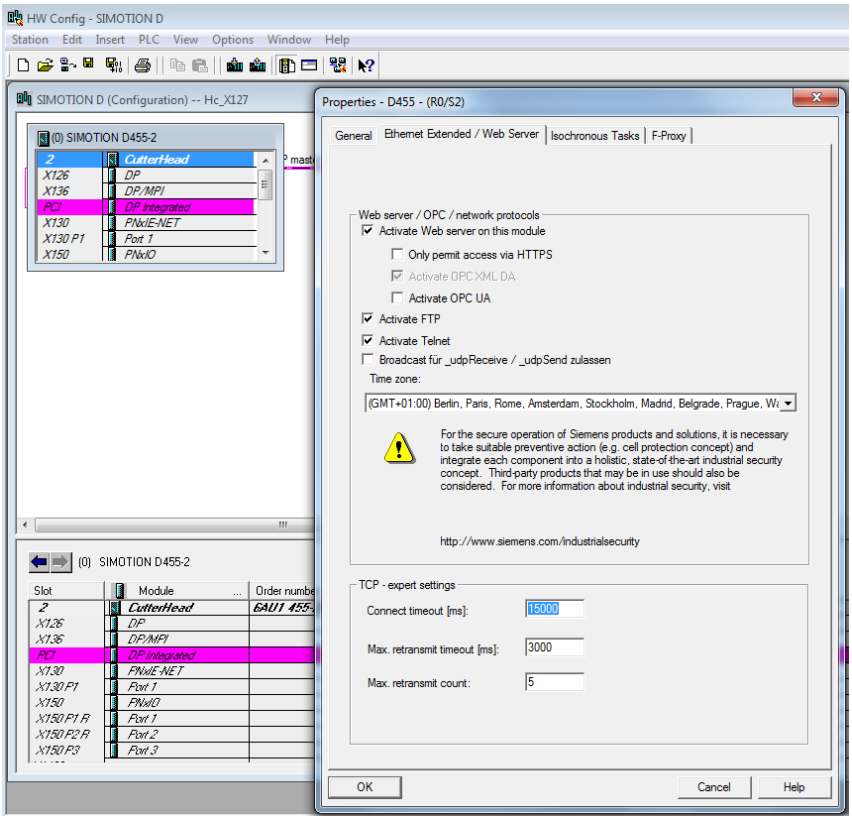


Figure 6-149 HW Config settings

The screenshot shows the default Web server settings. When you deactivate a service, the corresponding communications port is closed. If no service is activated, the Web server of the controller is also deactivated.

Click **Activate OPC UA** to activate the access.

Activation of OPC UA in the TIA Portal

To activate the Web server in the TIA Portal, proceed as follows:

1. Select the SIMOTION device in the network view / device view.
2. In the Inspector window, select the "Properties" tab and click the "General" tab.
3. Select "Web server / OPC / Network protocols".
   The Web server is deactivated in the basic setting. You must activate the relevant checkboxes so that the CPU displays the Web pages.

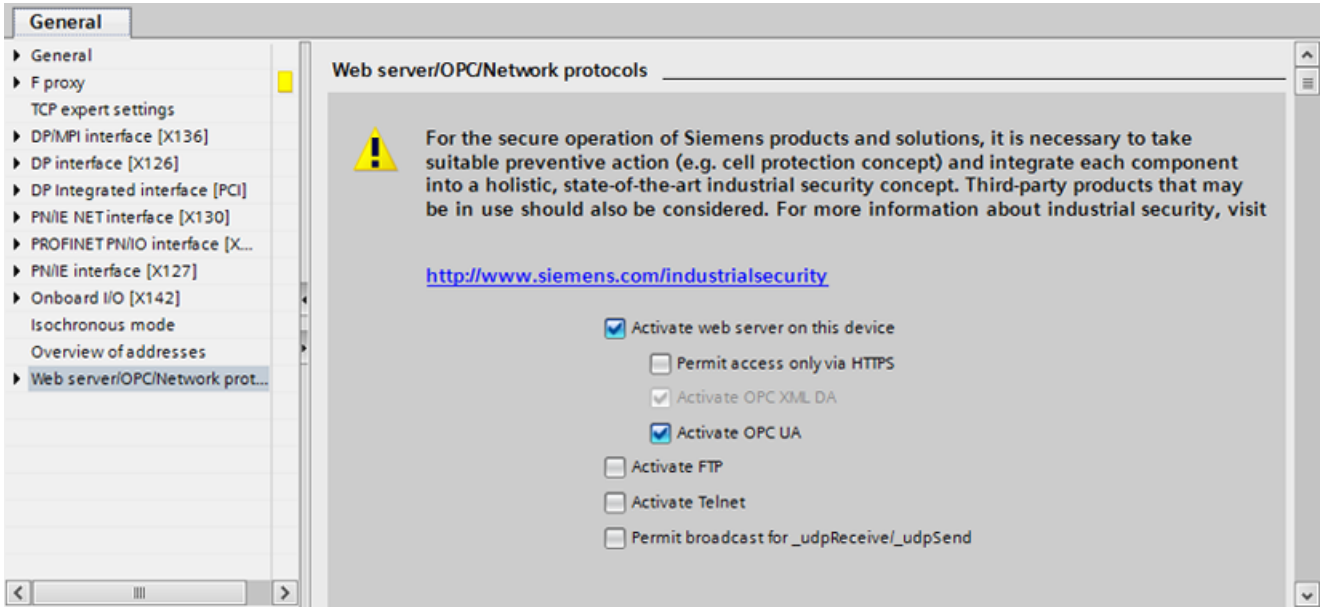


Figure 6-150 Activating the Web server

Activate the Web server and OPC UA.

Configuration prior to Version 4.5

WebCfg.xml must be changed in order to enter the OPC UA IP address.

The IP address with the port is stored in the <ENDPOINTDESCRIPTION> element.

```
<CONFIGURATION_DATA>
   ...
   <OPCUAAPPLICATION READ="<ReadRealm>" WRITE="<WriteRealm>"
      <ENDPOINTDESCRIPTION URL="X127:4840"/>
   </OPCUAAPPLICATION>
   ...
</CONFIGURATION_DATA>
```

The user authentication is supported with the READ, WRITE and BROWSE attributes. These attributes link Realm with the associated access type. The associated rights are assigned in the Realm.

The IP address with which the server can be accessed is entered in the URL attribute of the <ENDPOINTDESCRIPTION> element.

Example SIMOTION IP address (factory setting)

```
<ENDPOINTDESCRIPTION URL="X127:4840"/>
```

SIMOTION IT configuration

Further information for configuring the controller can be found in the SIMOTION IT Diagnostics and Configuration Manual. The 'Commissioning' section contains information for activating the Web server in the HW Config and in TIA Portal. The 'SIMOTION IT configuration' section explains the structure of the WebCfg.xml configuration file and the user administration.

6.3.5.2 SIMOTION IT configuration

Settings via the Web server

OPC UA access to the controller requires that some settings must be set via the SIMOTION IT Web server:

- Configuration of a user database UserDataBase.xml
- Access protection with OPC UA certificates

Note

Commissioning prerequisite

The successful commissioning, as described in the 'Commissioning' chapter, is prerequisite for configuring the controller.

This chapter includes several sections from the "SIMOTION IT Diagnostics and Configuration" manual. This chapter explains the user database and accessing with certificates to the controller.

Log-in administration

Structure of the login administration

SIMOTION IT uses a user database to safeguard access to a device. The UserDataBase.xml file contains this user data.

If the controller is started without a user database, a user database is automatically created when the controller starts up. This user database contains no users and is therefore empty.

If Web pages are called in this condition, the anonymous user `Anonymous` is active. This user has no special access rights.

The Web pages can only be used if the Web server has been activated via SCOUT or HW Config. If the Web server has not been activated, communication with the device is not possible. The services are activated by default when a new device is set up, and they must be explicitly deactivated to prevent access. In TIA Portal, the services are disabled by default.

The user administration is based on the Administrator user group. If no user who belongs to the Administrator user group exists in the UserDataBase.xml, no users can be set up, edited, or deleted via the User's & Passwords Web page.

There are many application cases related to the Web server and UserDataBase.xml that differ in terms of the individual files on the memory card.

Empty memory card, no SCOUT project exists on the memory card and empty UserDataBase.xml

The memory card only contains the firmware and the licenses.

In the as-delivered state, file UserDataBase.xml contains no users and is "empty" as far as the system is concerned.

In this case, the status of the controller is **Security Level Low**. To make it possible to perform commissioning via the Web server, all the Web pages can be used without login in this status. The FTP and Telnet can be accessed with any user name and password.

Users can be set up in the following ways to create a valid user database.

1. Call page **Manage Config > SIMOTION IT > Users & Passwords**. Add a user with Administrator group. As soon as the user is saved, the Web server switches to the **Security Level normal** condition because the user database now contains a valid entry.
2. Create a UserDataBase.xml file with content as described below. Upload via the Web page **Manage Config > SIMOTION IT > Users & Passwords**.
3. Create a **UserDataBase.xml** file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
4. Create a UserDataBase.xml file with content as described below. Use the device update tool and save the UserDataBase.xml file to the USERDATABASE folder in directory IT Config.

SCOUT project exists on the memory card and empty UserDataBase.xml

If a valid project exists on the card, the **Security Level normal** status applies to the Web server, in which the Web pages, FTP and Telnet are protected by a login. However, if the UserDataBase.xml file is in the as-delivered state, it contains no users. In this case, login will not be possible.
The user database can be edited in any of the following ways:

1. Delete project with **Delete user data on card** in SCOUT. The Web server assumes the **Security Level Low** status. The user database can be edited as described above.
2. Create a UserDataBase.xml file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
3. Create a UserDataBase.xml file with content as described below. Use the device update tool (but not via the Web pages) and store file UserDataBase.xml in a folder USERDATABASE in directory IT Config.
4. Service switch 8, simotion.ini or the PSTATE program can be used to reset to Security Level low and thus change the password.

SCOUT project exists on the memory card and UserDataBase.xml contains valid users

If a valid project exists on the card, the **Security Level normal** status applies to the Web server, in which the Web pages, FTP and Telnet are protected by a login.

The user database can be edited in any of the following ways:

1. Call page **Manage Config > SIMOTION IT > Users & Passwords**. After logging in successfully with administrator rights, new users can be created and existing users edited. This however assumes that at least one user who belongs to the Administrator group has already been set up.
2. Create a UserDataBase.xml file with content as described below. Connect the CF card to the PC via a card reader and save the XML file at /USER/SIMOTION/HMICFG/USERDATABASE/.
3. Create a UserDataBase.xml file with content as described below. Use of the device update tool (but not via the Web pages) and storage of the file UserDataBase.xml in a folder USERDATABASE in directory IT Config.

Authentication

The authentication is constructed as follows:

- There are USERs.
- Every USER has a password that can be entered as plain text before startup. After startup, the password exists as A1 Hash .
- Users belong to groups (GROUP).
- Web pages, directories, and applications are protected by secure realms (REALM) defined for each group.
- Only users that belong to the realm can access the protected page.
- Each realm has a group of users who are authorized for access.
- A user can belong to multiple groups.

If a login attempt fails, this is logged in the system log.

Note

Editing the UserDataBase.xml file

- If the UserDataBase.xml file is not adapted, after the SCOUT project has been downloaded, it will no longer be possible to login to the Web pages or access with FTP and Telnet, because no valid user exists.
- The user database UserDataBase.xml must contain at least one user who is a member of group Administrator. Group Administrator is the REALM that the system expects for accessing protected applications, whose access rights cannot be set via WebCfg.xml.
- The editor used for editing UserDataBase.xml must be set to UTF-8 encoding.
- If file UserDataBase.xml contains illegal characters or the XML syntax contains errors, the file cannot be evaluated by the system. This makes login impossible.
- After startup, all plain text passwords are deleted and only exist in encrypted form. Neither can they be ascertained by the administrator. However, the administrator can assign a new password without knowing the old password.
- Since the password is no longer available in plain text in UserDataBase.xml, the password must be entered again each time a change is made to the groups of an existing user, otherwise the A1-Hash cannot be calculated.
- After loading via FTP, the controller must be restarted to transfer the UserDataBase.xml file. A restart the Web server does not suffice.

Structure of the UserDataBase.xml file

The user data is stored in UserDataBase.xml. UserDataBase.xml is located in directory /USER/SIMOTION/HMICFG/USERDATABASE

Sample configuration

UserDataBase.xml before startup

```
<?xml version="1.0" encoding="UTF-8"?>
<UserDataBase>
  <USER NAME="service"
    PASSWORD="a67_YjH"
    ChangePassword="never"
    DESCRIPTION="Administrator with all rights"
    REAL_NAME="">
      <GROUP NAME="Anyone"/>
      <GROUP NAME="Administrator"/>
  </USER>
  <USER NAME="user1"
    PASSWORD="93!ujEa"
    ChangePassword="allowed"
    DESCRIPTION="Normal user"
    REAL_NAME="">
      <GROUP NAME="Anyone"/>
  </USER>
</UserDataBase>
```

UserDataBase after startup

```
<?xml version="1.0" encoding="UTF-8"?>
<UserDataBase>
  <USER NAME="service"
    ChangePassword="never"
    DESCRIPTION="Administrator with all rights"
    REAL_NAME="">
      <GROUP NAME="Anyone"         A1="0302831a41b222c5f5bfc22e5ff80620"/>
      <GROUP NAME="Administrator"
A1="fa712df9294b40baa1e7504f8dd2b0d5" />
  </USER>
  <USER NAME="user1"
    ChangePassword="allowed"
    DESCRIPTION="Normal user"
    REAL_NAME="">
      <GROUP NAME="Anyone" A1="c5a15667e4d0cadff85d35354ea0fbb6"/>
  </USER>
</UserDataBase>
```

Table 6-58 Attributes of the USER node

Attribute	Permissible values	Description
NAME	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Login name
PASSWORD	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Password in plain text
CHANGEPASSWORD	ALLOWED ⇒ The password can be changed by the user in the Web page (default setting). NEVER ⇒ The password cannot be changed by the user in the Web page.	Behavior when user logs in via Web pages. No effect when opening the file in the file system
DESCRIPTION	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Description of the user
REAL_NAME	Numerals, letters, special characters but not: =, " , <,>, %, &, \, `,'	Actual name of the user

Table 6-59 Attributes of the GROUP node

Attribute	Permissible values	Description
NAME	Numerals, letters, special characters but not: : =, " , <,>, %, &, \, `,'	Name of the group.
A1	Valid hash value (numerals, letters)	Hash value that is expressed as a MD5 checksum via USER NAME, USER PASSWORD and GROUP NAME. If none exists, generated after the controller starts up.

NOTICE
Invalid XML file
Using impermissible values may invalidate the XML file so that it cannot be read.

Login and WebCfg.xml

Differentiated protection of Web pages, directories, and applications with WebCfg.xml

The realms are assigned for individual Web pages, directories, and applications in configuration file WebCfg.xml. Content requiring protection is labeled REALM Administrator. The users belonging to this group are specified in file UserDataBase.xml.

Besides REALM Administrator used by the system, the user can create and apply his own realms to protect Web pages, etc.

Example

Excerpt UserDataBase.xml:

```
...
<USER NAME="user1"
```

```
    PASSWORD=""
    ChangePassword="allowed"
    DESCRIPTION="Service with restricted rights"
    REAL_NAME="John Smith">
<GROUP NAME="Anyone"
        A1="c5a15667e4d0cadff85d35354ea0fbb6"/>
<GROUP NAME="Servicegroup"
        A1="45735fdcee4d0cdfafde825354ea0aa17"/>
</USER>
…
```

Excerpt WebCfg.xml:

```
…
<settings.mwsl.cms ALIAS="html/standard/settings.mwsl.cms"
REALM="Servicegroup" READ="Servicegroup" WRITE="Servicegroup"
MODIFY="Servicegroup"/>
…
```

`user1` has been inserted. This user belongs to the new group `Servicegroup` and has access to the `settings.mwsl` page. However, any user who wants to open the Settings page must belong to the `Servicegroup` group. It is therefore recommended that administrators belong to all the groups that exist in the user database.

Realms for applications

Besides the realms for individual MWSL pages and directories, the REALM of some of the applications of the Web server are also defined in the configuration file WebCfg.xml.

These realms can be adapted if necessary.

⚠ CAUTION

Deleting a REALM

If you delete a REALM, the associated pages can be accessed without a login. So carefully check which pages were protected by the REALM.

- Web service for OPC-XML DA and therefore reading, writing, monitoring of the variables of all providers
  `<WEBSERVICE NAME="OpcXml" URL="/SOAP/OPCXML"`
  `REALM="Administrator" />`

  Note

  As-delivered state without REALM

  In the as-delivered state, this value has no REALM to ensure downward compatibility reasons! It is therefore recommended to prepare the OPC-XML DA client currently being used for password and user name use and to set the REALM here.

- Application for writing variables in all providers on the HTML diagnostics pages:
  <VarApp REALM="Administrator" />
- Application for updating project and firmware:
  <FWUpdtApp REALM="Administrator" />

- Application for reading and writing the user database UserDataBase.xml
  <UserDataBaseApp REALM="Administrator" />
- Application of Jamaica VM for calling servlets
  <JApp REALM="Administrator" />

In addition, there are system applications that require login of a user who belongs to the Administrator group.

As delivered

So that you can access the SIMOTION controller via HTTPS in the delivery state of the SIMOTION IT diagnostics standard pages, a server certificate and a private key are supplied as a file on the device.

When you attempt HTTPS access using the key files supplied with the system, you will be warned that the certificate is unknown and that the current address of the controller does not match the name of the controller in the certificate.

Note

Secure data transmission

A HTTPS connection via the preinstalled certificate is not the most secure way of accessing the controller. The preinstalled certificate should therefore only be used if no self-created or purchased certificate can be used.

Creating key files with the script cert.pl (V4.1 and higher)

Overview

Note

HTTPS connections are supported as of SIMOTION V3.2.

If no Certification Authority (CA) is available in your organization, we recommend that you follow the steps described in the following section. The certificate and the key files are created with the OpenSSL tool and a Perl script cert.pl

Carry out the following steps:

No.	Working step	Remark
1.	Install a Perl runtime environment	If Perl is not installed
2.	Install OpenSSL	
3.	Create the certificate and key files with Perl script	
4.	Import the created certificate to the PC browser	This step must be performed once for each PC.

HTTPS access is available after the SIMOTION controller ramps up.

Installation of a Perl runtime environment

Install Perl if the Perl runtime environment is not present on your PC. You can download a free setup for Windows from the following websites, for example:

- http://www.activestate.com
- http://www.perl.org

Installation of OpenSSL

You can download a free OpenSSL setup for Windows, for example, from the following website:

- http://slproweb.com/products/Win32OpenSSL.html

Installation of cert.pl

The cert.pl Perl script generates certificates for the controller. The script is on the AddOn-DVD 2 in the \Addon\4_Accessories\SIMOTION_IT\6_Tools directory.

A new <CertDir> directory (e.g. c:/cert) is first created on the PC and the cert.pl file is copied into it.

Call syntax cert.pl

```
Usage: perl cert.pl [-h][-?][-cert CertPath][-site <Site name>][-cpu
<CPU name>]
                   [-ip <IPAddr>[,<IPAddr>,...]][-ossl <path>][-tools
<path>]
                     [-d <duration>][-img <path>][-wcfg <WebCfgPath>]
                     [-ca][-srvn][-srvu][-ksize size]

Options:

  -cert <certpath>: Workspace used for the creation of certificates
(default: current directory)

  -site <site name>: Name of the site the cpu is belonging to

  -cpu <CPU name>: Name of the cpu

  -ip <IPAddr>[,<IPAddr>,...]: List of IP addresses belonging to 1
cpu (no spaces allowed)

  -ca: Create new root CA

  -srvrn: Create new server certificate

  -srvru: update existing server certificate

  -d <duration>: Duration of validity (in days)

  -tools <path>: Path to the tools dir containg eg. 7za.exe

  -img <path>: Path to the output dir (default: <certpath>)

  -e: Export the certificates of 1 cpu to the path specified by the -
img option

  -ossl <path>: Path to an openssl installation (eg. C:/OpenSSL-Win32)

  -ksize <size>: Key size (default: 2048)
```

```
-h: Print this help
-?: Print this help
-wcfg WebCfgFile: Use <WebCfgFile> as a template
```

The path to the OpenSSL installation is determined via the "OPENSSL_CONF" environment variable from the program. This environment variable is created during the installation of OpenSSL with a setup program. If the environment variable is not set, then the "-ossl" option must be used.

Creating a SSL certificate yourself

The cert.pl Perl tool can be used to generate the certificates required for customer systems (sites) and combine them into packages for loading.

Generation of root and server certificates

As of SIMOTION Version 4.4, there are two applications for which the tool can be used:

Automatic generation of the certificate

In the first application, the required server certificates and their private keys are generated automatically on first access to the controller via HTTPS. A root certificate and the associated private key are required for this purpose.
The root certificate and the associated private key are generated using the Perl tool.

Call: `perl cert.pl -ca -ossl C:/OpenSSL-Win64`

Name of the server certificate:	ITDiagRootCA.crt
Name of the private key:	ITDiagRootCA.key
Storage location in the file system:	<certpath>/CA

Note

UaExpert

Version 1.4.4 of the UaExpert functions only with a binary certificate in DER format. The certificate in DER format is stored in the /USER/SIMOTION/HMICFG/CERTSTORE/CA path in the "ITDiagRootCA.zip" ZIP file on the memory card of the controller.

The data of the certification institution is queried:

- Country (2-character code, e.g. DE)
- State (e.g. Bavaria)
- City (e.g. Erlangen)
- Company (e.g. MyCompany Corp.)
- Department (e.g. IT Development)
- Common name (e.g. ITDiagRootCA)
- E-mail (e.g. sepp@MyCompany.com)

Self-generated certificates

In this case, the required server certificates must be generated in addition to the root certificate.

Call:

```
perl cert.pl [-ca] [-cert <certpath>] [-site <sitename>] -cpu
<cpuname> -ip <IP-Addr1>,<IP-Addr2>,.... -srvn -ossl <opensslpath>
or
perl cert.pl [-ca] [-cert <certpath>] [-site <sitename>] -cpu
<cpuname> -ip <IP-Addr1>,<IP-Addr2>,.... -opcuacert -ossl
<opensslpath>
```

Name of the generated root certificate	ITDiagRootCA.crt, bzw. ITDiagRootCA.zip
Name of the private key	ITDiagRootCA.key
Storage location in the file system	<certpath>/CA

Name of the generated server certificates	<IP-Addr>.SSL.crt (z.B. 192.168.2.90.SSL.crt)
Name of the private key	<IP-Addr>.SSL.key (z.B. 192.168.2.90.SSL.key)

or

Name of the generated server certificates	<IP-Addr>_OPCUA.crt (z.B. 192.168.2.90_OPCUA.crt)
Name of the private key	<IP-Addr>_OPCUA.key (z.B. 192.168.2.90_OPCUA.key)
Storage location in the file system	<certpath>/<sitename>/<cpuname>/<IP-Addr>

The root certificate will only be generated if none already exists. For all subsequent calls, the existing root certificate is used to sign the newly generated server certificates. The generation of a new root certificate can be forced with the -ca option.

The list of IP addresses (<IP-Addr1>, <IP-Addr2>) must not contain any blanks. This also applies to all other parameters.

The data of the applicant is queried when creating the first server certificate of a site. If `-site` was not specified, of a CPU, the applicant data is queried:

- Country (2-character code, e.g. DE)
- State (e.g. Bavaria)
- City (e.g. Erlangen)
- Company (e.g. MyCompany Corp.)
- Department (e.g. IT Development)
- E-mail (e.g. sepp@MyCompany.com)

Note

Validity duration of the certificates

The default validity is 30 years (effectively infinite).

The d option generates certificates with a shorter validity. In this case, HTTPS communication will no longer function after the validity has expired.

The user is responsible for installing the new valid certificates on all affected controllers.

Update of existing server certificates

If one of the parameters essential for the generation of the server certificates (e.g. the server certificate, the lifetime or the configuration) changes, an update can be started for the server certificates.

Call: `perl cert.pl [-cert <certpath>] [-site <sitename>] [-cpu <cpuname>] -srvu -ossl <opensslpath>`

This call also affects the certificates for OPC UA.

If the `-cpu` parameter is missing, all certificates of the CPUs belonging to the site are renewed.

If the `-site` parameter is also missing, all certificates are renewed.

Deletion of existing server certificates

Server certificates can be deleted:

Call: `perl cert.pl [-cert <certpath>] [-site <sitename>] [-cpu <cpuname>] [-ip <IP-Addr1>,<IP-Addr2>,....] -svrr -ossl <opensslpath>`

Export of existing server certificates

The generated certificates can be exported for each CPU.

Call: `perl cert.pl [-cert <certpath>] [-img <path>] [-site <sitename>] -cpu <cpuname> [-ip <IP-Addr1>,<IP-Addr2>,....] -e -ossl <opensslpath>`

The path to the exported images can be specified with the `-img` option.

Storage location in the file system: `<path>/images/<sitename>/<cpuname>`

A directory structure can be found at `<imgpath>/images/<sitename>/<cpuname>/image` which can be copied to the `/USER/SIMOTION/HMICFG` directory of the CF card. An upload-capable ZIP archive (<cpuname>.zip) is also generated under `<imgpath>/images/<sitename>/<cpuname>` if the zipper 7z.exe (7za.exe in older versions) is in `<toolspath>` (option `-tools <toolspath>`).

- The archive can be unpacked in the HMICFG directory. Any existing server certificates have to be removed. To do this, the complete `/USER/SIMOTION/HMICFG/certstore/servercerts` directory is deleted. The controller then has to be restarted.
- The server certificates can also be loaded to the CPU via the **Certificates** website at **Manage Config**. Unnecessary files and directories are deleted and a restart of the Web server triggered.

SIMOTION versions prior to Version 4.4

For SIMOTION versions prior to Version 4.4, the functionality of the tool is retained.

Generated server certificates are entered in a copy of a template of the WebCfg.xml file.

The template is sought in one of the following directories in the specified order:

- –wcfg Option
- <certpath>/<sitename>/<cpuname>/<ipaddr>
- <certpath>/<sitename>/<cpuname>
- <certpath>/<sitename>
- <certpath>

6.3.6 Operator control (software)

6.3.6.1 Functionality

Functionality

The following functionality is available:

- Browse the variable management area
- Read and write variables
- Subscriptions

6.3.6.2 Access to the variable management area

Variable access

Access can be made to the SIMOTION variable management area data:

- Device system variables
- TO system variables
- TO configuration variables
- Drive parameters
- Global user variables
- Interface variables for units
- I/O variables

As of version 4.5, all data of the Simotion OPC UA provider is available. Only simple data types (arrays), such as Integer, Float, character strings and enumerations, are supported.

As of version 5.3, structures are also available.

Data types

The assignment of the SIMOTION data types to the data types in OPC UA is shown in the following:

SIMOTION data type	OPC UA data type (UaExpert)
BOOL	Boolean
BYTE	Byte
WORD	UInt16
DWORD	UInt32
SINT	Sbyte
USINT	Byte
INT	Int16
DINT	Int32
UDINT	UInt32
REAL	Float
LREAL	Double
UINT	Uint16
TIME	String
DATE	String
TIME_OF_DAY	String
DATE_AND_TIME	DateTime
STRING	String

UAExpert section

Data Access View

#	Server	Node Id	Display Name	
1	SIMOTION OPC UA	NS2|String|unit/VariableList.varBOOL	unit/VariableList.varBOOL	true
2	SIMOTION OPC UA	NS2|String|unit/VariableList.varBYTE	unit/VariableList.varBYTE	177
3	SIMOTION OPC UA	NS2|String|unit/VariableList.varWORD	unit/VariableList.varWORD	13489
4	SIMOTION OPC UA	NS2|String|unit/VariableList.varDWORD	unit/VariableList.varDWORD	152081
5	SIMOTION OPC UA	NS2|String|unit/VariableList.varSINT	unit/VariableList.varSINT	-79
6	SIMOTION OPC UA	NS2|String|unit/VariableList.varUSINT	unit/VariableList.varUSINT	177
7	SIMOTION OPC UA	NS2|String|unit/VariableList.varINT	unit/VariableList.varINT	13489
8	SIMOTION OPC UA	NS2|String|unit/VariableList.varUINT	unit/VariableList.varUINT	13489
9	SIMOTION OPC UA	NS2|String|unit/VariableList.varDINT	unit/VariableList.varDINT	152081
10	SIMOTION OPC UA	NS2|String|unit/VariableList.varUDINT	unit/VariableList.varUDINT	152081
11	SIMOTION OPC UA	NS2|String|unit/VariableList.varREAL	unit/VariableList.varREAL	152104
12	SIMOTION OPC UA	NS2|String|unit/VariableList.varLREAL	unit/VariableList.varLREAL	152113
13	SIMOTION OPC UA	NS2|String|unit/VariableList.varTIME	unit/VariableList.varTIME	T#40d
14	SIMOTION OPC UA	NS2|String|unit/VariableList.varDATE	unit/VariableList.varDATE	D#000
15	SIMOTION OPC UA	NS2|String|unit/VariableList.varTimeOfDay	unit/VariableList.varTimeOfDay	TOD#0
16	SIMOTION OPC UA	NS2|String|unit/VariableList.varDateAndTime	unit/VariableList.varDateAndTime	1992-0
17	SIMOTION OPC UA	NS2|String|unit/VariableList.varSTRING	unit/VariableList.varSTRING	This is

Figure 6-151 UAExpert

6.4 SIMOTION IT Virtual Machine and Servlets

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P

- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

6.4.1 Fundamental safety instructions

6.4.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

6.4.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

6.4.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

6.4.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

6.4.2 Introduction

6.4.2.1 Overview of SIMOTION IT

Overview of SIMOTION IT manuals

The "SIMOTION IT Ethernet-based HMI and diagnostic functions" are described in four manuals (IT = Information Technology):

- **SIMOTION IT Diagnostics and Configuration**
  This manual describes the direct diagnosis of SIMOTION devices. Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.
  See the SIMOTION IT Diagnostics and Configuration Manual.
- **SIMOTION IT Programming and Web Services**
  This manual describes the creation of user-defined Web pages and access to the diagnostic functions via the two Web services provided by SIMOTION IT.
  A Web service enables users to create their own client applications in any programming language. These applications then communicate with the SIMOTION device using Web technologies. The SOAP (Simple Object Access Protocol) communication protocol is used for transferring commands.
  The manual includes information on programming such clients, as well as a description of the SIMOTION IT Web services (OPC XML-DA, Trace via SOAP TVS) via which data and operating states of the controller can be accessed and the variable trace functions can be used.
  See the SIMOTION IT Programming and Web Services Manual.
- **SIMOTION IT Virtual Machine and Servlets**
  This manual describes the Java-based function packages. The Jamaica Virtual Machine (JamaicaVM) is a runtime environment for Java applications on the SIMOTION device. It is an implementation of the "Java Virtual Machine Specification".
  The Servlets section of the manual describes the use of servlets in a SIMOTION device.
- **SIMOTION IT OPC UA**
  The manual describes access to SIMOTION devices via OPC UA.

See also

PDF in the Internet: SIMOTION IT Diagnostics and Configuration (https://support.industry.siemens.com/cs/ww/en/view/109476533)

PDF in the Internet: SIMOTION IT Programming and Web Services (https://support.industry.siemens.com/cs/ww/en/view/109476528)

6.4.2.2 SIMOTION IT Virtual Machine

The Jamaica Virtual Machine (JamaicaVM) is an implementation of the "Java Virtual Machine Specification", and is a runtime system for executing Java applications.

This document describes how the SIMOTION-integrated JamaicaVM is parameterized, and how a Java user program is commissioned.

It also contains essential information on how to use the SIMOTION API, which contains classes for data access and logging. More detailed information about the SIMOTION API exists in HTML format.

The SIMOTION-integrated JamaicaVM will only be started automatically if the configuration file VMCONFIG.INI is on the target system (see also VMCONFIG.INI (Page 3945) and Installation (Page 3912)).

The Java main program ("InvocationManager"), which is also contained in the SIMOTION API, is executed by default. The InvocationManager reads the settings from the relevant configuration file and then starts the Java user programs (see VMCONFIG.INI (Page 3945)).

The InvocationManager can also open a TCP socket interface, which can be used to load and execute user programs interactively (see Interactive InvocationManager interface (Page 3951)).

Program and system variables of the target system, for example, can be accessed in the user programs. Network support (DatagramSocket, ServerSocket, Socket, ...) and file functions are also available.

6.4.2.3 SIMOTION IT Servlets

SIMOTION IT Servlets are the implementation of a servlet container.

Servlets are programs written in Java, which extend the capabilities of servers that communicate via a request/response protocol. Theoretically, they are not bound to a server protocol (HTTP), but they are usually used only in conjunction with HTTP/web servers.

The Web container provides the runtime environment for Web applications with its servlets, and it can run on a different machine to the Web server (load distribution, safety/redundancy).

In a SIMOTION context, servlets run in SIMOTION IT Virtual Machine (implementation of a Java runtime environment) and are, therefore, afforded the option of using the SIMOTION API (giving access to SIMOTION RT, NVRAM, and logging objects).

The following are described in the relevant chapters

- Implemented scope of functions (Page 3958)
- Parameterization of the servlet container (Page 3960)
- Start-up of a servlet using an example (Page 3965).

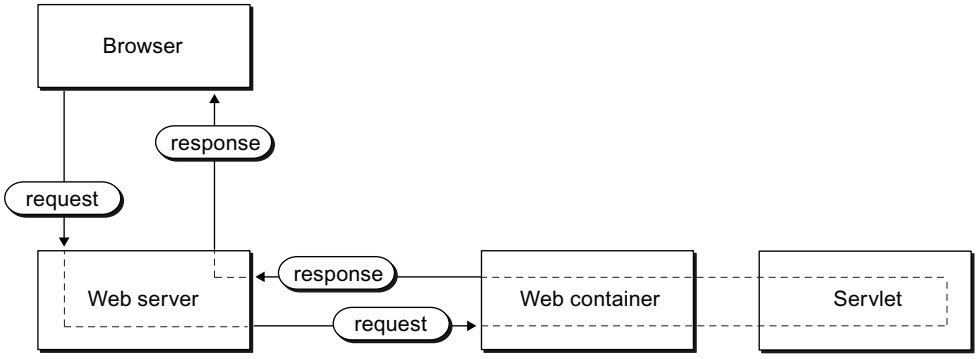


Figure 6-152 Servlet command sequence

6.4.3 Software installation

6.4.3.1 SIMOTION IT Virtual Machine

The JamaicaVM is only started if configuration file VMCONFIG.INI is available on the target system.

A configuration file that contains the directory settings specified in the recommendations referred to above, and which can be used to activate the JamaicaVM, is included in the SIMOTION IT Virtual Machine scope of delivery.

To make special settings in the configuration file, see also: JamaicaVM configuration (VMCONFIG.INI). (Page 3945)

Configuration file VMCONFIG.INI and Java library SIMOTION.JAR contain the InvocationManager and the API for data access and logging. These must be stored on the target system. The default setting dictates that the Java library is expected to be in the root directory of drive B: However, the root directory can also be set differently if required (see BOOTCLASSPATH (Page 3945)).

User programs must be zipped into a Java library (JAR or ZIP file) and stored in the current working directory of the target system.

On the target system, the default current working directory is the root directory of drive B:, but this can be changed if required (see CURRENTDIR (Page 3945)).

To start a user program automatically, all you need to do is enter the relevant information in configuration file JINVOKE.XML. This configuration file must also be stored in the current working directory of the target system.

6.4.3.2 SIMOTION IT Servlets

Installation and configuration of SIMOTION IT Virtual Machine is a requirement for SIMOTION servlets.

To operate SIMOTION servlets, Java libraries svltapi.jar and svltimpl.jar must be stored on the target system.

In the case of an empty CF card, these Java libraries must be stored in the following directory as of SIMOTION V4.1 SP4

- /SIEMENS/SIMOTION

The configuration files supplied assume that the SIMOTION Java system libraries are located in this directory.

meaning that Java library svltapi.jar, which contains the standard servlet API, needs to be incorporated into the JamaicaVM path.

The most straightforward way of doing this is to expand the setting for the BOOTCLASSPATH accordingly in VMCONFIG.INI:

BOOTCLASSPATH= b:/siemens/simotion/simotion.jar;b:/siemens/simotion/svltapi.jar

Note

In SIMOTION IT Virtual Machine, the semicolon must always be specified as a CLASSPATH separator.

To start the servlet container automatically, you need to insert the following line in JINVOKE.XML:

<AUTOSTART LIBRARY="B:/SIEMENS/SIMOTION/SVLTIMPL.JAR" CLASSNAME="com.siemens.ad.SIMOTION.servlets.ServletContainer"/>

User servlets must be zipped into a Java library (JAR or ZIP file) and stored in the current working directory of the target system.

To load a user servlet, all you need to do is enter the relevant information in configuration file JSERVER.XML. This configuration file must also be stored in the current working directory of the target system.

Note

The software/libraries are available on the U&A DVD at \VOL2\AddOn\4_Accessories\SIMOTION_IT\5_SIMOTION_VM

6.4.4 Software programming

6.4.4.1 SIMOTION IT Virtual Machine

Scope of delivery

The following files are integral parts of SIMOTION IT Virtual Machine:

File name	Description
SIMOTION.JAR	Java library containing the SIMOTION API and the default main program (InvocationManager).
SIMOTION_DOC.ZIP	Online SIMOTION API documentation (HTML format).
SIMOTION_DEMO.ZIP	Java source code for sample programs and tools.
JINVOKE.XML	Sample AUTOSTART configuration file
VMCONFIG.INI	JamaicaVM configuration file
Jam1.st	Sample ST program that defines and exports ST program variables.
nanoxml.jar	Java library that is required to start the InvocationManager program on a development computer.
simotion_systemclasses.jar	Java library that can be used as a reference for compilation on a development computer.

Supported Java packages

In principle, the following packages from JDK 6 are supported by the JamaicaVM integrated in SIMOTION.

- java.io
- java.lang
- java.lang.reflect
- java.math
- java.net
- java.text
- java.util
- java.util.jar
- java.util.zip

Additional Java libraries can be integrated into the user code by importing.

Default directories

SIMOTION V4.1. SP4 includes some important new features in terms of both the directories recommended for SIMOTION IT Virtual Machine and the Java programs based on these.

In particular, specific default directories for Java program files and Java data files are recommended so that the new CPU update feature can be used for SIMOTION IT Virtual Machine as well.

All of the configuration files and sample programs supplied with SIMOTION V4.1. SP4 and higher are compatible with these new features.

The recommended default directories are specified as follows:

- Default directory for Java program files and SIMOTION IT Virtual Machine configuration file JINVOKE.XML: /USER/SIMOTION/HMI/JAVA/PROG/
- Default directory for Java data files: /USER/SIMOTION/HMI/JAVA/FSROOT/

For reasons of compatibility, the default values for the relevant settings have not been changed, meaning that you can continue to use existing configurations/installations.

However, these recommended default directories are used for the relevant settings in the configuration files supplied.

In particular, the /USER/SIMOTION/HMI/JAVA/PROG/ directory is set as a value for CURRENTDIR in the VMCONFIG.INI file supplied; CURRENTDIR denotes the current working directory and is, therefore, the default directory for Java program files, for example.

SIMOTION.JAR

As of SIMOTION V4.1 SP4, the Java library SIMOTION.JAR must be stored in the following directory:

- /SIEMENS/SIMOTION

The configuration files supplied assume that the SIMOTION Java system libraries are located in this directory.

Since old SIMOTION Java system libraries can, in principle, be used in newer versions of SIMOTION (without the additional features of these newer versions, of course), it is also possible to continue using existing configurations/installations without making any changes to them.

However, in such cases it is strongly recommended that you adapt the configuration of SIMOTION IT Virtual Machine too, so that the /SIEMENS/SIMOTION directory is used for these Java libraries (see BOOTCLASSPATH (Page 3945)).

Only in environments that are adapted in this way is it possible to automatically use the SIMOTION Java libraries that are compatible with the relevant firmware version during the course of a CPU update.

Restrictions and special features

Overview

Limited resources (memory) and special implementations (file system, network) result in some restrictions and special features being present on the target system. These are described in the section that follows.

Number of threads

The required number of Java user threads can be set (see SIMOTIONVM_NUMUSERTHREADS (Page 3945)).

If user programs request more threads than are actually available, an error message is written to the output file (see Output file JCONSOLE.TXT (Page 3949)).

Network functions

No name service is available on the target system. For this reason, only the "raw" IP addresses can be used and not symbolic host names.

Examples:

- 157.163.237.50 instead of r1014.erlf.siemens.de.
- Especially also 127.0.0.1 instead of localhost.

The `get-/setSoTimeout` functions, for the socket functions' time watchdog, are not implemented. Consequently, it is not possible to apply a timeout to the process of receiving data via a socket connection, for example.

File system

The file system on the target system has the following features:

- All file names – with the exception of the target file when renaming files – must contain a drive letter (usually A: or B:) .
- A slash (/) is used as a file separator.
- No distinction is made between upper- and lower-case letters.

However, to write independent user programs from the platform, the file names must not be specially initialized for the target system (e.g. B:/DEMO/MYFILE.TXT).

Instead, a file name should be created dynamically (e.g. from the current working directory, the separator to be used, and the actual file name).

It is possible, for example, to test programs on a development computer this way.

Example:

```
// Retrieve the current working directory
String currDir = System.getProperty("user.home");
// Compile the file name
File f;
f = new file(currDir, "DEMO" + File.separator + "MYFILE.TXT");
```

The `get-/setLastModified` functions, for handling file time stamps, are not implemented.

Synchronization with the PLC

The JamaicaVM continues running even when the PLC is in the STOP state. It is possible to synchronize with the control section by accessing the current operating state of the PLC. However, no special notifications are given if the operating state changes.

Effects on the target system and user programs

It goes without saying that using the JamaicaVM will have consequences for the entire system.

Depending on the application case, approx. 3 MB of RAM are required to operate the JamaicaVM. This may make it impossible to load any larger user programs. As of SIMOTION V4.1 SP1, the memory required for the JamaicaVM is no longer provided by the USER memory, which means that operating the JamaicaVM no longer creates a drain on the memory resources available for user programs.

The individual Java threads are tied to their corresponding SIMOTION tasks on a round-robin level. This means that any other SIMOTION tasks on this level will have correspondingly less computing time.

Connecting to SIMOTION SCOUT

The JamaicaVM is not connected to SIMOTION SCOUT.

SIMOTION SCOUT merely indicates a missing license or a serious error when the JamaicaVM starts up by means of a corresponding entry in the diagnostics buffer.

Debugging Java programs

With SIMOTION V4.2 and higher, remote debugging of Java programs is possible on the target system.

A network connection is always used for communication between the debugger and JamaicaVM. The target system is configured as standard as the debugging server. For this purpose, the JamaicaVM must be made to open a remote debugging port by means of parameterization.

To connect with a debugger, a network connection to the target system is thus necessary. In addition, the debugger must be activated by means of the relevant specifications in VMCONFIG.INI.

Eclipse or also jdb (the Java debugger of Sun Microsystems Inc.) can be used, for example, as the debugger frontend.

Example of connecting with the jdb, specifying the IP address and port number:

```
jdb -connect
com.sun.jdi.SocketAttach:hostname=192.168.214.1,port=8000
```

For the setting of breakpoints to function correctly, it is extremely important that the Java application loaded in the target system and the Java source files used match precisely. The Java source files are used for determining breakpoints. If, for example, a Java source file that has been changed in relation to the Java application loaded in the target system is used by mistake, it can happen that the debugger simply does not stop at the breakpoint. This behavior arises because the relevant line number in the Java source file within the loaded Java application does not describe an executable program line.

However, the SIMOTION API can be operated in simulation mode for the software development phase. In this way, the application can be tested and debugged on the development computer as an initial step (basic program execution). See DATA_ACCESS element (Page 3944)

Debugging with Eclipse

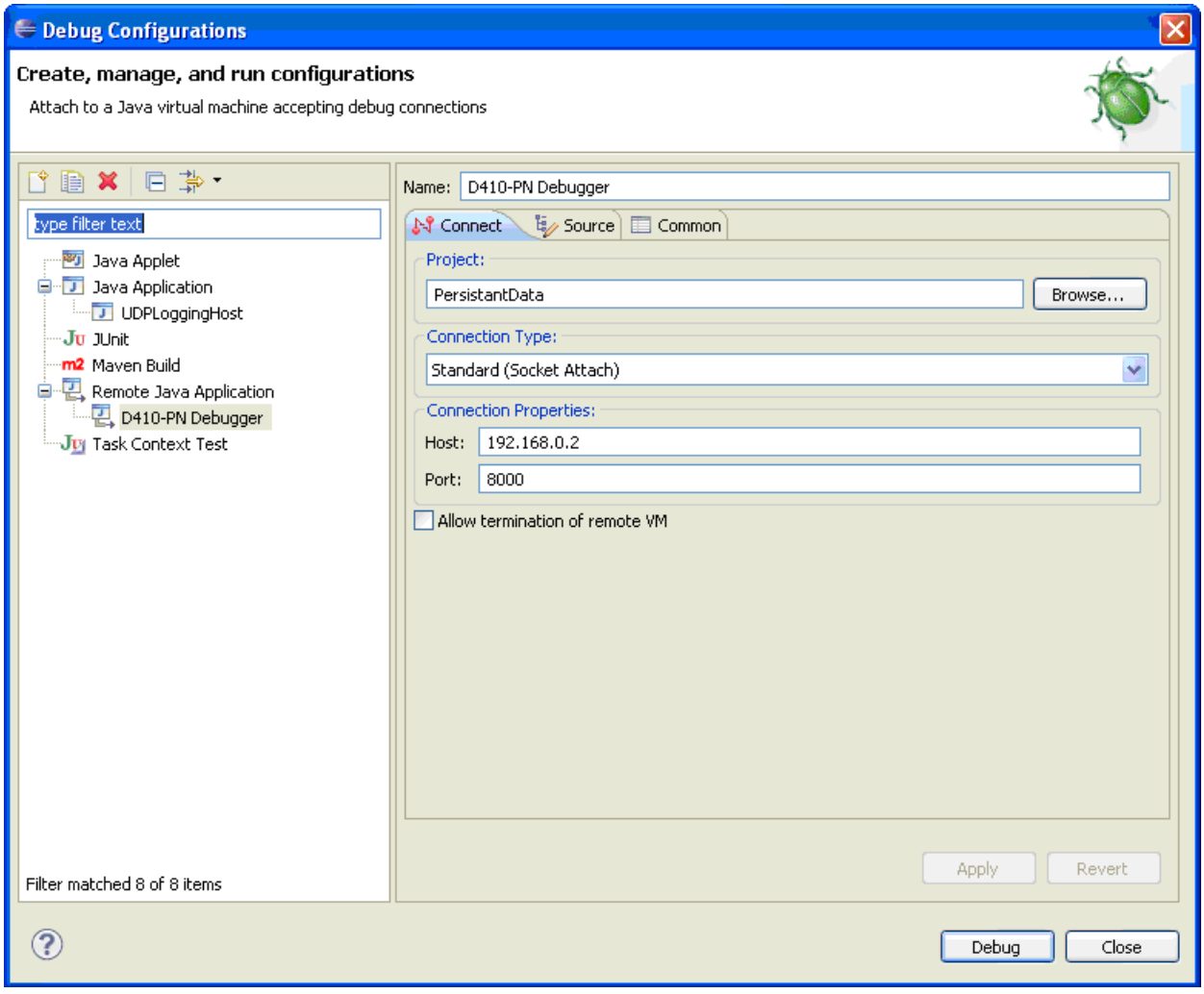


Figure 6-153 Example of remote debugging settings in Eclipse

See also

VMCONFIG.INI (Page 3945)

SIMOTION API

Overview

The SIMOTION API contains Java classes. These allow you to:

- Access runtime system variables, including the option of marshalling (variable values can be set on the basis of byte arrays and written to byte arrays)
- Call system functions of a technology object (TO)

- Access the parameters, parameter description, and error buffer of a drive
- Output logging information

Variable access

Overview

Relevant Java package:

- com.siemens.ad.SIMOTION.da

The "object server" (RTObjectServer), acting as the central class, and classes for the type-safe representation of variables are available for the purpose of accessing data on the runtime system. There is a corresponding Java class for each ST standard data type (RTVariableBOOL, etc.).

The variable provider of the runtime system supports the conversion of elementary data types to strings and vice versa. This means, all variables of the runtime system can also be accessed using a "string" variable.

Table 6-60 Mapping elementary data types to variable classes

ST standard data type	Java variable class
BOOL	RTVariableBOOL
SINT	RTVariableSINT
USINT, BYTE	RTVariableUSINT
INT	RTVariableINT
UINT, WORD	RTVariableUINT
DINT	RTVariableDINT
UDINT, DWORD	RTVariableUDINT
REAL	RTVariableREAL
LREAL	RTVariableLREAL
TIME	RTVariableSTRING
DATE	RTVariableSTRING
TIME_OF_DAY (abbreviation: TOD)	RTVariableSTRING
DATE_AND_TIME (abbreviation: DT)	RTVariableSTRING
STRING	RTVariableSTRING
Any ST standard data type	RTVariableANY

Regardless of this, any Java RTVariable (except for RTVariableANY) can provide its value in Java string format, or set it according to a specified string.

RTVariableANY is a special case. This type of variable can be used to read the native data type and content of a runtime system variable (but only in the form of "raw bytes"). However, the runtime system variable must be of one of the elementary data types listed in the table above.

After an RTVariableANY variable has been successfully read, a corresponding typesafe variable can be created using the object server.

The object server has the following tasks:

- Creating variables
- Reading/writing variables or variable groups
- Deleting variables
- Browsing variables
- Reading variable properties

The variables have the following tasks:

- Storing the variable value
- Type-safe reading/setting of the variable value
- Converting the variable value from/to string format

Converting the variable value from/to byte array format (marshaling)

Accessing the configuration data of a TO

Accessing the configuration data of a TO enables you to configure an axis, or even to determine the logical base address of the drive assigned to the axis.

Please note that there are several types of configuration data:

- Active configuration (current value)
- Configuration that may have changed but is not yet active (next value)

It is possible to access both the currently active configuration and the configuration that is not yet active. In SIMOTION SCOUT, these data sets are referred to as "current value" and "next value" respectively.

The specified variable name is used to determine which data set is to be accessed (see also Variable names (Page 3920)).

The parameters that are available depend on the TO type and, in the case of an axis, for example, on the type of axis. In addition to this, specific parameter settings demonstrate interdependencies (these are implemented in the SCOUT by integrating the test routines created during generation). There is no special support for access via Java in this case.

Please also note the various options for changing configuration data:

- Download (can no longer be changed)
- Restart (the TO must be restarted to make the change effective)
- Immediate (the change is effective immediately).

Reading the variable properties allows you to determine whether a configuration variable can be changed, and whether it will be necessary to restart the TO following a change (see also Reading variable properties (Page 3923)).

Variable names

No distinction is made between upper- and lower-case letters when it comes to variable names.

They consist of a prefix and the name of the variable itself.

For device system variables, the prefix is "var/". For variables from a source (ST/MCC/LAD/FBD), the prefix is "unit/".

Only the variables declared in the interface are available.

When accessing system or configuration variables of a TO, the following rules apply:

- The prefix for TO system variables is: to/<TOname>.
- The prefix for TO configuration variables is: cfg/<TO-Name>.
  When accessing the current configuration variables, activeConfigData must also be specified after the TO name. For accessing the configuration to be adopted, which is designated in SIMOTION SCOUT as the Next value, setConfigData must be specified after the TO name.

Table 6-61 Accessible variables (variable name structure and example):

Variable type	Variable name structure	Example
Device system variables	var/	var/userData.user1
TO system variables	to/<TOname>.<Variable>	to/axis_1.basicMotion.position
TO configuration variables	cfg/<TOname>.activeConfigData.<Variable>	cfg/axis_1.activeConfigData.Modulo.state
TO configuration variables	cfg/<TOname>.setConfigData.<Variable>	cfg/axis_1.setConfigData.Modulo.state
Global device variables	glob/ <Variable>	Glob/myGlobalVar1
IO variables	io/_direct .<Variable> io/_image .<Variable> io/_quality .<Variable>	io/_direct.myOut1 io/_image.myOut1 io/_quality.myOut1
Program interface variables	unit/<Unitname>.<Variable>	unit/awp01.ug_stringvar1

Access to I/O variables has been possible since Version V4.2. Access to the diagnostics buffer is currently not possible.

Access to SIMOTION variables

```
// Create the object server
RTObjectServer myServer = RTObjectServer.createRTObjectServer();

RTVariableUDINT myVar = zero;

// Create an RTVariable to access a SIMOTION variable
myVar = myServer.createRTVariableUDINT(
                        "unit/delay.ug_udruntimedelay" );

// Read the variables (the SIMOTION variable value is retrieved in the
// RTVariable).
myServer.read( myVar );

// Read the variable value
long myValue = myVar.getValue();

// Convert to string format
// Any RTVariable can supply its value in string format
String: myValueAsString = myVar.getAsString();
```

```
// Marshalling
Byte [] myBuffer = new byte[20];
int myIndex = 2;
boolean bAsBigEndian = true;

// Any RTVariable can write its value to a byte array
//(beginning with the index specified); option of
// Big or Little Endian format
myVar.getValue( myBuffer, myIndex, bAsBigEndian );

// Delete the RTVariable
// Important for re-enabling the
// management information created within the variable providers.
myServer.delete( myVar );
```

It is also possible to access a group of RTVariables simultaneously:

```
RTVariable [] myVars = new RTVariable[2];
myVars[0] = myServer.createRTVariableUDINT(
                        "unit/delay.ug_udruntimedelay" );
myVars[1] = myServer.createRTVariableREAL(
                        "unit/delay.ug_rvalue" );

myServer.read( myVars );
```

Example: Using RTVariableANY

```
// Create the object server
RTObjectServer myServer = RTObjectServer.createRTObjectServer();

RTVariableANY v = zero;
String myStringValue = "";

try
{
    // Use RTVariableANY
    v = myServer.createRTVariableANY("to/axis_1.error" );

    // Read the variables (reads type)
    myServer.read( v );

    // Information about the data type read
    short vDatatype = v.getDatatype();

    // Use the information read to create a
    // typesafe variable
    RTVariable v2 = myServer.createRTVariable( v );

    // Read the variables (reads value)
    myServer.read( v2 );
```

```
    // Following this, a value can be assigned to the variables
    // in string form.
    v2.setFromString(myStringValue);
}
catch( DataAccessException e )
{
    …
}
```

Browsing variables

The object server offers the option of browsing variables. Browsing means that all the variable names will be listed starting from a particular level (as specified by a search pattern).

Browsing is particularly worthwhile in applications where it is necessary to determine dynamically which technology objects are present in the runtime environment, so that these can subsequently be accessed (e.g. for the purpose of reading out the error variable of each technology object).

Example: Browsing variables

```
// Use the RTObjectServer to list all
// technology objects
String [] vTOs = mServer.listVariables( "to/*" );
```

Reading variable properties

The `readProperties` function is available for the purpose of reading variable properties.

Reading variable properties is particularly worthwhile in generic applications.

It enables you to determine during runtime whether a variable may only contain special values of an enumeration, for example. For this kind of enumeration variable, a list containing all the permitted values (the enumeration itself) is also provided in the variable properties. This can then be used, for example, to construct a drop-down list box containing these values.

In the case of configuration variables, the "effectiveness" of a value change is also indicated in the variable properties. In other words, a program can establish whether a value can ever be changed during runtime, and whether it will be necessary to restart the technology object following a value change.

The `readProperties` function delivers the variable properties in an object within the `RTProperties` class.

The `RTProperties` class has the following tasks:

- Defining the names for the individual properties

Currently, the following property names have been defined in `RTProperties`:

- DATA_TYPE — Data type
- VALUE — Current value
- ACCESS_RIGHTS — Access rights

- ENUMERATION_TYPE — Enumeration type
- ENUMERATION_INFO — The enumeration itself (if enumeration type)
- EFFECTIVENESS — Effectiveness

Not all properties are available for all variables. For instance, ST program variables do not have the EFFECTIVENESS property.

Please also note that enumeration variables of an ST program are mapped internally as straightforward DINT variables, meaning that the list of permitted values is not available for these ST program enumeration variables.

The variable properties also contain the current value for the variable at hand. However, particularly where cyclic access to the variable value is concerned, a corresponding `RTVariable` should be used for reasons of performance.

Example: Reading variable properties

```
// Use the RTObjectServer to read the properties of
// the "var/traceState[0].fctGenState" variable
RTProperties myProperties =
    mServer.readProperties("var/traceState[0].fctGenState" );

// Retrieve the enumeration type
String sEnum =
    myProperties.getProperty( RTProperties.ENUMERATION_TYPE );

// Establish whether the variable may only contain values
// of an enumeration
if( "enumerated".equals( sEnum ) )
{
    // Continue by, for example, retrieving the enumeration itself,
    // i.e. ENUMERATION_INFO
    ...
}
```

Accessing drive data

Relevant Java package:

- com.siemens.ad.SIMOTION.sys

The `RTDrive` class is available for accessing the drive data of a runtime system. Each drive is identified by its logical base address, which must be specified when creating an `RTDrive` object.

The `RTDrive` class has the following tasks:

- Reading/writing drive parameters
- Reading the description of drive parameters
- Reading the error memory of a drive

The `RTDriveParameter` class is used to access the drive parameters themselves. This class stores information on which parameter is to be accessed (parameter number; parameter index for array parameter where applicable), and transports the actual parameter values.

A Java `double` is always used for setting/reading the value within an `RTDriveParameter` object, which makes it easy to create generic applications (regardless of the actual data type of a parameter, its value can be easily read and displayed, e.g. for diagnostic purposes).

If an application requires type-safe values (e.g. for calculations), these values can easily be converted to the right data type:

byte vValue = (byte)param.getValue();

The `RTDriveParameter` class has the following tasks:

- Storing the information on the parameter that is to be accessed
- Storing the parameter value
- Storing the data type of the parameter
- Reading/setting the parameter value

The parameter data type can only be detected following the first successful access procedure (read access or write access). If the parameter has not yet been accessed, the stored data type corresponds to the value `RTDatatypes.DT_UNKNOWN`.

The `RTDriveParameterDescription` class is used to determine drive parameter properties. Reading drive parameter properties allows you to determine, for example, the actual data type, or information on whether you are dealing with an array parameter or not (incl. the array length).

The `RTDriveParameterDescription` class has the following tasks:

- Storing information about the drive parameter
- Reading parameter information

If any errors occur when accessing drive parameters, the special exception `DriveAccessFailedException` will be triggered.

The `DriveAccessFailedException` class has the following tasks:

- Storing the parameter numbers
- Storing the error code of the access function
- Storing the parameter-specific error code

This information can be read out from the relevant exception object when a `DriveAccessFailedException` occurs.

Refer to the SIMOTION documentation (e.g. SIMOTION SCOUT online help) for the meanings of the error codes.

Example: Accessing drive data

```
Try
{
    // Example of a logical base address of a drive,
    // depends on the SIMOTION project configuration.
    int vLogDriveAddress = 256;

    // Create the RTDrive object
    RTDrive vDrive   = new RTDrive( vLogDriveAddress );
```

```
    // Example of a parameter number; describes the:
    // "Drive state" parameter of a SINAMICS G110
    long vParameterNumber = 2;

    // Create the parameter
    RTDriveParameter vParameter =
                      new RTDriveParameter( vParameterNumber );

    // Read the parameter
    vDrive.read( vParameter );

    // Check the "Drive ready" state of the drive
    if( (int)vParameter.getValue() == 1 )
    {
        ...
    }
}
catch( DriveAccessFailedException e )
{
    // Access to parameter failed
    ...
}
catch( DataAccessException e )
{
    // Another error,
    // e.g. BadArgumentException...
    ...
}
```

Accessing target system functions

Overview

Relevant Java package:

- com.siemens.ad.SIMOTION.sys

The `RTDevice` class is available for accessing target system functions.

The `RTDevice` class has the following tasks:

- Reading/setting the current operating mode
- Copying actual data to RAM
- Copying RAM to ROM
- Unzipping a SIMOTION archive file (e.g. project archive)
- Restarting the target system
- Reading version information

When target system functions are called, in addition to purely technical causes of errors (such as a "System function failed" call), other causes of errors such as "Function cannot be executed in current state" are possible.

So that errors occurring as a result of purely technical problems can be distinguished from others, special exceptions of type `DeviceFunctionFailedException` are triggered in these cases.

The `DeviceFunctionFailedException` class does not have any other tasks, nor does it transport any other special information apart from, of course, the error message, which is as detailed as possible and is included in all exceptions.

The majority of situations causing errors can be determined. An example is when there is not enough space on the memory medium for copying from RAM to ROM:
`Device function failed with result: DISK_FULL`

If it is not possible to set the operating state due to the setting of the mode switch on the target system:
`Device function failed with result: NOT_ALLOWED`

However, the exact cause of an error cannot be determined in all situations. For example, it is not possible to distinguish between a situation where copying from RAM to ROM cannot be executed because the target system is not in the STOP mode, and one where the SCOUT or an ST program has already initiated the function at the same time. In this kind of case, the `DeviceFunctionFailedException` would only contain general information: `Device Function execution failed`.

Accessing the target system operating state

The `getOperatingState/setOperatingState` functions are available for reading/ setting purposes.

The operating state of the target system is returned as a string in this way. When writing, the operating state must also be transferred as a string.

The operating states that are currently supported are:

- `RUN`
- `STOPU`
- `STOP`
- `SERVICE`

Comment:

When the current operating state is being read, values that deviate from the currently supported operating states may be provided. The reason for this is that the system adopts various interim states during the course of changing its operating state.

The SERVICE target system operating state is not currently supported for writing operations.

The target system's state selector switch is crucial when it comes to selecting the target system operating state. The program cannot set the operating state if the target system operating state is STOP.

Refer to the SIMOTION SCOUT documentation for the exact meanings of the individual operating states.

Example: Accessing the target system operating state

In the following example, the current operating state is queried. If the target system is not in STOP, it will be stopped.

```
try
{
    //  Create object for accessing
    RTDevice vDevice = RTDevice.createRTDevice();

    // Query the current operating state
    String: vCurrentState = vDevice.getOperatingState();

    // If the target system is not
    // in the STOP operating state...
    if( !RTDevice.STOP.equals( vCurrentState ) )
    {
        // ... Stop
        vDevice.setOperatingState( RTDevice.STOP );
    }

    // If the program gets to this point,
    // the target system is in the STOP state
}
catch( DataAccessException e )
{
    // Access to operating state failed
}
```

Copying actual data to RAM/from RAM to ROM

The `copyActualToRam/copyRamToRom` functions are available for the purpose of "copying actual data to RAM"/"copying from RAM to ROM".

Example: Copying actual data to RAM/from RAM to ROM

In the example below, the actual data is copied to RAM first, and then from RAM to ROM.

```
try
{
    //  Create object for accessing
    RTDevice vDevice = RTDevice.createRTDevice();

    // Copy actual data...
    vDevice.copyActualToRam();

    // Copy from RAM to ROM...
    vDevice.copyRamToRom();
}
catch( DeviceFunctionFailedException e )
{
    // Function could not be executed,
    // e.g. Device function failed with result: DISK_FULL
```

```
}
catch( DataAccessException e )
{
    // Another error has occurred
}
```

Unzipping a SIMOTION archive file (e.g. project archive)

The `unpackArchive` function is available for unzipping a SIMOTION archive file.

This system function can be used to unzip a SIMOTION archive file, such as a user project that has been properly stored using SIMOTION SCOUT.

It must be noted that the target system must be in the STOP operating state to allow unzipping of the archive file.

Following successful unzipping, both the temporary work directory and the archive file are deleted by the system function.

The modified data does not become active until the target system is restarted.

Restarting the target system

The `restart` function is available for restarting the target system.

This function restarts (resets) the target system.

It is necessary to restart the target system once a user project has been successfully unzipped, for example.

Example: Unzipping a user project and restarting the target system

In the example below, the user project archive file is unzipped, and the target system is then restarted so that the new user project can be activated.

```
File myPrjFile = new file( System.getProperty("user.dir"),
                           "MYPROJ.ZIP" );

// Create work directory
// This is necessary because the system function removes the
// specified work directory after unzipping of the archive file
File workDir = new File( System.getProperty("java.io.tmpdir"),
"JVM_PRJUPD" );
if( !workDir.isDirectory() )
{
    if( !workDir.mkdir() )
    {
        // Creation of work directory has failed
        return;
    }
}

try
{
    //  Create object for accessing
```

```
    RTDevice vDevice = RTDevice.createRTDevice();

    // Unzip the archive file
    vDevice.unpackArchive(
                          workDir.getCanonicalPath(),
                          myPrjFile.getCanonicalPath()
                         );

    try
    {
        // Caution: If successful, it will not be possible to undo
this call!
        vDevice.restart ();
    }
    catch( OperationFailedException e )
    {
        // Restart failed...
    }
}
catch( Exception e )
{
    // Unzipping the archive has failed;
    // there may not be a correct SIMOTION user project archive
}
```

Reading version information

The `getFirmwareVersion/getDriveVersion` functions are available for the purpose of reading version information.

The `getFirmwareVersion` function provides the version number of the SIMOTION firmware.

The `getDriveVersion` function provides the version number of the integrated drive. On target systems without an integrated drive, a corresponding exception will be triggered.

Example: Reading version information

In the example below, the version numbers of the SIMOTION firmware and the integrated drive are read. The firmware version number is read first, as the access to the integrated drive that follows may trigger an exception (on target systems without an integrated drive).

```
String v1 = "n/a";
String v2 = "n/a";

 try
{
    //  Create object for accessing
    RTDevice vDevice = RTDevice.createRTDevice();

          short [] v;

          v = vDevice.getFirmwareVersion();
          v1 = "V " + v[0] + "." + v[1]+ "." + v[2]+ "." + v[3];
```

```
            v = vDevice.getDriveVersion();
            v2 = "V " + v[0] + "." + v[1]+ "." + v[2]+ "." + v[3];
}
catch( OperationFailedException e )
{
    // Reading of the version failed;
    // possibly target system without integrated drive
}
```

Additional functions

The following additional functions are available:

- `getName`
- `copyFile`

The `getName` function provides the name assigned to the target system within the SIMOTION user project.

The `copyFile` function copies an existing file with the aid of the relevant file system function. Particularly in the case of large files, using this function may represent a much faster option than using a straightforward file copying function implemented in Java.

Calling system functions

Overview

Relevant Java package:

- com.siemens.ad.SIMOTION.sys

The `RTSystemFunctions` class is available for the purpose of calling system functions of a technology object (TO).

The `RTSystemFunctions` class has the following tasks:

- Creating a command ID
- Calling the system functions of a TO
- Reading the type UID of a TO

Creating a command ID

Most system functions expect a command ID as a parameter. The `getCommandId` method provides an object from the `RTCommandId` class as a return value.

Please note that a command ID consists of two values (internally): Id_low and Id_high. If the method used to call a system function is the one that expects the input parameters to be received in the form of an object[], the two values must be transferred separately (Id_low first).

The `RTCommandId` class has the following tasks:

- Storing a unique command ID
- Reading the two internal command ID values, so that they can be specified as parameters for calling a system function

Calling the system functions of a technology object

The `callSystemFunction` method is a general function that can be used to call a system function of a TO. The method receives the name of the TO as a parameter, the name of the system function to be called, the input parameters of the system function, and dummies for the output parameters of the system function called (incl. the number of output parameters).

The current system function can always be found by specifying the function name. If a new version of SIMOTION includes any changes to the signature or semantics of a system function, this must be taken into account in the user program.

Currently, it is possible to call the following system functions:

- _move
- _getStateOfAxisCommand
- _getAxisErrorNumberState
- _stopEmergency
- _stop
- _enableQFAxis
- _resetAxis
- _resetAxisError
- _disableAxis
- _disableQFAxis
- _continue
- _getAxisErrorState
- _enableAxis
- _pos
- _homing
- _ getIPConfig
- _ setIPConfig

Important note:

Please note that, for the majority of system functions, the name of the TO for which the system function is to be executed must be specified again within the input parameters. The object is usually the first input parameter of a system function. Here, please note that this parameter describes another TO, which may not even be available. However, this is not a mistake. Possible consequences of this may be the execution of functions for another TO, or error situations such

as Function not found" (if this parameter describes an existing TO that is not supported by the system function on account of the TO type), or even "TO not found".

Two versions of the `callSystemFunction` method are available.

Since various parameter types must be transported, and the number and sequence of the input parameters also depends on the system function that has actually been called, each input parameter is passed as either a straightforward `Object[]` or an `ArrayList`, depending on the version.

The `java.lang.Long` and `java.lang.String` object types are supported by both versions as input parameter types. Numerical parameters must be stored in a long object, with floating-point numbers converted accordingly (`doubleToLongBits` method of `java.lang.Double`). TO names should be passed as string-format parameters.

The method that expects the input parameters in `ArrayList` format is more user-friendly; as well as this, it supports `java.lang.Double` and `RTCommandId` parameters. However, due to the type monitoring that is necessary during runtime and the parameter conversions, it is considerably slower than the version that expects the input parameters in `Object[]` format.

A command ID is required for executing system functions. This command ID must be created anew for each call, and passed in the input parameters accordingly.

Refer to the SIMOTION documentation (e.g. SIMOTION SCOUT online help) for the number and sequence of input parameters in a system function. Precisely this number of input parameters must be stored in the passed parameter container (Array or ArrayList), in precisely this sequence. In other words, the container with the input parameters must not contain any additional elements.

This method provides the call result in the form of a return value (`ErrorCodes.RT_ERR_IS_OK` if it was possible to execute the system function call; otherwise, one of the predefined `ErrorCodes`).

Here, potential errors include, in particular:
`ErrorCodes.RT_ERR_SYSFCT_TO_NOT_FOUND:`

The TO could not be found. A project that contains the TO with the specified name must be loaded on the target system.
`ErrorCodes. RT_ERR_SYSFCT_FCT_NOT_FOUND:`

The function could not be found. For example, a straightforward speed-controlled axis does not support the "_pos" system function.

`ErrorCodes. RT_ERR_SYSFCT_MISSING_INPUT_PARAM:`

Too few input parameters were specified.
`ErrorCodes. RT_ERR_SYSFCT_CANT_WRITE_OUTPUT_PARAM:`

The dummy array for the output parameters of the system function is too small.

The actual return value of the system function is always located in the output parameters of the system function.

An `Object[]` must be specified as a dummy for the output parameters. The length of the array must at least equal the number of output parameters of the actual system function, so that all output parameters can be accommodated. The individual array elements do not, however, have to be initialized. Once the system function has been successfully called, an object corresponding to each output parameter is created; here, only the `java.lang.Long` and `java.lang.String` object types are supported.

Refer to the SIMOTION documentation (e.g. SIMOTION SCOUT online help) for the number and sequence of output parameters in a system function.

Reading the type UID of a technology object

Currently, the actual type of a TO (drive axis, position axis, synchronized axis, etc.) can be queried.

The `RTUID` class has the following tasks:

- Storing a unique UID for the runtime system
- Predefining recognized type UIDs (e.g. `RTUID.POS_AXIS`), in order to compare them to a read type UID.

The comparison method `equals` must be used to carry out a comparison with one of the UIDs predefined in `RTUID`.

Example: Calling a technology object system function

In the example below, the system function "resetAxisError" is called for the TO named "Axis_1".

```
// Create object for accessing system functions
RTSystemFunctions vSystemFunctions = new RTSystemFunctions();

// The name of the TO
String vName = "Axis_1";

// ArrayList for the input parameters
      ArrayList vParamsIn = new ArrayList();

// TO name (name of the axis)
vParamsIn.add( vTOName );

// errorResetMode - ALL_ERRORS
vParamsIn.add( new long( 10 ) );

// errorNumber - n/a because ResetMode == ALL_ERRORS
vParamsIn.add( new long( 0 ) );

// nextCommand - IMMEDIATELY
vParamsIn.add( new long( 60 ) );

// commandId
vParamsIn.add(  vSystemFunctions.getCommandId() );

// Array as dummy for the output parameter
Object [] vParamsOut = new object[1];

// Dummy for the number of output parameters
int [] vNumParamsOut = new int[1];

// Call system function
short callResult = vSystemFunctions.callSystemFunction
```

```
(
        vName,
        "_resetAxisError",
        vParamsIn,
        vParamsOut,
        vNumParamsOut
);

// If the system function call has been successfully executed and
// the function result has been stored in the output parameter array
if( callResult == ErrorCodes.RT_ERR_IS_OK && vNumParamsOut[0] == 1 )
{
    // Can the actual function result be retrieved and
    // evaluated.
    int fctResult = (int)((Long)vParamsOut[0]).longValue();

    if( fctResult == 0 )
    {
        // All executed without errors
    }
    else
    {
        // Error code as per description for _resetAxisError
    }
}
else
{
    // System function call unsuccessful
    // Error code is one of the predefined
    // ErrorCodes
}
```

Example: Querying the actual type of a TO

In the example below, the actual type of the "Axis_1" TO is determined.

```
// Create object for accessing system functions
RTSystemFunctions vSystemFunctions = new RTSystemFunctions();

// The name of the TO
String vName = "Axis_1";

// Read type UID
RTUID vId = vSystemFunctions.getTypeUID( vName );

if( RTUID.POS_AXIS.equals( vId ) )
{
    // Position axis recognized,
    // _pos system function could be used...
}
```

Using persistent data (NVRAM)

Overview

Relevant Java package:

- com.siemens.ad.SIMOTION.da

The `RTNVRAMData` class is available for the purpose of using persistent data.

The `RTNVRAMData` class has the following tasks:

- Creating/opening areas within the NVRAM
- Reading/writing areas within the NVRAM
- Querying whether NVRAM areas are valid/setting valid areas
- Closing/deleting areas within the NVRAM

Persistent memory areas are identified by a name.

A non-existent memory area must be created and an existing memory area must be opened.

The data can then be read (though, in the case of newly created memory areas, only after data has been initialized by means of writing), or written.

Where cyclic access is concerned, from a performance standpoint it is better to keep the memory area open, rather than reopening it before and closing it after each access procedure.

Data exchange takes place via a byte array; a start index (within both the persistent memory area and the transferred byte array) and length can be specified.

Only one instance of each persistent memory area can be opened at a time, so the latest point at which memory areas must be closed is on exiting of the user program (or when another user program needs to open the same persistent memory area).

Persistent memory areas that are no longer required can also be deleted.

A straightforward internal marking procedure has been implemented in order to support recognition of inconsistencies within the stored data. This stores information regarding whether the most recent write operation was completed successfully. This marking procedure applies across the entire data area.

For each write operation, the data is marked as "invalid" immediately before it is actually written to the NVRAM, and is then reset to "valid" immediately after the data has been written. If the most recent write operation was not completed successfully, the data area is marked as invalid.

If the data has not been written in its entirety (e.g. there was a power failure during the write operation), an attempt to read the data area triggers the special exception `NVRAMInvalidContentException`, allowing the user program to react accordingly (e.g. by using default values).

As far as handling invalid data areas is concerned, there are also methods of not only determining whether the data area is valid, but also marking it as valid.

The ability to determine the validity of a data area even before it has been read serves as an alternative to troubleshooting using exceptions.

Once a valid data area has been set, `NVRAMInvalidContentException` will not be triggered while the data is being read. The user program is responsible for interpreting the data, which is likely to be inconsistent. This is useful for user programs which implement fault-tolerant archiving of data and can, therefore, reconstruct the most recently valid data.

Newly created data areas are also marked as "invalid", i.e. their data must be "initialized" by means of a write operation before it can then be read. The user program is responsible for initializing the entire data area. Non-initialized parts of the data area cannot be recognized by the internal marking procedure (important for user programs which access individual parts of the data area, and do not always access the entire data area).

Example: Initialization sequence within a user program

```
try
{
    // Length and buffer for application data
    int vLength = 10;
    byte [] vData = new byte [vLength];

    // Variable, whether valid data has been read
    boolean bInitializationDataAvailable = false;

    RTNVRAMData vNVRAM = new RTNVRAMData( "MyData" );

    try
    {
        // Initial attempts to open the area
        vNVRAM.open();

        try
        {
            // Open, data can be read
            vNVRAM.read( vData );

            // After reading, it is ensured that
            // valid initialization data is available
            bInitializationDataAvailable = true;
        }
        catch( DataAccessException e )
        {
            // Reading failed
            // No initialization data available
        }
    }
    catch( DataAccessException e )
    {
        // Opening failed, create memory area
        vNVRAM.create( vLength );

        // Memory area created
        // Initialization data not available
```

```
    }

    if( bInitializationDataAvailable )
    {
        // Valid initialization data available
        // This can be evaluated accordingly
        ...
    }
    else
    {
        // Since no initialization data is available,
        // default values, for example, must be used
        ...
    }
}
catch( DataAccessException e )
{
    // Neither opening nor creation has functioned correctly
    // This is a serious problem, e.g. output exception in log
    // in order to determine cause of error...
}
```

Logging

Overview

Relevant Java package:

- com.siemens.ad.SIMOTION.log

The `Logger` is the central class of the logging API.

Currently, logging outputs are possible in a file or via a socket connection (UDP).

For a description of the format for logging outputs, refer to Format of logging outputs (Page 3956).

Example: Logging

The easiest way of using the logger is as follows:

```
static final String APP_NAME = "MyApplication";

public static void main( String [] args )
{
  // Create the logging object
  Logger vLogger = Logger.getLogger( APP_NAME );

  // Logging output
  vLogger.log( Logger.INFO, "Starting up" );
```

In the example above, exactly the same settings as those in configuration file JINVOKE.XML are used for logging (see LOGGING element (Page 3941)). These have been stored as Java system properties by the InvocationManager.

However, it is also possible to make explicit settings:

```
// Create a logger that only outputs error messages
// (i.e. FATAL and ERROR) without stack trace information
vLogger = Logger.getLogger( APP_NAME, Logger.ERROR, false );

// The messages should be output to log file MYFILE.LOG,
// and the file size should not exceed 16 KB.
// It is IMPORTANT to specify where the logging outputs
// are to be output (file or UDP) before the
// logger is used for the first time.
vLogger.addDestination( new FileLog( "MYFILE.LOG", 16 ) );

// Logging output
vLogger.log( Logger.FATAL, "Terminating" );
```

Note

Particularly in cases where several user programs are being operated simultaneously and logging outputs are output to a file, it may be helpful to assign each user program its own log file. Since the process of writing to the log file is synchronized, logging outputs from different user programs to the same log file could impair performance.

Creating user programs

User programs are created on a development computer.

To enable the Java compiler to find the classes referenced within the SIMOTION API, the relevant Java library must be specified during compilation.

For example:

```
javac –classpath simotion.jar ...
```

To ensure that the user program will not use any Java system classes that are not available within the SIMOTION Jamaica runtime environment, the Java library with system classes can also be specified as a reference during compilation.

For example:

```
javac –bootclasspath simotion_systemclasses.jar –classpath
simotion.jar ...
```

Following compilation, all of the user program classes must be zipped into a Java library.

For example:

```
jar –cvf test.jar ...
```

The Java library and user program can then be transferred to the target system and started up.

Configuration files

AUTOSTART configuration (JINVOKE.XML)

Overview

The settings for user programs that are to be started automatically must be made in configuration file JINVOKE.XML.

Settings for the interactive InvocationManager interface and logging can also be made in this file.

XML notation must be used for the structure of the configuration file. A simple XML parser (Nano/XML/Lite) is integrated in SIMOTION IT Virtual Machine that is perfectly adequate for editing the configuration file.

Before storing the configuration file on the target system, you are advised to display it in a web browser, for instance, so that any syntax errors can be made immediately apparent (if the configuration file has not been created using a special XML editor). NanoXML, in particular, deals with error messages very economically.

The individual XML configuration file elements are described below.

ROOT element

SIMOTION must be used as the ROOT element:

```
<?xml version="1.0" standalone="yes"?>
<SIMOTION>
  <!-- AUTOSTART elements, for example, are listed here -->
</SIMOTION>
```

All other elements must then be specified within the Root element. The above example contains only one comment line.

AUTOSTART element

An AUTOSTART element contains information about a user program that is to be started automatically.

You can specify several AUTOSTART elements. Therefore, it is possible to start several user programs. The start sequence corresponds to the sequence in which the AUTOSTART elements are specified.

The following attributes are evaluated:

LIBRARY

Name of the Java library (JAR or ZIP file) that contains the program to be started.

The name specification can be either absolute or relative.

If the specification is relative, i.e. only the file name is specified without the path, the file itself must be in the current working directory (see CURRENTDIR (Page 3945)).

CLASSNAME

Name of the class containing the method to be executed.

METHODNAME

Name of the method to be executed.

This method must be declared as `public static`, and must accept a string array as a single argument. Any value can be returned for the method.

Specification of the method name is also unnecessary. In this case, the main method is searched for.

ARGUMENT element

An AUTOSTART element can contain any number of ARGUMENT elements.

The arguments are stored in the sequence in which they were specified in the string array transferred to the method to be executed.

An ARGUMENT element only has one attribute, called VALUE.

Example: AUTOSTART elements

In this example, the MyClass class is first loaded from Java library USER.JAR, and its main method is then executed. The string array transferred to the main method contains the Argument1 and Argument2 strings.

Following this, the Test class is loaded from Java library TEST.JAR and its main method is executed. Since no arguments are specified, the string array transferred to the main method is empty.

```
<AUTOSTART LIBRARY="USER.JAR" CLASSNAME="MyClass">
  <ARGUMENT VALUE="Argument1"/>
  <ARGUMENT VALUE="Argument2"/>
</AUTOSTART>

<AUTOSTART LIBRARY="TEST.JAR" CLASSNAME="Test"/>
```

LOGGING element

Overview

The LOGGING element contains default settings for logging.

These default settings take effect when user programs use the logging API without explicit settings (see Logging (Page 3938)).

The following attributes are evaluated:

DEFAULT_DESTINATION

Currently, the default settings dictate that messages are output to a file and transferred to a host via the UDP network connection.

Possible specifications:

- FILE
- UDP

DEFAULT_LEVEL

This determines the default logging level.

Possible specifications:

- FATAL
- ERROR
- WARN
- INFO
- DEBUG

If nothing has been specified, or a specification that does not appear in the list above has been made, the logging level is set to DEBUG.

PRINT_STACKTRACE

Stack trace information is generally useful for error analyses in Java programs. If PRINT_STACKTRACE is activated, when an exception is logged the relevant stack trace information will automatically be written to the log.

Possible specification:

- true

If nothing has been specified, or a specification other than "true" has been made, no stack trace will be logged.

FILE element

The FILE element writes the default values for the log outputs to a file, and contains the following attributes.

FILENAME ... Name of the log file
MAXSIZE_IN_KB ... Maximum size of the log file in kilobytes

The file name specification can be either absolute or relative.

If the file name specification is relative and without the path, the file is generated in the current working directory (see CURRENTDIR (Page 3945)).

The log file resembles a ring buffer in terms of the way entries are made, i.e. when the maximum size that has been specified is exceeded, the process loops back to the beginning of the file. The most recent log entry is always followed by the character string `$$END$$`, which acts as a kind of separator between old and new log entries and makes it easy to find the most recent log entry.

UDP element

The UDP element writes the default values for the log outputs via a UDP socket connection, and contains the following attributes:

HOST	... IP address of the log outputs host
PORTNUMBER	... Port number for the log outputs receiver

Example: LOGGING element

In this example, the logging level is: DEBUG, and logging outputs, including stack trace information, are output to log file LOGFILE.LOG by default; their maximum size is limited to 64 KB.

The default values set for UDP logging are 157.163.237.50 for the host and 20000 for the port number.

```
<LOGGING DEFAULT_DESTINATION="FILE" DEFAULT_LEVEL="DEBUG"
PRINT_STACKTRACE="true">
  <FILE FILENAME="LOGFILE.LOG" MAXSIZE_IN_KB="64"/>
  <UDP HOST="157.163.237.50" PORTNUMBER="20000"/>
</LOGGING>
```

INVOCATION_MANAGER element

Overview

The INVOCATION_MANAGER element may contain other information for the InvocationManager, which is mainly used for setting the interactive socket interface.

The following attributes are evaluated:

PORTNUMBER

If a port number has been specified, the InvocationManager opens a server socket under this number and waits for requests.

It is then possible to load and start programs interactively via a socket connection.

If no port number has been specified, the InvocationManager ends after the AUTOSTART elements have been processed.

PORTRANGE

PORTRANGE is useful while developing and testing user programs if the JamaicaVM (and, therefore, the InvocationManager) is restarted frequently. Many TCP/IP implementations reserve this port number for a specific time. During this time, renewed opening of the server socket is not permitted under this port number.

If a port range has been specified, the InvocationManager attempts to use the next highest group of port numbers, until the server socket can be opened.

The port number that is currently used to access the InvocationManager is always output by means of a logging output.

Comment:

The InvocationManager attempts to create the server socket cyclically, i.e. even if no port range is specified, the socket connection can be reestablished after the delay time has elapsed.

CONSOLE_OUTPUT

Instead of the default target for logging outputs, the InvocationManager uses the screen. The screen output is useful in simulation environments, since it means that no log files need to be evaluated, for example.

Possible specification:

- true

If no specification has been made, or something other than "true" has been specified, the default target for logging outputs is used.

Example: INVOCATION_MANAGER element

In this example, once the AUTOSTART entries have been processed, the InvocationManager opens a server socket under port number 10000 and waits for requests.

If this port number cannot be used, the InvocationManager also tries port numbers 10001 to 10010.

```
<INVOCATION_MANAGER PORTNUMBER="10000" PORTRANGE="10"/>
```

DATA_ACCESS element

Overview

The DATA_ACCESS element is optional: Only the SIMULATION attribute, via which simulation mode can be activated for the data access API, is evaluated.

SIMULATION

Possible specification:

- true

Simulation mode is activated by specifying true. No accesses are made to the target hardware. Simulation mode is useful for the fundamental testing of user programs, also in advance, (e.g. on a development computer) without the need for target hardware.

When reading or writing variables and drive parameters, the current content of each Java variable remains unchanged. Calling system functions returns a positive function result in each case.

However, for the purpose of accessing persistent memory areas (NVRAM), simulation mode is restricted in that an attempt to open a memory area will always fail. No persistent data is simulated.

If nothing has been specified, or a specification other than "true" has been made, simulation mode will be deactivated.

JamaicaVM configuration (VMCONFIG.INI)

Overview

The configuration file VMCONFIG.INI is only evaluated in the target system. This file is not evaluated in tests on a development computer.

Default values are permanently specified for all JamaicaVM settings, i.e. it is possible to operate the JamaicaVM with the relevant default values without the need to make any special settings.

Specific default directories for Java program files and Java data files are recommended as of SIMOTION V4.1 SP4 so that the CPU update feature can be used for Java programs as well. These settings are included in the configuration files supplied and should be accepted without making any modifications to them.

However, the file must be available for the JamaicaVM to start at all.

If necessary, settings for the JamaicaVM itself can be made in configuration file VMCONFIG.INI. The settings take effect when the JamaicaVM is started up.

Each parameter must be specified in a single line, with the following format:

Parameter=Value

.

Blank lines are permitted. Spaces are also permitted, but not within the parameter or the value.

Comment lines must begin with "#".

Each of the parameters and their default values are outlined below:

VMCONFIG.INI

With SIMOTION V4.1 SP4 and higher, the configuration file VMCONFIG.INI is searched for primarily in the directory:

- /USER/SIMOTION/HMICFG/

.

This means that if file VMCONFIG.INI is found in directory /USER/SIMOTION/HMICFG/, it is evaluated and the JamaicaVM is started with the relevant settings.

For reasons of compatibility, VMCONFIG.INI is still evaluated even if it is located in the root directory, However, evaluation only takes place if no VMCONFIG.INI file has been found in the /USER/SIMOTION/HMICFG/ directory. This means that you can continue to use existing configurations/installations without making any changes to them.

SIMOTIONVM_HEAPSIZE

The size of the memory reserved for the JamaicaVM in bytes. The JamaicaVM reserves the entire memory, even when starting up. If the memory is not available, an error message is output to the output file (see Output file JCONSOLE.TXT (Page 3949)).

Default: 4194304

SIMOTIONVM_NUMUSERTHREADS

Number of Java user threads required.

The JamaicaVM creates the required number of threads as soon as it starts up. If the JamaicaVM is restarted, it will no longer be possible to change this setting.

If the required number of threads cannot be created, an error message is output to the output file (see Output file JCONSOLE.TXT (Page 3949)).

Default: 4

Maximum number that can be set: 28

This number also contains the main Java program executed by default ("InvocationManager").

In the default setting, three Java threads are thus available for user programs. This means three user programs can be run in parallel.

For special solutions, the InvocationManager can also be replaced by a user-defined program: This makes all Java user threads available for user programs.

SIMOTIONVM_STACKSIZE

Size of each Java thread stack.

Default: 40000

Maximum number that can be set: 81920

This parameter must only be changed in special cases. Such a special case could be a memory problem, for example. Reducing the stack size could help with this problem.

If the JamaicaVM is restarted, it will no longer be possible to change this setting.

SIMOTIONVM_ANALYSE

The JamaicaVM can be instructed to output information relating to the maximum amount of memory required during execution of the previous Java programs, once the JamaicaVM has been exited.

Memory analysis mode provides a tool for estimating memory requirements. However, the information in the output file is only the result of the measurement in the most recently run program, and does not indicate the exact memory requirement of the JamaicaVM. Running programs following this can, therefore, result in a larger memory requirement due to external influences (different data, different scheduling of threads due to temporal influences, etc.). For safety reasons, approx. 20 % more memory than is actually required should be available.

Information relating to the memory requirement is written to the output file (see Output file JCONSOLE.TXT (Page 3949)).

This parameter determines the accuracy of the memory analysis as a percentage, i.e. if 10 is specified, the maximum memory requirement specified is accurate to within 10%. Consequently, the results of the analysis may then be up to 10 % higher than the memory that is actually required. However, the analysis slows down the execution of Java programs. If the value is zero, no analysis is carried out.

Default: 0

HOME

Mapped onto the "user.home" Java system property.

Default: B:/USER/SIMOTION/HMI/JAVA/FSROOT/

CURRENTDIR

Mapped onto the "user.dir" Java system property.

"user.dir" is the current working directory for all Java programs. In particular, both the configuration file for the InvocationManager and the Java libraries containing the user programs to be started automatically must be in this directory.

Default: B:/USER/SIMOTION/HMI/JAVA/PROG/

TEMPDIR

Mapped onto the "java.io.tmpdir" Java system property.

Default: A:/

BOOTCLASSPATH

Bootclasspath for the JamaicaVM. The JamaicaVM has the actual system classes "on-board". However, the Java library containing the InvocationManager and the API for data access and logging must be specified. When specifying several libraries, the libraries must be separated by semicolons.

The file name and the path must be specified.

Default: B:/SIEMENS/SIMOTION/SIMOTION.JAR

CLASSPATH

Additionally required Java libraries can be specified here using the CLASSPATH parameter.

If several libraries are specified, they must be separated using dash-points.

In the case of CLASSPATH, the file name and the path must be specified.

Default: No default value

MAINCLASSNAME

Name of the Java program that is executed when the JamaicaVM is started up.

The default SIMOTION Java start class (i.e. the InvocationManager) executes the user programs in accordance with the settings specified in JINVOKE.XML.

For this reason, if required the InvocationManager can be abandoned and a user program executed immediately instead. Arguments cannot be specified.

Default: com.siemens.ad.SIMOTION.invocation.InvocationManager

DEBUGGER

Setting options for the debugger

Several parameters are possible for setting the debugger. These parameters must be formulated in a line, like individual assignments (param=value) in each case. If several parameters are specified, they must each be separated by a comma.

The possible specifications are **address**, **suspend** and **server**, followed by the detailed description in each case.

address

If debugger mode is activated, the address used by the debugger must be specified. The target system is operated as standard as the debugging server. The address then only describes the relevant port number.

suspend

This parameter is optional. The parameter determines whether or not the JamaicaVM waits for a debugger to connect with it before running the Java applications. The possible values for the parameter are 'y' or 'n'.

The default value is 'y'. This means, if debugger mode is activated, the JamaicaVM waits as standard until a debugger has connected to it before running the applications. Different debugger frontends can handle connection buildup to the JamaicaVM differently.

For example, when using Eclipse at the start of the debugging session after connection buildup, any set breakpoints and the command for running the Java application are sent simultaneously. In this case, it is necessary, for example, to set desired breakpoints before starting the debugging session so that they take effect immediately when the debugging session is started.

server

This parameter is optional and determines whether or not the target system is to be operated as the debugging server. The possible values for the parameter are 'y' or 'n'.

The default value is 'y'. The target system is then operated as the debugging server. If debugger mode is active, the JamaicaVM opens a remote debugging server port.

By specifying 'n', the target system is operated as the debugging client. The JamaicaVM attempts to establish the network connection to a debugging server. The address specification must additionally include the TCP/IP address of the debugging server. The format is then:
`<TCP/IP address>:<port number>.`

Examples:
`DEBUGGER=suspend=n,address=8000`

The debugger is activated and opens the remote debugging port with the port number 8000. When the JamaicaVM is started up, it does not wait to run the Java applications.

`DEBUGGER=suspend=y,server=n,address=192.168.214.2:8000`

The debugger is activated and attempts to establish the network connection to the specified address. When the JamaicaVM is started up, it waits before running the Java applications.

If the specification is missing, debugger mode is not activated.

Default: No default value (debugger not activated)

Example

Example

Setting the memory size for the JamaicaVM and the current working directory for all Java programs.

```
SIMOTIONVM_HEAPSIZE=4500000
CURRENTDIR=B:/JAVA
```

Note

To determine the memory size required by the JamaicaVM, see also setting option SIMOTIONVM_ANALYSE (Page 3945).

Transferring user programs to the target system

Transfer via FTP is an extremely user-friendly option for transferring the user program from the development computer to the target system in the form of a Java library (JAR or ZIP file). (An FTP server is normally active on the target system).

All you need to do is ensure that "binary mode" is always set, so that the file will be transferred securely. Otherwise, the file could become changed during transmission. This will mean that classes within the Java library cannot be found.

Diagnostics buffer entry

A diagnostics buffer entry is created for errors that prevent the JamaicaVM from starting up.

The diagnostics buffer entry is only generated in the target system. The diagnostics buffer entry is not generated in tests on a development computer.

The diagnostics buffer entry contains a note if a problem arises when starting the JamaicaVM. In addition, the diagnostics buffer entry contains a simple error number for roughly determining the cause of the error.

Table 6-62 Possible error numbers

Error number	Error cause
1	The JCONSOLE.TXT output file could not be initialized.
2	The VMCONFIG.INI configuration file contains an invalid setting.
3	Error while starting the JamaicaVM (memory could not be reserved, insufficient memory for starting up, start class not found, etc.).

If error number 1 appears, the file system must be checked.

If error numbers 2 and 3 appear, the exact description of the error cause can be found in the JCONSOLE.TXT output file.

The error can subsequently be eliminated by adjusting the configuration of the JamaicaVM (see JamaicaVM configuration (VMCONFIG.INI) (Page 3945)).

Output file JCONSOLE.TXT

The output file JCONSOLE.TXT is only generated in the target system. This file is not generated in tests on a development computer.

All internal outputs of the JamaicaVM (such as information on the start settings, error messages, information after it has been exited, etc.) are written to a file.

This output file is called JCONSOLE.TXT. It is recreated each time the entire system is started up, and always in directory /USER/SIMOTION/HMI/SYSLOG/ of drive B:.

The output resembles a ring buffer. This means, if the maximum size of 128 KB is exceeded, a start is made again at the beginning of the file. The most recent entry is always followed by the character string $$END$$ to make it easy to find the entry. The character string serves as a separator between the old and new entries.

A typical example of the content of this output file:

```
jvmCreate
jamaica_main_task_entry: coming up
jamaica_main_task_entry: processing jvm configuration
SIMOTIONVM_HEAPSIZE = 4194304
SIMOTIONVM_NUMUSERTHREADS = 4
SIMOTIONVM_STACKSIZE = 40000
SIMOTIONVM_ANALYSE = 0
HOME = B:/USER/SIMOTION/HMI/JAVA/FSROOT/
CURRENTDIR = B:/USER/SIMOTION/HMI/JAVA/PROG/
TEMPDIR = A:/
BOOTCLASSPATH = B:/SIEMENS/SIMOTION/SIMOTION.JAR;
CLASSPATH =
MAINCLASSNAME = com.siemens.ad.SIMOTION.invocation.InvocationManager
jamaica_main_task_entry: waiting for prepared worker tasks
jamaica_main_task_entry: going to startup JamaicaVM:
jamaica
-Xbootclasspath:B:/SIEMENS/SIMOTION/SIMOTION.JAR;
com.siemens.ad.SIMOTION.invocation.InvocationManager
InvocationManager enters main...
Current working directory is: B:/
OK - starting user program
OK - starting user program
$$END$$
```

The example shows the following:

- The outputs for the power-up phase of SIMOTION Java
- The corresponding JamaicaVM parameters, the actual start of the JamaicaVM and the InvocationManager
- Including the start of two user programs.

More detailed InvocationManager outputs, e.g. which user programs have been started up, are output via logging (in accordance with the settings made in LOGGING element (Page 3941)).

The output file can be transferred for viewing from the target system to the development computer using FTP (an FTP server is normally active on the target system).

Information on the JamaicaVM memory requirement

If the JamaicaVM is operated in memory analysis mode via the SIMOTIONVM_ANALYSE setting (see also SIMOTIONVM_ANALYSE (Page 3945)), information on the maximum amount of memory required during a cycle is also written to the output file after the JamaicaVM has been exited.

The output might look like this:

```
### Application used at most 3773001 bytes for the Java heap
(accuracy 10%).
```

```
### Non-Java heap memory used: 209404 bytes.
###
###                     Worst case allocation overhead:
###     heapSize        dynamic GC      static GC
###     12954k          6               3
###     10854k          7               4
###     9509k           8               4
###     8598k           9               4
###     7929k           10              4
###     7028k           12              5
###     6461k           14              5
###     6054k           16              6
###     5762k           18              6
###     5537k           20              7
###     5218k           24              8
###     5003k           28              9
###     4851k           32              10
###     4731k           36              11
###     4639k           40              12
###     4504k           48              14
###     4411k           56              17
###     4345k           64              19
###     4188k           96              27
###     4112k           128              36
###     4035k           192             53
###     4000k           256             69
###     3965k           384             100
```

In this case, the minimum heap size is 3965 KB. The JamaicaVM could thus also be operated with a heap size of 4060160 bytes. The second column shows that in the worst-case scenario 384 allocation steps could be required to allocate an object. If, on the other hand, a 4035 KB heap were available, the worst-case scenario would involve 192 allocation steps.

The memory requirement is made up of 3773001 bytes for the actual Java heap and 209404 bytes for the "non-Java heap".

All objects managed by the garbage collector, and also internal objects of the VM, are stored on the Java heap. The non-Java heap contains the stacks and the static structures.

Interactive InvocationManager interface

Overview

The interactive InvocationManager interface allows user programs to be tested repeatedly without the need, for example, to exit the JamaicaVM each time.

The relevant commands for this can be sent to the InvocationManager via a socket connection.

A typical "development cycle" might progress as follows:

- Create Java library with user program on development computer
- Store Java library with user program on target system

- Open Java library via InvocationManager request
- Start user program via InvocationManager request
- Once the user program has been exited, the Java library can be closed by means of an InvocationManager request and then recreated, once expansions have been made on the development computer. Following this, it can be updated on the target system (e.g. via FTP transfer) and reopened, and the user program can be started up.

The InvocationManager protocol

All parts of a request must be separated by/ended with a line feed character (0x0A). It is crucial that you do not forget the final line feed character. The InvocationManager waits until it receives the line feed character before executing the request. If the socket connection was terminated before that, the InvocationManager terminates processing of the request.

A request must begin with one of the following keywords:

- cmd... followed by the actual command and, if applicable, its parameters
- job... followed by the class name, method name, and arguments (optional)
- task... followed by the class name, method name, and arguments (optional)

Commands

openArchive

This command opens the relevant Java library.

The command must be followed by the name of the Java library to be opened.

The name specification can be either absolute or relative.

If the specification is relative, i.e. only the file name is specified without the path, the file itself must be in the current working directory (see CURRENTDIR (Page 3945))

Only one Java library can be opened via the interactive interface at any one time. Before the specified Java library is opened, any other Java library that may currently be open is closed automatically.

closeArchive

This command closes the Java library that is currently open.

setLoggingLevel

The command must be followed by the name of the new logging level.

For the correct name, see DEFAULT_LEVEL (Page 3941).

This command can be used during runtime to change the appropriate logging level for the InvocationManager logging outputs.

exit

This command is used to exit the InvocationManager.

By exiting the InvocationManager, the JamaicaVM is also exited, but only if no other user program is active.

job/task requests

Both types of request enable a user program to be executed. The only difference lies in how the response sent to the client is handled.

The method to be called must be declared as public static, and must accept a string array as a single argument. Any value can be returned for the method.

The "job" type of request is intended for user programs with a "function result" (i.e. the method called supplies a result object, preferably after a relatively short runtime). With a "job" request, the InvocationManager holds off on sending the response until the method called returns, converts the result object into a string, and sends this string to the client. If the method does not supply a result object (e.g. a method with a void return type), the default response is sent to the client
OK - request finished

.

The "task" type of request is intended for user programs without a "function result" (these may have an extremely long runtime, or may never end). With a "task" request, the InvocationManager sends the default response to the client immediately before the relevant method is called
OK - starting user program

It then terminates the socket connection. If the client receives the response, it is guaranteed that the specified class has been successfully loaded and that the specified method is also available. However, a direct function result is not available.

If errors occur, the InvocationManager sends the relevant error message to the client as a response.

Once the response has been sent, the InvocationManager terminates the socket connection.

Note

The user program is started in a separate Java thread in the case of both request types. This means that the InvocationManager itself does not block any requests while the user program is running, but can instead continue to receive further requests at the same time.

Sample requests

Open Java library USERPROG.JAR.

```
cmd\nopenArchive\nUSERPROG.JAR\n
```

Load the `MyClass` Java class and execute the `runIt` method with arguments `aArgument1` and `aArgument2` ("zipped" as a string array).

```
job\nMyClass\nrunIt\naArgument1\naArgument2\n\n
```

Note

The two line feed characters at the end of the request must be available. An "empty argument" indicates the end of the argument list to the InvocationManager.

Sample programs

In addition to the documentation, sample programs in Java source code format are available as basic help methods (see Scope of delivery (Page 3913)).

Examples of data access, file functions, marshalling, and multithreading are available.

Examples of implementing tools, such as an InvocationManager client and a UDP logging host, are also available.

com/siemens/ad/SIMOTION/demo/da/NVRAMDemo.java

Example of using persistent data (stored in the NVRAM)

com/siemens/ad/SIMOTION/demo/da/Simple.java

Example of simple data access

com/siemens/ad/SIMOTION/demo/da/STVariable.java

Example of accessing variables for an ST program

com/siemens/ad/SIMOTION/demo/da/SystemVariable.java

Example of accessing system variables

com/siemens/ad/SIMOTION/demo/da/VariableBrowser.java

Example of browsing variables, including how to determine the data type of scalar variables using RTVariableANY

com/siemens/ad/SIMOTION/demo/da/VariableProperties.java

Example of reading variable properties.

com/siemens/ad/SIMOTION/demo/drive/DriveAccess.java

Example of how parameters, their description, and the error buffer of a drive can be accessed.

Note

For the demo program to process successfully, not only is a relevant project required on the target system, but also an addressable drive that is connected to the target system. The example requires a SINAMICS at the logic address 258.

com/siemens/ad/SIMOTION/demo/file/Simple.java

Example of file access; in particular, how file names can evolve independently of a platform

com/siemens/ad/SIMOTION/demo/invocation/InvocationClient.java

Example of how requests are sent to the InvocationManager

com/siemens/ad/SIMOTION/demo/marshalling/Receiver.java

Example of how the values of the Java variables can be set on the basis of the raw data received (UDP data telegram).

com/siemens/ad/SIMOTION/demo/marshalling/UDPRawDataSender.java

Utility program which can be used to send raw data to the marshaling receiver for testing purposes.

com/siemens/ad/SIMOTION/demo/log/UDPLoggingHost.java

Example of how the logging outputs can be received and output via UDP.

com/siemens/ad/SIMOTION/demo/sys/DeviceFunctions.java

Example of how device functions can be called

com/siemens/ad/SIMOTION/demo/sys/SystemFunctions.java

Example of how system functions of a TO can be called

com/siemens/ad/SIMOTION/demo/thread/Simple.java

Example of a user program with several threads. The main thread creates and starts a worker thread, which sends data cyclically to a UDP receiver.

At the same time, the main thread waits at a UDP socket for incoming data. When data is received, the main thread stops the worker thread and also stops itself.

Note

The UDPLoggingHost can be used as a UDP receiver and the UDPRawDataSender as a UDP sender (for the purpose of sending the stop request to the sample program). All you need to do is adapt the connection information (if applicable).

Format of logging outputs

The format of the logging outputs is as follows:

```
<Timestamp> | <LoggingLevel> | <LogTopic> | <LogMessage>
```

Timestamp

Time at which the logging output was created, in milliseconds from 1/1/1970, 00:00.

LoggingLevel

LoggingLevel that was specified with the logging output.

LoggingTopic

LoggingTopic that was specified when the logger was created. This allows logging outputs from different user programs to be distinguished from one another.

LoggingMessage

Message that was specified with the logging output.

If an exception (Java exception) is logged, the message also contains the description of the exception, including the stack trace information (only when automatic logging of the stack trace information is activated; see PRINT_STACKTRACE (Page 3941)).

Example

Below is an example of the program lines:

```
static final String APP_NAME = "Simple";

    public static void main( String [] args )
    {
        Logger vLogger = Logger.getLogger( APP_NAME );
        vLogger.log( Logger.INFO, "Starting up" );
        vLogger.log( Logger.INFO,
                     "CurrentDirectory is: " +
                      System.getProperty( "user.dir" ) );
```

and the relevant log outputs for a user program:

```
694224004323 | INFO | Simple | Starting up
694224004323 | INFO | Simple | CurrentDirectory is: B:/
```

Restarting the JamaicaVM

When the last Java thread has ended, the JamaicaVM will also stop.

To restart the JamaicaVM, the jvmrestart.mwsl program can be restarted via the Web browser:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="ISO-8859-1">
    <title>Call JVMRESTART with CSRF Token</title>
  </head>
  <body>
    <a href="http://192.168.0.1/JApp/jvmRestart?<%=SingleUseToken
%>">http://192.168.0.1/JApp/jvmRestart</a>
  </body>
</html>
```

The IP address 192.168.0.1 must be changed in accordance with the own configuration.

The CSRF protection is implemented for this link by using the Server Side Include < %=SingleUseToken%>. The <%=SingleUseToken%> tag is resolved into a string such as: "SingleUseToken=i9/mSm1Bvc+BkNgOWNE+4FYTRNk=". The first "=" is followed by the token, but only if a user is logged on. The SSI <%RawSingleUseToken%> can be used for applications in which only the pure token value is to be used (e.g. within an HTML form "<form>"):

```
<html>
 <head>
  <meta charset="ISO-8859-1">
  <title>Call JVMRESTART with CSRF Token</title>
 </head>
<body>
<form method="get" action="/JApp/jvmRestart" >
<!-- CSRF Token input _must_ be at 1. place in form !!! -->
        <input type="hidden" name="SingleUseToken" value="<%=RawSingleUseToken%>"/>
        <input type="submit" value="Restart JVM now"/>
</form>
</body>
</html>
```

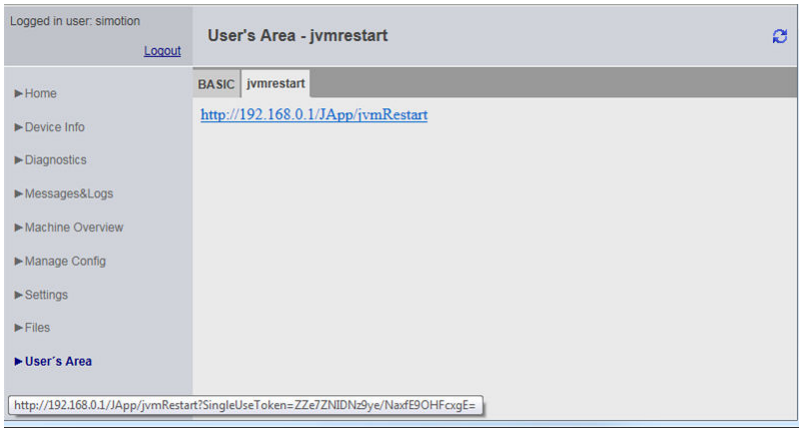


Figure 6-154 Restart JamaicaVM

Further information can be found in the *CSRF protection* chapter in the *SIMOTION IT Diagnostics and Configuration* manual.

If the CSRF token is not used, the browser outputs an error message.

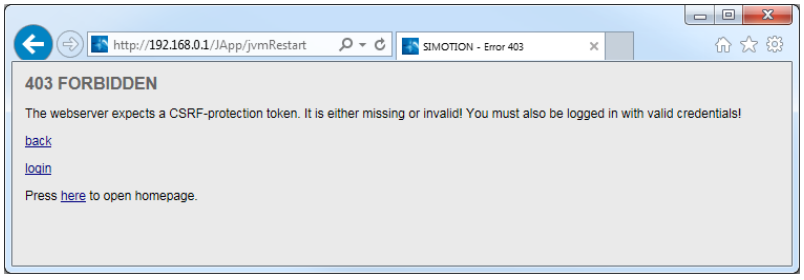


Figure 6-155 New restart of the JamaicaVM without CSRF token

Call of jvmRestart up to V5.2

In the older versions, a direct start of jvmRestart in the browser is possible.
`http://<host>/JApp/jvmRestart`

Note

Restart of the JamaicaVM is not forced by calling `jvmRestart`. A restart will only be successful if the JamaicaVM has shut down in the correct manner.

6.4.4.2 SIMOTION IT Servlets

Scope of functions implemented

"Java™ Servlet Specification, V2.2" supports SIMOTION IT Servlets.

However, not all of the concepts from the standard servlet API are implemented in SIMOTION IT Servlets.

The following are implemented:

- Differentiation between type of request (GET, POST)
- Access to all parameters of the request (from both the URL and form data)
- Access to the content of the request
- Access to the request header
- Specification of the response status
- Specification of the response header
- Writing of the response content
- Logging
- Initialization parameters from configuration file for servlets
- Servlet context with attribute management and initialization parameters

These enable the implementation of fully-fledged servlets, including structure and entry form editing features (e.g. for web browsers).

The section titled Implementing the servlet API (Page 3960) contains a detailed description of the functions implemented.

More advanced concepts such as the following are not implemented:

- Support for HTTP sessions
- Support for RequestDispatcher
- Support for security functionality
- Access to client information (remote host, etc.)
- Internationalization (only the ISO-8859-1 character set is available)
- Java server pages

Note

Security functionality

An authentication feature is implemented within the connection to the SIMOTION web server, which enables you to limit access to servlets as standard in the interests of security.

However, the Java security functions defined within the servlet API (such as the role-based security concept) are not implemented.

Scope of delivery

The following files are integral parts of SIMOTION IT Servlets:

File name	Description
svltapi.jar	Java library containing the standard servlet API.
svltimpl.jar	Java library containing the implementation of the servlet container.
simotion_servlets_doc.zip	Online documentation of the servlet API (HTML format).
simotion_servlets_demo.zip	Java source code for sample servlet.
JSERVER.XML	Sample servlet configuration file.

Implementing the servlet API

SIMOTION IT Servlets supports the "Java™ Servlet Specification, V2.2".

For the servlet API itself, the relevant implementation of the "Apache Jakarta Project" is used. This has been developed by the Apache Software Foundation (http://www.apache.org/ (http://www.apache.org/)) and does not provide support for Java Server Pages.

Essentially, this means that unless Java Server Pages need to be supported, all of the specified interfaces and classes for the servlet API are available. Additionally, the online documentation for the servlet API provides full descriptions of all the interfaces and classes it contains.

Certain interfaces, as well as certain interface methods, are not supported in the first version of SIMOTION IT Servlets, as they are not required for simple servlets.

Generally speaking, methods referred to in the servlet specification as "deprecated" (meaning that they are outdated and should no longer be used) are not supported.

If certain interface methods are not supported, calling them will trigger an exception of type "UnsupportedOperationException".

The table below provides both a detailed list of what is included in the servlet API and a description of the scope of functions implemented in SIMOTION servlets.

javax.servlet package	
	RequestDispatcher Not implemented
	ServletConfig Fully implemented
	ServletContext The following methods are implemented: public int getMajorVersion() public int getMinorVersion() public void log(String msg) public void log(String msg, Throwable throwable) public String getServerInfo() public String getInitParameter(String name) public Object getAttribute(String name) public Enumeration getAttributeNames() public void setAttribute(String name, Object object) public void removeAttribute(String name) public String getContextPath()

	ServletRequest The following methods are implemented: public int getContentLength() public String getContentType() public ServletInputStream getInputStream() throws IOException public String getParameter(String name) public Enumeration getParameterNames() public String[] getParameterValues(String name) public String getProtocol() public String getServerName()
	ServletResponse The following methods are implemented: public ServletOutputStream getOutputStream() throws IOException public void setContentType(String type) public void reset() public boolean isCommitted() public void flushBuffer() throws IOException
	SingleThreadModel Already implemented as appropriate in servlet API
	GenericServlet Already implemented as appropriate in servlet API
	ServletInputStream Fully implemented
	ServletOutputStream Fully implemented
	ServletException Already implemented as appropriate in servlet API
	UnavailableException Already implemented as appropriate in servlet API
javax.servlet.http package	
	HttpServletRequest The following methods are implemented: public Cookie[] getCookies() public long getDateHeader(String name) public String getHeader(String name) public Enumeration getHeaderNames() public int getIntHeader(String name) public String getMethod() public String getContextPath() public String getPathInfo() public String getQueryString() public String getRequestURI() public String getServletPath()

	HttpServletResponse The following methods are implemented: public void addCookie(Cookie cookie) public boolean containsHeader(String name) public void sendError(int sc, String msg) throws IOException public void sendError(int sc) throws IOException public void setDateHeader(String name, long date) public void setHeader(String name, String value) public void setIntHeader(String name, int value) public void setStatus(int sc)
	HttpSession Not implemented
	HttpSessionBindingListener Not implemented
	HttpSessionContext Not implemented
	Cookie Already implemented as appropriate in servlet API
	HttpServlet Already implemented as appropriate in servlet API
	HttpSessionBindingEvent Already implemented as appropriate in servlet API
	HttpUtils Already implemented as appropriate in servlet API
javax.servlet.jsp package This package (along with its subpackages) is not included in SIMOTION servlets.	

Creating user servlets

User servlets are created on a development computer.

To enable the Java compiler to find the classes referenced within the servlet API, the relevant Java library must be specified during compilation.

For example:
```
javac –classpath svltapi.jar ...
```

If classes are referenced by the SIMOTION API, it is also necessary to specify the relevant Java library during compilation.

For example:
```
javac –classpath svltapi.jar;simotion.jar ...
```

Remark:

The separator within the CLASSPATH specification is platform-specific. If a Windows development computer is used, the separator is the semicolon, as in the above example. On a Unix development computer, a colon must be used.

To ensure that the user servlet will not use any Java system classes that are not available within the SIMOTION IT Virtual Machine runtime environment, during compilation the Java library can also be specified with the system classes as a reference.

For example
```
javac –bootclasspath simotion_systemclasses.jar –classpath
svltapi.jar ...
```

Following compilation, all of the user program classes must be zipped into a Java library.

For example:
```
jar –cvf testsvlt.jar ...
```

Following this, the Java library can be transferred to the target system along with the user servlet and loaded by the servlet container.

Configuration file (JSERVER.XML)

The configuration file JSERVER.XML contains the settings for the SIMOTION IT Servlet container and the servlets to be loaded.

XML notation must be used for the structure of the configuration file. A simple XML parser (Nano/XML/Lite) is integrated in SIMOTION IT Virtual Machine that is perfectly adequate for editing the configuration file.

Before storing the configuration file on the target system, you are advised to display it in a web browser, for instance, so that any syntax errors can be made immediately apparent (if the configuration file has not been created using a special XML editor). NanoXML, in particular, deals with error messages very economically.

The individual XML configuration file elements are described below:

<table>
<tr><td colspan="2">ROOT element
SIMSERV must be used as the root element:
<pre><?xml version="1.0" standalone="yes"?>
<SIMSERV>
  <!—Initialization parameter of context -->
  <INITPARAM NAME="AParameter" VALUE="aValue"/>
  <!-- SERVLET elements, for example, are listed here -->
</SIMSERV></pre>All other elements must then be specified within the ROOT element.
The example above contains just one initialization parameter, as described below.</td></tr>
<tr><td colspan="2">INITPARAM element
The ROOT element can contain any number of INITPARAM elements. These initialization parameters can be queried from servlets using the Servlet Context,
meaning that they are available for all servlets.
The following attributes are evaluated:</td></tr>
<tr><td></td><td>NAME
Name of the initialization parameter.
If this attribute is missing, the element will not be evaluated any further.</td></tr>
<tr><td></td><td>VALUE
Value of the initialization parameter.
If this attribute is missing, the empty string is used as the value.</td></tr>
</table>

SERVLET element A SERVLET element contains information about the servlet to be loaded. The name of the Java class that implements the servlet and the path (part of the URL following the ContextPath) under which the servlet is to be registered must be specified for a servlet that is to be loaded. You can specify several SERVLET elements. meaning that it is possible to load several servlets. The following attributes are evaluated:	
	LIBRARY Name of the Java library (JAR or ZIP file) that contains the servlet. The name specification can be either absolute or relative. If the specification is relative, i.e. only the file name is specified without the path, the file itself must be in the current working directory. Specifying this is optional. If no Java library is specified, the Java class for the servlet must be located in the CLASSPATH.
	CLASSNAME Name of the Java class for the servlet.
	PATH Path under which the servlet is to be registered within the ServletContext.
	NAME Name of the servlet. Providing a name is optional and serves a purely informational purpose (the getServletName() method returns the name). If no specification is made, the name of the Java class is used as the servlet name.
	INITPARAM element A SERVLET element can contain any number of INITPARAM elements. There initialization parameters can be queried within the servlet by means of the servlet configuration, and apply to this servlet only. In terms of structure, the INITPARAM element for servlets is identical to the one for the ServletContext (see INITPARAM element).

An example of SERVLET elements

In this example, two servlets are loaded.

Java class MyServlet is loaded from the MYSVLT.JAR Java archive and registered under /myServlet within the ServletContext. The servlet can query the two initialization parameters. The name of the servlet matches that of the Java class.

A client references the servlet via http://<host>/servlet/myServlet.

The Java class TestServlet is loaded via the system class loader and registers within the ServletContext under the path /test . In this case, the Java class TestServlet must be located in the CLASSPATH of SIMOTION IT Virtual Machine.

The servlet has no initialization parameters. The name of the servlet is "A test servlet".

A client references the servlet via http://<host>/servlet/test

```
<SERVLET LIBRARY="MYSVLT.JAR" CLASSNAME="MyServlet" PATH="/
myServlet">
 <INITPARAM NAME="AParameter" VALUE="aValue"/>
 <INITPARAM NAME="AnotherParameter" VALUE="anotherValue"/>
</SERVLET>
```

```
<SERVLET CLASSNAME="TestServlet" PATH="/test"  NAME="A test servlet"/
>
```

Sample servlet and configuration

A sample servlet in Java source code format is included in the scope of delivery as a basic help method. Sample configuration JSERVER.XML may be used accordingly.

DemoServlet.java

The following section contains an example illustrating the basic principles of how a client request is edited within a concrete servlet.

```
/*
 * ================================================================
 * A simple servlet to demonstrate how to build a user-servlet.
 *
 * The servlet generates some "plain-text" only.
 *
 * Copyright (C) 2004 Siemens AG. All rights reserved.
 * ================================================================
 */
// Necessary servlet API
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
// Possible io exceptions
import java.io.IOException;
import java.io.PrintWriter;
/*
 * A simple servlet, which generates only a textual response.
 */
public class DemoServlet extends HttpServlet
{
    /*
     * Generate response for a GET-Request.
     */
    public void doGet( HttpServletRequest request,
                       HttpServletResponse response )
                throws ServletException, IOException
    {
        // "request" could be used to query additional
        // information
        // e.g. parameters ...

        // Set ContentType of the response
        response.setContentType("text/plain");

        // Obtain PrintWriter for writing response
        PrintWriter out = response.getWriter();
```

```
        // Write response
        out.println("Hello SIMOTION Servlets!");
    }
}
```

The sample servlet only generates simple text and does not require any parameters.

A servlet that wishes to have an entry form displayed in a browser generates the form in HTML and writes it to the browser as a response to a GET request. In order to edit the entry form, this type of servlet implements editing for the POST request as well, evaluates the form parameters within this, and generates a relevant response again.

JSERVER.XML

The configuration file supplied shows how the sample servlet can be loaded and how a DumpServlet contained in SIMOTION IT Servlets can be activated. This is useful for the purpose of viewing the current initialization parameters in the ServletContext.

```
<?xml version="1.0" standalone="yes"?>
<SIMSERV>

  <!-- Example of a SIMOTION IT Servlets configuration -->

  <!-- ServletContext parameters -->

  <INITPARAM NAME="ContextParam1" VALUE="ContextValue1"/>
  <INITPARAM NAME="ContextParam2" VALUE="ContextValue2"/>

  <!--
   The previously compiled demo servlet is loaded
   and must be zipped into SVLTDEMO.JAR and stored
   on the target system.
   The demo servlet is registered for path
   /demo, and can, therefore, by accessed by the client via http://
<host>/servlet/demo
  -->

  <SERVLET LIBRARY="SVLTDEMO.JAR" CLASSNAME="DemoServlet"
           PATH="/demo" NAME="Demo-Servlet"/>


  <!--
   No LIBRARY needs to be specified for the DumpServlet included as
standard.
   By way of a response, the DumpServlet provides information about
the current request and initialization
   parameters for the context and servlet (if specified),
   and can be registered for simple checking of the context
initialization parameters,
   for example.
   The dump servlet is registered for path
   /dump/*, for example, and can, therefore, be accessed by the
client via http://<host>/servlet/dump
   or even http://<host>/servlet/dump/APathInfo
```

```
  -->

  <SERVLET CLASSNAME="com.siemens.ad.SIMOTION.servlets.DumpServlet"
           PATH="/dump/*" NAME="Sample DumpServlet">
      <INITPARAM NAME="ServletParam1" VALUE="ServletValue1"/>
      <INITPARAM NAME="ServletParam2" VALUE="ServletValue2"/>
  </SERVLET>

</SIMSERV>
```

Restart of the servlet container

To stop the ServletContainer by calling an URL on the server, the so-called AdminServlet, a system servlet, must be loaded in the configuration file jserver.xml.

```
   <SERVLET
CLASSNAME="com.siemens.ad.SIMOTION.servlets.impl.AdminServlet"
            PATH="/admin/*" NAME="AdminServlet" />
```

The servlet is then registered on the subpath: "admin" and it knows the command (written as an http parameter):

```
   Action=StopContainer
```

By calling the URL

`<host><simotion-servlet-prefix>/admin?Action=StopContainer` (an HTTP-GET request is output)

the servlet container can be stopped.

The servlet restarts if the SIMOTION VM is restarted. This is done by calling the URL:

```
   <host><simotion-servlet-prefix>/JApp/jvmRestart
```

SIMOTION Programming 7

7.1 SIMOTION MCC Motion Control Chart

Preface

Scope

This document is part of the **SIMOTION Programming** documentation package.

This document applies to SIMOTION SCOUT, the engineering system of the SIMOTION product family in product version V5.3 in conjunction with:

- A SIMOTION device with the following versions of a SIMOTION Kernel:
  - V5.3
  - V5.2
  - V5.1
  - V4.5
  - V4.4
  - V4.3
  - V4.2
  - V4.1 [1]
  - V4.0 [1]
  - V3.2 [1]

  [1] V4.5 is the last product version of SIMOTION SCOUT that will support these versions of the SIMOTION Kernel.
- The relevant version of the following SIMOTION Technology Packages, depending on the kernel:
  - Cam
  - Path (Kernel as of V4.1)
  - Cam_ext
  - TControl

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.3:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

7.1.1 Fundamental safety instructions

7.1.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

7.1.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

7.1.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

7.1.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

7.1.2 Description

7.1.2.1 Overview

This chapter introduces the MCC (Motion Control Chart) graphical programming language and also describes the programming principles and procedures involved.

7.1.2.2 Introduction to MCC (Motion Control Chart)

MCC is a graphical programming language which has been designed to reduce the complexity of automating production machines.

Many production machines are very complex. They require a control system (SIMOTION) that is capable of handling motion control and technologies with a wide variety of motion functions, as well as PLC functions, arithmetic functions, and data management tasks.

MCC is the neutral description tool represented as a flowchart. This flowchart is referred to as "MCC chart" in this document. MCC provides you with all of the descriptive symbols you will need to define your automation task quickly and efficiently. It also offers a wide range of tools for structuring large-scale automation jobs.

With an MCC chart, you can create a program, a function block, or a function that complies with IEC 61131-3.

An MCC unit may contain no MCC charts, or one or more MCC charts. All the MCC charts contained in an MCC unit access the same data storage area.

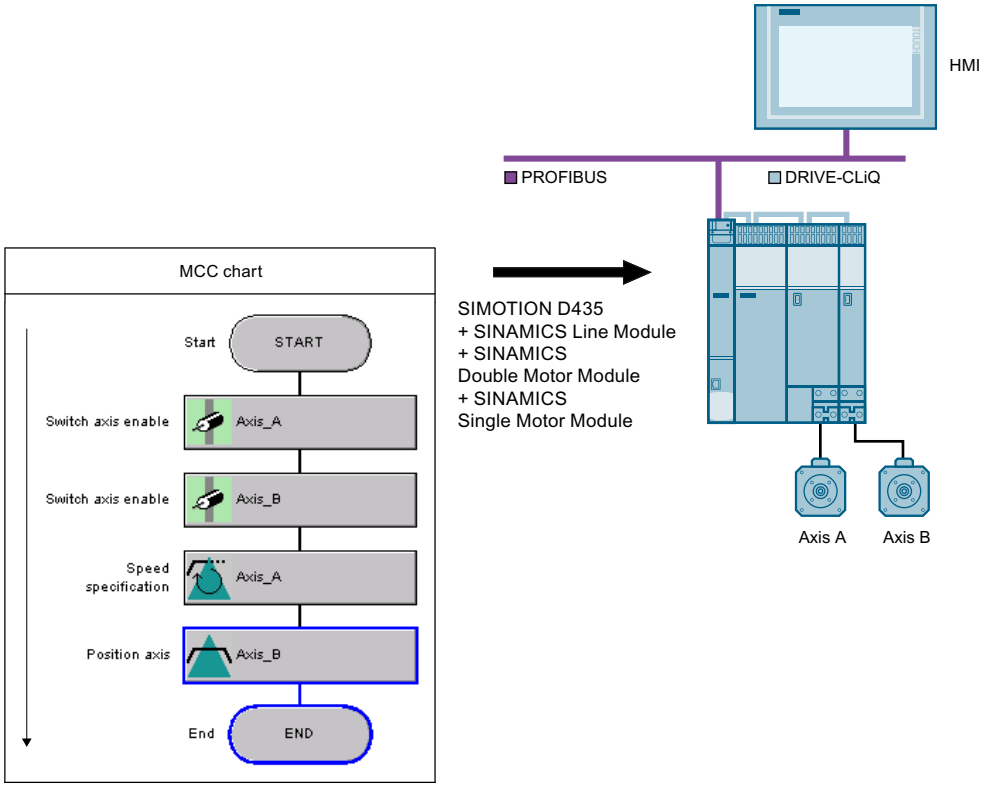


Figure 7-1 MCC for automating production machines

Application of MCC

MCC facilitates automation by offering the following features:

- The motion sequences on the machine can be easily programmed and are clearly defined.
- The programmer's logic is supported.
- Structured programming available through use of subprograms and modular creation of commands and library functions.
- Wait commands for quick responses to events.
- Simultaneous starting of axes.
- Online functions allow tracking of the program execution (program execution monitoring and breakpoints).
- Integrated online help.

These features help the inexperienced programmer to achieve results fast and let the experienced programmer create complex programs more efficiently.

7.1.2.3 Principles of programming

The SIMOTION motion control system offers powerful functions for motion control in production machines. As a graphical programming language, MCC helps you to formulate process and motion sequences easily by creating a sequence of graphical MCC commands.

The characteristics of every MCC command can be assigned individually. Control structures such as IF commands make for easy implementation of alternative branches.

The flowchart format makes it easy to follow the logical execution sequence. Furthermore, each MCC command has its own graphic design, making the functionality of each action within an MCC chart immediately evident. Consequently, the system greatly facilitates the programming of automation tasks and enhances readability. Supplementary online functions enable you to locate errors quickly or simply help you to keep track of the current states.

7.1.2.4 Procedure for programming

Let us look at an example of how MCC programming works.

Function

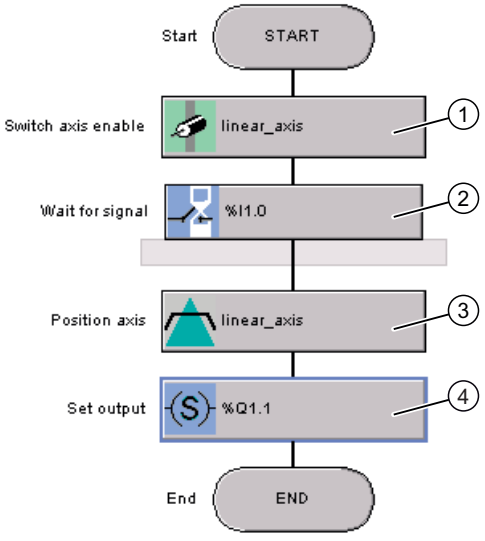
When you press a key, an axis will travel to position 1,000 mm at a velocity of 100 mm/s. Once this position has been reached, an output is to be set (for example, lamp ON).

This task is now broken down into several subtasks:

- Set axis enable signals
- Wait until the key is pressed
- Traverse axis to position
- Set output when motion ends

An MCC command is available for each of the listed subtasks. Each command is represented by a rectangular symbol in MCC. The parameters for individual commands (position = 1000, input = 1.0, etc.) are entered using a command dialog box. This opens when you double-click the command.

This example deals only with the MCC programming aspect. For this reason, information about the keys, axis, and lamp should already be available to the control system.



① Set enables:
drive enable, pulse enable, position controller enable
② Wait until the key with input I1.0 is pressed
③ Traverse the axis to position at the specified speed
④ Activate output Q1.1 for a lamp

Figure 7-2 MCC chart for example task

7.1.3 Software interface

7.1.3.1 User interface in MCC editor

This chapter describes the various operating features offered by the MCC editor. The user interface is graphically displayed and described.

7.1.3.2 Representation of MCC chart and MCC source file in Workbench

The Workbench is divided into 3 main windows:

- Project navigator: Displays the project structure
- Working area: Displays the MCC chart or MCC unit and parameter screen forms
- Detail view: Displays the variables, alarms, error messages, etc., depending on which element is selected in the project navigator

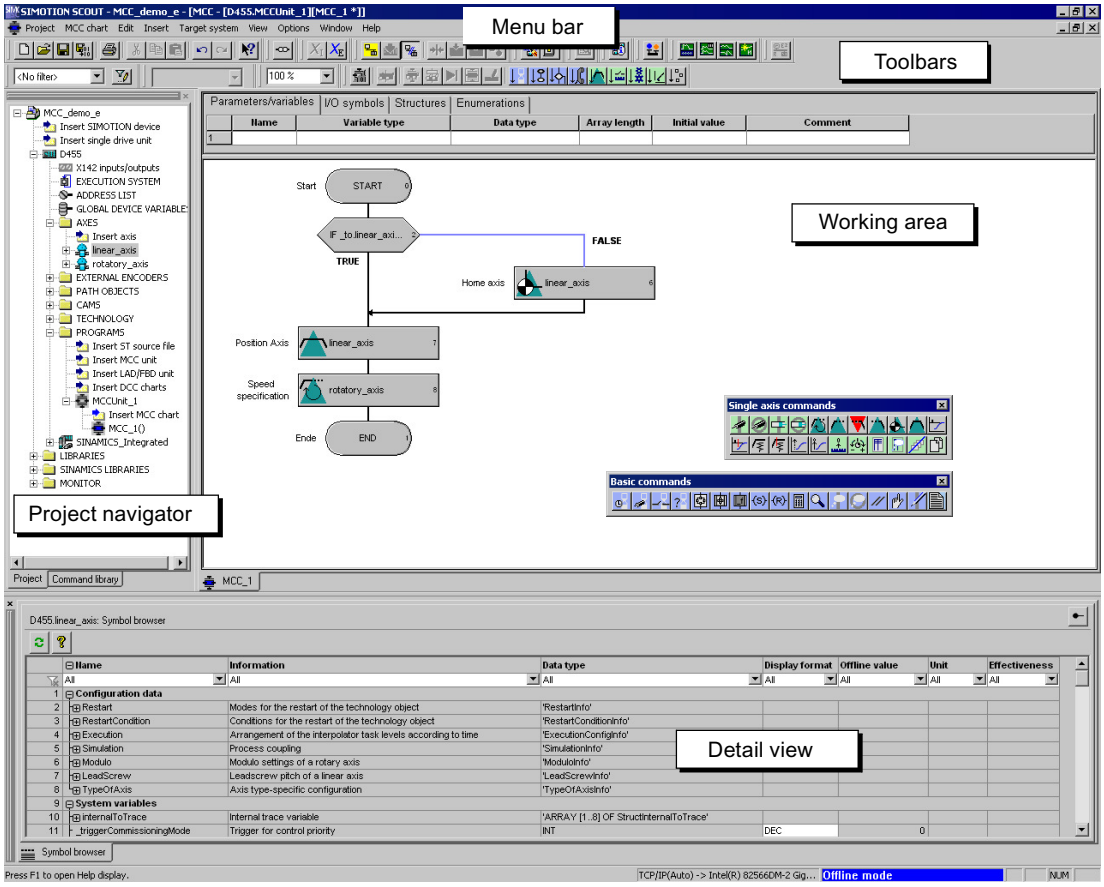


Figure 7-3 Workbench view

For additional information on using the Workbench:

- Maximizing working area and detail view
- Zooming in and out of an MCC chart
- Moving the MCC chart or MCC unit to the foreground
- Hiding and displaying the declaration table
- Enlarging/reducing the declaration table

Maximizing working area and detail view

The windows working area and detail view can be set to maximum zoom.

Select **View > Maximize working area** or **View > Maximize detail view** from the menu.

Zooming in and out of an MCC chart

There are various options available for changing the size of an MCC chart, i.e. the size of the command icons:

- Changing the zoom factor via the **Zoom factor** combo box.
  Select a factor from the list, or enter an integer value of your own choice.
- Press the **Ctrl** key while moving the mouse wheel.

The zoom factor selected applies to the declaration table and the MCC chart.

The changes always apply to the currently selected MCC chart.

Moving the MCC chart or MCC unit to the foreground

If several MCC charts or MCC units are open in the working area, they will be cascaded in most cases. Therefore, only the top window will be visible.

You can use different methods to move concealed windows to the foreground:

- By selecting the appropriate tab below the working window
- By selecting the appropriate name in the Window menu

Hiding and displaying the declaration table

If you need more space, you can completely hide the declaration table of an MCC chart.

- Double-click the separation line.

In order to display the declaration line again, double-click the separation line again.

Enlarging/reducing the declaration table

- Move the pointer over the separation line until it turns into a double line.
- Keep pressing the left mouse button while you move the separation line:
  - Up, to reduce the declaration area
  - Down, to enlarge the declaration area

7.1.3.3 Operator input options

The MCC editor provides the programmer with a variety of different operator input options. Alternatives for executing individual operator inputs include the following:

- Via the menu bar
- Via the context menu
- Via the toolbar

- Via key combinations
- Texts and variables can be moved into the input field using a drag and drop operation.

Menu bar

There are two separate menu bars for the MCC unit and the MCC chart. Each contains a complete set of action commands.

The appropriate menu bar is displayed for the active window in the working area.

Context menu

The context menu of an MCC chart opened in the working area contains the complete range of commands, see Inserting commands (Page 4013).

To use the context menu for an object, proceed as follows:

1. Select the appropriate object with the left mouse button.
2. Briefly click the right mouse button.
3. Left-click the appropriate menu item.

Note

In this document, work steps and action commands are executed wherever possible via the context menu. However, some action commands can only be executed via the menu bar or toolbar.

Toolbar

The toolbars contain important operator input options, for saving or pasting commands, for example. The toolbars can be positioned as required within the Workbench.

Using the menu **View > Toolbars**, you can display or hide these.

The MCC editor toolbar contains the full range of MCC commands. The list of commands is displayed when you place the cursor on the appropriate button, see Inserting commands (Page 4013).

Key combination

You can input commands quickly in the MCC editor using key combinations. The key combinations available in the MCC editor are listed in the appendix titled Key combinations (Page 4574).

The online help system is called up with Shift+F1 or F1.

Drag and drop

Variables can be moved from the detail view (*Symbol browser* tab) to the input field using a drag and drop operation.

Left-click the line number of the variable you wish to move. The line with the variable is selected. Keeping the left mouse button pressed, drag the line number into the input field of the parameter screen form. As soon as you release the left mouse button, the variable will be inserted at the appropriate position.

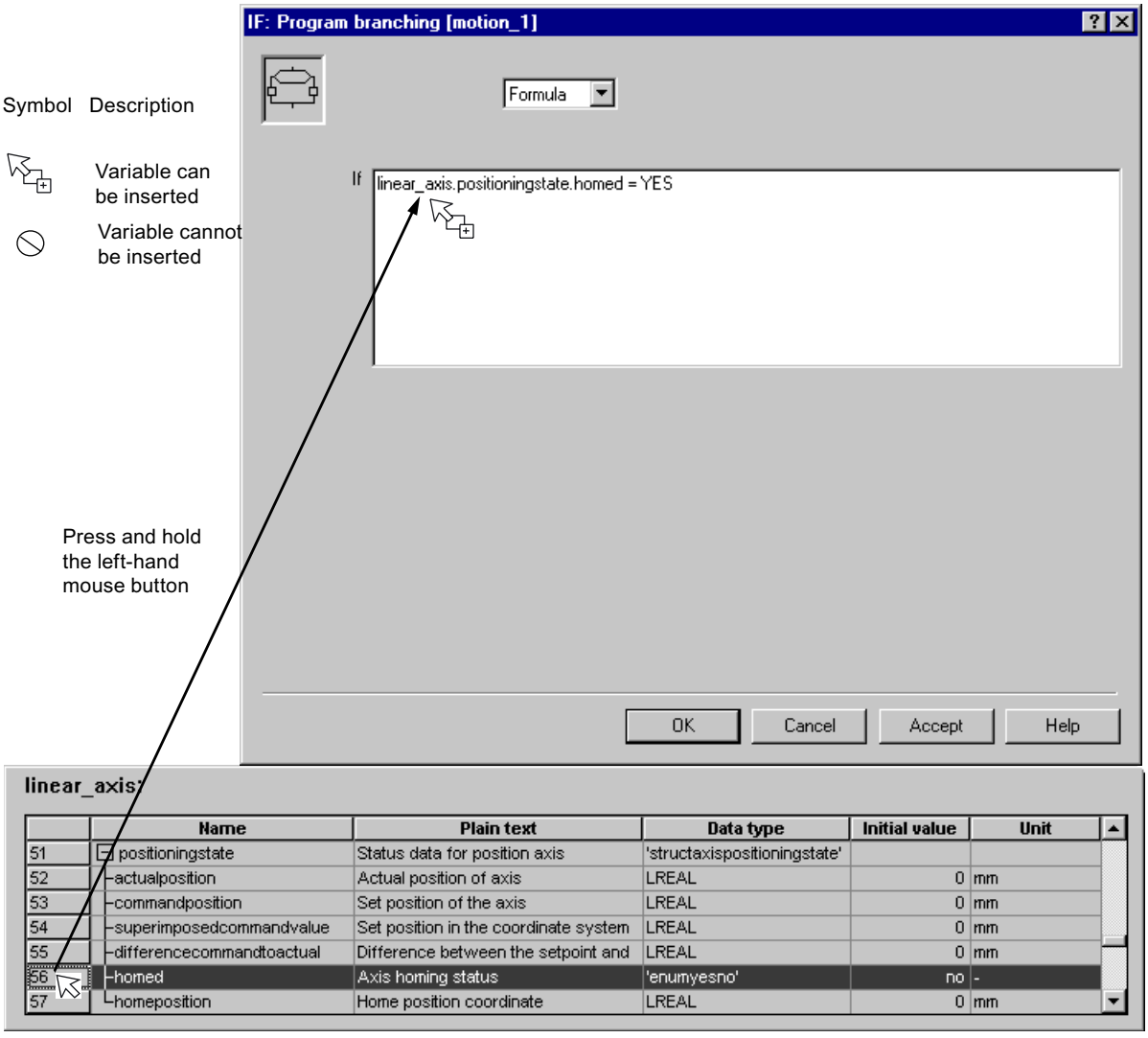


Figure 7-4 Inserting a variable using drag and drop

Automatic completion (Autocomplete)

In the MCC editor, you can automatically complete identifiers. A drop-down list box with identifiers that begin with the previously entered characters will be displayed. This operates on a context-sensitive basis, whereby the expected type for the identifier being sought and its visibility in the current program context determine which entries are displayed as options in the drop-down list box.

Depending on the context, the entries displayed in the drop-down list box are filtered and sorted:

- The filtering process may determine, for example, that only structure components are displayed for a structure.
- Entries are sorted according to their relevance to the context, with the more relevant identifiers appearing higher in the list (e.g. local variables are listed before global variables).

With MCC units and MCC charts, the identifiers can be completed automatically in the following input fields/editable drop-down list boxes:

- Declaration table of the MCC unit
  - "Variable type" column
  - "Data type" column
  - "Type" column
- Declaration table of the MCC chart
  - "Variable type" column
  - "Data type" column
- IF, WHILE, UNTIL, synchronous start MCC commands:
  - Condition in the Formula programming language
- FOR MCC command
  - "Variable" input field (completing a variable name)
  - "Start", "End", and "Increment" input field (completing an identifier within an expression)
- ST zoom MCC command
  - Editing in the ST programming language is supported here, so that automatic completion works the same as with the ST editor.
- Variable assignment MCC command
  - "Variable" column (completing a variable name)
  - "Expression" column (completing an identifier within an expression)
- Subprogram call MCC command
  - "Subprogram" editable drop-down list box
    A connection to the program source (if required) is automatically set up when the subprogram is selected.
  - "Library" editable drop-down list box
    A connection to the library (if required) is automatically set up when the subprogram is selected.
  - "Instance" editable drop-down list box
  - "Value" column
    A connection to the program source/library (if required) is automatically set up when the subprogram is selected.

- System function call MCC command
  - "System function" input field
  - "Return value" input field
  - "Value" column
- Input fields/editable drop-down list boxes in the parameter screen forms for the MCC commands
  All inputs in the input fields/editable drop-down list boxes of the parameter screen forms are supported by automatic completion.

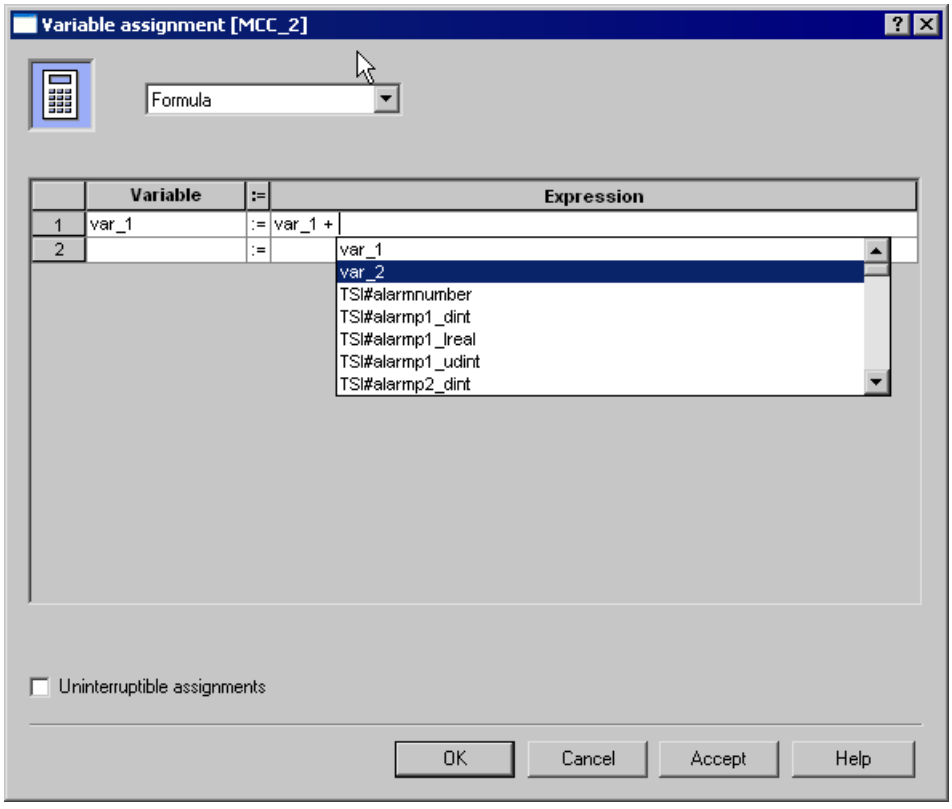


Figure 7-5 Automatic completion of an identifier within an expression

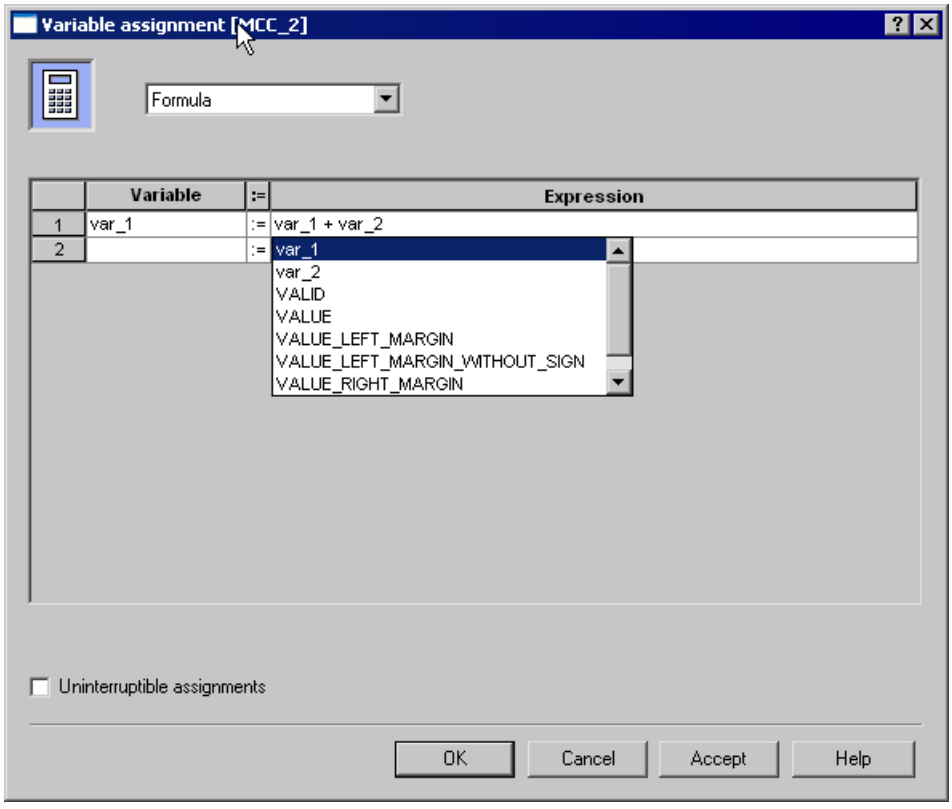


Figure 7-6 Replacing an identifier with another identifier

Procedure

To complete an identifier automatically (Autocomplete), proceed as follows:

1. Write the first characters of the identifier (e.g. the first letters of a word) in the input field/ editable drop-down list box. Alternatively, you can place the cursor anywhere in an existing identifier.
2. Press the **Ctrl+space** key combination.
   A drop-down list box opens containing the filtered/sorted identifiers for the current context. The identifiers listed start with the characters input so far or the characters up to where the cursor is positioned within an existing identifier.

   Note

   When the drop-down list box has been expanded and is editable (by clicking the ▾ symbol), automatic completion is already activated.

3. Expand or refine the options displayed:
   - Enter additional characters.
   - Delete characters.
   - Move the cursor with the left/right arrow keys.

4. Select the required identifier with the up/down arrow keys.
5. Press the **Return** key.
   The identifier is accepted by the input field/editable drop-down list box and the current word is overwritten.

Note

If only a single identifier is offered for selection, the selection window will not be opened and the identifier completed immediately.

Functional description

The following identifiers that begin with the specified characters will be offered:

- ST expressions
- ST program sections
- Identifiers from the command library
- Library, unit, POU, or system function names
- For technology objects including their system variables and configuration data
- Identifiers for the relevant MCC unit / MCC chart:
  - Program organization units (POU)
  - Data types
  - Variables and constants
  - Structure elements
- Identifiers from imported program sources

Note

Identifiers from the relevant MCC unit / MCC chart or from imported program sources will only be displayed correctly if the corresponding program source has been compiled.

The display operates on a context-sensitive basis, i.e. only those types of identifiers that are appropriate at the associated location of the MCC unit / MCC chart are offered:

- Within a declaration table, data types and variable types only
- Within a program organization unit (POU), no data types
- For a structure, structure components only (e.g. var_struct.xx)

7.1.3.4 Settings for the MCC editor

You can adjust important properties of the MCC editor to suit your individual requirements.

1. Select the menu **Options > Settings**.
   The **Settings** dialog box opens.
2. Select the **MCC editor** tab.
3. Make the required settings here.
4. Click **OK** or **Accept** to confirm.

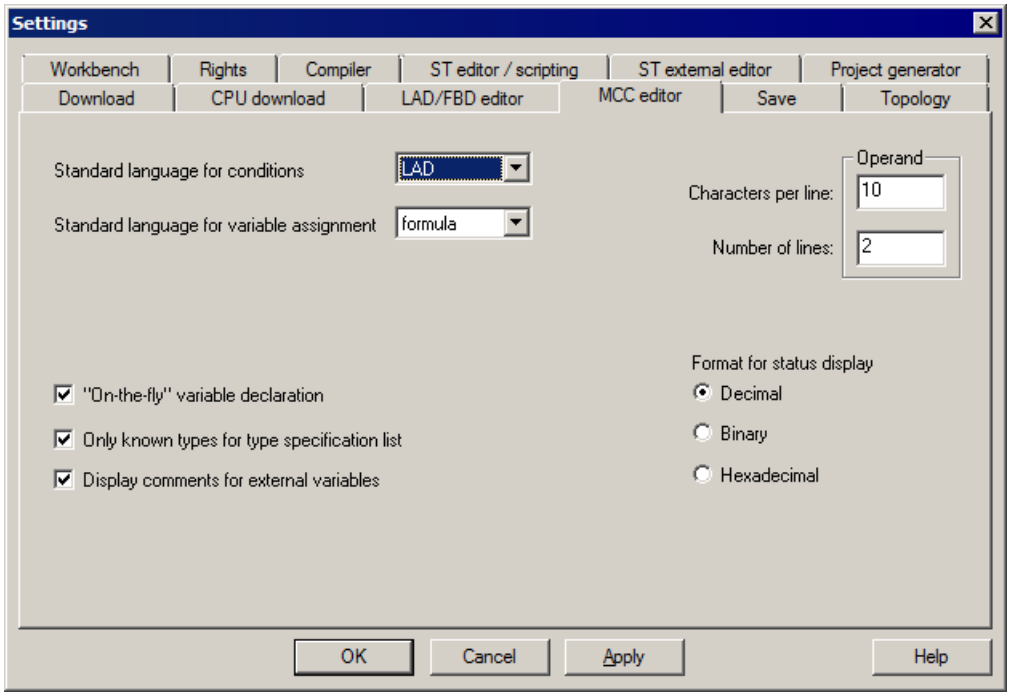


Figure 7-7 Settings for the MCC editor

The table below contains a description of the individual parameters.

Table 7-1 MCC editor parameter settings

Parameter	Description
Standard language for conditions	In the case of conditions, you can define the language set on opening. Possible settings are LAD, FBD, or Formula.
Standard language for variable assignment	In the case of a variable assignment, you can define the language set on opening. Possible settings are LAD, FBD, or Formula.
"On-the-fly" variable declaration	If this is active, a dialog box appears when an unknown symbol is input in the parameter screen form for an MCC command. You can perform the variables declaration in this box (see Defining global user variable and local variable in the Variables declaration dialog box ("on-the-fly" variables declaration)) (Page 4067).

Parameter	Description
Only known types if type lists exist	You can choose which function blocks are displayed as data types in the declaration tables for the MCC units and MCC charts: • When the checkbox is activated, the declaration table contains only function blocks that are defined in the same MCC unit or in linked source files or libraries (see Connections to other program sources or to libraries (Page 4114)). • When the checkbox is deactivated, the declaration table contains all function blocks defined in the project.
Display comments for external variables	If activated, the comments specified in the declaration table are displayed in the MCC editor for variables. The comment is displayed in a tool tip at the point where the variable is being used within the parameter screen form for an MCC command. This also applies to external variables declared in another MCC unit or LAD/FBD source file and connected to the current MCC unit. The other MCC unit or LAD/FBD source file needs to have been recompiled beforehand. Comments for variables declared in ST source files cannot be displayed. This also applies to variables of other source files connected to the current MCC unit via an ST source file.
Operand	You can change the options for displaying the operand in the case of conditions or variable assignments (Page 4152) for the LAD and FBD languages by entering the **Number of characters per line** and the **Number of lines** for the Operand field.
Format for status display	Format in which the variable values with bit data type are displayed for program status (Page 4542) and variable status (Page 4530).

7.1.3.5 Calling up the online help

The online help can provide assistance for many of the operating steps.

Call up the online help using either the:

- **Help** menu
  - Help topics
  - Context-sensitive help
  - Getting Started
- Using the **Help** button in an open parameter screen form
- Using the **F1** button for general help
- Using keystroke combination **Shift+F1** for context-sensitive help

7.1.4 MCC Source Files and MCC Charts

This chapter describes how to create and work with an MCC unit containing MCC charts.

7.1.4.1 General

MCC units are assigned to the SIMOTION device on which the programs (MCC charts) contained in the MCC unit will subsequently be run (e.g. SIMOTION D455-2). They are stored in the project navigator under the SIMOTION device in the PROGRAMS folder.

MCC charts are the individual program organization units (program, function, function block) in an MCC unit. They are stored under the MCC unit in the project navigator.

Note

ST source files, LAD/FBD units and DCC charts are also stored in the **PROGRAMS** folder under the SIMOTION device.

For a description of the SIMOTION ST (Structured Text) programming language, refer to the SIMOTION ST Programming and Operating Manual.

For a description of the SIMOTION LAD (Ladder Diagram) and SIMOTION FBD (Function Block Diagram) programming languages, refer to the SIMOTION LAD/FBD Programming and Operating Manual.

7.1.4.2 Inserting and managing MCC source files

Inserting a new MCC source file

MCC units are assigned to the SIMOTION device on which the programs (MCC charts) contained in the unit will subsequently be run (e.g. SIMOTION D455-2). They are stored in the project navigator under the SIMOTION device in the PROGRAMS folder.

The individual program organization units (POU, MCC charts) are stored under an MCC unit.

You can insert a new MCC unit in the following ways:

- In the project navigator: in the **PROGRAMS** folder using the **Insert MCC unit** element
- Select the **PROGRAMS** folder in the project navigator followed by **Insert > Program > MCC unit**
- Select the **PROGRAMS** folder in the project navigator followed by **Insert new object > MCC unit** in the context menu

Procedure

To insert a new MCC unit using the context menu, proceed as follows:

1. Open the appropriate SIMOTION device in the project navigator.
2. Select the **PROGRAMS** folder.
3. Select the **Insert new object > MCC unit** context menu.

4. Enter the name of the MCC unit.
   The names of program sources must comply with the rules for identifiers (Page 4047): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters.
   The permissible length of the name depends on the SIMOTION Kernel version:
   - SIMOTION Kernel as of version V4.1: Maximum 128 characters.
   - SIMOTION Kernel up to version V4.0: Maximum 8 characters.

   Names must be unique within the SIMOTION device. Protected or reserved identifiers are not permitted.
   Existing program sources (e.g. ST source files, MCC units) are displayed.
5. If necessary, select the **Compiler** tab and make any local compiler settings; see Local compiler settings (Page 3997).
6. You can also enter an author, version, and a comment.
7. Activate the **Open editor automatically** checkbox.
8. Confirm with **OK**.

Note

When you click **OK**, the MCC unit will only be transferred to the project. The data, together with the project, is only saved to the data carrier if you select, for example, **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

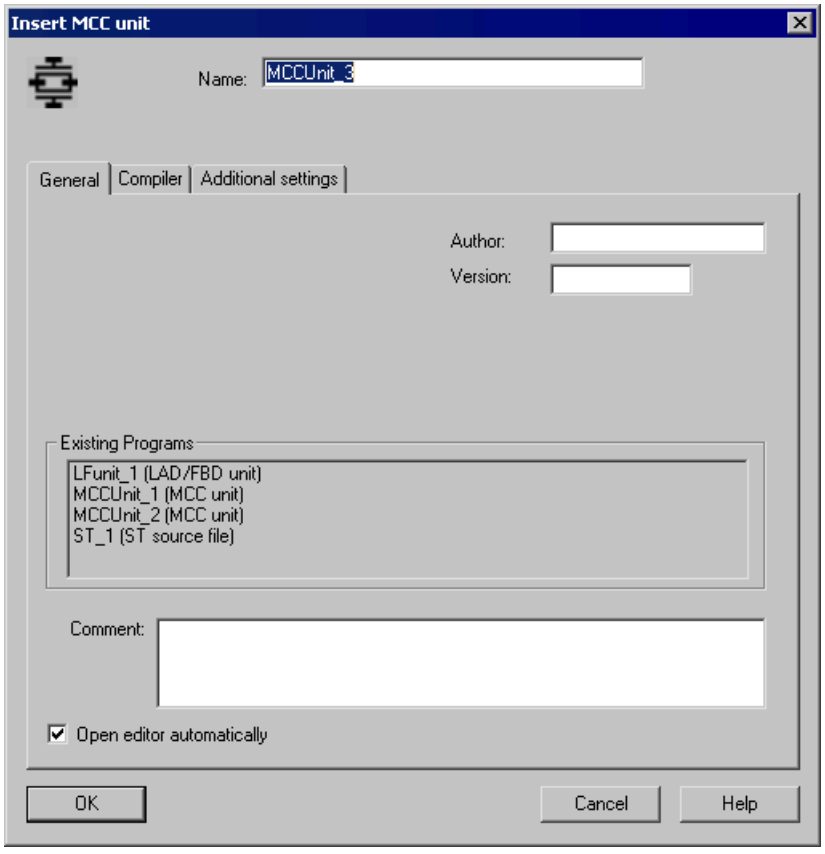


Figure 7-8 Creation dialog for a new MCC unit

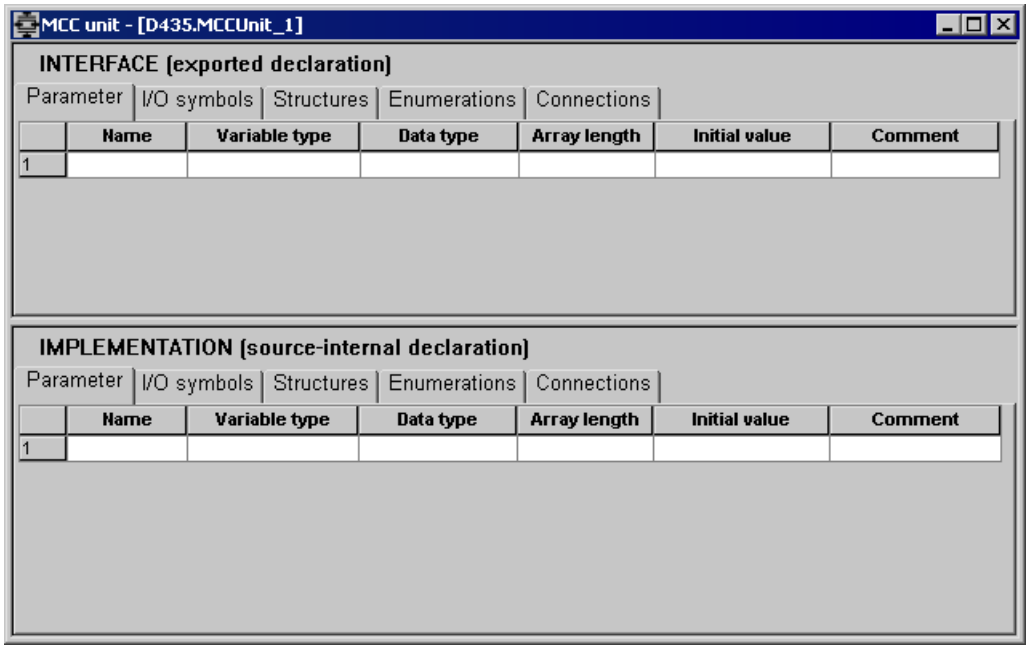


Figure 7-9 New MCC unit (declaration table for the interface and implementation sections)

Opening existing MCC source files

Procedure

To open an MCC unit:

1. Open the subtree of the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Select the required MCC unit.
4. Select the **Open** context menu.
5. Only for MCC units with know-how protection (Page 3992):
   If the MCC unit (or a chart for this unit) is not already open and the login assigned to the MCC unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (until the unit and all its charts are closed).
   - If required, activate the "Use login as a standard login" checkbox.
     You will be logged in with this login and can now open additional units to which the same login is assigned without having to re-enter the password.

The MCC unit (declaration table) is opened in the working window. Multiple units can be opened.

Note

You can also double-click the required MCC unit to open it.

Transferring and compiling an MCC source file

Procedure:

1. To transfer an MCC unit with its associated MCC charts to the project and start its compilation, proceed as follows:
2. Make sure that the MCC unit or one of the associated MCC charts is the active window in the Workbench.
   Select one of the following menu commands:
   - **Accept and compile** button in the **MCC unit** or **MCC editor** toolbar
   - **MCC unit > Accept and compile** menu item or **MCC chart > Accept and compile** menu item

Alternative way of saving and compiling:

- Select the MCC unit or an MCC chart in the project navigator; then, in the context menu, select **Accept and compile**.

Note

The **Accept and compile** command transfers the changes in the MCC unit and associated MCC charts to the project only. The data, together with the project, is only saved to the data carrier if you select **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

As of Version V4.2 of SIMOTION SCOUT, you no longer need to observe the POU order in the MCC unit, although the following exception applies:

When a project is saved in a format older than Version V4.2 of SIMOTION SCOUT, the MCC chart order in the MCC unit is of relevance as far as compilation is concerned. A subprogram (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the MCC chart of the subprogram appears above the chart in which it is used in the project navigator. If necessary, resort the charts (see Subprogram call of function (FC) (Page 4129)).

Before saving a project in a format older than Version 4.2, you can check the project for downward compatibility (including the correct order of the POUs) via **Project > Old project format > Check the project for downward compatibility**.

You can also save (export) an MCC unit outside the project (see Exporting and importing MCC units (Page 3992)).

Error messages and warnings relating to compilation are displayed in the Compile/check output tab in the detail view.

Note

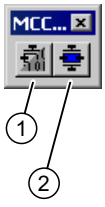
If certain commands (for example, Synchronous Start, Uninterruptible Variable Assignment) are included in the MCC chart, *UserInterruptTask_1* must be configured in the execution system in order to ensure error-free compilation. You must program the error reactions to these commands in *UserInterruptTask_1*.

If you receive an applicable error message, proceed as follows:

1. Insert an MCC chart for a program, and program the appropriate error reactions.
2. Assign the program to *UserInterruptTask_1*.

MCC unit toolbar

This toolbar contains important command buttons for an MCC unit:



① Accept and compile
② Insert MCC chart

Figure 7-10 MCC unit toolbar

This toolbar is active if an MMC unit is visible in the foreground of the Workbench working area.

Closing an open MCC source file

To close an open MCC unit in the working window, do one of the following:

- Click the **x** button in the header
- Select the **MCC unit > Close** menu command
- Select the **Window > Close all windows** menu command

If the changes have not yet been saved in the project, you can save or cancel them, or abort the close operation.

Cutting, copying, and deleting an MCC source file

You can cut or copy an MCC source file and all of its associated MCC charts and paste them under the same or a different SIMOTION device.

If the MCC source file is deleted after it is copied, its data will also be deleted from the clipboard and the MCC source file can no longer be inserted.

Note

When an MCC unit or MCC chart is copied under a SIMOTION device with an earlier version of SIMOTION Kernel, the reliability of commands and their parameterization is only checked when the MCC chart is opened or when the MCC unit or the project is compiled.

Procedure

Follow these steps:

1. Select the MCC source file in the project navigator.
2. In the context menu, select the appropriate item (**Cut**, **Copy**, or **Delete**).

Pasting a cut or copied MCC source file

Procedure

To insert a cut or copied MCC unit, proceed as follows:

1. Under the SIMOTION device, select the **PROGRAMS** folder.
2. In the context menu, select **Paste**.
   The MCC unit is inserted (with a different name, if necessary).
3. Change the name, if necessary (see Renaming MCC source files (Page 3995)).

Note

When an MCC unit or MCC chart is copied under a SIMOTION device with an earlier version of SIMOTION Kernel, the reliability of commands and their parameterization is only checked when the MCC chart is opened or when the MCC unit or the project is compiled.

Know-how protection for MCC source files

You can protect MCC units from access by unauthorized third parties. Protected MCC units and all associated MCC charts can only be opened or exported as ST source files when the password is entered.

The SIMOTION online help provides additional information on know-how protection.

Note

If you export in XML format, the MCC units are exported in an encrypted format. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

Exporting and importing an MCC source file

The export and import functions offer you the option of saving an MCC unit outside the project on your hard disk so that you can copy it from there into another project.

You can export an MCC unit as a text file for the SIMOTION ST (Structured Text) programming language. You can either import this file as an ST source file or edit it with any ASCII editor.

You can also export/import an MCC unit as an encoded XML file.

Exporting an MCC source file as an ST source file

You can export an MCC unit as a text file for the SIMOTION ST (Structured Text) programming language. You can either import this file as an ST source file or edit it with any ASCII editor.

Procedure

Follow these steps to export an MCC unit as an ST source file in ASCII format:

1. Open the MCC unit (Page 3989), entering the password if necessary (for MCC units with know-how protection (Page 3992)).
2. Make sure the declaration table for the MCC unit is the active window.
3. Select the **MCC unit > Export as ST** menu command.
4. Enter the path and file name for the ST source file and click **Save** to confirm.

The MCC unit is saved as an ST source file in ASCII format; the file name is given the default extension *.st

Alternatively, you can also proceed as follows:

1. Select the MCC unit in the project navigator.
2. Select the **Export as ST** context menu.
3. Only for MCC units with know-how protection (Page 3992) and which are not already open: If the MCC unit (or a program of this unit) is not already open and the login assigned to the MCC unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (for this export).
   - If required, activate the **Use login as a standard login** checkbox.
     You will be logged in with this login and can now export or open additional units to which the same login is assigned without having to re-enter the password.
4. Enter the path and file name for the ST source file and click **Save** to confirm.

Note

Text files for the ST programming language cannot be imported as MCC units.

An MCC unit with know-how protection is exported without protection.

Exporting an MCC source file in XML format

Procedure

This operation allows you to save an MCC unit in encoded form in a directory outside the project.

1. Select the MCC unit in the project navigator.
2. Select the **Expert > Save project and export object** context menu.
3. Select the directory for the XML export and confirm with **OK**.

Note

Know-how-protected MCC units can also be exported in XML format. The know-how protection is retained when the files are imported. The MCC units are exported in encrypted format. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

Importing an MCC source file from XML data

Procedure

This operation allows you to import MCC source files that were exported as encoded files in XML format.

1. In the project navigator, select the **PROGRAMS** folder.
2. In the context menu, select the **Import object** command.
3. Select the XML data to be imported.

The entire project is saved and recompiled.

Note

Individual objects (e.g. MCC source files) from a project that was exported as XML data cannot be imported selectively.

Properties of an MCC source file

Procedure

1. Under the SIMOTION device, open the **PROGRAMS** folder.
2. Select the required MCC unit.
3. Select the **Edit > Object properties** menu command.
4. If necessary, select further tabs to make local settings (only valid for this MCC unit):
   - **General** tab: General details on the MCC unit, e.g. time stamp of the last change, storage location of the project (see figure).
   - **Compiler** tab: Local settings of the compiler (Page 3997) for code generation and message display.
   - **Additional settings** tab: Display of the compiler options in accordance with the current compiler settings (refer to SIMOTION ST Programming and Operating Manual).
   - **Compilation** tab: Display of the compiler options the last time the MCC unit was compiled (refer to SIMOTION ST Programming and Operating Manual).
   - **Object address** tab: Set the internal object address of the MCC unit. The object addresses of the other program sources are displayed.

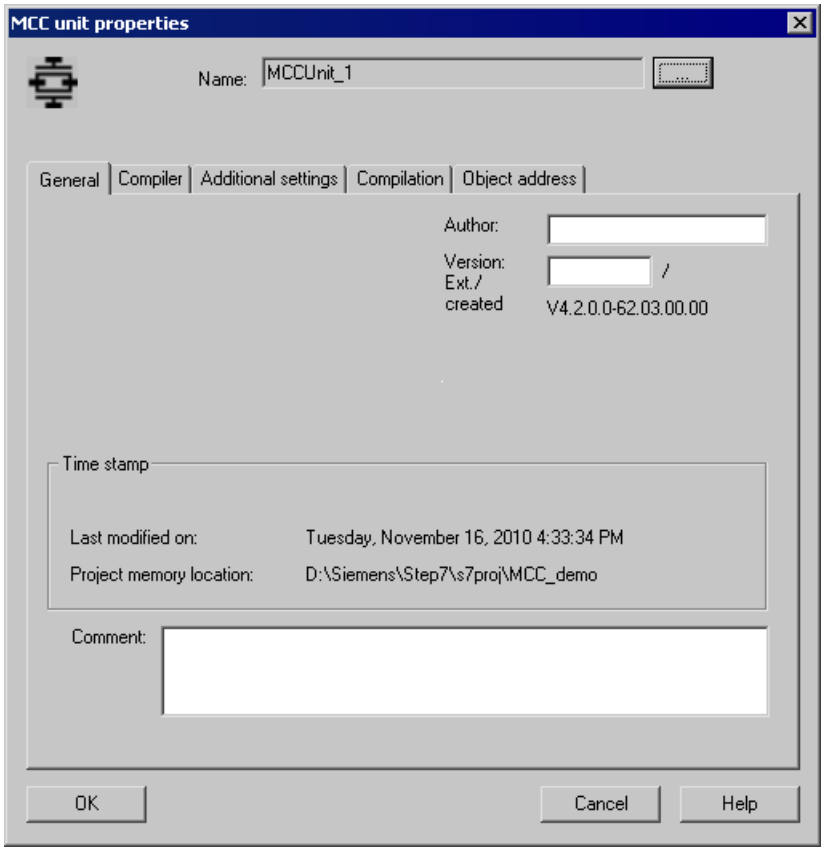


Figure 7-11 Properties of an MCC unit

Renaming MCC source files

To rename an MCC source file, proceed as follows:

1. Open the Properties window of the MCC source file.
2. Click ... .
3. Click **OK** to confirm.

using test functions

You can make use of various test functions during program execution. Additional code must be generated for some test functions (e.g. program status, single step or trace). You enter these settings in the compiler options:

1. Select the **Compiler** tab.
2. Make the necessary settings (see Local compiler settings (Page 3997)).

These functions facilitate debugging of your program.

Configuring compiler settings

You can configure the compiler settings as follows:

- Globally for the SIMOTION project, always valid for all programming languages, see Global settings of the compiler (Page 3996)
- Locally for an individual MCC source file within the SIMOTION project, see Local settings of the compiler (Page 3997)

Global compiler settings

The global settings are always valid for all programming languages within the SIMOTION project. If there are global settings which only apply to specific programming languages, this is specified on the **Compiler** tab.

Procedure

1. Select the menu **Options > Settings**.
2. Select the **Compiler** tab.
3. Make the settings in accordance with the parameter description in the SIMOTION ST Programming and Operating Manual.
4. Confirm with **OK**.

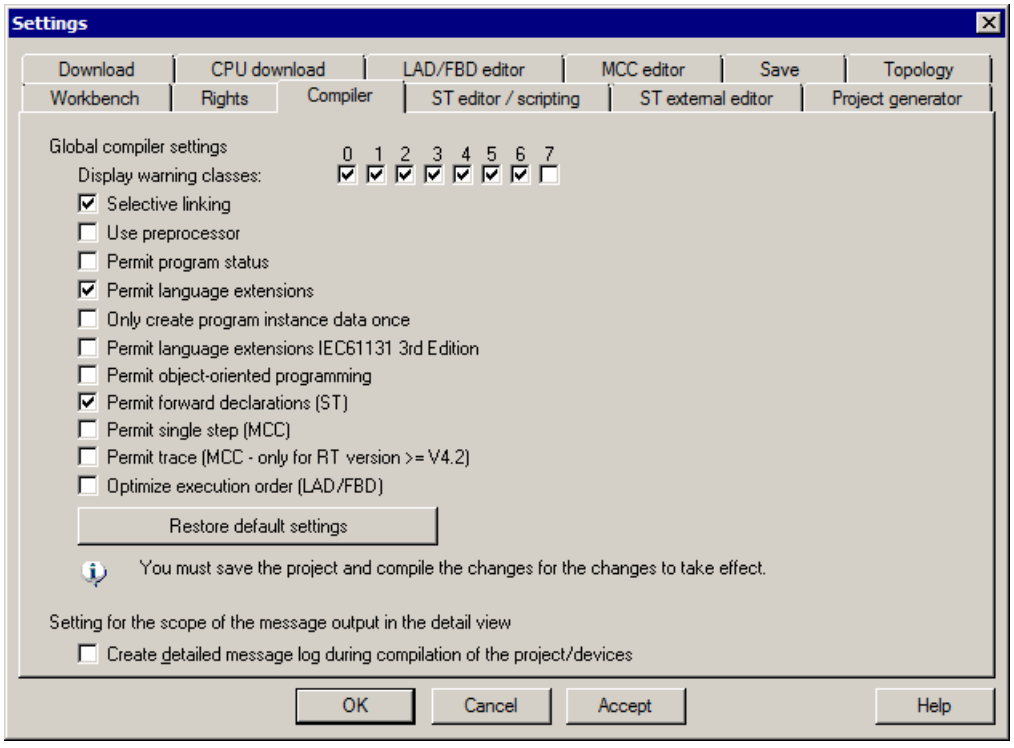


Figure 7-12 Global compiler settings

Parameter

For a description of the parameters of the global compiler settings, refer to SIMOTION ST Programming and Operating Manual.

Local compiler settings

Local settings are configured individually for each MCC source file; local settings overwrite global settings.

Procedure

To select the compiler options, proceed as follows:

1. Open the Properties window for the MCC source file, see Properties of an MCC source file (Page 3994).
2. Select the **Compiler** tab.
3. Define the settings according to the following table.
4. Click **OK** to confirm.

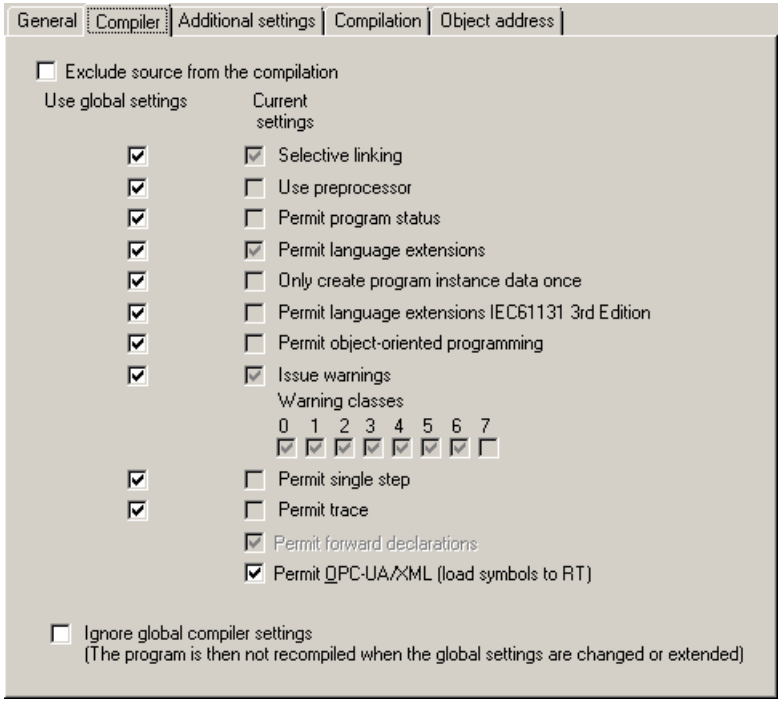


Figure 7-13 Local compiler settings for MCC source files in the Properties window

The current compiler options (the combination of global and local compiler settings which currently applies) for the program source are displayed on the **Additional settings** tab. The compiler options used the last time the program source was compiled can be seen on the **Compilation** tab.

The SIMOTION ST Programming and Operating Manual contains further information on what the compiler options mean.

Table 7-2 Local compiler settings

Parameter	Description
Exclude the source from the compilation	**Active**: This source is not compiled upon compilation of the project, the device or the library (e.g. Menu **Project > Save and recompile all**). The source is marked accordingly in the project navigator. Corresponding information is provided upon compilation. **Inactive** (standard): The source is compiled upon compilation of the project, the device or the library.
Use global settings	This checkbox is available for every parameter which also has a global setting. This is where you define whether the global settings are adopted or whether the local settings will apply. See the description under Effectiveness of global or local settings in the SIMOTION ST Programming and Operating Manual. Use the second checkbox or the other checkboxes for the relevant parameters (described below) to define the local settings.
Selective linking[1]	**Active**: Unused code is removed from the executable program. **Inactive**: Unused code is retained in the executable program. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).
Use preprocessor[1]	**Active**: Preprocessor is used, see SIMOTION ST Programming and Operating Manual. **Inactive**: Preprocessor is not used. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).
Enable program status[1]	**Active**: Additional program code is generated to enable monitoring of program variables (including local variables). **Inactive**: Program status not possible. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Program status (Page 4542).
Permit language extensions[1]	**Active**: Language elements are permitted that do not comply with IEC 61131-3. • Direct bit access to variables of a bit data type, see the SIMOTION ST Programming and Operating Manual. • Accessing the input parameter of a function block when outside the function block, see the SIMOTION ST Programming and Operating Manual. • Calling a program while in a different program, see the SIMOTION ST Programming and Operating Manual. **Inactive**: Only language elements are permitted that comply with IEC 61131-3. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).

Parameter	Description
Only create program instance data once[1]	**Active**: The local variables of a program are only stored once in the user memory of the unit. This setting is required when calling a program while in a different program, see the SIMOTION ST Programming and Operating Manual. **Inactive**: The local variables of a program are stored according to the task assignment in the user memory of the associated task. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Memory areas for the variable types in the SIMOTION ST Programming and Operating Manual. For further information, refer to the SIMOTION Basic Functions Function Manual.
Permit language extensions, IEC61131 3rd edition[1]	**Active**: Additional language elements can be used in accordance with IEC 61131-3 3rd edition. The corresponding keywords are locked as reserved or protected identifiers. • Nested block comments (e.g. in the ST magnifying glass (Page 4188)) • MCC command Continue (Page 4217) • Standard-compliant system functions LOWER_BOUND and UPPER_BOUND for determining the limits of a dynamic ARRAY • Additional system functions: – FROM_BIG_ENDIAN – FROM_LITTLE_ENDIAN – IS_VALID – TO_BIG_ENDIAN – TO_LITTLE_ENDIAN **Inactive**: The additional language elements cannot be used. The corresponding keywords are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.
Permit object-oriented programming[1]	**Active**: The keywords for object-oriented programming according to IEC 61131-3 3rd edition are locked as reserved or protected identifiers. General references (Page 4086) can also be formed. **Inactive** (standard): The keywords for object-oriented programming are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** Independently of the setting the program organization units created in ST sources using object-oriented programming (e.g. classes, methods) can be used in this source. No program organization units can be created in MCC using object-oriented programming. When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

Parameter	Description
Issue warnings Warning classes[1]	In addition to error messages, the compiler can issue warnings and information. You can set the scope of the warning messages to be issued. "Issue warnings" checkbox: **Active**: The compiler issues the warnings and information according to the warning class selection that follows. **Inactive**: The compiler suppresses all warnings and information concerning this source file. The checkboxes for the warning classes are hidden. **Gray background** (display only): Operating on a global setting basis, the compiler always issues warnings and information in accordance with the global warning class selection shown below (if "Use global settings" = active). "Warning classes" checkboxes (only if "Issue warnings" = active): **Active**: The compiler issues warnings and information for the selected class. **Inactive**: The compiler suppresses warnings and information for the associated class. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See also Meaning of the warning classes in the SIMOTION ST Programming and Operating Manual.
Permit single step[1]	**Active**: An additional program code is created, which enables individual program steps to be monitored. **Inactive**: Single step is not possible. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). This function facilitates debugging of your program. See Tracking individual program steps (Page 4534).
Permit trace[1]	Only as of version V4.2 of the SIMOTION Kernel. **Active**: An additional program code is created, which enables monitoring of the program execution in program branches which are executed cyclically. **Inactive**: Trace is not possible. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). This function facilitates debugging of your program. See Tracking program execution using the trace (Page 4538).
Permit forward declarations	Forward declarations enable you to use program organization units (POUs) before they are completely defined. See the SIMOTION ST Programming and Operating Manual. This setting is always **active**. Forward declarations are always permitted for the MCC programming language.
Permit OPC-UA / -XML (load symbols to RT)	**Active:** Symbol information for the unit variables of the MCC source file is available in the SIMOTION device. This is required for: • The *_exportUnitDataSet* and *_importUnitDataSet* functions; see the SIMOTION Basic Functions Function Manual • The watch function of IT DIAG **Inactive:** Symbol information is not created.

Parameter	Description
Ignore global compiler settings (The program is not compiled again if the global settings are changed or extended.)	**Active:** The global settings of the compiler are ineffective for all parameters. The "Use global settings" checkboxes cannot be selected and are grayed out. When changing the global compiler settings, the MCC source is not recompiled. **Inactive:** The checkboxes "Use Global Settings" can be selected for all parameters and are presented against a white background. These checkboxes specify whether the global properties are taken over for the corresponding parameters. See the description under Effectiveness of global or local settings in the SIMOTION ST Programming and Operating Manual.
[1] Global setting is also possible (**Options > Settings > Compiler** menu), see Global compiler settings (Page 3996). See also the description on Effectiveness of global or local compiler settings in the SIMOTION ST Programming and Operating Manual.	

7.1.4.3 Inserting and managing MCC charts

MCC charts are displayed in the project navigator under the respective MCC source file.

Inserting a new MCC chart

You can insert a new MCC chart for an existing MCC unit in any of the following ways (see Insert new MCC unit (Page 3986)):

- In the project navigator: Below an MCC unit using the **Insert MCC chart** command
- Select the required MCC unit in the project navigator followed by **Paste > Program > MCC chart**
- Open the required MCC unit and select the **Insert MCC chart** symbol in the **MCC unit** toolbar

Procedure

To insert a new MCC chart into an existing MCC unit via the menu, proceed as follows:

1. Open the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Select the relevant MCC unit in the project navigator.
4. Select **Paste > Program > MCC chart**.
5. Enter the name of the MCC chart.
   The names for MCC charts must comply with the Rules for identifiers (Page 4047): They are made up of letters (A ... Z, a ... z), digits (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters. Protected or reserved identifiers are not permitted.
   The permissible length of the name is 25 characters.
   The names must be unique within the MCC unit. The names of all exportable program organization units (POUs) must also be unique within the SIMOTION device. The names of all MCC charts of the program source as well as the names of all exportable POUs of the device are displayed.

6. Optionally enter the identifier of an object-oriented namespace.
   Identifiers for namespaces must comply with the Rules for identifiers (Page 4047) (see above).
   When an identifier is entered, the MCC chart is assigned to the specified namespace.
   The compiler option (Page 3996) "Permit object-oriented programming" must be activated.
   For further information on object-oriented namespaces, see the appropriate section in the SIMOTION ST Programming and Operating Manual.
7. For creation type, select **Program**. For creation types *Function* and *Function block*, see Subprogram (Page 4117).
8. Activate the **Exportable** checkbox if you would also like the program to be available in other program sources (LAD/FBD, MCC or ST source files) or in the execution system (see Assigning programs to a task (Page 4514)).
9. You can also enter an author, version, and a comment.
10. Activate the **Open editor automatically** checkbox.
11. Confirm with **OK**.

Note

When you click **OK**, the MCC chart will only be transferred to the project. The data, together with the project, is only saved to the data carrier if you select, for example, **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

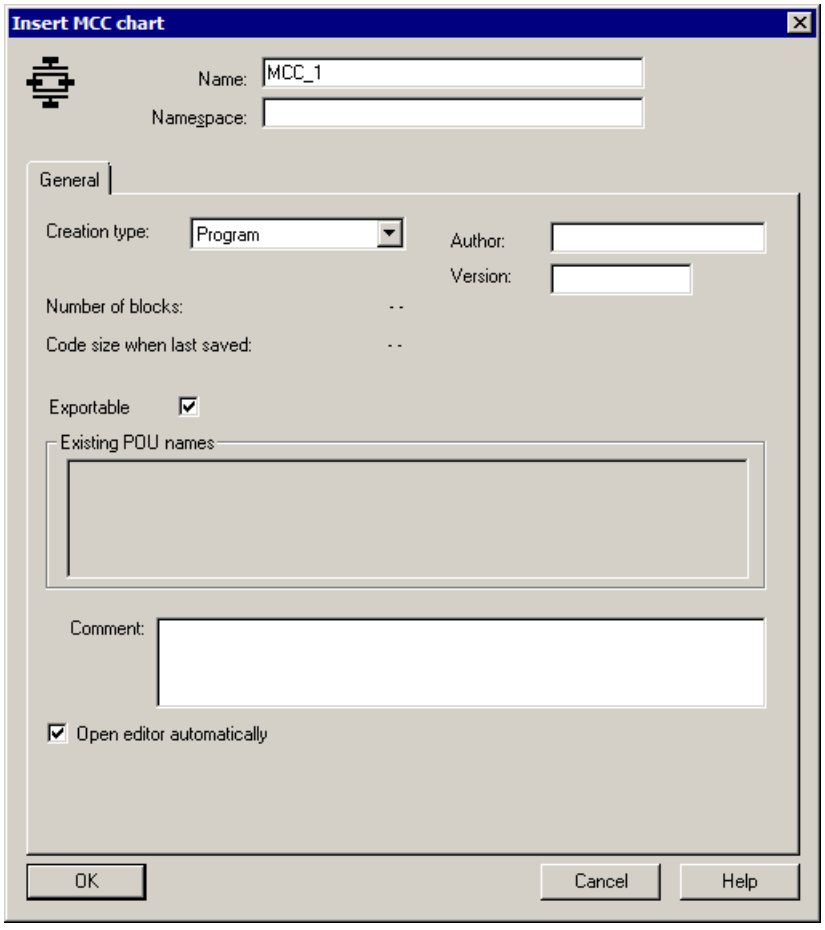


Figure 7-14 Dialog box for creating a new MCC chart

Note

Existing POE names are only displayed if the check box "Show existing POE names (insert dialog POE for LAD/FBD and MCC)" is set in the Settings > Workbench. If there are a large number of POEs in the project, it is recommended to deselect this option, as the computer memory for the process may not be sufficient to refresh the display.

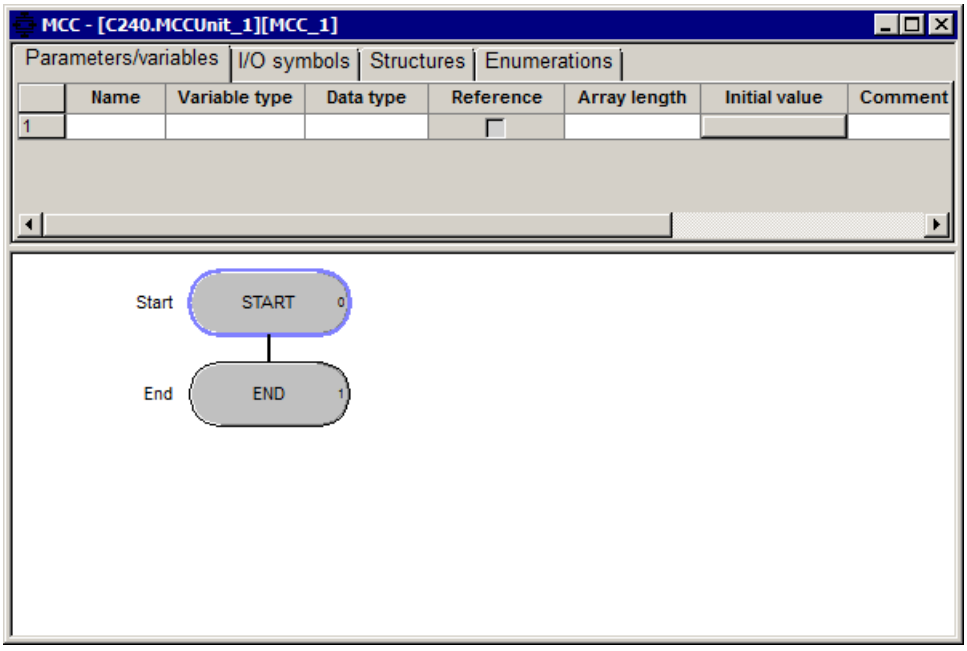


Figure 7-15 New MCC chart with declaration table and starting and end nodes

Opening an existing MCC chart

All MCC charts for an MCC unit are located in the project navigator below the relevant MCC unit.

Procedure

To open an MCC chart, do one of the following:

1. Open the subtree of the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Open the MCC unit containing the required MCC chart.
4. Select the required MCC chart.
5. Select the **Open** context menu.
6. Only for charts below MCC units with know-how protection (Page 3992):
   If the MCC unit (or a chart for this unit) is not already open and the login assigned to the MCC unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (until the unit and all its charts are closed).
   - If required, activate the "Use login as a standard login" checkbox.
     You will be logged in with this login and can now open additional units to which the same login is assigned without having to re-enter the password.

The MCC chart is opened in the working window. Multiple MCC charts can be opened.

Note

You can also double-click the required MCC chart to open it.

Transferring and compiling an MCC chart

You cannot compile an MCC chart independently. The MCC source file is always compiled with all of its associated MCC charts (see Transferring and compiling an MCC source file (Page 3989)).

Closing an open MCC chart

To close an open MCC chart in the working window, do one of the following:

- Click the x button in the header
- Select the **MCC chart > Close** menu command
- Select the **Window > Close All Windows** menu command

If the changes have not yet been saved in the project, you can save or cancel them, or abort the close operation.

Cutting, copying, and deleting an MCC chart

You can cut or copy an MCC chart and paste it under the same or a different MCC source file (as well as under a different SIMOTION device).

An MCC chart that has been deleted can no longer be pasted.

Note

When an MCC unit or MCC chart is copied under a SIMOTION device with an earlier version of SIMOTION Kernel, the reliability of commands and their parameterization is only checked when the MCC chart is opened or when the MCC unit or the project is compiled.

Procedure

Follow these steps:

1. Select the MCC chart in the project navigator.
2. In the context menu, select the appropriate item (**Cut**, **Copy**, or **Delete**).

Pasting a cut or copied MCC chart

Procedure

Proceed as follows to paste a cut or copied MCC chart:

1. Select an MCC source file in the project navigator.
2. In the context menu, select **Insert**.
   The MCC chart is inserted with a different name.
3. Change the name, if necessary (see Renaming MCC charts (Page 4009)).

Note

When an MCC unit or MCC chart is copied under a SIMOTION device with an earlier version of SIMOTION Kernel, the reliability of commands and their parameterization is only checked when the MCC chart is opened or when the MCC unit or the project is compiled.

Specifying the order of MCC charts in the MCC source file

As of Version V4.2 of SIMOTION SCOUT, the user no longer needs to respect the POE order in the MCC unit, as the order no longer has any significance in terms of the ability to perform compilation. The only reason to change the POE order is to ensure the POEs are arranged in a more logical way from the user's perspective.

Exception

When a project is saved in a format older than Version V4.2 of SIMOTION SCOUT, the MCC chart order in the MCC unit is of relevance as far as compilation is concerned. A subroutine (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the MCC chart of the subroutine appears above the chart in which it is used in the project navigator. If necessary, reorder the charts (see Subroutine call of function (FC) (Page 4129)).

Before saving a project in a format older than Version 4.2, you can check the project for downward compatibility (including the correct order of the POUs) via **Project > Old project format > Check the project for downward compatibility**.

Procedure

To change the order:

1. Select an MCC chart in the project navigator.
2. In the shortcut menu, select **Up/Down**.

Exporting and importing an MCC chart

The export and import functions offer you the option of saving an MCC chart outside the project on your hard disk so that you can copy it from there into another project.

If you have exported an MCC chart using an older software version, it can be imported and processed with any of the later software versions.

Exporting an MCC chart in XML format

You can use an XML export to save individual MCC units in a directory outside the project, independently of any particular version or platform.

Procedure

To export an MCC chart in XML format, proceed as follows:

1. Select the MCC chart in the project navigator.
2. Select the **Export as XML** command in the context menu.
3. Only for MCC charts in MCC units with know-how protection (Page 3992) and which are not already open:
   If the associated MCC unit (or a program of this unit) is not already open and the login assigned to the MCC unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this chart is temporarily canceled (for this export).
   - If required, activate the **Use login as a standard login** checkbox.
     You will be logged in with this login and can now export or open additional units to which the same login is assigned without having to re-enter the password.
4. Enter the path and file name for the XML export and click **Save** to confirm.

The MCC chart is saved as XML format; the file name is given the default extension *.xml

Note

An MCC chart with know-how protection is exported without protection.

Importing an MCC chart from XML data

This operation allows you to import an MCC chart that was exported as a decoded file in XML format.

Procedure

1. Select the MCC unit in the project navigator.
2. In the context menu, select the **Import MCC chart > From MCC format** command.
3. Select the XML data to be imported.

Note

Individual objects (e.g. MCC charts) that were exported as XML data cannot be imported selectively.

Importing an MCC chart

This operation imports an MCC chart that was exported as an MCC chart in an earlier version of SIMOTION SCOUT into an existing MCC unit.

Note

An MCC chart cannot be exported into the corresponding format in the current version of SIMOTION SCOUT.

Procedure

1. Select the relevant MCC source file in the project navigator.
2. In the context menu, select **Import MCC Chart > From MCC Format**.
3. Select the directory and file name (name.mcc).
   The dialog window for inserting an MCC chart is displayed.
4. Enter the name of the MCC chart in the project.

If the existing technology objects do not match the MCC chart to be imported, it is possible to create them later.

Properties of an MCC chart

Procedure

The properties of an MCC chart are already defined when you insert it. However, these properties can be viewed and modified by doing the following:

1. Select the MCC chart in the project navigator.
2. In the shortcut menu, select **Properties**.
   The **MCC chart properties** dialog box opens.

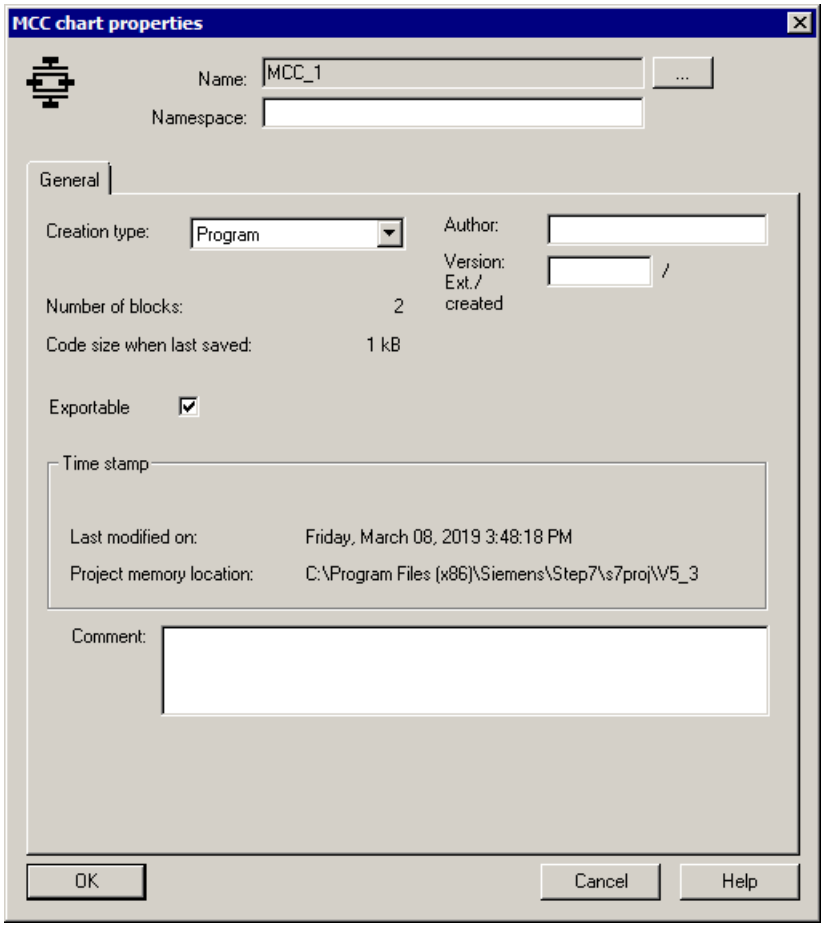


Figure 7-16 Properties of an MCC chart

Renaming MCC charts

To change the name of an MCC chart, proceed as follows:

1. Open the Properties window of the MCC chart.
2. Click ... .
3. Enter the new name for the MCC chart (see Rules for identifiers (Page 4047)).
4. Click **OK** to confirm.

Assigning an MCC chart to an object-oriented namespace

Proceed as follows to assign an MCC chart to an object-oriented namespace:

1. Open the Properties window of the MCC chart.
2. Enter the identifier of an object-oriented namespace in the **Namespace** field. Identifiers for namespaces must comply with the Rules for identifiers (Page 4047).
3. Confirm with **OK**.

When an identifier is entered, the MCC chart is assigned to the specified namespace. The compiler option (Page 3996) "Permit object-oriented programming" must be activated.

For further information on object-oriented namespaces, see the appropriate section in the SIMOTION ST Programming and Operating Manual.

Changing the creation type of the MCC chart

To change the creation type of an MCC chart, proceed as follows:

1. Open the Properties window of the MCC chart.
2. Select the new creation type (**Program**, **Function** or **Function block**).
3. Click **OK** to confirm.

Changing usability in other program sources (export capability)

To change the export capability of an MCC chart, proceed as follows:

1. Open the Properties window of the MCC chart.
2. If required, activate the **Exportable** checkbox.
   With activated checkbox, the MCC chart can be used in other program sources (e.g. MCC units, ST source files). A program can be assigned to a task only when this checkbox is activated (see assigning programs to a task (Page 4514)).
   When the checkbox is deactivated, the MCC chart can only be used in the associated MCC unit.
3. Click **OK** to confirm.

MCC chart overview

As the width and height of MCC charts are capable of exceeding the size of the editor window, you can create a separate overview window for every MCC chart. You can move this overview window outside of the Workbench, e.g. to a second monitor. You can gain an aerial view of the MCC chart by adjusting the size of the window and the zoom factor.

Call up overview window

Requirement: The MCC chart must be opened in the editor and be in the foreground of the working range.

Proceed as follows to open the overview window (alternatives):

- Select the **MCC chart > Overview** menu command.
- Select **Overview** from the shortcut menu.
- Press the CTRL+SHIFT+O shortcut key.

The Overview window opens; it is specific to the relevant MCC chart.

Using the Overview window

You can move the window on the screen as you wish and move it outside of the Workbench e.g. to a second screen. You can maximize or minimize the window as you wish by dragging the edges of the window.

The window is merely an overview. It cannot be used for editing the MCC chart, e.g. for selecting or editing commands. The MCC command currently selected in the editor is marked in blue.

You can change the representation size of the chart gradually using the  and  buttons.

If the "Editor frame" check box is enabled the area of the MCC chart visible in the editor is marked by a black frame.

Process for modules, see Module formation (Page 4019):

- When opening the module (in the Editor shortcut menu **Edit modules > Zoom into module**): A new "Levels" tab is opened in the overview window.
- When closing the module (in the Editor shortcut menu **Edit modules > Zoom out of module**): The last tab opened in the overview window is closed.

The overview window is also closed when the MCC chart is closed.

7.1.5 Programming in MCC

This chapter contains detailed instructions on how to program in the MCC language.

7.1.5.1 Principles of programming

General principles of programming

Every new MCC chart already contains a starting and an end node. You program commands and instructions between these two nodes. The commands you program are executed from the starting node in the direction of the end node.

Motion commands

Motion commands are executed based the transition behavior and step enabling conditions that are selected in the parameter screen forms.

Motion commands can be issued from all tasks.

The overall status of an axis motion is recorded in the system and can be queried at any time using system variables. A command can be waiting, active, or inactive.

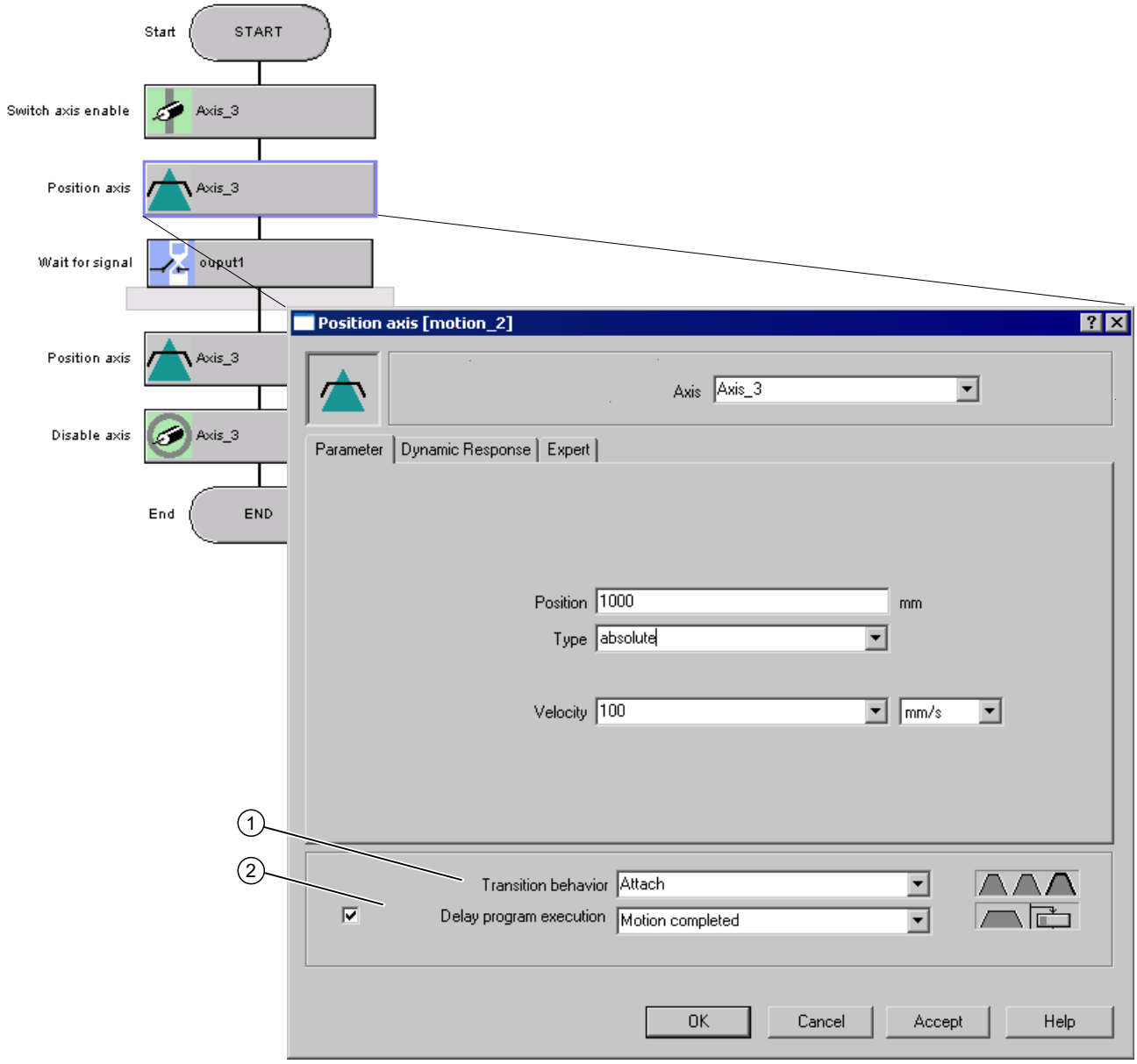
wait commands

Execution of the task is delayed until the programmed event (that is, the condition) occurs. The program is not executed again cyclically, and the task does not require any processor time. All other tasks continue to run unaffected.

The events are scanned in the interpolator cycle clock (IPO cycle clock). If the appropriate event occurs, the highest-priority task in its own level is continued. In the case of MotionTasks, the subsequent motion command is executed in the following IPO cycle clock.

Simplified example of programming principles

Axis_3 will move to absolute position 1,000 mm at a velocity of 100 mm/s. As soon as output 1 is switched, axis_3 will move back to position 0 mm.



① Attach transition behavior:
After switching the axis enable, the program switches to the positioning command.

② Delay program execution:
The next command is started when the axis motion has ended.

Figure 7-17 Example of programming principles

7.1.5.2 Managing MCC commands

inserting commands

You can insert commands into the MCC chart using one of the following:

- The context menu or
- MCC editor toolbar
- **MCC chart > Paste** menu command

Context menu

The context menu of an MCC chart opened in the working area contains the complete range of commands. The commands are divided into the same command groups as for the MCC editor function bar, but without the **Important commands** command group.

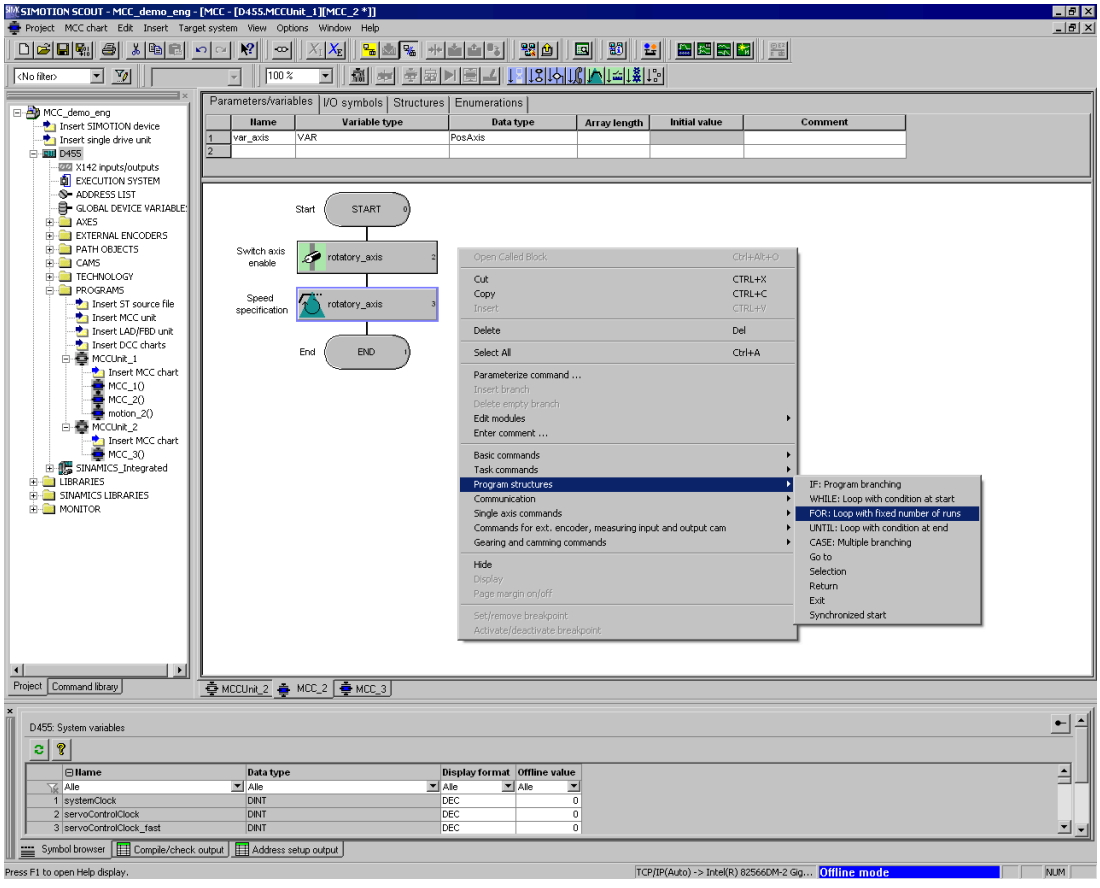


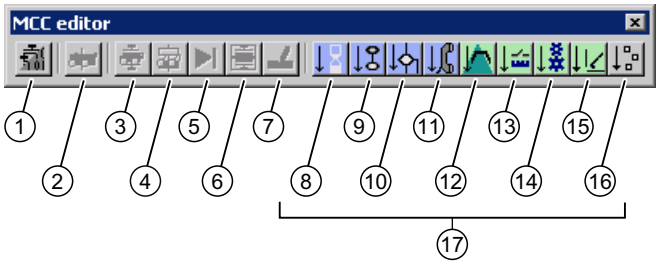
Figure 7-18 MCC editor context menu

Note

The selection of command groups or command lists available in the context menu depends on which technology packages are currently selected for the SIMOTION device. To select the required technology packages, highlight the relevant SIMOTION device in the project navigator and choose the **Select technology packages** command in the context menu.

MCC editor toolbar

This toolbar contains important programming command buttons (for example, Insert MCC chart, Accept and compile, and Monitor) together with the complete range of commands. The commands are subdivided into command groups.



①	Accept and compile	⑩	Program structures
②	Program status	⑪	Communication
③	Monitoring	⑫	Single axis commands
④	Single-step	⑬	Commands for external encoders, measuring inputs and output cams
⑤	Next step	⑭	Commands for synchronous operation and camming
⑥	Trace on/off	⑮	Commands for path interpolation
⑦	Rerecord trace	⑯	Important commands
⑧	Basic commands	⑰	Command groups
⑨	Task commands		

Figure 7-19 MCC editor toolbar

Each of these groups is represented by a button on the toolbar. The list of commands is displayed when you move the cursor over the button; you can insert the appropriate command by clicking it with the left mouse button.

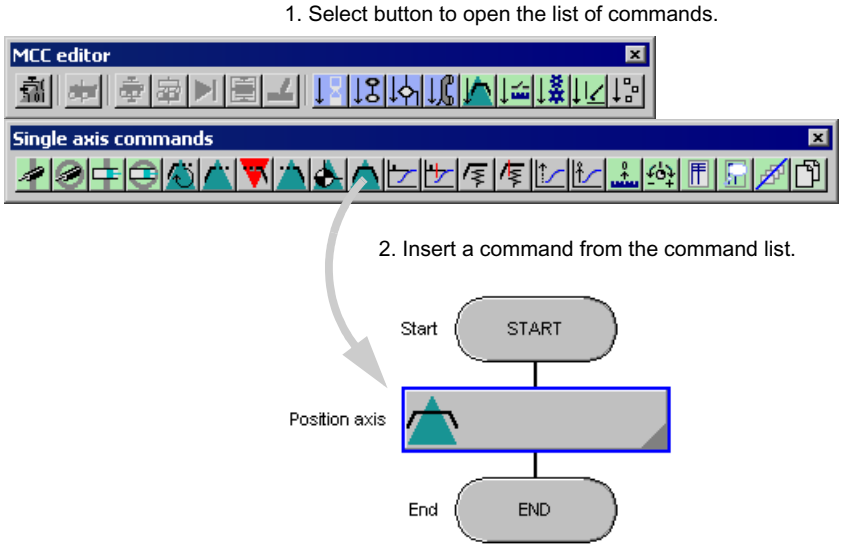


Figure 7-20 Inserting a command from the MCC editor toolbar

The command selected by clicking is always inserted after the selected command.

You can detach the displayed command list using the cursor and place it anywhere on the screen. It will then be easily accessible while you are creating your program.

If all of the command lists have been detached, every command can be inserted in the MCC chart with one click. The figure below shows a working window with an MCC chart and opened command lists.

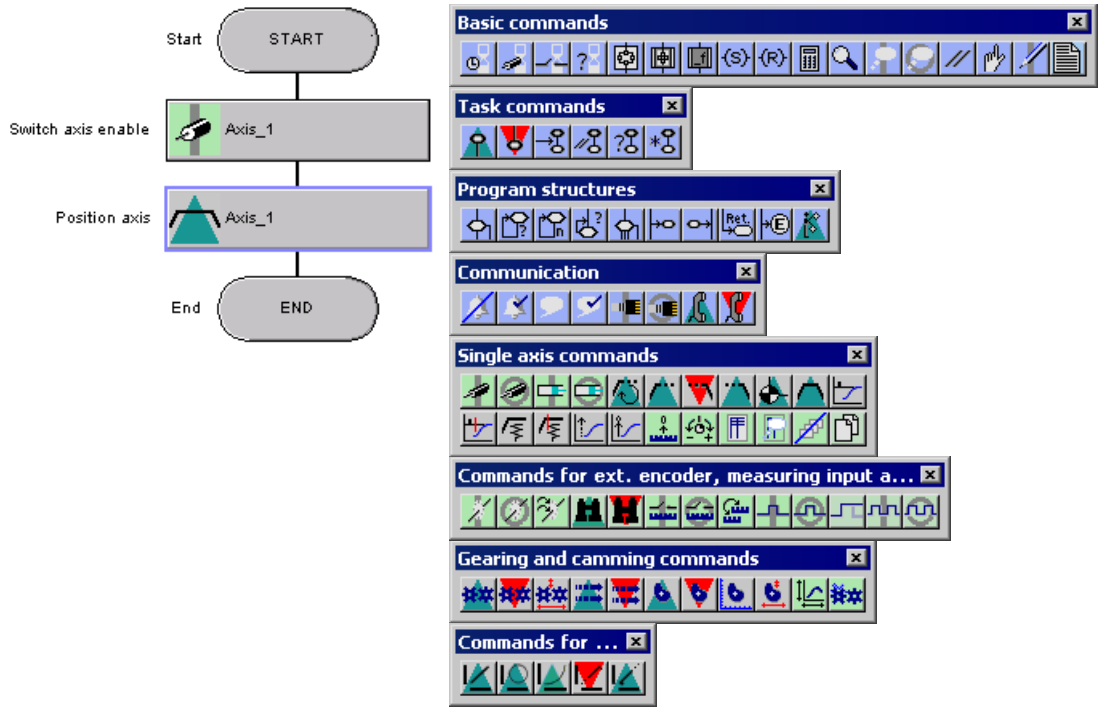


Figure 7-21 Opened command lists in the Workbench making every command available with one click

Note

The selection of command groups or command lists available for the MCC editor toolbar depends on which technology packages are currently selected for the SIMOTION device. To select the required technology packages, highlight the relevant SIMOTION device in the project navigator and choose the **Select technology packages** command in the context menu.

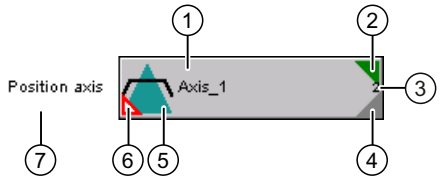
For example, the **Commands for path interpolation** list is only available in the PATH and CAM_EXT technology packages as of Version 4.1.

Representation of commands in the MCC chart

Commands are represented as a rectangular block. The starting and end nodes are oval, the conditions are diamond-shaped. All commands are assigned a symbol that graphically represents the command function. The commands are also color-coded:

Color	Commands
Light blue background	Basic commands, task commands. Communication
Light green background	Commands for technology objects that are not motion commands
Green	Motion commands for starting a movement
Red	Motion commands for stopping a movement

The figure below shows how most of the commands are represented in an MCC chart. The symbolic representation of commands makes it immediately obvious that axis_1 is being positioned. You can modify the brief comment if desired.



① Name of the technology object, e.g., axis name

② Marking:
Command comment present

③ Serial numbers for the command

④ Marking:
Command has not been parameterized yet or has been assigned incorrect parameters

⑤ Positioning the symbol

⑥ Hold point set

⑦ Brief comment

Figure 7-22 Representation of a motion command in the MCC chart

Brief comment

The brief comment is used to document the MCC chart or the command. It is preset with the name of the relevant command. Clicking the text allows you to change it, if necessary.

There is no maximum text length. The text length that can be displayed on the screen is dependent on the following factors:

- Font size and type
- Screen resolution

Command comment

You can enter an additional comment for each command:

- Select the required command in the MCC chart.
- In the context menu, select **Enter comment**.
- Enter the text of the comment in the displayed window.
- Click **OK** to confirm.

If a command comment is present, a green indicator appears at the top right of the command.

If you place the cursor above the command in the MCC chart, the comment is displayed via a tool tip.

Comment block

In addition, there is also a separate command for entering comments (see Comment block (Page 4197)).

Translating comments

Brief comments and command comments can be translated into various languages and displayed.

To do so, use the **Project > Language-Dependent Texts** menu command. Select **MCC comments** as the text source.

The remaining steps are outlined in the online help.

Numbering of commands

When a command is inserted, it is automatically assigned a serial number. This number is unique and is used to identify the command, for example, in the cross-reference list.

Note

The user is not able to change this number.

A number that is freed up after deletion of a command is reassigned only after the MCC chart has been saved and closed and then reopened.

Selecting commands

If you wish to copy a command, for example, you must select it beforehand.

You select a command by clicking it. You can select several commands by moving the cursor over them with the left mouse button depressed. If you wish to select all of the commands, choose **Select all** in the context menu.

Selected commands are identified by a thick blue border; in addition, the border of the last selected command flashes.

Hiding and displaying commands

You can hide commands for test purposes; hidden commands are excluded from the code generation process and are unavailable for selection.

Follow these steps:

1. Select the commands you want to hide.
2. On the shortcut menu, select **Mask Out**.

If you hide a control structure, all of the commands it contains and their parameters are also hidden.

To show hidden commands again, proceed as follows:

1. Select the commands you want to show.
2. On the shortcut menu, select **Display**.

Copying, deleting, cutting, or pasting commands

The functions **Copy**, **Delete** etc. can be selected either from the context menu or the **Edit** menu.

If you copy or cut a command and then paste it, the programmed parameters are also copied or cut and pasted. You can also paste the copied command into other MCC charts.

You cannot paste a command that has been deleted.

Note

If you delete a command by mistake, you can undo the delete operation. Select the **Edit > Undo** menu command.

If you copy a control structure, all of the commands it contains and their parameters are also copied. The same applies when **Delete**, **Cut**, or **Paste** is used.

Undo/redo

All actions can be undone in reverse order. Select the **Edit > Undo** menu command or the **Undo** button.

Note

The following actions cannot be undone:

- Save
- Accept and compile

If you want to redo one or more undone actions, use the **Edit > Redo** menu command or the **Redo** button.

Module creation

Modules are used to structure the MCC chart. Modules comprise a number of interrelated commands that are joined to make one single command. This feature allows even complex MCC charts to be structured in a clear, easily readable fashion.

A module can contain all types of commands. You can also nest modules and copy them for multiple use in different charts.

Command sequences that are to be used in several MCC charts must be swapped out as subprograms (Page 4117).

Procedure for creating a module

You can use two different methods to create a module:

- You can insert an empty module using the appropriate MCC editor toolbar command. You can then open the module by double-clicking it or using the context menu, and program the commands.
- You can use the context menu command directly to combine existing commands in the MCC chart to form a module:
  Select the required commands (**Shift** + press and hold the left mouse button and move the cursor over the MCC commands to be selected) and create the module via the context menu **Edit modules > Create module**.

Opening and closing a module

The module opens when you double-click the module command or select **Edit modules >Zoom in to module** in the context menu.

Close the module again by selecting **Edit modules > Zoom out of module** in the context menu.

Change the name of the module

Change the name of a created module by selecting **Edit module > Set name of module** in the context menu.

Canceling a module

You can cancel a created module by selecting **Edit modules > Cancel module** in the context menu. The commands previously grouped in the module are then displayed in the MCC chart instead of the module command.

Changing the module structure

If you want to change the module structure, cancel existing modules and create new ones.

Note

In offline mode, the changes to the module structure (creating or canceling modules) affect the project. If taken into the project, the MCC unit must be compiled again.

In online mode, a change to the module structure only affects the display, and the MCC unit does not have to be compiled again. Only when online mode is exited, are the changes to the module structure taken into the project and a new compilation is required.

7.1.5.3 Processing MCC commands

Assigning command parameters

All commands are assigned parameters in parameter screen forms. The structure of the parameter screen forms varies depending on the command.

Opening the parameter screen form

To open a parameter screen form, double-click the command or select **Parameterize command** in the context menu. You can open and edit any number of parameter screen forms from the same or different charts.

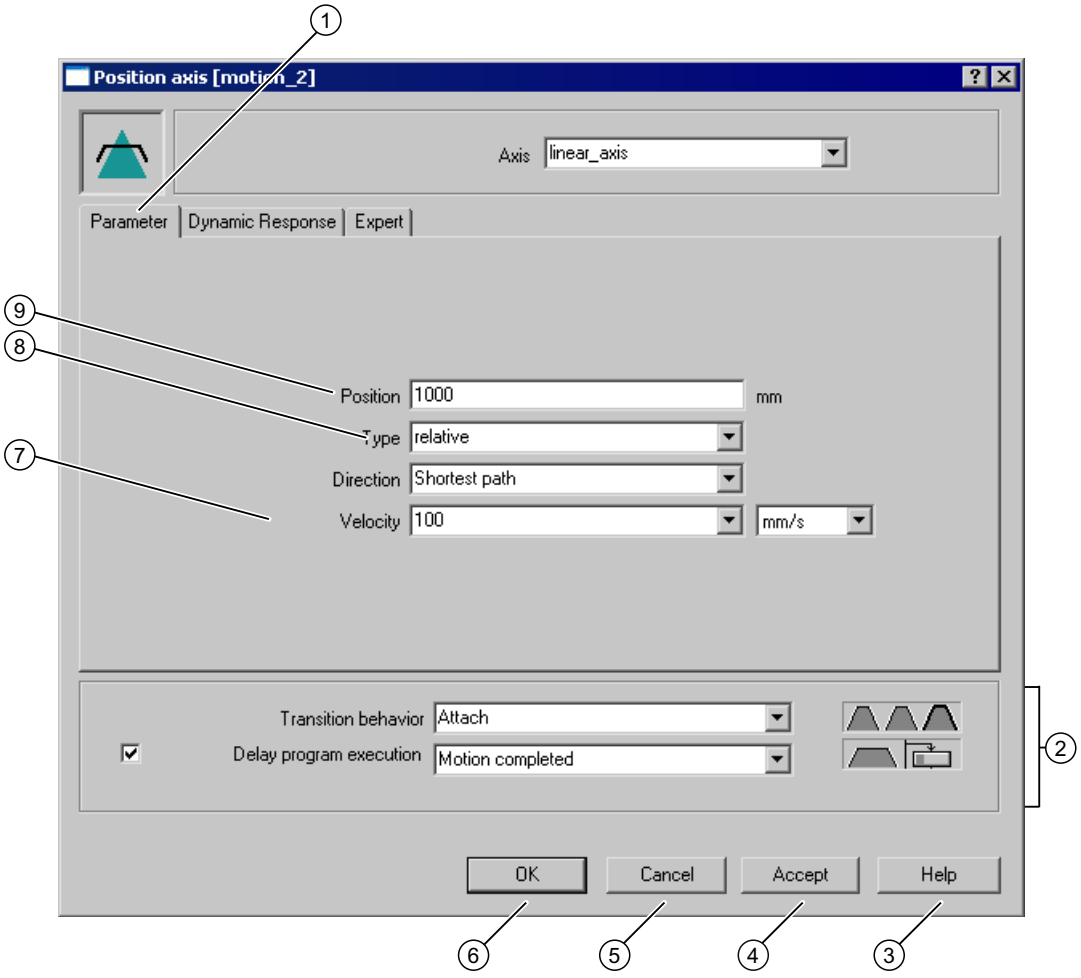
Changes in the symbol browser and in the project structure and commissioning are possible while a parameter screen form is open.

Structure of parameter screen forms for motion commands

The axis that is to be moved or stopped must always be specified in the top half of the box.

The center section contains different input fields and selection lists relating to the axis motion or stop commands. For reasons of clarity, this section contains several tabs. You need only assign the parameters in the first tab to obtain an executable command. All of the other tabs are optional and designed for special settings (such as jerk, acceleration, velocity profile, etc.).

The lower half of the box contains the settings for the transition behavior of the previous command and the step enabling condition for the next command.



①	Tab	⑥	Apply the parameter assignment and close the screen form
②	Transition behavior and step enabling condition	⑦	Editable selection list
③	Online help	⑧	Selection list (combo box)
④	Apply the parameter assignment	⑨	Input field
⑤	Close the screen form (discard the parameter assignment)		

Figure 7-23 Parameter screen form for a motion command

If there are many commands, the layout of the parameter screen form (presentation and selection options) depends on:

- Configuration data and system variables of the selected technology object, see Expert tab (Page 4034).
- Entries you have already made in the parameter screen form. Read the note about the table in Selection list (combo box) (Page 4022).

Structure of parameter screen forms for conditions and branches

You can program conditions and branches in the MCC editor using the LAD, FBD, or Formula languages, see LAD/FBD/formula (Page 4152).

Input fields and selection lists

Input field

Each input field must contain one of the following entries:

- Value
  In many cases, the entered value is checked for a maximum value that was specified during configuration of the technology object. This value is displayed in the tool tip.
- Unit variables (of the MCC source file) and global device variables or I/O variables
  These can be moved from the symbol browser to the input field with a drag-and-drop operation (see Formula (Page 4158)).
- Local variables of the MCC chart
- Formula
  You can drag-and-drop system functions and operators from the command library (see Using the command library (Page 4161)). When a system function is inserted, an automatic reliability check is performed within the relevant context (e.g. data type suitability).

Selection list (combo box)

Combo boxes provide different selection options. Frequently occurring selection options are listed in the following table below.

Table 7-3 Common selection options in combo boxes

Selectable options	Meaning
Default	The default value in the associated system variable is used. Default values can be defined during configuration of the technology object (see SIMOTION Basic Functions Function Manual). The "Set Axis Parameter" (Page 4325) command is used to change frequently used default Axis values.
Last programmed	The last programmed value is entered.
Last programmed velocity	For velocities only: The last programmed velocity is applied. For superimposed motion, the velocity of the base motion and the superimposition are added up.
Current	For velocities only: The active axis velocity is applied. Use for on-the-fly transitions and substitute motions.

Note

Not all of the options listed here are available in every case.

Note the following regarding the "default" and "last programmed" selection options: If the display of additional fields in the parameter screen form depends on this entry, these fields may not be displayed or some selection options may not be available.

Extension of the selection options by variables

You can extend the specified selection options in a selection list by defining variables of the appropriate data type in the declaration table of the MCC unit or the MCC chart. The names of the variables are also offered in the selection list.

- In selection lists for technology objects:
  Define a variable of the data type of the corresponding technology object (see Using a variable of the technology object data type (Page 4024)). The data type of the technology object (Page 4060) to be used is displayed in the tooltip of the selection list.
- In selection lists with enumeration elements:
  Define a variable of the corresponding enumeration data type. The enumeration data type to be used is displayed in the tool tip of the selection list.
  The variable can be assigned an element of the enumeration data type.
  A list of the elements of the respective enumeration data type is contained in the description of the relevant system function for the respective MCC command, see, for example Technology Packages, System Functions List Manual. The relevant system function is also specified for every MCC command in the Description of the MCC commands (Page 4173).

You can also declare the variables "on-the-fly". Enter the identifier of the variables in the combo box and press the **Return** key. For this, see Defining variables in the Variable declaration dialog box ("on-the-fly" variable declaration) (Page 4067).

Editable selection list (editable combo box)

Editable selection lists combine the properties of a Selection list (combo box) (Page 4022) and an Input field (Page 4022). They are used to input the numerical values of physical quantities (e.g. position, velocity). They are always linked to a unit of measure.

The input field of the editable selection list is associated with a numeric data type (usually LREAL). This is displayed in the tooltip.

- Select an entry from the selection options offered.
  You can select from the following:
  - For various enumeration elements, see Selection list (combo box) (Page 4022).
    These enumeration elements cannot be extended by variables.
  - All variables that are of the numeric data type of the input field and are effective in the MCC chart.
    Here, you select the variable that contains the numerical value of the physical quantity.
    You can define additional variables in the declaration tables or "on-the-fly" (Page 4067).

or

- Enter the value directly in the same way as in an entry field (Page 4022) (as a value or formula). To do this, select the entry in the selection list and overwrite it.

Note

When you select "**%**" in the "**Unit**" selection field: note that the value entered represents a percentage of the associated default value.

Unit

Here you can select what the value in the adjacent editable selection list represents. The following are available:

- The unit of the physical quantity defined during configuration of the technology object
  The value in the adjacent field represents a physical quantity with the indicated unit.
- "%" (percentage)
  The value in the adjacent field represents a percentage of the associated default value.

Assigning a technology object or a technology object-type variable to a command

Using a technology object

One or more technology objects (TOs) can be assigned to the command via its parameter screen form. Either a technology object which is already configured in the project navigation or a technology object-type variable can be assigned.

Assigning a technology object

A technology object which has already been configured (e.g. a linear axis with an incremental encoder) can be assigned to a command via its parameter screen form. Specific configuration data from the technology object is used to influence the design and/or selection options in the parameter screen form.

This relevant configuration data is displayed on the "Expert" tab in the "TO properties" list and saved in the command.

By assigning a technology object, the values displayed in the "TO properties" list on the "Expert" tab are fixed entries in the parameter screen form and cannot be changed.

Using a technology object-type variable

Assigning a technology object-type variable

Defining a technology object-type variable only serves to create a reference to a technology object.

As a place holder for a technology object, the variable can be assigned to a command via its parameter screen form. A technology object is only assigned to variables during the runtime of a program.

The variable defaults which influence the design and/or selection options in the parameter screen form are displayed in the "TO properties" list in the "Expert" (Page 4034) tab.

Through the assignment of a variable, the TO properties values shown in the "TO properties" list in the "Expert" tab are not fixed entries in the parameter screen form and can be changed in the parameter screen form.

However, these TO properties only appear in the parameter screen form and do not exist as the configuration data/system variables of a technology object. The command only receives the configuration data/system variables of a technology object if the technology object is assigned to the variable during the program runtime.

Example of an absolute encoder

A "posAxis" (position axis)-type variable is defined and assigned to the MCC "Home axis" command via its parameter screen form. An axis which is configured on the SIMOTION device, with an absolute encoder, is assigned to the variable as a technology object during the program runtime. Therefore, the parameter screen form must be adapted accordingly in the "Expert" (Page 4034) tab in order to select the "Absolute encoder adjustment" or "Absolute encoder calibration with specification of the position value" homing types.

Procedure

1. Define a technology object-type variable in the MCC chart declaration table by specifying the "posAxis" data type (see Description of the technology objects data types (Page 4060)). This means that a position axis-type technology object can be assigned to the variable during the program runtime.

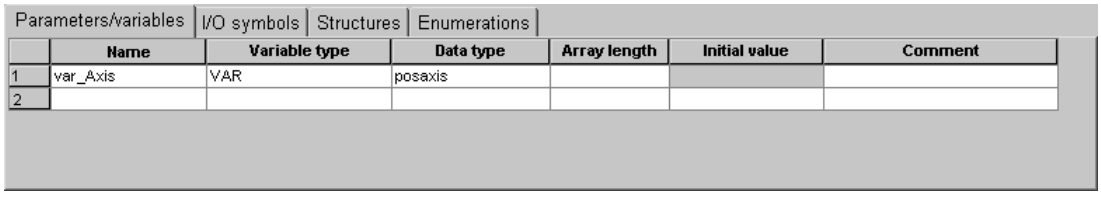


Figure 7-24 Defining a technology object-type variable in the declaration table

2. Insert the "Home axis" command into the MCC chart.
3. Double-click on the command.
   The parameter screen form opens.
4. Select the defined variable in the "Axis" dialog box.
   The variable is thus assigned to the command as a place holder for a position axis. Since a variable has been assigned to the command, and not a technology object, the TO properties, e.g. the encoder type, can be changed on the "Expert" tab.

5. Select the "Expert" tab.
   The variable defaults that influence the parameter screen form (design and/or selection options) are displayed in the "TO properties" list. For example, the "Incremental" default value appears in the "Value" column for the encoder type, i.e. an incremental encoder is preset. Based on the default "Incremental", only the homing types for an incremental encoder are available for selection in the "Homing type" dialog box in the "Parameters" tab. In order to select the homing types for an absolute encoder, the encoder type must be changed to absolute encoder beforehand.

6. In the "TO properties" list, select the entry "Absolute" or "Absolute cyclic" in the encoder type row from the drop-down list in the "Value" column.
   The encoder type is changed from incremental encoder to absolute encoder.

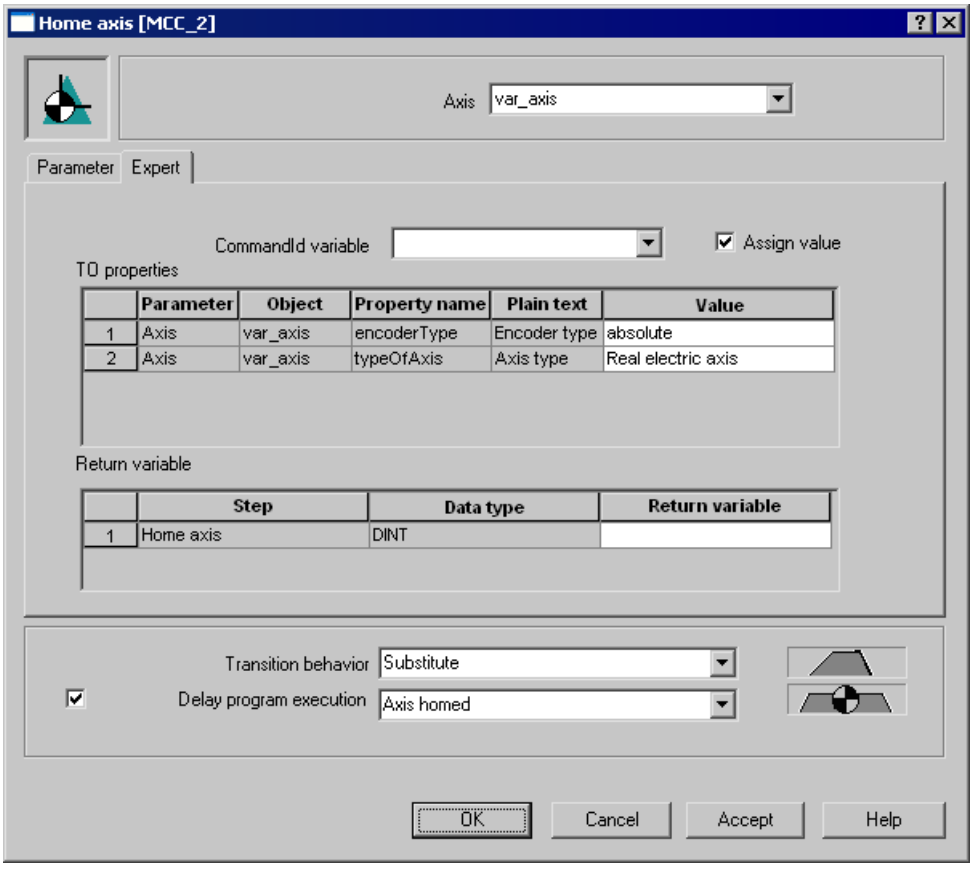


Figure 7-25 Selecting an absolute encoder via the Expert tab

7. Select the "Parameters" tab.
   The only options available for selection now in the "Homing type" dialog box are the "Absolute encoder adjustment" or "Absolute encoder calibration with specification of the position value" homing types.

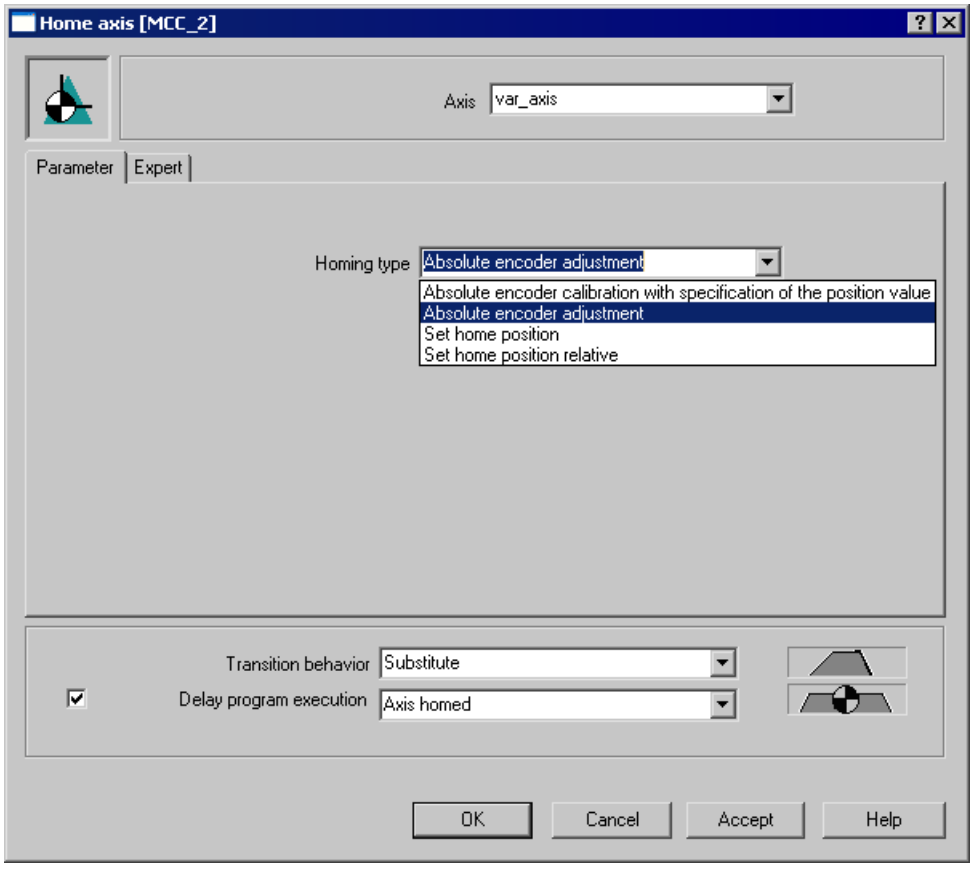


Figure 7-26 "Absolute encoder adjustment" homing type

Example of time-based cams

An "outputCamType" (output cam)-type variable is defined and assigned to the MCC "Switch output cam on" command via its parameter screen form. A time-based cam on a rotary axis which is configured on the SIMOTION device is assigned to the variable as a technology object during the program runtime. Therefore, the parameter screen form has to be adapted accordingly in the "Expert" (Page 4034) tab.

Procedure

1. Define a technology object-type variable in the MCC chart declaration table by specifying the "outputCamType" data type (see Description of the technology objects data types (Page 4060)). This means that an "output cam"-type technology object can be assigned to the variable during the program runtime.

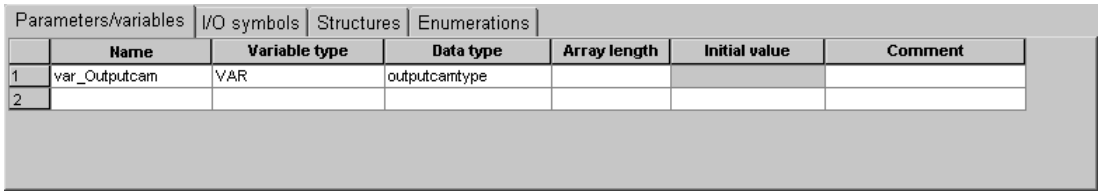


Figure 7-27 Defining a technology object-type variable in the declaration table

2. Insert the "Switch output cam on" command into the MCC chart.
3. Double-click on the command.
   The parameter screen form opens.
4. Select the entry "<Reference>" in the "Axis/encoder" dialog box, as the output cam is predefined as a reference (variable), and not as a technology object.
   Thus, the variable is assigned to the command as a place holder for an output cam in the "Output cam" dialog box. Since a variable has been assigned to the command, and not a technology object, the TO properties, e.g. the output cam type, can be changed on the "Expert" tab.

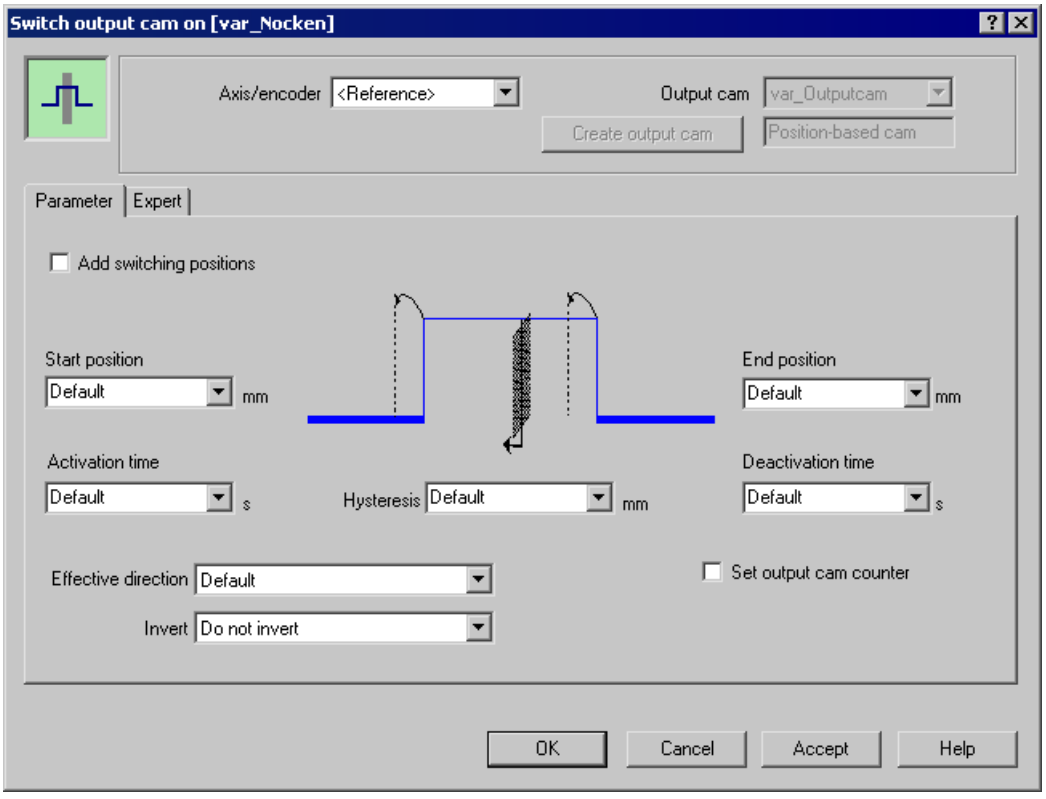


Figure 7-28 Parameter screen form: Switch output cam on for linear axis and position-based cam

5. Select the "Expert" tab.
   The variable defaults that influence the parameter screen form (design and/or selection options) are displayed in the "TO properties" list. For example, the "Position-based cam" default value appears in the "Value" column for the output cam type, i.e. a position-based cam is preset.
6. In the "TO properties" list, select the entry "Time-based cam" in the output cam type row from the drop-down list in the "Value" column.
   The output cam type is changed from position-based cam to time-based cam.
7. In the "TO properties" list, select the entry "Rotary axis" in the axis type row from the drop-down list in the "Value" column.
   The axis type is changed from linear axis to rotary axis.
8. Select the "Parameters" tab.
   The information displayed here has changed markedly because the parameters are now shown for a rotary axis and for a time-based cam.

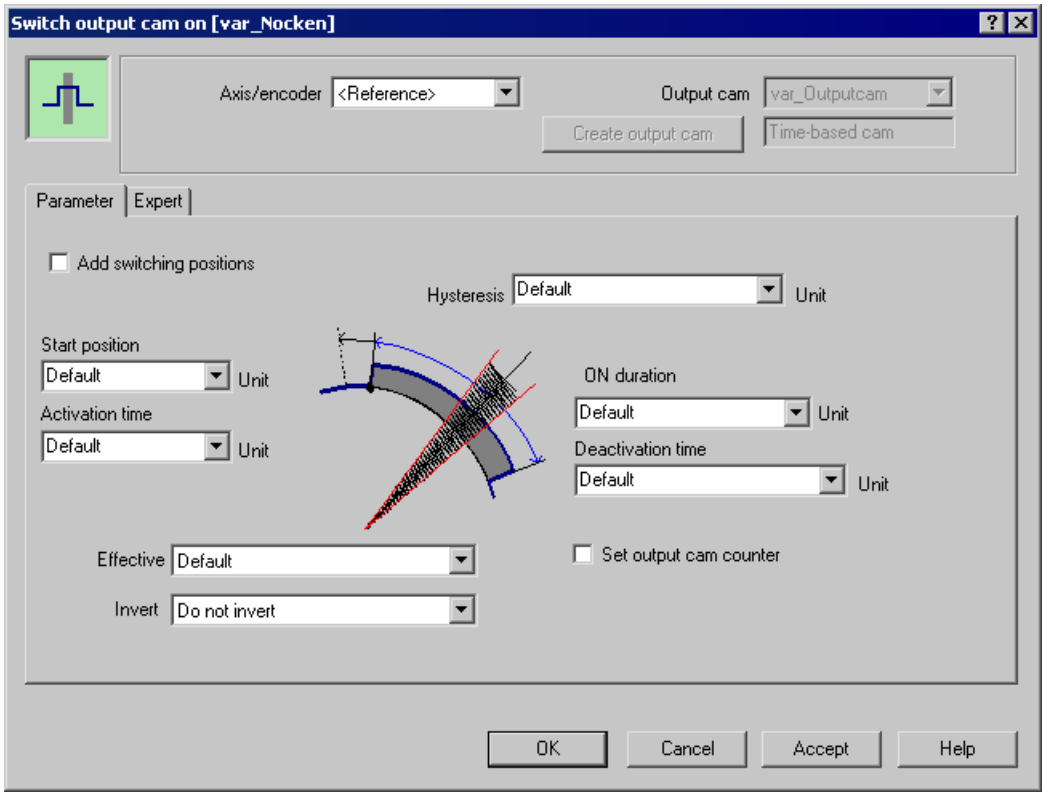


Figure 7-29 Parameter screen form: Switch output cam on for rotary axis and time-based cam

Dynamics tab

The parameter dialog boxes for most of the motion commands include a **Dynamic response** tab. Here you can specify the velocity profile type and the associated values for acceleration, deceleration, and jerk.

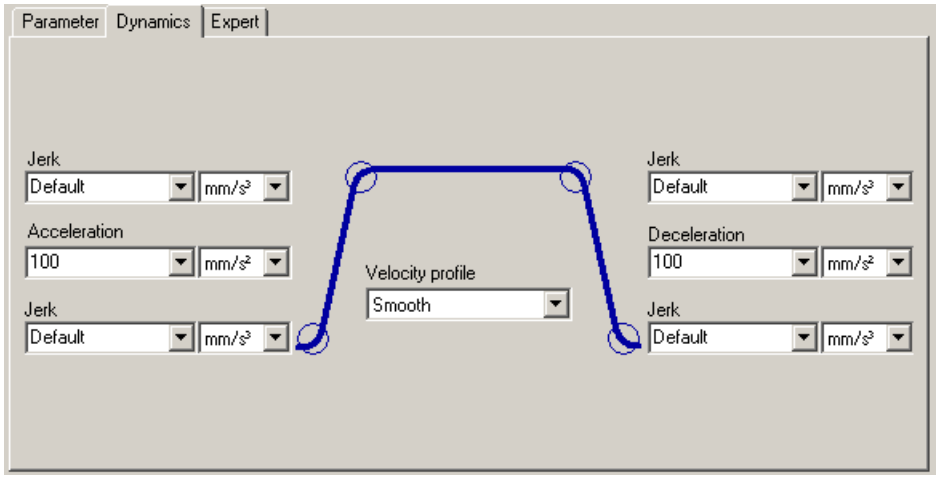


Figure 7-30 Dynamic response tab for motion commands

Overview of parameters

Table 7-4 Overview of parameters for Dynamic response tab

Field/Button	Explanation/Instructions
Velocity profile	The velocity profile is used to specify the transitions between the individual motion phases. The velocity profile influences the motion transitions as follows: • At the start and end of the acceleration phase, the transition to the constant acceleration or constant velocity phase • At the start and end of the deceleration phase, the transition to the constant deceleration or constant velocity phase Select the velocity profile. **Trapezoidal** A trapezoidal velocity profile is in effect for the command. Only the acceleration and deceleration can be programmed. **Constant** The velocity profile with smooth acceleration characteristic is in effect for the command. The change in acceleration/deceleration (jerk) can be verified. With the "Gearing on" (Page 4385) command, a constant velocity profile only takes effect if configuration data syncingMotion.smoothAbsoluteSynchronization = YES is set. **Default** **Last programmed** See Selection list (combo box) (Page 4022). The system variable for the default is given together with the description of each command.
Acceleration	The acceleration value during the constant acceleration phase. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed** See Selection list (combo box) (Page 4022). The meaning depends on the acceleration model used, which can be selected via the configuration data TypeOfAxis.DecodingConfig.directionDynamic: • **Non-direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = NO (default) The value causes acceleration in the motion of the axis, independent of the direction of motion. • **Direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = YES The value causes acceleration when the direction of rotation is positive and deceleration when the direction of rotation is negative. For additional information on specifying the acceleration and deceleration (acceleration model), see the TO Axis Electric/Hydraulic Function Manual. The system variable for the default is given together with the description of each command.

Field/Button	Explanation/Instructions
Deceleration	Deceleration value during the constant deceleration phase Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed** See Selection list (combo box) (Page 4022). The meaning depends on the acceleration model used, which can be selected via the configuration data TypeOfAxis.DecodingConfig.directionDynamic: • **Non-direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = NO (default) The value causes deceleration in the motion of the axis, independent of the direction of motion. • **Direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = YES The value causes deceleration when the direction of rotation is positive and acceleration when the direction of rotation is negative. For additional information on specifying the acceleration and deceleration (acceleration model), see the TO Axis Electric/Hydraulic Function Manual. The system variable for the default is given together with the description of each command.
Jerk	Only active with constant velocity profile. Change in acceleration or deceleration • At acceleration start • At acceleration end • At deceleration start • At deceleration end Irrespective of the acceleration model selected, the two jerk values for acceleration and deceleration always relate to the parameter (acceleration or deceleration) to which they are assigned via the **Dynamic response** tab. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed** See Selection list (combo box) (Page 4022). The system variable for the default is given together with the description of each command.
Velocity	Not always included on the **Dynamic response** tab. Velocity value during the constant velocity phase Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed velocity** **Current** See Selection list (combo box) (Page 4022). The system variable for the default is given together with the description of each command.

Field/Button	Explanation/Instructions
Constant traversing time	Only for the "Start axis position-controlled" (Page 4277) and "Speed specification" (Page 4273) commands. A time limit can be programmed for this command. • Activate the checkbox if the movement of the axis is to be time limited. If you do not activate the checkbox, the axis moves until it receives a new command. • Enter the duration of the constant motion phase (from the end of acceleration up to the start of deceleration). At the end of this time, the axis is decelerated until its speed reaches 0. If no time is specified, then the axis moves until it receives a new command. See also input field (Page 4022).
Time for deceleration	Only for the "Stop axis" (Page 4281) command. Duration of braking operation for a quick stop within a defined time period Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4022). **Default** **Last programmed** See Selection list (combo box) (Page 4022). System variable for default: userDefaultDynamics.stopTime

Expert tab

The parameter screen forms for many motion commands contain an **Expert** tab. The content of this tab depends on the motion command:

- You can define the variable for the CommandID.
- You can view configuration data or system variables that affect the parameter screen form, see Using a variable of the technology object data type (Page 4024).
- You can define the variable to which the return value is written (return variable).

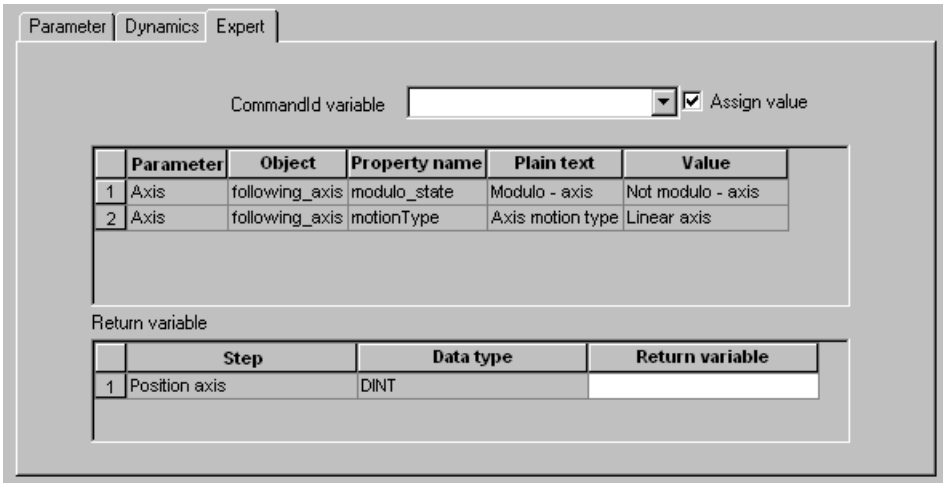


Figure 7-31 Expert tab for motion commands

Overview of parameters

Table 7-5 Overview of parameters for Expert tab

Field/Button	Explanation/Instructions
CommandID variable	The status of many motion commands can be tracked using the unique, project-wide CommandId. • If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable. The variables of this data type that were declared previously in the MCC unit or MCC chart are available for selection. – When the **Assign value** checkbox is activated, a unique project-wide CommandId is automatically generated, stored in the variables, and transferred to the command. You can use the stored CommandId to track the command status until the command has been completed properly or has been aborted. – When the **Assign value** checkbox is cleared, only the variable is transferred to the command. You must take steps to ensure that a unique project-wide CommandId is stored in the variable. You obtain the CommandId with the _getCommandId system function. In addition, you can buffer the CommandId in this case so that it is available even after the command has been completed. • If you leave the field empty, the CommandID is not assigned to any variables; this means that you cannot access the CommandID (default). When the field is empty, the **Assign value** checkbox must be activated.
TO properties	The technology object configuration data or system variables that affect the parameter screen form (presentation or selection options) are displayed in the "TO properties" list (see "TO properties list" table below). If you have selected a technology object-type variable (see Description of the technology object data types (Page 4060)) as the technology object (e.g. in a library), you can select a value in order to adapt the parameter screen form to the properties of the technology object. See Using a technology object-type variable (Page 4024). This selection does not change the configuration data or system variables of the technology object.
Return variable	Many commands are mapped onto one or more system functions (command steps). These usually have a return value that provides information to the user on the result of the system function. For each command step, you can specify a return variable in which the respective return value is stored: • When you enter the name of a variable of the specified data type, the return value of the command step is stored in this variable. The variables of this data type that were declared previously in the MCC unit or MCC chart are available for selection. A reference to the documentation describing the meaning of the individual values is given for each command. • If you leave the field empty, the return value of the command step is assigned to an internal variable that you cannot access (default). See also Relevant system function, system variable or command in the ST programming language (Page 4039) and Return values (Page 4039).

Table 7-6 TO properties list

Field/Button	Explanation/Instructions
Parameter	Indicates the name of the field for the technology object whose configuration data or system variables affect the parameter dialog box.
Object	Indicates the technology object selected in the parameter dialog box.
Property Name	Indicates the configuration data or system variable affecting the parameter dialog box.
Plain text	Indicates the meaning of the configuration data or system variable.
Value	The behavior depends on whether the technology object is defined on the SIMOTION device or whether it is a variable of the data type of a technology object: • If the technology object is defined on the SIMOTION device: The value defined during configuration is indicated. • If the technology object is a variable of the data type of a technology object: You can select a value from among the values of the enumeration data type of the configuration data or the system variable. The parameter dialog box is presented according to the selected value. This selection does not affect the configuration data or system variables of the technology object.

Transition behavior and step enabling condition

The transition behavior for all motion commands must be indicated by the motion command that is currently active on the axis. The transition behavior always relates to motion commands on the same axis irrespective of the program that issued the commands.

You can also program a step enabling condition to the next command in the flow chart.

Transition behavior from the currently active motion command

For motion commands, you specify the transition behavior from the command that is currently active on the axis. The table shows the transition behavior of the currently active command on the axis (thick line = programmed command, thin line = active command on the axis):

Transition behavior	Graph	Description
Substitute		The programmed command is executed immediately. The active command is aborted.
Attach		The programmed motion is attached to the active motion for this axis. Pending commands will be executed.
Attach and delete existing command		The programmed motion is attached to the active motion for this axis. Pending commands will be discarded.

Transition behavior	Graph	Description
Blending		The transition from the velocity of the active command to that of the programmed command takes place smoothly when deceleration begins in the active command. Therefore the active command should be programmed with step enabling condition **Start of deceleration phase**.
Superimpose		The programmed and active motions are superimposed. The command does not affect superimposed motions that are already active. The motion is started immediately.

Note

Not all of the options listed here are available in every case.

Delay program execution (step enabling condition)

This checkbox determines when the next command is to be executed. MCC is optimized for sequential programming; therefore, this checkbox is activated by default.

If the checkbox is clear ( graphic), the next command in the chart is started immediately after the current command or, where applicable, not until the command has been entered in the command buffer, see the detailed description in the "Overview of parameters" table for the relevant MCC command (Page 4173).

The table shows the step enabling conditions for the subsequent command when the checkbox is activated (trapezoid = current command, "MCC command" symbol = next command):

Wait for ...	Graph	Description
Motion start Interpolation start		The next command is started when the interpolation for the currently programmed motion begins.
Acceleration end		The next command is started when the acceleration phase for the active motion ends.
Speed/velocity reached		The next command is started when the speed/velocity for the active motion is reached.
Start of deceleration phase Deceleration start		The next command is started when the deceleration phase for the active motion begins.
End of setpoint interpolation		The next command is started when setpoint interpolation for the active motion has ended.
Motion is finished Axis stopped		The next command is started when the active motion has been completed (e.g. the minimum dwell time of the actual value in the configured position window around the target position has elapsed).
When clamping value is reached		The next command is started when the clamping value for the active motion is reached.
When command is completed or aborted		The next command is started when the current command is completed or aborted.

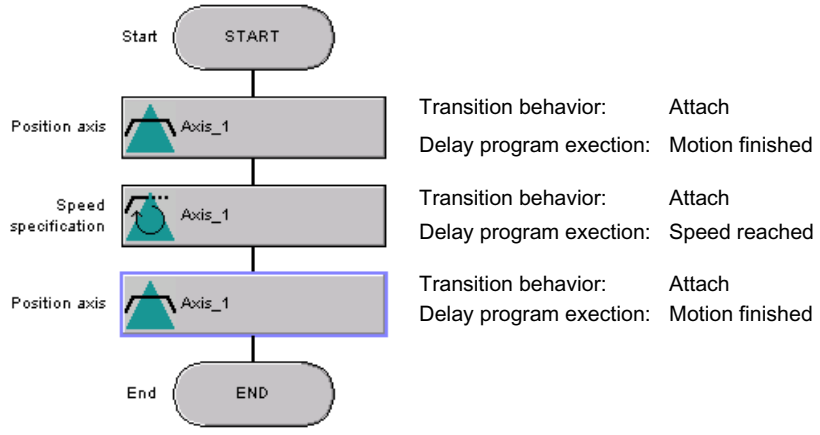
Wait for ...	Graph	Description
As soon as torque is limited		The next command is started as soon as torque limiting is triggered.
As soon as torque limitation is switched off		The next command is started as soon as torque limitation is switched off.
Axis synchronized		The next command is started when the axis is synchronized for the active motion.
Axis homed		The next command is started when the zero mark is detected for the active homing motion.

Note

Not all of the options listed here are available in every case.

You can query the current status of the axis by means of system variables. For a selection with corresponding values, see Axis status (Page 4177).

Example of transition behavior and step enabling condition



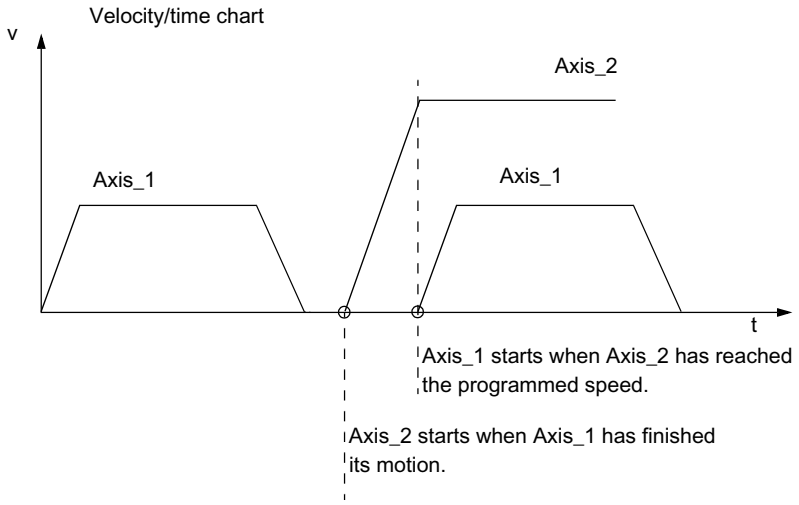


Figure 7-32 Example of transition behavior and step enabling condition

Note

The status of a motion command can be queried using the CommandID variables, see Expert tab (Page 4034). This may be necessary, in particular, if a transition to the next command is made before a motion has been completed (**Delay program execution** parameter is deactivated or set to an appropriate value, such as Motion Start, Velocity Reached).

The following system functions can be used for this purpose:

- For querying single axis commands:
  _getStateOfAxisCommand
  _getMotionStateOfAxisCommand
- For querying commands for synchronous operation and camming:
  _getStateOfFollowingObjectCommand
  _getMotionStateOfFollowingObjectCommand

For a description of the functions, refer to the SIMOTION Cam Technology Package, System Functions, List Manual.

ST commands can be programmed using the "System function call" (Page 4185) or "ST zoom" (Page 4188) command, for example.

Closing the parameter screen form

Click **OK** to close the parameter dialog box.

The syntax of the assigned command is checked when you close the screen form. Any errors are displayed.

Note

When a syntax check is performed on an MCC command, the declaration table in the MCC chart and the MCC source file are also read. Inconsistent data within the declaration tables may, therefore, cause unexpected error messages during parameter assignment.

Relevant system function, system variable, or command in the ST programming language

Each MCC command is mapped to a system function, an ST programming language command, or a value assignment to a system variable. This is specified for each MCC command.

For system function, the way in which the parameters of the MCC command are assigned to the parameters of the system function is also specified.

Return values

Many commands are mapped onto one or more system functions (command steps). These usually have a return value that provides information to the user on the result of the system function.

For these MCC commands, you can specify a return variable in which the respective return value is stored for each command step in the "Expert" tab (Page 4034):

- When you enter the name of a variable of the specified data type, the return value of the command step is stored in this variable. The variables of this data type that were declared previously in the MCC unit or MCC chart are available for selection.
  A reference to the documentation describing the meaning of the individual values is given for each command.
- If you leave the field empty, the return value of the command step is assigned to an internal variable that you cannot access (default).

Return value for the system functions of the technology packages

The return values for the system functions of the Cam, Path and Cam_EXT technology packages are described in the table.

Table 7-7 Description of return value for Cam, Path and Cam_EXT technology package functions

Error code	Meaning
0	No error
1	Invalid command parameter
2	Invalid range information in command parameters
3	Command aborted
4	Unknown command
5	Command cannot be executed due to current object state
6	Command aborted due to termination of user task
7	Command rejected due to suspension of command interpretation of the addressed technology object
8	Command aborted due to full command buffer
9	Insufficient memory
10	A connection to a technology object required for this operation does not exist
11	No object configuration
12	The error to be reset cannot be reset due to its configuration
13	Axis is not homed
14	Measurement job on virtual axis not possible
15	Ambiguous commandId
16	Command not implemented
17	Read access denied
18	Write access denied
19	Command argument not supported
20	The cam has already been interpolated and cannot be manipulated
21	The interpolation condition was violated
22	The programmed jerk is 0
23	The alarm to be deleted is not present
24	Command not possible on a virtual axis
25	A synchronized start of this command is not possible

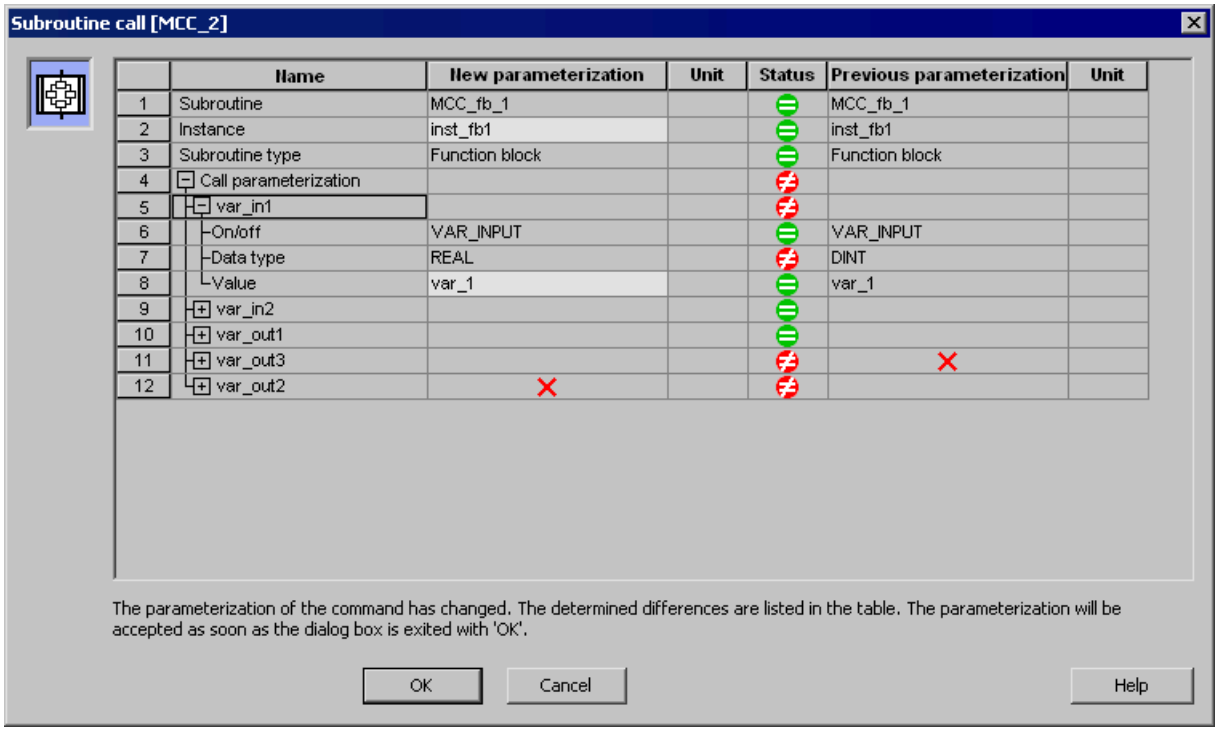
Error code	Meaning
26	Higher-level command was aborted because it is not permitted by the active command
27	Time exceeded during communication with the drive
28	Actual values are not valid
29	This command cannot be executed when velocity control is active
30	This command cannot be executed when position control is active
31	This command cannot be executed in torque-reduced operation or during travel to fixed limit stop
32	This command can only be executed when force/pressure control is active
33	This command cannot be executed when force/pressure control is active
34	This command can only be executed when pressure limiting is active
35	Master values are not valid
36	Slave values are not valid
37	No slave value can be determined for a master value
38	No master value can be determined for a slave value
39	This command cannot be executed when synchronous operation is inactive
40	This command cannot be executed because of a synchronization error.
41	The command cannot be executed when gearing or camming is active.
42	This command cannot be executed when camming is inactive
43	This command can only be used for an interpolated cam
44	This command cannot be executed when pressure limiting is active
45	Insufficient interpolation points are available to interpolate the cam
46	The specified path point cannot be reached due to restrictions of the kinematics.
47	Path axis values are not valid.
48	Maximum number of active commands exceeded.
49	Command only possible on a CPU local technology object
50	The command cannot be selected in this technology object configuration.
51	The multiple instantiation of the PLCopen function block on a technology object is not permitted.
52	No entry with the specified ID found.
53	The number of cam interpolation points exceeds the configured maximum limit.
≥ 10000	Internal error.

Updating a command in the event of parameterization changes

The MCC editor can provide support if the current parameterization for an MCC command differs from that saved in the MCC chart. This may happen, for example, in the following cases:

- For the "subroutine call" command:
  The parameterization of the subroutine (e.g. function or function block) called has changed, for example:
  - Additional or deleted parameters
  - Changed identifiers of parameters or their data types
- For motion commands:
  - Changes to the technology object's configuration data or system variables affecting the layout or selection options of the parameter screen form (e.g. linear axis/rotary axis, position-based cam/uni-directional output cam), see Using the technology object (Page 4024)
  - Expansions of a command's function (e.g. new parameters, new selection options)
    This may be the case, for example, when the device is being upgraded to a new version of SIMOTION Kernel or when an MCC chart or MCC unit is being copied from an earlier project.

A comparison screen form opens, as well as the parameter screen form, when parameters are reassigned for the relevant MCC command. This comparison screen form lists the previous parameterization (e.g. parameter identifiers, data types) and the new parameterization in a tabular format. Graphical symbols in the "Status" column highlight the differences and similarities. The symbol in the relevant column indicates that parameters or parameter properties are missing. You can also assign new values to parameters in the "New parameterization" column.



The new parameterization (parameter and parameter property) matches the old one.

The new parameterization (parameter or parameter property) differs from the old one.

Parameter or parameter property is missing for the relevant parameterization.

Figure 7-33 Comparison screen form for changed parameterization

Procedure

Proceed as follows if the parameterization for an MCC command has changed and the comparison screen form opens as well as the parameter screen form:

1. Look for the ≠ and ✕ symbols to help you find the parameterization differences in the comparison screen form.
   If necessary, open the subtree for a parameter to find out more details.
2. If necessary, assign new values to the parameters.
3. Click **OK** to confirm.
   The new parameterization and the assignments you have made are accepted by the parameter screen form.
   (The existing parameterization for the command is retained if you click **Cancel**.)
4. Check the parameterization in the parameter screen form once more.
5. Click **OK** to confirm.
   The changed parameterization is accepted by the MCC chart.

Note

Once you have had the changed parameterization accepted by the parameter screen form and then the MCC chart, the comparison screen form will not appear the next time the parameter screen form is opened.

7.1.5.4 General information about variables and data types

Overview of variable types

The following table shows all the variable types available for programming with ST.

- System variables of the SIMOTION device and the technology objects
- Global user variables (I/O variables, device-global variables, unit variables)
- Local user variables (variables within a program, a function, or a function block)

System variables

Variable type	Meaning
System variables of the SIMOTION device	Each SIMOTION device and technology object has specific system variables. These can be accessed as follows: • Within the SIMOTION device from all programs • From HMI devices You can monitor system variables in the symbol browser.
System variables of technology objects	

Global user variables

Variable type	Meaning
I/O variables	You can assign symbolic variable names to the I/O addresses of the SIMOTION device or the peripherals. This allows you to have the following direct accesses to the I/O: • Within the SIMOTION device from all programs • From HMI devices You create these variables in the symbol browser after you have selected the I/O element in the project navigator. You can monitor I/O variables in the symbol browser.
Global device variables	User-defined variables which can be accessed by all SIMOTION device programs and HMI devices. You create these variables in the symbol browser after you have selected the GLOBAL DEVICE VARIABLES element in the project navigator. Global device variables can be defined as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor global device variables in the symbol browser.
Unit variables	User-defined variables that all programs (programs, function blocks, and functions) can access within a unit (source file). You declare these variables in the declaration table of the source file: • In the interface section: These variables are exported and can be used in other units (e.g. ST source files, MCC source files, LAD/FBD source files) after a connection has been defined (Page 4115). They are also available on HMI devices as standard. • In the implementation section: You can only access these variables within the source file. You can declare unit variables as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor unit variables in the symbol browser.

Local user variables

Variable type	Meaning
	User-defined variables that can only be accessed within the program/chart (program, function, function block) in which they were defined.
Variable of a program (program variable)	Variable is declared in a program. The variable can only be accessed within this program. A differentiation is made between static and temporary variables: • Static variables are initialized according to the memory area in which they are stored. Specify this memory area by means of a compiler option. By default, the static variables are initialized depending on the task to which the program is assigned (see *SIMOTION Basic Functions* Function Manual). You can monitor static variables in the symbol browser. • Temporary variables are initialized every time the program in a task is called. Temporary variables cannot be monitored in the symbol browser.

Variable type	Meaning
Variable of a function (FC variable)	Variable is declared in a function (FC). The variable can only be accessed within this function. FC variables are temporary; they are initialized each time the FC is called. They cannot be monitored in the symbol browser.
Variable of a function block (FB variable)	Variable is declared in a function (FB). The variable can only be accessed within this function block. A differentiation is made between static and temporary variables: • Static variables retain their value when the FB terminates. They are initialized only when the instance of the FB is initialized; this depends on the variable type with which the instance of the FB was declared. You can monitor static variables in the symbol browser. • Temporary variables lose their value when the FB terminates. The next time the FB is called, they are reinitialized. Temporary variables cannot be monitored in the symbol browser.

Scope of the declarations

Scope of variable and data type declarations according to location of declaration

Location of declaration	What can be declared here	Scope
Symbol browser	• Global device variables • I/O variables	The declared variables are valid in all units (e.g., ST source files, MCC source files, LAD/FBD source files) of the SIMOTION device. All programs, function blocks, and functions in all units of the device can access the variables.
Interface section of the unit[1]	• Unit variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., are valid in the entire unit (e.g., ST source file, MCC source file, LAD/FBD source file); all programs, function blocks, and functions within the unit can access them. In addition, they are also available in other units after connection (see Define connections (Page 4115)).
Implementation section of the unit[1]	• Unit variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., are valid in the entire unit (e.g., ST source file, MCC source file, LAD/FBD source file); all programs, function blocks, and functions within the source file can access them.
POU (program/ function block/ function)[2]	• Local variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., can only be accessed within the POU in which they were declared.

[1] MCC and LAD/FBD programming languages: in the declaration table of the respective source file.

[2] MCC and LAD/FBD programming languages: in the declaration table of the respective chart/program.

Rules for identifiers

Names for variables, data types, charts/programs must comply with the following rules for identifiers:

1. They are made up of letters (A to Z, a to z), numbers (0 to 9), and underscores (_).
2. The first character must be a letter or underscore.
3. This can be followed by as many letters, digits or underscores as needed in any order.
4. Exception: You must not use more than one underscore in succession.
5. Both upper- and lower-case letters are allowed. No distinction is made between upper- and lower-case notation (thus, for example, Anna and AnNa are regarded as identical).

Note

Reserved identifiers

Reserved identifiers may only be used as predefined. You may not declare a variable or data type with the name of a reserved identifier.

There is no distinction between upper- and lower-case notation.

You can find a list of all the identifiers whose meanings are predefined in SIMOTION in the SIMOTION Basic Functions Function Manual.

Note

Identifiers for SIMOTION devices

Identifiers for SIMOTION devices do not have to comply with rules specified above. When used in SIMOTION SCOUT they must be enclosed in double inverted commas (", ASCII code $22). See the SIMOTION ST Programming and Operating Manual.

Frequently used arrays in declarations

Reference (as of kernel V4.5)

References can be formed as of version V4.5 SIMOTION Kernel. The compiler option (Page 3996) "Object-oriented programming" must also be activated.

By activating the **Reference** checkbox, you specify that the declared variable (or the component of a structure) contains the reference to the specified data type. This means that the variable can contain address information for a variable of the specified data type.

The following are permitted as data types to which references can be formed:

- Elementary data types (e.g. INT, DINT, REAL, WORD, TIME, STRING)
- User-defined data types (UDT)
- System data types
- Function blocks, provided they contain at least one static variable
- Classes (from ST sources)

No references can be formed to the following data types because references exist already:

- Technology object data types
- Object-oriented interfaces (from ST sources)
- General references
- I/O references (from ST sources)

Array length and array element

A field is a chain of variables of the same type that can be addressed with the same name and different indices.

You can define the variable as a field [0...N-1] by entering a field length N. You have the following options for entering the field length:

- You can enter a constant positive integer value.
- You can enter a value range with ".." separating the min. and max. values.
- You can enter a constant expression of data type DINT (or of a data type which is implicitly convertible to DINT).

If the field is empty, a single variable is set up rather than a field.

Example definition of an field in the declaration table

	Name	Variable type	Data type	Array length	Initial value	Comment
1	const_1	VAR_GLOBAL CONSTANT	INT		11	constant
2	const_2	VAR_GLOBAL CONSTANT	INT		5	constant
3	array_4	VAR_GLOBAL	INT	11	11(5)	specification of the array length by value
4	array_5	VAR_GLOBAL	INT	const_1	11(5)	specification of the array length by constant expression
5	array_6	VAR_GLOBAL	INT	-5 .. 5	11(5)	specification of the array length by range of values
6	array_7	VAR_GLOBAL	INT	const_2 .. 3 * const_2	11(5)	specification of the array length by range of values as constant expression
7						

Figure 7-34 Defining the length of a field

Example of use of field elements in a variable assignment

	Variable	:=	Expression
1	array_1[0]	:=	4
2	array_2[3]	:=	array_2[4]
3		:=	

Figure 7-35 Use of field elements in a variable assignment

Initial value

You can specify an initialization value in this column. The field cannot be edited, but is provided as button.

1. Click the button. A window opens:

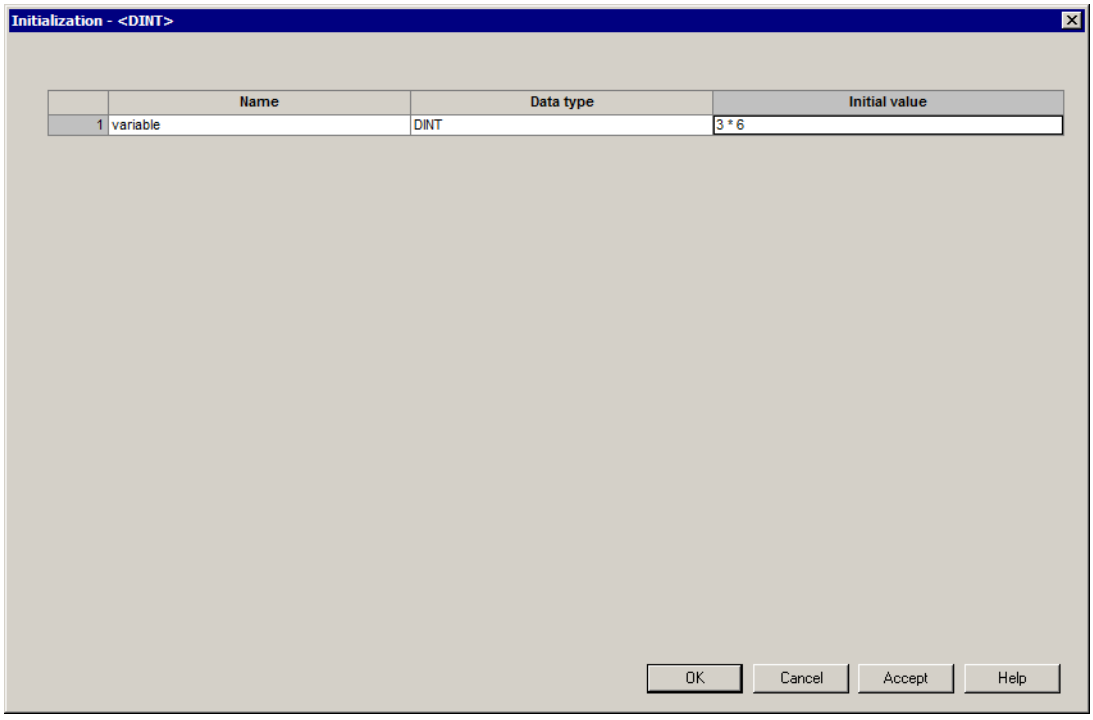


Figure 7-36 Definition an initialization value

2. Enter the initialization value in the **Initial value** column.
   You can specify this initialization value as a constant or as an expression. The following are permissible:
   - Constants
   - Arithmetic operations
   - Bit slice and data conversion functions
3. Confirm with **OK**.

Variables of a technology object data type cannot be assigned an initialization value. They are always initialized by the compiler with TO#NIL.

Initialization of arrays

Per default, the compiler assigns all array elements the initialization value of the data type. The initialization values can be changed by specifying an array initialization list according to the following example.

Table 7-8 Preassignment of array elements

10(1)	10 array elements [0..9] are pre-assigned the same value "1".
1,2,3,4,5	5 array elements [0..4] are pre-assigned different values "1", "2", "3", "4" and "5".
5(3),10(99),3(7),2(1)	The following array elements are pre-assigned the following values: • Five array elements [0..4] with the same value "3". • 10 array elements [5..14] with the same value "99". • 3 array elements [15..17] with the same value "7". • 2 array elements [18..19] with the same value "1".

Definition of these initialization values in the declaration table:

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	array_1	VAR_GLOBAL	INT		10	10(1)	all array elements equal
2	array_1_	VAR_GLOBAL	INT		5	1, 2, 3, 4, 5	all array elements diverse
3	array_1_	VAR_GLOBAL	INT		20	5(3), 10(99), 3(7), 2(1)	several array elements respectively equal
4							

Figure 7-37 Definition of the initialization values of an array

Initialization of structures

Per default, the compiler assigns all components of a structure the initialization value of their data type. By specifying an initial value for a component, its initialization can be changed.

When using the structure in another declaration (e.g. variable or structure definition), the initialization values of individual components can be changed by assigning a structure initialization list, enclosed in round brackets ( ), in accordance with the following example.

	Structure name	Element name	Data type	Array length	Initial value	Comment
1	type_struct_1	s11	INT		1	
2		s12	DINT	3	3(2)	
3		s13	UDINT		3	
4	type_struct_2	s21	LREAL		3.0	
5		s22	DWORD		16#0000_FFFF	
6	type_struct_3	s31	type_struct_1		(s11 := 10, s12 := [3(20)])	
7		s32	type_struct_2		(s22 := 16#0000_00FF)	
8		s33	REAL		10.0 / 3.0	
9						

Figure 7-38 Definition of the initialization values of a structure

Instance-specific initialization of classes and function blocks

For the instance formation of classes and function blocks created during the object-oriented programming, variables declared as VAR OVERRIDE or VAR PUBLIC can be initialized instance-specific. The initialization values specified for the definition of the associated program organization unit (POU) are overwritten.

Proceed as follows to initialize the instance variables instance-specific:

1. Click the button in the **Initial value** column. A window opens.

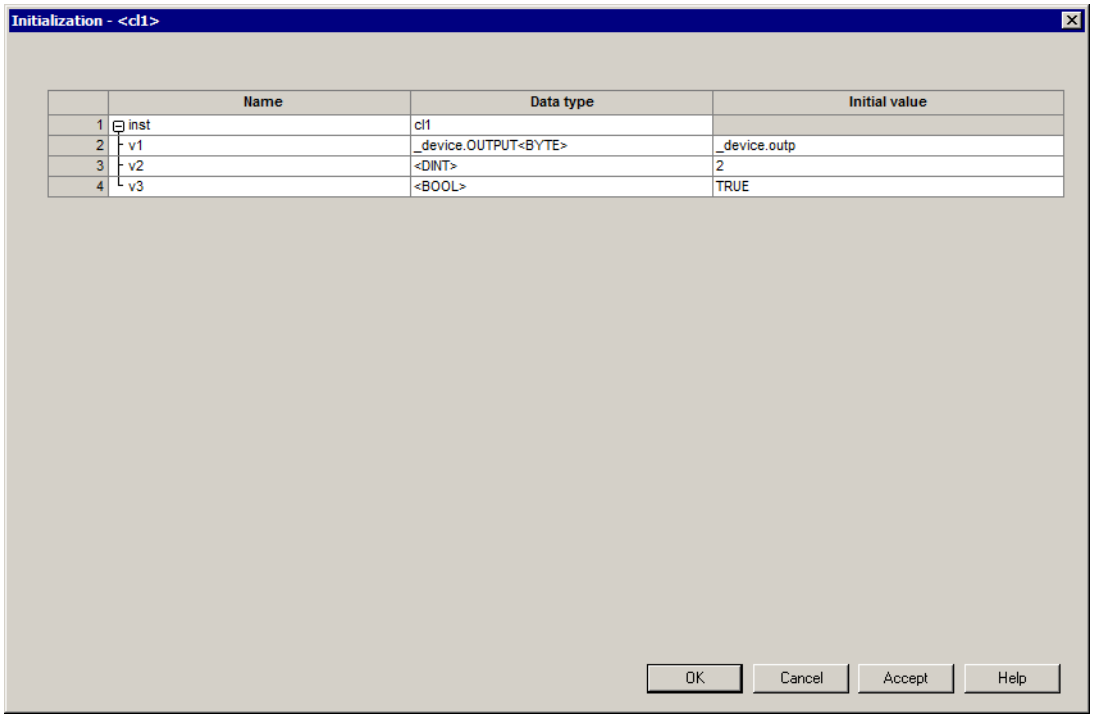


Figure 7-39 Definition of the initialization values of a class or function block

   Variables that must be initialized are already displayed in this window with their name and data type; the associated line is color-highlighted.

2. You can add additional lines for variables whose initialization is optional:
   - Click the number in line 1 and select **Add new line** in the context menu.
   - Enter the name of the variable. The data type is then initialized automatically.
3. Enter the initialization values of the associated variables.
4. Click **OK** to confirm.

Comments

A comment can be entered in this column. It may contain any characters or special characters.

Sorting in the declaration tables

You can specify a sorting order for a column in the declaration tables. The lines of the declaration table are arranged so that the entries (character strings) of the relevant column are sorted according to the sorting criterion. The characters are sorted according to their ANSI code; the sorting is not case-sensitive.

The following sorting criteria are available:

- **Sort in ascending order**
  Alphabetic sorting is ascending order (0 ... 9, a ...z).
- **Sort in descending order**
  Alphabetic sorting is descending order (z ... a, 9 ... 0).
- **Original order**
  No sorting. The lines of the declaration table are arranged in the order of declaration.

As long as the sorting order is not accepted (see below), the following applies:

- The sorting only affects the display. The data storage remains unchanged.
- The declaration table is saved in the original order.
- The compiler processes the declaration table in the original order.

Special features when sorting structures and enumerations

When sorting structures (Page 4058) and enumerations (Page 4059), the following applies:

- When sorting according to the **Structure name** or **Enumeration name** column:
  - The structures or enumerations are sorted according to their names.
  - The elements of these structures and enumerations remain in their original order.
- When sorting according to the other columns:
  - The structures and enumerations remain in their original order.
  - The elements within the structures and enumerations are sorted.

Special feature of pragma lines

For the declaration tables of a unit (interface section and implementation section), the following applies:

If in the declaration of unit variables (Page 4064) (Parameters tab), pragma lines (Page 4070) are available, the lines are sorted between the individual pragma lines.

Note

In the sorted view of a declaration table, the following is not possible:

- Insertion of new elements (e.g. variables, structures, enumerations as well as elements of structures and enumerations).
- Insertion of new or copied lines.
- Insertion of pragma lines.

Procedure

To sort the entries of the declaration table:

- Double-click the appropriate column header.
  The sorting changes between the sorting criteria.
  A double-click in another column header results sorting in ascending order according to this column.
- Or alternatively:
  - Move the cursor to the appropriate column header.
  - Select the sorting criterion in the context menu

  The sorting is performed according to the selected sorting criterion.
  The original order can only be selected in the context menu of a sorted column.

Accepting the sorting order

Proceed as follows:

1. Move the cursor to an arbitrary column header.
2. Select **Accept sorting order** in the context menu.

The sorting order is taken over as original order.

Note

Observe the following when accepting the sorting order in a declaration table:

- No "Undo" possible.
- The data storage and the declaration order are changed.
- The corresponding unit is changed and must be compiled again.
- The compiler processes the declaration table in the changed order.
- The relevant variable blocks are initialized during the download of the unit.
- HMI-relevant data is no longer consistent.

7.1.5.5 Data types

Data types

A data type is used to determine how the value of a variable or constant in a program source is to be used.

The following data types are available to the user:

- Elementary data types (Page 4054)
- User-defined data types (UDT) (Page 4057)
  - Enumerations
  - Structures (Struct)

- Technology object data types (Page 4060)
- System data types (Page 4061)

Elementary data types

Elementary data types define the structure of data that cannot be broken down into smaller units. An elementary data type describes a memory area with a fixed length and stands for bit data, integers, floating-point numbers, duration, time, date and character strings.

All the elementary data types are listed in the table below:

Table 7-9 Bit widths and value ranges of the elementary data types

Type		Reserv. word	Bit width	Range of values
Bit data type Data of this type uses either 1 bit, 8 bits, 16 bits, or 32 bits. The initialization value of a variable of this data type is 0.				
	Bit	BOOL	1	0, 1 or FALSE, TRUE
	Byte	BYTE	8	16#0 to 16#FF
	Word	WORD	16	16#0 to 16#FFFF
	Double word	DWORD	32	16#0 to 16#FFFF_FFFF
Numeric types These data types are available for processing numeric values. The initialization value of a variable of this data type is 0 (all integers) or 0.0 (all floating-point numbers).				
	Short integer	SINT	8	-128 to 127 (-2**7 to 2**7-1)
	Unsigned short integer	USINT	8	0 to 255 (0 to 2**8-1)
	Integer	INT	16	-32_768 to 32_767 (-2**15 to 2**15-1)
	Unsigned integer	UINT	16	0 to 65_535 (0 to 2**16-1)
	Double integer	DINT	32	-2_147_483_648 to 2_147_483_647 (-2**31 to 2**31-1)
	Unsigned double integer	UDINT	32	0 to 4_294_967_295 (0 to 2**32-1)
	Floating-point number (per IEEE -754)	REAL	32	-3.402_823_466E+38 to -1.175_494_351E-38, 0.0, +1.175_494_351E-38 to +3.402_823_466E+38 Accuracy: 23-bit mantissa (corresponds to 6 decimal places), 8-bit exponent, 1-bit sign.
	Long floating-point number (in accordance with IEEE-754)	LREAL	64	-1.797_693_134_862_315_7E+308 to -2.225_073_858_507_201_4E-308, 0.0, +2.225_073_858_507_201_4E-308 to +1.797_693_134_862_315_7E+308 Accuracy: 52-bit mantissa (corresponds to 15 decimal places), 11-bit exponent, 1-bit sign.
Time types These data types are used to represent various date and time values.				

Type		Reserv. word	Bit width	Range of values
	Duration in increments of 1 ms	TIME	32	T#0d_0h_0m_0s_0ms to T#49d_17h_2m_47s_295ms Maximum of 2 digits for the values day, hour, minute, second and a maximum of 3 digits for milliseconds Initialization with T#0d_0h_0m_0s_0ms
	Date in increments of 1 day	DATE	32	D#1992-01-01 to D#2200-12-31 Leap years are taken into account, year has four digits, month and day are two digits each Initialization with D#0001-01-01
	Time of day in increments of 1 ms	TIME_OF_DAY (TOD)	32	TOD#0:0:0.0 to TOD#23:59:59.999 Maximum of two digits for the values hour, minute, second and maximum of three digits for milliseconds Initialization with TOD#0:0:0.0
	Date and time	DATE_AND_TIME (DT)	64	DT#1992-01-01-0:0:0.0 to DT#2200-12-31-23:59:59.999 DATE_AND_TIME consists of the data types DATE and TIME Initialization with DT#0001-01-01-0:0:0.0
String type Data of this type represents character strings, in which each character is encoded with the specified number of bytes. The length of the string can be defined at the declaration. Indicate the length in "[" and "]", e.g. STRING[100]. The default setting consists of 80 characters. The number of assigned (initialized) characters can be less than the declared length.				
	String with 1 byte/character	STRING	8	You may use all the characters in the extended ASCII character set (ASCII code $00 to $FF). Default ' ' (empty string)

Note

During variable export to other systems, the value ranges of the corresponding data types in the target system must be taken into account.

See also

Value range limits of elementary data types (Page 4055)

General data types (Page 4056)

Elementary system data types (Page 4057)

Value range limits of elementary data types

The value range limits of certain elementary data types are available as constants.

Table 7-10 Symbolic constants for the value range limits of elementary data types

Symbolic constant	Data type	Value	Hex notation
SINT#MIN	SINT	-128	16#80
SINT#MAX	SINT	127	16#7F
INT#MIN	INT	-32768	16#8000

Symbolic constant	Data type	Value	Hex notation
INT#MAX	INT	32767	16#7FFF
DINT#MIN	DINT	-2147483648	16#8000_0000
DINT#MAX	DINT	2147483647	16#7FFF_FFFF
USINT#MIN	USINT	0	16#00
USINT#MAX	USINT	255	16#FF
UINT#MIN	UINT	0	16#0000
UINT#MAX	UINT	65535	16#FFFF
UDINT#MIN	UDINT	0	16#0000_0000
UDINT#MAX	UDINT	4294967295	16#FFFF_FFFF
T#MIN TIME#MIN	TIME	T#0ms	16#0000_0000[1]
T#MAX TIME#MAX	TIME	T#49d_17h_2m_47s_295ms	16#FFFF_FFFF[1]
TOD#MIN TIME_OF_DAY#MIN	TOD	TOD#00:00:00.000	16#0000_0000[1]
TOD#MAX TIME_OF_DAY#MAX	TOD	TOD#23:59:59.999	16#0526_5BFF[1]
[1] Internal display only			

General data types

General data types are often used for the input and output parameters of system functions and system function blocks. The subroutine can be called with variables of each data type that is contained in the general data type.

The following table lists the available general data types:

Table 7-11 General data types

General data type	Data types contained
ANY_BIT	BOOL, BYTE, WORD, DWORD
ANY_INT	SINT, INT, DINT, USINT, UINT, UDINT
ANY_REAL	REAL, LREAL
ANY_NUM	ANY_INT, ANY_REAL
ANY_DATE	DATE, TIME_OF_DAY (TOD), DATE_AND_TIME (DT)
ANY_ELEMENTARY	ANY_BIT, ANY_NUM, ANY_DATE, TIME, STRING
ANY	ANY_ELEMENTARY, user-defined data types (UDT), system data types, data types of the technology objects

Note

You **cannot** use general data types as type identifiers in variable or type declarations.

The general data type is retained when a user-defined data type (UDT) is derived directly from an elementary data type (only possible with the SIMOTION ST programming language).

Elementary system data types

In the SIMOTION system, the data types specified in the table are treated in a similar way to the elementary data types. They are used with many system functions.

Table 7-12 Elementary system data types and their use

Identifier	Bit width	Use
StructAlarmId	32	Data type of the alarmId for the project-wide unique identification of the messages. The alarmId is used for the message generation. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTALARMID#NIL
StructTaskId	32	Data type of the taskId for the project-wide unique identification of the tasks in the execution system. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTTASKID#NIL

Table 7-13 Symbolic constants for invalid values of elementary system data types

Symbolic constant	Data type	Meaning
STRUCTALARMID#NIL	StructAlarmId	Invalid AlarmId
STRUCTTASKID#NIL	StructTaskId	Invalid TaskId

User-defined data types (UDT)

Defining user-defined data types (UDT)

You can create user-defined data types in units and programs/charts:

- Structures (Page 4058)
- Enumerations (Page 4059)

The scope of the data type declaration (Page 4058) depends on the location of the declaration.

Scope of the data type declaration

You create derived data types in the declaration tables of the source file or the program/chart. The scope of the data type declaration depends on the location of the declaration.

- In the declaration table of the unit, **Interface (exported declaration)** section:
  The data type is valid for the entire source file; all programs/charts (programs, function blocks, and functions) within the source file can access the data type.
  These variables are also available, if appropriately connected (see Define connections (Page 4115)), in other source files (or other units).
- In the declaration table of the unit, **Implementation (unit-internal declaration)** section:
  The data type is valid in the source file; all programs/charts (programs, function blocks, and functions) within the source file can access the data type.
- In the declaration table of the program/chart:
  The data type can only be accessed within the program/chart in which it is declared.

Defining structures

You define structures in the declaration tables of the unit or the program/chart. The scope of the structures depends on the location of the declaration.

To define structures, proceed as follows:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Structures** tab.
3. Enter the name of the structure.
4. In the same line, enter:
   - The name of the first component in the **Element name** field.
   - Data type of the component.
     You can select already defined data types.
     See also:
     Elementary data types (Page 4054).
     User-defined data types (Page 4057).
     Technology object data types (Page 4060).
     System data types (Page 4061).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 4047).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 4048).
   - Optional initial value (initialization value).
     See also: Initial value (Page 4049).
   - Optional comment.
     See also: Comment (Page 4051).

5. Enter additional elements of the structure in the following lines; leave the **Structure name** field empty.
6. Begin the definition of the new structure by entering a new name in the **Structure name** field.

Example

This example shows the definition of a structure with three components:

	Structure name	Element name	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	t_struct	comp_1	INT	☐			
2		comp_2	REAL	☐	5	3(2), 2(3)	
3		compp_3	STRING	☐		'Example'	
4				☐			

Figure 7-40 Definition of a structure

Defining enumerations

You define **enumerations** in the declaration tables of the unit or the program/chart. The scope (Page 4058) of the enumerations depends on the location of the declaration.

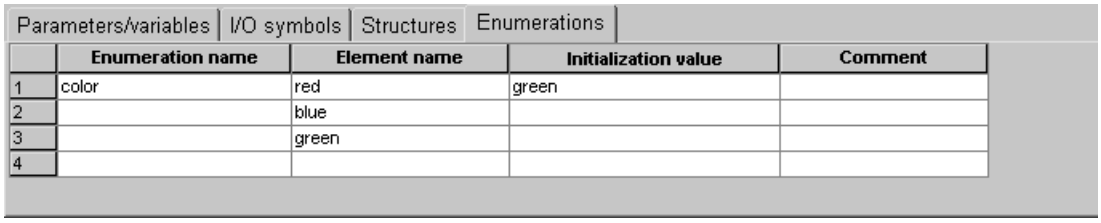
To define enumerations, proceed as follows:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Enumerations** tab.
3. Enter the name of the enumeration.
4. In the same line, enter:
   - The name of the first element
   - Optionally, the initialization value of the enumeration data type
5. Enter additional elements of the enumeration in the following lines; leave the **Enumeration name** field empty.
6. You begin the definition of the new enumeration by entering a new name in the **Enumeration name** field.

Example

This example shows the definition of an enumeration data type with the name **Color** and the enumeration elements **Red**, **Blue**, and **Green**, as well as the initialization value (initial value) **Green**.

If no initialization value is entered during the enumeration definition (data type declaration), the first value of the enumeration is assigned to the data type. In this example, this means **Red** would be used for the initialization because it is defined as the first enumeration element.


Parameters/variables | I/O symbols | Structures | Enumerations

	Enumeration name	Element name	Initialization value	Comment
1	color	red	green	
2		blue		
3		green		
4				

Figure 7-41 Definition of an enumeration data type

Technology object data types

Description of the technology object data types

You can declare variables with the data type of a technology object (TO). The following table shows the data types for the available technology objects in the individual technology packages.

For example, you can declare a variable with the data type *posaxis* and assign it an appropriate instance of a position axis. Such a variable is often referred to as a reference.

Table 7-14 Data types of technology objects (TO data type)

Technology object	Data type	Contained in the technology package
Drive axis	DriveAxis	CAM, PATH[1], CAM_EXT
External encoder	ExternalEncoderType	CAM, PATH[1], CAM_EXT
Measuring input	MeasuringInputType	CAM, PATH[1], CAM_EXT
Output cam	OutputCamType	CAM, PATH[1], CAM_EXT
Cam track	_CamTrackType	CAM, PATH[1], CAM_EXT
Position axis	PosAxis	CAM, PATH[1], CAM_EXT
Following axis	FollowingAxis	CAM, PATH[1], CAM_EXT
Following object	FollowingObjectType	CAM, PATH[1], CAM_EXT
Cam	CamType	CAM, PATH[1], CAM_EXT
Path axis[1]	_PathAxis	PATH[1], CAM_EXT
Path object[1]	_PathObjectType	PATH[1], CAM_EXT
Fixed gear	_FixedGearType	CAM_EXT
Addition object	_AdditionObjectType	CAM_EXT
Formula object	_FormulaObjectType	CAM_EXT
Sensor	_SensorType	CAM_EXT
Controller object	_ControllerObjectType	CAM_EXT
Temperature channel	TemperatureControllerType	TControl
General data type, to which every TO can be assigned	ANYOBJECT	

[1] Available as of version V4.1.

You can access the elements of technology objects (configuration data and system variables) via structures (see SIMOTION Basic Functions Function Manual).

Table 7-15 Symbolic constants for invalid values of technology object data types

Symbolic constant	Data type	Meaning
TO#NIL	ANYOBJECT	Invalid technology object

Inheritance of the properties for axes

Inheritance for axes means that all of the data types, system variables and functions of the TO driveAxis are fully included in the TO positionAxis. Similarly, the position axis is fully included in the TO synchronizedAxis, the following axis in the TO pathAxis. This has, for example, the following effects:

- If a function or a function block expects an input parameter of the driveAxis data type, you can also use a position axis or a synchronized axis or a path axis when calling.
- If a function or a function block expects an input parameter of the posAxis data type, you can also use a synchronized axis or a path axis when calling.

System data types

There are a number of system data types available that you can use without a previous declaration. And, each imported technology packages provides a library of system data types.

Additional system data types (primarily enumeration and STRUCT data types) can be found

- In parameters for the general standard functions (see SIMOTION Basic Functions Function Manual)
- In parameters for the general standard function blocks (see SIMOTION Basic Functions Function Manual)
- In system variables of the SIMOTION devices (see relevant parameter manuals)
- In parameters for the system functions of the SIMOTION devices (see relevant parameter manuals)
- In system variables and configuration data of the technology objects (see relevant parameter manuals)
- In parameters for the system functions of the technology objects (see relevant parameter manuals)

7.1.5.6 Variables

Variables are an important component of programming and provide structure to programs. They are placeholders which can be assigned values that can be accessed several times in the program.

Variables have:

- A specific initialization behavior and scope of validity
- A data type and operations which are defined for that data type

User and system variables are differentiated. User variables can be defined by the user. System variables are provided by the system.

Keywords for variable types

The various keywords for variable types are shown in the following table.

Description of keywords for variable types

Keyword	Description	Use
Global user variables (declared in the interface or implementation section of the unit[1])		
VAR_GLOBAL	Unit variable; can be accessed by all POUs within the source file. If the variable was declared in the interface section, it can be used in another source file once a connection has been defined in its declaration table (see Define connections (Page 4115)).	FB, FC, program
VAR_GLOBAL RETAIN	Retentive unit variable; retained during power outage.	FB, FC, program
VAR_GLOBAL CONSTANT	Unit constant; cannot be changed from the program.	FB, FC, program
Local user variables (declared within a POU[2])		
VAR	Local variable (static for FB and program, temporary for FC)	FB, FC, program
VAR_TEMP	Temporary local variable	FB, FC, program
VAR_INPUT	Input parameters: Local variable; value is supplied from external source and can only be read in the FB or FC.	FB, FC
VAR_OUTPUT	Output parameters: Local variable; value is sent to an external destination by the FB. It can be read as an instance variable after being called by the FB (FB instance name.variable name).	FB, FC
VAR_IN_OUT	In/out parameter; the FB or FC accesses this variable directly (by means of a reference) and can change it directly.	FB, FC
VAR OVERRIDE	Static variable whose initialization value can be overwritten for an instance formation. The compiler option (Page 3994) "Permit object-oriented programming" must be activated.	FB
VAR CONSTANT	Local constant; cannot be changed from the program.	FB, FC, program
[1] MCC and LAD/FBD programming languages: in the declaration table of the associated source file. [2] MCC and LAD/FBD programming languages: in the declaration table of the associated chart/program.		

Defining variables

Variables are defined in the symbol browser or in the declaration table of the source file or chart/program. The following table provides an overview of where the relevant variable is defined.

Definition of variables

Variable type	Defined in...
Global device user variables	Symbol browser
unit variable	Declaration table of the source file as VAR_GLOBAL, VAR_GLOBAL RETAIN or VAR_GLOBAL CONSTANT
Local variable	Declaration table of the program/chart as: • VAR, VAR_TEMP, or VAR CONSTANT • Additionally for function blocks as VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT • Additionally for functions as VAR_INPUT, VAR_IN_OUT
I/O variable	Symbol browser
Symbolic access to the fixed process image of the BackgroundTask	• Declaration table of the source file • Declaration table of the program/chart (programs and FB only)

Use of global device variables

Global device variables are user-defined variables that you can access from all program sources (e.g. ST source files, MCC source files) of a SIMOTION device.

Global device variables are created in the symbol browser tab of the detail view; to do this, you must be working in offline mode.

Here is a brief overview of the procedure:

1. In the project navigator of SIMOTION SCOUT, select the **GLOBAL DEVICE VARIABLES** element in the SIMOTION device subtree.
2. In the detail view, select the **Symbol browser** tab and scroll down to the end of the variable table (empty row).
3. In the last (empty) row of the table, enter or select the following:
   - **Name** of variable
   - **Data type** of variable (only elementary data types are permitted)
4. Optionally, you can make the following entries:
   - Activation of **Retain** checkbox (This declares the variable as retentive, so that its value will be retained after a power failure.)
   - **Field length** (array size)
   - **Initial value** (if array, for each element)
   - **Display format** (if array, for each element)

You can now access this variable using the symbol browser or any program of the SIMOTION device.

In ST source files, you can use a global device variable, just like any other variable.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the global device variable with *_device.var-name*.

An alternative to global device variables is the declaration of unit variables in a separate unit, which is imported into other units. This has the following advantages:

1. Variable structures can be used.
2. The initialization of the variables during the STOP-RUN transition is possible (via Program in StartupTask).
3. For newly created global unit variables, a download in RUN is also possible.

Please refer to the SIMOTION Basic Functions Function Manual.

Declaring a unit variable in the source file

The unit variable is declared in the unit. The scope of validity of the variable is dependent on the section of the declaration table in which the variable is declared:

- In the interface section of the declaration table (INTERFACE):
  The unit variable is valid for the entire unit; all programs/charts (programs, function blocks, and functions) within the unit can access the unit variable. It is exported and can be used in other source files or units (e.g. ST source files, MCC units, LAD/FBD units) after a connection has been defined (Page 4115). It is also available on HMI devices as standard. The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.
- In the implementation section of the declaration table (IMPLEMENTATION):
  The unit variable is valid in the unit only; all programs/charts (programs, function blocks, and functions) within the unit can access the unit variable.

If you insert pragma lines (Page 4070) into the declaration table, you can split the unit variables into data blocks with a separate version code (Page 4081). You can also use the pragma lines to define a separate initialization behavior (Page 4076) for each of these data blocks and change the defaults for HMI export (Page 4082).

Proceed as follows; the unit (declaration table) is open, see Open existing program source (Page 3989):

1. In the declaration table, select the section for the desired scope.
2. Then select the **Parameters** tab.
3. Enter:
   - Name of the variable.
   - Variable type.
     See also: Keywords for variable types (Page 4062).
   - Data type of the variables.
     You can select already defined data types.
     See also: Data types (Page 4053).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 4047).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 4048).
   - Optional initial value (initialization value).
     See also: Initial value (Page 4049).
   - Optional comment.
     See also: Comment (Page 4051).

The variable is then declared and can be used immediately within the unit.

Note

Outside the unit (e.g. in the symbol browser), the variable is only available after the unit has been compiled.

INTERFACE (exported declaration)

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	var_in1	VAR_GLOBAL	REAL	☐			
2				☐			

IMPLEMENTATION (source-internal declaration)

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	var_in2	VAR_GLOBAL	BOOL	☐			
2	var_out	VAR_GLOBAL	INT	☐	10		
3				☐			

Figure 7-42 Example: Declaring a unit variable in the unit

Note

The declaration table of the unit is read each time parameters are assigned for a command. Inconsistent data within the declaration table may, therefore, cause unexpected error messages during parameter assignment.

Declaring local variables

A local variable can only be accessed within the program/chart (program, function, function block) in which it is declared.

We distinguish between the following:

- Static variables:
  Static variables retain their value over all passes of the unit section (block memory).
- Temporary variables:
  Temporary variables are initialized each time the unit section is called again.

See also: Initialization of local variables (Page 4077).

If you insert a pragma line (Page 4070) at the start of the declaration table, you can influence the initialization of static variables of programs (Page 4078).

Proceed as follows; the program/chart with the declaration table is open (see Open existing program source (Page 3989)):

1. In the declaration table, select the **Parameters/Variables** tab.
2. Enter:
   - Name of the variable.
   - Variable type for variables.
     See also: Keywords for variable types (Page 4062).
   - Data type of the variables.
     You can select already defined data types.
     See also: Data types (Page 4053).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 4047).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 4048).
   - Optional initial value (initialization value).
     See also: Initial value (Page 4049).
   - Optional comment.
     See also: Comment (Page 4051).

The variable is now declared and can be used immediately.

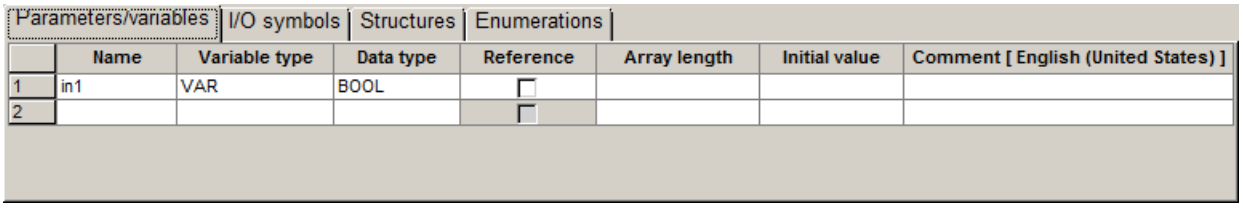


Figure 7-43 Example: Declaring a local variable in the chart/program

Note

The declaration table of the program/chart is read each time parameters are assigned for a command. Inconsistent data within the declaration table may, therefore, cause unexpected error messages during parameter assignment.

Defining variables in the Variable declaration dialog box ("on-the-fly" variable declaration)

As soon as you enter an unknown variable in the parameter screen form for an MCC command or an LAD/FBD graphic, the **Variable declaration** dialog box appears.

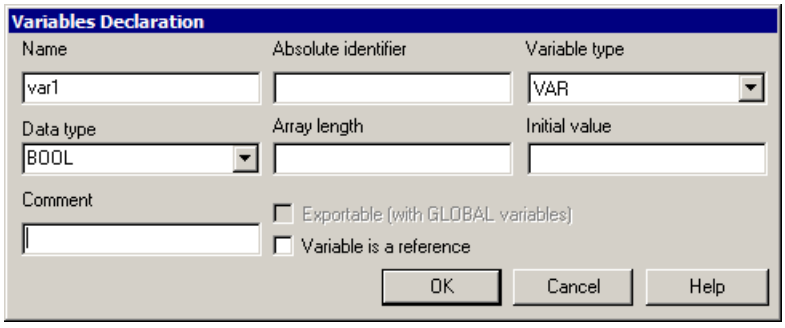


Figure 7-44 Variable declaration dialog box

Note

In order for the **Variable declaration** dialog box to appear, the **on-the-fly variable declaration** checkbox must be activated in the **Settings** dialog box.

Procedure

To define variables "on-the-fly" in the **Variable declaration** dialog box, proceed as follows:

1. Enter only the name of the variables in an input field of the parameter screen for an MCC command or LAD/FBD graphic and press the **Return** key.
   If the entered identifier is not a valid variable name, the dialog box **Variable declaration** appears.
2. Enter:
   - Another variable name, if required.
   - The **Data type** of the variables
     Available for selection are all elementary data types (Page 4054) and, where appropriate, the data type suitable for the input field.
     Select the data type or enter the identifier of a data type.
   - The **variable type**.
     Available for selection are all permitted keywords for variable types (Page 4062).
     Select the variable type.
     If you select a global variable type (e.g. VAR_GLOBAL), the checkbox **Exportable** becomes active.
   - The **Absolute identifier** for access to the fixed process image of the background task (Page 4103).
     You can only enter the identifier for the absolute access to the fixed process image of the background task if you have selected a data type that is included in the general data types (Page 4056) ANY_BIT or ANY_INT.
     Enter the absolute identifier according to the Syntax (Page 4110).
     The variable is displayed in the register **I/O symbols** of the declaration table.
   - Optional **array length**.
     Here, you specify the size of the array.
     Not available if an absolute identifier is entered.
     See also: Array length and array element (Page 4048).
   - Optional **initial value** (initialization value).
     Not available if an absolute identifier is entered.
     See also: Initial value (Page 4049).
   - Optional **comment**.
     See also:Comment (Page 4051).
   - Optionally, activate the **Variable is a reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 4047).

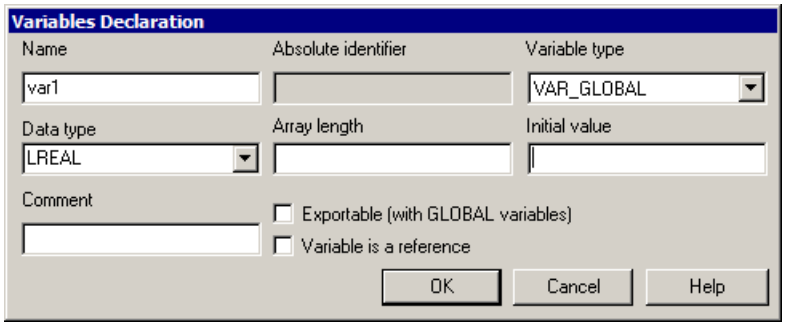


Figure 7-45 Example: Variable declaration

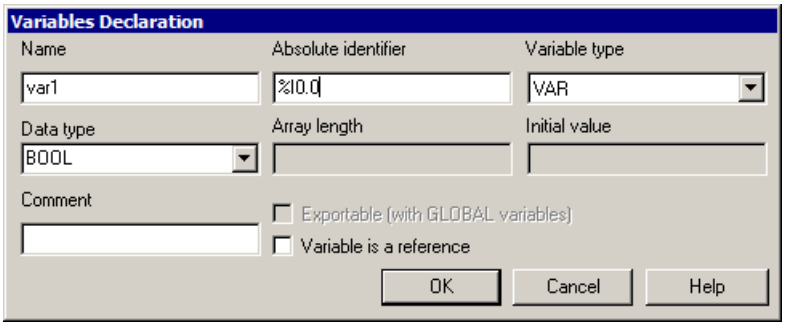


Figure 7-46 Example: Variable declaration (absolute name)

3. Confirm with **OK**.

Result

The variable is defined and entered in the declaration table of the source or the MCC chart or the LAD/FBD program, depending on the selected variable type and the **Exportable checkbox:**

- With local variables (e.g. VAR), the **Exportable** checkbox has a gray background and the new variable is entered in the declaration table of the MCC chart or the LAD/FBD program.
- With global variables (e.g. VAR_GLOBAL), the **Exportable** checkbox is shown activated:
  - If the **Exportable** checkbox is activated, the new variable is entered in the implementation section of the unit's declaration table.
  - If the **Exportable** checkbox is not activated, the new variable is entered in the interface section of the unit's declaration table.

Note

If you leave the **Variable declaration** dialog box by clicking **Cancel**, your input remains as it is, and the variable is not created.

Pasting pragma lines during variable definition

Pragma lines in declaration tables have the following function:

- In declaration tables of units (declaration of unit variables):
  Pragma lines in the interface section or implementation section of the declaration table split the variables of the respective section into different subsections:
  - The 1st subsection begins at the start of the relevant table and finishes at the 1st Pragma line.
  - All the other subsections start at one pragma line and finish at the next (or at the end of the table if starting at the last pragma line).

  Within these subsections of the table, each of the variables declared with VAR_GLOBAL or VAR_GLOBAL RETAIN forms a data block with a separate version code (Page 4081). If the version code is changed, only the data block concerned will be initialized during a download. Configure the pragma line in order to change the initialization behavior or the HMI export of the next data block (see following Section *Pragmas in the declaration tables of the unit variables*).
- In the declaration table of a program/chart (declaration of local variables):
  Line 1 may only contain one pragma line for changing the initialization behavior of programs' static variables (see the section titled *Pragmas in the declaration tables for local variables* later in this document).

Note

The SIMOTION Kernel version partially determines the effectiveness of pragma lines. This is specified for individual parameters.

Inserting a pragma line

To insert a pragma line into the declaration table, proceed as follows:

1. Select the **Parameters** tab (for unit variables) or **Parameters/variables** tab (for local variables).
2. In the declaration table, set the cursor to the number of the line at which a new data block is to be started.
   Only line 1 can be used in the declaration table for local variables.
3. Select the **Insert pragma line** context menu.
   A new line containing the **Pragmas** button only is inserted above the line.
4. Left-click the **Pragmas** button.
   A window containing configuration options for the relevant declaration table opens.
5. Enter the settings.
6. Confirm with **OK**.

Pragmas in the declaration tables for unit variables

A pragma line has been inserted into the declaration table for unit variables. If you click the "Pragmas" button in the table, the following window appears:

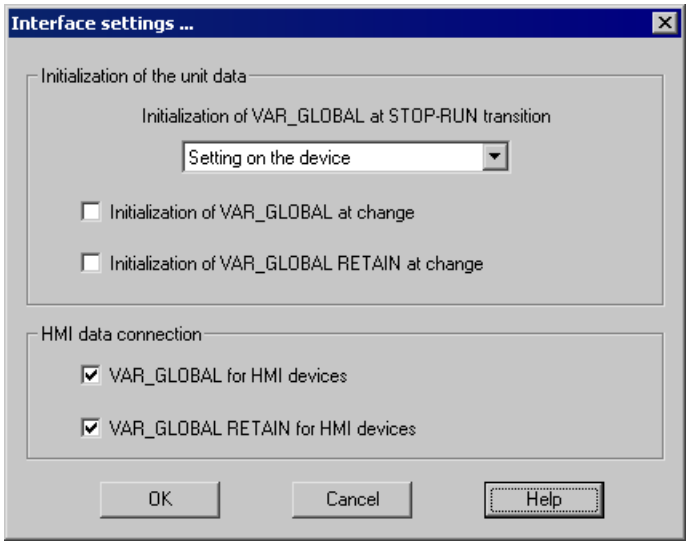


Figure 7-47 Pragma settings in the declaration table for unit variables (e.g. interface section)

This window is for making the settings below. These influence the data block following the pragma line:

Table 7-16 Parameters for pragma settings in the declaration table for unit variables

Parameter		Description
Initialization of unit data		
	Initialization of VAR_GLOBAL during STOP-RUN transition	Only as of version V4.1 of the SIMOTION Kernel. This setting determines whether the next data block of the non-retentive unit variables is initialized during the transition from STOP to RUN operating state. **Setting on the device** (standard): • As of version V4.2 of the SIMOTION Kernel: The setting made on the SIMOTION device applies. • Up to version V4.1 of the SIMOTION Kernel: Variables are **not** initialized. **Always**: Variables are initialized. **Never**: Variables are **not** initialized.
	Initialization of VAR_GLOBAL during a change	This setting determines whether a download for the next data block of the non-retentive unit variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
	Initialization of VAR_GLOBAL RETAIN during a change	This setting determines whether a download for the next data block of the retentive unit variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
HMI data connection		

Parameter		Description
		Use the settings below to change which unit variables are available on which HMI devices by default. Detailed description of the HMI export, in particular the effect of the settings depending on SIMOTION Kernel version: see Variables and HMI devices (Page 4082).
	VAR_GLOBAL for HMI devices	**Active** (standard in the interface section): The next data block of the non-retentive unit variables is available on HMI devices. **Inactive** (standard in the implementation section): The next data block of the non-retentive unit variables is **not** available on HMI devices.
	VAR_GLOBAL RETAIN for HMI devices	**Active** (standard in the interface section): The next data block of the retentive unit variables is available on HMI devices. **Inactive** (standard in the implementation section): The next data block of the retentive unit variables is **not** available on HMI devices.

Pragmas in the declaration tables for local variables

A pragma line has been inserted into line 1 of the declaration table for local variables. If you click the "Pragmas" button in the table, the following window appears:

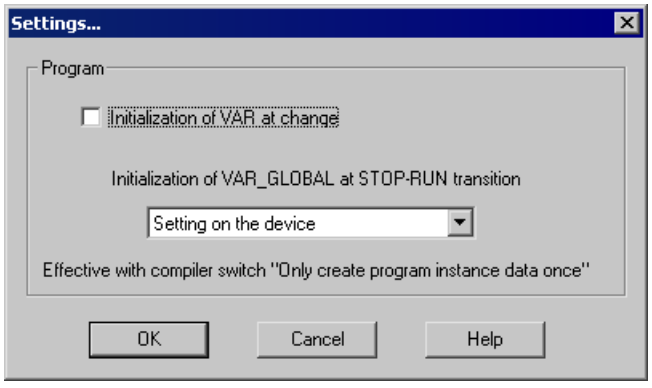


Figure 7-48 Pragma settings in the declaration table for local variables

You can make the following settings in this window:

Note

These settings only take effect in the following scenarios:

1. The "Only create program instance data once" compiler option is active on the higher-level unit.
2. The creation type of the program/chart is "Program".

Table 7-17 Parameters for pragma settings in the declaration table for local variables

Parameter	Description
Initialization of VAR during a change	This setting determines whether a download for the next static variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
Initialization of VAR during STOP-RUN transition	Only as of version V4.1 of the SIMOTION Kernel. This setting determines whether the next static variables are initialized during the transition from STOP to RUN operating state. **Setting on the device** (standard): • As of version V4.2 of the SIMOTION Kernel: The setting made on the SIMOTION device applies. • Up to version V4.1 of the SIMOTION Kernel: Variables are **not** initialized. **Always**: Variables are initialized. **Never**: Variables are **not** initialized.

Time of the variable initialization

The timing of the variable initialization is determined by:

- Memory area to which the variable is assigned
- Operator actions (e.g. source file download to the target system)
- Execution behavior of the task (sequential, cyclic) to which the program was assigned.

All variable types and the timing of their variable initialization are shown in the following tables. You will find basic information about tasks in the *SIMOTION Basic Functions* Function Manual.

The behavior for variable initialization during download can be set: To do this, as a default setting select the **Options > Settings** menu and the **Download** tab or define the setting during the current download.

Note

You can upload values of unit variables or global device variables from the SIMOTION device into SIMOTION SCOUT and save them in XML format.

1. Save the required data segments of the unit variables or global device variables as a data set with the function *_saveUnitDataSet*.
2. Use the **Save variables** function in SIMOTION SCOUT.

You can use the **Restore variables** function to download these data sets and variables back to the SIMOTION device.

For more information, refer to the SIMOTION SCOUT Configuration Manual.

This makes it possible, for example, to obtain this data, even if it is initialized by a project download or if it becomes unusable (e.g. due to a version change of SIMOTION SCOUT).

See also

Initialization of retentive global variables (Page 4074)
Initialization of non-retentive global variables (Page 4076)
Initialization of local variables (Page 4077)
Initialization of static program variables (Page 4078)
Initialization of instances of function blocks (FBs) or classes (Page 4079)
Initialization of system variables of technology objects (Page 4080)
Version ID of global variables and their initialization during download (Page 4081)

Initialization of retentive global variables

Retentive variables retain their last value after a loss of power. All other data is reinitialized when the device is switched on again.

Retentive global variables are initialized:

- When the backup or buffer for retentive data fails.
- When a memory reset (MRES) is performed.
- With the restart function (Del. SRAM) in SIMOTION P320 or P350.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the retentive data.
- When a download is performed according to the following description.
- With a firmware update (upgrade) or activation of a kernel in accordance with the following description.

Behavior during download

Table 7-18 Initializing retentive global variables during download

Variable type	Time of the variable initialization
Retentive global device variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive global device variables are initialized. • **No**[3]: The retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 4081).
Retentive unit variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 4081).

[1] Default setting in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the retentive unit variables corresponds to a VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** and **SIMOTION LAD/FBD** programming languages:
A data block of the retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines (Page 4070) within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute has been specified within a pragma: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 4070) is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL RETAIN during a change*. All the variables declared with VAR_GLOBAL RETAIN up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see the SIMOTION Basic Functions Function Manual.

Behavior during upgrade or configuration change

When the SIMOTION device is upgraded to a new version of the SIMOTION Kernel or if the configuration is changed, the retentive variables are initialized as described below:

Table 7-19 Initialization of retentive global variables during upgrade or configuration change

Variable type	Time of the variable initialization
Retentive global device variables	This data is always initialized.
Retentive unit variables	This data can be retained. See section "Retaining retentive data" in the "Basic Functions for Modular Machines" Function Manual.

Initialization of non-retentive global variables

Non-retentive global variables lose their value during power outages. They are initialized:

- During the initialization of retentive global variables (Page 4074), e.g. during a firmware update or overall reset (MRES).
- During switch-on.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the non-retentive data.
- During a download as described in the following table.
- During the transition from STOP to RUN mode as described at the end of the section.

Behavior during download

Table 7-20 Initializing non-retentive global variables during download

Variable type	Time of the variable initialization
Non-retentive global device variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive global device variables are initialized. • **No**[3]: The non-retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 4081).
Non-retentive unit variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the non-retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 4081).

[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the non-retentive unit variables corresponds to a VAR_GLOBAL/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL in the interface section or implementation section of the declaration table: Pragma lines (Page 4070) within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN, provided the following condition is fulfilled:
With the **SIMOTION ST**
programming language: The following attribute has been specified within a pragma in the relevant declaration block: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 4070) has been pasted into the declaration table and the following check box is activated: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Behavior during STOP-RUN transition

The values of non-retentive global variables are retained by default during the transition from STOP to RUN mode.

You can, however, make a setting whereby the non-retentive global variables are initialized during the STOP-RUN transition:

- **As of Version V4.2 of the SIMOTION Kernel**, by activating the **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
  With non-retentive unit variables, this setting can be overwritten by a pragma or pragma line (Page 4070) in the relevant data blocks of the program sources.
- **As of Version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the relevant data blocks of the program sources (only with non-retentive unit variables):
  - With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the relevant VAR_GLOBAL/END_VAR declaration block: { BlockInit_OnDeviceRun := ALWAYS; }
  - With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    Paste a pragma line (Page 4070) with the following setting into the declaration table: "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block which is initialized during the STOP-RUN transition.

Note

With SIMOTION devices up to SIMOTION Kernel Version V4.0, non-retentive global variables are never initialized during the STOP-RUN transition.

Initialization of local variables

Local variables are initialized:

- For the initialization of retentive unit variables (Page 4074).
- For the initialization of non-retentive unit variables (Page 4076).
- Also, according to the following description:

Table 7-21 Initialization of local variables

Variable type	Time of the variable initialization
Local program variables	Local variables of programs are initialized differently: • Static variables (VAR) are initialized according to the memory area in which they are stored. See: Initialization of static program variables (Page 4078). • Temporary variables (VAR_TEMP) are initialized every time the program of the task is called.
Local variables of function blocks (FB)	Local variables of function blocks are initialized differently: • Static variables (VAR, VAR_IN, VAR_OUT) are only initialized when the FB instance is initialized. See: Initialization of instances of function blocks (FBs) (Page 4079). • Temporary variables (VAR_TEMP) are initialized every time the FB instance is called.
Local variables of functions (FC)	Local variables of functions are temporary and are initialized every time the function is called.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4148) function.

Initialization of static program variables

The following versions affect the following static variables:

- Local variables of a unit program declared with VAR
- Function block instances declared with VAR within a unit program, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT).

The initialization behavior is determined by the memory area in which the static variables are stored. This is determined by the "Only create program instance data once" (Page 3996) compiler option.

- For the deactivated "Only create program instance data once" compiler option (default):
  The static variables are stored in the user memory of each task which is assigned to the program.
  The initialization of the variables thus depends on the execution behavior of the task to which the program is assigned (see SIMOTION Basic Functions Function Manual):
  - Sequential tasks (MotionTasks, UserInterruptTasks, SystemInterruptTasks, StartupTask, ShutdownTask): The static variables are initialized every time the task is started.
  - Cyclic tasks (BackgroundTask, SynchronousTasks, TimerInterruptTasks): The static variables are only initialized only during the transition from STOP to RUN operating state.
- For the activated "Only create program instance data once" compiler option:
  This setting is necessary, for example, if a program is to be called within a program.
  The static variables of all programs from the program source (unit) involved are only stored once in the user memory of the unit.
  They are thus initialized together with the non-retentive unit variables, see Initialization of non-retentive global variables (Page 4076).
  They are not initialized by default during the transition from STOP to RUN operating state.
  You can, however, make a setting whereby they are initialized during the STOP-RUN transition:
  - **As of version V4.2 of the SIMOTION Kernel**, by activating the **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
    This setting can be overwritten by a pragma or pragma line (Page 4070) in the data block of the relevant program organization unit (POU).
  - **As of version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the data block of the relevant program organization unit (POU):
    With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the VAR/END_VAR declaration block:
    { BlockInit_OnDeviceRun := ALWAYS; }
    With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    The declaration table starts with a pragma line (Page 4070) containing the following setting: "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR in the table are initialized during the STOP-RUN transition.

Initialization of instances of function blocks (FBs) or classes

The initialization of a function block instance (Page 4133) or a class instance (as of version V4.5 of the SIMOTION Kernel) (Page 4140) is determined by the location of its declaration:

- Global declaration (within VAR_GLOBAL/END_VAR in the interface of implementation section):
  Initialization as for a non-retentive unit variable, see Initialization of non-retentive global variables (Page 4076).
- Local declaration in a program (within VAR / END_VAR):
  Initialization as for static variables of programs, see Initialization of static variables of programs (Page 4078).

- Local declaration in a function block (within VAR / END_VAR):
  Initialization as for an instance of this function block.
- Declaration as in/out parameter in a function block or a function (within VAR_IN_OUT / END_VAR):
  For the initialization of the POU, only the reference (pointer) will be initialized with the instance of the function block remaining unchanged.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4148) function.

Initialization of system variables of technology objects

The system variables of a technology object are usually not retentive. Depending on the technology object, a few system variables are stored in the retentive memory area (e.g. absolute encoder calibration).

The initialization behavior (except in the case of download) is the same as for retentive and non-retentive global variables. See Initialization of retentive global variables (Page 4074) and Initialization of non-retentive global variables (Page 4076).

The behavior during the download is shown below for:

- Non-retentive system variables
- Retentive system variables

Table 7-22 Initializing technology object system variables during download

Variable type	Time of the variable initialization
Non-retentive system variables	Behavior during download, depending on the *Initialization of all non-retentive data for technology objects* setting[1]: • **Yes**[2]: All technology objects are initialized. – All technology objects are restructured and all non-retentive system variables are initialized. – All technological alarms are cleared. • **No**[3]: Only technology objects changed in SIMOTION SCOUT are initialized. – The technology objects in question are restructured and all non-retentive system variables are initialized. – All alarms that are pending on the relevant technology objects are cleared. – If an alarm that can only be acknowledged with *Power On* is pending on a technology object that will not be initialized, the download is aborted.
Retentive system variables	Only if a technology object was changed in SIMOTION SCOUT, will its retentive system variables be initialized. The retentive system variables of all other technology objects are retained (e.g. absolute encoder calibration).

[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding checkbox is active.

[3] The corresponding checkbox is inactive.

Version ID of global variables and their initialization during download

Table 7-23 Version code of global variables and their initialization during download

Data segment		Description of version code
Global device variables		
	Retentive global device variables Non-retentive global device variables	• Separate version code for each data segment of the global device variables. • The version code of the data segment changes for: – Add or remove a variable within the data segment – Change of the identifier or the data type of a variable within the data segment • This version code does not change on: – Changes in the other data segment – Changes to initialization values[1] • During downloading[2], the rule is: This data segment is only initialized when the version code of a data segment is changed.
Unit variables of a unit		
	Retentive unit variables in the interface section Retentive unit variables in the implementation section Non-retentive unit variables in the interface section Non-retentive unit variables in the implementation section	• Several data blocks (= declaration blocks)[3] are possible in each data segment. • **Separate version code** for each data block. • The version code of the data block changes for: – Add or remove a variable in the associated declaration block – Change of variable sequence in the relevant declaration block – Change of the identifier or the data type of a variable in the associated declaration block – Change of a data type definition (from a separate or imported[4] unit) used in the associated declaration block – Add or remove declaration blocks within the same data segment **before** the associated declaration block • This version code does not change on: – Add or remove declaration blocks in other data segments – Add or remove declaration blocks within the same data segment **after** the associated declaration block – Changes in other data blocks – Changes to initialization values[1] – Changes to data type definitions that are not used in the associated data block – Changes to functions • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
Retentive variables of function blocks and classes (The instances are declared as non-retentive unit variables)		

Data segment		Description of version code
	Non-retentive unit variables in the interface section	Only as of version V4.5 of the SIMOTION Kernel. • The retentive variables of all instances within a declaration block (VAR_GLOBAL / END_VAR) are summarized as a separate retentive data block • to which a separate version code is assigned. • The version code of this retentive data block changes: – The data structure of the retentive local variables for the instances within the declaration block changes. – The sequence of the declaration blocks (VAR_GLOBAL / END_VAR) within the program source changes. • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
	Non-retentive unit variables in the implementation section	

[1] Changed initialization values are not effective until the data block or data segment in question is initialized.

[2] If *Initialization of retentive program data and retentive global device variables* = No and *Initialization of non-retentive program data and non-retentive global device variables* = No.
In the case of other settings: See the sections "Initialization of retentive global variables (Page 4074)" and "Initialization of non-retentive global variables (Page 4076)".

[3] With the **SIMOTION ST** programming language:
A data block corresponds to a VAR_GLOBAL/END_VAR or VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL or VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines (Page 4070) within a section of the declaration table separate the variables into different data blocks.

[4] The use of units depends on the programming language, refer to the relevant section (Page 4115).

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided that the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute has been specified within a pragma: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 4070) is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Variables and HMI devices

Exported variables

The following variables are exported to HMI devices where they are available:

- System variables of the SIMOTION device
- System variables of technology objects
- I/O variables
- Global device variables

- Retentive and non-retentive unit variables of the interface section (default setting).
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line (Page 4070), if you deactivate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
- Non-retentive static variables (VAR) of programs provided that the compiler option "Only create program instance data once" is activated
- Non-retentive static variables (VAR) of function blocks
  This is the default setting when the compiler option "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Non-retentive public variables (VAR PUBLIC) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.

Note

The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.

The effect of the `{ HMI_Export := FALSE; }` / `{ HMI_Export := TRUE; }` pragma (or the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* settings in a pragma line) depends on the SIMOTION Kernel version:

- As of Version V4.1 of the SIMOTION Kernel:
  The pragma affects the export of the corresponding declaration block to HMI devices **and** the structure of the HMI address space:
  - Only those variables in declaration blocks exported to HMI devices occupy the HMI address space.
  - Within the HMI address space, the variables are arranged according to order of their declaration.
- Up to Version V4.0 of the SIMOTION Kernel:
  The pragma affects **only** the export of the corresponding declaration block to HMI devices. The HMI address space is also occupied by unit variables of the interface section whose declaration blocks are not assigned to HMI devices.
  Within the HMI address space, the variables are sorted in the following order:
  - Retentive unit variables of the interface section (exported and not exported).
  - Retentive unit variables of the implementation section (only exported).
  - Non-retentive unit variables of the interface section (exported and not exported).
  - Non-retentive unit variables of the implementation section (only exported).

  Within these segments, the variables are arranged according to order of their declaration.

Non-exported variables

The following variables are **not** exported to HMI devices and are **not** available there:

- Retentive and non-retentive unit variables of the implementation section (default setting)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`
    See also Controlling compiler with attributes.
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line (Page 4070), if you activate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Local variables (VAR, VAR_TEMP) of functions or methods
- Temporary variables (VAR_TEMP) of programs, function blocks or classes
- Retentive local variables (VAR RETAIN) of programs (as of version 4.5 of the SIMOTION Kernel)

- Retentive local variables (VAR RETAIN) of function blocks (as of version 4.5 of the SIMOTION Kernel)
  This is the default setting when the compiler option "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Retentive local variables (VAR RETAIN) of classes (default setting, as of version 4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Non-retentive static variables (VAR) of programs provided that the compiler option "Only create program instance data once" is **not** activated
- Non-retentive protected or private variables (VAR, VAR PROTECTED, VAR PRIVATE) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.

Example for the SIMOTION ST programming language

Table 7-24 Example for the control of the HMI export with the corresponding pragma

```
INTERFACE
    VAR_GLOBAL
        // HMI export
        x1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := FALSE; }
        // No HMI export
        x2 : DINT;
    END_VAR
    // ...
END_INTERFACE

IMPLEMENTATION
    VAR_GLOBAL
        // No HMI export
        y1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := TRUE; }
        // HMI export
        y2 : DINT;
    END_VAR
    // ...
END_IMPLEMENTATION
```

7.1.5.7 General references (as of kernel V4.5)

Defining general references

General references contain the address assignment to a variable or an instance of a function block or class:

General references require a SIMOTION Kernel as of version V4.5. The compiler option (Page 3996) "Permit object-oriented programming" also must be activated.

References can be defined as variables or elements of structures.

To do this, activate the **Reference** checkbox in the **Parameters/variables** or **Structures** tab in the declaration table of the program source or program organization unit. The declaration is also possible as array element. See also the description for the "Reference (Page 4047)" checkbox.

This creates the "Reference to data type" (REF_TO *data_type*) that can contain the address to an existing data type (reference data type).

The following are permitted as data types to which references can be formed:

- Elementary data types (e.g. INT, DINT, REAL, WORD, TIME, STRING)
- User-defined data types (UDT)
- System data types

- Function blocks, provided they contain at least one static variable
- Classes (from ST sources)

No references can be formed to the following data types because references exist already:

- Technology object data types
- Object-oriented interfaces (from ST sources)
- General references
- I/O references (from ST sources)

Variables of general references can be declared in the interface and implementation sections as well as in all program organization units and classes. The data type to be referenced must be declared beforehand and lie within the scope of the POU. The declaration as element of a structure or an ARRAY is also possible.

All variable types for the relevant program organization unit are permitted with the exception of:

- VAR_IN_OUT.
  Arrays, structures, function blocks or classes that contain general references may nonetheless be transferred as VAR_IN_OUT.
- VAR RETAIN or VAR_GLOBAL RETAIN is not permitted.
  However, structures that contain at least one element other than reference types (general references, data types of the technology objects, object-oriented interfaces), can be stored retentive.
- VAR CONSTANT or VAR_GLOBAL CONSTANT is not meaningful.
  General references must be initialized with "NULL".

Forming general references

A reference to a variable is formed with the REF standard function (in := *var_name*). This function is described in detail in the "SIMOTION Basic Functions" Function Manual.

The REF function can be used on the following variables:

- Retentive and non-retentive unit variables or instances of classes or function blocks or their elements declared as such.
- In methods within classes or function blocks:
  - Static variables (VAR) of the higher-level class or function block (including instance variables for classes or function blocks).
  - The THIS keyword. Enables access to the instance variable of the higher-level class or function block.
- In function blocks:
  - Static variables (VAR) of the function block (including instance variables of classes or function blocks).
  - The THIS keyword. Permits access to the instance variable of the function block.

The variable must be able to be read and written.

Not permitted are:

- Variables of technology object data types
- Variables of object-oriented interfaces
- Variables of general references
- Retentive local variables within classes or function blocks
- I/O variables
- Global device variables
- Constants

The return value has the data type "Reference to the data type of the input parameter *in*". The return value can be assigned with the assignment operator ":=" to a variable that has REF_TO *type* data type. A data type conversion is not possible.

Note

Note for instances of function blocks that are passed as in/out parameter (VAR_IN_OUT):

- The REF function cannot be used for static variables (VAR) of function blocks imported from technology packages or device-independent libraries.

The restriction does not apply to class instances.

Operations with general references

Assignment operator ":="

For the assignment of a reference variable, the address of the variable rather than the variable value is passed. Consequently, the reference data types must be identical on both sides of the operator. No implicit data type conversion is possible.

The only exception is for references to classes. A reference to a derived class of a reference to the base class can be assigned here.

For the assignment, no check is made for the validity of the reference; the assignment is always possible.

An explicit assignment with NULL is also possible.

The validity of references should be checked by comparing with NULL before use.

Dereferencing with operator "^"

The reference content can be accessed with the postfix operator "^", e.g. *ref_var*^.

If the reference does not contain a valid value at runtime, the processing task is aborted and the ExecutionFaultTask called.

Before access, a check for validity by comparison with NULL is recommended.

Multilevel dereferencing is possible, e.g. reference to a structure that contains further references. Dereferenced variables can be used anywhere in expressions.

If a method is called via a reference to a class or function block, the class or function block must be dereferenced.

Dereferenced values cannot be used as actual parameters for the _releaseSemaphore() and _testandSetSemaphore() system functions.

Dynamic type conversion with the operator "?="

The operator "?=" can be used for dynamic type conversion only with references to classes and with interface variables (variables for object-oriented interfaces). It supports the following variants:

- Assignment between references to classes.
- Assignment of a reference to a class for an interface variable.
  Whereby, the reference to a class must be specified with the dereferencing operator "^".
- Assignment of an interface variable for a reference to a class.
- Assignment between interface variables

A dynamic type conversion is performed if the following two conditions are satisfied:

1. The variable to the right of the operator contains the reference to the instance of a class.
2. For the variable to the left of the operator, the following applies:
   - The variable is an interface variable:
     The object-oriented interface that this variable has as data type must implement the class on the right-hand side.
   - The variable is a reference to a class:
     This class corresponds to the class on the right-hand side or is a base class for this class.

In this case, the reference on the right-hand side is converted and transferred to the left-hand side.

If the type conversion fails, the variable on the left-hand side contains the "NULL" value.

Comparison operators

Only the comparison operators "=" and "<>" can be used to check for equality or inequality.

The operators ">", "<", ">=", "<=" as well as the MIN and MAX standard functions are not permitted.

7.1.5.8 Access to inputs and outputs (process image, I/O variables)

Overview of access to inputs and outputs

SIMOTION provides several possibilities to access the device inputs and outputs of the SIMOTION device as well as the central and distributed I/O:

- Via direct access with I/O variables
  Direct access is used to access the corresponding I/O address directly.
  Define an I/O variable (name and I/O address) without assigning a task to it. The entire address space of the SIMOTION device can be used.
  It is preferable to use direct access with sequential programming (in MotionTasks); access to current input and output values at a particular point in time is especially important in this case.
  Further information: Direct access and process image of the cyclic tasks (Page 4094).
- Via the process image of cyclic tasks using I/O variables
  The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming).
  Define an I/O variable (name and I/O address) and assign a task to it. The entire address range of the SIMOTION device can be used.
  Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.
  Further information: Direct access and process image of the cyclic tasks (Page 4094).
- Using the fixed process image of the BackgroundTask
  The process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device, on which a subset of the I/O address space of the SIMOTION device is mirrored. The mirror image is refreshed with the BackgroundTask and is consistent throughout the entire cycle. This process image is used preferentially when programming the BackgroundTask (cyclic programming).
  The address range 0 .. 63 can be used. Exception: I/O addresses that are accessed using the process image of the cyclic task can only be used with the setting “Common process image” (Page 4105) (as of Kernel V4.2).
  Further information: Access to the fixed process image of the BackgroundTask (Page 4103).

A comparison of the most important properties is contained in "Important properties of direct access and process image" (Page 4091).

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4113).

Note

An access via the process image is more efficient than direct access.

Important features of direct access and process image access

Table 7-25 Important properties of direct access and process image access

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Permissible address range	Entire address range of the SIMOTION device **Exception**: I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously (example: PIW63 or PQD62 are not permitted).		Addresses 0 .. 63. **Exception**: Up to version V4.1 of the SIMOTION Kernel or the "Separate process image" setting, addresses used for the process image of the cyclic tasks are not permitted.
Address configuration	Necessary. The addresses used must be present in the I/O and appropriately configured. The "Rules for I/O addresses for direct access and the process image of the cyclic tasks" (Page 4097) must be observed.		Not necessary. Addresses that are not present in the I/O or have not been configured can also be used.
Assigned task	None.	Cyclic task for selection: • SynchronousTasks, • TimerInterruptTasks, • BackgroundTask.	BackgroundTask.
Memory area for process images	-	Depends on the SIMOTON Kernel version: • Up to version V4.1: Separate memory areas in all cases • As of version V4.2: Option to select a common memory area	

<table>
<tr><th></th><th>Direct access</th><th>Access to process image of cyclic tasks</th><th>Access to fixed process image of the BackgroundTask</th></tr>
<tr><td>Update</td><td>• Onboard I/O of SIMOTION devices C230-2, C240, and C240 PN:
Update occurs in a cycle clock of 125 µs.
• With I/O devices
– via PROFIBUS DP or PROFINET on an isochronous SIMOTION device[1]
– via DRIVE-CLiQ
– Onboard I/O of SIMOTION D devices:
The update is performed in the position control cycle clock[2].
• With I/O devices
– via PROFIBUS DP or PROFINET on a non-isochronous SIMOTION device[1]
– via P-Bus:
The update is performed in the interpolation cycle[2] [3].
Inputs are read at the start of the cycle clock.
Outputs are written at the end of the cycle clock.</td><td>Update occurs with the assigned task:
• Inputs are read before the assigned task is started and transferred to the process input image.
• Process output image is written to the outputs after the assigned task has been completed.</td><td>An update is made with the BackgroundTask:
• Inputs are read before the BackgroundTask is started and is transferred to the process input image.
• Process output image is written to the outputs when the BackgroundTask is complete.</td></tr>
<tr><td rowspan="2">Consistency</td><td>–</td><td>During the entire cycle of the assigned task.
Exception: Direct access to output occurs.</td><td rowspan="2">During the entire cycle of the BackgroundTask.
Exception: Direct access to output occurs.</td></tr>
<tr><td colspan="2">Consistency is only ensured for elementary data types.
When using arrays, the user is responsible for ensuring data consistency.</td></tr>
<tr><td>Use</td><td>Preferred in MotionTasks</td><td>Preferred in the assigned task</td><td>Preferred in the BackgroundTask</td></tr>
<tr><td>Declaration as variable</td><td colspan="2">Necessary, for the entire device as an I/O variable in the symbol browser.
Each byte of the address range may only be assigned to a single I/O variable.
Syntax of I/O address: e.g. PIW1022, PQ63.3.</td><td>Possible, but not necessary:
• For the entire device as I/O variable in the symbol browser
• As unit variable
• As local static variable in a program</td></tr>
<tr><td>Download of new or changed I/O variables</td><td colspan="2">Only possible in STOP mode.</td><td>-</td></tr>
<tr><td>Use the absolute address</td><td colspan="2">Not supported.</td><td>Possible, with the following syntax: E.g. %IW62, %Q63.3.</td></tr>
</table>

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Byte order when forming the process image	-	As supplied by the I/O	Depends on the SIMOTION Kernel version and the memory area setting for the process images: • Up to version V4.1 or the "Separate process image" setting: Always Big Endian • As of version V4.2 and the "Common process image" setting: As supplied by the I/O
Byte order during access	Depends on I/O		Always Big Endian
Writeability of inputs	No	Depends on the SIMOTION Kernel version: • Up to version V4.1: No • As of version V4.2: Yes	Yes
Write protection for outputs	Possible; **Read only** status can be selected.	Not supported.	Not supported.
Declaration of arrays	Possible.		Not supported.
Further information	Direct access and process image of the cyclic tasks (Page 4094).		Access to the fixed process image of the BackgroundTask (Page 4103).
Responses in the event of an error	Error during access from user program, alternative reactions available: • CPU Stop[4] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, alternative reactions available: • CPU stop[5] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, reaction: CPU stop[5] **Exception**: If a direct access has been created at the same address, the behavior set there applies.
Access			
• In the RUN operating state	Without any restrictions.	Without any restrictions.	Without any restrictions.
• During the StartupTask	Possible with restrictions: • Inputs can be read. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
• During the ShutdownTask	Without any restrictions.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.

[1] A SIMOTION device is considered isochronous if at least one PROFIBUS DP or PROFINET interface is operated isochronously (constant bus cycle time). For SIMOTION D there is an exception with the DP Integrated interface to which the SINAMICS Integrated is connected.

[2]The following SIMOTION devices are updated in the Servo_fast cycle or IPO_fast cycle, if the cycles are configured: D445-2 DP/PN, D455-2 DP/PN (as of version V4.2) and D435-2 DP/PN (as of version V4.3).

3 IPO or IPO_2 adjustable, see “Setting system cycle clocks" section in the Basic Functions Function Manual.

[4] Call the ExecutionFaultTask.

[5] Call the PeripheralFaultTask.

Direct access and process image of cyclic tasks

Property

Direct access to inputs and outputs and access to the process image of the cyclic task always take place via I/O variables. The entire address range of the SIMOTION device (Page 4096) can be used.

A comparison of the most important properties, including in comparison to the fixed process image of the BackgroundTask (Page 4103) is contained in "Important properties of direct access and process image" (Page 4091).

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4097).

It is particularly important that every address used in an I/O variable is available in the I/O and configured; each byte in the address range may be assigned to no more than one I/O variable (does not apply to access with data type BOOL).

A detailed status of I/O variables (Page 4101) can be read as of version V4.2 of the SIMOTION Kernel, for example, in order to check the availability of the I/O variables.

Direct access

Direct access is used to access the corresponding I/O address directly. Direct access is used primarily for sequential programming (in MotionTasks). The access to the current value of the inputs and outputs at a specific time is particularly important.

For direct access, you define an I/O variable (Page 4098) without assigning it a task.

Process image of the cyclic task

The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming). The consistency during the complete cycle of the task is particularly important.

For the process image of the cyclical task you define an I/O variable (Page 4098) and assign it a task.

Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.

Note

An access via the process image is more efficient than direct access.

Additional properties as of version V4.2 of the SIMOTION Kernel

As of version V4.2 of the SIMOTION Kernel, direct access to inputs/outputs and the process image of the cyclic tasks offers additional properties:

- As far as the process image of the cyclic tasks is concerned, a common memory area with the fixed process image of the BackgroundTask can be set (standard with newly created devices)
- As far as the process image of the cyclic tasks is concerned, I/O variables for inputs can be written to (i.e. they can be assigned values).
- A detailed status of I/O variables (Page 4101) can be read, for example, in order to check the availability of the I/O variables.

Memory area with the fixed process image of the BackgroundTask

- **As of version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks.
- **Up to version V4.1 of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas.

Note

If (and only if) you are also using the fixed process image of the BackgroundTask, it is important to consider the effects of the "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask (Page 4103).

Table 7-26 Effect of "Common process image" or "Separate process image" settings on the process image of the cyclic tasks

	Common process image	Separate process image
Availability	Only available as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Up to version V4.1 of the SIMOTION Kernel applies: • System characteristic, not configurable. The following applies as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Download of new or changed I/O variables	Only possible in STOP mode.	Only possible in STOP mode.
Byte order when forming the process image and during access	Depends on connected I/O	
Effects on the fixed process image of the BackgroundTask	See the relevant table in "Access to the fixed process image of the BackgroundTask" (Page 4104).	
Further information	Common process image (Page 4105)	Separate process image (Page 4107)

Address range of the SIMOTION devices

The address range of the SIMOTION devices is specified in the following table according to the version of the SIMOTION Kernel concerned. The complete address range can be used for direct access and process image of the cyclical tasks.

Table 7-27 Address range of the SIMOTION devices according to the version of the SIMOTION Kernel

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
C230-2	0 .. 2047 [3]	0 .. 2047 [3]	0 .. 2047 [3]	–	–	–
C240	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]
C240 PN [1]	–	–	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]
D410 DP	–	–	0 .. 8191 [3]	0 .. 8191 [3]	0 .. 8191 [3]	–
D410 PN	–	–	0 .. 8191 [4]	0 .. 8191 [4]	0 .. 8191 [4]	–
D410-2	–	–	–	–	0 .. 8191 [3 4]	0 .. 8191 [3 4]
D425	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D425-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D435	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D435-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D445	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–	–
D445-1 [1]	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D445-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D455-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
P320 [2]	–	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
P320-4	–	–	–	–	–	0 .. 4095 [3]
P350	0 .. 2047 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

[1] Available as of V4.1 SP2 HF4

[2] Available as of V4.1 SP5

[3] For distributed I/O (over PROFIBUS DP), the transmission volume is restricted to 1024 bytes per PROFIBUS DP line.

[4] For distributed I/O (over PROFINET), the transmission volume is restricted to 4,096 bytes per PROFINET segment.

Rules for I/O addresses for direct access and the process image of the cyclical tasks

Note

You must observe the following rules for the I/O variable addresses for direct access and the process image of the cyclic task (Page 4094). Compliance with the rules is checked during the consistency check of the SIMOTION project (e.g. during the download).

1. Addresses used for I/O variables must be present in the I/O and configured appropriately in the HW Config.
2. I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously.
   The following I/O addresses are not permitted:
   - Inputs: PIW63, PID61, PID62, PID63
   - Outputs: PQW63, PQD61, PQD62, PQD63
3. All addresses of an I/O variable comprising more than one byte (e.g. WORD, ARRAY data type) must be within a continuous address range configured in HW Config, e.g. within the address range (slot or subslot) of *one* I/O module.
4. An I/O address (input or output) can only be used by a single I/O variable of data type BYTE, WORD or DWORD or an array of these data types. Access to individual bits with I/O variables of data type BOOL is possible.
5. If several processes (e.g. I/O variable, technology object, PROFIdrive telegram) access an I/O address, the following applies:
   - Only a single process can have write access to an I/O address of an output (BYTE, WORD or DWORD data type).
     Read access to an output with an I/O variable that is used by another process for write access, is possible.
   - All processes must use the same data type (BYTE, WORD, DWORD or ARRAY of these data types) to access this I/O address. Access to individual bits is possible irrespective of this. Please be aware of the following, for example, if you wish to use an I/O variable to read the PROFIdrive telegram transferred to or from the drive: The length of the I/O variable must match the length of the telegram.
   - Write access to different bits of an address is possible from several processes; however, write access with the data types BYTE, WORD or DWORD is then not possible.

Note

These rules do not apply to accesses to the fixed process image of the BackgroundTask (Page 4103). These accesses are not taken into account during the consistency check of the project (e.g. during download).

Creating I/O variables for direct access or process image of cyclic tasks

Create I/O variables for direct access or a process image of the cyclic tasks in the address list of the detail view.

This is only possible in offline mode.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select **Insert new line**
   or
   Scroll to the end of the table of variables (empty line).
3. In the empty row of the table, enter or select the following:
   - Names of the **I/O variables**
   - **I/O address**
     Select the "IN" or "OUT" entries if you wish to assign symbols to the I/O variable (input or output). As of Version V4.2 of the SIMOTION Kernel the symbolic assignment must be activated, menu **Project > Use symbolic assignment**.
     Or enter a fixed address according to "Syntax for entering I/O addresses" (Page 4100).
   - Optional for outputs:
     Activate the **Read only** checkbox if you only want to have read access to the output. You can then read an output that is already being written by another process (e.g. output of an output cam, PROFIdrive telegram).
     A read-only output variable cannot be assigned to the process image of a cyclic task.
   - **Data type** of the variables in accordance with "Possible data types of the I/O variables" (Page 4101).

4. Optionally, you can also enter or select the following (not for data type BOOL):
   - **Array length** (array size).
   - **Process image** or direct access:
     Can only be assigned if the **Read only** checkbox is deactivated.
     For process image, select the cyclic task to which you want to assign the I/O variable. To select a task, it must have been activated in the execution system.
     For direct access, select the blank entry.
   - **Strategy** for behavior in the event of an error, see SIMOTION Basic Functions Function Manual.
   - **Display format** (if array, for each element), when you monitor the variable in the address list
   - **Substitute value** (if array, for each element).
5. Only if you have selected "IN" or "OUT" as the I/O address (symbolic assignment).
   - In the **Assignment** column, click the **[...]** button.
     A window opens displaying the possible assignment targets of the SIMOTION device and, if necessary, of SINAMICS Integrated. Only those assignment targets are displayed that match the data direction (input/output) and data type.
   - Select the assignment target.
     The **Assignment status** column indicates whether the assignment was successful or not.

   For details regarding symbolic assignment, refer to the SIMOTION Basic Functions Function Manual.

You can now access this variable using the address list or any program of the SIMOTION device.

Details on how to manage the address list can be found in the online help.

Note

Note the following for the process image for cyclic tasks:

- A variable can only be assigned to one task.
- Each byte of an input or output can only be assigned to one I/O variable.

In the case of data type BOOL, please note:

- The process image for cyclic tasks and a strategy for errors cannot be defined. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access or CPU stop).
- The individual bits of an I/O variable can also be accessed using the bit access functions.

Take care when making changes within the I/O variables (e.g. inserting and deleting I/O variables, changing names and addresses):

- In some cases the internal addressing of other I/O variables may change, making all I/O variables inconsistent.
- If this happens, all program sources that contain accesses to I/O variables must be recompiled.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and then use them in your program sources (e.g. ST sources, MCC sources, LAD/FBD sources).

Outputs can be read and written to, but inputs can only be read.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4113).

Syntax for entering I/O addresses

Syntax

For the input of the I/O address for the definition of an I/O variable for direct access or process image of cyclical tasks (Page 4094), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

Table 7-28 Syntax for the input of the I/O addresses for direct access or process image of the cyclic tasks

Data type	Syntax for		Permissible address range					
	Input	Output	Direct access		Process image		e.g. direct access D435 V4.1	
BOOL	PIn.x	PQn.x	n: x:	0 .. *MaxAddr* 0 .. 7		-[1]	n: x:	0 .. 16383 0 .. 7
BYTE	PIBn	PQBn	n:	0 .. *MaxAddr*	n:	0 .. *MaxAddr*	n:	0 .. 16383
WORD	PIWn	PQWn	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. 16382
DWORD	PIDn	PQDn	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. 16380
n = logical address x = bit number								
MaxAddr =	Maximum I/O address of the SIMOTION device depending on the SIMOTION Kernel version, see Address range of the SIMOTION devices (Page 4096).							
[1] For data type BOOL, it is not possible to define the process image for cyclic tasks. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access).								

Examples

Input at logic address 1022, WORD data type: **PIW1022**.

Output at logical address 63, bit 3, BOOL data type: **PQ63.3**.

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4097).

Possible data types of I/O variables

The following data types can be assigned to the I/O variables for direct access and process image of the cyclical tasks (Page 4094). The width of the data type must correspond to the data type width of the I/O address.

If you assign a numeric data type to the I/O variables, you can access these variables as integer.

Table 7-29 Possible data types of the I/O variables for direct access and the process image of the cyclical tasks

Data type of I/O address	Possible data types for I/O variables
BOOL (PIn.x, PQn.x)	BOOL
BYTE (PIBn, PQBn)	BYTE, SINT, USINT
WORD (PIWn, PQWn)	WORD, INT, UINT
DWORD (PIDn, PQDn)	DWORD, DINT, UDINT

For details of the data type of the I/O address, see also "Syntax for entering I/O addresses" (Page 4100).

Detailed status of the I/O variables (as of Kernel V4.2)

As of version V4.2 of the SIMOTION Kernel, the status of an I/O variable can be queried using *_quality.var-name*, for example, in order to check the availability of the I/O variables. It is supplied as an OR logic operation of the following status values in the DWORD data type and can be assigned to an appropriate variable, for example. The value 16#0000_0000 indicates the connected I/O are operating without error.

The same value is supplied for every I/O variable within an address range (slot or subslot) configured in HW Config.

Table 7-30 Meanings of status values of I/O variables

Value (DWORD)	Bit x = 1	Meaning
16#0000_0000	-	No error occurred.
16#0000_0001	0	**Maintenance required** The connected module signals that it requires maintenance. The component needs to be checked within a foreseeable period (e.g. the printer cartridge must be changed within a period of several days).
16#0000_0002	1	**Maintenance demanded** The connected module demands maintenance. The component needs to be checked soon (e.g. the printer cartridge must be changed immediately).
16#0000_0004	2	**Warning pending** (drive warning, TM17 warning, etc.) The connected module has signaled a warning. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.
16#0000_0008	3	**Fault pending** (diagnostic interrupt, drive fault, TM17 fault, etc.) The connected module has signaled an error. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.

Value (DWORD)	Bit x = 1	Meaning
16#0000_0010	4	**This parameter assignment does not match the parameter assignment being compared.** A difference was detected when this parameter assignment was compared with the parameter assignment of the connected module. As such, the required functionality cannot be guaranteed. Remedy: Save the project and compile changes, reload both application and counterpart.
16#0000_0020	5	**Application and counterpart are not isochronous** (error involving the dynamic life-sign). Certain telegrams (axis, synchronous operation, output cam, measuring input telegrams) are synchronized by exchanging cyclic life-signs. Errors are detected when the cyclic life-sign is checked. This invalidates the data in the telegram. Remedy: Await synchronization, check the parameter assignment (e.g. does the master application cycle set on the device in HW Config match the position control cycle clock), save the project and compile changes, reload both application and counterpart.
16#0000_0040	6	**I/O cannot be used synchronously in all cycles.** A fast application cycle (Servo_fast) and a slow application cycle (Servo) are running asynchronously in relation to one another. The I/O can only be used synchronously in the cycles associated with the bus cycle. Access from other cycles is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0080	7	**I/O cannot be used synchronously** The SIMOTION control is the sync slave on a bus. The bus connection is running synchronously in relation to the sync master, but is not yet running synchronously in relation to the application cycles of the SIMOTION control. Access to the I/O is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0100	8	**Bus connection (sync slave) is not isochronous in relation to the sync master.** The SIMOTION control is the sync slave on a bus and has not yet synchronized its bus connection with the sync master. The isochronous I/O on this bus cannot be used yet. Remedy: Switch on/connect the sync master.
16#0000_0200	9	**DP station is deactivated.** The partner module has been deactivated. Remedy: Activate the partner module (_activateDpSlave() function).
16#0000_0400	10	**The partner of the inputs (e.g. I-device, I-slave) is in STOP.** The connected module is in STOP mode and not sending any new data as a result. Remedy: Switch the connected module to RUN.
16#0000_0800	11	**PROFINET: Failure detected by submodule** (e.g. channel error) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_1000	12	**PROFINET: Failure detected by module** (e.g. submodule failed, removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_2000	13	**PROFINET: Failure detected by device** (e.g. device in STOP, module removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config

Value (DWORD)	Bit x = 1	Meaning
16#0000_4000	14	**PROFINET: Failure detected by controller** (e.g. not connected, etc.) There is no connection to a partner on PROFINET. Possible cause: Partner is switched off, cable pulled out, incorrect parameter assignment for connection Troubleshooting: Best to use the PROFINET topology editor in HW Config.
16#0000_8000	15	**Slot/subslot is not connected** (disconnection alarm). The connection to the connected device is OK. The error must be searched for in the connected device (e.g. module/submodule removed). Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0001_0000	16	**Device is not connected** (station failure). There is no connection to a partner. Possible cause: Partner is switched off, cable pulled out.
16#0002_0000	17	**Substitute value behavior during access** There is no connection to the counterpart (sum signal from bits 9 to 16), i.e. there is no valid input data or the output data is not reaching the terminal. The substitute value behavior set (substitute value, last value) takes effect during direct access to this address or during process image updates.
16#4000_0000	30	**Diagnostics address only** No cyclic I/O data is configured for this address. It is possible, however, to query submodule diagnostic information.
16#8000_0000	31	**Address gap** There is no hardware configured for this logical address.

Access to fixed process image of the BackgroundTask

The fixed process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device on which a subset of the I/O address space of the SIMOTION device is mirrored. Preferably, it should be used for programming the BackgroundTask (cyclic programming) as it is consistent throughout the entire cycle.

The size of the fixed process image of the BackgroundTask for all SIMOTION devices is 64 bytes (address range 0 .. 63).

Note

The fixed process image of the BackgroundTask can be used to access addresses that are not available in the I/O or not configured in HW Config. These are treated like normal memory addresses.

Memory area

- **As of Version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks.
  I/O addresses can be read and written to using both the fixed process image of the BackgroundTask and the process image of the cyclic tasks.
- **With Version V4.1 and lower of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of Version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas. I/O addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.

Table 7-31 Effect of "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask

	Common process image	Separate process image
Availability	Only available as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Version V4.1 and lower of the SIMOTION Kernel applies: • System characteristic, not configurable The following applies as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Memory area	Subset of the memory area for the process image of the cyclic tasks	Separate memory area for the process image of the cyclic tasks
Using I/O addresses accessed using the process image of the cyclic tasks	Possible. Updates use the configured cyclic tasks.	Not supported. The addresses are treated like normal memory addresses.
Byte order when forming the process image	As supplied by the I/O	Always Big Endian
Byte order when accessing the process image	Always Big Endian	Always Big Endian
Access to I/O operating in the Little Endian byte order	Same result as during direct access or for the process image of cyclic tasks (apart from WORD or DWORD data types).	Results differ depending on the I/O variables created for direct access.
Effects on the process image of the cyclic tasks	See the relevant table in "Direct access and process image of the cyclic tasks (Page 4096)".	
Further information	Common process image (Page 4105)	Separate process image (Page 4107)

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

A comparison of the most important properties in comparison to the direct access and process image of the cyclic tasks (Page 4094) is contained in "Important properties of direct access and process image" (Page 4091).

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4097) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

You can access the fixed process image of the BackgroundTask by means of:

- Using an absolute PI access (Page 4109): The absolute PI access identifier contains the address of the input/output and the data type.
- Using a symbolic PI access (Page 4111): You declare a variable that references the relevant absolute PI access:
  - A unit variable
  - A static local variable in a program.
- Using an I/O variable (Page 4113): In the symbol browser, you define a valid I/O variable for the entire device that references the corresponding absolute PI access.

Common process image (as of Kernel V4.2)

As of Version V4.2 of the SIMOTION Kernel, the "Common process image" setting can be selected on the SIMOTION device. This means addresses 0 .. 63 of the process image of the cyclic tasks and the fixed process image of the BackgroundTask occupy the same memory area.

This is the default for SIMOTION devices newly created in the project as of Version V4.2.

Property of the common process image

1. The memory area for the fixed process image of the BackgroundTask (Page 4103) is a subset of the memory area for the process image of the cyclic tasks (Page 4094).
2. This means I/O addresses already accessed using the process image of the cyclic tasks may also continue to be used for the fixed process image of the BackgroundTask. Updates, however, use the configured cyclic tasks.
3. The following applies when forming the fixed process image of the BackgroundTask:
   The byte order is the same as supplied by the I/O:
   - Big Endian, e.g. for I/O via PROFIBUS DP, PROFINET, P-Bus, DRIVE-CLiQ
   - Little Endian, e.g. for onboard I/O of C240, C240 PN SIMOTION devices

   Any I/O variable created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks has no effect on the byte order.

4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C240, C240 PN SIMOTION devices).
   If the fixed process image of the BackgroundTask is used for access, this leads to the following behavior, regardless of whether I/O variables have been created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks:
   - Access to individual bytes always supplies the same result via an I/O variable or the fixed process image of the BackgroundTask.
   - With the fixed process image of the BackgroundTask, bytes only change places if data type WORD is used for access.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for common process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access or the process image of the cyclic tasks:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access or access to the process image of the cyclic tasks:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte function (direct access only)
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-32 "Common process image" setting (as of Kernel V4.2): Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access or access to the process image of the cyclic tasks	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	16#08
	%IB61	16#00	16#00	16#00
	%IW60	**16#0800** [2]	**16#0800** [2]	**16#0800** [2]
	%I60.3	TRUE	TRUE	TRUE
	%I61.3	FALSE	FALSE	FALSE

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access or the process image of the cyclic tasks; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian byte order is being read using Big Endian.

Separate process image (up to Kernel V4.1)

With Version V4.1 and below of the SIMOTION Kernel, the process image of the cyclic task and the fixed process image of the BackgroundTask are stored in different memory areas (separate process image).

As of Version V4.2 of the SIMOTION Kernel, the "Separate process image" setting can be selected on the SIMOTION device. This setting ensures there is compatibility with earlier Kernel versions.

It is the default for SIMOTION devices upgraded to Version V4.2 or higher.

Property of the separate process image

1. The fixed process image of the BackgroundTask (Page 4103) and the process image of the cyclic tasks (Page 4094) are stored in different memory areas.
2. This means I/O addresses that are already accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.
3. I/O variables for direct access influence the fixed process image of the BackgroundTask:
   – The fixed process image of the BackgroundTask is always formed for the relevant addresses in the Big Endian byte order.
4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C230-2, C240, C240 PN SIMOTION devices). If an I/O variable is created for the relevant addresses for the purpose of direct access using data type WORD and access takes place using the fixed process image of the BackgroundTask, this leads to the following behavior:
   – Access with the data type WORD supplies the same result via the I/O variable and the fixed process image of the BackgroundTask.
   – Access to individual bytes using the _getInOutByte function (see SIMOTION Basic Functions Function Manual) supplies these in the Little Endian order.
   – Access to the individual bytes or bits with the fixed process image of the BackgroundTask supplies these in the Big Endian order.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for separate process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte function
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-33 "Separate process image" setting or Kernel up to Version V4.1: Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	**16#00** [3]
	%IB61	16#00	16#00	**16#08** [3]
	%IW60	**16#0800** [2]	**16#0800** [2]	16#0008
	%I60.3	TRUE	TRUE	**FALSE** [3]
	%I61.3	FALSE	FALSE	**TRUE** [3]

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian order is being read using Big Endian.

[3] The two adjacent bytes change places, as the relevant word is saved in the Big Endian order.

Absolute access to the fixed process image of the BackgroundTask (absolute PI access)

You make absolute access to the fixed process image of the BackgroundTask (Page 4103) by directly using the identifier for the address (with implicit data type). The syntax of the identifier (Page 4110) is described in the following section.

You can use the identifier for the absolute PI access in the same manner as a normal variable.

Note

Outputs can be read and written to, but inputs can only be read.

Syntax for the identifier for an absolute process image access

For the absolute access to the fixed process image of the BackgroundTask (Page 4109), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

You also use these identifiers:

- For the declaration of a symbolic access to the fixed process image of the BackgroundTask (Page 4111).
- For the creation of an I/O variables for accessing the fixed process image of the BackgroundTask (Page 4113).

Table 7-34 Syntax for the identifier for an absolute process image access

Data type	Syntax for		Permissible address range	
	Input	Output		
BOOL	%In.x or %IXn.x [1]	%Qn.x or %QXn.x[1]	n: x:	0 .. 63 [2] 0 .. 7
BYTE	%IBn	%QBn	n:	0 .. 63 [2]
WORD	%IWn	%QWn	n:	0 .. 63 [2]
DWORD	%IDn	%QDn	n:	0 .. 63 [2]
n = logical address x = bit number				
[1] The syntax %IXn.x or %QXn.x is not permitted when defining I/O variables. [2] For a separate process image (Page 4107), the following applies: No addresses that are used in the process image of the cyclic tasks. See note below.				

Examples

Input at logic address 62, WORD data type: **%IW62**.

Output at logical address 63, bit 3, BOOL data type: **%Q63.3**.

Note

Up to Version V4.1 of the SIMOTION Kernel or the "Separate process image" (Page 4107) setting on the device (as of Version V4.2 of the SIMOTION Kernel), the following applies:

- Addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask.

This restriction no longer applies as of Version V4.2 of the SIMOTION Kernel or with the "Common process image" (Page 4105) setting on the device.

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4097) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

Defining symbolic access to the fixed process image of the BackgroundTask

You create symbolic access to the fixed process image of the BackgroundTask in the declaration tables of the source file, MCC chart, or LAD/FBD program (only in the case of programs). The scope of the symbolic process image access is dependent on the location of the declaration:

- In the interface section of the declaration table of the source file (INTERFACE):
  Symbolic process image access behaves like a unit variable; it is valid for the entire source file; all MCC charts or LAD/FBD programs (programs, function blocks, and functions) within the source file can access the process image.
  In addition, these variables are available on HMI devices and, once connected, in other source files (or other units), as well.
  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the source file (IMPLEMENTATION):
  Symbolic process image access behaves like a unit variable; it is only valid in the source file; all MCC charts or LAD/FBD programs (programs, function blocks, and functions) within the source file can access it.
- In the declaration table for the MCC chart or LAD/FBD program (only in the case of programs):
  Symbolic process image access behaves like a local variable; it can only be accessed within the MCC chart or LAD/FBD program in which it is declared.
  No symbolic process image access can be declared in functions or function blocks.

Proceed as follows; the source file or the MCC chart or LAD/FBD program (in the case of programs only) with the declaration table is opened:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the I/O Symbols tab.
3. Enter:
   - Name of symbol (variable name)
   - For Absolute ID, the identifier of the absolute process image access (Page 4110).
   - Data type of symbol (Page 4112) (this must agree with the length of the process image access).

Possible data types for symbolic PI access

In the following cases, a data type that differs from that of the absolute PI access can be assigned to the fixed process image of the BackgroundTask (Page 4103). The data type width must correspond to the data type width of the absolute PI access.

- For the declaration of a symbolic PI access (Page 4111).
- For the creation of an I/O variable (Page 4113).

If you assign a numeric data type to the symbolic PI access or to the I/O variables, you can access these variables as integer.

Table 7-35 Possible data types for symbolic PI access

Data type of the absolute PI access	Possible data types of the symbolic PI access
BOOL (%In.x, %IXn.x, %Qn.x. %QXn.x)	BOOL
BYTE (%IBn, %QBn)	BYTE, SINT, USINT
WORD (%IWn, %QWn)	WORD, INT, UINT
DWORD (%IDn, %PQDn)	DWORD, DINT, UDINT

For the data type of the absolute PI access, see also "Syntax for the identifier for an absolute PI access (Page 4110)".

Example: Defining symbolic access to the fixed process image of the BackgroundTask

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Absolute identifier	Data type	Comment
1	input_1	%IB62	SINT	
2	output_1	%QB62	BYTE	
3				

Figure 7-49 Example: Defining symbolic access to the fixed process image of the BackgroundTask

Creating an I/O variable for access to the fixed process image of the BackgroundTask

You create I/O variables for access to the fixed process image for the background task in the symbol browser in the detail view; you must be in offline mode to do this.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select Insert new line
   or
   Scroll to the end of the table of variables (empty line).
3. In the detail view, select the Symbol browser tab and scroll down to the end of the variable table (empty row).
4. In the empty row of the table, enter or select the following:
   - **Name** of variable.
   - Under **I/O address**, the absolute PI access according to the "Syntax for the identifier for an absolute PI access" (Page 4110)
     (exception: The syntax %IXn.x or %QXn.x is not permitted for data type BOOL).
   - **Data type** of the I/O variables according to the "Possible data types of the symbolic PI access" (Page 4112).
5. Select optionally the display format used to monitor the variable in the symbol browser.

You can now access this variable using the address list or any program of the SIMOTION device.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and use them in your program sources.

Note that you can read and write outputs but you can only read inputs.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4113).

Accessing I/O variables

You have created an I/O variable for:

- Direct access or process image of the cyclic tasks (Page 4094).
- Access to the fixed process image of the BackgroundTask (Page 4103).

You can use this I/O variable just like any other variable.

Note

Consistency is only ensured for elementary data types.

When using arrays, the user is responsible for ensuring data consistency.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the I/O variable using *_device.var-name* (predefined name space, see the "Predefined name spaces" table in "Name spaces").

It is possible to directly access an I/O variable that you created as a process image of a cyclic task. Specify direct access with *_direct.var-name* or *_device._direct.var-name*.

If you want to deviate from the default behavior when errors occur during variable access, you can use the _getSafeValue and _setSafeValue functions (see *SIMOTION Basic Functions* Function Manual).

For Errors associated with access to I/O variables, see *SIMOTION Basic Functions* Function Manual.

7.1.5.9 Connections to other program source files or libraries

In the declaration table of a unit, you can define connections to:

- LAD/FBD units under the same SIMOTION device
- MCC units under the same SIMOTION device
- ST source files under the same SIMOTION device
- Libraries

This will then allow you to access the following in this unit:

- For connected program sources (Page 4115), the following items which are defined there
  - Functions
  - Function blocks
  - Classes and methods contained therein (only for ST source files)
  - Programs (optional)
  - Unit variables
  - User-defined data types (structures, enumerations)
  - Symbolic accesses to the fixed process image of the BackgroundTask
- For connected libraries (Page 4116), the following items which are defined there
  - Functions
  - Function blocks
  - Classes and methods contained therein (only for ST libraries)
  - Programs (optional)
  - User-defined data types (structures, enumerations)

Program sources and libraries must be compiled beforehand.

For information about the library concept, see also the SIMOTION ST Programming Manual.

Note

Libraries can be created in all programming languages (MCC, ST, or LAD/FBD).

Defining connections

Procedure for defining connections to other program sources (units)

Connections to other units (program sources) are defined in the declaration table of the source file. The mode of action of a connection is dependent on the section of the declaration table in which it is defined.

- In the interface section of the declaration table:
  The imported functions, variables, etc., will continue to be exported to other units and to HMI devices. This can lead to name conflicts.
  This setting is necessary, for example, if unit variables are declared in the interface section of the source file with a data type that is defined in the imported program source.
- In the implementation section of the declaration table:
  The imported functions, variables, etc. will no longer be exported.
  This setting is usually sufficient.

Proceed as follows; the source file (declaration table) is open (see Open existing program sources (Page 3989)):

1. In the declaration table, select the section for the desired mode of action.
2. Select the **Connections** tab.
3. For the connection type, select: **Program/Unit**
4. In the same line, select the name of the unit to be connected:
   Units (program sources) must be compiled beforehand.

Procedure for defining connections to libraries

Connections to libraries are defined in the declaration table of the source file.

Proceed as follows; the unit (declaration table) is open, see Open existing program sources (Page 3989):

1. In the interface section of the declaration table, select the **Connections** tab.
2. For the connection type, select: **Library**.
3. In the same line, select the name of the library to be connected.
   Libraries must be compiled beforehand.
4. Optionally, you can define a name space for libraries, see Using name space (Page 4116):
   To do this, enter a name under **Name space**.

Note

When programming the "Subprogram call" command (see Inserting and parameterizing subroutine calls (Page 4121)) with a library function or a library function block, the connection to the library is automatically entered into the declaration table of the program source.

Using the name space

You can optionally assign a namespace to every connected library. You define the designation of the namespace when connecting the library (see How to define connections to libraries (Page 4116)).

It is important to specify the namespace if the current LAD/FBD program / MCC chart or program source contains variables, data types, functions, or function blocks with the same name as the connected library. The namespace will then allow you specific access to the variables, data types, functions, or function blocks in the library. This can also resolve naming conflicts between connected libraries.

If you wish to use variables, data types, functions, or function blocks from the connected library in a command in the LAD/FBD program or MCC chart, insert the designation of the namespace in front of the variable name, etc., from the library and separate them with a period (for example, namespace.var_name, namespace.fc_name).

Predefined namespaces

Namespaces are predefined for device-specific and project-specific variables, direct accesses to I/O variables, and variables of TaskId and AlarmId in the following table: If necessary, write their designation before the variable name, separated by a period, e.g. *_device.var_name* or *_task.task_name*.

Table 7-36 Predefined namespaces

Namespace	Description
_alarm	For AlarmId: The _alarm.name variable contains the AlarmId of the message with the name identifier – see SIMOTION Basic Functions Function Manual.
_device	For device-specific variables (global device user variables, I/O variables, system variables, and system variables of the SIMOTION device)
_direct	For direct access to I/O variables – see Direct access and process image of the cyclic tasks (Page 4094). Local namespace for _device. Nesting as in _device._direct.*name* is permitted.
_project	For names of SIMOTION devices in the project; only used with technology objects on other devices. With unique project-wide names of technology objects, used also for these names and their system variables
_quality	As of Version V4.2 of the SIMOTION Kernel: For the detailed status of I/O variables (Page 4101). A value with data type DWORD is supplied. Local namespace for _device. Nesting as in _device._quality.*name* is permitted.
_task	For TaskID: The _task.name variable contains the TaskId of the task with the *name* identifier – see SIMOTION Basic Functions Function Manual.
_to	For technology objects configured on the SIMOTION device and their system variables and configuration data Not for system functions and data types of the technology objects. In this case, use the user-defined namespace for the imported technology package, if necessary.

Object-oriented namespaces

You can assign functions, function blocks or programs to an object-oriented namespace. Enter the identifier of the object-oriented namespace when creating (Page 4001) or in the Properties window (Page 4008) of the respective program organization unit (POU). The compiler option (Page 3996) "Permit object-oriented programming" must be activated.

A POU within the namespace can only be accessed from a POU outside the namespace when the identifier of the namespace is added as prefix to the identifier of the POU, separated by a dot.

In the project navigator, the POU is shown as subelement of the namespace below the program source.

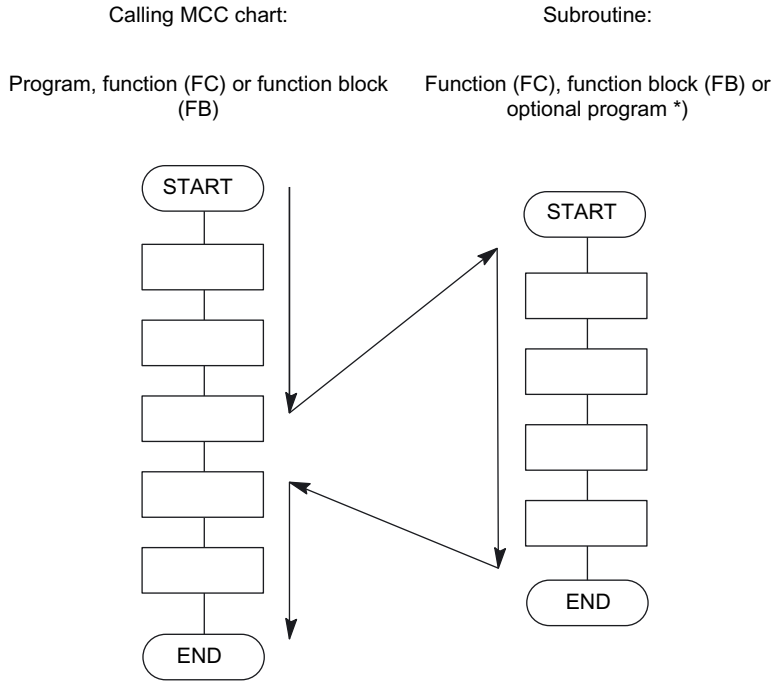
7.1.5.10 Subroutine

Universal, reusable sections of a program can be created in the form of subroutines.

When a subroutine is called, the program branches from the current task into the subroutine. The commands in the subroutine are executed. The program then jumps back to the previously active task.

Subroutines can be called any number of times by one or more MCC programs of the SIMOTION device.

Example of executing a subroutine



*) A program can be called as a subroutine by another program or function block (FB) ("program in program"). Specific compiler options must be set for this (see below).

Figure 7-50 Execution of a subroutine

Subroutine as a function (FC), function block (FB), or program (optional)

The creation type of a subroutine can be a function (FC), a function block (FB) or, as an option, a program ("program in program").

- Function
  A function (FC) is a subroutine without static data, that is, all local variables lose their value when the function has been executed. They are re-initialized when the function is next started.
  Data are transferred to the function using input or in/out parameters; the output of a function value (return value) is also possible.
- Function block
  A function block (FB) is a subroutine with static data, that is, local variables retain their value after the function block has been executed. Only variables that have been explicitly declared as temporary lose their value.
  An instance has to be defined before using an FB: Define a variable (VAR or VAR_GLOBAL) and enter the name of the FB as data type. The FB static data is saved in this instance. You can define several FB instances; each instance is independent from the others.
  The static data of an FB instance remain stored until the instance is next called; they are reinitialized when the variable type of the FB instance is initialized again (see Initialization of instances of function blocks (FB) (Page 4079)).
  Data are transferred to the FB using input parameters or in/out parameters; the data are returned from the FB using in/out or output parameters.
- Program ("program in program")
  You also have the option of calling a program within a different program or a function block. This requires the following compiler options to be activated (see Global compiler settings (Page 3996) and Local compiler settings (Page 3997)):
  - "Permit language extensions" for the program source of the calling program or function block and
  - "Only create program instance data once" for the program source of the called program. The static data of the called program is stored in the user memory of the program source (unit) of said called program.

  Most of the programming work involved in assigning the programs to the tasks can be performed by calling up programs within another program. In the execution system, only one associated calling program needs to be assigned to the tasks concerned.
  A program is called without parameters or return values.
  Further information on calling a program within a program can be found in the SIMOTION ST Programming and Operating Manual.

Note

The activated "Only create program instance data once" compiler option causes:

- The static variables of the programs (program instance data) to be stored in the user memory of the program source (unit) (see the SIMOTION ST Programming and Operating Manual). This also causes the initialization behavior to change (see the SIMOTION ST Programming and Operating Manual).
- All called programs with the same name use the same program instance data.

Exchange of information between the subroutine and calling program

Function (FC) and function block (FB) as a subroutine

Information is exchanged between the subroutine and the calling program using transfer parameters or global variables (e.g. unit variables).

Transfer parameters can be input, input/output or output parameters. They are defined in the declaration table for the subroutine (MCC chart):

- Input parameters: As variable type VAR_INPUT
- In/out parameter: As variable type VAR_IN_OUT
- Output parameter (for FB only): As variable type VAR_OUTPUT

For functions, a function value can be returned; you specify the data type of the return value when you insert (create) the function (see Insert function (FC) or function block (FB) (Page 4120)).

You assign current values to the input and/or in/out parameters when you call the subroutine (FC or FB instance). You may only assign user-defined variables to the in/out parameters of an FB because the called FB accesses the assigned variables directly and can therefore change them.

The output parameters of an FB can be read-accessed as often as required in the calling program.

A function does not formally contain any output parameters, since the result of the function can in this case be assigned to the return value of the function.

See also the examples of functions (Page 4128) and function blocks (Page 4132).

Program as subroutine ("program in program")

A program is called without parameters or return values. This means that information can only be exchanged between the calling program and the called program (subroutine) using global variables (e.g. unit variables).

See also

Inserting a subroutine call into the MCC chart and assigning parameters (Page 4121)

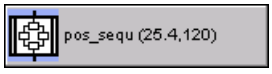
Inserting a function (FC) or function block (FB)

The creation dialog is similar to that of an MCC program:

1. The MCC unit must already exist (see Inserting an MCC unit (Page 3986)).
2. Select the relevant MCC unit in the project navigator.

3. Select the **Insert > Program > MCC chart** command from the menu.
   The input screen form opens.
   - Enter the name of the MCC chart.
     Names for MCC charts must comply with the Rules for identifiers (Page 4047): They are made up of letters (A ... Z, a ... z), digits (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper and lower case letters. Protected or reserved identifiers are not permitted.
     The permissible length of the name is 25 characters.
     The names must be unique within the MCC unit. The names of all exportable program organization units (POUs) must also be unique within the SIMOTION device. The names of all MCC charts of the program source as well as the names of all exportable POUs of the device are displayed.
   - Optionally enter the identifier of an object-oriented namespace.
     Identifiers for namespaces must comply with the Rules for identifiers (Page 4047) (see above).
     When an identifier is entered, the MCC chart is assigned to the specified namespace.
     The compiler option (Page 3996) "Permit object-oriented programming" must be activated. For further information on object-oriented namespaces, see the appropriate section in the SIMOTION ST Programming and Operating Manual.
   - For the creation type, select **Function** or **Function block**.
   - With creation type Function only:
     Select the data type of the return value as the return type (<--> for no return value).
   - Activate the **Exportable** checkbox if the function or function block is to be used in other program sources (LAD/FBD unit, MCC unit or ST source files).
     If the checkbox is deactivated, the MCC chart can only be used in the associated MCC unit (see also Changing usability in other charts (export capability) (Page 4010)).
   - You can also enter an author, version, and a comment.
   - Click **OK** to confirm.
4. Program the instructions in the function or function block.
   Assign an expression to the return value of a function (= function name) or to the output parameters of a function block.
5. Accept and compile the MCC unit. The subprogram you have created will then be displayed in the list.

Inserting a subroutine call into the MCC chart and assigning parameters



The Subprogram call command is available on the MCC editor toolbar (Basic commands command list).

You can use this command to call:

- Functions or function blocks of the same MCC source file or a different program source file (e.g., MCC source file, ST source file).
- Library functions or library function blocks from a program library.

- Optional programs (of the same MCC unit or a different program source) or library programs (from a program library). Specific compiler options must be set for this; see Parameter overview (Page 4124).
  A program can be called within a different program or a function block (FB).
- Public methods (access identifier PUBLIC) within classes or function blocks that are declared in ST source files or libraries. You can only generate methods and classes in the Structured Text (ST) programming language (object-oriented programming).
  The use of methods in classes requires a SIMOTION Kernel as of version V4.5.

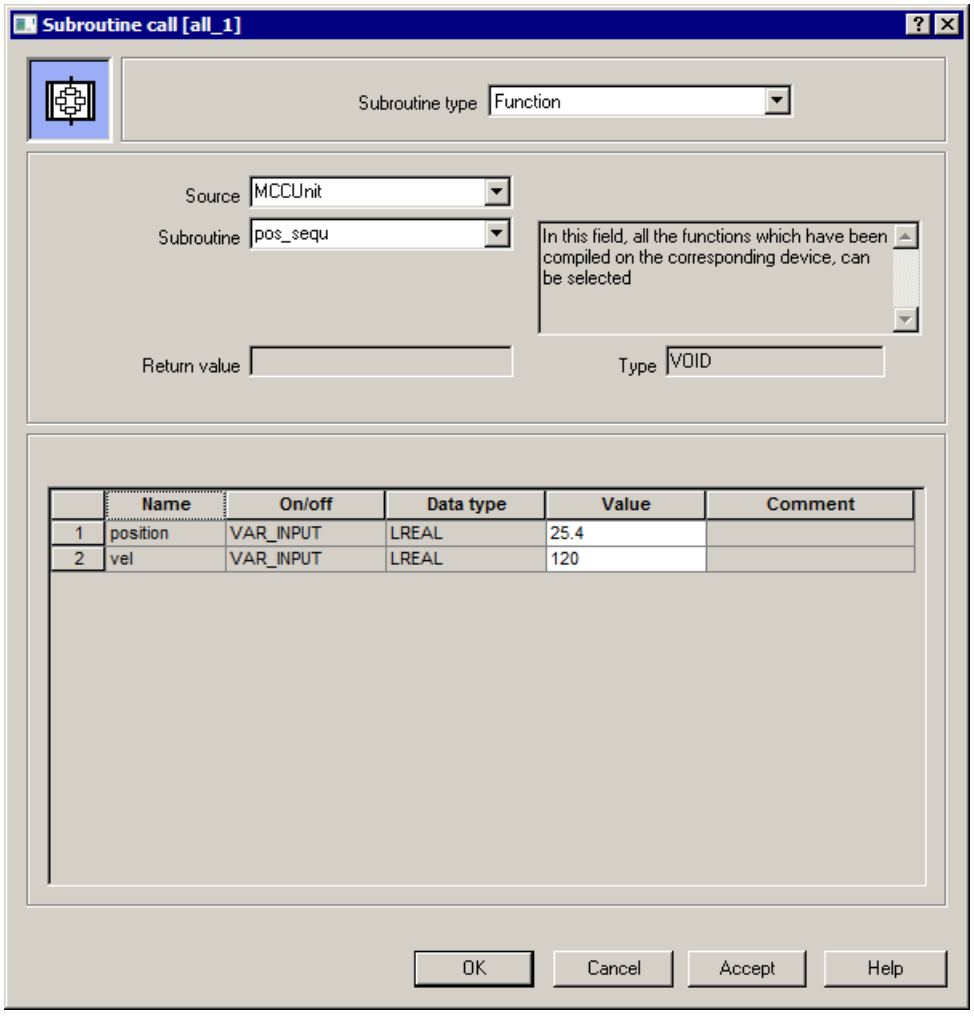


Figure 7-51 Subprogram call for a function

Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the MCC charts in an MCC unit. A subprogram (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the MCC chart of the subprogram appears above the chart in which it is used in the project navigator If necessary, reorder the charts.

See also: Subprogram call of function (FC) (Page 4129)

Overview of subprogram call parameters

You can set the following parameters:

Table 7-37 Overview of subprogram call parameters

Field/button	Meaning/instruction
Subprogram type	Here, select the subprogram type. **Function** (default value) A function is a subprogram that has no memory beyond the call. In other words, the data is lost once the function has run and the output parameters and return value have been transferred. A function can also have a return value (function value) in addition to input, in/out and output parameters. **Function block** A function block is a subprogram which has a memory beyond the call. In other words, the values of the function block are retained after it has run. Variables of the data type for this function block must be declared. These variables are identified as instances of the function block; they include the memory of the function block beyond the calls. A function block only has input, in/out and output parameters but does not have a return value. The output parameters of the function block can be accessed even at a later time (after the function block has run) at the function block instance. **Library function** Function from library **Library function block** Function block from the library **Program** ("program in program") You also have the option of calling a program within a different program or a function block. This requires the following compiler options to be activated, see Global compiler settings (Page 3996) and Local compiler settings (Page 3997): • "Permit language extensions" for the program source of the calling program or function block and • "Only create program instance data once" for the program source of the called program. The static data of the called program is stored in the user memory of the program source (unit) of the called program. The same program instance data is used every time the program is called. A program is called without parameters or return values. **Library program** Program from library. The same restrictions apply as when calling a program. The static data for the called program is stored once in the user memory of the device that the library program was called on. The same program instance data is used every time the program is called on the same device.

Field/button	Meaning/instruction
Subprogram type (continued)	**Method** You can call up methods that have been defined in ST source files (object-oriented programming): • SIMOTION Kernel as of version V4.5: Methods that have been defined within classes or object-oriented interfaces. • Irrespective of the SIMOTON Kernel version: Methods that have been defined within function blocks. Activating the following compiler option is recommended, see Global settings of the compiler (Page 3996) and Local settings of the compiler (Page 3997): • "Permit object-oriented programming" for the MCC unit of the MCC chart being called. You cannot generate any methods in the MCC programming language. **Library method** (as of kernel V4.5) Method from library. The same restrictions apply as when calling a method.
Source	With "Function", "Function Block", "Program" or "Method" subprogram types only. All program sources for the SIMOTION device are displayed in the drop-down list. Select the program source which contains the subprogram.
Library	With "Library function", "Library function block", "Library program" or "Library method" subprogram types only. All libraries available in the project are displayed in the drop-down list box. Select the library which contains the subprogram.
Subprogram	With the following subprogram types only: • "Function" or "Library function" • "Function block" or "Library function block" • "Program" or "Library program" The drop-down list displays all of the subprograms for the selected subprogram type available in the selected program source or library. For subprogram assigned to an object-oriented namespace, the identifier of the namespace is added as a prefix, separated by a dot. Select the subprogram to be called. **Note** The functions, function blocks and programs themselves can be generated in every available programming language. They must be available in compiled form. A connection to the program source (if required) is automatically set up when the subprogram is selected. Library functions, function blocks and programs can be generated in every available programming language. A connection to the library (if required) is automatically set up when the subprogram is selected.
Program organization unit type	With "Method" or "Library method" subprogram types only. Select the program organization unit (POU) from which a method is to be called. The following are available: **Class** (default value) **Interface** **Function block** Classes, object-oriented interfaces and function blocks with methods can only be generated in the Structured Text (ST) programming language.

Field/button	Meaning/instruction
POU name	With "Method" or "Library method" subprogram types only. The drop-down list displays all of the program organization units (POUs) of the selected source or library and of the selected type. For POUs assigned to an object-oriented namespace, the identifier of the namespace is added as a prefix, separated by a dot. Select the name of the POU from which a method is to be called.
Instance	With "Function Block", "Library function block", "Method" or "Library method" subprogram types only. • The following applies to "Function block" or "Library function block" subprogram types: Select the name of the function block instance or enter it here. The instance contains the memory of the function block in the form of instance data. You define the instance as a variable whose data type is the name of the function block in one of the following ways: – in the declaration table of the MCC unit as VAR_GLOBAL or – in the declaration table of the MCC chart as VAR. • The following applies to "Method" or "Library method" subprogram types: – With POU type "Class" or "Function block" Select the name of the class or function block instance to which the method belongs or enter it here. The instance contains the memory of the class or the function block in the form of instance data. – With POU type "Interface" Select the name of the variable of the object-oriented interface to which the method belongs or enter it here. Before the subprogram call for an instance this variable must be assigned to a class which implements the instance. A corresponding test can be initiated before calling the method by selecting the "Test call to NULL" checkbox. You define the instance as a variable whose data type is the name of the class, the object-oriented interface or the function block in one of the following ways: – in the declaration table of the MCC unit as VAR_GLOBAL or – in the declaration table of the MCC chart as VAR.
Method	With "Method" or "Library method" subprogram types only. The drop-down list displays all of the public methods for the selected program organization unit (class, interface, function block). Select or enter the method to be called. A connection to the program source or library (if required) is automatically set up when the subprogram is selected.
Test call to NULL	With "Method" or "Library method" subprogram types **and** POU type "Interface". Activate the checkbox if the method is just to be called, if the variable for the object-oriented interface for an instance has been assigned to an implemented class.
Return value	With "Function", "Library function", "Method" or "Library method" subprogram types: Here, you enter the variable in which the return value is to be stored. The type of variable must match the return value type.
Type	With "Function", "Library function", "Method" or "Library method" subprogram types: The data type of the return value is displayed.
List of transfer parameters	
Name	The name of the transfer parameter is displayed here.

Field/button	Meaning/instruction
On/off	The variable type of the transfer parameter is displayed here. **VAR_INPUT** Input parameters (for functions, function blocks and methods) **VAR_IN_OUT** In/out parameters (for functions, function blocks and methods) **VAR_OUTPUT** Output parameters (for functions, function blocks and methods)
Data type	The data type of the transfer parameter is displayed here.
Value	Mandatory parameters are marked with "<???>" and optional parameters with "...". • The following applies to functions (FC) and methods: – Input parameters without a declared initial value and in/out parameters are mandatory parameters. – Input parameters with a declared initial value and output parameters are optional parameters. A subprogram call will only be functional if all the mandatory parameters are set. • The following applies to function blocks (FB): All parameters are optional. • The following applies to programs ("program in program"): No parameters present. Here, you assign current variables or values to the transfer parameters: • Input parameter (variable type VAR_INPUT): Here, you enter a variable name or an expression. The assignment of system variables or I/O variables is permissible; type transformations are possible. • In/out parameter (variable type VAR_IN): Enter a variable name; it must be directly writable and readable. System variables of SIMOTION devices and technology objects are not permitted nor are I/O variables. The data type of the in/out parameter must correspond to that of the assigned variables; application of type transformation functions is not possible. • Output parameter (variable type VAR_OUTPUT): The assignment of an output parameter to a variable is optional. – The following applies to function blocks: You can also access an output parameter after executing the function block. – The following applies to functions and methods: The values for the output parameters are lost after executing the function or method unless they have been assigned in the parameter screen. When assigned in this parameter screen form: Enter a variable name. The data type of the output parameter must correspond to that of the assigned variables; the application of type transformation functions is not possible.
Comment	The comment for the transfer parameters is displayed here.

Note

When selecting the subprogram type, the distinction as to whether the subprogram is defined on the device or in a library (e.g. function / library function), is only an aid in the selection of the respective subprogram. It has no effect on the code generation.

This way you can subsequently move the source of a subprogram from the device to a connected library (Page 4116) without changing the parameters of the "Subprogram call" command. The MCC unit is still compiled correctly.

If you parameterize the "Subprogram" command again, you may receive an error message that the subprogram has been deleted or renamed. Close the error message and select the new subprogram type (e.g. library function) and the library which contains the subprogram.

A check as to whether an identifier of a subprogram is hidden by an identifier with the same name in the MCC unit, is not performed. You can avoid name conflicts by using namespaces (Page 4116).

Example: Function (FC)

You want to create a subprogram with a circumference calculation for a circle. The calculation is performed in a function (FC). This is named Circumference.

The circle circumference calculation can thus be called as a subprogram by any task.

Formula for circumference calculation: Circumference = PI * 2 * radius

You define the Radius and PI variables in the declaration table of the function.

Creating and programming the function (FC)

1. In the **PROGRAMS** folder in the project navigator, open the MCC source file in which you want to create the function.
2. In the MCC source file, double-click on the entry **Insert MCC chart** and enter the following parameters in the open **Insert MCC chart** dialog window:
   - Enter the name **Circumference**.
   - For creation type, select **Function**.
   - For return type (data type of return value), select **REAL**.
   - Click **OK** to confirm.
3. In the declaration table, define the input parameter **radius** and the constant **PI** (see figure).
4. Program the variable assignment for the return value (see figure).
5. Accept and compile the MCC source file.

You have now finished programming the **Circumference** function.

	Name	Variable type	Data type	Array length	Initial value	Comment
1	radius	VAR_INPUT	REAL			
2	PI	VAR CONSTANT	REAL		3.14159	
3						

Figure 7-52 Declaring variables (e.g., input parameters) in the MCC chart

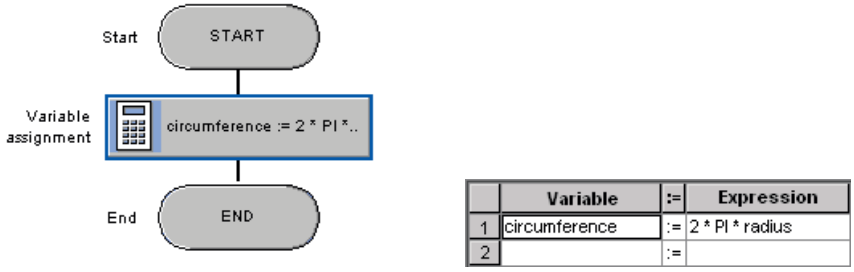


Figure 7-53 Programming the Circumference subroutine (e.g. assignment to a return value)

Subroutine call of function (FC)

The function (FC) is called from a program in the example.

Procedure

To call the function (FC) in a program, proceed as follows:

1. Create an MCC chart as a program in the same MCC unit (see Inserting a new MCC chart (Page 4001)).
2. In the MCC unit or MCC chart, declare the following (see figure):
   - The **mycircum** variable. The return value of the "Circumference" function is assigned to this variable.
   - The **myradius** variable. This variable contains the radius and is assigned to the input parameter "Radius" of the "Circumference" function.

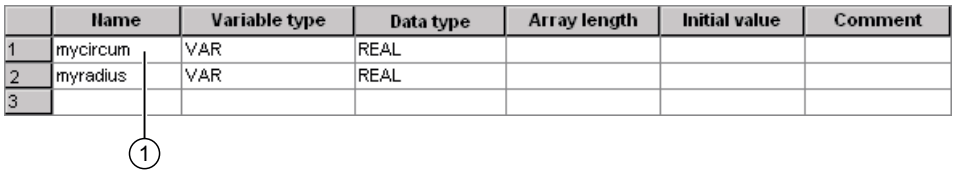
   Note that the validity range of the variables is dependent on the declaration location (see Defining variables (Page 4062)).
3. Insert the **Subprogram call** command.
4. Assign parameters in the parameter screen form (see figure).

   Note

   Mandatory parameters are marked with "<???>" and optional parameters with "...". A subprogram call will only be functional if all the mandatory parameters are set.

5. Accept and compile the MCC unit.

You have now finished programming the subprogram call.



① You can continue to use the mycircum variable in the program.

Figure 7-54 Declaring a variable in the MCC chart

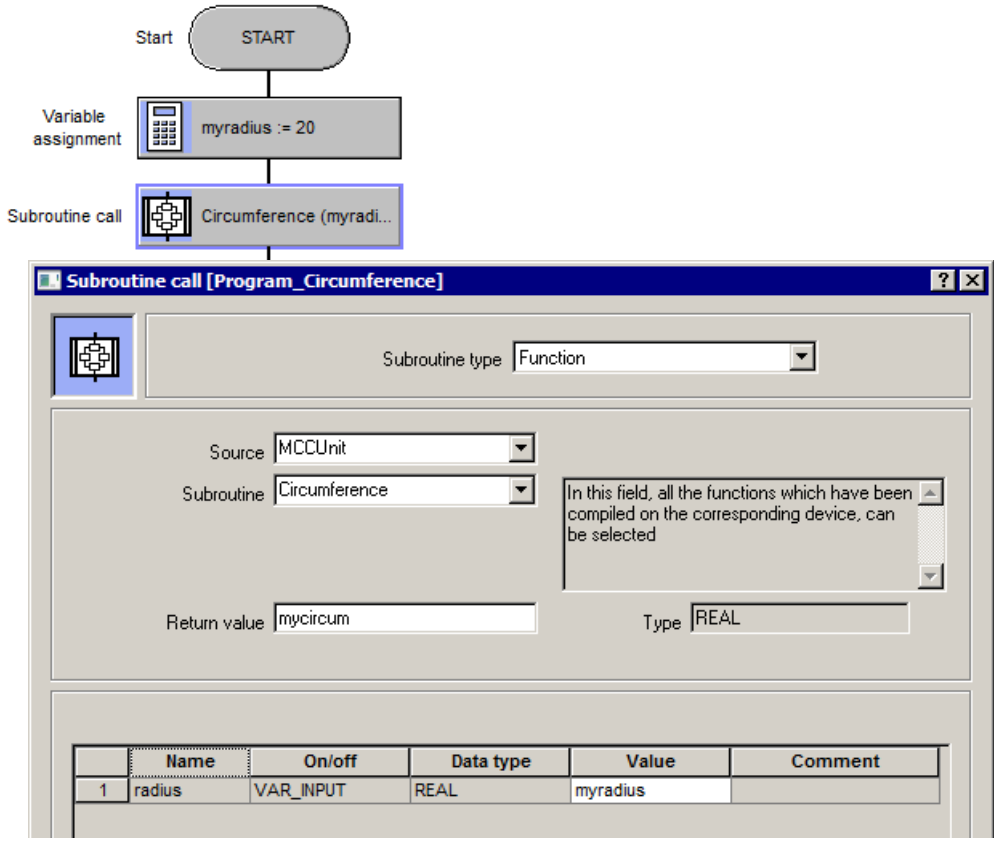


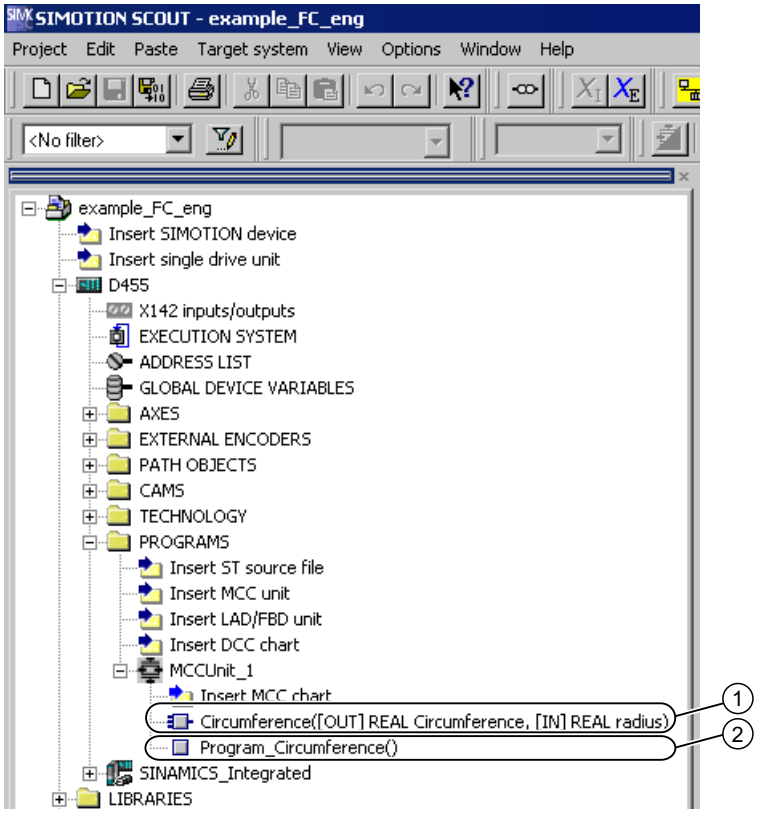
Figure 7-55 Opened parameter screen form for the subprogram call

Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the MCC charts in the MCC unit (see figure below). A subprogram (function or function block) must be defined before it is used. This is the case when the MCC chart of the subprogram appears above the chart in which it is used in the project navigator.

As far as the example described here is concerned, the MCC chart with the function must appear in the project navigator above the chart containing the program with the subprogram call. If necessary, rearrange the charts by selecting the relevant MCC chart in the project navigator and selecting the command **Down** or **Up** in the context menu.



① MCC chart with the function

② MCC chart containing the program with the subprogram call

Figure 7-56 Sequence of the MCC charts

Opening the function (FC) directly following the subroutine call

You can open subroutines directly following their respective subroutine call via the context menu:

- This works regardless of the programming language (ST, MCC, LAD/FBD) used to create the subroutine.
- The subroutine opens in a separate editor or an editor already opened for the subroutine is moved to the foreground.

A subroutine may be a:

- User-defined FC
- User-defined FB
- User-defined program called as a subroutine ("program in program")
- Library function
- Library function block
- Library program called as a subroutine ("program in program")

Procedure

To open the function (FC) directly following the subroutine call, proceed as follows:

1. In the MCC chart, select the **Subroutine call** command used to call the function (FC).
2. Select the **Open called block** command in the context menu (Ctrl+Alt+O hotkey).
   The function (FC) opens in a separate editor or an editor already opened for the function (FC) is moved to the foreground.

Note

If the function (FC) called has not yet been created, the **Open called block** command appears inactive (grayed out) in the context menu.

Note

If the function (FC) called is in a program source with know-how protection, the steps required when opening the protected program source directly must also be performed here (see Know-how protection for MCC units (Page 3992)).

Example: Function block (FB)

You want to calculate a following error. The calculation is performed in a function block (FB) named FollError. The following error calculation can thus be called as a subprogram by any task.

Formula for following error calculation: Difference = Specified position – Actual position

Define the required input and output parameters Set position, Actual position, and Difference (with the other variables, if necessary) in the MCC chart (function block) or MCC unit.

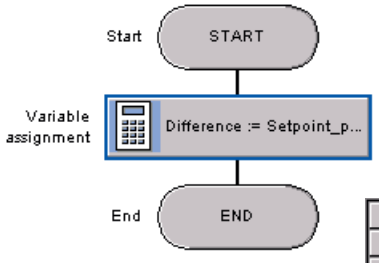
Creating and programming the function block (FB)

1. In the **PROGRAMS** folder in the project navigator, open the MCC source file in which you want to create the function block.
2. In the MCC source file, double-click on the entry **Insert MCC chart** and enter the following parameters in the open **Insert MCC chart** dialog window:
   - Enter the name **FollError**.
   - For creation type, select **Function block**.
   - Click **OK** to confirm.
3. In the declaration table, define the variables (e.g. input and output parameters; see figure).
4. The following error is calculated in a variable assignment (see figure).
5. Accept and compile the MCC source file.

You have now finished programming the **"FollError"** function block.

	Name	Variable type	Data type	Array length	Initial value	Comment
1	Setpoint_position	VAR_INPUT	LREAL			
2	Actual_position	VAR_INPUT	LREAL			
3	Difference	VAR_OUTPUT	LREAL			
4						

Figure 7-57 Declaring variables (e.g. input and output parameters) in the MCC chart



Formula language

	Variable	:=	Expression
1	Difference	:=	Setpoint_position - Actual_position
2		:=	

Figure 7-58 Programming a variable assignment

Subroutine call of function block (FB)

In this example, the function block (FB) is called from a program.

1. Generate an MCC chart as a program, see Insert new MCC chart (Page 4001).
2. Create a function block instance.
   – In the MCC source file or MCC chart, declare the instances of the function block along with the variables.

   Note that the validity range of the instance and variables is dependent on the declaration location (see Define variables (Page 4062)).
3. Call the function block:
   – Program the **Subroutine call** command.
4. After executing an instance of the function block, you can access the output parameters at any location in the calling program.
   – Program the **Variable assignment** command.
5. Accept and compile the program.

You have now finished programming the subroutine call.

Creating a function block instance

Before you can use a function block, you must define an instance. Each instance of an FB is independent of the others; once an instance has ended, its static variables remain stored.

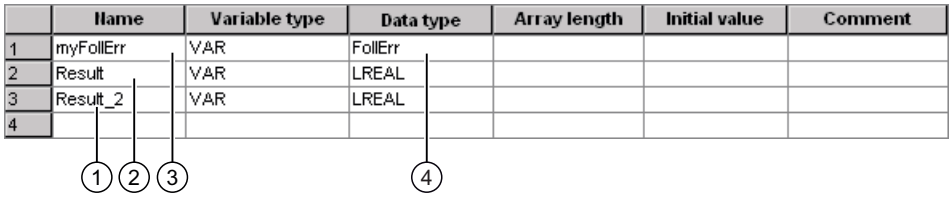
You define the instances of an FB in the declaration table of the MCC source file or MCC chart. The scope of the instance declaration is dependent on the location of the declaration:

- In the interface section of the declaration table of the MCC source file:
  The instance behaves like a unit variable; it is valid for the entire MCC source file; all MCC charts (programs, function blocks, and functions) within the MCC source file can access the instance.
  The instance is also available:
  - On HMI devices.
  - In other MCC source files (or other units) once connected, see "How to define connections to other units (program sources)" (Page 4115).

  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the MCC source file:
  The instance behaves like a unit variable that is valid only for the MCC source file; all MCC charts (programs, function blocks, and functions) within the MCC source file can access the instance.
- In the declaration table of the MCC chart (for programs and function blocks only):
  The instance behaves like a local variable; it can only be accessed within the MCC chart in which it is declared.

Proceed as follows; the MCC source file or the MCC chart with the declaration table is open, see "Open existing MCC source files" (Page 3989) or "Open existing MCC chart" (Page 4004):

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Parameters** tab.
3. Enter or select the following:
   - **Name** of instance (variable name – see Rules for identifiers (Page 4047))
   - **Variable type** VAR or VAR_GLOBAL, depending on the declaration location (in MCC chart or MCC source file, respectively)
   - Designation of the function block as **data type**.
4. Declare the other variables.



① The output parameter Difference is assigned to the variable Result_2 during subsequent program runtime. You can use the Result_2 variable for other purposes in the program.

② The output parameter Difference is assigned to the variable Result in the subroutine call. You can use the Result variable for other purposes in the program.

③ Creating an instance

④ Select the required FB as the data type.

The created function blocks are offered as data types in the drop-down list box depending on the MCC editor settings (Page 3984):

- Only function blocks with the same program source or from connected program sources or libraries
- All function blocks defined in the project

Figure 7-59 Define an instance of the function block and variables in the MCC chart or MCC source file

Programming the subroutine call of the function block

1. Insert the Subroutine Call command in the MCC chart.
2. Program the command as follows:
   - For subroutine type, select **Function block**.
   - Select the designation of the function block as the subroutine.
     The input, in/out parameters and output parameters for the FB are displayed.
   - Enter the instance defined in the declaration table in the **Instance** field.
3. Assign the current values to the transfer parameters:
   - Input parameters: Variable or expression
   - In/out parameter: Directly readable/writable variable
   - Output parameter (optional): Variable

   You can paste unit and system variables from the detail view using a drag-and-drop operation.

   Note

   Mandatory parameters are marked with "<???>" and optional parameters with "...". Function blocks (FBs) only have optional parameters.

4. Click **OK** to confirm.

Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the MCC charts in an MCC unit. A subroutine (function or function block) must be defined before it is used. This is the case when the MCC chart of the subroutine appears above the chart in which it is used in the project navigator. If necessary, reorder the charts.

See also: Subprogram call of function (FC) (Page 4129)

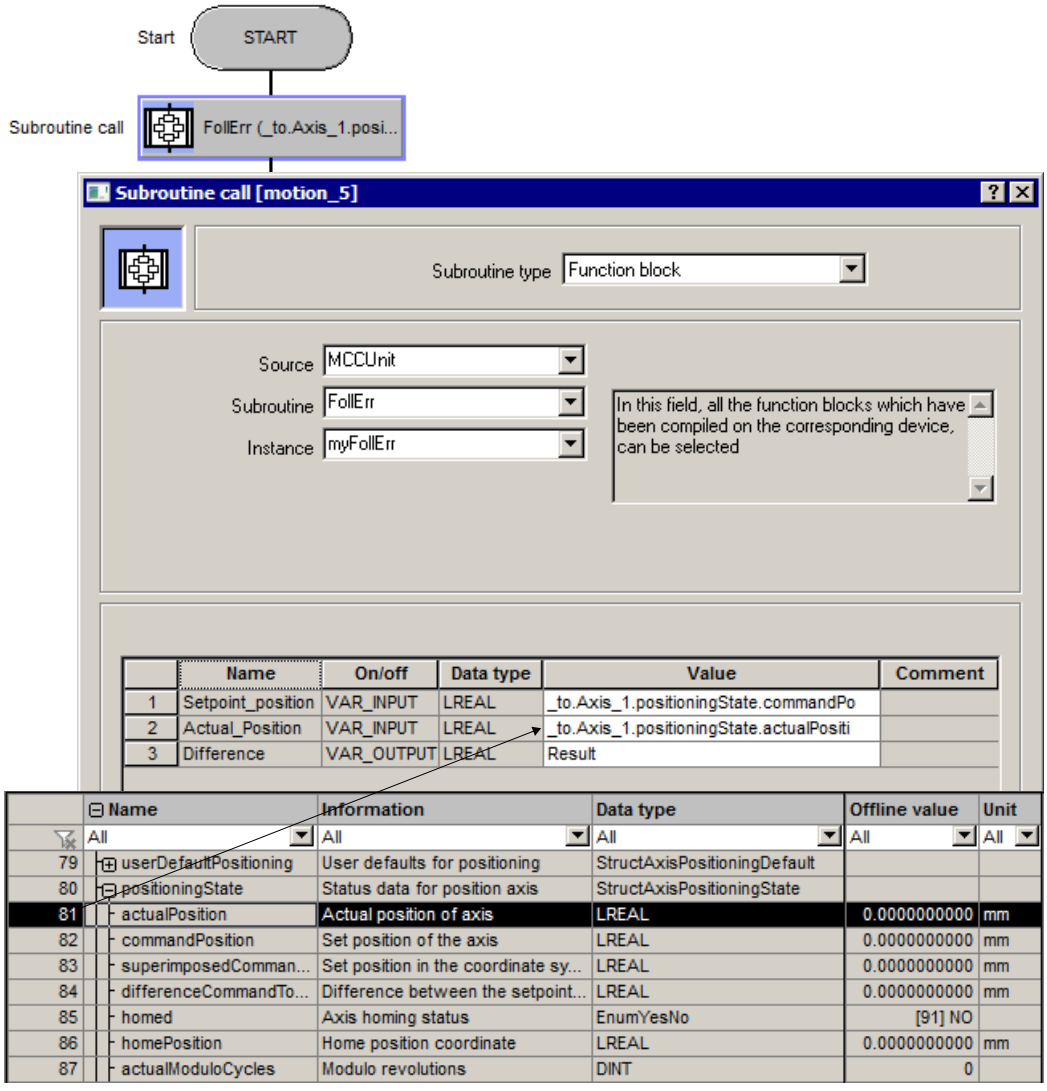


Figure 7-60 Opened parameter screen form for the subroutine call

Opening the function block (FB) directly following the subroutine call

You can open subprograms directly from their respective subprogram call using the context menu:

- This works regardless of the programming language (ST, MCC, LAD/FBD) used to create the subprogram.
- The subprogram opens in a separate editor or an editor already opened for the subprogram is moved to the foreground.

A subprogram may be a:

- User-defined FC
- User-defined FB
- User-defined program called as a subprogram ("program in program")
- Library function
- Library function block
- Library program called as a subprogram ("program in program")
- Class method
- Interface method

Procedure

To open the function block (FB) directly following the subprogram call, proceed as follows:

1. In the MCC chart, select the **Subprogram call** command used to call the function block (FB).
2. Select the **Open called block** command in the context menu (Ctrl+Alt+O hotkey).
   The function block (FB) opens in a separate editor, or an editor already opened for the function block (FB) is moved to the foreground.

Note

If the function block (FB) called has not yet been created, the **Open called block** command appears inactive (grayed out) in the context menu.

Note

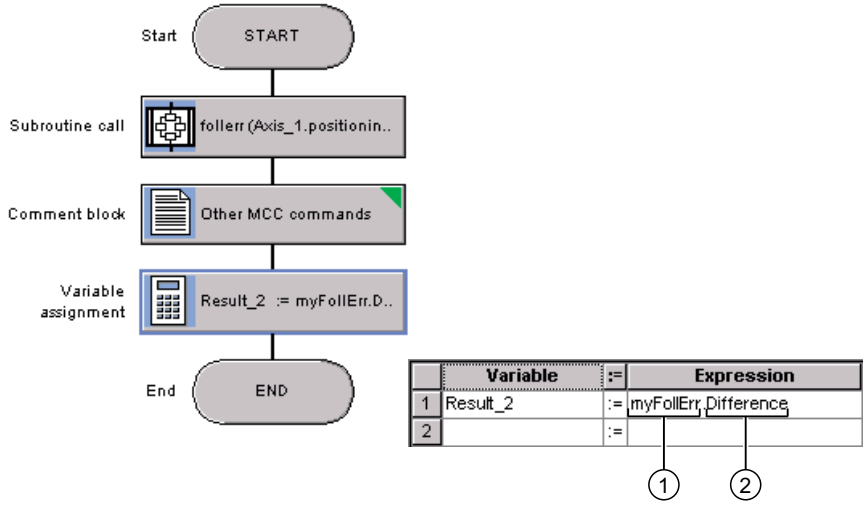
If the function block (FB) called is in a program source with know-how protection, the steps required when opening the protected program source directly must also be performed here. See Know-how protection for MCC units (Page 3992).

Accessing the output parameters of the function block retrospectively

After an instance of the function block has been executed, the static variables of the function block (including the output parameters) are retained. You can access the output parameters at any point in the calling program.

If you have defined the FB instance as VAR_GLOBAL, you can also access the output parameter in other MCC charts.

1. Insert the **Variable assignment** command in the MCC chart.
2. Program the command (see figure).
3. Click **OK** to confirm.



① Name of the FB instance ② Output parameter

Figure 7-61 Programming a variable assignment

Example: Method

In MCC charts you can call up public methods which have been defined in ST source files within classes and function blocks.

You can use methods in function blocks irrespective of the SIMOTION Kernel version.

Methods in classes can only be used in SIMOTION Kernel as of version V4.5.

No classes or methods can be defined in the MCC programming language itself. The same applies to methods within function blocks.

Requirement

In an ST source file the COUNTER class is defined with both public methods UP and DOWN: Both methods increment or decrement an internal counter variable until either the maximum value or minimum value is reached.

- The UP method has the following interface:
  - Input parameter INC with data type INT and initial value 1.
    You enter the increment here (standard = 1).
  - Output parameter QU with data type BOOL and initial value FALSE.
    The maximum value is reached when QU = TRUE.
  - Return value with data type INT.
    The return value shows the latest status for the internal counter variables.
- The DOWN method has the following interface:
  - Input parameter DEC with data type INT and initial value 1.
    You enter the increment here (standard = 1).
  - Output parameter QU with data type BOOL and initial value TRUE.
    The minimum value is reached when QU = FALSE.
  - Return value with data type INT.
    The return value shows the latest status for the internal counter variables.

The maximum value and minimum value for the counting range are stipulated in the private variables MAX_Val and MIN_Val for the class. They are pre-assigned with 100 or 0 respectively as standard. However, the initial values can be overwritten with the declaration of an instance for this class.

Subprogram call of the method

The method is called from a program in the example.

1. Generate an MCC chart as a program, see Insert new MCC chart (Page 4001).
2. Create an instance for the higher-level class or function block.
   - In the MCC source file or MCC chart, declare the instance of the class or the function block along with the variables, see: Creating an instance for the class or the function block (Page 4140).

   Note that the validity range of the instance and variables is dependent on the declaration location (see Define variables (Page 4062)).
3. Call the method:
   - Program the **Subprogram call** command, see: Programming the subprogram call of the method (Page 4141).
4. Accept and compile the program.

You have now finished programming the subroutine call.

Creating an instance for the class or the function block

Before you can use a method, you must define an instance for the higher-level class or the higher-level function block. Each instance of a class or FB is independent of the others; once an instance has ended, its static variables remain stored.

You define the instances of a class or FB in the declaration table of the MCC source file or MCC chart. The scope of the instance declaration is dependent on the location of the declaration:

- In the interface section of the declaration table of the MCC source file:
  The instance behaves like a unit variable; it is valid for the entire MCC source file; all MCC charts (programs, function blocks, and functions) within the MCC source file can access the instance.
  The instance is also available:
  - On HMI devices.
  - In other MCC source files (or other units) once connected, see "How to define connections to other units (program sources)" (Page 4115).

  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the MCC source file:
  The instance behaves like a unit variable that is valid only for the MCC source file; all MCC charts (programs, function blocks, and functions) within the MCC source file can access the instance.
- In the declaration table of the MCC chart (for programs and function blocks only):
  The instance behaves like a local variable; it can only be accessed within the MCC chart in which it is declared.

Proceed as follows; the MCC source file or the MCC chart with the declaration table is open, see "Open existing MCC source files" (Page 3989) or "Open existing MCC chart" (Page 4004):

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Parameters** tab.
3. Enter or select the following:
   - **Name** of instance (variable name – see Rules for identifiers (Page 4047))
   - **Variable type** VAR or VAR_GLOBAL, depending on the declaration location (in MCC chart or MCC source file, respectively)
   - Designation of the function block as **data type**.
4. Declare the other variables.

Programming the subprogram call of the method

1. Insert the Subroutine Call command in the MCC chart.
2. Program the command as follows:
   - Select **Method** as the subprogram type.
     The classes or function blocks exported from ST source files are displayed with their public methods in the following format: *Class_Name.Method_Name* or FB_name.Methods
   - Select the designation of the class with its method as the subprogram.
     The input, in/out parameters and output parameters for the method are displayed.
   - Enter the instance for the higher-level class or the higher-level function block defined in the declaration table in the **Instance** field.
   - Enter the variable assigned the return value after the method has ended in the **Return value** field.

3. Assign the current values to the transfer parameters:
   - Input parameters: Variable or expression
   - In/out parameter: Directly readable/writable variable
   - Output parameter (optional): Variable

   You can paste unit and system variables from the detail view using a drag-and-drop operation.

Note

Mandatory parameters are marked with "<???>" and optional parameters with "...".

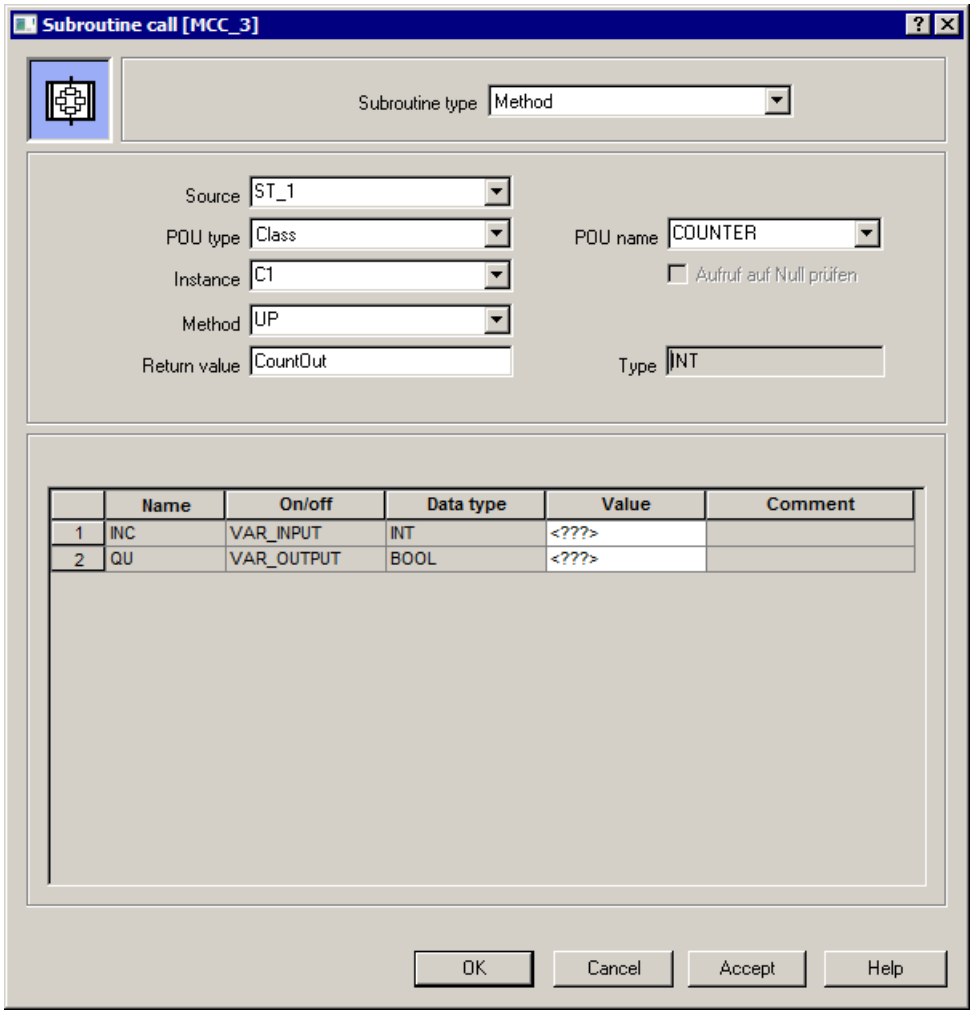


Figure 7-62 Subprogram call for a method

4. Click **OK** to confirm.

Note

Saving a project in a format older than Version V4.5 of SIMOTION SCOUT

Projects in which classes and their methods are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

The following program uses the COUNTER program which was declared in the ST source file ST_1, see requirements in "Example:Methods" (Page 4138). It calls the methods UP and DOWN defined in this class. A C1 instance for the COUNTER class was declared for this; provided that the upper value is not reached (UpValreached = FALSE), the Method COUNTER.UP is called, otherwise the Method COUNTER.DOWN is called until the lower limit value is reached.

This program must be assigned to a cyclic task of the execution system (e.g.: BackgroundTask).

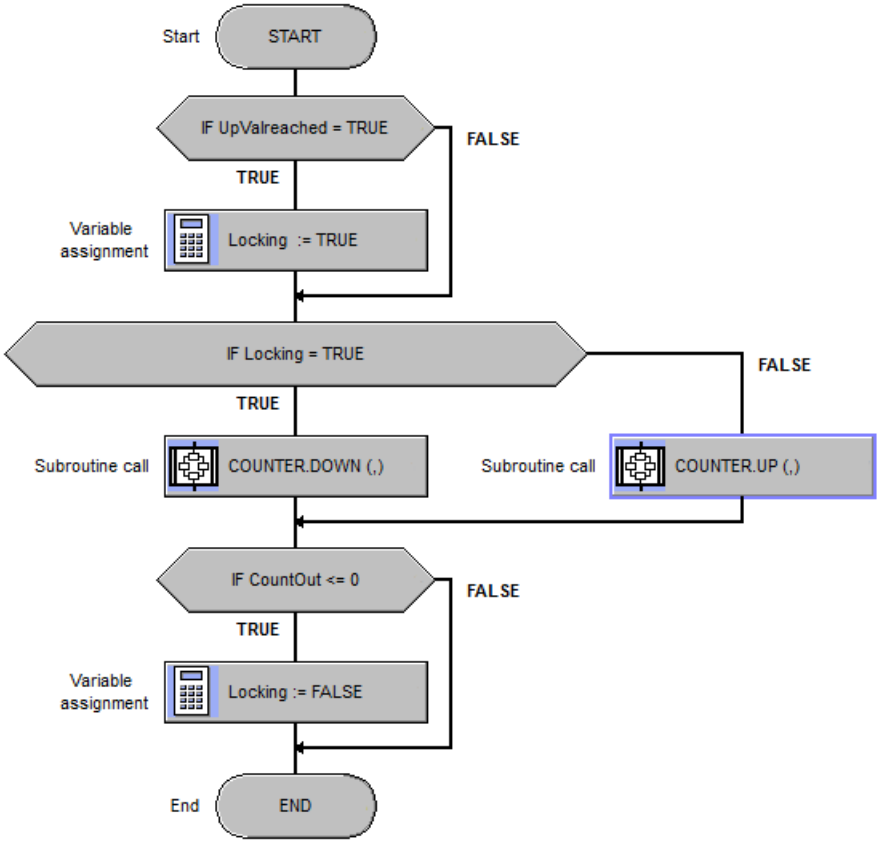


Figure 7-63 Example: Call of the methods in the COUNTER class

7.1.5.11 Reference data

The reference data provide you with an overview of:

- on utilized identifiers with information about their declaration and use (Cross-reference list (Page 4144)).
- on function calls and their nesting (Program structure (Page 4148))
- on the memory requirement for various data areas of the program sources (Code attributes (Page 4150))

Cross reference list

The cross-reference list shows all identifiers in program sources (e.g. ST source files, MCC units):

- Declared as variables, data types, or program organization units (program, function, function block)
- Used as previously defined types in declarations
- Used as variables in the statement section of a program organization unit.

Generating and updating a cross-reference list

Initially, the cross-reference list is generated automatically when opening. Open a cross-reference list, e.g. after selecting an ST source file or library via the **Edit > Display reference data > Cross-references** menu. After changes, the update is partly performed automatically and can always be triggered manually in the detail view via the  button. When updating via the button, a check is performed as to whether a compilation is required. If a compilation is required, this is indicated by a yellow triangle next to the button.

Update of the cross-reference list when selecting a program

The list is updated automatically when opening the selected program. The local defined identifiers are therefore up-to-date.

External identifiers in the program are not updated when opening.

The opened cross-reference list is also updated automatically when compiling the program.

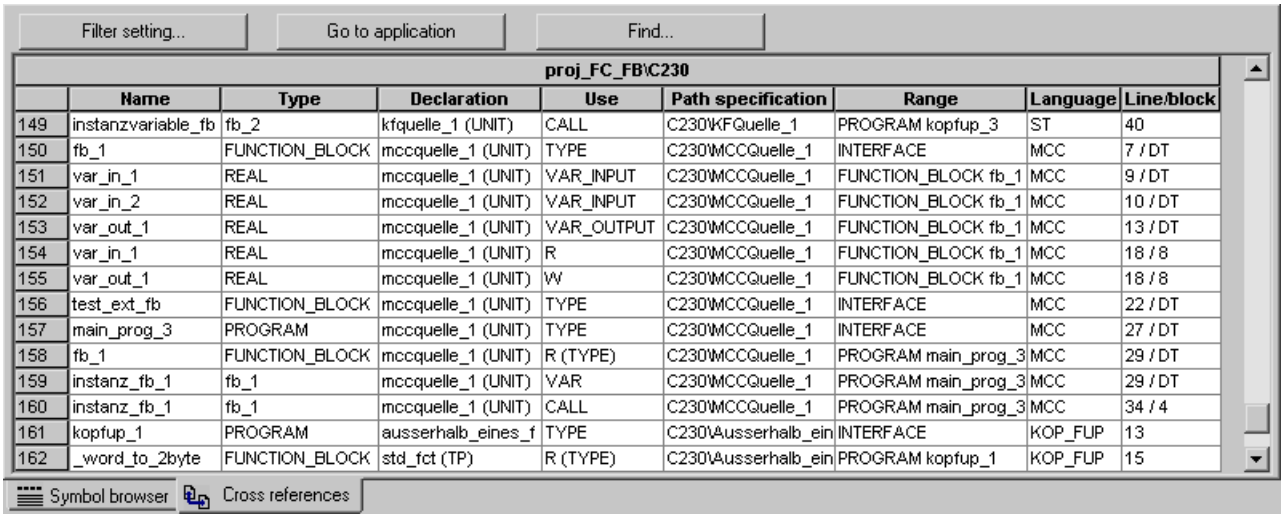
Update of the cross-reference list when selecting a tree (CPU, project, library, program folder)

The cross-reference list is generated automatically when opening the first time. There is no automatic update when opened again.

Note

An error-free compilation is required for a correct, consistent display of the reference data. If required, compile the project, the CPU, the program or the library first.

Example: Cross-reference tab in detail view



	Name	Type	Declaration	Use	Path specification	Range	Language	Line/block
149	instanzvariable_fb	fb_2	kfquelle_1 (UNIT)	CALL	C230\KFQuelle_1	PROGRAM kopfup_3	ST	40
150	fb_1	FUNCTION_BLOCK	mccquelle_1 (UNIT)	TYPE	C230\MCCQuelle_1	INTERFACE	MCC	7 / DT
151	var_in_1	REAL	mccquelle_1 (UNIT)	VAR_INPUT	C230\MCCQuelle_1	FUNCTION_BLOCK fb_1	MCC	9 / DT
152	var_in_2	REAL	mccquelle_1 (UNIT)	VAR_INPUT	C230\MCCQuelle_1	FUNCTION_BLOCK fb_1	MCC	10 / DT
153	var_out_1	REAL	mccquelle_1 (UNIT)	VAR_OUTPUT	C230\MCCQuelle_1	FUNCTION_BLOCK fb_1	MCC	13 / DT
154	var_in_1	REAL	mccquelle_1 (UNIT)	R	C230\MCCQuelle_1	FUNCTION_BLOCK fb_1	MCC	18 / 8
155	var_out_1	REAL	mccquelle_1 (UNIT)	W	C230\MCCQuelle_1	FUNCTION_BLOCK fb_1	MCC	18 / 8
156	test_ext_fb	FUNCTION_BLOCK	mccquelle_1 (UNIT)	TYPE	C230\MCCQuelle_1	INTERFACE	MCC	22 / DT
157	main_prog_3	PROGRAM	mccquelle_1 (UNIT)	TYPE	C230\MCCQuelle_1	INTERFACE	MCC	27 / DT
158	fb_1	FUNCTION_BLOCK	mccquelle_1 (UNIT)	R (TYPE)	C230\MCCQuelle_1	PROGRAM main_prog_3	MCC	29 / DT
159	instanz_fb_1	fb_1	mccquelle_1 (UNIT)	VAR	C230\MCCQuelle_1	PROGRAM main_prog_3	MCC	29 / DT
160	instanz_fb_1	fb_1	mccquelle_1 (UNIT)	CALL	C230\MCCQuelle_1	PROGRAM main_prog_3	MCC	34 / 4
161	kopfup_1	PROGRAM	ausserhalb_eines_f	TYPE	C230\Ausserhalb_ein	INTERFACE	KOP_FUP	13
162	_word_to_2byte	FUNCTION_BLOCK	std_fct (TP)	R (TYPE)	C230\Ausserhalb_ein	PROGRAM kopfup_1	KOP_FUP	15

Figure 7-64 Cross-reference tab in detail view

Content of the cross-reference list

The cross-reference list contains all the identifiers assigned to the element selected in the project navigator. The applications for the identifiers are also listed in a table:

Details of how to work with the cross-reference list are provided in the section titled "Working with the cross-reference list".

Table 7-38 Meanings of columns and selected entries in the cross-reference list

Column	Entry in column	Meaning
Name		Identifier name
Type		Identifier type
	Name	• Data type of a variable (e.g. REAL, INT) • POU type (e.g. PROGRAM, FUNCTION)
	DERIVED	Derived data type
	DERIVED ANY_OBJECT	TO data type
	ARRAY ...	ARRAY data type
	ENUM ...	Enumerator data type
	STRUCT ...	STRUCT data type
Declaration		Location of declaration
	Name (UNIT)	Declaration in the program source *name*
	Name (LIB)	Declaration in the library *name*
	Name (TO)	System variable of the technology object *name*
	Name (TP)	Declaration in the default library specified: • Technology package *name* • std_fct = IEC library • device = device-specific library

Column	Entry in column	Meaning
	Name (DV)	Declaration on the SIMOTION device *name* (e.g. I/O variable or global device variable)
	_project	Declaration in the project (e.g. technology object)
	_device	Internal variable on the SIMOTION device (e.g. TaskStartInfo)
	_task	Task in the execution system
Use		Use of identifier
	CALL	Call as subprogram (static binding)
	CALL VTAB	Call of a method of the dynamic binding
	ENUM *name*	As element when declaring the enumerator data type *name*
	I/O	Declaration as I/O variable
	R	Read access
	R (TYPE)	As data type in a declaration
	R/W	Read and write access
	STRUCT *name*	As component when declaring the structure *name*
	TYPE	Declaration as data type or POU
	Variable type (e.g. VAR, VAR_GLOBAL)	Declaration as variable of the variable type specified
	W	Write access
Path specification		Path specification for the SIMOTION device or program source
	Name	SIMOTION device *name*
	Name1/*Name2*	• Program source *name2* on SIMOTION device *name1* • Program source *name2* in library *name1*
	Name/taskbind.hid	Execution system of the SIMOTION device *name*
Range		Range within the SIMOTION device or program source
	INTERFACE	Interface section of the program source
	POE type name (i.e. CLASS *name*, FUNCTION *name*, FUNCTION_BLOCK *name* INTERFACE *name* PROGRAM *name*)	Program Organization Unit (POU) *Name* within the program source. • In an MCC chart: Additional serial numbers for the command (block numbers) • In a LAD/FBD program: Additional serial numbers of the network
	I/O address	I/O variable
	METHOD *Name_1*::*Name_2*	Method *Name_2* within the class or the function block *Name_1*
	TASK *name*	Assignment for the task *name*
	UNIT	Implementation section of the program source, additional specification as POU to be made public in the interface section of the program source
	UNIT - IMPLEMENTATION	Implementation section of the program source
	_device	Global device variable

Column	Entry in column	Meaning
Language		Programming language of the program source
Line/Block		• In an ST source: Line number within the program source • In an MCC or LAD/FBD source: Relative line number within the command (block) or network. **Note** The absolute line number within the program source, which you need, for example, for the trace function "Trigger to code point", is obtained as follows: – Press the **Determine program line** button. – In the dialog box, press the button **Copy program line to the clipboard**.

Note

Single-step tracking and trace diagnostic functions in MCC programming

Additional variables and functions are created or used for these diagnostics functions:

- The variables TSI#dwuser_1 and TSI#dwuser_2 of the TaskStartInfo are used for the single-step tracking diagnostic function.
- Various internal functions and variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. The TSI#currentTaskId variable of the TaskStartInfo is also used.

With activated diagnostic function, these variables and functions are used for the control of the diagnostics function. These variables and functions must not be used in the user program.

Working with a cross-reference list

In the cross-reference list you are able to:

- Sort the column contents alphabetically:
  - To do this, click the header of the appropriate column.
- Search for an identifier or entry:
  - Click the "Search" button and enter the search term.
- Filter (Page 4148) the identifiers and entries displayed.
- Copy contents to the clipboard in order to paste them into a spreadsheet program, for example.
  - Select the appropriate lines and columns.
  - Press the CTRL+C shortcut.

- Print the content (**Project > Print**).
- Open the referenced program source and position the cursor on the relevant line of the ST source file (or MCC command or LAD/FBD element):
  - Double-click on the corresponding line in the cross-reference list.
    or
  - Place the cursor in the corresponding line of the cross-reference list and click the "Go to application" button.

Further details about working with cross-reference lists can be found in the online help.

Filtering the cross-reference list

You can filter the entries in the cross-reference list so that only relevant entries are displayed:

1. Click the "Filter settings" button.
   The "Filter Setting for Cross References" window will appear.
2. Activate the "Filter active" checkbox.
3. If you also want to display system variables and system functions:
   - Deactivate the "Display user-defined variables only" checkbox.
4. Set the desired filter criterion for the relevant columns:
   - Select the relevant entry from the drop-down list box or enter the criterion.
   - If you want to search for a character string within an entry: Deactivate the "Whole words only" checkbox.
5. Confirm with **OK**.

The contents of the cross-reference list will reflect the filter settings selected.

Note

A filter is automatically activated after the cross-reference list has been created.

Program structure

The program structure contains all the function calls and their nesting within a selected element.

You can display the program structure selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the program structure.
2. Select the **Edit > Display reference data > Program structure** menu command.
   The cross-reference tab is replaced by the program structure tab in the detail view.

Note

The display data is updated every time the program structure is opened.

You can update the detail view of an opened program structure with the F5 key.

Example: Program Structure tab in the detail view

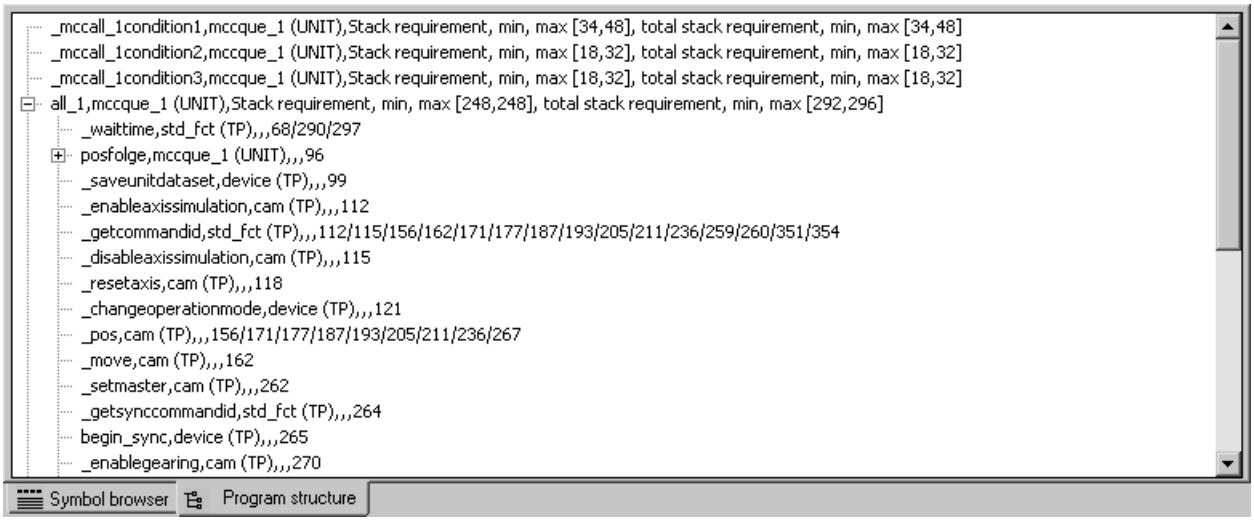


Figure 7-65 Example of a program structure

Content of the program structure

A tree structure appears, showing:

- as base respectively
  - the program organization units (programs, functions, function blocks) declared in the program source, or
  - the execution system tasks used
- below these, the subroutines referenced in this program organization unit or task.

For structure of the entries, see table:

Table 7-39 Elements of the display for the program structure

Element	Description
Base (declared POU or task used))	List separated by a comma • Identifier of the program organization unit (POU) or task The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Identifier of the program source in which the POU or task was declared, with add-on [UNIT] • Minimum and maximum stack requirement (memory requirement of the POU or task on the local data stack), in bytes [Min, Max] • Minimum and maximum overall stack requirement (memory requirement of the POU or task on the local data stack including all called POUs), in bytes [Min, Max]
Referenced POU	List separated by a comma: • Identifier of called POU The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Optionally: Identifier of the program source / technology package in which the POU was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library Add-on (TP): System function from technology package • with function blocks or methods only: Identifier of the instance of the function block or the higher-level class • with function blocks or methods only: Identifier of program source in which the instance of the function block or of the higher-level class was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library • Number of the line in the (compiled) source in which the POE is called; several line numbers are separated by "/".

Code attributes

You can find information on or the memory requirement of various data areas of the program sources under code attribute.

You can display the code attributes selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the code attributes.
2. Select the **Edit > Display reference data > Code attributes** menu command.
   The **Cross-references** tab is now replaced by the **Code attributes** tab in the detail view.

Note

The display data is updated every time the code attributes are opened.

You can update the detail view of the opened code attributes with the F5 key.

Code attribute contents

The following are displayed in a table for all selected program sources:

- Identifier of program source,
- Memory requirement, in bytes, for the following data areas of the program source:
  - **Dynamic data**: All unit variables (retentive and non-retentive, in the interface and implementation sections),
  - **Retain data**: Retentive unit variables in the interface and implementation section,
  - **Interface data**: Unit variables (retentive and non-retentive) in the interface section,
- the **Code size** during the last compilation in bytes,
- the **Number of referenced sources**:
  The maximum number of connected sources is displayed (including system libraries), regardless of whether they are downloaded to the target system at a later date.

Reference to variables

If you have selected the identifier of a variable in the open Editor window for each programming language, you can use the other places of use over the context menu to list or to skip these variables.

You select the identifier of a variable:

- In SIMOTION ST: In the Editor window of an ST source.
- In SIMOTION MCC: In the input field on the parameter screen of an open MCC command within an MCC chart
- In SIMOTION LAD/FBD: In the Editor window of a LAD/FBD program

An identifier is recognized as a variable under the following conditions:

1. The identifier is declared as a variable. The scope of the variable includes the respective window (ST source, MCC chart, LAD/FBD program).
2. The program source is compiled.
3. The variable is selected as follows (in an open parameter screen within an MCC chart):
   – The identifier is fully marked
   or
   – The cursor is within the identifier.

Note

In arrays and structures, only the variable can be selected, not a single element.

Using the **Go to** context menu, you have the following options:

- To jump to the next local place of use:
  Select the context menu **Go to > Local use >>**.
  The next place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the previous local place of use:
  Select the context menu **Go to > Local use <<**.
  The previous place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the declaration position:
  Select the context menu **Go to > Declaration position**.
  The declaration position of the variables is selected. The corresponding program source is opened, if necessary.
- List all places of use
  Select the context menu **Go to > Places of use ...**.
  In the detailed view, all places of use of the variables within their scope (including the declaration position) are listed The structure of this list is similar to the List of cross references (Page 4145).
  This is how you jump to a preferred place of use:
  – Double-click on the corresponding line.
  or
  – Place the cursor in the corresponding line and click the "Go to application" button.

7.1.5.12 LAD/FBD/formula

The creation language for a condition or variable assignment is either the ladder logic (LAD) or function block diagram (FBD) programming language, or a formula according to IEC61131-3. One of these programming languages can be selected as a standard language for conditions and variable assignments (see MCC editor settings (Page 3984)).

The language can be selected individually for the following commands:

- Variable assignment, see Variable assignment (Page 4187)
- Wait for condition, see Wait for condition (Page 4180)
- IF program branching, see IF: Program branching (Page 4206)
- WHILE loop with condition at the start, see WHILE: Loop with condition at the start (Page 4208)
- UNTIL loop with condition at the end, see UNTIL: Loop with condition at the end (Page 4211)
- Synchronous start, see Synchronous start (Page 4219)

You can switch between the different programming languages. A calculation programmed in LAD or FBD can be expanded in Formula in accordance with IEC61131-3. However, you can then only switch back to the LAD or FBD display if the expression can be displayed in these languages.

Note

If you have not yet programmed in these languages, you can find a brief description of how to do so in Appendix "Basic principles of LAD/FBD/Formula for MCC" (Page 4566).

Insert variable

You can insert variables from the detail view (Symbol browser tab) into the appropriate input field using a drag and drop operation.

Ladder diagram (LAD)

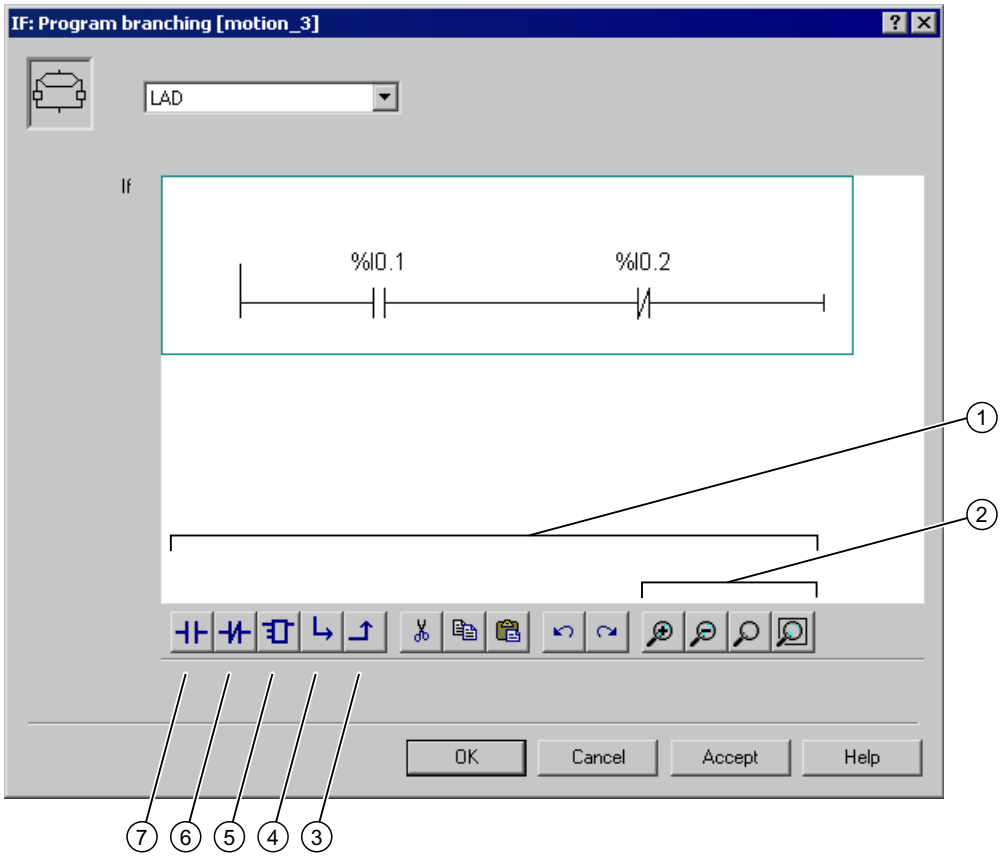
If you select the LAD language, an empty ladder diagram is initially opened.

New elements are inserted using the context menu commands or the buttons on the toolbar. New elements are always inserted after the selected element.

You can select individual elements using the cursor or the keyboard cursor keys. The selected element is displayed in blue.

Selected elements can be copied, pasted, cut, or deleted using the context menu.

Input screen form for ladder diagram



① Toolbar
② Enlarge/reduce the graphic display
③ Close branch
④ Open branch
⑤ Comparator
⑥ NC contact
⑦ NO contact

Figure 7-66 Input screen form for ladder diagram

Description of LAD elements

Description of normally open (NO) contact

NO contact	
Description	Signal = 1 if the operand has signal status 1. The operand specifies the bit or Boolean variable whose signal status is queried.
Data types	BOOL

Description of normally closed (NC) contact

NC contact	
Description	Signal = 1 if the operand has signal status 0. The operand specifies the bit or Boolean variable whose signal status is queried.
Data types	BOOL

Description of comparator

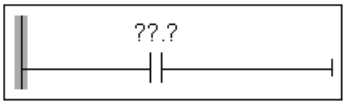
Comparator		
Description	Signal = 1 if the comparison of the two operands is true. Possible modes of comparison:	
	=	Equal to
	<>	Not equal to
	>	Greater than
	<	Less than
	>=	Greater than or equal to
	<=	Less than or equal to
Data types	SINT, INT, DINT, REAL, LREAL, BOOL, BYTE	
	SINT	8-bit integer comparator, Parameter: Byte
	INT	16-bit integer comparator, Parameter: Words
	DINT	32-bit integer comparator and time comparator, Parameter: Double words and times
	REAL	32-bit floating-point comparator, Parameter: Double words
	LREAL	64-bit floating-point comparator, Parameter: 64-bit floating point values

A detailed description of the elements can be found in Appendix "Ladder diagram (LAD) for MCC" (Page 4566).

Opens a branch

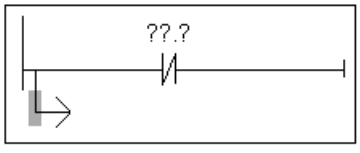
Procedure: Open branch

Use the cursor to select the position at which the branch is to be opened.



Open the branch using 

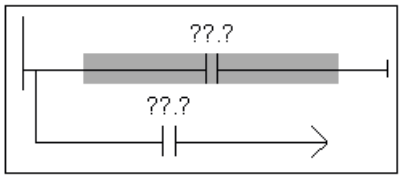
The branch is inserted after the selected element.



Closing a branch

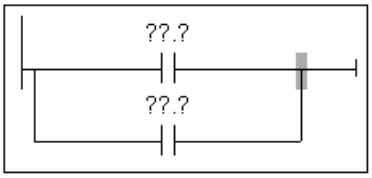
Procedure: Close branch

Use the cursor to select the position at which the branch must be closed.



Close the branch using 

The branch is closed after the selected element.



Function block diagram (FBD)

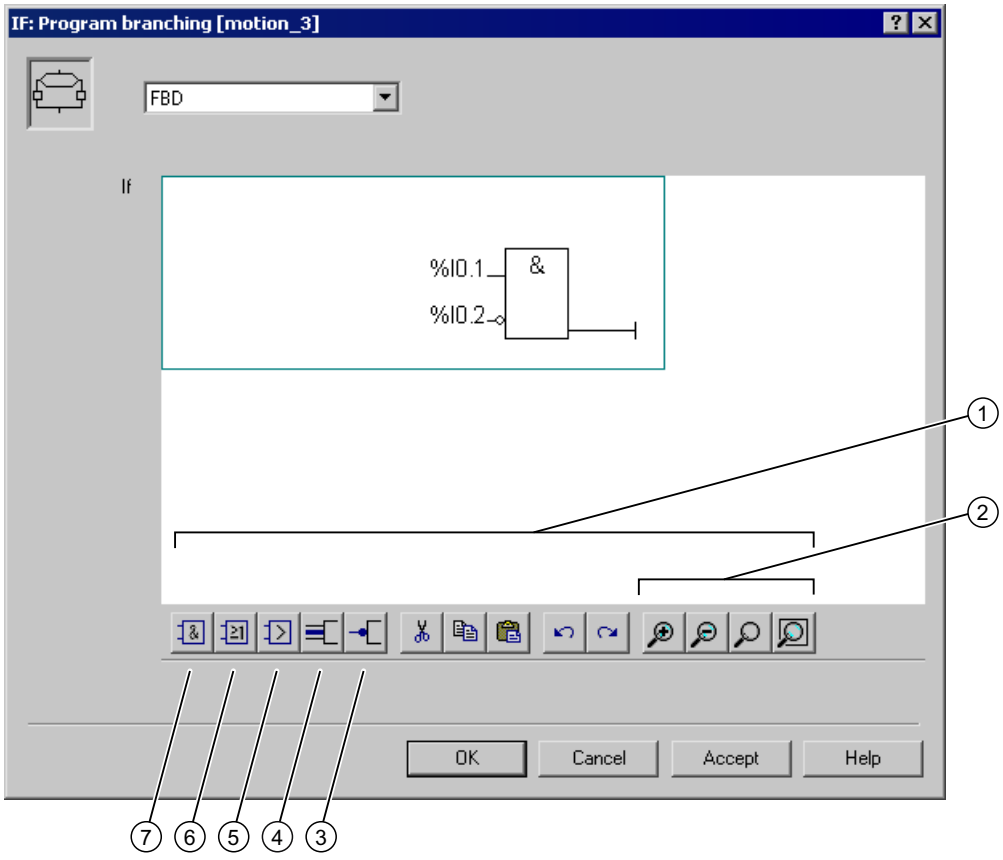
If you select the FBD language, an empty function block diagram is opened first.

New elements are inserted using the context menu commands or the buttons on the toolbar. New elements are always inserted after the selected element.

You can select individual elements using the cursor or the keyboard cursor keys. The selected element is displayed in blue.

Selected elements can be copied, pasted, cut, or deleted using the context menu.

Input screen form for function block diagram



① Toolbar
② Enlarge/reduce the graphic display
③ Negate input
④ Add input
⑤ Comparator
⑥ OR operation
⑦ AND operation

Figure 7-67 Input screen form for function block diagram

Description of FBD elements

Description of an AND operation

AND operation	
Description	The AND operation forms the signal states of two or more specified operands at the inputs of an AND box. If the signal status of all operands is 1, the condition is fulfilled and the result of the operation is 1. If the signal status of one operand is 0, the condition is not fulfilled and the result of the operation is 0.
Data types	BOOL

Description of an OR operation

OR operation	
Description	The OR operation forms the signal states of two or more specified operands at the inputs of an OR box. If the signal status of one operand is 1, the condition is fulfilled and the operation returns a result of 1. If the signal status of all operands is 0, the condition is not fulfilled and the result of the operation is 0.
Data types	BOOL

Description of comparator

Comparator	
Description	See Description of LAD elements (Page 4154).
Data types	See Description of LAD elements (Page 4154).

Description of Add input

Add input	
Description	An additional input is added to an AND or an OR operation.
Data types	----

Description of Negate input

Negate input	
Description	Signal is negated.
Data types	----

A detailed description of the elements can be found in Appendix "Function block diagram (FBD) for MCC" (Page 4569).

Formula

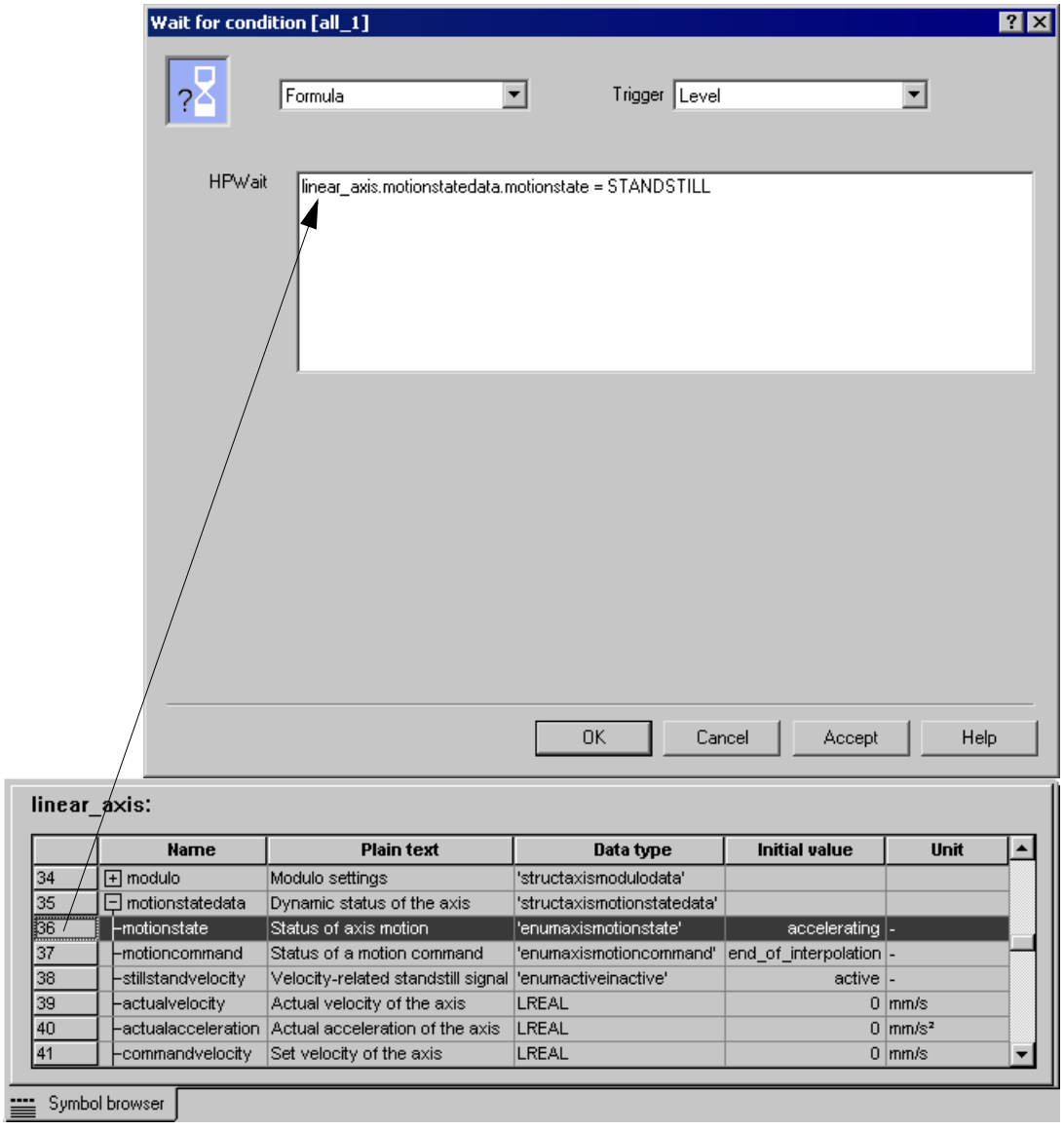
A condition or value can be programmed in Formula language.

You can move the operators you need from the Command library tab in the project navigator to the input field using a drag and drop operation.

The system variables are moved from the Symbol browser tab in the detail view to the input field using a drag and drop operation (see figure).

Closed block comments (* ... *) are permitted, line comments (starting with //) not. Use the command comment on the MCC command (**Enter comment ...** context menu), see also "Display of commands in the MCC chart" (Page 4016).

Programming a condition in the Formula language



A detailed description can be found in Appendix "Formula for MCC" (Page 4573).

7.1.5.13 Command library and system function

Command library

The command library is automatically displayed as a project navigator tab if you program the following commands:

- ST zoom
- System function call

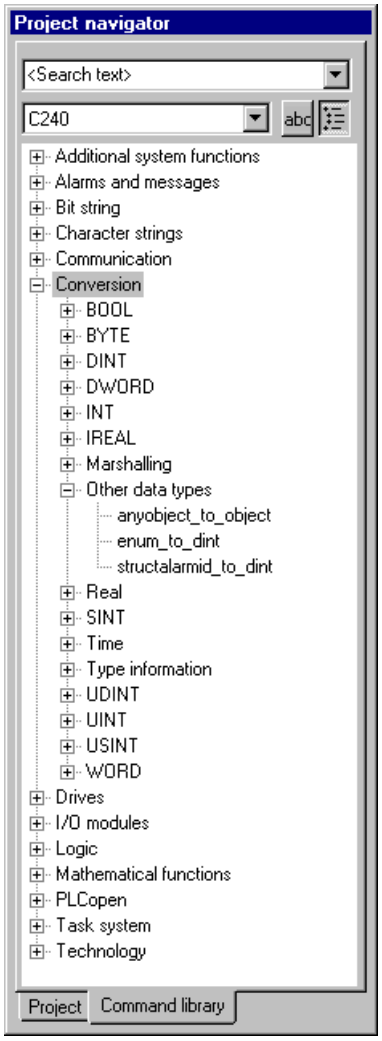


Figure 7-68 Command library tab of the project navigator

Note

You can move system functions from the command library to the parameter screen forms of the "ST zoom" and "System function call" commands using a drag-and-drop operation.

The command library stays open even when the programming window is closed.

Using the command library

Proceed as follows to insert operands or functions from the command library into a programming window:

Move the entry from the command library to the input field of the window using a drag-and-drop operation.

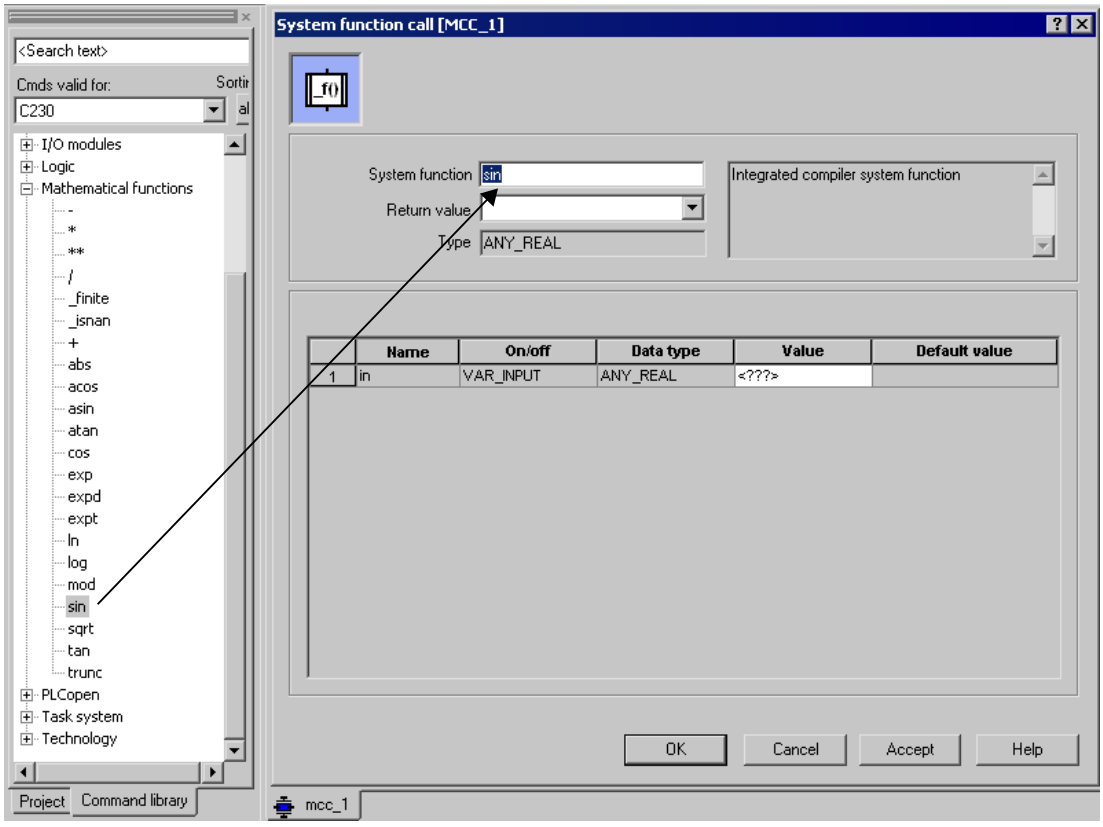


Figure 7-69 Inserting a command from the command library

Using the System function call command

The System function call command is available on the MCC editor toolbar (Basic commands command list). This command enables you to use any system functions and system function blocks in the MCC chart and to program their call conveniently.

When the command is called, the programmed system function or system function block is executed. Once the system function or system function block is executed, the execution of the MCC chart resumes after the command.

System function

The system function can be moved from the command library to the input field using a drag and drop operation or can be entered directly.

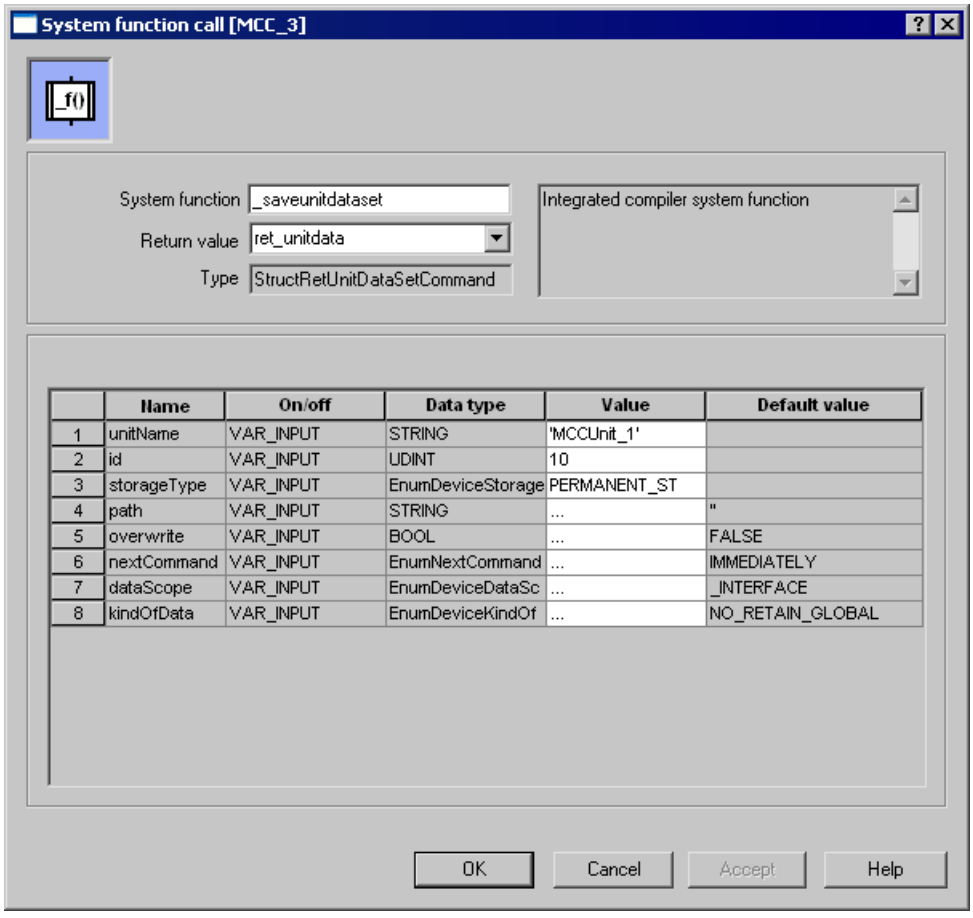
Return value

In the case of a system function, you specify the variable containing the return value of the function.

Instance

In the case of a system function block, you specify the instance. This instance has been declared as a variable beforehand (data type is the name of the system function block). See also Creating a function block instance.

Example of System function call



See also

Overview of parameters for System function call (Page 4163)

Overview of parameters for System function call

You can set the following parameters:

Table 7-40 Overview of parameters for System function call

Field/button	Meaning/instruction
System function	The desired system function can be selected from the command library. You can either enter the system function in the field or move it from the project navigator (command library) to the field using a drag and drop operation.
Return value	For a system function: Here, you enter the variable in which the return value is to be stored. The type of variable must match the return value type.
Type	For a system function: The data type of the return value is displayed.
Instance	For a system function block: Here, you enter the name of the system function block instance. The instance contains the memory of the function block in the form of instance data. Define the instance as VAR_GLOBAL in the declaration table of the MCC unit or VAR in the declaration table of the MCC chart.
List of transfer parameters	
Name	The name of the transfer parameter is displayed here.
On/off	The variable type of the transfer parameter is displayed here. **VAR_INPUT** Input parameter (for system functions and system function blocks) **VAR_IN_OUT** In/out parameter (for system function blocks only) **VAR_OUTPUT** Output parameter (for system function blocks only)
Data type	The data type of the transfer parameter is displayed here.

Field/button	Meaning/instruction
Value	Mandatory parameters are marked with "<???>" and optional parameters with "...". A system function call will only be functional if all the mandatory parameters are set. Here, you can assign current variables or values to the transfer parameters: • Input parameter (variable type VAR_INPUT): Here, you enter a variable name or an expression. The assignment of system variables or I/O variables is permissible; type transformations are possible. Optional input parameters are indicated by an entry in the **Default value** column. If no value has been transferred to it, it is automatically assigned the default value. • In/out parameter (variable type VAR_IN_OUT – for system function blocks only): Enter a variable name; it must be directly writable and readable. System variables of SIMOTION devices and technology objects are not permitted nor are I/O variables. The data type of the in/out parameter must correspond to that of the assigned variables; application of type transformation functions is not possible. • Output parameter (variable type VAR_OUTPUT – for system function blocks only): The assignment of an output parameter to a variable in this parameter screen form is optional; you can also access an output parameter after executing the function block. When assigned in this parameter screen form: Enter a variable name. The data type of the output parameter must correspond to that of the assigned variables; the application of type transformation functions is not possible.
Default value	Here, the default value is displayed for optional input parameters. If no value has been transferred to an optional input parameter, it is automatically assigned the default value.

7.1.5.14 MCC charts in libraries

Libraries provide you with user-defined types, unit variables, functions, and function blocks which you can use from any SIMOTION device.

Libraries can be written to in all available programming languages. They can be used in all program sources (e.g., MCC source files, ST source files).

In order to use a library in an MCC source file, you must link the library in the declaration table of the MCC source file (with specification of a name space, if necessary) (see How to define connections to libraries (Page 4116)).

You can obtain more details on inserting and managing libraries in the online help.

Using technology packages in libraries

Libraries can also contain the following:

- Commands that act on technology objects.
- Access to system variables of technology objects.

Note

Only variables of this data type (see Technology object data types (Page 4060)) are permitted as technology objects.

You specify the SIMOTION device and technology package for which the library is compiled in the object properties of the library:

1. Select the library in the project navigator.
2. Select the **Edit > Object Properties** menu command.
3. Then select the **Technology package** tab.
4. Select the SIMOTION devices (including the version number) and the technology packages for which the library is to be compiled.

Note

To compile a project without errors, observe the rules governing the selection of SIMOTION devices and technology packages in the following table.

Selection of devices and technology packages in a library

Selection	Description
Device-independent	You must also select: • The technology packages • The version number of the selected technology packages **Please note:** 1. The library is compiled without reference to a SIMOTION device. For this reason, the following must not be used: – System functions and system variables of SIMOTION devices – Version-dependent system functions of the SIMOTION ST programming language (see SIMOTION ST Programming Manual) Additional MCC commands (Page 4167) are not permitted. 2. The library is compiled precisely to the version selected. The use of system functions or variables which are not available in the selected version will result in a compilation error. 3. If a device-independent library is to be made available for another version it must be copied and inserted under a different name. This copy must be recompiled with a different version reference.
SIMOTION device including version (multiple selection possible)	Only those technology packages are displayed that are available for all of the selected devices. **Please note:** 1. The library is compiled for all of the selected devices and technology packages (of the selected device versions). 2. The use of system functions or variables which are not available for one of the selected devices, or the technology package of the respective device version, will result in a compilation error. 3. The library can only be used for the selected devices and technology packages. When you use the library in an MCC unit, the following is therefore checked: – Whether the library for the SIMOTION device (including version) that contains the MCC unit to be imported has been compiled. – Whether the technology package that is used on the SIMOTION device matches that of the library. Any inconsistencies will result in compilation errors.

Compiling a library

In libraries, you can use all MCC commands except for the ones listed in the table below. In addition, you are not permitted to access some variables; these variables are also listed in this table.

MCC commands and variable accesses not permitted in libraries

Table 7-41 Illegal MCC commands in libraries

In all libraries:

- Following MCC task command:
  - Determine TaskId (Page 4205)

See the SIMOTION ST Programming and Operating Manual.

Additional commands, if the library was compiled on a **device-independent** basis:

- Following MCC basic commands:
  - Change operating state (Page 4196)
  - Activate trace (Page 4197)
- All MCC task commands (Page 4198)
- Following MCC command for program structure:
  - Synchronous start (Page 4219)
- Following MCC commands for communication:
  - Incoming message (Page 4228)
  - Outgoing message (Page 4231)
  - Establish connection using TCP/IP (Page 4233)
  - Remove connection using TCP/IP (Page 4237)
  - Send data (Page 4239)
  - Receive data (Page 4246)
- System functions of SIMOTION devices (see List Manual for SIMOTION Devices).

Table 7-42 Illegal variable access in libraries

- Unit variables (retentive and non-retentive)
- Global device variables (retentive and non-retentive)
- I/O variables
- Instances of the technology objects and their system variables
- Variables of task names and configured messages (for name spaces _task and _alarm)
- If the library is not device-dependent (i.e. compiled without reference to a SIMOTION device or SIMOTION Kernel version):
  - System variables of SIMOTION devices (see List Manuals for SIMOTION Devices)
  - Configuration data of technology objects (see List Manual of Configuration Data for the Relevant SIMOTION Technology Package)

Compile a library as follows:

1. Select the library in the project navigator.
2. Select the SIMOTION devices and the technology packages for which the library is to be compiled (Menu **Edit > Object properties**, **Technology packages** tab - see Using technology packages in libraries (Page 4165))
3. Select the **Accept and compile** context menu.

7.1.5.15 Find and replace

The following options are available when searching for a given piece of text or for variables only:

- Find or find and replace in the entire project (project-wide search).
  This is described in detail in the Online help.

  Note

  Following a find and replace operation in the whole project, it may be necessary to compile the project, so as to update all symbol information.

- Find or find and replace in a certain program source or their subprograms (local search).
  The following is described in this manual:
  - Searching in an MCC unit or an MCC chart (Page 4168)
  - Searching and replacing in an MCC unit or an MCC chart (Page 4170)

  See the respective manuals for a description of the local search in the other programming languages.

Searching in an MCC unit or an MCC chart

Range of the local search

With local searching, only the contents of the window (MCC unit or MCC chart) in which the local search was triggered are searched.

- The following are searched in an MCC unit:
  - All tabs of the declaration table (INTERFACE and IMPLEMENTATION)

  The assigned MCC charts are **not** searched.
- The following are searched in an MCC chart:
  - All tabs of the declaration table
  - MCC commands (brief comments, input fields and selection lists of the parameter screen form, command comments)

The dialog box open during the local search is assigned to the respective window and is hidden when changing to another window and displayed again when returning.

Procedure

If you want to conduct local searching for arbitrary text in an MCC unit or MCC chart, proceed as follows:

1. Open the desired MCC unit (Page 3989) or the MCC chart (Page 4004).
2. In the menu, select **Edit > Find** (shortcut Ctrl+F).
   The **Find** dialog box opens.
   A character string selected in the MCC editor is automatically taken as a search term.
3. Enter the required search term in the **Find** input field.
   Wildcards such as "*" or "?" are not permitted.
4. If required, select a search option as well as the search direction (**Up**/**Down**).
   - If you activate the **Whole words only** checkbox, only a whole word is searched for.
   - If you activate the **Match case** checkbox, the search takes upper-/lower-case into account.
   - If you activate the **Variables only** checkbox, the search is performed only in fields that contain identifiers of variables.
5. Click the **Find next** button (shortcut F3).
   The search is started in the selected direction: The first matching text pattern is highlighted.
6. Click the **Find next** button again to display the next matching text pattern.

Note

You can also continue searching with F3 even when the dialog box is closed.

Displaying search results

- The relevant tab is automatically activated in a declaration table and the found search term is highlighted.
- The found search term is highlighted in the brief comment for an MCC command.
- The following applies for the parameter screen form of an MCC command:
  - The relevant tab is activated automatically. The found search term is highlighted in the input fields (not in selection lists).
  - If necessary, the parameter screen form of an MCC command that is not open is opened automatically and closed again at the first search result outside the MCC command. Changes to the command must be confirmed manually (**OK** or **Accept**).
- The following applies for the command comment of an MCC command:
  The **Enter Comment** dialog box is opened automatically and the found search term highlighted. When the first search result occurs outside the command comment, the dialog box closes automatically once more and the changes are saved.

The respective editor window can also be edited when the Find dialog box is open.

Finding and replacing in an MCC source file or an MCC chart

Introduction

The process of finding and replacing on a local basis works like local finding (Page 4168), but also makes it possible to specify an expression to replace a search term once found.

Procedure

If you want to conduct a local search for any text and replace it in an MCC source file or an MCC chart, proceed as follows:

1. Open the desired MCC source file (Page 3989) or the MCC chart (Page 4004).
2. In the menu, select **Edit > Replace** (shortcut Ctrl+H).
   The **Replace** dialog box opens.
   A character string selected in the MCC editor is automatically taken as a search term.
3. Enter the required search term in the **Find** input field.
   Wildcards such as "*" or "?" are not permitted.

   Note

   If you select a term in the MCC editor before opening the **Replace** dialog box, this is entered in the **Find** input field automatically.

4. If required, select a search option.
   - If you activate the **Whole words only** checkbox, the search looks for a whole word.
   - If the **Match case** checkbox is activated, the search takes account of uppercase and lowercase.
   - If you activate the **Variables only** checkbox, the search is performed only in fields that contain identifiers of variables.
5. Enter the expression that should replace the search term in the **Replace with** input field.
6. Click the **Find next** button (shortcut F3).
   The search begins in the **Down** direction and the first matching text pattern is highlighted. The **Replace** button also becomes active if a replacement is permitted at this position, see rules for replacing.
   - If you want to replace the search term found, click the **Replace** button.
     The search term found, now selected and displayed, is replaced and the next length of text to match the criteria is displayed.
   - If you do not want to replace the search term found, click the **Find next** button.
     The next length of text to match the criteria is displayed.

   With parameter screen forms, any replacements are taken over in the command. The syntax is checked when the parameter screen form is automatically closed. Errors are displayed in the detail view in the **Local replacement** tab. Also in the event of errors during the syntax check, the MCC command is closed and the local search continued.

Rules for replacing

When replacing, a check is made whether the resulting term after replacement is permitted in principle at this position, for example:

- For identifiers of variables and data types, a check is made whether the Rules for identifiers (Page 4047) have been observed.
- For selection fields (combo boxes), a check is made whether the resulting term is available as a selection option. Some examples are shown below:
  - In the declaration table of an MCC source file, VAR_GLOBAL cannot be replaced by VAR or VAR_TEMP.
  - A selection option in a selection field (combo box) in the parameter screen for an MCC command can only be replaced by a different selection option.
- Wildcards, such as " ... " or "???" generally cannot be replaced.

Note

If an item cannot be replaced at a particular position, the **Replace** button appears inactive (grayed out).

Further checks are not made, for example

- Whether variables or data types have already been defined.
- Whether there are further dependencies between fields and will be violated by the replacement.
- Whether the changes have an effect on the design of the parameter screen form (e.g. display or selection options for parameters).

In declaration tables, some columns are generally blocked for replacing, for example:

- The **Absolute identifier** column in the **I/O symbols** tab
- The **Type** column in the **Connections** tab

7.1.5.16 Print

The MCC source file or MCC chart displayed in the working window can be output to a printer.

1. Select the **Project > Print** menu command.
2. In the **Print project** window, select the print options (see table below):
   - When printing an MCC source file:
     Select print options for this MCC source file and each MCC chart it contains.
   - When printing an MCC chart:
     Select print options for this MCC chart.
3. Then, select:
   - **Print** if you want to output the result to the printer immediately
   - **Print preview** if you want to check the print result on the screen before printing

Print options for an MCC source file or MCC chart

Option	Meaning
Printing a declaration table (MCC source file and MCC chart)	When this check box is activated, the declaration table of the MCC source file or the MCC chart is printed. In the Select column below, select: **Default column widths** The table will be printed with the default column widths. **Scale column widths to screen** The table is printed with the column widths that were set in the declaration table window
Print chart (MCC chart only)	When this check box is activated, the selected MCC chart will be printed. Depending on the selected zoom factor, the printout can have multiple pages.
Printing the index page (MCC chart only)	When this check box is activated, an index page displaying the entire MCC chart on one page and the page boundaries of the individual pages is also printed. If the printout of the MCC chart consists of several pages, this will make it easier to read.
Display numbers of adjoining pages (MCC chart only)	When this check box is activated, the number of the next page will be printed on the margin of each page of a multi-page MCC chart printout.

Option	Meaning
Zoom (MCC chart only)	Select a zoom factor for the printout of the MCC chart: **Scale graphics to page width** The width of the MCC chart is scaled to the page width. Depending on the size, the chart may be divided vertically onto several pages. **Scale graphics to page height** The height of the MCC chart is scaled to the page height. Depending on the size, the chart may be divided horizontally onto several pages. **Scale graphics to one page** The height and width of the MCC chart are scaled to the page size. The chart is always presented on one page. **Graphic at 100%** The MCC chart will be printed in its original size. The chart can be divided onto several pages, both vertically and horizontally. **Apply zoom factor from screen** The MCC chart is printed with the zoom factor setting for its MCC editor window. The chart can be divided onto several pages, both vertically and horizontally.
Blank pages (MCC chart only)	If the printout of the MCC chart consists of several pages, blank pages can occur. In the Select column, select which blank pages should be printed: **Print all** All blank pages are printed. **Omit at end** Blank pages at the end of a series are not printed. If the Index page option is selected, the non printed pages are marked with an X on the index page. **Omit all** All blank pages (even in the middle of a series) are not printed. If the Index page option is selected, the non printed pages are marked with an X on the index page.

7.1.6 MCC commands

7.1.6.1 Basic commands

Wait time

Execution of the task is halted for the programmed time period.

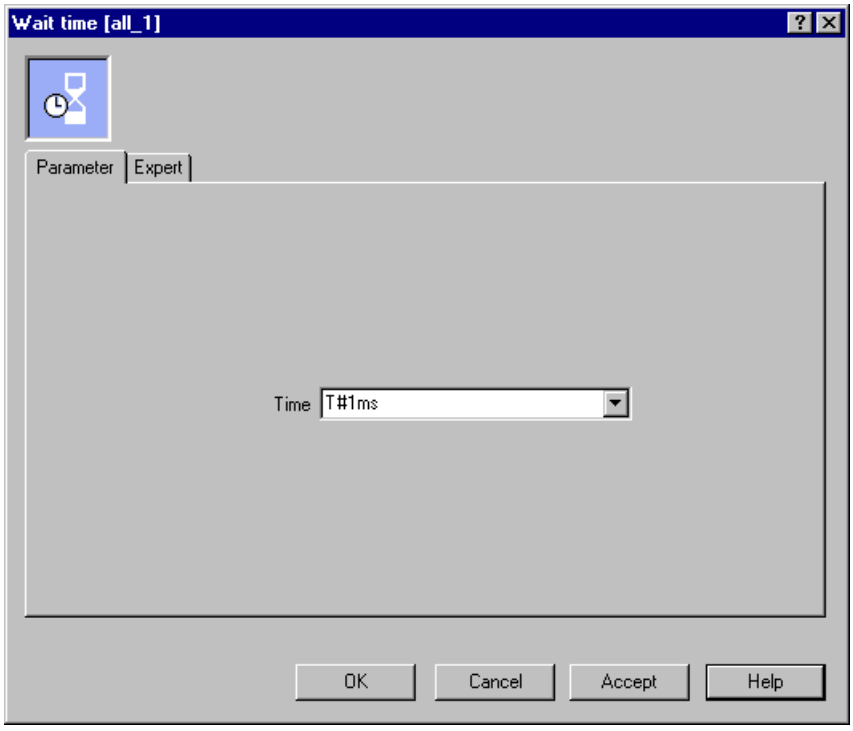


Figure 7-70 Parameter screen form: Wait time

Note

The command should be used in MotionTasks only; using it in cyclic tasks may lead to time monitoring errors!

- With SynchronousTasks: You can configure whether the time watchdog is suspended. Time monitoring is active by default.
  With IPOsynchronousTask, additionally take the following into account: UserInterruptTasks will no longer be started by their triggering event!
- With other cyclic tasks (BackgroundTask, TimerInterruptTasks): Time monitoring is always active.

Overview of parameters for Wait time

You can set the following parameters:

Table 7-43 Overview of parameters for Wait time

Field/Button	Explanation/instructions
Parameters tab	See Overview of parameters for Wait time - Parameters tab (Page 4175)
Expert **tab**	See Overview of parameters for Wait time - Expert tab (Page 4175)

Overview of parameters for Wait time - Parameters tab

Table 7-44 Overview of parameters for Wait time - Parameters tab

Field/Button	Explanation/instructions
Time	Enter the wait time here as a value or variable of type TIME. Granularity is 1 ms. The accuracy with which the time setting is applied is dependent on the clock grid in which the task is executed or, in the case of MotionTasks, on the interpolation cycle clock. If the wait time is 0, the task is continued without any interruption. **T#0ms** (default value) **T#0d_0h_0m_0s_0ms**

Overview of parameters for Wait time - Expert tab

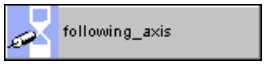
Table 7-45 Overview of parameters for Wait time - Expert tab

Field/Button	Explanation/instructions
	The **Expert** tab is described in detail in Overview of parameters for Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT, the value is always equal to 0.

Relevant system functions for Wait time

ST system function: _waitTime.

Wait for axis



Execution of the task is halted until the axis fulfills the programmed condition. Possible conditions are:

- **Axis status**: A status of the axis determined on the basis of system variables of the axis or the associated synchronous object (in the case of a synchronized axis)
- **Comparison value** : Comparison of a value of the axis (such as position, velocity) with a specified value

If both conditions are selected, they are combined using a logical AND.

The statuses and values of the axis relate to the total motion.

The condition is checked in the interpolator cycle clock. If the condition is fulfilled, the priority of the MotionTask is temporarily raised. The commands within the shaded area under the wait command are started in the next IPO cycle clock and executed with the highest priority.

Once the commands in the shaded area have been executed, the priority of the MotionTask is reset.

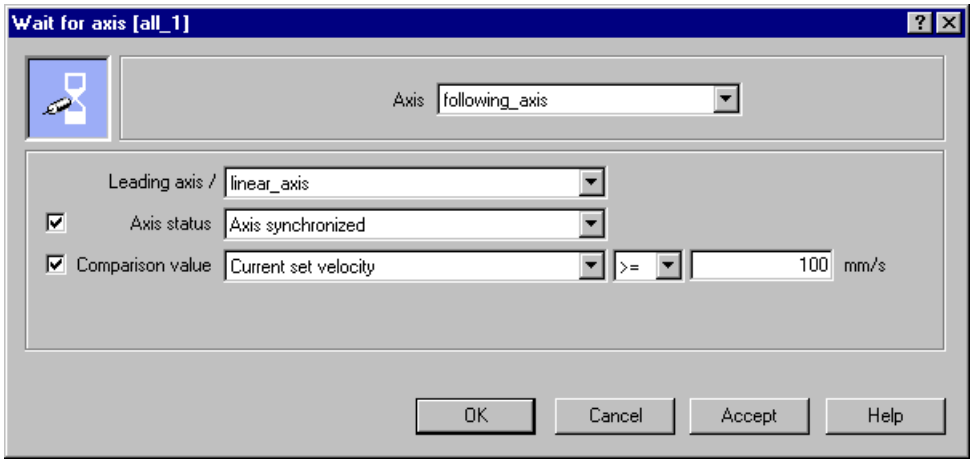


Figure 7-71 Parameter screen form: Wait for axis

Note

This command may only be programmed for MotionTasks.

Note

If you want to program a wait condition with axis statuses or axis values other than those provided in the selection lists:

Use the **Wait for condition** command and the corresponding system variables of the axis (see SIMOTION Cam Technology Package, System Variables Parameter Manual).

Overview of parameters for Wait for axis

You can set the following parameters:

Table 7-46 Overview of parameters for Wait for axis

Field/button	Meaning/instruction
Axis	Here, select the axis whose status or value you want to check. You can select from: • All axes that are configured on the device • All variables of the MCC chart with the following data types of technology objects, see Data types of technology objects (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Leading axis / encoder	Visible for all synchronous axes. **Note** The **Leading axis / encoder** selection is required only for synchronous axes with more than one synchronous object (superimposed synchronous motions). This input is ignored in the case of synchronous axes with only one synchronous object. By making this selection, you are indirectly selecting the synchronous object of the main motion or the superimposed motion.

Field/button	Meaning/instruction
Axis status	Here, you select an axis status as the condition • Select this checkbox if you want to evaluate the axis status. • Select the axis status. Different options are offered depending on the technology of the selected axis, see Axis status table (Page 4177).
Comparison value	Here, you select the comparison of an axis value as the condition. • Select this checkbox if you wish to compare an axis value. Possible axis values are shown in the subsequent selection list. • Select the axis value that you want to compare. Different options are offered depending on the technology of the selected axis. • Select the comparison operator, see Comparison operators for axis values table (Page 4178). • Enter the comparison value.
Tolerance	Only for comparisons for equality (=) or inequality (<>): Maximum absolute value of the permitted difference between the selected axis value and the programmed comparison value that will allow the equality condition to be satisfied.

Axis status

Axis status	Technology of the axis[1]	System variable and value
Axis moving at constant velocity	D, P, F	motionStateData.motionState = CONSTANT_MOVE
Axis in motion	D, P, F	motionStateData.motionCommand = IN_MOTION
Axis homed	P, F	positioningState.homed = YES
Axis at standstill (default value)	D, P, F	motionStateData.motionState = STANDSTILL Standstill monitoring is activated
Axis synchronized	G	syncState = YES (synchronous object)
Accelerating	D, P, F	motionStateData.motionState = ACCELERATING
Motion is finished	D, P, F	motionStateData.motionCommand = MOTION_DONE Axis is in the tolerance window and standstill monitoring is activated
Rotating	D, P, F	moveCommand.state <> INACTIVE The **Start axis position-controlled** or **Speed specification** command is active on the axis.
Synchronous operation active	G	state = GEARING (synchronous object) The **Gearing on** command is active on the axis.
Cam active	G	state = CAMMING (synchronous object) The **Cam on** command is active on the axis.
Quick stop active	D, P, F	stopEmergencyCommand = ACTIVE
Positioning	P, F	posCommand.state <> INACTIVE The **Position axis** command is active on the axis.
Simulation active	D, P, F	simulation = ACTIVE

Axis status	Technology of the axis[1]	System variable and value
Decelerating	D, P, F	motionStateData.motionState = DECELERATING
1) Technology of the axis D: Speed-controlled axis (driveAxis) P: Positioning axis (posAxis) G: Synchronous axis (followingAxis)		

Comparison operators for axis values

Operator	Condition satisfied if:
< (default value).	Selected axis value is less than the programmed comparison value.
<=	Selected axis value is less than or equal to the programmed comparison value.
<>	For position values only: The absolute value of the difference between the selected axis value and the programmed comparison value is greater than the specified tolerance.
=	For position values only: The absolute value of the difference between the selected axis value and the programmed comparison value is less than or equal to the specified tolerance.
>	Selected axis value is greater than the programmed comparison value.
>=	Selected axis value is greater than or equal to the programmed comparison value.

Relevant commands of the ST (Structured Text) programming language for Wait for axis

- EXPRESSION/END_EXPRESSION for formulation of the condition
- WAITFORCONDITION/END_WAITFORCONDITION

Wait for signal

Task execution is halted until the signal status at a digital input or output (data type BOOL) fulfills the programmed condition:

- Static 1 (TRUE) or 0 (FALSE)
- Rising or falling edge

The condition is checked in the interpolator cycle clock. If the condition is fulfilled, the priority of the MotionTask is temporarily raised. The commands within the shaded area under the wait command are started in the next IPO cycle clock and executed with the highest priority.

Once the commands in the shaded area have been executed, the priority of the MotionTask is reset.

The relevant digital input/output is specified as follows:

- Direct access and access to the process image of the cyclic task (Page 4094) (always via I/O variables of the BOOL data type).
- Following Access to the fixed process image of the BackgroundTask (Page 4103):
  - Absolute access to the fixed process image of the BackgroundTask (Page 4110) (BOOL data type).
  - Symbolic access to the fixed process image of the BackgroundTask (Page 4111) (BOOL data type that is defined in the interface section of the MCC unit declaration table.

Global device variables (Page 4063) of the BOOL data type can also be specified.

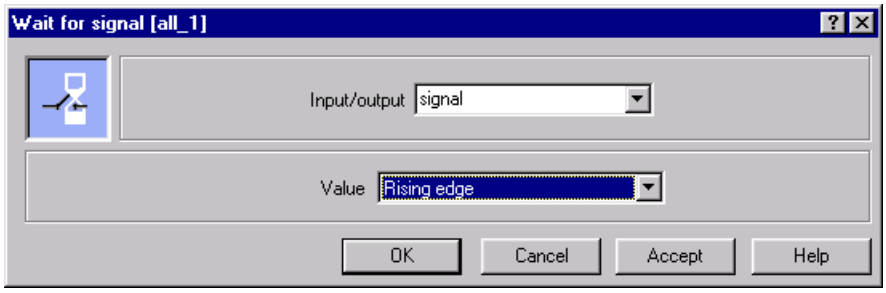


Figure 7-72 Parameter screen form for Wait for signal

Note

This command may only be programmed for MotionTasks.

Note

If you want to program a wait condition with I/O variables or process image accesses whose data type is not BOOL):

Use the "Wait for condition" (Page 4180) command.

Overview of parameters for Wait for signal

You can set the following parameters:

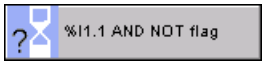
Table 7-47 Overview of parameters

Field/button	Meaning/instruction
Input/output	Here, you select the digital input or output to be evaluated for the wait condition. You can select from the following: • I/O variables (Page 4098) with BOOL data type. • Global device variables (Page 4063) with BOOL data type. • Symbolic access to the fixed process image of the BackgroundTask (Page 4111) with BOOL data type that has been defined in the interface section of the respective MCC unit declaration table. In addition, you can enter absolute process image access (Page 4110) with BOOL data type (e.g. %I10.0).
Value	Here, you can decide whether the condition evaluation is to be edge-triggered or level-triggered. **Rising edge** (default value) The condition is fulfilled if the input changes from 0 (FALSE) to 1 (TRUE). **Falling edge** The condition is fulfilled if the input changes from 1 (TRUE) to 0 (FALSE). **Level 'FALSE'** The condition is fulfilled if the input is static at 0 (FALSE). **Level 'TRUE'** The condition is fulfilled if the input is static at 1 (TRUE).

Relevant commands of the ST (Structured Text) programming language for Wait for signal

- EXPRESSION/END_EXPRESSION for formulation of the condition
- WAITFORCONDITION/END_WAITFORCONDITION

Wait for condition



Execution of the task is halted until the programmed condition is fulfilled.

If the condition is fulfilled, then the priority of the MotionTask is temporarily raised. The commands within the shaded area under the wait command are started in the next IPO cycle clock and executed with the highest priority.

Once the commands in the shaded area have been executed, the priority of the MotionTask is reset.

The condition can be composed of:

- Unit variables of the same MCC unit or imported program sources
- Global device variables
- Constants
- I/O variables and process image accesses (inputs)
- Operators

The expression may not contain:

- Function calls
- Local variables
- Grinding

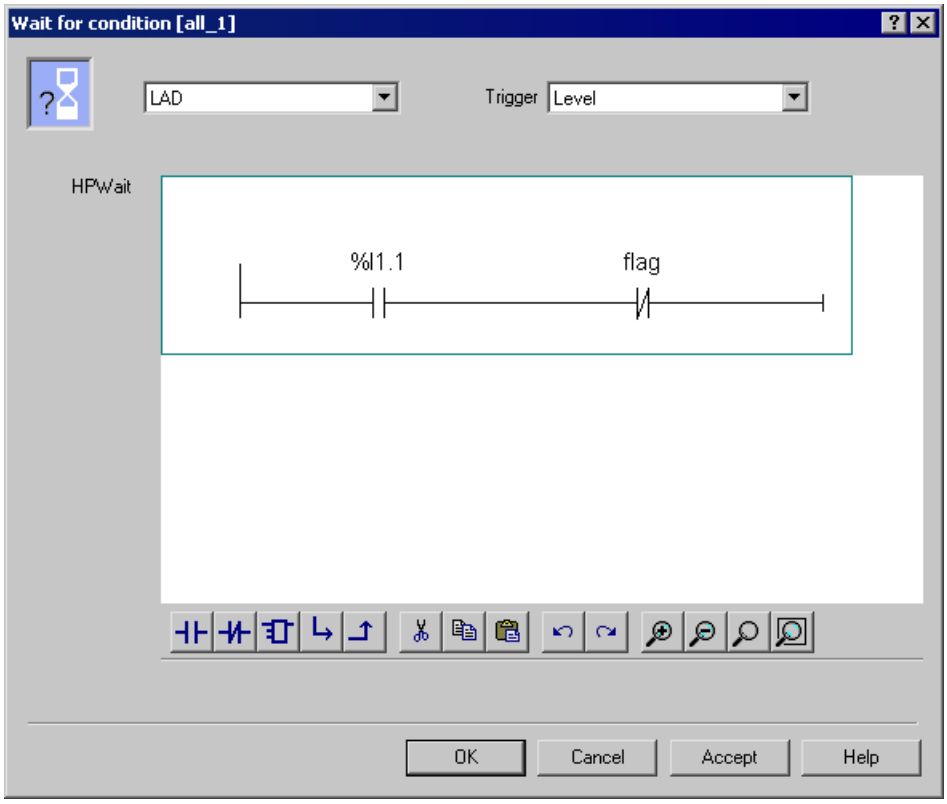


Figure 7-73 Parameter screen form: Wait for condition

Note

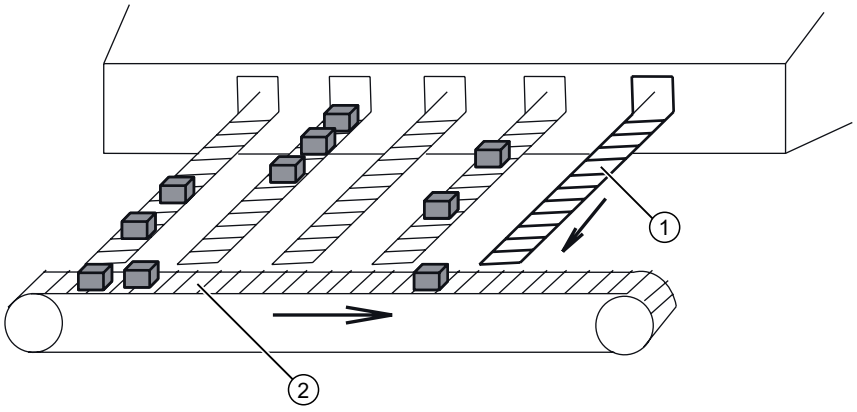
The Wait command may only be programmed for MotionTasks.

The condition is programmed in the ladder diagram (LAD), function block diagram (FBD), or Formula languages.

Example with Wait for condition

Task: A conveyor belt is in operation with several feeders. An additional new feeder must now be installed and programmed.

The packages will be moved onto the conveyor belt only if a package is present on the feeder and there is sufficient room for it on the conveyor belt.



① New feeder: Driven by axis_2

② Conveyor belt: Driven by axis_1

Figure 7-74 Installing and programming a new feeder

Solution

The conveyor belt is driven by axis_1 in position-controlled mode. The feeder unit is driven by axis_2. Two sensors are applied to input1 and input2. These monitor whether a package is on the feeder unit and whether there is sufficient room for it on the conveyor belt. If both are true, the feeder belt is moved (positioned relatively).

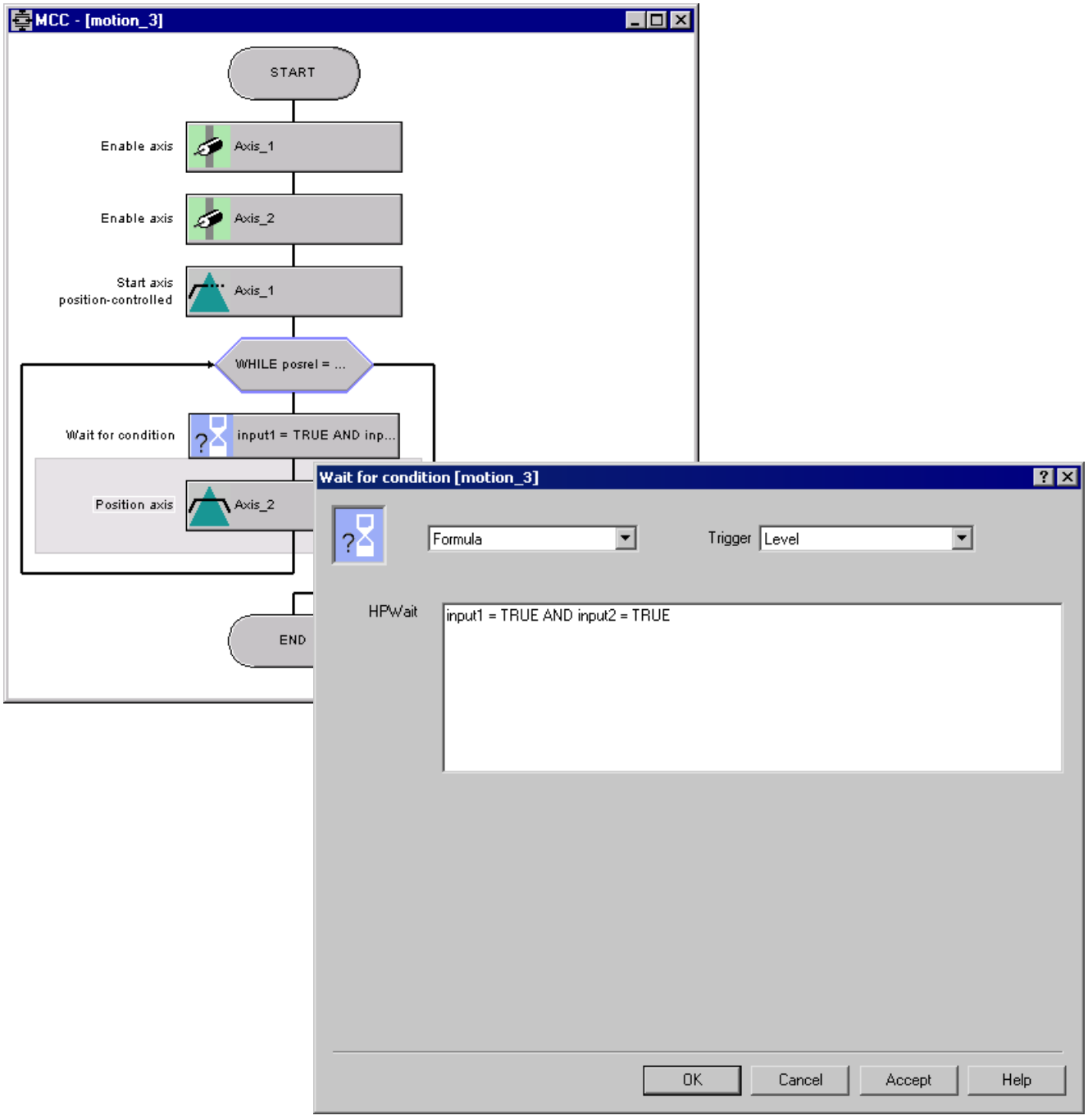


Figure 7-75 Section of MCC chart with open parameter screen form for Wait for condition command; if packages are on feeder belt, WHILE loop is executed (Posrel=True).

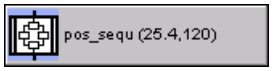
Relevant commands of the ST (Structured Text) programming language for Wait for condition

- EXPRESSION/END_EXPRESSION for formulation of the condition
- WAITFORCONDITION/END_WAITFORCONDITION

Module

Several commands are grouped in a module (see Module creation (Page 4019)).

Subroutine call



The **Subroutine call** command enables you to use user-defined functions and function blocks as well as library functions and library function blocks in the MCC chart and to program their call conveniently (see Insert and parameterize subroutine call in the MCC chart (Page 4121)).

As an option, you can also call user-defined programs ("program in program", see Subroutine (Page 4117)) or library programs.

When the command is called, the programmed function, function block or program is executed. Once the subroutine has been executed, execution of the MCC chart is resumed with the subsequent command.

Marking of mandatory parameters and optional parameters

In the parameter screen form for the command, mandatory parameters are marked with "<???>" and optional parameters with "...". A subroutine call will only be functional if all the mandatory parameters are set.

All the parameters shown are transfer parameters for the function or function block called, with function blocks (FBs) only having optional parameters. Called programs ("program in program") have no transfer parameters.

The following applies to the parameters of a function (FC):

- Transfer parameters without a declared initial value are shown as mandatory parameters
- Transfer parameters with a declared initial value are shown as optional parameters.

Opening user-defined FCs/FBs via the context menu

Regardless of the programming language (ST, MCC, LAD/FBD) in which they were created, called user-defined FCs/FBs/programs ("program in program") and library functions/library function blocks/library programs ("program in program") can be opened in a separate editor via the context menu directly following the subroutine call (see Opening the function (FC) directly following the subroutine call (Page 4131) or Opening the function block (FB) directly following the subroutine call (Page 4137)).

System function call

The **System function call** command enables you to use any system functions and system function blocks in the MCC chart and to program their call conveniently (see Using the system function call command (Page 4161)).

When the command is called, the programmed system function or system function block is executed. Once the system function or system function block is executed, the execution of the MCC chart resumes after the command.

In the parameter screen form for the command, mandatory parameters are marked with "<???>" and optional parameters with "...". A system function call will only be functional if all the mandatory parameters are set.

Set output

This command sets each bit of an output to 1.

Assigned values for Set output command according to data type

Data type	Assigned value	
		(decimal)
BOOL	TRUE	1
BYTE (SINT, USINT)	16#FF	–1 or 255
WORD (INT, UINT)	16#FFFF	–1 or 65535
DWORD (DINT, UDINT)	16#FFFFFFFF	–1 or 4294967295

If you wish to assign another value to the output, you must use the *Variable assignment* command.

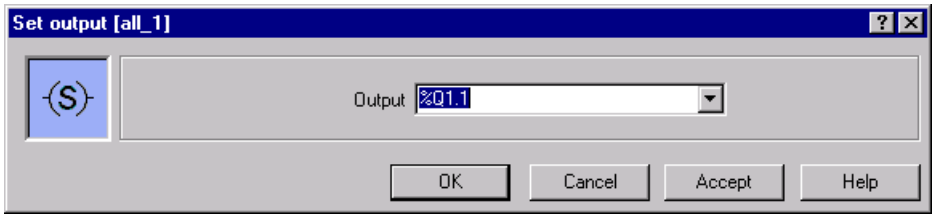


Figure 7-76 Parameter screen form: Set output

Overview of parameters for Set output

You can set the following parameters:

Table 7-48 Overview of parameters for Set output

Field/Button	Explanation/instructions
Output	Here, you select which output is to have its bits set to 1. You can select from the following (outputs only): I/O variables Symbolic accesses to the process image of the BackgroundTask In addition, you can enter absolute process image accesses (outputs). For information about absolute process image access, see Syntax for the identifier for absolute PI access.

Relevant commands of the ST (Structured Text) programming language for Set output

Value assignment to output

Reset output

This command sets each bit of an output to 0.

Table 7-49 Assigned values for Reset output command according to data type

Data type	Assigned value	
		(decimal)
BOOL	FALSE	0
BYTE (SINT, USINT)	16#0	0
WORD (INT, UINT)	16#0	0
DWORD (DINT, UDINT)	16#0	0

If you wish to assign another value to the output, you must use the **Variable assignment** command.

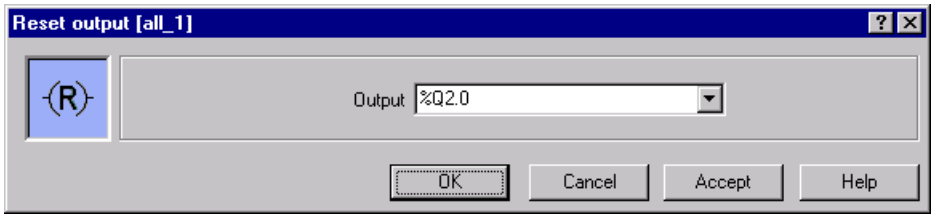


Figure 7-77 Parameter screen form: Reset output parameter

Overview of parameters for Reset output

You can set the following parameters:

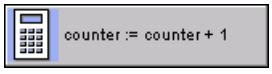
Table 7-50 Overview of parameters for Reset output

Field/Button	Explanation/instructions
Output	Here, you select which output is to have its bits set to 0. You can select from the following (outputs only): • I/O variables • Symbolic accesses to the process image of the BackgroundTask In addition, you can enter absolute process image accesses (outputs).

Relevant commands of the ST (Structured Text) programming language for Reset output

Value assignment to output

Variable assignment



You can use the Variable assignment command to assign values to user or system variables. The assignment takes place when the command is executed in the MCC chart.

A variable can be moved from the symbol browser to the input field with a drag and drop operation. Multiple variables can be assigned.

Note

Assignment is programmed in the ladder diagram (LAD) (Page 4153), function block diagram (FBD) (Page 4156) or Formula (Page 4158) languages.

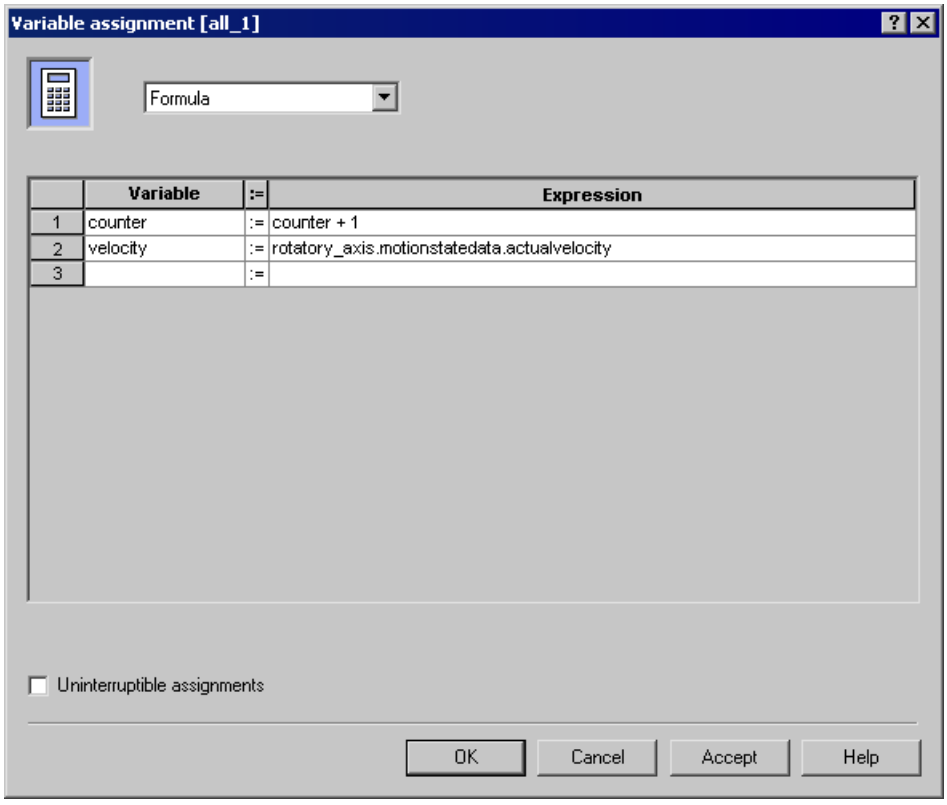


Figure 7-78 Parameter screen form with Formula language

Activate this checkbox if you want the data to be transferred without interruption by other tasks.

Note

If you activate the Uninterruptible assignments checkbox, you must set up UserInterruptTask_1 because the MCC calls this in the event of an error. You can program an error response in this task.

You can assign the following to a variable:

- A value (e.g.: a := 10, b := TRUE)
- Another variable (e.g.: c := d, e := f)
- A system variable (e.g.: axis_1.positioningstate.actualposition)
- An output variable of an FB
- The return value of a function (FC)
- An expression (e.g.: n := 2*h+10*m)

ST zoom

You can use this command to insert one or more ST commands in the MCC chart.

Enter the ST commands in the parameter screen form.

Note

Use shortcut keys Ctrl + TAB to enter a tab in the screen form.

Example with ST zoom

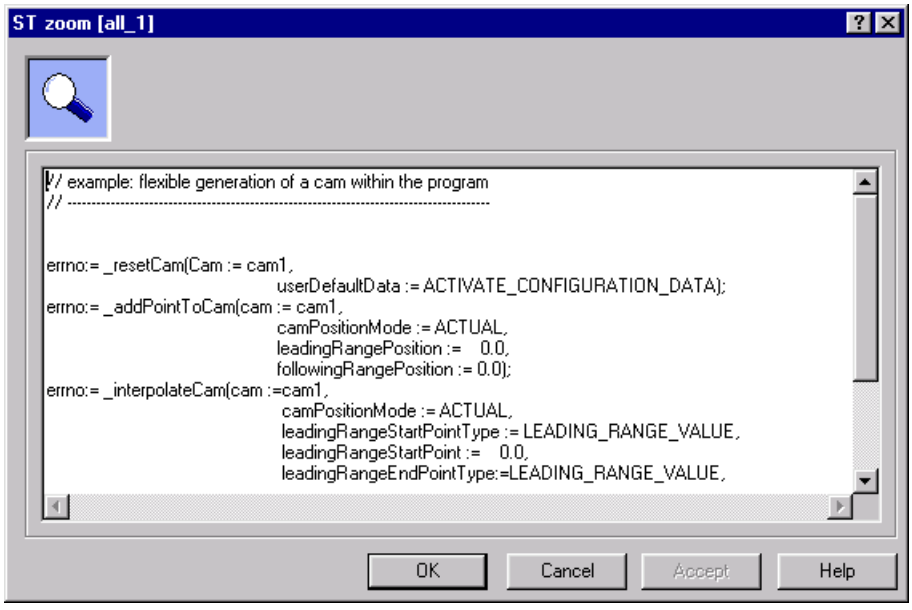


Figure 7-79 Parameter mask: ST zoom

Activate simulation for object

With this command, the selected technology objects (axes, output cams, measuring inputs, synchronous objects) are switched to simulation mode. The setpoint output for axes is suppressed. The output of output cams is not switched.

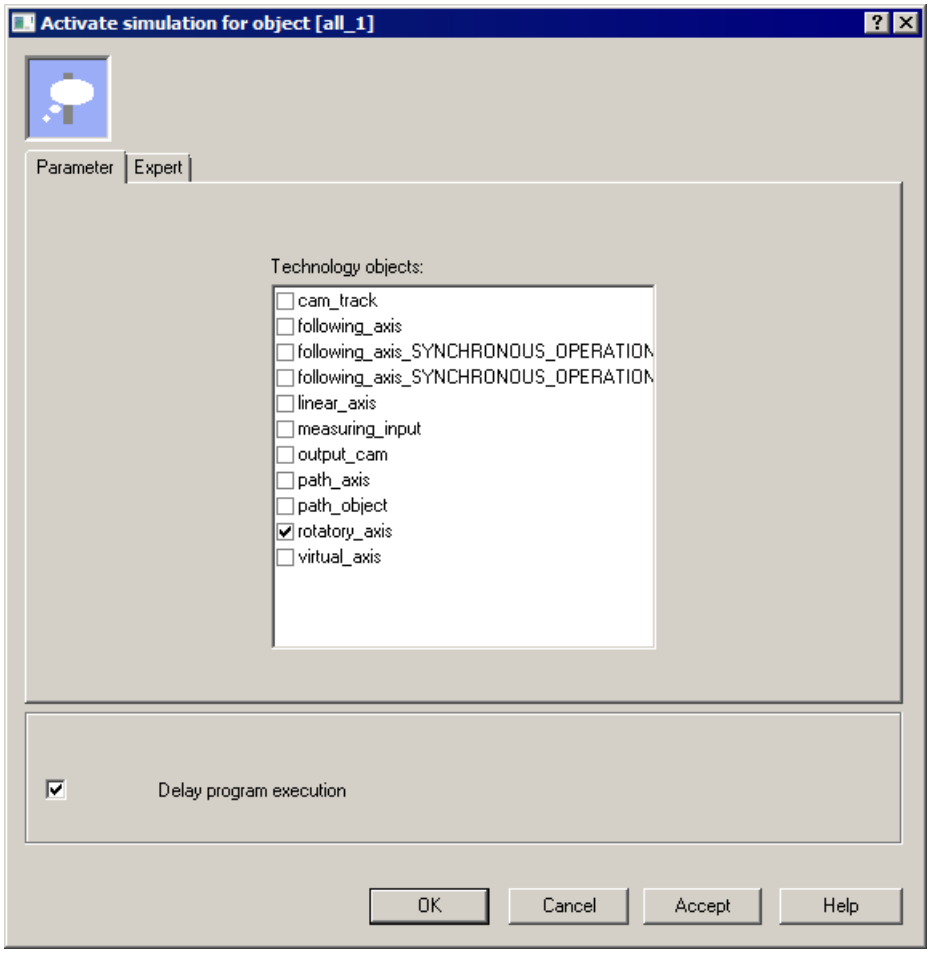


Figure 7-80 Parameter screen form: Activate simulation for object

Overview of parameters for Activate simulation for object

Table 7-51 Overview of parameters for Activate simulation for object

Field/button	Meaning/instruction
Parameters tab	See Overview of parameters for Activate simulation for object - Parameters tab (Page 4191)
Expert tab	See Overview of parameters for Activate simulation for object - Expert tab (Page 4191)
Delay program execution	Activate the checkbox if execution of the following command should be delayed until this command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Activate simulation for object - Parameters tab

Table 7-52 Overview of parameters for Activate simulation for object - Parameters tab

Field/button	Meaning/instruction
Technology objects	Here you select the technology objects that are to be switched to simulation mode. You can select several objects simultaneously. The following are available: • All objects that are defined on the device • All variables with data types of technology objects declared in the MCC unit or MCC chart (see Technology object data types (Page 4060))

Overview of parameters for Activate simulation for object - Expert tab

Table 7-53 Overview of parameters for Activate simulation for object - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Activate simulation for object

Cam technology package:

- _enableAxisSimulation for speed-controlled axes, positioning axes and synchronous axes
- _enableCamTrackSimulation for cam tracks
- _enableFollowingObjectSimulation for synchronous objects
- _enableMeasuringInputSimulation for measuring inputs
- _enableOutputCamSimulation for output cams

Path technology package:

- _enableAxisSimulation for path axes
- _enablePathObjectSimulation for path objects

Deactivate simulation for object

With this command, the selected technology objects (e.g. axes, output cams, measuring inputs, synchronous objects) are switched from simulation mode to normal mode. The setpoint output for axes is activated. The cam output of the output cams is switched switched.

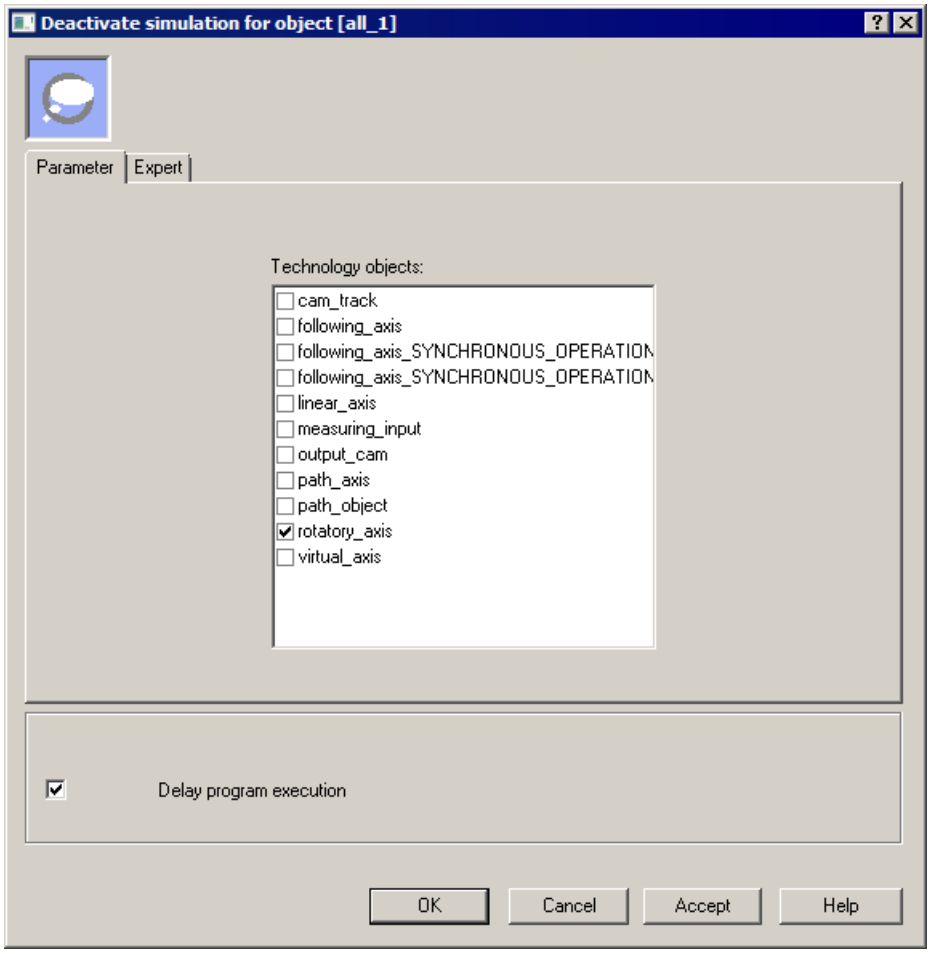


Figure 7-81 Parameter screen form: Deactivate simulation for object

Overview of parameters for Deactivate simulation for object

Table 7-54 Overview of parameters for Deactivate simulation for object

Field/button	Meaning/instruction
Parameters tab	See Overview of parameters for Deactivate simulation for object - Parameters tab (Page 4193)
Expert tab	See Overview of parameters for Deactivate simulation for object - Expert tab (Page 4193)
Delay program execution	Activate the checkbox if execution of the following command should be delayed until this command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Deactivate simulation for object - Parameters tab

Table 7-55 Overview of parameters for Deactivate simulation for object - Parameters tab

Field/button	Meaning/instruction
Technology objects	Here, you select the technology objects that are to be switched from simulation mode back to normal operation. You can select several objects simultaneously. The following are available: • All objects that are defined on the device • All variables with data types of technology objects declared in the MCC unit or MCC chart (see Technology object data types (Page 4060))

Overview of parameters for Deactivate simulation for object - Expert tab

Table 7-56 Overview of parameters for Deactivate simulation for object - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Deactivate simulation for object

Cam technology package:

- _disableAxisSimulation for speed-controlled axes, positioning axes and synchronous axes
- _disableCamTrackSimulation for cam tracks
- _disableFollowingObjectSimulation for synchronous objects
- _disableMeasuringInputSimulation for measuring inputs
- _disableOutputCamSimulation for output cams

Path technology package:

- _disableAxisSimulation for path axes
- _disablePathObjectSimulation for path objects

Reset object

The selected technology objects is reset to the initial state.

Optionally, the system variables of the selected technology objects can be reset to the configured default values.

You should use this command, for example, if an axis or a synchronous group is in an imprecisely defined state as the result of a programming or parameter assignment error.

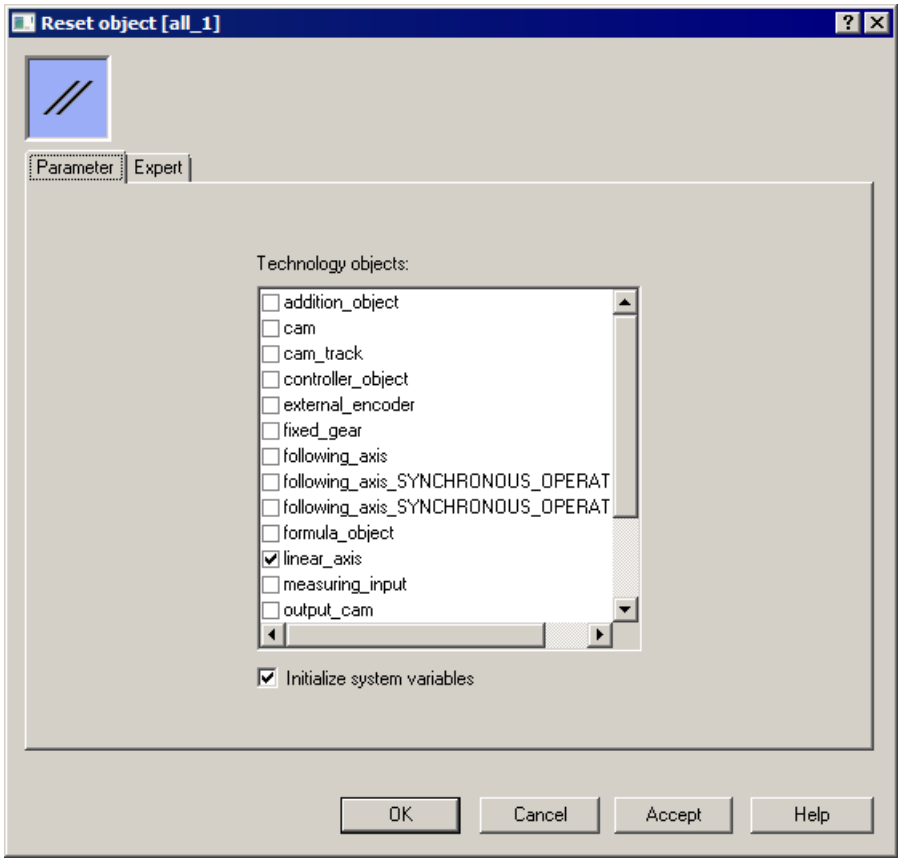


Figure 7-82 Parameter screen form: Reset object

Overview of parameters for Reset object

Table 7-57 Overview of parameters for Reset object

Field/button	Meaning/instruction
Parameters tab	See Overview of parameters for Reset object - Parameters tab (Page 4195)
Expert tab	See Overview of parameters for Reset object - Expert tab (Page 4195)

Overview of parameters for Reset object - Parameters tab

Table 7-58 Overview of parameters for Reset object - Parameters tab

Field/button	Meaning/instruction
Technology objects	Here, you select the axes, cams, etc., to be reset. You can select several objects simultaneously. The following are available: • All objects that are defined on the device • All variables with data types of technology objects declared in the MCC unit or MCC chart (see Technology object data types (Page 4060))
Initialize system variables	Activate the checkbox if the system variables of the selected technology objects are to be reset to the configured default values. If the checkbox is not activated, the system variables remain unchanged.

Overview of parameters for Reset object - Expert tab

Table 7-59 Overview of parameters for Reset object - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Reset object

Cam technology package:

- _resetAxis for speed-controlled axes, positioning axes and synchronous axes
- _resetCam for cams
- _resetCamTrack for cam tracks
- _resetExternalEncoder for external encoders
- _resetFollowingObject for synchronous objects
- _resetMeasuringInput for measuring inputs
- _resetOutputCam for output cams

Path technology package:

- _resetAxis for path axes
- _resetPathObject for path objects

Cam_EXT technology package:

- _resetAdditionObject for addition objects
- _resetControllerObject for controller objects
- _resetFixedGear for fixed gears

- _resetFormulaObject for formula objects
- _resetSensor for sensor objects

TControl technology package:

- _resetTController for temperature channels

Overview of parameters for Reset object / _reset*TO*

Table 7-60 Parameters (MCC command Reset object compared to system functions _reset*TO*)

Parameters of the MCC command Reset object	**Parameter of the system functions _reset*TO***
Initialize system variables	userDefaultData

Change operating mode

You can use this command to place the SIMOTION device in STOP or STOP U mode.

Note

To place the SIMOTION device in RUN mode again, activate the software switch in SIMOTION SCOUT or the mode switch on the SIMOTION device.

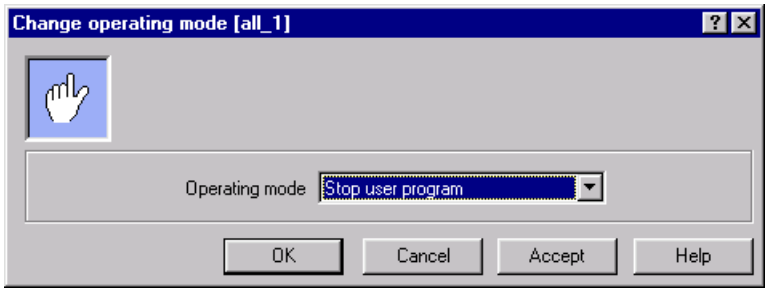


Figure 7-83 Parameter screen form: Change operating mode

Overview of parameters for Change operating mode

You can set the following parameters:

Table 7-61 Overview of parameters for Change operating mode

Field/Button	Explanation/instructions
Operating state	**STOP** STOP mode is selected. • Technology objects inactive (enables deleted, no axis motion) • User program is not executed • Loading a user program is possible • All system services are active (communication, etc.) • All analog and digital outputs set to 0 **STOP user program** STOPU mode is selected. • Technology objects active • Technology objects can execute jobs for testing and commissioning functions. • Otherwise identical to STOP mode

Relevant system functions for Change operating mode

System function of SIMOTION device: _changeOperationMode

Return value for Change operating mode

None.

Activate trace

This command enables you to log signal characteristics and axis status characteristics (see Activate trace (Page 4547)).

Comment block

You can use the comment block to structure the MCC chart and insert a comment anywhere within the program sequence. The size of the comment block is automatically adjusted to the size of the comment text. The block is displayed larger than normal command blocks in the chart.

If you place the cursor above the command in the MCC chart, the comment is displayed via a tool tip.

Example with Comment block

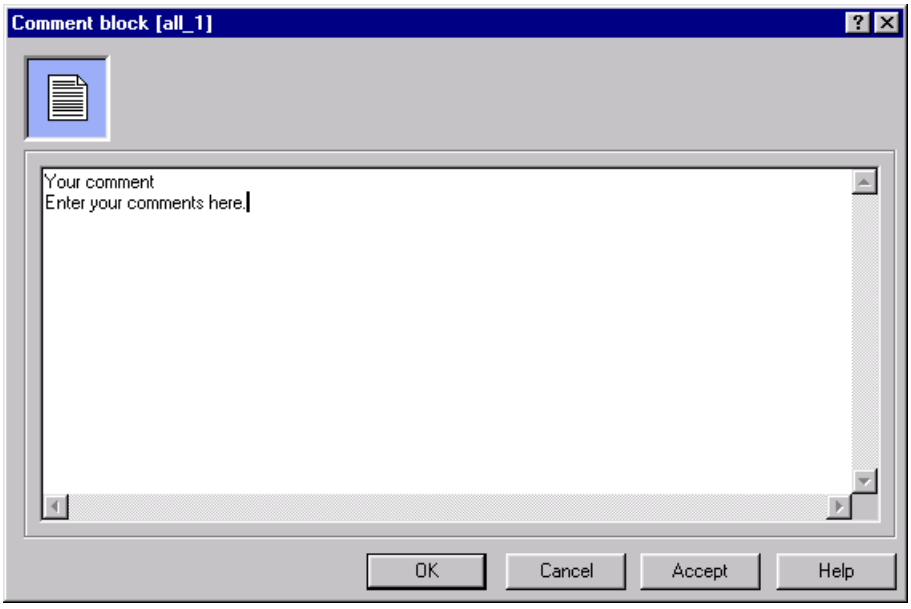
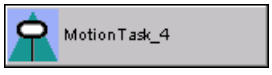


Figure 7-84 Parameter screen form: Comment block

7.1.6.2 Task commands

Start task



This command starts a MotionTask with initialization of data.

Note

If the task is already active, it is stopped and restarted with data initialization.

The task to be started can be selected as follows:

- Using its name (as specified in the execution system)
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructTaskId
  This variable contains the TaskId that you obtained using the "Determine taskID" function.
  In this form, the command can be used in libraries.

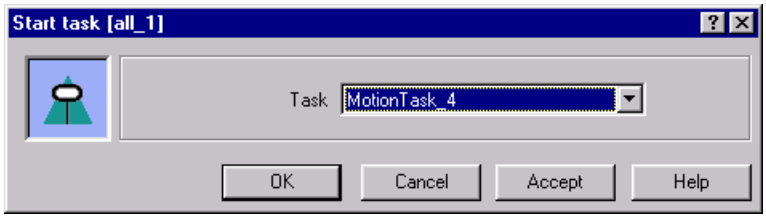


Figure 7-85 Parameter screen form: Start task

Note

This command may only be programmed for MotionTasks.

Call from the BackgroundTask

This command may be issued only once in the BackgroundTask. Otherwise, the selected MotionTask would be started at the beginning every time the BackgroundTask was executed.

To prevent this, you can query and evaluate the status of the MotionTask with the Task status command (see Example of use of Task status command (Page 4204)).

Overview of parameters for Start task

You can set the following parameters:

Table 7-62 Overview of parameters for Start task

Field/Button	Explanation/Instructions
Task	Select the task to be started. You can select from: • All MotionTasks (not in libraries) • All variables of data type StructTaskID declared in the MCC source file or MCC chart

Relevant system functions for Start task

ST system function: _restartTaskId

The _getTaskId system function is also called if the task was selected by name.

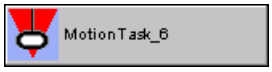
Return value for Start task

For ST system function _restartTaskId only:

The return value informs the user about the result of the command call (see description of the _restartTaskId function in the SIMOTION Basic Functions Function Manual).

Variable _MccRetDWORD with data type DWORD.

Interrupt task



The task is interrupted at its present position.

This command does not stop any axis motions that have already been started from this task.

Any active UserInterruptTask is stopped. If the condition for restarting this UserInterruptTask is true, it is not started nor is the fulfillment of the condition stored.

The task can be resumed with the "Continue task (Page 4201)" command.

The task to be interrupted can be selected as follows:

- Using its name (as specified in the execution system)
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructTaskId
  This variable contains the TaskId that you obtained using the "Determine taskID" function.
  In this form, the command can be used in libraries.

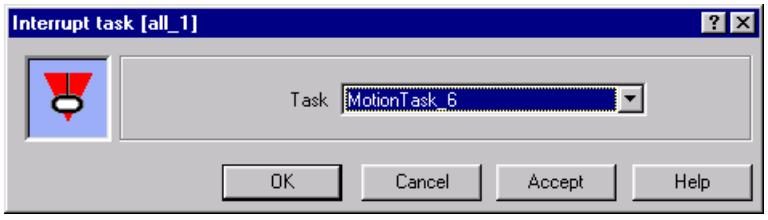


Figure 7-86 Parameter screen form: Interrupt task

Note

The command can be programmed for MotionTasks, UserInterruptTasks, and TimerInterruptTasks.

Overview of parameters for Interrupt task

You can set the following parameters:

Table 7-63 Overview of parameters for Interrupt task

Field/Button	Explanation/instructions
Task	Select the task to be interrupted. You can select from: • All MotionTasks (not in libraries) • UserInterruptTasks and TimerInterruptTasks configured on the device (not in libraries) • All variables of data type StructTaskId declared in the MCC source file or MCC chart

Relevant system functions for Interrupt task

ST system function: _suspendTaskId

The _getTaskId system function is also called if the task was selected by name.

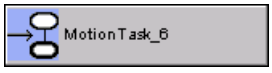
Return value for Interrupt task

For ST system function _suspendTaskId only:

The return value informs the user about the result of the command call (see description of the _suspendTaskId function in the SIMOTION Basic Functions Function Manual).

Variable _MccRetDWORD with data type DWORD.

Continue task



This command continues a task that was interrupted with the "Interrupt task (Page 4200)" command at the point it was interrupted.

Interrupted motions are not automatically continued. Resumption of interrupted motions must be programmed explicitly.

The task to be continued can be selected as follows:

- Using its name (as specified in the execution system)
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructTaskId
  This variable contains the TaskId that you obtained using the "Determine taskID" function.
  In this form, the command can be used in libraries.

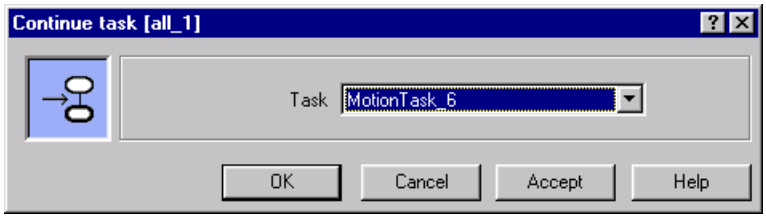


Figure 7-87 Parameter screen form: Continue task

Note

The command can be programmed for MotionTasks, UserInterruptTasks, and TimerInterruptTasks.

Overview of parameters for Continue task

You can set the following parameters:

Table 7-64 Parameter screen form for Continue task

Field/Button	Explanation/instructions
Task	Select the task to be continued. You can select from: • All MotionTasks (not in libraries) • UserInterruptTasks and TimerInterruptTasks configured on the device (not in libraries) • All variables of data type StructTaskID declared in the MCC source file or MCC chart

Relevant system functions for Continue task

ST system function: _resumeTaskId

The _getTaskId system function is also called if the task was selected by name.

Return value for Continue task

For ST system function _resumeTaskId only:

The return value informs the user about the result of the command call (see description of the _resumeTaskId function in the SIMOTION Basic Functions Function Manual).

Variable _MccRetDWORD with data type DWORD.

Reset task

This command stops a MotionTask. The initialization of the data can be restarted with the "Start task" command.

The task to be reset can be selected as follows:

- Using its name (as specified in the execution system)
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructTaskId
  This variable contains the TaskId that you obtained using the "Determine taskID" function.
  In this form, the command can be used in libraries.

Note

The "Reset task" and "Start task" commands cannot follow each other in direct succession. Instead, use only the "Start task" command.

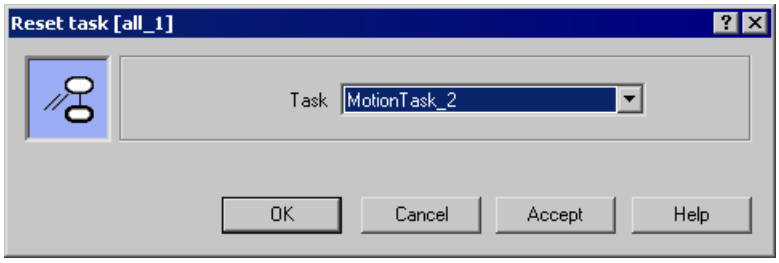


Figure 7-88 Parameter screen form: Reset task

Note

This command may only be programmed for MotionTasks.

Overview of parameters for Reset task

Table 7-65 Overview of parameters for Reset task

Field/Button	Explanation/instructions
Task	Select the task to be reset. You can select from: • All MotionTasks (not in libraries) • All variables of data type StructTaskID declared in the MCC source file or MCC chart

Relevant system functions for Reset task

ST system function: _resetTaskId

The _getTaskId system function is also called if the task was selected by name.

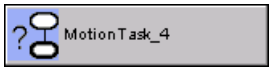
Return value for Reset task

For ST system function _resetTaskId only:

The return value informs the user about the result of the command call (see description of the _resetTaskId function in the SIMOTION Basic Functions Function Manual).

Variable _MccRetDWORD with data type DWORD.

Task status



This command returns the status of a task.

The task can be selected as follows:

- Using its name (as specified in the execution system)
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructTaskId
  This variable contains the TaskId that you obtained with the "Determine TaskId" function.
  In this form, the command can be used in libraries.

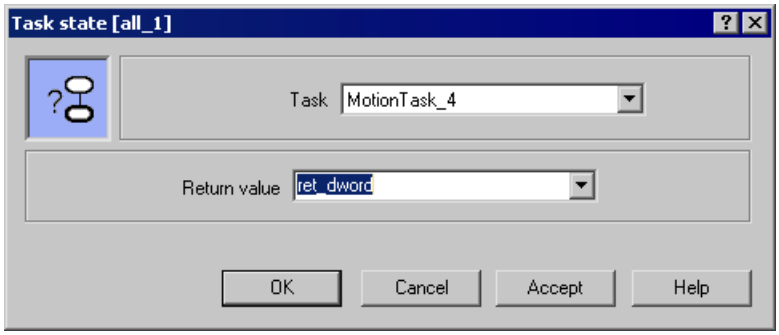


Figure 7-89 Parameter screen form: Task status

Note

The command can be programmed for MotionTasks, UserInterruptTasks, and TimerInterruptTasks.

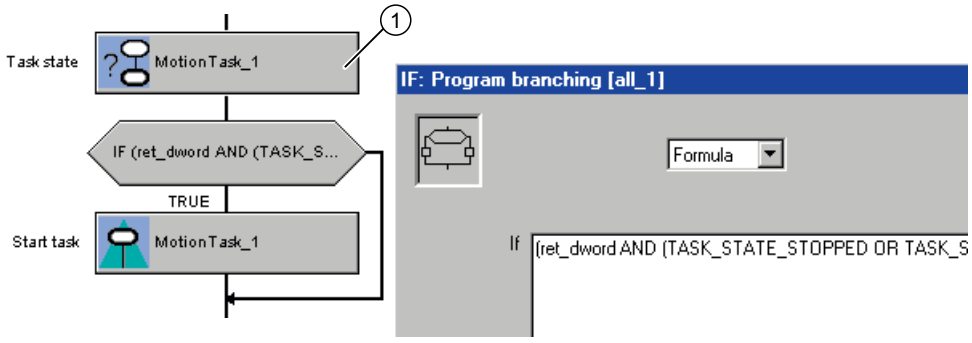
Overview of parameters for Task status

Table 7-66 Parameter screen form for Task status

Field/button	Meaning/instruction
Task	Select the task whose current state is to be determined. You can select from: • All MotionTasks (not in libraries) • UserInterruptTasks and TimerInterruptTasks configured on the device (not in libraries) • All variables of data type StructTaskID declared in the MCC unit or MCC chart
Return value	Here, you enter the return variable of type DWORD in which the task status will be stored as the command result (see description of the _getStateOfTaskId function in the SIMOTION Basic Functions Function Manual). This variable must be defined in a declaration table or the symbol browser.

Example of use of Task status command

In the following example, the task status of a MotionTask is queried to decide whether it can be started. The relevant bits of the return value are evaluated for this. If the result is positive, the MotionTask is started.



① The return value of the Task status command is stored in local variable ret_dword with data type DWORD. It indicates the status of MotionTask_1, which is then evaluated.

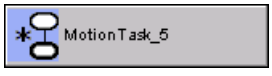
Figure 7-90 Example of a query whether a MotionTask can be started

Relevant system functions for Task status

ST system function: _getStateOfTaskId

The _getTaskId system function is also called if the task was selected by name.

Determine TaskId



This command generates a project-wide unique TaskId from the name of a task. This TaskId is assigned to a variable of data type StructTaskId.

This function must **not** be used in libraries.

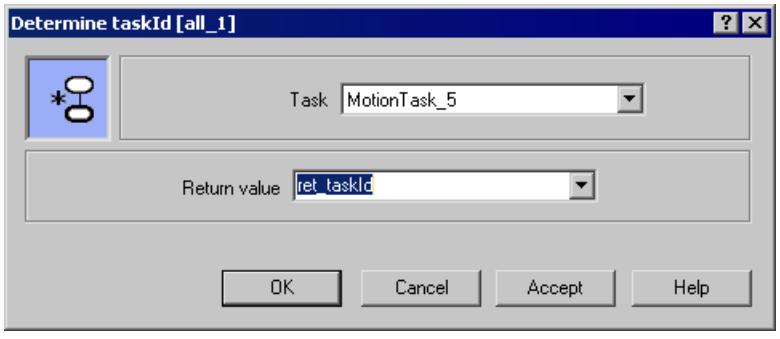


Figure 7-91 Parameter screen form: Determine TaskId

Note

The command can be programmed for MotionTasks, UserInterruptTasks, and TimerInterruptTasks.

Overview of parameters for Determine TaskId

You can set the following parameters:

Table 7-67 Overview of parameters for Determine TaskId

Field/Button	Explanation/instructions
Task	Select the name of the task whose project-wide unique TaskId is to be determined. You can select from: • All MotionTasks • UserInterruptTasks and TimerInterruptTasks configured on the device
Return value	Here, you enter the return variable of type StructTaskId in which the TaskId will be stored. This variable must be defined in a declaration table or the symbol browser.

Relevant system functions for Determine TaskId

ST system functions: _getTaskId.

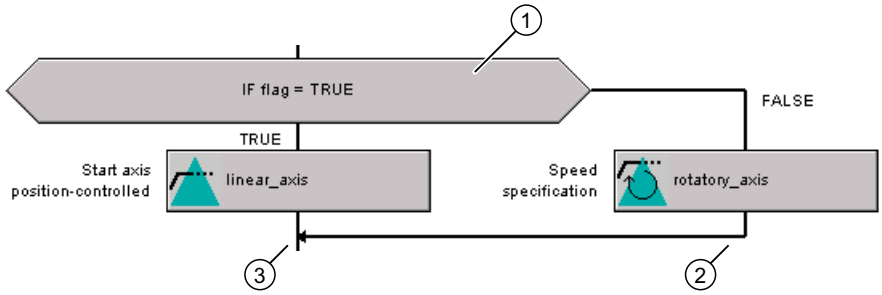
7.1.6.3 Program structures

IF: Program branching

A specified condition causes the program flow to branch to the TRUE branch or FALSE branch.

The program flow branches as follows:

- Condition is fulfilled: Execution continues down the TRUE path.
- Condition is not fulfilled: Execution continues down the FALSE path.



① Depending on the "flag" variables, either the linear axis or the rotary axis are started.
② Condition is not fulfilled.
③ Condition is fulfilled.

Figure 7-92 Example of a branch (IF) in a flowchart

Note

The condition is programmed in the ladder diagram (LAD) (Page 4153), function block diagram (FBD) (Page 4156) or Formula (Page 4158) languages.

Note on inserting the IF statement

The IF statement is normally inserted after the selected command. The two branches are empty.

If two or more commands are selected, the IF statement is inserted **before** the selected commands. The selected commands are automatically shifted to the TRUE branch; see figure below.

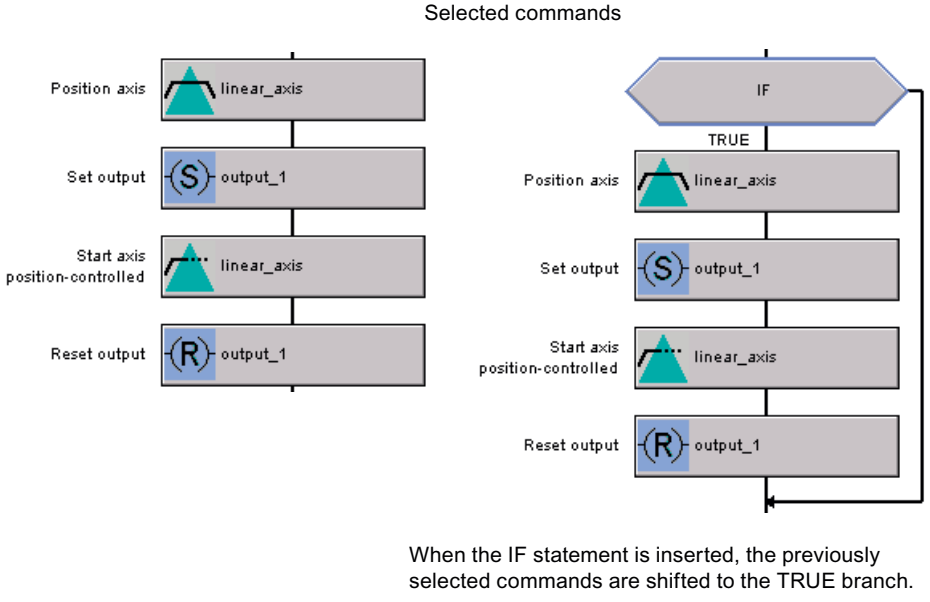


Figure 7-93 Inserting the IF statement when several commands are selected

Note on deleting/cutting or copying an IF statement

When you delete, cut, or copy an IF statement, the commands assigned in the branches are also deleted or copied.

If the commands within the control structure are not to be deleted or cut when the structure is deleted or cut, you must copy them first and repaste them outside the control structure.

Relevant commands of the ST (Structured Text) programming language for IF: Program branching

IF / ELSE / END_IF

WHILE: Loop with condition at the start

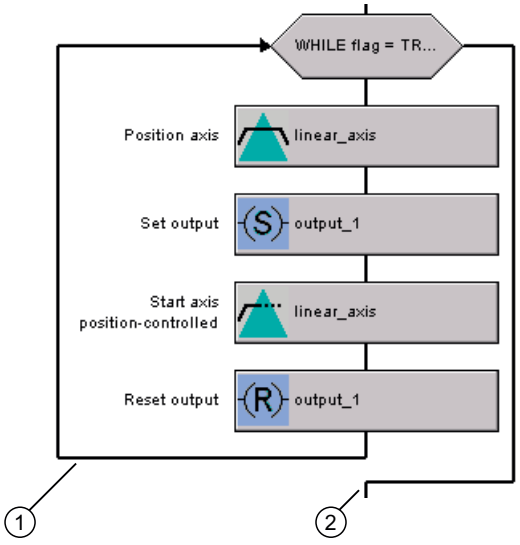
Commands programmed within the loop are executed as long as the loop condition is fulfilled.

The condition is programmed at the start of the loop.

When the WHILE command is reached, the program flow branches as follows:

- Condition is fulfilled: Program runs through loop.
- Condition is not fulfilled: Program continues.

Example of WHILE loop with condition at start



① Condition is fulfilled

② Condition is not fulfilled

Figure 7-94 Example of repetition with query at beginning (WHILE) in the flowchart

Note

The condition is programmed in the ladder diagram (LAD) (Page 4153), function block diagram (FBD) (Page 4156) or Formula (Page 4158) languages.

Refer to the notes regarding inserting (Page 4207), deleting, cutting, or copying of the IF statement (Page 4208); they apply analogously.

Relevant commands of the ST (Structured Text) programming language for WHILE: Loop with condition at the start

WHILE /END_WHILE

FOR: Loop with fixed number of runs

The FOR loop repeats a program section for a fixed number of runs. Once the loop has been run the specified number of times, the program continues.

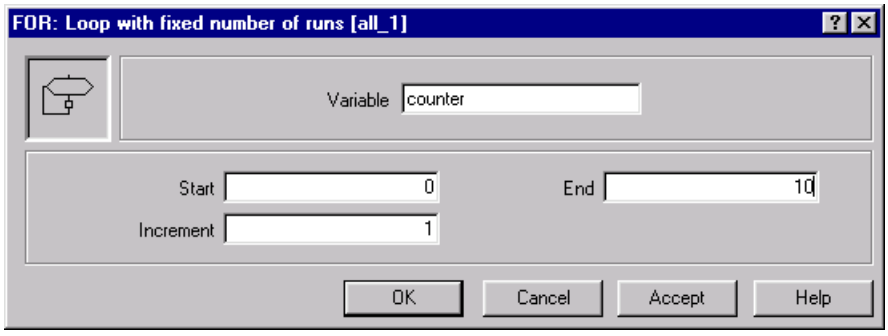


Figure 7-95 Parameter screen form: FOR statement

You use a FOR loop when the number of cycles is known at the programming stage.

If the number of cycles is not known, a loop with the WHILE (Page 4208) or UNTIL (Page 4211) command is more suitable.

Overview of parameters for FOR: Loop with fixed number of runs

You can set the following parameters:

Table 7-68 Overview of parameters for FOR loop with fixed number of runs

Field/button	Meaning/instruction
Variable	Enter the count variable for the FOR loop here. The variable must be of the SINT, USINT, INT, UINT or DINT data type. The variable name can be entered or moved from the declaration table using drag and drop.
Start	Enter the initial value of the count variable here. The count variable is set to the specified value at the first pass of the loop start. The initial value can be directly entered as a value or, alternatively, as a formula. Variables can be used. The expression must be of the count variables data type.
End	Enter the final value of the count variable here. The loop is executed repeatedly until the count variable exceeds the specified final value. The value can be directly entered as a value or, alternatively, as a formula. Variables can be used. The expression must be of the count variables data type.
Increment	Enter the increment here. The count variable is increased by the entered increment each time the loop is run. The value can be directly entered as a value or, alternatively, as a formula. Variables can be used. The expression must be of the count variables data type.

The values of these parameters must not be changed during the execution of the FOR loop.

Example of a FOR loop in a program

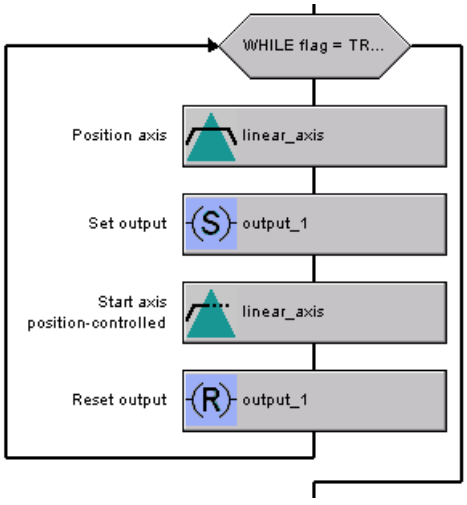


Figure 7-96 Example of a FOR loop in a program

Note

Refer to the notes regarding inserting (Page 4207), deleting, cutting, or copying of the IF statement (Page 4208); they apply analogously.

Example of a FOR statement

The program loop is to be executed 11 times. The variable must be defined and of the SINT, USINT, INT, UINT or DINT data type.

The parameter screen form is configured as shown in the figure below.

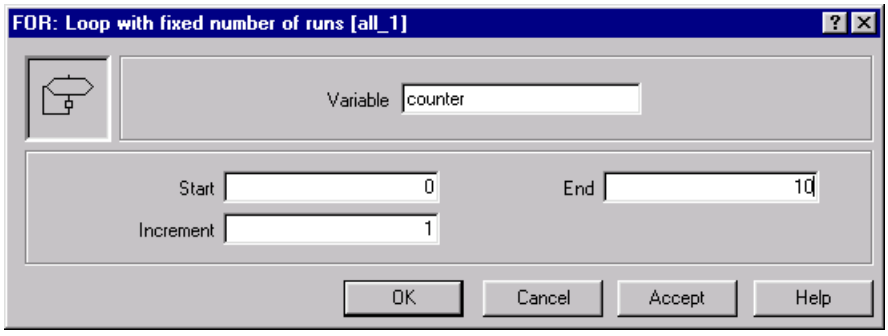


Figure 7-97 Example of a FOR statement

Relevant commands of the ST (Structured Text) programming language for FOR: Loop with fixed number of runs

FOR / END_FOR

UNTIL: Loop with condition at the end

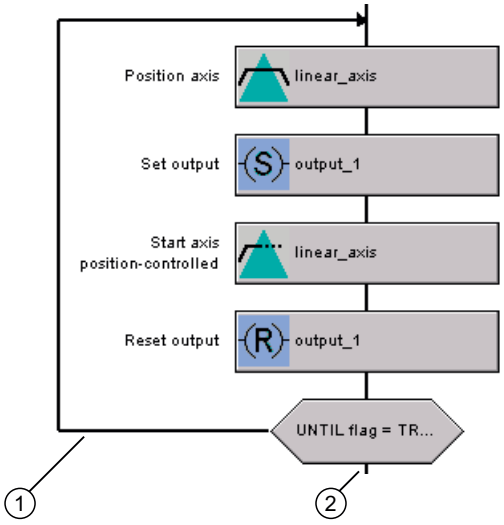
The commands programmed within the loop are executed **until** the loop condition is fulfilled.

The condition is programmed at the end of the loop. For this reason, the commands in the loop are executed at least once.

When the UNTIL command is reached, program flow branches as follows:

- Condition is not fulfilled: Program runs through loop.
- Condition is fulfilled: Program continues.

Example of UNTIL loop with condition at end



① Condition is not fulfilled

② Condition is fulfilled

Figure 7-98 Example of repetition with query at the end (UNTIL) in the flowchart

Note

The condition is programmed in the ladder diagram (LAD) (Page 4153), function block diagram (FBD) (Page 4156) or Formula (Page 4158) languages.

Refer to the notes regarding inserting (Page 4207), deleting, cutting, or copying of the IF statement (Page 4208); they apply analogously.

Relevant commands of the ST (Structured Text) programming language for UNTIL: Loop with condition at the end

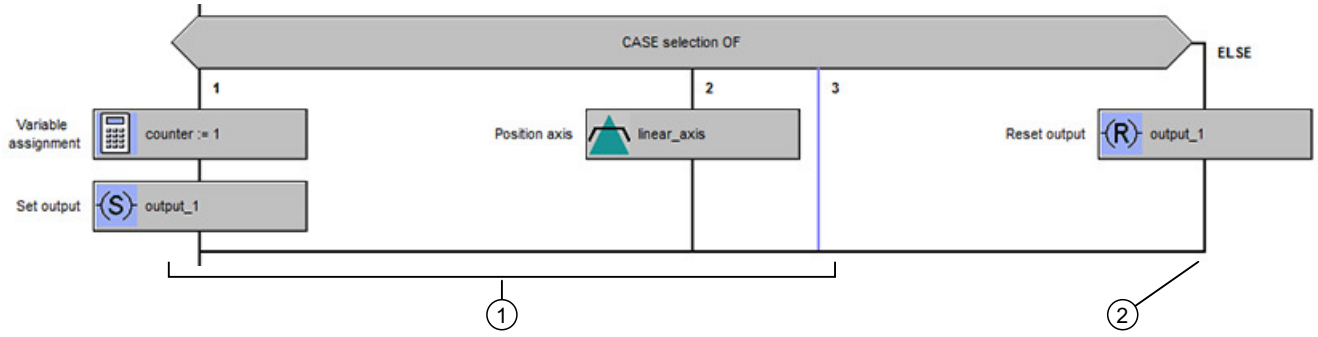
REPEAT / UNTIL / END_REPEAT

CASE: Multiple branching

Depending on the value of a variable, several alternative program branches may be executed. The following variable data types are permitted:

- General data type ANY_INT
- Any enumeration data type

Example of CASE Multiple branching



① Comparison value corresponds to one of the programmed constants

② Comparison value does not correspond to any of the programmed constants

Figure 7-99 Description of the variable and ELSE branch

Inserting a branch in the CASE statement

After insertion of this command, there are two program branches. To insert additional branches, proceed as follows:

- To insert a branch in the first position (left):
  Select the command, and select **Insert branch** from the context menu.
- To insert additional branches:
  Select the branch, and select **Insert branch** from the context menu.
  The branch is inserted on the right next to the selected branch.

Deleting a branch from the CASE statement

To delete a branch, proceed as follows:

1. Delete all commands from the branch.
2. Select the branch.
3. Select **Delete empty branch** from the context menu.

Comparison constant in the CASE statement

Overwrite the default ??? in every branch with a constant of the same data type as the variable. A value range can also be specified, for example, 1,2,3 or 4 to 10.

It is not possible to overwrite the default value in an ELSE branch.

The number of displayable characters for the comparison constant on a branch is limited. Long comparison constants are therefore shown in a shortened form with omission dots in the middle of the character string. However, the entire comparison constant is shown in the tooltip.

Overview of parameters for CASE: Multiple branching

You can set the following parameters:

Table 7-69 Overview of parameters for CASE multiple branching

Field/button	Meaning/instruction
Variable	Enter a variable of the following data types: • General data type ANY_INT • Any enumeration data type The variable name can be entered or moved from the symbol browser using a drag and drop operation. The CASE branch whose label is consistent with the variable content is executed. If the content of the variable does not match any of the constants or constant ranges, then the ELSE branch is executed.

Note

Refer to the notes regarding inserting (Page 4207), deleting, cutting, or copying of the IF statement (Page 4208); they apply analogously.

Relevant commands of the ST (Structured Text) programming language for CASE: Multiple branching

CASE / ELSE / END_CASE

Go to

You can program jumps within an MCC chart using the Go to command. A jump is defined as a jump label (Go to command) and a jump destination (Selection command), each with the same name. A circle symbol with a name marks the exit and entry point in each case. Several exit points with the same name can be programmed, but only one entry point.

The following jumps are illegal:

- Jumps to control structures
- Jumps out of the shaded area after a Wait command

After you have inserted a Go to command, you must enter the name of the jump destination. For this, double-click inside the circle.

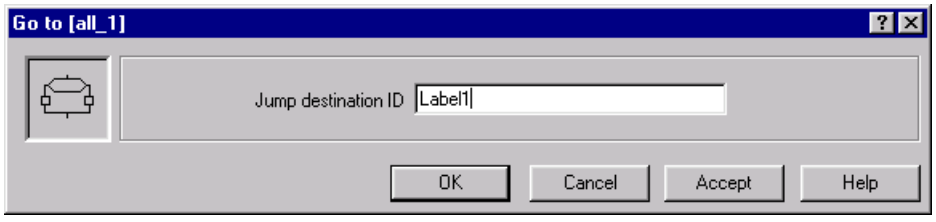


Figure 7-100 Parameter screen form: Go to

Overview of parameters for Go to

You can set the following parameters:

Table 7-70 Overview of parameters for Go to

Field/Button	Explanation/instructions
Jump destination ID	Enter the name of the jump destination. The name is made up of the following: • The first character is a letter or underscore symbol • The remaining characters are letters, underscore symbols, or digits It is irrelevant whether the letters are upper or lower case. The name must be identical to the newly assigned name for the jump label – see Selection (Page 4215) command.

Relevant commands of the ST (Structured Text) programming language for Go to:

GOTO

Selection

This command is used to identify the entry point for the Go To command. A jump is defined as a jump label (Go to command) and a jump destination (Selection command), each with the same name. A circle symbol with a name marks the exit and entry point in each case. Several exit points with the same name can be programmed, but only one entry point.

The following jumps are illegal:

- Jumps to control structures
- Jumps out of the shaded area after a Wait command

After you have inserted the Selection command, you must enter the name of the jump label. For this, double-click inside the circle.

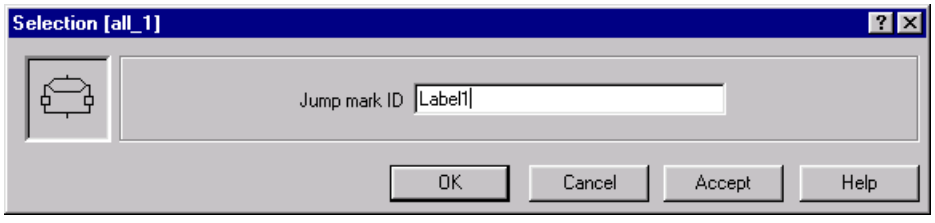


Figure 7-101 Parameter screen form: Selection (jump label)

Overview of parameters for Selection

You can set the following parameters:

Table 7-71 Overview of parameters for Selection

Field/Button	Explanation/instructions
Jump mark ID	Enter the name of the jump label. The name is made up of the following: • The first character is a letter or underscore symbol • The remaining characters are letters, underscore symbols, or digits It is irrelevant whether the letters are upper or lower case. The name must be identical to the newly assigned name for the jump destination – see Go to (Page 4214) command.

Relevant commands of the ST (Structured Text) programming language for Selection:

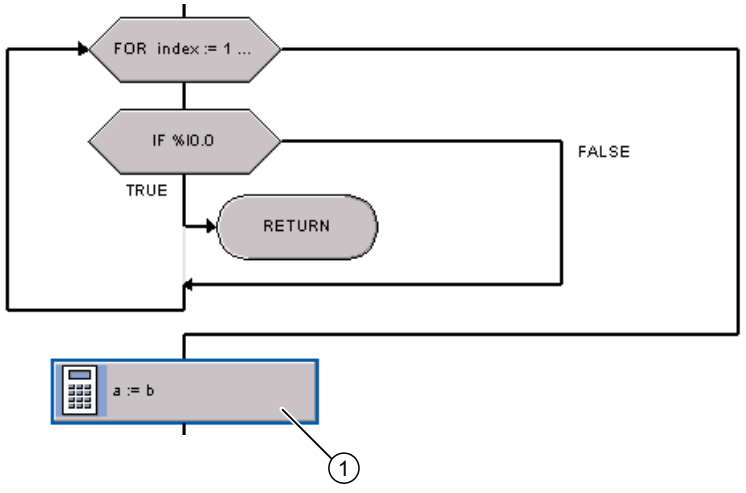
Definition of a jump label

Return

This command causes the MCC chart (program, function, function block) to end.

When a function or a function block is ended, program execution continues in the higher-level MCC chart (or program organization unit) after the position where the function or function block was called.

Example of Return



① The variable assignment is executed after the regular end of the FOR loop, but not after the execution of the Return command.

Figure 7-102 Example of Return

Relevant commands of the ST (Structured Text) programming language for Return:

RETURN

Exit

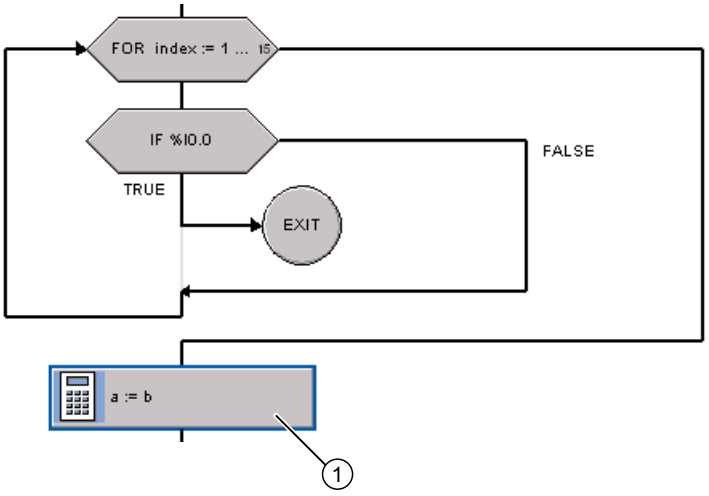
This command is permitted in the following program structures (loops):

- WHILE: Loop with condition at the start (Page 4208)
- FOR: Loop with fixed number of runs (Page 4209)
- UNTIL: Loop with condition at the end (Page 4211)

It enables the loop to be exited at any location without requiring the abort condition to be fulfilled.

The program is continued at the end of the loop.

Example of Exit



① The variable assignment is executed after the execution of the Exit command and the regular end of the FOR loop.

Figure 7-103 Example of Exit

Relevant commands of the ST (Structured Text) programming language for Exit:

EXIT

Continue

This command is only available if the compiler option **Language extensions IEC61131 3rd Edition** is enabled at the MCC source file, see Global settings of the compiler (Page 3996) and Local settings of the compiler (Page 3997).

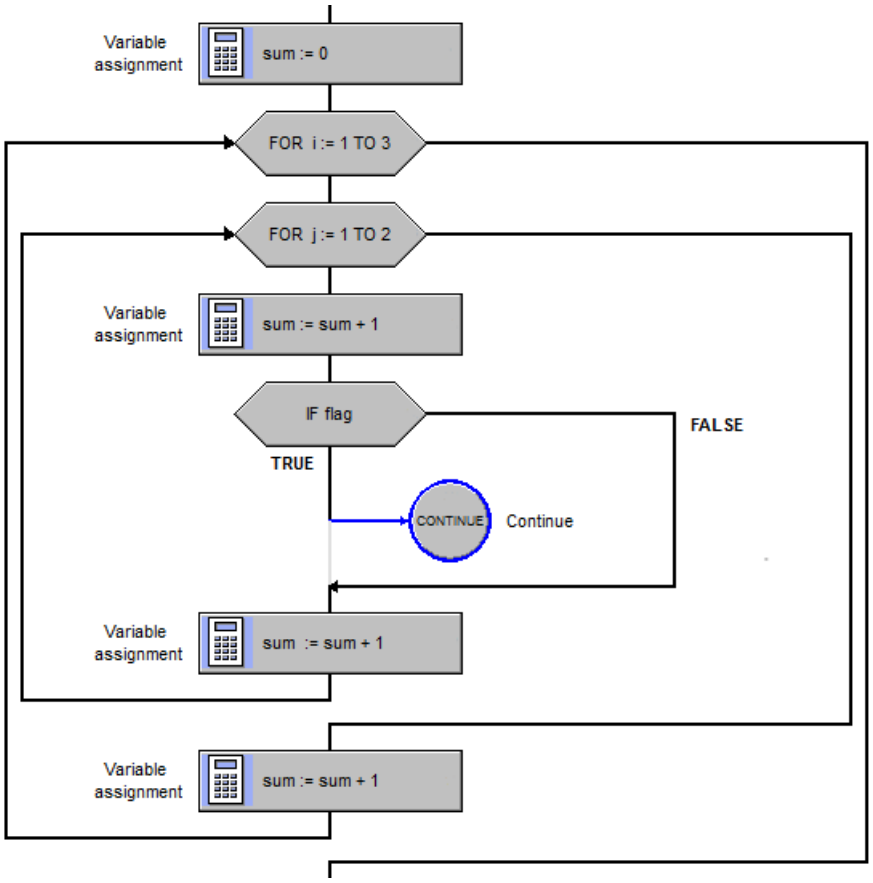
This command is permitted in the following program structures (loops):

- WHILE: Loop with condition at the start (Page 4208)
- FOR: Loop with fixed number of runs (Page 4209)
- UNTIL: Loop with condition at the end (Page 4211)

The command instigates a jump at the start of the loop.

- The condition is reviewed with the WHILE loop.
- The counter variable is incremented and compared with the full-scale value with the FOR loop.

Example for Continue



Value of the "sum" variable after the FOR loops have run

flag = FALSE: sum = 15

flag = TRUE: sum = 9

Figure 7-104 Example for Continue

Relevant commands of the ST (Structured Text) programming language for Continue

CONTINUE

Synchronous start

You can use this command to start several commands simultaneously. The permissible commands are listed in **Permissible commands**.

The next command is started once all the commands in the branches have been executed and ended. In addition, details of when a command is ended are entered in **Permissible commands**.

To check the correct completion of the synchronous start, you can query whether the commands, which were started synchronously, have been completed without error via the group variable _MccRetSyncStart. The group variable is set to not equal to 0 if a command within the synchronous start had a return value not equal to 0 (see SIMOTION Basic Functions Function Manual).

The following MCC sample program shows a synchronous start with two axes and subsequent query of the group variable _MccRetSyncStart. This is not equal to 0 when a command within the synchronous start had a return value not equal to 0.

There is no limit to the number of commands started simultaneously nor to the number of synchronous starts within an MCC chart.

This command should only be used in MotionTasks.

Note

When you use the Synchronous Start command, you must generate the UserInterruptTask_1 since this is called in the event of an error. You can program an error response in this task.

The task controller is temporarily shut down during the synchronous start. It is not restarted until the following conditions are met:

- All single-axis commands and synchronous operation and cam commands in the branches have been started
- All basic commands in the branches have been completed

Start interruptions by other tasks (except SynchronousTasks) are thus prevented. This can cause a timeout to occur in cyclic tasks (BackgroundTask, TimerInterruptTasks). This error can be detected and caught by programming the TimeFaultBackgroundTask or TimeFaultTask appropriately.

For information about execution levels and tasks, see Execution levels and tasks in SIMOTION in the SIMOTION ST Programming Manual.

Example: Synchronous start of three commands

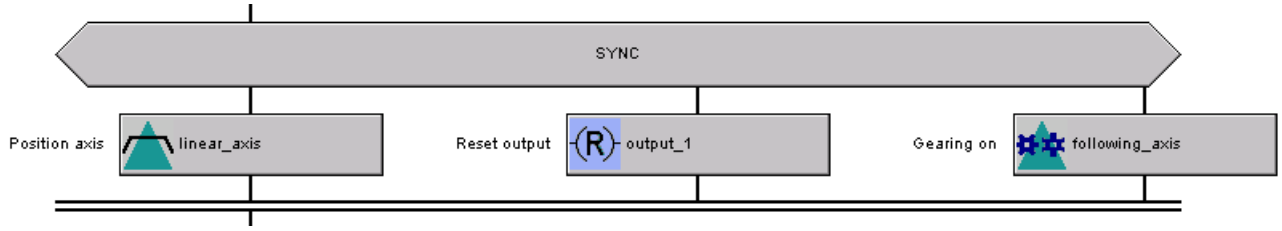


Figure 7-105 Synchronous start of three commands

Programming a condition

You can program a condition for the synchronous start. TRUE is the default setting when the command is called. This causes immediate synchronous start.

Programming a condition corresponds to the Wait for condition command. Consequently, you must note the following:

The condition can be composed of:

- Unit variables of the same MCC unit or imported program source files
- Global device variables
- Constants
- I/O variables and process image accesses (inputs)
- Operators

The expression may not contain:

- Function calls
- Local variables
- Loops

Note

The condition is programmed in the ladder diagram (LAD), function block diagram (FBD), or Formula languages.

Checkbox

If you activate the **Start command if condition is fulfilled** check box, the commands in the branches are not executed until the programmed condition is fulfilled.

Insert branch

After insertion of this command, there are two program branches. To insert additional branches, proceed as follows:

- To insert a branch in the first position (left):
  Select the command, and select **Insert branch** from the context menu.
- To insert additional branches:
  Select the branch, and select **Insert branch** from the context menu.
  The branch is inserted on the right next to the selected branch.

Deleting a branch

To delete a branch, proceed as follows:

1. Delete all commands from the branch.
2. Select the branch.
3. Select **Delete empty branch** from the context menu.

Permissible commands

The following table lists the commands permitted for the synchronous start.

Note

In the case of single axis commands and synchronous operation and cam commands, the synchronous start can only be guaranteed if the following two conditions are satisfied:

The commands embedded in the synchronous start must act on different axes.

The axes involved must be on the same ExecutionLevel (IPO or IPO_2).

The following table shows the permissible commands for a synchronous start with specification of timing for proper completion of commands; timing depends on whether an additional command is present on the relevant axis.

Table 7-72 Permissible commands for a synchronous start with specification of timing for proper completion of commands; timing depends on whether an additional command is present on the relevant axis

Permissible command	Timing for proper completion of command	
	Another command is present	No other command is present
Basic commands		
Subroutine call	-	-
Set output	-	-
Reset output	-	-
Variable assignment	-	-
Single axis commands		
Homed axis	Motion is finished	Motion is finished

Permissible command	Timing for proper completion of command	
	Another command is present	**No other command is present**
Start axis position-controlled	• With time specification: End of setpoint interpolation • Without time specification: Infinite	• With time specification: Motion is finished • Without time specification: Infinite
Speed preset	• With time specification: End of setpoint interpolation • Without time specification: Infinite	• With time specification: Standstill signal of axis • Without time specification: Infinite
Position axis	End of setpoint interpolation	Motion is finished
Time-dependent position profile	End of setpoint interpolation	Motion is finished
Time-dependent velocity profile	End of setpoint interpolation	• Speed-controlled (axis: Standstill signal of axis • Positioning axis or synchronous axis: Motion is finished
Stop axis	End of setpoint interpolation	• Speed-controlled (axis: Standstill signal of axis • Positioning axis or synchronous axis: Motion is finished
Commands for synchronous operation and camming		
Gearing on	Infinite	Infinite
Gearing off	End of setpoint interpolation	End of setpoint interpolation
Cam on	• Cyclic cam Infinite • Non-cyclic cam End of setpoint interpolation	• Cyclic cam Infinite • Non-cyclic cam End of setpoint interpolation
Cam off	End of setpoint interpolation	End of setpoint interpolation

Transition behavior

All single axis commands and commands for synchronous operation and camming embedded in the synchronous start are programmed with the **Substitute** transition behavior.

For the following commands, the **Superimpose** transition behavior can also be selected:

- Start axis position-controlled
- Speed preset
- Position axis
- Time-dependent position profile
- Time-dependent velocity profile

Step enabling condition

For each single axis command or command for synchronous operation and camming embedded in the synchronous start, the **Delay program execution** check box is activated. The user cannot change this setting. This indicates that once the synchronous start is executed, the MotionTask is placed in the waiting state until the last command enclosed within the synchronous start has been completed or aborted.

The timing for proper completion of a command depends on whether another axis command is pending, and is indicated in the table above in the Permissible commands section.

Deleting/cutting or copying

When you delete, cut, or copy the synchronous start command, any commands programmed in the branches are also deleted or copied.

Relevant system functions and commands of the ST (Structured Text) programming language for synchronous start

System functions of SIMOTION devices:

- BEGIN_SYNC / END_SYNC
- _disableScheduler / _enableScheduler
- _startSyncCommands

ST system function:

- _getSyncCommandId
- _waitTime

ST commands:

- EXPRESSION / END_EXPRESSION for formulation of the wait condition
- WAITFORCONDITION / END_WAITFORCONDITION

7.1.6.4 Communication

Acknowledge technology object alarms

This command acknowledges the pending technology alarms on one or more technology objects.

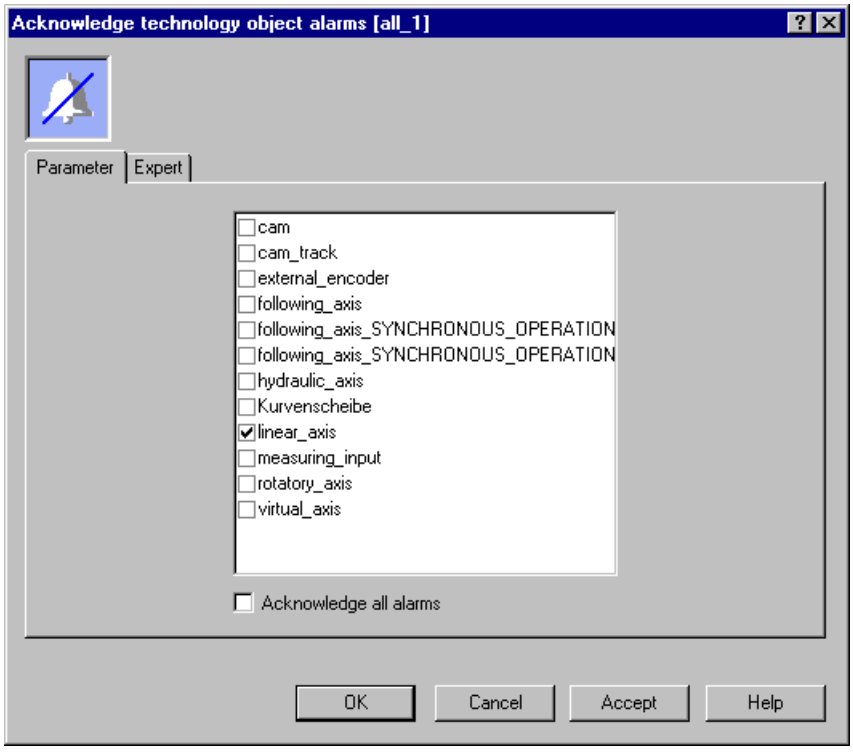


Figure 7-106 Parameter screen form: Acknowledge technology object alarms

Overview of parameters for Acknowledge technology object alarms

Table 7-73 Overview of parameters for Acknowledge technology object alarms

Field/Button	Explanation/instructions
Parameters tab	See Overview of parameters for Acknowledge TO alarms - Parameter tab (Page 4224)
Expert **tab**	See Overview of parameters for Acknowledge TO alarms - Expert tab (Page 4225)

Overview of parameters for Acknowledge technology object alarms - Parameters tab

Table 7-74 Overview of parameters for Acknowledge technology object alarms - Parameters tab

Field/Button	Explanation/Instructions
Technology object	Here, you select the technology objects (axes, cams, etc.) whose alarms are to be acknowledged. You can select several objects simultaneously. The following are available: • All objects that are defined on the device • All variables with data types of technology objects declared in the MCC source file or MCC chart (see Technology object data types (Page 4060) table).
Acknowledge all alarms	Activate this check box if all alarms queued on the device are to be acknowledged.

Overview of parameters for Acknowledge technology object alarms - Expert tab

Table 7-75 Overview of parameters for Acknowledge technology object alarms - Expert tab

Field/Button	Explanation/Instructions
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. DINT data type. • When the **Acknowledge all alarms** check box is activated: The return value is always 0. • When the **Acknowledge all alarms** check box is cleared: For a description, see Return value for the system functions of the technology packages (Page 4040).

Relevant system functions for Acknowledge technology object alarms

When the **Acknowledge all alarms** check box is active:

- System function of SIMOTION devices
  - _resetTechnologicalErrors for resetting all objects

When the **Acknowledge all alarms** check box is cleared:

- Cam technology package:
  - _resetAxisError for drive axes, position axes, synchronized axes, and path axes
  - _resetCamError for cams
  - _resetCamTrackError for cam tracks
  - _resetExternalEncoderError for external encoders
  - _resetFollowingObjectError for synchronous objects
  - _resetMeasuringInputError for measuring inputs
  - _resetOutputCamError for output cams
- Path technology package:
  - _resetPathObjectError for path objects
- Cam_EXT technology package:
  - _resetAdditionObjectError for addition objects
  - _resetControllerObjectError for controller objects
  - _resetFixedGearError for fixed gear
  - _resetFormulaObjectError for formula objects
  - _resetSensorError for sensor objects
- TControl technology package:
  - _resetTControllerError for temperature channels

Each with parameter errorResetMode = ALL_ERRORS (default)

See also the **Acknowledge all alarms** field/button in the Overview of parameters for Acknowledge technology object alarms - Parameters tab (Page 4224) section.

Acknowledge specific technology object alarm

The command acknowledges all alarms or a specific alarm at a technology object.

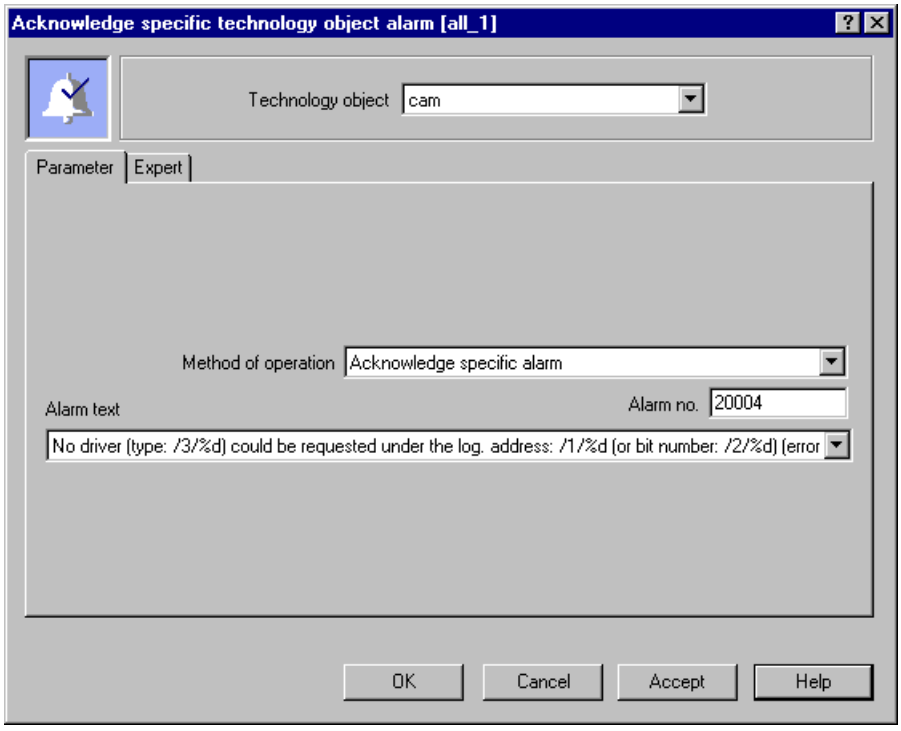


Figure 7-107 Parameter screen form: Acknowledge specific technology object alarm

Overview of parameters for Acknowledge specific technology object alarm

You can set the following parameters:

Table 7-76 Overview of parameters for Acknowledge specific technology object alarm

Field/Button	Explanation/instructions
Technology object	Here, you select the technology objects (axes, cams, etc.) whose alarms are to be acknowledged. You can select several objects simultaneously. The following are available: • All objects that are defined on the device • All variables with data types of technology objects declared in the MCC source file or MCC chart (see Technology object data types table).
Parameters tab	See Overview of parameters for Acknowledge specific TO alarm - Parameter tab (Page 4227)
Expert **tab**	See Overview of parameters for Acknowledge specific TO alarm - Expert tab (Page 4227)

Overview of parameters for Acknowledge specific technology object alarm - Parameters tab

Table 7-77 Overview of parameters for Acknowledge specific technology object alarm - Parameters tab

Field/Button	Explanation/instructions
Method of operation	Select which alarms on the technology object are to be acknowledged: **Acknowledge all alarms** All alarms on the technology object are acknowledged. **Acknowledge a specific alarm** The alarm specified in the "Alarm No." or "Alarm text" field is acknowledged.
Alarm number	Enter the alarm number. The corresponding text is displayed in the alarm text field.
Alarm text	Select the alarm text. The corresponding number of the alarm is displayed in the alarm number field.

Overview of parameters for Acknowledge specific technology object alarm - Expert tab

Table 7-78 Overview of parameters for Acknowledge specific technology object alarm - Expert tab

Field/button	Meaning/instruction
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Acknowledge specific technology object alarm

Cam technology package:

- _resetAxisError for drive axes, position axes, synchronized axes, and path axes
- _resetCamError for cams
- _resetCamTrackError for cam tracks
- _resetExternalEncoderError for external encoders
- _resetFollowingObjectError for synchronous objects
- _resetMeasuringInputError for measuring inputs
- _resetOutputCamError for output cams

Path technology package:

- _resetPathObjectError for path objects

Cam_EXT technology package:

- _resetAdditionObjectError for addition objects
- _resetControllerObjectError for controller objects
- _resetFixedGearError for fixed gear

- _resetFormulaObjectError for formula objects
- _resetSensorError for sensor objects

TControl technology package:

- _resetTControllerError for temperature channels

Each with parameter errorResetMode = SPECIFIC_ERROR

Incoming message

This command notifies configured nodes that a message has arrived.

The message can be selected as follows:

- By the name configured during message configuration (**Project > Messages > Configure** menu command).
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructAlarmId
  In this form, the command can be used in libraries.
  This variable contains the AlarmId of a configured message. You can obtain the AlarmId outside of libraries with _alarm.*Message* and assign to them a variable of data type StructAlarmId.

The message is sent to all of the configured nodes.

See the online help for information about message configuration.

If the specified message is already pending, then there is no change. The message is not reset.

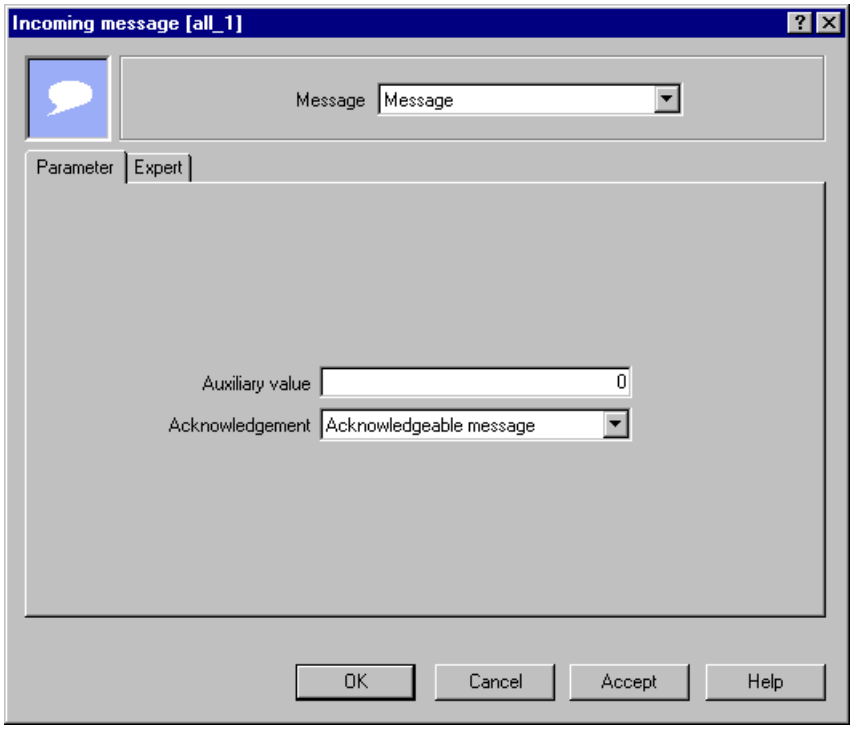


Figure 7-108 Parameter screen form: Incoming message

Overview of parameters for Incoming message

You can set the following parameters:

Table 7-79 Overview of parameters for Incoming message

Field/Button	Explanation/instructions
Alarm	Here you select the message. You can select from: • All configured messages (not in libraries). To configure new messages, select the **Project > Messages > Configure** menu command. • All variables of data type StructAlarmId declared in the MCC source file or MCC chart You can obtain the AlarmId of a configured message outside of libraries with _alarm.*Message* and assign to them a variable of data type StructAlarmId.
Parameters tab	See Overview of parameters for Incoming message - Parameter tab (Page 4230)
Expert tab	See Overview of parameters for Incoming message - Expert tab (Page 4230)

Overview of parameters for Incoming message - Parameters tab

Table 7-80 Overview of parameters for Incoming message - Parameters tab

Field / button	Meaning/instruction
Auxiliary value	Enter the auxiliary values for the message. The auxiliary value must be specified if auxiliary values have been defined during message configuration in SIMOTION SCOUT. • Irrespective of the SIMOTON Kernel version: 1 Auxiliary value possible, integer numbers, numbers with decimal points, and positive and negative values are permitted. Variables or constants of data types ANY_NUM or ANY_BIT can also be specified. • SIMOTION Kernel as of version V4.5: Up to 12 auxiliary values of data type BYTE (or SINT, USINT) possible, these are transferred in a field of 12 BYTE (ARRAY [0..11] OF BYTE). Auxiliary values with data types that include multiple bytes (e.g. WORD, DINT, REAL) are possible, these reduce the number of possible auxiliary values. The user must ensure that their values are saved in the permitted field elements in accordance with the big endian byte arrangement. The function AnyType_to_BigByteArray can be used for this, see Basic Functions Function Manual. For the arrangement of the auxiliary value in the byte field: see section "Example of messages with multiple auxiliary values (as of Kernel V4.5)" in the Basic Functions Function Manual.
Acknowledgment	Here, you select whether the message can be acknowledged. **Acknowledgeable message** (default value) acknowledgeable message: When the message arrives, the user must acknowledge it on the operator panel. **Message not acknowledgeable** Message not acknowledgeable: The message is visible until the program has issued an outgoing message.

Overview of parameters for Incoming message - Expert tab

Table 7-81 Overview of parameters for Incoming message - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DWORD; for a description, see _alarmSId and _alarmSqId functions in the SIMOTION Basic Functions Function Manual.

Relevant system functions for Incoming message

ST system functions:

- _alarmSId (for non-acknowledgeable message)
- _alarmSqId (for acknowledgeable message)

each with parameter sig:= TRUE

The _getAlarmId system function is also called if a configured message has been selected.

Outgoing message

This command notifies configured stations that a message is no longer pending.

The message can be selected as follows:

- By the name configured during message configuration (**Project > Messages > Configure** menu command).
  In this form, the command **cannot** be used in libraries.
- Using a variable of data type StructAlarmId
  In this form, the command can be used in libraries.
  This variable contains the AlarmId of a configured message. You can obtain the AlarmId outside of libraries with _alarm.*Message* and assign to them a variable of data type StructAlarmId.

Acknowledgeable messages can then be acknowledged on the OP.

See the online help for information about message configuration.

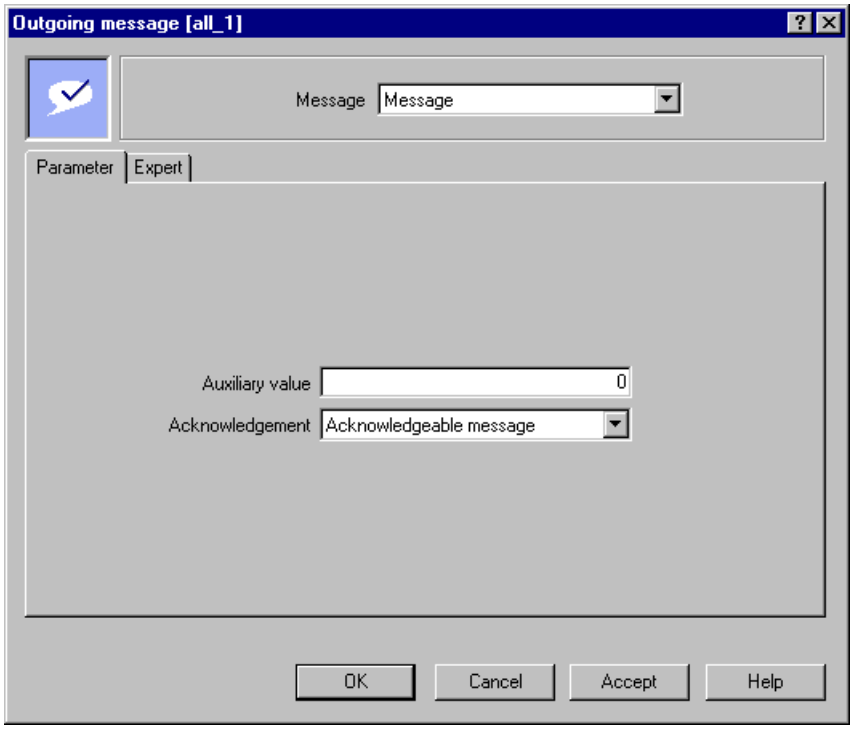


Figure 7-109 Parameter screen form: Outgoing message

Overview of parameters for Outgoing message

You can set the following parameters:

Table 7-82 Overview of parameters for Outgoing message

Field/Button	Explanation/instructions
Alarm	Here you select the message. You can select from: • All configured messages (not in libraries). To configure new messages, select the **Project > Messages > Configure** menu command. • All variables of data type StructAlarmId declared in the MCC source file or MCC chart. You can obtain the AlarmId of a configured message outside of libraries with _alarm.*Message* and assign to them a variable of data type StructAlarmId.
Parameters tab	See Overview of parameters for Outgoing message - Parameter tab (Page 4232)
Expert tab	See Overview of parameters for Outgoing message - Expert tab (Page 4233)

Overview of parameters for Outgoing message - Parameters tab

Table 7-83 Overview of parameters for Outgoing message - Parameters tab

Field / button	Meaning/instruction
Auxiliary value	Enter the auxiliary values for the message. The auxiliary value must be specified if auxiliary values have been defined during message configuration in SIMOTION SCOUT. • Irrespective of the SIMOTON Kernel version: 1 Auxiliary value possible, integer numbers, numbers with decimal points, and positive and negative values are permitted. Variables or constants of data types ANY_NUM or ANY_BIT can also be specified. • SIMOTION Kernel as of version V4.5: Up to 12 auxiliary values of data type BYTE (or SINT, USINT) possible, these are transferred in a field of 12 BYTE (ARRAY [0..11] OF BYTE). Auxiliary values with data types that include multiple bytes (e.g. WORD, DINT, REAL) are possible, these reduce the number of possible auxiliary values. The user must ensure that their values are saved in the permitted field elements in accordance with the big endian byte arrangement. The function AnyType_to_BigByteArray can be used for this, see Basic Functions Function Manual. For the arrangement of the auxiliary value in the byte field: see section "Example of messages with multiple auxiliary values (as of Kernel V4.5)" in the Basic Functions Function Manual.
Acknowledgment	Here, you select whether the message can be acknowledged. **Acknowledgeable message** (default value) acknowledgeable message: If the message has been sent, the user must acknowledge it on the OP. **Message not acknowledgeable** Message not acknowledgeable: The message is visible until the program has issued an outgoing message.

Overview of parameters for Outgoing message - Expert tab

Table 7-84 Overview of parameters for Outgoing message - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DWORD; for a description, see _alarmSId and _alarmSqId functions in the SIMOTION Basic Functions Function Manual.

Relevant system functions for Outgoing message

ST system functions:

- _alarmSId (for non-acknowledgeable message)
- _alarmSqId (for acknowledgeable message)

each with parameter sig:= FALSE

The _getAlarmId system function is also called if a configured message has been selected.

Establish connection using TCP/IP

Before you can use the TCP/IP protocol of the "Send data" (Page 4239) or "Receive data" (Page 4246) commands, you must establish the TCP/IP connection for both communication partners:

1. You specify one communications partner as the server.
   After the call, the command waits for a connection request at the specified port.
2. You specify the other communications partner as the client and specify the server it is to access.
   After the command is called, a connection to the server addressed by the IP address and port number is established.

The connection remains active until it is terminated by the "Remove connection via TCP/IP" (Page 4237) command.

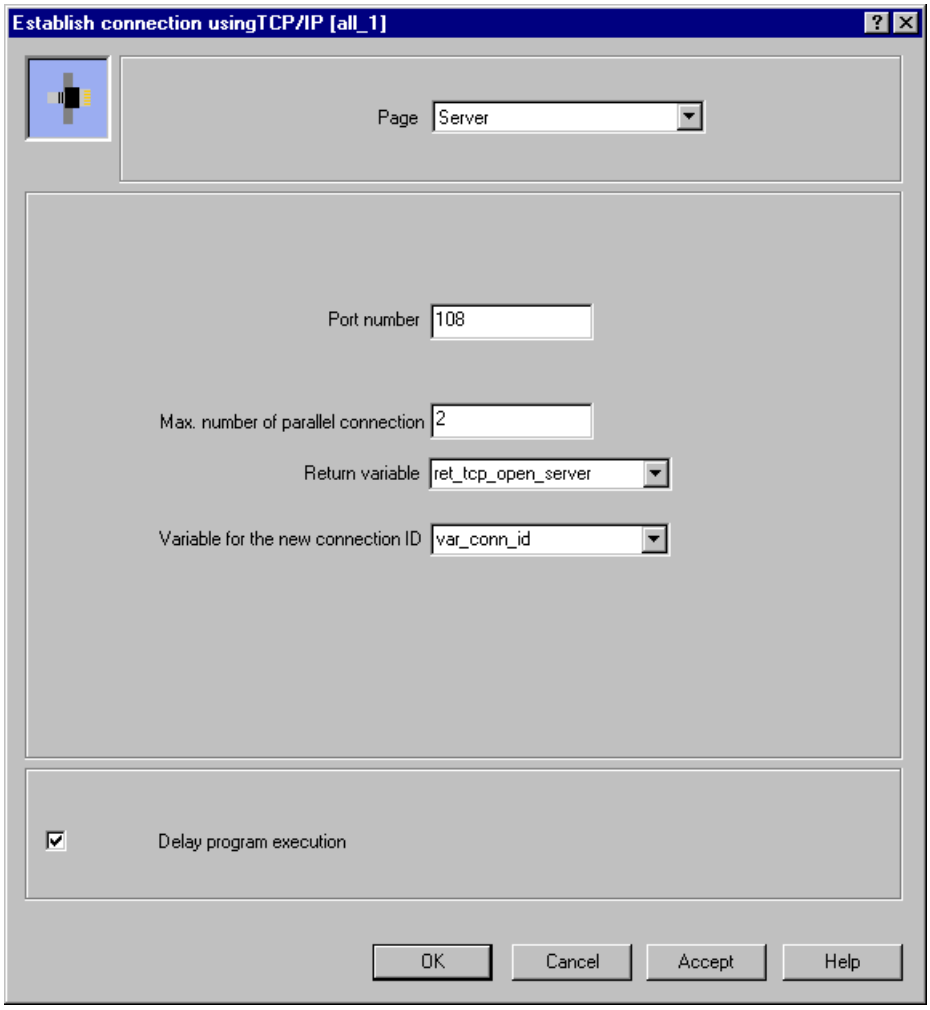


Figure 7-110 Parameter screen form: Establish connection via TCP/IP

Overview of parameters for Establish connection using TCP/IP

You can set the following parameters:

Table 7-85 Overview of parameters for Establish connection via TCP/IP

Field/button	Meaning/instruction
Page	Select whether the SIMOTION device is to act as a server or client in the TCP/IP connection. The parameter dialog box changes according to the selected value. **Server** The SIMOTION device acts as a server in the TCP/IP connection. **Client** The SIMOTION device acts as a client in the TCP/IP connection.
Parameters for	

Field/button		Meaning/instruction
	Page = Server	See Table "Overview of parameters for Establish connection via TCP/IP - Parameters for server"
	Page = Client	See Table "Overview of parameters for Establish connection via TCP/IP - Parameters for client"
Delay program execution		Activate the checkbox if execution of the following command should be delayed until this command has been completed. If the checkbox is not activated, the next command is executed immediately.

Table 7-86 Overview of parameters for Establish connection via TCP/IP - Parameters for server

Field/button	Meaning/instruction
Port number	Here, you enter the number of the port where the command waits for a connection request.
Max. number of parallel connection requests	Here, you enter the maximum number of parallel connection requests.
Return variable	Here, you enter the return variable of data type StructRetTcpOpenServer in which the error message and the connection parameters are stored. This variable must be defined in a declaration table. For the data type StructRetTcpOpenServer: See Return value for Establish connection via TCP/IP (Page 4237).
Variable for new connection ID	Here, you enter a variable of data type DINT. It contains the new connection ID. The value is identical to the connectionId component of the return variable.

Table 7-87 Overview of parameters for Establish connection via TCP/IP - Parameters for client

Field/button	Meaning/instruction
Port number	Here, you enter the port number of the client.
IP address of the server	Here, you enter the IP address of the server.
Port number of the server	Here, you enter the port number of the server.
Return variable	Here, you enter the return variable of data type StructRetTcpOpenClient in which the error message and the connection parameters are stored. This variable must be defined in a declaration table. For the data type StructRetTcpOpenClient: See Return value for Establish connection via TCP/IP (Page 4237).
Variable for new connection ID	Here, you enter a variable of data type DINT. It contains the new connection ID. The value is identical to the connectionId component of the return variable.

Relevant system functions for Establish connection using TCP/IP

System function of SIMOTION devices:

- _tcpOpenServer, if **Page = Server** is selected.
- _tcpOpenClient, if **Page = Client** is selected.

Overview of parameters for Establish connection via TCP/IP (Page = Server) / _tcpOpenServer

Table 7-88 Parameters (MCC command Establish connection via TCP/IP compared to system function _tcpOpenServer)

Parameters of the MCC command Establish connection via TCP/IP	Parameters of the system function _tcpOpenServer
Page	-
Port number	port
Max. number of parallel connection requests	backlog
Return variable	-
Variable for new connection ID	connectionId component of the return value (Page 4237)
Delay program execution	nextCommand

Overview of parameters for Establish connection via TCP/IP (Page = Client) / _tcpOpenClient

Table 7-89 Parameters (MCC command Establish connection via TCP/IP compared to system function _tcpOpenClient)

Parameters of the MCC command Establish connection via TCP/IP	Parameters of the system function _tcpOpenClient
Page	-
Port number	port
IP address of the server	serverAddress
Port number of the server	serverPort
Return variable	-
Variable for new connection ID	connectionId component of the return value (Page 4237)
Delay program execution	nextCommand

Return value for Establish connection using TCP/IP

The return value informs the user about the result of the command call and contains important connection parameters.

You must declare the return variable in a declaration table:

- If parameter **Page = Server** is selected:
  As variable of the StructRetTcpOpenServer data type.
- If parameter **Page = Client** is selected:
  As variable of the StructRetTcpOpenClient data type.

You select this variable in the **Return variable** parameter, see Overview of parameters for Establish connection via TCP/IP (Page 4234).

The following table lists the meaning of the individual elements of the StructRetTcpOpenServer and StructRetTcpOpenClient data types.

Table 7-90 Structure of the return value for Page = Server (TYPE StructRetTcpOpenServer)

Parameter (data type)	Meaning/values	Values
functionResult (DINT)	Error number	< **16#8000:** if command execution is okay >= **16#8000:** if an error has occurred See description for _tcpOpenServer command in the parameter manuals for the SIMOTION devices.
connectionId (DINT)	New connection ID	-
clientAddress (ARRAY [0..3] OF USINT)	IP address of the connecting client	-
clientPort (UINT)	Port number of the connecting client	-

Table 7-91 Structure of the return value for Page = Client (TYPE StructRetTcpOpenClient)

Parameter (data type)	Meaning/values	Values
functionResult (DINT)	Error number	< **16#8000:** if command execution is okay >= **16#8000:** if an error has occurred See description for _tcpOpenClient command in the parameter manuals for the SIMOTION devices.
connectionId (DINT)	New connection ID	-

Remove connection using TCP/IP

This command is used to cancel a TCP/IP connection.

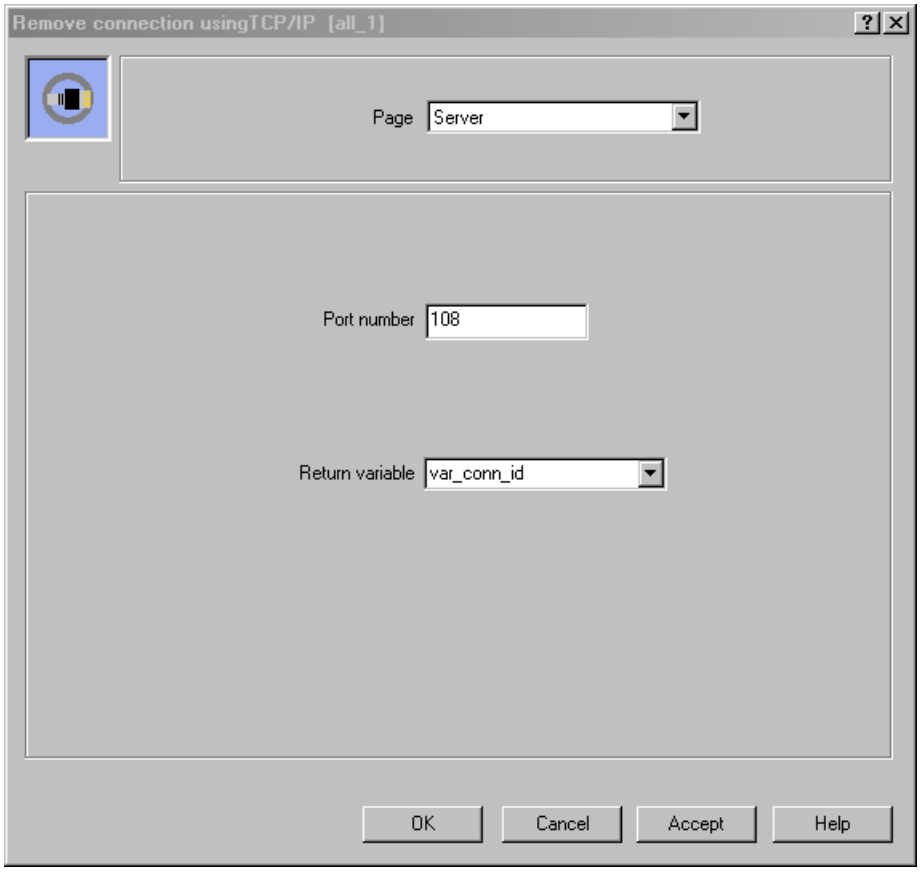


Figure 7-111 Parameter screen form: Remove connection via TCP/IP

Overview of parameters for Remove connection using TCP/IP

You can set the following parameters:

Table 7-92 Overview of parameters for Remove connection via TCP/IP

Field/button	Meaning/instruction
Page	Select whether you want to cancel an existing server connection or client connection. The parameter dialog box changes according to the selected value. **Server** A server connection is canceled. **Client** A client connection is canceled.
Port number	Only when **Page = Server** is selected. Here, you enter the number of the port where the command waits for a connection request.

Field/button	Meaning/instruction
Return variable	Enter the return variable of data type DINT in which the error message will be stored. This variable must be defined in a declaration table or the symbol browser. < **16#8000:** if command execution is okay >= **16#8000**: if an error has occurred See description for _tcpCloseServer and _tcpCloseConnection functions in the parameter manuals for the SIMOTION devices.
Connection ID	Only when **Page = Client** is selected. Here, you enter a variable of data type DINT that contains the connection ID of the connection to be removed.
Delay program execution	Activate the checkbox if execution of the following command should be delayed until this command has been completed. If the checkbox is not activated, the next command is executed immediately.

Relevant system functions for Remove connection using TCP/IP

System function of SIMOTION devices:

- _tcpCloseServer, if **Page = Server** is selected.
- _tcpCloseConnection, if **Page = Client** is selected.

Overview of parameters for Remove connection via TCP/IP (Page = Server) / _tcpCloseServer

Table 7-93 Parameters (MCC command Remove connection via TCP/IP compared to system function _tcpCloseServer)

Parameters of the MCC command Remove connection via TCP/IP	Parameters of the system function _tcpCloseServer
Page	-
Port number	port
Return variable	-

Overview of parameters for Remove connection via TCP/IP (Page = Client) / _tcpCloseConnection

Table 7-94 Parameters (MCC command Remove connection via TCP/IP compared to system function _tcpCloseConnection)

Parameters of the MCC command Remove connection via TCP/IP	Parameters of the system function _tcpCloseConnection
Page	-
Return variable	-
Connection ID	connectionId

Send data

This command sends data to a communications partner (such as a SIMATIC S7 station) using various protocols.

- XSend/XReceive protocol
  The connection to the communications partner is established via PROFIBUS or MPI and does not have to be configured.
  SIMOTION devices can send a maximum of 200 bytes in one unit; the actual user data length is dependent on the communications partner.
- UDP protocol
  The connection is established via Ethernet with the UDP protocol.
  SIMOTION devices can send a maximum of 1470 bytes in one unit; the actual user data length is dependent on the communications partner.
- TCP/IP protocol
  The connection is established via Ethernet with the TCP/IP protocol. You must have configured the TCP/IP connection beforehand. Use the "Establish connection via TCP/IP" (Page 4233) command.
  SIMOTION devices can send a maximum of 4096 bytes in one unit; the actual user data length is dependent on the communications partner.

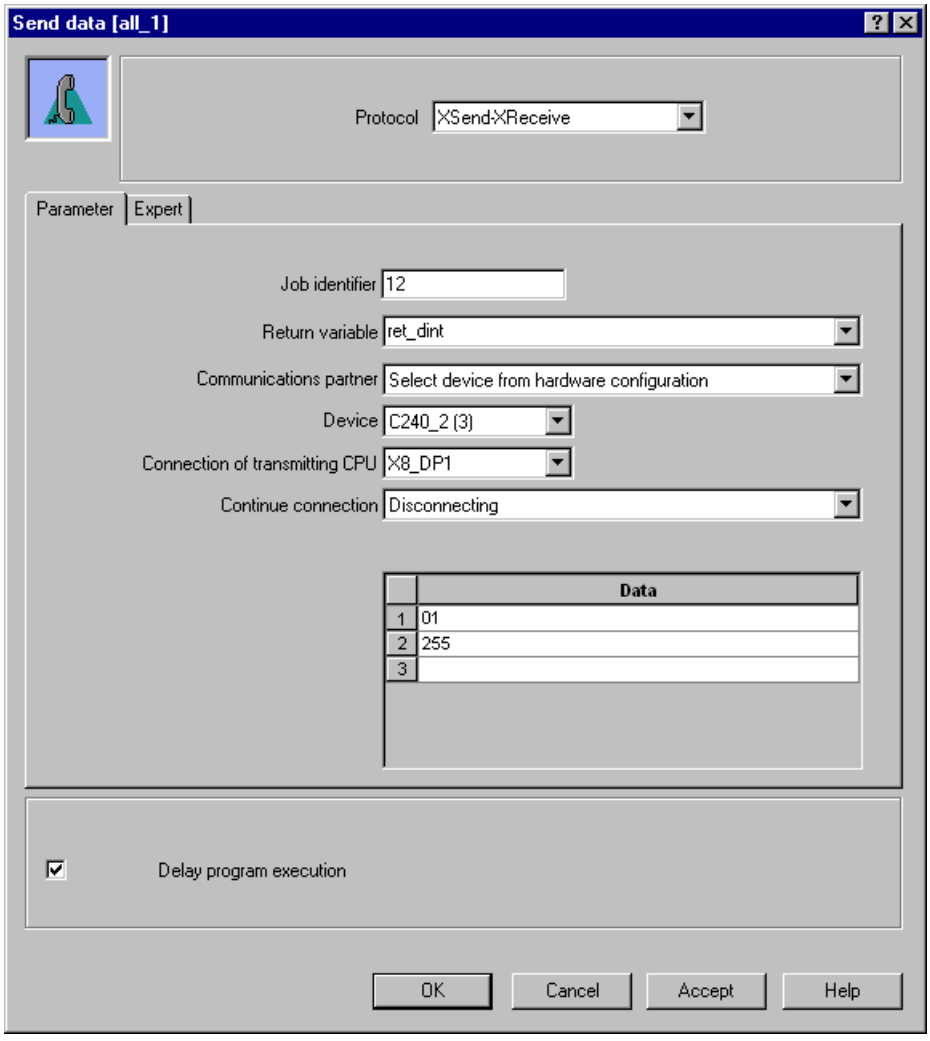


Figure 7-112 Parameter screen form: Send data

Overview of parameters for Send data

You can set the following parameters:

Table 7-95 Overview of parameters for Send data

Field/button	Meaning/instruction
Protocol	You can choose between the different protocols here. The parameter dialog box changes, accordingly. **XSend-XReceive** The connection is established via PROFIBUS or MPI. **UDP** The connection is established via Ethernet, and the UDP protocol is selected. **TCP/IP** The TCP/IP protocol is selected. You must have configured the TCP/IP connection beforehand. Use the "Establish connection via TCP/IP" (Page 4233) command.
Parameters tab	• For **Protocol = XSend-XReceive** See Table Overview of parameters for Send data - Parameters tab for Protocol = XSend-XReceive (Page 4241) • For **Protocol = UDP** See Table Overview of parameters for Send data - Parameters tab for Protocol = UDP (Page 4242) • For **Protocol = TCP/IP** See Table Overview of parameters for Send data - Parameters tab for Protocol = TCP/IP (Page 4243)
Expert tab	See Overview of parameters for Send data - Expert tab (Page 4243)
Delay program execution	Only when **Protocol = XSend-XReceive** or **Protocol = TCP/IP** is selected. • Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately. If **Protocol = UDP** is selected, the next command is immediately processed.

Overview of parameters for Send data - Parameters tab

Table 7-96 Overview of parameters for Send data - Parameters tab for Protocol = XSend-XReceive

Field/button	Meaning/instruction
Job identifier	Enter the job identifier here. The data type is UDINT.
Return variable	If you enter the name of a variable of the data type DINT, you can use this variable to find out the result of the command call. This variable must be defined in a declaration table or the symbol browser. < **16#8000**: if command execution is okay **>= 16#8000**: if an error has occurred See description for _Xsend function in the List Manuals for the SIMOTION Devices.

Field/button	Meaning/instruction
Communications partner	Here, you select the communications partner to which you wish to send data. **Specify address directly** This setting enables you to enter the communications partner's address directly. Use this selection if the communications partner is not defined in the project (for example, SIMATIC S7 stations). **Select device from hardware configuration** (default value) This setting enables you to select a SIMOTION device from the current hardware configuration. This setting cannot be used in libraries.
Device	This selection is only active if **Select device from hardware configuration** is selected in the "Communications partner" selection field. Here, you select the receiving device. You can choose from all SIMOTION devices that can be reached directly by means of a PROFIBUS or MPI subnet.
Connection of transmitting CPU	This selection is only active if **Select device from hardware configuration** is selected in the "Communications partner" selection field. Here, you select the connection of the transmitting SIMOTION device. Once the receiving device (communications partner) is selected, the PROFIBUS or MPI connection of the transmitting SIMOTION device is offered.
Destination address of communications partner	This selection is active only if **Specify address directly** is active in the "Communications partner" list. Here, you enter the name of the variable in which the destination address of the communications partner is stored. You define the variable in a declaration table as StructXSendDestAddr (XSend-XReceive protocol).
Continue connection	Here, decide whether the connection should remain after data transfer. Select **Disconnecting** if the device in question is communicating with a large number of nodes. **Disconnecting** (default value) The connection is discontinued after the data transfer. **Connection remains active** The connection remains active after the data transfer.
Data	Enter the communication data here. You can enter up to 200 values. The data type must be BYTE; furthermore, variables of type BYTE can be entered.

Table 7-97 Overview of parameters for Send data - Parameters tab for Protocol = UDP

Field/button	Meaning/instruction
Port number of sender	Here, you enter the port number of the transmitting SIMOTION device.
IP address of receiver	Here, you enter the IP address of the receiver.
Port number of receiver	Here, you enter the port number of the receiver.

Field/button	Meaning/instruction
Socket lifetime	Here, decide whether the connection should remain after data transfer. Select Socket will be closed if the selected device communicates with many nodes. **Socket remains** (default value) The connection remains active after the data transfer. **Close socket** The connection is discontinued after the data transfer.
Data	Enter the communication data here. You can enter up to 1470 values. The data type must be BYTE; furthermore, variables of type BYTE can be entered.

Table 7-98 Overview of parameters for Send data – Parameters tab for Protocol = TCP/IP

Field/button	Meaning/instruction
Connection ID	Here, you enter the connection ID. This value was obtained when the TCP/IP connection was established with the "Establish connection via TCP/IP" (Page 4233) command.
Data	Enter the communication data here. You can enter up to 4096 values. The data type must be BYTE; furthermore, variables of type BYTE can be entered.

Overview of parameters for Send data - Expert tab

Table 7-99 Overview of parameters for Send data – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	Only when **Protocol = XSend-XReceive** is selected If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	• When **Protocol = UDP** or **Protocol = TCP/IP** is selected If you enter the name of a variable of the data type DINT, you can use this variable to find out the result of the command call. < **16#8000:** if command execution is okay >= **16#8000:** if an error has occurred See description for _udpSend and _tcpSend functions in the parameter manuals for the SIMOTION devices. • When **Protocol = XSend-XReceive** is selected If you have specified a return variable in the Parameters (Page 4241) tab, it is displayed.

Querying the command status for Send data with XSend-XReceive protocol

When the XSend/XReceive protocol is used, you can query the status of the command on the basis of the CommandID variable using the _GetStateOfXCommand system function (see List Manuals for SIMOTION Devices). This is required, in particular, if the **Delay program execution** parameter is deactivated. In this way you can avoid an overflow of the send buffer due to multiple calls of the command.

ST commands can be programmed using the ST zoom (Page 4188), for example.

You cannot query the command status in the UDP or TCP/IP protocols.

Structure of destination address of the communications partner when address is entered directly (XSend-XReceive protocol)

When you enter the destination address of the communications partner directly in the XSend/XReceive protocol, you must declare a variable of data type StructXSendDestAddr in a declaration table or in the symbol browser.

The following table lists the meaning of the individual elements of these data types.

Table 7-100 Structure of the destination address of the communications partner when the address is entered directly (TYPE StructXSendDestAddr – XSend-XReceive protocol)

Parameter (data type)	Meaning	Values	
deviceId (USINT)	Point of the connection	C230-2, C240, C240 PN	**1** for X8 **2** for X9
		P350, P350-3	**1** for X101 **2** for X102
		D410 DP	**1** for X21
		D410-2 DP	**2** for X21 **1** for X24
		D410-2 DP/PN	**2** for X21
		D4x5, D4x5-2	**1** for X126 **2** for X136
remoteSubnetIdLength (USINT)	Length of the subnet dialog box	**0** for MPI, PROFIBUS	
remoteStaddrLength (USINT)	Length of the station address (station number) of the destination system.	**1** for MPI, PROFIBUS	
nextStaddrLength (USINT)	Length of the router address	**0** for MPI, PROFIBUS	
remoteSubnetId (ARRAY [0..5] OF USINT)	Subnet mask	(Irrelevant for MPI, PROFIBUS)	
remoteStaddr (ARRAY [0..5] OF USINT)	Station address of the destination system (actual destination address)	Node number for MPI, PROFIBUS: e.g. remoteStaddr[0] = 25	
nextStaddr (ARRAY [0..5] OF USINT)	Router address	(Irrelevant for MPI, PROFIBUS)	

Relevant system functions for Send data

System function of SIMOTION devices:

- _Xsend, if **Protocol = XSend-XReceive** is selected
- _udpSend, if **Protocol = UDP** is selected
- _tcpSend, if **Protocol = TCP/IP** is selected

Overview of parameters for Send data (XSend-XReceive protocol) / _Xsend

Table 7-101 Parameters (MCC command Send data compared to system function _Xsend)

Parameters of the MCC command Send data	Parameters of the system function _Xsend
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Job identifier	messageId
Return variable	-
Communications partner	-
Device	address.remoteStaddr[0]
Connection of transmitting PCU	address.deviceId
Destination address of communications partner	address
Continue connection	communicationMode
Data	dataLength, data
Expert tab	
CommandID variable	commandId

Overview of parameters for Send data (TCP/IP protocol) / _tcpSend

Table 7-102 Parameters (MCC compared Send data command to system function _tcpSend)

Parameters of the MCC command Send data	Parameters of the system function _tcpSend
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Connection ID	connectionId
Data	dataLength, data
Expert tab	
Return variable	-

Overview of parameters for Send data (TCP/IP protocol) / _udpSend

Table 7-103 Parameters (MCC command Send data compared to system function _udpSend)

Parameters of the MCC command Send data	Parameters of the system function _udpSend
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Port number of sender	sourcePort
IP address of receiver	destinationAddress

Parameters of the MCC command Send data	Parameters of the system function _udpSend
Port number of receiver	destinationPort
Data	dataLength, data
Expert tab	
Return variable	-

Receive data

This command is used to receive data from a communications partner (such as a SIMATIC S7 station) using various protocols.

- XSend-XReceive protocol
  The connection to the communications partner is established via PROFIBUS or MPI and does not have to be configured.
  SIMOTION devices can receive a maximum of 200 bytes in one unit; the actual user data length is dependent on the communications partner.
- UDP protocol
  The connection is established via Ethernet with the UDP protocol.
  SIMOTION devices can receive a maximum of 1470 bytes in one unit; the actual user data length is dependent on the communications partner.
- TCP/IP protocol
  The connection is established via Ethernet with the TCP/IP protocol. You must have configured the TCP/IP connection beforehand. Use the "Establish connection via TCP/IP" (Page 4233) command.
  SIMOTION devices can receive a maximum of 4096 bytes in one unit; the actual user data length is dependent on the communications partner.

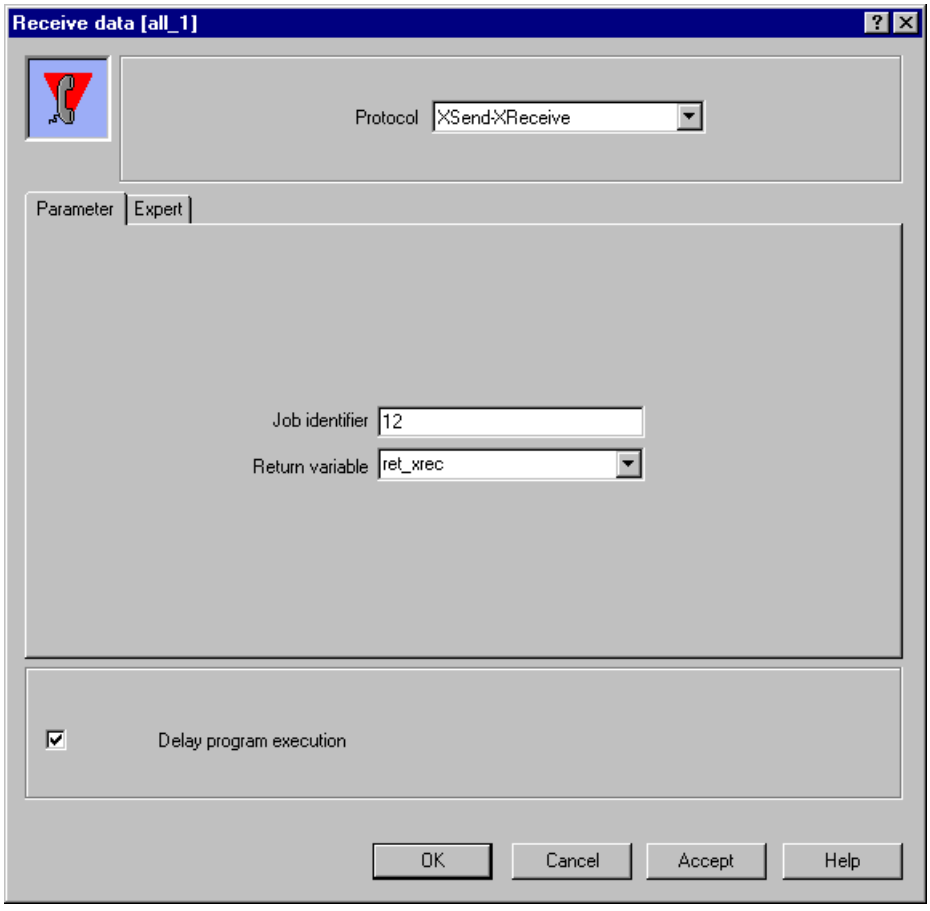


Figure 7-113 Parameter screen form: Receive data

Overview of parameters for Receive data

You can set the following parameters:

Table 7-104 Overview of parameters for Receive data

Field/button	Meaning/instruction
Protocol	You can choose between the different protocols here. The parameter dialog box changes, accordingly. **XSend-XReceive** The connection is established via PROFIBUS or MPI. **UDP** The connection is established via Ethernet, and the UDP protocol is selected. **TCP/IP** The TCP/IP protocol is selected. You must have configured the TCP/IP connection beforehand. Use the **Establish connection via TCP/IP** command.
Parameters tab	• For **Protocol = XSend-XReceive** See Table Overview of parameters for Receive data - Parameters tab for Protocol = XSend-XReceive (Page 4249) • For **Protocol = UDP** See Table Overview of parameters for Receive data - Parameters tab for Protocol = UDP (Page 4249) • For **Protocol = TCP/IP** See Table Overview of parameters for Receive data - Parameters tab for Protocol = TCP/IP (Page 4249)
Expert tab	See Overview of parameters for Receive data - Expert tab (Page 4250)
Delay program execution	Only when **Protocol = XSend-XReceive** or **Protocol = TCP/IP** is selected. • Activate the checkbox if execution of the following command should be delayed until this command has been completed. If the checkbox is not activated, the next command is executed immediately. Only when **Protocol = UDP** is selected • Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **Synchronous** The next command is executed after the current command has been completed or aborted. **Abort** The active command is aborted if the port number is the same as that of the previous call of the command. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Receive data - Parameters tab

Table 7-105 Overview of parameters for Receive data – Parameters tab for Protocol = XSend-XReceive

Field/button	Meaning/instruction
Job identifier	Enter the job identifier here. The data type is UDINT.
Return variable	Here, you enter the return variable of data type StructRetXReceive in which the data, length and error message are stored. This variable must be defined in a declaration table or the symbol browser.

Table 7-106 Overview of parameters for Receive data - Parameters tab for Protocol = UDP

Field/button	Meaning/instruction
Return variable	Here, you enter the return variable of data type StructRetUdpReceive in which the connection parameters, length and error message are stored. This variable must be defined in a declaration table or the symbol browser.
Port number of receiver	Here, you enter the port number of the receiver.
Socket lifetime	Here, decide whether the connection should remain after data transfer. Select **Socket will be closed** if the selected device communicates with many nodes. **Socket remains** (default value) The connection remains active after the data transfer. **Close socket** The connection is discontinued after the data transfer.
Received user data bytes	Here, you enter the variable of data type ARRAY[0.0.1469] OF BYTE in which the received user data is stored. The number of bytes of received user data is available in the dataLength component of the return variable.

Table 7-107 Overview of parameters for Receive data – Parameters tab for Protocol = TCP/IP

Field/button	Meaning/instruction
Return variable	Here, you enter the return variable of data type StructRetTcpReceive in which the connection parameters, length and error message are stored. This variable must be defined in a declaration table or the symbol browser.
Connection ID	Here, you enter the connection ID. This value was obtained when the TCP/IP connection was established with the "Establish connection via TCP/IP" (Page 4233) command.
Received user data bytes	Here, you enter the variable of data type ARRAY[0..4095] OF BYTE in which the received user data is stored. The number of bytes of received user data is available in the dataLength component of the return variable.

Overview of parameters for Receive data - Expert tab

Table 7-108 Overview of parameters for Receive data – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	Only when **Protocol = XSend-XReceive** is selected If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.

Querying the command status for Receive data with XSend-XReceive protocol

When the XSend/XReceive protocol is used, you can query the status of the command on the basis of the CommandID variable using the _GetStateOfXCommand system function (see List Manuals for SIMOTION Devices). This is required, in particular, if the **Delay program execution** parameter is deactivated. In this way you can avoid an overflow of the send buffer due to multiple calls of the command.

ST commands can be programmed using the ST zoom (Page 4188), for example.

You cannot query the command status in the UDP or TCP/IP protocols.

Relevant system functions for Receive data

System function of SIMOTION devices:

- _Xreceive, if **Protocol = XSend- XReceive** is selected
- _udpReceive, if **Protocol = UDP** is selected
- _tcpReceive, if **Protocol = TCP/IP** is selected

Overview of parameters for Receive data (XSend- XReceive protocol), _Xreceive

Table 7-109 Parameters (MCC command Receive data compared to system function _Xreceive)

Parameters of the MCC command Receive data	Parameters of the system function _Xreceive
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Job identifier	messageId
Return variable	-
Expert tab	
CommandID variable	commandId

Overview of parameters for Receive data (TCP/IP protocol), _tcpReceive

Table 7-110 Parameters (MCC command Receive data compared to system function _tcpReceive)

Parameters of the MCC command Receive data	Parameters of the system function _tcpReceive
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Return variable	-
Connection ID	connectionId
Received user data bytes	receiveVariable
Expert tab	
Return variable	-

Overview of parameters for Receive data (UDP protocol), _udpReceive

Table 7-111 Parameters (MCC command Receive data compared to system function _udpReceive)

Parameters of the MCC command Receive data	Parameters of the system function _udpReceive
Protocol	-
Delay program execution	nextCommand
Parameters tab	
Return variable	-
Port number of receiver	port
Socket lifetime	communicationMode
Received user data bytes	receiveVariable
Expert tab	
Return variable	-

Return value for Receive data

The return value informs the user about the result of the command call and contains important connection parameters.

You must declare the return variable as a variable of the following data types in a declaration table or the symbol browser and specify it when assigning the command parameters.

- StructRetXReceive, if **Protocol = XSend-XReceive** is selected
- StructRetUdpReceive, if **Protocol = UDP** is selected
- StructRetTcpReceive, if **Protocol = TCP/IP** is selected

The following table lists the meaning of the individual elements of these data types.

Table 7-112 Structure of the return value (TYPE StructRetXReceive) for XSend-XReceive protocol

Parameter (data type)	Meaning/values	Values
functionResult (DINT)	Error number	**< 16#8000:** if command execution is okay **>= 16#8000:** if an error has occurred See description for _Xreceive command in the parameter manuals for the SIMOTION devices.
dataLength (UDINT)	Length of received data packet	
data (ARRAY [0..199] OF BYTE)	Received data	

Table 7-113 Structure of the return value (TYPE StructRetUdpReceive) for UDP protocol

Parameter (data type)	Meaning/values	Values
functionResult (DINT)	Error number	**< 16#8000:** if command execution is okay **>= 16#8000:** if an error has occurred See description for the _udpReceive command in the parameter manuals for the SIMOTION devices.
sourceAddress (ARRAY [0..3] OF USINT)	IP address of transmitter	
sourcePort (UINT)	Port number of transmitter	
dataLength (UDINT)	Length of received data packet	

Table 7-114 Structure of the return value (TYPE StructRetTcpReceive) for TCP/IP protocol

Parameter (data type)	Meaning/values	Values
functionResult (DINT)	Error number	**< 16#8000:** if command execution is okay >= **16#8000:** if an error has occurred See description for the _tcpReceive command in the parameter manuals for the SIMOTION devices.
dataLength (UDINT)	Length of received data packet	

7.1.6.5 Single axis commands

Switch axis enable

You use this command to switch the enables on an axis with an electric drive (for axes with a hydraulic drive, you use the Switch QF-axis enable function).

The force/pressure control can also be enabled or disabled on axes with force/pressure control.

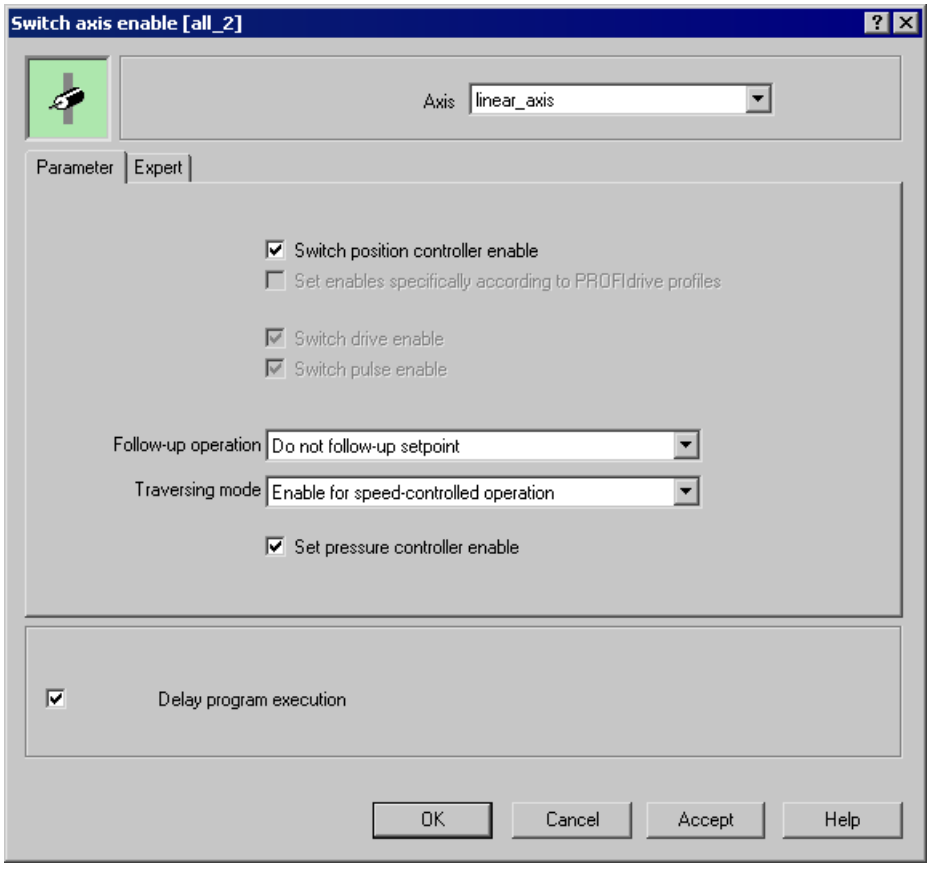


Figure 7-114 Parameter screen form: Switch axis enable

The following conditions must be fulfilled in order to execute motion commands on the axis:

1. Drive enable issued
2. Pulse enabled (power unit enabled)
3. Additional conditions for position, synchronized and path axes: Position controller enable is issued
4. Follow-up mode canceled

Until all conditions are fulfilled, the axis remains in follow-up mode (see Follow-up mode for various axis technologies).

The current status of enables can be accessed via system variables, which are specified in the description of the respective parameter.

Table 7-115 Follow-up mode for various axis technologies

Drive axis	**Position axis** **Synchronous axis** **Path axis**
Note Motion commands for the axis cannot be executed in follow-up mode.	
	The following applies to position control: Position control is canceled. The actual position value continues to be measured. The position setpoint is corrected to the current actual position value. **Note** Once follow-up mode is canceled, homing of the axis does not have to be repeated.
Current status for a real axis, indicating whether motion commands can be executed: System variable: control.	

For additional information on **axis enables**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Switch axis enable

Table 7-116 Overview of parameters for Switch axis enable

Field/button	**Meaning/instruction**
Axis	In Axis, select the axis for which the axis enable is to be switched. The list contains: • All axes with an electric drive that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Switch axis enable – Parameters tab (Page 4255)
Expert tab	See Overview of parameters for Switch axis enable – Expert tab (Page 4256)
Delay program execution	Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Switch axis enable – Parameters tab

Table 7-117 Overview of parameters for Switch axis enable – Parameters tab

Field/button	Meaning/instruction
Switch position controller enable	Only for position, synchronized and path axes: Activate the checkbox if the position controller enable is to be switched. If the checkbox is activated: • The **Switch enables individually according to PROFIdrive profiles** checkbox is deactivated. • The **Switch drive enable** and **Switch pulse enable** checkboxes are activated. If the checkbox is not activated, the current status of the position controller enable is not changed. In the case of virtual axes, the position controller enable is always set, even if this checkbox has not been activated. The position controller enable must also be set if only speed-controlled operation has been selected for a position-controlled axis, see parameter **Traversing mode**. Current status of position controller enable for a real axis: System variable: servoMonitorings.controlState.
Switch enables individually according to PROFIdrive profiles	Activate the checkbox if you want to be able to switch enables individually in accordance with the PROFIdrive profile. If the checkbox is activated: • The **Switch drive enable** and **Switch pulse enable** checkboxes are hidden. • Seven check boxes are displayed for setting the individual bits of control word 1 (STW1) in accordance with PROFIdrive, see Meaning of bits of control word 1 (STW1) in accordance with PROFIdrive (Page 4257). – Activate the checkbox if you want to assign the corresponding enable. – Deactivate the checkbox if you want to retain the previous setting (enable assigned or not assigned) for the corresponding enable. All enables must be assigned for normal operation of the drive. If the checkbox is deactivated, the **Switch drive enable** and **Switch pulse enable** checkboxes are displayed.
Switch drive enable	Only if the **Switch enables individually according to PROFIdrive profiles** checkbox is deactivated. Activate the checkbox if the drive enable is to be switched. If the checkbox is not activated, the current status of the drive enable is not changed. Current status of drive enable for a real axis: System variable: actorMonitorings.driveState.
Switch pulse enable	Only if the **Switch enables individually according to PROFIdrive profiles** checkbox is deactivated. Activate the checkbox if the pulse enable (power enable) is to be switched. If the checkbox is not activated, the current status of the pulse enable is not changed. Current status of pulse enable for a real axis: System variable: actorMonitorings.power.

Field/button	Meaning/instruction
Follow-up mode	Here, you decide whether the axis is to be switched out of follow-up mode, see Table Follow-up mode for various axis technologies (Page 4257). **Do not follow up setpoint** (default value) Follow-up mode is disabled. Motion commands for the axis can be executed. In real axes, **Do not follow up setpoint** is only effective if all the other enables have been assigned. **Follow up setpoint** Follow-up mode is enabled. Motion commands for the axis cannot be executed. Current status for a real axis, indicating whether motion commands can be executed: System variable: control.
Traversing mode	**Maintain last setting** (default value) The axis is enabled with the most last set traversing mode (position and speed control). **Enable for speed- and position-controlled operation** The axis is enabled for speed- and position-controlled operation. This setting cannot be selected for drive axes. **Enable for speed-controlled operation** The axis is enabled for speed-controlled operation. If only speed-controlled operation is selected for a position-controlled axis the position controller enable still has to be set.
Set pressure controller enable	For axes with force/pressure control only. Activate the checkbox if you want to enable force/pressure control. If the checkbox is not activated, force/pressure control is disabled.

See also

Switch axis enable (Page 4253)

Overview of parameters for Switch axis enable – Expert tab

Table 7-118 Overview of parameters for Switch axis enable – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Meaning of bits of control word 1 (STW1) in accordance with PROFIdrive

The following table indicates the meaning of the bits of control word 1 (STW1) for speed-controlled drives in accordance with PROFIdrive.

Table 7-119 Meaning of bits of control word 1 (STW1) in accordance with PROFIdrive

Bit STW1	MCC command 'Switch axis enable'	MCC command 'Remove axis enable'	MCC command 'Switch axis enable' / 'Remove axis enable'
	Check box selected Enable assigned - the corresponding bit of the control word is set to 1	**Check box selected** Remove enable - the corresponding bit of the control word is set to 0	**Check box cleared** the previous setting is retained - the corresponding bit of the control word remains unchanged
0	ON	Brakes on the ramp-function generator (OFF1)	Retain
1	Cancel coasting down (no OFF2)	Coasting down (OFF2)	Retain
2	Cancel quick stop (no OFF3)	Quick stop (OFF3)	Retain
3	Enable operation	Disable operation	Retain
4	Enables the ramp-function generator	Disable ramp-function generator	Retain
5	Do not freeze ramp-function generator	Freeze ramp-function generator	Retain
6	Enable setpoint	Disable setpoint	Retain

Relevant system function for Switch axis enable

Cam technology package:

- _enableAxis

Overview of parameters for Switch axis enable /_enableAxis

Table 7-120 Parameters (MCC command Switch axis enable compared to system function _enableAxis)

Parameters of the MCC command Switch axis enable	Parameters of the system function _enableAxis
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Switch position controller enable	enableMode, servoControlMode
Switch enables individually according to PROFIdrive profiles	enableMode, stwbitset
Switch drive enable	enableMode
Switch pulse enable	enableMode
Follow-up mode	servoCommandToActualMode
Traversing mode	movingMode
Set pressure controller enable	forcecontrolMode
Expert tab	
CommandID variable	commandId
Return variable	–

Remove axis enable

You use this command to remove the enables from an axis with an electric drive (for axes with a hydraulic drive, you use the Remove QF axis enable function).

The force/pressure control can also be disabled or enabled on axes with force/pressure control.

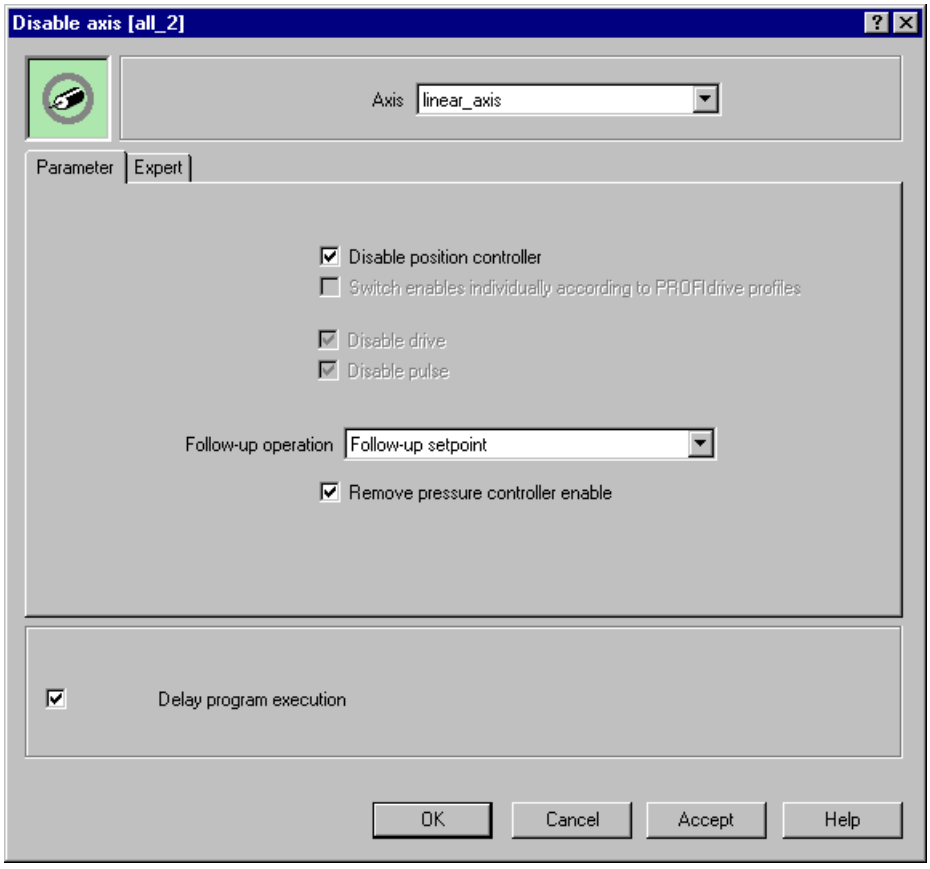


Figure 7-115 Parameter screen form: Remove axis enable

The following conditions must be fulfilled in order to execute motion commands on the axis:

1. Drive enable issued
2. Pulse enabled (power unit enabled)
3. Additional conditions for position, synchronized and path axes: Position controller enable is issued
4. Follow-up mode canceled

Until all conditions are fulfilled, the axis remains in follow-up mode (see Follow-up mode for various axis technologies (Page 4253) table).

The current status of enables can be accessed via system variables, which are specified in the description of the respective parameter (see Overview of parameters for Remove axis enable (Page 4259)).

For additional information on **axis enables**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Remove axis enable

Table 7-121 Overview of parameters for Remove axis enable

Field/button	Meaning/instruction
Axis	In Axis, select the axis for which the command is to be programmed. The list contains: • All axes with an electric drive that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis or _PathAxis
Parameters tab	See Overview of parameters for Remove axis enable – Parameters tab (Page 4260)
Expert tab	See Overview of parameters for Remove axis enable – Expert tab (Page 4261)
Delay program execution	Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Remove axis enable – Parameters tab

Table 7-122 Overview of parameters for Remove axis enable – Parameters tab

Field/Button	Explanation/instructions
Remove pos. controller enable	Only for position, synchronized and path axes: Activate the check box if the position controller enable is to be removed. If the check box is activated: • The **Switch enables individually according to PROFIdrive profiles** check box is cleared. • The **Remove drive enable** and **Remove pulse enable** check boxes are activated. If the check box is not activated, the current status of the position controller enable is not changed. In the case of virtual axes, the position controller enable is always set, even if this checkbox has not been activated. Current status of position controller enable for a real axis: System variable: servoMonitorings.controlState.
Switch enables individually according to PROFIdrive profiles	Activate the check box if you want to be able to switch enables individually in accordance with the PROFIdrive profile. If the check box is activated: • The **Remove drive enable** and **Remove pulse enable** check boxes are hidden. • Seven check boxes are displayed for deleting the individual bits of control word 1 (STW1) in accordance with PROFIdrive (see Meaning of bits of control word 1 (STW 1) in accordance with PROFIdrive). – Enable the check box if you want to remove the corresponding enable. – Clear the check box if you want to retain the previous setting (enable assigned or not assigned) for the corresponding enable. All enables must be assigned for normal operation of the drive. If the check box is cleared, the **Remove drive enable** and **Remove pulse enable** check boxes are displayed.
Disable drive	Only if the **Switch enables individually according to PROFIdrive profiles** check box is cleared. Activate the check box if the drive enable is to be removed. If the check box is not activated, the current status of the drive enable is not changed. Current status of drive enable for a real axis: System variable: actorMonitorings.driveState.
Disable pulse	Only if the **Switch enables individually according to PROFIdrive profiles** check box is cleared. Activate the check box if the pulse enable (power enable) is to be removed. If the check box is not activated, the current status of the pulse enable is not changed. Current status of drive enable for a real axis: System variable: actorMonitorings.power.

Field/Button	Explanation/instructions
Follow-up mode	Here, you decide whether the axis is to be switched into follow-up mode (see Table Follow-up mode for various axis technologies (Page 4253)). **Follow up setpoint** (default value) Follow-up mode is enabled. Motion commands for the axis cannot be executed. **Do not follow up setpoint** Follow-up mode is disabled. Motion commands for the axis can be executed. In real axes, **Do not follow up setpoint** is only effective if all the other enables have been assigned. Current status for a real axis, indicating whether motion commands can be executed: System variable: control.
Remove pressure controller enable	For axes with force/pressure control only. Activate the check box if you want to disable force/pressure control. If the check box is not activated, force/pressure control is enabled.

Overview of parameters for Remove axis enable – Expert tab

Table 7-123 Overview of parameters for Remove axis enable – Expert tab

Field/Button	Meaning/Instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call.

Relevant system function for Remove axis enable

Cam technology package:

- _disableAxis

Overview of parameters for Remove axis enable /_disableAxis

Table 7-124 Parameters (MCC command Remove axis enable compared to system function _disableAxis)

Parameters of the MCC command Remove axis enable	Parameters of the system function _disableAxis
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Remove pos. controller enable	disableMode, servoControlMode
Switch enables individually according to PROFIdrive profiles	disableMode, stwbitset

Parameters of the MCC command Remove axis enable	Parameters of the system function _disableAxis
Remove drive enable	disableMode
Remove pulse enable	disableMode
Follow-up mode	servoCommandToActualMode
Remove pressure controller enable	forcecontrolMode
Expert tab	
CommandID variable	commandId
Return variable	–

Switch QF axis enable

You use this command to switch the enables on an axis with a hydraulic drive (for axes with an electric drive, you use the Switch axis enable function).

The force/pressure control can also be enabled or disabled on axes with force/pressure control.

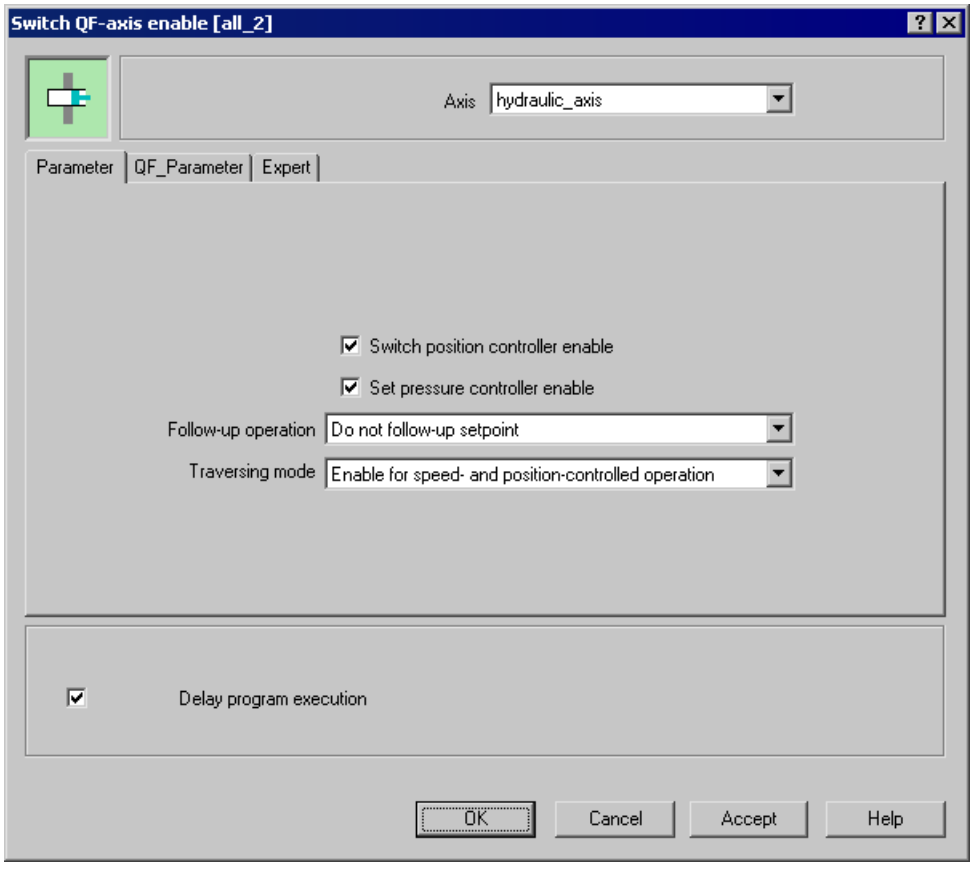


Figure 7-116 Parameter screen form: Switch QF axis enable

The following conditions must be fulfilled in order to execute motion commands on the axis:

1. If the Q-valve (actuator for volumetric flow) is assigned to multiple axes: Access to the Q-valve is ensured
2. Enable for the Q-valve is issued
3. Additional conditions for axes with force/pressure control by means of a pressure limitation valve (P-valve, actuator for force/pressure control):
   - If the pressure limitation valve is assigned to multiple axes: Access to the pressure limitation valve is ensured
   - Enable for the pressure limitation valve is issued
4. Additional conditions for position, synchronized and path axes: Position controller enable is issued
5. Follow-up mode canceled

Until all conditions are fulfilled, the axis remains in follow-up mode (see Follow-up mode for various axis technologies (Page 4253) table).

The current status of enables can be accessed via system variables, which are specified in the description of the respective parameter (see Overview of parameters for Switch QF axis enable (Page 4263)).

For additional information on **axis enables**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Switch QF axis enable

Table 7-125 Overview of parameters for Switch QF axis enable

Field/button	Meaning/instruction
Axis	In Axis, select the axis for which the axis enable is to be switched. The list contains: • All axes with a hydraulic drive that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Switch QF axis enable – Parameters tab (Page 4264)
QF_parameters tab	See Overview of parameters for Switch QF axis enable – QF_parameters tab (Page 4265)
Expert tab	See Overview of parameters for Switch QF axis enable – Expert tab (Page 4266)
Delay program execution	Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Switch QF axis enable – Parameters tab

Table 7-126 Overview of parameters for Switch QF axis enable – Parameters tab

Field/Button	Explanation/instructions
Switch position controller enable	Only for position, synchronized and path axes: Activate the check box if the position controller enable is to be switched. If the check box is not activated, the position controller enable is removed. In the case of virtual axes, the position controller enable is always set, even if this check box has not been activated. Current status of position controller enable for a real axis: System variable: servoMonitorings.controlState.
Set pressure controller enable	For axes with force/pressure control only. Activate the check box if you want to enable force/pressure control. If the check box is not activated, force/pressure control is disabled.
Follow-up mode	Here, you decide whether the axis is to be switched out of follow-up mode (see Follow-up mode for various axis technologies (Page 4253) table). **Do not follow up setpoint** (default value) Follow-up mode is disabled. Motion commands for the axis can be executed. In real axes, **Do not follow up setpoint** is only effective if all the other enables have been assigned. **Follow up setpoint** Follow-up mode is enabled. Motion commands for the axis cannot be executed. Current status for a real axis, indicating whether motion commands can be executed: System variable: control.
Traversing mode	**Maintain last setting** (default value) The axis is enabled with the most last set traversing mode (position and speed control). **Enable for speed- and position-controlled operation** The axis is enabled for speed- and position-controlled operation. This setting cannot be selected for drive axes. **Enable for speed-controlled operation** The axis is enabled for speed-controlled operation. This setting cannot be used to traverse a hydraulic axis with closed-loop velocity control using open-loop velocity-control.

Overview of parameters for Switch QF axis enable – QF_Parameters tab

Table 7-127 Overview of parameters for Switch QF axis enable – QF_Parameters tab

Field/Button	Explanation/Instructions
Access to Q-valve	If the Q-valve (actuator for volumetric flow) is assigned to multiple axes: Here, you select whether or not the axis has access to the Q-valve. **No change in access** (default value) There are no changes in access to the Q-valve **Request valve access** Access to the Q-valve is requested. Access is blocked for other axes. **Enable valve access** Access to the Q-valve is enabled. Another axis can access the Q-valve. Current status for access to the Q-valve for a real axis: System variable: actorMonitoring.qOutputState.
Set Q-valve enable	Here, you select the enables for the Q-valve. **Set enable** Enable for the Q-valve is set **Remove enable** Enable for the Q-valve is removed **No change in enables** (default value) The current enable for the Q-valve is not changed Current status of enable for the Q-valve for a real axis: Systemvariable actorMonitoring.driveState.
Increase limitation value	The rise of the manipulated variable is limited: • During a transition from or to a replacement value (requesting or enabling access to the Q-valve). • During a characteristic change Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default** System variable for default: userDefaultQFAxis.maxDerivative.QOutput
Access to pressure limitation valve	If the pressure limitation valve (P-valve, actuator for force/pressure control) is assigned to multiple axes: Here, you select whether or not the axis has access to the pressure limitation valve. **No change in access** (default value) There are no changes in access to the pressure limitation valve **Request valve access** Access to the pressure limitation valve is requested. Access is blocked for other axes. **Enable valve access** Access to the pressure limitation valve is enabled. Another axis can access the pressure limitation valve. Current status for access to the pressure limitation valve for a real axis: System variable: actorMonitoring.fOutputState.

Field/Button	Explanation/Instructions
Set pressure limitation valve enable	Here, you select the enables for the pressure limitation valve. **Set enable** Enable for the pressure limitation valve is set **Remove enable** Enable for the pressure limitation valve is removed **No change in enables** (default value) The current enable for the pressure limitation valve is not changed Current status of enable for the pressure limitation valve for a real axis: Systemvariable actorMonitoring.fOutputEnable.
Increase limitation value	The rise of the manipulated variable is limited: During a transition from or to a replacement value (requesting or enabling access to the pressure limitation valve). During a characteristic change Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default** System variable for default: userDefaultQFAxis.maxDerivative.FOutput

Overview of parameters for Switch QF axis enable – Expert tab

Table 7-128 Overview of parameters for Switch QF axis enable – Expert tab

Field/Button	Meaning/Instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call.

Relevant system function for Switch QF axis enable

Cam technology package:

- _enableQFAxis

Overview of parameters for Switch QF axis enable/_enableQFAxis

Table 7-129 Parameters (MCC command Switch QF axis enable compared to system function _enableQFAxis)

Parameters of the MCC command Switch QF axis enable	Parameters of the system function _enableQFAxis
Axis	axis
Delay program execution	nextCommand

Parameters of the MCC command Switch QF axis enable	Parameters of the system function _enableQFAxis
Parameters tab	
Switch position controller enable	controlMode
Set pressure controller enable	forcecontrolMode
Follow-up mode	commandToActualMode
Traversing mode	movingMode
QF_parameters tab	
Access to Q-valve	qoutput
Set Q-valve enable	qoutputenable
Increase limitation value	qoutputmaxderivative
Access to pressure limitation valve	foutput
Set pressure limitation valve enable	foutputenable
Increase limitation value	foutputmaxderivative
Expert tab	
CommandID variable	commandId
Return variable	–

Remove QF axis enable

You use this command to remove the enables from an axis with a hydraulic drive (for axes with an electric drive, you use the Remove axis enable function).

The force/pressure control can also be enabled or disabled on axes with force/pressure control.

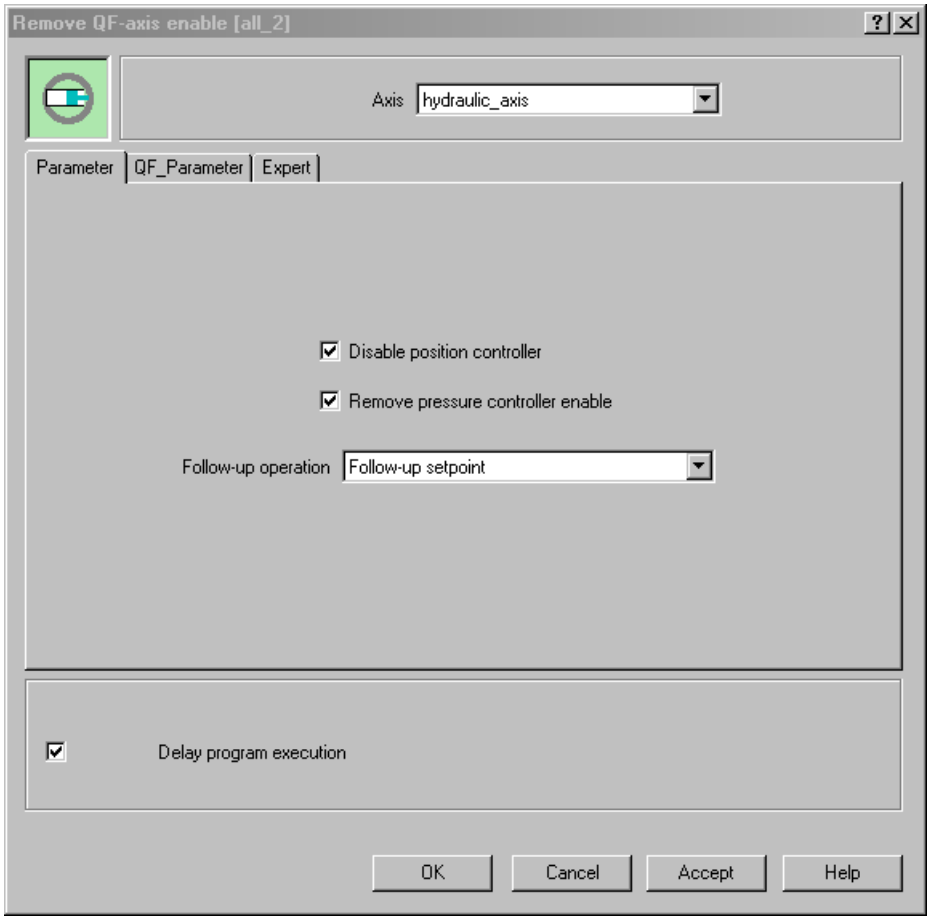


Figure 7-117 Parameter screen form: Remove QF axis enable

The following conditions must be fulfilled in order to execute motion commands on the axis:

1. If the Q-valve (actuator for volumetric flow) is assigned to multiple axes: Access to the Q-valve is ensured
2. Enable for the Q-valve is issued
3. Additional conditions for axes with force/pressure control by means of a pressure limitation valve (P-valve, actuator for force/pressure control):
   - If the pressure limitation valve is assigned to multiple axes: Access to the pressure limitation valve is ensured
   - Enable for the pressure limitation valve is issued
4. Additional conditions for position, synchronized and path axes: Position controller enable is issued
5. Follow-up mode canceled

Until all conditions are fulfilled, the axis remains in follow-up mode (see Follow-up mode for various axis technologies (Page 4253) table).

The current status of enables can be accessed via system variables, which are specified in the description of the respective parameter (see Overview of parameters for Remove QF axis enable (Page 4253)).

For additional information on **axis enables**, see the TO Axis Electric/Hydraulic Function Manual.

See also

Overview of parameters for Remove QF axis enable (Page 4269)

Overview of parameters for Remove QF axis enable

Table 7-130 Overview of parameters for Remove QF axis enable

Field/button	Meaning/instruction
Axis	In Axis, select the axis for which the axis enable is to be switched. The list contains: • All axes with a hydraulic drive that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Remove QF axis enable – Parameters tab (Page 4270)
QF_parameters tab	See Overview of parameters for Remove QF axis enable – QF_parameters tab (Page 4271)
Expert tab	See Overview of parameters for Remove QF axis enable – Expert tab (Page 4272)
Delay program execution	Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Remove QF axis enable – Parameters tab

Table 7-131 Overview of parameters for Remove QF axis enable – Parameters tab

Field/Button	Explanation/instructions
Remove pos. controller enable	Only for position, synchronized and path axes: Activate the check box if the position controller enable is to be removed. If the check box is not activated, the position controller enable is switched. In the case of virtual axes, the position controller enable is always set, even if this check box has not been activated. Current status of position controller enable for a real axis: System variable: servoMonitorings.controlState.
Remove pressure controller enable	For axes with force/pressure control only. Activate the check box if you want to disable force/pressure control. If the check box is not activated, force/pressure control is enabled.
Follow-up mode	Here, you decide whether the axis is to be switched into follow-up mode (see Follow-up mode for various axis technologies). **Follow up setpoint** (default value) Follow-up mode is enabled. Motion commands for the axis cannot be executed. **Do not follow up setpoint** Follow-up mode is disabled. Motion commands for the axis can be executed. In real axes, **Do not follow up setpoint** is only effective if all the other enables have been assigned. Current status for a real axis, indicating whether motion commands can be executed: System variable: control.

Overview of parameters for Remove QF axis enable – QF_Parameters tab

Table 7-132 Overview of parameters for Remove QF axis enable – QF_Parameters tab

Field/Button	Explanation/Instructions
Access to Q-valve	If the Q-valve (actuator for volumetric flow) is assigned to multiple axes: Here, you select whether or not the axis enables access to the Q-valve. **No change in access** (default value) There are no changes in access to the Q-valve **Enable valve access** (default value) Access to the Q-valve is enabled. Another axis can access the Q-valve. Current status for access to the Q-valve for a real axis: Systemvariable actorMonitoring.qOutputState
Set Q-valve enable	Here, you select whether the enables for the Q-valve are removed. **Remove enable** Enable for the Q-valve is removed **No change in enables** (default value) The current enable for the Q-valve is not changed Current status of enable for the Q-valve for a real axis: Systemvariable actorMonitoring.driveState.
Replacement value for the Q-valve is set	Here, you select whether a replacement value is to be set for the Q-valve. The replacement value is applied if no axis has access to the Q-valve. **Set replacement value** You enter the replacement value in the subsequent input field. **No change in the replacement value** (default value) Replacement value is not changed.
Replacement value	Here, you enter the replacement value for the Q-valve.
Increase limitation value	The rise of the manipulated variable is limited: • During a transition from or to a replacement value (requesting or enabling access to the Q-valve). • During a characteristic change Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). Default System variable for default: userDefaultQFAxis.maxDerivative.QOutput
Access to pressure limitation valve	If the pressure limitation valve (P-valve, actuator for force/pressure control) is assigned to multiple axes: Here, you select whether or not the axis enables access to the pressure limitation valve. **No change in access** (default value) There are no changes in access to the pressure limitation valve **Enable valve access** Access to the pressure limitation valve is enabled. Another axis can access the pressure limitation valve. Current status for access to the pressure limitation valve for a real axis: System variable: actorMonitoring.fOutputState.

Field/Button	Explanation/Instructions
Set pressure limitation valve enable	Here, you select whether the enables for the pressure limitation valve are removed. **Remove enable** Enable for the pressure limitation valve is removed **No change in enables** (default value) The current enable for the pressure limitation valve is not changed Current status of enable for the pressure limitation valve for a real axis: Systemvariable actorMonitoring.fOutputEnable.
Replacement value for pressure limitation valve enable	Here, you select whether a replacement value is to be set for the pressure limitation valve. The replacement value is applied if no axis has access to the pressure limitation valve. **Set replacement value** You enter the replacement value in the subsequent input field. **No change in the replacement value** (default value) Replacement value is not changed
Replacement value	Here, you enter the replacement value for the pressure limitation valve.
Increase limitation value	The rise of the manipulated variable is limited: During a transition from or to a replacement value (requesting or enabling access to the pressure limitation valve). During a characteristic change Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default** System variable for default: userDefaultQFAxis.maxDerivative.FOutput

Overview of parameters for Remove QF axis enable – Expert tab

Table 7-133 Overview of parameters for Remove QF axis enable – Expert tab

Field/Button	Meaning/Instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call.

Relevant system function for Remove QF axis enable

Cam technology package:

- _disableQFAxis

Overview of parameters for Remove QF axis enable / _disableQFAxis

Table 7-134 Parameters (MCC command Remove QF axis enable compared to system function _disableQFAxis)

Parameters of the MCC command Remove QF axis enable	Parameters of the system function _disableQFAxis
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Remove pos. controller enable	controlMode
Remove pressure controller enable	forcecontrolMode
Follow-up mode	commandToActualMode
QF-Parameters tab	
Access to Q-valve	qoutput
Set Q-valve enable	qoutputenable
Substitute value for the Q-valve is set	qoutputvaluesetmode
Substitute value	qoutputvalue
Increase limitation value	qoutputmaxderivativetype, qoutputmaxderivative
Access to pressure limitation valve	foutput
Set pressure limitation valve enable	foutputenable
Substitute value for pressure control valve enable	foutputvaluesetmode
Substitute value	foutputvalue
Increase limitation value	foutputmaxderivativetype, foutputmaxderivative
Expert tab	
CommandID variable	commandId
Return variable	–

Speed preset

This command starts an axis in speed-controlled mode.

The axis is accelerated or decelerated to the programmed rotational speed (velocity). Once this speed is reached, it is held constant.

When the constant motion phase is limited (**Constant traversing time** parameter in the Dynamic response tab), the axis is decelerated to the set speed of 0 once the specified time has elapsed.

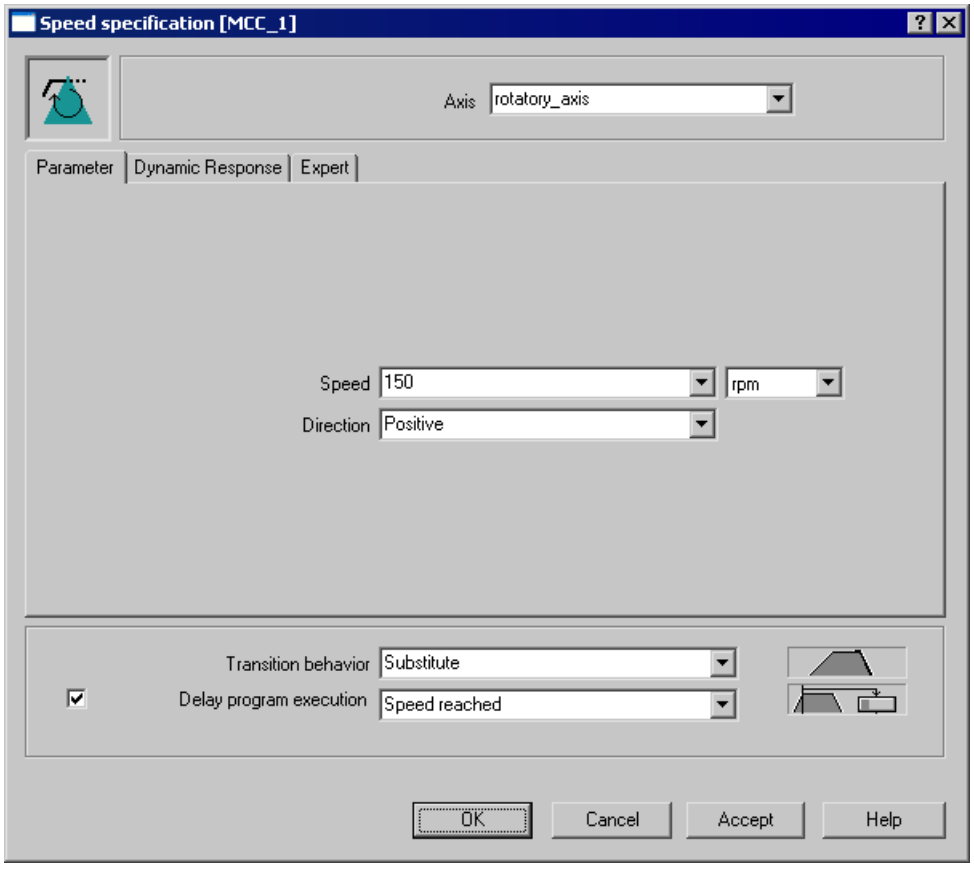


Figure 7-118 Parameter screen form: Speed specification

For additional information on **moving axes**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for speed specification

Table 7-135 Overview of parameters for speed specification

Field/button	Meaning/instruction
Axis	In Axis, select which axis the command is being programmed for. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Speed specification - Parameters tab (Page 4275)
Dynamic response tab	See Overview of parameters for Speed specification - Dynamic response tab (Page 4276)
Expert tab	See Overview of parameters for Speed specification - Expert tab (Page 4276)

Field/button	Meaning/instruction
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036)
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Speed specification – Parameters tab

Table 7-136 Overview of parameters for Speed specification – Parameters tab

Field/Button	Explanation/Instructions
speed	Speed value during the constant speed phase. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Resulting velocity** An existing acceleration or deceleration motion of the axis is ended, taking into account the jerk parameters in the Dynamic response tab. The axis continues at the resulting velocity. The programmed transition behavior must be "Substitute". **Current** **last speed programmed** **Default** See also Selection list (combo box) (Page 4022) System variable for default: userDefaultDynamics.velocity See also the general description of the Dynamic response tab (Page 4031).
Direction	Select the direction of rotation. **From speed sign** Direction is obtained from speed sign. **Positive** Positive direction **Negative** Negative direction **last direction programmed** **Default** See also Selection list (combo box) (Page 4022) System variable for default: userDefaultDynamics.direction

Overview of parameters for Speed specification – Dynamic response tab

Table 7-137 Overview of parameters for Speed specification – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Overview of parameters - Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk
Constant traversing time	A time limit can be programmed for the command. To do so, activate the check box and enter the time duration of the constant motion phase.

Overview of parameters for Speed specification – Expert tab

Table 7-138 Overview of parameters for Speed specification – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for speed specification

Cam technology package:

- _move

Overview of parameters for speed specification / _move

Table 7-139 Parameters (MCC command Speed specification compared system function to _move)

Parameters of the MCC command Speed specification	Parameters of the system function _move
	movingMode = SPEED_CONTROLLED
Axis	axis
Transition behavior	mergemode
Delay program execution	nextCommand
Parameters tab	
Speed	velocitytype, velocity
Direction	direction
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Constant traversing time	movetimeouttype, movetimeout
Expert tab	
CommandID variable	commandId
Return variable	–

Start axis position-controlled

The command starts a position or synchronized axis in position-controlled mode.

The axis is accelerated or decelerated to the programmed velocity. Once this velocity is reached, it is held constant.

When the constant motion phase is limited (**Constant traversing time** parameter in the Dynamic response tab), the axis is decelerated to the set velocity of 0 once the specified time has elapsed.

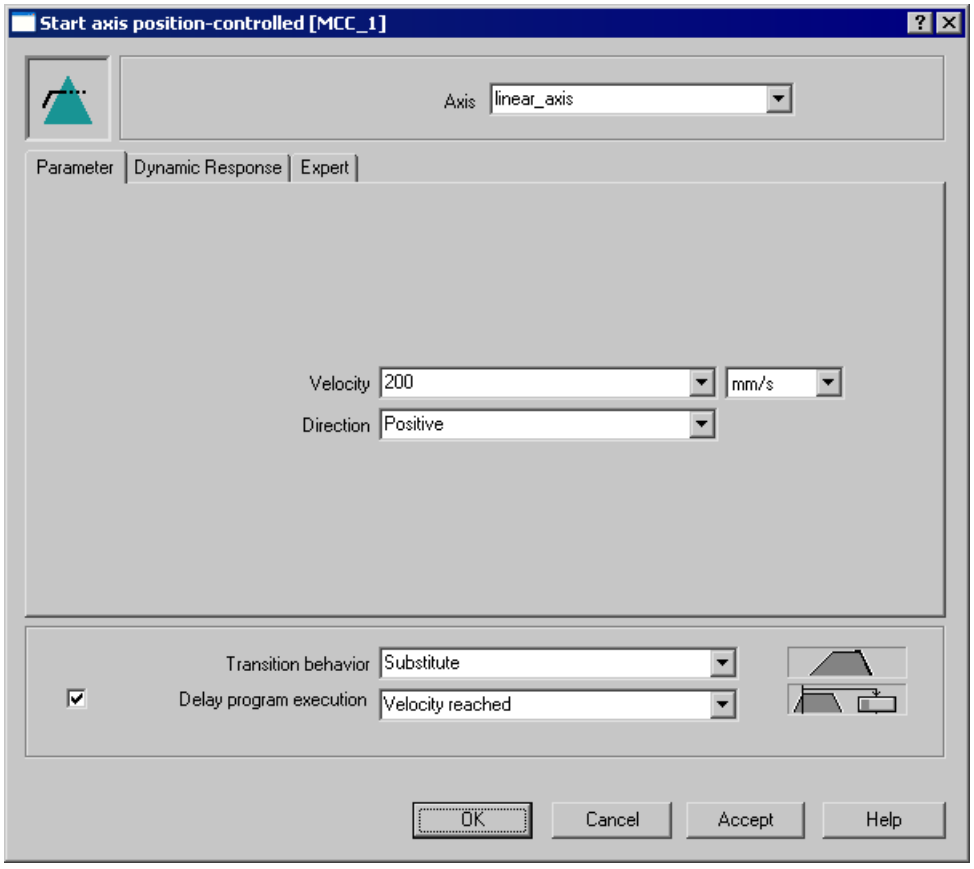


Figure 7-119 Parameter screen form: Start axis position-controlled

For additional information on **moving axes**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Start axis position-controlled

Table 7-140 Overview of parameters for Start axis position-controlled

Field/button	Meaning/instruction
Axis	In Axis, select the axis for which the command is to be programmed. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Parameters tab	See Overview of parameters for Start axis position-controlled – Parameters tab (Page 4279)
Dynamic response tab	See Overview of parameters for Start axis position-controlled – Dynamic response tab (Page 4280)
Expert tab	See Overview of parameters for Start axis position-controlled – Expert tab (Page 4280)

Field/button	Meaning/instruction
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Start axis position-controlled – Parameters tab

Table 7-141 Overview of parameters for Start axis position-controlled – Parameters tab

Field/Button	Explanation/Instructions
Velocity	Velocity value during the constant velocity phase. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Resulting velocity** An existing acceleration or deceleration motion of the axis is ended, taking into account the jerk parameters in the Dynamic response tab. The axis continues at the resulting velocity. **Current** **Last programmed velocity** **Default value** System variable for default: userDefaultDynamics.velocity
Direction	Select the direction of motion here. **From velocity sign** Direction is obtained from the sign of the velocity. **Negative** Negative direction **Positive** Positive direction **Last programmed direction** **Default value** System variable for default: userDefaultDynamics.direction

Overview of parameters for Start axis position-controlled – Dynamic response tab

Table 7-142 Overview of parameters for Start axis position-controlled – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk
Constant traversing time	A time limit can be programmed for the command. To do so, activate the check box and enter the time duration of the constant motion phase.

Overview of parameters for Start axis position-controlled – Expert tab

Table 7-143 Overview of parameters for Start axis position-controlled – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call.

Relevant system function for Start axis position-controlled

Cam technology package:

- _move

Overview of parameters for Start axis position-controlled / _move

Table 7-144 Parameters (MCC command Start axis position-controlled compared to system function _move)

Parameters of the MCC command Start axis position-controlled	Parameters of the system function _move
	movingMode = POSITION_CONTROLLED
Axis	axis
Transition behavior	mergemode
Delay program execution	nextCommand
Parameters tab	
Velocity	velocitytype, velocity
Direction	direction
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Constant traversing time	movetimeouttype, movetimeout
Expert tab	
CommandID variable	commandId
Return variable	–

Stop axis

This command stops the motion of an axis and is effective for all single axis motions. The motion can be stopped with Normal stop or Quick stop.

- Normal stop: This command acts on all single axis motions (positioning and speed motions), but not on synchronous motions.
- Quick stop: This command also acts on synchronous motions. In addition, the axis is disabled with respect to further motion commands; this state can be canceled with the commands "Remove axis enable" (Page 4258)or "Reset object" (Page 4193).

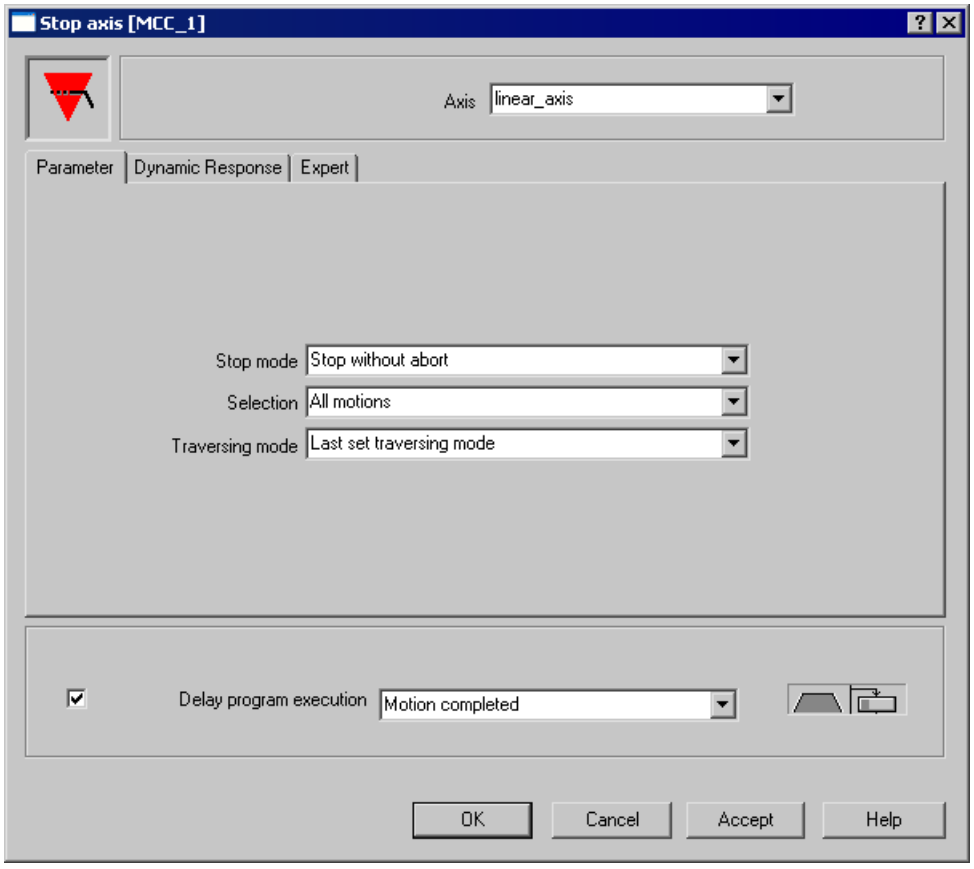


Figure 7-120 Parameter screen form: Stop axis

For additional information on the **quick stop** function, see the TO Axis Electric/Hydraulic Function Manual.

For additional information on the **normal stop** function, see the TO Axis Electric/Hydraulic Function Manual.

For additional information on the **traversing mode**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Stop axis

Table 7-145 Overview of parameters for Stop axis

Field/button	Meaning/instruction
Axis	In Axis, select which axis is to be stopped. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Stop axis – Parameters tab (Page 4284)
Dynamic response tab	See Overview of parameters for Stop axis – Dynamic response tab (Page 4285)

Field/button	Meaning/instruction
Expert tab	See Overview of parameters for Stop axis – Expert tab (Page 4286)
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Stop axis – Parameters tab

Table 7-146 Overview of parameters for Stop axis – Parameters tab

Field/button	Meaning/instruction
Stop mode	Select a stop mode here. Only those motions that have been stopped in "normal stop without abort" mode can be continued with the "Continue motion" (Page 4287) command. No other motions can be continued. In the case of a quick stop, the axis is disabled with respect to further motion commands; this state can be cancelled with the commands "Remove axis enable" (Page 4258) or "Reset object" (Page 4193). **Normal stop without abort** (default value) The motion specified in the **Selection** field is stopped with the programmed dynamic response parameters (**Dynamic response** tab – see Overview of parameters for Stop axis - Dynamic response tab (Page 4285)). It can be continued with the "Continue motion" (Page 4287) command. No other commands for the axis may be programmed between the stop command and the continue command. The command does not act on synchronous motions. **Normal stop with abort** The motion specified in the **Selection** field is stopped with the programmed dynamic response parameters (**Dynamic response** tab – see Overview of parameters for Stop axis - Dynamic response tab (Page 4285)). The motion cannot be continued. The command does not act on synchronous motions. **Quick stop at maximum deceleration** The motion is stopped according to interpolation using the maximum dynamic values on the axis. It cannot be continued. **Quick stop within defined period** The motion can be brought to a standstill in the programmed time. You program the time in the **Time for deceleration** parameter in the **Dynamic response** tab, see Overview of parameters for Stop axis - Dynamic response tab (Page 4285). The motion cannot be continued. **Stopping with preassigned braking ramp** The motion is stopped via the preconfigured braking ramp. This is set during configuration (configuration data emergencyRampGenerator.maxDeceleration). The motion cannot be continued. **Quick stop with dynamic response parameters** The motion is stopped with the programmed dynamic response parameters (**Dynamic response** tab – see Overview of parameters for Stop axis - Dynamic response tab (Page 4285)). The motion cannot be continued.
Selection	Select whether you want to stop the motion as a whole, only the basic motion, or only the superimposed motion. This parameter can only be selected for the following stop modes: • Normal stop without abort (default value) • Normal stop with abort **All motions** (default value) All motions of the programmed axis are stopped. **Basic motion** The basic motion is stopped. **Superimposed motion** The superimposed motion is stopped.

Field/button	Meaning/instruction
Traversing mode	**Position-controlled** For positioning and synchronous axes only: Axis is switched from the current traversing mode (for example, speed control, force control, or torque control) to position control, and stopped. **Speed controlled** Axis is switched from the current traversing mode (for example, position control, force control, or torque control) to speed control, and stopped. The speed ramp takes effect immediately; an existing following error does not have to be removed first. If position-controlled motions are stopped in speed-controlled mode, the axis is disabled with respect to further motion commands; this state can be cancelled with the commands **Remove axis enable** or **Reset object**. **Last set traversing mode** (default value) Axis is switched from the current traversing mode (for example, position control, speed control, force control, or torque control) to the last set traversing mode (position or speed control), and stopped.

Overview of parameters for Stop axis – Dynamic response tab

Table 7-147 Overview of parameters for Stop axis – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile
Acceleration	The meaning depends on the acceleration model used, which can be selected via the configuration data TypeOfAxis.DecodingConfig.directionDynamic: • **Non-direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = NO (default value) The value has no effect. • **Direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = YES The value causes deceleration when the direction of rotation is negative and has no effect when the direction of rotation is positive. The entered value acts during the constant deceleration phase. For additional information on specifying the acceleration and deceleration (acceleration model), see the TO Axis Electric/Hydraulic Function Manual. System variable for default: userDefaultDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The meaning depends on the acceleration model used, which can be selected via the configuration data TypeOfAxis.DecodingConfig.directionDynamic: • **Non-direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = NO (default) The value causes deceleration in the motion of the axis, independent of the direction of motion. • **Direction-based acceleration model** TypeOfAxis.DecodingConfig.directionDynamic = YES The value causes deceleration when the direction of rotation is positive and has no effect when the direction of rotation is negative. The entered value acts during the constant deceleration phase. For additional information on specifying the acceleration and deceleration (acceleration model), see the TO Axis Electric/Hydraulic Function Manual. System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk
Time for deceleration	The entered value determines the duration of the braking operation in the **Quick stop within defined period** stop mode. System variable for default: userDefaultDynamics.stopTime

See also

Dynamics tab (Page 4031)

Overview of parameters for Stop axis – Expert tab

Table 7-148 Overview of parameters for Stop axis – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Stop axis

Cam technology package:

- _stop (for Normal stop without/with abort stop mode)
- _stopEmergency (for Quick stop with ... stop mode)

Overview of parameters for Stop axis, _stop, _stopEmergency

Table 7-149 Parameters (MCC command Stop axis compared to system functions _stop, _stopEmergency)

Parameters of the MCC command Stop axis	Parameters of the system functions _stop, _stopEmergency
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Stop mode	stopMode (system function _stop, Normal stop ...) stopDriveMode (system function _stopEmergency, Quick stop with ...)
Selection	stopSpecification (system function _stop)
Traversing mode	movingMode
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Time for deceleration	stopTimeType, stopTime (system function _stopEmergency)
Expert tab	
CommandID variable	commandId
Return variable	–

Continue motion

This command continues a motion that has been stopped. The following motions can be continued:

- All speed-controlled motions that were brought to a standstill using the **Normal stop without abort** stop mode,
- all position-controlled motions that were brought to a standstill using the **Normal stop without abort** stop mode and not in speed-controlled mode.

The axis must not receive any new motion commands between interruption and continuation of the interrupted motion.

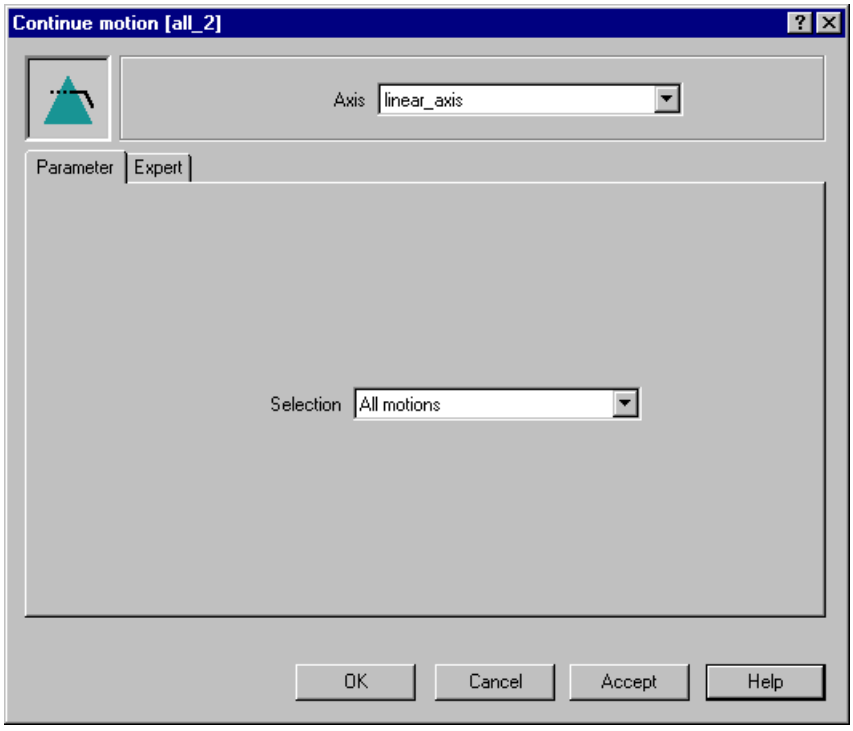


Figure 7-121 Parameter screen form: Continue motion

For additional information on **continuing motions**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Continue motion

Table 7-150 Overview of parameters for Continue motion

Field/button	Meaning/instruction
Axis	In Axis, select which axis for which motion is to be continued. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Continue motion - Parameters tab (Page 4289)
Expert tab	See Overview of parameters for Continue motion - Expert tab (Page 4289)

Overview of parameters for Continue motion - Parameters tab

Table 7-151 Overview of parameters for Continue motion - Parameters tab

Field/Button	Explanation/instructions
Selection	Here, select which axis motion component is to be continued. **All motions** (default value) All motions of the programmed axis are continued. **Basic motion** The basic motion is continued. **Superimposed motion** The superimposed motion is continued.

Overview of parameters for Continue motion - Expert tab

Table 7-152 Overview of parameters for Continue motion - Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Continue motion

Cam technology package:

- _continue

Overview of parameters for Continue motion, _continue

Table 7-153 Parameters (MCC command Continue motion compared to system function _continue)

Parameters of the MCC command Continue motion	Parameters of the system function _continue
Axis	axis
Parameters tab	
Selection	continuespecification
Expert tab	
Return variable	–

Home axis

With a position axis or synchronized axis, the position values that are displayed or entered refer to the coordinate system of the axis. The coordinate system of the axis must be aligned to the real physical position of the axis.

If you are using an absolute measuring system, the alignment need only be done once during commissioning. When commissioning is completed, the position value will be known when the machine is switched on.

With an incremental measuring system, the alignment must be repeated each time the machine is switched on. The alignment is achieved via homing.

Note

Traversing commands with relative position specification can always be executed.

The axis can be configured to indicate whether traversing commands with absolute position specification can also be executed on a non-homed axis. Configuration data: TypeOfAxis.homing.referencingNecessary.

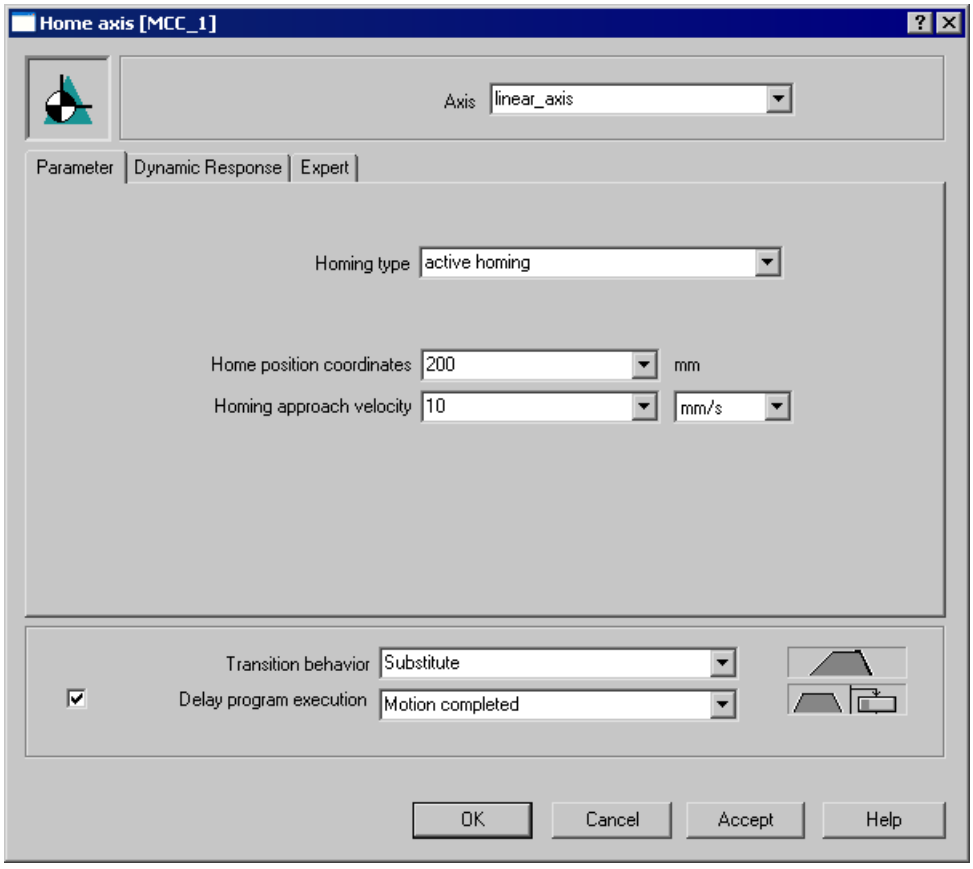


Figure 7-122 Parameter screen form: Home axis

For additional information on **homing**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Home axis

Table 7-154 Overview of parameters for Home axis

Field/button	Meaning/instruction
Axis	In Axis, select which axis is to be homed. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Parameters tab	See Overview of parameters for Home axis – Parameters tab (Page 4292)
Dynamic response tab	See Overview of parameters for Home axis – Dynamic response tab (Page 4294)
Expert tab	See Overview of parameters for Home axis – Expert tab (Page 4294)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036)
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Home axis – Parameters tab

Table 7-155 Overview of parameters for Home axis – Parameters tab

Field/button	Meaning/instruction
Homing type	Select among the permitted homing types: **Active homing** (default value) Homing occurs by means of a separate traversing motion of the axis. The homing mode is specified when you configure the axis during the commissioning phase. **Passive homing** The homing command itself does not trigger any active traversing motion of the axis. Rather, homing occurs during the next axis motion. The homing mode is specified when you configure the axis during the commissioning phase. Required settings: 1. For **Home axis** command: Deactivate the **Delay program execution** checkbox. 2. For the subsequent command: Select **Attach** as the **Transition behavior** from the active command. **Set home position** The value of the home position coordinate is assigned to the current position (actual value) of the axis. There is no active traversing motion. **Relative home position setting** The value of the home position coordinate is added to the current position (actual value) of the axis. There is no active traversing motion. **Absolute encoder adjustment** Absolute encoder adjustment is only possible if the axis was configured as an absolute encoder or a cyclic absolute encoder during commissioning. The offset between the axis zero point and the encoder zero point is determined by selecting a value for the absolute encoder offset. There is no active traversing motion. Generally speaking, the adjustment only needs to be repeated in exceptional circumstances (see Motion Control TO Axis Electric/Hydraulic, External Encoder Function Manual). **Absolute encoder adjustment with specification of the position value** Absolute encoder adjustment with specification of the position value is only possible if the axis was configured as an absolute encoder or a cyclic absolute encoder during commissioning. The value of the home position coordinate is assigned to the current position (actual value) of the axis, and the offset (absolute encoder offset) between the axis zero point and the encoder zero point is calculated from this. There is no active traversing motion. Generally speaking, the adjustment only needs to be repeated in exceptional circumstances (see Motion Control TO Axis Electric/Hydraulic, External Encoder Function Manual).
Home position coordinates	Not when homing type = absolute encoder adjustment. Coordinates of the home position in the reference system of the axis. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefaultHoming.homingPosition

Field/button	Meaning/instruction
Homing approach velocity	Only when homing type = Active homing. Velocity at which the axis approaches the homing output cam (only with homing mode **With homing output cam and encoder zero mark** or **Encoder zero mark only**) Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Current** **last programmed** **default value** See Selection list (combo box) (Page 4022). System variable for default: userDefaultHoming.homingApproachVelocity

Note

Note that when the axis to be homed is defined on the device and configured with several axis data sets and different encoder types (e.g. absolute encoder or incremental encoder):

Only the homing types permitted for one encoder type are offered in the Parameters tab. This encoder type can be identified in the "Expert" (Page 4034) tab at **TO properties**. The configuration data or system variables that affect the parameter screen form are displayed there. These settings cannot be changed. See also Section "Using a technology object (Page 4024)".

Remedy (three options):

- Use the "System function call (Page 4161)" MCC command and configure the **_homing** system function.
  For the assignment of the parameters of the **_homing** system function to the parameters of the **Home axis** MCC command, see Section "Relevant system functions for Home axis (Page 4294)".
- Use the "ST zoom (Page 4188)" MCC command and program the **_homing** system function in the SIMOTION ST (Structured Text) programming language.
- Instead of using an axis defined on the device, use a variable of the appropriate data type, see Section "Data types of the technology objects (Page 4060)".
  In the "Expert (Page 4034)" tab, the configuration data or system variables that affect the parameter screen form are displayed at **TO properties**. Since the assignment of the variables to a technology object is not performed until the program runtime, the values of the configuration data or system variables can be freely selected and the parameter screen form adapted accordingly. See also Section "Using a variable of the technology object data type (Page 4024)".
  In this case, make sure the technology object is assigned correctly to the variables.

Overview of parameters for Home axis – Dynamic response tab

Table 7-156 Overview of parameters for Home axis – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters in the Dynamic response tab are evaluated only with homing type Active homing.
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk

Overview of parameters for Home axis – Expert tab

Table 7-157 Overview of parameters for Home axis – Expert tab

Field/Button	Meaning/Instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Home axis

Cam technology package:

- _homing

Overview of parameters for Home axis / _homing

Table 7-158 Parameters (MCC command Home axis compared to system function _homing)

Parameters of the MCC command Home axis	Parameters of the system function _homing
Axis	axis
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Homing type	homingmode
Home position coordinates	homepositiontype, homeposition
Reference point approach velocity	velocitytype, velocity
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Referencing mode

Various homing modes are possible for Active homing and Passive homing homing types. See table below.

You specify the homing modes separately for each encoder system when configuring the axis.

Table 7-159 Homing mode for active and passive homing

Referencing mode	Description	
	Active homing	**Passive homing**
	Homing occurs by means of a separate traversing motion of the axis. Configuration data for homing mode: TypeOfAxis.NumberOfEncoders. Encoder_x.IncHomingEncoder. homingMode.	The homing command itself does not trigger any active traversing motion of the axis. Rather, homing occurs during the next axis motion. Configuration data for homing mode: TypeOfAxis.NumberOfEncoders. Encoder_x.IncHomingEncoder. passiveHomingMode.
With homing output cam and encoder zero mark	The homing command initiates axis motion towards the homing output cam. After the homing output cam is crossed, the axis moves to the next zero mark of the measuring system. Once the first zero mark is detected, the measuring system is synchronized. The axis is then moved by the amount of the home position offset. The axis is then at the home position. The position value is set to the value indicated in the home position coordinates.	After the homing output cam is detected, the next zero mark of the measuring system is active for synchronization. Once the first zero mark is detected after the homing output cam, synchronization takes place. The position value of the axis is set to the value indicated in the home position coordinates.
External zero mark only	The homing command initiates axis motion towards the external zero mark (e.g., homing output cam). Once the external zero mark is crossed, the measuring system is synchronized to the edge of the external zero mark. The axis is then moved by the amount of the home position offset. The axis is then at the home position. The position value is set to the value indicated in the home position coordinates.	Once the external zero mark (e.g. homing output cam) has been detected, synchronization with the edge takes place. The position value of the axis is set to the value indicated in the home position coordinates.
Encoder zero mark only	The homing command initiates an axis motion towards the zero mark of the encoder. When the zero mark is crossed, the measuring system of the axis is synchronized to this zero mark. The axis is then moved by the amount of the home position offset. The axis is then at the home position. The position value is set to the value indicated in the home position coordinates.	Once the zero mark is detected, synchronization occurs. The position value of the axis is set to the value indicated in the home position coordinates.

Home position offset

If you do not wish the axis to be positioned on the synchronization point after homing, but traversed further by a defined distance, then you must enter a home position offset.

The home position is then calculated from the position of the zero mark and the amount of the home position offset.

Position axis

The command traverses the defined axis via a parameterizable velocity profile to the programmed target position. The position can be specified as an absolute or relative position.

The programmed position must lie within the software limit switches.

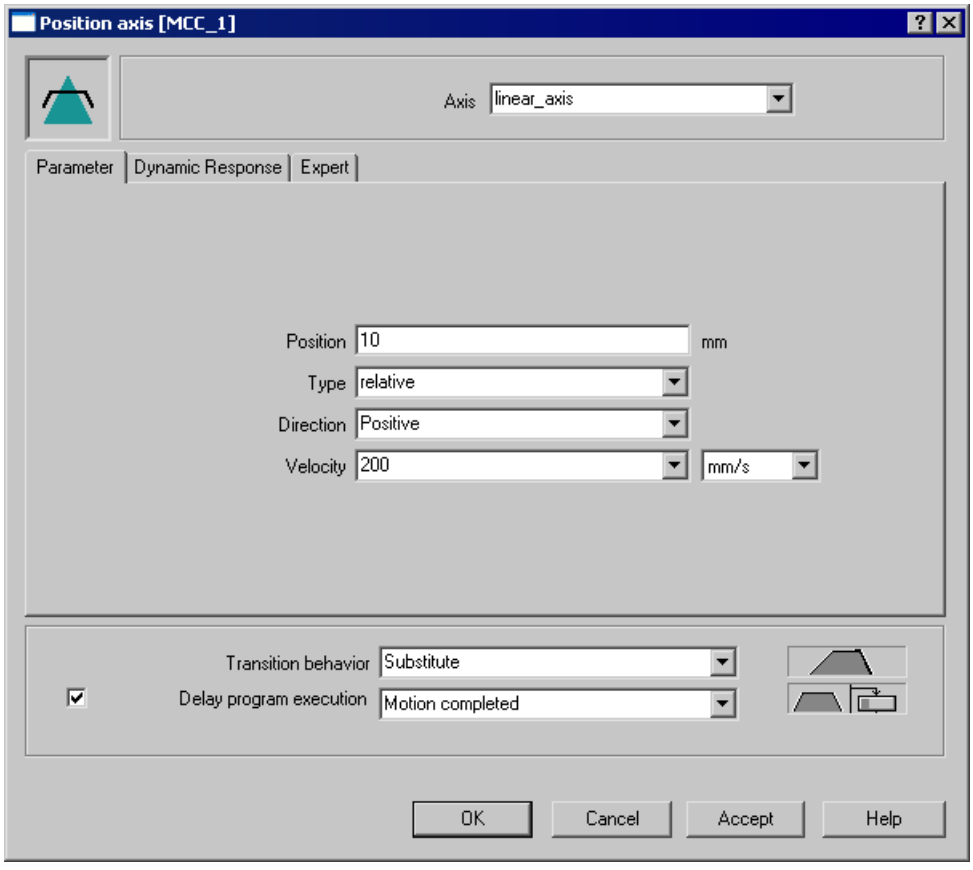


Figure 7-123 Parameter screen form: Position axis

For additional information on **position axes**, see the TO Axis Electric/Hydraulic Function Manual.

Parameter screen form: Position axis

Table 7-160 Parameter screen form: Position axis

Field/button	Meaning/instruction
Axis	In Axis, select which axis is to travel to a target position. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Parameters tab	See Overview of parameters for Position axis – Parameters tab (Page 4298)
Dynamic response tab	See Overview of parameters for Position axis – Dynamic response tab (Page 4299)
Expert tab	See Overview of parameters for Position axis – Expert tab (Page 4300)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Position axis – Parameters tab

Table 7-161 Overview of parameters for Position axis – Parameters tab

Field/button	Meaning/instruction
Position	The meaning depends on the following **type** parameter: • "Absolute" type: End position of the motion • "Relative" type: Distance traversed during the motion, starting from the current axis position. Enter the value as a signed floating-point number.
Type	Here, you specify the meaning of the programmed position in more detail (see above). **Absolute** (default value) The programmed position value is the end position of the motion. **Relative** The value entered in Position generates the traversing distance of the motion, starting from the current axis position.

Field/button	Meaning/instruction
Direction	The direction of motion must be specified in the following cases: • Type of motion (see above) is relative. • Type of motion (see above) is absolute **and** the axis is a modulo axis. If a positive or negative direction is programmed, the direction has a higher priority than the position. The position is approached as an absolute value, a negative absolute value, or as a function of sign. The velocity sign is determined by the specified direction. Select the direction of motion. **From position** The direction of motion is determined by the position sign. **Positive** The direction of motion for the command is in the positive axis direction. With relative motion, the sign of the position is ignored. **Negative** The direction of motion for the command is in the negative axis direction. With relative motion, the sign of the position is ignored. **Shortest path** • For type absolute and modulo axis: The direction of motion for the current command is the direction in which the programmed target position can be reached via the shortest path. • For type relative: The direction of motion is determined by the position sign. **Last direction set in the program** See Selection list (combo box) (Page 4022)
Velocity	Velocity value during the constant velocity phase. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Current** **Last programmed velocity** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefaultDynamics.velocity See also Dynamic response tab (Page 4031).

Overview of parameters for Position axis – Dynamic response tab

Table 7-162 Overview of parameters for Position axis – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel

Field/Button	Explanation/Instructions
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk

Overview of parameters for Position axis – Expert tab

Table 7-163 Overview of parameters for Position axis – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter dialog box as needed to reflect the effects of axis configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Example of positioning

An axis must travel at a velocity of 100 mm/s to the position at 1,000 mm. The position controller enable must be set for the axis before it can be positioned. The pulse and drive enables must also be set.

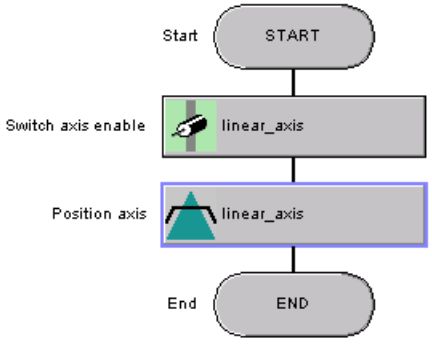


Figure 7-124 Example of positioning

Relevant system functions for Position axis

Cam technology package:

- _pos

Overview of parameters for Position axis / _pos

Table 7-164 Parameters (MCC command Position axis compared to system function _pos)

Parameters of the MCC command Position axis	Parameters of the system function _pos
Axis	axis
Transition behavior	mergemode
Delay program execution	nextCommand
Parameters tab	
Position	position
Type	positioningMode
Direction	direction
Velocity	velocitytype, velocity
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Travel to fixed endstop

This command activates

- Monitoring of "Travel to fixed endstop" in parallel with an axis motion activated by a motion command
- Retention of a clamping torque after reaching the fixed end stop

This process is also referred to as "clamping". The method for detecting that the end stop has been reached (evaluation of the following error or axis torque) is specified during configuration.

This command can also be used to switch over the clamping torque during active clamping.

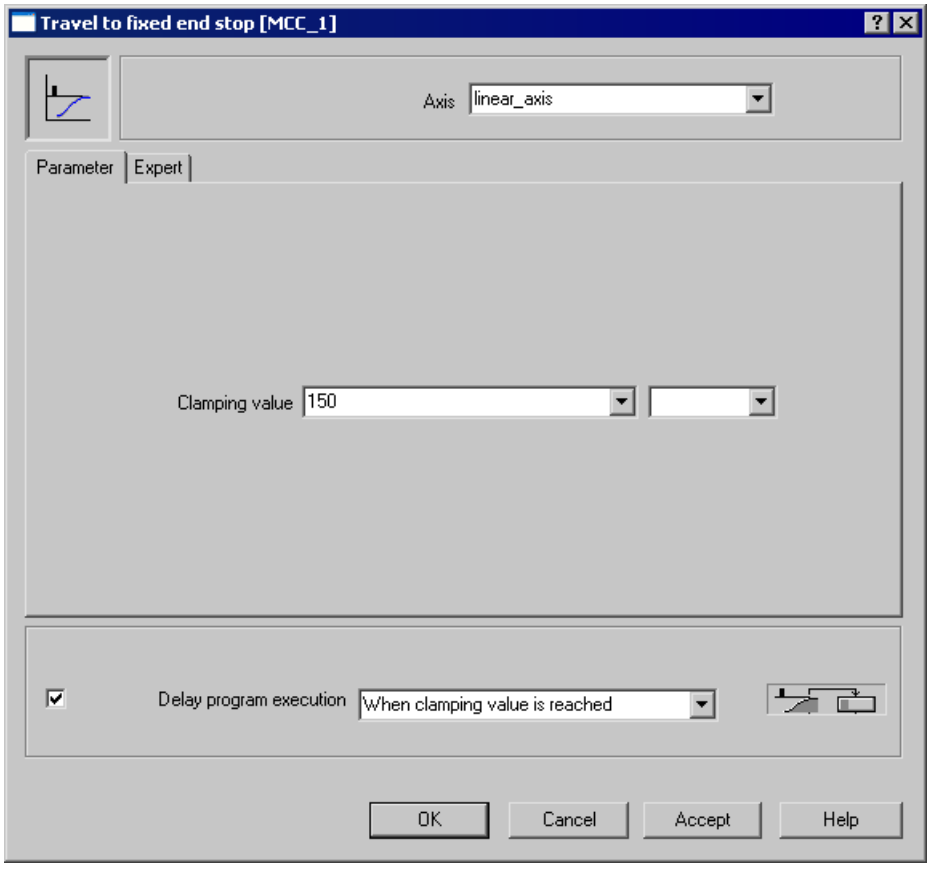


Figure 7-125 Parameter screen form: Travel to fixed end stop

For additional information on **traveling to fixed end stop**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Travel to fixed endstop

Table 7-165 Overview of parameters for Travel to fixed endstop

Field/button	Meaning/instruction
Axis	In Axis, select which axis the command is being programmed for. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Parameters tab	See Overview of parameters for Travel to fixed end stop – Parameters tab (Page 4303)

Field/button	Meaning/instruction
Expert tab	See Overview of parameters for Travel to fixed end stop – Expert tab (Page 4303)
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **If command has been completed or aborted** The next command is executed after the current command has been completed or aborted. **If clamping value is reached** The next command is executed as soon as the clamping value is reached. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Travel to fixed endstop – Parameters tab

Table 7-166 Overview of parameters for Travel to fixed endstop – Parameters tab

Field/Button	Explanation/Instructions
Clamping value	Value to which the torque is limited. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Last programmed torque** **Default** See also Selection list (combo box) (Page 4022). System variable for default: userDefaultClamping.ClampingValue.

Overview of parameters for Travel to fixed endstop – Expert tab

Table 7-167 Overview of parameters for Travel to fixed endstop – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Travel to fixed endstop

Cam technology package:

- _enableMovingToEndStop

Overview of parameters for Travel to fixed endstop / _enableMovingToEndStop

Table 7-168 Parameters (MCC command Travel to fixed endstop compared to system function _enableMovingToEndStop)

Parameters of the MCC command Travel to fixed endstop	Parameters of the system function _enableMovingToEndStop
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Clamping value	clampingvaluetype, clampingvalue, torquelimitunit
Expert tab	
CommandID variable	commandId
Return variable	–

Remove fixed endstop

This command is used to deactivate monitoring of "Travel to fixed end stop" in parallel with an axis motion activated by a motion command and to discontinue retention of a clamping torque after reaching the fixed end stop.

The axis is placed in position-controlled operation.

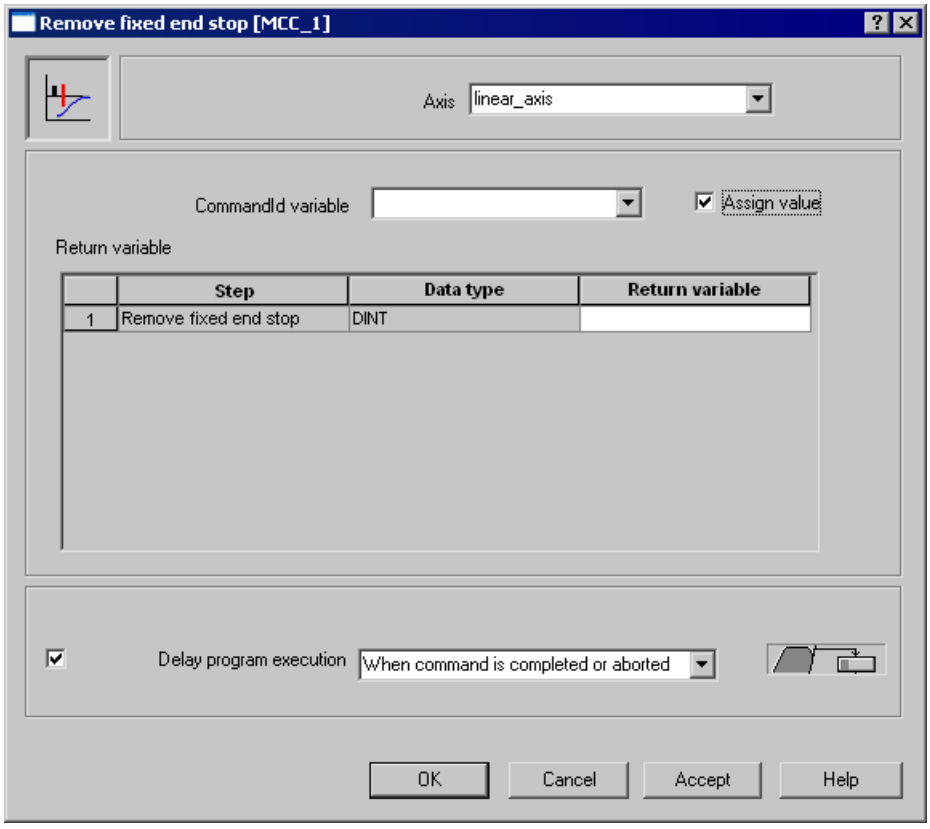


Figure 7-126 Parameter screen form: Remove fixed end stop

For additional information on **traveling to fixed end stop**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Remove fixed endstop

Table 7-169 Overview of parameters for Remove fixed endstop

Field/button	Meaning/instruction
Axis	In Axis, select which axis the command is being programmed for. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
CommandID variable	Enter the name of a variable of data type CommandIdType here. With the help of this variable, you can, for example, trace back the status of the command. Variables of data type CommandIdType declared in the MCC unit or MCC chart are provided for selection. If you leave the CommandID variable field empty, the CommandID is not assigned to any variables; this means that you cannot access the CommandID (standard). See also Overview of parameters for the Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **If command has been completed or aborted** The next command is executed only after the current command has been completed or aborted. See also Delay program execution (step enabling condition) (Page 4037).

Relevant system function for Remove fixed endstop

Cam technology package:

- _disableMovingToEndStop

Overview of parameters for Remove fixed endstop / _disableMovingToEndStop

Table 7-170 Parameters (MCC command Remove fixed endstop compared to system function _disableMovingToEndStop)

Parameters of the MCC command Remove fixed endstop	Parameters of the system function _disableMovingToEndStop
Axis	axis
CommandID variable	commandId
Return variable	–
Delay program execution	nextCommand

Switch on torque limitation

This command is used to enable torque limitation in parallel to the motion. The torque limitation acts immediately.

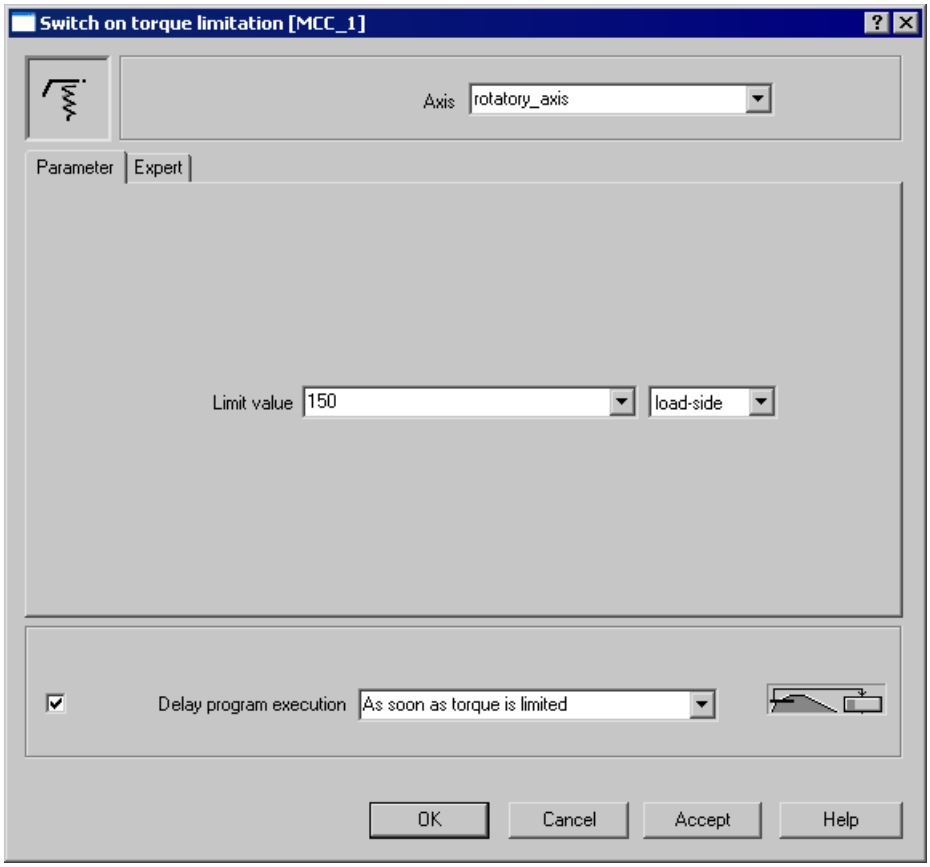


Figure 7-127 Parameter screen form: Switch on torque limitation

For additional information on **torque limitation**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Switch on torque limitation

Table 7-171 Overview of parameters for Switch on torque limiting

Field/button	Meaning/instruction
Axis	In Axis, select which axis the command is being programmed for. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Switch on torque limiting – Parameters tab (Page 4308)
Expert tab	See Overview of parameters for Switch on torque limiting – Expert tab (Page 4310)
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **If command has been completed or aborted** The next command is executed after the current command has been completed or aborted. **As soon as the torque is limited** The next command is executed as soon as limiting is initiated. **As soon as torque limiting is disabled** The next command is executed as soon as limiting is disabled. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Switch on torque limitation – Parameters tab

Table 7-172 Overview of parameters for Switch on torque limitation – Parameters tab

Field/button	Meaning/instruction
Limiting value	Value to which the torque or force is limited How this value is interpreted, is specified in the **Unit** field. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Last programmed** **Default value** See: Selection list (combo box) (Page 4022) System variable for default: userDefaultTorqueLimiting.torqueLimit. **Note:** In the **Unit** field, specify how this value will be interpreted: • As an absolute value on the motor side or load side in the unit selected on the axis. • As a percentage value of the default value on the motor side or load side.

Field/button	Meaning/instruction
Units	Definition of how the value in the **Limiting value** field will be interpreted. **Load side** The value in the **Limiting value** field will be interpreted as an absolute value for the load side: • For rotary axes: As torque in the unit selected on the axis, e.g. Nm • For linear axes: As force in the unit selected on the axis, e.g. N **Motor side** The value in the **Limiting value** field will be interpreted as an absolute value for the drive side: • For standard motors on rotary or linear axes: As torque in the unit selected on the axis, e.g. Nm • With linear motors: As force in the unit selected on the axis, e.g. N **Load side %** The interpretation of the value in the **Limiting value** field depends on whether a value has been entered directly or an option has been selected: • With direct value specification: The entered value is interpreted as a percentage of the default value (system variable userDefaultTorqueLimiting.torqueLimit). The absolute limiting value acts on the load side and is calculated as follows: Limiting value * userDefaultTorqueLimiting.torqueLimit / 100 – For rotary axes: As torque in the unit selected on the axis, e.g. Nm – For linear axes: As force in the unit selected on the axis, e.g. N • If you select the entry **Last programmed** or **Default value**: Same as the **Load side** selection. **Motor side %** The interpretation of the value in the **Limiting value** field depends on whether a value has been entered directly or an option has been selected: • With direct value specification: The entered value is interpreted as a percentage of the default value (system variable userDefaultTorqueLimiting.torqueLimit). The absolute limiting value acts on the drive side and is calculated as follows: Limiting value * userDefaultTorqueLimiting.torqueLimit / 100 – For standard motors on rotary or linear axes: As torque in the unit selected on the axis, e.g. Nm – With linear motors: As force in the unit selected on the axis, e.g. N • If you select the entry **Last programmed** or **Default value**: Same as the **Motor side** selection.

Overview of parameters for Switch on torque limitation – Expert tab

Table 7-173 Overview of parameters for Switch on torque limitation – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Switch on torque limitation

Cam technology package:

- _enableTorqueLimiting

Overview of parameters for Switch on torque limiting / _enableTorqueLimiting

Table 7-174 Parameters (MCC command Switch on torque limiting compared to system function _enableTorqueLimiting)

Parameters of the MCC command Switch on torque limiting	Parameters of the system function _enableTorqueLimiting
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Limiting value	torquelimittype, torquelimit
Unit	torquelimitunit
Expert tab	
CommandID variable	commandId
Return variable	–

Deactivate torque limitation

This command is used to switch off torque limitation superimposed on motion commands.

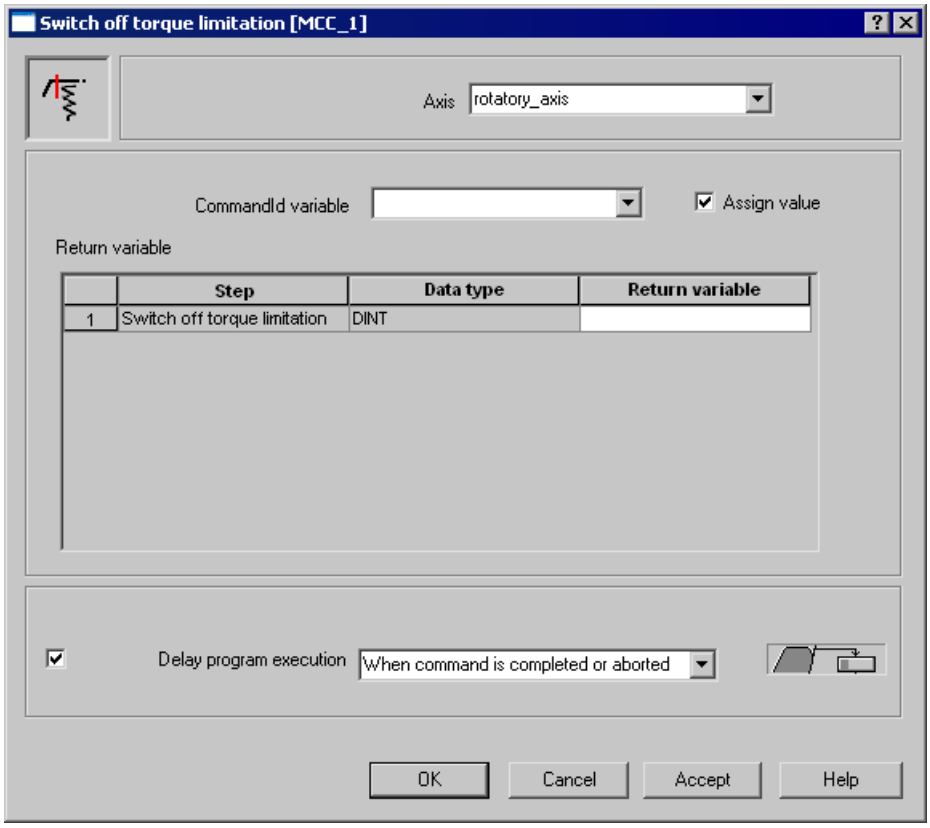


Figure 7-128 Parameter screen form: Switch off torque limitation

For additional information on **torque limitation**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Switch off torque limitation

Table 7-175 Overview of parameters for Switch off torque limiting

Field/button	Meaning/instruction
Axis	In Axis, select which axis the command is being programmed for. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object data type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
CommandID variable	Enter the name of a variable of data type CommandIdType here. With the help of this variable, you can, for example, trace back the status of the command. Variables of data type CommandIdType declared in the MCC unit or MCC chart are provided for selection. If you leave the CommandID variable field empty, the CommandID is not assigned to any variables; this means that you cannot access the CommandID (standard). See also Overview of parameters for the Expert tab (Page 4034).

Field/button	Meaning/instruction
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **If command has been completed or aborted** The next command is executed after the current command has been completed or aborted. See also Delay program execution (step enabling condition) (Page 4037).

Relevant system function for Switch off torque limitation

Cam technology package:

- _disableTorqueLimiting

Overview of parameters for switch off torque limiting / _disableTorqueLimiting

Table 7-176 Parameters (MCC command Switch off torque limiting compared to system function _disableTorqueLimiting)

Parameters of the MCC command Switch off torque limiting	Parameters of the system function _disableTorqueLimiting
Axis	axis
CommandID variable	commandId
Return variable	–
Delay program execution	nextCommand

Time-dependent velocity profile

With this command, the axis moves along a velocity profile preset by a cam. The definition range (x-axis) of the cam is interpreted as time, the value range (y-axis) as the corresponding velocity.

The profile is traversed from a selectable starting time to the end of the cam.

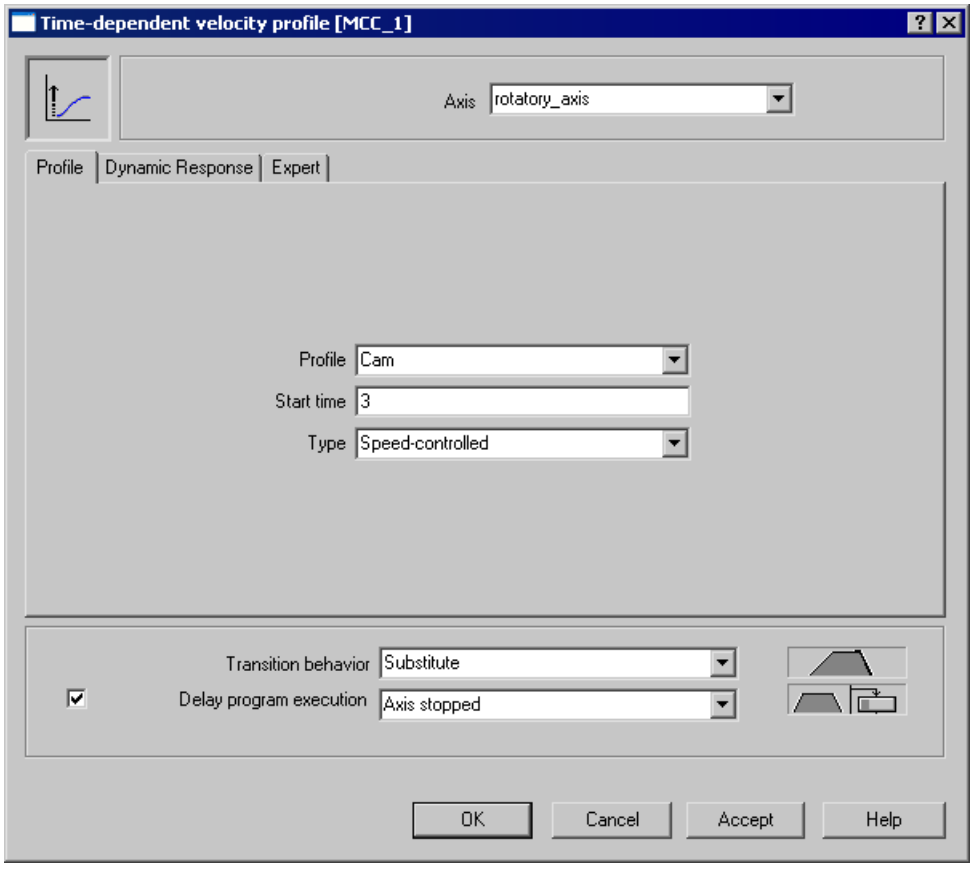


Figure 7-129 Parameter screen form: Time-dependent velocity profile

For additional information on the **time-dependent velocity profile**, see the TO Axis Electric/ Hydraulic Function Manual.

For additional information on the **position-related velocity profile**, see the TO Axis Electric/ Hydraulic Function Manual.

Overview of parameters for Time-dependent velocity profile

Table 7-177 Overview of parameters for Time-dependent velocity profile

Field/button	Meaning/instruction
Axis	In Axis, select the axis that is to move along the velocity profile. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Profile tab	See Overview of parameters for Time-dependent velocity profile – Profile tab (Page 4314)
Dynamic response tab	See Overview of parameters for Time-dependent velocity profile – Dynamic response tab (Page 4314)

Field/button	Meaning/instruction
Expert tab	See Overview of parameters for Time-dependent velocity profile – Expert tab (Page 4315)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Time-dependent velocity profile – Profile tab

Table 7-178 Overview of parameters for Time-dependent velocity profile – Profile tab

Field/Button	Explanation/Instruction
Profile	Here, you select the cam that describes the time-dependent velocity profile. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All technology object-type variables declared in the MCC source file or MCC chart (see Technology object data types (Page 4060)): camType.
Start time	Here you define the start time within the cam for profile execution. If you enter 0.0, the profile is executed from the starting point of the cam. Enter the value as a floating-point number. See also input field (Page 4022).
Type	Here, you determine whether the motion should be performed in a speed-controlled or position-controlled mode. **Position-controlled**(default value) Motion is position-controlled. **Closed-loop speed controlled** Motion is speed-controlled.

Overview of parameters for Time-dependent velocity profile – Dynamic response tab

Table 7-179 Overview of parameters for Time-dependent velocity profile – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). Just like all the parameters in the Dynamic response tab, this parameter is only evaluated while approaching and exiting the profile.
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile

Field/Button	Explanation/Instructions
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk

Overview of parameters for Time-dependent velocity profile – Expert tab

Table 7-180 Overview of parameters for Time-dependent velocity profile – Expert tab

Field/Button	Explanation/instructions
	The tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Time-dependent velocity profile

Cam technology package:

- _runTimeLockedVelocityProfile

Overview of parameters for Time-dependent velocity profile / _runTimeLockedVelocityProfile

Table 7-181 Parameters (MCC command Time-dependent velocity profile compared to system function _runTimeLockedVelocityProfile)

Parameters of the MCC command Time-dependent velocity profile	Parameters of the system function _runTimeLockedVelocityProfile
Axis	axis
Transition behavior	mergemode
Delay program execution	nextCommand
Profile tab	
Profile	profile
Start time	starttime

Parameters of the MCC command Time-dependent velocity profile	Parameters of the system function _runTimeLockedVelocityProfile
Type	movingmode
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Time-dependent position profile

With this command, the axis moves along a position profile preset by a cam. The definition range (x-axis) of the cam is interpreted as time, the value range (y-axis) as the corresponding position.

The profile is traversed from a selectable starting time to the end of the cam.

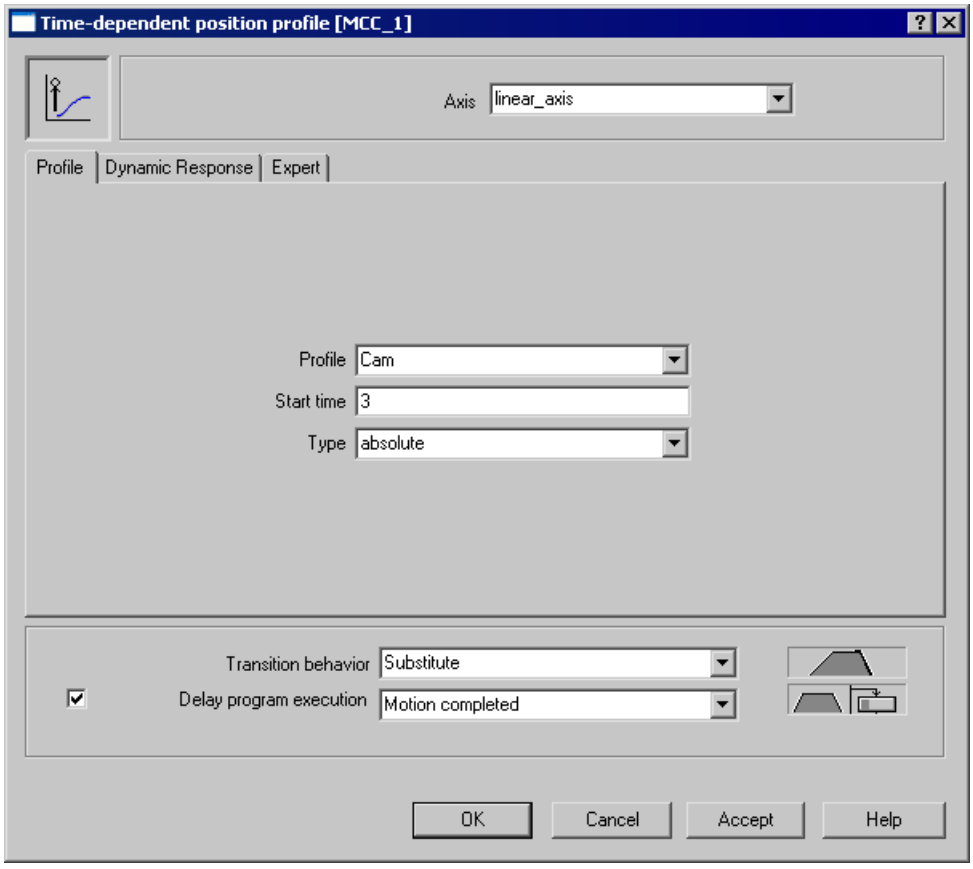


Figure 7-130 Parameter screen form: Time-dependent position profile

For additional information on the **time-dependent position profile**, see the TO Axis Electric/ Hydraulic Function Manual.

Overview of parameters for Time-dependent position profile

Table 7-182 Overview of parameters for Time-dependent position profile

Field/button	Meaning/instruction
Axis	In Axis, select the axis that is to move along the position profile. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart (see Description of the technology object data types (Page 4060)): PosAxis, FollowingAxis or _PathAxis.
Profile tab	See Overview of parameters for Time-dependent position profile – Profile tab (Page 4318)
Dynamic response tab	See Overview of parameters for Time-dependent position profile – Dynamic response tab (Page 4318)
Expert tab	See Overview of parameters for Time-dependent position profile – Expert tab (Page 4319)

Field/button	Meaning/instruction
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Time-dependent position profile – Profile tab

Table 7-183 Overview of parameters for Time-dependent position profile – Profile tab

Field/Button	Explanation/Instruction
Profile	Here, you select the cam that describes the time-dependent position profile. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All technology object-type variables declared in the MCC source file or MCC chart (see Description of the technology object data types (Page 4060)): camType.
Start time	Here you define the start time within the cam for profile execution. If you enter 0.0, the profile is executed from the starting point of the cam. Enter the value as a floating-point number. See also input field (Page 4022).
Type	Here you can decide how the positions defined by the cam are approached: **Absolute**(default value) The y-values of the cam represent absolute position values. **Relative** The y-values of the cam represent relative references to the current axis position.

Overview of parameters for Time-dependent position profile – Dynamic response tab

Table 7-184 Overview of parameters for Time-dependent position profile – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). Just like all the parameters in the Dynamic response tab, this parameter is only evaluated while approaching and exiting the profile.
Velocity	The entered value acts during the constant velocity phase. System variable for default: userDefaultDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefaultDynamics.profile

Field/Button	Explanation/Instructions
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefaultDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefaultDynamics.positiveAccelStartJerk userDefaultDynamics.positiveAccelEndJerk userDefaultDynamics.negativeAccelStartJerk userDefaultDynamics.negativeAccelEndJerk

Overview of parameters for Time-dependent position profile – Expert tab

Table 7-185 Overview of parameters for Time-dependent position profile – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Time-dependent position profile

Cam technology package:

- _runTimeLockedPositionProfile

Overview of parameters for Time-dependent position profile / _runTimeLockedPositionProfile

Table 7-186 Parameters (MCC command Time-dependent position profile compared to system function _runTimeLockedPositionProfile)

Parameters of the MCC Time-dependent position profile command	Parameters of the _runTimeLockedPositionProfile system function
Axis	axis
Transition behavior	mergemode
Delay program execution	nextCommand
Profile tab	
Profile	profile
Start time	starttime

Parameters of the MCC Time-dependent position profile command	Parameters of the _runTimeLockedPositionProfile system function
Type	profiledatamode
Dynamic response tab	
Velocity	velocitytype, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Shift measuring system

This command redefines the actual position. It affects the programming, system variable, and display. You can choose whether to set either a new actual value or a new setpoint.

Note

The command is not associated with a motion.

Actual value and setpoint can also be altered by a defined amount while the axis is in motion (in the case of type relative).

Relative motions are not affected.

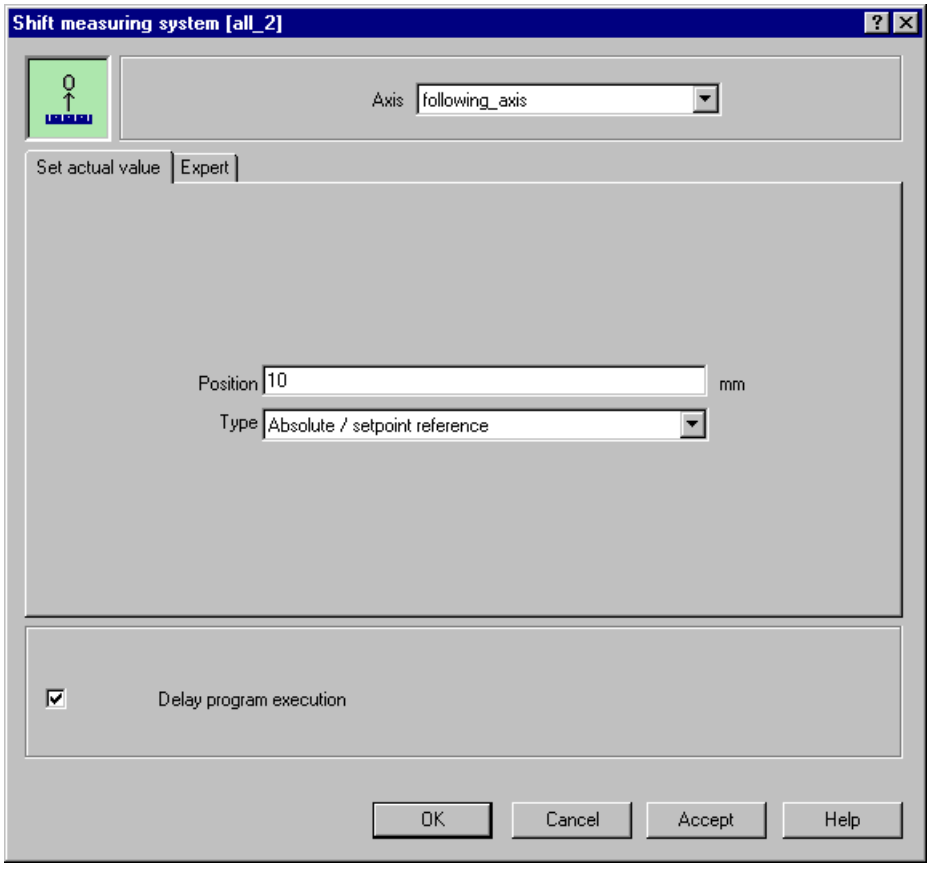


Figure 7-131 Parameter screen form: Shift measuring system

For additional information on **shifting the coordinate system**, see the TO Axis Electric/ Hydraulic Function Manual.

Overview of parameters for Shift measuring system

Table 7-187 Overview of parameters for Shift measuring system

Field/button	Meaning/instruction
Axis	Here, select the axis for which the position is to be reset. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object data type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Set actual value tab	See Overview of parameters for Shift measuring system – Set actual value tab (Page 4322)
Expert tab	See Overview of parameters for Shift measuring system – Expert tab (Page 4322)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Shift measuring system – Set actual value tab

Table 7-188 Overview of parameters for Shift measuring system – Set actual value tab

Field/Button	Explanation/Instruction
Position	Here, you enter the new position value. See also input field (Page 4022).
Type	Here, you select the position type. **Absolute / actual value reference** The programmed position is set as a new actual value, and the setpoint is corrected, taking into account the following error, and set. **Absolute/setpoint reference (default value)** The programmed position is set as a new setpoint, and the actual value is corrected, taking into account the following error, and set. **Absolute/setpoint reference of superimposed coordinate system** For superimposed motions: The programmed position is set as a new setpoint of the superimposed motion, and the actual value is corrected, taking into account the following error, and set. **Absolute/setpoint reference of basic coordinate system** For superimposed motions: The programmed position is set as the new setpoint of the main motion, and the actual value is corrected, taking into account the following error, and set. **Relative** The programmed position is added to the current setpoint, actual value.

Overview of parameters for Shift measuring system – Expert tab

Table 7-189 Overview of parameters for Shift measuring system – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Shift measuring system

Cam technology package:

- _redefinePosition

Overview of parameters for Shift measuring system / _redefinePosition

Table 7-190 Parameters (MCC command Shift measuring system compared to system function _redefinePosition)

Parameters of the MCC command Shift measuring system	Parameters of the system function _redefinePosition
Axis	axis
Delay program execution	nextCommand
Set actual value tab	
Position	position
Type	redefineSpecification, redefineMode
Expert tab	
CommandID variable	commandId
Return variable	–

Online correction

The correction value you enter for overriding velocity and acceleration acts on currently active commands and all following commands.

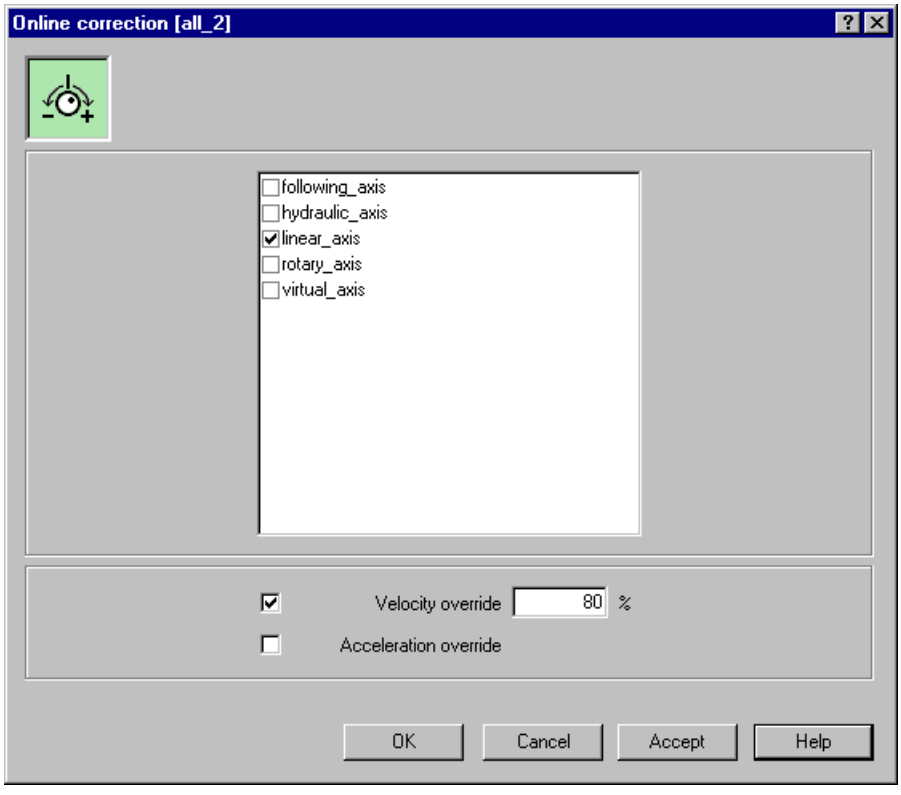


Figure 7-132 Parameter screen form: Online correction

For additional information on **online correction (override)**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Online correction

Table 7-191 Overview of parameters for Online correction

Field/button	Meaning/instruction
Axis	Select the axes for which you wish to program an override. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Velocity override	• Select this checkbox if you want to program a velocity override. • Enter the override factor here. The current velocity value is corrected by the specified percentage value. Units: Percentage Range of values: 0 - 200 The value acts on all presently active commands as well as on all of the subsequent commands. See also Input fields and selection lists (Page 4022).
Acceleration override	• Select this checkbox if you want to program an acceleration override. • Enter the override factor here. The current acceleration value is corrected by the specified percentage value. Units: Percentage Range of values: 1 - 1000 The value acts on all presently active commands as well as on all of the subsequent commands. See also Input fields and selection lists (Page 4022).

Relevant system variable for Online correction

System variable for drive axis, position axis and following axis technology objects:

- override

Table 7-192 Adjusts the velocity and acceleration override to correct the current velocity and acceleration by the specified percentage

Parameters of the MCC command Online correction	System variable of the technology object *axis*
Axis	
Velocity override	_to.*axis*.override.velocity
Acceleration override	_to.*axis*.override.acceleration

Set axis parameter

This command changes the default parameters (USERDEFAULT) for the specified axis.

The programmed axis parameters go into effect when the relevant parameter is set to the default in a subsequent motion command.

The USERDEFAULT value settings made during axis configuration take effect again after:

- Power On
- The **Reset object** command is made by activating the **Initialize system variables** checkbox.

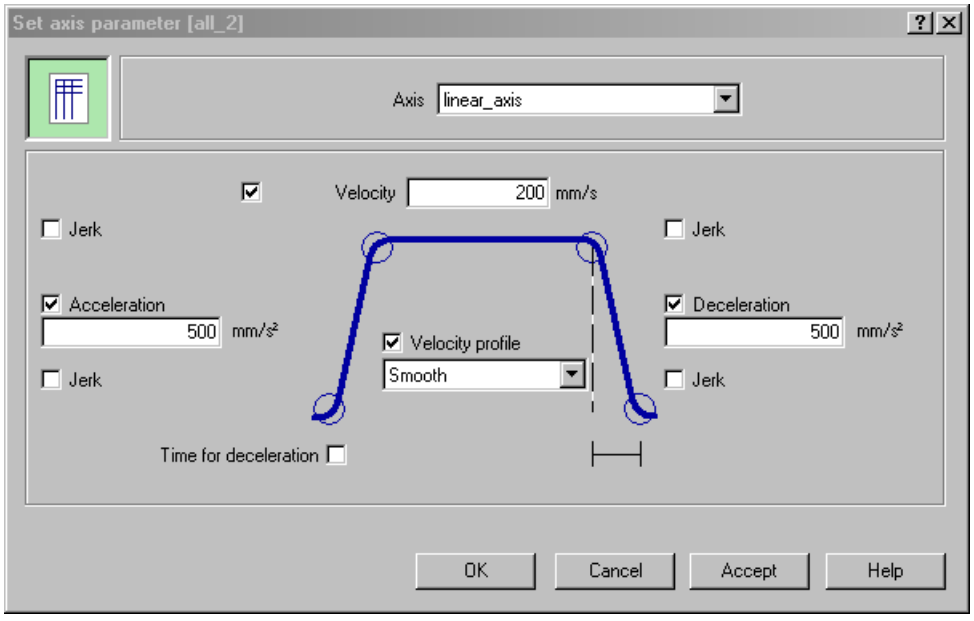


Figure 7-133 Parameter screen form: Set axis parameter

For additional information on **setting axis parameters**, see the TO Axis Electric/Hydraulic Function Manual.

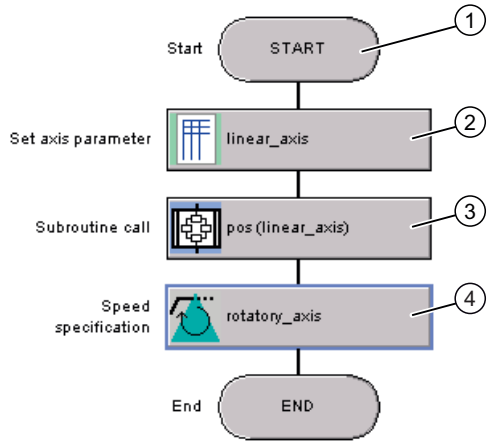
Overview of parameters for Set axis parameter

Table 7-193 Overview of parameters for Set axis parameter

Field/button	Meaning/instruction
Axis	In Axis, select the configured axis for which you wish to set one or more new preassigned values. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Speed	For speed-controlled axes only Default value for the speed during the constant speed phase. System variable: userDefaultDynamics.velocity • Select this checkbox if you want to alter the default value for the speed. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Velocity	Only for position, synchronized and path axes Preassigned value for velocity during the constant velocity phase. System variable: userDefaultDynamics.velocity • Select this checkbox if you want to alter the default value for velocity. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Velocity profile	Preassigned value for the velocity profile System variable: userDefaultDynamics.profile The velocity profile is used to specify the transitions between the individual motion phases. The velocity profile influences the motion transitions as follows: • At the start and end of the acceleration phase, the transition to the constant acceleration or constant velocity phase • At the start and end of the deceleration phase, the transition to the constant deceleration or constant velocity phase Proceed as follows to change the preassigned value for the velocity profile: • Activate the checkbox. • Select the preassigned value. **Smooth** A velocity profile with smooth acceleration characteristic takes effect for this command. Application: Velocity profile with controllable jerk characteristic **Trapezoidal** (default value) A trapezoidal velocity profile takes effect for this command. Acceleration and deceleration can be specified. See also Overview of parameters in the Dynamic response tab (Page 4031).

Field/button	Meaning/instruction
Acceleration	Preassigned value for the acceleration during the constant acceleration phase. System variable for default: userDefaultDynamics.positiveAccel • Select this checkbox if you want to alter the default value for acceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Deceleration	Preassigned value for deceleration during the constant acceleration phase. System variable: userDefaultDynamics.negativeAccel • Select this checkbox if you want to alter the default value for deceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Jerk at start of acceleration	Default value for the jerk (change in acceleration) at the start of the acceleration phase. System variable: userDefaultDynamics.positiveAccelStartJerk • Activate this checkbox if you want to alter the default value for the jerk at the start of the acceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Jerk at end of acceleration	Default value for the jerk (change in acceleration) at the end of the acceleration phase. System variable: userDefaultDynamics.positiveAccelEndJerk • Activate this checkbox if you want to alter the default value for the jerk at the end of the acceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Jerk at start of deceleration	Default value for the jerk (change in deceleration) at the start of the deceleration phase. System variable: userDefaultDynamics.negativeAccelStartJerk • Activate this checkbox if you want to alter the default value for the jerk at the start of the deceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Jerk at end of deceleration	Default value for the jerk (change in deceleration) at the end of the deceleration phase. System variable: userDefaultDynamics.negativeAccelEndJerk • Activate this checkbox if you want to alter the default value for the jerk at the end of the deceleration. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).
Time for deceleration	Default value for the duration of the braking operation for a quick stop within a defined period, see "Stop axis" (Page 4281). System variable: userDefaultDynamics.stopTime • Activate the checkbox if you want to alter the default value for the duration of the braking operation. • Enter the default value, see input field (Page 4022). See also Overview of parameters in the Dynamic response tab (Page 4031).

Example of Set axis parameter



① Velocity set during configuration in USERDEFAULT = 300 m/s.
② A velocity of 500 m/s is programmed as the USERDEFAULT.
③ If all motion commands in the subroutine have been programmed with the default, the velocity is 500 m/s.
④ A speed of 200 rpm is programmed. The axis rotates at this speed.

Figure 7-134 Example of Set axis parameter

Relevant system variables for Set axis parameter

System variable for drive axis, position axis and following axis technology objects:

- userDefaultDynamics

Table 7-194 Adjusts the default parameters of the specified axis

Parameters of the MCC command Set axis parameter	**System variable of the technology object *axis***
Axis	
Velocity	_to.*axis*.userDefaultDynamics.velocity
Velocity profile	_to.*axis*.userDefaultDynamics.profile
Time for deceleration	_to.*axis*.userDefaultDynamics.stopTime
Acceleration	_to.*axis*.userDefaultDynamics.positiveAccel
Deceleration	_to.*axis*.userDefaultDynamics.negativeAccel
Jerk at start of acceleration	_to.*axis*.userDefaultDynamics.positiveAccelStartJerk
Jerk at end of acceleration	_to.*axis*.userDefaultDynamics.positiveAccelEndJerk
Jerk at start of deceleration	_to.*axis*.userDefaultDynamics.negativeAccelStartJerk
Jerk at end of deceleration	_to.*axis*.userDefaultDynamics.negativeAccelEndJerk

Set virtual axis values

Use the Set virtual axis values command in order to transfer the position, velocity, and acceleration values for a real master axis or an encoder to a virtual axis. The axis can then, for example, be programmed using a positioning command.

The axis values of the virtual axis are set only when the master axis is moving. If the command is executed when the master axis is stationary, the axis values will not be set until the master axis is next started.

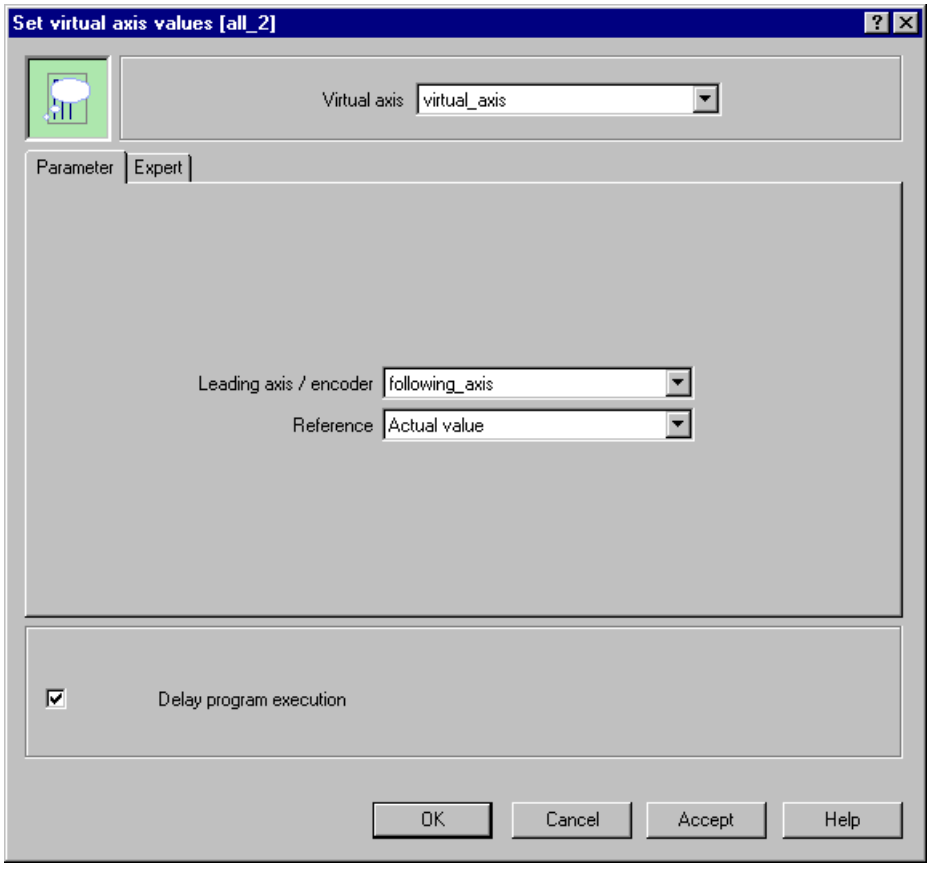


Figure 7-135 Parameter screen form: Set virtual axis values

For additional information on **shifting the coordinate system**, see the TO Axis Electric/ Hydraulic Function Manual.

Overview of parameters for Set virtual axis values

Table 7-195 Overview of parameters for Set virtual axis values

Field/button	Meaning/instruction
Virtual axis	Here, select the axis to be synchronized. All of the virtual axes of the device that have positioning or synchronous operation functionality are displayed.
Parameters tab	See Overview of parameters for Set virtual axis values – Parameters tab (Page 4330)
Expert tab	See Overview of parameters for Set virtual axis values – Expert tab (Page 4331)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Set virtual axis values – Parameters tab

Table 7-196 Overview of parameters for Set virtual axis values – Parameters tab

Field/button	Meaning/instruction
Leading axis / encoder / external master value	Here, select the axis for which the position, velocity, or acceleration is to be transferred to the virtual axis. The list contains: • All position, following, and path axes and external encoders that are available on the device or a DP master. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types: PosAxis, FollowingAxis, _PathAxis and ExternalEncoderType
Reference	Here, decide whether the virtual axis is to be synchronized to the setpoints or actual values of the leading axis. **Setpoint** The setpoints of the leading axis are accepted for the virtual axis. **Actual value**(default value) The actual values of the leading axis are accepted for the virtual axis.

See also

Description of the technology object data types (Page 4060)

Overview of parameters for Set virtual axis values – Expert tab

Table 7-197 Overview of parameters for Set virtual axis values – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Set virtual axis values

Cam technology package:

- _redefinePosition
  with parameter redefineSpecification := VIRTUAL_AXIS.

Overview of parameters for Set virtual axis values / _redefinePosition

Table 7-198 Parameters (MCC command Set virtual axis values compared to system function _redefinePosition)

Parameters of the MCC command Set virtual axis values	Parameters of the system function _redefinePosition
Virtual axis	axis
Delay program execution	nextCommand
Parameters tab	
Leading axis / encoder / external master value	
Reference	• For reference = actual value: redefineMode := ABSOLUTE position := *LeadingAxis*.positioningstate.actualposition velocity := *LeadingAxis*.motionstatedata.actualvelocity acceleration := *LeadingAxis*.motionstatedata.actualacceleration • For reference = setpoint: redefineMode := ABSOLUTE position := *LeadingAxis*.positioningstate.commandposition velocity := *LeadingAxis*.motionstatedata.commandvelocity acceleration := *LeadingAxis*.motionstatedata.commandacceleration
Expert tab	
CommandID variable	commandId
Return variable	–

Delete command queue

The command queue is a memory for motion commands that have been issued by the program, but are not yet active. Commands that have not yet been executed can be deleted from the command queue.

This command allows commands for an axis to be selectively deleted after the axis has been stopped or the motion sequence has been modified.

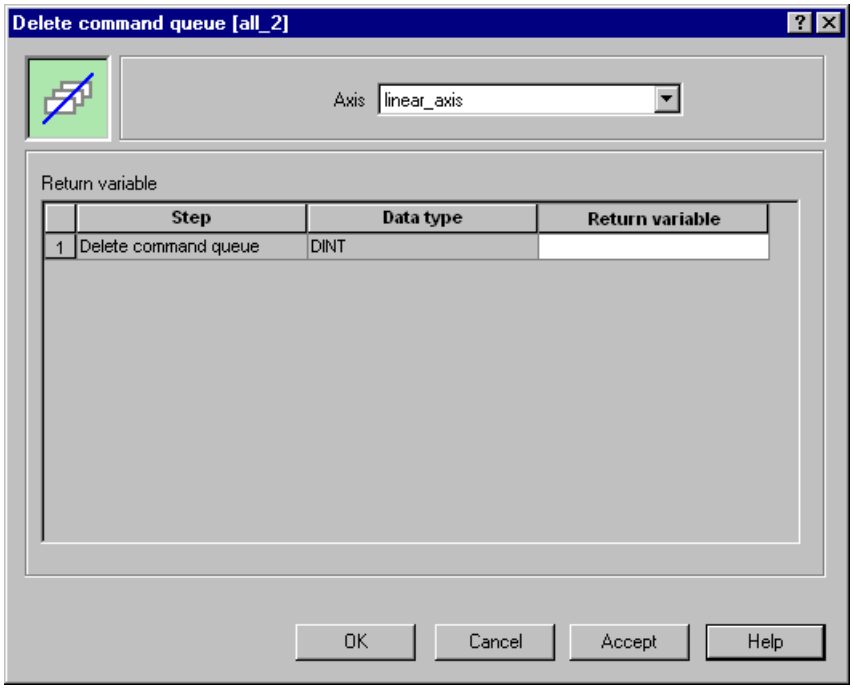


Figure 7-136 Parameter screen form: Delete command queue

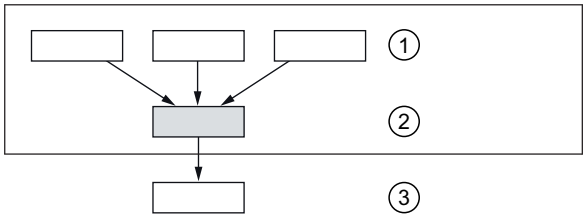
For additional information on **deleting the command queue (motion buffer)**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Delete command queue

Table 7-199 Overview of parameters for Delete command queue

Field/button	Meaning/instruction
Axis	In Axis, select the axis for which the command is to be programmed. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4060).

Example of Delete command queue



① Waiting
② Command is deleted
③ Current command (cannot be deleted)

Figure 7-137 Example of a command queue

Relevant system function for Delete command queue

Cam technology package:

- _resetMotionBuffer

Overview of parameters for Delete command queue /_resetMotionBuffer

Table 7-200 Parameters (MCC command Delete command queue compared to system function _resetMotionBuffer)

Parameters of the MCC command Delete command queue	Parameters of the system function _resetMotionBuffer
Axis	axis
Return variable	–

Switch parameter set

Using this command, you can switch over the active controller data set.

Several data sets can be assigned to one Axis technology object, e.g. for multiple measuring systems/encoders. You can switch among multiple measuring systems/encoders, for example, by means of data set changeover. All configured measuring systems are internally active, and the measured values are updated cyclically.

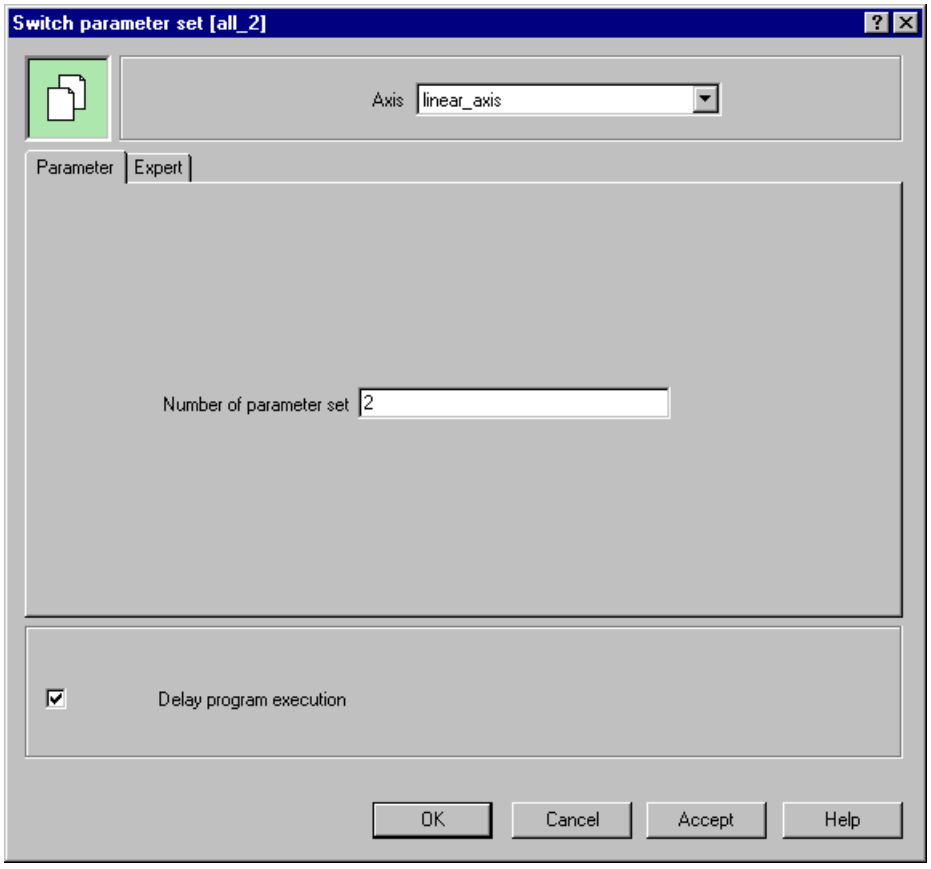


Figure 7-138 Parameter screen form: Switch parameter set

For additional information on **activating data sets**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Switch parameter set

Table 7-201 Overview of parameters for Switch parameter set

Field/button	Meaning/instruction
Axis	Here, select the axis for which the controller data set is to be switched over. The list contains: • All axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): DriveAxis, PosAxis, FollowingAxis, or _PathAxis.
Parameters tab	See Overview of parameters for Switch parameter set – Parameters tab (Page 4335)
Expert tab	See Overview of parameters for Switch parameter set – Expert tab (Page 4335)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Switch parameter set – Parameters tab

Table 7-202 Overview of parameters for Switch parameter set – Parameters tab

Field/Button	Explanation/Instructions
Number of the parameter set	Here, select the number of the parameter set that is to be activated. Numbers 1 to 31 are listed. The value can be entered directly or by means of a variable or a calculation formula. Use a drag-and-drop operation to move variables from the symbol browser. Drag-and-drop the system functions from the command library.

Overview of parameters for Switch parameter set – Expert tab

Table 7-203 Overview of parameters for Set virtual axis values – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Switch parameter set

Cam technology package:

- _setAxisDataSetActive

Overview of parameters for Switch parameter set / _setAxisDataSetActive

Table 7-204 Parameters (MCC command Switch parameter set compared to system function _setAxisDataSetActive)

Parameters of the MCC command Switch parameter set	Parameters of the system function _setAxisDataSetActive
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Number of the parameter set	dataSetNumber
Expert tab	
CommandID variable	commandId
Return variable	–

7.1.6.6 Commands for external encoders, measuring inputs and output cams

External encoder on

This command enables the external encoder for recording measured values.

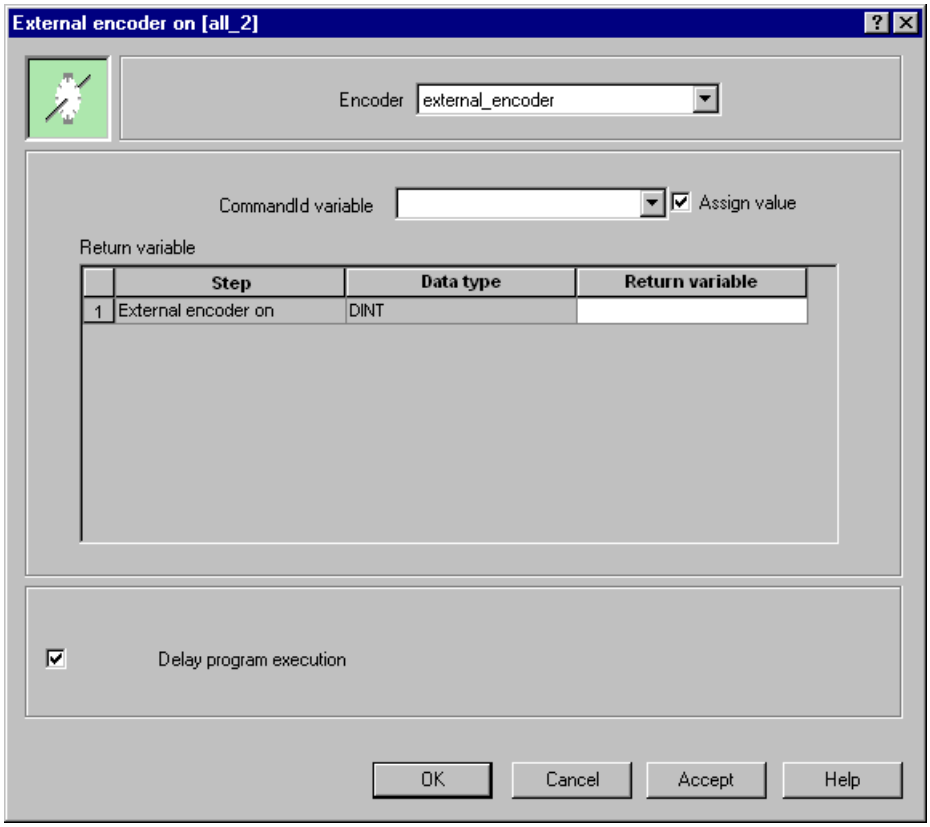


Figure 7-139 Parameter screen form: External encoder on

For additional information on the **external encoder**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for External encoder on

Table 7-205 Overview of parameters for External encoder on

Field/Button	Explanation/Instructions
Encoder	Here, you select the external encoder. The list contains: • All encoders that are defined on the relevant device. An external encoder is defined in the EXTERNAL ENCODERS folder in the project navigator. • All technology object-type variables declared in the MCC source file or MCC chart (see Description of the technology object data types (Page 4060)): externalEncoderType.
CommandID variable	Enter the name of a variable of data type CommandIdType here. With the help of this variable, you can, for example, trace back the status of the command. Variables of data type CommandIdType declared in the MCC source file or MCC chart are provided for selection. If you leave the CommandID variable field empty, the CommandID is not assigned to any variables; this means that you cannot access the CommandID (standard). See also Overview of parameters for the Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).
Delay program execution	Activate the check box if you wish to delay execution of the following command until the current command has been completed. If the check box is not activated, the next command is executed immediately.

Relevant system function for External encoder on

Cam technology package:

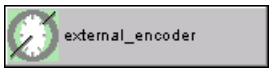
- _enableExternalEncoder

Overview of parameters for External encoder on / _enableExternalEncoder

Table 7-206 Parameters (MCC External encoder on command compared to _enableExternalEncoder system function)

Parameters of the MCC External encoder on command	Parameters of the _enableExternalEncoder system function
Encoder	externalEncoder
CommandID variable	commandId
Return variable	–
Delay program execution	nextCommand

External encoder off



This command disables the external encoder for recording measured values.

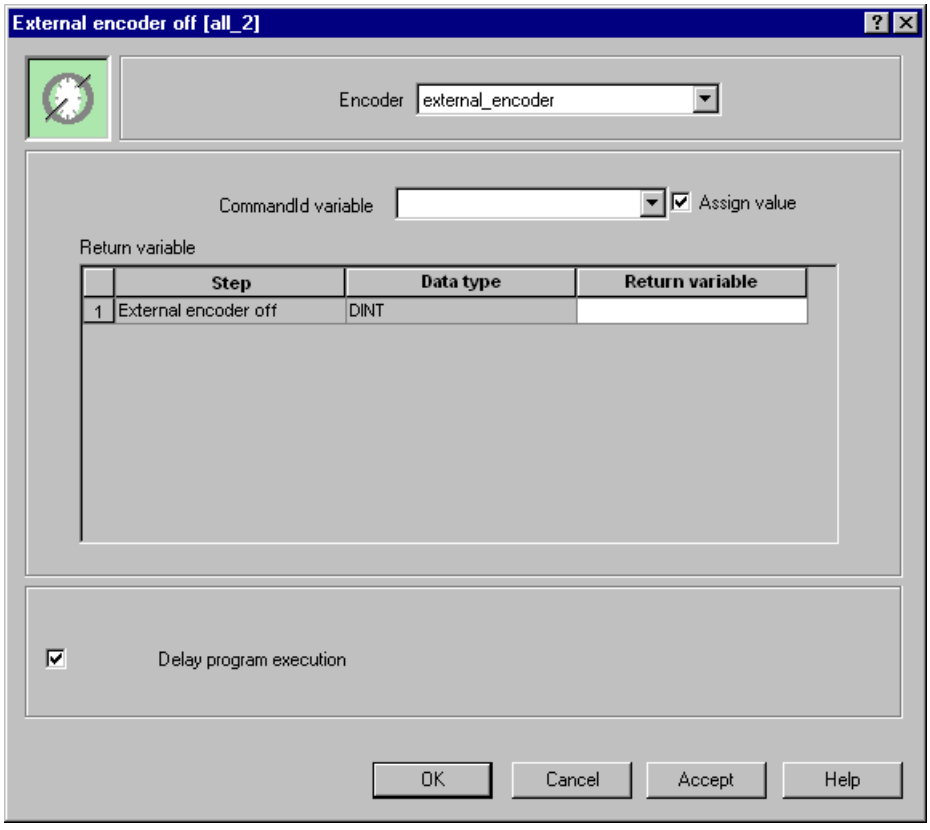


Figure 7-140 Parameter screen form: External encoder off

For additional information on the **external encoder**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for External encoder off

Table 7-207 Overview of parameters for External encoder off

Field/Button	Explanation/Instructions
Encoder	Here, you select the external encoder. The list contains: • All encoders that are defined on the relevant device. An external encoder is defined in the EXTERNAL ENCODERS folder in the project navigator. • All technology object-type variables declared in the MCC source file or MCC chart (see Description of the technology object data types (Page 4060)): externalEncoderType.
CommandID variable	Enter the name of a variable of data type CommandIdType here. With the help of this variable, you can, for example, trace back the status of the command. Variables of data type CommandIdType declared in the MCC source file or MCC chart are provided for selection. If you leave the CommandID variable field empty, the CommandID is not assigned to any variables; this means that you cannot access the CommandID (standard). See also Overview of parameters for the Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).
Delay program execution	Activate the check box if you wish to delay execution of the following command until the current command has been completed. If the check box is not activated, the next command is executed immediately.

Relevant system function for External encoder off

Cam technology package:

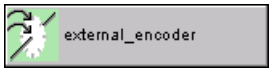
- _disableExternalEncoder

Overview of parameters for External encoder off/_disableExternalEncoder

Table 7-208 Parameters (MCC External encoder off command compared to _disableExternalEncoder system function)

Parameters of the MCC External encoder off command	Parameters of the _disableExternalEncoder system function
encoders	externalEncoder
CommandID variable	commandId
Return variable	–
Delay program execution	nextCommand

Synchronize external encoder



This command homes the measuring system.

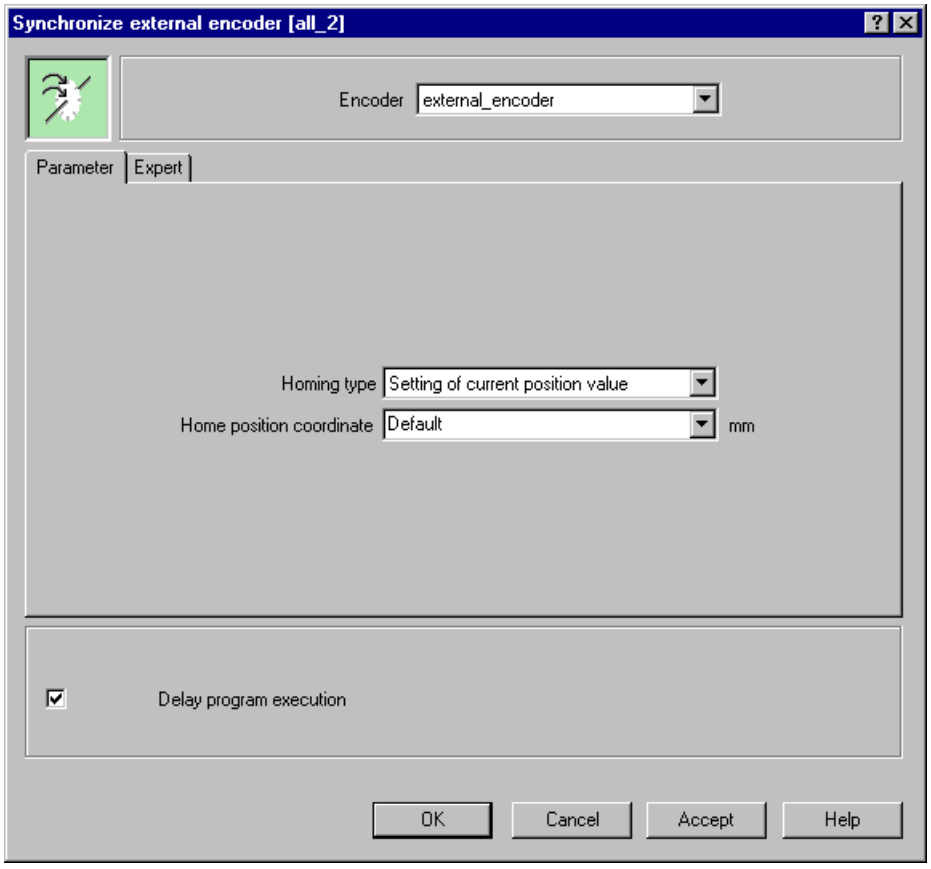


Figure 7-141 Parameter screen form: Synchronize external encoder

For additional information on the **external encoder**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Synchronize external encoder

Table 7-209 Overview of parameters for Synchronize external encoder

Field/Button	Explanation/Instructions
Encoder	Select the external encoder to be synchronized in this field. The list contains: • All encoders that are defined on the relevant device. An external encoder is defined in the EXTERNAL ENCODERS folder in the project navigator. • All technology object-type variables declared in the MCC source file or MCC chart (see Description of the technology object data types (Page 4060)): externalEncoder-Type.
Parameters tab	See Overview of parameters for Synchronize external encoder – Parameters tab (Page 4341)
Expert tab	See Overview of parameters for Synchronize external encoder – Expert tab (Page 4341)
Delay program execution	Activate the check box if you wish to delay execution of the following command until the current command has been completed. If the check box is not activated, the next command is executed immediately.

Overview of parameters for Synchronize external encoder – Parameters tab

Table 7-210 Overview of parameters for Synchronize external encoder – Parameters tab

Field/Button	Explanation/Instructions
Homing type	Here, you select the homing type. **Set home position** (default value) The home position coordinates are directly set to the current axis coordinates. The current axis position is assigned to the value of the home coordinates. There is no active traversing motion. **Passive homing** The next time that the axis moves, the first available zero mark is searched for and the measuring system is synchronized to this zero mark. There is no active traversing motion as a result of the homing command. **Absolute encoder adjustment** This selection is only available for absolute encoders, i.e. encoders that have been configured as type "absolute encoder" or "absolute encoder, cyclic absolute".
Home position coordinate	Coordinates of the home position in the reference system of the axis. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Last programmed home position coordinate** **Default value** See Selection list (combo box) (Page 4022) System variables for default: userDefaultHoming.homePosition

Overview of parameters for Synchronize external encoder – Expert tab

Table 7-211 Overview of parameters for Synchronize external encoder – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a CommandId-type variable, you can track the command status with this variable.
TO properties	Here, you can adapt the parameter dialog box as needed to reflect the effects of encoder configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Synchronize external encoder

Cam technology package:

- _synchronizeExternalEncoder

Overview of parameters for Synchronize external encoder / _synchronizeExternalEncoder

Table 7-212 Parameter (MCC Synchronize external encoder command compared to _synchronizeExternalEncoder system function)

Parameters of the MCC Synchronize external encoder command	**Parameters of the _synchronizeExternalEncoder system function**
Encoder	externalEncoder
CommandID variable	commandId
Delay program execution	nextCommand
Parameters tab	
Homing type	synchronizingMode
Home position coordinate	syncPositionType, syncPosition
Expert tab	
CommandID variable	commandId
TO properties	–
Return variable	–

Encoder monitoring on

This command enables encoder monitoring.

At least two encoder systems must be configured for one axis for this purpose. The difference between the two encoder systems is monitored; an alarm is generated when the maximum permissible difference is exceeded.

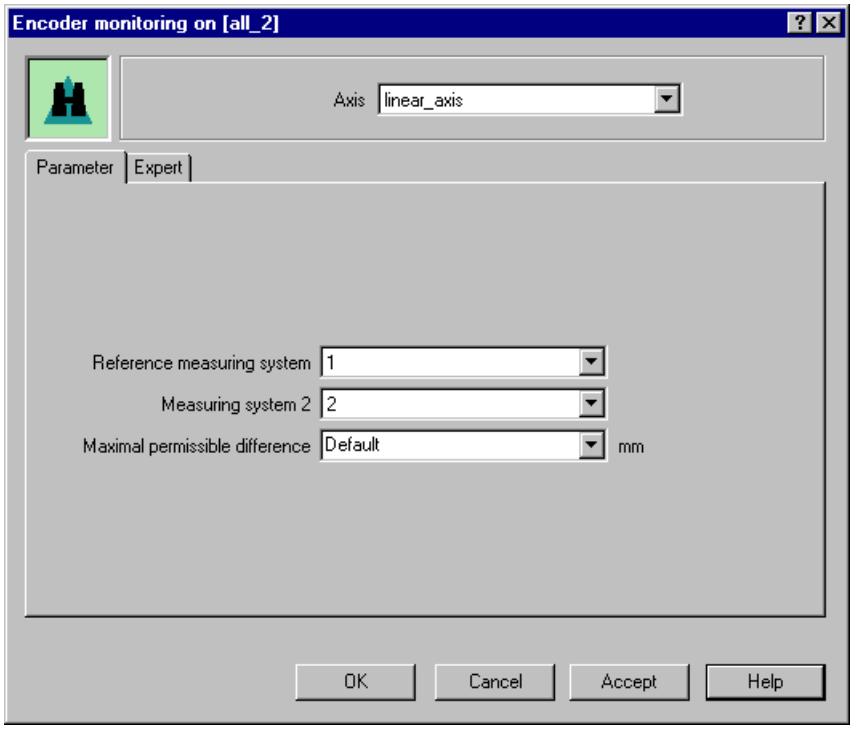


Figure 7-142 Parameter screen form: Encoder monitoring on

For additional information on the **external encoder**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Encoder monitoring on

Table 7-213 Overview of parameters for Encoder monitoring on

Field/Button	Explanation/instructions
Axis	Here, you select the axis for which encoder monitoring is to be enabled. At least two encoder systems must have been configured for the selected axis during commissioning (configuration data TypeOfAxis.NumberOfEncoders.numberOfEncoders > 1).
Parameters tab	See Overview of parameters for Encoder monitoring on - Parameter tab (Page 4344)
Expert tab	See Overview of parameters for Encoder monitoring on - Expert tab (Page 4344)

Overview of parameters for Encoder monitoring on – Parameters tab

Table 7-214 Overview of parameters for Encoder monitoring on – Parameters tab

Field/Button	Explanation/Instructions
Reference measuring system	Select the reference measuring system from the encoder systems configured for the axis.
Measuring system 2	Select the second measuring system from the remaining encoder systems configured for the axis.
Maximum permissible difference	Maximum permissible difference between the two encoder systems. Enter the value in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default value** See Selection list (combo box) (Page 4022) System variable for default: sensormonitoring.MaximalSensorDifference

Overview of parameters for Encoder monitoring on – Expert tab

Table 7-215 Overview of parameters for Encoder monitoring on – Expert tab

Field/Button	Explanation/instructions
	The Expert tab is described in detail in Expert tab (Page 4034).
TO properties	Here, you can adapt the parameter dialog box as needed to reflect the effects of encoder configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Encoder monitoring on

Cam technology package:

- _enableMonitoringOfEncoderDifference

Overview of parameters for Encoder monitoring on / _enableMonitoringOfEncoderDifference

Table 7-216 Parameters (MCC Encoder monitoring on command compared to _enableMonitoringOfEncoderDifference system function)

Parameters of the MCC Encoder monitoring on command	Parameters of the _enableMonitoringOfEncoderDifference system functions
Axis	externalEncoder
Parameters tab	
Reference measuring system	referenceencodertype, referenceEncoder, encoder
Measuring system 2	–
Maximum permissible difference	maximalencoderdifferencetype, maximalencoderdifference
Expert tab	

Parameters of the MCC Encoder monitoring on command	Parameters of the _enableMonitoringOfEncoderDifference system functions
TO properties	–
Return variable	–

Encoder monitoring off

This command disables encoder monitoring.

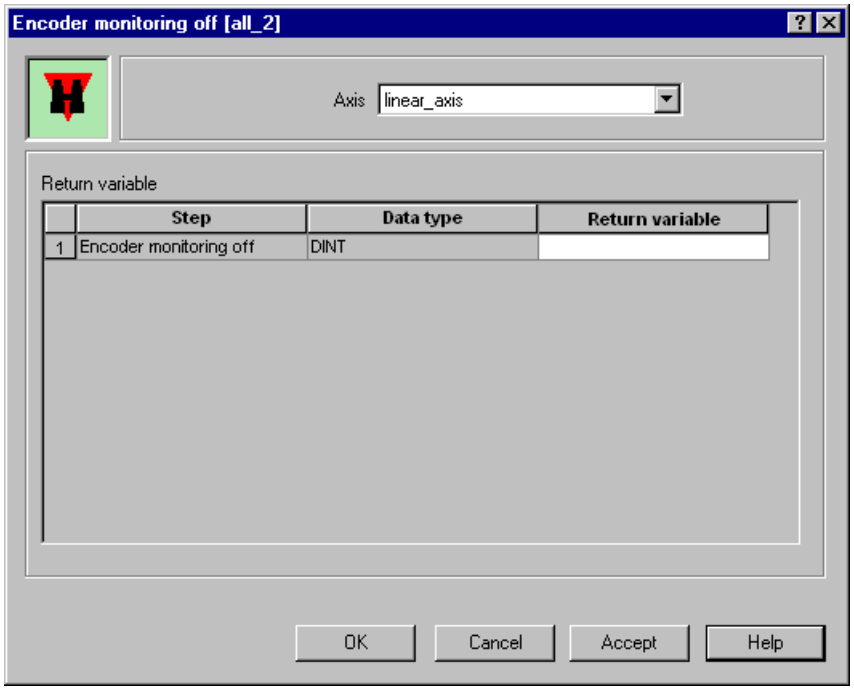


Figure 7-143 Parameter screen form: Encoder monitoring off

For additional information on the **external encoder**, see the TO Axis Electric/Hydraulic Function Manual.

Overview of parameters for Encoder monitoring off

Table 7-217 Overview of parameters for Encoder monitoring off

Field/Button	Explanation/instructions
Axis	Here, you select the axis for which encoder monitoring is to be disabled. At least two encoder systems must have been configured for the selected axis during commissioning (configuration data TypeOfAxis.NumberOfEncoders.numberOfEncoders > 1).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Encoder monitoring off

Cam technology package:

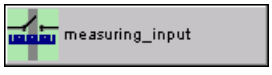
- _disableMonitoringOfEncoderDifference

Overview of parameters for Encoder monitoring off / _disableMonitoringOfEncoderDifference

Table 7-218 Parameters (MCC Encoder monitoring off command compared to _disableMonitoringOfEncoderDifference system function)

Parameters of the MCC Encoder monitoring off command	Parameters of the _disableMonitoringOfEncoderDifference system function
Axis	axis
Return variable	–

Activate measuring input



This command activates a measuring input for an axis or external encoder. A change in the signal (edge) at the configured measuring input (e.g. on the SIMOTION device or drive) causes the current act. position val. to be saved.

You can select how often the measurement is performed:

- One-time: The measuring process must be reactivated for each new measurement. Several IPO cycle clocks must be performed between the measurements.
- The measuring accuracy depends on the accuracy of the hardware used. It lies in the range of microseconds.
- Cyclic: The measurement is made cyclically in the configured processing cycle clock of the Measuring Input technology object.
  Cyclical measuring can only be performed with the following devices:
  - TM17 High Feature
  - C240 (inputs B1 to B4)
  - SIMOTION D (onboard measuring inputs)
  - SIMOTION CX32/CX32-2 Controller Extension
  - SINAMICS S120 Control Units

  When using global measuring inputs, you can detect and measure up to two edges in each processing cycle (a maximum of 2 edges every 3 servo cycles with D4xx, D4x5-2 X122/X132, and CX32/CX32-2 devices and SINAMICS S120 Control Units).
  The measured values must be read out by the user program before they are overwritten by a new measurement, i.e. if possible, in the SynchronousTask that corresponds to the processing cycle clock of the Measuring Input technology object.

For both types, you can also specify:

- The triggering edge of the input signal (e.g. rising, falling, both)
- A measuring range in which the measured value will be acquired

If the measurement is successful, the measurement result is available in the system variables below in the next cycle clock:

1. The measured values are stored in system variables measuredValue1 and measuredValue2 (for meaning, see the table below).
2. With one-time measuring, system variable state has the value TRIGGER_OCCURED.
3. With cyclic measuring, system variables counterMeasuredValue1 and counterMeasuredValue2 are incremented by 1 on each measurement result. New results can be traced immediately and can also be read from non-IPO-synchronous tasks.
   The only time these counters are set to 0 is in the event of a system ramp-up or technology object reset.

Table 7-219 Content of system variables measuredValue1 and measuredValue2 according to the measurement and the triggering edge.

Measurement	Trigger	measuredValue1	measuredValue2
once-only	One edge (rising or falling)	Measured value [1]	
	Two edges	1st measured value chronologically[1]	2nd measured value chronologically

Measurement	Trigger	measuredValue1	measuredValue2
Cyclic	One edge (rising or falling)	1st measured value chronologically[2]	2nd measured value chronologically[3][4]
	Two edges	Measured value at 1st rising edge[2]	Measured value at 1st falling edge[4]

1. In addition, system variable 'state' has the value TRIGGER_OCCURED.
2. In addition, system variable counterMeasuredValue1 is incremented by 1.
3. With 2 measured values in the same processing cycle clock.
4. In addition, system variable counterMeasuredValue2 is incremented by 1.

With one-time measuring, the command is active until the measurement result is received or the command is terminated by the "Deactivate measuring input" (Page 4353) command.

With cyclic measurements, the command is active until it is terminated by the "Deactivate measuring input" (Page 4353) command.

Various response times must be taken into account in the application for the measuring function, depending on the axis/external encoder connection (onboard C2xx, PROFIBUS axis), the drive used (611U, MASTERDRIVES MC, SINAMICS), and the processing cycle clock (IPO, IPO2 or position control cycle clock).

An Excel tool, which provides support when determining these times, is included in the SIMOTION Utilities & Applications. Depending on the type of measuring input used, you can determine the response times for the measuring input function and also the lead times for dynamic switching processes within the measuring range.

SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

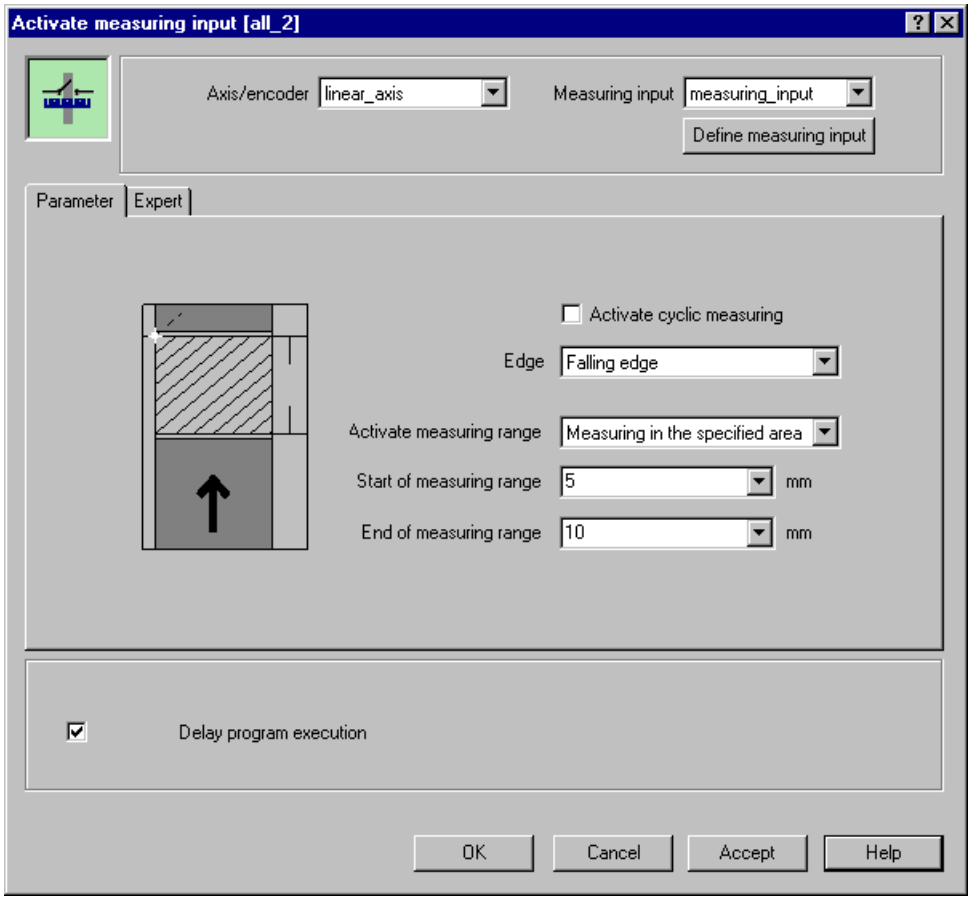


Figure 7-144 Parameter screen form: Activate measuring input

Further information:

- For information on the **Measuring Input technology object**, see the Output Cams and Measuring Inputs Function Manual
- For information on **commands on the Measuring Input technology object**, see the Output Cams and Measuring Inputs Function Manual
- For **general information on cyclic measuring**, see the Output Cams and Measuring Inputs Function Manual

Overview of parameters for Activate measuring input

Table 7-220 Overview of parameters for Activate measuring input

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder for which the actual value is to be measured using a measuring input. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The measuring inputs associated with the selected axis or external encoder are automatically identified and displayed in the **Measuring input** field for selection, as appropriate. • **<Reference>** You select this entry if the measuring input is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type MeasuringInputType (references to measuring inputs) are available for selection in the **Measuring input** field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated measuring inputs. Instead, select the reference to the measuring input directly (variable of data type MeasuringInputType).
Measuring input	Here, the permissible measuring inputs are displayed according to the selected Axis/Encoder and are available for selection, as appropriate: • A position, following or path axis or external encoder defined on the device was selected as the axis/encoder: The measuring inputs associated with the selected axis/encoder are available for selection. If you wish to create a new measuring input, click the Create measuring input button and enter a new name. The measuring input is created for the technology object that has been selected in the Axis/Encoder field (not for variables). The configuration can be changed in the project navigator under the respective axis or external encoder in the MEASURING INPUTS folder. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: MeasuringInputType. These variables are references to measuring inputs.
Parameters tab	See Overview of parameters for Activate measuring input – Parameters tab (Page 4351)
Expert tab	See Overview of parameters for Activate measuring input – Expert tab (Page 4352)
Delay program execution	Only if the **Activate cyclic measuring** checkbox is not activated. • Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately. If the **Activate cyclic measuring** checkbox is activated, the next command is always executed immediately.

Overview of parameters for Activate measuring input – Parameters tab

Table 7-221 Overview of parameters for Activate measuring input – Parameters tab

Field/button	Meaning/instruction
Activate cyclic measuring (SIMOTION Kernel V4.0 and higher)	Activate this checkbox if the actual values are to be acquired cyclically. If the checkbox is not activated, the actual value is acquired one time.
Edge for cyclic measuring (SIMOTION Kernel V4.0 and higher)	Only if the **Activate cyclic measuring** checkbox is activated. Here, you select whether the actual position is to be measured at a rising or falling edge of the measuring input. **All edges** The actual position is measured at both edges of the measuring input. **Rising edges only** The actual position is measured only at a rising edge of the measuring input. **Falling edges only** The actual position is measured only at a falling edge of the measuring input. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.measuredEdgeCyclicMode
Edge	Only if the **Activate cyclic measuring** checkbox is not activated. Here, you select whether the actual position is to be measured at a rising or falling edge of the measuring input. **Falling edge** The actual position is measured at the falling edge of the measuring input. **Both** The actual position is measured at both edges of the measuring input. **Both (starting at rising edge)** The actual position is measured at both edges of the measuring input; the first measurement takes place at a rising edge. **Both (starting at falling edge)** The actual position is measured at both edges of the measuring input; the first measurement takes place at a falling edge. **Rising edge** The actual position is measured at the rising edge of the measuring input. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.measuredEdge

Field/button	Meaning/instruction
Activate measuring range	Select whether the measuring input is only to be active within a certain measuring range. **Measurement in specified range** Measured value acquisition only takes place in the range specified by the Start of measuring range and End of measuring range fields. **Measurement without specified range** Measured value acquisition takes place without range limitation; the Start of measuring range and End of measuring range fields are not visible. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.measuringRangeMode
Start of measuring range	Enter the starting point of the measuring range in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed** See Selection list (combo box) (Page 4022) System variable for default: userDefault.measuringRangeStartPosition
End of measuring range	Enter the end point of the measuring range in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** **Last programmed** See Selection list (combo box) (Page 4022) System variable for default: userDefault.measuringRangeEndPosition

Overview of parameters for Activate measuring input – Expert tab

Table 7-222 Overview of parameters for Activate measuring input – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Activate measuring input

Cam technology package

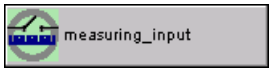
- _enableMeasuringInput, if the **Activate cyclic measuring** checkbox is not activated.
- _enableMeasuringInputCyclic, if the **Activate cyclic measuring** checkbox is activated.

Overview of parameters for Activate measuring input / _enableMeasuringInput, _enableMeasuringInputCyclic

Table 7-223 Parameters (MCC command Activate measuring input compared to system functions _enableMeasuringInput, _enableMeasuringInputCyclic)

Parameters of the MCC command Activate measuring input	**Parameters of the system functions _enableMeasuringInput, _enableMeasuringInputCyclic**
Axis/encoder	–
Measuring input	measuringInput
Parameters tab	
Activate cyclic measuring	Call of _enablemeasuringinput or _enablemeasuringinputcyclic system function
Edge for cyclic measuring	measuredEdgeMode
Activate measuring range	measuringRangeMode
Start of measuring range	measuringRangeStartPositionType, measuringRangeStartPosition
End of measuring range	measuringRangeEndPositionType, measuringRangeEndPosition
Expert tab	
CommandID variable	commandId
Return variable	–

Deactivate measuring input



This command disables a measuring input on an axis or external encoder. Measured value acquisition is disabled.

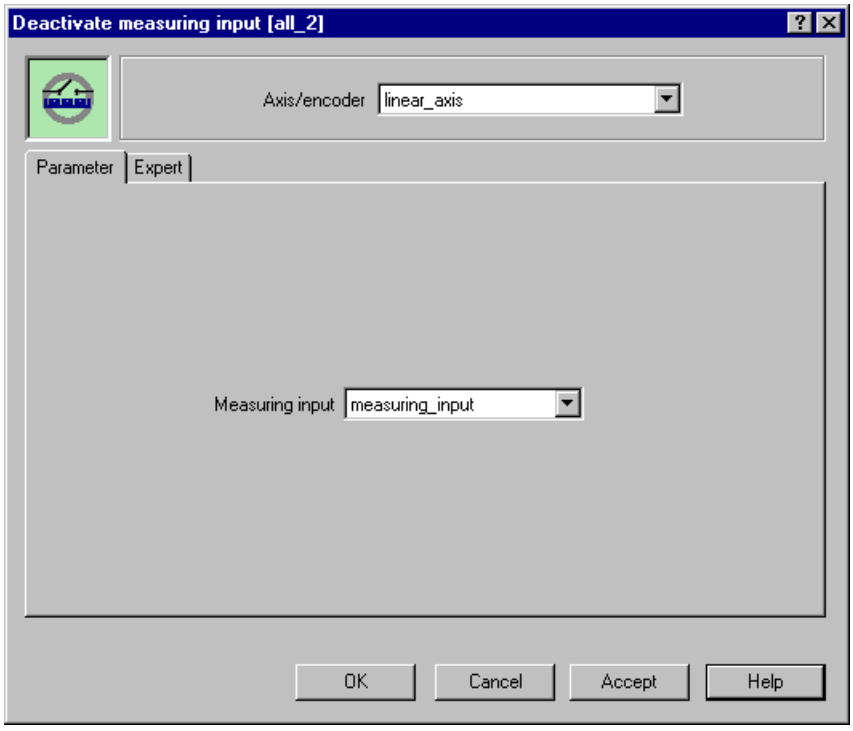


Figure 7-145 Parameter screen form: Deactivate measuring input

Further information:

- For information on the **Measuring Input technology object**, see the Output Cams and Measuring Inputs Function Manual
- For information on **commands on the Measuring Input technology object**, see the Output Cams and Measuring Inputs Function Manual
- For **general information on cyclic measuring**, see the Output Cams and Measuring Inputs Function Manual

Overview of parameters for Deactivate measuring input

Table 7-224 Overview of parameters for Deactivate measuring input

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder for which the actual value acquisition via a measuring input is to be ended. The list contains: All position, synchronized, and path axes and external encoders that are defined on the relevant device. The measuring inputs associated with the selected axis or external encoder are automatically identified and displayed in the **Measuring input** field for selection, as appropriate. **<Reference>** You select this entry if the measuring input is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type MeasuringInputType (references to measuring inputs) are available for selection in the **Measuring input** field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated measuring inputs. Instead, select the reference to the measuring input directly (variable of data type MeasuringInputType).
Parameters tab	See Overview of parameters for Deactivate measuring input – Parameters tab (Page 4355)
Expert tab	See Overview of parameters for Deactivate measuring input – Expert tab (Page 4356)

Overview of parameters for Deactivate measuring input – Parameters tab

Table 7-225 Overview of parameters for Deactivate measuring input – Parameters tab

Field/button	Meaning/instruction
Measuring input	Here, the permissible measuring inputs are displayed according to the selected **Axis/Encoder**, where they are available for selection, as appropriate: A positioning or synchronous axis or external encoder defined on the the device was selected as the axis/encoder: The measuring inputs associated with the selected axis/encoder are available for selection. <Reference> was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: MeasuringInputType. These variables are references to measuring inputs.

Overview of parameters for Deactivate measuring input – Expert tab

Table 7-226 Overview of parameters for Deactivate measuring input – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Deactivate measuring input

Cam technology package:

- _disableMeasuringInput

Overview of parameters for Deactivate measuring input / _disableMeasuringInput

Table 7-227 Parameters (MCC command Deactivate measuring input compared to system function _disableMeasuringInput)

Parameters of the MCC command Deactivate measuring input	Parameters of the system function _disableMeasuringInput
Axis/encoder	externalEncoder
Parameters tab	
Measuring input	referenceencodertype, referenceEncoder, encoder
Expert tab	
Return variable	–

Synchronize measuring system

This command synchronizes two measuring systems or returns the difference between the specified measuring systems.

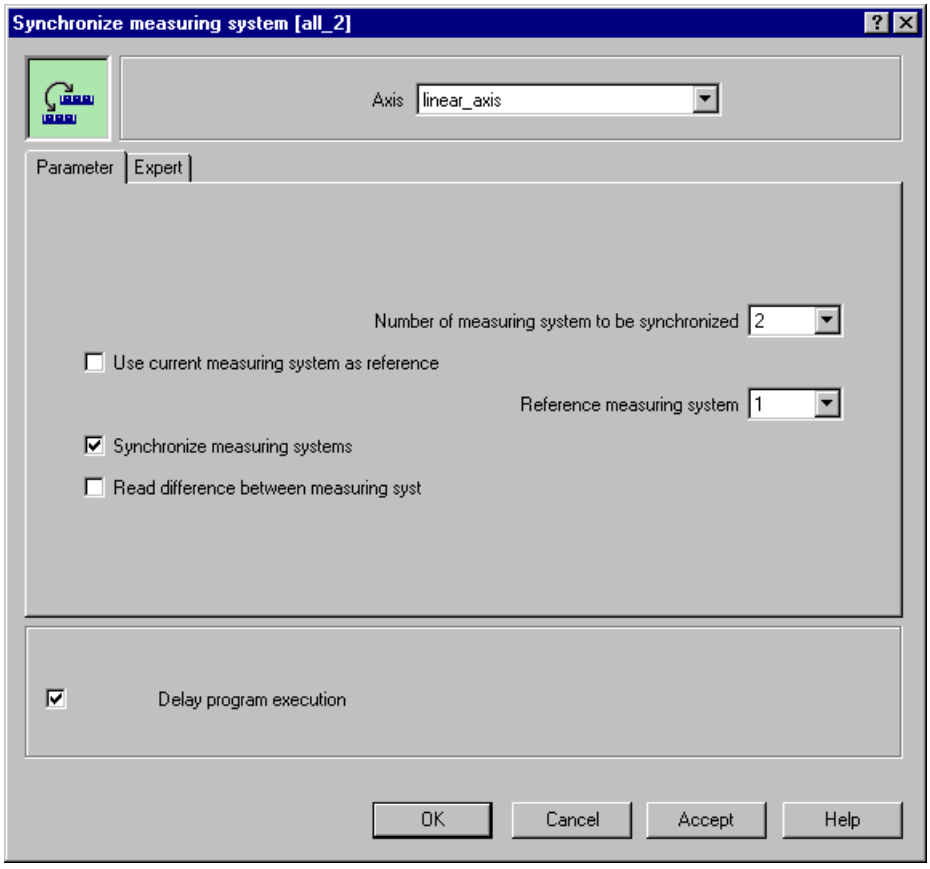


Figure 7-146 Parameter screen form: Synchronize measuring system

Overview of parameters for Synchronize measuring system

Table 7-228 Overview of parameters for Synchronize measuring system

Field/button	Meaning/instruction
Axis	Select the axis whose measuring system is to be switched over. The list contains: • All position, synchronized and path axes that are defined on the relevant device. The axis is defined in the AXES folder in the project navigator. • All technology object data type variables declared in the MCC unit or MCC chart, see Description of technology object data types (Page 4060): PosAxis, FollowingAxis or _PathAxis.
Parameters tab	See Overview of parameters for Synchronize measuring system – Parameters tab (Page 4358)
Expert tab	See Overview of parameters for Synchronize measuring system – Expert tab (Page 4358)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Synchronize measuring system – Parameters tab

Table 7-229 Overview of parameters for Synchronize measuring system – Parameters tab

Field/button	Meaning/instruction
Number of the measuring system to be synchronized	Measuring system to be selected; possible values are 1 to 8. **1** (default value) Measuring system 1
Use current measuring system as reference	Activate this checkbox if the current measuring system is to be used as a reference measuring system.
Reference measuring system	Here, you select the reference measuring system if you do not wish to use the current measuring system. You can enter a value between 1 and 8. This field is grayed-out by default; it is activated only if you deactivate the checkbox **Use current measuring system as reference**. **1** (default value) Measuring system 1
Synchronize measuring systems	Activate this checkbox if the measuring systems are to be synchronized with one another.
Read difference between measuring systems	Activate this checkbox if you want the difference between the two measuring systems to be stored in a variable.
Differential variable	In this field, a variable of type LREAL can be specified to contain the difference between measuring systems once the command is executed. This field is only visible, if the "Read difference between measuring systems" checkbox is activated.

Overview of parameters for Synchronize measuring system – Expert tab

Table 7-230 Overview of parameters for Synchronize measuring system – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call and to manage the received data. Data type DINT, for a description see table below.

Table 7-231 Structure of the return value (TYPE StructRetEncoderValue)

Parameter / data type	Meaning/values	Values
functionResult (DINT)	Error number	0: if command execution is okay <> 0: if an error has occurred See Return value for the system functions of the technology packages (Page 4040).
encoderDifference (LREAL)	Difference of specified measuring systems	

Relevant system function for Synchronize measuring system

Cam technology package:

- _setAndGetEncoderValue

Based on the **Read difference between measuring systems** checkbox, the value from _MccRetStructRetEncoderValue.encoderDifference is copied to the differential variable.

Overview of parameters for Synchronize measuring system / _setAndGetEncoderValue

Table 7-232 Parameters (MCC command Synchronize measuring system compared to system function _setAndGetEncoderValue)

Parameters of the MCC command Synchronize measuring system	**Parameters of the system function _setAndGetEncoderValue**
Axis	axis
Delay program execution	nextCommand
Parameters tab	
Number of the measuring system to be synchronized	encoder
Use current measuring system as reference	referenceEncoderType
Reference measuring system	referenceEncoderType, referenceEncoder
Synchronize measuring systems	mode
Read difference between measuring systems	–
Differential variable	–
Expert tab	
CommandID variable	commandId
Return variable	–

Switch output cam on

This command can be used to assign parameters for an output cam and then to switch it on.

An output cam generates position-dependent switching signals for position or synchronous axes or external encoders. The output cam types below are available. You specify the type during configuration of the output cam:

- Position-based cam
  The switching signal occurs between the starting position and end position of the output cam.
- Time-based cam
  The switching signal is supplied for a specified time period after the starting position is reached.
- Uni-directional output cam
  The switching signal occurs when the axis reaches the starting position; it can only be reset by switching off the output cam.

Position-based and time-based cams can also be assigned as counter cams.

The switching signal is available in system variable *state* and, if configured accordingly, also on a digital output.

The output cam can be switched off:

- With the commands
  - "Switch output cam off" (Page 4368) or
  - "Switch output cam signal" (Page 4371),
- By switching on the same output cam again with different parameters, if necessary

The parameter dialog box displays different parameters depending on the axis type and the configured output cam type.

Below are the parameter dialog boxes for a rotary axis and a linear axis showing the settings for different output cam types.

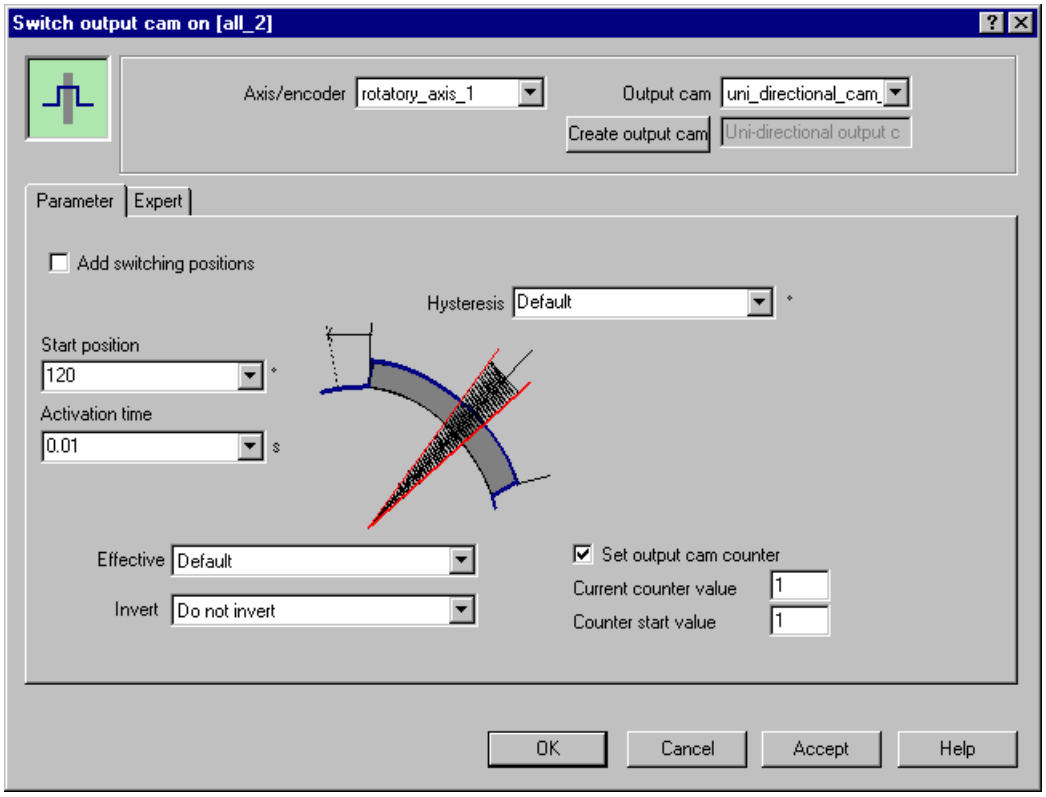


Figure 7-147 Parameter screen form: "Switch output cam on" for rotary axis with uni-directional output cam

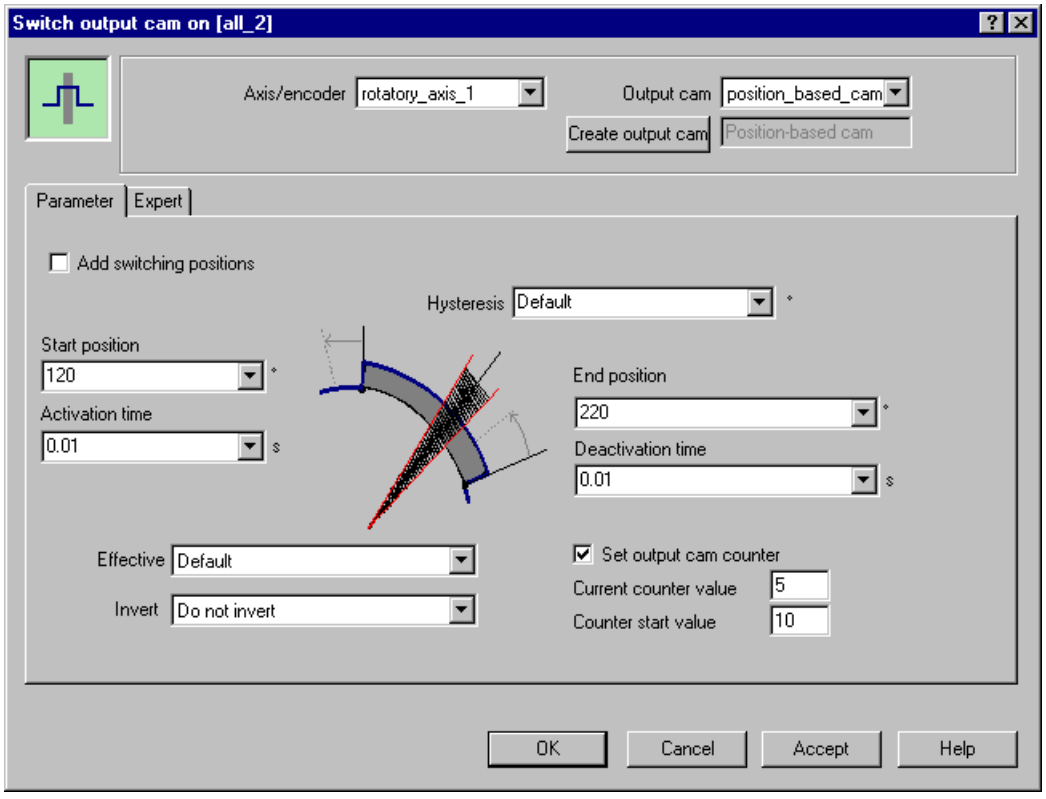


Figure 7-148 Parameter screen form: "Switch output cam on" for rotary axis with position-based cam

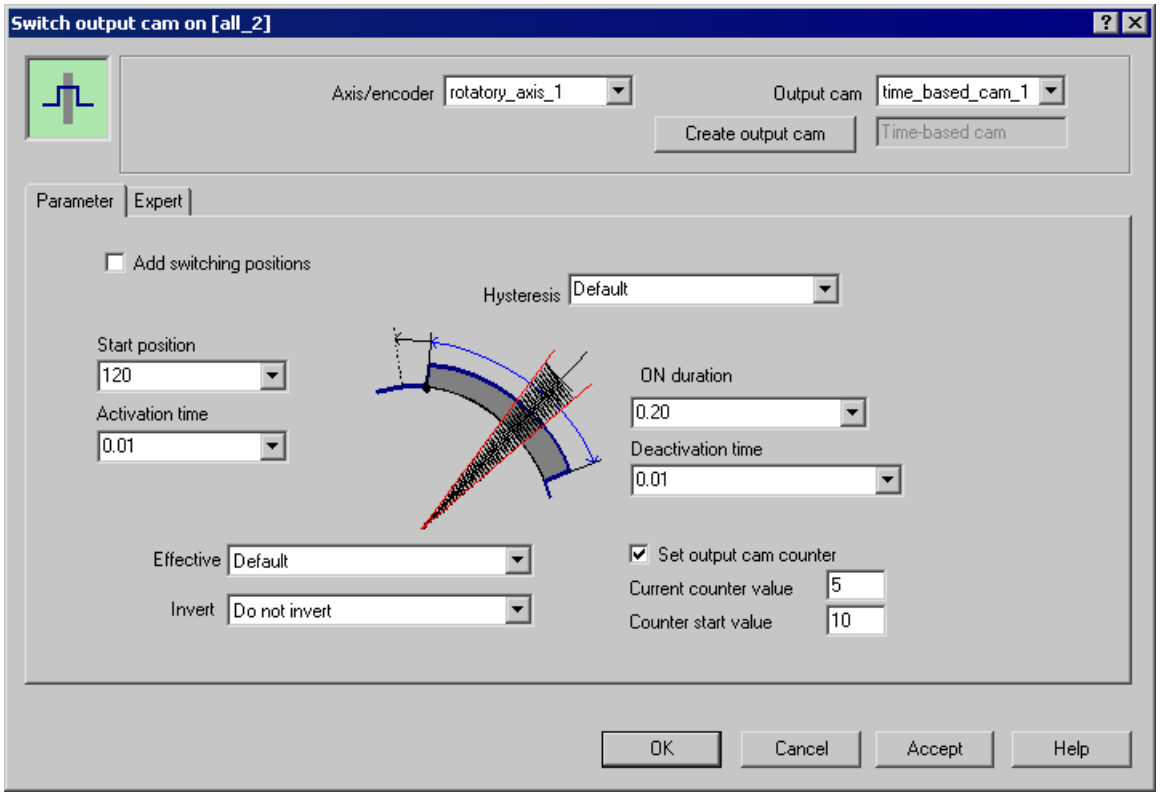


Figure 7-149 Parameter screen form: "Switch output cam on" for rotary axis with time-based cam

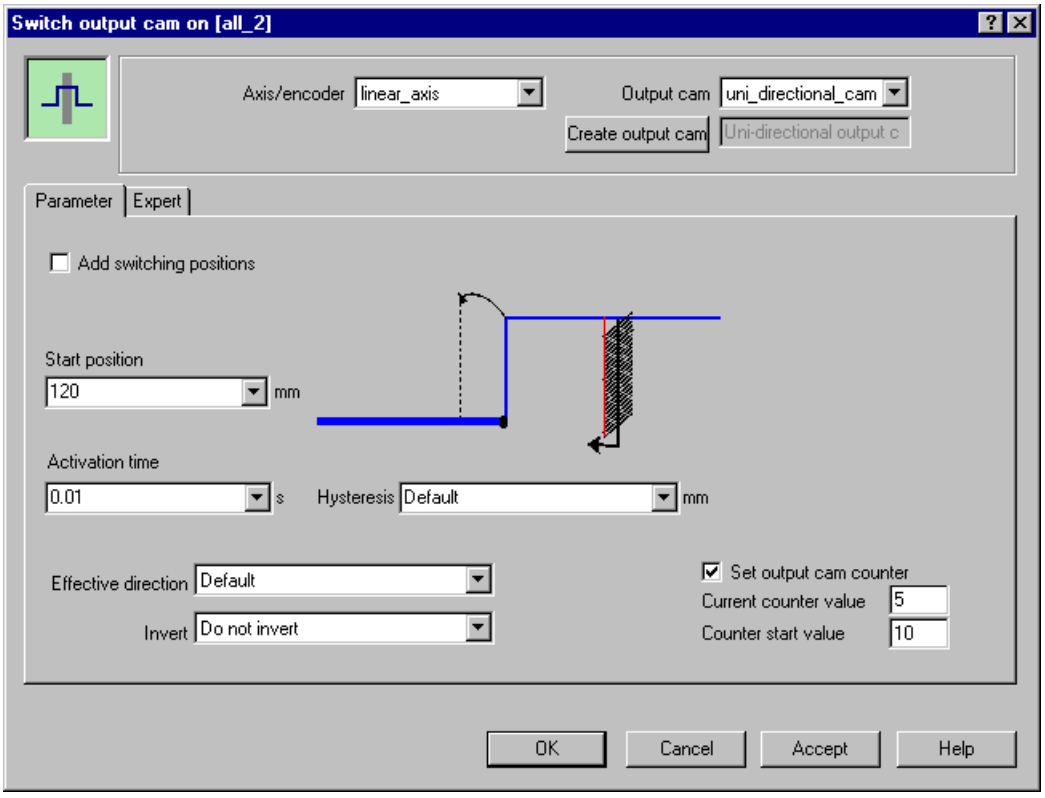


Figure 7-150 Parameter screen form: "Switch output cam on" for linear axis with uni-directional output cam

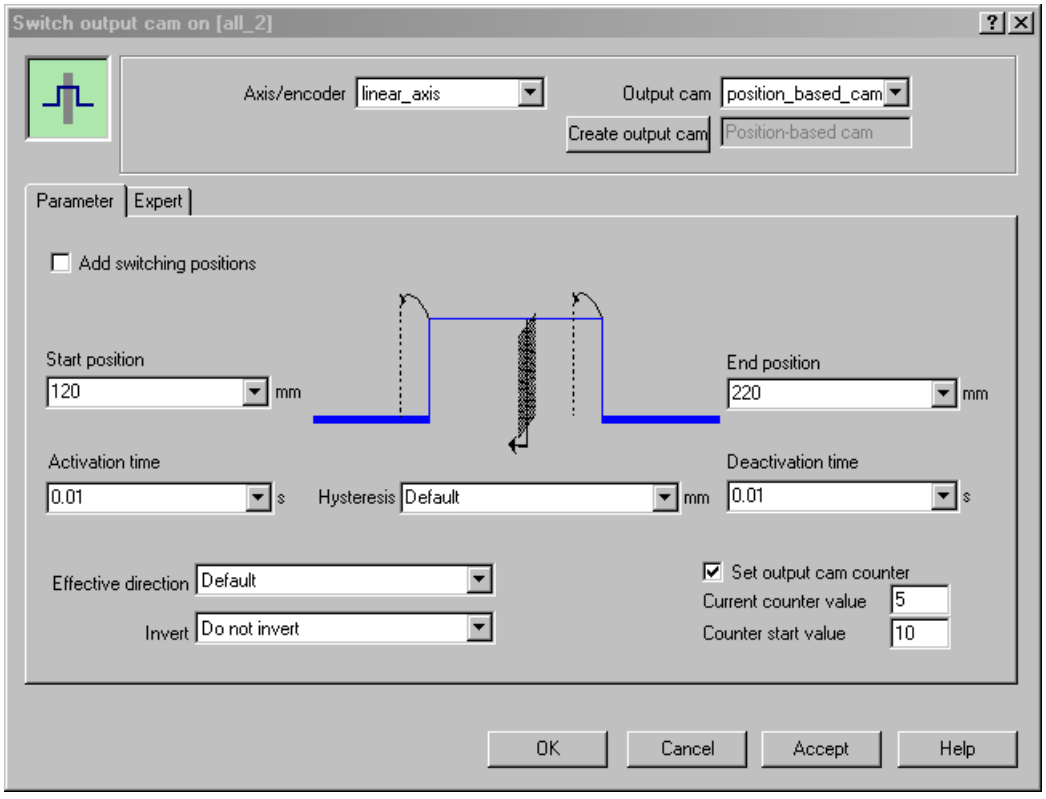


Figure 7-151 Parameter screen form: "Switch output cam on" for linear axis with position-based cam

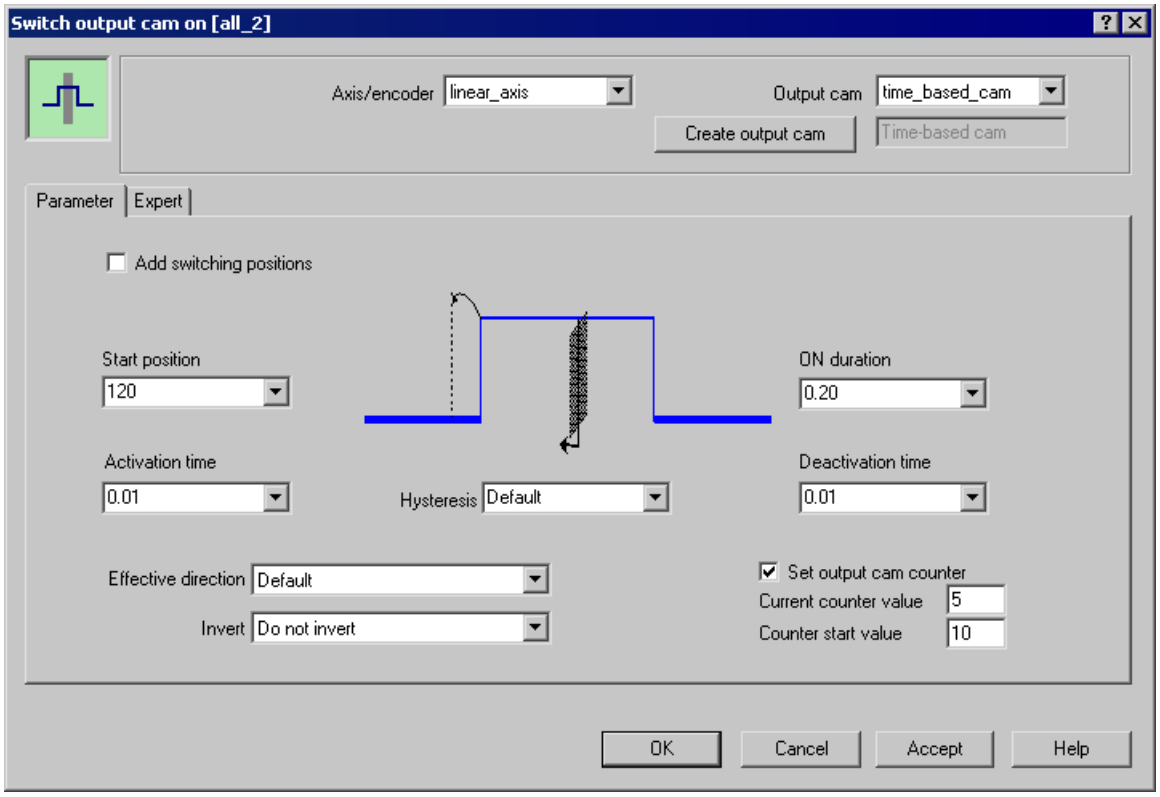


Figure 7-152 Parameter screen form: "Switch output cam on" for linear axis with time-based cam

For additional information on the **Output Cam technology object**, see the Output Cams and Measuring Inputs Function Manual.

For additional information on the **commands on the Output Cam technology object**, see the Output Cams and Measuring Inputs Function Manual.

Overview of parameters for Switch output cam on

Table 7-233 Overview of parameters for Switch output cam on

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder with which the output cam to be switched on is associated. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The output cams associated with the selected axis or external encoder are automatically identified and displayed in the Output cam field for selection, as appropriate. • **<Reference>** You select this entry if the output cam is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type OutputCamType (references to output cams) are available for selection in the Output cam field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated output cams. Instead, select the reference to the output cam directly (variable of data type OutputCamType).
Output cam	Here, the permissible output cams are displayed according to the selected Axis/Encoder and are available for selection, as appropriate: • A position, synchronized, or path axis or external encoder defined on the device was selected as the axis/encoder: The output cams associated with the selected axis/encoder are available for selection. If you wish to create a new output cam, click the Create output cam button and enter a new name. The output cam is created for the technology object that has been selected in the Axis/Encoder field (not for variables). The configuration can be changed in the project navigator under the respective axis or external encoder in the OUTPUT CAMS folder. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: OutputCamType. These variables are references to output cams.
Parameters tab	See Overview of parameters for Switch output cam on - Parameters tab (Page 4365)
Expert tab	See Overview of parameters for Switch output cam on - Expert tab (Page 4367)

Overview of parameters for Switch output cam on – Parameters tab

Table 7-234 Overview of parameters for Switch output cam on – Parameters tab

Field/button	Meaning/instruction
Starting position	With position-based cams: switching signal is activated between the starting position and end position. With time-based cams and unidirectional output cams: Position at which the switching signal is activated. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). The meaning is dependent on the **Add switching positions** checkbox. • Checkbox inactive: The entered value represents an absolute position. • Checkbox active: The entered value is relative and is added to the last programmed starting position. **Last programmed position** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.switchOnPosition
End position	For position-based cams only: switching signal is activated between the starting position and end position. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). The meaning is dependent on the **Add switching positions** checkbox. • Checkbox inactive: The entered value represents an absolute position. • Checkbox active: The entered value is relative and is added to the last programmed end position. **Last programmed position** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.switchOffPosition
Add switching positions	Activate the checkbox, if you want the entered values for the start position and the end position to be regarded as relative positions. The values are then added to the last programmed position each time. If this checkbox is deactivated, the entered values for the start position and the end position are regarded as absolute positions.
ON duration	ON duration for time-based cams. Once the axis has crossed the starting position, the switching signal of a time-based cam remains enabled for the programmed ON duration. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed ON duration** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.switchOnDuration

Field/button	Meaning/instruction
Activation time	The activation time shifts the switch-on instant of the output cam. Method of operation: • Negative value: Output cam activation is moved up. For example, this allows you to compensate for delay times of digital outputs and connected switching elements. • Positive value: Output cam activation is delayed. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.activationTime.
Deactivation time	The deactivation time shifts the switch-off instant of the output cam. Method of operation: • Negative value: Output cam deactivation is moved up. For example, this allows you to compensate for delay times of digital outputs and connected switching elements. • Positive value: Output cam deactivation is delayed. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.deactivationTime.
Hysteresis	The hysteresis defines a filter range around the switching position. The output signal is not switched if the axis moves in the hysteresis range around the switching position. This avoids flickering switching status changes. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed hysteresis** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.noSwitchingRange
Effective direction	Here, select the current effective direction. The output cam only switches when the direction of motion and effective direction are identical. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Both directions** The output cam switches in both axis directions. **Positive** The output cam switches when axis direction is positive. **Negative** The output cam switches when axis direction is negative. **Last programmed effective direction** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.forceDirection.

Field/button	Meaning/instruction
Invert	Here, you choose whether or not the switching state of the output of the output cam is to be inverted. **Do not invert** (default value) The output of the output cam is not inverted **Invert** The output of the output cam is inverted **Maintain current switching state** The current setting for inverting the output of the output cam is kept.
Set output cam counter	Activate this checkbox if the output cam is to be configured as a counter cam. If the checkbox is deactivated, the status of the counter cam remains unchanged. For a counter cam, it can be specified whether the output cam is to be output every time it switches or every nth time it switches. Every counter cam has a counter start value and a current counter value. The current count value for the output cam is reduced by 1 every time the output cam switches: • If the current count value reaches 0, the output cam is output (*state* system variable and output cam output). At the same time, the current counter value is reset to the counter start value. • If the current counter value does not reach 0, the output of the output cam is suppressed. The current counter value and the counter start value both have a preassigned value of 1; the current values can be queried with system variables counterCamData.actualValue and counterCamData.startValue. These values are not reset by the system, e.g. with the **Switch output cam off** command.
Current counter value	Only if the **Set output cam counter** checkbox is activated: Here, you enter the current counter value as an integer. See also input field (Page 4022).
Counter start value	Only if the **Set output cam counter** checkbox is activated: Here, you enter the counter start value as an integer. See also input field (Page 4022).

Overview of parameters for Switch output cam on – Expert tab

Table 7-235 Overview of parameters for Switch output cam on – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
TO properties	Here, you can adapt the parameter screen form as needed to reflect the effects of axis/encoder or software output cam configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Switch output cam on

Cam technology package:

- _enableOutputCam
- _setOutputCamCounter (if assigned as a counter cam)

Based on the **Set output cam counter** checkbox, the _setOutputCamCounter function is called first.

Overview of parameters for Switch output cam on / _enableOutputCam, _setOutputCamCounter

Table 7-236 Parameters (MCC command Switch output cam on compared to system functions _enableOutputCam and _setOutputCamCounter)

Parameters of the MCC command Switch output cam on	**Parameters of the system functions _enableOutputCam and _setOutputCamCounter**
Axis/encoder	–
Output cam	outputCam
Parameters tab	
Starting position	switchOnPosition
End position	switchOffValue
Add switching positions	switchOnPositionType, switchOffValueType
ON duration	switchOffValue
Activation time	activationTimeType, activationTime
Deactivation time	deactivationTimeType, deactivationTime
Hysteresis	noSwitchingRangeType, noSwitchingRange
Effective direction	forceDirection
Invert	invertOutput
Set output cam counter	Call of _setOutputCamCounter system function
Current counter value	actualValue
Counter start value	startValueMode, startValue
Expert tab	
Return variable	–

Switch output cam off

This command is used to switch off an output cam that was switched on with the "Switch output cam on" (Page 4359) command.

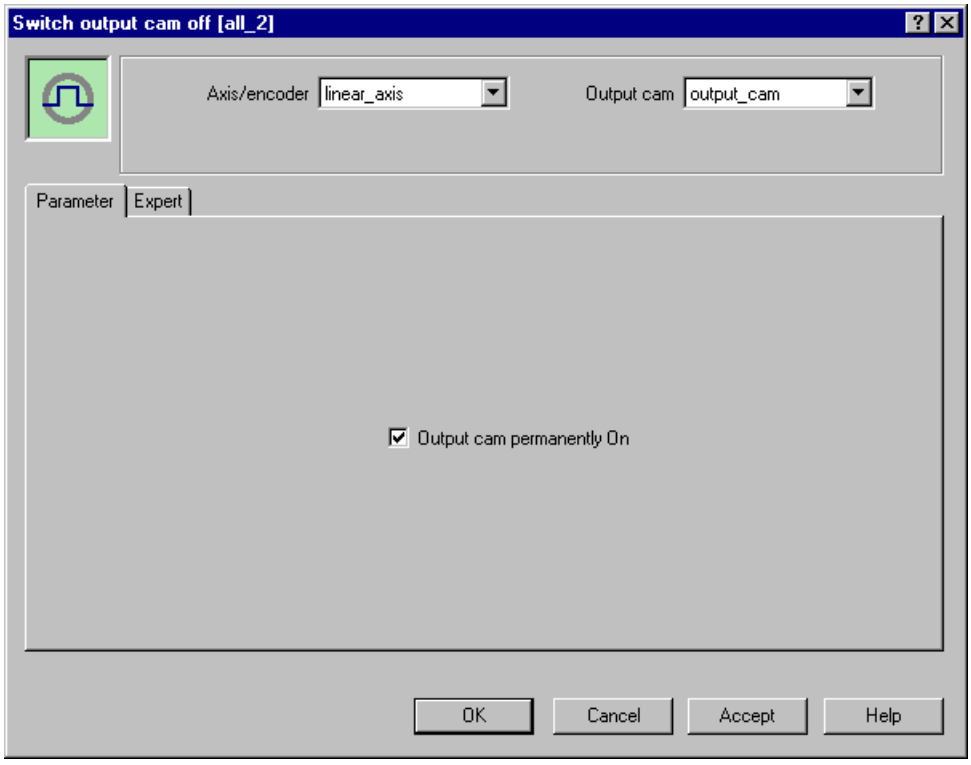


Figure 7-153 Parameter screen form: Switch output cam off

For additional information on the **Output Cam technology object**, see the Output Cams and Measuring Inputs Function Manual.

For additional information on the **commands on the Output Cam technology object**, see the Output Cams and Measuring Inputs Function Manual.

Overview of parameters for Switch output cam off

Table 7-237 Overview of parameters for Switch output cam off

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder to which the output cam to be deactivated is associated. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The output cams associated with the selected axis or external encoder are automatically identified and displayed in the Output cam field for selection, as appropriate. • **<Reference>** You select this entry if the output cam is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type OutputCamType (references to output cams) are available for selection in the Output cam field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated output cams. Instead, select the reference to the output cam directly (variable of data type OutputCamType).
Output cam	Here, the permissible output cams are displayed according to the selected Axis/Encoder and are available for selection, as appropriate: • A position, synchronized, or path axis or external encoder defined on the device was selected as the axis/encoder: The output cams associated with the selected axis/encoder are available for selection. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Description of the technology object data types (Page 4060)) are available for selection: OutputCamType. These variables are references to output cams.
Parameters tab	See Overview of parameters for Switch output cam off - Parameters tab (Page 4370)
Expert tab	See Overview of parameters for Switch output cam off - Expert tab (Page 4371)

Overview of parameters for Switch output cam off – Parameters tab

Table 7-238 Overview of parameters for Switch output cam off – Parameters tab

Field/button	Meaning/instruction
Output cam permanently On	Select this checkbox if you wish the output cam to remain ON permanently after it is disabled. If the checkbox is not activated, the output of the output cam is reset when you disable the output cam.

Overview of parameters for Switch output cam off – Expert tab

Table 7-239 Overview of parameters for Switch output cam off – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Switch output cam off

Cam technology package:

- _disableOutputCam, if the **Output cam permanently On** checkbox is not activated.
- _setOutputCamState, if the **Output cam permanently On** checkbox is activated.

Overview of parameters for Switch output cam off / _disableOutputCam, _setOutputCamState

Table 7-240 Parameters (MCC command Switch output cam off compared to system functions _disableOutputCam and _setOutputCamState)

Parameters of the MCC command Switch output cam off	Parameters of the system functions _disableOutputCam and _setOutputCamState
Axis/encoder	–
Output cam	outputCam
Parameters tab	
Output cam permanently On	Call of _disableOutputCam or _setOutputCamState system function
Expert tab	
Return variable	–

Switch output cam signal

This function is available in SIMOTION Kernel as of Version 4.1.

This command allows you to switch an output cam and set its switching signal to the specified value. This output cam signal is available in the *state* system variable and, if configured accordingly, is also available at a digital output. This terminates any current output cam processing that was started, for example, with the "Switch output cam on" (Page 4359) command.

If the output cam has been configured as a high-speed output cam on the relevant devices (e.g. SIMOTION D, C240, TM15, TM17 High Feature), the following also applies:

- You can switch the assigned digital output at specific times during the configured processing cycle clock of the output cam.
- The time offset specified in the command is added to the system-dependent output delay of the output cam signal (tOutput system variable). The resulting time offset (programmed time offset + value of tOutput system variable) must be smaller than the duration of the processing cycle clock.

Note

During a processing cycle clock, the output cam signal can only be switched on and switched off once. If an attempt is made to switch the output cam on or off more than once during a processing cycle clock, the values of the last effective command apply.

If the output cam is not configured as a high-speed output cam (output cam without time stamp, e.g. digital outputs on the SIMOTION C230-2 device), a programmed time offset is ignored. The tOutput system variable is assigned the value 0.0.

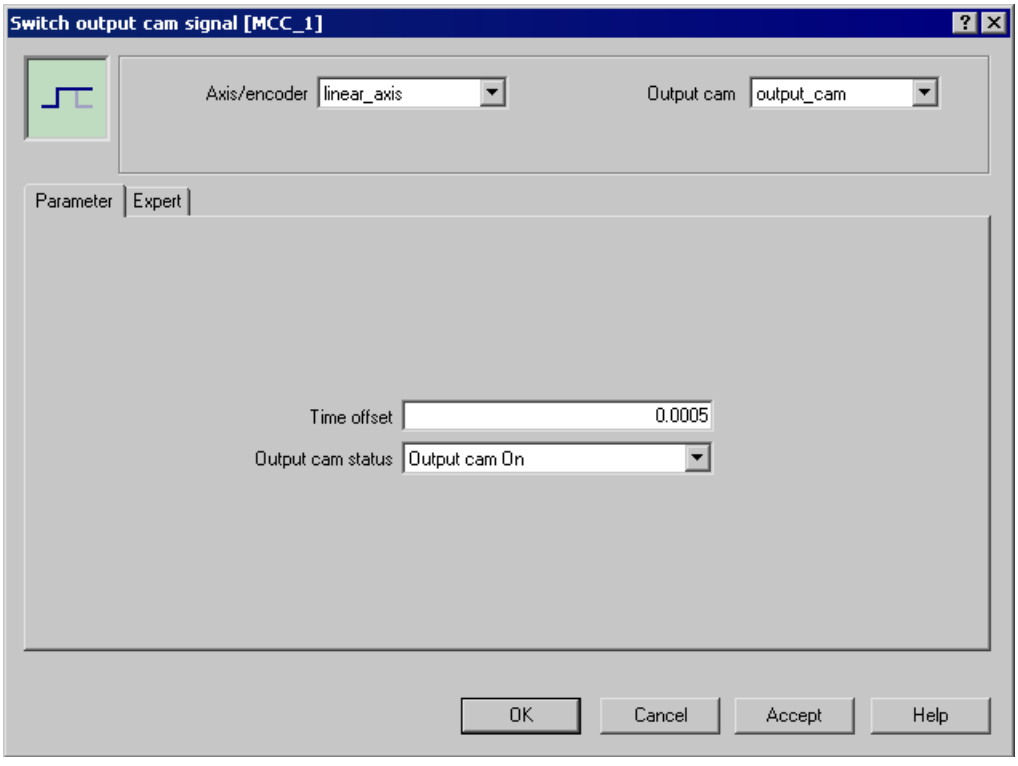


Figure 7-154 Parameter screen form: Switch output cam signal

For additional information on the **Output Cam technology object**, see the Output Cams and Measuring Inputs Function Manual.

For additional information about **exact time setting of an output and exact time output cams**, see the Output Cams and Measuring Inputs Function Manual.

Overview of parameters for Switch output cam signal

Table 7-241 Overview of parameters for Switch output cam signal

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder to which the output cam to be switched on is associated. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The output cams associated with the selected axis or external encoder are automatically identified and displayed in the Output cam field for selection, as appropriate. • **<Reference>** You select this entry if the output cam is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type OutputCamType (references to output cams) are available for selection in the Output cam field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated output cams. Instead, select the reference to the output cam directly (variable of data type OutputCamType).
Output cam	Here, the permissible output cams are displayed according to the selected Axis/Encoder and are available for selection, as appropriate: • A position, synchronized, or path axis or external encoder defined on the device was selected as the axis/encoder: The output cams associated with the selected axis/encoder are available for selection. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: OutputCamType. These variables are references to output cams.
Parameters tab	See Overview of parameters for Switch output cam signal - Parameters tab (Page 4374)
Expert tab	See Overview of parameters for Switch output cam signal - Expert tab (Page 4374)

Overview of parameters for Switch output cam signal – Parameters tab

Table 7-242 Overview of parameters for Switch output cam off – Parameters tab

Field/button	Meaning/instruction
Time offset	This parameter is only evaluated if the output cam was configured as a high-speed output cam on the corresponding hardware (e.g. SIMOTION D, C240, TM15, and TM17 High Feature SIMOTION devices). Here, you specify the time offset with which you can switch the assigned digital output at specific times during the configured processing cycle clock of the output cam. The specified time offset is added to the system-dependent output delay of the output cam signal (tOutput system variable). The resulting time offset (programmed time offset + value of tOutput system variable) must be smaller than the duration of the processing cycle clock. Enter the value as a signed floating-point number. See the input field (Page 4022). **Note** During a processing cycle clock, the output cam signal can only be switched on once and/or switched off once. If an attempt is made to switch the output cam on or off more than once during a processing cycle clock, the values of the last effective command apply.
Output cam status	You select the output cam signal (switching signal of the output cam) here. **Output cam on** The output cam signal is switched on. **Output cam off** The output cam signal is switched off.

Overview of parameters for Switch output cam signal – Expert tab

Table 7-243 Overview of parameters for Switch output cam signal – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Switch output cam signal

Cam technology package:

- _setOutputCamState

Overview of parameters for Switch output cam signal / _setOutputCamState

Table 7-244 Parameters (MCC command Switch output cam signal compared to system function _setOutputCamState)

Parameters of the MCC command Switch output cam signal	Parameters of the system function _setOutputCamState
Axis/encoder	–
Output cam	outputCam
Parameters tab	
Time offset	timeOffset
Output cam status	stateType
Expert tab	
Return variable	–

Output cam track On

This command is used to assign parameters for a cam track and then to switch it on.

A cam track provides the switching signal of multiple output cams of the same type (single output cams) in system variable *state*. If configured accordingly, the switching signal is also available on a digital output.

The output cam types below are available for single output cams. You specify the type during configuration of the cam track:

- Position-based cam
  The switching signal occurs between the starting position and end position of a single output cam.
- Time-based cam
  The switching signal is supplied for a specified time period after the starting position of a single cam is reached.
- Time-based cam with maximum ON length
  The switching signal is supplied for a specified time period after the starting position of a single cam is reached.
  A maximum ON length can also be defined. It limits the duration of the switching signal to the time it takes to cover this distance

The cam track can be switched off:

- With the "Cam track off" (Page 4381) command
- By switching on the same cam track again with different parameters, if necessary

Different screen forms are displayed depending on the axis type.

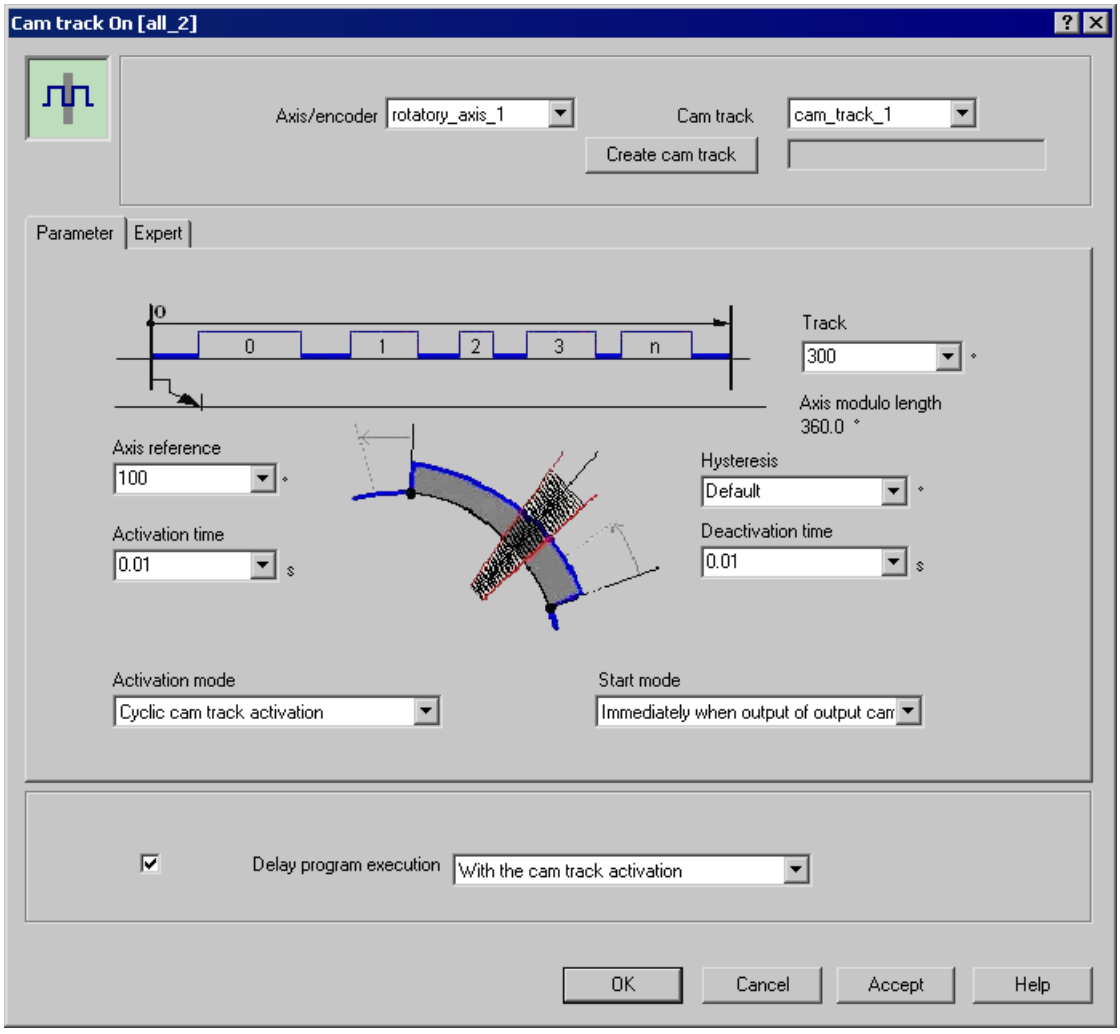


Figure 7-155 Parameter screen form: Cam track on for a rotary axis

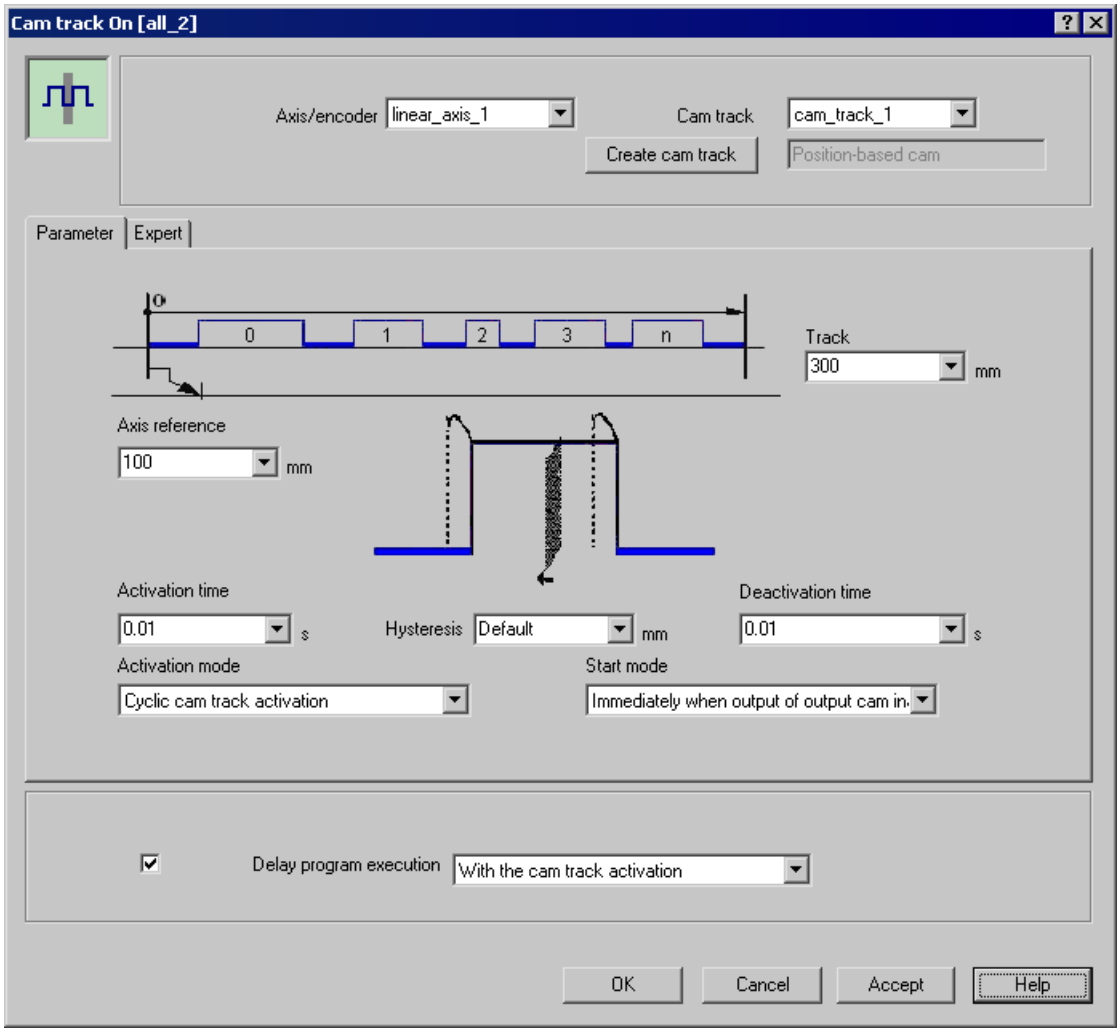


Figure 7-156 Parameter screen form: Cam track on for a linear axis

For additional information on the **Cam track technology object**, see the Output Cams and Measuring Inputs Function Manual.

For additional information on the **commands on the Cam Track technology object**, see the Output Cams and Measuring Inputs Function Manual.

Overview of parameters for Output cam track On

Table 7-245 Overview of parameters for Cam track on

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder to which the cam track to be activated is associated. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The cam tracks associated with the selected axis or external encoder are automatically identified and displayed in the Cam track field for selection as appropriate. • **<Reference>** You select this entry if the cam track is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type _CamTrackType (references to cam tracks) are available for selection in the Cam track field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated cam tracks. Instead, select the reference to the cam track directly (variable of data type _CamTrackType).
Cam track	Here, the permissible cam tracks are displayed according to the selected Axis/Encoder and are available for selection, as appropriate: • A position, following or path axis or external encoder defined on the device was selected as the axis/encoder: The cam tracks associated with the selected axis/encoder are available for selection. If you wish to create a new cam track, click the Create cam track button and enter a new name. The cam track is created for the technology object that has been selected in the Axis/Encoder field (not for variables). The configuration can be changed in the project navigator under the respective axis or external encoder in the OUTPUT CAMS folder. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: _CamTrackType. These variables are references to cam tracks.
Parameters tab	See Overview of parameters for Cam track on - Parameters tab (Page 4379)
Expert tab	See Overview of parameters for Cam track on - Expert tab (Page 4380)
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. **At cam track end** The next command is executed if the cam track processing is finished. **When cam track is activated** The next command is executed as soon as the cam track is activated. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Output cam track On – Parameters tab

Table 7-246 Overview of parameters for Cam track on – Parameters tab

Field/button	Meaning/instruction
Track length	Here, you define the length of the cam track. The individual output cams are mapped onto the cam track by converting their position values to modulo values with respect to the track length. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.camTrackLength.
Axis reference position	Here, you specify how the cam track is mapped onto the axis, i.e. the axis position starting from which the cam track is to be output. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.axisReferencePosition.
Activation time	The activation time shifts the switch-on instant of the single output cam. Method of operation: Negative value: Single output cam activation is moved up. For example, this allows you to compensate for delay times of digital outputs and connected switching elements. Positive value: Single output cam activation is delayed. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.activationTime
Deactivation time	The deactivation time shifts the switch-off instant of the single output cam. Method of operation: • Negative value: Single output cam deactivation is moved up. For example, this allows you to compensate for delay times of digital outputs and connected switching elements. • Positive value: Single output cam deactivation is delayed. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.deactivationTime.

Field/button	Meaning/instruction
Hysteresis	The hysteresis defines a filter range around the switching position. The output signal is not switched if the axis moves in the hysteresis range around the switching position. This avoids flickering switching status changes. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.hysteresisRange.
Activation mode	**Non-cyclical cam track activation** The cam track is mapped starting from the axis reference position, output once, and terminated automatically after it is exited. **Cyclical cam track activation** The track length of the cam track starting from the axis reference position is mapped and cyclically continued/repeated. **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.activationMode.
Start mode	The start mode is used to define when the cam track should become effective after activation or the process for switching between cam tracks. **With next track cycle** The cam track processing takes effect at the next cam cycle, i.e. as soon as the first single output cam switches the new track. Up to that point, a time-based cam of the previous cam track will be output. **Immediately** The cam track processing takes effect immediately. If an cam track is already active, it is aborted. **Immediately when output cam output is inactive** The cam track processing takes effect as soon as there are no more active single output cams of the previous cam track. An active single output cam of the previous cam track will be output completely. **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.startMode.

Overview of parameters for Output cam track On – Expert tab

Table 7-247 Overview of parameters for Cam track on – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.

Field/button	Meaning/instruction
TO properties	Here, you can adapt the parameter dialog box as needed to reflect the effects of axis/encoder or cam track configuration data or system variables.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Output cam track On

Cam technology package:

- _enableCamTrack

Overview of parameters for Cam track on / _enableCamTrack

Table 7-248 Parameters (MCC command Cam track on compared to system function _enableCamTrack)

Parameters of the MCC command Cam track on	Parameters of the system function _enableCamTrack
Axis/encoder	–
Cam track	camtrack
Delay program execution	nextCommand
Parameters tab	
Track length	camtrackLengthType, camtrackLength
Axis reference position	axisReferencePosition
Activation time	activationTimeType, activationTime
Deactivation time	deactivationTimeType, deactivationTime
Hysteresis	hysteresisRangeType, hysteresisRange
Activation mode	activationMode
Start mode	startMode
Expert tab	
CommandID variable	commandId
TO properties	–
Return variable	–

Output cam track Off

This command is used to switch off a cam track that was switched on with the "Cam track on" (Page 4375) command.

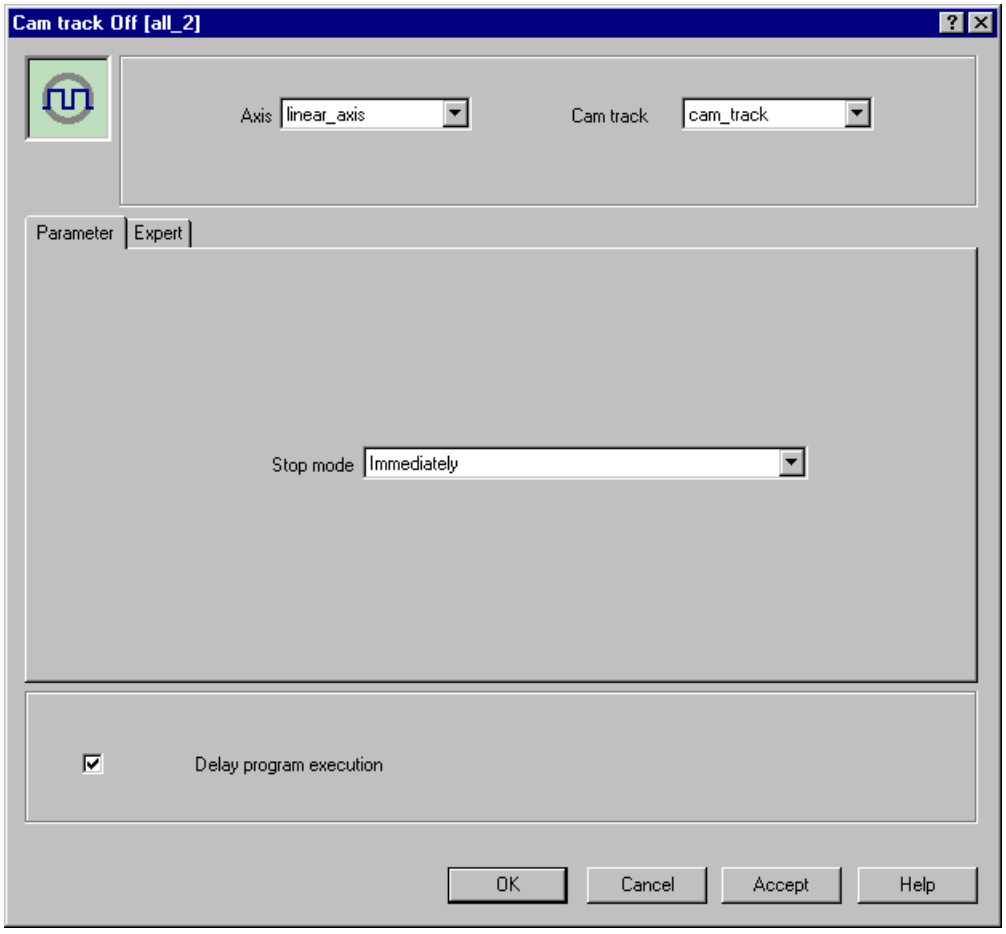


Figure 7-157 Parameter screen form: Cam track off

For additional information on the **Cam track technology object**, see the Output Cams and Measuring Inputs Function Manual.

For additional information on the **commands on the Cam Track technology object**, see the Output Cams and Measuring Inputs Function Manual.

Overview of parameters for Output cam track Off

Table 7-249 Overview of parameters for Cam track off

Field/button	Meaning/instruction
Axis/encoder	Here, you select the axis or external encoder to which the cam track to be deactivated is associated. The list contains: • All position, synchronized, and path axes and external encoders that are defined on the relevant device. The cam tracks associated with the selected axis or external encoder are automatically identified and displayed in the **Cam track** field for selection as appropriate. • **<Reference>** You select this entry if the cam track is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type _CamTrackType (references to cam tracks) are available for selection in the **Cam track** field, see also Technology object data types (Page 4060). **Note** You cannot select references to an axis or external encoder (variables of data type PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType). There is no assignment for the reference to the associated cam tracks. Instead, select the reference to the cam track directly (variable of data type _CamTrackType).
Cam track	Here, the permissible cam tracks are displayed according to the selected **Axis/Encoder** and are available for selection, as appropriate: • A position, following or path axis or external encoder defined on the device was selected as the axis/encoder: The cam tracks associated with the selected axis/encoder are available for selection. • **<Reference>** was selected as the axis/encoder: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: _CamTrackType. These variables are references to cam tracks.
Parameters tab	See Overview of parameters for Cam track off - Parameters tab (Page 4384)
Expert tab	See Overview of parameters for Cam track off - Expert tab (Page 4384)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Output cam track Off – Parameters tab

Table 7-250 Overview of parameters for Cam track off – Parameters tab

Field/button	Meaning/instruction
Stop mode	The stop mode is used to define the behavior of the cam track on deactivation. **Immediately** The cam track processing is deactivated immediately. Any active single cam of the cam track is aborted. **Immediately when output cam output is inactive** The cam track processing is deactivated as soon as there are no more active single output cams of the cam track. An active single output cam will be output completely. **At end of cam track** The cam track processing is deactivated as soon as the last single cam of the track switches. Up to that point, a time-based cam will be output. The stop mode is used to define when the cam track should become effective after activation, or how tracks should be changed. **Last programmed** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.startMode.

Overview of parameters for Output cam track Off – Expert tab

Table 7-251 Overview of parameters for Cam track off – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Output cam track Off

Cam technology package:

- _disableCamTrack

Overview of parameters for Cam track off / _disableCamTrack

Table 7-252 Parameters (MCC command Cam track off compared to system function _disableCamTrack)

Parameters of the MCC command Cam track off	Parameters of the system function _disableCamTrack
Axis/encoder	–
Cam track	camtrack
Delay program execution	nextCommand
Parameters tab	
Stop mode	stopMode
Expert tab	
CommandID variable	commandId
Return variable	–

7.1.6.7 Commands for synchronous operation and camming

Gearing on

Gearing is characterized by a constant gear ratio between a master value (position value of a leading axis or external encoder) and a slave value (position value of a following axis). Gearing is thus in positional synchronism.

This transmission ratio (gear ratio) can be specified as the ratio of two integers (numerator/ denominator) or as a decimal number.

An offset in the zero point (a phase displacement) can also be specified.

When gearing is started, the slave value is synchronized to the master value with the programmed synchronization settings.

The master value can be defined in the command; the master value can be modified later with the "Switch master value" (Page 4458).

Other master value sources for SIMOTION devices as of Version V4.2 of SIMOTION Kernel only

As of Version V4.2 of SIMOTION Kernel, the instances of the following technology objects and the variables of the relevant technology object data type are available as master value sources:

- Drive axis
- Path object
- Addition object
- Controller object
- Fixed gear

- Formula object
- Sensor

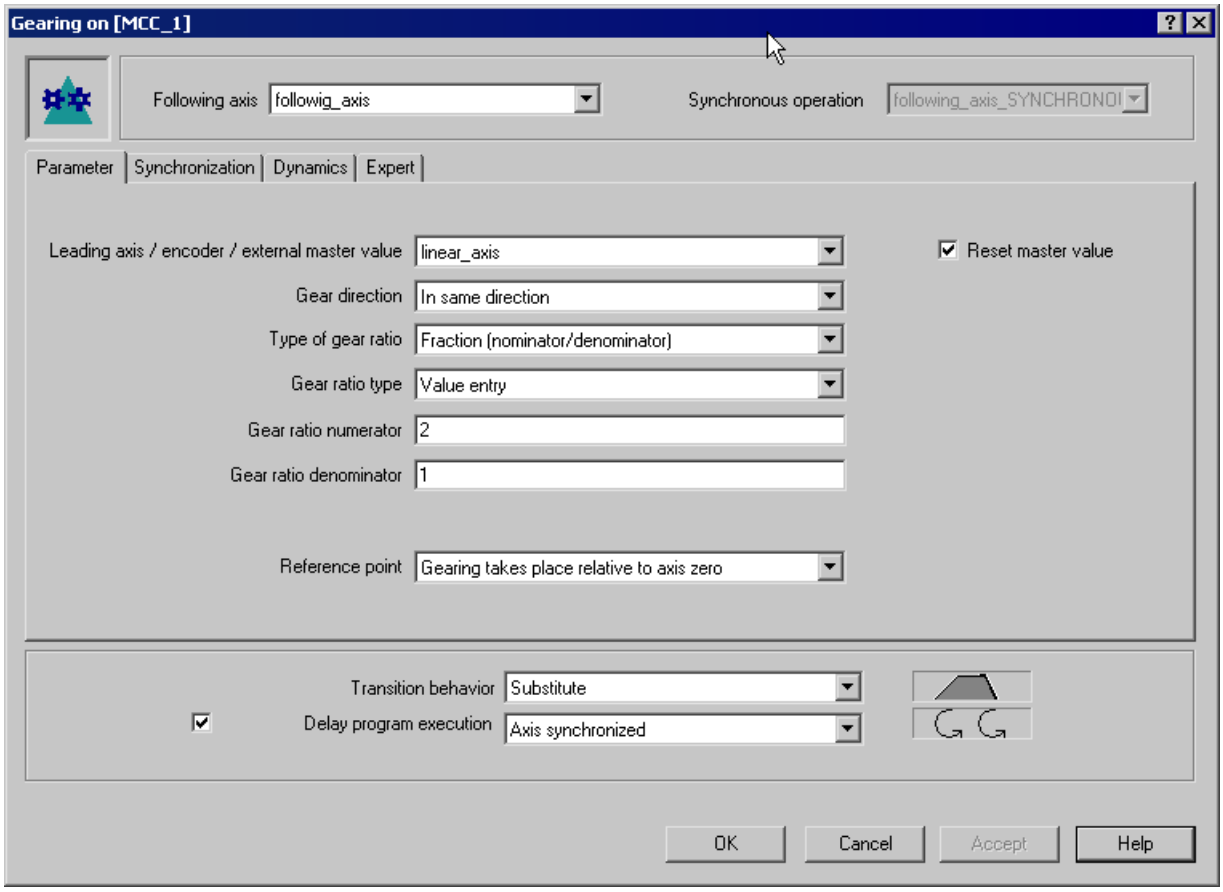


Figure 7-158 Parameter screen form: Gearing on

Active gearing can be offset both on the side of the master value and on the side of the slave value. The "Set offset on the gearing" (Page 4406) command is used for this purpose.

Gearing can be ended:

- With the "Gearing off" (Page 4399) command
- By starting another synchronous operation on the same synchronous object, e.g. with another "Gearing on" command
- By a canceling single axis command on the following axis

Note

The "Stop axis" (Page 4281) command will only terminate the synchronous operation if **Quick stop...** has been selected as the stop mode.

Further information:

- **General information on gearing**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on synchronization**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Gearing on

Table 7-253 Overview of parameters for Gearing on

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be synchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • **<Reference>** You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Gearing on - Parameters tab (Page 4388)
Synchronization tab	See Overview of parameters for Gearing on - Synchronization tab (Page 4391)
Dynamic response tab	See Overview of parameters for Gearing on - Dynamic response tab (Page 4397)
Expert tab	See Overview of parameters for Gearing on - Expert tab (Page 4397)

Field/button	Meaning/instruction
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Example:

See Example of Gearing on (Page 4398)

Overview of parameters for Gearing on – Parameters tab

Table 7-254 Overview of parameters for Gearing on – Parameters tab

Field/button	Meaning/instruction
Reset master value	Activate this checkbox if you want to set the master value of the synchronous relationship (default) If the checkbox is deactivated, a previous master value setting is retained. **Note** • As of Version V4.1 of the SIMOTION Kernel, the master value conversion takes place during the synchronization of the synchronous axis (in accordance with the transition behavior "with next synchronization" (Page 4461) with the command "Switch master value"). • From Version V4.0 of the SIMOTION Kernel, the master value conversion takes place immediately at the start of the command (in accordance with the transition behavior "direct" (Page 4461) with the command "Switch master value").
Leading axis / encoder / external master value	An entry can only be made in this field if the **Reset master value** checkbox is activated. Here, you select the axis or external encoder that generates the master value in the synchronous relationship. You can select from: • All position, following, and path axes and external encoders that are available on the device or a DP master. The following are also available as of Version V4.2 of SIMOTION Kernel: Drive axes, path objects, addition objects, controller objects, fixed gear, formula objects and sensors. • All technology object-type variables declared in the MCC unit or MCC chart, see Description of the technology object data types (Page 4060): PosAxis, FollowingAxis, _PathAxis and ExternalEncoderType The following are also available as of Version V4.2 of SIMOTION Kernel: DriveAxis, _PathObjectType, _AdditionObjectType, _ControllerObjectType, _FixedGearType, _FormulaObjectType, _SensorType. The master value remains assigned to the synchronous object until a change occurs.

Field/button	Meaning/instruction
Gear direction	Here, you select the effective direction of the gearing. **From sign of gear ratio** (default value) The effective direction of gearing is defined by the sign of the gear ratio. **Opposite direction** The following axis moves in the opposite direction from the leading axis. **Same direction** The following axis moves in the same direction as the leading axis. **Opposite current gearing direction** The gearing direction opposes the current gearing direction. **Current direction** **Last programmed direction** **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.gearingSettings.direction
Gear ratio type	The gear ratio is the ratio between the distances covered by the following axis (= numerator) and the leading axis (= denominator), each expressed in the configured units. Here, you select whether the gear ratio is specified as a fraction (numerator/denominator) or as a floating-point number. **Fraction (numerator/denominator)** The ratio is specified as a fraction **Floating-point number** The ratio is specified as a floating-point number **Last programmed** **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.gearingSettings.defineMode
Gear ratio type	Here you select which values are used for the gear ratio **Value entry** The values are entered directly in the parameter dialog box. • As fraction (numerator / denominator): In the Gear ratio numerator/denominator fields, or • As floating-point number: In the Gear ratio field **Last programmed value** **Default** See Selection list (combo box) (Page 4022) System variables for default: • userDefault.gearingSettings.numerator and userDefault.gearingSettings.denominator or • userDefault.gearingSettings.ratio
Gear ratio numerator	Here, you enter the numerator of the gear ratio as an integer. The unit configured for the following axis is displayed for your information. See also input field (Page 4022).

Field/button	Meaning/instruction
Gear ratio denominator	Here, you enter the denominator of the gear ratio as an integer. The unit configured for the leading axis is displayed for your information. See also input field (Page 4022).
Gear ratio	Enter the gear ratio as a floating-point number here. See also input field (Page 4022).
Reference point	Here, you select the reference point for gearing. **Gearing takes place relative to axis zero (default value)** Absolute gearing: In each case, the linear coupling of the slave value to the master value refers to the axis zero of the axes involved. Any offset between the master value and the slave value at the start of gearing is compensated for during synchronization. For some of the choices, an offset can be specified in the **Start of synchronization** field of the Synchronization tab (see Overview of parameters for Gearing on - Synchronization tab (Page 4391)). This value is retained as a constant phase shift between slave value and master value after conclusion of the synchronization operation. Otherwise, the phase shift is equal to 0. **Gearing relative to start position** Relative gearing: In each case, the linear coupling of the slave value to the master value refers to the position value of the axes involved at the start of gearing. Any offset between the master value and the slave value at the start of gearing is **not** compensated for during synchronization. This offset is retained as a constant phase shift between slave value and master value after conclusion of the synchronization operation. In addition, for some of the choices, an offset can be specified in the **Start of synchronization** field of the Synchronization tab (see Overview of parameters for Gearing on - Synchronization tab (Page 4391)). This value acts as a constant phase shift between slave value and master value after conclusion of the synchronization operation. **Last programmed reference point** **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.syncProfile.syncProfileReference

Overview of parameters for Gearing on – Synchronization tab

Table 7-255 Overview of parameters for Gearing on – Synchronization

Field/button	Meaning/instruction
Synchronization reference	Here, you select the reference for the synchronization operation. **Leading axis** Length-related synchronization: Synchronization occurs (depending on the leading axis motion) within a defined range of the master value (**Synchronization length**). • Advantage: Synchronization occurs within a parameterizable range of the master value. • Disadvantage: The dynamic response of synchronization is dependent on the change in the master value (velocity). The dynamic response limit values of the following axis are not taken into account. **Time** Time-related synchronization: Synchronization occurs on the basis of specified dynamic values. Program the relevant values in the Dynamic response tab, see Overview of parameters for Gearing on - Dynamic response tab (Page 4397). • Advantage: The synchronization operation always takes place with the specified dynamic values. • Disadvantage: The master value range in which synchronization occurs cannot be predicted. **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncProfileReference

Field/button	Meaning/instruction
Start of synchronization	Here, you select when synchronization of gearing begins. Additional specifications are required for many of the choices; these are indicated in the description. **At leading axis position** The gear is enabled at a programmed leading axis position. Entries are required in the following fields: • Reference point of leading axis position • Leading axis position Synchronization occurs with respect to the following values: • Master value = programmed leading axis position • Slave value: Depending on the selection in **Reference point**, see Overview of parameters for Gearing on – Parameters tab (Page 4388): – For **gearing, relative to axis zero (default value)**: Slave value = gear ratio * position of leading axis – For **gearing, relative to start position**: Slave value = current slave value **At leading axis position with offset** The gear is enabled at a programmed leading axis position. An offset for the following axis is additionally programmed. Entries are required in the following fields: • Offset of the following axis • Reference point of leading axis position • Leading axis position Synchronization occurs with respect to the slave values: • Master value = programmed leading axis position • Slave value: See description in the field **Offset of the following axis** **Synchronize immediately** The gear is enabled immediately. Synchronization occurs with respect to the slave values: • Master value = current master value • Slave value: Depending on the selection in **Reference point**, see Overview of parameters for Gearing on – Parameters tab (Page 4388): – For **gearing, relative to axis zero (default value)**: Slave value = gear ratio * current master value – For **gearing, relative to start position**: Slave value = current slave value **...** (Continued in next row of table)

Field/button	Meaning/instruction
Start of synchronization (continued)	... **Synchronize immediately with offset** The gear is enabled immediately. An offset for the following axis is additionally programmed. An entry is required in the following field: • Offset of the following axis Synchronization occurs with respect to the slave values: • Master value = current master value • Slave value: See description in the field **Offset of the following axis** **At the following axis position** The gear is enabled at a programmed following axis position. Entries are required in the following fields: • Reference point of leading axis position • Following axis position Synchronization occurs with respect to the slave values: • Master value: Depending on the selection in **Reference point**, see Overview of parameters for Gearing on – Parameters tab (Page 4388): – For **gearing, relative to axis zero (default value)**: Master value = position of leading axis / gear ratio – For **gearing, relative to start position**: Master value = current master value • Slave value = last programmed following axis position **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.synchronizingMode
Offset of the following axis	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position with offset • Synchronize immediately with offset Here, you enter a value that is used to calculate the offset of the following axis for linear coupling between the leading axis and following axis. Depending on the selection in **Reference point** (see Overview of parameters for Gearing on – Parameters tab (Page 4388)), the entered value has the following meaning: • For **gearing, relative to axis zero (default value)**: Slave value relative to which synchronization occurs • For **gearing, relative to start position:** Additional offset to any offset between the following axis and leading axis Depending on the selection in **Reference point**, see Overview of parameters for Gearing on – Parameters tab (Page 4388), the slave value relative to which synchronization occurs is thus: • For **gearing, relative to axis zero (default value)**: Slave value = programmed offset • For **gearing, relative to start position**: Slave value = current slave value + programmed offset See also input field (Page 4022).

Field/button	Meaning/instruction
Reference point of leading axis position	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position • At leading axis position with offset • At the following axis position Here, you select how the programmed position (see the **Following axis position** or **Leading axis position** field) acts relative to the selected synchronization profile. **Synchronize before synchronization position** Synchronization is finished at the programmed position (preceding synchronization). **Symmetrical** This choice is available only for leading axis synchronization reference. Synchronization occurs in such a way that the programmed position is located symmetrically within the synchronization length. At the programmed position, the leading axis has covered half the distance that is required for synchronization. **Synchronize from synchronization position** Synchronization starts at the programmed position (succeeding synchronization). **Last programmed reference point of leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncPositionReference
Synchronization length	An entry is required in this field if you have selected the following in the **Synchronization reference** field: • Leading axis Enter the synchronization length in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed synchronization length** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncLength
Following axis position	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At the following axis position Enter the following axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed following axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.syncPositionSlave

Field/button	Meaning/instruction
Leading axis position	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position • At leading axis position with offset Enter the leading axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.syncPositionMaster

Field/button	Meaning/instruction
Synchronization with look-ahead	Per default, the position and the velocity of the master value are taken into account in the calculations for the synchronization. With succeeding synchronization (i.e. the synchronization starts at the programmed position), you can also select whether a constant acceleration or deceleration of the master value is to be taken into account (extended look-ahead). **Note** Extended look-ahead requires a lot of computing time. An entry in this field is possible in the following cases: • In the **Start of synchronization** field, you have selected**:** – Synchronize immediately – Synchronize immediately with offset • With a different selection in the **Start of synchronization** field, you ave also selected in the **Reference point of the leading axis position** field: – Synchronize from synchronization position – Default Select one of the following menu commands: **Standard** Position and velocity of the master value are taken into account in the synchronization calculations. **Expanded look ahead** In addition to the position and velocity of the master value, a constant acceleration or deceleration of the master value is also taken into account in the synchronization calculations. With preceding or symmetrical synchronization, this selection has no effect. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.synchronizingWithLookAhead
Synchronization direction	Here, you select the direction of motion in which the following axis is to be synchronized. **Retain system behavior** Synchronization occurs over the shortest path without direction specification. In the case of moving axes, a check is made to determine whether the current direction of motion can be maintained. **Maintain direction of the following axis** Synchronization takes place in the direction of motion of the following axis. **Positive** Synchronization occurs in the positive direction of motion. **Negative** Synchronization occurs in the negative direction of motion. **Shortest path** Synchronization occurs over the shortest path without direction specification. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.synchronizingDirection

Overview of parameters for Gearing on – Dynamic response tab

Table 7-256 Overview of parameters for Gearing on – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters on the Dynamic response tab are only evaluated for time synchronization reference.
Velocity	The entered value acts during the constant velocity phase. System variable for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefault.syncDynamics.profile **Note** A constant velocity profile only takes effect if configuration data syncingMotion.smoothAbsoluteSynchronization = YES is set. When syncingMotion.smoothAbsoluteSynchronization = NO (default), a trapezoidal velocity profile is always active.
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk, userDefault.syncDynamics.positiveAccelEndJerk, userDefault.syncDynamics.negativeAccelStartJerk, userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Gearing on – Expert tab

Table 7-257 Overview of parameters for Gearing on – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Example of Gearing on

An axis (rotaryAxis_1) is to rotate at a velocity of 360 degrees/s. A second axis (synchronous axis) is to rotate in synchronism using a gear ratio.

The synchronous axis that is to be coupled with rotaryAxis_1 by means of a gear ratio must be assigned as a synchronous axis in SIMOTION SCOUT.

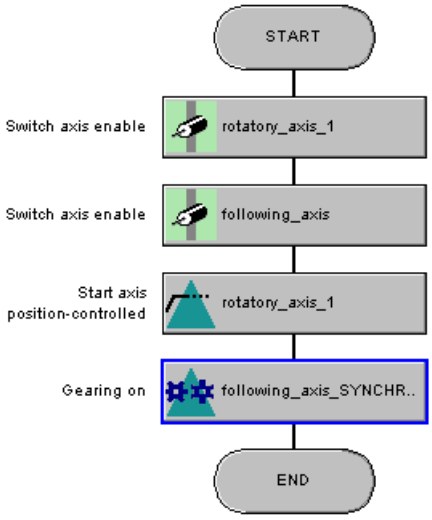


Figure 7-159 Example: MCC chart for gearing

Relevant system functions for Gearing on

Cam technology package:

- _enableGearing
- _setMaster

If the **Reset master value** checkbox is selected, the _setMaster system function is called first before _enableGearing.

Overview of parameters for Gearing on/_enableGearing, _setMaster

Table 7-258 Parameters (MCC command Gearing on compared to system functions _enableGearing and _setMaster)

Parameters of the MCC command Gearing on	**Parameters of the system functions _enableGearing and _setMaster**
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	

Parameters of the MCC command Gearing on	Parameters of the system functions _enableGearing and _setMaster
Reset master value	Call of _setMaster system function • Up to SIMOTION Kernel Version V4.0 with parameter transientBehavior = DIRECT • SIMOTION Kernel Version V4.1 and higher with parameter transientBehavior = WITH_NEXT_SYNCHRONIZING
Leading axis / encoder / external master value	master (system function _setMaster)
Gear direction	direction
Gear ratio type	gearingMode
Gear ratio type	gearingRatioType
Gear ratio numerator	gearingNumerator
Gear ratio denominator	gearingDenominator
Gear ratio	gearingRatio
Reference point	gearingType
Synchronization tab	
Synchronization reference	syncProfileReference
Start of synchronization	synchronizingMode
Offset of the following axis	syncPositionSlave
Reference point of leading axis position	syncPositionReference
Synchronization length	syncLengthType
Following axis position	syncPositionSlaveType, syncPositionSlave
Leading axis position	syncPositionMasterType, syncPositionMaster
Synchronization with look-ahead	synchronizingWithLookAhead
Synchronization direction	synchronizingDirection
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Gearing off

This command enables you to terminate gearing that was started with the "Gearing on" (Page 4385) command. Desynchronization occurs with the programmed desynchronization settings.

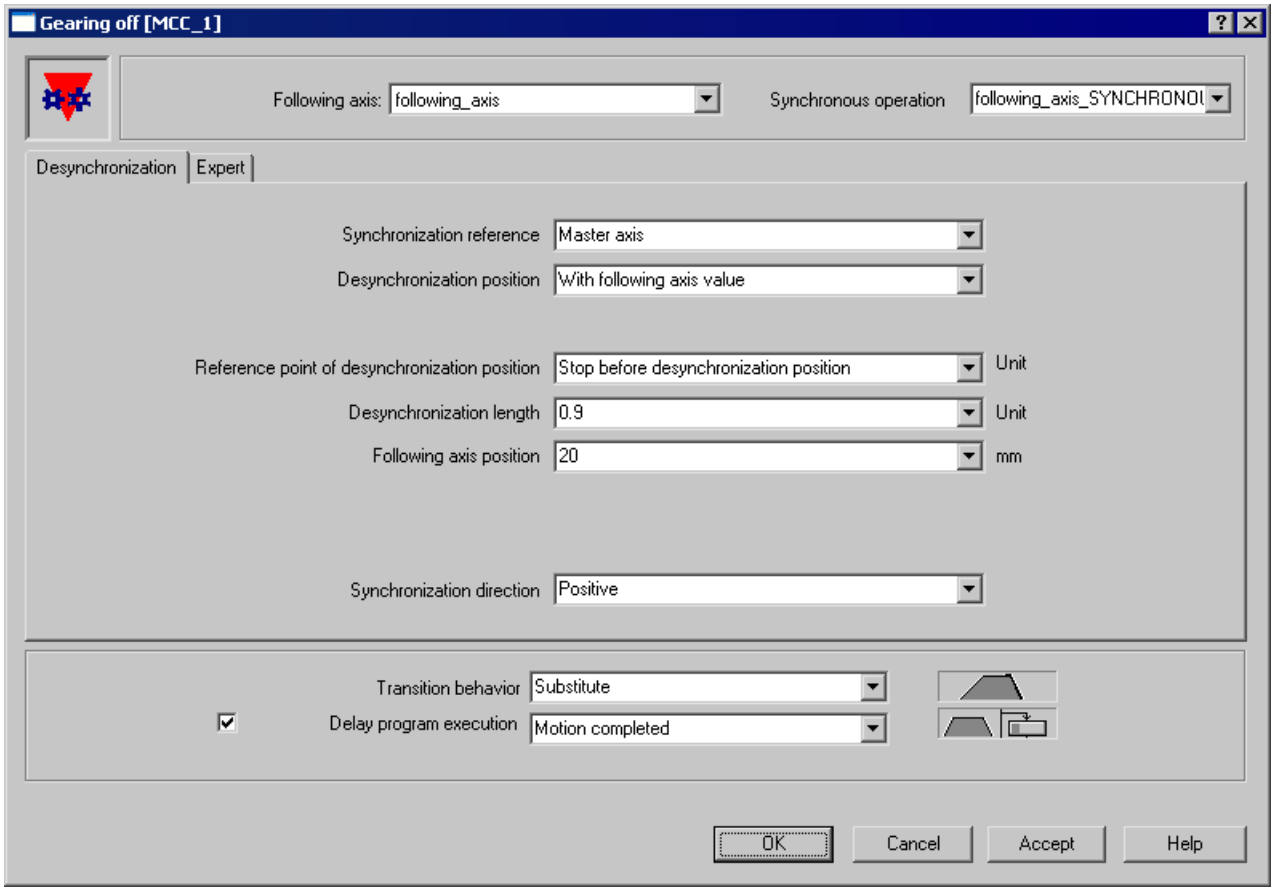


Figure 7-160 Parameter screen form: Gearing off

Further information:

- For **general information on gearing**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on synchronization**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Gearing off

Table 7-259 Overview of parameters for Gearing off

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be desynchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be desynchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Description of the technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Desynchronization tab	See Overview of parameters for Gearing off – Desynchronization tab (Page 4402)
Dynamic response tab	See Overview of parameters for Gearing off – Dynamic response tab (Page 4404)
Expert tab	See Overview of parameters for Gearing off – Expert tab (Page 4405)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036)
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Gearing off – Desynchronization tab

Table 7-260 Overview of parameters for Gearing off – Desynchronization tab

Field/button	Meaning/instruction
Synchronization reference	Here you select the reference for the desynchronization operation. **Leading axis** Length-related synchronization: Synchronization occurs (depending on the leading axis motion) within a defined range of the master value (**Desynchronization length**). • Advantage: Synchronization occurs within a parameterizable range of the master value. • Disadvantage: The dynamic response of synchronization is dependent on the change in the master value (velocity). The dynamic response limit values of the following axis are not taken into account. **Time** Time-related synchronization: Synchronization occurs on the basis of specified dynamic values. Program the relevant values in the Dynamic response tab, see Overview of parameters for Gearing off - Dynamic response tab (Page 4404)). • Advantage: The synchronization operation always takes place with the specified dynamic values. • Disadvantage: The master value range in which synchronization occurs cannot be predicted. **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncProfileReference
Desynchronization position	Here, you select when desynchronization of gearing begins. Additional specifications are required for some of the choices; these are indicated in the description. **At leading axis value** Gearing is disabled at a programmed leading axis position. Entries are required in the following fields: • Reference point of desynchronization position • Leading axis position **Desynchronize immediately** Gearing is disabled immediately **At following axis value** Gearing is disabled at a programmed slave axis position. Entries are required in the following fields: • Reference point of desynchronization position • Following axis position **Last programmed start of desynchronization** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.syncOffMode

Field/button	Meaning/instruction
Reference point of desynchronization position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At leading axis value • At following axis value Here, you select how the programmed position (see the **Following axis position** or **Leading axis position** field) acts relative to the selected desynchronization profile. **Stop before desynchronization position** Synchronization is finished at the programmed position. **Symmetrical** This choice is available only for leading axis synchronization reference. Desynchronization occurs in such a way that the programmed position is located symmetrically within the desynchronization length. At the programmed position, the leading axis has covered half the distance that is required for synchronization. **Stop from desynchronization position** Desynchronization starts at the programmed position. **Last programmed reference point of leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncOffPositionReference
Desynchronization length	An entry is required in this field if you have selected the following in the **Synchronization reference** field: • Leading axis Enter the synchronization length in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed desynchronization length** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncOffLength
Following axis position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At following axis value Enter the following axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncOffPositions.Slave

Field/button	Meaning/instruction
Leading axis position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At leading axis value Enter the leading axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncOffPositions.Master
Synchronization direction	Here, you select the direction of motion in which the following axis is to be desynchronized. **Retain system behavior** Synchronization occurs over the shortest path without direction specification. In the case of moving axes, a check is made to determine whether the current direction of motion can be maintained. **Maintain direction of the following axis** Synchronization takes place in the direction of motion of the following axis. **Positive** Synchronization occurs in the positive direction of motion. **Negative** Synchronization occurs in the negative direction of motion. **Shortest path** Synchronization occurs over the shortest path without direction specification. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.gearingSettings.synchronizingDirection

Overview of parameters for Gearing off – Dynamic response tab

Table 7-261 Overview of parameters for Gearing off – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters on the Dynamic response tab are only evaluated for time synchronization reference.
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Gearing off – Expert tab

Table 7-262 Overview of parameters for Gearing off – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Gearing off

Cam technology package:

- _disableGearing

Overview of parameters for Gearing off / _disableGearing

Table 7-263 Parameters (MCC command Gearing off compared to system function _disableGearing)

Parameters of the MCC command Gearing off	Parameters of the system function _disableGearing
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Desynchronization tab	
Synchronization reference	syncProfileReference
Desynchronization position	syncOffMode
Reference point of desynchronization position	syncOffPositionReference
Desynchronization length	syncLengthType
Following axis position	syncOffPositionSlaveType, syncOffPositionSlave

Parameters of the MCC command Gearing off	Parameters of the system function _disableGearing
Leading axis position	syncOffPositionMasterType, syncOffPositionMaster
Synchronization direction	synchronizingDirection
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Set offset on the gearing

This command causes an offset in the leading axis or following axis range during the gearing operation. You can use a parameter to determine when the programmed offset is to take effect (as well as other properties).

- For the active gearing command
  and/or
- For subsequent gearing commands

Offsets that act on the active gearing command and offsets that are stored and applied to subsequent gearing commands can be read from the gearingAdjustment system variable.

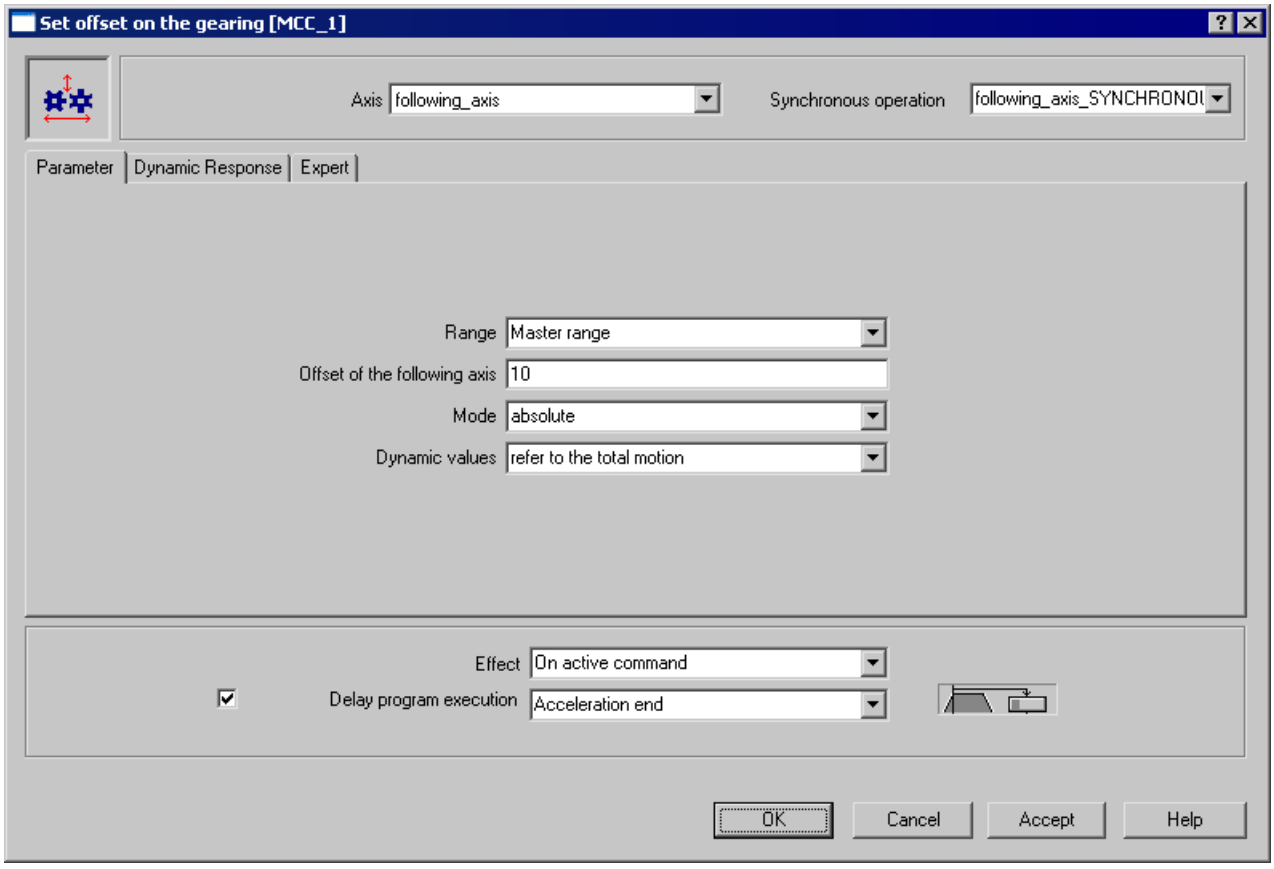


Figure 7-161 Parameter screen form: Set offset on the gearing

Further information:

- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For information on **scaling and offset**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Set offset on the gearing

Table 7-264 Overview of parameters for Set offset on the gearing

Field/button	Meaning/instruction
Axis	Here, you select the axis for which gearing is to be offset. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as an axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as the axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Set offset on the gearing – Parameters tab (Page 4409)
Dynamic response tab	See Overview of parameters for Set offset on the gearing – Dynamic response tab (Page 4410)
Expert tab	See Overview of parameters for Set offset on the gearing – Expert tab (Page 4410)

Field/button	Meaning/instruction
Effect	Here, you select which commands the offset will act on. **On active command** (default value) The offset acts on the active gearing command and remains active until this command is replaced or desynchronized using the "Gearing off" (Page 4399) command. The offset value is stored in system variables gearingAdjustment.master.offset or gearingAdjustment.slave.offset (depending on the range). **on following commands** The active gearing command is executed; the offset is stored and in effect for subsequent "Gearing on" (Page 4385) commands. The offset value is stored in system variables gearingAdjustment.defaultValueMaster.offset or gearingAdjustment.defaultValueSlave.offset (depending on the range). **On active command and following commands** The offset acts on the current gearing command and on all subsequent "Gearing on" (Page 4385) commands. The offset value is stored in system variables gearingAdjustment.master.offset and gearingAdjustment.defaultValueMaster.offset or gearingAdjustment.slave.offset and gearingAdjustment.defaultValueSlave.offset (depending on the range).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Set offset on the gearing – Parameters tab

Table 7-265 Overview of parameters for Set offset on the gearing – Parameters tab

Field/button	Meaning/instruction
Range	Here, you select whether the leading axis or following axis range will be offset. **Master range** (default value) The leading axis range is offset. **Slave range** The following axis range is offset.
Offset	Offset for the selected range. Enter the value as a signed floating-point number. See also input field (Page 4022).
Mode	Select the type of offset. **Absolute**(default value) The offset is applied as an absolute value. **Relative** The offset is relative to the corresponding value in the gearingAdjustments system variables.

Overview of parameters for Set offset on the gearing – Dynamic response tab

Table 7-266 Overview of parameters for Set offset on the gearing – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4034). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4034). System variables for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Set offset on the gearing – Expert tab

Table 7-267 Overview of parameters for Set offset on the gearing – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Set offset on the gearing

Cam technology package:

- _setGearingOffset

Overview of parameters for Set offset on the gearing / _setGearingOffset

Table 7-268 Parameters (MCC command Set gearing on the offset compared to system function _setGearingOffset)

Parameters of the MCC command Set offset on the gearing	**Parameters of the system function _setGearingOffset**
Axis	–
Synchronous operation	followingObject
Effect	activationMode
Delay program execution	nextCommand
Parameters tab	
Range	offsetRange
Offset	offsetValue
Mode	offsetMode
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Synchronous velocity operation on

Velocity gearing is characterized by a constant gear ratio between a master value (velocity of a leading axis or external encoder) and a slave value (velocity of a following axis). Unlike gearing, velocity gearing is **not** in positional synchronism.

When velocity gearing is started, the slave value is synchronized to the master value with the programmed dynamic response settings.

The gear ratio is specified as a decimal number.

The master value can be defined in the command; the master value can be modified later with the "Switch master value" (Page 4458).

Other master value sources for SIMOTION devices as of Version V4.2 of SIMOTION Kernel only

As of Version V4.2 of SIMOTION Kernel, the instances of the following technology objects and the variables of the relevant technology object data type are available as master value sources:

- Drive axis
- Path object
- Addition object
- Controller object
- Fixed gear
- Formula object
- Sensor

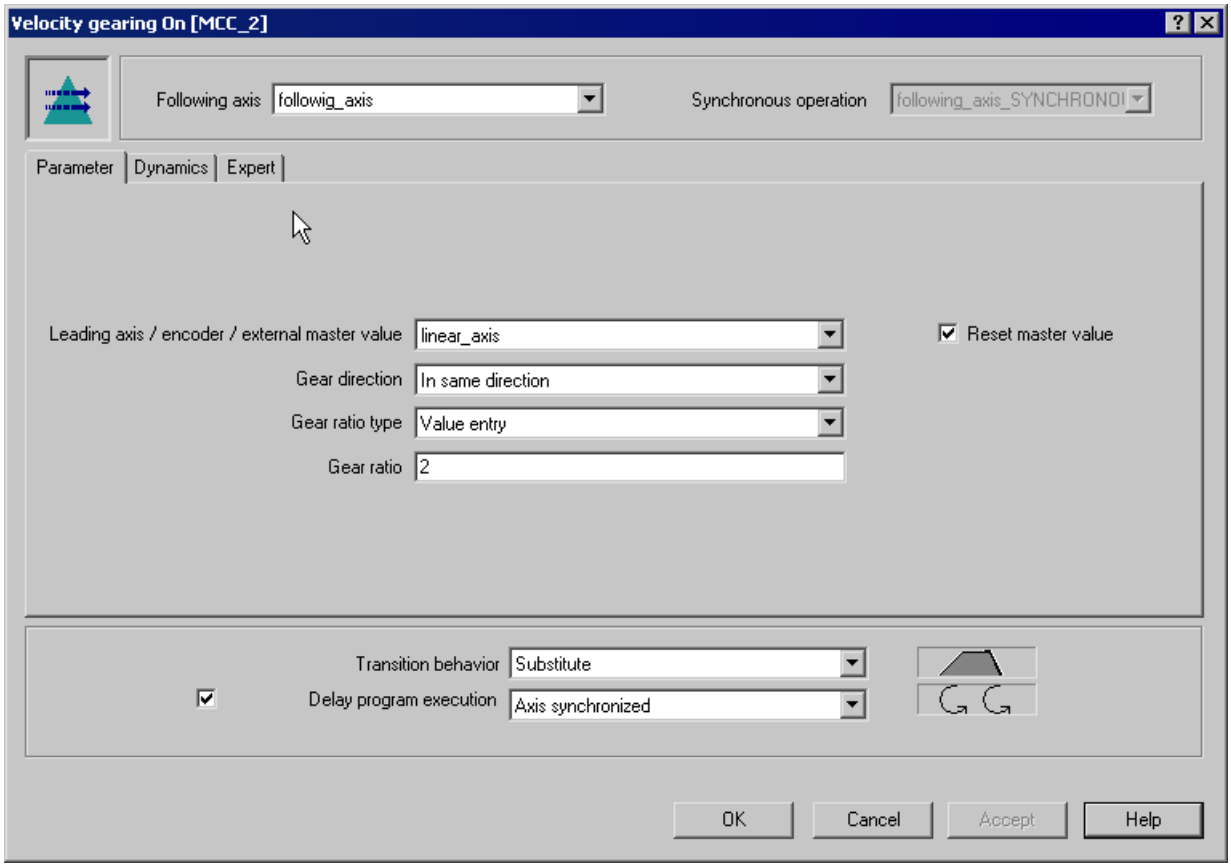


Figure 7-162 Parameter screen form: Velocity gearing on

Velocity gearing can be terminated:

- With the "Velocity gearing off" (Page 4417) command
- By starting another synchronous operation on the same synchronous object, e.g. with another "Velocity gearing on" command
- With a canceling single axis command on the following axis

Note

The "Stop axis" (Page 4281) command will only terminate the velocity gearing if **Quick stop...** has been selected as the stop mode.

Further information:

- For **general information on velocity gearing**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of Synchronous velocity operation on

Table 7-269 Overview of Velocity gearing on

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be synchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Velocity gearing on - Parameters tab (Page 4414)
Dynamic response tab	See Overview of parameters for Velocity gearing on - Dynamic response tab (Page 4415)

Field/button	Meaning/instruction
Expert tab	See Overview of parameters for Velocity gearing on - Expert tab (Page 4416)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Synchronous velocity operation on – Parameters tab

Table 7-270 Overview of parameters for Velocity gearing on – Parameters tab

Field/button	Meaning/instruction
Reset master value	Activate this checkbox if you want to set the master value of the velocity relationship (default). If the checkbox is deactivated, a previous master value setting is retained. **Note** • As of Version V4.1 of the SIMOTION Kernel, the master value conversion takes place during the synchronization of the synchronous axis (in accordance with the transition behavior "with next synchronization" (Page 4461) with the command "Switch master value"). • From Version V4.0 of the SIMOTION Kernel, the master value conversion takes place immediately at the start of the command (in accordance with the transition behavior "direct" (Page 4461) with the command "Switch master value").
Leading axis / encoder / external master value	An entry can only be made in this field if the **Reset master value** checkbox is activated. Here, you select the axis or external encoder that generates the master value in the velocity relationship. You can select from: • All position, following, and path axes and external encoders that are available on the device or a DP master. The following are also available as of Version V4.2 of SIMOTION Kernel: Drive axes, path objects, addition objects, controller objects, fixed gear, formula objects and sensors. • All variables with the following technology object data types declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)): PosAxis, FollowingAxis, _PathAxis and ExternalEncoderType. The following are also available as of Version V4.2 of SIMOTION Kernel: DriveAxis, _PathObjectType, _AdditionObjectType, _ControllerObjectType, _FixedGearType, _FormulaObjectType, _SensorType. The master value remains assigned to the velocity object until a change occurs.

Field/button	Meaning/instruction
Gear direction	Here, you select the effective direction of the gearing. **From sign of gear ratio** (default value) The effective direction of gearing is defined by the sign of the gear ratio. **Opposite direction** The following axis moves in the opposite direction from the leading axis. **Same direction** The following axis moves in the same direction as the leading axis. **Opposite current gearing direction** The gearing direction opposes the current gearing direction. **Current direction** **Last programmed direction** **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.gearingSettings.direction
Gear ratio type	Here you select which values are used for the gear ratio **Value entry** The values are entered directly in the parameter dialog box (in the gear ratio field) **Last programmed value** **Default** See Selection list (combo box) (Page 4022) System variables for default: userDefault.gearingSettings.ratio
Gear ratio	Enter the gear ratio as a floating-point number here. See also input field (Page 4022).

Overview of parameters for Synchronous velocity operation on – Dynamic response tab

Table 7-271 Overview of parameters for Velocity gearing on – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Synchronous velocity operation on – Expert tab

Table 7-272 Overview of parameters for Velocity gearing on – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Example of Synchronous velocity operation on

An axis (rotaryAxis_1) is to rotate at a velocity of 360 degrees/s. A second axis (synchronous axis) is to rotate in synchronism using a gear ratio.

The synchronous axis that is to be coupled with rotaryAxis_1 by means of a gear ratio must be assigned as a synchronous axis in SIMOTION SCOUT.

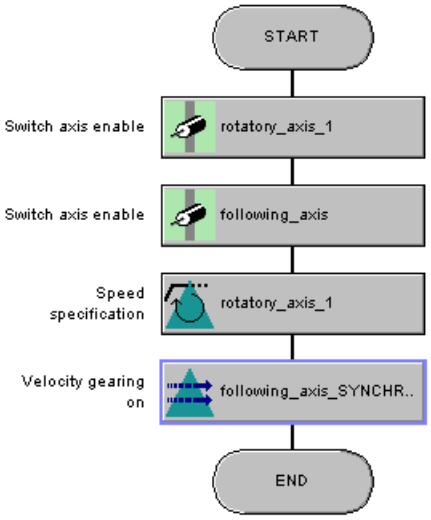


Figure 7-163 Example: MCC chart for velocity gearing

Relevant system functions for Synchronous velocity operation on

Cam technology package:

- _enableVelocityGearing
- _setMaster

Overview of parameters for Velocity gearing on /_enableVelocityGearing, _setMaster

Table 7-273 Parameters (MCC command Velocity gearing on compared to system functions _enableVelocityGearing and _setMaster)

Parameters of the MCC command Velocity gearing on	**Parameters of the system functions _enableVelocityGearing and _setMaster**
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Reset master value	Call of _setMaster system function • Up to SIMOTION Kernel Version V4.0 with parameter transientBehavior = DIRECT • SIMOTION Kernel Version V4.1 and higher with parameter transientBehavior = WITH_NEXT_SYNCHRONIZING
Leading axis / encoder / external master value	master (system function _setMaster)
Gear direction	direction
Gear ratio type	gearingRatioType
Gear ratio	gearingRatio
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Synchronous velocity operation off

This command enables you to terminate gearing that was started with the "Velocity gearing on" (Page 4411) command. Desynchronization occurs with the programmed dynamic response settings.

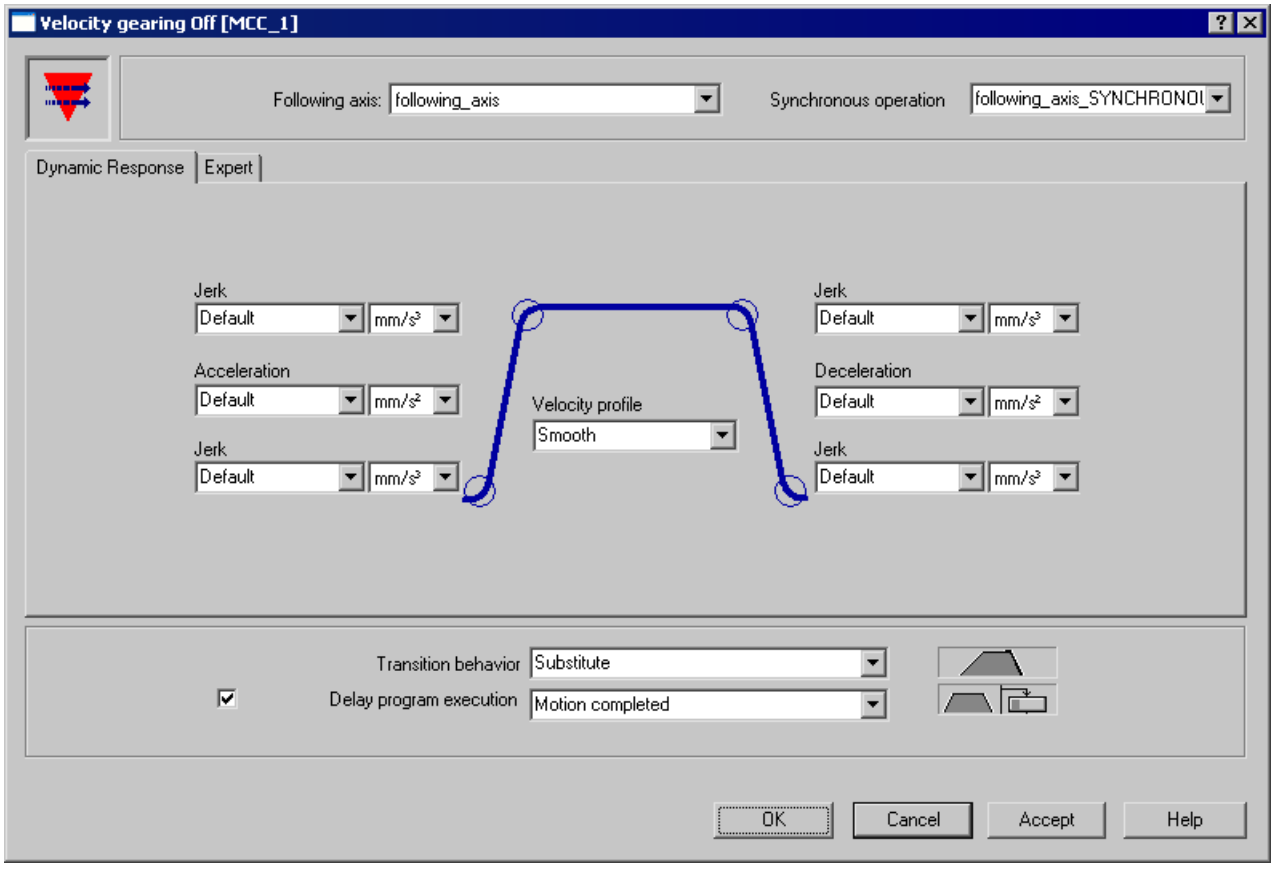


Figure 7-164 Parameter screen form: Velocity gearing off

Further information:

- For **general information on velocity gearing**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Synchronous velocity operation off

Table 7-274 Overview of parameters for Velocity gearing off

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be desynchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be desynchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Dynamic response tab	See Overview of parameters for Velocity gearing off - Dynamic response tab (Page 4420)
Expert tab	See Overview of parameters for Velocity gearing off - Expert tab (Page 4420)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036)
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Synchronous velocity operation off – Dynamic response tab

Table 7-275 Overview of parameters for Velocity gearing off – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031)
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Synchronous velocity operation off – Expert tab

Table 7-276 Overview of parameters for Velocity gearing off – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Synchronous velocity operation off

Cam technology package:

- _disableVelocityGearing

Overview of parameters for Velocity gearing off / _disableVelocityGearing

Table 7-277 Parameters (MCC command Velocity gearing off compared to system function _disableVelocityGearing)

Parameters of the MCC command Velocity gearing off	Parameters of the system function _disableVelocityGearing
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Dynamic response tab	
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Cam on

Camming is characterized by a variable gear ratio between a master value (position value of a leading axis or external encoder) and a slave value (position value of a following axis). Camming is thus in positional synchronism.

The variable gear ratio is described by a cam (transmission function).

You can adapt the camming functionality to your requirements using various programmable axis evaluations and cam execution modes.

When camming is started, the slave value is synchronized to the master value with the programmed synchronization settings. It is also possible to switch to another cam at the end of an active cam.

The master value can be defined in the command; the master value can be modified later with the "Switch master value" (Page 4458).

Other master value sources for SIMOTION devices as of Version V4.2 of SIMOTION Kernel only

As of Version V4.2 of SIMOTION Kernel, the instances of the following technology objects and the variables of the relevant technology object data type are available as master value sources:

- Drive axis
- Path object
- Addition object

- Controller object
- Fixed gear
- Formula object
- Sensor

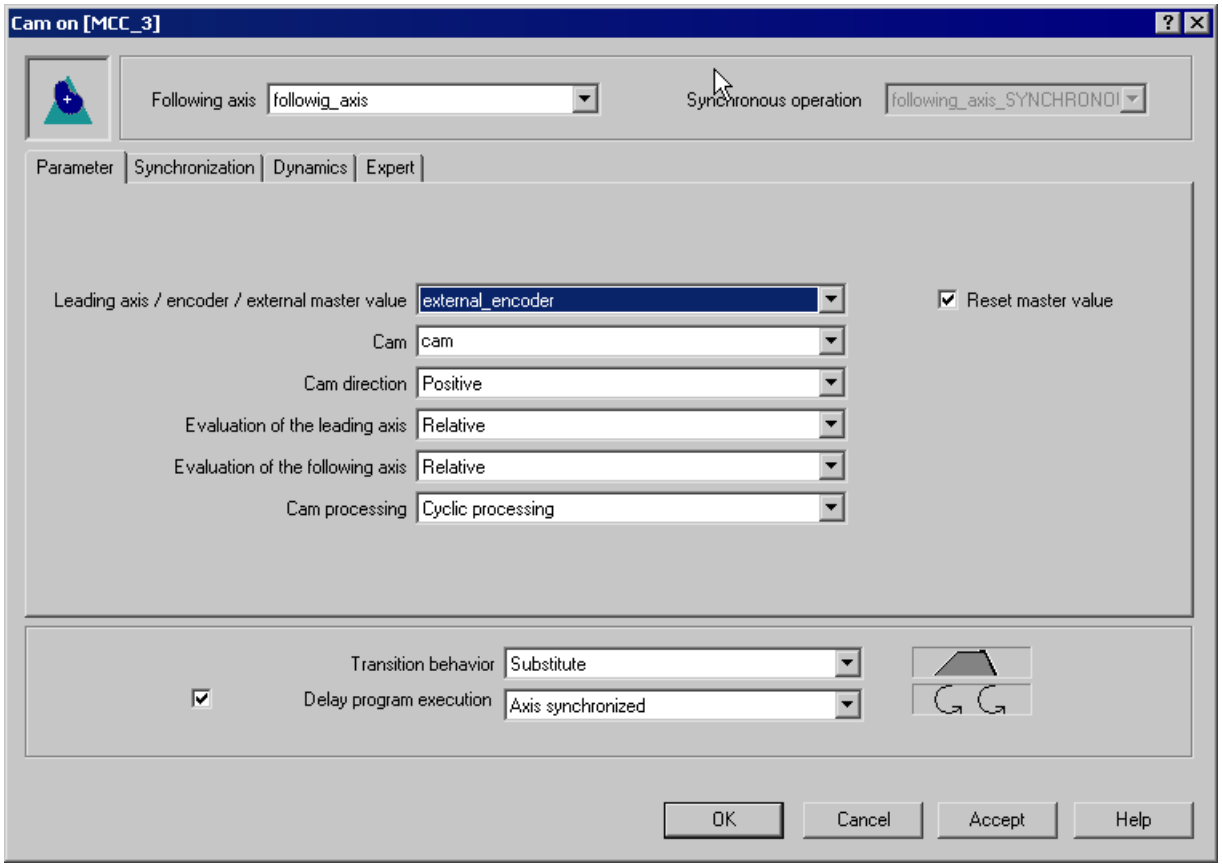


Figure 7-165 Parameter screen form: Cam on

Active camming can be scaled and offset both on the side of the master value and on the side of the slave value. This enables a cam to be adjusted individually in its definition and value range.

The following commands are used for this purpose.

- "Set scaling on camming" (Page 4442) and
- "Set offset on camming" (Page 4447)

In addition, the cam itself can be scaled and offset with the "Parameterize cam" (Page 4453) command.

Camming can be terminated:

- With the "Cam off" (Page 4435) command
- By starting another synchronous operation on the same synchronous object, e.g. with another "Cam on" command
- With a canceling single axis command on the following axis

Note

The "Stop axis" (Page 4281) command will only terminate the synchronous operation if **Quick stop...** has been selected as the stop mode.

Further information:

- For **general information on camming**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Cam on

Table 7-278 Overview of parameters for Cam on

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be synchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Cam on - Parameters tab (Page 4424)
Synchronization tab	See Overview of parameters for Cam on - Synchronization tab (Page 4427)

Field/button	Meaning/instruction
Dynamic response tab	See Overview of parameters for Cam on - Dynamic response tab (Page 4432)
Expert tab	See Overview of parameters for Cam on - Expert tab (Page 4433)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Cam on – Parameters tab

Table 7-279 Overview of parameters for Cam on – Parameters tab

Field/button	Meaning/instruction
Reset master value	Activate this checkbox if you want to set the master value of the synchronous relationship (default) If the checkbox is deactivated, a previous master value setting is retained. **Note** • As of Version V4.1 of the SIMOTION Kernel, the master value conversion takes place during the synchronization of the synchronous axis (in accordance with the transition behavior "with next synchronization" (Page 4461) with the command "Switch master value"). • From Version V4.0 of the SIMOTION Kernel, the master value conversion takes place immediately at the start of the command (in accordance with the transition behavior "direct" (Page 4461) with the command "Switch master value").
Leading axis / encoder / external master value	An entry can only be made in this field if the **Reset master value** checkbox is activated. Here, you select the axis or external encoder that generates the master value in the synchronous relationship. You can select from: • All position, following, and path axes and external encoders that are available on the device or a DP master. The following are also available as of Version V4.2 of SIMOTION Kernel: Drive axes, path objects, addition objects, controller objects, fixed gear, formula objects and sensors. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): PosAxis, FollowingAxis, _PathAxis and ExternalEncoderType. The following are also available as of Version V4.2 of SIMOTION Kernel: DriveAxis, _PathObjectType, _AdditionObjectType, _ControllerObjectType, _FixedGearType, _FormulaObjectType, _SensorType. The master value remains assigned to the synchronous object until a change occurs.

Field/button	Meaning/instruction
Cam	Here, select the cam that describes the synchronous relationship. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): CamType. **Note** In the latter case, you must set up the relationships (links) between the cam and the synchronous object of the slave axis in the project navigator. The MCC Editor does this for standard cases.
Cam direction	Here, you select the direction in which the cam is moved when master values are increasing. **Positive** The cam is executed in the **same direction** as the change in the master value. • If the change in the master value is positive, the cam is executed in the direction of increasing definition range values (to the right). • If the change in the master value is negative, the cam is executed in the direction of decreasing definition range values (to the left). **Negative** The cam is executed in the **opposite direction** as the change in the master value. • If the change in the master value is positive, the cam is executed in the direction of decreasing definition range values (to the left). • If the change in the master value is negative, the cam is executed in the direction of increasing definition range values (to the right). **Last programmed direction** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.direction
Evaluation of the leading axis	Here, you select whether the master value behaves absolutely or relatively with respect to the cam. **Absolute** The master value is applied as an absolute value in the definition range of the cam. **Relative** The master value is applied as a relative value (relative to the starting point of camming) in the definition range of the cam. Please note that the value for the **Offset to cam starting point** field (see the parameter overview for Cam on – Synchronization tab (Page 4427)) must be within the cam's definition range. **Last programmed value** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.cammingSettings.masterMode

Field/button	Meaning/instruction
Evaluation of the following axis	Here, you select whether the slave value behaves absolutely or relatively with respect to the cam. **Absolute** The slave value can be calculated as an absolute value from the value range of the cam. If the cam is processed cyclically (see the **Cam processing** field), the slave value begins with the initial value at each new cam cycle. **Relative** The slave value can be calculated as a relative value (relative to the starting point of camming) from the value range of the cam. If the cam is processed cyclically (see the **Cam processing** field), the slave value continues with the end value of the previous cam cycle at each new cam cycle. **Last programmed value** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.cammingSettings.slaveMode
Processing the cam	Here, you select whether the cam is to be processed cyclically. **Cyclic processing** The master values are calculated modulo to the cam length. If the master value reaches the end point of the definition range of the cam, the cam continues to be processed with the start value. **Non-cyclic processing** The master values are limited to the definition range of the cam; the cam is executed once in the definition range. If the master value reaches the starting point or end point of the definition range of the cam, the cam is no longer processed. When the master values are passed through again in the same direction (modulo axis), the slave values do not change. **Last programmed cam mode** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.CammingSettings.cammingMode

Overview of parameters for Cam on – Synchronization tab

Table 7-280 Overview of parameters for Cam on – Synchronization tab

Field/button	Meaning/instruction
Synchronization reference	Here, you select the reference for the synchronization operation. **Leading axis** Length-related synchronization: Synchronization occurs (depending on the leading axis motion) within a defined range of the master value (**Synchronization length**). • Advantage: Synchronization occurs within a parameterizable range of the master value. • Disadvantage: The dynamic response of synchronization is dependent on the change in the master value (velocity). The dynamic response limit values of the following axis are not taken into account. **Time** Time-related synchronization: Synchronization occurs on the basis of specified dynamic values. Program the relevant values in the Dynamic response tab, see Overview of parameters for Cam on - Dynamic response tab (Page 4432). • Advantage: The synchronization operation always takes place with the specified dynamic values. • Disadvantage: The master value range in which synchronization occurs cannot be predicted. **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncProfileReference

Field/button	Meaning/instruction
Start of synchronization	Here, you select when synchronization of the cam begins. Additional specifications are required for many of the choices; these are indicated in the description. **At leading axis position** Camming is enabled at a programmed leading axis position. Entries are required in the following fields: • Reference point of leading axis position • Leading axis position Synchronization occurs with respect to the slave values: • Master value = last programmed leading axis position • Slave value: Depending on the selection in **Evaluation of the following axis**, see Overview of parameters for Cam on – Parameters tab (Page 4424): – For **Absolute**: Slave value is calculated with the cam from the programmed leading axis position, taking into account: Evaluation of the leading axis Processing the cam If applicable, offset to cam starting point – For **Relative**: Slave value = current slave value **At leading axis position with offset** Camming is enabled at a programmed leading axis position. An offset for the following axis is additionally programmed. Entries are required in the following fields: • Offset of the following axis • Reference point of leading axis position • Leading axis position Synchronization occurs with respect to the slave values: • Master value = programmed leading axis position • Slave value: Depending on the selection in **Evaluation of the following axis**, see Overview of parameters for Cam on – Parameters tab (Page 4424): – For **Absolute**: Slave value = calculated slave value + programmed offset The calculated slave value is determined the same way as for the **At leading axis position** selection. – For **Relative**: Slave value = current slave value + programmed offset ... (Continued in next row of table)

Field/button	Meaning/instruction
Start of synchronization (continued)	... **Synchronize immediately** Camming is enabled immediately. Synchronization occurs with respect to the slave values: • Master value = current master value • Slave value: Depending on the selection in **Evaluation of the following axis**, see Overview of parameters for Cam on – Parameters tab (Page 4424): – For **Absolute**: Slave value is calculated with the cam from the current master value, taking into account: Evaluation of the leading axis Processing the cam If applicable, offset to cam starting point – For **Relative**: Slave value = current slave value **Synchronize immediately with offset** Camming is enabled immediately. An offset for the following axis is additionally programmed. An entry is required in the following field: • Offset of the following axis Synchronization occurs with respect to the slave values: • Master value = current master value • Slave value: Depending on the selection in **Evaluation of the following axis**, see Overview of parameters for Cam on – Parameters tab (Page 4424): – For **Absolute**: Slave value = calculated slave value + programmed offset The calculated slave value is determined the same way as for the **Synchronize immediately** selection. – For **Relative**: Slave value = current slave value + programmed offset ... (Continued in next row of table)

Field/button	Meaning/instruction
Start of synchronization (continued)	... **At end of cam cycle** This choice is available only for Evaluation of the leading axis as Relative. Camming is enabled when both of the following occur: 1. A camming operation of this synchronous object is already active with another cam 2. The master value of the active camming operation reaches the end of the cam or the cam cycle This enables a switch from one cam to another at a defined point in time. An entry is required in the following field: • Reference point of leading axis position Synchronization occurs with respect to the slave values: • Master value = master value at the end of the cam cycle • Slave value: Depending on the selection in **Evaluation of the following axis**, see Overview of parameters for Cam on – Parameters tab (Page 4424): – For **Absolute**: Slave value is calculated with the cam from the master value at the end of the cam cycle, taking into account: Processing the cam Offset in relation to cam starting point – For **Relative**: Slave value = current slave value **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.synchronizingMode
Offset of the following axis	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position with offset • Synchronize immediately with offset Here, you enter an offset that will be added to the calculated or current following axis position. See also input field (Page 4022).

Field/button	Meaning/instruction
Reference point of leading axis position	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position • At leading axis position with offset • At end of cam cycle Here, you select how the programmed position (see the Leading axis position field) acts relative to the selected synchronization profile. **Synchronize before synchronization position** Synchronization is finished at the programmed position. **Symmetrical** This choice is available only for leading axis synchronization reference. Synchronization occurs in such a way that the programmed position is located symmetrically within the synchronization length. At the programmed position, the leading axis has covered half the distance that is required for synchronization. **Synchronize from synchronization position** Synchronization starts at the programmed position. **Last programmed reference point of leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncPositionReference
Synchronization length	An entry is required in this field if you have selected the following in the **Synchronization reference** field: • Leading axis The synchronization length is the master value range in which synchronization occurs. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed synchronization length** **Default** See Selection list (combo box) (Page 4022). System variable for default: userdefault.syncProfile.syncLength
Starting point in the cam for relative camming	An entry is required in this field if you have selected the following in the **Evaluation of the leading axis** field (see Overview of parameters for Cam on – Parameters tab (Page 4424)): • Relative The cam starting point for relative camming specifies the starting point of the cam within its definition range. Enter the value in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed starting point** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.camStartPosition
Following axis position	Enter the following axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed following axis position** **Default** See Selection list (combo box) (Page 4022). System variable for default: userdefault.cammingSettings.syncPositionSlave

Field/button	Meaning/instruction
Leading axis position	An entry is required in this field if you have selected the following in the **Start of synchronization** field: • At leading axis position • At leading axis position with offset Enter the leading axis position in the editable selection list, see Editable selection list (editable combo box) (Page 4023). **Last programmed leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.syncPositionMaster
Synchronization direction	Here, you select the direction of motion in which the following axis is to be synchronized. **Retain system behavior** Synchronization occurs over the shortest path without direction specification. In the case of moving axes, a check is made to determine whether the current direction of motion can be maintained. **Maintain direction of the following axis** Synchronization takes place in the direction of motion of the following axis. **Positive** Synchronization occurs in the positive direction of motion. **Negative** Synchronization occurs in the negative direction of motion. **Shortest path** Synchronization occurs over the shortest path without direction specification. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.synchronizingDirection

Overview of parameters for Cam on – Dynamic response tab

Table 7-281 Overview of parameters for Cam on – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters on the Dynamic response tab are only evaluated for time synchronization reference.
Velocity	The entered value acts during the constant velocity phase. System variable for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Cam on – Expert tab

Table 7-282 Overview of parameters for Cam on – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Example of Cam on

An axis (rotaryAxis_1) is to rotate at a velocity of 360 degrees/s. A second axis (synchronous axis) is to be coupled to rotary axis_1 by means of a cam.

The cam must be created and parameterized in SIMOTION SCOUT. The synchronous axis that is to be coupled with rotary axis_1 by means of a cam must be assigned as a synchronous axis in SIMOTION SCOUT.

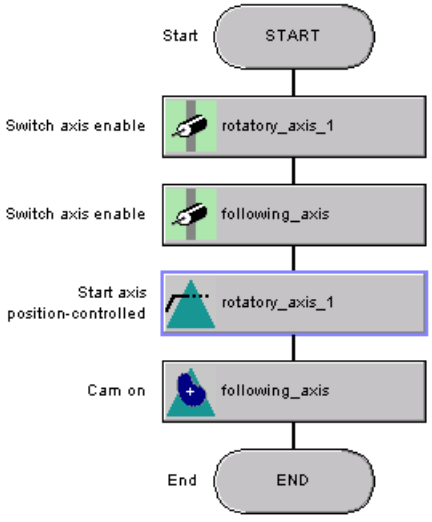


Figure 7-166 Example: MCC chart for a camming operation

Relevant system functions for Cam on

Cam technology package:

- _enableCamming
- _setMaster

If the **Reset master value** checkbox is activated, the _setMaster system function is called first before _enableCamming.

Overview of parameters for Cam on/_enableCamming, _setMaster

Table 7-283 Parameters (MCC command Cam on compared to system functions _enableCamming and _setMaster)

Parameters of the MCC command Cam on	**Parameters of the system functions _enableCamming and _setMaster**
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Reset master value	Call of _setMaster system function • Up to SIMOTION Kernel Version V4.0 with parameter transientBehavior = DIRECT • SIMOTION Kernel Version V4.1 and higher with parameter transientBehavior = WITH_NEXT_SYNCHRONIZING
Leading axis / encoder / external master value	master (system function _setMaster)
Cam	cam
Cam direction	direction
Evaluation of the leading axis	masterMode
Evaluation of the following axis	slaveRatioType
Processing the cam	cammingModer
Synchronization tab	
Synchronization reference	syncProfileReference
Start of synchronization	synchronizingMode
Offset of the following axis	syncPositionSlave
Reference point of leading axis position	syncPositionReference
Synchronization length	syncLengthType
Offset in relation to cam starting point	camStartPositionMasterType, camStartPositionMaster
Following axis position	syncPositionSlaveType, syncPositionSlave
Leading axis position	syncPositionMasterType, syncPositionMaster
Synchronization direction	synchronizingDirection
Dynamic response tab	
Velocity	velocityType, velocity

Parameters of the MCC command Cam on	Parameters of the system functions _enableCamming and _setMaster
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Cam off

This command enables you to terminate camming that was started with the "Cam on" (Page 4421) command. Desynchronization occurs with the programmed desynchronization settings.

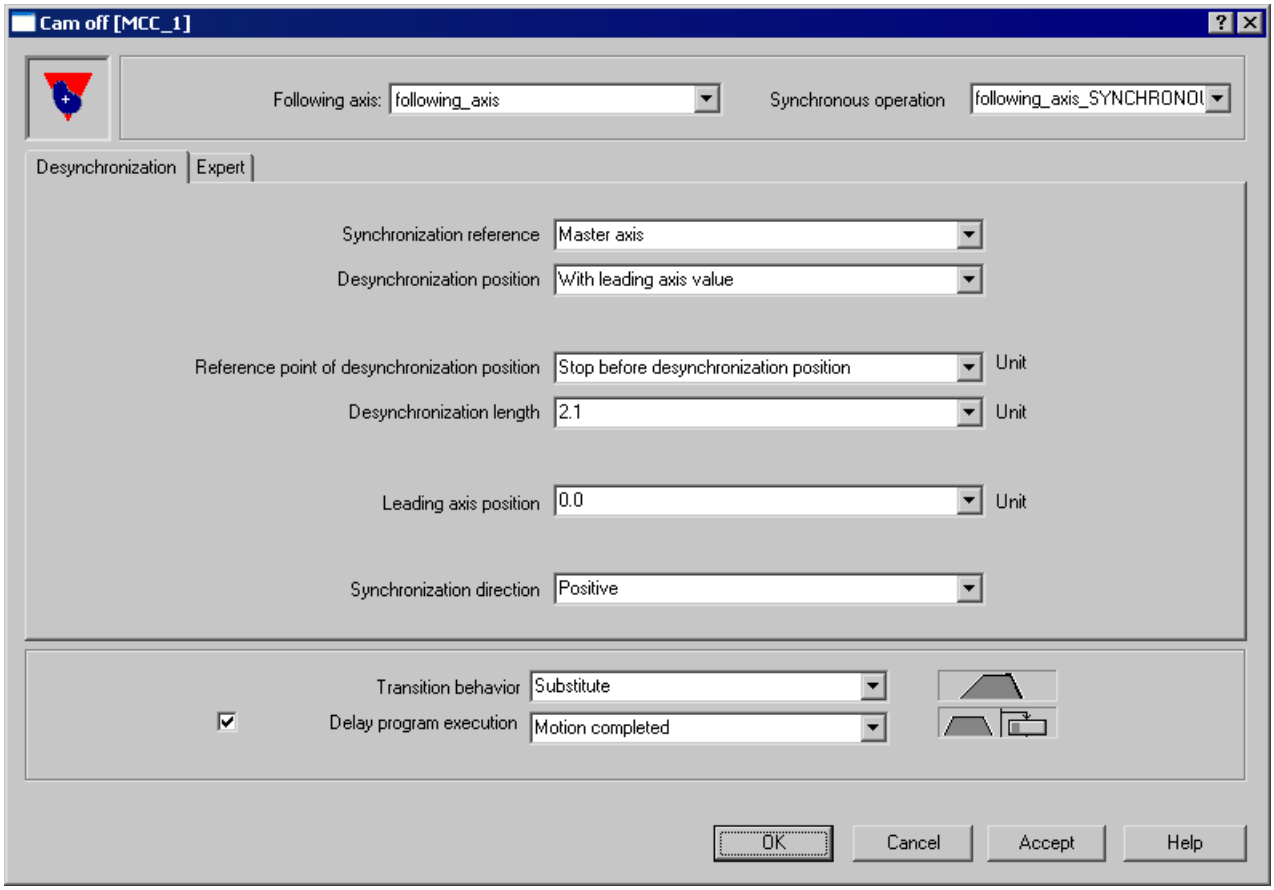


Figure 7-167 Parameter screen form: Cam off

Further information:

- For **general information on camming**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Cam off

Table 7-284 Overview of parameters for Cam off

Field/button	Meaning/instruction
Following axis	Here, you select the axis to be desynchronized. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be desynchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Following axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Desynchronization tab	See Overview of parameters for Cam off - Desynchronization tab (Page 4438)
Dynamic response tab	See Overview of parameters for Cam off - Dynamic response tab (Page 4440)
Expert tab	See Overview of parameters for Cam off - Expert tab (Page 4441)

Field/button	Meaning/instruction
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axis. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the command must be entered in the command buffer before the next command is executed. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Cam off – Desynchronization tab

Table 7-285 Overview of parameters for Cam off – Desynchronization tab

Field/button	Meaning/instruction
Synchronization reference	Here you select the reference for the desynchronization operation. **Leading axis** Length-related synchronization: Synchronization occurs (depending on the leading axis motion) within a defined range of the master value (**Desynchronization length**). • Advantage: Synchronization occurs within a parameterizable range of the master value. • Disadvantage: The dynamic response of synchronization is dependent on the change in the master value (velocity). The dynamic response limit values of the following axis are not taken into account. **Time** Time-related synchronization: Synchronization occurs on the basis of specified dynamic values. Program the relevant values in the Dynamic response tab, see Overview of parameters for Cam off - Dynamic response tab (Page 4440). • Advantage: The synchronization operation always takes place with the specified dynamic values. • Disadvantage: The master value range in which synchronization occurs cannot be predicted. **Last programmed setting** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncProfileReference
Desynchronization position	Here, you select when desynchronization of camming begins. Additional specifications are required for some of the choices; these are indicated in the description. **At leading axis value** Camming is disabled at a programmed leading axis position. Entries are required in the following fields: • Reference point of desynchronization position • Leading axis position **Desynchronize immediately** Camming is disabled immediately. **At following axis value** Camming is disabled at a programmed slave axis position. Entries are required in the following fields: • Reference point of desynchronization position • Following axis position **At end of cam cycle** Camming is disabled if the slave value reaches the end of the cam or the cam cycle. An entry is required in the following field: • Reference point of desynchronization position **Last programmed start of desynchronization** **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.syncOffMode

Field/button	Meaning/instruction
Reference point of desynchronization position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At leading axis value • At following axis value • At end of cam cycle Here, you select how the programmed position (see the Following axis position or Leading axis position field) acts relative to the selected desynchronization profile. **Stop before desynchronization position** Synchronization is finished at the programmed position. **Symmetrical** This choice is available only for leading axis synchronization reference. Desynchronization occurs in such a way that the programmed position is located symmetrically within the desynchronization length. At the programmed position, the leading axis has covered half the distance that is required for synchronization. **Stop from desynchronization position** Desynchronization starts at the programmed position. **Last programmed reference point of leading axis position** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncOffPositionReference
Desynchronization length	An entry is required in this field if you have selected the following in the **Synchronization reference** field: • Leading axis Enter the synchronization length in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Last programmed desynchronization length** **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncProfile.syncOffLength
Following axis position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At following axis value Enter the following axis position in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncOffPositions.Slave

Field/button	Meaning/instruction
Leading axis position	An entry is required in this field if you have selected the following in the **Desynchronization position** field: • At leading axis value Enter the leading axis position in the editable selection list (see Editable selection list (editable combo box) (Page 4023)). **Default** See Selection list (combo box) (Page 4022) System variable for default: userdefault.syncOffPositions.Master
Synchronization direction	Here, you select the direction of motion in which the following axis is to be desynchronized. **Retain system behavior** Synchronization occurs over the shortest path without direction specification. In the case of moving axes, a check is made to determine whether the current direction of motion can be maintained. **Maintain direction of the following axis** Synchronization takes place in the direction of motion of the following axis. **Positive** Synchronization occurs in the positive direction of motion. **Negative** Synchronization occurs in the negative direction of motion. **Shortest path** Synchronization occurs over the shortest path without direction specification. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userdefault.cammingSettings.synchronizingDirection

Overview of parameters for Cam off – Dynamic response tab

Table 7-286 Overview of parameters for Cam off – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters in the Dynamic response tab are evaluated only for time synchronization reference.
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variables for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Cam Off – Expert tab

Table 7-287 Overview of parameters for Cam off – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Cam off

Cam technology package:

- _disableCamming

Overview of parameters for Cam off / _disableCamming

Table 7-288 Parameters (MCC command Cam off compared to system function _disableCamming)

Parameters of the MCC command Cam off	Parameters of the system function _disableCamming
Following axis	–
Synchronous operation	followingObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Desynchronization tab	
Synchronization reference	syncProfileReference
Desynchronization position	syncOffMode
Reference point of desynchronization position	syncOffPositionReference
Desynchronization length	syncOffLengthType, syncOffLength
Following axis position	syncOffPositionSlaveType, syncOffPositionSlave
Leading axis position	syncOffPositionMasterType, syncOffPositionMaster

Parameters of the MCC command Cam off	Parameters of the system function _disableCamming
Synchronization direction	synchronizingDirection
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Set scaling on camming

This command causes scaling of the leading axis or following axis range of the associated cam during camming. You can use parameters to determine the following (among other things):

- Timing for when scaling becomes effective (immediately/next cam cycle)
- Effectiveness on the active camming command and/or subsequent camming commands

Scaling operations that act on the active camming command as well as scaling operations that are stored and act on subsequent commands can be read from the cammingAdjustment system variable.

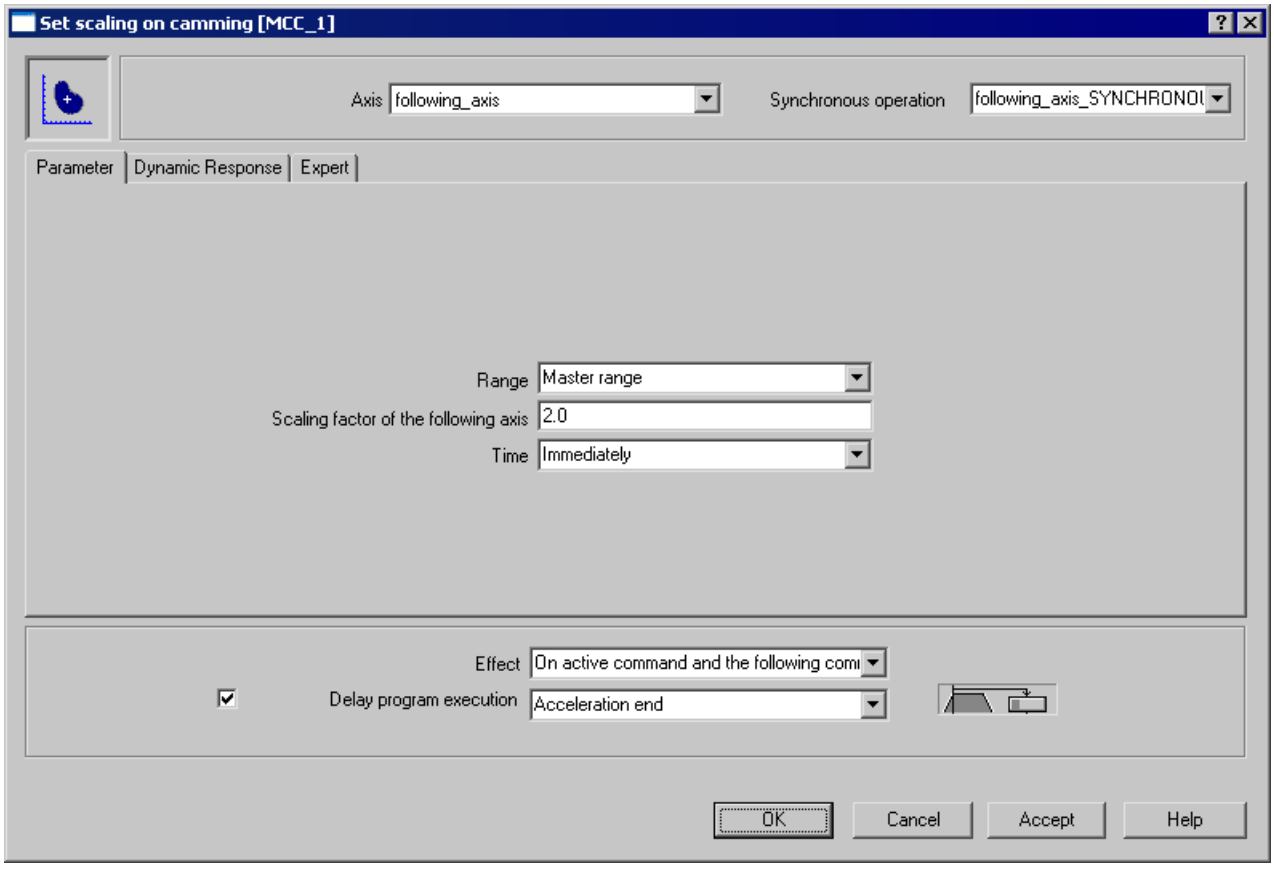


Figure 7-168 Parameter screen form: Set scaling on camming

Further information:

- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For information on **scaling and offset for a cam**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Set scaling on camming

Table 7-289 Overview of parameters for Set scaling on camming

Field/button	Meaning/instruction
Axis	Here, you select the axis for which the cam is to be scaled. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as an axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as the axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Set scaling on camming – Parameters tab (Page 4445)
Dynamic response tab	See Overview of parameters for Set scaling on camming – Dynamic response tab (Page 4446)
Expert tab	See Overview of parameters for Set scaling on camming – Expert tab (Page 4446)

Field/button	Meaning/instruction
Effect	Here you select the commands on which scaling is active. **On active command** (default value) Scaling acts on the active camming command and remains active until this command is replaced or desynchronized using the "Cam off" (Page 4435) command. The scaling factor is stored (depending on the cam range) in system variable cammingAdjustment.master.scale or cammingAdjustment.slave.scale. **on following commands** The active camming command is executed; the scaling is stored and used for the following "Cam on" (Page 4421) commands. The scaling factor is stored (depending on the cam range) in system variable cammingAdjustment.defaultValueMaster.scale or cammingAdjustment.defaultValueSlave.scale. **On active command and following commands** The scaling acts on the current camming command and on all following "Cam on" (Page 4421) commands. The scaling factor is stored (depending on the cam range) in system variables cammingAdjustment.master.scale and cammingAdjustment.defaultValueMaster.scale, or cammingAdjustment.slave.scale and cammingAdjustment.defaultValueSlave.scale.
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Set scaling on camming – Parameters tab

Table 7-290 Overview of parameters for Set scaling on camming – Parameters tab

Field/button	Meaning/instruction
Range	Here, you select whether the leading axis or following axis range of the cam is to be scaled. **Master range** (default value) The leading axis range of the cam is scaled. **Slave range** The following axis range of the cam is scaled.
Offset	Scaling factor for the selected range. Enter the value as a signed floating-point number. See also input field (Page 4022).
Time	Here, you select when the scaling takes effect. **Immediately** (default value) The scaling of the selected range goes into effect immediately. **At next cycle** The scaling of the selected range goes into effect at the next cam cycle (only for cyclic cam).

Overview of parameters for Set scaling on camming – Dynamic response tab

Table 7-291 Overview of parameters for Set scaling on camming – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variables for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Set scaling on camming – Expert tab

Table 7-292 Overview of parameters for Set scaling on camming – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Set scaling on camming

Cam technology package:

- _setCammingScale

Overview of parameters for Set scaling on camming / _setCammingScale

Table 7-293 Parameters (MCC command Set scaling on camming compared to system function _setCammingScale)

Parameters of the MCC command Set scaling on camming	Parameters of the system function _setCammingScale
Axis	–
Synchronous operation	followingObject
Effect	activationMode
Delay program execution	nextCommand
Parameters tab	
Range	scalingRange
Offset	scaleValue
Time	scaleSpecification
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Set offset on camming

This command causes an offset of the leading axis or following axis range of the associated cam during camming. You can use parameters to determine the following (among other things):

- Timing for when the offset takes effect (immediately/next cam cycle)
- Effectiveness on the active camming command and/or subsequent camming commands

Offsets that act on the active camming command and offsets that are stored and act on subsequent camming commands can be read from the cammingAdjustment system variable.

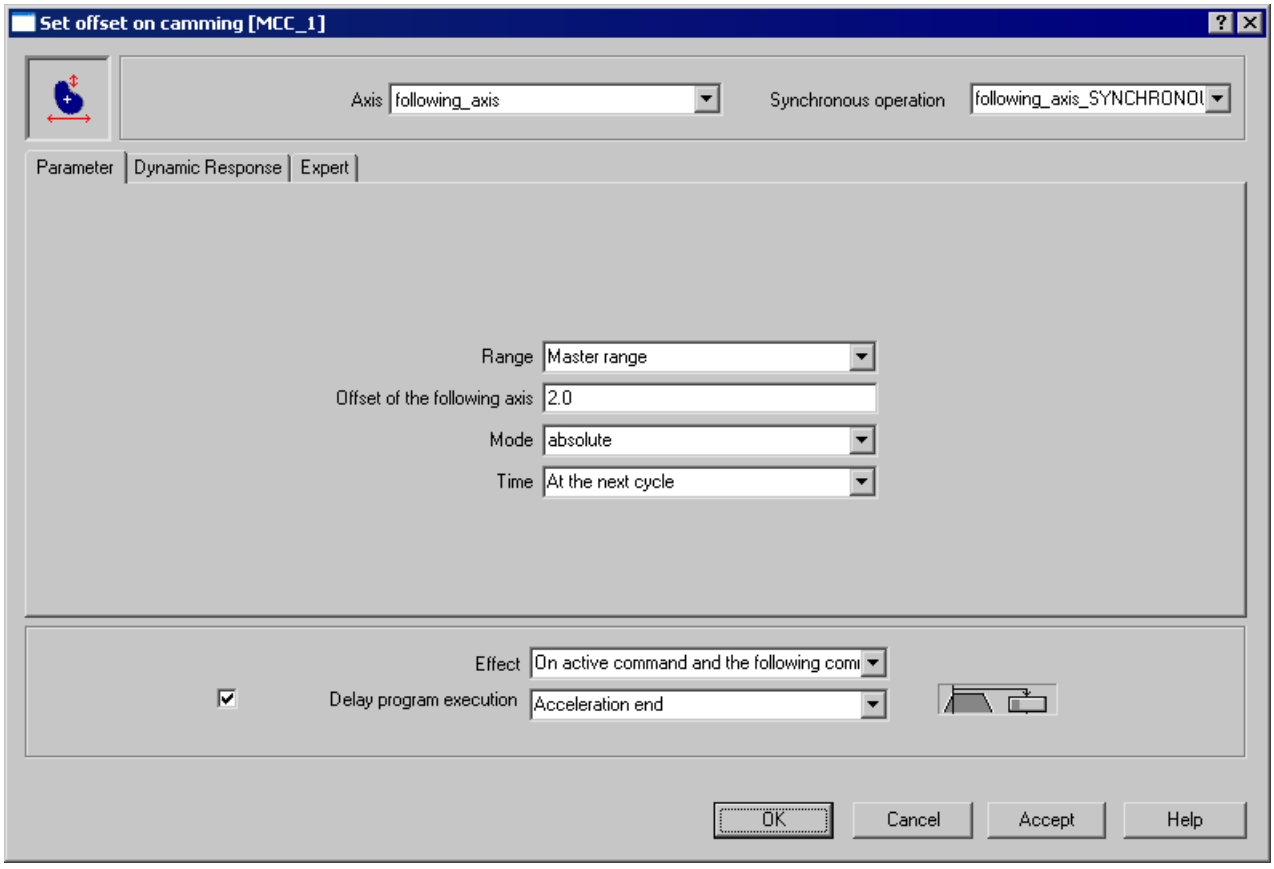


Figure 7-169 Parameter screen form: Set offset on camming

Further information:

- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For information on **scaling and offset for a cam**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Set offset on camming

Table 7-294 Overview of parameters for Set offset on camming

Field/button	Meaning/instruction
Axis	Here, you select the axis for which the cam is to be offset. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected **Axis** and are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as an axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <**Reference**> has been selected as the axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Set offset on camming – Parameters tab (Page 4450)
Dynamic response tab	See Overview of parameters for Set offset on camming – Dynamic response tab (Page 4451)
Expert tab	See Overview of parameters for Set offset on camming – Expert tab (Page 4452)

Field/button	Meaning/instruction
Effect	Here, you select which commands the offset will act on. **On active command** (default value) The offset acts on the active camming command and remains active until this command is replaced or desynchronized using the "Cam off" (Page 4435) command. The offset value is stored in system variable cammingAdjustment.master.offset or cammingAdjustment.slave.offset (depending on the range of the cam). **on following commands** The active camming command is executed; the offset is stored and used for the following "Cam on" (Page 4421) commands. The offset value is stored (depending on the cam range) in system variable cammingAdjustment.defaultValueMaster.offset or cammingAdjustment.defaultValueSlave.offset. **On active command and following commands** The offset acts on the current camming command and on all following "Cam on" (Page 4421) commands. The offset value is stored in system variables cammingAdjustment.master.offset and cammingAdjustment.defaultValueMaster.offset or cammingAdjustment.slave.offset and cammingAdjustment.defaultValueSlave.offset (depending on the range of the cam).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037)

Overview of parameters for Set offset on camming – Parameters tab

Table 7-295 Overview of parameters for Set offset on camming – Parameters tab

Field/button	Meaning/instruction
Range	Here, you select whether the leading axis or following axis range of the cam will be offset. **Master range** (default value) The leading axis range of the cam is offset. **Slave range** The following axis range of the cam is offset.
Offset	Offset for the selected range. Enter the value as a signed floating-point number. See also input field (Page 4022).

Field/button	Meaning/instruction
Mode	Select the type of offset. **Absolute**(default value) The offset is applied as an absolute value. **Relative** The offset is relative to the corresponding value in the cammingAdjustments system variables.
Time	Here, you select when the offset takes effect **Immediately** (default value) The offset of the selected range goes into effect immediately. **At next cycle** The offset of the selected range takes effect at the next cam cycle (for cyclic cam only).

Overview of parameters for Set offset on camming – Dynamic response tab

Table 7-296 Overview of parameters for Set offset on camming – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031).
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variables for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Set offset on camming – Expert tab

Table 7-297 Overview of parameters for Set offset on camming – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Set offset on camming

Cam technology package:

- _setCammingOffset

Overview of parameters for Set offset on camming / _setCammingOffset

Table 7-298 Parameters (MCC command Set offset on camming compared to system function _setCammingOffset)

Parameters of the MCC command Set offset on camming	Parameters of the system function _setCammingOffset
Axis	–
Synchronous operation	followingObject
Effect	activationMode
Delay program execution	nextCommand
Parameters tab	
Range	offsetRange
Offset	offsetValue
Mode	offsetMode
Time	offsetSpecification
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Parameterize cam

This command enables a defined electronic cam to be scaled and/or offset.

The master and slave axes can be offset and scaled independently of one another.

Scaling takes place over the complete cam or within a range defined by the starting and end points. Scaling of the complete cam is referred to below as basic scaling; scaling of a range is referred to as range scaling. The definition range and value range each have one basic scaling and two range scalings.

For basic scaling, the zero point of the coordinate axis becomes the scaling point, while for range scaling the starting point of the specified scaling range is used.

This command enables simultaneous specification of one offset and/or basic scaling plus one range scaling each for the master and slave axes.

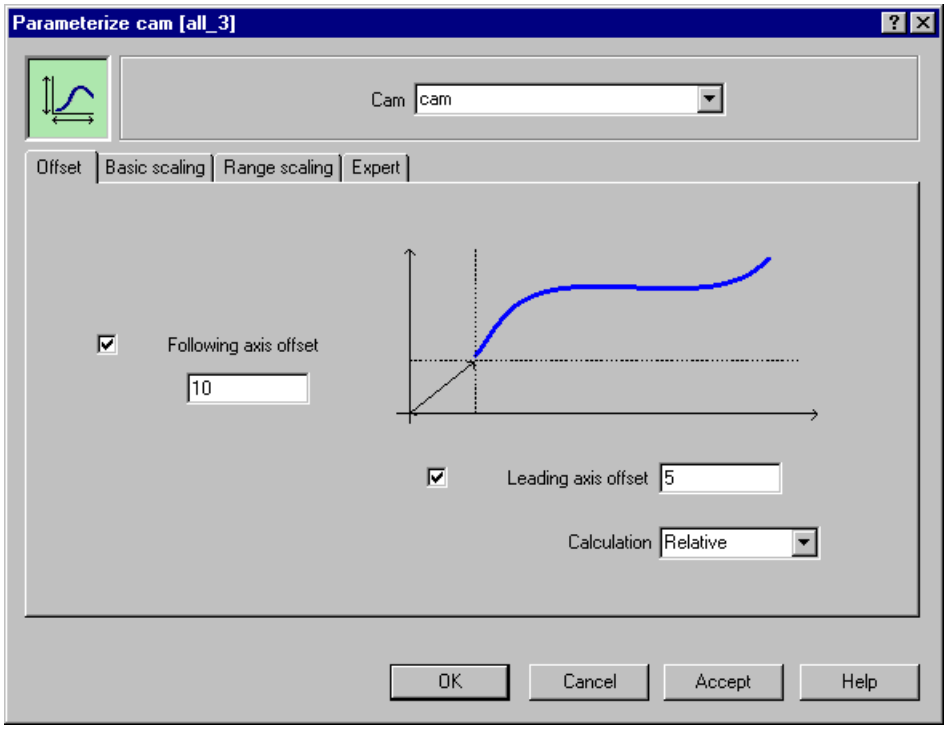


Figure 7-170 Parameter screen form: Parameterize cam for the Offset tab

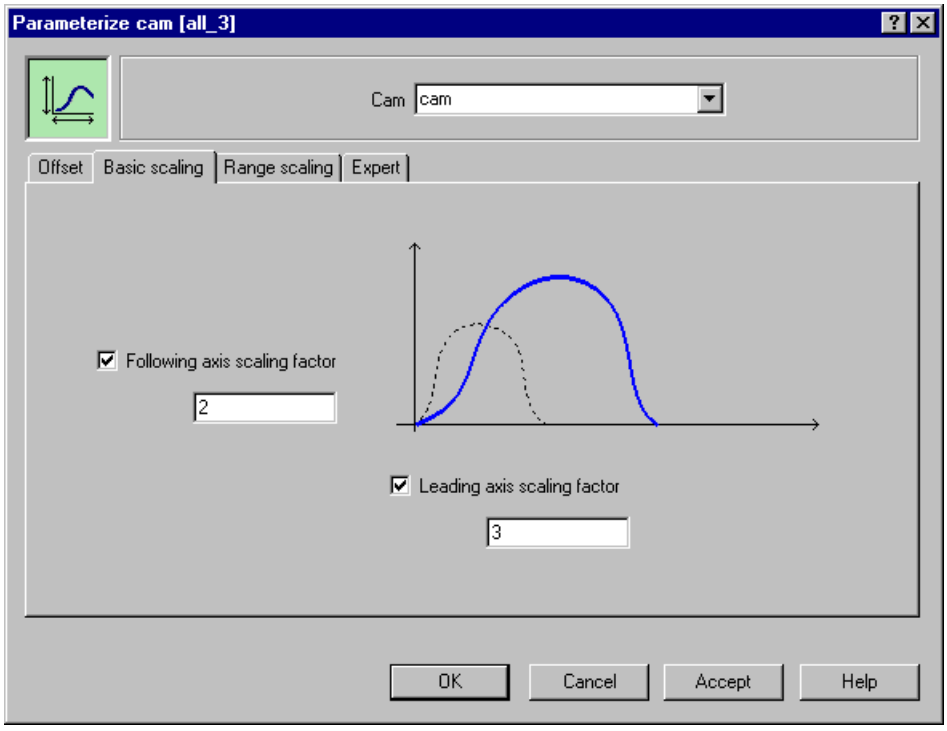


Figure 7-171 Parameter screen form: Parameterize cam for the Basic scaling tab

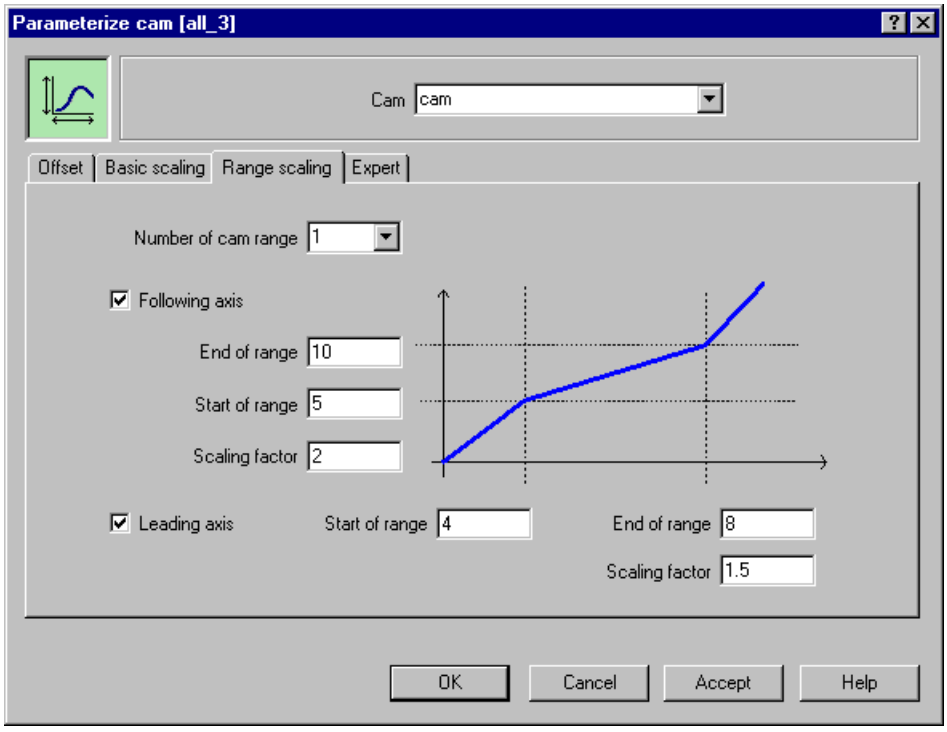


Figure 7-172 Parameter screen form: Parameterize cam for the Range scaling tab

Further information:

- For **cam fundamentals**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For information on **scaling and offset for a cam**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Parameterize cam

Table 7-299 Overview of parameters for Parameterize cam

Field/button	Meaning/instruction
Cam	Here, you select the cam to be offset or scaled. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): CamType.
Offset tab	See Overview of parameters for Parameterize cam – Offset tab (Page 4455)
Basic scaling tab	See Overview of parameters for Parameterize cam – Basic scaling tab (Page 4456)
Range scaling tab	See Overview of parameters for Parameterize cam – Range scaling tab (Page 4456)
Expert tab	See Overview of parameters for Parameterize cam – Expert tab (Page 4457)

Overview of parameters for Parameterize cam – Offset tab

Table 7-300 Overview of parameters for Parameterize cam – Offset tab

Field/button	Meaning/instruction
Following axis offset	Select this checkbox if you want to program an offset for the following axis. Enter the offset for the following axis here. See also input field (Page 4022).
Leading axis offset	Select this checkbox if you want to program an offset for the leading axis. Enter the leading axis offset in this box. See also input field (Page 4022).
Calculation	Here, you select whether the newly programmed offset is applied absolutely or is added to the previously active offset. **Absolute**(default value) The programmed value is applied absolutely. **Relative** The programmed value is applied relatively.

Overview of parameters for Parameterize cam – Basic scaling tab

Table 7-301 Overview of parameters for Parameterize cam – Basic scaling tab

Field/button	Meaning/instruction
Following axis scaling factor	Select this checkbox if you wish to program a scaling factor for the complete following axis range. Here, you enter a scaling factor that is valid for the following axis, independent of the range. Using the range scaling (see Overview of parameters for Parameterize cam – Range scaling tab (Page 4456)), you can also activate scaling factors for individual ranges. These are superimposed over the complete scaling. See also input field (Page 4022).
Leading axis scaling factor	Select this checkbox if you want to program a scaling factor for the complete range of the leading axis. Here, you enter a scaling factor that is valid for the leading axis, independent of the range. Using the range scaling (see Overview of parameters for Parameterize cam – Range scaling tab (Page 4456)), you can also activate scaling factors for individual ranges. These are superimposed over the complete scaling. See also input field (Page 4022).

Overview of parameters for Parameterize cam – Range scaling tab

Table 7-302 Overview of parameters for Parameterize cam – Range scaling tab

Field/button	Meaning/instruction
Number of cam range	Here, you select the number of the scaling range (1 or 2). If you select the same number for a subsequent command, you can overwrite the previously programmed values.
Following axis	Activate this checkbox in order to program a scaling factor in a specific range of the following axis.
Start of range (following axis)	Here, you enter the following axis position at which scaling starts. See also input field (Page 4022).
End of range (following axis)	Here, you enter the following axis position at which scaling ends. See also input field (Page 4022).
Scaling factor (following axis)	Here, you enter the scaling factor for the following axis. This is effective within the specified range. See also input field (Page 4022).
Leading axis	Activate this checkbox to program a scaling factor in a specific range of the leading axis.
Start of range (leading axis)	Here, you enter the position of the leading axis at which scaling begins. See also input field (Page 4022).
End of range (leading axis)	Here, you enter the position of the leading axis at which scaling ends. See also input field (Page 4022).
Scaling factor (leading axis)	Here, you enter the scaling factor for the leading axis. This is effective within the specified range. See also input field (Page 4022).

Overview of parameters for Parameterize cam – Expert tab

Table 7-303 Overview of parameters for Parameterize cam – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
Return variable	If you enter the name of a variable of the specified data type for each command step, you can use this variable to find out the result of the command step. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Parameterize cam

Cam technology package:

- _setCamOffset
- _setCamScale

Overview of parameters for Parameterize cam / _setCamOffset, _setCamScale

Table 7-304 Parameters (MCC command Parameterize cam compared to system functions _setCamOffset and _setCamScale)

Parameters of the MCC command Parameterize cam	Parameters of the system functions _setCammingOffset and _setCamScale
Cam	cam
Offset tab	
Following axis offset	Call of system function _setCamOffset. offsetRange, offset
Leading axis offset	Call of system function _setCamOffset. offsetRange, offset
Calculation	offsetMode
Basic scaling tab	
Following axis scaling factor	Call of system function setCamScale. scalingRange, scalingSpecification, scaleValue
Leading axis scaling factor	Call of system function setCamScale. scalingRange, scalingSpecification, scaleValue
Range scaling tab	
Number of cam range	specificRangeNumber
Following axis	Call of system function setCamScale
Start of range (following axis)	specificRangeStartPointType, specificRangeStartPoint
End of range (following axis)	specificRangeEndPointType, specificRangeEndPoint
Scaling factor (following axis)	scalingRange, scalingSpecification, scaleValue
Leading axis	Call of system function setCamScale
Start of range (leading axis)	specificRangeStartPointType, specificRangeStartPoint
End of range (leading axis)	specificRangeEndPointType, specificRangeEndPoint
Scaling factor (leading axis)	scalingRange, scalingSpecification, scaleValue

Parameters of the MCC command Parameterize cam	Parameters of the system functions _setCammingOffset and _setCamScale
Expert tab	
Return variable	–

Switch master setpoint

This command switches over the master value for an existing synchronous operation relationship. Switchover is possible when synchronous operation is active.

Other master value sources for SIMOTION devices as of Version V4.2 of SIMOTION Kernel only

As of Version V4.2 of SIMOTION Kernel, the instances of the following technology objects and the variables of the relevant technology object data type are available as master value sources:

- Drive axis
- Path object
- Addition object
- Controller object
- Fixed gear
- Formula object
- Sensor

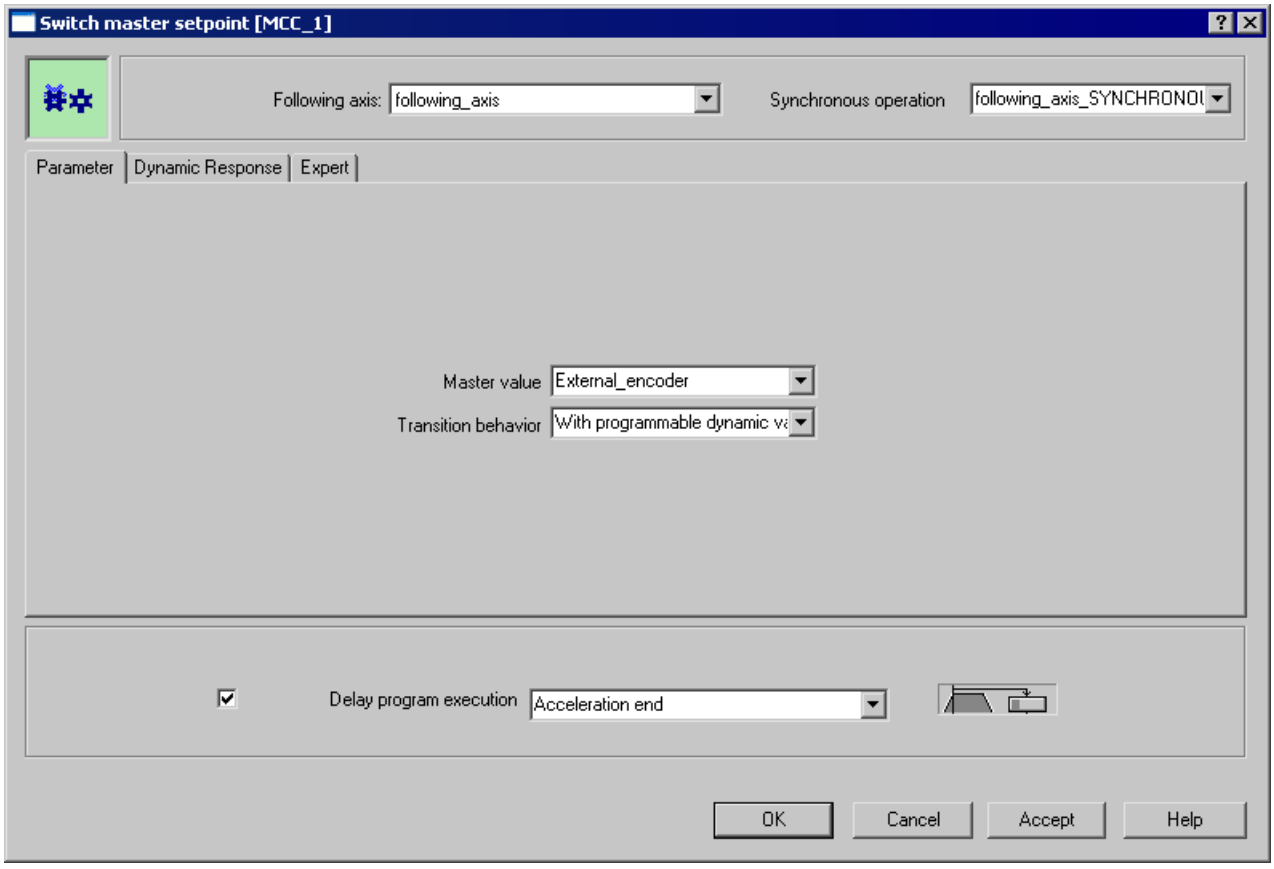


Figure 7-173 Parameter screen form: Switch master value

Further information:

- For **general information on switching over the master value source**, see the Technology Objects Synchronous Operation, Cam Function Manual
- For the **fundamentals of synchronous operation**, see the Technology Objects Synchronous Operation, Cam Function Manual

Overview of parameters for Switch master setpoint

Table 7-305 Overview of parameters for Switch master value

Field/button	Meaning/instruction
Following axis	Here, you select the axis for which the master value is to be switched. The following are available: • All synchronous axes that are defined on the relevant device. The axes are defined in the AXES folder in the project navigator. The synchronous objects associated with the selected synchronous axis are automatically identified and displayed in the **Synchronous operation** field for selection, as appropriate. • <**Reference**> You select this entry if the axis to be synchronized is not defined on the device but rather is specified as a reference (variable). All variables declared in the MCC unit or MCC chart with data type FollowingObjectType (references to synchronous objects) are available for selection in the **Synchronous operation** field, see also Technology object data types (Page 4060). **Note** You cannot select any references to a synchronous axis (variable of FollowingAxis data type). There is no assignment for the reference to the associated synchronous objects. Instead, select the reference to the synchronous object directly (variable of data type FollowingObjectType)
Synchronous operation	Here, the permissible synchronous objects are displayed according to the selected following axis, where they are available for selection, as appropriate: • A synchronous axis defined on the device has been selected as a following axis: The synchronous object associated with the selected following axis is displayed If more than one synchronous object is available (e.g. superimposed synchronous object), you can select the synchronous object. • <Reference> has been selected as a following axis: All technology object-type variables declared in the MCC unit or MCC chart (see Technology object data types (Page 4060)) are available for selection: FollowingObjectType. These variables are references to synchronous objects.
Parameters tab	See Overview of parameters for Switch master value – Parameters tab (Page 4461)
Dynamic response tab	See Overview of parameters for Switch master value – Dynamic response tab (Page 4461)
Expert tab	See Overview of parameters for Switch master value – Expert tab (Page 4462)
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Switch master setpoint – Parameters tab

Table 7-306 Overview of parameters for Switch master value – Parameters tab

Field/button	Meaning/instruction
Master value	Here, you select the axis or external encoder that generates the master value in the synchronous relationship. You can select from: • All position, following, and path axes and external encoders that are available on the device or a DP master. The following are also available as of Version V4.2 of SIMOTION Kernel: Drive axes, path objects, addition objects, controller objects, fixed gear, formula objects and sensors. • All variables with the following technology object data types declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): PosAxis, FollowingAxis, _PathAxis or ExternalEncoderType. The following are also available as of Version V4.2 of SIMOTION Kernel: DriveAxis, _PathObjectType, _AdditionObjectType, _ControllerObjectType, _FixedGearType, _FormulaObjectType, _SensorType. The master value remains assigned to the synchronous object until a change occurs.
Transition behavior	Here, you choose how the switchover to the selected master value is to occur. **direct** Selected master value is active immediately **With next synchronization** Selected master value is active the next time the following axis is synchronized. **With programmed dynamic values** The transition to the selected master value takes place with the values set in the Dynamic response tab (see Overview of parameters for Switch master value – Dynamic response tab (Page 4461)).

Overview of parameters for Switch master setpoint – Dynamic response tab

Table 7-307 Overview of parameters for Switch master value – Dynamic response tab

Field/button	Meaning/instruction
	The Dynamic response tab is described in detail in Dynamic response tab (Page 4031). The parameters on the Dynamic response tab are only evaluated with programmed dynamic values during transition behavior.
Velocity	The entered value acts during the constant velocity phase. System variables for default: userDefault.syncDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. System variables for default: userDefault.syncDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variable for default: userDefault.syncDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used, see the general description of the Dynamic response tab (Page 4031). System variables for default: userDefault.syncDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk userDefault.syncDynamics.negativeAccelEndJerk

Overview of parameters for Switch master setpoint – Expert tab

Table 7-308 Overview of parameters for Switch master value – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Switch master setpoint

Cam technology package:

- _setMaster

Overview of parameters for Switch master value / _setMaster

Table 7-309 Parameters (MCC command Switch master value compared to system function _setMaster)

Parameters of the MCC command Switch master value	Parameters of the system function _setMaster
Following axis	–
Synchronous operation	followingObject
Delay program execution	nextCommand
Parameters tab	
Master value	master
Transition behavior	transientBehavior
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel

Parameters of the MCC command Switch master value	Parameters of the system function _setMaster
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

7.1.6.8 Commands for path interpolation

As of version V4.1 of the SIMOTION Kernel, SIMOTION provides path interpolation functions. This functionality enables up to 3 path axes to travel along specified paths. In addition, a position axis can be traversed synchronously with the path.

The **Path object** technology object (data type _PathObjectType) provides this functionality.

- With the Path object technology object, the following additional technology objects are interconnected during configuration:
  - At least 2 and up to 3 axes with the path interpolation technology (**path axes** - data type _PathAxis)
  - Optionally, 1 position axis (data type PosAxis, FollowingAxis or _PathAxis) that is traversed synchronously with the path motion. This facilitates, for example, the synchronous method of this axis to a preset endpoint (or by a predetermined travel) or the output of the actual length of the path motion on this axis.
  - Optionally, 1 cam (data type CamType) in order to specify a special velocity profile for the path motion
  - SIMOTION Kernel as of version V4.4:
    Optional 1 other position axis (data type PosAxis or FollowingAxis or _PathAxis) to output the current length of the path movement if the path-synchronous axis is moved to a preset endpoint (or by a predetermined travel).
- In addition, the motion inputs (MotionIn) of position axes can be interconnected with the 3 motion outputs (MotionOut) of the TO path object. This enables outputting of the 3 coordinates of the basic coordinate system on axes.

- The programming of the path motions is either:
  - In the basic coordinate system (BCS)
    The basic coordinate system is a static, right-handed, rectangular (Cartesian) coordinate system in accordance with DIN 66217
  - In an object coordinate system (OCS - as of version V4.3 of the SIMOTION Kernel)
    An object coordinate system is a coordinate system defined on the path object. It is defined with the system function _setPathObjectOcs. For this purpose, the translation (offset) along the 3 axes and the rotation around the 3 axes must be specified with regard to the basic coordinate system. In addition, it can also be coupled in the x direction synchronously to another motion (e.g. an axis).
    Up to 3 object coordinate systems can be defined on the path object.
- Adaptation of machine kinematics (i.e. axis coordinate system) to the basic coordinate system is done by means of kinematics transformations. You specify this when configuring the TO path object.

The following applies to the path motion:

- The paths can comprise individual path segments.
- These individual segments can be the following 2-dimensional or 3-dimensional motions (interpolations):
  - Linear motion: Traverse path linearly (Page 4464) command
  - Circular motion: Traverse path circularly (Page 4477) command
  - Polynomial motion: Traverse path using polynomials (Page 4491) command
- Blending is used to link individual segments in series to form one complete motion with no intermediary stop.
  As of version V4.3 of the SIMOTION Kernel, optionally a transition geometry (blending segment) can be inserted.
- The following applies for the dynamic response planning:
  - The following applies as of version V4.3 of the SIMOTION Kernel:
    The dynamic response planning is generally performed over 3 successive path segments. A transition geometry (blending segment) is also counted.
  - The following applies up to version V4.2 of the SIMOTION Kernel:
    The dynamic response planning is generally performed over 2 successive path segments.

Traverse path linearly

This function is available in SIMOTION Kernel as of Version 4.1.

This command moves the path axes interconnected with the specified path object on a straight line to an end point. The motion can be carried out in:

- Three dimensions
- Two dimensions in one of the main Cartesian planes

The end point entered can be absolute or relative.

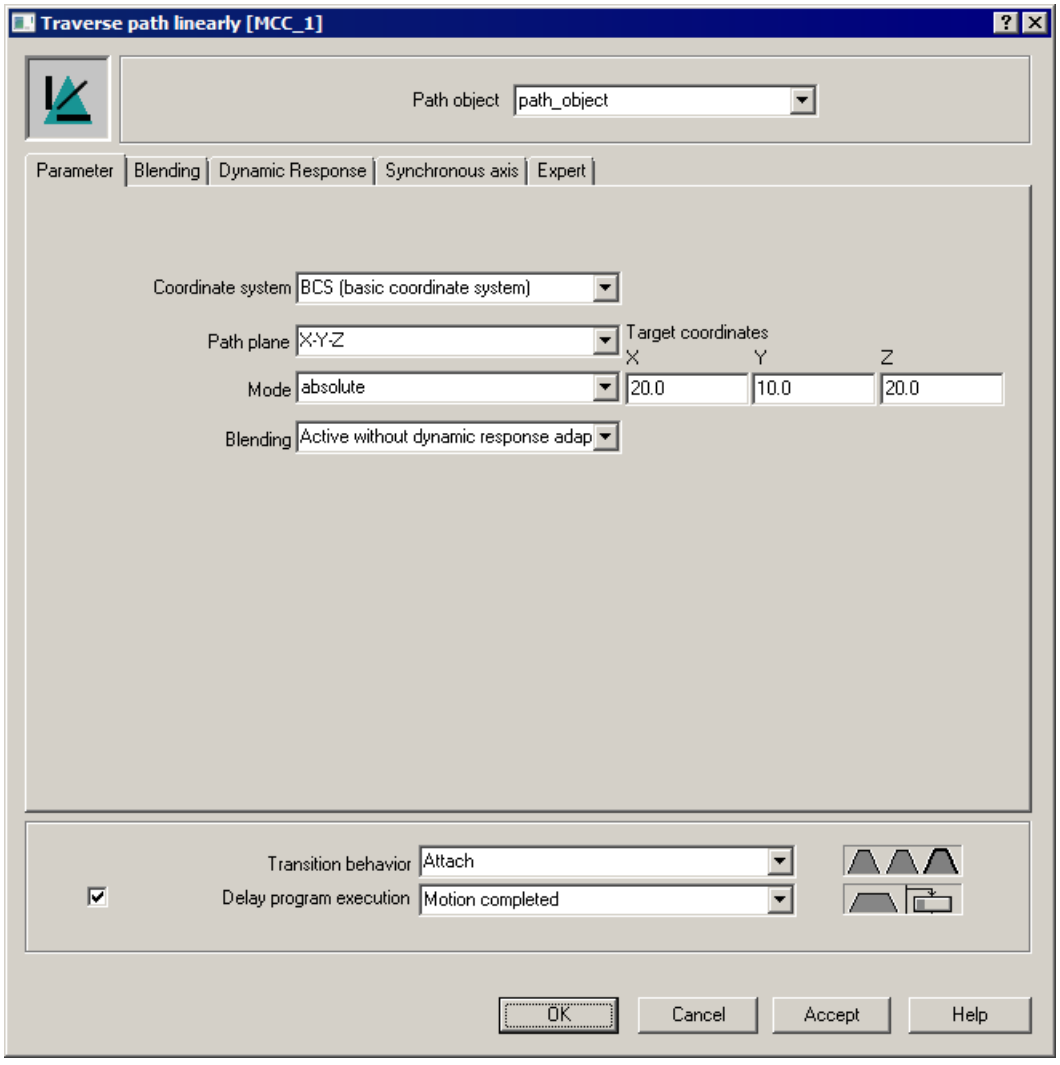


Figure 7-174 Parameter screen form: Traverse path linearly

For the **fundamentals of path interpolation**, see the TO Path Object Function Manual

For further information on **linear paths**, see the TO Path Object Function Manual.

Overview of parameters for Traverse path linearly

Table 7-310 Overview of parameters for Traverse path linearly

Field/button	Meaning/instruction
Path object	In Path object, select the path object whose assigned path axes are to be traversed to the end point. The list contains: • All path objects that are defined on the relevant device. The path objects are defined in the PATH OBJECTS folder in the project navigator. • All variables with the following data type of a technology object declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): _PathObjectType.
Parameters tab	See Overview of parameters for Traverse path linearly - Parameters tab (Page 4466)

Field/button	Meaning/instruction
Blending tab	See Overview of parameters for Traverse path linearly - Blending tab (Page 4469)
Dynamic response tab	See Overview of parameters for Traverse path linearly - Dynamic response tab (Page 4471)
Synchronous axis tab	See Overview of parameters for Traverse path linearly - Path-synchronous motion tab (Page 4474)
Expert tab	See Overview of parameters for Traverse path linearly - Expert tab (Page 4475)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the path object or on the axes involved. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Traverse path linearly – Parameters tab

Table 7-311 Overview of parameters for Traverse path linearly – Parameters tab

Field/button	Meaning/instruction
Coordinate system	This field is available as of SIMOTION Kernel version V4.3. Here you select the coordinate system to which the path motion to be programmed refers. **BCS (basic coordinate system)** (default value) Static, right-handed, rectangular (Cartesian) coordinate system in accordance with DIN 66217. **OCS (object coordinate system)** Coordinate system defined on the path object that is offset and rotated with regard to the basic coordinate system. It can also be coupled synchronously to another motion (e.g. an axis). Up to 3 object coordinate systems can be defined.
Number	This field is available as of SIMOTION Kernel version V4.3. An entry is required in this field if you have selected the following in the **Coordinate system** field: • OCS (object coordinate system) Select the number of the object coordinate system to be used. Possible values are 1 to 3. **1** (default value) Object coordinate system 1

Field/button	Meaning/instruction
Path plane	Here, you select whether the motion is to be carried out in 3 dimensions (3D) or 2 dimensions (2D). **X-Y-Z** Three-dimensional motion **X-Y main plane** **Y-Z main plane** **Z-X main plane** 2-dimensional motion in the selected main plane of the Cartesian basic coordinate system. The target coordinate located outside the selected plane is ignored. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.plane
Mode	The meaning of the subsequently programmed target coordinates is defined in more detail here. **Absolute** The programmed target coordinates refer to the zero point of the basic coordinate system. **Relative** The programmed target coordinates refer to the path position within the basic coordinate system when the path command is executed (point in time of change). **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.mode
Target coordinates X, Y, Z	The end point of the motion is specified here. The meaning depends on the **Mode** parameter (see above)**:** • Absolute mode: Absolute coordinates with respect to the zero point of the basic coordinate system. • Relative mode: Relative coordinates with respect to the path position within the basic coordinate system when the path command is executed (point in time of change). Enter the values as signed floating-point numbers. See the input field (Page 4022).

Field/button	Meaning/instruction
Blending	Using blending, the path segment programmed with this command can be linked to the previous path segment to form one complete path with no stop. At the transition, the system provides for a constant path velocity and a constant path acceleration. As of version V4.3 of the SIMOTION Kernel, a transition geometry (blending segment) can also be inserted. **Active with dynamic response adaptation** The dynamic response limit values of the individual path axes are also applied in the blending area or blending segment. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. A blending segment is not inserted. The path velocity is nearly reduced to zero at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4469) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel. **Active without dynamic response adaptation** In the blending area or blending segment, path interpolation takes into account only the scalar dynamic response limit values of the path (path velocity, path acceleration and jerk). The dynamic response limit values specific to the axes are first taken into account during movement of the individual path axes. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. Axis-specific deviations from the path can occur at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4469) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel.
Blending (continued)	**Inactive** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The motion with its dynamic response parameters programmed with this command immediately replaces the previous path motion. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.blending.mode

Overview of parameters for Traverse path linearly – Blending tab

Table 7-312 Overview of parameters for Traverse path linearly – Blending tab

Field/button	Meaning/instruction
	This tab is available as of SIMOTION Kernel version V4.3. It is only displayed when the value **Inactive** has **not** been selected in the Blending field of the Parameters tab. Here, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment.

Field/button	Meaning/instruction
Blending segment	Here, you specify whether a blending segment to the previous path segment will be inserted. **Do not insert a segment** (default value) A blending segment is not inserted. This corresponds to the behavior up to version V4.2 of the SIMOTION Kernel. See the description for the values **Active with dynamic response adaptation** and **Active without dynamic response adaptation** in the "Blending" field of the Parameters tab (Page 4466). **Insert circle segment** A circular blending segment is inserted. This is only possible between linear path segments (MCC command **Traverse path linearly**). The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: Starting from the end point or start point of the two path segments, the start point and end point of the blending segment is calculated with the blending distance. • Transition behavior = **Substitute**: The blending segment is inserted at the current path position. A fictitious end point or start point of the two path segments and the end point of the blending segment is calculated with the blending distance and the tangent at the path position. The blending segment is calculated so that at the transitions to the two path segments, the position and the velocity are smooth. **Insert polynomial segment** A blending segment described by a polynomial is inserted. This is always possible. The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: Starting from the end point or start point of the two path segments, the start point and end point of the blending segment is calculated with the blending distance. • Transition behavior = **Substitute**: The blending segment is inserted at the current path position. A fictitious end point or start point of the two path segments and the end point of the blending segment is calculated with the blending distance and the tangent at the path position. The blending segment is calculated so that at the transitions to the two path segments, the position, the velocity and the acceleration are smooth. **Stop** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The previous path motion is stopped immediately. The motion programmed with this command starts at the stop position. The dynamic response parameters of this command are also effective for the deceleration of the previous path motion. The path velocity and path acceleration are zero at the transition.

Field/button	Meaning/instruction
Blending distance	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment The blending distance is used to specify the distance from the end point or start point of the two path segments at which the transition geometry (blending segment) is inserted. The blending distance is measured as path length along the two path segments. Enter the value as a signed floating-point number. See also input field (Page 4022).
Velocity in the blending segment	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment Here you specify the velocity that is to be used in the blending segment. Acceleration and jerk are always taken from the current path command **High velocity** The higher velocity of the current and previous path command is used **Low velocity** The lower velocity of the current and previous path command is used

Overview of parameters for Traverse path linearly – Dynamic response tab

Table 7-313 Overview of parameters for Traverse path linearly – Dynamic response tab

Field/button	Meaning/instruction
	Here, you specify the dynamic response of the path motion. In standard cases, you define the dynamic response by means of a velocity profile with the associated values for the velocity, acceleration, deceleration and jerk. A detailed description of these parameters is provided in Overview of parameters – Dynamic response tab (Page 4031). Optionally, you can select a special velocity profile that is described by a cam.
Velocity	The entered value acts during the constant velocity phase. This parameter is evaluated only when **Special velocity profile** = **No**. System variable for default: userDefault.pathDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.profile

Field/button	Meaning/instruction
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. The parameters are evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variables for default: userDefault.pathDynamics.positiveAccelStartJerk userDefault.pathDynamics.positiveAccelEndJerk userDefault.pathDynamics.negativeAccelStartJerk userDefault.pathDynamics.negativeAccelEndJerk
Dynamic response adaptation	Here, you select whether the dynamic response limit values of the individual path axes are to be taken into account during path interpolation. **Inactive** The dynamic response limit values of the individual path axes are **not** taken into account during path interpolation. However, they remain active, and an error is triggered if they are exceeded. This setting should only be selected if it is ensured by other means (e.g. an appropriate specific velocity profile) that the limit values will not be exceeded. **Without segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. Path velocity and path acceleration are adapted for the entire path motion if necessary. **With segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. The path motion is divided up into individual segments, as determined by the system, if necessary. Within each segment, the path velocity and the path acceleration are adapted to the maximum axial dynamic response limit values. **Default** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.dynamicAdaption

Field/button	Meaning/instruction
Specific velocity profile	Here, you select whether the path dynamic response is specified with the velocity profile defined above or with a velocity profile that is defined by a cam. **No** (default value) The path dynamic response is specified with the velocity profile defined above with the parameters: **velocity profile**, **velocity**, **acceleration**, **deceleration**, and **jerk**. **Yes** The path dynamic response is specified with a velocity profile that is defined by a cam.
Cam	This field is only visible when **Specific velocity profile** = **Yes**. Here, you select the cam which defines the velocity profile. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All variables with the following technology object data type declared in the MCC source file or MCC chart (see Technology object data types (Page 4060)): camType. The **Profile position at start of path** and **Profile position at end of path** parameters are used to select an area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. The value range (y axis) is interpreted as the velocity in the unit configured on the path object.
Profile position at start of path Profile position at end of path	These two values determine the area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path linearly – Synchronous axis tab

Table 7-314 Overview of parameters for Traverse path linearly – Synchronous axis tab

Field/button	Meaning/instruction
	Here, you specify values for the motion of the additional position axis that is assigned to the path object and is traversed synchronously with the path motion.
Synchronous axis mode	Here, you specify how the synchronous axis (axis for path-synchronous motion) should be traversed with the path motion: **Absolute** The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. **Relative** The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. **Output path length** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) is outputted linearly, beginning from zero. **Output path length added** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w absolute and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w relative and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.w.mode

Field/button	Meaning/instruction
Synchronous axis direction	Select the direction of the path-synchronous motion. The direction must be specified in the following cases: • The **Synchronous axis mode** is relative • The axis for the path-synchronous motion is a modulo axis. **Positive** The direction of motion is the positive direction of the axis. With relative motion, the sign of the position is ignored. **Negative** The direction of motion is the negative direction of the axis. With relative motion, the sign of the position is ignored. **From position** The direction of motion is determined by the sign of the **Synchronous axis position**. **Shortest path** • With **Synchronous axis mode** relative: The direction of motion is determined by the sign of the **Synchronous axis position**. • With modulo axes: The direction of motion is the direction in which the programmed target position can be reached via the shortest path. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.w.direction
Synchronous axis position	This parameter is evaluated only for the following settings for **Synchronous axis mode** and has different meanings: • **Absolute** or **w absolute ...** : End point of the path-synchronous motion. • **Relative** or **w relative ...** : Traversing distance of the path-synchronous motion. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path linearly – Expert tab

Table 7-315 Overview of parameters for Traverse path linearly – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Traverse path linearly

Path technology package:

- _movePathLinear

Overview of parameters for Traverse path linearly / _movePathLinear

Table 7-316 Parameters (MCC command Traverse path linearly compared to system function _movePathLinear)

Parameters of the MCC command Traverse path linearly	Parameters of the system function _movePathLinear
Path object	pathObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Coordinate system	csType
Number	csNumber
Path plane	pathPlane
Mode	pathMode
Target coordinate X	x
Target coordinate Y	y
Target coordinate Z	z
Blending	blendingMode
Blending tab	
Blending segment	transitionType
Blending distance	blendingDistance
Velocity in the blending segment	transitionVelocityMode:
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Dynamic response adaptation	dynamicAdaption
Specific velocity profile	specificVelocityProfile
Cam	profileReference
Profile position at start of path	profileStartPosition
Profile position at end of path	profileEndPosition
Synchronous axis tab	
Synchronous axis mode	wMode
Synchronous axis direction	wDirection
Synchronous axis position	w
Expert tab	
CommandID variable	commandId
Return variable	–

Traverse path circularly

This function is available in SIMOTION Kernel as of Version 4.1.

This command moves the path axes interconnected with the specified path object on an arc to an end point. You can define the arc as follows:

- With the end point of the motion and the radius and orientation of the arc (enables only two-dimensional motion in one of the main Cartesian planes)
- With the center and angle of the arc (enables only two-dimensional motion in one of the main Cartesian planes)
- With the intermediate and endpoint of the motion (enables three-dimensional or two-dimensional motion in one of the main Cartesian planes)

The center, intermediate point, and end point can be specified as absolute or relative.

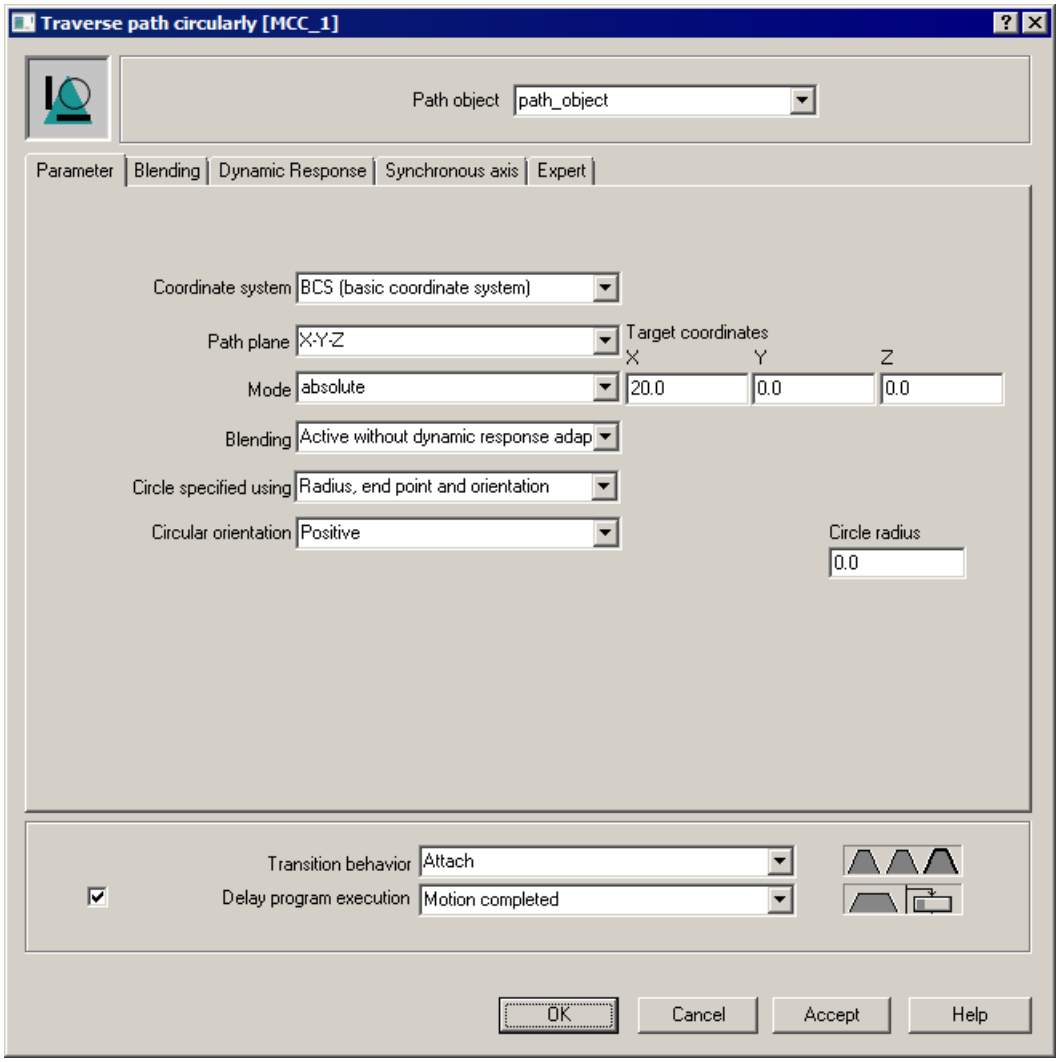


Figure 7-175 Parameter screen form: Traverse path circularly

For the **fundamentals of path interpolation**, see the TO Path Object Function Manual

For further information on **circular paths**, see the TO Path Object Function Manual.

Overview of parameters for Traverse path circularly

Table 7-317 Overview of parameters for Traverse path circularly

Field/button	Meaning/instruction
Path object	In Path object, select the path object whose assigned path axes are to be traversed to the end point. The list contains: • All path objects that are defined on the relevant device. The path objects are defined in the PATH OBJECTS folder in the project navigator. • All variables with the following data type of a technology object declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): _PathObjectType.
Parameters tab	See Overview of parameters for Traverse path circularly - Parameters tab (Page 4479)
Blending tab	See Overview of parameters for Traverse path circularly - Blending tab (Page 4484)
Dynamic response tab	See Overview of parameters for Traverse path circularly - Dynamic response tab (Page 4485)
Synchronous axis tab	See Overview of parameters for Traverse path circularly - Synchronous axis tab (Page 4488)
Expert tab	See Overview of parameters for Traverse path circularly - Expert tab (Page 4489)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the path object or on the axes involved. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Select the checkbox if you wish the system to delay execution of the following command in the MCC chart until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Traverse path circularly – Parameters tab

Table 7-318 Overview of parameters for Traverse path circularly – Parameters tab

Field/button	Meaning/instruction
Coordinate system	This field is available as of SIMOTION Kernel version V4.3. Here you select the coordinate system to which the path motion to be programmed refers. **BCS (basic coordinate system)** (default value) Static, right-handed, rectangular (Cartesian) coordinate system in accordance with DIN 66217. **OCS (object coordinate system)** Coordinate system defined on the path object that is offset and rotated with regard to the basic coordinate system. It can also be coupled synchronously to another motion (e.g. an axis). Up to 3 object coordinate systems can be defined.
Number	This field is available as of SIMOTION Kernel version V4.3. An entry is required in this field if you have selected the following in the **Coordinate system** field: • OCS (object coordinate system) Select the number of the object coordinate system to be used. Possible values are 1 to 3. **1** (default value) Object coordinate system 1
Path plane	Here, you select whether the motion is to be carried out in 3 dimensions (3D) or 2 dimensions (2D). **X-Y-Z** Three-dimensional motion **X-Y main plane** **Y-Z main plane** **Z-X main plane** 2-dimensional motion in the selected main plane of the Cartesian basic coordinate system. The target coordinate located outside the selected plane is ignored. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.plane
Mode	This parameter is evaluated only for: • Circle specified using radius, end point, and orientation • Circle specified using intermediate and end point The meaning of the subsequently programmed target coordinates is defined in more detail here. **Absolute** The programmed target coordinates refer to the zero point of the basic coordinate system. **Relative** The programmed target coordinates refer to the path position within the basic coordinate system when the path command is executed (point in time of change). **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.mode

Field/button	Meaning/instruction
Target coordinates X, Y, Z	These parameters are evaluated only for: • Circle specified using radius, end point, and orientation • Circle specified using intermediate and end point The end point of the motion is specified here. The meaning depends on the **Mode** parameter (see above)**:** • Absolute mode: Absolute coordinates with respect to the zero point of the basic coordinate system. • Relative mode: Relative coordinates with respect to the path position within the basic coordinate system when the path command is executed (point in time of change) Enter the values as signed floating-point numbers. See the input field (Page 4022).
Blending	Using blending, the path segment programmed with this command can be linked to the previous path segment to form one complete path with no stop. At the transition, the system provides for a constant path velocity and a constant path acceleration. As of version V4.3 of the SIMOTION Kernel, a transition geometry (blending segment) can also be inserted. **Active with dynamic response adaptation** The dynamic response limit values of the individual path axes are also applied in the blending area or blending segment. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. A blending segment is not inserted. The path velocity is nearly reduced to zero at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4484) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel. **Active without dynamic response adaptation** In the blending area or blending segment, path interpolation takes into account only the scalar dynamic response limit values of the path (path velocity, path acceleration and jerk). The dynamic response limit values specific to the axes are first taken into account during movement of the individual path axes. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. Axis-specific deviations from the path can occur at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4484) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel.

Field/button	Meaning/instruction
Blending (continued)	**Inactive** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The motion with its dynamic response parameters programmed with this command immediately replaces the previous path motion. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.blending.mode
Circle specified via	Here, you select the characteristic values specifying the arc to be traversed. **Radius, end point, and orientation** Only 2D interpolation is possible in the main plane of the coordinate system selected in the Path plane parameter. The end point specified with the target coordinates is approached on an arc starting from the current position. You define the arc with its radius and circular orientation. Note that for the circular orientation, in addition to the direction of rotation, you must specify whether traversing is to be performed along the large or small arc. **Center and angle** Only 2D interpolation is possible in the main plane of the coordinate system selected in the Path plane parameter. Starting from the current position, an arc is traversed. You define this arc by its center coordinates, the angle, and the circular orientation. **Intermediate and end point** 2D and 3D interpolation are possible. The end point specified with the target coordinates is approached on an arc starting from the current position. You define the arc using an intermediate point that you specify with its intermediate point coordinates. With 3D interpolation, you define the path plane through the current position at the beginning of the path, the intermediate point, and the end point. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.circularType

Field/button	Meaning/instruction
Circular orientation	This parameter is evaluated only for: • Circle specified using radius, end point, and orientation • Circle specified using center and angle Here, you enter the orientation of the arc. **Positive** (default value) Positive direction of rotation in the selected main plane. For a circle specified using radius, end point, and orientation, the angle traveled is less than or equal to 180 °. **Negative** Negative direction of rotation in the selected main plane. For a circle specified using radius, end point, and orientation, the angle traveled is less than or equal to 180 °. **Positive on large arc** Positive direction of rotation in the selected main plane. For a circle specified using radius, end point, and orientation, the angle traveled is greater than or equal to 180 °. **Negative on large arc** Negative direction of rotation in the selected main plane. For a circle specified using radius, end point, and orientation, the angle traveled is greater than or equal to 180 °.
Intermediate point mode	This parameter is evaluated only for: • Circle specified using center and angle • Circle specified using intermediate and end point The meaning of the subsequently programmed center and intermediate point coordinates is defined in more detail here. **Absolute** The programmed center and intermediate point coordinates refer to the zero point of the basic coordinate system. **Relative** The programmed center and intermediate coordinates refer to the path position within the basic coordinate system when the path command is executed (point in time of change). **As end point coordinate** The setting from the Mode parameter is taken for the programmed center and intermediate point coordinates. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.ijkMode
Center coordinates Center point coordinates I, J, K	These parameters are evaluated only for: • Circle specified using center and angle • Circle specified using intermediate and end point Here, you specify the center of the arc and the intermediate point of the circular motion. The meaning depends on the **Intermediate point mode** parameter (see above)**:** • Intermediate point absolute mode: Absolute coordinates with respect to the zero point of the basic coordinate system. • Intermediate point relative mode: Relative coordinates with respect to the path position within the basic coordinate system when the path command is executed (point in time of change). Enter the values as signed floating-point numbers. See Selection list (combo box) (Page 4022).

Field/button	Meaning/instruction
Angle	The parameter is evaluated only for circle specification with center point and angle: Here, you specify the angle of rotation of the arc. Enter the value as a signed floating-point number. See the input field (Page 4022).
Radius	The parameter is evaluated only for circle specification with radius, end point, and orientation: Here, you specify the radius of the arc. Enter the value as a signed floating-point number. See the input field (Page 4022).

Overview of parameters for Traverse path circularly – Blending tab

Table 7-319 Overview of parameters for Traverse path circularly – Blending tab

Field/button	Meaning/instruction
	This tab is available as of SIMOTION Kernel version V4.3. It is only displayed when the value **Inactive** has **not** been selected in the Blending field of the Parameters tab. Here, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment.
Blending segment	Here, you specify whether a blending segment to the previous path segment will be inserted. **Do not insert a segment** (default value) A blending segment is not inserted. This corresponds to the behavior up to version V4.2 of the SIMOTION Kernel. See the description for the values **Active with dynamic response adaptation** and **Active without dynamic response adaptation** in the "Blending" field of the Parameters tab (Page 4479). **Insert polynomial segment** A blending segment described by a polynomial is inserted. This is always possible. The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: Starting from the end point or start point of the two path segments, the start point and end point of the blending segment is calculated with the blending distance. • Transition behavior = **Substitute**: The blending segment is inserted at the current path position. A fictitious end point or start point of the two path segments and the end point of the blending segment is calculated with the blending distance and the tangent at the path position. The blending segment is calculated so that at the transitions to the two path segments, the position, the velocity and the acceleration are smooth. **Stop** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The previous path motion is stopped immediately. The motion programmed with this command starts at the stop position. The dynamic response parameters of this command are also effective for the deceleration of the previous path motion. The path velocity and path acceleration are zero at the transition.

Field/button	Meaning/instruction
Blending distance	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment The blending distance is used to specify the distance from the end point or start point of the two path segments at which the transition geometry (blending segment) is inserted. The blending distance is measured as path length along the two path segments. Enter the value as a signed floating-point number. See also input field (Page 4022).
Velocity in the blending segment	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment Here you specify the velocity that is to be used in the blending segment. Acceleration and jerk are always taken from the current path command **High velocity** The higher velocity of the current and previous path command is used **Low velocity** The lower velocity of the current and previous path command is used

Overview of parameters for Traverse path circularly – Dynamic response tab

Table 7-320 Overview of parameters for Traverse path circularly – Dynamic response tab

Field/button	Meaning/instruction
	Here, you specify the dynamic response of the path motion. In standard cases, you define the dynamic response by means of a velocity profile with the associated values for the velocity. Acceleration, deceleration, and jerk. A detailed description of these parameters is provided in Overview of parameters – Dynamic response tab (Page 4031). Optionally, you can select a special velocity profile that is described by a cam.
Velocity	The entered value acts during the constant velocity phase. This parameter is evaluated only when **Special velocity profile** = **No**. System variable for default: userDefault.pathDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.profile

Field/button	Meaning/instruction
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. The parameters are evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variables for default: userDefault.pathDynamics.positiveAccelStartJerk userDefault.pathDynamics.positiveAccelEndJerk userDefault.pathDynamics.negativeAccelStartJerk userDefault.pathDynamics.negativeAccelEndJerk
Dynamic response adaptation	Here, you select whether the dynamic response limit values of the individual path axes are to be taken into account during path interpolation. **Inactive** The dynamic response limit values of the individual path axes are **not** taken into account during path interpolation. However, they remain active, and an error is triggered if they are exceeded. This setting should only be selected if it is ensured by other means (e.g. an appropriate specific velocity profile) that the limit values will not be exceeded. **Without segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. Path velocity and path acceleration are adapted for the entire path motion if necessary. **With segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. The path motion is divided up into individual segments, as determined by the system, if necessary. Within each segment, the path velocity and the path acceleration are adapted to the maximum axial dynamic response limit values. **Default** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.dynamicAdaption

Field/button	Meaning/instruction
Specific velocity profile	Here, you select whether the path dynamic response is specified with the velocity profile defined above or with a velocity profile that is defined by a cam. **No** (default value) The path dynamic response is specified with the velocity profile defined above with the parameters: **velocity profile**, **velocity**, **acceleration**, **deceleration**, and **jerk**. **Yes** The path dynamic response is specified with a velocity profile that is defined by a cam.
Cam	This field is only visible when **Specific velocity profile** = **Yes**. Here, you select the cam which defines the velocity profile. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All variables with the following technology object data type declared in the MCC unit or MCC chart (Description of technology object data types (Page 4060)): camType. The **Profile position at start of path** and **Profile position at end of path** parameters are used to select an area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. The value range (y axis) is interpreted as the velocity in the unit configured on the path object.
Profile position at start of path Profile position at end of path	These two values determine the area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path circularly – Synchronous axis tab

Table 7-321 Overview of parameters for Traverse path circularly – Synchronous axis tab

Field/button	Meaning/instruction
	Here, you specify values for the motion of the additional position axis that is assigned to the path object and is traversed synchronously with the path motion.
Synchronous axis mode	Here, you specify how the synchronous axis (axis for path-synchronous motion) should be traversed with the path motion: **Absolute** The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. **Relative** The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. **Output path length** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) is outputted linearly, beginning from zero. **Output path length added** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w absolute and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w relative and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.w.mode

Field/button	Meaning/instruction
Synchronous axis direction	Select the direction of the path-synchronous motion. The direction must be specified in the following cases: • The **Synchronous axis mode** is relative • The axis for the path-synchronous motion is a modulo axis. **Positive** The direction of motion is the positive direction of the axis. With relative motion, the sign of the position is ignored. **Negative** The direction of motion is the negative direction of the axis. With relative motion, the sign of the position is ignored. **From position** The direction of motion is determined by the sign of the **Synchronous axis position**. **Shortest path** • With **Synchronous axis mode** relative: The direction of motion is determined by the sign of the **Synchronous axis position**. • With modulo axes: The direction of motion is the direction in which the programmed target position can be reached via the shortest path. **Default value** See Selection list (combo box) (Page 4022) System variable for default: userDefault.w.direction
Synchronous axis position	This parameter is evaluated only for the following settings for **Synchronous axis mode** and has different meanings: • **Absolute** or **w absolute ...** : End point of the path-synchronous motion. • **Relative** or **w relative ...** : Traversing distance of the path-synchronous motion. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path circularly – Expert tab

Table 7-322 Overview of parameters for Traverse path circularly – Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Traverse path circularly

Path technology package:

- _movePathCircular

Overview of parameters for Traverse path circularly / _movePathCircular

Table 7-323 Parameters (MCC command Traverse path circularly compared to system function _movePathCircular)

Parameters of the MCC command Traverse path circularly	Parameters of the system function _movePathCircular
Path object	pathObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Coordinate system	csType
Number	csNumber
Path plane	pathPlane
Mode	pathMode
Target coordinate X	x
Target coordinate Y	y
Target coordinate Z	z
Blending	blendingMode
Circle specified via	circularType
Circular orientation	circleDirection
Intermediate point mode	ijkMode
Center/intermediate point coordinate I	i
Center/intermediate point coordinate J	j
Center/intermediate point coordinate K	k
Angle	arc
Radius	radius
Blending tab	
Blending segment	transitionType
Blending distance	blendingDistance
Velocity in the blending segment	transitionVelocityMode:
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Dynamic response adaptation	dynamicAdaption
Specific velocity profile	specificVelocityProfile
Cam	profileReference
Profile position at start of path	profileStartPosition
Profile position at end of path	profileEndPosition
Synchronous axis tab	

Parameters of the MCC command Traverse path circularly	Parameters of the system function _movePathCircular
Synchronous axis mode	wMode
Synchronous axis direction	wDirection
Synchronous axis position	w
Expert tab	
CommandID variable	commandId
Return variable	–

Traverse path using polynomials

This function is available in SIMOTION Kernel as of Version 4.1.

This command moves the path axes interconnected with the specified path object on a path described by a fifth order polynomial.

$\mathbf{P}(q) = \mathbf{a}_0 + \mathbf{a}_1 q + \mathbf{a}_2 q^2 + \mathbf{a}_3 q^3 + \mathbf{a}_4 q^4 + \mathbf{a}_5 q^5$, where $q \in [0, 1]$.

Here the polynomial coefficients $\mathbf{a}_0$ to $\mathbf{a}_5$ are three-dimensional vectors. You may specify these in the following ways:

- **By calculation from geometric data**
  In this case you specify the following values:
  - Start point (current position) and end point (target coordinates)
  - First geometric derivative (tangential vector) and second geometric derivative (curvature vector) at the start point
  - First geometric derivative (tangential vector) and second geometric derivative (curvature vector) at the end point.

  You can select how the first and second geometric derivatives are specified at the start point:
  - With explicit specification of values
  - By accepting the corresponding values at the end point of the previous path command

  The polynomial coefficients $\mathbf{a}_0$ to $\mathbf{a}_5$ are calculated from this geometric data.
  System functions are available for performing a geometric analysis of paths (e.g. determining the geometric derivatives at specified points).
- **By direct specification of the polynomial coefficients**
  In this case you specify the following values:
  - The start point (current position) corresponds to $\mathbf{P}(0)$.
  - The end point (target coordinates) corresponds to $\mathbf{P}(1)$.
  - The polynomial coefficients $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$ and $\mathbf{a}_5$.

  The polynomial coefficients $\mathbf{a}_0$ and $\mathbf{a}_1$ are calculated from this data:
  - $\mathbf{a}_0 = \mathbf{P}(0)$
  - $\mathbf{a}1 = \mathbf{P}(1) - \mathbf{P}(0) - \mathbf{a}_2 - \mathbf{a}_3 - \mathbf{a}_4 - \mathbf{a}_5$.

The end point entered can be absolute or relative.

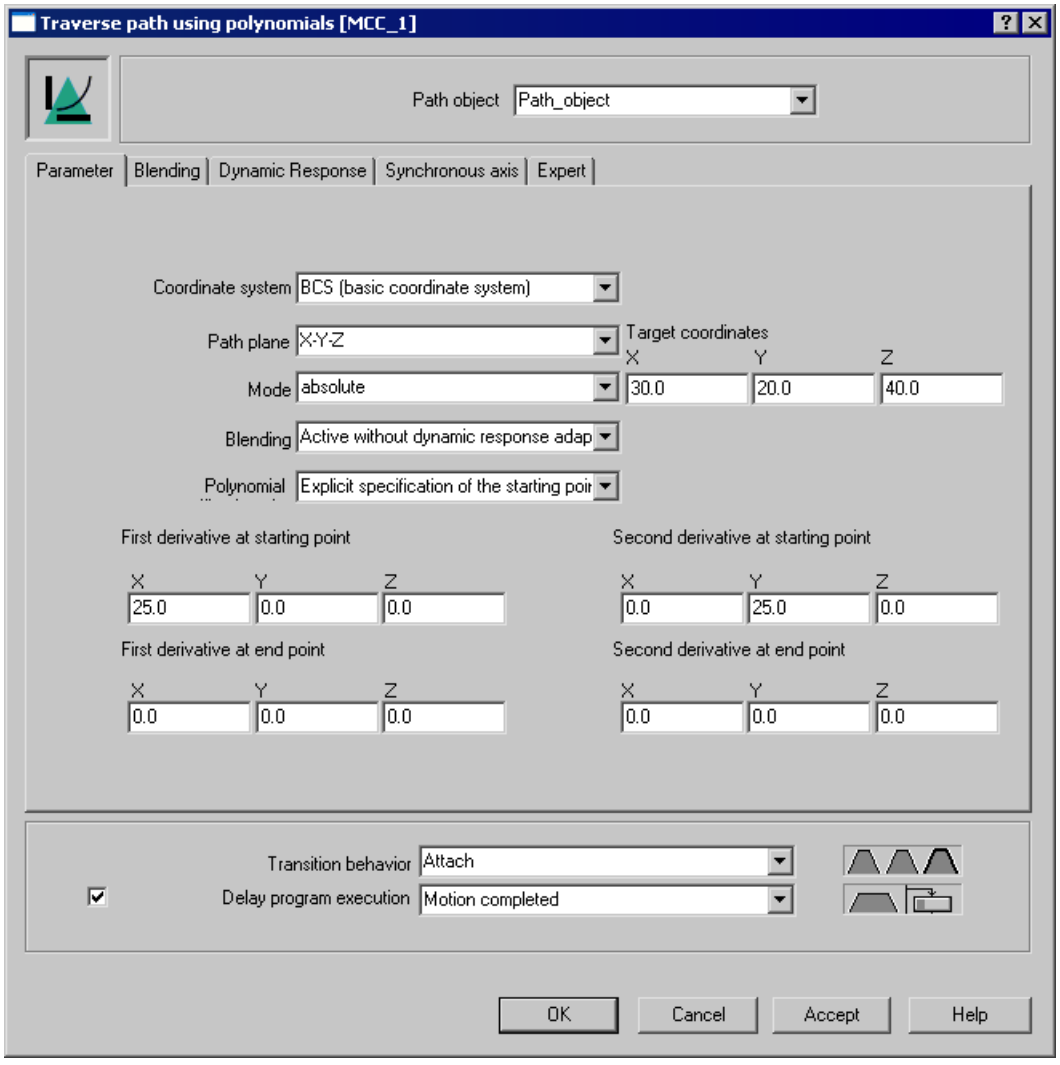


Figure 7-176 Parameter screen form: Traverse path using polynomials

For the **fundamentals of path interpolation**, see the TO Path Object Function Manual

For further information on **polynomial paths**, see the TO Path Object Function Manual.

Overview of parameters for Traverse path using polynomials

Table 7-324 Overview of parameters for Traverse path using polynomials

Field/button	Meaning/instruction
Path object	In Path object, select the path object whose assigned path axes are to be traversed to the end point. The list contains: • All path objects that are defined on the relevant device. The path objects are defined in the PATH OBJECTS folder in the project navigator. • All variables with the following data type of a technology object declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): _PathObjectType.
Parameters tab	See Overview of parameters for Traverse path using polynomials - Parameters tab (Page 4494)
Blending tab	See Overview of parameters for Traverse path using polynomials - Blending tab (Page 4499)
Dynamic response tab	See Overview of parameters for Traverse path using polynomials - Dynamic response tab (Page 4500)
Synchronous axis tab	See Overview of parameters for Traverse path using polynomials - Path-synchronous motion tab (Page 4503)
Expert tab	See Overview of parameters for Traverse path using polynomials - Expert tab (Page 4504)
Transition behavior	Here, you program the transition behavior between the programmed command and the command currently active on the axes involved. The selected behavior determines the position of the command in the command queue. See also Transition behavior from the currently active motion command (Page 4036).
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Traverse path using polynomials – Parameters tab

Table 7-325 Overview of parameters for Traverse path using polynomials – Parameters tab

Field/button	Meaning/instruction
Coordinate system	This field is available as of SIMOTION Kernel version V4.3. Here you select the coordinate system to which the path motion to be programmed refers. **BCS (basic coordinate system)** (default value) Static, right-handed, rectangular (Cartesian) coordinate system in accordance with DIN 66217. **OCS (object coordinate system)** Coordinate system defined on the path object that is offset and rotated with regard to the basic coordinate system. It can also be coupled synchronously to another motion (e.g. an axis). Up to 3 object coordinate systems can be defined.
Number	This field is available as of SIMOTION Kernel version V4.3. An entry is required in this field if you have selected the following in the **Coordinate system** field: • OCS (object coordinate system) Select the number of the object coordinate system to be used. Possible values are 1 to 3. **1** (default value) Object coordinate system 1
Path plane	Here, you select whether the motion is to be carried out in 3 dimensions (3D) or 2 dimensions (2D). **X-Y-Z** Three-dimensional motion **X-Y main plane** **Y-Z main plane** **Z-X main plane** 2-dimensional motion in the selected main plane of the Cartesian basic coordinate system. The target coordinate located outside the selected plane is ignored. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.plane
Mode	The meaning of the subsequently programmed target coordinates is defined in more detail here. **Absolute** The programmed target coordinates refer to the zero point of the basic coordinate system. **Relative** The programmed target coordinates refer to the path position within the basic coordinate system when the path command is executed (point in time of change). current axis position and specify the (straight) traversing path. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.mode

Field/button	Meaning/instruction
Target coordinates X, Y, Z	The end point of the motion is specified here. The meaning depends on the **Mode** parameter (see above)**:** • Absolute mode: Absolute coordinates with respect to the zero point of the basic coordinate system. • Relative mode: Relative coordinates with respect to the path position within the basic coordinate system when the path command is executed (point in time of change). Enter the values as signed floating-point numbers. See the input field (Page 4022).
Blending	Using blending, the path segment programmed with this command can be linked to the previous path segment to form one complete path with no stop. At the transition, the system provides for a constant path velocity and a constant path acceleration. As of version V4.3 of the SIMOTION Kernel, a transition geometry (blending segment) can also be inserted. **Active with dynamic response adaptation** The dynamic response limit values of the individual path axes are also applied in the blending area or blending segment. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. A blending segment is not inserted. The path velocity is nearly reduced to zero at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4499) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel. **Active without dynamic response adaptation** In the blending area or blending segment, path interpolation takes into account only the scalar dynamic response limit values of the path (path velocity, path acceleration and jerk). The dynamic response limit values specific to the axes are first taken into account during movement of the individual path axes. • The following applies up to version V4.2 of the SIMOTION Kernel: This setting is evaluated only with the **Attach** or **Attach and discard the pending command** transition behavior. Axis-specific deviations from the path can occur at transitions of path segments in which the gradients or curvatures are not smooth. For the **Substitute** transition behavior, see setting **Inactive**. • The following applies as of version V4.3 of the SIMOTION Kernel: The "Blending" tab (Page 4499) is displayed. There, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment. With the **Do not insert a segment** selection, the same behavior applies as up to version V4.2 of the SIMOTION Kernel.

Field/button	Meaning/instruction
Blending (continued)	**Inactive** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The motion with its dynamic response parameters programmed with this command immediately replaces the previous path motion. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.blending.mode

Field/button	Meaning/instruction
Polynomial specified via	Here you select how the 5. order polynomial that describes the motion path is specified. **Explicit specification of start point data** The polynomial is calculated from geometric data. Specify explicitly: • Start point (current position) and end point (target coordinates) • First geometric derivative (tangential vector) and second geometric derivative (curvature vector) at the start point • First geometric derivative (tangential vector) and second geometric derivative (curvature vector) at the end point. System functions are available for performing a geometric analysis of paths (e.g. determining the geometric derivatives at specified points). **Attach continuously** The polynomial is calculated from geometric data. Specify explicitly: • Start point (current position) and end point (target coordinates) • First geometric derivative (tangential vector) and second geometric derivative (curvature vector) at the end point. The first geometric derivative (tangential vector) and the second geometric derivative (curvature vector) at the start point (current position) are set equal to the geometric derivatives at the end point of the previous path command. If the geometric derivatives cannot be determined in the start point (if no current motion is available), the command is not executed and error message 50002 "Calculation of the geometry element not possible, reason 3" is outputted. System functions are available for performing a geometric analysis of paths (e.g. determining the geometric derivatives at specified points). **Direct specification of the polynomial coefficients** Specify the polynomial coefficients explicitly through the following values: • Start point **P**(0) (current position). • End point **P**(1) (target coordinates). • Polynomial coefficients $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$ and $\mathbf{a}_5$. The polynomial coefficients $\mathbf{a}_0$ and $\mathbf{a}_1$ are calculated from this data: • $\mathbf{a}_0 = \mathbf{P}(0)$ • $\mathbf{a}1 = \mathbf{P}(1) - \mathbf{P}(0) - \mathbf{a}_2 - \mathbf{a}_3 - \mathbf{a}_4 - \mathbf{a}_5$. **Default value** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.polynomialMode
First derivative at the start point X, Y, Z	This parameter is evaluated only with polynomial specification via **Explicit specification of the start point data**. Here, you specify the first geometric derivative (tangential vector) at the start point (current position). Enter the values as signed floating-point numbers. See the input field (Page 4022).
Second derivative at the start point X, Y, Z	This parameter is evaluated only with polynomial specification via **Explicit specification of the start point data**. Here, you specify the second geometric derivative (curvature vector) at the start point (current position). Enter the values as signed floating-point numbers. See the input field (Page 4022).

Field/button	Meaning/instruction
First derivative at the end point X, Y, Z	This parameter is evaluated only with polynomial specification via **Explicit specification of the start point data** and **Attach continuously**. Here, you specify the first geometric derivative (tangential vector) at the end point (target coordinates). Enter the values as signed floating-point numbers. See the input field (Page 4022).
Second derivative at the end point X, Y, Z	This parameter is evaluated only with polynomial specification via **Explicit specification of the start point data** and **Attach continuously**. Here, you specify the second geometric derivative (curvature vector) at the end point (target coordinates). Enter the values as signed floating-point numbers. See the input field (Page 4022).
Coefficient a2 X, Y, Z	These parameters are evaluated only with polynomial specification via **Direct specification of the polynomial coefficients**. Here you specify the coefficients $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$ and $\mathbf{a}_5$ of the 5. order polynomial. Enter the values as signed floating-point numbers. See the input field (Page 4022).
Coefficient a3 X, Y, Z	
Coefficient a4 X, Y, Z	
Coefficient a5 X, Y, Z	

Overview of parameters for Traverse path using polynomials – Blending tab

Table 7-326 Overview of parameters for Traverse path using polynomials – Blending tab

Field/button	Meaning/instruction
	This tab is available as of SIMOTION Kernel version V4.3. It is only displayed when the value **Inactive** has **not** been selected in the Blending field of the Parameters tab. Here, for example, you can define whether a transition geometry (blending segment) will be inserted between the path segment programmed with this command and the previous path segment.
Blending segment	Here, you specify whether a blending segment to the previous path segment will be inserted. **Do not insert a segment** (default value) A blending segment is not inserted. This corresponds to the behavior up to version V4.2 of the SIMOTION Kernel. See the description for the values **Active with dynamic response adaptation** and **Active without dynamic response adaptation** in the "Blending" field of the Parameters tab (Page 4494). **Insert polynomial segment** A blending segment described by a polynomial is inserted. This is always possible. The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: Starting from the end point or start point of the two path segments, the start point and end point of the blending segment is calculated with the blending distance. • Transition behavior = **Substitute**: The blending segment is inserted at the current path position. A fictitious end point or start point of the two path segments and the end point of the blending segment is calculated with the blending distance and the tangent at the path position. The blending segment is calculated so that at the transitions to the two path segments, the position, the velocity and the acceleration are smooth. **Stop** The behavior depends on the selection at "Transition behavior": • Transition behavior = **Attach** or **Attach and discard existing command**: The motion programmed with this command does not begin until the setpoint interpolation of the previous command is complete and its motion has reached the target coordinates. The path velocity and path acceleration are therefore zero at the transition. • Transition behavior = **Substitute**: The previous path motion is stopped immediately. The motion programmed with this command starts at the stop position. The dynamic response parameters of this command are also effective for the deceleration of the previous path motion. The path velocity and path acceleration are zero at the transition.

Field/button	Meaning/instruction
Blending distance	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment The blending distance is used to specify the distance from the end point or start point of the two path segments at which the transition geometry (blending segment) is inserted. The blending distance is measured as path length along the two path segments. Enter the value as a signed floating-point number. See also input field (Page 4022).
Velocity in the blending segment	An entry is required in this field if you have selected the following in the **Blending segment** field: • Insert circle segment • Insert polynomial segment Here you specify the velocity that is to be used in the blending segment. Acceleration and jerk are always taken from the current path command **High velocity** The higher velocity of the current and previous path command is used **Low velocity** The lower velocity of the current and previous path command is used

Overview of parameters for Traverse path using polynomials – Dynamic response tab

Table 7-327 Overview of parameters for Traverse path using polynomials – Dynamic response tab

Field/Button	Meaning/instruction
	Here, you specify the dynamic response of the path motion. In standard cases, you define the dynamic response by means of a velocity profile with the associated values for the velocity. Acceleration, deceleration, and jerk. A detailed description of these parameters is provided in Overview of parameters – Dynamic response tab (Page 4031). Optionally, you can select a special velocity profile that is described by a cam.
Velocity	The entered value acts during the constant velocity phase. This parameter is evaluated only when **Special velocity profile** = **No**. System variable for default: userDefault.pathDynamics.velocity
Velocity profile	In this field, you define the transitions between the individual motion phases. The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.profile

Field/Button	Meaning/instruction
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.positiveAccel
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). The parameter is evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variable for default: userDefault.pathDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. The parameters are evaluated only in the following cases: • When **Special velocity profile** = **No** • When **Special velocity profile** = **Yes** and when accelerating to the velocity specified in the special velocity profile. System variables for default: userDefault.pathDynamics.positiveAccelStartJerk userDefault.pathDynamics.positiveAccelEndJerk userDefault.pathDynamics.negativeAccelStartJerk userDefault.pathDynamics.negativeAccelEndJerk
Dynamic response adaptation	Here, you select whether the dynamic response limit values of the individual path axes are to be taken into account during path interpolation. **Inactive** The dynamic response limit values of the individual path axes are **not** taken into account during path interpolation. However, they remain active, and an error is triggered if they are exceeded. This setting should only be selected if it is ensured by other means (e.g. an appropriate specific velocity profile) that the limit values will not be exceeded. **Without segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. Path velocity and path acceleration are adapted for the entire path motion if necessary. **With segmentation across the path** The dynamic response limit values of the individual path axes for velocity and acceleration are taken into account during path interpolation. If a position axis is configured for the path-synchronous motion, its dynamic response limit values are also included. The path motion is divided up into individual segments, as determined by the system, if necessary. Within each segment, the path velocity and the path acceleration are adapted to the maximum axial dynamic response limit values. **Default** See Selection list (combo box) (Page 4022). System variable for default: userDefault.path.dynamicAdaption

Field/Button	Meaning/instruction
Specific velocity profile	Here, you select whether the path dynamic response is specified with the velocity profile defined above or with a velocity profile that is defined by a cam. **No** (default value) The path dynamic response is specified with the velocity profile defined above with the parameters: **velocity profile**, **velocity**, **acceleration**, **deceleration**, and **jerk**. **Yes** The path dynamic response is specified with a velocity profile that is defined by a cam.
Cam	This field is only visible when **Specific velocity profile** = **Yes**. Here, you select the cam which defines the velocity profile. The following are available: • All cams that are defined on the relevant device. The cams are defined in the CAMS folder in the project navigator. • All variables with the following technology object data type declared in the MCC source file or MCC chart (see Technology object data types (Page 4060)): camType. The **Profile position at start of path** and **Profile position at end of path** parameters are used to select an area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. The value range (y axis) is interpreted as the velocity in the unit configured on the path object.
Profile position at start of path Profile position at end of path	These two values determine the area within the definition range of the cam (x axis) that is mapped by scaling to the length of the programmed path. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path using polynomials – Synchronous axis tab

Table 7-328 Overview of parameters for Traverse path using polynomials – Path-synchronous motion tab

Field/button	Meaning/instruction
	Here, you specify values for the motion of the additional position axis that is assigned to the path object and is traversed synchronously with the path motion.
Synchronous axis mode	Here, you specify how the synchronous axis (axis for path-synchronous motion) should be traversed with the path motion: **Absolute** The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. **Relative** The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. **Output path length** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) is outputted linearly, beginning from zero. **Output path length added** At the position axis for the path-synchronous motion w, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w absolute and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the end point of the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **w absolute and output of path length in w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) is outputted linearly beginning from zero. **w relative and output of path length added to w2** (As of Kernel V4.4) The programmed **Synchronous axis position** is the path traversed by the path-synchronous motion w. At the position axis interconnected with w2, the path length (i.e. the distance covered by the path motion) added to the existing value is outputted linearly at the beginning of the path motion. **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.w.mode

Field/button	Meaning/instruction
Synchronous axis direction	Select the direction of the path-synchronous motion. The direction must be specified in the following cases: • The **Synchronous axis mode** is relative • The axis for the path-synchronous motion is a modulo axis. **Positive** The direction of motion is the positive direction of the axis. With relative motion, the sign of the position is ignored. **Negative** The direction of motion is the negative direction of the axis. With relative motion, the sign of the position is ignored. **From position** The direction of motion is determined by the sign of the **Synchronous axis position**. **Shortest path** • With **Synchronous axis mode** relative: The direction of motion is determined by the sign of the **Synchronous axis position**. • With modulo axes: The direction of motion is the direction in which the programmed target position can be reached via the shortest path. **Default** See Selection list (combo box) (Page 4022) System variable for default: userDefault.w.direction
Synchronous axis position	This parameter is evaluated only for the following settings for **Synchronous axis mode** and has different meanings: • **Absolute** or **w absolute ...** : End point of the path-synchronous motion. • **Relative** or **w relative ...** : Traversing distance of the path-synchronous motion. Enter the values as signed floating-point numbers. See the input field (Page 4022).

Overview of parameters for Traverse path using polynomials – Expert tab

Table 7-329 Overview of parameters for Traverse path using polynomials – Expert tab

Field/Button	Explanation/Instructions
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Traverse path using polynomials

Path technology package:

- _movePathPolynomial

Overview of parameters for Traverse path using polynomials / _movePathPolynomial

Table 7-330 Parameters (MCC command Traverse path using polynomials compared to system function _movePathPolynomial)

Parameters of the MCC command Traverse path using polynomials	**Parameters of the system function _movePathPolynomial**
Path object	pathObject
Transition behavior	mergeMode
Delay program execution	nextCommand
Parameters tab	
Coordinate system	csType
Number	csNumber
Path plane	pathPlane
Mode	pathMode
Target coordinate X	x
Target coordinate Y	y
Target coordinate Z	z
Blending	blendingMode
Polynomial specified via	polynomialMode
First derivative at start point X	vector1x
First derivative at start point Y	vector1y
First derivative at start point Z	vector1z
Second derivative at start point X	vector2x
Second derivative at start point Y	vector2y
Second derivative at start point Z	vector2z
First derivative at end point X	Dependent on **Polynomial specified via** parameter: • With Explicit specification of start point data: vector3x • With Attach continuously: vector1x
First derivative at end point Y	Dependent on **Polynomial specified via** parameter: • With Explicit specification of start point data: vector3y • With Attach continuously: vector1y
First derivative at end point Z	Dependent on **Polynomial specified via** parameter: • With Explicit specification of start point data: vector3z • With Attach continuously: vector1z
Second derivative at end point X	Dependent on **Polynomial specified via** parameter: • With Explicit specification of start point data: vector4x • With Attach continuously: vector2x
Second derivative at end point Y	Dependent on **Polynomial specified via** parameter: • With Explicit specification of start point data: vector4y • With Attach continuously: vector2y
Second derivative at end point Z	Dependent on **Polynomial specified via** parameter: • Explicit specification of start point data: vector4z • Attach continuously: vector2z
Coefficient a2 X	vector1x

Parameters of the MCC command Traverse path using polynomials	Parameters of the system function _movePathPolynomial
Coefficient a2 Y	vector1y
Coefficient a2 Z	vector1z
Coefficient a3 X	vector2x
Coefficient a3 Y	vector2y
Coefficient a3 Z	vector2z
Coefficient a4 X	vector3x
Coefficient a4 Y	vector3y
Coefficient a4 Z	vector3z
Coefficient a5 X	vector4x
Coefficient a5 Y	vector4y
Coefficient a5 Z	vector4z
Blending tab	
Blending segment	transitionType
Blending distance	blendingDistance
Velocity in the blending segment	transitionVelocityMode:
Dynamic response tab	
Velocity	velocityType, velocity
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Dynamic response adaptation	dynamicAdaption
Specific velocity profile	specificVelocityProfile
Cam	profileReference
Profile position at start of path	profileStartPosition
Profile position at end of path	profileEndPosition
Synchronous axis tab	
Synchronous axis mode	wMode
Synchronous axis direction	wDirection
Synchronous axis position	w
Expert tab	
CommandID variable	commandId
Return variable	–

Stop path motion

This function is available in SIMOTION Kernel as of Version 4.1.

This command stops the current path motion (including the path-synchronous motion) with the programmed dynamic response parameters (deceleration ramp). You can choose whether the motion is aborted:

- Stop without abort: The motion can be continued with the Continue path motion (Page 4510) command.
- Stop with abort: The motion cannot be continued.

The last calculated positions of each path axis and the synchronous axis (setpoint values) are stored in the abortPosition system variable.

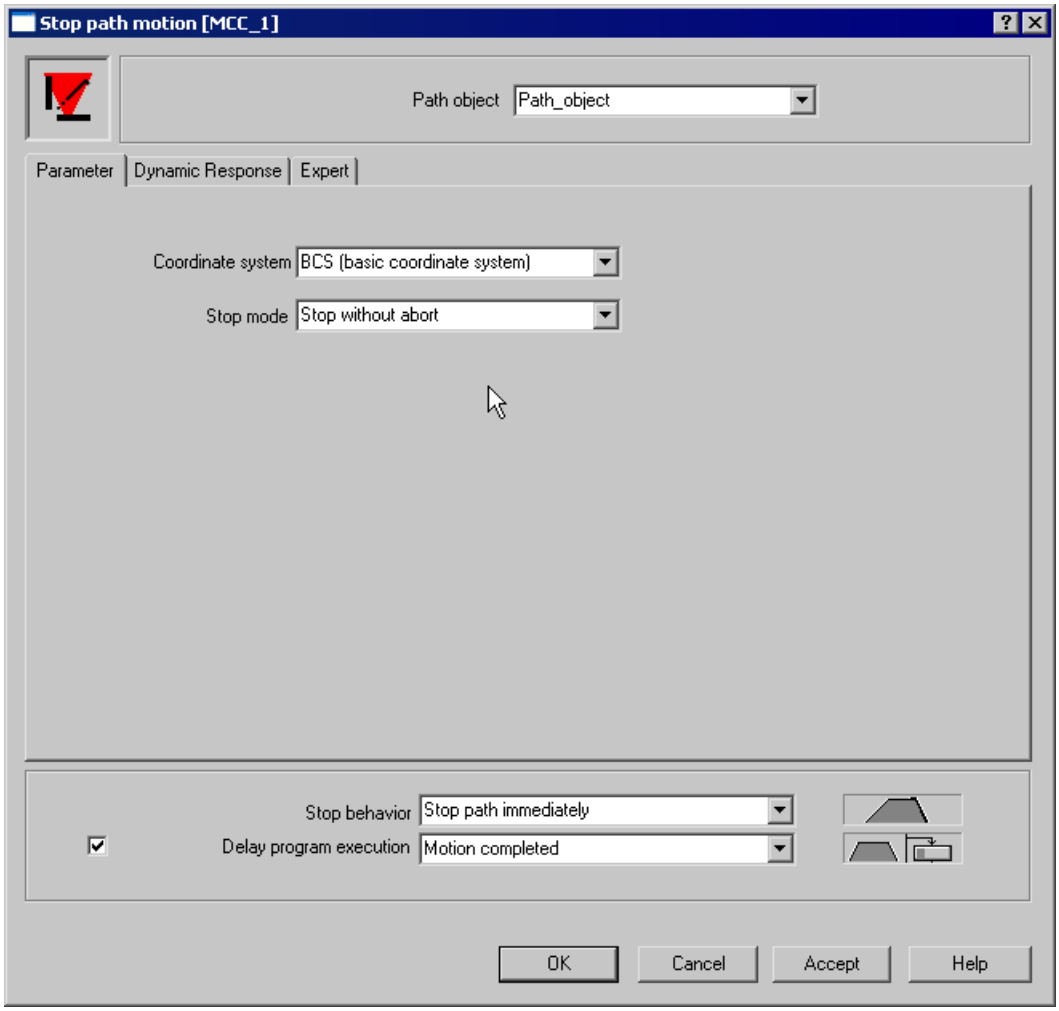


Figure 7-177 Parameter screen form: Stop path motion

For the **fundamentals of path interpolation**, see the TO Path Object Function Manual

For further information on **stopping and continuing path motion**, see the TO Path Object Function Manual.

Overview of parameters for Stop path motion

Table 7-331 Overview of parameters for Stop path motion

Field/button	Meaning/instruction
Path object	In Path object, select the path object whose assigned axes (path axes and synchronous axis) are to be stopped. The list contains: • All path objects that are defined on the relevant device. The path objects are defined in the PATH OBJECTS folder in the project navigator. • All variables with the following data type of a technology object declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): _PathObjectType.
Parameters tab	See Overview of parameters for Stop path motion - Parameters tab (Page 4508)
Dynamic response tab	See Overview of parameters for Stop path motion - Dynamic response tab (Page 4509)
Expert tab	See Overview of parameters for Stop path motion - Expert tab (Page 4509)
Stop behavior	Specifies when the command takes effect relative to the motion. **Stop path immediately** The active path motion is substituted immediately. All motion commands still pending in the command buffer are deleted. **Stop at the end of the path command.** Not relevant.
Delay program execution	• Activate the checkbox if execution of the following command in the MCC chart should be delayed until the selected condition has been satisfied. If the checkbox is not activated, the next command is executed immediately. • Select the condition that has to be satisfied before the system continues execution of the following command in the MCC chart. See also Delay program execution (step enabling condition) (Page 4037).

Overview of parameters for Stop path motion – Parameters tab

Table 7-332 Overview of parameters for Stop path motion – Parameters tab

Field/button	Meaning/instruction
Coordinate system	This field is available as of SIMOTION Kernel version V4.3. Here you select the coordinate system to which the path motion to be stopped refers. **BCS (basic coordinate system)** (default value) The motion is stopped in the basic coordinate system. **OCS (object coordinate system)** The motion is stopped in the currently effective object coordinate system.
Stop mode	Select a stop mode here. The path motion is stopped with the programmed dynamic response parameters (Dynamic response tab (Page 4509)). The last calculated positions (setpoint values) of the path axes and the synchronous axis assigned to the path object are stored in the abortPosition system variable. **Stop without abort** (default value) The motion can be continued with the **Continue path motion** command. **Stop with abort** The motion cannot be continued.

Overview of parameters for Stop path motion – Dynamic response tab

Table 7-333 Overview of parameters for Stop path motion – Dynamic response tab

Field/Button	Explanation/Instructions
	The Dynamic response tab is described in detail in Overview of parameters - Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefault.pathDynamics.profile
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefault.pathDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.pathDynamics.positiveAccelEndJerk userDefault.pathDynamics.negativeAccelStartJerk userDefault.pathDynamics.negativeAccelEndJerk

Overview of parameters for Stop path motion – Expert tab

Table 7-334 Overview of parameters for Stop path motion – Expert tab

Field/Button	Explanation/Instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system functions for Stop path motion

Path technology package:

- _stopPath

Overview of parameters for Stop path motion / _stopPath

Table 7-335 Parameters (MCC command Stop path motion compared to system function _stopPath)

Parameters of the MCC command Stop path motion	Parameters of the system function _stopPath
Path object	pathObject
Stop behavior	mergeMode
Delay program execution	nextCommand

Parameters of the MCC command Stop path motion	Parameters of the system function _stopPath
Parameters tab	
Coordinate system	csType
Stop mode	stopMode
Dynamic response tab	
Velocity profile	velocityProfile
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk
Expert tab	
CommandID variable	commandId
Return variable	–

Continue path motion

This function is available in SIMOTION Kernel Version 4.1 and higher.

This command continues a path motion that has been stopped. It is only possible to continue path motions that were stopped with the **Stop without abort** stop mode.

During the time between the interruption and the continuation, the path object and all axes (path axes and axes for path-synchronous motion) assigned to it are not permitted to receive any new motion commands.

The following applies for the dynamic response parameters when continuing the path motion:

- The following applies up to version V4.2 of the SIMOTION Kernel:
  The dynamic response parameters of the stopped path motion are used.
- The following applies as of version V4.3 of the SIMOTION Kernel:
  Either the dynamic response parameters of the stopped path motion or specific dynamic response parameters (e.g. for velocity profile, acceleration, jerk) are used.

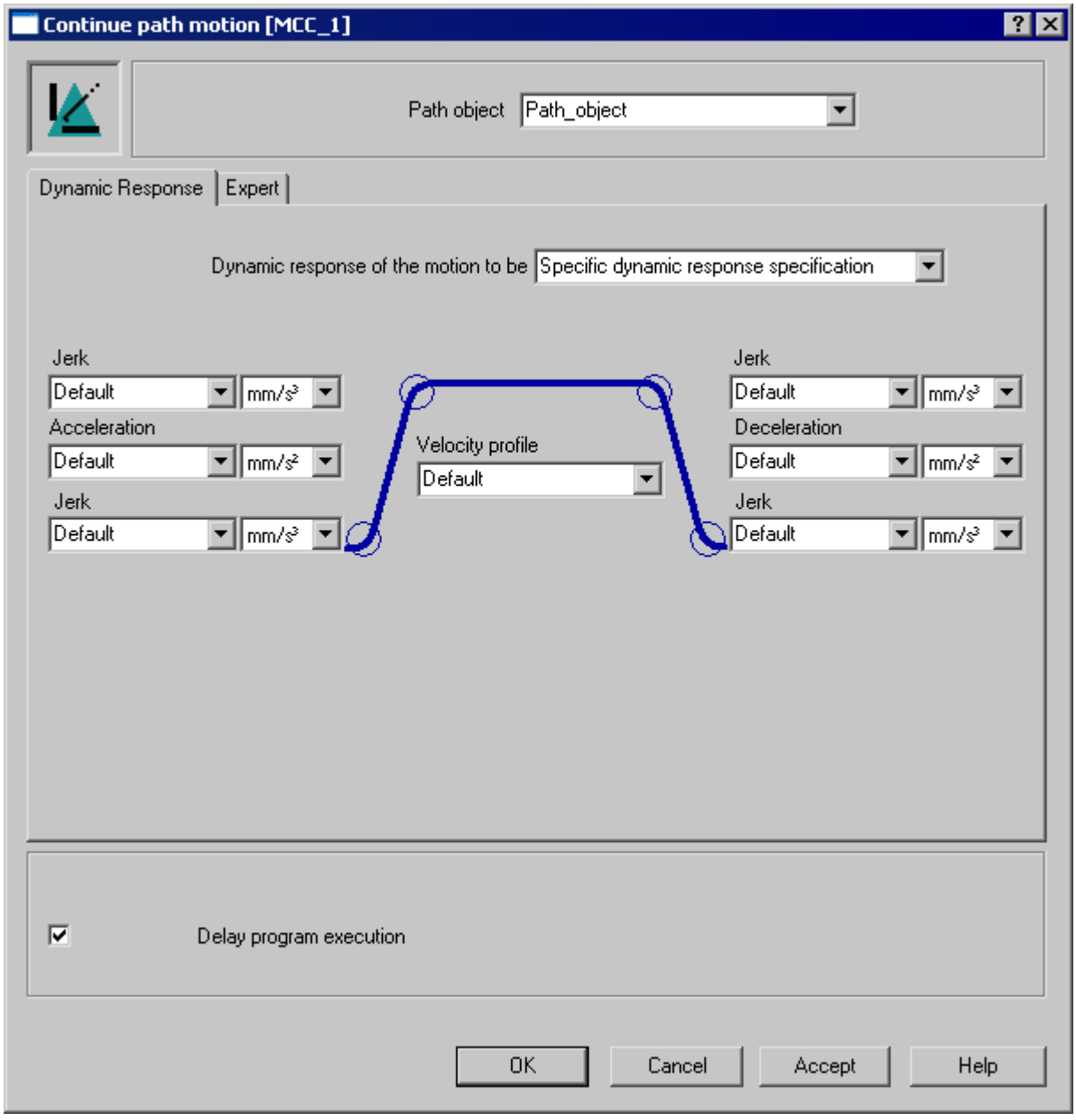


Figure 7-178 Parameter screen form: Continue path motion

For the **fundamentals of path interpolation**, see the TO Path Object Function Manual

For further information on **stopping and continuing path motion**, see the TO Path Object Function Manual.

Overview of parameters: Continue path motion

Table 7-336 Overview of parameters: Continue path motion

Field/button	Meaning/instruction
Path object	In Path object, select the path object for which the motion of the assigned axes (path axes and synchronous axis) is to be continued. The list contains: • All path objects that are defined on the relevant device. The path objects are defined in the PATH OBJECTS folder in the project navigator. • All variables with the following data type of a technology object declared in the MCC unit or MCC chart, see Technology object data types (Page 4060): _PathObjectType.
Dynamic response tab	See Overview of parameters for Continue path motion - Dynamic response tab (Page 4512)
Expert tab	See Overview of parameters for Continue path motion - Expert tab (Page 4513)
Delay program execution	Activate the checkbox if you wish to delay execution of the following command until the current command has been completed. If the checkbox is not activated, the next command is executed immediately.

Overview of parameters for Continue path motion - Dynamic response tab

Table 7-337 Overview of parameters for Continue path motion - Dynamic response tab

Field/button	Meaning/instruction
	This tab is available as of SIMOTION Kernel version V4.3.
Dynamic response of the motion to be continued	Here, you specify which dynamic response parameters are used to when continuing the stopped path motion. **Dynamic response of the interrupted motion** (default value) All dynamic response parameters the stopped path motion are used. This corresponds to the behavior up to version V4.2 of the SIMOTION Kernel. **Specific dynamic response specification** Velocity profile, acceleration, deceleration and jerk are specified for the motion to be continued. The corresponding input fields are displayed. The other dynamic response parameters are taken from the stopped path motion.
	The following fields are only displayed if you have selected the following in the **Dynamic response of the motion to be continued** field: • Specific dynamic response specification A detailed description of these parameters is provided in Overview of parameters – Dynamic response tab (Page 4031).
Velocity profile	In this field, you define the transitions between the individual motion phases. System variable for default: userDefault.pathDynamics.profile
Acceleration	The entered value acts during the constant acceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefault.pathDynamics.positiveAccel

Field/button	Meaning/instruction
Deceleration	The entered value acts during the constant deceleration phase. The meaning depends on the acceleration model used (see the general description of the Dynamic response tab (Page 4031)). System variable for default: userDefault.pathDynamics.negativeAccel
Jerk	The entered value limits the change in the acceleration or deceleration at the start and end of the respective phase. System variables for default: userDefault.pathDynamics.positiveAccelStartJerk userDefault.pathDynamics.positiveAccelEndJerk userDefault.pathDynamics.negativeAccelStartJerk userDefault.pathDynamics.negativeAccelEndJerk

Overview of parameters for Continue path motion - Expert tab

Table 7-338 Overview of parameters for Continue path motion - Expert tab

Field/button	Meaning/instruction
	The Expert tab is described in detail in Overview of parameters for Expert tab (Page 4034).
CommandID variable	If you enter the name of a variable of data type CommandIdType, you can track the command status with this variable.
Return variable	If you enter the name of a variable of the specified data type, you can use this variable to find out the result of the command call. Data type DINT; for a description, see Return value for system functions of the technology packages (Page 4040).

Relevant system function for Continue path motion

Path technology package:

- _continuePath

Overview of parameters for Continue path motion / _continuePath

Table 7-339 Parameters (MCC command Continue path motion compared to system function _continuePath)

Parameters of the MCC command Continue path motion	Parameters of the system function _continuePath
Path object	pathObject
Delay program execution	nextCommand
Dynamic response tab	
Dynamic response of the motion to be continued	continueDynamicType
Velocity profile	velocityProfile
Acceleration	positiveAccelType, positiveAccel
Deceleration	negativeAccelType, negativeAccel
Jerk	positiveAccelStartJerkType, positiveAccelStartJerk, positiveAccelEndJerkType, positiveAccelEndJerk, negativeAccelStartJerkType, negativeAccelStartJerk, negativeAccelEndJerkType, negativeAccelEndJerk

Parameters of the MCC command Continue path motion	Parameters of the system function _continuePath
Expert tab	
CommandID variable	commandId
Return variable	–

7.1.7 Commissioning (software)

7.1.7.1 Assigning programs to a task and downloading them to the target system

assigning programs to a task

Programs must be assigned to a task before they can be downloaded to the target system (the SIMOTION device).

Various tasks are made available by SIMOTION, each with different priorities or system responses (e.g. during initialization). The Execution levels and tasks in SIMOTION table contains a brief description of the available tasks.

For further information, refer to the SIMOTION Basic Functions Function Manual.

Note

Programs need only be assigned to a task once; this assignment is retained if the program is recompiled.

Assigning programs to a task:

1. In the project navigator, double-click the EXECUTION SYSTEM element below the relevant SIMOTION device.
   The window for configuring the execution system is displayed. See figure below.
2. Select the required task (e.g. MotionTask_1) from the left pane.
3. Select the Program assignment tab.
4. Select the program to be assigned from the Programs list.
5. Click the >> button.
6. Select the Task Configuration tab, and specify any other required settings for the task there.

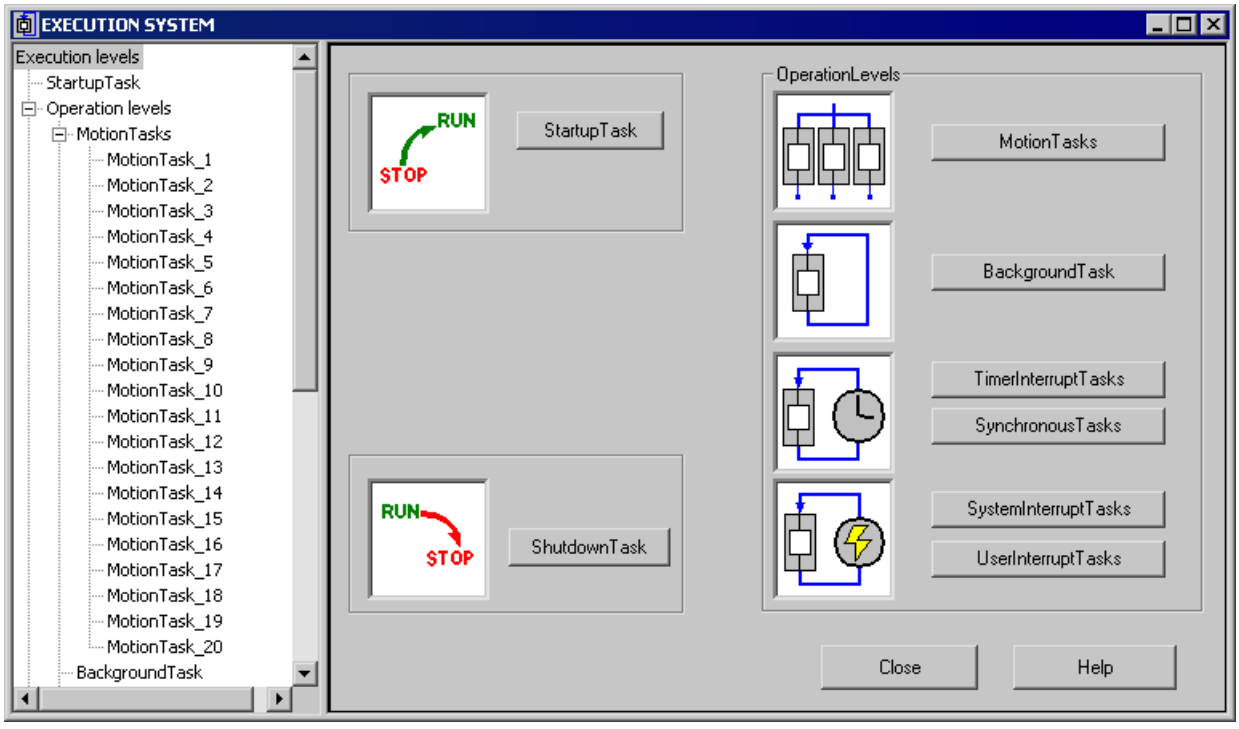


Figure 7-179 Configure execution system

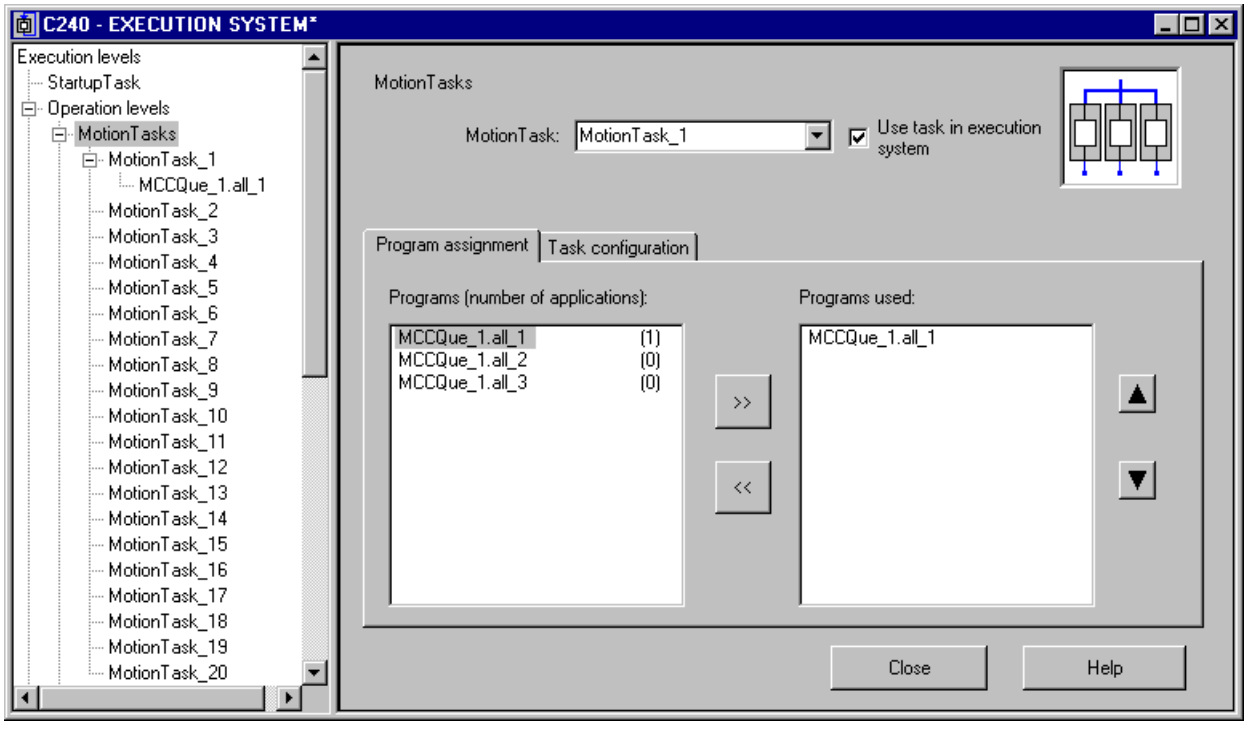


Figure 7-180 Assigning a program to a motion task

Execution levels and tasks in SIMOTION

Here the execution levels and the tasks assigned are shown in tabular format for an initial overview.

Further information on execution levels and tasks can be found in the SIMOTION Basic Functions Function Manual.

Table 7-340 Description of the execution levels and of the tasks

Execution level	Description
Time-controlled	Cyclic tasks: They are restarted automatically once the assigned programs have been executed.
• SynchronousTasks	Tasks are started periodically, synchronous with specified system cycle clock. • ServoTask_Fast: Synchronous with Servo_fast cycle clock. The Servo_fast cycle clock is a second servo cycle clock and only available: – For D445-2 DP/PN and D455-2 DP/PN as of version V4.2 – For D435-2 DP/PN as of version V4.3. • ServoSynchronousTask: Synchronous with the position control cycle clock • IpoTask_Fast: Synchronous with IPO_fast cycle clock. The IPO_fast cycle clock is the IPO cycle clock for the second servo cycle clock and only available: – For D445-2 DP/PN and D455-2 DP/PN as of version V4.2 – For D435-2 DP/PN as of version V4.3. • IPOsynchronousTask: Synchronous with interpolator cycle clock IPO • IPOsynchronousTask_2: Synchronous with interpolator cycle clock IPO_2 • PWMsynchronousTask: Synchronous with PWM cycle clock (for TControl technology package) • InputSynchronousTask_1: Synchronous with Input1 cycle clock (for TControl technology package) • InputSynchronousTask_2: Synchronous with Input2 cycle clock (for TControl technology package) • PostControlTask_1: Synchronous with Control1 cycle clock (for TControl technology package) • PostControlTask_2: Synchronous with Control2 cycle clock (for TControl technology package)
• TimerInterruptTasks	Tasks are started periodically in a fixed time frame. This time frame must be a multiple of interpolator cycle clock IPO.
Interrupts	Sequential tasks: They are executed once after the start and then terminated.
• SystemInterruptTasks	Started when a system event occurs: • ExecutionFaultTask: Error processing a program • PeripheralFaultTask: Error on I/O • TechnologicalFaultTask: Error on the technology object • TimeFaultBackgroundTask: BackgroundTask timeout • TimeFaultTask: TimerInterruptTask timeout
• UserInterruptTasks	They are started when a user-defined event occurs.

Execution level	Description
Round robin	MotionTasks and BackgroundTasks share the free time remaining after execution of the higher-priority system and user tasks. The proportion of the two levels can be assigned.
• MotionTasks	Sequential tasks: They are executed once after the start and then terminated. Start takes place: • Explicitly via a task control command in a program assigned to another task. • Automatically when RUN operating state is attained if the corresponding attribute was set during task configuration. The priority of a MotionTask can be increased temporarily: • In the MCC programming language with the "Wait for..." commands, see Wait for axis (Page 4175), Wait for signal (Page 4178), Wait for condition (Page 4180). • In the ST programming language with the WAITFORCONDITION statement.
• BackgroundTask	Cyclic task: It is restarted automatically once the assigned programs have been executed. The task cycle time depends on the runtime.
StartupTask	Task is executed once when there is a transition from STOP or STOP U operating state to RUN operating state. SystemInterruptTasks are started by their triggering system event.
ShutdownTask	Task is executed once when there is a transition from RUN operating state to STOP or STOP U operating state. STOP or STOP U operating state is reached by: • Activating the operating state switch • Calling the relevant system function, for example, MCC Change operating state command • Occurrence of a fault with the appropriate error response SystemInterruptTasks and PeripheralFaultTasks are started by their triggering system event.
For information on the behavior of sequential and cyclic tasks: • During initialization of local program variables: See Initialization of local variables (Page 4077). • In the event of execution errors in the program: See SIMOTION Basic Functions Function Manual. For information about options for accessing the process image and I/O variables: see Important properties for direct access and process image (Page 4091).	

Task start sequence

When the StartupTask is completed, RUN mode is reached.

The following tasks are then started:

- SynchronousTasks
- TimerInterruptTasks

- BackgroundTask
- MotionTasks with startup attribute.

Note

The sequence in which these tasks are first started after RUN mode has been reached does not conform to the task priorities.

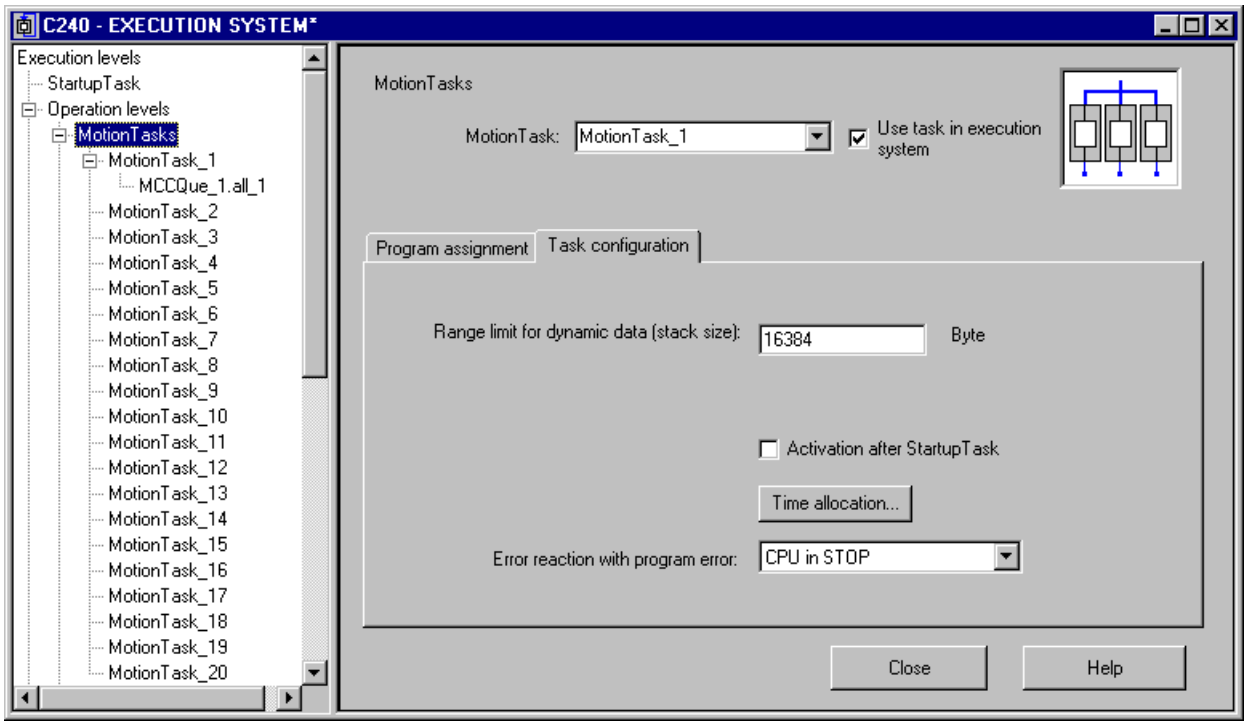


Figure 7-181 Task configuration for a MotionTask

downloading programs to the target system

The program has to be downloaded into the target system, together with the technology objects etc., before being executed.

To download the program to the target system, proceed as follows:

1. Select the **Project > Save and compile changes** menu command.
   The project is saved locally on the hard disk with due regard for all dependencies and only those changes made since the last compilation are compiled.
2. Select the **Project > Check consistency** menu command to check the project for consistency. This is not necessary if the option **Check consistency before loading** is activated in menu option **Options > Settings** in the **Download** tab (the default for this option is to be activated). This means the consistency check is performed automatically during the download to the target system.

3. Select the **Project > Connect to selected target devices** menu command or click the button.
   Online mode is activated.
4. Select the **Target system > Load > Project to target system** menu command or click the button.
   The project data (including the sample program) and the data of the hardware configuration are downloaded to the RAM of the target system.

For more information about downloading a program to the target system, see the SIMOTION Basic Functions Function Manual.

7.1.8 Error Handling and Program Test

7.1.8.1 Operating modes for program testing

Modes of the SIMOTION devices

Various SIMOTION device operating modes are available for program testing.

Table 7-341 Operating modes of a SIMOTION device

Operating mode	Meaning
Process mode	Program execution on the SIMOTION device is optimized for maximum system performance. The following diagnostic functions are available, although they may have only restricted functionality because of the optimization for maximum system performance: • Monitor variables in the symbol browser (Page 4525) or a watch table (Page 4529) • Program status (Page 4542) (only restricted): – Restricted monitoring of variables (e.g. variables in loops, return values for system functions). – Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored. • Trace tool (only restricted) with measuring functions for drives and function generator, see online help: – Maximum of **1** trace on each SIMOTION device.
Test mode	The diagnostic functions of the process mode are available to the full extent: • Monitor variables in the symbol browser (Page 4525) or a watch table (Page 4529) • Program status (Page 4542): – Monitoring of all variables possible. – As of version V4.0 of the SIMOTION Kernel: Several program sources (e.g. ST source files, MCC units, LAD/FBD units)[1] can be monitored per task. – For version V3.2 of the SIMOTION Kernel: Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored per task. • Trace tool with measuring functions for drives and function generator, see online help: – Maximum of **4** traces on each SIMOTION device. In addition, the following diagnostics function is available: • Trace (Page 4538) for monitoring the program execution in program branches which are executed cyclically (only for the MCC programming language and for SIMOTION Kernel V4.2 and higher). **Note** Runtime and memory utilization increase as the use of diagnostic functions increases.

Operating mode	Meaning
Debug mode	In addition to the diagnostic functions of the test mode, you can use the following functions: • Breakpoints Within a program source, you can set breakpoints (Page 4550). When an activated breakpoint is reached, selected tasks will be stopped. • Controlling MotionTasks On the "Task Manager" tab of the device diagnostics, you can use task control commands for MotionTasks; see the SIMOTION Basic Functions Function Manual. No more than 1 SIMOTION device of the project can be switched to debug mode. SIMOTION SCOUT is in online mode, i.e. connected to the target system. Observe the following section: Important information about the life-sign monitoring (Page 4522).

[1] Each with 1 MCC chart or 1 LAD/FBD program in a program source.

Selecting the operating mode

How to select the operating mode of a SIMOTION device:

1. Make sure a connection to the target system has been established (online mode).
2. Highlight the SIMOTION device in the project navigator.
3. Select the "Operating mode" context menu.
4. Select the required operating mode (see the table above).
   If you have selected "Debug mode":
   – Accept the safety information.
   – Parameterize the sign-of-life monitoring.

   Observe the following section: Important information about the life-sign monitoring (Page 4522).
5. Confirm with **OK**.
   The SIMOTION device switches to the selected operating mode (apart from with debug mode; see the explanation below).

Special features with debug mode

Debug mode can only be selected for one SIMOTION device.

If you have selected debug mode, only SIMOTION SCOUT switches to it; the SIMOTION device is in test mode.

- The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.
- The breakpoints toolbar (Page 4555) is displayed.

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device.

The status bar indicates that debug mode is activated for the SIMOTION device.

Important information about the life-sign monitoring.

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode or a control panel only with the life-sign monitoring function activated with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

In the following cases, the SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection:

- In debug mode with activated breakpoints.
- When controlling an axis or a drive via the control panel (control priority at the PC):

If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis is brought to a standstill.
  - The enables are reset.

Accept safety notes

After selecting the debug mode or a control panel, you must accept the safety notes. You can set the parameters for the life-sign monitoring.

Proceed as follows:

1. Click the **Settings** button.
   The "Debug Settings" window opens.
2. Read there, as described in the following section, the safety notes and parameterize the life-sign monitoring.

Parameterizing the life-sign monitoring

In the "Life-Sign Monitoring Parameters" window, proceed as described below:

1. Read the warning!
2. Click the **Safety notes** button to open the window with the detailed safety notes.
3. Do not make any changes to the defaults for life-sign monitoring.
   Changes should only be made in special circumstances and in observance of all danger warnings.
4. Click **Accept** to confirm you have read the safety notes and have correctly parameterized the life-sign monitoring.

Note

The life-sign monitoring also responds in the following cases:

- Pressing the spacebar.
- Switching to a different Windows application.
- Too high a communication load between the SIMOTION device and SIMOTION SCOUT (e.g. by uploading task trace data).

The following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis or the drive is brought to a standstill.
  - The enables are reset.

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Life-sign monitoring parameters

Table 7-342 Life-sign monitoring parameter description

Field	Description
Life-sign monitoring	The SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection. If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered: • In debug mode for activated breakpoints: – The SIMOTION device switches to the STOP operating state. – The outputs are deactivated (ODIS). • For controlling an axis or a drive using the control panel (control priority for the PC): – The axis is brought to a standstill. – The enables are reset. The following parameterizations are possible: • Checkbox **active**: If the checkbox is activated, life-sign monitoring is active. The deactivation of the life-sign monitoring is not always possible. • Monitoring time: Enter the timeout. **Prudence** Do not make any changes to the defaults for life-sign monitoring, if possible. Changes should only be made in special circumstances and in observance of all danger warnings.
Safety information	**Please observe the warning!** Click the button to obtain further safety information. See: Important information about the life-sign monitoring (Page 4522)

7.1.8.2 Editing program sources in online mode

Online editing in process or test mode

If SIMOTION SCOUT is connected to a target system which is in the "process mode" or "test mode" operating mode, program sources (e.g. ST source files, MCC units with MCC charts) can generally be edited, compiled, and loaded to the target system in STOP operating mode. For information on downloading in RUN operating mode, see the corresponding section in the "SIMOTION Basic Functions" Function Manual.

However, you can only activate the "program status", "monitor program execution" (only for MCC), and trace (only for MCC) test functions for a program source or a program organization unit (POU) if the following conditions are met:

1. This program source or any POU of this source (e.g. MCC chart) does not contain any changes which have not been saved.
2. The program source (unit) in SCOUT is consistent with the target system.

Note

If the "program status" test function is activated, editing of the corresponding program source or one of its POUs is disabled.

If an MCC unit or MCC chart is changed and the "monitor program execution" or trace test functions are active for that unit or chart, the test functions are canceled.

Online editing in debug mode

If SIMOTION SCOUT is in debug mode, editing is possible as long as the SIMOTION device is not in debug mode, i.e. no breakpoints are activated.

You can only activate breakpoints and, as a result, switch the SIMOTION device to debug mode if the corresponding program source and all its POUs are saved, compiled so they are up to date, and consistent with the target system.

If you attempt to edit a program source or POU when the SIMOTION device is in debug mode, you are requested to deactivate all breakpoints and, as a result, to switch the SIMOTION device out of debug mode.

Note

If breakpoints have been activated and the SIMOTION device is in debug mode:

Entering a space switches the SIMOTION device to STOP operating mode and deactivates all outputs (ODIS).

7.1.8.3 Monitoring variables in the symbol browser and watch tables

Symbol browser

Characteristics

In the symbol browser, you can view and, if necessary, change the name, data type, and variable values. You can see the following variables in particular:

- Unit variables and static variables of a program or function block
- System variables of a SIMOTION device or a technology object
- I/O variables or global device variables.

For these variables, you can:

- View a snapshot of the variable values
- Monitor variable values as they change
- Change (modify) variable values

However, the symbol browser can only display/modify the variable values if the project has been loaded in the target system and a connection to the target system has been established.

Using the symbol browser

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system (see Download programs to the target system (Page 4518)).
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

The procedure depends on the memory area in which the variables to be monitored are stored.

Procedure

Proceed as follows:

1. Select the appropriate element in the project navigator in accordance with the following table.
2. In the detail view, click the **Symbol browser** tab.
   The corresponding variables are displayed in the symbol browser.
3. Select how each variable in the "Display format" column should be displayed.

Table 7-343 Elements in the project navigator and variables to be monitored in the symbol browser

Variables to be monitored in the symbol browser	Element to be selected in the project navigator
Variables in the unit's user memory or in the retentive memory, see SIMOTION ST Programming and Operating Manual: • Retentive and non-retentive unit variables of the interface section of a program source (unit) • Retentive and non-retentive unit variables of the implementation section of a program source (unit) • Static variables of the function blocks whose instances are declared as unit variables. • In addition, if the program source (unit) has been compiled with the "Only create program instance data once" compiler option (Page 3996): – Static variables of the programs. – Static variables of the function blocks whose instances are declared as static variables of programs.	Program source (unit)
Variables in the user memory of the task, see SIMOTION ST Programming and Operating Manual: If the program source (unit) was compiled without the "Only create program instance data once" (default) compiler option (Page 3996), the user memory of the task to which the program was assigned contains the following variables: • Static variables of the programs. • Static variables of the function blocks whose instances are declared as static variables of programs.	EXECUTION SYSTEM
System variables of a SIMOTION device	SIMOTION device
System variables of a technology object	Instance of the technology object
Global device variables	GLOBAL DEVICE VARIABLES
I/O variables (in the Address list tab of the detail view). The Address list tab of the detail view can be opened by double-clicking the ADDRESS LIST element in the project navigator.	ADDRESS LIST

Note

You can monitor temporary variables (together with unit variables and static variables) with **Program status** (see Properties of the program status (Page 4542)).

Note

Trace diagnostic function for MCC programming

Various internal variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. These variables are displayed in the symbol browser.

With activated diagnostic function, these variables are used for the control of the diagnostics function. These variables must not be used in the user program.

Status and controlling variables

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox in this column
3. Click the **Immediate control** button.

The values you entered are written to the selected variables.

⚠ WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the symbol browser. There is therefore no guarantee of consistency.

Working with the symbol browser

The functions of the symbol browser and how you work with them are described in detail in the online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the symbol browser (independently of the SIMOTION device):

Table 7-344 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

Watch table

Properties

With the symbol browser you see only the variables of an object within the project. With program status you see only the variables in certain MCC commands.

With watch tables, by contrast, you can monitor selected variables from different sources as a group (e.g. program sources, technology objects, SINAMICS drives - even on different devices).

You can see the data type of the variables in offline mode. You can view and also modify the value of the variables in online mode.

Creating a watch table

Procedure for creating a watch table and assigning variables:

1. In the project navigator, select the **Monitor** folder.
2. Select **Insert new object > Watch table** to create a watch table, and enter the name of the watch table. A watch table with this name appears in the **Monitor** folder.
3. In the project navigator, click the object from which you want to move variables to the watch table.
4. In the symbol browser, select the corresponding variable line by clicking its number in the left column.
5. From the context menu, select **Add to watch table** and the appropriate watch table, e.g. **Watch table_1**.
6. If you click the watch table, you will see in the detail view of the **Watch table** tab that the selected variable is now in the watch table.
7. Repeat steps 3 to 6 to monitor the variables of various objects.

If you are connected to the target system, you can monitor the variable contents.

Status and controlling variables

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox in this column
3. Click the **Immediate control** button.

The values you entered are written to the selected variables.

⚠ WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the watch table. There is therefore no guarantee of consistency.

Working with the watch table

The functions of the watch table and how you work with them are described in detail in the Online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the watch table (independently of the SIMOTION device):

Table 7-345 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN - Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

7.1.8.4 Variable status

"Variable status" enables you to monitor the current value for an individual variable, selected using the cursor, in an open program source or program organization unit (e.g. ST source file, MCC chart, LAD program).

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system. For information on loading a project, see "downloading programs to the target system (Page 4518)".
- The program source containing the program organization unit (POE) whose variables you want to monitor must be consistent with the target system.

- The associated source (e.g. ST source file, MCC chart, LAD program) must be open.
- With the MCC programming language only: The parameter screen form for the command in which the variable you want to monitor is being used must be open.
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

Procedure

To monitor an individual variable using variable status:

1. Position the cursor above the identifier for a variable.
   - With the ST programming language: in the open ST source file
   - With the ST programming language: within an input field in the open parameter screen form
   - With the LAD/FBD programming language: within a network of the LAD/FBD program
2. Briefly position the cursor above the identifier.

The tool tip shows the current value of the variable. If you keep the cursor above the identifier for a longer period, the value is updated on an ongoing basis.

Note

With "variable status", the current value for the variable is displayed, wherever the selected variable is being used.

The "variable status" function enables you to monitor all those variables you are also able to monitor in the symbol browser (Page 4526) or the address list. These are:

- System variables of SIMOTION devices
- System variables of technology objects
- Global device variables
- Retentive and non-retentive unit variables of the interface section of a program source (unit)
- Retentive and non-retentive unit variables of the implementation section of a program source (unit)
- Static variables of the programs
- Static variables of the function blocks whose instances are declared as unit variables
- Static variables of the function blocks whose instances are declared as static variables of programs
- I/O variables

7.1.8.5 Monitoring the program execution

Tracking program execution via monitoring

Monitoring makes it possible to track program execution for sequential tasks, as these are low-priority tasks in the execution system, with predominantly "slow" commands being executed.

Tracking the execution of programs in sequential tasks does not affect the actual execution of the program, but does increase the communication load. This has an impact on the execution of MotionTasks and the BackgroundTask.

Program execution tracking can be switched on and off during program operation.

When program execution monitoring is enabled, the current active command is shown in yellow. The parameter screen forms of the active MCC chart can be opened and read, but no changes can be made.

Starting the program execution monitoring

Monitoring is started via an open MCC chart which is visible in the foreground and assigned to a task. The monitoring function, however, is not started at program level but at task level. This automatically starts the monitoring for all MCC charts assigned to this task.

If the MCC chart is assigned to multiple tasks, you can select whether the monitoring should be started for one, several or all of these tasks. This automatically starts the monitoring for all MCC charts assigned to the selected tasks.

Requirement

- The MCC chart is open and visible in the foreground.
- The MCC chart is assigned directly to one or more tasks as a program or indirectly to one or more tasks via the calling program as a subprogram. This indirect assignment can also be multistage if the subprogram is called in another subprogram, or up to the subprogram that is called in a program. The calling POUs can be created in the ST, MCC or LAD/FBD programming language.

Procedure

To start the monitoring function in the open MCC chart, proceed as follows:

- **The MCC chart is only assigned to one task**
  - Select the **MCC chart > Monitor** menu command or click the button.
    The monitoring of the program execution is started for this task and thereby for all MCC charts assigned to this task.
    The relevant task is highlighted in yellow ( ) in the "Task status" toolbar (Page 4564).
- **The MCC chart is assigned to more than one task**
  - Select the **MCC chart > Monitor** menu command or click the button.
    The "Selection of the Task To Be Monitored" dialog box opens.
  - By activating the relevant checkbox, select the tasks in which you want to monitor the program in this dialog box.
    The monitoring of the program execution is started for the selected tasks and thereby for all MCC charts assigned to these tasks.
    The relevant tasks concerned are highlighted in yellow ( ) in the "Task status" toolbar (Page 4564).
  - In the "Task status" function bar, select a task in which you want to monitor program execution.
    You can switch to another task at any time by selecting the required task in the "Task status" function bar.

The MCC_1 symbol on the tab for an open MCC chart indicates that monitoring has begun, as does the symbol for the combination of currently activated test functions (monitoring, single step, trace, program status, breakpoints).

MCC charts that are called as subprograms within the open MCC chart can also be monitored. To do so, open the relevant charts, as well.

Stopping the program execution monitoring

To stop the monitoring function in the open MCC chart, proceed as follows:

- **The MCC chart is only assigned to one task**
  - Select the **MCC chart > Monitor** menu command or click the button.
    The monitoring of the program execution is stopped for this task and thereby for all MCC charts assigned to this task.
- **The MCC chart is assigned to more than one task**
  - Select the **MCC chart > Monitor** menu command or click the button.
    The "Selection of the Task To Be Monitored" dialog box opens.
  - By deactivating the relevant checkbox, select the tasks in which you want to stop the monitoring of the program execution in this dialog box.
    The monitoring of the program execution is stopped for the selected tasks and thereby for all MCC charts assigned to these tasks.

The marking of the current command is deleted and the icon is removed from the tab.

Tracking program execution using single step

You can track the program execution of one or more MCC charts in single steps. This function is available in the **Process** and **Test** operating modes (Page 4520).

Note

Availability in the Debug mode

The availability in the **Debug** operating mode (Page 4520) depends on the SIMOTION Kernel version:

- SIMOTION Kernel up to version V4.3:
  This single-step monitoring is available. However, it cannot be used simultaneously with activated breakpoints (Page 4550).
- SIMOTION Kernel as of version V4.4:
  This form of single-step monitoring is not available. Instead, you can resume the program execution in single steps (Page 4563) from the activated breakpoints (Page 4550).

Single-step monitoring has no influence on the program execution itself, but it does extend the execution time of the program. This function can be switched on and off during program operation.

The program can be assigned to more than one task. Single-step monitoring can be started for none, one, several or all of these tasks. The program sequence is suspended in the tasks in which single-step monitoring has been started until the user initiates the next program step. The next command to be executed is highlighted in light blue. Once the next single step has been activated, this command is executed as an active command and remains highlighted in light blue until execution of the command has been completed. The next command to be executed then turns light blue and the task is suspended again.

In the tasks in which single-step monitoring has not been started, the program sequence is not suspended and therefore the program can run normally.

Note

Additional program code is generated when the compiler option **Permit single step** is activated. This results in a slightly modified runtime behavior and may lead to timeouts.

As the execution time of the program is extended, the time watchdog must be adapted accordingly for cyclical tasks (TimerInterruptTask, BackgroundTask) or, if necessary, disabled.

Jumps in functions and function blocks can also be monitored if single-step monitoring is enabled in their MCC charts. The relevant MCC chart is then opened when the function or function block is called.

Note

Single-step monitoring is not possible within the Synchronous start (Page 4219) command.

The parameter screen forms of the active MCC chart can be opened and read, but no changes can be made.

Note

When the compiler option **Permit single step** is activated, access to the TSI#dwuser_1 and TSI#dwuser_2 of the TaskStartInfo is created automatically by the compiler. These are displayed in the cross-reference list.

With activated single-step monitoring, these variables are used for the control of the single-step monitoring. These variables must not be used in the user program.

Activating single-step monitoring

Proceed as follows in order to enable single-step monitoring for an MCC unit:

1. Open the Properties window for the MCC unit, see Properties of an MCC unit (Page 3994).
2. Activate the **Permit single step** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit.

   Note

   Alternatively, select **Options > Settings > Compiler**, then activate the **Permit single step (MCC)** compiler option as the global compiler setting (Page 3996) for all MCC units.

3. Select the MCC unit in the project navigator, followed by **Accept and compile** in the context menu.

   Note

   If the **Permit single step (MCC)** global compiler option is changed, you need to select **Project > Save and compile changes** to compile all the MCC units affected by the change.

4. Download the programs to the target system.

Starting single-step monitoring

If the MCC chart is assigned to multiple tasks, you can select whether single-step monitoring should be started for one, several or all of these tasks. As a result, single-step monitoring is started automatically for all MCC charts assigned to the selected tasks and for whose MCC units single-step monitoring is switched on.

Single-step monitoring is started via an open MCC chart which is visible in the foreground and assigned to a task. The single-step monitoring function, however, is not started at program level but at task level. Here, single-step monitoring is started automatically for all MCC charts assigned to this task and for whose MCC units single-step monitoring is switched on.

- The MCC chart is open and visible in the foreground.
- Single-step monitoring is switched on for the MCC unit of the open MCC chart.

- The MCC chart is assigned directly to one or more tasks as a program or indirectly to one or more tasks via the calling program as a subprogram. This indirect assignment can also be multistage if the subprogram is called in another subprogram, or up to the subprogram that is called in a program. The calling POUs can be created in the ST, MCC or LAD/FBD programming language.
- No activated breakpoints in **Debug mode**.

Procedure

To start the single-step monitoring function in the open MCC chart, proceed as follows:

- **The MCC chart is only assigned to one task**
  - Select the **MCC chart > Single step** menu command or click the button.
    Single-step monitoring is started for this task and thereby for all MCC charts assigned to this task and for whose MCC units single-step monitoring is switched on.
    The relevant task is highlighted in blue ( ) in the "Task status" toolbar (Page 4564).
- **The MCC chart is assigned to more than one task**
  - Select the **MCC chart > Single step** menu command or click the button.
    The "Selection of the task to be monitored" dialog box opens.
  - By activating the relevant checkbox, select the tasks in which you want to monitor the program using single-step monitoring in this dialog box.
    Single-step monitoring is started for the selected tasks and thereby for all MCC charts assigned to these tasks and for whose MCC units single-step monitoring is switched on.
    The relevant tasks are highlighted in blue ( ) in the "Task status" toolbar (Page 4564).
  - In the "Task status" function bar, select a task in which you want to monitor program execution using single-step monitoring.
    You can switch to another task at any time by selecting the required task in the "Task status" function bar.

The MCC_1 symbol on the tab for an open MCC chart indicates that single-step monitoring has begun, as does the symbol for the combination of currently activated test functions (monitoring, single step, trace, program status, breakpoints).

Initiating the next single step

You can initiate the next single step in one of the following ways:

- By clicking the **Next step** button in the MCC editor function bar
- By selecting the **MCC chart > Next step** menu command, or
- By selecting the **Next step** command in the context menu in the project navigator.

If you jump to a subprogram whose MCC unit is also compiled with the "Permit single step" option, the following applies:

- The relevant MCC chart is opened and executed in single-step mode.
- The "Subprogram call" command in the calling MCC chart is marked with a cyan-colored triangle until the subprogram has been executed.

Note

Single-step monitoring in MCC charts with know-how protection

If during single-step monitoring, an MCC chart is called whose unit is know-how protected, the following applies:

- If the MCC unit (or a chart of this unit) is not already open and the login for the MCC unit has not been registered, then the MCC chart is not opened until the password for the displayed login has been entered.
- If the password cannot be entered (e.g. **Cancel** button), the MCC chart is not opened and executed time-controlled.
  Please note: The execution time of the program is significantly extended.

Note

Single-step monitoring for several MCC charts which are assigned to different tasks

If single-step monitoring is activated for several MCC charts which are assigned to different tasks, note the following:

If you change the active window (MCC charts), you may also have to select the relevant task in the "Task status" toolbar (Page 4564).

Note

The program stays in single-step mode if the SIMOTION device goes offline or the MCC chart is closed.

Note

Program status and single-step monitoring

If you have enabled program status (Page 4542) and single-step monitoring, the monitored variables for a command are only updated once the next single step has been triggered, i.e. not until you have changed from this command to the next command to be executed.

Stopping single-step monitoring

To stop single-step monitoring in the open MCC chart, proceed as follows:

- **The MCC chart is only assigned to one task**
  - Select the **MCC chart > Single step** menu command or click the  button.
    The single-step monitoring of the program execution is stopped for this task and thereby for all MCC charts assigned to this task.
- **The MCC chart is assigned to more than one task**
  - Select the **MCC chart > Single step** menu command or click the  button.
    The "Selection of the task to be monitored" dialog box opens.
  - By deactivating the relevant checkbox, select the tasks in which you want to stop the single-step monitoring of the program execution in this dialog box.
    The single-step monitoring of the program execution is stopped for the selected tasks and thereby for all MCC charts assigned to these tasks.

Deactivating single-step monitoring

1. First stop the "Single step" function (see Stopping single-step monitoring).
2. Open the Properties window for the MCC unit (see Properties of an MCC unit (Page 3994)).
3. Deactivate the **Permit single step** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit and click **OK** to confirm.

   Note

   If the **Permit single step (MCC)** compiler option is activated as the global compiler setting (Page 3996), you can deactivate this for either one MCC unit by deactivating the relevant **Use global settings** checkbox (see Local compiler settings (Page 3997)) or for all the MCC units by selecting **Options > Settings > Compiler**.

4. Recompile the program and download it to the target system.

   Note

   If the **Permit single step (MCC)** global compiler option is changed, you need to select **Project > Save and compile changes** to compile all the MCC units affected by the change.

Tracking program execution using the trace (as of Kernel V4.2)

This function is available as of SIMOTION Kernel version V4.2.

It supports program execution tracking for cyclic tasks and cyclic programs in MotionTasks.

Cyclic tasks are high-priority tasks in the execution system, with predominantly "fast" commands being executed. Cyclic programs in MotionTasks mainly contain commands capable of being executed "fast". Tracking program execution via monitoring (Page 4532) is too slow for "fast" commands and results in display errors.

Tracking the execution of programs using the trace does not affect the actual execution of the program, but does increase the communication load. This has an impact on the execution of MotionTasks and the BackgroundTask.

Program execution tracking can be switched on and off during program operation.

If program execution monitoring is enabled via the trace, the commands for program branches executed cyclically are displayed in yellow, which results in a trace. This provides a quick initial overview of the program branches being executed, making it possible, for example, to use program status (Page 4542) in a targeted manner within the executed program branches. The parameter screen forms of the active MCC chart can be opened and read, but no changes can be made.

Note

Additional program code is generated when the compiler option **Permit trace** is activated. This results in a slightly modified runtime behavior and may lead to timeouts.

If loops are programmed in the MCC chart that are run through quickly and frequently, all the executed commands may not be registered and displayed in yellow in SIMOTION SCOUT because of the limited buffer capacity.

Activating the trace function

You can only use the trace to monitor program execution if additional code has been created during compilation:

1. Select the **Project > Connect to selected target devices** menu command or click the button.
   Online mode is activated.
2. Select the SIMOTION device, followed by **Operating mode** in the context menu.
3. Select **Test mode** or **Debug mode**.
   The trace can only be activated for these two operating modes (see Operating modes for SIMOTION devices (Page 4520)).
4. Open the Properties window for the MCC unit (see Properties of an MCC unit (Page 3994)).
5. Activate the **Permit trace** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit.
   This compiler option can also be activated as the global compiler setting (Page 3996) (same as section "Enabling single-step monitoring" in Tracking program execution using single step (Page 4534)).
6. Select the **MCC unit > Accept and compile** menu command.
   The MCC unit is compiled.
7. Select the **Target system > Load > Project to target system** menu command or click the button.
   The programs are downloaded to the target system.

Note

Additional internal variables and functions, whose identifier begins with an underscore, are automatically created by the compiler when the **Permit trace** compiler option is activated. The TSI#currentTaskId variable of the TaskStartInfo is also used. These variables and functions are displayed in the cross-reference list and partly in the symbol browser.

With activated trace, these variables and functions are used for the control of the trace. These variables and functions must not be used in the user program.

Starting the trace

You can only start the trace for one MCC chart at a time.

Requirement

- **Test mode** or **Debug mode** is selected for the SIMOTION device.
- The relevant MCC chart is open and visible in the foreground.
- The trace function is switched on for the MCC unit of the open MCC chart.
- The MCC chart is assigned directly to one or more tasks as a program or indirectly to one or more tasks via the calling program as a subprogram. This indirect assignment can also be multistage if the subprogram is called in another subprogram, or up to the subprogram that is called in a program. The calling POUs can be created in the ST, MCC or LAD/FBD programming language.

- The MCC chart must not contain any changes that have not been saved.
- The MCC unit in SCOUT must be consistent with the target system.

Procedure

To start the trace:

1. Select the **MCC chart > Trace on/off** menu command or click the button.
   The **Trace Settings** dialog box opens.
2. Make the required settings in the dialog box (see the parameter description below) and click **OK** to confirm.
3. The trace function starts.
   The trace is visualized immediately or when the selected trigger condition is entered.
   The MCC_1 symbol on the tab for the open MCC chart indicates that the trace has started, as does the symbol for the combination of currently activated test functions (monitoring, single step, trace, program status, breakpoints).

The table below contains a description of the individual parameters.

Table 7-346 Parameters for trace settings

Parameter		Description
Call path		You can enter the call path here for the MCC chart for which the trace is to be activated.
	Calling task	This drop-down list box lists all the tasks whose open MCC chart has been directly/indirectly assigned in the execution system: • You have the option to select a single task here. The trace is only activated for the MCC chart in this task. • You have the option not to select any task. The trace is activated for the MCC chart in all tasks.
	Calling program	If the MCC chart is a subprogram, all call paths of the subprogram that are found within the task selected in the "Calling task" drop-down list box or in all tasks are listed in this drop-down list box: • Only when a single task is selected: You can select a single call path here (with the name of the program source, line number, name of POU). The trace will only be activated for this time call of the subprogram. • You cannot select a call path or "all call positions as of this call level". The trace will be activated every time the subprogram is called.
Trigger		You can select the required trigger condition here: • Endless recording (default setting) • Condition A trigger condition can be specified in the form of any kind of Boolean condition, e.g. myGlobalVar_1=MCC_1.myVar_2. Global device variables, I/O variables, and exported unit variables are permitted (see Overview of variable types (Page 4044)).

Parameter	Description
Display update	You can specify here when the trace display is to be updated: • Cyclic with manual new recording (default setting) [1)] • Cyclic with cycle-controlled new recording [2)] • Time-controlled with manual recording: You can select a value, based on 100 ms steps, within a range from 100 ms to 2 s.
Activate log	If this checkbox is activated, the MCC commands executed while the "trace" function is enabled are also logged. The log is output in the detail view when the trace has been stopped. You can enter the maximum possible number for the log entries here either directly as a value or select an increment of 100. The factory setting is 1000.

1) Manual new recording means that the trace displayed so far has to be reset manually, for example, via the menu **MCC chart > Rerecord trace**.
The log is not reset.

2) With cycle-controlled new recording, the trace is reset automatically before each cycle. This also applies to loop cycles.
The log is not reset.

Visualization of the trace

The trace is visualized in the MCC chart for the currently selected task.

Note

Make sure that the calling task is also selected in the "Task status" toolbar for the visualization of the trace. If required, select the **MCC chart > Trace on/off** menu command again or click the  button.

You can switch to another task at any time by selecting the required task in the "Task status" function bar. The trace for the previously selected task is deleted and the trace for the newly selected task is visualized immediately, provided that the trace function was also started for the MCC chart in the newly selected task.

This allows separate visualization of the MCC chart cycles in the various tasks. This has no impact on logging, however.

Stopping the trace

To stop the trace:

1. Select the **MCC chart > Trace on/off** menu command or click the  button.
   The trace function stops.

The visualized trace continues to be displayed and can be manually deleted by clicking the  button.

If the logging function is activated, the log is output in the detail view:

- A log entry containing the following information is created for each executed MCC command:
  - Time stamp of the SIMOTION device
  - Task name
  - Name of the MCC unit and the MCC chart
  - Command number
  - Command type
- By double-clicking on an entry, the corresponding MCC command is selected in the MCC chart.
- The log can be saved as a text file via the context menu of the detail view.
- The log is deleted when the trace is restarted or can be deleted manually via the context menu of the detail view.

Deactivating the trace function

When the trace is deactivated, the additional code is removed and CPU resources released.

To deactivate the trace, proceed as follows:

1. First stop the trace function (see Stopping the trace).
2. Open the Properties window for the MCC unit (see Properties of an MCC unit (Page 3994)).
3. Deactivate the **Permit trace** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit and click **OK** to confirm.
   If this compiler option is activated as the global compiler setting, the global compiler setting (Page 3996) must be deactivated accordingly (same as section "Disabling single-step monitoring" in Tracking program execution using single step (Page 4534)).
4. Select the SIMOTION device, followed by **Operating mode** in the context menu.
5. Select **Process mode**.
   Program execution is optimized for this operating mode to ensure maximum performance (see Operating modes for SIMOTION devices (Page 4520)).
6. Recompile the program and download it to the target system.

7.1.8.6 Program status

Program status can be used to monitor variables for the following commands.

- IF: Program branch (Page 4206)
- WHILE: Loop with condition at the start (Page 4208)
- UNTIL: Loop with condition at end (Page 4211)
- Variable assignment (Page 4187)
- ST zoom (Page 4188)
- Subprogram call (Page 4184)
- System function call (Page 4185)

The values of the following variables are displayed:

- Variables with elementary data type (INT, REAL, etc.)
- Individual elements of a structure, provided an assignment is given
- Individual elements of a field (array), provided an assignment is given
- Variables with enumeration data types

Note

The values of constants are not displayed.

In addition, the following values are not displayed for the IF (Page 4206), WHILE (Page 4208), UNTIL (Page 4211) or "Variable assignment" (Page 4187) commands that have been programmed in the LAD and FBD programming languages, see LAD/FBD/Formula (Page 4152):

- Variables of enumeration data types (Page 4059)
- Hexadecimal or binary values

Due to the restricted buffer capacity and the requirement for minimum runtime tampering, the following variables cannot be displayed:

- Complete arrays
- Complete structures

However, individual array elements or individual structure elements are displayed if an assignment is made in the MCC chart.

Program status requires additional CPU resources.

Activating program status

Before you can work with program status, additional code must be generated during compilation:

1. Select the **Project > Connect to selected target devices** menu command or click the button.
   Online mode is activated.
2. Select the SIMOTION device, followed by **Operating mode** in the context menu.
3. Select **Test mode**.
   Program status is available in this operating mode without restrictions, see Operating modes for SIMOTION devices (Page 4520).
4. Open the Properties window for the MCC unit, see Properties of an MCC source file (Page 3994).
5. Activate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit.
   This compiler option can also be activated as the global compiler setting (Page 3996) (same as section "Enabling single-step monitoring" in Tracking program execution using single step (Page 4534)).

6. Compile the MCC unit (**MCC unit > Accept and compile** menu command).
7. Select the **Target system > Load > Project to target system** menu command or click the button.
   The programs are downloaded to the target system.

Starting the Status program

Requirement

- The relevant MCC chart must be open.
- The MCC chart must not contain any changes that have not been saved.
- The MCC unit in SCOUT must be consistent with the target system.

Procedure

To start program status, proceed as follows:

1. Select the **MCC chart > Program status** menu command or click the button for **Program status** (keyboard shortcut Ctrl+F7).
   Program status is started.
2. If the MCC chart is assigned to more than one task, the **Call path/task selection Program status** dialog box opens:
   – In this dialog box, select the task in which you want to monitor the program.

   The MCC_1 symbol on the tab for the open MCC chart indicates that program status has begun, as does the symbol for the combination of currently activated test functions (monitoring, single step, trace, program status, breakpoints).
3. Open the commands for which you want to monitor the variables.
   You can open more than one command at a time, but you will only be able to monitor the variables in the active window (that is, the window in the foreground). You can size the window with the cursor so longer variable names can also be seen without any difficulty.

Note

Start program status for multiple windows for each SIMOTION device

A window can be an open ST source file, an open LAD/FBD program or an open MCC chart. An active window is located in the foreground and the current status values are displayed (MCC: open MCC command).

In **Process mode**, program status can only be started for one window.

In **Test mode**, program status can be started simultaneously for several windows per task. The maximum possible number depends on the utilization of the SIMOTION device.

If program status is started for an additional window (**Process mode**) or starting it causes the maximum possible number to be exceeded (**Test mode**), program status is automatically deactivated for the currently active window (**Process mode**) or the oldest active window (**Test mode**), i.e. momentarily paused, but not stopped. If a window with deactivated program status is brought to the foreground (MCC: open MCC command), program status is automatically reactivated and the current status values are displayed again.

Note

Program status and single-step monitoring

If you have activated program status and single-step monitoring, see Tracking single steps in the program (Page 4534), the monitored variables from a command are only updated once the next single step has been triggered, i.e. not until you have changed from this command to the next command to be processed.

Stopping program status

To stop program status, proceed as follows:

1. Select the **MCC chart > Program status** menu command or click the  button for **Program status** (keyboard shortcut Ctrl+F7).
   Program status is stopped.

Deactivating program status

Disabling program status frees up CPU resources.

To disable program status, proceed as follows:

1. First, stop the program status function, see Stopping program status.
2. Open the Properties window for the MCC unit, see Properties of an MCC source file (Page 3994).
3. Deactivate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (Page 3997) for this MCC unit and click **OK** to confirm.
   If this compiler option is activated as the global compiler setting, the global compiler setting (Page 3996) must be deactivated accordingly (same as section "Disabling single-step monitoring" in Tracking program execution using single step (Page 4534)).
4. Select the SIMOTION device, followed by **Operating mode** in the context menu.
5. Select **Process mode**.
   Program execution is optimized for this operating mode to ensure maximum performance, see Operating modes for SIMOTION devices (Page 4520).
6. Recompile the program and download it to the target system.

7.1.8.7 Program run

Program run: Display code location and call path

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path.

Follow these steps:

1. Click the **Show program run** button on the Program run toolbar.
   The "Program run call stack (Page 4546)" window opens.
2. Select the desired MotionTask.
3. Click the **Update** button.

The window shows:

- The position in the code being executed (e.g. line of the ST source file) stating the program source and the POU.
- Recursively positions in the code of other POUs that call the code position being executed.

The following names are displayed for the SIMOTION RT program sources:

Table 7-347 SIMOTION RT program sources

Name	Meaning
taskbind.hid	Execution system
stdfunc.pck	IEC library
device.pck	Device-specific library
tp-name.pck	Library of the *tp-name* technology package, e.g. cam.pck for the library of the CAM technology package

Program run parameters

You can display the following for all configured tasks:

- the current code position in the program code (e.g. line of an ST source file)
- the call path of this code position

Table 7-348 Program run parameter description

Array	Description
Selected CPU	The selected SIMOTION device is displayed.
Refresh	Clicking the button reads the current code positions from the SIMOTION device and shows them in the open window.
Calling task	Select the task for which you want to determine the code position being executed. All configured tasks of the execution system.
Current code position	The position being executed in the program code (e.g. line of an ST source file) is displayed (with the name of the program source, line number, name of the POU).
is called by	The code positions that call the code position being executed within the selected task are shown recursively (with the name of the program source, line number, name of the POU, and name of the function block instance, if applicable).

For names of the SIMOTION RT program sources, refer to the table in Program run (Page 4545).

Program run toolbar

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path with this toolbar.

Table 7-349 Program run toolbar

Symbol	Meaning
	Display program run Click this symbol to open the Program run call stack window. In this window, you can display the currently active code position with its call path. See: Program run: Display code position and call path (Page 4545)

7.1.8.8 Trace

Trace allows you to log and graphically display the characteristics of signals and axis states.

You must configure the data that you wish to record in the SIMOTION trace tool (for description, see the online help).

Activate trace

Figure 7-182 Trace trigger

This command can be used to log the characteristics of signals and axis states at a defined position in the program.

In order for trace recording to start when the command is run, the trace must already have been started in the trace tool by clicking the ▶ button. Additional details about working with the trace tool can be found in the online help.

System variable traceState[n].tracestate on the SIMOTION device can be used to query whether the trace is still in progress or has already finished. You can monitor this variable in the symbol browser if you have selected the SIMOTION device in the project navigator.

If another trace recording is required after the current one has finished, the trace will need to be started again by clicking the ▶ button.

Overview of parameters

You can set the following parameters:

Field/Button	Explanation/Instructions
Trace Trigger 1	Trace is started when the following condition is fulfilled: "Trigger with TraceTrigger 1 program call".
Trace Trigger 2	Trace is started when the following condition is fulfilled: "Trigger with TraceTrigger 2 program call".

Example of Activate trace

The simultaneous starting of two axes (axis_1 and axis_2) will be recorded.

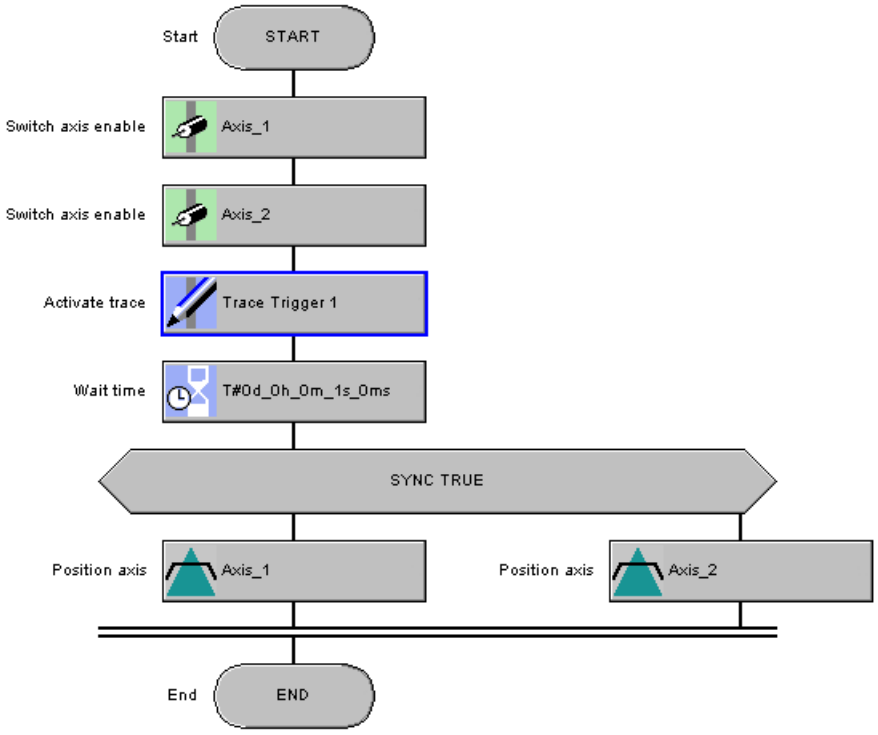


Figure 7-183 Trace recording of a synchronous start of two axes

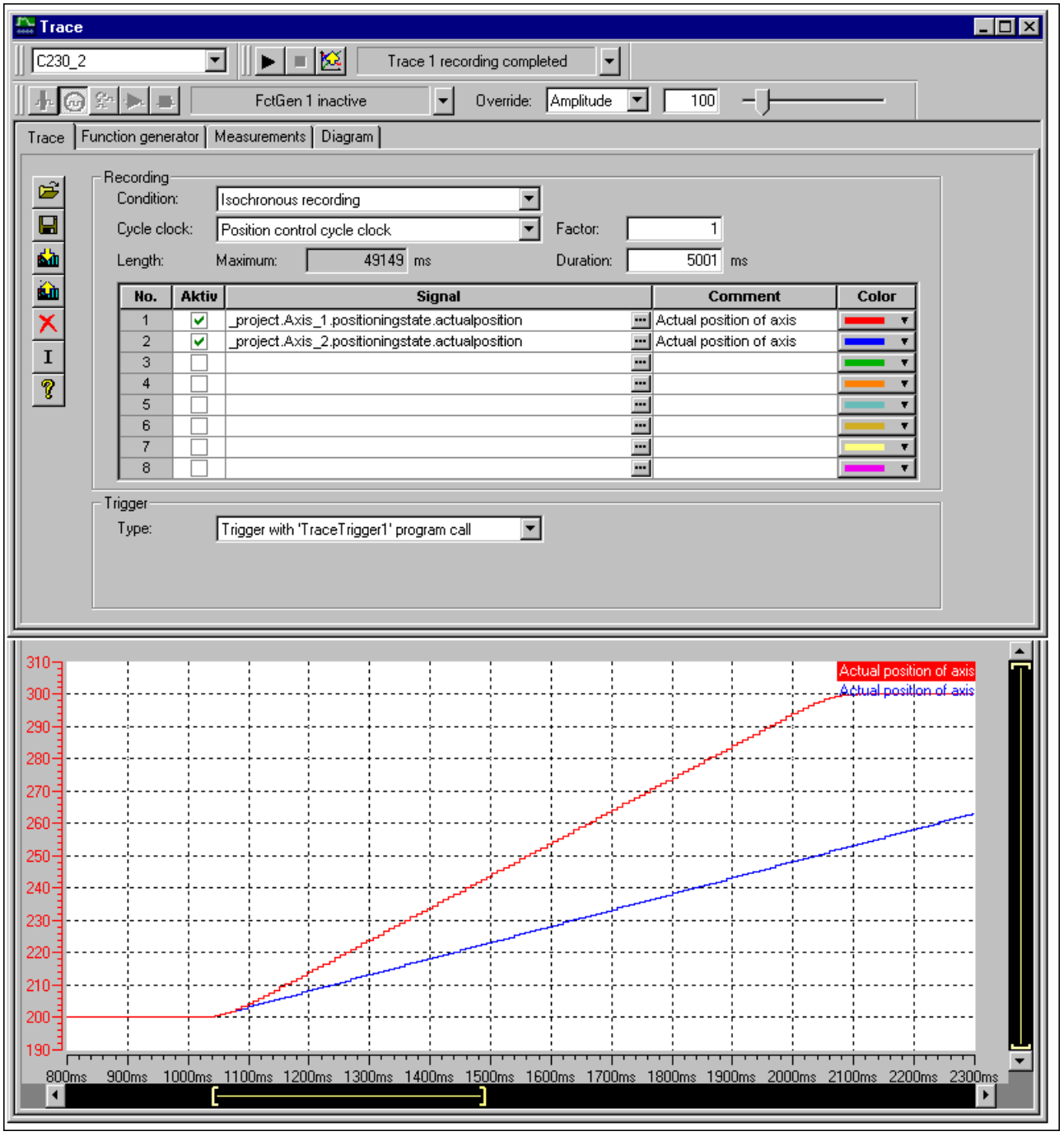


Figure 7-184 Trace recording of synchronous start

Relevant system variable for Activate trace

System variable of SIMOTION device: TraceControl

7.1.8.9 Breakpoints

General procedure for setting breakpoints

You can set breakpoints within a POU of a program source (e.g. ST source, MCC chart, LAD/FBD source). On reaching an activated breakpoint, the task in which the POU with the breakpoint is called is stopped. If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser.

Requirement:

- The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.

Procedure

Follow these steps:

1. Select "Debug mode" for the associated SIMOTION device; see Setting debug mode (Page 4550).
2. Specify the tasks to be stopped, see Specifying the debug task group (Page 4552).
3. Set breakpoints, see Setting breakpoints (Page 4553).
4. Define the call path, see Defining a call path for a single breakpoint (Page 4556).
5. Activate the breakpoints, see Activating breakpoints (Page 4560).

Setting the debug mode

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode only with activated life-sign monitoring (Page 4522) with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect!

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Requirement

1. A connection to the target system must have been established (online mode)
2. Debug mode must not be selected for any SIMOTION device.

Procedure

To set the debug mode, proceed as follows:

1. Highlight the SIMOTION device in the project navigator.
2. Select **Operating mode** from the context menu.
3. Select **Debug** mode (Page 4520).
4. Accept the safety information
5. Parameterize the sign-of-life monitoring.
   See also section: Important information about the life-sign monitoring (Page 4522).
6. Confirm with **OK**.

SIMOTION SCOUT switches to debug mode for this device; the SIMOTION device itself remains in "test mode", as long as at least one breakpoint is activated:

The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.

The breakpoints toolbar (Page 4555) is displayed.

As long as no breakpoints are activated, you can edit program sources in debug mode (Page 4524).

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device. The status bar indicates that debug mode is activated for the SIMOTION device.

Note

Pressing the spacebar or switching to a different Windows application causes the following to happen if the SIMOTION device is in debug mode (breakpoints activated):

- The SIMOTION device switches to the STOP operating state.
- The outputs are deactivated (ODIS).

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Define the debug task group

On reaching an activated breakpoint, all tasks that are assigned to the debug task group are stopped.

Requirement

1. A connection to the target system must have been established (online mode).
2. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4550).

Procedure

How to assign a task to the debug task group:

1. Highlight the relevant SIMOTION device in the project navigator.
2. Select **Debug task group** from the context menu.
   The Debug Task group window opens.
3. Select the tasks to be stopped on reaching the breakpoint:
   - If you only want to stop individual tasks (in RUN operating state): Activate the **Debug task group** selection option.
     Assign all tasks to be stopped on reaching a breakpoint to the **Tasks to be stopped** list.
   - If you only want to stop individual tasks (in HOLD operating state): Activate the **All tasks** selection option.
     In this case, also select whether the outputs and technology objects are to be released again after resumption of program execution.

Note

Note the different behavior when an activated breakpoint is reached, see the following table.

Table 7-350 Behavior at the breakpoint depending on the tasks to be stopped in the debug task group.

Property		Tasks to be stopped	
		Single selected tasks (debug task group)	All tasks
Behavior on reaching the breakpoint			
	Operating state	RUN	STOP
	Stopped tasks	Only tasks in the debug task group	All tasks
	Outputs	Active	Deactivated (ODIS activated)
	Technology	Closed-loop control active	No closed-loop control (ODIS activated)
	Runtime measurement of the tasks	Active for all tasks	Deactivated for all tasks
	Time monitoring of the tasks	Deactivated for tasks in the debug task group	Deactivated for all tasks
	Real-time clock	Continues to run	Continues to run

Property		Tasks to be stopped	
		Single selected tasks (debug task group)	All tasks
Behavior on resumption of program execution			
	Operating state	RUN	RUN
	Started tasks	All tasks in the debug task group	All tasks
	Outputs	Active	The behavior of the outputs and the technology objects depends on the **'Continue' activates the outputs (ODIS deactivated)** checkbox. • Active: ODIS will be deactivated. All outputs and technology objects are released. • Inactive: ODIS remains activated. All outputs and technology objects are only enabled for one STOP-RUN transition.
	Technology	Closed-loop control active	

Note

You can only make changes to the debug task group if no breakpoints are active.

The settings of the debug task group are retained after exiting "Debug mode".

Proceed as follows:

1. Set breakpoints (see Setting breakpoints (Page 4553)).
2. Define the call path (see Defining a call path for a single breakpoint (Page 4556)).
3. Activate the breakpoints (see Activating breakpoints (Page 4560)).

Setting breakpoints

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4550).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4552).

Procedure

How to set a breakpoint:

1. Select the code location where no breakpoint has been set:
   - SIMOTION ST: Place the cursor on a line in the ST source file that contains a statement.
   - SIMOTION MCC: Select an MCC command in the MCC chart (except module or comment block).
   - SIMOTION LAD/FBD: Set the cursor in a network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove a breakpoint, proceed as follows:

1. Select the code position with the breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Remove all breakpoints** menu command (shortcut CTRL+F5).
  - Click the  button in the Breakpoints toolbar.

Note

You cannot set breakpoints:

- For SIMOTION ST: In lines that contain only comment.
- For SIMOTION MCC: On the module or comment block commands.
- For SIMOTION LAD/FBD: Within a network.
- At code locations in which other debug points (e.g. trigger points) have been set.

You can list the debug points in all program sources of the SIMOTION device in the debug table:

- Click the  button for "debug table" in the Breakpoints toolbar.

In the debug table, you can also remove all breakpoints (in all program sources) of the SIMOTION device:

- Click the button for "Clear all breakpoints".

The breakpoints set also remain saved after leaving debug mode; they are displayed in debug mode only.

You can use the program status (Page 4542) diagnosis functions and breakpoints together in a program source or POU. However, the following restrictions apply depending on the program languages:

- SIMOTION ST: For version V3.2 of the SIMOTION Kernel, the (marked) ST source file lines to be tested with program status must not contain a breakpoint.
- SIMOTION MCC and LAD/FBD: The commands of the MCC chart (or networks of the LAD/FBD program) to be tested with program status must not contain a breakpoint.

Proceed as follows

1. Define the call path, see Defining a call path for a single breakpoint (Page 4556).
2. Activate the breakpoints, see Activating breakpoints (Page 4560).

Breakpoints toolbar

This toolbar contains important operator actions for setting and activating breakpoints:

Table 7-351 Breakpoints toolbar

Symbol	Meaning
	Set/remove breakpoint Click this icon to set at breakpoint for the selected code position or to remove an existing breakpoint. See: Setting breakpoints (Page 4553).
	Activate/deactivate breakpoint Click this icon to activate or deactivate the breakpoint at the selected code position. See: Activating breakpoints (Page 4560).
	Edit the call path Click this icon to define the call path for the breakpoints: • If a code position **with** breakpoint is selected: The call path for this breakpoint. • If a code position **without** breakpoint is selected: The call path for all breakpoints of the POU. See: Defining the call path for a single breakpoint (Page 4556), Defining the call path for all breakpoints (Page 4558).
	Activate all breakpoints of the active POU Click this symbol to activate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 4560).
	Deactivate all breakpoints of the active POU Click this symbol to deactivate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 4560).
	Remove all breakpoints of the active POU Click this symbol to remove all breakpoints from the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Setting breakpoints (Page 4553).

Symbol	Meaning
	Debug table Click this icon to display the debug table. See: Debug table parameters.
	Display call stack Click this icon after reaching an activated breakpoint to: • View the call path at the current breakpoint. • View the code positions at which the other tasks of the debug task group have been stopped together with their call path. See: Displaying the call stack (Page 4562).
	Resume Click this icon to continue the program execution after reaching an activated breakpoint. See: Resuming program execution (Page 4563), Displaying the call stack (Page 4562).
	Next step (SIMOTION Kernel as of version V4.4) Only available for the MCC and LAD/FBD programming languages: Click this icon to resume the program execution until the next MCC command or LAD/FBD network is reached. See: Resume program execution in single steps (Page 4563).
	Step through the subprogram (SIMOTION Kernel as of version V4.4) Only available for the MCC programming language. Click this icon to jump to the called subprogram and stop at the first command. The subprogram must be created in the MCC or LAD/FBD programming language. See: Resume program execution in single steps (Page 4563).

Defining the call path for a single breakpoint

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4550).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4552).
5. Breakpoint is set, see Setting breakpoints (Page 4553).

Procedure

To define the call path for a single breakpoint, proceed as follows:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the Call path / task selection breakpoint window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when the selected breakpoint is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when the activated breakpoint in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when the breakpoint in the selected task is reached. The task must be in the debug task group.
     The specification of a call path is possible.
4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at the activated breakpoint if the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at the activated breakpoint only when the POU is called from the selected code position.
     If the POU of the selected calling code position is also called from other code positions, further lines are displayed successively in which you proceed similarly.
5. If the breakpoint is only to be activated after the code position has been reached several times, select the number of times.

Note

You can also define the call path to the individual breakpoints in the debug table:

1. Click the button for "debug table" in the Breakpoints toolbar.
   The "Debug table" window opens.
2. Click the appropriate button in the "Call path" column.
3. Proceed in the same way as described above:
   – Specify the task.
   – Define the call path (only for functions and function blocks).
   – Specify the number of passes after which the breakpoint is to be activated.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 4560).

Note

You can use the "Display call stack (Page 4562)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for all breakpoints (Page 4558)

Defining the call path for all breakpoints

With this procedure, you can:

- Select a default setting for all future breakpoints in a POU (e.g. MCC chart, LAD/FBD program or POU in an ST source file).
- Accept and compare the call path for all previously set breakpoints in this POU.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4550).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4552).

Procedure

To define the call path for all future breakpoints of a POU, proceed as follows:

1. Select the code location where **no** breakpoint has been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the "Call path / task selection all breakpoints for each POU" window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when a breakpoint in this POU is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when an activated breakpoint of the POU in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when a breakpoint in the selected task is reached.
     The task must be in the debug task group.
     The specification of a call path is possible.
4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at an activated breakpoint when the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at an activated breakpoint only when the POU is called from the selected code position.
     If the selected calling code position is in turn called by other code positions, further lines are displayed successively in which you proceed similarly.
5. If a breakpoint is only to be activated after the code position has been reached several times, select the number of times.
6. If you want to accept and compare this call path for all previously set breakpoints in this POU:
   - Click **Accept**.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 4560).

Note

You can use the "Display call stack (Page 4562)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for a single breakpoint (Page 4556)

Activating breakpoints

Breakpoints must be activated if they are to have an effect on program execution.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4550).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4552).
5. Breakpoints are set, see Setting breakpoints (Page 4553).
6. Call paths are defined, see Defining a call path for a single breakpoint (Page 4556).

Activating breakpoints

How to activate a single breakpoint:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source file.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the  button in the Breakpoints toolbar.

To activate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Activate all breakpoints** menu command.
  - Click the  button in the Breakpoints toolbar.

Once the first breakpoint has been activated, the SIMOTION device switches to debug mode. It remains in this mode until the last breakpoint is deactivated.

In the Task status function bar, (Page 4564) the tasks with activated breakpoints are highlighted in gray ( ).

Note

Breakpoints of all program sources of the SIMOTION device can also be activated and deactivated in the debug table:

1. Click the  button for "debug table" in the Breakpoints toolbar.
   The "Debug Table" window opens.
2. Perform the action below, depending on which breakpoints you want to activate or deactivate:
   - Single breakpoints: Check or clear the corresponding checkboxes.
   - All breakpoints (in all program sources): Click the corresponding button.

The following applies up to version V4.3 of the SIMOTION Kernel:

- In the case of activated breakpoints, the "Single step" (Page 4534) test function of the SIMOTION MCC programming language cannot be used.

The following applies as of version V4.4 of the SIMOTION Kernel:

- The "Single step" test function of the SIMOTION MCC programming language is not available in the Debug mode.

Breakpoints cannot be activate if the control priority is at the axis control panel. Conversely, you cannot fetch the control priority for the axis control panel when a breakpoint activated.

Behavior at the activated breakpoint

On reaching an activated breakpoint (possibly using the selected call path (Page 4556)), all tasks assigned to the debug task group will be stopped. The behavior depends on the tasks in the debug task group and is described in "Defining a debug task group (Page 4552)". The breakpoint is highlighted.

In the Task status function bar, (Page 4564) the task in which the breakpoint was reached is highlighted in red ( ).

The following applies to the programming languages MCC or LAD/FBD: If the debug task group is stopped by a breakpoint, then the user has the option to change to another task, belonging to the debug task group, in the combo box. Always the breakpoint of the currently selected task is visualized.

If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser (Page 4526).

You can use the "Display call stack (Page 4562)" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Resuming program execution

You can resume the execution of the stopped tasks, see "Resuming program execution" (Page 4563).

As of version V4.4 of the SIMOTION Kernel, you can resume the task in single steps that has been stopped at the activated breakpoint in the MCC and LAD/FBD programming languages, see Resuming program execution in single steps (Page 4563).

Deactivate breakpoints

To deactivate a single breakpoint, proceed as follows:

1. Select the code position with the activated breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the  button in the Breakpoints toolbar.

To deactivate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Deactivate all breakpoints** menu command.
  - Click the  button in the Breakpoints toolbar.

Once the last breakpoint has been deactivated, the SIMOTION device switches to "test mode"; SIMOTION SCOUT continues to run in debug mode.

Display call stack

You can use the "Display call stack" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Requirement

The user program is stopped at an activated breakpoint, i.e. the tasks of the debug task group (Page 4552) have been stopped.

Procedure

To call the "Display call stack" function, proceed as follows:

- Click the  button for "display call stack" in the Breakpoints toolbar.
  The "Breakpoint call stack" dialog opens. The current call path (including the calling task and the number of the set passes) is displayed.
  The call path cannot be changed.

To use the "Display call stack" function, proceed as follows:

1. Keep the "Breakpoint call stack" dialog open.
2. To display the code position at which the other task was stopped, proceed as follows:
   – Select the appropriate task. All tasks of the debug task group can be selected.

   The code position, including the call path, is displayed. If the code position is contained in a user program, the program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) will be opened and the code position marked.
3. How to resume program execution:
   – Click the  button for "resume" (Ctrl+F8 shortcut) on the Breakpoint toolbar.

   When the next activated breakpoint is reached, the tasks of the debug task group will be stopped again. The current call path, including the calling task, is displayed.
4. Click **OK** to close the "Breakpoint call stack" dialog box.

For names of the SIMOTION RT program sources, refer to the table in "Program run (Page 4545)".

Resuming program execution

How to resume program execution:

- Perform the following (alternatives):
  – Select the **Debug > Continue** menu command (shortcut CTRL+F8).
  – Click the  button on the Breakpoint toolbar (Page 4555) to "Continue".

The stopped task is continued until the next active breakpoint is reached.

Resuming program execution in single steps (as of Kernel V4.4)

This function is available as of SIMOTION Kernel version V4.4.

In the MCC and LAD/FBD programming languages you can resume the task in single steps that was stopped at an activated breakpoint (Page 4550).

The current MCC command or the current LAD/FBD network and all stopped tasks of the debug task group are executed.

All tasks that are assigned to the debug task group are stopped at the following MCC command or LAD/FBD network or the next activated breakpoint within the debug task group.

Next step

To execute the current MCC command or the current LAD/FBD network at which the program has been stopped, proceed as follows:

- Perform the following (alternatives):
  – Select the **Debug > Next step** menu command (shortcut CTRL+F10).
  – Click the  button for "Next step" in the Breakpoints toolbar (Page 4555).

The current MCC command or the current LAD/FBD network is executed. The program is stopped at the following MCC command or the current LAD/FBD network.

A subprogram call is executed without interruption as long as no breakpoint is activated within the subprogram. If a

Stepping through the subprogram (MCC only)

If the program execution has been stopped at the "Subprogram call" (Page 4184) command in the MCC programming language, you can jump to the subprogram and run through it in single steps. The subprogram must have been created in the MCC or LAD/FBD programming language.

- Perform the following (alternatives):
  - Select the **Debug > Step through subprogram** menu command (shortcut CTRL+SHIFT +F10).
  - Click the  button on the Breakpoint toolbar (Page 4555) to "Step through subprogram".

The appropriate MCC chart or LAD/FBD program is opened and the program execution stopped at the first MCC command or LAD/FBD network.

Note

Stepping is not possible in MCC charts and LAD/FBD programs whose sources are in libraries.

Stepping is not possible in ST source files.

You can only open MCC charts and LAD/FBD programs with know-how protection if you have the required authorization (e.g. password).

7.1.8.10 Task status function bar

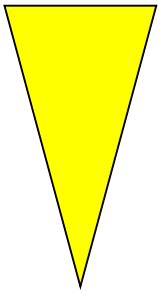
In a combo box, the Task status function bar displays all tasks of the active SIMOTION device, to which a program is assigned.

They are displayed under the following conditions:

1. SIMOTION SCOUT is in online mode.
2. The affected SIMOTION device is active, e.g.
   - In the project navigator, the SIMOTION device or an element in its subtree is selected (such as program source, technology object).
   - In the working area, an open window is active that belongs to an element in the subtree of the SIMOTION device.
3. The SIMOTION device is consistent.

A background color highlights the occurrence of specific events in the affected task, see the following table. The task in question is displayed in the combo box of the function bar according to the event priority.

Table 7-352 Meaning of background colors in the Task status function bar

Background color	Meaning	Priority
Cyan	The affected task waits for a command (Page 4534) at the "Single step" test function (only for SIMOTION MCC programming language).	Highest  Lowest
Red	The affected task is located at a breakpoint (Page 4560).	
Blue	In the affected task, the "Single step" test function (Page 4534) is activated (only for SIMOTION MCC programming language)	
Gray	In the affected task, at least 1 breakpoint (Page 4560) is activated.	
Yellow	In the affected task, the "Monitoring" test function (Page 4532) is activated (only for SIMOTION MCC programming language).	
White	In the affected task, none of the above-mentioned test functions are activated.	

Note

A selection of a task in the combo box is only possible:

- For the following test functions of the SIMOTION MCC programming language:
  - Monitoring (Page 4532)
  - Single step (Page 4534)
  - Trace (Page 4538)
- at activated breakpoints (Page 4560) in the MCC or LAD/FBD programming languages.

7.1.8.11 Project comparison

SIMOTION SCOUT has a **project comparison** function (start this via the **Start object comparison** button) for comparing objects within the same project and/or objects from different projects (online or offline).

Project comparison allows you to establish any differences and, if necessary, run a data transfer to rectify them.

Objects are devices and their sub-objects, programs, technology objects (TOs) or drive objects (DOs), and libraries. Comparing projects is useful if you need to carry out service work on the system.

Further information on project and detail comparisons can be found in the SIMOTION Project Comparison Function Manual.

7.1.9 Appendix

7.1.9.1 Basics of LAD/FBD/Formula for MCC

Ladder logic (LAD) for MCC

The LAD graphical programming language represents the program in the form of a circuit diagram. LAD enables you to track the flow of signals easily.

Note

You can switch between LAD and FBD provided that the programmed functions can be displayed in both languages.

The ladder diagram (LAD) for MCC has a limited set of operations.

The following operations are available in MCC:

- NO contact
- NC contact
- Comparator (CMP)
- Open/close branch

Every logic operation queries the signal status (0 or 1) of an electrical contact. The result is then stored or used to execute another operation.

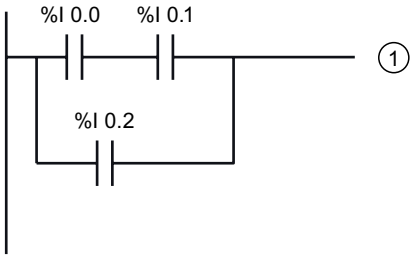
NO contact

This operation can be programmed to scan the signal status of a contact:

- Signal status=0: Contact is open.
- Signal status=1: Contact is closed.

The logic operation can be executed in a series connection or a parallel connection:

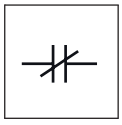
- In a **series connection**, the operation gates the result of its signal state scan according to the AND truth table.
- In a **parallel connection**, the operation gates the result of its signal state scan according to the OR truth table.



① The result of gating the three NO contacts is 1 if:
%I 0.0 is closed AND %I 0.1 is closed
OR
%I 0.2 is closed

Figure 7-185 Example of Normally open contact in a parallel circuit

NC contact

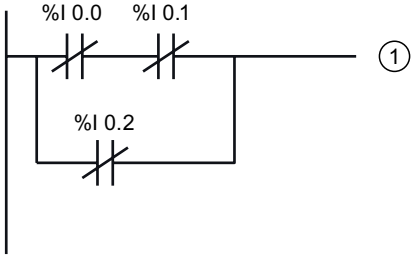


This operation can be programmed to scan the signal status of a contact:

- Signal status=0: Contact is closed.
- Signal status=1: Contact is open.

The logic operation can be executed in a series connection or a parallel connection:

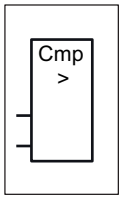
- In a **series connection**, the operation gates the result of its signal state scan according to the AND truth table.
- In a **parallel connection**, the operation gates the result of its signal state scan according to the OR truth table.



① The result of gating the three NC contacts is 1 if:
%I 0.0 is open AND %I 0.1 is open
OR
%I 0.2 is open

Figure 7-186 Example of Normally closed contact in a parallel circuit

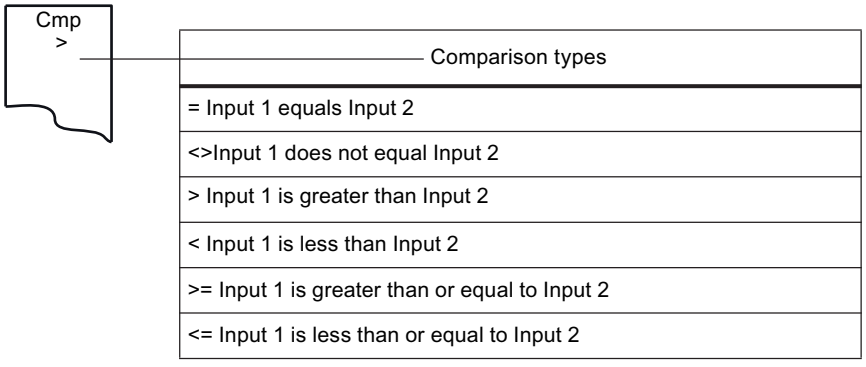
Comparator



This operation performs a comparison operation on integers or floating-point numbers.

Input 1 and Input 2 are compared according to the comparison type (see figure). If the result of the comparison is "true", then the result is "1" (otherwise "0").

There is no negation of the comparison result as this can be achieved by the opposite comparison operation in each case.

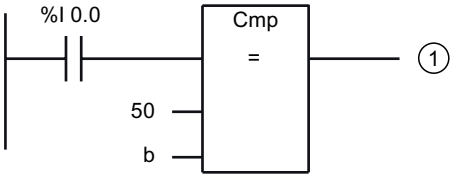


Comparison types
= Input 1 equals Input 2
<>Input 1 does not equal Input 2
> Input 1 is greater than Input 2
< Input 1 is less than Input 2
>= Input 1 is greater than or equal to Input 2
<= Input 1 is less than or equal to Input 2

Figure 7-187 Possible comparison types

You can use one of two different addressing methods:

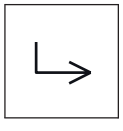
- Direct addressing with constant as operand
- Variable addressing with variable as operand



① The result is 1 if the following conditions are fulfilled:
%I 0.0 is closed AND the constant 50 is equal to variable b

Figure 7-188 Example of direct and variable addressing

Open/close branch



You can open and close parallel branches to fork the current flow. All operands and branches must be interlinked. Each branch must contain at least one operand.

Example of NO contact operations in a parallel circuit

The result is 0 or 1 depending on the position of the contacts (see figure).

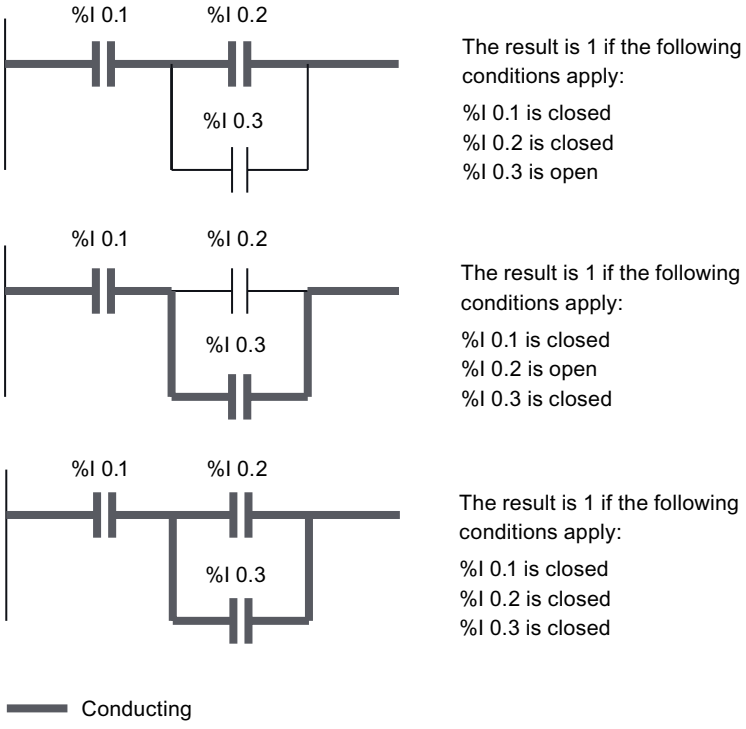


Figure 7-189 Example of NO contact operations in a parallel circuit

Function block diagram (FBD) for MCC

Function block diagram (FBD) is a graphical programming language that represents logic using the graphical logic symbols normally associated with Boolean algebra.

Note

You can switch between LAD and FBD provided that the programmed functions can be displayed in both languages.

The following operations are available in MCC:

- AND operation
- OR operation

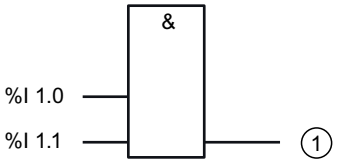
- Inverted input
- Comparator
- AND operation before OR operation
  OR operation before AND operation

Every logic operation queries the signal status (0 or 1) of an electrical contact. The result is then stored or used to execute another operation.

AND operation

The signal statuses of two or more operands are scanned:

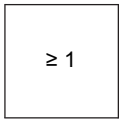
If the signal status of all operands is 1, the condition is fulfilled, otherwise the result of the operation is 0.



① The result is 1 if the following condition is fulfilled:
%I 1.0 = 1 AND %I 1.1 = 1

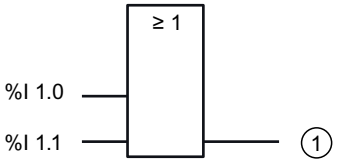
Figure 7-190 Example of an AND operation

OR operation



The signal statuses of two or more operands are scanned:

The condition is fulfilled if one operand has signal status "1", otherwise the result is "0".

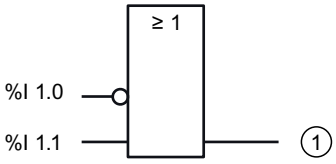


① The result is 1 if the following condition is fulfilled:
%I 1.0 = 1 OR %I 1.1 = 1

Figure 7-191 Example of an OR operation

Inverted input

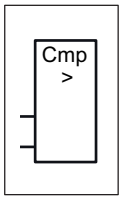
The signal status of the input is inverted by a dot at an operator input.



1 The result is 1 if the following condition is fulfilled:
%I 1.0 = 0 OR %I 1.1 = 1

Figure 7-192 Example of an OR operation with an inverted input

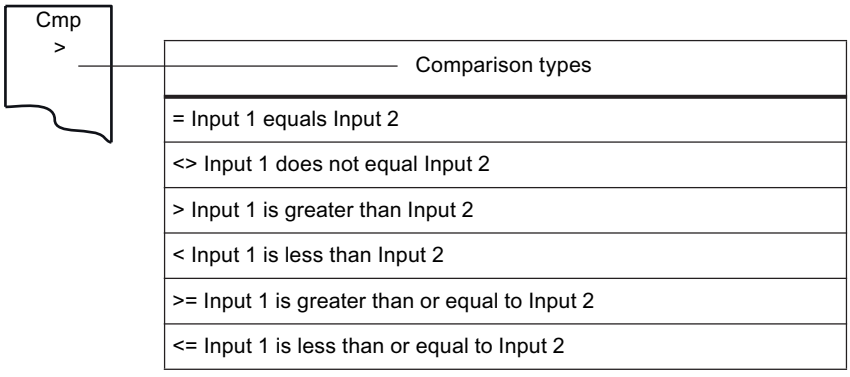
Comparator



This operation performs a comparison operation on integers or floating-point numbers.

Input 1 and Input 2 are compared according to the comparison type (see figure). If the result of the comparison is "true", then the result is "1" (otherwise "0").

There is no negation of the comparison result as this can be achieved by the opposite comparison operation in each case.

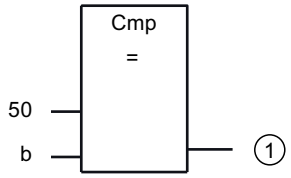


Comparison types
= Input 1 equals Input 2
<> Input 1 does not equal Input 2
> Input 1 is greater than Input 2
< Input 1 is less than Input 2
>= Input 1 is greater than or equal to Input 2
<= Input 1 is less than or equal to Input 2

Figure 7-193 Possible comparison types

You can use one of two different addressing methods:

- Direct addressing with constant as operand
- Variable addressing with variable as operand



① Constant 50 is the actual value at which Input 1 of the box should work. Constant 50 is the direct operand of the box.

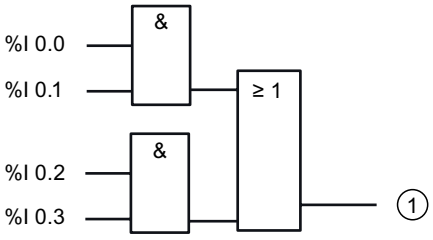
Input 2 is to work with the value of the variable b. Variable b is a direct operand.

Figure 7-194 Example of direct and variable addressing

AND operation before OR operationOR operation before AND operation

With the AND operation before OR operation, it is possible to scan the result of a signal status scan according to the OR truth table.

The result is "1" if at least one AND operation is fulfilled.

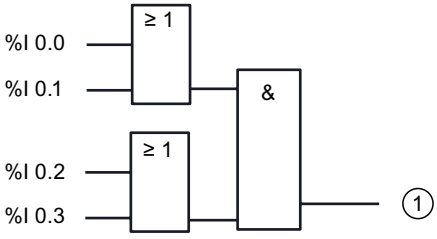


① The result of the operation is 1 if at least one AND operation is fulfilled.

Figure 7-195 Example of an AND operation before OR operation

With the OR before AND operation, it is possible to query the result of a signal state scan according to the AND truth table.

The result is "1" if all OR operations are fulfilled.



① The result of the operation is "1" if both OR operations are fulfilled.

Figure 7-196 Example of an OR operation before AND operation

Formula for MCC

Formula is a text-based, high-level language that essentially complies with IEC61131-3 in terms of language definition. It is suitable among other things for programming formula calculations and complex optimization algorithms.

System functions and operators can be moved from the command library to the programming window using a drag and drop operation.

Closed block comments (* ... *) are permitted, line comments (starting with //) not. Use the command comment on the MCC command (**Enter comment ...** context menu), see also "Display of commands in the MCC chart" (Page 4016).

Note

You can switch from Formula to LAD or FBD and vice versa provided that the programmed functions can be displayed in the other language.

Simple examples

The two examples illustrate an AND operation and an OR operation.

Table 7-353 Example of an AND operation (keyword AND)

Instruction	Description
%I0.0 AND a = 50	The result is 1 if input 0.0 is closed and variable a has a value of 50.

Table 7-354 Example of an AND operation (keyword AND)

Instruction	Description
%I0.0 OR %I0.1	The result is 1 if input 0.0 or input 0.1 is closed.

Querying a system variable

Formula is often used for scanning system variables; refer to the following example:

Table 7-355 Example of an AND operation (keyword AND)

Instruction	Description
Axis_1.positioningstate.homed = YES	The result = 1, if the axis has reached its homing position and is therefore homed.

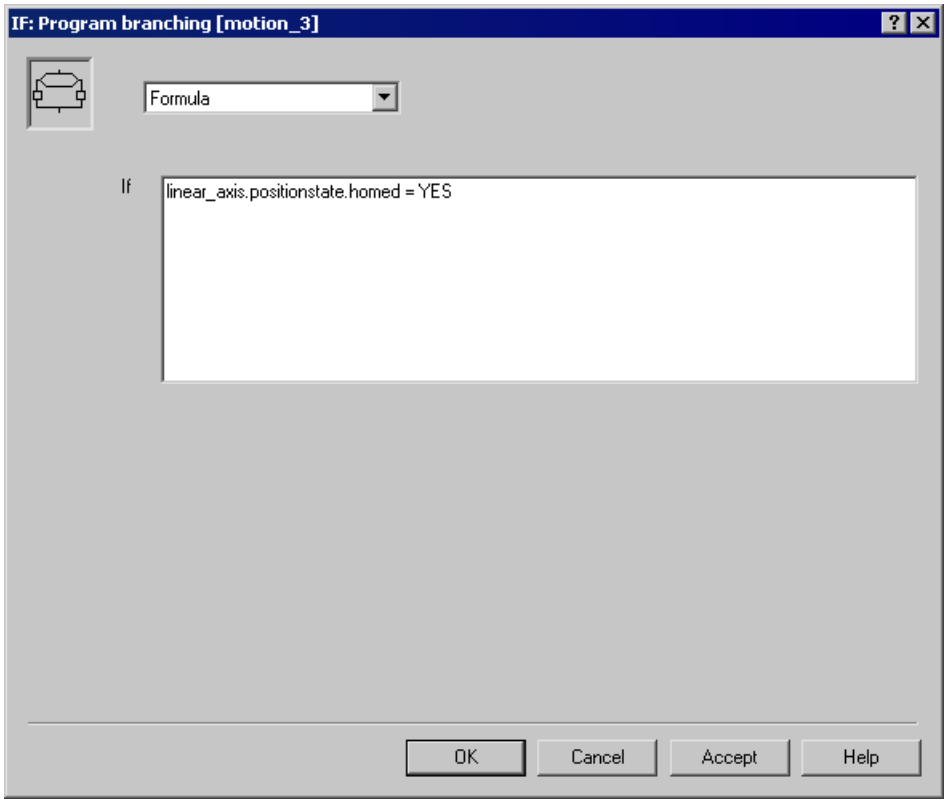


Figure 7-197 Scan whether an axis has been homed

7.1.9.2 Key combinations

The following key combinations are available:

Table 7-356 Shortcuts

Shortcut	Meaning
When MCC unit is open	
Ctrl+F4	Closes the active MCC unit.
Ctrl+B	Accepts and compiles the active MCC unit
Alt+Enter	Displays the properties of the active MCC unit for editing
Ctrl+R	Inserts an MCC chart in the MCC unit
Ctrl+P	Prints the MCC charts contained in the MCC unit
Ctrl+space	Automatic completion (Autocomplete) (Page 3979)
When an MCC chart is open	
Ctrl+F4	Closes the active MCC chart.
Ctrl+F7	Switches the program status (Page 4542) function on and off.
Shift+F7	Switches the observe (Page 4532) function on and off.
Ctrl+F9	Switches the single step (Page 4534) function on and off.
Alt+Enter	Displays the properties of the active MCC chart for editing

Shortcut		Meaning
	Ctrl+space	Automatic completion (Autocomplete) (Page 3979)
	Ctrl+Alt+O	Opens the user-defined FCs/FBs/programs ("program in program") and library FCs/FBs/programs ("program in program") directly following the subprogram call.
	Ctrl+Shift+O	Opens the overview window for the relevant MCC chart.
	↑	Selects the preceding command in the MCC chart
	↓	Selects the next command in the MCC chart.
	←	Selects the command to the left of the currently selected command (parallel branching).
	→	Selects the command to the right of the currently selected command (parallel branching).
	Ctrl+B	Accepts and compiles the MCC unit from the active MCC chart
	Ctrl+K	With a command selected, opens the brief comment box
	Ctrl+P	Prints the active MCC chart
	Ctrl+Z	Closes the brief comment box
	Pos1	Selects the left-hand branch (e.g. in a CASE statement)
	End	Selects the right-hand branch (e.g. in a CASE statement)
	Ctrl+Pos1	Selects the first command in the MCC chart
	Ctrl+End	Selects the last command in the MCC chart
	Pg Up	Moves the visible contents of the working area upwards by a distance of one window
	Pg Dn	Moves the visible contents of the working area downwards by a distance of one window
Edit menu		
	Ctrl+Z	Undoes the last action (except: Save).
	Ctrl+Y	Redoes the last action which was undone.
	Ctrl+X	Cuts a command
	Ctrl+C	Copies a command
	Ctrl+V	Inserts a command
	Del	Deletes selected commands in the MCC chart.
	Alt+Enter	Displays the properties of the active/selected object for editing.
	Enter	Opens the selected object.
	Ctrl+A	Highlights all the objects in the current window.
	Ctrl+B	Saves and compiles the active/selected object.
	Ctrl+F	Local search
	Ctrl+H	Local find and replace
	F3	Find next (for local search)
	Ctrl+Shift+F	Find in the project
	Ctrl+Shift+G	Find and replace in a project
	Ctrl+J	Next position (for search in the project)
Window menu		
	Ctrl+Shift+F5	Rearranges all windows opened in this application in horizontal tiled format.
	Ctrl+Shift+F3	Rearranges all windows opened in this application in vertical tiled format.
	Alt+F4	Closes all windows and ends the application.
View menu		

Shortcut	Meaning
Ctrl+F11	Maximizes the working area
Ctrl+F12	Maximizes the detail view
Ctrl+Num+	Enlarges the contents of the working area.
Ctrl+Num-	Reduces the contents of the working area.
Ctrl+Num /	Displays working area in its original size (100%).
F5	Updates the view.
Debug menu	
F12	Activate or deactivate a set breakpoint.
Ctrl+F8	Continue program execution at the activated breakpoint.
Ctrl+F10	Next step.
Ctrl+Shift+F10	Step through subprogram.
LAD and FBD: A number of MCC commands provide a limited command range for the LAD and FBD graphic programming languages (seeLAD/FBD/Formula (Page 4152)). The following shortcuts apply to both programming languages LAD and FBD.	
Cursor keys	With a selected operator: Navigation between the individual operators When an edit field is open: Navigates between individual operands
Del	Deletes an operator
Tab / Shift+Tab	Jumps forward to next button / input field / jumps back to previous button / input field
Pg Up	Reduces (zooms) display
Pg Dn	Enlarges (zooms) display
Pos1	Zooms to 100%
End	Zooms to total view
Return	Opens the edit field of the current operand or confirms the entry made in the edit field
Esc	Aborts the entry while edit field is open
Alt+I	Inserts a new variable (only active with variable assignment for the MCC command)
Alt+D	Deletes a variable (only active with variable assignment for the MCC command)
Within LAD The following shortcuts only apply to the LAD programming language.	
Alt+C	Inserts an NO contact
Alt+N	Inserts an NC contact
Alt+V	Inserts a comparator
Alt+P	Opens a branch
Alt+L	Closes a branch
Within FBD: The following shortcuts only apply to the FBD programming language.	
Alt+A	Inserts an AND
Alt+O	Inserts an OR
Alt+B	Adds an input

Shortcut		Meaning
	Alt+N	Negates an input
	Alt+V	Inserts a comparator

7.2 SIMOTION ST Structured Text

Preface

Scope

This document is part of the **SIMOTION Programming** documentation package.

This document applies to SIMOTION SCOUT, the engineering system of the SIMOTION product family in product version V5.4 in conjunction with:

- A SIMOTION device with the following versions of a SIMOTION Kernel:
  - V5.4
  - V5.3
  - V5.2
  - V5.1
  - V4.5
  - V4.4
  - V4.3
  - V4.2
  - V4.1 [1]
  - V4.0 [1]
  - V3.2 [1]

  [1] V4.5 is the last product version of SIMOTION SCOUT that will support these versions of the SIMOTION Kernel.
- The relevant version of the following SIMOTION Technology Packages, depending on the kernel:
  - Cam
  - Path (Kernel as of V4.1)
  - Cam_ext
  - TControl

This document describes the syntax and implementation of the SIMOTION ST Structured Text programming language for this version of SIMOTION SCOUT. It also includes information on the following topics:

- ST Editor and Compiler with program example
- Data storage and data management on SIMOTION devices
- Options for diagnosis and troubleshooting

The scope of the SIMOTION ST programming language may contain new syntax elements compared to earlier versions. These have only been tested using the current version of the SIMOTION kernel and are released only for this kernel version or higher versions.

Conversion of existing projects to the current SIMOTION SCOUT version

It is possible to upgrade existing projects to the current version of SIMOTION SCOUT and the SIMOTION ST programming language. In some cases, recompilation using the current version of the compiler can change the version identifiers in the data storage areas of the programs, thus resulting in deletion and initialization of all retentive and non-retentive data on the SIMOTION device. In exceptional cases, minor changes to the program source files may also be required.

If new syntax elements of the SIMOTION ST programming language are used on a SIMOTION device with an older version of the SIMOTION Kernel, the compiler outputs warning 16700. If these syntax elements are used anyway, the project can be stored in the old project format, but can no longer be converted using the compiler of an older version of SIMOTION SCOUT.

Information in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- **Grouped safety messages** (Chapter 1)
- **Introduction** (Chapter 2)
- **Getting Started with ST** (Chapter 3)
  Requirements for creating programs and a sample program
- **ST Basics** (Chapter 4)
  Elements of the ST programming language, variable and data type declarations, statements
- **Functions, Function Blocks and Programs** (Chapter 5)
  Programming and call of the program organization units (POU)
- **Object-Oriented Programming - OOP (as of Kernel V4.5)** (Chapter 6)
  Programming and calling classes and methods
- **Integration of ST into SIMOTION** (Chapter 7)
  Behavior of variables, access to inputs and outputs, libraries, preprocessor
- **Error Sources and Program Test** (Chapter 8)
  Information on error sources, efficient programming, and program testing

- **Appendices**
  - **Formal Language Description** (Appendix A.1)
  - **Compiler Error Messages and Remedies** (Appendix A.2)
  - **Template for Example Unit** (Appendix A.3)
- **Index**

If you want to get started immediately, begin by working through Chapter 2.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

7.2.1 Fundamental safety instructions

7.2.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

7.2.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

7.2.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

7.2.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

7.2.2 Introduction

In addition to conventional open and closed-loop control tasks, today's automation systems are increasingly required to handle data management functions and complex mathematical calculations. ST (Structured Text) is specially designed for these tasks. Standardized to IEC 61131-3 (German standard DIN EN-61131-3), this programming language makes your job as a programmer easier.

7.2.2.1 High-level programming language

ST is a high-level, PASCAL-based programming language. This language is based on the IEC 61131-3 standard, which standardizes programming languages for programmable controllers (PLC). ST is based on the *Structured Text* part of this standard.

Using a high-level language like ST to program control systems offers the user a wide range of possibilities, for example:

- Data management
- Process optimization
- Mathematical/statistical calculations

7.2.2.2 Programming language with technology commands

In addition to IEC 61131-3 compliance, the SIMOTION ST programming language also contains commands for SIMOTION devices, motion control and technology.

Technology objects represent a technological functionality, e.g. positioning an axis or assigning parameters for an output cam. Technology commands are language commands provided by the technology objects. Such commands may be used, for example, to activate camming or to control motion sequences, for example, in order to position an axis.

7.2.2.3 Execution levels

The SIMOTION execution system provides different execution levels (cyclic, synchronous, time-controlled, alarm-controlled and sequential) for optimal support of the various tasks involved in creating user programs.

SIMOTION SCOUT is the engineering system of the SIMOTION product family. ST is the high-level language for creating user programs; in ST, you can develop user programs for the various execution levels.

The execution of user programs can be time-driven if you want them to run synchronously with the system clock or a defined time cycle. They can be interrupt-driven if they are to start and run once in response to a particular event. Alternatively, they can run sequentially or cyclically at the round robin execution level.

7.2.2.4 ST editor with tools for writing and testing programs

An easy-to-use text editor is provided for creating programs.

The ST compiler converts the edited program into executable code and indicates any syntax errors, specifying the program line and the cause of the error.

SIMOTION SCOUT provides test functions for testing ST programs. You can test and visualize your programs online.

7.2.3 Getting Started with ST

This chapter uses a simple example to describe how to write a program, compile it into executable code, run it, and test it.

7.2.3.1 Integration of ST in SCOUT

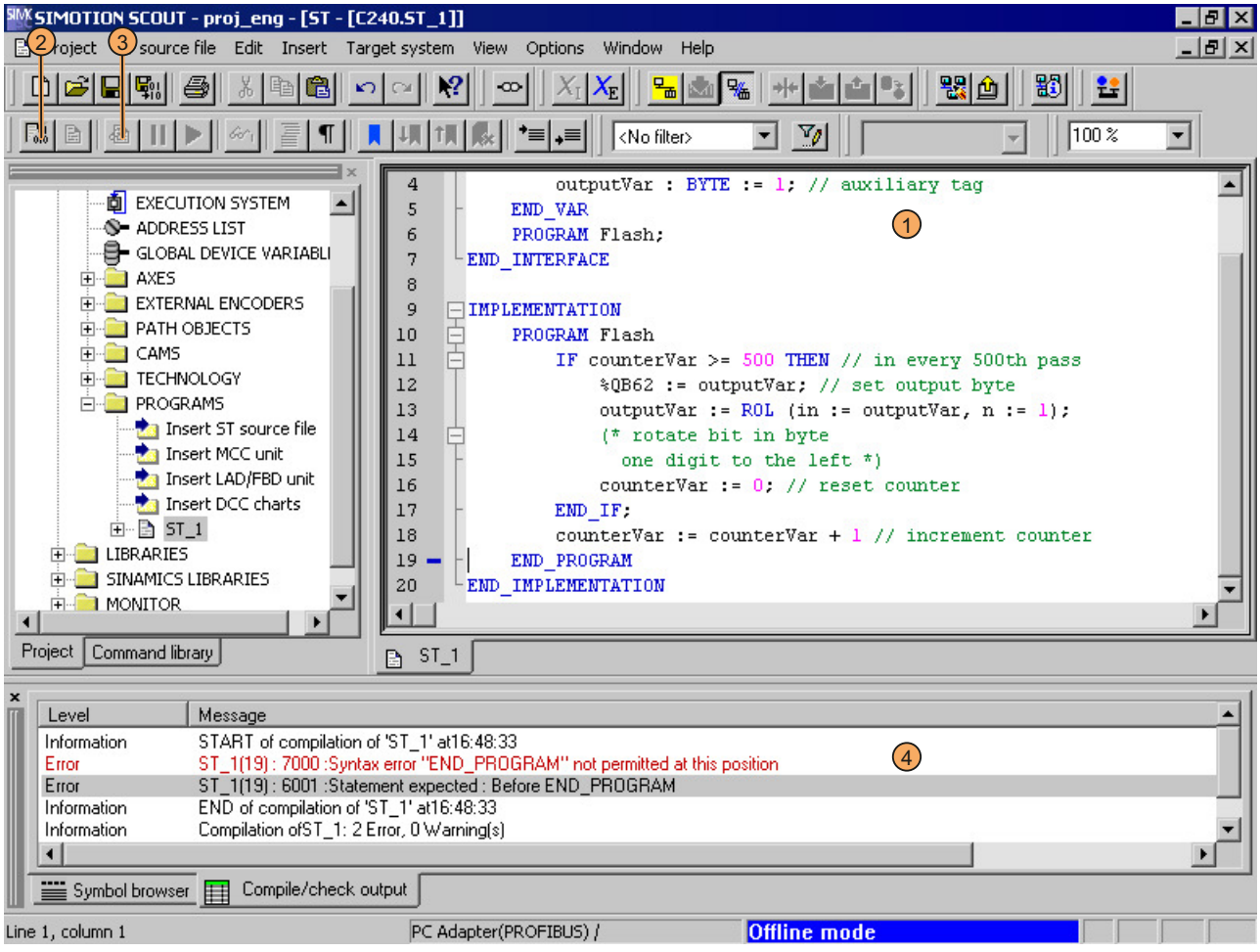
The program environment for ST comprises the following components:

- An **editor** for creating programs, consisting of functions (FC), function blocks (FB), and user-defined data types (UDT), etc.
- A **compiler** for compiling the previously edited ST program into executable machine code

- several diagnostics functions (e.g. **program status**) for assisting your search for logical program errors in the running program;
- a **detail view**, in which, for example, error messages of the compiler are displayed. An important tab of the detail view is the **Symbol browser**, where you can monitor and change variables.

The individual components are easy to use. They are integrated directly in the SIMOTION SCOUT workbench.

For more information about the operation of the workbench and its tools, refer to the SIMOTION SCOUT Configuration Manual.

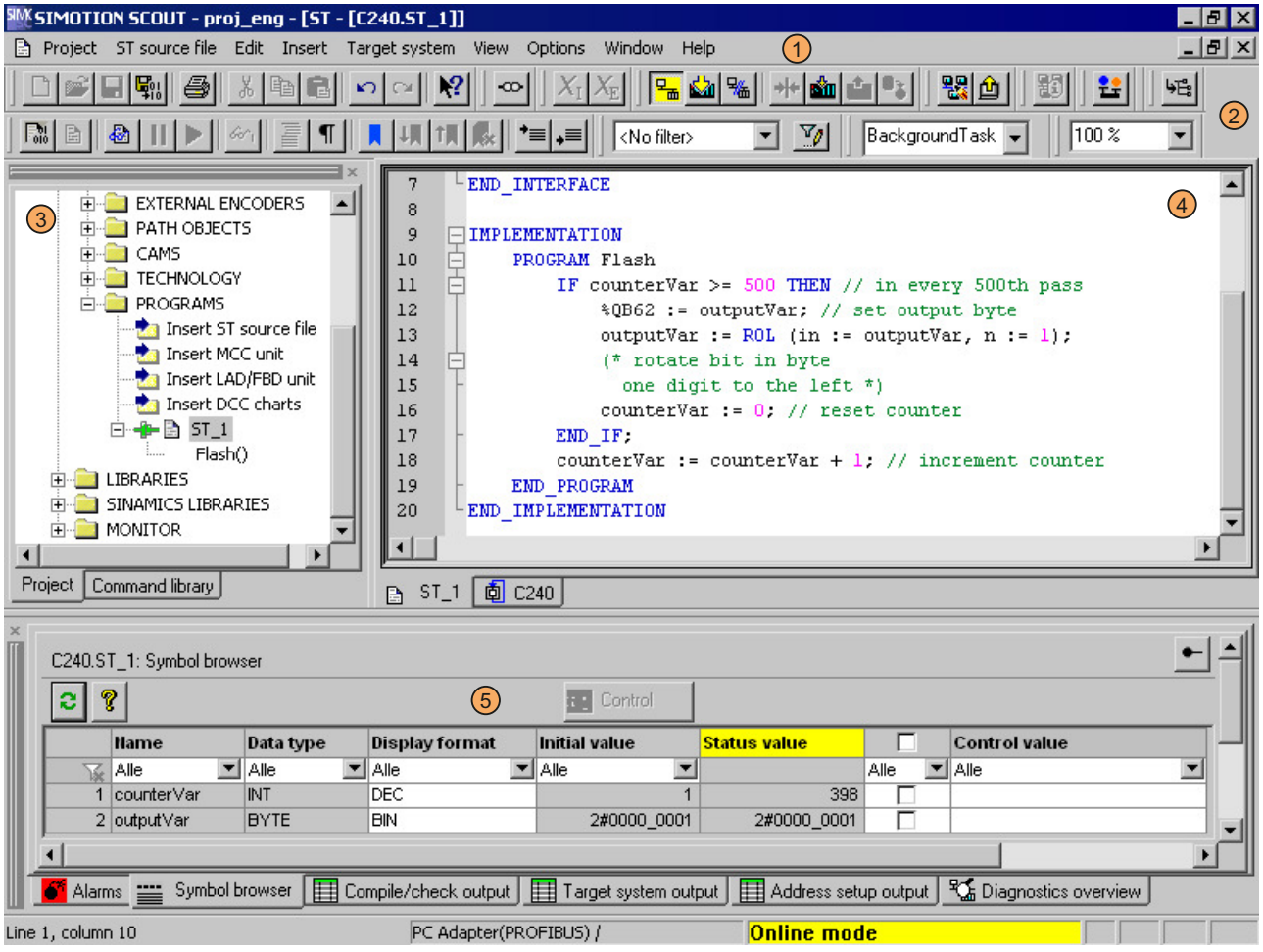


① Editor
② Compiler
③ Program status
④ Detail view (Compile/check output tab)

Figure 7-198 Development environment of ST

Getting to know the elements of the workbench

The workbench represents the framework for SIMOTION SCOUT. Its tools allow you to perform all the steps necessary to configure, optimize and program a machine for your application.



1) Menu bar
2) Toolbars
3) Project navigator
4) Working area
5) Detail view (Symbol browser tab)

Figure 7-199 Workbench elements

The workbench contains the following elements:

- Menus
  Menus contain menu commands with which you can control the workbench and call tools, etc.
- Toolbars
  You can execute many of the available menu commands by clicking the corresponding button in one of the toolbars.

- Project navigator
  The project navigator displays the entire project and its elements (e.g. CPU, axes, programs, cams) in a tree structure.
- Working area
  This window allows you to perform specific tasks either independently (by programming) or using wizards (by configuring).
- Detail view
  The detail view displays additional information about the elements selected in the project navigator, e.g. all global variables for a program or the **Compile/Test Output** window.

7.2.3.2 Requirements for program creation

This section describes the general conditions you will need to meet before writing a program. You will find detailed information in the SIMOTION SCOUT Configuring Manual and the SIMOTION Motion Control function descriptions.

Add or open a project

The project is the highest level in the data management hierarchy. SIMOTION SCOUT saves all data which belongs, for example, to a production machine, in the project directory.

This means that the project therefore brackets together all SIMOTION devices, drives, etc. belonging to one machine.

Once you have created a project, you can:

- Configure hardware
- Insert and configure technology objects

Configuring hardware

Within the project, the hardware used must be made known to the system, including:

- SIMOTION device
- Centralized I/O (with I/O addresses)
- Distributed I/O (with I/O addresses)

A SIMOTION device must be configured before you can insert and edit ST source files.

Insert and configure technology objects

The functionality of axes, output cams, etc. is represented in SIMOTION by technology objects (TOs).

You cannot program technology objects using system functions and access their system variables until you have inserted and configured them.

7.2.3.3 Working with the ST editor and the compiler

In this section, you will learn how to use the ST editor and the compiler.

Insert ST source file

ST source files are assigned to the SIMOTION device on which the programs contained in the source file will subsequently be run (e.g. SIMOTION C240).

They are stored in the project navigator under the SIMOTION device in the PROGRAMS folder.

Note

MCC units, LAD/FBD units and DCC charts are also stored in the **PROGRAMS** folder under the SIMOTION device.

For a description of the SIMOTION MCC (Motion Control Chart) programming language, refer to the SIMOTION MCC Programming and Operating Manual.

For a description of the SIMOTION LAD (Ladder Diagram) and SIMOTION FBD (Function Block Diagram) programming languages, refer to the SIMOTION LAD/FBD Programming and Operating Manual.

Procedure

1. Open the appropriate SIMOTION device in the project navigator.
2. Select the **PROGRAMS** folder.
3. Select the menu **Insert > Program > ST source file**.
4. Enter the name of the ST source file.
   The names of program sources must comply with the rules for identifiers (Page 4656): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters.
   The permissible length of the name depends on the SIMOTION Kernel version:
   - As of version V4.1 of the SIMOTION Kernel: maximum 128 characters.
   - Up to version V4.0 of the SIMOTION Kernel: maximum 8 characters.

   Names must be unique within the SIMOTION device.
   Protected or reserved identifiers (Page 4657) are not permitted.
   Existing program sources (e.g. ST source files, MCC units) are displayed.
5. You can also enter an author, version, and a comment.
6. Activate the **Open editor automatically** checkbox.
7. If necessary, select further tabs to make local settings (only valid for this ST source file):
   - **Compiler** tab: Local settings of the compiler (Page 4626) for code generation and message display.
   - **Additional settings** tab: Definitions for preprocessor (Page 4635)
8. Confirm with **OK**.

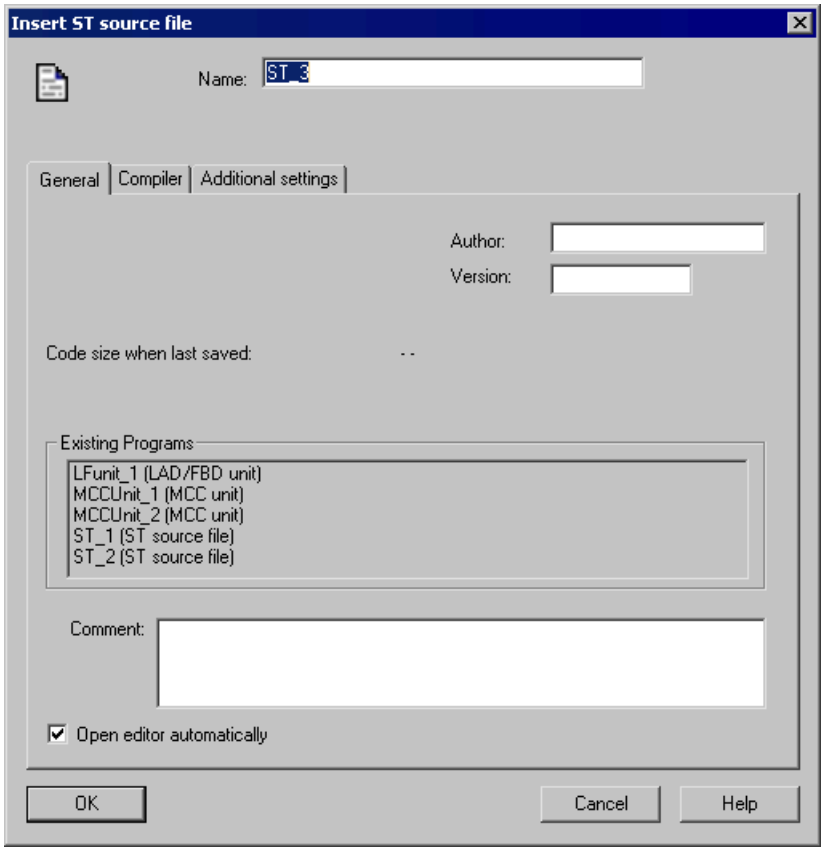


Figure 7-200 Insert ST source file

Note

When you click **OK**, the ST source file will only be transferred to the project. The data, together with the project, is only saved to the data carrier if you select, for example, **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

Opening an existing ST source file

Procedure

How to open an ST source file:

1. Open the subtree of the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Select the desired ST source file.

4. Select the **Edit > Open object** menu command.
5. Only for ST source files with know-how protection:
   If the ST source file is not already open and the log-in details assigned to it have not yet been used to log in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this source file is canceled temporarily (until the file is closed).
   - If required, activate the "Use login as a standard login" checkbox.
     You will be logged in with this login and can now open additional units to which the same login is assigned without having to re-enter the password.

This ST source file is opened with the saved folding information (Page 4596) and bookmarks (Page 4607). Multiple units can be opened.

Note

You can also double-click the required ST source file to open it.

Changing the properties of an ST source file

Procedure

1. Under the SIMOTION device, open the **PROGRAMS** folder.
2. Select the desired ST source file.
3. Select the **Edit > Object Properties** menu command.
4. If necessary, select further tabs to make local settings (only valid for this ST source file):
   - **General** tab: General details for the ST source file, e.g. code size for the last compilation, time stamp of the last change, storage location of the project (see figure).
   - **Compiler** tab: Local settings of the compiler (Page 4626) for code generation and message display.
   - **Additional settings** tab: Definitions for the preprocessor (Page 4635) and display of compiler options (Page 4633) as specified in the current settings of the compiler. You also have the possibility of setting the ST source as read-only (Read-Only-mode).
   - **Compilation** tab: Display of the compiler options (Page 4633) for the last compilation of the ST source.
   - **Object address** tab: Set the internal object address of the ST source. The object addresses of the other program sources are displayed.

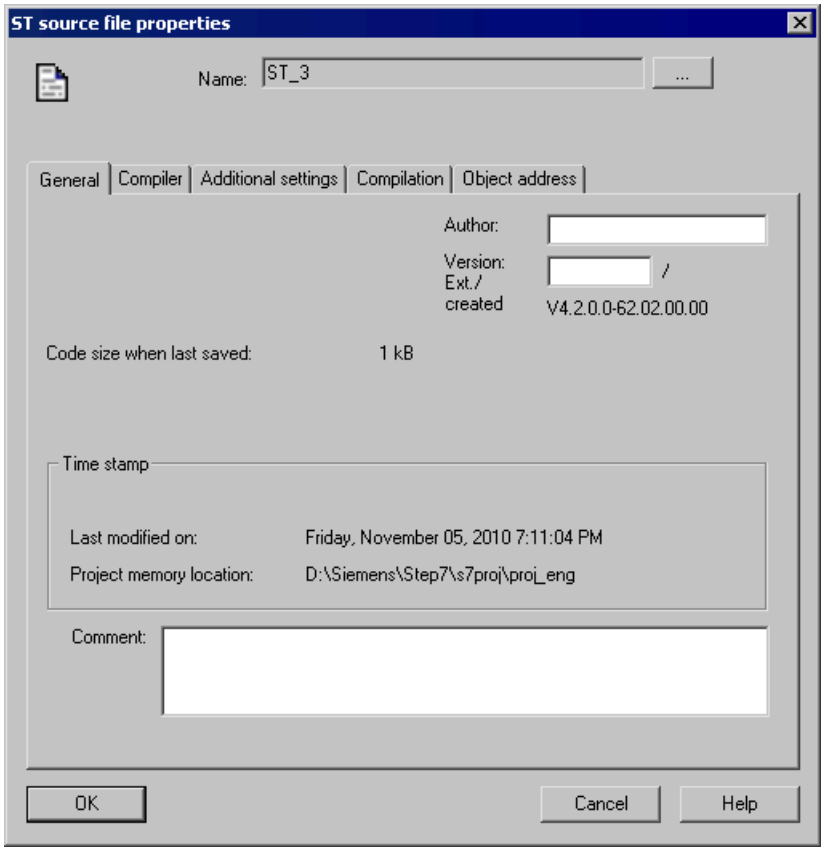


Figure 7-201 Properties of an ST source file

Changing the name of an ST source file

You can also change the names of the ST source file here. To do this, click the [...] button.

Names for program source files must satisfy the rules for identifiers: They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper and lower case letters.

The permissible length of the name depends on the SIMOTION Kernel version:

- As of Version V4.1 of the SIMOTION Kernel: maximum 128 characters.
- Up to Version V4.0 of the SIMOTION Kernel: maximum 8 characters.

Names must be unique within the SIMOTION device.

Protected or reserved identifiers (Page 4657) are not allowed.

Existing program sources (e.g. ST source files, MCC units) are displayed.

Note

With versions of the SIMOTION Kernel up to V4.0, a violation of the permissible length of the program source file name may not be detected until a consistency check or a download of the program source file is performed!

Read-only ST sources

You can set ST sources as read-only to prevent unintentional code changes during commissioning. To do this, navigate to "Additional settings" and set the option "Read-Only-Mode". Functions that would change the ST source are disabled in read-only mode. For this reason, the corresponding context menu entries and icons are inactive.

The following functions can still be used when an ST source is read-only:

- Status program
- Breakpoints
- Displaying the variable in the tooltip
- Pasting/removing bookmarks, jumping to bookmarks
- Selecting variables
- Searching (Ctrl +F)
- Compiling

Working with the ST editor

The ST editor makes it easier for you to work with the ST source file, variables and technology objects through the following operator controls:

- Syntax coloring
- Drag and drop
- Menu commands and shortcuts

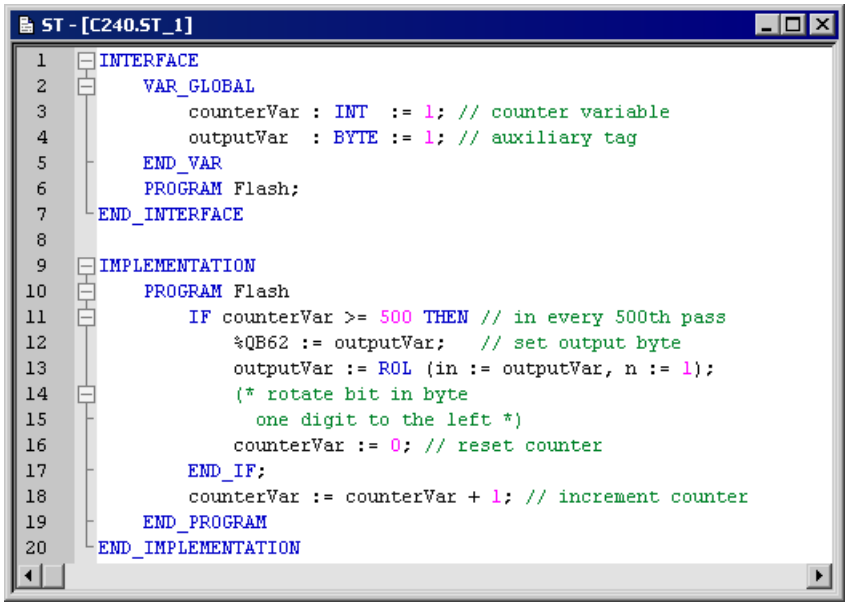


Figure 7-202 Opened ST source file in the ST editor

Syntax coloring

The ST editor represents language elements in different colors:

- Blue: Keywords and compiler built-in functions
- Magenta: Numbers, values
- Green: Comments
- Black: Technology objects, user code, variables

Drag&drop

Drag&drop

A drag-and-drop operation (dragging while keeping the left mouse button pressed) enables you to:

- Move selected text areas within an ST source file or to another opened ST source file.
- Copy names of variables from the symbol browser to the ST source file.
- Copy names (e.g. of technology objects, functions or function blocks) from the project navigator to the ST source file.
- Copy system functions from the command library to the ST source file.

To copy names of variables from the symbol browser to the ST source file:

1. Select the entire line of the desired variable in the symbol browser. To do this, click the line number at the start of the line.
2. Press the left mouse button and drag the line number to the desired position in the ST source file.
   The name of the selected variable is inserted in the ST source file.

To copy the name of an element (e.g. a technology object, a function or a function block) from the project navigator to the ST source file:

1. Select the **Project** tab in the project navigator.
2. Select the element in the project navigator.
3. Press the left mouse button and drag the element to the desired position in the ST source file.
   The name of the selected element is inserted in the ST source file.

To copy a system function from the command library to the ST source file:

1. Select the **Command Library** tab in the project navigator.
2. Select the system function in the command library.
3. Press the left mouse button and drag the system function to the desired position in the ST source file.
   The system function is inserted in the ST source file with its parameters.

Settings of the ST editor

Procedure:

1. Select the menu **Options > Settings**.
2. Select the **ST editor / scripting** tab.
3. Enter the settings.
4. Click **OK** or **Accept** to confirm.

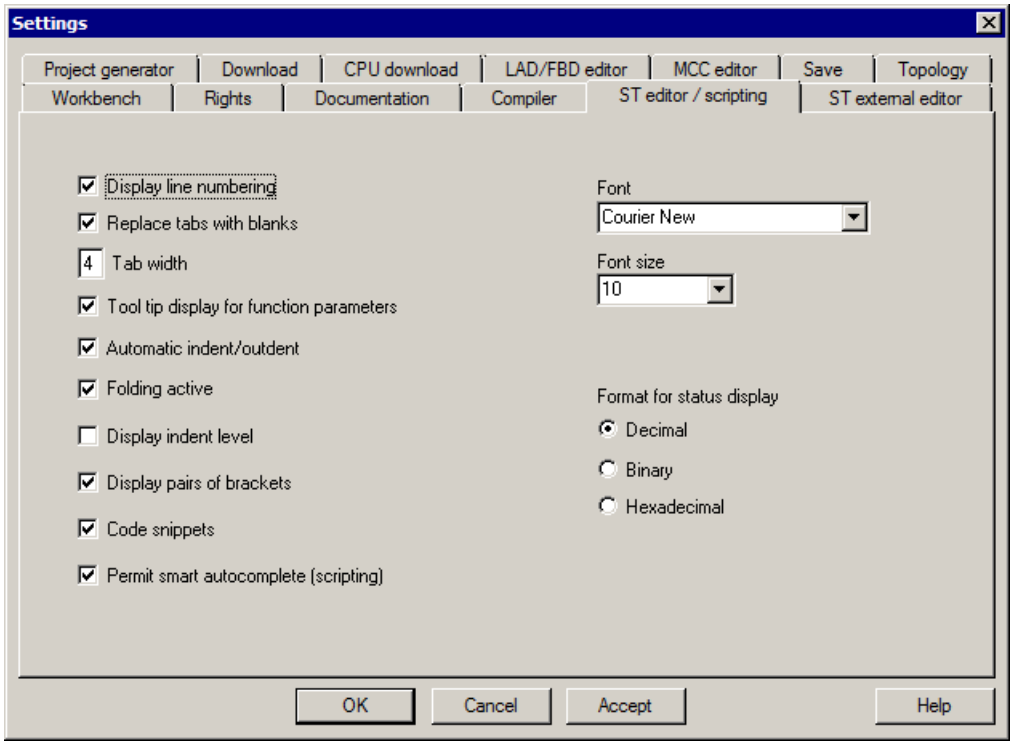


Figure 7-203 ST editor / scripting settings

The settings also apply to the script editor.

The table below contains a description of the individual parameters.

Table 7-357 ST editor / scripting parameter settings

Parameter	Description
Display line numbering	If active, the line numbers are displayed. See: Other ST editor tools (Page 4613).
Replace tabs with blanks	You select here how text indentation is performed (for the automatic indentation or by pressing the Tab key): • If active: By adding the appropriate number of space characters ($20). • If inactive: By adding the tab character ($09). See: Indentations and tabs (Page 4594).

Parameter	Description
Tab width	Number of characters skipped by a tab. See: Indentations and tabs (Page 4594).
Tooltip display for function parameters	When active, the parameters are displayed as tooltips for the functions.
Automatic indent/outdent	If active, for the text input, source file sections and blocks are indented automatically by the set tab width. See: Indentations and tabs (Page 4594).
Folding active	If active, the column with the folding information is displayed at the left-hand side next to the edit area. You can then hide blocks in an ST source file so that only the first line of the block remains visible. See: Folding (showing and hiding blocks) (Page 4596).
Display indentation level	If active, you can optically highlight the indent and outdent for blocks using vertical help lines (in accordance with the set tab size). See: Indentations and tabs (Page 4594).
Display bracket pairs	If active, the associated bracket of the pair that belongs to the bracket where the cursor is located will be found and optically highlighted. See: Other ST editor tools (Page 4613).
Code snippets	If active, after input of certain keywords (such as IF, WHILE, VAR), a window opens to display a code snippet for the associated keyword (e.g. IF (*condition*) ... END_IF). This code snippet can be transferred to the ST source file. See: Code snippets (Page 4611).
Permit Smart-autocomplete (scripting) [1]	If active, automatic completion of terms is permitted for the scripting editor.
Font	Font for the display of the text in the ST editor. All non-proportionally spaced fonts installed on the PC are available for selection.
Font size	Font size (in pt) for the display of the text in the ST editor. See: Change the font size in the ST editor (Page 4602).
Format for status display [2]	Format in which the variable values with bit data type are displayed for the program status or the variable status (for ST editor only). See: Properties of the program status (Page 4949), variable status (Page 4946)

1 Only for the scripting editor

2 Only for ST editor

Indentations and tabs

Specify tab width

The standard tab width for all ST sources is specified in the settings of the ST editor (Page 4593).

This setting is used for all ST source files opened subsequently.

Indent using tabs or spaces

You can select in the settings of the ST editor (Page 4593) how the text will be indented (e.g. with the automatic indent and outdent when the Tab key is pressed):

- By adding the appropriate number of space characters ($20).
- By adding the tab character ($09).

This setting is used for all ST source files opened subsequently.

Automatically indent and outdent blocks

The ST editor recognizes blocks introduced with a keyword and terminated with another keyword, e.g.:

- INTERFACE / END_INTERFACE
- IMPLEMENTATION / END_IMPLEMENTATION
- Declaration blocks (e.g. TYPE / END_TYPE, VAR / END_VAR)
- Program organization units (e.g. PROGRAM / END_PROGRAM)
- Control statements (e.g. IF / END_IF, FOR / END_FOR)

During the text input, the ST editor can automatically indent text within blocks by the tab size. The end line of the block will be outdented automatically.

This function is activated in the settings of the ST editor (Page 4593).

Note

This setting affects only the behavior during the text input. It does not have any effect on existing text in the ST sources.

Format current selection

You can use this function to force the blocks (see above) in an existing text to be indented by the tab size in accordance with their hierarchy. The number of the leading spaces or tabs will be changed:

- As specified by the current tab size of the ST source file.
- As specified by the current setting for the type of the indent (with tabs or spaces).

Follow these steps:

1. Select the text area in the ST editor that you want to format (see Select text (Page 4603)).
2. Select the **View > Format current selection** context menu.

Note

Leading tabs or spaces will be replaced in a line only when the formatting changes their number.

Indent or outdent selected area

You can use this function to indent or outdent selected sections of text by one tab width.

Follow these steps:

1. Select the text area that you want to indent or outdent in the ST editor (see Selecting text (Page 4603)).
   The selected text area must cover multiple lines.
2. Select the **Indent selected area** or **Undo selected area** context menu.

Indent help (Display indent level)

You can optically highlight the indent and outdent for blocks using vertical help lines (in accordance with the set tab size).

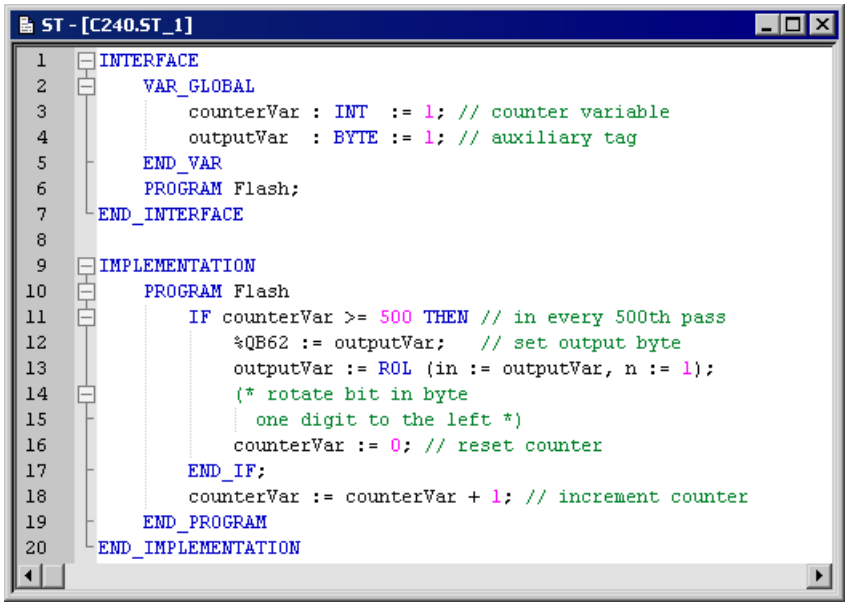


Figure 7-204 ST source with visible indent aid

You can activate or deactivate this function:

- For the active ST source
  - Select the **View > Indent help** context menu.

  This setting is not saved when the ST source file is closed.
- For all open ST sources:
  - Activate or deactivate the **Display indentation level** checkbox in the ST editor settings (Page 4593).

  This setting is also used for all ST source files opened subsequently.

Folds (show and hide blocks)

You can hide blocks in an ST source file so that only the first line of the block remains visible. This increases legibility when editing or reading a longer ST source file.

A block is introduced with a keyword and terminated with another keyword, e.g.:

- INTERFACE / END_INTERFACE
- IMPLEMENTATION / END_IMPLEMENTATION
- Declaration blocks (e.g. TYPE / END_TYPE, VAR / END_VAR)
- Program organization units (e.g. PROGRAM / END_PROGRAM)
- Control statements (e.g. IF / END_IF, FOR / END_FOR)
- Block comment (* / *)

Folding must be activated (see below) in order to show and hide blocks. Once this function has been activated, the column containing the folding information (to the left of the edit area) is displayed.

How to recognize that a block is displayed with folding activated:

- The ⊟ (minus) character appears next to the first line of the block.

How to recognize that a block is hidden:

- The ⊞ (plus) character appears next to the first visible line of the block.
- A hyphen is displayed below this line. This hyphen is visible even if the column containing the folding information is hidden.

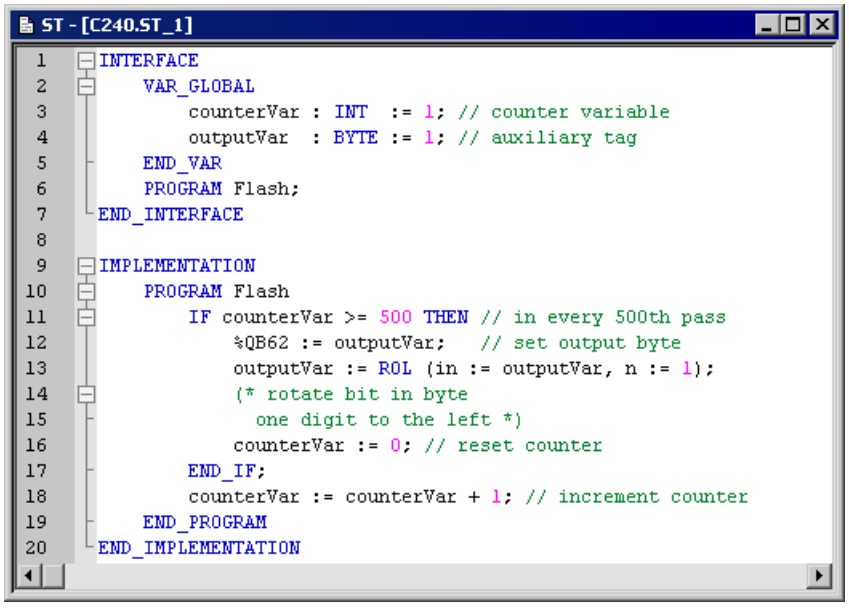


Figure 7-205 ST source file for which all blocks are shown

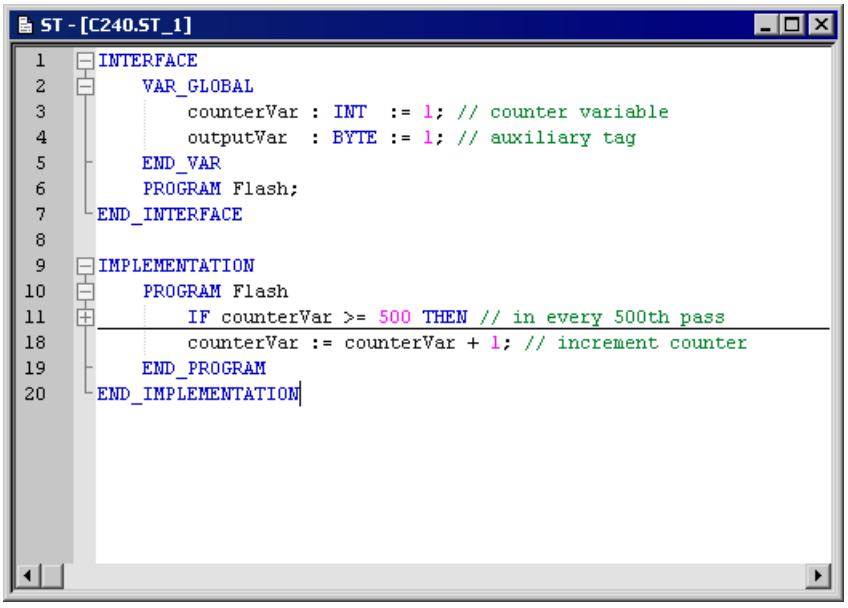


Figure 7-206 ST source file with hidden IF block (including block comment)

Activating or deactivating folding

Folding must be activated in order to be able to show and hide blocks. Once this function has been activated, the column containing the folding information (to the left of the edit area) is displayed.

How to activate or deactivate folding:

- For the active ST source file:
  - Select the **View > Folding** context menu.

  This setting is not saved when the ST source file is closed.
- For all open ST sources:
  - Activate or deactivate the **Folding active** checkbox in the settings of the ST editor (Page 4593).

  This setting is also used for all ST source files opened subsequently.

Showing and hiding blocks

You can only show or hide blocks if folding has been activated for the source file in question (see above). Once this function has been activated, the column containing the folding information is displayed.

How to show or hide blocks:

- Hide an individual block (options):
  - Click the ⊟ (minus) character in the column containing the folding information.
  - Position the cursor on the corresponding line of the block and press the **CTRL+ALT+T** shortcut.

  Only the first line of the block remains visible. All subsequent lines of the block (including lines of subordinate blocks) will be hidden.
  The show/hide status of the subordinate blocks is saved. This is reinstated when individual blocks are shown.
- Show an individual block (options):
  - Click the ⊞ (plus) character in the column containing the folding information.
  - Position the cursor on the visible line of the block and press the **CTRL+ALT+T** shortcut.

  All subsequent lines of the block will be shown. Subordinate blocks are shown as follows: If the show/hide status has been saved (when hiding individual blocks, for example), this will be reinstated.
- Hide all blocks:
  - Press the **CTRL+ALT+C** shortcut.

  All the blocks of the ST source file (including all the subordinate blocks) will be hidden. In each case, only the first line of the 1st-level blocks remain visible (usually INTERFACE and IMPLEMENTATION).
- Show all blocks:
  - Press the **CTRL+ALT+D** shortcut.

  All the blocks of the ST source file (including all the subordinate blocks) will be shown. All the lines of the ST source file will be visible.
- Hide subordinate blocks:
  - Position the cursor on the corresponding line of the block and press the **CTRL+ALT+V** shortcut.

  All the subordinate blocks for the current block will be hidden and the current block itself will be shown. Only the lines of the current block and the first lines of the blocks on the next level will be visible.
- Show subordinate blocks:
  - Position the cursor on the corresponding line of the block and press the **CTRL+ALT+R** shortcut.

  The current block and all subordinate blocks will be shown. All the lines of these blocks will be visible.

Note

The information relating to displayed and hidden blocks is saved in the project when the ST source file is closed.

This information is not transferred when the ST source file is exported (Page 4636).

Splitting the editor window

You can split the ST editor window into two segments, giving you two views of the same ST source file.

Splitting a window or canceling a split

How to split the current editor window into two segments or cancel such a split:

- Select the **ST source file > Split window** menu command.

Following the split, the cursor is located in the upper window segment.

The settings for splitting the editor window are not saved when the ST source file is closed. ST source files are always opened in an entire window (i.e. one that is not split).

Note

The editor window cannot be split if the program status (Page 4951) test function is being used.

The snippet editor is disabled when split window is used.

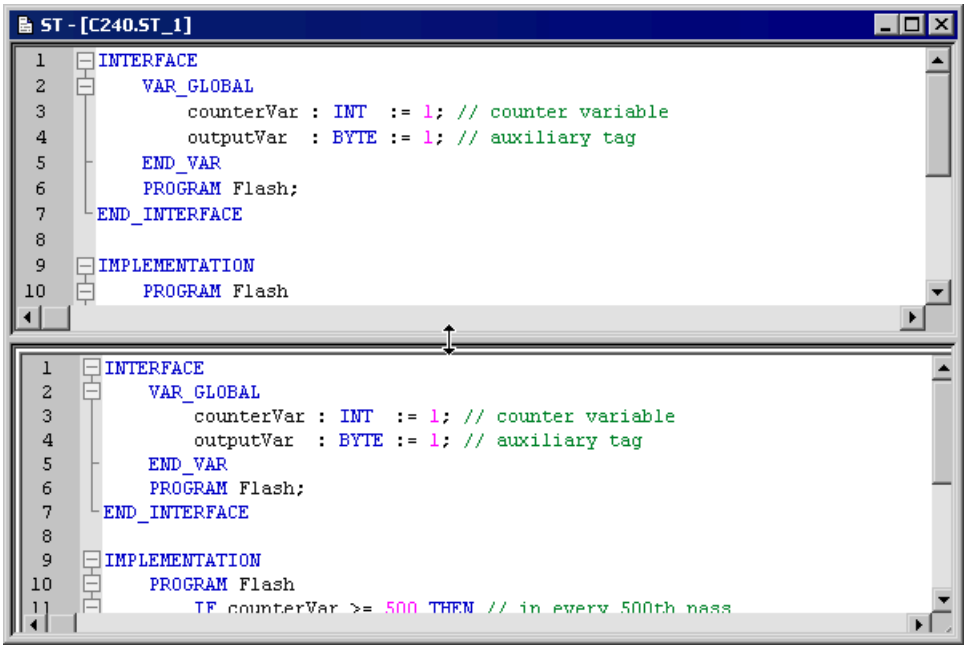


Figure 7-207 ST editor with split window

Adjusting the segment height

You can adjust the height of the two segments to your individual requirements:

1. Use the mouse to move the cursor to the dividing line between the two segments, until it turns into a double-headed arrow (see screenshot above).
2. Press and hold the left mouse button and drag the dividing line to the required position.

Displaying spaces, tabs and line ends

You can display spaces and tabs in the ST source files.

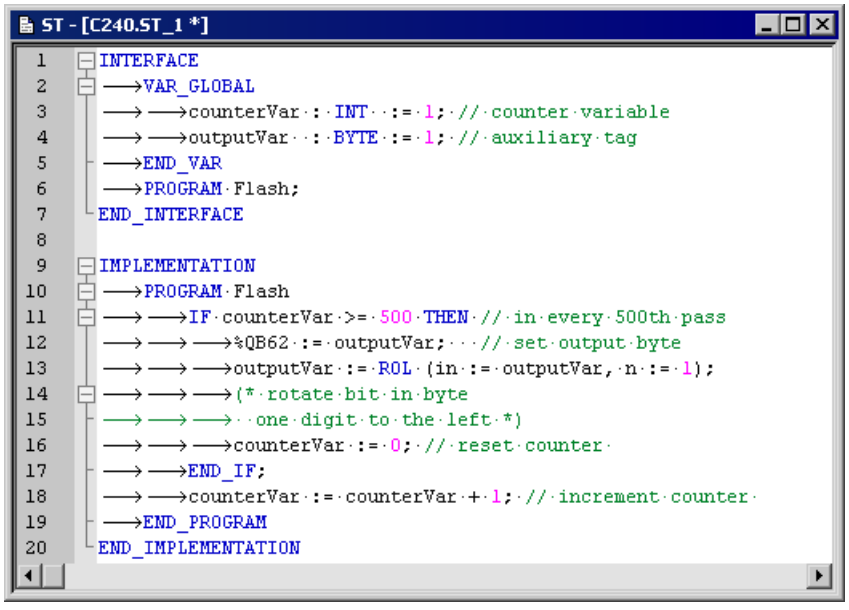


Figure 7-208 ST source file with visible spaces and tabs

In addition, you can display the line end (CR/LF).

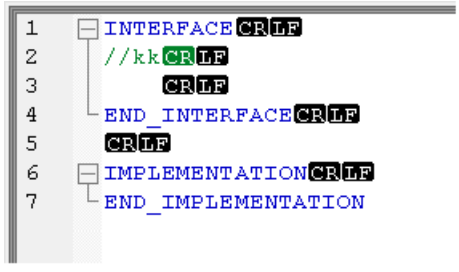


Figure 7-209 ST source with visible line end

Procedure

How to specify whether spaces and tabs are displayed in the active ST source file:

1. Set the cursor in the opened ST source.
2. Select the **View > Formatting symbols** context menu.

This setting is not saved when the ST source is closed.

Changing the font size in the ST editor

You can change the font size of the ST source in the editor. The font size of the line numbers and the size of other display elements (e.g. fold marks, bookmarks) will also be changed.

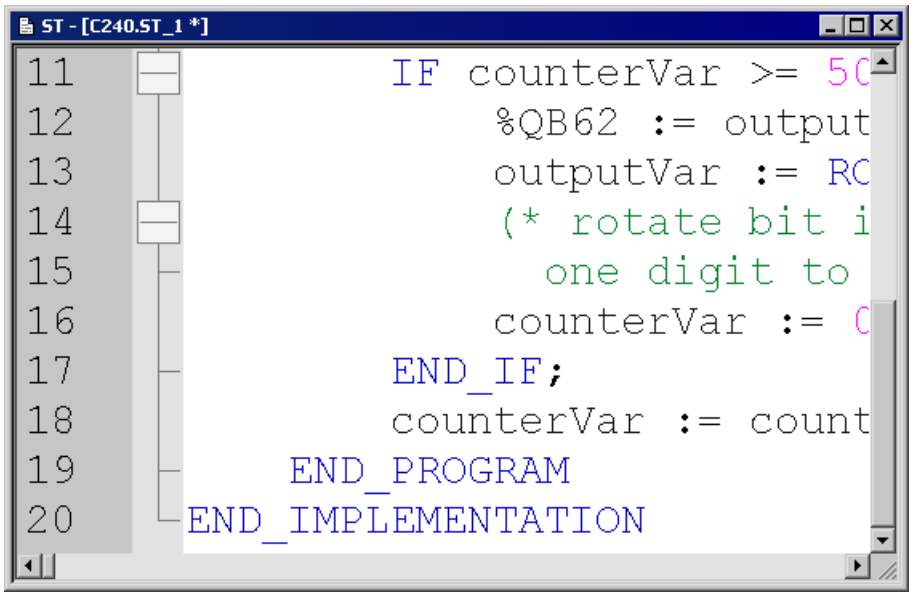


Figure 7-210 Increased size display of the ST source

Proceed as follows

You can change the font size:

- For the current ST source
- For ST source files to be opened subsequently

How to change the font size for the current ST source (alternative):

- Press the **CTRL** key while moving the mouse wheel
- Press concurrently the **CTRL** key and one of the following keys on the numeric block:
  - **ADD** (**+**) to increase,
  - **MINUS** (**-**) to reduce,
  - **DIV** for 100%.

How to change the font size for ST sources to be opened subsequently:

1. Open the settings for the ST editor (see Settings of the ST editor (Page 4593)).
2. Enter the required font size.

This setting will used for all ST sources that will be opened subsequently. It does not affect the currently opened ST sources.

Select text

Selecting lines of text

How to select lines of text:

- With the mouse:
  - With pressed left mouse button, scan the text to be selected.

  or

- With the keyboard or the mouse:
  - Place the cursor with the arrow keys of the keyboard or with the mouse at the start of the text to be selected.
  - Press the **Shift** key while placing the cursor at the end of the text to be selected. Please also refer to the keyboard shortcuts (Page 4619).

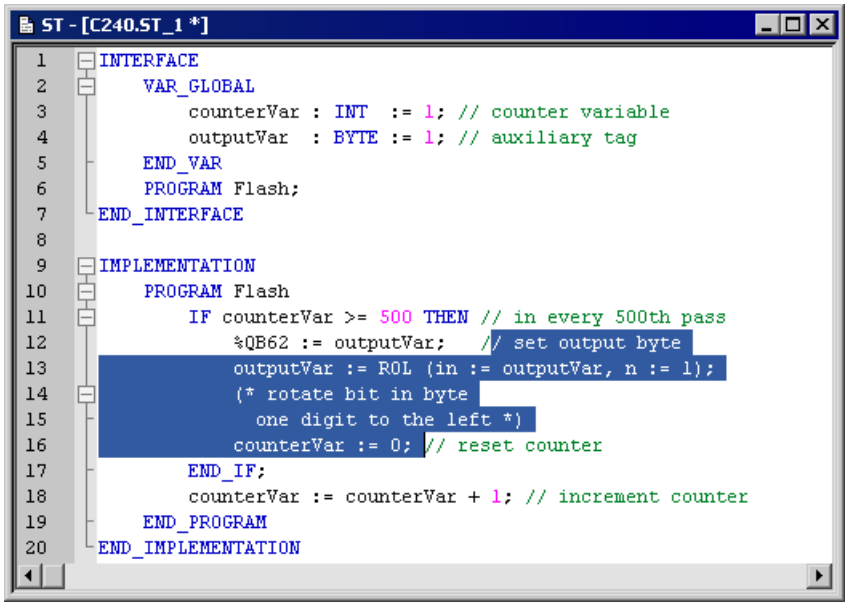


Figure 7-211 ST source with selected lines of text

Selecting columns of text

How to select columns of text:

- With the mouse:
  - Press the **Alt** key while keeping the left mouse button pressed, scan the text to be selected.

  or
- With the keyboard or the mouse:
  - Place the cursor with the arrow keys of the keyboard or with the mouse at the start of the text to be selected.
  - Press the **ALT+SHIFT** key combination while placing the cursor at the end of the text to be selected. Please also refer to the keyboard shortcuts (Page 4619).

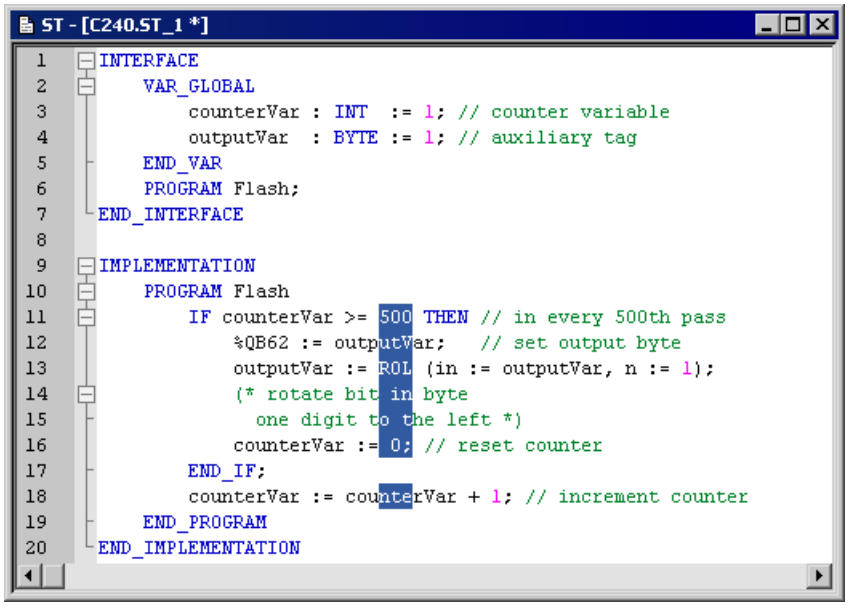


Figure 7-212 ST source with selected columns of text

Selecting a single line

How to select a single line:

- Left-click to the right of the line number of the appropriate line.

Selecting the complete text

How to select the complete text (alternatives):

- Press the **CTRL** key while clicking with the left mouse button in the column with the line numbers.
- Press the **CTRL+A** key combination.

Generating a simple series of numbers (generating a sequence)

You can generate simple series of integer numbers (sequences) that follow a specific pattern in the ST editor. This can be useful, for example, for the initialization of a field.

Procedure

1. In the ST editor, select all the numbers that you want to replace with a series of numbers (column-by-column, for example); see Selecting text (Page 4603).
2. Press the **CTRL+SHIFT+F7** shortcut.

The ST editor interprets all selected numbers as being possible elements and attempts to identify the pattern of the series from the initial elements and to calculate the subsequent elements (see below). The initial elements of the series must represent integer values. After the calculation has been performed, the ST editor replaces all the selected numbers which follow the initial elements with the calculated values.

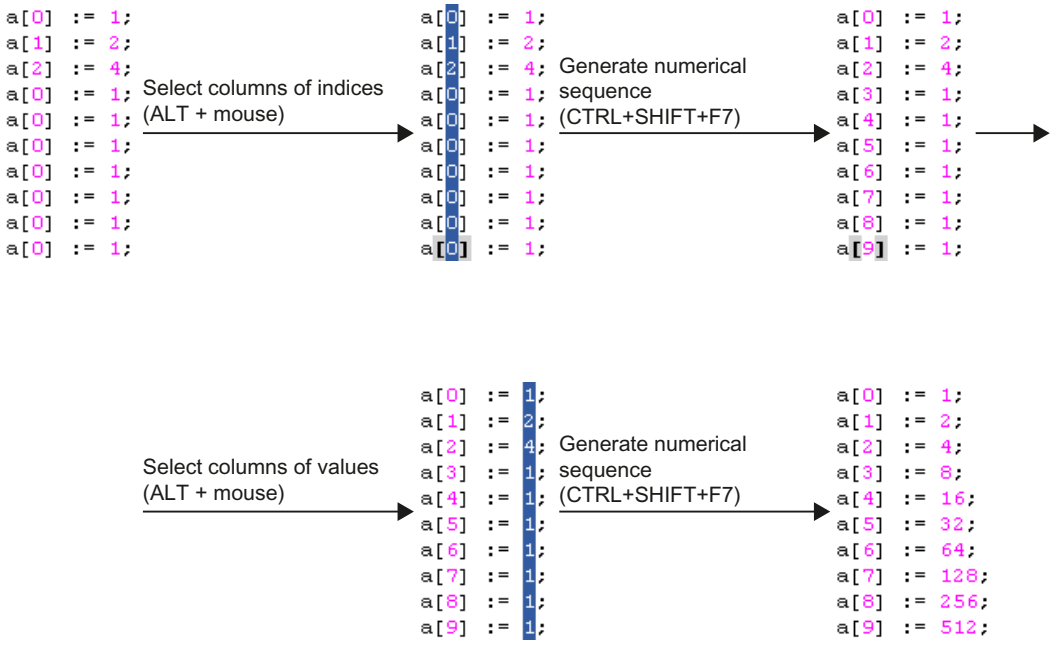


Figure 7-213 Example for generating series of numbers in the ST editor

Calculation of the elements

All the numbers selected in the ST editor are interpreted as being possible elements. The initial elements from which the series is calculated must represent integer values.

When determining the series of numbers, it is assumed that an element a_{n+1} can be determined from the previous element a_n using a linear function with the integer coefficients j and k. This means the following equation applies:

$$a_{n+1} = f(a_n) = j \cdot a_n + k$$

Coefficients j and k are calculated from the initial elements in the series:

1. First of all, an attempt is made to determine the integer coefficients j and k from the three initial elements of the series of numbers a_1, a_2 and a_3 ($j \geq 0$).
2. If this is not possible, only the first two elements a_1 and a_2 are used for the calculation:
   - Coefficient j will be set to 1.
   - Coefficient k will be determined from the resulting formula $a_2 = a_1 + k$.

These values are used to calculate the series of numbers. All selected numbers will be replaced by the calculated elements; the initial elements of the series which are used remain unchanged.

The following table shows some examples for series of numbers calculated in this manner.

Table 7-358 Examples for linear series of numbers with specified initial elements

Initial elements of the series			Calculated result			Resulting series of numbers
a_1	a_2	a_3	j	k	Equation	
0	1	2	1	1	$a_{n+1} = a_n + 1$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ...
1	2	3	1	1	$a_{n+1} = a_n + 1$	1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ...
0	2	4	1	2	$a_{n+1} = a_n + 2$	0, 2, 4, 6, 8, 10, 12, 14, 16, 18 ...
1	2	4	2	0	$a_{n+1} = 2a_n$	1, 2, 4, 8, 16, 32, 64, 128, 256, 512 ...
2	3	5	2	-1	$a_{n+1} = 2a_n - 1$	2, 3, 5, 9, 17, 33, 65, 129, 257, 513 ...
2	1	1	0	1	$a_{n+1} = 1$	2, 1, 1, 1, 1, 1, 1, 1, 1, 1 ...
3	2	1	1	-1	$a_{n+1} = a_n - 1$	3, 2, 1, 0, -1, -2, -3, -4, -5, -6 ...
0	1	0 [1]	1	1	$a_{n+1} = a_n + 1$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ...
0	0	2 [1]	1	0	$a_{n+1} = a_n$	0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ...

[1] Only the initial elements a_1 and a_2 of the series are used, a_3 is ignored and will be calculated.

Note

The initial elements of the series used for the calculation must represent integer values. However, they can also be entered as floating-point numbers.

When the series is calculated, the numbers to be taken into account must all be selected.

The selected area must only contain numbers and certain special characters (e.g. opening and closing brackets, commas).

The first time an impermissible character is encountered in the selected area, the calculation of the series will be aborted.

See the examples in the following figure.

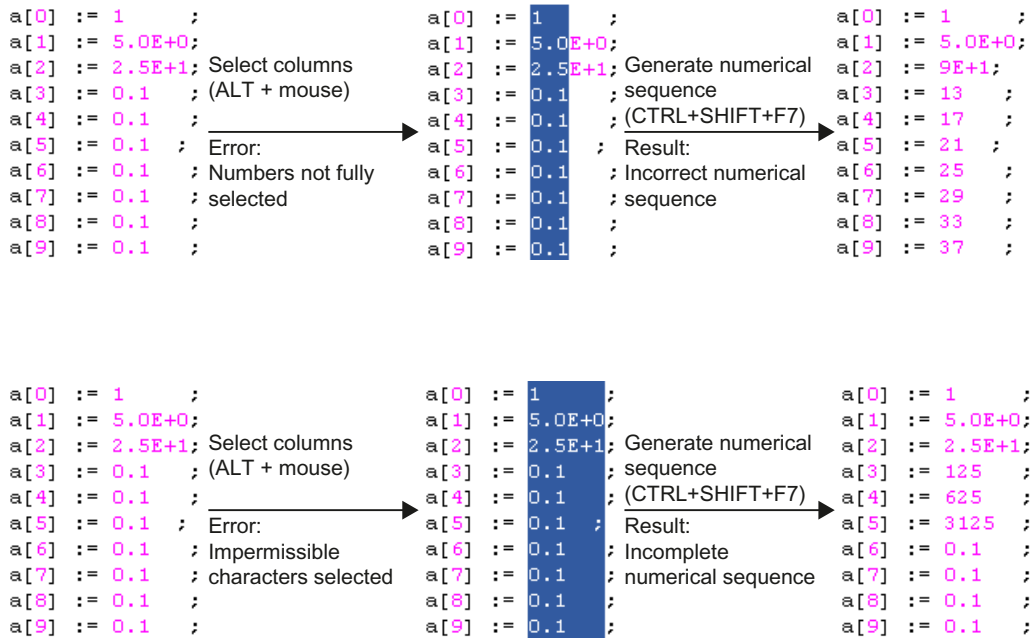


Figure 7-214 Examples of how incorrect selections cause unwanted results when generating a series of numbers in the ST editor

Use bookmarks

You can set bookmarks in the ST editor. This allows you to jump to specific selected lines within the ST source file.

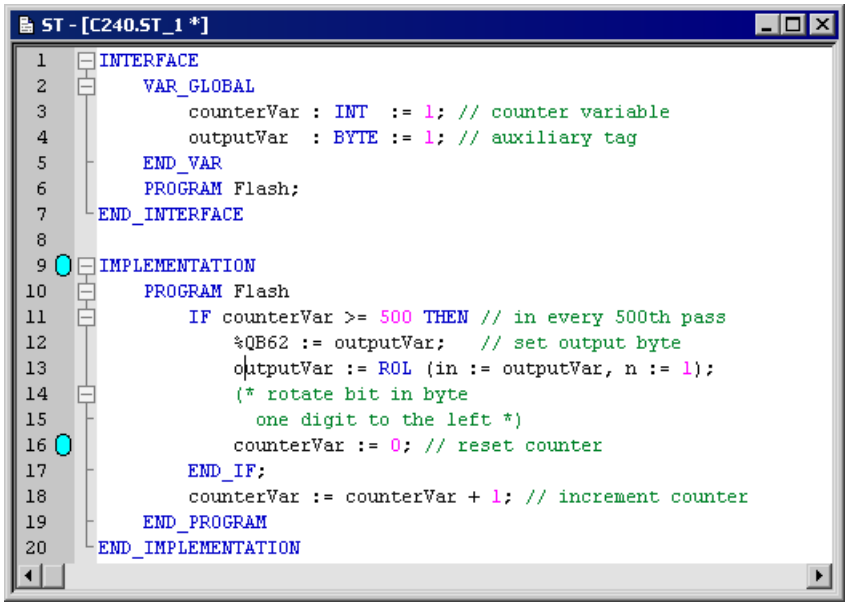


Figure 7-215 ST source with bookmarks

Setting and deleting bookmarks

How to set a bookmark for a line of the active ST source file or to delete an existing bookmark:

- With the keyboard and the mouse:
  - Press the **Ctrl** key.
  - Simultaneously, click with the left mouse button at the right-hand side next to the line number of the appropriate line.
- With the mouse:
  - Set the cursor in the appropriate line of the ST source.
  - Select the **Bookmarks > Insert/remove bookmark** context menu.

Note

Bookmarks are saved when the ST source file is closed.

Jump to bookmark

How to jump to the next bookmark within the ST source:

- Select the **Bookmarks > Next bookmark** context menu.

How to jump to the previous bookmark within the ST source:

- Select the **Bookmarks > Previous bookmark** context menu.

Delete all bookmarks

How to delete all bookmarks in an ST source:

- Select the **Bookmarks > Remove all bookmarks** context menu.

Automatic completion

In the ST editor, you can automatically complete identifiers. A selection list with identifiers that begin with the previously entered characters will be displayed.

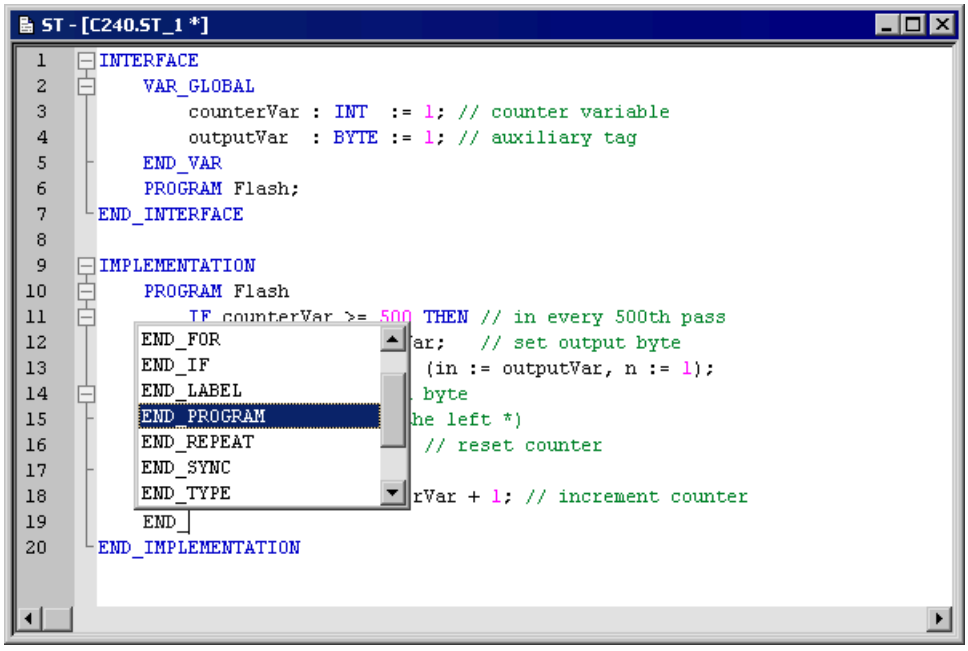


Figure 7-216 ST editor, automatic completion of an identifier (e.g. END_)

Procedure

How to automatically complete an identifier:

1. Write the first characters of the identifier (e.g. the letters of a word).
2. Press the **Ctrl+space** key combination.
   The selection possibilities are displayed in a window.
3. Expand or refine the selection options displayed:
   - Enter additional characters
   - Delete characters
   - Use the left/right arrow keys to move the cursor
4. Use the up/down arrow keys to select the required identifier.
5. Accept the identifier. The current word is overwritten.

Note

If only a single identifier is offered for selection, the selection window will not be opened and the identifier completed immediately.

Functional description

The following identifiers that begin with the specified character will be offered:

- Keywords of the Structured Text language
- Identifiers from the command library
- For technology objects including their system variables and configuration data
- Identifiers of the own ST source:
  - Program organization units (POU)
  - Data types
  - Variables and constants
  - Structure elements
- Identifiers from program sources used

Note

Identifiers from the own ST source or from program sources used will be displayed correctly only when the corresponding program source has been compiled.

The display is made context-sensitive, only those types of identifiers that are appropriate at the associated location of the ST source will be offered:

- Within a declaration block, only data types and keywords
- Within a program organization unit (POU), no data types
- For a structure (e.g. var_struct.xx), only structure components

Opening a called block

From the ST editor you can directly open and edit a user-defined program organization unit (POU), such as a function or a function block, which is called in the ST source file.

1. Position the cursor in the identifier of a program organization unit (POU), e.g. when calling a function or declaring an instance of a function block.
2. Select the **Open called block** context menu.

The subsequent response will depend on the programming language in which the source file of the called POU was written:

- SIMOTION ST programming language:
  The corresponding ST source file is opened, if necessary. The cursor is positioned at the start of the relevant POU in the implementation section.
  The cursor jumps to the start of the POU in the implementation section within the same ST source file.
  If a library is connected with an online CPU and is consistent online, the online functions are available under this library for all units.
- SIMOTION MCC or SIMOTION LAD/FBD programming languages:
  The corresponding POU (MCC chart, LAD/FBD program) is opened in the associated editor.

Code snippets

After the complete input of certain keywords (such as IF, WHILE, VAR), a window opens above the cursor to display a code snippet for the associated keyword (e.g. IF (*condition*) ... END_IF).

Further keywords with the same initial letter with available code snippets are displayed in the lower part of this window.

Pressing the **TAB** key transfers this code snippet to the ST source file. Pressing any other key closes this window.

Activating and deactivating code snippets

Code snippets must be activated before they can be used.

How to activate or deactivate code snippets:

- For the active ST source file:
  - Select the **View > Snippet Editor** context menu.

  This setting is not saved when the ST source file is closed.
- For all open ST source files:
  - Activate or deactivate the **Code snippets** checkbox in the ST editor settings.

  This setting is also used for all ST source files opened subsequently.

Creating user-defined snippets

You can create your own snippets for use in the ST editor. To do this, use the snippet editor, which you can display in the partitioned window at the same time as the ST editor. Observe the following rules for creation:

Rule	Description
Creating and naming snippets	You create new snippets with the snippet editor. Enter a unique name. Permissible characters are A-Z, a-z, 0-9 and _.
Renaming snippets	You can rename snippets that have already been created.
Deleting snippets	You can delete snippets that have already been created. Perform the Save function afterwards. Only then can the name of the deleted snippet be used again for new snippets.

Working with the snippet editor

Display the snippet editor via "ST source file > Snippet editor". Alternatively, you can reach the snippet editor via "ST editor > View > Snippet editor". The snippet editor shares the working area with the associated ST editor. In this way, the snippet editor is always tied to one, and only one, existing ST editor window. The snippet editor consists of three main sections:

- Snippet toolbar
- Snippet tree
- Snippet code editor

Snippet toolbar

The toolbar contains the following functionalities:

Function	Description
Load snippets into the editor – without saving	This function loads the changes that have been made in the snippet editor into the ST editor.
Save	This function saves the selected snippet or all of the snippets under the selected node in the tree.
Save all	This function saves all of the changes that have been made since the last save.
Reject	This function rejects the changes that have been made to the selected snippet or all of the changes in the selected node in the tree.
Reject all	This function rejects all of the changes that have been made since the last save.
Import	This function loads the user-defined snippets from the XML input file. The validity of the file is checked. If there is an error, a corresponding message is output. All existing user-defined snippers are deleted during the import.
Export	This function saves the user-defined snippets into an XML file. For this purpose, you create a new file or overwrite an existing one. The changes in the snippet are saved before the export.
Help	This function calls the context-sensitive online help.
Close	This function closes the snippet editor. Save your snippets before closing the editor.

Snippet tree

The snippet editor shows the hierarchy of all of the available snippets in a clear tree structure. The tree has a root node "Snippets" with two subordinate elements by the name of "STSnippets" for non_OOP- and "STSnippetsOOP for OOP-relevant snippets". Both contain pre-defined and user-defined snippets. The snippet IDs in the snippet tree are sorted alphabetically and all of the pre-defined snippets (recognizable by the lock icon) are listed first, in alphabetical order.

Only user-defined snippets can be created, changed, or deleted. Master elements, scope IDs or pre-defined snippets cannot be changed, deleted, or created. After you have created a new snippet, the new snippet key (name) will be alphabetically sorted into the user-defined snippets in the right position in the tree.

Snippet code editor

You insert the snippets in the snippet tree via the context menu. After inserting, assign the name and edit the snippet in the snippet code editor.

The snippet code editor is a simple text editor field in which you edit the associated part of the code (the currently selected element in the snippet tree). The text can be entered or inserted from the clipboard. Pre-defined snippets can be selected and copied, but are not edited.

Navigating between the identifiers

Snippets can have more identifiers marked with #identifier as placeholders to indicate that an identifier is missing. When a snippet is inserted into the ST source, the first #identifier that can be overwritten by the correct identifier is always marked. You then need to select all other identifiers. You jump to the next identifier with TAB. You jump to the previous identifier with Shift +TAB.

Other help for the ST editor

Display bracket pairs

The two brackets of a bracket-pair can be optically highlighted.

To do this, place the cursor next to a bracket. The editor attempts to find the associated brackets of the pair and highlights both brackets, where applicable. This simplifies the recognition of bracket pairs, in particular for nesting.

How to switch this function on or off:

- For the active ST source file:
  - Select the **View > Display bracket pairs** context menu.

  This setting is not saved when the ST source file is closed.
- For all open ST sources:
  - Activate or deactivate the **Display bracket pairs** checkbox in the ST editor settings (Page 4593).

  This setting is also used for all ST source files opened subsequently.

Showing and hiding line numbering

Line numbers can be displayed in the ST editor:

How to switch this function on or off:

- For the active ST source file:
  - Select the **View > Line numbering** context menu.

  This setting is not saved when the ST source file is closed.
- For all open ST sources:
  - Activate or deactivate the **Display line numbers** checkbox in the ST editor settings (Page 4593).

  This setting is also used for all ST source files opened subsequently.

Changing upper/lower case

You can change the case of selected text to upper case or lower case:

1. Select the text whose case you want to change; see Selecting text.
2. Select the **Selection in upper case** or **Selection in lower case** context menu.

Tool tip display

The ST editor provides support for some identifiers by displaying tool tips. These are shown if you briefly hover the cursor over the identifier.

For the identifiers of variables, for example, the associated data type is displayed in the tool tip. More tool tips will be available soon.

Using the command library

The command library is a tab in the project navigator. It contains the available system functions, system function blocks, and operators.

You can drag these elements from the command library to the ST editor window with drag&drop.

ST editor toolbar

This toolbar contains important operating actions for programming:

Table 7-359 ST editor toolbar

Symbol	Meaning
	Accept and compile Click this symbol to transfer the active ST source file to the project and compile it into executable code. See: Starting the compiler (Page 4622).
	Insert ST source file Click this icon to create a new ST source file. The icon is active only when the **PROGRAMS** folder where the ST source file is to be saved is selected in the project navigator. See: Insert ST source file (Page 4587).
	Program status Click this icon to start the **program status** test mode. During the program execution, you can monitor the values of the variables marked in the ST source. The following prerequisites are necessary: 1. The program must be compiled with the appropriate compiler option. 2. The project and the program must be loaded into the target system. 3. An online connection to the target system must have been established. Reclick this icon to end the **program status**. See: Using the program status (Page 4951). **Note:** Status program and the snippet editor cannot be active at the same time. • Close the snippet editor to open the status program.
	Stop monitoring of the program variables Click this icon in the **program status** test mode to stop the monitoring of the program variables. See Using the program status (Page 4951).
	Continue monitoring of the program variables Click this icon in the **program status** test mode to continue the monitoring of the program variables. See: Using the program status (Page 4951).
	Refresh Click this symbol in the **program status** test mode to force updating of the values displayed. The monitoring of the program variables must have been activated. See: Using the program status (Page 4951).

Symbol	Meaning
	Format current selection Click this symbol to indent the blocks in the selected text area by one tab width, in accordance with the block hierarchy. See Indentations and tabs (Page 4594).
	Formatting symbols on/off Click this symbol to show or hide blanks and tabs in the active ST source file. See Displaying blanks and tabs (Page 4601).
	Insert/remove bookmark Click this symbol to set a bookmark in the current line of the active ST source file or to delete an existing bookmark. See: Using bookmarks (Page 4607).
	Next bookmark Click this symbol to jump to the next bookmark in the active ST source file. See: Using bookmarks (Page 4607).
	Previous bookmark Click this symbol to jump to the previous bookmark in the active ST source file. See: Using bookmarks (Page 4607).
	Remove all bookmarks Click this symbol to remove all bookmarks from the active ST source file. See: Using bookmarks (Page 4607).
	Go to start of block Click this symbol to move the cursor to the start of the current or higher-level block.
	Go to end of block Click this symbol to move the cursor to the end of the current block.
	Insert comment character Click this icon to convert to comment on the selected lines or line fragments or to insert a comment character. See Comments (Page 4670).
	Remove comment character Click this icon to reactivate commented lines or line fragments. See Comments (Page 4670).

ST editor context menu

The ST editor context menu is shown if you right-click in an open ST source file.

Depending on the active application/editor or the mode (ONLINE/OFFLINE), certain commands are not displayed or cannot be selected.

As far as editor functions are concerned, this context menu also applies to the script editor.

You can select the following functions:

Table 7-360 ST editor and script editor context menu

Function	Meaning/information
Open called block [1]	If the cursor is positioned in the identifier of a user-defined program organization unit (POU), e.g. when calling a function or declaring an instance of a function block: Select this command to open and edit the source of this POU (ST source file, MCC chart, LAD/FBD program). See: Open called block (Page 4610).
Refactoring	You can change the case or rename. The function provides a list of suggested renames of the symbols used in the actual unit, which differ in terms of notation from the symbols declared in the cross-references. You can decide which changes need to be applied. To avoid inconsistencies, compile the project afterwards.
Surround with	Select this command to surround the selected text with pre-defined code snippets. E.g. surround the selected variable with an If instruction.
Cut	Select this command to cut the selected area from the ST source file and save it to the clipboard.
Copy	Select this command to copy the selected area to the clipboard.
Paste	Select this command to insert the contents of the clipboard into the ST source file at the current cursor position.
Copying text with syntax coloring	Select this command to copy the selected area in RTF format to the clipboard. This allows the insertion of text including syntax coloring into text processing programs. The copied text cannot be inserted into the ST editor.
Delete	Select this command to delete the selected area or the character to the right of the cursor.
Select all	Select this command to select all of the text in the ST source file. See: Select text (Page 4603).
View	

Function		Meaning/information
	Line numbering on/off	Select this command to show or hide the line numbers in the active ST source file. See: Other ST editor tools (Page 4613).
	Indent help	Select this command to highlight the indents and outdents for blocks in the active ST source file by means of vertical auxiliary lines (in accordance with the set tab width). See: Indentations and tabs (Page 4594).
	Display bracket pairs	Select this command to highlight both brackets of a pair in the active ST source file, if the cursor is positioned at one of the two brackets. See: Other ST editor tools (Page 4613).
	Formatting symbols	Select this command to show or hide blanks and tabs in the active ST source file. See: Displaying blanks and tabs (Page 4601).
	Folding	Select this command to activate or deactivate folding in the active ST source file. The folding information in the active ST source file is then displayed or hidden. See: Folding (showing and hiding blocks) (Page 4596).
	Format current selection	Select this command to indent the blocks in the selected text area by one tab width, in accordance with the block hierarchy. See: Indentations and tabs (Page 4594).
	Split window	Select this command to split the active window of the ST editor into two segments horizontally, giving you two views of the same ST source file. See: Splitting the editor window (Page 4600).
	Code snippets	Select this command to activate or deactivate code snippets in the activate ST source file. If active, after input of certain keywords (such as IF, WHILE, VAR), a window opens to display a code snippet for the associated keyword (e.g. IF (*condition*) ... END_IF). This code snippet can be transferred to the ST source file. See: Code snippets (Page 4611).
Bookmarks		
	Insert/remove bookmark	Select this command to set a bookmark in the current line of the active ST source file or to delete a bookmark which has been set there. See: Using bookmarks (Page 4607).
	Next bookmark	Select this command to jump to the next bookmark in the active ST source file. See: Using bookmarks (Page 4607).
	Previous bookmark	Select this command to jump to the previous bookmark in the active ST source file. See: Using bookmarks (Page 4607).
	Remove all bookmarks	Select this command to remove all bookmarks from the active ST source file. See: Using bookmarks (Page 4607).
Print		
	Complete source	Select this command to print the complete active ST source file.
	Selection	Select this command to print the selected lines of the active ST source file.
Variable declaration		Select this command to declare a variable "on-the-fly". See: Declaring variables in the Variable declaration dialog box ("on-the-fly" variable declaration) (Page 4860).
Selection in upper case		Select this command to change the selected text to upper case. See: Other ST editor tools (Page 4613).
Selection in lower case		Select this command to change the selected text to lower case. See: Other ST editor tools (Page 4613).

Function		Meaning/information
Code indentation		Select this command to format the text automatically.
Indent selected area		• If no text is selected: Select this command to move the text on the right of the cursor to the next tab position. • If text is selected in one single line: Select this command to delete the selected text and move the subsequent text to the next tab position. • If text is selected in multiple lines: Select this command to indent the selected area (Page 4594). A tab character ($09) or the equivalent number of spaces ($20) will be inserted, depending on the settings for the ST editor (Page 4593).
Undo selected area		• If no text is selected: Select this command to move the cursor to the previous tab position. • If text is selected in one single line: Select this command to cancel the selection and move the cursor to the previous tab position. • If text is selected in multiple lines: Select this command to outdent the selected area (Page 4594).
Go to start of block		Select this command to move the cursor to the start of the current or higher-level block.
Go to end of block		Select this command to move the cursor to the end of the current block.
Go to start of block, level 0		Select this command to move the cursor to the start of the higher-level block, 1st level.
Go to start of block, level 1		Select this command to move the cursor to the start of the higher-level block, 2nd level.
Go to line		Select this command to move the cursor to a specific line. A dialog box for specifying the line number is opened.
Set/remove breakpoint [1]		Select this command to set a breakpoint at the selected code position or to remove an existing breakpoint. See: Setting breakpoints (Page 4959).
Activate/deactivate breakpoint [1]		Select this command to activate or deactivate the breakpoint at the selected code position. See: Activating breakpoints (Page 4967).
Add to watch table [1]		
	New watch table	If the cursor is within a variable or if the variable is selected: Select this command to create a new watch table and to take it into this variable.
	(Name of a watch table)	If the cursor is within a variable or if the variable is selected: Select this command to take the variable into the selected watch table
Go to [1]		
	Local use >>	If the cursor is within a variable or if the variable is selected: Select this command to jump to the next use of the variable in the ST source file.
	Local use <<	If the cursor is within a variable or if the variable is selected: Select this command to jump back to the previous use of the variable in the ST source file.

Function		Meaning/information
	Declaration position	If the cursor is within a variable or if the variable is selected: Select this command to jump to the declaration position of the variable. If the variable is declared in a used source or library, the source or library is opened.
	Points of use	If the cursor is within a variable or if the variable is selected: Select this command to list all the points of use of the variable in the detail view.

[1] Not available in the script editor context menu.

Shortcuts

The ST editor also provides keyboard shortcuts. Some commands can also be called via the **Edit** or **ST editor** menus:

The keyboard shortcuts related to editor functions also apply to the script editor.

Table 7-361 Keyboard shortcuts for ST editor and script editor

Shortcuts	Description
F2	Jump to the next bookmark.
F3	Find next (menu **Edit > Find next**). A search is performed on the last text in the search field, even if it has been closed (see CTRL +F).
F9 [1]	Set or remove a breakpoint (menu **Debug > Set/remove breakpoint**).
F12 [1]	Activate or deactivate a set breakpoint (menu **Debug > Activate/deactivate breakpoint**).
BACK	Delete the character to the left of the cursor.
INS	Switch between insert mode and overwrite mode.
DEL	Delete the selected area or the character to the right of the cursor (menu **Edit > Delete**).
Arrow key	Move the cursor.
POS1	Move cursor to the beginning of the line.
END	Move cursor to the end of the line.
PG UP	Move up one page. The cursor follows.
PG DN	Move down one page. The cursor follows.
TAB	• If no text is selected: Move the text on the right of the cursor to the next tab position. • If text is selected in one single line: Delete the selected text and move the subsequent text to the next tab position. • If text is selected in multiple lines: Indent selected area. A tab character ($09) or the equivalent number of spaces ($20) will be inserted, depending on the settings for the ST editor.
SHIFT+F2	Jump to the previous bookmark.
SHIFT+BACK	Delete the character to the left of the cursor.
SHIFT+INS	Paste clipboard contents (menu **Edit > Paste**).
SHIFT+DEL	Cut the selected area (menu **Edit > Cut**).
SHIFT+Arrow key	Select line of text.
SHIFT+POS1	Select text back to the beginning of the line.

Shortcuts	Description
SHIFT+END	Select text to the end of the line.
SHIFT+PG UP	Move up one page. Select lines of text up to the new cursor position.
SHIFT+PG DN	Move down one page. Select lines of text up to the new cursor position.
SHIFT+TAB	• If no text is selected: Jump to the preceding tab position. • If text is selected in one single line: Jump to the preceding tab position. • If text is selected in multiple lines: Outdent selected area.
CTRL+A	Select all text (menu **Edit > Select All**).
CTRL+B [1]	Accept and compile ST source file (menu **ST source file > Accept and compile**).
CTRL+C	Copy the selected area to the clipboard (menu **Edit > Copy**).
CTRL+D	Duplicate the current line or the area selected.
CTRL+F	Find text in ST source file (menu **Edit > Find**) If text is selected in a single line, this is taken into the search screen form.
CTRL+H	Replace text in ST source file (menu **Edit > Replace**).
CTRL+J	Display the next search result in the project-wide search (menu **Edit > Display next position**).
CTRL+L	Copy the current line or the selected area to the clipboard.
CTRL+U	Change selected text to lower case.
CTRL+V	Paste clipboard contents (menu **Edit > Paste**).
CTRL+X	Cut the selected area (menu **Edit > Cut**).
CTRL+Y	Redo the last action (menu **Edit > Redo**).
CTRL+Z	Undo the last action (menu **Edit > Undo**).
CTRL+space	Automatic completion
CTRL+F2	Set or remove bookmarks.
CTRL+F4	Close the active window (e.g. menu **ST source file > Close**).
CTRL+F5 [1]	Remove all the breakpoints (in all the program source) in the SIMOTION device (menu **Debug > Remove all breakpoints**).
CTRL+F7 [1]	Activate or deactivate the program status function (menu **ST source file > Program status on/off**).
CTRL+F8 [1]	Continue to execute the program at the activated breakpoint (menu **Debug > Continue**).
CTRL+BACK	Delete the word to the left of the cursor.
CONTROL+INS	Copy the selected area to the clipboard (menu **Edit > Copy**).
CTRL+DEL	Delete the word to the right of the cursor.
CTRL+arrow key (left/right)	Move cursor left or right by one word.
CTRL+arrow key (up/down)	Scroll one row up or down. The cursor remains in the same position for as long as it is visible in the window.
CTRL+POS1	Move cursor to the beginning of the ST source file.
CTRL+END	Move cursor to the end of the ST source file.
CTRL+SHIFT+B	Highlight bracket pairs in the current ST source file.
CTRL+SHIFT+F	Search for texts within the project (menu **Edit > Search in the project**)
CTRL+SHIFT+G	Replace texts within the project (menu **Edit > Replace in the project**)
CTRL+SHIFT+H	Assign parameters to a call. Under development.
CTRL+SHIFT+L	Go to a line in the ST editor.

Shortcuts	Description
CTRL+SHIFT+S	Code indentation.
CTRL+SHIFT+U	Change selected text to upper case.
CTRL+SHIFT+F2	Remove all bookmarks from the ST source file.
CTRL+SHIFT+F3	Arrange windows, tile horizontally.
CTRL+SHIFT+F5	Arrange windows, tile vertically.
CTRL+SHIFT+F7	Generate a simple series of numbers (sequence) in the selected area.
CTRL+SHIFT+F8	Format selected area.
CTRL+SHIFT+F9	Move cursor to the start of the current or higher-level block.
CTRL+SHIFT+F10	Move cursor to the end of the current block.
CTRL+SHIFT+F11	Move cursor to the start of the higher-level block, 1st level.
CTRL+SHIFT+F12	Move cursor to the start of the higher-level block, 2nd level.
CTRL+SHIFT+BACK	Delete text to the left of the cursor up to the beginning of the line.
CTRL+SHIFT+DEL	Delete text to the right of the cursor up to the end of the line.
CTRL+SHIFT+arrow key (left/ right)	Select word to the left or right of the cursor.
CTRL+SHIFT+POS1	Select lines of text back to the beginning of the ST source file.
CTRL+SHIFT+END	Select lines of text up to the end of the ST source file.
CTRL+ALT+C	Folding: Hide all blocks of the current ST source file.
CTRL+ALT+D	Folding: Show all blocks of the current ST source file.
CTRL+ALT+F	Folding: Show or hide folding information in the current ST source file.
CTRL+ALT+I	Show indentation level in the current ST source file.
CTRL+ALT+L	Show or hide line numbers in the current ST source file.
CTRL+ALT+O [1]	If the cursor is in the identifier of a program organization unit (POU): Open called block, i.e. open program source of the POU and position the cursor.
CTRL+ALT+R	Folding: Show all subordinate blocks.
CTRL+ALT+S	Split window or cancel split (menu **ST source file > Split window**).
CTRL+ALT+T	Folding: Show/hide block.
CTRL+ALT+V	Folding: Hide all subordinate blocks.
CTRL+ALT+W	Show or hide blanks and tabs in the current ST source file.
CTRL+ADD (numeric keypad)	Increase font size in the current ST source file.
CTRL+MINUS (numeric keypad)	Decrease font size in the current ST source file.
CTRL+DIV (numeric keypad)	Change font size in the current ST source file to 100%.
ALT+N	Declare variable "on-the-fly" in the Variable declaration dialog box (Page 4860).
ALT+SHIFT+L	Change selected text to upper case.
ALT+SHIFT+S	Activate or deactivate code snippets (Page 4611).
ALT+SHIFT+U	Change selected text to lower case.
ALT+SHIFT+Arrow key	Select text by column.
ALT+SHIFT+POS1	Select columns of text back to the beginning of the line.
ALT+SHIFT+END	Select columns of text to the end of the line.
ALT+SHIFT+Pg Up	Move down one page. Select columns of text up to the new cursor position.
ALT+SHIFT+PG DN	Move down one page. Select columns of text up to the new cursor position.
ALT+Arrow key	Moves the selected code section up or down in the editor.

[1] Keyboard shortcut does not apply to the script editor.

Table 7-362 Combined keyboard and mouse actions for ST editor and script editor

Keyboard	Mouse	Description
	Single left-click in text	Set cursor.
	Double left-click in text	Select word.
	Press left button and drag mouse	Select line of text.
	Single left-click on line number	Select line.
	Turn mouse wheel	Scroll vertically. Cursor remains unchanged.
SHIFT	Single left-click in text	Select line of text.
SHIFT	Turn mouse wheel	Scroll horizontally.
CTRL	Single left-click on line number	Select all text (menu **Edit > Select All**).
CTRL	Single left-click in bookmark column	Set or delete bookmarks.
CTRL	Turn mouse wheel	Change font size.
ALT	Press left button and drag mouse	Select text by column.
ALT+SHIFT	Single left-click in text	Select text by column.

Starting the compiler

Requirement

The ST source file has been opened with the ST editor.

Proceed as follows

1. Click in the window with the ST editor. The dynamic ST source file menu appears.
2. Select the **ST source file** > **Accept and compile** menu command.

Note

The ST source file menu is dynamic. It only appears if the window of the ST editor is active.

The compiler checks the syntax of the ST source file. The "Compile/check output" tab of the detail view displays the successful compilation of the source text or compiler errors. The error details include: The name of the ST source file, the number of the line in which the error occurred, the error number and the error description.

Help for the error correction

To obtain help during error correction:

- Double-click the error message in the **Compile/check output** tab of the detail view.

The cursor is placed at the relevant line in the ST source file.

Making settings for the compiler

You can define the compiler settings (compiler options) as follows:

- Globally for the SIMOTION project, valid for all programming languages, see Global settings of the compiler (Page 4623)
- Locally for an individual ST source file within the SIMOTION project, see Local settings of the compiler (Page 4626)

Global compiler settings

The global settings are valid for all programming languages within the SIMOTION project.

Procedure

1. Select the menu **Options > Settings**.
2. Select the **Compiler** tab.
3. Define the settings according to the following table.
4. Confirm with **OK**.

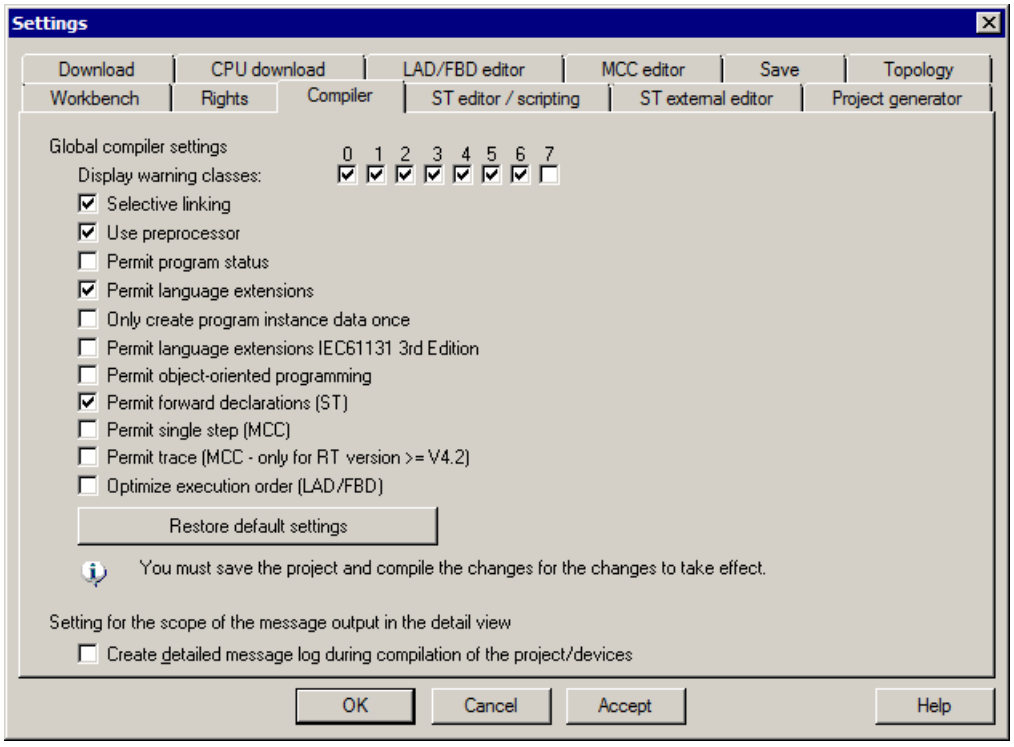


Figure 7-217 Global compiler settings

Parameter

Table 7-363 Parameters for global compiler settings

Parameter	Description
Display warning classes[1]	In addition to error messages, the compiler can issue warnings and information. You can set the scope of the warning messages to be output: **Active**: The compiler issues warnings and information for the selected class. **Inactive**: The compiler suppresses warnings and information for the respective class. See also Meanings of the warning classes (Page 4633).
Selective linking[1]	**Active** (standard): Unused code is removed from the executable program. **Inactive**: Unused code is retained in the executable program.
Use preprocessor[1]	**Active**: Preprocessor is used. **Inactive** (standard): Preprocessor is not used. See Controlling the preprocessor (Page 4918).
Enable program status[1]	**Active**: Additional program code is generated to enable monitoring of program variables (including local variables). **Inactive** (standard): Program status not possible. See Properties of the program status (Page 4949).
Permit language extensions[1]	**Active**: Language elements are permitted that do not comply with IEC 61131-3. • Direct bit access to variables of a bit data type (Page 4705) • Accessing the input parameter of a function block while outside the function block (Page 4756) • Calling a program while in a different program (Page 4762) **Inactive** (standard): Only language elements that comply with IEC 61131-3 are permitted.
Only create program instance data once[1]	**Active**: The local variables of a program are only stored once in the user memory of the unit. This setting is required for calling a program while inside a different program (Page 4762). This setting also improves the download in RUN mode, see corresponding section in the Basic Functions Function Manual. **Inactive** (standard): The local variables of a program are stored according to the task assignment in the user memory of the respective task. See Memory areas of the variable types (Page 4844).

Parameter	Description
Permit additional languages, IEC61131 3rd edition[1]	**Active**: Additional language elements can be used in accordance with IEC 61131-3 3rd edition. The corresponding keywords are locked as reserved or protected identifiers. • Nested block comments (Page 4670) • CONTINUE statement (Page 4727) • Standard-compliant system functions LOWER_BOUND and UPPER_BOUND (Page 4747) for determining the limits of a dynamic ARRAY • Additional system functions (see Basic Functions Function Manual): – FROM_BIG_ENDIAN – FROM_LITTLE_ENDIAN – IS_VALID – TO_BIG_ENDIAN – TO_LITTLE_ENDIAN **Inactive** (standard): The additional language elements cannot be used. The corresponding keywords are not locked. **Note** When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.
Permit object-oriented programming[1]	**Active**: With the ST programming language additional language elements for object-oriented programming can be used in accordance with IEC 61131-3 3rd edition. The corresponding key words are locked as reserved or protected identifiers in all programming languages. **Inactive** (standard): The keywords for object-oriented programming are not locked in all programming languages. The additional language elements cannot be used with the ST programming language. **Note** Independently of the setting the program organization units generated in ST sources using object-oriented programming (e.g. classes, methods) can be used in program sources for all programming languages. When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.
Permit forward declarations (ST)	Only for the ST programming language. Forward declarations enable you to use program organization units (POUs) before they are fully defined. Prototype declarations of program organization units are possible prior to their use, but are only required for an instance declaration of a function block or a class. **Active**: Forward declarations are permitted. **Inactive** (standard): Forward declarations are not permitted. See Forward declarations (Page 4929). Forward declarations are always permitted with the MCC and LAD/FBD programming languages. It is also possible to make a local setting on the ST source file (Page 4626). Please also refer to the description of the effectiveness of global or local compiler settings (Page 4630).

Parameter	Description
Permit single step (MCC)	Only for the MCC programming language. **Active**: An additional program code is created which enables individual program steps to be monitored. **Inactive**: Single step is not possible. This function facilitates debugging of your program. See "Tracking single steps in the program" in the MCC Programming Manual. It is also possible to make a local setting on the MCC unit. Please also refer to the description of the effectiveness of global or local compiler settings (Page 4630).
Permit trace (MCC - only for RT versions >= 4.2)	Only for the MCC programming language and for SIMOTION Kernel as of version V4.2. **Active**: An additional program code is created which enables monitoring of the program execution in program branches which are executed cyclically. **Inactive**: Trace is not possible. This function facilitates debugging of your program. See "Tracking program execution per trace" in the MCC Programming Manual. It is also possible to make a local setting on the MCC unit. Please also refer to the description of the effectiveness of global or local compiler settings (Page 4630).
Optimize execution order (LAD/FBD)	Only for LAD/FBD programming languages. **Active**: LAD/FBD networks are calculated in the optimized execution order. **Inactive**: LAD/FBD networks are calculated in the non-optimized execution order.
Default setting	Click the button to restore the default setting.
Create detailed message log during compilation of the project/devices[2]	Here, you can control the scope of the message log that will be displayed in the workbench's detail view when you call the **Save and compile changes** command in SIMOTION SCOUT. **Active**: A detailed message log is created that is similar to that for single compilation of an ST source file. **Inactive**: A compressed message log is created.
[1] Local settings also possible, see Local settings of the compiler (Page 4626). Please also refer to the description of the effectiveness of global or local compiler settings (Page 4630). [2] User-specific settings. Valid for all SIMOTION projects that the user processes.	

Note

You may have to save and compile the project for the settings to take effect.

Local compiler settings

Local settings are configured individually for each ST source file; local settings overwrite global settings.

Procedure

1. Open the Properties window for the ST source file, see Changing the properties of an ST source file (Page 4589):
   Select the ST source file in the project navigator and select the **Edit > Object properties** menu command.
2. Select the **Compiler** tab.

3. Define the settings according to the following table.
4. Confirm with **OK**.

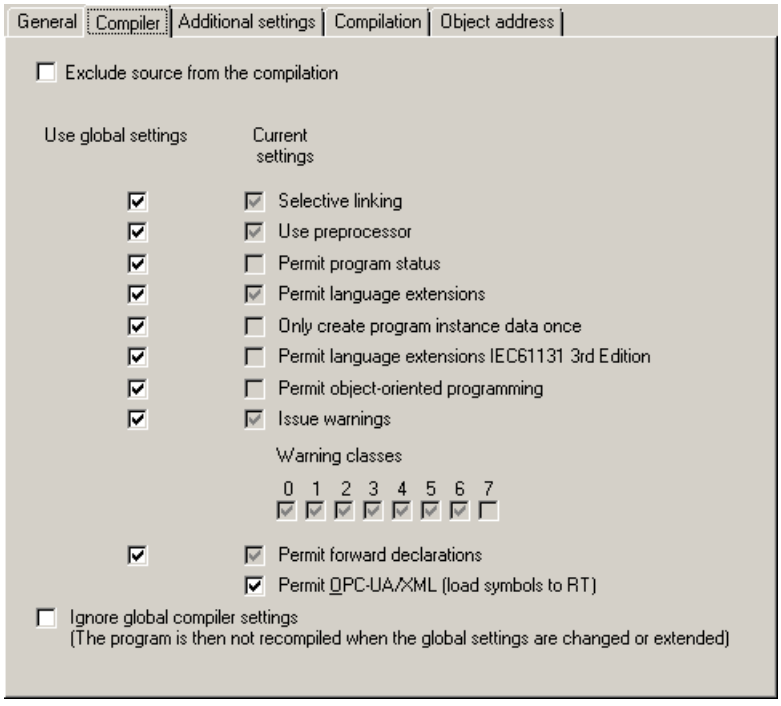


Figure 7-218 Local compiler settings for the ST source file

Parameter

Table 7-364 Parameters for the local compiler settings for the ST source file

Parameter	Description
Exclude source from the compilation	**Active**: This source is not compiled upon compilation of the project, the device or the library (e.g. Menu **Project > Save and recompile all**). The source is marked accordingly in the project navigator. Corresponding information is provided upon compilation. **Inactive** (standard): The source is compiled upon compilation of the project, the device or the library.
Use global settings	This checkbox is available for every parameter which also has a global setting. It can only be selected when the "Do not recompile the source if global compiler settings have been changed" checkbox is inactive. This is where you define whether the global settings are adopted or whether the local settings will apply. See the description under "Effectiveness of global or local compiler settings (Page 4630)". Use the second checkbox or the other checkboxes for the relevant parameters (described below) to define the local settings.
Selective linking[1]	**Active**: Unused code is removed from the executable program. **Inactive**: Unused code is retained in the executable program. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).

Parameter	Description
Use preprocessor[1]	**Active**: Preprocessor is used. **Inactive**: Preprocessor is not used. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Controlling the preprocessor (Page 4918).
Enable program status[1]	**Active**: Additional program code is generated to enable monitoring of program variables (including local variables). **Inactive**: Program status not possible. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Properties of the program status (Page 4949).
Permit language extensions[1]	**Active**: Language elements are permitted that do not comply with IEC 61131-3. • Direct bit access to variables of a bit data type (Page 4705) • Accessing the input parameter of a function block while outside the function block (Page 4756) • Calling a program while in a different program (Page 4762) **Inactive**: Only language elements that comply with IEC 61131-3 are permitted. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).
Permit additional languages, IEC61131 3rd edition[1]	**Active**: Additional language elements can be used in accordance with IEC 61131-3 3rd edition. The corresponding keywords are locked as reserved or protected identifiers. • Nested block comments (Page 4670) • CONTINUE statement (Page 4727) • Standard-compliant system functions LOWER_BOUND and UPPER_BOUND (Page 4747) for determining the limits of a dynamic ARRAY • Additional system functions (see Basic Functions Function Manual): – FROM_BIG_ENDIAN – FROM_LITTLE_ENDIAN – IS_VALID – TO_BIG_ENDIAN – TO_LITTLE_ENDIAN **Inactive**: The additional language elements cannot be used. The corresponding keywords are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

Parameter	Description
Permit object-oriented programming[1]	**Active**: Additional language elements for object-oriented programming can be used in accordance with IEC 61131-3 3rd edition. The corresponding keywords are locked as reserved or protected identifiers. **Inactive**: The additional language elements cannot be used. The corresponding keywords are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** Independently of the setting the program organization units created in other ST source files using object-oriented programming (e.g. classes, methods) can be used in this source. When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.
Only create program instance data once[1]	**Active**: The local variables of a program are only stored once in the user memory of the unit. This setting is required for calling a program while inside a different program (Page 4762). This setting also improves the download in RUN mode, see corresponding section in the Basic Functions Function Manual. **Inactive**: The local variables of a program are stored according to the task assignment in the user memory of the respective task. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Memory areas of the variable types (Page 4844). For further information, refer to the SIMOTION Basic Functions Function Manual.
Issue warnings Warning classes[1]	In addition to error messages, the compiler can issue warnings and information. You can set the scope of the warning messages to be issued. "Issue warnings" checkbox: **Active**: The compiler issues the warnings and information according to the warning class selection that follows. **Inactive**: The compiler suppresses all warnings and information concerning this unit. The checkboxes for the warning classes are hidden. **Gray background** (display only): Operating on a global setting basis, the compiler always issues warnings and information in accordance with the global warning class selection shown below (if "Use global settings" = active). "Warning classes" checkboxes (only if "Issue warnings" = active): **Active**: The compiler issues warnings and information for the selected class. **Inactive**: The compiler suppresses warnings and information for the respective class. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See also Meanings of the warning classes (Page 4633).
Permit forward declarations[1]	Forward declarations enable you to use program organization units (POUs) before they are fully defined. Prototype declarations of POUs are possible prior to their use, but only required for an instance declaration of a function block. **Active**: Forward declarations are permitted. **Inactive**: Forward declarations are not permitted. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Forward declarations (Page 4929).

Parameter	Description
Permit OPC-UA / -XML (load symbols to RT)	**Active** (standard): Symbol information for the unit variables of the ST source file is available in the SIMOTION device. This is required for: • The *_exportUnitDataSet* and *_importUnitDataSet* functions; see the SIMOTION Basic Functions Function Manual • The watch function of IT DIAG. **Inactive:** Symbol information is not created.
Ignore global compiler settings (The program is not compiled again if the global settings are changed or extended.)	**Active:** The global settings of the compiler are ineffective for all parameters. The "Use global settings" checkboxes cannot be selected and are grayed out. When changing the global compiler settings, the ST source file is not recompiled. This setting is required for libraries that have a know-how protection program (Page 4635) with a "High" level: • For all program sources in this library. • For the library itself. **Inactive** (standard): The checkboxes "Use Global Settings" can be selected for all parameters and are presented against a white background. These checkboxes specify whether the global properties are taken over for the corresponding parameters. This setting means that in the following case the ST source file is compiled even though all "Accept global settings" checkboxes are inactive: • The global settings of the compiler have been changed and the menu **Project > Save and compile changes** is selected. See the description under "Effectiveness of global or local compiler settings (Page 4630)".
[1] Global setting also possible, see Global settings of the compiler (Page 4623). Please also refer to the description of the effectiveness of global or local compiler settings (Page 4630).	

Effectiveness of global or local compiler settings

You control the effectiveness of the global and local settings of a parameter using the local compiler settings (Page 4626).

"Ignore global compiler settings" checkbox

With the "Ignore global compiler settings" checkbox, you specify whether the global properties of the compiler influence the program source.

- **Active:** The global settings of the compiler are ineffective for all parameters. The "Use global settings" checkboxes cannot be selected and are grayed out. When changing the global compiler settings, the ST source file is not recompiled.
  This setting is required for libraries that have a know-how protection program (Page 4635) with a "High" level:
  - For all program sources in this library.
  - For the library itself.
- **Inactive:** The checkboxes "Use global settings" can be selected for all parameters and are presented against a white background. These checkboxes specify whether the global properties are applied to the corresponding parameter.
  This setting means that in the following case the program source is compiled even though all "Use global settings" checkboxes are inactive:
  - The global settings of the compiler have been changed and the menu **Project > Save and compile changes** is selected.

  This causes problems for libraries that have a know-how protection program (Page 4635) with a "High" level: Online inconsistencies can also occur for projects that have been converted from an earlier version of SIMOTION SCOUT.

Checkboxes for the other parameters

Every parameter that has a global setting also has at least two checkboxes:

- "Use global settings" checkbox:
  You use this checkbox to define whether global settings will be used. It can only be selected when the "Do not recompile the source if global compiler settings have been changed" checkbox is inactive.
  The following settings are possible:
  - **Active**:
    The global settings will be used for the corresponding parameter.
    No other checkboxes may be selected for this parameter; they are grayed out and show the global setting.
  - **Inactive**:
    The global settings are ignored for the corresponding parameter. Only the local settings, which you define with the other checkboxes, are effective.
- One or more checkboxes for current (local) settings:
  The function of these checkboxes depends on the "Use global settings" checkbox:
  - If the "Use global settings" checkbox is active:
    The global settings will be used for the corresponding parameter. The checkboxes for the current settings cannot be selected. They are grayed out and show the relevant global setting.
  - If the "Use global settings" checkbox is inactive:
    The checkboxes for the current settings can be selected and are displayed on a white background. You can use them to define the local settings for the corresponding parameter. The global settings are ignored.

This behavior applies to the following compiler settings:

- Selective linking
- Use preprocessor
- Enable program status
- Permit language extensions
- Only create program instance data once
- Permit additional languages, IEC61131 3rd edition
- Permit object-oriented programming
- Issue warnings with warning classes
- Permit forward declarations (only for the ST programming language)
- Permit single step (only in MCC programming language)
- Permit trace (only for the MCC programming language and for SIMOTION Kernel as of version V4.2)

Note

You can check the current compiler options which will be effective the next time the program source is compiled.

- To do this, select the "Additional settings" tab (Page 4633) in the Properties window of the program source.

Meanings of the warning classes

The table lists the warning classes and their meanings.

Table 7-365 Meanings of the warning classes

Warning class	Meaning
0	Warnings for unreferenced or unused code sections and data
1	Warnings for hidden identifiers
2	Warnings for data type conversion, e.g. for data change
3	Warnings about set compiler options
4	Warnings about semaphores (potentially faulty functions)
5	Warnings about alarm functions
6	Warnings about constructs in libraries (unit variables declared)
7	Messages of the preprocessor

For the detailed description of the compiler error messages, specify which warning classes are assigned to the individual warnings (Page 5058) and information (Page 5065).

Display of the compiler options

You can view for a program source the following:

- The current compiler options using the global or local settings of the compiler.
- The compiler options used for the last compilation of the program source.

Requirement

The Properties window of the program source (Page 4589) is open.

Procedure

To display the current compiler options using the global or local settings of the compiler (Page 4623):

- Select the **Additional settings** tab.
  The current compiler options for the program source are displayed. They are valid for a future compilation.

To display the compiler options used for the last compilation of the program source:

- Select the **Compiler** tab.
  The following are displayed for the last compilation of the program source:
  - The version of the used compiler.
  - The used compiler options.

Meaning of the compiler options

Compiler option	Meaning
-c[2]	Do not create debug and symbol information.
-C exclude	Exclude source from the compilation.
-C lang_enable_oop	"Permit object-oriented programming" active.
-C lang_disable_oop	"Permit object-oriented programming" inactive.
-C lang_enable_v3ext	"Permit language extensions IEC61131 3rd Edition" active.
-C lang_disable_V3ext	"Permit language extensions IEC61131 3rd Edition" inactive.
-C lang_ext	"Permit language extensions"[1] active.
-C lang_iec	"Permit language extensions" inactive.
-C opcsym	"Permit OPC-UA/-XML"[1] active.
-C no_opcsym	"Permit OPC-UA/-XML" inactive.
-C opcsym	"Use preprocessor"[1] active.
-C no_preproc	"Use preprocessor" inactive.
-C prog_once	"Only create program instance data once"[1] active.
-C prog_multi	"Only create program instance data once" inactive.
-C scan_twice	"Permit forward declarations"[1] active.
-C scan_once	"Permit forward declarations" inactive.
-D *text*	Preprocessor definition (Page 4635).
-e user[2]	Only global settings act.
-e local	"Do not recompile the source if global compiler settings have been changed"[1] active. Only local settings act. No details (default): "Do not recompile the source if global compiler settings have been changed" inactive. Global settings will be augmented with local settings.
-I[2]	Accept the package settings from device or library.
-l sel	"Selective linking"[1] active.
-l no_sel	"Selective linking" inactive.
-s	"Enable program status"[1] active.
-s_off	"Enable program status" inactive.
-w no_warn	"Suppress warnings"[1] active.
-w all_warn	Display all warnings.
-w *n*_off	Warning class *n* inactive[1].
-w *n*_on	Warning class *n* active[1].
Further options	Internal options of the SIMOTION compiler.

[1] Meaning of the compiler option: See "Local compiler settings" (Page 4626).
[2] Only when the compiler is called from the command line, e.g. using scripting.

Note

The compiler options can also be specified when the compiler is called from the command line, e.g. using scripting.

Know-how protection for ST source files

You can protect ST source files from access by unauthorized third parties. Protected ST source files can only be opened or exported as plain text files by entering a password.

The SIMOTION online help provides additional information on know-how protection.

Note

If you export in XML format, the ST source files are exported in an encrypted form. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

See also

Know-how protection for libraries (Page 4895)

Making preprocessor definitions

You can make definitions for the preprocessor (see Controlling the preprocessor (Page 4918)) in the Properties dialog box of the ST source file. This enables you to control the preprocessor for ST source files with know-how protection too (see Know-how protection for ST source files (Page 4635)).

Making preprocessor definitions in the Properties dialog box

1. Open the Properties window for the ST source file
   (see Changing the properties of an ST source file (Page 4589)):
   Select the ST source file in the project navigator, followed by the **Edit > Object properties** menu command.
2. Select the **Additional settings** tab.
3. Enter the preprocessor definitions (syntax as shown in the following table).
4. Confirm with **OK**.

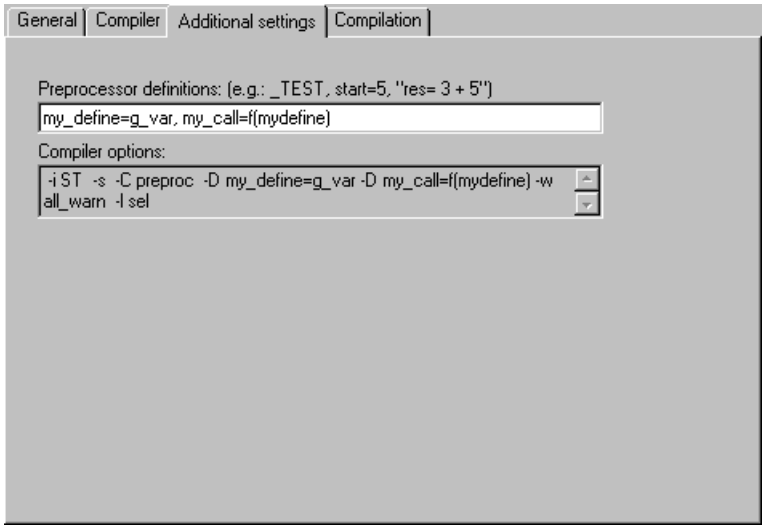


Figure 7-219 Preprocessor definitions

Table 7-366 Syntax of the preprocessor definitions

Syntax	Meaning
Identifier=text	The specified *identifier* is defined and replaced in the ST source file by the specified *text*. Permissible characters: See table footnote. If the expression contains blanks (e.g. in the text), the syntax *"Identifier=text"* must be used.
'Identifier=text'	
"Identifier=text"	
Identifier	The specified *identifier* is defined and replaced in the ST source file by blank text. Permissible characters: See table footnote.
Multiple preprocessor definitions are separated by commas: *Definition_1, Definition_2, ...* Permissible characters: • For *identifiers*: In accordance with the rules for identifiers: Series of letters (A ... Z, a ... z), digits (0 ... 9) or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper and lower case letters. • For *text*: Sequence of any characters other than \ (backslash), ' (single quote) and " (double quote). The keywords USES, USELIB and USEPACKAGE are not permitted.	

Note

Preprocessor definitions, which are made within an ST source file with pragmas, overwrite the definitions in the Properties dialog box.

Note the information relating to preprocessor statements (Page 4919).

Exporting, importing and printing an ST source file

An overview is provided here of the export, import and printing of an ST source file.

Exporting an ST source file as a text file (ASCII)

You can export an ST source file as a text file in ASCII format and then either reimport this file as an ST source file or edit it with any ASCII editor.

Procedure

To export an ST source file as an ASCII file:

1. Open the ST source file (Page 4588), entering the password if necessary (for ST source files with know-how protection (Page 4635)).
2. Make sure that the cursor is in the ST editor.
3. Select the **ST source file > Export** menu command.
4. Enter the path and file name for the ASCII file and click **Save** to confirm.

The ST source file is saved as an ASCII file; the file name is given the default extension *.st

Alternatively, you can also proceed as follows:

1. Select the ST source file in the project navigator.
2. Select **Export** from the context menu.
3. Only for ST source files with know-how protection (Page 4635) and which are not already open:
   If the user with the log-in details assigned to the ST source file has not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (for this export).
   - If required, activate the **Use login as a standard login** checkbox.
     You will be logged in with this login and can now export or open additional units to which the same login is assigned without having to re-enter the password.
4. Enter the path and file name for the ASCII file and click **Save** to confirm.

Note

Folding information (Page 4596) and bookmarks (Page 4607) are not exported.

An ST source file with know-how protection is exported without protection.

Exporting an ST source file in XML format

Follow these steps to export an ST source file in XML format:

1. Select the ST source file in the project navigator.
2. Select the context menu **Expert > Save project and export object**.
3. Specify the path for the XML export, and confirm with **OK**.

An XML file with the ST source file name and a folder with additional associated XML files are saved in the specified path.

Note

Know-how-protected ST source files can also be exported in XML format. The ST source files are exported encrypted. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

Folding information (Page 4596) and bookmarks (Page 4607) are not exported.

Importing a text file (ASCII) as an ST source file

To import an ASCII file as an ST source file:

1. Select the **PROGRAMS** folder under the appropriate SIMOTION device in the project navigator.
2. Select the menu **Insert > External source > ST source file**.
3. Select the ASCII file to be imported, and click **Open** to confirm.
   The dialog box for inserting an ST source file is displayed.
4. Enter the name of the ST source file and select the additional options (see Insert ST source file (Page 4587)).

The ASCII file is incorporated into the current project directory as an ST source file and can be opened.

Importing XML data into ST source files

Follow these steps to import XML data into an ST source file:

1. If applicable, insert a new ST source file (see Insert ST source file (Page 4587)).
2. Select the ST source file in the project navigator.
3. Select the context menu **Expert > Import object**.
4. Select the XML data to be imported.
   The imported XML data overwrites existing data in the selected ST source file. The entire project is saved and recompiled.

Alternative:

1. In the project navigator, select the **PROGRAMS** folder.
2. In the context menu, select **Import object**.
3. Select the XML data to be imported.
   A new ST source file is created, and the XML data is imported. This ST source file is assigned the name which is saved in the XML data; if a naming conflict occurs, it is automatically renamed. The entire project is saved and recompiled.

Note

Note that if the XML data to be imported was exported from an ST source file that was know-how protected: When importing the encrypted XML data, the know-how protection, including login and password, is retained.

Printing an ST source file

To print an ST source file:

1. Open the ST source file.
2. Make sure that the cursor is in the ST editor.
3. Select the menu **Project > Print**.

The program is printed with the name and date.

Using an external editor

What external editors can be used?

As an alternative to the default ST editor, you can use any other ASCII editor that supports the following function:

- External programs (for example, compiler) can be called and run on the active window.

In addition, the editor should be capable of highlighting certain text passages of the ST source file in color (syntax coloring).

Note

If you use an external editor, the dynamic ST source file menu and its entries are not available. The corresponding toolbar is also inactive.

It must be possible to start compilation of the ST source file from the external editor.

Program status (Page 4951) continues with the ST editor.

Settings for the use of an external editor

The settings are made in the SCOUT workbench:

1. Select the menu **Options > Settings**.
2. Select the **ST external editor** tab (see figure).
3. Activate the **Use external ST editor** checkbox.
4. Enter the path of the external editor:
   – Click **Browse...** and select the path and file name of the editor.

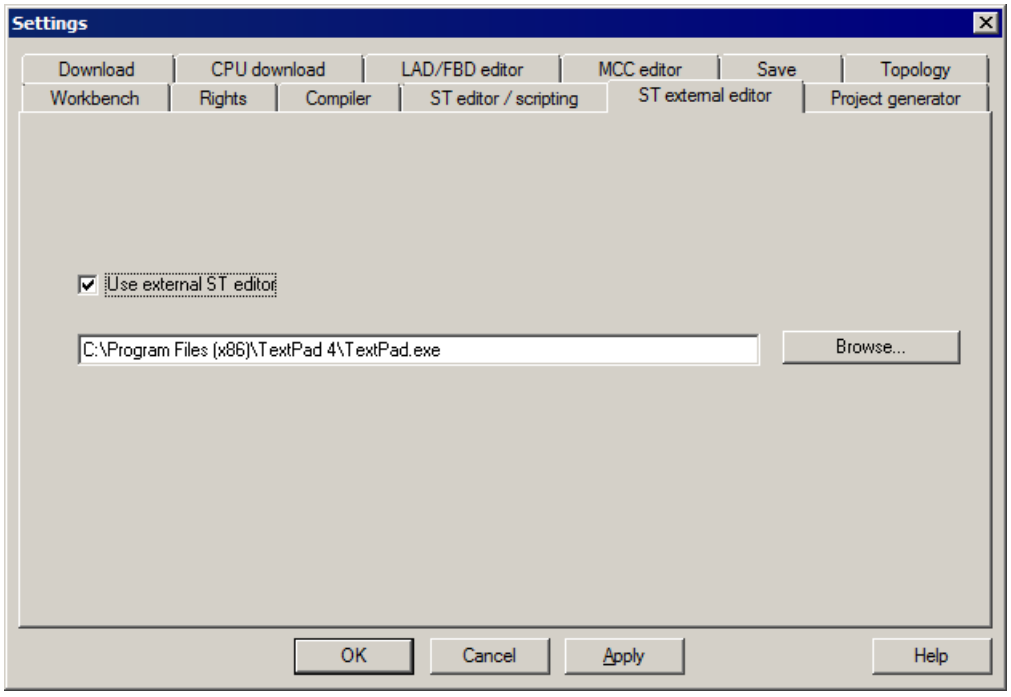


Figure 7-220 Settings for the use of an external editor

Making settings in the external editor

The following notes are of a general nature. Compare the operator instructions of the external editor.

1. Configure the path to the ST compiler in the external editor. The compiler is located in the STEP7 installation directory s7bin\u7wstcax.exe.
2. Syntax files are supplied for various editors. These enable the editor to highlight text passages in color (syntax coloring). Copy the syntax file to the relevant directory and configure the editor accordingly.

Note the following when using an external editor:

NOTICE
Data loss possible If you close a project or exit SIMOTION SCOUT before all windows of the external editor have been closed, data can be lost! Close all windows of the external editor before you close a project or exit SIMOTION SCOUT.

ST source file menus

ST source file menu

Depending on the active application/editor or the mode (ONLINE/OFFLINE), certain commands are not displayed or cannot be selected. The menu is only displayed if the ST editor is active in the working area.

You can select the following functions:

Table 7-367 ST Source File Menu

Function	Meaning/information
Close	Select this command to close the active ST source file. In the event of changes, you can decide whether you want to transfer the changed source file to the project and whether the changes should be saved in the snippet editor.
Properties	Select this command to display the properties of the active ST source file. Several tabs are provided to make local settings for this source. See: Changing the properties of an ST source file (Page 4589).
Accept and compile	Choose this command to transfer the current ST source file to the project and compile into executable code. See: Starting the compiler (Page 4622).
Execute preprocessor	As an option, the preprocessor scans an ST source file before compiling and can, for example, replace character strings in the file, which will then be taken into account during the compilation. You can specifically execute the preprocessor statements with this menu command.
Export	Select this command to export the active ST source file as text file (ASCII). See: Exporting an ST source file as a text file (ASCII) (Page 4637).
Split window	Select this command to split the active window of the ST editor into two segments horizontally, giving you two views of the same ST source file. See: Splitting the editor window (Page 4600).
Program status on/off	Select this command to start the **program status** test mode. During the program execution, you can monitor the values of the variables marked in the ST source. The following prerequisites are necessary: 1. The program must be compiled with the appropriate compiler option. 2. The project and the program must be loaded into the target system. 3. An online connection to the target system must have been established. Select the command again to close the **program status**. See: Using the program status (Page 4951).
Save variables	You can save retain, unit and global variables with this menu command. You can save these variables from the RAM/ROM memory of the target device and store them on a data medium as XML file. When these variables are restored, they can be written from the data medium to the RAM/ROM memory of the target device.
Restore variables	You can restore retain, unit and global variables from the previously exported variables with this menu command. When these variables are restored, they can be written from the data medium to the RAM/ROM memory of the target device.

ST source file context menu

The ST source file context menu is shown if you select an ST source file in the project navigator and then right-click it.

Depending on the active application/editor or the mode (ONLINE/OFFLINE), certain commands are not displayed or cannot be selected.

You can select the following functions:

Table 7-368 ST source file context menu

Function		Meaning/information
Open		Select this command to open the selected ST source file. See: Opening an existing ST source file (Page 4588).
Cut		The selected ST source files are deleted and saved on the clipboard.
Copy		The selected ST source files are copied to the clipboard.
Paste		The contents of the clipboard are inserted in the selected folder.
Delete		The selected ST source file is deleted, including all the data.
Rename		Select this command in order to change the name of the selected ST source file. Please note that with name changes, it is not possible to change the referencing to this name and that the new name must comply with the Rules for identifiers (Page 4656).
Save variables		You can save retain, unit and global variables with this menu command. You can save these variables from the RAM/ROM memory of the target device and store them on a data medium as XML file. When these variables are restored, they can be written from the data medium to the RAM/ROM memory of the target device.
Restore variables		You can restore retain, unit and global variables from the previously exported variables with this menu command. When these variables are restored, they can be written from the data medium to the RAM/ROM memory of the target device.
Expert		
	Import object	Select this command to import XML data to the selected ST source file from an ST source file which you have previously exported to another project. The existing data in the ST source file being imported is overwritten. See: Importing XML data into an ST source file (Page 4638).
	Save project and export object	Select this command to export the selected ST source file in XML format. You can import the exported data into other projects. See: Exporting an ST source file in XML format (Page 4637).
Accept and compile		Choose this command to transfer the current ST source file to the project and compile into executable code. See: Starting the compiler (Page 4622).
Execute preprocessor		As an option, the preprocessor scans an ST source file before compiling and can, for example, replace character strings in the file, which will then be taken into account during the compilation. You can specifically execute the preprocessor statements with this menu command.

Function	Meaning/information
Program status on/off	Select this command to start the **program status** test mode. During the program execution, you can monitor the values of the variables marked in the ST source file. The following requirements are necessary: 1. The program must be compiled with the appropriate compiler option. 2. The project and the program must be loaded into the target system. 3. An online connection to the target system must have been established. Select the command again to close the **program status**. See: Using the program status (Page 4951).
Export	Select this command to export the selected ST source file as a text file (ASCII). See: Exporting an ST source file as a text file (ASCII) (Page 4637).
Know-how protection	
Set	Select this command to protect the selected ST source file from unauthorized access by third parties. Protected ST source files can only be opened or exported as plain text files by entering a password. See: Know-how protection for ST source files (Page 4635).
Delete	Select this command to cancel the know-how protection for the selected source file permanently. The password needs to be entered in order to do this. See: Know-how protection for ST source files (Page 4635).
Display reference data	An error-free compilation is required for a correct, consistent display of the reference data. If required, compile the project, the CPU, the program or the library first. See: Reference data (Page 4910), creating a cross-reference list (Page 4910).
Cross references	The cross-reference list of the selected ST source file is generated and displayed. The cross-reference list contains the declaration and uses all the identifiers for the selected ST source file. See: Content of the cross-reference list (Page 4911).
Program structure	The program structure of the selected ST source file is generated and displayed. The program structure contains all the subprogram calls and their nesting within the selected ST source file. See: Content of the program structure (Page 4914).
Code attributes	The code attributes of the selected ST source file are generated and displayed. The code attributes contain information about the storage requirements of various data areas of the selected ST source file. See: Code attribute contents (Page 4916).
Print	Select this command to print the selected ST source file. You can choose whether you wish to print the text of the ST source file and/or their properties.
Print preview	Choose this command to generate a preview of the expected print output.
Properties	Select this command to display the properties of the selected ST source file. Several tabs are provided to make local settings for this source file. See: Changing the properties of an ST source file (Page 4589).

7.2.3.4 Creating a sample program

In this section, we create a short program to illustrate the steps involved, including starting and testing. Testing is described in Program test (Page 4934).

Function

The *Flash* program sets a bit in an output byte of your target system and rotates it within this byte. This causes each bit of the output byte to be set and reset in succession. After the last bit of the byte, the first bit is to be set again. You can observe the result of the program at the outputs of your target system.

Requirements

To create the sample program, you need

- A SIMOTION project and
- A SIMOTION device (e.g. SIMOTION C240) within the project whose output is configured at address 62.

Opening or creating a project

Projects contain all the information about the hardware and configuration. This includes the programs you use to control the hardware.

Proceed as follows

If a project does not yet exist, proceed as follows:

1. Select **Project** in the menu bar.
2. Select **New** or **Open**.
3. Specify a name for a new project, and click **OK** to confirm.

For details, see the online help.

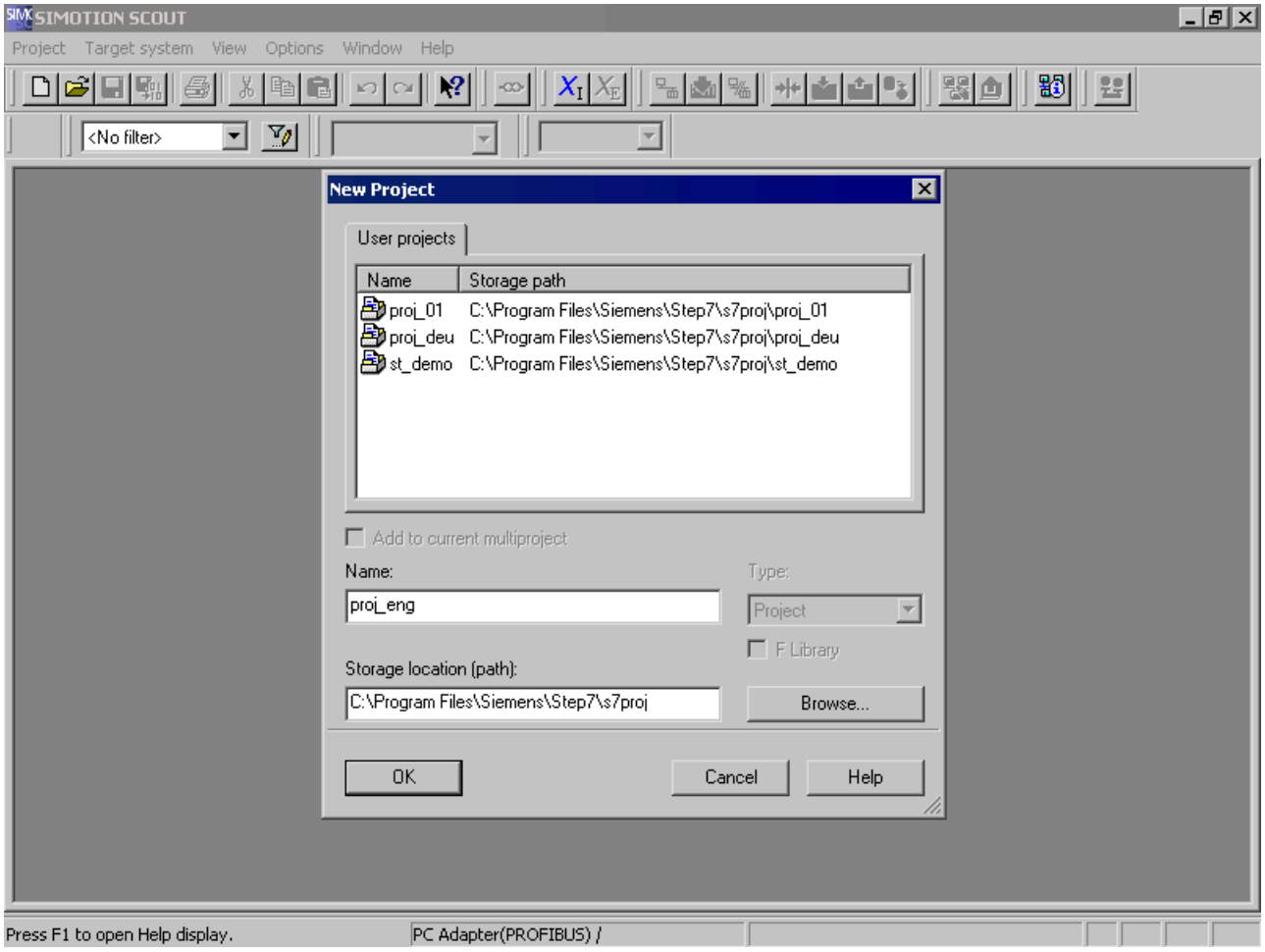


Figure 7-221 Creating a new project

Making the hardware known

The steps are as follows:

1. Create and configure a new SIMOTION device (e.g. C240 V4.2).
2. Configure an output in HW Config at Address 62.

For more details on steps 1 and 2, refer to the online help.

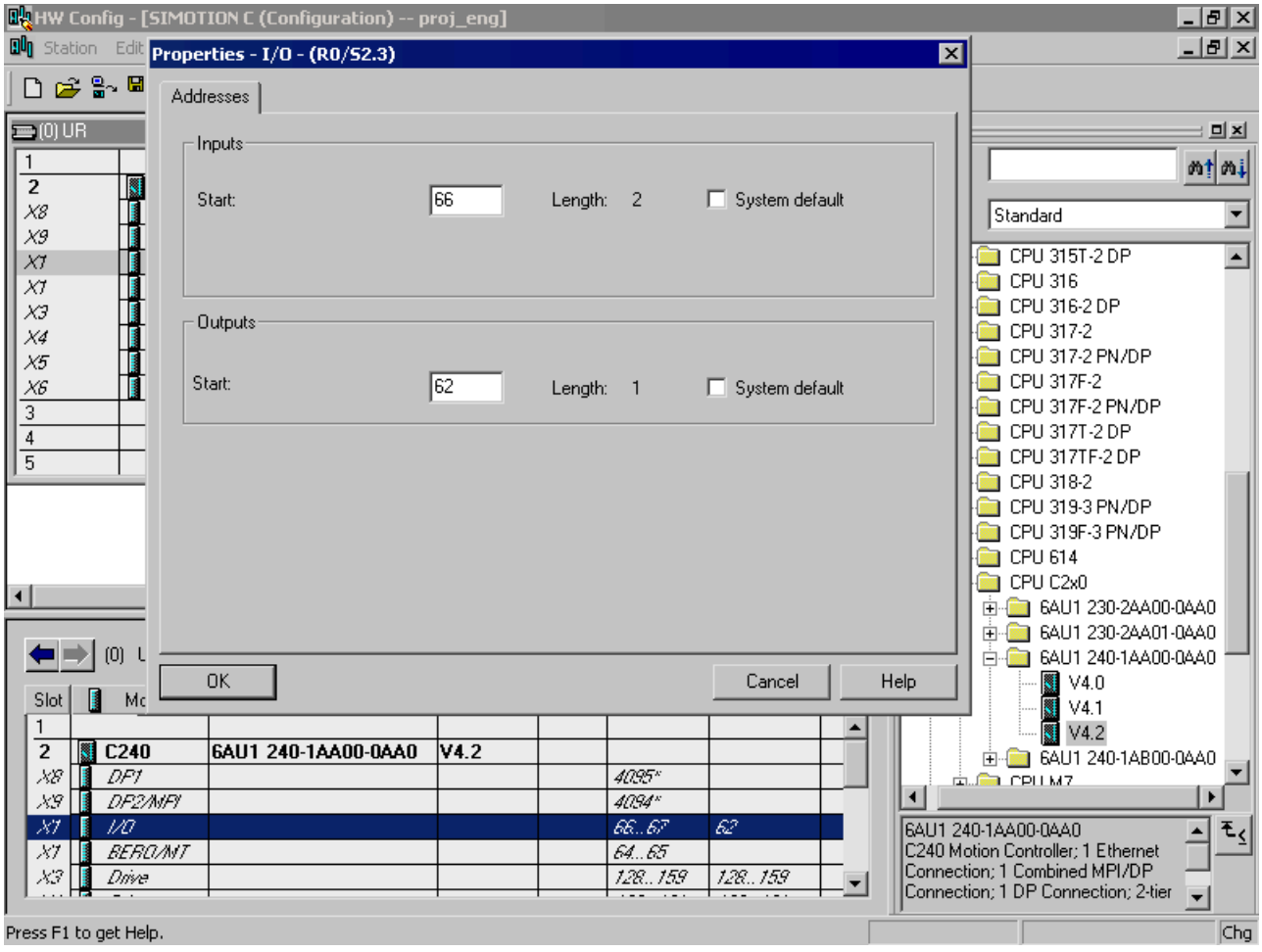


Figure 7-222 Change in HW Config

Entering source text with the ST editor

Proceed as follows

1. In the project navigator, open the tree for your SIMOTION device (programs are assigned to the SIMOTION device on which they are to run).
2. Select the **PROGRAMS** folder and choose **Insert > Program > ST source file**.
3. Enter a name for the ST source file consisting of up to 128 characters (see figure), e.g. **ST_1**, and click **OK** to confirm the entries.
   The ST editor appears in the working area. The ST source file **ST_1** is inserted in the navigator.
4. Enter the source text from Source text of the sample program (Page 4648), preferably with indented lines. To do this, press the TAB key.
   The features of the ST editor are described in Working with the ST editor (Page 4591); the structure of an ST source file is described in detail in Structure of the ST source file (Page 4668) and in Source file sections (Page 4806).

5. Use comments as often as possible. Enter your comment after the **//** characters if the comment fits on one line of text. If the comment extends across several lines, insert it between character pairs (* and *).
6. Save the complete project with **Project > Save**.

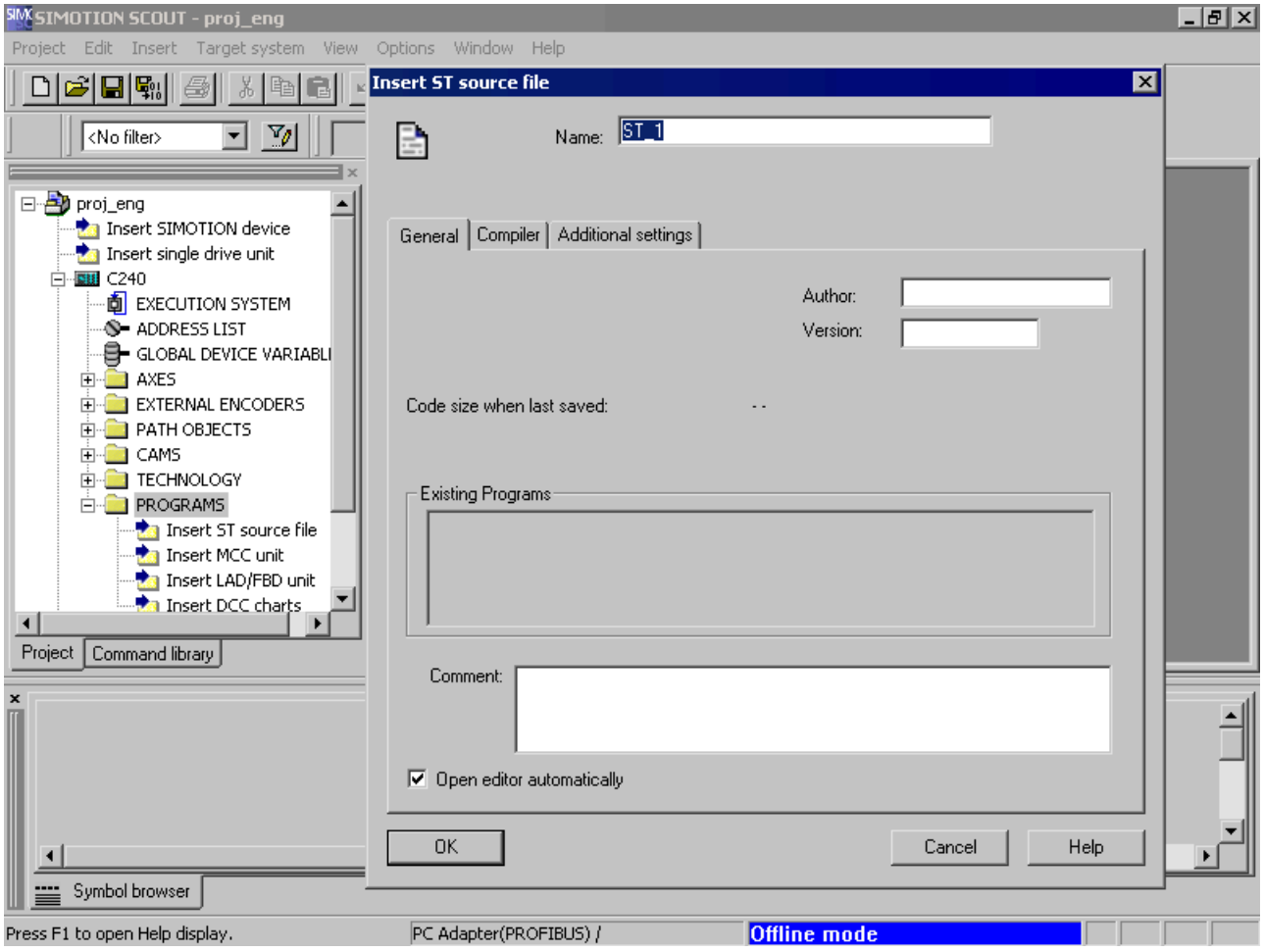


Figure 7-223 Naming the ST source file

Functions of the editor

In addition to simple text input, the ST editor provides the following advanced/convenience functions for documenting the functionality of your source text:

- Standard Windows user features (for example, Undo with Ctrl+Z or Redo with Ctrl+Y)
- Syntax coloring (different colors for different language elements)
- Source file printout in an appropriate layout with page number, source file name and printing date
- Export/import of the source file
- Source file archiving (via the project)

A detailed description of the functions is contained in Working with the ST editor (Page 4591) and in Making settings for the compiler (Page 4623).

Source text of the sample program

The table shows the source code of the sample program. You need to enter it in the same way to create executable code.

Table 7-369 Flash sample program

```
INTERFACE
    VAR_GLOBAL
        counterVar : INT  := 1; // counter variable
        outputVar  : BYTE := 1; // auxiliary tag
    END_VAR
    PROGRAM Flash;
END_INTERFACE

IMPLEMENTATION
    PROGRAM Flash
        IF counterVar >= 500 THEN // in every 500th pass
            %QB62 := outputVar;   // set output byte
            outputVar := ROL (in := outputVar, n := 1);
            (* // rotate bit in byte
              one digit to the left*)
            counterVar := 0; // reset counter
        END_IF;
        counterVar := counterVar + 1; // increment counter
    END_PROGRAM
END_IMPLEMENTATION
```

Compiling a sample program

Before you can run or test your program, you must compile it into executable machine code. The compiler performs this task.

Starting the compiler

Before you can run or test your program, you must compile it into executable machine code. The ST compiler performs this task.

Start the compiler as follows:

1. Click in the window with the ST editor to display the **ST source file** menu. This menu is a dynamic menu and is only displayed if the window of the ST editor is active.
2. Start the compiler by selecting the **ST source file > Accept and compile** menu command.

Correcting errors

The compiler checks the syntax of the ST source file. The **Compile/check output** tab of the detail view displays the successful compilation of the source text or compiler errors. The error details include: Name of the ST source file, the line number where the error occurred, the error number and an error description.

Proceed as follows to correct an error in the sample program:

1. Double-click the error message. The cursor is placed at the relevant line in the ST source file. See Example for error messages (Page 4649).
2. Start debugging the first error.
3. Start the compilation operation again.
4. Repeat the entire operation until no more errors are displayed (**0 errors**).

After a successful compilation, you will have created an application program with the name **flash**. This program is displayed in the project navigator below the **ST_1** program source file.

Example of error messages

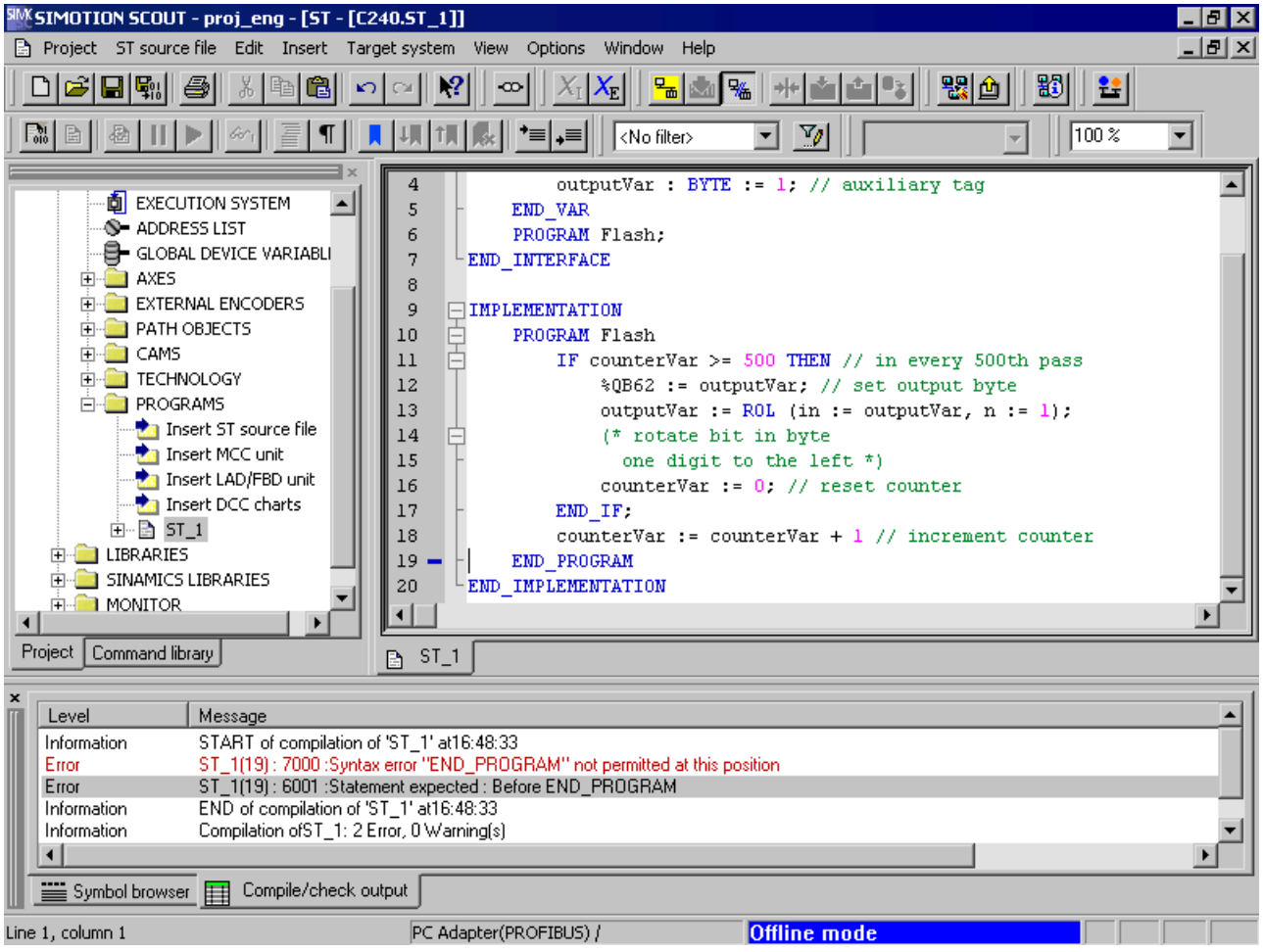


Figure 7-224 Error messages during ST source file compilation

The figure shows an example of compiling the ST source file ST_1 (see Source text of the sample program (Page 4648), in which the following change has been made: The semicolon is missing in the statement "counterVar := counterVar + 1" at the end of line 18.

The compiler does not detect the error until Line19, because it continues with the compilation after the missing semicolon.

Once the missing semicolon is added, the ST source file is compiled without errors.

A detailed list of all compiler error messages can be found in Compiler error messages and their correction (Page 5040).

Running the sample program

Before you can run the program, you must assign it to an execution level or task. When you have done this, you can establish the connection to the target system, download the program to the target system and then start it.

Assigning a sample program to an execution level

The execution levels specify the order in which the programs run. Each execution level contains one or more tasks to which you can assign programs.

The assignment of a program to a task can only be performed after compilation and before the program is loaded onto the target system.

Assign the sample program to the *BackgroundTask*. The *BackgroundTask* is provided for the programming of cyclic sequences without a fixed time frame. It is executed cyclically in the round robin execution level, which means it will be automatically restarted on completion.

How to assign the sample program to the *BackgroundTask*:

1. When you double-click the **Execution system** element in the project navigator, the window containing the execution system and the program assignment appears in the working area.
2. Click **BackgroundTask** to select it for the program assignment.
   The program assignment on the left side of the window shows you all the compiled programs that can be assigned to tasks.
3. In the **Programs** list, click sample program **ST_1.flash**. Then, click the **>>** button to assign the program to the BackgroundTask.
   The result is shown in the following figure. The program **ST_1.flash** is displayed in the **Programs used** list box.

For more information on the execution system and assignment of programs to tasks, see *SIMOTION Motion Control Basic Functions* Function Description.

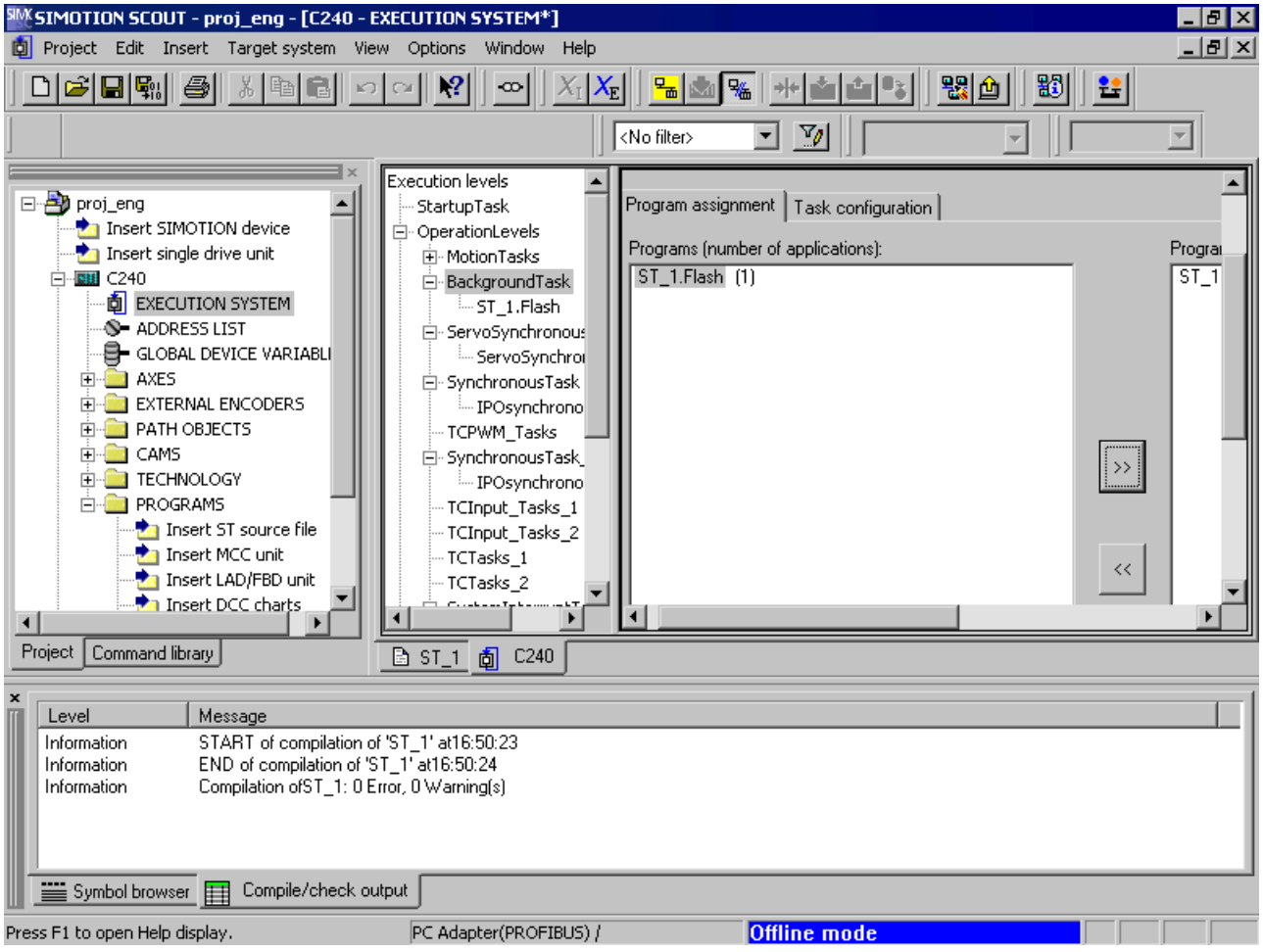


Figure 7-225 Assigning the sample program to the BackgroundTask

Establishing a connection to the target system

Before a connection to the target system can be set up, the PC interface card must be configured and connected to the target system.

Proceed as follows to connect to the target system:

1. Select the **Project > Connect to selected target devices** menu command.
   The **Diagnostics overview** tab is opened in the detail view. The diagnostics overview shows you the operating state, memory allocation and CPU utilization for the device you are connected to. You can see at the lower right edge of the screen that you are connected to the target system.

Note

For more detailed information, refer to the SIMOTION SCOUT Configuration Manual and SIMOTION SCOUT online help.

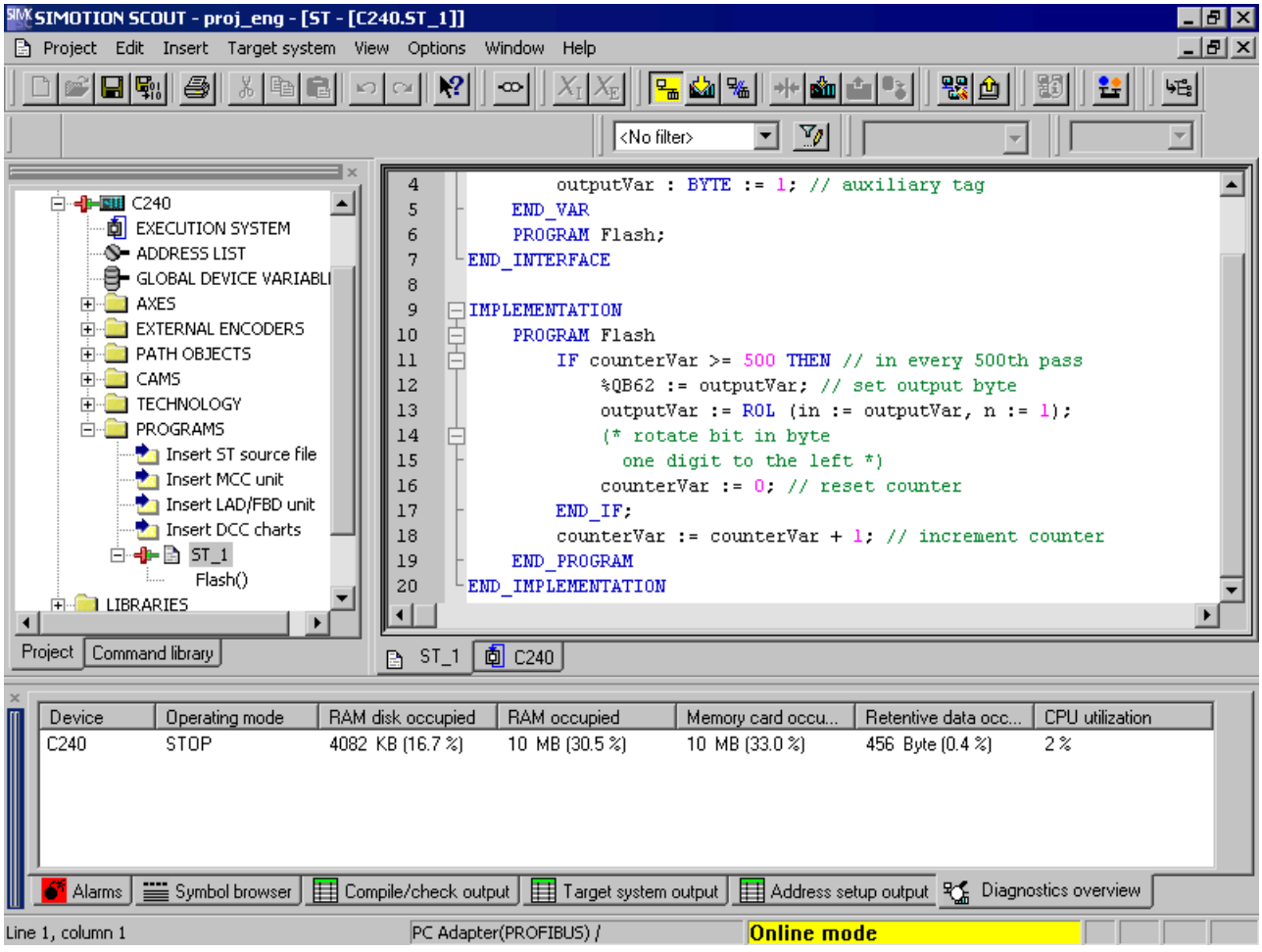


Figure 7-226 Establishing a connection to the target system

Downloading the sample program to the target system

Proceed as follows to download the sample program to the target system:

1. Switch the target system to **STOP**.
2. Select the **Target system > Download > Download project to target system** menu command.
3. Confirm all further queries.

The Target system output window in the detail view opens and displays the result of the download.

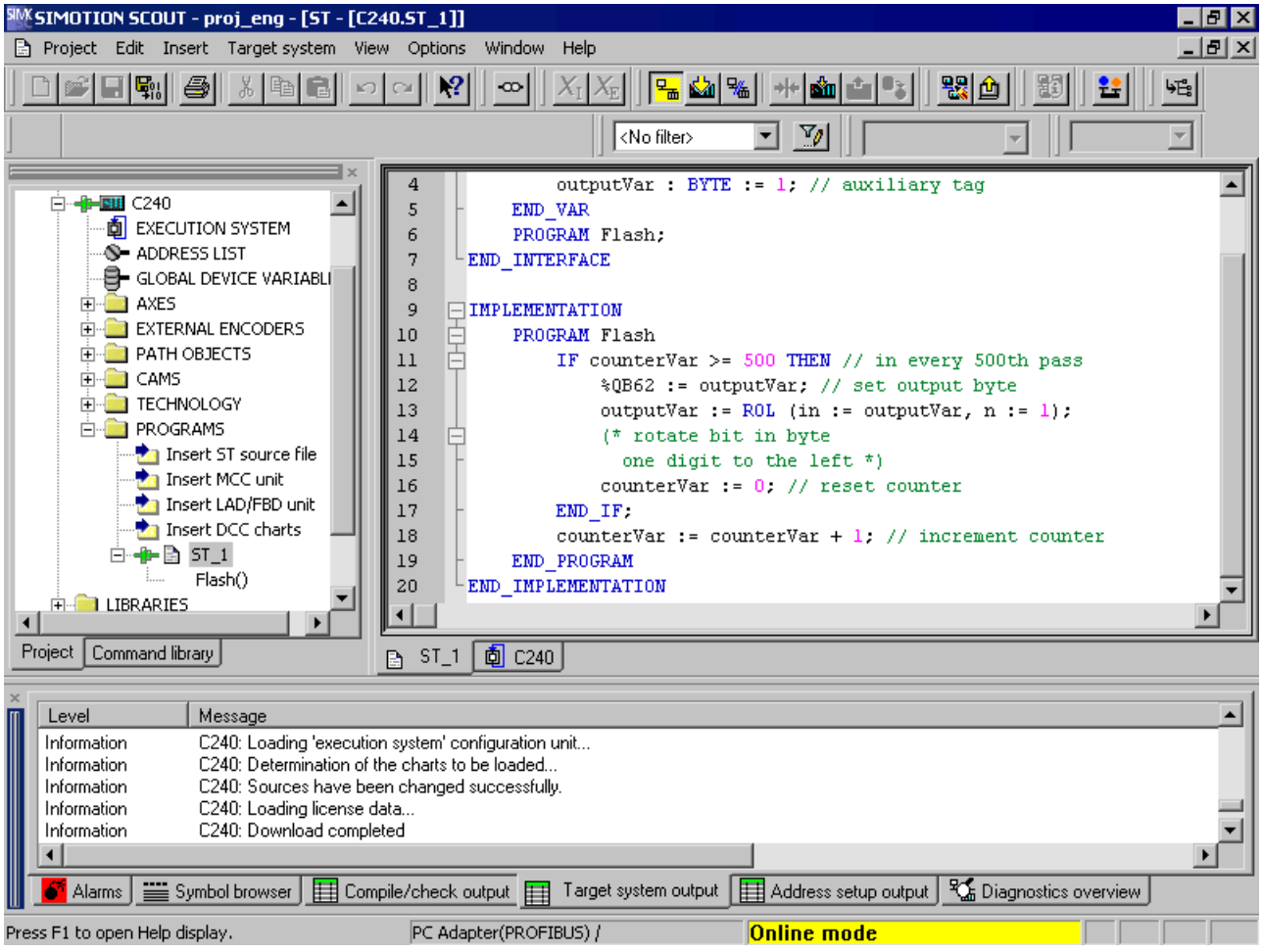


Figure 7-227 Downloading the sample program to the target system

Starting and testing the sample program

Starting sample program

Proceed as follows to start the sample program:

- Switch your target system to RUN (see hardware description).

The lamps flash in sequence at the outputs of your target system.

Testing a sample program

See Program test (Page 4934).

7.2.4 ST Fundamentals

This section describes the language resources available in ST and how to use them. Please note that functions, function blocks and the task control system are described in the following chapters. For a complete formal language description containing all the syntax diagrams, see Appendix Rules (Page 4988).

7.2.4.1 Language description resources

Syntax diagrams are used as a basis for the language description in the following sections of the manual. They provide you with an invaluable insight into the syntactic (i.e. grammatical) structure of ST.

Syntax diagram

The syntax diagram is a graphical representation of the language structure. The structure is described by a sequence of rules. A rule can build on existing rules.

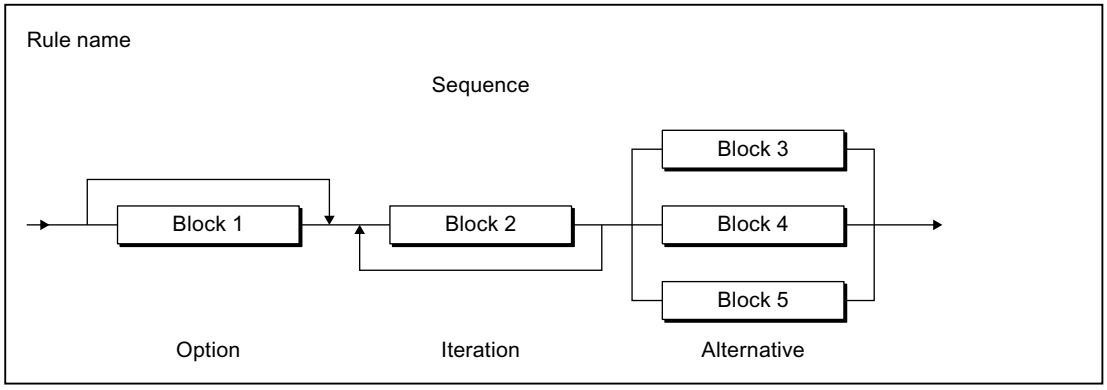


Figure 7-228 Syntax diagram

The syntax diagram in the previous figure is read from left to right. The following rule structures must be observed:

- Sequence: Sequence of blocks
- Option: Statement(s) that can be skipped
- Iteration: Repetition of one or more statements
- Alternative: Branch

Blocks in syntax diagrams

A block is a basic element or an element that is itself composed of blocks. The figure shows the symbol types used to represent the blocks:

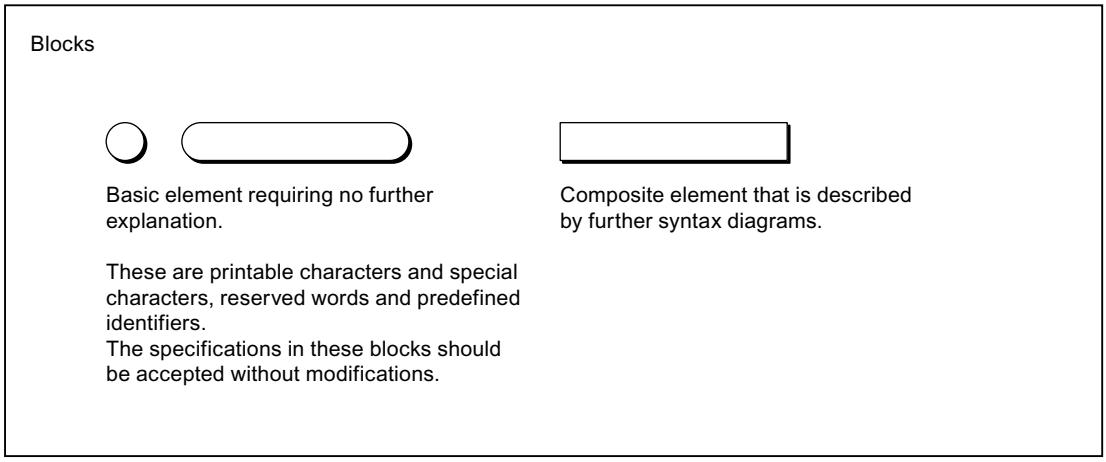


Figure 7-229 Blocks

Formatted and unformatted rules must be observed when entering source text, i.e. when converting the blocks or elements of a syntax diagram into source text, see Help for the language description (Page 4973).

Meaning of the rules (semantics)

Only the formal structure of the language can be represented in the rules. The meaning (i.e. semantics) is not always apparent. For this reason, additional information is written beside the rules if the meaning is critical. Examples include:

- Where elements of the same kind have a different meaning, an additional name is appended. For example, an addition is specified in the *date* rule for every *decimal digit string* element – either *year, month* or *day*, see Literals (Page 4989). The name indicates the usage.
- Important restrictions are noted next to the rules. For example, in the *integer* rule for - (minus), it is noted that the minus can appear only in front of decimal digit strings of data types SINT, INT, and DINT, see Literals (Page 4989).

7.2.4.2 Basic elements of the language

The basic elements of the ST language include the ST character set, reserved identifiers constructed from the ST character set (e.g. language commands), self-defined identifiers and numbers.

The ST character set and the reserved identifiers are basic elements (terminals) as they are described verbally and not by another rule. Self-defined identifiers and numbers are not terminals as they are described by other rules.

In the syntax diagrams, terminals are represented by circles or oval symbols, while composite elements are represented by rectangles, see Blocks in syntax diagrams (Page 4655). Below is a selection of the main terminals; for a complete overview, refer to Basic elements (terminals) (Page 4975).

ST character set

ST uses the following **letters and digits** from the ASCII character set:

- The lower and upper case letters from A to Z
- The Arabic digits from 0 to 9

Letters and digits are the most commonly used characters. For example, identifiers (see Identifiers in ST (Page 4656)) consist of a combination of letters, digits and the underscore. The underscore is one of the special characters.

Special characters have a fixed meaning in ST, see Basic elements (terminals) (Page 4975).

Identifiers in ST

Identifiers are names in ST. These names can be defined by the system, such as language commands. However, the names can also be user-defined, for example, for a constant, variable or function.

Rules for identifiers

Valid identifiers

Identifiers are made up of letters (A ... Z, a ... z), numbers (0 ... 9) or single underscores (_) in any order, whereby the first character must be a letter or underscore.

No distinction is made between upper and lower case letters (e.g. Anna and AnNa are considered to be identical by the compiler).

An identifier can by represented formally by the following syntax diagram:

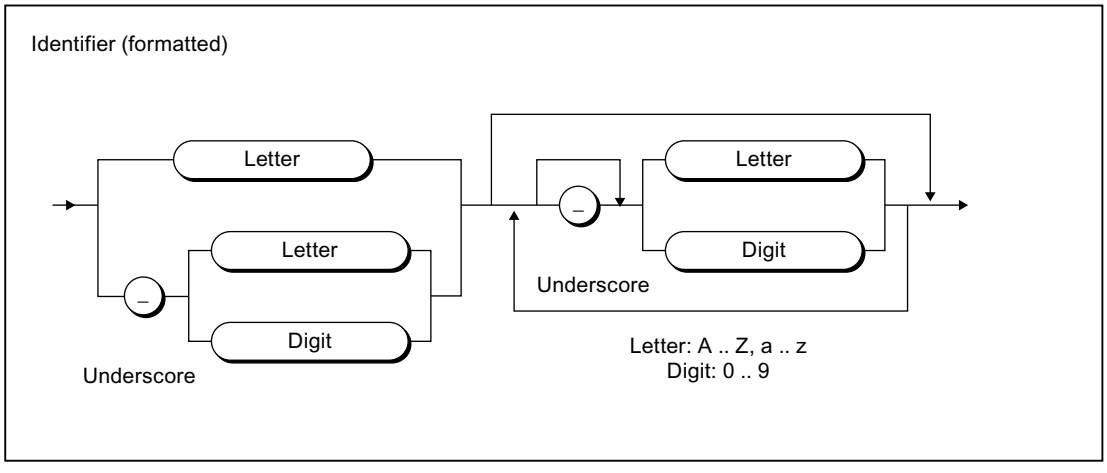


Figure 7-230 Syntax: Identifier

According to this syntax diagram, the first character of an identifier must be a letter or an underscore. An underscore must be followed by a letter or number, i.e. more than one underscore in succession is not allowed. This can be followed by any number or sequence of underscores, letters or numbers. The only exception here again is that two underscores may not appear together.

Permissible identifiers, which you can define (user-defined identifiers)

Identifiers which you define yourself, e.g. in variables declarations, data type declarations, and function names, must not be the same as reserved identifiers (Page 4657). When assigning a name, it is best to choose a unique, meaningful name that contributes to the clarity of the program.

Examples of identifiers

Examples of valid identifiers

The following names are valid identifiers:

```
x           y12         sum         temperature    P_CONTROLLER
name        area        myFB        table
```

Examples of invalid identifiers

The following names are **not** valid identifiers:

Invalid identifier	Reason
`4ter`	The first character must be a letter or underscore.
`*#AB`	Special characters (except underscores) are not permitted.
`RR__20`	Two underscores in succession are not permitted.
`S value`	Blank spaces are not permitted as they are special characters.
`Array`	ARRAY, REAL, and _sizeOf are valid identifiers from a technical point of view, but they are also reserved identifiers (Page 4657), i.e. they may only be used as predefined. This means you cannot use this name for your own purposes, for example, for a variable.
`REAL`	
`_sizeOf`	

Reserved identifiers

Reserved identifiers may only be used as predefined. You may not declare a variable or data type, for example, with the name of a reserved identifier.

There is no distinction between upper and lower case notation.

- Protected identifiers in the ST programming language (Page 4658)
- Reserved identifiers in the ST programming language (Page 4663)
- Other reserved identifiers (Page 4664), such as standard functions, system functions, system variables

A list of all identifiers with a predefined meaning can be found in the SIMOTION Basic Functions Function Manual.

Protected identifiers in the ST programming language

The protected identifiers of the ST language are listed in the table. They can only be used as predefined. A brief explanation can be found in the Appendix, under Reserved words (Page 4979). The associated syntax diagrams (Page 4654) can be found in the Appendix, under Rules (Page 4988).

Table 7-370 Protected identifiers in ST programming language

A	
ABS	ANYTYPE_TO_LITTLEBYTEARRAY
ABSTRACT [2]	ARRAY
ACOS	AS
AND	ASIN
ANYOBJECT	AT
ANYOBJECT_TO_OBJECT	ATAN
ANYTYPE_TO_BIGBYTEARRAY	
B	
BIGBYTEARRAY_TO_ANYTYPE	BY
BOOL	BYTE
BOOL_TO_BYTE	BYTE_TO_BOOL
BOOL_TO_DWORD	BYTE_TO_DINT
BOOL_TO_WORD	BYTE_TO_DWORD
BOOL_VALUE_TO_DINT	BYTE_TO_INT
BOOL_VALUE_TO_INT	BYTE_TO_SINT
BOOL_VALUE_TO_LREAL	BYTE_TO_UDINT
BOOL_VALUE_TO_REAL	BYTE_TO_UINT
BOOL_VALUE_TO_SINT	BYTE_TO_USINT
BOOL_VALUE_TO_UDINT	BYTE_TO_WORD
BOOL_VALUE_TO_UINT	BYTE_VALUE_TO_LREAL
BOOL_VALUE_TO_USINT	BYTE_VALUE_TO_REAL
C	
CASE	CTD_DINT
CLASS [2]	CTD_UDINT
CONCAT	CTU
CONCAT_DATE_TOD	CTU_DINT
CONSTANT	CTU_UDINT
CONTINUE [1]	CTUD
COS	CTUD_DINT
CTD	CTUD_UDINT
D	

DATE	DO
DATE_AND_TIME	DT
DATE_AND_TIME_TO_DATE	DT_TO_DATE
DATE_AND_TIME_TO_TIME_OF_DAY	DT_TO_TOD
DELETE	DWORD
DINT	DWORD_TO_BOOL
DINT_TO_BYTE	DWORD_TO_BYTE
DINT_TO_DWORD	DWORD_TO_DINT
DINT_TO_INT	DWORD_TO_INT
DINT_TO_LREAL	DWORD_TO_REAL
DINT_TO_REAL	DWORD_TO_SINT
DINT_TO_SINT	DWORD_TO_UDINT
DINT_TO_STRING	DWORD_TO_UINT
DINT_TO_UDINT	DWORD_TO_USINT
DINT_TO_UINT	DWORD_TO_WORD
DINT_TO_USINT	DWORD_VALUE_TO_LREAL
DINT_TO_WORD	DWORD_VALUE_TO_REAL
DINT_VALUE_TO_BOOL	
E	
ELSE	END_PROGRAM
ELSIF	END_REPEAT
END_CASE	END_STRUCT
END_CLASS [2]	END_TYPE
END_EXPRESSION	END_VAR
END_FOR	END_WAITFORCONDITION
END_FUNCTION	END_WHILE
END_FUNCTION_BLOCK	ENUM_TO_DINT
END_IF	EXIT
END_IMPLEMENTATION	EXP
END_INTERFACE	EXPD
END_LABEL	EXPRESSION
END_METHOD [2]	EXPT
END_NAMESPACE [2]	EXTENDS [2]
F	
F_TRIG	FROM_BIG_ENDIAN [1]
FALSE	FROM_LITTLE_ENDIAN [1]
FINAL [2]	FUNCTION
FIND	FUNCTION_BLOCK
FOR	
G	
GOTO	
I	

IF	INT_TO_SINT
IMPLEMENTATION	INT_TO_TIME
IMPLEMENTS [2]	INT_TO_UDINT
INSERT	INT_TO_UINT
INT	INT_TO_USINT
INT_TO_BYTE	INT_TO_WORD
INT_TO_DINT	INT_VALUE_TO_BOOL
INT_TO_DWORD	INTERFACE
INT_TO_LREAL	INTERNAL [2]
INT_TO_REAL	IS_VALID [1]
L	
LABEL	LREAL_TO_REAL
LEFT	LREAL_TO_SINT
LEN	LREAL_TO_STRING
LIMIT	LREAL_TO_UDINT
LITTLEBYTEARRAY_TO_ANYTYPE	LREAL_TO_UINT
LN	LREAL_TO_USINT
LOG	LREAL_VALUE_TO_BOOL
LOWER_BOUND [1]	LREAL_VALUE_TO_BYTE
LREAL	LREAL_VALUE_TO_DWORD
LREAL_TO_DINT	LREAL_VALUE_TO_WORD
LREAL_TO_INT	
M	
MAX	MIN
METHOD [2]	MOD
MID	MUX
N	
NAMESPACE [2]	NULL [2]
NOT	
O	
OF	OVERLAP
OR	OVERRIDE [2]
P	
PRIVATE [2]	PROTECTED [2]
PROGRAM	PUBLIC [2]
R	

R_TRIG	REAL_VALUE_TO_DWORD
REAL	REAL_VALUE_TO_WORD
REAL_TO_DINT	REF [2]
REAL_TO_DWORD	REF_TO [2]
REAL_TO_INT	REPEAT
REAL_TO_LREAL	REPLACE
REAL_TO_SINT	RETAIN
REAL_TO_STRING	RETURN
REAL_TO_TIME	RIGHT
REAL_TO_UDINT	ROL
REAL_TO_UINT	ROR
REAL_TO_USINT	RS
REAL_VALUE_TO_BOOL	RTC
REAL_VALUE_TO_BYTE	
S	
SEL	SINT_TO_WORD
SHL	SINT_VALUE_TO_BOOL
SHR	SQRT
SIN	SR
SINT	STRING
SINT_TO_BYTE	STRING_TO_DINT
SINT_TO_DINT	STRING_TO_LREAL
SINT_TO_DWORD	STRING_TO_REAL
SINT_TO_INT	STRING_TO_UDINT
SINT_TO_LREAL	STRUCT
SINT_TO_REAL	StructAlarmId
SINT_TO_UDINT	STRUCTALARMID_TO_DINT
SINT_TO_UINT	StructTaskId
SINT_TO_USINT	SUPER [2]
T	
TAN	TOF
THEN	TON
THIS [2]	TO_BIG_ENDIAN [1]
TIME	TO_LITTLE_ENDIAN [1]
TIME_OF_DAY	TP
TIME_TO_INT	TRUE
TIME_TO_REAL	TRUNC
TO	TYPE
TOD	
U	

UDINT	UINT_TO_USINT
UDINT_TO_BYTE	UINT_TO_WORD
UDINT_TO_DINT	UINT_VALUE_TO_BOOL
UDINT_TO_DWORD	UNIT
UDINT_TO_INT	UNTIL
UDINT_TO_LREAL	UPPER_BOUND [1]
UDINT_TO_REAL	USELIB
UDINT_TO_SINT	USEPACKAGE
UDINT_TO_STRING	USES
UDINT_TO_UINT	USING [2]
UDINT_TO_USINT	USINT
UDINT_TO_WORD	USINT_TO_BYTE
UDINT_VALUE_TO_BOOL	USINT_TO_DINT
UINT	USINT_TO_DWORD
UINT_TO_BYTE	USINT_TO_INT
UINT_TO_DINT	USINT_TO_LREAL
UINT_TO_DWORD	USINT_TO_REAL
UINT_TO_INT	USINT_TO_SINT
UINT_TO_LREAL	USINT_TO_UDINT
UINT_TO_REAL	USINT_TO_UINT
UINT_TO_SINT	USINT_TO_WORD
UINT_TO_UDINT	USINT_VALUE_TO_BOOL
V	
VAR	VAR_OUTPUT
VAR_GLOBAL	VAR_TEMP
VAR_IN_OUT	VOID
VAR_INPUT	
W	
WAITFORCONDITION	WORD_TO_INT
WHILE	WORD_TO_SINT
WITH	WORD_TO_UDINT
WORD	WORD_TO_UINT
WORD_TO_BOOL	WORD_TO_USINT
WORD_TO_BYTE	WORD_VALUE_TO_LREAL
WORD_TO_DINT	WORD_VALUE_TO_REAL
WORD_TO_DWORD	
X	
XOR	

1 Only with active compiler option "Permit language extensions, IEC61131 3rd edition".

2 Only with active compiler option "Permit object-oriented programming".

Reserved identifiers in the ST programming language

The table contains additional reserved identifiers that are reserved for future expansions to the ST programming language.

Table 7-371 Reserved identifiers in the ST programming language

A	
ACTION ADD ADD_DT_TIME	ADD_TIME ADD_TOD_TIME
B	
BCD_TO_BYTE BCD_TO_DINT BCD_TO_DWORD BCD_TO_INT	BCD_TO_LWORD BCD_TO_SINT BCD_TO_WORD BYTE_TO_BCD
C	
CONFIGURATION CTD_LINT CTD_ULINT CTU_LINT	CTU_ULINT CTUD_LINT CTUD_ULINT
D	
DINT_TO_BCD DIV	DIVTIME DWORD_TO_BCD
E	
EN END_ACTION END_CONFIGURATION END_RESOURCE	END_STEP END_TRANSITION ENO EQ
F	
F_EDGE	FROM
G	
GE	GT
I	
INITIAL_STEP	INT_TO_BCD
L	
LE LINT LT	LWORD LWORD_TO_BCD
M	
MUL	MULTIME
N	
NE	
R	
R_EDGE	RESOURCE
S	

SEMA SINT_TO_BCD STEP SUB SUB_DATE_DATE	SUB_DT_DT SUB_DT_TIME SUB_TIME SUB_TOD_TIME SUB_TOD_TOD
T	
TRANSITION	
U	
ULINT	
V	
VAR_ACCESS VAR_ALIAS	VAR_EXTERNAL VAR_OBJECT
W	
WORD_TO_BCD	

Additional reserved identifiers

User definitions must avoid not only protected identifiers (Page 4658) and reserved identifiers (Page 4663) of the ST programming language, but also identifiers which have a defined meaning in SIMOTION or which are intended for future expansions.

Identifiers with a defined meaning in SIMOTION

Do not declare any identifiers (in variables declarations or data type declarations, for example) which are already defined in SIMOTION. If you do, your declarations can hide the identifiers defined in SIMOTION, which may make it impossible to access them. Under certain circumstances, the compiler may not be able to issue a corresponding warning if, for example, the associated technology package is not imported.

The following are defined as identifiers in SIMOTION:

- General standard functions including the associated data types
  (see also the SIMOTION Basic Functions Function Manual)
- General standard function blocks including the associated data types
  (see also the SIMOTION Basic Functions Function Manual)
- Functions for task control, runtime measurement, and message programming including the associated data types
  (see also the SIMOTION Basic Functions Function Manual)
- System functions and system variables of SIMOTION devices including the associated data types
  (see also the list manuals of the SIMOTION devices)
- System functions, system variables, and configuration data of technology objects including the associated data types
  (see also the list manuals of the technology packages)
- Protected and reserved identifiers of other programming languages

A list of all identifiers with a defined meaning can be found in the SIMOTION Basic Functions Function Manual.

Identifiers which are intended for future SIMOTION expansions

Avoid user-defined identifiers which are intended for future SIMOTION expansions. This is a precaution to ensure that your user-defined identifiers will not hide any identifiers defined in future versions of SIMOTION.

In particular, all identifiers starting with the following character strings are intended for future SIMOTION expansions:

- _ (underscore)
- **Command**
- **Enum**
- **Struct**

If you declare an identifier that starts with one of these character strings, a corresponding warning (e.g. warning number 16026) is issued.

Numbers and Boolean values

Numbers can be written in various ways in ST. A number can contain a sign, a decimal point or an exponent. The following rules apply to all numbers:

- Commas and blanks may not appear within a number.
- An underscore (_) is allowed as a visual separator.
- The number can be preceded by a plus (+) or minus (–). If no sign is used, it is assumed that the number is positive.
- Numbers may not violate certain maximum and minimum values.

Integers

An integer contains neither a decimal point nor an exponent. An integer is thus a sequence of numeric digits that can be preceded with a sign.

The following are valid integers:

```
0                1               +1              -1
743              -5280           60_000          -32_211_321
```

The following integers are **invalid** for the reasons indicated:

`123.456`	Commas are not permitted.
`36.`	An integer may not contain a decimal point.
`10 20 30`	Blanks are not permitted.

In ST, you can represent integers in different number systems. This is achieved by inserting a keyword prefix for the number system.

The following are used:

- 2# for the binary system
- 8# for the octal system
- 16# for the hexadecimal system.

Valid representations of the decimal number 15 are:

```
2#1111          8#17          16#F
```

Floating-point numbers

A floating-point number can contain a decimal point or an exponent (or both). A decimal point must appear between two digits. A floating-point number therefore cannot start or end with a decimal point.

The following are valid floating-point numbers:

```
0.0             1.3           -0.2          827,602
0000.0          +0.000743     60_000.15     -315.0066
```

The following floating-point numbers are **invalid**:

`1.`	A numeric digit must be present before the decimal point and after the decimal point.
`1,000.0`	Commas are not permitted.
`1,333,333`	Two points are not permitted.

Exponents

An exponent can be included to define the position of the decimal point. If no decimal point appears, it is assumed that it is on the right side of the digit. The exponent itself must be either a positive or negative integer. Base 10 is expressed by the letter E.

The magnitude 3×10^8 can be represented in ST by the following correct floating-point numbers:

```
3.0E+8          3.0E8          3e+8          3E8          0.3E+9
0.3e9           30.0E+7        30e7
```

The following floating-point numbers are **invalid**:

`3.E+8`	A numeric digit must be present before the decimal point and after the decimal point.
`8e2.3`	The exponent must be an integer.
`.333e-3`	A numeric digit must be present before the decimal point and after the decimal point.
`30 E8`	Blanks are not permitted.

Boolean values

Boolean values are bit constants. They must be represented by a value of zero (0) or one (1) or by the keywords FALSE or TRUE.

Example:

```
a := 1;         // corresponds to a := TRUE
b := FALSE;     // corresponds to b := 0
```

Data types of numbers

The compiler automatically selects the elementary data type that is suitable for the number depending on its value and use (in an expression or a value assignment).

You can also specify the data type directly: Place the data type (numeric data type or bit data type) and the character "#" in front of the number.

Examples:

```
INT#255                 INT#16#FF               INT#8#377
WORD#255                WORD#16#FF              WORD#8#377
REAL#255                REAL#16#FF              REAL#8#377
REAL#255.0              REAL#2.55E2             LREAL#255.0
```

Note

Floating-point numbers can only be assigned to data types REAL and LREAL.

Character strings

What is a character string?

A character string is a sequence of 0 or more characters with an apostrophe at the start and at the end. Each character is encoded with 1 byte (8 bits) in the string.

A character can be entered as follows:

- As printable characters (ASCII code $20 to $7E, $80 to $FF), except the dollar signs (ASCII code $24) and apostrophe (ASCII code $27), as these have a special function within the string
- As the two-digit hexadecimal ASCII code of the relevant character preceded by the dollar sign ($)
- As a combination of two characters according to the following table:

Table 7-372 Two-character combinations for special characters in strings

Character combination			Meaning
$$			Dollar sign $ ($24)
$'			Apostrophe ' ($27)
$L	or	$l	Line Feed LF ($0A)

Character combination			Meaning
$N	or	$n	Carriage Return + Line Feed CR + LF ($0D$0A)
$P	or	$p	Form Feed FF ($0C)
$R	or	$r	Carriage Return CR ($0D)
$T	or	$t	Horizontal tab (HT) ($09)

Examples:

''	Empty string (length 0).
'A'	String of length 1 containing the letter A.
' '	String of length 1 containing a blank.
'$''	String of length 1 containing an apostrophe.
'RL' '$0D$0A'	Two equivalent representations for a string of length 2 containing the characters CR and LF.
'$$1.00'	String of length 5 containing $1.00.
'Text$R$L'	String of length 6 containing the word Text followed by the characters CR and LF.
'ÄÖÜ' '$C4$D6$FC'	Two equivalent representations for a string of length 3 containing the German umlauts ÄÖü (A, O, u with diaresis).

7.2.4.3 Structure of an ST source file

An ST source basically consists of continuous text. This text can be structured by dividing it into logical sections. Detailed rules for this can be found in Source file modules (Page 4806).

A brief summary is given below:

- An **ST source file** is a logical unit that you create in your project and that can appear several times. It is often referred to as a unit.
- The logic sections of an ST source file are called **Sections** (see table).
- A **user program** is the sum of all program sources (e.g. ST source files, MCC units).

Each logical section of the ST source file has a beginning and end denoted by specific keywords:

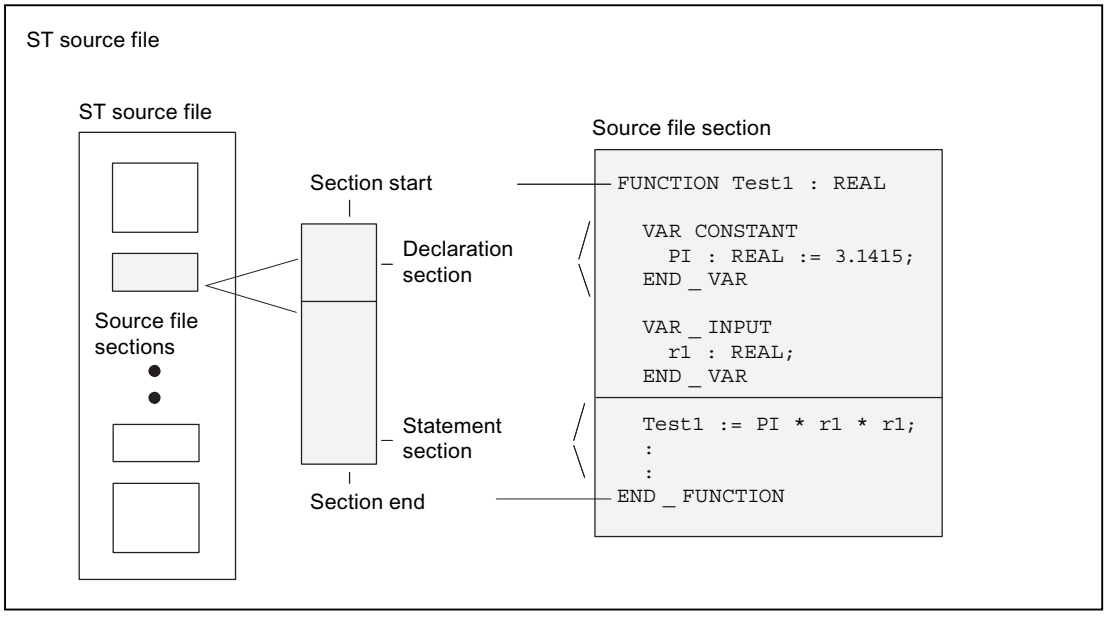


Figure 7-231 Structure of an ST source file

You do not have to program every function yourself. You can also make use of SIMOTION system components. These are preprogrammed sections such as system functions or the functions of the technology objects (TO functions).

Table 7-373 Major sections of an ST source file

Source file section	Description
Unit statement (optional)	Contains the name of the ST
Interface section	Contains instructions for making variables, data types and program organization units (POU) public, and for using other program sources.
Implementation section	Contains instructions for using other program sources and the executable sections of the ST source file.
POU (program organization unit)	Single executable section of the ST source file (program, function, function block)
Declaration section	Contains declarations (e.g. of variables and data types), can be included in the interface section and the implementation section as well as in a POU.
Statement section	Contains executable statements of a POU.

Note

An extensively annotated *template for example unit* is also available in the online help. You can use it as a template for a new ST source file.

Call the ST editor Help and click the relevant link. Copy the text to the open window of the ST editor and modify the template according to your requirements.

The section "Template for example unit" (Page 5068) contains a copy of this template.

Statements

The statement section of a program organization unit (POU – program, function, function block) consists of repeated single statements. It follows the declaration section of a POU and ends with the end of the POU. There are no explicit keywords for the start and end.

There are three basic types of statements in ST:

- Value assignments
  Assignment of an expression to a variable; see Variables declaration (Page 4694)
- Control statements
  Repetition or branching of statements; see Control statements (Page 4719)
- Subroutine execution
  Functions (FC) and function blocks (FB); see Functions, function blocks, programs (Page 4736)

Table 7-374 Examples of statements

```
...
// Value assignment
Status := 17;

// Control statement
IF a = b THEN
    FOR c := 1 TO 10 DO
        b := b + c;
    END_FOR;
END_IF;

// Function call
retVal := Test1(10.0);
...
```

Comments

Comments are used for documentation purposes and to help the user understand the source file section. After compilation, they have no meaning for the program execution.

There are two types of comments:

- Line comment
- Block comment

The line comment is preceded by **//**. The compiler will process the text which follows until the end of the line as a comment.

You can enter a block comment over several lines if it is preceded by (* and ends with *).

You can also use the corresponding buttons in the ST editor toolbar to show or hide comments. This is helpful for showing or hiding code positions.

Please note the following when inserting comments:

- The character pairs (* and *) are ignored within the line comment.
- Nesting of block comments is not allowed as standard. However, you can nest line comments in block comments.
  With activated compiler option “Permit language extensions IEC61131 3rd edition” only: block comments can be nested.
- You can use the complete extended ASCII character set in comments.
- Comments can be inserted at any position, but not in rules that have to be maintained, such as in names of identifiers. For more information about these rules, refer to Help for the language description (Page 4973).

Table 7-375 Examples of comments

```
// This is a one-line comment.
a := 5;

// This is an example of a single-line
// comment used several times in succession.
b := 23;

(* The example above is easier to maintain as a block comment.
*)
c := 87;

(* Block comments can only be nested with compiler option "Admit language
extensions IEC61131 3rd edition".
(* Nested block comment *)
*)
```

7.2.4.4 Data types

A data type is used to determine how the value of a variable or constant is to be used in a program source.

The following data types are available to the user:

- Elementary data types (Page 4672)
- User-defined data types (UDT) (Page 4675)
  - Simple derivatives
  - Arrays
  - Enumerators
  - Structures (Struct)
- Technology object data types (Page 4690)
- System data types (Page 4693)

Elementary data types

Elementary data types define the structure of data that cannot be broken down into smaller units. An elementary data type describes a memory area with a fixed length and stands for bit data, integers, floating-point numbers, duration, time, date and character strings.

All the elementary data types are listed in the table below:

Table 7-376 Bit widths and value ranges of the elementary data types

Type		Reserv. word	Bit width	Range of values
Bit data type Data of this type uses either 1 bit, 8 bits, 16 bits, or 32 bits. The initialization value of a variable of this data type is 0.				
	Bit	BOOL	1	0, 1 or FALSE, TRUE
	Byte	BYTE	8	16#0 to 16#FF
	Word	WORD	16	16#0 to 16#FFFF
	Double word	DWORD	32	16#0 to 16#FFFF_FFFF
Numeric types These data types are available for processing numeric values. The initialization value of a variable of this data type is 0 (all integers) or 0.0 (all floating-point numbers).				
	Short integer	SINT	8	-128 to 127 (-2**7 to 2**7-1)
	Unsigned short integer	USINT	8	0 to 255 (0 to 2**8-1)
	Integer	INT	16	-32_768 to 32_767 (-2**15 to 2**15-1)
	Unsigned integer	UINT	16	0 to 65_535 (0 to 2**16-1)
	Double integer	DINT	32	-2_147_483_648 to 2_147_483_647 (-2**31 to 2**31-1)
	Unsigned double integer	UDINT	32	0 to 4_294_967_295 (0 to 2**32-1)
	Floating-point number (per IEEE -754)	REAL	32	-3.402_823_466E+38 to -1.175_494_351E-38, 0.0, +1.175_494_351E-38 to +3.402_823_466E+38 Accuracy: 23-bit mantissa (corresponds to 6 decimal places), 8-bit exponent, 1-bit sign.
	Long floating-point number (in accordance with IEEE-754)	LREAL	64	-1.797_693_134_862_315_7E+308 to -2.225_073_858_507_201_4E-308, 0.0, +2.225_073_858_507_201_4E-308 to +1.797_693_134_862_315_7E+308 Accuracy: 52-bit mantissa (corresponds to 15 decimal places), 11-bit exponent, 1-bit sign.
Time types These data types are used to represent various date and time values.				

Type		Reserv. word	Bit width	Range of values
	Duration in increments of 1 ms	TIME	32	T#0d_0h_0m_0s_0ms to T#49d_17h_2m_47s_295ms Maximum of 2 digits for the values day, hour, minute, second and a maximum of 3 digits for milliseconds Initialization with T#0d_0h_0m_0s_0ms
	Date in increments of 1 day	DATE	32	D#1992-01-01 to D#2200-12-31 Leap years are taken into account, year has four digits, month and day are two digits each Initialization with D#0001-01-01
	Time of day in increments of 1 ms	TIME_OF_DAY (TOD)	32	TOD#0:0:0.0 to TOD#23:59:59.999 Maximum of two digits for the values hour, minute, second and maximum of three digits for milliseconds Initialization with TOD#0:0:0.0
	Date and time	DATE_AND_TIME (DT)	64	DT#1992-01-01-0:0:0.0 to DT#2200-12-31-23:59:59.999 DATE_AND_TIME consists of the data types DATE and TIME Initialization with DT#0001-01-01-0:0:0.0
String type Data of this type represents character strings, in which each character is encoded with the specified number of bytes. The length of the string can be defined at the declaration. Indicate the length in "[" and "]", e.g. STRING[100]. The default setting consists of 80 characters. The number of assigned (initialized) characters can be less than the declared length.				
	String with 1 byte/character	STRING	8	You may use all the characters in the extended ASCII character set (ASCII code $00 to $FF). Default ' ' (empty string)

Note

During variable export to other systems, the value ranges of the corresponding data types in the target system must be taken into account.

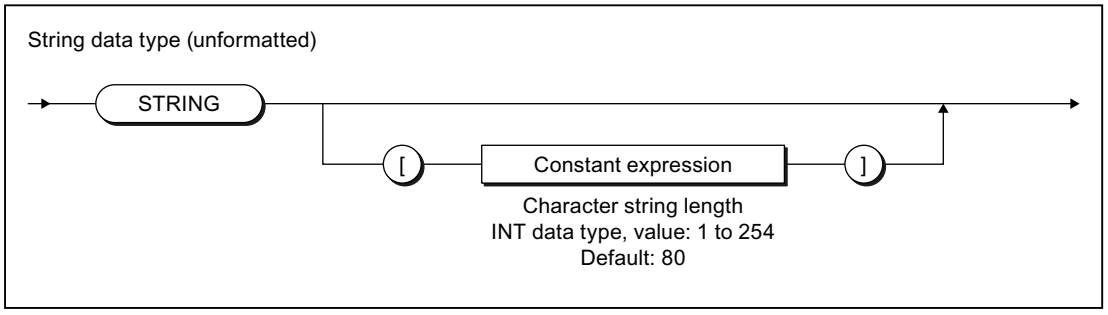


Figure 7-232 Syntax: STRING data type

Value range limits of elementary data types

The value range limits of certain elementary data types are available as constants.

Table 7-377 Symbolic constants for the value range limits of elementary data types

Symbolic constant	Data type	Value	Hex notation
SINT#MIN	SINT	-128	16#80
SINT#MAX	SINT	127	16#7F
INT#MIN	INT	-32768	16#8000
INT#MAX	INT	32767	16#7FFF
DINT#MIN	DINT	-2147483648	16#8000_0000
DINT#MAX	DINT	2147483647	16#7FFF_FFFF
USINT#MIN	USINT	0	16#00
USINT#MAX	USINT	255	16#FF
UINT#MIN	UINT	0	16#0000
UINT#MAX	UINT	65535	16#FFFF
UDINT#MIN	UDINT	0	16#0000_0000
UDINT#MAX	UDINT	4294967295	16#FFFF_FFFF
T#MIN TIME#MIN	TIME	T#0ms	16#0000_0000[1]
T#MAX TIME#MAX	TIME	T#49d_17h_2m_47s_295ms	16#FFFF_FFFF[1]
TOD#MIN TIME_OF_DAY#MIN	TOD	TOD#00:00:00.000	16#0000_0000[1]
TOD#MAX TIME_OF_DAY#MAX	TOD	TOD#23:59:59.999	16#0526_5BFF[1]
[1] Internal display only			

General data types

General data types are often used for the input and output parameters of system functions and system function blocks. The subroutine can be called with variables of each data type that is contained in the general data type.

The following table lists the available general data types:

Table 7-378 General data types

General data type	Data types contained
ANY_BIT	BOOL, BYTE, WORD, DWORD
ANY_INT	SINT, INT, DINT, USINT, UINT, UDINT
ANY_REAL	REAL, LREAL
ANY_NUM	ANY_INT, ANY_REAL
ANY_DATE	DATE, TIME_OF_DAY (TOD), DATE_AND_TIME (DT)
ANY_ELEMENTARY	ANY_BIT, ANY_NUM, ANY_DATE, TIME, STRING
ANY	ANY_ELEMENTARY, user-defined data types (UDT), system data types, data types of the technology objects

Note

You **cannot** use general data types as type identifiers in variable or type declarations.

The general data type is retained when a user-defined data type (UDT) is derived directly from an elementary data type (only possible with the SIMOTION ST programming language).

Elementary system data types

In the SIMOTION system, the data types specified in the table are treated in a similar way to the elementary data types. They are used with many system functions.

Table 7-379 Elementary system data types and their use

Identifier	Bit width	Use
StructAlarmId	32	Data type of the alarmId for the project-wide unique identification of the messages. The alarmId is used for the message generation. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTALARMID#NIL
StructTaskId	32	Data type of the taskId for the project-wide unique identification of the tasks in the execution system. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTTASKID#NIL

Table 7-380 Symbolic constants for invalid values of elementary system data types

Symbolic constant	Data type	Meaning
STRUCTALARMID#NIL	StructAlarmId	Invalid AlarmId
STRUCTTASKID#NIL	StructTaskId	Invalid TaskId

User-defined data types

User-defined data types (UDT) are created with the construct TYPE/END_TYPE in the declaration subsections of subsequent source file modules (see Breakdown of ST source file (Page 4668) and Source file modules (Page 4806)):

- Interface section
- Implementation section
- Program organization unit (POU)

You can continue to use the data types you created in the declaration section. The source file section determines the range of the type declaration.

See also

Syntax of user-defined data types (type declaration) (Page 4676)
Derivation of elementary or derived data types (Page 4677)
Derived data type ARRAY (Page 4678)
Derived data type - Enumerator (Page 4681)
Derived data type STRUCT (structure) (Page 4682)

Syntax of user-defined data types (type declaration)

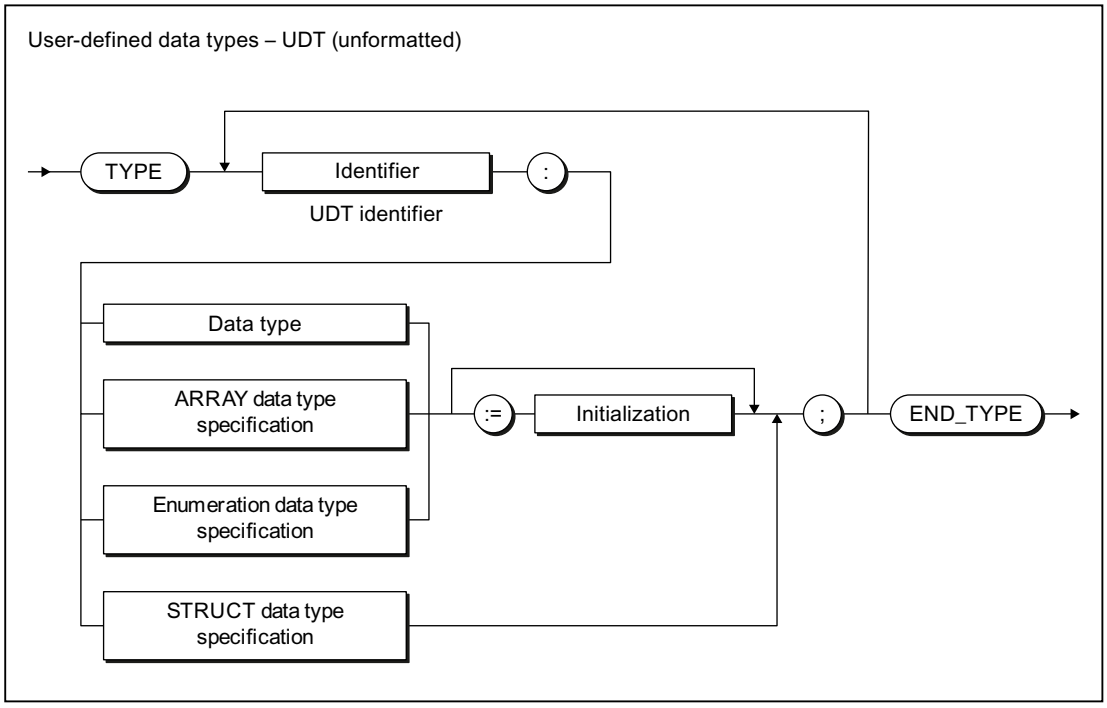


Figure 7-233 Syntax: User-defined data type

The declaration of the UDT is introduced with the keyword TYPE.

For each data type to be declared, this is followed by, see figure:

1. Name:
   The name of the data type must comply with the rules for identifiers.
2. Data type specification
   The term *data type* comprises, see Derivation of elementary or derived data types (Page 4677):
   - Elementary data types
   - Previously declared UDTs
   - TO data types
   - System data types

   The following data type specifications are also possible:
   - ARRAY data type specification, see Derived data type ARRAY - array (Page 4678)
   - Enumerator data type specification, see Derived data type enumeration – enumerator (Page 4681)
   - STRUCT data type specification, see Derived data type STRUCT – structure (Page 4682)

   The references in brackets refer to the following sections, in which the respective data type specification is described in detail.
3. Optional initialization:
   You can specify an initialization value for the data type. If you subsequently declare a variable of this data type, the initialization value is assigned to the variable.
   Exception: With the STRUCT data type specification, each individual component is initialized within the data type specification.
   See also Initialization of variables or data types (Page 4697).

The complete UDT declaration is terminated with the keyword END_TYPE. You can create any number of data types within the TYPE/END_TYPE construct. You can use the defined data types to declare variables or parameters.

UDTs can be nested in any way as long as the syntax in the figure is observed. For example, you can use previously defined UDTs or nested structures as a data type specification. Type declarations can only be used sequentially and not in nested structures.

Note

You can learn how to declare variables and parameters in Overview of all variable declarations (Page 4695), and how to assign values with UDT in Syntax for value assignment (Page 4703).

Below is a description of individual data type specifications for UDTs and examples demonstrating their use.

Derivation of elementary or derived data types

In the derivation of data types, an elementary or user-defined data type (UDT) is assigned to the data type to be defined in the TYPE/END_TYPE construct:

TYPE identifier: Elementary data type { := initialization } ; END_TYPE

TYPE identifier: User-defined data type { := initialization } ; END_TYPE

Once you have declared the data type, you can define variables of derived data type identifier. This is equivalent to declaring variables as data type *elementary data type*.

Table 7-381 Examples of derivation of elementary data types

```
TYPE
    I1: INT;      // Elementary data type
    R1: REAL;     // Elementary data type
    R2: R1;       //  Derived data type (UDT)
END_TYPE
VAR
    // These variables can be used wherever
    // variables of type INT can be used.
    myI1 : I1;
    myI2 : INT;   // No derived data type!

    // These variables can be used wherever
    // variables of type REAL can be used.
    myR1 : R1;
    myR2 : R2;
END_VAR
myI1 := 1;
myI2 := 2;
myR1 := 2.22;
myR2 := 3.33;
```

Derived data type ARRAY

The ARRAY derived data type combines a defined number of elements of the same data type in the TYPE/END_TYPE construct. The syntax diagram in the following figure shows this data type which is specified more precisely after the reserved identifier OF.

TYPE identifier: ARRAY data type specification { := initialization } ; END_TYPE

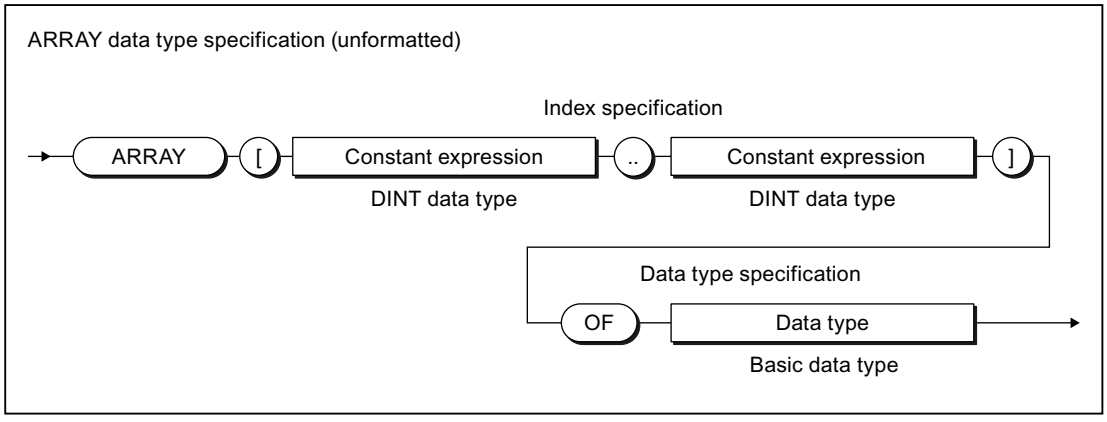


Figure 7-234 Syntax: ARRAY data type specification

The **index specification** describes the limits of the array:

- The array limits specify the minimum and maximum value for the index. They can be specified using constants or constant expressions; the data type is DINT (or can be implicitly converted to DINT – see Converting elementary data types (Page 4732)).
- The array limits must be separated by two periods.
- The entire index specification is enclosed in square brackets.
- The index itself can be an integer value of data type DINT (or it can be implicitly converted to DINT – see Converting elementary data types (Page 4732)).

Note

If array limits are violated during runtime, a processing error occurs in the program (see the "SIMOTION Basic Functions" Function Manual).

You specify the data type of the array elements with the **data type specification** (basic data type). All of the options described in this section can be used as data types, for example, even user-defined data types (UDT).

You can define the size of the memory space occupied by the ARRAY with _sizeOf (in := *array-name*). The following applies: _sizeOf (in := *array_of_type*) := _lengthIndexOf (in := *array_of_type*) * _sizeOf (in := *type*). The syntax of these functions is described in the "SIMOTION Basic Functions" Function Manual.

The _arrayCopy function can be used to copy an area of an array into another array with the same basic data type. The syntax of these functions is described in the "SIMOTION Basic Functions" Function Manual.

Note

As of SIMOTION Kernel version V4.2, you can declare arrays with a dynamic length as in-out parameters in functions and function blocks. For more information, refer to the corresponding section (Page 4747) for the FB and FC declaration subsection (Page 4741).

One- and multidimensional arrays

There are several different ARRAY types:

- The one-dimensional ARRAY type is a list of data elements arranged in ascending order.
- The two-dimensional ARRAY type is a table of data consisting of lines and columns. The first dimension refers to the line number, the second to the column number.
- The higher-dimensional ARRAY type is an expansion of the two-dimensional ARRAY type that includes additional dimensions.

Table 7-382 Examples of one-dimensional arrays

```
TYPE
    x : ARRAY[0..9] OF REAL;
    y : ARRAY[1..10] OF C1;
END_TYPE
```

Two-dimensional arrays are comparable to a table with lines and columns. You can create two- or multi-dimensional arrays by means of a multi-level type declaration, see example:

Table 7-383 Examples of multi-dimensional arrays

```
TYPE
    a       : ARRAY[1..3] OF INT;     // Single-dimensional array
                                      // (3 columns)
    matrix1: ARRAY[1..4] OF a;        // Two-dimensional array
                                      // (4 lines with 3 columns)
    b       : ARRAY[4..8] OF INT;     // single-dimensional field
                                      // (5 columns)
    matrix2: ARRAY[10..16] OF b;      // Two-dimensional array
                                      // (7 lines with 5 columns)
END_TYPE

VAR
    m : matrix1;     // Variable m of data type
                     // Two-dimensional array
    n : matrix2;     // Variable n of data type
                     // Two-dimensional array
END_VAR

m[4][3] := 9;       // Write in matrix1 in line 4, column 3
n[16][8] := 10;     // Write in matrix2 in line 7, column 5
```

In the example, you can define:

1. Table columns a[1] to a[3] as a one-dimensional array that will contain integers.
2. Table lines matrix1[1] to matrix1[4] also as an array, but take as the data type specification the array a you just created with the columns of the table.
   When you specify an array in the data type specification, you create a second dimension. You can create further dimensions in this way.

Now declare a variable using the data type created for the table. You address each dimension of the table using square brackets, in this case specifying the line and column.

Redeclaration

The data type definitions of arrays are recognized as identical (even in different sources) if the following conditions are met:

1. The identifier of the data type is identical.
2. Both array limits are identical.
3. The data type of the array elements is identical.
4. The optional array initialization list is identical (including repeat factors and constant expressions).

Initialization

Per default, the array elements are assigned the initialization value of the basic data type. Optionally the initialization value of the array element can be changed by assigning an array initialization list enclosed in square brackets [ ], see"Initialization of variables or data types" (Page 4697).

Derived data type - Enumerator

In the case of enumeration data types, a restricted set of identifiers or names is assigned to the data type to be defined in the TYPE/END_TYPE construct:

TYPE identifier: Enumeration data type specification { := initialization } ; END_TYPE

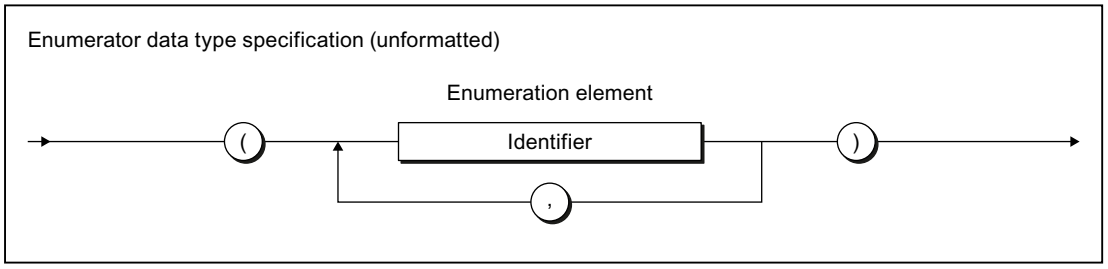


Figure 7-235 Syntax: Enumeration data type specification

Once you have declared the *identifier* data type, you can define variables in the enumeration data type. In the statement section, you can assign only elements from the list of defined identifiers (enumeration elements) to these variables.

You can also specify the data type directly: Place the enumeration data type identifier and the "#" sign in front of the enumeration element (see Table *Examples of enumeration data types*).

You can obtain the first and last value of an enumeration data type with *enum_type*#MIN and *enum_type*#MAX respectively, whereby *enum_type* is the enumeration data type identifier.

You can obtain the numeric value of an enumeration element with the ENUM_TO_DINT conversion function. The syntax of these functions is described in the "SIMOTION Basic Functions" Function Manual.

Example

Table 7-384 Examples of enumeration data types

```
TYPE
    C1 : (RED, GREEN, BLUE);
END_TYPE

VAR
    myC11, myC12, myC13 : C1;
END_VAR

myC11 := GREEN;
myC11 := C1#GREEN;
myC12 := C1#MIN;        // RED
myC13 := C1#MAX;        // BLUE
```

Note

You will also find enumeration data types as system data types.

Enumeration data types can be components of a structure, meaning that they can be found at any lower level in the user-defined data structure.

Redeclaration

The data type definitions of enumeration data types are recognized as identical (even in different sources) if the following conditions are met:

1. The identifier of the data type is identical.
2. All enumeration elements are identical.
3. The sequence of the enumeration elements is identical, so their numerical values are identical.
4. The optional initialization value is identical.

Initialization

Per default, an enumeration data type is assigned the 1st value of the enumeration as initialization value. Optionally the initialization value can be changed by assigning another enumeration element, see"Initialization of variables or data types" (Page 4697).

Derived data type STRUCT (structure)

The derived data type STRUCT, or structure, encompasses an area of a fixed number of components in the TYPE/END_TYPE construct; the data types of these components can vary:

TYPE identifier: STRUCT data type specification; END_TYPE

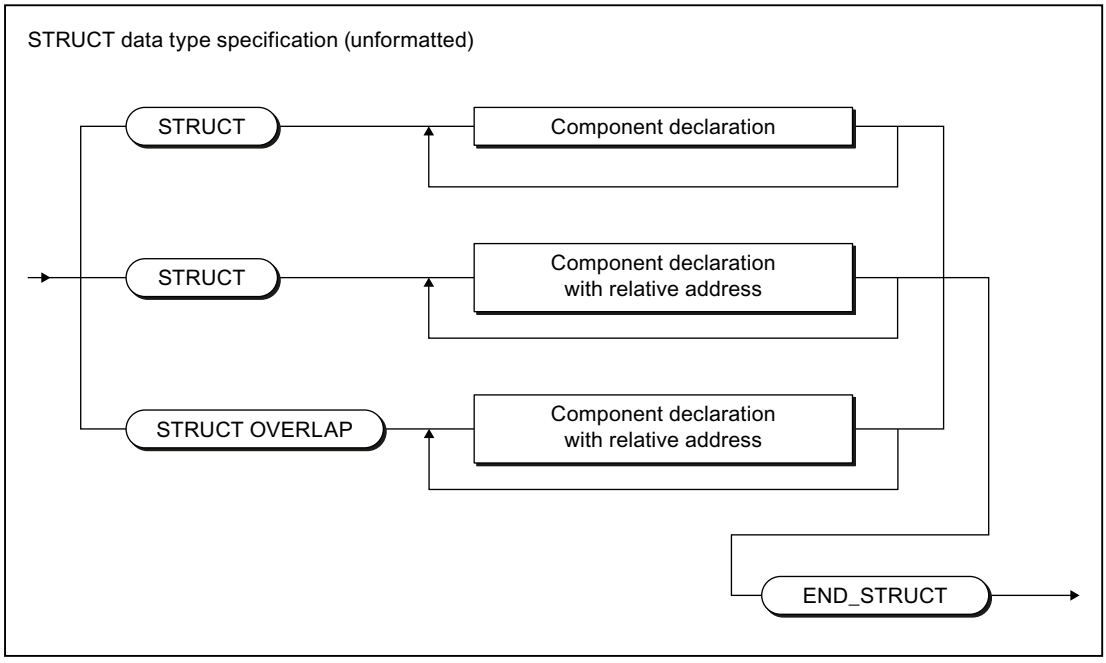


Figure 7-236 Syntax: STRUCT data type specification

Component declaration

Usually, a structure contains the definitions of the individual components between the keywords STRUCT and END_STRUCT.

The syntax of the component declaration is shown in the following figure. The components are arranged in the order of their declaration with this syntax. The relative addresses of the components within the structure are assigned automatically by the compiler.

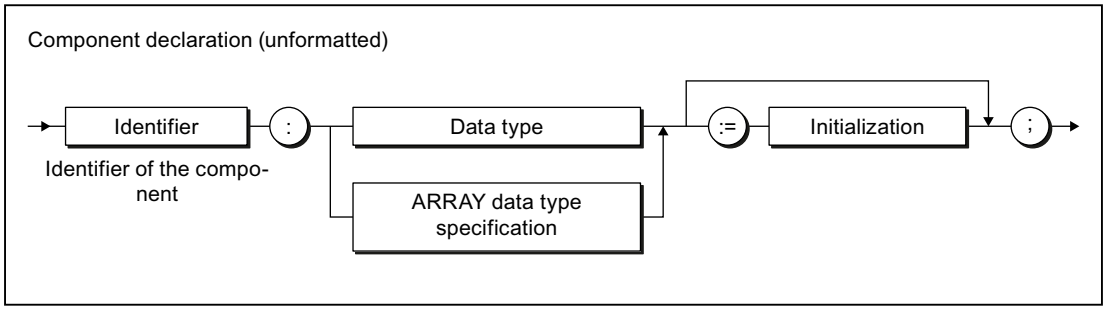


Figure 7-237 Syntax: Component declaration

The following are permitted as data types:

- Elementary data types
- Previously declared UDTs
- System data types
- TO data types
- ARRAY data type specification

You also have the option to assign initialization values to the components see "Initialization" in this section.

Note

The following data type specifications cannot be used directly within a component declaration:

- STRUCT data type specifications
- Enumeration data type specifications

Remedy:

Declare a UDT (user-defined data type) beforehand with the above-mentioned specifications and use this in the component declaration.

This allows you to nest STRUCT data types.

You will also find STRUCT data types as system data types.

Example

This example shows how a UDT is defined and how this data type is used within a variable declaration.

Table 7-385 Examples of derived data type STRUCT

```
TYPE    // UDT definition
    S1 : STRUCT
        var1 : INT;
        var2 : WORD := 16#AFA1;
        var3 : BYTE := 16#FF;
        var4 : TIME := T#1d_1h_10m_22s_2ms;
    END_STRUCT;
END_TYPE

VAR
    myS1 : S1;
END_VAR

myS1.var1 := -4;
myS1.var4 := T#2d_2h_20m_33s_2ms;
```

Redeclaration

The data type definitions of structures are recognized as identical (even in different sources) if the following conditions are met:

1. The identifier of the data type is identical.
2. All components of the structure are identical in terms of their:
   - Sequence
   - Identifier
   - Data types or ARRAY data type specifications
   - Initialization values (including any array or structure initialization lists)

Initialization

Per default, each component is assigned the initialization value of its data type. Optionally, the initialization value of the component can be changed by assigning an appropriate initialization.

When using the structure in another declaration (e.g. variable or data type declaration), the initialization values of individual components can be changed by assigning a structure initialization list, enclosed in round brackets ( ).

See "Initialization of variables or data types" (Page 4697).

Component declaration with specification of the relative addresses

You can assign the relative addresses of the components within the structure with the keyword AT in the component declarations, see following syntax diagram.

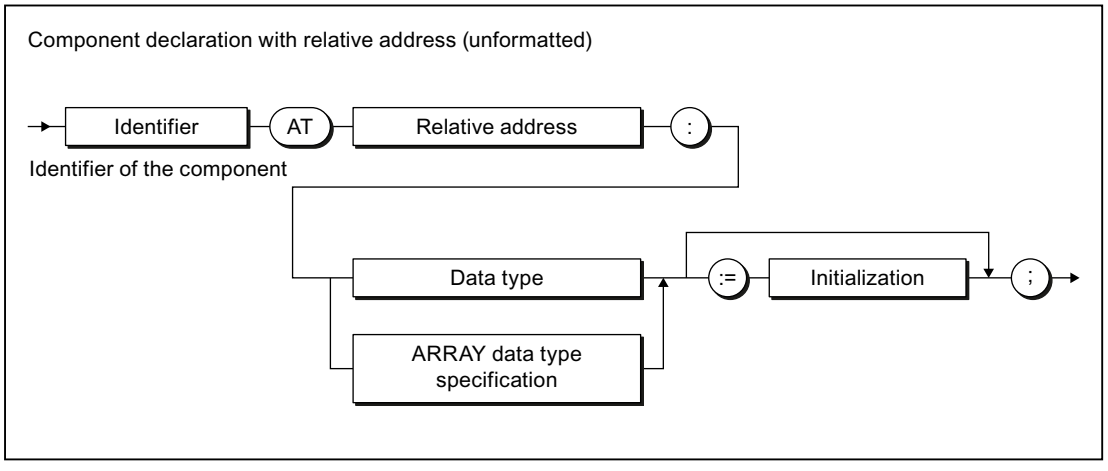


Figure 7-238 Syntax: Component declaration with relative address

Specify the relative address of the component (byte offset) as follows:

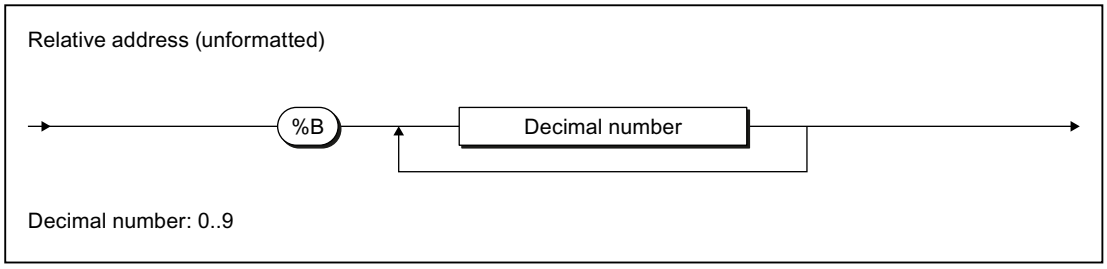


Figure 7-239 Syntax: Relative address

If you specify relative addresses of the structure components, you must do this for all components of this structure.

The permissible relative addresses of the components depends on their data type, see following table:

Table 7-386 Permissible addresses and length of various data types

Data type	Permissible address	Length
Elementary data types		
BOOL, BYTE, SINT, USINT	Any	1 byte
WORD, INT, UINT	Can be divided by 2	2 bytes
DWORD, DINT, UDINT, REAL TIME, DATE, TIME_OF_DAY (TOD)	Can be divided by 4	4 bytes
LREAL DATE_AND_TIME (DT)	Can be divided by 8	8 bytes
STRING	Any address	Declared length + 2 bytes (Default: 82 bytes)
User-defined data types (UDT)		
Derived data type	Corresponding to the basic data type	
ARRAY	Corresponding to the basic data type	Corresponding to the index specification and the basic data type
Enumerator	Can be divided by 4	4 bytes
STRUCT (structure)	Corresponding to the largest data type within the structure	Corresponding to the declaration of the structure
Further data types		
StructAlarmId, StructTaskId	Can be divided by 4	4 bytes
System data type	Corresponding to its declaration as STRUCT or enumerator	
TO data type, ANYOBJECT	Can be divided by 8	8 bytes

If the above addressing rules are violated, the compiler issues an error message which indicates the required divisor.

Table 7-387 Example of a structure with specification of the relative address

```
myLocatedStruct : STRUCT
    memb1 AT %B0 : INT;     // Relative address 0
                            // Bytes 0 and 1 assigned
    memb2 AT %B8 : REAL;    // Relative address 8
                            // Bytes 8 .. 11 assigned
                            // Bytes 2 .. 7 not assigned (gap)
END_STRUCT
```

The declaration order of the components is arbitrary when specifying the relative addresses. The structure in the following example is identical to that in the previous example.

Table 7-388 Identical structure to the previous example

```
myLocatedStruct : STRUCT
    memb2 AT %B8 : REAL;    // Relative address 8
                            // Bytes 8 .. 11 assigned
    memb1 AT %B0 : INT;     // Relative address 0
                            // Bytes 0 and 1 assigned
                            // Bytes 2 .. 7 not assigned (gap)
END_STRUCT
```

The address ranges of the components must not overlap except when the structure is declared with the keyword STRUCT OVERLAP, see following section.

Structure with overlapping address ranges (UNION)

If you make the component declarations with specification of the relative address (see above) between the keywords STRUCT OVERLAP and END_STRUCT, the address ranges can overlap.

The syntax diagram for the component declaration with relative address (see above) applies for the component declaration. See Table "Permissible addresses and length of various data types" for the permissible addresses.

Table 7-389 Example of a structure with overlapping address ranges

```
myOverLapStruct : STRUCT OVERLAP
    memb1 AT %B0 : REAL;    // Relative address 0
                            // Bytes 0 .. 3
    memb2 AT %B0 : INT;     // Also on relative address 0
                            // Bytes 0 and 1
    ar    AT %B0 : ARRAY [0..3] OF BYTE;
                            // Relative address 0 again
                            // Bytes 0 .. 3
END_STRUCT
```

If the address ranges of two components overlap, the following data types are **not** permitted in these components:

- STRING
- ANYOBJECT
- TO data type

They are permitted in components whose address ranges do not overlap. For example, the following declaration is permitted:

```
myOverLapStruct_2 : STRUCT OVERLAP
    dint_1  AT %B0 : DINT;
    udint_1 AT %B4 : UDINT;
    ar      AT %B0 : ARRAY [0..7] OF BYTE;
            //Bytes 0 .. 7 overlapping
    pos_1   AT %B8 : PosAxis;
            // Bytes 8 .. 15 not overlapping
END_STRUCT
```

Note

The declaration of a structure with overlapping ARRAY OF BYTE, as specified in the examples, does not replace the marshalling functions. The byte order (Little Endian or Big Endian) depends on the respective SIMOTION device. For information on the Little Endian and Big Endian byte order as well as the marshalling functions, see SIMOTION Basic Functions Function Manual.

Visibility of the components with overlapping address ranges

Symbol information (OPC-XML data) is only generated for the following components:

- Components without overlapping of address ranges
- For several components with overlapping address ranges:
  Only for the last declared component that uses the respective address range. The compiler issues a warning for the hidden components.

Only these components are visible:

- For the watch function of IT DIAG
- For the *_exportUnitDataSet* and *_importUnitDataSet* functions, see SIMOTION Basic Functions Function Manual
- For automatic displays in SIMOTION SCOUT (e.g. symbol browser (Page 4940), program status (Page 4949)).

Note

Hidden components can used in the programming. In the ST editor, they are shown in the "Automatic completion" (Page 4609) function.

You can monitor individual hidden components in a watch table (Page 4944), see the appropriate information in Section "Using a watch table" (Page 4944).

As an example, the visible components of the following structure declaration are explained in the table below.

The following examples of a structure declaration explain the behavior of the visible and hidden components. The visible components and the associated relative addresses are specified in the following table.

```
mystru : STRUCT OVERLAP
    a  AT %B0 : INT;    // Warning: Not visible for OPC-XML
    b  AT %B6 : INT;    // Warning: Not visible for OPC-XML
    c  AT %B8 : INT;    // Visible
    ar AT %B0 : ARRAY [0..6] OF BYTE; // Visible
END_STRUCT
```

	Relative address									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Declared components	a						b		c	
	ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]			
Visible components	ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]	-	c	

A change in the declaration order has an effect on the visible components:

```
mystru1 : STRUCT OVERLAP
    ar AT %B0 : ARRAY [0..6] OF BYTE; // Warning: For OPC-XML
                                      // Not visible
    a  AT %B0 : INT;    // Visible
    b  AT %B6 : INT;    // Visible
    c  AT %B8 : INT;    // Visible
END_STRUCT
```

	Relative address									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Declared components	ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]			
	a						b		c	
Visible components	a		-	-	-	-	b		c	

Initialization of the components with overlapping addresses

Initialization values are ignored for components that are not visible. The compiler issues an appropriate warning.

If initialization values are specified for the visible components, they are taken into account. These initialization values are also valid for the commonly used addresses of hidden components. The following applies to the addresses of hidden components that are not overlapped by the other components: They are assigned the default initialization value that was defined in the declaration of the appropriate data type, or zero.

The following examples of a structure declaration explain this initialization behavior. The initialization values of the respective relative addresses are specified in the following table.

```
mystru_init : STRUCT OVERLAP
    a  AT %B0 : INT := 16#AAAA;     // Initialization value warning
                                    // OPC-XML warning
    b  AT %B6 : INT := 16#BBBB;     // Initialization value warning
                                    // OPC-XML warning
    c  AT %B8 : INT := 16#CCCC;
    ar AT %B0 : ARRAY [0..6] OF BYTE := [7(1)];
END_STRUCT
```

	Relative address									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Declared components	a						b		c	
	ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]			
Initialization value	16#01	16#01	16#01	16#01	16#01	16#01	16#01	16#00	16#CCCC	

A change in the declaration order has an effect on the initialization values:

```
mystru_init1 : STRUCT OVERLAP
    ar AT %B0 : ARRAY [0..6] OF BYTE := [7(1)];
                                    // Initialization value warning
                                    // OPC-XML warning
    a  AT %B0 : INT := 16#AAAA;
    b  AT %B6 : INT := 16#BBBB;
    c  AT %B8 : INT := 16#CCCC;
END_STRUCT
```

	Relative address									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Declared components	ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]			
	a						b		c	
Initialization value	16#AAAA		16#00	16#00	16#00	16#00	16#BBBB		16#CCCC	

Technology object data types

Description of the technology object data types

You can declare variables with the data type of a technology object (TO). The following table shows the data types for the available technology objects in the individual technology packages.

For example, you can declare a variable with the data type *posaxis* and assign it an appropriate instance of a position axis. Such a variable is often referred to as a reference.

Table 7-390 Data types of technology objects (TO data type)

Technology object	Data type	Contained in the technology package
Drive axis	DriveAxis	CAM, PATH[1], CAM_EXT
External encoder	ExternalEncoderType	CAM, PATH[1], CAM_EXT
Measuring input	MeasuringInputType	CAM, PATH[1], CAM_EXT
Output cam	OutputCamType	CAM, PATH[1], CAM_EXT
Cam track	_CamTrackType	CAM, PATH[1], CAM_EXT
Position axis	PosAxis	CAM, PATH[1], CAM_EXT
Following axis	FollowingAxis	CAM, PATH[1], CAM_EXT
Following object	FollowingObjectType	CAM, PATH[1], CAM_EXT
Cam	CamType	CAM, PATH[1], CAM_EXT
Path axis[1]	_PathAxis	PATH[1], CAM_EXT
Path object[1]	_PathObjectType	PATH[1], CAM_EXT
Fixed gear	_FixedGearType	CAM_EXT
Addition object	_AdditionObjectType	CAM_EXT
Formula object	_FormulaObjectType	CAM_EXT
Sensor	_SensorType	CAM_EXT
Controller object	_ControllerObjectType	CAM_EXT
Temperature channel	TemperatureControllerType	TControl
General data type, to which every TO can be assigned	ANYOBJECT	

[1] Available as of version V4.1.

You can access the elements of technology objects (configuration data and system variables) via structures (see SIMOTION Basic Functions Function Manual).

Table 7-391 Symbolic constants for invalid values of technology object data types

Symbolic constant	Data type	Meaning
TO#NIL	ANYOBJECT	Invalid technology object

See also

Inheritance of the properties for axes (Page 4692)

Examples of the use of technology object data types (Page 4692)

Inheritance of the properties for axes

Inheritance for axes means that all of the data types, system variables and functions of the TO driveAxis are fully included in the TO positionAxis. Similarly, the position axis is fully included in the TO synchronizedAxis, the following axis in the TO pathAxis. This has, for example, the following effects:

- If a function or a function block expects an input parameter of the driveAxis data type, you can also use a position axis or a synchronized axis or a path axis when calling.
- If a function or a function block expects an input parameter of the posAxis data type, you can also use a synchronized axis or a path axis when calling.

Examples of the use of technology object data types

Below, you will see an example of optional use of a variable with a technology object data type (you will find an example of mandatory use of a variable with a TO data type in the *SIMOTION Basic Functions* Function Manual). A second example shows the alternative without using a variable with TO data type.

A TO function will be used to enable an axis in the main part of a program so that the axis can be positioned. After the positioning operation, the current position of the axis will be recorded using a structure access.

The first example uses a variable with TO data type to demonstrate its use.

Table 7-392 Example of the use of a data type for technology objects

```
VAR
    myAxis : posAxis;   // Declaration variable for axis
    myPos : LREAL;      // Variable for position of axis
    retVal: DINT;      // Variable for return value of the
                // TO function
END_VAR
myAxis := Axis1;       // The name Axis1 was defined when the axis
                       // was configured in the project navigator.

// Call of function with variables of TO data type:
retVal := _enableAxis(axis := myAxis, commandId := _getCommandId());

// Axis is positioned.
retVal := _pos(axis := myAxis,
               position := 100,
               commandId:= _getCommandId() );

// Scan the position using structure access
myPos := myAxis.positioningState.actualPosition;
```

The second example does not use a variable with TO data type.

Table 7-393 Example of using a technology object

```
VAR
    myPos : LREAL;      // Variable for position of axis
    retVal: DINT;  // Variable for return value of TO function
END_VAR

// Call of function without variable of TO data type
// The name Axis1 was defined when the axis
// was configured in the project navigator.
retVal := _enableAxis(axis := Axis1,
                      commandId:= _getCommandId() );

// Axis is positioned.
retVal := _pos(axis := Axis1
               position := 100,
               commandId:= _getCommandId() );

// Scan the position using structure access
myPos := Axis1.positioningState.actualPosition;
```

You will find details for configuration of technology objects in the SIMOTION Motion Control function descriptions.

System data types

There are a number of system data types available that you can use without a previous declaration. And, each imported technology packages provides a library of system data types.

Additional system data types (primarily enumerator and STRUCT data types) can be found

- In parameters for the general standard functions (see *SIMOTION Basic Functions* Function Manual)
- In parameters for the general standard function modules (see *SIMOTION Basic Functions* Function Manual)
- In system variables of the SIMOTION devices (see relevant parameter manuals)
- In parameters for the system functions of the SIMOTION devices (see relevant parameter manuals)
- In system variables and configuration data of the technology objects (see relevant parameter manuals)
- In parameters for the system functions of the technology objects (see relevant parameter manuals)

7.2.4.5 Variable declaration

A variable defines a data item with variable contents that can be used in the ST source file. A variable consists of an identifier (e.g. *myVar1*) that can be freely selected and a data type (e.g. BOOL). Reserved identifiers (see Reserved identifiers (Page 4657)) must not be used as identifiers.

Syntax of variable declaration

Variables are always created according to the same pattern in the declaration section of a source file section:

1. Start a declaration block with an appropriate keyword (e.g. VAR, VAR_GLOBAL – see Overview of all variable declarations (Page 4695)).
2. This is followed by the actual variable declarations (see figure); you can create as many of these as you wish. The order is arbitrary.
3. End the declaration block with END_VAR.
4. You can create further declaration blocks (also with the same keyword).

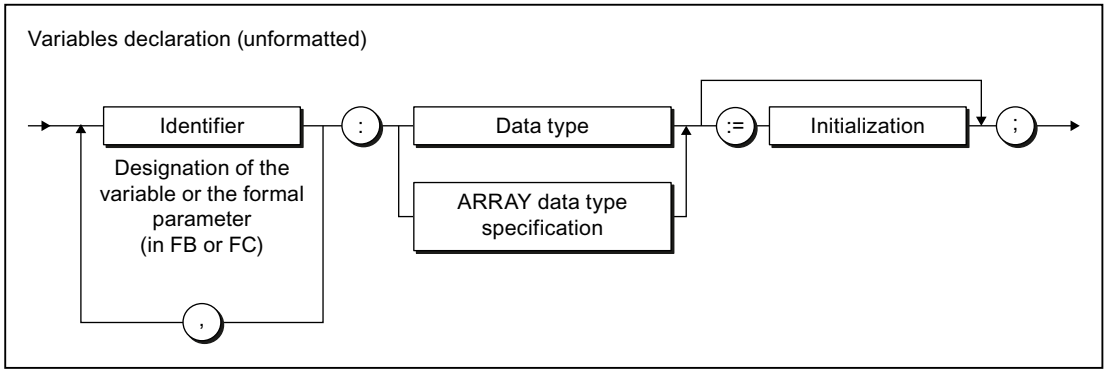


Figure 7-240 Syntax: Variable declaration

Note the following:

- The variable name must be an identifier, i.e. it can only contain letters, numbers or an underscore, but not special characters.
- The following are permissible as data types:
  - Elementary data types
  - UDT (user-defined data types)
  - System data types
  - TO data types
  - ARRAY data type specifications
  - Designation of a function block (instance declaration – see Calling functions and function modules (Page 4749)).
- You can assign initial values to the variables in the declaration statement. This is known as initialization (see Initialization of variables or data types (Page 4697)).

Deviations from the pattern presented are as follows:

- For constant declarations (a constant **must** be initialized with a value, see Constants (Page 4701)),
- For process image access (see Overview of all variable declarations (Page 4695)):
  - A variable declaration is not required for absolute process image access,
  - Initialization is not permitted for symbolic process image access.

Table 7-394 Examples of variable declarations

```
VAR CONSTANT
    PI : REAL := 3.1415;
END_VAR

VAR
    // Declaration of a variable ...
    var1 : REAL;
    // ... or if there are several variables of the same type:
    var2, var3, var4 : INT;
    // Declaration of a one-dimensional array:
    a1 : ARRAY[1..100] OF REAL;
    // Declaration of a character string:
    str1 : STRING[40];
END_VAR
```

Overview of all variable declarations

You declare the name, data type, and initial values of variables in the variable and parameter declarations. You always execute these declarations in the declaration sections of the following source file sections:

- Interface section
- Implementation section
- POU (program, function, function block, expression)

The source file section also determines which variables you can declare (see table), as well as their range.

For additional information about source file sections, refer to Breakdown of ST source file (Page 4668) and Source file sections (Page 4806).

Table 7-395 Keywords for declaration blocks

Keyword	Meaning	Declaration in the following declaration sections
VAR	Declaration of temporary or static variables See Variable model (Page 4831)	Any POU
VAR_GLOBAL	Declaration of unit variables See Variable model (Page 4831)	Interface section Implementation section

Keyword	Meaning	Declaration in the following declaration sections
VAR_IN_OUT	Variable declaration of in/out parameter; the POU accesses this variable directly (using a reference) and can change it immediately. See Defining functions (Page 4736), Defining function blocks (Page 4737), Defining methods (Page 4769)	Function Function block Expression Method [1]
VAR_INPUT	Variable declaration of input parameter, value is externally supplied and cannot be changed within the POU. See Defining functions (Page 4736), Defining function blocks (Page 4737), Defining methods (Page 4769)	Function Function block Expression Method [1]
VAR_OUTPUT	Variable declaration of output parameter; value is transmitted from the function block See Defining functions (Page 4736), Defining function blocks (Page 4737), Defining methods (Page 4769)	Function Function block Method [1]
VAR_TEMP	Declaration of temporary variables See Variable model (Page 4831)	Program Function Function block Method [1]
RETAIN	Declaration of retentive variables See Variable model (Page 4831)	Only as a supplement: • To VAR [2] in POU: – Program – Function block – Class [1] • To VAR_GLOBAL in the: – Interface section – Implementation section
CONSTANT	Declaration of constants See Constants (Page 4701)	Only as a supplement: • To VAR in POU: – Program – Function – Function block – Class [1 2] – Method [1] • To VAR_GLOBAL in the – Interface section – Implementation section
[1] Only with compiler option (Page 4623) "Permit object-oriented programming". [2] Only as of version V4.5 of the SIMOTION Kernel.		

Initialization of variables or data types

The assignment of initial values to the variables or data types within a declaration is optional; see Syntax of variables declaration (Page 4694) and Syntax of user-defined data types (Page 4676).

- If there is no initialization specified in the variable declaration, the compiler automatically assigns the initialization value specified in the data type declaration to the variables. The following applies for arrays (ARRAY): The array elements are assigned the initialization value of the basic data type.
- If there is no initialization specified in the data type declaration either, the compiler assigns the value of zero to the variables or data types.
  Exception:
  - For time data types: The initialization value for each data type.
  - For enumeration data types: 1st value of the enumeration.
  - For arrays (ARRAY): The array elements are assigned the initialization value of the basic data type.
  - For structures (STRUCT): The structure components are assigned the initialization value of the respective data type.

You preassign a variable or a user-defined data type with initial values by assigning a value (**:=**) after the data type specification.

- Assign the elementary data types (or data types derived from elementary data types) a constant expression in accordance with Figure *Syntax: constant expression*.
- Assign an array initialization list to an array (ARRAY) in accordance with the Figure *Syntax: Array initialization list*.
- Assign a structure initialization list to a structure (STRUCT) in accordance with Figure *Syntax: Structure initialization list* (even when the structure is used in other declarations).
  See also the explanation at the end of this section.
- Assign an enumeration element to an enumeration data type.

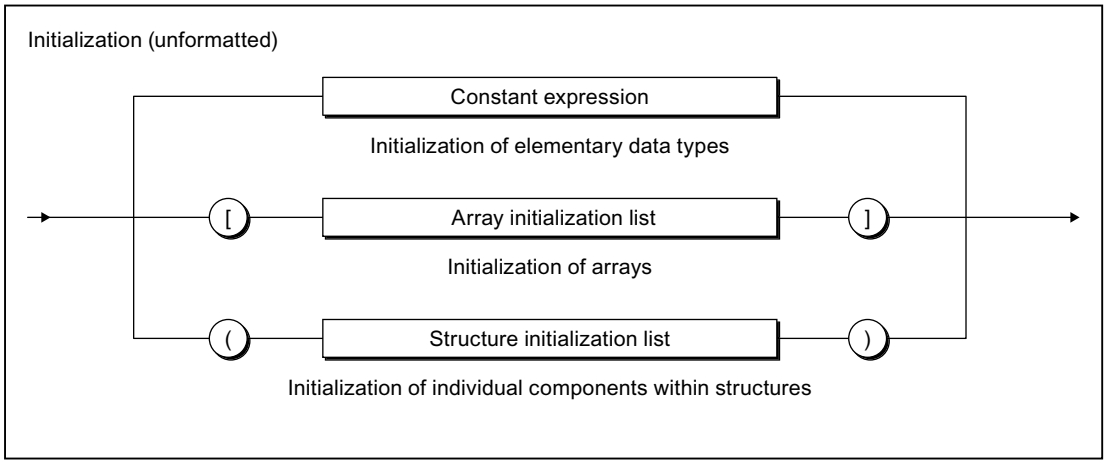


Figure 7-241 Syntax: Variable initialization

The initialization value assigned to a variable is calculated from the constant expression at the time of the compilation. For information about the syntax of the constant expression, see the figure titled *Syntax: Constant expression*.

Note that a variable list (a1, a2, a3, ... : INT := ...) can be initialized with a common value. In this case, you do not have to initialize the variables individually (a1 : INT := ... ; a2 : INT := ... ; etc.).

Note

The constant expressions used for initialization are calculated in the data type of the declared variables or in the declared data type.

Variables of a technology object data type cannot be assigned an initialization value. They are always initialized by the compiler with TO#NIL.

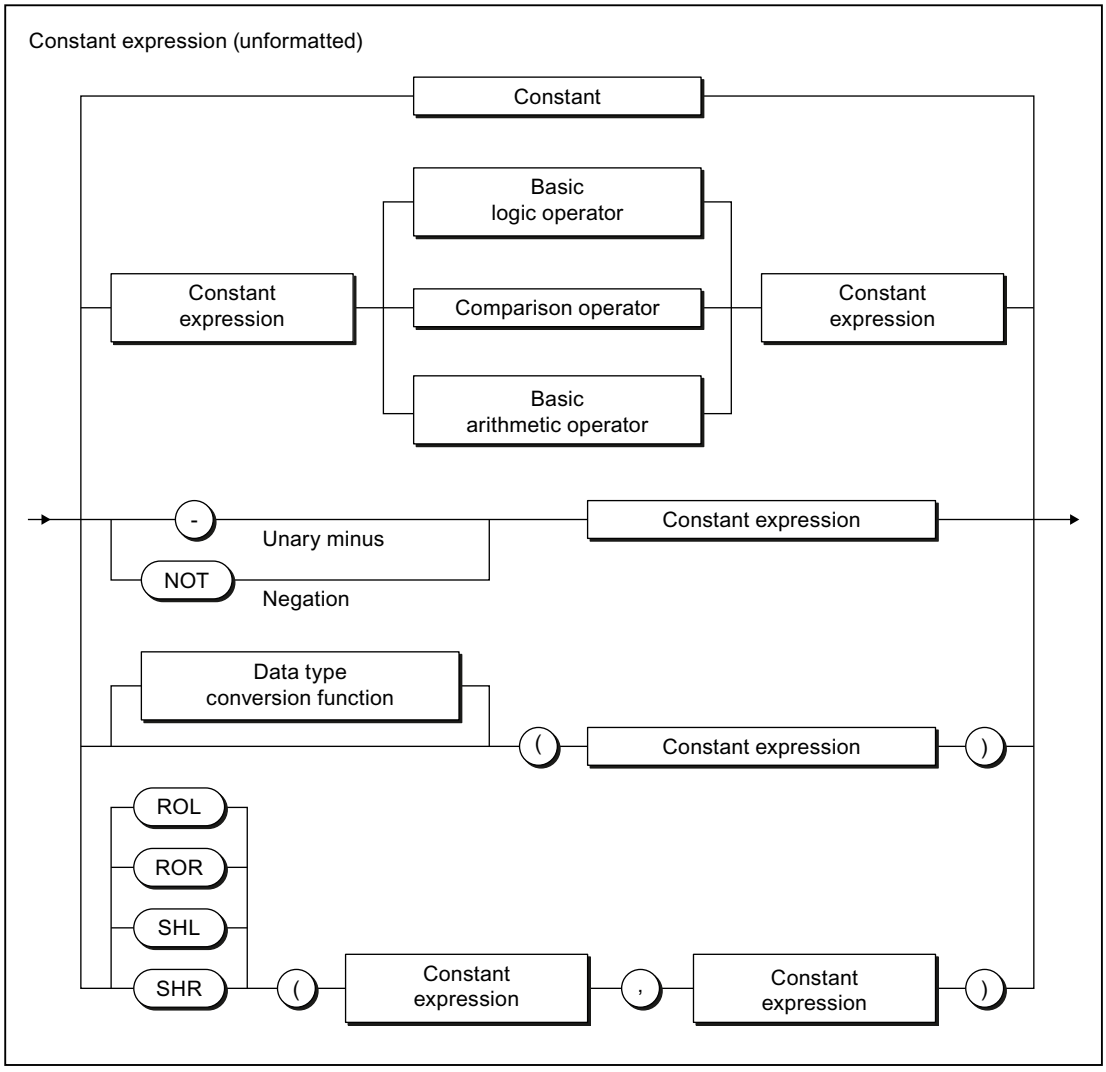


Figure 7-242 Syntax: Constant expression

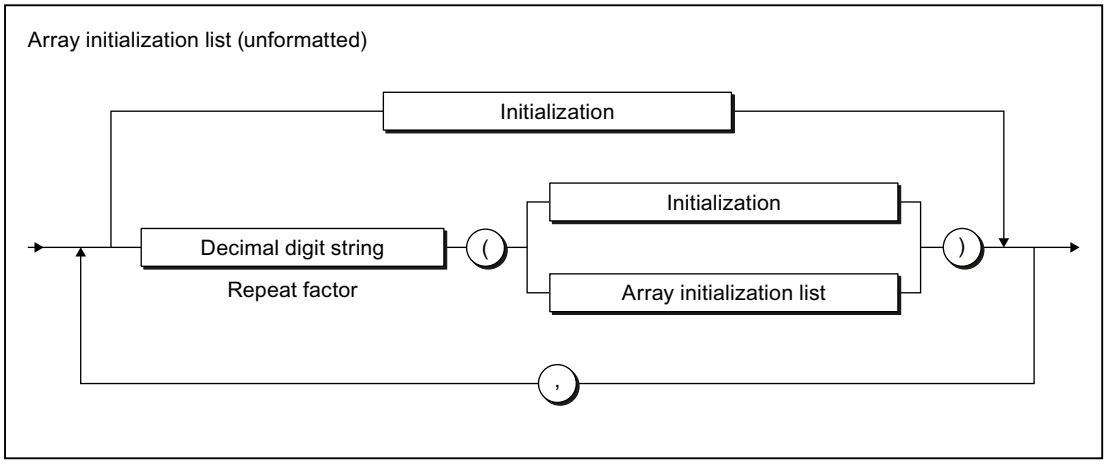


Figure 7-243 Syntax: Array initialization list

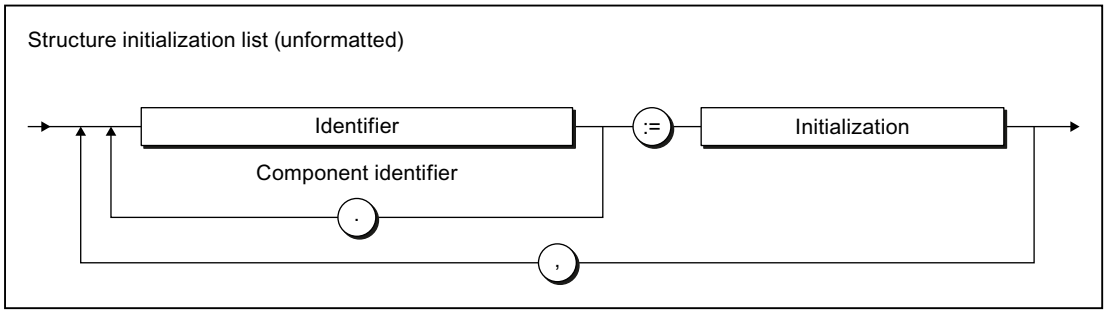


Figure 7-244 Syntax: Structure initialization list

Table 7-396 Examples of variable initialization

```
VAR
    // Declaration of a variable ...
    var1 : REAL := 100.0;
    // ... or if there are several variables of the same type:
    var2, var3, var4 : INT := 1;
    var5 : REAL := 3 / 2;
    var6 : INT := 5 * SHL(1, 4);
    myC1 : C1 := GREEN;
    array1 : ARRAY [0..4] OF INT := [1, 3, 8, 4, 0];
    array2 : ARRAY [0..5] OF DINT := [6 (7)];
    array3 : ARRAY [0..10] OF INT := [2 (2(3),3(1)),0];
                // is equivalent to [2(3),3(1),2(3),3(1)),0]
                // Initialization as follows:
                // Array elements 0, 1        with 3;
                // Array elements 2, 3, 4        with 1;
                // Array elements 5, 6        with 3;
                // Array elements 7, 8, 9        with 1;
                // Array element 10            with 0
    myAxis : PosAxis := TO#NIL;
END_VAR
```

Table 7-397 Examples of data type initialization

```
TYPE
    // Initialization of a derived data type
    type1 : REAL := 10.0;
    // Initialization of an enumeration data type
    cmyk_colour : (cyan, magenta, yellow, black) := yellow;
    // Initialization of structures
    var_rgb_colour : STRUCT
            red, green, blue : USINT := 255;// white
        END_STRUCT;
    new_colour : var_rgb_colour := (red := 0, blue := 0);// green
    myInt : INT := 9;
    myArray : ARRAY [0..5] OF myInt := [1, 2, 3];
                // Initialization as follows:
                // Array element 0            with 1;
                // Array element 1            with 2;
                // Array element 2            with 3;
                // Array elements 3, 4, 5     with 9
END_TYPE
```

Information about the circumstances when variables are initialized can be found in Section "Time of the variable initialization" (Page 4849).

Initialization of structures

To initialize a structure (data type STRUCT), assign to it a structure initialization list enclosed in brackets "(" ")". Assign the initialization value to the desired component in this initialization list, see the syntax diagram above.

The following methods can be used to initialize components of lower-level structures:

- You assign another structure initialization list enclosed in brackets to a component and then "nest" the lists inside one another.
- You aggregate the identifiers of the component, subcomponent, etc. as follows (use periods as separators): *comp_name.subcomp_name.subsubsomp_name* and assign the initialization value to this structure.

Note

Saving in an old project format:

Projects in which aggregated identifiers are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

You can use both notations within the same structure initialization list provided that you are initializing different components.

You can initialize arrays within structures by assigning an array initialization list enclosed in square brackets "[" "]" to the component. See example below:

Table 7-398 Examples of structure initialization

```
TYPE
    s_base : STRUCT
        i : INT;
        j : DINT;
    END_STRUCT
    s_deri : STRUCT
        x : INT;
        v_b : s_base := (i := 10);
        arr_b : ARRAY [0..2] OF s_base
           := [(i := 10, j := 15), (i := 12), (j := 13)];
    END_STRUCT
    s_up : STRUCT
        vup_d : s_deri := (x := 3);
        vup_a : ARRAY [0..1] OF s_deri;
    END_STRUCT
END_TYPE
VAR
    // Initialization with nested
    // structure initialization lists
    var_1 : s_up := (vup_d := (v_b := (i := 1) , x := 2));
    // Alternative with aggregated component identifiers
    var_2 : s_up := ( vup_d.v_b.i := 1, vup_d.x := 2);
    // Initialization of arrays of structures
    var3 : s_up := (vup_a := [ (v_b := (i := 5)), (v_b.i := 6) ] );
    var4 : ARRAY [0..7] OF s_up := [ 5((vup_d := (v_b := (i := 1)))),
                                     3((vup_d.v_b.i := 1)) ];
END_VAR
```

Constants

Constants are data with a fixed value that you cannot change during program runtime. Constants are declared in the same way as variables:

- In the declaration section of a POU for local constants (see Figure *Syntax: Constant block in a POU* and *syntax: Constant declaration*).
- In the interface or implementation section of the ST source file for unit constants (see Figure *Syntax: Unit constants in the interface or implementation section* and *syntax: Constant declaration*). You can import unit constants declared in the interface section into other ST source files (see Variable model (Page 4831)).

The source file section also determines the range of the constant declaration.

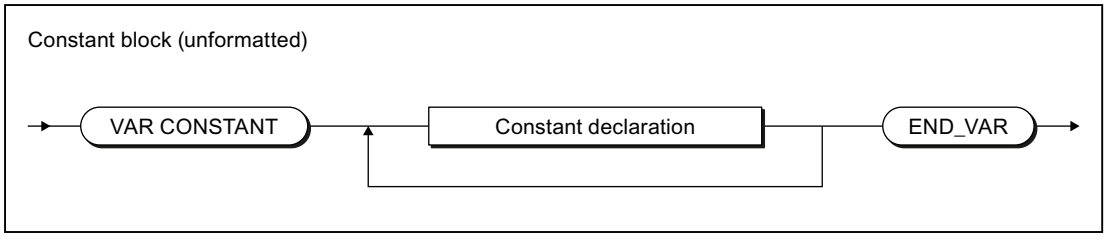


Figure 7-245 Syntax: Constant block in a POU

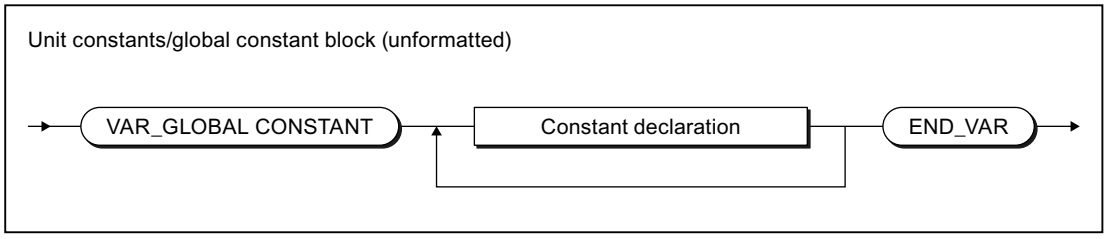


Figure 7-246 Syntax: Unit constants in interface or implementation section

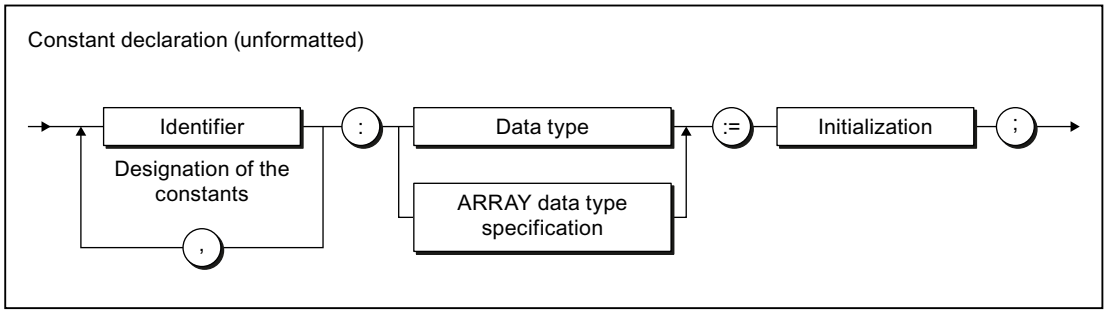


Figure 7-247 Syntax: Constant declaration

To initialize a constant, see "Initialization of variables or data types (Page 4697)". The value assigned to a constant is calculated from the constant expression at the time of compilation.

Table 7-399 Examples of constants

```
VAR CONSTANT
    PI            : REAL := 3.1415;
    intConst      : INT  := 10;
    sintConst     : SINT := 0;
    dintConst     : DINT := 10_000;
    timeConst     : TIME := TIME#1h;
    strConst      : STRING[40] := 'Example of a string';
    Two_PI        : REAL := 2 * PI;
END_VAR
```

7.2.4.6 Value assignments and expressions

You have no doubt already created value assignments with the character string :=. This may have been for a statement as part of an example (see table titled *Examples of statements* in Statements (Page 4670)) or when initializing variables in the declaration subsection of a source file module.

However, this is only a small range of the options available for formulating value assignments. This section of the manual now describes this important topic in detail using a large number of examples for illustration purposes.

See also

Notes on avoiding errors and on efficient programming (Page 4934)

Value assignments

Syntax of the value assignment

A value assignment is used to assign the value of an expression to a variable. The previous value is overwritten. Before a value can be correctly assigned, a variable must be declared in the declaration section (see Syntax of variable declaration (Page 4694)).

As shown in the following syntax diagram, the expression is evaluated on the right side of the assignment operator :=. The result is stored in the variable whose name is on the left side of the assignment operator (target variable). All target variables supported from a formal viewpoint are shown in the figure.

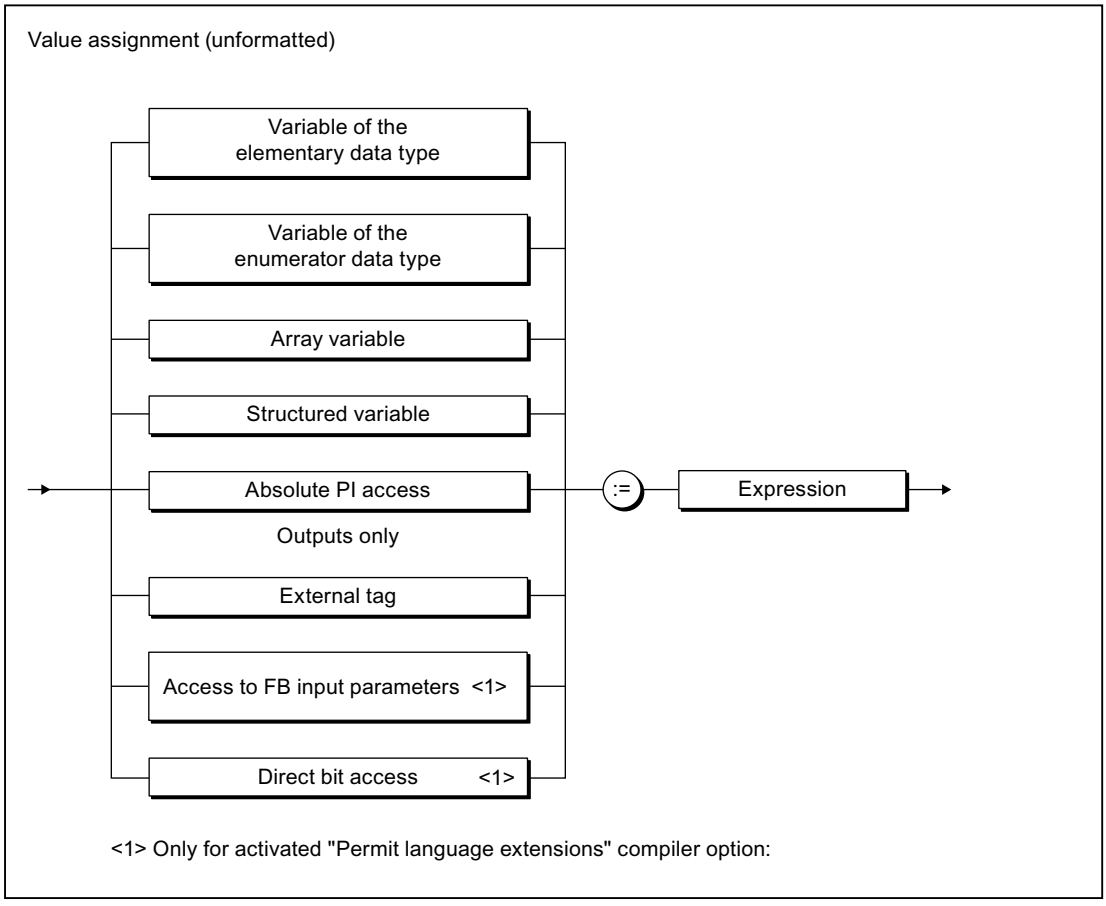


Figure 7-248 Syntax: Value assignment

The following contains explanations and examples for the left side of the value assignment:

- Value assignments with variables of an elementary data type (Page 4704) ,
- Value assignments with variables of the derived enumerator data type (Page 4707)
- Value assignments with variables of the derived ARRAY data type (Page 4707)
- Value assignments with variables of the derived STRUCT data type (Page 4708)
- Value assignments with absolute PI access (to addresses of the process image), see: Absolute access to the fixed process image of the BackgroundTask (absolute PI access) (Page 4887).

How the right side of a value assignment, i.e. an expression, is formed, is described in Expressions (Page 4709).

Value assignments with variables of an elementary data type

An expression with an elementary data type (Page 4672) can be assigned to a variable when one of the following conditions is fulfilled:

- Expression and target variable have the same data type.
  Note the following information on the STRING data type (Page 4704).
- The data type of the expression can be implicitly converted to the data type of the target variable (see Conversion of elementary data types (Page 4732) and *Functions for the conversion of numerical data types and bit data types* in the *SIMOTION Basic Functions* Function Manual).

Examples

```
elemVar         := 3*3;
elemVar         := elemVar1;
```

See also

Value assignments with variables of a bit data type (Page 4705)

Value assignments with variables of the STRING elementary data type

Assignments between variables of the STRING data type

There are no restrictions to assignments between variables of the STRING data type (character strings) that have been declared with different lengths. If the declared length of the target variable is shorter than the current length of the assigned character string, the character string is truncated to the length of the target variable.

Exception: The following applies for an in/out assignment (parameter transfer to an in/out parameter): The declared length of the assigned variable (actual parameter) must be greater than or equal to the declared length of the target variable (formal in/out parameter). See Parameter transfer to in/out parameters (Page 4751).

Please also refer to Syntax diagram of STRING data type (Page 4672):

Examples:

```
string20 := 'ABCDEFG';
string20 := string30;
```

Access to elements of a string

The individual elements of a string can be addressed in the same way as the elements of an array [1..n]. These elements are converted implicitly to the elementary data type BYTE. In this way assignments between string elements and variables of the BYTE data type are possible.

Examples:

```
byteVar := string20[5];
string20[10] := byteVar;
```

The following special cases have to be taken into account:

1. When assigning a variable of the BYTE data type to a string element (e.g. `stringVar[n] := byteVar`):
   - The string element to which the value is to be assigned lies outside of the declared length of the string:
     The string remains unchanged, TSI#ERRNO is set to 1.
   - The string element to which the value is to be assigned lies outside of the assigned length of the string (n > LEN(stringVar)), but within the declared length:
     The length of the string is adjusted, the string elements between LEN(stringvar) and n are set to $00.
2. When assigning a string element to a variable of the BYTE data type (`byteVar := stringVar[n]`):
   - The string element to which the variable is to be assigned lies outside of the assigned length of the string (n > LEN(stringVar)):
     The variable is set to 16#00, TSI#ERRNO to 2.

Editing strings

Various system functions are available for the editing of strings, such as the joining of strings, replacement and extraction of characters, see *SIMOTION Basic Functions* Function Manual.

Converting between numbers and strings

Various system functions are available for conversion between variables of numeric data types and strings; see Converting elementary data types (Page 4732) and the *SIMOTION Basic Functions* Function Manual.

Value assignments with variables of a bit data type

Access to individual bits of a bit data type variable

You can also access the individual bits of a variable of data type BYTE, WORD or DWORD:

- With standard functions, see SIMOTION Basic Functions Function Manual:
  You can read, write or invert any bit of a bit string with the functions _getBit, _setBit and _toggleBit.
  You can specify the number of the bit via a variable.
- With direct bit access:
  You can define the bit of the variable that you want to access as a constant, via a separate point behind the variable.
  You can only specify the number of the bit via a constant.
  To be able to use this option, you must activate the "Permit language extensions" compiler option, see Global compiler settings (Page 4623) and Local compiler settings (Page 4626).

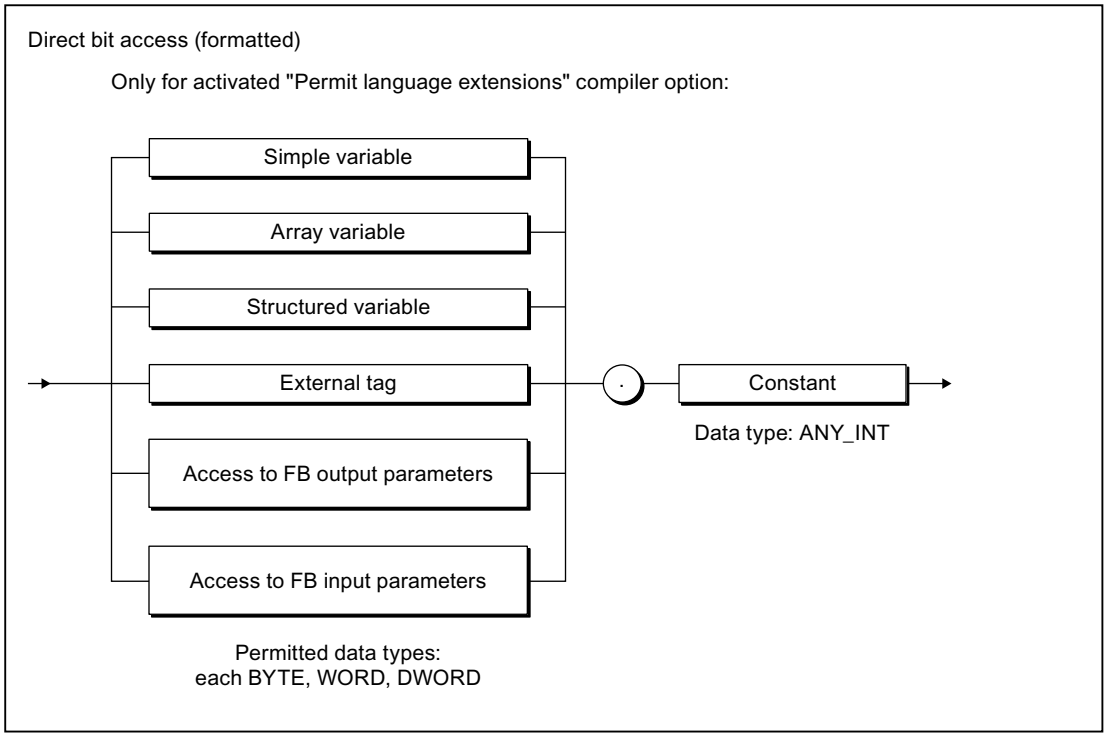


Figure 7-249 Syntax: Direct bit access

Table 7-400 Example of direct bit access

```
// Only with compiler option "Permit language extensions"
FUNCTION f : VOID
    VAR CONSTANT
        BIT_7 : INT := 7;
    END_VAR
    VAR
        dw : DWORD;
        b  : BOOL;
    END_VAR
    b := dw.BIT_7;  // Access to Bit 7
    b := dw.3;      // Access to Bit 3
//  b := dw.33;     // Compilation error:
                    // Bit 33 not permitted.
    dw.BIT_7 := b   // Access to Bit 7
    dw.3     := NOT dw.3  // Access to Bit 3
END_FUNCTION
```

Note

The access to bits of an I/O variable or system variable can be interrupted by other tasks. There is therefore no guarantee of consistency.

Editing variables of the bit data types

You can:

1. Combine several variables of the same data type into one variable of a higher-level data type (e.g. two variables of the BYTE data type into one of the WORD data type). Various system functions are available for this, e.g. WORD_FROM_2BYTE.
2. Split one variable into several variables of a lower-level data type (e.g. one variable of the DWORD data type into four of the BYTE data type). Various system functions are available for this, e.g. DWORD_TO_4BYTE.
3. Rotate or shift the bits within a variable. The bit string standard functions ROL, ROR, SHL and SHR are available for this.

These system functions and system function blocks are described in the *SIMOTION Basic Functions* Function Manual.

Logic operators

Variables of the bit data types can be combined with logic operators; see Logic expressions and bit-serial expressions (Page 4717).

Value assignments with variables of the derived enumerator data type

Each expression and each variable of the derived enumerator data type (see also: Derived enumerator data type (Page 4681)) can be assigned to another variable of the same type.

```
type1        := BLUE;
```

Value assignments with variables of the derived ARRAY data type

An array consists of several dimensions and array elements, all of the same type (see also: Derived ARRAY data type (Page 4678)).

There are various ways to assign arrays to variables. You can assign complete arrays, individual elements, or parts of arrays:

- A complete array can be assigned to another array if both the data types of the components and the array limits (the smallest and largest possible array indices) are the same. Valid assignments are:

```
array_1     := array_2;
```

- An individual array element is addressed by the array name followed by the index value in square brackets. An index must be an arithmetic expression of the data type SINT, USINT, INT, UINT or DINT.

```
elem1        := array [i];
array_1 [2]  := array_2 [5];
array [j]    := 14;
```

- A value assignment for a valid subarray can be obtained by omitting a pair of square brackets for each dimension of the array, starting at the right. This addresses a partial area of the array whose number of dimensions is equal to the number of remaining indices (see example below).
  Consequently, you can reference rows and individual components within a matrix but not closed columns (closed in the sense of from...to). Valid assignments are:

```
matrix1 [i]   := matrix2 [k];
array1        := matrix2 [k];
```

Value assignments with variables of the derived STRUCT data type

Variables of a user-defined data type that contain STRUCT data type specifications are called structured variables (see also Derived STRUCT data type (Page 4682)). They can either represent a complete structure or a component of this structure.

Valid parameters for a structure variable are:

```
struct1             // Identifier for a structure
struct1.elem1       // Identifier for a structure component
struct1.array1      // Identifier of a simple array
                    //within a structure
struct1.array1[5]   // Identifier of an array component
                    //within a structure
```

There are two ways to assign structures to variables. You can reference complete structures or structure components:

- A complete structure can only be assigned to another structure if the data type and the name of both structure components match.
  A valid assignment is:

```
struct1 := struct2;
```

- You can assign a type-compatible variable, a type-compatible expression or another structure component to each structure component.
  Valid assignments are:

```
struct1.elem1       := Var1;
struct1.elem1       := 20;
struct1.elem1       := struct2.elem1;
struct1.array1      := struct2.array1;
struct1.array1[10]  := 100;
```

Note

You also use structured variables in the *FBInstanceName.OutputParameter* format, e.g. *myCircle.circumference* to access the output variables of a function block, i.e. the result of the function block. For more detailed information about function blocks, refer to the explanations in Defining functions (Page 4736) and Defining function blocks (Page 4737).

A further application of structured variables is to access TO variables and the variables of the basic system.

Expressions

An expression represents a value that is calculated when the program is compiled or executed. It consists of operands (e.g. constants, variables or function values) and operators (e.g. *, /, +, -).

The data types of the operands and the operators involved determine the expression type.

ST uses the following types of expression:

- Arithmetic expressions (Page 4712)
- Relational expressions (Page 4715)
- Logic expressions and bit-serial expressions (Page 4717)

Result of an expression

The result of an expression can be:

- Assigned to a variable
- Used as a condition for a control statement
- Used as a parameter for a function or function block call.

The data type of the result of an arithmetical or bit-serial expression is determined by the data types of the operands. The data type used is the lowest common data type to which both operands can be implicitly converted.

An expression value can only be assigned to a variable (or a parameter of a function or function block) in the following cases:

- The expression calculated and the variable to be assigned are of the same data type.
- The data type of the calculated expression can be implicitly converted to the data type of the variable to be assigned.

For more information on this error source and its solution, see *SIMOTION Basic Functions* Function Manual.

Note

Expressions containing only the following elements can be used for variable initialization and index specification in ARRAY declarations (for initialization expressions – see Figure *Syntax: Constant expression* in Initialization of variables or data types (Page 4697)):

- Constants
- Basic arithmetic operations
- Logic and relational operations
- Bit string standard functions

The constant expressions used for initialization are calculated in the data type of the declared variables or in the declared data type.

Interpretation order of an expression

The interpretation order of an expression depends on the following:

- The priority of the operators used,
- The left-to-right rule,
- The use of parentheses (for operators of the same priority).

Expressions are processed according to specific **rules**:

- Operators are executed according to priority (see table in Operator priority (Page 4718)).
- Operators of the same priority are executed from left to right.
- A minus symbol in front of an identifier denotes multiplication by -1.
- An arithmetic operator cannot be followed immediately by another. The expression *a * -b* is therefore invalid, but *a * (-b)* is allowed.
- Parentheses override the operator priority order, i.e. parentheses have the highest priority.
- Expressions in parentheses are treated as individual operands and are always evaluated first.
- The number of opening parentheses must equal the number of closing parentheses.
- Arithmetic operations cannot be used on characters or logic data. For this reason, expressions such as *(n<=0) + (n<0)* are invalid.

Table 7-401 Examples of expressions

```
testVar          // Operand
A AND (B)        // Logical expression
A AND (NOT B)    // Logical expression with negation
(C) < (D)        // Comparison expression
3+3*4/2          // Arithmetic expression
```

Operands

Definition

Operands are objects which can be used to formulate expressions. Operands can be represented by the syntax diagram:

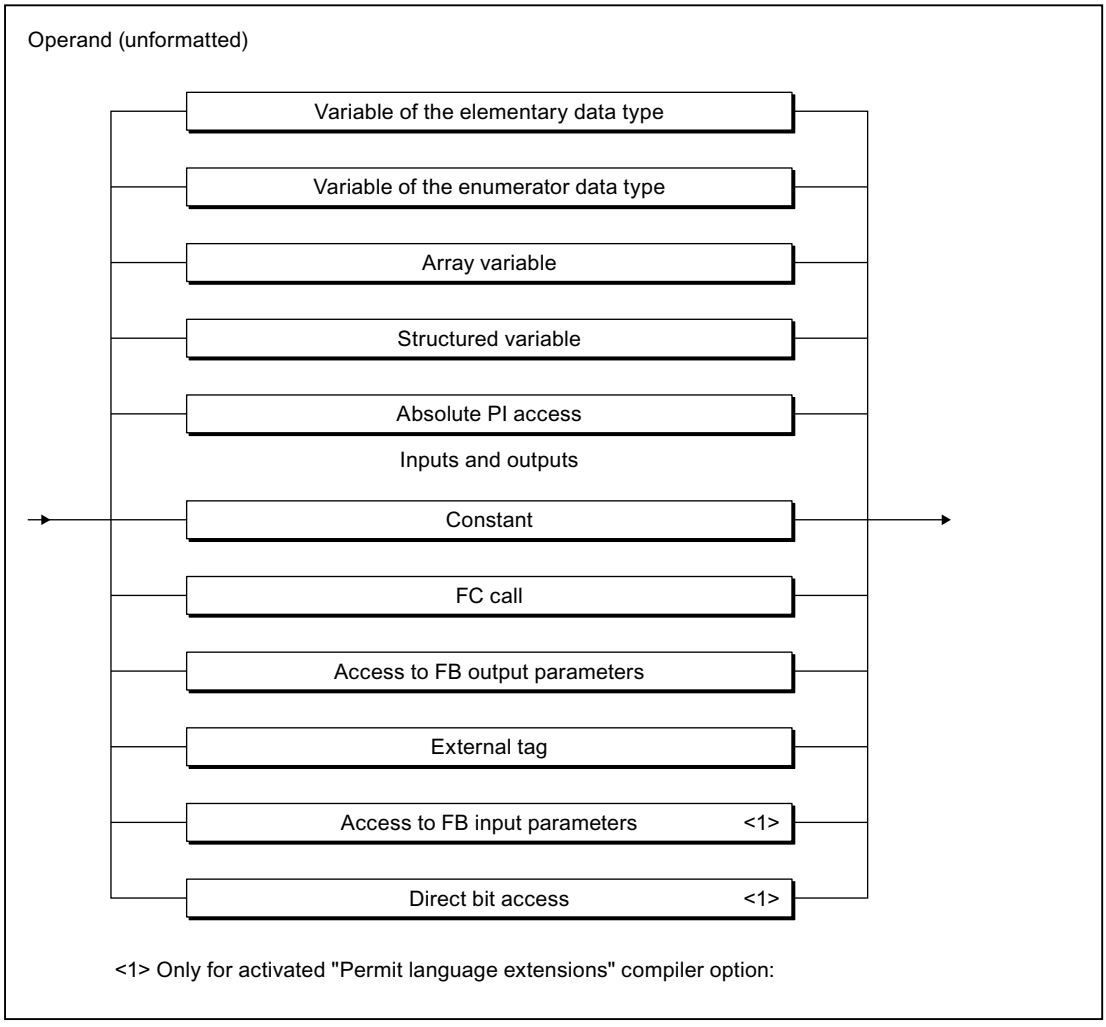


Figure 7-250 Syntax: Operand

Table 7-402 Examples of operands

```
intVar
5
%I4.0
PI
NOT TRUE
axis1.motionStateData.actualVelocity
```

Arithmetic expressions

An arithmetic expression is an expression formed with arithmetical operators. These expressions allow numerical data types to be processed.

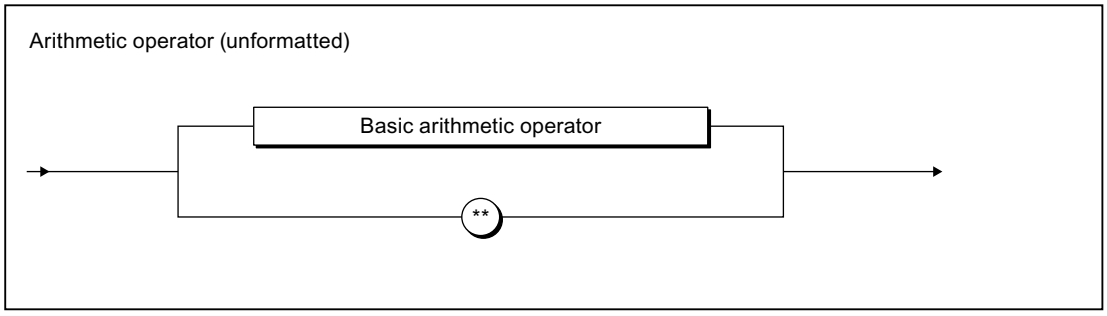


Figure 7-251 Syntax: Arithmetic operator

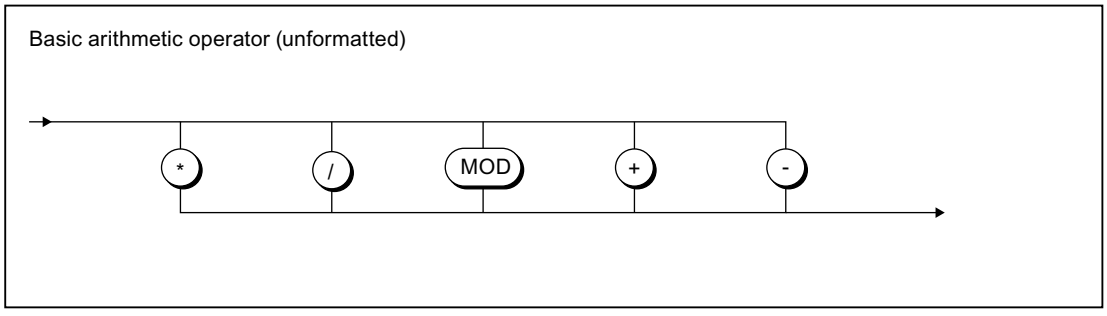


Figure 7-252 Syntax: Basic arithmetic operator

The following table shows for each arithmetic operation:

- The arithmetic operator
- The permitted data types of the operands
- The data type of the result.

Some of the General data types (Page 4674) are used here.

Note

Further operations are possible with standard numeric functions, see *Standard numeric functions* in the *SIMOTION Basic Functions* Function Manual.

It is recommended to enclose negative numbers in parentheses, even in cases where it is not absolutely necessary, in order to enhance legibility.

The arithmetic operators are processed in accordance with their rank (Page 4718).

Table 7-403 Arithmetic operators

Instruction	Operator	Data type		
		1st operand	2nd operand	Result[1]
Exponential (See also EXPT function)	**	ANY_REAL[2]	ANY_REAL	ANY_REAL[3]
Unary minus	–	ANY_NUM	(None)	ANY_NUM
Multiplication	*	ANY_NUM	ANY_NUM	ANY_NUM
		ANY_BIT[4]	ANY_BIT[4]	ANY_BIT
		TIME	ANY_NUM	TIME
Division	/	ANY_NUM	ANY_NUM[5]	ANY_NUM
		ANY_BIT[4]	ANY_BIT[4] [5]	ANY_BIT
		TIME	ANY_NUM[5]	TIME
		TIME	TIME[5]	UDINT
Modulo division	MOD	ANY_INT	ANY_INT[5]	ANY_INT
		ANY_BIT[4]	ANY_BIT[4] [5]	ANY_BIT
Addition	+	ANY_NUM	ANY_NUM	ANY_NUM
		ANY_BIT[4]	ANY_BIT[4]	ANY_BIT
		TIME	TIME	TIME[6]
		TOD	TIME	TOD[6]
		DT	TIME	DT[7]
Subtraction	–	ANY_NUM	ANY_NUM	ANY_NUM
		ANY_BIT[4]	ANY_BIT[4]	ANY_BIT
		TIME	TIME	TIME
		TOD	TIME[8]	TOD
		DATE	DATE	TIME[9]
		TOD	TOD	TIME[9]
		DT	TIME	DT
		DT	DT	TIME[9]

1) The data type of the result (unless explicitly stated) is the lowest common data type to which both operands can be implicitly converted.

2) The 1st operand must be greater than zero.
Exceptions as of version V4.1 of the SIMOTION Kernel:
– If the 2nd operand is an integer, the 1. operand can be less than zero.
– If the 2nd operand is positive, the 1st operand can be equal to zero.
The following applies up to version V4.0 of the SIMOTION Kernel: If the 1st operand is equal to zero, an error message can be caught with ExecutionFaultTask.

3) Data type of the 1st operand.

4) Other than BOOL data type. The calculation is made using the unsigned integer of the same bit width.

5) The 2nd operand must not be equal to zero.

6) Addition, possibly with overflow.

7) Addition with date correction.

8) Restriction of TIME to TOD before calculation.

9) These operations are based on the modulo of the maximum value of the TIME data type.

Note

If the limits of the value range are exceeded in operations with variables of the general ANY_REAL data type, the result contains the equivalent bit pattern according to IEEE 754.

In order to establish whether the value range was exceeded in the operation, you can verify the result using the function *_finite* (see *SIMOTION Basic Functions* Function Manual).

Examples of arithmetic expressions

Examples of arithmetic expressions with numbers

Assuming that *i* and *j* are integer variables (e.g. of data type INT) with the values of 11 and -3 respectively, some example integer expressions and their corresponding values are presented below:

Expression	Value
`i + j`	`8`
`i - j`	`14`
`i * j`	`-33`
`i MOD j`	`-2`
`i / j`	`-3`

Examples of valid arithmetic expressions with time specifications

Assume the following variables:

Variables	Content	Data type
`t1`	`T#1D_1H_1M_1S_1MS`	`TIME`
`t2`	`T#2D_2H_2M_2S_2MS`	`TIME`
`d1`	`D#2004-01-11`	`DATE`
`d2`	`D#2004-02-12`	`DATE`
`tod1`	`TOD#11:11:11.11`	`TIME_OF_DAY`
`tod2`	`TOD#12:12:12.12`	`TIME_OF_DAY`
`dt1`	`DT#2004-01-11-11:11:11.11`	`DATE_AND_TIME`
`dt2`	`DT#2004-02-12-12:12:12.12`	`DATE_AND_TIME`

Some expressions with these variables and their values are shown in the example.

Expression	Value
`t1 + t2`	`T#3D_3H_3M_3S_3MS`
`dt1 + t1`	`DT#2004-01-12-12:12:12.111`
`t1 - t2`	`T#48D_16H_1M_46S_295MS`
`t1 * 2`	`T#2D_2H_2M_2S_2MS`
`t1 / 2`	`T#12H_30M_30S_500MS`

```
DATE_AND_TIME_TO_TIME_OF_DAY(dt1)     TOD#11:11:11.110
DATE_AND_TIME_TO_DATE(dt1)            D#2004-01-11
```

Relational expressions

Definition

A relational expression is an expression of the BOOL data type formed with relational operators (see figure).

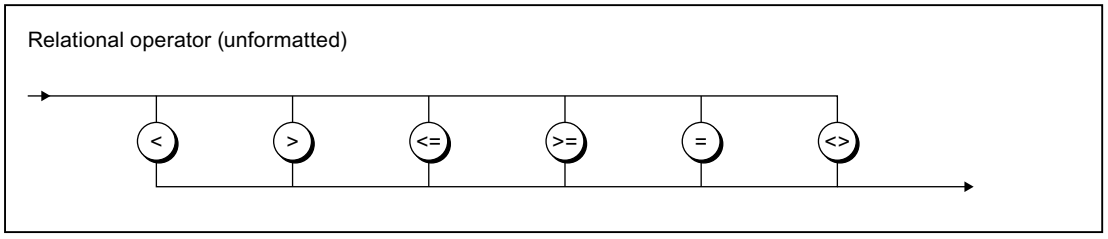


Figure 7-253 Syntax: Relational operators

Relational operators compare the values of 2 operands (see table) and return a Boolean value as result.

1st Operand Operator 2nd Operand -> Boolean value

Table 7-404 Meaning of relational operators

Operator	Meaning
>	1. operand is **greater than** the 2nd operand
<	1. operand is **less than** the 2nd operand
>=	1. operand is **greater than or equal to** the 2nd operand
<=	1. operand is **less than or equal to** the 2nd operand
=	1. operand is **equal to** the 2nd operand
<>	1. operand is **not equal to** the 2nd operand

The result of the relational expression is:

- 1 (TRUE), when the comparison is satisfied
- 0 (FALSE), when the comparison is not satisfied.

The following table shows permissible combinations of the data types for the two operands and relational operators.

Table 7-405 Relational expressions: Permissible combinations of the data types and relational operators

Data type		Permissible relational operators
1. Operand	2. Operand	
ANY_NUM	ANY_NUM[1)]	<, >, <=, >=, =, <>
ANY_BIT	ANY_BIT	<, >, <=, >=, =, <>
DATE	DATE	<, >, <=, >=, =, <>

Data type		Permissible relational operators
1. Operand	2. Operand	
TIME_OF_DAY (TOD)	TIME_OF_DAY (TOD)	<, >, <=, >=, =, <>
DATE_AND_TIME (DT)	DATE_AND_TIME (DT)	<, >, <=, >=, =, <>
TIME	TIME	<, >, <=, >=, =, <>
STRING	STRING[2)]	<, >, <=, >=, =, <>
Enumerator data type	Enumerator data type[3)]	=, <>
ARRAY	Field (ARRAY)[3)]	=, <>
Structure (STRUCT)	Structure (STRUCT)[3)]	=, <>

1) The comparison is made in the lowest common data type to which both operands can be implicitly converted.

2) Variables of the STRING data type can be compared irrespective of the declared length of the string. To compare two variables of the STRING data type with different lengths, the shorter character string is expanded to the length of the longer character string by inserting $00 on the right-hand side. The comparison starts from left to right and is based on the ASCII code of the respective characters. Example: 'ABC' < 'AZ' < 'Z' < 'abc' < 'az' < 'z'.

3) Data type of the 1. operand.

Relational expressions and variables or constants of the BOOL data type can be combined with logic operators to form logic expressions (see Logic expressions and bit-serial expressions (Page 4717)). This enables the implementation of queries such as *If a < b and b < c, then ...*.

Note

Relational operators have a higher priority than logic operators in an expression (see Operator priority (Page 4718)). Therefore the operands of a relational expression must be placed in brackets if they themselves are logic expressions or bit-serial expressions.

Note that errors can occur when comparing REAL or LREAL variables (also the corresponding system variables, e.g. axis position).

Table 7-406 Examples of relational expressions

```
IF A = 2 THEN
    ; //...
END_IF;

var_1 := B < C;          // var_1 of BOOL data type

IF D < E OR var_2 THEN  // var_2 of BOOL data type
    ; // ...
END_IF;

var_3 := 0 < F < 10      // var_3 of BOOL data type, TRUE value
    // The calculation is from left to right
    // in the following form: (0 < F) < 10
    // Initially 0 < F is calculated.
    // The result is from the BOOL data type (FALSE or TRUE)
    // and is compared with BYTE#10.
    // Equivalent expression:
var_3 : = BOOL_TO_BYTE (0 < F) < BYTE#10
```

Logic expressions and bit-serial expressions

Definition

With the logic operators AND, &, XOR, and OR, it is possible to combine operands and expressions of the general data type ANY_BIT (BOOL, BYTE, WORD, or DWORD).

With the logic operator NOT it is possible to negate operands and expressions of data type ANY_BIT.

The table provides information about the available operators:

Table 7-407 Logic operators

Instruction	Operator	1. Operand	2. Operand	Result[1]
Negation	NOT	ANY_BIT	-	ANY_BIT
Conjunction	AND or &	ANY_BIT	ANY_BIT	ANY_BIT
Exclusive disjunction	XOR	ANY_BIT	ANY_BIT	ANY_BIT
Disjunction	OR	ANY_BIT	ANY_BIT	ANY_BIT
[1] The data type of the result is determined by the most powerful data type of the operands.				

The expression is designated

- a **logic expression**, if only operands of data type BOOL are used.
  The operators have the effect on the operands stated in the following truth table.
  The result of a logic expression is 1 (TRUE) or 0 (FALSE).
- a **bit-serial expression**, if operands of data type BYTE, WORD, or DWORD are used.
  The operators have the effect on individual bits of the operands stated in the following truth table.

Table 7-408 Truth table of the logic operators

Operands (data type BOOL)		Result (data type BOOL)				
a	b	NOT a	NOT b	a AND b a & b	a XOR b	a OR b
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	1

Examples

Expression (let n = 10)	Value
`(n>0) AND (n<20)`	`TRUE`
`(n>0) AND (n<5)`	`FALSE`
`(n>0) OR (n<5)`	`TRUE`

```
(n>0) XOR (n<20)                          FALSE
NOT ((n>0) AND n<20))                     FALSE
```

Expression	Value
2#01010101 AND 2#11110000	2#01010000
2#01010101 OR 2#11110000	2#11110101
2#01010101 XOR 2#11110000	2#10100101
NOT 2#01010101	2#10101010

Expression in query (let value1 be 2#01, let value2 be 2#11)

```
IF (value1 AND value2) = 2#01 THEN...
```

Condition returns TRUE, because bit-serial expression returns 2#01.

Priority of operators

Some general rules for the formulation of expressions were described in Expressions (Page 4709). The table shows you the priority of the individual operators within an expression.

Instruction	Symbol	Priority
Parentheses	**(**Expression**)**	Highest
Function evaluation	**Identifier (**argument list**)** e.g. LN(a), EXPT (a,b), etc.	
Negation Complement	– **NOT**	
Exponentiation	**	
Multiplication Division Modulo	* / **MOD**	
Addition Subtraction	**+** **–**	
Comparison	**<, >, <=, >=**	
Equal Not equal	= <>	
Boolean AND	**&, AND**	
Boolean EXCLUSIVE OR	**XOR**	
Boolean OR	**OR**	Lowest

7.2.4.7 Control statements

Few source file sections can be programmed such that all statements are executed in sequence from start to end. Usually, some statements will be executed only if a condition is true (alternatives) and some will be executed repeatedly (loops). Program control statements within a source file section are the means for accomplishing this.

IF statement

Description

The IF statement is a conditional statement. It specifies one or more options and selects one (or none) of its statement sections for execution.

The specified logic expressions are evaluated when the conditional statement is executed. If the value of an expression is TRUE, the condition is fulfilled, if the value is FALSE, it is not fulfilled.

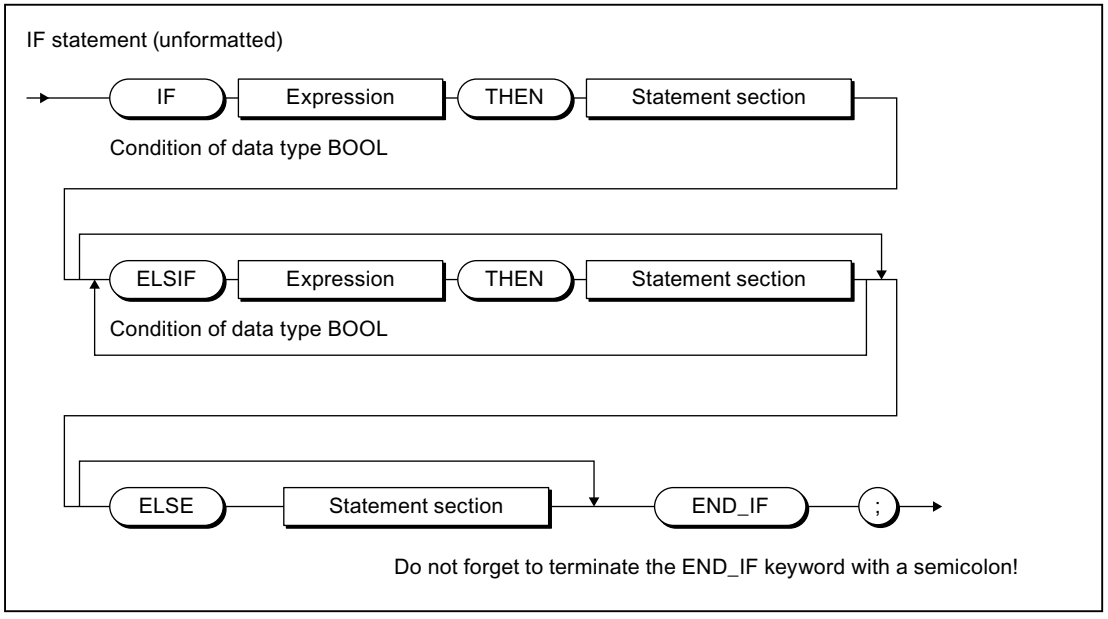


Figure 7-254 Syntax: IF statement

Sequence of execution

The IF statement is processed according to the following rules:

1. If the value of the first expression is TRUE, the statement section after the THEN is executed. The program is subsequently resumed after the END_IF.
2. If the value of the first expression is FALSE, the expressions in the ELSIF branches are evaluated. If a Boolean expression in one of the ELSIF branches is TRUE, the statement section following THEN is executed.
   The program is subsequently resumed after the END_IF.
3. If none of the Boolean expressions in the ELSIF branches is TRUE, the sequence of statements after the ELSE is executed (or, if there is no ELSE branch, no further statements are executed). The program is subsequently resumed after the END_IF.

Any number of ELSIF statements can be programmed.

Note that there may not be any ELSIF branches and/or ELSE branch. This is interpreted in the same way as if the branches existed with no statements.

Note

An advantage of using one or more ELSIF branches rather than a sequence of IF statements is that the logic expressions following a valid expression are no longer evaluated. This helps to reduce the processing time required for the program and to prevent execution of unwanted program routines.

Example

The following example illustrates the use of the IF statement:

Example of the IF statement

```
IF A = B THEN
    n:= 0;
END_IF;

IF temperature < 5.0 THEN
    %Q0.0 := TRUE;
ELSIF temperature > 10.0 THEN
    %Q0.2 := TRUE;
ELSE
    %Q0.1 := TRUE;
END_IF;
```

CASE statement

Description

The CASE statement is used to select 1 of n program sections.

This selection determines a selection expression (selector):

- Expression of general data type ANY_INT
- Variable of an enumeration data type (enumerator)

The selection is made from a list of values (value list), whereby a section of the program is assigned to each value or group of values.

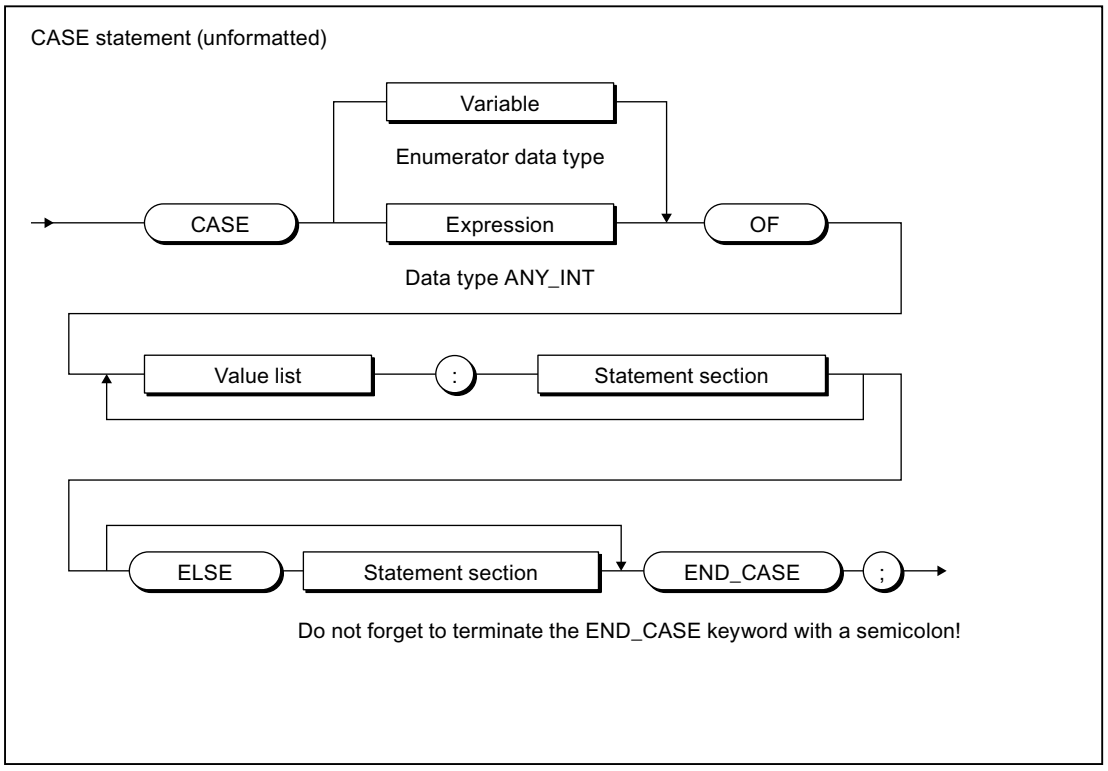


Figure 7-255 Syntax: CASE statement

Sequence of execution

The CASE statement is processed according to the following rules:

1. The selection expression (selector) is calculated. It must return a value of general data type ANY_INT (integer) or an enumeration data type.
2. Then a check is performed to determine whether the selector value is contained in the value list. Each value in the list represents one of the allowed values for the selection expression.
3. If a match is found, the program section assigned in the list is executed.
4. The ELSE branch is optional. It is executed if no match is found.
5. If the ELSE branch is missing and no match is found, the program is resumed after END_CASE.

Value list

The value list contains the allowed values for the selection expression.

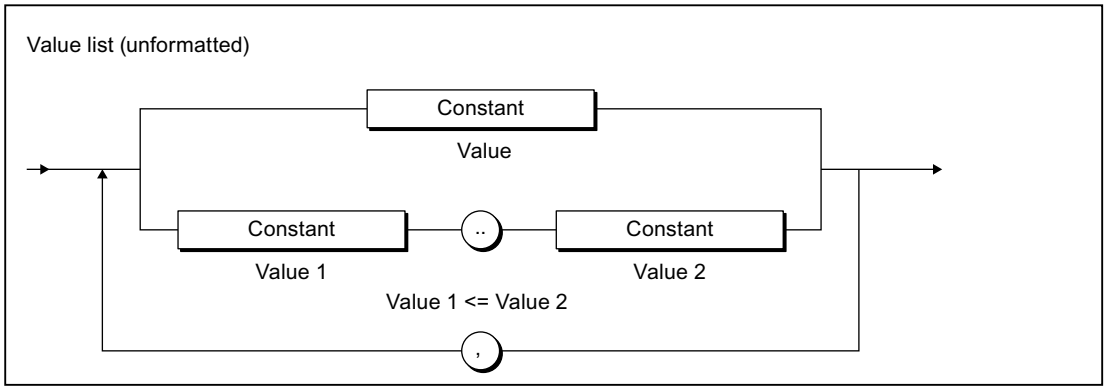


Figure 7-256 Syntax: Value list

Note the following when formulating the value list:

- Each value list can begin with a constant (*value*), a constant list (*value1, value2, value3, etc.*) or a constant range (*value1 to value2*).
- Values in the value list must be integer constants or elements of the enumeration data type of the selector.

Note

A value should only occur once in the value lists of a CASE statement.

In the event of multiple occurrence of a value, the compiler will issue an alarm, and only the section of the statement corresponding to the value list in which the value occurred first is executed.

Example

The following example illustrates the use of the CASE statement:

Example of the CASE statement

```
CASE intVar OF
    1    : a := 1;
    2,3  : b := 1;
    4..9 : c := 1; d := 2;
ELSE
    e := 5;
END_CASE;
```

FOR statement

Description

A FOR statement or a repeat statement executes a series of statements in a loop, whereby values are assigned to a variable (a count variable) on each pass. The count variable must be a local variable of type SINT, INT or DINT.

The definition of a loop with FOR includes the specification of a start and end value. Both variables must be the same data type as the count variable.

Note

You use the FOR statement when the number of loop passes is known at the programming stage.

If the number of cycles is not known, the WHILE or REPEAT statement is more suitable, see WHILE statement (Page 4725) and REPEAT statement (Page 4726).

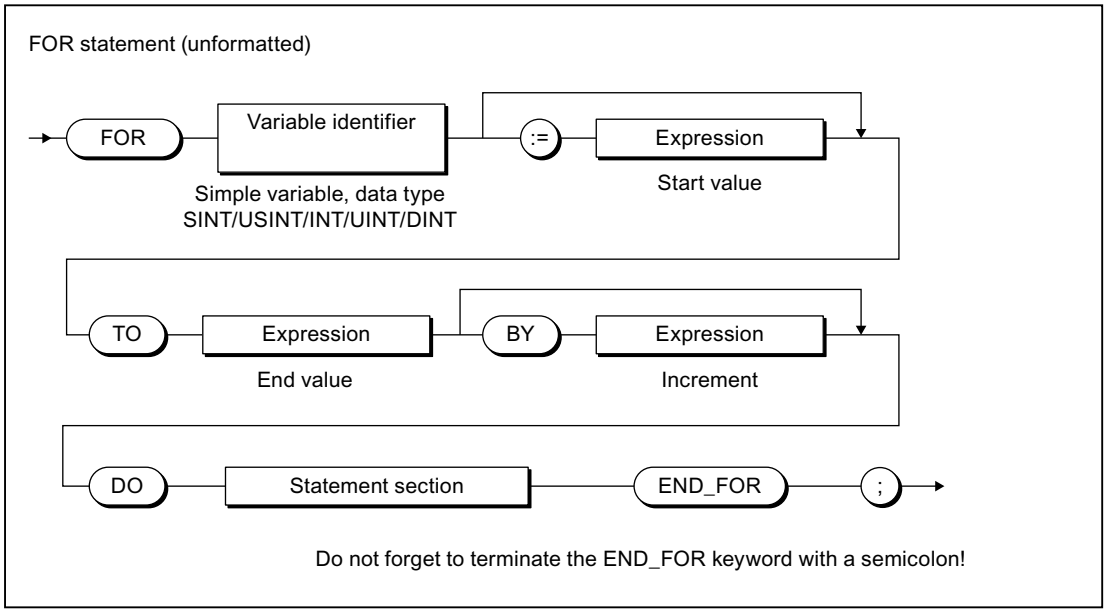


Figure 7-257 Syntax: FOR statement

Sequence of execution

The FOR statement is processed according to the following rules:

1. At the start of the loop, the count variable is set to the **start value** and is increased (positive increment) or decreased (negative increment) by the specified increment after each loop pass until the **end value** is reached. After the first loop pass, the content of the count variable is known as the **current value**.
2. On each pass, the system checks whether the following conditions are true:
   - **Start value or current value <= end value (**for **positive increment)** or
   - **Start value or current value >= end value (**for **negative increment)**

   If the condition is fulfilled, the sequence of statements is executed.
   If the condition is not fulfilled, the loop and, thus, the sequence of statements is skipped and the program is resumed after END_FOR.
3. If the FOR loop is not executed due to Step 2, the count variable retains the current value.

Rules

The following rules apply to the FOR statement:

- The *BY [increment]* specification can be omitted. If no increment is specified, the default is +1.
- The start value, end value and increment are expressions, see Expressions (Page 4709). The expression is evaluated once at the beginning of the FOR statement.
- If the start value and end value are of the data type DINT, the amount from (end value - start value) must be less than DINT#MAX (2**31 - 1), see also Value range limits of elementary data types (Page 4674).
- The count variable contains the value which triggers the loop exit, i.e. it is incremented before the loop is exited.
- During the loop execution, the count variable (current value) as well as the start value, the end value and the increment must not be changed.

Example

The following example illustrates the use of the FOR statement:

Example of the FOR statement

```
FOR k := 1 TO 10 BY 2 DO
    l := l + 1;
    // ...
END_FOR;
```

WHILE statement

Description

The WHILE statement allows a sequence of statements to be executed repeatedly under the control of an iteration condition. The iteration condition is formulated in accordance with the rules for a logic expression.

Note

You use the WHILE statement when the number of loop passes is not known at the programming stage.

If the number of passes is known, the FOR statement (Page 4723) is more suitable.

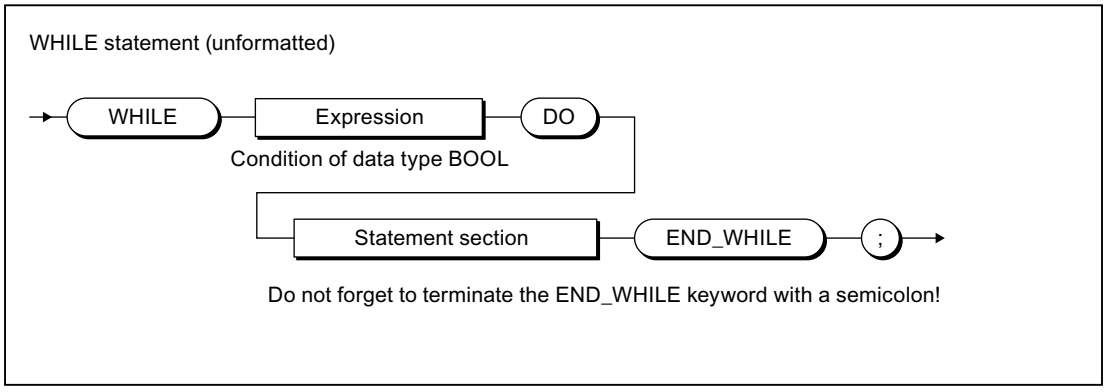


Figure 7-258 Syntax: WHILE statement

The statement section after DO is repeated until the iteration condition has the value TRUE.

Sequence of execution

The WHILE statement is processed according to the following rules:

1. The iteration condition is evaluated each time **before** the statement section is executed.
2. If the value is TRUE, the statement section is executed.
3. If the value is FALSE, the WHILE statement is terminated (this can occur the first time the condition is evaluated) and the program is resumed after END_WHILE.

Example

The following example illustrates the use of the WHILE statement:

Example of the WHILE statement

```
WHILE Index <= 50 DO
      Index:= Index + 2;
END_WHILE;
```

REPEAT statement

Description

A REPEAT statement causes a sequence of statements programmed between REPEAT and UNTIL to be executed repeatedly until a termination condition is true. The termination condition is formulated in accordance with the rules for a logic expression.

Note

You use the REPEAT statement when the number of loop passes is not known at the programming stage.

If the number of passes is known, the FOR statement (Page 4723) is more suitable.

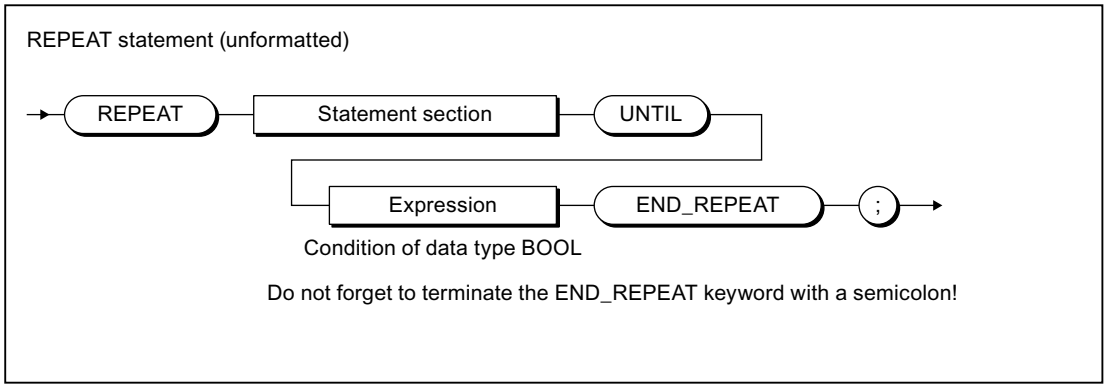


Figure 7-259 Syntax: REPEAT statement

The condition is checked **after** the statement section is executed. That means the statement section is executed at least once, even if the termination condition is true at the start.

Sequence of execution

The REPEAT statement is processed according to the following rules:

1. The iteration condition is evaluated each time **after** the statement section is executed.
2. If the value is FALSE, the statement section is executed again.
3. If the value is TRUE, execution of the REPEAT statement is terminated and program execution is resumed after END_REPEAT.

Example

The following example illustrates the use of the REPEAT statement:

Example of the REPEAT statement

```
Index := 1;
REPEAT
    Index := Index + 2;
UNTIL Index > 50
END_REPEAT;
```

EXIT statement

Description

An EXIT statement is used to exit a loop (FOR, WHILE or REPEAT loop) at any point, irrespective of whether the termination condition is true or false.

This statement has the effect of jumping directly out of the loop immediately surrounding the EXIT statement.

The program resumes after the end of the loop (e.g. after END_FOR).

Example

The following example illustrates the use of the EXIT statement:

Example of the EXIT statement

```
Index := 1;
FOR Index := 1 to 51 BY 2 DO
    IF %I0.0 THEN
        EXIT;
    END_IF;
END_FOR;
(*
The following value assignment is performed after the execution of EXIT or
after the regular end of the FOR loop.
*)
Index_find := Index_2;
```

CONTINUE statement

Description

The CONTINUE statement is only available if the compiler option **Language extensions IEC61131 3rd edition** is enabled, see Global settings of the compiler (Page 4623) and Local settings of the compiler (Page 4626).

It acts as a jump to the start of a loop (FOR, WHILE or REPEAT loop) from any point. Use this to jump to the start of the repeat statement which immediately surrounds the CONTINUE statement.

- With a FOR loop the run tag is increased by the programmed increment and compared with the full-scale value.
- With a WHILE loop the execution condition is reviewed.

Example

The following example illustrates the use of the CONTINUE statement:

Example of the CONTINUE statement:

```
sum := 0;
FOR i:= 1 TO 3 DO
    FOR j:= 1 TO 2 DO
        sum:= sum + 1;
        IF %I0.0 THEN
            CONTINUE;
        END_IF;
        sum:= sum + 1;
    END_FOR;
    sum:= sum + 1;
END_FOR;
(*
Once the loop has run the "sum" variable has the following value:
With %I0.0 = FALSE: sum = 15
With %I0.0 = TRUE:  sum =  9
*)
```

RETURN statement

Description

A RETURN statement causes termination of the POU currently being processed (program, function, function block).

When a function or a function block is terminated, program execution continues in the higher-level POU after the position where the function or function block was called.

Example

The following example illustrates the use of the RETURN statement:

Example of the RETURN statement

```
Index := 1;
FOR Index := 1 to 51 BY 2 DO
    IF %I0.0 THEN
        RETURN;
    END_IF;
END_FOR;
(*
The following value assignment is executed after the regular end of the FOR
loop, but not after the execution of RETURN.
*)
Index_find:= Index_2;
```

WAITFORCONDITION statement

Description

You can use the WAITFORCONDITION statement to wait for a programmable event or condition in a MotionTask. The statement suspends execution of the calling MotionTask until the condition is true. You program this condition in an Expression (Page 4763).

More information about the WAITFORCONDITION and expressions in this regard is contained in the *SIMOTION Basic Functions* Function Manual.

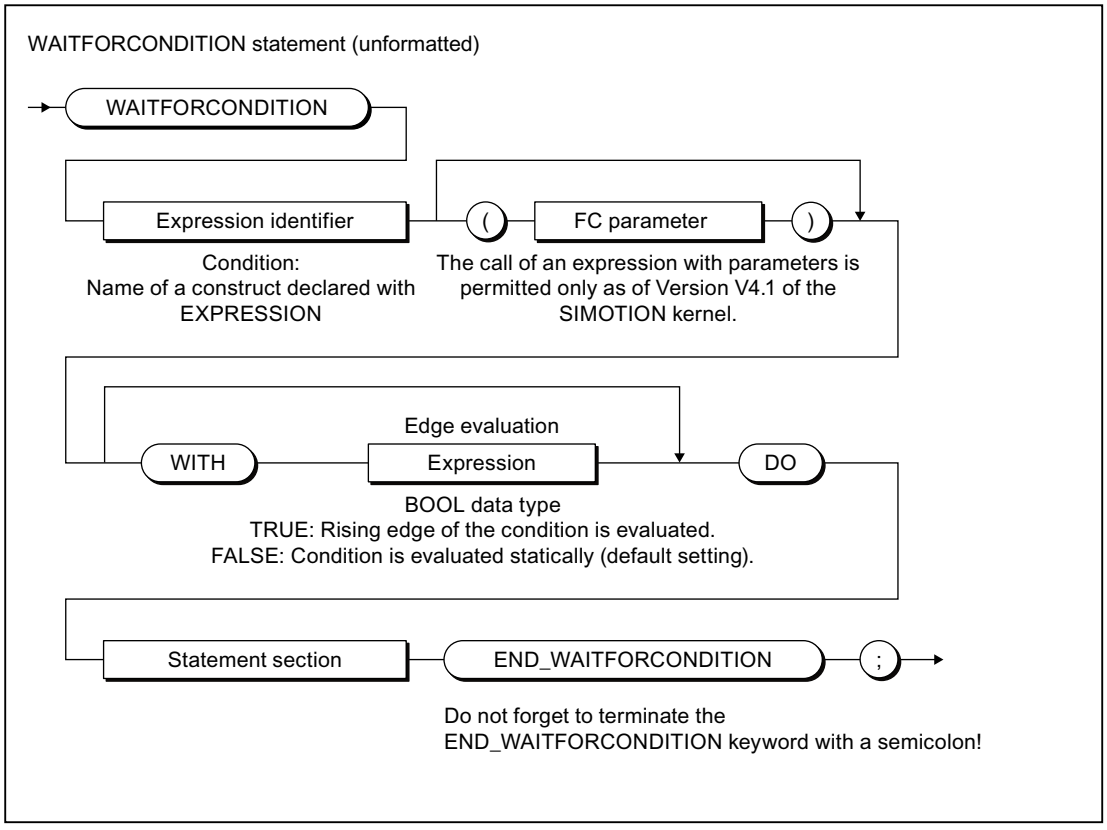


Figure 7-260 Syntax: WAITFORCONDITION statement

Expression identifier is a construct declared with EXPRESSION; its value defines (together with *WITH edge evaluation*, if necessary) whether the condition is considered as been satisfied.

The *WITH edge evaluation* sequence is optional. *Edge evaluation* is an expression of data type BOOL; it determines how the value of *expression identifier* is interpreted:

- *Edge evaluation* = TRUE:
  The rising edge of *expression identifier* is interpreted; i.e. the condition is satisfied when the value of *expression identifier* **changes** from FALSE to TRUE.
- *Edge evaluation* = FALSE:
  The static value of *expression identifier* is evaluated; i.e. the condition is satisfied when the value of *expression identifier* **is** TRUE.

If *WITH edge evaluation* is not specified, the default setting is FALSE, i.e. the static value of *expression identifier* is evaluated.

The statement section must contain at least one statement (empty statements also possible).

Example

The following example illustrates the use of the WAITFORCONDITION statement:

Example of the WAITFORCONDITION statement

```
// ...
// Call of the command with name of the expression
WAITFORCONDITION myExpression WITH TRUE DO
// At least one statement here, will be executed with higher priority, e.g.
    %Q0.0 := TRUE;
END_WAITFORCONDITION;
// ...
```

For a complete example, refer to the description for the Expression (Page 4763).

GOTO statement

The GOTO statement causes a jump to the jump label specified in the command (see Jump statement and labeling (Page 4933)).

You program jump statements with the GOTO statement and specify the jump label to which you want to jump. Jumps are only permitted within a POU.

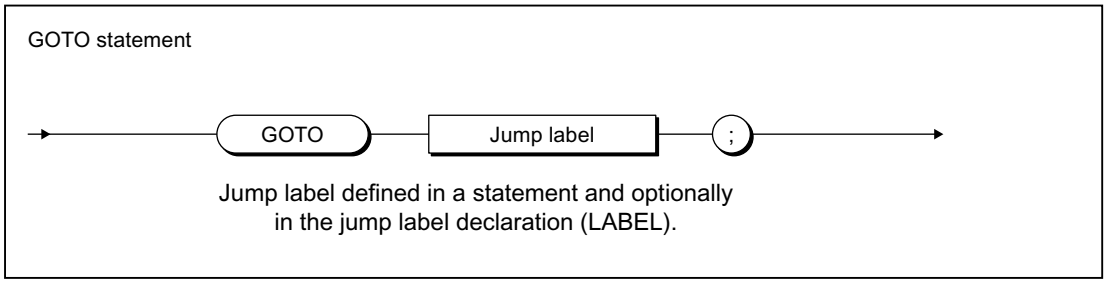


Figure 7-261 Syntax: GOTO statement

Note

You should only use the GOTO statement in special circumstances (for example, for troubleshooting). It should not be used at all according to the rules for structured programming.

Jumps are only permitted within a POU.

The following jumps are illegal:

- Jumps to subordinate control structures (WHILE, FOR, etc.)
- Jumps from a WAITFORCONDITION structure
- Jumps within CASE statements

Jump labels can only be declared in the POU in which they are used. If jump labels are declared, only the declared jump labels may be used.

7.2.4.8 Data type conversions

This section describes how you can implicitly and explicitly convert between elementary data types. It also contains an overview of the additional conversion possibilities.

Elementary data type conversion

The table presents an overview of the conversion options between numerical data types and bit data types. A distinction is made between:

- Implicit conversion (Page 4732): Conversion is automatic when different data types are used in an expression or when values are assigned by the compiler.
- Explicit conversion (Page 4734): Conversion is carried out when the user calls a conversion function (see *SIMOTION Basic Functions* Function Manual).

Table 7-409 Type conversion of numeric data types and bit data types

Source data type	Target data type												
	BOOL	**BYTE**	**WORD**	**DWORD**	**USINT**	**UINT**	**UDINT**	**SINT**	**INT**	**DINT**	**REAL**	**LREAL**	**STRING**
BOOL	–	Im/Ex	Im/Ex	Im/Ex	Val	Val	Val	Val	Val	Val	Val	Val	–
BYTE	Ex	–	Im/Ex	Im/Ex	Ex	Ex	Ex	Ex	Ex	Ex	Val	Val	Elem
WORD	Ex	Ex	–	Im/Ex	Ex	Ex	Ex	Ex	Ex	Ex	Val	Val	–
DWORD	Ex	Ex	Ex	–	Ex	Ex	Ex	Ex	Ex	Ex	Ex/Val	Val	–
USINT	Val	Ex	Ex	Ex	–	Im/Ex	Im/Ex	Ex	Im/Ex	Im/Ex	Im/Ex	Im/Ex	–
UINT	Val	Ex	Ex	Ex	Ex	–	Im/Ex	Ex	Ex	Im/Ex	Im/Ex	Im/Ex	–
UDINT	Val	Ex	Ex	Ex	Ex	Ex	–	Ex	Ex	Ex	Ex	Ex	Ex
SINT	Val	Ex	Ex	Ex	Ex	Ex	Ex	–	Im/Ex	Im/Ex	Im/Ex	Im/Ex	–
INT	Val	Ex	Ex	Ex	Ex	Ex	Ex	Ex	–	Im/Ex	Im/Ex	Im/Ex	–
DINT	Val	Ex	Ex	Ex	Ex	Ex	Ex	Ex	Ex	–	Ex	Im/Ex	Ex
REAL	Val	Val	Val	Ex/Val	Ex	Ex	Ex	Ex	Ex	Ex	–	Im/Ex	Ex
LREAL	Val	Val	Val	Val	Ex	Ex	Ex	Ex	Ex	Ex	Ex	–	Ex
STRING	–	Elem	–	–	–	–	Ex	–	–	Ex	Ex	Ex	–

Im: Implicit data type conversion possible

Ex: Explicit data type conversion possible by means of type conversion function *source data type*_TO_*target data type*

Val: Explicit data type conversion possible by means of type conversion function*source data type*_VALUE_TO_*target data type*

Elem: Implicit data type conversion with an element of the STRING data type

For information on conversion functions for date and time data types: Please refer to the *SIMOTION Basic Functions* Function Manual.

Implicit data type conversions

Implicit type conversion is always possible if an enlargement of the value range does not cause any value loss, e.g. from REAL to LREAL or from INT to REAL. The result is always defined.

The following figure provides a graphics-based view of all implicit type conversion chains. Each stage in the type conversion chain - reading from left to right or from top to bottom - always represents an enlargement of the value range.

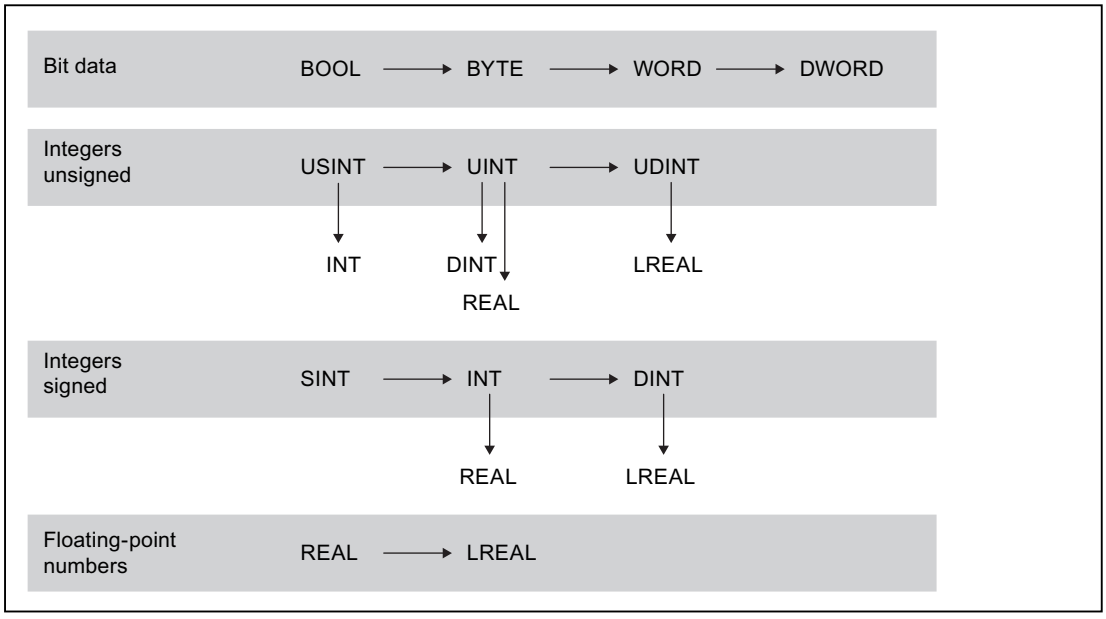


Figure 7-262 Implicit type conversion chains (one or more levels from left to right or one level from top to bottom)

The following implicit type conversions are supported:

1. Horizontally (from left to right) over one or more levels (e.g. USINT to UDINT)
2. Vertically (from top to bottom) over one level (e.g. UINT to REAL)

The implicit type conversions can be combined in the following order (e.g. INT to LREAL).

All other type conversions cannot be performed implicitly (e.g. UDINT to REAL), that is, you must use an explicit function (Page 4734) (see SIMOTION Basic Functions Function Manual).

Note

In arithmetic expressions, the result is always calculated in the largest number format contained in the expression.

A value can only be assigned to the expression if:

- The calculated expression and the variable to be assigned are of the same data type.
- The data type of the calculated expression can be implicitly converted to the data type of the variable to be assigned.

For more information on this error source and its solution: Please refer to the *SIMOTION Basic Functions* Function Manual.

Table 7-410 Example of data types in expressions and value assignments

```
VAR
    usint_var  : USINT;
    real_var   : REAL;
    byte_var   : BYTE;
    string_var : STRING[80] := 'example for string';
END_VAR

usint_var := 234 / 10;       // Expression data type: USINT
                             // Result = 23

real_var  := 234 / 10;       // Expression data type: USINT
                             // Implicit conversion possible
                             // Result = 23.0

usint_var := 234 / SINT#10;  // Expression data type: INT
                             // Implicit conversion and
                             // value assignment not possible

real_var  := 234 / 10.0;     // Expression data type: REAL
                             // Result = 23.4

usint_var := 234 / 10.0;     // Expression data type: REAL
                             // Implicit conversion and
                             // value assignment not possible

byte_var  := string_var[5];  // Implicit conversion possible
                             // Result = 16#70 ('p')

string_var[10] := byte_var;  // Implicit conversion possible
                             // Result = 'example fpr string'
```

Note

If applicable, specify the data type explicitly for numbers (e.g. UINT#127, if the number 127 is to be of data type UINT instead of USINT).

Explicit data type conversions

Explicit conversion is always required if information could be lost, for example, if the value range is decreased or the accuracy is reduced, as is the case for conversion from LREAL to REAL.

The conversion functions for numeric data types and bit data types are listed in the *SIMOTION Basic Functions* Function Manual.

The compiler outputs warnings when it detects conversions associated with loss of precision.

Note

The type conversion may cause errors when the program is running, which will trigger the error response set in the task configuration (see "Execution errors in programs" in the *SIMOTION Basic Functions* Function Manual).

Special attention is required when converting DWORD to REAL. The bit string from DWORD is taken unchecked as the REAL value. You must make sure that the bit string in DWORD corresponds to the bit pattern of a normalized floating-point number in accordance with IEEE. To do this, you can use the _finite and _isNaN functions.

Otherwise, an error can be triggered (FPU exception) as soon as the REAL value is first used for an arithmetic operation (for example, in the program or when monitoring in the symbol browser).

Note

The following applies if the value range limits are exceeded during conversion from LREAL to REAL:

- Underflow (absolute value of LREAL number is smaller than the smallest positive REAL number):
  Result is 0.0
- Overflow (absolute value of LREAL number is larger than the largest positive REAL number):
  The error response specified during task configuration is triggered.

Supplementary conversions

The ST system functions and ST system functions also permit the following conversions:

- **Combining bit-string data types**
  These functions combine multiple variables of a bit string data type into one variable of a higher-level data type.
- **Splitting bit-string data types**
  These function blocks split up a variable of a bit string data type into multiple variables of a higher-level data type.
- **Converting between any data types and byte arrays**
  They are commonly used to create defined transmission formats for data exchange between various devices.
  For further information (e.g. on the arrangement of the byte arrays, application example): Please refer to the *SIMOTION Basic Functions* Function Manual.
- **Conversion of technology object data types**
  It converts variables of a hierarchical TO data type (driveAxis, posAxis, or followingAxis) or of the general ANYOBJECT type to a compatible TO data type.

For Application Examples and further information: Please refer to the *SIMOTION Basic Functions* Function Manual.

7.2.5 Functions, Function Blocks, and Programs

This chapter describes how to create and call user-defined functions and function blocks. Standard functions are already available in the system for type conversion, trigonometry, and bit string manipulation. The *SIMOTION Basic Functions* Function Manual describes how to use system functions and functions of technology objects (TO functions).

A **function** (FC) is a logic block with no static data. All local variables lose their value when you exit the function and are reinitialized the next time you call the function.

A **function block** (FB) is a code block with static data. Since an FB has memory, its output parameters can be accessed at any time and from any point in the user program. Local variables retain their values between calls.

Programs are similar to FBs, but have no parameters. However, they can be assigned execution levels and tasks (see *SIMOTION Basic Functions* Function Manual).

FCs and FBs have the advantage that they can be reused, because they are encapsulated source file sections to which parameters can be assigned.

Functions, function blocks, and programs are program organization units (POUs), i.e. they are executable source file sections. You will find an overview of all source file sections in Use of the source file sections (Page 4807).

7.2.5.1 Creating and calling functions and function blocks

The following description explains how to create and call functions (FCs) and function blocks (FBs). For a complete example showing the differences between FCs and FBs see Comparison of functions and function blocks (Page 4758).

The order in which you must define and call the stipulated source file sections is given in Use of the source file sections (Page 4807).

For a description of how to make FCs and FBs public at other program sources or make them available for use by other program sources, see Declaring ST source files public and using them (Page 4827).

Defining functions

You can define a function (FC) within the implementation section of an ST source file in the section for program organization units (POE). If the compiler option (Page 4623) "Allow forward declaration" is not activated a FC must be defined before the POE (program, FB or FC) in which it is called.

Use the following syntax:

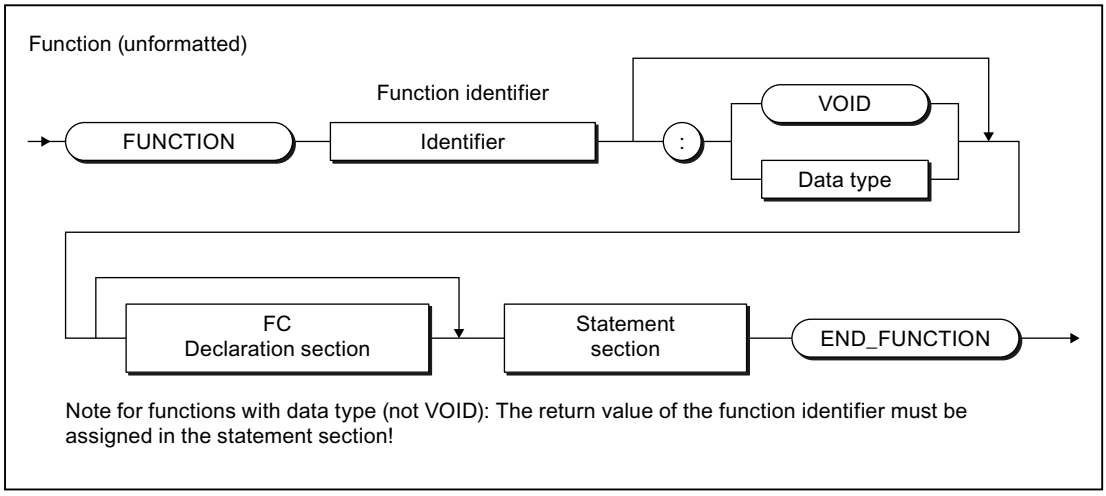


Figure 7-263 Syntax: Function (FC)

Enter in the following sequence, see example in Source file with comments (Page 4760)):

- The FUNCTION keyword
- An identifier as FC name
- If the function has a return value, the data type for the return value following a colon
  The symbolic data type VOID can also be entered for functions with no return value.

This is then followed by:

- The optional declaration section (Page 4741)
- The statement section (Page 4746)
- The END_FUNCTION keyword

Defining function blocks

Defining classic function blocks

You can define a function block (FB) within the implementation section of an ST source file in the section for program organization units (POE). If the compiler option (Page 4623) "Allow forward declaration" is not activated a FB must be defined before the POE (program, FB or FC) in which it is called.

Use the following syntax:

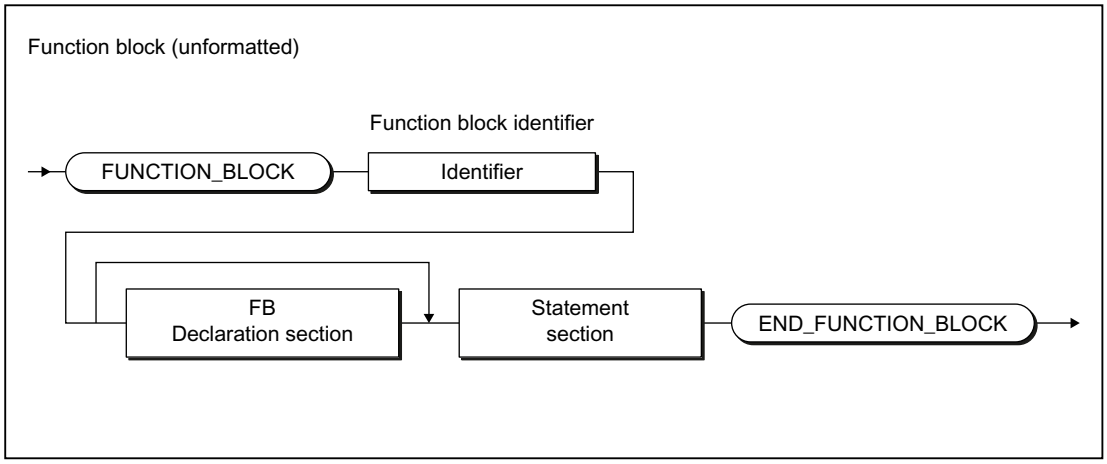


Figure 7-264 Syntax: Function block (FB)

Enter in the following sequence, see example in Source file with comments (Page 4760)):

- The FUNCTION_BLOCK keyword
- An identifier as FB name

This is then followed by:

- The optional declaration section (Page 4741)
- The statement section (Page 4746)
- The END_FUNCTION_BLOCK keyword

Defining object-oriented function blocks with methods

Function blocks can also be defined with methods. Reduced object-oriented programming is possible with this irrespective of the SIMOTON Kernel version. Only the compiler option (Page 4623) “Permit object-oriented programming" needs to be activated.

You can define object-oriented function blocks (FB) with methods and public variables within the implementation section of an ST source file in the section for program organization units (POE). If the compiler option (Page 4623) “Allow forward declaration" is not activated a FB must be defined before the POE (program, FB or FC) in which it is called.

Use the following syntax:

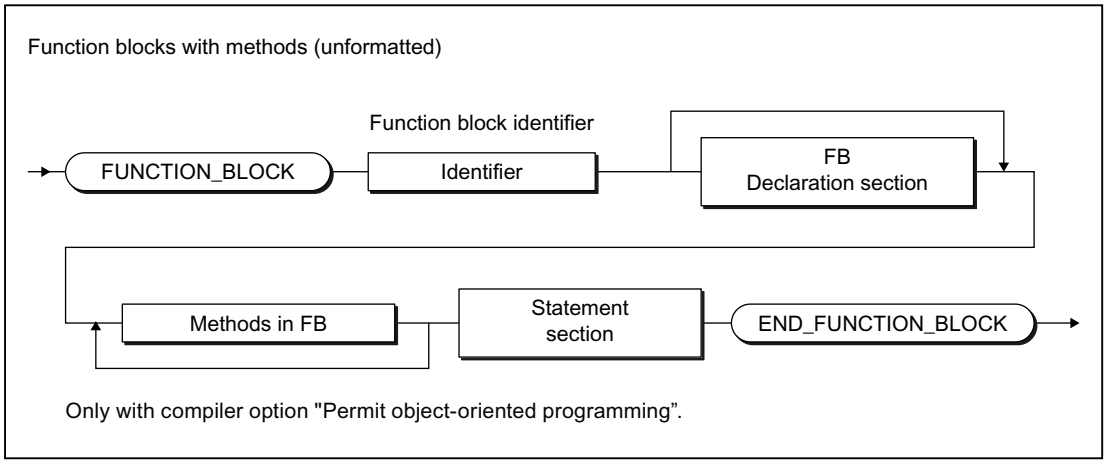


Figure 7-265 Syntax: Function block (FB) with methods

Enter in the following sequence:

- The FUNCTION_BLOCK keyword
- An identifier as FB name

This is then followed by:

- The optional declaration section (Page 4741)
- the methods in the function block
- The statement section (Page 4746)
- The END_FUNCTION_BLOCK keyword

Note

Object-oriented function blocks cannot be derived. Inheritance is therefore not possible for methods.

Defining methods in object-oriented FBs

Executable statements for an object-oriented function block can be summarized as methods. The methods are defined within a function block before the statement section for the FB.

A method roughly corresponds to a function (Page 4736). Specification of an access identifier determines the environment from which the method can be called.

If the compiler option (Page 4623) "Allow forward declaration" is not activated, a method must be defined before the program organization unit (program, FB, FC or class) or method in which it is called.

Use the following syntax:

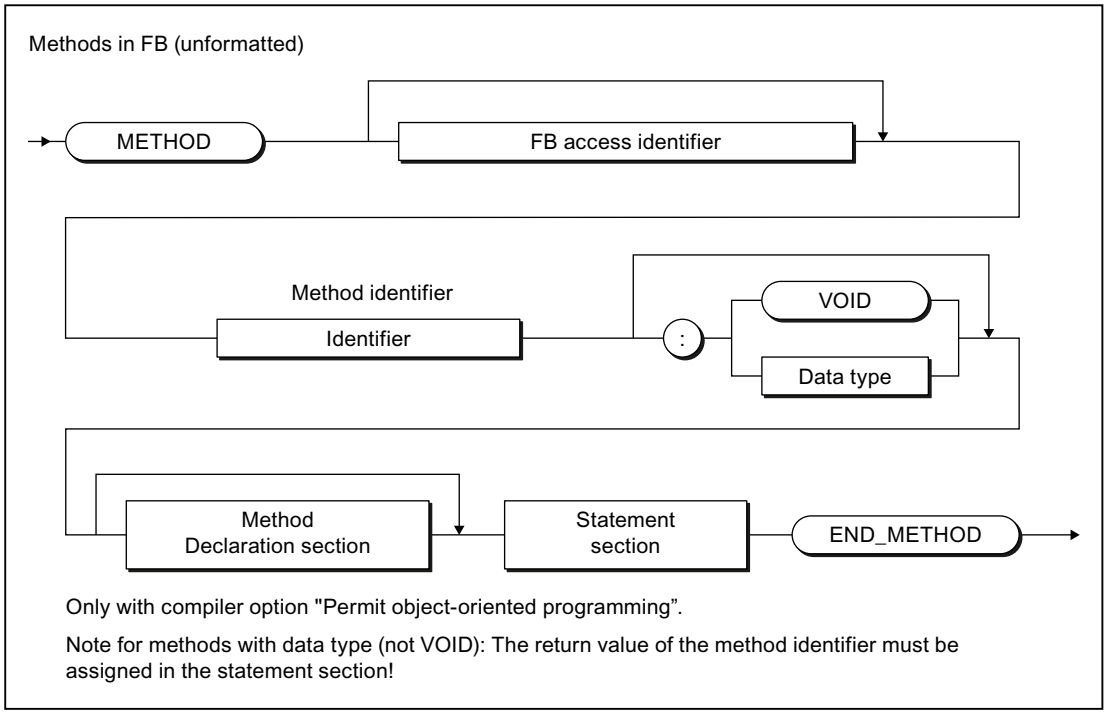


Figure 7-266 Syntax: Methods in the function block

Enter in the following sequence:

- The METHOD keyword.
- The optional FB access identifier (PUBLIC, PRIVATE or INTERNAL), default PRIVATE.
  For access identifier, see section "Access identifiers within object-oriented FBs" (Page 4740).
- An identifier as method name.
- if the method has a return value, the data type for the return value following a colon.
  The symbolic data type VOID can also be entered for a method with no return value.

This is then followed by:

- The optional declaration section (Page 4771)
- The statement section (Page 4773)
- The END_METHOD keyword

Access identifiers within object-oriented FBs

An access identifier may be specified within object-oriented function blocks for declaration blocks or methods:

- With declaration blocks:
  Directly after the keyword for the relevant block (e.g. TYPE, VAR).
  The access identifier is valid for all identifiers within this declaration block.
- With methods:
  Directly after the keyword METHOD.
  The access identifier is valid for the relevant method.

You use the access identifier to determine whether the identifiers or the method can be used only inside or outside of the function block, see the following table.

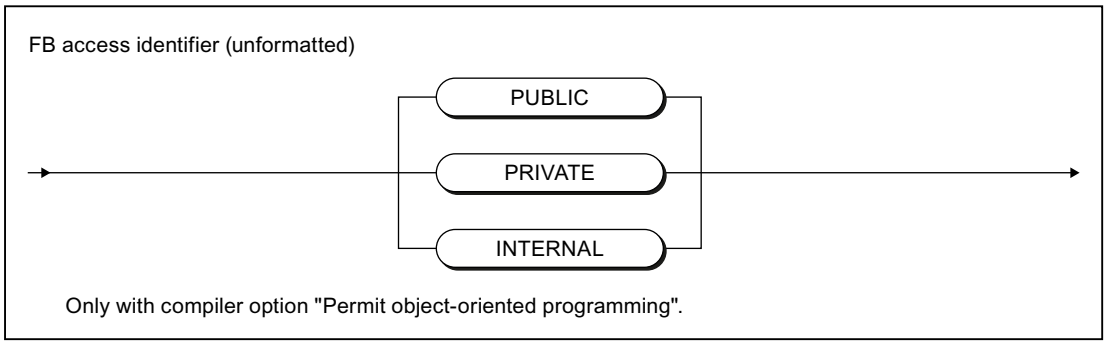


Figure 7-267 Syntax: FB access identifier

Table 7-411 Possible access identifiers within an object-oriented function block

Access identifier	Meaning
PRIVATE	The identifiers for the relevant declaration block (variables, data types, etc.) or the method can be used only within this function block. This is the default when no access identifier is specified.
INTERNAL	Only when the identifier is defined within an object-oriented namespace (Page 4904). The identifiers of the relevant declaration block (variables, data types, etc.) or the method can be used only within the higher-level namespace. Within the higher-level namespace, the following apply: • The declared identifiers can also be accessed outside the function block. This does not apply to retentive local variables (VAR RETAIN). • The method can also be called outside the function block.
PUBLIC	The identifiers for the relevant declaration block (variables, data types, etc.) or the method are public. • The declared identifiers can also be accessed outside the function block and outside the higher-level object-oriented namespace. • The method can also be called outside the function block and outside the higher-level object-oriented namespace. This access identifier is not permitted in declaration blocks for retentive local variables (VAR RETAIN / END_VAR).

Declaration section of FB and FC

A declaration section is subdivided into various declaration blocks that are each identified by a separate pair of keywords. Each block contains a declaration list for similar data, such as constants, local variables and parameters. Each type of block may only appear once; the blocks may appear in any order.

The following options are then available for the declaration section of an FC and an FB, see also the example in Source file with comments (Page 4760):

Permissible declaration blocks

Table 7-412 Declaration blocks for FC and FB: Options

Data	Syntax	FB	FC	Method in FB
Data type	**TYPE**[1] *Declaration list*[4] **END_TYPE**	X	X	–
Constant	**VAR CONSTANT**[1] *Declaration list*[4] **END_VAR**	X	X	X
Input parameter	**VAR_INPUT** *Declaration list*[4] **END_VAR**	X	X	X
In/out parameter	**VAR_IN_OUT** *Declaration list*[4] **END_VAR**	X	X	X
Output parameters	**VAR_OUTPUT** *Declaration list*[4] **END_VAR**	X	X	X
Local static variable	**VAR**[1 3] *Declaration list*[4] **END_VAR**	X	–	–
Local temporary variable	**VAR** *Declaration list*[4] **END_VAR**	–	X	X
	VAR_TEMP *Declaration list*[4] **END_VAR**	X	X	X
Retentive variable (as of version 4.5 of the SIMOTION Kernel)	**VAR RETAIN**[2 3] *Declaration list*[4] **END_VAR**	X	–	–

[1] Additional specification of the FB access identifier possible with FB and compiler option (Page 4623) "Permit object-oriented programming" activated: **PUBLIC**, **PRIVATE** or **INTERNAL**, default **PRIVATE**; see section "Access identifier within object-oriented FBs (Page 4740)". The relevant blocks may occur multiple times.

[2] Additional specification of the access identifier **PRIVATE** or **INTERNAL** possible with FB and activated compiler option (Page 4623) "Permit object-oriented programming". The relevant blocks may occur multiple times.

[3] Additional specification of the keyword **OVERRIDE** possible with FB and activated compiler option (Page 4623) "Permit object-oriented programming". This keyword enables the initialization values of variables to be overridden with the **PRIVATE** access identifier when the instance is declared.

[4] *Declaration list*: The list of identifiers of the type to be declared

Parameter blocks

Parameters are local data and are formal parameters of a function block, a function or a method in FB. When the FB or FC is called, the formal parameters are substituted by the actual parameters, thus providing a means of exchanging information between the called and calling source file sections.

- Formal input parameters receive the actual input values (data flow inwards).
- Formal output parameters are used to transfer output values (data flow outwards).
- Formal in/out parameters act as input and output parameters.

The following figures show the syntax for the parameter declaration of an FB, an FC or a method in the FB.

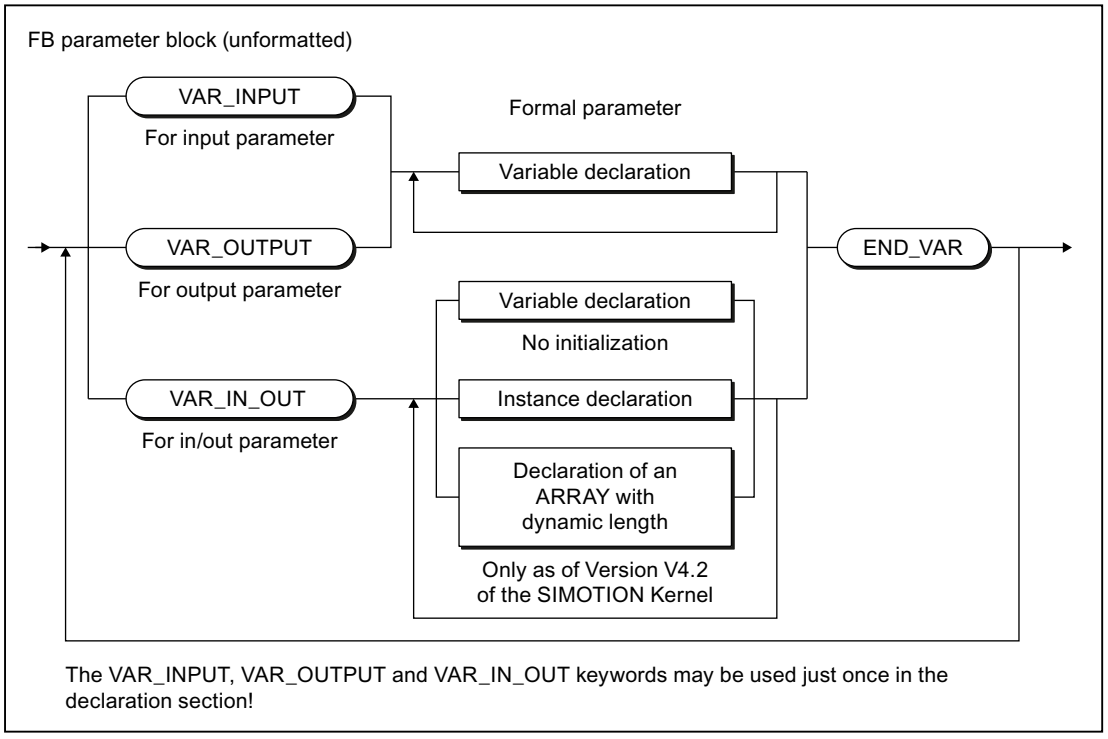


Figure 7-268 Syntax: FB parameter block

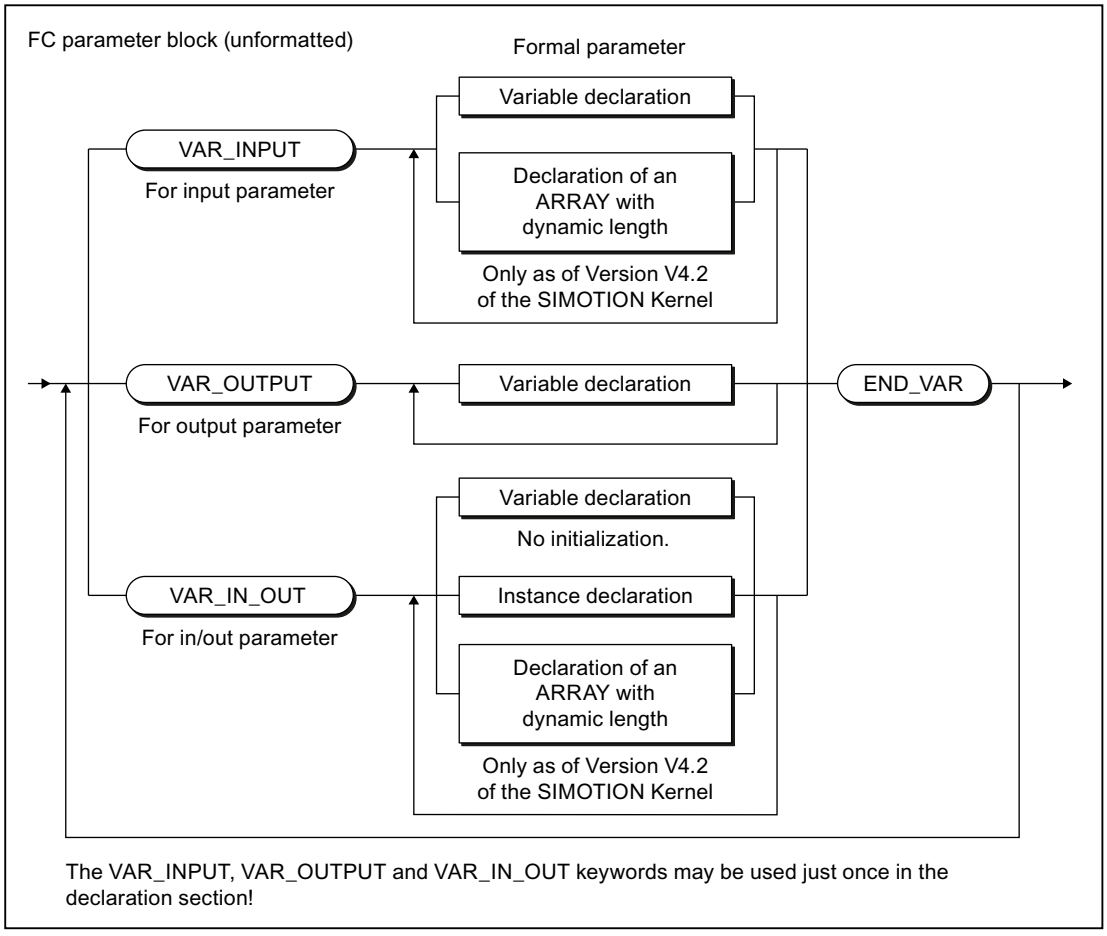


Figure 7-269 Syntax: FC parameter block

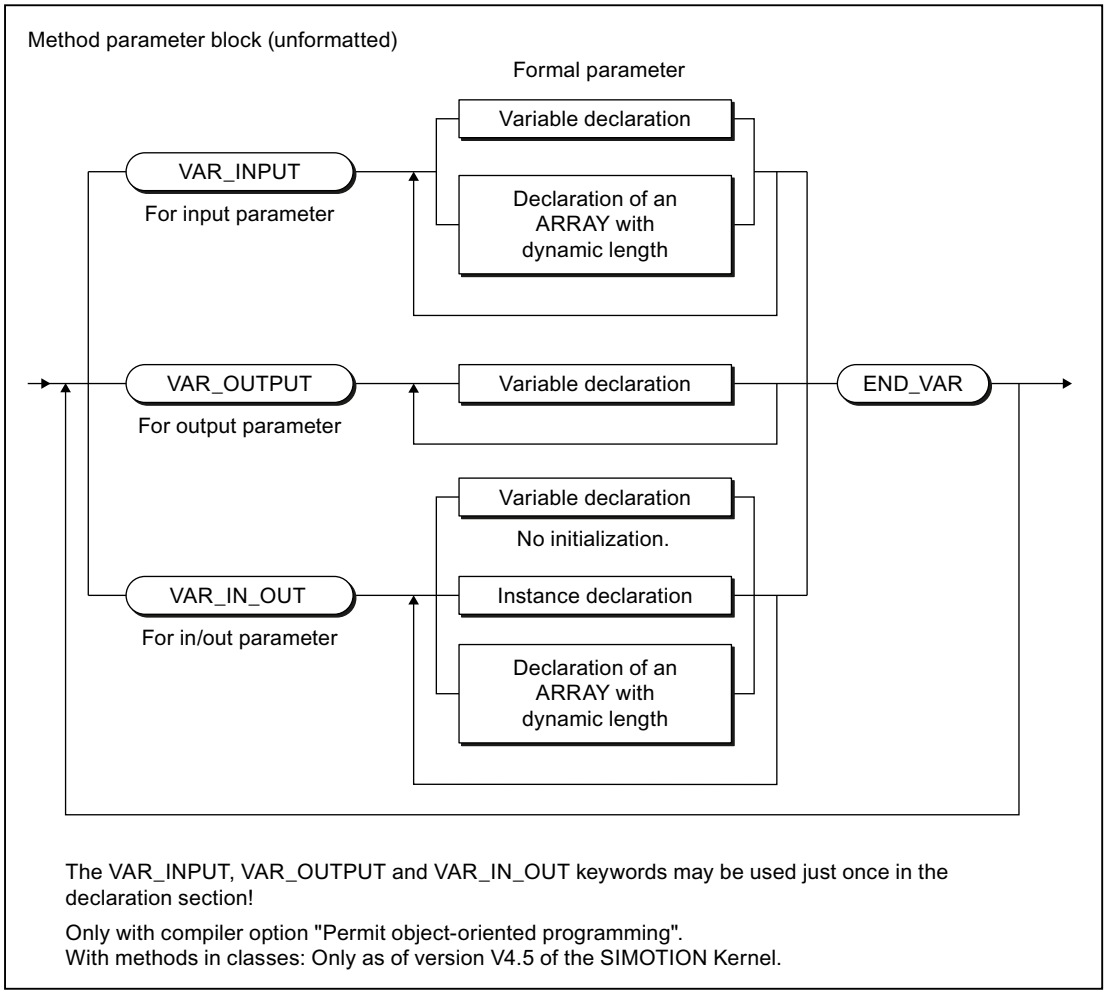


Figure 7-270 Syntax: Method parameter block

Apart from variables you can also declare the following in the parameter blocks:

- in the parameter block for in/out parameters:
  - Instances of function blocks (Page 4754)
  - Arrays with a dynamic length (Page 4747), as of version V4.2 of the SIMOTION Kernel.
- with functions in the parameter block for input parameters:
  - Arrays with a dynamic length (Page 4747), as of version V4.2 of the SIMOTION Kernel.

You can use the declared parameters the same as other variables within the FB or FC, with the following exception: You cannot assign values to input parameters.

From outside of an FB or an FC, you can access:

- The input and output parameters of an FB by means of structured variables (Page 4682). Input parameters can only be accessed when the compiler option (Page 4623) "Permit language extensions" has been activated.
  Data access to the output parameter is possible as standard.
- The return value of an FC by using the function in an expression and assigning this expression to a variable, for example. Specifying the function name calls the function and returns a result at the same time.

Note

When saving the project in the old project format:

Projects in which the output parameters for a function are declared cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

Statement section of FB and FC

The statement section of the FC or FB contains instructions that are executed when the FC or FB is called. There is no difference compared to the formal rules for creating a statement section; however, you should note the information in the following table.

Note

For tips on the efficient use of parameters, please refer to the Runtime-optimized Programming section in the SIMOTION Basic Functions Function Manual.

Table 7-413 Use of parameters and variables in FCs and FBs

Parameter/variable	Use
Input parameters	With the call of an FC or an FB, assign the current values to the input parameters. These values are used for data processing within the FC or the FB, for example, for calculations, but cannot be modified themselves. Only with "Permit language extensions" compiler option enabled (see Global compiler settings (Page 4623) or Local compiler settings (Page 4626)): The input parameters of an FB can be read and written using structured variables, including outside the FB (e.g. in the calling source file module).
In/out parameter	You assign a variable to an in/out parameter for the call of the FC or FB. The FC or the FB accesses this variable directly and can change this directly. Type conversions are not supported. The variable assigned to an in/out parameter must be able to be read and written directly. Therefore, system variables (of the SIMOTION device or a technology object), I/O variables or process image accesses cannot be assigned to an in/out parameter.

Parameter/variable	Use
Output parameters	You assign a variable to an output parameter for the call of an FC or FB using the **=>** operator. The value of the output parameter (result) is transferred to the variables when the FC or FB is closed. The following also applies to FBs: The output parameters of an FB can be read using structured variables, including outside the FB (e.g. in the calling source file section).
Local variables	Local variables are variables that are declared and used only within the block. All local variables (VAR ... END_VAR or VAR_TEMP ... END_VAR) are temporary in an FC, i.e. they lose their value when the FC is terminated. The next time the FC is called, they are reinitialized. A differentiation between static and temporary local variables is made in the FB: • Static variables (VAR ... END_VAR) retain their value when the FB is closed. • Temporary variables (VAR_TEMP ... END_VAR) lose their value when the FB is closed. The next time the FB is called, they are reinitialized. The value of a local variable cannot be queried directly by the calling block. This is only possible using an output parameter.

ARRAY with a dynamic length (as of Kernel V4.2)

Declaration as an in-out parameter

As of SIMOTION Kernel Version 4.2, you can declare arrays with a dynamic length in functions and function blocks. This is possible:

- With functions:
  - in the parameter block for in/out parameters (Page 4741) (VAR_IN_OUT / END_VAR).
  - in the parameter block for input parameters (Page 4741) (VAR_INPUT / END_VAR).
- With function blocks:
  in the parameter block for in/out parameters (Page 4741) (VAR_IN_OUT / END_VAR).

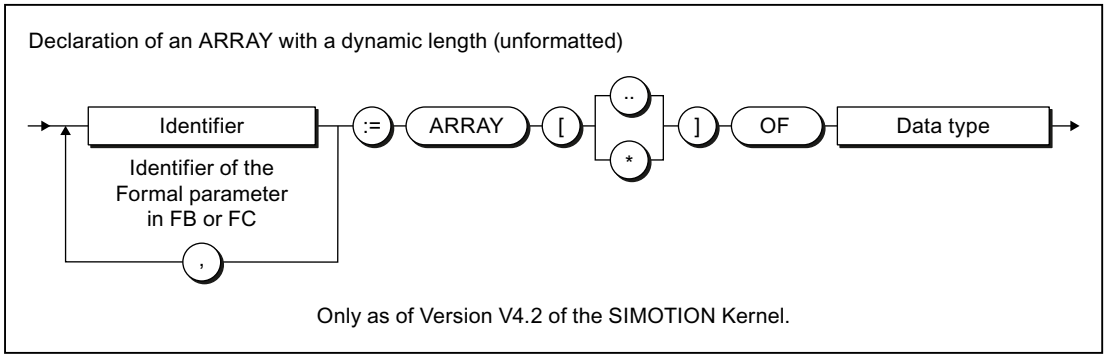


Figure 7-271 Syntax: Declaration of an ARRAY with a dynamic length

Note

When saving the project in the old project format:

Projects in which [*] is used to identify a dynamic array cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

Using an ARRAY with a dynamic length

When the function or an instance of the function block is called, arrays of any length of the declared data type can be transferred. The reference to the transferred array and its index limits are saved on the local data stack (Page 4848).

You can use the _firstIndexOf (in := *array-name*) and _lastIndexOf (in := *array-name*) functions within the function or function block to define the lower and upper index limits. The _lengthIndexOf (in := *array-name*) function supplies the number of elements in the ARRAY. The following applies: _lengthIndexOf (x) := _lastIndexOf (x) - _firstIndexOf (x) + 1.

Note

With compiler option (Page 4623) "Permit language extensions IEC61131 3rd edition" enabled the standardized functions LOWER_BOUND (in := *array-name*) or UPPER_BOUND (in := *array-name*) can be used to determine the lower and upper index limits respectively.

You can define the size of the memory space occupied by the ARRAY with _sizeOf (in := *array-name*). The following applies: _sizeOf (in := *array_of_type*) := _lengthIndexOf (in := *array_of_type*) * _sizeOf (in := *type*).

The syntax of these functions is described in the "SIMOTION Basic Functions" Function Manual.

Note

The above functions are executed during runtime for an ARRAY with a dynamic length.

Example

Table 7-414 Example of using an ARRAY with a dynamic length

```
FUNCTION_BLOCK example_dyn_array
    VAR_IN_OUT
        flexarray : ARRAY [..] OF DINT;
    END_VAR

    VAR_TEMP
        i : DINT := 0;
    END_VAR

    i := _firstIndexOf (in := flexarray);

    WHILE i <= _lastIndexOf (in := flexArray) DO
        flexArray[i] := i;
        i := i + 1;
    END_WHILE;
END_FUNCTION_BLOCK

PROGRAM test_dyn_array
    VAR
        array_1 : ARRAY [0 .. 29] OF DINT;
        fb_example : example_dyn_array;
    END_VAR
    // ...
    fb_example (flexarray := array_1);
    // ...
END_PROGRAM
```

Call of functions and function block calls

This provides an overview of the call of the functions and function blocks.

Principle of parameter transfer

When you call an FC or FB, data exchange takes place between the calling and the called block. The parameters to be transferred must be specified as a parameter list in the call. The parameters are written in parentheses. Several parameters are separated by commas.

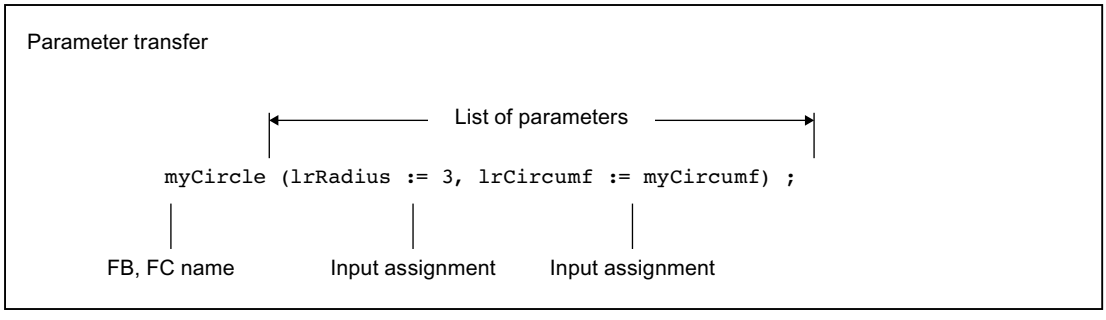


Figure 7-272 Principle of parameter transfer for the call

Input and in/out parameters are normally specified as a value assignment. In this way, you assign values (actual parameters) to the parameters you have defined in the declaration section of the called block (formal parameters).

The assignment of output parameters is made using the **=>** operator. In this way, you assign a variable (actual parameter) to the output parameters you have defined in the declaration section of the called block (formal parameters).

Parameter transfer to input parameters

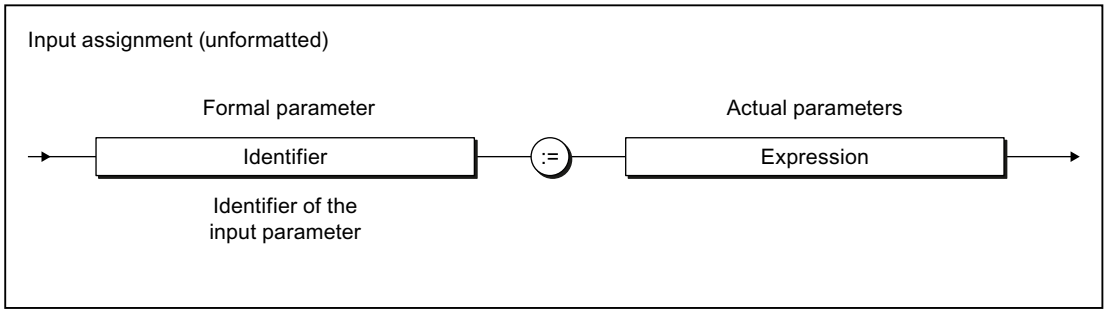


Figure 7-273 Syntax: Input assignment

You transfer the data (actual parameters) to the formal input parameters of an FB or FC by means of input assignments. You can specify the actual parameters in the form of expressions. You can use the formal input parameters in statements within the FB or FC, but you cannot modify their values.

A short form of parameter transfer is supported, but should not be applied in conjunction with user-defined FBs. This short form is required only for some FCs, see *SIMOTION Basic Functions* Function Manual.

The assignment of actual parameters is optional for an FB. If no input assignment is specified, the values of the last call are retained because an FB is a source file section with memory.

The assignment of an actual parameter is optional for an FC when an initialization expression was specified for the declaration of the formal parameter.

Also refer to the examples in Calling functions (Page 4753) and Calling function blocks (instance calls) (Page 4754).

You can also gain read and write access to an FB's input parameter at any time outside the FB. For further details, see: Accessing the FB's input parameter outside the FB (Page 4756).

Parameter transfer to in/out parameters

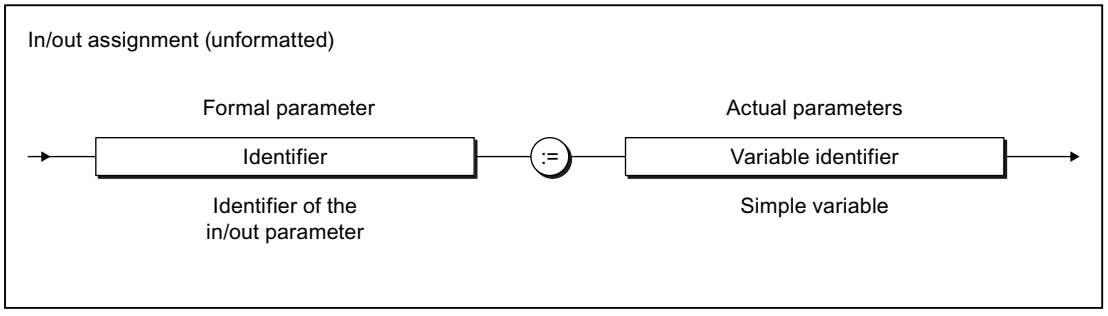


Figure 7-274 Syntax: In/out assignment

You transfer the data (actual parameters) to the formal in/out parameters of an FB or an FC using in/out assignments. You can only assign a variable of the same type to the formal in/out parameter, data type conversions are not possible.

You can use and change the formal in/out parameters in statements within the FC or the FB. The FC or the FB accesses the variable of the actual parameter directly and can change it directly.

See also the examples in Calling functions (Page 4753) and Calling function blocks (instance calls) (Page 4754).

When using the STRING data type in in/out assignments, the declared length of the actual parameter must be greater than or equal to the length of the formal in/out parameter (see following example).

Table 7-415 Example of the use of the STRING data type in in/out assignments

```
FUNCTION_BLOCK REF_STRING
    VAR_IN_OUT
        io : STRING[80];
    END_VAR
    ;    // Statements
END_FUNCTION_BLOCK

FUNCTION_BLOCK test
    VAR
        my_fb : REF_STRING;
        str1  : STRING [100];
        str2  : STRING [50] ;
    END_VAR
    my_fb (io := str1);    // Permissible call
    my_fb (io := str2);    // Illegal call,
                          // Compiler error message
END_FUNCTION_BLOCK
```

The variable assigned to an in/out parameter must be able to be read and written directly. Therefore, system variables (of the SIMOTION device or a technology object), I/O variables or process image accesses cannot be assigned to an in/out parameter.

Please note the different parameter access times!

Parameter transfer to output parameters

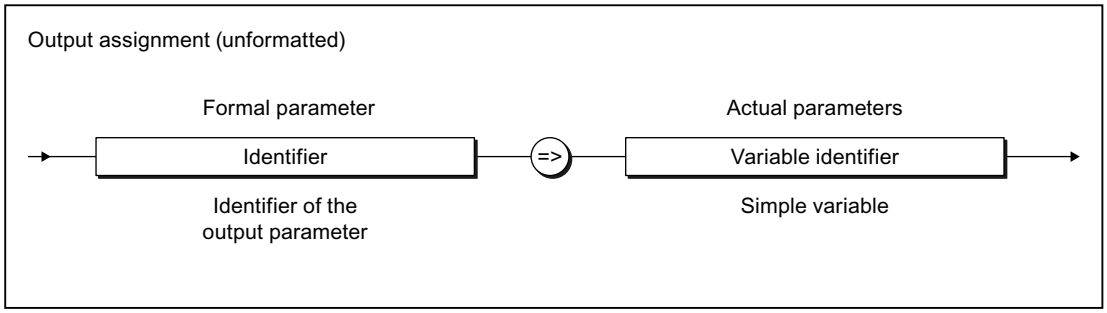


Figure 7-275 Syntax: Output assignment

You use an output assignment to assign the formal output parameters of an FC or FB to the variables (actual parameter) that accept the value of the formal output parameter when the FB is closed.

You can use and change the formal output parameters in statements within the FC or the FB.

See also the examples in Calling functions (Page 4753) and Calling function blocks (instance calls) (Page 4754).

Output assignments are optional for the parameter transfer. If there is no output assignment for an output parameter, the following applies:

- With functions (FC):
  The values for the output parameters are lost after the FC is terminated.
- With function blocks (FB):
  You can gain read and write access to an FB's output parameter at any time, including outside the FB. For further details, see: Accessing the FB's output parameter outside the FB (Page 4756).

Parameter access times

The types of access and thus the parameter access times are different:

- In the case of input assignments, the values of the actual parameters are copied into the formal parameters. If large structures, such as arrays, are copied and the FC or FB is called frequently, this can limit performance.
- Values are not copied in in/out assignments. Rather, in this case a link is established between the memory addresses of the formal parameters and those of the actual parameters. Transferring the variables is therefore faster than input assignments (especially where large volumes of data are involved). However, accessing variables from the FB can be slower.
- If you are using unit variables, nothing is copied to the function or function block because these variables are valid in the entire ST source file (see Variable model (Page 4831)).

Note

Using in/out parameters instead of input parameters is only faster if a large volume of data is to be passed to the function block.

If unit variables are used predominantly instead of parameters, the resulting program structure will be complex and confusing: object orientation, data encapsulation, multiple use of variable names (encapsulation of validity ranges), etc., are no longer possible.

Calling a function

A function is called as follows:

Function with return value (data type other than VOID)

The function is placed on the right-hand side of a value assignment. It can also appear as an operand within an expression. After calling the function, its return value is used at the appropriate point to calculate the expression.

Examples:

```
y := sin(x);
y := sin(in := x);
y := sqrt (1 - cos(x) * cos(x));
```

Note

In the function itself, the result (return value) is assigned to the function name.

Function without return value (no data type or VOID data type)

The assignment consists only of the function call.

The following example is valid provided a funct1 function has already been defined with the in1 and in2 input parameters along with the inout in/out parameter and the out1 and out2 output parameters.

Example:

```
funct1 (in1 := var11, in2 := var12, inout1 := var13, out1 => var14, out2 =>
var15);
```

Calling function blocks (declaring and calling instances)

Declaring an instance of a function block

Before you call a function block (FB), you must declare an instance. You declare a variable and enter the name of the function block as the data type. You declare this instance:

- Locally (within VAR/END_VAR in the declaration section of a program or function block)
- Globally (within VAR_GLOBAL/END_VAR in the interface of implementation section)
- As an in/out parameter (within VAR_IN_OUT / END_VAR in the declaration section of a function block or a function).

Instance-specific initialization of individual local variables of the function block is possible in the case of function blocks declared in ST source files by means of the compiler option (Page 4626) "Permit object-oriented programming". In this case, you override the initialization values that were stipulated in the function block declaration. Specification of the variables to be initialized is similar to the process for initializing a structure, and involves specifying a structure initialization list (Page 4699) that is enclosed in brackets.

You can in all cases initialize public variables of the function block (i.e. variables with the PUBLIC access identifier) specific to the instance. In order to be able to initialize private variables of the function block (i.e. variables with PRIVATE access identifier), the keyword OVERRIDE must be stated at their declaration block after the access identifier.

The following generally applies: The function block must be declared in the ST source file before an instance can be declared.

Exception: It is sufficient to define a POU prototype (Page 4929) beforehand if initialization is not specified and when compiler option (Page 4623) "Permit forward declarations" is activated.

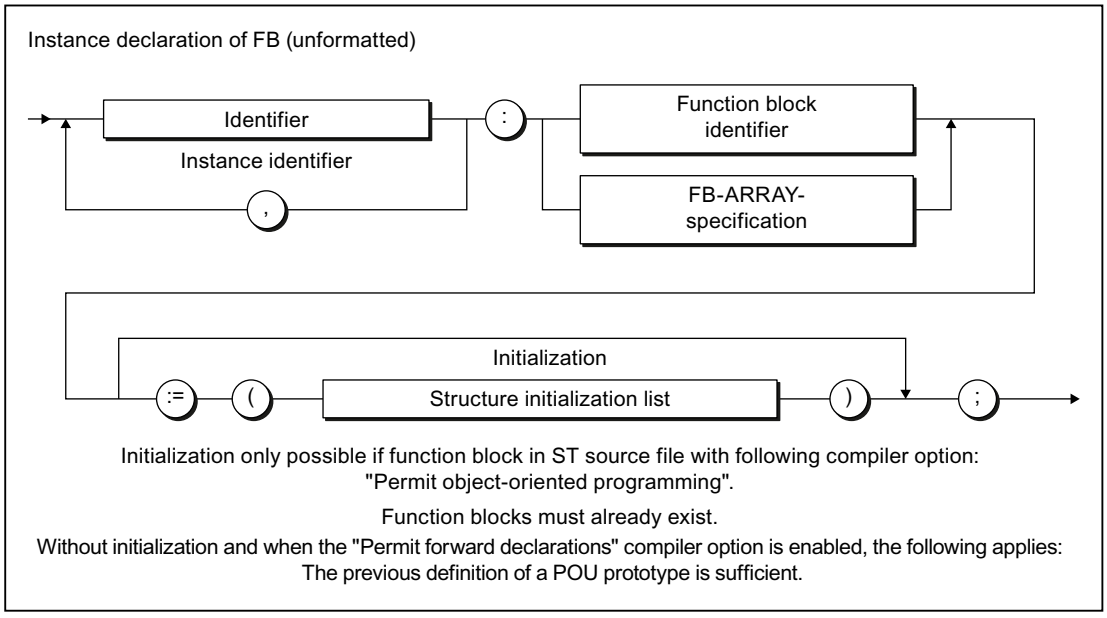


Figure 7-276 Syntax: Instance declaration of FB

Example of the instance declaration of an FB with initialization:

```
FB_inst : FB_name := (Var_publ1 := 12,
                      Var_publ2 := 123.456);
```

The instance declaration can also be an array, e.g.:

```
FB_inst : ARRAY [1..2] OF FB_name;
```

Note

Pay attention to the different initialization times for different variable types.

Calling an instance of a function block

You call a function block instance in the statement section of a POU (for information about syntax, see Figure). FB parameters are the input, in-out and output assignments separated by commas.

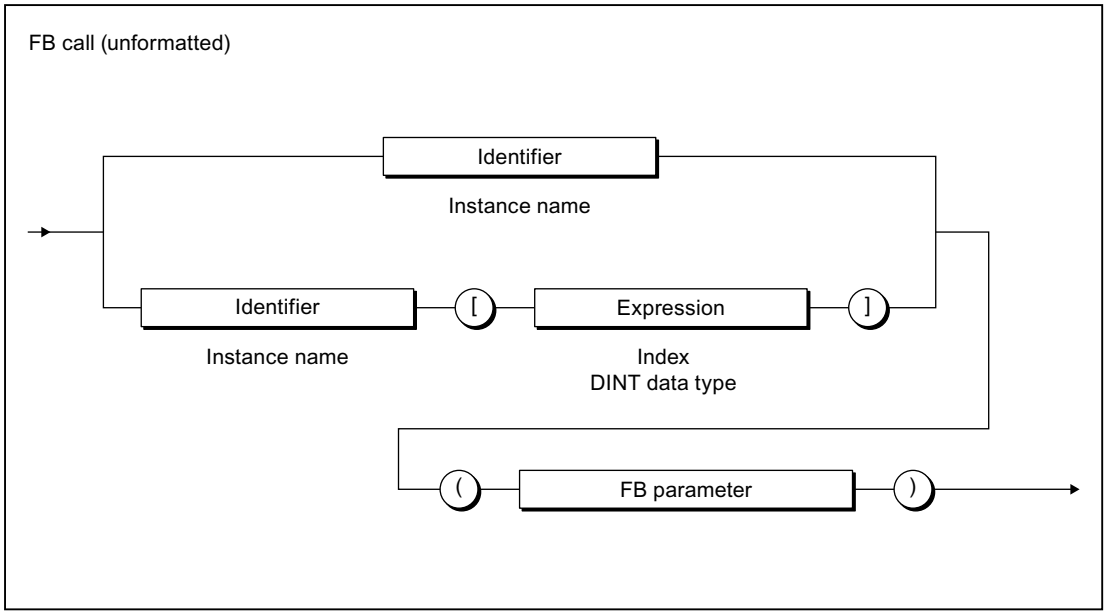


Figure 7-277 FB call syntax

The example in the following table is applicable, assuming that the *supply and motor* function blocks have already been defined:

- FB Supply:
  Input parameters in1, in2; in/out parameter inout; output parameter out
- FB motor:
  In/out parameters inout1, inout2; output parameters out1, out2

Table 7-416 Example of instance declaration, FB call, and access to output parameters

```
VAR
    Supply1, Supply2: Supply;
    Motor1 : Motor;
END_VAR

// Parameter transfer (output assignment) when calling the instance of an FB
Supply1 (in1 := var11, in2 := expr12, inout := var13, out => var14) ;
Supply2 (in1 := var21, in2 := expr22, inout := var23, out => var24) ;
Motor1 (inout1 := var31, inout2 := var32, out1 => var33, out2 => var34);
// ...

// Accessing the FB's output parameter outside the FB
var15 := Supply1.out;
var25 := Supply2.out;
var35 := Motor1.out1;
var36 := Motor1.out2;
var41 := Motor1.out1 * Motor1.out2 * (Supply1.out + Supply2.out);
```

Accessing the FB's output parameter outside the FB

In addition to the output assignment (Page 4752) for the call of an FB, it is always possible to access an FB's output parameter outside the FB.

To do so, use structured variables (Page 4682) in the *FB instance name.output parameter* format, e.g. *Supply1.out*.

Also refer to the examples in Calling function blocks (instance calls) (Page 4754).

The instance name of the FB itself must not be used in a value assignment!

Accessing the FB's input parameter outside the FB

In addition to the input assignment (Page 4750) for the call of an FB, it is always possible to read and write an FB's input parameter outside the FB.

To do so, use structured variables (Page 4682) in the *FB instance name.input parameter* format, e.g. *Supply1.in1*.

Note

To be able to use this option, the "Permit language extensions" compiler option must be activated, see Global compiler settings (Page 4623) and Local compiler settings (Page 4626).

The instance name of the FB itself must not be used in a value assignment!

Table 7-417 Example of assignment to input parameter

```
// Only with compiler option "Permit language extensions" activated
VAR
    var_fb   : _WORD_TO_2BYTE;
    var_word : WORD;
END_VAR
var_fb.wordin := var_word;
// ..
var_fb();
```

Access an FB's public variables outside of the FB

How to access a public local variable of a function block (i.e. variable with PUBLIC access identifier):

1. Declare an instance of the FB (unless this has already been done).
   See Calling function blocks (declaring and calling instances) (Page 4754).
2. For access purposes use a structured variable (Page 4708) in the format *FB-instancename.variablename*, e.g. *fb_inst.var_name*.

You can have read/write access to the public variable.

You access the public constants or data types of a function block in a similar manner.

Call methods within and outside of a FB

Inside a FB you can call all methods defined within this, irrespective of the access identifier (PUBLIC or PRIVATE). The call works in the same way as with a function, see Calling a function (Page 4753).

You can also call the public methods for a function block (i.e. methods with PUBLIC access identifier) outside of this FB, e.g. in programs, functions, function blocks or methods from other function blocks. Proceed as follows:

1. Declare an instance of the FB (unless this has already been done).
   See Calling function blocks (declaring and calling instances) (Page 4754).
2. To call the method, write the name of the FB instance and a period before the method name (similar to a structured variable (Page 4708)).
   The parameter transfer takes place as described in the Calling a function (Page 4753) section.

Error sources in FB calls

Note the following when calling a function block instance:

- **Only assign in/out parameters with variables that are stored directly in the memory.**
  Only the following variables are permissible actual parameters:
  - Global variables (unit variables and global device user variables)
  - Local variables
  - Variables of the data type of the TO (TO instances)

  The following are not possible, in particular:
  - System variables (TO variables)
  - Names of technological objects from the Engineering System
  - I/O variables
  - Absolute and symbolic process image access
- **Do not use functions (FCs) as in/out parameters.**
  The FC return value, i.e. the FC call, cannot be an actual parameter in an in/out assignment. You must first store the result of the FC in a local variable and then use this variable as an actual parameter in the in/out assignment.
- **Do not use constants as in/out parameters.**
  Only variables can be used as actual parameters of an in/out assignment because the value is written back.
- **In/out parameters cannot be initialized.**

7.2.5.2 Comparison of functions and function blocks

The differences between user-defined function blocks (FBs) and functions (FCs) are succinctly illustrated below using a complete example.

Description of example

The following example illustrates the differences between FBs and FCs. For simplicity, each type of parameter is used only once, although, in reality, you can define any number of parameters. The terms used are defined both in the detailed descriptions in Define functions (Page 4736) and Define function blocks (Page 4737).

A block will be created as an FB and an FC in the implementation section of an ST source file for use in calculating the circumference and the area of a circle for a radius input variable:

- An input parameter is defined for the radius.
- An in/out parameter is defined for the circumference of the circle, i.e. the value of the transferred variable is assigned directly during the call of the FB or the FC.
- There are several ways of defining the area of the circle for the FB and the FC:
  - For the FB, an output parameter is defined.
  - For the FC, its return value is used; the data type of the return value is defined appropriately.

- Each FB and FC call will be recorded in a counter (local variable). The explanations for the example state: We will see that this value will continue to be counted only in the FB.
- In the program section, the FB or the FC is called and the actual parameters assigned to the following formal parameters:
  - For the FB: Input, in/out and output parameters
  - For the FC: Input and in/out parameters.

  The values for the circumference and the area are available after calling the FB or the FC:
  - For the FB: in the actual parameters of the in/out and output parameter.
    The output parameter can be read even outside the FB.
  - For the FC: in the return value of the function and in the actual parameter of the in/out parameter.

Source file with comments

Table 7-418 Example of differences between FB and FC

Function block (FB)

```
INTERFACE
  PROGRAM CircleCalc1;
END_INTERFACE
IMPLEMENTATION
  FUNCTION_BLOCK Circle1
    // Constant declaration
    VAR CONSTANT
      PI : LREAL := 3.1415 ;
    END_VAR
    // Input parameter
    VAR_INPUT
      Radius : LREAL ;
    END_VAR
    // In-out parameter
    VAR_IN_OUT
      circumference : LREAL ;
    END_VAR
    // Output parameter
    VAR_OUTPUT
      Area : LREAL ;
    END_VAR
    // Local variables, static
    VAR
      Counter : DINT ;
      (* Variable retains its value
      between calls *)
    END_VAR
    // Call counter
    Counter := Counter + 1 ;
    Circumference := 2 * PI * Radius ;
    Area := PI * Radius**2 ;
  END_FUNCTION_BLOCK
  PROGRAM CircleCalc1
    VAR
      myCircle1        : Circle1 ;
      myArea1, myArea2 : LREAL ;
      myCircf          : LREAL ;
    END_VAR ;
    myCircle1(Radius := 3
      , Circumference := myCircf
      , Area => myArea1) ;
    myArea2 := myCircle1.Area ;
    // myCircf has the value 18,849
    // myArea1 has the value 28,274
    // myArea2 has the value 28,274
  END_PROGRAM
END_IMPLEMENTATION
```

Function (FC)

```
INTERFACE
  PROGRAM CircleCalc2;
END_INTERFACE
IMPLEMENTATION
  FUNCTION Circle2 : LREAL
    // Constant declaration
    VAR CONSTANT
      PI : LREAL := 3.1415 ;
    END_VAR
    // Input parameter
    VAR_INPUT
      Radius : LREAL ;
    END_VAR
    // In-out parameter
    VAR_IN_OUT
      circumference : LREAL ;
    END_VAR
    // Output parameter
    // Not possible


    // Local variables, temporary
    VAR
      Counter : DINT ;
      (* Variable will be initialized
      with 0 for each call *)
    END_VAR
    // Call counter
    Counter := Counter + 1 ;
    Circumference := 2 * PI * Radius ;
    Circle2 := PI * Radius**2 ;
  END_FUNCTION
  PROGRAM CircleCalc2
    VAR

      myArea  : LREAL ;
      myCircf : LREAL ;
    END_VAR ;
    myArea := Circle2(Radius := 3
      , Circumference := myCircf);


    // myCircf has the value 18,849
    // myArea has the value 28,274

  END_PROGRAM
END_IMPLEMENTATION
```

Table 7-419 Example of the differences between FB and FC for the previous example

Function block (FB)	Function (FC)
Comments	
Reserved words for the definition: FUNCTION_BLOCK and END_FUNCTION_BLOCK	Reserved words for the definition: FUNCTION and END_FUNCTION
No return value permitted.	The data type of the return value must be specified after the name (VOID data type, if no return value).
Input parameters can be used to transfer values to the FB.	Input parameters can be used to transfer values to the FC.
In/out parameters can be used to read and write the transferred variables in the FB.	In/out parameters can be used to read and write the transferred variables in the FC.
Output parameters can be used to return values from an FB.	No output parameters permitted.
The local variables are static, i.e. they retain their value between FB calls. The *Counter* local variable is incremented; its value is retained when the FB is closed. The variable is, therefore, incremented each time the FB is called. To see this behavior: Assign the value of the local variables to a global variable in the FB. Monitor the value of the global variable after repeated FB calls.	The local variables are temporary, i.e. they lose their value when the function is terminated. Although the *Counter* local variable is incremented, its value is lost when the FC is exited. The variable is reinitialized (to 0 in the example) at the next FC call. To see this behavior: Assign the value of the local variables to a global variable in the FC. The value of the global variable remains unchanged after repeated FC calls.
In the statement section, the results (return values) are assigned to the output or in/out parameters.	In the statement section, the result (return value) is assigned to the function name (except when VOID data type is specified).
In the declaration section of the block that executes the call, an instance of the FB is declared: you declare a variable and specify the name of the FB as its data type. You use the declared instance name to call the FB and to access its output parameters. The name of the FB itself must not be used in the statement section.	
• You assign a variable to the in/out parameters when the FB instance is called. • With the call, you can assign the output parameters to a variable. • You can read an FB's output parameters, even outside the FB. For this purpose, use structured variables in the following format: *FB-instancename.outputparameter*.	• You assign a variable to the in/out parameters when the FB instance is called. • To obtain the return value of the FC: – Assign the function to a variable. – Use the function in an expression on the right side of a value assignment.
The program that executes the call cannot access variables other than the in/out variables and output parameters of the FB. Exception: If the "Permit language extensions" compiler option is activated (see Global compiler settings (Page 4623) or Local compiler settings (Page 4626)), the calling program can also access the input parameters of an FB. For this purpose, use structured variables in the following format: *FB-instancename.inputparameter*.	The program that executes the call cannot access any variables other than the return value.

7.2.5.3 Programs

Programs are a series of statements placed between the PROGRAM and END_PROGRAM keywords.

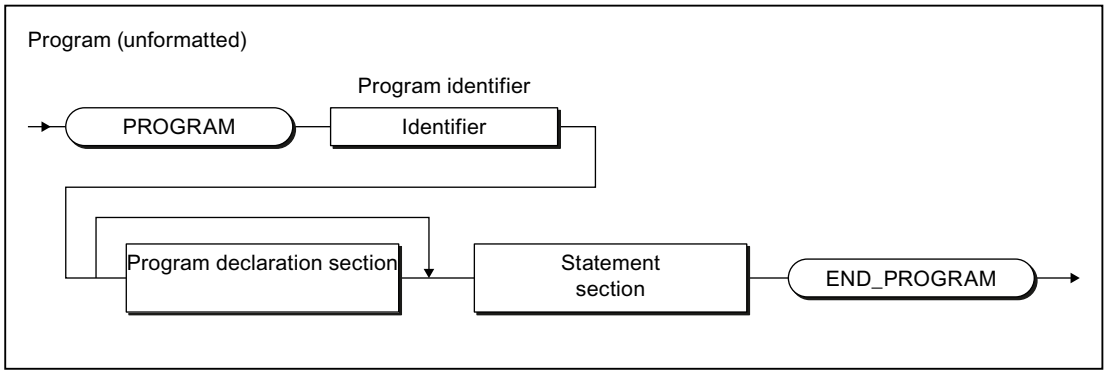


Figure 7-278 Syntax: Program

Programs are declared in the implementation section (Page 4809) of an ST source file and are comparable with the FB. Static local variables (VAR ... END_VAR), retentive local variables (VAR RETAIN ... END_VAR) or temporary local variables (VAR_TEMP ... END_VAR) can be created, for example. However, they do not have any formal parameters and so cannot be called with arguments. Examples of programs are contained in the Source file with comments (Page 4760) and Source text of the sample program (Page 4648) sections.

Assignment of a program in the execution system

By default, programs in the execution system are assigned to a task. The execution behavior of the programs, e.g. initialization of variables, is determined by the relevant task. For more information about the execution system and the tasks, refer to the *SIMOTION Basic Functions* Function Manual. This requires the program in the interface section (Page 4807) of the ST source file to be specified as a program organization unit to be declared public.

Calling a program in the program ("program in program")

Optionally, a program can also be called within a different program or a function block. This requires the following compiler options to be activated, see Global compiler settings (Page 4623) and Local compiler settings (Page 4626):

1. "Permit language extensions" for the program source of the calling program or function block and
2. "Create program instance data only once" for the program source of the calling program.

The call is performed as for a function with parameters and return value, see following example.

Note

The activated "Create program instance data only once" compiler option causes:

- The static variables of the programs (program instance data) to be stored in a different memory area (Page 4844). This also changes the initialization behavior (Page 4854).
- All called programs with the same name to use the same program instance data.

Table 7-420 Example for calling a program in a program

```
PROGRAM my_prog
    ; // ...
END_PROGRAM

PROGRAM main_prog
    ; // ...
    my_prog();
    ; // ...
END_PROGRAM
```

Note

Most of the programming work involved in assigning programs to tasks can be done if programs are called from within a program. In the execution system, only one calling program needs to be assigned to the associated tasks in each case.

7.2.5.4 Expressions

The expression is a special case of a function declaration:

- The data type of the return value is defined as BOOL and is not specified explicitly.

It is used in conjunction with the WAITFORCONDITION statement (Page 4729).

An expression can only be declared in the implementation section of the ST source file.

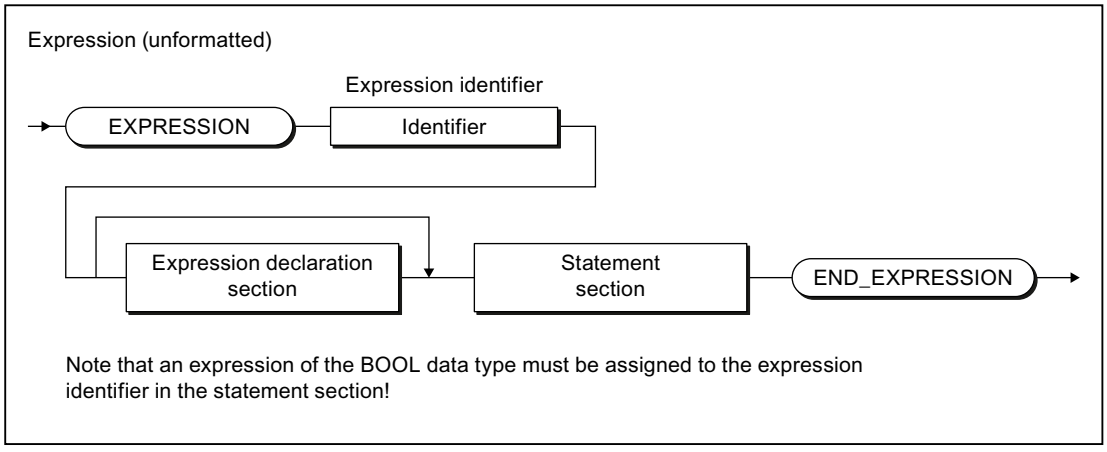


Figure 7-279 Syntax: Expression

Optionally, the following can be declared in the declaration section:

- Local (temporary) variables
- Local constants
- User-defined data types (UDT)
- Input and in/out parameters (as of version V4.1 of the SIMOTION Kernel)

The following can be accessed in the statement section:

- The local variables of the expression
- The input and in/out parameters (provided their declaration is permitted)
- Unit variables
- Global device variables, I/O variables, and the process image

An expression of data type BOOL must be assigned to the expression name in the statement section of the expression (see figure).

Note

The statement section of the expression cannot contain any function calls or loops.

Example

The following example assumes that the feeder program is running in a MotionTask. The option Activation after StartupTask is selected for this MotionTask. The assignment of programs to tasks is performed in SIMOTION SCOUT (see SIMOTION Motion Control Basic Functions Description of Functions).

Table 7-421 Example of an EXPRESSION and the WAITFORCONDITION statement

```
INTERFACE
    USEPACKAGE cam;
    PROGRAM feeder; // in MotionTask_1
END_INTERFACE

IMPLEMENTATION
    // Condition for WAITFORCONDITION statement
    EXPRESSION automaticExpr
        automaticExpr := IOfeedCam; // Digital input
    END_EXPRESSION

    PROGRAM feeder
        VAR
            retVal : DINT ;
        END_VAR ;
        retVal := _enableAxis (axis := realAxis,
            enableMode := ALL,
            servoCommandToActualMode := INACTIVE,
            nextCommand := WHEN_COMMAND_DONE,
            commandId := _getCommandId() );

        // Wait until the start condition is satisfied
        WAITFORCONDITION automaticExpr WITH TRUE DO
            // High-priority execution of all statements
            // to the END_WAITFORCONDITION command
            retVal := _pos (axis := realAxis,
                positioningMode := RELATIVE,
                position := 500,
                velocityType := DIRECT,
                velocity := 300,
                velocityProfile := TRAPEZOIDAL,
                mergeMode := IMMEDIATELY,
                nextCommand := WHEN_MOTION_DONE,
                commandId:= _getCommandId() );
        END_WAITFORCONDITION;

        retVal := _disableAxis (axis := realAxis,
            disableMode := ALL,
            servoCommandToActualMode := INACTIVE,
            nextCommand := WHEN_COMMAND_DONE,
            commandId := _getCommandId() );
    END_PROGRAM
END_IMPLEMENTATION
```

Further examples are contained in the SIMOTION Motion Control Basic Functions Function Manual. In particular, the manual describes how, as of version V4.1 of the SIMOTION Kernel, you

use an EXPRESSION with parameters and, for example, program a time monitoring in a WAITFORCONDITION statement.

7.2.6 Object-oriented programming - OOP (as of kernel V4.5)

7.2.6.1 Important note on object-oriented programming

As of version V4.5 the Structured Text programming language provides options for object-oriented programming.

You can use the object-oriented programming in individual or in all ST source files under the following conditions.

Conditions for object-oriented programming

The following conditions must be met in order to use the object-oriented programming in an ST source file:

- The higher-level SIMOTION device must be configured with a version of the SIMOTION Kernel as of V4.5.
- The corresponding version of the SIMOTION Kernel must be installed on the target device.
- The ST source file must be compiled with the compiler option **Permit object-oriented programming**, see Global compiler settings (Page 4623) and Local compiler settings (Page 4626).

Note

When using the compiler option **Permit object-oriented programming** there is an expanded list of protected identifiers (Page 4658) and of reserved identifiers (Page 4663)in the ST programming language.

The aforementioned conditions apply to the entire section **Object-Oriented Programming – OOP** in this Manual.

Further information and usage examples of object-oriented programming can be found in the following references:

Braun, Michael / Horn, Wolfgang: Objektorientiertes Programmieren mit SIMOTION (Object-Oriented Programming with SIMOTION) Grundlagen, Programmbeispiele und Softwarekonzepte nach IEC 61131-3 (Basic Principles, Example Programs and Software Concepts according to IEC61131-3) 1st edition, Erlangen: Publicis Publishing 2016. ISBN 978-3-89578-455-2.

7.2.6.2 Classes and methods

Creating classes (basic classes)

Defining classes (basic classes)

You define a class in the declaration part of the implementation section wherever possible before the section of the source file (program, FB, FC or class) in which it is called.

Use the following syntax:

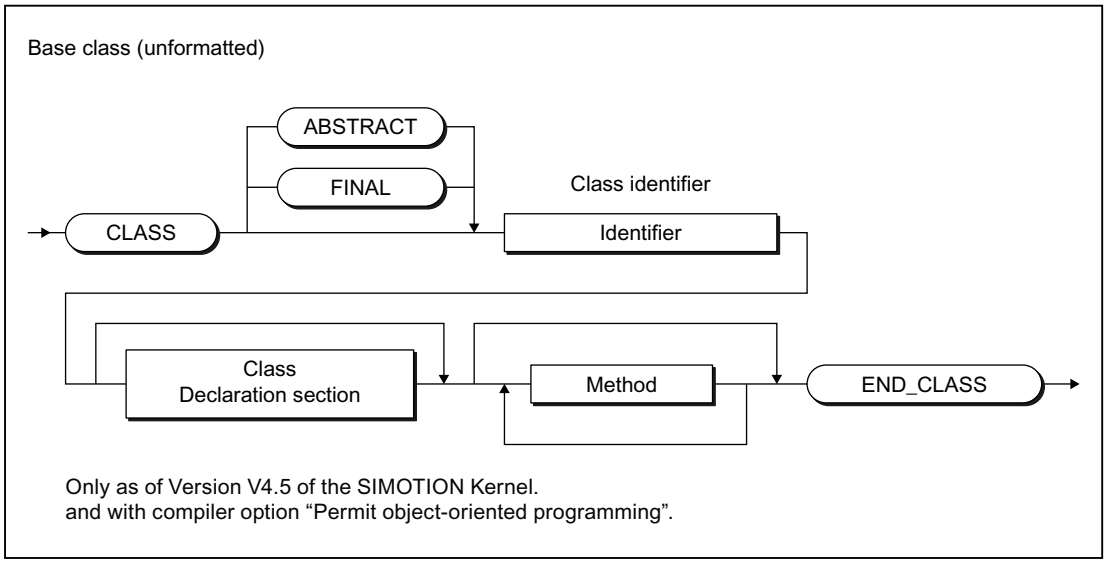


Figure 7-280 Syntax: Base class

Enter an identifier as the class name after the keyword CLASS and optionally FINAL or ABSTRACT.

Then enter, see example in Table 7-428 Example: Up and Down counter as a class (Page 4777):

- The optional declaration section (Page 4767)
- The methods
- Keyword END_CLASS

Declaration subsection of a class

A declaration section is subdivided into various declaration blocks that are each identified by a separate pair of keywords. Each block contains a declaration list for similar data, such as constants and local variables. Each block type may occur multiple times, any sequence is possible for the blocks.

Permissible declaration blocks

Table 7-422 Declaration blocks of a class: Options

Data	Syntax
Data type	**TYPE** *Accessidentifier_class* (optional) *Declaration list* **END_TYPE**
Constant	**VAR CONSTANT** *Accessidentifier_class* (optional) *Declaration list* **END_VAR**
Local variable (static)	**VAR** *Accessidentifier_class* (optional) **OVERRIDE** (optional) *Declaration list* **END_VAR**
Retentive variable	**VAR RETAIN** *access identifier* (optional) **OVERRIDE** (optional) *Declaration list* **END_VAR** (Only PROTECTED or PRIVATE access identifier permitted, default PROTECTED.)
Accessidentifier_class: **PUBLIC**, **PROTECTED**, **PRIVATE** or **INTERNAL**, default **PROTECTED**; see section "Access identifiers within classes" (Page 4768). **OVERRIDE**: Specifying this keyword enable an override of the initialization values for values with the access identifier PROTECTED or PRIVATE for the instance declaration (Page 4778). *Declaration list*: The list of identifiers of the type to be declared	

Access identifier within classes

An access identifier may be stated within classes with declaration blocks or methods:

- With declaration blocks:
  Directly after the keyword for the relevant block (e.g. TYPE, VAR).
  The access identifier is valid for all identifiers within this declaration block.
- With methods:
  Directly after the keyword METHOD.
  The access identifier is valid for the relevant method.

You use the access identifier to determine whether the identifier or the method can only be used within the class, in its derivations or outside of the class, see following table.

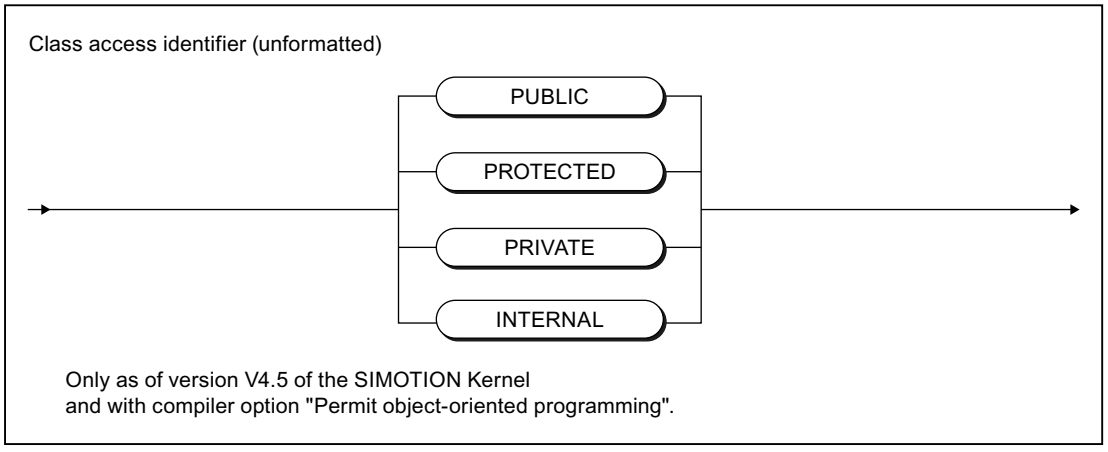


Figure 7-281 Syntax: Class access identifier

Table 7-423 Possible access identifiers within a class

Access identifier	Meaning
PRIVATE	The identifiers for the relevant declaration block (variables, data types, etc.) or the method can only be used within this class. These identifiers or the method cannot be used in derived classes.
PROTECTED	The identifier for the relevant declaration block (variables, data types, etc.) or the method can be used within this class and in classes derived from this. This is the default if there is no access identifier stated.
INTERNAL	Only when the identifier is defined within an object-oriented namespace (Page 4904). The identifiers of the relevant declaration block (variables, data types, etc.) or the method can be used only within the higher-level namespace. Within the higher-level namespace, the following apply: • The declared identifiers can also be accessed outside of the class. This does not apply to retentive local variables (VAR RETAIN). • The method can also be called outside of the class.
PUBLIC	The identifiers for the relevant declaration block (variables, data types, etc.) or the method are public. • The declared identifiers can also be accessed outside the class and outside the higher-level object-oriented namespace. • The method can also be called outside the class and outside the higher-level object-oriented namespace. This access identifier is not permitted in declaration blocks for retentive local variables (VAR RETAIN / END_VAR).

Creating methods

Defining methods

The executable instructions of a class are summarized in methods. Within a class the methods are defined after the declaration subsection for the class.

A method roughly corresponds to a function (Page 4736). Specification of an identifier determines the environment from which the method can be called.

If the compiler option (Page 4623) "Allow forward declaration" is not activated a method must be defined before the program organization unit (program, FB, FC or class) or method in which it is called.

Use the following syntax:

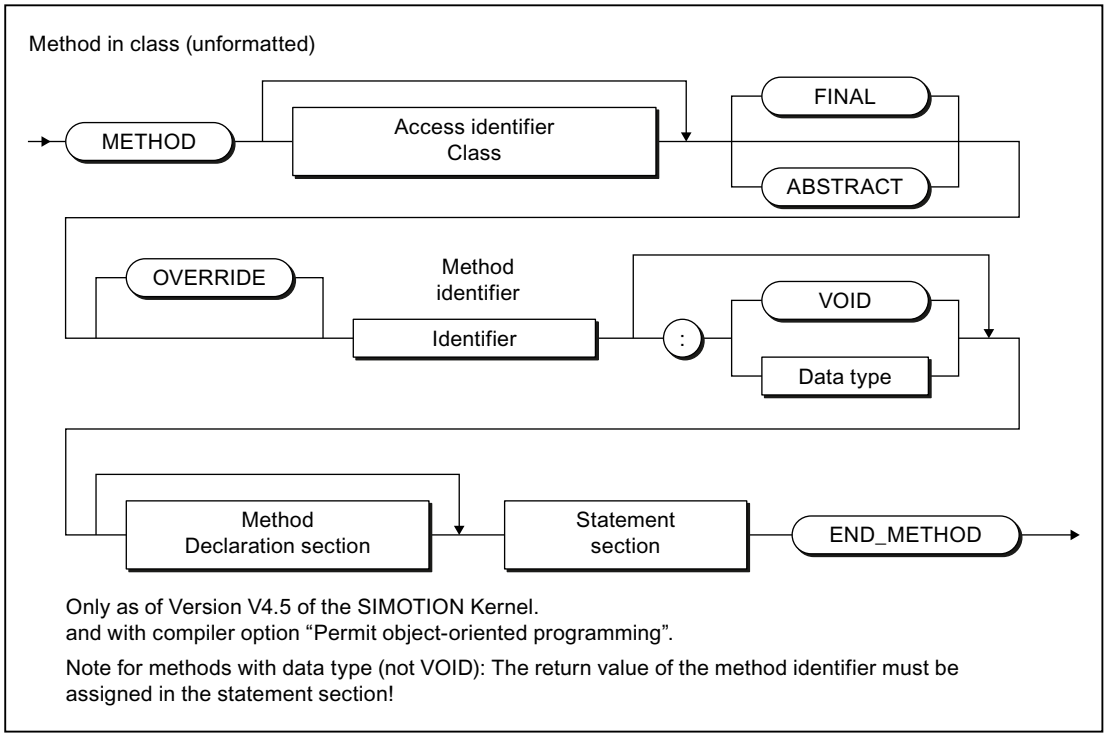


Figure 7-282 Syntax: Method

Enter in the following sequence, see Table 7-428 Example: Up and Down counter as a class (Page 4777):

- The METHOD keyword.
- the optional class access identifier (PUBLIC, PROTECTED or PRIVATE), default PROTECTED. See section "Access identifiers within Classes" (Page 4768) for access identifiers.
- The keywords FINAL, ABSTRACT and OVERRIDE are only relevant for methods in classes that are derived or are to be derived:
  - FINAL: Method cannot be overridden in a derived class.
  - ABSTRACT: Method cannot be called, it must be overridden in a class to be derived. Furthermore, this keyword renders the class in which the method is defined an abstract class. In this case, it is not possible to declare an instance of this class, see "Abstract classes" (Page 4790).
  - OVERRIDE: Method overrides the existing method with the same name in a derived class.
- An identifier as method name.
- if the method has a return value, the data type for the return value following a colon. The symbolic data type VOID can also be entered for a method with no return value.

This is then followed by:

- The optional declaration section (Page 4771)
- The statement section (Page 4773) (except in methods with keyword ABSTRACT)
- The END_METHOD keyword

Declaration subsection of methods

A declaration section is subdivided into various declaration blocks that are each identified by a separate pair of keywords. Each block contains a declaration list for similar data, such as constants, local variables and parameters. Each type of block may only appear once; the blocks may appear in any order.

The following options are then available for the declaration section of a method:

Permissible declaration blocks

Table 7-424 Declaration blocks for methods: Options

Data	Syntax
Constant	**VAR CONSTANT** *Declaration list* **END_VAR**
Input parameters	**VAR_INPUT** *Declaration list* **END_VAR**
In/out parameter	**VAR_IN_OUT** *Declaration list* **END_VAR**
Output parameters	**VAR_OUTPUT** *Declaration list* **END_VAR**
Local variable (temporary)	**VAR** or **VAR_TEMP** *Declaration list* **END_VAR**
Declaration list: The list of identifiers of the type to be declared	

Parameter blocks

Parameters are local data and are formal parameters of a method. When the method is called, the formal parameters are substituted by the actual parameters, thus providing a means of exchanging information between the called and calling source file sections.

- Formal input parameters receive the actual input values (data flow inwards).
- Formal output parameters are used to transfer output values (data flow outwards).
- Formal in/out parameters act as input and output parameters.

The following figures show the syntax for the parameter declaration of a method. This is identical to the syntax for the parameter declaration of a function (FC).

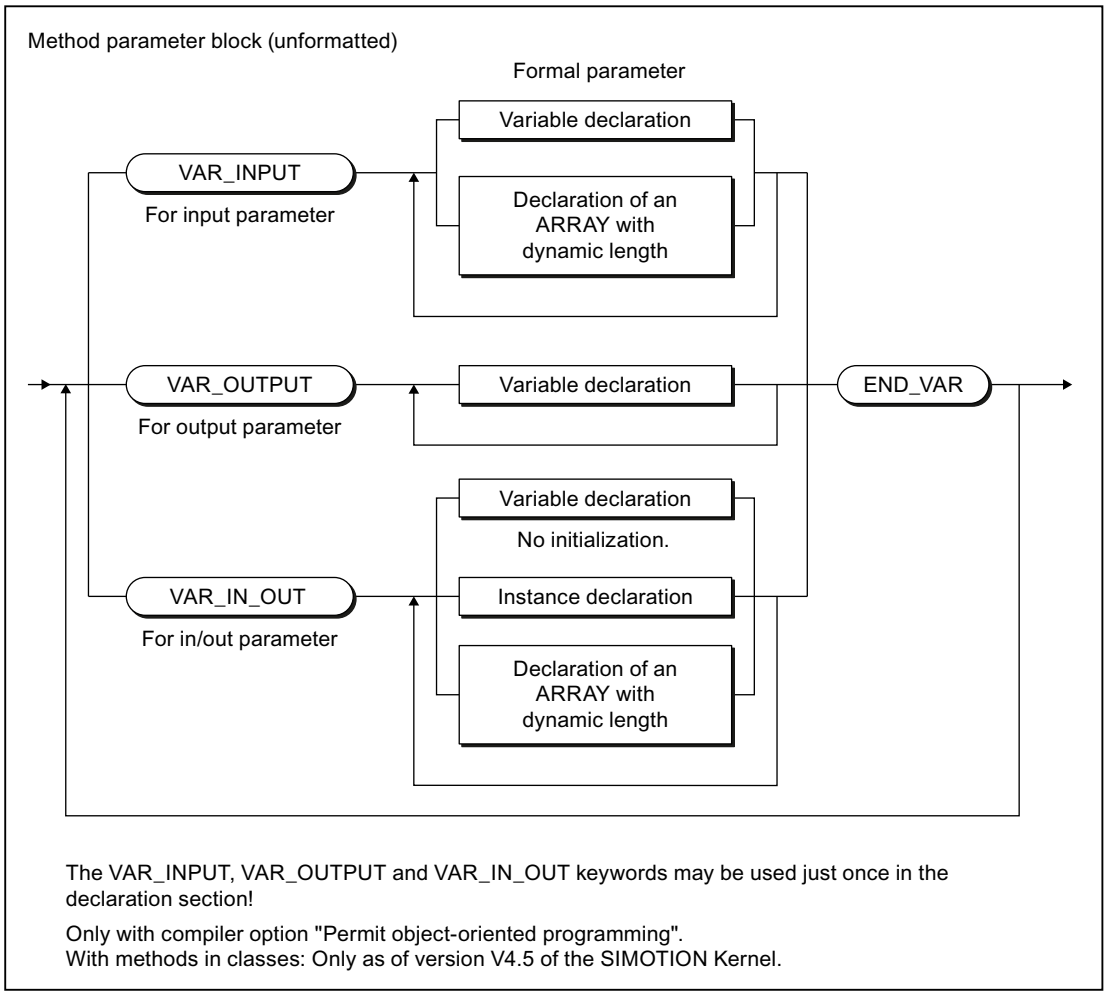


Figure 7-283 Syntax: Method parameter block

Apart from variables you can also declare the following in the parameter blocks:

- in the parameter block for in/out parameters:
  - Instances of function blocks (Page 4754)
  - Instances of classes (Page 4778)
  - Arrays with a dynamic length (Page 4747), as of version V4.2 of SIMOTION Kernel.
- in the parameter block for input parameters:
  - Arrays with a dynamic length (Page 4747), as of version V4.2 of SIMOTION Kernel.

You can use the declared parameters in the same way as other variables within a method, with the following exception: You cannot assign values to input parameters.

From outside the method you can only access the parameters by calling the method (including return value).

Statement section of methods

The statement section of a method contains statements that are executed when the method is called. There is no difference compared to the formal rules for creating a statement section; however, you should note the information in the following table.

Note

For tips on the efficient use of parameters, please refer to the Runtime-optimized Programming section in the SIMOTION Basic Functions Function Manual.

Table 7-425 Use of parameters and variables in methods

Parameter/variable	Use
Input parameters	With the call of a method, assign the current values to the input parameters. These values are used for data processing within the method, for example, for calculations, but cannot be modified themselves.
In/out parameter	You assign a variable to an in/out parameter for the call of the method. The method accesses this variable directly and can change it directly. Type conversions are not supported. The variable assigned to an in/out parameter must be able to be read and written directly. Therefore, system variables (of the SIMOTION device or a technology object), I/O variables or process image accesses cannot be assigned to an in/out parameter.
Output parameters	You assign a variable to an in/out parameter for the call of a method using the **=>** operator. The value of the output parameter (result) is transferred to the variables when the method is closed.
Local variables	Local variables are variables that are declared and used only within the block. All local variables (VAR ... END_VAR or VAR_TEMP ... END_VAR) are temporary in methods, i.e. they lose their value when the method is terminated. The next time the method is called, they are reinitialized.

Note

The statement section is omitted from methods that are declared with the keyword ABSTRACT.

Calling methods within the class

Calling methods

Methods can be called in other methods within a class. As with functions, the call is made by specifying the method identifier and the parameter list (FC parameters).

The method can also be specified more precisely by stating the keywords THIS or SUPER or by stating a class identifier, see syntax in the following figure as well as the following table.

- The following applies to methods with no return value or with data type VOID:
  The method call is an instruction within a different method.
- The following applies to methods with return values:
  The method call is part of an expression on the right side of a value assignment.

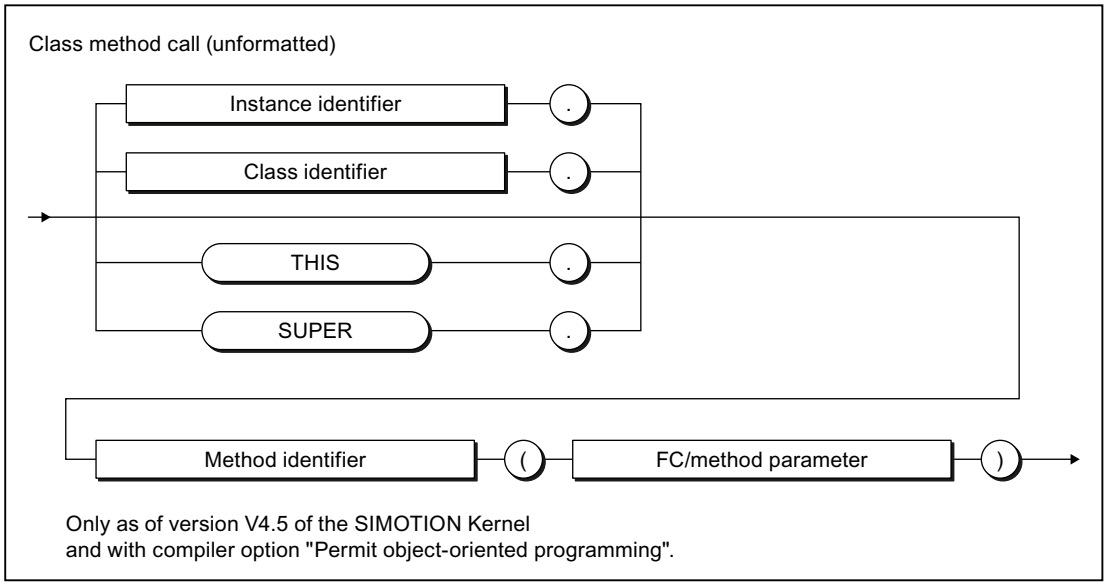


Figure 7-284 Syntax: Class method call

Table 7-426 Description of the method specification

Method specification	Description
(without)	Call of a method in a separate class with static binding. This means: If the stated method is overridden and called in a derived class, the original method is still called. Only methods that are defined in a separate class can be specified (i.e. not methods inherited from the base class).
THIS.	Call of a method in a separate class with dynamic binding. This means: If the stated method in a derived class is overridden and called, the current (overriding) method is called. All methods that are recognized in the class, including methods inherited from the base class, can be specified.
SUPER.	With derived classes: Explicit call of a method of the base class (static binding).
Class identifier.	Explicit call of a method of the stated class (static binding). Permitted class identifiers: Separate class and all base classes within a method implementation
Instance identifier.	Call of a method outside of the class, see Calling public methods of a class outside of this class (Page 4780).

Parameter transfer to input parameters

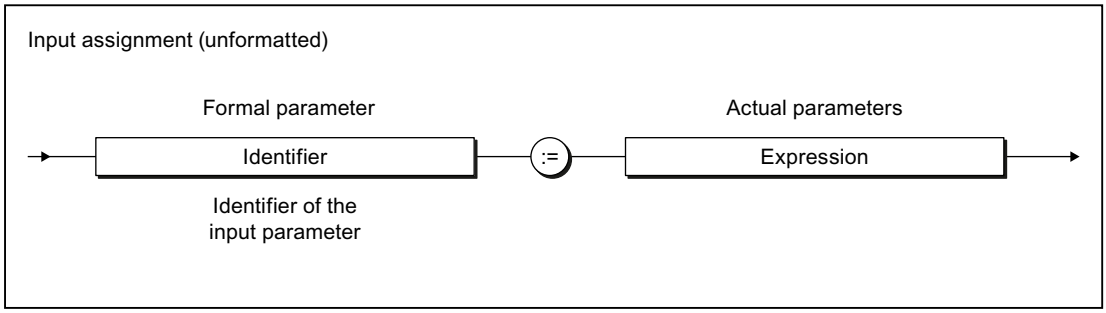


Figure 7-285 Syntax: Input assignment

You transfer the data (actual parameters) to the formal input parameters of a method by means of input assignments. You can specify the actual parameters in the form of expressions. You can use the formal input parameters in statements within the method, but you cannot modify their values.

The assignment of an actual parameter is optional when an initialization expression was specified for the declaration of the formal parameter.

See also Table 7-428 Example: Up and Down counter as a class (Page 4777).

Parameter transfer to in/out parameters

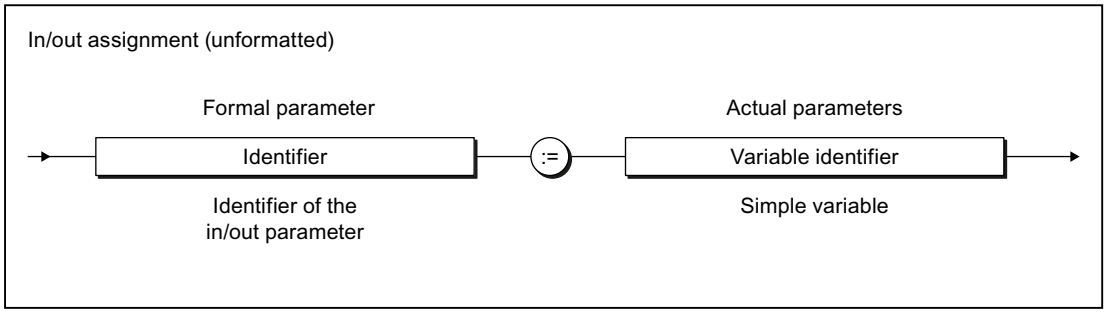


Figure 7-286 Syntax: In/out assignment

You transfer the data (actual parameters) to the formal in/out parameters of a method by means of in/out assignments. You can only assign a variable of the same type to the formal in/out parameter, data type conversions are not possible.

You can use and change the formal in/out parameters in statements within the method. The method accesses the variable of the actual parameter directly and can change it directly.

The variable assigned to an in/out parameter must be able to be directly read and written. Therefore, system variables (of the SIMOTION device or a technology object), I/O variables or process image accesses cannot be assigned to an in/out parameter.

When using the STRING data type in in/out assignments, the declared length of the actual parameter must be greater than or equal to the length of the formal in/out parameter (see following example).

Table 7-427 Example of the use of the STRING data type in in/out assignments

```
METHOD REF_STRING
    VAR_IN_OUT
        io : STRING[80];
    END_VAR
    ;    // Statements
END_METHOD

METHOD test
    VAR
        str1 : STRING [100];
        str2 : STRING [50] ;
    END_VAR
    REF_STRING (io := str1);    // Permissible call
    REF_STRING (io := str2);    // Illegal call,
                                // Compiler error message
END_METHOD
```

Please note the different parameter access times!

Parameter transfer to output parameters

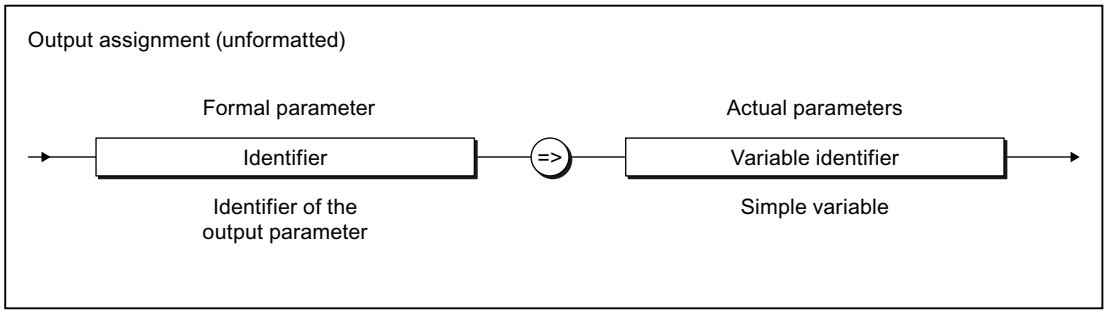


Figure 7-287 Syntax: Output assignment

You use an output assignment to assign the formal output parameters of a method to the variables (actual parameter) that adopt the value of the formal output parameter when the method is closed.

You can use and change the formal output parameters in statements within the method.

See also example: Table 7-429 Example: Call of the methods in the COUNTER class (Page 4781)

Output assignments are optional for the parameter transfer. If there is no output assignment for an output parameter, the values for the output parameters are lost once the method is terminated.

Example classes and methods

Example: Counter as a class

The classes and methods are explained in the following example using a counter.

In the COUNTER class the two methods UP and DOWN are defined for counting a variable CV up or down. They have the access identifier PUBLIC so that they can be called from outside, e.g. from a program.

The UP method increases the CV counter variable with every run-through by the input parameter INC (default = 1) until the upper limit value MAX_Val is reached.

The DOWN method decreases the CV counter variable with every run-through by the input parameter DEC (default = 1) until the lower limit value MIN_Val is reached. It calls the UP method internally and transfers the negative value of the input parameter DEC to it. Adding THIS. before the call for the UP method means that if the UP method is exceeded in a derived class the UP method effective there is called.

The declaration of the variables MAX_Val and MIN_Val with the keyword OVERRIDE allows the initialization values for the instance declaration of the class to be changed.

The return value for both methods contains the current value of the counter variables CV. The output parameter QU signals whether the upper limit value MAX_Val has been reached or the lower limit value MIN_Val has not been reached as relevant.

Table 7-428 Example: Up and Down counter as a class

```
CLASS COUNTER
    VAR
        CV : INT; // Current value of the counter
    END_VAR
    VAR OVERRIDE
        MAX_Val : INT := 100;
        MIN_Val : INT := 0;
    END_VAR

    // Method for incrementing by INC
    METHOD PUBLIC UP: INT
        VAR_INPUT
            INC : INT := 1;
        END_VAR
        VAR_OUTPUT
            QU : BOOL;
        END_VAR
        // Detection of upper limit
        IF CV <= MAX_Val - INC THEN
            CV := CV + INC; // Increment of the counter CV
            QU := FALSE;
        ELSE
            QU := TRUE; // Upper level limit reached
        END_IF;
        UP := CV; // Result of the method
    END_METHOD
```

```
    // Method for decrementing by DEC
    METHOD PUBLIC DOWN : INT
        VAR_INPUT
            DEC : INT := 1;
        END_VAR
        IF CV > MIN_Val THEN
            // Internal call for the UP method
            DOWN := THIS.UP (INC := - DEC);
        END_IF;
    END_METHOD
END_CLASS
```

Creating and using instances of classes

Declaring an instance of a class

Declaring an instance of a class

Before using methods or variables of a class you must declare an instance of the class. You declare a variable and enter the name of the class as the data type. You declare this instance:

- locally:
  within VAR / END_VAR in the declaration section of a program, a function block or a class
- globally:
  within VAR_GLOBAL / END_VAR in the interface section or implementation section of the ST source file
- as an in/out parameter:
  within VAR_IN_OUT / END_VAR in the declaration section of a function block, a function or a method

It is in all cases possible to declare the instance of a class in a program source at which the compiler option (Page 4623) "Permit object-oriented programming" is **not** active.

Exception: The compiler option (Page 4623) "Permit object-oriented programming" **must** be active at the ST source file in the following cases:

- With local declaration within classes.
  The access identifier class (Page 4768) (PUBLIC, PROTECTED or PRIVATE) can also be stated as an option after the keyword VAR.
- With local declaration within function blocks as long as the access identifier FB (PUBLIC or PRIVATE) is stated after the keyword VAR.
- With declaration as an in/out parameter in methods.

Instance-specific initialization of individual local variables of the class is possible. You thereby override the initialization values that were stipulated with the declaration of the class. Specification of the variables to be initialized is similar to the process for initializing a structure, and involves specifying a structure initialization list (Page 4699) in brackets.

You can in all cases initialize public variables of the class (i.e. variables with the PUBLIC access identifier) specific to the instance. In order to be able to initialize protected or private variables

of the class (i.e. variables with PROTECTED or PRIVATE access identifiers), the keyword OVERRIDE must be stated at their declaration block after the access identifier.

The following generally applies: The class must be declared in the ST source file before an instance can be declared.

Exception: It is sufficient to define a POU prototype (Page 4929) beforehand if initialization is not specified and when compiler option (Page 4623) "Permit forward declarations" is activated.

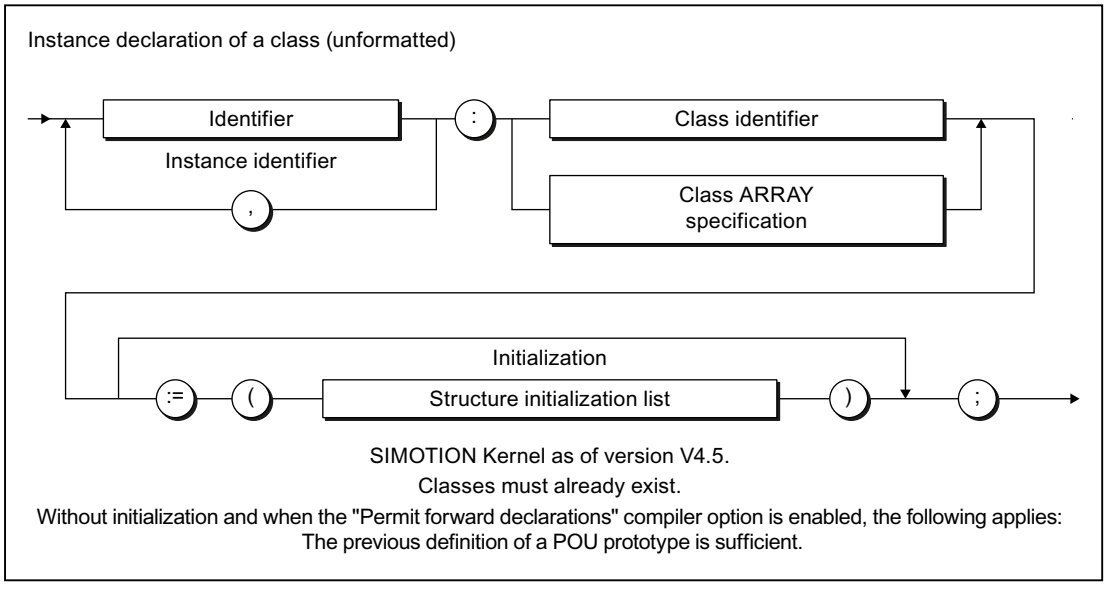


Figure 7-288 Syntax: Instance declaration of a class

Example of the instance declaration of a class:

```
Class_inst : Class_name := (Var_publ1 := 12,
                            Var_publ2 := 123.456);
```

The instance declaration can also be an array, e.g.:

```
Class_inst : ARRAY [1..2] OF Class_name;
```

Note

Pay attention to the different initialization times for different variable types.

Calling public methods of a class outside of this class

You can also call the public methods for a class (i.e. methods with PUBLIC access identifier) outside of this class, e.g. in programs, functions, function blocks or methods from other classes. Proceed as follows:

1. Declare an instance of the class (unless this has already been done).
   See Declaring an instance of a class (Page 4778).
2. To call the method, write the name of the method instance and a period before the method name (similar to a structured variable (Page 4708)).
   The parameter transfer takes place as described in the Calling methods within the class (Page 4773) section.

Accessing the public variables of a class outside of the class

How to access a public local variable of a class (i.e. variable with PUBLIC access identifier):

1. Declare an instance of the class (unless this has already been done).
   See Declaring an instance of a class (Page 4778).
2. For access purposes use a structured variable (Page 4708) in the format *class-instancename.variablenname*, e.g. *Class_inst.var_name*.

You can have read/write access to the public variable.

Access the public constants or data types of a class in a similar manner.

Example: Call of the counter method

The following program uses the COUNTER class declared in Table 7-428 Example: Up and Down counter as a class (Page 4777). It calls the methods UP and DOWN defined there. A C1 instance of the COUNTER class is declared for this, with the initialization values of the variables MAX-Val and MinVal modified in this process. Provided that Locking = FALSE, method C1.UP is called. Provided that the upper value is reached (UpValreached = TRUE), Locking becomes TRUE and the method C1.DOWN is called until the lower value is reached.

This program must be assigned to a cyclic task of the execution system (e.g. BackgroundTask).

The complete ST source file includes the class COUNTER from Table 7-428 Example: Up and Down counter as a class (Page 4777) in front of the program CallCounter.

Table 7-429 Example: Call of the methods in the COUNTER class

```
PROGRAM CallCounter_ST
    VAR
         C1 : COUNTER := (MAX_Val := 1000, MIN_Val := 0); // Instance
         CountOut     : INT;
         Locking      : BOOL;
         UpValreached : BOOL;
    END_VAR

    // Call of methods for incrementing and decrementing
    IF UpValreached = TRUE THEN
        Locking := TRUE;
    END_IF;
    IF Locking = FALSE THEN
        CountOut := C1.UP (QU => UpValreached); // Increment
    END_IF;
    IF Locking = TRUE THEN
        CountOut := C1.DOWN (); // Decrement
    END_IF;
    IF CountOut <= 0 THEN
        Locking      := FALSE;
        UpValreached := FALSE;
    END_IF;
END_PROGRAM
```

7.2.6.3 Inheritance of classes and methods

Inheritance of classes

In object-oriented programming, it is possible to derive subclasses from an existing class (base class). When a subclass is derived, it inherits all the properties of the base class, although the source code is not provided in the derived class:

- all data types that were declared in the base class,
- all variables and constants that were declared in the base class,
- all methods that were declared in the base class,
- all instance declarations of classes and function blocks within the base class.

The complete syntax diagram of a class definition is provided below.

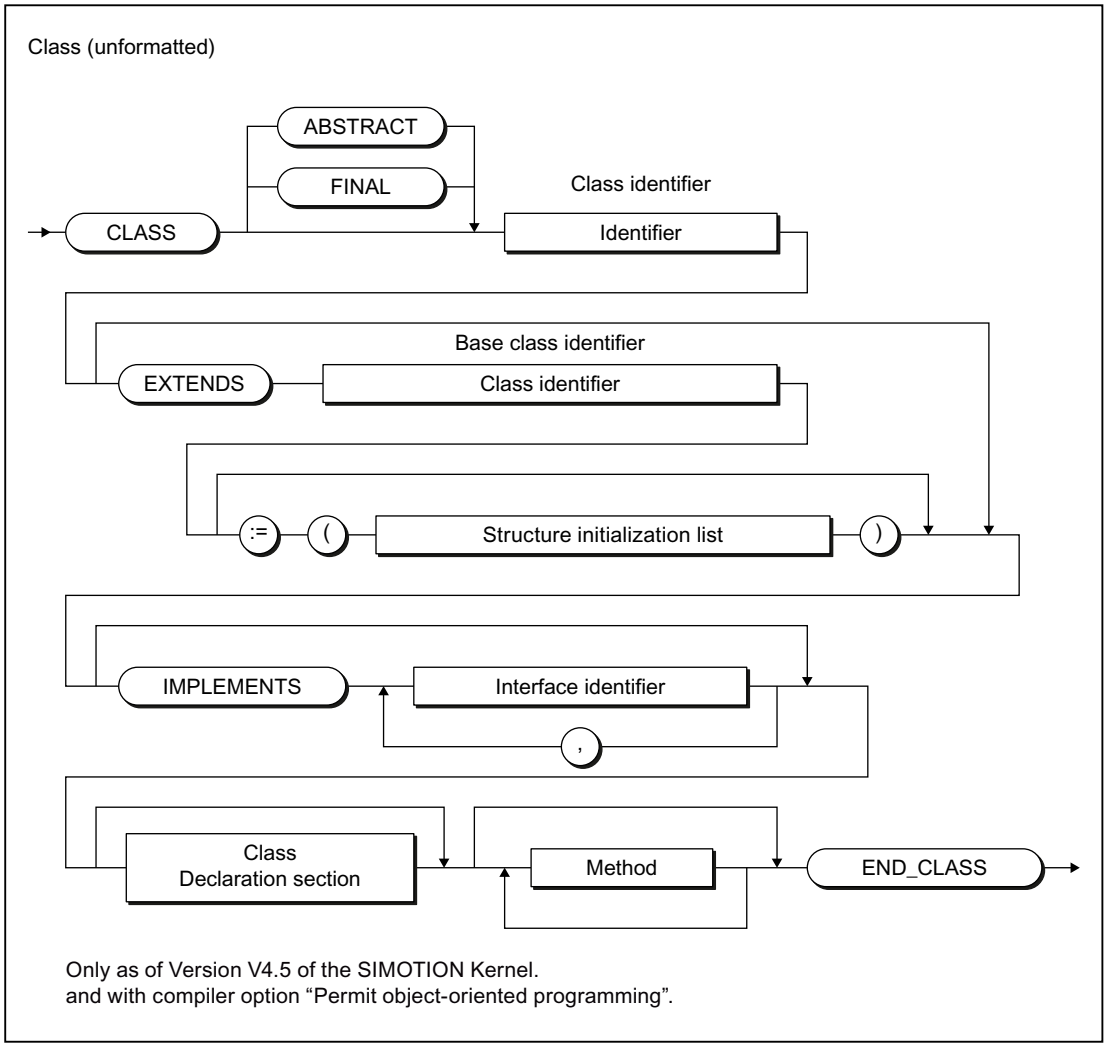


Figure 7-289 Syntax: Class

Use the keyword EXTENDS with the identifier for the base class in order to derive a class.

The properties for the derived class can now be adapted to the new circumstances:

- The initialization values for the variables from the base class can be modified:
  - public or protected variables (access identifiers PUBLIC or PROTECTED) without restrictions.
  - private variables (access identifier PRIVATE) only when the keyword OVERRIDE has been stated at their declaration block after the access identifier.
- Additional data types, variables and constants can be declared.
- Additional instances of classes and function blocks can be declared.
- Additional methods can be defined.
- Existing methods can be overridden.

Note

Data types, variables and constants that have been declared with the access identifier PRIVATE in the base class cannot be accessed in the derived class. Access is only possible via methods inherited from the base class.

Methods that have been defined with the access identifier PRIVATE in the base class cannot be called in the derived class. They can only be called from methods inherited from the base class.

Overriding methods

Public or protected methods of the base class (access identifiers PUBLIC or PROTECTED) can be overridden in a derived class.

For this proceed as described in the “Defining methods" (Page 4769) section and use the keyword OVERRIDE.

The following must be consistent in the overriding method and the methods from the base class:

- Access identifier
- Method name
- Data type of the return value
- Parameters in the declaration subsection of the method:
  - Type (input, in/out, output parameters)
  - Number and sequence
  - Identifier
  - Data type

The overriding method and the method from the base class may be different:

- with the temporary variables (VAR / END_VAR or VAR_TEMP / /END_VAR)
- In the statement section
  You can also call the same method from the base class in the statement section. Use the keyword SUPER or the identifier for the base class. See “Calling methods" (Page 4773).

The following methods from the base class cannot be overridden:

- Methods with the keyword FINAL.
- Methods with the access identifier PRIVATE.

The keyword ABSTRACT can only be used in the overriding method if it was also used in the method for the base class. Note that a class is abstract if it contains at least one abstract method. No instances can be declared from abstract classes.

Example: Counter in steps of 5 through inheritance

The following example is based on the COUNTER class from the Table 7-428 Example: Up and Down counter as a class (Page 4777) example. It shows how to obtain a COUNTER_5STEP class whose methods count up and down in steps of 5 by deriving from the COUNTER class and overriding its methods.

With the definition of the COUNTER_5STEP class the keyword EXTENDS signifies that this class is derived from the COUNTER class (= base class). The derived class inherits from the base class all variables, constants, instance declarations and methods that have been defined with the PUBLIC or PROTECTED access identifiers. In the example, these are:

- the variables CV, MAX_Val and MIN_Val,
- the methods UP and DOWN

The UP method is now overridden so that it counts up in steps of 5. With the method definition the keyword OVERRIDE signals that the method for the base class is overridden. The overriding method and the method from the base class must be consistent on the following points:

- Access identifier (PUBLIC in the example)
- Method identifier (UP in the example)
- Data type of the return value (INT in the example)
- Parameters in the declaration subsection of the method (input parameter INC and output parameter QU in the example)

The statement section for the overriding method UP is entirely new. The UP method is called from the base class with the keyword SUPER and 5x of the input parameter IC of the overriding class is transferred to it as the input parameter. Similarly the return value and the output parameter QU of the base function are transferred to the overriding method. It may be confusing to see the same parameter identifiers in the input and output assignments on both sides. However, these are functions independent of the parameters. The formal parameters of the method of the base class are on the left side of the assignments, the formal parameters of the overriding class are on the right side as actual parameter.

Table 7-430 Up and Down counter in steps of 5

```
CLASS COUNTER_5STEP EXTENDS COUNTER
    METHOD PUBLIC OVERRIDE UP: INT // Method override
        VAR_INPUT
            INC : INT := 1;
        END_VAR
        VAR_OUTPUT
            QU : BOOL;
        END_VAR

        UP := SUPER.UP (INC := 5 * INC, QU => QU);

    END_METHOD
END_CLASS
```

The DOWN method does not need to be overridden. Since the COUNTER_5STEP classes is derived from the COUNTER class then it also inherits the DOWN method. The DOWN method calls the UP method with the following statement: `DOWN := THIS.UP (INC := - DEC);`. The keyword

THIS means that the method applicable there is called in derived classes. Consequently the DOWN method calls the following methods:

- in the COUNTER class:
  the UP method for counting upwards in steps of 1
- in the COUNTER_5STEP class:
  the UP method for counting upwards in steps of 5

This means that the DOWN method

- in the COUNTER class counts downwards in steps of 1
- in the COUNTER_5STEP class counts downwards in steps of 5.

Example: Call of the methods for both counters

Call in a program

The following program extends the Table 7-429 Example: Call of the methods in the COUNTER class (Page 4781) program to include the call of the methods in the COUNTER_5STEP class. A C2 instance of the COUNTER_5STEP classes is declared for this: The statements for calling the UP and DOWN methods of the C1 instance of the COUNTER classes are duplicated and modified accordingly.

This program must be assigned to a cyclic task of the execution system (e.g. BackgroundTask).

The complete ST source file also includes the classes COUNTER and COUNTER_5_STEP from Table 7-430 Up and Down counter in steps of 5 (Page 4784) and Table 7-428 Example: Up and Down counter as a class (Page 4777) respectively in front of the program CallCounter_ST2.

Table 7-431 Call of the methods of the COUNTER and COUNTER_5STEP classes in stages

```
PROGRAM CallCounter_ST2
    VAR
        C1 : COUNTER;       // COUNTER instance
        C2 : COUNTER_5STEP; // COUNTER_5STEP instance
        CountOut      : INT;
        CountOut2     : INT;
        Locking       : BOOL;
        Locking2      : BOOL;
        UpValreached  : BOOL;
        UpValreached2 : BOOL;
    END_VAR
```

```
    // Call of methods from COUNTER (C1)
    IF UpValreached = TRUE THEN
        Locking := TRUE;
    END_IF;
    IF Locking = FALSE THEN
        CountOut := C1.UP (QU => UpValreached); // Increment
    END_IF;
    IF Locking = TRUE THEN
        CountOut := C1.DOWN (); // Decrement
    END_IF;
    IF CountOut <= 0 THEN
        Locking      := FALSE;
        UpValreached := FALSE;
    END_IF;
    // Call of methods from COUNTER_5STEP (C2)
    IF UpValreached2 = TRUE THEN
        Locking2 := TRUE;
    END_IF;
    IF Locking2 = FALSE THEN
        CountOut2 := C2.UP (QU => UpValreached2); // Increment
    END_IF;
    IF Locking2 = TRUE THEN
        CountOut2 := C2.DOWN (); // Decrement
    END_IF;
    IF CountOut2 <= 0 THEN
        Locking2      := FALSE;
        UpValreached2 := FALSE;
    END_IF;
END_PROGRAM
```

Call via function

The following example moves the same types of statements which call the methods of both classes in the above example to the CallSingleCounter function. The class and additional variables are transferred to the function via in/out parameters. Program maintenance is essentially simplified because the individual statements for calling the UP and DOWN methods do not need to be maintained twice.

The program CallCounter_ST3 must be assigned to a cyclic task of the execution system (e.g. BackgroundTask).

The complete ST source file also includes the classes COUNTER and COUNTER_5_STEP from Table 7-430 Up and Down counter in steps of 5 (Page 4784) and Table 7-428 Example: Up and Down counter as a class (Page 4777) respectively in front of the function CallSingleCounter.

Table 7-432 Call of the methods of the COUNTER and COUNTER_5STEP classes via a function

```
FUNCTION CallSingleCounter : VOID
    VAR_IN_OUT
        C : COUNTER;
        CountOut     : INT;
        Locking      : BOOL;
        UpValreached : BOOL;
    END_VAR
    // Call COUNTER ( C ) for increment/decrement
    IF UpValreached = TRUE THEN
        Locking := TRUE;
    END_IF;
    IF Locking = FALSE THEN
        CountOut := C.UP (QU => UpValreached); // increment
    END_IF;
    IF Locking = TRUE THEN
        CountOut := C.DOWN (); // decrement
    END_IF;
    IF CountOut <= 0 THEN
        Locking := FALSE;
    END_IF;
END_FUNCTION

PROGRAM CallCounter_ST3
    VAR
        C1 : COUNTER;
        C2 : COUNTER_5STEP;
        CountOut      : INT;
        CountOut2     : INT;
        Locking       : BOOL;
        Locking2      : BOOL;
        UpValreached  : BOOL;
        UpValreached2 : BOOL;
    END_VAR
    // Call COUNTER (C1) for increment/decrement
    CallSingleCounter (
        C := C1,
        CountOut     := CountOut,
        Locking      := Locking,
        UpValReached := UpValReached);
    // Call COUNTER_5STEP (C2) for increment/decrement
    CallSingleCounter (
        C := C2,
        CountOut     := CountOut2,
        Locking      := Locking2,
        UpValReached := UpValReached2);
END_PROGRAM
```

Initialization of derived classes and their instances

Initialization of derived classes

As stated in the syntax diagram Figure 7-289 Syntax: Class (Page 4782) the initialization values of the following variables of the base class can be modified when deriving from a class:

- Public and protected variables (access identifiers PUBLIC or PROTECTED) without restrictions.
- Private variables (access identifier PRIVATE) only when the keyword OVERRIDE has been stated at their declaration block after the access identifier.

How to change the initialization values of the variables of a derived class:

1. Enter the assignment operator := after the name of the base class.
2. Next state the variables to be amended in brackets and their initialization values in the form of a structure initialization list (Page 4699) (similar process as for the initialization with the user-defined data type structure).
   If the base class is itself a derived class, you can modify the initialization values of the variables inherited from its base class in the following way (as an alternative):
   - Enter the base class itself as an element of the structure initialization list or
   - Aggregate the identifiers of the base class and the variables in the form *class_name.var_name*.

   See also section "Initialization of structures" in chapter Initialization of variables or data types (Page 4697) and the example below.

Example

The example below supplements the example Table 7-430 Up and Down counter in steps of 5 (Page 4784).

Table 7-433 Derivation and initialization of the COUNTER class

```
CLASS COUNTER_5STEP EXTENDS COUNTER
                := (MAX_Val := 500)
    // MAX_Val = 500, MIN_Val = 0
    // ...
    // Override of method UP
END_CLASS

CLASS COUNTER_10STEP EXTENDS COUNTER_5STEP
                := (COUNTER := (MIN_Val := -500))
             // Alternative:
             // := (COUNTER.MIN_Val := -500)
    // MAX_Val = 500, MIN_Val = -500
    // ...
    // Override of method UP
END_CLASS
```

Initialization of instances of derived classes

As stated in the syntax diagram Figure 7-288 Syntax: Instance declaration of a class Instance declaration of a classSyntax (Page 4779) the initialization values of the following variables can be modified with the instance declaration of a class:

- Public variables (access identifier PUBLIC) without restrictions.
- Protected and private variables (access identifier PROTECTED or PRIVATE) only when the keyword OVERRIDE has been stated at their declaration block after the access identifier.

How to change the initialization values of the variables of an instance:

1. Enter the assignment operator := after the name of the class.
2. Next state the variables to be amended in brackets and their initialization values in the form of a structure initialization list (Page 4699) (similar process as for the initialization with the user-defined data type structure).
   If the class is itself a derived class, you can modify the initialization values of the variables inherited from its base class in the following way (as an alternative):
   - Enter the base class itself as an element of the structure initialization list.
   - Aggregate the identifiers of the base class and the variables in the form *class_name*.*var_name*.

   See also section "Initialization of structures" in chapter Initialization of variables or data types (Page 4697) and the example below.

Example

The following example supplements the examples Table 7-429 Example: Call of the methods in the COUNTER class (Page 4781) and Table 7-431 Call of the methods of the COUNTER and COUNTER_5STEP classes in stages (Page 4785).

Table 7-434 Initialization of instances of the COUNTER class and the derivations

```
VAR
    C1 : COUNTER := (MIN_Val := -100);
         // MAX_Val = 100, MIN_Val = -100

    C2 : COUNTER_5STEP
            := (COUNTER := (MIN_Val := -100));
         // Alternative:
         // := (COUNTER.MIN_Val := -100);
         // MAX_Val = 500, MIN_Val = -100

    C3 : COUNTER_10STEP
            := (COUNTER_5STEP := (COUNTER := (MAX_Val := 1000)));
         // Alternative:
         // := (COUNTER_5STEP.COUNTER.MAX_Val := 1000);
         // MAX_Val = 1000, MIN_Val = -500
END_VAR
```

7.2.6.4 Abstract classes

Definition

A class is described as abstract if one of the two following conditions is met:

- The class is marked with the keyword ABSTRACT, see Class syntax diagram (Page 4782).
- At least one method of the class is abstract, i.e. is marked with the keyword ABSTRACT, see Method syntax diagram (Page 4770).

No instances can be declared of an abstract class.

In order to be able to declare instances, derived classes must be declared from abstract classes with all abstract methods overridden and their statement sections formulated. Derived classes in which not all inherited methods are implemented remain abstract.

Use

Abstract classes are a means of structuring and abstraction often used in object-oriented programming. The common variables and common methods are defined with their interfaces (formal parameters) in an abstract class. The methods and specified in the derived classes and overridden and implemented in accordance with the relevant requirements.

There are different motor types in a machine, e.g. motors that can be switched on directly via a contactor or via a star-delta switch and speed-controlled drives. The concrete methods for switching the motors on and off differ. Nevertheless the methods for switching on and off in an abstract class can be stipulated with their signatures (identifier, access identifier, formal parameters) with variables also defined for the status of the motor (e.g. running, stationary, starting, stopping, actual speed, error). These methods and variables should be subject to uniform usage in all classes. In the derived classes the methods are specified in accordance with the motor type with the corresponding values assigned to the variables.

7.2.6.5 Object-oriented interface

Defining an object-oriented interface

Object-oriented interfaces define the call interface for public methods (access identifier PUBLIC). Method prototypes are defined for this purpose in the object-oriented interface. These methods with the definition "prototype" are formulated in classes which implement the object-oriented interfaces.

Object-oriented interfaces are defined in the interface section (Page 4807) or implementation section (Page 4809) of an ST source file. In most cases, they are declared in the interface section because they are generally declared PUBLIC.

They must always be declared in full before they are used in the declaration of an implementing class or an interface variable. This applies irrespective of the compiler option (Page 4623) "Permit forward declarations".

Note

While object-oriented interfaces start and end with the same keywords as the interface section (Page 4807) of a unit (INTERFACE / END_INTERFACE), they have different meanings and are used in different ways.

Syntax of the object-oriented interface

Use the following syntax:

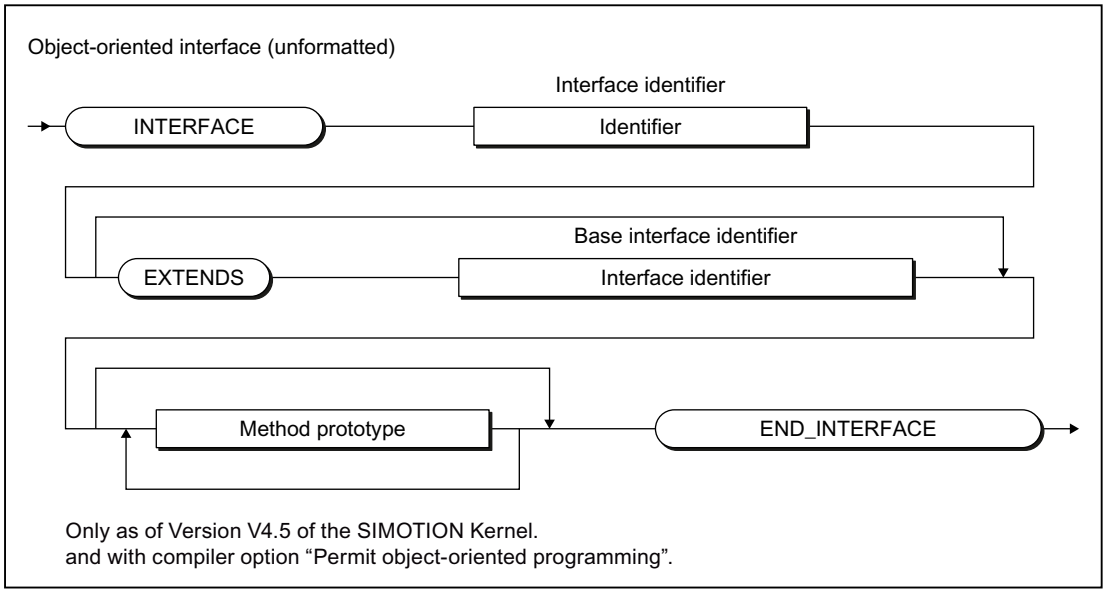


Figure 7-290 Syntax: Object-oriented interface

Enter an identifier as the interface name after the keyword INTERFACE.

This is then followed by:

- Optionally the keyword EXTENDS and the identifier of a different interface.
  This is how you derive an interface from an existing interface (base interface).
- The method prototypes in accordance with the following syntax
- The keyword END_INTERFACE

Method prototype

A method prototype determines the call interface for a method. This call interface includes:

- the method identifier (name),
- optionally the data type of the return value
- the input, in-out parameters and output parameters for the method

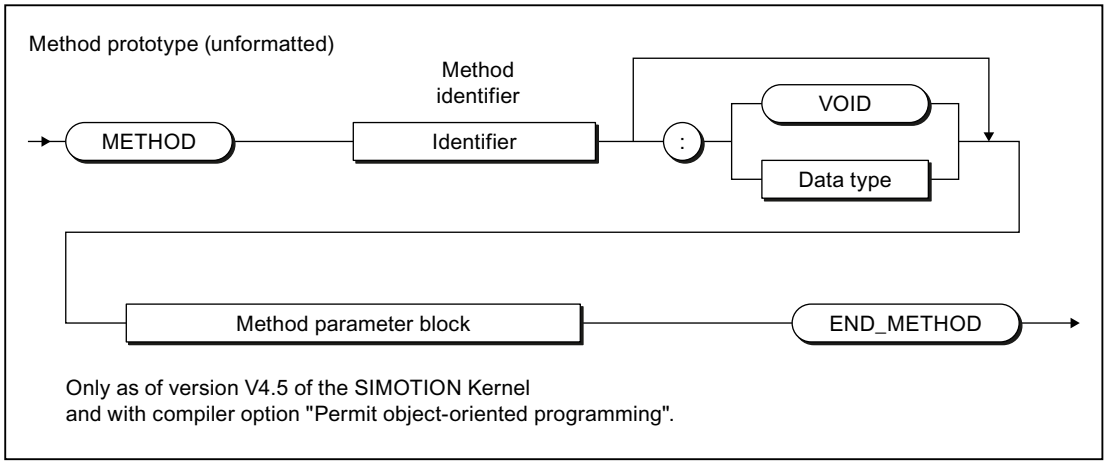


Figure 7-291 Syntax: Method prototype

These method prototypes must be formulated as public methods (access identifier PUBLIC) in the classes which implement the interface. Observe the following here:

- The following must still be adopted here:
  - the method identifier
  - the data type of the return value (if present)
  - the input, in/out and output parameters in the parameter block
- The following must or can be added:
  - Optional: Property of the method (FINAL, ABSTRACT)
  - Mandatory: Access identifier PUBLIC
  - Optional: Declaration of variables and data types
  - Mandatory: Statement section (except for methods with the property ABSTRACT: the statement section is omitted in this case).

Derivation of an object-oriented interface

With the keyword EXTENDS you can derive an object-oriented interface from an existing one. Only single derivations are possible, i.e. it is not possible to derive multiple interfaces.

Existing method prototypes cannot be changed or overridden in the derived object-oriented interface. Only method prototypes can be added.

Implementation of object-oriented interfaces in classes

Object-oriented interfaces are implemented in classes It is essential to declare the interfaces to be implemented before the implementing class in the ST source file.

The figure below repeats the syntax diagram for the class from the “Inheritance of classes" (Page 4781) section.

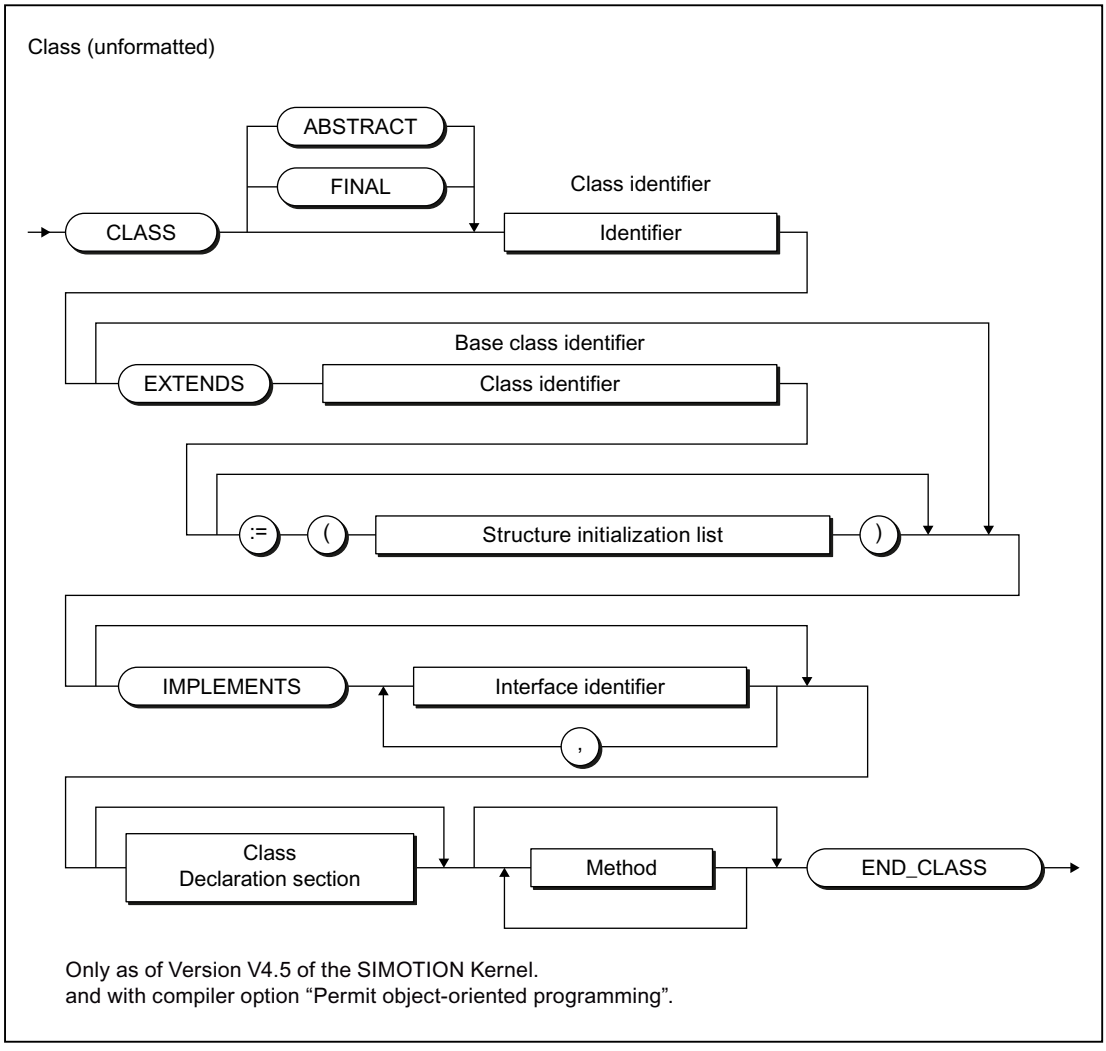


Figure 7-292 Syntax: Class

To implement one or more object-oriented interfaces use the keyword IMPLEMENTS, followed by the lists of the interfaces to be implemented separated by commas. A class can implement multiple interfaces.

The following must or can be supplemented for the definition of the implementing class:

- Optional: the class property (FINAL, ABSTRACT)
- Mandatory: the class identifier (name)
- Optional: the keyword EXTENDS with the identifier of a base class to be derived as well as the initialization (if applicable)
- Optional: the declaration subsection with the local variables and data type for the class

- Mandatory: all methods that have been defined as prototypes in the interfaces to be implemented.
  The following must still be adopted with this:
  - the method identifier
  - the data type of the return value (if present)
  - the input, in/out and output parameters in the parameter block

  The following must or can be added:
  - Optional: Property of the method (FINAL, ABSTRACT)
  - Mandatory: Access identifier PUBLIC
  - Optional: Declaration of local variables and data types
  - Mandatory: Statement section (except for methods with the property ABSTRACT: the statement section is omitted in this case).
- Optional: Additional methods (any access identifier possible).
  See: Creating methods (Page 4769).

All method identifiers must be unique within the class. This means that the object-oriented interface to be implemented

Derived classes and derived object-oriented interfaces

An object-oriented interface (base interface) is implemented in a class (base class). A further object-oriented interface containing additional method prototypes is derived from the base interface. This derived object-oriented interface is now implemented in another class that is also derived from the base class.

This derived class then inherits all the methods of the base class including those that have been defined as prototypes in the base interface. The additional method prototypes of the derived object-oriented interface are then the only elements still to be formulated in the derived class.

Variables of object-oriented interfaces

You can declare variables of object-oriented interfaces in program organization units and classes (e.g. in programs, function blocks, functions). Enter the name of the interface as its data type. Interface variables are initialized with "NULL". Only "NULL" is permissible for explicit specification of the initialization.

All variable types for the relevant program organization unit are permitted with the exception of:

- VAR_IN_OUT.
  Arrays, structures, function blocks or classes which contain interface variables may nonetheless be transferred as VAR_IN_OUT.
- VAR RETAIN or VAR_GLOBAL RETAIN is not meaningful.
  The interface variables are non-retentive when saved.
- VAR CONSTANT or VAR_GLOBAL CONSTANT is not meaningful.
  The interface variables must be initialized with "NULL".

Using the interface variables

Interface variables contain references to classes which implement the corresponding object-oriented interface. They permit access to the methods of the interface with the following syntax: *var_name.method-name*.

Once an instance of an implementing class has been assigned to the interface variable with the assignment operator :=, the method of the referencing class is called with the call in accordance with the syntax stated above.

An invalid reference is marked with the value "NULL" as the content of the variable. A check for a valid reference is easy therefore with the sequence `IF var_name <> NULL THEN (* … *) ; END_IF`, see example below.

Dynamic type conversion with the operator "?="

the operator ?= allows a dynamic type conversion for interface variables. It takes place during the program execution time.

A dynamic type conversion is performed if the following two conditions are satisfied:

1. The interface variable to the right of the operator contains the reference to the instance of a class.
2. This class implements the object-oriented interface that is the data type of the interface variable to the left of the operator.

In this case, the reference on the right-hand side is converted and transferred to the left-hand side.

If the type conversion fails, the variable on the left-hand side contains the "NULL" value.

This dynamic type conversion of references makes it easy to determine whether classes have implemented specific interfaces. The methods defined in the relevant interface can then be called. It is not necessary to know how the methods have actually been programmed.

Forcing instance-specific initialization in classes and function blocks

When instances of classes and function blocks are set up, their static variables can be initialized specific to the instance.

Instance-specific initialization can be forced for the variables of object-oriented interfaces. To do this, specify " `:= *` " after the identifier of the object-oriented interface when declaring these variables in classes or function blocks. The relevant static variable block must be assigned the OVERRIDE or PUBLIC identifier so that it can be initialized from an external source.

If a variable is declared in this way, the compiler checks whether all variables of this kind have been initialized at least with instances in VAR_GLOBAL or in the VAR declaration blocks of programs. In this case, initialization can take place even the instance was previously included in other classes or function blocks.

If a class or a function block possesses this kind of variable, the following needs to be noted when the instance is declared in the VAR blocks of other classes or function blocks:

- Either the instance itself is initialized,
- Or the variable block in which the instance is declared must be capable of initialization (OVERRIDE or PUBLIC identifier).

Initialization with NULL is permissible.

This feature can be used to define links between different instances via interfaces and to monitor by means of the compiler whether they have been meaningfully initialized.

Table 7-435 Example of forcing instance-specific initialization

```
INTERFACE IfTest
    // ...
END_INTERFACE
CLASS cl IMPLEMENTS IfTest
    // ...
END_CLASS
CLASS clTest
    VAR PROTECTED OVERRIDE
        myInterf : IfTest := *;
    END_VAR
END_CLASS
PROGRAM prog
    VAR
        clInst : cl;
        testInst : clTest := (myInterf := clInst);
    END_VAR
    ; // ...
END_PROGRAM
```

Example of interface variables

The example below illustrates the use of interface variables.

```
INTERFACE ITF1
    METHOD m1 END_METHOD
END_INTERFACE
INTERFACE ITF2
    METHOD m2 END_METHOD
END_INTERFACE
CLASS A IMPLEMENTS ITF1
    METHOD PUBLIC m1 (* ... *) ; END_METHOD
    // ...
END_CLASS
CLASS B IMPLEMENTS ITF2
    METHOD PUBLIC m2 (* ... *) ; END_METHOD
    // ...
END_CLASS
CLASS C IMPLEMENTS ITF1, ITF2
    METHOD PUBLIC m1 (* ... *) ; END_METHOD
    METHOD PUBLIC m2 (* ... *) ; END_METHOD
    // ...
END_CLASS
```

```
FUNCTION func_if1 : VOID
    VAR_INPUT i : ITF1; END_VAR
    IF i <> NULL THEN
        i.m1();
    END_IF;
END_FUNCTION
FUNCTION func_if2 : VOID
    VAR_INPUT i    : ITF2; END_VAR
    VAR       tmp  : ITF1; END_VAR
    IF i <> NULL THEN
        i.m2();
    END_IF;
    tmp ?= i;
    (* Dynamic type conversion to ITF1 *)
    IF tmp <> NULL THEN
        tmp.m1();
    END_IF;
END_FUNCTION
PROGRAM P
    VAR
        inst_a : A; // Instance of class A
        inst_b : B; // Instance of class B
        inst_c : C; // Instance of class C
        interf1: ITF1; // interf1 has value NULL
        interf2: ITF2; // interf2 has value NULL
    END_VAR

    interf1 := inst_a;
    (* interf1 contains a valid reference to inst_a. *)
    func_if1(interf1);
    (* The method inst_a.m1() is therefore called within the function. *)

    func_if1(inst_a);
   (* Equivalent call to the last two code lines *)

    interf2 := inst_b;
    (* interf2 contains a valid reference to inst_b. *)
    func_if2(interf2);
    (* The method inst_b.m2() is therefore called within the function.
       The method m1() is not called. The variable tmp contains the value NULL after
execution of the operator ?= because the class B does not implement the interface ITF1. *)

    interf2 := inst_c;
    (* interf2 contains a valid reference to inst_c. *)
    func_if2(interf2);
    (* The method inst_b.m2() is therefore called within the function.
      The method inst_c.m1() is also called. The variable tmp contains a valid reference to
inst_c after execution of the operator ?= because the class C also implements the interface
ITF1. *)
END_PROGRAM
```

Two object-oriented interfaces (ITF1 and ITF2) are defined in the example shown. Each interface contains one method as relevant: m1() in ITF1 and m2() in ITF2.

The following 3 classes implement both of these interfaces:

- Class A implements interface ITF1
- Class B implements interface ITF2
- Class C implements both interfaces ITF1 and ITF2.

For both functions func_if1 and func_if2:

- You have the option of adopting an interface variable via an input variable i with the relevant interface type.
- You can use the query " i<> NULL" to check whether i contains a valid reference.
  If the reference is valid, the functions call the methods of the relevant interfaces.
- Function func_if2 has the additional option of transferring (tmp ?= i) the reference of i to the tmp variable of type ITF1.
  - If i contains a reference that matches interface ITF1, the reference is transferred by the ?= operator.
  - If the reference does not match, a NULL is entered in tmp.

  If the reference in tmp is valid, method m1() is called.

The instances of classes A, B and C are generated and the interface variables interf1 and interf2 are declared in program P.

In the program statement section, the class instances are assigned to the relevant interface variables interf1 and interf2. Following each of these assignments one of the two functions is called with the corresponding interface variable also transferred. This results in different calls for the interface methods within the functions:

- First call of func_if1 (with transfer of the interface variables interf1):
  Since interf1 contains a valid reference to the instance of class A (inst_a); the function calls the method m1().
- Second call of func_if1 (with the transfer of instance inst_a):
  Since class A implements the interface ITF1, this call is functionally identical to the first call.
- First call of func_if2 (with transfer of the interface variables interf2):
  Since interf2 contains a valid reference to the instance of class B (inst_b); the function calls the method m2().
  The method m1() is not called, since the class B only implements the interface ITF2.
- Second call of func_if2 (with transfer of the interface variables interf2):
  Since interf2 contains a valid reference to the instance of class C (inst_c); the function calls both methods m1() and m2().
  Class C implements both interfaces (ITF1 and ITF2). The operator ?= therefore transfers the reference to inst_c successfully to the variable tmp of data type ITF1. The function therefore also calls the method m1().

This example demonstrates perfectly how interface variables with corresponding check functions can be handled and the degree of programming flexibility offered by this approach.

Note

This mechanism of course works in exactly the same way in methods, But we have used functions here instead of methods because the example was simpler.

This option of forming object references and interface variables makes it possible to independently develop and test program modules. Interfaces are a description of the interfaces shared by the program modules containing the program code of the methods to be implemented. This data description can then be used, for example, to perform a test in a program area with dummy methods and data supplied by reference transfer. When the formulated program sections are created later on, the tested sections will function automatically. Of course, this arrangement will work successfully only if the interfaces are not changed retrospectively. But the compiler monitors interfaces carefully for any changes.

7.2.6.6 Comparison between abstract class and object-oriented interface

If we compare the definition of an abstract class (Page 4790) and compare it to the definition of an object-oriented interface (Page 4790), we will find that both constructs are very similar. Both constructs contain prototype methods for which the appropriate functions must be programmed in a class.

The programmer must now decide which construct to use for the specific task in hand. The following information should provide a useful decision-making guide.

The table below compares the properties of the abstract class with the properties of the object-oriented interface.

Table 7-436 Comparison between abstract class and object-oriented interface

Abstract class	Object-oriented interface
Abstract classes can contain abstract methods as well as fully programmed methods, i.e. real methods.	An interface may contain only prototype methods (keyword ABSTRACT does not need to be specified).
Abstract classes can possess defined properties (attributes).	An interface cannot possess any properties (attributes).
The properties of methods can be selected (PUBLIC, PRIVATE, PROTECTED).	All methods in the interface are PUBLIC and cannot be changed retrospectively.
An abstract class cannot be instantiated.	Interface variables can be created.
The methods of an abstract class must be fully programmed when they are derived unless the derivation itself is also ABSTRACT.	All the methods of an interface must be programmed in the class in which the method is implemented, or identified as ABSTRACT
A class can be derived from an (abstract) class.	An interface can be derived from a base interface.
An (abstract) class can implement multiple interfaces.	-

Both constructs define methods as prototypes. This guarantees that the methods can be used according to their definition in the derived or implementing classes. A programmer can thus feel confident that the use (call) of methods defined as prototypes will function reliably when the entire program is finished and the individual program sections are joined together.

The basic difference between the two constructs can be summarized as follows:

- An abstract class constitutes a contract for use in the derivation chain, i.e. within the class hierarchy.
- Object-oriented interfaces constitute an external contract between different program sections.

The following decision-making guide can be derived from the above information:

- Object-oriented interfaces are used if the following two conditions are satisfied:
  - A more generalized definition between different programs is required.
  - The methods should or may be accessed by anyone.
- Abstract classes are used if one of the following conditions is satisfied:
  - A structure must be defined for a program area..
  - Methods must be concealed (i.e. must not be PUBLIC).

Abstract classes can be combined very effectively with object-oriented interfaces in object-oriented software and thus provide a wonderful tool for planning and designing the software structure.

The benefits of abstract classes are:

- Abstract classes allow full programming of methods. thereby reducing the time and effort involved in writing program sections that can be used by everyone in the same way. The methods are simply inherited when real classes are derived.
- The specification of access rights to methods enhances the security of programs.

The benefits of object-oriented interfaces are:

- Object-oriented interfaces allow different program sections to be connected in a highly flexible manner,
- so making it possible for software development teams to work more independently of one another.

7.2.6.7 General references

Defining general references

General references contain the address assignment to a variable or an instance of a function block or class. To do this, the REF_TO keyword creates a data type that can contain the address for an existing data type (reference data type).

- In a data type declaration:
```
TYPE
    ref_type : REF_TO data_type
END_TYPE
```
The declaration is also possible in an ARRAY or as element of a structure.
- In a variable declaration:
A reference already declared as data type or the REF_TO statement can be used directly.
```
VAR
    ref_var : ref_type // or
    ref_var := REF_TO data_type
END_VAR
```
The declaration is also possible in an ARRAY.

The associated declared data type is "Reference to the data type" (REF_TO *data_type*).

The following are permitted as data types to which references can be formed:

- Elementary data types (e.g. INT, DINT, REAL, WORD, TIME, STRING)
- User-defined data types (UDT)
- System data types
- Function blocks, provided they contain at least one static variable
- Classes

No references can be formed to the following data types because references exist already:

- Technology object data types
- Object-oriented interfaces
- General references
- I/O references

Variables of general references can be declared in the interface and implementation sections as well as in all program organization units and classes. The data type to be referenced must be declared beforehand and lie within the scope of the POU. The declaration as element of a structure or an ARRAY is also possible.

All variable types for the relevant program organization unit are permitted with the exception of:

- VAR_IN_OUT.
  Arrays, structures, function blocks or classes that contain general references may nonetheless be transferred as VAR_IN_OUT.
- VAR RETAIN or VAR_GLOBAL RETAIN is not permitted.
  However, structures that contain at least one element other than reference types (general references, data types of the technology objects, object-oriented interfaces), can be stored retentive.
- VAR CONSTANT or VAR_GLOBAL CONSTANT is not meaningful.
  General references must be initialized with "NULL".

Forming general references

A reference to a variable is formed with the REF standard function (in := *var_name*). This function is described in detail in the "SIMOTION Basic Functions" Function Manual.

The REF function can be used on the following variables:

- Retentive and non-retentive unit variables or instances of classes or function blocks or their elements declared as such.
- In methods within classes or function blocks:
  - Static variables (VAR) of the higher-level class or function block (including instance variables for classes or function blocks).
  - The THIS keyword. Enables access to the instance variable of the higher-level class or function block.
- In function blocks:
  - Static variables (VAR) of the function block (including instance variables of classes or function blocks).
  - The THIS keyword. Permits access to the instance variable of the function block.

The variable must be able to be read and written.

Not permitted are:

- Variables of technology object data types
- Variables of object-oriented interfaces
- Variables of general references
- Retentive local variables within classes or function blocks
- I/O variables
- Global device variables
- Constants

The return value has the data type "Reference to the data type of the input parameter *in*". The return value can be assigned with the assignment operator ":=" to a variable that has REF_TO *type* data type. A data type conversion is not possible.

Note

Note for instances of function blocks that are passed as in/out parameter (VAR_IN_OUT):

- The REF function cannot be used for static variables (VAR) of function blocks imported from technology packages or device-independent libraries.

The restriction does not apply to class instances.

Operations with general references

Assignment operator ":="

For the assignment of a reference variable, the address of the variable rather than the variable value is passed. Consequently, the reference data types must be identical on both sides of the operator. No implicit data type conversion is possible.

The only exception is for references to classes. A reference to a derived class of a reference to the base class can be assigned here.

For the assignment, no check is made for the validity of the reference; the assignment is always possible.

An explicit assignment with NULL is also possible.

The validity of references should be checked by comparing with NULL before use.

Dereferencing with operator "^"

The reference content can be accessed with the postfix operator "^", e.g. *ref_var*^.

If the reference does not contain a valid value at runtime, the processing task is aborted and the ExecutionFaultTask called.

Before access, a check for validity by comparison with NULL is recommended.

Multilevel dereferencing is possible, e.g. reference to a structure that contains further references. Dereferenced variables can be used anywhere in expressions.

If a method is called via a reference to a class or function block, the class or function block must be dereferenced.

Dereferenced values cannot be used as actual parameters for the _releaseSemaphore() and _testandSetSemaphore() system functions.

Dynamic type conversion with the operator "?="

The operator "?=" can be used for dynamic type conversion only with references to classes and with interface variables (variables for object-oriented interfaces). It supports the following variants:

- Assignment between references to classes.
- Assignment of a reference to a class for an interface variable.
  Whereby, the reference to a class must be specified with the dereferencing operator "^".
- Assignment of an interface variable for a reference to a class.
- Assignment between interface variables

A dynamic type conversion is performed if the following two conditions are satisfied:

1. The variable to the right of the operator contains the reference to the instance of a class.
2. For the variable to the left of the operator, the following applies:
   - The variable is an interface variable:
     The object-oriented interface that this variable has as data type must implement the class on the right-hand side.
   - The variable is a reference to a class:
     This class corresponds to the class on the right-hand side or is a base class for this class.

In this case, the reference on the right-hand side is converted and transferred to the left-hand side.

If the type conversion fails, the variable on the left-hand side contains the "NULL" value.

Comparison operators

Only the comparison operators "=" and "<>" can be used to check for equality or inequality.

The operators ">", "<", ">=", "<=" as well as the MIN and MAX standard functions are not permitted.

7.2.6.8 I/O references (as of kernel V5.1)

Forming I/O references

Declaration of I/O references

I/O references allow symbolic access to I/O variables within classes and function blocks. This allows the classes and function blocks also to be used in libraries.

The I/O references are declared exclusively in classes and function blocks in the declaration block for static variables (VAR). The declaration as retentive variable (VAR RETAIN) or constant VAR CONSTANT is not possible.

var_name AT %I : data_type* ; *ll* for inputs
var_name AT %Q* : *data_type ; ll* for outputs

The following are permissible as data types:

- Bit data types (e.g. BOOL, WORD)
- Integer data types (e.g. SINT, UINT)
- Arrays of these data types, other than arrays of the BOOL data type

Provided the variable block is not identified as PUBLIC, it must be identified with the OVERRIDE keyword so that the variables can be initialized instance-specific for the instance formation of the class or function block.

Example

```
FUNCTION_BLOCK myFB
  VAR OVERRIDE
    myInput AT %I*  : INT; // Declaration input
    myOutput AT %Q* : INT; // Declaration output
    myOutArr AT %Q* : ARRAY[0..3] OF BYTE;
                           // Array of outputs
  END_VAR
  ; // ...
END_FUNCTION_BLOCK
```

Initialization for the definition

I/O references are initialized with NULL. Only "NULL" is permissible for explicit specification of the initialization. With the specification of NULL as initialization value, for the instance declaration of the class or of the function block, the I/O reference must not necessarily be linked with an I/O variable from the address list. This allows the software to be structured so that, if necessary, it adapts itself to non-linked I/O references.

An I/O reference identified with NULL in the initialization must be checked for validity before use, as also usual for all other reference types. A comparison with NULL can be used for this purpose.

```
FUNCTION_BLOCK myFB
  VAR OVERRIDE
    myInput AT %I* : INT := NULL;
  END_VAR
  IF (myInput <> NULL) THEN
    // Validity check of the I/O reference
    Testvar := myInput;
    // Access to the value
  END_IF;
  // ...
END_FUNCTION_BLOCK
```

Principle

The following restrictions apply to I/O reference variables:

- I/O references cannot be forwarded as in/out parameters (VAR_INOUT) to other program organization units (POU).
- I/O references cannot be assigned general references and vice versa. In particular, the REF() standard function cannot be used on I/O variables to assign I/O references.
- I/O references cannot be passed to input parameters (VAR_INPUT) defined as variable-length ARRAY.
- I/O references are not displayed in the Symbol browser and in watch tables.
- I/O references cannot be read or written per HMI, Web access or OPC access.
- I/O references must not be declared in retentive variables (VAR RETAIN).
- Bit access is possible to I/O references of the BYTE, WORD or DWORD data type. This also applies to the associated standard functions (e.g. _setbit()).
- Use as argument for marshalling and endian functions is not possible.

Linking I/O references to I/O variables

For the instance declaration of the higher-level class or function block, I/O references are linked with an I/O variable. This is done with the instance-specific initialization of the class or function block. Whereby, the specified I/O reference is assigned an I/O variable from the address list rather than a constant.

The following apply:

- I/O references for inputs (%I*) can be linked with any I/O variable of an appropriate data type.
- I/O references for outputs (%Q*) can be linked only with an I/O variable for an output of an appropriate data type. This variable may not be identified with the attribute "read only".

For partial initialization, as usual for structures, the initialization value of the associated data type is used for non-explicitly specified components.

I/O references declared as ARRAY must be linked completely with an appropriate I/O ARRAY variable of the address list. Individual variables as I/O reference can also be linked with the individual elements of an I/O ARRAY variable by specifying a constant index value.

```
VAR_GLOBAL
  myFbInst1 : myFB := (myInput  := InputVarW3,
                       myOutput := OutputVarW10);
  myFbInst2 : myFB := (myInput  := InputVarW4,
                       myOutput := OutputVarW14);
END_VAR
```

The compiler checks whether all I/O references are linked with I/O variables at least with instances in VAR_GLOBAL or in the VAR declaration blocks of programs. Linking is also possible even such an instance was included previously in other classes or function blocks.

If a class or a function block possesses an I/O reference, the following must be observed when the instance is declared in the VAR blocks of other classes or function blocks:

- Either the instance itself is initialized,
- Or the variable block in which the instance is declared must be capable of initialization (OVERRIDE or PUBLIC identifier).

The above-mentioned checks are omitted when the I/O reference for the declaration was initialized with NULL.

7.2.7 Integration of ST in SIMOTION

This section describes the interoperability of ST programs and SIMOTION SCOUT.

7.2.7.1 Source file sections

An overview of the meaning of the source file sections has been provided in Structure of an ST source file (Page 4668). There you will find detailed information about, for example:

- the syntax of source file sections (Page 4807)
- how you can declare public and use (Page 4827) data between different ST source files.

Use of the source file sections

You must follow certain structure and syntax rules in your source file sections so that the ST source file can be compiled. A few general guidelines are presented here; details on source file sections are presented later in this section:

- When creating the source file, you should always pay attention to the order of the source file sections. A section that is to be called must always precede the calling section so that the former is recognized by the latter.
  For example, variables must always be declared before they are used and functions must be defined before they are called.
- The source text for the most common source file sections – program, function or function block – consists of the following:
  - Start of section with reserved word and identifier
  - Declaration section (optional)
  - Statement section
  - End of section with reserved word
- Identifiers for source file sections – hereinafter referred to as *name* or *name_list* - follow the general syntax rules for identifiers, see Identifiers in ST (Page 4656).

Note

A template with all possible source file sections (template for example unit) is available in the online help.

Interface section

The interface section contains statements for declaring data public and using data (data types, variables, function blocks, functions, programs). Technology packages and libraries can also be downloaded.

The interface section has the following syntax:

Syntax

```
INTERFACE
// Interface statements (optional)
END_INTERFACE
```

An individual identifier of the section cannot be specified.

Optionally, interface statements exist in the following order between reserved words INTERFACE and END_INTERFACE.

1. Specification of utilized technology package. Syntax:
   `USEPACKAGE tp-name [AS namespace];`
   For more details, refer to the *SIMOTION Basic Functions* Function Manual.
2. Specification of utilized libraries.
   Syntax:
   `USELIB library-name-list [AS namespace];`
   For more information, see "Using data types, functions and function blocks from libraries (Page 4896)".
3. Reference to other units in order to use their public components.
   The syntax is:
   `USES unit_name-list;`
   For more information, see "USES statement in a using unit (Page 4830)".
4. Declarations and information regarding declaration as public to other program sources:
   - Data type definitions (Page 4823):
     User-defined data types (UDT) that are public and valid within the entire ST source file
   - Variable declarations (Page 4824):
     Unit variables and unit constants that are public and valid within the entire ST source file.
     Permissible keywords: See Section "Variable declaration" (Page 4824).
   - Object-oriented interfaces (Page 4820):
     Object-oriented interfaces that are public and valid within the entire ST source file.
     As of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) “Permit object-oriented programming” enabled.

     Note

     Object-oriented interfaces utilize the same keywords as the interface section (INTERFACE / END_INTERFACE).

   - Specification of the program organization units (POUs) to be declared public.
     Syntax:
     `FUNCTION fc_name;`
     `FUNCTION_BLOCK fb_name;`
     `PROGRAM program_name;`
     `CLASS class_name;` (as of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) "Permit object-oriented programming” enabled)
     If the "Permit forward declarations" compiler option (Page 4623) is activated, they are also interpreted as POU prototypes for the forward declaration (Page 4929).
   - POU prototypes for the forward declaration (Page 4929).
     Specification of the prototypes for program organization units with forward declaration (only effective if the "Permit forward declarations" compiler option (Page 4623) is activated).
     They are also interpreted as POUs to be exported.

All technology packages, libraries, units used, data type declarations, variable declarations, object-oriented interfaces and program organization units listed in the interface section are declared public. For further information about declaring public, see "Interface section of a unit with Declare Public function (Page 4828)".

Sequence

The interface section is the first section of an ST source file.

Note

Optionally, the unit statement can precede the interface section, see "Identifier of the unit (Page 4827)".

The order of the interface statements 1 to 4 is fixed.

Within number 4, any order is permitted. The individual declaration blocks for data type definitions (including POU prototypes) and variables declarations can appear more than once.

Please note: Identifiers must be declared before they are used.

Only if the "Permit forward declarations" compiler option (Page 4623) is activated: When declaring an instance of a function block or a class, it is sufficient for the prototype of the function block to have been declared beforehand. It is not possible to initialize the function block or the class.

Frequency

Once per ST source file

Mandatory section

Yes

Implementation section

The implementation section contains the executable sections, comprising the main part of the ST source file.

The implementation section has the following syntax:

Syntax

```
IMPLEMENTATION
// Implementation statements (optional)
END_IMPLEMENTATION
```

An individual identifier of the section cannot be specified.

Optionally, implementation statements (main part of the ST source file) exist in the following order between the reserved words IMPLEMENTATION and END_IMPLEMENTATION:

1. Reference to other units in order to use their public components. Syntax:
   `USES unit_name-list;`
   For more information, see "USES statement in a using unit (Page 4830)".
2. Declarations
   - Data type definitions (Page 4823):
     User-defined data types (UDT) that are valid within the entire ST source file
   - Variable declarations (Page 4824):
     Unit variables and unit constants that are valid within the entire ST source file.
     Permissible keywords: See Section "Variable declaration" (Page 4824).
   - Object-oriented interfaces (Page 4820):
     Object-oriented interfaces that are valid within the entire ST source file
     As of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) "Permit object-oriented programming" activated.
   - POU prototypes for the forward declaration (Page 4929).
     Specification of the prototypes for program organization units with forward declaration (only effective if the "Permit forward declarations" compiler option (Page 4623) is activated).

   Note

   Data types, variables or object-oriented interfaces that are to be declared public and used in other program sources must be fully declared in the interface section (Page 4807).
3. Program organization units (Page 4811).

Sequence

Always follows the interface section (Page 4807).

If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:

- The order of the implementation statements indicated above is mandatory; within numbers 2 and 3, any order is permitted.
- All identifiers must be declared before they are used.

If the "Permit forward declarations" compiler option (Page 4623) is activated, the following applies:

- As regards the order of the interface statements, the following applies:
  Numbers 2 and 3 need no longer be strictly separated. Declarations may be freely programmed between program organization units.
  Exception: POU prototypes must be declared before the relevant program organization unit is defined or used.
- As long as the following conditions are satisfied, the identifiers of program organization units (Page 4811) can be used before they are defined.
  - There are no restrictions with respect to functions (Page 4812), programs (Page 4815), expressions (Page 4816) or methods (Page 4819). Any order is permitted.
    A POU prototype may be optionally declared in advance for programs or functions.
  - This is possible for function blocks (Page 4813) or classes (Page 4817) subject to the following restrictions (e.g. when the instance is declared):
    Any order is permitted if initialization is not specified. In this case, however, it is absolutely essential that the prototype of the function block or class is declared beforehand.
    If initialization is specified, the function block or the class must be fully defined beforehand.
- All other identifiers (e.g. data types, variables) must be declared before they are used.

Frequency

Once per ST source file

Mandatory section

Yes

Program organization units (POUs)

Program organization units (POUs) are the executable source file sections:

- Functions (FC) (Page 4812)
- Function blocks (FB) (Page 4813)
- Programs (Page 4815)
- Expressions (Page 4816)
- Classes (Page 4817) (as of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) "Permit object-oriented programming" activated)

The following applies for the sequence of the POUs in an ST source file:

- Usually, the called POUs must precede the calling POUs in the source file so that the former are recognized by the latter.
- **Exception**: Only if the "Permit forward declarations" compiler option (Page 4623) is activated.
  The sequence is arbitrary: The following applies to the specification of a prototype for forward declaration (Page 4929) in the interface section (Page 4807) or the implementation section (Page 4809):
  - Essential for declaring instances with function blocks and classes
  - Optional when calling functions and programs

Functions (FCs)

Functions (FC) are classified as program organization units (POUs). Functions are parameterized source file sections with temporary data that can be called from programs and function blocks. All internal variables lose their values when the function is exited and are reinitialized the next time the function is called.

FCs have the following syntax:

Syntax

```
FUNCTION name : function_data_type
// Declaration section
// Statement section
END_FUNCTION
```

name stands for the identifier of the function, while *function_data_type* stands for the data type of the return value.

Permissible keywords for the variable declaration in the declaration section (Page 4821): See Section "Variable declaration" (Page 4824).

Note the following for functions with *function_data_type* <> VOID: In the body (Page 4822), an expression of data type *function_data_type* must be assigned to the function identifier.

Sequence

FCs can only be defined in the implementation section (Page 4809).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  In the source file, functions must precede the POU from which they are called.
- If the "Permit forward declarations" compiler option (Page 4623) is activated, the following applies:
  The sequence is arbitrary: There is an option to specify a prototype for the forward declaration (Page 4929) in the interface section (Page 4807) or the implementation section (Page 4809).

The declaration section (Page 4821) must precede the statement section (Page 4822).

Frequency

Any number of times per ST source file

Mandatory section

No

Further information on functions (FC)

See Section "Creating and calling functions and function blocks" (Page 4736).

Function blocks (FBs)

Function blocks (FB) are classified as program organization units (POUs). They are source file sections with static data that can be called from programs and assigned parameters (internal variables retain their value between calls). Since an FB has memory, its output parameters can be accessed at any time and from any point in the user program.

FBs have the following syntax:

Syntax

```
FUNCTION_BLOCK name
// Declaration section
    // Methods (Only with compiler option (Page 4623) "Permit object-oriented programming")
// Statement section
END_FUNCTION_BLOCK
```

name stands for the identifier of the function block.

Permissible keywords for the declaration of variables in the declaration section: See Section "Variable declaration" (Page 4824).

Special features

Before you call a function block (FB), you must declare an instance: You declare a variable and enter the identifier of the function block as the data type. You can declare the instance locally (within VAR / END_VAR in the declaration sections of a program or a function block).

You can also declare the instance globally (within VAR_GLOBAL/END_VAR in the interface or implementation section).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  This is only possible with FBs which make used program sources and libraries available, but not with FBs which are defined in the same ST source file.
- If the "Permit forward declarations" compiler option (Page 4623) is activated, the following applies:
  You can declare global instances of FBs which are defined in the same ST source file.
  - For instance declaration without specification of initialization:
    The instance may be declared before the FB is fully defined. In this case, it is essential that a prototype for the forward declaration (Page 4929) is specified in the interface section (Page 4807) or the implementation section (Page 4809).
  - For instance declaration with specification of initialization:
    The instance must always be declared after the FB is fully defined.

You cannot declare an instance of an FB in functions or methods.

Sequence

FBs can only be defined in the implementation section (Page 4809).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  In the source file, an FB must precede the POU in which it is used, e.g. in which an instance of the FB is declared as local variable.
- If the "Permit forward declarations" compiler option (Page 4623) is enabled, the following applies:
  - Without specification of initialization:
    The POU which uses the FB may precede the definition of the FB. In this case, it is essential that a prototype for the forward declaration (Page 4929) is specified in the interface section (Page 4807) or the implementation section (Page 4809).
  - With specification of initialization (only with compiler option (Page 4623) "Permit object-oriented programming"):
    The FB must be fully defined before the POU in which it is used.

The declaration section (Page 4821) must precede the statement section (Page 4822).

Frequency

Any number of times per ST source file

Mandatory section

No

Further information about function blocks (FB)

See Section "Creating and calling functions and function blocks" (Page 4736).

Programs

Programs are classified as program organization units (POUs). They are called on the target system according to their task assignment (see *Configuring the execution system* in the *SIMOTION Basic Functions* Function Manual) and can call functions (FC) and function blocks (FB), as well as methods in classes and function blocks.

Programs have the following syntax:

Syntax

```
PROGRAM name
// Declaration section
// Statement section
END_PROGRAM
```

name stands for the name of the program.

Permissible keywords for the variable declaration in the declaration section: See Section "Variable declaration" (Page 4824).

Sequence

Programs can only be defined in the implementation section (Page 4809).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  Programs are expediently placed after FCs, FBs, expressions and classes. The program then recognizes these source file sections and can make use of them.
- If the "Permit forward declarations" compiler option (Page 4623) is enabled, the following applies:
  The sequence is arbitrary: When a program is called within a program, there is an option to specify a prototype for the forward declaration (Page 4929) in the interface section (Page 4807) or the implementation section (Page 4809).

The declaration section (Page 4821) must precede the statement section (Page 4822).

Frequency

Any number of times per ST source file

Mandatory section

No

Further information on programs

See Section "Programs" (Page 4762).

Expressions

Expressions are a special case of a function declaration with the specified data type BOOL of the return value. The expression within the EXPRESSION <expression identifier> ... END_EXPRESSION reserved words assigned to the function name is evaluated.

You can use the WAITFORCONDITION (Page 4729) construct to wait directly for a programmable event or condition in a MotionTask. The statement suspends the task that called it until the condition (expression) is true.

Expressions have the following syntax:

Syntax

```
EXPRESSION name
// Declaration section
// Statement section
END_EXPRESSION
```

name stands for the identifier of the expression.

Permissible keywords for the variable declaration in the declaration section: See Section "Variable declaration" (Page 4824).

Please note: In the body (Page 4822), an expression of data type BOOL must be assigned to the expression identifier.

Sequence

An expression can only be declared in the implementation section (Page 4809) of an ST source file.

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  Expressions must precede the program organization unit in which they are called from a WAITFORCONDITION control structure.
- If the "Permit forward declarations" compiler option (Page 4623) is activated, the following applies:
  The sequence is arbitrary: There is no option to specify a prototype for the forward declaration (Page 4929) in the interface section (Page 4807) or the implementation section (Page 4809).

The declaration section (Page 4821) must precede the statement section (Page 4822).

Frequency

Any number of times per ST source file

Mandatory section

No

Further information on expressions

See Section "Expressions" (Page 4763).

Examples in conjunction with the WAITFORCONDITION statement: See SIMOTION Basic Functions Function Manual

Classes (as of Kernel V4.5)

Classes belong to the program organization units (POE). They form a capsule for variables and methods (Page 4819). Only variables and methods that have the access identifier PUBLIC are visible outside of the class. Another class can be derived from a class (base class). This derived class inherits all the variables and methods from the base class.

A SIMOTION Kernel as of version V4.5 is mandatory for the purposes of defining and using classes. Classes can only be defined when compiler option (Page 4623) "Permit object-oriented programming" is enabled.

Classes have the following syntax:

Syntax (simplified)

```
CLASS name
// Declaration section
    // methods
END_CLASS
```

name stands for the identifier of the class.

Permissible keywords for the variable declaration in the declaration section: See Section "Variable declaration" (Page 4824).

For a complete syntax diagram see Figure 7-289 Syntax: Class (Page 4782).

Special features

Before using public methods or variables of a class you must declare an instance of the class: You declare a variable and enter the identifier of the class as the data type. You can declare the instance locally (within VAR / END_VAR in the declaration sections of a program, a function block or a class).

You can also declare the instance globally (within VAR_GLOBAL/END_VAR in the interface or implementation section).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  This is only possible with classes which make used program sources and libraries available, but not with classes which are defined in the same ST source file.
- If the "Permit forward declarations" compiler option (Page 4623) is activated, the following applies:
  You can declare global instances of classes which are defined in the same ST source file.
  - For instance declaration without specification of initialization:
    The instance may be declared before the class is fully defined. In this case, it is essential that a prototype for the forward declaration (Page 4929) is specified in the interface section (Page 4807) or the implementation section (Page 4809).
  - For instance declaration with specification of initialization:
    The instance must always be declared after the class is fully defined.

You cannot declare an instance of a class in functions or methods.

Sequence

Classes can only be defined in the implementation section (Page 4809).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  In the source file, a class must precede the POU in which it is used, e.g. in which an instance of the class is declared as a local variable.
- If the "Permit forward declarations" compiler option (Page 4623) is enabled, the following applies:
  - Without specification of initialization:
    The POU which uses the class may precede the definition of the class. In this case, it is essential that a prototype for the forward declaration (Page 4781) is specified in the interface section (Page 4807) or the implementation section (Page 4809).
  - With specification of initialization:
    The class must be fully defined before the POU in which it is used.

The declaration section (Page 4821) must precede the methods (Page 4819).

Frequency

Any number of times per ST source file

Mandatory section

No

Additional information on classes

See “Object-Oriented Programming - OOP (as of kernel V4.5)” (Page 4766) section

Methods

Methods belong to the program organization units (POE). They are callable and parameterizable source file sections with temporary data within classes (Page 4817) or function blocks (Page 4813). All internal variables lose their values when the method is exited and are reinitialized the next time the method is called.

Methods can only be defined when compiler option (Page 4623) “Permit object-oriented programming" is enabled.

Methods have the following syntax:

Syntax

```
METHOD access-spec name : method_data_type
    // Variable declaration
// Statement section
END_METHOD
```

access-spec stands for the optional access identifier (Page 4768):

- In classes: PUBLIC, PROTECTED, PRIVATE. Default setting: PROTECTED.
- In function blocks: PUBLIC, PRIVATE. PRIVATE is default setting.

name stands for the identifier of the method, while *method_data_type* stands for the data type of the return value.

Permissible keywords for the variable declaration in the declaration section (Page 4821): See Section "Variable declaration" (Page 4824).

Note the methods with *method_data_type* <> VOID: In the statement section (Page 4822), an expression of data type *method_data_type* must be assigned to the method identifier!

For a complete syntax diagram see Figure 7-282 Syntax: Method (Page 4770).

Special feature

The compiler option (Page 4623) “Permit object-oriented programming" needs to be enabled.

Sequence

Methods can only be defined within classes (Page 4817) and function blocks (Page 4813).

- If the "Permit forward declarations" compiler option (Page 4623) is **not** activated, the following applies:
  In the source file, methods must precede the POU from which they are called.
- If the "Permit forward declarations" compiler option (Page 4623) is activated.
  The sequence is arbitrary:

The declaration section (Page 4821) must precede the statement section (Page 4822).

Frequency

Any number of times per class or function block.

Mandatory section

No

Additional information on methods

See “Object-Oriented Programming - OOP (as of kernel V4.5)” (Page 4766) section

Object-oriented interface (as of Kernel V4.5)

Object-oriented interfaces define the call interface for public methods (Page 4819) (access identifier PUBLIC). Method prototypes are defined for this purpose in the object-oriented interface. These methods with the definition "prototype" are formulated in classes (Page 4817) which implement the object-oriented interfaces.

Another interface can be derived from an object-oriented interface (base interface). This derived interface inherits all of the method prototypes of the base interface.

Note

While object-oriented interfaces start and end with the same keywords as the interface section (Page 4807) of a unit (INTERFACE / END_INTERFACE), they have different meanings and are used in different ways.

Object-oriented interfaces can only be defined and used with a SIMOTION Kernel as of version V4.5. The compiler option (Page 4623) "Permit object-oriented programming" also needs to be activated.

Object-oriented interfaces have the following syntax:

Syntax (simplified)

```
INTERFACE name
    // Method prototype
    METHOD method_name : method_data_type
        // Variable declaration (parameters only)
    END_METHOD
    // Other method prototypes
END_INTERFACE
```

name stands for the identifier of the object-oriented interface.

For complete syntax diagrams see Figure 7-290 Syntax: Object-oriented interface (Page 4791) and Figure 7-291 Syntax: Method prototype (Page 4792).

Special features

In classes (Page 4817) that implement object-oriented interfaces, all methods (Page 4819)that are defined as prototypes in the interfaces must be formulated. These methods must always be declared public (access identifier PUBLIC). The call interface (input, output and in/out parameters) specified in the interface must be transferred unchanged.

Variables of object-oriented interfaces can also be declared. Instances of classes that implement the same object-oriented interface can be assigned to these variables.

Sequence

You can define object-oriented interfaces in the interface section (Page 4807) and the implementation section (Page 4809). In most cases, they are declared in the interface section because they are generally declared PUBLIC.

They must always be declared in full before they are used in the declaration of an interface variable or an implementing class. This applies irrespective of the compiler option (Page 4623) "Permit forward declarations".

Frequency

Any number of times per ST source file

Mandatory section

No

Further information about object-oriented interfaces

See section "Object-oriented interfaces" (Page 4790).

Declaration section

The declaration section of a program organization unit (POU) contains the data type definition and the variable declaration of the POU.

The declaration section has the following structure:

Structure

```
// Data type definition
// Variable declaration
```

Sequence

The declaration section has no explicit keywords at the start or end. It begins after the keyword of the respective program organization unit (POU) and ends with the first executable statement of the statement section.

It contains the following in any order:

- Data type definitions (Page 4823):
  User-defined data types (UDT) that are valid locally in the POU.
  Data types cannot be defined within methods.
- Variable declarations (Page 4824):
  Variables and constants that are valid locally in the POU
  Permissible keywords according to the respective POU: See Section "Variable declaration" (Page 4824).

Please note: Identifiers must be declared before they are used.

Frequency

Once per POU

Mandatory section

No

Statement section

The statement section of a POU consists of the individual (executable) statements.

The statement section has the following structure:

Configuration

```
// Statements
```

Sequence

The statement section has no explicit keywords at the start or end. It begins after the declaration section and ends with the keyword of the respective POU.

Frequency

Once per POU

Mandatory section

No

Further information on statements

See following sections:

- Value assignments and expressions (Page 4702)
- Control statements (Page 4719)

- Call of functions and function block calls (Page 4749)
- Calling methods (Page 4773)

Data type definition

For the data type definition, you specify user-defined data types (UDT). You can use them for variable declarations. UDTs can be defined in the interface section, the implementation section, and the declaration section of FCs, FBs, and programs.

The data type definition has the following syntax:

Syntax

```
TYPE
name : data_type_specification;
    // ...
END_TYPE
```

name represents the name of the individual data type that you use for the Variable declarations.

data_type_specification stands for any data type or a structure. Any number of individual data types can appear between TYPE and END_TYPE.

Sequence

You can define UDTs as follows:

- In the Interface section:
  The UDTs are recognized within the ST source file and will be exported
  They can be used in the interface and implementation section for declaration of unit variables and in all POUs for declaration of local variables.
  In addition, they can be used in all units which import this ST source file (in SIMOTION ST with the USES statement).
- In the Implementation section:
  The UDTs are recognized within the ST source file
  They can be used in the implementation section for declaration of unit variables and in all POUs for declaration of local variables.
- In the declaration section of a POU (FC, FB, program, expression, class):
  The UDTs are only recognized locally within the POU
  They can only be used within the POU for declaration of local variables.

UDTs must be defined before they are used in a variable declaration.

Frequency

The TYPE / END_TYPE declaration block may appear more than once in a source file section; any number of UDTs are possible within a declaration block.

Mandatory section

No

Further information on user-defined data types (UDT)

See Section "User-defined data types" (Page 4675).

Variable declaration

A declaration section contains variable declarations and can itself be contained in FCs, FBs, and programs (POUs) as well as in the interface section and the implementation section.

The variable declaration has the following syntax:

Syntax

```
variable_type
   name_list : data_type;
    // ...
END_VAR
```

variable_type represents the keyword of the variable type being declared. The permitted keywords depend on the source file section.

- In the Interface section or Implementation section of an ST source file:
  `VAR_GLOBAL`: Non-retentive unit variable
  `VAR_GLOBAL CONSTANT`: Unit constant
  `VAR_GLOBAL RETAIN`: Retentive unit variable
- In the Declaration section of a function:
  `VAR`: Local variable
  `VAR CONSTANT`: Local constant
  `VAR_INPUT`: Input parameter
  `VAR_OUTPUT`: Output parameters
  `VAR_IN_OUT`: In/out parameter

- In the Declaration section of a function block:
  VAR: Local variable
  - With activated compiler option (Page 4623) "Permit object-oriented programming", the following applies: An access identifier (Page 4740) (PUBLIC, PRIVATE, INTERNAL) can be optionally specified after the keyword. Default: PRIVATE.

  VAR CONSTANT: Local constant
  - With activated compiler option (Page 4623) "Permit object-oriented programming", the following applies: An access identifier (Page 4740) (PUBLIC, PRIVATE, INTERNAL) can be optionally specified after the keyword. Default: PRIVATE.

  VAR RETAIN: Retentive variable
  - With activated compiler option (Page 4623) "Permit object-oriented programming", the following applies: An access identifier (Page 4740) (PRIVATE, INTERNAL) can be optionally specified after the keyword. Default: PRIVATE.

  VAR_TEMP: Temporary variable
  VAR_INPUT: Input parameter
  VAR_OUTPUT: Output parameter
  VAR_IN_OUT: In/out parameter
- In the Declaration section of a program:
  VAR: Local variable
  VAR CONSTANT: Local constant
  VAR_TEMP: Temporary variable
- In the Declaration section of an expression:
  VAR: Local variable
  VAR CONSTANT: Local constant
  VAR_INPUT: Input parameter (as of Version 4.1 of the SIMOTION Kernel)
  VAR_IN_OUT: In/out parameter (as of Version 4.1 of the SIMOTION Kernel)
- In declaration section of a class (Page 4817) (as of version 4.5 of the SIMOTION Kernel):
  VAR: Local variable (static)
  - An access identifier (Page 4768) (PUBLIC, PROTECTED, PRIVATE, INTERNAL) can be optionally specified after the keyword. Default: PROTECTED.

  VAR CONSTANT: Local constant
  - An access identifier (Page 4768) (PUBLIC, PROTECTED, PRIVATE, INTERNAL) can be optionally specified after the keyword. Default: PROTECTED.

  VAR RETAIN: Retentive variable
  - An access identifier (Page 4768) (PROTECTED, PRIVATE, INTERNAL) can be optionally specified after the keyword. Default setting: PROTECTED.

- In the declaration section of a method (Page 4819):
  `VAR`: Local variable
  `VAR CONSTANT`: Local constant
  `VAR RETAIN`: Retentive variable
  `VAR_INPUT`: Input parameters
  `VAR_OUTPUT`: Output parameters
  `VAR_IN_OUT`: In/out parameter
- In the method prototypes within an object-oriented interface (Page 4820):
  `VAR_INPUT`: Input parameter
  `VAR_OUTPUT`: Output parameter
  `VAR_IN_OUT`: In-out parameter

name_list is the list of identifiers of the *data_type* data type to be declared.

Sequence

The variable is declared:

- In the Interface section of the ST source file:
  Permissible keywords: See above at *Syntax*.
  The unit variables are recognized within the ST source file and will be exported.
  They can be used in all POUs of the ST source file.
  In addition, they can be used in all units which import this ST source file (in SIMOTION ST with the USES statement).
- In the Implementation section of the ST source file:
  Permissible keywords: See above at *Syntax*.
  The unit variables are recognized within the ST source file.
  They can be used in all POUs of the ST source file.
- In the declaration section of a class:
  Permissible keywords: See above under *Syntax*.
  The variables are only recognized locally within the class.
- In the declaration section of a POU (FC, FB, program, expression):
  Permissible keywords according to the type of POU: See above at *Syntax*.
  The variables are only recognized locally within the POU.
  Exceptions:
  - You can also access the output parameters of a function block outside the FB.
  - You can access the input parameters of a function block from outside the FB when the compiler option (Page 4623) "Permit language extensions" has been activated.
- In the declaration section of a method:
  Permissible keywords: See above under *Syntax*.
  The variables are only recognized locally within the method.
- In the method prototypes within an object-oriented interface (Page 4820):
  Permissible keywords: See above under *Syntax*.
  The parameters define the call interface of the method.

Variables must be declared before they are used.

Frequency

The number of times the *variable_type* / END_VAR declaration block of a specific variable type can appear depends on the associated source file section:

- In the interface and implementation section of the ST source:
  The declaration blocks may appear more than once.
- In the declaration section of a POU (FC, FB, program, expression, class, method):
  Each declaration block (other than VAR CONSTANT / END_VAR) may appear just once in the declaration section.
- In the method prototypes within an object-oriented interface:
  Each declaration block may only occur once in each method prototype.

Permitted declaration blocks and keywords depending on the associated source file section: See above at *Syntax*.

Any number of variable declarations are possible within a declaration block.

Mandatory section

No

Further information about variable declaration

See Section "Variable declaration" (Page 4694).

Declaring ST source files public and using them

ST has a unit concept which can be used to access global variables, data types, functions (FC), function blocks (FB) and programs from other source files. Thus, for example, you can compile reusable subroutines and make them available.

Unit identifier

Below, unit refers to a program source file (e.g. ST source file, MCC source file). The name of the program source file defined in SIMOTION SCOUT is applied as the identifier.

Optionally, you can set the unit statement as first statement for an ST source file (preceding the interface section). Syntax:

```
UNIT name;
```

name corresponds to the name of the ST source file defined in SIMOTION SCOUT, see Add ST source (Page 4587) or Change the properties of an ST source file (Page 4589).

The unit statement is ignored if the name specified there differs from the name of the ST source file.

Interface section of a unit with Declare Public function

You can enter the following constructs in the interface section of a unit with Declare Public function. The syntax of the constructs is only implied here, for details, see "Interface section (Page 4807)".

- The data type declarations to be declared public
  TYPE
  User-defined data types with their complete declaration.
- The variable declarations to be declared public
  VAR_GLOBAL, VAR_GLOBAL RETAIN, or VAR_GLOBAL CONSTANT
  Non-retentive and retentive unit variables and unit constants with their complete declaration.
- The POU to be declared public (functions, function blocks, programs and classes)
  Specify each POU (function, function block, program or class) to be declared public with the relevant keyword (optional within the TYPE / END_TYPE construct). Close each entry with a semicolon.
  - FUNCTION_BLOCK *fb_name*;
  - FUNCTION *fc_name*;
  - PROGRAM *program_name*;
  - CLASS *class_name*; (as of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) "Permit object-oriented programming" activated)

  If the "Permit forward declarations" compiler option (Page 4623) is activated, they are also interpreted as POU prototypes for the forward declaration (Page 4929).
  You program the POU itself in the implementation section (Page 4809) of the ST source file.
- The object-oriented interfaces to be declared public as of version V4.5 of the SIMOTION Kernel with compiler option (Page 4623) "Permit object-oriented programming" activated.
  INTERFACE *if_name* ... END_INTERFACE
  Object-oriented interfaces with their complete declaration.

The information can be specified in any sequence:

Note

The following further specifications are possible in the interface section, they are listed **before** the data types, variables and POUs to be declared public:

1. Specification of utilized technology packages (USEPACKAGE ...).
2. Specification of utilized libraries (USELIB ...).
3. Reference to other units in order to use their public components (USES ...).

These imported technology packages, libraries and units are also declared public. For inheritance, see "USES statement in a using unit (Page 4830)".

You must adhere to the order presented for the specifications in the interface section of a unit (ST source file), see "Interface section (Page 4807)". Otherwise, error-free compilation of the ST source file will not be possible.

Programs which are to be assigned to a task in the execution system must be listed in the interface section (see *Configuring the execution system* in the *SIMOTION Basic Functions*

Function Manual). The compiler outputs a warning message if programs are not declared public in the interface section of an ST source file.

Functions, function blocks, and programs that are only used in the ST source file should not be listed in the interface section.

Example of a unit with Declare Public function

Below is an example of a unit with Declare Public function (*myUnit_A*). It is imported by *myUnit_B* (see Example of a using unit (Page 4831)).

Table 7-437 Example of a unit with Declare Public function

```
UNIT myUnit_A;    // Optional, name of the ST source file

INTERFACE
    // ... USES statement also possible here
    TYPE        // Declaration of data types to be declared public
        color : (RED, GREEN, BLUE);
    END_TYPE
    VAR_GLOBAL
        cycle : INT := 1;  // Declaration of unit variables to be declared public
                           // unit variables to be exported
    END_VAR
    FUNCTION myFC;         // DECLARE PUBLIC statement of an FC
    FUNCTION_BLOCK myFB;   // DECLARE PUBLIC statement of an FB
    CLASS myClass;         // DECLARE PUBLIC statement of a class
                           // (as of version V4.5 of the SIMOTION Kernel)
                           // and activated compiler option
                           // "Permit object-oriented programming")
    PROGRAM myProgram_A;   // Export statement of a program
                           // (to interface with the execution system)
END_INTERFACE
IMPLEMENTATION
    Function myFC : LREAL   // Function written out
        ;     // ... (Statements)
    END_FUNCTION

    Function_BLOCK myFB     // Function block formulated
        ;     // ... (Statements)
    END_FUNCTION_BLOCK

    CLASS myClass          // Class formulated
                           // (as of version V4.5 of the SIMOTION Kernel)
                           // and activated compiler option
                           // "Permit object-oriented programming")
        ;     // ... (Statements)
    END_CLASS;

    PROGRAM myProgram_A    // Program formulated
        ;     // ... (Statements)
    END_PROGRAM
END_IMPLEMENTATION
```

USES statement in a using unit

Enter the following statement in the interface section or implementation section of a using unit:

```
USES unit_name-list
```

unit_name-list is a list of the units to be used separated by commas.

Example:

```
USES unit_1, unit_2, unit_3;
```

This enables you to access the following elements that are declared public in the interface section of the units used (e.g. ST source file, MCC unit):

- User-defined data types (UDT)
- Unit variables and unit constants
- Programs, functions, function blocks and classes
- Used technology packages, libraries and units

You can use the used elements as if they existed in the current unit.

Note

The keyword USES can only occur once in the interface section or in the implementation section of a unit. When multiple units are to be used, enter them as a list separated by commas after the keyword USES.

The USES statement can appear in either the interface section or the implementation section of a unit. This has far-reaching implications:

Table 7-438 Implications regarding placement of USES statement in interface section or in implementation section of the ST source file

Effect	USES statement in the interface section	USES statement in the implementation section
Inheritance	The current unit continues declaring the used unit public; the used unit is inherited by all other units that access the current unit. Example: 1. Unit B uses Unit A in the **interface** section. 2. Unit C in turn uses Unit B. 3. Then Unit C also uses Unit A automatically. A → B → C ⇒ A → C Because of inheritance, Unit A need not be used explicitly in Unit C.	Inheritance is interrupted. Example: 1. Unit B uses Unit A in the **implementation** section. 2. Unit C in turn uses Unit B. 3. Then Unit C has no automatic access to Unit A. Unit C must explicitly use Unit A if it wants to access Unit A.
Variable declaration	The declaration of a unit variable of a used data type is possible in: • Interface section • Implementation section	The declaration of a unit variable of a used data type is only possible in the implementation section.

Note

You will find tips for use of unit variables in the SIMOTION Basic Functions Function Manual.

Example of a using unit

Below is an example of a using unit (*myUnit_B*). This uses the *myUnit_A* unit from the Example of a unit with Declare Public function (Page 4829).

Table 7-439 Example of a using unit

```
UNIT myUnit_B;        // Optional, name of the ST source file
INTERFACE
    // ...  if applicable USES statement
    PROGRAM myProgram_B;
    // Specification of declared public programs, FBs, FCs, classes,
    // Data types and unit variables
END_INTERFACE

IMPLEMENTATION
    USES myUnit_A;              // Specification of used unit

    VAR_GLOBAL
        myInstance : myFB;      // Declaration of an instance
                                // of the public FB of the used unit
        mycolor : color;        // Declaration of a variable
                                // of the public data type of the used unit
    END_VAR

    PROGRAM myProgram_B
        mycolor := GREEN;       // Value assignment to variables
                                // of the public data type of the used unit
        cycle := cycle + 1;     // Value assignment to
                                // public variables of the used unit
    END_PROGRAM
END_IMPLEMENTATION
```

7.2.7.2 Variables in SIMOTION

This summarizes the variables available in ST.

Variable model

The following table shows all the variable types available for programming with ST.

- System variables of the SIMOTION device and the technology objects
- Global user variables (I/O variables, device-global variables, unit variables)
- Local user variables (variables within a program, a function, a method, a function block or a class)

System variables

Variable type	Meaning
System variables of the SIMOTION device	Each SIMOTION device and technology object has specific system variables. These can be accessed as follows: • Within the SIMOTION device from all programs • From HMI devices You can monitor system variables in the symbol browser.
System variables of technology objects	

Global user variables

Variable type	Meaning
I/O variables	You can assign symbolic names to the I/O addresses of the SIMOTION device or the peripherals. This allows you to have the following direct accesses and process image accesses to the I/O: • Within the SIMOTION device from all programs • From HMI devices You create these variables in the symbol browser after you have selected the I/O element in the project navigator. You can monitor I/O variables in the symbol browser.
Global device variables	User-defined variables which can be accessed by all SIMOTION device programs and HMI devices. You create these variables in the symbol browser after you have selected the GLOBAL DEVICE VARIABLES element in the project navigator. Global device variables can be defined as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor global device variables in the symbol browser.
Unit variables	User-defined variables that all programs, function blocks, and functions (e.g. ST source, MCC source, LAD/FBD source) can access within a unit. Declare these variables in the unit: • In the interface section: These variables are declared public and can be used in other units (e.g. ST source files, MCC units, LAD/FBD units). They are also available on HMI-device as standard. • In the implementation section: You can only access these variables within the associated unit. You can declare unit variables as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor unit variables in the symbol browser.

Local user variables

Variable type	Meaning
	User-defined variables which can be accessed from within the program (or function, function block) in which they were defined.
Variable of a program (program variable)	Variable is declared in a program. The variable can only be accessed within this program. A differentiation is made between static, temporary and retentive variables: • Static variables are initialized according to the memory area in which they are stored. Specify this memory area by means of a compiler option. By default, the static variables are initialized depending on the task to which the program is assigned (see *SIMOTION Basic Functions* Function Manual). You can monitor static variables in the symbol browser. • Temporary variables are initialized every time the program in a task is called. Temporary variables cannot be monitored in the symbol browser. • Retentive variables (as of version V4.5 of the SIMOTION Kernel) retain their value even when the SIMOTION device power supply is disconnected. You can monitor retentive variables in the symbol browser.
Variable of a function or method	Variable is declared in a function (FC) or method. The variable can only be accessed within this FC or method. Variables in functions or methods are temporary; they are initialized each time the FC or method is called. They cannot be monitored in the symbol browser.
Variable of a function block	Variable is declared in a function (FB). The variable can only be accessed within this function block. A differentiation is made between static, temporary and retentive variables: • Static variables retain their value when the FB terminates. They are initialized only when the instance of the FB is initialized; this depends on the variable type with which the instance of the FB was declared. You can monitor static variables in the symbol browser. Only when compiler option "Permit object-oriented programming" is activated: Static variables with the PUBLIC access identifier can be accessed externally. • Temporary variables lose their value when the FB terminates. The next time the FB is called, they are reinitialized. Temporary variables cannot be monitored in the symbol browser. • Retentive variables (as of version V4.5 of the SIMOTION Kernel) retain their value even when the SIMOTION device power supply is disconnected. You can monitor retentive variables in the symbol browser.
Variables of a class (as of version V4.5 of the SIMOTION Kernel)	Variable is declared in a class. A differentiation is made between static and retentive variables: • Static variables are initialized only when the instance of the class is initialized; this depends on the variable type with which the instance of the class has been declared. Static variables can only be accessed from within this class and derived classes. Only static variables with the PUBLIC access identifier can be accessed from outside the class. • Retentive variables retain their value even when the SIMOTION device power supply is disconnected. Retentive variables can only be accessed from within this class and derived classes. They cannot be accessed from outside the class. You can monitor the variables in the symbol browser.

Further information is available from the following sources:

- In the corresponding list manuals, you can find the compressed information on all system variables of the SIMOTION technology packages and SIMOTION devices.
- For more details on the use of system variables of technology objects, please refer to the *SIMOTION Motion Control Technology Objects* Function Manuals.
- In the *SIMOTION Basic Functions* Function Manual you can find information on how to access system variables and configuration data.
- This documentation contains information on:
  - Accessing I/O addresses with I/O variables (see Direct access and process image of the cyclic tasks (Page 4872))
  - Accessing the process image (see Access to the fixed process image of the BackgroundTask (Page 4881))
  - Creating and using global device variables (see Using global device variables (Page 4843))
  - Use of unit variables and local variables (static and temporary variables).

Note

Please note that downloading the ST source file to the target system and running tasks affect variable initialization and thus the contents of the variables, see Time of the variable initialization (Page 4849).

Unit variables

Unit variables are valid throughout the entire ST source file, i.e. they can be accessed in any source file section.

Unit variables are declared in the interface and/or implementation section of an ST source file; the location of the declaration determines the validity of the unit variable:

- If you declare the unit variables in the interface section, you create variables that can be used in other program sources (e.g. ST source files, MMC units). More information about declaring data public and using public data in other program sources can be found in Declaring ST source files public and using them (Page 4827).
  By default, these unit variables are also available on HMI devices. The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.
- If you declare the unit variables in the implementation section, you create variables that can be used by all program organization units (POUs) of the current source file.

You can change the default setting for the HMI export of the unit variables using a pragma within a declaration block, see Variables and HMI devices (Page 4864) and Controlling compiler with attributes (Page 4922).

You can define unit variables with different behavior, e.g. in case of power failure:

- Non-retentive unit variables (keyword VAR_GLOBAL): its value is lost in the event of a power failure.
- Retentive unit variables (keyword VAR_GLOBAL RETAIN): its value remains in the event of a power failure.
- Unit constants (keyword VAR_GLOBAL CONSTANT): its value is retained unchanged (see Constants (Page 4701)).

You will find tips for the efficient use of unit variables in the *SIMOTION Basic Functions* Function Manual.

Non-retentive unit variables

Non-retentive unit variables lose their value in the event of a power failure.

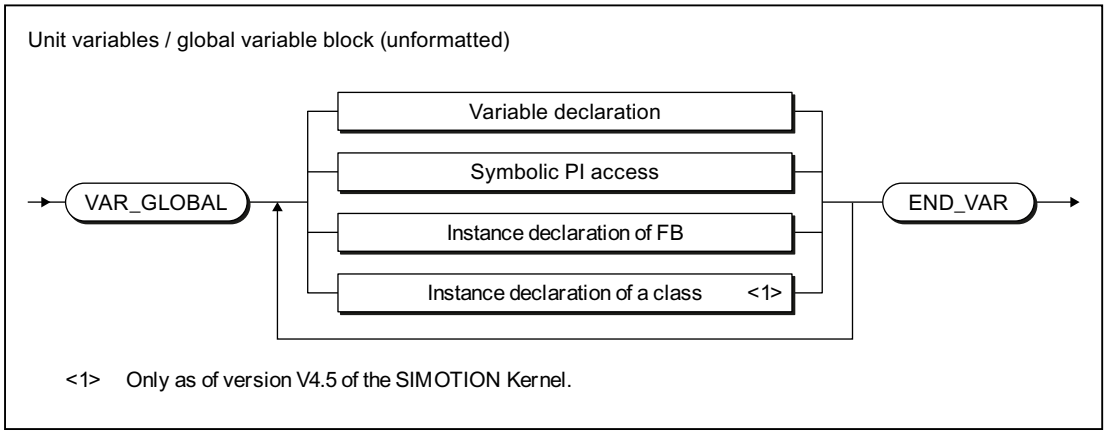


Figure 7-293 Syntax: Unit variables

This declaration block may appear more than once within an interface or implementation section. You specify the variable name and data type for the variable declaration (see Overview of all variable declarations (Page 4695) and Initialization of variables or data types (Page 4697)).

For the scope of the declaration and the HMI export, see Unit variables (Page 4834).

Note

For initialization of the non-retentive unit variables:

- See Initialization of non-retentive global variables (Page 4851).
- The behavior during downloading can be set (**Options > Settings** menu command, **Download** tab, **Initialize non-retentive program data and global device variables** checkbox)
- The type of version ID and therefore the initialization behavior on downloading depends on the SIMOTION Kernel version. For details, see Version ID of global variables and their initialization during download (Page 4858).

Table 7-440 Examples of non-retentive unit variables

```
INTERFACE
    VAR_GLOBAL    // These variables are declared public.
        rotation1        : INT;
        field1           : ARRAY [1..10] OF REAL;
        flag1            : BOOL;
        motor1           : motor;    // Instance declaration
    END_VAR
END_INTERFACE

IMPLEMENTATION
    VAR_GLOBAL    // These variables are not declared public.
        rotation2        : INT;
        field2           : ARRAY [1..10] OF REAL;
        flag2            : BOOL;
        motor2           : motor;    // Instance declaration
    END_VAR
END_IMPLEMENTATION
```

Retentive unit variables

Retentive unit variables permit permanent storage of variable values even throughout a power failure.

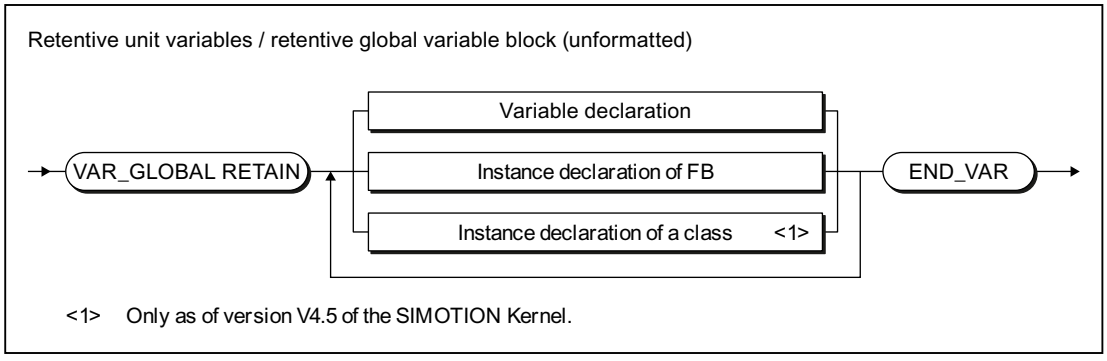


Figure 7-294 Syntax: Retentive global variable block

This declaration block may appear more than once within an interface or implementation section. You specify the variable name and data type for the variable declaration (see Overview of all variable declarations (Page 4695) and Initialization of variables or data types (Page 4697)).

For the scope of the declaration and the HMI export, see Unit variables (Page 4834).

Note

- For initialization of the retentive unit variables:
  - See Initialization of retentive global variables (Page 4849).
  - The behavior during downloading can be set (**Options > Settings** menu command, **Download** tab, **Initialize retentive program data and global device variables** checkbox)
  - The type of version ID and therefore the initialization behavior on downloading depends on the SIMOTION Kernel version. For details, see Version ID of global variables and their initialization during download (Page 4858).
- The amount of memory available for retentive variables depends on the device (see quantity framework in the SIMOTION SCOUT Configuration Manual).
  To make efficient use of limited memory space, use the memory in a single ST source file and sort the variables in descending order!
- Check the capacity utilization of the retentive memory in SIMOTION SCOUT.
  In online mode, call the **device diagnostics** of the SIMOTION device to be checked (see online help). In the **System utilization** tab under **Retentive data**, you can see how much memory is available.

Table 7-441 Examples of retentive unit variables

```
VAR_GLOBAL RETAIN
    Measuring field : ARRAY[1..10] OF REAL;
    Pass : INT;
    Switch : BOOL;
END_VAR
```

Local variables (static and temporary variables)

Local variables are valid only in the source file section (e.g. program, FC or FB) in which they were declared. We distinguish between the following:

- Static variables (Page 4840):
  Static variables retain their value over all passes of the source file section (block memory).
- Temporary variables (Page 4841):
  Temporary variables are initialized each time the unit section is called again.
- Retentive local variables (Page 4841)
  SIMOTION Kernel as of version V4.5.
  They retain their value:
  - in the event of a power failure
  - when the task is started
  - over all passes of the source file section

See also: Initialization of local variables (Page 4853).

Note

Local variables cannot be accessed outside the source file section in which they were declared.

The following table provides an overview of the declaration of retentive, static and temporary variables. It shows the source file sections in which these variables can be declared and the keywords that can be used to declare them.

Table 7-442 Keywords for declaring retentive, static and temporary variables depending on source file section

Source file section	Keywords for the declaration		
	Static variables	Temporary variables	Retentive local variables (as of kernel V4.5)
Function	–	VAR / END_VAR or VAR_TEMP / END_VAR or VAR_INPUT / END_VAR or VAR_IN_OUT / END_VAR[2] or VAR_OUTPUT / /END_VAR	–
Expression	–	VAR/ END_VAR or VAR_INPUT / END_VAR or VAR_IN_OUT / END_VAR[2]	–
Function block	VAR / END_VAR[1] or VAR_INPUT / END_VAR[1] or VAR_OUTPUT / END_VAR[1]	VAR_TEMP / END_VAR or VAR_IN_OUT / END_VAR[2]	VAR RETAIN / END_VAR
Program	VAR / END_VAR[3]	VAR_TEMP / END_VAR	VAR RETAIN / END_VAR
Methods (within a class or a function block)[4]	–	VAR / END_VAR or VAR_TEMP / END_VAR or VAR_INPUT / END_VAR or VAR_IN_OUT / END_VAR[2] or VAR_OUTPUT / /END_VAR	–
Class[4]	VAR / END_VAR[1]	–	VAR RETAIN / END_VAR

[1] The initialization of the variable depends on initialization of the declared instance. See Initialization of instances of function blocks (FBs) or classes (Page 4856).

[2] The reference (pointer) for the transferred variable is temporary.

[3] The initialization of the variables depends on the memory area in which they are stored. See Initialization of static program variables (Page 4854).

[4] Only with active compiler option “Permit object-oriented programming”.

Note

Please note that downloading the ST source file to the target system and running tasks affect variable initialization and thus the contents of the variables, see Time of the variable initialization (Page 4849).

Table 7-443 Examples of static and temporary variables

```
IMPLEMENTATION
    FUNCTION testFkt
        VAR            // Declaration of temporary variables
            flag : BOOL;
        END_VAR
    END_FUNCTION
    FUNCTION_BLOCK testFbst
        VAR RETAIN     // Declaration of retentive variables
            position1 : LREAL
        END_VAR

        VAR            // Declaration of static variables
            rotation1 : INT;
        END_VAR

        VAR_TEMP       // Declaration of temporary variables
            help1, help2 : REAL;
        END_VAR
    END_FUNCTION_BLOCK
    PROGRAM testPrg;
        VAR RETAIN     // Declaration of retentive variables
            position2 : LREAL
        END_VAR

        VAR           // Declaration of static variables
            rotation2 : INT;
        END_VAR

        VAR_TEMP      // Declaration of temporary variables
            help1, help2 : REAL;
        END_VAR
    END_PROGRAM
    CLASS testcls
        VAR RETAIN     // Declaration of retentive variables
            position3 : LREAL
        END_VAR

        VAR           // Declaration of static variables
            rotation3 : INT;
        END_VAR
    END_CLASS
END_IMPLEMENTATION
```

Static variables

Static variables retain their most recent value when the source file section is exited. This value is used again at the next call.

The following source file sections contain static variables:

- Programs
- Function blocks
- Classes (as of version V4.5 of the SIMOTION Kernel and with compiler option "Permit object-oriented programming" enabled)

Static variables are declared in a static variable block.

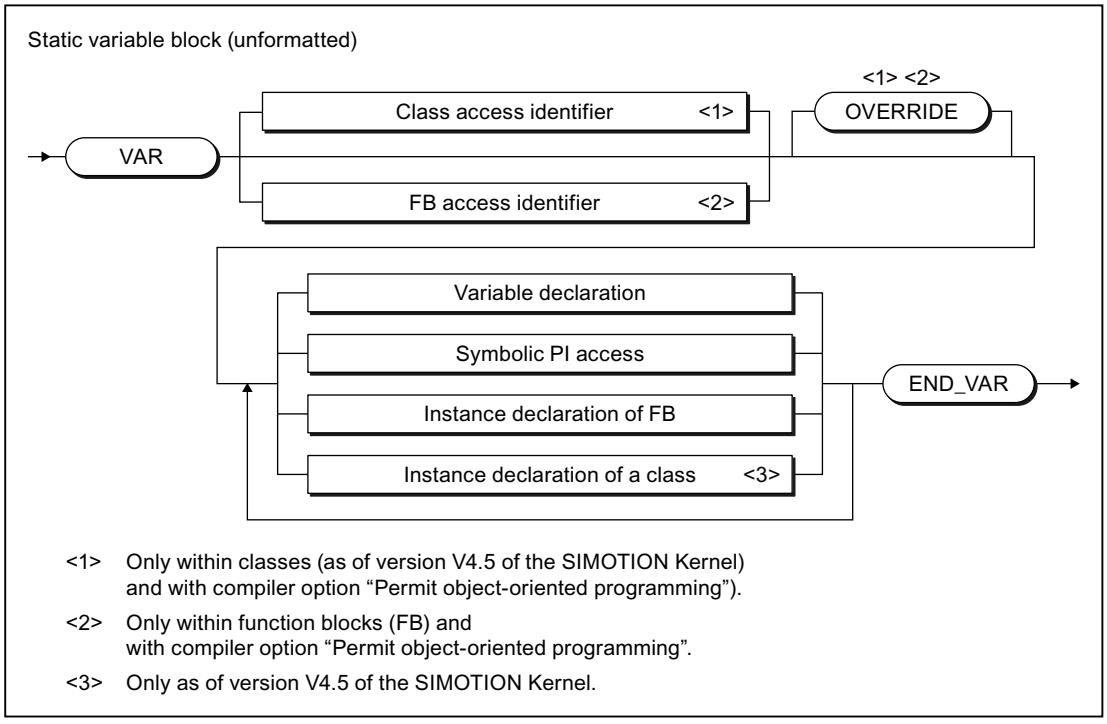


Figure 7-295 Syntax: Static variable block

You can do the following in the static variable block, according to the syntax in the figure:

- Declare variables (name and data type), optionally with initialization.
- Declare symbolic accesses to the process image of the BackgroundTask.
- Declare instances of the function blocks.
- Declaring instances of classes (SIMOTION Kernel as of version V4.5)

For initialization of the static variables:

- In programs: Depending on the execution behavior to which the program is assigned (see *SIMOTION Basic Functions* Function Manual).
  See also Initialization of static program variables (Page 4854).
- In function blocks: Depending on the initialization of the declared instance.
  See also Initialization of instances of function blocks (FBs) (Page 4856).

Temporary variables

Temporary variables are initialized each time the source file section is called. Their value is retained only during execution of the source file section.

The following source file sections contain temporary variables:

- Programs
- Function blocks
- Functions
- Methods (within classes or function blocks, with compiler option "Permit object-oriented programming")
- Expressions

In functions and expressions, you declare temporary variables in the FB temporary variable block (see following figure):

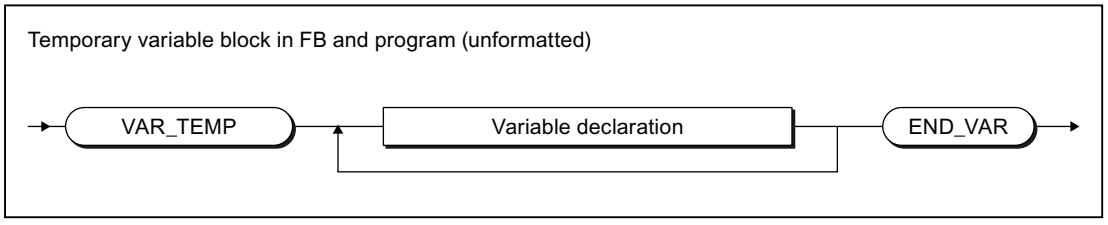


Figure 7-296 Syntax: Temporary variable block in the FB or program

In functions, methods and expressions, you declare temporary variables in the FC temporary variable block (see following figure):

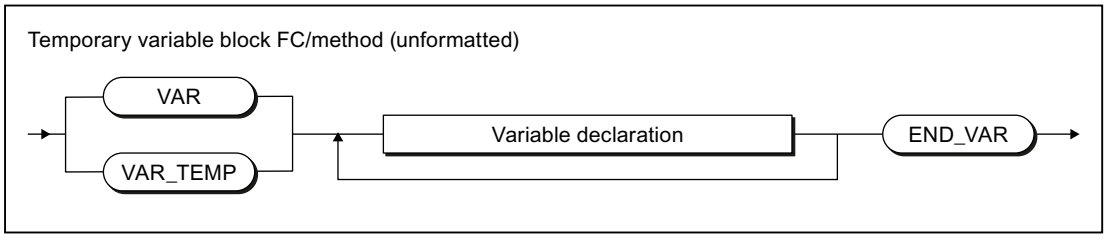


Figure 7-297 Syntax: Temporary variable block in an FC or a method

Retentive local variables (as of kernel V4.5)

You can declare retentive local variables in SIMOTION Kernel as of version V4.5.

They retain their value over all passes of the source file section in the event of a power failure or if the task is started.

The following source file sections contain retentive local variables:

- Programs
- Function blocks
- Classes (with compiler option "Permit object-oriented programming")

They are declared in the retentive local variable block.

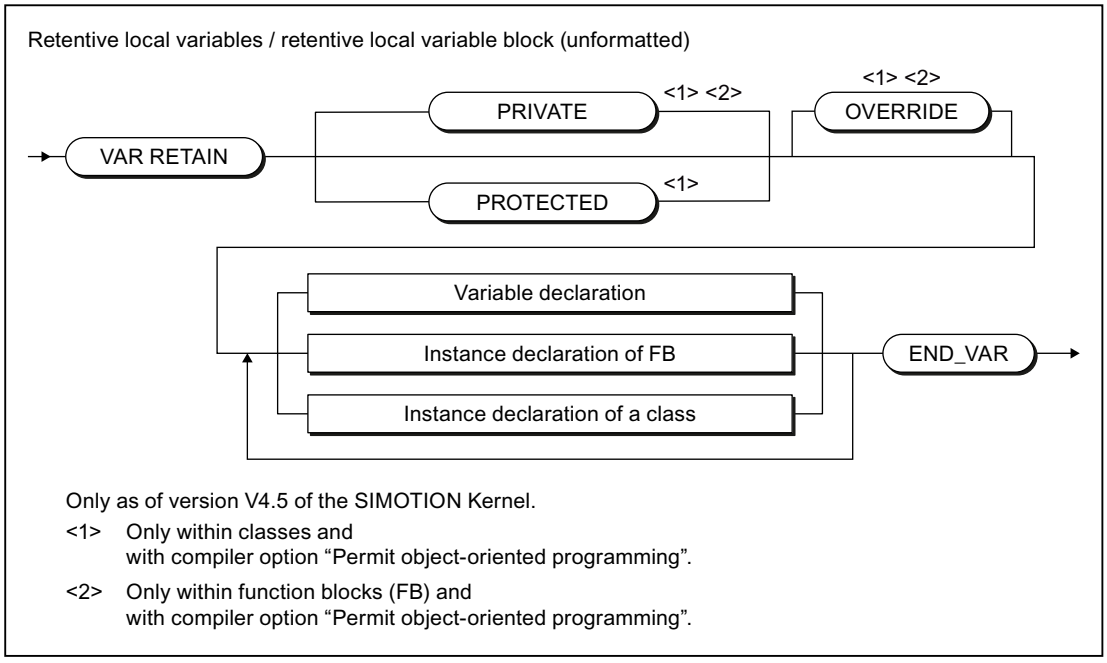


Figure 7-298 Syntax: Retentive local variable block

You can do the following in the retentive local variable block, according to the syntax in the figure:

- Declare variables (name and data type), optionally with initialization.
  Exception. No variables of data types can be declared in the retentive local variable block if they exclusively contain elements that cannot be saved (e.g. variables of data types of the technology objects).
- Declare instances of the function blocks.
  Exception: No instances of function blocks can be declared in the retentive local variable block if they contain the following:
  - Retentive local variables
  - Instances of system function blocks
- Declaring instances of classes (SIMOTION Kernel as of version V4.5)
  Exception: No instances of classes can be declared in the retentive local variable block if they contain the following:
  - Retentive local variables
  - Instances of system function blocks

For initialization of retentive local variables:

- In programs: dependent on the compiler option "Only create program instance data once”.
  See also Initialization of retentive local variables of programs (Page 4855).
- In function blocks or classes: dependent on the initialization of the retentive variables of the declared instance.
  See also Initialization of retentive local variables of function blocks and classes (Page 4857).

Use of global device variables

Global device variables are user-defined variables that you can access from all program sources (e.g. ST source files, MCC source files) of a SIMOTION device.

Global device variables are created in the symbol browser tab of the detail view; to do this, you must be working in offline mode.

Here is a brief overview of the procedure:

1. In the project navigator of SIMOTION SCOUT, select the **GLOBAL DEVICE VARIABLES** element in the SIMOTION device subtree.
2. In the detail view, select the **Symbol browser** tab and scroll down to the end of the variable table (empty row).
3. In the last (empty) row of the table, enter or select the following:
   - **Name** of variable
   - **Data type** of variable (only elementary data types are permitted)
4. Optionally, you can make the following entries:
   - Activation of **Retain** checkbox (This declares the variable as retentive, so that its value will be retained after a power failure.)
   - **Field length** (array size)
   - **Initial value** (if array, for each element)
   - **Display format** (if array, for each element)

You can now access this variable using the symbol browser or any program of the SIMOTION device.

In ST source files, you can use a global device variable, just like any other variable.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the global device variable with *_device.var-name*.

An alternative to global device variables is the declaration of unit variables in a separate unit, which is imported into other units. This has the following advantages:

1. Variable structures can be used.
2. The initialization of the variables during the STOP-RUN transition is possible (via Program in StartupTask).
3. For newly created global unit variables, a download in RUN is also possible.

Please refer to the SIMOTION Basic Functions Function Manual.

Memory ranges of the variable types

The different variable types are stored in different memory areas, which are initialized at different times. The table shows:

- The available memory areas for variable types that are declared in ST source files (possibly dependent on the version of the SIMOTION Kernel).
- The initialization time for each memory area.

An explanation using an example is contained in the Example for memory areas (Page 4846) section.

Table 7-444 Memory areas assigned to different variable types and their initialization

Memory area	Assigned variable types	Initialization[3]
Retentive memory	• Retentive unit variables • As of version V4.5 of the SIMOTION Kernel: Retentive local variables of programs and function blocks	During download using the download settings
User memory of unit	• Non-retentive unit variables • Function block instances declared with VAR_GLOBAL, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT) • As of version V4.5 of the SIMOTION Kernel: Instances of classes declared with VAR_GLOBAL, including the associated variables (VAR) Also for the activated "Create program instance data only once" compiler option (Page 4623): • Local variables of the unit programs declared with VAR • Function block instances declared with VAR_GLOBAL, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT) • As of version V4.5 of the SIMOTION Kernel: Instances of classes declared with VAR within the programs of the unit, including the associated variables (VAR)	• When the device is switched on • During download using the download settings • For transition to the RUN mode: – Version V4.2 and higher of the SIMOTION Kernel: By activating the check box on the SIMOTION device, **Initialization of non-retentive global variables and program data during STOP-RUN transition**. – As of Version V4.1 of the SIMOTION Kernel: If the associated declaration block specifies the following pragma: { BlockInit_OnDeviceRun := ALWAYS; } See also Controlling compiler with attributes (Page 4922)
User memory of task	For the deactivated "Create program instance data only once" compiler option (Page 4623) (default): • Local variables declared with VAR of the assigned programs • Function block instances declared with VAR within the assigned programs, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT) • As of version V4.5 of the SIMOTION Kernel: Instances of classes declared with VAR within the programs of the unit, including the associated variables (VAR)	According to execution behavior of task: • Sequential tasks: Each time task is started • Cyclic tasks: For CPU transition to the RUN mode

Memory area	Assigned variable types	Initialization[3]
Local data stack of task[2]	• Reference (pointer) to the program called in the task • Local variables declared with VAR_TEMP of the program called in the task	On each call of the program in the task
	• Reference (pointer) to called function block instances • Local variables of function blocks declared with VAR_TEMP • In/out parameters of function blocks declared with VAR_IN_OUT[1]	Each time the function block instance is called
	• Variables of called functions or methods declared with VAR, VAR_INPUT, VAR_OUTPUT or VAR_IN_OUT[1] • Return value of called functions or methods	Each time the function or method is called

[1] References (pointers) to the transferred variables.

[2] See also Memory requirement of the variables on the local data stack (Page 4848).

[3] For a detailed description of the initialization behavior of the individual variable types, see Time of the variable initialization (Page 4849).

Example of memory areas

Table 7-445 Example of memory areas of the variable types - Part 1

```
INTERFACE
// The statements in the interface section specify,
// which source contents are declared public.
    FUNCTION FC1;
    FUNCTION_BLOCK FB1;
    PROGRAM p1;

    // Unit variables of the interface section are also visible
    // on HMI devices.
    VAR_GLOBAL               // Non-retentive unit variables
                             // are present in the UNIT user memory
        u1_if : INT;
    END_VAR
    VAR_GLOBAL CONSTANT      // Unit constants are located
                             // in the unit user memory
    END_VAR
    VAR_GLOBAL RETAIN        // Retentive unit variables are located
                             // in the retentive (power-fail-safe) memory
    END_VAR
END_INTERFACE

IMPLEMENTATION
// The implementation section contains the executable code sections
// in different program organization units (POU)
// A POU can be a program, FC, or FB.
    // Unit variables of the implementation section can only be used
    // within the source file.
    VAR_GLOBAL               // Non-retentive unit variables are located
                             // in the unit user memory
        u1_glob : INT;
    END_VAR
    VAR_GLOBAL CONSTANT      // Unit constants are located
                             // in the unit user memory
    END_VAR
    VAR_GLOBAL RETAIN        // Retentive unit variables are located
                             // in the retentive (power-fail-safe) memory
    END_VAR
//-----------------------------------------------------------------------
```

Table 7-446 Example of memory areas of the variable types - Part 2

```
// Continuation
//-----------------------------------------------------------------------
    FUNCTION_BLOCK FB1 // Declaration of an instance
    // instance determines where its data are located:
    // - as VAR_GLOBAL in a unit:
    // in the unit user memory
    // - as VAR in a program:
    // in the user memory of the task (default)
    // - As VAR in a function block:
    //     in the user memory of the unit or task,
    //     depending on the instance declaration of the higher-level FB
    // When the instance is called, a pointer to the instance data
    // is placed on the stack of the calling task

        VAR_INPUT        // Input parameters
                         // are in the user memory
                         // are written when the instance is called
            fb_in     : INT;
        END_VAR
        VAR_OUTPUT       // Output parameters
                         // are in the user memory
            fb_out    : INT;
        END_VAR
        VAR_IN_OUT       // In/out parameter
                         // references are in the user memory
                         // are written when the instance is called
            fb_in_out : INT;
        END_VAR

        VAR              // Static variables
                         // are in the user memory
                         // can be used locally in the FB
             fb_var1   : INT;
        END_VAR

        VAR_TEMP         // Temporary variables
                         // are on the stack of the calling task
                         // are initialized on each call
            fb_temp1  : INT;
        END_VAR

        // Code is in the user memory of the unit
        fb_var1   := fb_var1 + 1;
        fb_out    := fb_var1;
        fb_temp1  := fb_in_out;
        fb_in_out := fb_temp1 + fb_in;
    END_FUNCTION_BLOCK
//-----------------------------------------------------------------------
```

Table 7-447 Example of memory areas of the variable types - Part 3

```
// Continuation
//-----------------------------------------------------------------------
    FUNCTION FC1 : INT     // The function data is on the
        // stack of the calling task; they are initialized each time
        // the function is called.
        // The return value is on the stack of the calling task

        VAR_INPUT        // Input parameters
                         // are on the stack of the calling task
                         // are written when the function is called
            fc_in   : INT;
        END_VAR

        VAR              // Temporary variables
                         // are on the stack of the calling task
            fc_var  : INT;
        END_VAR
        // Code is in the user memory of the unit
        fc_var := 567;
        fc1 := fc_in + fc_var;
    END_FUNCTION

    PROGRAM p1
        VAR              // By default, variables are located in the
                         // in the user memory of the task
            p_var   : INT;
            p_varFB : FB1;
        END_VAR

        VAR_TEMP         // Temporary variables
                         // are on the stack of the task,
                         // are initialized on each task pass
            p_temp  : INT;
        END_VAR

        // Code is in the user memory of the unit
        p_temp  := p_var;
        p_varFB (fb_in_out := p_temp);
        u1_glob := 4711;
    END_PROGRAM
END_IMPLEMENTATION
```

Memory requirement of the variables on the local data stack

The variables stored on the local data stack of a task are listed in Memory ranges of the variable types (Page 4844). You set the stack size for each task in the **Task configuration** tab. For more information, see the Description for each user task in the Basic Functions Function Manual.

Note the following for memory requirements in the local stack:

- Temporary local variables require their own size on the stack.
- Global variables and static local variables do not require any resources on the stack.
  If you are using them as input parameters for a function, however, they require their own data size on the stack.

- Even if a function is called more than once in a task, it only uses the stack's resources once.
- Variables of type BOOL require 1 byte on the stack.

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4914) function.

Time of the variable initialization

The timing of the variable initialization is determined by:

- Memory area to which the variable is assigned
- Operator actions (e.g. source file download to the target system)
- Execution behavior of the task (sequential, cyclic) to which the program was assigned.

All variable types and the timing of their variable initialization are shown in the following tables. You will find basic information about tasks in the *SIMOTION Basic Functions* Function Manual.

The behavior for variable initialization during download can be set: To do this, as a default setting select the **Options > Settings** menu and the **Download** tab or define the setting during the current download.

Note

You can upload values of unit variables or global device variables from the SIMOTION device into SIMOTION SCOUT and save them in XML format.

1. Save the required data segments of the unit variables or global device variables as a data set with the function *_saveUnitDataSet*.
2. Use the **Save variables** function in SIMOTION SCOUT.

You can use the **Restore variables** function to download these data sets and variables back to the SIMOTION device.

For more information, refer to the SIMOTION SCOUT Configuration Manual.

This makes it possible, for example, to obtain this data, even if it is initialized by a project download or if it becomes unusable (e.g. due to a version change of SIMOTION SCOUT).

Initialization of retentive global variables

Retentive variables retain their last value after a loss of power. All other data is reinitialized when the device is switched on again.

Retentive global variables are initialized:

- When the backup or buffer for retentive data fails.
- When a memory reset (MRES) is performed.
- With the restart function (Del. SRAM) in SIMOTION P320 or P350.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the retentive data.

- When a download is performed according to the following description.
- With a firmware update (upgrade) or activation of a kernel in accordance with the following description.

Behavior during download

Table 7-448 Initializing retentive global variables during download

Variable type	Time of the variable initialization
Retentive global device variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive global device variables are initialized. • **No**[3]: The retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 4858).
Retentive unit variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 4858).

[1] Default setting in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the retentive unit variables corresponds to a VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** and **SIMOTION LAD/FBD** programming languages:
A data block of the retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute has been specified within a pragma: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL RETAIN during a change*. All the variables declared with VAR_GLOBAL RETAIN up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see the SIMOTION Basic Functions Function Manual.

Behavior during upgrade or configuration change

When the SIMOTION device is upgraded to a new version of the SIMOTION Kernel or if the configuration is changed, the retentive variables are initialized as described below:

Table 7-449 Initialization of retentive global variables during upgrade or configuration change

Variable type	Time of the variable initialization
Retentive global device variables	This data is always initialized.
Retentive unit variables	This data can be retained. See section "Retaining retentive data" in the "Basic Functions for Modular Machines" Function Manual.

Initialization of non-retentive global variables

Non-retentive global variables lose their value during power outages. They are initialized:

- During the initialization of retentive global variables (Page 4849), e.g. during a firmware update or overall reset (MRES).
- During switch-on.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the non-retentive data.
- During a download as described in the following table.
- During the transition from STOP to RUN mode as described at the end of the section.

Behavior during download

Table 7-450 Initializing non-retentive global variables during download

Variable type	Time of the variable initialization
Non-retentive global device variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive global device variables are initialized. • **No**[3]: The non-retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 4858).
Non-retentive unit variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the non-retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 4858).

[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the non-retentive unit variables corresponds to a VAR_GLOBAL/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL in the interface section or implementation section of the declaration table: Pragma lines within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN, provided the following condition is fulfilled:
With the **SIMOTION ST**
programming language: The following attribute has been specified within a pragma in the relevant declaration block: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line has been pasted into the declaration table and the following check box is activated: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Behavior during STOP-RUN transition

The values of non-retentive global variables are retained by default during the transition from STOP to RUN mode.

You can, however, make a setting whereby the non-retentive global variables are initialized during the STOP-RUN transition:

- **As of Version V4.2 of the SIMOTION Kernel**, by activating the (Page 4929) **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
  With non-retentive unit variables, this setting can be overwritten by a pragma or pragma line in the relevant data blocks of the program sources.
- **As of Version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the relevant data blocks of the program sources (only with non-retentive unit variables):
  - With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the relevant VAR_GLOBAL/END_VAR declaration block: { BlockInit_OnDeviceRun := ALWAYS; }
  - With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    Paste a pragma line with the following setting into the declaration table: "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block which is initialized during the STOP-RUN transition.

Note

With SIMOTION devices up to SIMOTION Kernel Version V4.0, non-retentive global variables are never initialized during the STOP-RUN transition.

Initialization of local variables

Local variables are initialized:

- For the initialization of retentive unit variables (Page 4849).
- For the initialization of non-retentive unit variables (Page 4851).
- Also, according to the following description:

Table 7-451 Initialization of local variables

Variable type	Time of the variable initialization
Local program variables	Local variables of programs are initialized differently: • Static variables (VAR) are initialized according to the memory area in which they are stored. See: Initialization of static program variables (Page 4854). • Temporary variables (VAR_TEMP) are initialized every time the program of the task is called.
Local variables of function blocks (FB)	Local variables of function blocks are initialized differently: • Static variables (VAR, VAR_IN, VAR_OUT) are only initialized when the FB instance is initialized. See: Initialization of instances of function blocks (FBs) (Page 4856). • Temporary variables (VAR_TEMP) are initialized every time the FB instance is called.
Local variables of functions (FC)	Local variables of functions are temporary and are initialized every time the function is called.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4914) function.

Initialization of static program variables

The following versions affect the following static variables:

- Local variables of a unit program declared with VAR
- Function block instances declared with VAR within a unit program, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT).

The initialization behavior is determined by the memory area in which the static variables are stored. This is determined by the "Only create program instance data once" (Page 4623) compiler option.

- For the deactivated "Only create program instance data once" compiler option (default):
  The static variables are stored in the user memory of each task which is assigned to the program.
  The initialization of the variables thus depends on the execution behavior of the task to which the program is assigned (see SIMOTION Basic Functions Function Manual):
  - Sequential tasks (MotionTasks, UserInterruptTasks, SystemInterruptTasks, StartupTask, ShutdownTask): The static variables are initialized every time the task is started.
  - Cyclic tasks (BackgroundTask, SynchronousTasks, TimerInterruptTasks): The static variables are only initialized only during the transition from STOP to RUN operating state.
- For the activated "Only create program instance data once" compiler option:
  This setting is necessary, for example, if a program is to be called within a program.
  The static variables of all programs from the program source (unit) involved are only stored once in the user memory of the unit.
  They are thus initialized together with the non-retentive unit variables, see Initialization of non-retentive global variables (Page 4851).
  They are not initialized by default during the transition from STOP to RUN operating state.
  You can, however, make a setting whereby they are initialized during the STOP-RUN transition:
  - **As of version V4.2 of the SIMOTION Kernel**, by activating the (Page 4929) **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
    This setting can be overwritten by a pragma or pragma line in the data block of the relevant program organization unit (POU).
  - **As of version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the data block of the relevant program organization unit (POU):
    With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the VAR/END_VAR declaration block:
    { BlockInit_OnDeviceRun := ALWAYS; }
    With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    The declaration table starts with a pragma line containing the following setting:
    "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR in the table are initialized during the STOP-RUN transition.

Initialization of retentive local variables of programs

The following versions relate to the following retentive variables:

- Local variables of a unit program declared with VAR RETAIN,
- Function block or class instances declared with VAR RETAIN within a unit program, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT).
- Function block instances declared with VAR within a unit program, or classes for their associated retentive variables (VAR RETAIN).

These retentive variables are initialized:

- When retentive unit variables are initialized (Page 4849).
- And also when a download is performed according to the following description.

The initialization behavior when downloading is determined by the compiler option (Page 4623) "Only create program instance data once".

- For the deactivated "Only create program instance data once" compiler option (default):
  The retentive variables are initialized in the following cases:
  - The data structure of the VAR RETAIN declaration blocks of the instances involved changes.
  - Additional tasks are activated in the execution system.
  - The assignment or sequence of the programs with retentive variables changes within a task.
- For the activated "Only create program instance data once" compiler option:
  The retentive variables are initialized in the following cases:
  - The data structure of the VAR RETAIN declaration blocks of the instances involved changes.
  - The sequence of the VAR_GLOBAL data blocks or of the programs (PROGRAM) changes in the implementation section of the source.

Initialization of instances of function blocks (FBs) or classes

The initialization of a function block instance (Page 4754) or a class instance (as of version V4.5 of the SIMOTION Kernel) (Page 4778) is determined by the location of its declaration:

- Global declaration (within VAR_GLOBAL/END_VAR in the interface of implementation section):
  Initialization as for a non-retentive unit variable, see Initialization of non-retentive global variables (Page 4851).
- Local declaration in a program (within VAR / END_VAR):
  Initialization as for static variables of programs, see Initialization of static variables of programs (Page 4854).
- Local declaration in a function block (within VAR / END_VAR):
  Initialization as for an instance of this function block.
- Declaration as in/out parameter in a function block or a function (within VAR_IN_OUT / END_VAR):
  For the initialization of the POU, only the reference (pointer) will be initialized with the instance of the function block remaining unchanged.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4914) function.

Initialization of retentive local variables of function blocks (FBs) and classes

The initialization of retentive variables of a function block (FB) or a class (as of version V4.5 of the SIMOTION Kernel) is determined by the location at which the instance of a function block (Page 4754) or the instance of a class (Page 4778) is declared:

- Global declaration (within VAR_GLOBAL/END_VAR in the interface of implementation section):
  The retentive variables are initialized:
  - When retentive unit variables are initialized (Page 4849).
  - In addition to the download:
    The data structure of the VAR RETAIN declaration blocks of the instances involved changes.
    The sequence of the VAR_GLOBAL declaration blocks in the current source changes.
- Local declaration in a program (within VAR / END_VAR):
  Initialization as for retentive local variables of programs, see Initialization of retentive local variables of programs (Page 4855).
- Local declaration in a function block (within VAR / END_VAR):
  Initialization as with the retentive variables of an instance of this function block.
- Declaration as in/out parameter in a function block or a function (within VAR_IN_OUT / END_VAR):
  For the initialization of the POU, only the reference (pointer) will be initialized with the instance of the function block remaining unchanged.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 4914) function.

Initialization of system variables of technology objects

The system variables of a technology object are usually not retentive. Depending on the technology object, a few system variables are stored in the retentive memory area (e.g. absolute encoder calibration).

The initialization behavior (except in the case of download) is the same as for retentive and non-retentive global variables. See Initialization of retentive global variables (Page 4849) and Initialization of non-retentive global variables (Page 4851).

The behavior during the download is shown below for:

- Non-retentive system variables
- Retentive system variables

Table 7-452 Initializing technology object system variables during download

Variable type	Time of the variable initialization
Non-retentive system variables	Behavior during download, depending on the *Initialization of all non-retentive data for technology objects* setting[1]: • **Yes**[2]: All technology objects are initialized. – All technology objects are restructured and all non-retentive system variables are initialized. – All technological alarms are cleared. • **No**[3]: Only technology objects changed in SIMOTION SCOUT are initialized. – The technology objects in question are restructured and all non-retentive system variables are initialized. – All alarms that are pending on the relevant technology objects are cleared. – If an alarm that can only be acknowledged with *Power On* is pending on a technology object that will not be initialized, the download is aborted.
Retentive system variables	Only if a technology object was changed in SIMOTION SCOUT, will its retentive system variables be initialized. The retentive system variables of all other technology objects are retained (e.g. absolute encoder calibration).
[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download. [2] The corresponding checkbox is active. [3] The corresponding checkbox is inactive.	

Version ID of global variables and their initialization during download

Table 7-453 Version code of global variables and their initialization during download

Data segment		Description of version code
Global device variables		
	Retentive global device variables Non-retentive global device variables	• Separate version code for each data segment of the global device variables. • The version code of the data segment changes for: – Add or remove a variable within the data segment – Change of the identifier or the data type of a variable within the data segment • This version code does not change on: – Changes in the other data segment – Changes to initialization values[1] • During downloading[2], the rule is: This data segment is only initialized when the version code of a data segment is changed.
Unit variables of a unit		

Data segment		Description of version code
	Retentive unit variables in the interface section Retentive unit variables in the implementation section Non-retentive unit variables in the interface section Non-retentive unit variables in the implementation section	• Several data blocks (= declaration blocks)[3] are possible in each data segment. • **Separate version code** for each data block. • The version code of the data block changes for: – Add or remove a variable in the associated declaration block – Change of variable sequence in the relevant declaration block – Change of the identifier or the data type of a variable in the associated declaration block – Change of a data type definition (from a separate or imported[4] unit) used in the associated declaration block – Add or remove declaration blocks within the same data segment **before** the associated declaration block • This version code does not change on: – Add or remove declaration blocks in other data segments – Add or remove declaration blocks within the same data segment **after** the associated declaration block – Changes in other data blocks – Changes to initialization values[1] – Changes to data type definitions that are not used in the associated data block – Changes to functions • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
Retentive variables of function blocks and classes (The instances are declared as non-retentive unit variables)		

Data segment		Description of version code
	Non-retentive unit variables in the interface section	Only as of version V4.5 of the SIMOTION Kernel. • The retentive variables of all instances within a declaration block (VAR_GLOBAL / END_VAR) are summarized as a separate retentive data block • to which a separate version code is assigned. • The version code of this retentive data block changes: – The data structure of the retentive local variables for the instances within the declaration block changes. – The sequence of the declaration blocks (VAR_GLOBAL / END_VAR) within the program source changes. • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
	Non-retentive unit variables in the implementation section	

[1] Changed initialization values are not effective until the data block or data segment in question is initialized.

[2] If *Initialization of retentive program data and retentive global device variables* = No and *Initialization of non-retentive program data and non-retentive global device variables* = No.
In the case of other settings: See the sections "Initialization of retentive global variables (Page 4849)" and "Initialization of non-retentive global variables (Page 4851)".

[3] With the **SIMOTION ST** programming language:
A data block corresponds to a VAR_GLOBAL/END_VAR or VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL or VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines within a section of the declaration table separate the variables into different data blocks.

[4] The use of units depends on the programming language, refer to the relevant section (Page 4830).

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided that the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute (Page 4917) has been specified within a pragma (Page 4922): { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Declaring variables in the Variable declaration dialog box ("on-the-fly" variable declaration)

You can declare a new variable in an ST source file without placing the cursor in the desired declaration block. For example, the cursor can remain at the code position in the body of a POU at which you want to use the new variable. You declare the variable "on-the fly" in the **Variable Declaration** dialog box.

Procedure

To define variables "on-the-fly" in the **Variable Declaration** dialog box, proceed as follows:

1. Place the cursor at the place of use of the new variable.
2. Optionally, enter the name of the new variable.

3. Select the **Variable declaration** context menu.
   The **Variable Declaration** dialog box appears.
   If necessary, the **Name** field is preassigned with the name of the new variable.

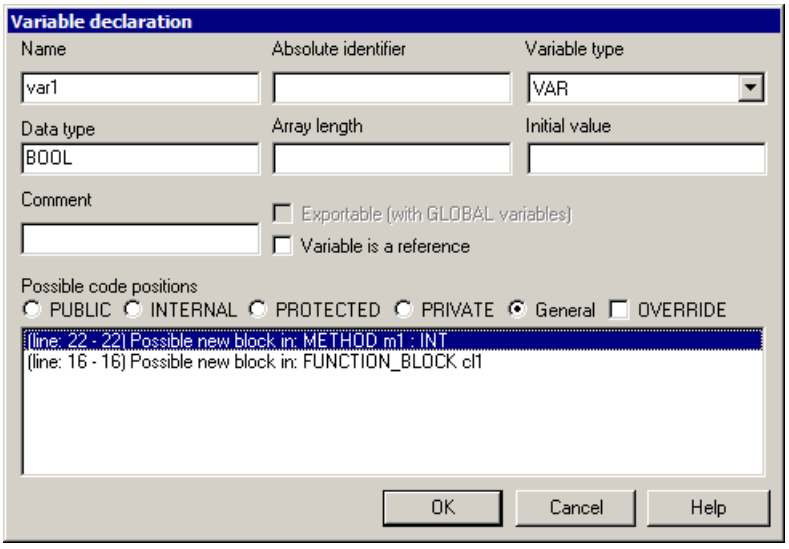


Figure 7-299 Variable Declaration dialog box

4. Make the other entries in the dialog box according to the following table.
5. Set the desired filter criteria according to the following table and select the desired code position.
6. Confirm with **OK**.

Principle

Table 7-454 Variable declaration parameter description

Field	Description
Name	Enter the name of the variable to be declared. The name must comply with the Rules for identifiers (Page 4656). The field is preassigned with the word in which the cursor is positioned when the window is called.
Absolute identifier	You define the variable as access to the fixed process image of the BackgroundTask (Page 4881) by entering an absolute identifier. The field is only active when a data type is selected that is contained in the general data types (Page 4674) ANY_BIT or ANY_INT. Enter the absolute identifier according to the syntax (Page 4888).
Variable type	Select the variable type. Available for selection are all permitted keywords for variable types (Page 4695). Default is VAR when the cursor is within a POU when the window is called, otherwise VAR_GLOBAL.
Data type	Enter the data type. You are supported by the "Automatic completion" (Page 4609) function (CTRL+space).

Field		Description
Array length		Not available when an absolute identifier is entered or the variable is a reference. You can define the variable as an array [0..N-1] by entering an array length N. You have the following options for entering the array length: • You can specify a constant positive integer value. • You can enter a value range with ".." separating the min. and max. values. • You can enter a constant expression of data type DINT (or of a data type which is implicitly convertible to DINT). If the array is empty, a single variable is created rather than an array.
Initial value		Not available when an absolute identifier is entered. You can specify this initialization value as a constant or as an expression. The following are permissible: • Constants • Arithmetic operations • Bit slice and data conversion functions
Comment		A comment can be entered in this column. All characters and special characters are permitted.
Exportable (for GLOBAL variables)		Only available when the keyword for a unit variable or unit constant has been selected as variable type (e.g. VAR_GLOBAL). If the checkbox is activated: • The variable declaration is inserted in the interface section of the ST source file. If the checkbox is deactivated: • The variable declaration is inserted in the implementation section of the ST source file.
Variable is a reference		Only available when a valid data type has been entered. With activated checkbox, a general reference (Page 4800) is created for the entered data type. The keyword REF_TO is set as prefix for the specified data type. The checkbox is activated automatically when the entered data type starts with the keyword REF_TO.
Possible code positions		
	Filter criteria	You can select the following as filter criteria: • An access identifier within classes, function blocks or namespaces – **PUBLIC** – **INTERNAL** – **PROTECTED** – **PRIVATE** – **General** (default, not an access identifier) • The keyword **OVERRIDE**.
	List of the code positions	A list of the possible code positions is displayed at which the variable can be declared. A distinction is made between: • Insertion in an existing variable block • Creation of a new variable block The first entry in the list is selected. In the editor window, the code position is displayed and selected at which the variable or the new variable block will be inserted. Select from the

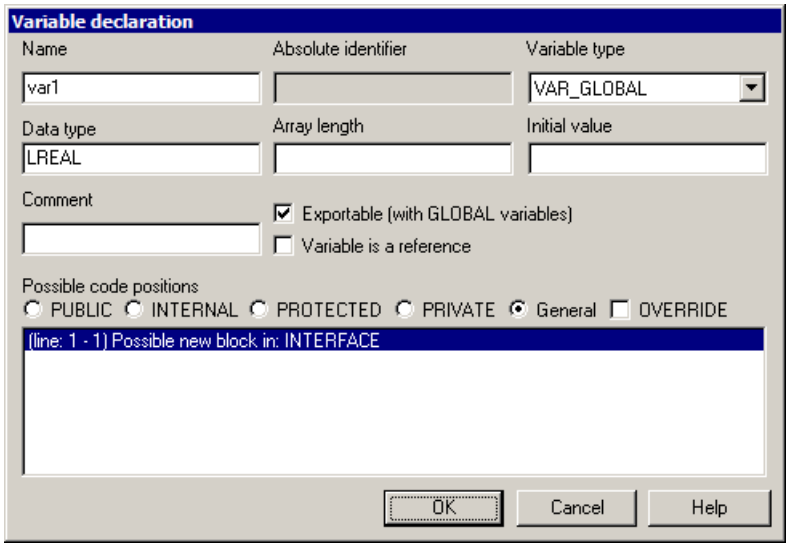


Figure 7-300 Example: Variable declaration

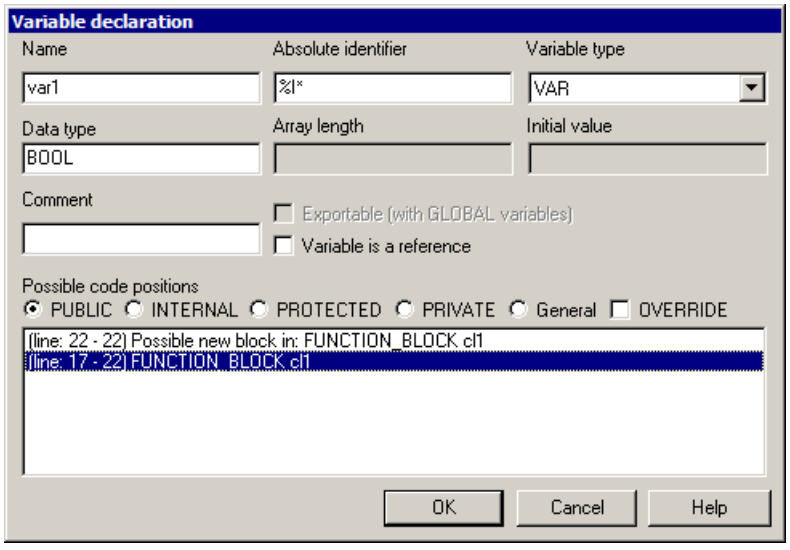


Figure 7-301 Example: Variable declaration (absolute name)

Result

The variable or the new variable block is inserted at the selected code position in the ST source file.

The cursor is at the same position before the dialog box was called.

Variables and HMI devices

Exported variables

The following variables are exported to HMI devices where they are available:

- System variables of the SIMOTION device
- System variables of technology objects
- I/O variables
- Global device variables
- Retentive and non-retentive unit variables of the interface section (default setting).
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes (Page 4922).
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line, if you deactivate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
- Non-retentive static variables (VAR) of programs provided that the compiler option (Page 4623) "Only create program instance data once" is activated
- Non-retentive static variables (VAR) of function blocks
  This is the default setting when the compiler option (Page 4623) "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes (Page 4922).

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Non-retentive public variables (VAR PUBLIC) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes (Page 4922).

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.

Note

The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.

The effect of the `{ HMI_Export := FALSE; } / { HMI_Export := TRUE; }` pragma (or the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* settings in a pragma line) depends on the SIMOTION Kernel version:

- As of Version V4.1 of the SIMOTION Kernel:
  The pragma affects the export of the corresponding declaration block to HMI devices **and** the structure of the HMI address space:
  - Only those variables in declaration blocks exported to HMI devices occupy the HMI address space.
  - Within the HMI address space, the variables are arranged according to order of their declaration.
- Up to Version V4.0 of the SIMOTION Kernel:
  The pragma affects **only** the export of the corresponding declaration block to HMI devices. The HMI address space is also occupied by unit variables of the interface section whose declaration blocks are not assigned to HMI devices.
  Within the HMI address space, the variables are sorted in the following order:
  - Retentive unit variables of the interface section (exported and not exported).
  - Retentive unit variables of the implementation section (only exported).
  - Non-retentive unit variables of the interface section (exported and not exported).
  - Non-retentive unit variables of the implementation section (only exported).

  Within these segments, the variables are arranged according to order of their declaration.

Non-exported variables

The following variables are **not** exported to HMI devices and are **not** available there:

- Retentive and non-retentive unit variables of the implementation section (default setting)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`
    See also Controlling compiler with attributes (Page 4922).
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line, if you activate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Local variables (VAR, VAR_TEMP) of functions or methods
- Temporary variables (VAR_TEMP) of programs, function blocks or classes
- Retentive local variables (VAR RETAIN) of programs (as of version 4.5 of the SIMOTION Kernel)

- Retentive local variables (VAR RETAIN) of function blocks (as of version 4.5 of the SIMOTION Kernel)
  This is the default setting when the compiler option (Page 4623) "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes (Page 4922).

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Retentive local variables (VAR RETAIN) of classes (default setting, as of version 4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes (Page 4922).

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Non-retentive static variables (VAR) of programs provided that the compiler option (Page 4623) "Only create program instance data once" is **not** activated
- Non-retentive protected or private variables (VAR, VAR PROTECTED, VAR PRIVATE) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes (Page 4922).

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.

Example for the SIMOTION ST programming language

Table 7-455 Example for the control of the HMI export with the corresponding pragma

```
INTERFACE
    VAR_GLOBAL
        // HMI export
        x1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := FALSE; }
        // No HMI export
        x2 : DINT;
    END_VAR
    // ...
END_INTERFACE

IMPLEMENTATION
    VAR_GLOBAL
        // No HMI export
        y1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := TRUE; }
        // HMI export
        y2 : DINT;
    END_VAR
    // ...
END_IMPLEMENTATION
```

7.2.7.3 Access to inputs and outputs (process image, I/O variables)

Overview of access to inputs and outputs

SIMOTION provides several possibilities to access the device inputs and outputs of the SIMOTION device as well as the central and distributed I/O:

- Via direct access with I/O variables
  Direct access is used to access the corresponding I/O address directly.
  Define an I/O variable (name and I/O address) without assigning a task to it. The entire address space of the SIMOTION device can be used.
  It is preferable to use direct access with sequential programming (in MotionTasks); access to current input and output values at a particular point in time is especially important in this case.
  Further information: Direct access and process image of the cyclic tasks (Page 4872).
- Via the process image of cyclic tasks using I/O variables
  The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming).
  Define an I/O variable (name and I/O address) and assign a task to it. The entire address range of the SIMOTION device can be used.
  Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.
  Further information: Direct access and process image of the cyclic tasks (Page 4872).
- Using the fixed process image of the BackgroundTask
  The process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device, on which a subset of the I/O address space of the SIMOTION device is mirrored. The mirror image is refreshed with the BackgroundTask and is consistent throughout the entire cycle. This process image is used preferentially when programming the BackgroundTask (cyclic programming).
  The address range 0 .. 63 can be used. Exception: I/O addresses that are accessed using the process image of the cyclic task can only be used with the setting “Common process image” (Page 4883) (as of Kernel V4.2).
  Further information: Access to the fixed process image of the BackgroundTask (Page 4881).

A comparison of the most important properties is contained in "Important properties of direct access and process image" (Page 4869).

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4892).

Note

An access via the process image is more efficient than direct access.

Important features of direct access and process image access

Table 7-456 Important properties of direct access and process image access

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Permissible address range	Entire address range of the SIMOTION device **Exception**: I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously (example: PIW63 or PQD62 are not permitted).		Addresses 0 .. 63. **Exception**: Up to version V4.1 of the SIMOTION Kernel or the "Separate process image" setting, addresses used for the process image of the cyclic tasks are not permitted.
Address configuration	Necessary. The addresses used must be present in the I/O and appropriately configured. The "Rules for I/O addresses for direct access and the process image of the cyclic tasks" (Page 4875) must be observed.		Not necessary. Addresses that are not present in the I/O or have not been configured can also be used.
Assigned task	None.	Cyclic task for selection: • SynchronousTasks, • TimerInterruptTasks, • BackgroundTask.	BackgroundTask.
Memory area for process images	-	Depends on the SIMOTON Kernel version: • Up to version V4.1: Separate memory areas in all cases • As of version V4.2: Option to select a common memory area	

	Direct access	**Access to process image of cyclic tasks**	**Access to fixed process image of the BackgroundTask**
Update	• Onboard I/O of SIMOTION devices C230-2, C240, and C240 PN: Update occurs in a cycle clock of 125 µs. • With I/O devices – via PROFIBUS DP or PROFINET on an isochronous SIMOTION device[1] – via DRIVE-CLiQ – Onboard I/O of SIMOTION D devices: The update is performed in the position control cycle clock[2]. • With I/O devices – via PROFIBUS DP or PROFINET on a **non**-isochronous SIMOTION device[1] – via P-Bus: The update is performed in the interpolation cycle[2 3]. Inputs are read at the start of the cycle clock. Outputs are written at the end of the cycle clock.	Update occurs with the assigned task: • Inputs are read before the assigned task is started and transferred to the process input image. • Process output image is written to the outputs after the assigned task has been completed.	An update is made with the *BackgroundTask*: • Inputs are read before the *BackgroundTask* is started and is transferred to the process input image. • Process output image is written to the outputs when the *BackgroundTask* is complete.
Consistency	–	During the entire cycle of the assigned task. **Exception**: Direct access to output occurs.	During the entire cycle of the *BackgroundTask*. **Exception**: Direct access to output occurs.
	Consistency is only ensured for elementary data types. When using arrays, the user is responsible for ensuring data consistency.		
Use	Preferred in MotionTasks	Preferred in the assigned task	Preferred in the BackgroundTask
Declaration as variable	Necessary, for the entire device as an I/O variable in the symbol browser. Each byte of the address range may only be assigned to a single I/O variable. Syntax of I/O address: e.g. PIW1022, PQ63.3.		Possible, but not necessary: • For the entire device as I/O variable in the symbol browser • As unit variable • As local static variable in a program
Download of new or changed I/O variables	Only possible in STOP mode.		-
Use the absolute address	Not supported.		Possible, with the following syntax: E.g. %IW62, %Q63.3.

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Byte order when forming the process image	-	As supplied by the I/O	Depends on the SIMOTION Kernel version and the memory area setting for the process images: • Up to version V4.1 or the "Separate process image" setting: Always Big Endian • As of version V4.2 and the "Common process image" setting: As supplied by the I/O
Byte order during access	Depends on I/O		Always Big Endian
Writeability of inputs	No	Depends on the SIMOTION Kernel version: • Up to version V4.1: No • As of version V4.2: Yes	Yes
Write protection for outputs	Possible; **Read only** status can be selected.	Not supported.	Not supported.
Declaration of arrays	Possible.		Not supported.
Further information	Direct access and process image of the cyclic tasks (Page 4872).		Access to the fixed process image of the BackgroundTask (Page 4881).
Responses in the event of an error	Error during access from user program, alternative reactions available: • CPU Stop[4] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, alternative reactions available: • CPU stop[5] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, reaction: CPU stop[5] **Exception**: If a direct access has been created at the same address, the behavior set there applies.
Access			
• In the RUN operating state	Without any restrictions.	Without any restrictions.	Without any restrictions.
• During the StartupTask	Possible with restrictions: • Inputs can be read. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
• During the ShutdownTask	Without any restrictions.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.

[1] A SIMOTION device is considered isochronous if at least one PROFIBUS DP or PROFINET interface is operated isochronously (constant bus cycle time). For SIMOTION D there is an exception with the DP Integrated interface to which the SINAMICS Integrated is connected.

[2]The following SIMOTION devices are updated in the Servo_fast cycle or IPO_fast cycle, if the cycles are configured: D445-2 DP/PN, D455-2 DP/PN (as of version V4.2) and D435-2 DP/PN (as of version V4.3).

3 IPO or IPO_2 adjustable, see “Setting system cycle clocks" section in the Basic Functions Function Manual.

[4] Call the ExecutionFaultTask.

[5] Call the PeripheralFaultTask.

Direct access and process image of cyclic tasks

Property

Direct access to inputs and outputs and access to the process image of the cyclic task always take place via I/O variables. The entire address range of the SIMOTION device (Page 4874) can be used.

A comparison of the most important properties, including in comparison to the fixed process image of the BackgroundTask (Page 4881) is contained in "Important properties of direct access and process image" (Page 4869).

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4875).

It is particularly important that every address used in an I/O variable is available in the I/O and configured; each byte in the address range may be assigned to no more than one I/O variable (does not apply to access with data type BOOL).

A detailed status of I/O variables (Page 4879) can be read as of version V4.2 of the SIMOTION Kernel, for example, in order to check the availability of the I/O variables.

Direct access

Direct access is used to access the corresponding I/O address directly. Direct access is used primarily for sequential programming (in MotionTasks). The access to the current value of the inputs and outputs at a specific time is particularly important.

For direct access, you define an I/O variable (Page 4876) without assigning it a task.

Process image of the cyclic task

The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming). The consistency during the complete cycle of the task is particularly important.

For the process image of the cyclical task you define an I/O variable (Page 4876) and assign it a task.

Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.

Note

An access via the process image is more efficient than direct access.

Additional properties as of version V4.2 of the SIMOTION Kernel

As of version V4.2 of the SIMOTION Kernel, direct access to inputs/outputs and the process image of the cyclic tasks offers additional properties:

- As far as the process image of the cyclic tasks is concerned, a common memory area with the fixed process image of the BackgroundTask can be set (standard with newly created devices)
- As far as the process image of the cyclic tasks is concerned, I/O variables for inputs can be written to (i.e. they can be assigned values).
- A detailed status of I/O variables (Page 4879) can be read, for example, in order to check the availability of the I/O variables.

Memory area with the fixed process image of the BackgroundTask

- **As of version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device (Page 4929) ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks.
- **Up to version V4.1 of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas.

Note

If (and only if) you are also using the fixed process image of the BackgroundTask, it is important to consider the effects of the "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask (Page 4881).

Table 7-457 Effect of "Common process image" or "Separate process image" settings on the process image of the cyclic tasks

	Common process image	Separate process image
Availability	Only available as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Up to version V4.1 of the SIMOTION Kernel applies: • System characteristic, not configurable. The following applies as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Download of new or changed I/O variables	Only possible in STOP mode.	Only possible in STOP mode.
Byte order when forming the process image and during access	Depends on connected I/O	
Effects on the fixed process image of the BackgroundTask	See the relevant table in "Access to the fixed process image of the BackgroundTask" (Page 4882).	
Further information	Common process image (Page 4883)	Separate process image (Page 4885)

Address range of the SIMOTION devices

The address range of the SIMOTION devices is specified in the following table according to the version of the SIMOTION Kernel concerned. The complete address range can be used for direct access and process image of the cyclical tasks.

Table 7-458 Address range of the SIMOTION devices according to the version of the SIMOTION Kernel

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
C230-2	0 .. 2047 [3]	0 .. 2047 [3]	0 .. 2047 [3]	–	–	–
C240	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]
C240 PN [1]	–	–	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]
D410 DP	–	–	0 .. 8191 [3]	0 .. 8191 [3]	0 .. 8191 [3]	–
D410 PN	–	–	0 .. 8191 [4]	0 .. 8191 [4]	0 .. 8191 [4]	–
D410-2	–	–	–	–	0 .. 8191 [3 4]	0 .. 8191 [3 4]
D425	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D425-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D435	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D435-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D445	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–	–
D445-1 [1]	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D445-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D455-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
P320 [2]	–	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
P320-4	–	–	–	–	–	0 .. 4095 [3]
P350	0 .. 2047 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

[1] Available as of V4.1 SP2 HF4

[2] Available as of V4.1 SP5

[3] For distributed I/O (over PROFIBUS DP), the transmission volume is restricted to 1024 bytes per PROFIBUS DP line.

[4] For distributed I/O (over PROFINET), the transmission volume is restricted to 4,096 bytes per PROFINET segment.

Rules for I/O addresses for direct access and the process image of the cyclical tasks

Note

You must observe the following rules for the I/O variable addresses for direct access and the process image of the cyclic task (Page 4872). Compliance with the rules is checked during the consistency check of the SIMOTION project (e.g. during the download).

1. Addresses used for I/O variables must be present in the I/O and configured appropriately in the HW Config.
2. I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously.
   The following I/O addresses are not permitted:
   - Inputs: PIW63, PID61, PID62, PID63
   - Outputs: PQW63, PQD61, PQD62, PQD63
3. All addresses of an I/O variable comprising more than one byte (e.g. WORD, ARRAY data type) must be within a continuous address range configured in HW Config, e.g. within the address range (slot or subslot) of *one* I/O module.
4. An I/O address (input or output) can only be used by a single I/O variable of data type BYTE, WORD or DWORD or an array of these data types. Access to individual bits with I/O variables of data type BOOL is possible.
5. If several processes (e.g. I/O variable, technology object, PROFIdrive telegram) access an I/O address, the following applies:
   - Only a single process can have write access to an I/O address of an output (BYTE, WORD or DWORD data type).
     Read access to an output with an I/O variable that is used by another process for write access, is possible.
   - All processes must use the same data type (BYTE, WORD, DWORD or ARRAY of these data types) to access this I/O address. Access to individual bits is possible irrespective of this. Please be aware of the following, for example, if you wish to use an I/O variable to read the PROFIdrive telegram transferred to or from the drive: The length of the I/O variable must match the length of the telegram.
   - Write access to different bits of an address is possible from several processes; however, write access with the data types BYTE, WORD or DWORD is then not possible.

Note

These rules do not apply to accesses to the fixed process image of the BackgroundTask (Page 4881). These accesses are not taken into account during the consistency check of the project (e.g. during download).

Creating I/O variables for direct access or process image of cyclic tasks

Create I/O variables for direct access or a process image of the cyclic tasks in the address list of the detail view.

This is only possible in offline mode.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select **Insert new line**
   or
   Scroll to the end of the table of variables (empty line).
3. In the empty row of the table, enter or select the following:
   - Names of the **I/O variables**
   - **I/O address**
     Select the "IN" or "OUT" entries if you wish to assign symbols to the I/O variable (input or output). As of Version V4.2 of the SIMOTION Kernel the symbolic assignment must be activated, menu **Project > Use symbolic assignment**.
     Or enter a fixed address according to "Syntax for entering I/O addresses" (Page 4878).
   - Optional for outputs:
     Activate the **Read only** checkbox if you only want to have read access to the output. You can then read an output that is already being written by another process (e.g. output of an output cam, PROFIdrive telegram).
     A read-only output variable cannot be assigned to the process image of a cyclic task.
   - **Data type** of the variables in accordance with "Possible data types of the I/O variables" (Page 4879).

4. Optionally, you can also enter or select the following (not for data type BOOL):
   - **Array length** (array size).
   - **Process image** or direct access:
     Can only be assigned if the **Read only** checkbox is deactivated.
     For process image, select the cyclic task to which you want to assign the I/O variable. To select a task, it must have been activated in the execution system.
     For direct access, select the blank entry.
   - **Strategy** for behavior in the event of an error, see SIMOTION Basic Functions Function Manual.
   - **Display format** (if array, for each element), when you monitor the variable in the address list
   - **Substitute value** (if array, for each element).
5. Only if you have selected "IN" or "OUT" as the I/O address (symbolic assignment).
   - In the **Assignment** column, click the **[...]** button.
     A window opens displaying the possible assignment targets of the SIMOTION device and, if necessary, of SINAMICS Integrated. Only those assignment targets are displayed that match the data direction (input/output) and data type.
   - Select the assignment target.
     The **Assignment status** column indicates whether the assignment was successful or not.

   For details regarding symbolic assignment, refer to the SIMOTION Basic Functions Function Manual.

You can now access this variable using the address list or any program of the SIMOTION device.

Details on how to manage the address list can be found in the online help.

Note

Note the following for the process image for cyclic tasks:

- A variable can only be assigned to one task.
- Each byte of an input or output can only be assigned to one I/O variable.

In the case of data type BOOL, please note:

- The process image for cyclic tasks and a strategy for errors cannot be defined. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access or CPU stop).
- The individual bits of an I/O variable can also be accessed using the bit access functions.

Take care when making changes within the I/O variables (e.g. inserting and deleting I/O variables, changing names and addresses):

- In some cases the internal addressing of other I/O variables may change, making all I/O variables inconsistent.
- If this happens, all program sources that contain accesses to I/O variables must be recompiled.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and then use them in your program sources (e.g. ST sources, MCC sources, LAD/FBD sources).

Outputs can be read and written to, but inputs can only be read.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4892).

Syntax for entering I/O addresses

Syntax

For the input of the I/O address for the definition of an I/O variable for direct access or process image of cyclical tasks (Page 4872), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

Table 7-459 Syntax for the input of the I/O addresses for direct access or process image of the cyclic tasks

Data type	Syntax for		Permissible address range					
	Input	Output	Direct access		Process image		e.g. direct access D435 V4.1	
BOOL	PIn.x	PQn.x	n: x:	0 .. *MaxAddr* 0 .. 7		-[1]	n: x:	0 .. 16383 0 .. 7
BYTE	PIBn	PQBn	n:	0 .. *MaxAddr*	n:	0 .. *MaxAddr*	n:	0 .. 16383
WORD	PIWn	PQWn	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. 16382
DWORD	PIDn	PQDn	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. 16380
n = logical address x = bit number								
MaxAddr =	Maximum I/O address of the SIMOTION device depending on the SIMOTION Kernel version, see Address range of the SIMOTION devices (Page 4874).							
[1] For data type BOOL, it is not possible to define the process image for cyclic tasks. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access).								

Examples

Input at logic address 1022, WORD data type: **PIW1022**.

Output at logical address 63, bit 3, BOOL data type: **PQ63.3**.

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4875).

Possible data types of I/O variables

The following data types can be assigned to the I/O variables for direct access and process image of the cyclical tasks (Page 4872). The width of the data type must correspond to the data type width of the I/O address.

If you assign a numeric data type to the I/O variables, you can access these variables as integer.

Table 7-460 Possible data types of the I/O variables for direct access and the process image of the cyclical tasks

Data type of I/O address	Possible data types for I/O variables
BOOL (PIn.x, PQn.x)	BOOL
BYTE (PIBn, PQBn)	BYTE, SINT, USINT
WORD (PIWn, PQWn)	WORD, INT, UINT
DWORD (PIDn, PQDn)	DWORD, DINT, UDINT

For details of the data type of the I/O address, see also "Syntax for entering I/O addresses" (Page 4878).

Detailed status of the I/O variables (as of Kernel V4.2)

As of version V4.2 of the SIMOTION Kernel, the status of an I/O variable can be queried using *_quality.var-name*, for example, in order to check the availability of the I/O variables. It is supplied as an OR logic operation of the following status values in the DWORD data type and can be assigned to an appropriate variable, for example. The value 16#0000_0000 indicates the connected I/O are operating without error.

The same value is supplied for every I/O variable within an address range (slot or subslot) configured in HW Config.

Table 7-461 Meanings of status values of I/O variables

Value (DWORD)	Bit x = 1	Meaning
16#0000_0000	-	No error occurred.
16#0000_0001	0	**Maintenance required** The connected module signals that it requires maintenance. The component needs to be checked within a foreseeable period (e.g. the printer cartridge must be changed within a period of several days).
16#0000_0002	1	**Maintenance demanded** The connected module demands maintenance. The component needs to be checked soon (e.g. the printer cartridge must be changed immediately).
16#0000_0004	2	**Warning pending** (drive warning, TM17 warning, etc.) The connected module has signaled a warning. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.
16#0000_0008	3	**Fault pending** (diagnostic interrupt, drive fault, TM17 fault, etc.) The connected module has signaled an error. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.

Value (DWORD)	Bit x = 1	Meaning
16#0000_0010	4	**This parameter assignment does not match the parameter assignment being compared.** A difference was detected when this parameter assignment was compared with the parameter assignment of the connected module. As such, the required functionality cannot be guaranteed. Remedy: Save the project and compile changes, reload both application and counterpart.
16#0000_0020	5	**Application and counterpart are not isochronous** (error involving the dynamic life-sign). Certain telegrams (axis, synchronous operation, output cam, measuring input telegrams) are synchronized by exchanging cyclic life-signs. Errors are detected when the cyclic life-sign is checked. This invalidates the data in the telegram. Remedy: Await synchronization, check the parameter assignment (e.g. does the master application cycle set on the device in HW Config match the position control cycle clock), save the project and compile changes, reload both application and counterpart.
16#0000_0040	6	**I/O cannot be used synchronously in all cycles.** A fast application cycle (Servo_fast) and a slow application cycle (Servo) are running asynchronously in relation to one another. The I/O can only be used synchronously in the cycles associated with the bus cycle. Access from other cycles is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0080	7	**I/O cannot be used synchronously** The SIMOTION control is the sync slave on a bus. The bus connection is running synchronously in relation to the sync master, but is not yet running synchronously in relation to the application cycles of the SIMOTION control. Access to the I/O is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0100	8	**Bus connection (sync slave) is not isochronous in relation to the sync master.** The SIMOTION control is the sync slave on a bus and has not yet synchronized its bus connection with the sync master. The isochronous I/O on this bus cannot be used yet. Remedy: Switch on/connect the sync master.
16#0000_0200	9	**DP station is deactivated.** The partner module has been deactivated. Remedy: Activate the partner module (_activateDpSlave() function).
16#0000_0400	10	**The partner of the inputs (e.g. I-device, I-slave) is in STOP.** The connected module is in STOP mode and not sending any new data as a result. Remedy: Switch the connected module to RUN.
16#0000_0800	11	**PROFINET: Failure detected by submodule** (e.g. channel error) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_1000	12	**PROFINET: Failure detected by module** (e.g. submodule failed, removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_2000	13	**PROFINET: Failure detected by device** (e.g. device in STOP, module removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config

Value (DWORD)	Bit x = 1	Meaning
16#0000_4000	14	**PROFINET: Failure detected by controller** (e.g. not connected, etc.) There is no connection to a partner on PROFINET. Possible cause: Partner is switched off, cable pulled out, incorrect parameter assignment for connection Troubleshooting: Best to use the PROFINET topology editor in HW Config.
16#0000_8000	15	**Slot/subslot is not connected** (disconnection alarm). The connection to the connected device is OK. The error must be searched for in the connected device (e.g. module/submodule removed). Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0001_0000	16	**Device is not connected** (station failure). There is no connection to a partner. Possible cause: Partner is switched off, cable pulled out.
16#0002_0000	17	**Substitute value behavior during access** There is no connection to the counterpart (sum signal from bits 9 to 16), i.e. there is no valid input data or the output data is not reaching the terminal. The substitute value behavior set (substitute value, last value) takes effect during direct access to this address or during process image updates.
16#4000_0000	30	**Diagnostics address only** No cyclic I/O data is configured for this address. It is possible, however, to query submodule diagnostic information.
16#8000_0000	31	**Address gap** There is no hardware configured for this logical address.

Access to fixed process image of the BackgroundTask

The fixed process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device on which a subset of the I/O address space of the SIMOTION device is mirrored. Preferably, it should be used for programming the BackgroundTask (cyclic programming) as it is consistent throughout the entire cycle.

The size of the fixed process image of the BackgroundTask for all SIMOTION devices is 64 bytes (address range 0 .. 63).

Note

The fixed process image of the BackgroundTask can be used to access addresses that are not available in the I/O or not configured in HW Config. These are treated like normal memory addresses.

Memory area

- **As of Version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device (Page 4929) ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks. I/O addresses can be read and written to using both the fixed process image of the BackgroundTask and the process image of the cyclic tasks.
- **With Version V4.1 and lower of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of Version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas. I/O addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.

Table 7-462 Effect of "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask

	Common process image	Separate process image
Availability	Only available as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Version V4.1 and lower of the SIMOTION Kernel applies: • System characteristic, not configurable The following applies as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Memory area	Subset of the memory area for the process image of the cyclic tasks	Separate memory area for the process image of the cyclic tasks
Using I/O addresses accessed using the process image of the cyclic tasks	Possible. Updates use the configured cyclic tasks.	Not supported. The addresses are treated like normal memory addresses.
Byte order when forming the process image	As supplied by the I/O	Always Big Endian
Byte order when accessing the process image	Always Big Endian	Always Big Endian
Access to I/O operating in the Little Endian byte order	Same result as during direct access or for the process image of cyclic tasks (apart from WORD or DWORD data types).	Results differ depending on the I/O variables created for direct access.
Effects on the process image of the cyclic tasks	See the relevant table in "Direct access and process image of the cyclic tasks (Page 4874)".	
Further information	Common process image (Page 4883)	Separate process image (Page 4885)

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

A comparison of the most important properties in comparison to the direct access and process image of the cyclic tasks (Page 4872) is contained in "Important properties of direct access and process image" (Page 4869).

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4875) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

You can access the fixed process image of the BackgroundTask by means of:

- Using an absolute PI access (Page 4887): The absolute PI access identifier contains the address of the input/output and the data type.
- Using a symbolic PI access (Page 4889): You declare a variable that references the relevant absolute PI access:
  - A unit variable
  - A static local variable in a program.
- Using an I/O variable (Page 4891): In the symbol browser, you define a valid I/O variable for the entire device that references the corresponding absolute PI access.

Common process image (as of Kernel V4.2)

As of Version V4.2 of the SIMOTION Kernel, the "Common process image" setting (Page 4929) can be selected on the SIMOTION device. This means addresses 0 .. 63 of the process image of the cyclic tasks and the fixed process image of the BackgroundTask occupy the same memory area.

This is the default for SIMOTION devices newly created in the project as of Version V4.2.

Property of the common process image

1. The memory area for the fixed process image of the BackgroundTask (Page 4881) is a subset of the memory area for the process image of the cyclic tasks (Page 4872).
2. This means I/O addresses already accessed using the process image of the cyclic tasks may also continue to be used for the fixed process image of the BackgroundTask. Updates, however, use the configured cyclic tasks.
3. The following applies when forming the fixed process image of the BackgroundTask: The byte order is the same as supplied by the I/O:
   - Big Endian, e.g. for I/O via PROFIBUS DP, PROFINET, P-Bus, DRIVE-CLiQ
   - Little Endian, e.g. for onboard I/O of C240, C240 PN SIMOTION devices

   Any I/O variable created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks has no effect on the byte order.

4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C240, C240 PN SIMOTION devices).
   If the fixed process image of the BackgroundTask is used for access, this leads to the following behavior, regardless of whether I/O variables have been created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks:
   - Access to individual bytes always supplies the same result via an I/O variable or the fixed process image of the BackgroundTask.
   - With the fixed process image of the BackgroundTask, bytes only change places if data type WORD is used for access.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for common process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access or the process image of the cyclic tasks:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access or access to the process image of the cyclic tasks:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte function (direct access only)
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-463 "Common process image" setting (as of Kernel V4.2): Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access or access to the process image of the cyclic tasks	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	16#08
	%IB61	16#00	16#00	16#00
	%IW60	**16#0800** [2]	**16#0800** [2]	**16#0800** [2]
	%I60.3	TRUE	TRUE	TRUE
	%I61.3	FALSE	FALSE	FALSE

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access or the process image of the cyclic tasks; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian byte order is being read using Big Endian.

Separate process image (up to Kernel V4.1)

With Version V4.1 and below of the SIMOTION Kernel, the process image of the cyclic task and the fixed process image of the BackgroundTask are stored in different memory areas (separate process image).

As of Version V4.2 of the SIMOTION Kernel, the "Separate process image" setting (Page 4929) can be selected on the SIMOTION device. This setting ensures there is compatibility with earlier Kernel versions.

It is the default for SIMOTION devices upgraded to Version V4.2 or higher.

Property of the separate process image

1. The fixed process image of the BackgroundTask (Page 4881) and the process image of the cyclic tasks (Page 4872) are stored in different memory areas.
2. This means I/O addresses that are already accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.
3. I/O variables for direct access influence the fixed process image of the BackgroundTask:
   – The fixed process image of the BackgroundTask is always formed for the relevant addresses in the Big Endian byte order.
4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C230-2, C240, C240 PN SIMOTION devices). If an I/O variable is created for the relevant addresses for the purpose of direct access using data type WORD and access takes place using the fixed process image of the BackgroundTask, this leads to the following behavior:
   – Access with the data type WORD supplies the same result via the I/O variable and the fixed process image of the BackgroundTask.
   – Access to individual bytes using the _getInOutByte function (see SIMOTION Basic Functions Function Manual) supplies these in the Little Endian order.
   – Access to the individual bytes or bits with the fixed process image of the BackgroundTask supplies these in the Big Endian order.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for separate process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte (Page 4872) function
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-464 "Separate process image" setting or Kernel up to Version V4.1: Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	**16#00** [3]
	%IB61	16#00	16#00	**16#08** [3]
	%IW60	**16#0800** [2]	**16#0800** [2]	16#0008
	%I60.3	TRUE	TRUE	**FALSE** [3]
	%I61.3	FALSE	FALSE	**TRUE** [3]

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian order is being read using Big Endian.

[3] The two adjacent bytes change places, as the relevant word is saved in the Big Endian order.

Absolute access to the fixed process image of the BackgroundTask (absolute PI access)

You make absolute access to the fixed process image of the BackgroundTask (Page 4881) by directly using the identifier for the address (with implicit data type). The syntax of the identifier (Page 4888) is described in the following section.

You can use the identifier for the absolute PI access in the same manner as a normal variable (Page 4888).

Note

Outputs can be read and written to, but inputs can only be read.

Syntax for the identifier for an absolute process image access

For the absolute access to the fixed process image of the BackgroundTask (Page 4887), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

You also use these identifiers:

- For the declaration of a symbolic access to the fixed process image of the BackgroundTask (Page 4889).
- For the creation of an I/O variables for accessing the fixed process image of the BackgroundTask (Page 4891).

Table 7-465 Syntax for the identifier for an absolute process image access

Data type	Syntax for		Permissible address range	
	Input	Output		
BOOL	%In.x or %IXn.x [1]	%Qn.x or %QXn.x[1]	n: x:	0 .. 63 [2] 0 .. 7
BYTE	%IBn	%QBn	n:	0 .. 63 [2]
WORD	%IWn	%QWn	n:	0 .. 63 [2]
DWORD	%IDn	%QDn	n:	0 .. 63 [2]
n = logical address x = bit number				
[1] The syntax %IXn.x or %QXn.x is not permitted when defining I/O variables. [2] For a separate process image (Page 4885), the following applies: No addresses that are used in the process image of the cyclic tasks. See note below.				

Examples

Input at logic address 62, WORD data type: **%IW62**.

Output at logical address 63, bit 3, BOOL data type: **%Q63.3**.

Note

Up to Version V4.1 of the SIMOTION Kernel or the "Separate process image" (Page 4885) setting on the device (as of Version V4.2 of the SIMOTION Kernel), the following applies:

- Addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask.

This restriction no longer applies as of Version V4.2 of the SIMOTION Kernel or with the "Common process image" (Page 4883) setting on the device.

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 4875) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

Several examples for the assignment of variables of the same type follow:

Table 7-466 Examples of absolute CPU memory access

```
status1 := %I1.1; // BOOL data type
status2 := %IB10; // BYTE data type
status3 := %IW20; // WORD data type
status4 := %ID20; // DWORD data type

%Q1.1 := status1; // BOOL data type
%QB20 := status2; // BYTE data type
%QW20 := status3; // WORD data type
%QD20 := status4; // DWORD data type
```

Symbolic access to the fixed process image of the BackgroundTask (symbolic PI access)

You can access the fixed process image of the BackgroundTask (Page 4881) symbolically without needing to always specify the absolute process image access.

You can declare symbolic access:

- As a static variable of a program (within the VAR/END_VAR structure in the declaration section)
- As a unit variable (within the VAR_GLOBAL / END_VAR structure in the interface or implementation section of the ST source file)

The syntax for declaring a symbolic name for the PI access is shown in the figure:

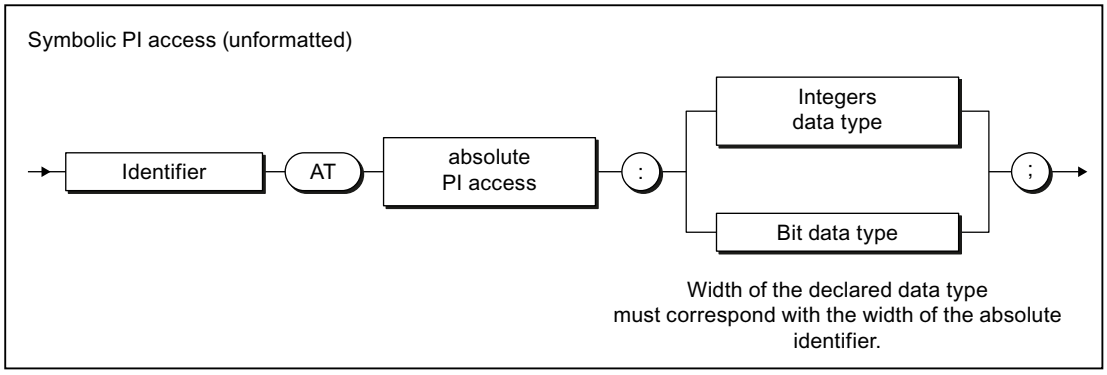


Figure 7-302 Declaration of a symbolic access to the process image

For the absolute PI access, see "Syntax for the identifier for an absolute PI access (Page 4888)".

The range of the declared integer or bit data type must correspond to the range of the absolute PI access, see "Possible data types of the symbolic PI access (Page 4890)". After declaring a numerical data type, you can address the contents of the process image as an integer.

See also Example for the declaration (Page 4891).

Possible data types for symbolic PI access

In the following cases, a data type that differs from that of the absolute PI access can be assigned to the fixed process image of the BackgroundTask (Page 4881). The data type width must correspond to the data type width of the absolute PI access.

- For the declaration of a symbolic PI access (Page 4889).
- For the creation of an I/O variable (Page 4891).

If you assign a numeric data type to the symbolic PI access or to the I/O variables, you can access these variables as integer.

Table 7-467 Possible data types for symbolic PI access

Data type of the absolute PI access	Possible data types of the symbolic PI access
BOOL (%In.x, %IXn.x, %Qn.x. %QXn.x)	BOOL
BYTE (%IBn, %QBn)	BYTE, SINT, USINT
WORD (%IWn, %QWn)	WORD, INT, UINT
DWORD (%IDn, %PQDn)	DWORD, DINT, UDINT

For the data type of the absolute PI access, see also "Syntax for the identifier for an absolute PI access (Page 4888)".

Example of symbolic PI access

If, for example, you want to access the CPU memory area (absolute PI access (Page 4888)) %IB10, but can respond flexibly to changes in your program, then declare a *myInput* variable with this CPU memory area as follows:

```
VAR
    myInput AT %IB10 : BYTE;
END_VAR
```

If you want to use the integer value of the memory area, declare the *myInput* variable as follows:

```
VAR
    myInput AT %IB10 : SINT;
END_VAR
```

If you want to use a CPU memory area other than %IB10 in your program at a later time, you only need to change the absolute PI access in the variable declaration.

Creating an I/O variable for access to the fixed process image of the BackgroundTask

You create I/O variables for access to the fixed process image for the background task in the symbol browser in the detail view; you must be in offline mode to do this.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select Insert new line
   or
   Scroll to the end of the table of variables (empty line).
3. In the detail view, select the Symbol browser tab and scroll down to the end of the variable table (empty row).
4. In the empty row of the table, enter or select the following:
   - **Name** of variable.
   - Under **I/O address**, the absolute PI access according to the "Syntax for the identifier for an absolute PI access" (Page 4888)
     (exception: The syntax %IXn.x or %QXn.x is not permitted for data type BOOL).
   - **Data type** of the I/O variables according to the "Possible data types of the symbolic PI access" (Page 4890).
5. Select optionally the display format used to monitor the variable in the symbol browser.

You can now access this variable using the address list or any program of the SIMOTION device.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and use them in your program sources.

Note that you can read and write outputs but you can only read inputs.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 4892).

Accessing I/O variables

You have created an I/O variable for:

- Direct access or process image of the cyclic tasks (Page 4872).
- Access to the fixed process image of the BackgroundTask (Page 4881).

You can use this I/O variable just like any other variable.

Note

Consistency is only ensured for elementary data types.

When using arrays, the user is responsible for ensuring data consistency.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the I/O variable using *_device.var-name* (predefined name space, see the "Predefined name spaces" table in "Name spaces").

It is possible to directly access an I/O variable that you created as a process image of a cyclic task. Specify direct access with *_direct.var-name* or *_device._direct.var-name*.

If you want to deviate from the default behavior when errors occur during variable access, you can use the _getSafeValue and _setSafeValue functions (see *SIMOTION Basic Functions* Function Manual).

For Errors associated with access to I/O variables, see *SIMOTION Basic Functions* Function Manual.

7.2.7.4 Using libraries

Libraries provide you with user-defined data types , functions and function blocks that can be used from all SIMOTION devices.

Libraries can be written in all programming languages; they can be used in all program sources (e.g. ST source files, MCC units).

You can obtain more details on inserting and managing libraries in the online help.

Note

The same rules as for the names of program source files apply to the library names, see Insert ST source file (Page 4587). In particular, the permissible length of the name depends on the SIMOTION Kernel version:

- As of Version V4.1 of the SIMOTION Kernel: maximum 128 characters.
- Up to Version V4.0 of the SIMOTION Kernel: maximum 8 characters.

With versions of the SIMOTION Kernel up to V4.0, a violation of the permissible length of the library name may not be detected until a consistency check or a download of the project is performed!

There is also the option of having a library make programs available, which can be called from other programs or function blocks. Please refer to the conditions which apply when calling a "program in a program" (Page 4762). In each case, the static data for the program called is stored once in the user memory of the device on which the library program is called. The same program instance data is used every time the program is called on the same device. A library program cannot be assigned to the execution system.

Compiling a library

In libraries, you can use all ST commands except for the ones listed in the table. In addition, you are not allowed to access some variables; these variables are also listed in this table .

Table 7-468 Illegal ST commands and variable access in libraries

Prohibited commands:

- *_getTaskId* function (see *SIMOTION Basic Functions* Function Manual).
- *_getAlarmId* function (see *SIMOTION Basic Functions* Function Manual).
- *_checkEqualTask* function (see *SIMOTION Basic Functions* Function Manual).
- If the library is not device-dependent (i.e. compiled without reference to a SIMOTION device or SIMOTION Kernel version):
  - System functions of SIMOTION devices (see the Parameter Manual for SIMOTION devices)
  - Version-dependent system functions

Prohibited variable accesses:

- Unit variables (retentive and non-retentive)
- Global device variables (retentive and non-retentive)
- I/O variables
- Instances of the technology objects and their system variables
- Variables of task names and configured messages (*_task* and *_alarm* namespaces, see Namespaces (Page 4902), *Predefined namespaces* table)
- If the library is not device-dependent (i.e. compiled without reference to a SIMOTION device or SIMOTION Kernel version):
  - System variables of SIMOTION devices (see the Parameter Manual for SIMOTION devices)
  - Configuration data of technology objects (see Parameter Manual of configuration data for the relevant SIMOTION technology package)

Note

The **Program status** debug function is not available in libraries.

Compiling an individual library

To compile an individual library, proceed as follows:

1. Select the library in the project navigator.
2. Select the **Edit > Object Properties** menu command.
3. Select the **Technology package** tab.
4. Select the SIMOTION devices (with SIMOTION kernel version) and the technology packet that you want to use as a basis for compiling the library; see the *SIMOTION Basic Functions* Function Manual.
5. Select **Accept and compile** from the context menu.

The library is compiled with reference to **all** selected SIMOTION devices, SIMOTION kernel versions and technology packages (and independently of devices).

Note

If the library to be compiled imports another library, note the following:

1. For the imported library, at least the same devices and SIMOTION kernel versions must be selected as for the importing library.
   Alternatively, the imported library can be compiled independently of devices if the prerequisites for this are fulfilled (refer to the *SIMOTION Basic Functions* Function Manual).
2. The imported library must already be compiled individually with reference to all configured devices, kernel versions and technology packages.
   Compilation of the library as part of a project-wide compilation is generally not sufficient.

Compiling a library as part of a project-wide compilation

When you compile the whole SIMOTION project (e.g. by choosing **Project > Save and recompile all** or by performing an XML import), the libraries used are also compiled.

Note

When performing project-wide compilation, note the following:

1. The system automatically identifies dependencies between libraries and selects the appropriate compilation sequence.
2. A library is **only** compiled with reference to the SIMOTION devices (including versions of the SIMOTION kernel) that are configured in the project and which use the library.
3. Other SIMOTION devices and kernel versions set for the library are ignored.

Know-how protection for libraries

You can protect libraries and their source files against unauthorized access by third parties. Protected libraries or sources can only be opened or exported as plain text files by entering a password.

You can:

- Provide individual sources of a library with know-how protection:
  Only the sources are protected against unauthorized access.
  The setting of the SIMOTION devices including the versions of the SIMOTION Kernel and the technology packages, for which the library is to be compiled, can still be changed and adapted by the user. Please refer to the *SIMOTION Basic Functions* Function Manual.
  The user can thus use the library for other SIMOTION devices and kernel versions.
- Provide the library with know-how protection:
  The following is then protected against unauthorized access:
  - All sources of the library
  - The setting of the SIMOTION device including the versions of the SIMOTION Kernel and the technology packages for which the library is to be compiled.

  You thus prevent that the user can use the library for other SIMOTION devices and kernel versions.
  Only use this setting if this is intended.

The SIMOTION online help provides additional information on know-how protection.

Note

If you export in XML format, the libraries or sources are exported in an encrypted form. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

Using data types, functions and function blocks from libraries

Before using data types, functions or function blocks from libraries, you must make them known to the ST source file. To do so, use the following construct in the interface section of the ST source file:

```
USELIB library-name [AS namespace];
```

In this case, *library-name* is the name of the library as it appears in the project navigator.

When multiple libraries are to be specified, enter them as a list separated by commas, e.g.:

```
USELIB library-name_1 [AS namespace_1],
    library-name_2 [AS namespace_2],
    library-name_3 [AS namespace_1], ...
```

You can use the optional *AS namespace* add-on to define a namespace (Page 4902).

- You can then access data types, functions, and function blocks in the library that have the same name as such an ST source file of a SIMOTION device (in the PROGRAMS folder).
- You can also use namespaces to change the names of data types, functions and function blocks in the library so that they have different names.

You can also assign the same namespace to different libraries.

Table 7-469 Example of use of namespaces with libraries

```
INTERFACE
    USELIB Bib_1 AS NS_1, Bib_2 AS NS_2;
    PROGRAM Main_Program;
END_INTERFACE

IMPLEMENTATION
    FUNCTION Function1 : VOID
        VAR
            ComID : CommandIdType;
        END_VAR
        ComId := _getCommandId();
    END_FUNCTION

    PROGRAM Main_program
        function1();         // Function from this source
        NS_1.Var1:=1;
        NS_2.Var1:=2;
        NS_1.function1();    // Function from the Bib1 library
        NS_2.function1();    // Function from the Bib2 library
    END_PROGRAM
END_IMPLEMENTATION
```

7.2.7.5 Use of the same identifiers and namespaces

Use of the same identifiers

General Information

It is possible to use unit variables and local variables (program variables, FB variables, FC variables) with the same name. When compiling a program source, the compiler searches for identifiers beginning with the current POU. The smaller validity range always takes priority over the larger validity range.

You can therefore use the same identifiers in different source file sections, as long as the rules below are adhered to. If a higher-level identifier is hidden by an identifier in a unit or POU, the compiler issues a warning.

Note

Under certain circumstances, the compiler may not issue a warning if, for example, the associated technology package is not imported.

Identifier in a program organization unit (POU)

All following identifiers in a POU must be unique:

- Local variables of the POU.
- Local data types of the POU.

They must not be identical to the following identifiers either:

- Reserved identifiers.
- Identifiers of the POU itself.

The compiler issues a warning when the following identifiers are hidden:

- Unit variables, data types and POU or the same or imported units
- Standard system functions, standard system function blocks, and associated data types.
- System functions and system data types of the SIMOTION device.
- Program organization units (POUs) and data types from imported libraries
  - This can be resolved by entering a user-defined namespace.
- System functions and system data types from imported technology packages.
  - This can be resolved by entering a user-defined namespace.
- SIMOTION device variables (system variables, I/O variables, global device variables).
  - This can be resolved by entering the predefined namespace *_device*.
- Technology objects configured on the SIMOTION device
  - This can be resolved by entering the predefined namespace *_to*.

Identifiers in a unit

Public identifiers of all units (unit variables, data types, and POUs) must be unique throughout the device.

All the following identifiers must be unique within a unit:

- Unit variables (declared in the interface or implementation section)
- Data types (declared in the interface or implementation section)
- Program organization units (POUs)

They must not be identical to the following identifiers either:

- Reserved identifiers.
- Unit variables, data types and POU imported units.
- Standard system functions, standard system function blocks, and associated data types.
- System functions and system data types of the SIMOTION device.

- Program organization units (POUs) and data types from imported libraries
  - This can be resolved by entering a user-defined namespace.
- System functions and system data types from imported technology packages.
  - This can be resolved by entering a user-defined namespace.

The compiler issues a warning when the following identifiers are hidden:

- SIMOTION device variables (system variables, I/O variables, global device variables).
  - This can be resolved by entering the predefined namespace *_device*.
- Technology objects configured on the SIMOTION device.
  - This can be resolved by entering the predefined namespace *_to*.

Identifiers on the SIMOTION device (e.g. I/O variables, global device variables)

All the following identifiers on the SIMOTION device must be unique:

- I/O variables
- Global device variables
- System variables of the SIMOTION device
- System functions and system data types of the SIMOTION device.

They must not be identical to the following identifiers either:

- Reserved identifiers.
- Standard system functions, standard system function blocks, and associated data types.

Example

The following example illustrates this situation. It shows that for use of identical names for unit variables (large validity range) and FC variables (small variable scope), only the variables declared in the function are valid within this source file section. The unit variables are only valid in POUs in which no local variables of the same name were declared. See the example.

Example of identifier validity

```
TYPE
    type_a : (enum1, enum2, enum3);
END_TYPE

VAR_GLOBAL
    var_a, var_b : DINT; // Unit variables
END_VAR

FUNCTION fc_1 : VOID
    VAR
        var_a : type_a;  // Declaration hides UNIT variable
        var_c : DINT;    // Local variable
    END_VAR
    // Permitted statements
    var_a := enum2;      // Access to local variable
    var_b := 100;        // Access to unit variable
    var_c := -1;         // Access to local variable
    // Invalid statement
    // var_a := 200;
END_FUNCTION

FUNCTION fc_2 : VOID
    VAR
        var_b : type_a;  // Declaration hides UNIT variable
        var_c : type_a;  // Local variable
    END_VAR
    // Permitted statements
    var_a := -100;       // Access to unit variable
    var_b := enum3;      // Access to local variable
    var_c := enum1;      // Access to local variable
    // Invalid statement
    // var_b := 200;
END_FUNCTION
```

Namespaces

You can also access data types, unit variables, functions, and function blocks defined outside of a program source (e.g. in libraries, technology packages, and on the SIMOTION device) using their names.

When compiling a program source, the compiler searches for identifiers beginning with the current POU. The data types, variables, functions, or function blocks declared in a program source therefore hide identifiers with the same name which have been defined outside the source, see Use of the same identifiers (Page 4897). In order to still access these hidden identifiers, you can use namespaces in certain cases.

Namespaces for libraries and technology packages

In the import statement for libraries and technology packages, you can define namespaces in order to access the data types, functions, or function blocks of these libraries and technology packages.

```
USELIB library-name_1 [AS lib_namespace_1],
    library-name_2 [AS lib_namespace_2],
    library-name_3 [AS lib_namespace_1], ...

USEPACKAGE tp-name_1 [AS tp_namespace_1],
    tp-name_2 [AS tp_namespace_2],
    tp-name_3 [AS tp_namespace_1], ...
```

You can also use namespaces to make names consistent within different libraries.

If you wish to use a data type, a function or a function block from a library or a technology package, place the namespace identifier in front of the name, separated by a period, for example, *namespace.fc-name*, *namespace.fb-name*, *namespace.type-name*

Example

The following example shows how to select the Cam technology package, assign it the namespace Cam1 and use the namespace:

Example of selecting a technology package and using a namespace

```
INTERFACE
    USEPACKAGE Cam AS Cam1;
    USES ST_2;
    FUNCTION function1;
END_INTERFACE

IMPLEMENTATION
    FUNCTION function1 : VOID
    VAR_INPUT
        p_Axis : posAxis;
    END_VAR
    VAR
        retVal : DINT;
    END_VAR

    retVal:= Cam1._enableAxis (
            axis := p_Axis,
            nextCommand := Cam1.WHEN_COMMAND_DONE,
            commandId := _getCommandId() );
    END_FUNCTION
END_IMPLEMENTATION
```

Note

If a namespace is defined for an imported library or technology package, this must **always** be specified if a function, function block, or data type from this library or technology package is being used. See above example: Cam1._enableAxis, Cam1.WHEN_COMMAND_DONE.

Predefined namespaces

Namespaces are predefined for device- and project-specific variables as well as TaskID and AlarmID variables. If necessary, write their designation before the variable names, separated by a period, for example, *_device.var-name* or *_task.task-name*

Table 7-470 Predefined namespaces

Namespace	Description
_alarm	For AlarmId: The _alarm.*name* variable contains the AlarmId of the message with the *name* identifier (see SIMOTION Basic Functions Function Manual).
_device	For device-specific variables (global device variables, I/O variables, and system variables of the SIMOTION device).
_direct	For direct access to I/O variables – see Direct access and process image of the cyclic tasks (Page 4872). Local namespace for _device. Nesting as in _device._direct.*name* is permitted.
_project	For names of SIMOTION devices in the project; only used with technology objects on other devices. With unique project-wide names of technology objects, used also for these names and their system variables.
_task	For TaskID: The _task.*name* variable contains the TaskId of the task with the *name* identifier (see SIMOTION Basic Functions Function Manual).
_quality	As of version V4.2 of the SIMOTION Kernel: For the detailed status of I/O variables (Page 4879). A value with data type DWORD is supplied. Local namespace for _device. Nesting as in _device._quality.*name* is permitted.
_to	For technology objects configured on the SIMOTION device, and their system variables and configuration data. Not for system functions and data types of the technology objects. In this case, if necessary, use the user-defined namespace for the imported technology package

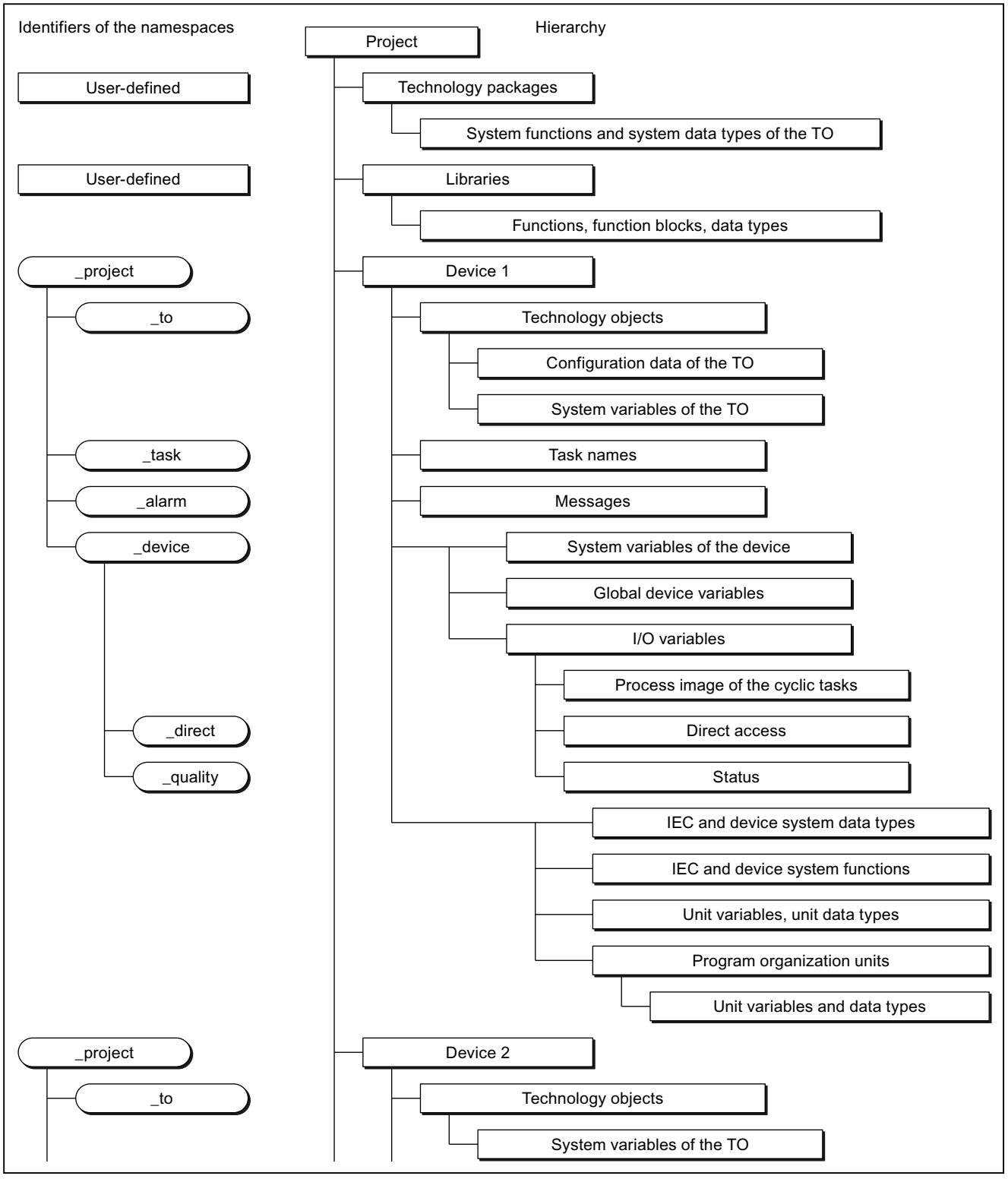


Figure 7-303 Namespaces and identifier hierarchy

Object-oriented namespace

Declaration of object-oriented namespaces

You declare object-oriented namespaces in the interface or implementation section of an ST source file or within another object-oriented namespace.

Use the following syntax:

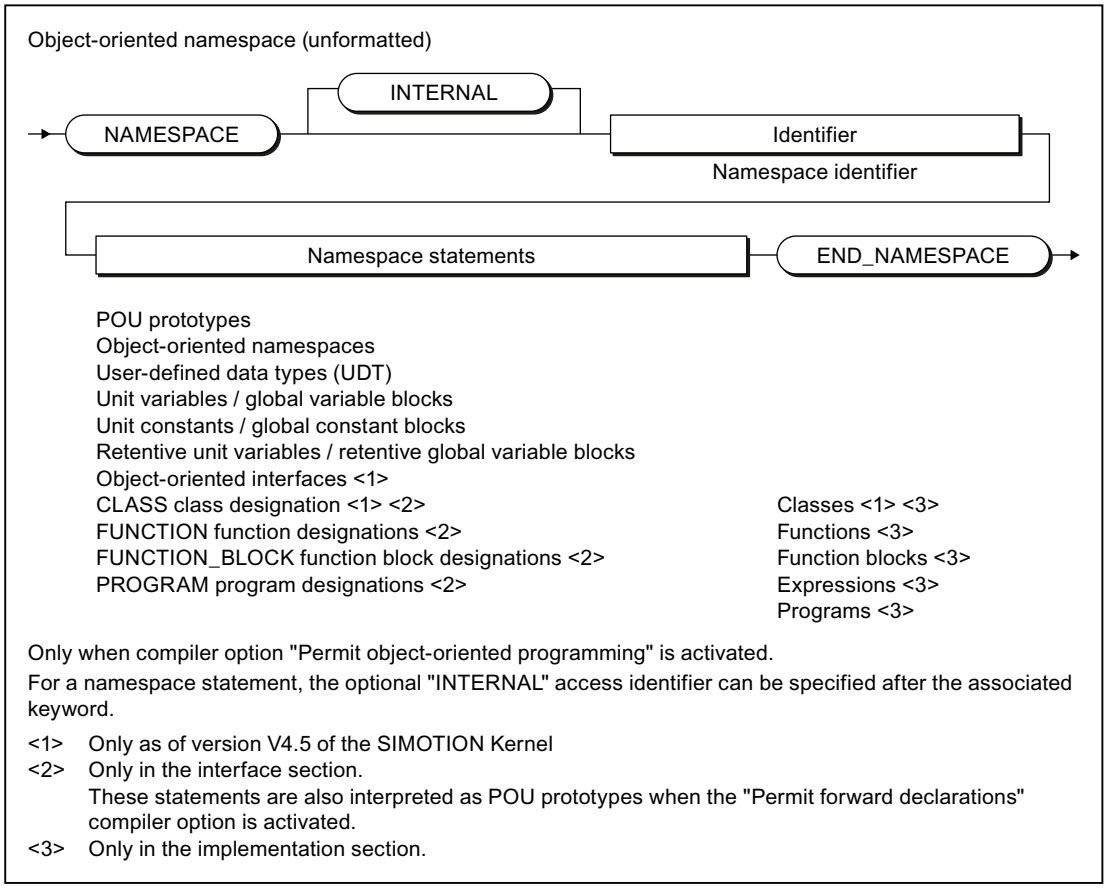


Figure 7-304 Syntax: Object-oriented namespace

Specify an identifier for the name of the namespace after the NAMESPACE and the optional INTERNAL keywords.

This is then followed by:

- The statements permitted in the namespace, e.g.:
  - Declarations of user-defined data types
  - Global variable declarations
  - Object-oriented interfaces
  - Program organization units, such as classes, functions, function blocks, programs and their prototypes
  - Other namespaces
- The END_NAMESPACE keyword

The INTERNAL keyword is possible only in namespace declarations within a namespace.

Object-oriented namespaces can be nested as required. Whereby, the nesting can be abbreviated by specifying a structured namespace path of the form *namespace_1.namespace_2*.

```
// Declaration of a nested namespace
NAMESPACE ns1
  NAMESPACE INTERNAL ns_int
    (* Elements of ns1.ns_int *)
  END_NAMESPACE
  (* Further elements of ns1 *)
END_NAMESPACE
// Declaration of a nested namespace
// by fully qualified name
// ns1 is public; INTERNAL acts only on the last name part
NAMESPACE INTERNAL ns1.ns_int
  (* Elements of ns1.ns_int *)
END_NAMESPACE
```

As for structured variables, access can be made from outside an object-oriented namespace to its elements by prefixing the namespace identifier separated with a period to the element identifier: *namespace.element*.

The INTERNAL access identifier can be specified for every element defined within a namespace (default is public). The INTERNAL access identifier immediately follows the associated keyword, e.g. VAR_GLOBAL, VAR_GLOBAL CONSTANT, TYPE, CLASS, INTERFACE, FUNCTION_BLOCK, FUNCTION, PROGRAM or NAMESPACE. Methods, type and variable declarations, as well as constants, can also be declared within INTERNAL classes and function blocks; see sections "Access identifier within classes (Page 4768)" and "Access identifiers within object-oriented FBs (Page 4740)". Access can be made to elements identified with INTERNAL only within the higher-level namespace.

Name conflicts for elements defined within a namespace are not permitted.

The definition of a namespace does not need to be closed. If a previously defined namespace with the same name is found, the content of the declaration is included there. This also applies when the namespace is specified with a USES statement.

Predefined namespaces (Page 4902) (e.g. _task, _alarm, _device, _project) cannot be extended. Consequently, it is not permitted to use these identifiers for a namespace definition. Access per USING with these namespaces is also not possible.

If a previously defined namespace (defined in a separate source or imported from a source or library with USES or USELIB) is redeclared, the previous access identifier must be retained.

- If no access identifier is specified with a new declaration of a previously declared INTERNAL namespace, a compiler warning is issued and the namespace remains INTERNAL.
- A compiler error message will be issued if an attempt is made to declare a previously published namespace as INTERNAL subsequently.
- If the same namespace is declared in two different sources or libraries as INTERNAL and public, and these sources/libraries are used together in another source, the namespace is indicated as INTERNAL there.

This ensures that the elements of a namespace declared as INTERNAL cannot be made public subsequently.

If public program organization units (POUs) are classified in namespaces, the namespace must be specified in the interface section and in the implementation section of the source.

Access to namespaces and their elements

Within a namespace, all included elements can be used without name prefix, even when they were declared as INTERNAL. If no element was found for an identifier within the namespace, a search is made for the identifier in the enclosing scope that can also be a namespace or the global scope of the source. Access without name prefix is also possible to such found elements. This also applies to elements declared as INTERNAL in the enclosing namespace, because the lower-level namespace in which the identifier search begins always belongs to the current namespace.

In addition to the above-mentioned search in the scope, it is also possible to access elements contained in the namespaces. This requires the complete designation of the namespace written separated with a period before the element identifier: *namespace_1.namespace_11.element*. Because an external access is typically involved, no access is possible to elements declared as INTERNAL.

Table 7-471 Example: Access to elements of a class declared as INTERNAL

```
NAMESPACE ns1
  CLASS INTERNAL cl_int
    METHOD PUBLIC m_public
      // Warning: Method is INTERNAL.
      // (The complete class is INTERNAL)
      ;
    END_METHOD
    METHOD INTERNAL m_internal
      ;
    END_METHOD
  END_CLASS
  VAR_GLOBAL
    vg_clint : cl_int;
    // Instance can be created, because within ns1.
  END_VAR
END_NAMESPACE
```

```
FUNCTION_BLOCK fb_test
  VAR
    v_clint : ns1.cl_int;
    // Instance cannot be created
    // cl_int is INTERNAL in ns1.
  END_VAR
  ns1.vg_clint.m_public();
  (* Access to method m_public not possible Although the instance vg_clint from ns1 is
visible, access is not possible to its elements *)
END_FUNCTION_BLOCK
```

Table 7-472 Example: Access to elements of a namespace declared as INTERNAL

```
NAMESPACE ns1
  NAMESPACE INTERNAL ns_int
    CLASS cl
      METHOD PUBLIC   m_public   ; END_METHOD
      METHOD INTERNAL m_internal ; END_METHOD
    END_CLASS
  END_NAMESPACE
  VAR_GLOBAL
    vg_cl : ns_int.cl;
    // Instance can be created, because within ns1.
  END_VAR
END_NAMESPACE
FUNCTION_BLOCK fb_test1
  VAR
    v_cl : ns1.ns_int.cl;
    // Instance cannot be created
    // ns_int is INTERNAL in ns1.
  END_VAR
  ns1.vg_cl.m_public();
  // Access possible to method m_public
  ns1.vg_cl.m_internal();
  // Access not possible to method m_internal
END_FUNCTION_BLOCK
```

USING directive

The USING directive has the following syntax:

```
USING namespace-list;
```

The namespace list is a list of namespace identifiers each separated by a comma.

Example:

```
USING namespace_1, namespace_2;
```

The USING directive affects the scope of the source file section in which it is specified. In this scope, the public elements of the namespaces specified in the USING directive can be used without details of the namespace identifier. These elements are considered as if they were declared themselves there.

This means name conflicts between the elements from different namespaces shown with USING, and elements of the current scope are not permitted. Compiler error messages are issued.

The USING directive always acts in the scope of the source file section in which it is specified. It does not apply to external accesses in this source file section. Consequently, the USING directive is always restricted to the current source. It has no effect in other sources that the current source uses.

The namespace identifier specified on the USING directive (possibly cascaded) must reference a namespace valid at this time. A USING directive does not implicitly declare namespaces.

USING directives can be specified at the following places, possibly several times successively:

- In the interface section of the source:
  Immediately after statements for importing technology packages, libraries and other sources (USEPACKAGE, USELIB, USES).
  This allows, for example, declared elements in namespaces for libraries to be shown in the global namespace.
- For the **first** declaration of a namespace within a source:
  Immediately on the following namespace identifier itself.
- Within a program organization unit (POU), e.g. class, function block, object-oriented interface, program, method:
  Immediately before the declaration section of the POU.

Examples

Table 7-473 Example: USING directive in the interface section of a source

```
INTERFACE
  USELIB mylib;   // Link with the mylib library
  USING ns_mylib; // Show the elements of the namespace
       // ns_mylib in the global namespace of the source
  // ...
END_INTERFACE
```

Table 7-474 Example: USING directive in a namespace

```
NAMESPACE ns2
  USING ns1;
  USING ns1.ns11;
  // ...
END_NAMESPACE
```

Table 7-475 Example: USING directive in the interface section of a program organization unit

```
CLASS cl
  USING ns1;
  USING ns1.ns11;
  // ...
  METHOD m : VOID
    USING ns1.ns11.ns111;
    ;
  END_METHOD
END_CLASS
```

Note

The USING directive cannot be used on namespaces included in the namespace path of the current source file section.

Table 7-476 Example: USING directive to a higher-level namespace not possible

```
NAMESPACE ns1.ns11.ns111
  CLASS cltest
    USING ns11; // Error: ns11 is part of its own declaration path
    // ...
  END_CLASS
END_NAMESPACE
```

Note

Namespace identifiers of other USING directives are not visible within a USING directive.

Table 7-477 Example: USING directive for nested namespaces

```
NAMESPACE ns1.ns11
// ...
END_NAMESPACE
NAMESPACE ns_test
  USING ns1;
  USING ns11;     // Error:
    // Namespace ns11 not visible for USING directive
  USING ns1.ns11; // Specification must be made as such
END_NAMESPACE
```

7.2.7.6 Reference data

The reference data provide you with an overview of:

- on utilized identifiers with information about their declaration and use (Cross-reference list (Page 4910)).
- on function calls and their nesting (Program structure (Page 4914))
- on the memory requirement for various data areas of the program sources (Code attributes (Page 4915))

Cross-reference list

The cross-reference list shows all identifiers in program sources (e.g. ST source files, MCC units):

- Declared as variables, data types, or program organization units (program, function, function block)
- Used as previously defined types in declarations
- Used as variables in the statement section of a program organization unit.

Generating and updating a cross-reference list

Initially, the cross-reference list is generated automatically when opening. Open a cross-reference list, e.g. after selecting an ST source file or library via the **Edit > Display reference data > Cross-references** menu. After changes, the update is partly performed automatically and can always be triggered manually in the detail view via the  button. When updating via the button, a check is performed as to whether a compilation is required. If a compilation is required, this is indicated by a yellow triangle next to the button.

Update of the cross-reference list when selecting a program

The list is updated automatically when opening the selected program. The local defined identifiers are therefore up-to-date.

External identifiers in the program are not updated when opening.

The opened cross-reference list is also updated automatically when compiling the program.

Update of the cross-reference list when selecting a tree (CPU, project, library, program folder)

The cross-reference list is generated automatically when opening the first time. There is no automatic update when opened again.

Note

An error-free compilation is required for a correct, consistent display of the reference data. If required, compile the project, the CPU, the program or the library first.

Content of the cross-reference list

The cross-reference list contains all the identifiers assigned to the element selected in the project navigator. The applications for the identifiers are also listed in a table:

Details of how to work with the cross-reference list are provided in the section titled "Working with the cross-reference list (Page 4913)".

Table 7-478 Meanings of columns and selected entries in the cross-reference list

Column	Entry in column	Meaning
Name		Identifier name
Type		Identifier type
	Name	• Data type of a variable (e.g. REAL, INT) • POU type (e.g. PROGRAM, FUNCTION)
	DERIVED	Derived data type
	DERIVED ANY_OBJECT	TO data type
	ARRAY ...	ARRAY data type
	ENUM ...	Enumerator data type
	STRUCT ...	STRUCT data type
Declaration		Location of declaration
	Name (UNIT)	Declaration in the program source *name*
	Name (LIB)	Declaration in the library *name*
	Name (TO)	System variable of the technology object *name*
	Name (TP)	Declaration in the default library specified: • Technology package *name* • std_fct = IEC library • device = device-specific library
	Name (DV)	Declaration on the SIMOTION device *name* (e.g. I/O variable or global device variable)
	_project	Declaration in the project (e.g. technology object)
	_device	Internal variable on the SIMOTION device (e.g. TaskStartInfo)
	_task	Task in the execution system
Use		Use of identifier
	CALL	Call as subprogram (static binding)
	CALL VTAB	Call of a method of the dynamic binding
	ENUM *name*	As element when declaring the enumerator data type *name*
	I/O	Declaration as I/O variable
	R	Read access
	R (TYPE)	As data type in a declaration
	R/W	Read and write access
	STRUCT *name*	As component when declaring the structure *name*
	TYPE	Declaration as data type or POU
	Variable type (e.g. VAR, VAR_GLOBAL)	Declaration as variable of the variable type specified
	W	Write access

Column	Entry in column	Meaning
Path specification		Path specification for the SIMOTION device or program source
	Name	SIMOTION device *name*
	Name1/*Name2*	• Program source *name2* on SIMOTION device *name1* • Program source *name2* in library *name1*
	Name/taskbind.hid	Execution system of the SIMOTION device *name*
Range		Range within the SIMOTION device or program source
	INTERFACE	Interface section of the program source
	POE type name (i.e. CLASS *name*, FUNCTION *name*, FUNCTION_BLOCK *name* INTERFACE *name* PROGRAM *name*)	Program Organization Unit (POU) *Name* within the program source. • In an MCC chart: Additional serial numbers for the command (block numbers) • In a LAD/FBD program: Additional serial numbers of the network
	I/O address	I/O variable
	METHOD *Name_1*::*Name_2*	Method *Name_2* within the class or the function block *Name_1*
	TASK *name*	Assignment for the task *name*
	UNIT	Implementation section of the program source, additional specification as POU to be made public in the interface section of the program source
	UNIT - IMPLEMENTATION	Implementation section of the program source
	_device	Global device variable
Language		Programming language of the program source
Line/Block		• In an ST source: Line number within the program source • In an MCC or LAD/FBD source: Relative line number within the command (block) or network. **Note** The absolute line number within the program source, which you need, for example, for the trace function "Trigger to code point", is obtained as follows: – Press the **Determine program line** button. – In the dialog box, press the button **Copy program line to the clipboard**.

Note

Single-step tracking and trace diagnostic functions in MCC programming

Additional variables and functions are created or used for these diagnostics functions:

- The variables TSI#dwuser_1 and TSI#dwuser_2 of the TaskStartInfo are used for the single-step tracking diagnostic function.
- Various internal functions and variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. The TSI#currentTaskId variable of the TaskStartInfo is also used.

With activated diagnostic function, these variables and functions are used for the control of the diagnostics function. These variables and functions must not be used in the user program.

Working with a cross-reference list

In the cross-reference list you are able to:

- Sort the column contents alphabetically:
  - To do this, click the header of the appropriate column.
- Search for an identifier or entry:
  - Click the "Search" button and enter the search term.
- Filter (Page 4913) the identifiers and entries displayed.
- Copy contents to the clipboard in order to paste them into a spreadsheet program, for example.
  - Select the appropriate lines and columns.
  - Press the CTRL+C shortcut.
- Print the content (**Project > Print**).
- Open the referenced program source and position the cursor on the relevant line of the ST source file (or MCC command or LAD/FBD element):
  - Double-click on the corresponding line in the cross-reference list.
    or
  - Place the cursor in the corresponding line of the cross-reference list and click the "Go to application" button.

Further details about working with cross-reference lists can be found in the online help.

Filtering the cross-reference list

You can filter the entries in the cross-reference list so that only relevant entries are displayed:

1. Click the "Filter settings" button.
   The "Filter Setting for Cross References" window will appear.
2. Activate the "Filter active" checkbox.
3. If you also want to display system variables and system functions:
   - Deactivate the "Display user-defined variables only" checkbox.
4. Set the desired filter criterion for the relevant columns:
   - Select the relevant entry from the drop-down list box or enter the criterion.
   - If you want to search for a character string within an entry: Deactivate the "Whole words only" checkbox.
5. Confirm with **OK**.

The contents of the cross-reference list will reflect the filter settings selected.

Note

A filter is automatically activated after the cross-reference list has been created.

Program structure

The program structure contains all the function calls and their nesting within a selected element.

You can display the program structure selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the program structure.
2. Select the **Edit > Display reference data > Program structure** menu command.
   The cross-reference tab is replaced by the program structure tab in the detail view.

Note

The display data is updated every time the program structure is opened.

You can update the detail view of an opened program structure with the F5 key.

Content of the program structure

A tree structure appears, showing:

- as base respectively
  - the program organization units (programs, functions, function blocks) declared in the program source, or
  - the execution system tasks used
- below these, the subroutines referenced in this program organization unit or task.

For structure of the entries, see table:

Table 7-479 Elements of the display for the program structure

Element	Description
Base (declared POU or task used))	List separated by a comma • Identifier of the program organization unit (POU) or task The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Identifier of the program source in which the POU or task was declared, with add-on [UNIT] • Minimum and maximum stack requirement (memory requirement of the POU or task on the local data stack), in bytes [Min, Max] • Minimum and maximum overall stack requirement (memory requirement of the POU or task on the local data stack including all called POUs), in bytes [Min, Max]
Referenced POU	List separated by a comma: • Identifier of called POU The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Optionally: Identifier of the program source / technology package in which the POU was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library Add-on (TP): System function from technology package • with function blocks or methods only: Identifier of the instance of the function block or the higher-level class • with function blocks or methods only: Identifier of program source in which the instance of the function block or of the higher-level class was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library • Number of the line in the (compiled) source in which the POE is called; several line numbers are separated by "/".

Code attributes

You can find information on or the memory requirement of various data areas of the program sources under code attribute.

You can display the code attributes selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the code attributes.
2. Select the **Edit > Display reference data > Code attributes** menu command.
   The **Cross-references** tab is now replaced by the **Code attributes** tab in the detail view.

Note

The display data is updated every time the code attributes are opened.

You can update the detail view of the opened code attributes with the F5 key.

Code attribute contents

The following are displayed in a table for all selected program sources:

- Identifier of program source,
- Memory requirement, in bytes, for the following data areas of the program source:
  - **Dynamic data**: All unit variables (retentive and non-retentive, in the interface and implementation sections),
  - **Retain data**: Retentive unit variables in the interface and implementation section,
  - **Interface data**: Unit variables (retentive and non-retentive) in the interface section,
- the **Code size** during the last compilation in bytes,
- the **Number of referenced sources**:
  The maximum number of connected sources is displayed (including system libraries), regardless of whether they are downloaded to the target system at a later date.

Reference to variables

If you have selected the identifier of a variable in the open Editor window for each programming language, you can use the other places of use over the context menu to list or to skip these variables.

You select the identifier of a variable:

- In SIMOTION ST: In the Editor window of an ST source.
- In SIMOTION MCC: In the input field on the parameter screen of an open MCC command within an MCC chart
- In SIMOTION LAD/FBD: In the Editor window of a LAD/FBD program

An identifier is recognized as a variable under the following conditions:

1. The identifier is declared as a variable. The scope of the variable includes the respective window (ST source, MCC chart, LAD/FBD program).
2. The program source is compiled.
3. The variable is selected as follows (in an open parameter screen within an MCC chart):
   - The identifier is fully marked
     or
   - The cursor is within the identifier.

Note

In arrays and structures, only the variable can be selected, not a single element.

Using the **Go to** context menu, you have the following options:

- To jump to the next local place of use:
  Select the context menu **Go to > Local use >>**.
  The next place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the previous local place of use:
  Select the context menu **Go to > Local use <<**.
  The previous place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the declaration position:
  Select the context menu **Go to > Declaration position**.
  The declaration position of the variables is selected. The corresponding program source is opened, if necessary.
- List all places of use
  Select the context menu **Go to > Places of use ...**.
  In the detailed view, all places of use of the variables within their scope (including the declaration position) are listed The structure of this list is similar to the List of cross references (Page 4911).
  This is how you jump to a preferred place of use:
  - Double-click on the corresponding line.
    or
  - Place the cursor in the corresponding line and click the "Go to application" button.

7.2.7.7 Controlling the preprocessor and compiler with pragmas

A pragma is used to insert an ST source file text (e.g. statements), which influences the compilation of the ST source file.

Pragmas are enclosed in { and } brackets and can contain (see figure):

- Preprocessor statements for controlling the preprocessor, see Controlling the preprocessor (Page 4918).
  The pragmas with preprocessor statements contained in an ST source file are evaluated by the preprocessor and interpreted as control statements.
- Attributes for compiler options to control the compiler, see Controlling compiler with attributes (Page 4922).
  The pragmas with attributes for compiler options contained in an ST source file are evaluated by the compiler and interpreted as control statements.
- Non-assigned compiler messages, see Issuing non-assigned compiler messages (Page 4926).
  Non-assigned compiler messages are issued in the "Compile/check output" tab in the detailed view.

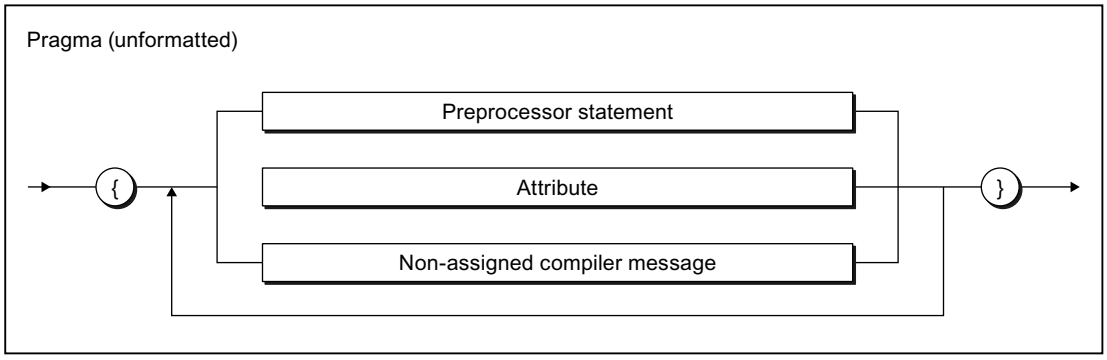


Figure 7-305 Syntax: Pragma

Note

Be sure to use the correct pragma syntax (e.g. upper- and lower-case notation of attributes).

Unrecognized pragmas are ignored with no warning message.

Controlling the preprocessor

The preprocessor prepares an ST source file for compilation. For example, character strings can be defined as replacement texts for identifiers, or sections of the source program can be hidden/ shown for compilation.

The preprocessor is disabled by default. You can activate it as follows:

- Globally for all program source files and programming languages within the project, see "Global settings of the compiler (Page 4623)".
- Local for a program source file, see "Local compiler settings (Page 4626)".

During the compilation of a program source file, you will be informed about the preprocessor actions. This requires, however, that the display of class 7 warnings is activated, see Meanings of the warning classes (Page 4633). You specify the details for issued warnings and information:

- In the global or local settings of the compiler.
- With the _U7_PoeBld_CompilerOption := warning:*n*:off or warning:*n*:on attribute within an ST source file, see "Controlling compiler with attributes (Page 4922)".

Like all compiler messages, information about the preprocessor actions is shown on the "Compile/check output" tab of the detail view.

Note

You can also view the text of the ST source file modified by the preprocessor:

1. Open the ST source file.
2. Select the **ST source file > Execute preprocessor** menu command.

The modified source text is shown in the "Compile/check output" tab of the detail view.

Preprocessor statement

You can control the preprocessor by means of statements in pragmas. The statements specified in the following syntax diagram can be used. These statements act on all subsequent lines of the ST source file.

They can be used in ST source files of a SIMOTION device or a library.

You can make definitions for the preprocessor (see Making preprocessor definitions (Page 4635)) in the property dialog box of the ST source file. This enables you also to control the preprocessor with ST source files with know-how protection (see Know-how protection for ST source files (Page 4635)).

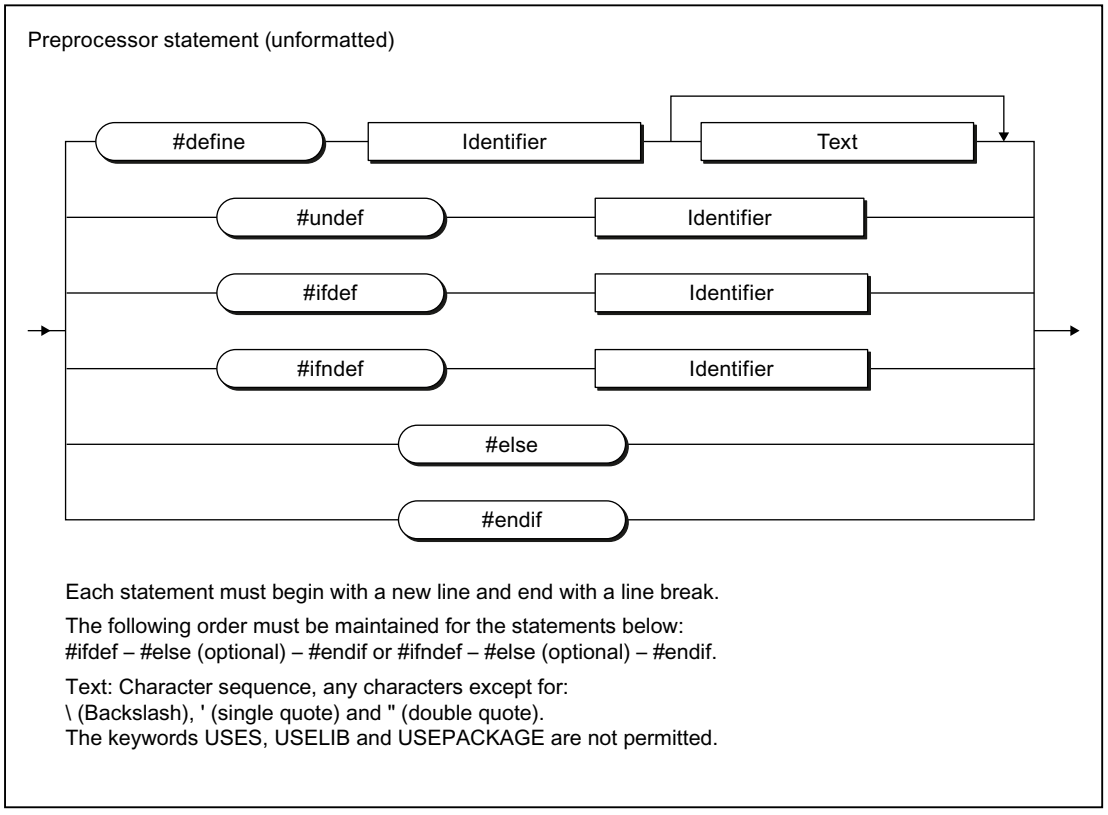


Figure 7-306 Syntax: Preprocessor statement

Table 7-480 Preprocessor statements

Statement	Meaning
#define	The specified identifier will be replaced below by the specified text. Permissible characters: See table footnote.
#undef	The replacement rule for the identifier is cancelled.
#ifdef	For variant formation (conditional compilation) If the specified identifier is defined, the following program lines (until the next pragma that contains #else or #endif) are compiled by the compiler.
#ifndef	For variant formation (conditional compilation) If the specified identifier is **not** defined, the following program lines (until the next pragma that contains #else or #endif) are compiled by the compiler.
#else	For variant formation (conditional compilation) Alternative branch to #ifdef or #ifndef. The following program lines (until the next pragma containing #endif) are compiled by the compiler, if the preceding query with #ifdef or #ifndef was not fulfilled.
#endif	Concludes variant formation with #ifdef or #ifndef.
Permissible characters: • For identifiers: In accordance with the rules for identifiers (Page 4656). • For text: Sequence of any characters other than \ (backslash), ' (single quote) and " (double quote). The keywords USES, USELIB and USEPACKAGE are not permitted.	

Note

Each preprocessor statement must begin with a new line and end with a line break. Consequently, the curly brackets ({ and }) enclosing the pragma must be placed in separate lines of the ST source file!

In the case of pragmas with #define statements, please note:

- Pragmas with #define statements in the interface section of an ST source file are declared public. The defined identifiers can be used with the USES statement into other ST source files of the same SIMOTION device or of the same library.
- Identifiers defined in pragmas of libraries cannot be imported into ST source files of a SIMOTION device.
- Redefinition of reserved identifiers is not possible.

You can also make preprocessor definitions in the Properties dialog box of the ST source file. In the case of different definitions of the same identifiers, #define statements within the ST source file have priority.

Example of preprocessor statements

Table 7-481 Example of preprocessor statements

ST source file With preprocessor statements

```
INTERFACE
    FUNCTION_BLOCK fb1;
    VAR_GLOBAL
        g_var     : INT;
    END_VAR
// Preprocessor definitions
{
#define my_define g_var
#define my_call f(my_define)
}
//     my_define -> g_var
//     my_call   -> f(g_var)
END_INTERFACE

IMPLEMENTATION
    FUNCTION f : INT
        VAR_INPUT
            i : INT;
        END_VAR
        f := i;
    END_FUNCTION

    FUNCTION_BLOCK fb1
        VAR_INPUT
            i_var : INT;
        END_VAR
        VAR_OUTPUT
            o_var : INT;
        END_VAR
        my_define := i_var;
// Delete the preprocessor definition
// For my_define
{
#undef my_define
}
        o_var := my_call + 1;
{
#ifdef my_define
}
        my_define := i_var;
{
#endif
}
    END_FUNCTION_BLOCK
END_IMPLEMENTATION
```

Preprocessor output

```
INTERFACE
    FUNCTION_BLOCK fb1;
    VAR_GLOBAL
        g_var     : INT;
    END_VAR

{

}


END_INTERFACE

IMPLEMENTATION
    FUNCTION f : INT
        VAR_INPUT
            i : INT;
        END_VAR
        f := i;
    END_FUNCTION

    FUNCTION_BLOCK fb1
        VAR_INPUT
            i_var : INT;
        END_VAR
        VAR_OUTPUT
            o_var : INT;
        END_VAR
        g_var := i_var;


{

}
        o_var := f(g_var) + 1;
{


}
    END_FUNCTION_BLOCK
END_IMPLEMENTATION
```

Controlling compiler with attributes

You can use attributes within a pragma to control the compiler.

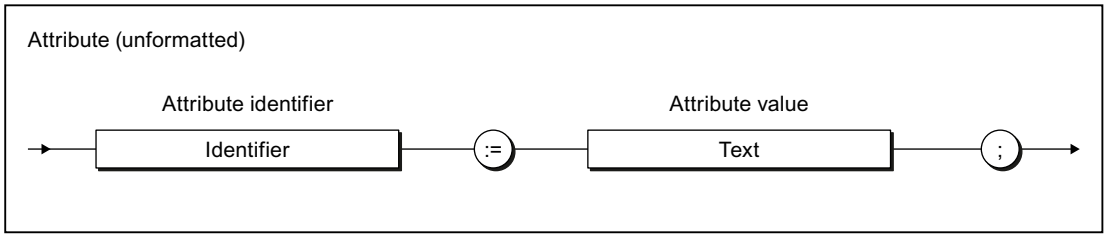


Figure 7-307 Syntax: Attributes for compiler options

Table 7-482 Permissible attributes for compiler options

Attribute identifier	Attribute value	Meaning
_U7_PoeBld_CompilerOption	The attribute affects the output of compiler warnings and information within an ST source file, as well as all subsequent lines of the ST source file.	
	warning:*n*:err	Outputs the warning or information specified by the number *n* as an error. Permissible values for *n*: *n* = 16000 and higher: Number for a warning or information:
	warning:*n*:off	Warnings or information specified by the number *n* are not displayed. Permissible values for *n*: *n* = 0 to 7: Warning class, see Meanings of the warning classes (Page 4633). *n* = 16000 and higher: Number for a warning or information:
	warning:*n*:on	Warnings specified by the number *n* are displayed Permissible values for *n*: *n* = 0 to 7: Warning class, see Meanings of the warning classes (Page 4633). *n* = 16000 and higher: Number for a warning or information:

Attribute identifier	Attribute value	Meaning
HMI_Export	The attribute changes the variables available on HMI devices by default. It must be placed directly after the associated keyword of the following declaration blocks: • In the interface or implementation section of an ST source file: – VAR_GLOBAL – VAR_GLOBAL RETAIN • In the declaration section of a function block provided that the compiler option (Page 4623) "Permit object-oriented programming" is activated: – VAR, VAR PUBLIC, VAR PRIVATE – VAR RETAIN, VAR PRIVATE RETAIN • In the declaration section of a class (as of version V4.5 of the SIMOTION Kernel): – VAR PUBLIC – VAR, VAR PROTECTED, VAR PRIVATE – VAR RETAIN, VAR PROTECTED RETAIN, VAR PRIVATE RETAIN It affects only the unit variables declared in the associated declaration block. Detailed description of the HMI export, in particular the effect of the attribute depending on the version of the SIMOTION kernel: see Variables and HMI devices (Page 4864).	
	FALSE	This attribute value is permissible: • In the interface section of an ST source file • In the following declaration blocks of a function block provided that the compiler option (Page 4623) “Permit object-oriented programming" is activated: – VAR, VAR PUBLIC, VAR PRIVATE • In the following declaration blocks of a class (as of version V4.5 of the SIMOTION Kernel): – VAR PUBLIC The variables declared in the associated declaration block are **not** available on HMI devices.
	TRUE	This attribute value is permissible: • In the implementation section of an ST source file. • In the following declaration blocks of a function block provided that the compiler option (Page 4623) "Permit object-oriented programming" is activated: – VAR RETAIN, VAR PRIVATE RETAIN • In the following declaration blocks of a class (as of version V4.5 of the SIMOTION Kernel): – VAR, VAR PROTECTED, VAR PRIVATE – VAR RETAIN, VAR PROTECTED RETAIN, VAR PRIVATE RETAIN The variables declared in the associated declaration block are available on HMI devices.

Attribute identifier	Attribute value	Meaning
BlockInit_OnChange	The attribute changes the standard definition whether a download in RUN mode is possible when a change is made to the version identification of the associated declaration block. It must be placed directly after the associated keyword of the following declaration blocks: • VAR_GLOBAL (in the interface and implementation section) • VAR_GLOBAL RETAIN (in the interface and implementation section) • VAR (only for programs in a unit if the "Create program instance data only once" compiler option is active). It affects only the variables declared in the associated declaration block. See also Version ID of global variables and their initialization during download (Page 4858).	
	FALSE	Download in RUN mode is **not** possible when the version identification of the declaration block is changed (default).
	TRUE	Download in RUN mode is possible despite the change to the version identification of the declaration block. The variables of the declaration block are initialized in the process.
BlockInit_OnDeviceRun	Only as of Version V4.1 of the SIMOTION kernel. The attribute changes the standard definition whether the variables of the associated declaration block will be initialized for the transition to the RUN mode. It must be placed directly after the associated keyword of the following declaration blocks: • VAR_GLOBAL (in the interface and implementation section) • VAR (only for programs in a unit if the "Create program instance data only once" compiler option is active). It affects only the variables declared in the associated declaration block. See also Memory ranges of the variable types (Page 4844).	
	DISABLE	The variables declared in the associated declaration block are **not** initialized during the transition of the operating mode from STOP to RUN.
	ALWAYS	The variables declared in the associated declaration block are initialized in the transition of the mode from STOP to RUN.
	Default: Up to version V4.1 of the SIMOTION Kernel: DISABLE Version V4.2 and higher of the SIMOTION Kernel: In accordance with the setting on the device	

Note

Be sure to use the correct upper- and lower-case notation for attributes!

Note

The insertion, deletion or changing of the HMI_Export, BlockInit_OnChange or BlockInit_OnDeviceRun attributes in a declaration block does not change its version identification!

Table 7-483 Example of attributes for compiler options

```
INTERFACE
    VAR_GLOBAL
        { HMI_Export := FALSE;
          BlockInit_OnChange := TRUE; }
        // No HMI export, download in RUN possible
        x : DINT;
    END_VAR
    FUNCTION_BLOCK fb1;
END_INTERFACE
IMPLEMENTATION
    VAR_GLOBAL
        { HMI_Export := TRUE;
          BlockInit_OnDeviceRun := ALWAYS; }
        // HMI export, initialization for the STOP -> RUN transition
        y : DINT;
    END_VAR
    FUNCTION_BLOCK fb1
        VAR_INPUT
            i_var    : INT;
        END_VAR
        VAR_OUTPUT
            o_var    : INT;
        END_VAR
        { _U7_PoeBld_CompilerOption := warning:2:on; }
        o_var := REAL_TO_INT(1.0);    // Warning 16004
        { _U7_PoeBld_CompilerOption := warning:2:off; }
        o_var := REAL_TO_INT(1.0);    // No warning 16004
        { _U7_PoeBld_CompilerOption := warning:16004:on; }
        o_var := REAL_TO_INT(1.0);    // Warning 16004
        { _U7_PoeBld_CompilerOption := warning:16004:off; }
        o_var := REAL_TO_INT(1.0);    // No warning 16004
        { _U7_PoeBld_CompilerOption := warning:2:off;
          _U7_PoeBld_CompilerOption := warning:16004:on; }
        o_var := REAL_TO_INT(1.0);    // Warning 16004
    END_FUNCTION_BLOCK
END_IMPLEMENTATION
```

Issue non-assigned compiler message

Within a Pragma (Page 4917) you are able to prompt the computer to write non-assigned compiler messages to the "Compile/check output" tab of the detailed view.

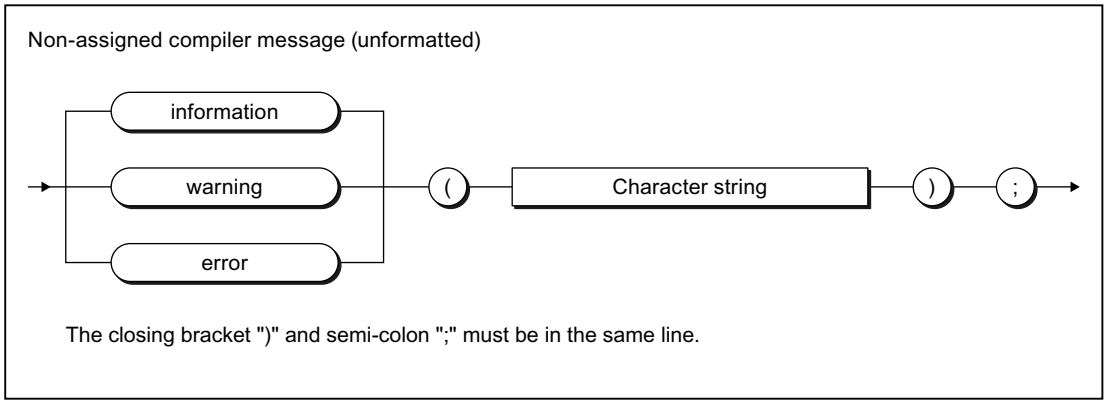


Figure 7-308 Syntax: Non-assigned compiler message

The keywords have the following meaning:

Keyword	Message number	Description
information	22002	Non-assigned information
warning	22001	Non-assigned warning
error	22000	Non-assigned error message

The text of the non-assigned message is stated after the keyword in brackets as a character string (Page 4667). A semi-colon completes the statement within the pragma.

Note

These pragma statements are not included when checking the backward compatibility of the project (Menu **Project > Old project format > Check the project for backward compatibility**).

The non-assigned compiler messages can e.g. be controlled with version formation by using preprocessor statements (Page 4919), see example below.

```
{ #ifdef test }
    ; // Statements
{ #else
    warning ('This message appears when "test" has not been defined');
#endif }
```

Pragmas for message output which are in hidden program sequences with the version formation are not taken into account.

7.2.7.8 SIMOTION devices

Rules for identifiers of the SIMOTION devices

General device identifiers

Identifiers for SIMOTION devices in the project navigator do not have to comply with the general Rules for identifiers (Page 4656).

All characters of the extended ASCII character set (ASCII code $20 to $7E, $80 to $FF) that can be displayed are permitted for the identifier of a SIMOTION device, except the following special characters:

- " (double inverted commas, ASCII code $22),
- & (Et character, ampersand, ASCII code $26),
- * (star, ASCII code $2A),
- : (colon, ASCII code $3A),
- < (less than character, ASCII code $3C),
- > (greater than character, ASCII code $3E),
- ? (question mark, ASCII code $3F),
- \ (backslash, ASCII code $5C),
- | (vertical line, ASCII code $7C).

Identifiers for the SIMOTION devices can only be specified in SIMOTION SCOUT or HW Config and only used in the programming languages.

Examples of valid device identifiers

- C240.2
- D455-2-DP (1)
- D445-2.PN-1

Device identifiers that do not comply with the general rules for identifiers can be used for SIMOTION devices of all versions (or all versions of the SIMOTION Kernel).

Note

Projects that contain device identifiers, which do not comply with the general Rules for identifiers (Page 4656), cannot be saved in the old project format (up to and including V4.2).

Device identifiers for PROFINET IO (name of station)

Device identifiers that are used as device names in PROFINET IO (name of station), must comply with the following- DNS conventions:

- Permissible lengths: 1 to 127 characters
- Organization using points "." is permissible in several labels; length of a single label: 1 to 63 characters
- Characters permitted within a label:
  - Letters "A" to "Z" and "a" to "z".
  - Numbers "0" to "9" (not at the beginning of the label)
  - Special character hyphen "-" (not at the beginning or end of a label)
  - Other special characters (such as accented characters, blank spaces, brackets, underscores) are not permitted.
- The following identifiers are not permitted for device names:
  - Identifiers that start with "port-xyz-" (x, y, z = 0 ... 9),
  - Identifiers of the form n.n.n.n (n = 0 ... 999).

Use of device identifiers in SIMOTION SCOUT

Identifiers for SIMOTION devices that do not comply with the general rules for identifiers **must** be enclosed in double inverted commas (", ASCII code $22) when used in SIMOTION SCOUT (e.g. in the programming languages).

Example:

- "D455-2 DP (1)".axis_1. motionStateData.actualVelocity
  Access to the system variable *motionStateData.actualVelocity* of the technology object *axis_1* on the device *D455-2 DP (1)*.

Note

Device identifiers that comply with the general Rules for identifiers (Page 4656) **can** also be enclosed in double inverted commas.

Example: The following notations are permitted for access to the system variable *motionStateData.motionState* of the technology object *axis_2* on the device *D435_2*:

- D435_2.axis_2.motionStateData.motionState
- "D435_2".axis_2.motionStateData.motionState

Making settings on the device (as of Kernel V4.2)

As of version V4.2 of the SIMOTION Kernel, you can make the following settings, among others, on the SIMOTION device:

- Memory area for the process image of the cyclic tasks and the fixed process image of the BackgroundTasks:
  - Separate memory areas for both process images - separate process image (Page 4885)
  - Common process image (Page 4883)
- Initialization of non-retentive global variables and program data during STOP-RUN transition
- Perform time synchronization with SINAMICS drive units
- Permit OPC-UA/XML for global device variables or I/O variables
  The symbol information of these variables is available in the SIMOTION device. This is necessary for the watch function of IT DIAG, for example.
- Restrict area for automatic address assignment during the message frame configuration (as of version V4.3 of the SIMOTION Kernel).
  When the checkbox is activated, the specified address area for the automatic message frame configuration is blocked. Specification of the blocked area has no effect when the message frames have already been configured.

You will find a detailed description in the online help.

Procedure

Settings on the device can be made as follows:

1. Select the SIMOTION device in the project navigator.
2. Select the **Edit > Object Properties** menu command.
3. Select the **Settings** tab.
4. Enter the settings.
5. Click **OK** to confirm.

7.2.7.9 Forward declarations

When creating the source file, you should always pay attention to the order of the source file modules. A module that is to be called must always precede the calling module so that the former is recognized by the latter.

For example, variables must be declared before they are used, functions must be defined before they are called, and function blocks must be defined prior to instance declaration.

Forward declaration for program organization units (POUs)

The compiler option (Page 4623) "Permit forward declarations" has the following effect:

- Declarations (e.g. global variable declarations) may be freely programmed between program organization units in the implementation section of an ST source file.
- The following statements within the TYPE / END_TYPE construct in the interface section (Page 4807) or the implementation section (Page 4809) of an ST source file are interpreted as prototypes of the relevant POU:
  - FUNCTION_BLOCK*fb-name*;
  - FUNCTION*fc-name*;
  - PROGRAM*prog-name*;
  - CLASS *class-name*; (as of version V4.5 of the SIMOTION Kernel and with compiler option (Page 4623) "Permit object-oriented programming" activated)

  See below for syntax diagram.
- After the corresponding POU prototype has been declared for a class or function block, the following declarations can be programmed before the relevant POU is defined:
  - Instance declaration of the function block (Page 4754) as a global variable or local variable in a program, in another function block or in a class
  - Instance declaration of the class (Page 4778) as a global variable or local variable in a program, in a function block or in another class
  - Use of the function block or class as a parameter in the definition of a function block, function or method

  Instance-specific initialization is not possible in this case.

  Note

  With instance-specific initialization (only when compiler option (Page 4623) "Permit object-oriented programming" is activated), the following applies:

  It is imperative that the function block or class is defined before the instance is declared or used. Declaration of the relevant POU prototype is not sufficient and therefore superfluous.

- All of the statement sections of program organization units are compiled in a second compiler run. For this reason, the following calls are possible regardless of the declaration position:
  - Call of a function block instance
  - Call of a function (Page 4753)
  - Call of a method
  - Call of a program within a program (Page 4762), provided that the other relevant requirements have been met.

  If a function or a program is called before it is implemented, the declaration of the prototype is optional; the call can still be carried out even if the prototype is not declared. Please note the additional relevant requirements when calling a program within a program.

POU prototypes

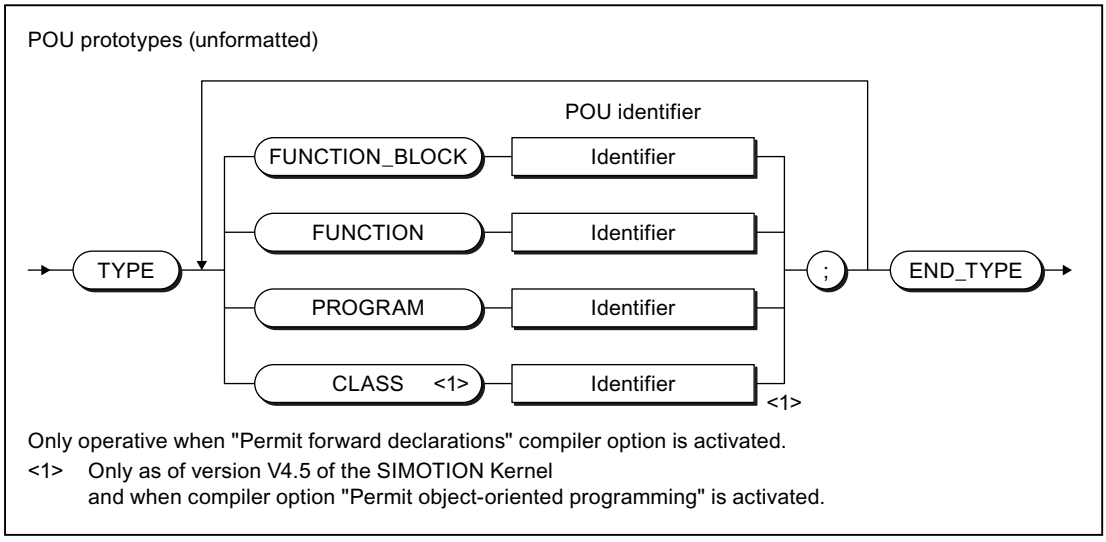


Figure 7-309 Syntax: POU prototypes

Note

If the POU prototypes are declared within the interface section, the corresponding POUs are declared public.

DECLARE PUBLIC statements for POUs within the interface section
(e.g. FUNCTION_BLOCK *fb-name;*) are also interpreted as prototypes.

If the "Permit forward declarations" compiler option (Page 4623) is not activated, the POU prototypes are ignored. Only the prototypes in the interface section are interpreted as DECLARE PUBLIC statements for the corresponding POU.

Example

Table 7-484 Sample program with forward declaration

```
(*
Only if the "Permit forward declarations" compiler option is activated.
*)

(*
The following compiler options also need to be activated due to a program
being called within the program:
"Permit language extensions" and "Only create program instance data once".
*)

INTERFACE
    PROGRAM prog_main;
END_INTERFACE
```

```
IMPLEMENTATION
    TYPE                        // POU prototypes
        FUNCTION_BLOCK fb_1; // Required for instance declaration
        FUNCTION fc_1;       // Optional
        PROGRAM prog_1;      // Optional
    END_TYPE
    // Instance declaration prior to implementation
    VAR_GLOBAL
        var_fb_1 : fb_1;
    END_VAR
    PROGRAM prog_main
        VAR
            var_1, var_2 : DINT;
            var_3, var_4 : INT;
        END_VAR

        var_fb_1 (x_in := var_3, x_out => var_4);

        // Call prior to implementation
        var_2 := fc_1 (var_1);
        prog_1();
    END_PROGRAM
    // Implementations

    FUNCTION_BLOCK fb_1
        VAR_INPUT
            x_in : INT;
        END_VAR

        VAR_OUTPUT
            x_out : INT;
        END_VAR

        x_out := x_in;
    END_FUNCTION_BLOCK
    FUNCTION fc_1: DINT
        VAR_INPUT
            x_in : DINT;
        END_VAR

        fc_1 := x_in;
    END_FUNCTION
    PROGRAM prog_1
        VAR
            var_int1, var_int2 : INT;
        END_VAR

        var_int1 := var_int2;
    END_PROGRAM
END_IMPLEMENTATION
```

7.2.7.10 Jump statement and label

As additional control statement (Page 4719), a jump statement is also available.

You program a jump statement (Page 4731) with the GOTO statement and specify the jump label to which you want to jump. Jumps are only permitted within a POU.

Enter the jump label (separated by a colon) in front of the statement at which you want the program to resume.

Alternatively, you can declare the jump labels in the POU (with the structure LABEL/END_LABEL in the POU). Only the declared jump labels can then be used in the statement section.

Syntax of jump statements and labels:

Example of syntax for jump statements

```
FUNCTION func : VOID
    VAR
        x, y, z : BOOL;
    END_VAR
    LABEL
        lab_1, lab_2;      // Declaration of the jump labels
    END_LABEL

    x := y;
  lab_1 : y := z;          // Jump label with statement
    IF x = y THEN
        GOTO lab_2;        // Jump statement
    END_IF;
    GOTO lab_1;            // Jump statement
  lab_2 : ;                // Jump label with blank statement
END_FUNCTION
```

Note

You should only use the GOTO statement in special circumstances (for example, for troubleshooting). It should not be used at all according to the rules for structured programming.

Jumps are only permitted within a POU.

The following jumps are illegal:

- Jumps to subordinate control structures (WHILE, FOR, etc.)
- Jumps from a WAITFORCONDITION structure
- Jumps within CASE statements

Jump labels can only be declared in the POU in which they are used. If jump labels are declared, only the declared jump labels may be used.

7.2.8 Error Sources and Program Debugging

This chapter describes various sources of programming errors and shows you how to program efficiently. You also learn what options are available for program testing. For all possible compilation error messages, i.e. compiler errors, see Compiler error messages and their remedies (Page 5040). Possible reactions and remedies are described for each error.

7.2.8.1 Notes on avoiding errors and on efficient programming

The SIMOTION *Basic Functions* Function Manual lists some common error sources, which hinder the compilers or prevent the proper execution of a program. There are notes on, e.g.:

- Data types for assigning arithmetic expressions
- Starting functions in cyclic tasks
- Wait times in cyclic tasks
- Errors on download
- CPU does not switch to RUN
- CPU goes to STOP
- Size of the local data stack
- etc.

In addition, you will also find notes on efficient programming there, particularly for

- runtime-oriented programming
- change-optimized programming

7.2.8.2 Program debugging

Syntax errors are detected and displayed by the ST compiler during the compilation procedure. Runtime errors in the execution of the program are displayed by system alarms or lead to the operating mode STOP. You can find logical programming errors with the test functions of ST, e.g. with the symbol browser, status program, trace.

To achieve the same results as shown below using the test functions, use of the sample program in Creating a sample program (Page 4643) is recommended.

Operating modes for program testing

Modes of the SIMOTION devices

Various SIMOTION device operating modes are available for program testing.

Table 7-485 Operating modes of a SIMOTION device

Operating mode	Meaning
Process mode	Program execution on the SIMOTION device is optimized for maximum system performance. The following diagnostic functions are available, although they may have only restricted functionality because of the optimization for maximum system performance: • Monitor variables in the symbol browser or a watch table • Program status (only restricted): – Restricted monitoring of variables (e.g. variables in loops, return values for system functions). – Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored. • Trace tool (only restricted) with measuring functions for drives and function generator, see online help: – Maximum of **1** trace on each SIMOTION device.
Test mode	The diagnostic functions of the process mode are available to the full extent: • Monitor variables in the symbol browser or a watch table • Program status: – Monitoring of all variables possible. – As of version V4.0 of the SIMOTION Kernel: Several program sources (e.g. ST source files, MCC units, LAD/FBD units)[1] can be monitored per task. – For version V3.2 of the SIMOTION Kernel: Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored per task. • Trace tool with measuring functions for drives and function generator, see online help: – Maximum of **4** traces on each SIMOTION device. In addition, the following diagnostics function is available: • Trace for monitoring the program execution in program branches which are executed cyclically (only for the MCC programming language and for SIMOTION Kernel V4.2 and higher). **Note** Runtime and memory utilization increase as the use of diagnostic functions increases.

Operating mode	Meaning
Debug mode	In addition to the diagnostic functions of the test mode, you can use the following functions: • Breakpoints Within a program source, you can set breakpoints (Page 4954). When an activated breakpoint is reached, selected tasks will be stopped. • Controlling MotionTasks On the "Task Manager" tab of the device diagnostics, you can use task control commands for MotionTasks; see the SIMOTION Basic Functions Function Manual. No more than 1 SIMOTION device of the project can be switched to debug mode. SIMOTION SCOUT is in online mode, i.e. connected to the target system. Observe the following section: Important information about the life-sign monitoring (Page 4937).

[1] Each with 1 MCC chart or 1 LAD/FBD program in a program source.

Selecting the operating mode

How to select the operating mode of a SIMOTION device:

1. Make sure a connection to the target system has been established (online mode).
2. Highlight the SIMOTION device in the project navigator.
3. Select the "Operating mode" context menu.
4. Select the required operating mode (see the table above).
   If you have selected "Debug mode":
   – Accept the safety information.
   – Parameterize the sign-of-life monitoring.

   Observe the following section: Important information about the life-sign monitoring (Page 4937).
5. Confirm with **OK**.
   The SIMOTION device switches to the selected operating mode (apart from with debug mode; see the explanation below).

Special features with debug mode

Debug mode can only be selected for one SIMOTION device.

If you have selected debug mode, only SIMOTION SCOUT switches to it; the SIMOTION device is in test mode.

- The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.
- The breakpoints toolbar is displayed.

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device.

The status bar indicates that debug mode is activated for the SIMOTION device.

Important information about the life-sign monitoring.

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode or a control panel only with the life-sign monitoring function activated with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

In the following cases, the SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection:

- In debug mode with activated breakpoints.
- When controlling an axis or a drive via the control panel (control priority at the PC):

If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis is brought to a standstill.
  - The enables are reset.

Accept safety notes

After selecting the debug mode or a control panel, you must accept the safety notes. You can set the parameters for the life-sign monitoring.

Proceed as follows:

1. Click the **Settings** button.
   The "Debug Settings" window opens.
2. Read there, as described in the following section, the safety notes and parameterize the life-sign monitoring.

Parameterizing the life-sign monitoring

In the "Life-Sign Monitoring Parameters" window, proceed as described below:

1. Read the warning!
2. Click the **Safety notes** button to open the window with the detailed safety notes.
3. Do not make any changes to the defaults for life-sign monitoring.
   Changes should only be made in special circumstances and in observance of all danger warnings.
4. Click **Accept** to confirm you have read the safety notes and have correctly parameterized the life-sign monitoring.

Note

The life-sign monitoring also responds in the following cases:

- Pressing the spacebar.
- Switching to a different Windows application.
- Too high a communication load between the SIMOTION device and SIMOTION SCOUT (e.g. by uploading task trace data).

The following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis or the drive is brought to a standstill.
  - The enables are reset.

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Life-sign monitoring parameters

Table 7-486 Life-sign monitoring parameter description

Field	Description
Life-sign monitoring	The SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection. If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered: • In debug mode for activated breakpoints: – The SIMOTION device switches to the STOP operating state. – The outputs are deactivated (ODIS). • For controlling an axis or a drive using the control panel (control priority for the PC): – The axis is brought to a standstill. – The enables are reset. The following parameterizations are possible: • Checkbox **active**: If the checkbox is activated, life-sign monitoring is active. The deactivation of the life-sign monitoring is not always possible. • Monitoring time: Enter the timeout. **Prudence** Do not make any changes to the defaults for life-sign monitoring, if possible. Changes should only be made in special circumstances and in observance of all danger warnings.
Safety information	**Please observe the warning!** Click the button to obtain further safety information. See: Important information about the life-sign monitoring (Page 4937)

Editing program sources in online mode

Online editing in process or test mode

If SIMOTION SCOUT is connected to a target system which is in the "process mode" or "test mode" operating mode, program sources (e.g. ST source files, MCC units with MCC charts) can generally be edited, compiled, and loaded to the target system in STOP operating mode. For information on downloading in RUN operating mode, see the corresponding section in the "SIMOTION Basic Functions" Function Manual.

However, you can only activate the "program status", "monitor program execution" (only for MCC), and trace (only for MCC) test functions for a program source or a program organization unit (POU) if the following conditions are met:

1. This program source or any POU of this source (e.g. MCC chart) does not contain any changes which have not been saved.
2. The program source (unit) in SCOUT is consistent with the target system.

Note

If the "program status" test function is activated, editing of the corresponding program source or one of its POUs is disabled.

If an MCC unit or MCC chart is changed and the "monitor program execution" or trace test functions are active for that unit or chart, the test functions are canceled.

Online editing in debug mode

If SIMOTION SCOUT is in debug mode, editing is possible as long as the SIMOTION device is not in debug mode, i.e. no breakpoints are activated.

You can only activate breakpoints and, as a result, switch the SIMOTION device to debug mode if the corresponding program source and all its POUs are saved, compiled so they are up to date, and consistent with the target system.

If you attempt to edit a program source or POU when the SIMOTION device is in debug mode, you are requested to deactivate all breakpoints and, as a result, to switch the SIMOTION device out of debug mode.

Note

If breakpoints have been activated and the SIMOTION device is in debug mode:

Entering a space switches the SIMOTION device to STOP operating mode and deactivates all outputs (ODIS).

Symbol Browser

Properties of the symbol browser

In the symbol browser, you can view and, if necessary, change the name, data type, and variable values. In particular, you can: see the following variables:

- Unit variables and static variables of a program or function block
- System variables of a SIMOTION device or a technology object
- I/O variables or global device variables.

For these variables, you can:

- View a snapshot of the variable values
- Monitor variable values as they change
- Change (modify) variable values

However, the symbol browser can only display/modify the variable values if the project has been loaded in the target system and a connection to the target system has been established.

Using the symbol browser

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system. To load the project with the sample program, see "Executing the sample program (Page 4650)".
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

The procedure depends on the memory area in which the variables to be monitored are stored.

Procedure

Proceed as follows:

1. Select the appropriate element in the project navigator in accordance with the following table.
2. In the detail view, click the **Symbol browser** tab.
   The corresponding variables are displayed in the symbol browser.
3. Select how each variable in the "Display format" column should be displayed.

Table 7-487 Elements in the project navigator and variables to be monitored in the symbol browser

Variables to be monitored in the symbol browser	Element to be selected in the project navigator
For variables in the user memory of the unit or in the retentive memory, refer to Memory areas of the variable types (Page 4844): • Retentive and non-retentive unit variables of the interface section of a program source (unit) • Retentive and non-retentive unit variables of the implementation section of a program source (unit) • Static variables of the function blocks whose instances are declared as unit variables. • In addition, if the program source (unit) has been compiled with the "Only create program instance data once" compiler option (Page 4623): – Static variables of the programs. – Static variables of the function blocks whose instances are declared as static variables of programs.	Program source (unit)
Variables in the user memory of the task, refer to Memory areas of the variable types (Page 4844): If the program source (unit) was compiled without the "Only create program instance data once" (default) compiler option (Page 4623), the user memory of the task to which the program was assigned contains the following variables: • Static variables of the programs. • Static variables of the function blocks whose instances are declared as static variables of programs.	EXECUTION SYSTEM
System variables of a SIMOTION device	SIMOTION device
System variables of a technology object	Instance of the technology object
Global device variables	GLOBAL DEVICE VARIABLES
I/O variables (in the Address list tab of the detail view). The Address list tab of the detail view can be opened by double-clicking the ADDRESS LIST element in the project navigator.	ADDRESS LIST

Note

You can monitor temporary variables (together with unit variables and static variables) with **Program status** (see Properties of the program status (Page 4949)).

Note

Trace diagnostic function for MCC programming

Various internal variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. These variables are displayed in the symbol browser.

With activated diagnostic function, these variables are used for the control of the diagnostics function. These variables must not be used in the user program.

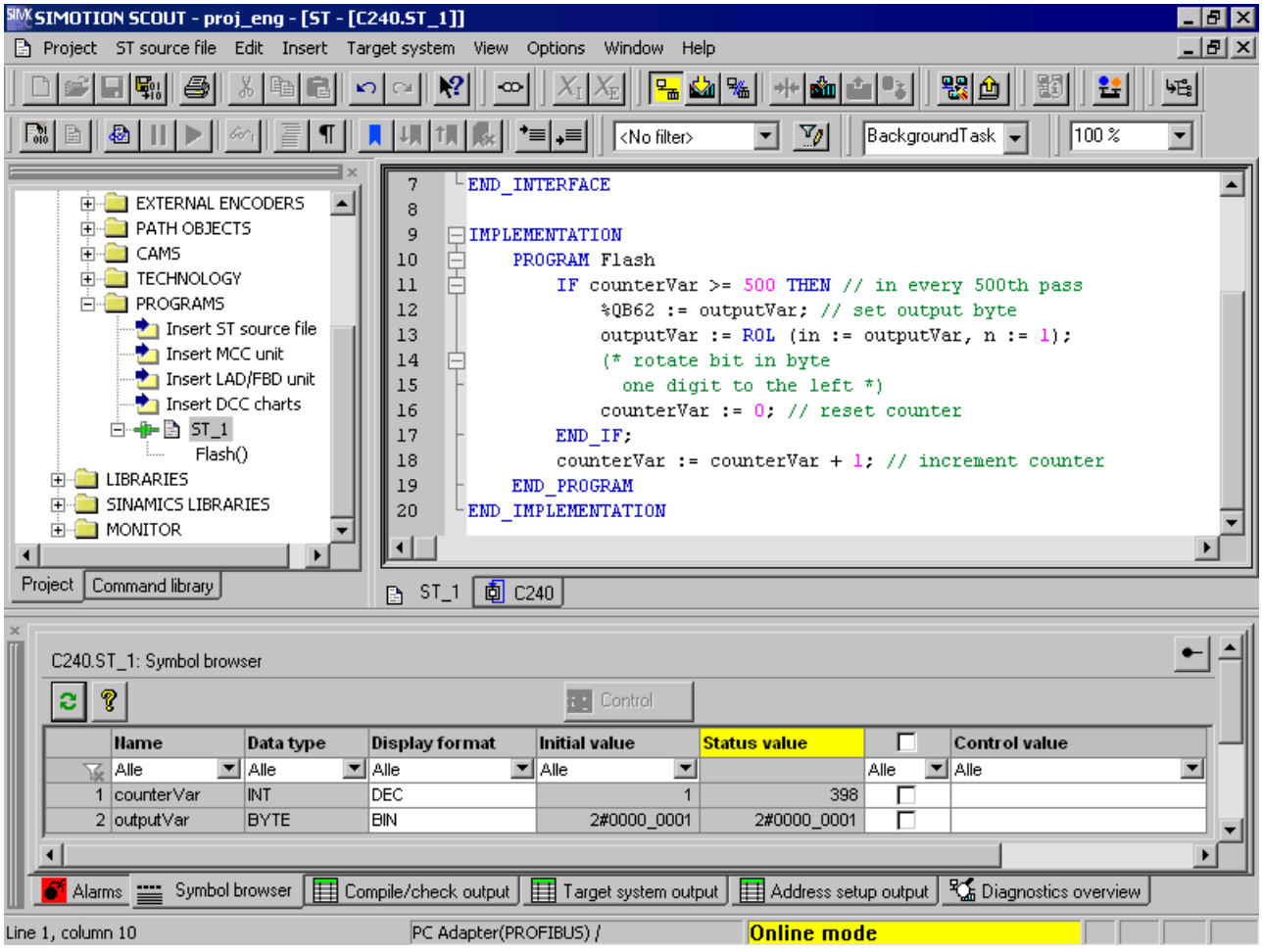


Figure 7-310 Viewing variables in the symbol browser

Status and controlling variables

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox adjacent to this column
3. Click the **Control** button.

The values you entered are written to the selected variables.

WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the symbol browser. There is therefore no guarantee of consistency.

Working with the symbol browser

The functions of the symbol browser and how you work with them are described in detail in the online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the symbol browser (independently of the SIMOTION device):

Table 7-488 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

Monitoring variables in watch table

Variables in the watch table

With the symbol browser you see only the variables of an object within the project. With program status you see only the variables of an ST source file within a freely selectable monitoring area.

With watch tables, in contrast, you can monitor selected variables from different sources as a group (e.g. program sources, technology objects, SINAMICS drives - even on different devices).

You can see the data type of the variables in offline mode. You can view and modify the value of the variables in online mode.

Using watch tables

You can group variables from various program sources, technology objects, SIMOTION devices, etc. (even on different devices), in a watch table where you can monitor them together and, if necessary, change them.

Creating a watch table

Procedure for creating a watch table and assigning variables:

1. In the Project navigator, select the **MONITOR** folder.
2. Select **Insert > Watch table** to create a watch table, and enter the name of the watch table. A watch table with this name appears in the **MONITOR** folder.
3. In the project navigator, click the object from which you want to move variables to the watch table.
4. In the symbol browser, select the corresponding variable line by clicking its number in the left column.
5. From the context menu, select **Add to watch table** and the appropriate watch table, e.g. **Watch table_1**.
6. If you click the watch table, you will see in the detail view of the **Watch table** tab that the selected variable is now in the watch table.
7. Repeat steps 3 to 6 to monitor the variables of various objects.

Status and controlling variables

If you are connected to the target system, you can monitor the variable contents.

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox in this column
3. Click the **Immediate control** button.

The values you entered are written to the selected variables.

⚠ WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the watch table. There is therefore no guarantee of consistency.

Working with the watch table

The functions of the symbol browser and how you work with them are described in detail in the online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the watch table (independently of the SIMOTION device):

Table 7-489 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN - Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

Variable status

"Variable status" enables you to monitor the current value for an individual variable, selected using the cursor, in an open program source or program organization unit (e.g. ST source file, MCC chart, LAD program).

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system. For information on loading a project, see "Running the sample program (Page 4650)".
- The program source containing the program organization unit (POE) whose variables you want to monitor must be consistent with the target system.
- The associated source (e.g. ST source file, MCC chart, LAD program) must be open.
- With the MCC programming language only: The parameter screen form for the command in which the variable you want to monitor is being used must be open.
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

Procedure

To monitor an individual variable using variable status:

1. Position the cursor above the identifier for a variable.
   - With the ST programming language: in the open ST source file
   - With the ST programming language: within an input field in the open parameter screen form
   - With the LAD/FBD programming language: within a network of the LAD/FBD program
2. Briefly position the cursor above the identifier.

The tool tip shows the current value of the variable. If you keep the cursor above the identifier for a longer period, the value is updated on an ongoing basis.

Note

With "variable status", the current value for the variable is displayed, wherever the selected variable is being used.

The "variable status" function enables you to monitor all those variables you are also able to monitor in the symbol browser (Page 4941) or the address list. These are:

- System variables of SIMOTION devices
- System variables of technology objects
- Global device variables
- Retentive and non-retentive unit variables of the interface section of a program source (unit)
- Retentive and non-retentive unit variables of the implementation section of a program source (unit)
- Static variables of the programs
- Static variables of the function blocks whose instances are declared as unit variables
- Static variables of the function blocks whose instances are declared as static variables of programs
- I/O variables

Program run

Program run: Display code location and call path

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path.

Follow these steps:

1. Click the **Show program run** button on the Program run toolbar.
   The "Program run call stack (Page 4948)" window opens.
2. Select the desired MotionTask.
3. Click the **Update** button.

The window shows:

- The position in the code being executed (e.g. line of the ST source file) stating the program source and the POU.
- Recursively positions in the code of other POUs that call the code position being executed.

The following names are displayed for the SIMOTION RT program sources:

Table 7-490 SIMOTION RT program sources

Name	Meaning
taskbind.hid	Execution system
stdfunc.pck	IEC library
device.pck	Device-specific library
tp-name.pck	Library of the *tp-name* technology package, e.g. cam.pck for the library of the CAM technology package

Program run parameters

You can display the following for all configured tasks:

- the current code position in the program code (e.g. line of an ST source file)
- the call path of this code position

Table 7-491 Program run parameter description

Array	Description
Selected CPU	The selected SIMOTION device is displayed.
Refresh	Clicking the button reads the current code positions from the SIMOTION device and shows them in the open window.
Calling task	Select the task for which you want to determine the code position being executed. All configured tasks of the execution system.
Current code position	The position being executed in the program code (e.g. line of an ST source file) is displayed (with the name of the program source, line number, name of the POU).
is called by	The code positions that call the code position being executed within the selected task are shown recursively (with the name of the program source, line number, name of the POU, and name of the function block instance, if applicable).

For names of the SIMOTION RT program sources, refer to the table in Program run (Page 4947).

Program run toolbar

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path with this toolbar.

Table 7-492 Program run toolbar

Symbol	Meaning
	Display program run Click this symbol to open the Program run call stack window. In this window, you can display the currently active code position with its call path. See: Program run: Display code position and call path (Page 4947)

Program status

Properties of the program status

Status program enables monitoring the variable values accurately to the cycle during program execution.

You can select a monitoring area in the ST source file and monitor, in addition to global and static local variables, also temporary local variables (e.g. within a function) there.

The values of the following variables are displayed:

- Simple data type variables (INT, REAL, etc.)
- Individual elements of a structure, provided an assignment is made
- Individual elements of an array, provided an assignment is made
- Enumeration data type variables

Return values of the system function _trcVal are also displayed. This way interim results can be displayed in expressions. For the _trcVal function please refer to the "SIMOTION Basic Functions Function Manual (Page 1560)".

Note

The values of constants are not displayed.

Due to the restricted buffer capacity and the requirement for minimum runtime corruption, the following variables cannot be displayed:

- Complete arrays
- Complete structures

Individual array elements or individual structure elements are displayed, however, provided an assignment is made in the ST source file.

Method of operation of program status

On the SIMOTION device:

- While the selected monitoring range is running in the ST source file, the buffer for the variables to be monitored is filled with the corresponding values.
  The recording of the values depends on the operating mode (Page 4935) (process mode or test mode) of the SIMOTION device, see following table.
- Only after leaving the selected monitoring range or abort of the recording is the buffer displayed in SIMOTION SCOUT.

SIMOTION SCOUT calls the values which have been made available at regular intervals and displays them in the format you selected in the ST editor settings (Page 4593) under "Format for status display".

In the case of functions and function blocks, you can select a location in an ST source file where a function or instance of a function block is called (call path). This enables you to observe the variable values specifically for this call.

Table 7-493 Differences between process mode and test mode in Program Status

	Process mode	**Test mode**
Optimization of program execution	For maximum system performance, only restricted diagnostics are possible.	For full diagnosis options
Maximum number of monitored program sources (e.g. ST source files, MCC units, LAD/FBD units)	Maximum of 1 program source[1]	• As of version V4.0 of the SIMOTION Kernel: Multiple program sources[1] per task • For version V3.2 of the SIMOTION Kernel: Maximum of 1 program source[1] per task
Loops (e.g. WHILE, REPEAT, FOR)	The recording is interrupted on the first repeat loop. The values are provided. Therefore, the following applies for completely selected loops: • The appropriate values are displayed after the first run of the loop. • Changes or the values are not displayed during further runs.	If there are repeats, the recording continues correctly. Therefore, the following applies for completely selected loops: • As long as the loop is being run through, no values are displayed. • Only after leaving the monitoring range are the values displayed on the last run of the loop.
System functions that contain internal loops (e.g. functions for processing strings)	The recording is interrupted during the execution of the system function. Values are not correctly displayed in some cases.	The recording is performed correctly during the call of the system function. Values are displayed correctly.

[1] Each with 1 MCC chart or 1 LAD/FBD program in a program source

Note

Program status requires additional CPU resources.

Please note if you want to monitor several programs at the same time with the status program:

- Test mode must be activated (see Operating modes of the SIMOTION devices (Page 4935)).
- In version V3.2 of the SIMOTION Kernel, the programs must be assigned to various tasks.

Using the status program

Before you can work with the Status program, you must instruct the system to run in a special mode:

1. Make sure that the ST source file generates the additional debug code during compilation:
   - Select the ST source file in the project navigator and select the **Edit > Object properties** menu command.
   - Select the **Compiler** tab to change the local settings of the compiler (Page 4626).
   - Make sure that the **Enable Status program** checkbox is activated and confirm with **OK**. You can also change this compiler option at global settings of the compiler (Page 4623).
2. Open the ST source file and recompile it with **ST source file > Accept and compile**.
3. Download and start the program in the usual way.
4. Click the button for **program status** in the ST editor toolbar (Page 4614) to start this test mode.

The ST editor window is now divided vertically:

- In the left-hand pane of the window, you can see the ST source file. You can select an area here.
- The right-hand pane of the window displays the variables for the selected area and their values.

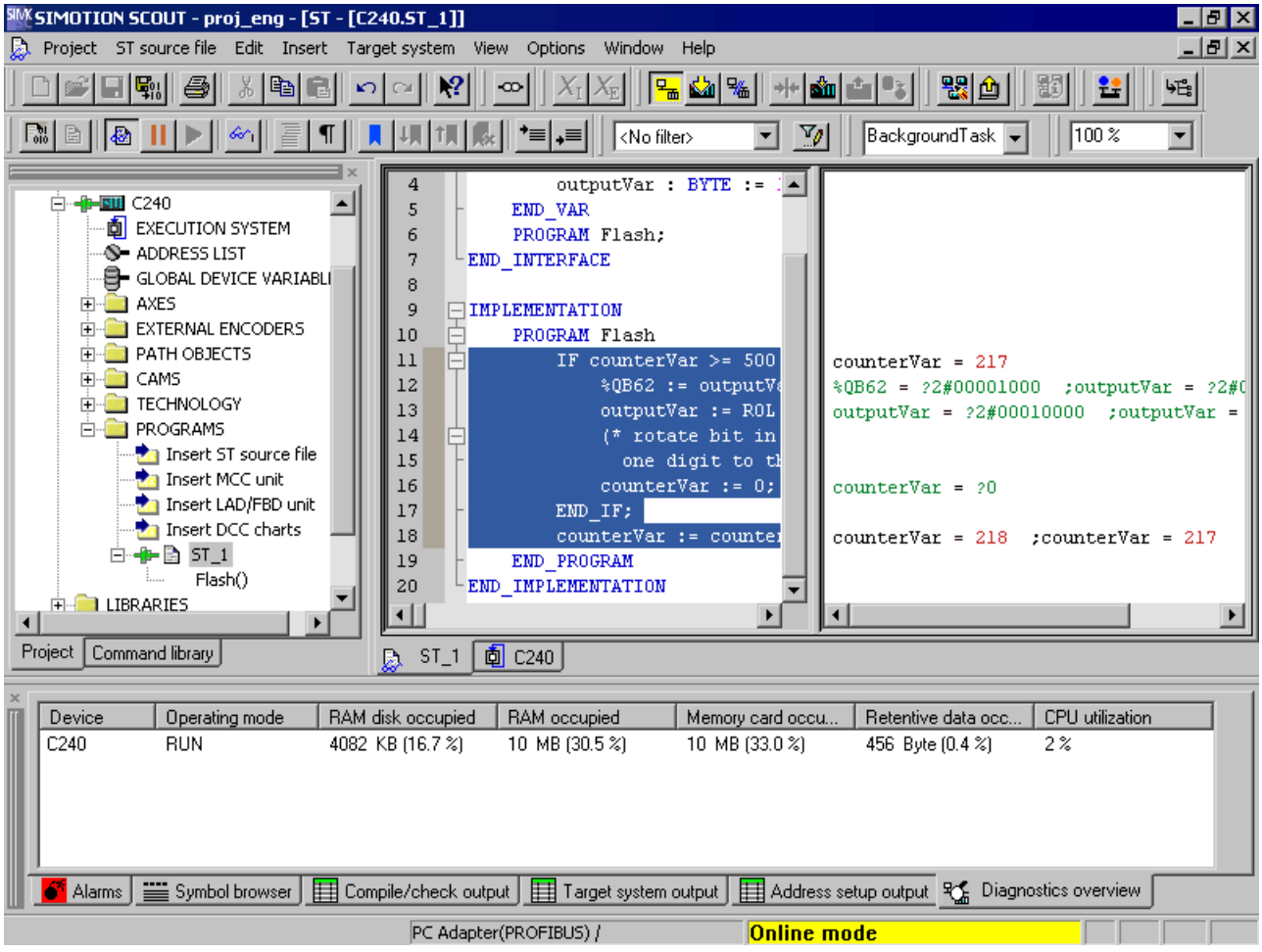


Figure 7-311 Part of an ST program in program status test mode

Follow the procedure below to test with program status:

1. In the left-hand pane, select the section of the ST source file you want to test.
2. If you have selected a section of a POU that is called by several positions in a program source file or several tasks:
   Enter the call path for program status (Page 4953).

The right-hand pane of the window displays the variables for the selected area and their values. The display of the variable values is updated cyclically. Variables with bit data type are updated cyclically and correspond to the format you selected in the ST editor settings (Page 4593) under "Format for status display":

- Values that have changed in the current pass are displayed in **red**.
- Values that have not changed are displayed in **black**.
- Variables without values, e.g. variables in an unused IF branch are shown in **green** and marked with a question mark.

If the display of the variable values changes too fast:

- Click the  button for **Stop monitoring of program variables** in the ST editor toolbar (Page 4614) to stop the display.
- Click the  button for **Continue monitoring of program variables** in the ST editor toolbar (Page 4614) to continue the display.

You can force the update of the displayed values:

- Click the  button for **Update** on the ST editor toolbar (Page 4614).
  The buffer of the SIMOTION device is read, even if the selected monitoring range has not yet been completely processed and the values are incomplete. This can be useful, for example, if the program is waiting for a WAITFORCONDITION statement.
  The monitoring of the program variables must have been activated.

Call path for program status

You can specify the call path when monitoring variable values of functions and function blocks. This enables you to observe the variable values specifically for this call.

For this purpose, the **Call path** window automatically opens in the following cases:

- You have selected a section of a function:
  The function is called at various points in the program source files (e.g. ST source files) of the SIMOTION device.
- You have selected a section of a function block:
  There are several instances of the function block or the instance is called at various points in the program source files (e.g. ST source files) of the SIMOTION device.
- You have selected a section of a program:
  The program is assigned to more than one task.

How to select the call path:

In the **Call path status program** window, the marked section of the POU (code position) is displayed (with the name of the ST source file, line number, name of the POU).

1. If the code position is called in several tasks:
   – Select the task.
2. Select the code position to be called (in the calling POU).
   You can select from the following:
   – The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
   If the selected calling code position is in turn called by several code positions, further lines are displayed in which you proceed similarly.
   – **All:**
   All displayed code positions are selected. Moreover, all code positions (up to the top level of the hierarchy) are selected from which the displayed code positions are called.

Parameter call path status program

Table 7-494 Program status call path parameter description

Field	Description
Calling task	Select the task. All tasks in which the selected code position is called are available for selection.
Current code position	The selected section of the POU (code position) is shown (with the name of the ST source file, line number, name of the POU)
is called by	Select the calling code position. The following are available: • The code positions to be called within the selected task (with the name of the program source, line number, name of the POU). If the selected calling code position is in turn called by several code positions, further lines are displayed in which you proceed similarly. • **All:** All displayed code positions are selected. Moreover, all code positions (up to the top level of the hierarchy) are selected from which the displayed code positions are called.

Breakpoints

General procedure for setting breakpoints

You can set breakpoints within a POU of a program source (e.g. ST source, MCC chart, LAD/FBD source). On reaching an activated breakpoint, the task in which the POU with the breakpoint is called is stopped. If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser.

Requirement:

- The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.

Procedure

Follow these steps:

1. Select "Debug mode" for the associated SIMOTION device; see Setting debug mode (Page 4955).
2. Specify the tasks to be stopped, see Specifying the debug task group (Page 4956).
3. Set breakpoints, see Setting breakpoints (Page 4959).
4. Define the call path, see Defining a call path for a single breakpoint (Page 4962).
5. Activate the breakpoints, see Activating breakpoints (Page 4967).

Setting the debug mode

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode only with activated life-sign monitoring (Page 4937) with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect!

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Requirement

1. A connection to the target system must have been established (online mode)
2. Debug mode must not be selected for any SIMOTION device.

Procedure

To set the debug mode, proceed as follows:

1. Highlight the SIMOTION device in the project navigator.
2. Select **Operating mode** from the context menu.
3. Select **Debug** mode (Page 4935).
4. Accept the safety information
5. Parameterize the sign-of-life monitoring.
   See also section: Important information about the life-sign monitoring (Page 4937).
6. Confirm with **OK**.

SIMOTION SCOUT switches to debug mode for this device; the SIMOTION device itself remains in "test mode", as long as at least one breakpoint is activated:

The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.

The breakpoints toolbar (Page 4961) is displayed.

As long as no breakpoints are activated, you can edit program sources in debug mode (Page 4939).

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device. The status bar indicates that debug mode is activated for the SIMOTION device.

Note

Pressing the spacebar or switching to a different Windows application causes the following to happen if the SIMOTION device is in debug mode (breakpoints activated):

- The SIMOTION device switches to the STOP operating state.
- The outputs are deactivated (ODIS).

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Define the debug task group

On reaching an activated breakpoint, all tasks that are assigned to the debug task group are stopped.

Requirement

1. A connection to the target system must have been established (online mode).
2. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4955).

Procedure

How to assign a task to the debug task group:

1. Highlight the relevant SIMOTION device in the project navigator.
2. Select **Debug task group** from the context menu.
   The Debug Task group window opens.
3. Select the tasks to be stopped on reaching the breakpoint:
   - If you only want to stop individual tasks (in RUN operating state): Activate the **Debug task group** selection option.
     Assign all tasks to be stopped on reaching a breakpoint to the **Tasks to be stopped** list.
   - If you only want to stop individual tasks (in HOLD operating state): Activate the **All tasks** selection option.
     In this case, also select whether the outputs and technology objects are to be released again after resumption of program execution.

Note

Note the different behavior when an activated breakpoint is reached, see the following table.

Table 7-495 Behavior at the breakpoint depending on the tasks to be stopped in the debug task group.

Property		Tasks to be stopped	
		Single selected tasks (debug task group)	**All tasks**
Behavior on reaching the breakpoint			
	Operating state	RUN	STOP
	Stopped tasks	Only tasks in the debug task group	All tasks
	Outputs	Active	Deactivated (ODIS activated)
	Technology	Closed-loop control active	No closed-loop control (ODIS activated)
	Runtime measurement of the tasks	Active for all tasks	Deactivated for all tasks
	Time monitoring of the tasks	Deactivated for tasks in the debug task group	Deactivated for all tasks
	Real-time clock	Continues to run	Continues to run
Behavior on resumption of program execution			
	Operating state	RUN	RUN
	Started tasks	All tasks in the debug task group	All tasks
	Outputs	Active	The behavior of the outputs and the technology objects depends on the **'Continue' activates the outputs (ODIS deactivated)** checkbox. • Active: ODIS will be deactivated. All outputs and technology objects are released. • Inactive: ODIS remains activated. All outputs and technology objects are only enabled for one STOP-RUN transition.
	Technology	Closed-loop control active	

Note

You can only make changes to the debug task group if no breakpoints are active.

The settings of the debug task group are retained after exiting "Debug mode".

Proceed as follows:

1. Set breakpoints (see Setting breakpoints (Page 4959)).
2. Define the call path (see Defining a call path for a single breakpoint (Page 4962)).
3. Activate the breakpoints (see Activating breakpoints (Page 4967)).

Debug task group parameters

Use this window to define the debug task group. On reaching an activated breakpoint, all tasks that are assigned to the debug task group are stopped.

This requires that the relevant SIMOTION device is in debug mode, see Modes of the SIMOTION devices (Page 4935).

Table 7-496 Debug settings parameter description

Field	Description
Debug task group	Select this selection option if you only want to stop individual tasks. The SIMOTION device remains in RUN mode after an activated breakpoint is reached. Outputs and technology objects remain activated. Assign all tasks to be stopped on reaching a breakpoint to the **Tasks to be stopped** list.
All tasks	Select this selection option if you only want to stop all user tasks. The SIMOTION device remains in STOP mode after an activated breakpoint is reached, all outputs and technology objects will be deactivated (ODIS activated). In this case, also select whether the outputs and technology objects are to be released again after resumption of program execution.
'Resume' activates the outputs (ODIS deactivated).	Only if **All tasks** is selected. Activate the checkbox, to release again the outputs and technology objects after program execution has been resumed. All outputs and technology objects can only be released after a download of the project with deactivated checkbox.

Note

Note the different behavior at the activated breakpoint depending on the tasks to be stopped, see table in Define the debug task group (Page 4956).

You can only make changes to the debug task group if no breakpoints are active.

Debug table parameter

The debug table shows all breakpoints in the program sources of a SIMOTION device.

Table 7-497 Debug table parameter description

Field		Description
Debug points (table)		
	Active	The activation state of the corresponding breakpoint is displayed and can be modified: **Active**: The breakpoint is activated. **Inactive**: The breakpoint is deactivated. See: Activating breakpoints (Page 4967).
	Source, line (POU)	The code position is shown with the set breakpoint (with the name of the program source file, line number, name of the POU).

Field		Description
	Call path	Click the button to define the call path for the breakpoint. See: Defining the call path for a single breakpoint (Page 4962).
All breakpoints ...		
	Activate	Click the button to activate all breakpoints (in all program sources) of the SIMOTION device. See: Activating breakpoints (Page 4967).
	Deactivate	Click the button to deactivate all breakpoints (in all program sources) of the SIMOTION device. See: Activating breakpoints (Page 4967).
	Delete	Click the button to clear all breakpoints (in all program sources) of the SIMOTION device. See: Setting breakpoints (Page 4959).

Setting breakpoints

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4955).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4956).

Procedure

How to set a breakpoint:

1. Select the code location where no breakpoint has been set:
   - SIMOTION ST: Place the cursor on a line in the ST source file that contains a statement.
   - SIMOTION MCC: Select an MCC command in the MCC chart (except module or comment block).
   - SIMOTION LAD/FBD: Set the cursor in a network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove a breakpoint, proceed as follows:

1. Select the code position with the breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Remove all breakpoints** menu command (shortcut CTRL+F5).
  - Click the  button in the Breakpoints toolbar.

Note

You cannot set breakpoints:

- For SIMOTION ST: In lines that contain only comment.
- For SIMOTION MCC: On the module or comment block commands.
- For SIMOTION LAD/FBD: Within a network.
- At code locations in which other debug points (e.g. trigger points) have been set.

You can list the debug points in all program sources of the SIMOTION device in the debug table:

- Click the  button for "debug table" in the Breakpoints toolbar.

In the debug table, you can also remove all breakpoints (in all program sources) of the SIMOTION device:

- Click the button for "Clear all breakpoints".

The breakpoints set also remain saved after leaving debug mode; they are displayed in debug mode only.

You can use the program status (Page 4951) diagnosis functions and breakpoints together in a program source or POU. However, the following restrictions apply depending on the program languages:

- SIMOTION ST: For version V3.2 of the SIMOTION Kernel, the (marked) ST source file lines to be tested with program status must not contain a breakpoint.
- SIMOTION MCC and LAD/FBD: The commands of the MCC chart (or networks of the LAD/FBD program) to be tested with program status must not contain a breakpoint.

Proceed as follows

1. Define the call path, see Defining a call path for a single breakpoint (Page 4962).
2. Activate the breakpoints, see Activating breakpoints (Page 4967).

Breakpoints toolbar

This toolbar contains important operator actions for setting and activating breakpoints:

Table 7-498 Breakpoints toolbar

Symbol	Meaning
	Set/remove breakpoint Click this icon to set at breakpoint for the selected code position or to remove an existing breakpoint. See: Setting breakpoints (Page 4959).
	Activate/deactivate breakpoint Click this icon to activate or deactivate the breakpoint at the selected code position. See: Activating breakpoints (Page 4967).
	Edit the call path Click this icon to define the call path for the breakpoints: • If a code position **with** breakpoint is selected: The call path for this breakpoint. • If a code position **without** breakpoint is selected: The call path for all breakpoints of the POU. See: Defining the call path for a single breakpoint (Page 4962), Defining the call path for all breakpoints (Page 4964).
	Activate all breakpoints of the active POU Click this symbol to activate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 4967).
	Deactivate all breakpoints of the active POU Click this symbol to deactivate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 4967).
	Remove all breakpoints of the active POU Click this symbol to remove all breakpoints from the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Setting breakpoints (Page 4959).
	Debug table Click this icon to display the debug table. See: Debug table parameters (Page 4958).
	Display call stack Click this icon after reaching an activated breakpoint to: • View the call path at the current breakpoint. • View the code positions at which the other tasks of the debug task group have been stopped together with their call path. See: Displaying the call stack (Page 4970).
	Resume Click this icon to continue the program execution after reaching an activated breakpoint. See: Resuming program execution (Page 4971), Displaying the call stack (Page 4970).

Symbol	Meaning
	Next step (SIMOTION Kernel as of version V4.4) Only available for the MCC and LAD/FBD programming languages: Click this icon to resume the program execution until the next MCC command or LAD/FBD network is reached. See: Resume program execution in single steps.
	Step through the subprogram (SIMOTION Kernel as of version V4.4) Only available for the MCC programming language. Click this icon to jump to the called subprogram and stop at the first command. The subprogram must be created in the MCC or LAD/FBD programming language. See: Resume program execution in single steps.

Defining the call path for a single breakpoint

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4955).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4956).
5. Breakpoint is set, see Setting breakpoints (Page 4959).

Procedure

To define the call path for a single breakpoint, proceed as follows:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the Call path / task selection breakpoint window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when the selected breakpoint is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when the activated breakpoint in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when the breakpoint in the selected task is reached. The task must be in the debug task group.
     The specification of a call path is possible.

4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at the activated breakpoint if the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at the activated breakpoint only when the POU is called from the selected code position.
     If the POU of the selected calling code position is also called from other code positions, further lines are displayed successively in which you proceed similarly.
5. If the breakpoint is only to be activated after the code position has been reached several times, select the number of times.

Note

You can also define the call path to the individual breakpoints in the debug table:

1. Click the  button for "debug table" in the Breakpoints toolbar.
   The "Debug table" window opens.
2. Click the appropriate button in the "Call path" column.
3. Proceed in the same way as described above:
   - Specify the task.
   - Define the call path (only for functions and function blocks).
   - Specify the number of passes after which the breakpoint is to be activated.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 4967).

Note

You can use the "Display call stack (Page 4970)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for all breakpoints (Page 4964)

Breakpoint call path / task selection parameters

Table 7-499 Breakpoint call path / task selection parameter description

Field	Description
Selected CPU	The selected SIMOTION device is displayed.
Calling task	Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when the selected breakpoint is reached. The following are available: • **All calling locations starting at this call level** The user program will always be started when the activated breakpoint in any task of the debug task group is reached. • The individual tasks from which the POU with the selected breakpoint can be reached. The user program will be stopped only when the breakpoint in the selected task is reached. The task must be in the debug task group. The specification of a call path is possible.
Current code position	The code position is shown with the set breakpoint (with the name of the program source file, line number, name of the POU).
Is called by	Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU). The following are available: • **All calling locations starting at this call level** **No** call path is specified. The user program will always be stopped at the activated breakpoint when the POU in the tasks is reached. • Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU). The call path is specified. The user program will be stopped at the activated breakpoint only when the POU is called from the selected code position. If the POU of the selected calling code position is also called from other code positions, further lines are displayed successively in which you proceed similarly.
The breakpoint will be activated at each nth pass.	If you do not want the breakpoint to be activated until the code position has been reached a certain number of times, set this number.

Note

You can only make changes to the debug task group if no breakpoints are active.

Defining the call path for all breakpoints

With this procedure, you can:

- Select a default setting for all future breakpoints in a POU (e.g. MCC chart, LAD/FBD program or POU in an ST source file).
- Accept and compare the call path for all previously set breakpoints in this POU.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4955).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4956).

Procedure

To define the call path for all future breakpoints of a POU, proceed as follows:

1. Select the code location where **no** breakpoint has been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the "Call path / task selection all breakpoints for each POU" window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when a breakpoint in this POU is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when an activated breakpoint of the POU in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when a breakpoint in the selected task is reached.
     The task must be in the debug task group.
     The specification of a call path is possible.
4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at an activated breakpoint when the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at an activated breakpoint only when the POU is called from the selected code position.
     If the selected calling code position is in turn called by other code positions, further lines are displayed successively in which you proceed similarly.
5. If a breakpoint is only to be activated after the code position has been reached several times, select the number of times.
6. If you want to accept and compare this call path for all previously set breakpoints in this POU:
   - Click **Accept**.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 4967).

Note

You can use the "Display call stack (Page 4970)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for a single breakpoint (Page 4962)

Call path / task selection parameters of all breakpoints per POU

Here you can define a presetting for the call path of all future breakpoints to be set in a POU. Moreover, you can also accept this setting for all previously set breakpoints of this POU.

Table 7-500 Call path / task selection parameter description of all breakpoints per POU

Field	Description
Selected CPU	The selected SIMOTION device is displayed.
Calling task	Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when a breakpoint in this POU is reached. The following are available: • **All calling locations starting at this call level** The user program will always be started when an activated breakpoint of the POU in any task of the debug task group is reached. • The individual tasks from which the selected breakpoint can be reached. The user program will be stopped only when an activated breakpoint in the selected task is reached. The task must be in the debug task group. The specification of a call path is possible.
Current POU	The POU in which the cursor is located is displayed (with the name of the program source file, name of the POU).

Field	Description
Is called by	Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU). The following are available: • **All calling locations starting at this call level** **No** call path is specified. The user program will always be stopped at an activated breakpoint when the POU in the selected tasks is called. • Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU). The call path is specified. The user program will be stopped at an activated breakpoint only when the POU is called from the selected code position. If the POU of the selected calling code position is also called from other code positions, further lines are displayed successively in which you proceed similarly.
The breakpoint will be activated at each nth pass.	If you do not want the breakpoint to be activated until the code position has been reached a certain number of times, set this number.
Apply this call path to all previous breakpoints of this POU	Click the **Apply** button, if you want to apply the call path to all previously set breakpoints of the current POU. Any existing settings will be overwritten.

Activating breakpoints

Breakpoints must be activated if they are to have an effect on program execution.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 4955).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 4956).
5. Breakpoints are set, see Setting breakpoints (Page 4959).
6. Call paths are defined, see Defining a call path for a single breakpoint (Page 4962).

Activating breakpoints

How to activate a single breakpoint:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source file.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the  button in the Breakpoints toolbar.

To activate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Activate all breakpoints** menu command.
  - Click the  button in the Breakpoints toolbar.

Once the first breakpoint has been activated, the SIMOTION device switches to debug mode. It remains in this mode until the last breakpoint is deactivated.

In the Task status function bar, (Page 4971) the tasks with activated breakpoints are highlighted in gray ().

Note

Breakpoints of all program sources of the SIMOTION device can also be activated and deactivated in the debug table:

1. Click the  button for "debug table" in the Breakpoints toolbar.
   The "Debug Table" window opens.
2. Perform the action below, depending on which breakpoints you want to activate or deactivate:
   - Single breakpoints: Check or clear the corresponding checkboxes.
   - All breakpoints (in all program sources): Click the corresponding button.

The following applies up to version V4.3 of the SIMOTION Kernel:

- In the case of activated breakpoints, the "Single step" test function of the SIMOTION MCC programming language cannot be used.

The following applies as of version V4.4 of the SIMOTION Kernel:

- The "Single step" test function of the SIMOTION MCC programming language is not available in the Debug mode.

Breakpoints cannot be activate if the control priority is at the axis control panel. Conversely, you cannot fetch the control priority for the axis control panel when a breakpoint activated.

Behavior at the activated breakpoint

On reaching an activated breakpoint (possibly using the selected call path (Page 4962)), all tasks assigned to the debug task group will be stopped. The behavior depends on the tasks in the debug task group and is described in "Defining a debug task group (Page 4956)". The breakpoint is highlighted.

In the Task status function bar, (Page 4971) the task in which the breakpoint was reached is highlighted in red ( ).

The following applies to the programming languages MCC or LAD/FBD: If the debug task group is stopped by a breakpoint, then the user has the option to change to another task, belonging to the debug task group, in the combo box. Always the breakpoint of the currently selected task is visualized.

If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser (Page 4941).

You can use the "Display call stack (Page 4970)" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Resuming program execution

You can resume the execution of the stopped tasks, see "Resuming program execution" (Page 4971).

As of version V4.4 of the SIMOTION Kernel, you can resume the task in single steps that has been stopped at the activated breakpoint in the MCC and LAD/FBD programming languages, see Resuming program execution in single steps.

Deactivate breakpoints

To deactivate a single breakpoint, proceed as follows:

1. Select the code position with the activated breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the button in the Breakpoints toolbar.

To deactivate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Deactivate all breakpoints** menu command.
  - Click the button in the Breakpoints toolbar.

Once the last breakpoint has been deactivated, the SIMOTION device switches to "test mode"; SIMOTION SCOUT continues to run in debug mode.

Display call stack

You can use the "Display call stack" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Requirement

The user program is stopped at an activated breakpoint, i.e. the tasks of the debug task group (Page 4956) have been stopped.

Procedure

To call the "Display call stack" function, proceed as follows:

- Click the  button for "display call stack" in the Breakpoints toolbar.
  The "Breakpoint call stack" dialog opens. The current call path (including the calling task and the number of the set passes) is displayed.
  The call path cannot be changed.

To use the "Display call stack" function, proceed as follows:

1. Keep the "Breakpoint call stack" dialog open.
2. To display the code position at which the other task was stopped, proceed as follows:
   – Select the appropriate task. All tasks of the debug task group can be selected.

   The code position, including the call path, is displayed. If the code position is contained in a user program, the program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) will be opened and the code position marked.
3. How to resume program execution:
   – Click the  button for "resume" (Ctrl+F8 shortcut) on the Breakpoint toolbar.

   When the next activated breakpoint is reached, the tasks of the debug task group will be stopped again. The current call path, including the calling task, is displayed.
4. Click **OK** to close the "Breakpoint call stack" dialog box.

For names of the SIMOTION RT program sources, refer to the table in "Program run (Page 4947)".

Breakpoints call stack parameter

When an activated breakpoint (Page 4967) is reached, you can display the following for each task in the debug task group (Page 4956):

- The position in the program code (e.g. line of an ST source file) at which the task stopped.
- The call path of this code position.

Table 7-501 Breakpoint call path parameter description

Field	Description
Selected CPU	The selected SIMOTION device is displayed.
Calling task	Select the task for which you want to display the code position at which the task was stopped. All tasks of the debug task group can be selected.
Current code position	The position in the program code (e.g. line of an ST source file) at which the selected task was stopped is displayed (with the name of the program source file, line number, name of the POU).
is called by	The code positions that call the current code position within the selected task are shown recursively (with the name of the program source file, line number, name of the POU, and name of the function block instance, if applicable).

For names of the SIMOTION RT program sources, refer to the table in "Program run (Page 4947)".

Resuming program execution

How to resume program execution:

- Perform the following (alternatives):
  - Select the **Debug > Continue** menu command (shortcut CTRL+F8).
  - Click the  button on the Breakpoint toolbar (Page 4961) to "Continue".

The stopped task is continued until the next active breakpoint is reached.

Task status function bar

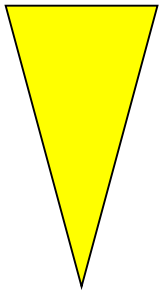
In a combo box, the Task status function bar displays all tasks of the active SIMOTION device, to which a program is assigned.

They are displayed under the following conditions:

1. SIMOTION SCOUT is in online mode.
2. The affected SIMOTION device is active, e.g.
   - In the project navigator, the SIMOTION device or an element in its subtree is selected (such as program source, technology object).
   - In the working area, an open window is active that belongs to an element in the subtree of the SIMOTION device.
3. The SIMOTION device is consistent.

A background color highlights the occurrence of specific events in the affected task, see the following table. The task in question is displayed in the combo box of the function bar according to the event priority.

Table 7-502 Meaning of background colors in the Task status function bar

Background color	Meaning	Priority
Cyan	The affected task waits for a command at the "Single step" test function (only for SIMOTION MCC programming language).	Highest  Lowest
Red	The affected task is located at a breakpoint (Page 4967).	
Blue	In the affected task, the "Single step" test function is activated (only for SIMOTION MCC programming language)	
Gray	In the affected task, at least 1 breakpoint (Page 4967) is activated.	
Yellow	In the affected task, the "Monitoring" test function is activated (only for SIMOTION MCC programming language).	
White	In the affected task, none of the above-mentioned test functions are activated.	

Note

A selection of a task in the combo box is only possible:

- For the following test functions of the SIMOTION MCC programming language:
  - Monitoring
  - Single step
  - Trace
- at activated breakpoints (Page 4967) in the MCC or LAD/FBD programming languages.

Trace

Using the **trace tool**, you can record and store the course of variable values over time (z. B. unit variables, local variables, system variables, I/O variables). This allows you to document the optimization, for example, of axes.

You can set the recording time, display up to four channels, select trigger conditions, parameterize timing adjustments, select between different curve displays and scalings, etc.

Aside from isochronous recording, you can also select **Recording at code position**. This lets you record the values of variables whenever the program runs through a specific point in the ST source file.

The trace tool is described in detail in the online help.

Project comparison

SIMOTION SCOUT has a **project comparison** function (start this via the **Start object comparison** button) for comparing objects within the same project and/or objects from different projects (online or offline).

Project comparison allows you to establish any differences and, if necessary, run a data transfer to rectify them.

Objects are devices and their sub-objects, programs, technology objects (TOs) or drive objects (DOs), and libraries. Comparing projects is useful if you need to carry out service work on the system.

Further information on project and detail comparisons can be found in the SIMOTION Project Comparison Function Manual.

7.2.9 Appendix

7.2.9.1 Formal Language Description

In this chapter, you will find overviews of the basic elements of ST and a complete compilation of all syntax diagrams with the language elements. This appendix summarizes the basic features of the ST language.

Language description resources

Syntax diagrams are used as a basis for the language description in the individual sections. They provide you with an invaluable insight into the syntactic (i.e. grammatical) structure of ST.

Instructions for using syntax diagrams were presented in *Language description resources*. Information about the difference between formatted and unformatted rules, of interest to the advanced user, is presented below.

Formatted rules (lexical rules)

The lexical rules describe the structure of the elements processed by the compiler during lexical analysis. This means that the notation is formatted and the rules must be followed. In particular, that means:

- Insertion of formatting characters is not allowed.
- Block and line comments cannot be inserted.
- Attributes for identifiers cannot be inserted.

The following figure shows a lexical rule for legal identifiers.

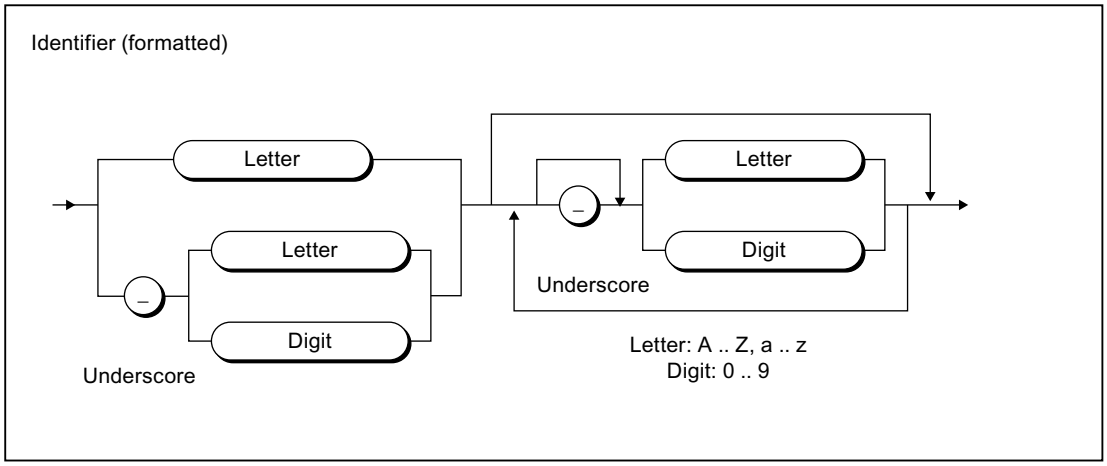


Figure 7-312 Example of a lexical rule

Valid examples according to this rule include:

```
R_CONTROLLER3
_A_ARRAY
_100_3_3_10
```

Unformatted rules (syntactic rules)

The syntactic rules build on the lexical rules and describe the structure of ST. You can write your ST program unformatted within the framework of these rules.

The unformatted property means:

- Formatting characters can be inserted anywhere.
- Block and line comments can be inserted.

The following example shows the syntactic rule for assigning a value in a statement.

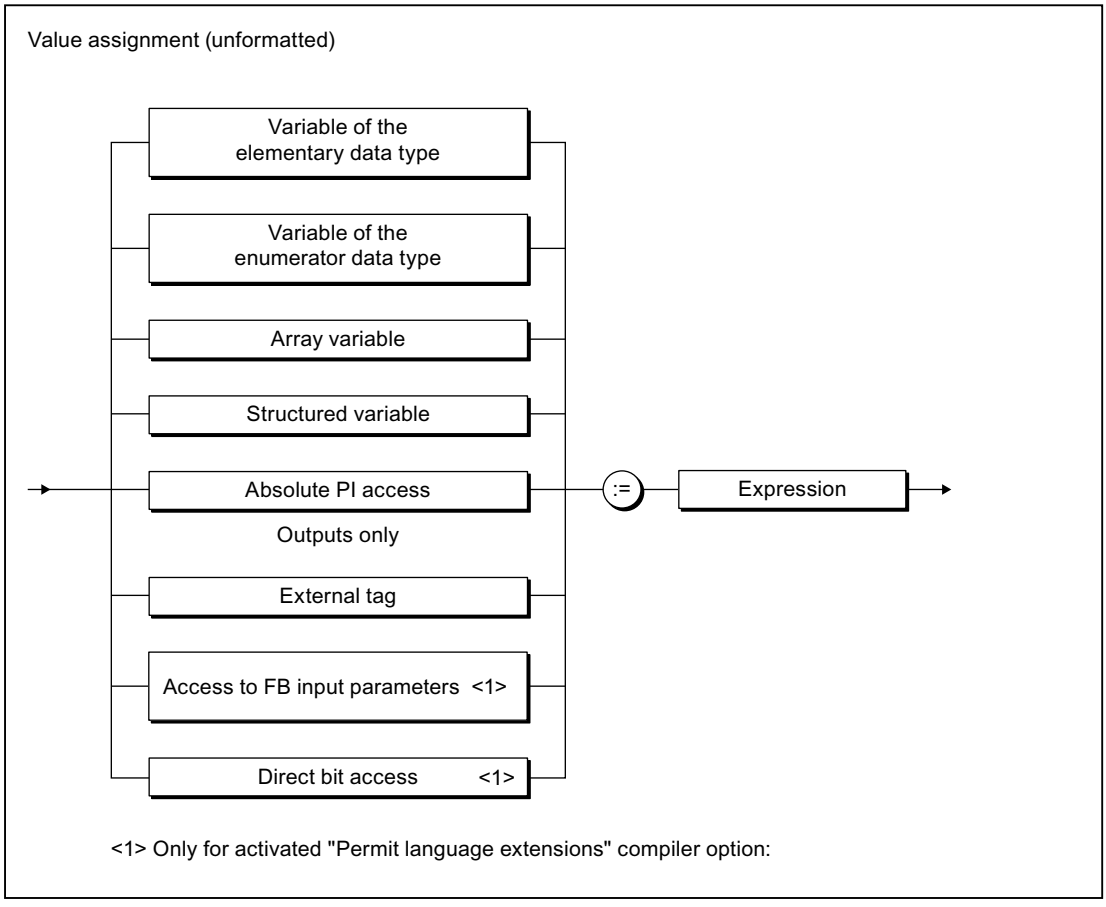


Figure 7-313 Example of a syntactic rule

Valid examples according to this rule include:

```
VARIABLE_1 := 100; SWITCH := FALSE;
//'This is a comment
VARIABLE_2:=3.2 +VARIABLE_1;
```

Basic elements (terminals)

A terminal is a basic element that is declared verbally and not by a further rule. It is represented in the syntax diagrams by an oval or circle.

Letters, digits and other characters

Letters and digits are the most commonly used characters. The *identifier*, for example, consists of a combination of letters, digits, and the underscore. The underscore is one of the special characters.

Table 7-503 Letters and digits

Characters	Subgroup	Character set elements
Letter	Upper case	A .. Z
	Lower case	a .. z
Digit	Decimal digit	0 .. 9
Octal digit	Octal digit	0 .. 7
Hexadecimal digit	Hexadecimal digit	0 .. 9, A .. F, a .. f
Bit	Binary digit	0, 1

You can use the complete extended ASCII character set in comments. You can use all printable ASCII code characters starting from decimal equivalent 32 (blank).

For language commands, identifiers, constants, expressions and operators, you can use special characters, i.e. characters other than letters and digits, only according to certain rules.

Formatting characters and separators in the rules

Formatting characters and separators are used differently in formatted (lexical) and unformatted (syntactic) rules. Language description resources (Page 4973) describes the differences between syntactic and lexical rules.

In the tables below, you will find the formatting characters and separators of the lexical and syntactic rules. You are also provided with a description and a list of all rules in which the formatting characters and separators are used as terminals (see Rules (Page 4988)).

Table 7-504 Formatting characters and separators in lexical rules

Characters	Description	Lexical rule
:	Separator between hours, minutes, and seconds	Time of day information
.	Separator for floating-point representation, time interval representation, absolute addressing	Floating-point representation, time-of-day information, decimal representation, access to local or global instance
_ Underscore	Separator for identifiers, separator for numerical values in constants	Identifiers, decimal digit string, binary digit string, octal digit string, hexadecimal digit string, sequence representation
%	Prefix for direct identifier on CPU memory access	Simple memory access
//	Comment	Line comment
(**)	Comment	Block comment

Table 7-505 Formatting characters and separators in syntactic rules

Characters	Description	Syntactic rule
:	Separator for type information	Function, variable declaration, component declaration, CASE statement, instance declaration
;	Ends a declaration or statement	Constant block, statement, variable declaration, instance declaration, component declaration, statement section
,	Separator for lists	Variable declaration, array initialization list, instance declaration, ARRAY data type specification, FB parameter, FC parameter, value list
..	Range information	Array data type specification, value list
.	Structure access	Structured variable
( )	Initialization list for arrays, parentheses in expressions, function and function block calls	Array initialization list, expression, simple multiplication, operand, exponent, FB call, function call
[ ]	Array declaration, structured variable section of array	Array data type specification

See also

Language description resources (Page 4654)

Formatting characters and separators for constants

Below, you will find all formatting characters and separators for constants with information on the lexical rule in which they are used.

Table 7-506 Formatting characters and separators for constants

Characters	Code for	Lexical rule
2#	Integer constant	Binary digit string
8#	Integer constant	Octal digit string
16#	Integer constant	Hexadecimal digit string
E	Separator for floating-point constants	Exponent
E	Separator for floating-point constants	Exponent
D#	Time information	Date
DATE#	Time information	Date
DATE_AND_TIME#	Time information	Date and time
DT#	Time information	Date and time
T#	Time information	Duration
TIME#	Time information	Duration
TIME_OF_DAY#	Time information	Time of day
TOD#	Time information	Time of day
d	Separator for time interval (day)	Days (rule: Sequence representation)

Characters	Code for	Lexical rule
h	Separator for time interval (hours)	Hours (rule: Sequence representation)
m	Separator for time interval (minutes)	Minutes (rule: Sequence representation)
ms	Separator for time interval (milliseconds)	Milliseconds (rule: Sequence representation)
s	Separator for time interval (seconds)	Seconds (rule: Sequence representation)

Predefined identifiers for process image access

Below is a list of all predefined variables in ST that you can use to access CPU memory areas (absolute identifiers). Note that you can read and write outputs but you can only read inputs.

Table 7-507 Absolute identifier

Identifier	Description	Lexical rule
%In.x or **%IXn.x**	CPU input range with byte and bit address	Absolute PI access
%IBn	CPU input range with byte address	Absolute PI access
%IWn	CPU input range with word address	Absolute PI access
%IDn	CPU input range with double word address	Absolute PI access
%Qn.x or **%QXn.x**	CPU output range with byte and bit address	Absolute PI access
%QBn	CPU output range with byte address	Absolute PI access
%QWn	CPU output range with word address	Absolute PI access
%QDn	CPU output range with double word address	Absolute PI access

Operators

Below is a list of all ST operators and the syntactic rules in which they are used.

Table 7-508 ST operators

Operator	Description	Rule
:=	Assignment operator (also for initialization values)	Value assignment, input assignment, in/out assignment, variable declaration, constant declaration, user-defined data types, component declaration, instance declaration
+, –	Arithmetic operators: Unary operators, sign	Expression, exponent
+, –, *, / **MOD**	Basic arithmetic operators	Expression, basic arithmetic operator
**	Arithmetic operators: Power operator	Expression
NOT	Logic operators: Negation	Expression, operand
AND, &, OR, XOR	Basic logic operators	Basic logic operator
<, >, <=, >=, =, <>	Relational operator	Relational operator
=>	Assignment operator	Output assignment
?= [1]	Dynamic type conversion	Dynamic type conversion
^	Dereferencing operator	Dereferencing of general references

[1] Only with compiler option "Permit object-oriented programming".

Reserved words

Below is an alphabetical list of keywords, predefined identifiers, and standard functions of the basic ST system. You are also provided with a description and the syntactic rule from *rules* in which they are used as terminals. An exception is standard functions, which are included only implicitly in the syntactic rule for *function calls* as the standard function name.

Note

Variables must not be assigned the names of keywords or predefined identifiers. For more information about identifiers, see *Identifiers in ST*. You will find an overview of the identifiers reserved for technology objects and other reserved identifiers in *Reserved identifiers*.

Table 7-509 ST keywords and predefined identifiers in the basic ST system

Keyword/identifier	Description	Rule
ABS	Standard numeric function	Function call
ABSTRACT [2]	Keyword	Class (Page 5002), Method in class (Page 5003)
ACOS	Standard numeric function	Function call
AND	Logic operator	Basic logic operator
ANYOBJECT	General data type for technology objects	TO data type

Keyword/identifier	Description	Rule
ANYOBJECT_TO_OBJECT	Standard function for type conversion (technology objects)	Function call
ANYTYPE_TO_BIGBYTEARRAY	Standard function (marshalling)	Function call
ANYTYPE_TO_LITTLEBYTEARRAY	Standard function (marshalling)	Function call
ARRAY	Introduces the specification of an array and is followed by the index list between **[** and **]**	ARRAY data type specification
AS	Introduces a namespace	–
ASIN	Standard numeric function	Function call
AT	Keyword for address specification	Component declaration with relative address, symbolic PI access
ATAN	Standard numeric function	Function call
BIGBYTEARRAY_TOANYTYPE	Standard function (marshalling)	Function call
BOOL	Elementary data type for binary data	Bit data type
BOOL_TO_BYTE	Standard function for type conversion	Function call
BOOL_TO_DWORD	Standard function for type conversion	Function call
BOOL_TO_WORD	Standard function for type conversion	Function call
BOOL_VALUE_TO_DINT	Standard function for type conversion	Function call
BOOL_VALUE_TO_INT	Standard function for type conversion	Function call
BOOL_VALUE_TO_LREAL	Standard function for type conversion	Function call
BOOL_VALUE_TO_REAL	Standard function for type conversion	Function call
BOOL_VALUE_TO_SINT	Standard function for type conversion	Function call
BOOL_VALUE_TO_UDINT	Standard function for type conversion	Function call
BOOL_VALUE_TO_UINT	Standard function for type conversion	Function call
BOOL_VALUE_TO_USINT	Standard function for type conversion	Function call
BY	Introduces the increment	FOR statement (Page 5036)
BYTE	Elementary data type	Bit data type
BYTE_TO_BOOL	Standard function for type conversion	Function call
BYTE_TO_DINT	Standard function for type conversion	Function call
BYTE_TO_DWORD	Standard function for type conversion	Function call
BYTE_TO_INT	Standard function for type conversion	Function call
BYTE_TO_SINT	Standard function for type conversion	Function call
BYTE_TO_UDINT	Standard function for type conversion	Function call
BYTE_TO_UINT	Standard function for type conversion	Function call
BYTE_TO_USINT	Standard function for type conversion	Function call
BYTE_TO_WORD	Standard function for type conversion	Function call
BYTE_VALUE_TO_LREAL	Standard function for type conversion	Function call
BYTE_VALUE_TO_REAL	Standard function for type conversion	Function call
CASE	Introduces a control statement for selection	CASE statement
CLASS [2]	Introduces the class	Class (Page 5002)
CONCAT	Standard function for string editing	Function call
CONCAT_DATE_TOD	Standard function for type conversion	Function call
CONSTANT	Introduces a constant definition	Constant block
CONTINUE [1]	Jump directly to the beginning of a loop	CONTINUE statement

Keyword/identifier	Description	Rule
COS	Standard numeric function	Function call
CTD	Down counter	Function block call
CTD_DINT	Down counter	Function block call
CTD_UDINT	Down counter	Function block call
CTU	Up counter	Function block call
CTU_DINT	Up counter	Function block call
CTU_UDINT	Up counter	Function block call
CTUD	Up/down counter	Function block call
CTUD_DINT	Up/down counter	Function block call
CTUD_UDINT	Up/down counter	Function block call
DATE	Elementary data type for date	Time data type
DATE_AND_TIME	Elementary data type for date and time	Time data type
DATE_AND_TIME_TO_DATE	Standard function for type conversion	Function call
DATE_AND_TIME_TO_TIME_OF_DAY	Standard function for type conversion	Function call
DELETE	Standard function for string editing	Function call
DINT	Elementary data type for double precision integer with value range -2**31 .. 2**31-1	Numeric data type
DINT_TO_BYTE	Standard function for type conversion	Function call
DINT_TO_DWORD	Standard function for type conversion	Function call
DINT_TO_INT	Standard function for type conversion	Function call
DINT_TO_LREAL	Standard function for type conversion	Function call
DINT_TO_REAL	Standard function for type conversion	Function call
DINT_TO_SINT	Standard function for type conversion	Function call
DINT_TO_STRING	Standard function for type conversion	Function call
DINT_TO_UDINT	Standard function for type conversion	Function call
DINT_TO_UINT	Standard function for type conversion	Function call
DINT_TO_USINT	Standard function for type conversion	Function call
DINT_TO_WORD	Standard function for type conversion	Function call
DINT_VALUE_TO_BOOL	Standard function for type conversion	Function call
DO	Introduces the body for a FOR statement, WHILE statement, or WAITFORCONDITION statement	FOR statement (Page 5036), WHILE statement (Page 5036), WAITFORCONDITION statement (Page 5038)
DT	Shorthand notation for DATE_AND_TIME	Time data type
DT_TO_DATE	Standard function for type conversion	Function call
DT_TO_TOD	Standard function for type conversion	Function call
DWORD	Elementary data type for double word	Bit data type
DWORD_TO_BOOL	Standard function for type conversion	Function call
DWORD_TO_BYTE	Standard function for type conversion	Function call
DWORD_TO_DINT	Standard function for type conversion	Function call
DWORD_TO_INT	Standard function for type conversion	Function call
DWORD_TO_REAL	Standard function for type conversion	Function call
DWORD_TO_SINT	Standard function for type conversion	Function call
DWORD_TO_UDINT	Standard function for type conversion	Function call

Keyword/identifier	Description	Rule
DWORD_TO_UINT	Standard function for type conversion	Function call
DWORD_TO_USINT	Standard function for type conversion	Function call
DWORD_TO_WORD	Standard function for type conversion	Function call
DWORD_VALUE_TO_LREAL	Standard function for type conversion	Function call
DWORD_VALUE_TO_REAL	Standard function for type conversion	Function call
ELSE	Introduces the clause to be executed if no condition true	IF statement, CASE statement
ELSIF	Introduces alternative condition	IF statement
END_CASE	Ends the CASE statement	CASE statement
END_CLASS [2]	Ends class	Class (Page 5002)
END_EXPRESSION	Ends the EXPRESSION statement	Expression (Page 5001)
END_FOR	Ends the FOR statement	FOR statement (Page 5036)
END_FUNCTION	Ends the function	Function (Page 5001)
END_FUNCTION_BLOCK	Ends the function block	Function block (Page 5001)
END_IF	Ends the IF statement	IF statement
END_IMPLEMENTATION	Ends implementation section	Implementation section
END_INTERFACE	Ends interface section, ends object-oriented interface [2]	Interface section, object-oriented interface (Page 5004)
END_LABEL	Ends the LABEL statement	Jump label declaration (Page 5015)
END_METHOD [2]	Ends method	Method in class (Page 5003), Method in FB (Page 5005), Method prototype (Page 5004)
END_NAMESPACE [2]	Ends the namespace	Namespace
END_PROGRAM	Ends the program	Program (Page 5002)
END_REPEAT	Ends the REPEAT statement	REPEAT statement (Page 5037)
END_STRUCT	Ends the specification of a structure	STRUCT data type specification
END_TYPE	Ends the UDT definition	User-defined data type
END_VAR	Ends a declaration block	Variable block, parameter block, constant block
END_WAITFORCONDITION	Ends the control statement for a task waiting for a programmable event	WAITFORCONDITION statement (Page 5038)
END_WHILE	Ends the WHILE statement	WHILE statement (Page 5037)
ENUM_TO_DINT	Standard function for type conversion	Function call
EXIT	Direct exit from loop execution	EXIT statement (Page 5037)
EXP	Standard numeric function	Function call
EXPD	Standard numeric function	Function call
EXPRESSION	Programmable event for waiting task	Expression (Page 5001)
EXPT	Standard numeric function	Function call
EXTENDS [2]	Derivation of a class	Class (Page 5002)
F_TRIG	Detects falling edge	Function block call

Keyword/identifier	Description	Rule
FALSE	Predefined Boolean constant: Logical condition false, value equal to 0	–
FINAL [2]	Keyword	Class (Page 5002), Method in class (Page 5003)
FIND	Standard function for string editing	Function call
FOR	Introduces the control statement for loop execution	FOR statement (Page 5036)
FROM_BIG_ENDIAN [1]	Standard function for endian conversion	Function call
FROM_LITTLE_ENDIAN [1]	Standard function for endian conversion	Function call
FUNCTION	Introduces the function	Function (Page 5001)
FUNCTION_BLOCK	Introduces the function block	Function block (Page 5001)
GOTO	Jump	GOTO statement (Page 5038)
IF	Introduces a control statement for selection	IF statement
IMPLEMENTATION	Introduces the implementation section	Implementation section
IMPLEMENTS [2]	Implements object-oriented interfaces	Class (Page 5002)
INSERT	Standard function for string editing	Function call
INT	Elementary data type for single precision integer with value range -2**15.. 2**15-1	Numeric data type
INT_TO_BYTE	Standard function for type conversion	Function call
INT_TO_DINT	Standard function for type conversion	Function call
INT_TO_DWORD	Standard function for type conversion	Function call
INT_TO_LREAL	Standard function for type conversion	Function call
INT_TO_REAL	Standard function for type conversion	Function call
INT_TO_SINT	Standard function for type conversion	Function call
INT_TO_TIME	Standard function for type conversion	Function call
INT_TO_UDINT	Standard function for type conversion	Function call
INT_TO_UINT	Standard function for type conversion	Function call
INT_TO_USINT	Standard function for type conversion	Function call
INT_TO_WORD	Standard function for type conversion	Function call
INT_VALUE_TO_BOOL	Standard function for type conversion	Function call
INTERFACE	Introduces interface section, introduces object-oriented interface [2]	Interface section, object-oriented interface (Page 5004)
IS_VALID [1]	Standard validity check function	Function call
LABEL	Definition of jump labels	Jump label declaration (Page 5015)
LEFT	Standard function for string editing	Function call
LEN	Standard function for string editing	Function call
LIMIT	Standard function for selection	Function call
LITTLEBYTEARRAY_TOANYTYPE	Standard function (marshalling)	Function call
LN	Standard numeric function	Function call
LOG	Standard numeric function	Function call
LOWER_BOUND [1]	Standard function for lower array boundary	Function call

Keyword/identifier	Description	Rule
LREAL	Elementary data type for 64-bit double-precision floating-point number (long real)	Numeric data type
LREAL_TO_DINT	Standard function for type conversion	Function call
LREAL_TO_INT	Standard function for type conversion	Function call
LREAL_TO_REAL	Standard function for type conversion	Function call
LREAL_TO_SINT	Standard function for type conversion	Function call
LREAL_TO_STRING	Standard function for type conversion	Function call
LREAL_TO_UDINT	Standard function for type conversion	Function call
LREAL_TO_UINT	Standard function for type conversion	Function call
LREAL_TO_USINT	Standard function for type conversion	Function call
LREAL_VALUE_TO_BOOL	Standard function for type conversion	Function call
LREAL_VALUE_TO_BYTE	Standard function for type conversion	Function call
LREAL_VALUE_TO_DWORD	Standard function for type conversion	Function call
LREAL_VALUE_TO_WORD	Standard function for type conversion	Function call
MAX	Standard function for selection	Function call
METHOD [2]	Introduces the method	Method in class (Page 5003), Method in FB (Page 5005), Method prototype (Page 5004)
MID	Standard function for string editing	Function call
MIN	Standard function for selection	Function call
MOD	Arithmetic operator for division remainder	Basic arithmetic operator
MUX	Standard function for selection	Function call
NOT	Logic operator, belongs to the unary operators	Expression, operand
NAMESPACE [2]	Introduces the namespace	Namespace
NULL	Invalid reference	-
OF	Keyword	ARRAY data type specification, CASE statement
OR	Logic operator	Basic logic operator
OVERLAP	Introduction for a structure with overlapping address ranges	STRUCT data type specification
OVERRIDE [2]	Keyword	Method in class (Page 5003), Static variable block, Retentive local variable block
PRIVATE [2]	Access identifier	Class access identifier (Page 5003), FB access identifier (Page 5005)
PROGRAM	Introduces the program	Program (Page 5002)
PROTECTED [2]	Access identifier	Class access identifier (Page 5003)
PUBLIC [2]	Access identifier	Class access identifier (Page 5003), FB access identifier (Page 5005)
R_TRIG	Detects rising edge	Function block call
REAL	Elementary data type for 32-bit single precision floating-point number (real)	Numeric data type

Keyword/identifier	Description	Rule
REAL_TO_DINT	Standard function for type conversion	Function call
REAL_TO_DWORD	Standard function for type conversion	Function call
REAL_TO_INT	Standard function for type conversion	Function call
REAL_TO_LREAL	Standard function for type conversion	Function call
REAL_TO_SINT	Standard function for type conversion	Function call
REAL_TO_STRING	Standard function for type conversion	Function call
REAL_TO_TIME	Standard function for type conversion	Function call
REAL_TO_UDINT	Standard function for type conversion	Function call
REAL_TO_UINT	Standard function for type conversion	Function call
REAL_TO_USINT	Standard function for type conversion	Function call
REAL_VALUE_TO_BOOL	Standard function for type conversion	Function call
REAL_VALUE_TO_BYTE	Standard function for type conversion	Function call
REAL_VALUE_TO_DWORD	Standard function for type conversion	Function call
REAL_VALUE_TO_WORD	Standard function for type conversion	Function call
REF [2]	Formation of general references	General reference
REF_TO [2]	Declaration of general references	General reference
REPEAT	Introduces the control statement for loop execution	REPEAT statement (Page 5037)
REPLACE	Standard function for string editing	Function call
RETAIN	Declaration of buffered variables	Retentive variable block
RETURN	Control statement for returning from subprogram	RETURN statement (Page 5037)
RIGHT	Standard function for string editing	Function call
ROL	Bit string standard functions	Function call
ROR	Bit string standard functions	Function call
RS	Bistable function block (priority reset)	Function block call
RTC	Real-time clock	Function block call
SEL	Standard function for selection	Function call
SHL	Bit string standard functions	Function call
SHR	Bit string standard functions	Function call
SIN	Standard numeric function	Function call
SINT	Elementary data type for short integer with value range -128 .. 127	Numeric data type
SINT_TO_BYTE	Standard function for type conversion	Function call
SINT_TO_DINT	Standard function for type conversion	Function call
SINT_TO_DWORD	Standard function for type conversion	Function call
SINT_TO_INT	Standard function for type conversion	Function call
SINT_TO_LREAL	Standard function for type conversion	Function call
SINT_TO_REAL	Standard function for type conversion	Function call
SINT_TO_UDINT	Standard function for type conversion	Function call
SINT_TO_UINT	Standard function for type conversion	Function call
SINT_TO_USINT	Standard function for type conversion	Function call
SINT_TO_WORD	Standard function for type conversion	Function call

Keyword/identifier	Description	Rule
SINT_VALUE_TO_BOOL	Standard function for type conversion	Function call
SQRT	Standard numeric function	Function call
SR	Bistable function block (set as priority)	Function block call
STRING	Elementary data type for character strings	String data type
STRING_TO_DINT	Standard function for type conversion	Function call
STRING_TO_LREAL	Standard function for type conversion	Function call
STRING_TO_REAL	Standard function for type conversion	Function call
STRING_TO_UDINT	Standard function for type conversion	Function call
STRUCT	Introduces the specification of a structure and is followed by a list of components	STRUCT data type specification
StructAlarmId	Data type for AlarmId	–
StructAlarmId_TO_DINT	Standard function for type conversion	Function call
StructTaskId	Data type for TaskId	–
SUPER [2]	Keyword for calling the method of the base class	Class method call
TAN	Standard numeric function	Function call
THEN	Introduces subsequent actions if condition true	IF statement
THIS [2]	Keyword for calling the currently valid method	Class method call, FB method call
TIME	Elementary data type for time information	Time data type
TIME_OF_DAY	Elementary data type for time of day	Time data type
TIME_TO_INT	Standard function for type conversion	Function call
TIME_TO_REAL	Standard function for type conversion	Function call
TO	Introduces end value	FOR statement (Page 5036)
TOD	Shorthand notation for TIME_OF_DAY	Time data type
TOF	OFF delay	Function block call
TON	ON delay	Function block call
TO_BIG_ENDIAN [1]	Standard function for endian conversion	Function call
TO_LITTLE_ENDIAN [1]	Standard function for endian conversion	Function call
TP	Pulse	Function block call
TRUE	Predefined Boolean constant: Logical condition true, value not equal to 0	–
TRUNC	Standard numeric function	Function call
TYPE	Introduces the UDT definition	User-defined data type
UDINT	Elementary data type for unsigned double precision integer with value range 0 .. 2**32-1	Numeric data type
UDINT_TO_BYTE	Standard function for type conversion	Function call
UDINT_TO_DINT	Standard function for type conversion	Function call
UDINT_TO_DWORD	Standard function for type conversion	Function call
UDINT_TO_INT	Standard function for type conversion	Function call
UDINT_TO_LREAL	Standard function for type conversion	Function call
UDINT_TO_REAL	Standard function for type conversion	Function call
UDINT_TO_SINT	Standard function for type conversion	Function call
UDINT_TO_STRING	Standard function for type conversion	Function call
UDINT_TO_UINT	Standard function for type conversion	Function call

Keyword/identifier	Description	Rule
UDINT_TO_USINT	Standard function for type conversion	Function call
UDINT_TO_WORD	Standard function for type conversion	Function call
UDINT_VALUE_TO_BOOL	Standard function for type conversion	Function call
UINT	Elementary data type for unsigned single precision integer with value range 0 .. 2**16-1	Numeric data type
UINT_TO_BYTE	Standard function for type conversion	Function call
UINT_TO_DINT	Standard function for type conversion	Function call
UINT_TO_DWORD	Standard function for type conversion	Function call
UINT_TO_INT	Standard function for type conversion	Function call
UINT_TO_LREAL	Standard function for type conversion	Function call
UINT_TO_REAL	Standard function for type conversion	Function call
UINT_TO_SINT	Standard function for type conversion	Function call
UINT_TO_UDINT	Standard function for type conversion	Function call
UINT_TO_USINT	Standard function for type conversion	Function call
UINT_TO_WORD	Standard function for type conversion	Function call
UINT_VALUE_TO_BOOL	Standard function for type conversion	Function call
UNIT	Introduces the UNIT section	Unit section
UNTIL	Introduces exit condition for REPEAT statement	REPEAT statement (Page 5037)
UPPER_BOUND [1]	Standard function for upper array boundary	Function call
USELIB	Introduces the library name	–
USEPACKAGE	Introduces the package name	–
USES	Introduces a reference to other units	–
USING [2]	Use of a namespace	Namespace
USINT	Elementary data type for unsigned short integer with value range 0 .. 255	Numeric data type
USINT_TO_BYTE	Standard function for type conversion	Function call
USINT_TO_DINT	Standard function for type conversion	Function call
USINT_TO_DWORD	Standard function for type conversion	Function call
USINT_TO_INT	Standard function for type conversion	Function call
USINT_TO_LREAL	Standard function for type conversion	Function call
USINT_TO_REAL	Standard function for type conversion	Function call
USINT_TO_SINT	Standard function for type conversion	Function call
USINT_TO_UDINT	Standard function for type conversion	Function call
USINT_TO_UINT	Standard function for type conversion	Function call
USINT_TO_WORD	Standard function for type conversion	Function call
USINT_VALUE_TO_BOOL	Standard function for type conversion	Function call
VAR	Introduces a declaration block for local variables or constants	Static variable block, temporary variable block in FC, constant block
VAR_GLOBAL	Introduces a declaration block for unit variables (global variables) or unit constants	Unit variables, unit constants
VAR_IN_OUT	Introduces a declaration block	Parameter block
VAR_INPUT	Introduces a declaration block	Parameter block

Keyword/identifier	Description	Rule
VAR_OUTPUT	Introduces a declaration block	Parameter block
VAR_TEMP	Introduces a declaration block	Temporary variable block in FB/program
VOID	No return value for function	Function (Page 5001)
WAITFORCONDITION	Introduces the control statement for a task waiting for a programmable event	WAITFORCONDITION statement (Page 5038)
WHILE	Introduces the control statement for loop execution	WHILE statement (Page 5037)
WITH	Use in conjunction with WAITFORCONDITION	WAITFORCONDITION statement (Page 5038)
WORD	Elementary data type for word	Bit data type
WORD_TO_BOOL	Standard function for type conversion	Function call
WORD_TO_BYTE	Standard function for type conversion	Function call
WORD_TO_DINT	Standard function for type conversion	Function call
WORD_TO_DWORD	Standard function for type conversion	Function call
WORD_TO_INT	Standard function for type conversion	Function call
WORD_TO_SINT	Standard function for type conversion	Function call
WORD_TO_UDINT	Standard function for type conversion	Function call
WORD_TO_UINT	Standard function for type conversion	Function call
WORD_TO_USINT	Standard function for type conversion	Function call
WORD_VALUE_TO_LREAL	Standard function for type conversion	Function call
WORD_VALUE_TO_REAL	Standard function for type conversion	Function call
XOR	Logic operator	Basic logic operator

1 Only with active compiler option "Permit language extensions, IEC61131 3rd edition".

2 Only with active compiler option "Permit object-oriented programming".

Rules

The following syntax rules of the ST language are subdivided into rules with formatted notation (lexical rules) and unformatted notation (syntactic rules). *Language description resources* describes the differences between syntactic and lexical rules.

Identifiers

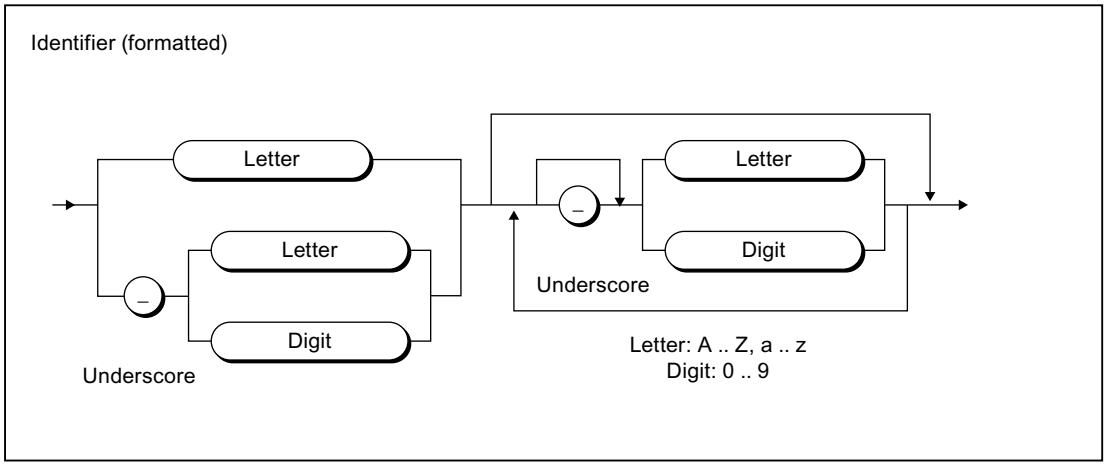


Figure 7-314 Identifier

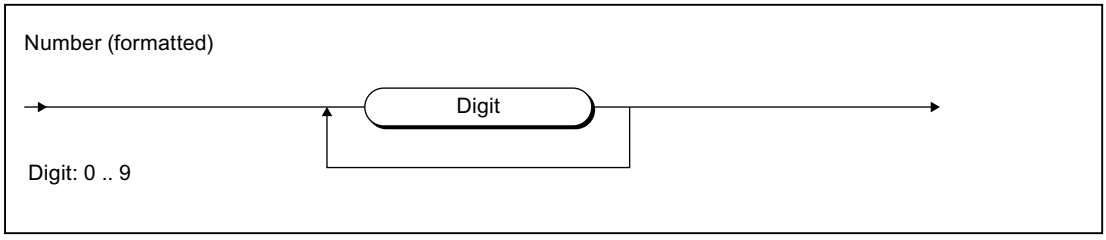


Figure 7-315 Number

Notation for constants (literals)

Literals

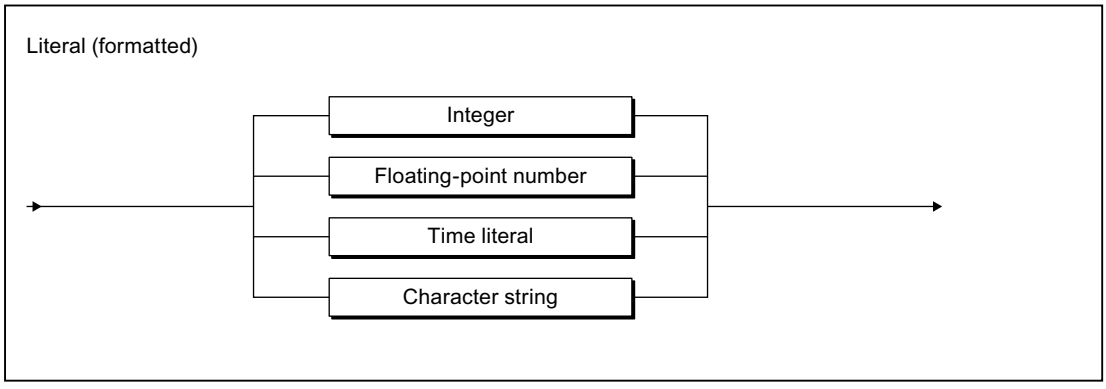


Figure 7-316 Literal

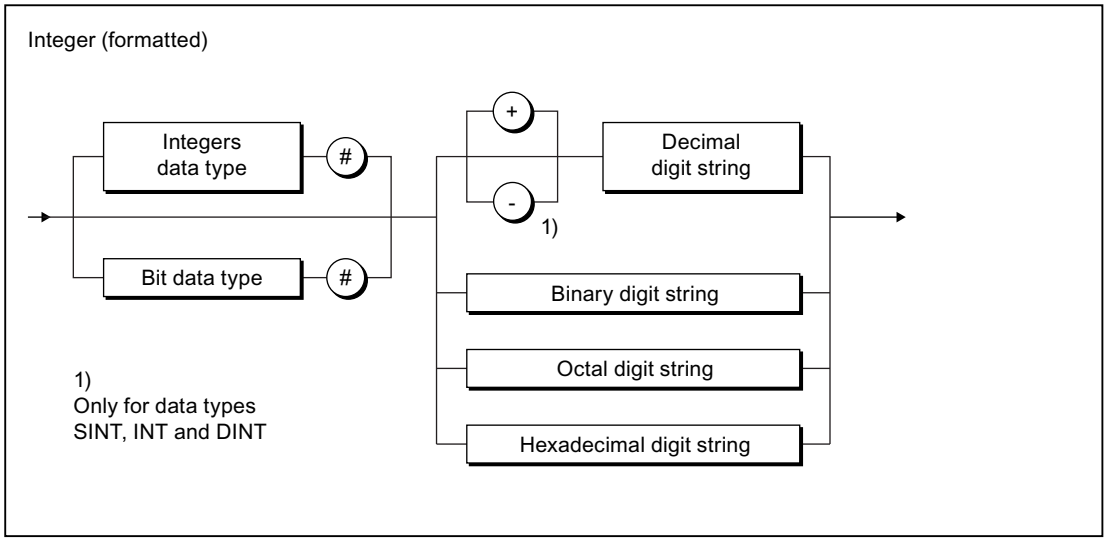


Figure 7-317 Integer

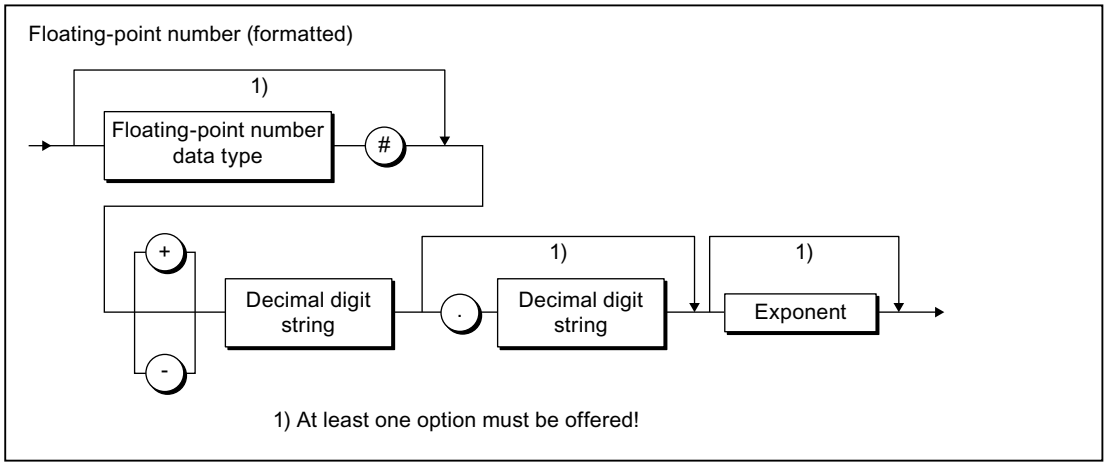


Figure 7-318 Floating-point number

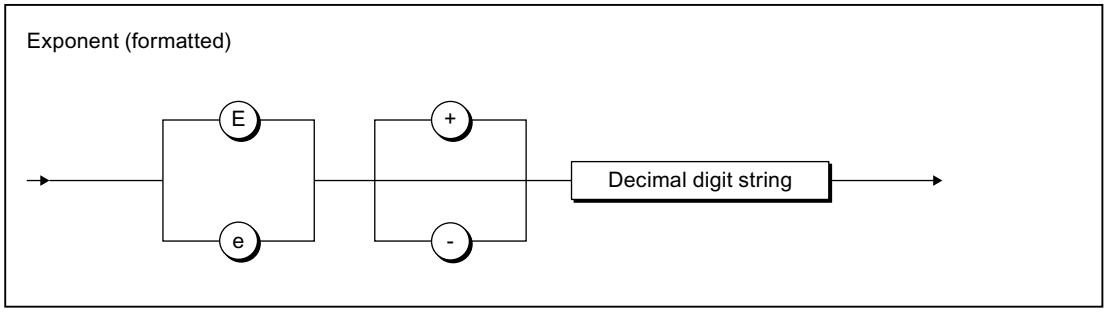


Figure 7-319 Exponent

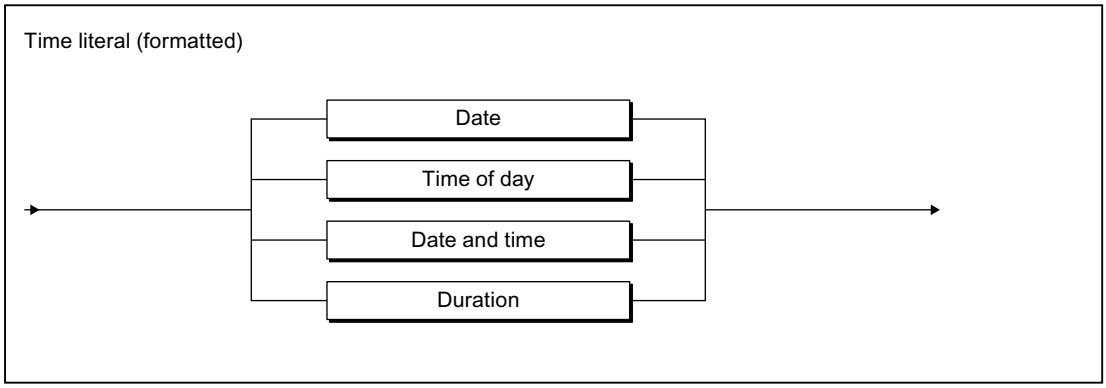


Figure 7-320 Time literal

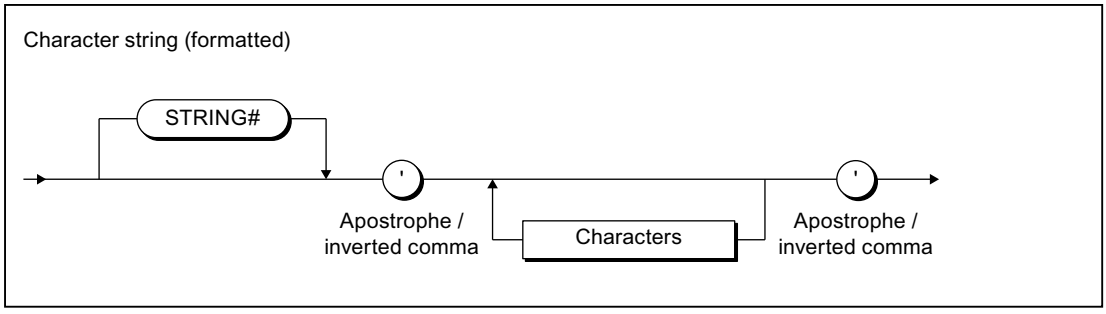


Figure 7-321 Character string

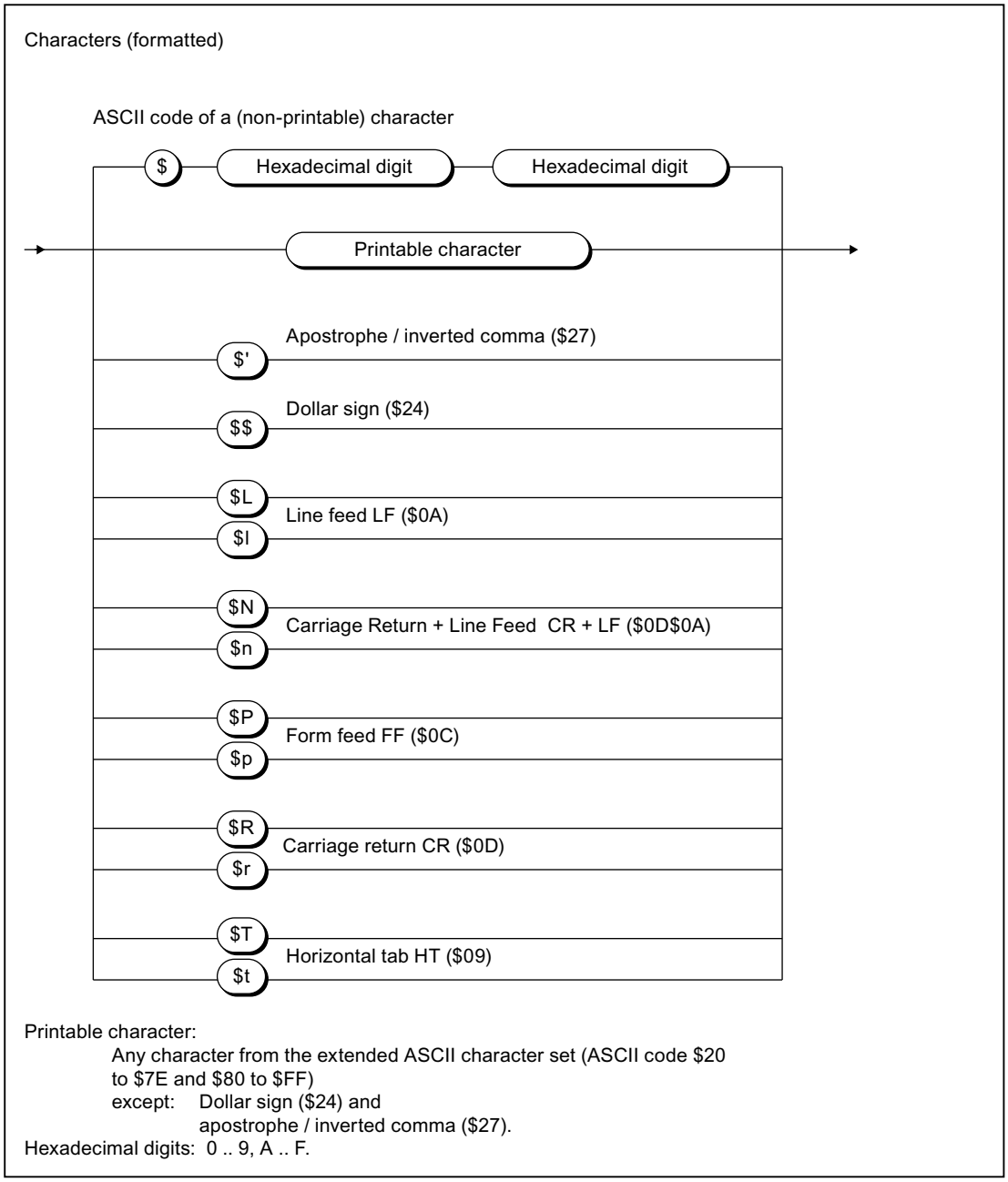


Figure 7-322 Character

Digit string

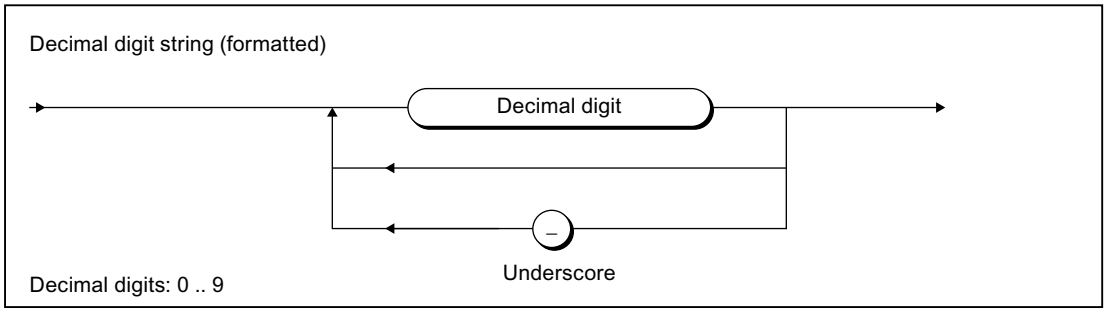


Figure 7-323 Decimal digit string

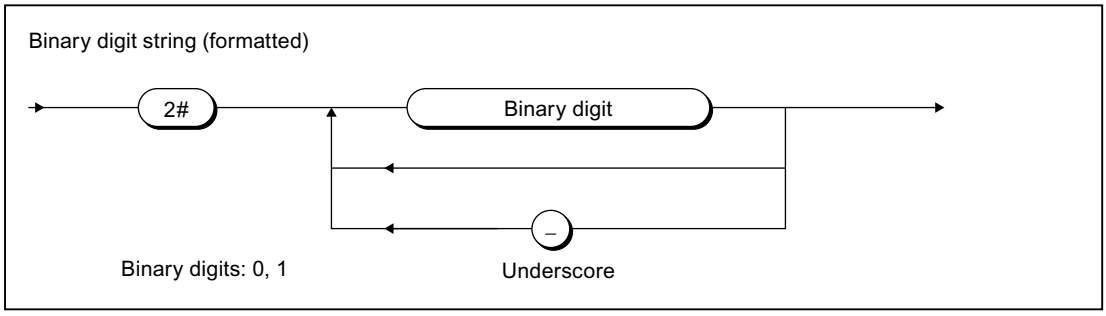


Figure 7-324 Binary digit string

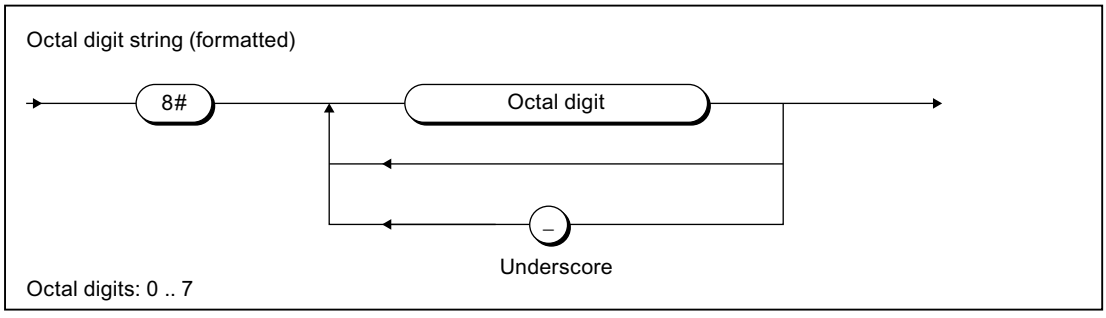


Figure 7-325 Octal digit string

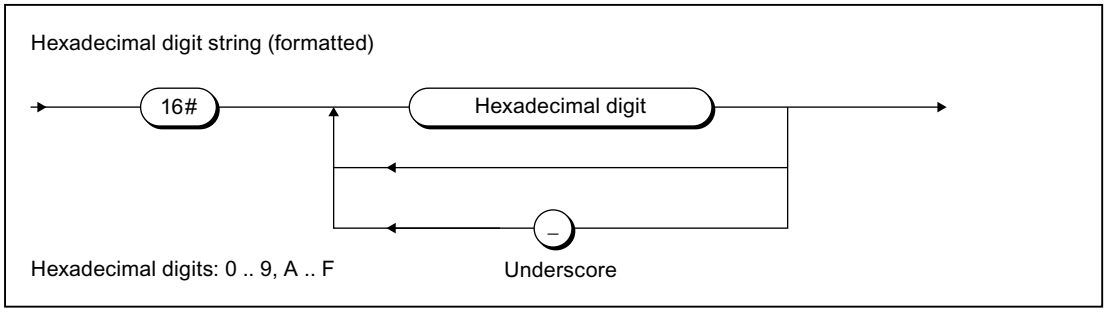


Figure 7-326 Hexadecimal digit string

Date and time

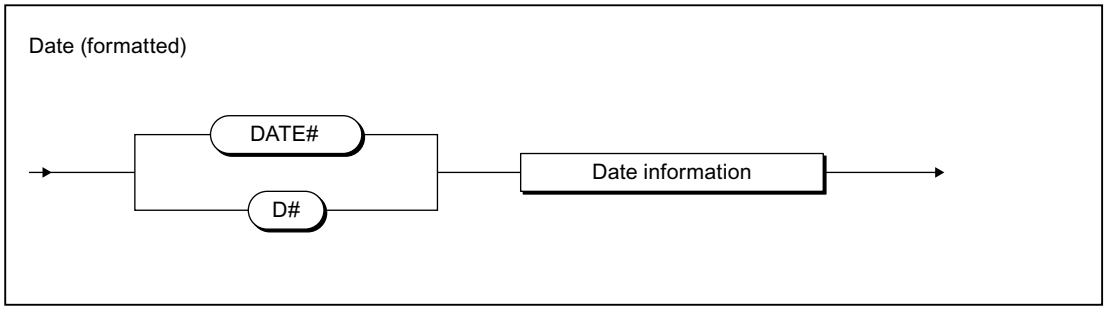


Figure 7-327 Date

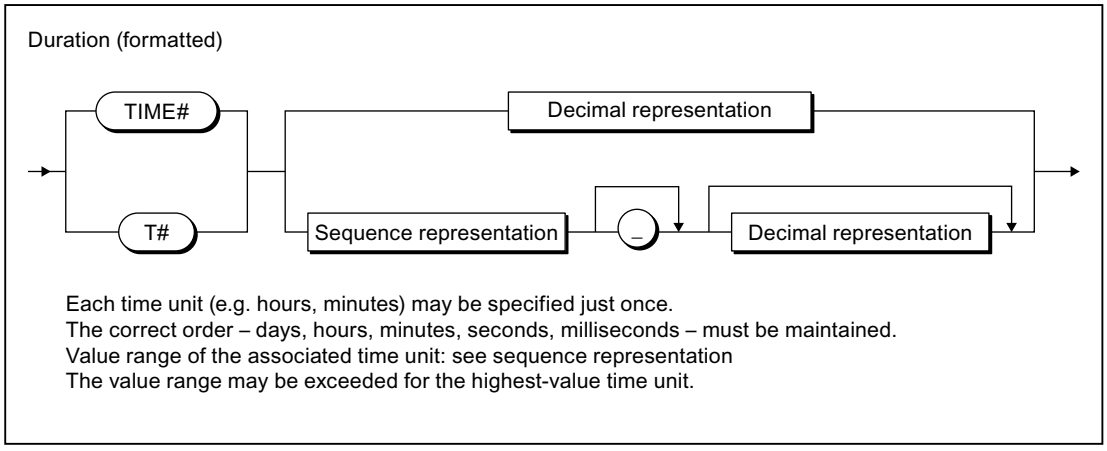


Figure 7-328 Time

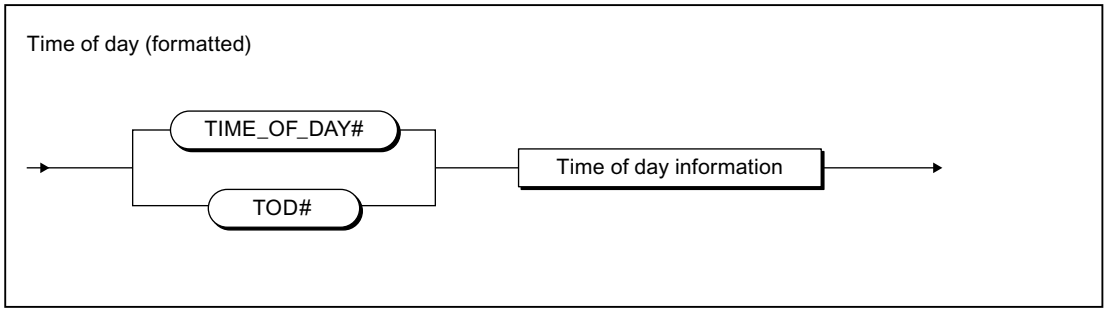


Figure 7-329 Time

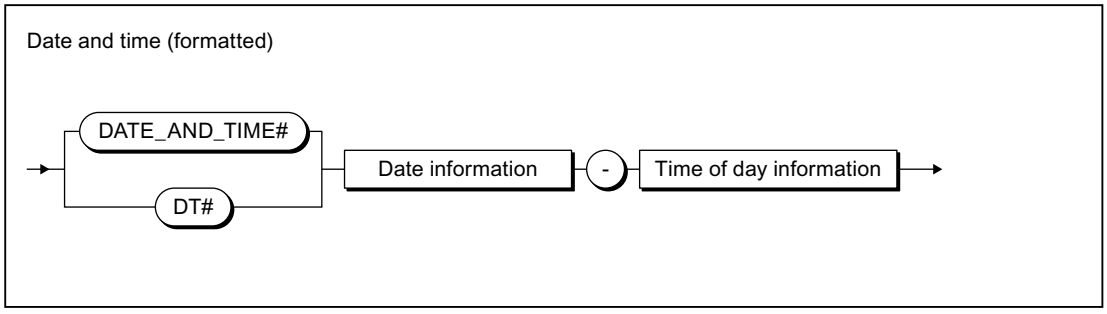


Figure 7-330 Date and time

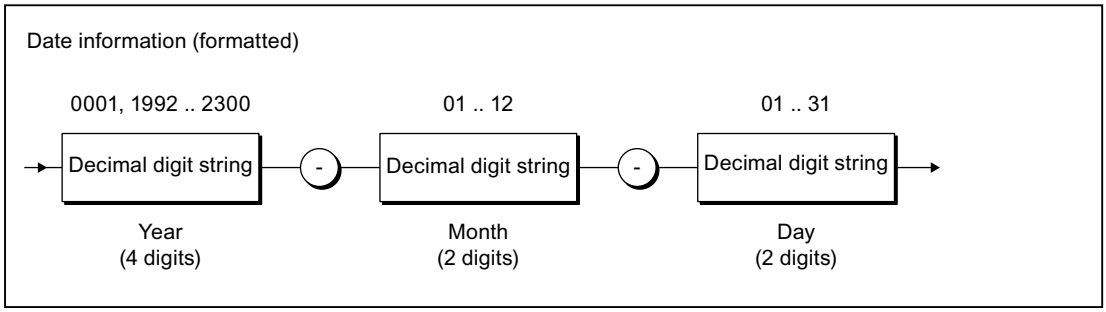


Figure 7-331 Date information

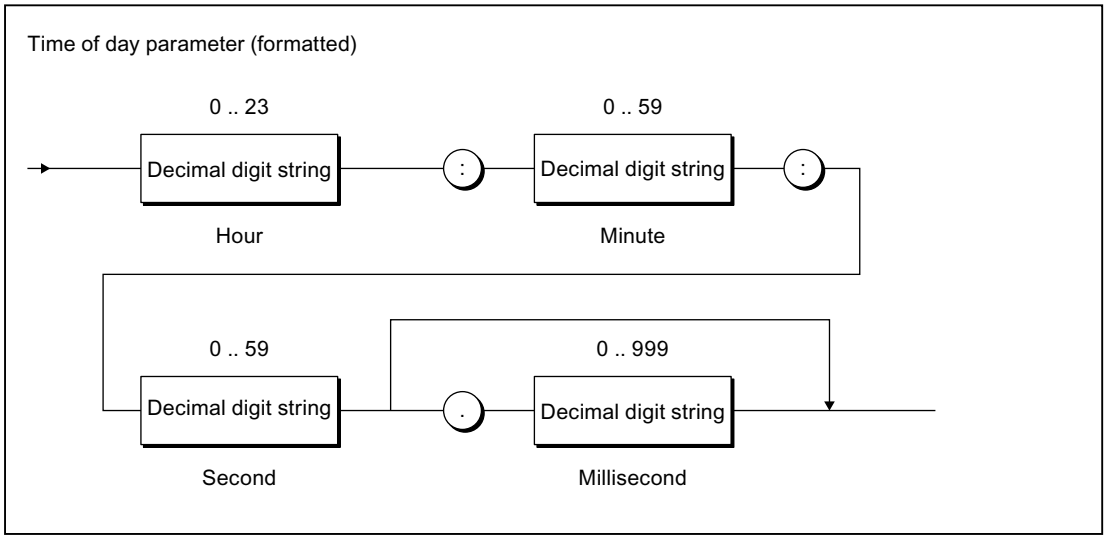


Figure 7-332 Time of day information

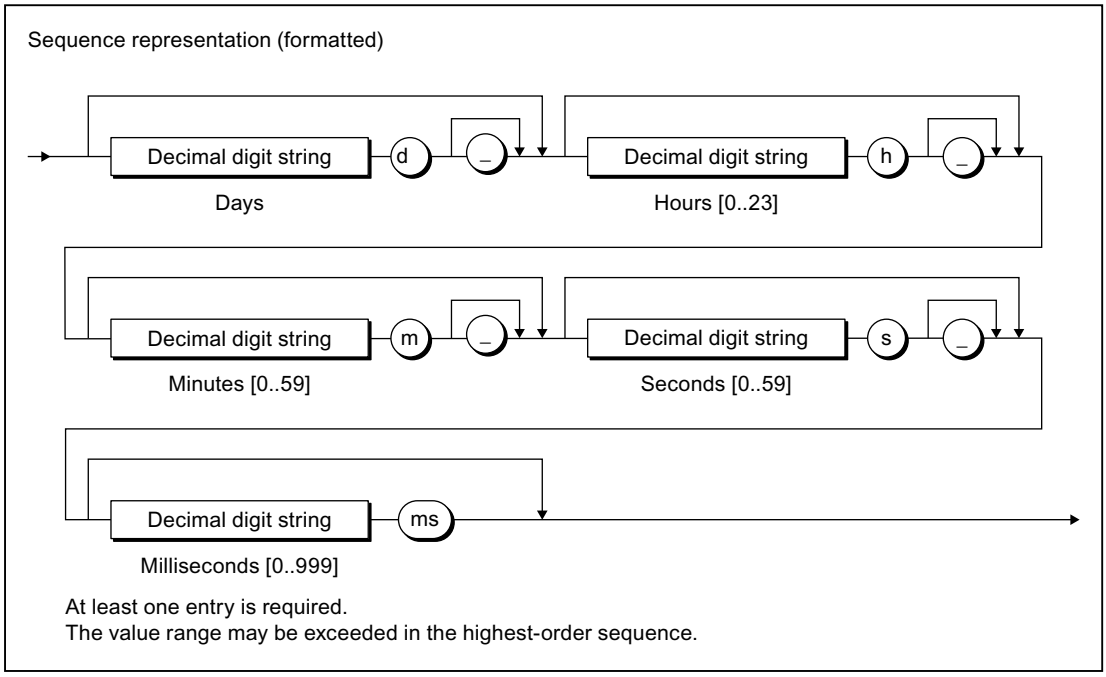


Figure 7-333 Sequence representation

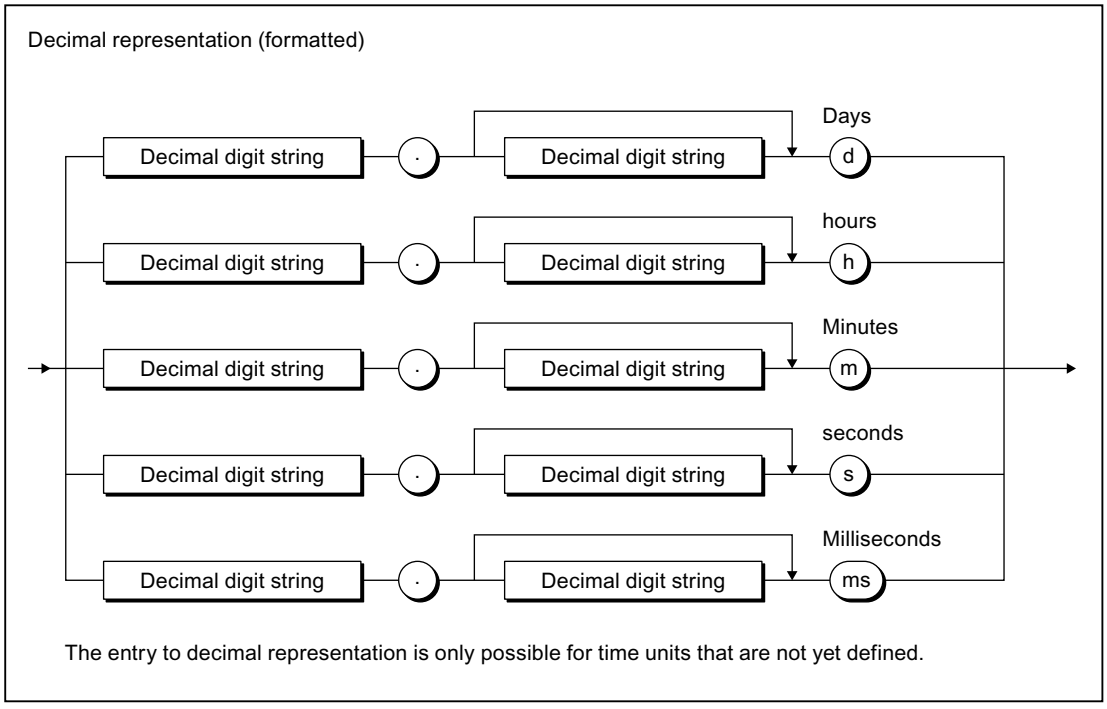


Figure 7-334 Decimal representation

Comments

Note the following when inserting comments:

- The character pairs (* and *) are ignored within the line comment.
- Nesting of block comments is not allowed as standard. However, you can nest line comments in block comments.
  With activated compiler option "Permit language extensions IEC61131 3rd edition" only: It is possible to nest block comments.
- You can use the complete extended ASCII character set in comments.
- Comments are not allowed in formatted (lexical) rules.

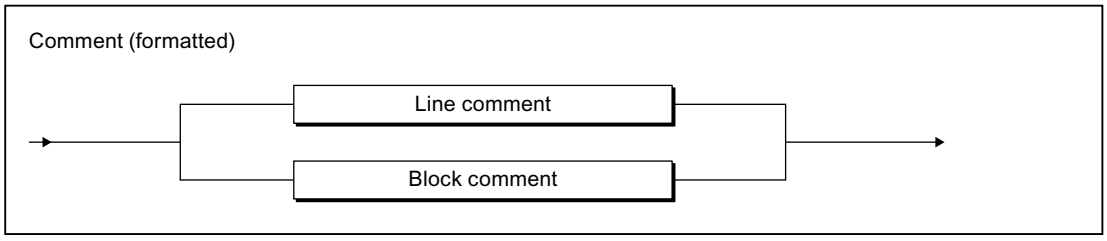


Figure 7-335 Comment

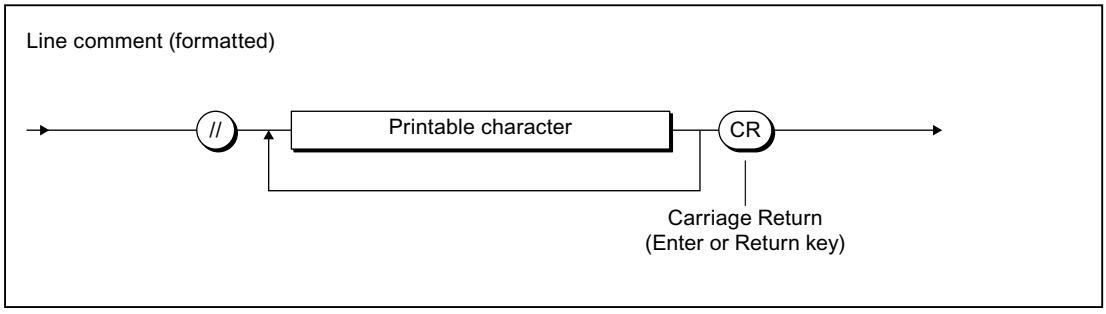


Figure 7-336 Line comment

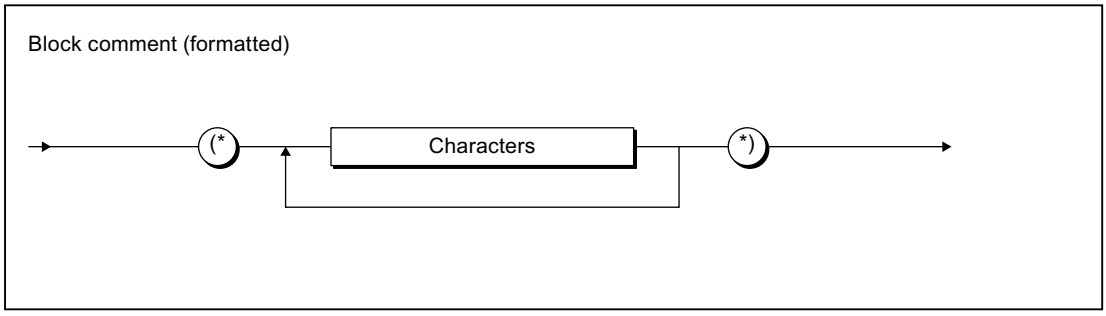
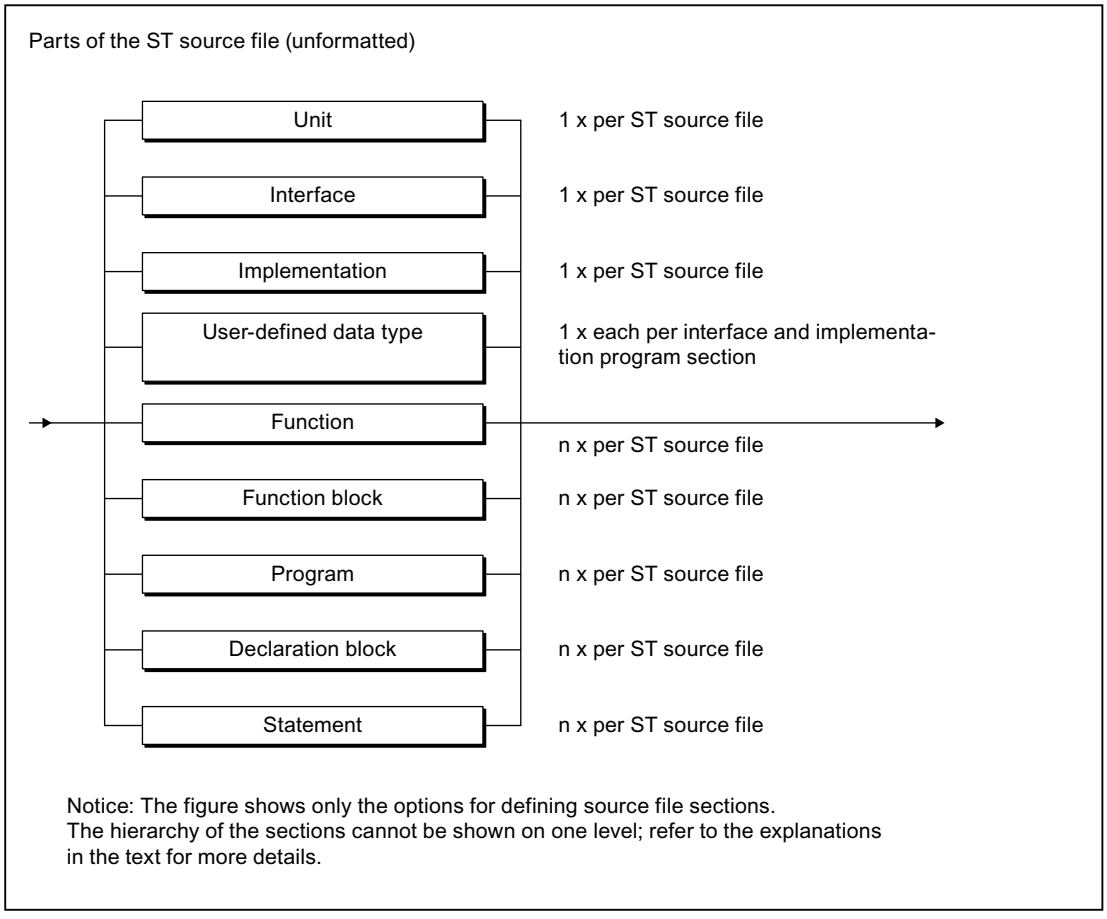


Figure 7-337 Block comment

Sections of the ST source file



Sections of the ST source file

Structures of ST source files

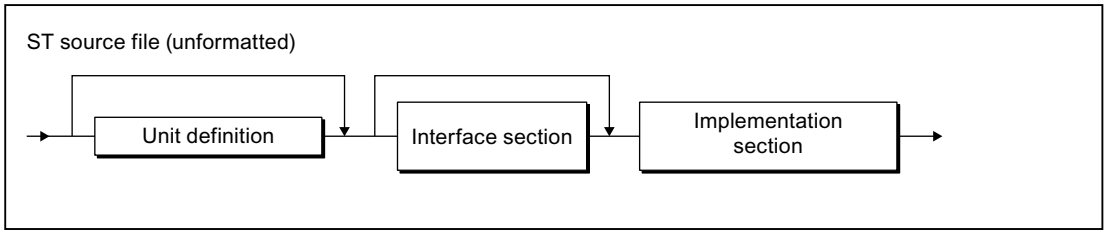


Figure 7-338 ST source file

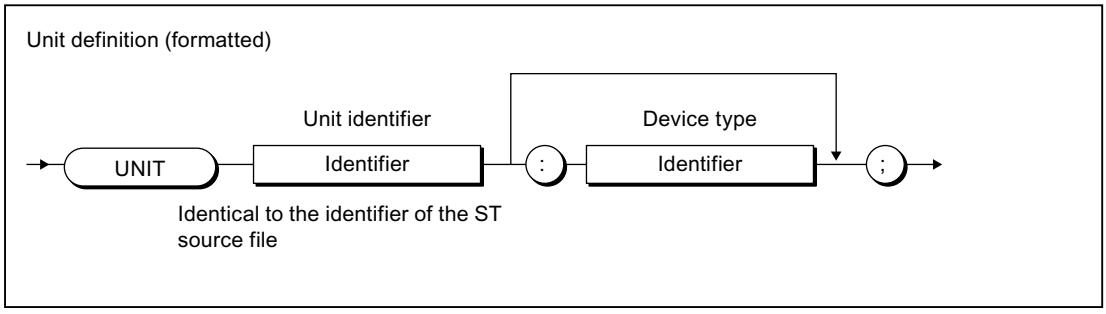


Figure 7-339 Unit definition

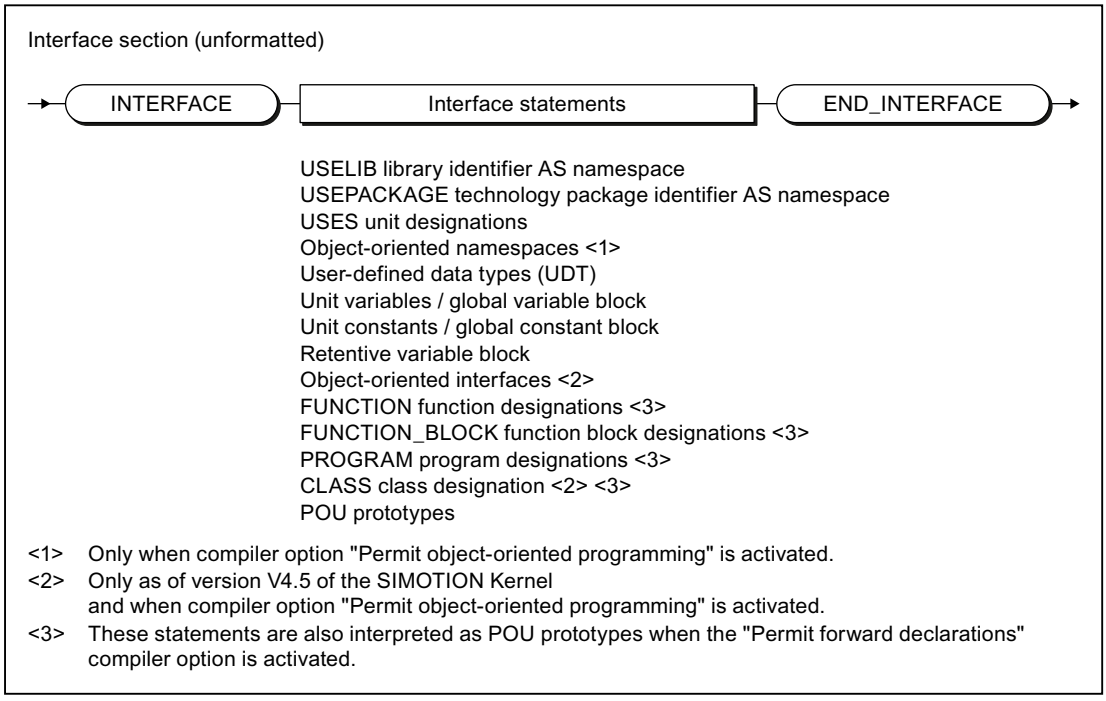


Figure 7-340 Interface section

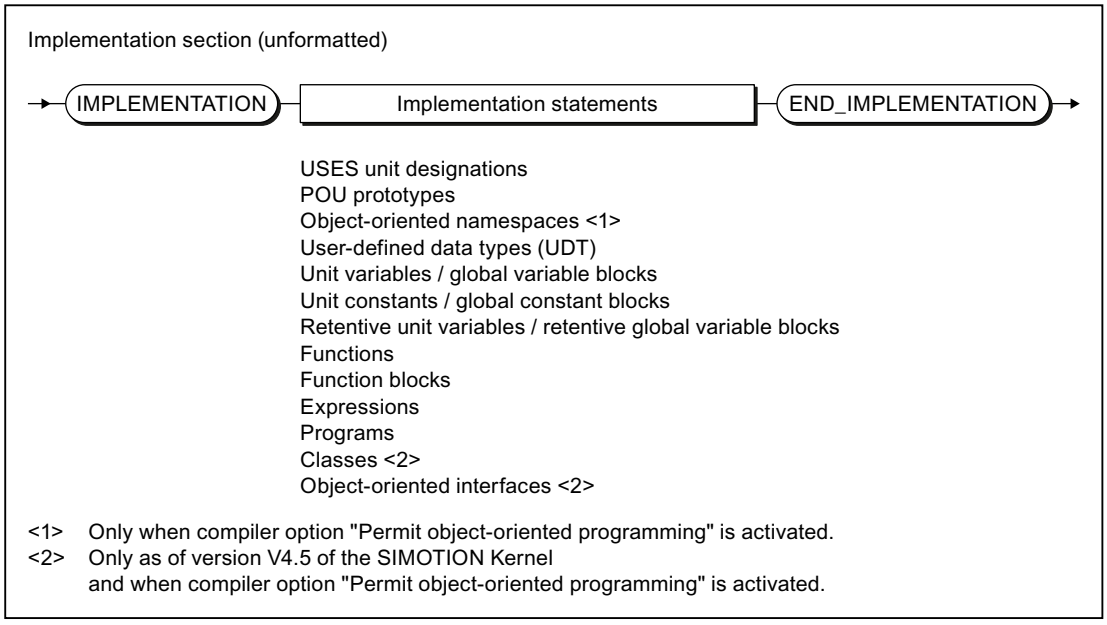


Figure 7-341 Implementation section

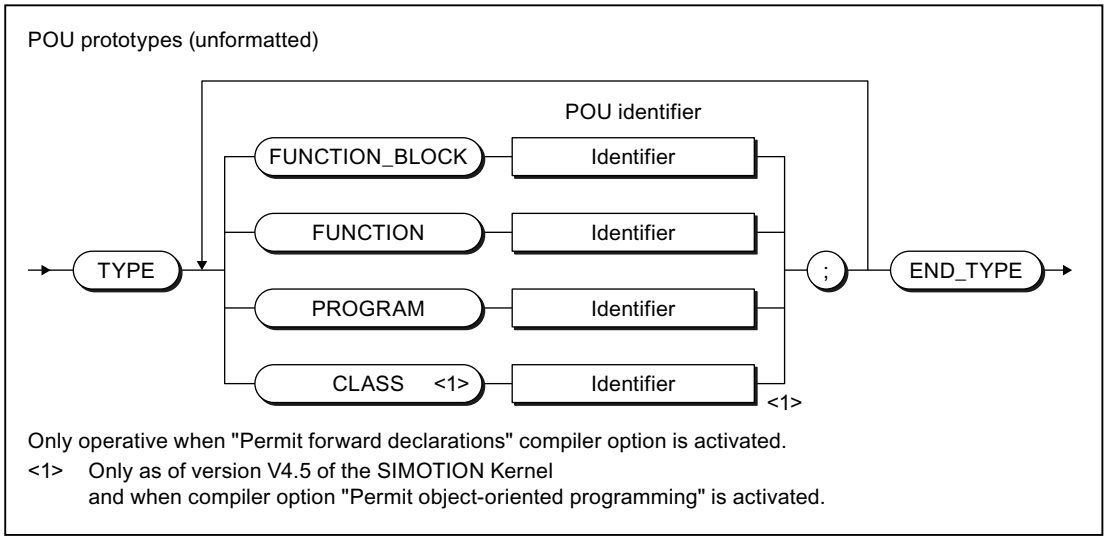


Figure 7-342 POU prototypes

Program organization units (POU)

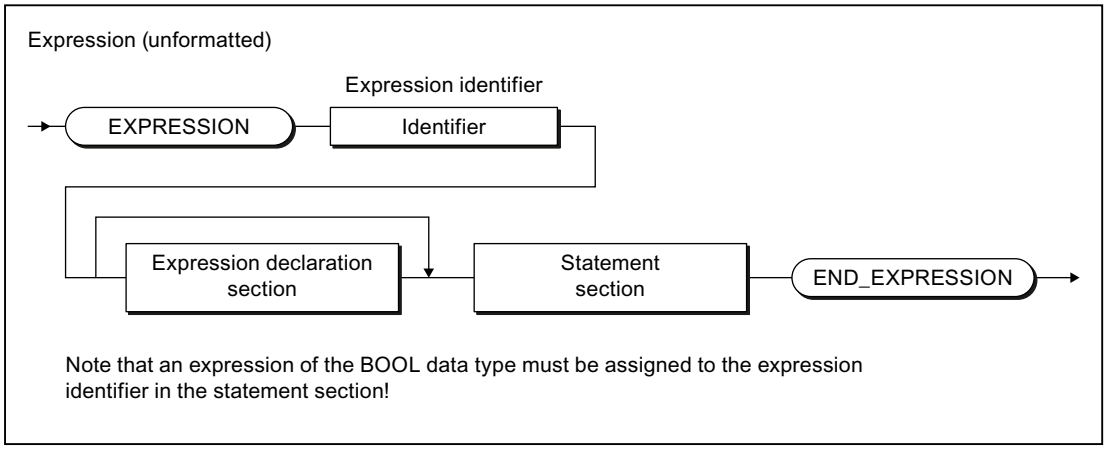


Figure 7-343 Expression

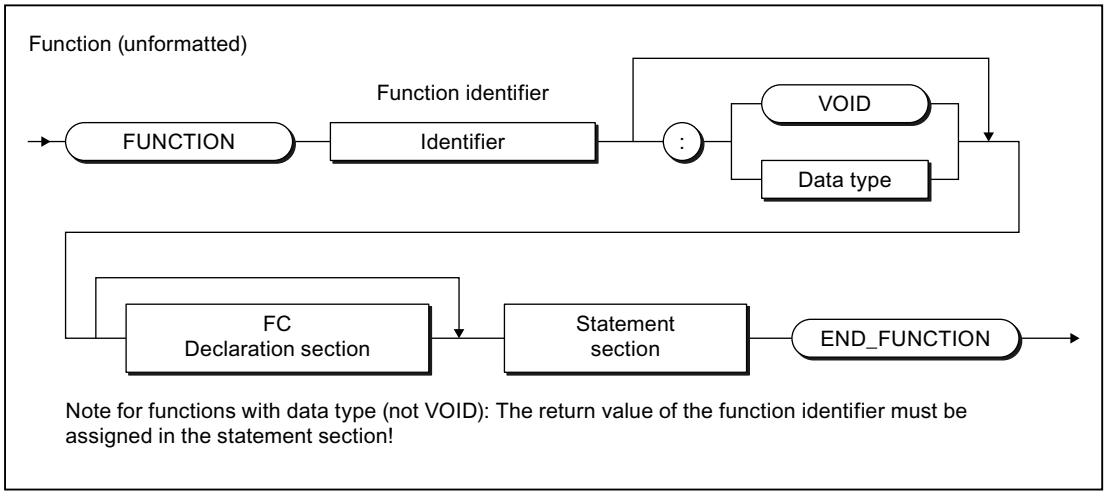


Figure 7-344 Function (FC)

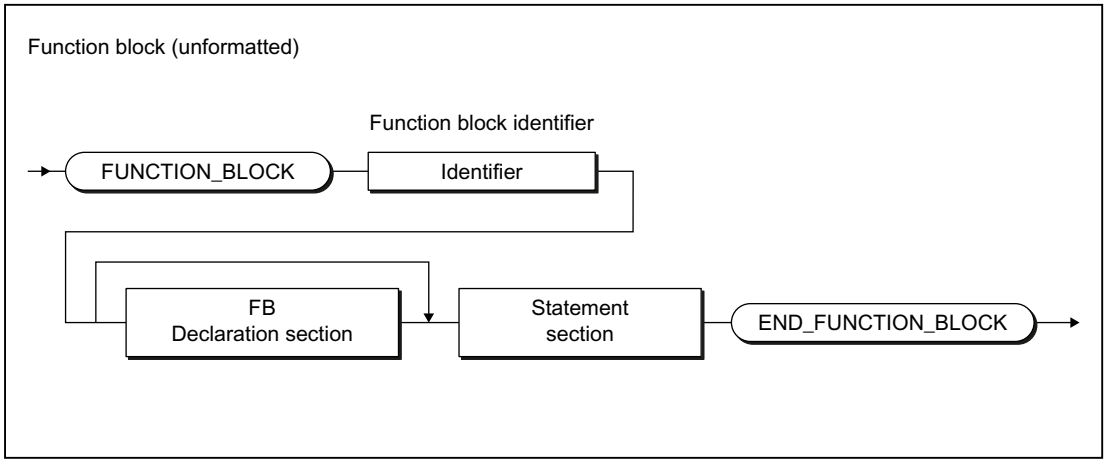


Figure 7-345 Function block (FB)

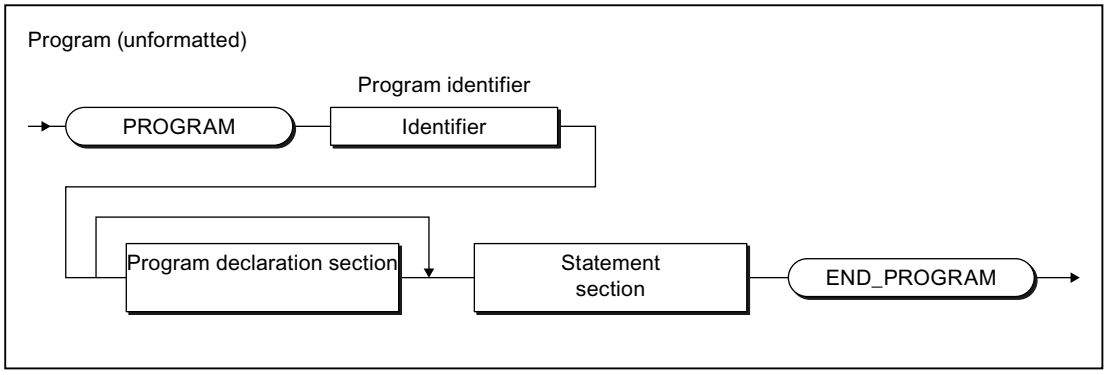


Figure 7-346 Program

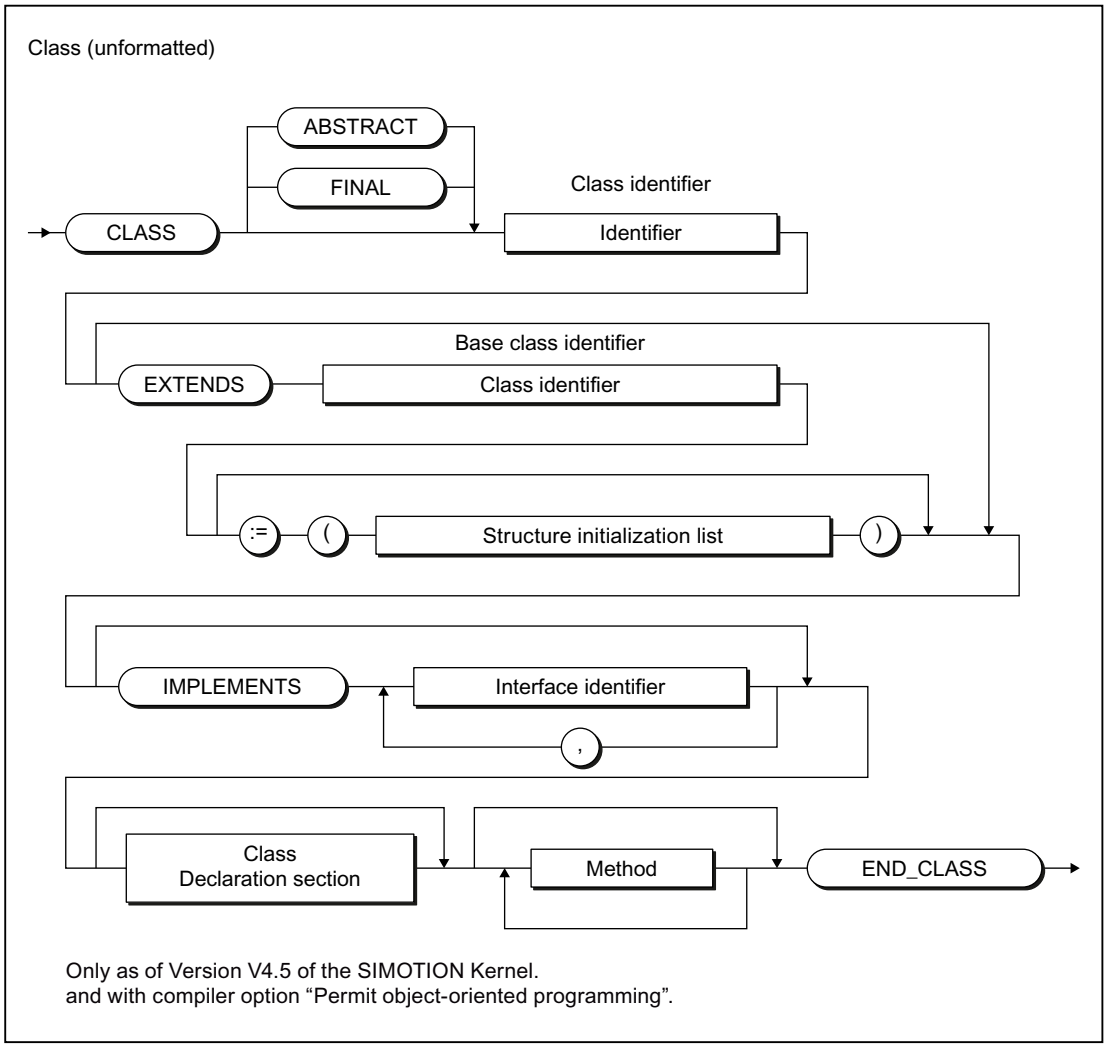


Figure 7-347 Syntax: Class

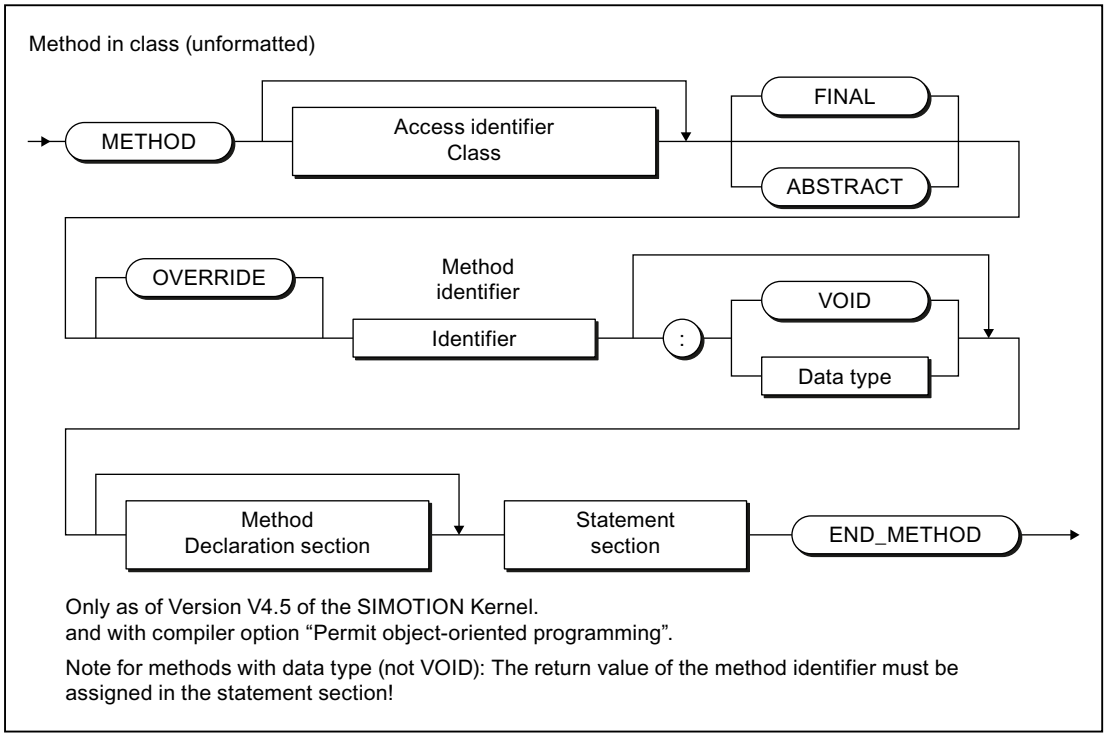


Figure 7-348 Syntax: Method

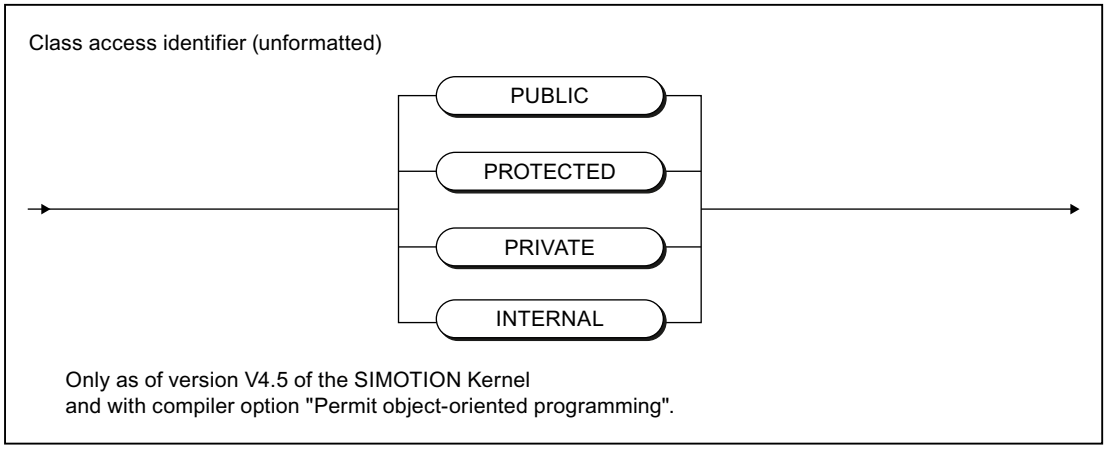


Figure 7-349 Syntax: Class access identifier

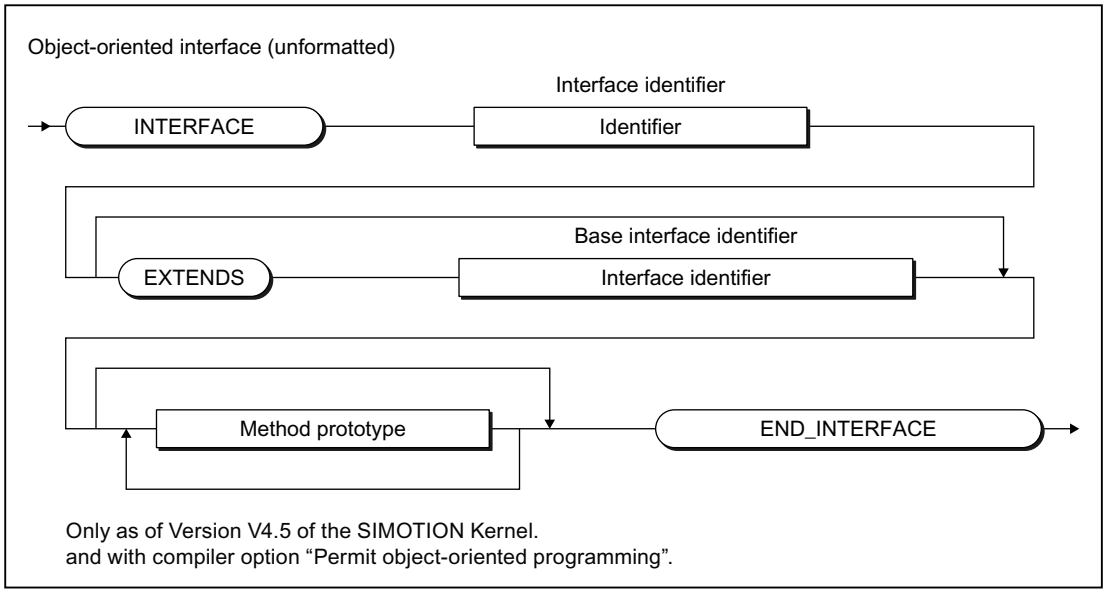


Figure 7-350 Syntax: Object-oriented interface

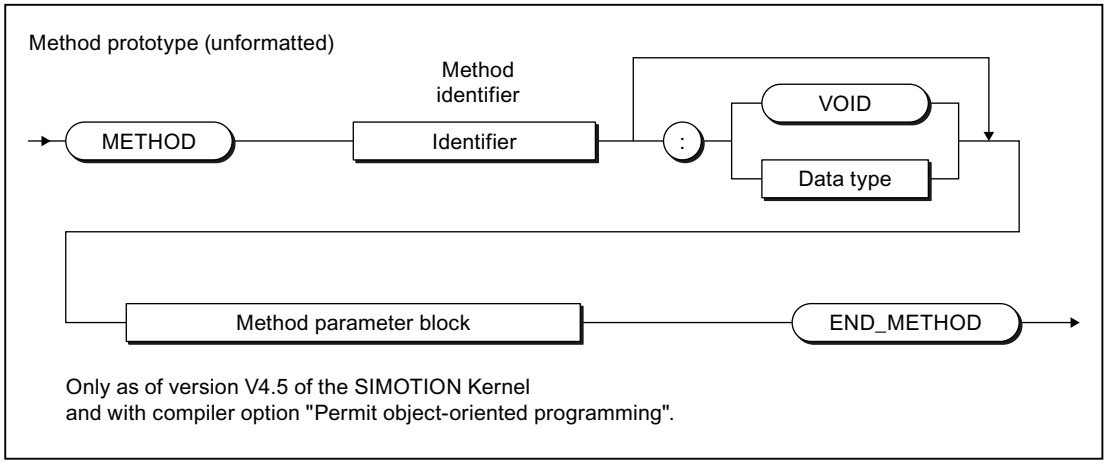


Figure 7-351 Syntax: Method prototype

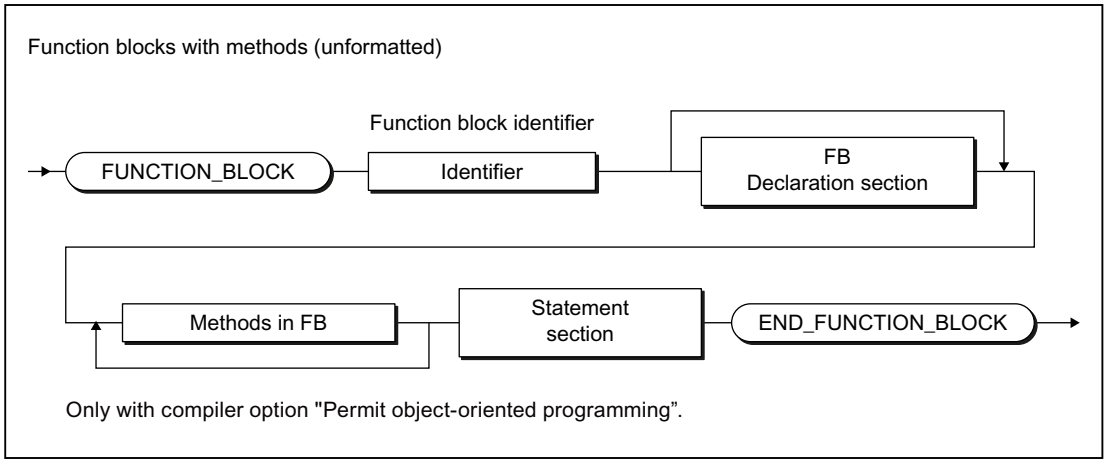


Figure 7-352 Syntax: Function block (FB) with methods

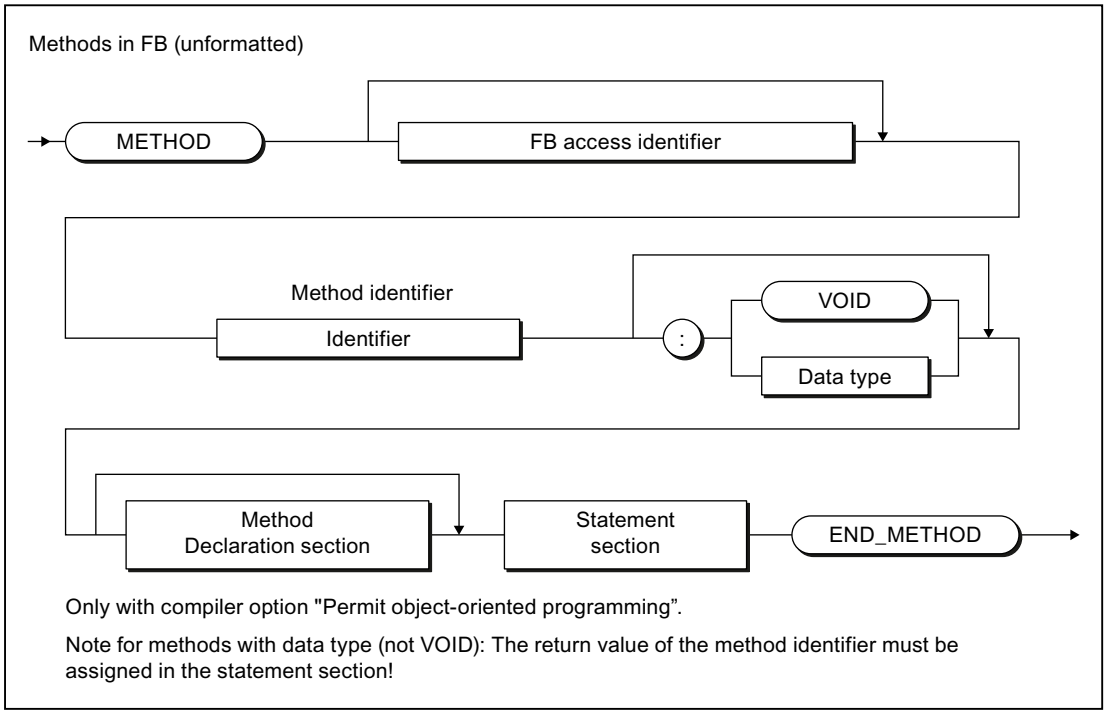


Figure 7-353 Syntax: Methods in the function block

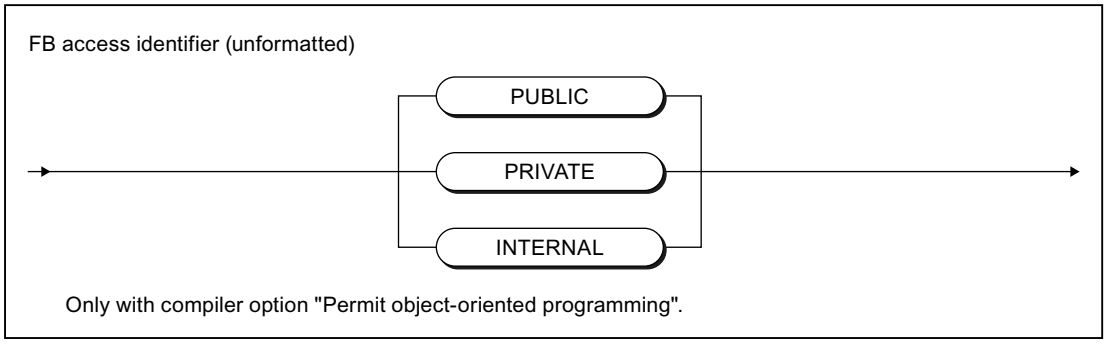


Figure 7-354 Syntax: FB access identifier

Declaration sections

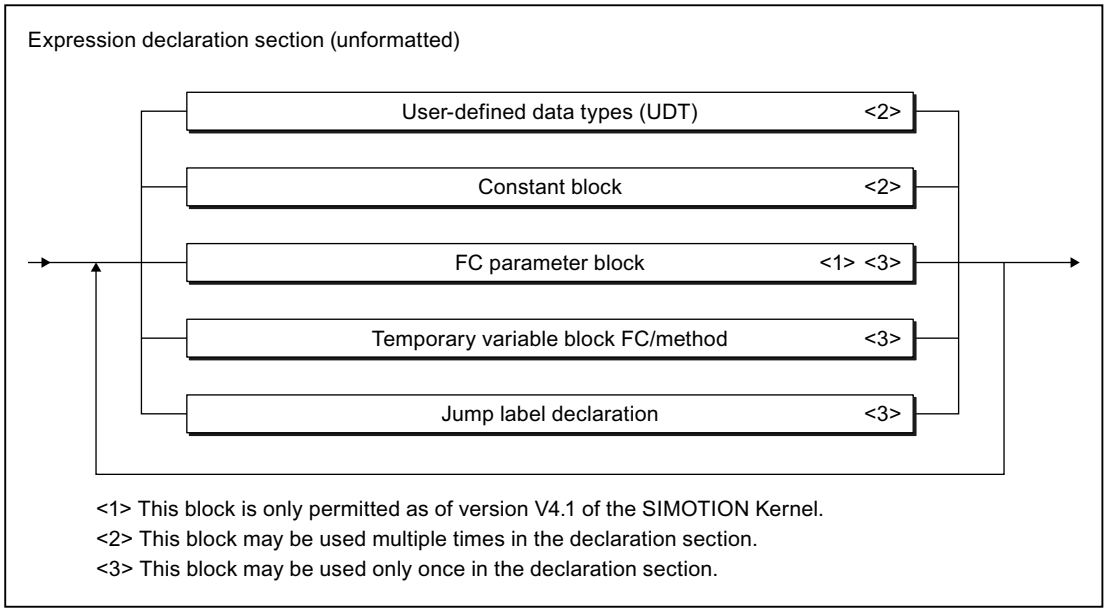


Figure 7-355 Expression declaration section

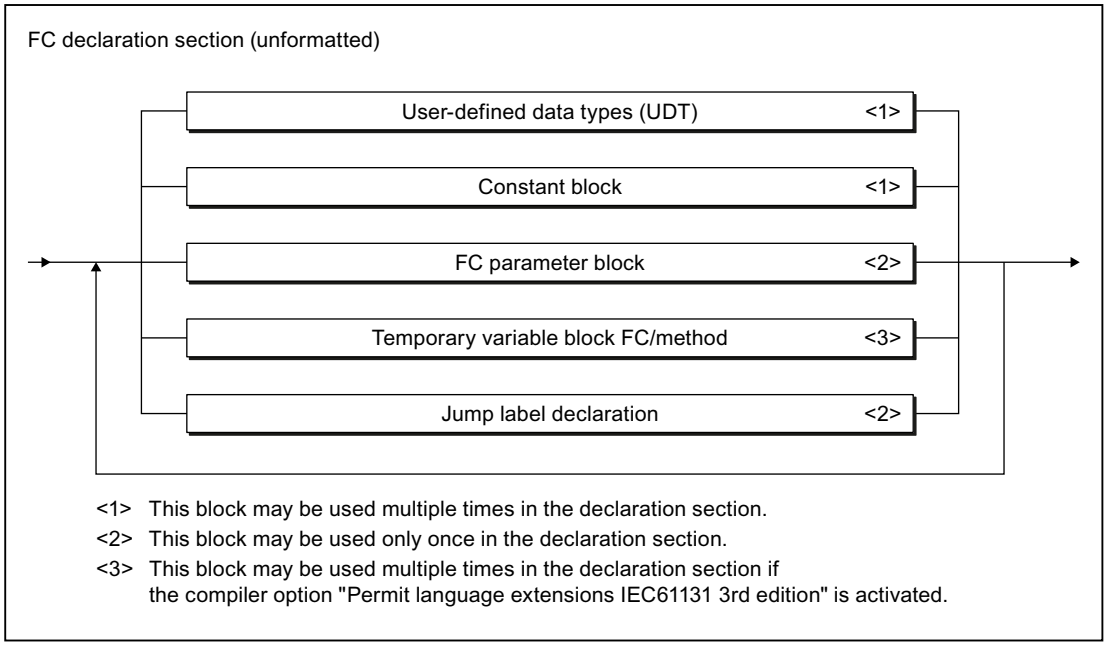


Figure 7-356 FC declaration section

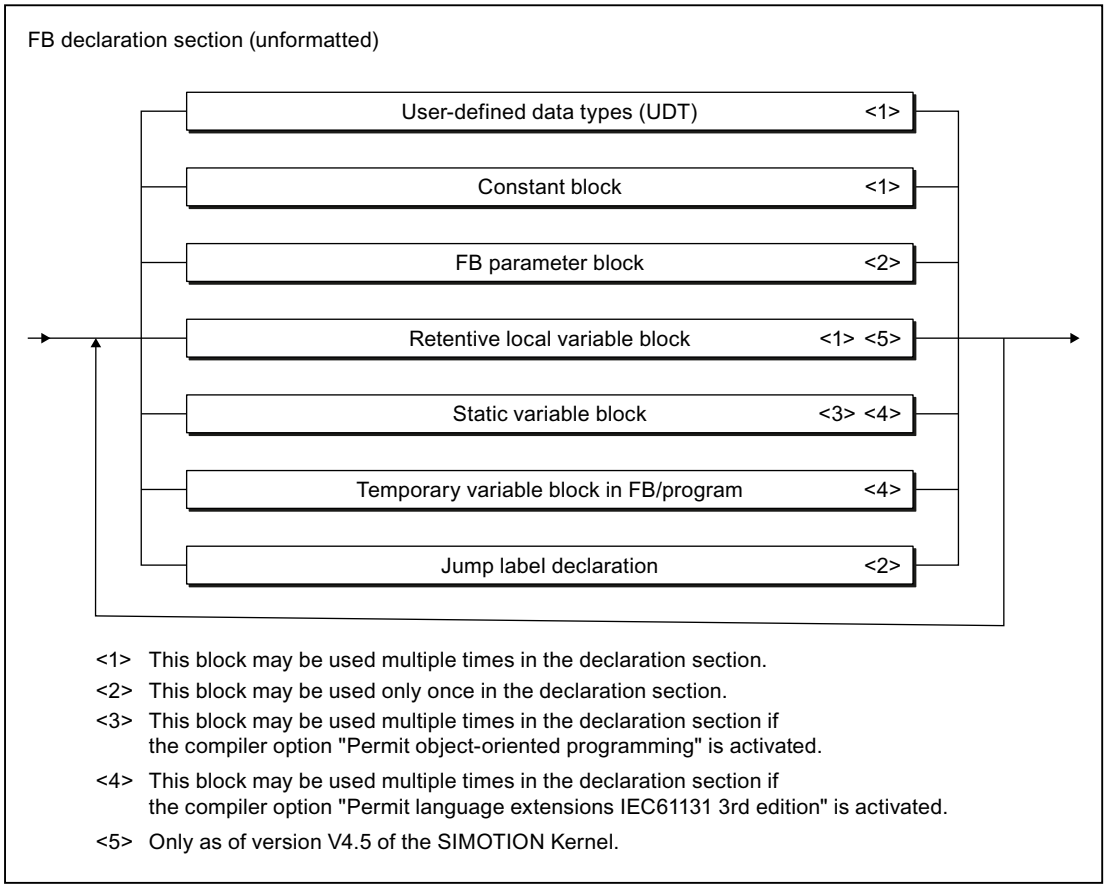


Figure 7-357 FB declaration section

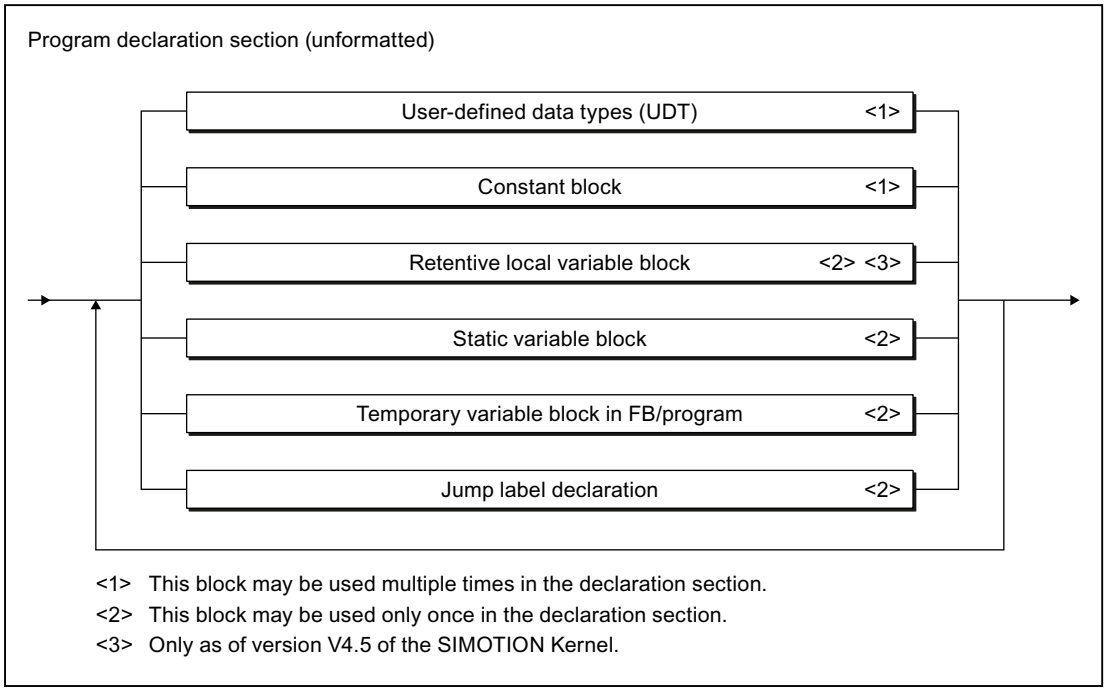


Figure 7-358 Program declaration section

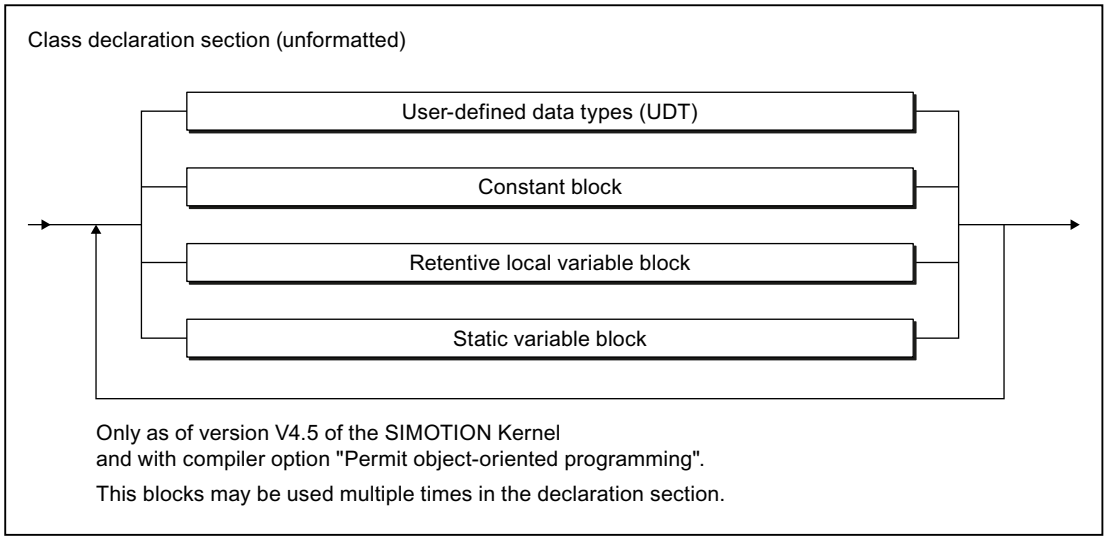


Figure 7-359 Class declaration section

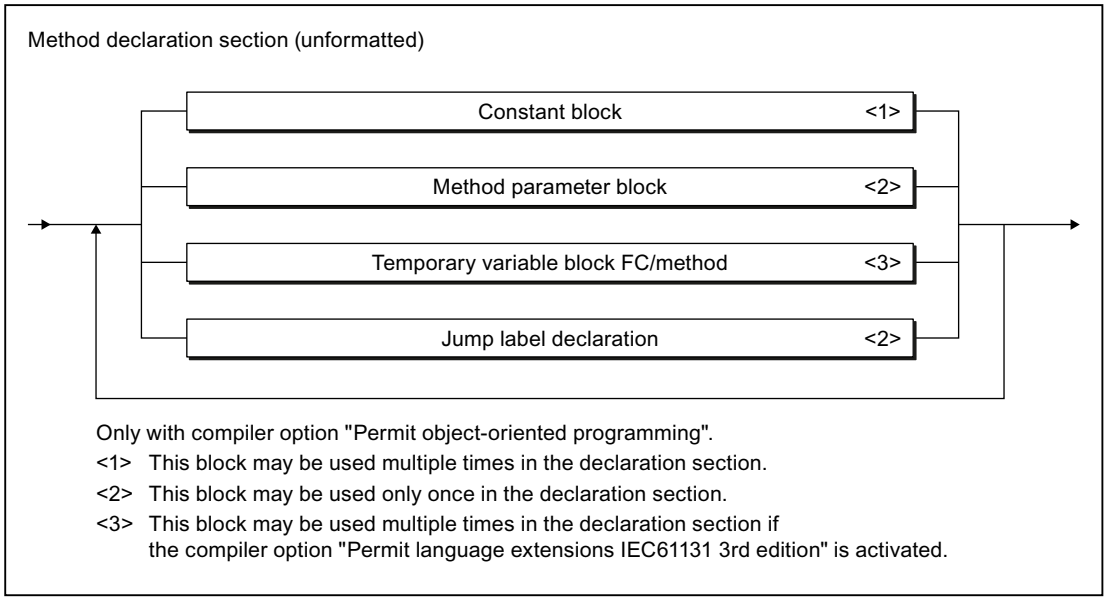


Figure 7-360 Method declaration section

Structure of the declaration blocks

Constant blocks

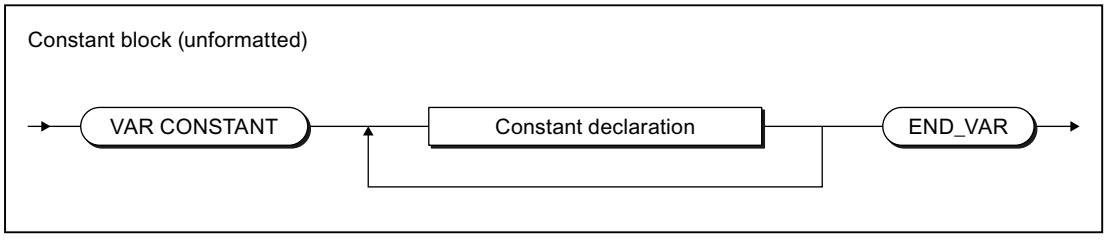


Figure 7-361 Constant block

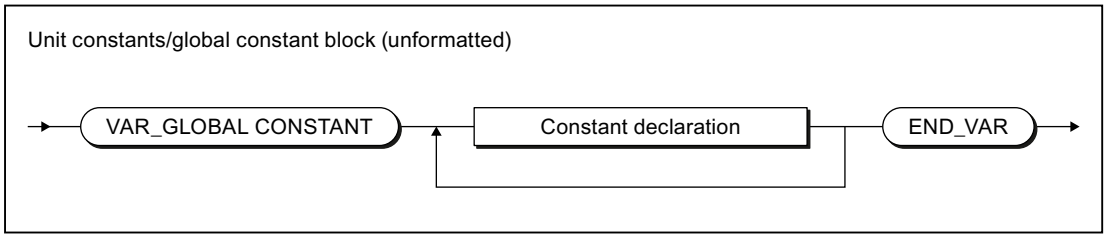


Figure 7-362 Unit constants / global constant block

Variable blocks

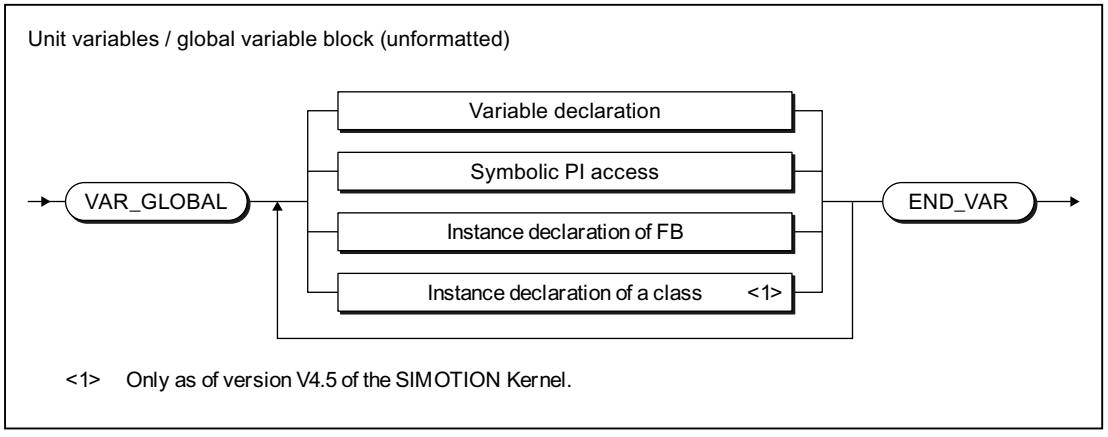


Figure 7-363 Unit variables / global variable block

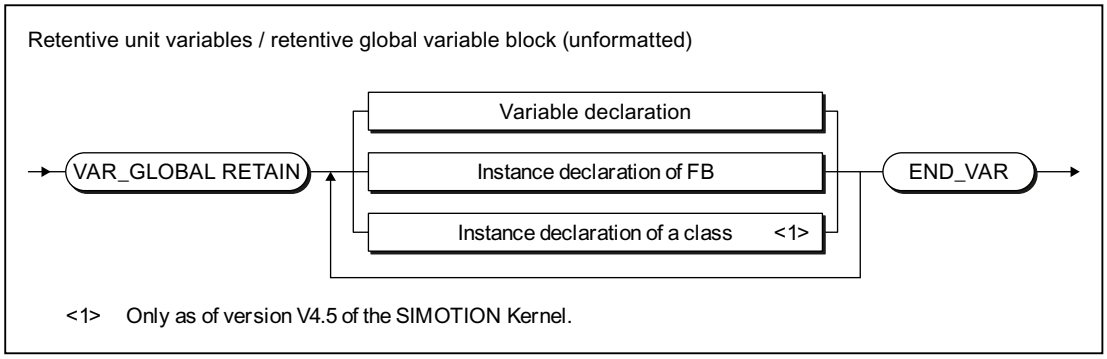


Figure 7-364 Retentive global variable block

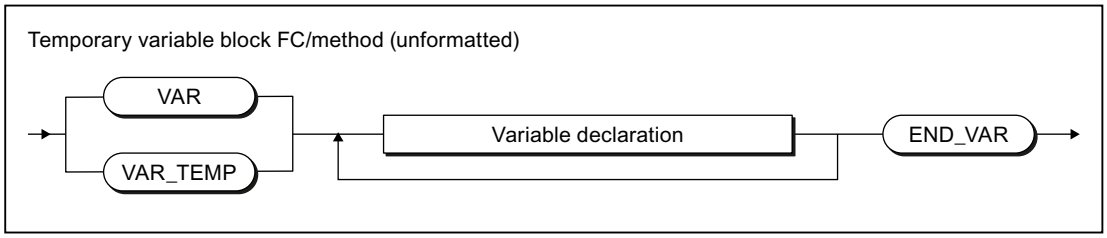


Figure 7-365 Temporary variable block FC/method

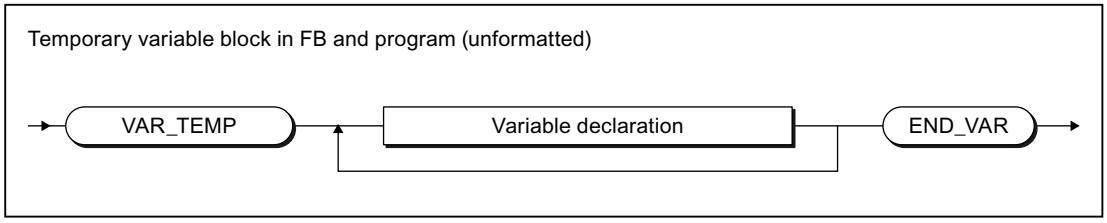


Figure 7-366 Temporary variable block in the FB/program

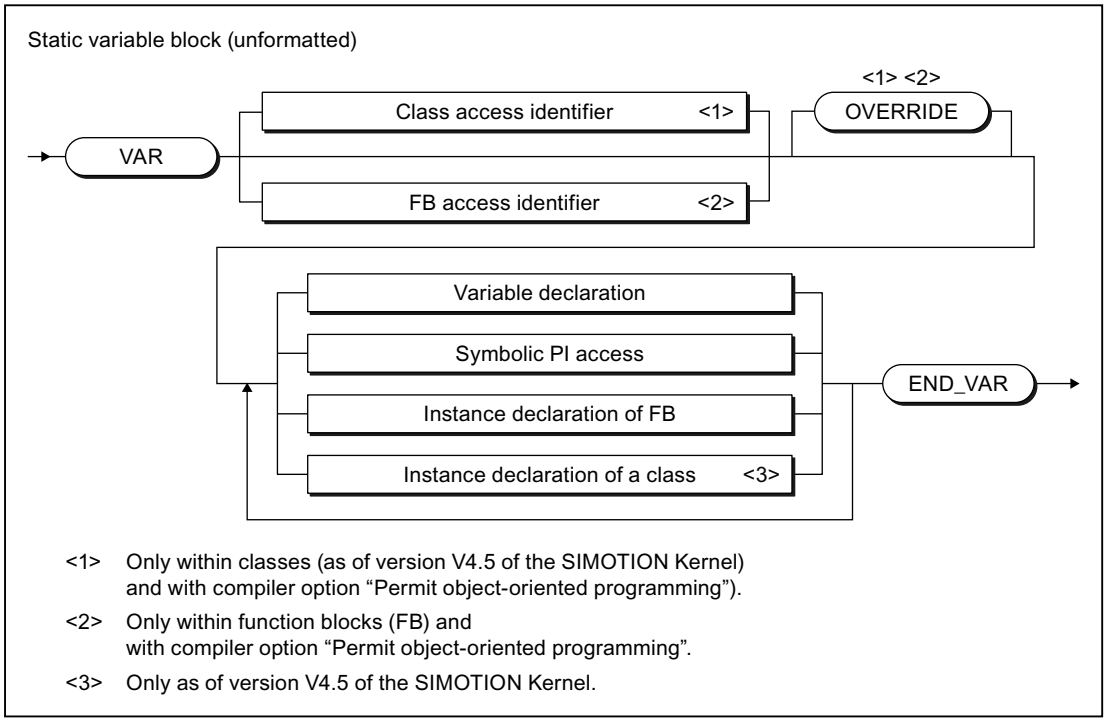


Figure 7-367 Static variable block

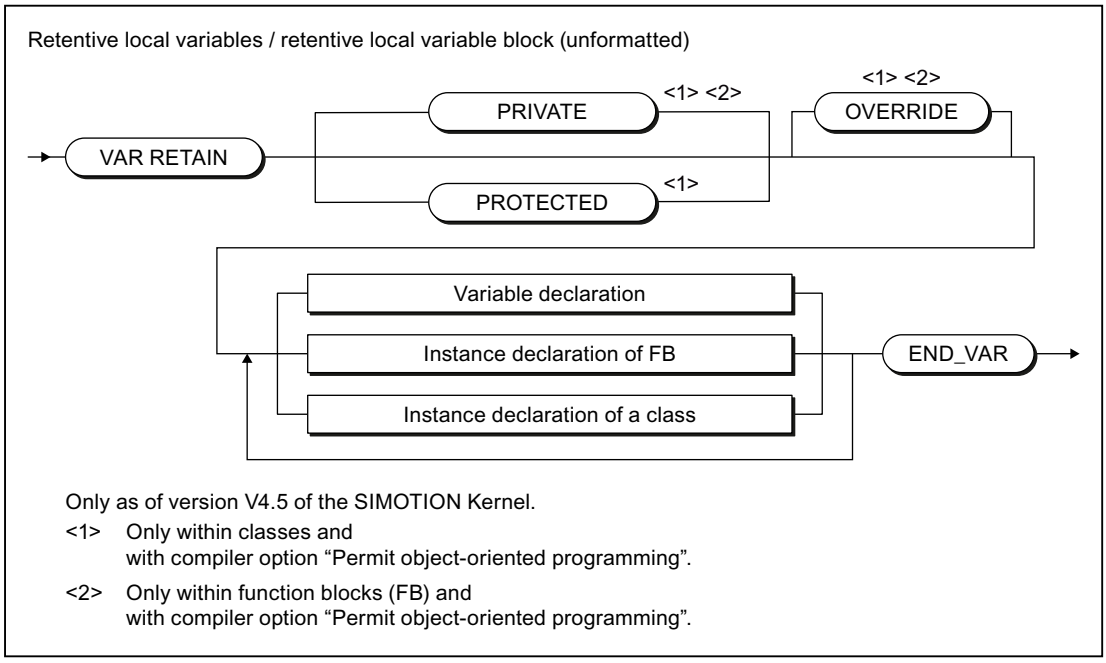


Figure 7-368 Syntax: Retentive local variable block

Parameter fields

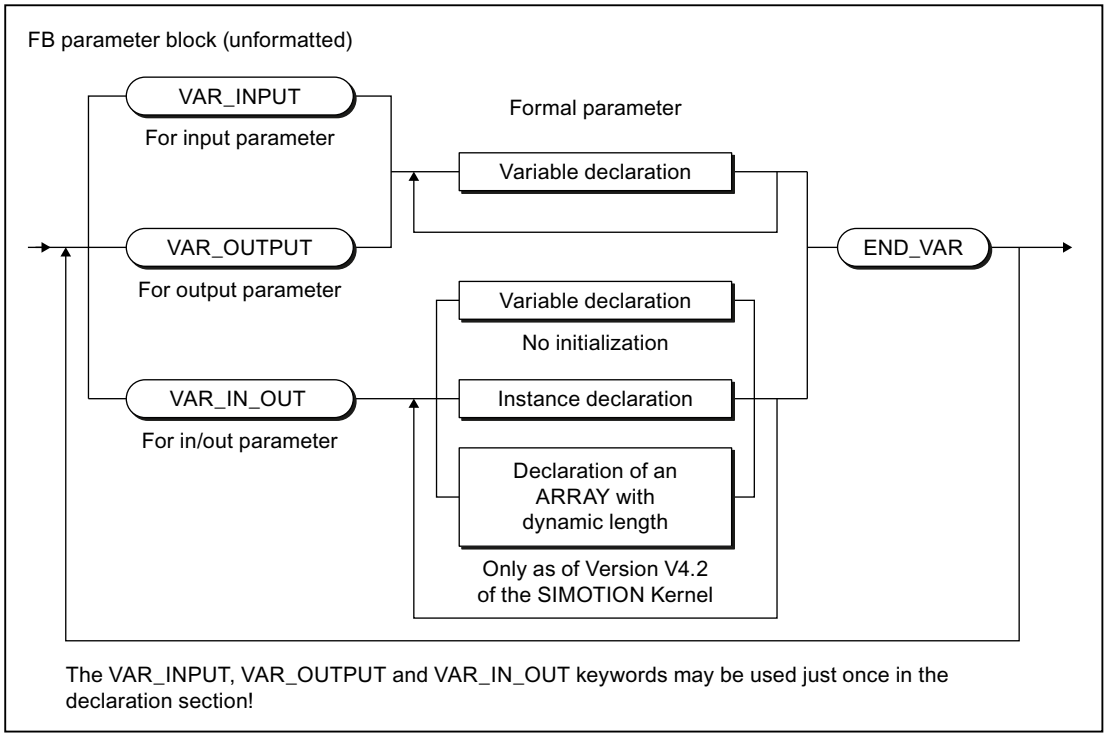


Figure 7-369 FB parameter block

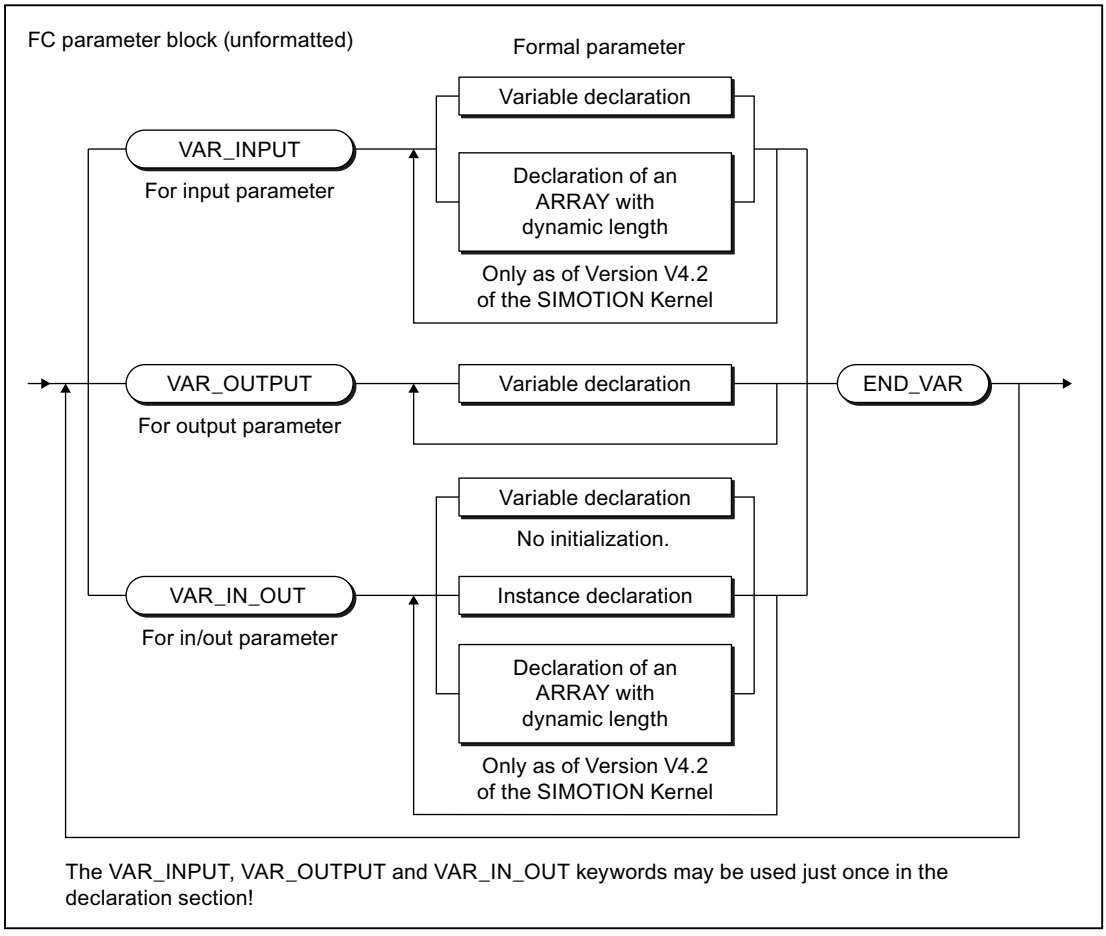


Figure 7-370 FC parameter block

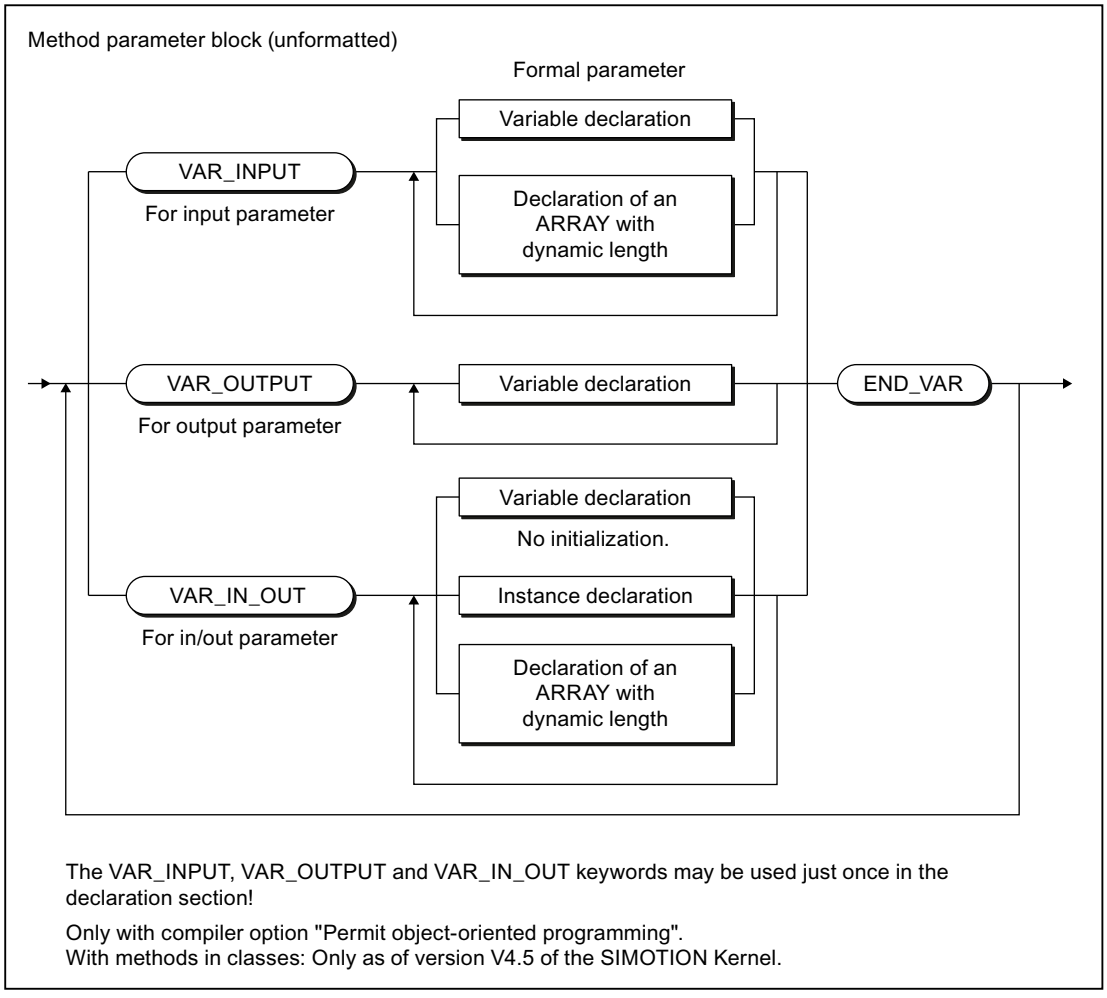


Figure 7-371 Method parameter block

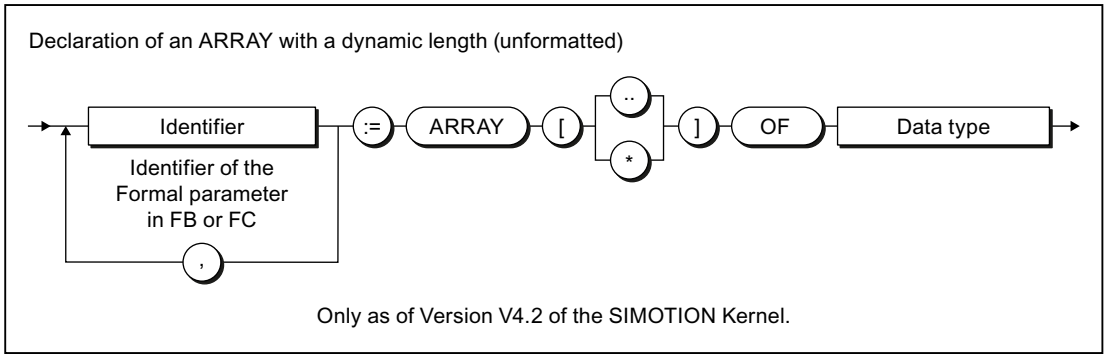


Figure 7-372 Declaration of an ARRAY with a dynamic length

Jump labels

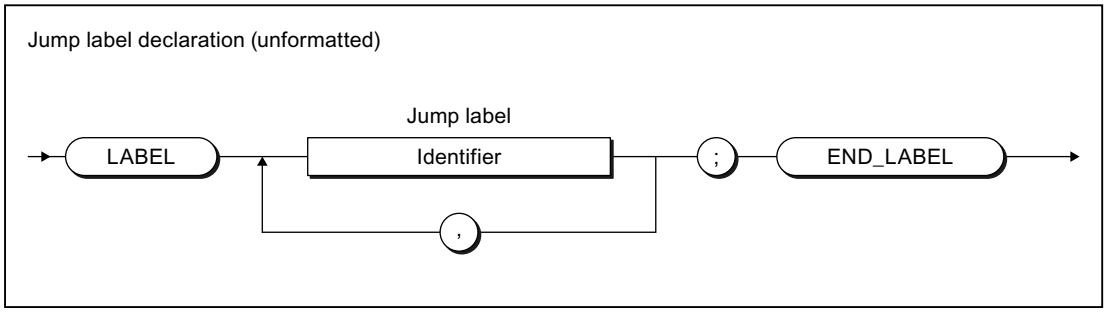


Figure 7-373 Jump label declaration

Declarations

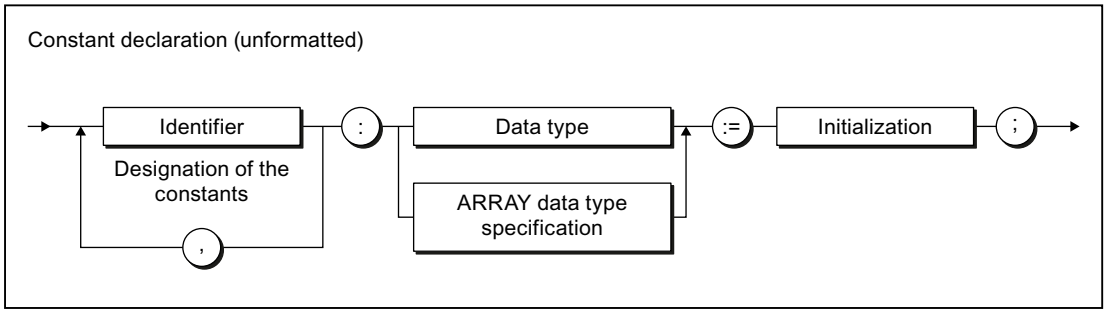


Figure 7-374 Constant declaration

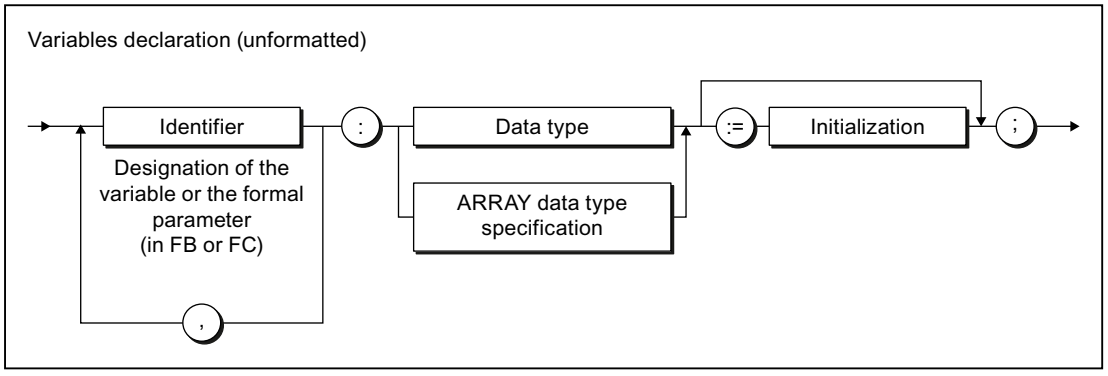


Figure 7-375 Variable declaration

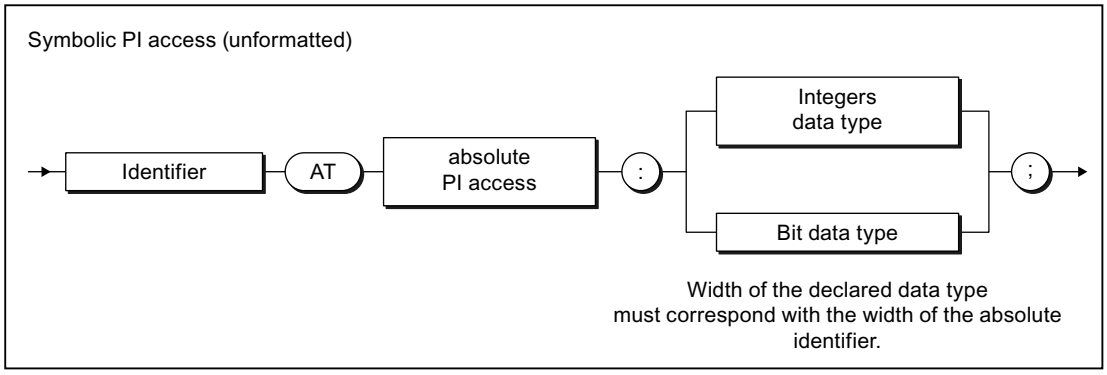


Figure 7-376 Symbolic PI access

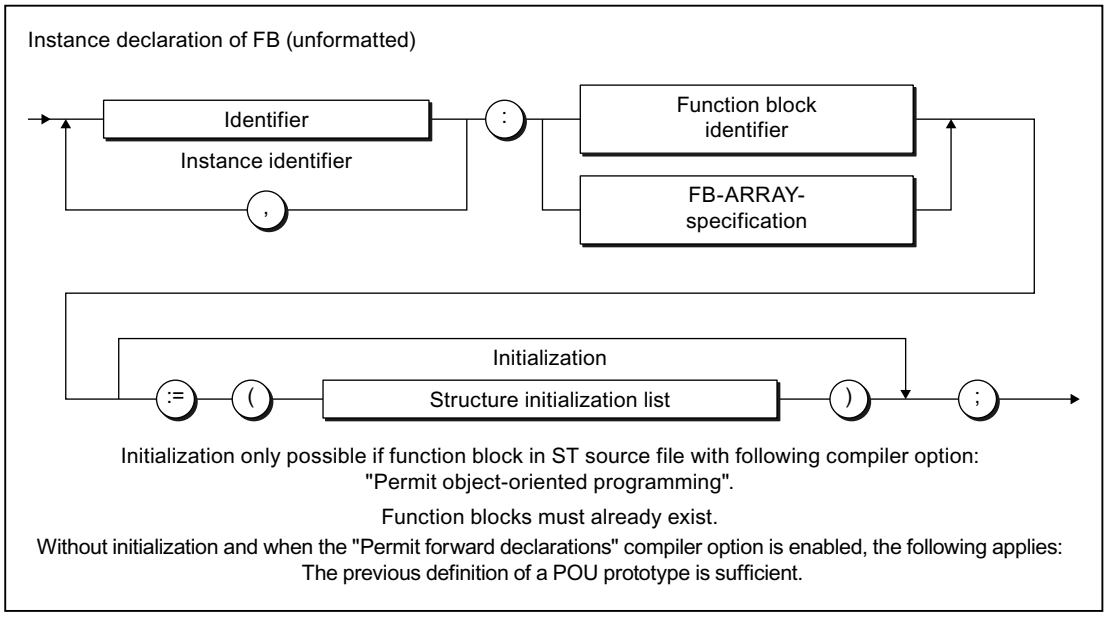


Figure 7-377 Instance declaration of FB

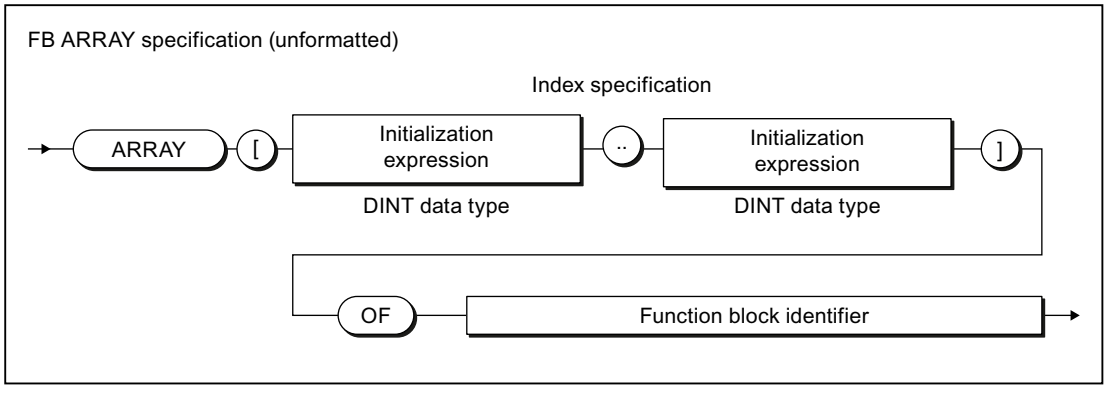


Figure 7-378 FB ARRAY specification

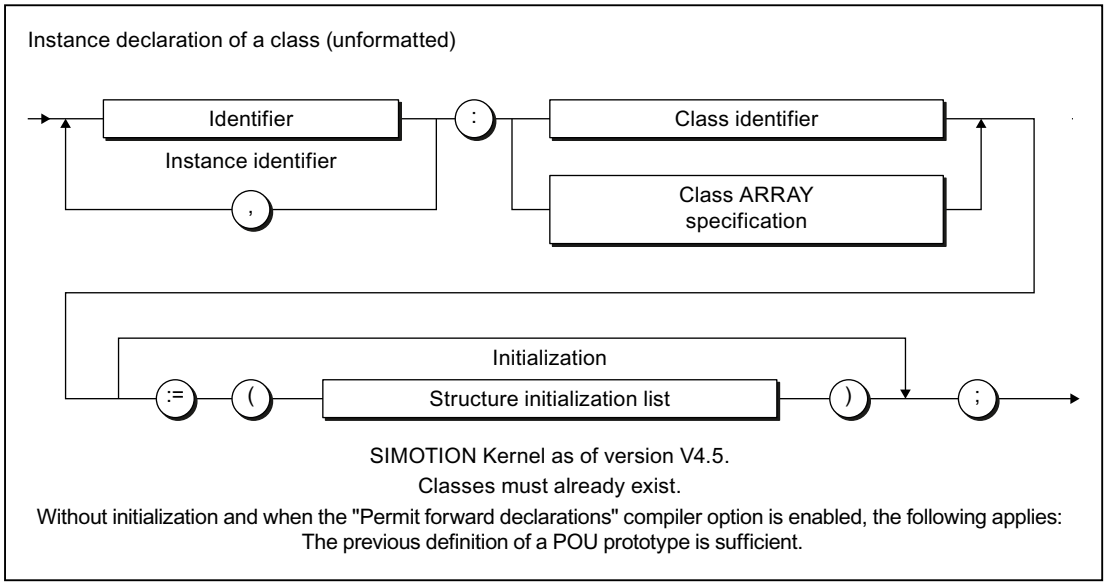


Figure 7-379 Syntax: Instance declaration of a class

Initialization

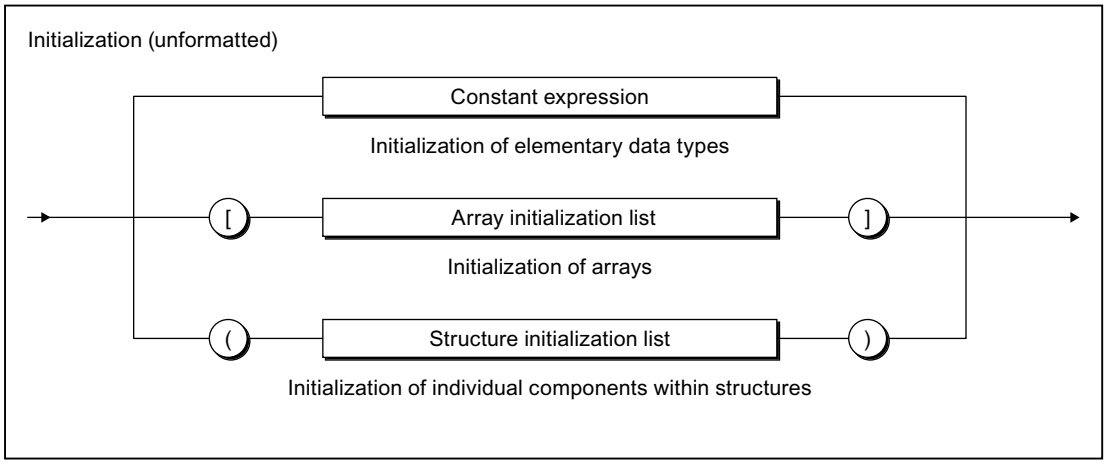


Figure 7-380 Initialization

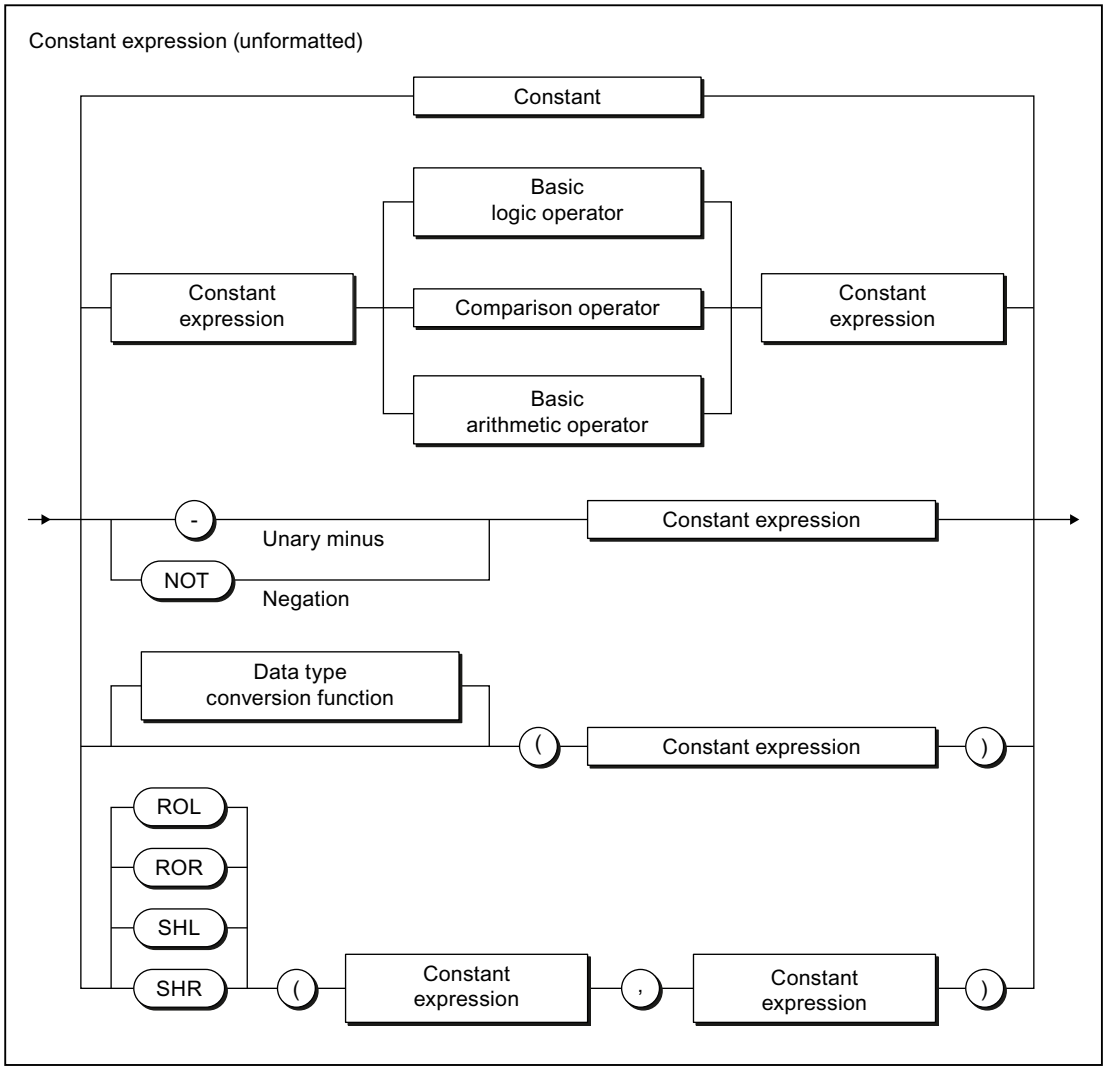


Figure 7-381 Constant expression

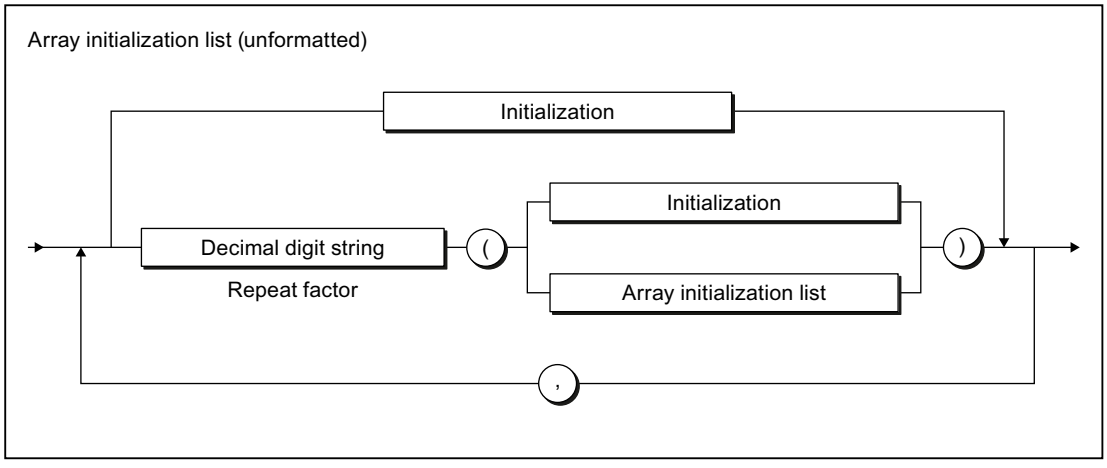


Figure 7-382 Array initialization list

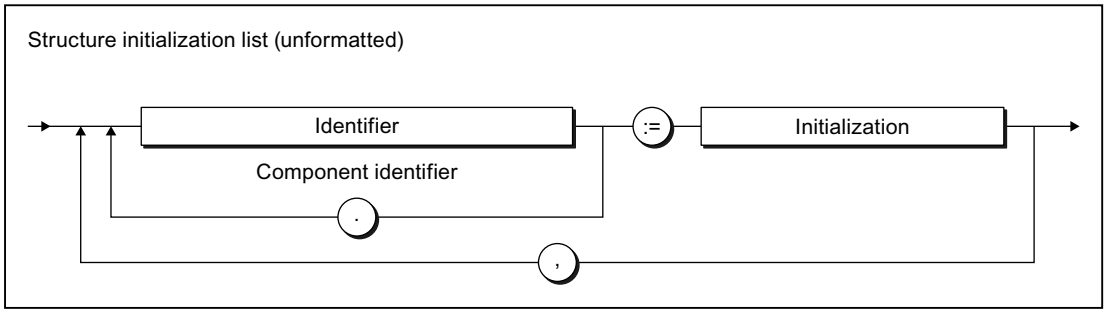


Figure 7-383 Structure initialization list

Data types

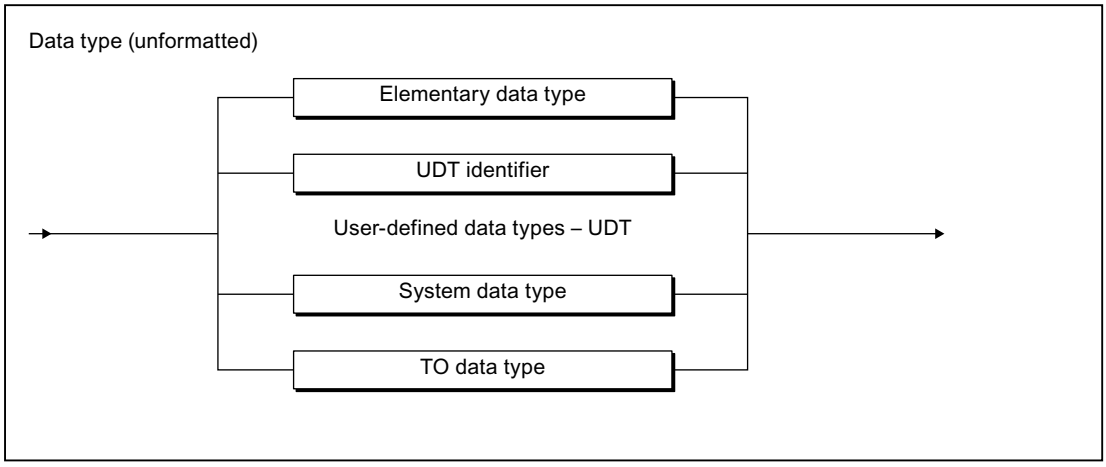


Figure 7-384 Data type

Elementary data types

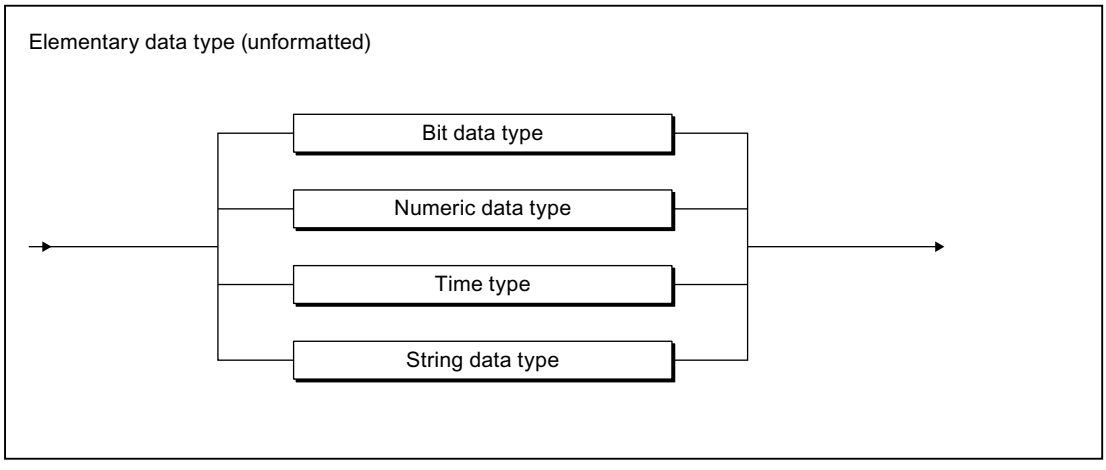


Figure 7-385 Elementary data type

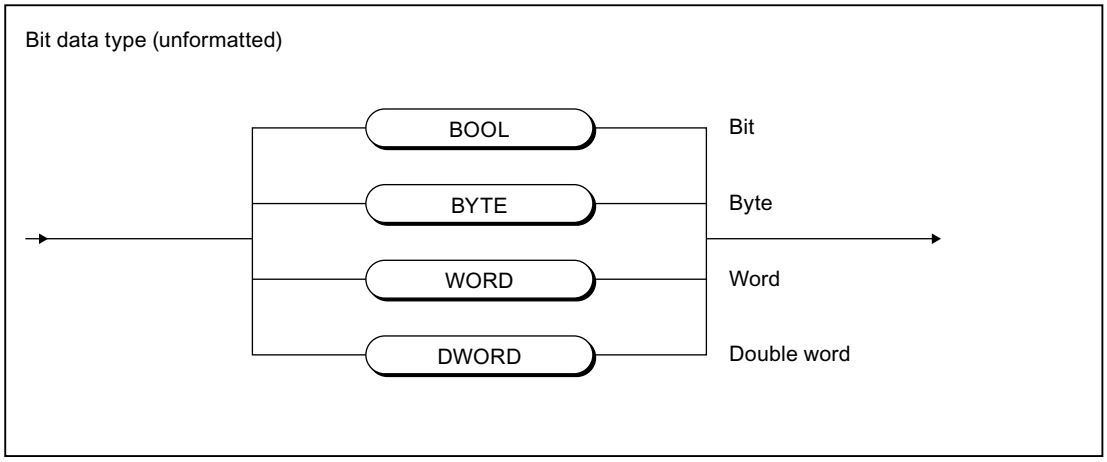


Figure 7-386 Bit data type

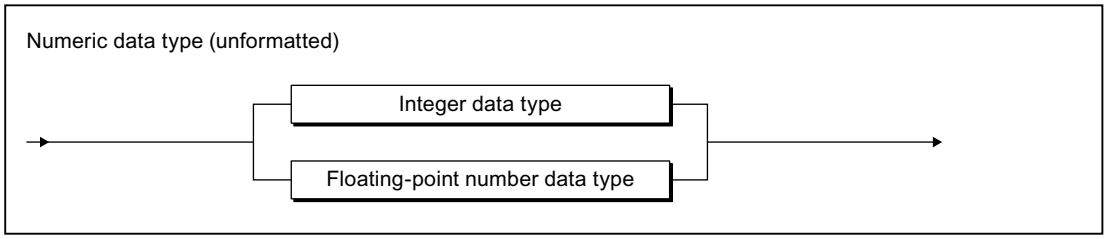


Figure 7-387 Numeric data type

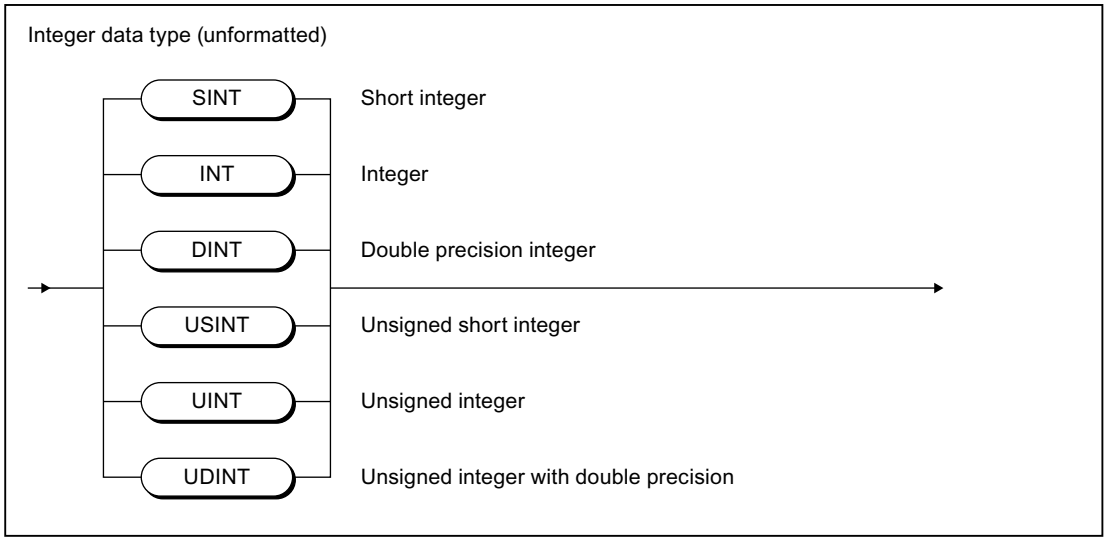


Figure 7-388 Integer data type

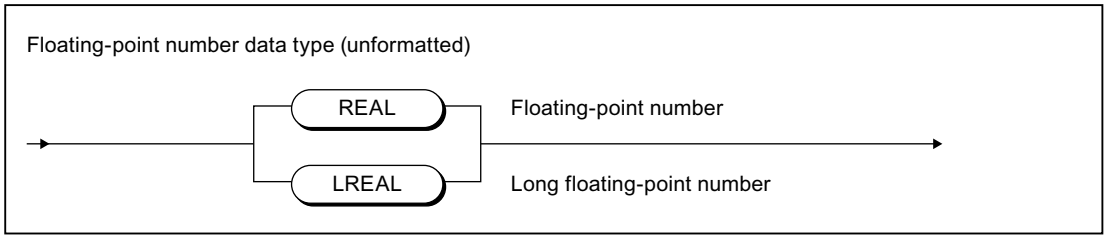


Figure 7-389 Floating-point number data type

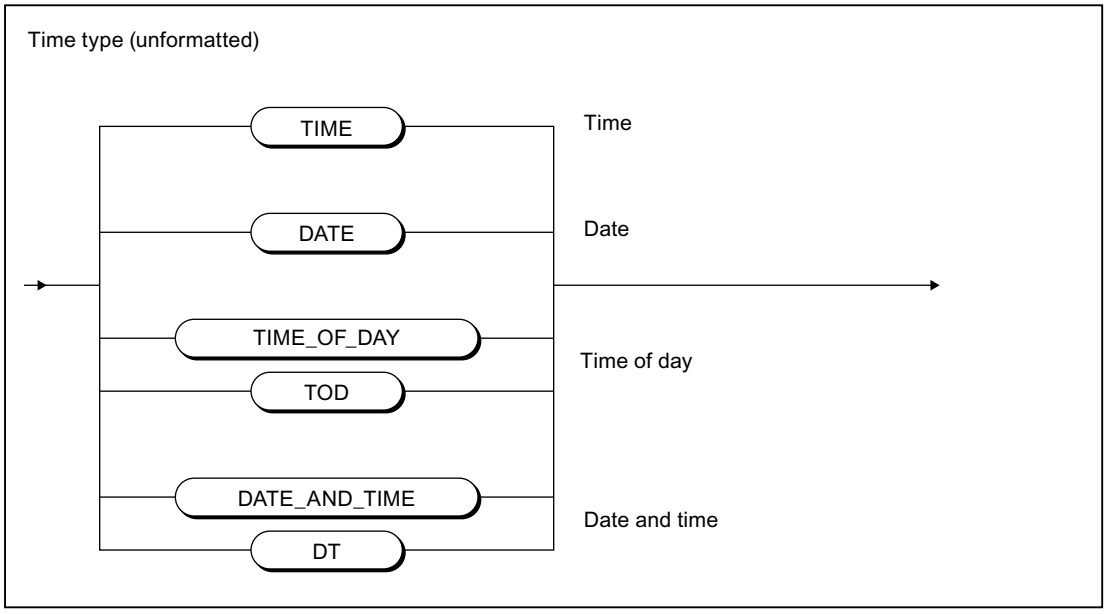


Figure 7-390 Time data type

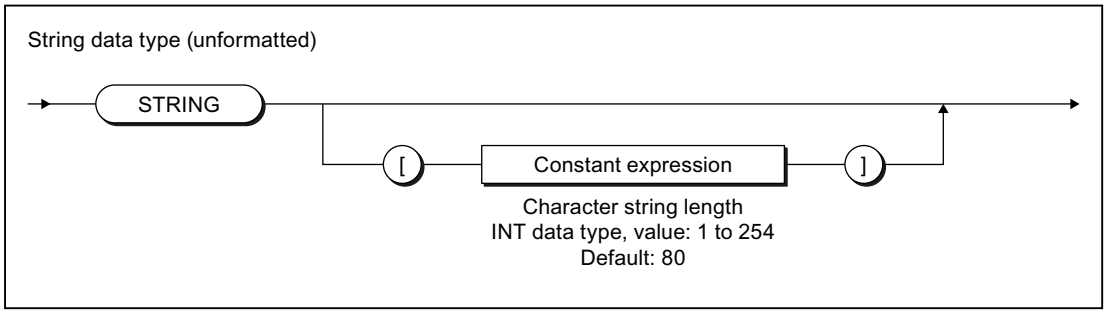


Figure 7-391 String data type

User-defined data types

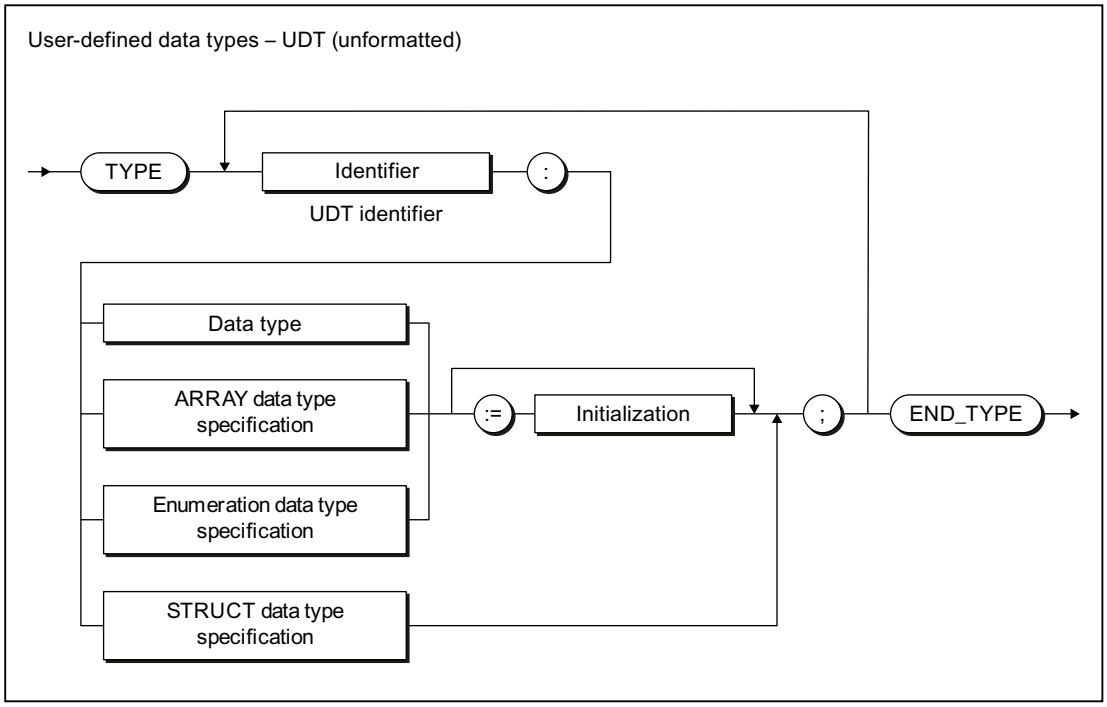


Figure 7-392 User-defined data type

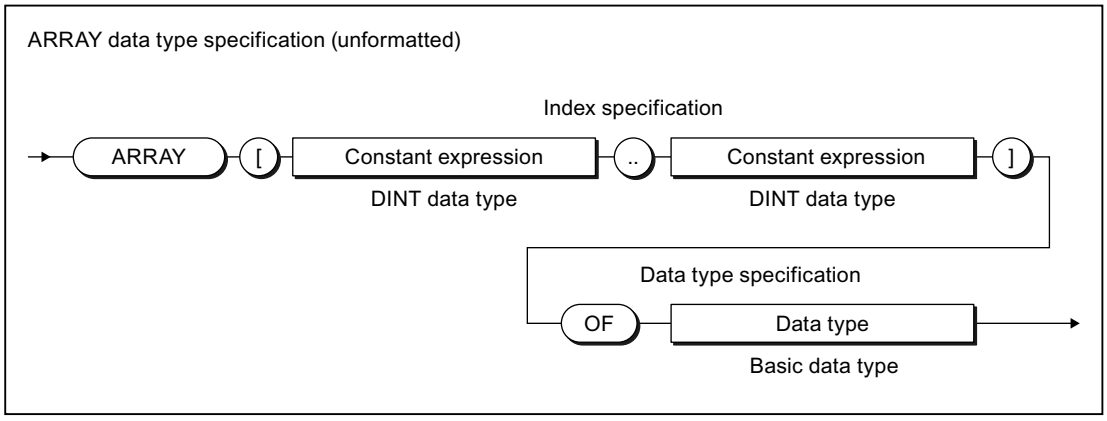


Figure 7-393 ARRAY data type specification

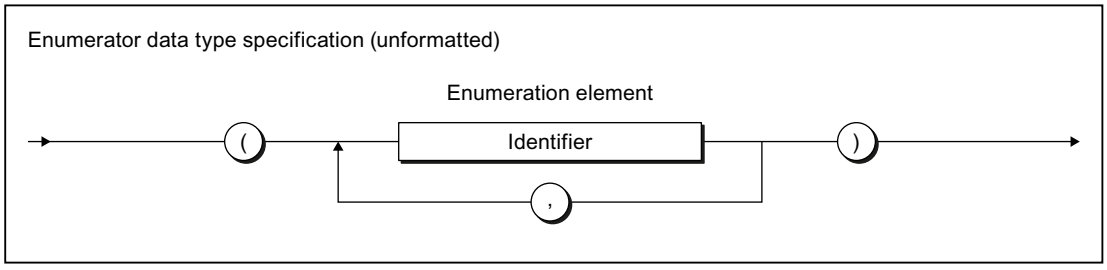


Figure 7-394 Enumeration data type specification

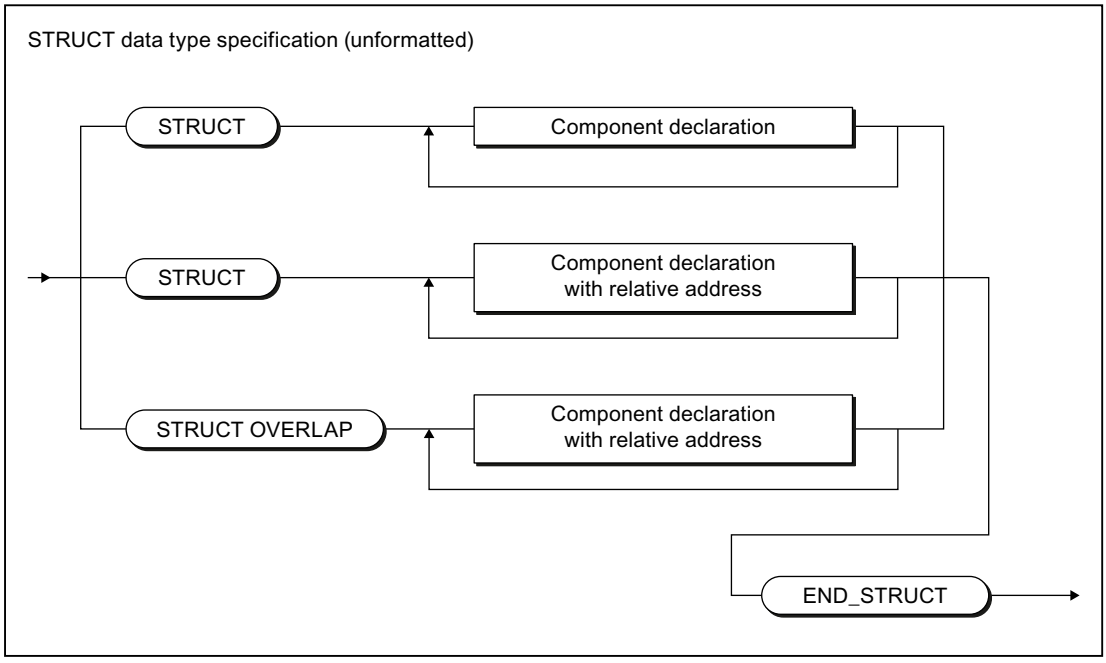


Figure 7-395 STRUCT data type specification

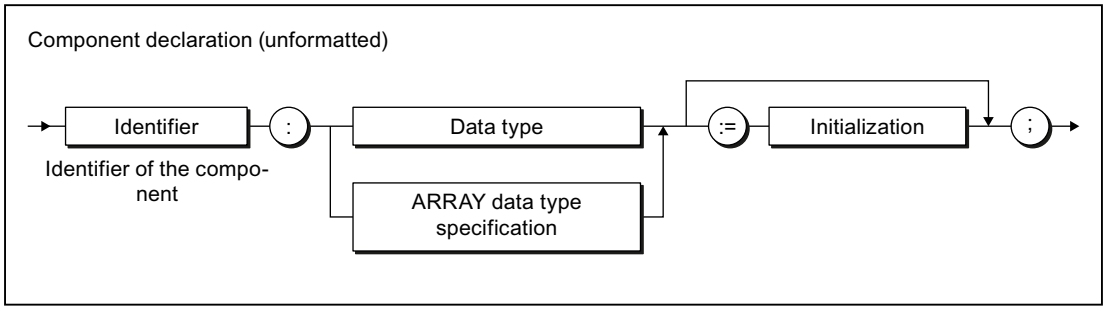


Figure 7-396 Component declaration

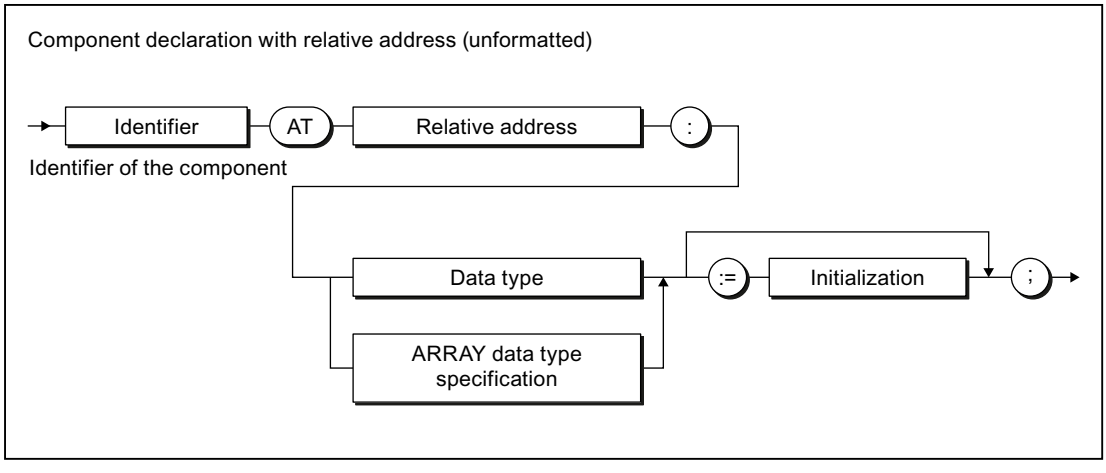


Figure 7-397 Component declaration with relative address

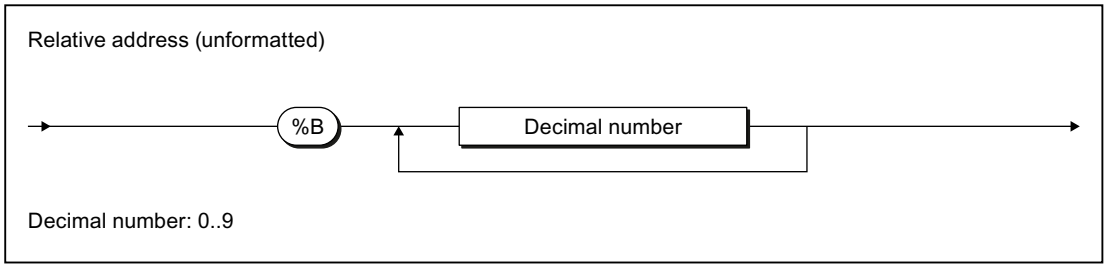


Figure 7-398 Relative address

Statement section

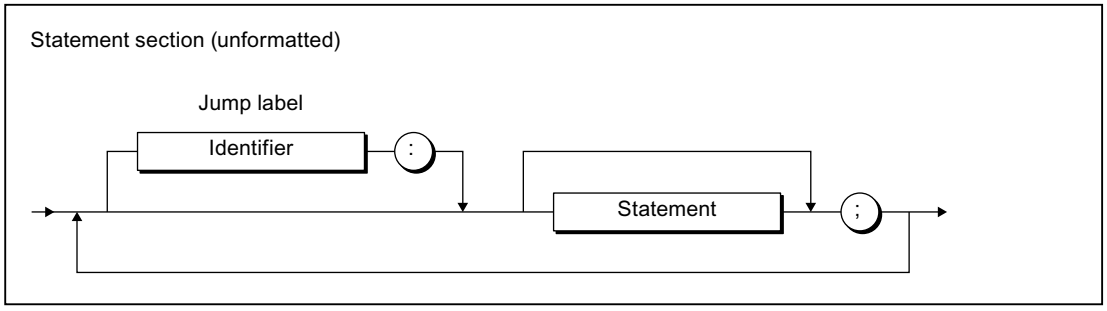


Figure 7-399 Statement section

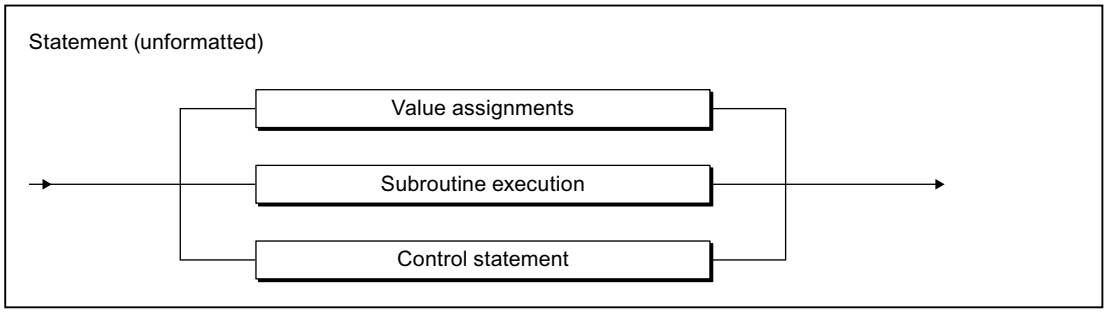


Figure 7-400 Statement

Value assignments and operations

Value assignment and expression

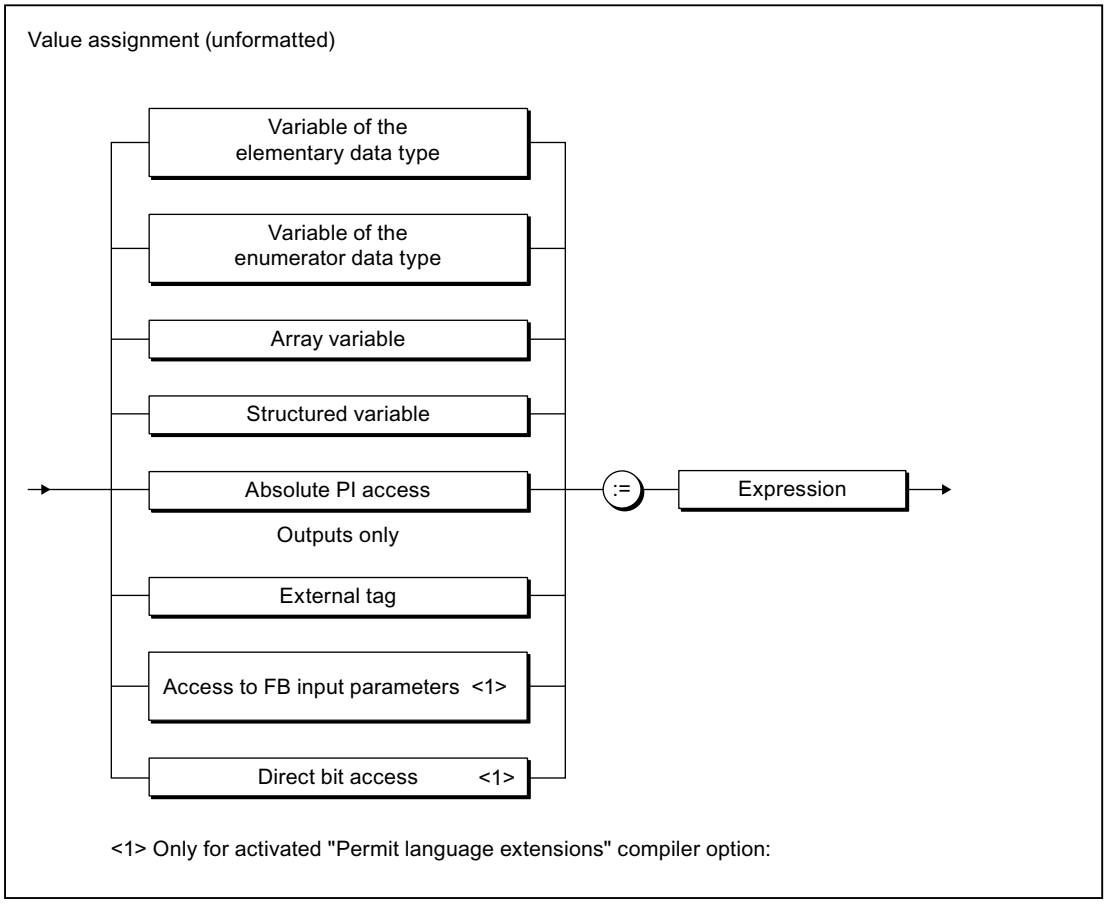


Figure 7-401 Value assignments

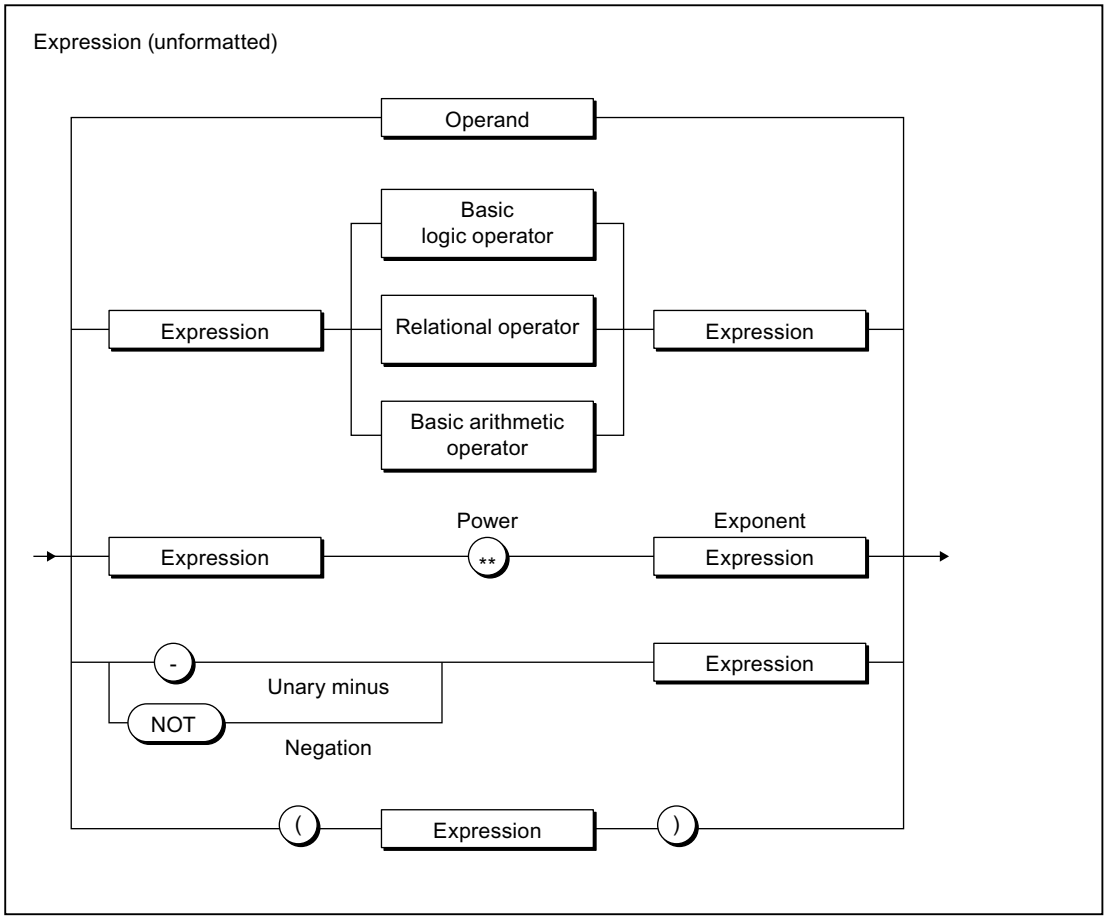


Figure 7-402 Expression

Operands

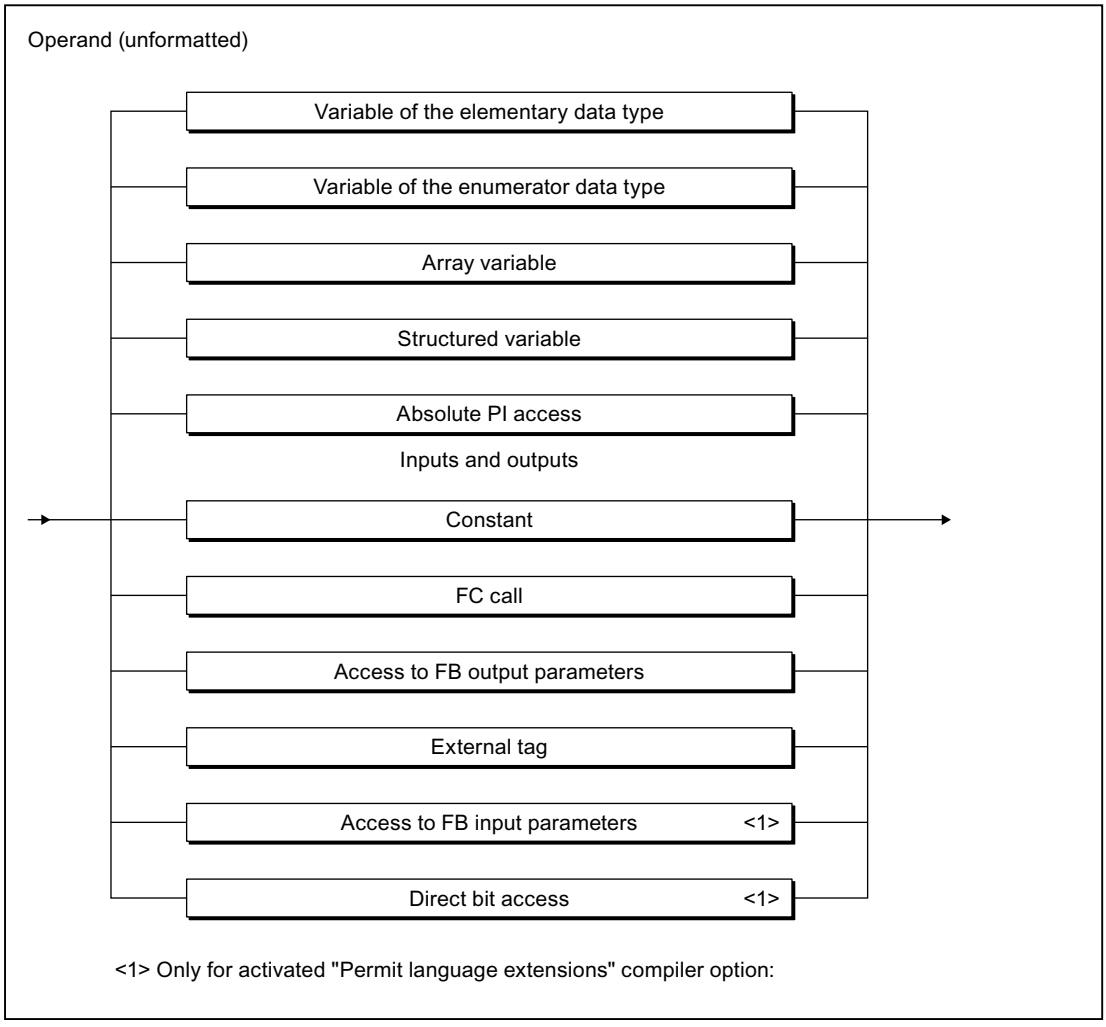


Figure 7-403 Operand

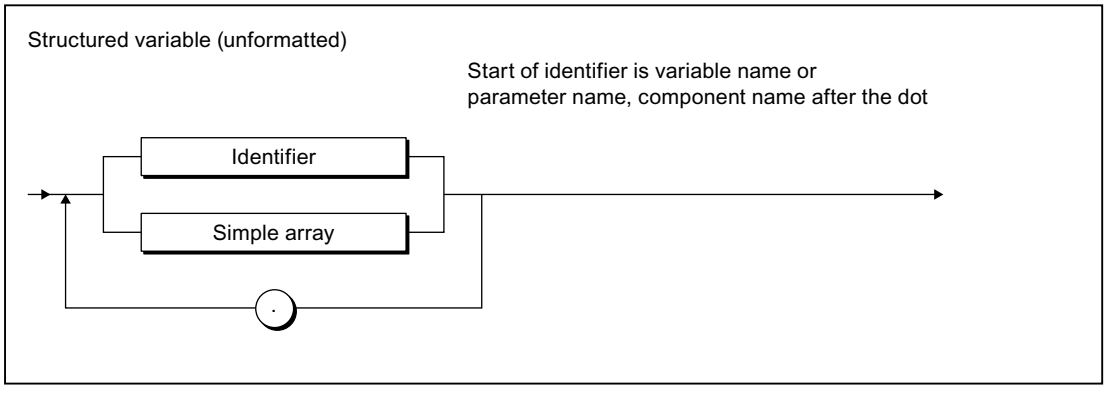


Figure 7-404 Structured variable

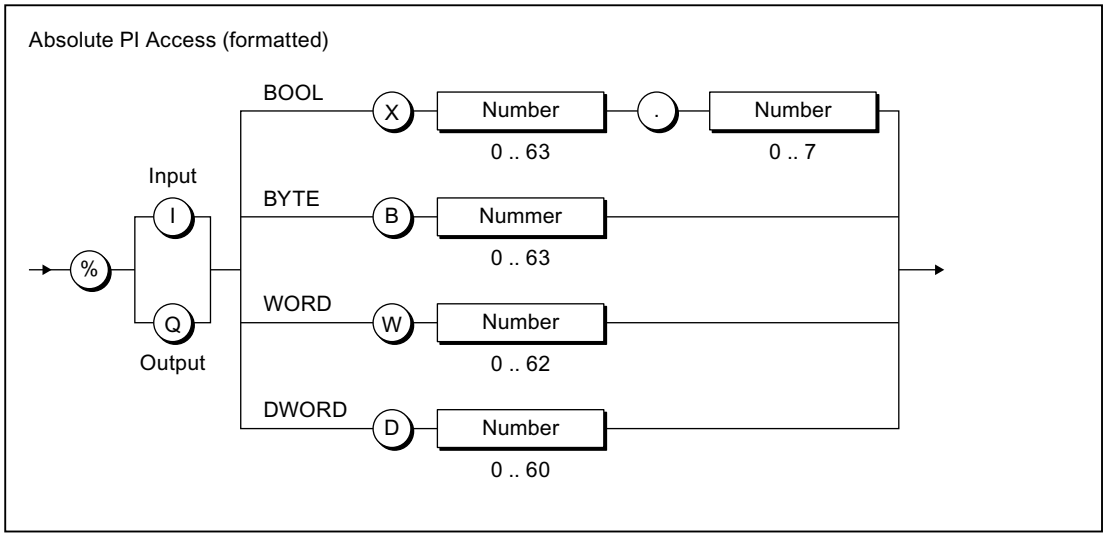


Figure 7-405 Absolute PI access

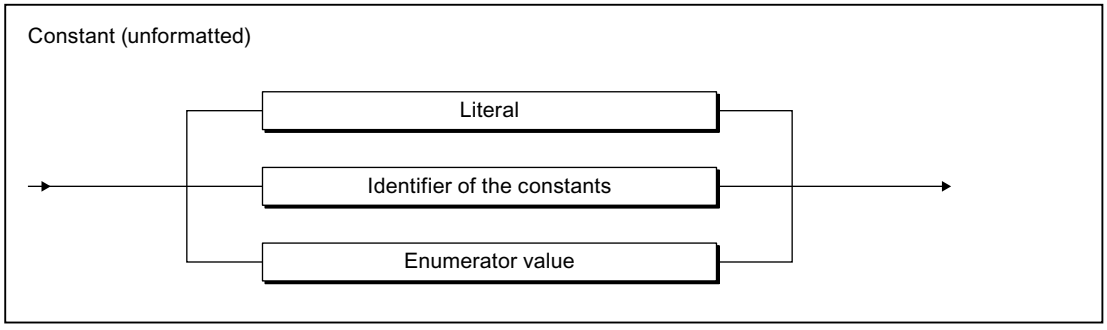


Figure 7-406 Constant

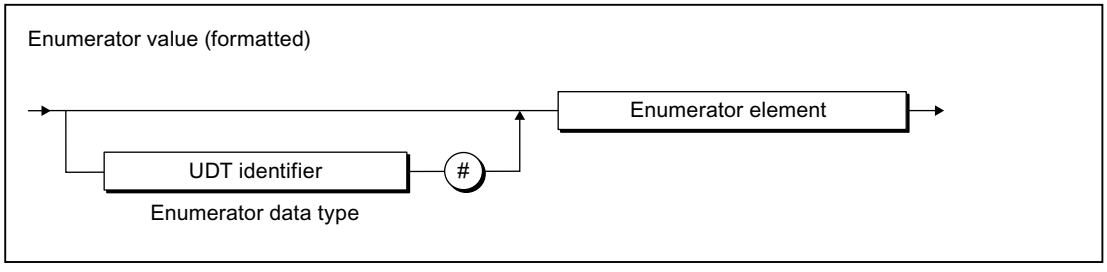


Figure 7-407 Enumerator value

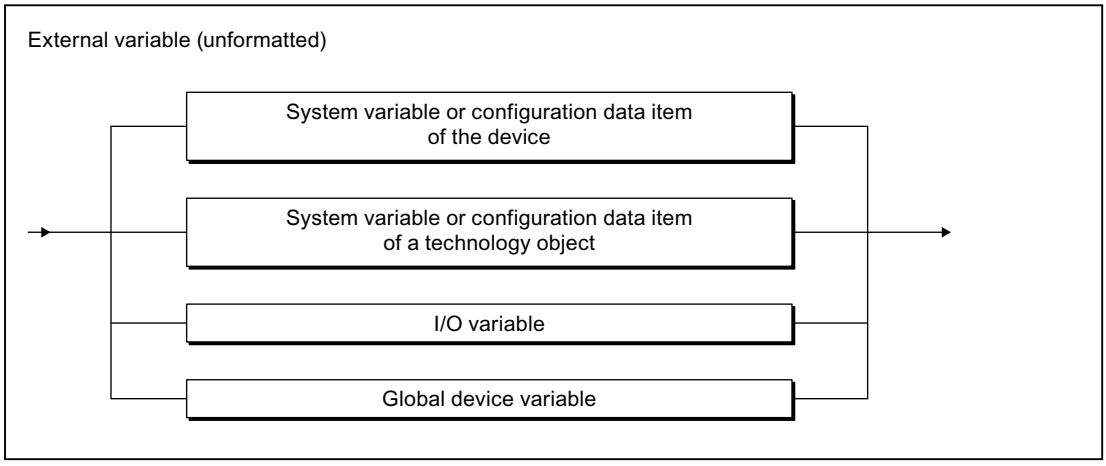


Figure 7-408 External tag

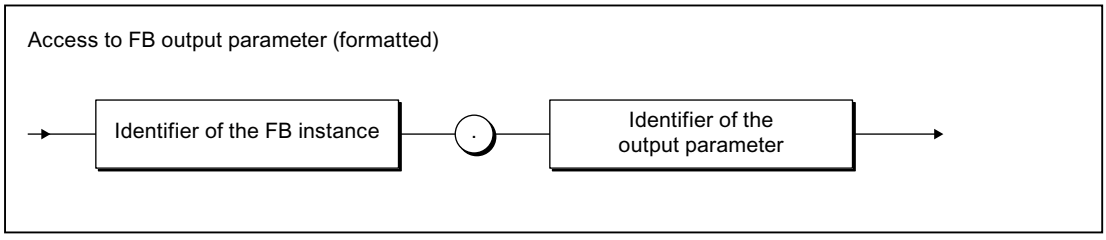


Figure 7-409 Access to FB output parameters

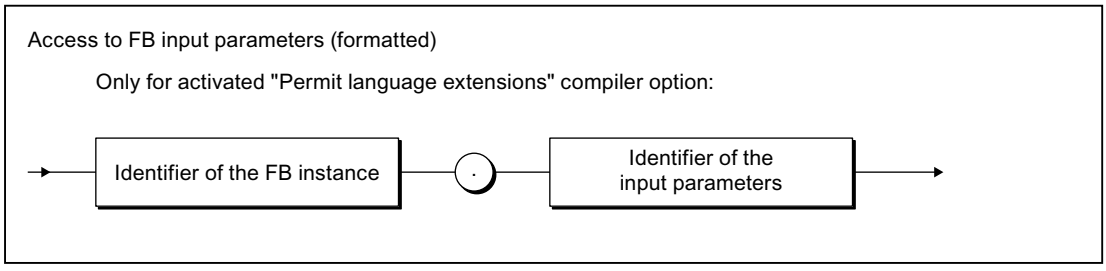


Figure 7-410 Access to FB input parameters

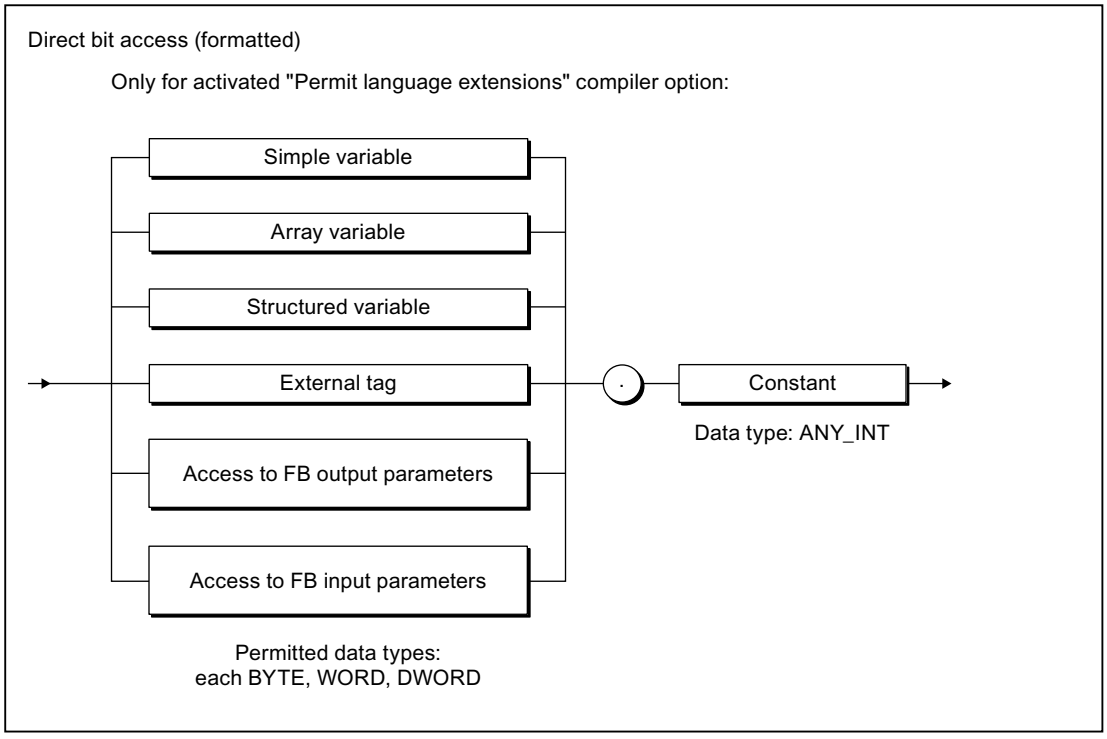


Figure 7-411 Bit access

Operators

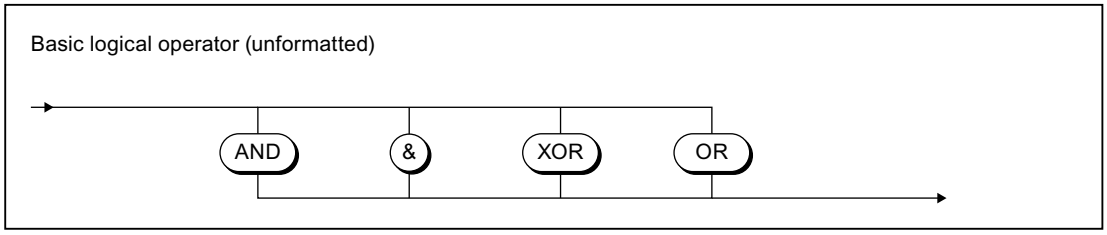


Figure 7-412 Basic logic operator

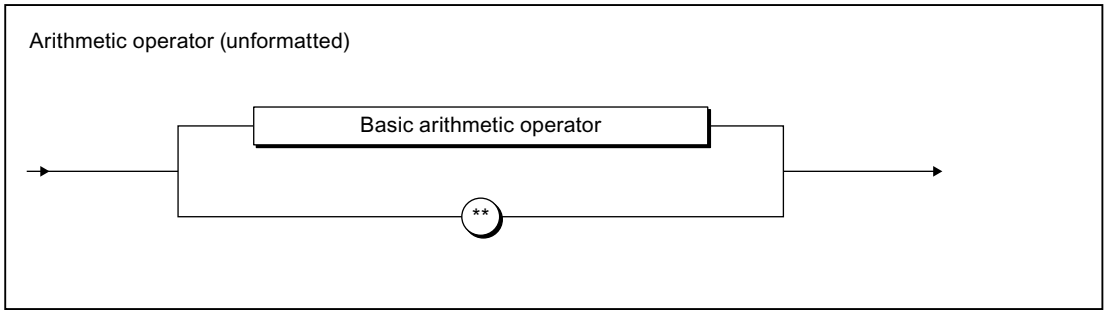


Figure 7-413 Arithmetic operator

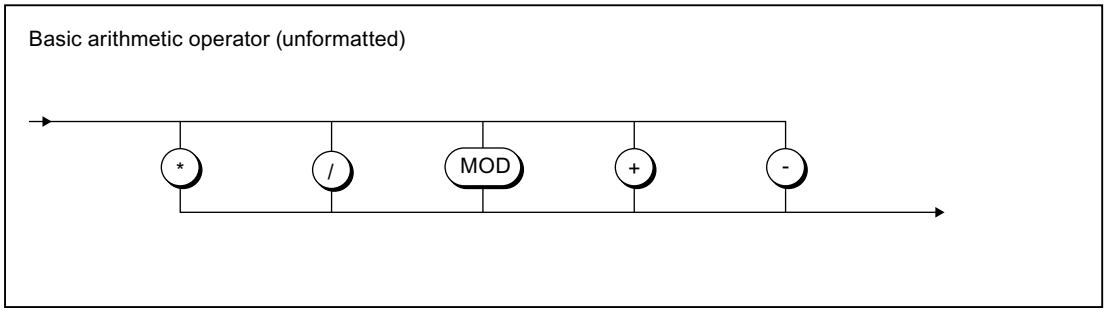


Figure 7-414 Basic arithmetic operator

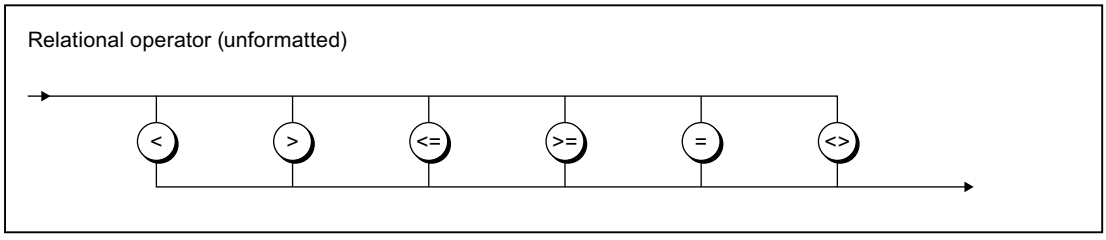


Figure 7-415 Relational operators

Call of functions, function blocks and methods

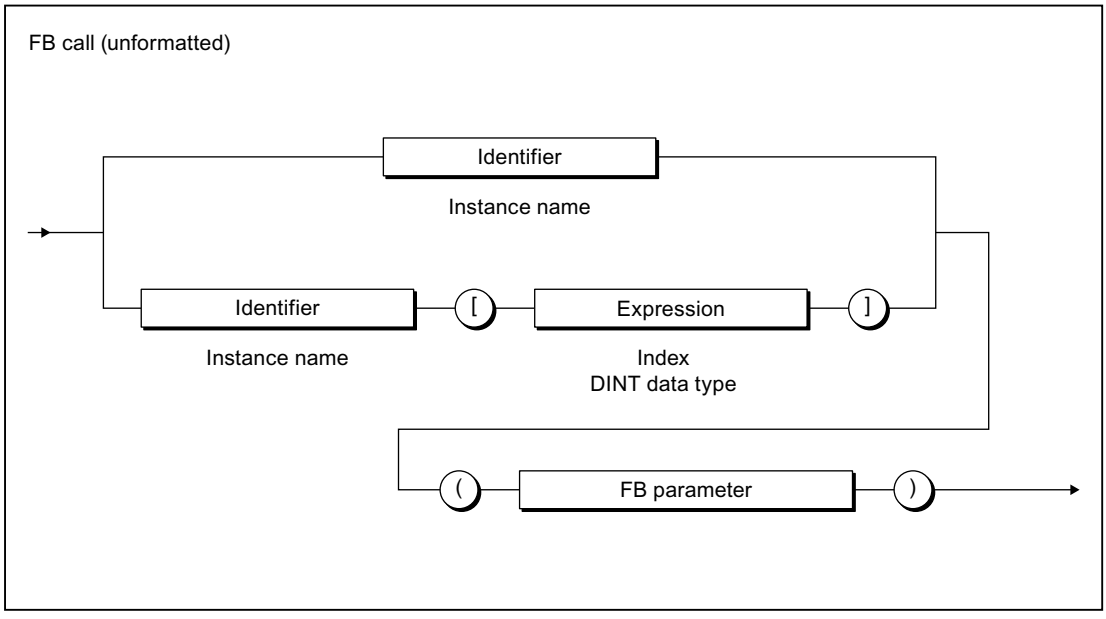


Figure 7-416 FB call

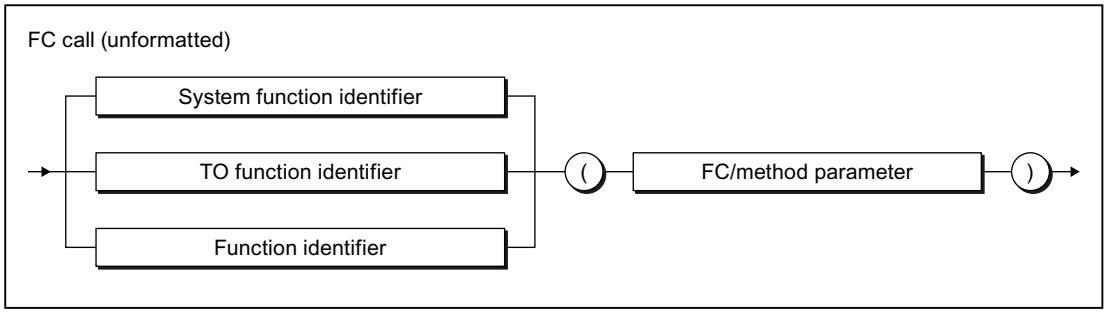


Figure 7-417 FC call

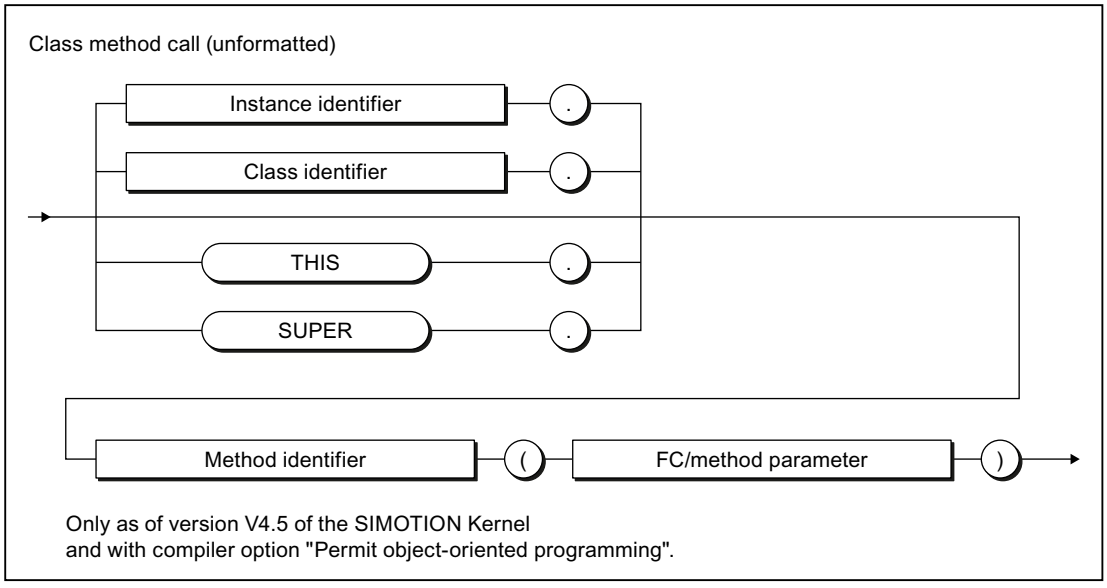


Figure 7-418 Class method call

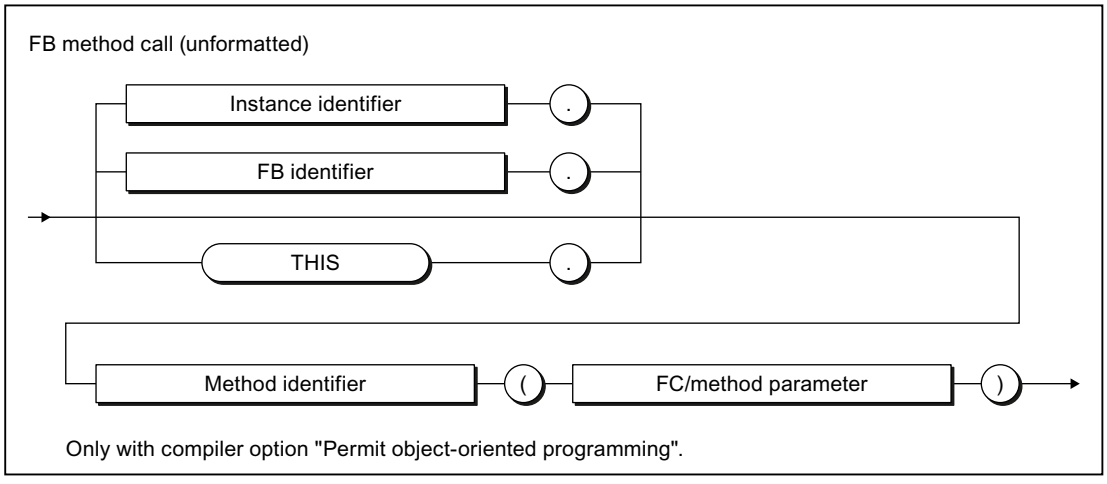


Figure 7-419 FB method call

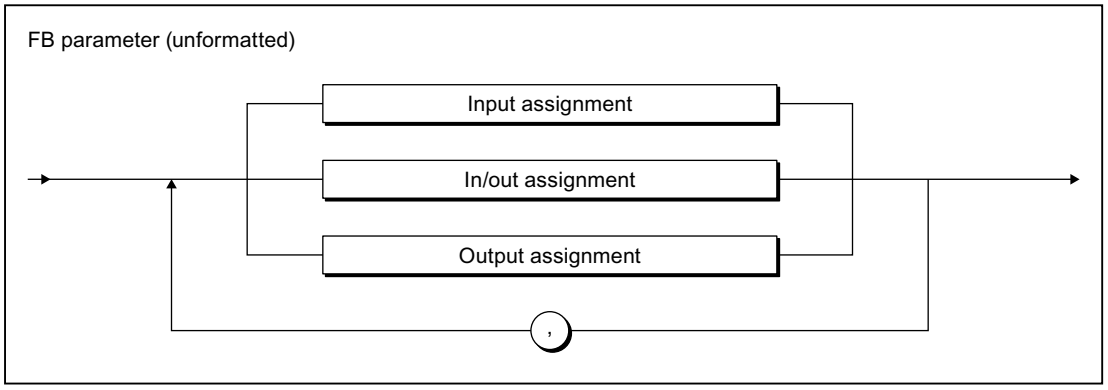


Figure 7-420 FB parameter

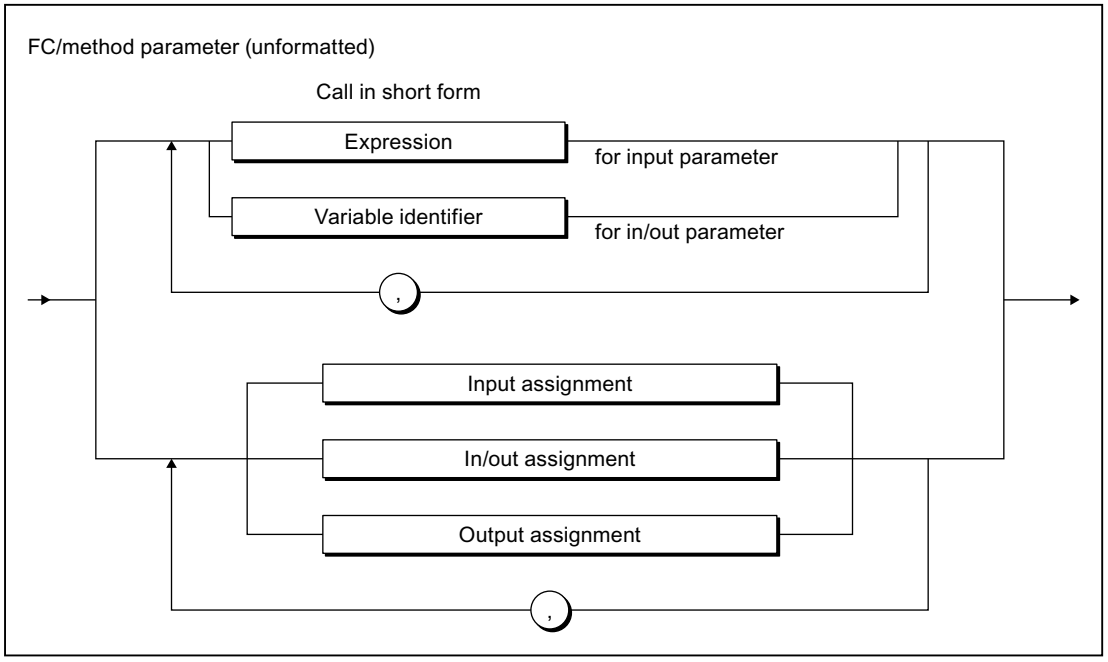


Figure 7-421 FC parameter

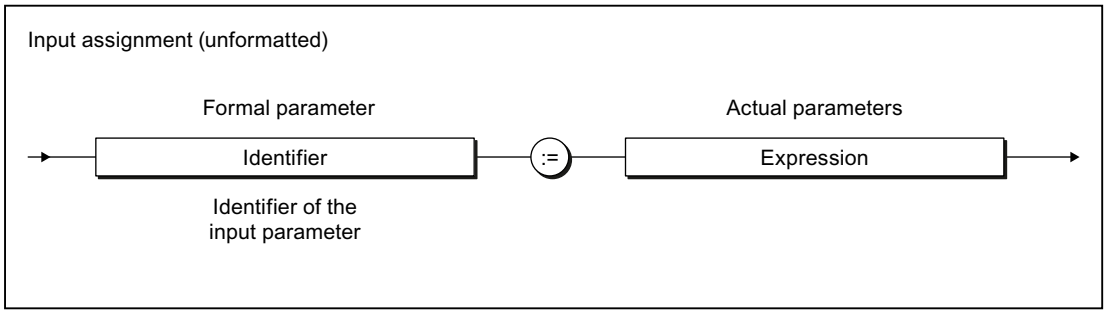


Figure 7-422 Input assignment

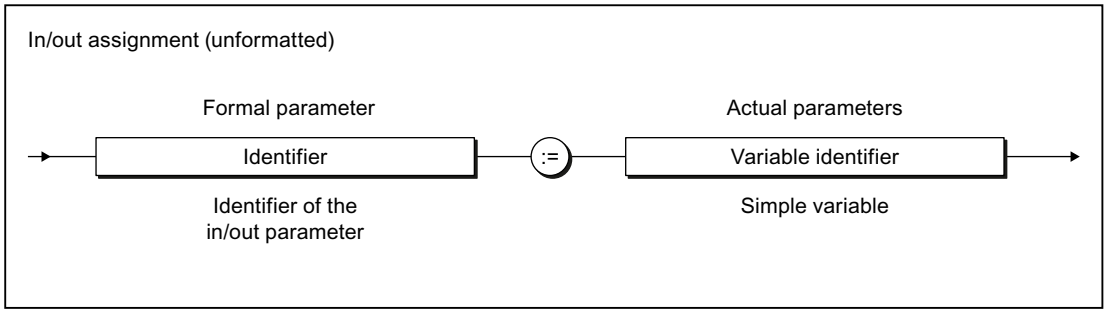


Figure 7-423 In/out assignment

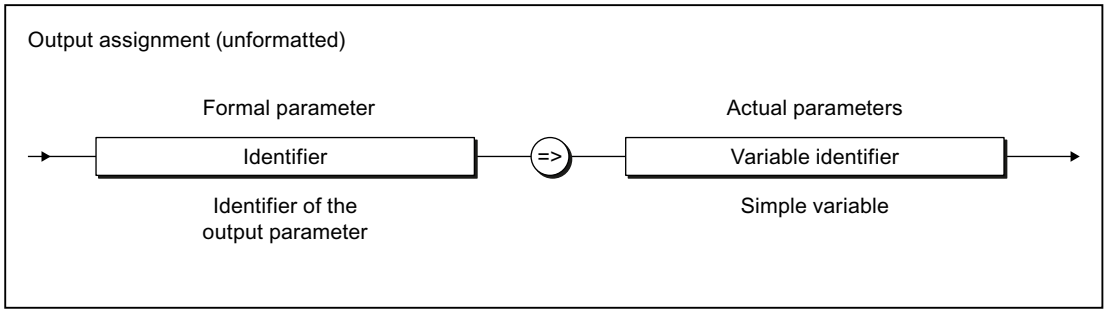


Figure 7-424 Output assignment

Control statements

Branches

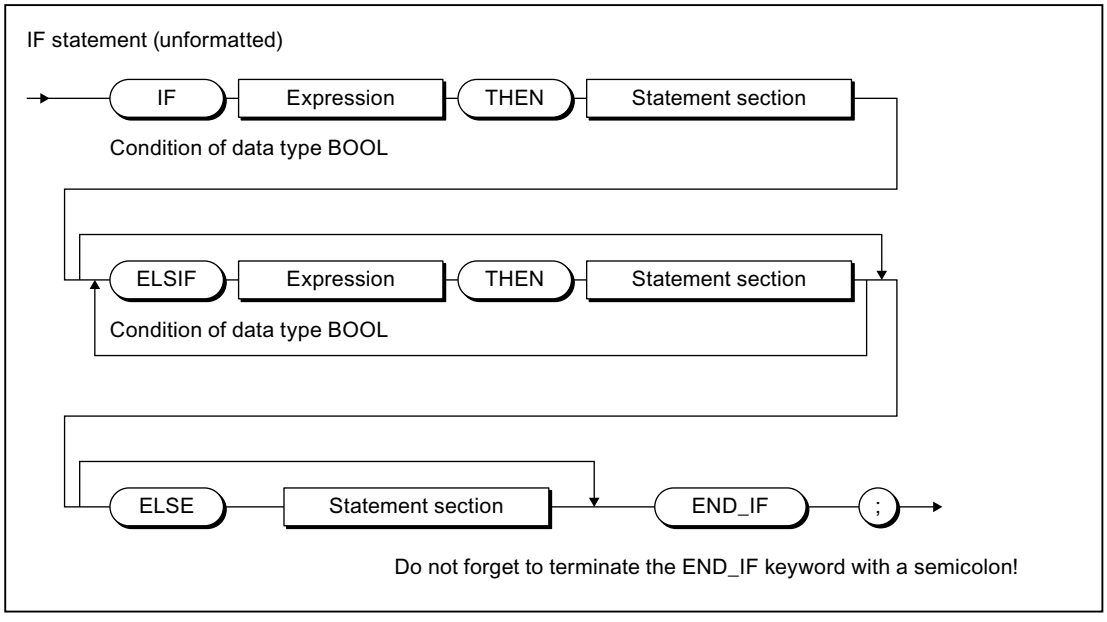


Figure 7-425 IF statement

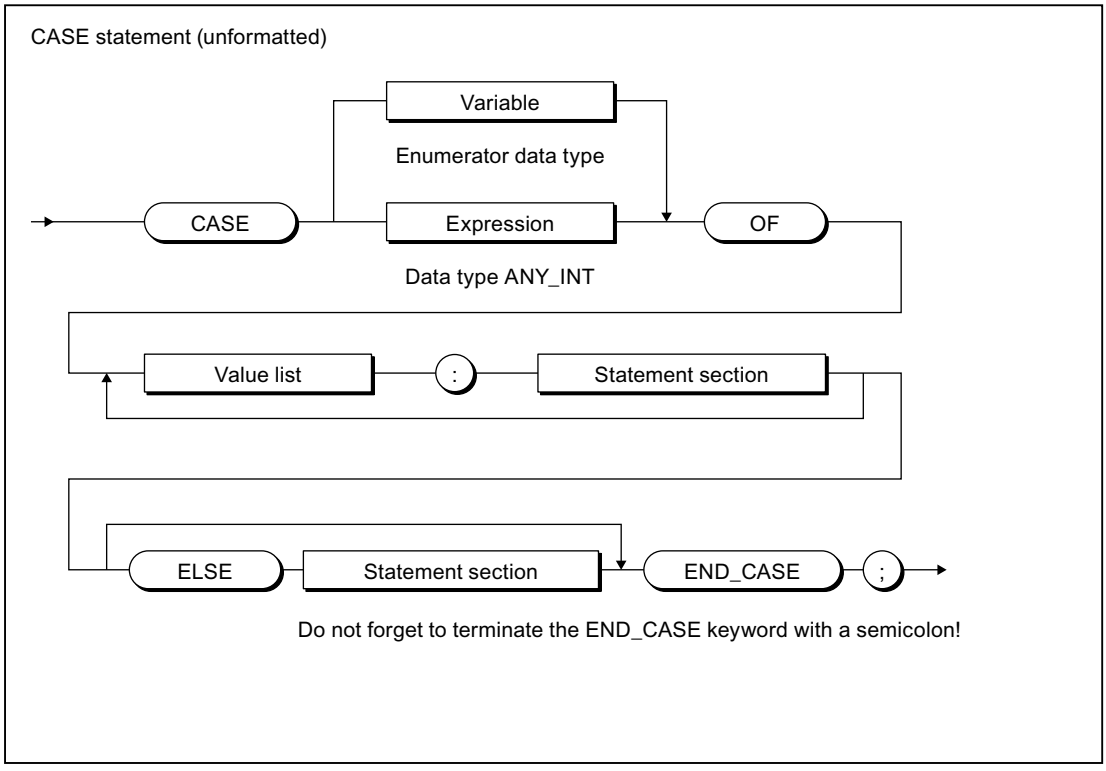


Figure 7-426 CASE statement

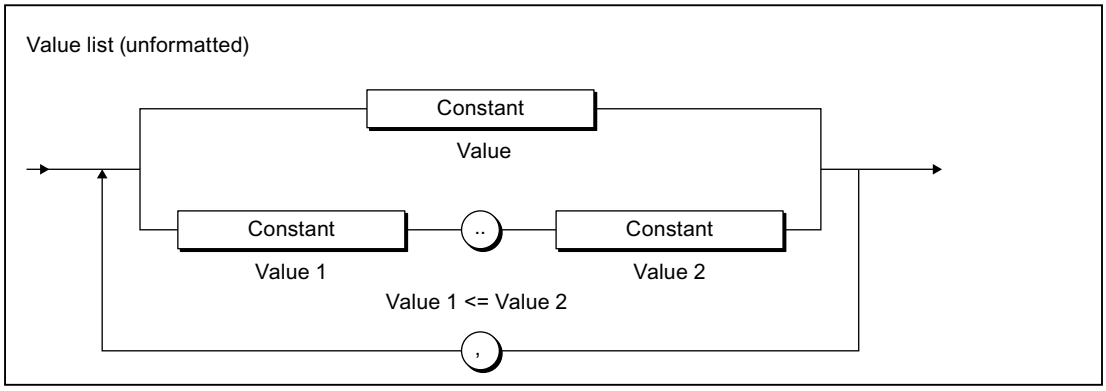


Figure 7-427 Value list

Repetition statements and jump statements

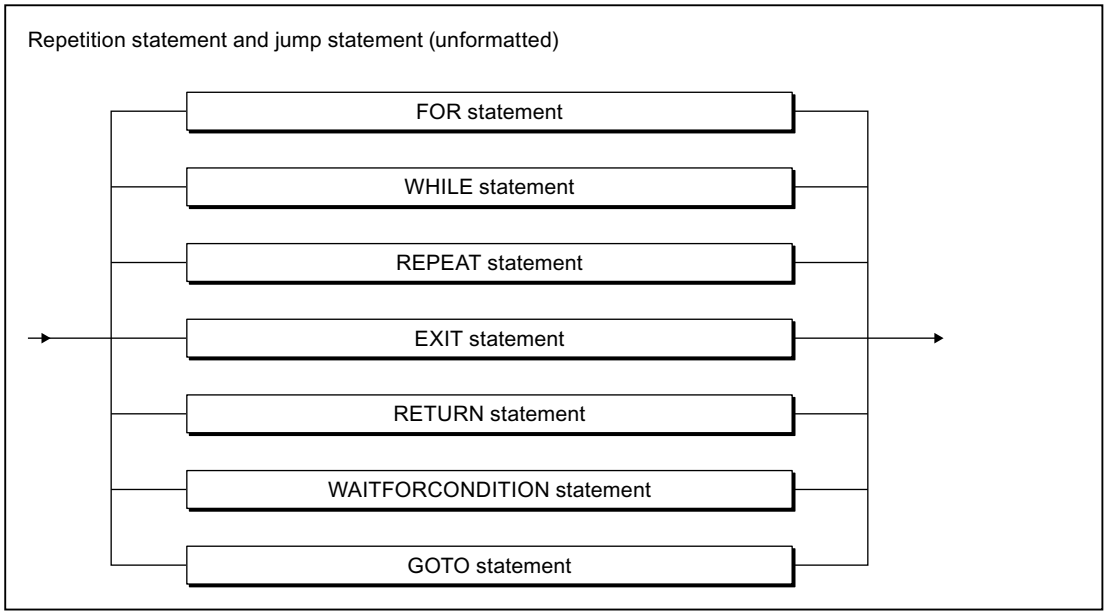


Figure 7-428 Repetition statement and jump statements

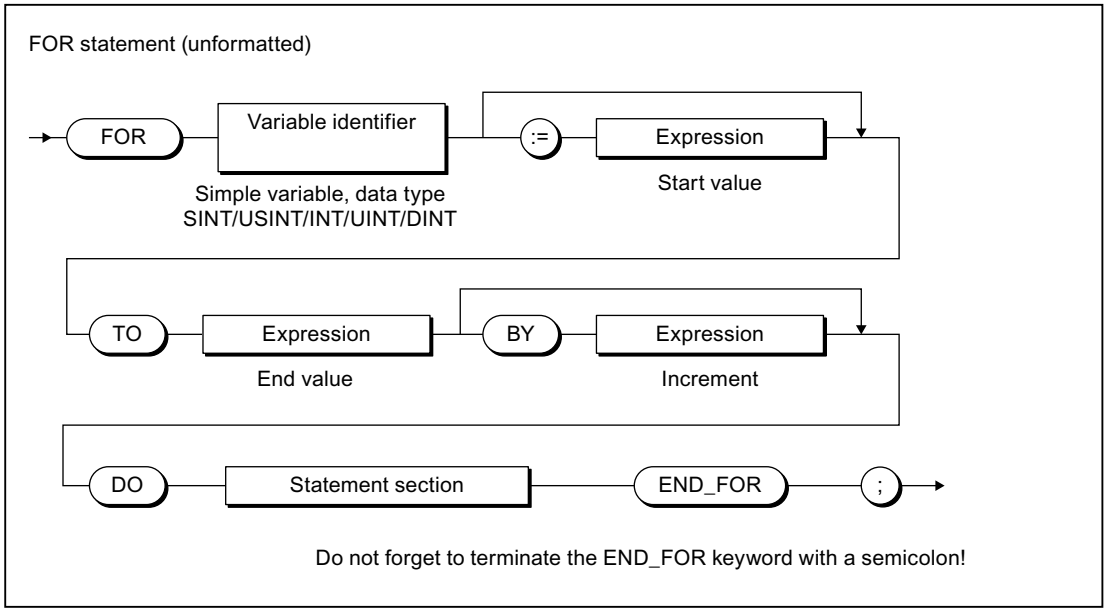


Figure 7-429 FOR statement

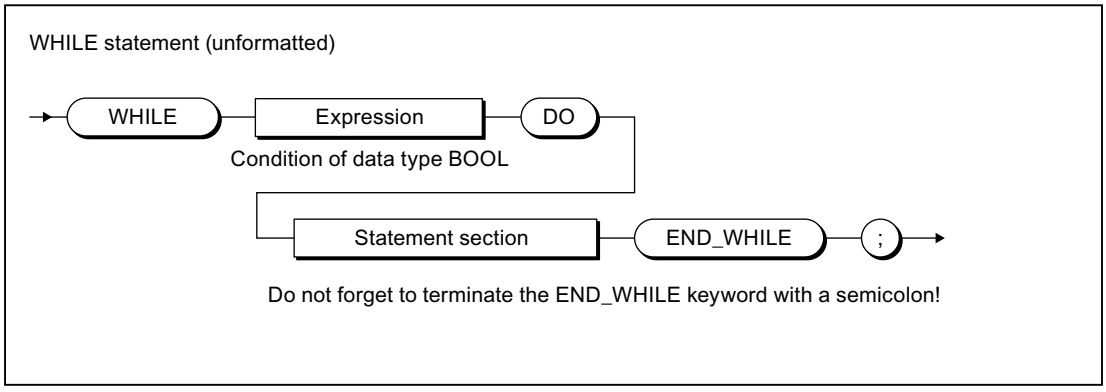


Figure 7-430 WHILE statement

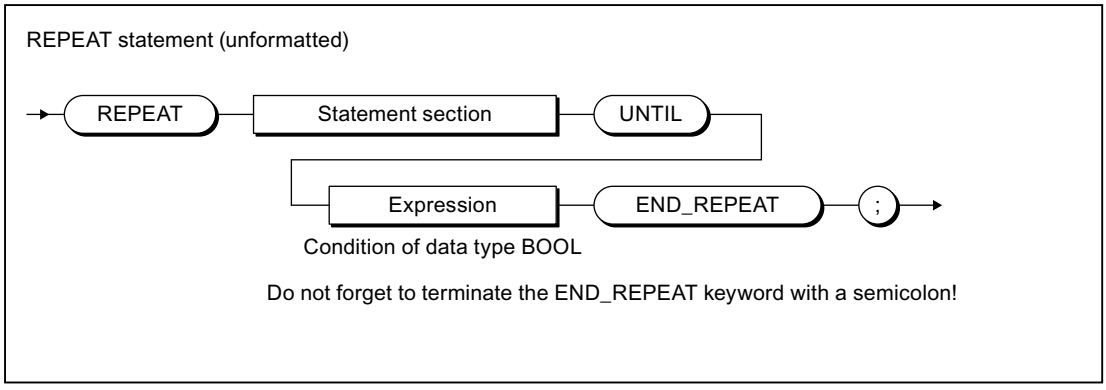


Figure 7-431 REPEAT statement

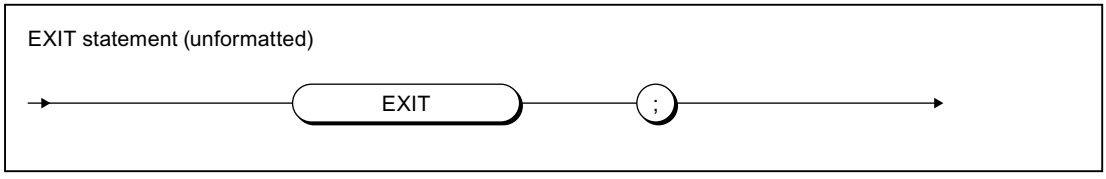


Figure 7-432 EXIT statement

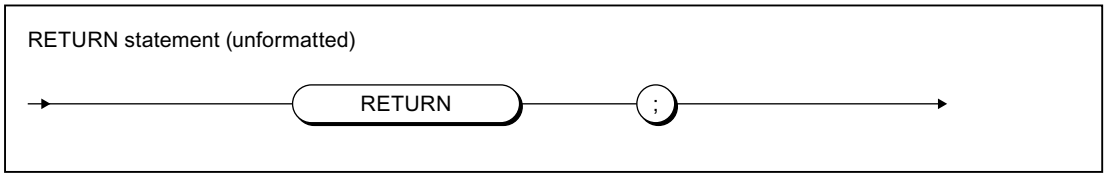


Figure 7-433 RETURN statement

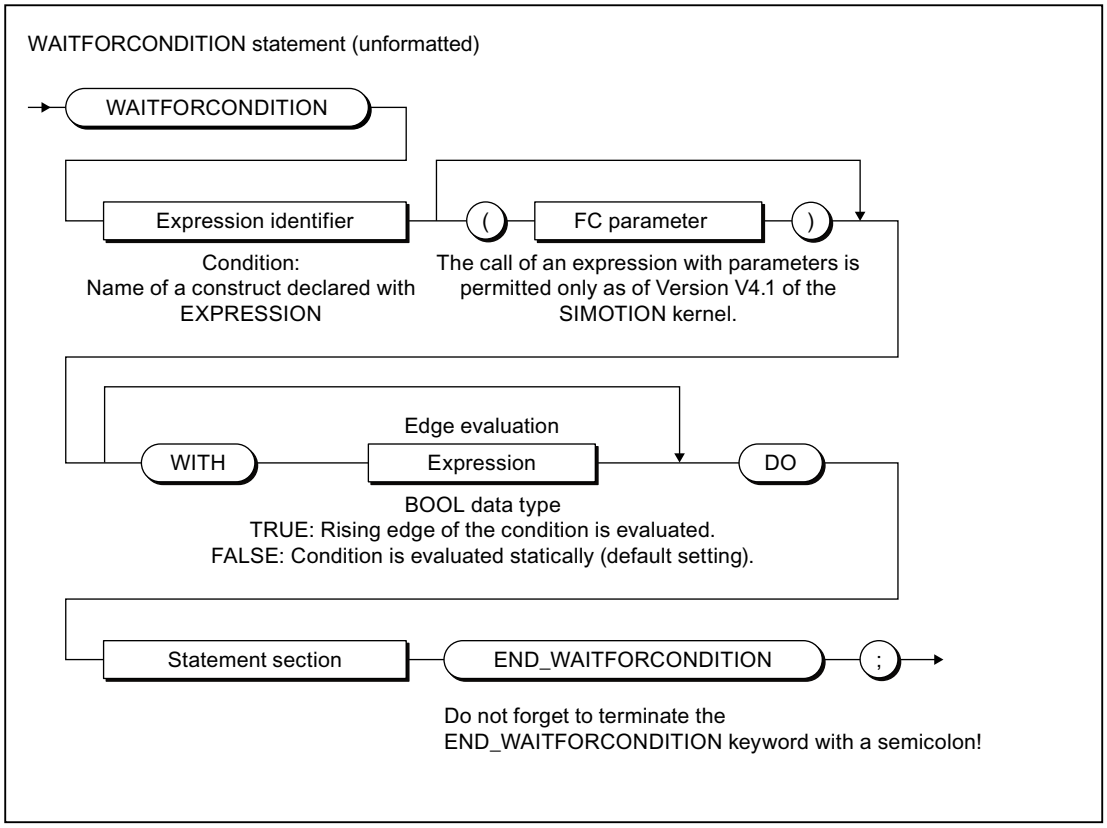


Figure 7-434 WAITFORCONDITION statement

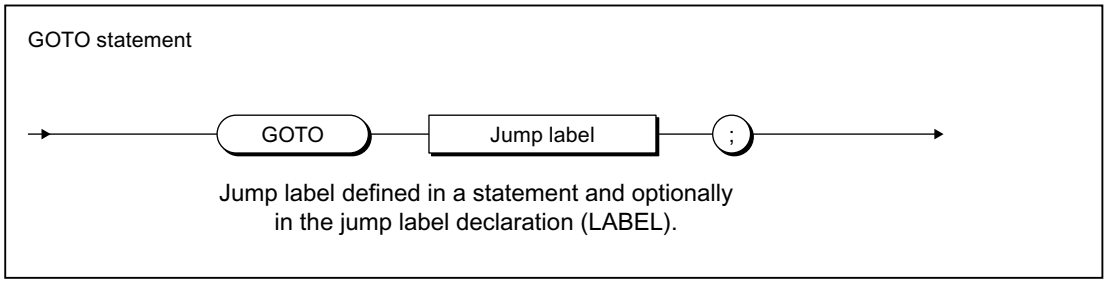


Figure 7-435 GOTO statement

Pragmas

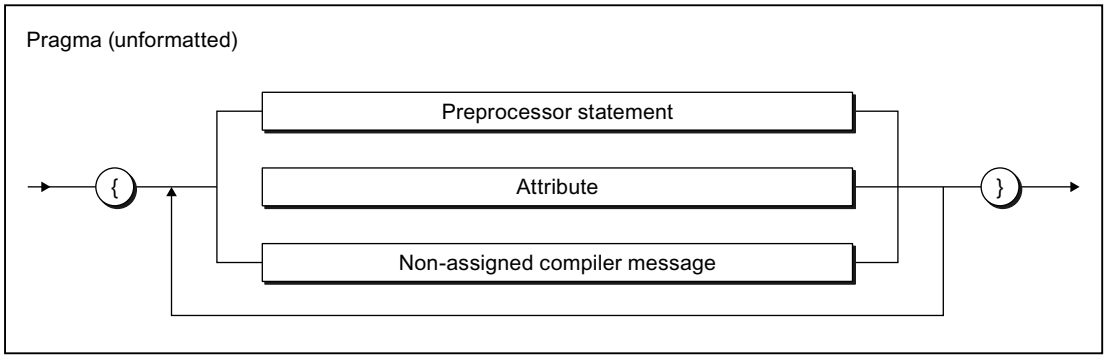


Figure 7-436 Syntax: Pragma

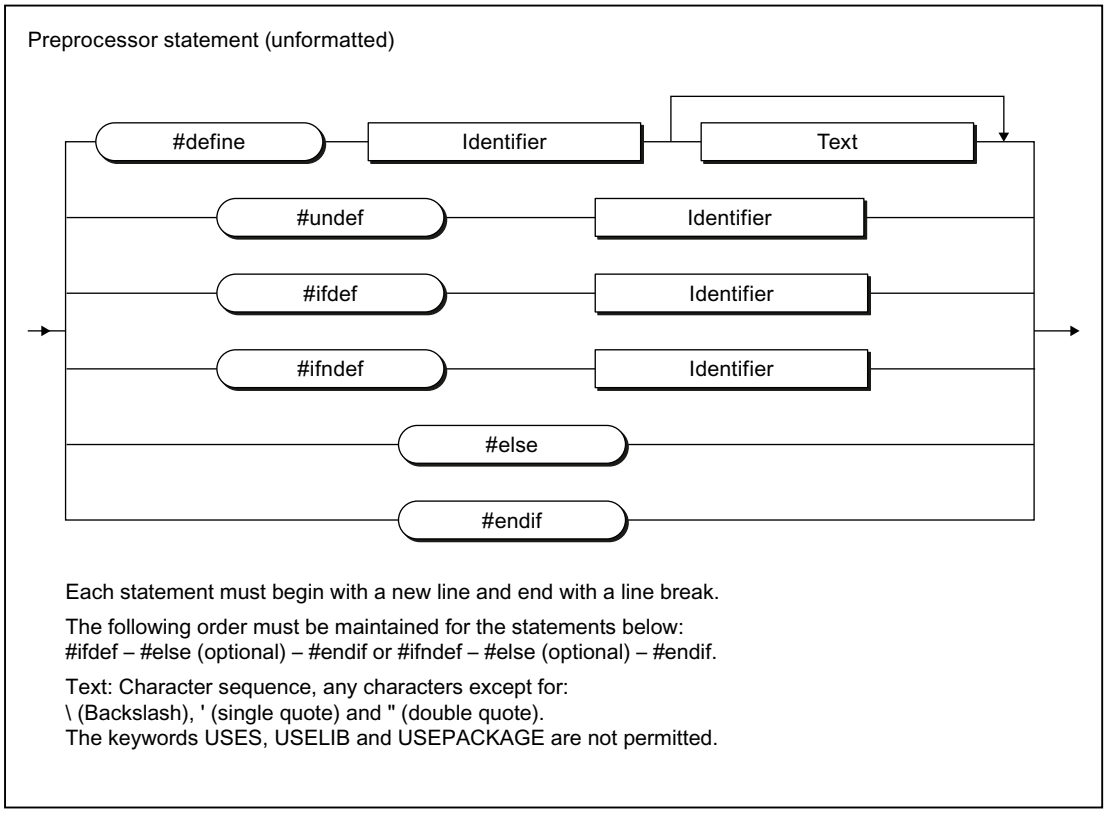


Figure 7-437 Syntax: Preprocessor statement

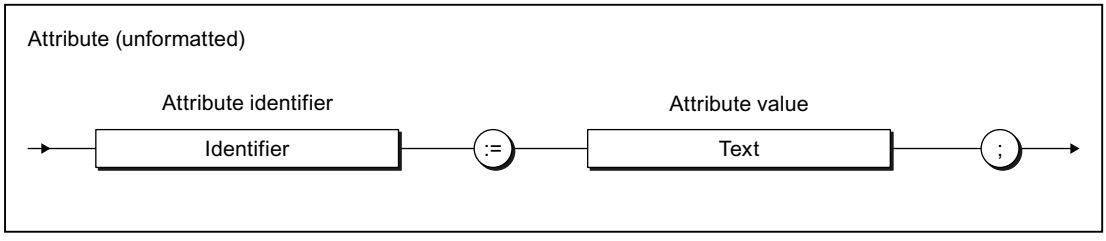


Figure 7-438 Syntax: Attributes for compiler outputs

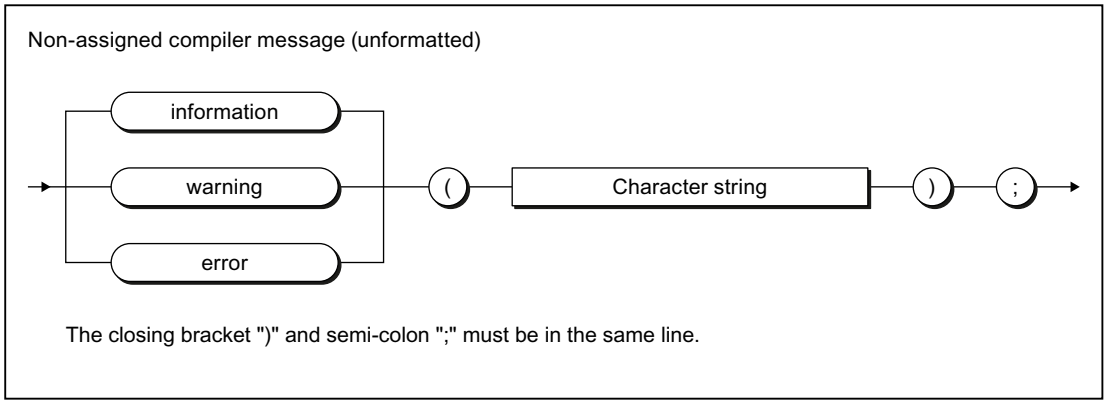


Figure 7-439 Syntax: Non-assigned compiler message

7.2.9.2 Compiler Error Messages and Remedies

This section provides an overview of the compiler error messages and their correction.

File access errors (1000 ... 1100)

Table 7-510 File access errors (1000 ... 1100)

Error	Description
1000	A read/write error has occurred on file access.
1001	Unable to load the file with the plain text error messages; cannot output error message texts. Please refer to the online help using the error number!
1002	The created code could not be stored. Please close some windows and recompile!
1003	A read/write error has occurred on opening the file. Please close the application and try again!
1100	The option for stating a preprocessor definition contains an invalid identifier as the defined token. The correct syntax of the call option is: -D identifier[=[text]] Examples: • -D myident // Definition of myident; this can be queried using #ifdef. • -D myident= // myident is defined as empty character string • -D "myident=This is a text" // myident is defined as character string 'This is a text'. The quotation marks only have to be used if the replacement text contains a blank.

Scanner errors (2001, 2002)

Table 7-511 Scanner errors (2001, 2002)

Error	Description
2001	The specified character is illegal.
2002	The specified identifier contains illegal characters or combinations of characters. According to IEC 61131, an identifier must start with a letter or an underscore. Any number of letters, digits, or underscores may follow, but no more than one underscore in a row.

Declaration errors in POU (3002 ... 3161)

Table 7-512 Declaration errors in POU (3002 ... 3100)

Error	Description
3002	Keyword "IMPLEMENTATION" to identify the code section of the load unit is expected.
3003	The specified declaration block is not permitted in this context.
3004	The VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR CONSTANT, VAR RETAIN variable declaration blocks are permitted only once for each POU.
3005	TASK statement: The task link has already been made in the source file for the specified task. Further task linking not possible.
3006	Incorrect stack size for task specified. Only positive integers are permitted.
3007	The specified identifier must be a task identifier; see task configuration.
3008	The specified identifier must be a program identifier. The declaration is made in the statement PROGRAM xx ... END_PROGRAM.
3009	The EXPRESSION keyword must be followed by an identifier. The declaration is made in the statement EXPRESSION xx ... END_EXPRESSION.
3010	The specified identifier is not an EXPRESSION identifier. Check whether the declaration was made using the statement EXPRESSION xx ... END_EXPRESSION.
3011	The TASK statement is not permitted in the unit. Use the task configuration in the Workbench.
3012	The specified identifier has already been declared at another position. It cannot be used again as a function identifier.
3013	The specified identifier has already been declared at another position. It cannot be used again as a function block identifier.
3014	The UNIT statement is expected. The following forms are permissible: • UNIT myunit; • UNIT myunit : dvtype; The UNIT statement is only required when compiling at the ASCII file level. It is optional when the compiler is called from the Workbench.
3015	The source file is not ended with END_IMPLEMENTATION. Observe the structure for a source file!
3016	No further statements may be specified after keyword END_IMPLEMENTATION.
3017	The task declaration is not ended with END_TASK. Observe the structure for a source file!
3018	The POU declaration is not ended with END_FUNCTION, END_FUNCTION_BLOCK, or END_PROGRAM. Observe the structure for a source file!
3019	A POU starting with keywords FUNCTION, FUNCTION_BLOCK, or PROGRAM is expected.
3020	The task linking statement is expected. Structure: TASK tname ... END_TASK;
3021	Methods can only be declared within classes, function blocks or interfaces. In this case, no further statements may be inserted between the variable or data type declaration and the methods.
3022	The keyword INTERFACE is expected. See the structure for a source file.
3023	Keyword INTERFACE or IMPLEMENTATION is expected. See the structure for a source file.
3024	Syntax error in TASK statement. Correct structure: TASK tname ... END_TASK;
3025	The specified identifier has already been declared at another position. It cannot be used again as a program identifier.
3026	The WAITFORCONDITION statement cannot be used recursively. An attempt was made to use a WAITFORCONDITION statement a second time within a WAITFORCONDITION statement. This is not possible.

Error	Description
3027	An attempt was made to insert a WAITFORCONDITION statement within an EXPRESSION ... END_EXPRESSION block. This is not possible. The WAITFORCONDITION statement cannot be used within an expression.
3028	The specified identifier has already been declared at another position. It cannot be used again as a class identifier.
3029	The specified identifier has already been declared at another position. It cannot be used again as an interface identifier.
3040	The specified identifier does not reference a namespace. It cannot be used in the USING statement.
3041	The specified identifier references a namespace that cannot be accessed. The namespace is declared INTERNAL.
3042	The specified identifier references a namespace whose symbols could not be added to the current context.
3043	The specified identifier references a namespace used system-internal. The identifier cannot be used as user namespace.
3050	A class definition starting with the keyword CLASS is expected.
3051	The class definition does not end with END_CLASS. Observe the structure for a source file.
3052	A method definition starting with the keyword METHOD is expected.
3053	A method definition ends with END_METHOD. The method definition contains a syntax error. Observe the structure for a source file.
3060	It is not permissible to use the specified access identifier at this position. For example, the access identifier PROTECTED cannot be used within a FUNCTION_BLOCK.
3061	The specified method property cannot be accepted. For example, OVERRIDE, ABSTRACT or FINAL cannot be declared for a method within a FUNCTION_BLOCK. OVERRIDE can be used for methods in CLASS only if a base class exists in which the method has already been declared. In this case, OVERRIDE cannot be declared for PRIVATE methods.
3062	If a method is overridden in a derived class, the access identifiers must be identical. Accordingly, a PROTECTED method can only be overridden with PROTECTED and a PUBLIC method only with PUBLIC.
3063	If a method is to be overridden in a derived class, it is absolutely essential to specify the keyword OVERRIDE in the method declaration.
3064	A method that is already defined in the base class cannot be overridden with "ABSTRACT".
3065	The keyword "OVERRIDE" is used in the declaration of the specified method. Since the method is not defined in the base class, it is not permissible to specify this keyword.
3066	The method signatures are different in the specified classes. This is not compatible with override mechanisms. For example, not all the parameters have the same name or they are defined in a different sequence or they have different data types. Even the return value must have the same data type.
3067	If a method is declared with the PUBLIC identifier in a base class and a method of the same name is required by an object-oriented interface to be implemented, this method must already be declared PUBLIC in the base class. Methods in object-oriented interfaces are always PUBLIC. It is not possible to change the access level for derivations.
3068	A method that is already defined in the base class and identified as "FINAL" cannot be overridden.
3069	A method cannot be declared ABSTRACT if the class is identified as FINAL. This combination cannot be used because it is not possible to create any instances of this class.
3070	The specified data type cannot be used as a base class or a base interface. A CLASS can only ever be derived from another CLASS. An INTERFACE can only ever be derived from another INTERFACE. It is not possible to use mixtures of the two or to create derivations of other data types.
3071	The specified CLASS is identified with the keyword FINAL. A CLASS identified as FINAL cannot be used as a base class.

Error	Description
3072	The specified CLASS is not exported in the interface section of the unit or stems from a unit imported in the implementation section and cannot therefore be used as a base class of an exported CLASS. • If the CLASS mentioned in the error message is declared in the current unit, it must be exported in the interface section. CLASS <classname>; must be added there for this purpose. • If it is a CLASS from another unit, the USES statement for this unit must be moved from the implementation section to the interface section.
3073	The specified CLASS is identified with the keyword FINAL. However, the relevant class does not implement all of the methods (in base classes) identified as ABSTRACT and therefore cannot be instantiated. All ABSTRACT methods must either be implemented in the class, or the class cannot be identified as FINAL.
3075	The specified identifier does not refer to an INTERFACE. Only INTERFACE identifiers can be used for IMPLEMENTS.
3076	The specified INTERFACE is already implemented by the CLASS itself or by the base class(es). An INTERFACE can only ever be implemented once by a CLASS.
3077	The specified INTERFACE is not exported in the interface section of the unit or stems from a unit imported in the implementation section and cannot therefore be used at an exported CLASS (IMPLEMENTS). • If the INTERFACE mentioned in the error message is declared in the current unit, this declaration must be included in the interface section. For this purpose, the entire declaration must be moved to the interface section. • If it is an INTERFACE from another unit, the USES statement for this unit must be moved from the implementation section to the interface section.
3078	The specified INTERFACE is identified as a prototype. This is a programming error. Object-oriented interfaces must always be defined in full.
3080	A method that is identified as PRIVATE cannot be declared as either ABSTRACT or FINAL. It must always be implemented and no override mechanisms of any kind may be used.
3081	A method identifier is concealing a global identifier. This is only permissible if you activate the compiler option 'Permit forward declarations'. This ensures that calls of this method within the current CLASS or the current FUNCTION_BLOCK without a preceding name refer to this method in all other methods. The global identifier can no longer be accessed in the class because it is concealed by the method identifier.
3100	The specified identifier is already defined and identified as public in the base class. It cannot be declared again in the derived class.
3150	The specified identifier has already been identified as public without INTERNAL being specified. It is not possible to declare it INTERNAL again differently. • If this situation occurs for a POU, this POU has already been identified as public without INTERNAL in the INTERFACE section of the source. Add the INTERNAL identifier here to use the POU only within the namespace. • If this situation occurs for a namespace, an identically named namespace declaration from a source or library imported via USES/USELIB may also be present. In this case, all declarations must be either INTERNAL or not INTERNAL.
3151	The specified identifier has already been declared as INTERNAL in the current source or in an imported source or library. The current declaration location, however, does not contain the INTERNAL identifier. Add INTERNAL for the declaration of the affected element.

Error	Description
3160	The specification of USING within a namespace is possible only for the first declaration within the source and then applies to all blocks of the affected namespace. Move the USING statement to this position.
3161	The specification of USING within a namespace or a POU is permissible only for namespaces not contained in the declaration path. For example, a namespace NS1 or class, function, etc., declared in NS1, may not itself contain any USING statement for NS1.

Declaration errors in data type declarations (4001 ... 4105)

Table 7-513 Declaration errors in data type declarations (4001 ... 4105)

Error	Description
4001	The specified identifier is a standard function identifier that cannot be overwritten. Choose a different identifier.
4002	The specified identifier has already been used. Use as a type identifier is not possible. Choose a different identifier.
4003	The specified identifier has already been used. Use as a constant identifier is not possible. Choose a different identifier.
4004	The specified initialization value has an incorrect format. Choose the initialization value that corresponds to the data type declaration.
4005	Syntax error in type declaration.
4006	Syntax error in the structure element specification in the structure declaration.
4007	Syntax error in declaration of an ARRAY data type.
4008	Syntax error in the identifier list specification. The identifiers must be separated by commas.
4009	The specified constant identifier has been assigned different values. This occurs when enumeration data types are declared. Identical enumeration elements in different enumeration data types must be located in the same position in the type declaration.
4010	The specified type identifier is not exported from the source file, although the POU in which it is used, is exported. Use a different data type or declare the data type in the implementation section.
4011	A constant declaration requires the specification of an initialization value. Example: x : DINT := 5;
4012	The specified data type cannot be used at the current position. • When compiler option "Permit object-oriented programming" is not activated, type identifiers declared locally in the POU may not be used with VAR_INPUT, VAR_OUTPUT, and VAR_IN_OUT because they must also be known outside the POU for parameter transfer purposes. • When compiler option "Permit object-oriented programming" is activated, the local data types used for variables, parameters and other type declarations must have at least the same access definition as that applicable at the point of use. For example, a local data type used at the transfer parameters of a PROTECTED method must itself be declared as at least PROTECTED. PRIVATE data types may only be used at local variables.
4013	The specified value is used several times in the enumeration data type. The values in the enumeration data type must differ, however.
4020	The specified identifier has already been used as a data type identifier. However, the definition differs from the current definition. This is not permitted. Either choose a different identifier or adapt the type definitions. If this message appears on loading libraries or technology packages, you can use namespaces here too (e.g. USELIB mylib AS Namespace_1).

Error	Description
4050	The data type or variable declaration creates a data type that is larger than the specified maximum permissible data size.
4051	The variable declaration requires a memory area that is larger than the specified maximum permissible memory size.
4100	The definition of the structure component requires the specification of a user-defined offsets. If offsets are explicitly specified in a structure definition, the offset specification is necessary for all structure components. Furthermore, the keyword OVERLAP requires the explicit specification of the offsets.
4101	The offset specified in the definition of the structure component is not allowed because a component without an explicitly specified offset has already been defined. If offsets are explicitly specified in a structure definition, the offset specification is necessary for all structure components.
4102	The offset value specified in the definition of the structure component is not allowed. The value must be a multiple of the number specified in the error message so that the data elements are in the correct alignment.
4103	The offset values declared in the structure definition result in overlaps in the memory layout. This is only allowed for identification of the structure as OVERLAP.
4105	The overlaps in the memory space are not allowed for the following data types: STRING, ANYOBJECT and all TO-references derived thereof!

Declaration errors in variable declarations (5001 ... 5509)

Table 7-514 Declaration errors in variable declarations (5001 ... 5509)

Error	Description
5001	The specified constant value causes the value range to be exceeded and cannot be converted to the requested type.
5002	The specified identifier has already been used. Use as a variable identifier is not possible. Choose a different identifier.
5003	Syntax error in variable declaration.
5004	The specification of a data type is expected (simple or derived data type).
5005	The specified constant value has the wrong data type or causes the value range to be exceeded.
5006	Check the number of initialization values for array initialization.
5007	Syntax error in the specification of the time and date literals.
5008	An instance of a function block or a class cannot be created at the specified position. For example, instances of function blocks or classes cannot be created in functions or methods. The output parameters (VAR_OUTPUT) of function blocks cannot be FB instances. Furthermore, classes and function blocks that have been compiled under the compiler option "Permit object-oriented extensions" cannot be used as input parameters (VAR_INPUT). It is not possible to copy classes and function blocks of the kind that would be created if they were to be used in this way.
5009	The data type specified in the declaration cannot be applied to the variable with absolute address. An integer or bit data type with matching bit width must be used.
5010	An attempt was made to assign a memory address to a variable. This is not possible at the specified position. Use this assignment only within the VAR_GLOBAL declaration of a unit or within the VAR declaration of a PROGRAM.
5011	The data type stated in the declaration cannot be accessed. It is not identified as public. The data type must be declared as PUBLIC to allow external access and PROTECTED to allow access from a derived class.
5012	The specified variables cannot be preassigned an initialization value.

Error	Description
5014	Incorrect initialization of a data structure. The initialization value for a component was specified more than once.
5016	The initialization of variables and data types with technology objects defined in the project is not possible. Technology objects are themselves variables and so cannot be used for the initialization.
5030	If it is necessary for variable declarations and POU implementations to be compiled in an optional sequence in the implementation section of the source file, the compiler option "Permit forward declarations" must be activated.
5040	The specified data type cannot be used within a declaration of retentive variables (VAR RETAIN, VAR_GLOBAL RETAIN) because it already contains retentive elements (VAR RETAIN).
5041	The specified data type cannot be used within a declaration of retentive variables (VAR RETAIN, VAR_GLOBAL RETAIN). The data type does not contain any information that can be backed up or restored in the retentive memory. This message is issued, for example, if a class or a function block does not contain instance data of any kind. This message is also output for variables of type INTERFACE or with the data type of a technology object.
5042	The specified data type cannot be used within a declaration of retentive variables (VAR RETAIN, VAR_GLOBAL RETAIN). The data type contains at least one system function block in the instance data or is one itself. Since its information cannot be restored, this data type cannot be used in retentive variable declarations.
5050	The class definition is identified as ABSTRACT. As a result, it is not possible to create any instances of the relevant class.
5051	The specified method is identified as ABSTRACT and has not yet been implemented in a method override mechanism. As a result, it is not possible to create any instances of the relevant class.
5052	The class in which a method is defined cannot be used as a variable or parameter at methods within a class declaration. The same applies to function blocks and interfaces. In this case as well, the data type of the function block or interface cannot be used within the declaration section as a data type for variables or transfer parameters.
5053	A variable with the specified data type cannot be directly declared as an in/out parameter (VAR_IN_OUT). Since dynamic type conversion is not supported for in/out parameter calls, it is not possible, for example, to return an INTERFACE value or a reference to VAR_IN_OUT. Declare the variable as either VAR_INPUT or VAR_OUTPUT.
5054	The specified class does not possess any instance data. As a result, it is not possible to create any instances of the relevant class in the form of an ARRAY.
5055	Initialization is only possible with the stated value at the specified position. For example, INTERFACE variables that are defined within structures, methods or functions, or as VAR_TEMP or VAR_IN_OUT, cannot be initialized with class instances. In this case, only NULL can be specified as a value. References to technology objects that are defined within structures, methods or functions, or as VAR_TEMP or VAR_IN_OUT, may only be initialized with the value TO#NIL.
5056	An initialization value may not contain a variable ARRAY index. An initialization value is not a constant.
5057	The instance variables of classes or function blocks of type INTERFACE may only be initialized with exported global class instances. They can also be initialized with local class instances defined with the class or the base classes. They cannot be initialized with class instances from the IMPLEMENTATION area.
5058	Variables of type INTERFACE can only be initialized with the value NULL or with specified class instances. They cannot be initialized with the value of other variables of type INTERFACE.
5059	Method arguments of type INTERFACE cannot be initialized with class instances that are declared as retentive local variables (VAR RETAIN) within the CLASS or the FUNCTION_BLOCK.
5070	The specified variable must be initialized. Please enter a valid initialization value. To do this, use the valid syntax for instance initialization, i.e.: <var_name> : <var_type> := (<element_name1> := <value1>, <element_name2> := <value2>);

Error	Description
5071	The specified variable must be initialized. The variable must be declared before an initialization value can be specified: • either in a declaration section made accessible for initialization with OVERRIDE • or in a declaration section identified as PUBLIC.
5072	The specified data type has been declared by "forward declaration". It is not possible to specify instance-specific initialization values at the stated position. If it is a variable in the implementation section of the source file, check whether you can place it after the declaration of the relevant CLASS or FUNCTION_BLOCK. It is not generally possible to initialize variables created by "forward declaration" in the interface section of the source file.
5080	It is not permissible to specify the initialization value " * " for the specified data type. Instance-specific initialization can be forced using " * "' only for variables of data type INTERFACE or for the references of technology objects.
5081	The initialization value " * " may only be specified for static variables of the CLASS and FUNCTION_BLOCK. Furthermore, the declaration block that contains the variable(s) must permit initialization by OVERRIDE or PUBLIC.
5090	An I/O reference can be declared only within VAR declaration blocks for classes and function blocks. An attempt was made to declare such a variable in a different declaration section. This is not permitted.
5091	The specified data type is not permitted. Only the following data types can be used for I/O references: • Integer data types, e.g. SINT, UINT • Bit data types, e.g. BOOL, WORD • Arrays of these elementary data types, other than arrays of the BOOL data type Other data types cannot be used, because they cannot be specified in the address list.
5092	The I/O reference cannot be linked with the specified I/O variable. The I/O reference requires that the affected variable must be read/write. • I/O references declared with %I* can be linked with all variables of an appropriate data type from the address list. • I/O references declared with %Q* require a variable from the address list for an output. This variable may not be identified with the attribute "read only".
5100	The specified variables cannot be preassigned an initialization value.
5110	The following specifications containing the special character $... are permitted: $$, $', $L, $N, $P, $R, $T. Moreover, the numeric value of a character can be stated using $xx, where xx stands for the two-digit hexadecimal specification of the character code.
5111	The special character can only be specified via $... . This applies to: $L, $N, $P, $R, $T
5112	Multi-line character string constants are not permitted. To produce a new line in the output, use the appropriate special character in the character string, e.g. $N, RL.

Error	Description
5120	No general reference can be generated to the specified data type. General references can be generated for the following data types: • Elementary data types • User-defined data types • Classes • Function blocks with at least one static variable. References to the following data types are not permitted: • General references • Object-oriented interfaces • Technology object data types • Function blocks without static variables.
5200	The data type definition contains a recursion, either directly or indirectly. This is not permitted. Do not use this data type at the position concerned.
5201	The function call creates a recursion, either directly or indirectly. This is not permitted. Do not call the function at the position concerned.
5500	The specified jump label identifier was already defined. Choose a different name.
5501	The specified jump label identifier has not been defined. Include this identifier in the LABEL declaration.
5502	The jump label identifier has been assigned more than once. However, each jump label can only be used once as a label.
5503	The jump label is specified as a jump destination, but the associated label is missing.
5504	No jumps are possible in subordinate control structures (e.g. WHILE loops). The specified jump label cannot be used at this position.
5505	No jumps are possible in subordinate control structures (e.g. WHILE loops). The specified jump destination cannot be reached.
5506	No jumps are possible in WAITFORCONDITION blocks. The specified jump label cannot be used at this position.
5507	No jumps are possible in WAITFORCONDITION blocks. The specified jump destination cannot be reached.
5509	Jump labels cannot be used within a CASE statement. The syntax does not allow any differentiation between a jump label and the value list of the CASE statement.

Errors in the expression (6001 ... 6302)

Table 7-515 Errors in the expression (6001 ... 6302)

Error	Description
6001	Syntax error: A statement terminated with a semicolon is expected, e.g. a := b*c;
6002	Syntax error: An expression is expected, e.g. x < y .
6003	The specified identifier is no variable identifier. You must specify a variable identifier. Check whether the indicated identifier is covered. Up to and including V4.0, access to global device identifiers was possible within a program or function block of the same name despite warning 16021.
6004	The index for array access must be the DINT data type. Perform a suitable type conversion or use another expression.

Error	Description
6005	Type conflict in the expression. One of the operands cannot be converted to the data type of the calculation, or the result assignment produces a type conflict.
6006	The specified variable cannot be accessed. Therefore it cannot be used in the expression. Possible causes: • Variable cannot be read. • Attempt to access a local variable of a function or function block from outside.
6007	Cannot write specified variable. A value assignment is not possible.
6008	The specified function does not supply a return value. An application in the expression is therefore not possible (function declared with a return value of VOID).
6009	The specified identifier does not refer to a function or a function block instance. Therefore it cannot be used as function identifier. When calling a program, the compiler option "Allow language extensions" should have been set (-C lang_ext).
6010	The specified identifier is not included as an input parameter (VAR_INPUT) or in/out parameter (VAR_IN_OUT) in the declaration of the POU (function or function block). It cannot be used in the POU call.
6011	The number of function arguments in the call differs from the declaration, or the call parameters required are missing in the call.
6012	RETURN is not permitted syntactically at this position. RETURN may only be used in functions.
6013	EXIT is not permitted syntactically at this position. EXIT can only be used within FOR, WHILE, and REPEAT.
6014	The specified index value is outside the array limits. Only index values that match the array declaration are permissible.
6015	The specified task control command cannot be applied to the task. It is not permitted for this type of task.
6016	The specified task is deactivated in the execution system. It must be enabled before it can be used.
6017	Syntax error on specifying programs within a task. The programs must be listed by name and separated by commas.
6018	The specified identifier does not refer to a PROGRAM. Therefore it cannot be used as a program identifier.
6019	Multiple assignment of program to task. Only one assignment is possible.
6020	Syntax error on specifying directly displayed variables. Inputs must have the syntax %Ix.y and outputs the syntax %Qx.y. Further valid address declarations are %IBx or %QBx for byte access operations, %IWx or %QWx for word access operations and %IDx or %QDx for double word access operations.
6021	The specified byte offset of the directly displayed variables lies outside the permissible address space.
6022	The specified byte offset of the directly displayed variables lies outside the permissible address space. Values 0 to 7 are permissible.
6023	The return value of the function or method has not been assigned. An assignment is however imperative.
6024	A variable with the specified identifier is not included in the task start information.
6025	The condition variable and condition values of a CASE statement must be of the data type SINT, INT, DINT, USINT, UINT or UDINT. It must be possible to implicitly convert the condition values to the data type of the condition variables.
6026	The specified message identifier is not contained in the message configuration. Switch to the message configuration and add the identifier.

Error	Description
6027	System variable access is only possible directly by means of a technology object reference. Access by means of a structure or array is not possible. Create a local variable of type TO and assign the TO reference to this variable. You can then access the required system variable by means of this local TO variable.
6028	Type conflict in expression at specified operation. One of the operands cannot be converted to the data type of the calculation, or the result assignment produces a type conflict. The specified data type in the expression is expected.
6029	The specified function parameter does not have a default value, so it is imperative to specify a value when the function is called.
6030	An attempt was made to transfer an expression to an in/out parameter (VAR_IN_OUT). This is not possible. User variables must be specified as in/out parameters.
6031	An attempt was made to transfer a system variable (TO, I/O direct access) to an in/out parameter (VAR_IN_OUT). This is not possible. User variables must be specified as in/out parameters.
6032	An attempt was made to transfer a variable in the process image to an in/out parameter (VAR_IN_OUT). This is not possible. User variables must be specified as in/out parameters.
6033	An attempt was made to transfer a variable with a non-matching data type to an in/out parameter (VAR_IN_OUT). However, an Implicit type conversion is not possible. User variables with the correct data type must be specified as in/out parameters.
6034	An attempt was made to transfer a read only variable to an in/out parameter (VAR_IN_OUT). This is not possible. In/out parameters must be read/write.
6035	An attempt was made to transfer a constant to an in/out parameter (VAR_IN_OUT). This is not possible. In/out parameters must be user variables.
6036	An operation is applied to a constant. The value of the constant is outside the definition range for the function. Examples are: • Application of SQRT to a negative number. • Use of logarithmic functions on a number <= 0. • Use of ASIN or ACOS on a number outside the interval [0..1]
6037	An attempt was made to divide a constant by zero. This operation is not permitted.
6038	The specified function parameter occurs more than once in the argument list.
6039	The specified POU (function or function block) cannot be used. Possible causes: • The definition of the POU in the implementation section is missing. Only the prototype was specified in the interface section. • The POU is fully defined only after its use (e.g. call, instance declaration). If necessary, move this POU in the program source before the POU in which it is used. • An instance of the function block cannot be declared as unit variable in the same program source in which this function block is defined.
6040	Only simple variables may be used as semaphores; indexing is not possible.
6041	The message function requires an auxiliary value of the specified data type. Type conversion is not possible.
6042	The message function requires that you specify a message number. The specified message number is invalid.
6043	Only simple variables that belong to the static instance data of the class can be used as semaphores.
6050	There is a type conflict in the expression for the specified operation / variables. One of the operands cannot be converted to the type of the calculation, or the result assignment produces a type conflict. A conversion between source file type and target type is not possible.

Error	Description
6051	The expression contains a type conflict for the specified operation. One of the operands cannot be converted to the data type of the other operand to perform the calculation, or the operand data types are not permitted for this operation.
6052	Type conflict in the expression. The specified data type cannot be used for the operation (see marshalling functions).
6053	The expression contains a type conflict for the specified operation. This operation is not permissible on the specified data type.
6054	Type conflict in the expression. The specified variable cannot be used as indexed array variable.
6055	The expression contains a type conflict for the specified operation. Dynamic data type conversion is not permitted for the operand.
6060	At the function call, there is a mixture of assignments of function arguments and setting parameters. Use one form of the function call. Example: • f (x, y); or • f (in1 := x, in2 := y);
6061	The specified parameter of the function or the function block is an in/out parameter. Consequently, a variable must be assigned when the POU is called.
6062	The specified identifier cannot be used as a function argument. Only variables from the declaration blocks VAR_INPUT and VAR_IN_OUT are permitted.
6063	The specified identifier cannot be used as a function argument. Only variables from the declaration blocks VAR_INPUT and VAR_IN_OUT are permitted.
6064	The specified POU is a prototype, for which there is no implementation. Therefore, calling is not possible.
6070	Access to configuration data is only possible for variables that have been specified completely. Append the name according to the configuration data for the selected technology object.
6071	Access to configuration data is only possible for variables that have been specified completely. Therefore, array indices, which cannot be resolved until runtime, may not be used.
6080	The specified variable is no input or output variable that can be directly accessed. Such a variable must be declared in the I/O container of the associated device; it must have the syntax PI* or PQ*.
6085	The specified variable has been declared within the source file with a data type declared by forward declaration. It is not possible in this instance to use public subcomponents of this variable in other declaration sections as initialization values for other interface variables or references. The use of subcomponents by means of ARRAY access is also not possible.
6100	The specified construct can only be compiled if the device type to be used is stated. To do this, set the required device variants at the library. It is not possible to compile the construct as a device-neutral library.
6110	The specified construct cannot be used in libraries.
6111	The specified construct cannot be used in libraries.
6112	The specified construct cannot be used in libraries.
6113	Access to technology objects and devices is not permitted in libraries.
6114	A function block from a different source of the same library has been used. This is only possible within a library if the declaring source is compiled with the "Permit forward declarations" compiler setting activated.
6130	The specification of an interval is not permissible for the data type indicated in the CASE statement.
6140	The specification of a constant in ENUM_TO_DINT requires specifying the data type in the form of enum_type#value.

Error	Description
6141	An attempt has been made to access a value of an enumeration data type (ENUM) that cannot be assigned in the current scope. The specified value must be preceded by the declaration scope of the enumeration data type. This is the name of the CLASS, the FUNCTION_BLOCK or the NAMESPACE within which the relevant enumeration data type has been declared. Furthermore, you should enter the identifier of the enumeration data type before you specify the value. Use the following notation: <scope_name>.<enum_type_name>#<enum_value>.
6150	The specified bit offset lies outside the valid range for the specified data type.
6160	The stated array type without length specification is only permitted: • when VAR_IN_OUT parameters are declared in functions, function blocks or methods, • when VAR_INPUT parameters are declared in functions and methods. These kinds of variables cannot be declared in any other declaration sections.
6161	The direct assignment between arrays without length specification is not possible. You need to iterate over the elements for the assignment.
6170	The specified identifier is no variable identifier. You must specify a variable identifier. It should be noted that local variables of a POU are not usable in an independent EXPRESSION (condition).
6180	The specified function is not usable for variables that contain structures with overlapping memory spaces (OVERLAP).
6200	Only with compiler option "Permit language extensions" (-C lang_ext): The called PROGRAM contains instance data (VAR ... END_VAR declaration block) stored in the user memory of the assigned task. This means a call of the PROGRAM from another POU is not possible. Compile the source file with the "Create program instance data only once" compiler option (-C prog_once) or remove the instance data.
6201	Only with compiler option "Permit language extensions" (-C lang_ext): The call of a PROGRAM is not supported in functions. Such calls can be made only in function blocks or another PROGRAM.
6250	The specified method is not declared public. It has not been declared PUBLIC and cannot therefore be called.
6251	SUPER cannot be used in the current context. SUPER can be used in class methods only if the relevant class has a base class (EXTENDS required).
6252	The specified method is implemented in the base class. It cannot be called directly. To call the method, use either SUPER (call the method of the base class with static binding) or THIS (call with dynamic binding). A call that is not preceded by these names is supported only for methods implemented directly in the current class. The call syntax without a preceding name cannot be used for methods of the base class. An error message is also issued if a method without a preceding name is called that is programmed to be overridden after the current call point. In this case, the source file can be successfully compiled after the compiler option "Permit forward declarations" has been set.
6253	The specified method is not identified as public in the base class. It has not been declared as PUBLIC or PROTECTED and cannot therefore be called.
6254	The specified method is implemented in the base class. To be able to recognize any method override mechanism programmed in the current class, this method must either be specified before it is called for the first time or the compiler option "Permit forward declarations" must be activated. The method can be successfully compiled with this option even if the method is not to be overridden.

Error	Description
6260	A reference cannot be generated on the specified variable. References can only be generated on elements that are in the global memory of the controller (VAR_GLOBAL). It cannot be ensured that the specified variable is in the global memory of the controller. It is therefore not possible to generate references to the following variables: • Input parameters (VAR_INPUT) and output parameters (VAR_OUTPUT) for functions or methods • In/out parameters (VAR_IN_OUT) for functions, function blocks or methods • Temporary variables (VAR_TEMP as well as VAR in functions and methods) • Retentive local variables (VAR RETAIN) • Variables of technological objects • Variables of the address list
6261	A reference cannot be generated on the specified variable. Variables to which references are generated must be read/write. It is therefore not possible to generate references to constants.
6262	A reference cannot be generated on the specified variable. Type conversion and dereferencing is not possible within REF().
6300	Only when pragma "ToHookApplicable = YES" is specified: The call to the specified POU is not possible since this POU is not marked as ToHookApplicable. Only the other functions suitable for this can be used in hook functions.
6301	Only when pragma "ToHookApplicable = YES" is specified: The access to the specified variable is not possible while compiling for use in TO hooks. In TO hooks, the following variables cannot be accessed: • Variables of technology objects • I/O variables • Global device variables
6302	Only when pragma "ToHookApplicable = YES" is specified: It is not possible to call the specified method because this method requires a dynamic call (i.e. via the VTable). Only static method calls may be used in hook functions.

Syntax errors, errors in the expression (7000 ... 7014)

Table 7-516 Syntax errors, errors in the expression (7000 ... 7014)

Error	Description
7000	A syntax error has occurred. Possible causes: • Incorrectly ended control structures (e.g. END_IF missing) • Statements not terminated with ; • Missing brackets
7001	The specified identifier does not refer to a constant. Please enter one constant per value or identifier.
7002	A signed integer is expected. The integer can be of data type SINT, INT, or DINT.
7003	When specifying the interval, the initial value must be less than or equal to the end value. This applies to the declaration of arrays and the specification of the interval in CASE selection conditions.
7004	An initialization value is expected. The value must be a constant. Constants can be assigned as follows: • Directly per value • Symbolically via a preceding constant declaration • As an expression containing constants only

Error	Description
7005	If identical data types are to be initialized in different sources, this requires an identical initialization value too. Adapt the initialization values.
7009	An expression that supplies data type BOOL is expected as condition for WHILE, REPEAT, and IF. This can be specified as a variable of data type BOOL or via a comparison expression. You can also specify a function with a return value of data type BOOL.
7010	A syntax error has occurred. Possible causes: • Incorrectly terminated control structures (e.g. END_IF missing) • Statements not terminated with ; • Missing brackets
7011	A syntax error has occurred. Possible causes: • Incorrectly terminated control structures (e.g. END_IF missing) • Statements not terminated with ; • Missing brackets
7012	A syntax error in the statement, that starts at the specified line, has occurred. Possible causes: • Incorrectly terminated control structures (e.g. END_IF missing) • Statements not terminated with ; • Missing brackets
7013	A syntax error has occurred. An illegal construct is being used.
7014	A syntax error has occurred. Possible causes: • Incorrectly terminated control structures (e.g. END_IF missing) • Statements not terminated with ; • Missing brackets
7068	A source is being used by another source with incorrect RT interface stamp. Compile the used sources individually. In case of doubt, select "Project > Save and compile all".

Error when linking a source file (8001 ... 8010)

Table 7-517 Error when linking a source file (8001 ... 8010)

Error	Description
8001	The specified POU has been exported to the INTERFACE section, but an IMPLEMENTATION section is missing. Either delete the export statement or specify a valid implementation.
8002	The data structure of the specified POU cannot be determined. The cause of the problem is that the POU contains other POUs that are resulting in recursion in the data structure, or that POUs are used that have not been implemented. The message is issued in conjunction with the compiler option 'Permit forward declarations'. Note further error messages indicating the exact error location at which recursion or lack of implementation has been detected.
8010	The interface method is not implemented by the specified class. Since the interface is referenced at the class by IMPLEMENTS, it is absolutely essential that the method is implemented. This also applies if a base class already contains a method implementation of the same name. When an interface is linked, this implementation must be explicitly overridden in the class in which the interface link is programmed.

Errors while loading the interface of another UNIT or a technology package (10000 ... 10101)

Table 7-518 Errors while loading the interface of another UNIT or a technology package (10000 ... 10101)

Error	Description
10000	The specified unit has an invalid file format. Probably, the unit was created using an older version of the compiler or compiled using incompatible options. If a unit is involved, it should compiled first. Then repeat the current compilation. If a package is involved, a newer version should be installed.
10001	The unit name has an invalid format. The rules for identifiers in ST are also true for unit names; the following restrictions apply to their length: • Up to version V4.0 of the SIMOTION Kernel: 8 characters. • As of version V4.1 of the SIMOTION Kernel: 128 characters.
10002	Error while loading the interface of another UNIT, a library or technology package. The specified identifier is contained in two different imported units, libraries or technology packages. • Remove a unit, library or technology package from the import list or • Establish uniqueness between the identifiers in imported units, libraries or technology packages. Change the exporting units in the interface section or specify a namespace for a library or a technology package (USELIB ... AS namespace; USEPACKAGE ... AS namespace;).
10003	The specified data type has an invalid memory layout. Probably, the unit was created using an older version of the compiler or compiled using incompatible options. If a unit is involved, it should compiled first. Then repeat the current compilation. You can also perform "Save and recompile everything". If a package is involved, a newer version should be installed. If the error persists, inform the support department.
10004	The exported symbols of the stated unit could not be loaded. Compile the specified unit and try again. Alternatively, perform "Save and compile changes".
10005	A recursion was detected on loading packages. The specified package has already been loaded with USEPACKAGE and cannot be specified a second time.
10006	A recursion was detected on loading the unit. The specified unit has already been loaded with USES and cannot be specified a second time.
10007	The maximum number of imported units which can be referenced in a unit was exceeded. A maximum of 223 imported units per load unit are permissible. This number includes units that are imported directly with USES as well as units containing used variables or POUs that are indirectly imported by these units.
10008	The number of imported packages that can be referenced in a unit has been exceeded. A maximum of 127 imported packages per load unit are permissible.
10009	The specified package is used in the unit, but it is not available on the device. This error message occurs when you compile with the "implicit package utilization" option and have programmed a USEPACKAGE statement that has a different content than the packages specified on the device.
10010	The specified package is used in unit <a>, but not in unit <b>. This error message occurs when different packages have been specified with USEPACKAGE in units that reference each other with USES. Correct the USEPACKAGE statements.
10011	The specified unit is used directly or indirectly by itself via one or more units. Correct the USES statements.
10012	The specified unit is imported directly or indirectly into several units in different compilation versions. Recompile all units that reference the specified unit in the USES statement.
10013	The specified unit has not yet been compiled, or an error occurred during the last compilation. Compile this unit first to ensure successful compilation.

Error	Description
10014	The type of specified technology object (TO) is not supported by the package specified previously during compilation with USEPACKAGE. Use a package that contains the TO type.
10015	The maximum number of technology objects (TO) which can be referenced in a unit was exceeded. A maximum of 65535 TOs can be referenced.
10016	The device type parameter is not available. If the unit to be compiled is not allocated to a device, use the statement UNIT <unitname> : <typname>;
10017	The device type has not been specified uniquely. The statement UNIT <unitname> : <typname>; has been used to specify a device type in the unit that is different from the device type detected from the assignment of the unit to the device or library.
10018	The specified unit could not be found. Check whether the unit name is available in the PROGRAM container of Workbench or whether the specified file is contained in the current working directory (only u7bt00ax - command line).
10019	The specified technology package could not be found. Observe the preceding error outputs.
10020	Error occurred while loading the technology package. Observe further error outputs.
10021	The technology package is used in the specified source file, however, it is not selected on the device. Correct the USEPACKAGE statement, or select the technology package on the device.
10022	The specified technology package is being used with different versions. Correct the settings for the technology package selection on the device and, if required, in the library. Only one version of a technology package can be used on a device. If this message is issued after the CPU version has been upgraded, select "Save and recompile all".
10024	The specified technology package does not contain any components which can be used in the programming environment. Therefore, it cannot be used in the USEPACKAGE statement either.
10025	The specified identifier is not an identifier for a valid or an installed technology package. Therefore, it cannot be used in the USEPACKAGE statement either. If this technology package is an OEM package, the error message can be eliminated by installing this package. Otherwise remove the identifier from the statement.
10026	The technology package is used in the specified source file; however, it is not selected for the current device type on the library. Correct the USEPACKAGE statement in the source file or select the technology package for the device type.
10027	The specified source file is imported directly or indirectly into several units in different compilation versions. Recompile all units that use the specified unit. Do this using "Save and recompile all".
10030	The device type has not been specified uniquely. In the unit, the statement UNIT xx : dvtype; specifies a different device type than the one determined via the assignment of the unit to the library container.
10031	The specified library is used directly or indirectly by itself via one or more libraries. Correct the USELIB statements.
10032	The specified library could not be found. Check your project.
10033	A recursion was detected on loading the library. The specified library has already been loaded with USELIB and cannot be specified a second time.
10034	The specified library is not compiled for the required device type. Possible causes: • The device type is not selected at the library. • The specified library has not yet been compiled. The message might also be displayed if an error occurred in the last compilation. To remedy the situation, check the device settings at the library and compile the library.
10035	The specified library could not be found. Check whether the library name is available in the Workbench project or whether the specified file is contained in the current working directory (only u7bt00ax command line).

Error	Description
10036	The specified package is used in the source file, but it is not available in the library. Libraries are generally compiled against the package versions specified in the library container. You have programmed a USEPACKAGE statement that has a different content than the packages specified in the library. Either select the correct package version or remove the USEPACKAGE statement from the source file.
10037	The code variant for the current device type is not selected for the specified library. This means this library cannot be used. Activate the code variant for this library.
10038	A DCC library can only be used in the DCC. It is not permissible to integrate such a library into different programming languages.
10039	Compiling a library requires access to the sources. Access to the specified source is not possible due to the selected protection level of the know-how protection. You have to log in for compiling.
10050	All POUs exported by the specified libraries are identical. Simultaneous use of these libraries is not possible. Modify at least one exported element in one of the relevant libraries. Note: Typically, this error occurs in conjunction with know-how protected libraries when they are imported and used jointly in the project used to create the library.
10051	The specified POUs are identical. Simultaneous use of these classes and therefore these libraries is not possible. Note: Typically, this error occurs in conjunction with know-how protected libraries when they are imported and used jointly in the project used to create the library or when copies of such libraries are used jointly.
10100	The specified type of a technology object is contained in several packages that were referenced by the source file. Please choose the technology package that meets your requirements.
10101	The specified technology object is not compatible with the types of technology objects supported by the loaded packages Update the package or change the type of technology object.

Implementation restrictions (15001 ... 15700)

Table 7-519 Implementation restrictions (15001 ... 15700)

Error	Description
15001	The specified construct is not supported by the current version of the compiler.
15002	The currently selected device does not support the specified function. Select a different device version if you want to use this function. To do so, replace the CPU in the hardware catalog and, if necessary, update the firmware.
15003	The specified identifier is a keyword that is not supported and therefore cannot be used as user-specific in order to ensure compatibility with later compiler versions.
15004	The specified identifier denotes a standard function that is not supported and cannot be used as user-specific identifier in order to ensure compatibility with later compiler versions.
15005	The specified identifier denotes a non-supported standard function and cannot be used as user-specified identifier in order to ensure compatibility with later compiler versions.
15006	The specified construct can only be used in source files generated with MCC. Usage in ST is not possible.
15007	A source/library/package is used in the implementation section either directly or indirectly without specifying a namespace. In the interface section, it is used with a namespace. Solve this conflict by specifying a namespace in the interface section for the specified source/library/package.

Error	Description
15008	The specified source uses the library or the technology package with different namespace specification between the interface and implementation section. For successful compilation, the library or the technology package must have already been homed to the interface section with USELIB or USEPACKAGE.
15010	The current device setting does not support the specified function. Select a different device version if you want to use this function.
15060	The currently selected device does not support the specified function. Select a different device version if you want to use this function. To do this, exchange the CPU in the hardware catalog and, if necessary, update the firmware. For example, use of the keyword CLASS or INTERFACE requires a CPU version as of V4.5.
15061	The current device setting does not support the specified function. Select a different device version for the library if you want to use this functionality. For example, use of the keyword CLASS or INTERFACE requires a CPU version as of V4.5.
15070	The specified construct does not conform to the language standard; however, for compatibility reasons, it is supported for old platforms. Convert the usage to the specified alternative.
15152	A USES, USELIB, or USEPACKAGE statement was found in a source file section hidden by conditional compilation. This is not permitted. Source file sections that contain these statements cannot be complied conditionally.
15153	The specified definition is not considered during code generation. It is not possible to define keywords differently.
15154	It is not permitted to apply a line comment and to use multi-line comments in the definition section.
15200	The specification of a bit offset for a bitstring variable requires activation of the "Permit language extensions" compiler option (-C lang_ext).
15700	The specified construct is not supported by the version of SIMOTION SCOUT into which conversion is to be performed.

Warnings (16001 .. 16700)

You can control the output of warnings and information:

- In the global compiler settings
- In the local compiler settings
- In an ST source file by specifying the following attribute within a pragma:
  { _U7_PoeBld_CompilerOption := warning:*n*:on } or
  { _U7_PoeBld_CompilerOption := warning:*n*:off }, where *n* is the warning class or the number for the warning or information.

You can also redefine individual warnings and information as errors:

- In an ST source file by specifying the following attribute within a pragma:
  { _U7_PoeBld_CompilerOption := warning:*n*:err }, where *n* is the number of the warning or information.

Table 7-520 Warnings (16001 .. 16700)

Error	Description
16001	(Warning class: 0) Only in conjunction with the "Selective Linking" compiler option. The specified function, the function block, or the program are neither exported nor called in the current unit. No code is generated.
16002	(Warning class: 0) Only in conjunction with the "Selective Linking" compiler option. The specified unit does not contain any exported PROGRAM nor any task link. No code is generated for the unit.
16003	(Warning class: 2) The operands of the comparison operation do not contain any explicit type definition. The data type listed in the comparison can be seen in the warning message issued. Specify the data type of the used constants explicitly with <type>#<value>.
16004	(Warning class: 2) The specified type conversion may cause the variable value to change due to the reduced display width or inadequate accuracy of the target data type.
16005	(Warning class: 2) During type conversion, the dependency of the variable value can cause the sign to change.
16006	(Warning class: 2) The specified value will be rounded to the next displayable value due to insufficient display width. This is generally the value 0.0.
16007	(Warning class: 2) A loss of accuracy occurred during type conversion. Not all decimal places are considered.
16008	(Warning class: 2) A loss of accuracy occurred during initialization of the specified variables. The constant will be converted to the specified data type. Not all decimal places are considered.
16009	(Warning class: 0) Not in conjunction with compiler option "Selective linking". The specified unit does not contain any exported PROGRAMs nor a task link. Unable to access unit code. Unable to call relevant POU.
16010	(Warning class: 0) Specified program not exported to unit; therefore unable to use it in configuration of the execution level.
16011	(Warning class: 0) The source file does not contain any exported global variables. No data is loaded to the target system.
16012	(Warning class: 0) The specified source file name was taken over from the PROGRAMS container of the selected device. The designation of the source file in the UNIT statement was ignored.
16013	(Warning class: 2) Because of the marshalling function, the specified data type is not portably convertible. Only use SIMOTION devices in conjunction with this data type, or perform an explicit conversion of the data type.
16014	(Warning class: 2) With the specified operation, a data type conversion is performed between signed and unsigned. Because the bit string is adopted in this case, the resulting numerical value can differ from the specified value.
16015	(Warning class: 2) For the assignment of the character string constants to the variables, only part of the character string constants is transferred, because the length of the variable is insufficient to accept all characters.

Error	Description
16016	(Warning class: 2) The operands in the expression do not contain any explicit type definition. The data type of the operation is determined by specifying the values. The resulting data type in which the expression is calculated can be seen in the issued warning message. To define the data type: • Specify the data type of the used constants explicitly with <type>#<value> or • Use an explicit data type conversion.
16017	(Warning class: 2) The operands in the expression contain only constants. The data type of the operation can be determined by specifying the data type (in the form <type>#<value>) or explicit data type conversion. This output is used for troubleshooting, in particular, for the use of symbolic constants, because the data type of the operation cannot normally be determined easily.
16018	(Warning class: 2) The data type of the comparison operation is defined using the value of a constant that has a larger value range than the contained variable. The comparison is performed with the data type of the constant.
16019	(Warning class: 1) The declaration conceals the identifier declared in the scope of the class or the function block. This identifier can no longer be accessed from the POU in which the identifier is declared.
16020	(Warning class: 1) The declaration hides the specified identifier which has been globally defined in its own source file or an imported source file. Access to the global identifier is no longer possible from the POU where this identifier is declared locally.
16021	(Warning class: 1) The declaration hides the specified identifier which is defined on the device. You can access the global device identifier with _device.<name>.
16022	(Warning class: 1) The declaration conceals the specified identifier which is defined in the project (e.g. technology object or device). You can access the global project identifier with _project.<name>.
16023	(Warning class: 1) The declaration hides the specified identifier for the data type of a technology object. Access to the data type identifier is no longer possible.
16024	(Warning class: 1) The declaration hides the access to the technology object on the device. You can access this TO with _to.<name>.
16025	(Warning class: 1) The declaration hides the IEC standard function with the identical name. Access to this function is no longer possible in the current context.
16026	(Warning class: 1) The specified identifier is reserved by SIEMENS for potential extensions. The use of this identifier can cause compiler errors in later versions. If you want to avoid this, change this identifier.
16027	(Warning class: 1) The specified identifier which is reserved for access to I/O qualities, is already assigned on the device. This means that I/O qualities cannot be accessed.
16030	(Warning class: 1) A label has been specified several times in a CASE statement. Only the first label is ever evaluated. Other specifications have no effect.

Error	Description

16040	(Warning class: 1)
	Within a CASE statement, the stated identifier is interpreted as a jump label and not as a CASE selector. This may occur if a LABEL is defined in a control structure within a CASE statement:
	CASE sel OF
	first_val: IF TRUE THEN
	second_val: // -> warning 16040
	result := 2; END_IF;
	third_val: result := 3;
	END_CASE;
	This warning will not be issued if the jump label has been explicitly declared between LABEL and END_LABEL.

16102	(Warning class: 3) The option for output of code for the program status diagnosis function is ignored because no debug information was generated. Output of debug information was deactivated via compiler options.
16103	(Warning class: 3) The option for outputting code at the library for the program status diagnosis function is ignored. The code for program status is generated as defined in the option in the individual source files.
16110	(Warning class: 3) The specified pragma statement is not supported in the current version.
16111	(Warning class: 3) The specified pragma statement is not supported in the used context. The position of the pragma in the source text is not correct.
16112	(Warning class: 3) A valid pragma has been used. However, the value specified for the pragma is not valid. Use a valid value.
16113	(Warning class: 3) The pragma contains a syntax error in an attribute declaration. The valid syntax is as follows: <Attribute identifier> := <Attribute value>; .
16150	(Warning class: 7) A new definition has been made for the specified identifier. Consequently, the previous definition is invalid. This warning enables the work of the preprocessor to be tracked.
16151	(Warning class: 7) An attempt has been made to delete the definition of the specified identifier with #undef. However, the identifier is not defined or the definition is already deleted. This warning enables the work of the preprocessor to be tracked.
16152	(Warning class: 7) The specified definition is not considered during code generation. This may be caused by the preprocessor being deactivated for the compiled source.
16153	(Warning class: 7) The preprocessor is not active in the current source, even though preprocessor statements are used. Activate the preprocessor or remove the statements.
16154	(Warning class: 10) The preprocessor cannot be used to control the contents of USEPACKAGE, USELIB, and USES statements. It is no longer possible to automatically determine the dependencies between sources for the "Save project and compile changes" and "Save project and recompile all" functions.

Error	Description
16170	(Warning class: 10) The definitions from sources imported using USES are not considered during code generation.
16171	(Warning class: 10) The definition from the specified source imported using USES could not be loaded. Compile the specified source file beforehand.
16200	(Warning class: 4) The use of a semaphore requires a global variable to enable access to it from a different task. Local task operations do not have to be blocked via semaphores.
16210	(Warning class: 4) The basis of the exponential function (EXPT standard function or ** operator) is negative. The operation can be executed at run time only under the following conditions: 1. It can be used on a device with a version of the SIMOTION Kernel as of V4.1. 2. The exponent is an integer. The ExecutionFaultTask will be initiated for non-integer exponents or for use on a device with a version of the SIMOTION Kernel up to V4.0. The program will be aborted at this position.
16220	(Warning class: 4) The condition of an IF statement, WHILE statement or REPEAT statement is a constant expression.
16230	(Warning class: 4) The expression with the specified values does not cause any change to the result; optimized code will be created.
16240	(Warning class: 4) The expression with the specified values exceeds the definition range of the operation. The result may be incorrect.
16250	(Warning class: 4) A modification of the control variable of a FOR loop occurs inside this loop. This modification is either not effective or may cause an unexpected result when editing this loop. If the modification of the variables inside the loop is necessary, then use WHILE or REPEAT at this point instead of FOR.
16260	(Warning class: 4) With the values assigned to the specified function, it has no effect. Optimized code is generated.
16261	(Warning class: 4) The return value of the function or method has not been assigned. The result of the function is thus lost.
16270	(Warning class: 4) The operation for the dynamic type conversion is executed as assignment operation. The associated variables enable an evaluation at the time of the compilation. As a result of this message, it may be determined that a type conversion is not possible. An error message is output as for an assignment of the variables.
16280	(Warning class: 4) The INTERNAL access identifier was specified for an element not contained in any NAMESPACE. The identifier is ignored and the element is PUBLIC.
16281	(Warning class: 4) The specified identifier has already been declared as INTERNAL in the current source or in an imported source or library. The current declaration location, however, does not contain the INTERNAL identifier. Add INTERNAL for the declaration of the affected element.

Error	Description
16282	(Warning class: 4) The specified NAMESPACE was identified as public in one source and declared as INTERNAL in another source. This conflict is resolved by declaring the NAMESPACE as INTERNAL. Consequently, the contained elements can be accessed only in accordance with the rules that apply to INTERNAL.
16283	(Warning class: 4) Rather than the PUBLIC access identifier specified for the data type declaration, INTERNAL is used, because the complete POU was declared as INTERNAL. This identifies contained elements as INTERNAL, because they may not be accessed outside the NAMESPACE.
16300	(Warning class: 5) The auxiliary value has a data type that cannot be converted to the data type configured for the message.
16301	(Warning class: 5) The specified auxiliary value is not evaluated during output of the message.
16302	(Warning class: 5) The data type of the auxiliary value cannot be determined from the message configuration. The specified data type is used.
16303	(Warning class: 5) No auxiliary value has been specified for the function although the message configuration requires such a value. A default value of the corresponding data type was added.
16304	(Warning class: 5) An alarm accompanying value is specified using a constant or a constant expression. The resulting data type of the alarm accompanying value can be seen in the issued warning message. To define the data type: • Specify the data type of the used constants explicitly with <type>#<value> or • Use an explicit data type conversion.
16400	(Warning class: 6) A global variable has been declared in a library. This may mean that the library cannot be used more than once.
16401	(Warning class: 6) A PROGRAM has been declared in a library. This cannot be used in the process system. Set the compiler option "Only create program instance data once" (-C prog_once) or declare a FUNCTION_BLOCK.
16420	(Warning class: 6) The return value has not been assigned within the function or a method. If such a function or method is called, it returns a random value.
16421	(Warning class: 6) A variable that has neither been assigned nor read in the code has been declared. The output of this warning can only be suppressed for a POU by means of a pragma.
16430	(Warning class: 6) An attempt is being made to use a function block, which is either a system function block itself or includes one, as a temporary variable. This can result in runtime problems when using the block in a cyclic level.

Error	Description
16440	(Warning class: 10) A global variable that does not contain any user data that can be backed up has been created in the retentive memory area. These kinds of variables should not be declared in RETAIN areas. It is not meaningful to specify RETAIN in this case and this specification is supported only for compatibility reasons TO references are an example of a data type for which this message is output.
16441	(Warning class: 6) The specified data type contains some variables that cannot be restored in addition to those that can be backed up in RETAIN areas. Data types of this kind include, for example, TO references and INTERFACE variables.
16450	(Warning class: 10) A global variable has been created in the retentive memory range. This declaration is not permissible at the specified position.
16451	(Warning class: 10) The initialization of large arrays with values other than 0 causes a high data volume in the controller. This results in long load times as well as high memory utilization.
16452	(Warning class: 10) The specified program has a large quantity of instance data to be initialized. This can lead to a runtime violation when the task is started because both the initialization code and the user code are being executed. In particular, caution is advised in the case of SynchronousTasks.
16470	(Warning class: 10) The specified construct does not conform to the language standard; however, for compatibility reasons, it is supported for old platforms. Convert the usage to the specified alternative.
16600	(Warning class: 6) The specified variable is not contained in the initialization list. The default initialization value is used.
16601	(Warning class: 6) The specified variable is not contained in the initialization list. The default initialization value is used.
16602	(Warning class: 6) The specified variable is not contained in the initialization list. The default initialization value is used.
16603	(Warning class: 6) The specified function block does not contain any instance data and, as a result, has a size of 0 bytes. When transferred as a non-dimensional array for referencing purposes, the size of the element is defined as 1 element.
16604	(Warning class: 6) The data type used in the declaration has no unique initialization value. The initial value known last is used. In order to achieve a unique behavior at this point, the initial value for the variable or data type declaration must be specified explicitly.
16605	(Warning class: 6) The memory space of the specified variable overlaps with another variable. Thus, the initialization value cannot be included. The remaining memory space is assigned the default value 0.
16606	(Warning class: 6) The memory space of the specified variable overlaps with another variable. Thus, the data type cannot be transferred to the OPC XMP information. The component is symbolically unavailable and also not included in backups using _exportUnitDataSet.

Error	Description
16607	(Warning class: 6) The specified function block does not contain any instance data and, as a result, has a size of 0 bytes. The instance declaration in the form of an ARRAY is not appropriate.
16700	(Warning class: 3) The SIMOTION device can also be processed with previous versions of the SIMOTION SCOUT. The specified construct is not supported by all the earlier versions of the compiler.

See also

Global compiler settings (Page 4623)

Local compiler settings (Page 4626)

Controlling the preprocessor and compiler with pragmas (Page 4917)

Controlling compiler with attributes (Page 4922)

Non-assigned compiler messages (22000 ... 22002)

You create non-assigned compiler messages within a pragma (Page 4917). Enter the desired message text as a character string (Page 4667) enclosed in brackets after the keyword "information", "warning" or "error". End the statement within the pragma with a semi-colon. See "Issuing non-assigned compiler message" (Page 4926).

Table 7-521 Non-assigned compiler messages (22000 ... 22002)

Error	*Description*
22000	This message is a user-defined error message. See error text.
22001	(Warning class: 0) This message is a user-defined warning. See warning text.
22002	(Warning class: 0) This message is a user-defined information message. See information text.

Information (32010 ... 32653)

You control this output of information together with the warnings (Page 5058):

Table 7-522 Information (32010 ... 32653)

Error	Description
32010	(Warning class: 6) The specified jump label identifier has been declared but not used.
32020	(Warning class: 10) The specified variable was declared globally in this source file or in another source file with the indicated data type. This may be a variable, a constant, or another identifier. This information helps when searching for the cause of compilation errors. It is issued together with error messages.

Error	Description
32021	(Warning class: 10) The specified variable was declared on the device as an I/O variable, a global device variable, or a system variable. This information helps when searching for the cause of compilation errors. It is issued together with error messages.
32022	(Warning class: 10) The specified variable was declared in the project as a global identifier. This may be a device identifier, a technology object or another identifier from the project. This information helps when searching for the cause of compilation errors. It is issued together with error messages.
32023	(Warning class: 10) Until now, no valid declaration has been found for the specified identifier. This information is issued together with error messages.
32024	(Warning class: 0) The specified variable has been declared as a global identifier in the current unit or in an importing unit. This information helps when searching for the cause of compilation errors. It is issued together with error messages.
32025	(Warning class: 10) The specified identifier was declared in this source or in another source in the specified namespace. In order to be able to use this identifier, it must be prefixed with the identifier of the namespace. This may be a variable, a constant or another identifier. This information is helpful when searching for the cause of compilation errors. It is issued together with error messages.
32026	(Warning class: 0) The specified identifier is a standard function that can be activated via a compiler option. In order to use the function, activate the required compiler options for language extensions.
32030	(Warning class: 0) The stated array initialization does not conform to IEC 61131-3. For portable programs, you should enclose the array initialization values in square brackets. Example of field initialization in compliance with the standard: x : ARRAY [0 to 1] OF INT := [1, 2];
32050	(Warning class: 0) The maximum size that can be reached via an HMI is 65536 bytes. This limit has been exceeded with the specified variable. All subsequent variables cannot be reached either.
32060	(Warning class: 0) The specified identifier is a keyword that can be activated via a compiler option. In order to use the keyword, activate the compiler option "Language extensions IEC61131-3rd Edition". This message is output as a guide for debugging syntax errors.
32061	(Warning class: 0) The specified identifier is a keyword that can be activated via a compiler option. In order to use the keyword, activate the compiler option "Permit object-oriented programming". This message is output as a guide for debugging syntax errors.
32300	(Warning class: 1) A label has been specified several times in a CASE statement. Only the first label is ever evaluated. Other specifications have no effect.

Error	Description
32650	(Warning class: 7) The specified identifier will be replaced thereafter by the output text. This information enables the work of the preprocessor to be tracked.
32651	(Warning class: 7) The definition of the specified identifier has been deleted with #undef. This information enables the work of the preprocessor to be tracked.
32652	(Warning class: 7) The identifier will be used with the specified replacement text in the source file. Compilation takes place with the replacement text. This information enables the work of the preprocessor to be tracked.
32653	(Warning class: 7) The specified identifier will be replaced thereafter by the output text. This information appears if additional replacements are loaded with a USES statement. This information enables the work of the preprocessor to be tracked.

7.2.9.3 Template for Example Unit

Preliminary information

This chapter presents a comprehensively annotated template that you can call in the online help. You can use it as a template for a new ST source file.

```
//============================================================================
//(organization)
//(division / place)
//(c)Copyright 2009  All Rights Reserved
//----------------------------------------------------------------------------
//  project name: (name)
//  file name: (name as soon as saved)
//  library: (that the source is dedicated to)
//  system: (target system)
//  version: (SIMOTION / SCOUT version)
//  application: (relation to project/ product/ usage)
//  restrictions:
//  requirements: (hardware, technological package, memory needed, etc.)
//  search items: (with the purpose of browser usage)
//  functionality: (that is implemented)
//----------------------------------------------------------------------------
//  change log table:
//  version    date      expert in charge     changes applied
//
//============================================================================

INTERFACE
    //  All statements added between INTERFACE and END_INTERFACE/
    //  Keywords are used to define which source contents
    //  (variables, functions, function blocks, etc.) also in other
    //  sources (units) are available or exported.

    USEPACKAGE cam;
    // The technology packages to be used are known here and thus
    // made usable in the source. Technology object (TO)-specific
    // Commands can be used in this UNIT only when the
    // appropriate package has been included.
    // If a source file that uses USEPACKAGE cam is integrated via USES,
    // it will be "inherited". USEPACKAGE can then be omitted.
    // The package used in this example is "cam". However, other
    // technology packages can also be used (see documentation).
    // USELIB testlib;

    // If library functions are to be used in the source file, they must be made
    // known in the source, too. If the library
    // with the name "testlib" does not exist in the project,
    // the error message
    // "Error 10035, "testlib.lib" library could not be loaded"
    // "Error 10032, "testlib" library could not be loaded"
    // will be output.
    // If libraries are not being used, this line can be
    // deleted..
```

```
// USES header;

// USES is used to import contents exported from a different source
// exported content imported and made usable in "sttemp_1_de".
// If the source with the name "header" does not exist in the project,
// the error message
// "Error 10018, "header" source could not be loaded"
// will be output.
```

Type definition in the interface

```
// ******************************************************
// * Type definition in the INTERFACE                   *
// ******************************************************

VAR_GLOBAL CONSTANT
       PI             : REAL := 3.1415;
       ARRAY_MAX1     : INT  := 4;
       ARRAY_MAX2     : INT  := 4;
       COLLECTION_MAX : INT  := 6;
       GLOBARRAY_MAX  : INT  := 12;
END_VAR
// Declaration of a global constant. In the source file
// no other value can be assigned to the identifier.
// User defined variable types (UDT) are
// defined between TYPE and END_TYPE.
TYPE
    ai16Dim1 : ARRAY [0..ARRAY_MAX1-1] OF INT;
    // Definition of a one-dimensional array with four array elements from
    // type INT under the name "ai16Dim1". With "ai16Dim1" as the data type
    // in all source file segments, one-dimensional arrays can now
    // be declared by type INT.
    aaDim2 : ARRAY [0..ARRAY_MAX2-1] OF ai16Dim1;
    // A two-dimensional array is an array of one-dimensional arrays.
    // Here a two-dimensional field with 16 elements occurs
    // of the type INT under the name "aaDim2"
    eTrafficLight : (RED, YELLOW, GREEN);
    // Definition of enumerator "eTrafficLight" as a
    // user-defined variable type. Variables of this type can
    // only accept the values "RED", "YELLOW" and "GREEN".
```

```
    sCollection : STRUCT
        toAxisX          : posaxis;
        aInStructDim1    : ai16Dim1;
        eTrafficInStruct : eTrafficLight;
        i16Counter       : INT;
        b16Status        : WORD;
    END_STRUCT;
    // A user-defined structure is created here. It is possible to
    // combine elementary data types (here INT and WORD) or already defined
    // user data types (here "array1dim" and "eTrafficLight") into
    // one structure. In addition, types
    // of technology objects can also be used.
    // In the example, the structure contains an element of
    type // a position axis (posaxis).
    // In the definition, make certain to sort the variables
    // by size in increasing sequence
    // (ARRAY, STRUCT, LREAL, DWORD, INT, BOOL ...)
    aCollection : ARRAY [0..5] OF sCollection;
    // Nesting is also possible. The type "aCollection"
    // contains a field comprising six elements of type "sCollection"
END_TYPE
```

Variable declaration in the interface

```
// ******************************************************
// * Variable declaration in the INTERFACE                *
// ******************************************************

VAR_GLOBAL  // In the user memory of the UNIT.
            // Also visible using HMI services.
    gaMyArray : ARRAY [0..GLOBARRAY_MAX-1] OF REAL := [3 (2(4), 2(18))];
    // Example of a declaration of a one-dimensional array without
    // previous type declaration. The initialization performed here is
    // read as follows:
    // Two elements each are initialized with the value 4,
    // two elements with the value 18. This pattern is used in the array
    // "gaMyArray" three times in succession.
    // The field elements are thus assigned as follows:
    // 4, 4, 18, 18, 4, 4, 18, 18, 4, 4, 18, 18.
    gaMy2dim : aaDim2;
    // Example of a declaration of a two-dimensional array
    gaMy1dim : ai16Dim1;
    // Example of a declaration of a one-dimensional array with
    // use of a type declaration.
    gsMyStruct : sCollection;
    // Variable of the type or with the structure of
    // user_struct.
    gaMyArrayOfStruct : aCollection;
    // The variable generated here contains a field from
    // structural elements as declared in section TYPE/END_TYPE
    //.
    gtMyTime : TIME := T#0d_1h_5m_17s_4ms;
    // ...as elementary time types and derived data types.
```

```
        geMyTraffic : eTrafficLight := RED;
        // An enumerator of type "eTrafficLight" is created here and
        // assigned the value "RED".
        gi16MyInt : INT := -17;
        // Variables of an elementary numerical data type can
        // also be declared in variable declarations...

    END_VAR
    VAR_GLOBAL RETAIN
    END_VAR
    // The variables declared with the add-on RETAIN are
    // RETAIN stored in the data area of the hardware platform used and
    // are therefore safe from network failure.

    // The declaration of VAR, VAR CONSTANT, VAR_TEMP, VAR_INPUT, VAR_OUTPUT
    // and VAR_IN_OUT is not permissible here.
    // Variables that are defined in this section and thus exported
    // can be reimported by means of the USES "sttemp_1_de" into another source file (UNIT)
    .
    FUNCTION FCmyFirst;
    FUNCTION_BLOCK FBmyFirst;
    PROGRAM myPRG;
    // The function blocks (FBs),
    // functions (FCs) and programs defined in the interface part are exported
    // so that they can be used in other units.
    // Non-exported FBs and FCs can only be used in this source file
    // ("information hiding", placing in the interface only
    // what other units absolutely need).
    // A program that has not been exported cannot be assigned to any TASK
    //.

END_INTERFACE
```

Implementation

```
// ****************************************************
// * IMPLEMENTATION section                           *
// ****************************************************

IMPLEMENTATION
    // In the IMPLEMENTATION section of a unit, the executable code sections
    // are stored in various program organization units (POUs).
    // A POU can be a program, FC, or FB.

    VAR_GLOBAL CONSTANT
    END_VAR
    TYPE
    END_TYPE
    // The type definition can also be made in the IMPLEMENTATION section.
    // However, this definition cannot be imported in another source file.
      The type definition can, however, be used for variables
    // in all POUs of the source file "sttemp_1_de". The definition of types must
    // be performed before the declaration of a variable.
```

```
VAR_GLOBAL  // In the user memory of the UNIT
    gboDigInput1 : BOOL;
    // Boolean variable for "EXPRESSION" example (see below).
END_VAR
VAR_GLOBAL RETAIN
END_VAR
// The variables declared with the add-on RETAIN are
// stored in the RETAIN data area of the hardware platform used and
// are therefore safe from network failure.
// Variable declaration in the IMPLEMENTATION section.
// The declaration of VAR, VAR CONSTANT, VAR_TEMP, VAR_INPUT, VAR_OUTPUT
// and VAR_IN_OUT is not permissible here.
EXPRESSION xCond
    xCond := gboDigInput1;
END_EXPRESSION
// Definition of an EXPRESSION.
// An EXPRESSION is a special function case, which recognizes only the
// return value TRUE and FALSE. It is used in conjunction with the
// statement WAITFORCONDITON (see myPRG) and should only be used
// if the program is executed as part of
// a MotionTask. If "gboDigInput1" (usual in a digital input or a
// condition in the program) takes on the value 1, the return value of the
// EXPRESSION TRUE.
```

Function

```
// ******************************************************
// * FUNCTION                                           *
// ******************************************************

// The declaration of an FB or FC must be placed in the source file
// before the actual use (the call), so that the code of the
// block is already known to the calling point.

FUNCTION FCmyFirst : INT
    // The statement section of the POU FUNCTION begins here. The return value
    // of the function has the type integer in this case.
    // The stack of the calling TASK is initialized on each call
    //.  The return value is located on the stack and is
    // written by the FUNCTION.
    VAR CONSTANT
    END_VAR
    TYPE
    END_TYPE
    // The type declaration can also be made in POUs. The
    // basic difference is the validity of the
    // type declaration. A type declared in a POU can only
    // be used for variables within associated POU.
    VAR_INPUT   // In the stack of the calling TASK, will be placed on
                // stack on call, assignment optional.
    END_VAR
```

```
    VAR         // In the stack of the calling TASK,
                // is used in FUNCTION.
    END_VAR
    // Variable declaration in an FC.
    // The declaration of VAR_GLOBAL, VAR_GLOBAL CONSTANT,
    // VAR_GLOBAL RETAIN and VAR_RETAIN is not permissible here.

    // The use of unit-global variables for data acceptance in FCs
    // and FBs is the fastest option for the runtime. The use
    // of the input parameters VAR_INPUT and the return via the
    // return value is slower, since the values are copied respectively.

    // Comment: Variables declared with VAR and VAR CONSTANT are
    // temporary. On the next call, the contents from the latest
    // call are no longer available, in contrast to the FB.
    // ****************************************
    // * Area for FC code or statements *
    // ****************************************)
    // Code is in the user memory.

    geMyTraffic := YELLOW; // e.g. change the traffic light.

    FCmyFirst := 17;
    // In this example, the function returns the value "17" to the
    // calling program.

END_FUNCTION
```

Function block

```
// ****************************************************
// * FUNCTION_BLOCK                                   *
// ****************************************************

// The declaration of an FB or FC must be placed in the source file
// before the actual use (the call), so that the code of the
// block is already known to the calling point.

FUNCTION_BLOCK FBmyFirst
    // The statement section of the POU FUNCTION_BLOCK begins here.
    // Instance data are dependent where the instance is formed
    // (see comments at the template end) in the user memory of UNIT
    // or TASK and are initialized with STOP->RUN or starting the TASK

    // The pointer to the instance data is transferred during the call.
    VAR CONSTANT
    END_VAR
    // Variables declared with VAR and VAR CONSTANT are
    // static, i.e., on the next block call, their contents remain
    // available and valid.
```

```
    TYPE
    END_TYPE
    // The type definition can also be made in POUs. The
    // basic difference is the validity of the
    // Type definition. A type defined in a POU can only
    // be used for variables within associated POU.
    VAR_INPUT   // In the user memory of the UNIT or TASK,
                // assignment optional on call.
    END_VAR
    VAR_IN_OUT  // In the user memory of the UNIT or TASK,
                // reference must be assigned on call.
    END_VAR
    VAR_OUTPUT  // In the user memory of the UNIT or TASK.
    END_VAR
    VAR         // In the user memory of the UNIT or TASK,
                // can be used in the FB.
    END_VAR
    VAR_TEMP    // In the stack of the calling TASK,
                // is initialized on each call.
    END_VAR
    // Variable declaration in an FB.
    // The declaration of VAR_GLOBAL, VAR_GLOBAL CONSTANT and
    // VAR_GLOBAL RETAIN is not permissible here.
    // ******************************************
    // * Area for FB code or statements *
    // ******************************************

    geMyTraffic := GREEN;  // e.g. change the traffic light.

END_FUNCTION_BLOCK
```

Program

```
// ******************************************************
// * PROGRAM                                            *
// ******************************************************

PROGRAM myPRG
    // The statement section of the POU PROGRAM begins here.
    VAR CONSTANT
    END_VAR
    TYPE
    END_TYPE
    // The type definition can also be made in POUs. The
    // basic difference is the validity of the
    // type definition. A type defined in a POU can only
    // be used for variables within associated POU.
```

```
        VAR  // In the user memory of the TASK.
            instFBMyFirst : FBmyFirst;
            // In order to be able to call an FB, an area for static
            // variables (forming an instance) must be generated. This has to do with
            // the "memory" of the FB.

            retFCMyFirst : INT;
            // Variable for the return value of the function.
        END_VAR
        VAR_TEMP    // In the stack of the TASK, initialized in each pass.
        END_VAR
        // Variable declaration in a PROGRAM.
        // The declaration of VAR_GLOBAL, VAR_GLOBAL CONSTANT,
        // VAR_GLOBAL RETAIN, VAR_INPUT, VAR_OUTPUT and VAR_IN_OUT
        // is not permissible here.
        // Comment: Whether the local variables declared via VAR
        // are temporary variables depends on the task context in which the
        // PROGRAM is used.
        //
        // In non-cyclic tasks (StartupTask, ShutdownTask, MotionTasks,
        // SystemInterruptTasks and UserInterruptTasks), the previous
        // contents of VAR and VAR_TEMP are no longer available.
        // The variables are thus temporary.
        //
        // In cyclic tasks (BackgroundTask, IPOsynchronousTask,
        // IPOsynchronousTask_2 and TimerInterruptTasks), the contents
        // of variables declared in the VAR section remain the same
        // for the following run. The variables are thus static.
        // Variables from VAR_TEMP are always temporary.
        instFBMyFirst ();
        // FB call with a valid instance.

        retFCMyFirst := FCmyFirst();
        // FC call and assignment of return value.

        WAITFORCONDITION xCond WITH TRUE DO
            // The statements programmed here come immediately for
            // execution if the condition EXPRESSION defined in the associated
            // "xcond" is logically true.
            ;
        END_WAITFORCONDITION;
        // WAITFORCONDITION is generally used only in MotionTasks.
          These remain in the location and the
        // condition defined in the EXPRESSION is checked with high priority.

    END_PROGRAM

END_IMPLEMENTATION
```

7.3 SIMOTION LAD/FBD

Preface

Scope

This document is part of the **SIMOTION Programming** documentation package.

This document applies to SIMOTION SCOUT, the engineering system of the SIMOTION product family in product version V5.4 in conjunction with:

- A SIMOTION device with the following versions of a SIMOTION Kernel:
  - V5.4
  - V5.3
  - V5.2
  - V5.1
  - V4.5
  - V4.4
  - V4.3
  - V4.2
  - V4.1 [1]
  - V4.0 [1]
  - V3.2 [1]

  [1] V4.5 is the last product version of SIMOTION SCOUT that will support these versions of the SIMOTION Kernel.
- The relevant version of the following SIMOTION Technology Packages, depending on the kernel:
  - Cam
  - Path (Kernel as of V4.1)
  - Cam_ext
  - TControl

Information in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- **Description** (Chapter 1)
  This chapter shortly defines the LAD and FBD programming languages.
- **LAD/FBD Editor** (Chapter 2)
  In this chapter you can learn about the various operator control options in the LAD/FBD Editor.

- **Software Programming** (Chapter 3)
  This chapter shows how to proceed during programming.
- **Functions** (Chapter 4)
  This chapter describes how to apply individual LAD/FBD commands and gives an outline of their function.
- **Debugging Software / Error Handling** (Chapter 5)
  This chapter describes how to test a program and find errors in created programs.
- **Application Examples** (Chapter 6)
  You will be given an introduction to the LAD and FBD programming languages using some simple examples.
- **Appendix**
  - Key combinations
    This appendix contains the keystroke combinations for frequently used commands.
- **Index**
  Keyword index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

7.3.1 Fundamental safety instructions

7.3.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

7.3.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

7.3.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

7.3.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

7.3.2 Description

7.3.2.1 Description

This chapter will give you a brief overview of ladder logic (LAD) and function block diagram (FBD).

7.3.2.2 What is LAD?

LAD stands for ladder logic. LAD is a graphical programming language. The statement syntax corresponds to a circuit diagram. LAD enables simple tracking of the signal flow between conductor bars via inputs, outputs and operations.
LAD statements consist of elements and boxes which are graphically connected to networks (which are displayed in conformity with the IEC 61131-3 standard). LAD operations follow the rules of Boolean logic.

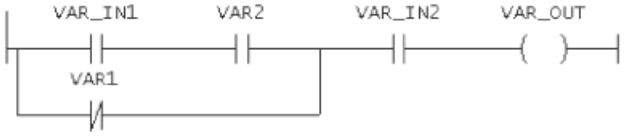


Figure 7-440 Representation of a network in LAD

The LAD program can also be displayed as an FBD program.

The LAD programming language

The LAD programming language features all the elements required for the creation of a complete user program. LAD features an extensive command set. This includes the various basic operations with a comprehensive range of operands and how to address them. The design of the functions and function blocks enables you to structure the LAD program clearly.

The program package

The LAD programming package is an integral part of the basic SIMOTION software, so that after your SIMOTION software has been installed, all editor, compiler and test functions for LAD are available for use.

7.3.2.3 What is FBD?

FBD stands for function block diagram. FBD is a graphics-based programming language that uses the same type of boxes used in boolean algebra to represent logic (networks are displayed in conformity with the IEC 61131-3 standard). In addition, complex functions (e.g. mathematical functions) can be represented directly in conjunction with the logic boxes.

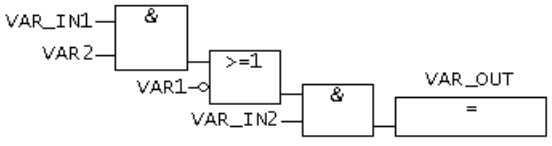


Figure 7-441 Representation of a network in FBD

The FBD program can usually also be displayed as an LAD program.

The FBD programming language

The FBD programming language features all the elements required for the creation of a complete user program. FBD features an extensive command set. This includes the various basic operations with a comprehensive range of operands and how to address them. The design of the functions and function blocks enables you to structure the FBD program clearly.

The program package

The FBD programming package is an integral part of the basic SIMOTION software, so that after your SIMOTION software has been installed, all editor, compiler and test functions for FBD are available for use.

7.3.2.4 Unit, program organization unit (POU) and program source

The term "unit" represents a program source.

The terms "program organization unit (POU)" and "LAD/FBD program" are generic terms and may refer to a program, a function (FC), or a function block (FB).

The term "program source file" is a generic term and may refer to a LAD/FBD unit, an MCC unit or an ST source file.

7.3.3 LAD/FBD editor

7.3.3.1 The LAD/FBD editor in the workbench

The workbench represents the framework for SIMOTION SCOUT. Using the workbench tools, you can carry out all the steps required to configure, optimize, and program a machine in order to complete a required task.

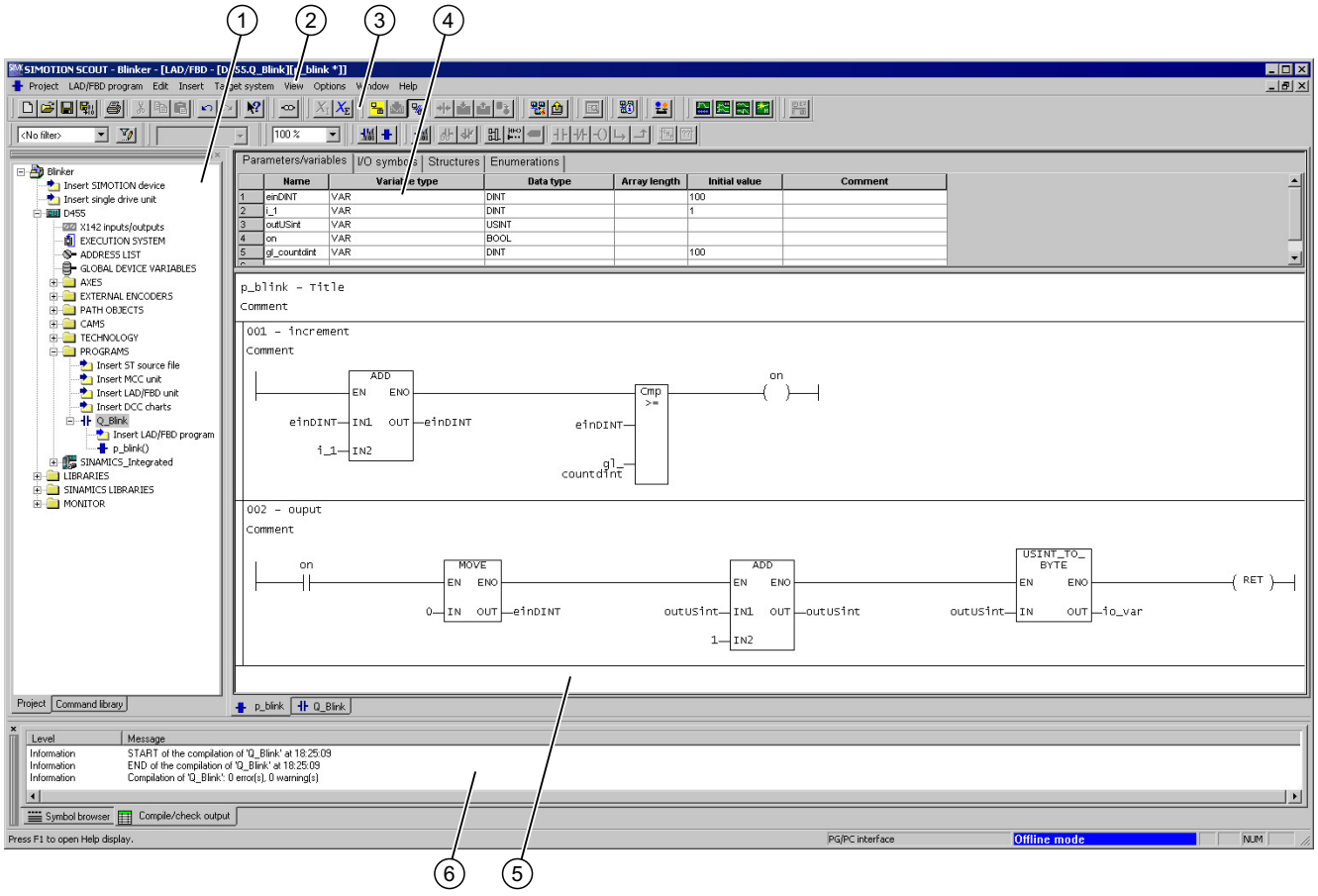


Figure 7-442 Elements of the workbench in a LAD/FBD program

The workbench contains the following elements:

- ① Project navigator
  The project navigator shows the entire project and its elements as a tree structure.
- ② Menu bar
  The menu bar contains menu commands which you can use to control the workbench, call up tools, etc.
- ③ Toolbars
  Many of the available menu commands can be executed by clicking the appropriate icon in one of the toolbars.
- ④ Declaration tables
  Declaration tables are used for LAD/FBD units and programs. You define variables and constants in the declaration tables.

- ③ Working area
  You carry out job-specific operations in this area. The working area contains an LAD/FBD program, a declaration table, and an editor for graphical displays.
- ⑥ Detail view
  More detailed information about the elements selected in the project navigator are displayed, e.g. the windows **Symbol browsers**, **Compile/check output**.

7.3.3.2 Maximizing working area and detail view

The windows working area and detail view can be set to maximum zoom.

The selection is made under the following menu items:

- **View > Maximize working area** (e.g., when creating programs)
  or
- **View > Maximized detail view** (e.g., monitoring global variables)

7.3.3.3 Enlarging or reducing the content of the working area

There are several options available to change the size of the LAD/FBD editor's display, i.e. the size of the elements in this area.

- **Zoom** list on the **Zoom factor** toolbar:
  Select a factor from the **Zoom** list, or enter an integer value of your own choice.
  - or -
  View > Zoom in menu command or **View > Zoom out.**
  - or -
  Key combination **Ctrl+Num+** (enlarge) or **Ctrl+Num-** (reduce).
  - or -
  Press the **Ctrl** key while turning the mouse wheel.

This change always applies to the active LAD/FBD editor. The setting is only saved when saving if changes have been made in the respective editor window.

7.3.3.4 Bringing the LAD/FBD editor to the foreground

If several LAD/FBD editors are open in the working area, these are usually overlaid. This means that only the top LAD/FBD editor is visible. There are several ways to bring the concealed editors to the foreground.

To bring the editor to the foreground, proceed as follows:

- Select the appropriate tab below the working window
  - or -
  Select the appropriate program name in the Window menu.
  - or -
  Press the **Ctrl+Tab** key combination as often as required.

7.3.3.5 Hiding and displaying the declaration table

If you need more space, you can hide the **Interface (exported declaration)** declaration area and/or the declaration area for a LAD/FBD program completely

To hide and display the declaration table, proceed as follows:

1. Double-click the separation line to hide the declaration table.
2. In order to display the declaration line again, double-click the separation line again.

7.3.3.6 Enlarging/reducing the declaration table

To change the size of the declaration table, proceed as follows:

1. Move the mouse cursor onto the separation line until the mouse pointer changes to a double line.
2. Hold down the left mouse button and drag the separation line upwards in order to reduce the size of the declaration area.
   - or -
   In order to enlarge the declaration area, move the separation line downwards.

7.3.3.7 Operation

Operating the LAD/FBD editor

The LAD/FBD editor provides the programmer with a variety of different operator input options. Alternatives for executing individual operator inputs include the following:

- The menu bar
- Context menus
- Toolbars
- Key combinations
- Texts and variables can be moved to the input field with drag-and-drop:
  - From the project navigator
  - From the declaration tables
  - From the detail view (Symbol browser tab, address list, watch table)
  - From the command library

Menu bar

You can start all of the programming functions from the menu bar.

The LAD/FBD program item only appears if a LAD/FBD editor is active in the working area.

Context menu

To use the context menu for an object, proceed as follows:

1. Select the appropriate object with the left mouse button (left click).
2. Briefly click the right mouse button.
3. Left-click the appropriate menu item.

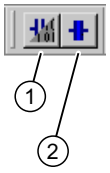
Toolbars

The dynamic toolbars contain icons for important, frequently used functions, e.g. for inserting or saving elements.

The "dynamic toolbar" changes depending on which workspace is active/selected, e.g. MCC chart, ST program or LAD/FBD program.

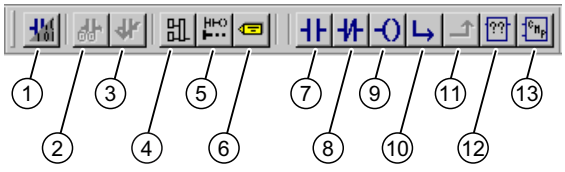
The toolbars can be positioned as required within the Workbench. Once moved, they can be shown or hidden using **View > Toolbars**.

The LAD/FBD editor toolbar contains the full range of LAD/FBD commands. The command list is displayed whenever the workspace for a program is active or open.



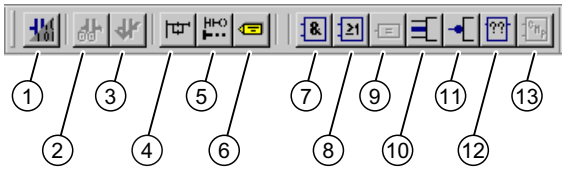
1) Accept and compile
2) Insert LAD/FBD program

Figure 7-443 Picture of the toolbar for a LAD/FBD unit



1) Accept and compile
2) Program status
3) Symbol check and type update
4) Switch to FBD
5) Insert network
6) Jump label ON/OFF
7) Insert NO contact
8) Insert NC contact
9) Insert coil
10) Open branch
11) Close branch
12) Insert comparator
13) Insert an empty box

Figure 7-444 View of the LAD editor toolbar



1) Accept and compile
2) Program status
3) Symbol check and type update
4) Switch to LAD
5) Insert network
6) Jump label ON/OFF
7) Insert AND box
8) Insert OR box
9) Insert assignment
10) Binary input
11) Negate binary input
12) Insert comparator
13) Insert an empty box

Figure 7-445 View of the FBD editor toolbar

Key combinations

Use the key combinations for fast operation in the LAD/FBD editor.

Within an LAD/FBD network, you can switch between LAD/FBD elements, select the required input/output of an LAD/FBD element and switch to the adjacent LAD/FBD network using the **left/right arrow buttons** and the **up/down arrow buttons**. Using the **Return** key, you can open the input field of a selected input/output and make an entry as well as close the field again using the **Return** key.

The shortcuts (Page 5408)available in the LAD/FBD editor are listed in the Appendix.

Drag&Drop of variables

Variables are dragged from the detail view (**Symbol browser**, **Watch table** or **Address list** tab) and dropped in the input field.

To paste in variables using drag&drop, proceed as follows:

1. Left-click the line number of the variable you wish to move.
   The line with the variables is highlighted.
2. Keeping the left mouse button pressed, drag the line number into the input field of the parameter screen form.
3. Release the left mouse button. The variable is pasted in at the selected position.

Drag&drop from the declaration tables

Variable names can be dragged from a declaration table and dropped into an LAD/FBD network.

To paste in variable names using drag&drop, proceed as follows:

1. Left-click the line number with the name of the variable you wish to move.
   The line is shown on a black background.
2. Continue to press the left mouse button as you drag the variable name to any input field.
3. Release the left mouse button.
   The variable name is pasted in at the selected position.

Drag&drop within the declaration table

You can change the order of the variable declaration in the declaration table.

To change the order using drag&drop, proceed as follows:

1. Left-click the line number of the variable you wish to move.
   The line is shown on a black background.
2. Press the **Shift** key and continue to press the left mouse button as you drag the line to the desired position in the declaration table.
   A red line indicates the point of insertion.
3. Release the left mouse button.
   The line moves to the corresponding position.

Note

To move several adjacent lines together, hold the **Shift** key down as you select the lines you wish to move.

Using Drag&Drop for LAD/FBD elements

LAD/FBD elements can be pasted into the LAD/FBD network from the project navigator (**Command library** tab) using drag-and-drop.

To paste in LAD/FBD elements using drag&drop, proceed as follows:

1. Left-click the required LAD/FBD element.
2. Hold the left mouse button down and drag the LAD/FBD element into the ladder diagram line of the LAD/FBD network.
3. Release the left mouse button.
   The LAD/FBD element is pasted in at the selected position.

Command call drag&drop

The commands in the command library can be inserted into LAD/FBD programs.

To insert command calls using drag&drop, proceed as follows:

1. Left-click the required command call.
2. Continue to hold the left mouse button down as you drag the command call to the LAD/FBD program.
3. Release the left mouse button.
   The command call is inserted at the selected position.

Drag&Drop of command names

Command names can be moved using drag-and-drop from the project navigator (tab **Command library**) into the input field of an empty box that has already been generated.

To paste in **command names** using drag&drop, proceed as follows:

1. Left-click the required command name.
2. Holding the left mouse button down, drag the command name into the input field of an empty box.
3. Release the left mouse button.
   The command name is pasted in at the selected position.

Using drag&drop for elements in a network

To insert elements in a network using drag&drop, proceed as follows:

1. Left-click the required LAD element.
2. To move an element, proceed as follows:
   Holding the left mouse button down, drag the element to the required position in the ladder diagram line.
3. To copy an element, proceed as follows:
   Keeping the **CTRL** key depressed, drag the element with the left mouse button to the required position in the ladder diagram line.
4. Release the left mouse button.
   The LAD element is inserted at the selected position.

Using drag&drop for functions and function blocks from other sources

Successfully compiled functions and function blocks from other source files can be pasted into a ladder diagram line from the project navigator. The connection to the "original source" is automatically entered in the **Connections** tab of the current source file.

To paste in functions and function blocks using drag&drop, proceed as follows:

1. Left-click the required FC/FB.
2. Holding the left mouse button down, drag the FC/FB into the input field of an empty box.
3. Release the left mouse button.
   An FC/FB call box is pasted in.

Automatic completion (Autocomplete)

In the LAD/FBD editor, you can automatically complete identifiers. A drop-down list box with identifiers that begin with the previously entered characters will be displayed. This operates on a context-sensitive basis, whereby the expected type for the identifier being sought and its visibility in the current program context determine which entries are displayed as options in the drop-down list box.

Depending on the context, the entries displayed in the drop-down list box are filtered and sorted:

- The filtering process may determine, for example, that only structure components are displayed for a structure.
- Entries are sorted according to their relevance to the context, with the more relevant identifiers appearing higher in the list (e.g. local variables are listed before global variables).

With LAD/FBD source files and LAD/FBD programs, the identifiers can be completed automatically in the following input fields/editable drop-down list boxes:

- Declaration table of the LAD/FBD unit
  - "Type" column
  - "Variable type" column
  - "Data type" column
- Declaration table of the LAD/FBD program
  - "Variable type" column
  - "Data type" column
- Input fields for the inputs of an LAD/FBD element (variables and other symbols for the expected type)
- Input fields for the outputs of an LAD/FBD element (variables for the expected type)
- Input field for the instance variable of a function block (variables for the expected FB type (box type))
- Input field for the NC contact, NO contact, and coil LAD elements (BOOL type variables)

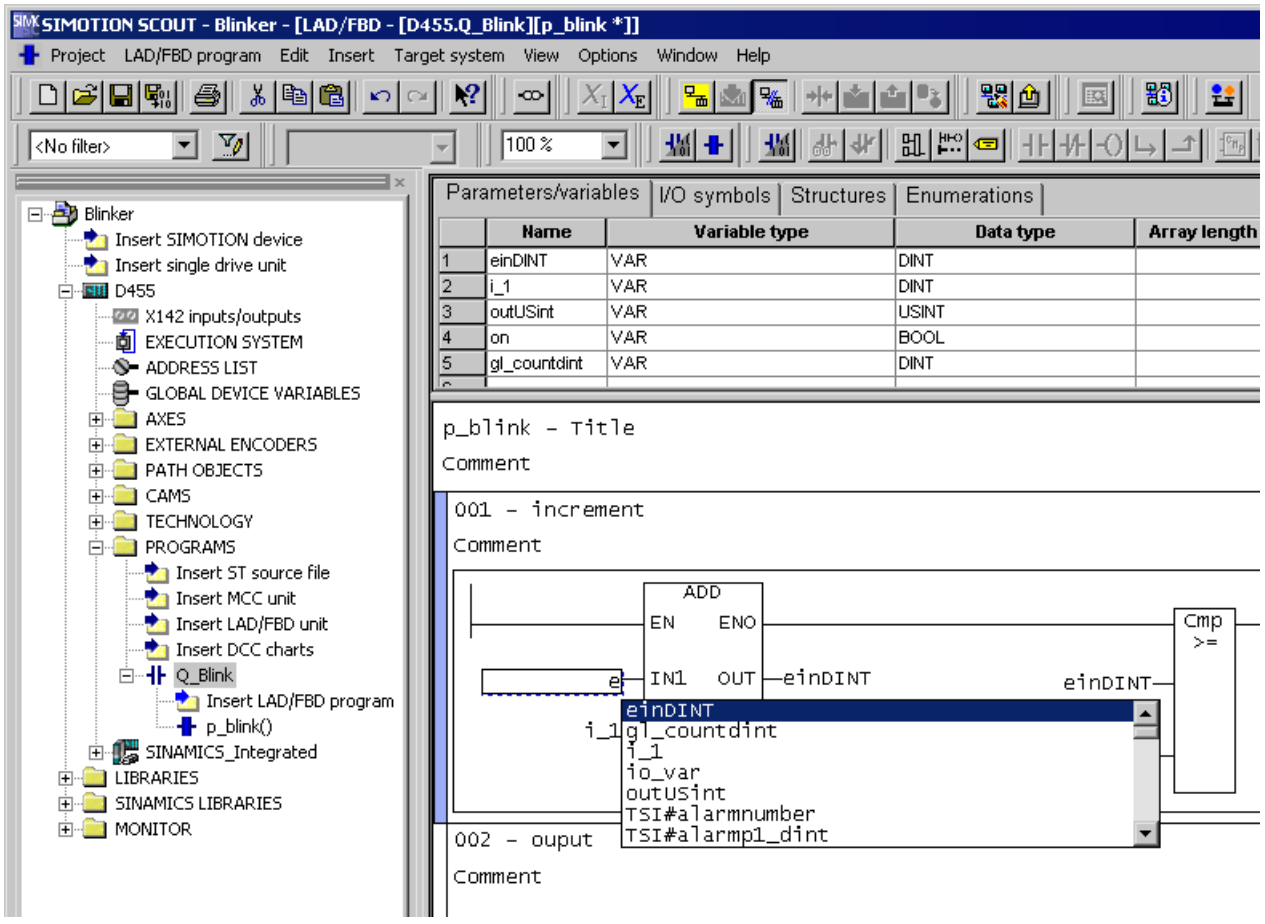


Figure 7-446 Automatic completion of an identifier at the input for an LAD/FBD element

Procedure

To complete an identifier automatically (Autocomplete), proceed as follows:

1. Write the first characters of the identifier (e.g. the first letters of a word) in the input field/ editable drop-down list box.
2. Press the **Ctrl+space** key combination.
   A drop-down list box opens containing the filtered/sorted identifiers for the current context. The identifiers listed start with the characters input so far or the characters up to where the cursor is positioned within an existing identifier.

   Note

   When the drop-down list box has been expanded and is editable (by clicking the symbol), automatic completion is already activated.

3. Expand or refine the options displayed:
   - Enter additional characters.
   - Delete characters.
   - Move the cursor with the left/right arrow keys.
4. Select the required identifier with the up/down arrow keys.
5. Press the **Return** key.
   The identifier is accepted by the input field/editable drop-down list box and the current word is overwritten.

   Note

   If only a single identifier is offered for selection, the selection window will not be opened and the identifier completed immediately.

Functional description

The following identifiers that begin with the specified characters will be offered:

- ST expressions
- ST program sections
- Identifiers from the command library
- Library, unit, POU, or system function names
- For technology objects including their system variables and configuration data

- Identifiers for the relevant LAD/FBD unit or LAD/FBD program:
  - Program organization units (POU) and FB instances
  - Data types
  - Variables and constants
  - Structure elements
- Identifiers from imported program sources

Note

Identifiers from the relevant LAD/FBD unit or LAD/FBD program or from imported program sources will only be displayed correctly if the corresponding program source has been compiled.

The display operates on a context-sensitive basis, i.e. only those types of identifiers that are appropriate at the associated location of the LAD/FBD unit or LAD/FBD program are offered:

- Within a declaration table, data types and variable types only
- Within a program organization unit (POU), no data types
- For a structure (e.g. var_struct.xx), only structure components

7.3.3.8 Settings

Settings in the LAD/FBD editor

You can define the layout for creating a program in the LAD/FBD programming language:

1. To change the settings, select the **Options > Settings** menu command.
   The **Settings** dialog box opens.
2. Select the tab **LAD/FBD editor**.

3. Make the required settings here.

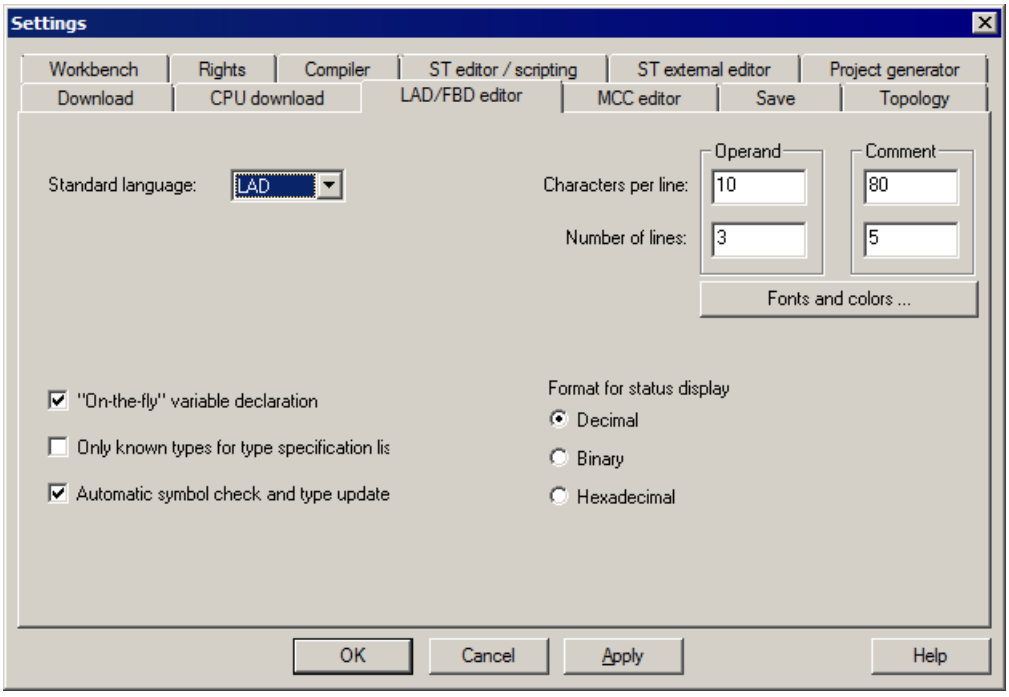


Figure 7-447 LAD/FBD editor settings

4. Click **OK** or **Accept** to confirm.

The table below contains a description of the individual parameters.

Table 7-523 LAD/FBD editor parameter settings

Parameter	Description
Default language	Defines the default graphical programming language for creating a new LAD/FBD program in the LAD/FBD editor.
"On-the-fly" variable declaration	If activated, a dialog box appears when an unknown symbol is entered in the LAD or FBD diagram. You can carry out the variable declaration in this dialog box. See: Defining variables in the Variable declaration dialog box ("on-the-fly" variable declaration) (Page 5175)
Only known types if type lists exist	If activated, only those function blocks which also have an entry in the Connections tab appear in the list of data types. See: Setting the data type list of the declaration table

Parameter	Description
Automatic symbol check and type update	If active, an automatic symbol check and type update take place once changes affecting the LAD/FBD program open in the LAD/FBD editor have been made in the project. The automatic symbol check and type update are activated by default in the LAD/FBD editor. See: Activating automatic symbol check and type update Example of a type update (Page 5096) Example of a symbol check (Page 5098) Deactivating automatic symbol check and type update Performing symbol check and type update at a specified time
Operand	You can change the options for displaying the operand by entering the number of **Characters per line** and **Number of lines** for the Operand field. Changes to settings are only applied to LAD/FDB editors that are opened afterwards.
Comment	You can change the options for displaying the comment by entering the number of **Characters per line** and **Number of lines** for the Comment field. Changes to settings are only applied to LAD/FDB editors that are opened afterwards.
Fonts and colors	You can change the fonts and colors of the LAD/FBD editor. See: Changing fonts Changing colors Changes to settings are only applied to LAD/FDB editors that are opened afterwards.
Format for status display	Format in which the values of variables with bit data type are displayed in program status and variable status. See: Starting and stopping the program execution monitoring (Page 5349) Variable status (Page 5345)

Activating automatic symbol check and type update

To enable automatic symbol check and type update, follow these steps:

1. Select the **Options > Settings** menu item.
2. Select the tab **LAD/FBD editor**.
3. Activate the checkbox **Automatic symbol check and type update**.
4. Confirm with **OK**.

Note

The automatic symbol check and type update is activated by default in the LAD/FBD editor.

If the symbol check is activated, the automatic symbol check and type update is performed for an LAD/FBD program if the following requirements are met:

- changes are made in the project (see list below) which impact upon the LAD/FBD program
- the focus is on the LAD/FBD program, i.e. it is opened in the KOP/FUP editor, or the LAD/FBD editor from the LAD/FBD program which is already open appears in the foreground (Page 5084)

In the event of subsequent changes within a project, automatic symbol check and type update is performed for an LAD/FBD program if the change affects the LAD/FBD program:

- A program source is changed, e.g. deleted or renamed.
  The changes are only identified for associated program sources and their LAD/FBD programs when the changed program source is compiled. In other words, to enable the automatic symbol check and type update to take place, the changed program source must be compiled before the focus changes to an LAD/FBD program from an associated program source.
- The declaration tables in the LAD/FBD source file and/or from LAD/FBD programs within the LAD/FBD source file are changed.
  The symbol check/type update takes place for an LAD/FBD program belonging to the LAD/FBD source file if the changes affect that LAD/FBD program.
  The following cases are possible:
  - A declaration table is changed within an LAD/FBD program.
    The automatic symbol check and type update only takes place after changing from the declaration table to the network.
  - The change within a declaration table takes place within an LAD/FBD program and affects another LAD/FBD program within the same program source.
    The automatic symbol check and type update takes place when the focus is on that other LAD/FBD program.
  - Changes within a declaration table take place within a program source and affect an LAD/FBD program in another program source.
    The automatic symbol check and type update for the LAD/FBD program from another program source can only take place once the changed program source has been compiled.
- A technology object (such as an axis) used by the LAD/FBD program is changed.
  The automatic symbol check and type update takes place when the focus is on the LAD/FBD program.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

If you click the mouse on the relevant place in the network (box parameter, input field, symbol highlighted in red) following a symbol check/type update, the tool tip indicates the following:

- the expected data type among the box parameters
- The data type of the variable with labeled input fields
- the cause of trouble in symbols which are highlighted in red whereby the symbol errors may have the following reasons:
  - The specified symbol does not exist
  - The specified symbol is not visible in the current context (incorrect or missing entry of the connections in the declaration table)
  - The specified variable does not have the appropriate type

Note

The symbol check is automatically updated as soon as the declaration table has been edited and left.
All errors, including errors in the declaration table, are displayed in the detail view.

Example of a type update

Introduction

During the type update all the data types used are updated:

- the list of permitted data types in an input field
- an LAD/FBD program's box type, i.e. in terms of the creation type (program, FB or FC) selected for the LAD/FBD program
- a box's interface in terms of the list of permitted data types per box parameter
- other data types (structures, enumerations, etc.)

For example, a structure AAA {a : BOOL} is used in an LAD/FBD program. If this structure is changed, e.g. AAA {a : BOOL, b : LREAL}, this change is applied by the LAD/FBD program during the type update.

Initial situation

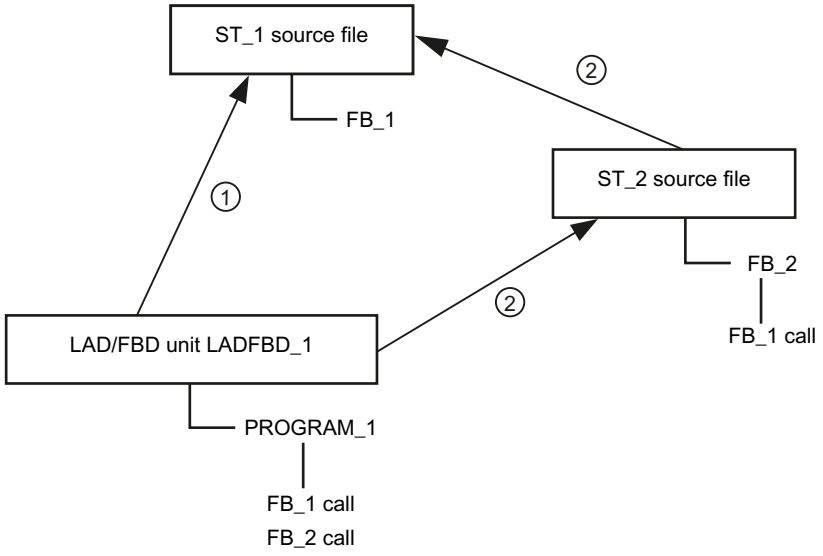
The following are present:

- an ST source ST_1 with a function block FB_1
- an ST source ST_2 with a function block FB_2
- an LAD/FBD unit LADFBD_1 with an LAD/FBD program PROGRAM_1
- FB_1 is called within ST_2, i.e. a connection (Page 5222) to ST_1 has to be declared in the ST_2 declaration table

- FB_1 is called within PROGRAM_1, i.e. a connection (Page 5222) to ST_1 has to be declared in the LADFBD_1 declaration table
- FB_2 is called within PROGRAM_1, i.e. a connection (Page 5222) to ST_2 has to be declared in the LADFBD_1 declaration table

How the type update works

This is a special scenario where a program source (in this case LADFBD_1) imports another program source (in this case ST_1) both explicitly and by means of inheritance. Inheritance takes place via another imported program source (in this case ST_2) which, in turn, imports other sources (in this case ST_1).



① LADFBD_1 explicitly imports ST_1

② LADFBD_1 imports ST_1 by means of inheritance (ST_1 → ST_2 → LADFBD_1)

Figure 7-448 Special scenario of explicit import and inheritance

Changes are now made at the interface of the FB_1, for example one or more input/output parameters are added (see also interface adjustment in FB/FC (Page 5252)) and ST_1 is recompiled.

When PROGRAM_1 is opened in the LAD/FBD editor, the FB_1 call is automatically updated, i.e. a type update occurs via which the changed interface is updated.

Despite the fact that the type update runs error-free, the compiler issues an error message during the compilation of LADFBD_1 because the import of ST_1 by inheritance via the imported ST_2 presupposes that ST_2 is also recompiled.

In order to recompile ST_2 without errors, the changed interface of the FB_1 call must be updated manually within ST_2.

Note

If program sources are imported which import other program sources themselves (inheritance), an error message may be output by the compiler during the compilation of the program source which is being imported, even if the type update runs error-free.

Example of a symbol check

Introduction

During the symbol check, the symbols used in the network are checked in context.

For example, the network includes a box with a BOOL-type input. An LREAL-type variable is now assigned to this input. The symbol check determines that this variable cannot be used in this context and, therefore, flags up this variable in red.

Initial situation

The following are present:

- an ST source ST_1 with the unit variable VAR_1
- an LAD/FBD unit LADFBD_1 with an LAD/FBD program PROGRAM_1
- the unit variable VAR_1 is used within PROGRAM_1, i.e. a connection (Page 5223) to ST_1 has to be declared in the LADFBD_1 declaration table

How the symbol check works

The unit variable VAR_1 is now changed, e.g. the name is changed.

If ST_1 is recompiled after this change, the unit variable VAR_1 is invalid in LADFBD_1.

When PROGRAM_1 is opened in the LAD/FBD editor, an automatic symbol check is performed, i.e. the changed variable is highlighted in red lettering.

The changed variable must be updated manually in PROGRAM_1.

Deactivating automatic symbol check and type update

The automatic update of the symbol check and type database should only be deactivated if computation speed is unduly reduced, e.g. if a large project is being processed on a slow computer and the hourglass is displayed after any change in the declaration table or in referenced external units.

To deactivate automatic symbol check and type update, proceed as follows:

1. Select the **Options > Settings** menu item.
2. Select the **LAD/FBD editor** tab.

3. Deactivate the **Automatic symbol check and type update** checkbox.
4. Confirm with **OK**.

Note

If the automatic symbol check is deactivated, the display may sometimes be inaccurate, e.g. after an external call has been inserted, the call box interface may be incorrectly displayed (i.e. not updated), or not displayed at all. This is because the current information only becomes available to the LAD/FBD editor after the "Symbol check and type update at a specified time" (Page 5099) has been called.

Perform symbol check and type update at a specified time

To perform a symbol check at a specified time, proceed as follows:

- Select the **LAD/FBD program > Symbol check and type update** menu item (shortcut Ctrl+T).
  - or -
  Click the **Symbol check and type update** icon.

Note

The symbol check at a specified time can only be performed when the "Automatic symbol check and type update" (Page 5094) is deactivated.

Setting the data type list of the declaration table

In the declaration area, all function blocks of the project not used in the program are stated in the list of data types by default.

To improve clarity, it is possible to set that only those function blocks are displayed for which an entry in the **Connections** tab exists.

To set the data type display, proceed as follows:

1. Select the **Options > Settings** menu item.
2. Select the tab **LAD/FBD editor**.
3. Click the **Only known types if type lists exist** check box.
4. Confirm with **OK**.

Changing fonts

Use the following procedure to change the font of the LAD/FBD editor:

1. Select the **Options > Settings** menu item.
2. Select the tab **LAD/FBD editor**.
3. Click the **Fonts and colors button**.
   The **Fonts and colors** dialog box with the **Fonts** tab appears.

4. Select the font, font size, type, or display you require.

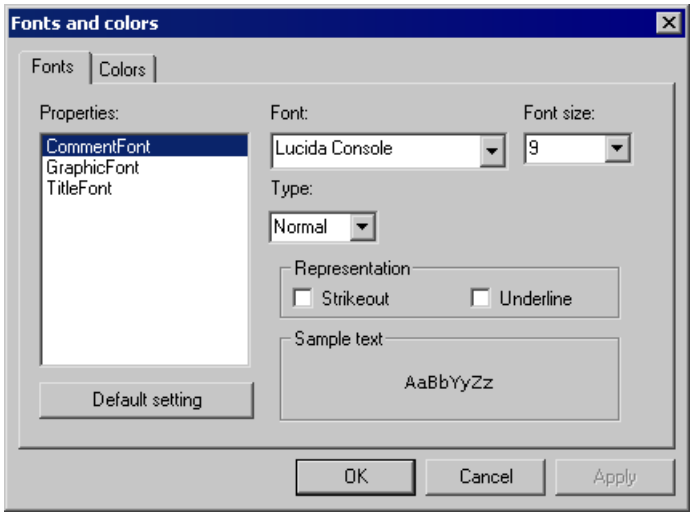


Figure 7-449 Fonts and colors dialog box

5. Confirm with **OK**.

Changes to settings are only applied to LAD/FDB editors that are opened afterwards.

Changing colors

Use the following procedure to change the colors of the LAD/FBD editor:

1. Select the **Options > Settings** menu item.
2. Select the tab **LAD/FBD editor**.
3. Click the **Fonts and colors button**.
   The **Fonts and colors** dialog box appears.
4. Click the **Colors** tab.

5. Choose a color.

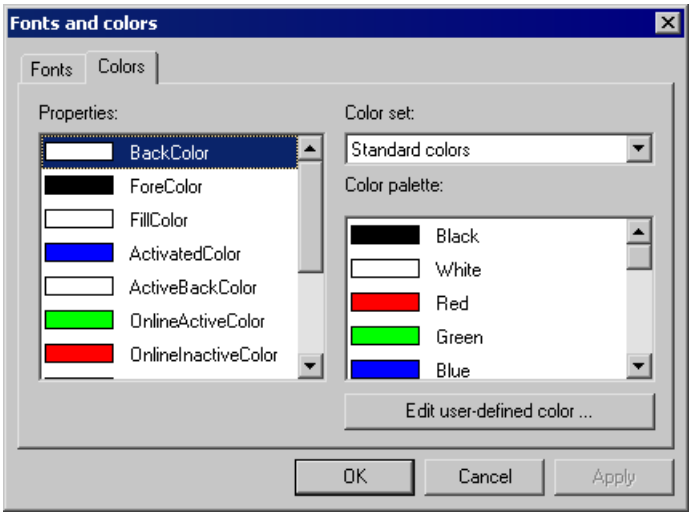


Figure 7-450 Fonts and colors dialog box

6. Confirm with **OK**.

Changes to settings are only applied to LAD/FDB editors that are opened afterwards.

Calling online help in the LAD/FBD editor

The online help can provide assistance for many of the operating steps. Call up the online help using either the:

- **Help** menu
  - Help topics
  - Context-sensitive help
  - Getting Started
- General help with the F1 key
- **Help** button, which appears in an open dialog box
- Context-sensitive help with the Shift+F1 key combination or the arrow with question mark icon (also for LAD/FBD elements in a network).

7.3.4 LAD/FBD programming

7.3.4.1 Programming software

This chapter describes the various operator control options in the LAD/FBD editor and the basic procedure for LAD/FBD programming.

7.3.4.2 Managing LAD/FBD source file

LAD/FBD units are assigned to the SIMOTION device on which the LAD/FBD programs contained in the unit will subsequently be run (e.g. SIMOTION D455-2). They are stored in the project navigator under the SIMOTION device in the PROGRAMS folder.

The individual program organization units (POU, LAD/FBD programs) are stored under a LAD/FBD unit.

Note

ST source files, MCC units and DCC charts are also stored in the **PROGRAMS** folder under the SIMOTION device.

For a description of the SIMOTION ST (Structured Text) programming language, refer to the SIMOTION ST Programming and Operating Manual.

For a description of the SIMOTION MCC (Motion Control Chart) programming language, refer to the SIMOTION MCC Programming and Operating Manual.

Inserting a new LAD/FBD source file

LAD/FBD units are assigned to the SIMOTION device on which the LAD/FBD programs contained in the unit will subsequently be run (e.g. SIMOTION D455-2).

There are several ways of inserting a new LAD/FBD unit.

- In the project navigator: in the **PROGRAMS** folder using the **Insert LAD/FBD unit** element
- Select the **PROGRAMS** folder in the project navigator and choose the command **Insert > Program > LAD/FBD unit** in the menu.
- Select the **PROGRAMS** folder in the project navigator and choose the command **Insert new object > LAD/FBD unit** in the context menu.

Procedure

To insert a new LAD/FBD unit using the context menu:

1. Open the appropriate SIMOTION device in the project navigator.
2. Select the **PROGRAMS** folder.
3. Select the **Insert new object > Insert LAD/FBD unit** context menu.

4. Enter the name of the LAD/FBD unit.
   The names of program sources must comply with the rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters.
   The permissible length of the name depends on the SIMOTION Kernel version:
   - SIMOTION Kernel as of version V4.1: a maximum of 128 characters.
   - SIMOTION Kernel up to version V4.0: a maximum of 8 characters.

   Names must be unique within the SIMOTION device. Protected or reserved identifiers (Page 5410) are not permitted.
   Existing program sources (e.g. LAD/FBD units, ST source files) are displayed.
5. You can also enter an author, version, and a comment.
6. Activate the **Open editor automatically** checkbox.
7. If necessary, select the **Compiler** tab and make any local compiler settings; see Local compiler settings (Page 5112).
8. Confirm with **OK**.

Note

When you click **OK**, the LAD/FBD unit is transferred to the project only. The data, together with the project, is only saved to the data carrier if you select, for example, **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

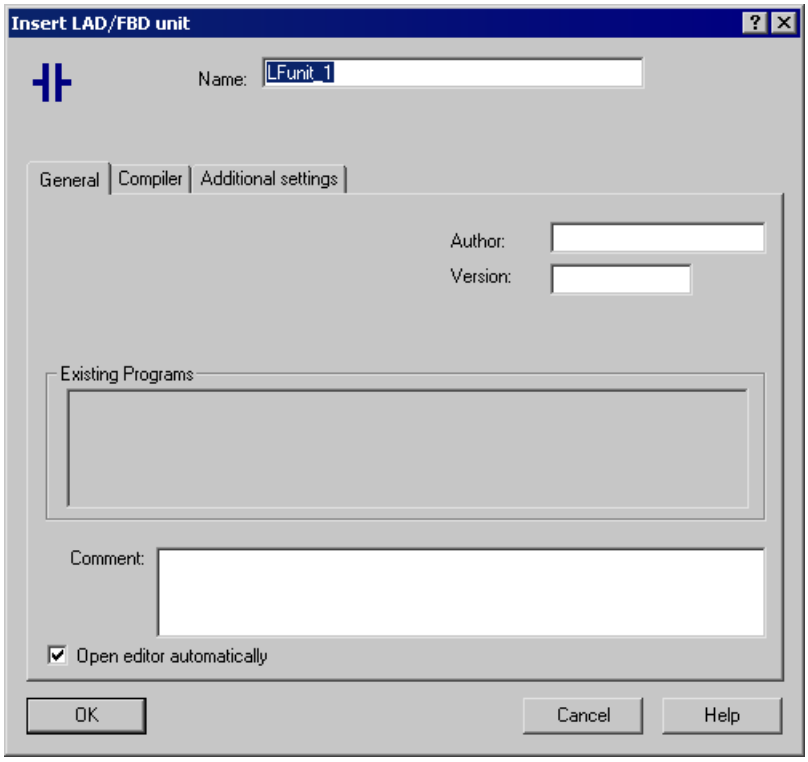


Figure 7-451 Insert LAD/FBD Unit dialog box

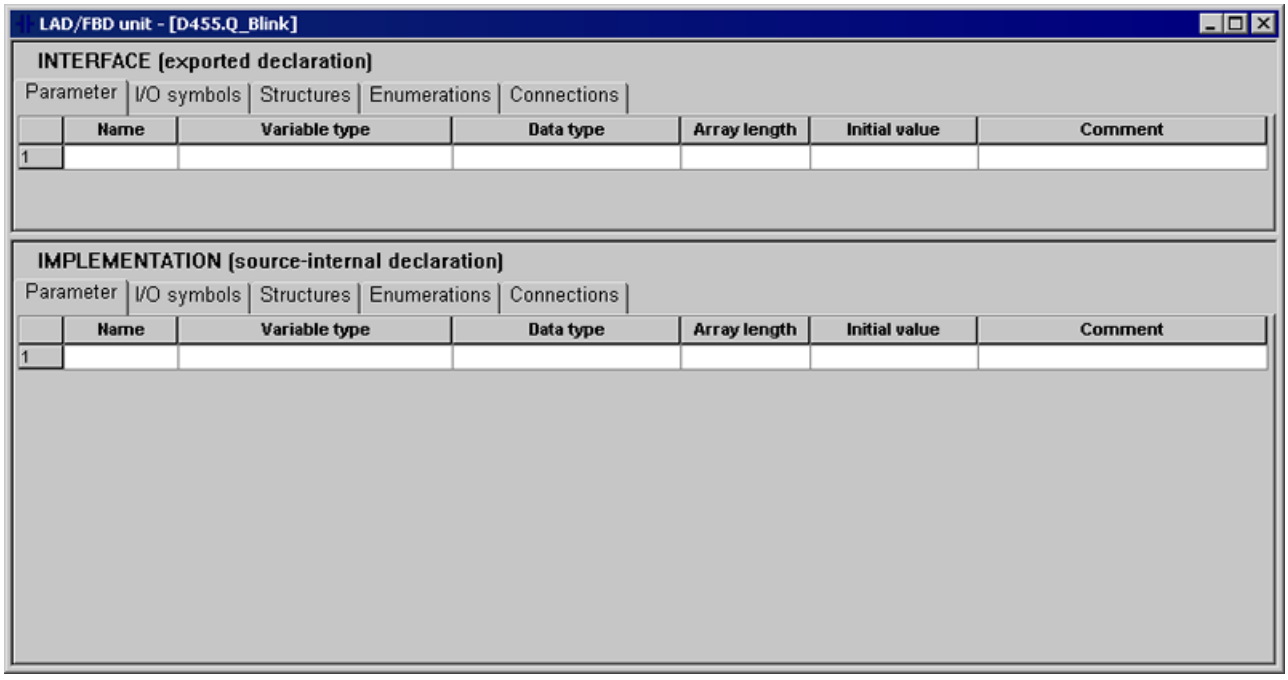


Figure 7-452 New LAD/FBD unit (declaration table for the interface and implementation sections)

Opening an existing LAD/FBD source file

Procedure

To open an existing LAD/FBD unit, proceed as follows:

1. Open the subtree of the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Select the required LAD/FBD unit.
4. Select the **Open** context menu.
5. Only for LAD/FBD units with know-how protection (Page 5106):
   If the LAD/FBD unit (or a program of this unit) is not already open and the login assigned to the LAD/FBD unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (until the unit and all its programs are closed).
   - If required, activate the "Use login as a standard login" checkbox.
     You will be logged in with this login and can now open additional units to which the same login is assigned without having to re-enter the password.

The LAD/FBD unit (declaration table) opens in the workspace. Multiple units can be opened.

Note

You can also double-click the required LAD/FBD unit to open it.

Saving and compiling a LAD/FBD source file

Requirements:

Ensure that the LAD/FBD unit or one of the associated LAD/FBD programs is the active window in the workbench.

To save a LAD/FBD unit and all its associated LAD/FBD programs in the project and start the compiler:

- Select the **Save and compile** icon in the LAD/FBD editor toolbar.
  - or -
  Select the **LAD/FBD unit > Save and compile** menu item.
  - or -
  Select the LAD/FBD unit or a LAD/FBD program in the project navigator and select **Save and compile** in the context menu.
  - or -
  Shortcut **Ctrl+B**.

Note

Save and compile only applies the changes to LAD/FBD units and associated LAD/FBD programs in the project. The data is only saved to the data carrier, together with the project, if you select **Project > Save** or **Project > Save and compile changes**.

A LAD/FBD unit can also be saved outside the project (exported).

Error messages and warnings relating to compilation are displayed in the **Compile/check output** tab in the detail view.

Closing a LAD/FBD source file

To close a LAD/FBD unit opened in the working window, proceed as follows:

1. Click the **x** button (cross) in the title bar of the dialog box of the LAD/FBD unit.
   - or -
   Select the **LAD/FBD unit > Close** menu item.
   - or -
   Select **Windows > Close all windows** menu item.
   - or -
   Shortcut **Ctrl+F4**.
   If the changes have not yet been saved in the project, you can save or cancel them, or abort the close operation.

Cut/copy/delete operations in a LAD/FBD source file

A LAD/FBD source file can be cut or copied together with all its associated LAD/FBD programs and pasted into the same or another SIMOTION device.

It is not possible to paste in a LAD/FBD source file that has been deleted.

To **cut**, **copy** or **delete**, proceed as follows:

1. In the project navigator, select the required LAD/FBD source file.
2. In the context menu, select the appropriate item (**Cut**, **Copy**, or **Delete**).
3. Change the name, if necessary (refer to "See also").

See also

Renaming a LAD/FBD source file (Page 5110)

Inserting a cut or copied LAD/FBD source file

To paste in a cut or copied LAD/FBD source file:

1. Under the SIMOTION device, select the **PROGRAMS** folder.
2. In the context menu, select **Paste**.
   The LAD/FBD source file is pasted in under a new name.
3. If required, amend the name.

Know-how protection for LAD/FBD source files

You can protect LAD/FBD units against being accessed by unauthorized third parties. Protected LAD/FBD units and all associated LAD/FBD programs can only be opened or exported in EXP format when the password is entered.

The SIMOTION online help provides additional information on know-how protection.

Note

If you export in XML format, the LAD/FBD units are exported in an encrypted format. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

7.3.4.3 Exporting and importing LAD/FBD source files

The export and import functions offer you the option of saving a LAD/FBD source file outside the project on your hard disk so that you can copy it from there into another project.

Exporting a LAD/FBD source file in XML format

You can use an XML export to save an LAD/FBD unit in a directory outside the project, independently of any particular version or platform.

To export a LAD/FBD unit in XML format:

1. In the project navigator, select the required LAD/FBD unit.
2. Select the **Expert > Save project and export object** context menu or the **Project > Save and export** menu.
3. Select the directory for the XML export and confirm with **OK**.

Note

Structures (e.g. several POUs in one unit, advance binary switching) can be used with SIMOTION Kernel as of version V4.1. These structures may not be supported by previous versions.

Note

LAD/FBD units with know-how protection can also be exported in XML format. The LAD/FBD units are exported in encrypted format. When importing the encrypted XML files, the know-how protection, including login and password, is retained.

Importing LAD/FBD source files as XML data

To import a LAD/FBD source file in XML format:

1. In the project navigator, select the **PROGRAMS** entry or a LAD/FBD source file.
2. In the context menu, select **Import object** or **Expert > Import object**.
3. Select the XML data to be imported and click **OK** to confirm.
   The LAD/FBD source file is inserted.

Exporting a POU in XML format

Note

Structures (e.g. several POUs in one unit, advance binary switching) can be used with SIMOTION Kernel as of version V4.1. These structures may not be supported by previous versions.

You can use an XML export to save individual program organization units in a directory outside the project, independently of any particular version or platform.

To export a POU in XML format, proceed as follows:

1. In the project navigator in the LAD/FBD unit, select the POU you want to export.
2. In the context menu, select **Export as XML**.

3. Only for POUs in LAD/FBD units with know-how protection and which are not already open:
   If the associated LAD/FBD unit (or a POU of this unit) is not already open and the login assigned to the LAD/FBD unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this chart is temporarily canceled (for this export).
   - If required, activate the **Use login as a standard login** checkbox.
     You will be logged in with this login and can now export or open additional units to which the same login is assigned without having to re-enter the password.
4. Select the directory for the XML export and confirm with **OK**.
5. Enter the path and file name for the XML export and click **Save** to confirm.

The POU is saved as XML format; the file name is given the default extension *.xml

Note

A POU with know-how protection is exported without protection.

Importing a POU from XML format

To import a POU in XML format, proceed as follows:

1. In the project navigator, select the **PROGRAMS** entry or a LAD/FBD unit.
2. In the context menu, select **Import object**.
3. Select the XML data to be imported and click **OK** to confirm.
   The POU is inserted.

Exporting a LAD/FBD source file in EXP format

With the export in the EXP format, you can save an LAD/FBD unit in a format that can also be interpreted by third-party controllers.

To export an LAD/FBD unit in EXP format, proceed as follows:

1. Select the LAD/FBD unit in the project navigator.
2. Select the context menu **Expert > Export as .EXP**.
3. Only for LAD/FBD units with know-how protection (Page 5106) and which are not already open:
   If the LAD/FBD unit (or a program of this unit) is not already open and the login assigned to the LAD/FBD unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (for this export).
   - If required, activate the **Use login as a standard login** checkbox.
     You will be logged in with this login and can now export or open additional units to which the same login is assigned without having to re-enter the password.
4. Enter the path and file name for the EXP export and click **Save** to confirm.

The LAD/FBD unit is saved in EXP format and given the file extension *.exp by default.

Note

The export of an LAD/FBD unit in EXP format and subsequent import of the EXP data can change the structure of the networks in the LAD/FBD programs. The compilation result remains unchanged.

An LAD/FBD unit with know-how protection is exported without protection.

Importing EXP data into a LAD/FBD source file

With the import of EXP data, you can create an LAD/FBD unit from third-party programs that are available EXP format.

To import EXP data into a LAD/FBD unit:

1. Select the LAD/FBD unit in the project navigator.
2. Select the context menu **Expert > Import from .EXP**.
3. Select the EXP file to be imported.
4. Select the program sources to which Connections (Page 5222) care to be created.
5. Confirm with **OK**.

The EXP file is imported into the LAD/FBD unit and the corresponding LAD/FBD programs created. The connections to the selected program sources are also created.

Note

Note the following when importing EXP data:

- It is possible to import from XOR to FBD.
- Preconnection with simple data types (signal connection) is not generally supported.
- The original structure is retained when you import data from EXP files. If the structure cannot be compiled due to type conflicts, the relevant parameters are highlighted in red.
  A type conflict can be resolved by manual revision.

7.3.4.4 LAD/FBD source files - defining properties

Defining the properties of a LAD/FBD source file

Procedure

1. Under the SIMOTION device, open the **PROGRAMS** folder.
2. Select the required LAD/FBD unit.

3. Select the **Edit > Object Properties** menu command.
4. If necessary, select further tabs to make local settings (only valid for this LAD/FBD unit):
   - **General** tab: General details for the LAD/FBD unit, e.g. timestamp of the last change and the storage location of the project (see figure).
   - **Compiler** tab: Local settings of the compiler (Page 5112) for code generation and message display.
   - **Additional settings** tab: Display of the compiler options in accordance with the current compiler settings (see the SIMOTION ST Programming and Operating Manual).
   - **Compilation** tab: Display of the compiler options during the last compilation of the LAD/FBD unit (see the SIMOTION ST Programming and Operating Manual).
   - **Object address** tab: Set the internal object address of the LAD/FBD unit. The object addresses of the other program sources are displayed.

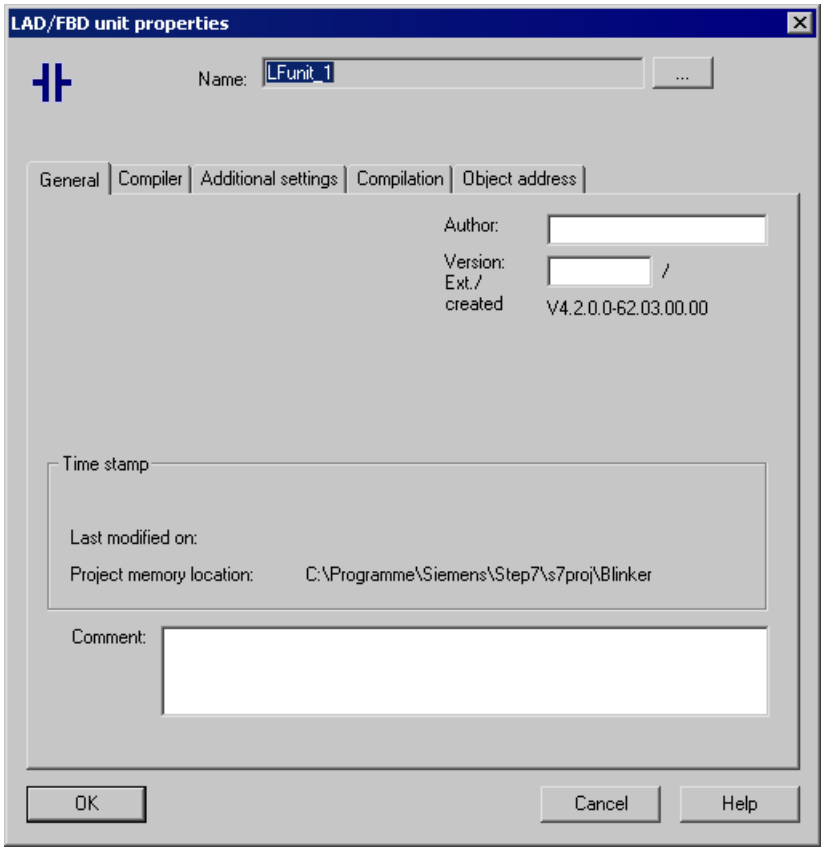


Figure 7-453 Properties of a LAD/FBD source file

Renaming a LAD/FBD source file

To rename a LAD/FBD source file:

1. Open the Properties window of the LAD/FBD source file.
2. Click ... .

3. Confirm the message with **OK** and enter the new name in the **New name** input field of the **Change Name** dialog box.
4. Acknowledge the entries with **Apply**.

Making settings for the compiler

You can define the compiler settings as follows:

- globally for the SIMOTION project, always applicable to all programming languages, see Global compiler settings (Page 5111)
- locally for an individual LAD/FBD source within the SIMOTION project, see Local compiler settings (Page 5112)

Global compiler settings

The global settings are always valid for all programming languages within the SIMOTION project. If there are global settings which only apply to specific programming languages, this is specified on the **Compiler** tab.

Procedure

1. Select the menu **Options > Settings**.
2. Select the **Compiler** tab.
3. Make the settings in accordance with the parameter description in the SIMOTION ST Programming and Operating Manual.
4. Confirm with **OK**.

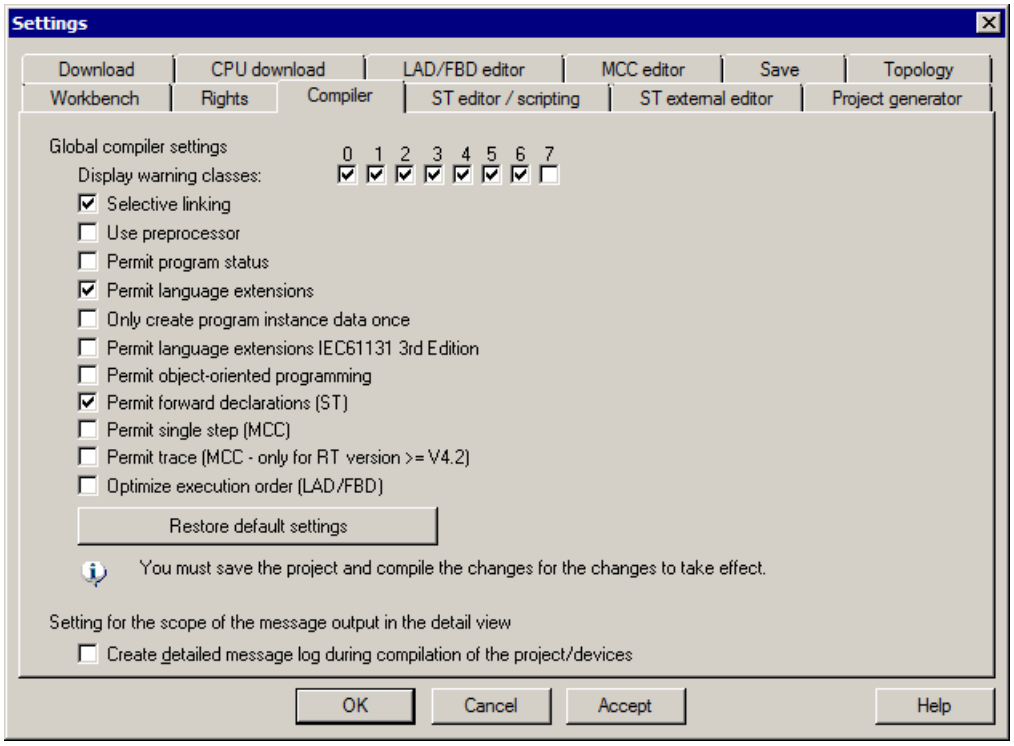


Figure 7-454 Global compiler settings

Parameter

For a description of the parameters of the global compiler settings, refer to SIMOTION ST Programming and Operating Manual.

Local compiler settings

Local settings are configured individually for each LAD/FBD unit; local settings overwrite global settings.

Procedure

To select the compiler options, proceed as follows:

1. Open the Properties window for the LAD/FBD unit (see Defining the properties of an LAD/FBD unit (Page 5109)).
2. Select the **Compiler** tab.
3. Define the settings according to the following table.
4. Click **OK** to confirm.

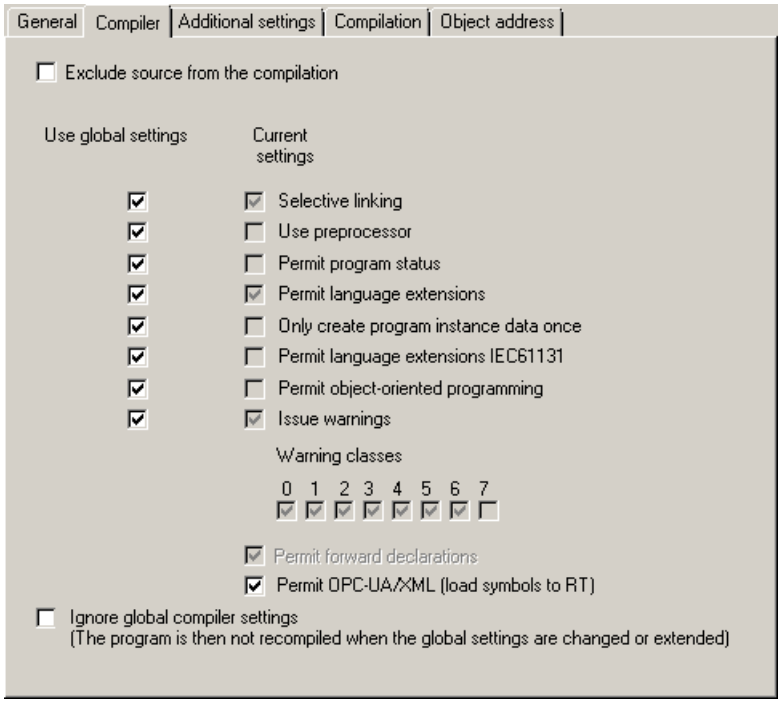


Figure 7-455 Local compiler settings for LAD/FBD units in the Properties window

The current compiler options (the combination of global and local compiler settings which currently applies) for the program source are displayed on the **Additional settings** tab. The compiler options used the last time the program source was compiled can be seen on the **Compilation** tab.

The SIMOTION ST Programming and Operating Manual contains further information on what the compiler options mean.

Table 7-524 Local compiler settings

Parameter	Description
Exclude the source from the compilation	**Active**: This source is not compiled upon compilation of the project, the device or the library (e.g. Menu **Project > Save and recompile all**). The source is marked accordingly in the project navigator. Corresponding information is provided upon compilation. **Inactive** (standard): The source is compiled upon compilation of the project, the device or the library.
Use global settings	This checkbox is available for every parameter which also has a global setting. This is where you define whether the global settings are adopted or whether the local settings will apply. See the description under Effectiveness of global or local settings in the SIMOTION ST Programming and Operating Manual. Use the second checkbox or the other checkboxes for the relevant parameters (described below) to define the local settings.
Selective linking[1]	**Active**: Unused code is removed from the executable program. **Inactive**: Unused code is retained in the executable program. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).

Parameter	Description
Use preprocessor[1]	**Active**: Preprocessor is used, see SIMOTION ST Programming and Operating Manual. **Inactive**: Preprocessor is not used. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).
Enable program status[1]	**Active**: Additional program code is generated to enable monitoring of program variables (including local variables). **Inactive**: Program status not possible. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Program status (Page 5348).
Permit language extensions[1]	**Active**: Language elements are permitted that do not comply with IEC 61131-3. • Direct bit access to variables of a bit data type, see the SIMOTION ST Programming and Operating Manual. • Accessing the input parameter of a function block when outside the function block, see the SIMOTION ST Programming and Operating Manual. • Calling a program while in a different program, see the SIMOTION ST Programming and Operating Manual. **Inactive**: Only language elements are permitted that comply with IEC 61131-3. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active).
Only create program instance data once[1]	**Active**: The local variables of a program are only stored once in the user memory of the unit. This setting is required when calling a program while in a different program, see the SIMOTION ST Programming and Operating Manual. **Inactive**: The local variables of a program are stored according to the task assignment in the user memory of the associated task. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See Memory areas for the variable types in the SIMOTION ST Programming and Operating Manual. For further information, refer to the SIMOTION Basic Functions Function Manual.

Parameter	Description
Permit language extensions, IEC61131 3rd edition[1]	**Active**: Additional language elements can be used in accordance with IEC 61131-3 3rd edition. The corresponding keywords are locked as reserved or protected identifiers. • Nested block comments (ST) • CONTINUE statement (ST) • Standard-compliant system functions LOWER_BOUND and UPPER_BOUND for determining the limits of a dynamic ARRAY • Additional system functions: – FROM_BIG_ENDIAN – FROM_LITTLE_ENDIAN – IS_VALID – TO_BIG_ENDIAN – TO_LITTLE_ENDIAN **Inactive**: The additional language elements cannot be used. The corresponding keywords are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.
Permit object-oriented programming[1]	**Active**: The keywords for object-oriented programming according to IEC 61131-3 3rd edition are locked as reserved or protected identifiers. General references (Page 5194) can also be formed. **Inactive** (standard): The keywords for object-oriented programming are not locked. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). **Note** Independently of the setting the program organization units created in ST sources using object-oriented programming (e.g. classes, methods) can be used in this source. No program organization units can be created in LAB/FBD using object-oriented programming. When saving the project in the old project format: Projects in which language elements that require this compiler option are used cannot be compiled correctly in SIMOTION SCOUT versions earlier than V4.5.

Parameter	Description
Issue warnings Warning classes[1]	In addition to error messages, the compiler can issue warnings and information. You can set the scope of the warning messages to be issued. "Issue warnings" checkbox: **Active**: The compiler issues the warnings and information according to the warning class selection that follows. **Inactive**: The compiler suppresses all warnings and information concerning this source file. The checkboxes for the warning classes are hidden. **Gray background** (display only): Operating on a global setting basis, the compiler always issues warnings and information in accordance with the global warning class selection shown below (if "Use global settings" = active). "Warning classes" checkboxes (only if "Issue warnings" = active): **Active**: The compiler issues warnings and information for the selected class. **Inactive**: The compiler suppresses warnings and information for the associated class. **Gray background** (display only): The global setting displayed is adopted (when "Use global settings" = active). See also Meaning of the warning classes in the SIMOTION ST Programming and Operating Manual.
Permit forward declarations	Forward declarations enable you to use program organization units (POUs) before they are completely defined. See the SIMOTION ST Programming and Operating Manual. This setting is always **active**. Forward declarations are always permitted for the LAD/FBD programming language.
Permit OPC-UA / -XML (load symbols to RT)	**Active:** Symbol information for the unit variables of the LAD/FBD unit is available in the SIMOTION device. This is required for: • The *_exportUnitDataSet* and *_importUnitDataSet* functions; see the SIMOTION Basic Functions Function Manual • The watch function of IT DIAG **Inactive:** Symbol information is not created.
Exclude the source from the compilation	**Active**: This source is not compiled upon compilation of the project, the device or the library (e.g. Menu **Project > Save and recompile all**). The source is marked accordingly in the project navigator. Corresponding information is provided upon compilation. **Inactive** (standard): The source is compiled upon compilation of the project, the device or the library.
Ignore global compiler settings (The program is not compiled again if the global settings are changed or extended.)	**Active:** The global settings of the compiler are ineffective for all parameters. The "Use global settings" checkboxes cannot be selected and are grayed out. When changing the global compiler settings, the LAD/FBD unit is not recompiled. **Inactive:** The checkboxes "Use Global Settings" can be selected for all parameters and are presented against a white background. These checkboxes specify whether the global properties are taken over for the corresponding parameters. See the description under Effectiveness of global or local settings in the SIMOTION ST Programming and Operating Manual.

[1] Global setting is also possible (**Options > Settings > Compiler** menu), see Global compiler settings (Page 5111). See also the description on Effectiveness of global or local compiler settings in the SIMOTION ST Programming and Operating Manual.

7.3.4.5 Managing LAD/FBD programs

LAD/FBD programs are the individual program organization units (program, function, function block) in a LAD/FBD source file. They are stored under the LAD/FBD source file in the project navigator.

Inserting a new LAD/FBD program

You can insert a new LAD/FBD program as program organization unit (POU) for an existing LAD/FBD source as follows (see Inserting a new LAD/FBD source (Page 5102)):

- In the project navigator: Below an LAD/FBD unit using the element **Insert LAD/FBD program**
- Select the required LAD/FBD unit in the project navigator followed by **Insert > Program > LAD/FBD program**
- Open the required LAD/FBD unit and select the **Insert LAD/FBD program** icon in the **LAD/FBD unit** toolbar

Procedure

To insert a new LAD/FBD program, proceed as follows:

1. Open the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder and a LAD/FBD unit.
3. Double-click the entry **Insert LAD/FBD program**.
4. Enter the name of the program in the **Insert LAD/FBD Program** dialog box.
   Names for LAD/FBD programs must comply with the Rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper and lower case letters. Protected or reserved identifiers (Page 5410) are not allowed. The permissible length of the name is 25 characters.
   The names must be unique within the LAD/FBD source. The names of all exportable program organization units (POUs) must also be unique within the SIMOTION device. The names of all LAD/FBD programs of the program source as well as the names of all exportable POUs of the device are displayed.
5. Optionally enter the identifier of an object-oriented namespace.
   Identifiers for namespaces must comply with the Rules for identifiers (Page 5155) (see above).
   When an identifier is entered, the MCC chart is assigned to the specified namespace.
   The compiler option (Page 5111) "Permit object-oriented programming" must be activated.
   For further information on object-oriented namespaces, see the appropriate section in the SIMOTION ST Programming and Operating Manual.
6. Select Program, Function, or Function block as the **Creation type**. See also Changing the LAD/FBD program creation type (Page 5124).
7. For the Function creation type only:
   Select Return value data type as the Return type
   (<--> for no return value).

8. Activate the **Exportable** checkbox if you want the LAD/FBD program to also be available in other program sources (LAD/FBD units, MCC units, or ST source files) or in the execution system.
9. You can also enter an author, version, and a comment.
10. Activate the **Open editor automatically** checkbox.
11. Click **OK** to confirm.

Note

When you click **OK**, the LAD/FBD program is transferred to the project only. The data is only saved to the data carrier, together with the project, if you select, for example, **Project > Save**, **Project > Save and compile changes**, or **Project > Save and recompile all**.

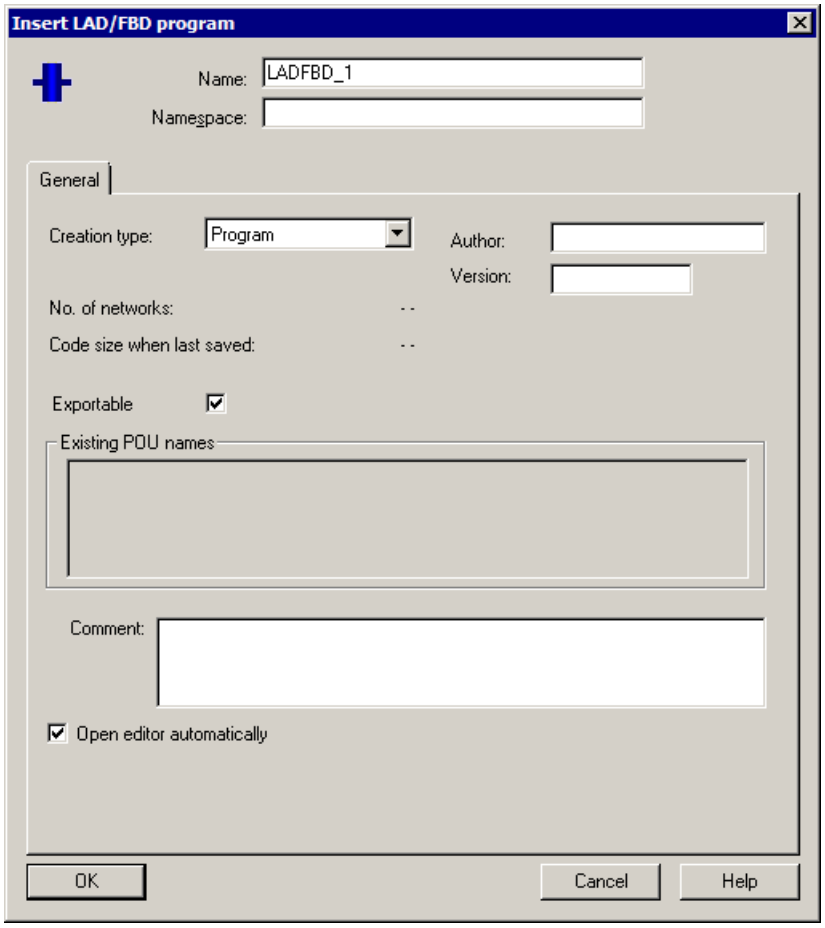
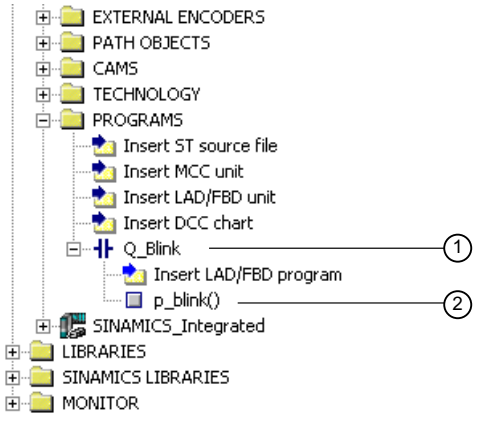


Figure 7-456 Insert LAD/FBD Program dialog box

Note

Existing POE names are only displayed if the check box "Show existing POE names (insert dialog POE for LAD/FBD and MCC)" is set in the Settings > Workbench. If there are a large number of POEs in the project, it is recommended to deselect this option, as the computer memory for the process may not be sufficient to refresh the display.



① Name of the LAD/FBD unit

② Name of the LAD/FBD program

Figure 7-457 Displaying the unit and program name in the project navigator

Opening an existing LAD/FBD program

All LAD/FBD programs belonging to an LAD/FBD unit are located in the project navigator underneath the LAD/FBD unit.

Procedure

To open an available program, proceed as follows:

1. Open the subtree of the appropriate SIMOTION device in the project navigator.
2. Open the **PROGRAMS** folder.
3. Open the LAD/FBD unit containing the required LAD/FBD program.
4. Select the required LAD/FBD program.

5. Select the **Open** context menu.
6. Only for programs below LAD/FBD units with know-how protection (Page 5106):
   If the LAD/FBD unit (or a program of this unit) is not already open and the login assigned to the LAD/FBD unit is not yet logged in:
   - Enter the corresponding password for the displayed login.
     The know-how protection for this unit is temporarily canceled (until the unit and all its programs are closed).
   - If required, activate the "Use login as a standard login" checkbox.
     You will be logged in with this login and can now open additional units to which the same login is assigned without having to re-enter the password.

The LAD/FBD program opens in the working area. Several LAD/FBD programs can be opened at the same time.

Note

You can also double-click the required LAD/FBD program to open it.

Defining the order of the LAD/FBD programs in the LAD/FBD source file

As of Version V4.2 of SIMOTION SCOUT, the user no longer needs to respect the POU order in the LAD/FBD unit, as this no longer has any role to play in terms of compilability. The user may wish to change the POU order so that the POUs are arranged in a more logical way from his or her own perspective.

Exception

When a project is saved in a format older than Version V4.2 of SIMOTION SCOUT, the order of the LAD/FBD programs in the LAD/FBD source file is of relevance as far as compilation is concerned. A subroutine (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the LAD/FBD program of the subroutine appears in the project navigator above the LAD/FBD program in which it is used. If necessary, reorder the charts (see Subroutine call of function (FC) (Page 5235)).

Prior to saving a project in a project format earlier than version 4.2, you can test the project for downward compatibility (i.e. the correct POU order, etc.) via Project > Old project format > Test the project for downward compatibility.

Procedure

To change the order:

1. Select a LAD/FBD program in the project navigator.
2. In the context menu, select **Up / Down**

Copying the LAD/FBD program

To copy a LAD/FBD program:

1. In your LAD/FBD unit, select the POU you want to copy.
2. In the context menu, select **Copy**.
3. Select the LAD/FBD unit which is to be inserted in the POU.
4. In the context menu, select **Insert.**
   The LAD/FBD program is inserted.

Saving and compiling a LAD/FBD program

An asterisk is appended in the title bar of the project to the name of a program which has been modified but not yet saved.

Note

The entire unit and its POUs are saved and compiled

To save the LAD/FBD program and start the compilation:

1. Click the **Save and compile** icon in the LAD/FBD editor toolbar.
   - or -
   Select the **LAD/FBD program > Save and compile** menu item.
   - or -
   Select the LAD/FBD program in the project navigator and select **Save and compile** in the context menu.
   - or -
   Shortcut **Ctrl+B**.
   - or -
   If you want to save and compile all the available LAD/FBD programs, select the **Project > Save and compile changes** menu command.
   If any errors occur during compilation, the error locations are displayed in the **Detail view**.
2. To fix an error, double-click an error message in the detail view in the **Compile / check output** tab.
   The faulty element is selected and positioned in the window.

Note

Backward compatibility

This SCOUT program version supports structures (e.g. several POUs in one unit, advance binary switching) which may not be able to be processed by previous versions.

Closing a LAD/FBD program

To close a LAD/FBD program opened in the working window, proceed as follows:

1. Click the **x** button (cross) in the title bar of the dialog box of the LAD/FBD **program**.
   - or -
   Select the **LAD/FBD program > Close** menu item.
   - or -
   Select the **Windows > Close all** menu item.
   - or -
   Shortcut **Ctrl+F4**.
   If the changes have not yet been saved, you can save or cancel them, or abort the close operation.

Deleting the LAD/FBD program

To delete a LAD/FBD program:

1. In the project navigator, select the required LAD/FBD program.
2. In the context menu, select **Delete**.

Note

It is not possible to insert a LAD/FBD program that has been deleted.

7.3.4.6 LAD/FBD programs - defining properties

The properties of a LAD/FBD program are specified when it is inserted.

However, these properties can be viewed and modified by doing the following:

1. Open the **PROGRAMS** folder under the SIMOTION device in the project navigator.
2. Open the required LAD/FBD unit.
3. Select the required LAD/FBD program.
4. From the context menu, select **Properties**.
   The **LAD/FBD program properties** dialog box opens.

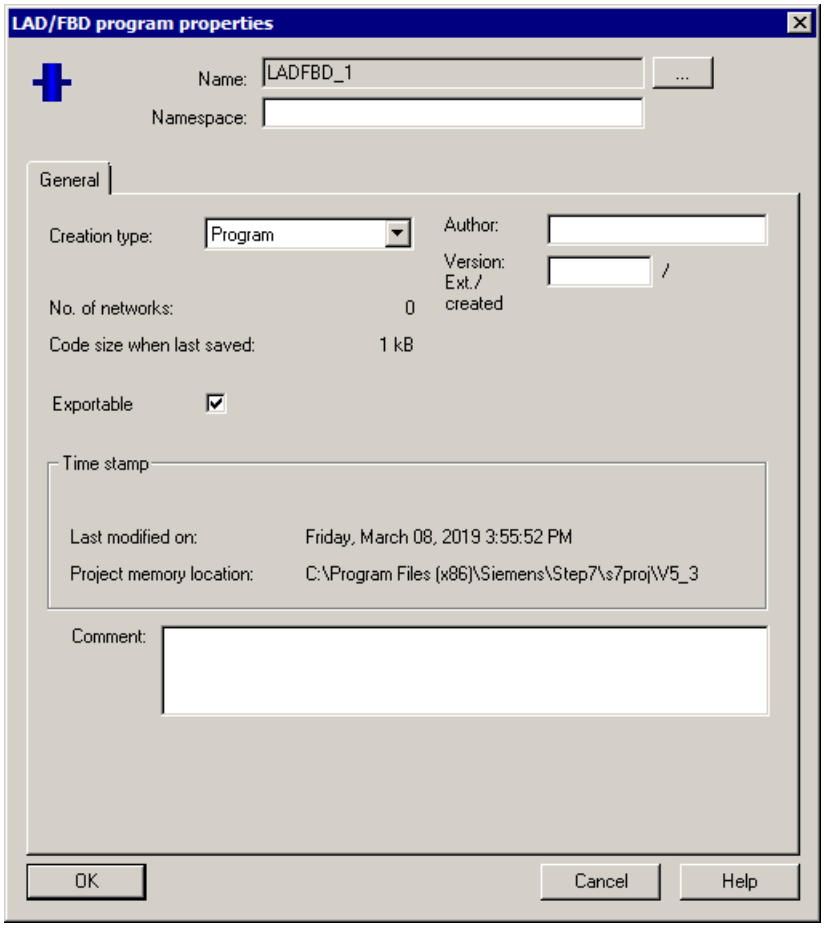


Figure 7-458 Properties of an LAD/FBD program

Renaming a LAD/FBD program

To rename a LAD/FBD program:

1. Open the property view for the LAD/FBD program.
2. Click ... .
3. Confirm the message with **OK** and enter the new name in the **New name** input field of the **Change Name** dialog box.
4. Acknowledge the entries with **Apply**.

Assigning an LAD/FBD program to an object-oriented namespace

Proceed as follows to assign an LAD/FBD program to an object-oriented namespace:

1. Open the Properties window of the LAD/FBD program.
2. Enter the identifier of an object-oriented namespace in the **Namespace** field. Identifiers for namespaces must comply with the Rules for identifiers (Page 5155).
3. Confirm with **OK**.

When an identifier is entered, the LAD/FBD program is assigned to the specified namespace. The compiler option (Page 5111) "Permit object-oriented programming" must be activated.

For further information on object-oriented namespaces, see the appropriate section in the SIMOTION ST Programming and Operating Manual.

Changing the LAD/FBD program creation type

To change the LAD/FBD program creation type:

1. Select the new creation type:
   Program
   Programs can be compared with function blocks. Local variables can be stored here either statically or temporarily. In contrast to FBs or FCs, programs can be assigned to a task or an execution level in SIMOTION SCOUT.
   Programs cannot be called up with parameters. Therefore, unlike FBs and FCs, programs do not have any formal parameters.
   Function block (FB)
   A function block (FB) is a program with static data, i.e. all local variables retain their values after the function block has been executed. Only variables explicitly declared as temporary variables lose their value between two calls.
   Before an FB is used, an instance must be defined: Define a variable (VAR or VAR_GLOBAL) and enter the name of the FB as data type. The FB static data is saved in this instance. You can define multiple instances of an FB, with each instance being independent of the others.
   The static data of an FB instance are retained until the next time the instance is called; the static data are reinitialized when the variable type of the FB instance is reinitialized.
   Data transfer to the FB takes place via input or input/output parameters, and the data return from the FB takes place via input/output parameters or output parameters.
   Function (FC)
   A function (FC) is a function block without static data, that is, all local variables lose their value when the function has been executed. They are reinitialized the next time the function is started.
   Data transfer to the function takes place by means of input parameters; output of a function value (return value) is possible.

7.3.4.7 Printing source files and programs

You can print general information about the LAD/FBD source files and programs. Various print options can be set for the printout.

To print LAD/FBD units and programs, proceed as follows:

1. Select a LAD/FBD source file or program in the project navigator.
2. From the context menu, select **Print** or **Print preview**.
   The **Print** dialog box will appear, enabling you to set various print options.

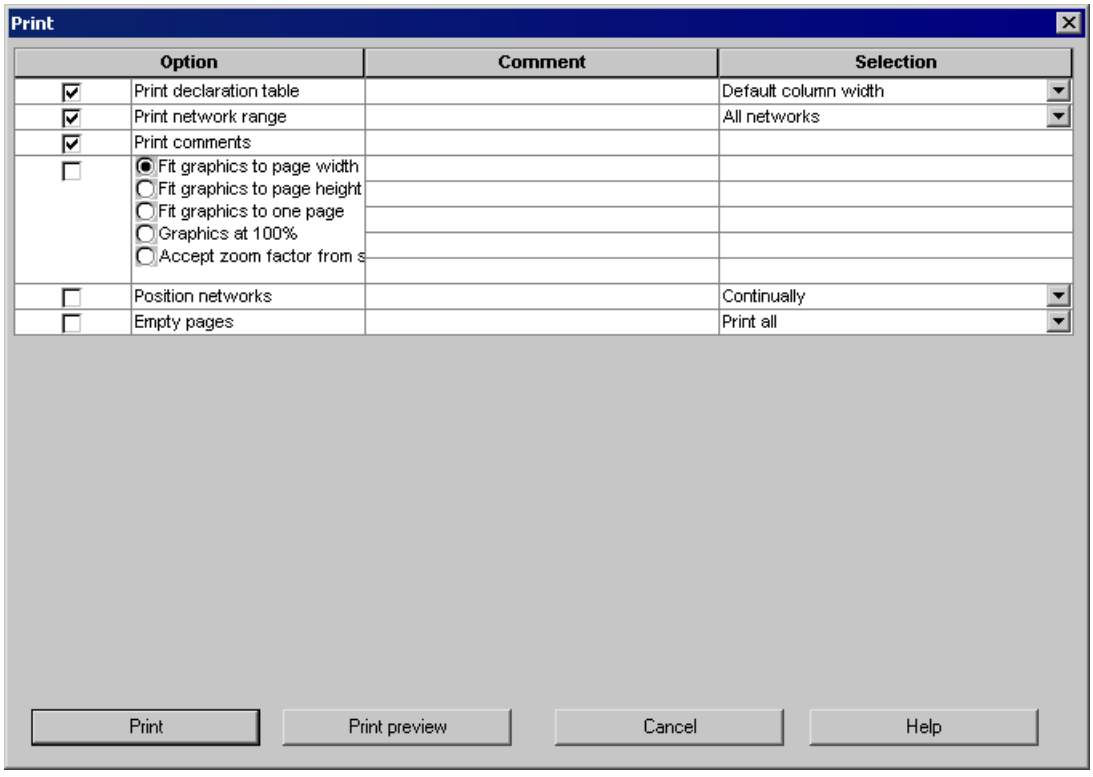


Figure 7-459 Dialog box for setting print options

3. Click the **Print** button.
   The source file or LAD/FBD program is printed with the selected options. General information, the declaration table and diagram all appear in the printout.

Printing a declaration table

To print a declaration table, proceed as follows:

1. Activate the **Print declaration table** check box.
2. Select **Column widths by screen**.
   The contents of the declaration table are printed with the set column widths.
   - or -
   Select **Default column widths**.

Printing a network area

To print a network area, proceed as follows:

1. Activate the **Print network area** check box.
2. Select **All networks**.
   - or -
   Select **Selected networks only**.
   Prints only the networks selected in the editor (blue selection mark on the left side).

Printing comments

You can only select this option if you have already selected the **Print network area** option.

To print comments, proceed as follows:

1. Activate the **Print comments** check box.
2. To obtain a shorter, more concise print image, unselect the **Print comments** option.

Defining print variants

To define the print variant, proceed as follows:

1. Activate the check box.
2. Select **Scale graphics to page width**.
   The print image is scaled so that the widest LAD/FBD network fits on one page width. The print image is one page in width and one or more pages in length, depending on the size of the program.
   - or -
   Select **Scale graphics to page height**.
   The print image is scaled so that the entire graphic fits on one page height. The print image is one page in length and takes up one or more page widths, depending on the width of the networks.
   - or -
   Select **Scale graphics to one page**.
   The print image is reduced so that all networks fit on one page.
   - or -
   Select **Graphic at 100%**.
   The image is printed in its original size. The print image can consist of more than one page vertically or horizontally.
   - or -
   Select **Save screen zoom factor**.
   The image is printed according to the zoom factor set in the editor. The print image can consist of more than one page vertically or horizontally.

Note

If the print image consists of more than one page, an index page is printed to give an overview.

Placing networks

With **Placing networks** you define how the networks are distributed over the pages for printing.

To place networks, proceed as follows:

1. Activate the **Place networks** check box.
2. Select **Continuous**.
   The networks are printed one after another. Page breaks are not taken into account in this case.
   - or -
   Select **All on new page**.
   All networks are printed beginning on a new page. If a network is longer than one page, it is printed on the next page.
   - or -
   Select **Optimized**.
   This minimizes the horizontal break between networks to save more space. E.g.: If a network does not fit on the current page and is not longer than one page, this network will be printed on the next page. If the network is longer, then a page break must be inserted.

Blank pages

You can select how blank pages are printed out. The layout is displayed on the index page. Pages marked with an X are omitted.

To set the printing of blank pages, proceed as follows:

1. Activate the **Blank pages** checkbox.
2. Select **Print all**.
   All blank pages are printed.
   - or -
   Select **Omit at end**.
   Blank pages at the end are not printed. Blank pages in the middle are retained.
   - or -
   Select **Omit all**.
   Blank pages in the middle and at the end are omitted.

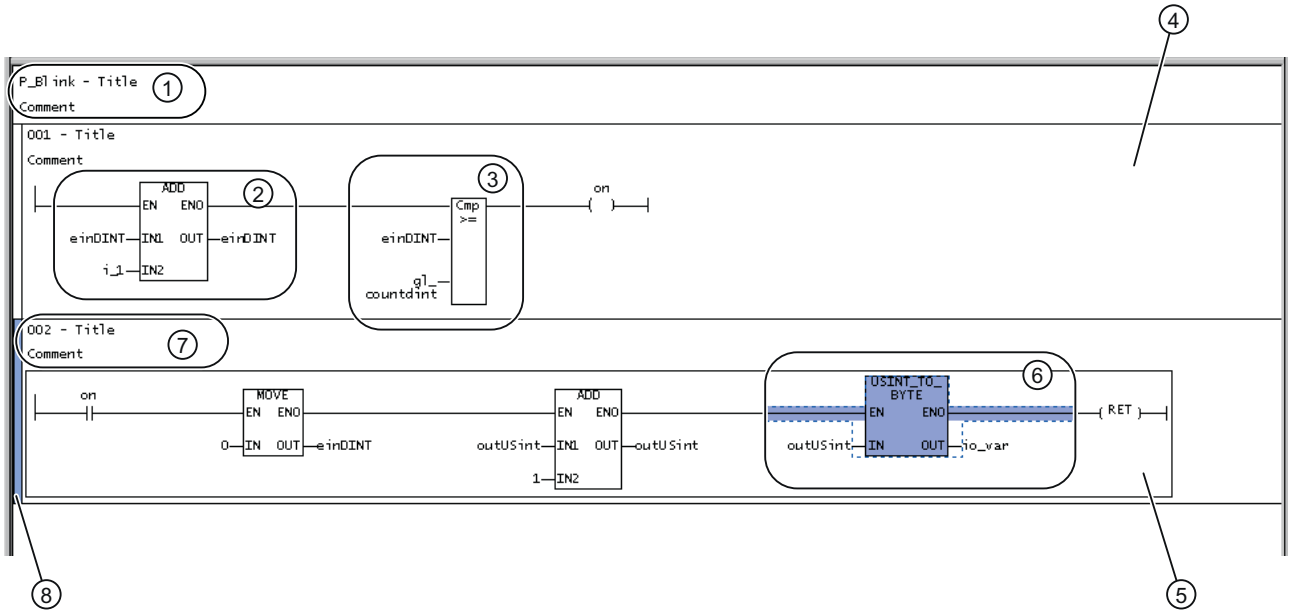
7.3.4.8 LAD/FBD networks and elements

The LAD/FBD program is organized in networks which are displayed in the editor area. A network contains a logic circuit representing the ladder diagram line.

The rules for the structure of a network according to IEC standard 61131-3 apply to the display of a network. Several LAD/FBD elements and boxes can be inserted, copied or deleted in a network.

Note

Use the key combinations (Page 5087)for fast operation in the LAD/FBD editor.



①	Name of program and comment field	⑤	Network 2
②	Call box	⑥	Selected LAD element
③	LAD element	⑦	Name of network and comment field
④	Network 1	⑧	Selected network

Figure 7-460 Display of the networks in the LAD/FBD editor

See also

Numbering the networks (Page 5129)

Inserting networks

To paste in a network:

1. Select an existing network or click in the working window of the open LAD/FBD program.
2. From the context menu, select **Insert network**.
   - or -
   Select **LAD/FBD program > Insert network**.
   - or -
   Click the **Insert network** icon.
   The new network is pasted in directly after the network which is currently selected. If no network is selected, the new network is pasted in at the front.

See also

LAD/FBD networks and elements (Page 5127)

Selecting networks

The relevant networks have to be selected before they can be copied.

To select networks:

1. Select the desired network.
   The network is selected (see figure).
   - or -
   If you want to select several adjacent networks, click the first required network and then, keeping the **Shift** key depressed, click the last one required.
   - or -
   If you want to select several networks which are not adjacent to one another, hold the **Ctrl** key down and click each network you need.

Selected networks are indicated by a light blue edge on their left-hand side. You can choose the selection color in the **LAD/FBD editor** tab of the **Settings** dialog box.

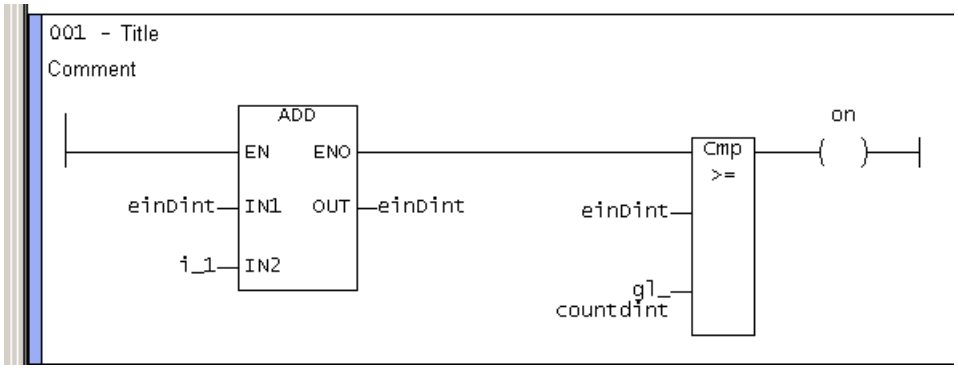


Figure 7-461 Selected network

See also

Settings in the LAD/FBD editor (Page 5092)

Numbering the networks

When a network is pasted in, it is automatically given the next consecutive number. This number is unique and is used to identify the network.

Note

You cannot change the numbering. When a network is deleted, the numbering is automatically adjusted.

See also

LAD/FBD networks and elements (Page 5127)

Enter title/comment

Titles and comments

By default, the LAD/FBD program and/or the network contain a title and a comment field. The title and comment texts are language-dependent.

Language-dependent texts

You can use the **Project > Language-dependent texts** menu item to import and export ASCII files containing translations of LAD/FBD network comments and symbol browser comments (I/O variables, global device variables).

The exported files (**Export** button) can be re-imported into the project using the import function (**Import** button) once they have been translated.

After a language change, the user-defined comments in the project are available in the respective compiled languages.

Assigning a title

The title/name is used for the documentation of the LAD/FBD program or network. It is initialized with the name "Title".

To enter a title, proceed as follows:

1. Click in the title line.
2. Enter a different title/name in the window which appears.
   There is no maximum text length. The length of text visible on the screen depends on the font, font size and screen resolution.

Entering/modifying comments

You can enter a comment in every program or network.

To enter a comment, proceed as follows:

1. Click in the comment line.
2. Enter the text of the comment in the window which appears.
3. To change an existing comment, double-click the existing comment.
4. Overwrite the now selected text.

Showing/hiding a comment line

In every program/network, you can hide a comment that has been entered:

To hide and show comments, proceed as follows:

1. Click in the working window of the open LAD/FBD program.
2. From the context menu, select **Display > Comments on/off**.
   - or -
   Select the **LAD/FBD program > Display > Comments on/off** menu item.
   - or -
   Shortcut **Ctrl+Shift+K**.

This change always applies to the active LAD/FBD editor. The setting is only saved when saving if changes have been made in the respective editor window.

Showing/hiding a jump label

You can paste a jump label in every network.

To paste in or hide jump labels, proceed as follows:

1. Select the network in which the jump label is to be pasted.
2. From the network context menu, select **Jump label ON/OFF**.
3. Enter the text of the jump label in the window that appears.
   Only alphanumeric characters and underscores are allowed during input. The text length of a label must not exceed 480 characters.

   Note

   The jump label is deleted if it contains an error and cannot be corrected.

4. If you want to hide a jump label, select the required jump label and select **Jump label ON/OFF** in the context menu.

See also

Overview of jump operations (Page 5310)

Copying/cutting/pasting networks

If a network is copied or cut, and then pasted in again, all LAD/FBD elements in the network are taken with it.

To copy a network, proceed as follows:

1. Select the required network.
2. In the context menu, select **Copy**or **Cut**.
   - or -
   Select the **Edit > Copy** or **Edit > Cut** menu item.
   The copied network can be pasted again at any place or even in other LAD/FBD programs. A new/copied or cut network is always pasted in after the selected network. If no network is selected, the new network is placed as the first network.

Undo/redo actions

Note

The following actions cannot be undone:

- Save
- Save and compile

To undo or redo actions, proceed as follows:

1. Select the **Edit > Undo** menu command or the **Undo** symbol.
   The actions are undone in reverse order.
2. If you want to redo one or more undone actions, select the **Edit > Redo** menu item or the **Redo** icon.

Deleting networks

To delete a network:

1. Click in a network in the open LAD/FBD program.
2. From the context menu, select **Delete network**.

7.3.4.9 Displaying LAD/FBD elements

LAD diagram

LAD diagram

The LAD diagram complies with Standard IEC 61131-3 and is organized around the binary ladder diagram line. The ladder diagram line begins with a vertical line (conductor bar) and ends with a coil, call-up (box) or with a jump to another network. In between, there are special LAD elements (NO contacts, NC contacts, connectors), general logical elements (SR, RS flipflop), system components call-ups (e.g arithmetic operations), and user functions or function blocks.

Rules for entering LAD statements

- Start of a LAD network
  The left conductor bar is the network's starting point. Crossed lines are not permitted in a LAD diagram. The following elements are not permitted at the beginning of a network: (P), (N), (#).
- LAD network termination
  Every LAD network must terminate with a coil or a box. Multiple outputs are possible.
  The following LAD elements may not be used to terminate a network:
  - (P),
  - (N),
  - POS,
  - NEG,
  - Comparator.
- Placement of empty boxes
  Empty boxes can be placed anywhere in a network except on the right-hand edge or in a parallel branch. Preconnection at binary inputs is supported.
- Placement of coils
  Coils are automatically placed on the right-hand edge of the network, where they are used to terminate a branch.

- Parallel branches
  Parallel branches are
  - opened downward and closed upward.
  - opened behind the selected LAD element.
  - closed behind the selected LAD element.

  Another branch can be inserted between two parallel branches.
  To delete a parallel branch, you must delete all LAD elements of this branch.
  When the last LAD element is removed from the branch, the rest of the branch is also removed.
  The following elements are not permitted in the parallel branch:
  - (P),
  - (N),
  - (#),
  - Empty box.

  The following elements are permitted in the parallel branch:
  - Contacts,
  - Comparators,
  - Edge detection (POS, NEG).

  Parallel branches which branch directly off the power rail are an exception to these placement rules: All elements can be placed in these branches.
- Constants and enums
  Binary operations can also be assigned constants (e.g. TRUE or FALSE).
  FB/FC parameters can be connected with constants that reflect the parameter data type.
  If a parameter is connected with an enum value that is not unique project-wide, preface the enum value with the enum type separated by #.

Meaning of EN/ENO

Enable input (EN) and enable output (ENO) of the LAD box

The LAD box enable input (EN) and enable output (ENO) parameters function according to the following principles:

- If EN is not enabled (i.e., the signal is set to "0"), then the box will not execute its function, and ENO is not enabled (i.e., the signal is also "0").
- If EN is enabled (i.e., the signal is set to "1"), then the appropriate box executes its function, and then ENO is also enabled (i.e., the signal is also "1").

FBD diagram

FBD diagram

An FBD diagram complies with IEC standard 61131-3.
The main binary signal line begins with a logic box (top left) and ends with an assignment, call (box), or with a jump to another network. In between, there are logic elements (AND, OR box), general logic elements (SR, RS flip-flop), system component call-ups (e.g. arithmetic operations), and user function or function block call-ups.

Rules for entering FBD statements

- Placement of boxes
  Empty boxes (flipflops, counters, timers, arithmetic operations, device-specific commands, TO-specific commands, etc.) can be attached to boxes with binary connections (&, =1, XOR). Preconnections on binary inputs (e.g. S input on flipflop) are allowed.
  Separate connections with separate outputs cannot be programmed in a network. Junctions are not supported.

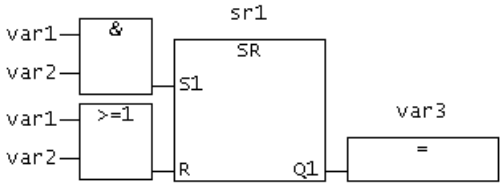


Figure 7-462 FBD with binary preconnection

- &, >=1, XOR boxes
  Binary inputs can be inserted, deleted, or negated in these boxes.
- Enable input/enable output
  Connection of the enable input EN and/or the enable output ENO of boxes is possible.
- Constants and enums
  Binary operations can also be assigned constants (e.g. TRUE or FALSE).
  FB/FC parameters can be connected with constants that reflect the parameter data type.
  If a parameter is connected with an enum value that is not unique project-wide, preface the enum value with the enum type separated by #.

Converting between LAD and FBD representation

Converting from LAD to FBD representation

To switch from **LAD to FBD representation**, proceed as follows:

1. Open an existing LAD project.
2. Select the **LAD/FBD program > Switch to FBD** menu item.
   - or -
   Click the button for "Switch to FBD" (Ctrl+3 shortcut) in the LAD editor toolbar.
   The project is now displayed in the **FBD** programming language.

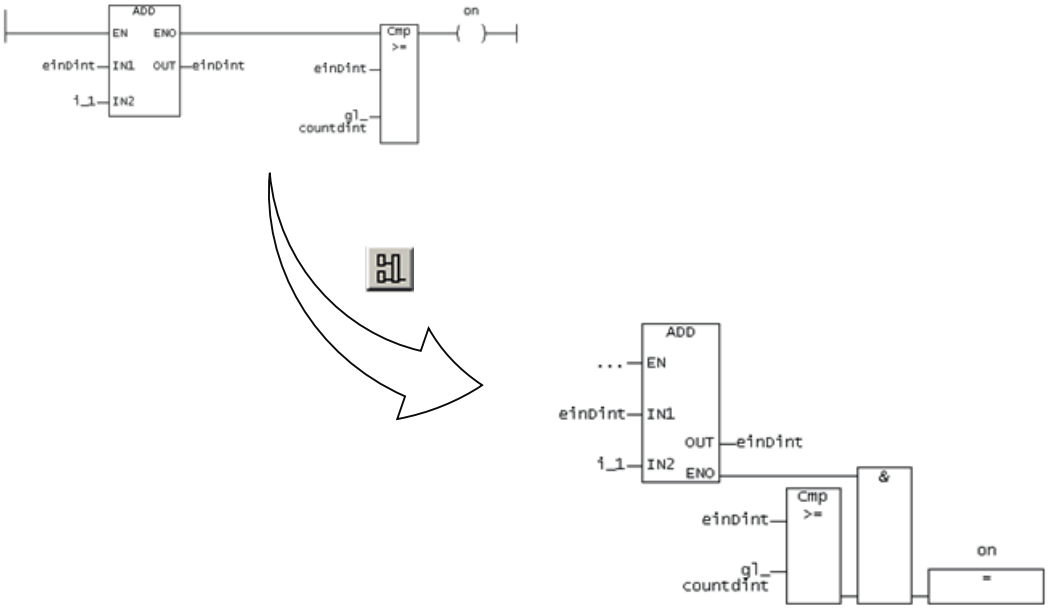


Figure 7-463 Switching from LAD to FBD

Note

A conversion sequence of **LAD - FBD - LAD** always produces the original network.

Anything generated in LAD can always be displayed in FBD.

Converting from FBD to LAD representation

To switch from **FBD to LAD representation**, proceed as follows:

1. Open an existing FBD project.
2. Select the **LAD/FBD program > Switch to LAD** menu command.
   - or -
   Click the button for "Switch to LAD" (Ctrl+1 shortcut) in the FBD editor toolbar.
   The project is now displayed in the **LAD** programming language.

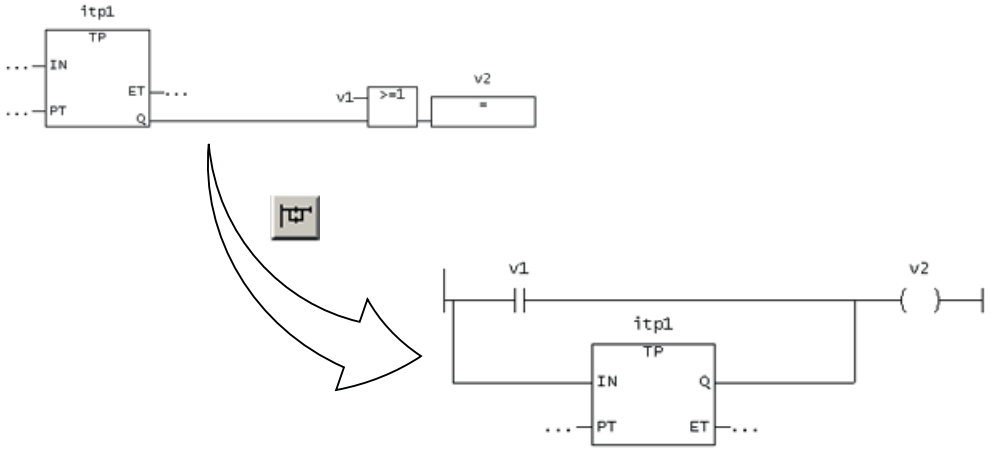


Figure 7-464 Switching from FBD to LAD, example with OR box

Note

A conversion sequence of **FBD- LAD- FBD** only produces the original LAD network if the FBD structure can be converted to LAD.

Something generated in FBD cannot always be displayed in LAD.

Example of a non-convertible FBD structure

Figure 7-465 FBD structure with binary XOR box

7.3.4.10 Editing LAD/FBD elements

Inserting LAD/FBD elements

LAD/FBD elements are usually inserted to the right of the selected position in the network.

FBD elements are usually inserted at a boolean input of a block or at an assignment (left).

Special case:

If the right-hand edge of a network or a coil (LAD) or an assignment (FBD) is selected, the next element is added in the network on the left-hand side of it.

To insert an LAD/FBD elements:

1. Select the position in a network behind which you want to insert an LAD/FBD element.
2. Insert an LAD/FBD element:
   - Via the icons on the toolbar
   - Using the menu item, e.g. **LAD/FBD program > Insert element > Empty box**
   - With a drag and drop operation from the **Command library** tab
   - By double-clicking the element in the **Command library** tab
   - By selecting the element in the **Command library** tab and confirming with the **Enter key**

   The selected LAD/FBD element is inserted and the placeholders and ... are inserted for variables and parameters.

Note

A red **???** symbol indicates mandatory parameters that must be connected.

A black ... character string indicates optional parameters that can be connected.

Move the cursor over the parameter name to display the expected data type.

Syntax check in LAD

An automatic syntax check during input prevents the incorrect placement of elements.

- NOT in parallel branch
- FB/FC call in parallel branch
- Connector in parallel branch
- Check 0 -> 1 edge and 1 -> 0 edge in parallel branch
- XOR in parallel branch

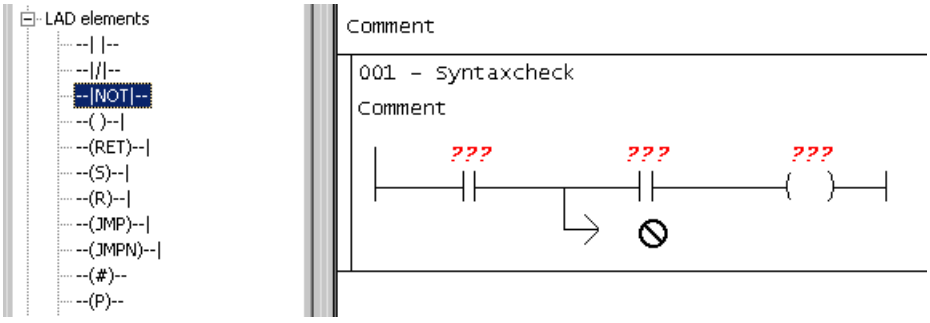


Figure 7-466 Syntax check

Selecting LAD/FBD elements

LAD/FBD networks must be selected before they can be deleted, for example.

To select an individual LAD/FBD element:

- Click the required LAD/FBD element.
  The LAD/FBD element is selected (see figure below).

To select several consecutive LAD/FBD elements, proceed as follows:

1. Click the first LAD/FBD element to select it.
2. Then, keeping the **Shift** key pressed, click the last LAD/FBD element to be selected.
   The consecutive LAD/FBD elements are selected.

To select several specific LAD/FBD elements, proceed as follows:

1. Click the first LAD/FBD element to select it.
2. Keeping the **Ctrl** key pressed, click all the other LAD/FBD elements to be selected.
   The specific LAD/FBD elements are selected.

Selected LAD/FBD elements have a light blue background. You can choose the selection color in the **LAD/FBD editor** tab of the **Settings** dialog box.

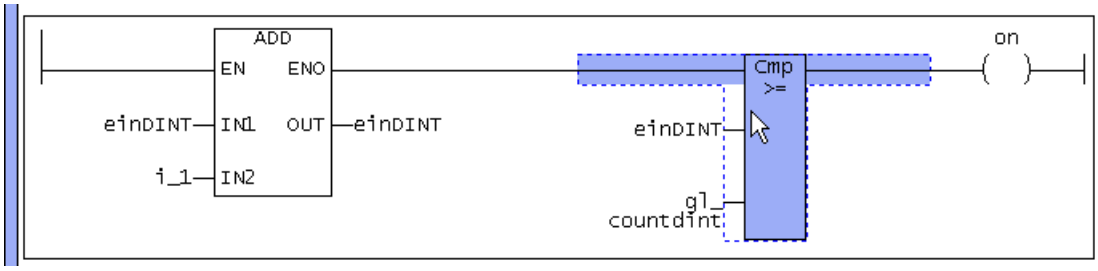


Figure 7-467 Selected LAD/FBD elements

Copy/cut/delete operations in LAD/FBD elements

To **copy/cut/delete**, proceed as follows:

1. Select a LAD/FBD element.
2. In the context menu or in the **Edit** menu, select e.g. **Copy**.
   The copied/cut LAD/FBD element can be inserted into other LAD/FBD programs.
   If you delete an FB/FC box with binary preconnections, this results in several open branches (in LAD) or sub-networks (in FBD) which can be further connected.

LAD/FBD elements - defining parameters (labeling)

To label the elements, proceed as follows:

1. Click the parameter.
2. Label the parameter:
   - Select the corresponding parameter from the pull-down menu (for box-type only).
     - or -
   - Enter the appropriate variable.
     - or -
   - Drag the corresponding variable from the declaration table using a drag-and-drop operation.
3. Confirm the entry with the **Return** key.

See also

Setting call parameters (Page 5147)

Defining variables in the Variable declaration dialog box ("on-the-fly" variable declaration) (Page 5175)

Labeling LAD/FBD elements with the symbol input help dialog

To label the element with the **Symbol input help**, proceed as follows:

1. Select the parameter you want to label.
2. Right-click to open the context menu.
3. Click the **Symbol input help menu.**
   - or -
   Call the symbol input help with the key shortcut **Ctrl+Alt+H.**
   The **Symbol input help** dialog box opens. The tree structure shows all variables which exist in the project and which can be used.
4. Select the desired variable and click **OK** to confirm.
   The label is entered in the selected parameter. If the variable is defined in another program source or in a library, a Connection (Page 5222) is created automatically.

Setting the LAD/FBD element display

In order to ensure a manageable view of relatively large call boxes, you can set the display mode of **LAD/FBD elements**.

To set the **LAD/FBD element** display, proceed as follows:

1. Click in the editor area of the LAD/FBD program.
2. Select the required display mode:
   - In the context menu, select **View > No box parameters** or the **LAD/FBD program > View > No box parameters** menu item.
     - or -
   - In the context menu, select **View > Only assigned box parameters** or the **LAD/FBD program > View > Only assigned box parameters** menu item.
     - or -
   - In the context menu, select **View > Mandatory and assigned box parameters** or the **LAD/FBD program > View > Mandatory and assigned box parameters** menu item.
     - or -
   - In the context menu, select **View > All box parameters** or the **LAD/FBD program > View > All box parameters** menu item.
     This box parameter setting is also saved when storing.

Note

If a call box has non-represented parameters, this is indicated by **...** at the bottom of the box.

Select box type with empty box

Requirement

You have inserted an empty box into the network, e.g. via the **LAD/FBD program > Insert element > Empty box** menu.

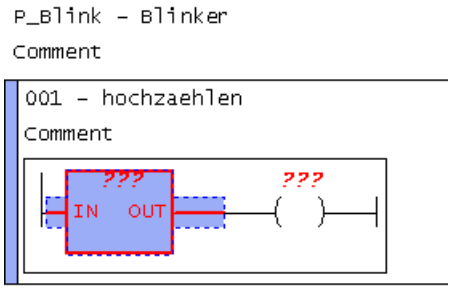


Figure 7-468 Empty box

Specify empty box

There are two alternatives for specifying the box type:

- Via an editable combo box (Page 5142)
- Via the call assistance (Page 5143)

Specify the box type via the editable combo box

Requirement

Empty box is inserted.

Procedure

How to open the editable combo box at the empty box (alternatives):

- Select Empty box and press **Enter**.
- Click on the **???** field.

The editable combo box opens.

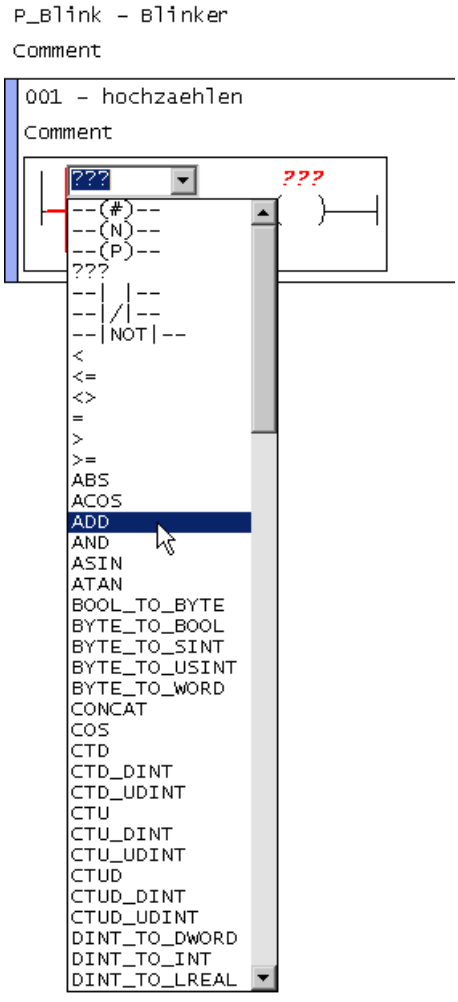


Figure 7-469 Drop-down menu for the combo box

The following are available for selection in the drop-down menu:

- The standard functions and function blocks, see Functions (Page 5269)
- The functions and function blocks of the own LAD/FBD source
- The functions and function blocks exported into connected program sources or libraries
- The public methods of exported classes (as of Kernel V4.5) and function blocks defined in connected ST source files or libraries.

Select the required box type. Alternatively, you may enter the box type into the combo box.

The empty box is replaced by the appropriate box for selection.

Specify box type via call assistance

Requirement

The empty box is inserted.

Procedure

How to open the call assistance:

- Double-click the **???** field.

The "Call Assistance" window opens.

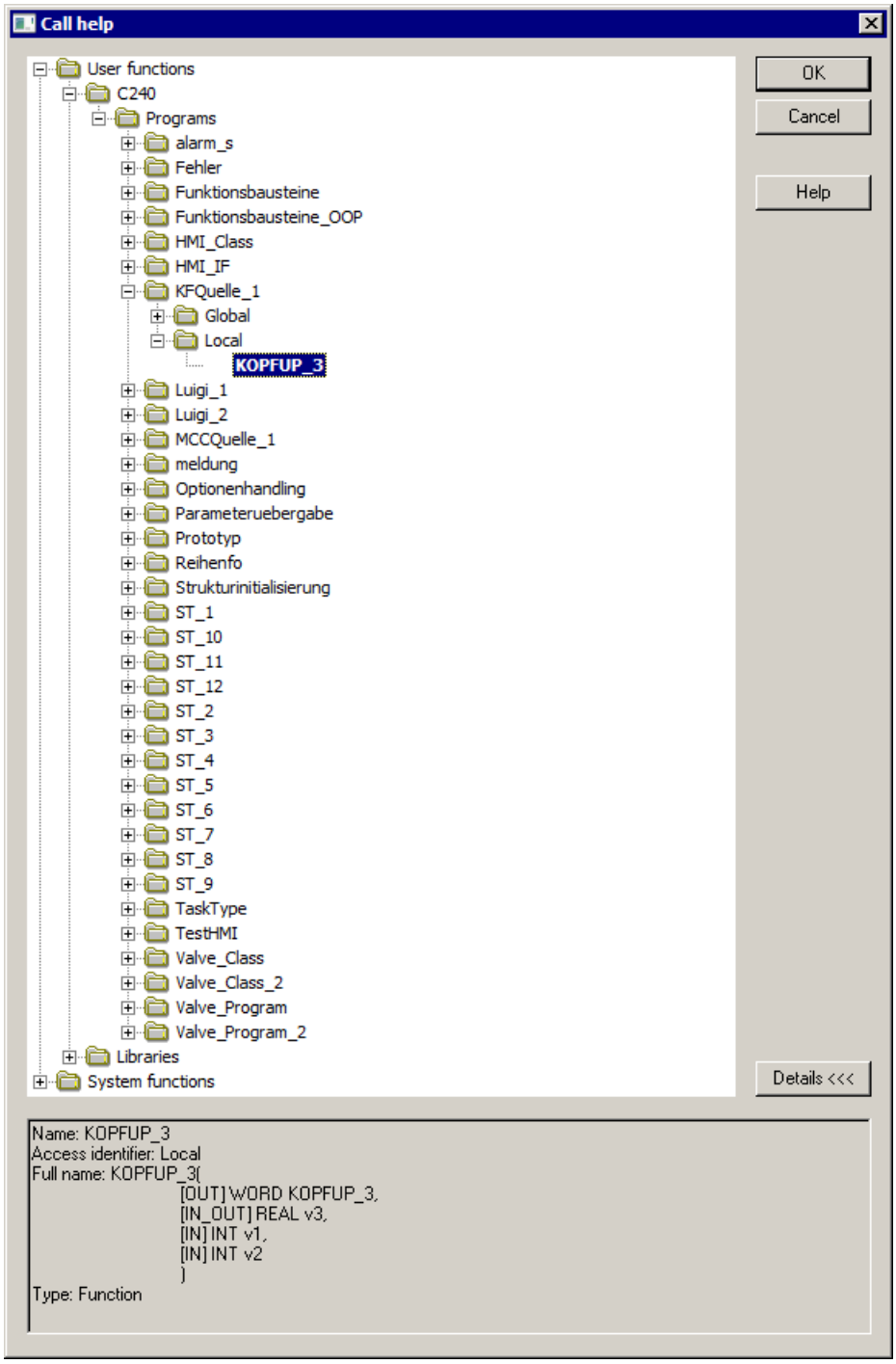


Figure 7-470 Call assistance

The following are displayed as roots in a tree topology:

- User functions
  All program sources for the SIMOTION device and for the libraries for the project are displayed beneath this. You can select the following program organization units (POUs):
  - All functions and function blocks of the own LAD/FBD unit
  - The functions and function blocks exported into program sources or libraries
  - The public methods of exported classes (as of Kernel V4.5) and function blocks defined in ST source files or libraries.

  Functions, function blocks or methods that are assigned to an object-oriented namespace are displayed below the program source or library as subelement of the namespace.
- System functions
  The standard functions and function blocks, see Functions (Page 5269)

Information about the selected POU can be found in the window footnote. This information can be hidden with the **Details <<<** button.

To select the desired POU (alternatives):

- Select the desired POU and click **OK**.
- Double-click the desired POU.

The "Call Assistance" window closes and the empty box is replaced by the appropriate box for selection.

Note

The LAD/FBD editor saves information on the origin of the subprogram when it is selected. This has no effect on the code generation.

This way you can subsequently move the source of a subprogram from the device to a connected library without changing the parameters of the box. The LAD/FBD unit is still compiled correctly.

A check as to whether an identifier of a subprogram is hidden by an identifier with the same name in the LAD/FBD unit, is not performed. You can avoid name conflicts by using namespaces.

Setting the call parameter for an individual parameter

To set an individual call parameter, proceed as follows:

1. Double-click the parameter input/output you want to set.
   The **Enter call parameter for individual parameter** dialog box appears.

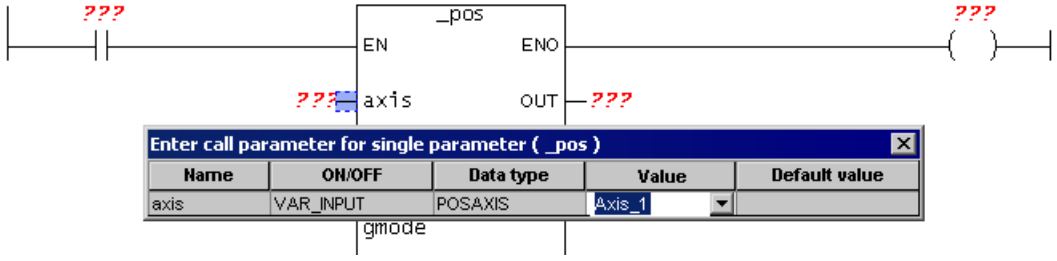


Figure 7-471 Dialog box for setting an individual call parameter

2. Assign a variable or value to the parameter from the **Values** list.
3. Confirm your selection twice with the Enter key to close the dialog box again.

Setting call parameters

To set the call parameters, proceed as follows:

1. Label the type parameters of the box.
2. Double-click the box.
   - or -
   In the shortcut menu, select **Call parameters ...**.
   The **Enter call parameters** dialog box appears.
   Only variables which have already been declared and symbols/variables offered by the system are displayed.

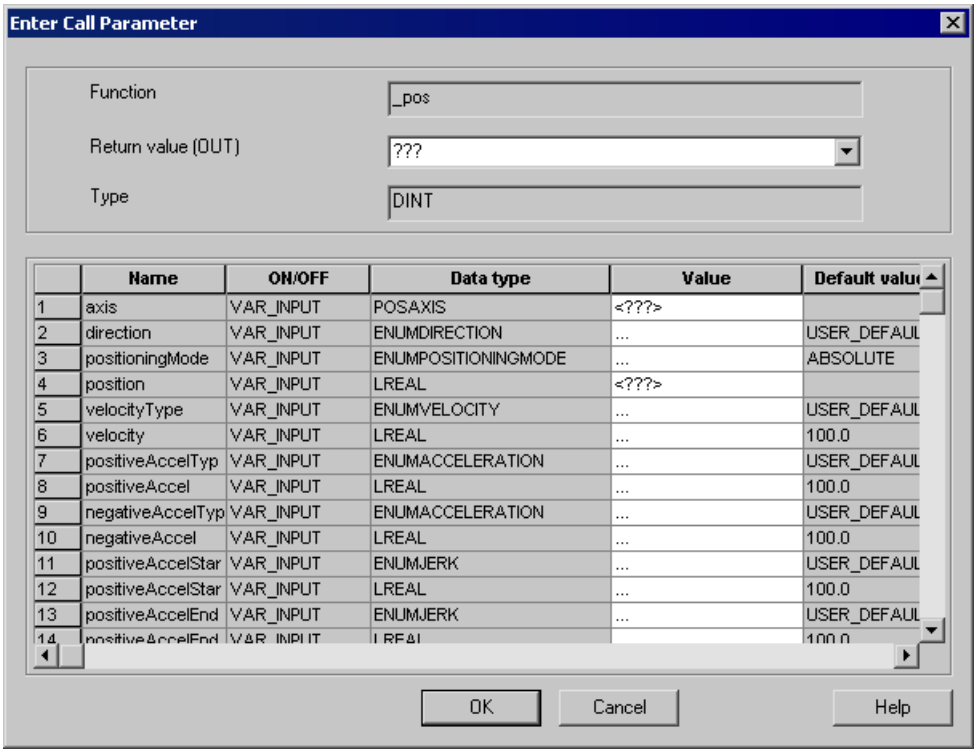


Figure 7-472 Dialog box for setting call parameters

3. Enter:
   - **Instance**
     Here, you enter the instance of the function block or the class.
   - **Return value**
     Here you assign the function or method return value to a variable of the calling program.
   - **Value**
     Here, you can assign current variables or values to the parameters.

   See also Overview of subprogram call parameters (Page 5230).
4. Confirm with **OK**.

Note

The **Value** list includes all visible symbols in the current target (variables, Enum values, etc.) whose type matches the data type of the parameter. Implicit data type conversion is taken into consideration here. You can select a symbol from the list or enter one yourself.

The value of string constants must be entered in inverted commas
(e.g. 'st_until')

7.3.4.11 Command library

LAD/FBD functions in the command library

The command library appears automatically as a tab in the project navigator. The command library stays open after the programming window is closed.

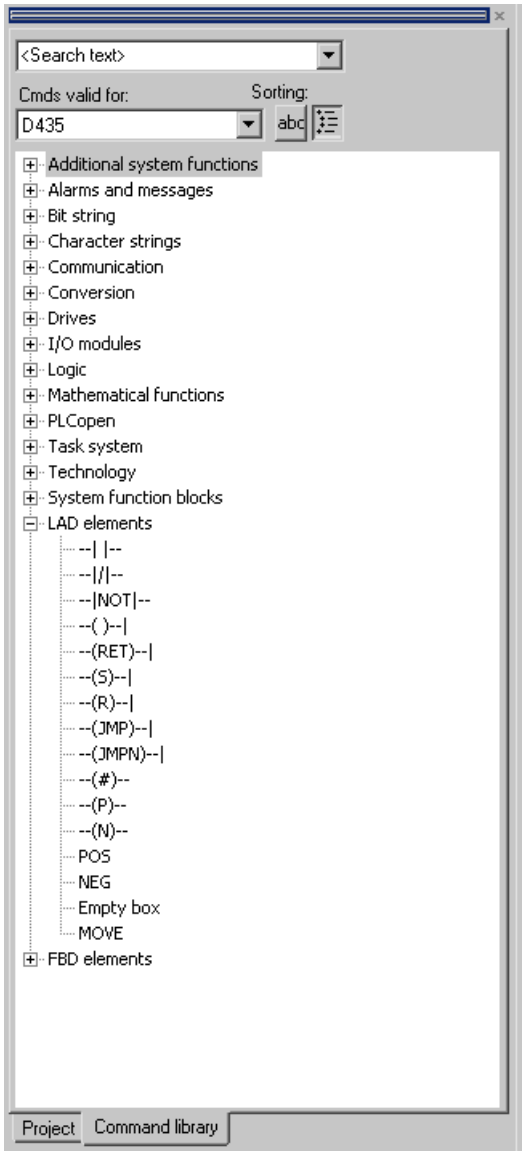


Figure 7-473 Command library tab of the project navigator

Inserting elements/functions from the command library

To paste elements/functions into a programming window:

1. Left-click the desired function in the command library, drag the function onto the editor window while keeping the left mouse button depressed and then release the left mouse button.
   - or -
   Double-click the desired function.
   - or -
   Select the desired function and press the Enter key.

LAD/FBD elements are usually pasted in to the right of the selected position in the network. FBD elements are usually pasted in at a boolean input of a block or at an assignment (left).

Description of PLCopen blocks

PLCopen blocks are preferably used for the programming of motion control tasks in the LAD/FBD programming language. The PLCopen blocks are intended for use in cyclic programs/tasks.

The following standard function blocks are available in the command library in SIMOTION. They are certified in accordance with "PLCopen Compliance Procedure for Motion Control Library V1.1".

The details and the use of the PLCopen blocks are described in the PLCopen Blocks Function Manual.

An example for the use of the PLCopen blocks is contained in Section "Positioning axis program (Page 5390)".

Table 7-525 Single-axis function blocks for the axis

Function block	Description
_MC_Power()	Enabling/disabling axis
_MC_Stop()	Stopping the axis
_MC_Home()	Homing axis/clearing absolute value encoder offset
_MC_MoveAbsolute()	Absolutely positioning axis
_MC_MoveRelative()	Relatively positioning axis
_MC_MoveVelocity()	Traversing axis at defined velocity
_MC_MoveAdditive()	Positioning relative to current target position (traversing axis using an additional, defined path, relative to current position setpoint)
_MC_MoveSuperimposed()	Superimposed positioning (traversing axis relative to current motion)
_MC_PositionProfile()	Traveling through position/time profile (traversing axis along a predefined, fixed position/time profile)
_MC_VelocityProfile()	Traveling through velocity/time profile (traversing axis along a predefined, fixed velocity/time profile)
Basic functions	
_MC_Reset()	Resetting errors/alarms on the axis or triggering a restart
_MC_ReadActualPosition()	Reading the actual position of axis
_MC_ReadStatus()	Reading the status of an axis
_MC_ReadAxisError()	Reading the error of an axis

Function block	Description
_MC_ReadParameter()	Reading axis parameter and outputting in data type LREAL
_MC_ReadBoolParameter()	Reading axis parameter and outputting in data type BOOL
_MC_WriteParameter()	Writing axis parameter of data type LREAL
_MC_WriteBoolParameter()	Writing axis parameter of data type BOOL
In addition to the standard function blocks, the following function block is available for an axis:	
_MC_Jog()	Continuous or incremental jogging

Table 7-526 Multi-axis function blocks for the axis

Function	Description
_MC_GearIn()	Starting gearing (synchronizing master and slave axis while taking into account a positional relationship described by a fixed gear ratio)
_MC_GearOut()	Terminating gearing (desynchronizing master and slave axis)
_MC_CamIn()	Starting camming (synchronizing master and slave axis while taking into account a positional relationship described by a cam)
_MC_CamOut()	Terminating camming (desynchronizing master and slave axis)
_MC_Phasing()	Changing phase shift between the leading axis and following axis

Table 7-527 Function blocks for the external encoder

Function	Description
_MC_Power()	Enabling external encoder
_MC_Reset()	Resetting external encoder
_MC_Home()	Homing external encoder
_MC_ReadActualPosition()	Reading actual position of external encoder
_MC_ReadStatus()	Reading external encoder status
_MC_ReadAxisError()	Reading external encoder error
_MC_ReadParameter()	Reading external encoder parameter and outputting in data type LREAL
_MC_ReadBoolParameter()	Reading external encoder parameter and outputting in data type BOOL

Special features of the command library

Special features

The following ST commands have no corresponding function that can be used with LAD/FBD:

- BOOL := _checkequaltask([IN]TASK, [IN]TASK)
  Two **StructTaskID** or two **StructAlarmID** can be compared to one comparator.
- StructAlarmID := _getalarmid([IN]ALARM)
  These alarm commands can be found in the **_alarm** name space (_alarm.myalarm).
- StructTaskID := _gettaskid([IN]TASK id:=TaskIdThis)
  The task commands can be found in the **_task** name space (_task.backgroundtask).

Examples for parameters of type _alarmid and _starttaskid

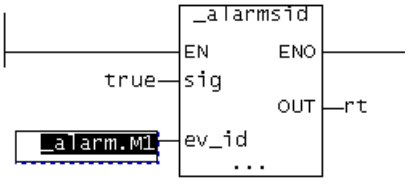


Figure 7-474 Examples for StructAlarmID

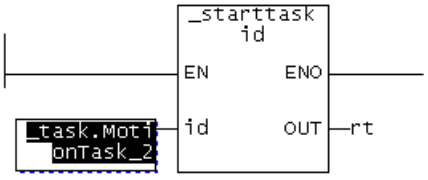


Figure 7-475 Example for StructTaskID

7.3.4.12 General information about variables and data types

Overview of variable types

The following table shows all the variable types available for programming with ST.

- System variables of the SIMOTION device and the technology objects
- Global user variables (I/O variables, device-global variables, unit variables)
- Local user variables (variables within a program, a function, or a function block)

System variables

Variable type	Meaning
System variables of the SIMOTION device	Each SIMOTION device and technology object has specific system variables. These can be accessed as follows: • Within the SIMOTION device from all programs • From HMI devices You can monitor system variables in the symbol browser.
System variables of technology objects	

Global user variables

Variable type	Meaning
I/O variables	You can assign symbolic variable names to the I/O addresses of the SIMOTION device or the peripherals. This allows you to have the following direct accesses to the I/O: • Within the SIMOTION device from all programs • From HMI devices You create these variables in the symbol browser after you have selected the I/O element in the project navigator. You can monitor I/O variables in the symbol browser.
Global device variables	User-defined variables which can be accessed by all SIMOTION device programs and HMI devices. You create these variables in the symbol browser after you have selected the GLOBAL DEVICE VARIABLES element in the project navigator. Global device variables can be defined as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor global device variables in the symbol browser.
Unit variables	User-defined variables that all programs (programs, function blocks, and functions) can access within a unit (source file). You declare these variables in the declaration table of the source file: • In the interface section: These variables are exported and can be used in other units (e.g. ST source files, MCC source files, LAD/FBD source files) after a connection has been defined (Page 5223). They are also available on HMI devices as standard. • In the implementation section: You can only access these variables within the source file. You can declare unit variables as retentive. This means that they will remain stored even when the SIMOTION device power supply is disconnected. You can monitor unit variables in the symbol browser.

Local user variables

Variable type	Meaning
	User-defined variables that can only be accessed within the program/chart (program, function, function block) in which they were defined.
Variable of a program (program variable)	Variable is declared in a program. The variable can only be accessed within this program. A differentiation is made between static and temporary variables: • Static variables are initialized according to the memory area in which they are stored. Specify this memory area by means of a compiler option. By default, the static variables are initialized depending on the task to which the program is assigned (see *SIMOTION Basic Functions* Function Manual). You can monitor static variables in the symbol browser. • Temporary variables are initialized every time the program in a task is called. Temporary variables cannot be monitored in the symbol browser.
Variable of a function (FC variable)	Variable is declared in a function (FC). The variable can only be accessed within this function. FC variables are temporary; they are initialized each time the FC is called. They cannot be monitored in the symbol browser.
Variable of a function block (FB variable)	Variable is declared in a function (FB). The variable can only be accessed within this function block. A differentiation is made between static and temporary variables: • Static variables retain their value when the FB terminates. They are initialized only when the instance of the FB is initialized; this depends on the variable type with which the instance of the FB was declared. You can monitor static variables in the symbol browser. • Temporary variables lose their value when the FB terminates. The next time the FB is called, they are reinitialized. Temporary variables cannot be monitored in the symbol browser.

Scope of the declarations

Scope of variable and data type declarations according to location of declaration

Location of declaration	What can be declared here	Scope
Symbol browser	• Global device variables • I/O variables	The declared variables are valid in all units (e.g., ST source files, MCC source files, LAD/FBD source files) of the SIMOTION device. All programs, function blocks, and functions in all units of the device can access the variables.
Interface section of the unit[1]	• Unit variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., are valid in the entire unit (e.g., ST source file, MCC source file, LAD/FBD source file); all programs, function blocks, and functions within the unit can access them. In addition, they are also available in other units after connection (see Define connections (Page 5223)).
Implementation section of the unit[1]	• Unit variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., are valid in the entire unit (e.g., ST source file, MCC source file, LAD/FBD source file); all programs, function blocks, and functions within the source file can access them.

Location of declaration	What can be declared here	Scope
POU (program/ function block/ function)[2]	• Local variables • Data types • Symbolic accesses to the fixed process image of the BackgroundTask	The declared variables, data types, etc., can only be accessed within the POU in which they were declared.

[1] MCC and LAD/FBD programming languages: in the declaration table of the respective source file.

[2] MCC and LAD/FBD programming languages: in the declaration table of the respective chart/program.

Rules for identifiers

Names for variables, data types, charts/programs must comply with the following rules for identifiers:

1. They are made up of letters (A to Z, a to z), numbers (0 to 9), and underscores (_).
2. The first character must be a letter or underscore.
3. This can be followed by as many letters, digits or underscores as needed in any order.
4. Exception: You must not use more than one underscore in succession.
5. Both upper- and lower-case letters are allowed. No distinction is made between upper- and lower-case notation (thus, for example, Anna and AnNa are regarded as identical).

Note

Reserved identifiers

Reserved identifiers may only be used as predefined. You may not declare a variable or data type with the name of a reserved identifier.

There is no distinction between upper- and lower-case notation.

You can find a list of all the identifiers whose meanings are predefined in SIMOTION in the SIMOTION Basic Functions Function Manual.

Note

Identifiers for SIMOTION devices

Identifiers for SIMOTION devices do not have to comply with rules specified above. When used in SIMOTION SCOUT they must be enclosed in double inverted commas (", ASCII code $22). See the SIMOTION ST Programming and Operating Manual.

Frequently used arrays in declarations

Reference (as of kernel V4.5)

References can be formed as of version V4.5 SIMOTION Kernel. The compiler option (Page 5111) "Object-oriented programming" must also be activated.

By activating the **Reference** checkbox, you specify that the declared variable (or the component of a structure) contains the reference to the specified data type. This means that the variable can contain address information for a variable of the specified data type.

The following are permitted as data types to which references can be formed:

- Elementary data types (e.g. INT, DINT, REAL, WORD, TIME, STRING)
- User-defined data types (UDT)
- System data types
- Function blocks, provided they contain at least one static variable
- Classes (from ST sources)

No references can be formed to the following data types because references exist already:

- Technology object data types
- Object-oriented interfaces (from ST sources)
- General references
- I/O references (from ST sources)

Array length and array element

A field is a chain of variables of the same type that can be addressed with the same name and different indices.

You can define the variable as a field [0...N-1] by entering a field length N.

You have the following options for entering the field length:

- You can enter a constant positive integer value.
- You can enter a value range with ".." separating the min. and max. values.
- You can enter a constant expression of data type DINT (or of a data type that is implicitly convertible to DINT).

If the field is empty, a single variable is set up rather than a field.

Example definition of a field in the declaration table

	Name	Variable type	Data type	Array length	Initial value	Comment
1	const_1	VAR_GLOBAL CONSTANT	INT		11	constant
2	const_2	VAR_GLOBAL CONSTANT	INT		5	constant
3	array_4	VAR_GLOBAL	INT	11	11(5)	specification of the array length by value
4	array_5	VAR_GLOBAL	INT	const_1	11(5)	specification of the array length by constant expression
5	array_6	VAR_GLOBAL	INT	-5 .. 5	11(5)	specification of the array length by range of values
6	array_7	VAR_GLOBAL	INT	const_2 .. 3 * const_2	11(5)	specification of the array length by range of values as constant expression
7						

Figure 7-476 Defining the length of a field

Example of use of field elements in a variable assignment

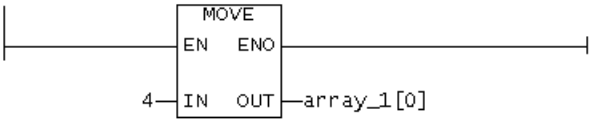


Figure 7-477 Use of field elements in a variable assignment

Initial value

You can specify an initialization value in this column. The field cannot be edited, but is provided as button.

1. Click the button. A window opens:

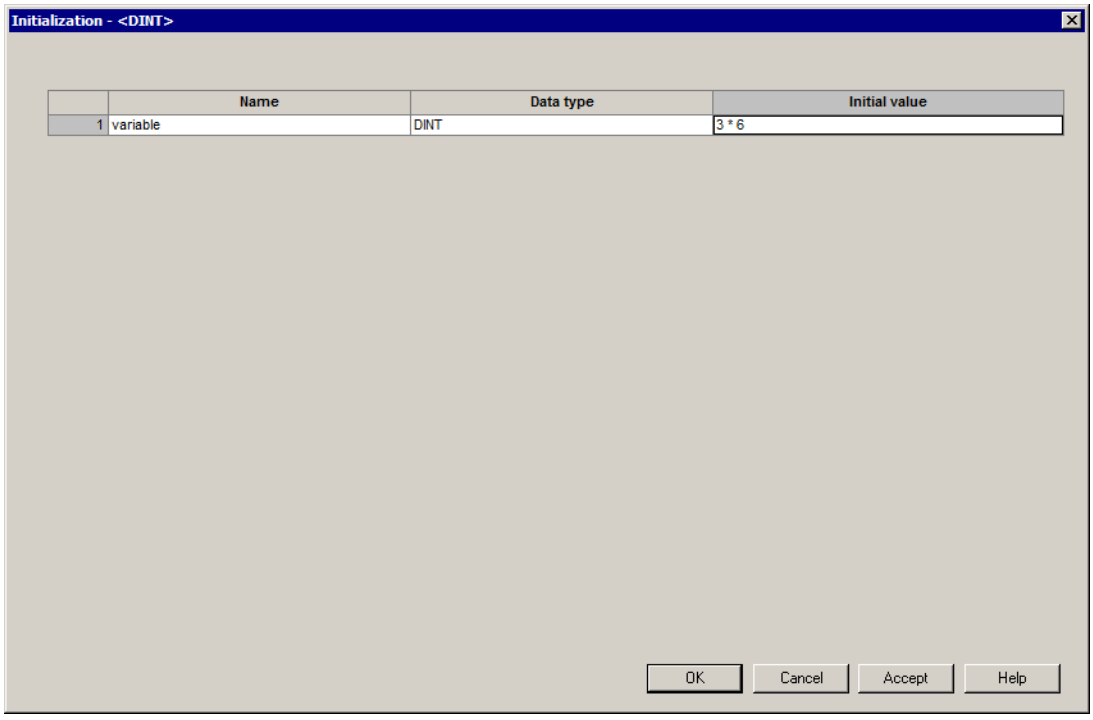


Figure 7-478 Definition an initialization value

2. Enter the initialization value in the **Initial value** column.
   You can specify this initialization value as a constant or as an expression. The following are permissible:
   - Constants
   - Arithmetic operations
   - Bit slice and data conversion functions
3. Confirm with **OK**.

Variables of a technology object data type cannot be assigned an initialization value. They are always initialized by the compiler with TO#NIL.

Initialization of arrays

Per default, the compiler assigns all array elements the initialization value of the data type. The initialization values can be changed by specifying an array initialization list according to the following example.

Table 7-528 Preassignment of array elements

10(1)	10 array elements [0..9] are pre-assigned the same value "1".
1,2,3,4,5	5 array elements [0..4] are pre-assigned different values "1", "2", "3", "4" and "5".
5(3),10(99),3(7),2(1)	The following array elements are pre-assigned the following values: • Five array elements [0..4] with the same value "3". • 10 array elements [5..14] with the same value "99". • 3 array elements [15..17] with the same value "7". • 2 array elements [18..19] with the same value "1".

Definition of these initialization values in the declaration table:

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	array_1	VAR_GLOBAL	INT		10	10(1)	all array elements equal
2	array_1_	VAR_GLOBAL	INT		5	1, 2, 3, 4, 5	all array elements diverse
3	array_1_	VAR_GLOBAL	INT		20	5(3), 10(99), 3(7), 2(1)	several array elements respectively equal
4							

Figure 7-479 Definition of the initialization values of an array

Initialization of structures

Per default, the compiler assigns all components of a structure the initialization value of their data type. By specifying an initial value for a component, its initialization can be changed.

When using the structure in another declaration (e.g. variable or structure definition), the initialization values of individual components can be changed by assigning a structure initialization list, enclosed in round brackets ( ), in accordance with the following example.

	Structure name	Element name	Data type	Array length	Initial value	Comment
1	type_struct_1	s11	INT		1	
2		s12	DINT	3	3(2)	
3		s13	UDINT		3	
4	type_struct_2	s21	LREAL		3.0	
5		s22	DWORD		16#0000_FFFF	
6	type_struct_3	s31	type_struct_1		(s11 := 10, s12 := [3(20)])	
7		s32	type_struct_2		(s22 := 16#0000_00FF)	
8		s33	REAL		10.0 / 3.0	
9						

Figure 7-480 Definition of the initialization values of a structure

Instance-specific initialization of classes and function blocks

For the instance formation of classes and function blocks created during the object-oriented programming, variables declared as VAR OVERRIDE or VAR PUBLIC can be initialized instance-specific. The initialization values specified for the definition of the associated program organization unit (POU) are overwritten.

Proceed as follows to initialize the instance variables instance-specific:

1. Click the button in the **Initial value** column. A window opens.

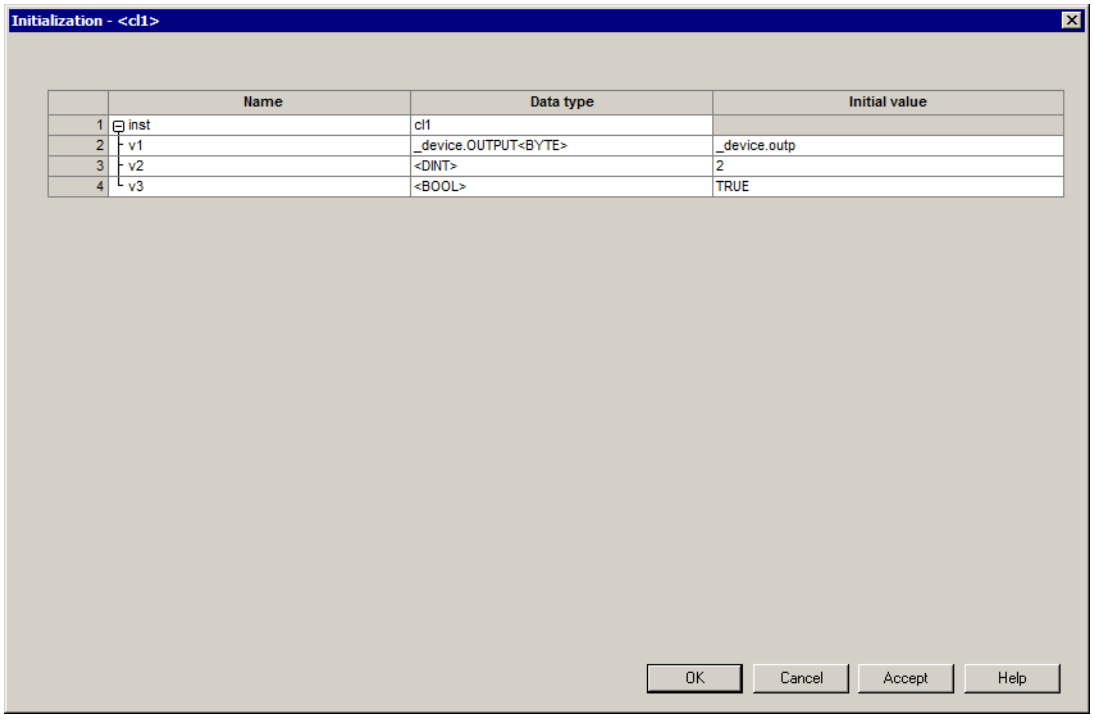


Figure 7-481 Definition of the initialization values of a class or function block

   Variables that must be initialized are already displayed in this window with their name and data type; the associated line is color-highlighted.

2. You can add additional lines for variables whose initialization is optional:
   - Click the number in line 1 and select **Add new line** in the context menu.
   - Enter the name of the variable. The data type is then initialized automatically.
3. Enter the initialization values of the associated variables.
4. Click **OK** to confirm.

Comments

A comment can be entered in this column. It may contain any characters or special characters.

Sorting in the declaration tables

You can specify a sorting order for a column in the declaration tables. The lines of the declaration table are arranged so that the entries (character strings) of the relevant column are sorted according to the sorting criterion. The characters are sorted according to their ANSI code; the sorting is not case-sensitive.

The following sorting criteria are available:

- **Sort in ascending order**
  Alphabetic sorting is ascending order (0 ... 9, a ...z).
- **Sort in descending order**
  Alphabetic sorting is descending order (z ... a, 9 ... 0).
- **Original order**
  No sorting. The lines of the declaration table are arranged in the order of declaration.

As long as the sorting order is not accepted (see below), the following applies:

- The sorting only affects the display. The data storage remains unchanged.
- The declaration table is saved in the original order.
- The compiler processes the declaration table in the original order.

Special features when sorting structures and enumerations

When sorting structures (Page 5166) and enumerations (Page 5167), the following applies:

- When sorting according to the **Structure name** or **Enumeration name** column:
  - The structures or enumerations are sorted according to their names.
  - The elements of these structures and enumerations remain in their original order.
- When sorting according to the other columns:
  - The structures and enumerations remain in their original order.
  - The elements within the structures and enumerations are sorted.

Special feature of pragma lines

For the declaration tables of a unit (interface section and implementation section), the following applies:

If in the declaration of unit variables (Page 5172) (Parameters tab), pragma lines (Page 5178) are available, the lines are sorted between the individual pragma lines.

Note

In the sorted view of a declaration table, the following is not possible:

- Insertion of new elements (e.g. variables, structures, enumerations as well as elements of structures and enumerations).
- Insertion of new or copied lines.
- Insertion of pragma lines.

Procedure

To sort the entries of the declaration table:

- Double-click the appropriate column header.
  The sorting changes between the sorting criteria.
  A double-click in another column header results sorting in ascending order according to this column.
- Or alternatively:
  - Move the cursor to the appropriate column header.
  - Select the sorting criterion in the context menu

  The sorting is performed according to the selected sorting criterion.
  The original order can only be selected in the context menu of a sorted column.

Accepting the sorting order

Proceed as follows:

1. Move the cursor to an arbitrary column header.
2. Select **Accept sorting order** in the context menu.

The sorting order is taken over as original order.

Note

Observe the following when accepting the sorting order in a declaration table:

- No "Undo" possible.
- The data storage and the declaration order are changed.
- The corresponding unit is changed and must be compiled again.
- The compiler processes the declaration table in the changed order.
- The relevant variable blocks are initialized during the download of the unit.
- HMI-relevant data is no longer consistent.

7.3.4.13 Data Types

A data type is used to determine how the value of a variable or constant in a program source is to be used.

The following data types are available to the user:

- Elementary data types (Page 5162)
- User-defined data types (UDT) (Page 5165)
  - Enumerations
  - Structures (Struct)
- Technology object data types (Page 5168)
- System data types (Page 5169)

Elementary data types

Elementary data types define the structure of data that cannot be broken down into smaller units. An elementary data type describes a memory area with a fixed length and stands for bit data, integers, floating-point numbers, duration, time, date and character strings.

All the elementary data types are listed in the table below:

Table 7-529 Bit widths and value ranges of the elementary data types

Type		Reserv. word	Bit width	Range of values
Bit data type Data of this type uses either 1 bit, 8 bits, 16 bits, or 32 bits. The initialization value of a variable of this data type is 0.				
	Bit	BOOL	1	0, 1 or FALSE, TRUE
	Byte	BYTE	8	16#0 to 16#FF
	Word	WORD	16	16#0 to 16#FFFF
	Double word	DWORD	32	16#0 to 16#FFFF_FFFF
Numeric types These data types are available for processing numeric values. The initialization value of a variable of this data type is 0 (all integers) or 0.0 (all floating-point numbers).				
	Short integer	SINT	8	-128 to 127 (-2**7 to 2**7-1)
	Unsigned short integer	USINT	8	0 to 255 (0 to 2**8-1)
	Integer	INT	16	-32_768 to 32_767 (-2**15 to 2**15-1)
	Unsigned integer	UINT	16	0 to 65_535 (0 to 2**16-1)
	Double integer	DINT	32	-2_147_483_648 to 2_147_483_647 (-2**31 to 2**31-1)
	Unsigned double integer	UDINT	32	0 to 4_294_967_295 (0 to 2**32-1)
	Floating-point number (per IEEE -754)	REAL	32	-3.402_823_466E+38 to -1.175_494_351E-38, 0.0, +1.175_494_351E-38 to +3.402_823_466E+38 Accuracy: 23-bit mantissa (corresponds to 6 decimal places), 8-bit exponent, 1-bit sign.
	Long floating-point number (in accordance with IEEE-754)	LREAL	64	-1.797_693_134_862_315_7E+308 to -2.225_073_858_507_201_4E-308, 0.0, +2.225_073_858_507_201_4E-308 to +1.797_693_134_862_315_7E+308 Accuracy: 52-bit mantissa (corresponds to 15 decimal places), 11-bit exponent, 1-bit sign.
Time types These data types are used to represent various date and time values.				

Type		Reserv. word	Bit width	Range of values
	Duration in increments of 1 ms	TIME	32	T#0d_0h_0m_0s_0ms to T#49d_17h_2m_47s_295ms Maximum of 2 digits for the values day, hour, minute, second and a maximum of 3 digits for milliseconds Initialization with T#0d_0h_0m_0s_0ms
	Date in increments of 1 day	DATE	32	D#1992-01-01 to D#2200-12-31 Leap years are taken into account, year has four digits, month and day are two digits each Initialization with D#0001-01-01
	Time of day in increments of 1 ms	TIME_OF_DAY (TOD)	32	TOD#0:0:0.0 to TOD#23:59:59.999 Maximum of two digits for the values hour, minute, second and maximum of three digits for milliseconds Initialization with TOD#0:0:0.0
	Date and time	DATE_AND_TIME (DT)	64	DT#1992-01-01-0:0:0.0 to DT#2200-12-31-23:59:59.999 DATE_AND_TIME consists of the data types DATE and TIME Initialization with DT#0001-01-01-0:0:0.0
String type Data of this type represents character strings, in which each character is encoded with the specified number of bytes. The length of the string can be defined at the declaration. Indicate the length in "[" and "]", e.g. STRING[100]. The default setting consists of 80 characters. The number of assigned (initialized) characters can be less than the declared length.				
	String with 1 byte/character	STRING	8	You may use all the characters in the extended ASCII character set (ASCII code $00 to $FF). Default ' ' (empty string)

Note

During variable export to other systems, the value ranges of the corresponding data types in the target system must be taken into account.

See also

Value range limits of elementary data types (Page 5163)

General data types (Page 5164)

Elementary system data types (Page 5165)

Value range limits of elementary data types

The value range limits of certain elementary data types are available as constants.

Table 7-530 Symbolic constants for the value range limits of elementary data types

Symbolic constant	Data type	Value	Hex notation
SINT#MIN	SINT	-128	16#80
SINT#MAX	SINT	127	16#7F
INT#MIN	INT	-32768	16#8000

Symbolic constant	Data type	Value	Hex notation
INT#MAX	INT	32767	16#7FFF
DINT#MIN	DINT	-2147483648	16#8000_0000
DINT#MAX	DINT	2147483647	16#7FFF_FFFF
USINT#MIN	USINT	0	16#00
USINT#MAX	USINT	255	16#FF
UINT#MIN	UINT	0	16#0000
UINT#MAX	UINT	65535	16#FFFF
UDINT#MIN	UDINT	0	16#0000_0000
UDINT#MAX	UDINT	4294967295	16#FFFF_FFFF
T#MIN TIME#MIN	TIME	T#0ms	16#0000_0000[1]
T#MAX TIME#MAX	TIME	T#49d_17h_2m_47s_295ms	16#FFFF_FFFF[1]
TOD#MIN TIME_OF_DAY#MIN	TOD	TOD#00:00:00.000	16#0000_0000[1]
TOD#MAX TIME_OF_DAY#MAX	TOD	TOD#23:59:59.999	16#0526_5BFF[1]
[1] Internal display only			

General data types

General data types are often used for the input and output parameters of system functions and system function blocks. The subroutine can be called with variables of each data type that is contained in the general data type.

The following table lists the available general data types:

Table 7-531 General data types

General data type	Data types contained
ANY_BIT	BOOL, BYTE, WORD, DWORD
ANY_INT	SINT, INT, DINT, USINT, UINT, UDINT
ANY_REAL	REAL, LREAL
ANY_NUM	ANY_INT, ANY_REAL
ANY_DATE	DATE, TIME_OF_DAY (TOD), DATE_AND_TIME (DT)
ANY_ELEMENTARY	ANY_BIT, ANY_NUM, ANY_DATE, TIME, STRING
ANY	ANY_ELEMENTARY, user-defined data types (UDT), system data types, data types of the technology objects

Note

You **cannot** use general data types as type identifiers in variable or type declarations.

The general data type is retained when a user-defined data type (UDT) is derived directly from an elementary data type (only possible with the SIMOTION ST programming language).

Elementary system data types

In the SIMOTION system, the data types specified in the table are treated in a similar way to the elementary data types. They are used with many system functions.

Table 7-532 Elementary system data types and their use

Identifier	Bit width	Use
StructAlarmId	32	Data type of the alarmId for the project-wide unique identification of the messages. The alarmId is used for the message generation. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTALARMID#NIL
StructTaskId	32	Data type of the taskId for the project-wide unique identification of the tasks in the execution system. Please refer to the *SIMOTION Basic Functions* Function Manual. Initialization with STRUCTTASKID#NIL

Table 7-533 Symbolic constants for invalid values of elementary system data types

Symbolic constant	Data type	Meaning
STRUCTALARMID#NIL	StructAlarmId	Invalid AlarmId
STRUCTTASKID#NIL	StructTaskId	Invalid TaskId

User-defined data types

Defining user-defined data types (UDT)

You can create user-defined data types in units and programs/charts:

- Structures (Page 5166)
- Enumerations (Page 5167)

The scope of the data type declaration (Page 5166) depends on the location of the declaration.

Scope of the data type declaration

You create derived data types in the declaration tables of the source file or the program/chart. The scope of the data type declaration depends on the location of the declaration.

- In the declaration table of the unit, **Interface (exported declaration)** section:
  The data type is valid for the entire source file; all programs/charts (programs, function blocks, and functions) within the source file can access the data type.
  These variables are also available, if appropriately connected (see Define connections (Page 5223)), in other source files (or other units).
- In the declaration table of the unit, **Implementation (unit-internal declaration)** section:
  The data type is valid in the source file; all programs/charts (programs, function blocks, and functions) within the source file can access the data type.
- In the declaration table of the program/chart:
  The data type can only be accessed within the program/chart in which it is declared.

Defining structures

You define structures in the declaration tables of the unit or the program/chart. The scope (Page 5166) of the structures depends on the location of the declaration.

To define structures, proceed as follows:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Structures** tab.
3. Enter the name of the structure.
4. In the same line, enter:
   - The name of the first component in the **Element name** field.
   - Data type of the component.
     You can select already defined data types.
     See also:
     Elementary data types (Page 5162).
     User-defined data types (Page 5165).
     Technology object data types (Page 5168).
     System data types (Page 5169).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 5155).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 5156).
   - Optional initial value (initialization value).
     See also: Initial value (Page 5157).
   - Optional comment.
     See also: Comment (Page 5159).

5. Enter additional elements of the structure in the following lines; leave the **Structure name** field empty.
6. Begin the definition of the new structure by entering a new name in the **Structure name** field.

Example

This example shows the definition of a structure with three components:

	Structure name	Element name	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	t_struct	comp_1	INT	☐			
2		comp_2	REAL	☐	5	3(2), 2(3)	
3		compp_3	STRING	☐		'Example'	
4				☐			

Figure 7-482 Definition of a structure

Defining enumerations

You define **enumerations** in the declaration tables of the unit or the program/chart. The scope (Page 5166) of the enumerations depends on the location of the declaration.

To define enumerations, proceed as follows:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Enumerations** tab.
3. Enter the name of the enumeration.
4. In the same line, enter:
   - The name of the first element
   - Optionally, the initialization value of the enumeration data type
5. Enter additional elements of the enumeration in the following lines; leave the **Enumeration name** field empty.
6. You begin the definition of the new enumeration by entering a new name in the **Enumeration name** field.

Example

This example shows the definition of an enumeration data type with the name **Color** and the enumeration elements **Red**, **Blue**, and **Green**, as well as the initialization value (initial value) **Green**.

If no initialization value is entered during the enumeration definition (data type declaration), the first value of the enumeration is assigned to the data type. In this example, this means **Red** would be used for the initialization because it is defined as the first enumeration element.

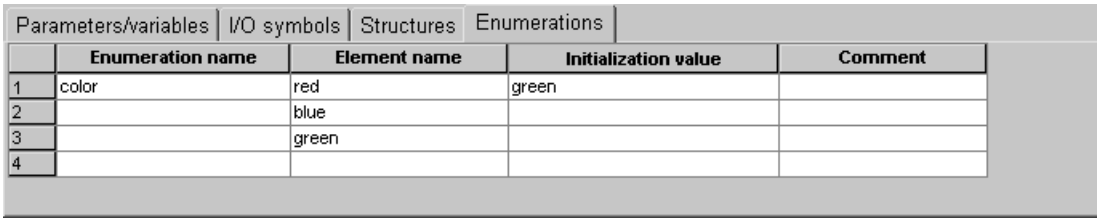


Figure 7-483 Definition of an enumeration data type

Technology object data types

Description of the technology object data types

You can declare variables with the data type of a technology object (TO). The following table shows the data types for the available technology objects in the individual technology packages.

For example, you can declare a variable with the data type *posaxis* and assign it an appropriate instance of a position axis. Such a variable is often referred to as a reference.

Table 7-534 Data types of technology objects (TO data type)

Technology object	Data type	Contained in the technology package
Drive axis	DriveAxis	CAM, PATH[1], CAM_EXT
External encoder	ExternalEncoderType	CAM, PATH[1], CAM_EXT
Measuring input	MeasuringInputType	CAM, PATH[1], CAM_EXT
Output cam	OutputCamType	CAM, PATH[1], CAM_EXT
Cam track	_CamTrackType	CAM, PATH[1], CAM_EXT
Position axis	PosAxis	CAM, PATH[1], CAM_EXT
Following axis	FollowingAxis	CAM, PATH[1], CAM_EXT
Following object	FollowingObjectType	CAM, PATH[1], CAM_EXT
Cam	CamType	CAM, PATH[1], CAM_EXT
Path axis[1]	_PathAxis	PATH[1], CAM_EXT
Path object[1]	_PathObjectType	PATH[1], CAM_EXT
Fixed gear	_FixedGearType	CAM_EXT
Addition object	_AdditionObjectType	CAM_EXT
Formula object	_FormulaObjectType	CAM_EXT
Sensor	_SensorType	CAM_EXT
Controller object	_ControllerObjectType	CAM_EXT
Temperature channel	TemperatureControllerType	TControl
General data type, to which every TO can be assigned	ANYOBJECT	

[1] Available as of version V4.1.

You can access the elements of technology objects (configuration data and system variables) via structures (see SIMOTION Basic Functions Function Manual).

Table 7-535 Symbolic constants for invalid values of technology object data types

Symbolic constant	Data type	Meaning
TO#NIL	ANYOBJECT	Invalid technology object

See also

Inheritance of the properties for axes (Page 5169)

Inheritance of the properties for axes

Inheritance for axes means that all of the data types, system variables and functions of the TO driveAxis are fully included in the TO positionAxis. Similarly, the position axis is fully included in the TO synchronizedAxis, the following axis in the TO pathAxis. This has, for example, the following effects:

- If a function or a function block expects an input parameter of the driveAxis data type, you can also use a position axis or a synchronized axis or a path axis when calling.
- If a function or a function block expects an input parameter of the posAxis data type, you can also use a synchronized axis or a path axis when calling.

System data types

There are a number of system data types available that you can use without a previous declaration. And, each imported technology packages provides a library of system data types.

Additional system data types (primarily enumeration and STRUCT data types) can be found

- In parameters for the general standard functions (see SIMOTION Basic Functions Function Manual)
- In parameters for the general standard function blocks (see SIMOTION Basic Functions Function Manual)
- In system variables of the SIMOTION devices (see relevant parameter manuals)
- In parameters for the system functions of the SIMOTION devices (see relevant parameter manuals)
- In system variables and configuration data of the technology objects (see relevant parameter manuals)
- In parameters for the system functions of the technology objects (see relevant parameter manuals)

7.3.4.14 Variables

Variables are an important component of programming and provide structure to programs. They are placeholders which can be assigned values that can be accessed several times in the program.

Variables have:

- A specific initialization behavior and scope of validity
- A data type and operations which are defined for that data type

User and system variables are differentiated. User variables can be defined by the user. System variables are provided by the system.

Keywords for variable types

The various keywords for variable types are shown in the following table.

Description of keywords for variable types

Keyword	Description	Use
Global user variables (declared in the interface or implementation section of the unit[1])		
VAR_GLOBAL	Unit variable; can be accessed by all POUs within the source file. If the variable was declared in the interface section, it can be used in another source file once a connection has been defined in its declaration table (see Define connections (Page 5223)).	FB, FC, program
VAR_GLOBAL RETAIN	Retentive unit variable; retained during power outage.	FB, FC, program
VAR_GLOBAL CONSTANT	Unit constant; cannot be changed from the program.	FB, FC, program
Local user variables (declared within a POU[2])		
VAR	Local variable (static for FB and program, temporary for FC)	FB, FC, program
VAR_TEMP	Temporary local variable	FB, FC, program
VAR_INPUT	Input parameters: Local variable; value is supplied from external source and can only be read in the FB or FC.	FB, FC
VAR_OUTPUT	Output parameters: Local variable; value is sent to an external destination by the FB. It can be read as an instance variable after being called by the FB (FB instance name.variable name).	FB, FC
VAR_IN_OUT	In/out parameter; the FB or FC accesses this variable directly (by means of a reference) and can change it directly.	FB, FC
VAR OVERRIDE	Static variable whose initialization value can be overwritten for an instance formation. The compiler option (Page 5111) "Permit object-oriented programming" must be activated.	FB
VAR CONSTANT	Local constant; cannot be changed from the program.	FB, FC, program

[1] MCC and LAD/FBD programming languages: in the declaration table of the associated source file.

[2] MCC and LAD/FBD programming languages: in the declaration table of the associated chart/program.

Defining variables

Variables are defined in the symbol browser or in the declaration table of the source file or chart/program. The following table provides an overview of where the relevant variable is defined.

Definition of variables

Variable type	Defined in...
Global device user variables	Symbol browser
unit variable	Declaration table of the source file as VAR_GLOBAL, VAR_GLOBAL RETAIN or VAR_GLOBAL CONSTANT
Local variable	Declaration table of the program/chart as: • VAR, VAR_TEMP, or VAR CONSTANT • Additionally for function blocks as VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT • Additionally for functions as VAR_INPUT, VAR_IN_OUT
I/O variable	Symbol browser
Symbolic access to the fixed process image of the BackgroundTask	• Declaration table of the source file • Declaration table of the program/chart (programs and FB only)

Use of global device variables

Global device variables are user-defined variables that you can access from all program sources (e.g. ST source files, MCC source files) of a SIMOTION device.

Global device variables are created in the symbol browser tab of the detail view; to do this, you must be working in offline mode.

Here is a brief overview of the procedure:

1. In the project navigator of SIMOTION SCOUT, select the **GLOBAL DEVICE VARIABLES** element in the SIMOTION device subtree.
2. In the detail view, select the **Symbol browser** tab and scroll down to the end of the variable table (empty row).
3. In the last (empty) row of the table, enter or select the following:
   - **Name** of variable
   - **Data type** of variable (only elementary data types are permitted)
4. Optionally, you can make the following entries:
   - Activation of **Retain** checkbox (This declares the variable as retentive, so that its value will be retained after a power failure.)
   - **Field length** (array size)
   - **Initial value** (if array, for each element)
   - **Display format** (if array, for each element)

You can now access this variable using the symbol browser or any program of the SIMOTION device.

In ST source files, you can use a global device variable, just like any other variable.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the global device variable with *_device.var-name*.

An alternative to global device variables is the declaration of unit variables in a separate unit, which is imported into other units. This has the following advantages:

1. Variable structures can be used.
2. The initialization of the variables during the STOP-RUN transition is possible (via Program in StartupTask).
3. For newly created global unit variables, a download in RUN is also possible.

Please refer to the SIMOTION Basic Functions Function Manual.

Declaring a unit variable in the source file

The unit variable is declared in the unit. The scope of validity of the variable is dependent on the section of the declaration table in which the variable is declared:

- In the interface section of the declaration table (INTERFACE):
  The unit variable is valid for the entire unit; all programs/charts (programs, function blocks, and functions) within the unit can access the unit variable. It is exported and can be used in other source files or units (e.g. ST source files, MCC units, LAD/FBD units) after a connection has been defined (Page 5223). It is also available on HMI devices as standard. The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.
- In the implementation section of the declaration table (IMPLEMENTATION):
  The unit variable is valid in the unit only; all programs/charts (programs, function blocks, and functions) within the unit can access the unit variable.

If you insert pragma lines (Page 5178) into the declaration table, you can split the unit variables into data blocks with a separate version code (Page 5189). You can also use the pragma lines to define a separate initialization behavior (Page 5184) for each of these data blocks and change the defaults for HMI export (Page 5190).

Proceed as follows; the unit (declaration table) is open, see Open existing program source (Page 5104):

1. In the declaration table, select the section for the desired scope.
2. Then select the **Parameters** tab.
3. Enter:
   - Name of the variable.
   - Variable type.
     See also: Keywords for variable types (Page 5170).
   - Data type of the variables.
     You can select already defined data types.
     See also: Data types (Page 5161).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 5155).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 5156).
   - Optional initial value (initialization value).
     See also: Initial value (Page 5157).
   - Optional comment.
     See also: Comment (Page 5159).

The variable is then declared and can be used immediately within the unit.

Note

Outside the unit (e.g. in the symbol browser), the variable is only available after the unit has been compiled.

INTERFACE (exported declaration)

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	var_in1	VAR_GLOBAL	REAL	☐			
2				☐			

IMPLEMENTATION (source-internal declaration)

Parameter | I/O symbols | Structures | Enumerations | Connections

	Name	Variable type	Data type	Reference	Array length	Initial value	Comment [English (United States)]
1	var_in2	VAR_GLOBAL	BOOL	☐			
2	var_out	VAR_GLOBAL	INT	☐	10		
3				☐			

Figure 7-484 Example: Declaring a unit variable in the unit

Note

The declaration table of the unit is read each time parameters are assigned for a command. Inconsistent data within the declaration table may, therefore, cause unexpected error messages during parameter assignment.

Declaring local variables

A local variable can only be accessed within the program/chart (program, function, function block) in which it is declared.

We distinguish between the following:

- Static variables:
  Static variables retain their value over all passes of the unit section (block memory).
- Temporary variables:
  Temporary variables are initialized each time the unit section is called again.

See also: Initialization of local variables (Page 5185).

If you insert a pragma line (Page 5178) at the start of the declaration table, you can influence the initialization of static variables of programs (Page 5186).

Proceed as follows; the program/chart with the declaration table is open (see Open existing program source (Page 5104)):

1. In the declaration table, select the **Parameters/Variables** tab.
2. Enter:
   - Name of the variable.
   - Variable type for variables.
     See also: Keywords for variable types (Page 5170).
   - Data type of the variables.
     You can select already defined data types.
     See also: Data types (Page 5161).
   - Activating the optional **Reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 5155).
   - Optional array length (to define the array size).
     See also: Array length and array element (Page 5156).
   - Optional initial value (initialization value).
     See also: Initial value (Page 5157).
   - Optional comment.
     See also: Comment (Page 5159).

The variable is now declared and can be used immediately.

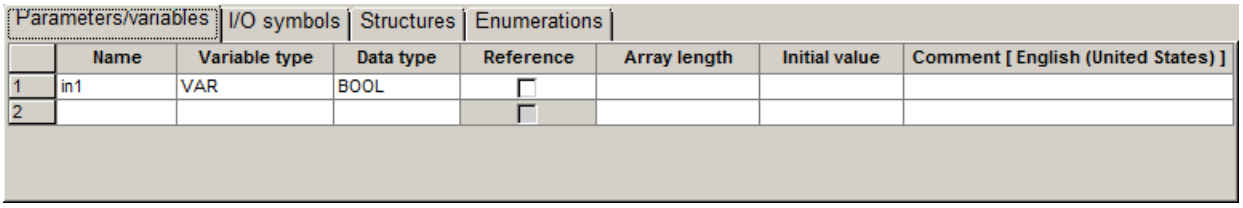


Figure 7-485 Example: Declaring a local variable in the chart/program

Note

The declaration table of the program/chart is read each time parameters are assigned for a command. Inconsistent data within the declaration table may, therefore, cause unexpected error messages during parameter assignment.

Defining variables in the Variable declaration dialog box ("on-the-fly" variable declaration)

As soon as you enter an unknown variable in the parameter screen form for an MCC command or an LAD/FBD graphic, the **Variable declaration** dialog box appears.

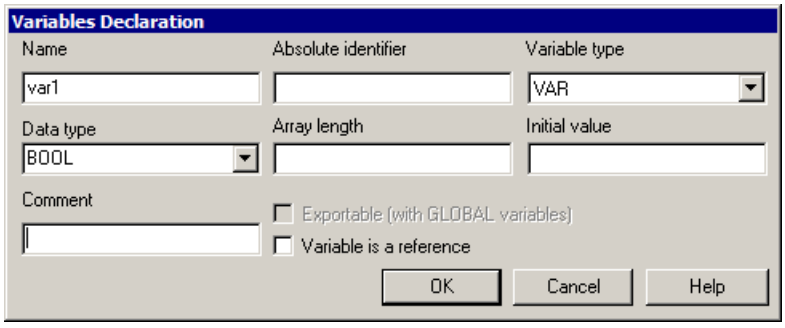


Figure 7-486 Variable declaration dialog box

Note

In order for the **Variable declaration** dialog box to appear, the **on-the-fly variable declaration** checkbox must be activated in the **Settings** dialog box.

Procedure

To define variables "on-the-fly" in the **Variable declaration** dialog box, proceed as follows:

1. Enter only the name of the variables in an input field of the parameter screen for an MCC command or LAD/FBD graphic and press the **Return** key.
   If the entered identifier is not a valid variable name, the dialog box **Variable declaration** appears.
2. Enter:
   - Another variable name, if required.
   - The **Data type** of the variables
     Available for selection are all elementary data types (Page 5162) and, where appropriate, the data type suitable for the input field.
     Select the data type or enter the identifier of a data type.
   - The **variable type**.
     Available for selection are all permitted keywords for variable types (Page 5170).
     Select the variable type.
     If you select a global variable type (e.g. VAR_GLOBAL), the checkbox **Exportable** becomes active.
   - The **Absolute identifier** for access to the fixed process image of the background task (Page 5211).
     You can only enter the identifier for the absolute access to the fixed process image of the background task if you have selected a data type that is included in the general data types (Page 5164) ANY_BIT or ANY_INT.
     Enter the absolute identifier according to the Syntax (Page 5218).
     The variable is displayed in the register **I/O symbols** of the declaration table.
   - Optional **array length**.
     Here, you specify the size of the array.
     Not available if an absolute identifier is entered.
     See also: Array length and array element (Page 5156).
   - Optional **initial value** (initialization value).
     Not available if an absolute identifier is entered.
     See also: Initial value (Page 5157).
   - Optional **comment**.
     See also:Comment (Page 5159).
   - Optionally, activate the **Variable is a reference** checkbox to declare a general reference to the data type.
     See also: Reference (Page 5155).

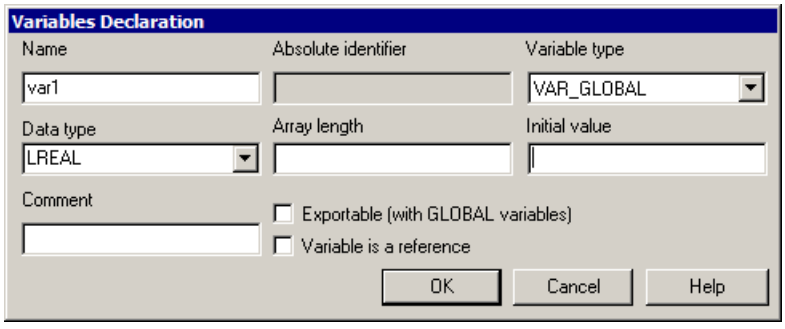


Figure 7-487 Example: Variable declaration

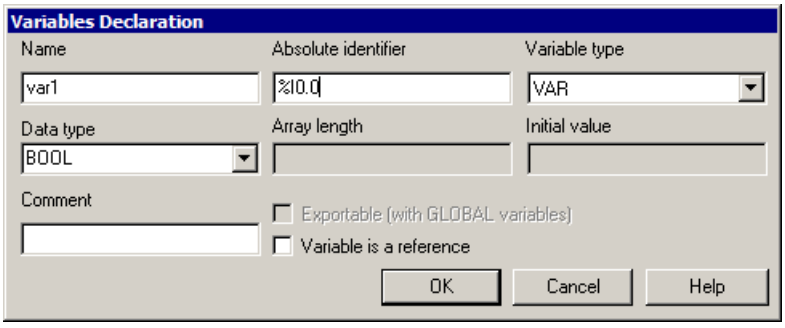


Figure 7-488 Example: Variable declaration (absolute name)

3. Confirm with **OK**.

Result

The variable is defined and entered in the declaration table of the source or the MCC chart or the LAD/FBD program, depending on the selected variable type and the **Exportable checkbox:**

- With local variables (e.g. VAR), the **Exportable** checkbox has a gray background and the new variable is entered in the declaration table of the MCC chart or the LAD/FBD program.
- With global variables (e.g. VAR_GLOBAL), the **Exportable** checkbox is shown activated:
  - If the **Exportable** checkbox is activated, the new variable is entered in the implementation section of the unit's declaration table.
  - If the **Exportable** checkbox is not activated, the new variable is entered in the interface section of the unit's declaration table.

Note

If you leave the **Variable declaration** dialog box by clicking **Cancel**, your input remains as it is, and the variable is not created.

Pasting pragma lines during variable definition

Pragma lines in declaration tables have the following function:

- In declaration tables of units (declaration of unit variables):
  Pragma lines in the interface section or implementation section of the declaration table split the variables of the respective section into different subsections:
  - The 1st subsection begins at the start of the relevant table and finishes at the 1st Pragma line.
  - All the other subsections start at one pragma line and finish at the next (or at the end of the table if starting at the last pragma line).

  Within these subsections of the table, each of the variables declared with VAR_GLOBAL or VAR_GLOBAL RETAIN forms a data block with a separate version code (Page 5189). If the version code is changed, only the data block concerned will be initialized during a download. Configure the pragma line in order to change the initialization behavior or the HMI export of the next data block (see following Section *Pragmas in the declaration tables of the unit variables*).
- In the declaration table of a program/chart (declaration of local variables):
  Line 1 may only contain one pragma line for changing the initialization behavior of programs' static variables (see the section titled *Pragmas in the declaration tables for local variables* later in this document).

Note

The SIMOTION Kernel version partially determines the effectiveness of pragma lines. This is specified for individual parameters.

Inserting a pragma line

To insert a pragma line into the declaration table, proceed as follows:

1. Select the **Parameters** tab (for unit variables) or **Parameters/variables** tab (for local variables).
2. In the declaration table, set the cursor to the number of the line at which a new data block is to be started.
   Only line 1 can be used in the declaration table for local variables.
3. Select the **Insert pragma line** context menu.
   A new line containing the **Pragmas** button only is inserted above the line.
4. Left-click the **Pragmas** button.
   A window containing configuration options for the relevant declaration table opens.
5. Enter the settings.
6. Confirm with **OK**.

Pragmas in the declaration tables for unit variables

A pragma line has been inserted into the declaration table for unit variables. If you click the "Pragmas" button in the table, the following window appears:

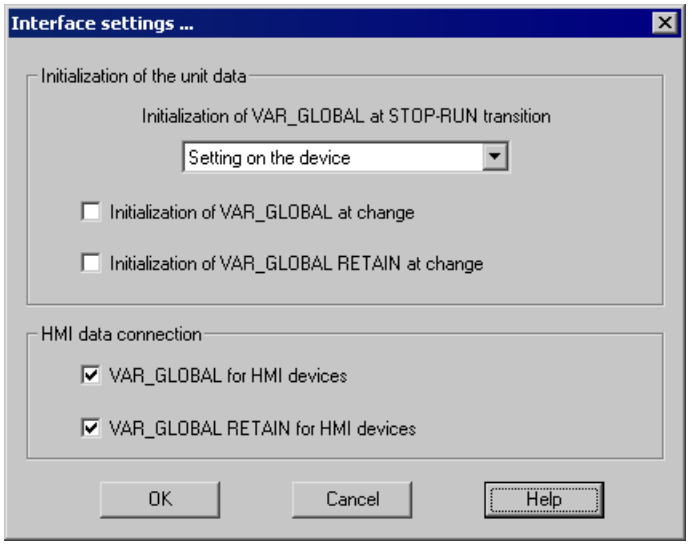


Figure 7-489 Pragma settings in the declaration table for unit variables (e.g. interface section)

This window is for making the settings below. These influence the data block following the pragma line:

Table 7-536 Parameters for pragma settings in the declaration table for unit variables

Parameter		Description
Initialization of unit data		
	Initialization of VAR_GLOBAL during STOP-RUN transition	Only as of version V4.1 of the SIMOTION Kernel. This setting determines whether the next data block of the non-retentive unit variables is initialized during the transition from STOP to RUN operating state. **Setting on the device** (standard): • As of version V4.2 of the SIMOTION Kernel: The setting made on the SIMOTION device applies. • Up to version V4.1 of the SIMOTION Kernel: Variables are **not** initialized. **Always**: Variables are initialized. **Never**: Variables are **not** initialized.
	Initialization of VAR_GLOBAL during a change	This setting determines whether a download for the next data block of the non-retentive unit variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
	Initialization of VAR_GLOBAL RETAIN during a change	This setting determines whether a download for the next data block of the retentive unit variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
HMI data connection		

Parameter		Description
		Use the settings below to change which unit variables are available on which HMI devices by default. Detailed description of the HMI export, in particular the effect of the settings depending on SIMOTION Kernel version: see Variables and HMI devices (Page 5190).
	VAR_GLOBAL for HMI devices	**Active** (standard in the interface section): The next data block of the non-retentive unit variables is available on HMI devices. **Inactive** (standard in the implementation section): The next data block of the non-retentive unit variables is **not** available on HMI devices.
	VAR_GLOBAL RETAIN for HMI devices	**Active** (standard in the interface section): The next data block of the retentive unit variables is available on HMI devices. **Inactive** (standard in the implementation section): The next data block of the retentive unit variables is **not** available on HMI devices.

Pragmas in the declaration tables for local variables

A pragma line has been inserted into line 1 of the declaration table for local variables. If you click the "Pragmas" button in the table, the following window appears:

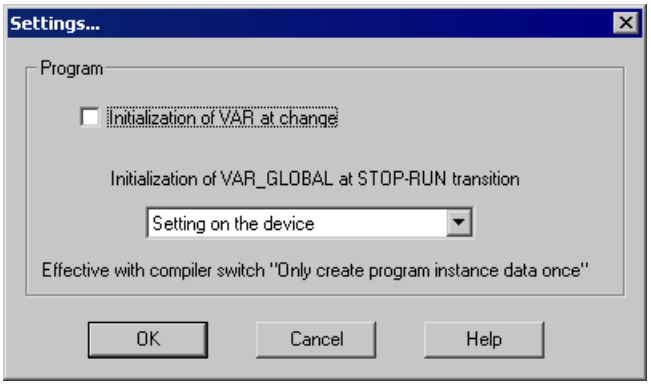


Figure 7-490 Pragma settings in the declaration table for local variables

You can make the following settings in this window:

Note

These settings only take effect in the following scenarios:

1. The "Only create program instance data once" compiler option is active on the higher-level unit.
2. The creation type of the program/chart is "Program".

Table 7-537 Parameters for pragma settings in the declaration table for local variables

Parameter	Description
Initialization of VAR during a change	This setting determines whether a download for the next static variables can be performed in RUN if the version code has changed. **Active**: A download in RUN is possible. **Inactive** (standard): A download in RUN is **not** possible.
Initialization of VAR during STOP-RUN transition	Only as of version V4.1 of the SIMOTION Kernel. This setting determines whether the next static variables are initialized during the transition from STOP to RUN operating state. **Setting on the device** (standard): • As of version V4.2 of the SIMOTION Kernel: The setting made on the SIMOTION device applies. • Up to version V4.1 of the SIMOTION Kernel: Variables are **not** initialized. **Always**: Variables are initialized. **Never**: Variables are **not** initialized.

Time of the variable initialization

The timing of the variable initialization is determined by:

- Memory area to which the variable is assigned
- Operator actions (e.g. source file download to the target system)
- Execution behavior of the task (sequential, cyclic) to which the program was assigned.

All variable types and the timing of their variable initialization are shown in the following tables. You will find basic information about tasks in the *SIMOTION Basic Functions* Function Manual.

The behavior for variable initialization during download can be set: To do this, as a default setting select the **Options > Settings** menu and the **Download** tab or define the setting during the current download.

Note

You can upload values of unit variables or global device variables from the SIMOTION device into SIMOTION SCOUT and save them in XML format.

1. Save the required data segments of the unit variables or global device variables as a data set with the function *_saveUnitDataSet*.
2. Use the **Save variables** function in SIMOTION SCOUT.

You can use the **Restore variables** function to download these data sets and variables back to the SIMOTION device.

For more information, refer to the SIMOTION SCOUT Configuration Manual.

This makes it possible, for example, to obtain this data, even if it is initialized by a project download or if it becomes unusable (e.g. due to a version change of SIMOTION SCOUT).

Initialization of retentive global variables

Retentive variables retain their last value after a loss of power. All other data is reinitialized when the device is switched on again.

Retentive global variables are initialized:

- When the backup or buffer for retentive data fails.
- When a memory reset (MRES) is performed.
- With the restart function (Del. SRAM) in SIMOTION P320 or P350.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the retentive data.
- When a download is performed according to the following description.
- With a firmware update (upgrade) or activation of a kernel in accordance with the following description.

Behavior during download

Table 7-538 Initializing retentive global variables during download

Variable type	Time of the variable initialization
Retentive global device variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive global device variables are initialized. • **No**[3]: The retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 5189).
Retentive unit variables	The behavior during the download depends on the *Initialization of retentive program data and retentive global device variables* setting[1]: • **Yes**[2]: All retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 5189).

[1] Default setting in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the retentive unit variables corresponds to a VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** and **SIMOTION LAD/FBD** programming languages:
A data block of the retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines (Page 5178) within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute has been specified within a pragma: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 5178) is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL RETAIN during a change*. All the variables declared with VAR_GLOBAL RETAIN up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see the SIMOTION Basic Functions Function Manual.

Behavior during upgrade or configuration change

When the SIMOTION device is upgraded to a new version of the SIMOTION Kernel or if the configuration is changed, the retentive variables are initialized as described below:

Table 7-539 Initialization of retentive global variables during upgrade or configuration change

Variable type	Time of the variable initialization
Retentive global device variables	This data is always initialized.
Retentive unit variables	This data can be retained. See section "Retaining retentive data" in the "Basic Functions for Modular Machines" Function Manual.

Initialization of non-retentive global variables

Non-retentive global variables lose their value during power outages. They are initialized:

- During the initialization of retentive global variables (Page 5182), e.g. during a firmware update or overall reset (MRES).
- During switch-on.
- By applying the *_resetUnitData* function, possible selectively for different data segments of the non-retentive data.
- During a download as described in the following table.
- During the transition from STOP to RUN mode as described at the end of the section.

Behavior during download

Table 7-540 Initializing non-retentive global variables during download

Variable type	Time of the variable initialization
Non-retentive global device variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive global device variables are initialized. • **No**[3]: The non-retentive global device variables are only initialized if their version code is changed. See: Version code of global variables and their initialization during download (Page 5189).
Non-retentive unit variables	The behavior during the download depends on the *Initialization of non-retentive program data and non-retentive global device variables* setting[1]: • **Yes**[2]: All non-retentive unit variables (all units) are initialized. • **No**[3]: A data block (= declaration block)[4] of the non-retentive unit variables in the interface or implementation section is only initialized[5] if its version code is changed. See: Version code of global variables and their initialization during download (Page 5189).

[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding check box is active.

[3] The corresponding check box is inactive.

[4] With the **SIMOTION ST** programming language:
A data block of the non-retentive unit variables corresponds to a VAR_GLOBAL/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL in the interface section or implementation section of the declaration table: Pragma lines (Page 5178) within a section of the declaration table separate the variables into different data blocks.

[5] Initialization of a changed data block also occurs during a download in RUN, provided the following condition is fulfilled:
With the **SIMOTION ST**
programming language: The following attribute has been specified within a pragma in the relevant declaration block: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 5178) has been pasted into the declaration table and the following check box is activated: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Behavior during STOP-RUN transition

The values of non-retentive global variables are retained by default during the transition from STOP to RUN mode.

You can, however, make a setting whereby the non-retentive global variables are initialized during the STOP-RUN transition:

- **As of Version V4.2 of the SIMOTION Kernel**, by activating the **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
  With non-retentive unit variables, this setting can be overwritten by a pragma or pragma line (Page 5178) in the relevant data blocks of the program sources.
- **As of Version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the relevant data blocks of the program sources (only with non-retentive unit variables):
  - With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the relevant VAR_GLOBAL/END_VAR declaration block: { BlockInit_OnDeviceRun := ALWAYS; }
  - With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    Paste a pragma line (Page 5178) with the following setting into the declaration table: "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block which is initialized during the STOP-RUN transition.

Note

With SIMOTION devices up to SIMOTION Kernel Version V4.0, non-retentive global variables are never initialized during the STOP-RUN transition.

Initialization of local variables

Local variables are initialized:

- For the initialization of retentive unit variables (Page 5182).
- For the initialization of non-retentive unit variables (Page 5184).
- Also, according to the following description:

Table 7-541 Initialization of local variables

Variable type	Time of the variable initialization
Local program variables	Local variables of programs are initialized differently: • Static variables (VAR) are initialized according to the memory area in which they are stored. See: Initialization of static program variables (Page 5186). • Temporary variables (VAR_TEMP) are initialized every time the program of the task is called.
Local variables of function blocks (FB)	Local variables of function blocks are initialized differently: • Static variables (VAR, VAR_IN, VAR_OUT) are only initialized when the FB instance is initialized. See: Initialization of instances of function blocks (FBs) (Page 5187). • Temporary variables (VAR_TEMP) are initialized every time the FB instance is called.
Local variables of functions (FC)	Local variables of functions are temporary and are initialized every time the function is called.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 5259) function.

Initialization of static program variables

The following versions affect the following static variables:

- Local variables of a unit program declared with VAR
- Function block instances declared with VAR within a unit program, including the associated static variables (VAR, VAR_INPUT, VAR_OUTPUT).

The initialization behavior is determined by the memory area in which the static variables are stored. This is determined by the "Only create program instance data once" (Page 5111) compiler option.

- For the deactivated "Only create program instance data once" compiler option (default):
  The static variables are stored in the user memory of each task which is assigned to the program.
  The initialization of the variables thus depends on the execution behavior of the task to which the program is assigned (see SIMOTION Basic Functions Function Manual):
  - Sequential tasks (MotionTasks, UserInterruptTasks, SystemInterruptTasks, StartupTask, ShutdownTask): The static variables are initialized every time the task is started.
  - Cyclic tasks (BackgroundTask, SynchronousTasks, TimerInterruptTasks): The static variables are only initialized only during the transition from STOP to RUN operating state.
- For the activated "Only create program instance data once" compiler option:
  This setting is necessary, for example, if a program is to be called within a program.
  The static variables of all programs from the program source (unit) involved are only stored once in the user memory of the unit.
  They are thus initialized together with the non-retentive unit variables, see Initialization of non-retentive global variables (Page 5184).
  They are not initialized by default during the transition from STOP to RUN operating state.
  You can, however, make a setting whereby they are initialized during the STOP-RUN transition:
  - **As of version V4.2 of the SIMOTION Kernel**, by activating the **Initialization of non-retentive global variables and program data during STOP-RUN transition** checkbox on the SIMOTION device.
    This setting can be overwritten by a pragma or pragma line (Page 5178) in the data block of the relevant program organization unit (POU).
  - **As of version V4.1 of the SIMOTION Kernel**, by a pragma or pragma line in the data block of the relevant program organization unit (POU):
    With the **SIMOTION ST** programming language:
    Specify the following attribute within a pragma in the VAR/END_VAR declaration block:
    { BlockInit_OnDeviceRun := ALWAYS; }
    With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
    The declaration table starts with a pragma line (Page 5178) containing the following setting: "*Initialization during STOP-RUN transition* = **Always**". All the variables declared with VAR in the table are initialized during the STOP-RUN transition.

Initialization of instances of function blocks (FBs) or classes

The initialization of a function block instance (Page 5240) or a class instance (as of version V4.5 of the SIMOTION Kernel) is determined by the location of its declaration:

- Global declaration (within VAR_GLOBAL/END_VAR in the interface of implementation section):
  Initialization as for a non-retentive unit variable, see Initialization of non-retentive global variables (Page 5184).
- Local declaration in a program (within VAR / END_VAR):
  Initialization as for static variables of programs, see Initialization of static variables of programs (Page 5186).

- Local declaration in a function block (within VAR / END_VAR):
  Initialization as for an instance of this function block.
- Declaration as in/out parameter in a function block or a function (within VAR_IN_OUT / END_VAR):
  For the initialization of the POU, only the reference (pointer) will be initialized with the instance of the function block remaining unchanged.

Note

You can obtain information about the memory requirements of a POU in the local data stack using the Program Structure (Page 5259) function.

Initialization of system variables of technology objects

The system variables of a technology object are usually not retentive. Depending on the technology object, a few system variables are stored in the retentive memory area (e.g. absolute encoder calibration).

The initialization behavior (except in the case of download) is the same as for retentive and non-retentive global variables. See Initialization of retentive global variables (Page 5182) and Initialization of non-retentive global variables (Page 5184).

The behavior during the download is shown below for:

- Non-retentive system variables
- Retentive system variables

Table 7-542 Initializing technology object system variables during download

Variable type	Time of the variable initialization
Non-retentive system variables	Behavior during download, depending on the *Initialization of all non-retentive data for technology objects* setting[1]: • **Yes**[2]: All technology objects are initialized. – All technology objects are restructured and all non-retentive system variables are initialized. – All technological alarms are cleared. • **No**[3]: Only technology objects changed in SIMOTION SCOUT are initialized. – The technology objects in question are restructured and all non-retentive system variables are initialized. – All alarms that are pending on the relevant technology objects are cleared. – If an alarm that can only be acknowledged with *Power On* is pending on a technology object that will not be initialized, the download is aborted.
Retentive system variables	Only if a technology object was changed in SIMOTION SCOUT, will its retentive system variables be initialized. The retentive system variables of all other technology objects are retained (e.g. absolute encoder calibration).

[1] Default in the **Options > Settings** menu, **Download** tab, or the current setting for the download.

[2] The corresponding checkbox is active.

[3] The corresponding checkbox is inactive.

Version ID of global variables and their initialization during download

Table 7-543 Version code of global variables and their initialization during download

Data segment		Description of version code
Global device variables		
	Retentive global device variables Non-retentive global device variables	• Separate version code for each data segment of the global device variables. • The version code of the data segment changes for: – Add or remove a variable within the data segment – Change of the identifier or the data type of a variable within the data segment • This version code does not change on: – Changes in the other data segment – Changes to initialization values[1] • During downloading[2], the rule is: This data segment is only initialized when the version code of a data segment is changed.
Unit variables of a unit		
	Retentive unit variables in the interface section Retentive unit variables in the implementation section Non-retentive unit variables in the interface section Non-retentive unit variables in the implementation section	• Several data blocks (= declaration blocks)[3] are possible in each data segment. • **Separate version code** for each data block. • The version code of the data block changes for: – Add or remove a variable in the associated declaration block – Change of variable sequence in the relevant declaration block – Change of the identifier or the data type of a variable in the associated declaration block – Change of a data type definition (from a separate or imported[4] unit) used in the associated declaration block – Add or remove declaration blocks within the same data segment **before** the associated declaration block • This version code does not change on: – Add or remove declaration blocks in other data segments – Add or remove declaration blocks within the same data segment **after** the associated declaration block – Changes in other data blocks – Changes to initialization values[1] – Changes to data type definitions that are not used in the associated data block – Changes to functions • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
Retentive variables of function blocks and classes (The instances are declared as non-retentive unit variables)		

Data segment		Description of version code
	Non-retentive unit variables in the interface section	Only as of version V4.5 of the SIMOTION Kernel. • The retentive variables of all instances within a declaration block (VAR_GLOBAL / END_VAR) are summarized as a separate retentive data block • to which a separate version code is assigned. • The version code of this retentive data block changes: – The data structure of the retentive local variables for the instances within the declaration block changes. – The sequence of the declaration blocks (VAR_GLOBAL / END_VAR) within the program source changes. • During downloading[2], the rule is: This data block is only initialized when the version code of a data block is changed[5]. • Functions for data backup and initialization take into account the version code of the data blocks.
	Non-retentive unit variables in the implementation section	

[1] Changed initialization values are not effective until the data block or data segment in question is initialized.

[2] If *Initialization of retentive program data and retentive global device variables* = No and *Initialization of non-retentive program data and non-retentive global device variables* = No.
In the case of other settings: See the sections "Initialization of retentive global variables (Page 5182)" and "Initialization of non-retentive global variables (Page 5184)".

[3] With the **SIMOTION ST** programming language:
A data block corresponds to a VAR_GLOBAL/END_VAR or VAR_GLOBAL RETAIN/END_VAR declaration block in the interface section or implementation section.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A data block of the non-retentive unit variables is formed as follows from the variables declared with VAR_GLOBAL or VAR_GLOBAL RETAIN in the interface section or implementation section of the declaration table: Pragma lines (Page 5178) within a section of the declaration table separate the variables into different data blocks.

[4] The use of units depends on the programming language, refer to the relevant section (Page 5223).

[5] Initialization of a changed data block also occurs during a download in RUN mode, provided that the following condition is fulfilled:
With the **SIMOTION ST** programming language
The following attribute has been specified within a pragma: { BlockInit_OnChange := TRUE; }.
With the **SIMOTION MCC** or **SIMOTION LAD/FBD** programming languages:
A pragma line (Page 5178) is inserted in the declaration table with the following check box enabled in this line: *Initialization of VAR_GLOBAL during a change*. All the variables declared with VAR_GLOBAL up to the next pragma line or the end of the table form a data block accordingly.
For information on the general conditions for a download in RUN, see SIMOTION Basic Functions Function Manual.

Variables and HMI devices

Exported variables

The following variables are exported to HMI devices where they are available:

- System variables of the SIMOTION device
- System variables of technology objects
- I/O variables
- Global device variables

- Retentive and non-retentive unit variables of the interface section (default setting).
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line (Page 5178), if you deactivate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
- Non-retentive static variables (VAR) of programs provided that the compiler option "Only create program instance data once" is activated
- Non-retentive static variables (VAR) of function blocks
  This is the default setting when the compiler option "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Non-retentive public variables (VAR PUBLIC) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := FALSE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are not exported to HMI devices. The HMI consistency check is also omitted for them during the download.

Note

The total size of the unit variables that can be exported to HMI devices is limited to 64 KB per unit.

The effect of the `{ HMI_Export := FALSE; }` / `{ HMI_Export := TRUE; }` pragma (or the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* settings in a pragma line) depends on the SIMOTION Kernel version:

- As of Version V4.1 of the SIMOTION Kernel:
  The pragma affects the export of the corresponding declaration block to HMI devices **and** the structure of the HMI address space:
  - Only those variables in declaration blocks exported to HMI devices occupy the HMI address space.
  - Within the HMI address space, the variables are arranged according to order of their declaration.
- Up to Version V4.0 of the SIMOTION Kernel:
  The pragma affects **only** the export of the corresponding declaration block to HMI devices. The HMI address space is also occupied by unit variables of the interface section whose declaration blocks are not assigned to HMI devices.
  Within the HMI address space, the variables are sorted in the following order:
  - Retentive unit variables of the interface section (exported and not exported).
  - Retentive unit variables of the implementation section (only exported).
  - Non-retentive unit variables of the interface section (exported and not exported).
  - Non-retentive unit variables of the implementation section (only exported).

  Within these segments, the variables are arranged according to order of their declaration.

Non-exported variables

The following variables are **not** exported to HMI devices and are **not** available there:

- Retentive and non-retentive unit variables of the implementation section (default setting)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`
    See also Controlling compiler with attributes.
  - In the SIMOTION MCC and SIMOTION LAD/FBD programming languages:
    For the variable declarations following a pragma line (Page 5178), if you activate the *VAR_GLOBAL for HMI devices* or *VAR_GLOBAL RETAIN for HMI devices* parameters in this pragma line.

  The unit variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Local variables (VAR, VAR_TEMP) of functions or methods
- Temporary variables (VAR_TEMP) of programs, function blocks or classes
- Retentive local variables (VAR RETAIN) of programs (as of version 4.5 of the SIMOTION Kernel)

- Retentive local variables (VAR RETAIN) of function blocks (as of version 4.5 of the SIMOTION Kernel)
  This is the default setting when the compiler option "Permit object-oriented programming" is enabled. Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
  The pragma is not evaluated if the compiler option "Permit object-oriented programming" is **not** activated. The export behavior cannot be changed.
- Retentive local variables (VAR RETAIN) of classes (default setting, as of version 4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.
- Non-retentive static variables (VAR) of programs provided that the compiler option "Only create program instance data once" is **not** activated
- Non-retentive protected or private variables (VAR, VAR PROTECTED, VAR PRIVATE) of classes (default setting, as of version V4.5 of the SIMOTION Kernel)
  Change this default as follows:
  - In the SIMOTION ST programming language:
    For each declaration block with the following pragma: `{ HMI_Export := TRUE; }`.
    See also Controlling compiler with attributes.

  The variables of a data block identified in this way are exported to HMI devices. Consequently, they undergo the HMI consistency check during downloading.

Example for the SIMOTION ST programming language

Table 7-544 Example for the control of the HMI export with the corresponding pragma

```
INTERFACE
    VAR_GLOBAL
        // HMI export
        x1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := FALSE; }
        // No HMI export
        x2 : DINT;
    END_VAR
    // ...
END_INTERFACE

IMPLEMENTATION
    VAR_GLOBAL
        // No HMI export
        y1 : DINT;
    END_VAR
    VAR_GLOBAL
        { HMI_Export := TRUE; }
        // HMI export
        y2 : DINT;
    END_VAR
    // ...
END_IMPLEMENTATION
```

7.3.4.15 General references (as of kernel V4.5)

Defining general references

General references contain the address assignment to a variable or an instance of a function block or class:

General references require a SIMOTION Kernel as of version V4.5. The compiler option (Page 5111) "Permit object-oriented programming" also must be activated.

References can be defined as variables or elements of structures.

To do this, activate the **Reference** checkbox in the **Parameters/variables** or **Structures** tab in the declaration table of the program source or program organization unit. The declaration is also possible as array element. See also the description for the "Reference (Page 5155)" checkbox.

This creates the "Reference to data type" (REF_TO *data_type*) that can contain the address to an existing data type (reference data type).

The following are permitted as data types to which references can be formed:

- Elementary data types (e.g. INT, DINT, REAL, WORD, TIME, STRING)
- User-defined data types (UDT)
- System data types

- Function blocks, provided they contain at least one static variable
- Classes (from ST sources)

No references can be formed to the following data types because references exist already:

- Technology object data types
- Object-oriented interfaces (from ST sources)
- General references
- I/O references (from ST sources)

Variables of general references can be declared in the interface and implementation sections as well as in all program organization units and classes. The data type to be referenced must be declared beforehand and lie within the scope of the POU. The declaration as element of a structure or an ARRAY is also possible.

All variable types for the relevant program organization unit are permitted with the exception of:

- VAR_IN_OUT.
  Arrays, structures, function blocks or classes that contain general references may nonetheless be transferred as VAR_IN_OUT.
- VAR RETAIN or VAR_GLOBAL RETAIN is not permitted.
  However, structures that contain at least one element other than reference types (general references, data types of the technology objects, object-oriented interfaces), can be stored retentive.
- VAR CONSTANT or VAR_GLOBAL CONSTANT is not meaningful.
  General references must be initialized with "NULL".

Forming general references

A reference to a variable is formed with the REF standard function (in := *var_name*). This function is described in detail in the "SIMOTION Basic Functions" Function Manual.

The REF function can be used on the following variables:

- Retentive and non-retentive unit variables or instances of classes or function blocks or their elements declared as such.
- In methods within classes or function blocks:
  - Static variables (VAR) of the higher-level class or function block (including instance variables for classes or function blocks).
  - The THIS keyword. Enables access to the instance variable of the higher-level class or function block.
- In function blocks:
  - Static variables (VAR) of the function block (including instance variables of classes or function blocks).
  - The THIS keyword. Permits access to the instance variable of the function block.

The variable must be able to be read and written.

Not permitted are:

- Variables of technology object data types
- Variables of object-oriented interfaces
- Variables of general references
- Retentive local variables within classes or function blocks
- I/O variables
- Global device variables
- Constants

The return value has the data type "Reference to the data type of the input parameter *in*". The return value can be assigned with the assignment operator ":=" to a variable that has REF_TO *type* data type. A data type conversion is not possible.

Note

Note for instances of function blocks that are passed as in/out parameter (VAR_IN_OUT):

- The REF function cannot be used for static variables (VAR) of function blocks imported from technology packages or device-independent libraries.

The restriction does not apply to class instances.

Operations with general references

Assignment operator ":="

For the assignment of a reference variable, the address of the variable rather than the variable value is passed. Consequently, the reference data types must be identical on both sides of the operator. No implicit data type conversion is possible.

The only exception is for references to classes. A reference to a derived class of a reference to the base class can be assigned here.

For the assignment, no check is made for the validity of the reference; the assignment is always possible.

An explicit assignment with NULL is also possible.

The validity of references should be checked by comparing with NULL before use.

Dereferencing with operator "^"

The reference content can be accessed with the postfix operator "^", e.g. *ref_var*^.

If the reference does not contain a valid value at runtime, the processing task is aborted and the ExecutionFaultTask called.

Before access, a check for validity by comparison with NULL is recommended.

Multilevel dereferencing is possible, e.g. reference to a structure that contains further references. Dereferenced variables can be used anywhere in expressions.

If a method is called via a reference to a class or function block, the class or function block must be dereferenced.

Dereferenced values cannot be used as actual parameters for the _releaseSemaphore() and _testandSetSemaphore() system functions.

Dynamic type conversion with the operator "?="

The operator "?=" can be used for dynamic type conversion only with references to classes and with interface variables (variables for object-oriented interfaces). It supports the following variants:

- Assignment between references to classes.
- Assignment of a reference to a class for an interface variable.
  Whereby, the reference to a class must be specified with the dereferencing operator "^".
- Assignment of an interface variable for a reference to a class.
- Assignment between interface variables

A dynamic type conversion is performed if the following two conditions are satisfied:

1. The variable to the right of the operator contains the reference to the instance of a class.
2. For the variable to the left of the operator, the following applies:
   - The variable is an interface variable:
     The object-oriented interface that this variable has as data type must implement the class on the right-hand side.
   - The variable is a reference to a class:
     This class corresponds to the class on the right-hand side or is a base class for this class.

In this case, the reference on the right-hand side is converted and transferred to the left-hand side.

If the type conversion fails, the variable on the left-hand side contains the "NULL" value.

Comparison operators

Only the comparison operators "=" and "<>" can be used to check for equality or inequality.

The operators ">", "<", ">=", "<=" as well as the MIN and MAX standard functions are not permitted.

7.3.4.16 Access to inputs and outputs (process image, I/O variables)

Overview of access to inputs and outputs

SIMOTION provides several possibilities to access the device inputs and outputs of the SIMOTION device as well as the central and distributed I/O:

- Via direct access with I/O variables
  Direct access is used to access the corresponding I/O address directly.
  Define an I/O variable (name and I/O address) without assigning a task to it. The entire address space of the SIMOTION device can be used.
  It is preferable to use direct access with sequential programming (in MotionTasks); access to current input and output values at a particular point in time is especially important in this case.
  Further information: Direct access and process image of the cyclic tasks (Page 5202).
- Via the process image of cyclic tasks using I/O variables
  The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming).
  Define an I/O variable (name and I/O address) and assign a task to it. The entire address range of the SIMOTION device can be used.
  Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.
  Further information: Direct access and process image of the cyclic tasks (Page 5202).
- Using the fixed process image of the BackgroundTask
  The process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device, on which a subset of the I/O address space of the SIMOTION device is mirrored. The mirror image is refreshed with the BackgroundTask and is consistent throughout the entire cycle. This process image is used preferentially when programming the BackgroundTask (cyclic programming).
  The address range 0 .. 63 can be used. Exception: I/O addresses that are accessed using the process image of the cyclic task can only be used with the setting “Common process image” (Page 5213) (as of Kernel V4.2).
  Further information: Access to the fixed process image of the BackgroundTask (Page 5211).

A comparison of the most important properties is contained in "Important properties of direct access and process image" (Page 5199).

You can use I/O variables like any other variable, see "Access I/O variables" (Page 5221).

Note

An access via the process image is more efficient than direct access.

Important features of direct access and process image access

Table 7-545 Important properties of direct access and process image access

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Permissible address range	Entire address range of the SIMOTION device **Exception**: I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously (example: PIW63 or PQD62 are not permitted).		Addresses 0 .. 63. **Exception**: Up to version V4.1 of the SIMOTION Kernel or the "Separate process image" setting, addresses used for the process image of the cyclic tasks are not permitted.
Address configuration	Necessary. The addresses used must be present in the I/O and appropriately configured. The "Rules for I/O addresses for direct access and the process image of the cyclic tasks" (Page 5205) must be observed.		Not necessary. Addresses that are not present in the I/O or have not been configured can also be used.
Assigned task	None.	Cyclic task for selection: • SynchronousTasks, • TimerInterruptTasks, • BackgroundTask.	BackgroundTask.
Memory area for process images	-	Depends on the SIMOTON Kernel version: • Up to version V4.1: Separate memory areas in all cases • As of version V4.2: Option to select a common memory area	

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Update	• Onboard I/O of SIMOTION devices C230-2, C240, and C240 PN: Update occurs in a cycle clock of 125 µs. • With I/O devices – via PROFIBUS DP or PROFINET on an isochronous SIMOTION device[1] – via DRIVE-CLiQ – Onboard I/O of SIMOTION D devices: The update is performed in the position control cycle clock[2]. • With I/O devices – via PROFIBUS DP or PROFINET on a **non**-isochronous SIMOTION device[1] – via P-Bus: The update is performed in the interpolation cycle[2 3]. Inputs are read at the start of the cycle clock. Outputs are written at the end of the cycle clock.	Update occurs with the assigned task: • Inputs are read before the assigned task is started and transferred to the process input image. • Process output image is written to the outputs after the assigned task has been completed.	An update is made with the *BackgroundTask*: • Inputs are read before the *BackgroundTask* is started and is transferred to the process input image. • Process output image is written to the outputs when the *BackgroundTask* is complete.
Consistency	–	During the entire cycle of the assigned task. **Exception**: Direct access to output occurs.	During the entire cycle of the *BackgroundTask*. **Exception**: Direct access to output occurs.
	Consistency is only ensured for elementary data types. When using arrays, the user is responsible for ensuring data consistency.		
Use	Preferred in MotionTasks	Preferred in the assigned task	Preferred in the BackgroundTask
Declaration as variable	Necessary, for the entire device as an I/O variable in the symbol browser. Each byte of the address range may only be assigned to a single I/O variable. Syntax of I/O address: e.g. PIW1022, PQ63.3.		Possible, but not necessary: • For the entire device as I/O variable in the symbol browser • As unit variable • As local static variable in a program
Download of new or changed I/O variables	Only possible in STOP mode.		-
Use the absolute address	Not supported.		Possible, with the following syntax: E.g. %IW62, %Q63.3.

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
Byte order when forming the process image	-	As supplied by the I/O	Depends on the SIMOTION Kernel version and the memory area setting for the process images: • Up to version V4.1 or the "Separate process image" setting: Always Big Endian • As of version V4.2 and the "Common process image" setting: As supplied by the I/O
Byte order during access	Depends on I/O		Always Big Endian
Writeability of inputs	No	Depends on the SIMOTION Kernel version: • Up to version V4.1: No • As of version V4.2: Yes	Yes
Write protection for outputs	Possible; **Read only** status can be selected.	Not supported.	Not supported.
Declaration of arrays	Possible.		Not supported.
Further information	Direct access and process image of the cyclic tasks (Page 5202).		Access to the fixed process image of the BackgroundTask (Page 5211).
Responses in the event of an error	Error during access from user program, alternative reactions available: • CPU Stop[4] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, alternative reactions available: • CPU stop[5] • Substitute value • Last value See SIMOTION Basic Functions Description of Functions.	Error during generation of process image, reaction: CPU stop[5] **Exception**: If a direct access has been created at the same address, the behavior set there applies.
Access			
• In the RUN operating state	Without any restrictions.	Without any restrictions.	Without any restrictions.
• During the StartupTask	Possible with restrictions: • Inputs can be read. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.	Possible with restrictions: • Inputs are read at the start of the StartupTask. • Outputs are not written until StartupTask is complete.

	Direct access	Access to process image of cyclic tasks	Access to fixed process image of the BackgroundTask
• During the ShutdownTask	Without any restrictions.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.	Possible with restrictions: • Inputs retain status of last update • Outputs are no longer written.

[1] A SIMOTION device is considered isochronous if at least one PROFIBUS DP or PROFINET interface is operated isochronously (constant bus cycle time). For SIMOTION D there is an exception with the DP Integrated interface to which the SINAMICS Integrated is connected.

[2]The following SIMOTION devices are updated in the Servo_fast cycle or IPO_fast cycle, if the cycles are configured: D445-2 DP/PN, D455-2 DP/PN (as of version V4.2) and D435-2 DP/PN (as of version V4.3).

3 IPO or IPO_2 adjustable, see “Setting system cycle clocks" section in the Basic Functions Function Manual.

[4] Call the ExecutionFaultTask.

[5] Call the PeripheralFaultTask.

Direct access and process image of cyclic tasks

Property

Direct access to inputs and outputs and access to the process image of the cyclic task always take place via I/O variables. The entire address range of the SIMOTION device (Page 5204) can be used.

A comparison of the most important properties, including in comparison to the fixed process image of the BackgroundTask (Page 5211) is contained in "Important properties of direct access and process image" (Page 5199).

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 5205).

It is particularly important that every address used in an I/O variable is available in the I/O and configured; each byte in the address range may be assigned to no more than one I/O variable (does not apply to access with data type BOOL).

A detailed status of I/O variables (Page 5209) can be read as of version V4.2 of the SIMOTION Kernel, for example, in order to check the availability of the I/O variables.

Direct access

Direct access is used to access the corresponding I/O address directly. Direct access is used primarily for sequential programming (in MotionTasks). The access to the current value of the inputs and outputs at a specific time is particularly important.

For direct access, you define an I/O variable (Page 5206) without assigning it a task.

Process image of the cyclic task

The process image of the cyclic tasks is a memory area in the RAM of the SIMOTION device, on which the whole I/O address space of the SIMOTION device is mirrored. The mirror image of each I/O address is assigned to a cyclic task and is updated using this task. The task remains consistent throughout the whole cycle. This process image is used preferentially when programming the assigned task (cyclic programming). The consistency during the complete cycle of the task is particularly important.

For the process image of the cyclical task you define an I/O variable (Page 5206) and assign it a task.

Direct access to this I/O variable is still possible: Specify direct access with *_direct.var-name*.

Note

An access via the process image is more efficient than direct access.

Additional properties as of version V4.2 of the SIMOTION Kernel

As of version V4.2 of the SIMOTION Kernel, direct access to inputs/outputs and the process image of the cyclic tasks offers additional properties:

- As far as the process image of the cyclic tasks is concerned, a common memory area with the fixed process image of the BackgroundTask can be set (standard with newly created devices)
- As far as the process image of the cyclic tasks is concerned, I/O variables for inputs can be written to (i.e. they can be assigned values).
- A detailed status of I/O variables (Page 5209) can be read, for example, in order to check the availability of the I/O variables.

Memory area with the fixed process image of the BackgroundTask

- **As of version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks.
- **Up to version V4.1 of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas.

Note

If (and only if) you are also using the fixed process image of the BackgroundTask, it is important to consider the effects of the "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask (Page 5211).

Table 7-546 Effect of "Common process image" or "Separate process image" settings on the process image of the cyclic tasks

	Common process image	Separate process image
Availability	Only available as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Up to version V4.1 of the SIMOTION Kernel applies: • System characteristic, not configurable. The following applies as of version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Download of new or changed I/O variables	Only possible in STOP mode.	Only possible in STOP mode.
Byte order when forming the process image and during access	Depends on connected I/O	
Effects on the fixed process image of the BackgroundTask	See the relevant table in "Access to the fixed process image of the BackgroundTask" (Page 5212).	
Further information	Common process image (Page 5213)	Separate process image (Page 5215)

Address range of the SIMOTION devices

The address range of the SIMOTION devices is specified in the following table according to the version of the SIMOTION Kernel concerned. The complete address range can be used for direct access and process image of the cyclical tasks.

Table 7-547 Address range of the SIMOTION devices according to the version of the SIMOTION Kernel

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
C230-2	0 .. 2047 [3]	0 .. 2047 [3]	0 .. 2047 [3]	–	–	–
C240	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]
C240 PN [1]	–	–	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]	0 .. 4095 [4]
D410 DP	–	–	0 .. 8191 [3]	0 .. 8191 [3]	0 .. 8191 [3]	–
D410 PN	–	–	0 .. 8191 [4]	0 .. 8191 [4]	0 .. 8191 [4]	–
D410-2	–	–	–	–	0 .. 8191 [3 4]	0 .. 8191 [3 4]
D425	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D425-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D435	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D435-2	–	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D445	0 .. 4095 [3]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–	–
D445-1 [1]	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]	–
D445-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
D455-2	–	–	–	0 .. 16383 [3 4]	0 .. 16383 [3 4]	0 .. 16383 [3 4]
P320 [2]	–	–	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

SIMOTION device	Address range for SIMOTION Kernel version					
	V3.2	V4.0	V4.1	4.2	V4.3	As of V4.4
P320-4	–	–	–	–	–	0 .. 4095 [3]
P350	0 .. 2047 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	0 .. 4095 [3]	–

[1] Available as of V4.1 SP2 HF4

[2] Available as of V4.1 SP5

[3] For distributed I/O (over PROFIBUS DP), the transmission volume is restricted to 1024 bytes per PROFIBUS DP line.

[4] For distributed I/O (over PROFINET), the transmission volume is restricted to 4,096 bytes per PROFINET segment.

Rules for I/O addresses for direct access and the process image of the cyclical tasks

Note

You must observe the following rules for the I/O variable addresses for direct access and the process image of the cyclic task (Page 5202). Compliance with the rules is checked during the consistency check of the SIMOTION project (e.g. during the download).

1. Addresses used for I/O variables must be present in the I/O and configured appropriately in the HW Config.
2. I/O variables comprising more than one byte must not contain addresses 63 and 64 contiguously.
   The following I/O addresses are not permitted:
   - Inputs: PIW63, PID61, PID62, PID63
   - Outputs: PQW63, PQD61, PQD62, PQD63
3. All addresses of an I/O variable comprising more than one byte (e.g. WORD, ARRAY data type) must be within a continuous address range configured in HW Config, e.g. within the address range (slot or subslot) of *one* I/O module.
4. An I/O address (input or output) can only be used by a single I/O variable of data type BYTE, WORD or DWORD or an array of these data types. Access to individual bits with I/O variables of data type BOOL is possible.
5. If several processes (e.g. I/O variable, technology object, PROFIdrive telegram) access an I/O address, the following applies:
   - Only a single process can have write access to an I/O address of an output (BYTE, WORD or DWORD data type).
     Read access to an output with an I/O variable that is used by another process for write access, is possible.
   - All processes must use the same data type (BYTE, WORD, DWORD or ARRAY of these data types) to access this I/O address. Access to individual bits is possible irrespective of this. Please be aware of the following, for example, if you wish to use an I/O variable to read the PROFIdrive telegram transferred to or from the drive: The length of the I/O variable must match the length of the telegram.
   - Write access to different bits of an address is possible from several processes; however, write access with the data types BYTE, WORD or DWORD is then not possible.

Note

These rules do not apply to accesses to the fixed process image of the BackgroundTask (Page 5211). These accesses are not taken into account during the consistency check of the project (e.g. during download).

Creating I/O variables for direct access or process image of cyclic tasks

Create I/O variables for direct access or a process image of the cyclic tasks in the address list of the detail view.

This is only possible in offline mode.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select **Insert new line**
   or
   Scroll to the end of the table of variables (empty line).
3. In the empty row of the table, enter or select the following:
   - Names of the **I/O variables**
   - **I/O address**
     Select the "IN" or "OUT" entries if you wish to assign symbols to the I/O variable (input or output). As of Version V4.2 of the SIMOTION Kernel the symbolic assignment must be activated, menu **Project > Use symbolic assignment**.
     Or enter a fixed address according to "Syntax for entering I/O addresses" (Page 5208).
   - Optional for outputs:
     Activate the **Read only** checkbox if you only want to have read access to the output. You can then read an output that is already being written by another process (e.g. output of an output cam, PROFIdrive telegram).
     A read-only output variable cannot be assigned to the process image of a cyclic task.
   - **Data type** of the variables in accordance with "Possible data types of the I/O variables" (Page 5209).

4. Optionally, you can also enter or select the following (not for data type BOOL):
   - **Array length** (array size).
   - **Process image** or direct access:
     Can only be assigned if the **Read only** checkbox is deactivated.
     For process image, select the cyclic task to which you want to assign the I/O variable. To select a task, it must have been activated in the execution system.
     For direct access, select the blank entry.
   - **Strategy** for behavior in the event of an error, see SIMOTION Basic Functions Function Manual.
   - **Display format** (if array, for each element), when you monitor the variable in the address list
   - **Substitute value** (if array, for each element).
5. Only if you have selected "IN" or "OUT" as the I/O address (symbolic assignment).
   - In the **Assignment** column, click the **[...]** button.
     A window opens displaying the possible assignment targets of the SIMOTION device and, if necessary, of SINAMICS Integrated. Only those assignment targets are displayed that match the data direction (input/output) and data type.
   - Select the assignment target.
     The **Assignment status** column indicates whether the assignment was successful or not.

   For details regarding symbolic assignment, refer to the SIMOTION Basic Functions Function Manual.

You can now access this variable using the address list or any program of the SIMOTION device.

Details on how to manage the address list can be found in the online help.

Note

Note the following for the process image for cyclic tasks:

- A variable can only be assigned to one task.
- Each byte of an input or output can only be assigned to one I/O variable.

In the case of data type BOOL, please note:

- The process image for cyclic tasks and a strategy for errors cannot be defined. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access or CPU stop).
- The individual bits of an I/O variable can also be accessed using the bit access functions.

Take care when making changes within the I/O variables (e.g. inserting and deleting I/O variables, changing names and addresses):

- In some cases the internal addressing of other I/O variables may change, making all I/O variables inconsistent.
- If this happens, all program sources that contain accesses to I/O variables must be recompiled.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and then use them in your program sources (e.g. ST sources, MCC sources, LAD/FBD sources).

Outputs can be read and written to, but inputs can only be read.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 5221).

Syntax for entering I/O addresses

Syntax

For the input of the I/O address for the definition of an I/O variable for direct access or process image of cyclical tasks (Page 5202), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

Table 7-548 Syntax for the input of the I/O addresses for direct access or process image of the cyclic tasks

Data type	Syntax for		Permissible address range					
	Input	Output	Direct access		Process image		e.g. direct access D435 V4.1	
BOOL	PIn.x	PQn.x	n: x:	0 .. *MaxAddr* 0 .. 7		-[1]	n: x:	0 .. 16383 0 .. 7
BYTE	PIBn	PQBn	n:	0 .. *MaxAddr*	n:	0 .. *MaxAddr*	n:	0 .. 16383
WORD	PIWn	PQWn	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. *MaxAddr* - 1	n:	0 .. 62 64 .. 16382
DWORD	PIDn	PQDn	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. *MaxAddr* - 3	n:	0 .. 60 64 .. 16380
n = logical address x = bit number								
MaxAddr =	Maximum I/O address of the SIMOTION device depending on the SIMOTION Kernel version, see Address range of the SIMOTION devices (Page 5204).							
[1] For data type BOOL, it is not possible to define the process image for cyclic tasks. The behavior defined via an I/O variable for the entire byte is applicable (default: direct access).								

Examples

Input at logic address 1022, WORD data type: **PIW1022**.

Output at logical address 63, bit 3, BOOL data type: **PQ63.3**.

Note

Observe the rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 5205).

Possible data types of I/O variables

The following data types can be assigned to the I/O variables for direct access and process image of the cyclical tasks (Page 5202). The width of the data type must correspond to the data type width of the I/O address.

If you assign a numeric data type to the I/O variables, you can access these variables as integer.

Table 7-549 Possible data types of the I/O variables for direct access and the process image of the cyclical tasks

Data type of I/O address	Possible data types for I/O variables
BOOL (PIn.x, PQn.x)	BOOL
BYTE (PIBn, PQBn)	BYTE, SINT, USINT
WORD (PIWn, PQWn)	WORD, INT, UINT
DWORD (PIDn, PQDn)	DWORD, DINT, UDINT

For details of the data type of the I/O address, see also "Syntax for entering I/O addresses" (Page 5208).

Detailed status of the I/O variables (as of Kernel V4.2)

As of version V4.2 of the SIMOTION Kernel, the status of an I/O variable can be queried using *_quality.var-name*, for example, in order to check the availability of the I/O variables. It is supplied as an OR logic operation of the following status values in the DWORD data type and can be assigned to an appropriate variable, for example. The value 16#0000_0000 indicates the connected I/O are operating without error.

The same value is supplied for every I/O variable within an address range (slot or subslot) configured in HW Config.

Table 7-550 Meanings of status values of I/O variables

Value (DWORD)	Bit x = 1	Meaning
16#0000_0000	-	No error occurred.
16#0000_0001	0	**Maintenance required** The connected module signals that it requires maintenance. The component needs to be checked within a foreseeable period (e.g. the printer cartridge must be changed within a period of several days).
16#0000_0002	1	**Maintenance demanded** The connected module demands maintenance. The component needs to be checked soon (e.g. the printer cartridge must be changed immediately).
16#0000_0004	2	**Warning pending** (drive warning, TM17 warning, etc.) The connected module has signaled a warning. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.
16#0000_0008	3	**Fault pending** (diagnostic interrupt, drive fault, TM17 fault, etc.) The connected module has signaled an error. This has been entered in the diagnostics buffer. The precise cause can be determined from the documentation for the relevant module.

Value (DWORD)	Bit x = 1	Meaning
16#0000_0010	4	**This parameter assignment does not match the parameter assignment being compared.** A difference was detected when this parameter assignment was compared with the parameter assignment of the connected module. As such, the required functionality cannot be guaranteed. Remedy: Save the project and compile changes, reload both application and counterpart.
16#0000_0020	5	**Application and counterpart are not isochronous** (error involving the dynamic life-sign). Certain telegrams (axis, synchronous operation, output cam, measuring input telegrams) are synchronized by exchanging cyclic life-signs. Errors are detected when the cyclic life-sign is checked. This invalidates the data in the telegram. Remedy: Await synchronization, check the parameter assignment (e.g. does the master application cycle set on the device in HW Config match the position control cycle clock), save the project and compile changes, reload both application and counterpart.
16#0000_0040	6	**I/O cannot be used synchronously in all cycles.** A fast application cycle (Servo_fast) and a slow application cycle (Servo) are running asynchronously in relation to one another. The I/O can only be used synchronously in the cycles associated with the bus cycle. Access from other cycles is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0080	7	**I/O cannot be used synchronously** The SIMOTION control is the sync slave on a bus. The bus connection is running synchronously in relation to the sync master, but is not yet running synchronously in relation to the application cycles of the SIMOTION control. Access to the I/O is asynchronous and inconsistent. Remedy: Call the _synchronizeDpInterfaces() function.
16#0000_0100	8	**Bus connection (sync slave) is not isochronous in relation to the sync master.** The SIMOTION control is the sync slave on a bus and has not yet synchronized its bus connection with the sync master. The isochronous I/O on this bus cannot be used yet. Remedy: Switch on/connect the sync master.
16#0000_0200	9	**DP station is deactivated.** The partner module has been deactivated. Remedy: Activate the partner module (_activateDpSlave() function).
16#0000_0400	10	**The partner of the inputs (e.g. I-device, I-slave) is in STOP.** The connected module is in STOP mode and not sending any new data as a result. Remedy: Switch the connected module to RUN.
16#0000_0800	11	**PROFINET: Failure detected by submodule** (e.g. channel error) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_1000	12	**PROFINET: Failure detected by module** (e.g. submodule failed, removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0000_2000	13	**PROFINET: Failure detected by device** (e.g. device in STOP, module removed, etc.) The connection to the connected device is OK. The error must be searched for in the connected device. Troubleshooting: Diagnostics buffer, device diagnostics with HW Config

Value (DWORD)	Bit x = 1	Meaning
16#0000_4000	14	**PROFINET: Failure detected by controller** (e.g. not connected, etc.) There is no connection to a partner on PROFINET. Possible cause: Partner is switched off, cable pulled out, incorrect parameter assignment for connection Troubleshooting: Best to use the PROFINET topology editor in HW Config.
16#0000_8000	15	**Slot/subslot is not connected** (disconnection alarm). The connection to the connected device is OK. The error must be searched for in the connected device (e.g. module/submodule removed). Troubleshooting: Diagnostics buffer, device diagnostics with HW Config
16#0001_0000	16	**Device is not connected** (station failure). There is no connection to a partner. Possible cause: Partner is switched off, cable pulled out.
16#0002_0000	17	**Substitute value behavior during access** There is no connection to the counterpart (sum signal from bits 9 to 16), i.e. there is no valid input data or the output data is not reaching the terminal. The substitute value behavior set (substitute value, last value) takes effect during direct access to this address or during process image updates.
16#4000_0000	30	**Diagnostics address only** No cyclic I/O data is configured for this address. It is possible, however, to query submodule diagnostic information.
16#8000_0000	31	**Address gap** There is no hardware configured for this logical address.

Access to fixed process image of the BackgroundTask

The fixed process image of the BackgroundTask is a memory area in the RAM of the SIMOTION device on which a subset of the I/O address space of the SIMOTION device is mirrored. Preferably, it should be used for programming the BackgroundTask (cyclic programming) as it is consistent throughout the entire cycle.

The size of the fixed process image of the BackgroundTask for all SIMOTION devices is 64 bytes (address range 0 .. 63).

Note

The fixed process image of the BackgroundTask can be used to access addresses that are not available in the I/O or not configured in HW Config. These are treated like normal memory addresses.

Memory area

- **As of Version V4.2 of the SIMOTION Kernel**, selecting a "Common process image" setting on the device ensures the memory area for the fixed process image of the BackgroundTask is a subset of the memory area for the process image of the cyclic tasks.
  I/O addresses can be read and written to using both the fixed process image of the BackgroundTask and the process image of the cyclic tasks.
- **With Version V4.1 and lower of the SIMOTION Kernel** or the "Separate process image" setting on the device (as of Version V4.2 of the SIMOTION Kernel), the fixed process image of the BackgroundTask and the process image of the cyclic tasks occupy different memory areas. I/O addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.

Table 7-551 Effect of "Common process image" or "Separate process image" settings on the fixed process image of the BackgroundTask

	Common process image	Separate process image
Availability	Only available as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard for newly created devices	Version V4.1 and lower of the SIMOTION Kernel applies: • System characteristic, not configurable The following applies as of Version V4.2 of the SIMOTION Kernel: • Setting available for selection • Standard with device upgrades
Memory area	Subset of the memory area for the process image of the cyclic tasks	Separate memory area for the process image of the cyclic tasks
Using I/O addresses accessed using the process image of the cyclic tasks	Possible. Updates use the configured cyclic tasks.	Not supported. The addresses are treated like normal memory addresses.
Byte order when forming the process image	As supplied by the I/O	Always Big Endian
Byte order when accessing the process image	Always Big Endian	Always Big Endian
Access to I/O operating in the Little Endian byte order	Same result as during direct access or for the process image of cyclic tasks (apart from WORD or DWORD data types).	Results differ depending on the I/O variables created for direct access.
Effects on the process image of the cyclic tasks	See the relevant table in "Direct access and process image of the cyclic tasks (Page 5204)".	
Further information	Common process image (Page 5213)	Separate process image (Page 5215)

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

A comparison of the most important properties in comparison to the direct access and process image of the cyclic tasks (Page 5202) is contained in "Important properties of direct access and process image" (Page 5199).

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 5205) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

You can access the fixed process image of the BackgroundTask by means of:

- Using an absolute PI access (Page 5217): The absolute PI access identifier contains the address of the input/output and the data type.
- Using a symbolic PI access (Page 5219): You declare a variable that references the relevant absolute PI access:
  - A unit variable
  - A static local variable in a program.
- Using an I/O variable (Page 5221): In the symbol browser, you define a valid I/O variable for the entire device that references the corresponding absolute PI access.

Common process image (as of Kernel V4.2)

As of Version V4.2 of the SIMOTION Kernel, the "Common process image" setting can be selected on the SIMOTION device. This means addresses 0 .. 63 of the process image of the cyclic tasks and the fixed process image of the BackgroundTask occupy the same memory area.

This is the default for SIMOTION devices newly created in the project as of Version V4.2.

Property of the common process image

1. The memory area for the fixed process image of the BackgroundTask (Page 5211) is a subset of the memory area for the process image of the cyclic tasks (Page 5202).
2. This means I/O addresses already accessed using the process image of the cyclic tasks may also continue to be used for the fixed process image of the BackgroundTask. Updates, however, use the configured cyclic tasks.
3. The following applies when forming the fixed process image of the BackgroundTask:
   The byte order is the same as supplied by the I/O:
   - Big Endian, e.g. for I/O via PROFIBUS DP, PROFINET, P-Bus, DRIVE-CLiQ
   - Little Endian, e.g. for onboard I/O of C240, C240 PN SIMOTION devices

   Any I/O variable created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks has no effect on the byte order.

4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C240, C240 PN SIMOTION devices).
   If the fixed process image of the BackgroundTask is used for access, this leads to the following behavior, regardless of whether I/O variables have been created for the relevant addresses for the purpose of direct access or the process image of the cyclic tasks:
   - Access to individual bytes always supplies the same result via an I/O variable or the fixed process image of the BackgroundTask.
   - With the fixed process image of the BackgroundTask, bytes only change places if data type WORD is used for access.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for common process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access or the process image of the cyclic tasks:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access or access to the process image of the cyclic tasks:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte function (direct access only)
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-552 "Common process image" setting (as of Kernel V4.2): Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access or access to the process image of the cyclic tasks	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	16#08
	%IB61	16#00	16#00	16#00
	%IW60	**16#0800** [2]	**16#0800** [2]	**16#0800** [2]
	%I60.3	TRUE	TRUE	TRUE
	%I61.3	FALSE	FALSE	FALSE

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access or the process image of the cyclic tasks; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian byte order is being read using Big Endian.

Separate process image (up to Kernel V4.1)

With Version V4.1 and below of the SIMOTION Kernel, the process image of the cyclic task and the fixed process image of the BackgroundTask are stored in different memory areas (separate process image).

As of Version V4.2 of the SIMOTION Kernel, the "Separate process image" setting can be selected on the SIMOTION device. This setting ensures there is compatibility with earlier Kernel versions.

It is the default for SIMOTION devices upgraded to Version V4.2 or higher.

Property of the separate process image

1. The fixed process image of the BackgroundTask (Page 5211) and the process image of the cyclic tasks (Page 5202) are stored in different memory areas.
2. This means I/O addresses that are already accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask. They are treated like normal memory addresses.
3. I/O variables for direct access influence the fixed process image of the BackgroundTask:
   – The fixed process image of the BackgroundTask is always formed for the relevant addresses in the Big Endian byte order.
4. Access to the fixed process image of the BackgroundTask always takes place using the Big Endian byte order.
5. These last two properties (nos. 3 and 4) affect access to inputs and outputs operating with the Little Endian byte order (e.g. onboard I/O of C230-2, C240, C240 PN SIMOTION devices). If an I/O variable is created for the relevant addresses for the purpose of direct access using data type WORD and access takes place using the fixed process image of the BackgroundTask, this leads to the following behavior:
   – Access with the data type WORD supplies the same result via the I/O variable and the fixed process image of the BackgroundTask.
   – Access to individual bytes using the _getInOutByte function (see SIMOTION Basic Functions Function Manual) supplies these in the Little Endian order.
   – Access to the individual bytes or bits with the fixed process image of the BackgroundTask supplies these in the Big Endian order.

   Please also refer to the example below.

For information on the order of the Little Endian and Big Endian bytes, please refer to the SIMOTION Basic Functions Function Manual.

Example for separate process image: Access to I/O operating with the Little Endian byte order

The digital inputs of the C240 SIMOTION device operate with the Little Endian byte order and occupy addresses 66 (bits 0 ..7) and 67 (bits 0.. 3) by default. The start address is changed to 60 in HW Config to ensure it is in the range occupied by the fixed process image of the BackgroundTask. Addresses 60 and 61 are now accessed using various I/O variables and the process image of the BackgroundTask.

The following three scenarios are considered, which differ in terms of whether and which I/O variables are created for direct access:

1. Scenario A:
   No I/O variables are created for addresses 60 and 61.
2. Scenario B:
   Two I/O variables with data type BYTE are created for addresses 60 and 61: io_byte_60 (PIB60) and io_byte_61 (PIB61).
3. Scenario C:
   For adresss 60, **one I/O-Variable** with data type WORD is created; this also covers address 61: io_word_60 (PIW60).

Two additional I/O variables are also created in each of the three scenarios, making it possible to access bit 3: io_bit_60_3 (PI60.3) and io_bit_61_3 (PI61.3).

The table below lists which values are generated with the following access types:

- Direct access:
  - Access to individual bytes or the word using the relevant I/O variables
  - Access to each individual byte using the _getInOutByte function
  - Access to the respective bit 3 using the relevant I/O variables
- Access to the fixed process image of the BackgroundTask:
  - Access to individual bytes using an absolute name
  - Access to the word using an absolute name
  - Access to the respective bit 3 using an absolute name

Table 7-553 "Separate process image" setting or Kernel up to Version V4.1: Different types of access to the process images of an input operating with the Little Endian byte order

	Access using	**Scenario A** [1]	**Scenario B** [1]	**Scenario C** [1]
Direct access	io_byte_60 (PIB60)	-	16#08	-
	io_byte_61 (PIB61)	-	16#00	-
	io_word_60 (PIW60)	-	-	16#0008
	_getInOutByte (IN, 60)	16#08	16#08	16#08
	_getInOutByte (IN, 61)	16#00	16#00	16#00
	io_bit_60_3 (PI60.3)	TRUE	TRUE	TRUE
	io_bit_61_3 (PI61.3)	FALSE	FALSE	FALSE
Access to the fixed process image of the BackgroundTask	%IB60	16#08	16#08	**16#00** [3]
	%IB61	16#00	16#00	**16#08** [3]
	%IW60	**16#0800** [2]	**16#0800** [2]	16#0008
	%I60.3	TRUE	TRUE	**FALSE** [3]
	%I61.3	FALSE	FALSE	**TRUE** [3]

[1] Scenarios A, B, or C determine whether and which I/O variables are created for direct access; see the explanation provided in the body of the document.

[2] The two bytes in the word change places, as a value saved in the Little Endian order is being read using Big Endian.

[3] The two adjacent bytes change places, as the relevant word is saved in the Big Endian order.

Absolute access to the fixed process image of the BackgroundTask (absolute PI access)

You make absolute access to the fixed process image of the BackgroundTask (Page 5211) by directly using the identifier for the address (with implicit data type). The syntax of the identifier (Page 5218) is described in the following section.

You can use the identifier for the absolute PI access in the same manner as a normal variable.

Note

Outputs can be read and written to, but inputs can only be read.

Syntax for the identifier for an absolute process image access

For the absolute access to the fixed process image of the BackgroundTask (Page 5217), use the following syntax. This specifies not only the address, but also the data type of the access and the mode of access (input/output).

You also use these identifiers:

- For the declaration of a symbolic access to the fixed process image of the BackgroundTask (Page 5219).
- For the creation of an I/O variables for accessing the fixed process image of the BackgroundTask (Page 5221).

Table 7-554 Syntax for the identifier for an absolute process image access

Data type	Syntax for		Permissible address range	
	Input	Output		
BOOL	%In.x or %IXn.x [1]	%Qn.x or %QXn.x[1]	n: x:	0 .. 63 [2] 0 .. 7
BYTE	%IBn	%QBn	n:	0 .. 63 [2]
WORD	%IWn	%QWn	n:	0 .. 63 [2]
DWORD	%IDn	%QDn	n:	0 .. 63 [2]
n = logical address x = bit number				
[1] The syntax %IXn.x or %QXn.x is not permitted when defining I/O variables.				
[2] For a separate process image (Page 5215), the following applies: No addresses that are used in the process image of the cyclic tasks. See note below.				

Examples

Input at logic address 62, WORD data type: **%IW62**.

Output at logical address 63, bit 3, BOOL data type: **%Q63.3**.

Note

Up to Version V4.1 of the SIMOTION Kernel or the "Separate process image" (Page 5215) setting on the device (as of Version V4.2 of the SIMOTION Kernel), the following applies:

- Addresses accessed using the process image of the cyclic tasks cannot be read or written to using the fixed process image of the BackgroundTask.

This restriction no longer applies as of Version V4.2 of the SIMOTION Kernel or with the "Common process image" (Page 5213) setting on the device.

Note

The rules for I/O addresses for direct access and the process image of the cyclical tasks (Page 5205) do **not** apply. Access to the fixed process image of the BackgroundTask is not taken into account during the consistency check of the project (e.g. during download).

Addresses not present in the I/O or not configured in HW Config are treated like normal memory addresses.

Defining symbolic access to the fixed process image of the BackgroundTask

You create symbolic access to the fixed process image of the BackgroundTask in the declaration tables of the source file, MCC chart, or LAD/FBD program (only in the case of programs). The scope of the symbolic process image access is dependent on the location of the declaration:

- In the interface section of the declaration table of the source file (INTERFACE):
  Symbolic process image access behaves like a unit variable; it is valid for the entire source file; all MCC charts or LAD/FBD programs (programs, function blocks, and functions) within the source file can access the process image.
  In addition, these variables are available on HMI devices and, once connected, in other source files (or other units), as well.
  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the source file (IMPLEMENTATION):
  Symbolic process image access behaves like a unit variable; it is only valid in the source file; all MCC charts or LAD/FBD programs (programs, function blocks, and functions) within the source file can access it.
- In the declaration table for the MCC chart or LAD/FBD program (only in the case of programs):
  Symbolic process image access behaves like a local variable; it can only be accessed within the MCC chart or LAD/FBD program in which it is declared.
  No symbolic process image access can be declared in functions or function blocks.

Proceed as follows; the source file or the MCC chart or LAD/FBD program (in the case of programs only) with the declaration table is opened:

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the I/O Symbols tab.
3. Enter:
   - Name of symbol (variable name)
   - For Absolute ID, the identifier of the absolute process image access (Page 5218).
   - Data type of symbol (Page 5220) (this must agree with the length of the process image access).

Possible data types for symbolic PI access

In the following cases, a data type that differs from that of the absolute PI access can be assigned to the fixed process image of the BackgroundTask (Page 5211). The data type width must correspond to the data type width of the absolute PI access.

- For the declaration of a symbolic PI access (Page 5219).
- For the creation of an I/O variable (Page 5221).

If you assign a numeric data type to the symbolic PI access or to the I/O variables, you can access these variables as integer.

Table 7-555 Possible data types for symbolic PI access

Data type of the absolute PI access	Possible data types of the symbolic PI access
BOOL (%In.x, %IXn.x, %Qn.x. %QXn.x)	BOOL
BYTE (%IBn, %QBn)	BYTE, SINT, USINT
WORD (%IWn, %QWn)	WORD, INT, UINT
DWORD (%IDn, %PQDn)	DWORD, DINT, UDINT

For the data type of the absolute PI access, see also "Syntax for the identifier for an absolute PI access (Page 5218)".

Example: Defining symbolic access to the fixed process image of the BackgroundTask

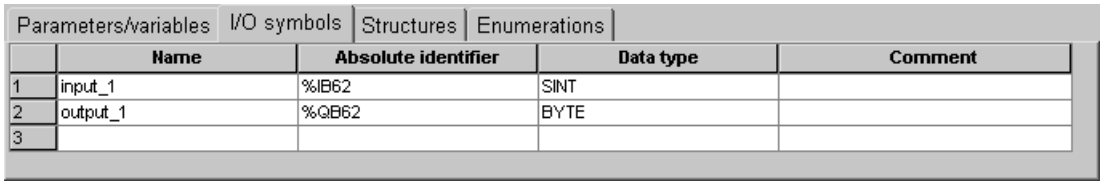


Figure 7-491 Example: Defining symbolic access to the fixed process image of the BackgroundTask

Creating an I/O variable for access to the fixed process image of the BackgroundTask

You create I/O variables for access to the fixed process image for the background task in the symbol browser in the detail view; you must be in offline mode to do this.

Here is a brief overview of the procedure:

1. Select the "Address list" tab in the detail view and choose the SIMOTION device
   or
   In the project navigator of SIMOTION SCOUT, double-click the "ADDRESS LIST" element in the SIMOTION device subtree.
2. Select the line before which you want to insert the I/O variable and, from the context menu, select Insert new line
   or
   Scroll to the end of the table of variables (empty line).
3. In the detail view, select the Symbol browser tab and scroll down to the end of the variable table (empty row).
4. In the empty row of the table, enter or select the following:
   - **Name** of variable.
   - Under **I/O address**, the absolute PI access according to the "Syntax for the identifier for an absolute PI access" (Page 5218)
     (exception: The syntax %IXn.x or %QXn.x is not permitted for data type BOOL).
   - **Data type** of the I/O variables according to the "Possible data types of the symbolic PI access" (Page 5220).
5. Select optionally the display format used to monitor the variable in the symbol browser.

You can now access this variable using the address list or any program of the SIMOTION device.

Note

I/O variables can only be created in offline mode. You create the I/O variables in SIMOTION SCOUT and use them in your program sources.

Note that you can read and write outputs but you can only read inputs.

Before you can monitor and modify new or updated I/O variables, you must download the project to the target system.

You can use I/O variables like any other variable, see "Access I/O variables" (Page 5221).

Accessing I/O variables

You have created an I/O variable for:

- Direct access or process image of the cyclic tasks (Page 5202).
- Access to the fixed process image of the BackgroundTask (Page 5211).

You can use this I/O variable just like any other variable.

Note

Consistency is only ensured for elementary data types.

When using arrays, the user is responsible for ensuring data consistency.

Note

If you have declared unit variables or local variables of the same name (e.g. *var-name*), specify the I/O variable using *_device.var-name* (predefined name space, see the "Predefined name spaces" table in "Name spaces").

It is possible to directly access an I/O variable that you created as a process image of a cyclic task. Specify direct access with *_direct.var-name* or *_device._direct.var-name*.

If you want to deviate from the default behavior when errors occur during variable access, you can use the _getSafeValue and _setSafeValue functions (see *SIMOTION Basic Functions* Function Manual).

For Errors associated with access to I/O variables, see *SIMOTION Basic Functions* Function Manual.

7.3.4.17 Connections to other program source files or libraries

In the declaration table of a unit, you can define connections to:

- LAD/FBD units under the same SIMOTION device
- MCC units under the same SIMOTION device
- ST source files under the same SIMOTION device
- Libraries

This will then allow you to access the following in this unit:

- For connected program sources (Page 5223), the following items which are defined there
  - Functions
  - Function blocks
  - Classes and methods contained therein (only for ST source files)
  - Programs (optional)
  - Unit variables
  - User-defined data types (structures, enumerations)
  - Symbolic accesses to the fixed process image of the BackgroundTask
- For connected libraries (Page 5224), the following items which are defined there
  - Functions
  - Function blocks
  - Classes and methods contained therein (only for ST libraries)
  - Programs (optional)
  - User-defined data types (structures, enumerations)

Program sources and libraries must be compiled beforehand.

For information about the library concept, see also the SIMOTION ST Programming Manual.

Note

Libraries can be created in all programming languages (MCC, ST, or LAD/FBD).

Defining connections

Procedure for defining connections to other program sources (units)

Connections to other units (program sources) are defined in the declaration table of the source file. The mode of action of a connection is dependent on the section of the declaration table in which it is defined.

- In the interface section of the declaration table:
  The imported functions, variables, etc., will continue to be exported to other units and to HMI devices. This can lead to name conflicts.
  This setting is necessary, for example, if unit variables are declared in the interface section of the source file with a data type that is defined in the imported program source.
- In the implementation section of the declaration table:
  The imported functions, variables, etc. will no longer be exported.
  This setting is usually sufficient.

Proceed as follows; the source file (declaration table) is open (see Open existing program sources (Page 5104)):

1. In the declaration table, select the section for the desired mode of action.
2. Select the **Connections** tab.
3. For the connection type, select: **Program/Unit**
4. In the same line, select the name of the unit to be connected:
   Units (program sources) must be compiled beforehand.

Procedure for defining connections to libraries

Connections to libraries are defined in the declaration table of the source file.

Proceed as follows; the unit (declaration table) is open, see Open existing program sources (Page 5104):

1. In the interface section of the declaration table, select the **Connections** tab.
2. For the connection type, select: **Library**.
3. In the same line, select the name of the library to be connected.
   Libraries must be compiled beforehand.
4. Optionally, you can define a name space for libraries, see Using name space (Page 5224):
   To do this, enter a name under **Name space**.

Note

When programming the "Subprogram call" command (see Inserting and parameterizing subroutine calls (Page 5228)) with a library function or a library function block, the connection to the library is automatically entered into the declaration table of the program source.

Using the name space

You can optionally assign a namespace to every connected library. You define the designation of the namespace when connecting the library (see How to define connections to libraries (Page 5224)).

It is important to specify the namespace if the current LAD/FBD program / MCC chart or program source contains variables, data types, functions, or function blocks with the same name as the connected library. The namespace will then allow you specific access to the variables, data types, functions, or function blocks in the library. This can also resolve naming conflicts between connected libraries.

If you wish to use variables, data types, functions, or function blocks from the connected library in a command in the LAD/FBD program or MCC chart, insert the designation of the namespace in front of the variable name, etc., from the library and separate them with a period (for example, namespace.var_name, namespace.fc_name).

Predefined namespaces

Namespaces are predefined for device-specific and project-specific variables, direct accesses to I/O variables, and variables of TaskId and AlarmId in the following table: If necessary, write their designation before the variable name, separated by a period, e.g. *_device.var_name* or *_task.task_name*.

Table 7-556 Predefined namespaces

Namespace	Description
_alarm	For AlarmId: The _alarm.name variable contains the AlarmId of the message with the name identifier – see SIMOTION Basic Functions Function Manual.
_device	For device-specific variables (global device user variables, I/O variables, system variables, and system variables of the SIMOTION device)
_direct	For direct access to I/O variables – see Direct access and process image of the cyclic tasks (Page 5202). Local namespace for _device. Nesting as in _device._direct.*name* is permitted.
_project	For names of SIMOTION devices in the project; only used with technology objects on other devices. With unique project-wide names of technology objects, used also for these names and their system variables
_quality	As of Version V4.2 of the SIMOTION Kernel: For the detailed status of I/O variables (Page 5209). A value with data type DWORD is supplied. Local namespace for _device. Nesting as in _device._quality.*name* is permitted.
_task	For TaskID: The _task.name variable contains the TaskId of the task with the *name* identifier – see SIMOTION Basic Functions Function Manual.
_to	For technology objects configured on the SIMOTION device and their system variables and configuration data Not for system functions and data types of the technology objects. In this case, use the user-defined namespace for the imported technology package, if necessary.

Object-oriented namespaces

You can assign functions, function blocks or programs to an object-oriented namespace. Enter the identifier of the object-oriented namespace when creating (Page 5117) or in the Properties window (Page 5122) of the respective program organization unit (POU). The compiler option (Page 5111) "Permit object-oriented programming" must be activated.

A POU within the namespace can only be accessed from a POU outside the namespace when the identifier of the namespace is added as prefix to the identifier of the POU, separated by a dot.

In the project navigator, the POU is shown as subelement of the namespace below the program source.

7.3.4.18 Subroutine

Universal, reusable sections of a program can be created in the form of subroutines.

When a subroutine is called, the program branches from the current task into the subroutine. The commands in the subroutine are executed. The program then jumps back to the previously active task.

Subroutines can be called repeatedly, as required, by one or more LAD/FBD programs of the SIMOTION device.

Subroutine as a function (FC), function block (FB), or program

The creation type of a subroutine can be a function (FC), a function block (FB) or, as an option, a program ("program in program").

- Function
  A function (FC) is a subroutine without static data, that is, all local variables lose their value when the function has been executed. They are re-initialized when the function is next started.
  Data are transferred to the function using input or in/out parameters; the output of a function value (return value) is also possible.
- Function block
  A function block (FB) is a subroutine with static data, that is, local variables retain their value after the function block has been executed. Only variables that have been explicitly declared as temporary lose their value.
  An instance has to be defined before using an FB: Define a variable (VAR or VAR_GLOBAL) and enter the name of the FB as data type. The FB static data is saved in this instance. You can define several FB instances; each instance is independent from the others.
  The static data of an FB instance remain stored until the instance is next called; they are reinitialized when the variable type of the FB instance is initialized again (see Initialization of instances of function blocks (FB) (Page 5187)).
  Data are transferred to the FB using input parameters or in/out parameters; the data are returned from the FB using in/out or output parameters.
- Program ("program in program")
  You also have the option of calling a program within a different program or a function block. This requires the following compiler options to be activated (see Global compiler settings (Page 5111) and Local compiler settings (Page 5112)):
  - "Permit language extensions" for the program source of the calling program or function block and
  - "Only create program instance data once" for the program source of the called program. The static data of the called program is stored in the user memory of the program source (unit) of said called program.

  Most of the programming work involved in assigning the programs to the tasks can be performed by calling up programs within another program. In the execution system, only one associated calling program needs to be assigned to the tasks concerned.
  A program is called without parameters or return values.
  Further information on calling a program within a program can be found in the ST Programming and Operating Manual.

Note

The activated "Only create program instance data once" compiler option causes:

- The static variables of the programs (program instance data) to be stored in the user memory of the program source (unit) (see the SIMOTION ST Programming and Operating Manual). This also causes the initialization behavior to change (see the SIMOTION ST Programming and Operating Manual).
- All called programs with the same name to use the same program instance data.

Exchange of information between the subroutine and calling program

Function (FC) and function block (FB) as a subroutine

Information is exchanged between the subroutine and the calling program using transfer parameters or global variables (e.g. unit variables).

Transfer parameters can be input, input/output or output parameters. They are defined in the declaration table for the subroutine:

- Input parameters: As variable type VAR_INPUT
- In/out parameter: As variable type VAR_IN_OUT
- Output parameter (for FB only): As variable type VAR_OUTPUT

For functions, a function value can be returned; you specify the data type of the return value when you paste in (create) the function (see Paste in function (FC) or function block (FB) (Page 5228)).

You assign current values to the input and/or in/out parameters when you call the subroutine (FC or FB instance). You may only assign user-defined variables to the in/out parameters of an FB because the called FB accesses the assigned variables directly and can therefore change them.

The output parameters of an FB can be read-accessed as often as required in the calling program.

A function does not formally contain any output parameters, since the result of the function can in this case be assigned to the return value of the function.

See also the examples of functions (Page 5233) and function blocks (Page 5238).

Program as subroutine ("program in program")

A program is called without parameters or return values. This means that information can only be exchanged between the calling program and the called program (subroutine) using global variables (e.g. unit variables).

See also

Inserting a subroutine call into the LAD/FBD program and assigning parameters (Page 5228)

Inserting a function (FC) or function block (FB)

The creation dialog is similar to that of an LAD/FBD program:

1. LAD/FBD unit must already exist (see Managing LAD/FBD programs (Page 5117)).
2. In the project navigator, open the relevant LAD/FBD unit.
3. Double-click the entry **Insert LAD/FBD program**.
   The input screen form opens.
   - Enter the name of the LAD/FBD program (see Rules for identifiers (Page 5155)).
   - For the creation type, select **Function** or **Function block**.
   - With creation type Function only:
     Select the data type of the return value as the return type (<--> for no return value).
   - Check the **Exportable** option if the function or function block is to be used in other program source files (LAD/FBD, MCC or ST source files).
     When the checkbox is cleared, the LAD/FBD program can only be used in the associated LAD/FBD unit.
   - You can also enter an author, version, and a comment.
   - Confirm with **OK**.
4. Program the instructions in the function or function block.
   Assign an expression to the return value of a function (= function name) or to the output parameters of a function block.
5. Accept and compile the LAD/FBD unit. The subroutine you have created will then be displayed in the list.

Inserting a subroutine call into the LAD/FBD program and assigning parameters

In order to execute a call of a subprogram (function, function block, or program), the relevant subprogram must have been inserted in the network of an LAD/FBD program from the project navigator using drag-and-drop. When the subprogram inserted in the network is reached during a program run, the subprogram is called and the program branches from the current task into the subprogram.

You can use drag-and-drop to insert the following FCs, FBs, methods and programs in an LAD/FBD program and call them as a subprogram:

- Functions, function blocks, or programs of the same LAD/FBD unit or a different program source (e.g. MCC unit, ST source file).
- Methods within classes or function blocks of an ST source file.
- Library functions, library function blocks or library methods from a program library.

The subprogram call is parameterized, i.e. specifications are made as to which variables are to be transferred when the subprogram is called and returned once it has been executed, in the **Enter Call Parameter** parameter screen form.

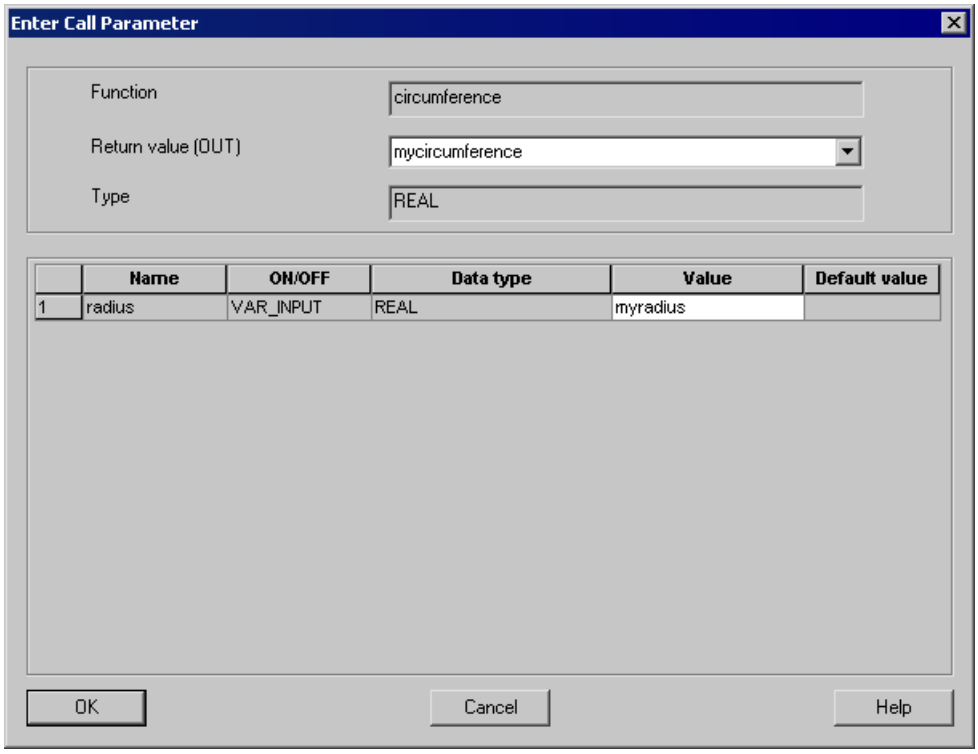


Figure 7-492 Parameterization of a function's subprogram call

Note

The LAD/FBD editor saves information on the origin of the subprogram when it is inserted in the network. This has no effect on the code generation.

This way you can subsequently move the source of a subprogram from the device to a connected library without changing the parameters. The LAD/FBD unit is still compiled correctly.

A check as to whether an identifier of a subprogram is hidden by an identifier with the same name in the LAD/FBD unit, is not performed. You can avoid name conflicts by using namespaces.

Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the LAD/FBD programs in an LAD/FBD unit. A subprogram (function, function block or program ("program in program")) must be defined before it is used. This is the case when the subprogram appears above the LAD/FBD program in which it is used in the project navigator. If necessary, reorder the LAD/FBD programs.

See also: Subprogram call of the function (FC) (Page 5235)

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC) or a function block (FB).

Overview of parameters for

You can set the following parameters when parameterizing the subroutine call:

Overview of Subroutine call parameters

Field/Button	Explanation/instructions
Subroutine type/ Subroutine	The type and name of the subroutine are displayed here: **Function** A function is a sub-program that has no memory beyond the call. In other words, the data is lost once the function has run and the output parameters and return value have been transferred. A function can also have a return value (function value) in addition to input, in/out and output parameters. **Function block** A function block is a sub-program which has a memory beyond the call. In other words, the values of the function block are retained after it has run. Variables of the data type for this function block must be declared. These variables are identified as instances of the function block; they include the memory of the function block beyond the calls. A function block only has input, in/out and output parameters but does not have a return value. The output parameters of the function block can be accessed even at a later time (after the function block has run) at the function block instance. **Program ("program in program")** You also have the option of calling a program within a different program or a function block. This requires the following compiler options to be activated, see Global compiler settings (Page 5111) and Local compiler settings (Page 5112): • "Permit language extensions" for the program source of the calling program or function block and • "Only create program instance data once" for the program source of the called program. The static data of the called program is stored in the user memory of the program source (unit) of said called program. The same program instance data is used every time the program is called. A program is called without parameters or return values.
Subprogram type/ subprogram (continued)	**Method** Methods are structuring tools for object-oriented programming. They largely correspond with functions, but are encapsulated in higher-level structuring tools (classes, function blocks). You can only generate methods and classes in the Structured Text (ST) programming language. Nevertheless classes and public methods (access identifier PUBLIC) can be used in LAB/FBD. A SIMOTION Kernel as of version V4.5 is mandatory for classes and their methods. Activating the following compiler options is recommended, see Global settings of the compiler (Page 5111) and Local settings of the compiler (Page 5112): • "Permit object-oriented programming" for the LAD/FBD source file of the LAD/FBD program being called. Variables of the data type for the higher-level structuring tools (class, function block) must be declared. These variables are identified as instances, and include the memory of the higher-level structuring tool beyond the calls. The methods themselves have no memory beyond the call. In other words, the data is lost once the method has run and the output parameters and return value have been transferred. A method can also have a return value (function value) in addition to input, in/out and output parameters.

Field/Button	Explanation/instructions
Instance	For subprogram type "function block" or “method": • In the case of a “function block” subprogram type: Here, enter the name of the function block instance. The instance contains the memory of the function block in the form of instance data. You define the instance as a variable whose data type is the name of the function block in one of the following ways: – In the declaration table of the LAD/FBD source as VAR_GLOBAL or – In the declaration table of the LAD/FBD program as VAR. • The following applies to “method” subprogram types: Enter the name of the class or function block instance to which the method belongs here. The instance contains the memory of the class or the function block in the form of instance data. You define the instance as a variable whose data type is the name of the class or of a class derived from this or the name of the function block in one of the following ways: – In the declaration table of the LAD/FBD source as VAR_GLOBAL or – In the declaration table of the LAD/FBD program as VAR.
Return value	For subprogram type "Function" or “Method": Here, you enter the variable in which the return value is to be stored. The type of variable must match the return value type.
Type	For subprogram type "Function" or “Method": The data type of the return value is displayed.
List of transfer parameters	
Name	The name of the transfer parameter is displayed here.
On / Off	The variable type of the transfer parameter is displayed here. **VAR_INPUT** Input parameters (for functions, function blocks and methods) **VAR_IN_OUT** In/out parameters (for functions, function blocks and methods) **VAR_OUTPUT** Output parameters (for functions, function blocks and methods)

Field/Button	Explanation/instructions
Data type	The data type of the transfer parameter is displayed here.
Value	Mandatory parameters are marked with "<???>" and optional parameters with "...". • The following applies to functions (FC) and methods: – Input parameters without a declared initial value and in/out parameters are mandatory parameters. – Input parameters with a declared initial value and output parameters are optional parameters. A subroutine call will only be functional if all the mandatory parameters are set. • The following applies to function blocks (FB): All parameters are optional. • The following applies to programs ("program in program"): No parameters present. Here, you can assign current variables or values to the transfer parameters: • Input parameter (variable type VAR_IN): Here, you enter a variable name or an expression. The assignment of system variables or I/O variables is permissible; type transformations are possible. • In/out parameter (variable type VAR_IN): Enter a variable name; the variable must be directly writable and readable. System variables of SIMOTION devices and technology objects are not permitted nor are I/O variables. The data type of the in/out parameter must correspond to that of the assigned variables; application of type transformation functions is not possible. • Output parameter (variable type VAR_OUTPUT): The assignment of an output parameter to a variable is optional. – The following applies to function blocks: You can also access an output parameter after executing the function block. – The following applies to functions and methods: The values for the output parameters are lost after executing the function or method unless they have been assigned in the parameter screen. When assigned in this parameter screen form: Enter a variable name. The data type of the output parameter must correspond to that of the assigned variables; the application of type transformation functions is not possible.

Example: Function (FC)

You want to create a subroutine with a circumference calculation for a circle. The calculation is performed in a function (FC). This is named **Circumference**.

The circle circumference calculation can thus be called as a subroutine by any task.

Formula for circumference calculation: Circumference = PI * 2 * radius

You define the Radius and PI variables in the declaration table of the function.

Creating and programming the function (FC)

1. In the project navigator, open the LAD/FBD unit in which you want to create the function.
2. Double-click the entry **Insert LAD/FBD program**.
   - Enter the name **Circumference**.
   - For creation type, select **Function**.
   - For return type (data type of return value), select **REAL**.
   - Click **OK** to confirm.
3. In the declaration table, define the **radius** input parameters, the **diameter** parameter, and the **PI** constant.

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	radius	VAR_INPUT	REAL			
2	PI	VAR CONSTANT	REAL		3.14159	
3	diameter	VAR	REAL			
4						

Figure 7-493 Declaring variables (e.g. input parameters) in the LAD/FBD program

4. Click the **Insert Network** button on the LAD editor toolbar.
   A network is inserted into the **Circumference** function.
5. Drag the LAD/FBD element **MUL** from the command library and drop it into the network of the **circumference** function twice.
6. Program the circumference calculation for the return value by assigning the variables accordingly to the input/output parameters of the two **MUL** LAD/FBD elements.

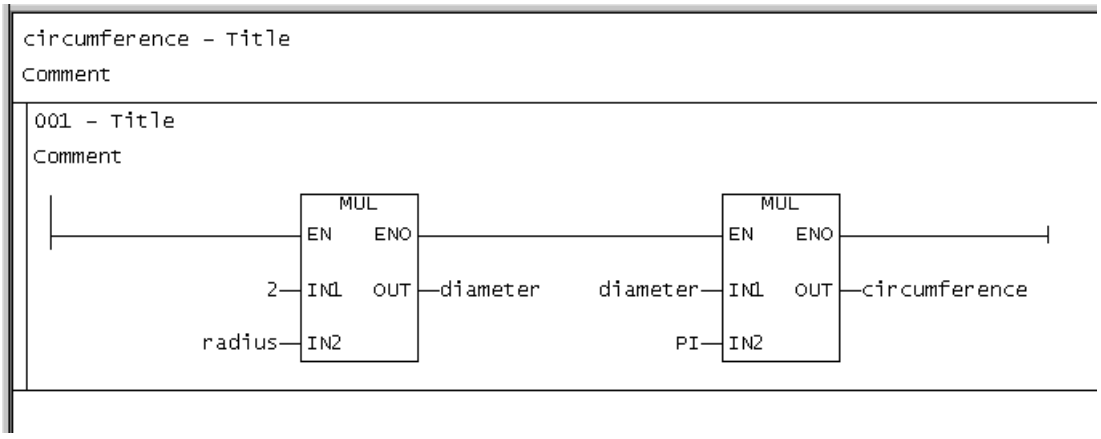


Figure 7-494 Programming the **Circumference** subroutine (e.g. assignment to a return value)

7. Accept and compile the LAD/FBD unit.

You have now finished programming the **Circumference** function.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

Subroutine call of function (FC)

The function (FC) is called from a program in the example.

1. Create an LAD/FBD program as a program in the same LAD/FBD unit (see Inserting a new LAD/FBD program (Page 5117)):
   - Enter the name **Program_circumference**.
   - For creation type, select **Program**.
   - Click **OK** to confirm.
2. Declare the following in the LAD/FBD unit or the LAD/FBD program:
   - The **mycircum** variable.
     The return value of the "Circumference" function is assigned to this variable.
   - The **myradius** variable.
     This variable contains the radius and is assigned to the input parameter **Radius** of the **Circumference** function.

   Note that the validity range of the variables is dependent on the declaration location (see Define variables (Page 5171)).

	Name	Variable type	Data type	Array length	Initial value	Comment
1	mycircum	VAR	REAL			
2	myradius	VAR	REAL			
3						

① You can continue to use the myumfang (mycircum) variable in the program.

Figure 7-495 Declaring a variable in the LAD/FBD program

3. Click the **Insert Network** button on the LAD editor toolbar.
   A network is inserted into the **Program_circumference** program.
4. Drag the **Circumference** function from the project navigator and drop it into the network of the **Program_circumference** program.
5. Select the inserted function, **Circumference**, followed by the **Parameterize call** command from the context menu.

6. Assign parameters to the subroutine call in the **Enter Call Parameter** parameter screen form.

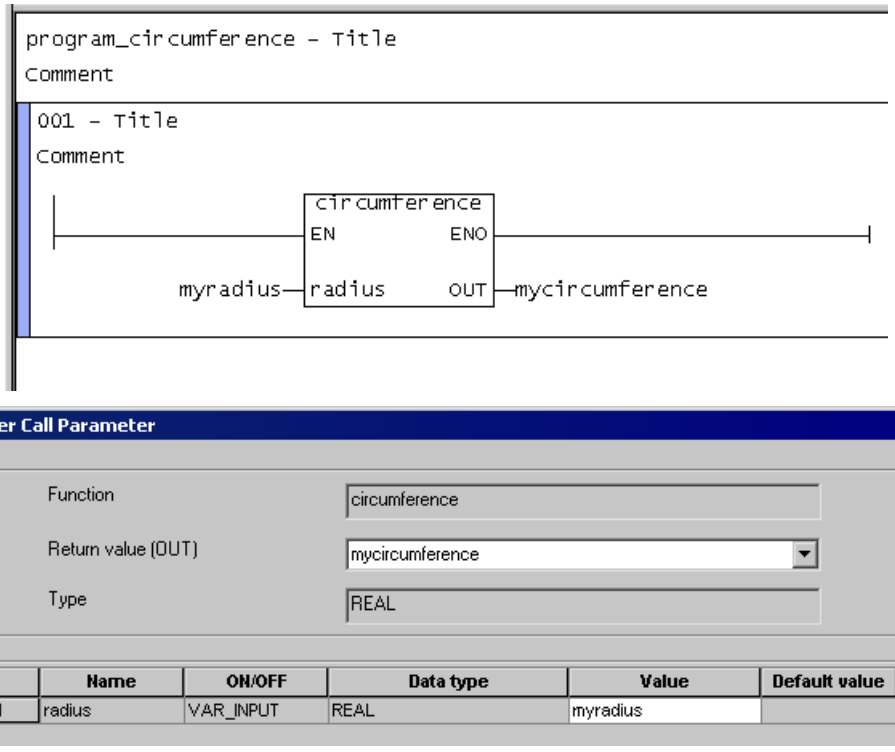


Figure 7-496 Opened parameter screen form for assigning parameters to the subroutine call

Note

Mandatory parameters are marked with "<???>" and optional parameters with "...". A subroutine call will only be functional if all the mandatory parameters are set.

7. Accept and compile the LAD/FBD unit.

You have now finished programming the subroutine call.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

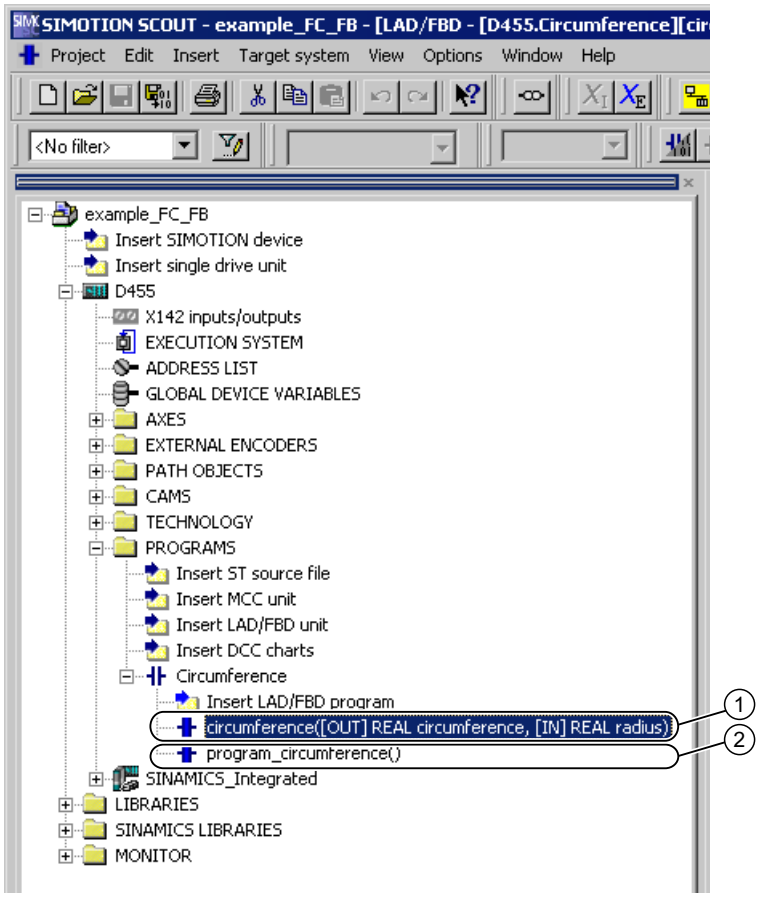
Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the LAD/FBD programs in the LAD/FBD unit (see figure below). A subroutine (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the MCC chart of the subroutine appears in the project navigator above the chart in which it is used.

As far as the example described here is concerned, the LAD/FBD program with the function (FC) must appear in the project navigator above the LAD/FBD program containing the program with the subroutine call.

In other words, the **circumference** function must be positioned above the **program_circumference** program in the project navigator. If necessary, reorder the LAD/FBD programs by selecting the relevant LAD/FBD program in the project navigator, then selecting the **Down** or **Up** command in the context menu.



① **Circumference** function

② **Program_circumference** program, which contains the subroutine call of the **Circumference** function

Order of LAD/FBD programs

Opening the function (FC) directly from the subroutine call

You can open subprograms directly from their respective subprogram call by using the context menu:

- This works regardless of the programming language (ST, MCC, LAD/FBD) used to create the subprogram.
- The subprogram opens in a separate editor or an editor already opened for the subprogram is moved to the foreground.

A subprogram may be a:

- User-defined function (FC)
- User-defined function block (FB)
- User-defined program called as a subprogram ("program in program")
- Library function
- Library function block
- Library program called as a subprogram ("program in program")

Procedure

To open the FC directly from the subprogram call, proceed as follows:

1. In the LAD/FBD network, select the subprogram call used to call the FC.
2. Select the **Open called block** command in the context menu (Ctrl+Alt+O shortcut).

The FC opens in a separate editor or an editor already opened for the FC is moved to the foreground.

Note

If the called FC has not yet been created, the **Open called block** command appears inactive (grayed out) in the context menu.

Note

If the called FC is in a program source with know-how protection, the same steps that apply when opening the protected program source directly also apply here (see Know-how protection for LAD/FBD units (Page 5106)).

Example: Function block (FB)

You want to calculate a following error. The calculation is performed in a function block (FB) named **FollError**. The following error calculation can thus be called as a subroutine by any task.

Formula for following error calculation: Difference = Specified position – Actual position

Define the required input and output parameters Set position, Actual position, and Difference (with the other variables, if necessary) in the LAD/FBD program (function block) or LAD/FBD unit.

Creating and programming the function block (FB)

1. In the project navigator, open the LAD/FBD unit in which you want to create the function block.
2. Double-click the entry **Insert LAD/FBD program**.
   - Enter the name **FollError**.
   - For creation type, select **Function block**.
   - Confirm with **OK**.
3. In the declaration table, define the variables (e.g. input and output parameters).

	Name	Variable type	Data type	Array length	Initial value	Comment
1	Setpoint_position	VAR_INPUT	LREAL			
2	Actual_position	VAR_INPUT	LREAL			
3	Difference	VAR_OUTPUT	LREAL			
4						

Figure 7-497 Declaring variables (e.g. input and output parameters) in the LAD/FBD program

4. Click the **Insert network** button on the LAD editor toolbar.
   A network is inserted into the **FollError** function block.
5. Drag the LAD/FBD element **SUB** from the command library and drop it into the network of the **FollError** function block.
6. Program the following error calculation by assigning the variables accordingly to the input/output parameters of the **SUB** LAD/FBD element.

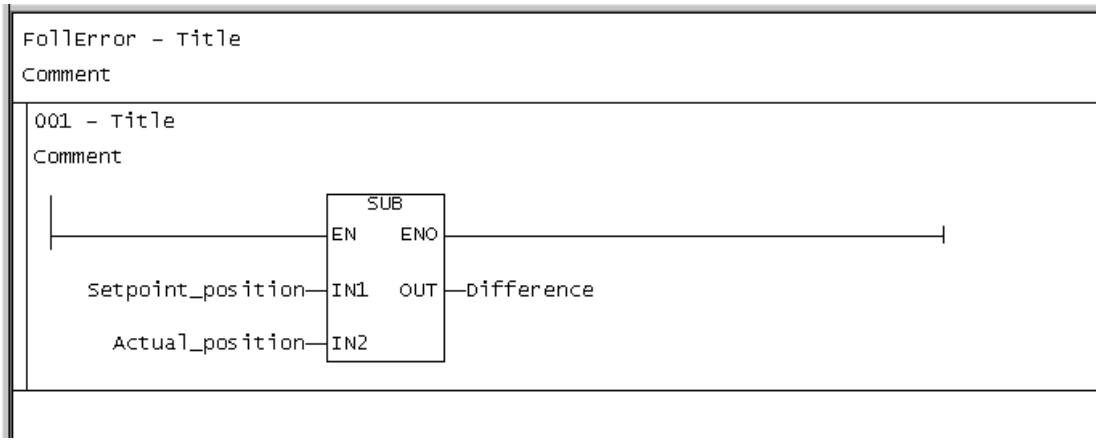


Figure 7-498 Programming the following error calculation

7. Accept and compile the LAD/FBD unit.

You have now finished programming the **FollError** function block.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

Subroutine call of function block (FB)

In this example, the function block (FB) is called from a program.

1. Create an LAD/FBD program as a program (see Inserting a new LAD/FBD program (Page 5117)).
2. Create a function block instance.
   – In the LAD/FBD unit or LAD/FBD program, declare the instances of the function block along with the variables.

   Note that the validity range of the instance and variables is dependent on the declaration location (see Define variables (Page 5171)).
3. Call the function block:
   – Program the subroutine call.
4. After executing an instance of the function block, you can access the output parameters at any location in the calling program.
   – Program the **MOVE** command.
5. Accept and compile the program.

You have now finished programming the subroutine call.

Creating a function block instance

Before you can use a function block, you must define an instance. Each instance of an FB is independent of the others; once an instance has ended, its static variables remain stored.

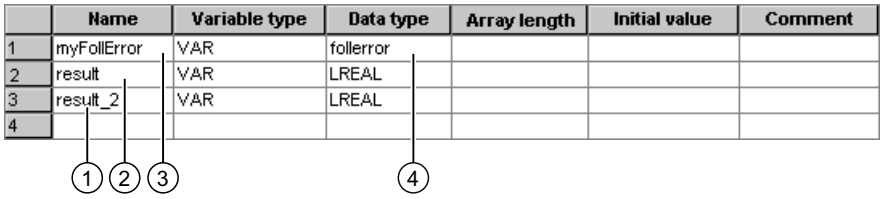
Instances of an FB are defined in the declaration tables of the LAD/FBD source or of the LAD/FBD program. The scope of the instance declaration is dependent on the location of the declaration:

- In the interface section of the declaration table of the LAD/FBD source:
  The instance behaves like a unit variable; it is valid for the entire LAD/FBD source; all LAD/FBD programs (programs, function blocks, and functions) within the LAD/FBD source can access the instance.
  The instance is also available:
  – On HMI devices
  – In other LAD/FBD source files (or other units) once connected, see “How to define connections to other units (program sources)” (Page 5223).

  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the LAD/FBD source:
  The instance behaves like a unit variable which is only valid in the LAD/FBD source; all LAD/FBD programs (programs, function blocks, and functions) within the LAD/FBD source can access the instance.
- In the declaration table of the LAD/FBD program (for programs and function blocks only):
  The instance behaves like a local variable; it can only be accessed within the LAD/FBD program in which it is declared.

Proceed as follows (the LAD/FBD source or the LAD/FBD program with the declaration table is open, see "Opening an existing LAD/FBD source" (Page 5104) or "Opening an existing LAD/FBD program" (Page 5119):

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Parameters** tab.
3. Enter or select the following:
   - **Name** of instance (variable name – see Rules for identifiers (Page 5155))
   - **Variable type** VAR or VAR_GLOBAL, depending on the declaration location (in LAD/FBD program or LAD/FBD source respectively)
   - Designation of the function block as **data type**.
4. Declare the other variables.



① The output parameter Difference is assigned to the variable Result_2 during subsequent program runtime. You can use the Result_2 variable for other purposes in the program.

② The output parameter Difference is assigned to the variable Result in the subroutine call. You can use the Result variable for other purposes in the program.

③ Creating an instance

④ Select the required FB as the data type.

The created function blocks are offered as data types in the drop-down list box depending on the LAD/FBD editor settings (Page 5099):

- Only function blocks with the same program source or from connected program sources or libraries
- All function blocks defined in the project

Figure 7-499 Defining an instance of the function block and variables in the LAD/FBD program or the LAD/FBD unit

Programming the subroutine call of the function block

1. Drag the **FollError** function block from the project navigator and drop it into the network of the **program_FollError** LAD/FBD program.
2. Select the inserted function block, **FollError**, followed by the **Display > All Box Parameters** command from the context menu.
   All the input/output parameters of the inserted function block **FollError** are shown.

Note

Mandatory parameters are marked with "???" in the LAD/FBD network and optional parameters with "...". Function blocks (FBs) only have optional parameters.

3. Select the inserted function block, **FollError**, followed by the **Parameterize call** command from the context menu.

4. Assign parameters to the subroutine call in the **Enter Call Parameter** parameter screen form:
   – Enter the instance **myfollerror**, defined in the declaration table, in the **Instance** field.

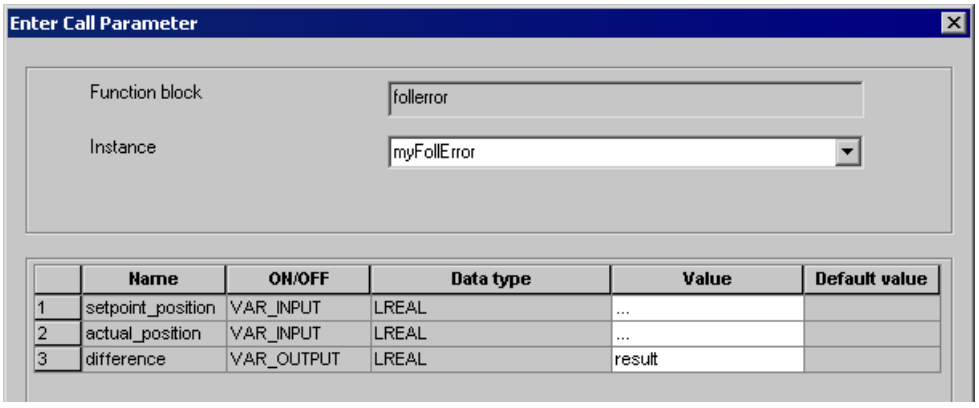


Figure 7-500 Opened parameter screen form for assigning parameters to the subroutine call

Note

Mandatory parameters are marked with "<???>" and optional parameters with "...". Function blocks (FBs) only have optional parameters.

Assign the current values to the transfer parameters:

– Input parameters: Variable or expression
– In/out parameter: Directly readable/writable variable
– Output parameter (optional): Variable
  In the example, for the **Difference** output parameter select the **Result** variable in the **Value** column.

You can use drag-and-drop to assign unit variables and system variables from the detail view to the input, output, or in/out parameters of the instance of the **FollError** function block inserted into the LAD/FBD network.

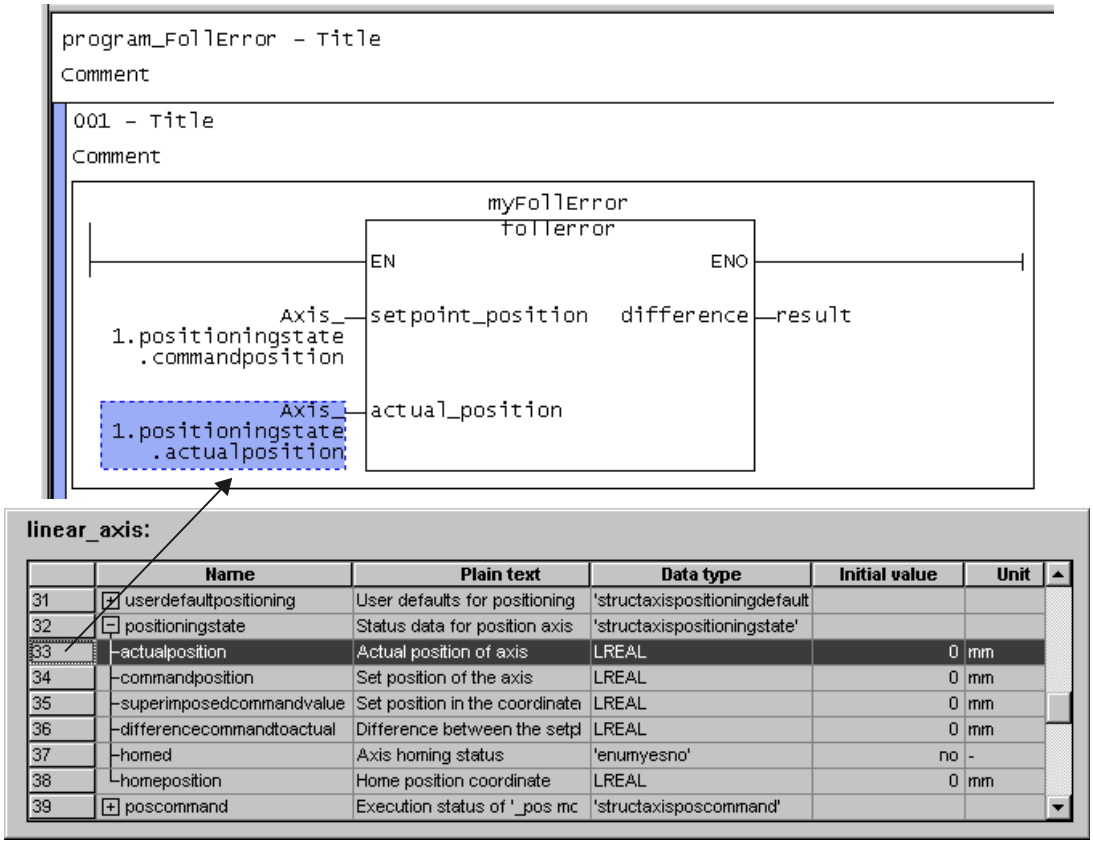


Figure 7-501 Assigning system variables from the detail view to the transfer parameters using drag-and-drop

5. Confirm with **OK**.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

Note

Saving a project in a format older than Version V4.2 of SIMOTION SCOUT

Pay attention to the order of the LAD/FBD programs in the LAD/FBD unit. A subroutine (function, function block, or program ("program in program")) must be defined before it is used. This is the case when the LAD/FBD program of the subroutine appears in the project navigator above the LAD/FBD program in which it is used.

As far as the example described here is concerned, the LAD/FBD program with the function block (FB) must appear in the project navigator above the LAD/FBD program containing the program with the subroutine call.

In other words, the **distance** function block must be positioned above the **program_distance** program in the project navigator. If necessary, reorder the LAD/FBD programs by selecting the relevant LAD/FBD program in the project navigator, then selecting the **Down** or **Up** command in the context menu.

See also: Subroutine call of the function (FC) (Page 5235).

Opening the function block (FB) directly from the subroutine call

You can open subprograms directly from their respective subprogram call by using the context menu:

- This works regardless of the programming language (ST, MCC, LAD/FBD) used to create the subprogram.
- The subprogram opens in a separate editor or an editor already opened for the subprogram is moved to the foreground.

A subprogram may be a:

- User-defined function (FC)
- User-defined function block (FB)
- User-defined program called as a subprogram ("program in program")
- Library function
- Library function block
- Library program called as a subprogram ("program in program")
- Class method
- Interface method

Procedure

To open the FB directly from the subprogram call, proceed as follows:

1. In the LAD/FBD network, select the subprogram call used to call the FB.
2. Select the **Open called block** command in the context menu (Ctrl+Alt+O shortcut).

The FB opens in a separate editor or an editor already opened for the FB is moved to the foreground.

Note

If the called FB has not yet been created, the **Open called block** command appears inactive (grayed out) in the context menu.

Note

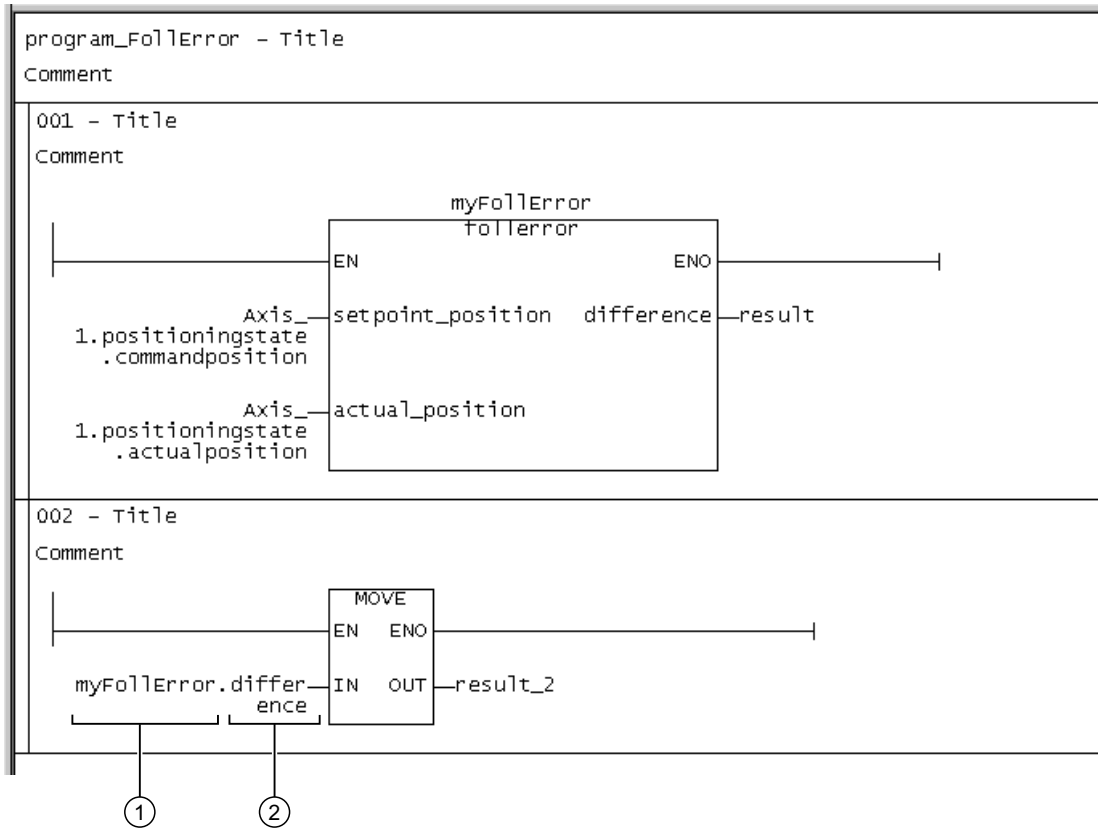
If the called FB is in a program source with know-how protection, the same steps that apply when opening the protected program source directly also apply here (see Know-how protection for LAD/FBD units (Page 5106)).

Accessing the output parameters of the function block retrospectively

After an instance of the function block has been executed, the static variables of the function block (including the output parameters) are retained. You can access the output parameters at any point in the calling program.

If you have defined the FB instance as VAR_GLOBAL, you can also access the output parameter in other LAD/FBD programs.

1. Insert the **MOVE** command into the LAD/FBD program.
2. Program the command (see figure).



① Name of the FB instance ② Output parameter

Figure 7-502 Programming a variable assignment

Example: Method

Example: Methods

In LAD/FDB programs you can call up public methods which have been defined in ST source files within classes and function blocks.

You can use methods in function blocks irrespective of the SIMOTION Kernel version.

Methods in classes can only be used in SIMOTION Kernel as of version V4.5.

No classes or methods can be defined in the LAD/FBD programming language itself. The same applies to methods within function blocks.

Requirements

In an ST source file the COUNTER class is defined with both public methods UP and DOWN: Both methods increment or decrement an internal counter variable until either the maximum value or minimum value is reached.

- The UP method has the following interface:
  - Input parameter INC with data type INT and initial value 1.
    You enter the increment here (standard = 1).
  - Output parameter QU with data type BOOL and initial value FALSE.
    The maximum value is reached when QU = TRUE.
  - Return value with data type INT.
    The return value shows the latest status for the internal counter variables.
- The DOWN method has the following interface:
  - Input parameter DECC with data type INT and initial value 1.
    You enter the increment here (standard = 1).
  - Output parameter QU with data type BOOL and initial value TRUE.
    The minimum value is reached when QU = FALSE.
  - Return value with data type INT.
    The return value shows the latest status for the internal counter variables.

The maximum value and minimum value for the counting range are stipulated in the private variables MAX_Val and MIN_Val for the class. They are pre-assigned with 100 or 0 respectively as standard. However, the initial values can be overwritten with the declaration of an instance for this class.

Subprogram call of the method

The method is called from a program in the example.

1. Generate an LAD/FBD program as a program (see Inserting a new LAD/FBD program (Page 5117)).
2. Create an instance for the higher-level class.
   - In the LAD/FBD source or LAD/FBD program, declare the instances of the class along with the variables.

   Note that the validity range of the instance and variables is dependent on the declaration location (see Define variables (Page 5171)).
3. Call the method:
   - Program the **Subroutine call** command.
4. Accept and compile the program.

You have now finished programming the subroutine call.

Creating an instance for the class or the function block

Before you can use a method, you must define an instance for the higher-level class or the higher-level function block. Each instance of a class or FB is independent of the others; once an instance has ended, its static variables remain stored.

Instances of a class or of an FB are defined in the declaration tables of the LAD/FBD source or of the LAD/FBD program. The scope of the instance declaration is dependent on the location of the declaration:

- In the interface section of the declaration table of the LAD/FBD source:
  The instance behaves like a unit variable; it is valid for the entire LAD/FBD source; all LAD/FBD programs (programs, function blocks, and functions) within the LAD/FBD source can access the instance.
  The instance is also available:
  - On HMI devices.
  - In other LAD/FBD source files (or other units) once connected, see "How to define connections to other units (program sources)" (Page 5223).

  The total size of all unit variables in the interface section is limited to 64 Kbytes.
- In the implementation section of the declaration table of the LAD/FBD source:
  The instance behaves like a unit variable which is only valid in the LAD/FBD source; all LAD/FBD programs (programs, function blocks, and functions) within the LAD/FBD source can access the instance.
- In the declaration table of the LAD/FBD program (for programs and function blocks only):
  The instance behaves like a local variable; it can only be accessed within the MCC chart in which it is declared.

Proceed as follows (the LAD/FBD source or the LAD/FBD program with the declaration table is open, see "Opening an existing LAD/FBD source" (Page 5104) or "Opening an existing LAD/FBD program" (Page 5119)):

1. Select the declaration table and, if applicable, the section of the declaration table for the desired scope.
2. Select the **Parameters** tab.
3. Enter or select the following:
   - **Name** of instance (variable name – see Rules for identifiers (Page 5155))
   - **Variable type** VAR or VAR_GLOBAL, depending on the declaration location (in LAD/FBD program or LAD/FBD source respectively)
   - Designation of the function block as **data type**.
4. Declare the other variables.

Programming the subprogram call of the method

1. From the project navigator drag the **UP** method from the **COUNTER** class to the network for the LAD/FBD program **KOP/FUP_1**.
2. Select the **counter.up** method inserted and select the **Display > All box parameters** command from the shortcut menu.
   All input/output parameters for the **UP** method inserted are displayed.

   Note

   Mandatory parameters are marked with "???" in the LAD/FBD network and optional parameters with "...".

3. Select the inserted **counter.up** method followed by the **Assign parameters to a call** command from the shortcut menu.
4. Assign parameters to the subroutine call in the **Enter Call Parameter** parameter screen form:
   - Enter the instance **C1** defined in the declaration table in the **Instance** field.
   - Select the **CountOut** variable in the return value field defined in the declaration table.

   Note

   Mandatory parameters are marked with "<???>" and optional parameters with "...".

5. Assign the current values to the transfer parameters:
   - Input parameters: Variable or expression
   - In/out parameter: Directly readable/writable variable
   - Output parameter (optional): Variable
     In the example for the output parameters **QU** in the **Value** column select the variable **UpValReached**.

   You can use drag-and-drop to assign unit variables and system variables from the detail view to the input, output, or in/out parameters of the instance of the **counter.up** method inserted into the LAD/FBD network.
6. Confirm with **OK**.

Note

The term "LAD/FBD program" is a generic term and may refer to a program, a function (FC), or a function block (FB).

Note

Saving a project in a format older than Version V4.5 of SIMOTION SCOUT

Projects which include classes and methods cannot be saved in a format older than Version V4.5 of SIMOTION SCOUT.

Opening the method directly from the subprogram call

You can open subprograms directly from their respective subprogram call by using the context menu:

- This works regardless of the programming language (ST, MCC, LAD/FBD) used to create the subprogram.
- The subprogram opens in a separate editor or an editor already opened for the subprogram is moved to the foreground.

A subprogram may be a:

- User-defined function (FC)
- User-defined function block (FB)
- User-defined program called as a subprogram ("program in program")
- Library function
- Library function block
- Library program called as a subprogram ("program in program")

Procedure

To open the method directly from the subprogram call, proceed as follows:

1. In the LAD/FBD network, select the subprogram call used to call the method.
2. Select the **Open called block** command in the context menu (Ctrl+Alt+O shortcut).

The method opens in the ST editor or the ST editor opened for the method is placed in the foreground.

Note

If the called method is in a program source with know-how protection, the same steps that apply when opening the protected program source directly also apply here, see Know-how protection for LAD/FBD sources (Page 5106).

Limitations with advance signal switching

An output from an LAD/FBD element can only be connected in advance of an LAP/FBD element input if both the input and output are of data type BOOL. As a result, only Boolean advance signal switching is possible in a network.

An output parameter from an FB or a return value from an FC cannot be switched to an input parameter of a different FB/FC, i.e. Boolean advance signal switching is not possible here either.

Non-Boolean advance signal switching and output/input parameter switching with the FB/FC can be implemented with the aid of an additional network and a temporary variable. If the output from an LAD/FBD element cannot be assigned directly to the input of the other LAD/FBD element, then the former LAD/FBD element is added to this additional network. The same temporary variable is assigned to both output and input, and so the output and input are switched via the temporary variable.

Alternatively, this can also be implemented with just one network and a temporary variable, with the result that both LAD/FBD elements are in the same network.

Example of output/input parameter switching with FB/FC

In the ST programming language, with TO commands from the commands library the "commandid" input parameter can be assigned directly with the *_getcommandid* function.

This output/input parameter switch with FB/FC can be implemented in the LAD/FBD programming language with an additional network for the *_getcommandid* function and a temporary variable.

The *_getcommandid* function is added to the upper network, and the temporary variable "var_commandid" is assigned to its output "OUT". The TO command *_pos* is added to the lower network, and the temporary variable "var_commandid" is likewise assigned to its input "commandid". The switching of the output "OUT" of *_getcommandid* and of the input "commandid" of *_pos* is thus effected using the temporary variable.

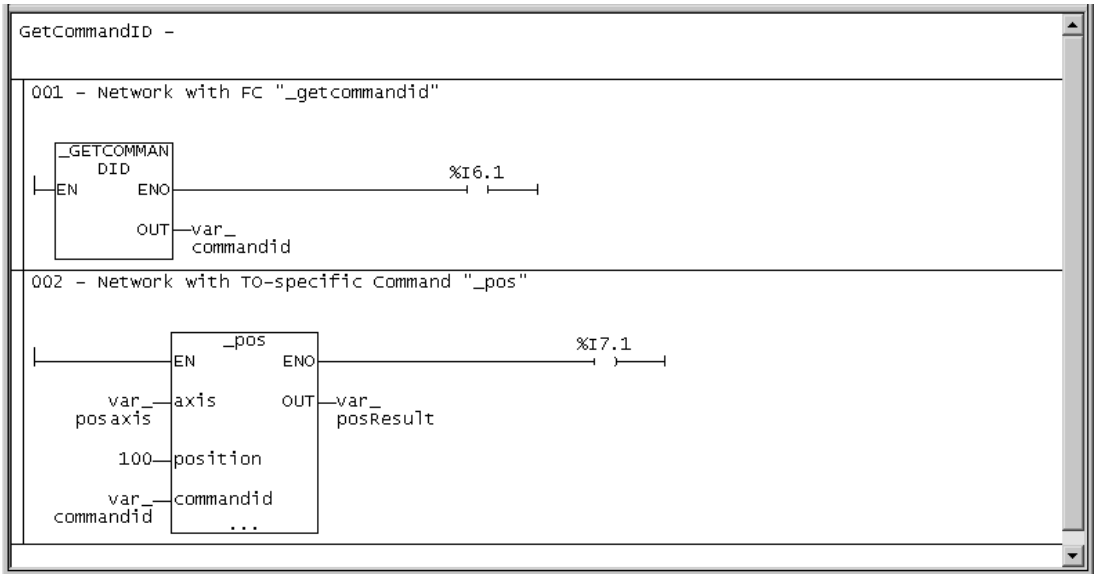


Figure 7-503 Output/input parameter switching with FB/FC

Interface adjustment with FB/FC

If the properties of an FB/FC that has been added to the network are modified, then in the following cases there will be an interface adjustment:

1. One or more new input/output parameters are added
2. One or more unused input/output parameters are deleted
3. One or more used input/output parameters are deleted

In cases 1 and 2 the network is updated immediately when it is opened in the LAD/FBD editor, or immediately after it is opened.

In case 3 the FB/FC call is shown in red in the network, and a manual update must be carried out. An existing Boolean advance switching of a deleted input parameter is not deleted; instead it is

merely separated off and, for instance, shown in the LAD display as an as an open ladder diagram in the network.

Manual update of a specific FB/FC call

To manually update a particular FB/FC call, follow these steps:

1. Click on the desired FB/FC call.
2. Select the **Update call** command in the context menu.
   The FB/FC call is shown with the input/output parameters which are currently present, i.e. without the deleted input/output parameters.

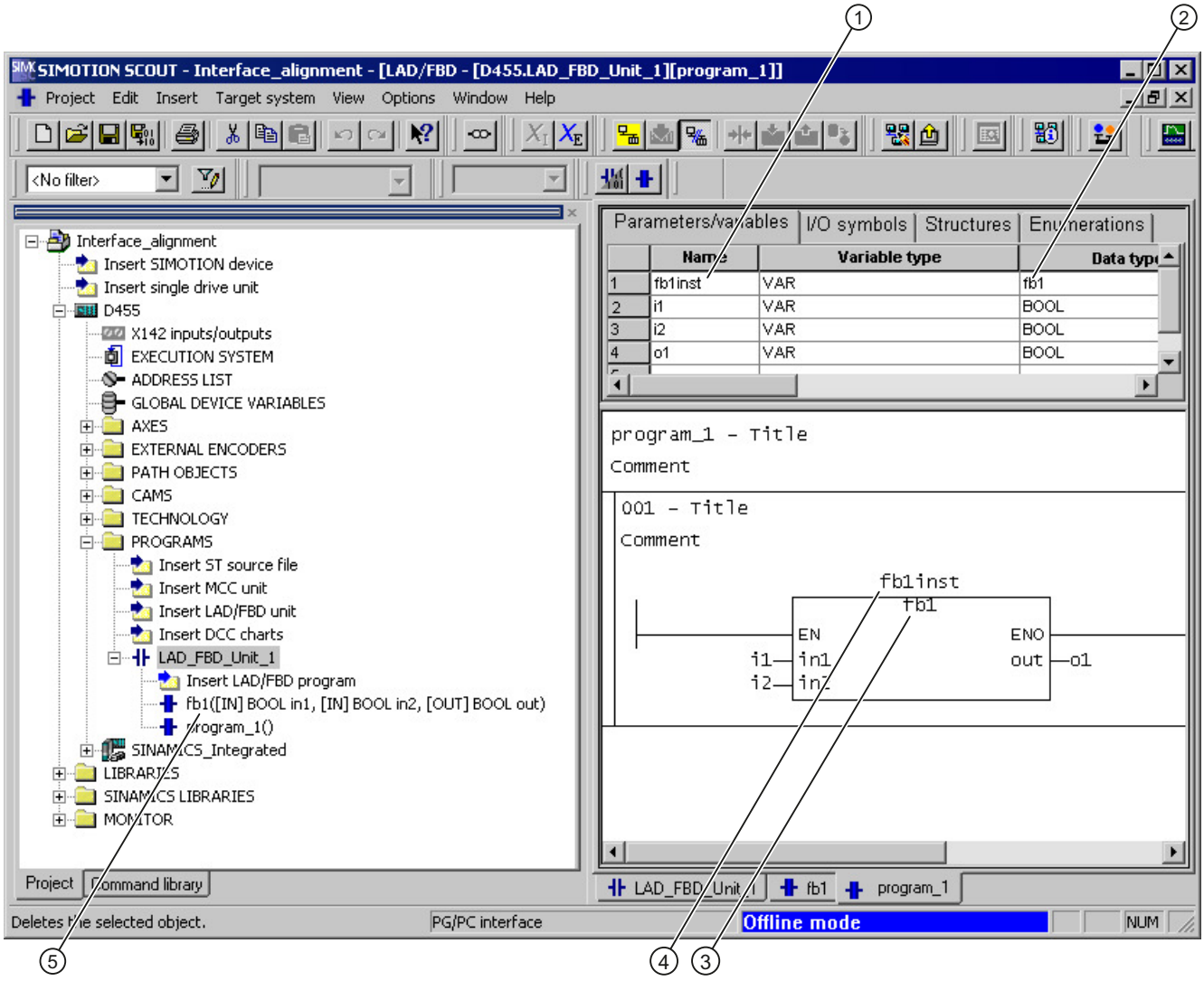
Manually updating all FB/FC calls

To manually update all the FB/FC calls for the program organization unit (POU) currently displayed in the working area, follow these steps:

1. Click on an empty position in the network.
2. Select the **Update all calls for all networks** command in the context menu.
   The FB/FC calls are shown with the input/output parameters which are currently present, i.e. without the deleted input/output parameters.

For the following cases there is no interface adjustment available, and so updating must be performed by entering data manually or with Find/Replace (Page 5263):

- Changing the name of the instance variable (only for FBs)
- Changing the name, the data type of the box type of the FB/FC call



① Instance variable (instance name)
Only used with FBs.
In the declaration table an FB instance is declared by specifying the instance variable with the name of the FB as the data type. This instance variable (instance name) is used for calling up the FB.

② Data type
The relevant FB is assigned to the instance variable as its data type.
In order for an FB call to work correctly, the data type and box type must be identical, otherwise the FB call will be displayed in red in the network.

③ Box type, consisting of:

- The type name, e.g. MOVE, ADD etc. or the names of user programs/FBs/FCs
- The type, i.e. the selected creation type (program, FB or FC) of the LAD/FBD program

④ Instance variable (instance name)
Only for FBs.

⑤ Name of the FB/FC (type of a (user-defined) FB/FC)
FB: The name of the FB is used as the data type in the FB instance declaration in the declaration table.
FC: The name of the FC is used as the box type.

Figure 7-504 Overview of FB/FC terms used

Display in the Detail view

Only in case 3 are the deleted input/output parameters and the deleted variables assigned to them displayed in the Detail view in the **Compile/check output** window immediately after a manual update. Deleted input parameters with Boolean advance switching are not displayed because the advance switching is merely separated off and therefore continues to exist in the network.

7.3.4.19 Reference data

The reference data provide you with an overview of:

- on utilized identifiers with information about their declaration and use (Cross-reference list (Page 5255)).
- on function calls and their nesting (Program structure (Page 5259))
- on the memory requirement for various data areas of the program sources (Code attributes (Page 5260))

Cross reference list

The cross-reference list shows all identifiers in program sources (e.g. ST source files, MCC units):

- Declared as variables, data types, or program organization units (program, function, function block)
- Used as previously defined types in declarations
- Used as variables in the statement section of a program organization unit.

Generating and updating a cross-reference list

Initially, the cross-reference list is generated automatically when opening. Open a cross-reference list, e.g. after selecting an ST source file or library via the **Edit > Display reference data > Cross-references** menu. After changes, the update is partly performed automatically and can always be triggered manually in the detail view via the button. When updating via the button, a check is performed as to whether a compilation is required. If a compilation is required, this is indicated by a yellow triangle next to the button.

Update of the cross-reference list when selecting a program

The list is updated automatically when opening the selected program. The local defined identifiers are therefore up-to-date.

External identifiers in the program are not updated when opening.

The opened cross-reference list is also updated automatically when compiling the program.

Update of the cross-reference list when selecting a tree (CPU, project, library, program folder)

The cross-reference list is generated automatically when opening the first time. There is no automatic update when opened again.

Note

An error-free compilation is required for a correct, consistent display of the reference data. If required, compile the project, the CPU, the program or the library first.

Content of the cross-reference list

The cross-reference list contains all the identifiers assigned to the element selected in the project navigator. The applications for the identifiers are also listed in a table:

Details of how to work with the cross-reference list are provided in the section titled "Working with the cross-reference list (Page 5258)".

Table 7-557 Meanings of columns and selected entries in the cross-reference list

Column	Entry in column	Meaning
Name		Identifier name
Type		Identifier type
	Name	• Data type of a variable (e.g. REAL, INT) • POU type (e.g. PROGRAM, FUNCTION)
	DERIVED	Derived data type
	DERIVED ANY_OBJECT	TO data type
	ARRAY ...	ARRAY data type
	ENUM ...	Enumerator data type
	STRUCT ...	STRUCT data type
Declaration		Location of declaration
	Name (UNIT)	Declaration in the program source *name*
	Name (LIB)	Declaration in the library *name*
	Name (TO)	System variable of the technology object *name*
	Name (TP)	Declaration in the default library specified: • Technology package *name* • std_fct = IEC library • device = device-specific library
	Name (DV)	Declaration on the SIMOTION device *name* (e.g. I/O variable or global device variable)
	_project	Declaration in the project (e.g. technology object)
	_device	Internal variable on the SIMOTION device (e.g. TaskStartInfo)
	_task	Task in the execution system
Use		Use of identifier
	CALL	Call as subprogram (static binding)
	CALL VTAB	Call of a method of the dynamic binding

Column	Entry in column	Meaning
	ENUM *name*	As element when declaring the enumerator data type *name*
	I/O	Declaration as I/O variable
	R	Read access
	R (TYPE)	As data type in a declaration
	R/W	Read and write access
	STRUCT *name*	As component when declaring the structure *name*
	TYPE	Declaration as data type or POU
	Variable type (e.g. VAR, VAR_GLOBAL)	Declaration as variable of the variable type specified
	W	Write access
Path specification		Path specification for the SIMOTION device or program source
	Name	SIMOTION device *name*
	Name1/*Name2*	• Program source *name2* on SIMOTION device *name1* • Program source *name2* in library *name1*
	Name/taskbind.hid	Execution system of the SIMOTION device *name*
Range		Range within the SIMOTION device or program source
	INTERFACE	Interface section of the program source
	POE type name (i.e. CLASS *name*, FUNCTION *name*, FUNCTION_BLOCK *name* INTERFACE *name* PROGRAM *name*)	Program Organization Unit (POU) *Name* within the program source. • In an MCC chart: Additional serial numbers for the command (block numbers) • In a LAD/FBD program: Additional serial numbers of the network
	I/O address	I/O variable
	METHOD *Name_1*::*Name_2*	Method *Name_2* within the class or the function block *Name_1*
	TASK *name*	Assignment for the task *name*
	UNIT	Implementation section of the program source, additional specification as POU to be made public in the interface section of the program source
	UNIT - IMPLEMENTATION	Implementation section of the program source
	_device	Global device variable
Language		Programming language of the program source
Line/Block		• In an ST source: Line number within the program source • In an MCC or LAD/FBD source: Relative line number within the command (block) or network. **Note** The absolute line number within the program source, which you need, for example, for the trace function "Trigger to code point", is obtained as follows: – Press the **Determine program line** button. – In the dialog box, press the button **Copy program line to the clipboard**.

Note

Single-step tracking and trace diagnostic functions in MCC programming

Additional variables and functions are created or used for these diagnostics functions:

- The variables TSI#dwuser_1 and TSI#dwuser_2 of the TaskStartInfo are used for the single-step tracking diagnostic function.
- Various internal functions and variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. The TSI#currentTaskId variable of the TaskStartInfo is also used.

With activated diagnostic function, these variables and functions are used for the control of the diagnostics function. These variables and functions must not be used in the user program.

Working with a cross-reference list

In the cross-reference list you are able to:

- Sort the column contents alphabetically:
  - To do this, click the header of the appropriate column.
- Search for an identifier or entry:
  - Click the "Search" button and enter the search term.
- Filter (Page 5258) the identifiers and entries displayed.
- Copy contents to the clipboard in order to paste them into a spreadsheet program, for example.
  - Select the appropriate lines and columns.
  - Press the CTRL+C shortcut.
- Print the content (**Project > Print**).
- Open the referenced program source and position the cursor on the relevant line of the ST source file (or MCC command or LAD/FBD element):
  - Double-click on the corresponding line in the cross-reference list.
    or
  - Place the cursor in the corresponding line of the cross-reference list and click the "Go to application" button.

Further details about working with cross-reference lists can be found in the online help.

Filtering the cross-reference list

You can filter the entries in the cross-reference list so that only relevant entries are displayed:

1. Click the "Filter settings" button.
   The "Filter Setting for Cross References" window will appear.
2. Activate the "Filter active" checkbox.

3. If you also want to display system variables and system functions:
   – Deactivate the "Display user-defined variables only" checkbox.
4. Set the desired filter criterion for the relevant columns:
   – Select the relevant entry from the drop-down list box or enter the criterion.
   – If you want to search for a character string within an entry: Deactivate the "Whole words only" checkbox.
5. Confirm with **OK**.

The contents of the cross-reference list will reflect the filter settings selected.

Note

A filter is automatically activated after the cross-reference list has been created.

Program structure

The program structure contains all the function calls and their nesting within a selected element.

You can display the program structure selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the program structure.
2. Select the **Edit > Display reference data > Program structure** menu command.
   The cross-reference tab is replaced by the program structure tab in the detail view.

Note

The display data is updated every time the program structure is opened.

You can update the detail view of an opened program structure with the F5 key.

Content of the program structure

A tree structure appears, showing:

- as base respectively
  – the program organization units (programs, functions, function blocks) declared in the program source, or
  – the execution system tasks used
- below these, the subroutines referenced in this program organization unit or task.

For structure of the entries, see table:

Table 7-558 Elements of the display for the program structure

Element	Description
Base (declared POU or task used))	List separated by a comma • Identifier of the program organization unit (POU) or task The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Identifier of the program source in which the POU or task was declared, with add-on [UNIT] • Minimum and maximum stack requirement (memory requirement of the POU or task on the local data stack), in bytes [Min, Max] • Minimum and maximum overall stack requirement (memory requirement of the POU or task on the local data stack including all called POUs), in bytes [Min, Max]
Referenced POU	List separated by a comma: • Identifier of called POU The specification *Name_1::Name_2* means: Method *Name_2* within the class or the function block *Name_1*. • Optionally: Identifier of the program source / technology package in which the POU was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library Add-on (TP): System function from technology package • with function blocks or methods only: Identifier of the instance of the function block or the higher-level class • with function blocks or methods only: Identifier of program source in which the instance of the function block or of the higher-level class was declared: Add-on (UNIT): User-defined program source Add-on (LIB): Library • Number of the line in the (compiled) source in which the POE is called; several line numbers are separated by "/".

Code attributes

You can find information on or the memory requirement of various data areas of the program sources under code attribute.

You can display the code attributes selectively for:

- An individual program source (e.g. ST source file, MCC unit, LAD/FBD source file)
- All program sources of a SIMOTION device
- All program sources and libraries of the project
- Libraries (all libraries, single library, individual program source within a library)

Proceed as follows:

1. In the project navigator, select the element for which you want to display the code attributes.
2. Select the **Edit > Display reference data > Code attributes** menu command.
   The **Cross-references** tab is now replaced by the **Code attributes** tab in the detail view.

Note

The display data is updated every time the code attributes are opened.

You can update the detail view of the opened code attributes with the F5 key.

Code attribute contents

The following are displayed in a table for all selected program sources:

- Identifier of program source,
- Memory requirement, in bytes, for the following data areas of the program source:
  - **Dynamic data**: All unit variables (retentive and non-retentive, in the interface and implementation sections),
  - **Retain data**: Retentive unit variables in the interface and implementation section,
  - **Interface data**: Unit variables (retentive and non-retentive) in the interface section,
- the **Code size** during the last compilation in bytes,
- the **Number of referenced sources**:
  The maximum number of connected sources is displayed (including system libraries), regardless of whether they are downloaded to the target system at a later date.

Reference to variables

If you have selected the identifier of a variable in the open Editor window for each programming language, you can use the other places of use over the context menu to list or to skip these variables.

You select the identifier of a variable:

- In SIMOTION ST: In the Editor window of an ST source.
- In SIMOTION MCC: In the input field on the parameter screen of an open MCC command within an MCC chart
- In SIMOTION LAD/FBD: In the Editor window of a LAD/FBD program

An identifier is recognized as a variable under the following conditions:

1. The identifier is declared as a variable. The scope of the variable includes the respective window (ST source, MCC chart, LAD/FBD program).
2. The program source is compiled.
3. The variable is selected as follows (in an open parameter screen within an MCC chart):
   – The identifier is fully marked
   or
   – The cursor is within the identifier.

Note

In arrays and structures, only the variable can be selected, not a single element.

Using the **Go to** context menu, you have the following options:

- To jump to the next local place of use:
  Select the context menu **Go to > Local use >>**.
  The next place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the previous local place of use:
  Select the context menu **Go to > Local use <<**.
  The previous place of use of the variables within the same Editor window (ST source, MCC chart, LAD/FBD program) is selected. In an MCC chart, the corresponding MCC command opens.
- Jump to the declaration position:
  Select the context menu **Go to > Declaration position**.
  The declaration position of the variables is selected. The corresponding program source is opened, if necessary.
- List all places of use
  Select the context menu **Go to > Places of use ...**.
  In the detailed view, all places of use of the variables within their scope (including the declaration position) are listed The structure of this list is similar to the List of cross references (Page 5256).
  This is how you jump to a preferred place of use:
  – Double-click on the corresponding line.
  or
  – Place the cursor in the corresponding line and click the "Go to application" button.

7.3.4.20 Find and replace

The following options are available when searching for a given piece of text or for variables only:

- Find or find and replace in the entire project (project-wide search).
  This is described in detail in the Online help.

  Note

  Following a find and replace operation in the whole project, it may be necessary to compile the project, so as to update all symbol information.

- Find or find and replace in a certain program source or their subprograms (local search).
  The following is described in this manual:
  - Finding in an LAD/FBD unit or an LAD/FBD program (Page 5263)
  - Finding and replacing in an LAD/FBD unit or an LAD/FBD program (Page 5264)

  See the respective manuals for a description of the local search in the other programming languages.

Find in LAD/FBD unit or LAD/FBD program

Range of the local search

With local searching, only the contents of the window (LAD/FBD unit or LAD/FBD program) in which the local search was triggered are searched.

- The following are searched in an LAD/FBD unit:
  - All tabs of the declaration table (INTERFACE and IMPLEMENTATION)

  The assigned LAD/FBD programs are **not** searched.
- The following are searched in an LAD/FBD program:
  - All tabs of the declaration table
  - Program title and comment
  - All networks (title, comment, LAD/FBD elements with parameter labelling)

The dialog box open during the local search is assigned to the respective window and is hidden when changing to another window and displayed again when returning.

Procedure

If you want to conduct local searching for arbitrary text in an LAD/FBD unit or LAD/FBD program, proceed as follows:

1. Open the desired LAD/FBD unit (Page 5104) or the LAD/FBD program (Page 5119).
2. In the menu, select **Edit > Find** (shortcut Ctrl+F).
   The **Find** dialog box opens.
   A character string selected in the LAD/FBD editor is automatically taken as a search term.

3. Enter the required search term in the **Find** input field.
   Wildcards such as "*" or "?" are not permitted.
4. If required, select a search option as well as the search direction (**Up**/**Down**).
   - If you activate the **Whole words only** checkbox, only a whole word is searched for.
   - If you activate the **Match case** checkbox, the search takes upper- and lower-case into account.
   - If you activate the **Variables only** checkbox, the search is performed only in fields that contain identifiers of variables.
5. Click the **Find next** button (shortcut F3).
   The search is started in the selected direction. The first matching text pattern is highlighted.
6. Click the **Find next** button again to display the next matching text pattern.

Note

You can also continue searching with F3 even when the dialog box is closed.

Displaying search results

- The relevant tab is automatically activated in a declaration table and the found search term is highlighted.
- The found search term is highlighted in the LAD/FBD program.

The respective editor window can also be edited when the dialog box is open.

Find and replace in LAD/FBD unit or LAD/FBD program

The process of finding and replacing on a local basis works like local finding (Page 5263), but also makes it possible to specify an expression to replace a search term once found.

Procedure

If you want to conduct local searching for arbitrary text in an LAD/FBD unit or LAD/FBD program, proceed as follows:

1. Open the desired LAD/FBD unit (Page 5104) or the LAD/FBD program (Page 5119).
2. In the menu, select **Edit > Replace** (shortcut Ctrl+H).
   The **Replace** dialog box opens.
   A character string selected in the LAD/FBD editor is automatically taken as a search term.
3. Enter the required search term in the **Find** input field.
   Wildcards such as "*" or "?" are not permitted.

4. If required, select a search option.
   - If you activate the **Whole words only** checkbox, the search looks for a whole word.
   - If you activate the **Match case** checkbox, the search takes upper- and lower-case into account.
   - If you activate the **Variables only** checkbox, the search is performed only in fields that contain identifiers of variables.
5. Enter the expression that should replace the search term in the **Replace with** input field.
6. Click the **Find next** button (shortcut F3).
   The search begins in the **Down** direction and the first matching text pattern is highlighted. The **Replace** button also becomes active if a replacement is permitted at this position, see Rules for replacing.
   - If you want to replace the search term found, click the **Replace** button.
     The search term found, now selected and displayed, is replaced and the next length of text to match the criteria is displayed.
   - If you do not want to replace the search term found, click the **Find next** button.
     The next length of text to match the criteria is displayed.

Rules for replacing

When replacing, a check is made as to whether the resulting term after replacement is permitted in principle at this position, for example:

- For identifiers of variables and data types, a check is made as to whether the Rules for identifiers (Page 5155) have been observed.
- For selection fields (combo boxes), a check is made whether the resulting term is available as a selection option. Some examples are shown below:
  - In the declaration table of an LAD/FBD unit, VAR_GLOBAL cannot be replaced by VAR or VAR_TEMP.
  - The LAD/FBD element ADD can be replaced by other mathematical functions, but not, for example, by MAX or a comparison operator.
  - A comparison operator can only be replaced by another comparison operator, but not, for example, by a mathematical or logical function.
- Generally, the following cannot be replaced:
  - Wildcards, such as "..." or "???"
  - The elements listed in the command library under "LAD elements" and "FBD elements".

Note

If an item cannot be replaced at a particular position, the **Replace** button appears inactive (grayed out).

Further checks are not made, for example

- Whether variables or data types have already been defined.
- Whether there are further dependencies between fields and will be violated by the replacement.

In declaration tables, some columns are generally blocked for replacing, for example:

- The **Absolute identifier** column in the **I/O symbols** tab
- The **Type** column in the **Connections** tab

7.3.4.21 Execution order

Non-optimized execution order

Requirements

The non-optimized execution order is active for LAD networks by default. In the "Global settings of the compiler" (Page 5111), the **Optimize execution order (LAD/FBD)** checkbox is inactive.

Calculation of the non-optimized execution order

The LAD network is calculated in the following order:

1. At the end of the last parallel branching (at ① in the example), the lower parallel branches are calculated first in the order 2 - 3 - 4 ... before the top parallel branch is calculated.
2. The elements are calculated from left to right within a parallel branch.
3. Within the lower parallel branches, further parallel branches (at ② in the example) are calculated in the normal order from top to bottom.
4. In the top parallel branch, further parallel branches (at ③ in the example) are calculated in the order described in 1.

This produces the specified order (1 ... 7) in the following example.

Example of a non-optimized execution order

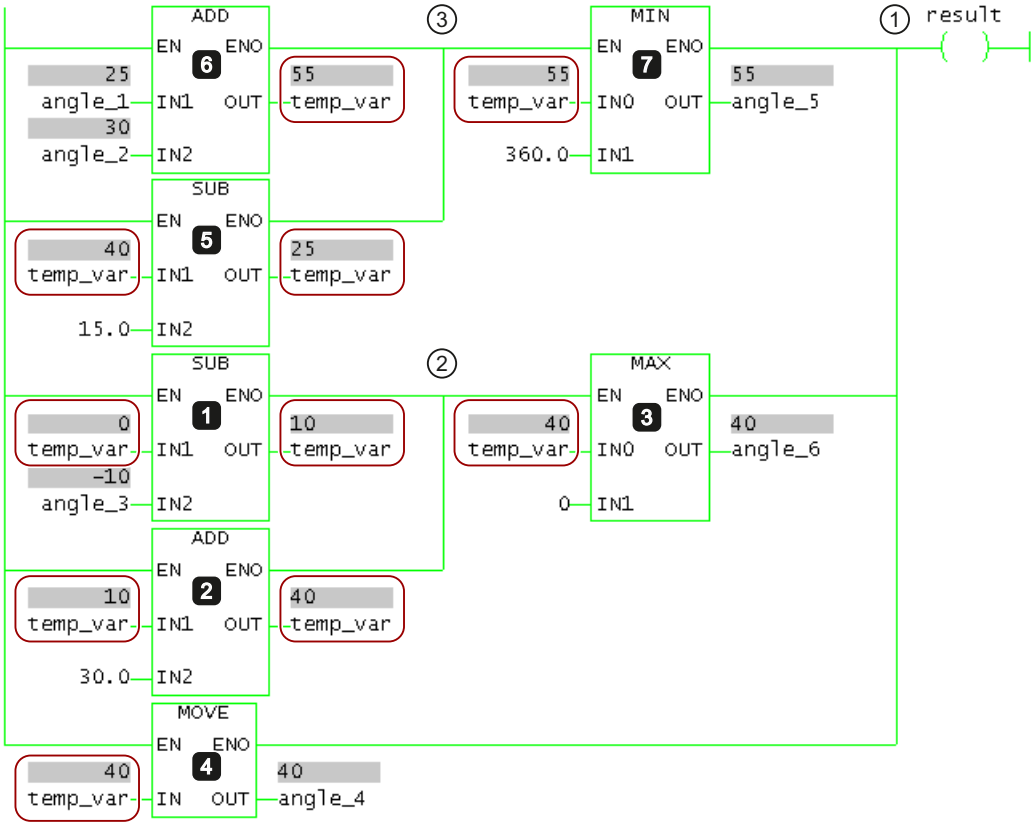


Figure 7-505 Example of a non-optimized execution order

Optimized execution order

Requirements

The setting for the optimized execution order is made globally for the entire project in the "Global compiler settings" (Page 5112). The **Optimize execution order (LAD/FBD)** checkbox is active.

Note

Note when saving the project in the old project format: In project formats up to version V4.2, the optimized execution order is not taken into account.

Calculation for the optimized execution order

The LAD network is calculated in the following order:

1. At the end of parallel branching (at ① in the example), the parallel branches are calculated from top to bottom.
2. The elements are calculated from left to right within a parallel branch.
3. Within the parallel branches, further parallel branches (at ② or ③ in the example) are calculated from top to bottom.

This produces the specified order (1 ... 7) in the following example.

Example of an optimized execution order

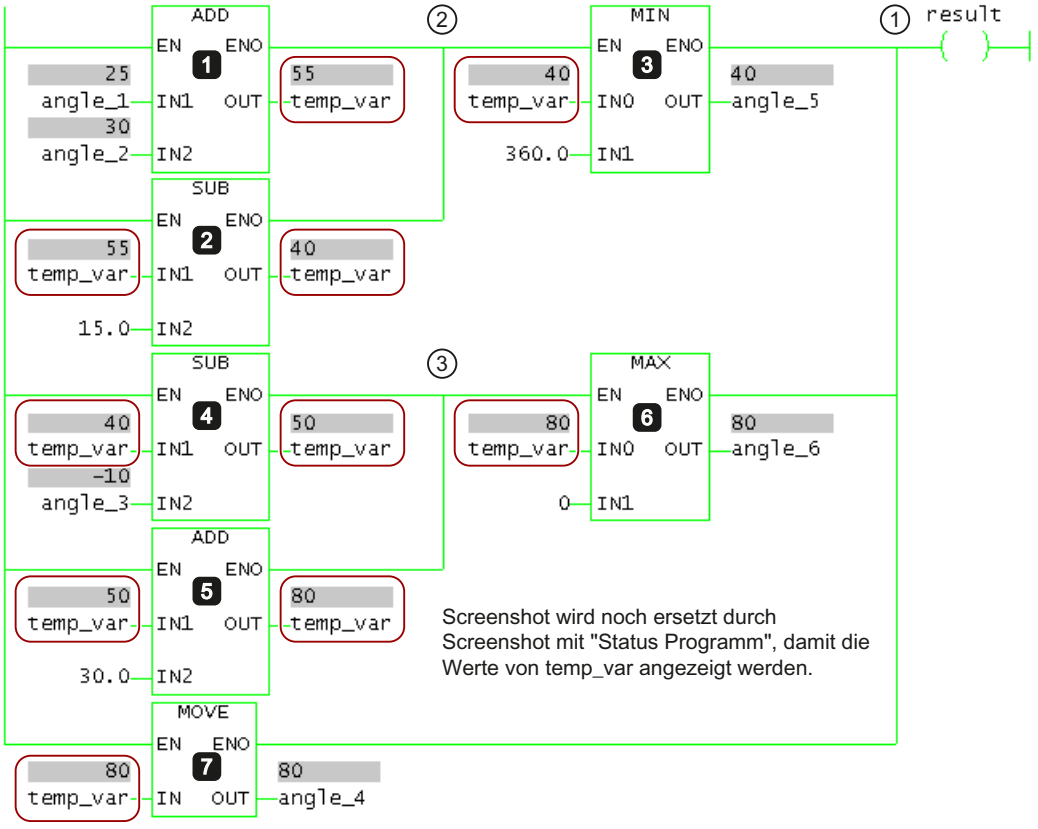


Figure 7-506 Example of an optimized execution order

7.3.5 Functions

This chapter provides a detailed description of the commands with the parameters. The commands described are applicable to both the LAD editor and the FBD editor. Examples are provided to illustrate individual commands in the LAD and FBD editors. Where there are differences such as bit logic, you should refer to the relevant LAD bit logic instructions (Page 5269) editor.

Note

Functions not described in this section can be found in the following sections of the "SIMOTION Basic Functions" Function Manual.

- AUTOHOTSPOT
- AUTOHOTSPOT
- AUTOHOTSPOT

7.3.5.1 LAD bit logic instructions

Bit logic operations work with the numbers **1** and **0**. These numbers form the basis of the binary system and are called binary digits or bits. In connection with AND, OR, XOR and outputs, a **1** stands for logic YES and a **0** for logic NO.

The bit logic operations interpret the signal states **1** and **0** and link them according to boolean logic.

The following bit logic operations are available:

- ---| |--- NO contact
- ---| / |--- NC contact
- XOR Linking EXCLUSIVE OR
- ---( ) Relay coil, output
- ---(#)--- Connector
- ---|NOT|--- Invert signal state

The following operations react to a signal state of **1**:

- ---(S) Set output
- ---(R) Reset output
- SR Prioritize set flip-flop
- RS Prioritize reset flipflop

Some operations react to a rising or falling edge change, so that you can perform one of the following operations:

- --(N)-- Scan edge 1 -> 0
- --(P)-- Scan edge 0 -> 1

- NEG edge detection (falling)
- POS edge detection (rising)

---| |--- NO contact

Symbol

<Operand>

---| |---

Parameter	Data type	Description
<Operand>	BOOL	Scanned bit

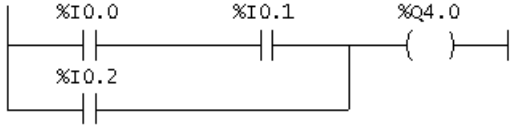
Description

---| |--- (NO contact) is closed if the value of the scanned bit, saved at the specified **<Operand>**, is equal to **1**.

Otherwise, if the signal state at the specified **<address>** is "0", the contact is open.

With series connections, the **---| |---** contact is linked by AND. With parallel connections, the contact is linked by OR.

Example



Output %Q 4.0 is **1**, if (%I 0.0 = **1** AND %I 0.1 = **1**) OR %I 0.2 = **1**.

---| / |--- NC contact

Symbol

<Operand>

—| / |—

Parameter	Data type	Description
<Operand>	BOOL	Scanned bit

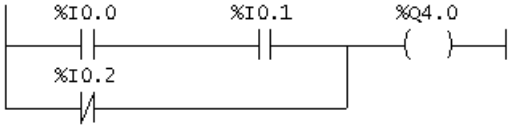
Description

---| / |--- (NC contact) is closed if the value of the scanned bit, saved at the specified **<Operand>**, is equal to **0**.

Otherwise, if the signal state at the specified **<address>** is "1", the contact is open.

With series connections, the **---| / |---** contact is linked bit for bit by AND. With parallel connections, the contact is linked by OR.

Example

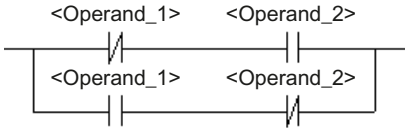


Output %Q 4.0 is **1**, if (%I 0.0 = **1** AND %I 0.1 = **1**) OR %I 0.2 = **0**.

XOR Linking EXCLUSIVE OR

Symbol

For the function **XOR**, a network of NC contacts and NO contacts must be created:

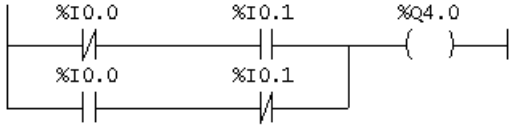


Parameter	Data type	Description
<Operand_1>	BOOL	Scanned bit
<Operand_2>	BOOL	Scanned bit

Description

The value of an **XOR** (Link EXCLUSIVE OR) link is **1** if the signal states of both specified bits are different.

Example



Output %Q 4.0 is **1** if (%I 0.0 = 0 AND %I 0.1 = 1) OR (%I 0.0 = 1 AND %I 0.1 = 0).

---|NOT|--- Invert signal state

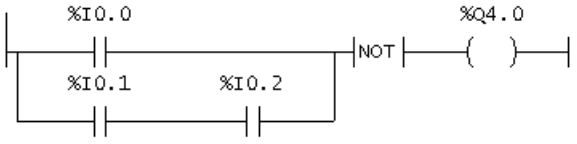
Symbol

---|NOT|---

Description

---|NOT|--- (Invert signal state) inverts the signal bit.

Example



Output %Q 4.0 is **0**, if %I 0.0 = **1** OR (%I 0.1 = **1** AND %I 0.2 = **1)**.

---( ) Relay coil, output

Symbol

<Operand>

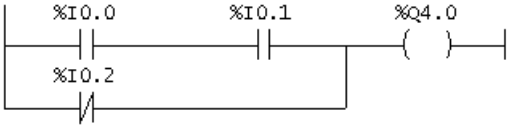
---( )

Parameter	Data type	Description
<Operand>	BOOL	Assigned bit

Description

--- ( ) (Relay coil, output) works like a coil in a circuit diagram. If current flows to the coil, the bit at the **<Operand>**is set to **1**. If no current flows to the coil, the bit at the **<Operand>** is set to **0**. An output coil can only be positioned at the right-hand end of a ladder diagram line in a ladder logic. A negated output can be created with the operation ---|NOT|---.

Example



Output %Q 4.0 is **1**, if (%I 0.0 = **1** AND %I 0.1 = **1**) OR %I 0.2 = **0**.

---(#)--- Connector (LAD)

Symbol

<Operand>

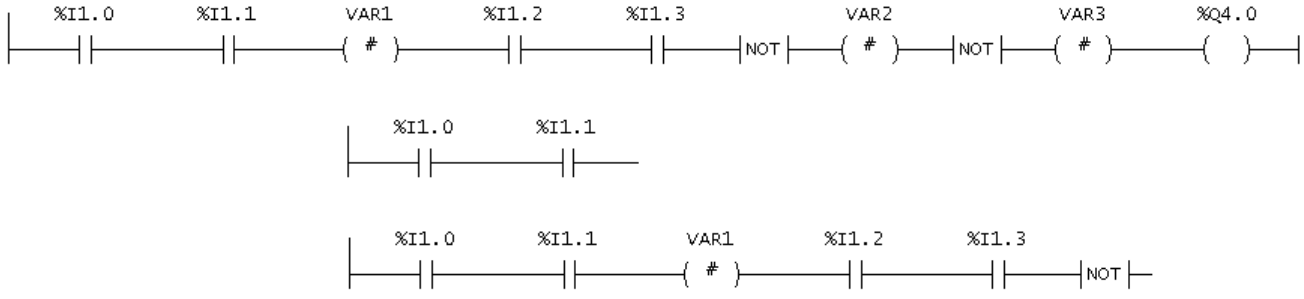
---(#)---

Parameters	Data type	Description
<Operand>	BOOL	Assigned bit

Description

---(#)--- (Connector) is an interposed element with assignment function which saves the current signal state of the signal flow at a specified **<Operand>**. This assignment element saves the bit logic of the last opened branch in front of the assignment element. If connected in series with other elements, the **---(#)---**operation is pasted in as a contact. The **---(#)---** element can never be connected to the conductor bar, nor positioned directly behind a branch, nor used as the end of a branch. A negated element **---(#)---** can be created with the element ---|NOT|--- (Invert signal state).

Example



---(R) Reset output (LAD)

Symbol

<Operand>

---(R)

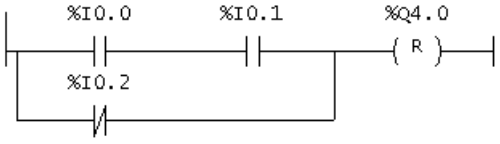
Parameter	Data type	Description
<Operand>	BOOL	Assigned bit

Description

---(R) (Reset output) is executed only if the signal state of the previous operations is **1** (signal flow at the coil). If the signal state is **1**, the specified **<Operand>** of the element is set to **0**.

A signal state of **0** (no signal flow at the coil) has no effect, so that the signal state of the operand of the specified element is not changed.

Example



Output %Q 4.0 is set to **0**, if (%I 0.0 = **1** AND %I 0.1 = **1**) OR %I 0.2 = **0**.

If the signal state is **0**, the signal state of %Q 4.0 remains the same.

---(S) Set output (LAD)

Symbol

<Operand>

---(S)

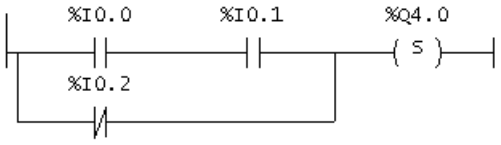
Parameter	Data type	Description
<Operand>	BOOL	Set bit

Description

---(S) (Set output) is executed only if the signal state of the previous operations is **1** (signal flow at the coil). If the signal state is **1**, the specified **<Operand>** of the element is set to **1**.

A signal state = 0 has no effect, so that the current signal state of the specified element's operand is not changed.

Example

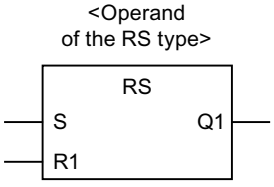


Output %Q 4.0 is set to **1**, if %I 0.0 = **1** AND %I 0.1 = **1**) OR %I 0.2 = **0**.

If the signal state is **0**, the signal state of %Q 4.0 remains the same.

RS Prioritize reset flipflop

Symbol



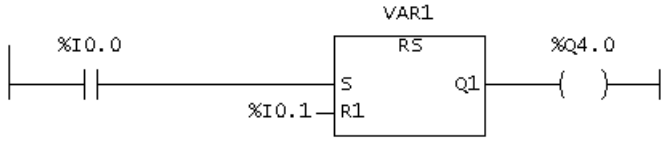
Parameters	Data type	Description
<Operand>	RS	Instance variable of FB type RS
S	BOOL	Set
R1	BOOL	Reset
Q1	BOOL	Signal state of <address>

Description

RS (Prioritize reset flip-flop) is reset if the state at input R1 is **1**. For the state at output Q1, this means:

- Q1 = **0**, if input R1 has state **1** irrespective of the state at the S input.
- Q1 = **1**, if input S has state **1** and input R1 has state **0**.
- Q1 unchanged, if both R1 and S inputs have state **0**.

Example



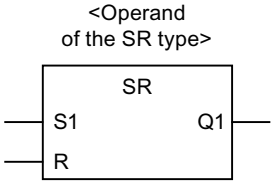
VAR1 is declared as a variable of data type RS and is used as instance of the function block. The I/O addresses %I 0.0 and %I 0.0 (process image of the inputs) are connected to the inputs S and R1 of the function block. Output Q1 of the function block is connected to I/O address %Q 4.0 (process image of the outputs).

The following applies for the signal state at output address %Q 4.0 depending of the signal state at input addresses %I 0.0 and %I0.1:

- %I 0.0 = **1** and %I 0.1 = **1** ⇒ %Q 4.0 = **0**.
- %I 0.0 = **1** and %I 0.1 = **0** ⇒ %Q 4.0 = **1**.
- %I 0.0 = **0** and %I 0.1 = **1** ⇒ %Q 4.0 = **0**.
- %I 0.0 = **0** and %I 0.1 = **0** ⇒ %Q 4.0 remains the same.

SR Prioritize set flipflop

Symbol



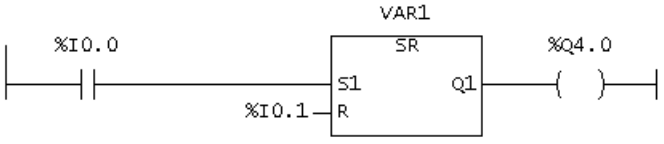
Parameters	Data type	Description
<Operand>	SR	Instance variable from FB type SR
S1	BOOL	Set
R	BOOL	Reset
Q1	BOOL	Signal state of <address>

Description

SR (Prioritize set flip-flop) is set if input S1 has state **1**. For the state at output Q1, this means:

- Q1 = **1**, if input S1 has state **1** irrespective of the state at the R input.
- Q1 = **0**, if input R has state **1** and input S1 has state **0**.
- Q1 unchanged, if both R and S1 inputs have state **0**.

Example



VAR1 is declared as a variable of data type SR and is used as instance of the function block. The I/O addresses %I 0.0 and %I 0.0 (process image of the inputs) are connected to the inputs S1 and R of the function block. Output Q1 of the function block is connected to I/O address %Q 4.0 (process image of the outputs).

The following applies for the signal state at output address %Q 4.0 depending of the signal state at input addresses %I 0.0 and %I0.1:

- %I 0.0 = **1** and %I 0.1 = **1** ⇒ %Q 4.0 = **1**.
- %I 0.0 = **1** and %I 0.1 = **0** ⇒ %Q 4.0 = **1**.
- %I 0.0 = **0** and %I 0.1 = **1** ⇒ %Q 4.0 = **0**.
- %I 0.0 = **0** and %I 0.1 = **0** ⇒ %Q 4.0 remains the same.

--(N)-- Scan edge 1 -> 0 (LAD)

Symbol

<Operand>

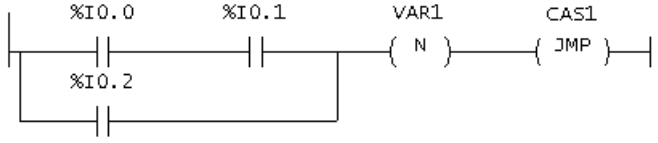
---(N)---

Parameters	Data type	Description
<Operand>	BOOL	N connector bit, saves the previous signal state

Description

---(N)--- (Scan edge 1 -> 0) recognizes a signal state change in the operand from **1** to **0** and displays this after the operation with signal state = 1. The current signal state is compared to the signal state of the operand, the N connector. If the signal state of the operand is **1** and the signal state before the operation is **0**, then the signal after the operation is **1** (pulse), in all other cases **0**. The signal before the operation is saved in the operand.

Example



The N-connector saves the signal state of the result of the entire bit logic.

If the signal state changes from **1** to **0** the jump to the CAS1 jump label is performed.

--(P)-- Scan edge 0 -> 1 (LAD)

Symbol

<Operand>

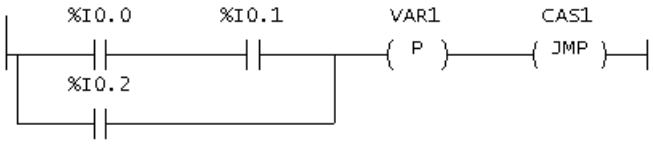
---(P)---

Parameters	Data type	Description
<Operand>	BOOL	P connector bit, saves the previous signal state

Description

---(P)--- (Scan edge 0 -> 1) recognizes a signal state change in the operand from **0** to **1** and displays this after the operation with signal state = 1. The current signal state is compared to the signal state of the operand, the P connector. If the signal state of the operand is **0** and the signal state before the operation is **1**, then the signal after the operation is **1** (pulse), in all other cases **0**. The signal before the operation is saved in the operand.

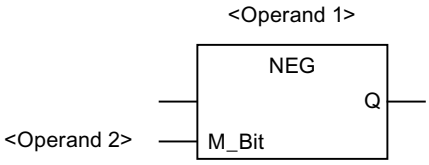
Example



The P-connector saves the signal state of the result of the entire bit logic. If the signal state changes from **0** to **1** the jump to the CAS1 jump label is performed.

NEG edge detection (falling)

Symbol

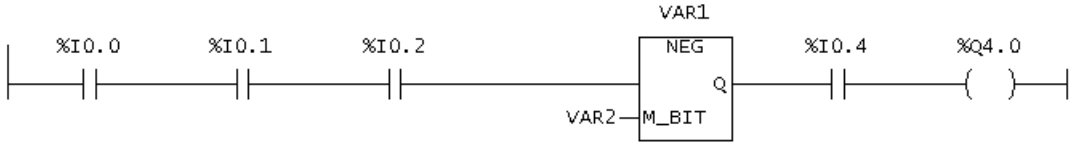


Parameters	Data type	Description
<Operand1>	BOOL	Scanned signal
<Operand2>	BOOL	Connector bit, saves the previous signal state from <Operand1>
Q	BOOL	Signal change detection

Description

NEG (edge detection) compares the signal state of **<Operand1>** with the signal state of the previous scan, which is saved in **<Operand2>**. If the current state of the signal is **0** and the previous state was **1** (detection of a falling edge), output Q is **1** after this function, in all other cases **0**.

Example

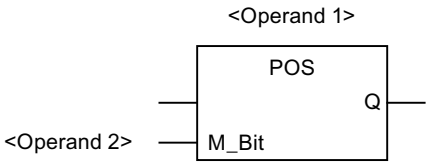


Output %Q 4.0 is **1** if:

(The state at %I 0.0 AND at %I 0.1 AND at %I 0.2 = **1**) AND VAR1 has a falling edge AND the state at %I 0.4 = **1**.

POS edge detection (rising)

Symbol

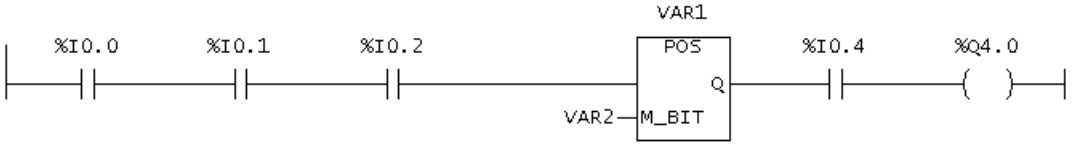


Parameters	Data type	Description
<Operand1>	BOOL	Scanned signal
<Operand2>	BOOL	Connector bit, saves the previous signal state from <Operand1>
Q	BOOL	Signal change detection

Description

POS (edge detection) compares the signal state of **<Operand1>** with the signal state of the previous scan, which is saved in **<Operand2>**. If the current state of the signal is **1** and the previous state was **0** (detection of a rising edge), output Q is **1** after this operation, in all other cases **0**.

Example



Output %Q 4.0 is **1** if:

(The state at %I 0.0 AND at %I 0.1 AND at %I 0.2 = **1**) AND VAR1 has a rising edge AND the state at %I 0.4 = **1**.

Open branch

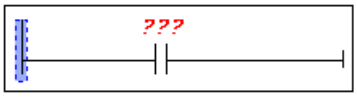
Parallel branches are opened downward.

Parallel branches are always opened behind the selected LAD element.

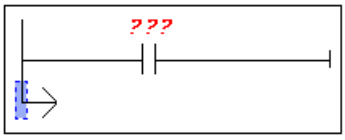
Procedure

To open a parallel branch downward, follow these steps:

1. Use the cursor to select the position where the branch is to be opened.



2. Click the button (shortcut Shift+F8) on the LAD editor toolbar.
   The branch is opened behind the selected element.



Close branch

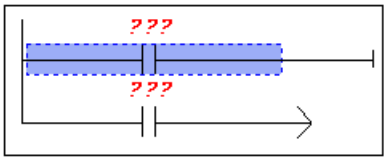
Parallel branches are closed upward.

Parallel branches are always closed behind the selected LAD element.

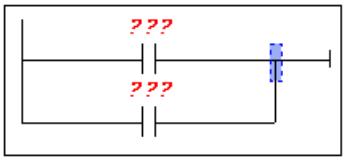
Procedure

To close a parallel branch upward, follow these steps:

1. Use the cursor to select the position where the branch is to be closed.

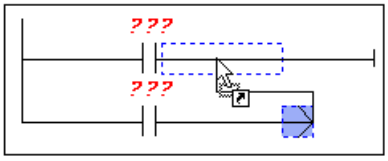


2. Click the button (shortcut Shift+F9) on the LAD editor toolbar.
   The branch is closed behind the selected element.

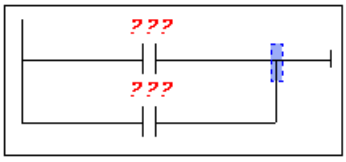


Or:

1. Using the cursor, select the parallel branch to be closed.
2. Holding down the left mouse button, move the selected branch to the position at which it is to be closed.



3. Release the left mouse button.
   The branch is closed at the selected position.



7.3.5.2 FBD bit logic instructions

FBD bit logic instructions

Bit logic operations work with the numbers **1** and **0**. These numbers form the basis of the binary system and are called binary digits or bits. In connection with AND, OR, XOR and outputs, a **1** stands for logic YES and a **0** for logic NO.

The bit logic operations interpret the signal states **1** and **0** and link them according to boolean logic.

The following bit logic operations are available in the FBD editor:

- & AND box
- >=1 OR box
- XOR Exclusive OR box
- [=] Assignment
- [#] Connector

The following operations react to a signal state of 1:

- [R] Reset assignment
- [S] Set assignment
- RS Prioritize reset flipflop
- SR Prioritize set flip-flop

Some operations react to a rising or falling edge change, so that you can perform one of the following operations:

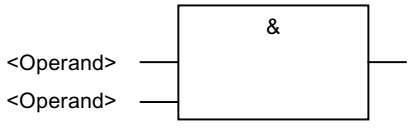
- [N] Scan edge 1 -> 0
- [P] Scan edge 0 -> 1
- NEG edge detection (falling)
- POS edge detection (rising)

The other operations directly affect the signal states:

- --| Inserting a binary input
- --o| Negating a binary input

& AND box

Symbol

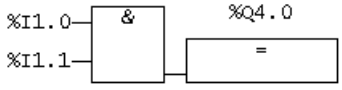


Parameters	Data type	Description
<Operand>	BOOL	Scanned bit

Description

With the **AND** operation, you can scan the signal states of two or more specified operands at the inputs of an AND box. If the signal status of all operands is **1**, the condition is fulfilled and the result of the operation is **1**. If the signal status of one operand is **0**, the condition is not fulfilled and the operation returns a result of **0**.

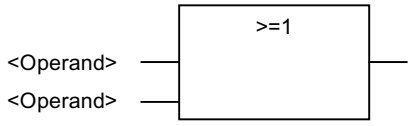
Example



Output %Q 4.0 is set when the signal state at input %I 1.0 AND %I 1.1 is **1**.

>=1 OR box

Symbol

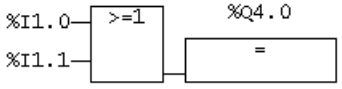


Parameters	Data type	Description
<Operand>	BOOL	Scanned bit

Description

With the **OR** operation, you can scan the signal states of two or more specified operands at the inputs of an OR box. If the signal status of one of the operands is **1**, the condition is fulfilled and the result of the operation is **1**. If the signal status of all operands is **0**, the condition is not fulfilled and the operation returns a result of **0**.

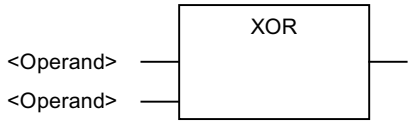
Example



Output %Q 4.0 is set when the signal state at input %I 1.0 OR %I 1.1 is **1**.

XOR EXCLUSIVE OR box

Symbol

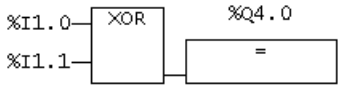


Parameters	Data type	Description
<Operand>	BOOL	Scanned bit

Description

In an **EXCLUSIVE OR** box, the signal state is **1** if the signal state of one of the two specified operands is **1**.

Example



At output %Q 4.0, the signal state is **1** if the signal state is **1** either EXCLUSIVELY at input %I 0.0 OR at input %I 0.1.

--| Inserting a binary input

Symbol

<Operand>

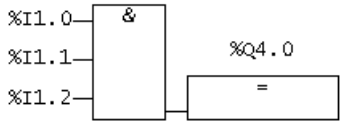
---|

Parameters	Data type	Description
<Operand>	BOOL	Scanned bit

Description

The **Insert binary input** operation inserts a binary input in an AND, OR, or XOR box behind the selection mark.

Example



Output % Q 4.0 is **1** when the state %I 1.0 AND %I 1.1 AND %I 1.2 = 1.

--o| Negating a binary input

Symbol

<Operand>

–o|

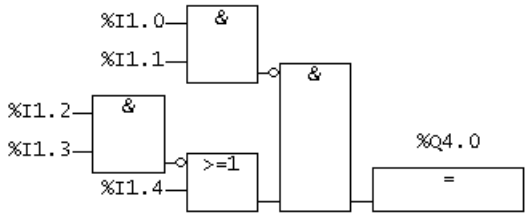
Parameters	Data type	Description
<Operand>	BOOL	Scanned bit

Description

The **Negate binary input** operation negates the signal state.

All binary inputs of any elements can be negated.

Example

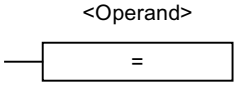


Output %Q 4.0 is **1** if:

- The signal state at %I 1.0 AND %I 1.1 is NOT 1
- AND the signal state at %I 1.2 AND %I 1.3 is NOT 1
- OR the signal state at %I 1.4 = 1.

[=] Assignment

Symbol



Parameters	Data type	Description
<Operand>	BOOL	Assigned bit

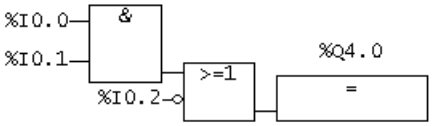
Description

The **Assignment** operation supplies the value. The box at the end of the logic operation carries a signal of 1 or 0 according to the following criteria:

- The output carries a signal of 1 when the conditions of the logic operation are fulfilled before the output box.
- The output carries a signal of 0 when the conditions of the logic operation are not fulfilled before the output box.

The FBD logic operation assigns the signal state to the output that is addressed by the operation. If the conditions of the FBD logic operation are fulfilled, the signal state at the output box is **1**. Otherwise, the signal state is **0**.

Example

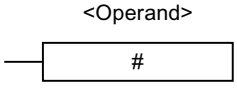


Output %Q 4.0 is **1** if:

- At inputs %I 0.0 AND %I 0.1, the signal state is 1,
- OR %I 0.2 = 0.

[#] Connector (FBD)

Symbol

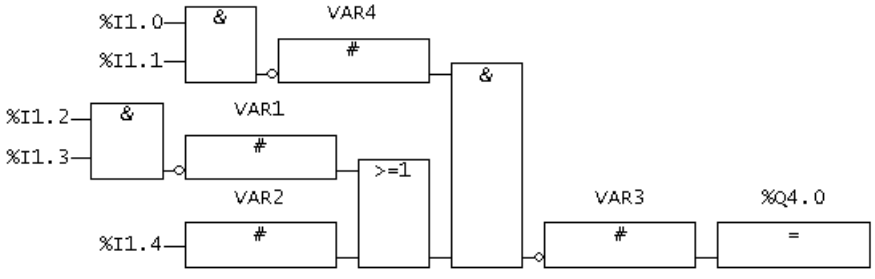


Parameters	Data type	Description
<Operand1>	BOOL	Assigned bit

Description

The **Connector** operation is an intermediate assignment element that stores the signal state. Specifically, this assignment element saves the bit logic of the last opened branch in front of the assignment element.

Example



Connectors store the following logic operation results:

VAR4 saves the negated signal state of

%I1.0

%I1.1

VAR1 saves the negated signal state of

%I1.2

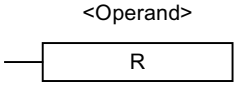
%I1.3

VAR2 saves the negated signal state of %I 1.4.

VAR3 saves the negated signal state of the entire bit logic operation.

[R] Reset assignment (FBD)

Symbol

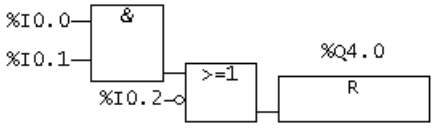


Parameters	Data type	Description
<Operand>	BOOL	Assigned bit

Description

The **Reset assignment** operation then is only performed when the signal state = 1. If the signal state = 1, the specified operand is reset to **0** by the operation. If the signal state = 0, the operation does not affect the specified operand. The operand remains unchanged.

Example



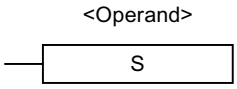
The signal state at output %Q 4.0 is only reset to **0** if:

- The signal state is 1 at inputs %I 0.0 AND %I 0.1
- OR the signal state at input %I 0.2 = 0

If the signal state of the branch = 0, the signal state at output %Q 4.0 is not changed.

[S] Set assignment (FBD)

Symbol

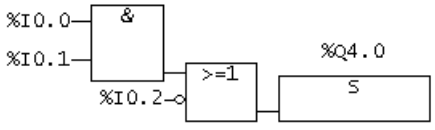


Parameters	Data type	Description
<Operand>	BOOL	Set bit

Description

The **Set assignment** operation is only performed when the signal state = 1. If the signal state = 1, the specified operand is reset to **1** by the operation. If the signal state = 0, the operation does not affect the specified operand. The operand remains unchanged.

Example



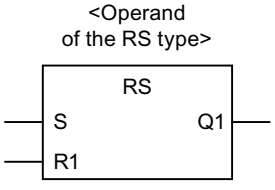
The signal state at output %Q 4.0 is only reset to **1** if:

- The signal state is 1 at inputs %I 0.0 AND %I 0.1
- OR the signal state at input %I 0.2 = 0

If the signal state of the branch = 0, the signal state of %Q 4.0 is not changed.

RS Prioritize reset flipflop

Symbol



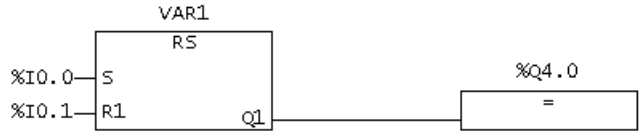
Parameters	Data type	Description
<Operand>	RS	Instance variable of FB type RS
S	BOOL	Enable set
R1	BOOL	Enable reset
Q1	BOOL	Signal state of <address>

Description

The **Prioritize reset flipflop** operation only performs operations such as Set assignment (S) or Reset assignment (R1) if the signal state = 1. A signal state of 0 has no effect on these operations; the operand specified in the operation is not changed.

Prioritize reset flipflop is reset when the signal state at input R1 = 1 and the signal state at input S = 0. The flipflop is set when input R1 = 0 and input S = 1. If the signal state at both inputs is **1**, the flipflop is reset.

Example

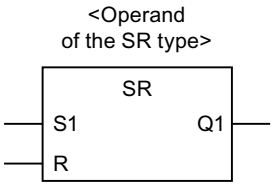


If the state at input %I 0.0 = **1** and at input %I 0.1 = **0**, the variable VAR1 is set and output %Q 4.0 is **1**. Otherwise, if the signal state at input %I 0.0 = **0** and the signal state at input %I 0.1 = **1**, the variable VAR1 is reset and %Q 4.0 is **0**.

If both signal states are **0**, nothing is changed. If both signal states are **1**, the reset operation has priority. VAR1 is reset and %Q 4.0 is **0**.

SR Prioritize set flipflop

Symbol



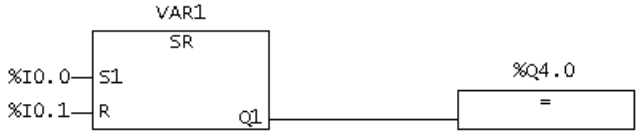
Parameters	Data type	Description
<Operand>	SR	Instance variable from FB type SR
S1	BOOL	Enable set
R	BOOL	Enable reset
Q1	BOOL	Signal state of <address>

Description

The **Prioritize set flipflop** operation only performs operations such as Set (S1) or Reset (R) if the signal state = 1. A signal state of **0** has no effect on these operations; the operand specified in the operation is not changed.

Prioritize set flipflop is set when the signal state at input S1 = 1 and the signal state at input R = 0. The flipflop is reset when input S1 = 0 and input R = 1. If the signal state at both inputs is **1**, the flipflop is set.

Example

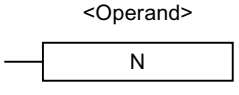


If the state at input %I 0.0 = **1** and at input %I 0.1 = **0**, the variable VAR1 is set and output %Q 4.0 is **1**. Otherwise, if the signal state at input %I 0.0 = 0 and the signal state at input %I 0.1 = 1, the variable VAR1 is reset and %Q 4.0 is **0**.

If both signal states are **0**, nothing is changed. If both signal states are **1**, the set operation has priority. VAR1 is set and %Q 4.0 is **1**.

[N] Scan edge 1 -> 0 (FBD)

Symbol

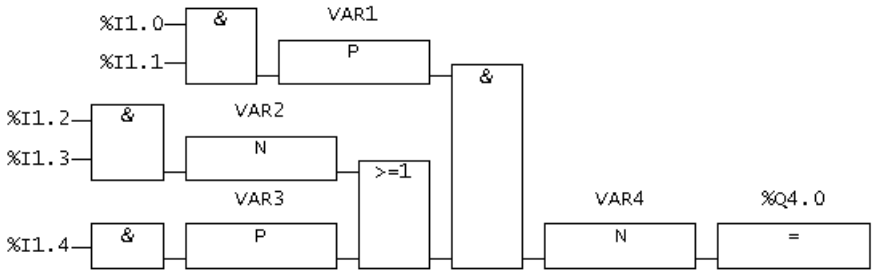


Parameters	Data type	Description
<Operand>	BOOL	N connector bit, saves the previous signal state

Description

The **Scan edge 1 -> 0** recognizes a signal state change in the specified operands from **1** to **0** (falling edge) and displays this after the operation with signal state = 1. The current signal state is compared to the signal state of the operand, the edge variable. If the signal state of the operand is **1** and the signal state before the operation is **0**, then the signal after the operation is **1** (pulse), in all other cases **0**. The signal state before the operation is saved in the operand.

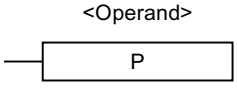
Example



The VAR4 variable saves the signal state.

[P] Scan edge 0 -> 1 (FBD)

Symbol

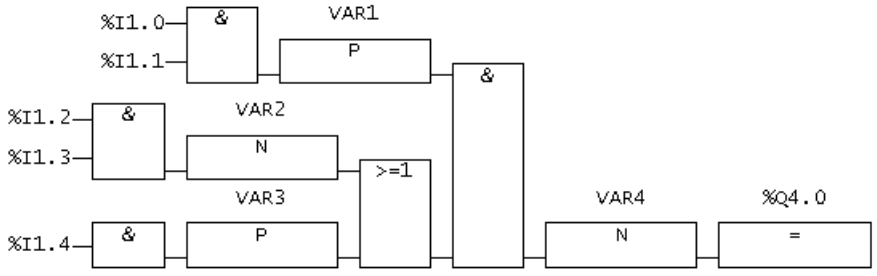


Parameters	Data type	Description
<Operand>	BOOL	P connector bit, saves the previous signal state

Description

The **Scan edge 0 -> 1** recognizes a signal state change in the specified operands from **0** to**1** (rising edge) and displays this after the operation with signal state = 1. The current signal state is compared to the signal state of the operand, the edge variable. If the signal state of the operand is **0** and the signal state before the operation is **1**, then the signal state after the operation is 1, in all other cases **0**. The signal state before the operation is saved in the operand.

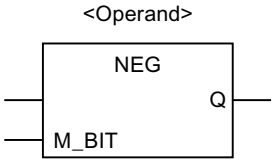
Example



The VAR1 variable saves the signal state.

NEG edge detection (falling)

Symbol

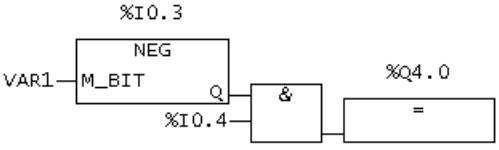


Parameters	Data type	Description
<Operand1>	BOOL	Scanned signal
M_BIT	BOOL	The M-BIT operand indicates the variable in which the previous signal state of NEG is saved.
Q	BOOL	Signal change detection

Description

The **Edge detection** (falling) operation compares the signal state of <Operand1> with the signal state of the previous scan, which is saved in M_BIT. If a change has occurred from **1** to **0**, then output Q = **1**, in all other cases **0**.

Example

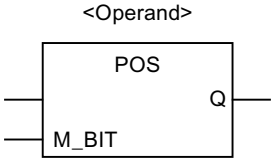


Output %Q 4.0 is **1** if:

- Input %I 0.3 has a falling edge
- AND the signal state at input %I 0.4 = 1.

POS edge detection (rising)

Symbol

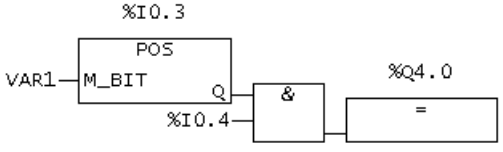


Parameters	Data type	Description
<Operand1>	BOOL	Scanned signal
M_BIT	BOOL	The M-BIT operand indicates the variable in which the previous signal state of POS is saved.
Q	BOOL	Signal change detection

Description

The **Edge detection** (rising) operation compares the signal state of <Operand1> with the signal state of the previous scan, which is saved in M_BIT. If a change has occurred from **0** to **1**, then output Q = **1**, in all other cases **0**.

Example



Output %Q 4.0 is **1** if:

- Input %I 0.3 has a rising edge
- AND the signal state at input %I 0.4 = 1.

7.3.5.3 Relational operators

Overview of comparison operations

Description

The inputs IN1 and IN2 are compared using the following comparison methods:

= IN1 is equal to IN2
<> IN1 is not equal to IN2
> IN1 is greater than IN2
< IN1 is less than IN2
>= IN1 is greater than or equal to IN2
<= IN1 is less than or equal to IN2

The following comparison operation is available:

- CMP comparator

CMP Compare numbers

Icons

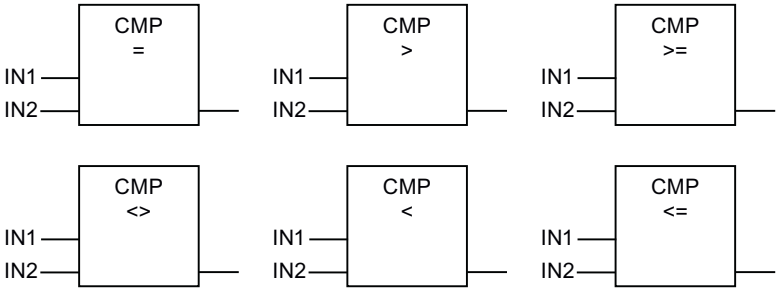


Table 7-559 Parameters for CMP <, CMP > CMP >=, CMP <=

Parameter	Data type	Description
Box output	BOOL	Result of the comparison, processing is only continued if the signal state at the box input = 1.
IN1	ANY_NUM[1] ANY_BIT DATE TIME_OF_DAY (TOD) DATE_AND_TIME (DT) TIME STRING[2]	First comparison value

Parameter	Data type	Description
IN2	ANY_NUM[1] ANY_BIT DATE TIME_OF_DAY (TOD) DATE_AND_TIME (DT) TIME STRING[2]	Second comparison value
The first and second comparison values must be of the same data type, e.g. ANY_NUM and ANY_NUM, DATE and DATE, STRING and STRING. [1] It must be possible to convert both comparison values into the most powerful data type by means of implicit conversion. [2] Variables of the STRING data type can be compared irrespective of the declared length of the string.		

Table 7-560 Parameter for CMP =, CMP <>

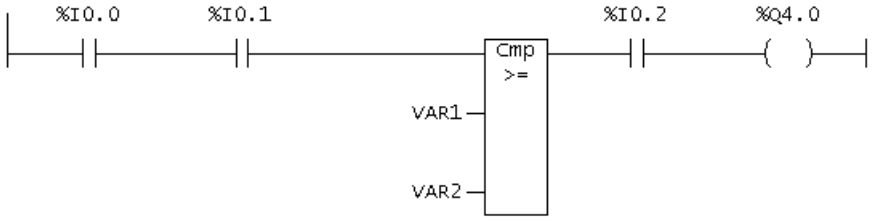
Parameter	Data type	Description
Box output	BOOL	Result of the comparison, processing is only continued if the signal state at the box input = 1.
IN1	ANY_NUM[1] ANY_BIT DATE TIME_OF_DAY (TOD) DATE_AND_TIME (DT) TIME STRING[2] Enumeration[3] Array[3] Structure[3] STRUCTTASKID STRUCTALARMID ANYOBJECT	First comparison value
IN2	ANY_NUM[1] ANY_BIT DATE TIME_OF_DAY (TOD) DATE_AND_TIME (DT) TIME STRING[2] Enumeration[3] Array[3] Structure[3] STRUCTTASKID STRUCTALARMID ANYOBJECT	Second comparison value
The first and second comparison values must be of the same data type, e.g. ANY_NUM and ANY_NUM, DATE and DATE, STRING and STRING. [1] It must be possible to convert both comparison values into the most powerful data type by means of implicit conversion. [2] Variables of the STRING data type can be compared irrespective of the declared length of the string. [3] The data type specifications (see the SIMOTION ST Programming and Operating Manual) must be identical for both comparison values.		

Description

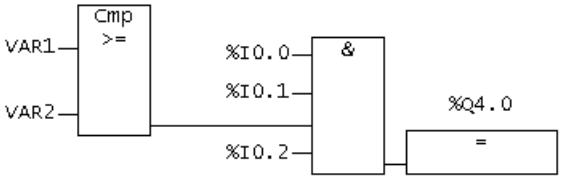
CMP (Compare numbers) can be used like a normal contact. The box can be used at the positions where a normal contact can also be positioned. IN1 and IN2 are compared with the comparison method selected by you.

If the comparison is true, then the value of the operation is **1**. The value of the whole ladder diagram line is linked by AND if the comparison element is connected in series or by OR if the box is connected in parallel.

Example



Representation in the LAD editor



Representation in the FBD editor

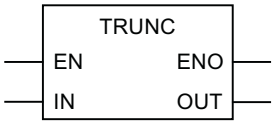
%Q 4.0 is set when:

- VAR1 >= VAR2
- AND the signal state at input %I 0.0 is (1).
  AND the signal state at input %I 0.1 is (1).
  AND the signal state at input %I 0.2 is (1).

7.3.5.4 Conversion instructions

TRUNC Generate integer

Symbol

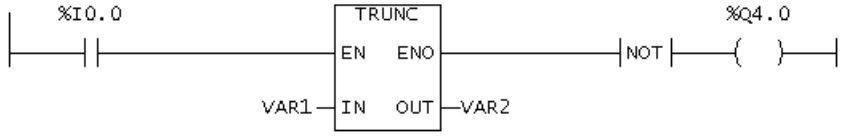


Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY_REAL	Number which is to be converted
OUT	ANY_INT	Integral part of the value from IN

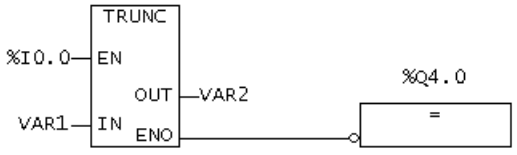
Description

TRUNC (generate integer) reads the contents of the IN parameter as a floating-point number and converts this value to an integer (32-bit). The result is the integral part of the floating-point number which is output by the OUT parameter.

Example



Representation in the LAD editor



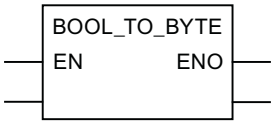
Representation in the FBD editor

If %I 0.0 = 1, the contents of VAR1 are read as a floating-point number and converted to an integer (32-bit). The result is the integral part of the floating-point number which is saved in VAR2.
Output %Q 4.0 is **1** if an overflow occurs or the statement is not processed (%I 0.0 = 0).

Generating numeric data types and bit data types

Symbol

e.g. BOOL_TO_TYPE



Description

You can carry out the explicit data type conversion with the standard functions listed in the tables below.

- Input parameters
  Each function for the conversion of a data type has exactly one input parameter.
- Function value
  The function value is always the return value of the function. The table shows the rules with which a data type can be converted.
- Naming
  Because the data types of the input parameter and the function value come from the respective function name, they are not listed specially in the table "Functions for conversion of numerical data types and bit data types": E.g. with function BOOL_TO_BYTE, the data type of the input parameter is BOOL, the data type of the function value BYTE.

Table 7-561 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
BOOL_TO_BYTE	Accept as least significant bit and fill the rest with 0.	yes
BYTE_TO_BOOL	Accept the least significant bit.	no
BYTE_TO_SINT	Accept bit string as SINT value.	no
BYTE_TO_USINT	Accept bit string as USINT value.	no
BYTE_TO_WORD	Accept least significant bit and fill the rest with 0.	yes

Table 7-562 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
DINT_TO_DWORD	Accept value as bit string.	no
DINT_TO_INT	Cut off two most significant bytes.	no
DINT_TO_LREAL	Accept value.	yes
DINT_TO_REAL	Accept value (accuracy may be lost).	no
DINT_TO_UDINT	Accept value as bit string.	no
DINT_TO_UINT	Cut off two most significant bytes.	no
DINT_TO_WORD	Cut off two most significant bytes.	no

Table 7-563 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
DWORD_TO_DINT	Accept bit string as DINT value.	no
DWORD_TO_INT	Accept the two least significant bytes as INT value.	no
DWORD_TO_REAL	Accept bit string as REAL value (validity check of the REAL number is **not** carried out!).	no
DWORD_TO_UDINT	Accept bit string as UDINT value.	no

Function name	Conversion rule	Implicit okay
DWORD_TO_UINT	Accept the two least significant bytes as UINT value.	no
DWORD_TO_WORD	Accept the two least significant bytes of the bit string.	no

Table 7-564 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
INT_TO_DINT	Accept value.	yes
INT_TO_LREAL	Accept value.	no
INT_TO_REAL	Accept value.	yes
INT_TO_SINT	Cut off the most significant byte.	no
INT_TO_UDINT	Accept value as bit string; the two most significant bytes are filled with the most significant bit of the input parameter.	no
INT_TO_UINT	Accept value as bit string.	no
INT_TO_WORD	Accept value as bit string.	no

Table 7-565 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
LREAL_TO_DINT	Round off to integer part.	no
LREAL_TO_INT	Round off to integer part.	no
LREAL_TO_REAL	Accept value (accuracy may be lost).	no
LREAL_TO_UDINT	Round off to integer part.	no
LREAL_TO_UINT	Round off to integer part.	no

Table 7-566 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
REAL_TO_DINT	Round off to integer part.	no
REAL_TO_DWORD	Accept bit string.	no
REAL_TO_INT	Round off to integer part.	no
REAL_TO_LREAL	Accept value.	yes
REAL_TO_UDINT	Round off to integer part.	no
REAL_TO_UINT	Round off to integer part.	no

Table 7-567 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
SINT_TO_BYTE	Accept bit string.	no
SINT_TO_INT	Accept value.	yes
SINT_TO_USINT	Accept bit string.	no

Table 7-568 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
UDINT_TO_DINT	Accept bit string.	no
UDINT_TO_INT	Cut off numerical sequence (2 most significant bytes).	no
UDINT_TO_DWORD	Accept bit string.	no
UDINT_TO_LREAL	Accept value.	yes
UDINT_TO_REAL	Accept value (accuracy may be lost).	no
UDINT_TO_UINT	Cut off numerical sequence (2 most significant bytes).	no
UDINT_TO_WORD	Cut off numerical sequence (2 most significant bytes).	no

Table 7-569 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
UINT_TO_DINT	Accept value.	yes
UINT_TO_DWORD	Accept bit string, fill rest with zeros.	no
UINT_TO_INT	Accept bit string.	no
UINT_TO_LREAL	Accept value (accuracy may be lost).	no
UINT_TO_REAL	Accept value.	yes
UINT_TO_UDINT	Accept value.	yes
UINT_TO_USINT	Cut off numerical sequence (most significant byte).	no
UINT_TO_WORD	Accept bit string.	no

Table 7-570 Functions for conversion of numerical data types and bit data types

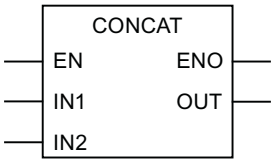
Function name	Conversion rule	Implicit okay
USINT_TO_BYTE	Accept bit string.	no
USINT_TO_INT	Accept value.	yes
USINT_TO_DINT	Accept value.	no
USINT_TO_SINT	Accept bit string.	no
USINT_TO_UINT	Accept value.	yes

Table 7-571 Functions for conversion of numerical data types and bit data types

Function name	Conversion rule	Implicit okay
WORD_TO_BYTE	Cut off most significant byte.	no
WORD_TO_DINT	Accept bit string, fill rest with zeros.	no
WORD_TO_DWORD	Accept the two least significant bytes and fill the rest with 0.	yes
WORD_TO_INT	Accept bit string and interpret this as an integer.	no
WORD_TO_UDINT	Accept bit string, fill rest with zeros.	no
WORD_TO_UINT	Accept bit string.	no

Generating date and time

Symbol



Description

The table below shows the standard functions for date and time data types:

Table 7-572 Standard functions for date and time

Function name	Data type of input parameter	Data type of function value	Description
CONCAT	1: DATE 2: TIME_OF_DAY	DATE_AND_TIME	Compress DATE and TIME_OF_DAY to DATE_AND_TIME.
DT_TO_TOD	DATE_AND_TIME	TIME_OF_DAY	Accept time of day.
DT_TO_DATE	DATE_AND_TIME	DATE	Accept date.

Note

Data type TIME can be converted to numerical data types as follows:

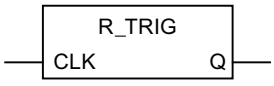
- To data type UDINT:
  Divide it by a standardization factor of data type TIME.
  Multiply the reconversion by the same standardization factor.

7.3.5.5 Edge detection

System function block R_TRIG can be used to detect a rising edge; F_TRIG can detect a falling edge. You can use this function, for example, to set up a sequence of your own function blocks.

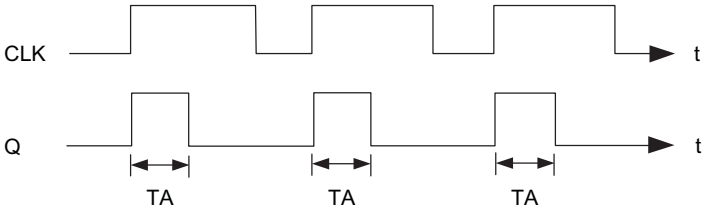
Detection of rising edge R_TRIG

Symbol



Description

If a rising edge (R_TRIG, Rising Trigger), i.e. a status change from 0 to 1, is present at the input, 1 is applied at the output for the duration of one cycle.



TA Cycle time

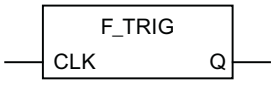
Figure 7-507 Mode of operation of R_TRIG (rising edge) function block

Table 7-573 Call parameters for R_TRIG

Identifier	Parameter	Data type	Description
CLK	Input	BOOL	Input for edge detection
Q	Output	BOOL	Status of edge

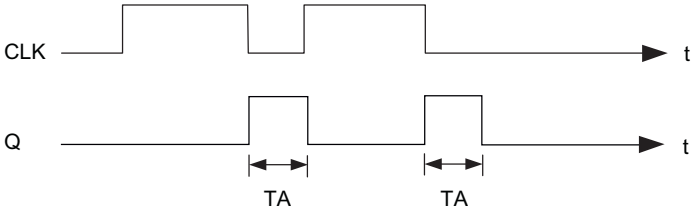
Detection of falling edge F_TRIG

Symbol



Description

When a falling edge (F_TRIG, falling trigger), i.e. a status change from 1 to 0, occurs at the input, the output is set to 1 for the duration of one cycle time.



TA Cycle time

Figure 7-508 Mode of operation of F_TRIG (falling edge) function block

Table 7-574 Call parameters for F_TRIG

Identifier	Parameter	Data type	Description
CLK	Input	BOOL	Input for edge detection
Q	Output	BOOL	Status of edge

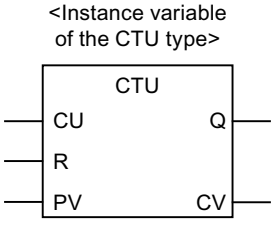
7.3.5.6 Counter operations

Overview of counter operations

Every call-up of the function block and the function should be recorded in a counter.

CTU up counter

Symbol



Description

The CTU counter allows you to perform upward counting operations:

- If the input is R = TRUE when the FB is called up, then the CV output is reset to 0.
- If the CU input changes from FALSE to TRUE (0 to 1) when the FB is called (positive edge), then the CV output is incremented by 1.
- Output Q specifies whether CV is greater than or equal to comparison value PV.

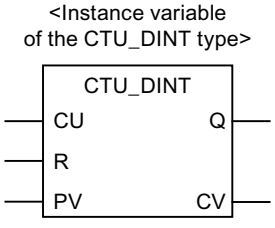
The CV and PV parameters are both INT data types, which means that the maximum counter reading possible is 32767 (= 16#7FFF).

Table 7-575 Parameters for CTU

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0
PV	Input	INT	Comparison value
Q	Output	BOOL	Status of counter (CV >= PV)
CV	Output	INT	Counter value

CTU_DINT up counter

Symbol



Description

The method of operation is the same as for the CTU incrementer except for the following:

The CV and PV parameters are both DINT data types, which means that the maximum counter reading possible is 2147483647 (= 16#7FFF_FFFF).

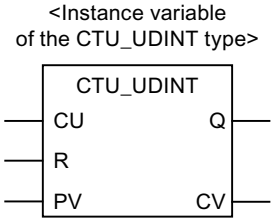
Table 7-576 Parameters for CTU_DINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0

Identifier	Parameter	Data type	Description
PV	Input	DINT	Comparison value
Q	Output	BOOL	Status of counter (CV >= PV)
CV	Output	DINT	Counter value

CTU_UDINT up counter

Symbol



Description

The method of operation is the same as for the CTU incrementer except for the following:

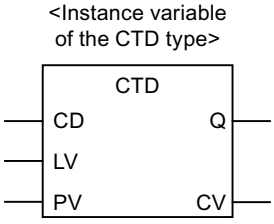
The CV and PV parameters are both UDINT data types, which means that the maximum counter reading possible is 4294967295 (=16# FFFF_FFFF).

Table 7-577 Parameters for CTU_UDINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0
PV	Input	UDINT	Comparison value
Q	Output	BOOL	Status of counter (CV >= PV)
CV	Output	UDINT	Counter value

CTD down counter

Symbol



Description

The CTD counter allows you to perform downward counting operations.

- If the LD input = TRUE when the FB is called, then the CV output is reset to start value PV.
- If the CD input changes from FALSE to TRUE (0 to 1) when the FB is called (positive edge), then the CV output is decremented by 1.
- Output Q specifies whether CV is less than or equal to 0.

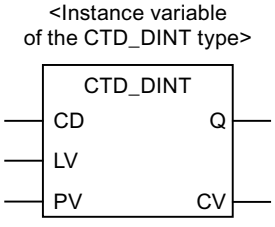
The CV and PV parameters are both INT data types, which means that the minimum counter reading possible is -32,768 (= 16#8000).

Table 7-578 Parameters for CTD

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	INT	Start value of counter
Q	Output	BOOL	Status of counter (CV <= 0)
CV	Output	INT	Counter value

CTD_DINT down counter

Symbol



Description

The method of operation is the same as for the CTD up counter except for the following:

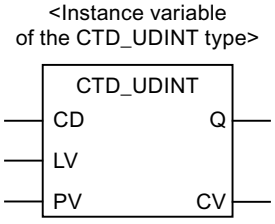
The CV and PV parameters are both DINT data types, which means that the minimum counter reading possible is -2147483648 (= 16#8000_0000).

Table 7-579 Parameters for CTD_DINT

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	DINT	Start value of counter
Q	Output	BOOL	Status of counter (CV <= 0)
CV	Output	DINT	Counter value

CTD_UDINT down counter

Symbol



Description

The method of operation is the same as for the CTD up counter except for the following:

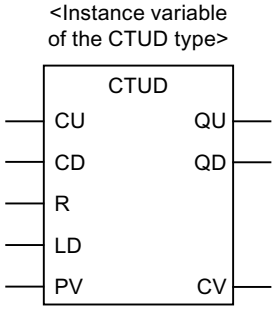
The CV and PV parameters are both UDINT data types, which means that the minimum counter value possible is 0.

Table 7-580 Parameters for CTD_DINT

Identifier	Parameter	Data type	Description
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
LD	Input	BOOL	Reset the counter to start value
PV	Input	UDINT	Start value of counter
Q	Output	BOOL	Status of counter (CV <= 0)
CV	Output	UDINT	Counter value

CTUD up/down counter

Symbol



Description

The CTUD counter allows you to perform both upward and downward counting operations.

- Reset the CV count variable:
  - If the input is R = TRUE when the FB is called up, then the CV output is reset to 0.
  - If the LD input = TRUE when the FB is called, then the CV output is reset to start value PV.
- Count:
  - If the CU input changes from FALSE to TRUE (0 to 1) when the FB is called (positive edge), then the CV output is incremented by 1.
  - If the CD input changes from FALSE to TRUE (0 to 1) when the FB is called up (positive edge), then the CV output is decremented by 1.
- Counter status QU or QD:
  - Output Q specifies whether CV is greater than or equal to comparison value PV.
  - Output QD specifies whether CV is less than or equal to 0.

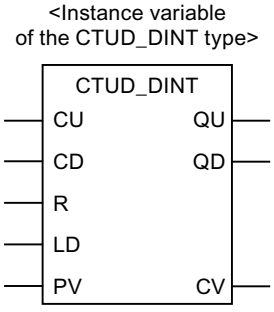
Table 7-581 Parameters for CTUD

Identifier	Parameters	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0 (up counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	INT	Comparison value (for up counter) Start value (for down counter)
QU	Output	BOOL	Status as up counter (CV >= PV)

Identifier	Parameters	Data type	Description
QD	Output	BOOL	Status as down counter (CV <= 0)
CV	Output	INT	Counter value

CTUD_DINT up/down counter

Symbol



Description

The method of operation is the same as for the CTUD up counter except for the following:

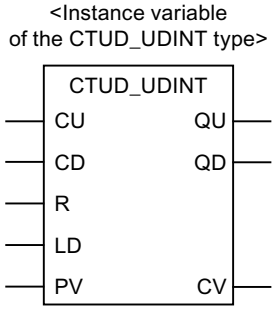
The CV and PV parameters are both DINT data types.

Table 7-582 Parameters for CTD_DINT

Identifier	Parameters	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0 (up counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	DINT	Comparison value (for incrementer) Start value (for decrementer)
QU	Output	BOOL	Status as up counter (CV >= PV)
QD	Output	BOOL	Status as down counter (CV <= 0)
CV	Output	DINT	Counter value

CTUD_UDINT up/down counter

Symbol



Description

The method of operation is the same as for the CTUD up counter except for the following:

The CV and PV parameters are both UDINT data types.

Table 7-583 Parameters for CTD_DINT

Identifier	Parameter	Data type	Description
CU	Input	BOOL	Count up if value changes from FALSE to TRUE (positive edge)
CD	Input	BOOL	Count down if value changes from FALSE to TRUE (positive edge)
R	Input	BOOL	Reset the counter to 0 (up counter)
LD	Input	BOOL	Reset the counter to PV start value (down counter)
PV	Input	UDINT	Comparison value (for incrementer) Start value (for decrementer)
QU	Output	BOOL	Status as up counter (CV >= PV)
QD	Output	BOOL	Status as down counter (CV <= 0)
CV	Output	UDINT	Counter value

7.3.5.7 Jump instructions

Overview of jump operations

Description

Jump operations can be used in all logic blocks, e.g. programs, function blocks (FBs), and functions (FCs).

Jump label as operand

The operand of a jump operation is a jump label. The jump label specifies the point to where the program is to jump.

Enter the jump label via the JMP coil. The jump label consists of up to 480 characters. The first character must be a letter, the other characters can be either letters or numbers (e.g. SEG3).

Jump label as target

The target jump label can be at the start of a network.

See also

Showing/hiding a jump label (Page 5131)

---(JMP) Jump in block if 1 (conditional)

Symbol

<jump label>

---(JMP)

Description

---(JMP) (Jump in block if 1) functions as a conditional jump if the pending signal of the previous logic operation is **1**.

There must also be a target (LABEL) for every ---(JMP).

The operations between the jump operation and the jump label are not executed!

Note

An unconditional jump is created by hanging the --(JMP) element directly on the power rail.

---(JMPN) Jump in block if 0 (conditional)

Symbol

<jump label>

---(JMPN)

Description

---(JMPN) (Jump in block if 0) functions as a conditional jump if the pending signal of the previous logic operation is **0**.

There must also be a target (LABEL) for every ---(JMPN).

The operations between the jump operation and the jump label are not executed!

LABEL Jump label

Symbol

Description

LABEL marks the target of a jump operation. It can have a maximum of 80 characters. The first character must be a letter, the other characters can be letters or numbers, e.g. CAS1.

There must be a target (LABEL) for every ---(JMP) or ---(JMPN).

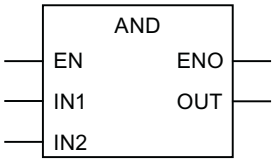
Note

Only alphanumeric characters are allowed during input. The jump label is deleted if it contains an error and cannot be corrected.

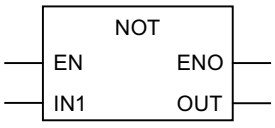
7.3.5.8 Non-binary logic

Symbol

e.g. AND



e.g. NOT



Description

Logic operations (AND, OR, XOR, NOT) are also provided as boxes with EN/ENO for non-binary values in the LAD/FBD editor.

The logical operations are listed in the table below.

There is only one operand for **NOT**.

Note

In the Command library tab of the project navigator, the elements **AND**, **XOR**, and **OR, NOT** are represented in the **Logic** entry.

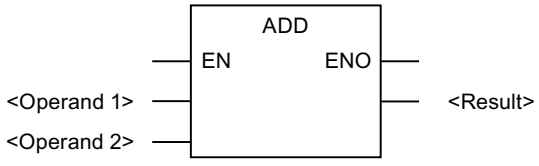
Table 7-584 Non-binary logic

Operator Parameters	AND	XOR	OR	NOT
IN1	ANY_BIT	ANY_BIT	ANY_BIT	ANY_BIT
IN2	ANY_BIT	ANY_BIT	ANY_BIT	
EN	BOOL	BOOL	BOOL	BOOL
ENO	BOOL	BOOL	BOOL	BOOL
OUT	ANY_BIT	ANY_BIT	ANY_BIT	ANY_BIT

7.3.5.9 Arithmetic operators

Symbol

e.g. addition



Description

An arithmetic expression is composed of arithmetic operators. These expressions allow numerical data types to be processed.

The divide operators DIV and MOD require that the second operand is not equal to zero.

Note

In the Command library tab of the project navigator, the elements **ADD**, **SUB**, **MUL**, and **DIV** are represented as **+**, **-**, *****, and **/**.

The execution of a network, for example, is simply aborted in the event of an overflow, and the relevant/assigned event-triggered task (ExecutionFaultTask) is started.

The table below contains a list of the arithmetic operators:

Table 7-585 Arithmetic operators

Instruction	Operator	1. Operand (IN1)	2. Operand (IN2)	Result (OUT)
Addition	ADD	ANY_NUM	ANY_NUM	ANY_NUM[1]
		BYTE	BYTE	BYTE
		WORD	WORD	WORD
		DWORD	DWORD	DWORD
		TIME	TIME	TIME[2]
		TOD	TIME	TOD[2]
		DT	TIME	DT[3]
Multiplication	MUL	ANY_NUM	ANY_NUM	ANY_NUM[1]
		BYTE	BYTE	BYTE
		WORD	WORD	WORD
		DWORD	DWORD	DWORD
		TIME	ANY_INT	TIME
Subtraction	SUB	ANY_NUM	ANY_NUM	ANY_NUM[1]
		BYTE	BYTE	BYTE
		WORD	WORD	WORD
		DWORD	DWORD	DWORD
		TIME	TIME	TIME
		TOD	TIME[4]	TOD
		TOD	TOD	TIME[5]
		DT	TIME	DT
		DT	DT	TIME[5]
Division	DIV	ANY_NUM	ANY_NUM[6]	ANY_NUM[1]
		BYTE	BYTE[6]	BYTE
		WORD	WORD[6]	WORD
		DWORD	DWORD[6]	DWORD
		TIME	ANY_INT[6]	TIME
		TIME	TIME[6]	UDINT
Modulo division.	MOD	ANY_INT	ANY_INT[6]	ANY_INT[1]
		BYTE	BYTE[6]	BYTE
		WORD	WORD[6]	WORD
		DWORD	DWORD[6]	DWORD

[1] The data types of the operands and of the result must be identical.

[2] Addition, possibly with overflow.

[3] Addition with date correction.

[4] Restriction of TIME to TOD before calculation.

[5] These operations are based on the modulo of the maximum value of the TIME data type.

[6] The second operand must not be equal to zero.

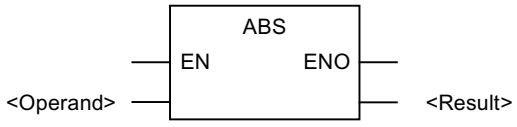
7.3.5.10 Numeric standard functions

Every numeric standard function has an input parameter. The result is always the function value.

General numeric standard functions

Symbol

e.g. absolute value



Description

General numeric standard functions are used for:

- Calculation of the absolute value of a variable
- Calculation of the square root of a variable

The table below shows the general numeric standard functions:

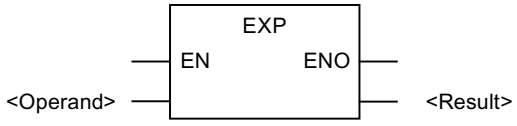
Table 7-586 General numeric standard functions

Function name	Input parameter data type (IN)	Function value data type (OUT)	Description
ABS	ANY_NUM	ANY_NUM[1]	Absolute value
SQRT	ANY_REAL	ANY_REAL[1]	Square root
[1] Identical to the data type of the input parameter IN			

Logarithmic standard functions

Symbol

e.g. exponential value



Description

Logarithmic standard functions are functions for calculating an exponential value or a logarithm of a value.

The table below shows the logarithmic standard functions:

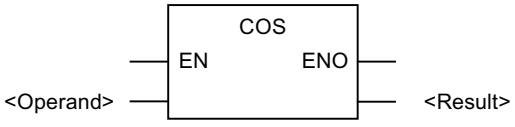
Table 7-587 Logarithmic standard functions

Function name	Input parameter data type (IN)	Data type of function value (OUT)	Description
EXP	ANY_REAL	ANY_REAL[1]	e^x (natural exponential function)
EXPD	ANY_REAL	ANY_REAL[1]	10^x (decimal exponential function)
EXPT	ANY_REAL (IN1) ANY_NUM (IN2)	ANY_REAL[2]	Exponentiation
LN	ANY_REAL	ANY_REAL[1]	Natural logarithm
LOG	ANY_REAL	ANY_REAL[1]	Common logarithm
[1] Identical to the data type of the input parameter IN [2] Identical to the data type of the input parameter IN1			

Trigonometric standard functions

Symbol

e.g. COS



Description

The trigonometric standard functions listed in the table expect and calculate variables of angles in radian measure.

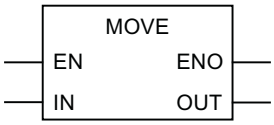
Table 7-588 Trigonometric standard functions

Function name	Input parameter data type	Data type of function value	Description
ACOS	ANY_REAL	ANY_REAL	Arc cosine (main value)
ASIN	ANY_REAL	ANY_REAL	Arc sine (main value)
ATAN	ANY_REAL	ANY_REAL	Arc tangent (main value)
COS	ANY_REAL	ANY_REAL	Cosine (radian measure input)
SIN	ANY_REAL	ANY_REAL	Sine (radian measure input)
TAN	ANY_REAL	ANY_REAL	Tangent (radian measure input)

7.3.5.11 Move

MOVE Transfer value

Symbol



Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY	Source value
OUT	ANY	Destination address

Description

MOVE (Assign a value) is activated by the enable input EN. The value specified by the IN input is copied to the value specified in the OUT output. ENO has the same signal state as EN.

7.3.5.12 Shifting operations

Overview of shifting operations

Description

The contents of input IN can be moved bit-by-bit to the left or right using shifting operations. A shift of n bits to the left multiplies the contents of input IN by 2 to the power of n; a shift of n bits to the right divides the contents of input IN by 2 to the power of n. If, for example, you move the binary equivalent of the decimal value 3 by 3 bits to the left, this gives the binary equivalent of the decimal value 24. Shift the binary equivalent of the decimal value 16 by 2 bits to the right, this gives the binary equivalent of the decimal value 4.

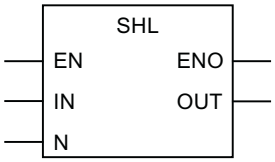
The number that you specify at input N indicates the number of bits by which to shift. The places which become free as a result of the shifting operation are filled up with zeroes.

The following shifting operations are available:

- shift bit to the left
- shift bit to the right

SHL Shift bit to the left

Symbol



Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY_BIT	Value to be shifted
N	USINT	Number of bit positions to be shifted
OUT	ANY_BIT	Result of shifting operation

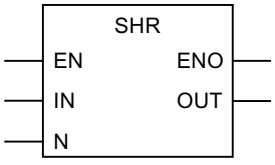
Description

SHL (e.g. shift left by 32 bits) is activated if the enable input (EN) has the signal state **1**. The operation SHL shifts the bits 0 to 31 of the input IN bit-by-bit to the left. Input N specifies the number of bit positions to be shifted. If N is greater than 32, the command writes a **0** in the OUT output. The same number (N) of zeros is shifted from the right in order to occupy the positions which have become free. The result of the shifting operation can be queried at output OUT.

ENO has the same signal state as EN.

SHR Shift bit to the right

Symbol



Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY_BIT	Value to be shifted
N	USINT	Number of bit positions to be shifted
OUT	ANY_BIT	Result of shifting operation

Description

SHR (e.g. shift right by 32 bits) is activated if the enable input (EN) has the signal state **1**. The operation SHR shifts the bits 0 to 31 of the input IN bit-by-bit to the right. Input N specifies the number of bit positions to be shifted. If N is greater than 32, the command writes a **0** in the OUT output. The same number (N) of zeros is shifted from the left in order to occupy the positions which have become free. The result of the shifting operation can be queried at output OUT.

ENO has the same signal state as EN.

7.3.5.13 Rotating operations

Overview of rotating operations

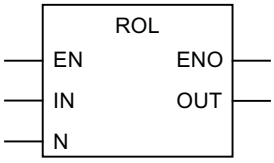
Description

The entire contents of input IN can be rotated bit-by-bit to the left or right using rotating operations. The positions which become free are filled up with the signal states of the bits which have been moved out of the IN input.

At the input N you can specify the number of bits for the rotation.

ROL Rotate bit to the left

Symbol



Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY_BIT	Value to be rotated
N	USINT	Number of bit positions to be rotated
OUT	ANY_BIT	Result of rotation operation

Description

ROL (e.g. rotate left by 32 bits) is activated if the enable input (EN) has the signal state **1**. The operation ROL rotates the entire contents of the IN input bit-by-bit to the left. Input N specifies the number of bit positions by which to rotate. If N is greater than 32, the double word IN is rotated by ((N-1) modulo 32)+1 positions. The bit positions coming from the right are occupied with the signal state of the bits which have been rotated to the left (left rotation). The result of the rotation operation can be queried at output OUT.

ENO has the same signal state as EN.

Example

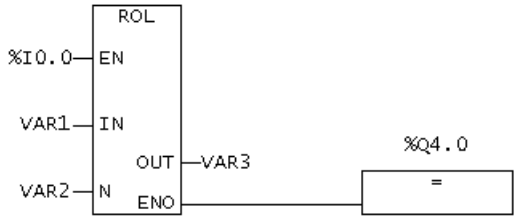


Figure 7-509 Representation in the FBD editor

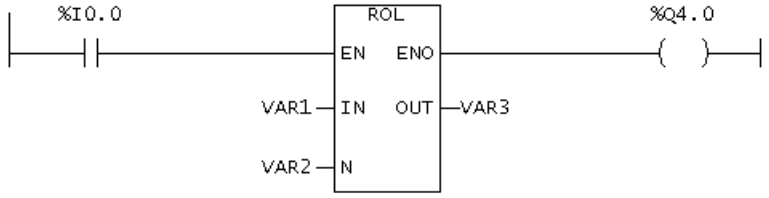
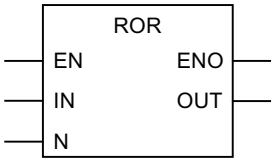


Figure 7-510 Representation in the LAD editor

The ROL box is executed if %I 0.0 = 1. VAR1 is loaded and rotated to the left by the number of bits specified in VAR2. The result is written to VAR3. %Q 4.0 is set.

ROR Rotate bit to the right

Symbol



Parameter	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN	ANY_BIT	Value to rotate
N	USINT	Number of bit positions to rotate
OUT	ANY_BIT	Result of rotation operation

Description

ROR (e.g. rotate right by 32 bits) is activated if the enable input (EN) has the signal state **1**. The operation ROR rotates the entire contents of the IN input bit-by-bit to the right. Input N specifies the number of bit positions by which to rotate. If N is greater than 32, the double word IN is rotated by ((N-1) modulo 32)+1 positions. The bit positions coming from the left are occupied by the signal state of the bits which have been rotated to the right (right rotation). The result of the rotation operation can be queried at output OUT.

ENO has the same signal state as EN.

Example

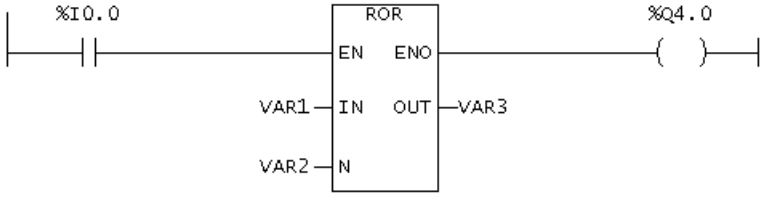


Figure 7-511 Representation in the LAD editor

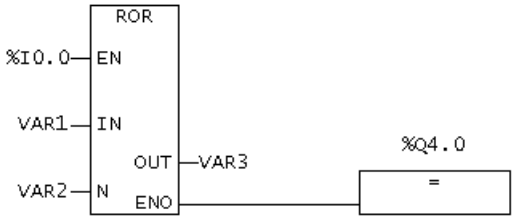


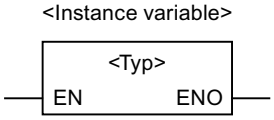
Figure 7-512 Representation in the FBD editor

The ROR box is executed if %I 0.0 = 1. VAR1 is loaded and rotated to the right by the number of bits specified in VAR2. The result is written to VAR3. %Q 4.0 is set.

7.3.5.14 Program control instructions

Calling up an empty box

Symbol



Parameters	Data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output

Parameters	Data type	Description
<Type>	FB / FC	FC/FB type
<Instance variable>	FB	FB instance variable

Description

The symbol for an empty box depends on the function block/function (according to how many parameters there are). EN, ENO and the name of the FB/FC must be available.

You do not have to specify EN/ENO in the variable declaration. The input and output are automatically allocated by the system.

The EN input can be used to inhibit a block call and redirect the block of the EN input to the ENO output.

It is not possible to control the ENO output in the block itself.

Note

You can use an empty box to insert a call (Page 5137). As soon as you enter the type, the box transforms and displays the parameters of the specified FB/FC call.

See also

Inserting LAD/FBD elements (Page 5137)

RET Jump back

Symbol

---(RET)

Description

RET (jump back) is used for the conditional exit from blocks. A preceding logic operation is necessary for this output.

Example

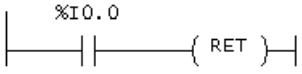


Figure 7-513 Representation in the LAD editor

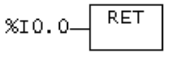


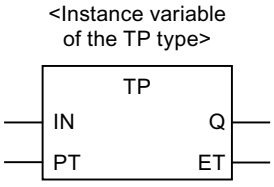
Figure 7-514 Representation in the FBD editor

The operation is executed if %I 0.0 = 1.

7.3.5.15 Timer instructions

TP pulse

Symbol



Description

With a signal state change from 0 to 1 at the IN input, time ET is started. Output Q remains at 1 until elapsed time ET is equal to programmed time value PT. As long as time ET is running, the IN input has no effect.

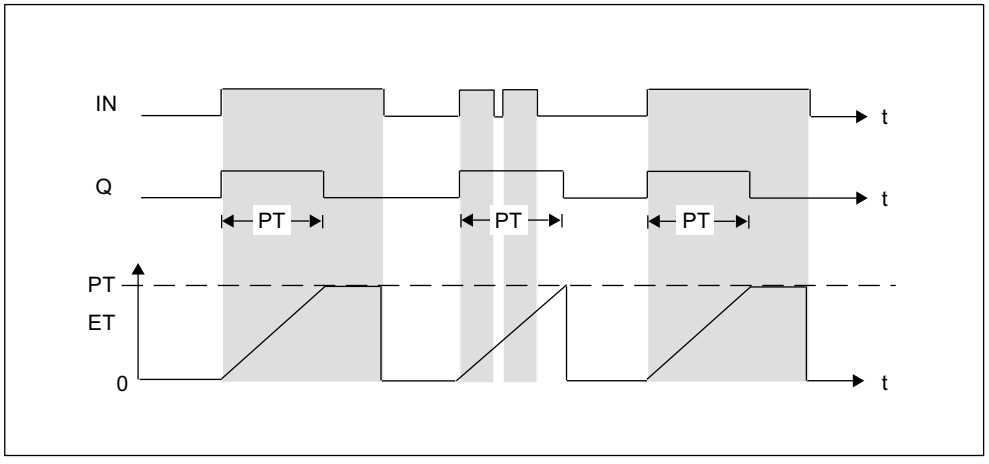


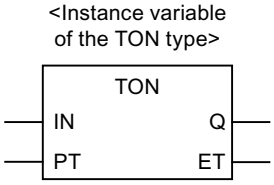
Figure 7-515 Mode of operation of TP pulse timer

Table 7-589 Call parameters for TP

Identifier	Parameters	Data type	Description
IN	Input	Input	Start input
PT	Input	TIME	Duration of pulse
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

TON ON delay

Symbol



Description

With the signal state change from 0 to 1 at the IN input, time ET is started. The output signal Q only changes from 0 to 1 if the time ET = PT has elapsed and the input signal IN still has the value 1, i.e. the output Q is switched on with a delay. Input signals of shorter durations than programmed time PT do not appear at the output.

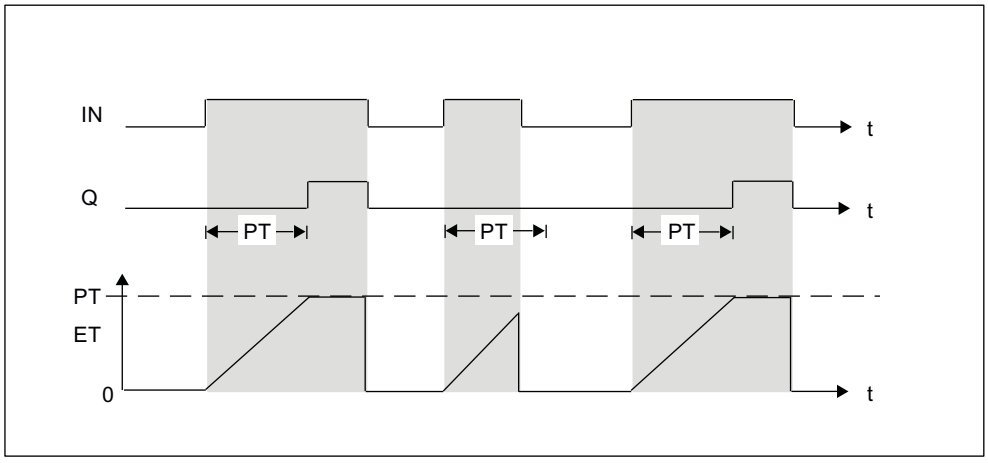


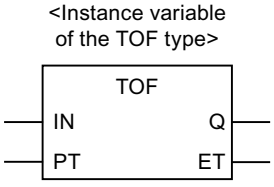
Figure 7-516 Mode of operation of TON on delay timer

Table 7-590 Call parameters for TON

Identifier	Parameters	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration for which the rising edge at input IN is delayed.
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

TOF OFF delay

Symbol



Description

With a signal state change from 0 to 1 at start input IN, state 1 appears at output Q. If the state at the start input IN changes from 1 to 0, then time ET is started. If a change occurs at input IN from 0 to 1 before time ET has elapsed, then the timer operation is reset. A start is initiated again when the state at input IN changes from 1 to 0. Only after the duration ET = PT has elapsed does output Q adopt a signal state of 0. This means that the output is switched off with a delay.

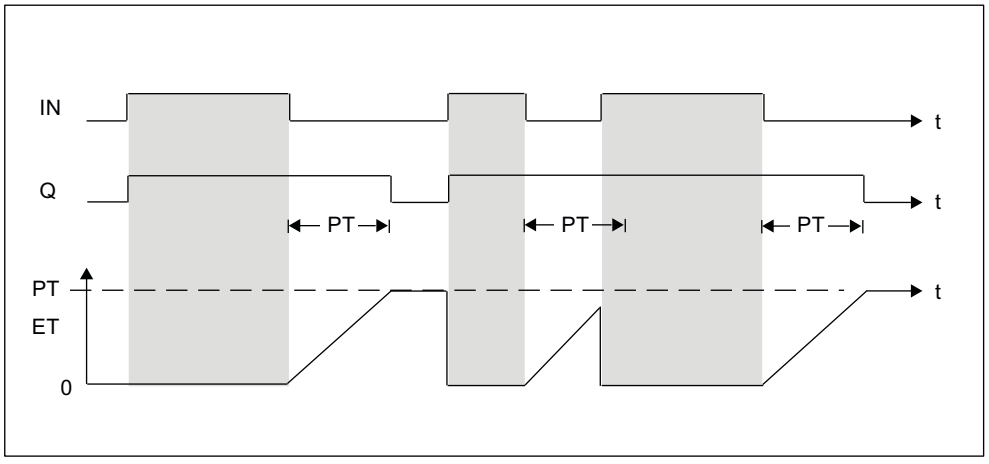


Figure 7-517 Mode of operation of TOF off delay timer

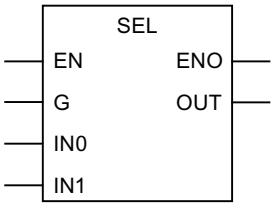
Table 7-591 Call parameters for TOF

Identifier	Parameters	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration for which the falling edge at input IN is delayed.
Q	Output	BOOL	Status of time
ET	Output	TIME	Elapsed time

7.3.5.16 Selection functions

SEL Binary selection

Symbol



Parameters	Input parameter data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
G	BOOL	Input parameter
IN0	ANY	Input parameter
IN1	ANY	Input parameter

Description

The function value is one of the input parameters IN0 or IN1, depending on the value of the input parameter G.

The input parameters IN0 and IN1 must be the same data type or must be capable of implicit conversion into the same data type.

The return value is data type ANY.

Selected input parameter

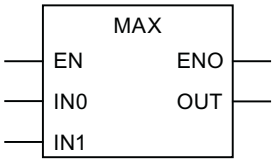
IN0 if G = 0 (FALSE)

IN1 if G = 1 (TRUE)

The data type corresponds to the common data type of the input parameters IN0 and IN1.

MAX Maximum function

Symbol



Parameter	Input parameter data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN0	ANY_ELEMENTARY	Input parameter
IN1	ANY_ELEMENTARY	Input parameter

Description

The function value is the maximum value of both input parameters IN0 and IN1.

The input parameters IN0 and IN1 must be the same data type or must be capable of implicit conversion into the most powerful data type.

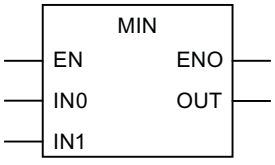
The return value is of data type ANY_ELEMENTARY.

Maximum of the input parameters.

The data type corresponds to the most powerful data type of the input parameters IN0 and IN1.

MIN Minimum function

Symbol



Parameter	Input parameter data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
IN0	ANY_ELEMENTARY	Input parameter
IN1	ANY_ELEMENTARY	Input parameter

Description

The function value is the minimum value of both input parameters IN0 and IN1.

All IN0 and IN1 input parameters must be the same data type or capable of implicit conversion into the most powerful data type.

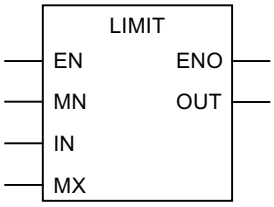
The return value is of data type ANY_ELEMENTARY.

Minimum of the input parameters.

The data type corresponds to the most powerful data type of the input parameters IN0 and IN1.

LIMIT Limiting function

Symbol



Parameters	Input parameter data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
MN	ANY_ELEMENTARY	Input parameter Lower limiting value
IN	ANY_ELEMENTARY	Input parameter Value to be limited
MX	ANY_ELEMENTARY	Input parameter Upper limiting value

Description

The input parameter IN is limited to values lying between the lower limit value MN and the upper limit value MX.

All input parameters must be the same data type or capable of conversion into the most powerful data type by implicit conversion.

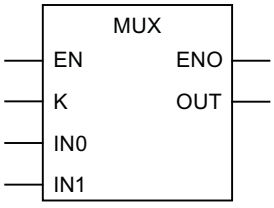
The return value is of data type ANY_ELEMENTARY.

MIN (MAX (IN, MN), MX)

The data type corresponds to the most powerful data type of the input parameters.

MUX Multiplex function

Symbol



Parameters	Input parameter data type	Description
EN	BOOL	Enable input
ENO	BOOL	Enable output
C	ANY_INT	Input parameter
IN0	ANY	Input parameter
IN1	ANY	Input parameter

Description

The function value is one of the two input parameters IN0 or IN1, depending on the value of the input parameter K.

The input parameters IN0 and IN1 must be the same data type or must be capable of implicit conversion into the same data type.

The return value is data type ANY.

The data type corresponds to the common data type of the input parameters IN0 and IN1.

7.3.6 Commissioning (software)

7.3.6.1 Commissioning

This chapter describes how to assign created programs to the task system of a control unit and how to download them to the target system.

7.3.6.2 Assigning programs to a task

Programs must be assigned to a task before they can be downloaded to the target system (the SIMOTION device).

Various tasks are made available by SIMOTION, each with different priorities or system responses (e.g. during initialization).

Further information can be found in the SIMOTION SCOUT Basic Functions Function Manual, and in SIMOTION online help.

Assigning programs to a task:

1. In the project navigator, double-click under the corresponding SIMOTION device the **EXECUTION SYSTEM** element.
   The configuration window for the execution system opens.
2. Select the required task (e.g. MotionTask_1) from the left pane.
3. Select the **Program assignment** tab.
4. Select the program to be assigned from the **Programs** list.

5. Click the button **>>**.
6. Select the **Task configuration** tab to specify additional settings for the task if required.

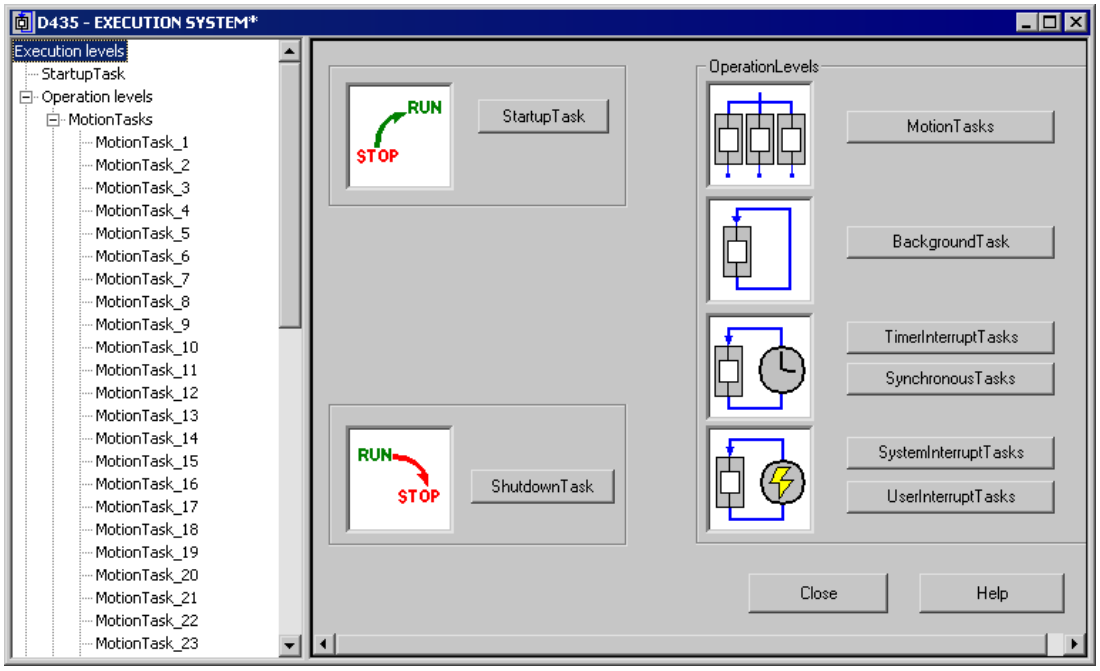


Figure 7-518 Configure execution system

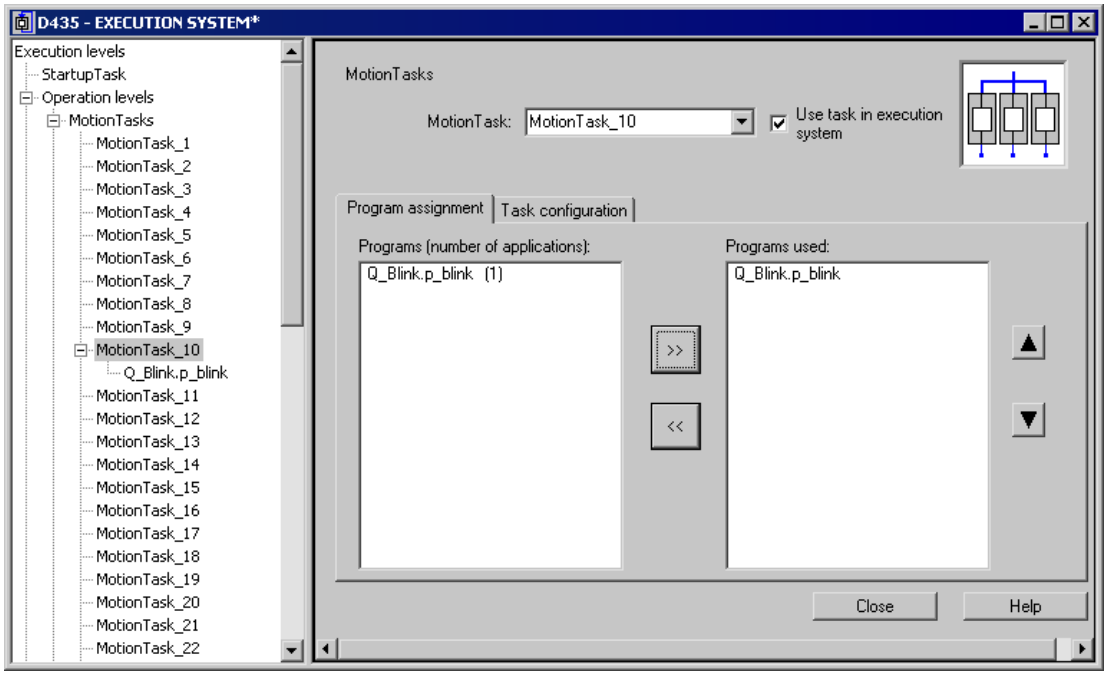


Figure 7-519 Assigning a program to a motion task

7.3.6.3 Execution levels and tasks in SIMOTION

Here the execution levels and the tasks assigned are shown in tabular format for an initial overview.

Further information on execution levels and tasks can be found in the SIMOTION Basic Functions Function Manual.

Table 7-592 Description of the execution levels and of the tasks

Execution level	Description
Time-controlled	Cyclic tasks: They are restarted automatically once the assigned programs have been executed.
• SynchronousTasks	Tasks are started periodically, synchronous with specified system cycle clock. • ServoTask_Fast: Synchronous with Servo_fast cycle clock. The Servo_fast cycle clock is a second servo cycle clock and only available: – For D445-2 DP/PN and D455-2 DP/PN as of version V4.2 – For D435-2 DP/PN as of version V4.3. • ServoSynchronousTask: Synchronous with the position control cycle clock • IpoTask_Fast: Synchronous with IPO_fast cycle clock. The IPO_fast cycle clock is the IPO cycle clock for the second servo cycle clock and only available: – For D445-2 DP/PN and D455-2 DP/PN as of version V4.2 – For D435-2 DP/PN as of version V4.3. • IPOsynchronousTask: Synchronous with interpolator cycle clock IPO • IPOsynchronousTask_2: Synchronous with interpolator cycle clock IPO_2 • PWMsynchronousTask: Synchronous with PWM cycle clock (for TControl technology package) • InputSynchronousTask_1: Synchronous with Input1 cycle clock (for TControl technology package) • InputSynchronousTask_2: Synchronous with Input2 cycle clock (for TControl technology package) • PostControlTask_1: Synchronous with Control1 cycle clock (for TControl technology package) • PostControlTask_2: Synchronous with Control2 cycle clock (for TControl technology package)
• TimerInterruptTasks	Tasks are started periodically in a fixed time frame. This time frame must be a multiple of interpolator cycle clock IPO.
Interrupts	Sequential tasks: They are executed once after the start and then terminated.
• SystemInterruptTasks	Started when a system event occurs: • ExecutionFaultTask: Error processing a program • PeripheralFaultTask: Error on I/O • TechnologicalFaultTask: Error on the technology object • TimeFaultBackgroundTask: BackgroundTask timeout • TimeFaultTask: TimerInterruptTask timeout
• UserInterruptTasks	They are started when a user-defined event occurs.

Execution level	Description
Round robin	MotionTasks and BackgroundTasks share the free time remaining after execution of the higher-priority system and user tasks. The proportion of the two levels can be assigned.
• MotionTasks	Sequential tasks: They are executed once after the start and then terminated. Start takes place: • Explicitly via a task control command in a program assigned to another task. • Automatically when RUN operating state is attained if the corresponding attribute was set during task configuration. The priority of a MotionTask can be increased temporarily: • In the MCC programming language with the "Wait for..." commands, see Wait for axis, Wait for signal, Wait for condition. • In the ST programming language with the WAITFORCONDITION statement.
• BackgroundTask	Cyclic task: It is restarted automatically once the assigned programs have been executed. The task cycle time depends on the runtime.
StartupTask	Task is executed once when there is a transition from STOP or STOP U operating state to RUN operating state. SystemInterruptTasks are started by their triggering system event.
ShutdownTask	Task is executed once when there is a transition from RUN operating state to STOP or STOP U operating state. STOP or STOP U operating state is reached by: • Activating the operating state switch • Calling the relevant system function, for example, MCC Change operating state command • Occurrence of a fault with the appropriate error response SystemInterruptTasks and PeripheralFaultTasks are started by their triggering system event.
For information on the behavior of sequential and cyclic tasks: • During initialization of local program variables: See Initialization of local variables (Page 5185). • In the event of execution errors in the program: See SIMOTION Basic Functions Function Manual. For information about options for accessing the process image and I/O variables: see Important properties for direct access and process image (Page 5199).	

7.3.6.4 Task start sequence

When the StartupTask is completed, RUN mode is reached.

The following tasks are then started:

- SynchronousTasks
- TimerInterruptTasks
- BackgroundTask
- MotionTasks with startup attribute.

Note

The sequence in which these tasks are first started after RUN mode has been reached does not conform to the task priorities.

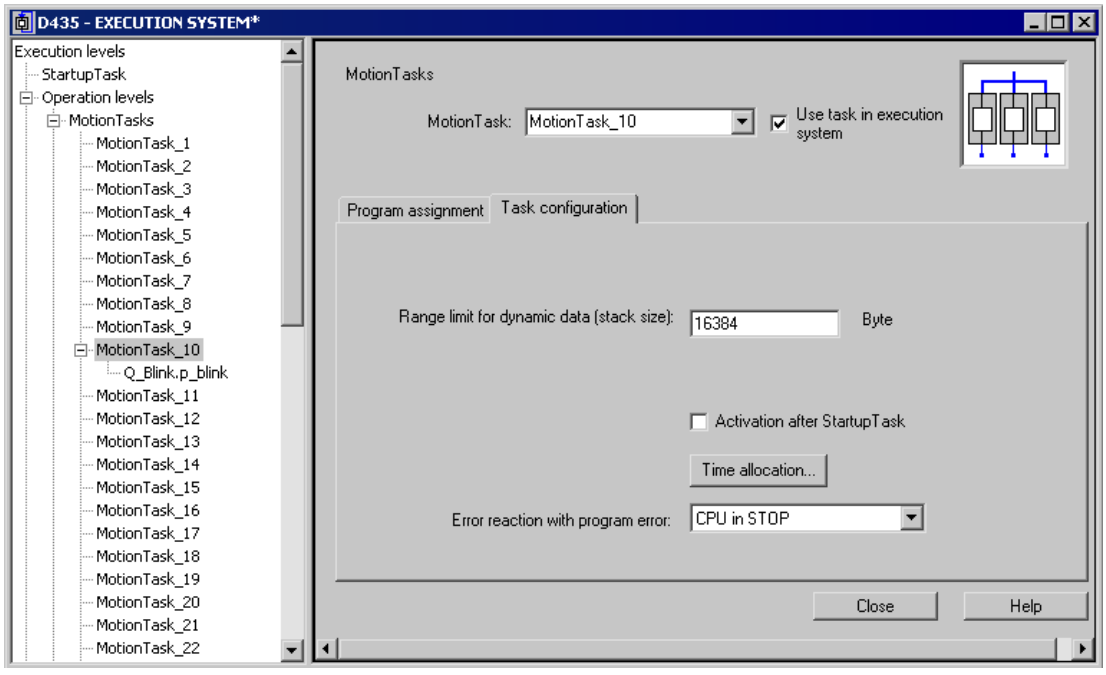


Figure 7-520 Task configuration of a motion task

7.3.6.5 Downloading programs to the target system

The program has to be downloaded into the target system, together with the technology objects etc., before being executed.

To download the program to the target system, proceed as follows:

1. Select **Project > Save and recompile all**.
   The project is locally saved on the hard disk and compiled, with due regard for all dependencies.
2. Select the **Project > Check consistency** menu command to check the project for consistency. This is not necessary if the **Check consistency before loading** option is activated under **Options > Settings** on the **Download** tab (this option is activated by default). This means the consistency check is performed automatically during the download to the target system.

3. Select the **Project > Connect to selected target devices** menu command or click  .
   Online mode is activated.
4. Select the **Target system > Load > Download project to target system** menu command or click  .
   The project data (including the sample program) and the data of the hardware configuration are downloaded to the RAM of the target system.

For more information about downloading a program to the target system, see the SIMOTION Basic Functions Function Manual.

7.3.7 Debugging Software / Error Handling

7.3.7.1 Operating modes for program testing

Modes of the SIMOTION devices

Various SIMOTION device operating modes are available for program testing.

Table 7-593 Operating modes of a SIMOTION device

Operating mode	Meaning
Process mode	Program execution on the SIMOTION device is optimized for maximum system performance. The following diagnostic functions are available, although they may have only restricted functionality because of the optimization for maximum system performance: • Monitor variables in the symbol browser or a watch table • Program status (only restricted): – Restricted monitoring of variables (e.g. variables in loops, return values for system functions). – Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored. • Trace tool (only restricted) with measuring functions for drives and function generator, see online help: – Maximum of **1** trace on each SIMOTION device.
Test mode	The diagnostic functions of the process mode are available to the full extent: • Monitor variables in the symbol browser or a watch table • Program status: – Monitoring of all variables possible. – As of version V4.0 of the SIMOTION Kernel: Several program sources (e.g. ST source files, MCC units, LAD/FBD units)[1] can be monitored per task. – For version V3.2 of the SIMOTION Kernel: Maximum of 1 program source (e.g. ST source file, MCC unit, LAD/FBD unit)[1] can be monitored per task. • Trace tool with measuring functions for drives and function generator, see online help: – Maximum of **4** traces on each SIMOTION device. In addition, the following diagnostics function is available: • Trace for monitoring the program execution in program branches which are executed cyclically (only for the MCC programming language and for SIMOTION Kernel V4.2 and higher). **Note** Runtime and memory utilization increase as the use of diagnostic functions increases.

Operating mode	Meaning
Debug mode	In addition to the diagnostic functions of the test mode, you can use the following functions: • Breakpoints Within a program source, you can set breakpoints (Page 5353). When an activated breakpoint is reached, selected tasks will be stopped. • Controlling MotionTasks On the "Task Manager" tab of the device diagnostics, you can use task control commands for MotionTasks; see the SIMOTION Basic Functions Function Manual. No more than 1 SIMOTION device of the project can be switched to debug mode. SIMOTION SCOUT is in online mode, i.e. connected to the target system. Observe the following section: Important information about the life-sign monitoring (Page 5337).

[1] Each with 1 MCC chart or 1 LAD/FBD program in a program source.

Selecting the operating mode

How to select the operating mode of a SIMOTION device:

1. Make sure a connection to the target system has been established (online mode).
2. Highlight the SIMOTION device in the project navigator.
3. Select the "Operating mode" context menu.
4. Select the required operating mode (see the table above).
   If you have selected "Debug mode":
   – Accept the safety information.
   – Parameterize the sign-of-life monitoring.

   Observe the following section: Important information about the life-sign monitoring (Page 5337).
5. Confirm with **OK**.
   The SIMOTION device switches to the selected operating mode (apart from with debug mode; see the explanation below).

Special features with debug mode

Debug mode can only be selected for one SIMOTION device.

If you have selected debug mode, only SIMOTION SCOUT switches to it; the SIMOTION device is in test mode.

- The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.
- The breakpoints toolbar (Page 5359) is displayed.

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device.

The status bar indicates that debug mode is activated for the SIMOTION device.

Important information about the life-sign monitoring.

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode or a control panel only with the life-sign monitoring function activated with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

In the following cases, the SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection:

- In debug mode with activated breakpoints.
- When controlling an axis or a drive via the control panel (control priority at the PC):

If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis is brought to a standstill.
  - The enables are reset.

Accept safety notes

After selecting the debug mode or a control panel, you must accept the safety notes. You can set the parameters for the life-sign monitoring.

Proceed as follows:

1. Click the **Settings** button.
   The "Debug Settings" window opens.
2. Read there, as described in the following section, the safety notes and parameterize the life-sign monitoring.

Parameterizing the life-sign monitoring

In the "Life-Sign Monitoring Parameters" window, proceed as described below:

1. Read the warning!
2. Click the **Safety notes** button to open the window with the detailed safety notes.
3. Do not make any changes to the defaults for life-sign monitoring.
   Changes should only be made in special circumstances and in observance of all danger warnings.
4. Click **Accept** to confirm you have read the safety notes and have correctly parameterized the life-sign monitoring.

Note

The life-sign monitoring also responds in the following cases:

- Pressing the spacebar.
- Switching to a different Windows application.
- Too high a communication load between the SIMOTION device and SIMOTION SCOUT (e.g. by uploading task trace data).

The following reactions are triggered:

- In debug mode for activated breakpoints:
  - The SIMOTION device switches to the STOP operating state.
  - The outputs are deactivated (ODIS).
- For controlling an axis or a drive using the control panel (control priority for the PC):
  - The axis or the drive is brought to a standstill.
  - The enables are reset.

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Life-sign monitoring parameters

Table 7-594 Life-sign monitoring parameter description

Field	Description
Life-sign monitoring	The SIMOTION device and SIMOTION SCOUT regularly exchange life-signs to ensure a correctly functioning connection. If the exchange of the life-signs is interrupted longer than the set monitoring time, the following reactions are triggered: • In debug mode for activated breakpoints: – The SIMOTION device switches to the STOP operating state. – The outputs are deactivated (ODIS). • For controlling an axis or a drive using the control panel (control priority for the PC): – The axis is brought to a standstill. – The enables are reset. The following parameterizations are possible: • Checkbox **active**: If the checkbox is activated, life-sign monitoring is active. The deactivation of the life-sign monitoring is not always possible. • Monitoring time: Enter the timeout. **Prudence** Do not make any changes to the defaults for life-sign monitoring, if possible. Changes should only be made in special circumstances and in observance of all danger warnings.
Safety information	**Please observe the warning!** Click the button to obtain further safety information. See: Important information about the life-sign monitoring (Page 5337)

7.3.7.2 Editing program sources in online mode

Online editing in process or test mode

If SIMOTION SCOUT is connected to a target system which is in the "process mode" or "test mode" operating mode, program sources (e.g. ST source files, MCC units with MCC charts) can generally be edited, compiled, and loaded to the target system in STOP operating mode. For information on downloading in RUN operating mode, see the corresponding section in the "SIMOTION Basic Functions" Function Manual.

However, you can only activate the "program status", "monitor program execution" (only for MCC), and trace (only for MCC) test functions for a program source or a program organization unit (POU) if the following conditions are met:

1. This program source or any POU of this source (e.g. MCC chart) does not contain any changes which have not been saved.
2. The program source (unit) in SCOUT is consistent with the target system.

Note

If the "program status" test function is activated, editing of the corresponding program source or one of its POUs is disabled.

If an MCC unit or MCC chart is changed and the "monitor program execution" or trace test functions are active for that unit or chart, the test functions are canceled.

Online editing in debug mode

If SIMOTION SCOUT is in debug mode, editing is possible as long as the SIMOTION device is not in debug mode, i.e. no breakpoints are activated.

You can only activate breakpoints and, as a result, switch the SIMOTION device to debug mode if the corresponding program source and all its POUs are saved, compiled so they are up to date, and consistent with the target system.

If you attempt to edit a program source or POU when the SIMOTION device is in debug mode, you are requested to deactivate all breakpoints and, as a result, to switch the SIMOTION device out of debug mode.

Note

If breakpoints have been activated and the SIMOTION device is in debug mode:

Entering a space switches the SIMOTION device to STOP operating mode and deactivates all outputs (ODIS).

7.3.7.3 Symbol Browser

Characteristics

In the symbol browser, you can view and, if necessary, change the name, data type, and variable values. You can see the following variables in particular:

- Unit variables and static variables of a program or function block
- System variables of a SIMOTION device or a technology object
- I/O variables or global device variables.

For these variables, you can:

- View a snapshot of the variable values
- Monitor variable values as they change
- Change (modify) variable values

However, the symbol browser can only display/modify the variable values if the project has been loaded in the target system and a connection to the target system has been established.

Using the symbol browser

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system (see Download programs to the target system (Page 5333)).
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

The procedure depends on the memory area in which the variables to be monitored are stored.

Procedure

Proceed as follows:

1. Select the appropriate element in the project navigator in accordance with the following table.
2. In the detail view, click the **Symbol browser** tab.
   The corresponding variables are displayed in the symbol browser.
3. Select how each variable in the "Display format" column should be displayed.

Table 7-595 Elements in the project navigator and variables to be monitored in the symbol browser

Variables to be monitored in the symbol browser	Element to be selected in the project navigator
Variables in the unit's user memory or in the retentive memory, see SIMOTION ST Programming and Operating Manual: • Retentive and non-retentive unit variables of the interface section of a program source (unit) • Retentive and non-retentive unit variables of the implementation section of a program source (unit) • Static variables of the function blocks whose instances are declared as unit variables. • In addition, if the program source (unit) has been compiled with the "Only create program instance data once" compiler option (Page 5111): – Static variables of the programs. – Static variables of the function blocks whose instances are declared as static variables of programs.	Program source (unit)
Variables in the user memory of the task, see SIMOTION ST Programming and Operating Manual: If the program source (unit) was compiled without the "Only create program instance data once" (default) compiler option (Page 5111), the user memory of the task to which the program was assigned contains the following variables: • Static variables of the programs. • Static variables of the function blocks whose instances are declared as static variables of programs.	EXECUTION SYSTEM
System variables of a SIMOTION device	SIMOTION device
System variables of a technology object	Instance of the technology object
Global device variables	GLOBAL DEVICE VARIABLES
I/O variables (in the Address list tab of the detail view). The Address list tab of the detail view can be opened by double-clicking the ADDRESS LIST element in the project navigator.	ADDRESS LIST

Note

You can monitor temporary variables (together with unit variables and static variables) with **Program status** (see Properties of the program status (Page 5348)).

Note

Trace diagnostic function for MCC programming

Various internal variables, whose identifier begins with an underscore, are automatically created by the compiler for the trace diagnostic function. These variables are displayed in the symbol browser.

With activated diagnostic function, these variables are used for the control of the diagnostics function. These variables must not be used in the user program.

Status and controlling variables

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox in this column
3. Click the **Immediate control** button.

The values you entered are written to the selected variables.

⚠ WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the symbol browser. There is therefore no guarantee of consistency.

Working with the symbol browser

The functions of the symbol browser and how you work with them are described in detail in the online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the symbol browser (independently of the SIMOTION device):

Table 7-596 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

7.3.7.4 Watch tables

Monitoring variables in watch table

Watch table options

With the symbol browser, you see only the variables of an object within the project. With program status, you see only the variables for a freely selectable monitoring area in the program

With watch tables, by contrast, you can monitor selected variables from different sources as a group (e.g. program sources, technology objects, SINAMICS drives - even on different devices).

You can see the data type of the variables in offline mode. You can view and also modify the value of the variables in online mode.

Creating a watch table

Procedure for creating a watch table and assigning variables:

1. In the project navigator, open the **Monitor** folder.
2. Double-click the **Insert watch table** entry to create a watch table and enter a name for it. A watch table with this name appears in the **Monitor** folder.
3. In the project navigator, click the object from which you want to move variables to the watch table.
4. In the symbol browser, select the corresponding variable line by clicking its number in the left column.
5. From the context menu, select **Add to watch table** and the appropriate watch table, e.g. **Watch table_1**.
6. If you click the watch table, you will see in the detail view of the **Watch table** tab that the selected variable is now in the watch table.
7. Repeat steps 3 to 6 to monitor the variables of various objects.

If you are connected to the target system, you can monitor the variable contents.

Status and controlling variables

In the **Status value** column, the current variable values are displayed and periodically updated.

You can change the value of one or several variables. Proceed as follows for the variables to be changed:

1. Enter a value in the **Control value** column.
2. Activate the checkbox in this column
3. Click the **Immediate control** button.

The values you entered are written to the selected variables.

⚠ WARNING

Dangerous plant states possible

You assign the entered values to the variables during control. This can result in dangerous plant states, e.g. unexpected axis motion.

Note

Note when you change the values of several variables:

The values are written sequentially to the variables. It can take several milliseconds until the next value is written. The variables are changed from top to bottom in the watch table. There is therefore no guarantee of consistency.

Working with the watch table

The functions of the watch table and how you work with them are described in detail in the Online help.

Display invalid floating-point numbers

Invalid floating-point numbers are displayed as follows in the watch table (independently of the SIMOTION device):

Table 7-597 Display invalid floating-point numbers

Display	Meaning
1.#QNAN -1.#QNAN	Invalid bit pattern in accordance with IEEE 754 (NaN Not a Number). There is no distinction between signaling NaN (NaNs) and quiet NaN (NaNq).
1.#INF -1.#INF	Bit pattern for + infinity in accordance with IEEE 754 Bit pattern for – infinity in accordance with IEEE 754
-1.#IND	Bit pattern for indeterminate

7.3.7.5 Variable status

"Variable status" enables you to monitor the current value for an individual variable, selected using the cursor, in an open program source or program organization unit (e.g. ST source file, MCC chart, LAD program).

Requirements

- Make sure that a connection to the target system has been established and a project has been downloaded to the target system. For information on loading a project, see "Downloading programs to the target system (Page 5333)".
- The program source containing the program organization unit (POE) whose variables you want to monitor must be consistent with the target system.

- The associated source (e.g. ST source file, MCC chart, LAD program) must be open.
- With the MCC programming language only: The parameter screen form for the command in which the variable you want to monitor is being used must be open.
- You can run the user program, but you do not have to. If the program is not run, you only see the initial values of the variables.

Procedure

To monitor an individual variable using variable status:

1. Position the cursor above the identifier for a variable.
   - With the ST programming language: in the open ST source file
   - With the ST programming language: within an input field in the open parameter screen form
   - With the LAD/FBD programming language: within a network of the LAD/FBD program
2. Briefly position the cursor above the identifier.

The tool tip shows the current value of the variable. If you keep the cursor above the identifier for a longer period, the value is updated on an ongoing basis.

Note

With "variable status", the current value for the variable is displayed, wherever the selected variable is being used.

The "variable status" function enables you to monitor all those variables you are also able to monitor in the symbol browser (Page 5341) or the address list. These are:

- System variables of SIMOTION devices
- System variables of technology objects
- Global device variables
- Retentive and non-retentive unit variables of the interface section of a program source (unit)
- Retentive and non-retentive unit variables of the implementation section of a program source (unit)
- Static variables of the programs
- Static variables of the function blocks whose instances are declared as unit variables
- Static variables of the function blocks whose instances are declared as static variables of programs
- I/O variables

7.3.7.6 Trace

Trace options

Trace allows you to record and save signal characteristics of inputs/outputs or the variable values. This allows you to document the optimization, for example, of axes.

You can set the recording time, display up to four channels with eight values each in the test or debug mode, select trigger conditions, parameterize timing adjustments, select between different curve displays and scalings, etc.

The SIMOTION online help provides additional information on the trace tool.

7.3.7.7 Program run

Program run: Display code location and call path

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path.

Follow these steps:

1. Click the **Show program run** button on the Program run toolbar.
   The "Program run call stack (Page 5347)" window opens.
2. Select the desired MotionTask.
3. Click the **Update** button.

The window shows:

- The position in the code being executed (e.g. line of the ST source file) stating the program source and the POU.
- Recursively positions in the code of other POUs that call the code position being executed.

The following names are displayed for the SIMOTION RT program sources:

Table 7-598 SIMOTION RT program sources

Name	Meaning
taskbind.hid	Execution system
stdfunc.pck	IEC library
device.pck	Device-specific library
tp-name.pck	Library of the *tp-name* technology package, e.g. cam.pck for the library of the CAM technology package

Program run parameters

You can display the following for all configured tasks:

- the current code position in the program code (e.g. line of an ST source file)
- the call path of this code position

Table 7-599 Program run parameter description

Array	Description
Selected CPU	The selected SIMOTION device is displayed.
Refresh	Clicking the button reads the current code positions from the SIMOTION device and shows them in the open window.
Calling task	Select the task for which you want to determine the code position being executed. All configured tasks of the execution system.
Current code position	The position being executed in the program code (e.g. line of an ST source file) is displayed (with the name of the program source, line number, name of the POU).
is called by	The code positions that call the code position being executed within the selected task are shown recursively (with the name of the program source, line number, name of the POU, and name of the function block instance, if applicable).

For names of the SIMOTION RT program sources, refer to the table in Program run (Page 5347).

Program run toolbar

You can display the position in the code (e.g. line of an ST source file) that a MotionTask is currently executing along with its call path with this toolbar.

Table 7-600 Program run toolbar

Symbol	Meaning
	Display program run Click this symbol to open the Program run call stack window. In this window, you can display the currently active code position with its call path. See: Program run: Display code position and call path (Page 5347)

7.3.7.8 Program status (monitoring program execution)

Monitoring the program execution

Monitoring the program execution does not affect the actual execution of the program, but does increase the communication load. This has an impact on the execution of MotionTasks and the BackgroundTask.

Program status can be switched on and off during all operating modes of a SIMOTION device (Page 5335).

Note

In the process mode, the program status can be called only once for a LAD/FBD program, FB or FC. If you do not observe the restriction, error message 25023 "No resources in the runtime" will appear.

In the test mode, the program status can be called simultaneously for several LAD/FBD programs, FB or FC. The maximum possible number depends on the utilization of the SIMOTION device.

The values of the following variables are displayed:

- Simple data type variables (INT, REAL, etc.)
- Individual elements of a structure, provided an assignment is made
- Individual elements of an array, provided an assignment is made
- Enumeration data type variables

Note

The values of constants are not displayed.

Due to the restricted buffer capacity and the requirement for minimum runtime tampering, the following variables cannot be displayed:

- Complete arrays
- Complete structures

Individual array elements or individual structure elements are displayed, however, provided an assignment is made in the ST source file.

Starting and stopping program status (monitoring program execution)

You can call up information on network status in two different ways using program status:

- You can select specific networks to display their status. Multiple selection (Shift key) and the selection of all networks (Ctrl+A) is possible. The boxes will be displayed in the same color as the corresponding output.
  The left-hand border of a selected network is highlighted in blue.
- If you do not select a network, the status is displayed for those networks that are visible on the screen. If you scroll down, the status of those visible networks is now displayed.

Preparing program status

Before you can work with program status, additional code must be generated during compilation:

1. Select the **Project > Connect to selected target devices** menu command or click the button.
   Online mode is activated.
2. Select the SIMOTION device, followed by **Operating mode** in the context menu.
3. Select **Test mode**.
   Program status is available in this operating mode without restrictions, see Operating modes for SIMOTION devices (Page 5335).
4. Open the Properties window for the LAD/FBD unit, see Defining the properties of an LAD/FBD unit (Page 5109).
5. Activate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (Page 5112) for this LAD/FBD unit.

   Note

   Alternatively, select **Options > Settings > Compiler**, then activate the **Permit program status** compiler option as the global compiler setting (Page 5111) for all program sources (ST, MCC and LAD/FBD units).

6. Select the **LAD/FBD unit > Accept and compile** menu command.
   The LAD/FBD unit is compiled.

   Note

   If the **Permit program status** global compiler option is changed, you need to select **Project > Save and compile changes** to compile all the program sources affected by the change.

7. Select the **Target system > Load > Project to target system** menu command or click the button.
   The programs are downloaded to the target system. Make sure that the target system is in STOP mode.

Starting the Status program

Requirement

- The corresponding LAD/FBD program is opened.
- The LAD/FBD program does not contain any unsaved changes.
- The LAD/FBD unit in SCOUT must be consistent with the target system.

Procedure

To start program status, proceed as follows:

1. Select the **LAD/FBD program > Program status on/off** menu command or click the button  for **Program status** (Ctrl+F7 shortcut) to start this test mode.
2. If the LAD/FBD program is assigned to more than one task, the **Call path/task selection Program status** dialog box opens:
   – In this dialog box, select the task in which you want to monitor the program.

   The 

symbol on the tab for the open LAD/FBD program indicates that program status has started, as does the symbol for the respective combination of the currently activated test functions (program status, breakpoints).

Note

Start program status for multiple windows for each SIMOTION device

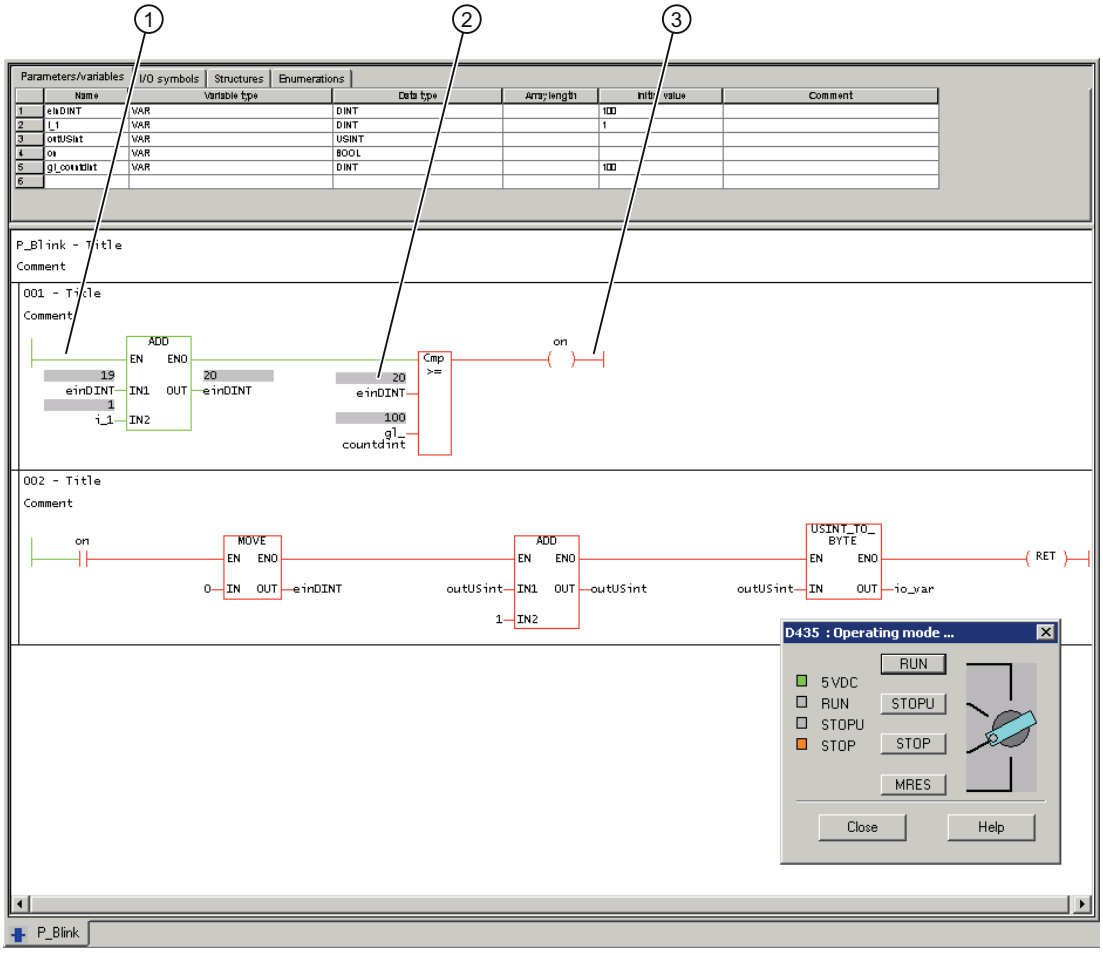
A window can be an open ST source file, an open LAD/FBD program or an open MCC chart. An active window is located in the foreground and the current status values are displayed (MCC: open MCC command).

In **Process mode**, program status can only be started for one window.

In **Test mode**, program status can be started simultaneously for several windows per task. The maximum possible number depends on the utilization of the SIMOTION device.

If program status is started for an additional window (**Process mode**) or starting it causes the maximum possible number to be exceeded (**Test mode**), program status is automatically deactivated for the currently active window (**Process mode**) or the oldest active window (**Test mode**), i.e. momentarily paused, but not stopped. If a window with deactivated program status is brought to the foreground (MCC: open MCC command), program status is automatically reactivated and the current status values are displayed again.

If the program execution monitoring is activated, the ladder diagram lines/signal paths of the selected networks or the networks displayed on the screen are colored according to the current values:



① Green: Lines: Value = TRUE (Data type BOOL)

Boxes: Result = TRUE or box is activated (enable input EN = TRUE).

Non-binary connections are shown in green.

Binary connections are displayed according to their values (green or red).

② Gray: Only when boxes are active: Non-binary values are shown on gray background.

③ Red: Lines: Value = FALSE (Data type BOOL)

Boxes: Result = TRUE or box is activated (enable input EN = TRUE).

Non-binary connections are shown in red.

When the boxes are inactive, binary connections are shown in cyan.

Figure 7-521 Online display in the LAD/FBD editor

Note

When a constant is used in a network, the current value of the constant is also displayed during program execution.

A constant which is not included is colored turquoise.

Stopping program status

To stop program status, proceed as follows:

1. Select the **LAD/FBD program > Program status on/off** menu command or click the button for **Program status** (Ctrl+F7 shortcut).
   Program status is stopped.

Disabling program status

Disabling program status frees up CPU resources.

To disable program status, proceed as follows:

1. First, stop the program status function, see Stopping program status.
2. Open the Properties window for the LAD/FBD unit, see Defining the properties of an LAD/FBD unit (Page 5109).
3. Deactivate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (Page 5112) for this LAD/FBD unit and confirm by clicking **OK**.

   Note

   If the **Permit program status** compiler option is activated as the global setting for the compiler (Page 5111), you can deactivate this for either one LAD/FBD unit by deactivating the relevant **Use global settings** checkbox, see Local compiler settings (Page 5112) or for all the program sources by selecting **Options > Settings > Compiler**.

4. Select the SIMOTION device, followed by **Operating mode** in the context menu.
5. Select **Process mode**.
   Program execution is optimized for this operating mode to ensure maximum performance, see Operating modes for SIMOTION devices (Page 5335).
6. Recompile the program and download it to the target system.

Note

If the **Permit program status** global compiler option is changed, you need to select **Project > Save and compile changes** to compile all the program sources affected by the change.

7.3.7.9 Breakpoints

General procedure for setting breakpoints

You can set breakpoints within a POU of a program source (e.g. ST source, MCC chart, LAD/FBD source). On reaching an activated breakpoint, the task in which the POU with the breakpoint is called is stopped. If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser.

Requirement:

- The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.

Procedure

Follow these steps:

1. Select "Debug mode" for the associated SIMOTION device; see Setting debug mode (Page 5354).
2. Specify the tasks to be stopped, see Specifying the debug task group (Page 5355).
3. Set breakpoints, see Setting breakpoints (Page 5357).
4. Define the call path, see Defining a call path for a single breakpoint (Page 5360).
5. Activate the breakpoints, see Activating breakpoints (Page 5364).

Setting the debug mode

⚠ WARNING

Dangerous plant states possible

If problems occur in the communication link between the PC and the SIMOTION device, this may result in dangerous plant states (e.g. the axis may start moving in an uncontrollable manner).

Therefore, use the debug mode only with activated life-sign monitoring (Page 5337) with a suitably short monitoring time!

You must observe the appropriate safety regulations.

The function is released exclusively for commissioning, diagnostic and service purposes. The function should generally only be used by authorized technicians. The safety shutdowns of the higher-level control have no effect!

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Requirement

1. A connection to the target system must have been established (online mode)
2. Debug mode must not be selected for any SIMOTION device.

Procedure

To set the debug mode, proceed as follows:

1. Highlight the SIMOTION device in the project navigator.
2. Select **Operating mode** from the context menu.
3. Select **Debug** mode (Page 5335).

4. Accept the safety information
5. Parameterize the sign-of-life monitoring.
   See also section: Important information about the life-sign monitoring (Page 5337).
6. Confirm with **OK**.

SIMOTION SCOUT switches to debug mode for this device; the SIMOTION device itself remains in "test mode", as long as at least one breakpoint is activated:

The project navigator indicates that debug mode is activated for SIMOTION SCOUT by means of a symbol next to the SIMOTION device.

The breakpoints toolbar (Page 5359) is displayed.

As long as no breakpoints are activated, you can edit program sources in debug mode (Page 5339).

Debug mode is not enabled for the SIMOTION device until at least one set breakpoint is activated. If all breakpoints are deactivated, debug mode is canceled for the SIMOTION device. The status bar indicates that debug mode is activated for the SIMOTION device.

Note

Pressing the spacebar or switching to a different Windows application causes the following to happen if the SIMOTION device is in debug mode (breakpoints activated):

- The SIMOTION device switches to the STOP operating state.
- The outputs are deactivated (ODIS).

⚠ WARNING

Dangerous plant states possible

This function is not guaranteed in all operating states.

Therefore, there must be an EMERGENCY STOP circuit in the hardware. The appropriate measures must be taken by the user.

Define the debug task group

On reaching an activated breakpoint, all tasks that are assigned to the debug task group are stopped.

Requirement

1. A connection to the target system must have been established (online mode).
2. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 5354).

Procedure

How to assign a task to the debug task group:

1. Highlight the relevant SIMOTION device in the project navigator.
2. Select **Debug task group** from the context menu.
   The Debug Task group window opens.
3. Select the tasks to be stopped on reaching the breakpoint:
   - If you only want to stop individual tasks (in RUN operating state): Activate the **Debug task group** selection option.
     Assign all tasks to be stopped on reaching a breakpoint to the **Tasks to be stopped** list.
   - If you only want to stop individual tasks (in HOLD operating state): Activate the **All tasks** selection option.
     In this case, also select whether the outputs and technology objects are to be released again after resumption of program execution.

Note

Note the different behavior when an activated breakpoint is reached, see the following table.

Table 7-601 Behavior at the breakpoint depending on the tasks to be stopped in the debug task group.

Property		Tasks to be stopped	
		Single selected tasks (debug task group)	**All tasks**
Behavior on reaching the breakpoint			
	Operating state	RUN	STOP
	Stopped tasks	Only tasks in the debug task group	All tasks
	Outputs	Active	Deactivated (ODIS activated)
	Technology	Closed-loop control active	No closed-loop control (ODIS activated)
	Runtime measurement of the tasks	Active for all tasks	Deactivated for all tasks
	Time monitoring of the tasks	Deactivated for tasks in the debug task group	Deactivated for all tasks
	Real-time clock	Continues to run	Continues to run
Behavior on resumption of program execution			
	Operating state	RUN	RUN
	Started tasks	All tasks in the debug task group	All tasks

Property		Tasks to be stopped	
		Single selected tasks (debug task group)	All tasks
	Outputs	Active	The behavior of the outputs and the technology objects depends on the **'Continue' activates the outputs (ODIS deactivated)** checkbox. • Active: ODIS will be deactivated. All outputs and technology objects are released. • Inactive: ODIS remains activated. All outputs and technology objects are only enabled for one STOP-RUN transition.
	Technology	Closed-loop control active	

Note

You can only make changes to the debug task group if no breakpoints are active.

The settings of the debug task group are retained after exiting "Debug mode".

Proceed as follows:

1. Set breakpoints (see Setting breakpoints (Page 5357)).
2. Define the call path (see Defining a call path for a single breakpoint (Page 5360)).
3. Activate the breakpoints (see Activating breakpoints (Page 5364)).

Setting breakpoints

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 5354).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 5355).

Procedure

How to set a breakpoint:

1. Select the code location where no breakpoint has been set:
   - SIMOTION ST: Place the cursor on a line in the ST source file that contains a statement.
   - SIMOTION MCC: Select an MCC command in the MCC chart (except module or comment block).
   - SIMOTION LAD/FBD: Set the cursor in a network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove a breakpoint, proceed as follows:

1. Select the code position with the breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Set/remove breakpoint** menu command (shortcut F9).
   - Click the  button in the Breakpoints toolbar.

To remove all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Remove all breakpoints** menu command (shortcut CTRL+F5).
  - Click the  button in the Breakpoints toolbar.

Note

You cannot set breakpoints:

- For SIMOTION ST: In lines that contain only comment.
- For SIMOTION MCC: On the module or comment block commands.
- For SIMOTION LAD/FBD: Within a network.
- At code locations in which other debug points (e.g. trigger points) have been set.

You can list the debug points in all program sources of the SIMOTION device in the debug table:

- Click the  button for "debug table" in the Breakpoints toolbar.

In the debug table, you can also remove all breakpoints (in all program sources) of the SIMOTION device:

- Click the button for "Clear all breakpoints".

The breakpoints set also remain saved after leaving debug mode; they are displayed in debug mode only.

You can use the program status (Page 5349) diagnosis functions and breakpoints together in a program source or POU. However, the following restrictions apply depending on the program languages:

- SIMOTION ST: For version V3.2 of the SIMOTION Kernel, the (marked) ST source file lines to be tested with program status must not contain a breakpoint.
- SIMOTION MCC and LAD/FBD: The commands of the MCC chart (or networks of the LAD/FBD program) to be tested with program status must not contain a breakpoint.

Proceed as follows

1. Define the call path, see Defining a call path for a single breakpoint (Page 5360).
2. Activate the breakpoints, see Activating breakpoints (Page 5364).

Breakpoints toolbar

This toolbar contains important operator actions for setting and activating breakpoints:

Table 7-602 Breakpoints toolbar

Symbol	Meaning
	Set/remove breakpoint Click this icon to set at breakpoint for the selected code position or to remove an existing breakpoint. See: Setting breakpoints (Page 5357).
	Activate/deactivate breakpoint Click this icon to activate or deactivate the breakpoint at the selected code position. See: Activating breakpoints (Page 5364).
	Edit the call path Click this icon to define the call path for the breakpoints: • If a code position **with** breakpoint is selected: The call path for this breakpoint. • If a code position **without** breakpoint is selected: The call path for all breakpoints of the POU. See: Defining the call path for a single breakpoint (Page 5360), Defining the call path for all breakpoints (Page 5362).
	Activate all breakpoints of the active POU Click this symbol to activate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 5364).
	Deactivate all breakpoints of the active POU Click this symbol to deactivate all breakpoints in the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Activating breakpoints (Page 5364).
	Remove all breakpoints of the active POU Click this symbol to remove all breakpoints from the active program source or POU (e.g. ST source file, MCC chart, LAD/FBD program). See: Setting breakpoints (Page 5357).

Symbol	Meaning
	Debug table Click this icon to display the debug table. See: Debug table parameters.
	Display call stack Click this icon after reaching an activated breakpoint to: • View the call path at the current breakpoint. • View the code positions at which the other tasks of the debug task group have been stopped together with their call path. See: Displaying the call stack (Page 5366).
	Resume Click this icon to continue the program execution after reaching an activated breakpoint. See: Resuming program execution (Page 5367), Displaying the call stack (Page 5366).
	Next step (SIMOTION Kernel as of version V4.4) Only available for the MCC and LAD/FBD programming languages: Click this icon to resume the program execution until the next MCC command or LAD/FBD network is reached. See: Resume program execution in single steps (Page 5367).
	Step through the subprogram (SIMOTION Kernel as of version V4.4) Only available for the MCC programming language. Click this icon to jump to the called subprogram and stop at the first command. The subprogram must be created in the MCC or LAD/FBD programming language. See: Resume program execution in single steps (Page 5367).

Defining the call path for a single breakpoint

Requirements:

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 5354).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 5355).
5. Breakpoint is set, see Setting breakpoints (Page 5357).

Procedure

To define the call path for a single breakpoint, proceed as follows:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the Call path / task selection breakpoint window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when the selected breakpoint is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when the activated breakpoint in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when the breakpoint in the selected task is reached. The task must be in the debug task group.
     The specification of a call path is possible.
4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at the activated breakpoint if the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at the activated breakpoint only when the POU is called from the selected code position.
     If the POU of the selected calling code position is also called from other code positions, further lines are displayed successively in which you proceed similarly.
5. If the breakpoint is only to be activated after the code position has been reached several times, select the number of times.

Note

You can also define the call path to the individual breakpoints in the debug table:

1. Click the button for "debug table" in the Breakpoints toolbar.
   The "Debug table" window opens.
2. Click the appropriate button in the "Call path" column.
3. Proceed in the same way as described above:
   – Specify the task.
   – Define the call path (only for functions and function blocks).
   – Specify the number of passes after which the breakpoint is to be activated.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 5364).

Note

You can use the "Display call stack (Page 5366)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for all breakpoints (Page 5362)

Defining the call path for all breakpoints

With this procedure, you can:

- Select a default setting for all future breakpoints in a POU (e.g. MCC chart, LAD/FBD program or POU in an ST source file).
- Accept and compare the call path for all previously set breakpoints in this POU.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 5354).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 5355).

Procedure

To define the call path for all future breakpoints of a POU, proceed as follows:

1. Select the code location where **no** breakpoint has been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Click the  button for "edit call path" in the Breakpoints toolbar.
   In the "Call path / task selection all breakpoints for each POU" window, the marked code position is displayed (with the name of the program source, line number, name of the POU).
3. Select the task in which the user program (i.e. all tasks in the debug task group) will be stopped when a breakpoint in this POU is reached.
   The following are available:
   - **All calling locations starting at this call level**
     The user program will always be started when an activated breakpoint of the POU in any task of the debug task group is reached.
   - The individual tasks from which the selected breakpoint can be reached.
     The user program will be stopped only when a breakpoint in the selected task is reached.
     The task must be in the debug task group.
     The specification of a call path is possible.
4. Only for functions and function blocks: Select the call path, i.e. the code position to be called (in the calling POU).
   The following are available:
   - **All calling locations starting at this call level**
     No call path is specified. The user program is always stopped at an activated breakpoint when the POU in the selected tasks is called.
   - Only when a single task is selected: The code positions to be called within the selected task (with the name of the program source, line number, name of the POU).
     The call path is specified. The user program will be stopped at an activated breakpoint only when the POU is called from the selected code position.
     If the selected calling code position is in turn called by other code positions, further lines are displayed successively in which you proceed similarly.
5. If a breakpoint is only to be activated after the code position has been reached several times, select the number of times.
6. If you want to accept and compare this call path for all previously set breakpoints in this POU:
   - Click **Accept**.

Proceed as follows:

- Activate the breakpoints, see Activating breakpoints (Page 5364).

Note

You can use the "Display call stack (Page 5366)" function to view the call path at a current breakpoint and the code positions at which the other tasks of the debug task group were stopped.

See also

Defining the call path for a single breakpoint (Page 5360)

Activating breakpoints

Breakpoints must be activated if they are to have an effect on program execution.

Requirements

1. The program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) is open.
2. A connection to the target system must have been established (online mode).
3. SIMOTION SCOUT is in debug mode for the corresponding SIMOTION device; see Setting debug mode (Page 5354).
4. The tasks to be stopped are specified, see Specifying the debug task group (Page 5355).
5. Breakpoints are set, see Setting breakpoints (Page 5357).
6. Call paths are defined, see Defining a call path for a single breakpoint (Page 5360).

Activating breakpoints

How to activate a single breakpoint:

1. Select the code location where a breakpoint has already been set:
   - SIMOTION ST: Set the cursor in an appropriate line of the ST source file.
   - SIMOTION MCC: Select an appropriate command in the MCC chart.
   - SIMOTION LAD/FBD: Set the cursor in an appropriate network of the LAD/FBD program.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the  button in the Breakpoints toolbar.

To activate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Activate all breakpoints** menu command.
  - Click the  button in the Breakpoints toolbar.

Once the first breakpoint has been activated, the SIMOTION device switches to debug mode. It remains in this mode until the last breakpoint is deactivated.

In the Task status function bar, (Page 5368) the tasks with activated breakpoints are highlighted in gray ( ).

Note

Breakpoints of all program sources of the SIMOTION device can also be activated and deactivated in the debug table:

1. Click the  button for "debug table" in the Breakpoints toolbar.
   The "Debug Table" window opens.
2. Perform the action below, depending on which breakpoints you want to activate or deactivate:
   - Single breakpoints: Check or clear the corresponding checkboxes.
   - All breakpoints (in all program sources): Click the corresponding button.

The following applies up to version V4.3 of the SIMOTION Kernel:

- In the case of activated breakpoints, the "Single step" test function of the SIMOTION MCC programming language cannot be used.

The following applies as of version V4.4 of the SIMOTION Kernel:

- The "Single step" test function of the SIMOTION MCC programming language is not available in the Debug mode.

Breakpoints cannot be activate if the control priority is at the axis control panel. Conversely, you cannot fetch the control priority for the axis control panel when a breakpoint activated.

Behavior at the activated breakpoint

On reaching an activated breakpoint (possibly using the selected call path (Page 5360)), all tasks assigned to the debug task group will be stopped. The behavior depends on the tasks in the debug task group and is described in "Defining a debug task group (Page 5355)". The breakpoint is highlighted.

In the Task status function bar, (Page 5368) the task in which the breakpoint was reached is highlighted in red ( ).

The following applies to the programming languages MCC or LAD/FBD: If the debug task group is stopped by a breakpoint, then the user has the option to change to another task, belonging to the debug task group, in the combo box. Always the breakpoint of the currently selected task is visualized.

If the breakpoint that initiated the stopping of the tasks is located in a program or function block, the values of the static variables for this POU are displayed in the "Variables status" tab of the detail display. Temporary variables (also in/out parameters for function blocks) are not displayed. You can monitor static variables of other POUs or unit variables in the symbol browser (Page 5341).

You can use the "Display call stack (Page 5366)" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Resuming program execution

You can resume the execution of the stopped tasks, see "Resuming program execution" (Page 5367).

As of version V4.4 of the SIMOTION Kernel, you can resume the task in single steps that has been stopped at the activated breakpoint in the MCC and LAD/FBD programming languages, see Resuming program execution in single steps (Page 5367).

Deactivate breakpoints

To deactivate a single breakpoint, proceed as follows:

1. Select the code position with the activated breakpoint.
2. Perform the following (alternatives):
   - Select the **Debug > Activate/deactivate breakpoint** menu command (shortcut F12).
   - Click the  button in the Breakpoints toolbar.

To deactivate all breakpoints (in all program sources) of the SIMOTION device, proceed as follows:

- Perform the following (alternatives):
  - Select the **Debug > Deactivate all breakpoints** menu command.
  - Click the  button in the Breakpoints toolbar.

Once the last breakpoint has been deactivated, the SIMOTION device switches to "test mode"; SIMOTION SCOUT continues to run in debug mode.

Display call stack

You can use the "Display call stack" function to:

- View the call path at the current breakpoint.
- View the code positions with the call path at which the other tasks of the debug task group have been stopped.

Requirement

The user program is stopped at an activated breakpoint, i.e. the tasks of the debug task group (Page 5355) have been stopped.

Procedure

To call the "Display call stack" function, proceed as follows:

- Click the  button for "display call stack" in the Breakpoints toolbar.
  The "Breakpoint call stack" dialog opens. The current call path (including the calling task and the number of the set passes) is displayed.
  The call path cannot be changed.

To use the "Display call stack" function, proceed as follows:

1. Keep the "Breakpoint call stack" dialog open.
2. To display the code position at which the other task was stopped, proceed as follows:
   – Select the appropriate task. All tasks of the debug task group can be selected.

   The code position, including the call path, is displayed. If the code position is contained in a user program, the program source with the POU (e.g. ST source file, MCC chart, LAD/FBD program) will be opened and the code position marked.
3. How to resume program execution:
   – Click the button for "resume" (Ctrl+F8 shortcut) on the Breakpoint toolbar.

   When the next activated breakpoint is reached, the tasks of the debug task group will be stopped again. The current call path, including the calling task, is displayed.
4. Click **OK** to close the "Breakpoint call stack" dialog box.

For names of the SIMOTION RT program sources, refer to the table in "Program run (Page 5347)".

Resuming program execution

How to resume program execution:

- Perform the following (alternatives):
  – Select the **Debug > Continue** menu command (shortcut CTRL+F8).
  – Click the button on the Breakpoint toolbar (Page 5359) to "Continue".

The stopped task is continued until the next active breakpoint is reached.

Resuming program execution in single steps (as of Kernel V4.4)

This function is available as of SIMOTION Kernel version V4.4.

In the MCC and LAD/FBD programming languages you can resume the task in single steps that was stopped at an activated breakpoint (Page 5353).

The current MCC command or the current LAD/FBD network and all stopped tasks of the debug task group are executed.

All tasks that are assigned to the debug task group are stopped at the following MCC command or LAD/FBD network or the next activated breakpoint within the debug task group.

Next step

To execute the current MCC command or the current LAD/FBD network at which the program has been stopped, proceed as follows:

- Perform the following (alternatives):
  – Select the **Debug > Next step** menu command (shortcut CTRL+F10).
  – Click the button for "Next step" in the Breakpoints toolbar (Page 5359).

The current MCC command or the current LAD/FBD network is executed. The program is stopped at the following MCC command or the current LAD/FBD network.

A subprogram call is executed without interruption as long as no breakpoint is activated within the subprogram. If a

Stepping through the subprogram (MCC only)

If the program execution has been stopped at the "Subprogram call" command in the MCC programming language, you can jump to the subprogram and run through it in single steps. The subprogram must have been created in the MCC or LAD/FBD programming language.

- Perform the following (alternatives):
  - Select the **Debug > Step through subprogram** menu command (shortcut CTRL+SHIFT +F10).
  - Click the  button on the Breakpoint toolbar (Page 5359) to "Step through subprogram".

The appropriate MCC chart or LAD/FBD program is opened and the program execution stopped at the first MCC command or LAD/FBD network.

Note

Stepping is not possible in MCC charts and LAD/FBD programs whose sources are in libraries.

Stepping is not possible in ST source files.

You can only open MCC charts and LAD/FBD programs with know-how protection if you have the required authorization (e.g. password).

7.3.7.10 Task status function bar

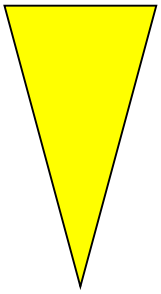
In a combo box, the Task status function bar displays all tasks of the active SIMOTION device, to which a program is assigned.

They are displayed under the following conditions:

1. SIMOTION SCOUT is in online mode.
2. The affected SIMOTION device is active, e.g.
   - In the project navigator, the SIMOTION device or an element in its subtree is selected (such as program source, technology object).
   - In the working area, an open window is active that belongs to an element in the subtree of the SIMOTION device.
3. The SIMOTION device is consistent.

A background color highlights the occurrence of specific events in the affected task, see the following table. The task in question is displayed in the combo box of the function bar according to the event priority.

Table 7-603 Meaning of background colors in the Task status function bar

Background color	Meaning	Priority
Cyan	The affected task waits for a command at the "Single step" test function (only for SIMOTION MCC programming language).	Highest  Lowest
Red	The affected task is located at a breakpoint (Page 5364).	
Blue	In the affected task, the "Single step" test function is activated (only for SIMOTION MCC programming language)	
Gray	In the affected task, at least 1 breakpoint (Page 5364) is activated.	
Yellow	In the affected task, the "Monitoring" test function is activated (only for SIMOTION MCC programming language).	
White	In the affected task, none of the above-mentioned test functions are activated.	

Note

A selection of a task in the combo box is only possible:

- For the following test functions of the SIMOTION MCC programming language:
  - Monitoring
  - Single step
  - Trace
- at activated breakpoints (Page 5364) in the MCC or LAD/FBD programming languages.

7.3.7.11 Project comparison

SIMOTION SCOUT has a **project comparison** function (start this via the **Start object comparison** button) for comparing objects within the same project and/or objects from different projects (online or offline).

Project comparison allows you to establish any differences and, if necessary, run a data transfer to rectify them.

Objects are devices and their sub-objects, programs, technology objects (TOs) or drive objects (DOs), and libraries. Comparing projects is useful if you need to carry out service work on the system.

Further information on project and detail comparisons can be found in the SIMOTION Project Comparison Function Manual.

7.3.8 Application Examples

7.3.8.1 Examples

You will be given an introduction to the LAD and FBD programming languages using two simple examples.

7.3.8.2 Creating sample programs

Requirements for program creation

The project is the highest level in the data management hierarchy. SIMOTION SCOUT saves all data which belongs, for example, to a production machine, in the project directory.

This means that the project therefore brackets together all SIMOTION devices, drives, etc., belonging to one machine.

Within the project, the hardware used must be made known to the system, including:

- SIMOTION device
- Centralized I/O (with I/O addresses)
- Distributed I/O (with I/O addresses)

A SIMOTION device must be configured before you can insert and edit LAD/FBD sources.

Sample programs

We will create two short programs (position blinker program, axis program) that demonstrate all the work steps from the creation through to the start and testing of a program.

7.3.8.3 Blinker program

Prerequisites

A project must have been created and the hardware used in the project must be known to the system.

Task specification

Output of a cyclically changing bit pattern after exceeding a limit value.

This task is divided into the following parts:

- Insert LAD/FBD source file
- Insert LAD/FBD program
  Network one
  A program variable is incremented and compared to a reference value
  Network two
  When the reference value is exceeded, the program variable is reset and a bit pattern is output
- Compiling
- Insert program in a task
- Download program onto target device

You can observe the result of your program at the outputs of your target system.

This example deals only with the LAD programming aspect.

Insert LAD/FBD source file

To insert a new LAD/FBD unit using the context menu:

1. Select the **PROGRAMS** folder of the relevant SIMOTION device in the project navigator.
2. Double-click the entry **Insert LAD/FBD unit**.

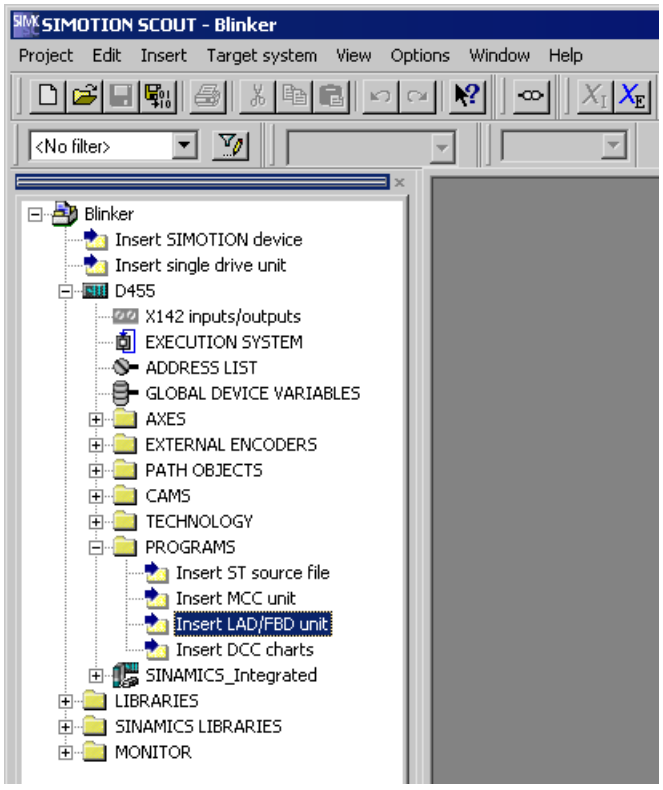


Figure 7-522 Project folder

The **Insert LAD/FBD Unit** dialog box appears.

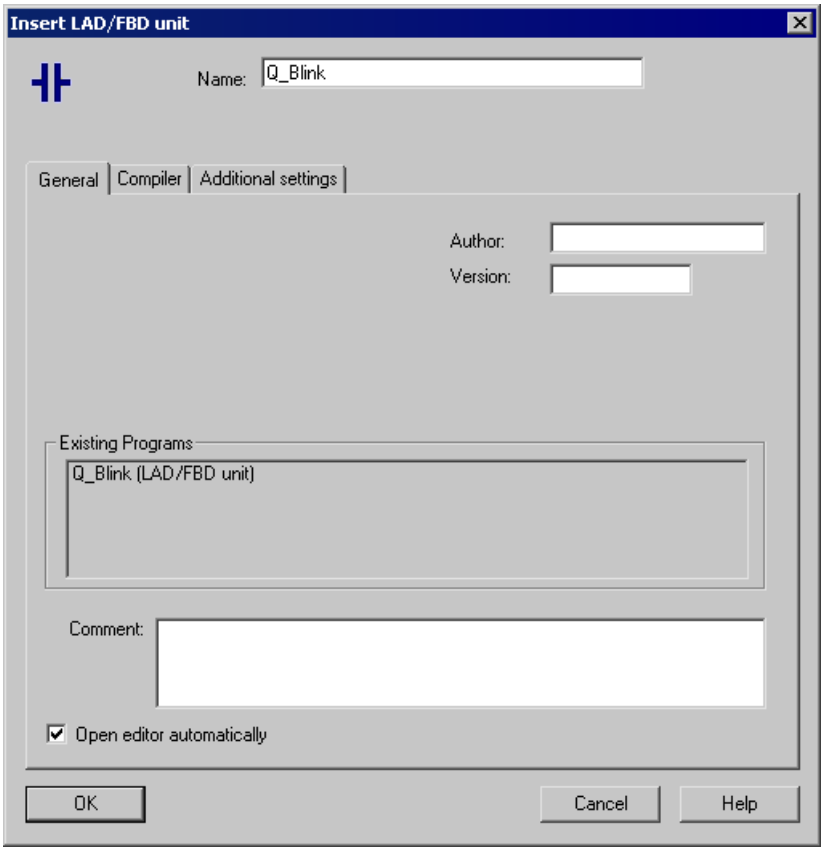


Figure 7-523 Insert LAD/FBD Unit dialog box

3. Enter the name of the LAD/FBD unit.
   The names of program sources must comply with the rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters.
   The permissible length of the name depends on the SIMOTION Kernel version:
   - SIMOTION Kernel as of version V4.1: Maximum 128 characters.
   - SIMOTION Kernel up to version V4.0: Maximum 8 characters.

   Names must be unique within the SIMOTION device. Protected or reserved identifiers (Page 5410) are not permitted.
   Existing program sources (e.g. LAD/FBD units, ST source files) are displayed.
4. Activate the **Permit program status** compiler option to be able to use the online status display later:
   - Deactivate the associated checkbox in the "Global settings" column.
   - Activate the associated checkbox in the "Current settings" column.
5. In the **Compiler** tab, activate the **Permit program status** checkbox
6. You can also enter an author, version, and a comment.

7. Activate the **Open editor automatically** checkbox.
8. Confirm with **OK**.
   The declaration tables for exported and source-internal variables appear in the working area.
   No variables are defined here in the sample program.

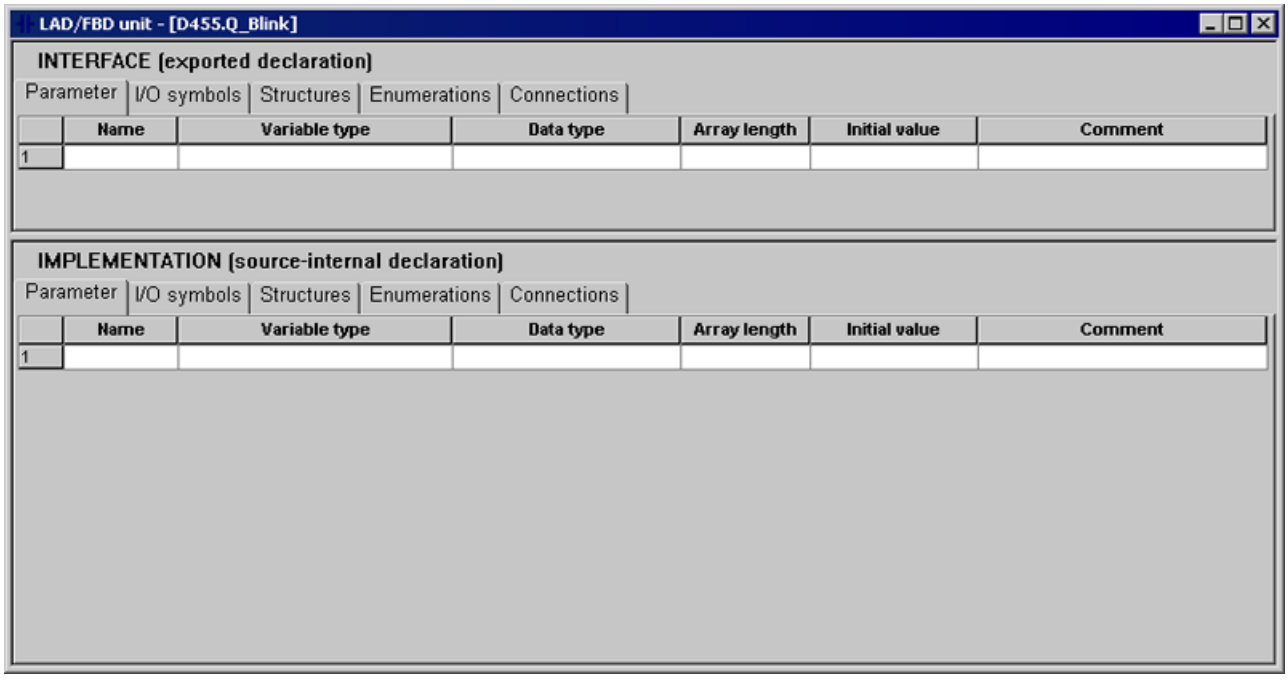


Figure 7-524 Declaration tables for exported and source-internal declarations

Insert LAD/FBD program

To insert an LAD/FBD program, proceed as follows:

1. In the **PROGRAMS** folder in the project navigator, open the LAD/FBD unit you just inserted.
2. Double-click the entry **Insert LAD/FBD program** in the LAD/FBD unit.

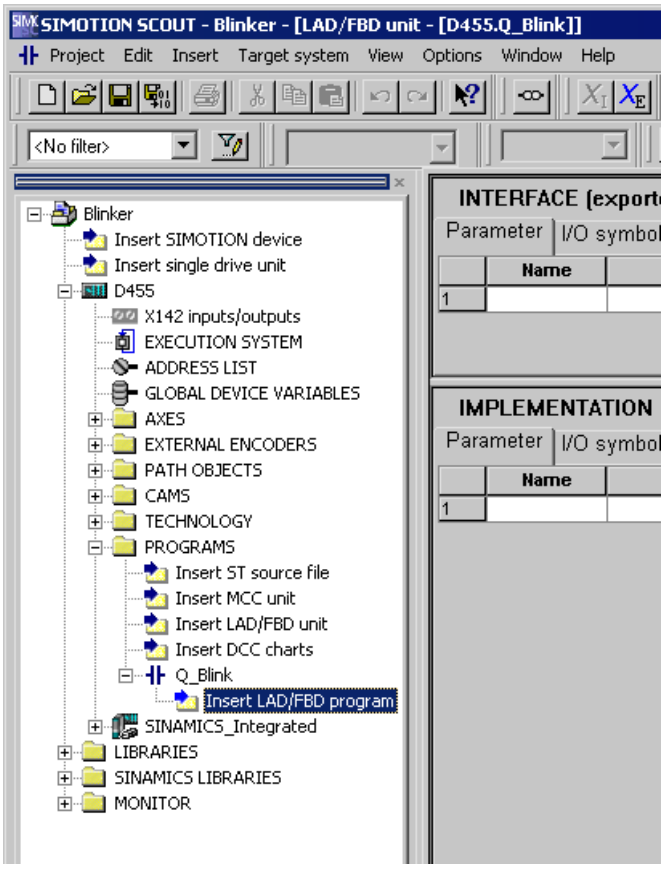


Figure 7-525 Opening a project folder

The **Insert LAD/FBD Program** dialog box appears.

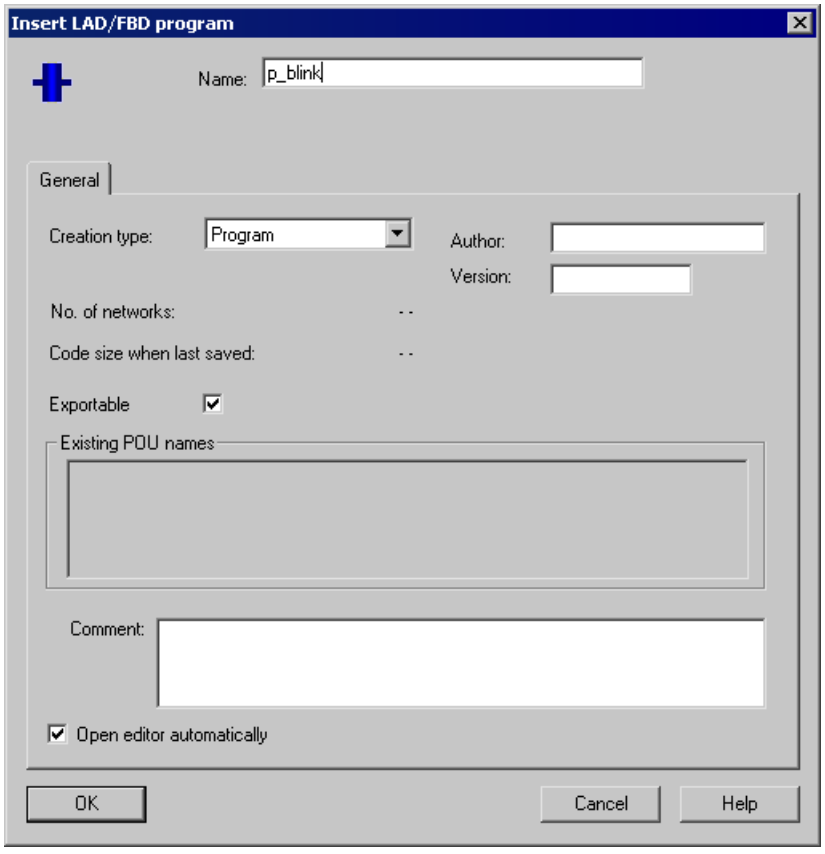


Figure 7-526 Insert LAD/FBD Program dialog box

3. Enter the name of the program in the **Insert LAD/FBD Program** dialog box.
   Names for LAD/FBD programs must comply with the Rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters. Protected or reserved identifiers (Page 5410) are not permitted.
   The permissible length of the name is 25 characters.
   The names must be unique within the LAD/FBD unit. The names of all exportable program organization units (POUs) must also be unique within the SIMOTION device. The names of all LAD/FBD programs of the program source as well as the names of all exportable POUs of the device are displayed.
4. For **Creation type**, select program.
5. Activate the **Exportable** checkbox that must be available for the LAD/FBD program in the execution system.
6. Activate the **Open editor automatically** checkbox.
7. Confirm with **OK**.

A blank LAD/FBD program is opened.

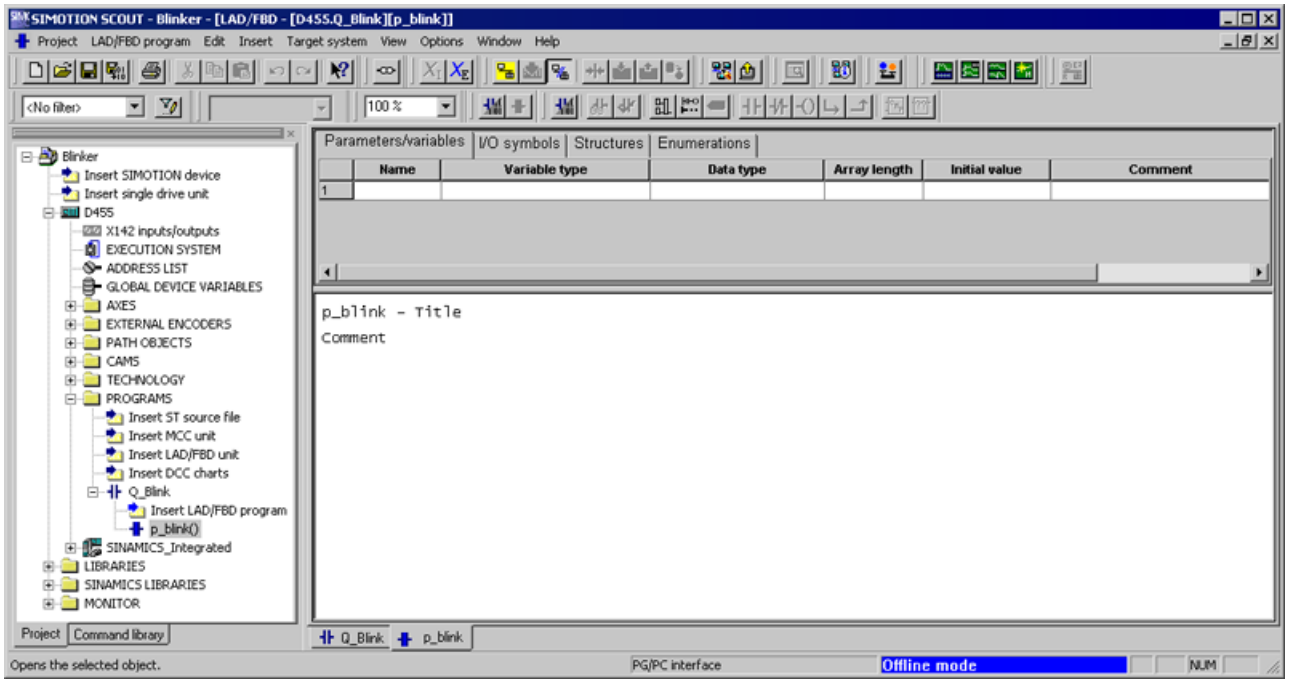


Figure 7-527 Open LAD/FBD program

Entering variables in the declaration table

To enter variables, proceed as follows:

1. Select the **Parameters/variables** tab.
2. Enter name, variable type, data type and/or start value in the declaration table (as shown in the figure below).

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	einDINT	VAR	DINT		100	
2	i_1	VAR	DINT		1	
3	outUSint	VAR	USINT			
4	on	VAR	BOOL			
5	gl_countdint	VAR	DINT		100	
6						

Figure 7-528 Variables in the declaration table

3. Select the **I/O Symbols** tab.
4. Enter the name and absolute identifier (in accordance with the figure below).
   The **data type** is entered automatically.

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Absolute identifier	Data type	Comment
1	io_var	%QB4	BYTE	
2				

Figure 7-529 I/O symbols in the declaration table

Entering a program title

To enter a program title, proceed as follows:

1. Click in the title line.
2. Enter the program name in the window.

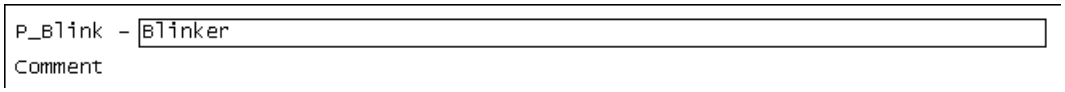


Figure 7-530 Program title

Inserting network

To paste in a network:

1. Select **LAD/FBD program > Insert network**.

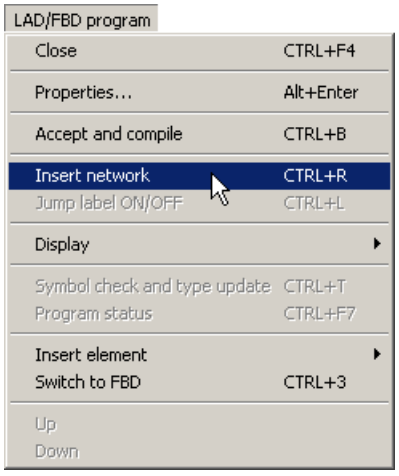


Figure 7-531 Menu selection

2. Click in the title line.
3. Enter the network name in the window.

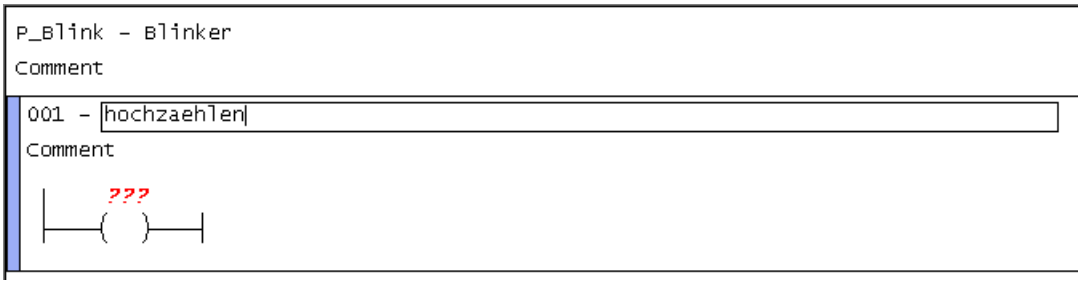


Figure 7-532 Network with entered name

4. Click the power rail to the left of the coil.

Inserting an empty box

To insert an empty box, proceed as follows:

1. From the menu, select the **LAD/FBD program > Insert element > Empty box** menu item.

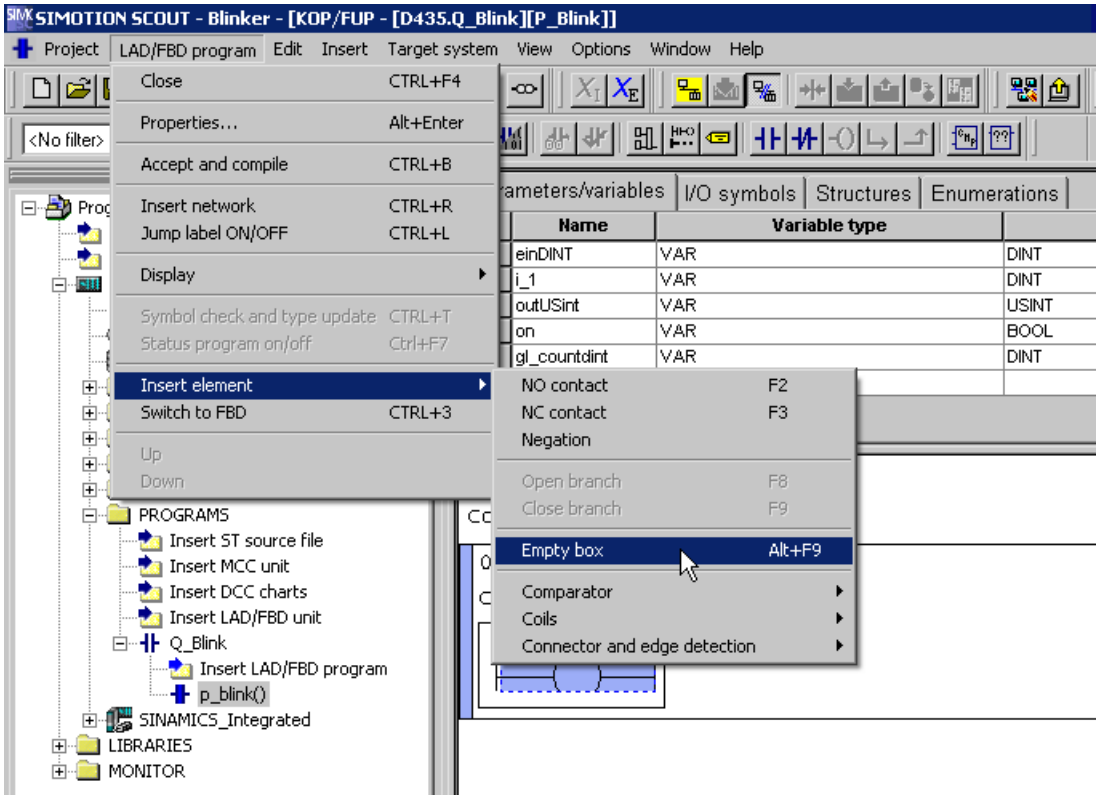


Figure 7-533 Insert an empty box

An empty box is inserted.

Mandatory parameters in a network are identified by **???**, optional parameters by **...** .

Selecting box type

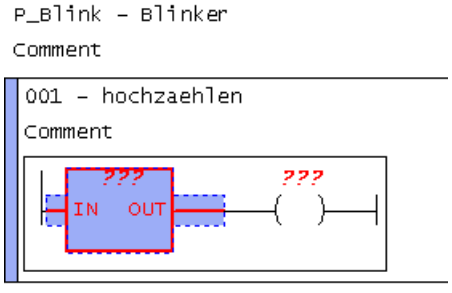


Figure 7-534 Empty box

To select a box type, proceed as follows:

1. Press the **Enter key** in the selected empty box.
   A drop-down menu appears.
2. Select the **ADD** box type from the drop-down menu and confirm your selection by pressing the **Enter key**.

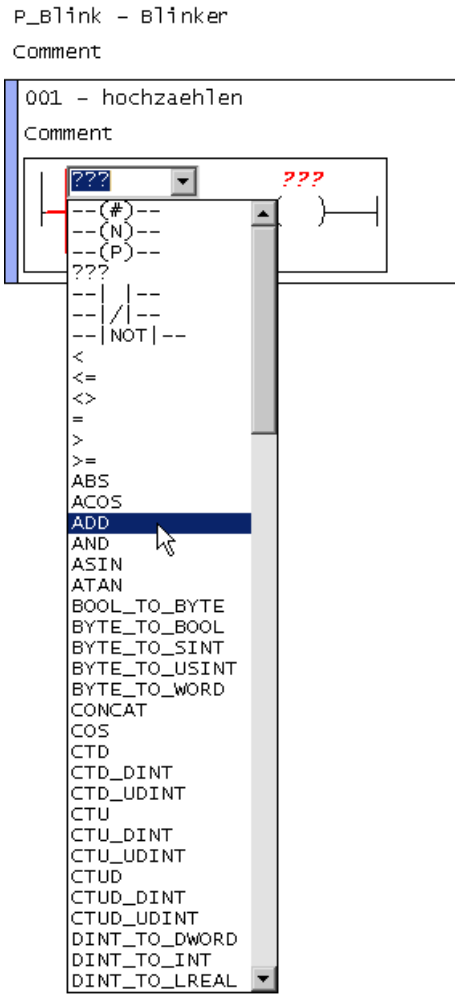


Figure 7-535 Selection of box type

Parameterizing the ADD call-up

To parameterize the ADD call-up, proceed as follows:

1. Click in the other mandatory input fields **???**.

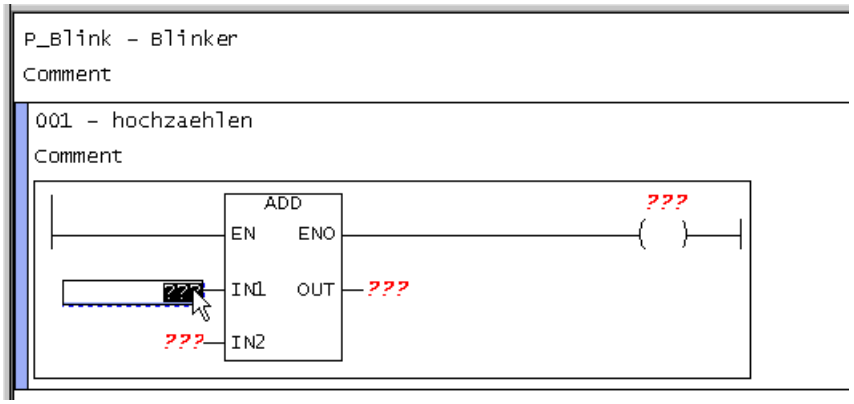


Figure 7-536 ADD box

2. Enter the appropriate values.

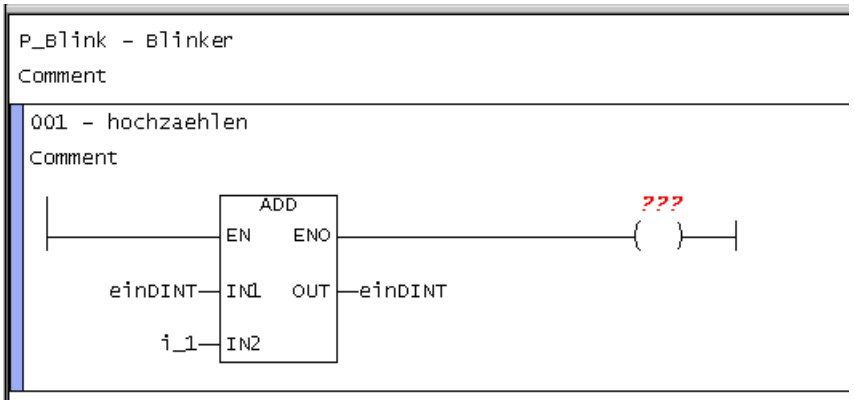


Figure 7-537 Parameterized ADD box

Inserting comparator

To insert a comparator, proceed as follows:

1. Select the ADD box.

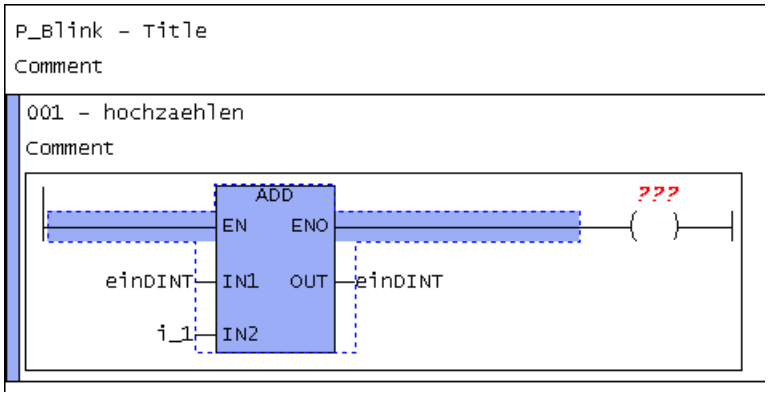


Figure 7-538 Selected ADD box

2. Select **LAD/FBD program > Insert element >Comparator > > =.**

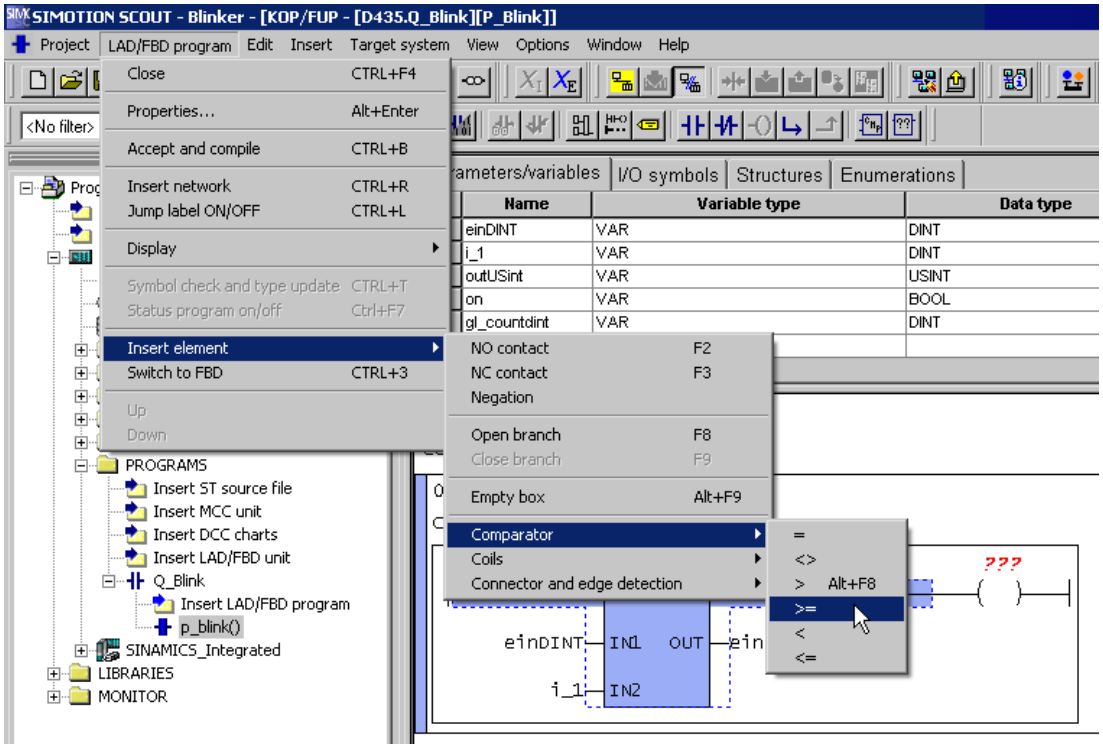


Figure 7-539 Select comparator

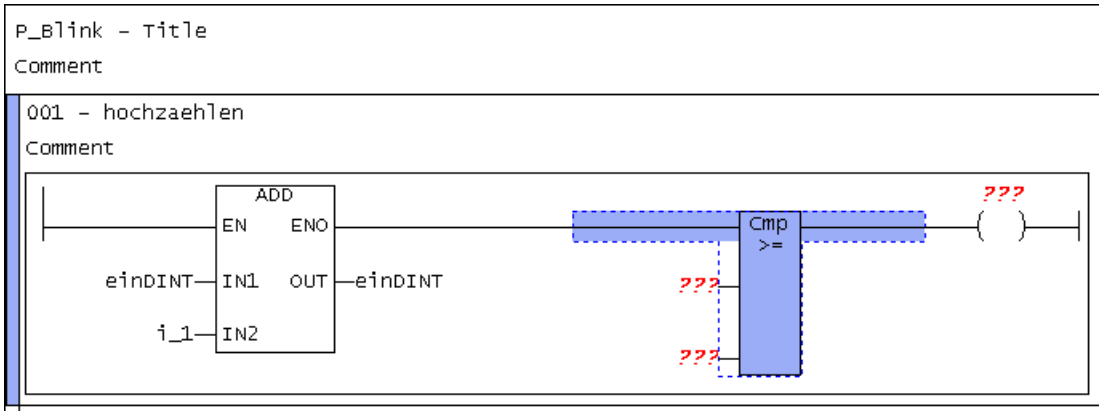


Figure 7-540 Inserted comparator

Labeling the comparator

To label the comparator, proceed as follows:

1. Click each comparator input field individually.
2. Enter the appropriate values for the comparator.

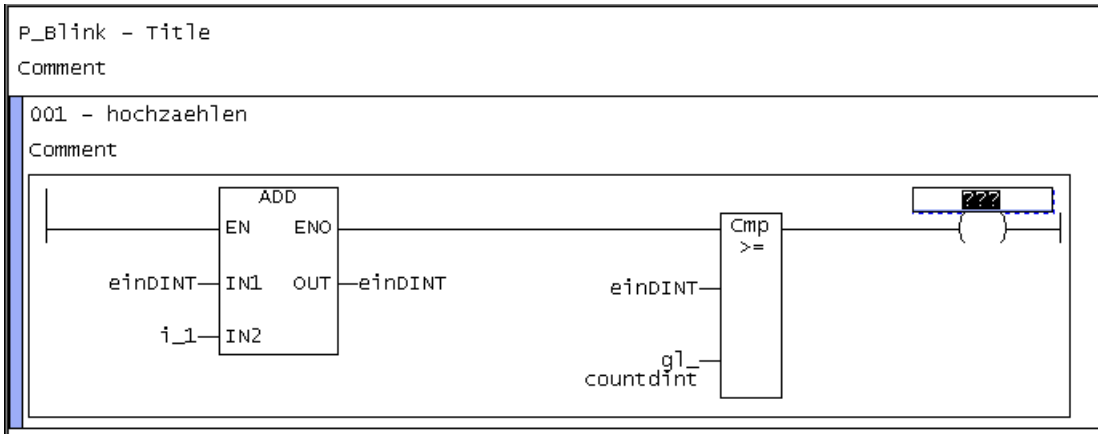


Figure 7-541 Labeled comparator

Initializing a coil

To initialize a coil, proceed as follows:

1. Click in the input field **???** of the coil.
2. Enter the appropriate variable.

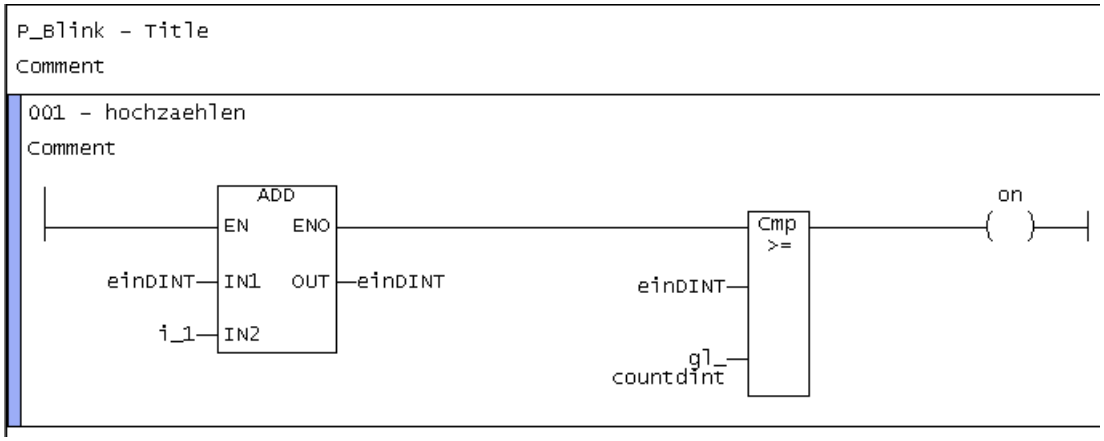


Figure 7-542 Initialized coil

Inserting next network

To paste in another network, proceed as follows:

1. To paste in the second network, repeat the steps used to paste in the first network.

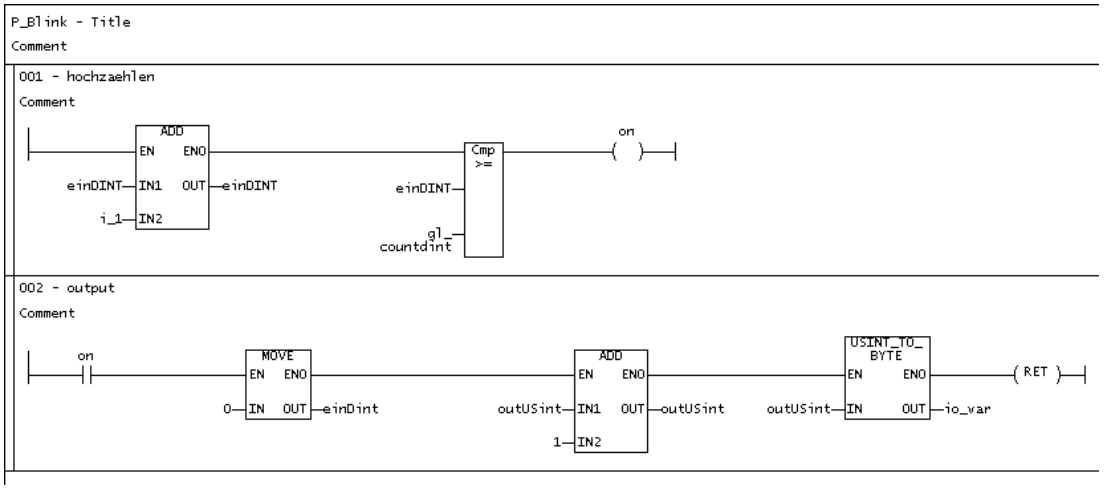


Figure 7-543 Project with two networks

Details view

To show the detail view, proceed as follows:

1. Select the **View > Detail view** menu command.
   Information, e.g. compiler messages, will be displayed during the compilation of a program.

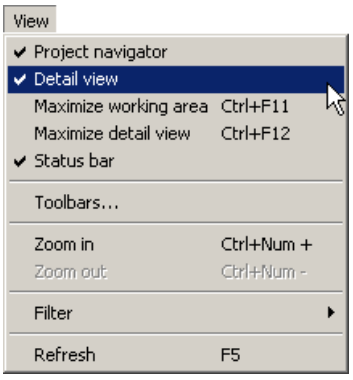


Figure 7-544 Detail view menu selection

Compiling

To compile the created program, proceed as follows:

1. Select the program in the project navigator.
2. Open the **LAD/FBD program** menu and select **Accept and compile**.
   The source file and its POUs are saved and compiled.
   During the compilation process, messages on the successful compilation status are displayed in the detail view. Should any error occur during compilation, they will be displayed in plain text there.

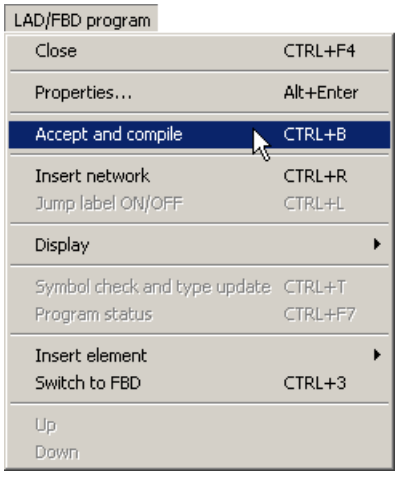


Figure 7-545 Save and compile menu selection

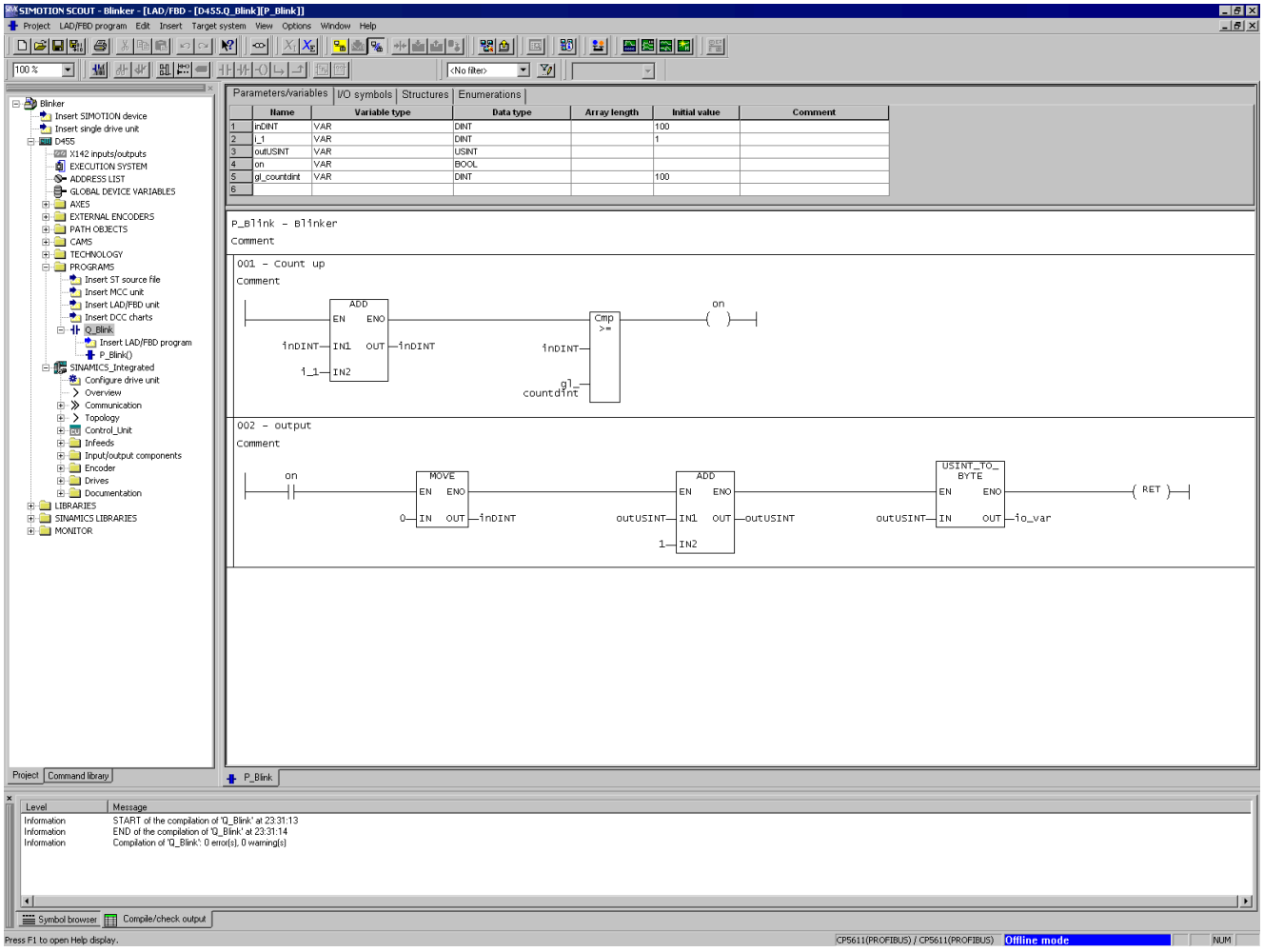


Figure 7-546 Compiled project with compiler information in the detail view

Assigning a sample program to an execution level

To assign a program to an execution level, proceed as follows:

1. Double-click the **EXECUTION SYSTEM** folder in the project navigator.
2. Click **BackgroundTask**.
3. Click the **Program assignment** tab.
4. Select the program **Q_blink.p_blink**.

5. Click the button **>>**.

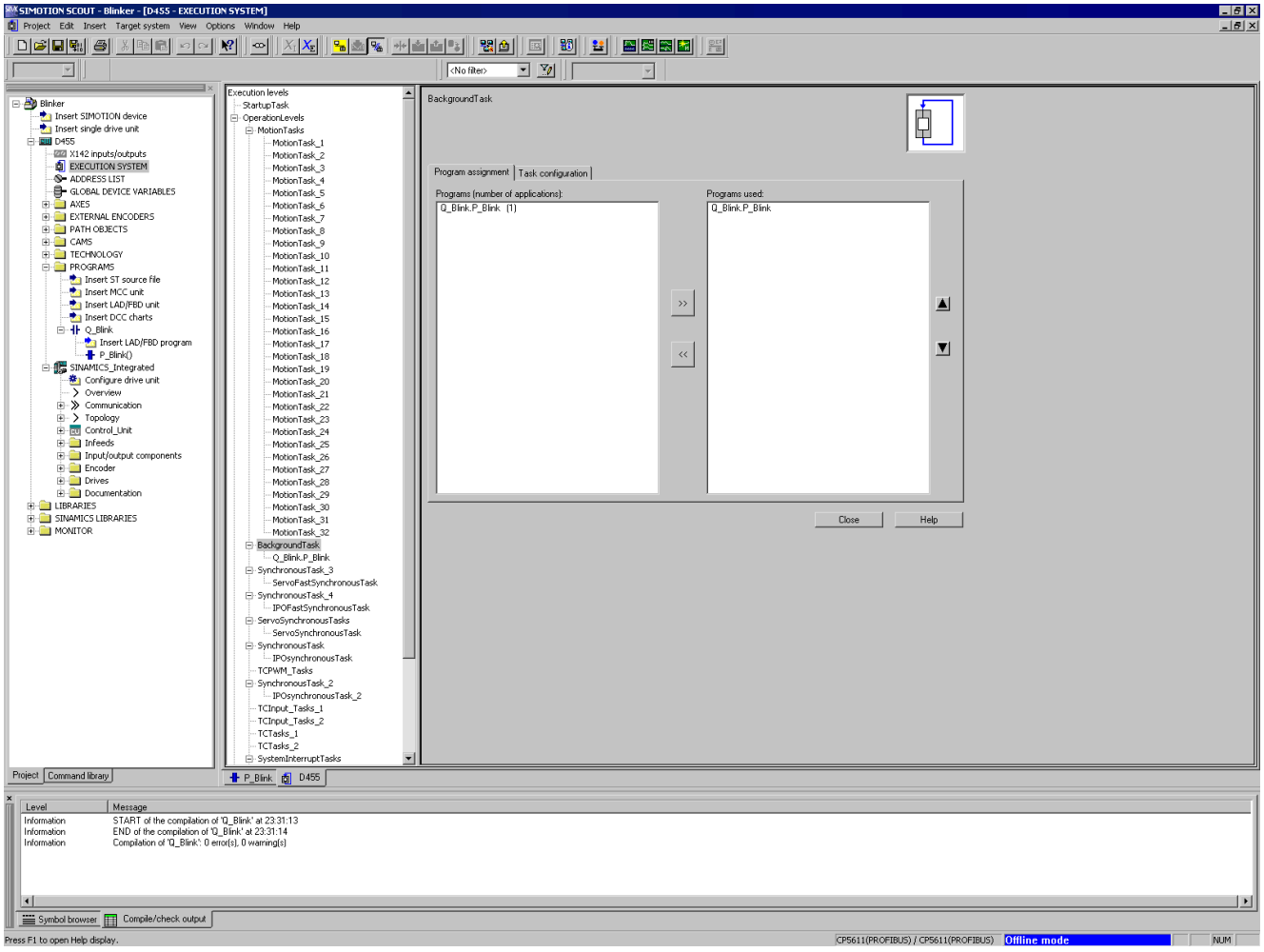


Figure 7-547 Assigning a program to the BackgroundTask

6. Click **Close** and acknowledge the message saying the execution system has changed by clicking **Yes**.
   The changes are accepted into the project.

Starting sample program

To start a program, proceed as follows:

1. Make sure the LAD/FBD unit creates the additional debug code for **program status** during compilation:
   Open the Properties window for the LAD/FBD unit (see Defining the properties of an LAD/FBD unit (Page 5109)).
2. Activate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (see Local compiler settings (Page 5112)) for this LAD/FBD unit.
   See also the description relating to Effectiveness of local or global compiler settings (see the SIMOTION ST Programming and Operating Manual).

3. Select **Project > Save and recompile all**.
   The project is locally saved on the hard disk and compiled.
4. Select the **Project > Connect to selected target devices** menu command or click .
   Online mode is activated.
5. Select the **Target system > Load > Download project to target system** menu command or click .
   The project data (including the sample program) and the data of the hardware configuration are downloaded to the RAM of the target system.
6. Select both networks and click the button for **program status** (Ctrl+F7 shortcut) in the LAD editor function bar (Page 5086).
   Monitoring the program execution (Page 5348) is switched on.
7. Mark the SIMOTION device in the project navigator and select **Target device > Operating mode** in the context menu.
   The **Operating mode** window with the software switch for modes opens.
8. Click the **RUN** button in the software switch.
   The SIMOTION device is in **RUN** mode. The sample program is run and the current paths/signal paths are color-coded in accordance with the current signal values (Page 5348).

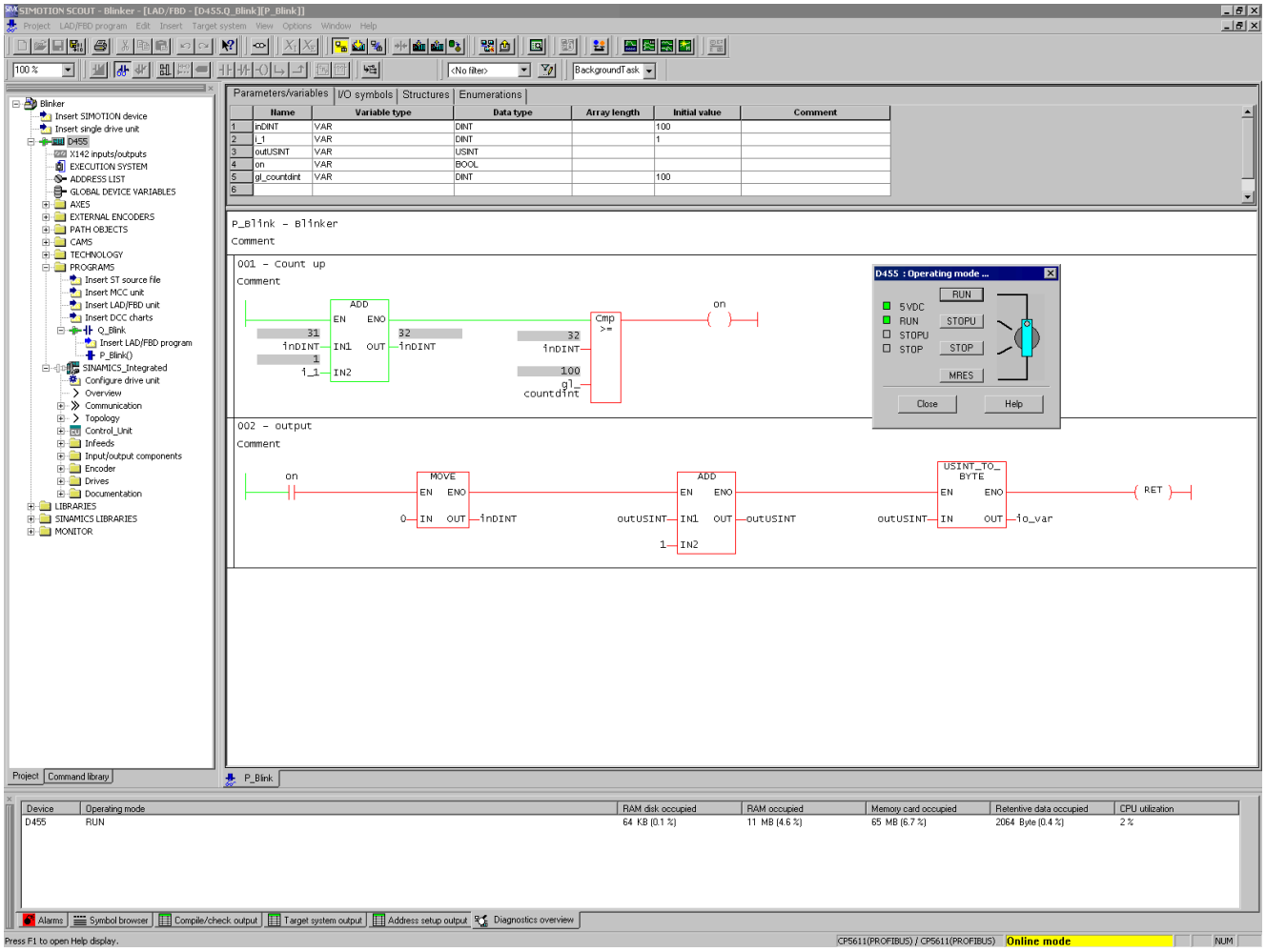


Figure 7-548 Sample program is started

7.3.8.4 Position axis program

Requirements

A project must be created. In the project, a CPU and a virtual position axis must be created (name of the axis: posAxis).

Task specification

An axis is to be traversed at a velocity of 10 mm/s from the current position 100 mm in the negative direction.

This task is divided into the following parts:

- Insert LAD/FBD unit
- Insert LAD/FBD program
  - Insert network
  - Set axis enable signals
  - Traverse axis to position
  - Remove axis enable
- Compile program
- Insert program in a task
- Download program onto target device

PLCopen blocks are used for the programming. The PLCopen blocks are designed for use in cyclic programs/tasks and enable motion control programming in a PLC environment. They are used primarily in the LAD/FBD programming language.

PLCopen blocks are available as standard functions (directly from the command library).

You can find further information about PLCopen blocks in the SIMOTION PLCopen Blocks Function Manual.

There is a TO-specific command available for the aforementioned subtasks **Set axis enable** and **Traverse axis to position/Remove axis enable**. Each command is represented by a box in LAD/FBD. The parameters for individual commands (position = 100, speed = 10 etc.) are entered

via the **Variable declaration** dialog box.

- or -

via the **Enter call parameters** dialog box

- or -

by entering the values in the input fields on each connector.

The task is implemented using LAD programming.

Insert LAD/FBD source file

To insert an LAD/FBD unit (for details of how to insert the unit and program, see also the blinker program (Page 5370) example), proceed as follows:

1. Open the **PROGRAMS** folder of the relevant SIMOTION device in the project navigator.
2. Double-click the entry **Insert LAD/FBD unit**.
   The **Insert LAD/FBD Unit** dialog box appears.
3. Enter the name of the LAD/FBD unit.
   The names of program sources must comply with the rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters.
   The permissible length of the name depends on the SIMOTION Kernel version:
   - SIMOTION Kernel as of version V4.1: Maximum 128 characters.
   - SIMOTION Kernel up to version V4.0: Maximum 8 characters.

   Names must be unique within the SIMOTION device. Protected or reserved identifiers (Page 5410) are not permitted.
   Existing program sources (e.g. ST source files, MCC units) are displayed.
4. In the **Compiler** tab, activate the **Permit program status** checkbox, to use the online status display later.
5. You can also enter an author, version, and a comment.
6. Activate the **Open editor automatically** checkbox.
7. Confirm with **OK**.
   The declaration tables for global and unit-local variables appear in the working area.

Insert LAD/FBD program

To insert an LAD/FBD program, proceed as follows:

1. In the **PROGRAMS** folder within the project navigator, open the LAD/FBD unit you just pasted in.
2. Double-click the entry **Insert LAD/FBD program** in the LAD/FBD unit.
   The **Insert LAD/FBD Program** dialog box appears.
3. Enter the name of the program in the **Insert LAD/FBD Program** dialog box.
   Names for LAD/FBD programs must comply with the Rules for identifiers (Page 5155): They are made up of letters (A ... Z, a ... z), numbers (0 ... 9), or single underscores (_) in any order, whereby the first character must be a letter or underscore. No distinction is made between upper- and lower-case letters. Protected or reserved identifiers (Page 5410) are not permitted.
   The permissible length of the name is 25 characters.
   The names must be unique within the LAD/FBD unit. The names of all exportable program organization units (POUs) must also be unique within the SIMOTION device. The names of all LAD/FBD programs of the program source as well as the names of all exportable POUs of the device are displayed.
4. For **Creation type**, select program.
5. Activate the **Open editor automatically** checkbox.

6. Confirm with **OK**.
   A blank LAD/FBD program is opened.
7. Click the working area and select **LAD/FBD program > Insert network** from the menu to paste in a new network.

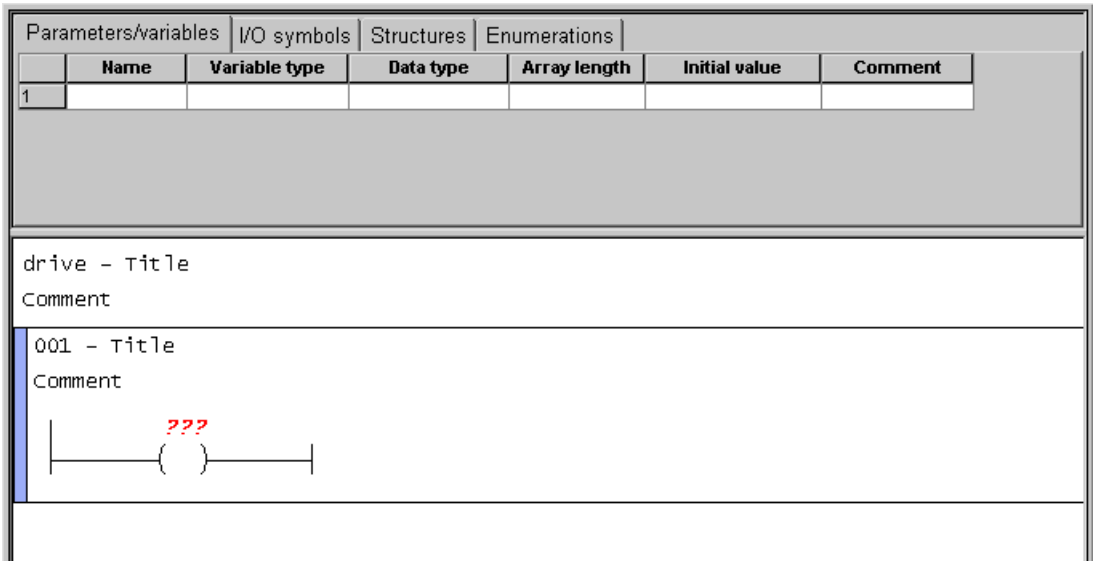


Figure 7-549 Pasted network

Note

Mandatory parameters in a network are identified by **???**, optional parameters by **...** .

8. Select the pasted box and select **Delete** in the context menu.
   The box is removed from the network.

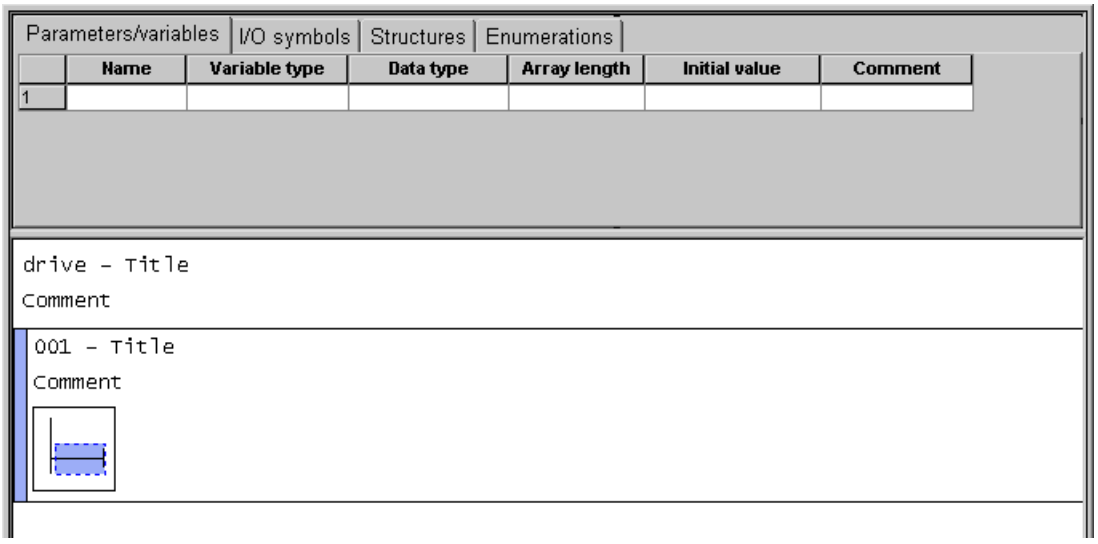


Figure 7-550 Pasted network without a box

Inserting a TO-specific command

To insert a **TO-specific command**, proceed as follows:

1. To enhance the display in the working area, open the shortcut menu of the network and select **Display > Mandatory and assigned box parameters**.
2. Select the **Command library** tab in the project navigator.
   The command groups appear.
3. Click the relevant plus sign to open the **PLCopen > SingleAxis** command group.

4. Drag and drop the **_MC_Power** command into the network (see Network with RET assignment).
   This command serves to enable the command.

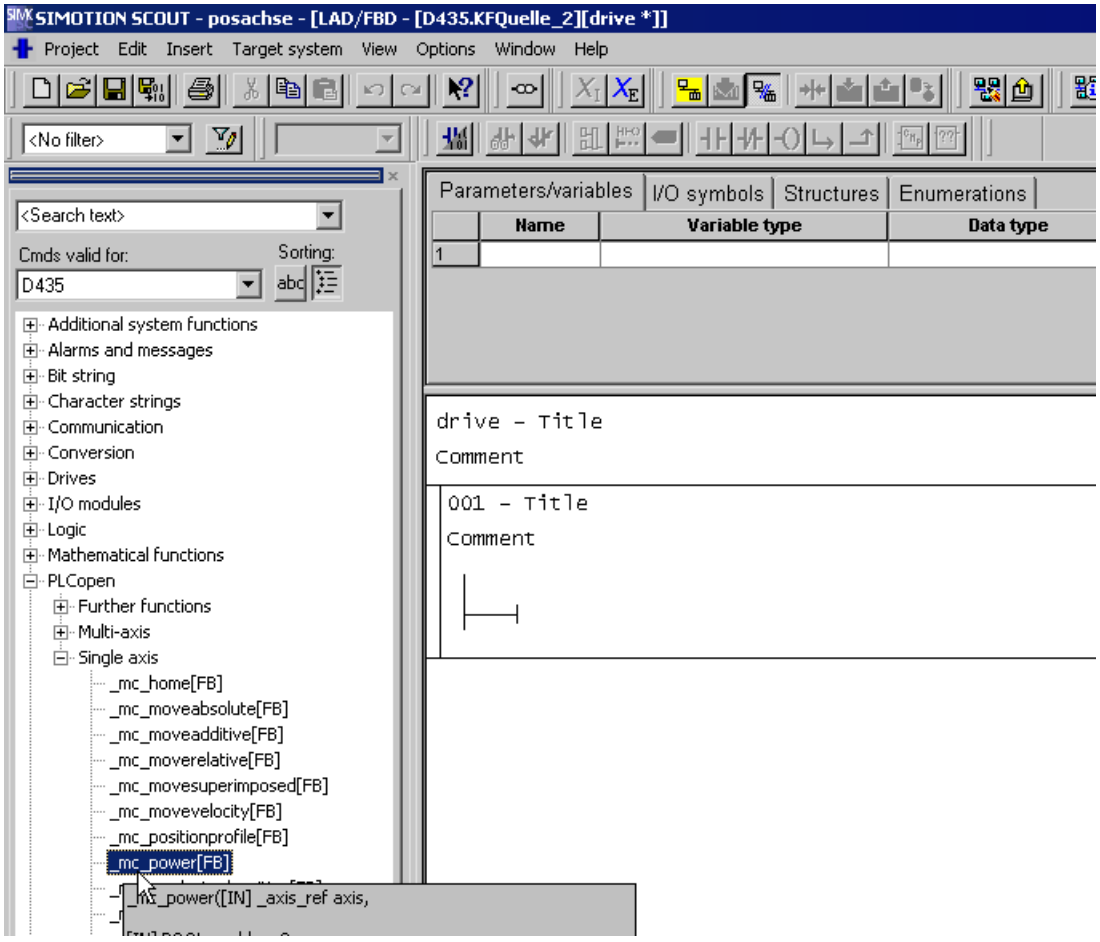


Figure 7-551 TO-specific command (_MC_Power) from the command library

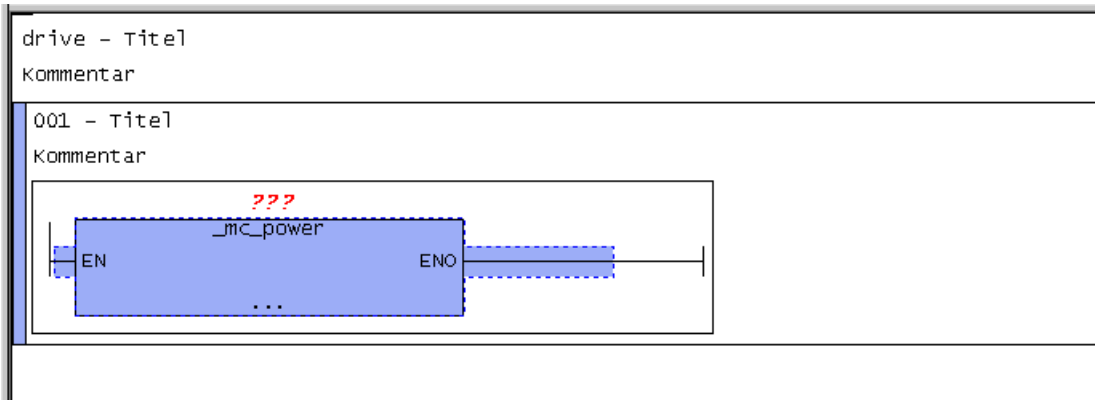


Figure 7-552 Network with inserted _MC_Power Box

5. Mark the network and select **LAD/FBD program > Insert network** from the menu to insert a second network.

6. Mark the inserted box from the second network and select **Delete** in the context menu.
   The box is removed from the network.
7. Drag and drop the **_MC_MoveRelative** command from the command library to the marked position in the second network.
   The axis is positioned at the specified speed with this command.

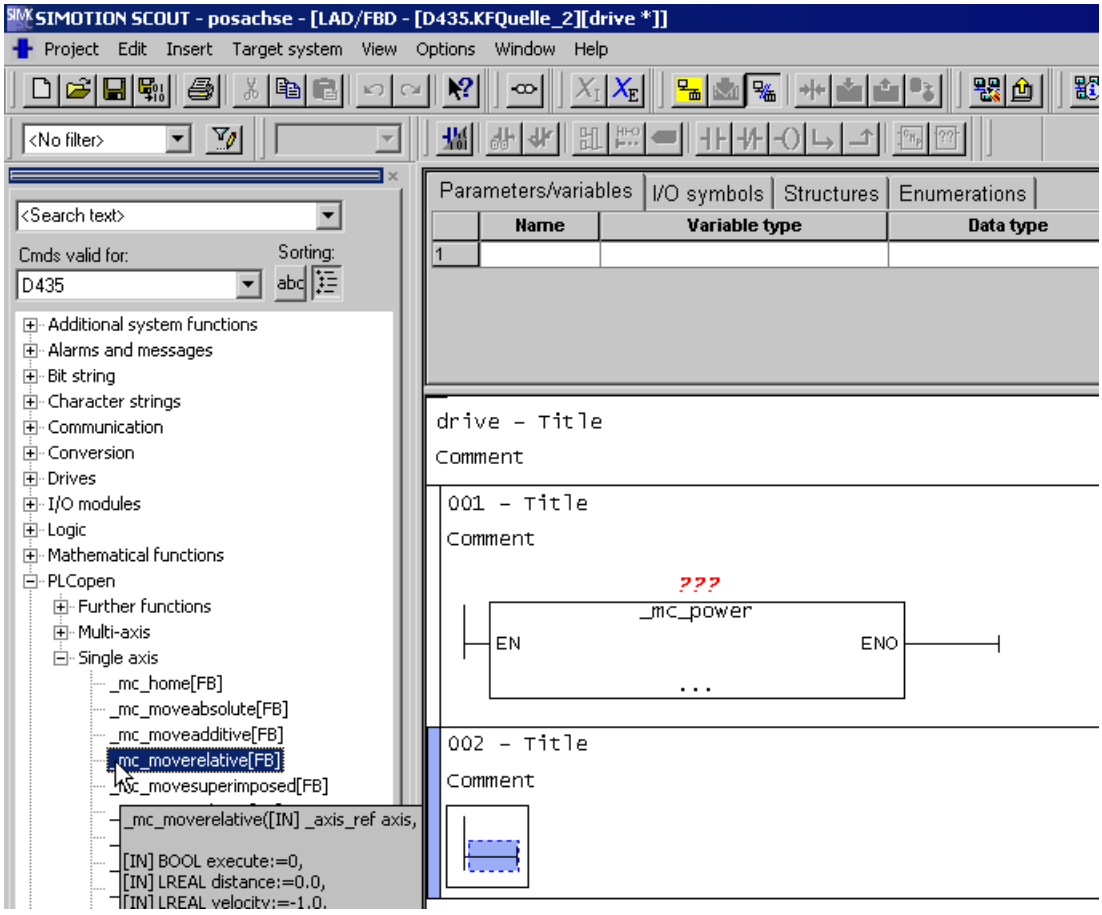


Figure 7-553 TO-specific command (_MC_MoveRelative) from the command library

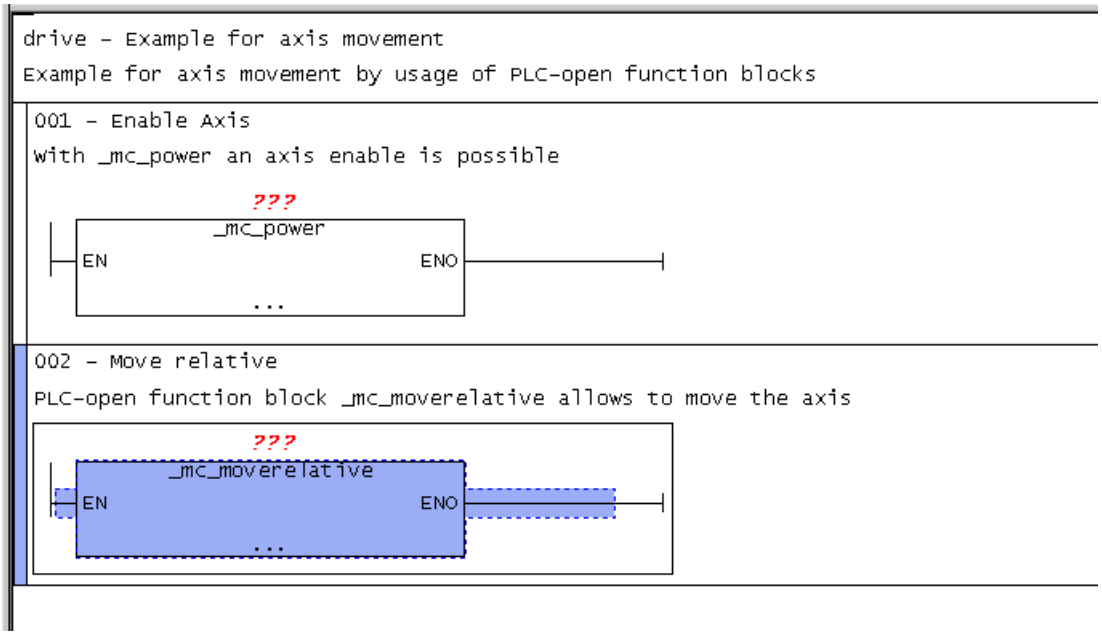


Figure 7-554 Network with inserted _MC_MoveRelative box

Connecting the enable inputs

The **enable** input for the **_MC_Power** command and the **execute** enable input for the **_MC_MoveRelative** command still have to be connected to NO contacts.

How to insert the NO contacts:

1. Click in the working area and select **Display > All box parameters** in the context menu. All the inputs and outputs of the boxes are shown.
2. Select the **Command library** tab in the project navigator. The command groups appear.
3. Click the plus symbol to open the command group **LAD elements**.

4. Drag and drop the **NO contact** LAD element to the **enable** input of the **_MC_Power** function block.

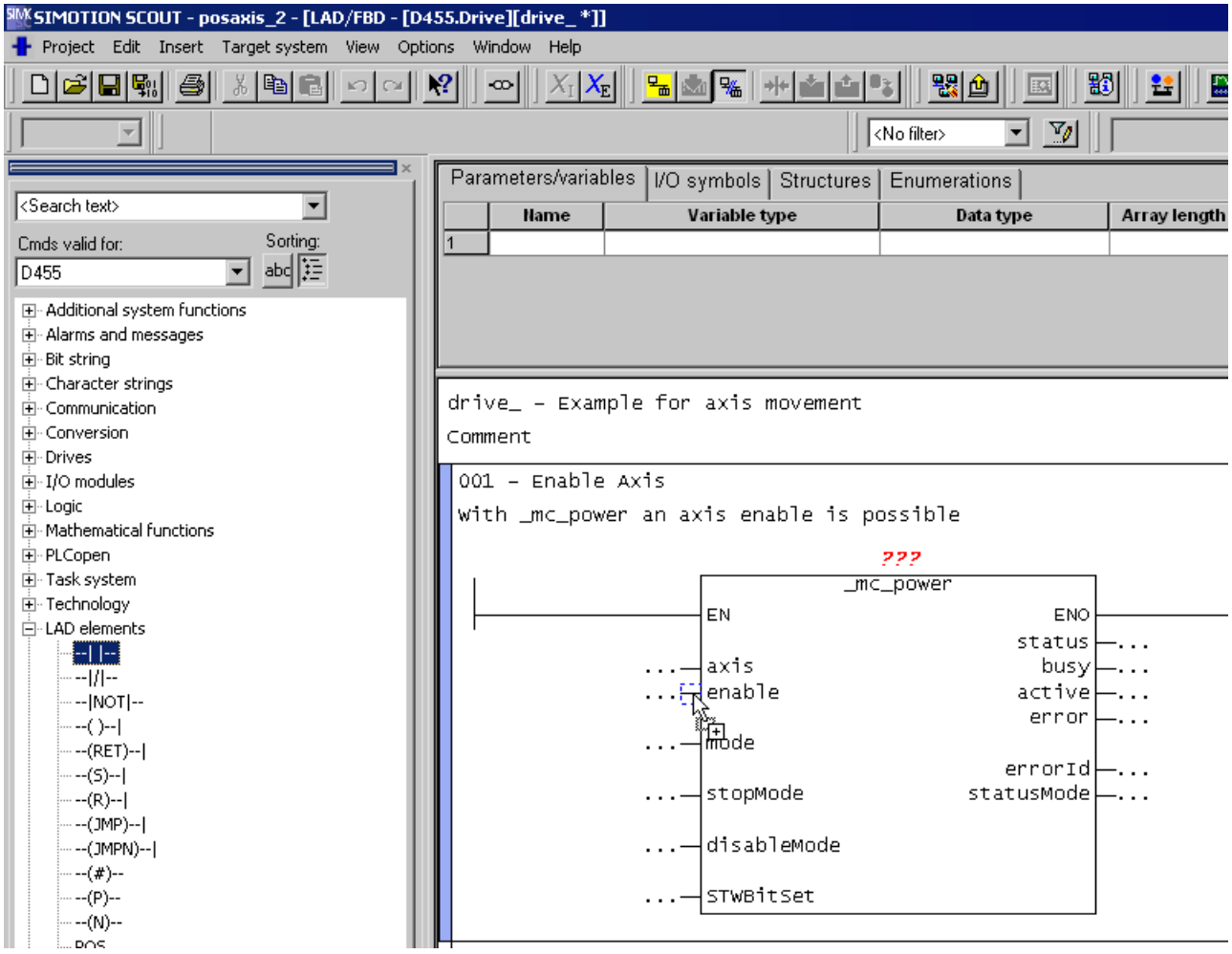


Figure 7-555 Drag&drop the NO contact LAD element to the connector of the enable input

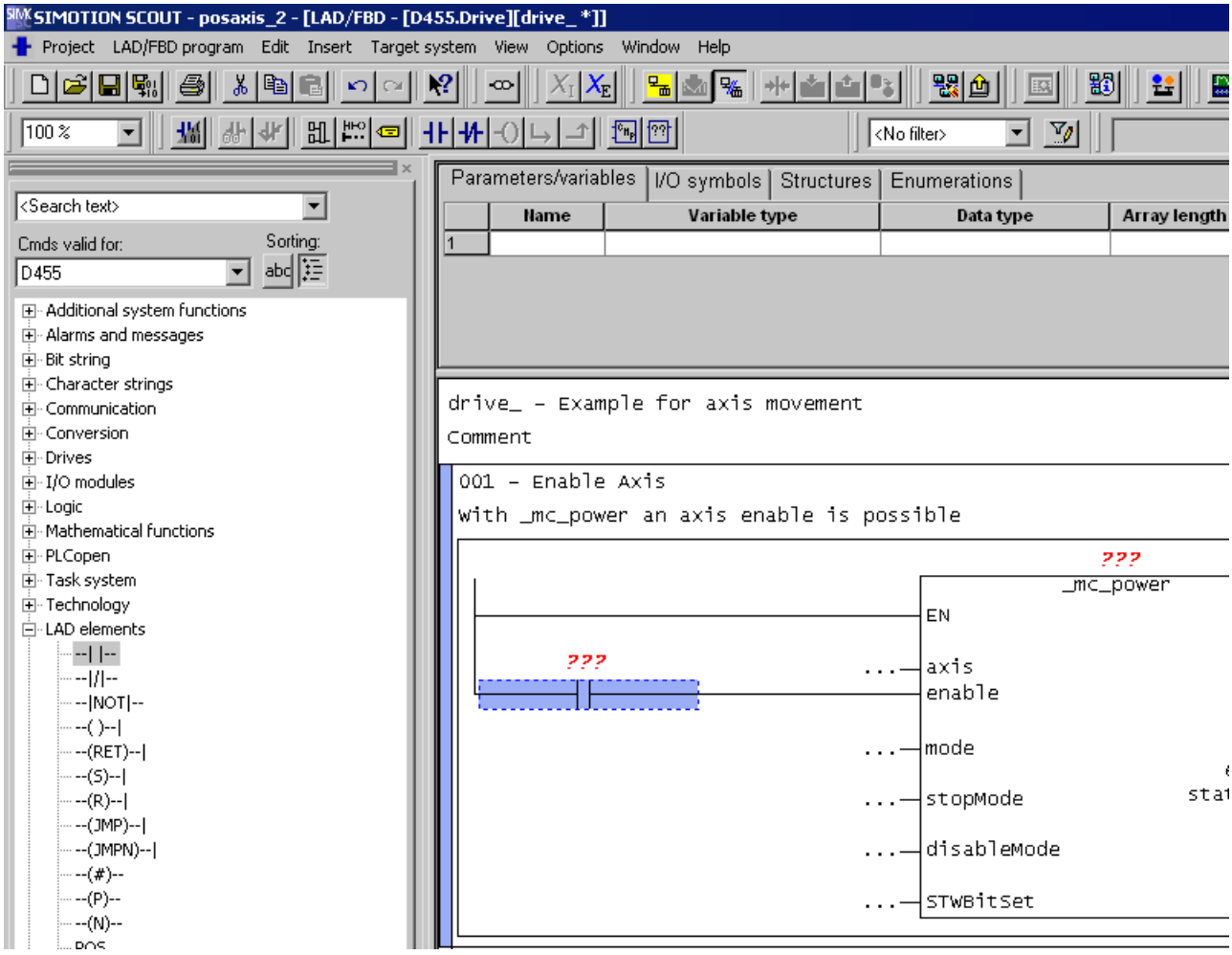


Figure 7-556 Network with a NO contact LAD element inserted

5. Drag and drop two **NO contact** LAD elements to the **execute** enable input of the **_MC_MoveRelative** function block.

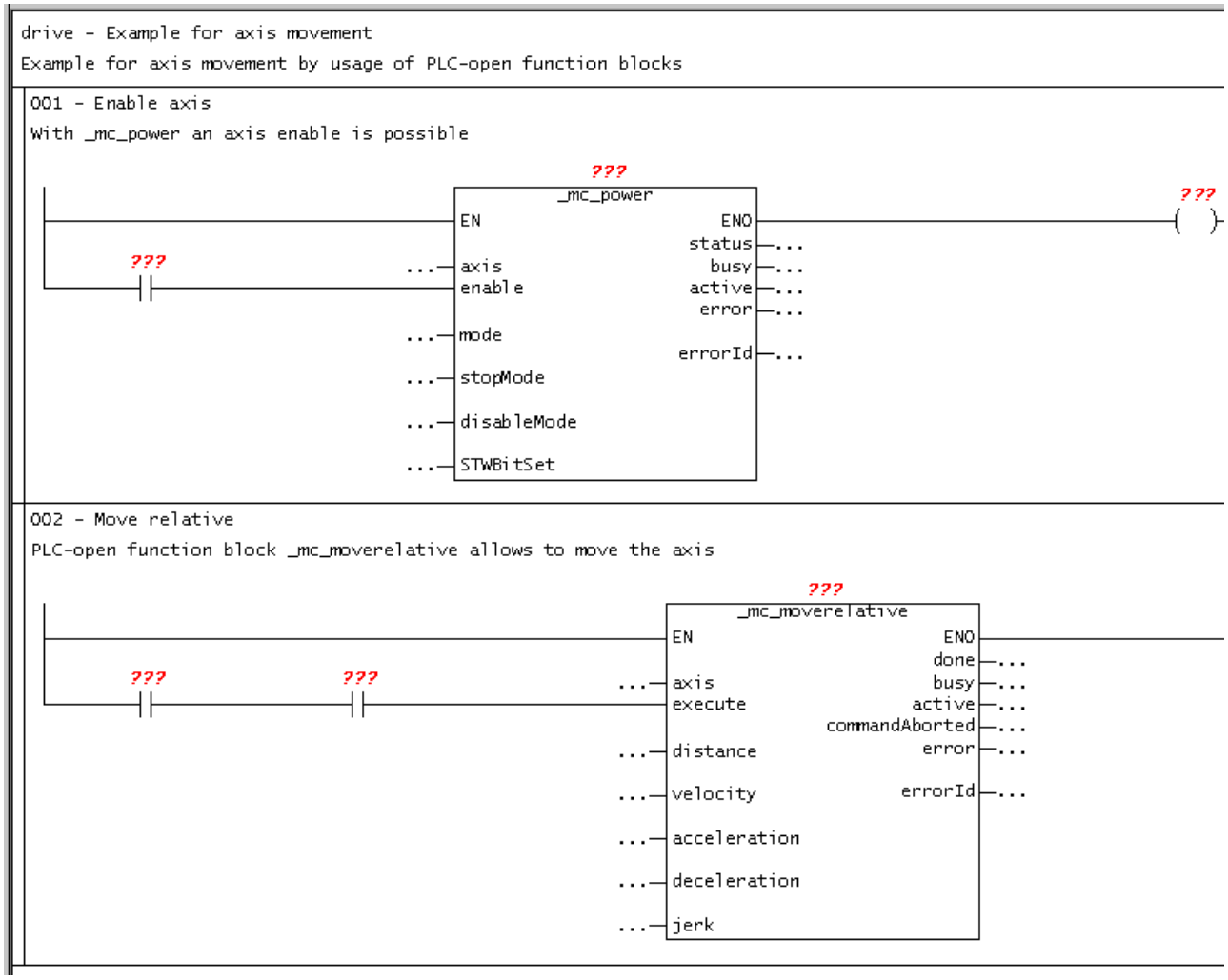


Figure 7-557 Networks with NO contacts inserted

Entering variables in the declaration table

To enter variables, proceed as follows:

1. Select the **Parameters/variables** tab.
2. Enter name, variable type, data type and/or start value in the declaration table (as shown in the figure below).
   In order to use PLCopen blocks, you must create one instance for each the used blocks **(_MC_Power** and **_MC_MoveRelative**). The data type of the instance corresponds to the block name. The variables **i_mc_power** and **i_mc_moverelative** are instance variables for the two function blocks **_MC_Power** and **_MC_MoveRelative**.
   You can find further information about the declaration and use of instance variables in Example: Function block (FB) (Page 5238).

Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	enable	VAR	BOOL			
2	move	VAR	BOOL			
3	i_mc_power	VAR	_MC_POWER			
4	i_mc_moverelative	VAR	_MC_MOVERELATIVE			
5	o_enabled	VAR	BOOL			
6						

Figure 7-558 Variables in the declaration table

Parameterization of the NO contacts

How to parameterize each of the NO contacts:

1. Click in the input field **???** for the NO contact.
2. Enter the appropriate variable.
3. Press **Enter**.

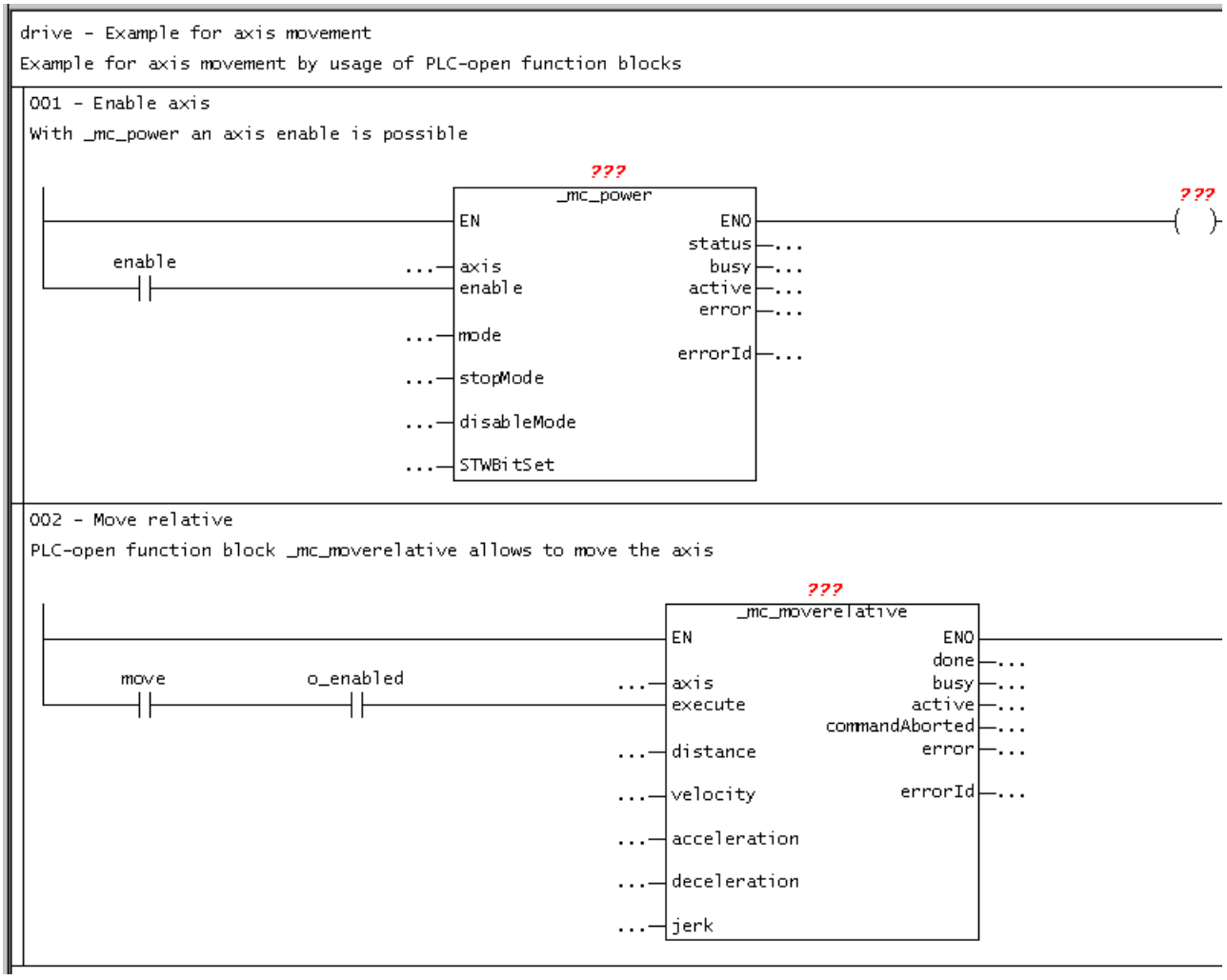


Figure 7-559 Parameterization of the NO contacts

Setting call parameters for the _MC_Power command

Note

The **Enter Call Parameters** dialog box displayed below is available for each SIMOTION command.

To set the call parameters, proceed as follows:

1. Double-click the box.
2. Select the instance, the homing axis, the axis enables to be set, stop mode and enable mode for the axis.
   If you print the project, your parameters appear in the printout according to your settings, e.g. **only allocated box parameters**.
3. Confirm with **OK**.

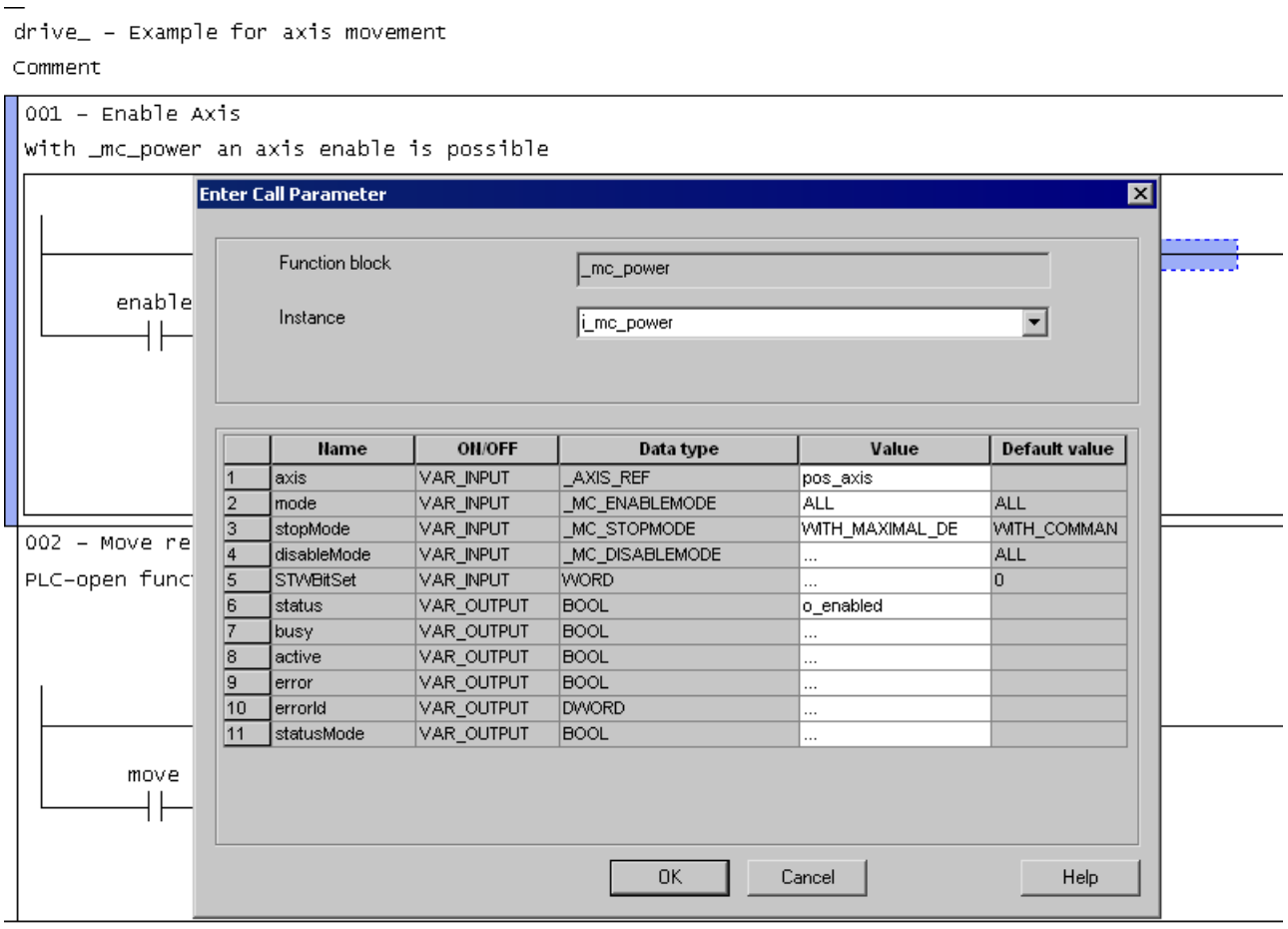


Figure 7-560 Set call parameters for the PLCopen block _MC_Power

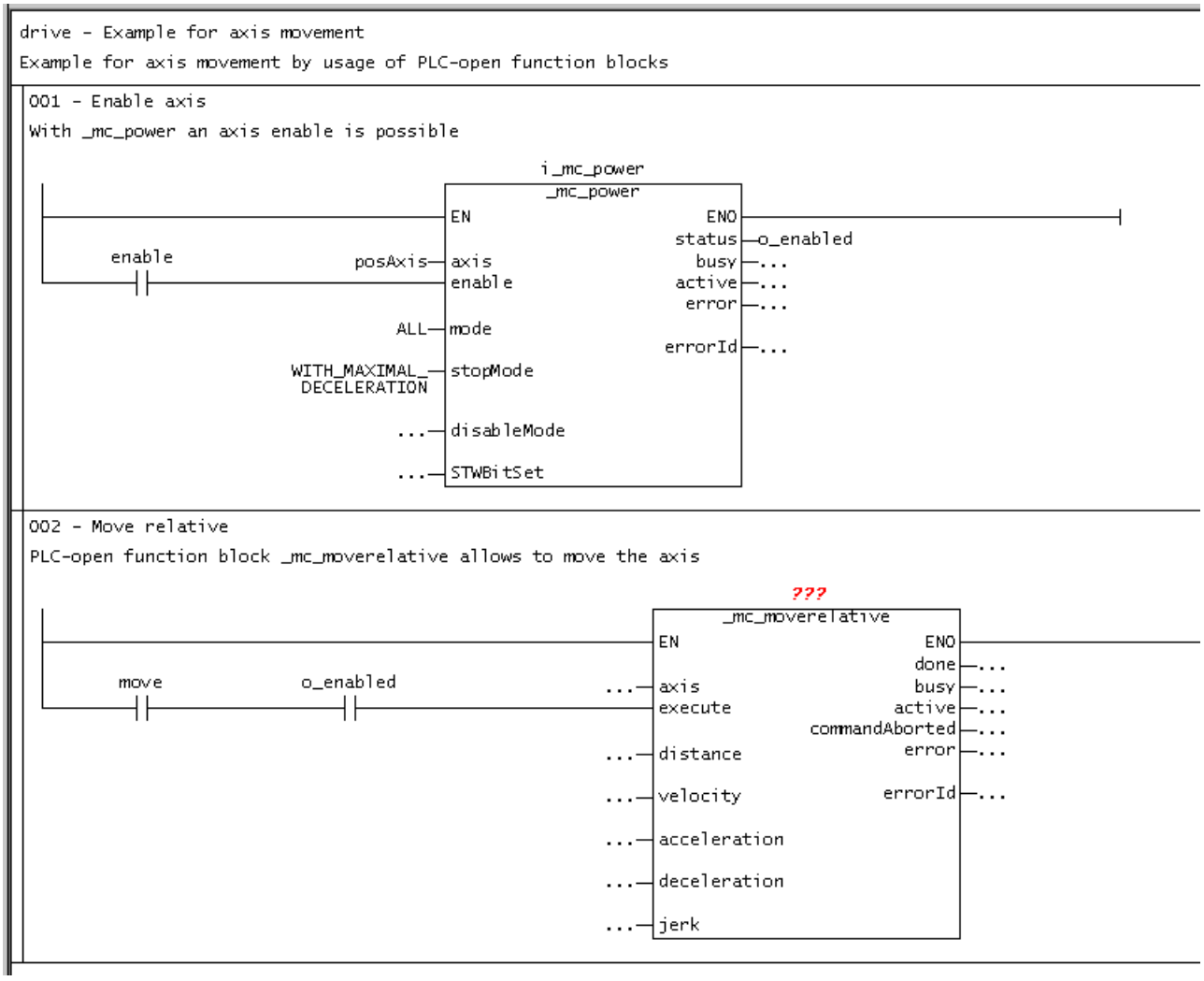


Figure 7-561 Labelled _MC_Power box

Setting call parameters for the _MC_MoveRelative command

To set the call parameters, proceed as follows:

1. Double-click the _mc_moverelative box.
2. Select the instance and the homing axis. Enter values for the difference in distance traveled and for the maximum speed of the axis.
   If you print the project, your parameters appear in the printout according to your settings, e.g. **only allocated box parameters**.
3. Confirm with **OK**.

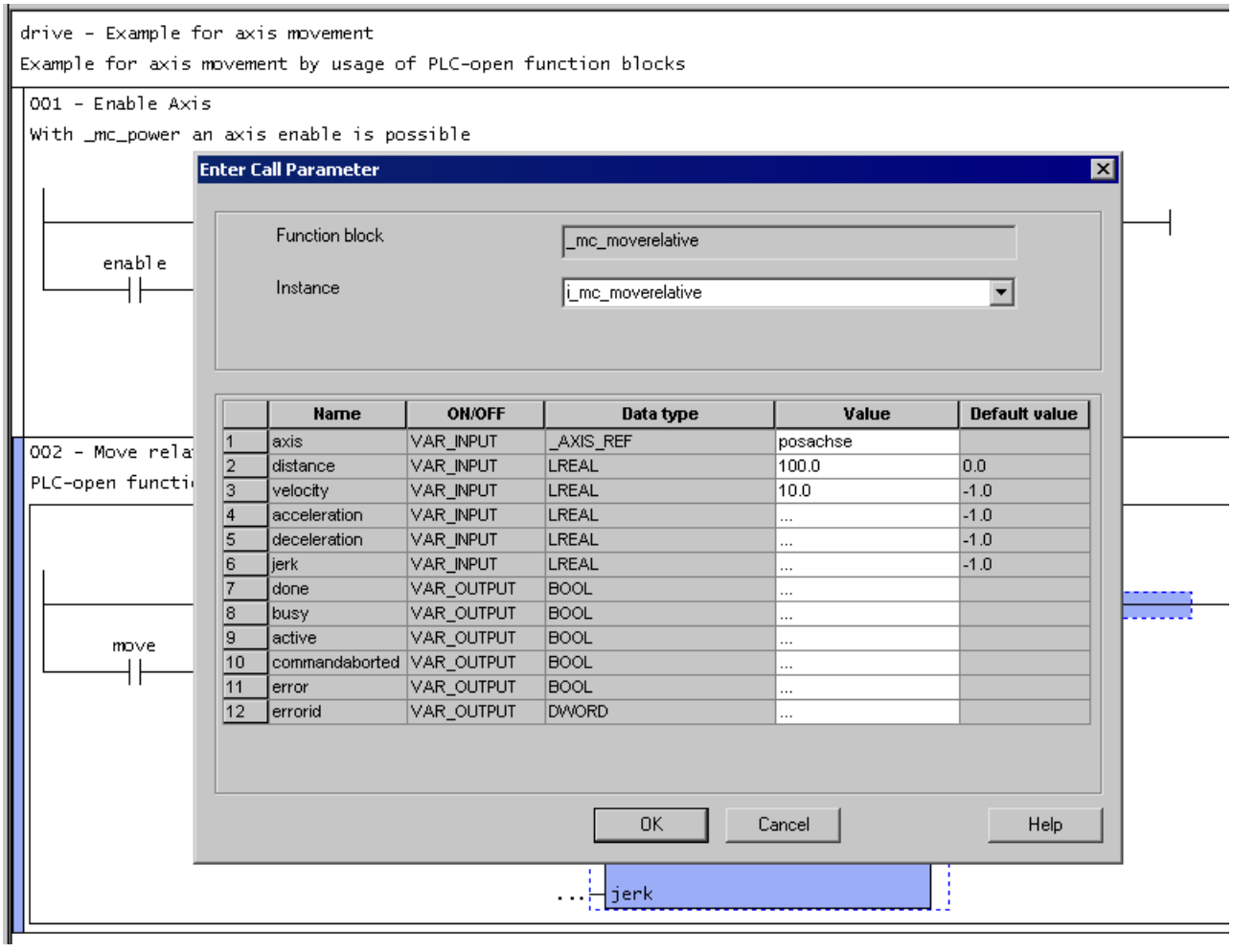


Figure 7-562 Set call parameters for the PLCopen block _MC_MoveRelative

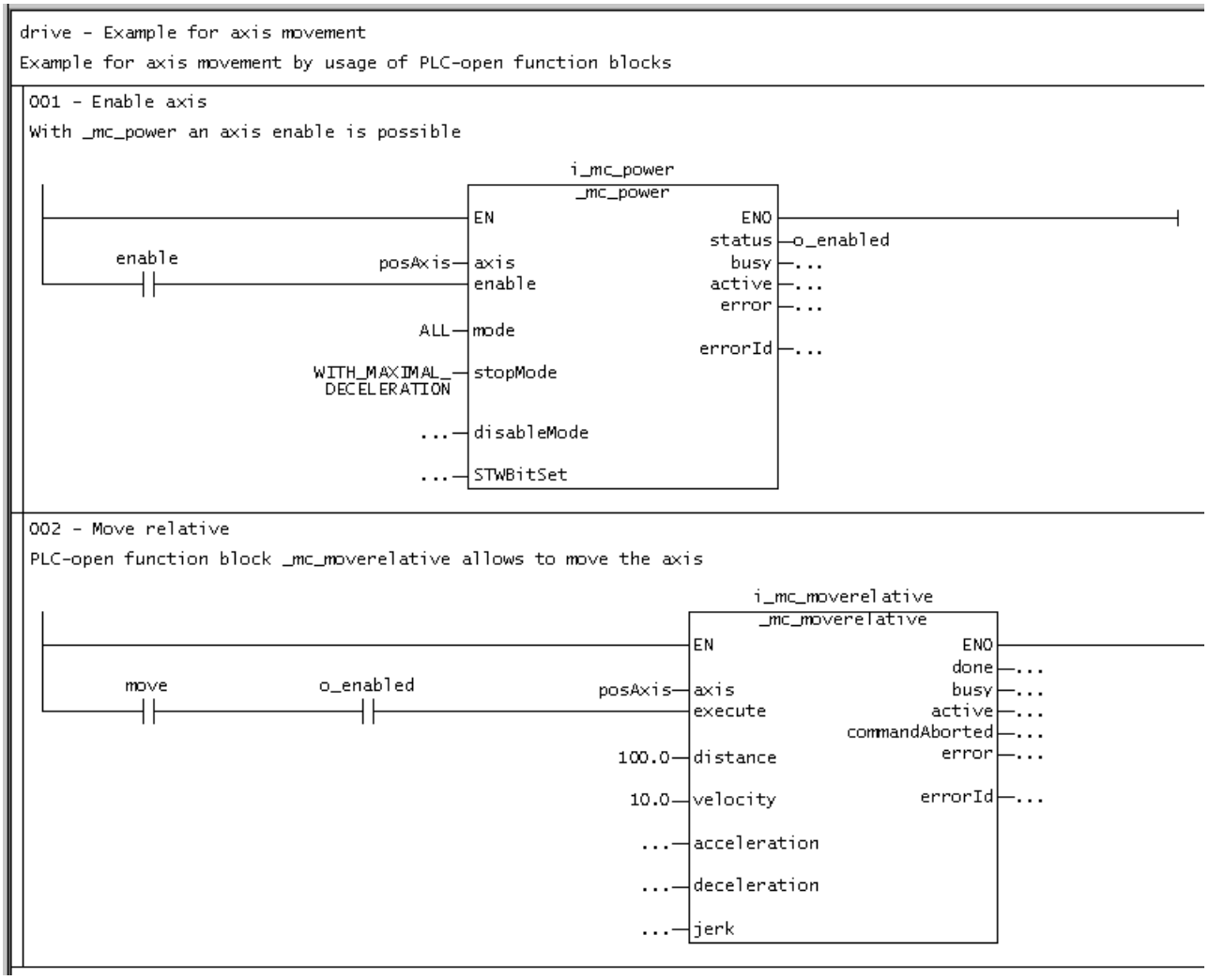


Figure 7-563 Network with entered variables

Details view

To show the detail view, proceed as follows:

1. Select the **View > Detail view** menu item.
   Information, e.g. compiler messages, will be displayed during the compilation of a program.

Compiling

To compile the program, proceed as follows:

1. Select the program in project navigation.
2. Open the **LAD/FBD program** menu and select **Accept and compile**.

During the compilation process, messages on the successful compilation status are displayed in the detail view. Should any error occur during compilation, they will be displayed in plain text there.

Assigning a sample program to an execution level

Before you can run the sample program, you must assign it to an execution level or a task. When you have done this, you can establish the connection to the target system, download the program to the target system, and then start it.

To assign the program to an execution level (see also the blinker program (Page 5387) example), proceed as follows:

1. Double-click the **EXECUTION SYSTEM** folder in the project navigator.
2. Mark the **BackgroundTask**.
3. Click the **Program assignment** tab.
4. Select the program.
5. Click the button **>>**.
6. Click **Close**.

See also

Assigning a sample program to an execution level (Page 5387)

Starting sample program

To start a program, proceed as follows:

1. Make sure the LAD/FBD unit creates the additional debug code for **program status** during compilation:
   Open the Properties window for the LAD/FBD unit (see Defining the properties of an LAD/FBD unit (Page 5109)).
2. Activate the **Permit program status** compiler option on the **Compiler** tab as the local compiler setting (see Local compiler settings (Page 5112)) for this LAD/FBD unit.
   See also the description relating to Effectiveness of local or global compiler settings (see the SIMOTION ST Programming and Operating Manual).
3. Select **Project > Save and recompile all**.
   The project is locally saved on the hard disk and compiled.
4. Select the **Project > Connect to selected target devices** menu command or click  .
   Online mode is activated.
5. Select the **Target system > Load > Download project to target system** menu command or click  .
   The project data (including the sample program) and the data of the hardware configuration are downloaded to the RAM of the target system.
6. Select both networks and click the  button for **program status** (CTRL+F7 shortcut) in the LAD editor function bar (Page 5086).
   Monitoring the program execution (Page 5349) is switched on.

7. Mark the SIMOTION device in the project navigator and select **Target device > Operating mode** in the context menu.
   The **Operating mode** window with the software switch for modes opens.
8. Click the **RUN** button in the software switch.
   The SIMOTION device is in **RUN** mode. The sample program is run and the current paths/ signal paths are color-coded in accordance with the current signal values (Page 5349).

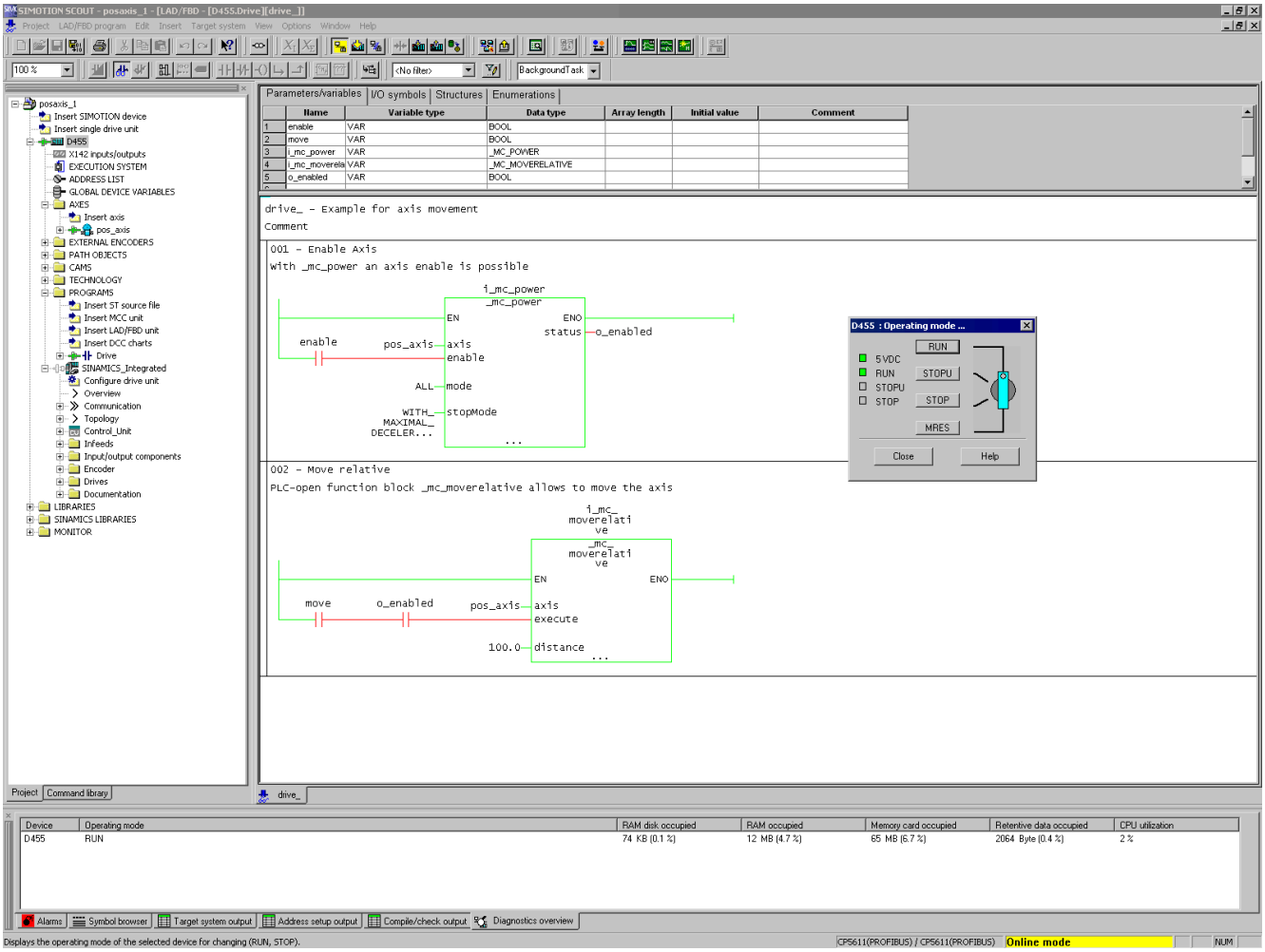


Figure 7-564 Sample program is started

7.3.9 Appendix

7.3.9.1 Key combinations

The following key combinations are available:

With LAD/FBD editor open	
Left/right arrow buttons Up/down arrow buttons	With a selected operator: Navigation between the individual operators
Ctrl+down arrow button	Selects previous network
Ctrl+up arrow button	Selects next network
Ctrl+F7	Switches the program status (Page 5348) function on and off
Ctrl+space	Automatic completion (Autocomplete) (Page 5089)
Pg Up	Selects the network at the start of the visible editor area
Pg Dn	Selects the network at the end of the visible editor area
Del	Deletes an operator
Tab / Shift+Tab	Jumps forward to next button / input field / jumps back to previous button / input field
Return	Opens the input field of the current operand or confirms the entry made in the input field
Esc	Aborts the entry while input field is open
Shift+F6	Switches between declaration table and editor area
Ctrl+Alt+H	Opens the symbol input help dialog window

Window menu	
Ctrl+Shift+F5	Rearranges all windows opened in this application in horizontal tiled format
Ctrl+Shift+F3	Rearranges all windows opened in this application in vertical tiled format
Ctrl+Tab	Switches between windows of the working area

View menu	
Ctrl+F11	Maximizes the working area
Ctrl+F12	Maximizes the detail view
Ctrl+Num+	Enlarges the contents of the working area
Ctrl+Num-	Reduces the contents of the working area
F5	Updates the view

LAD/FBD unit menu	
Ctrl+F4	Closes the LAD/FBD unit
Alt+Enter	Displays the properties of the active/selected object for editing
Ctrl+B	Accepts and compiles the active/selected object
Ctrl+R	Inserts a new network

LAD/FBD program menu	
Ctrl+F4	Closes the LAD/FBD program
Alt+Enter	Displays the properties of the active/selected object for editing
Ctrl+B	Accepts and compiles the active/selected object
Ctrl+R	Inserts a new network
Ctrl+L	Jump label ON/OFF
Ctrl+Shift+K	Shows/hides the comment line
Ctrl+Shift+B	Display options for boxes
Ctrl+T	Symbol check and type update
Ctrl+F7	Program status On/Off
Alt+Shift+F8	Inserts a comparator
Alt+Shift+F9	Inserts an empty box
Ctrl+1	Switches to LAD
Ctrl+3	Switches to FBD

Edit menu	
Ctrl+Z	Undoes the last action (except: Save)
Ctrl+Y	Redoes the last action which was undone
Ctrl+X	Cuts a command
Ctrl+C	Copies a command
Ctrl+V	Inserts a command
Del	Deletes selected commands in the LAD editor
F2	Renames the active/selected object
Alt+Enter	Displays the properties of the active/selected object for editing
Ctrl+Alt+O	Opens the selected object
Ctrl+A	Selects all objects in the current window
Ctrl+F	Local search
Ctrl+Shift+F	Find in the project
F3	Find next (for local search)
Ctrl+H	Local find and replace
Ctrl+Shift+G	Find and replace in a project
Ctrl+J	Next position (for search in the project)

Debug menu	
F12	Activates or deactivates a set breakpoint
Ctrl+F8	Continues the program execution at the activated breakpoint
Ctrl+F10	Next step

LAD elements	
Shift+F2	Inserts an NO contact
Shift+F3	Inserts an NC contact

LAD elements	
Shift+F7	Inserts a coil
Shift+F8	Opens a branch
Shift+F9	Closes a branch

FBD elements	
Shift+F2	Inserts an AND box
Shift+F3	Inserts an OR box
Shift+F4	Inserts an XOR box
Shift+F7	Assignment or jump
Shift+F8	Inserts a binary input
Shift+F9	Negates a binary input

7.3.9.2 Protected and reserved identifiers

Reserved identifiers may only be used as predefined. You may not declare a variable or data type with the name of a reserved identifier.

There is no distinction between upper and lower case notation.

The ST programming language includes protected and reserved identifiers (see the SIMOTION ST Programming and Operating Manual). The same list also applies to the LAD/FBD programming languages. The LAD/FBD programming language also includes the protected and reserved identifiers listed in the table.

You can find a list of all the identifiers whose meanings are predefined in SIMOTION in the SIMOTION Basic Functions Function Manual.

Table 7-604 Protected identifiers applicable only to the LAD/FBD programming language

A	
ANDN	
C	
CAL CALC	CALCN
J	
JMP JMPC	JMPCN
L	
LD	LDN
O	
ORN	
R	
RET RETC	RETCN
S	
ST	STN

X	
XORN	

7.4 SINAMICS/SIMOTION DCC Editor Description

Preface

Compliance with the General Data Protection Regulation

Siemens respects the principles of data privacy, in particular the data minimization rules (privacy by design).

For the product SINAMICS DCC this means:

The product stores personal data only in the project. The user is therefore responsible for ensuring compliance with the statutory data protection provisions. This applies in particular to the transfer of projects. If the user links this data with other data (e.g. shift plans) or if he/she stores person-related data on the same data medium (e.g. hard disk), thus personalizing this data, he/she has to ensure compliance with the applicable data protection stipulations.

The following data must be taken into account.

- Windows login
  In the standard configuration, the product saves the login details of the Windows user together with technical function data (e.g. time stamp) in the project. The specified data is saved in order to trace changes in large configurations.
  With SINAMICS DCC, a personal reference can be established via the project and all elements contained in it (e.g. charts).
  The specified data can be viewed in the properties of the project and the elements in SINAMICS DCC (property "Author") and subsequently changed, with the exception of the last modification of the project.

For the above-mentioned point, the details of the functions mentioned in the respective chapter in the information system of SINAMICS DCC must be observed.

By generating the login or username, personal data can be pseudonymized for the functions. Deleting the project will cause all personal data saved within it to be deleted too.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.3.1:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions

- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

SINAMICS documentation

The SINAMICS documentation is organized on 2 levels:

- General documentation/catalogs
- Manufacturer/service documentation

An overview of the current documentation in the respective available languages can be found on the Internet:

http://www.siemens.com/motioncontrol

Select the menu items "Support" --> "Technical Documentation" --> "Overview of Publications".

Information on the range of training courses and FAQs (Frequently Asked Questions) is available on the Internet:

http://www.siemens.com/motioncontrol

Select the menu item "Support".

Further documentation for the DCC editor

- SINAMICS/SIMOTION Function Manual, *Description of the standard DCC blocks*

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

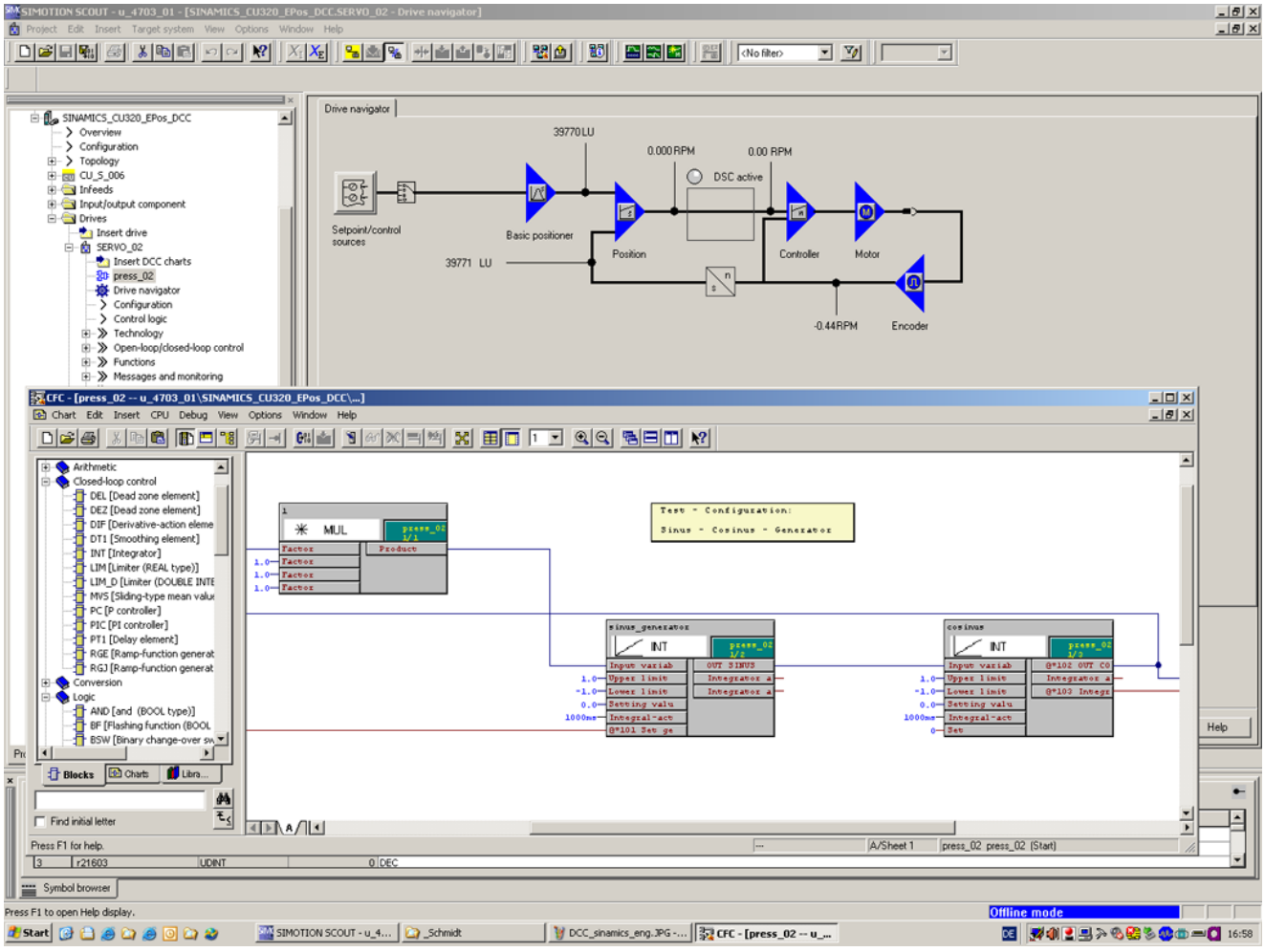
Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

7.4.1 Introduction

Drive Control Chart (DCC) for SINAMICS and SIMOTION means graphic configuration and expansion of the device functionality by means of freely available control, arithmetic and logic blocks

Drive Control Chart (DCC) expands the facility for the simplest possible configuring of technological functions both for the SIMOTION motion control system and the SINAMICS drive system. This opens up a new dimension for users for adapting the specified systems to the specific functions of their machines. DCC has no restriction with regard to the number of usable functions; this is only limited by the performance capability of the target platform.



DCC comprises the DCC editor and the DCB library (block library with standard DCC blocks).

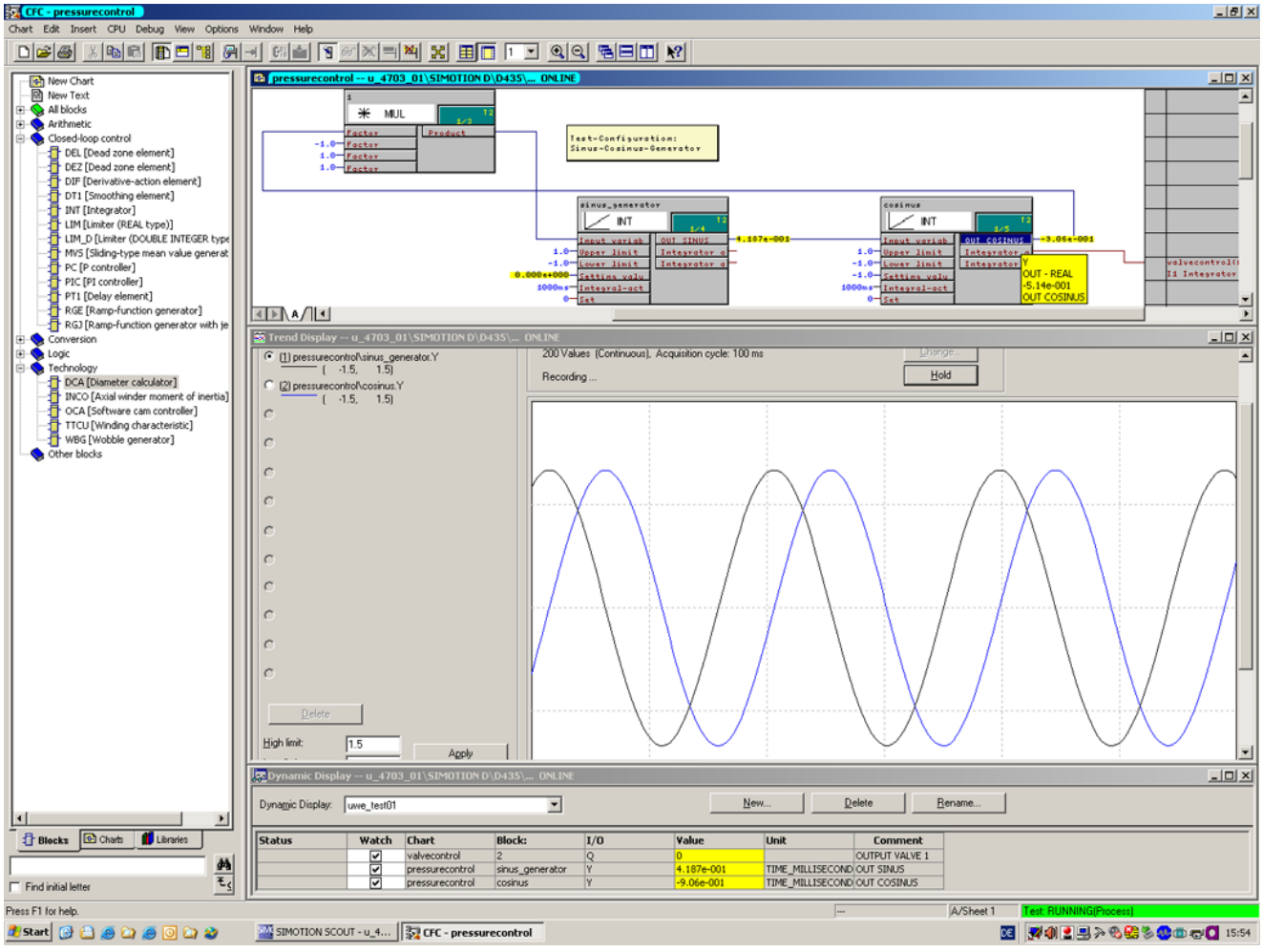
The user-friendly **DCC editor** enables easy graphic configuration and a clear representation of control loop structures as well as a high degree of reusability of existing charts.
The open-loop and closed-loop control functionality is defined by using multi-instance-capable blocks (**Drive Control Blocks, DCBs**) from a pre-defined library (**DCB library**) that are selected and graphically linked by dragging and dropping. Test and diagnostic functions allow verification of program behavior or the identification of causes in the event of errors.

The block library contains a large selection of control, arithmetic and logic blocks as well as extensive open-loop and closed-loop control functions.

All commonly used logic functions are available for selection (AND, XOR, On/Off delay, RS flip-flop, counters, etc.) for the logic operation, evaluation and acquisition of binary signals. Diverse arithmetic functions such as absolute-value generation, dividers and minimum/maximum analysis are available for monitoring and evaluating numeric variables. In addition to the drive control, axial winder functions, PI controllers, ramp-function generators or sweep generators can be configured simply and without problem.

Almost unlimited programming of control structures is possible in conjunction with the SIMOTION motion control system. These can then be combined with other program sections to form an overall program.

Drive Control Chart for SINAMICS drives also provides a convenient basis for resolving drive-level open-loop and closed-loop control tasks directly in the converter. This results in further adaptability of SINAMICS for the tasks set. On-site processing in the drive supports modular machine concepts and results in increased overall machine performance.



- With SINAMICS, DCC can be activated simultaneously on several drive objects (DOs) on a drive unit. With SIMOTION, several DCC charts can be created in a program container.
- Block library with management, arithmetic, control, logic and complex blocks.
- Graphic component connection editor with various editing, macro, help, search, comparison and print functions.
- Simple configuration of axial winder functions, PI controller, ramp-function generator or sweep generator.
- Execution environment for SIMOTION with sampling times that can be selected and mixed, and consistent data transfer between sampling times.
- Execution environment for SINAMICS with the option of embedding technology in SINAMICS using BICO technology so that applications can be set via configured parameters. Up to ten different sampling times can be configured.

- Diagnostics environment with signal display, diagnostic and trace functions.
- Scalability with different performance features and quantity structures in DCC SIMOTION and DCC SINAMICS.

Note

The DCC editor is a programming system based on CFC (Continuous Function Chart). The following sections describe the principles of DCC operation that have not already been covered in the manual titled "CFC Manual for S7".

Note

Further references for the DCC editor:

SINAMICS/SIMOTION Function Manual, *Description of the standard DCC blocks*

Differences between DCC SIMOTION and DCC SINAMICS

DCC SIMOTION and DCC SINAMICS differ to some extent in their mode of operation. The basic differences are listed below:

Table 7-605 Differences between DCC SIMOTION and DCC SINAMICS

	SIMOTION	SINAMICS
System integration	Via interconnection to variables of the basic system	Via adjustable parameter or interconnection via BICO parameter to the basic system
Execution system	Any number of execution groups that can be assigned five time slices (depending on the system cycle clocks)	Maximum of ten execution groups that can be assigned ten different sampling times
Consistency in the data transfer	Consistent data transfer also across the time slices	Consistency in the data transfer must be established by the user, when required, by means of standard blocks (SAH_X)
Scope of standard blocks	See SINAMICS/SIMOTION Function Manual, *Description of the standard DCC blocks*	See SINAMICS/SIMOTION Function Manual, *Description of the standard DCC blocks*

Availability of DCC features

Table 7-606 Availability of DCC features

Feature	Available as of version
Monitoring and tracing of published parameters	SINAMICS 2.5 / SIMOTION 4.1
Online changes in test mode and tracing of unpublished block connections	SINAMICS 2.6 / SIMOTION 4.1
Read back of BICO interconnections from the target device	SINAMICS 2.5 / DCC 2.0.3 / STARTER/SCOUT 4.1.3
Creation of installable libraries	SINAMICS 2.5 / DCC 2.0.3 / STARTER/SCOUT 4.1.3
DCC libraries	SINAMICS 4.4 / DCC 2.1 / STARTER/SCOUT 4.2
Installation of block libraries in the case of an opened SCOUT	SCOUT V4.1.2
Installation of block libraries in the case of an opened STARTER	STARTER V4.2

Feature	Available as of version
Online insertion and deletion of block instances	SINAMICS 2.6 / SIMOTION 4.1.2
Online changes, insertion and deletion of interconnections	SINAMICS 2.6 / SIMOTION 4.1.2
Display of reference data for DCC charts	SCOUT 4.1 / STARTER 4.1.3
Search and replace for DCC sheet bar interconnections	SCOUT/STARTER 4.1.2
Compiling without DCC license	SCOUT/STARTER 4.1.3
Creating C-block libraries automated from DCC libraries	DCC SIMOTION 2.0.3
Fixed execution group "BEFORE actual position value"	SINAMICS 4.3
Chart-by-chart XML export/import	SCOUT/STARTER 4.2 / DCC 2.1
User-defined structures, DCC SIMOTION	SCOUT 4.2 / DCC 2.1
Read back DCC chart sources from the target device	SINAMICS 4.4 (only in conjunction with TPdcblibV3.0_SINAMICS_4_4 or higher) / SCOUT / STARTER 4.2 / DCC 2.1
Automatic library exchange when upgrading the device version	SCOUT/STARTER 4.2
Exchanging target device families for DCC libraries	SCOUT 4.3 / DCC 2.2

7.4.2 Safety instructions and industrial security

7.4.2.1 Fundamental safety instructions

General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. Application examples do not represent specific customer solutions, but are only intended to provide support for typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit:

Industrial security (http://www.siemens.com/industrialsecurity)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at:

Industrial security (http://www.siemens.com/industrialsecurity)

Further information is provided on the Internet:

Industrial Security Configuration Manual (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ WARNING

Unsafe operating states resulting from software manipulation

Software manipulations (e.g. viruses, trojans, malware or worms) can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
- Make sure that you include all installed products into the holistic industrial security concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- Protect the drive against unauthorized changes by activating the "know-how protection" drive function.

7.4.2.2 Use write and know-how protection

Prevent unauthorized changes by means of know-how protection

⚠ WARNING

Danger to life through manipulation of DCC charts and DCC libraries

The use of unprotected DCCs and DCC libraries entails a higher risk of manipulation of DCCs, DCC libraries and backup files.

- Protect important DCC charts and DCC libraries by using **know-how protection programs** or via the **know-how protection for drive units** in the SCOUT/STARTER. You can prevent manipulation by assigning a strong password.
- For **know-how protection programs** and **the know-how protection of drive units**, use passwords which include at least eight characters, upper and lower cases, numbers and special characters.
- Make sure that only authorized personnel can access the passwords.
- Protect the backup files on your file system using a write protection.

7.4.2.3 Industrial Security Configuration Manual

Manual on industrial security

Further information can be found in the configuration manual "Industrial Security" for SINAMICS, SINUMERIK and SIMOTION at Security Manual (https://support.industry.siemens.com/cs/document/108862708/sinumerik-simotion-sinamics-industrial-security?dti=0&lc=en-DE).

Note in particular the explanations on the cell protection concept in Section "General security measures - Network segmentation".

7.4.3 DCC editor functionality

7.4.3.1 Overview

The product offers a modular, scalable technology option, which has chiefly been developed for drive-related, continuous open-loop and closed-loop control engineering tasks.

The DCC technology option for SIMOTION controllers and SINAMICS drives can be configured graphically using the Drive Control Chart editor (referred to below as DCC editor), which is based on CFC. The following figure illustrates the data flow of the configuration data when configuring with the DCC technology option:

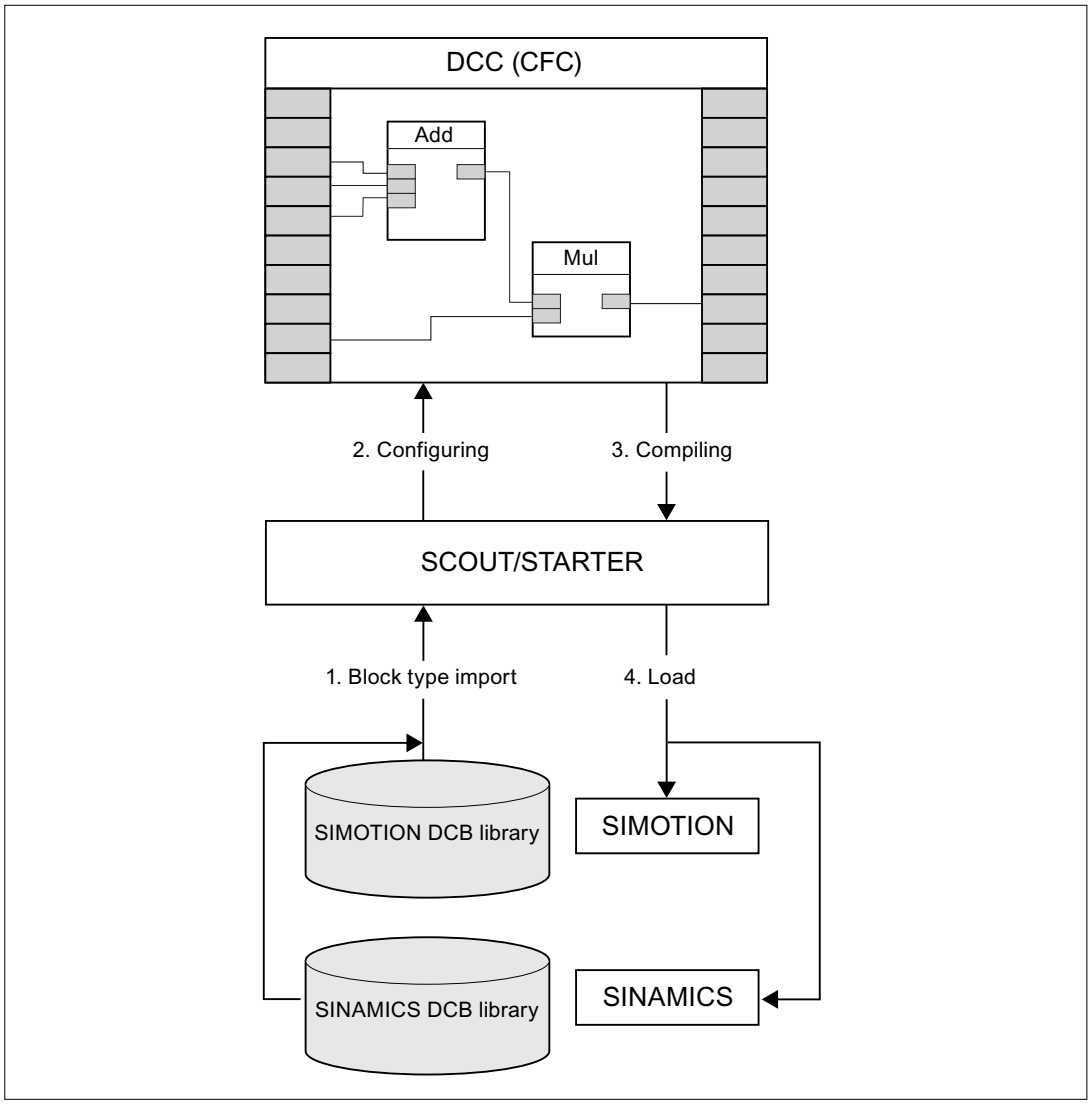


Figure 7-565 Flow of configuration data

1. When a new chart is created, the block types are taken from the device-specific block library and inserted in the DCC editor's block manager.
2. The DCC editor is used to create charts, in which you can insert, parameterize and interconnect blocks.
3. When you compile the charts, an intermediate code is created.
4. This is downloaded to the device or drive unit using STARTER/SCOUT.

7.4.3.2 Requirement

This description refers to the following software versions of the devices and engineering systems:

- DCC V3.3.0.1
- SIMOTION P, C, D as of Version 5.3.1
- SINAMICS Integrated for SIMOTION D as of Version 5.1
- SINAMICS S120, S150, G130, G150 as of Version 5.2
- SINAMICS SM150, GM150, GL150 and SL150 as of Version 4.8.2
- SINAMICS DCM as of Version 1.5
  Except for the utilization display, the SINAMICS DC MASTER as of Version 1.4 supports the same function scope as SINAMICS S120 as of Version 4.7. Detailed information about the SINAMICS DC MASTER Version 1.4 can be found in the SINAMICS DC MASTER 6RA80 DC Converters Operating Instructions, 01/2014 Edition
- SINAMICS DCP as of Version 1.2
- SIMOTION SCOUT/STARTER as of Version 5.3
- Required CFC versions for DCC V3.3.0.1:
  - STARTER: CFC V8.0, CFC V8.0.4, CFC V8.1, CFC 8.2 or CFC V9.0
  - SIMOTION SCOUT: CFC V9.0
- Engineering license required for DCC V3.3.0.1:
  - DCC SINAMICS: CFC for SINAMICS V7.1, V8.0, V8.1, V8.2 or V9.0
  - DCC SIMOTION: CFC for SIMOTION V7.1, V8.0, V8.2 or V9.0

Note

If errors occur when charts are being compiled in SIMOTION SCOUT/STARTER, it is possible to generate a comprehensive error report by compiling them again in the DCC editor. To do this, activate **Display all messages with 'Save and compile changes'** on the **Compiler** tab under the **Settings** menu command in SIMOTION SCOUT. A corresponding error report is automatically created in STARTER.

Note

The DCC editor is installed automatically with the SIMOTION SCOUT/STARTER engineering system.

- DCC setup 2.0.2 - 2.0.5 can be installed with all SIMOTION SCOUT/STARTER V4.1 service packs and hotfixes.
- DCC setup 2.0.2 - 2.0.5 can be installed in SIMOTION SCOUT/STARTER 4.2.
- DCC setup 2.1 can be installed as of SIMOTION SCOUT/STARTER 4.2.
- DCC setup 2.2 can be installed as of SIMOTION SCOUT/STARTER 4.3.
- DCC setup 2.3 can be installed as of SIMOTION SCOUT/STARTER 4.4.
- DCC setup 2.4 can be installed as of STARTER 4.5.
- The DCC setup 3.1 can be installed as of STARTER 5.1.
- DCC setup 3.3.0.1 can be installed as of STARTER 5.3.

An appropriate program license is required for the DCC editor. This is on the USB stick supplied with the DCC SIMOTION or DCC SINAMICS product.

The installation requirements are also stated in the Readme file for the software version.

Note

It is possible to print DCCs in the DCC editor, but not in SIMOTION SCOUT/STARTER.

Note

Only context-sensitive help is available for the DCC blocks, i.e. the descriptions cannot be accessed from the main help page.

Note

If the DCC charts are opened with a different CFC version than the one with which they were created, this may lead to inconsistencies within the project. The DCC chart must be recompiled and reloaded. DCC charts that have been edited with a newer CFC version can no longer be opened with an earlier CFC version. Backward conversion is not possible.

If you open a project whose charts have been created with an older CFC version, you can select whether the data format of the charts should be updated. If you reject this, you can in fact view the CFC charts, but cannot edit them any further. It is recommended to adapt the data format of the CFC charts to the currently installed CFC version.

7.4.3.3 New device versions

SIMOTION

The DCC functionality and performance of the new SIMOTION target devices is compatible with the previous versions, i.e. DCC V3.3.0.1 within the framework of SIMOTION SCOUT 5.3 supports the following target device versions:

- SIMOTION V4.1 devices
- SIMOTION V4.2 devices
- SIMOTION V4.3 devices
- SIMOTION V4.4 devices
- SIMOTION V4.5 devices
- SIMOTION V5.3.1 devices

The DCC functionality and performance of the new SOC2-based SIMOTION target devices (e.g. D410-2 DP) is compatible with the corresponding IDC-based previous devices and versions (e.g. D410 DP).

Note

Device versions

The device versions and software versions supported may be listed in the corresponding Readme files.

Note

A compatibility list is provided on the SIMOTION SCOUT DVD and at the following link on the Internet:
SIMOTION compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317)

SINAMICS

DCC SINAMICS supports in STARTER/SCOUT V5.3 all previously released device versions with DCC.

In addition, DCC is offered for:

- SINAMICS 4.8.2, 5.1, 5.1.1 and 5.2
- DCM 1.5

The following device types/device versions are supported with SINAMICS 5.2:

- S120 CU320-2/CU310-2
- S150 CU320-2
- G130 CU320-2
- G150 CU320-2

SINAMICS 4.8.2 supports the following device types/versions:

- GL150 CU320-2
- GM150 CU320-2
- SL150 CU320-2
- SM120 CU320-2

Note

With the installation of an SSP, the latest standard *dcblib* library available for the device is installed and can be used on the device.

7.4.3.4 Establish the project requirements

Create a project

You must create a new project in the SIMOTION SCOUT/STARTER engineering system before using the DCC editor.

Procedure

Proceed as follows to create a project:

1. Start the SIMOTION SCOUT/STARTER engineering system.
2. Execute the **Project > New** function from the menu bar.

3. In the **New Project** window, enter the name of the project in the **Name** field.

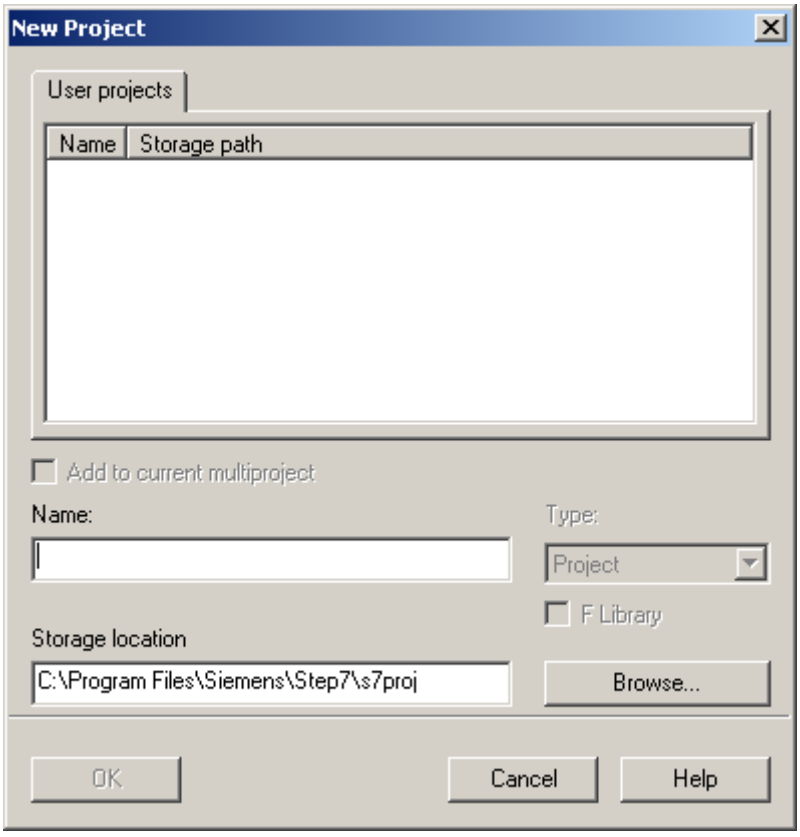


Figure 7-566 Window - New Project

4. Click **OK** to close the window.

The new project is created and then automatically opened.

Note

Convention for assigning names for projects

The project name may contain a maximum of 24 characters. The folder name is generated using the first eight characters of the project name. It is therefore important to ensure that the first eight characters of the project name are unambiguous.

Inserting a device into a project

SIMOTION

Proceed as follows to insert a device into a project:

1. Open an existing SIMOTION SCOUT project if a project is not already open.
2. Execute the **Create new device** command.

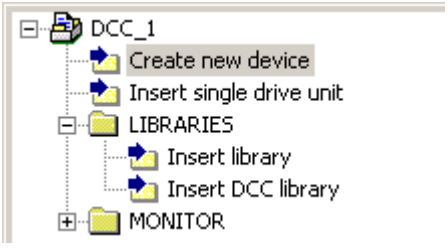


Figure 7-567 SIMOTION: Create new device

3. In the **Create new device** window, select the required device and close the window with **OK**.

Note

For further information on the **Open HW Config** switch, please refer to the documentation for SIMOTION SCOUT.

All requirements for creating a DCC in the project have now been met.

SINAMICS

1. Open an existing project or create a new project in which you want to insert a SINAMICS drive unit (e.g. SINAMICS S120 CU 320). Note that the SINAMICS S110 (CU 305 module) does not support DCC.

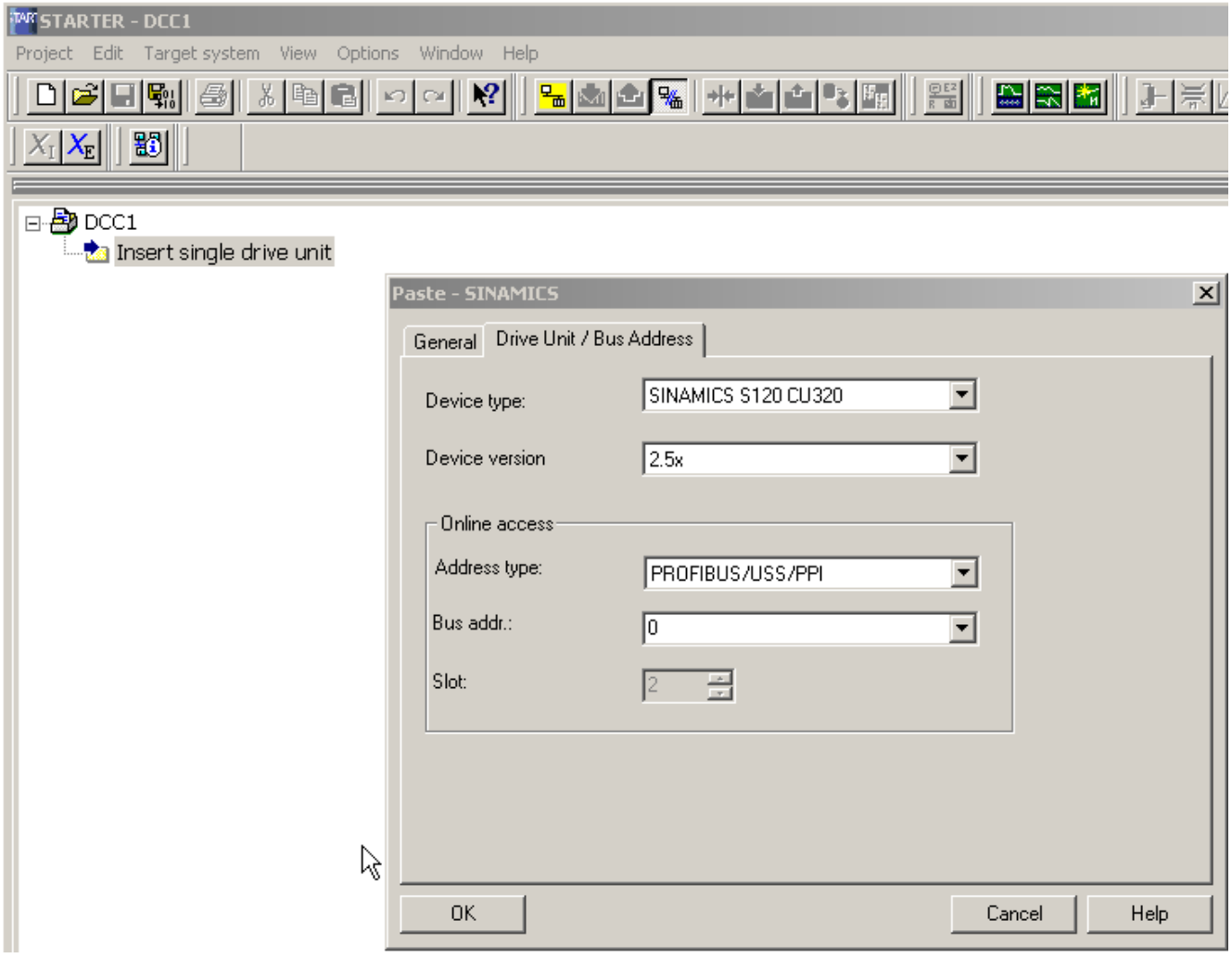


Figure 7-568 Inserting a device into a project

Inserting the DCC chart in a project

You can now insert a DCC chart into the existing project.

Procedure

1. Select a device from the project.
2. **SIMOTION:**
   Execute the **Insert DCC chart** function from the **PROGRAM** subitem of the device.
   SINAMICS, STARTER:
   Execute the **Insert DCC chart** function on the desired drive object. There may only be one DCC chart on a drive object.

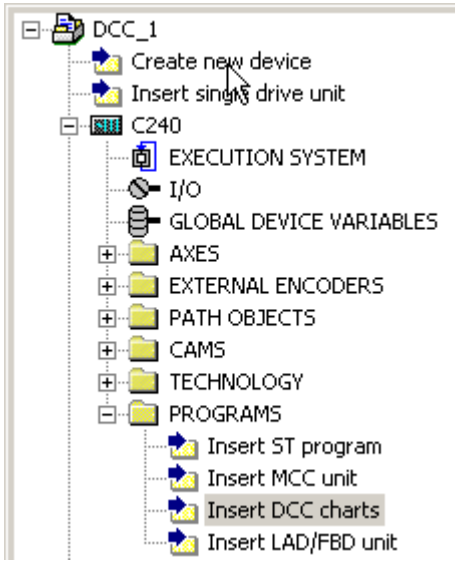


Figure 7-569 SIMOTION SCOUT: Inserting a DCC chart

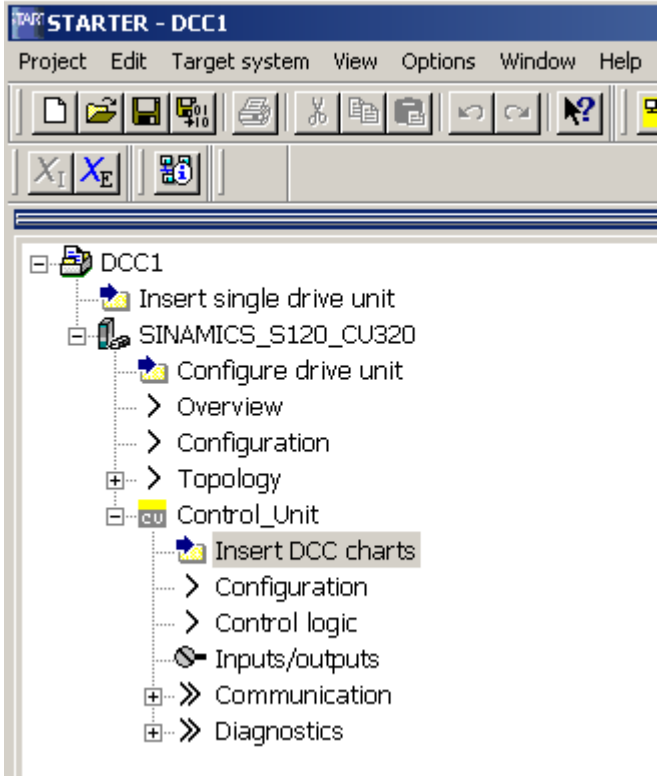


Figure 7-570 Inserting a DCC chart in the CU drive object of a SINAMICS CU3x0.x drive unit with STARTER

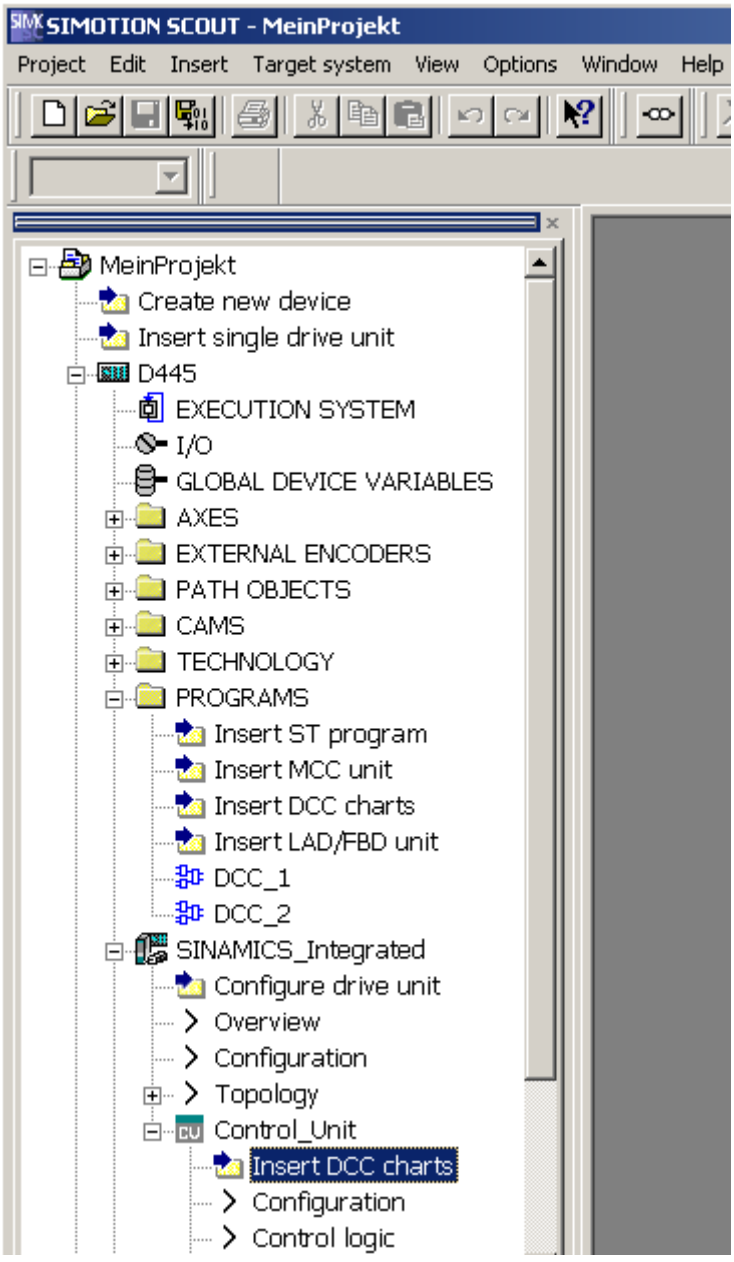


Figure 7-571 SIMOTION D4xx: Inserting a DCC SINAMICS chart on the drive object of the CU with SCOUT

3. The **Insert DCC Chart** window appears.

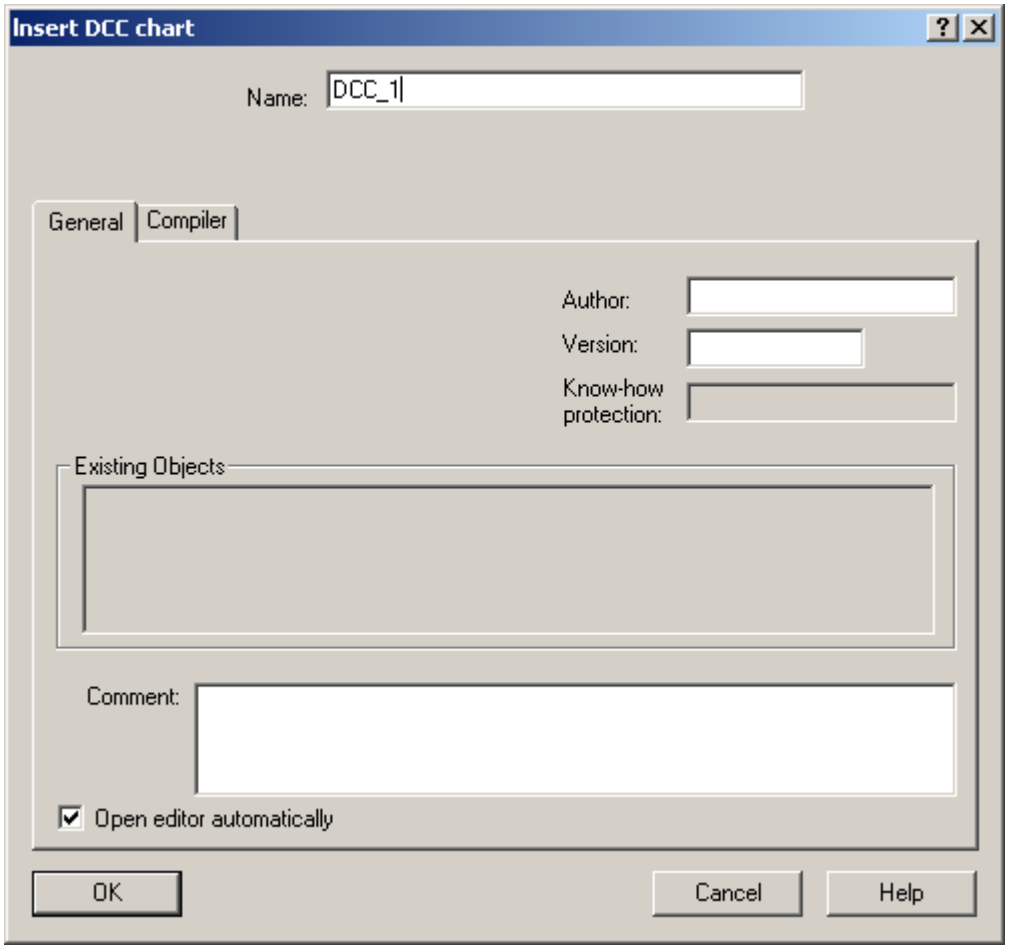


Figure 7-572 Window - Insert DCC Chart

4. Assign a name to the DCC chart.

The DCC chart has now been created.

If you have selected the **Open editor automatically** option in the **Insert DCC Chart** window then the DCC editor opens automatically. When opening the DCC chart for the first time, the **Import DCB Library** window is automatically displayed.

Note

Convention for assigning names to charts

The chart name may contain a maximum of 22 characters.

Explanation of the various types of chart

There are three different types of chart:

- Basic chart
- Chart partition
- Subchart

Charts that are visible within SIMOTION SCOUT/STARTER or the SIMATIC Manager are designated as basic charts. Every basic chart has up to 26 chart partitions and each of these partitions comprises six sheets. Embedded charts - the subcharts - can be used within each sheet. Each of these subcharts may also have its own chart partitions and subcharts. A maximum number of eight nesting levels with subcharts is possible.

Subcharts are not visible as charts in SIMOTION SCOUT/STARTER or in the SIMATIC Manager.

The following graphic clarifies the connection between the three types of chart.

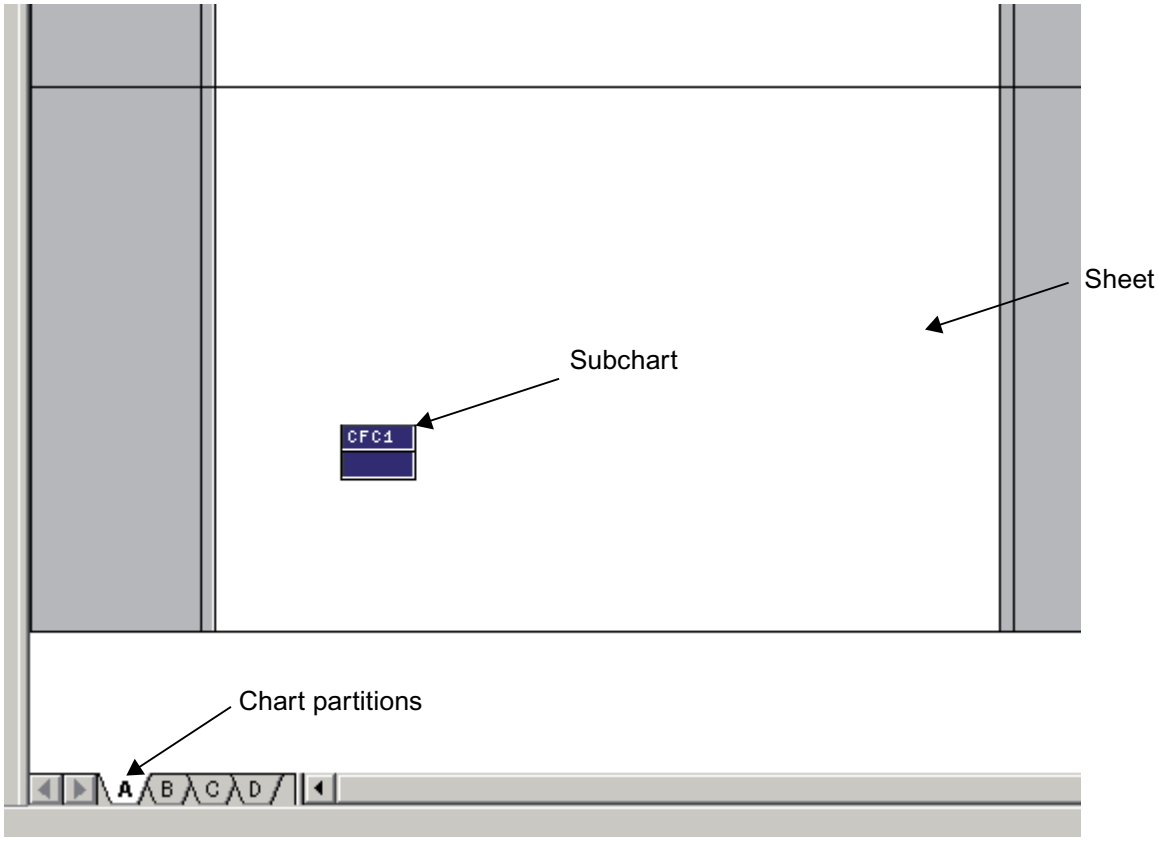


Figure 7-573 Connection between the types of chart in the DCC editor

Select the DCC chart in the project navigator and open the **DCC Chart Properties** dialog via the **Properties...** context menu to access certain properties of the DCC chart. On the **General** tab under **Time stamp** under **Last changed on (STEP7):** you can see the date on which the DCC chart was changed the last time by STEP 7 (CFC editor) or changes accepted in **DCC chart properties**. Under **Last changed on:** you can see the date on which the DCC chart was last

compiled or changes accepted in **DCC chart properties**. Under **Project storage location:** you can see the path under which the project was stored.

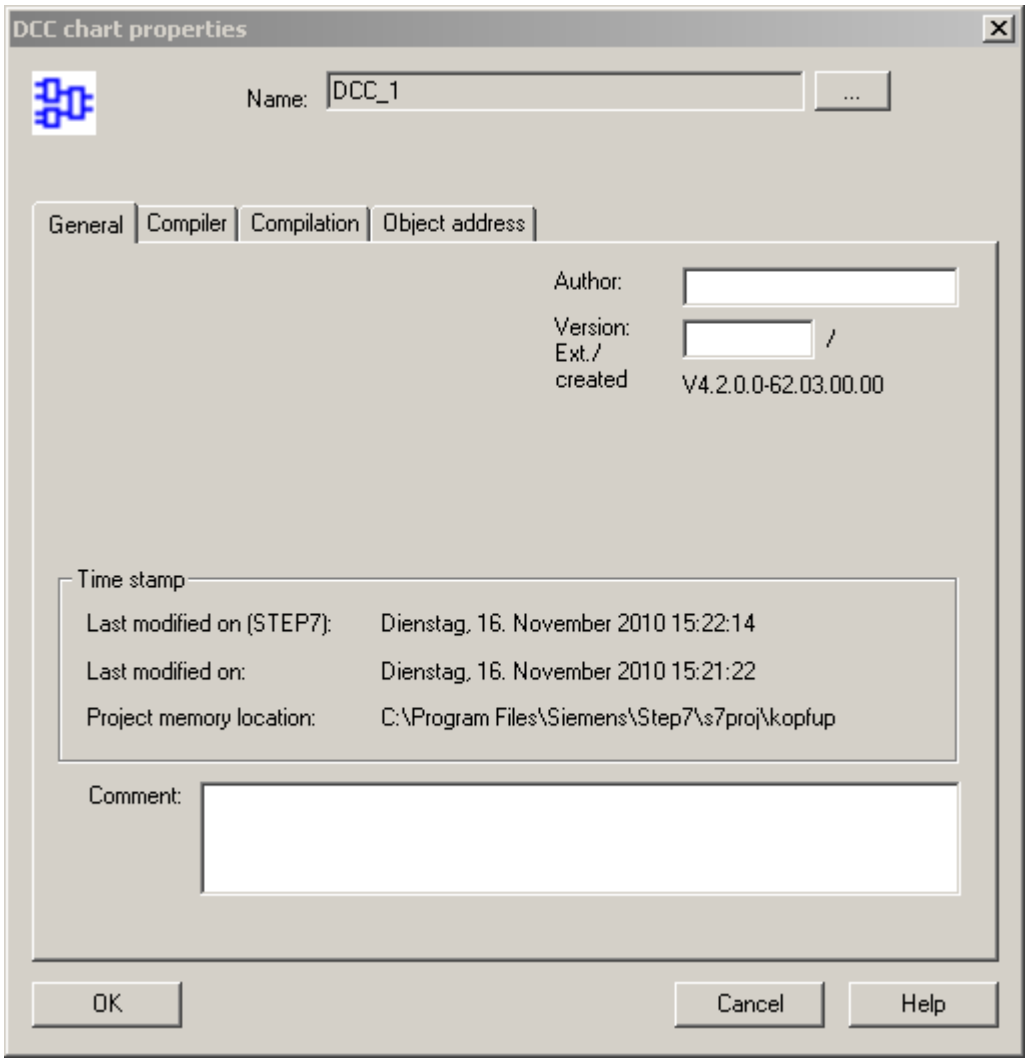


Figure 7-574 DCC chart properties

Inserting a new subchart

A chart (subchart) can be inserted in another chart (chart-in-chart-technique). Hierarchical structures can be formed here. Each chart that is inserted can be opened and modified. A chart can be encapsulated for further use, i.e. chart I/Os added. Which block connections are provided at the chart connections can also be specified individually.

Requirement

You have created a DCC chart in the SIMOTION SCOUT/STARTER engineering system which is opened in the DCC editor.

Procedure

1. Use the **View > Overview** menu command or the  button on the toolbar to switch from the page view to the chart overview. The six pages of the selected DCC chart are shown.
2. Use the **Insert new chart** context-menu command to insert a new subchart, then open the subchart by selecting **Open** from its context menu.

Note

DCC charts should always be created in STARTER/SCOUT.

If a DCC chart that is assigned to a SINAMICS drive object is open and additional DCC charts are created directly in the CFC editor, this can result in compilation errors.

Inserting new chart partitions

Requirement

You have already created a DCC in the SIMOTION SCOUT/STARTER engineering system which is opened in the DCC editor.

Procedure

1. Insert a new chart partition in the desired position using the menu items **Insert > Chart partition > In front of current chart partition** or **At the end**.
2. Alternatively, you can right-click an already existing chart partition on the tab and select **Insert chart partition in front of current chart partition** or **Insert chart partition at the end**.

View and representation

Going into the page view or overview representation

To change to the page view from the overview representation, right-click an empty space in the chart and select **Display this page** in the context menu that appears. The names of the block connections are displayed in this enlarged view.

To change to the overview representation from the page view, right-click an empty space in the chart and select **Overview** in the context menu that appears.

You can also switch to the page view and back to the overview again by double-clicking an empty area on a page.

It is also possible to switch between the page view and overview representation using the **View** menu.

You can switch to a page view with the block catalog displayed on the left using the **View -> Catalog** menu.

7.4.3.5 Handling blocks

Introduction

In this section, you will learn which block types are available and how you can insert blocks in a DCC and delete them. You will also learn how to edit block connections.

Note

The online help provides detailed information (incl. timing diagram and plant view) for the individual blocks. To start the help, select the required block in the chart or in the block catalog and press the F1 key.

User responsibility

The user is responsible for verifying and validating the DCC functions which have been programmed with DCC. The user is responsible for the proper operation of all DCC functions which have been created with DCC.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect programming of the DCC functions

Incorrect programming of DCC functions can cause machines to malfunction, which can lead to injuries or death.

- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).
- Verify and validate the DCC functions you have programmed.

Inserting blocks in the DCC editor

Block types in the DCC editor

The block type inventory featured in the block catalog depends on both the device type and the version of the library. You will find the directories for the block families, as well as the directories **All blocks** (containing all blocks) and **Other blocks** (blocks that are not assigned to a family), in the block catalog. The names of the block families in the DCC editor are always in English.

Inserting a block

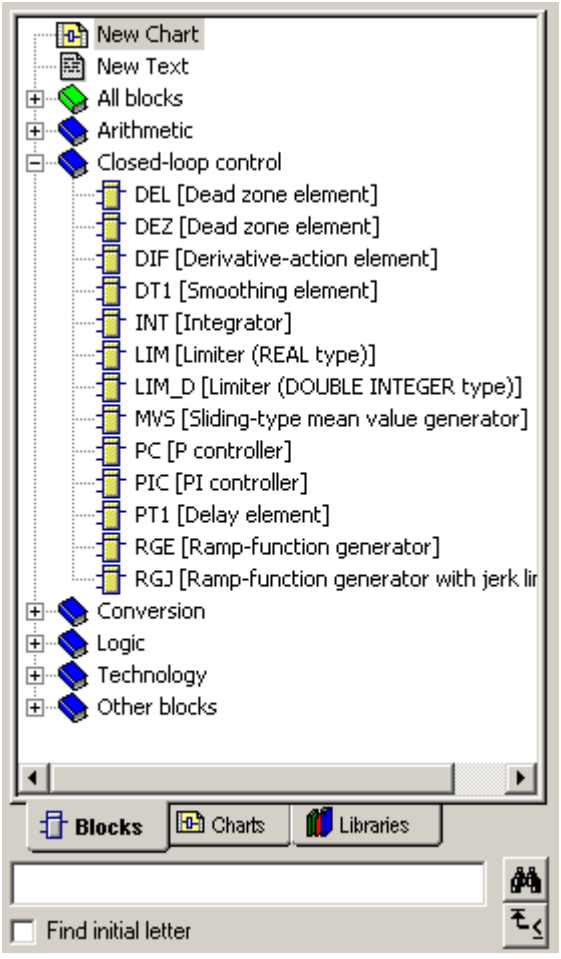


Figure 7-575 Inserting blocks

- Open a block family in the **Closed-loop control** family with the closed-loop control blocks.
- Select the required block and insert it in the chart using drag-and-drop. Only the outline of the block in dashed lines is displayed during the copying procedure. Release the mouse button at the required point.
- To search for a block, enter its name in the input field of the block catalog and click the **binoculars** button. The search process begins.

Note

If blocks are superimposed on the chart with other elements, such as other blocks or the sheet bar, the superimposed block will be displayed in gray and its connections will not be visible. You must reposition the blocks to ensure that all block information can be viewed.

Inserting text

You can add comments to your DCC that you enter in text fields. You can place these at any free position in the chart.

Procedure

Select the **New text** command via the directories of the block types and insert it in the chart using drag-and-drop. Release the mouse button at the required point.

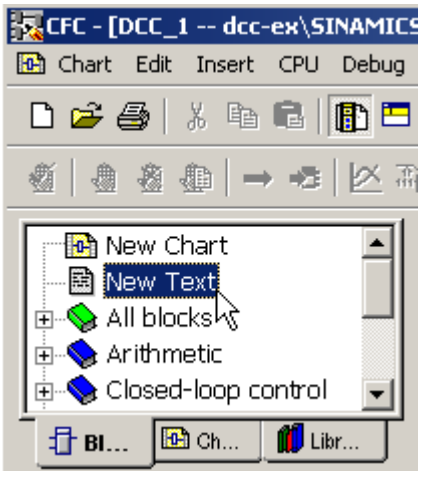


Figure 7-576 Inserting a text field

Alternatively, you can right-click at the desired position in the chart and select the **Insert new text** command in the context menu. Note that this option is only available offline.

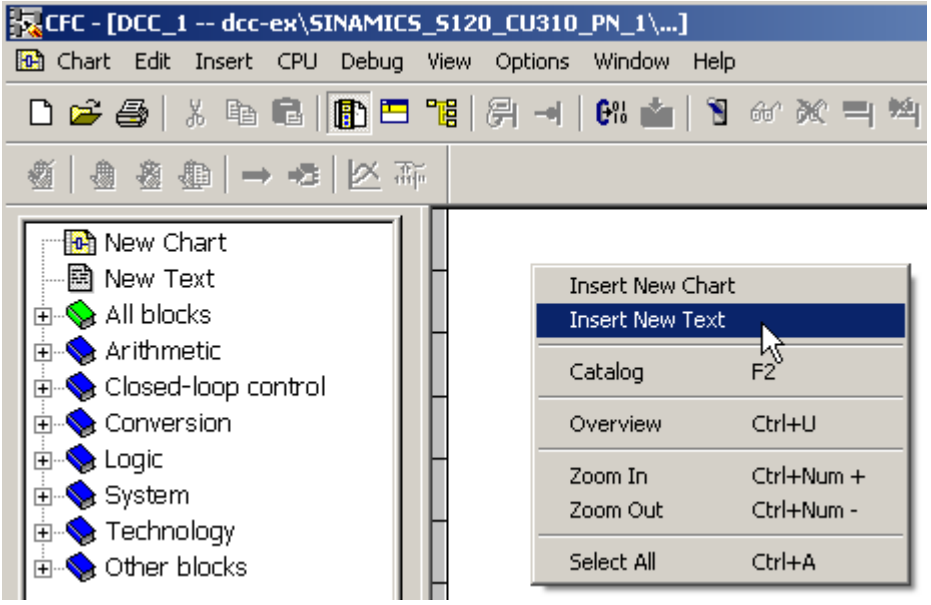


Figure 7-577 Inserting new text

You can adapt the size of the text field to your requirements by selecting the black points at the corners and the sides and dragging them to the desired size.

You can change your comments by clicking the text field and then entering or editing your text.

Specifying execution properties

You can display or change the execution properties of all the used blocks of the program. You can display the properties in the toolbar via **Edit > Execution sequence** or via the button.

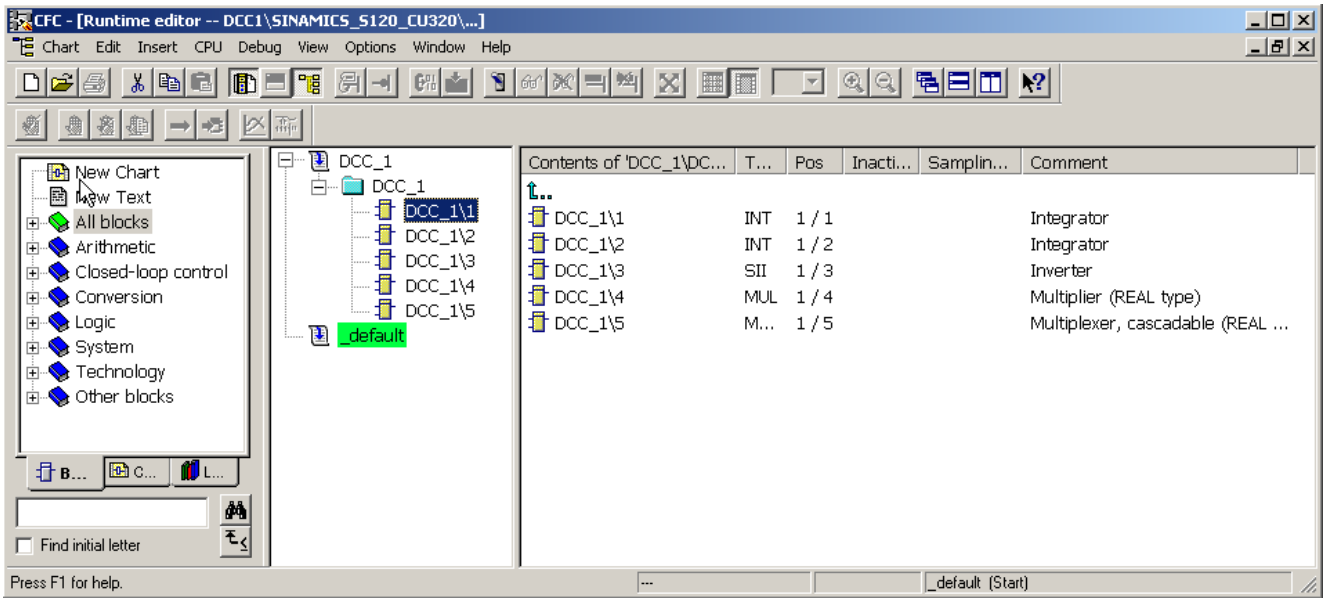


Figure 7-578 Execution editor of the dccReg1 chart with the execution groups Tsg_dccReg1 and Tsg2

In this window, you can also change the insert point in the execution sequence by dragging the block to the desired position. The assignment of a block to an execution group can be changed in the same way.

New inserted blocks are always placed in the execution system behind the block that is defined as predecessor. By default, this is always the block that has been inserted last. If a block is to be inserted in the execution sequence behind an already existing block, select the block with the desired offset in the overview, right-click and select the **Predecessor for insert point** function in the menu. The block now defined as predecessor block is displayed in light green in the DCC editor.

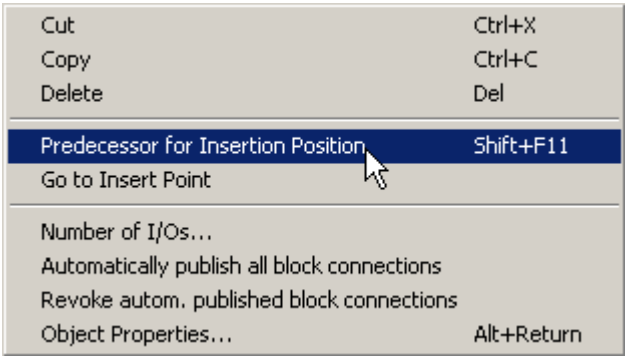


Figure 7-579 Block context menu: Defining predecessor for insert point

The active execution position is displayed in the status bar. This is at the bottom right in the execution editor.

Editing block connections

General

There are two types of block connection (inputs and outputs), each of which has a distinct function and is edited in a particular way.

The following sections contain further information about handling the block connections in the DCC.

Block connection properties

You can double-click each of the individual connections to parameterize it. Alternatively, the **Properties - Connection** window can also be displayed as follows:

1. Select the desired connection.
2. Select **Object properties** in the context menu.
3. The **Properties - Connection** window appears.

However, it is easier to parameterize the inputs as follows:

- Double-click a block header. The **Properties - Block** window appears. You can also open the **Properties - Block** window via the **Object properties** context menu command of the block or via the **Edit > Object properties** menu command.
- Click the **Connections** tab. The parameters in fields with a gray background cannot be changed.
- Enter the required values in the table and click **OK** to close the dialog box.

Input values

At the block inputs, a value can be entered in the **Value** field of the properties dialog box. If the input is not interconnected, it always has the specified value. With interconnected blocks, the output value of the upstream block always applies in the initialization phase and in the first cycle.

Output values

At the block outputs, a value can be entered in the **Value** field of the properties dialog box. In the first cycle however, the specified value is overwritten by the calculated value.

Note

Special feature with hidden block connections

In the DCC editor, you can hide block connections to improve the clarity of the configured charts. However, the hidden block connections remain active in the DCC, so their values are still evaluated. With generic blocks, you can also reduce the number of block connections in the DCC editor. The hidden connections are assigned default values. However, the hidden block connections remain active in the target system, so their values are still evaluated.

It should also be noted that the inputs of these blocks must be interconnected consecutively, starting from the first connection.

Number of block inputs

The number of inputs could be increased for the AND, ADD, MAS, MIS, MUL, NAND, NOR, OR and XOR blocks from the standard library. However, the DCC editor can only evaluate four input signals per block, and therefore this is not permitted. If the demand is greater, the block must be called several times. The note above also applies to data-type-specific variants of the blocks listed above.

Pseudo comments

Comments at block connections that start with @ are pseudo comments; they influence the function of the block connection and serve as interfaces to the basic system.

For further information about the pseudo comments, see:

- Creating customer-specific parameters ("declare") (Page 5543)

Block connection units

The block connection units that can be set in the Properties dialog box serve only as comments in the DCC editor - the values are not used for automatic conversions.

Configuring the block display

You can change the display of the blocks. You can change the block width via **Options > Settings > Block / sheet bar width.....**.

You can change the names of the block inputs/outputs via **Options > Settings > Display** in the submenu **Connections**.

The block type can be displayed in the form of both text and graphics. This can be configured via **Options > Settings > Display** in the submenu **Block headers**.

If you want to display more than the first eight characters of the comment, then select **Options** > **Settings** > **Block / sheet bar width**. Then in the **Blocks / Sheet Bars** window, set the block width to **Wide**. The first twelve characters of the comment are now displayed at the connection.

See also

@ variables (SIMOTION) (Page 5590)

Interconnecting blocks

Blocks can be interconnected with one another. The outputs of a block then form the inputs for further blocks.

Requirement

The inputs and outputs of the blocks must possess compatible data types so that they can be interconnected. An overview of which data types can be interconnected is shown below:

Table 7-607 Conversions

Input	Output	Description
WORD	INT	Interconnection of a word variable to an integer variable
INT	WORD	Interconnection of an integer variable to a word variable
DWORD	DINT	Interconnection of a double word variable to a double integer variable
DINT	DWORD	Interconnection of a double integer variable to a double word variable
BYTE	SINT	Interconnection of a byte variable to a short integer variable
SINT	BYTE	Interconnection of a short integer variable to a byte variable
USINT	BYTE	Interconnection of an unsigned short integer variable to a byte variable
BYTE	USINT	Interconnection of a byte variable to an unsigned short integer variable
USINT	SINT	Interconnection of an unsigned short integer variable to a short integer variable
SINT	USINT	Interconnection of a short integer variable to an unsigned short integer variable
UINT	WORD	Interconnection of an unsigned integer variable to a word variable
WORD	UINT	Interconnection of a word variable to an unsigned integer variable
UINT	INT	Interconnection of an unsigned integer variable to an integer variable
INT	UINT	Interconnection of an integer variable to an unsigned integer variable
UDINT	DWORD	Interconnection of an unsigned double integer variable to a double word variable
DWORD	UDINT	Interconnection of a double word variable to an unsigned double integer variable
UDINT	DINT	Interconnection of an unsigned double integer variable to a double integer variable
DINT	UDINT	Interconnection of a double integer variable to an unsigned double integer variable
SDTIME	REAL	Interconnection of an SDTime variable to a real variable

Procedure

Connect the output of the first block (source) with the input of the second block (sink). You can create this connection either by using drag-and-drop or by latching them together (clicking the relevant input connection and output connection once).

The connection line is automatically drawn from the output of the first block to the input of the second block.

Data type abbreviation in the DCC for connection and transformer blocks

Table 7-608 Table of data types

Abbreviation	Keyword	Name	Bits
BO/B	BOOL	Logical number	8
BY	BYTE	Sequence of 8 bits	8
DI/D	DINT	Double integer	32
DW	DWORD	Sequence of 32 bits	32
I	INT	Integer	16
PC	LREAL	Double floating-point number Accuracy according to IEEE754	64
R	REAL	Floating-point number	32
SI	SINT	Signed short integer	8
TS	SDTIME	The SDTIME data type is derived from the REAL data type; 1.0 corresponds to 1.0 ms Negative values are not defined.	32
UD	UDINT	Unsigned double integer	32
UI	UINT	Unsigned integer	16
US	USINT	Unsigned short integer	8
W	WORD	Sequence of 16 bits	16

Interconnection to chart connections

A chart can be encapsulated for further use, i.e. chart connections added. Which block connections are provided at the chart connections can also be specified individually.

Procedure

1. Model your DCC in the DCC editor.
2. Select the Chart Connections window via **View** -> **Chart connections** or with the button. The chart connections (IN, OUT, IN_OUT) are shown in this window.

   Note

   Please note that chart connections of type IN_OUT are not permitted in DCC!

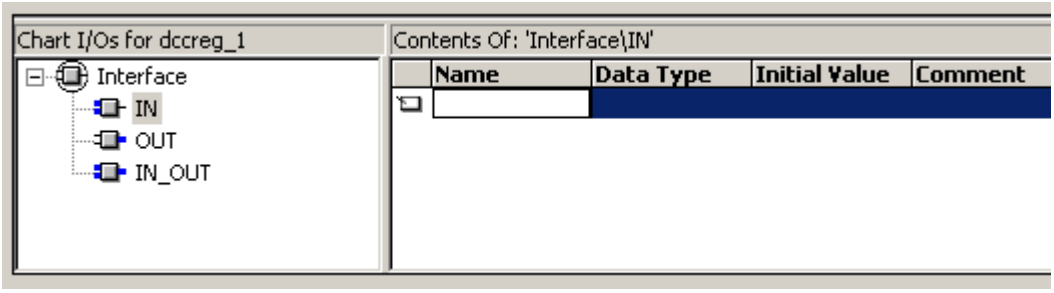


Figure 7-580 Screenshot showing chart connections area of the window

3. Define the required chart connections.
4. Interconnect the block connection to the chart connection. There are three ways of doing this:

   – Option 1:
   Drag the block connection using drag-and-drop to IN (for inputs) or to OUT (for outputs) and keep the Ctrl key pressed. A chart connection is now created automatically.

   – Option 2:
   Select the interface type (IN or OUT) from the left pane and then drag the desired block connection into the last (empty) line in the right pane using drag-and-drop.

   – Option 3:
   Right-click the block connection to be interconnected and select **Interconnection to the chart interface** in the context menu. Then select the appropriate chart connection in the **Insert/Change Interconnection to the Chart Interface** window. Click OK to close the window.

   Note

   The interconnection to the chart connection by the context menu is available only when at least one chart connection exists already. Up to CFC 7.0, the first interconnection to the chart I/O can only be established using drag-and-drop.

The block connection is interconnected to the chart connection. The assignment is displayed in the variables sheet bar of the DCC editor. The block interface is defined as part of assigning block connections to the interface.

Note

DCC SINAMICS: The use of chart I/Os is permitted for subcharts and DCC libraries.

Interconnection to global operands in DCC SIMOTION

Global operands are connection partners located outside of the DCCs.

Interconnections to global operands are entered in the sheet bar.

Where DCC is concerned, the interconnection of global operands serves as an interface to the basic system. You use this function to connect to ST/IO/system variables (in the case of DCC SIMOTION) or BICO parameters (in the case of DCC SINAMICS) in the drive.

Procedure

You can make an interconnection to a global operand as follows:

1. Open the DCC.
2. Right-click the block connection to be interconnected and select **Interconnection to operand** in the context menu.
3. Now select the global operand to be interconnected in the **DCC Signal Selection** window.
4. Click **OK** to close the window.

The block connection is interconnected to the selected global operand.

Deleting blocks

If you want to delete a block from the chart, select it and click **Edit > Delete**. You can also delete blocks using the context menu.

When blocks are deleted, the connections to the block connections are also removed. Output interconnections must first be deleted manually.

If you delete a block in online mode on which the outputs are interconnected with inputs of other blocks, the current signal values will become valid at the inputs of these blocks. These are taken over in the DCC and also saved to the card in the target system at the next RAM to ROM. In offline mode, the default values take effect again at the inputs after deletion of the upstream block.

Note

The STM block cannot be deleted or inserted online.

If you delete a block on which the inputs or outputs are interconnected to chart connections, the connections are deleted, but not the chart connections. If these are not required, you must delete them separately in the Chart connections field.

Rearranging parameter numbers

Call the **Rearrange Parameter Numbers** dialog box via the **Chart -> Rearrange parameter numbers** menu to move parameter numbers.

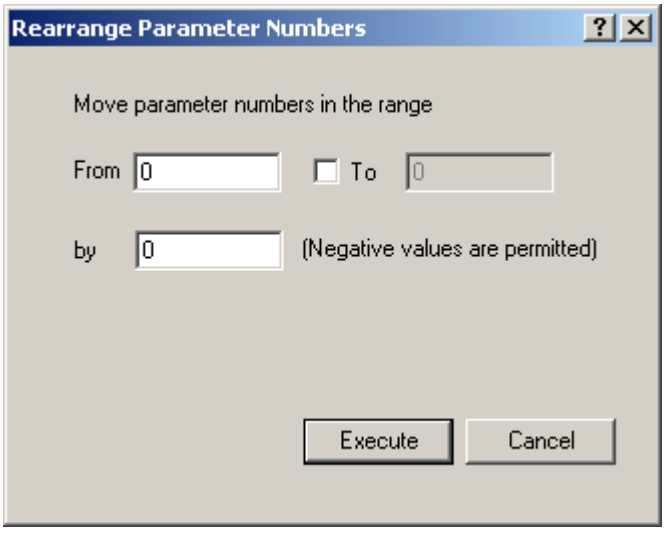


Figure 7-581 Rearranging parameter numbers

Enter the new values and accept these with the **Execute** button.

The following warning is output for an invalid entry.

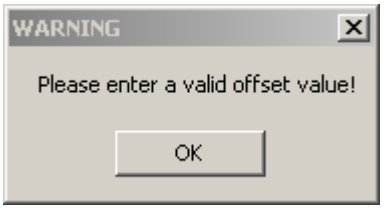


Figure 7-582 Warning message for an invalid entry

Correct your entries.

If your entries were valid, you will find detailed information about moving the parameter numbers in the following **Logs** dialog box on the **Rearrange parameter numbers** tab. In the case of errors, mark the entry and navigate to the problem position with the **Go to** button.

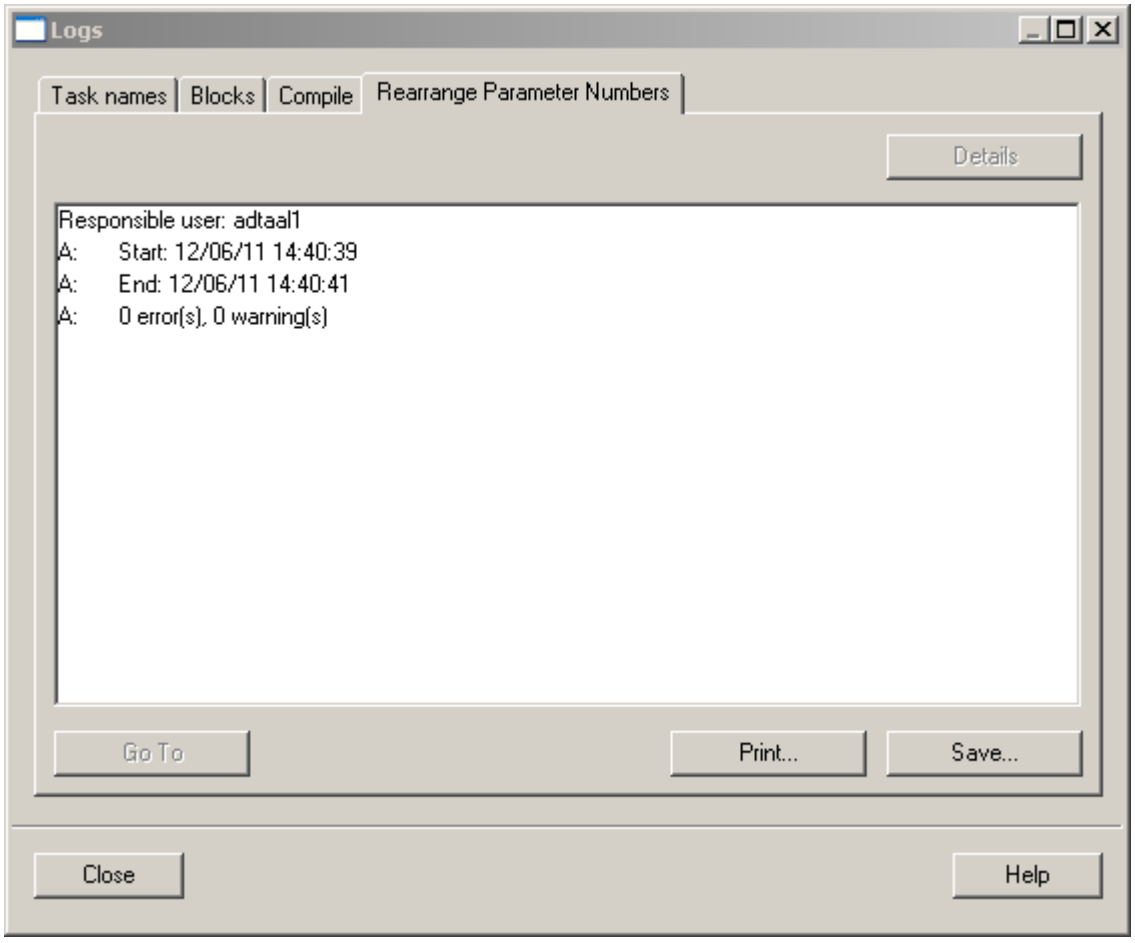


Figure 7-583 **Logs dialog box, Rearrange parameter numbers tab**

Publishing all connections

In the DCC editor, you can publish the connections of all blocks or the connections of one block.

For further information on the publishing of connections, refer to Creating customer-specific parameters ("declare") (Page 5543).

Publishing all connections of all blocks

Publish the connections of all blocks via the menu **Chart -> Publish all connections**.

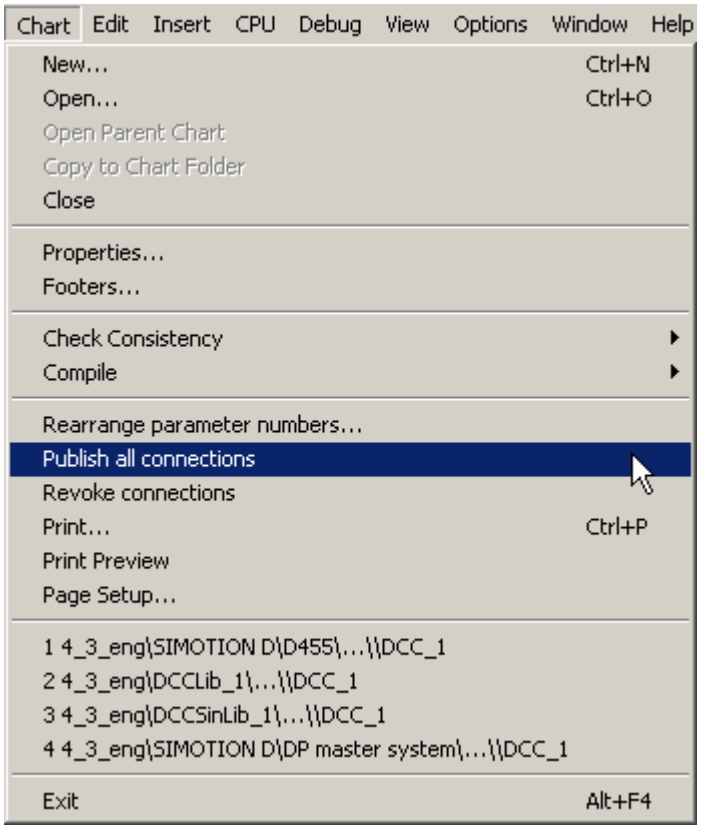


Figure 7-584 Menu **Chart -> Publish all connections**

Publishing all connections of one block

Select a block and publish its connections via the menu **Edit -> Publish all connections**.

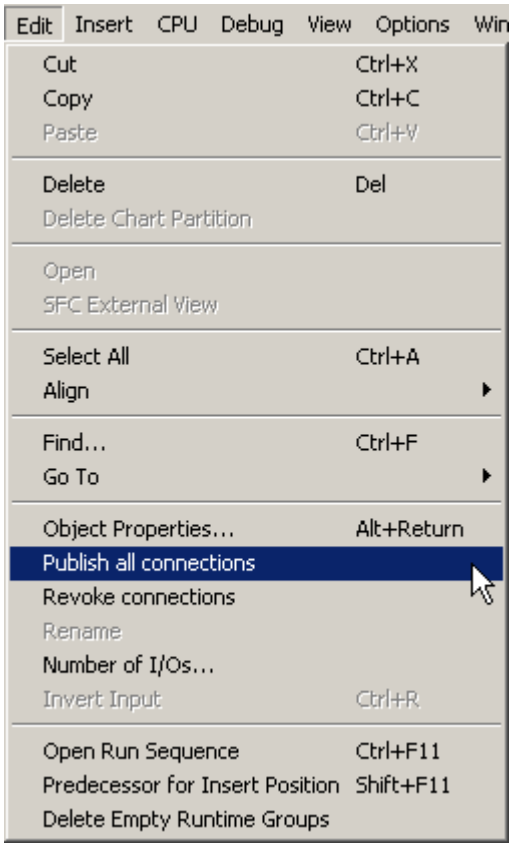


Figure 7-585 Menu **Edit -> Publish all connections**

You can also perform this via the block context menu.

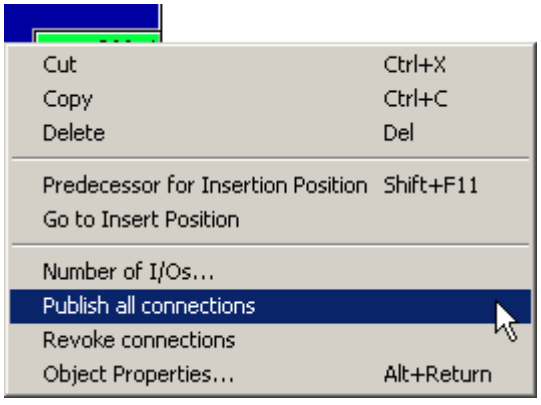


Figure 7-586 Context menu **Publish all connections**

Revoking connections

In the DCC editor, you can revoke the publication of the connections of all blocks and the publication of the connections of one block.

Chart -> Revoke connections

You can revoke the publication of the connections of all blocks via the menu **Chart -> Revoke connections**.

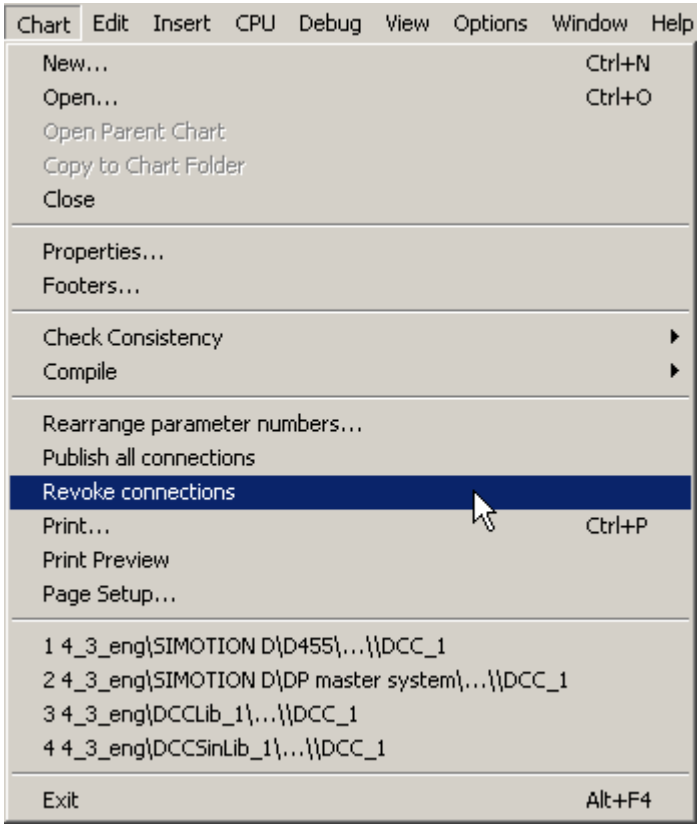


Figure 7-587 Menu **Chart -> Revoke connections**

Edit -> Revoke connections

You can revoke the publication of the connections of one block via the menu **Edit -> Revoke connections**.

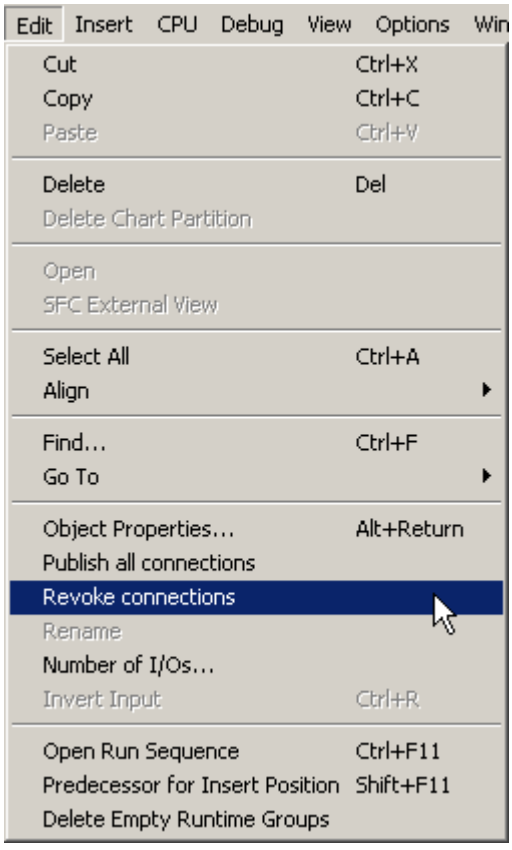


Figure 7-588 Menu **Edit -> Revoke connections**

This function is also available via the block context menu.

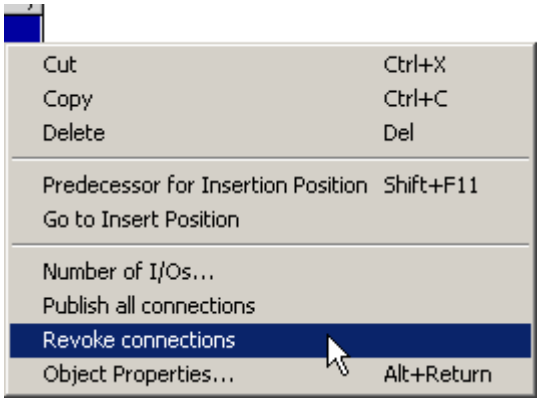


Figure 7-589 Block context menu **Revoke connections**

7.4.3.6 Compiling

Consistency check without code generation

General

Some of the conditions that have to be met if a valid configuration is to be created from DCCs can only be checked once charts have been created.

This procedure is carried out automatically at certain points, e.g. when a project or charts are compiled.

Performing a consistency check

The contents of DCCs can be checked at any time.

To check consistency, click **Chart > Check consistency > Charts as program...**

The **Logs** dialog box is displayed automatically after the consistency check. Here, errors are indicated by an "E" and warnings by a "W".

Error log

You can also display the result of the consistency check via **Options > Logs** in the **Consistency check** tab.

Note

If charts are deleted from previously compiled and downloaded projects, this can lead to inconsistencies during the next compilation/download, as the overall chart execution system has to be adjusted.

Note

Cross-chart consistency checks are performed only after the compilation of the charts.
The consistency check from the DCC chart always applies to the last compilation result.

Compiling the DCC in the DCC editor

Compiling

Please note that before the first compilation of a chart in a project, the project must first be saved in STARTER/SIMOTION SCOUT (via the menu command **Project** -> **Save** or with the button).

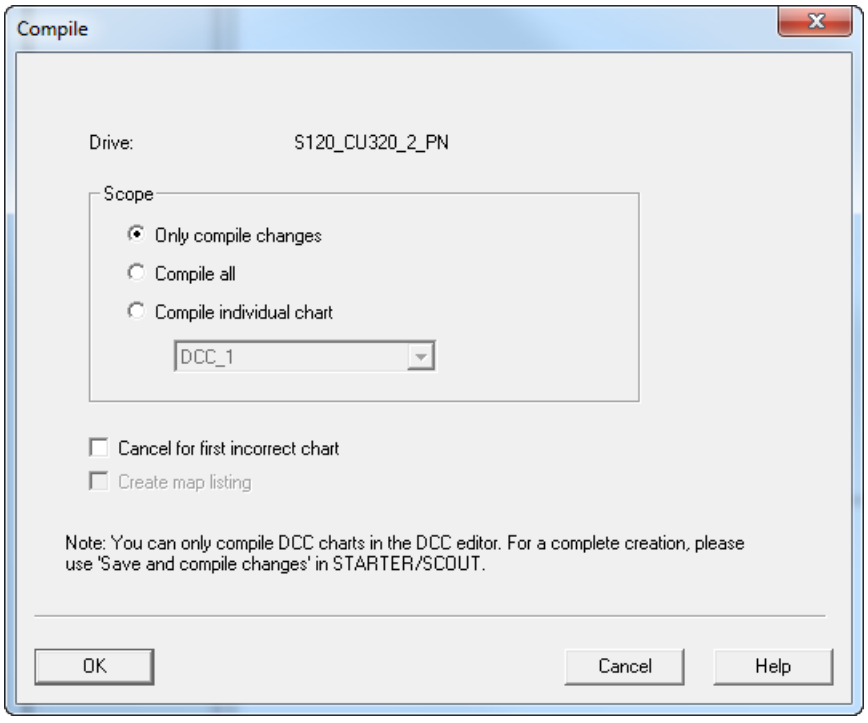


Figure 7-590 Compiling a DCC chart

You can begin compiling with **Chart > Compile > Charts as program ...** or with the button.

Compilation options

You can select the scope of the compilation:

- Scope: **Only compile changes**
  Only those parts of the configuration that have been changed since the last compilation are recompiled. When recompiling, this option reduces compilation time.

  Note

  All the configuration charts will be checked for consistency, even if you select the **Only compile changes** option.

- Scope: **Compile all**
  The entire contents of the configuration are compiled, regardless of whether the configuration has been changed since the last compilation.
- Scope: **Compile individual chart**
  The selected chart is compiled, regardless of whether the configuration has been changed since the last compilation.

Additionally, you can select whether a map listing is to be generated and whether it should be canceled in the event of an error occurring in a chart. A map listing is a list of the global objects, the cross reference and the blocks used in the chart.

The map listing is saved in subfolder U7\debug in the project path.

Note

You can only compile DCC charts in the DCC editor. To carry out the creation process in full, please use **Save and compile all** in STARTER/SCOUT.

As of STARTER/SCOUT V 4.2, this function is available under **Save and recompile all**.

As of STARTER/SCOUT V 4.3, this function is available under **Save and recompile all including DCC libraries**.

As of STARTER/SCOUT V 4.4, this function is available under **Save and recompile all.**

Reorganizing OCM variable interfaces

This option is only available for SIMOTION DCC charts. During the reorganization, the OCM variables are removed from the memory image that was already deleted previously. For further information, refer to HMI variables (Page 5589).

Error log

On completion of the compilation procedure, a detailed compilation log appears. Here, errors are indicated by an "E" and warnings by a "W".

- To navigate to the block that caused the error, select the error line in the log and click **Go to**, or double-click the error line.
- The log can be displayed again at a later point via the menu **Options > Logs** and, if necessary, can also be printed out.

7.4.3.7 Editing configurations further

Editing programs further: overview

This section describes more options for editing an existing program.

The following subjects are covered:

- Changing the block library
- Copy and paste charts or block groups between drive devices

Saving and restoring retain variables - SIMOTION only

DCB block instances may contain retain variables.

Select **Save Variables...** from the context menu of the DCC chart to save the contents of these variables to a file. The values can then be restored from this file. DCC retain data is retained when the platform is changed or the version is upgraded.

Changing the block library

If you want to transfer an existing configuration to a new SCOUT/STARTER version, you can change the library version at a later point. The blocks will be adapted accordingly.

Procedure

1. Open a chart from the relevant configuration and select **Options > Block types** in the menu.
2. The **Import** window is displayed. Click **OK** to close the window. In the window **Import DCB Libraries**, the block libraries installed on your programming device are listed under **Libraries installed in SCOUT/STARTER**. Under **Libraries imported in the chart**, all libraries which have already been imported into this configuration are listed.
3. Check which previously imported libraries have a new version installed, by inspecting the version of the library under **Libraries installed in SCOUT/STARTER** and **Libraries imported in the chart**.
4. Select the library to be updated under **Libraries installed in SCOUT/STARTER** and click **>>**.
5. Click **Accept**.
6. The progress of the update is displayed in a window.

Response

The changes made compared with the old version are transferred to your existing configuration.

Copying of charts or chart sections

In the STARTER and SIMOTION SCOUT engineering systems, charts can be copied within a drive object (SINAMICS) or device (SIMOTION), and between various SIMOTION devices or SINAMICS devices. It is also possible to select block groups within a DCC chart and insert them in other charts of the same device family using copy and paste.

Note

The copying of charts or block groups between SINAMICS devices and SIMOTION devices and vice versa is not supported.

You require a DCC license to copy a DCC chart as a block type into the SINAMICS library.

Note

Cross-chart interconnections

If you want to copy charts that contain cross-chart interconnections, this can lead to errors. Use the XML import/export instead.

Copying a chart (SIMOTION)

To copy a SIMOTION DCC chart, proceed as follows:

1. Select the source device from the project view.
2. Open the PROGRAMS subitem of the device.
3. Select an existing DCC chart and select the **Copy** command in the context menu of the chart.
4. Select the target device from the project view.
5. Open the PROGRAMS subitem of the device.
6. Select the **Paste** command in the context menu of the PROGRAMS subitem of the device.

The chart has been copied from the source device to the target device.

Copying a chart (SINAMICS)

To copy a SINAMICS DCC chart, proceed as follows:

1. Select the source drive unit in the project overview.
2. Open, for example, the Control_Unit subitem of the device.
3. Select an existing DCC chart and select the **Copy** command in the context menu of the chart.
4. Select the target drive unit in the project overview.
5. Open, for example, the Control_Unit subitem of the device.
6. Select the **Paste** command in the context menu of the Control_Unit subitem of the drive unit.

The chart has been copied from the source drive unit to the target drive unit.

Note

After copying, you must check whether the interconnections of block connections in the chart copy have to be adapted to another drive object as a result of the copying. During copying, the published block connections are automatically adapted to the basic system.

Cross-chart interconnections are lost.

DCC SIMOTION: When copying DCC charts, cross-chart interconnections are converted into textual interconnections. They are closed again via **Tools menu -> Close textual interconnections** .

Note

DCC SINAMICS and DCC SIMOTION

Only one SINAMICS DCC chart per drive object may be created.

Note

DCC SINAMICS and DCC SIMOTION

When copying a device, the associated charts are also copied.

An XML export or XML import of individual charts is possible. After the XML import, you must check whether the interconnections of block connections in the XML import have to be adapted to another drive object due to the XML import.

During a project export, the associated DCC charts are exported along with all interconnections.

Note

When copying DCC charts between different devices, the libraries used in the DCC charts are exchanged with the libraries present on the device. In the case of SINAMICS devices, this assumes that a version of the standard library is available for each device. As of SINAMICS 4.4, multiple versions of the standard library are available on a device for compatibility reasons. If the automatic selection option is used, the version with the highest firmware version is used. If multiple library versions are installed for a firmware version, the library with the next-highest version is selected.

For SIMOTION, *dcblibV2_0_simotion4_1_x* is also available on SIMOTION V4.2 devices.

V4.1 libraries are only available on device types that were also available in V4.1. V4.1 libraries cannot be executed on a D455-2 or D455-2. New block types can be found in *dcblibV3_0_simotion4_2*.

Copying of DCC charts to other projects

You can also copy DCC charts from one project to another. As only one project can be opened, you must start STARTER or SCOUT twice for this purpose and start another project.

Note

Please note that you can only copy and paste DCC charts via the engineering system; you cannot cut them.

If you want to copy a SINAMICS DCC chart to another project, then the chart must not use SINAMICS libraries that are only available in the source project. This applies to the libraries in the **SINAMICS LIBRARIES** subitem. In this case, an error message is displayed when the copy action is attempted, and the DCC chart is not copied.

You can use the functionality of the XML export and the XML import in order to always copy DCC charts correctly between projects.

Aborting copying processes with DCC charts

Errors may occur in the following cases when copying DCC charts:

- DCC charts were created using an older version of the DCC editor (CFC version)
- Copying without a DCC license

Copying block groups in the DCC editor

In the DCC editor, you can copy parts from one chart and insert them in another chart. To do this, you must open the source chart and the target chart in the DCC editor.

1. Use the lasso function to select the subsection of the source chart to be copied and select the **Edit > Copy** command in the menu bar.
2. Change to the target chart.
3. Select the **Edit > Paste** command in the menu bar.

The block group has been inserted in the target chart.

Block numbering

When copying blocks or block groups in the DCC editor, the name of the new block is formed as follows: if numeric characters are present at the end of the name, these are all deleted up to the first non-numeric character and replaced by the next free numeric character. It is therefore recommendable to allocate names consisting of letters (and numeric characters) such as block, block1, block2, etc.

Location of the insertion in the execution sequence

Blocks are always inserted into the execution sequence behind the selected block and the most recently added block is always selected automatically.
If a block is to be inserted behind an existing block in the execution sequence, select the block in the overview or in the chart, followed by the function **Predecessor for insert point** in the context menu.
You can change the execution sequence at any time via the **Edit > Execution sequence** command in the menu bar or the  button.

See also Specifying execution properties (Page 5437).

Deleting charts

Deleting a chart in SIMOTION SCOUT

To delete a chart in SIMOTION SCOUT, proceed as follows:

1. Open the SIMOTION SCOUT engineering system.
2. Select the required device in the project overview.
3. Open the PROGRAMS subitem of the device.
4. Select an existing DCC chart and select the **Delete** command in the context menu of the chart.

The chart has been deleted from the device.

Deleting a chart in STARTER

To delete a chart in STARTER, proceed as follows:

1. Open the STARTER engineering system.
2. Select the required drive unit in the project overview.
3. Open, for example, the Control_Unit subitem of the device.
4. Select an existing DCC chart and select the **Delete** command in the context menu of the chart.

The chart has been deleted from the drive unit.

Deleting higher-level elements

DCC charts are also deleted when a higher-level element (e.g. a DO) is deleted.

Search in the project from STARTER/SCOUT

You can use the sheet bar to search for variables and SINAMICS parameters in DCC charts in the open project. The contents of alias definitions can also be found using the search function.

Open the dialog via the menu **Edit > Search in project** or using the shortcut **Ctrl + Shift + F**.

The results are displayed in the search results tab of the detail view.

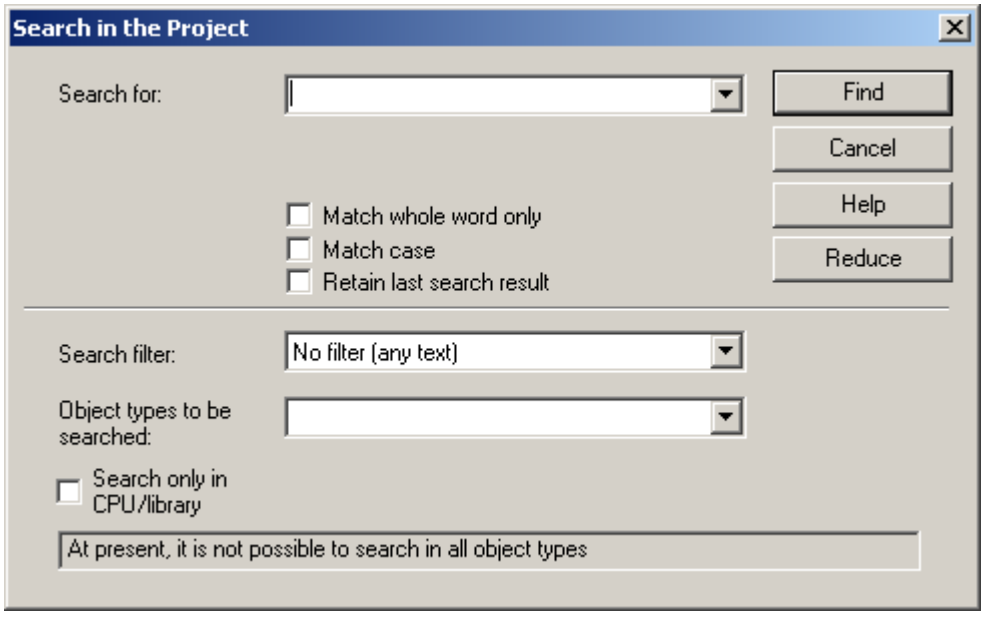


Figure 7-591 Searching in the project

Note

Symbols that are defined from the DCC, so-called @ parameters, DCC parameters (SINAMICS), links to BICO parameters are not considered when searching/replacing.

General conditions

- As of DCC 2.0.2, the symbols (variable in sheet bar) used in the DCC chart and the contents of alias definitions can also be searched for and replaced.
- The CFC editor must be closed during the search/replace operation.
- If the CFC chart sources have been deleted or the charts have know-how protection, searching/replacing of the sheet bar elements is not possible.
- It is not possible to undo these changes.

Replacing in the project

The **Replace in project** function is based on the **Search in project** function.

You can use the **Replace in project** function to quickly adjust the interconnections to the system after copying and inserting DCCs.

Open the dialog via the menu **Edit > Replace in project** or using the shortcut **Ctrl + Shift + G**.

When you carry out a replacement, both the results found and the replacement term are displayed in the **Search result** tab of the **detail view**. The text can be edited again here.

Use the **Replace** button to replace all search results selected using the check box.

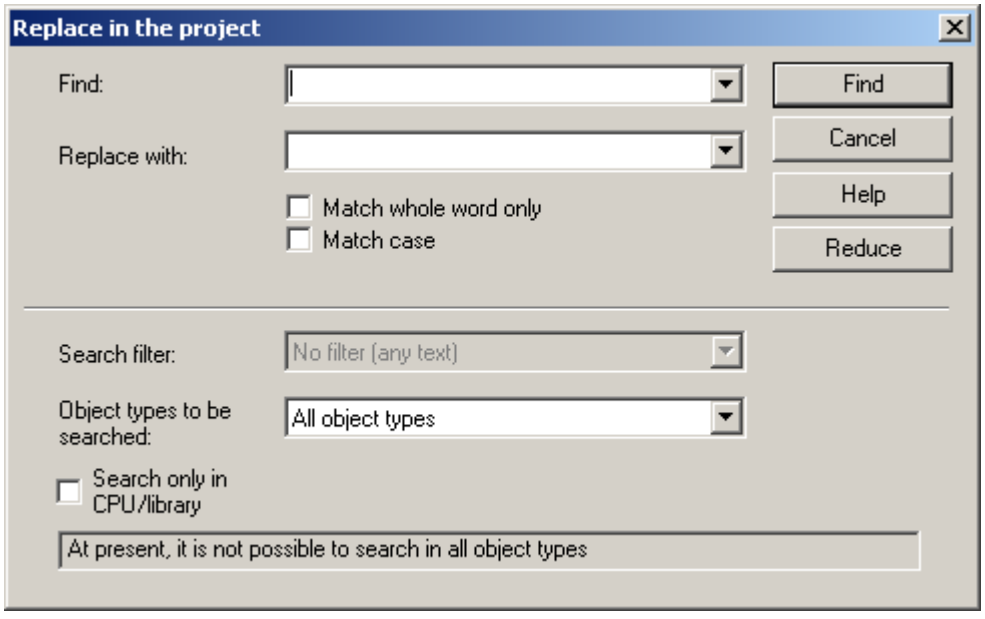


Figure 7-592 Replacing in the project

Note

Symbols that are defined from the DCC, so-called @ parameters, DCC parameters (SINAMICS), links to BICO parameters are not considered when searching/replacing.

Boundary conditions

- As of DCC 2.0.2, the symbols (variable in sheet bar) used in the DCC chart and the contents of alias definitions can also be searched for and replaced.
- The CFC editor must be closed during the search/replace operation.
- If the CFC chart sources have been deleted or the charts have know-how protection, searching/replacing of the sheet bar elements is not possible.
- It is not possible to undo these changes.

7.4.3.8 Test mode

Test modes

There are two types of test mode:

- **Process mode**
  Select this test mode if you want to monitor the behavior of individual instances, e.g. for the error analysis. When test mode is activated, all blocks are set to the status **Monitoring off**. In this test mode, you must select the relevant block connections and explicitly log them on for monitoring.
- **Laboratory mode**
  Laboratory mode is used for convenient, efficient testing and commissioning. When test mode is activated, all blocks are set to the status **Monitoring on**.

You can select the desired test mode in edit mode using the menu commands in the **Test** menu. Once a test mode has been selected it is not possible to switch between them.

Test settings

You can specify the **Monitoring cycle** via the menu **Test > Test settings**. This displays the **Test Settings** window.

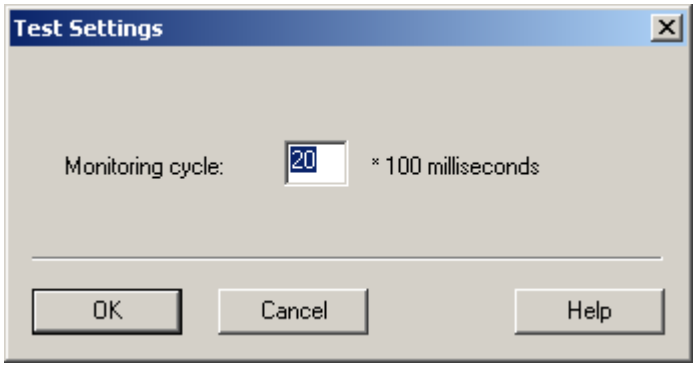


Figure 7-593 Test settings

In this window, you can set the **monitoring cycle** for the objects that have been logged on for testing, i.e. the cycle time for updating these objects. The monitoring cycles can be set in 1 to

100 steps each of 100 milliseconds (SIMOTION). Only whole seconds can be set as monitoring time for SINAMICS.

Note

If you have logged on many objects for the test, it is better to use a slower updating cycle.

Monitoring in laboratory mode

When test mode is activated, the **Monitoring on** function, or  , will also be activated for DCC charts in **Laboratory mode**.

This means that, in test mode, you can also display the development of the values for those block connections that are logged on for display, i.e. the values will be read out and displayed cyclically. You can change the options relating to this dynamic value display and the connection parameters in test mode.

Saving settings

The logging on of blocks for monitoring is rejected when the online test is exited. The settings of the block interfaces with regard to the test are saved in the project.

Display of values during the test

Current values of block interfaces are displayed when they are logged on for the test and their block is logged on for monitoring.

Monitoring in process mode

In **process mode**, the **Monitoring off** function, or  , is activated. This means that you must first select the blocks that you want to monitor (by highlighting them in the chart) before dynamic display can take place. You then need to execute the menu command **Monitoring on**.

Conversely, you can exclude individual blocks from being monitored if there are too many (highlight the block and execute the menu command **Monitoring off**).

Note

If the monitoring function has been deactivated and you highlight a connection in order to log it on for testing via  , monitoring is activated for this connection and for all previously logged-on connections of this block.

By activating test mode, connections are also established with the CPU for all connections listed within a value display window. Activate **monitoring** by selecting the individual connections in the **Monitoring** column.

Logging on/logging off connections for testing

In **edit mode** or **test mode** (process or laboratory mode), you can log on individual block or chart connections for testing:

- Highlight the connection and select **Test > Log on connections** or click the button in the toolbar.

When this is performed in **test mode**, monitoring is also activated, i.e. the connection is displayed with a yellow background and with its current value. If monitoring has been deactivated for this block, it will also be activated for all connections that were logged on for testing prior to this.

In **edit mode** or **test mode** (process or laboratory mode), you can log off individual block or chart connections for testing:

- Select **Test > Log off connections** or click the button in the toolbar.

Note

Alternatively, it is also possible to log on/log off all connections of a block for testing in edit mode by highlighting a block, selecting **Object properties** via the context menu, and selecting or deselecting the individual connections in the **For testing** column of the **Connections** tab.

Activating/deactivating connection monitoring

You can activate/deactivate monitoring (displaying the current values at the connection) of the block connections logged on for testing:

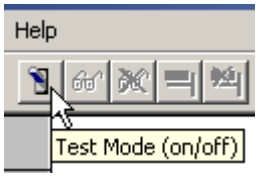
- Automatically, by activating test mode in laboratory mode.
- Via **Test > Monitoring on** or via the button in the toolbar. In both **laboratory mode** and **process mode**, this function concerns only the blocks previously selected in the chart.
- You can deactivate monitoring (meaning that values at the connections will no longer be updated) via **Test > Monitoring off** or by using the button. In both **laboratory mode** and **process mode**, this function concerns only the blocks previously selected in the chart.

All input and output values that have been activated for monitoring purposes are updated in accordance with the set monitoring cycle.

Activating test mode

To activate test mode, proceed as follows:

- Click the **Test mode** button in the toolbar



or click **Test > Test mode**.

Test mode is activated. The menu item is identified by a check mark. Depending on the selected test operating mode, the text **Test: RUN(laboratory mode)** or **Test: RUN(process mode)** is displayed in the status bar with a green background. Any menu functions that are not permitted in test mode are displayed as deactivated (grayed out).

Note

If the DCCs are different in the editor and in the target device, then they can behave differently, see Consistency of charts in test mode (Page 5470).

Requirements for starting test mode

The use of test mode requires an online connection to the device.

As long as the DCCs are identical online and offline, test mode is activated immediately. If the DCCs do not have the same version in the RT system and in the engineering system, values can still be monitored in test mode, but in order to make further changes the consistency must first be restored by uploading the changes from the target system (in this case the changes are not visible in the editor) or by downloading the current version from the engineering system.

Note

Once the project has been uploaded from the target device, it needs to be saved before DCC test mode can be started.

If you have made changes to connections or block types in the DCC, you will be informed that you can still monitor and trace values when test mode starts.

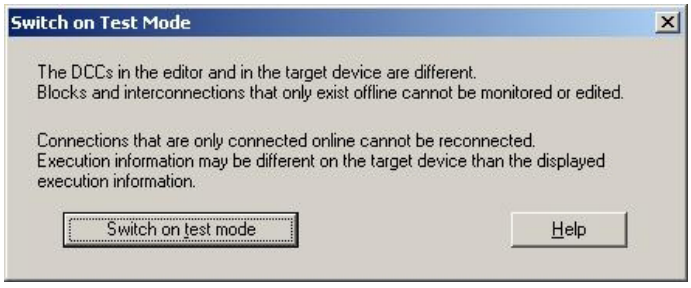


Figure 7-594 Test mode - changes to connections or block types

Online changes can only be performed after the DCC has been recompiled and downloaded to the target system. You are informed about this fact when starting test mode and can return to editing mode or activate test mode for monitoring purposes - as shown in the following dialog box.

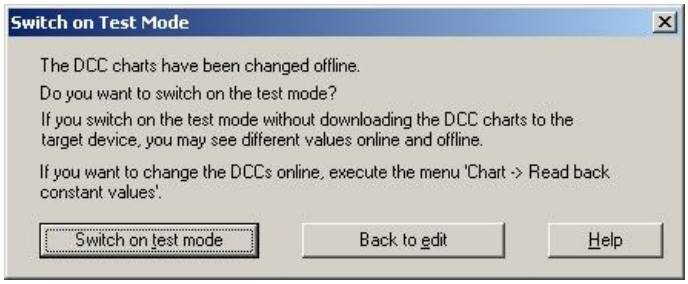


Figure 7-595 Test mode - loading changes to the target device

Monitoring test mode

In test mode, the values of the connections that have been logged on for monitoring are displayed with a yellow background.

Logging on connections for testing

You must explicitly log on to test the connections to be tested. For SIMOTION, the log on to the test can be performed either using the **Log on connection** button in the menu bar or using the context menu of the block connection. Select the **Log on connection** item in the context menu of the connection. For SINAMICS, go to the **Object properties** of the block connection (right-click) and activate the **For test** property. You can also log on the connections directly for the test by clicking with pressed Ctrl key.

Changing input values

You can also change all values of non-interconnected inputs in test mode. To show how changing a value affects execution, assign a new value to an input as follows:

- Double-click the input to be monitored.
- The **Properties - Connection** window appears. Enter the new value for the block input and confirm with **OK**.
- In the chart, you can now see how the value changes on the associated output.

Note

The changed value is only visible when the connection has been logged on for testing.

Logging off connections from testing

You can log off connections logged on for testing. The log off from the test can be performed either using the **Log off connection** button in the menu bar or using the context menu of the block connection. Select the **Log off connection** item in the context menu of the connection. You can also log off the connections directly from the test by clicking with pressed Ctrl key.

Enabling the value and trend display during a test

In test mode, you can use the value and trend display to analyze the input and output values of blocks.

Note

The dynamic display is limited to 256 values.

It includes the structure of a block connection with more than 256 single elements. As such, you can only insert individual elements selectively into the dynamic display and not the block connector in its entirety.

Drag individual elements to the dynamic display by means of drag-and-drop to display currently relevant values.

Please note that monitoring values in the dynamic display influences the performance of the target device

Enabling the value and trend display

You can open the value and trend display using the **View > Value display** and **View > Trend display** commands in the menu bar of the DCC editor. Each of these is an autonomous program window. The windows can be arranged using the minimize and maximize functions.

The block interfaces can now be added to the value or trend display using the **Insert in value display** and **Insert in trend display** commands in the context menu. Current values are displayed when the test mode is switched on.

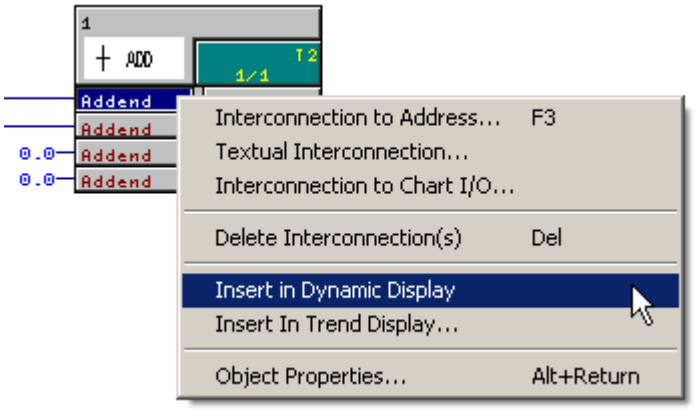


Figure 7-596 Inserting a block pin into the value or trend display

Settings in the Trend Display window

The **Trend Display** window displays the block connections added to the trend display in the form of curves. Each inserted block connection is called a **channel**, whereby a **lower limit** and an **upper limit** can be specified for each channel.

The number of sample values to be displayed on the time axis is specified in the **Display** area of the **Trend Display** window.

The desired trace parameters can be set in the **Trace Parameters** window, which can be accessed via the **Change** button.

Editing DCCs in test mode

To a large extent, you can continue to edit your configurations during test mode. An overview of the changes that can be made during test mode is shown below:

Table 7-609 Further editing of configurations during test mode

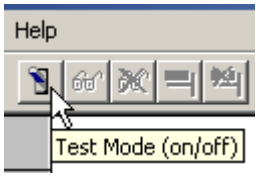
Handling blocks (Page 5434)	
Inserting blocks	Call the block catalog using the **View** > **Catalog** command. Open the block family and use drag-and-drop to move the selected block to the working area.
Deleting blocks	Select the block and use the **Edit** > **Delete** command to remove it. In DCC SINAMICS, blocks for which at least one connection has been published as a parameter cannot be deleted. The SAV, SAV_BY, SAV_D, SAV_I and STM blocks cannot be deleted or inserted online.
Moving blocks	Select a block and drag it to the desired position in the chart.
Renaming blocks	Select the block, right-click and select the **Rename** command. • The new block names apply only to the accesses to the input/outputs of the block using Java or Trace; the old names remain valid here. • For those blocks that use retain data, the retain data block on the device is also assigned the new name.
Interconnecting blocks (Page 5440)	
Creating interconnections	In the DCC, select the block connections between which you want to establish a connection. Note that in DCC SINAMICS, block inputs published as BICO parameters can only be connected to block outputs published as BICO parameters or BICO sources of the basic system. BICO inputs and BICO outputs of the basic system can still only be connected to block outputs and inputs published as BICO parameters in the DCC.
Deleting interconnections	In the DCC, select the block connection that you want to disconnect. Then select **Edit** > **Delete** to delete it.
Moving interconnections	In the DCC chart, select the block connection that you want to move. Then move it using drag-and-drop. When moving BICO interconnections, the signal can acquire the value 0 for a few cycles. The no longer interconnected input on the original connection is permanently assigned the value 0.
Changing the signal value of an input	Double-click the block input for which the value is to be changed. The "Properties - Connection" dialog box appears in which you can change the value.
Deleting global interconnections to the sheet bar	In the DCC, select the block connection that you want to disconnect. Then select **Edit** > **Delete** to delete it.
Comments (text)	

Inserting comments (text) in the chart	Select the **New text** command via the directories of the block types and insert it in the chart using drag-and-drop. Release the mouse button at the required point.
Changing comments (text) in the chart	You can move the text field in the DCC by selecting it and then dragging it to the desired position. You can change your comments by double-clicking the text field and then entering or editing your text.

Deactivating test mode

Deactivate test mode to return to edit mode.

- Click the **Test mode** button in the toolbar.



  or click **Test > Test mode**.

Edit mode is reactivated.

Changing online during test mode

Preliminary remark

As a general rule, test mode is used for making online changes to values, interconnections, and block instances without stopping the system.

It is not necessary to recompile after changes have been made online.

The changes are made in the target device and offline data storage area at the same time.

However, they must be saved with **Copy RAM to ROM** before Power Off , otherwise they are lost. This is especially important with regard to SAV blocks.

Note

Please note that online changes can only be made in test mode.

In test mode, the "Server busy" message may be intermittently displayed if you attempt to change values online.

You may need to acknowledge this message several times before the value that is being changed online is accepted.

Repair the project using an XML export/import.

Note

In SIMOTION SCOUT, deactivate the option **Options → Settings → Save → Automatic back-up copy of the project data** when working with large projects to prevent a drop in performance.

Otherwise, the entire project would be saved in test mode each time a change was made.

Changing values at block inputs online

Requirement

Only signal values at block inputs that are not interconnected can be changed online.

Procedure

The **Properties** - **Connection** window is opened by double-clicking the desired block connection. A new numerical value can now be entered for the block input in the **Value** line. The new value takes effect and is displayed in the chart when **Accept** is clicked. The window is closed by clicking **OK**.

Note

DCC-SINAMICS 2.5: Only signal values of block inputs can be changed online that have not been published as BICO parameters.

As of SINAMICS 2.6., all non-interconnected block inputs can be changed in test mode. They do not have to be published.

Note

DCC SIMOTION: The signal value at the block input can be changed online, but with active execution group is overwritten in the next cycle. Whether the signal value at the block input can be changed online does not depend on whether it has been declared as an HMI variable or not.

Deleting an interconnection online

Procedure

In the DCC, select the block connection that you want to disconnect. Then remove this with **Edit > Delete** or with the Del key.

Result

The connecting line between the connections disappears and the last value that was transferred on the connection appears as input value at the connection.

Note

DCC SIMOTION: Interconnections to chart connections cannot be deleted online!

Establishing an interconnection online

Procedure

In the DCC chart, select the block connection from which you want to establish an interconnection and drag an interconnection to the block connection to which the interconnection is to be established.

Result

The connecting line between the selected connections is established and the current value that has just been transferred appears at the output.

Note

Up to and including DCC-SINAMICS 2.5, a block input published as a BICO parameter may only be interconnected with outputs published as BICO parameters or with BICO outputs of the basic system.

Block connections cannot be published in test mode, i.e. new @ parameters inserted.

Note

DCC SIMOTION: Connections to published block inputs in the DCC or global operands cannot be recreated online.

Moving interconnections online

Procedure

Select the required interconnection and move it with drag&drop.

Note

When moving BICO interconnections, the signal can acquire the value 0 for a few cycles.

The no longer interconnected input on the original connection is permanently assigned the value 0.

Inserting a block online

Procedure

Call the block catalog using the **View > Catalog** command. Open the block family and use drag-and-drop to move the selected block to the working area.

The block instance is calculated with the next cycle.

The inserted block instance is assigned a standard name that can subsequently be changed online: Select the block, right-click and select **Rename**. Recompilation and loading is only required if the trend display, trace or monitoring of block connections has been activated.

Note

Whether a block can be added or deleted online is described under "Configuration data" in the online help for blocks.

Note

Renaming blocks

When blocks are renamed in online mode, a new block is created that replaces the existing block. This is then recalculated. The initialization values are newly set for the block.

Online renaming allows meaningful names to be assigned also for the online inserting of module instances.

Deleting a block online

Procedure

First delete all output interconnections by selecting the connection and then selecting **Edit > Delete** in the editor menu or with the Del key. Then select the block and remove it with the Del key or the **Edit > Delete** command.

Note

In SINAMICS, blocks for which at least one connection has been published as a parameter cannot be deleted.

Note

DCC SIMOTION: Blocks with interconnections to chart connections cannot be deleted online.

Note

The STM block cannot be deleted or inserted online.

Inserting comments in the chart

Procedure

Select the **New text** command via the directories of the block types and insert it in the chart using drag-and-drop. Release the mouse button at the required point.

Changing comments in the chart

Procedure

You can change your comments by clicking the text field and then entering or editing your text.

You can adapt the size of the text field to your requirements by selecting the black points at the corners and the sides and dragging them to the desired size.

Moving blocks in the chart

Procedure

Select the block and drag it to a suitable free position in the chart.

Consistency of the charts in test mode

As long as the DCC charts are identical in the DCC editor and in the target system, DCC charts can be monitored and changed in test mode.

If the DCC charts are different in the engineering system and in the target system, changes can still be made online as long as the DCC charts are not compiled.

If the charts are different in the editor and the target device and the charts have not yet been compiled, the following dialog box appears:

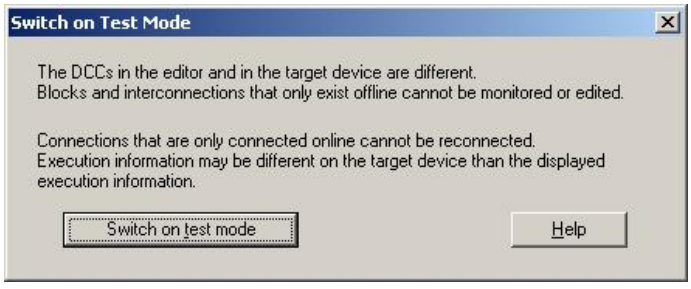


Figure 7-597 Activating test mode with inconsistencies

Click "Switch on test mode" to confirm the dialog.

If only the constant values of the chart are different in the CFC editor and in the target system, the user can recompile the DCC chart and transfer the changes from the target system to the CFC editor with **Options > Read back constant values**. Monitoring, tracing and further online changes can then be performed in test mode.

If the DCC charts differ online and offline and the DCC charts have already been compiled, a prompt appears that the DCC charts are different online and offline.

- If only the constant values of the charts are different, you can restore the consistency using the menu command **Options > Read back constant values**.

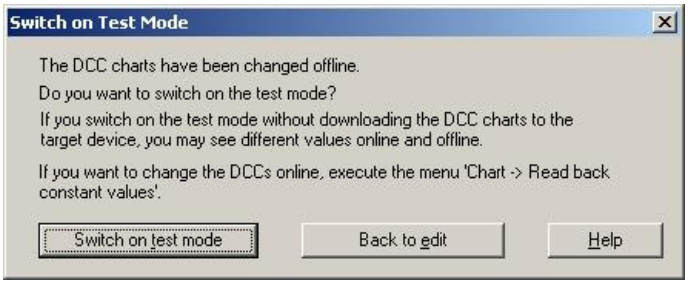


Figure 7-598 Activate test mode - Read back constant values

- If the DCC charts differ online and offline and have already been compiled, a prompt appears that the DCC charts are different online and offline. Download the changes to the target system so that monitoring and changes can be performed in test mode.
  The download is performed when the target device is in the STOP state. With SINAMICS, the download only functions in the "Power-on inhibit" and "Ready for power on" operating modes.

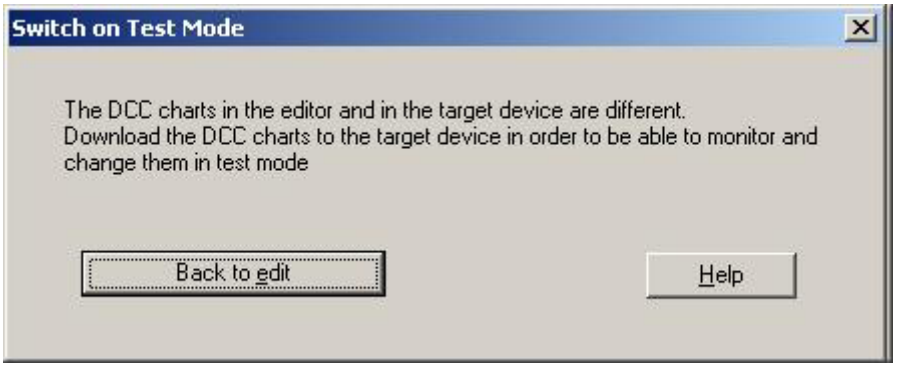


Figure 7-599 Activating test mode

7.4.3.9 Reference data

Chart reference data

As well as displaying the following information graphically in the DCC, you can use the "Chart reference data" (**Options > Chart reference data**) function to display it in the form of a list and print it out:

- Cross references of accesses to different objects
- Execution sequence
  Graphic representation of the entire run sequence of a CPU.

You can use this to check your configuration structure.

You can display and print out the following lists of chart reference data:

- Operand cross references
  This list displays all global operands used on the CPU along with the elements that access them.
- Execution group cross references
  This list displays the existing accesses of any DCCs for all execution groups.
- Block type cross references
  This list displays the block types used and the positions (on the DCC) where they are used.

Note

You can also generate the reference data of a DCC or a DCC library via the context menu with the **Reference data > Generate** command and then display it via **Reference data > Display**.

You can then always display the reference data again, but you only have to generate it once when first called or after changes to the @ variables.

List of block types

The list of block types shows where they are used. Unused block types can be deleted.

You can display the list by opening the **Chart Ref: Display Chart Reference Data** window via the **Options > Chart reference data** menu item. Now select the **View > Block types** menu item or use the  button in this window.

Column heading	Meaning
Block type	Block name (e.g. ADD)
Chart	Name of the DCC in which the block is used
Block	Name of the block instance (e.g. Integrator1)
Block comment	Block comment

Note

Double-clicking one of the displayed block types opens the associated chart and the selected block type is highlighted.

Cross References List Execution Groups (only for DCC-SIMOTION)

The list shows the existing accesses from arbitrary DCCs to the enable attribute of the execution group for all execution groups of the active CPU (the enable attribute can be used to switch individual execution groups on and off, see Enable attribute, execution groups (Page 5608)).

You can display the list by opening the **Chart Ref: Display Chart Reference Data** window via the **Options > Chart reference data** menu item. Now select the **View > Chart element cross references > Execution group** menu item or use the  button in this window.

Column heading	Meaning
Execution group	Group name
(R/W)	Read (R) or write (W) access

Column heading	Meaning
Chart	DCC name
Chart element	Name of the interconnected block and connection
Element comment	Block comment
Type	Block type (e.g. ADD)

Note

The names selected in the execution system (i.e. the names of the execution groups and execution levels) for each device must be unique.

List of operand cross references

This is a list of the global operands used on the CPU or the drive unit, along with the block connections that they are connected to.

You can display this list when you display the **Plan Ref: Chart Reference Data** window from the **Options > Open chart reference data** menu item. Now select the **View > Chart element cross references > Operand** menu item or use the  button in this window.

Table 7-610 Global operands

Column heading	Meaning
Symbol	Name of the global operand
Address	Blank
Data type	Keyword of the data type used
(R/W)	Read (R) or write (W) access
Chart	DCC name
Chart element	Name of the interconnected block and connection
Element comment	Block comment
Type	Block type (e.g. ADD)

7.4.3.10 Library handling

Library compatibility

Compatibility

Please refer to the following table for information on which kernel version supports which DCB libraries.

Kernel version	DCB lib version
SIMOTION 4.1.5	4.1.2, 4.1.4, 4.1.5
SIMOTION 4.2	4.1.5, 4.2

Kernel version	DCB lib version
SIMOTION 4.3	4.1.5, 4.2, 4.3
SINAMICS V2.5.SP1	SINAMICS V2.5
SINAMICS V2.6.x	SINAMICS V2.6
SINAMICS V4.3	SINAMICS V4.3
SINAMICS V4.4	SINAMICS V4.3, SINAMICS V4.4
SINAMICS V4.5	SINAMICS V4.3, SINAMICS V4.4, SINAMICS V4.5
SINAMICS V4.6	SINAMICS V4.3, SINAMICS V4.4, SINAMICS V4.5, SINAMICS V4.6
SINAMICS V4.7	SINAMICS V4.3, SINAMICS V4.4, SINAMICS V4.5, SINAMICS V4.6, SINAMICS V4.7
SINAMICS V4.8	SINAMICS V4.3, SINAMICS V4.4, SINAMICS V4.5, SINAMICS V4.6, SINAMICS V4.7, SINAMICS V4.8

SINAMICS libraries

The following standard libraries are executable with SINAMICS V4.8:

- SINAMICS V4.8 (dcblibV3_0_sinamics4_8)
- SINAMICS V4.7 (dcblibV3_0_sinamics4_7)
- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.7:

- SINAMICS V4.7 (dcblibV3_0_sinamics4_7)
- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.6:

- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.5:

- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V4.4, the following standard libraries are executable:

- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V4.3.x, the following standard libraries are executable:

- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V2.6.x, the following standard libraries are executable:

- SINAMICS V2.6 (dcblibV2_0_sinamics2_6)

With SINAMICS V2.5.SP1, the following standard libraries are executable:

- SINAMICS V2.5 (dcblibV2_0_sinamics2_5_1)

Handling DCC libraries and block types

The creation of SINAMICS DCC libraries is possible from DCC 2.1.

You can exchange block types within the DCC chart using the **Options -> Block types** menu command. If there are no DCC chart sources, the versions of the basic DCB libraries on the chart can be exchanged by selecting **Block types** from the context menu. This enables you to use the same DCC library on different device versions. The relevant DCC license is required for this.

Changes to execution groups are made automatically when saving under a different target device.

If a chart containing multiple execution groups is copied from the device to the library, error messages will appear during compilation. The user needs to correct the execution groups in this case.

It is possible that libraries and block types with the same names are present in the separate directories for the SIMOTION and SINAMICS libraries. Block type names must be unique within a library.

Note

Use the STM block in DCC library blocks that are only instantiated once per DO.

Importing block libraries

When first creating a chart in SCOUT/STARTER, you may be prompted to import a block library for the selected device platform.

The import process involves the DCB library that is already installed in SCOUT/STARTER being mapped to the Step7 data storage area for the device or DCC library, thus making it available for the charts in the DCC/CFC editor. DCC libraries located in the same project can also be imported for SIMOTION and SINAMICS devices.

You need to install libraries in SINAMICS/SIMOTION and then import them for the chart containers (**Programs** folder) before you can use the block types contained in DCC charts.

Only libraries that are appropriate for the device may be used. Only one version of a library should be imported. For a SIMOTION device 4.1, only libraries as of Version 4.1 + ServicePackX (e.g. 4.1.0, 4.1.5) should be imported.

Note

You can no longer use a SINAMICS 2.6 library as of SINAMICS Version 4.x.

If the DCC library required in the chart is not available for selection, this means that a basic library used within the library is not valid for the current device. You can change the basic library in the DCC editor under **Options > Block types**. If a chart source is not available, you can call up the **Block types** menu command from the context menu in the DCC library.

Name ambiguity in blocks from different libraries

If two blocks from different libraries (in the case of SIMOTION) have the same name, then the block from the library whose name comes first in the alphabet will apply.

If a DCC chart has not been inserted for a chart container (Device/Library), it is not possible to import a library.

Note

For SINAMICS, the libraries must always be imported under the device and not under the drive object.

Deleting the last chart of a device clears the selection of the DCB libraries.

When importing a DCC library, it must be compiled.

How to import block libraries:

- Open a chart from the relevant configuration and click **Options > Block types**.
  The **Import** window appears. Click **OK** to close the window. In the **Import DCB Libraries** window, the block libraries installed on your programming device are listed under **Libraries available for the device**. Under **Libraries imported in the chart**, all libraries which have already been imported into this configuration are listed.
- Select the library to be imported under **Libraries available for the device** and click **>>**. The import process is triggered when you click **Accept**.

Checks are performed when a block library is imported.

The names of the block libraries are defined on the basis of a naming convention. The individual parts of the name are explained in the following table. The part in question is shown in bold each time.

Table 7-611 Naming convention for block libraries

Part of the block library name	Meaning
TPdcblib_SIMOTION_4_1_2.2.0 [7.0]	Library identifier
TPdcblib_**SIMOTION**_4_1_2.2.0 [7.0]	Target platform

Part of the block library name	Meaning
TPdcblib_SIMOTION_**4_1_2**.2.0 [7.0]	Target system version
TPdcblib_SIMOTION_4_1_2.**2.0** [7.0]	Library version
TPdcblib_SIMOTION_4_1_2.2.0 **[7.0]**	Build version

Updating the block library

When a DCC chart is open, you can update the block libraries via the menu **Options > Block types**.

For DCC libraries without DCC chart sources, the versions of the basic DCB libraries can be exchanged by selecting **Block libraries** from the context menu.

Note

The library concerned must be selected for this purpose.

How to update block libraries:

- Open a chart from the relevant configuration and click **Options > Block types**.
- The **Import** window is displayed. Click **OK** to close the window. In the window **Import DCB Libraries**, the block libraries installed on your programming device are listed under **Libraries installed in SCOUT/STARTER**. Under **Libraries imported in the chart**, all libraries which have already been imported into this configuration are listed.

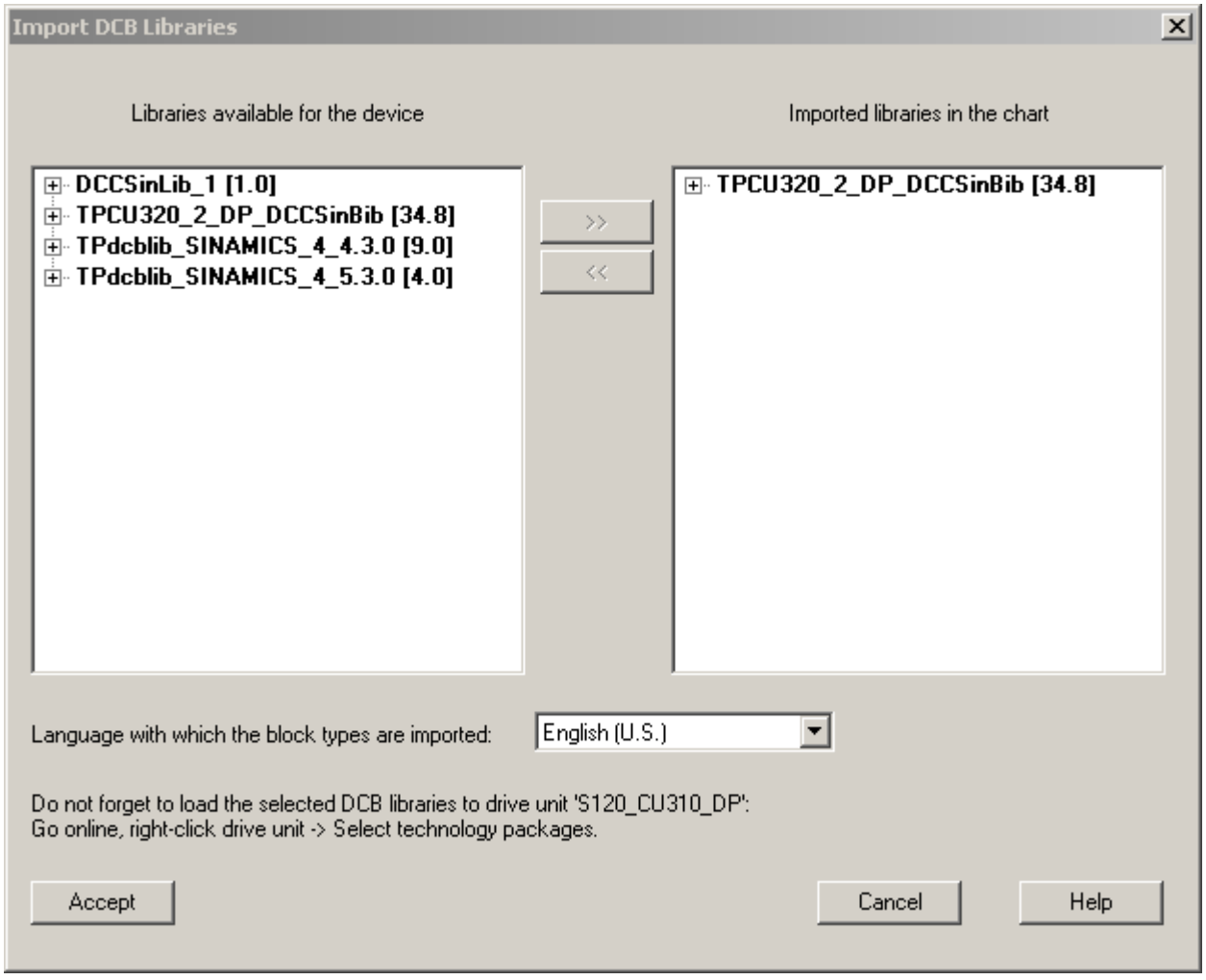


Figure 7-600 Updating the block library

Updating libraries after a device upgrade

As of STARTER 4.2, the system automatically swaps in the correct library version after a device version has been upgraded.

Detailed information about upgrading a device version can be found in the section titled Software upgrade and module exchange (Page 5512).

Note

The standard library is automatically installed during setup, but if you do uninstall it accidentally, you can reinstall it by means of the DCC setup process.

The libraries for SINAMICS can also be installed via the SSPs (SINAMICS Support Packages).

If you wish to reinstall the DCB library from the SSP DVD, switch to directory CD_1\SSP\Disk1 and unzip file dcblib.zip into any directory. Install the library dclbib_Vx.y_sinamics_w.z contained in unzipped files.

As of DCC V2.1 (STARTER 4.2), the libraries for SINAMICS are also available via the menu command **Options -> Install libraries**. See also Installing and uninstalling DCB libraries (SINAMICS).

The libraries for SIMOTION are also available via the menu command **Extras -> Install libraries**. See also Installing and uninstalling DCB libraries (SIMOTION) (Page 5496). Here you will also find information on where installable DCBLIB standard libraries are located and how to reinstall these.

- Check which previously imported libraries have a new version installed, by inspecting the version of the library under **Libraries installed in SCOUT/STARTER** and **Libraries imported in the chart**.
- Select the library to be updated under **Libraries installed in SCOUT/STARTER** and click **>>**.
- Click **Accept**.
- The progress of the update is displayed in a window.

Note

In SINAMICS, you can check a selected device in the engineering system using the **Select technology packages** function in the context menu as to whether all required block libraries and technology packages have been activated.

Note

In SIMOTION, the library is automatically loaded to the device during the project download. In SINAMICS, this action must be performed explicitly by the user before the project download. (Page 5567)

The library to be imported is checked to see whether it contains a block type that has already been imported with another library.

Error log

If errors are detected when a library is being imported, an error log with details of the causes of the errors is displayed.

Exchanging the basic library version for installed libraries

The basic libraries are selected from the DCC charts used in the dialog **Options → Block types** of the DCC editor.

If there are no DCC sources available at the DCC charts, you can open a dialog via the context menu **Block types** of the DCC chart, where you can exchange the block libraries used.

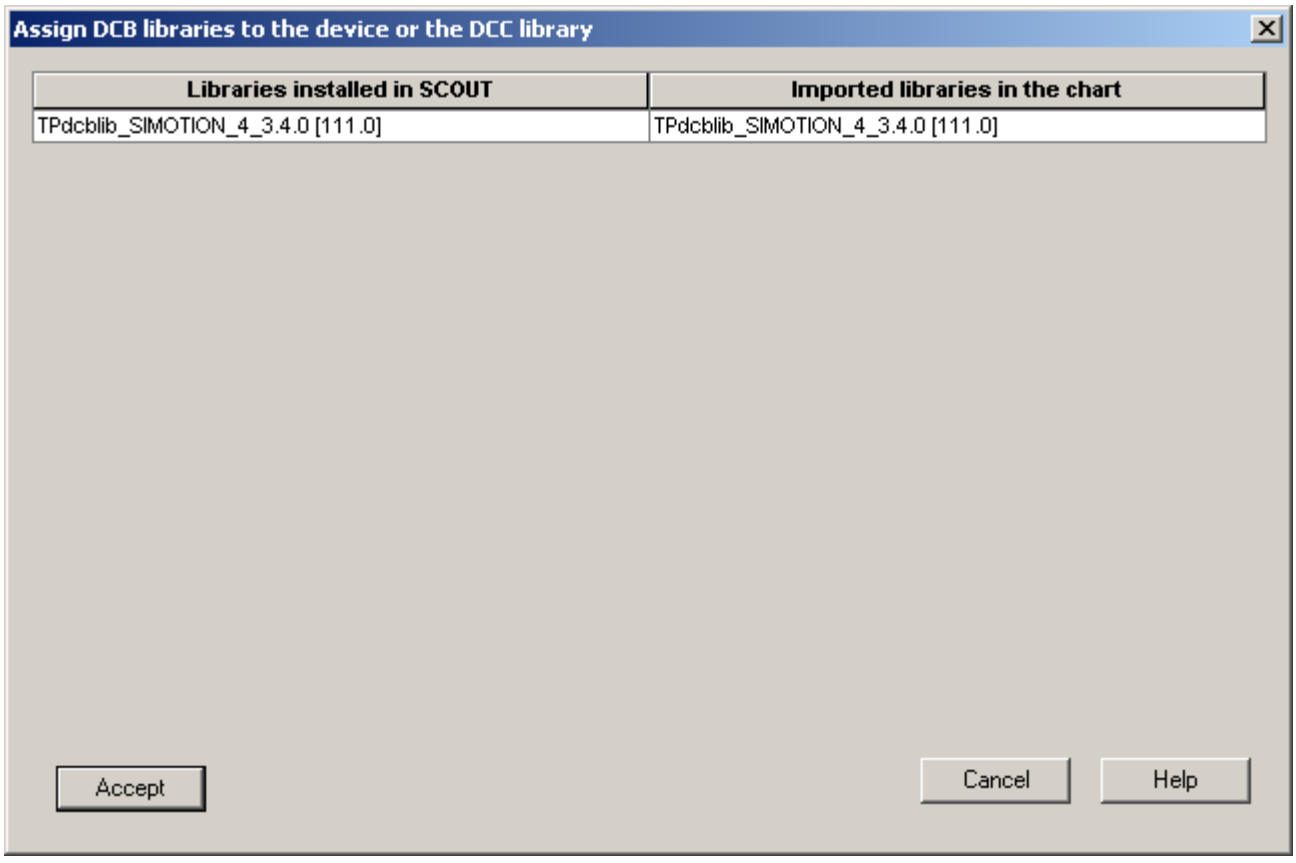


Figure 7-601 Block types

Mark the library in the left-hand column and press **Accept.**

Up to DCC V2.0.1, the version of the basic library used is permanently defined in a DCC library.

Exactly the same basic library version that was used to create the DCC library must be installed.

If the library sources have been deleted, it is not possible to select a new version of the basic library in the typical library.

With DCC 2.0.2, you can combine an installable library that has been supplied with different versions of a DCB library that is being used, provided the interface for the blocks called does not change.

Changing the block library language

You can customize the language of the block type comments via the menu **Options > Block types**.

How to change the language of the block types in a block library:

Checks are performed when a block library is updated.

- Open a chart from the relevant configuration and click **Options > Block types**.
- The **Import** window appears. Click **OK** to close the window.

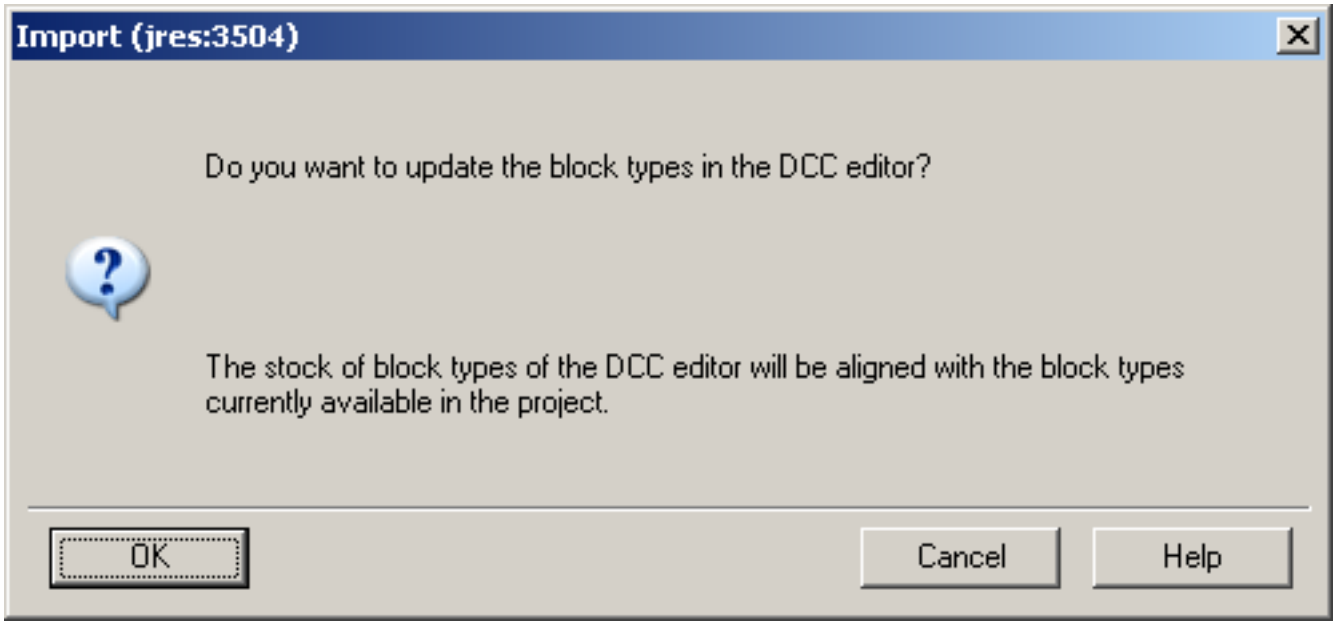


Figure 7-602 Import window

- The **Import DCB Libraries** window appears. The block libraries installed on your programming device are listed under **Libraries installed in SCOUT/STARTER**. Under **Libraries imported in the chart** all libraries which have already been imported into this configuration are listed.
- Select the desired language in the selection list of available languages. The selected language affects all imported libraries.

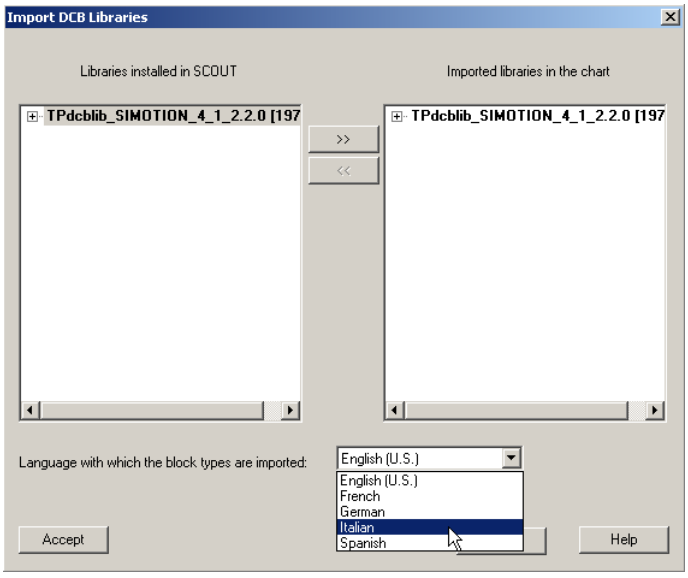


Figure 7-603 Changing the block library language

- Click **Accept**.
- The progress of the update is displayed in a window.

Note

Comments regarding the block types remain in the language in which they were created. The comments regarding block pins are translated if they have not already been altered manually.

Removing block libraries from the configuration

You can delete block libraries whose block types are no longer required in the configuration concerned via the menu **Options > Block types**.

How to delete block libraries:

- Ensure that the block types from the block library to be deleted are no longer to be used at all in the charts of the configuration.
- Open a chart from the relevant configuration and click **Options > Block types**.
- The **Import** window appears. Click **OK** to close the window.
- The **Import DCB libraries** window appears. The block libraries installed on your programming device are listed under **Libraries installed in SCOUT/STARTER**.
  Under **Libraries imported in the chart**, all libraries which have already been imported into this configuration are listed.
- Select the library to be deleted under **Libraries imported in the chart** and click **<<**. The deletion process is triggered when you click **Accept**.

Checks are performed when a block library is deleted. All unused block types of the library will be removed from the selection list of the block types in the DCC editor; used block types remain in the list. The chart cannot be compiled without an imported library if instances of the library are still present in the chart.

7.4.3.11 Creating block libraries

Fundamentals

The DCC editor offers a function for saving a chart as a **block library (DCB library = typical)**. This kind of chart relates to a connection created by the user, which features an interface so that it can be reused and is saved in a library. The charts as block library (DCB library) function therefore include the option of know-how protection, since the configured DCCs are only available as a transparent block within a block library. The DCC configuration that has been created is hidden. The DCC chart created for a block can be permanently deleted via the function **Delete source of the block types...** in the context menu of the library. This command is only available if all blocks in the library have been compiled without errors. If it is executed, all the corresponding chart sources are deleted leaving only what has been compiled in the project. You cannot undo the action. By right-clicking on the DCC chart and selecting the **Block types** command, however, you can open a dialog in which the block libraries used can be exchanged.

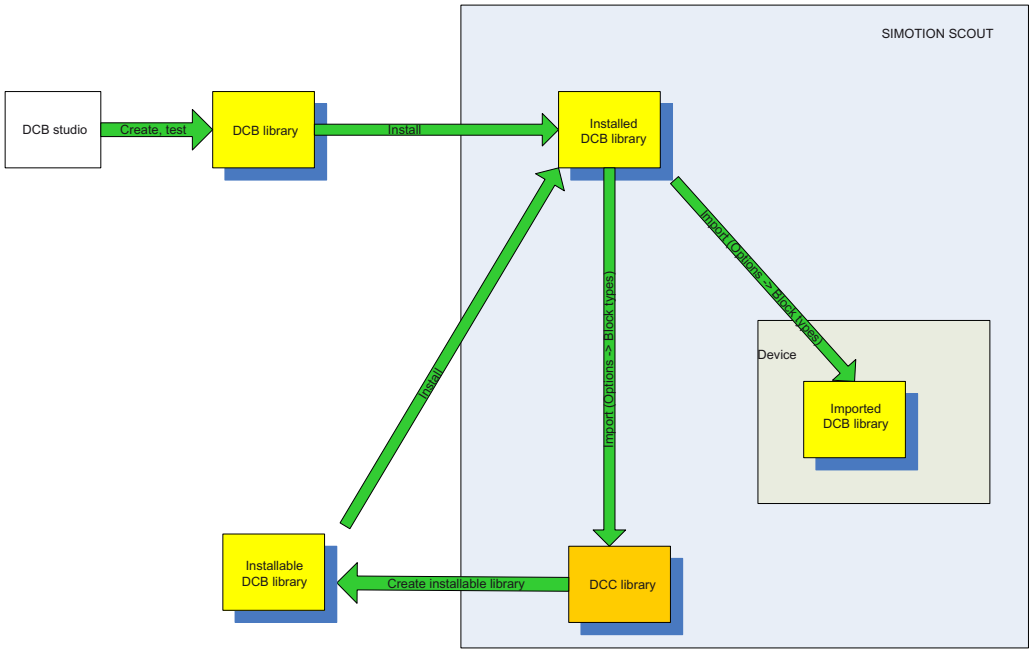


Figure 7-604 Installation of DCB libraries

Requirements

You have already created a new DCC library in the SIMOTION SCOUT engineering system. SIMOTION DCC libraries are located in the folder **Libraries** of the project navigator, DCC libraries in the **SINAMICS Libraries** folder of the project navigator.

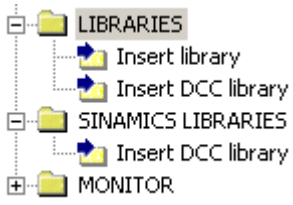


Figure 7-605 Inserting a DCC library

There is already a DCC available for saving in the DCB library.

The chart connections are displayed.

Note

You can display the chart connections via the button in the toolbar.

You can use a library either in DCC charts for SINAMICS or for SIMOTION.

Create DCC libraries for SIMOTION in SCOUT in the **Libraries** container.

As of DCC 2.1 / SCOUT / STARTER V4.2, DCC libraries can also be created for SINAMICS.

For this purpose, the project navigator contains a new program container for SINAMICS libraries under **Libraries -> SINAMICS LIBRARIES**.

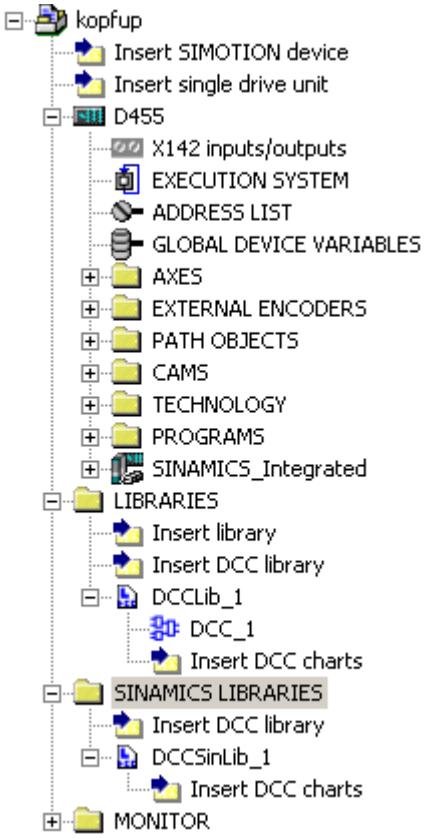


Figure 7-606 Library container for DCC libraries for SINAMICS devices in SIMOTION SCOUT

The DCC libraries for SINAMICS and SIMOTION are distinguished from one another on the basis of the different library containers in the project navigator.

Insert a DCC library for SIMOTION by selecting **Libraries -> Insert DCC library** from the context menu in the project navigator.

Insert a DCC library for SINAMICS by selecting **SINAMICS libraries -> Insert DCC library** from the context menu in the project navigator.

With DCC 2.0, you can insert SIMOTION DCC charts into the library container using copy and paste.

The chart connections of the inserted DCC chart must be connected to the chart connections of the SINAMICS DCC library without BICO interconnections and @ parameters before the compilation.

The DCC libraries, but not the DCB block libraries, are found in the library containers.

By selecting **Generate DCB library...** from the context menu for the selected DCC library, you can create installable DCB libraries from DCC libraries (typical libraries).

It is possible that libraries and block types with the same names are present in the separate directories for the SIMOTION and SINAMICS libraries.

Note

Know-how protection

Protect the DCC blocks, sources and libraries by means of the know-how protection function provided by STARTER/SCOUT.

Note

You need a DCC SIMOTION license to create and edit SIMOTION DCC libraries.

You need a DCC SINAMICS license to create and edit SINAMICS DCC libraries.

Rules for creating DCC libraries

- It is only possible to compile charts in DCC libraries.
- The chart may be hierarchical (chart-in-chart).
- Interconnections between charts in the library are not permitted.
- No HMI variables, BICO interconnections, @ variables or @ parameters must be configured at the chart connections of the basic chart and the blocks it contains.
- All blocks may only by integrated into the same execution level / execution group. Here, the chart blocks must directly follow one another.
- Inside the chart folder that needs to be compiled, all of the blocks from a basic chart must be placed in the execution group with the same name as the chart.
- Global operands are not permitted, but must be modeled as an input or output as chart connection.
- For the DCC library and its connections, please select only names starting with lower-case letters.

Procedure

Model your DCC in the DCC editor.

Insert an **ADD**er and a PI controller (**PIC**), for example, in the chart and interconnect them. See Interconnection to chart connections (Page 5441)

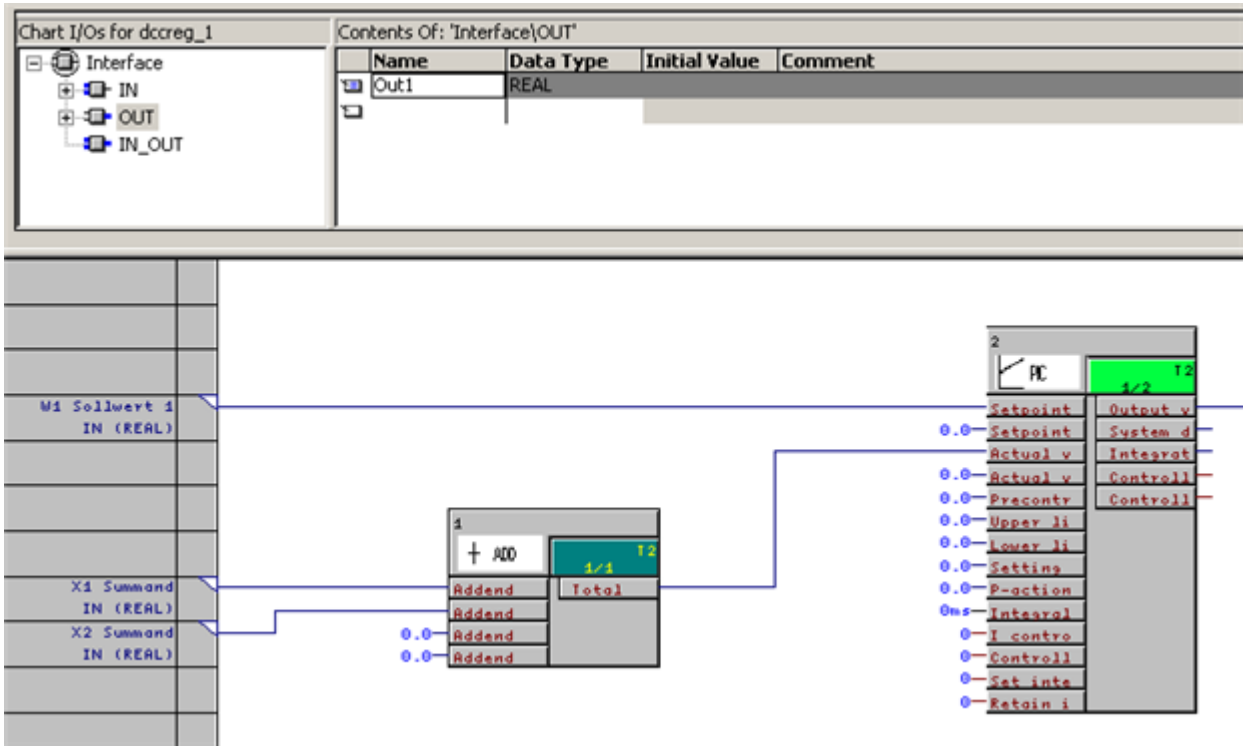


Figure 7-607 Example: Creating a chart as a program

Following successful configuration, the DCC can be compiled as a program. To do this, select **Chart > Compile > Chart as program** in the menu.

The newly created DCB library is available in the menu under **Options > Block types** in the DCC editor. The DCB library is not shown in SIMOTION SCOUT/STARTER; only native libraries are listed there.

See also

Creating an installable DCB library from DCC libraries (Page 5491)

Inserting and programming block types in DCC libraries

Within a library, a DCC chart can be created for each DCC block in the library. A new block type is created using the **Insert DCC chart** context menu.

SIMOTION and SINAMICS libraries are both subject to the same rules for assigning DCC chart names:

- Max. 22 characters, beginning with a letter
- If an underscore is used, only numbers may be used after it.

The following rules apply to the blocks:

- There is no restriction on the number of blocks for a DCC library.
- The name of the block type is formed on the basis of the chart name
- The block inputs and outputs are created in the DCC editor using **View -> Chart I/Os** and interconnected with the inputs and outputs of the block instances
- Within a library, all the blocks must be within the same execution group. This group bears the name of the chart
- The execution group in which the blocks run in the target device depends on the execution group of the block instance that calls the library block.

Special features of the 1:N interconnection of chart connection inputs

If you have defined an interconnection of a chart connection input with blocks for a chart, there are two different scenarios. First, there is the scenario of a 1:1 interconnection, which can be carried out without problems. However, the following must be noted for a multiple interconnection (1:N) of a chart connection input to several blocks.

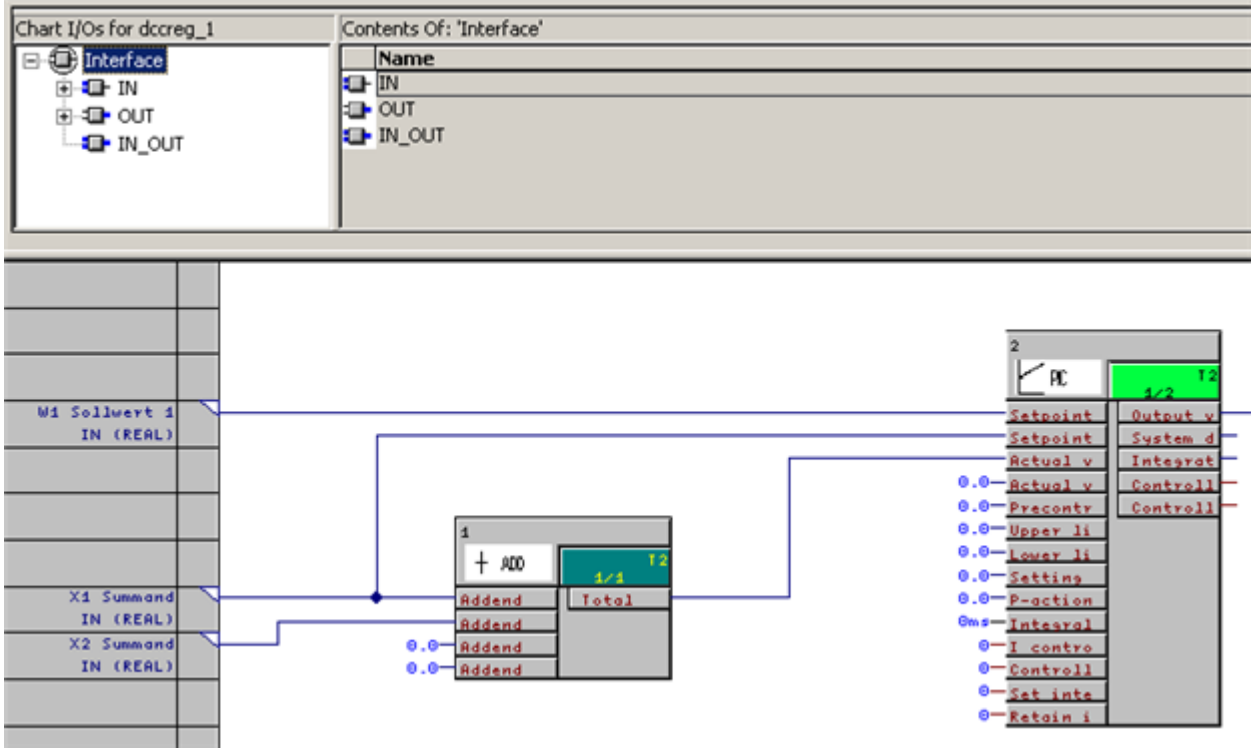


Figure 7-608 Connection example for a 1:N interconnection without NOP block

If there is a 1:N interconnection at an input within a hierarchical chart or block type, an **NOP_x** block must be used in the chart for consistency reasons - refer to the following example:

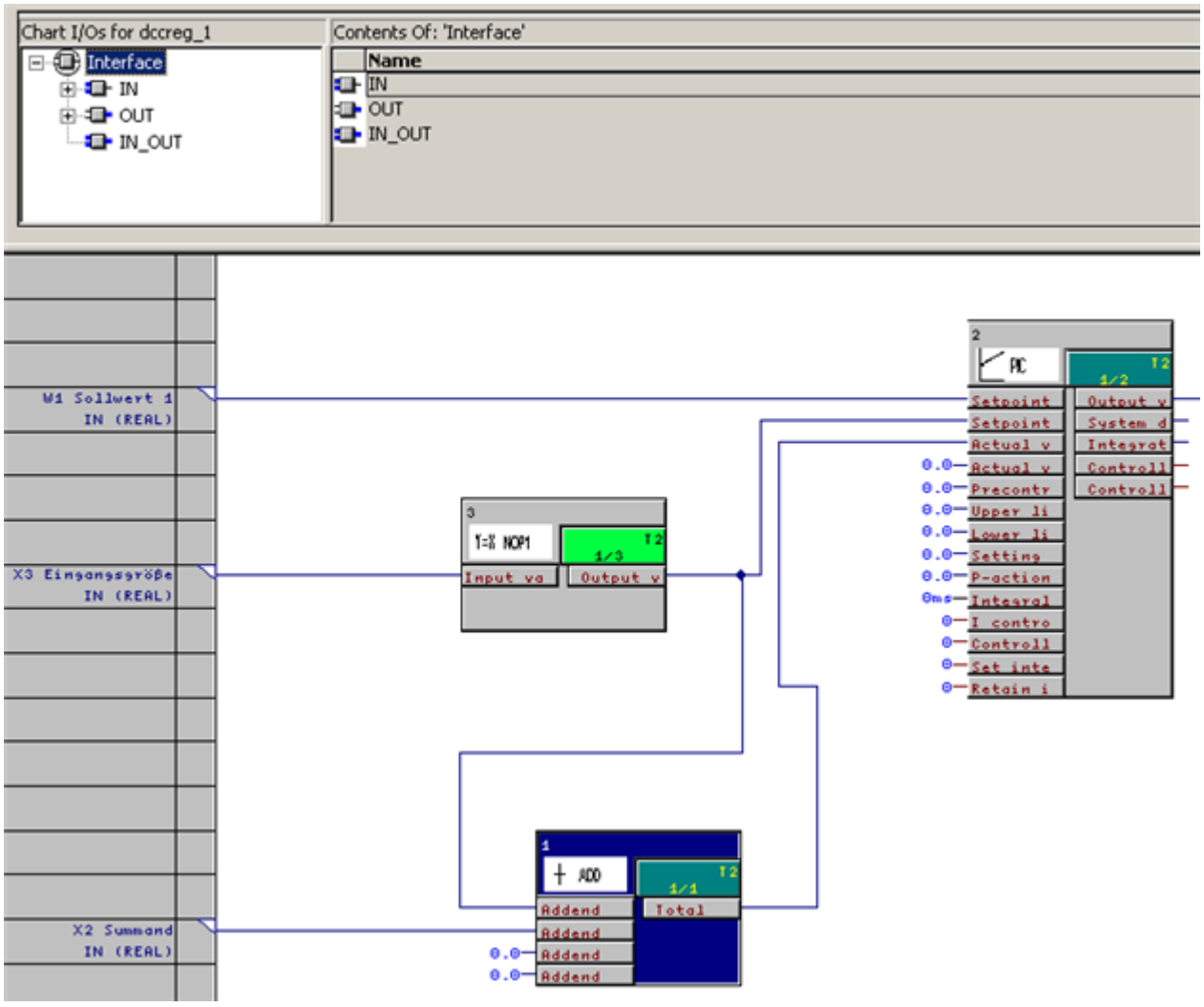


Figure 7-609 Connection example for a 1:N interconnection with NOP block

Creating comments and icons for DCC libraries you have created yourself, and assigning a block family to them

In the engineering system, you can save additional information on individual blocks of a DCB library that you have created yourself.

Procedure

Select the block library you want to edit in the engineering system.

In the context menu of the DCC library, select the **Block type properties** command.

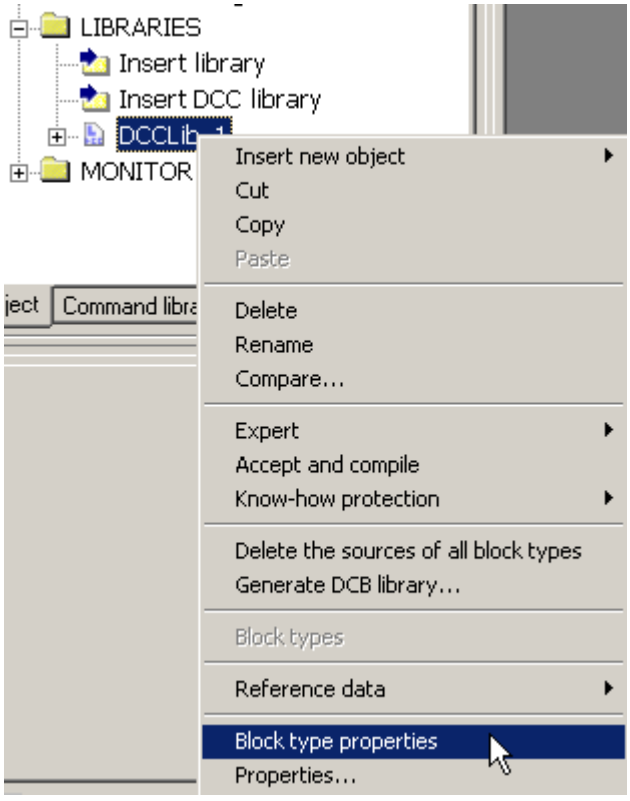


Figure 7-610 The **Block Type Properties** dialog opens.

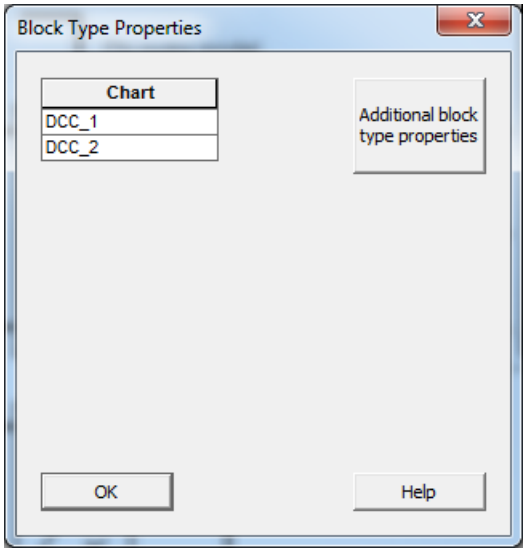


Figure 7-611 Block Type Properties

In the list under **Chart**, select the DCC chart you wish to edit and click the **Additional block type properties** button to open the **Block Type Properties** dialog.

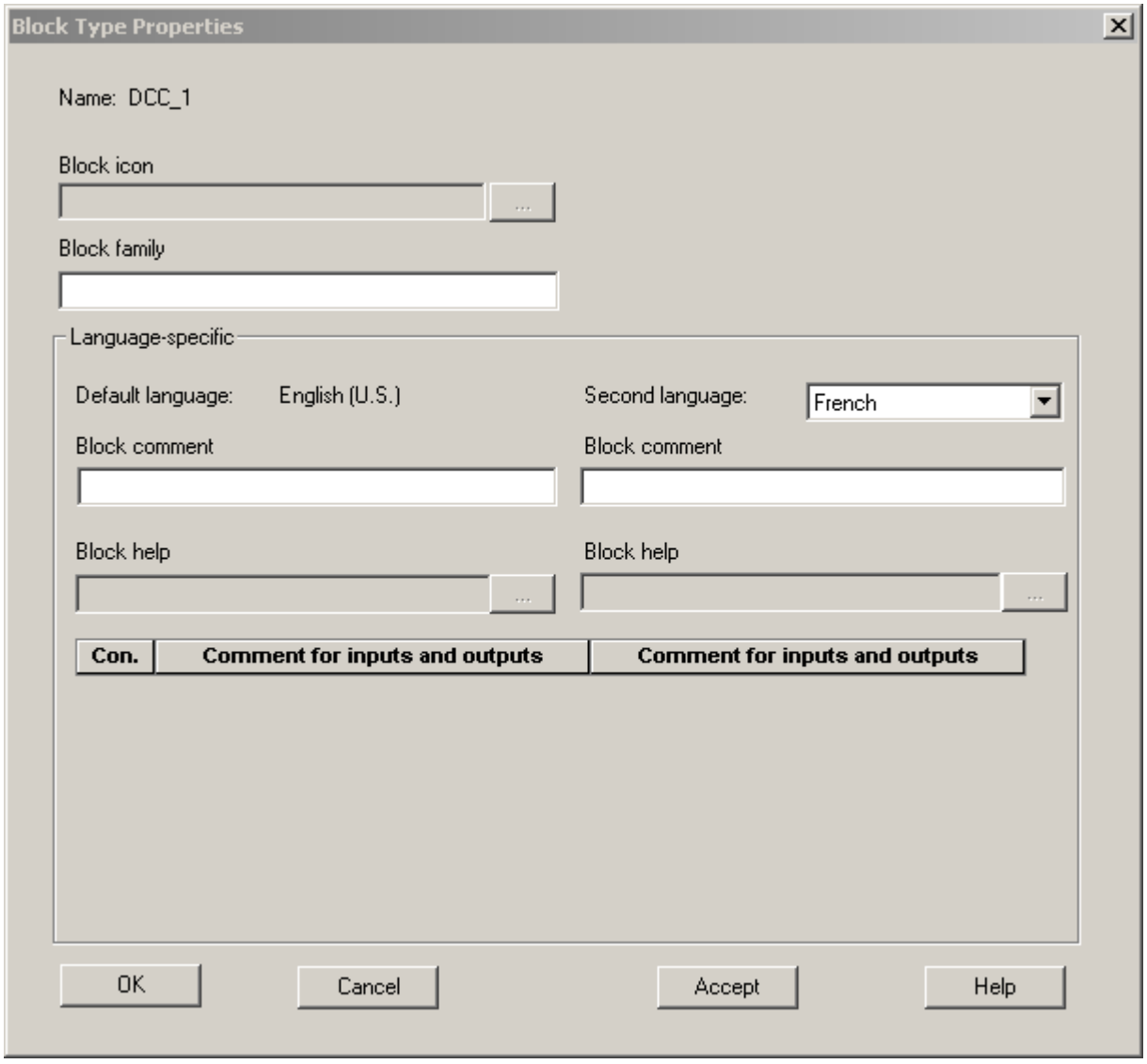


Figure 7-612 Block Type Properties

Block icon

Click the **...** button and load a BMP file you wish to use as a block icon for the DCC chart.

Block family

Enter the family to which you want to assign the DCB library. You can either choose an existing block family (e.g. Logic, System, Conversion, Arithmetic, etc.) or enter a new one, which will then be created automatically. The default language for block families is English.

Block comment

Here you have the option of entering a comment for your DCB library; this will appear in the block header. On the left-hand side, enter the comment in English (the default language). On the right-hand side, you have the option of entering the comment in additional languages. To do this, select the required language from the combo box above and to the right of the field, enter the comment, and click Accept. Repeat this process for all other languages in which you wish to create the comment.

Accept what you have entered using the **Accept** button and click **OK** to close the dialog.

Creating an installable DCB library from DCC libraries

Up to SIMOTION V4.1.1, a DCC library had to be transported as project. As of SIMOTION V4.1.2 / STARTER 4.2, it is possible to create an installable DCB library from a DCC library.

To do this, select the entry **Create DCB library...** in the context menu of the library; the **Create DCB Library** dialog appears.

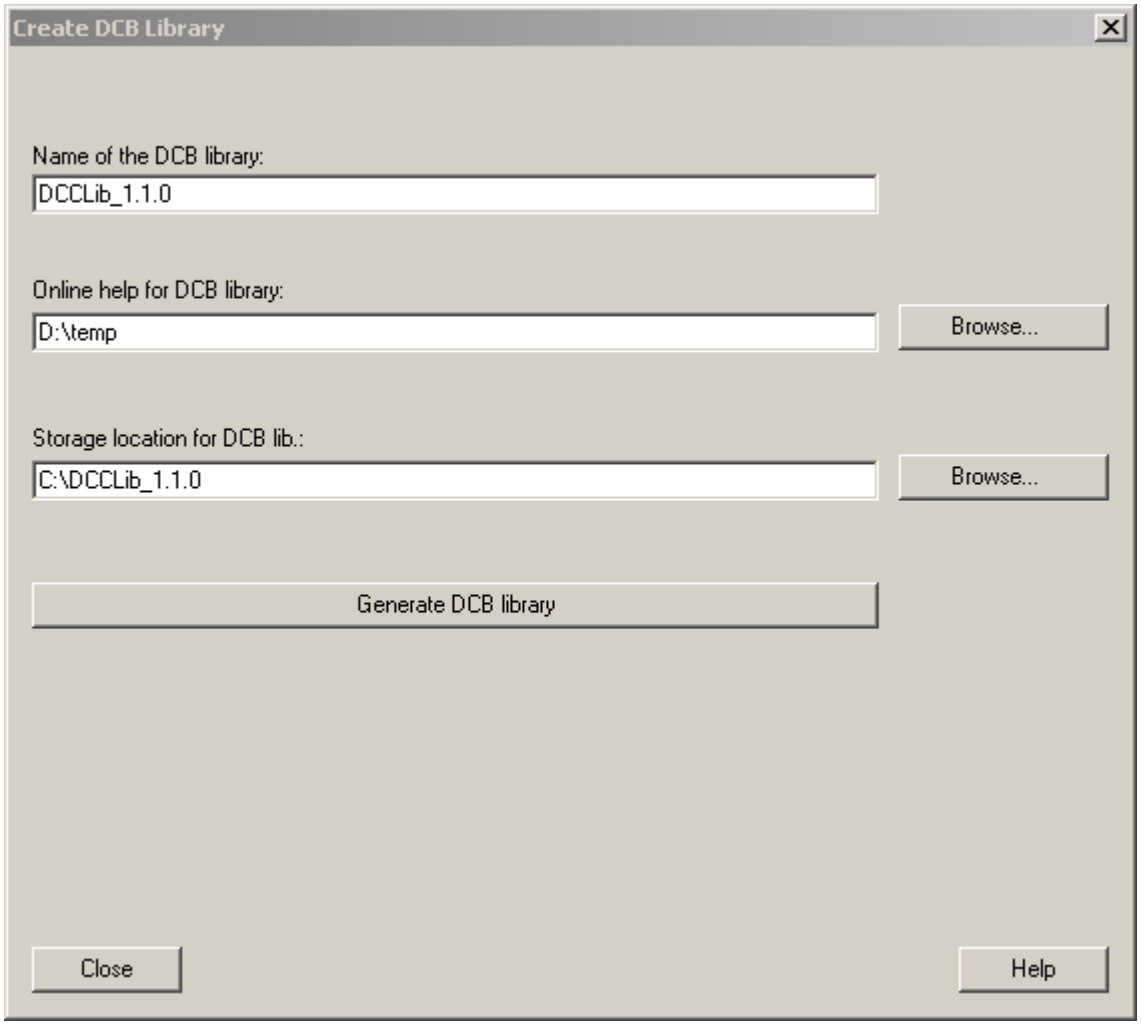


Figure 7-613 Create DCB library

In the dialog box, specify the online help directory for the library.

The name of the DCB library results from the name of the selected DCC library. You can freely select the storage location of the DCB library and then start the creation via the **Create DCB library** button. Information on the creation of the DCB library can be found in the detail view.

You have the option of deleting DCC chart sources for improved know-how protection.

Select the context menu Delete source of all block types in a compiled DCC library to delete the sources of the DCC charts. The charts can then no longer be edited.

By selecting the context menu **Block type -> Properties** at the library, you can allocate comments to block types and block connections in multiple languages.

An installable library may be installed on a different SCOUT/STARTER, regardless of the STEP7 project being used. This library does not contain the CFC chart sources.

Once you have installed the library using **Options -> Install libraries**, it is available for importing into DCC charts on the devices (Options -> Block types).

Note

If you create an installable library from a DCC library, you must replace the DCC library with the installable library in the DCC chart via the menu **Options -> Block types** of the CFC editor.

Note

The sources of the DCCs are no longer contained in the created DCB library. The contained functions can no longer be changed or monitored (know-how protection).

Information on how to create an online help for the DCB library created can be found in the section titled Creating the online help for block libraries (Page 5492).

Note

It is possible to create C block libraries from exported DCC libraries automatically. This results in improvements to the memory utilization in the execution performance of the DCC charts as well as increased know-how protection.

See also

Exchanging the basic library version for installed libraries (Page 5479)

Creating the online help for block libraries

Note

From SCOUT/STARTER 4.4 the block help editor is no longer part of the setup. A help can now be created as part of SIMOTION CLIB Studio or SINAMICS DCB Studio.

You can create a separate online help for the created libraries with the aid of a supplied editor, the **DCB help editor**. You edit your help files in the same view in which they are later displayed.

The DCB help editor can be found at ...*Siemens\Step7\S7BIN\helpeditor\DCB-Help-Editor.exe*.

Storage structure for online help

Save your online help files under the name **<block>_doc.xml**, whereby <block> is the name of the written block library. Create a folder structure comprising a **help** folder, containing the **help_a**, **help_b**, **help_c**, **help_d** and **help_e** folders. This results in the following assignment:

Table 7-612 Storage structure of the online help files

Folder	Subfolder	Language of the help file <block>_doc.xml
help	help_a	German
	help_b	English
	help_c	French
	help_d	Spanish
	help_e	Italian

A **<block>_doc.xml** file can be stored for each block in the help_x folder. For example, save your German language **<block>_doc.xml** file under **help > help_a**.

Procedure

1. Open the DCB help editor with a double-click on *helpeditor DCB-Help-Editor.exe*.
2. The STARTFILE_DOC.XML file, which is filled with elements and dummies, opens automatically.

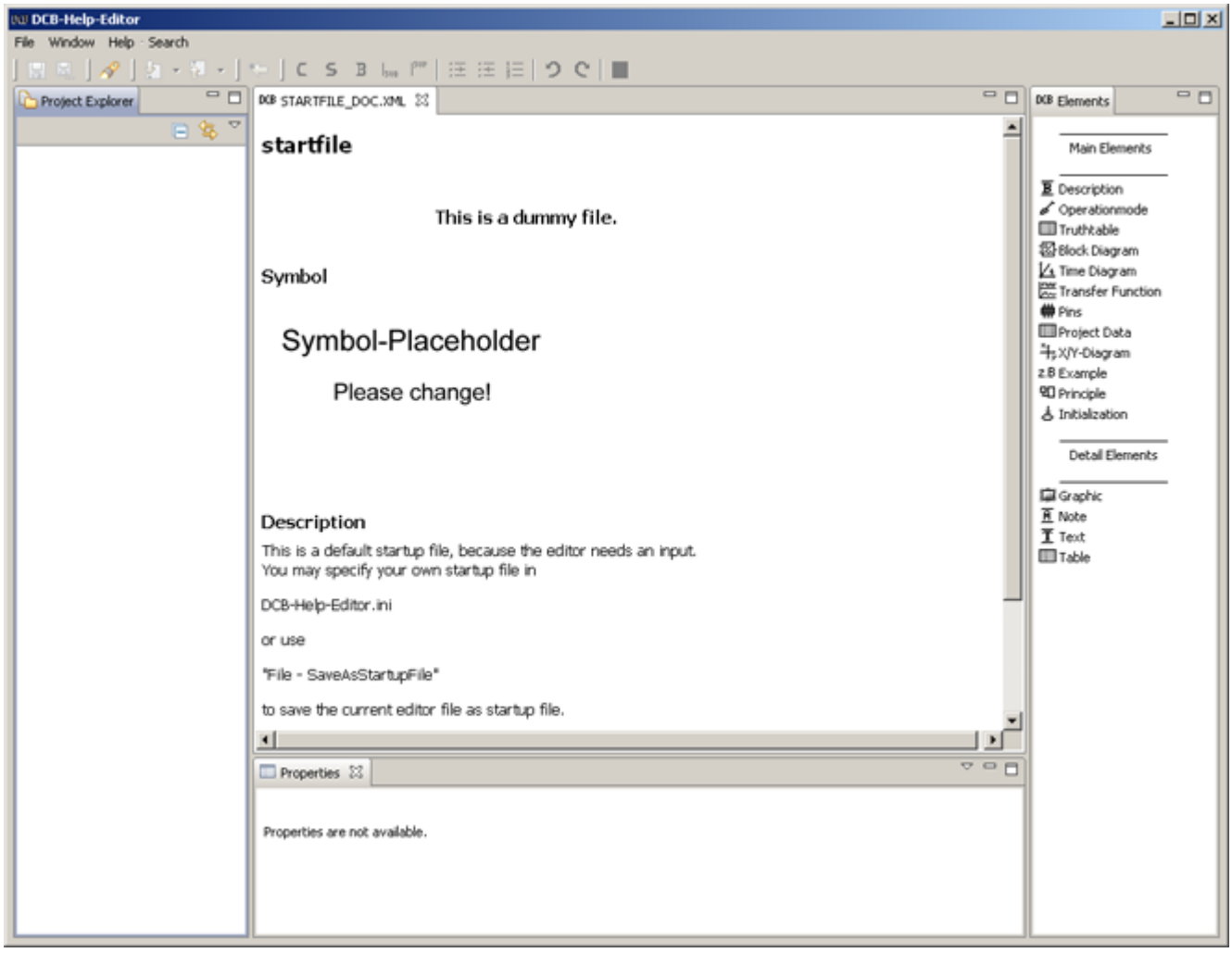


Figure 7-614 DCB help editor with STARTFILE

3. Replace these dummies with your own descriptions by selecting the dummies and entering or copying your text in the file.

4. The **Elements** pane on the right-hand side contains additional elements for the description (e.g. *Truth table*, *Block diagram*, *Example*, etc.), which you can drag to the desired position on your file using drag-and-drop and edit there.

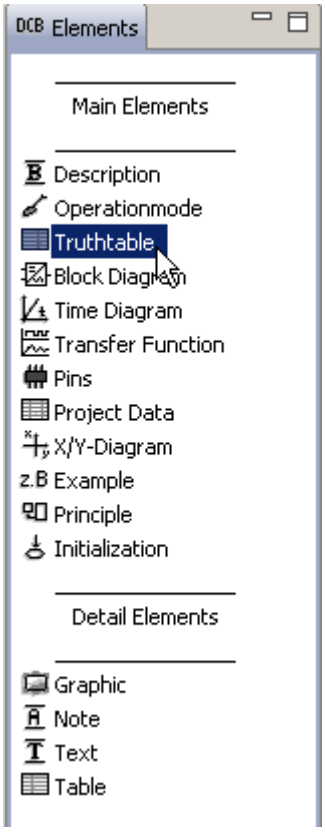


Figure 7-615 Elements for the description

5. Insert your own graphics by dragging the *Graphic* element to a suitable position in the file using drag-and-drop. In the following dialog box, enter the path under which the graphic is stored (use the **Browse** button).

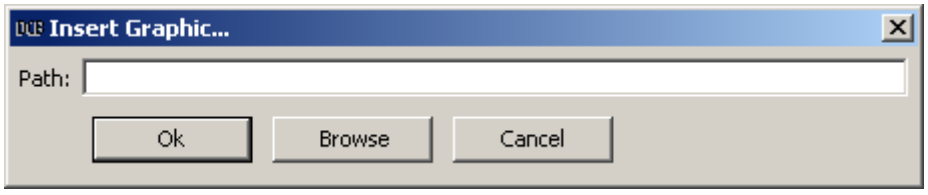


Figure 7-616 Inserting a graphic

6. Save the edited file with the button or via the **File > Save as...** menu in the storage structure described above.

Creating further online help files

You can open existing files via the **File > Open** menu.

You can create a new block description via **File > New**.

Note

Detailed operating instructions for the **DCB help editor** can be found in the menu under **Help > Help contents**.

Installation and uninstallation of DCB libraries

As of DCC Version 2.0.2, you have the option of installing or uninstalling DCB libraries from SIMOTION SCOUT using a dialog. As of DCC Version 2.1, you can install SINAMICS DCB libraries and SINAMICS DCC libraries in STARTER/SCOUT during operation and then use them in DCC charts without having to exit and restart STARTER/SCOUT.

You start the dialog box via the menu command **Options -> Installation of libraries and technology packages ...**

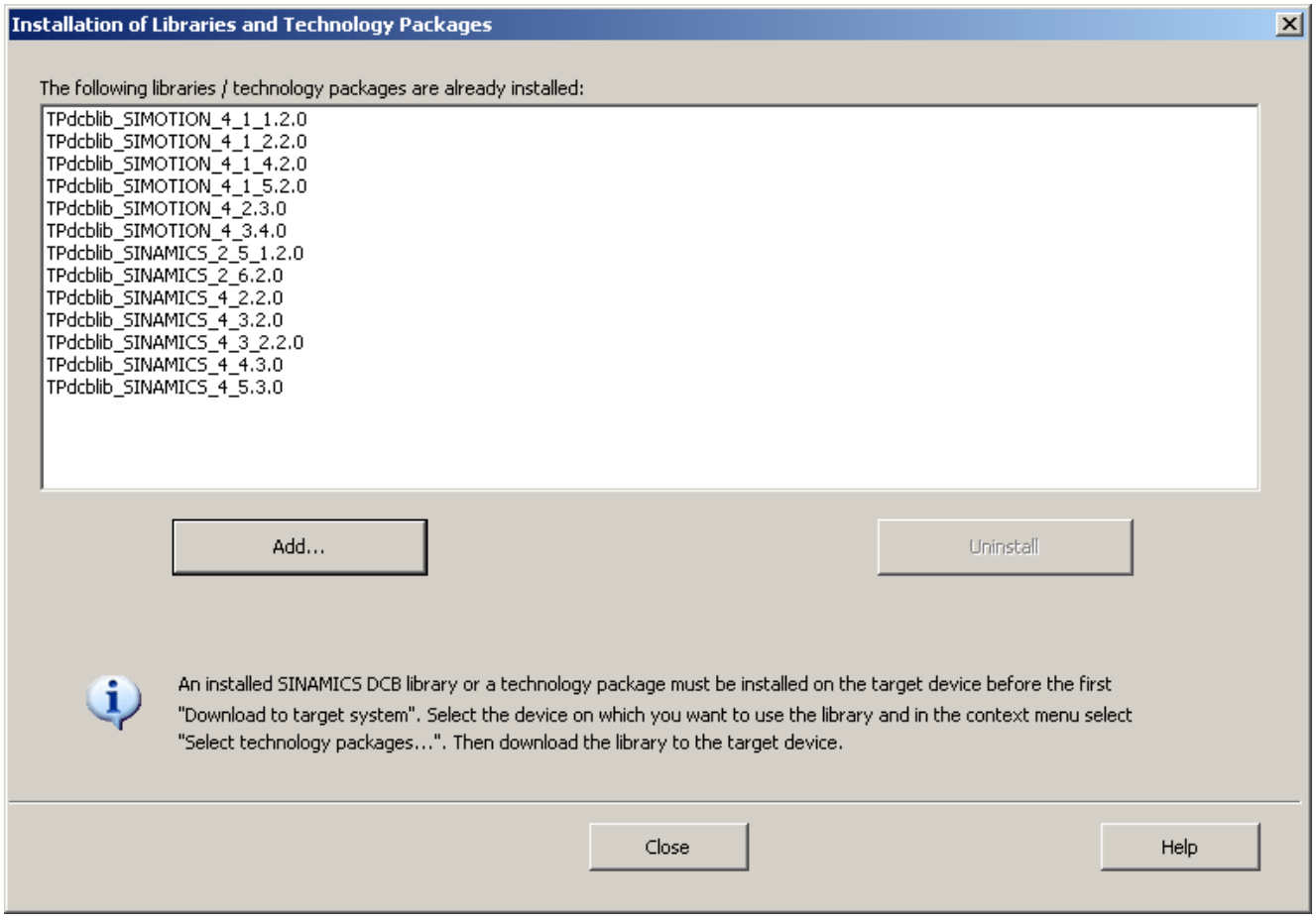


Figure 7-617 Installing and uninstalling DCB libraries - SIMOTION SCOUT

Uninstallation of DCB libraries

Under **Following libraries / technology packages are already installed:** you will see a list of the libraries that have already been installed.

- Select the library to be deleted and select **Uninstall**. The selected library will be uninstalled. A multiple selection is possible. Information on the uninstallation can be found in the detail view.

Installation of DCB libraries

Click the **Add...** button and navigate in the following Open File dialog box to the library to be installed. Select the library and install the selected library by double-clicking or by clicking the **Open** button. A multiple selection is possible. Information on the installation can be found in the detail view.

Exit the **Installation of Libraries and Technology Packages ...** dialog box via the **Close** button.

Note

The libraries are installed independently of the projects in SIMOTION SCOUT/STARTER. The libraries are not transported when projects are archived or exported. If the project is to be loaded on a different SCOUT/STARTER, the libraries used there must be reinstalled. This is also the case after SCOUT/STARTER has been reinstalled.

The DCC standard libraries DCBLIB (supplied from the factory) are preinstalled. If you explicitly require this as an installable library, you can find this library, e.g. "dcblibV2_0_simotion4_1_5.zip" (version-dependent name) after the installation of SCOUT in the"C:\Program Files\Siemens\Step7\U7umc\data\dcc\SIMOTION" directory or, if available, on the DCC DVD under "VOL1\CD_1\DCC\DCC_DCBLIB_SIMOTION\Disk1". You can also, for example, store the standard DCC libraries using the card reader in the USER directory on the CompactFlash card for later service assignments (the version of the standard DCC library may not be available in the engineering tool). This means that if service is required, you can use the CompactFlash card reader and the functions described here to reinstall the relevant libraries.

Note

For SINAMICS libraries as compared with SIMOTION libraries, there is the restriction that DCBLIB standard libraries can only be installed and uninstalled while no project is open.

The libraries are installed independently of the projects in STARTER. The libraries are not transported when projects are archived or exported. If the project is to be loaded on a different STARTER, the DCC libraries used there must be reinstalled. This is also the case after STARTER has been reinstalled.

The DCC standard libraries DCBLIB (supplied from the factory) are preinstalled. If you explicitly require this as an installable library, you can find this library, e.g. "dcblibV2_0_sinamics2_6.zip" (version-dependent name) after the installation of STARTER in the directory "C:\Program Files \Siemens\Step7\U7umc\data\dcc\SINAMICS" or, if available, on the DCC DVD under "VOL1\CD_1\DCC\DCC_DCBLIB_SINAMICS\Disk1". You can also, for example, store the standard DCC libraries using the CompactFlash card reader in the USER directory on the CompactFlash card for later service assignments (the version of the standard DCC library may not be available in the engineering tool). This means that if service is required, you can use the CompactFlash card reader and the functions described here to reinstall the relevant libraries.

See also

Updating the block library (Page 5477)

Exchanging the target device family for DCC libraries

As of SIMOTION SCOUT V4.3 you can exchange DCC libraries between the SINAMICS and SIMOTION target devices. In this way you can use a DCC library available for DCC SIMOTION on a SINAMICS target device or a DCC library available for DCC SINAMICS on a SIMOTION target device.

Note

This function is not available for STARTER.

Proceed as follows to insert DCC libraries in a different target device family:

In the project navigator, select the DCC library to be copied at the appropriate library container and select **Save as DCC SINAMICS library** or **Save as DCC SIMOTION library** in the context menu.

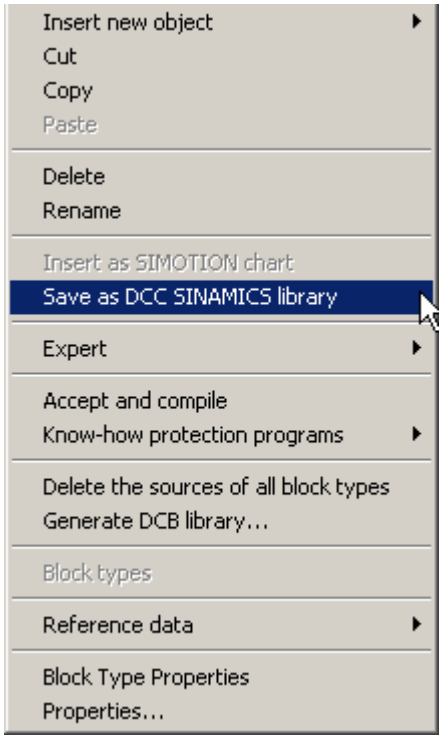


Figure 7-618 Save as DCC SINAMICS library

Proceed as follows to insert DCC library blocks in a DCC library of a different target device family:

In the project navigator, select the DCC chart to be copied on the DCC library at the library container and select **Copy** in the context menu. Switch to the library container for DCC library of the other target device family and insert the DCC chart via the context menu command **Insert as SINAMICS chart** or **Insert as SIMOTION chart**.

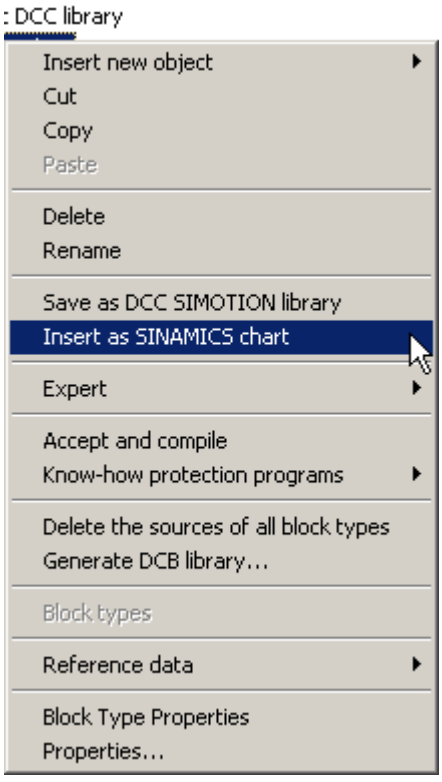


Figure 7-619 Insert as SINAMICS chart

When you insert the chart, the original library/chart name is taken over and an underscore and a number appended to the name. If there is already an underscore and a number appended to the name, the number is incremented.

The execution groups are changed automatically when saving under a different target device family.

Note

If chart sources of the DCC libraries are not available, no DCC editor is installed or there is no license for the DCC editor, you cannot insert DCC libraries into a different target device family. The **Save as** context menu command is then not available.

If block types from the library are not available on the target device, an error is output during compilation of the library. In this case, you must either delete or modify the block instances.

7.4.3.12 Know-how Protection

Know-how protection for libraries and programs

The SIMOTION SCOUT/STARTER engineering system provides know-how protection functions for your data. This enables you to prevent unauthorized access to your programs/parameters in the drive unit. There are two different types of know-how protection:

- Know-how protection for programs (ST source files, MCC charts, LAD/FBD programs and DCC charts) and libraries
- Know-how protection for drive units (as of SINAMICS V4.5)

Know-how protection for programs

The know-how protection installed in SIMOTION SCOUT protects the programs (ST source files, MCC charts, LAD/FBD programs and DCC charts) and libraries in your project.

In the project navigator, the protected programs contained in the project are displayed grayed-out. You can only open these programs for editing after you have entered the password. The know-how protection is retained after copy/paste and also after export/import.

Know-how protection for programs - setting-up/deleting a standard login

If you have not yet set up a standard login for the know-how protection, open the **Edit Standard Login** dialog box via the menu **Project -> Know-how protection for programs -> Edit standard login...** and enter a standard login there. Assign a password, confirm this and close the dialog box with the **OK** button.

⚠ WARNING

Danger to life through manipulation of DCC charts and DCC libraries

The use of unprotected DCC charts and libraries entails a higher risk of manipulation of DCC charts, DCC libraries and backup files.

- Protect important DCC charts and DCC libaries by using **know-how protection programs** or via the **know-how protection for drive units** in the SCOUT/STARTER. You can prevent manipulation by assigning a strong password.
- For **know-how protection programs**, use passwords which include at least eight characters, upper and lower cases, numbers and special characters.
- Make sure that only authorized personnel can access the passwords.

You can delete the standard login in this dialog box. A password is not required for this.

Configuring know-how protection for programs

Call the **Configure Know-how Protection for Programs...** dialog box via the menu **Project -> Know-how protection for programs** and select the desired coding type for programs and libraries.

Note

The dialog box is only available in SIMOTION SCOUT.

Standard (libraries and programs)

This level corresponds to the old security standard. Access to the programs is protected by a login and password. Programs and libraries can be recompiled at any time without knowing the password. This level is completely downward compatible, i.e. you can edit the project in an older version of SIMOTION SCOUT by saving the project in the old project format or by exporting and importing the project again.

Note

Only the standard coding can be used for DCC sources and libraries.

Medium (libraries and programs)

The coding of the password has been improved. Programs and libraries can be recompiled at any time without knowing the password. However, because of the changed coding, downward compatibility is no longer possible without knowing the password. If, for example, the program is exported and imported into an older SIMOTION SCOUT version or uploaded to a SIMOTION SCOUT project of an older version, the source text can no longer be displayed there.

High (for ST source files only)

In this level, (re)compilation is only possible after input of the password. However, libraries protected with this level can of course also be used after an export without knowledge of the password, because in this case the compilation result is also exported. Exports and imports are only possible in the same or higher SIMOTION SCOUT versions.

Note

The various levels of the know-how protection only apply to the new programs to be protected. You may have to delete, reconfigure and activate the know-how protection for already protected programs.

Activating know-how protection for programs

1. Open the project and select the PROGRAMS folder or the individual program in the project navigator.
2. Select the entry **Know-how protection for programs > Set** in the context menu.

Deleting know-how protection for programs

1. Select the PROGRAMS folder or the individual program in the project navigator.
2. Select the entry **Know-how protection for programs > Delete** in the context menu.
3. Enter the password in the following **Know-how Manager** dialog box and close the dialog with the **OK** button.

See also

Know-how protection for drive unit (Page 5502)

Know-how protection for drive unit

As of SINAMICS V4.5, know-how protection is available for drive units in STARTER / SIMOTION SCOUT. You can only activate/deactivate the know-how protection in online mode. When the know-how protection is set, p parameters can be neither read nor written. If certain parameters are to be excluded from the know-how protection, enter them in the expert list under parameter p7764 before you set the know-how protection.

Know-how protection for drive units - default setting

The know-how protection may already have been activated for drive units as soon as you have inserted a drive unit. In this case, consult the documentation for the drive unit.

If you want to change the pre-assigned password for the know-how protection, you must first deactivate the know-how protection via the context menu **Deactivate know-how protection**.

To do this, select the option **Definitive (password will be deleted)** in the **Deactivate know-how protection for drive unit** dialog box.

Enter the default password and close the dialog box with the **OK** button.

Activating know-how protection for a drive unit

When you activate know-how protection for the drive unit, a dialog for entering a password always appears. You can activate the SINAMICS know-how protection for drive units either with or without copy protection**.**

The **Know-how protection without copy protection** option is active by default.

You can select the following copy protection options for devices as of firmware V4.7:

- Basic copy protection (linked to the memory card)
- Extended copy protection (linked to the memory card and the Control Unit)

Only the **With copy protection** option is available for devices up to firmware V4.6; this corresponds to the extended copy protection.

If no know-how protection for the drive unit has been activated, but a password has already been defined (temporarily deactivated know-how protection for the drive unit), you can reactivate know-how protection for the drive unit with the previous password. You can, however, also change the password.

Define password

To define the password, you need to enter and confirm the new password. If you make a mistake, a message appears to inform you that the passwords entered do not match. If this happens, you need to reenter the passwords. Passwords are subject to the following rules:

- Min. length: 1 character, max. length: 30 characters.
- All read and write characters are permitted, see Opening a project under every set Windows language setting (language-neutral).
- It is the personal responsibility of the user to ensure adequate password security.

⚠ WARNING

Danger to life through manipulation of DCC charts and DCC libraries

The use of unprotected DCC charts and libraries entails a higher risk of manipulation of DCC charts, DCC libraries and backup files.

- Protect important DCC charts and DCC libaries by using **know-how protection programs** or via the **know-how protection for drive units** in the SCOUT/STARTER. You can prevent manipulation by assigning a strong password.
- For **the know-how protection of drive units**, use passwords which include at least eight characters, upper and lower cases, numbers and special characters.
- Make sure that only authorized personnel can access the passwords.

Opening a project with any language setting under Windows (language-neutral)

If you want to edit your projects in different Windows systems with different language settings, you should only use characters from the ASCII character set to ensure problem-free working with know-how protection. As a general rule, a project must first be enabled for this purpose in the SIMATIC Manager, but independently of the know-how protection. To do this, open the project in the SIMATIC Manager. Then go to the Edit menu and select the following option under Object properties: Can be opened with any Windows language setting (language-neutral).

Note

If you want to process your projects on Windows systems with different language settings, then you must define this in the project settings. Navigate via the menu **Project -> Properties** to the **Properties - Project** dialog box and activate the **Can be opened under every Windows language setting (language-neutral)** checkbox. Only use ASCII characters for the password of the know-how protection for drive units.

Diagnostics functions

If, despite active know-how protection, you still require the trace, measuring function or the function generator, select the **Permit diagnostic functions** option. You may have to activate the required parameters first in the exception list.

Note

Diagnostic functions possible as of V4.7

Diagnostic functions that work despite active know-how protection are available as of V4.7.

Deactivating know-how protection for a drive unit

To deactivate the feature, you must enter the correct password. The know-how protection for the drive unit can be:

- **temporarily** deactivated: The know-how protection is active again after switching off and switching on.
- **permanently** deactivated: The know-how protection remains deactivated even after switching off and switching on again.

If you select "Permanently", you can also carry out a data backup on the Control Unit with **Copy RAM to ROM**. The checkbox with the same name is active in this case and is automatically activated. If you deactivate this checkbox, you must perform a manual **RAM to ROM** data backup later if the know-how protection remains deactivated after switching off and on.

Absolute know-how protection for the drive unit

Absolute know-how protection that cannot be deactivated is created by deleting parameter p7766 from the exception list and then activating the know-how protection. Create a backup copy of your project before you activate absolute know-how protection.

Combining know-how protection for drive unit with write protection for drive units

You can combine the **Know-how protection for drive unit** with **Write protection for drive units**. Note that you must first set the **Know-how protection for drive unit** and then the **Write protection for drive units**.

If you have combined the **Know-how protection for drive unit** with the **Write protection for drive units**, the parameters of the exception list can only be read. The **Write protection for drive unit** does not apply for parameters with the WRITE_NO_LOCK attribute. These parameters can be written at any time. With combined write and know-how protection, the drive unit behaves similar to absolute know-how protection for drive unit, since a password can no longer be entered due to the write protection.

Functions available for activated know-how protection for drive units

When know-how protection for the drive unit is activated, the following functions remain available:

- Go online
- Restore factory settings
- Get information about the firmware version
- Read out errors (drive status, warnings, faults), alarms, the diagnostics buffer, and the alarm history
- Acknowledge errors/alarms
- Control panel functions
- Display created acceptance documentation
- Upload (restricted scope)

When know-how protection for the drive unit is activated, the following functions are not available:

- Download
- Export/import
- Trace
- Function generator
- Measuring functions
- Automatic controller setting
- Stationary/rotating measurement
- Delete alarm history
- Create acceptance documentation

The following functions are optional:

- Trace
- Function generator
- Measuring functions

These functions can only be executed if the diagnostic functions are permitted when the know-how protection is activated.

Behavior for devices with know-how protection for the drive unit

- Know-how protection always applies to the device, not to individual DOs.
- The content of the project navigator remains unchanged.
- All r parameters can still be read.
- p parameters can be neither read nor written with the following exceptions:
  - Parameters with the KHP_ACTIVE_READ or WRITE_NO_LOCK attribute
  - Parameters from the exception list

- Instead of a complete expert list, only a reduced expert list is displayed. This contains all the r parameters as well as the p parameters that have been enabled by attributes or the exception list. The behavior also applies to functions of the expert list, user-defined lists of values, and watch tables.
- An expert list comparison can be performed, the values of the know-how-protected parameters are displayed as **know-how protected**.
- No wizards are available.
- It is not permissible to delete this device in its entirety or to delete individual components from it.
- It is not permissible to rename the device.
- It is not possible to call dialogs containing functions that are not supported by know-how protection for the drive unit. An error message is displayed.
- Where windows contain functions that are not supported by the know-how protection for the drive unit, they do not show any active contents (with the exception of the expert list and version overview); instead, they indicate that know-how protection for the drive unit is activated.
- Scripts which need read or write access to protected parameters/functions, cannot grant appropriate access and are rejected.

See also

Know-how protection for libraries and programs (Page 5500)

Write protection for drive unit (Page 5510)

Setting know-how protection for a DCC chart

Setting the know-how protection function for a DCC chart

A DCC chart can be protected as follows:

1. Select the chart in the SIMOTION SCOUT/STARTER.
2. In the context menu of the DCC chart, execute the **Know-how protection -> Set** function in offline mode.

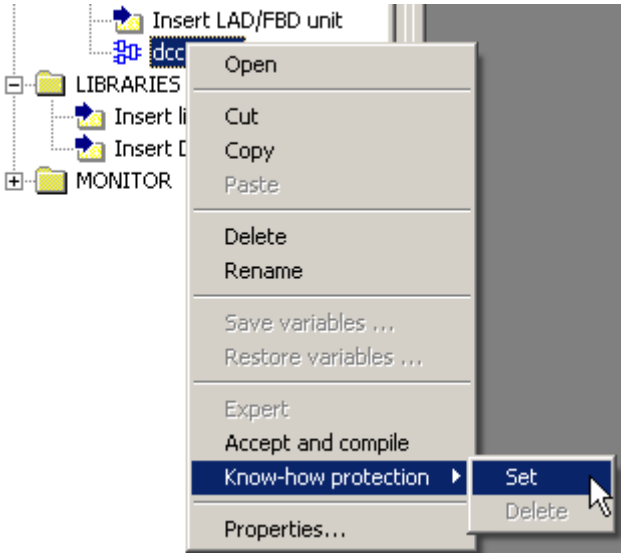


Figure 7-620 Setting know-how protection for a DCC chart

3. When using the know-how protection function for the first time, you will be prompted to enter the required access data (login and password) for protection of the chart in the **Know-How Manager** window. Close the **Know-How Manager** window via the **OK** button.

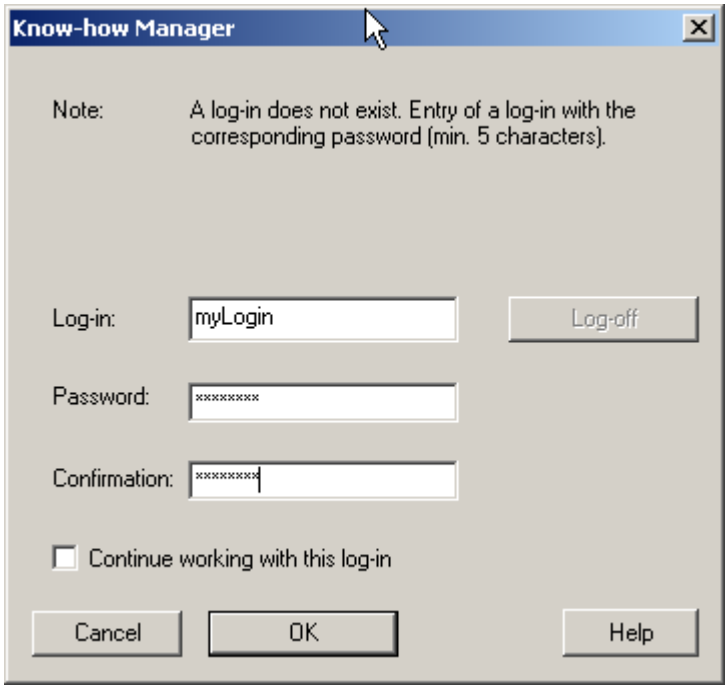


Figure 7-621 Know-how manager

The DCC chart is now shown grayed out.

You can also define a general login with password in the **Project -> Know-how protection** menu item, which then is used automatically for setting the know-how protection on the selected chart. Although you are then automatically logged in with this login, you can at anytime log off from the **Project -> Know-how protection** menu item.

Activating the know-how protection function

To activate the know-how protection, you must log off the logged-in user in the **Know-How Manager** window (access to the **Know-How-Manager** is via **Project -> Know-how protection** in the menu). The protected DCCs of the opened project are then locked and cannot be opened. Only by entering the password can you remove the lock and open the DCC charts.

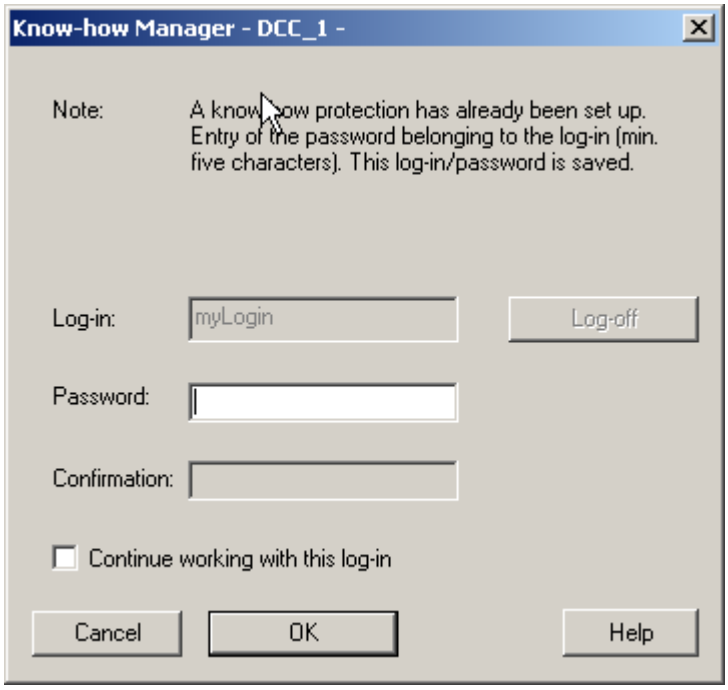


Figure 7-622 Know-how manager

Note

When DCC charts are protected, the execution sequence cannot be protected in the execution system. Although the chart can be moved, it cannot be edited. Note that if the execution sequence of a protected DCC chart is changed, the functionality can no longer be guaranteed or may fail.

Deleting the know-how protection function

To delete the know-how protection, you must first be logged in (if you are not already logged in, you are automatically prompted to do this during the procedure described below).

You can delete the know-how protection of a chart as follows:

1. Select the chart in the SIMOTION SCOUT/STARTER.
2. In the context menu of the DCC chart, execute **Know-how protection -> Delete**.
3. Close the **Know-How Manager** window via the **OK** button after entering the password.

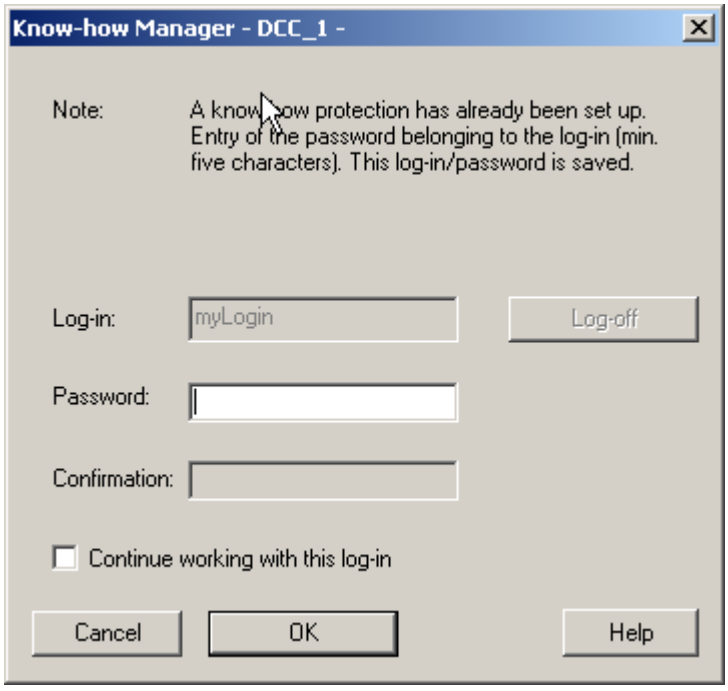


Figure 7-623 Know-how manager

Note

The know-how protection is not automatically reactivated in SIMOTION SCOUT/STARTER when you close the DCC chart.

Canceling the know-how protection

If you want to open a know-how-protected DCC chart, you can do this by double-clicking as usual. If you are already logged in, the know-how protection is automatically canceled, if you are not logged in, you are automatically prompted to do so when opening.

7.4.3.13 Write protection for drive unit

As of SINAMICS V4.5, write protection for drive units is available. You can only activate/deactivate this write protection in online mode.

Activate/deactivate the write protection via the context menu **Write protection for drive unit** of the drive unit or via parameter *p7761* of the expert list. A password is not required to activate/

deactivate the write protection. You can query the status of the write protection via parameter*r7760.0* in the expert list (value = 0 means that the write protection is not set).

When write protection is activated, the following functions can be used in offline mode:

- Inserting, editing, deleting DCC libraries
- Inserting, editing, deleting DCC charts
- Inserting, editing, deleting DCC blocks

Note

When write protection is activated for a SINAMICS target device, however, a download to the target device is not possible. This prevents any changes made offline in DCC charts taking effect in the target device.

The following functions cannot be used when write protection is activated:

- Editing a DCC chart in test mode
- Inserting or deleting blocks online
- Inserting, moving or deleting connections
- Compiling a DCC chart
- Saving and recompiling all
- Downloading the project

With active write protection, compilation of the DCC chart on the DO is rejected both from the workbench as well as from the DCC editor with the following error message:
"Write protection is currently active on the drive unit. The DCC chart cannot be compiled."

Write access to DCC parameters is rejected with the following error message:
"Value was rejected.
Parameter xxx could not be set to this value. The old value will take effect again."

Note

If you want to activate the write protection permanently, execute the **Copy RAM to ROM** function.

See also

Know-how protection for drive unit (Page 5502)

Know-how protection for libraries and programs (Page 5500)

7.4.3.14 Startup behavior

With regard to the startup behavior, note that the respective initialization value is active at the block connections during startup and then the calculated value in each following cycle.

For more detailed information on the start-up behavior, please see Chapter Characteristics of block connections (Page 5438).

7.4.3.15 Software upgrade and module exchange

SCOUT or STARTER projects that contain DCC charts and are compiled with V4.1.1 or V4.1.2 can be opened and loaded with SCOUT/STARTER V4.1.3, even without a DCC license.
This applies regardless of whether or not the DCC chart sources (i.e. original project) are available.

Note

This specification applies as long as the device version does not change. Following a replacement with a new device version, the DCB library versions present on the new device must be imported and the DCC chart recompiled. The corresponding DCC license (CFC for SIMOTION/ SINAMICS) is required for this.

Until DCC 2.1, the library version had to be updated manually after the upgrade of a device version.

As of DCC 2.1 / SCOUT/STARTER V4.2, the library version is upgraded automatically.

Saving and restoring retain variables - SIMOTION only

DCB block instances may contain retain variables.

Select **Save variables...** from the context menu of the DCC chart to save the contents of these variables to a file. The values can then be restored from this file. DCC retain data is retained when the platform is changed or the version is upgraded.

7.4.3.16 Version information

Proceed as follows to display the version information:

1. Left-click the *Help -> Info...* menu item in STARTER/SCOUT
2. Click the *System information...* button in the following *Information* dialog

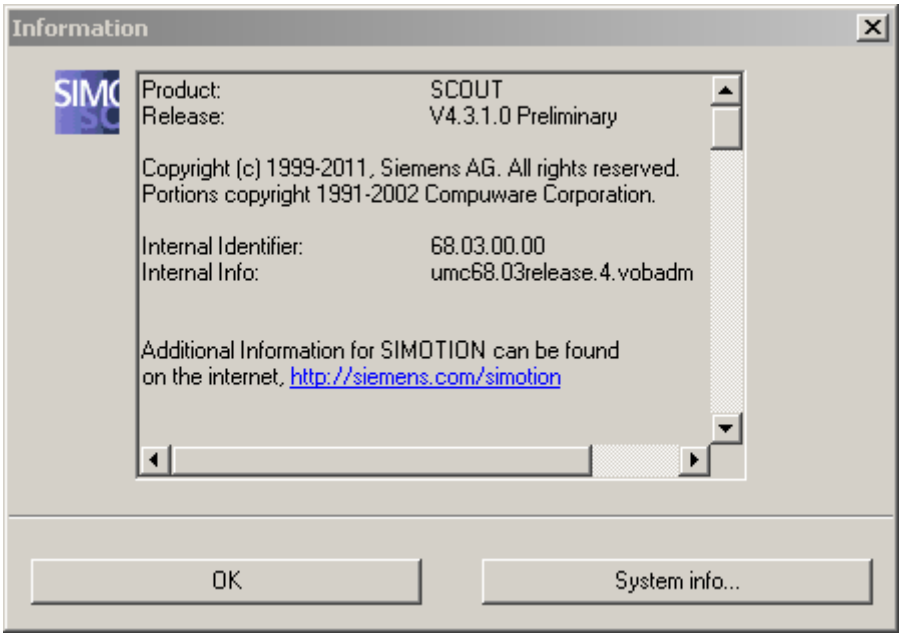


Figure 7-624 Version information - Information

3. The *System Information* dialog with the system information appears.

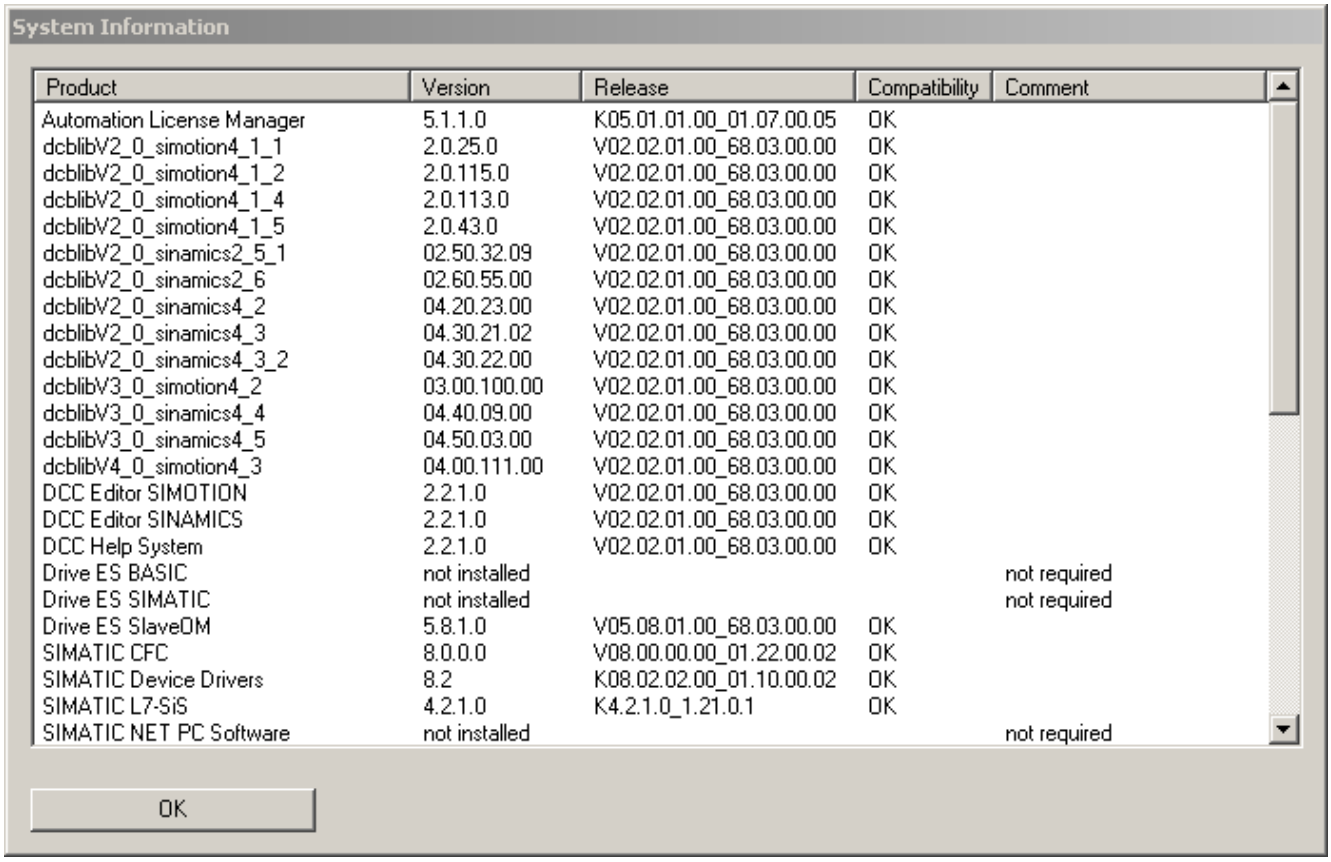


Figure 7-625 Version information - System information

7.4.3.17 XML export/import of DCC charts

During the project export, the CFC charts with the STEP7 data storage are exported in binary format.

From DCC Version 2.1, DCC charts can be individually exported in XML format and imported again during XML import.

This means that you can transfer a DCC chart to another project (from a project with SINAMICS 2.6 devices to a project with SINAMICS 4.x devices, for example).

The following restrictions apply to the XML export/import of DCC charts:

- DCC charts that have been exported from SIMOTION devices cannot be imported into SINAMICS devices, and vice versa
- DCC blocks that have been exported from SIMOTION DCC libraries cannot be imported into SINAMICS DCC libraries, and vice versa.

The table below outlines the compatibility of XML exports/imports in relation to the relevant SCOUT/STARTER version.

Compatibility	
Project	Projects exported with SCOUT/STARTER V4.1 can be imported with V4.2 without any loss of information. After opening the project, the user is asked to convert the CFC charts to CFC 7.1.
	Projects exported with SCOUT/STARTER V4.2 can be imported with V4.1 without any loss of information. However, it is not possible to edit DCC charts with CFC 7.0. In the case of projects that were created with V4.2 and imported using an older version of SCOUT/STARTER, a warning appears when the import process is carried out.
Device	Devices exported with SCOUT/STARTER V4.1 can be imported with V4.2 without any loss of information. If block types that were not available in V4.1 are used in a V4.2 project, the devices can be imported without any loss of information. Errors are reported when compilation is carried out, however.
	Devices exported with SCOUT/STARTER V4.2 can be imported with V4.1. The DCC chart sources are lost when they are imported with V4.1. No errors or warnings are output.
Drive object (DO)	Drive objects exported with SCOUT/STARTER V4.1 can be imported with V4.2. These do not contain any DCC charts.
	Drive objects exported with V4.2 can be imported with V4.1. The DCC chart sources are not imported.
	If a DCC chart is exported on a DO type A and imported on a DO type B, the adaptation process for the sheet bar interconnection is the same as copying/pasting DCC charts. When pasting on the new DO, the old execution group is also transferred. During compilation, the execution group that applies to the new DO is entered automatically.
CFC version	DCC charts that were created with an older CFC version can be imported with a newer CFC version. Downward compatibility is also supported.
	The export/import of projects and chart containers only supports upward compatibility. A chart created using CFC 7.1 can no longer be opened with CFC 7.0.

The following requirement applies to the XML export/import of DCC charts:

- The CFC editor must be installed

XML export/import of DCC charts is possible at the following levels:

- Project
- Device
- DO (drive object, CUxx, TBxx, TMxx)
- DCC chart
- DCC library

The assigned DCC charts are included in the XML export/import of the selected object.

Note

If a CFC editor is not installed, only the DCC charts (i.e. without chart sources) are exported/ imported. No error message is output.

No DCC license is required for exporting/importing DCC charts.

XML export at project level

In SCOUT/STARTER, select **Project -> Save and export...** from the menu or **Expert -> Save and export project** from the context menu, and in the **Export Project** dialog that appears, use the **Browse...** button to specify the target directory for the export or enter this directly in the text field.

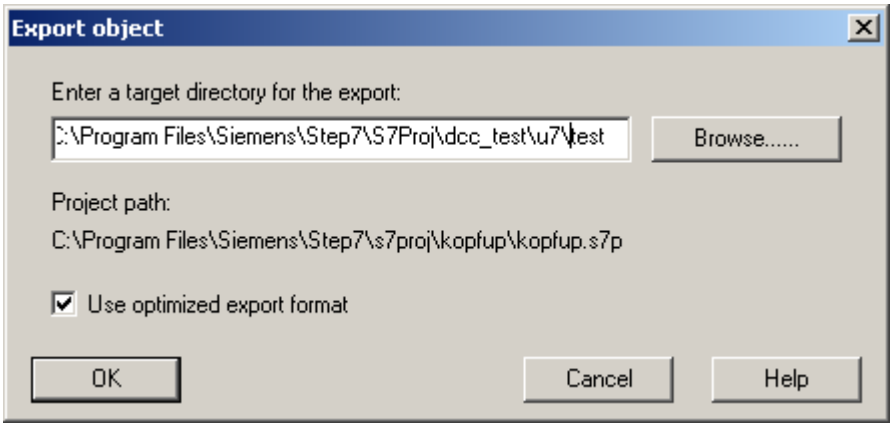


Figure 7-626 Project XML export

Click **OK** to start the export. You can track the progress of the export on the **XML export/import status display** tab.

If the target directory already exists, you will see the following dialog:

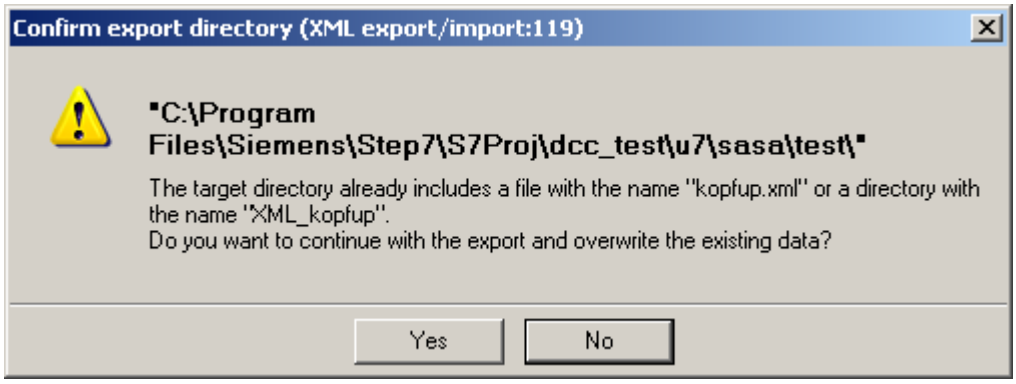


Figure 7-627 XML export - Error message

Choose **Yes** to overwrite the data in the target directory or **No** to cancel the export.

XML export at DCC chart level

Select the DCC chart you wish to export in the project navigator, then select **Expert -> Save project and export object** from the context menu in SCOUT/STARTER. In the following **Export object** dialog, use the **Browse...** button to specify the target directory for the export or enter this directly in the text field.

Click **OK** to start the export. You can track the progress of the export on the **XML export/import status display** tab.

Note

DCC charts that were exported from SCOUT/STARTER cannot be imported into the SIMATIC Manager.

XML import

DCC charts and DCC libraries can be individually exported from SCOUT/STARTER and reimported into a SCOUT/STARTER project. They have the same layout after the import as they did before the export. No information is lost from the DCC chart, i.e. the subcharts, library information, sheet bar interconnections, execution groups, alias definitions, p21000 settings, and know-how protection are all retained during an XML export/import. Know-how protection can be revoked after the import, making it possible to edit the charts again.

You also have the option of importing a previously exported DCC chart into an existing chart.

When importing DCC charts, a distinction is made between importing a DCC chart and importing into an existing DCC chart:

- Container in project navigator -> Expert -> Import object
- DCC chart -> Export/import object

For an XML import, proceed as follows:

In the project navigator, select the object into which you wish to import, and then select **Expert -> Import object** from the context menu in SCOUT/STARTER.

In the dialog that appears next (**Import object**), use the **Browse...** button to specify the source directory for the import or enter this directly in the text field.

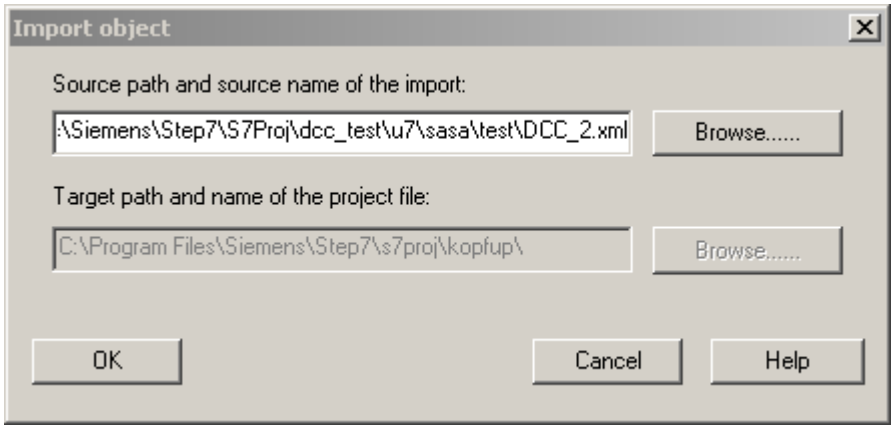


Figure 7-628 XML import

Click **OK** to close the dialog.

In the **Import object** dialog shown below, use the **OK** button to start the import or the **Cancel** button to cancel the import.

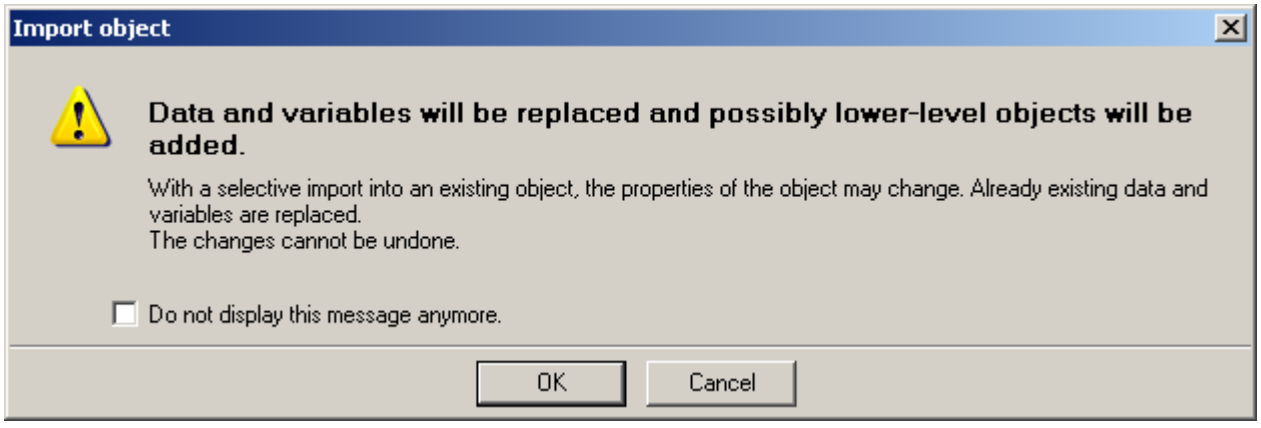


Figure 7-629 XML import warning

You can track the progress of the import on the **XML export/import status display** tab.

Note

It is possible to export/import DCC charts even if the chart in the current project has not been compiled or cannot be compiled without errors.

If the DCC chart has not been compiled so that it is up to date before the export/import, then the imported DCC chart will be empty or will not be up to date.

Even in this case, DCC charts can still be imported without any information being lost, provided that the libraries used in the chart are not yet installed or were imported on the program container. Reinstall the SIMOTION libraries by selecting **Options -> Install libraries...**, or by selecting **Select technology packages...** from the context menu in the case of SINAMICS.

The XML export/import can be carried out for DCC charts with existing DCC chart sources or for DCC charts without DCC chart sources. Please observe the following:

- During the import of DCC charts with DCC chart sources, the set values of the expert list are overwritten with the set values and BICO parameters from the DCC project.
- During the import of DCC charts without DCC chart sources, the adjustable parameters and BICO parameters (@ parameters) are not overwritten.

If the DCC chart sources are not available in a project because the DCC charts were obtained as a result of an upload from the target device or copying without a DCC license, or the DCC chart sources were explicitly deleted, the DCC chart is exported and imported without chart sources. The DCC chart that is created in this way can be loaded and compiled. You can select **Block types** from the context menu to exchange the version of the subordinate basic DCB libraries.

See also

Exchanging the basic library version for installed libraries (Page 5479)

7.4.3.18 XML export/import of DCC libraries

DCC libraries can be exported/imported in their entirety. Alternatively, it is possible to perform an XML export of individual DCC blocks from the DCC library and an XML import into a DCC library.

Data that is specific to the library (description of block connections, comments, block family, etc.) is exported/imported along with it.

Before the library sources are deleted, a check is performed to determine whether the libraries have been compiled so that they are up to date. As of SCOUT/STARTER 4.2, DCC chart sources are always exported/imported too if they are available on the library sources.

7.4.3.19 Reading back DCC chart sources from the target device

In DCC 2.0, DCC charts can be read back from the target device and loaded to another target device of the same type. To enable the DCC charts to be edited further, the DCC chart sources must be present in the original project.

As of DCC 2.1, DCC charts for which no up-to-date DCC chart sources or no DCC chart sources at all are present can be uploaded from the target device and read back to the DCC editor.

Chart sources for DCC libraries (typicals) are downloaded/uploaded via the library context menu. They are uploaded as part of the project uploading process.

Procedure

Under **Options -> Settings** in SCOUT, select the **Download** tab and activate the **Include DCC chart data** checkbox below the **Store additional data on the target device** checkbox to load the DCC chart sources to the target device.

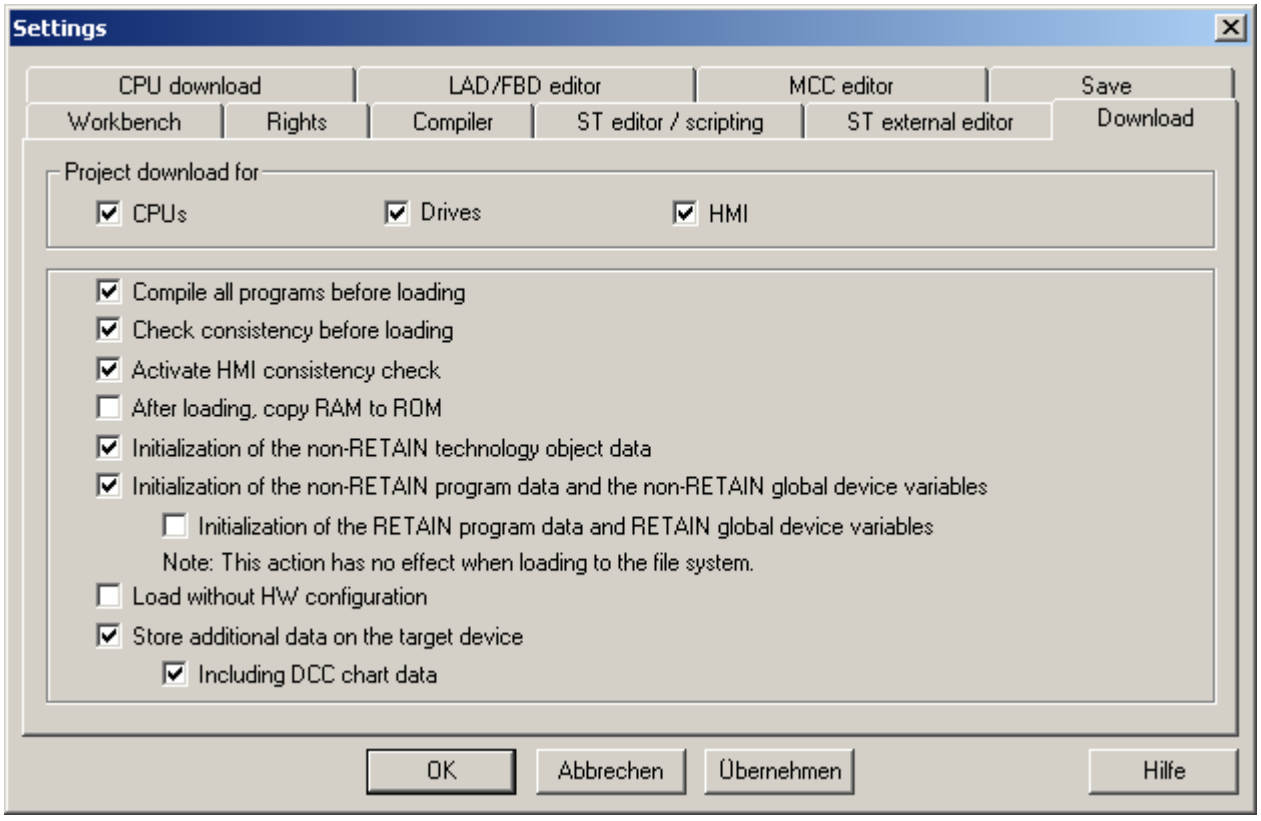


Figure 7-630 Storing additional data and sources on the target device - SCOUT

In STARTER, under **Options -> Settings,** select the **Download** tab and activate the **Store additional data on the target device** checkbox to download the DCC chart sources to the target device.

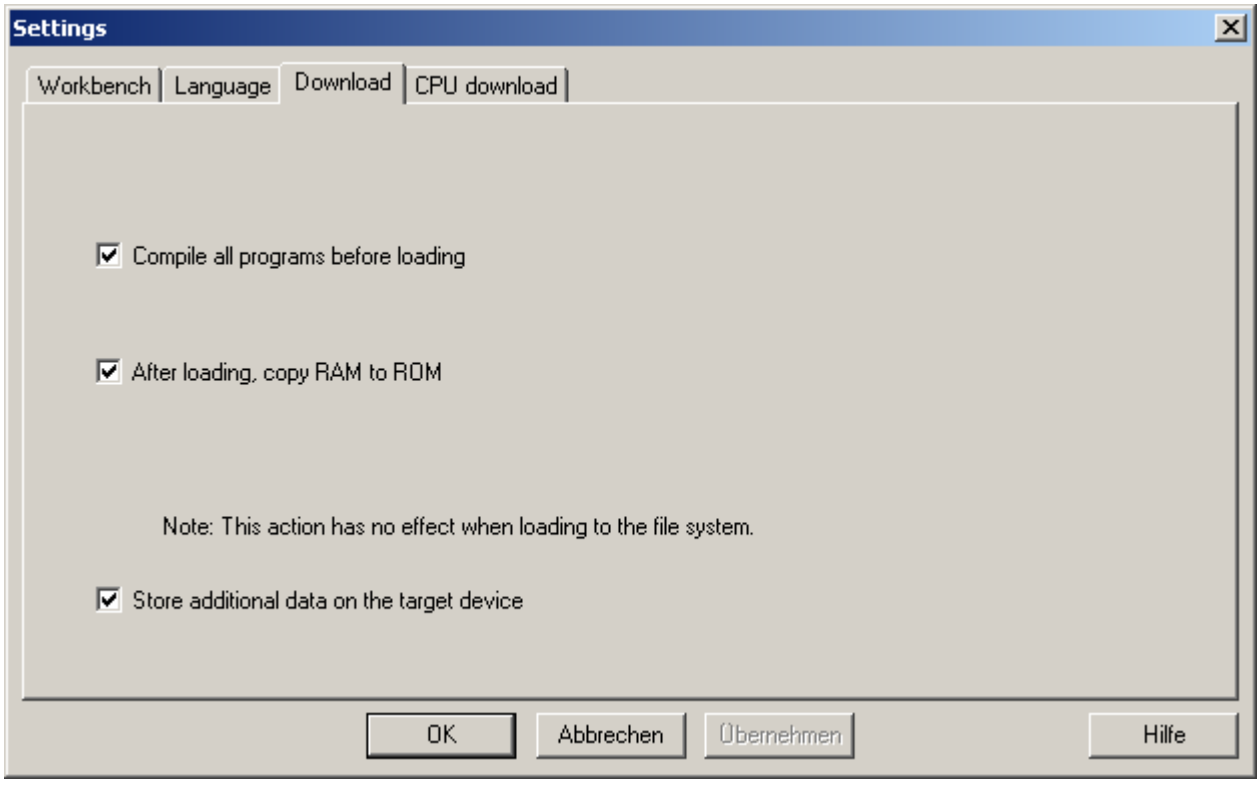


Figure 7-631 Storing additional data and sources on the target device - STARTER

During the CPU download, you can also select whether the DCC chart sources are to be downloaded to the target device for the CPU.

The default setting for the CPU-specific option is taken from **Options -> Settings -> Download**. The option selected for the CPU is then saved as the default setting for the next download.

Once you change the option under **Options -> Settings -> Download,** this is regarded as the default setting for all CPUs.

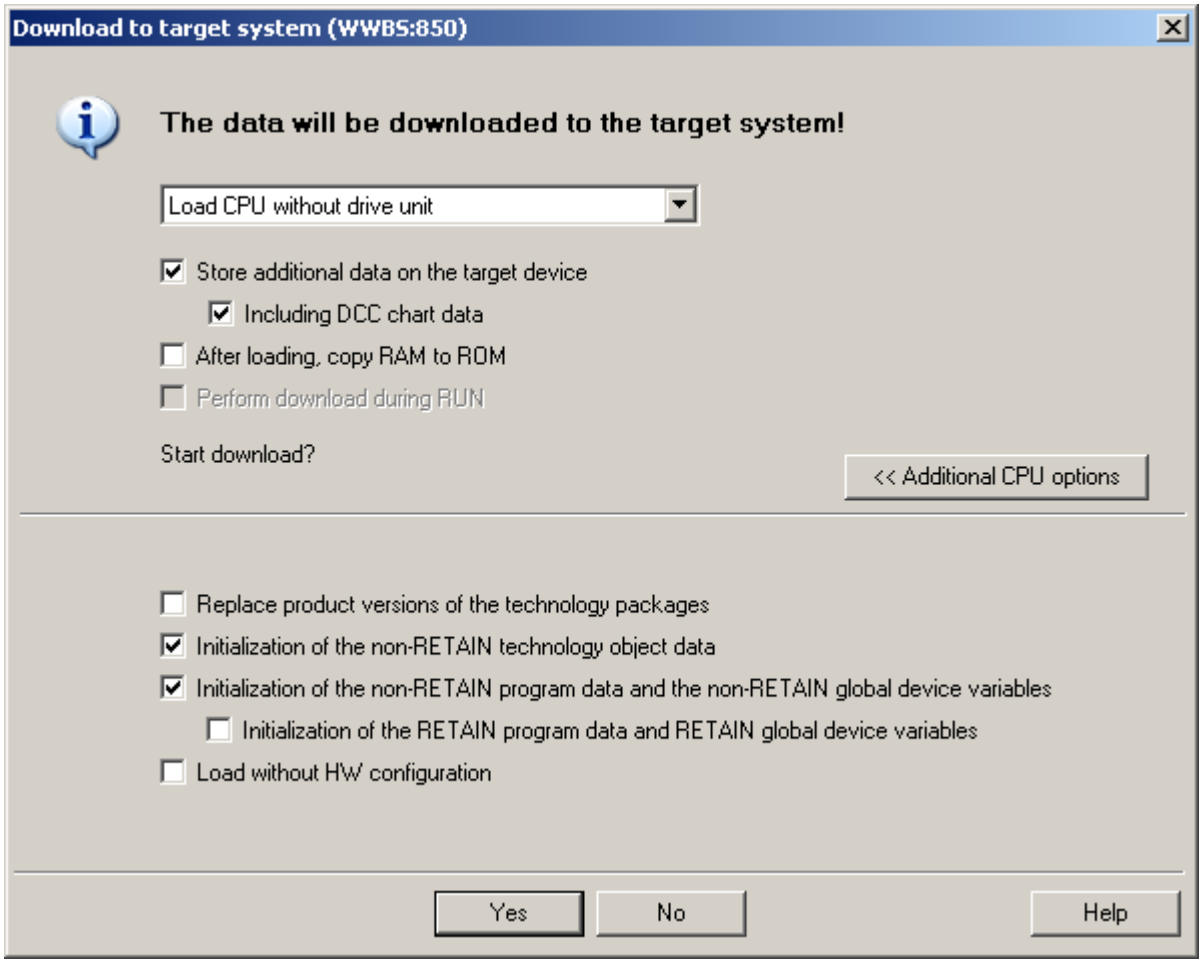


Figure 7-632 Downloading to the target device - SCOUT

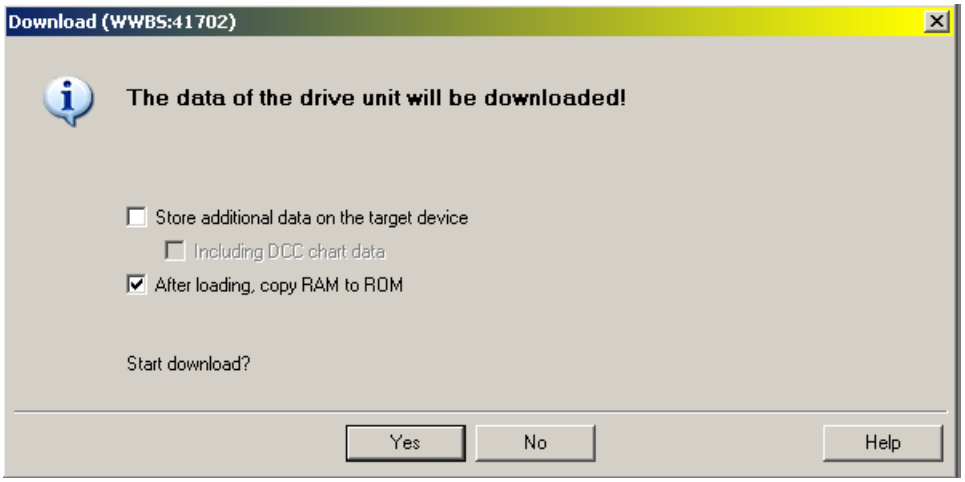


Figure 7-633 Downloading to the target device - STARTER

If you deactivate the **Store additional data on the target device** option, the chart sources in the target device are deleted during the next download.

If you activate the **Store additional data on the target device** option, the chart sources of the DCC libraries are downloaded to the target device, which are used in the DCC charts of the CU or CPU.

If DCC charts are available on the target device, the project is saved following upload.

When downloading to the PG, you must explicitly select whether the library sources from the target device will be loaded into the offline project. In this way, you can keep different library versions in different devices.

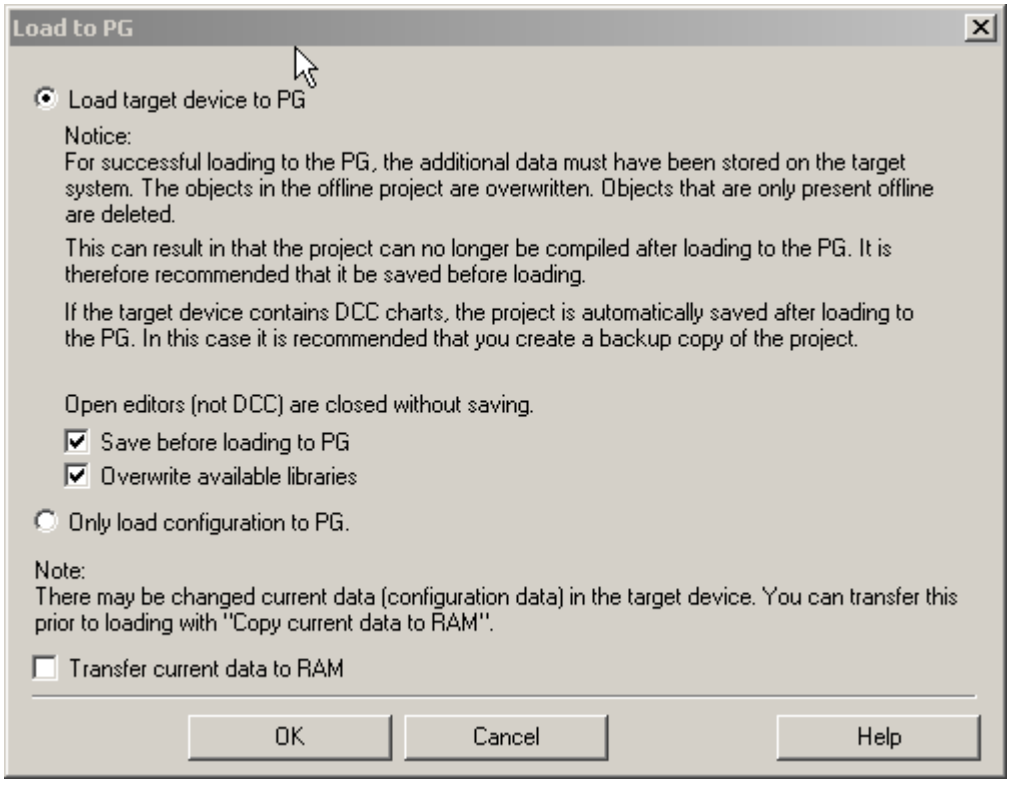


Figure 7-634 Downloading target device to PG - SCOUT

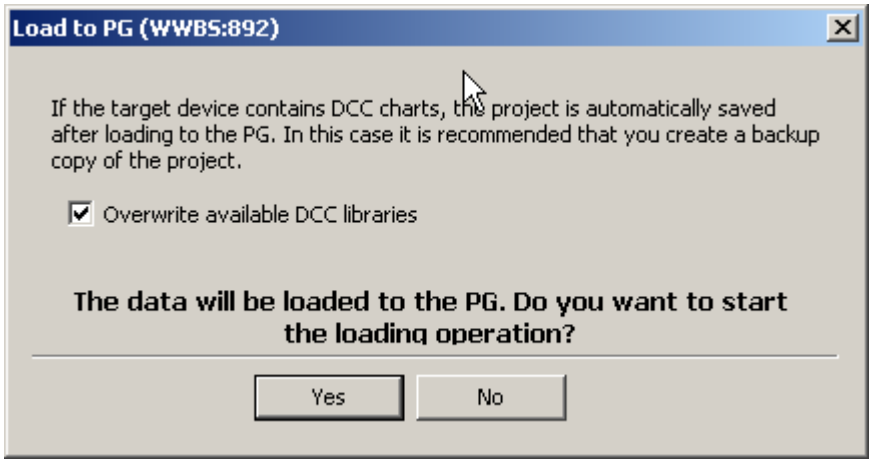


Figure 7-635 Downloading target device to PG - STARTER

Note

In the project comparison, DCC charts are displayed differently following read-back (incl. DCC chart sources), as the DCC chart sources are recreated and therefore have a different time stamp.

Conditions

- If the DCC chart is not open, the chart source will be updated automatically when the project is read back.
- If DCC charts are open in the editor during the project upload, the chart sources will not be updated automatically in the editor. In editor mode, the chart that has been read back can be updated from the CFC using function key **F5**.
- The chart must not be in test mode while it is being read back.
- If the chart sources are not present in SCOUT/STARTER, the DCC chart sources will be created automatically after the project has been uploaded.
- During the project upload, DCC charts with the same name that are already present in the engineering system will be overwritten.
- If the chart names are the same, DCC sources with know-how protection overwrite existing chart sources during the upload to the engineering system.
- DCC libraries are restored when the DCC chart sources are read back.
- The layout of the charts after the read-back process is the same as when they were edited: The positions of the block instances are restored and comments are available in the same position as they were before.
- The libraries used in the charts must be installed in SIMOTION SCOUT/STARTER so that the upload of DCC charts is performed without errors.

The process of reading back DCC charts is independent of the SIMOTION firmware version.

For SINAMICS, it is possible as of SINAMICS V4.4.

The additional data for the read-back process must have been loaded to the target device. Additional data for DCC charts can be created and loaded as of DCC 2.1.

When charts with know-how protection are read back, the know-how protection for the DCC charts is restored in the engineering system.

Changes that have been made online in DCC test mode are taken into account when DCC chart sources are read back.

The following restriction applies if you do not have a DCC license / the CFC has not been installed:

- DCC charts can be uploaded and loaded back onto the target device, but they can no longer be edited - even if the CFC is installed or a DCC license is made available after this.

Note

Up to STARTER/SCOUT V4.1.5, the process of reading back BICOs/parameters for DCC SINAMICS had to be explicitly initiated from the editor after the project had been read back from the target device.

With STARTER/SCOUT V4.2, however, the BICOs/parameters are automatically read back when the project is read back into the DCC chart.

7.4.4 DCC for SINAMICS

7.4.4.1 Overview

Introduction

This product brief is intended for first-time users who are not yet familiar with the DCC program package. Using a short example, you will find out how to create a new DCC, interconnect DCBs (drive control blocks; function blocks), compile the chart, download it to the target device and test it.

Note

In most cases, there are a number of options for working with the DCC editor (e.g. using the keyboard). In this example, one option is used. Apart from a few exceptions, no alternative methods of operation are explored here.

Note

The documentation on SINAMICS refers to versions STARTER 4.4, DCC 2.3 and SINAMICS Firmware 4.7.

Software requirements

The software requirements for DCC are the same as those for SINAMICS STARTER.

You will additionally need:

- **DCC SINAMICS** software option package

Note

The DCC SINAMICS option package contains on a USB stick the Step7 CFC license required for the DCC editor. The installation is made using the Step7 Automation License Manager (refer to the help of the Step7 Automation License Manager). The Step7 Automation License Manager program is installed automatically together with STARTER.

When the product is supplied, the DCC technology option is not yet available on the drive unit. The DCC technology option must be loaded onto the drive unit's CF card in a separate operation using SIMOTION SCOUT or STARTER. Afterwards, the SINAMICS CU3x0, SINAMICS DC MASTER or the SIMOTION D4xx must be switched off and then on again. Only then can the DCCs be downloaded and run on the drive objects.

From SINAMICS 4.3, the dcblib for S120 is already present on the card. For single drive units, POWER ON after the download of the dcblib is no longer necessary.

There is only ever one DCC on a drive object. DCC can be activated simultaneously on several drive objects on a drive unit. DCC is not available on SINAMICS S110 drive units (CU305 module).

SINAMICS system integration

Applications and features

Application

A logic operation, which connects several states (e.g. access control, plant status) to a control signal (e.g. ON command), is required for controlling the drive system in a wide variety of applications.

As well as logic operations, a number of arithmetical operations / storing elements are increasingly becoming a requirement in drive systems.

Note

This additional functionality increases the computing time load. This means that the maximum possible configuration for a Control Unit is restricted.

Drive Control Chart (DCC) functionality is available for every drive object of the drive unit listed in the following table.

Table 7-613 Drive object types for DCC SINAMICS

Drive object type	Object number (r0107)	Meaning
CU_S	1	Control Unit SINAMICS S (SINAMICS S120/ S150)
CU_G	2	Control Unit SINAMICS G (SINAMICS G130/ G150)
CU_I	3	Control Unit SINAMICS Integrated
CU_CX32	4	Controller Extension for boosting the computing performance
CU_GM	5	Control Unit SINAMICS GM
CU_DC	6	SINAMICS DC MASTER Control Unit
CU_GL	7	Control Unit SINAMICS GL
CU_SL	101	Control Unit SINAMICS SL
A_INF	10	Active Infeed control
SERVO	11	Servo control
VECTOR	12	Vector control
VECTORMV	13	Vector control for SINAMICS GM
VECTORGL	14	Vector control for SINAMICS GL
VECTORSL	16	Vector control for SINAMICS SL
DC_CTRL	17	Closed-loop control for DC drives
S_INF	20	Smart Infeed control
B_INF	30	Basic Infeed control
A_INFMV	40	Active Infeed control for SINAMICS SM150
B_INFMV	41	Basic Infeed control for SINAMICS GM150
TB30	100	Terminal Board 30
TM31	200	Terminal Module 31
TM41	201	Terminal Module 41
TM15DI/DO	204	Terminal Module 15 (for SINAMICS)
TM120	207	Temperature evaluation with safe electrical isolation
DO encoder	300	Drive object encoder
HLA	70	Hydraulic Linear Axis
RIC	21	Renewable Infeed Control

Execution groups in the DCC editor

Description

Exactly one DCC chart can be created per drive object (DO = drive object). This DCC chart can contain up to ten execution groups.

Execution groups in the DCC editor

Execution groups are groups of blocks. The blocks of an execution group are started in a defined sequence at a specified time and are calculated cyclically one after the other within a specified sampling time.

In the DCC editor a maximum of ten execution groups (execution group 1 to 10) can be defined per drive object (DO) and thereby per basic chart (see also: SINAMICS system integration > Execution sequence, creating new execution groups). In the project navigator of STARTER / SCOUT (context menu of the chart, menu command **Set execution groups ...**) fixed or free execution groups of the drive object can be assigned to the previously defined execution groups.

Note

The **Interconnection with execution group**... command is only supported with DCC SIMOTION.

⚠ WARNING

Changing the assignment of an execution group

If the assignment of an execution group is changed in the **Set Execution Groups** window (or in parameter p21000[]), the affected execution group is initially logged off during time slice management and then logged on once again with the new assignment. The execution group is not calculated during the period between logging off and logging back on. The log-on and log-off take place in a background process of the drive unit; the duration is therefore not defined and is determined by the current CPU time load of the drive unit. (This affects the path of the output signal in the case of time-dependent blocks, e.g. the DIF differentiator.) Prior to the first calculation cycle after logging back on, internal status variables of the blocks are partially reset. For both of these reasons, this can result in jumps in the output signal of blocks, which for example can affect the torque/force setpoint and (in the case of operated axes) also the torque/actual force value. Logic signals can also acquire an unexpected state at this point of operation.

If the change to the execution group also results in a change to the sampling time, internal constants or factors are automatically adjusted for time-dependent blocks (BF, DCA, DIF, DT1, INT, MFP, PCL, PDE, PDF, PIC, PST, PT1, RGE, RGJ, WBG). If you use these blocks in the following execution groups, you must assign parameter p2048 the value of the isochronous master cycle clock:

- Receive AFTER IF1 PROFIdrive PZD
- Send BEFORE IF1 PROFIdrive PZD
- Receive AFTER IF1 PROFIdrive flexible PZD
- Receive AFTER IF2 PZD
- Send BEFORE IF2 PZD
- Receive AFTER IF2 flexible PZD

Note that the execution groups created in the chart are only visible in the **Set Execution Groups** window after the chart has been compiled.

Parameter r21002 displays the basic sampling time hardware.

Parameter r21003 displays the basic sampling time software.

For more detailed information on these parameters, see the online help for DCC parameters or the "DCC blocks" Function Manual.

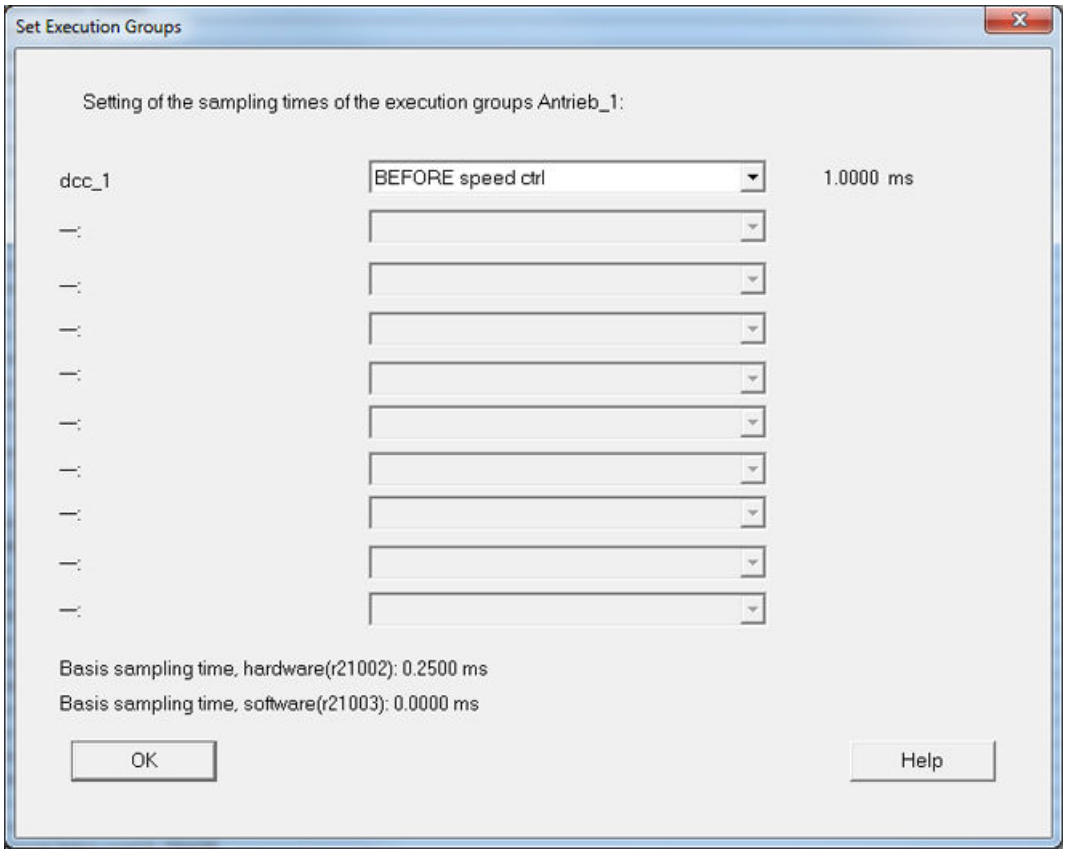


Figure 7-636 **Set Execution Groups** window

The execution groups created in the DCC editor must be assigned to the "Fixed execution groups" and the "Free execution groups" of the SINAMICS drive object.

Note

For parameter p21000 (Properties execution group) and, therefore, in the context menu of the "Set sampling times" chart, write access for the enabled controller is blocked in at least one drive axis or infeed. The old value automatically takes effect again in this case. If necessary, switch off all drive axes and infeeds, and make a new attempt.

Fixed execution groups

Fixed execution groups

A "fixed execution group" is called at a fixed position in the system execution using the sampling time of the corresponding system function.

The following fixed execution groups are available for SINAMICS as of FW Version 2.5:

- **Read in AFTER digital inputs**
  This execution group is called after the current values of the digital inputs have been read in on this drive object type and the corresponding binector outputs have been written.
  The sampling time of this execution group corresponds to the sampling time of the CU inputs/outputs (p0799) or the digital inputs/outputs of the TB30, TM31 and TM41 (p4099[0]).
- **Output BEFORE digital outputs**
  This execution group is called before the digital outputs are output on this drive object type.
  The sampling time of this execution group corresponds to the sampling time of the CU inputs/outputs (p0799) or the inputs/outputs of the TB30, TM31 and TM41 (p4099[0]).
- **Read in AFTER analog inputs**
  This execution group is called after the current values of the analog inputs have been read in on this drive object type and the corresponding binector outputs have been written.
  The sampling time of this execution group corresponds to the sampling time of the inputs/outputs of the TB30, TM31 and TM41 (p4099[1]).
- **Output BEFORE analog outputs**
  This execution group is called before writing is performed on the analog outputs.
  The sampling time of this execution group corresponds to the sampling time of the inputs/outputs of the TB30, TM31 and TM41 (p4099[1]).
- **BEFORE speed controller**
  This execution group is called before the speed controller additional setpoint values "n_reg n_set1" (p1155) and "n_reg n_set2" (p1160) are read into FP3080. The sampling time of the speed controller (p0115[1]) produces the call, but a minimum sampling time of 1 ms is required.
- **BEFORE speed setpoint channel**
  This execution group is called before function block diagrams 3010, 3020, 3030, 3040 and subsequent charts are calculated, if the setpoint channel has been activated (p0108.8 = 1). If no setpoint channel has been configured (p0108.8 = 0), calculation takes place before function block diagram 3095. The setpoint channel sampling time produces the call (p0115[3]).
- **BEFORE position controller**
  This execution group is called after the actual position value preparation (function block diagram 4010) has been calculated and before the position controller (function block diagrams 4015, 4020 and 4025) is calculated. The sampling time of this execution group corresponds to the sampling time of the position controller (p0115[4]).
- **BEFORE actual position value**
  This execution group is called before the actual position value preparation (function block diagram 4010) and the position controller (function block diagrams 4015, 4020, and 4025) are calculated. The sampling time of this execution group corresponds to the sampling time of the position controller (p0115[4]). (Available as of V4.3).
- **BEFORE basic positioner**
  This execution group is called before the basic positioner function module (function block diagrams 3610 to 3650) is calculated. The sampling time of this execution group corresponds to the sampling time of the basic positioner function module (p0115[5]).

- **BEFORE standard technology controller**
  This execution group is called before the technology controller function module is calculated (p0108.16 = 1) (function block diagrams 7950, 7954 and 7958). The sampling time of this execution group corresponds to the sampling time of the technology controller (p0115[6]).

- **Receive AFTER IF1 PROFIdrive PZD** [1)]
  This execution group is called after cyclic process data has been received via communication interface IF1 (e.g. via the integrated PROFIBUS interface) and output to connector outputs r2050[..], r2060[..], binector outputs r2090 - r2093 and connector binector transformers r2094 and r2095.
  The sampling time of this execution group corresponds to the PROFIdrive PZD sampling time.
  As of SINAMICS FW Version 2.5, communication interface IF1 (Interface 1) is always used by the PROFIBUS interface inside the CU or, if a CBE20 has been inserted in the option slot, PROFINET.
  Parameter p0092 can be used to set whether the utilization calculation (r9976) evaluates the execution group for isochronous operation (p0092 = 1) or non-isochronous operation (p0092 = 0) during SINAMICS ramp-up.
  Non-isochronous operation (r2043.1 = 0):
  For SINAMICS **software versions V2.5 and V2.6**, the PROFIBUS data is sent and received immediately after each other at the start of the IF1 PROFIdrive PZD sampling time.
  At the start of the IF1 PROFIdrive PZD sampling time set in p2048, the PROFIBUS data is sent first and then the process data received. In the case of non-isochronous cyclic communication at the IF1 communication interface, the execution group is cyclically calculated with the sampling time of the IF1 communication interface (p2048), i.e. after writing the connector outputs for the process data (PZD) r2050[..], r2060[..], r2090 - r2093 and after calculating the connector binector transformer r2094 and r2095.
  As of SINAMICS **software version V4.3**, the PROFIBUS data is received and sent at two separate times. The process data is received at the start of the IF1 PROFIdrive PZD sampling time set in p2048. In the case of non-isochronous cyclic communication at the IF1 communication interface, the execution group is cyclically calculated with the sampling time of the IF1 communication interface (p2048) after receiving the data, i.e. after writing the connector outputs for the process data (PZD) r2050[..], r2060[..], r2090 - r2093 and after calculating the connector binector transformer r2094 and r2095.
  For calculating the utilization of system r9976 correctly, p092 must be = 0 (== factory setting) in non-isochronous operation.
  Isochronous operation (r2043.1 = 1):
  In isochronous operation, the times at which the data is received from the master (T_o), the data is sent to the master (T_i) and the DP cycle time (T_{DP}) [3)] are configured in the master.
  Internally, the calls to T_o and T_i are realized using a state machine that is called cyclically with sampling time T_{Zu}, e.g. for standard servo drive objects with the current controller sampling time of 125 µs. The sampling time T_{Zu} = 125 µs, however, is at least as large as the largest current controller sampling time (e.g. for the vector, 250 µs, 375 µs or 500 µs).
  This means only integer multiples of T_{Zu} >= 125 µs can be set [3)] as times T_i and T_o for the drive.
  $\rightarrow T_i = n_i \bullet T_{Zu}$:

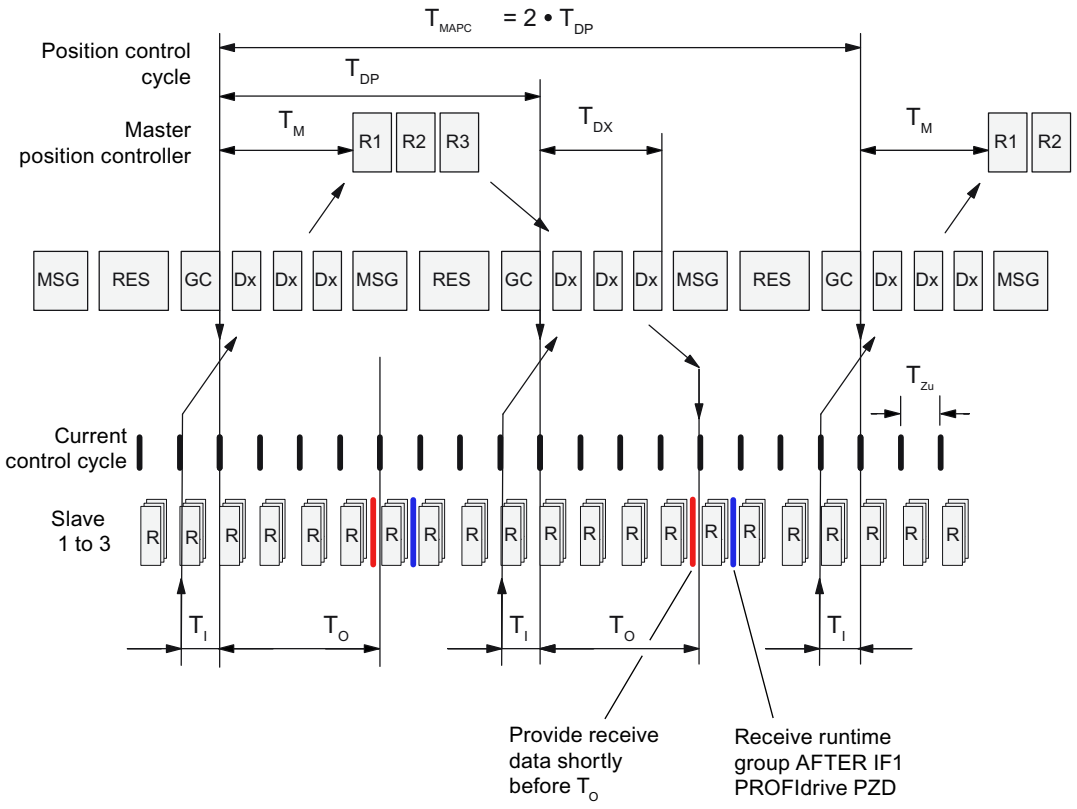


Figure 7-637 Calling the "Receive AFTER IF1 PROFIdrive PZD" or "Receive AFTER IF2 PROFIdrive PZD" execution group for isochronous PROFIBUS communication. See also [3)].

Receive data (sent by the master) is always processed and made available at the end of the sampling time (n_o -1) T_{Zu}, so that the received process data remains valid until the start of the next sampling period at the time $T_o = n_o \bullet T_{Zu}$ at r2050[..], r2060[..], r2090 - r2093. Thus, in the last sampling time T_{Zu} before T_o. The "Receive AFTER IF1 PROFIdrive PZD" execution group is then calculated in the following first sampling time T_{Zu} after T_o.

Note

Please take note that two connector-binector converters (function block diagram 2468: p2099, r2094, r2095) are NOT called synchronously in isochronous mode but may be processed within sampling time p2048 at any time depending on the computing time utilization.

The calculation must be completed within the sampling time T_{Zu}; otherwise, warning A01053 "System overload detected" or a time-slice overflow (F01205) will occur. For this reason, only the absolute minimum number of DCBs that are necessary for the required function should be calculated in this execution group.
The computing time available for this execution group decreases the more drive axes are calculated on the CU since the current controllers (and also the speed controller for the servo) of the drive axes are also calculated in the sampling time p0115[0] = T_{Zu} (>=125 µs).
If the utilization calculation is to be executed during ramp-up of the CU for isochronous operation (unfavorable case regarding degree of utilization), p092 = 1 must be set.
Otherwise (p092 = 0) the utilization in r9976 will be updated only after switchover to

isochronous operation. The increased maximum overall utilization in isochronous operation will be displayed only in r9976.
If you want to calculate a larger number of DCBs, please check whether you would not do better to configure this using the "Receive AFTER IF1 PROFIdr. flexible PZD" execution group.

Note

In the case of this execution group, please note that the higher computing time utilization that applies in isochronous operation is not actually taken into account in the utilization calculation until the time of the transition (specified by the PROFIBUS master) to isochronous operation. After the drive device has ramped up, this means the utilization of the complete system initially still lies within the valid range and it is only when the transition is made to isochronous operation that the drive device shuts down with fault F1054 "System limits exceeded".

- **Send BEFORE IF1 PROFIdrive PZD** [1)]
  This execution group is called before cyclic process data is sent via communication interface IF1 (e.g. the integrated PROFIBUS interface), i.e. before calculating the connector binector transformer p2080 – p2084 and reading in the connector inputs p2051[..], p2061[..].
  The sampling time of this execution group corresponds to the PROFIdrive PZD sampling time.
  As of SINAMICS firmware Version 2.5, communication interface IF1 is always used by the PROFIBUS interface inside the CU or, if a CBE20 is inserted in the option slot, PROFINET.
  Parameter p0092 can be used to set whether the utilization calculation (r9976) evaluates the execution group for isochronous operation (p0092 = 1) or non-isochronous operation (p0092 = 0) during SINAMICS ramp-up.
  Non-isochronous operation (r2043.1 = 0):
  For SINAMICS software versions **V2.5** and **V2.6**, the PROFIBUS data is sent and received (in this sequence) immediately after each other at the start of the IF1 PROFIdrive PZD sampling time.
  The "Send BEFORE IF1 PROFIdrive PZD" execution group is processed at the end of the IF1 PROFIdrive PZD sampling time before the data is sent at the start of the next sampling time. After computing the execution group, at the end of the sampling time, the connector binector transformer p2080 – p2084 is still calculated and the connector inputs p2051[..], p2061[..] read in. This means all send data is present completely at the end of the sampling time and will be sent at the start of the next sampling time.
  As of SINAMICS software version **V4.3**, the PROFIBUS data is received and sent at two separate times.
  The "Send BEFORE IF1 PROFIdrive PZD" execution group is processed at the end of the IF1 PROFIdrive PZD sampling time before the complete data is still sent at the end of the sampling time. For the non-isochronous (cyclical) communication at the communications interface IF1, the execution group is calculated cyclically with the sampling time of the interface IF1 (p2048) (before sending the data), i.e. also before calculating the connector binector transformer p2080 – p2084 and reading in the connector inputs p2051[..], p2061[..].
  For calculating the utilization of system r9976 correctly, p092 must be = 0 (== factory setting) in non-isochronous operation.
  Isochronous operation (r2043.1 = 1):
  In isochronous operation, the times at which the data is received from the master (T_o), and sent to the master (T_i) as well as the DP cycle time (T_{DP}) [3)] are configured in the master.
  Internally, the calls to T_o and T_i are realized using a state machine that is called cyclically with sampling time T_{Zu}, e.g. for standard servo drive objects with the current controller sampling time of 125 µs. The sampling time T_{Zu} = 125 µs, however, is at least as large as the largest current controller sampling time (e.g. for the vector 250 µs, 375 µs or 500 µs).
  This means only integer multiples of T_{Zu} >= 125 µs can be set [3)] as times T_i and T_o for the drive.

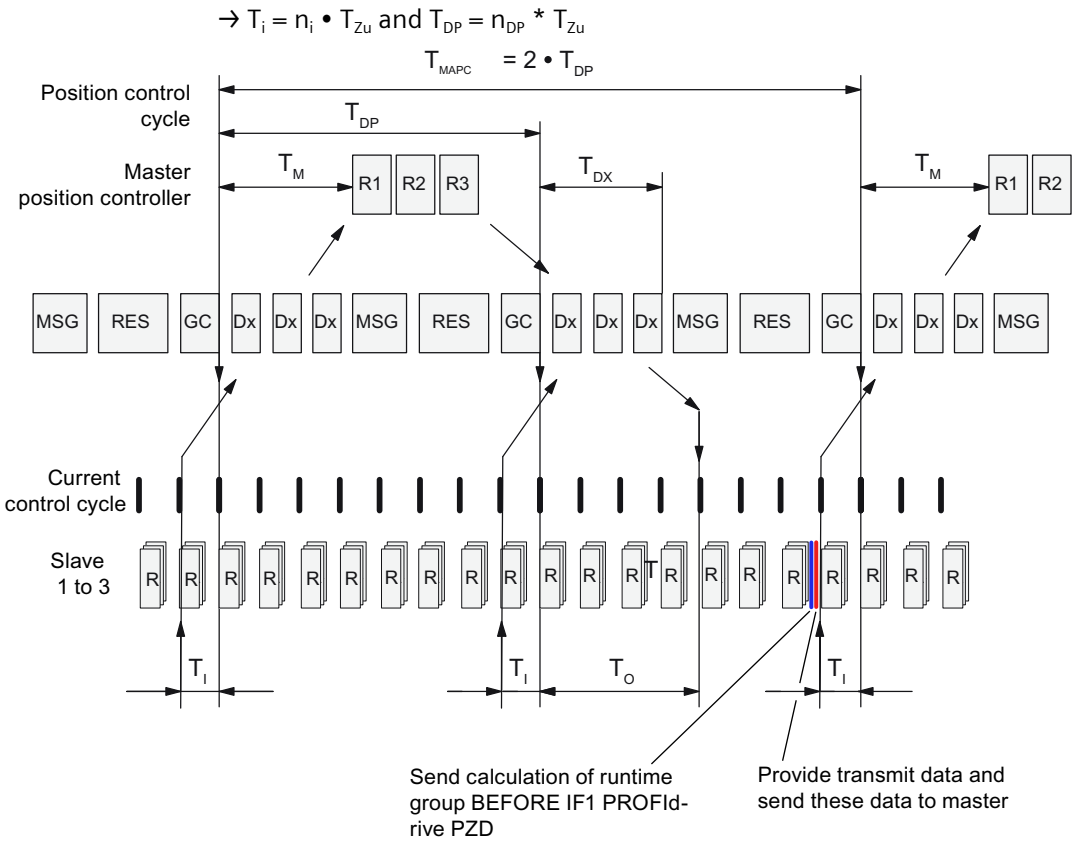


Figure 7-638 Calling the "Send BEFORE IF1 PROFIdrive PZD" or "Send BEFORE IF2 PROFIdrive PZD" execution group for isochronous PROFIBUS communication. See also [3)].

The "Send BEFORE IF1 PROFIdrive PZD" execution group is calculated in the last sampling time T_{Zu} before the data is sent to the master and before T_i and reading in the connector inputs p2051[..], p2061[..].

Note

Please note that five connector-binector converters (function block diagram 2472: p2080 ..., r2089) are also NOT called synchronously in isochronous operation but may be processed within sampling time p2048 at any time depending on the computing time utilization.

The calculation of the execution group must be completed within sampling time T_{Zu}]; otherwise, a fault F1054 "System limit exceeded" (V4.3), a warning A01053 "System overload measured" (V2.x) or a time-slice overflow (F01205) occurs. For this reason, only the absolute minimum number of DCBs that are necessary for the required function should be calculated in this execution group.
The computing time available for this execution group decreases as the number of drive axes calculated on the CU increases, since the current controller (and the speed controllers for the servo) of the drive axes are also calculated in sampling time p0115[0] = T_{Zu}.
If the utilization calculation during the ramp-up of the CU has already executed the isochronous operation (the more unfavorable configuration regarding the utilization of the CU), then p092 must be set to = 1. Otherwise (p092 = 0) the utilization in r9976 will be updated only after switchover to isochronous operation. The increased maximum overall utilization in isochronous operation will be displayed only in r9976.

Note

In the case of this execution group, please note that the higher computing time utilization that applies in isochronous operation is not actually taken into account in the utilization calculation until the time of the transition (specified by the PROFIBUS master) to isochronous operation. After the drive device has ramped up, this means the utilization of the complete system initially still lies within the valid range and it is only when the transition is made to isochronous operation that the drive device shuts down with fault F1054 "System limits exceeded".

- **Receive AFTER IF1 PROFIdr. flexible PZD** [1)]
  This execution group is called after cyclic process data has been received via communication interface IF1 (e.g. via the integrated PROFIBUS interface) and output to connector outputs r2050[..], r2060[..], binector outputs r2090 - r2093 and connector binector transformers r2094 and r2095.
  The sampling time of this execution group corresponds to the PROFIdrive PZD sampling time. The only difference from the Receive AFTER IF1 PROFIdrive PZD execution group is the way in which the execution group behaves in isochronous mode. Even in isochronous mode this execution group is called using the PROFIdrive PZD sampling time configured in the master as with any other sampling time. This means that initially all shorter sampling times are called at time T_o depending on the validity of the receive data (current controller; speed controller if applicable). This execution group is called first only when the T_{DP} sampling time starts to be processed. You basically do not know how often the shorter sampling times with higher priority of the current and speed controller are calculated before this execution group starts to be processed. In addition, processing of this execution group is interrupted due to shorter sampling times.
  This execution group should only be used if IF1 is usually operated in isochronous mode. The benefit of this execution group is that considerably more blocks can be calculated than in the Receive AFTER IF1 PROFIdrive PZD execution group because the calculation does not have to be completed after current controller sampling time p0115[0]. However (due to the higher priority and interruption caused by shorter sampling times), there is no longer a fixed synchronism between T_o and the time the execution group is called.

- **Receive AFTER IF2 PZD** [2)]
  The CAN bus interconnection CBC10, the CBE20, or the integrated PROFIBUS/PROFINET interface (see description p8839, p8815) on the CU can all serve as communications interface IF2. This execution group is available only on the CU_S, CU_G, CU_GM, CU_GL, SERVO, VECTOR, VECTORMV, VECTORSL and VECTOR_GL drive object types.
  Non-isochronous operation (PROFINET, PROFIBUS, or CAN bus):
  The execution group is called once the connectors r8850[..], r8860[..], r8890 - r8893 have been described with the receive data and the calculation of the connector-binector converters (function block diagram 2485.7: r8894 and r8895) has been executed. The sampling time of this execution group corresponds to the IF2 PZD sampling time in p8848.
  Isochronous operation (r2043.1 = 1):
  As of SINAMICS V4.4, isochronous PROFINET/PROFIBUS operation is also supported for the communications interface IF2. (See SINAMICS S120 Function Manual FH1 → Applications → Parallel operation of communications interfaces.)
  In isochronous operation, the times at which the data is received from the master (T_o), the data is sent to the master (T_i) and the DP cycle time (T_{DP}) [3)] are configured in the master. Internally, the calls to T_o and T_i are realized using a state machine that is called cyclically with sampling time T_{Zu}, e.g. for standard servo drive objects with the current controller sampling time of 125 µs. The sampling time T_{Zu} = 125 µs, however, is at least as large as the largest current controller sampling time (e.g. for the vector 250 µs, 375 µs, or 500 µs).
  This means only integer multiples of T_{Zu} >= 125 µs can be set as times T_i and T_o for the drive [3)].
  → $T_i = n_i \bullet T_{Zu}$ and $T_{DP} = n_{DP} * T_{Zu}$ See figure 3-2
  Receive data (sent by the master) is always processed and made available at the end of the sampling time (n_o -1) T_{Zu}, so that the received process data remains valid until the start of the next sampling period at the time $T_o = n_o \bullet T_{Zu}$ at r8850[..], r8860[..], r8890 - r8893 (function block diagram 2485). Thus, in the last sampling time T_{Zu} before T_o. The "Receive AFTER IF2 PZD" execution group is then calculated in the first sampling time T_{Zu} to follow T_o.

Note

Please take note that two connector-binector converters (function block diagram 2485: p8899, r8894, r8895) are NOT called synchronously in isochronous mode but may be processed within sampling time p8848 at any time depending on the computing time utilization.

The calculation must be completed within the sampling time T_{Zu}; otherwise, warning A01053 "System overload detected" or a time-slice overflow (F01205) will occur. For this reason, only the absolute minimum number of DCBs that are necessary for the required function should be calculated in this execution group. The computing time available for this execution group decreases as the number of drive axes calculated on the CU increases, since the current controllers (and also the speed controllers for the servo) of the drive axes are also calculated in the sampling time p0115[0] = T_{Zu} (>=125 µs). If the utilization calculation is to be executed during ramp-up of the CU for isochronous operation (unfavorable case regarding degree of utilization), p092 = 1 must be set. Otherwise (p092 = 0) the utilization in r9976 will be updated only after switchover to isochronous operation. The increased maximum overall utilization in isochronous operation will be displayed only in r9976. If you want to calculate a larger number of DCBs, please check whether you would not do better to configure this using the "Receive AFTER IF2 PROFIdr. flexible PZD" execution group.

Note

In the case of this execution group, please note that the higher computing time utilization that applies in isochronous operation is not actually taken into account in the utilization calculation until the time of the transition (specified by the PROFIBUS master) to isochronous operation. After the drive device has ramped up, this means the utilization of the complete system initially still lies within the valid range and it is only when the transition is made to isochronous operation that the drive device shuts down with fault F1054 "System limits exceeded".

- **Send BEFORE IF2 PZD** [2)]
  The CAN bus interconnection CBC10, the CBE20, or the integrated PROFIBUS/PROFINET interface (see description p8839, p8815) on the CU can all serve as communications interface IF2. This execution group is available only on the CU_S, CU_G, CU_GM, CU_GL, SERVO, VECTOR, VECTORMV, VECTORSL and VECTOR_GL drive object types.
  Non-isochronous operation (PROFINET, PROFIBUS, or CAN bus):
  The execution group is called before the binector connector transformers p8880 – p8884 are calculated and the before connector inputs p8851[..], p8861[..] with the send data are read. The sampling time of this execution group corresponds to the IF2 PZD sampling time in p8848. See function block diagrams 2487 and 2493.
  Isochronous operation (r2043.1 = 1):
  As of SINAMICS V4.4, isochronous PROFINET/PROFIBUS operation is also supported for the communications interface IF2.
  (See SINAMICS S120 Function Manual FH1 → Applications → Parallel operation of communications interfaces.)
  In isochronous operation, the times at which the data is received from the master (T_o), and sent to the master (T_i) as well as the DP cycle time (T_{DP}) [3)] are configured in the master. Internally, the calls to T_o and T_i are realized using a state machine that is called cyclically with sampling time T_{Zu}, e.g. for standard servo drive objects with the current controller sampling time of 125 µs. The sampling time T_{Zu} = 125 µs, however, is at least as large as the largest current controller sampling time (e.g. for the vector 250 µs, 375 µs, or 500 µs).
  This means only integer multiples of T_{Zu} >= 125 µs can be set as times T_i and T_o for the drive [3)].
  → $T_i = n_i \bullet T_{Zu}$ and $T_{DP} = n_{DP} * T_{Zu}$ See also figure 3-3
  The "Send BEFORE IF2 PZD" execution group is calculated in the last sampling time T_{Zu} before the data is sent to the master and before T_i and reading in the connector inputs p8851[..], p8861[..].

  Note

  Please note that five connector-binector converters (function block diagram 2489: p8880 ..., r8889) are NOT called synchronously in isochronous mode but may be processed within sampling time p8848 at any time depending on the computing time utilization.

  The calculation of the execution group must be completed within sampling time T_{Zu}]; otherwise, a fault F1054 "System limit exceeded" (V4.3), a warning A01053 "System overload measured" (V2.x) or a time-slice overflow (F01205) occurs. For this reason, only the absolute minimum number of DCBs that are necessary for the required function should be calculated in this execution group.
  The computing time available for this execution group decreases as the number of drive axes calculated on the CU increases, since the current controller (and the speed controllers for the servo) of the drive axes are also calculated in sampling time p0115[0] = T_{Zu}.
- **Receive AFTER IF2 flexible PZD** [1)]

This execution group is called after cyclic process data (PZD) has been received via communication interface IF2 (e.g. via the integrated PROFIBUS interface) and output to connector outputs r8850[..], r8860[..], binector outputs r8890 - r8893 and connector binector converters r8894 and r8895.
The sampling time of this execution group corresponds to the PROFIdrive PZD sampling time. The only difference from the Receive AFTER IF2 PROFIdrive PZD execution group is the way in which the execution group behaves in isochronous mode. Even in isochronous mode this execution group is called using the PROFIdrive PZD sampling time configured in the master as with any other sampling time. This means that initially all shorter sampling times are called at time T_o depending on the validity of the receive data (current controller; speed controller if applicable). This execution group is called first only when the T_{DP} sampling time starts to be processed. You basically do not know how often the shorter sampling times with higher priority of the current and speed controller are calculated before this execution group starts to be processed. In addition, processing of this execution group is interrupted due to shorter sampling times. This execution group should only be used if IF2 is usually operated in isochronous mode. The benefit of this execution group is that considerably more blocks can be calculated than in the Receive AFTER IF1 PROFIdrive PZD execution group because the calculation does not have to be completed after current controller sampling time p0115[0]. However (due to the higher priority and interruption caused by shorter sampling times), there is no longer a fixed synchronism between T_o and the time the execution group is called.

1) IF1 is the abbreviation for communication interface 1. With SINAMICS V2.5 and SIMOTION V4.1, this refers to the integrated PROFIBUS interface or - if a CBE20 has been inserted in the option slot - the PROFINET interface. IF1 supports the PROFIdrive profile and isochronous operation.
2) IF2 is the abbreviation for communication interface 2. IF2 of the CU320 and CU320-2 can only be used by the CAN bus if a CBC10 module has been inserted in the option slot and SINAMICS software as of V2.5 is used. The CBE20 or integrated PROFIBUS/PROFINET interface can be assigned to the IF2 via parameters (p8839, p8815). (See SINAMICS S120 Function Manual FH1 → Applications → Parallel operation of communications interfaces.) As of SINAMICS V4.4, isochronous mode is also supported on IF2.
3) For T_{DP}, T_i and T_o, the formulas and limit values listed in the Function Manual FH1→ Communication → Communication via PROFIBUS DP → Motion Control with PROFIBUS → "Time settings and Meanings" table apply.

The PROFIBUS functionalities described here also apply to PROFINET.

Not every fixed execution group is available on every drive object type. This means, for instance, that SERVO, VECTOR, VECTORMV... drive object types do not contain any digital or analog inputs; therefore, the fixed execution groups for the digital inputs/outputs and analog inputs/outputs are not available here.

Please ensure that the sampling time for DCC SINAMICS is restricted to below 1 ms. As a property of an execution group, select a fixed execution group in which the sampling time for the assigned system function is < 1 ms. This execution group will thus only be called with the sampling time of 1 ms, and will thereby deviate from the assigned system function. Fault F51004 (see r0947) is set to indicate this deviation. The fault value of the fault (r0949) + 1 gives the number of the execution group where the deviation has occurred.

Example for the automatic limitation of the sampling time:

On the SERVO drive object type, select "BEFORE speed controller" as a fixed execution group.
The associated system function is the speed controller.
The sampling time of the speed controller is p0115[1] = 125 µs in the factory setting of p0112 = 3.
In contrast to the sampling time of the speed controller, the sampling time of the DCC execution group is set automatically to 1 ms.
Also refer to the description of parameter p0112 in the SINAMICS S List Manual.

⚠ WARNING

In the case of an online change to the sampling time (execution group) the START method of the block is called. Initializations in the START method can result in jumps in the signal path.

Free execution groups

Free execution groups

The "free execution groups" are only defined by their sampling time (p21000 = 1 to 256 and p21000 = 1001 to 1096). The sampling times are provided centrally for all drive objects on a SINAMICS drive unit. A sampling time can be used simultaneously by several drive objects and several DCC execution groups. The sampling times are created in two different ways. The two methods have different limits for the maximum possible number of different sampling times.

- Free execution groups whose sampling times are created in the hardware:
  With the hardware sampling times, in p21000[0...9] each integer multiple of the basic sampling time (read-only in r21002) ranging from 1 * r21002 to 256 * r21002 can be generated with the following limits:
  - Minimum sampling time = 1 ms
  - Maximum sampling time = (r21003 - r21002) < r21003 (approx. 8 ms; on the D410 of the DP cycle).
  - On the D410 drive unit, r21003 always matches the DP cycle (PROFIBUS T_DP cycle clock) or with the send cycle clock for PROFINET. If 1 ms is configured as DP cycle / send cycle for PROFINET, this means no hardware sampling times (but only software sampling times) from DCC are used on this device.
  - The number of hardware sampling times on a drive device is limited. The number of still available hardware sampling times can be read from r7903 (as of SINAMICS V2.6). Further information is contained in Chapter DCC for SINAMICS → Overview → Computing time load, memory requirement and assignment of the HW sampling times → Number of possible different hardware sampling times (Page 5555).

    Note

    As regards offline configuration using STARTER commissioning software, values 0 - 256 can be entered in p21000[0..9], even if this violates the limits stated above for the hardware sampling times from 1 ms ... <r21003. This will only be detected once the Control Unit has been downloaded, and will result in the use of a replacement value and the F51004 fault. If the set value is too small, 1 ms will be set as substitute value; if it is too large, the next largest SW sampling time will be set.

  - On the drive objects of the CU, TB30, TM15 DIDO, TM 31 and TM41, the sampling time for additional functions p0115[0] = 4 ms is preset. If you want to configure a DCC execution group with a smaller sampling time on these drive objects, you should first set the sampling time for the additional functions p0115[0] on this drive object to the value of the smallest desired sampling time. To do this, p0009 = 3 must first be set. Only then is it possible to change p0115[0]. For the new value for p0115[0] to take effect, p0009 = 0 must be reset.
- Free execution groups whose sampling times are created in the software:
  The software sampling times are generated as an integer multiple of the basic value for software sampling times (read out in parameter r21003).
  The possible set values for the software sampling times (1 * r21003 - 96 * r21003) can be taken from the parameter description for p21000 (see Function Manual, *Description of the standard DCC blocks*, in the appendix in the *Parameters* section).

⚠ WARNING
The START method of the block is called for an online change to the sampling time (execution group). Initializations in the START method can result in jumps in the signal path.

Execution sequence order, creating new execution groups

When a CFC is created, a new execution group with the same name as the chart will be created automatically. All blocks that are inserted into the chart will be automatically assigned in the insertion sequence of this execution group.

The computing sequence of the blocks within the execution group is specified by the configuration. If no other execution sequence is specified by the user, the execution sequence corresponds to the insertion sequence of the blocks within an execution group.

If the execution sequence with this execution group is to be changed, new execution groups added or the assignment of blocks to the execution groups changed, these properties can be changed in the execution editor. To do this, go offline in the STARTER and then open the chart. In the DCC editor, the **execution sequence** button can be used to activate and deactivate the execution editor.

The center column of the execution editor contains the chart with its chart name. The execution group(s) are arranged hierarchically below. If you want to create additional execution groups, select the **Insert execution group...** item from the context menu of the chart (right-click the chart). The individual blocks can be moved between the execution groups or within an execution group using drag-and-drop. The details given in the Pos column (e.g. 1 / 2) correspond to the name in the bottom line in the right-hand part of the block header with a colored background. The 1st digit stands for the execution group (between 1... 10) and the 2nd digit for the sequence within the execution group. The execution sequence can be changed by dragging & dropping into the center column.

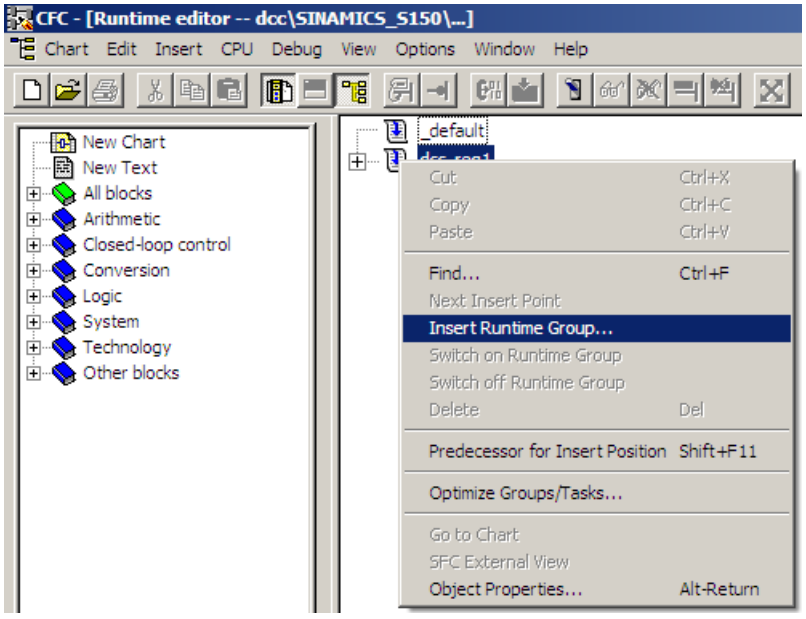


Figure 7-639 Context menu, **Insert execution group...**

Note

1. The assignment of identical names for DCCs in different drive objects is not permitted.
2. The assignment of identical names for execution groups in different drive objects is not permitted.

If the same fixed or free execution group of the drive object is assigned to several execution groups in the DCC editor, the execution groups of the DCC editor will be calculated in the sequence from top (calculated first) to bottom (calculated last) as shown in the "Set Execution Groups (Sampling Times)" window.

Creating customer-specific parameters ("declare")

The properties of the input and output connections of blocks can be edited in the DCC editor in the Properties window of the block connection. Input connections may not be interconnected in the default setting of the DCC editor. The Properties window of the connection is opened by double-clicking the connection or by selecting **Object properties** in the context menu of the connection.

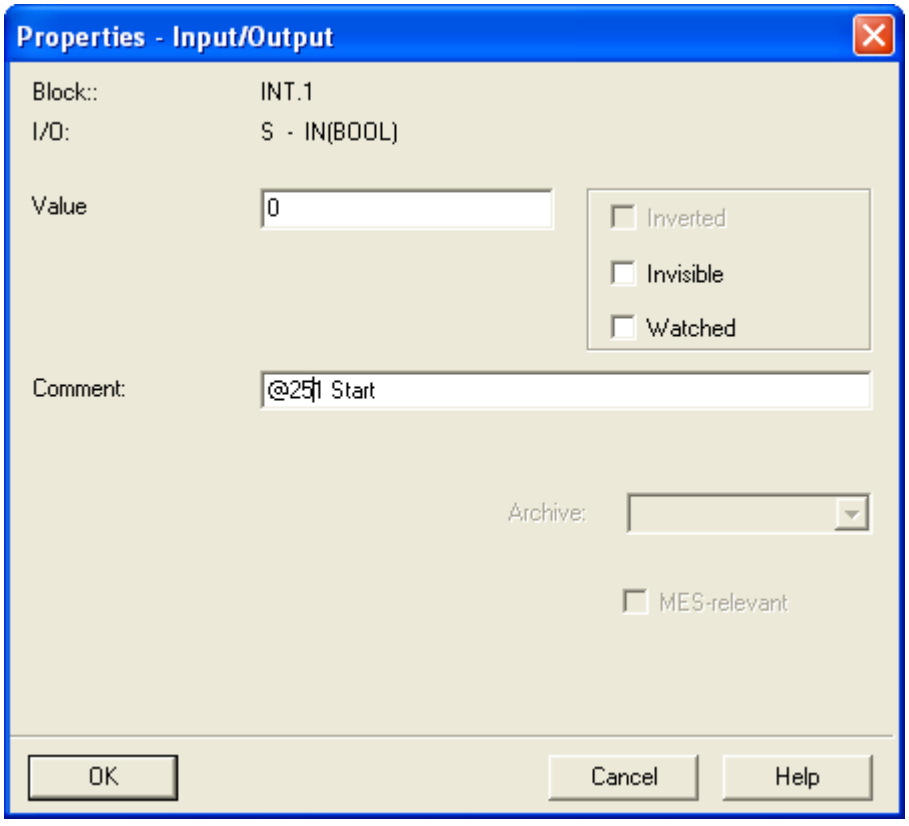


Figure 7-640 **Properties - Connection** window of the connection S (= Set) of the block INT (= INTegrator)

In SINAMICS, input and output connections of blocks can be published as parameters. "Publish" means that users create the parameters themselves (parameter numbers and parameter text). This is the prerequisite in order to interconnect the block connections using BICO parameters to the basic system in order to assign the values to the input connections using adjustable parameters and to monitor the values of the output connections using parameters. The parameter number 251 in the above figure and in the following figure is arbitrary.

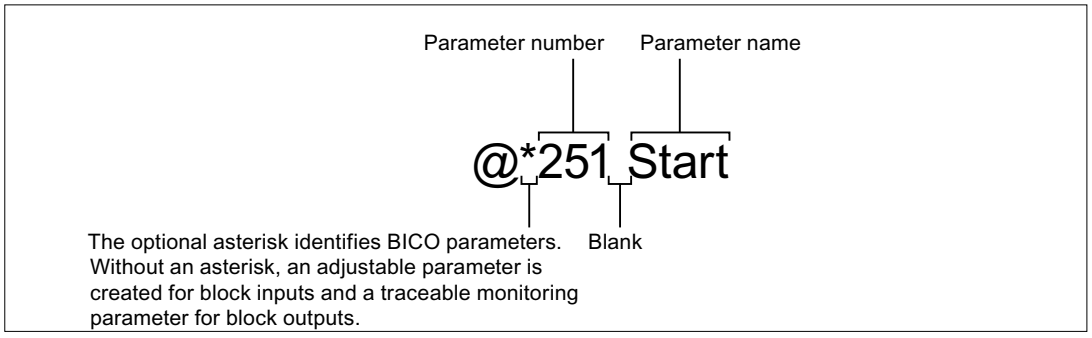


Figure 7-641 Structure of the block connection comment

The parameter name, which follows the parameter number separated by a space, is displayed in the STARTER expert list.

The data type of the published input/output is taken over by the block connection.

For the "publishing" of block inputs, a distinction must be made between adjustable parameters (without asterisk) and BICO parameters (with asterisk).

For the "publishing" of block outputs, the following distinction must be made:

- Without asterisk: pure monitoring parameter that cannot be interconnected and that can be recorded with the trace function.
- With asterisk: as for the case without asterisk, but additional interconnection possible.

Block inputs to be published may not be already interconnected in the default setting of the DCC editor. If necessary, an existing interconnection must first be deleted. To do this, click the interconnection in the chart to mark it and select **Delete interconnection(s)** in the context menu. In this DCC editor setting, the comment shown for a block input interconnected with a different block is always that of the connected block output.

If you want to remove this constraint, select **Options** > **Settings** > **Display ...** in the DCC editor and deactivate the **Interconnection comment** checkbox under **Parameters** and close the window with OK. Already interconnected block inputs can now be published without having to delete the interconnection first.

A parameter number may only be used just once in a chart. The DCC editor does not check when the parameter number is entered in the Properties window of the connection whether this parameter number is already used in the chart. The multiple use of the parameter numbers will be indicated as error only when the chart is compiled.

The parameter numbers of the charts are represented on the number range from p21500 to p25999 of the expert list of the STARTER. With each DCC chart, it is possible to specify where the parameter range of the chart is to begin. This can be performed in the **Parameters** tab in the Properties window of the chart in STARTER, using the parameter number base.

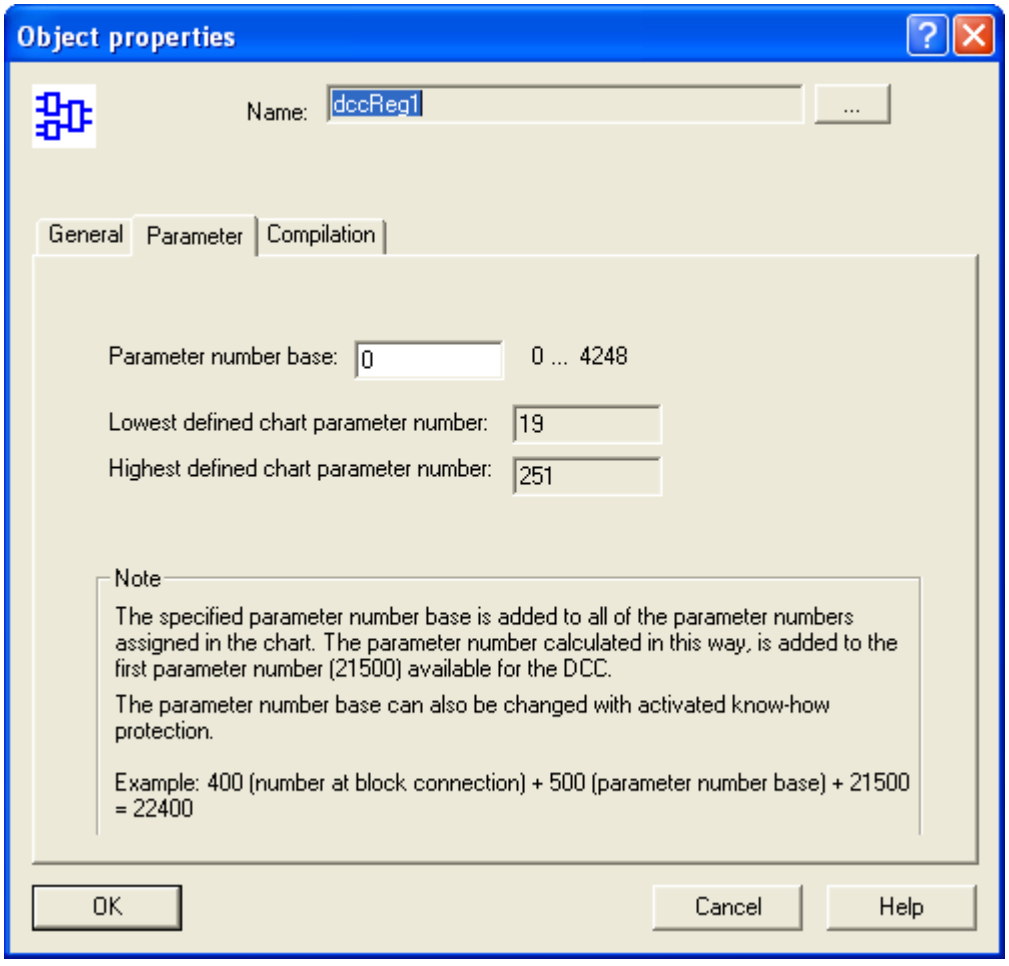


Figure 7-642 Properties window of the dccReg1 chart for setting the parameter number base

The parameter number in STARTER has the following form:

- Parameter number in the chart + base parameter number + 21500;
  where 21500 is the first parameter number available for DCC. The parameter number base must always be >= 0. The parameter number base can be changed.

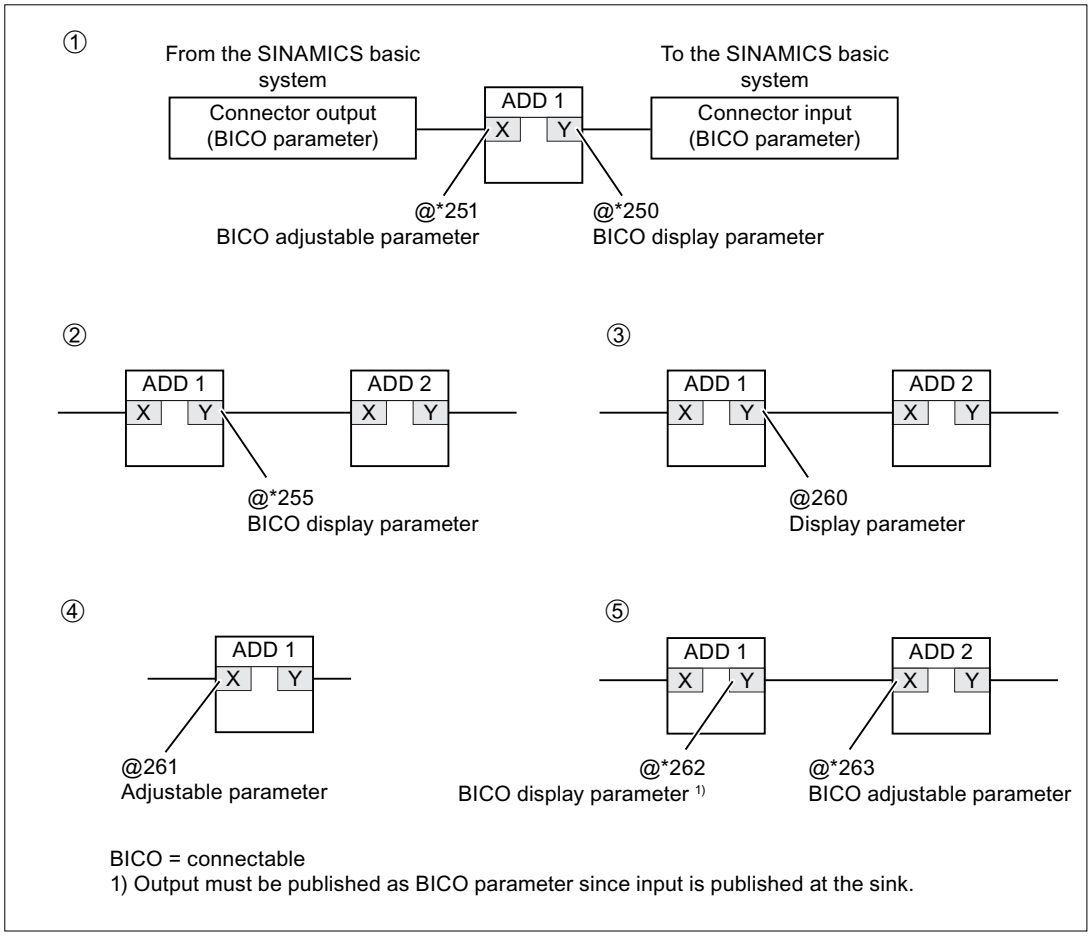


Figure 7-643 Examples of parameter definitions

Example 1: For an interconnection of the block connection with the SINAMICS basic system, the connection must be published as BICO parameter.

Example 2: Each block output connection can be published as BICO parameter. The signal on this output can be recorded with the trace.

Example 3: Each block output connection can be published as (not interconnectable with other BICO parameters) display parameter. The signal on this output can be recorded with the trace. The block input can be interconnected to every other non-published block input.

Example 4: The block input connection X of the ADD 1 block is published as adjustable parameter.

Example 5: Because the block input connection X of the ADD 2 is published as BICO parameter, the block output connection interconnected with it must also be published as BICO parameter. It is, however, possible in the DCC editor to interconnect ADD 1 Y with ADD 2 X when only the block input ADD 2 X is published. This results in an error message when the chart is compiled. The signal value of a block input published as BICO parameter cannot be recorded with the trace.

Note

No help function can be created for @parameters.

Display of the parameters

If the parameters configured on the block connections are to be displayed in the DCC editor, then select **Options** > **Settings** > **Display ...** and select **Comment** under **Connections**. The first eight characters of the comment are then displayed at the respective block connections in the block symbols.

Interconnection with SINAMICS parameters

You can interconnect with SINAMICS BICO parameters in the DCC editor.

Procedure

Interconnection can be performed as follows:

1. Creating customer-specific parameters ("Publishing", @ parameters); see Creating customer-specific parameters (Page 5543).

   Note

   In the DCC editor, block connections can initially be connected with the drive's BICO parameters without the need to publish them as parameters. However, this will cause errors when compiling the chart, since only block connections that have been published as @* parameters can be connected to the drive's BICO parameters.

2. Select the block connection to be interconnected.
3. Right-click and select **Interconnection with operand...** from the context menu. The **DCC Signal Selection** window appears.

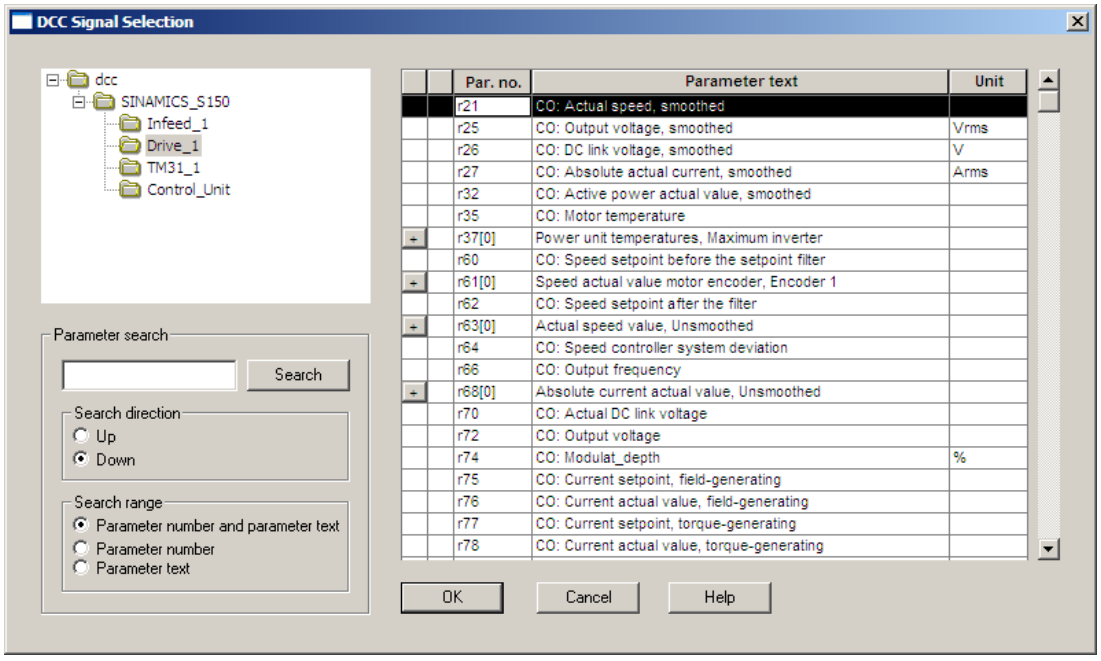


Figure 7-644 Signal selection

4. Select the parameter to be interconnected.
5. Click **OK** to close the window.

From this point on, the selected block connection is interconnected with the SINAMICS parameter and the parameter is displayed in the sheet bar of the DCC editor.

Note

BICO parameters defined in other DCC charts will not be displayed in the signal selection list.

DCC-local parameters can be directly interconnected in the CFC chart as an interconnection of block connections. This interconnection is then converted to a BICO interconnection.

Connections to BICO parameters in DCC charts on other DOs can be established graphically in the DCC editor using two charts open next to one another.

Note

With the BICO technology of the SINAMICS drive family you can interconnect BICO parameters with different data types, e.g. a BICO parameter of the FLOAT32 data type with a BICO parameter of the INT32 data type. In this case, the value range of the FLOAT32 data type exceeds the value range of the INT32 data type.

The SINAMICS basic system automatically ensures that in such cases the source parameters do not exceed the value ranges of the sinks.

You must explicitly ensure this restriction in the DCC chart for published BICO parameters, e.g. through an LIM block.

Computing time load, memory requirement and assignment of the HW sampling times

The following statements govern the use of DCC-SINAMICS on the SINAMICS S120, S150, G130, G150 and (on the SINAMICS_Integrated in) SIMOTION D4xx devices. Details about other devices are contained in the associated device documentation.

Computing time load for the SINAMICS V2.5 and V2.6 software versions

The processing of blocks requires computing time. This means with the computing of DCC execution groups, it is no longer possible to calculate the maximum quantity structure for drive objects of a CU3x0, D4xx and CX32.

The resulting computing time load is dependent on the following:

- Number of calculated execution groups
- Sampling time of execution groups
- Number of calculated blocks
- Calculated block types
- For some blocks, it depends on the parameterization (e.g. the setting of enable connections)

Parameter r9976 (system load; can be found on the CU's drive object) is available in the system as an online tool. The average value for the computing time load in r9976[1] should **always** be below 85.0%. Likewise, the maximum value for the computing time load in r9976[5] should **always** be below 85.0%. Exceeding the computing time load limit of 90% triggers warning A51003. The warning is reset if the computing time load falls below 88%.

The computing time load should not fall below the limit of 85.0% so that the response time of the STARTER/SCOUT engineering tool is not too greatly restricted.

If the computing time load increases to the point that it is no longer possible for all blocks and system functions assigned to a sampling time to be completed within this sampling time, the drive unit switches off and outputs fault F01205: "CU: Time-slice overflow".

Offline mode enables the SIZER engineering tool to make an approximate statement regarding whether a configuration can be calculated on a CU3x0 or D4xx. The additional computing time load is **not** taken into account by activating the DCCs.

Note

The computing time load reaches its maximum value when the following conditions are met:

- All necessary function modules are activated.
- All drives and infeeds are in the operational status.
- Closed-loop control in the final configuration is approved on all drives and infeeds.
- Configured isochronous data transfer is used.

The values displayed in r9976 are thoroughly smoothed internally. A change in the computing time load is therefore only displayed in full in r9976 after 2 - 3 minutes.

If, during OFFLINE configuration, it is not certain whether the limit value for the computing time load of 85% (r9976[1] and r9976[5]) will be adhered to, you can initially call the execution groups being used in a very large sampling time (e.g. 96 * r21003). After checking the actual system load in r9976, you can then gradually select execution groups with shorter sampling times right through to the execution group that is intended for the application in question.

During OFFLINE configuration, the following should be carried out:

- Activate all necessary function modules on the drive objects.
- Assign all necessary blocks to their execution groups.
- Establish all connections, both between the blocks and to the drive objects (as long as this still makes sense even if an increased sampling time is being used).

The following applies as a rough configuration aid to SERVO drives:

If one A-INF infeed is configured on a SINAMICS CU320 with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs) and three SERVO drive axes are configured with the speed and current controller sampling times set in the factory (p0112 = 3 == STANDARD → p0115[1] = p0115[0] =125 µs) and the function modules and without isochronous PROFIBUS/PROFINET communication set in the factory, one DCC chart with 50 blocks can be calculated on each of the three drive objects in an execution group with a sampling time of 2 ms.

The following (functional single) blocks have been used in a chart in this case:

AND, OR, XOR, NOT, BF, CTR, MUX8, NAND, NCM, SH_DW, ADD, SUB, MUL, DIV, AVA, MAS, MIS, PLI20, DEL, DEZ, SUB_D, SUB_I, MUL_D, MUL_I, DIV_D, DIV_I, AVA_D, SII, STM, DLB, DX8, DX8D, MFP, PCL, PDE, PDF, PST, RSR, DFR, BSW, NSW, LIM, PT1, INT, DIF, LVM, LIM_D, PC, PIC, RGE

In a sampling time of 1 ms, only around half the blocks can be calculated. Around twice as many can be calculated in a 4 ms sampling time.

A SERVO axis with standard sampling times omitted from the maximum configuration thus permits the use of approximately 50 blocks (of the above-mentioned types) for a sampling period of 2 ms.
Approximately 50 blocks (of the above-mentioned types) in an execution group increase the computing time load of the CU by approximately 11%.

The following applies as a rough configuration aid to VECTOR drives:

If one A-INF infeed is configured on a SINAMICS CU320 with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), two VECTOR drive axes are configured with the speed controller sampling times p0115[1] = 2000 µs and the current controller sampling times p0115[0] = 500 µs (p0115[2] = p0115[3] = p0115[4] = 2000 µs) and the function modules and without isochronous PROFIBUS/PROFINET communication set in the factory, one DCC chart with approx. 70 blocks (type selection, see above for SERVO) can be calculated on two drive objects in an execution group with a sampling time of 2 ms.
This means approx. 70 blocks (type selection, see above for SERVO) increase the computing time load by approx. 15%.

In a sampling time of 1 ms, only around half the blocks can be calculated. Around twice as many can be calculated in a 4 ms sampling time.

The following generally applies:

You can of course use any blocks you want in your project. Using other block types may produce other results in terms of the number of blocks that can be calculated and the computing time load. Blocks with a very extensive functionality of course require a greater computing time. Note that the computing time needed for some blocks also depends on the values configured at the connections, e.g. whether an enable connection is set.

When measuring the computing time load, all connections should therefore be set to their final configuration values wherever possible.

When commissioning a configuration, the current computing time load must always be checked online on the drive unit for compliance with the limits stated at the start for r9976[1] and r9976[5].

Computing time load as of software version V4.3

With software version SINAMICS V4.3, DCC is available for the new higher performance CU320-2 DP module. It is also available for the new modules CU310-2 DP, CU310-2 PN, and CU320-2 PN in the case of SINAMICS V4.4, and for SIMOTION D4x5-2. As of software V4.3, after a download (DL) or a parameter change (e.g. of the sampling time of a execution group), the control unit (CU) uses the configuration data to determine the expected computing time utilization (including the load due to DCCs) and indicates this for the complete system in parameter r9976 (the level of system utilization that can be found on the drive object of the CU). If the calculated average computing time utilization for the complete system r9976[1] or the maximum utilization in a sampling time r9976[5] (including interruptions due to shorter sampling times) exceeds the maximum value of 100.00%, this causes the setting of fault F1054 (CU: System limit exceeded). The fault F1054 causes all infeeds and drive axes calculated on the CU to be switched off. The utilization is calculated on the CU, i.e. the utilization values are displayed in the STARTER/SCOUT only in online mode.

The resulting computing time utilization caused by DCC depends on the following:

- Number of calculated execution groups
- Sampling time of the execution groups
- Number of calculated blocks
- Calculated block types
- Use of the execution groups "Receive AFTER IF1 PROFIdrive PZD ", "Send BEFORE IF1 PROFIdrive PZD", "Receive AFTER IF2 PROFIdrive PZD", and "Send BEFORE IF2 PROFIdrive PZD" of the bus configuration (isochronous/ not isochronous; see chapter Fixed execution groups (Page 5529))

The proportional computing time utilization caused by DCC is displayed on the drive objects on which DCCs are configured in the parameter r21005[0...9] for the execution groups 1 to 10 (as of SINAMICS V4.3). Note that the average computing time utilization for an execution group k can be calculated only when this is registered for cyclical processing (p21000[k-1] ≠ 0, in the STARTER/SCOUT in the context menu of the chart → Set sampling times).

In contrast to software versions V2.5 and V2.6, a parameter change (in online mode of the STARTER) that affects the computing time utilization (e.g. change of the sampling time of a DCC execution group) causes an immediate recalculation of r9976 (and r21005) by the drive device. For parameters that may only be changed in the device states C1 (commissioning 1) or C2 , i.e. may only be changed in STARTER/SCOUT offline mode (e.g. p0115), the new value of the computing time utilization in r9976 will be displayed only after the project download and the subsequent CU ramp-up.

As of V4.3, the permitted maximum value of the computing time utilization r9976 increases practically to 99.99%. (For values > 100.00%, fault F1054 will be set; this simultaneously causes an OFF2 on all drive objects of the CU.)

As rough configuration aid, the following is true for **SERVO** drives with the current controller sampling time p0115[0] = 125 µs and the speed controller sampling time p0115[1] = 125 µs:

1. If on a SINAMICS CU320-2 one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **six SERVO drive axes** with the speed and current controller sampling times set in the factory (p0112 = 3 == STANDARD → p0115[1] = p0115[0] =125 µs), the function modules set in the factory setting, one TB30 [2)] and with isochronous PROFIBUS/PROFINET communication with $T_{\_DP}$ >= 2 ms are configured, one (1) DCC with approximately 50 blocks can be calculated in an execution group with a sampling time of 2 ms.
2. If on a SINAMICS CU320-2 one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **5 SERVO drive axes** with the speed and current controller sampling times set in the factory (p0112 = 3 == STANDARD → p0115[1] = p0115[0] =125 µs), the function modules set in the factory setting, 1 TB30 [2)], 3 TM31 [2)] and isochronous PROFIBUS/PROFINET communication with T_{DP} >= 2 ms are configured, one (1) DCC chart with approximately 75 blocks can be calculated [1)] in an execution group with a sampling time of 2 ms.
3. For each additional omitted SERVO axis, approximately 75 additional blocks [1)] can be calculated in the sampling time of 2 ms.

As rough configuration aid, the following is true for **VECTOR** drives with the current controller sampling time p0115[0] = 500 µs and the speed controller sampling time p0115[1] = 2000 µs (p0115[3] = 2000 µs, p0115[4]=2000 µs):

For more than three VECTOR axes on a CU320-2, the current controller sampling time will be increased automatically to the value p0115[0] = 500 µs. This automatically increases the speed controller sampling time to 2 ms.

1. If, on a SINAMICS CU320-2, one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **six VECTOR drive axes** with the current controller sampling times p0115[0] = 500 µs, the speed controller sampling time p0115[1] = 2000 µs, the functions and function modules set with the factory setting, one TB30 [2)] and with isochronous PROFIBUS/PROFINET communication with $T_{\_DP}$ >= 2 ms are configured, one (1) DCC with approximately 50 blocks can also be calculated in an execution group with the sampling time 2 ms.
2. If, on a SINAMICS CU320-2, one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **five VECTOR drive axes** with the current controller sampling times p0115[0] = 500 µs, the speed controller sampling time p0115[1] = 2000 µs, the functions and function modules set with the factory setting, with isochronous PROFIBUS/PROFINET communication with $T_{\_DP}$ >= 2 ms, one TB30 [2)] and three TM31 [2)] are configured, one (1) DCC with approximately 75 blocks can also be calculated in an execution group with the sampling time 2 ms.
3. For each additional omitted VECTOR axis (with p0115[0] = 500 µs and p0115[1] = 2000 µs), approximately 75 additional blocks [1)] can be calculated in the sampling time of 2 ms.

As rough configuration aid, the following is true for **VECTOR** drives with the current controller sampling time p0115[0] = 250 µs and the speed controller sampling time p0115[1] = 1000 µs (p0115[3] = 1000 µs, p0115[4] = 1000 µs):

For up to three VECTOR axes on a CU320-2, the current controller sampling time and the speed controller sampling time have the values p0115[0] = 250 µs and 1 ms, respectively, in the factory setting.

1. If, on a SINAMICS CU320-2, one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **three VECTOR drive axes** with the current controller sampling times p0115[0] = 250 µs, the speed controller sampling time p0115[1] = 1000 µs, the functions and function modules set with the factory setting, one TB30 [2)] and with isochronous PROFIBUS/PROFINET communication with $T_{\_DP}$ >= 2 ms are configured, one (1) DCC with approximately 50 blocks can also be calculated in an execution group with the sampling time 2 ms.
2. If, on a SINAMICS CU320-2, one A-INF infeed with the sampling times set in the factory (p0112 = 2 == LOW → p115[0] = 250 µs), **two VECTOR drive axes** with the current controller sampling times p0115[0] = 250 µs, the speed controller sampling time p0115[1] = 1000 µs, the functions and function modules set with the factory setting, with isochronous PROFIBUS/PROFINET communication with $T_{\_DP}$ >= 2 ms, one TB30 [2)] and three TM31 [2)] are configured, one (1) DCC with approximately 150 blocks can also be calculated in an execution group with the sampling time 2 ms.
3. For each additional omitted VECTOR axis (with p0115[0] = 250 µs and p0115[1] = 1000 µs), approximately 150 additional blocks [1)] can be calculated in the sampling time of 2 ms.

The following generally applies:

You can of course use any blocks you want in your project. The use of other block types may produce different results in terms of the number of blocks that can be calculated and the computing time utilization. Blocks with a very extensive functionality of course require a greater computing time.

1) The chart with 75 blocks consists of an execution group with the following blocks in the listed sequence:

AND, OR, XOR, NOT, BF, CTR, MUX8, NAND, NCM, SH_DW, ADD, SUB, MUL, DIV, AVA, MAS, MIS, PLI20, DEL, DEZ, SUB_D, SUB_I, MUL_D, MUL_I, DIV_D, DIV_I, AVA_D, SII, STM, DLB, DX8, DX8_D, MFP, PCL, PDE, PDF, PST, RSR, DFR, BSW, NSW, LIM, PT1, INT, DIF, LVM, LIM_D, PC, PIC, RGE, AND, OR, XOR, NOT, BF, CTR, MUX8, NAND, NCM, SH_DW, ADD, SUB, MUL, DIV,AVA, MAS, MIS, PLI20, DEL, DEZ, SUB_D, SUB_I, MUL_D, MUL_I, DIV_D.

The chart with 150 blocks consists of an execution group that contains the previously listed 75 blocks twice.

2) All sampling times (p4099[]) of the TB30 and the TM31 have the value 4 ms (factory setting).

Memory requirement

The blocks and @ parameters in the DCC charts require memory on the drive unit. To use DCC-SINAMICS, for SINAMICS G130, G150, S120, S120 Chassis, S120CM, S150 and SIMOTION D4x5, at least one drive axis must be excluded in comparison to the maximum possible configurations (1 ALM + 6 servo axes + TB30 or 1 ALM + 4 vector axes + TB30).

Maximum quantity structures with a topology of 5 servo or 3 vector axes with 1 ALM and 1 TB30			
Drive unit		**CU320**[2]	**SINAMICS Integrated on SIMOTION D4x5**
SINAMICS 2.6.1	**Blocks**[1]	350	-
	@Parameters[1]	350	-
SINAMICS 2.6.2	**Blocks**[1]	500	200
	@Parameters[1]	500	200

2) SINAMICS G130, G150, S120, S120 Chassis, S120CM, S150

Note

The specified maximum numbers for blocks and @parameters always apply for the entire drive unit and should be regarded as guide values. Only the factory-set function modules are active. The individual blocks and @parameters may be arbitrarily distributed across several charts. Conserving @parameters has little effect on the quantity structure of the blocks; the specified maximum numbers for the blocks should therefore not be exceeded.

Savings in further drive axes can increase the maximum limits by 50 blocks and 50 @parameters for each axis saved.

Maximum limits when just one axis is used (including 1 ALM + TB30)					
Drive unit		**CU320 and CU310[2] with 1 servo axis**	**CU320 and CU310[2] with 1 vector axis**	**SINAMICS Integrated[3] with one servo axis**	**SINAMICS Integrated[3] with one vector axis**
SINAMICS 2.6.1	**Blocks**[1]	550	450	-	-
	@Parameters[1]	550	450	-	-
SINAMICS 2.6.2	**Blocks**[1]	700	600	400	300
	@Parameters[1]	700	600	400	300

2) SINAMICS G130, G150, S120, S120 Chassis, S120CM, S150

3) S120i on D4x5

When expanding the topology on a drive unit with TMxx or DMC20 modules, one drive unit must be omitted for each additional TMxx module.
Please note when connecting CX32 modules to a D4x5 that memory and CPU time are also allocated for each CX32 on the D4x5. When using the DCC, another servo or vector axis must be omitted for every 2 CX32s (i.e. the axis must be omitted for the first CX32).

Example for SINAMICS 2.6.2:

D4x5 with 1 ALM + 4 servo axes + 2 CX32 + 1TB30 à
Permits DCC with 200 DCBs + 200 @parameters.

D4x5 with 1 ALM + 2 vector axes + 2 CX32 + 1TB30 à
Permits DCC with 200 DCBs + 200 @parameters.

When using DCC on the CX32 the same quantity structure applies as on the CU320.

Note

The final limits are determined by the total memory available on the drive unit and the computing time load. If the above-mentioned recommended maximum limits are exceeded, this can result in errors during upload or download (e.g. fault F1105 CU: insufficient memory) and the drive can no longer be switched on. If it is not possible to perform a new download with a suitably adjusted project, a power OFF/ON must be performed on the affected drive unit.

By activating additional function modules (e.g. basic positioner EPOS), additional memory is allocated and the specified maximum limits are decreased.

[1]) The charts for the above-specified quantity structures are formed from n x (chart1 + chart2): Chart1 contains 50 blocks (of the following types: AND, OR, XOR, NOT, BF, CTR, MUX8, NAND, NCM, SH_DW, ADD, SUB, MUL, DIV, AVA, MAS, MIS, PLI20, DEL, DEZ, SUB_D,SUB_I, MUL_D, MUL_I, DIV_D, DIV_I, AVA_D, SII, STM, DLB, DX8, DX8D, MFP, PCL, PDE, PDF, PST, RSR, DFR, BSW, NSW,LIM, PT1, INT, DIF, LVM, LIM_D, PC, PIC, RGE) with 50 @parameters and approx. 90 connections (from block connector to block connector).

Chart2 contains 50 blocks (of block types AND, OR, ADD, MUL, DIV, B_DW, B_W, BY_W, D_I, D_R, D_UI, D_US, DW_B, DW_R, DW_W, I_D, I_R, I_UD, I_US, N2_R,N4_R, R_D, R_DW, R_I, R_N2, R_N4, R_UD, R_UI, R_US, UD_I, UD_R, UI_R, US_D, US_I, US_R, W_B, W_BY, W_DW, WBG,DCA, INCO, OCA, TTCU, ADD, ADD_D, ADD_I, ADD_M, AVA, AVA_D, RSS), 50 @parameters and approx. 90 connections (from block connector to block connector)

For example, 350 blocks and 350 @parameters are compiled from 4 * chart1 + 3 * chart2.

Memory requirement as of software version SINAMICS V4.3

The blocks and @ parameters in the DCC charts require memory on the drive unit. A maximum of 1,500 blocks and 1,500 @ parameters may be configured in DCC SINAMICS with the modules CU320-2 DP, CU320-2 PN (V4.4 or higher), CU310-2 DP (V.4.4 or higher), and CU310-2 PN (V.4.4 or higher) on SINAMICS S120, S150, G130, and G150.

For SINAMICS_Integrated V4.4 or higher (V4.2 or higher with SIMOTION), a maximum of 1,500 blocks and 1,500 @ parameters may be configured on the D4x5-2 modules.

Number of possible different hardware sampling times

The sampling times for the execution groups can be selected in p21000[x] as a multiple of r21002 (basic sampling time of hardware time slices), a multiple of r21003 (basic sampling time of software time slices), or on the basis of the sampling time of a basic SINAMICS system function (e.g. when p21000[x] = 9003 == "before setpoint channel" from the sampling time of the setpoint channel p0115[3]).

Only sampling times can be set as hardware sampling times, for which the following applies:

1 ms <= T_sampling <= r21003 - r21002 in p21000[x]

Hardware sampling times, assignment, and number

The assignment of the available hardware sampling times is displayed in r21008[0...24] as follows (STARTER/SCOUT: in online mode only):

- Value = 0.0 --> sampling time not assigned
- Value != 0.0 (not equal to 0.0) --> sampling time in ms
- Value = 9.9999e + 006 --> sampling time not supported

During configuration, note that the number of different hardware sampling times (1 ms = cycle duration T_sampling <r21003 - r21002) used by the basic SINAMICS system, active function modules (see r108) and Drive Control Chart is restricted as follows:

- CU310, CU320, D4xx --> no. of hardware sampling times = 13
- CU320-2 DP with SINAMICS V4.3 and higher --> no. of hardware sampling times = 25
- CU310-2 DP, CU310-2 PN, and CU320-2 PN with SINAMICS V4.4 and higher --> no. of hardware sampling times = 25

- SINAMICS_Integrated on the D4x5-2 with SINAMICS V4.4 (contained in SIMOTION V4.2) and higher --> no. of hardware sampling times = 25
- CUD (SINAMICS DC MASTER) --> number of hardware sampling times = 11

Hardware sampling times, usage

A sampling time can be used simultaneously by more than one execution group of DCC, function modules and the basic SINAMICS system.

For this reason, the execution groups should ideally be registered to existing sampling times or, if it makes more sense in relation to the function, the execution group "Before setpoint channel" should be used.

For internal reasons, the drive unit always requires at least one (or more, depending on how the basic sampling times p0115[0] of the drive objects are parameterized) free hardware sampling time, which is why the current **Number of free hardware sampling times** can be read in **r7903** (as of SINAMICS V2.6).

DCC and FBLOCKS together must not use more than 5 different hardware sampling times on one drive unit. (This only refers to sampling times that deviate from the hardware sampling times already present in the basic system.)

Note

Note that a long-term trace registers a sampling time of 2 ms and the trace registers sampling times in accordance with the selected trace cycle clock. If these sampling times have not already been registered by the basic SINAMICS system, "Free blocks" (FBLOCKS), or Drive Control Chart (DCC), these functions require additional free hardware sampling times.

The registered hardware sampling times can be read (if the DCC is activated) in r21008[0...12].

The current number of free hardware sampling times is displayed in r7903 (as of SINAMICS V2.6).

Project download, error message, and procedure

If too many different hardware sampling times are configured offline, an error message is not output until the project is downloaded.

In this case, proceed as follows:

1. With the project in offline mode, set all the free execution groups to which hardware sampling times are assigned to software sampling times.
   - Hardware sampling times (DCC: p21000 < 256)
   - Software sampling times (DCC: p21000 >= 1001)
     The assignment of fixed execution groups (DCC: p21000 >= 2000) does not need to be changed because the fixed execution groups use the same sampling time as the assigned basic SINAMICS system function.
2. Download the project again.

3. Once the project has been downloaded and the Control Unit has ramped up, check:
   – r7903: Number of hardware sampling times still available
   – r21008: Hardware sampling times already registered by the basic SINAMICS system.
4. Adjust the execution group parameters accordingly.

Note

The number of different hardware sampling times possible on a Control Unit is restricted. For this reason, software sampling times (multiple of r21003) or, if necessary, the fixed execution group (p21000[0...9] ≥ 2000) should ideally be used.

Upgrading SINAMICS firmware

Upgrading firmware on the CF card

If you upgrade the SINAMICS firmware directly on the CF card and replace the USER directory, you must also replace the DCB libraries used in the DCC chart in the OEM directory.

For a detailed description of the procedure, please refer to the Function Manual*SINAMICS S120 Drive Functions*.

Updating the firmware via the Web server

For a detailed description of the procedure, please refer to the Function Manual*SINAMICS S120 Drive Functions*.

7.4.4.2 Working with DCC SINAMICS

Preliminary remarks on configuration

Preliminary remarks

The following is a brief explanation of what you will be configuring in this chart.

Configuration example

The requirement for this example is that there must be a STARTER or SCOUT version (≥ V4.1.2) suitable for your SINAMICS SW version (≥ V2.6) installed on your computer. The SINAMICS Support Package (SSP) V2.6 must also be installed. Your computer must also have a CFC license to use the DCC editor. You can install this license from the USB stick provided with the help of the Step7 Automation License Manager. A CU310 or CU320 is required with which STARTER/SCOUT can connect online, e.g. via PROFIBUS.

The configuration example deals with a straightforward oscillating circuit that creates sinusoidal oscillation at its output.

It will only take you a few minutes to create the chart; then you can execute it in test mode as a demonstration.

The following blocks are used:

- Two integrators (**INT**)
- One inverter (**SII**)

As indicated by the differential equation **f"(x) = - f(x)**, the oscillating circuit is comprised of two integrator blocks that are linked by negation.

The frequency of the oscillating circuit is determined by the integrating time constant at the integrators.

The oscillation amplitude is specified by the initial value at the integrator output.

Configuration example structure

The configuration example is divided into the following steps:

1. Creating a new project.
2. Inserting the DCC in the project.
3. Inserting blocks in a DCC.
4. Interconnecting blocks.
5. Parameterizing block connections in the chart.
6. Publishing block connections as parameters.
7. Setting execution sequence within an execution group
8. Compiling the DCC in the DCC editor.
9. Setting sampling time for an execution group
10. Loading the DCC technology option on to the drive unit's CF card.
11. Downloading the compiled DCC to the drive.
12. Displaying values at block connections online.
13. Recording signals from the DCC with the trace.
14. Archiving the project.
15. Creating documentation.

Creating a new project

- Create a new project in the SCOUT or STARTER engineering system, e.g. **dcc_ex**, see Creating a project (Page 5424).
- Create a new device. To do this, insert a "SINAMICS S120 CUxxx" in the STARTER or SCOUT by double-clicking the "Insert single drive unit" command. Select "xxx" to suit your existing hardware (CU320, CU310DP, CU310PN).
- You can now insert a chart.

Inserting a DCC

- Open up the tree structure in the project navigator to the control unit.
- Call **Insert DCC** below the control unit (double-click).

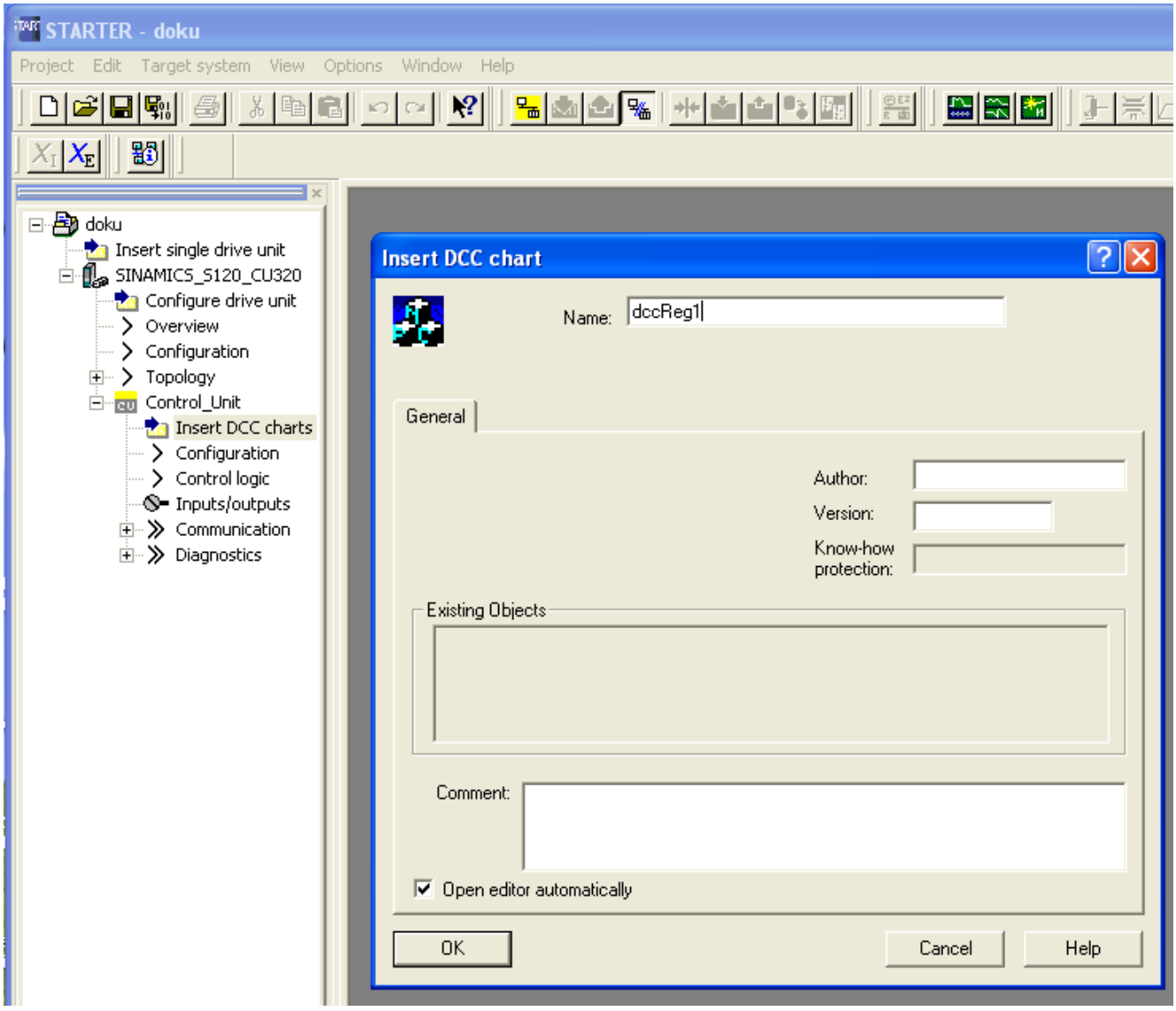


Figure 7-645 Inserting a DCC SINAMICS chart

- The **Insert DCC** window opens and you can enter a new name (of no more than 22 characters) for the chart (in our example *dccReg1*) and a comment. Please note that all characters after an underscore in the chart name can only be digits.
- Click OK to close the window. The DCC is opened when the **Open editor automatically** checkbox is activated
  Alternatively, you can open the chart at any time by double-clicking on the chart symbol in the project navigator.

- When first creating a chart in a project, you will be prompted to import a block library.

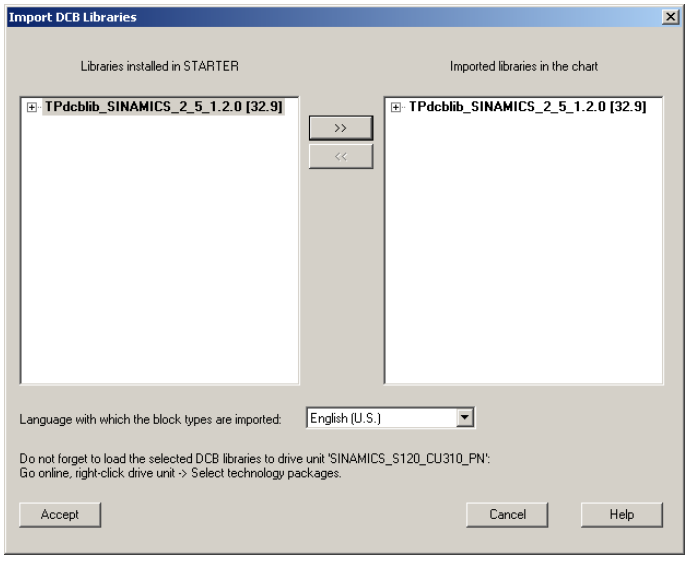


Figure 7-646 Importing a block library

- In the window **Import DCB Libraries**, select the block library in the left column **Libraries installed in STARTER.** Take the selected library into the right column by clicking the **>>** button.
- Close the window with the **Accept** command. The block library is loaded and the DCC editor opened with the chart.

The project structure has now been set up, a chart created and the block library loaded. All that remains is to create some activity within the chart, i.e. by inserting blocks and interconnecting them. Just one chart can be created for each drive object.

Inserting blocks

- Open a block family in the **Closed-loop control** family with the closed-loop control blocks.

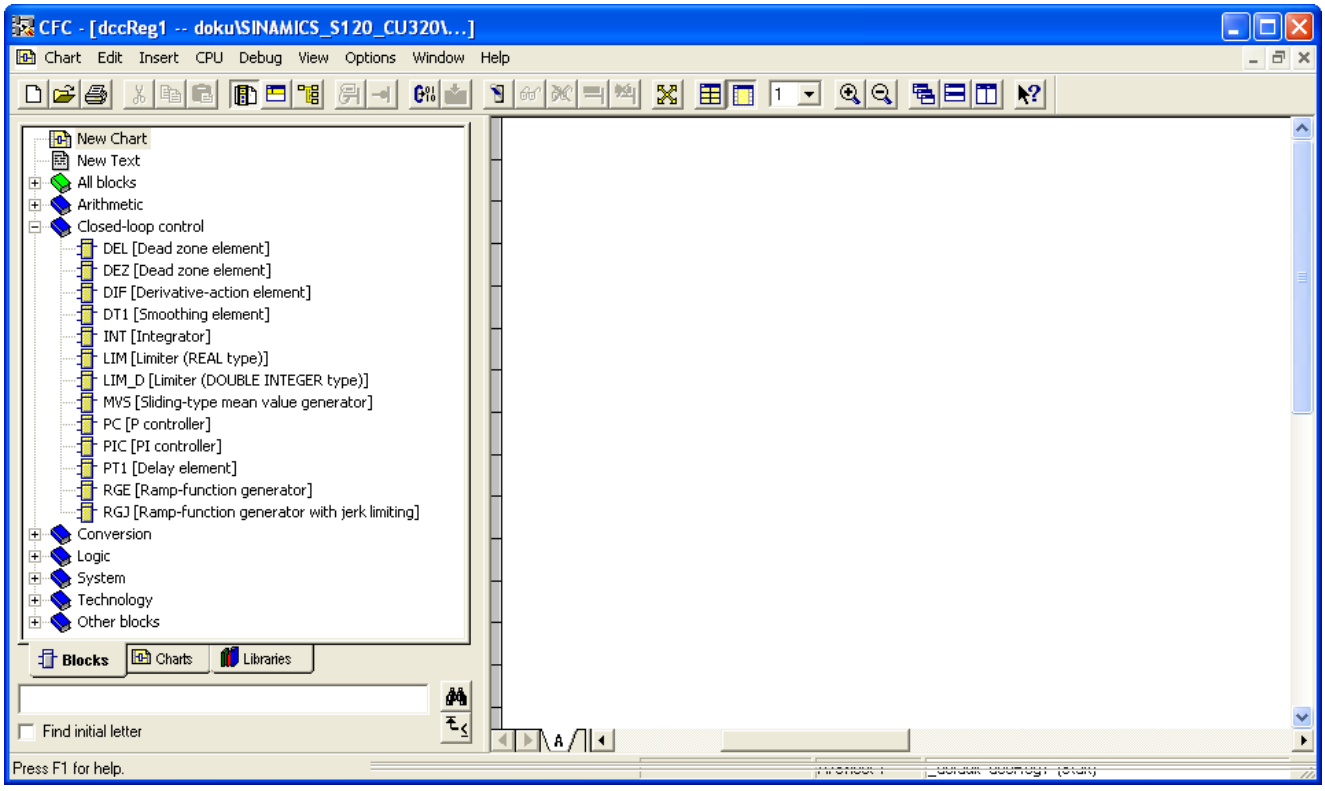


Figure 7-647 DCC editor with opened **Closed-Loop Control** DCB family

- Select the required block (e.g. INT) and insert it in the chart using drag-and-drop. Only the outline of the block in dashed lines is displayed during the copying procedure. Release the mouse button at the required point.

Note

If one block superimposes another block in the chart, the superimposed block will be displayed in gray and its connections will not be visible. You must reposition the blocks to ensure that all connections and block information are displayed.

Interconnecting blocks

Procedure

- Select the **Y** output of the first **INT**egrator, followed by the **X** input of the second **INT**egrator.
- Select the **Y** output of the second **INT**egrator, followed by the **X** input of the inverter (**SII**).
- Select the **Y** output of the inverter (**SII**), followed by the **X** input of the first **INT**egrator.

The autorouter creates the connecting lines from the outputs to the inputs and they are then interconnected.

Parameterizing block connections in the chart

For the first integrator INT 1/1, the following initial values should be assigned to the connections: LL = -10.0, LU = 10.0, SV = 2.0, Ti = 100 ms

For the second integrator INT 1/2, the following initial values should be assigned to the connections: LL = -10.0, LU = 10.0, Ti = 100 ms

To do this, open the **Properties - Connection** window of the respective block connection by double-clicking it. Enter the initial value for **Value** and click **OK** to close the window. Note that for the input of the values for Ti, the "ms" unit must follow the numeric value 100 without any spaces.

As the connections mentioned above are not interconnected, the entered values also remain valid after the initialization. The initialization of the block inputs and outputs is performed before the first cyclic call of the chart.

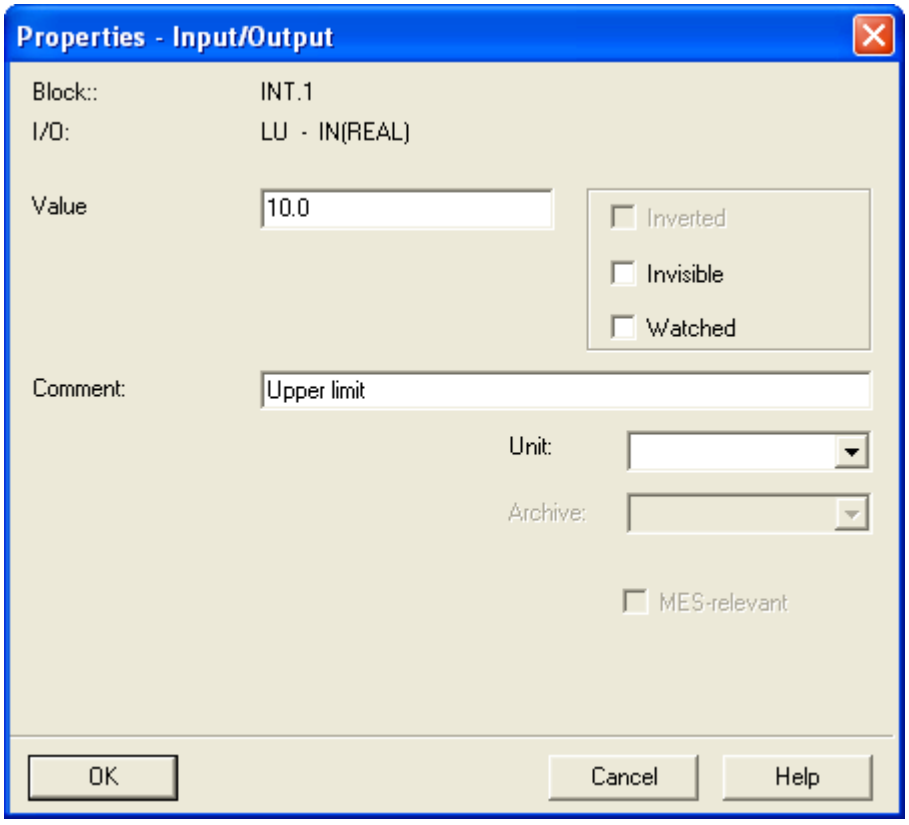


Figure 7-648 Properties window of the "LU" block connection

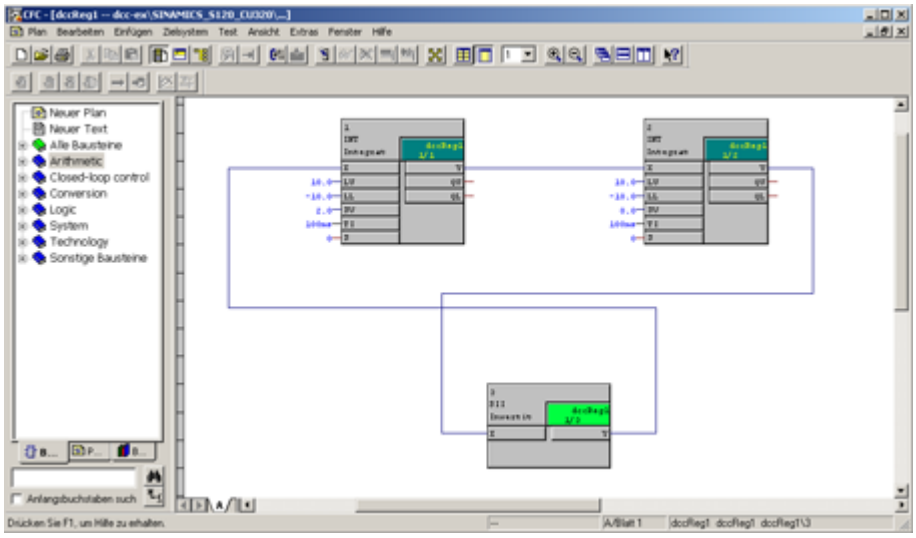


Figure 7-649 Chart "dccReg1" with interconnected blocks. Some connections have been assigned initial values that differ from the factory setting.

Publishing block connections as parameters

To be able to interconnect the output signal of the second (right-hand) integrator to the SINAMICS basic system, it must be published as an interconnectable parameters (i.e. as BICO parameter). To excite the oscillating circuit once, the integrator input of the first (left-hand) integrator must be published as adjustable parameter. The parameter numbers used in the following section have been chosen arbitrarily.

The Y connection of the second integrator is to be published as adjustable parameter. To do this, enter "@*20 output", for example, in the comment field in the Properties window (double-click connection Y). See Creating customer-specific parameters (Page 5543).

Enter "@1 start" in the comment field of connection S (set) of the first integrator. This publishes the block connection as adjustable parameter.

As the default setting of the parameter number base of a chart is always 0, when the chart is compiled, the new parameters r21501 and r21520 are created and displayed in the expert list of the CU drive object in STARTER.

Note

To make this assignment of the @ parameters to the connections in the chart visible, the display form of the connections in the DCC editor must be changed. To do this, open the **Display Settings** window in **Options > Settings > Display**, change there under **Connections** the display from **Name** to **Comment** and click **OK** to close the window.

The chart will then be compiled.

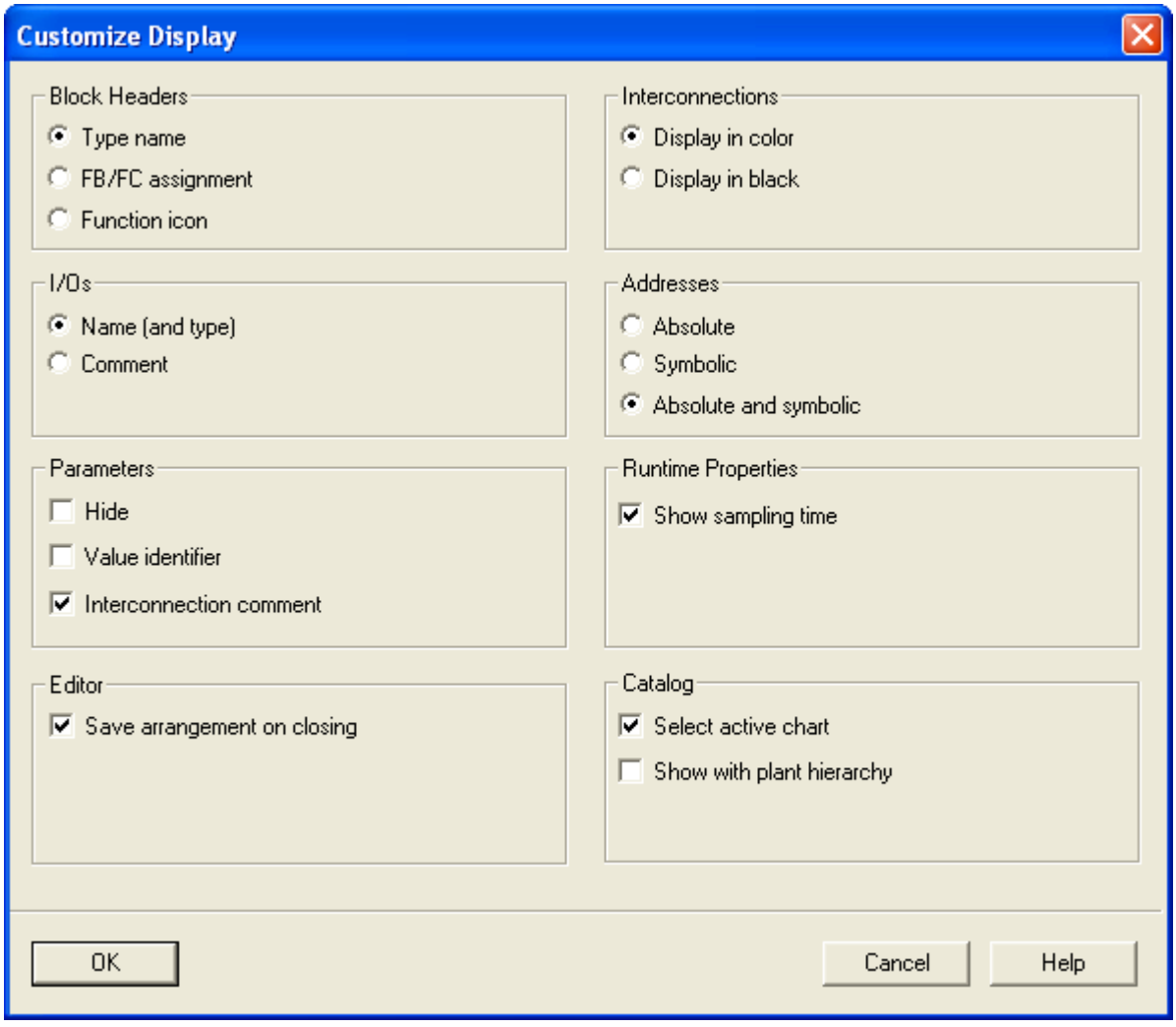


Figure 7-650 **Display Settings** window

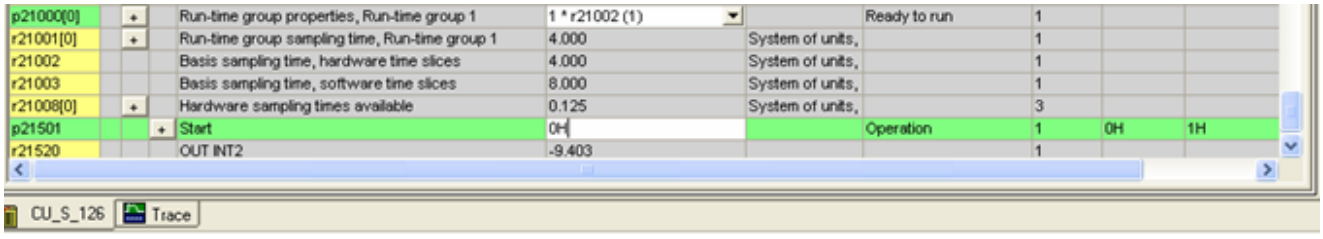


Figure 7-651 View from the expert list with the parameters defined in the DCC

Compiling the DCC chart in the DCC editor

Compiling

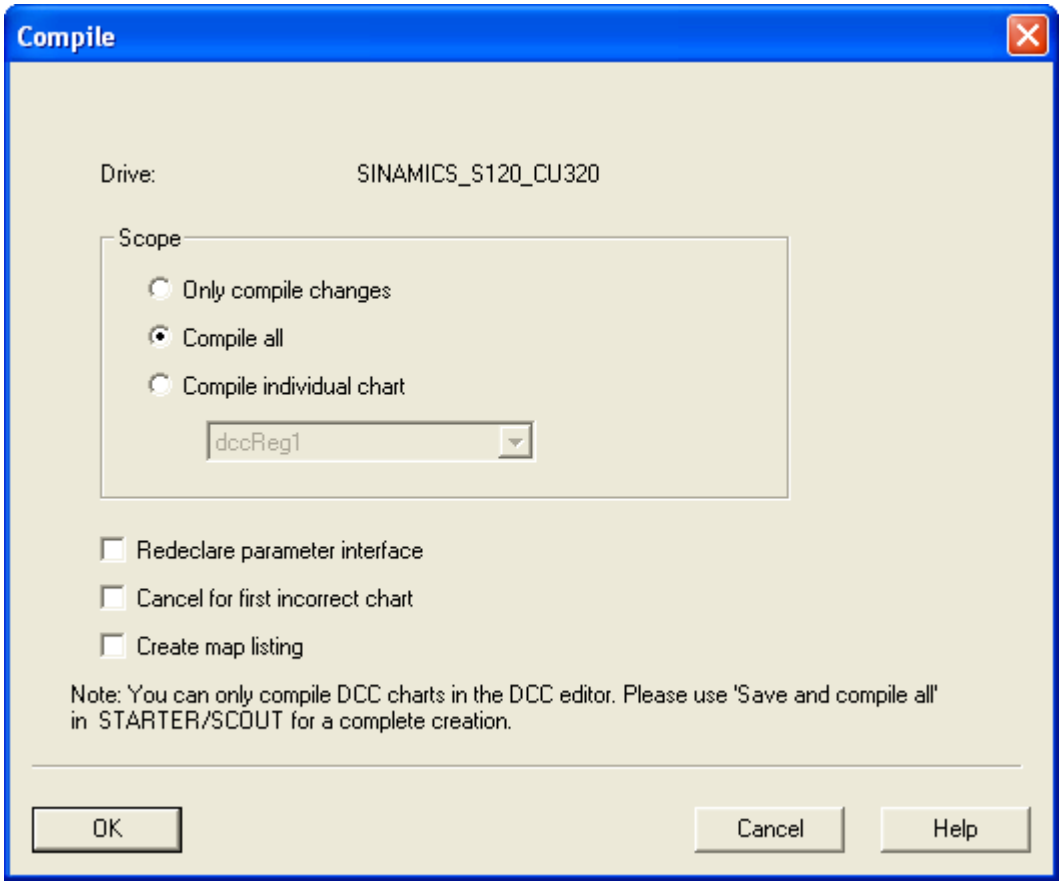


Figure 7-652 Compiling a dialog in the DCC editor

Before the first compilation from the DCC editor, the project must be saved once in STARTER (button).

You can begin compiling from the DCC editor with **Chart > Compile > Charts as program...** or via the button.

You can also start the compiling and saving of a project from the STARTER by clicking the button.

If errors occur during compilation, the **Logs** dialog box will automatically be displayed at the end of the procedure (just as in the case of the consistency check).

Compilation options

For detailed information about the compilation options, refer to Compiling (Page 5450).

After compilation

The compilation log is displayed after compilation. If error messages are displayed, the causes must be corrected before continuing.

Note

Once the DCC chart has been compiled, the interconnections from the DCC chart apply. For detailed information on accepting the interconnections from the expert list into the DCC chart, please refer to the section titled Reading back BICO interconnections and parameters (Page 5572).

Setting execution sequence within an execution group

These are automatically assigned to execution group 1 when the blocks are inserted in a new chart. Execution group 1 is automatically given the name of the chart. The sequence corresponds to the order in which they are inserted in the chart. The sequence can be displayed and edited in the execution editor. You can access the execution editor in the DCC editor via **Edit > Execution sequence** or with  . If an execution group (light blue folder) is selected in the center column in the tree structure, the blocks contained in the execution group are displayed in the right-hand column.

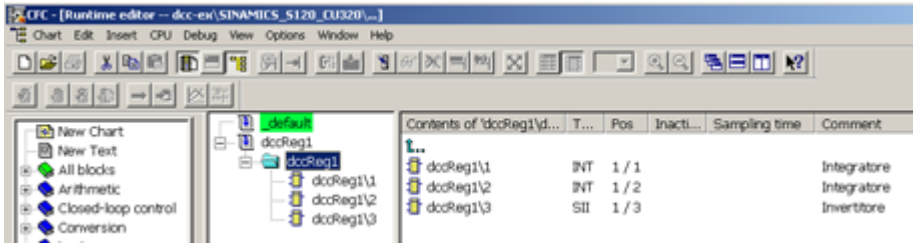


Figure 7-653 Execution editor with open execution group dccReg1

You can return to the DCC by selecting **Edit > Execution sequence** again or by clicking  again.

Setting sampling time for an execution group

The sampling time for an execution group is always set in the STARTER. The DCC you want is highlighted in the project navigator and the **Set execution groups** item called from the chart's context menu. The **Set Execution Groups** window opens.

Our chart was created on the CU drive object. r21002 = 4 ms in this case. A free execution group with a sampling time of 1*r21002 is set.

Note

In the offline mode of the STARTER / SCOUT V4.1.x, r21002 and r21003 are always displayed with the value. The r21002 value is identical with the p0115[0] value on the associated drive object.

When setting the sampling time, note that the minimum sampling time in the drive for DCC execution groups is 1 ms. Only times that are shorter than r21003 may be chosen as a multiple of r21002.

Please re-compile the chart once you have set the sampling time.

Loading the DCC technology option onto the CF card of the drive device

As of SINAMICS 4.3.1, the DCC technology package is located on the S120/S150 CompactFlash Card supplied from the factory for stand-alone drive units. With all other SINAMICS and SIMOTION D4xx CompactFlash Cards, the DCC technology package must be downloaded to the card by means of a technology package download

To load the technology package, call up SCOUT/STARTER and establish an online connection with the drive unit. SCOUT/STARTER is in online mode.

Then in the drive unit's context menu, select the **Select technology packages...** command.

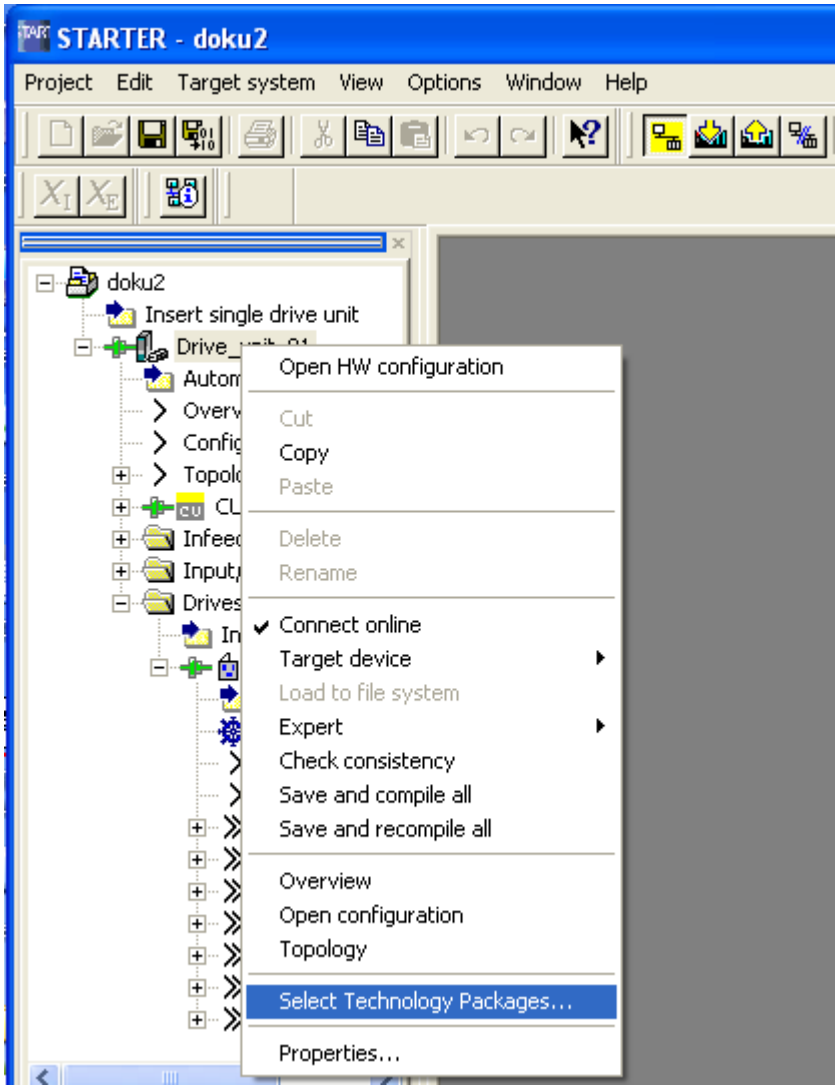


Figure 7-654 Context menu of the drive unit

In the **Select Technology Packages** window, select the desired block library and set **Download** for this library in the action column. Then click the **Perform actions** button in the bottom-right window above the progress bar. The label on the button changes to **Cancel**.

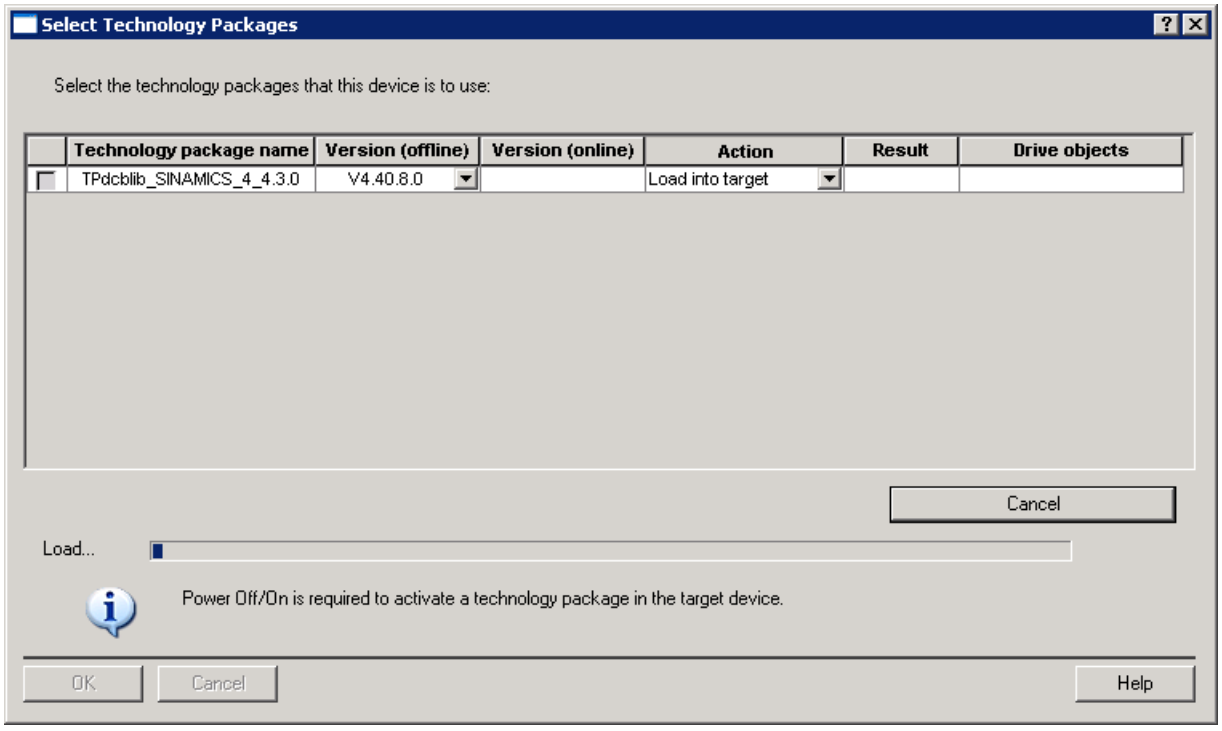


Figure 7-655 Select Technology Packages window

A progress bar shows how the downloading process is progressing. Once the downloading process is complete, the label on the button changes back to **Perform actions**.

The window is closed by double-clicking **OK**.

Note

Once the technology package has been downloaded, the drive unit has to be switched off and on again once. Only once the supply voltage to the CU and/or D4xx has been switched back on and it has been ramped up can DCCs be downloaded to the drive unit and run.

As of SINAMICS 4.4 and with the CU310-2 and CU320-2 modules, it is **not** necessary to switch off and restart the drive unit once the technology package has been downloaded. Once the technology package has been downloaded, an internal ramp-up process is performed automatically. DCC is loaded as part of this process.

When technology packages are deleted from the CompactFlash Card, all the DCC components of all the projects saved on it are deleted at the same time. As of SINAMICS V4.4 and with the CU310-2 and CU320-2 modules, an internal ramp-up process is then performed by the CU. This removes DCC from the CU memory.

Note

When downloading the technology package, as few people as possible should be connected to the bus, as the download time is significantly increased when multiple users are connected.

Downloading compiled DCC chart into the drive

To be able to execute the DCC program on a drive object, it must have first been downloaded to the drive unit. To do this, establish a connection to the drive unit and click the **Download** button in the Online/Offline Comparison window.

Downloading can take place in STARTER at any time via the **Download** function.

After the successful download, the DCC is calculated in cyclic operation on the CU drive object.

Displaying values of block connections online

As of SINAMICS V2.6 and SCOUT/STARTER V4.1.2, the DCC editor can display the values of block inputs and outputs online in test mode. The display is independent of whether a block input or output has been published as parameter. SCOUT/STARTER must be in online mode.

The signal value of block inputs (= connector inputs) published as BICO parameters cannot be displayed. Instead, the signal value of the supplying output can be displayed.

As soon as the test mode is exited again (e.g. by clicking ), the yellow fields with the value display disappear.

If you want to display the values of the connections logged on once for display, by selecting the test mode again, you must set **Test > Laboratory mode** in the DCC editor. This setting is only possible when the DCC editor is not in test mode.

In our example, the output values Y of the two integrators are to be displayed. To do this, open successively in the DCC editor with a double-click on the connection of the **Properties – Connection** window and with a mouse click set a tick for **For test** on the right-hand side in the middle. This registers these outputs for display in test mode. Then use **Test > Test mode** or Ctrl-T or  in the DCC editor to switch to test mode. The values of the two block outputs are highlighted in yellow and refreshed with a monitoring cycle of 2 seconds.

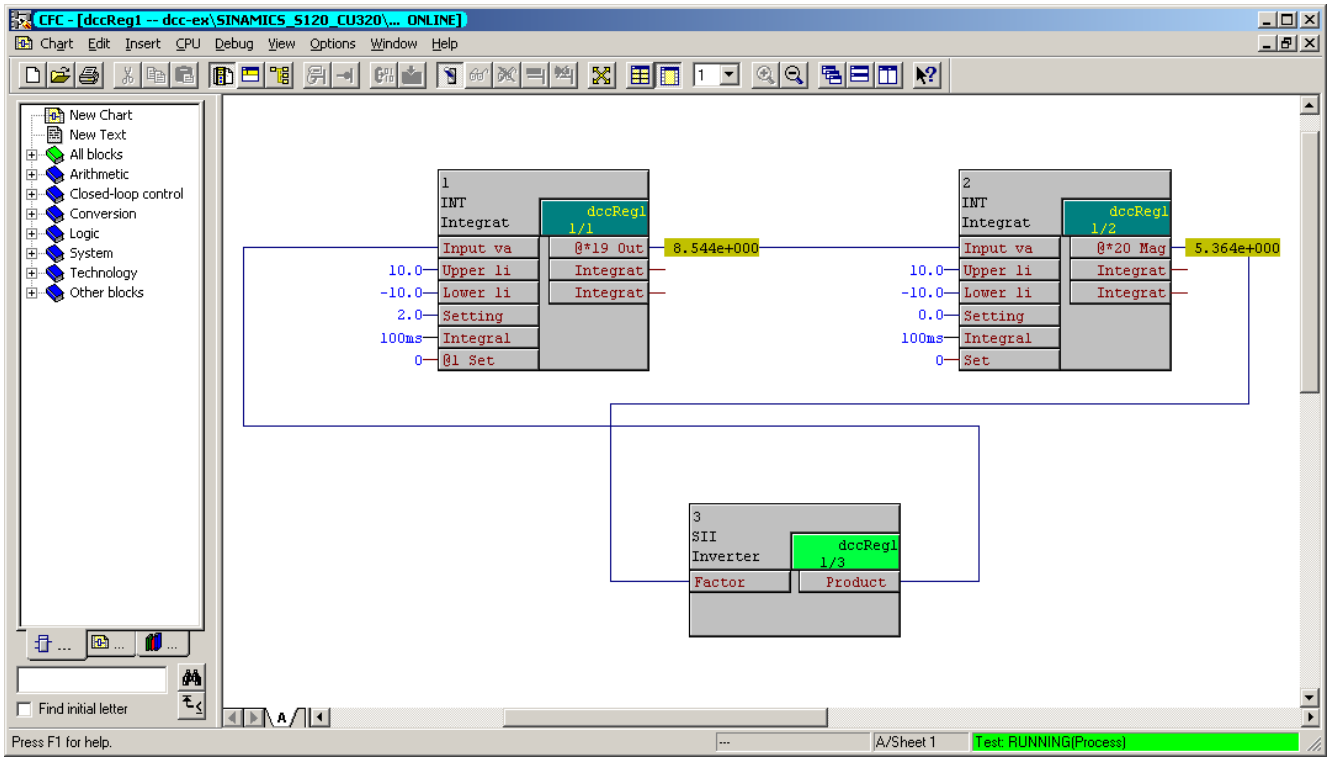


Figure 7-656 Display of the DCC dccReg1 in test mode

To make the oscillator vibrate, it must be excited once. To do this, set the p21501 parameter in the expert list of the STARTER once to "1" (the set value SV = 2.0 is applied at the output of the left-hand integrator) and then reset to "0". The oscillator vibrates and the output values of the integrators show changing values in the chart and in the expert list (r21520).

If you want to make further changes to the chart, you can exit the test mode with **Test > Test mode** or by clicking the button. It can take several seconds before the test mode of the DCC editor is exited.

If you want to display the values of further block connections in test mode, right-click the desired block connection and select **Log on connection** in the context menu.

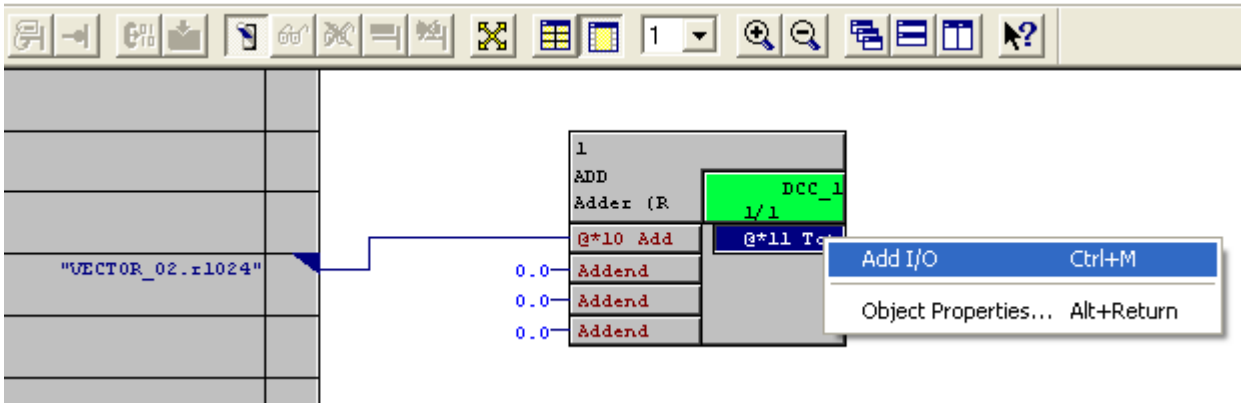


Figure 7-657 Log on connection context menu

If you want to display the signal value of a BICO output of the SINAMICS basic system, the signal value supplied to the DCC via the sheet bar cannot be displayed directly. In this case, an NOP_xx-

block with the correct data type is inserted. The signal value of the output of the NOP_xx- can then be displayed online.

Interconnection to the BICO parameters of the basic system in DCC SINAMICS

Procedure

You can make an interconnection to the BICO parameters of the basic system as follows:

1. Open the DCC chart.
2. Select the DCB connection to be interconnected.
3. Publish the selected connection as a BICO parameter (the comment for the connection must start with "@*", see Creating customer-specific parameters ("publishing") (Page 5543))
4. Right-click and select **Interconnection with operand...** from the context menu. The DCC Signal Selection window appears.
5. Select the parameter to be interconnected. Only those parameters are offered that are compatible according to the BICO interconnection rules. Further information on BICO interconnections can be found in the online help of STARTER/SCOUT under the BICO interconnections index entry.
6. Click OK to close the window.

The block connection is interconnected to the selected BICO parameter.

Note

The interconnected block connections must be published as BICO parameters for connections to signal outputs or signal inputs of the basic system.

The connection can first be made in the DCC editor without publishing the block connections. However, connecting an unpublished block connection to a BICO parameter of the basic system results in an error when compiling the chart.

Note

Interconnections with published block connections, which are established via the expert list, are not read back to the DCC editor. For this reason, subsequent compilation of the DCC chart will result in interconnections created previously in the expert list being overwritten again.

You can prevent interconnections from being overwritten by selecting **Chart -> Read back BICOs and parameters** in the open DCC chart in offline mode and then compiling the DCC chart.

Note

If **symbolic assignment** is enabled in the project, the **Standard/automatic** setting must be changed to **User-defined** in the **Telegram configuration** dialog box for the drive objects in which DCC is used.

The automatic telegram setting and automatic telegram extension must be deactivated.

Interconnection of different data types

When interconnecting to BICO parameters, the data types of the BICO parameter and pin may differ. In this case, you must ensure that the values are not outside the valid value range of the data types. Generally, the LIM DCC block should be connected between the input/output and the BICO parameter for this purpose.

If, for example, a block output with the REAL data type is connected to a BICO parameter of the UNSIGNED32 type, an exception error can occur when the value at the block output exceeds the value value range of an UNSIGNED32.

Deleting and moving BICO interconnections

When moving BICO interconnections, the signal can acquire the value 0 for a few cycles. The no longer interconnected input on the original connection is permanently assigned the value 0.

BICO interconnections and reading back parameters

To display up-to-date BICO interconnections, which have been changed online in the target device, in the DCC editor in online mode, the project must be read back from the target device.

It is also necessary to read back BICO interconnections and parameters to the DCC chart if these have been changed in the expert list. Unless these are read back, the configuration from/in the DCC chart is reactivated after a subsequent download.

Note

Implicit calling of the function "Read back BICO interconnections and parameters"

If drive objects are deleted, the function "Read back BICO interconnections and parameters" will be automatically called.

Procedure

BICO interconnections and parameter values changed subsequently can be read back from the target system via the DCC editor menu command **Chart -> Read back BICOs and parameters**. As of DCC Version 2.1, you can find this under **Options -> Read back BICOs and parameters**.

Input and output BICO interconnections are read back. The read back process always applies to all DCC charts of a device.

Requirements

- The structure of the block instances and interconnections must be the same both online and offline.
- The sources of the DCC chart must be available.

Note

It is not possible to read back BICO interconnections in the case of DCC charts that have been uploaded from a target device to an empty project.

Record with the trace signals from the DCC chart

The integrator output published as interconnectable parameter acts like any other connector output of the SINAMICS basic unit. This means this signal can also be recorded with the trace. The signal characteristic at block connections (that are not published as parameters) can still be recorded with the trace. The trace is called in STARTER with **Target system > Trace** or . In this example, the outputs of the two integrators in the DCC are recorded with a cycle clock of 1 ms. The output of the first (left-hand) integrator is to be recorded as first signal in the trace. The signal selection is opened with for this. Select the drive object on which the DCC is located (only CU_S_126). Open the directory tree and the dccReg1 chart is displayed. Open the directory tree of dccReg1 and the three blocks in the DCC are displayed. The desired connection is on block 1 (top left digit in the block symbol) of the dccReg1 chart. The designation of the connections in the signal selection of the trace is made up of the _ChartName_BlockNumber_, i.e. _dccReg1_1 in this case. The desired connection can now be selected in the list of block connections. The window is then closed by clicking **OK**. See the following figure.

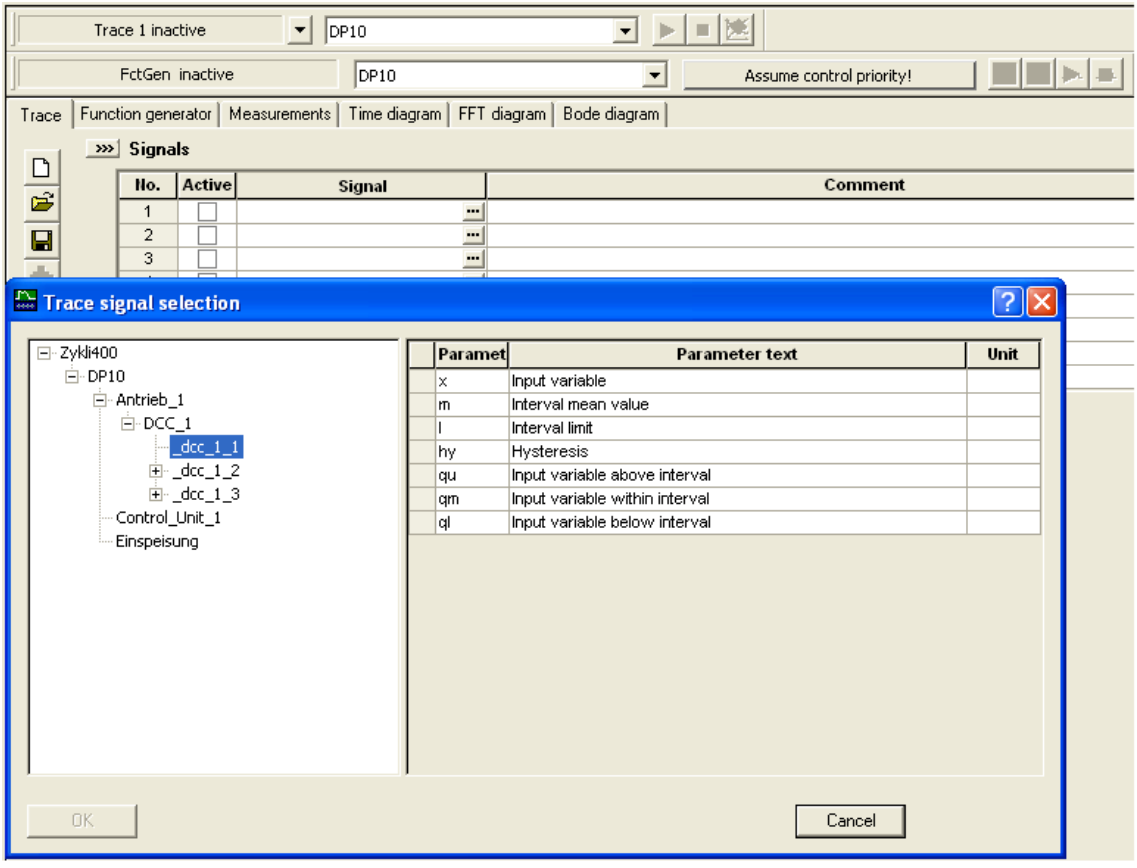


Figure 7-658 Trace signal selection for the block connections of the _dccreg1_1 block in the dccReg1 chart

The output of the second (right-hand) integrator is to be set as second signal. This block output is published as BICO parameter r21520. The recording can be started with the ▶ button.

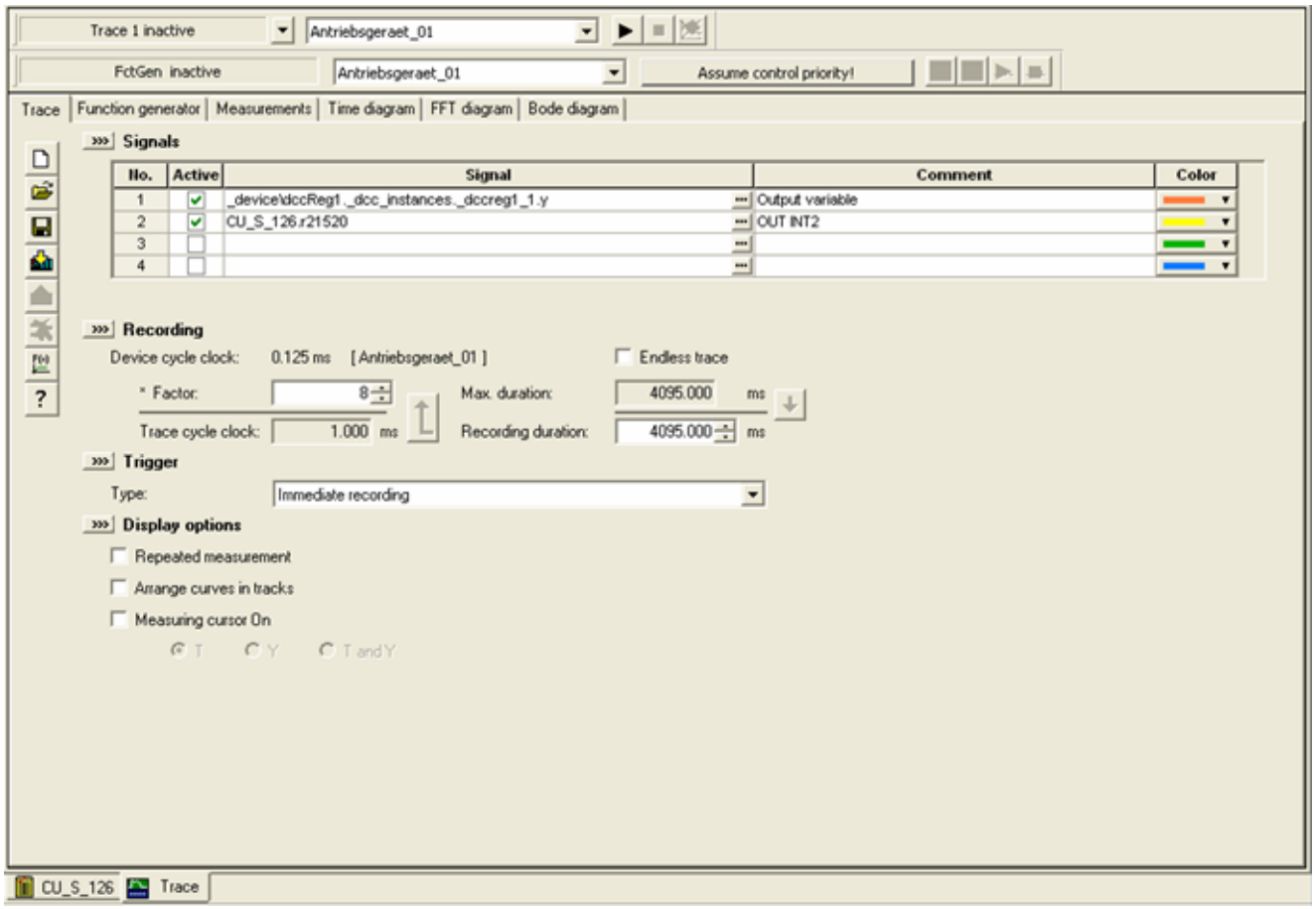


Figure 7-659 Parameterization of the trace

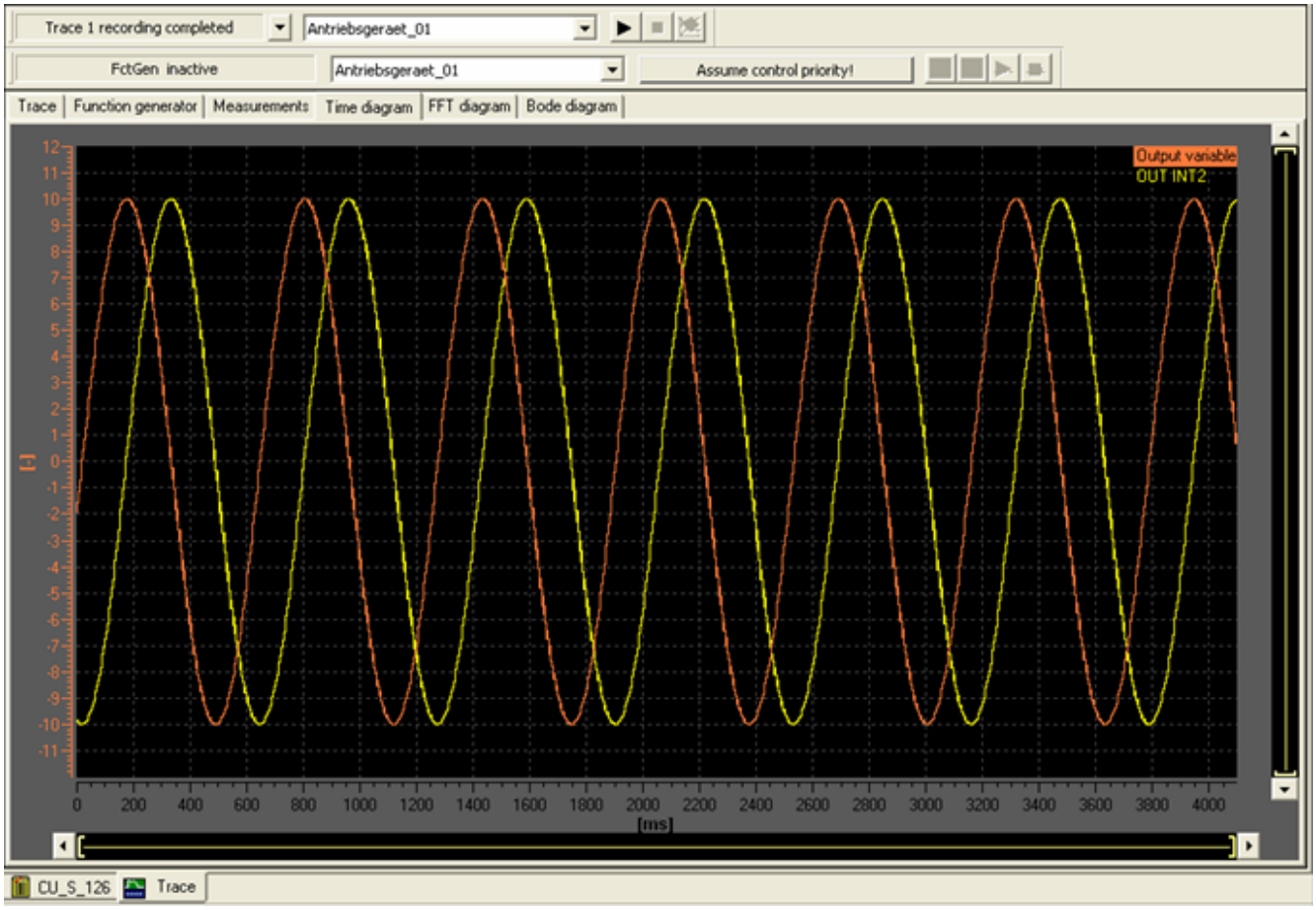


Figure 7-660 Display of the recorded signals as timing diagram

Note

When inserting block instances and connections online, the trace of the signal must be restarted.

Comparing DCC charts

For SINAMICS DCC charts on the drive object (DO) and in the DCC library, a detailed comparison of two DCC charts can be started in the project comparison as of STARTER V4.4 / SCOUT V4.4. The detailed comparison of the DCC charts can be started from the project comparison on the DCC chart.

The chart contents of two charts are compared on the basis of a chart source or compilation. The comparison result is displayed in tabular form. This feature is only available for SINAMICS DCC charts in SCOUT, not for SIMOTION DCC charts.

The detailed comparison can be started for the following comparison attributes:

- Source code (time stamp comparison)
- Blocks
- Interconnections

- Execution groups
- @ parameters and external references

The detailed comparison enables possible inconsistencies between the DCC charts to be precisely identified when going online or when activating the test mode.

The detailed comparison offers the following options:

- If the comparison attributes (source code, blocks, interconnections, execution groups, @parameters) are not consistent, a comparison result is provided with detailed information.
- Tabular list of the comparison result
- If the DCC charts are inconsistent, you can jump to the appropriate position in the DCC chart in order to avoid unnecessary searching in the DCC chart.
- As with the project comparison, the detailed comparison of DCC charts can be performed with the same comparison partners. This means that the detailed comparison can be performed between DCC charts and the project. The DCC charts and the project can be in the offline-offline or offline-online states respectively.
- The DCC libraries are also contained implicitly as "DCC chart" comparison object.

Further information on the DCC detailed comparison can be found in the Project Comparison Manual or in the STARTER/SCOUT online help (Project comparison index).

Archiving a project

If during or after the commissioning, you want to archive the project, including the DCCs, on a data carrier, this is possible with **Project > Archive** in STARTER.

Creating documentation

Complete documentation

The example has been successfully configured and tested. You can now create the complete documentation for your example.

Chart reference data

Alternatively, the chart reference data also displays the execution groups with the block types and the execution sequence. Use **Options > Chart reference data** in the DCC editor to open the **Chart Ref** window.

In the **Chart Ref** window, click the  button to also display the blocks in the configured execution groups. To begin printing, click: 

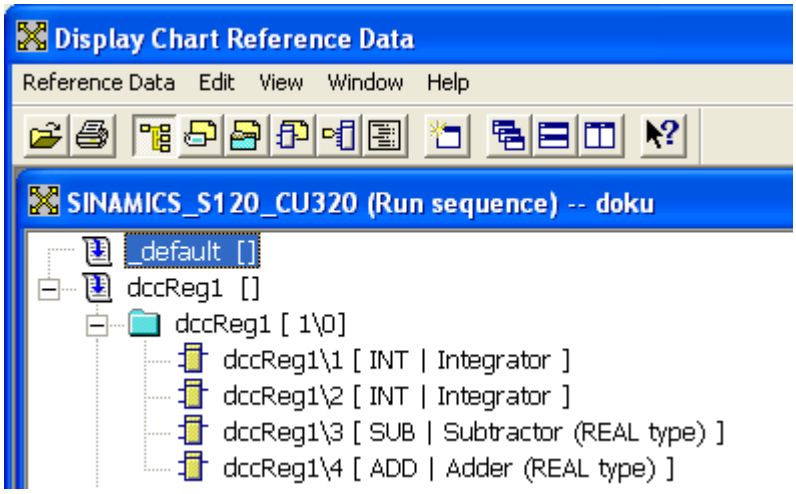


Figure 7-661 Chart Ref. window: Display chart reference data

Printing a chart

You should also print the chart to document the interconnection of the blocks. Because only one page was used on this chart, it suffices to print the current page. To begin printing, click: 

If your chart consists of several pages, we recommend that you print them individually in the page view. Click the  button to change to the page view.

Click  or  to customize the display so that all blocks are shown on a single page on the PC monitor.

Assignment of execution groups

To document the assignment of the execution groups, select the **Set execution groups** command in STARTER in the context menu of the chart. The **Set Execution Groups** window appears. Print a screenshot.

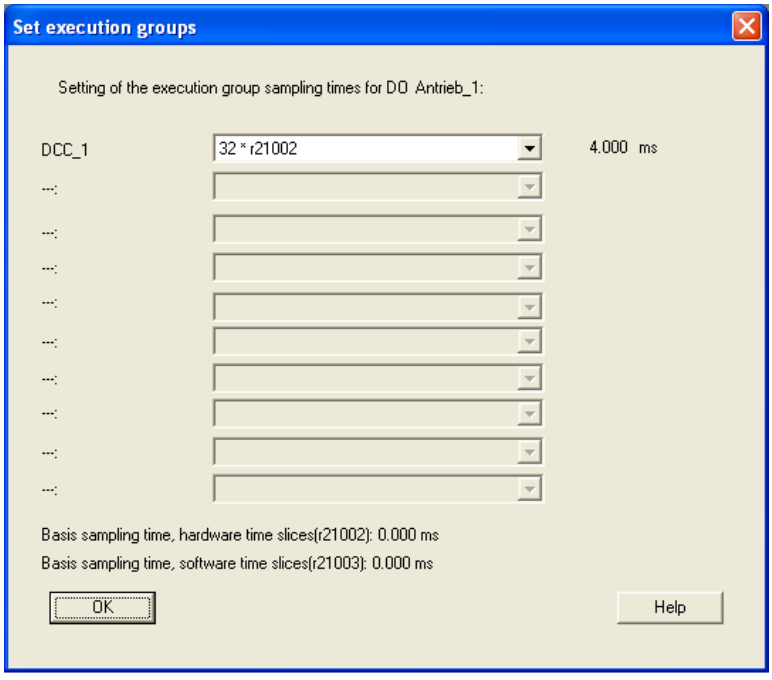


Figure 7-662 Set execution groups window

The change of the hardware r21002 basic sampling time displays the value for p0115[0] in the CU ramp-up of 1 ms. Here p0115[0] was changed from 4000 µs (factory setting) in offline mode to 1000 µs and then downloaded to the CU.

Update to a new SINAMICS version

When using DCC charts, the project must always be updated in the Engineering System and downloaded to the target device.

7.4.4.3 Connecting the DCC to the drive

Overview

Only DCB connections of a DCC chart declared as BICO parameters can be connected to the connector inputs (CI) and connector outputs (CO) of the drive.

All block connections of data type REAL that are published as BICO parameters are per-unit variables. This means that calculations are performed with per-unit signal values within DCC.

(1.0 corresponds to 100%). The conversion to the connector units of the drive is performed automatically.

With all other data types, no conversion to a per-unit variable takes place.

Calculating a DCC chart with per-unit variables

Example 1.1 (interconnecting input value)

Disconnect any online connection to the drive unit in STARTER.
– p1020 = 1 and
– p1021 = p1022 = p1023 = 0
(function block diagram 2505) are set on a drive object of the SERVO (the "Extended setpoint channel" function module must be activated) or VECTOR type. Fixed speed setpoint 1 (p1001 in function block diagram 3010) is then output on r1024 (Speed setpoint active).

The following are also set:
– p1001 = 1500 rpm and
– p2000 = 3000 rpm (reference speed).

Insert a DCC on the drive object.

Insert the ADD block in this chart. Connections X1 and Y of the ADD block are published as (interconnectable) BICO parameters.

Right-click twice on input X1. The context menu opens. Select **Interconnection to operand...**. In the signal selection window DCC now select r1024 (CO: actual speed value effective) and confirm with **OK**.

Compile the chart by selecting **Chart > Compile > Charts as program...**.

Select Set execution groups... in the context menu of the DCC, set the only execution group that is present, e.g. to 20 * r21003 and confirm with **OK**.

Go online with STARTER and download the project to the target system.

Then online in the DCC editor, switch the test mode on with **Test > Set test mode**. Now right-click the output of the adder and select **Log on connection** in the context menu.

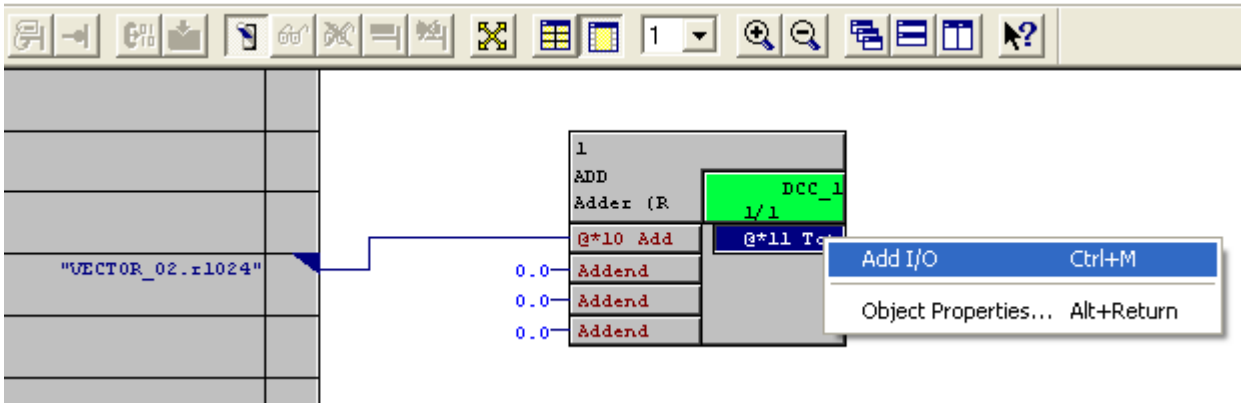


Figure 7-663 **Log on connection** context menu

The current output value (= per-unit speed setpoint = 1500 rpm / 3000 rpm = 0.5) is then displayed.

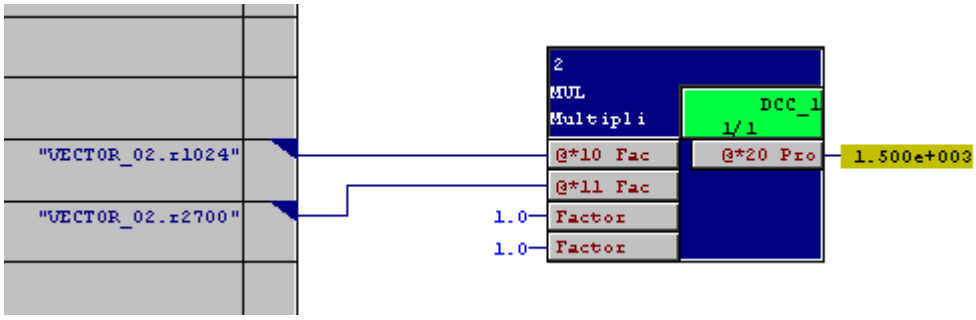


Figure 7-664 Connection value on example 1.1 is displayed

Example 1.2 (interconnecting the output value)

In the DCC chart, a value of e.g. 1.2 should be interconnected with the basic device. The value for this example will be configured as an initializing value to the X input of the NOP module. The module output (Y) has therefore the constant value 1.2, and will be configured as an interconnectable BICO parameter (@*20).

If a connector output of DCC is interconnected with the basic device, this will always be interpreted as a referred value of the basic system. The unit-related absolute value in the basic device now depends on which unit and therefore which reference variable the selected connector input has in the basic system:

When interconnecting the connector input for speed setpoint p1070 (function plan 3030), the referred output value from the DCC chart will automatically be multiplied with the reference variable "Reference speed" p2000 (in the example p2000 = 1500 min^{-1}). In the SINAMICS basic system, a speed setpoint of 1800 min-1 (= p2000 * 1.2 = 1500 min^{-1} * 1.2) will thus become effective.

For this reason, always take note in the case of BICO interconnections with the basic system if you work with absolute variables or with referred variables in DCC. Since the connector output of the DCC chart in the case of interconnection with a unit-related connector input of the basic system will always (internally) be multiplied with the reference variable (belonging to the unit).

Calculating a DCC chart with absolute variables

Example 2.1 (interconnecting input value)

If you want to work with absolute variables in the DCC (in our example using speed values), you have to convert the referenced value to which the variables are automatically converted by the drive into an absolute variable (in our example, a speed) using the reference variable. For DCC, the reference variables are provided as monitoring parameters r2700... r2707 which have no units and can be interconnected. Parameters r2700... r2707 have the special feature that the value is transferred 1:1 to DCC without being divided by the reference variable.

On SERVO type (the "Extended setpoint channel" function module must be activated) or VECTOR type drive objects,
– p1020 = 1 and

– p1021 = p1022 = p1023 = 0
(function block diagram 2505) are set. Fixed speed setpoint 1 (p1001 in function block diagram 3010) is then output on r1024 (speed setpoint active, function block diagram 3010).

The following are also set:
– p1001 = 1500 rpm and
– p2000 = 3000 rpm

r1024 (speed setpoint active, function block diagram 3010) then assumes the value 1500 rpm.

The interconnectable monitoring parameter for the r2700 reference speed indicates a value of 3000 in the expert system.

The multiplier MUL is dragged into the chart.

The 1st factor (X1) is published as a BICO parameter and interconnected with r1024. → Factor 1 = 0.5 (referenced speed setpoint active)

The 2nd factor (X2) is published as a BICO parameter and interconnected with r2700. → Factor 2 = 3000.0 (value of reference variable p2000)

At the multiplier output this results in:

0.5 * 3000.0 = 1500.0

The speed setpoint r1024 is available again in DCC in rpm.

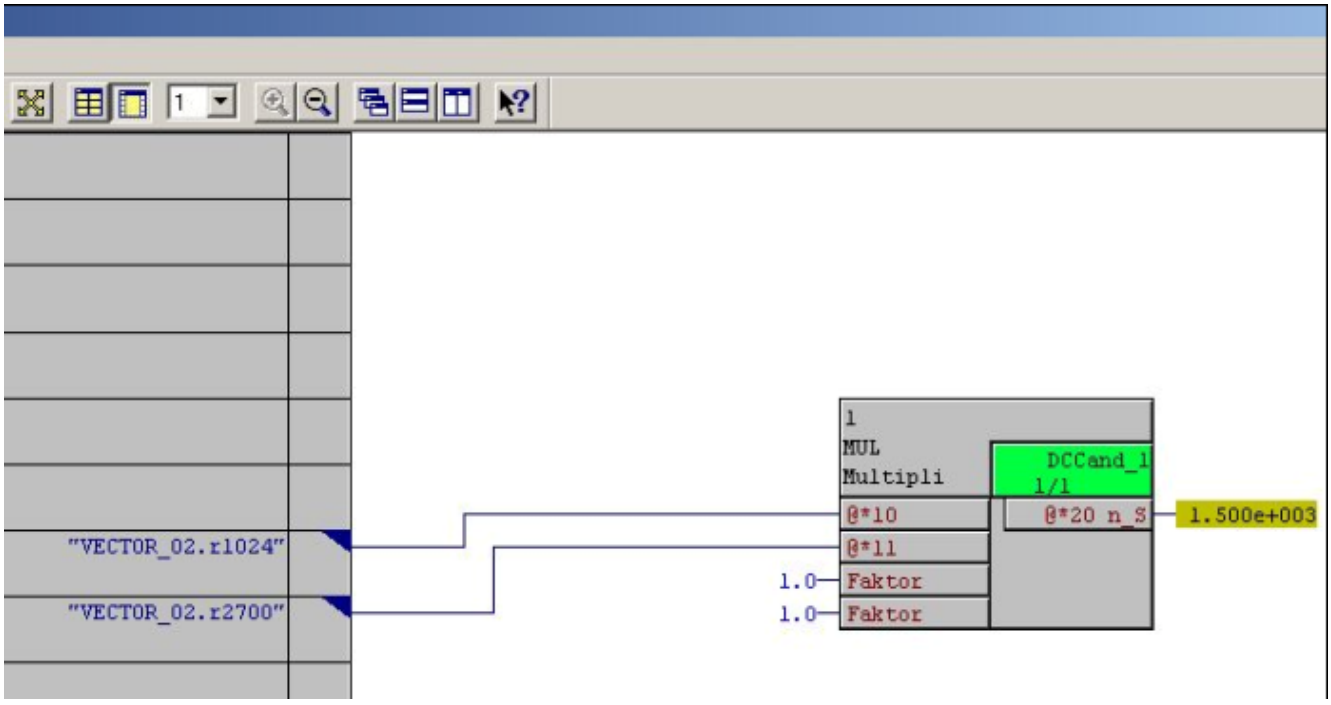


Figure 7-665 DCC for example 2.1 in online test mode

The block was automatically assigned to an execution group when inserted into the chart. The free execution group 20 * r21003 (selected at random) has been assigned to the execution group in the **Set execution group** menu.

Example 2.2 (interconnecting output value)

Absolute values are used in the DCC to calculate the torque in Nm units. The torque setpoint calculated is to be passed on to the "Extra torque M_extra 2" (p1513, function block diagram 6060) connector input. For this to happen, the absolute value must be converted into a referenced torque while still in the DCC. When connecting a real type (== floating point) connector output (CO) from the DCC with connector input (CI) p1513 M_extra 2 [torque in the Nm unit] on the drive, the automatic conversion in the BICO connection always assumes that DCC is providing a referenced signal.

Note

The LIM type block supplies the absolute torque of 0.204625 Nm calculated in the chart at its Y output. This absolute value is now divided by the reference variable for torque r2703 = 0.8185 Nm and the referenced torque value for interconnecting with the drive is thereby calculated. The referenced torque value is available at the block DIV's output and is published as BICO parameter r21530 (value = 0.2499 = 0.204625/0.8185). Block NOP1 has only been inserted so that the value of reference torque r2703 can be shown in the chart at its r21527 output published as a BICO parameter.

Monitoring parameter r1515 (total extra torque, function block diagram 6060) is only calculated if the speed controller is enabled and the drive is magnetized (r0056.4 = 1 = yes). (A machine must be connected to the motor modules and be running for this purpose.) Otherwise the value is set to zero.

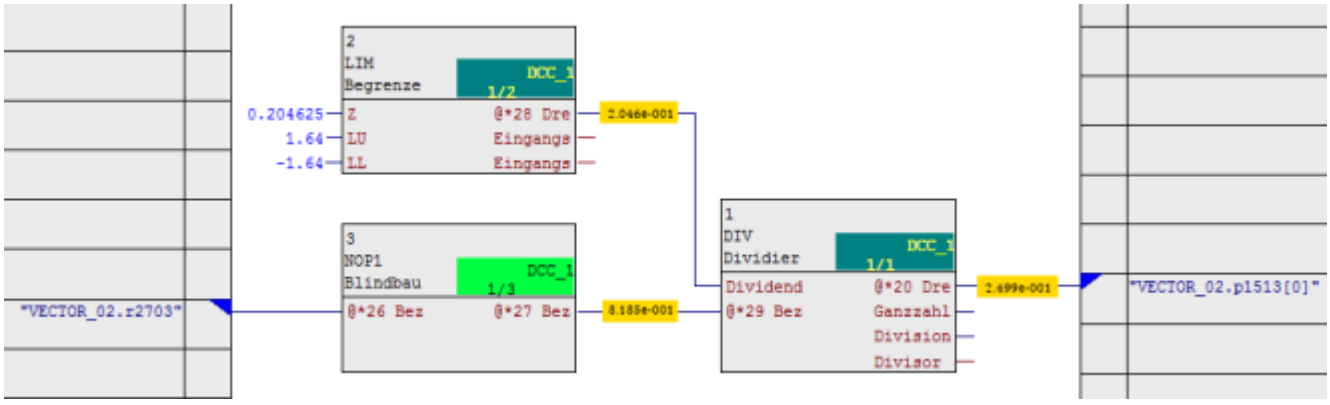


Figure 7-666 Interconnecting output value

Interconnecting DCC signals with communication interfaces IF1 and IF2

Preliminary remark

The basic system can be connected to process data interfaces IF1 and IF2 via free telegram configuration using BICO (p0922 = 999) or via (standard) telegrams depending on p0922. To interconnect with the basic system, the DCC connections needed must be published as BICO parameters. These DCC parameters should always be interconnected with the basic system in the DCC Editor by calling the context menu **Interconnection to operand**.

Interconnecting received process data with DCC

When interconnecting the received PZD data (see SINAMICS S Parameter Manual, function block diagrams 9206 and 9204), the particular way in which the connector outputs (COs) of PZD processing behave (for IF1 r2050, r2060 and for IF2 r8850 and r8860) should be noted. These COs (connector outputs) can either provide their signals in whole numbers (integers) or as floating point values (REAL). The data type provided is determined by the first signal linked to this CO. This can also be determined by a standard message frame having been selected (p0922 |= 999) on a drive object. The PZD COs are thereby automatically interconnected according to the definition of the message frame selected. This interconnection is not canceled when resetting p0922 = 999 == free message frame configuration with BICO.

p1070 = 2060[2] is set for example. r2060[2] (PZD received word 3 and 4) is thereby connected to the main setpoint (in function block diagram 3030) of a REAL variable of the basic system. This means that only connector inputs from DCC with the REAL signal data type can be interconnected on r2060[2].

Note

Connections to integer inputs can be made both online and offline in the DCC editor; a corresponding error message is only issued when the chart is downloaded.

Interconnecting sent process data with DCC

The send data is interconnected as with any other BICO connection.

Detailed descriptions about this topic are contained in the *SINAMICS S120 function manual*.

7.4.4.4 DCC SINAMICS specifications

Rules for assigning names in the DCC editor

Names are used for data exchange between SCOUT/STARTER and the DCC editor.

The names in the DCC editor must therefore abide by the following rules:

- Basic chart:
  - No keyword or previously defined name permitted
  - Must start with a letter or underscore
  - May contain numbers, letters, and underscores
  - An underscore must be followed by a number
- Subchart:
  - Must start with a letter
  - May contain numbers and letters
  - Underscores are not permitted
  - Keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined.
- Block instance:
  - Can start with a number
  - May contain numbers and letters
  - May not start with an underscore
  - Keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined.
- Execution group:
  - Keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined.
- BICO parameter:
  - No previously defined parameter numbers permitted
  - In the DCC editor, a check is performed to determine whether the parameter number has previously been defined.

Field/name lengths and conventions

Field/name lengths and conventions

Object	Length	Remark
Chart name	22 *)	May not contain the following characters: \ . : / * ? " < > | # % ( ) Use of the "_" character is subject to particular specifications.
Chart comment	255	All ANSI characters are permitted.
Execution group	22	Only letters, numbers and underscores may be used.
Block instance comment	80	All ANSI characters are permitted.
Parameter comment	80	All ANSI characters are permitted.
Block name (instance name)	16 *)	May not contain the following characters: \ . : / * ? " < > | # % Use of the "_" character is subject to particular specifications.
Global operand	Max. 49	Interconnection with BICO parameter.

*) The chart name and block name must jointly consist of a maximum of 24 characters, including separating characters.

Representation of the dynamic value display

The values are output next to the connections, according to their data type. They are displayed on the screen with a colored background.

Table 7-614 Representation of the dynamic value display

Representation	Meaning
Blue on white	Representation of the values in edit mode (offline)
Black asterisks on yellow	Values during transfer to the dynamic display
Black value on yellow	Representation of the values read from the drive object in test mode
#### on a red background	Representation of values while the dynamized values required from the drive object are missing (fault, overload)

7.4.5 DCC for SIMOTION

7.4.5.1 Overview

Introduction

This product brief is intended for experienced SIMOTION users who are not yet familiar with the DCC program package. Using a short example, you will find out how to create a project after starting up SIMOTION SCOUT, create a DCC chart, interconnect blocks, compile the chart, download it to the target device and test it online.

Note

In most cases, there are a number of options for working with the DCC editor (e.g. using the keyboard). In this example, the quickest or most suitable option is used. Apart from a few exceptions, no alternative methods of operation or procedures are explored here.

Software requirements

The software requirements for DCC are the same as those for SIMOTION SCOUT.

You require an appropriate license to use the DCC editor.

Please refer to the following table for information on which kernel version supports which DCB libraries.

Kernel version	DCB lib version
SIMOTION 4.1.5	4.1.2, 4.1.4, 4.1.5
SIMOTION 4.2	4.1.5, 4.2

SIMOTION system integration

Execution level, execution group and execution sequence

DCC tasks and execution groups

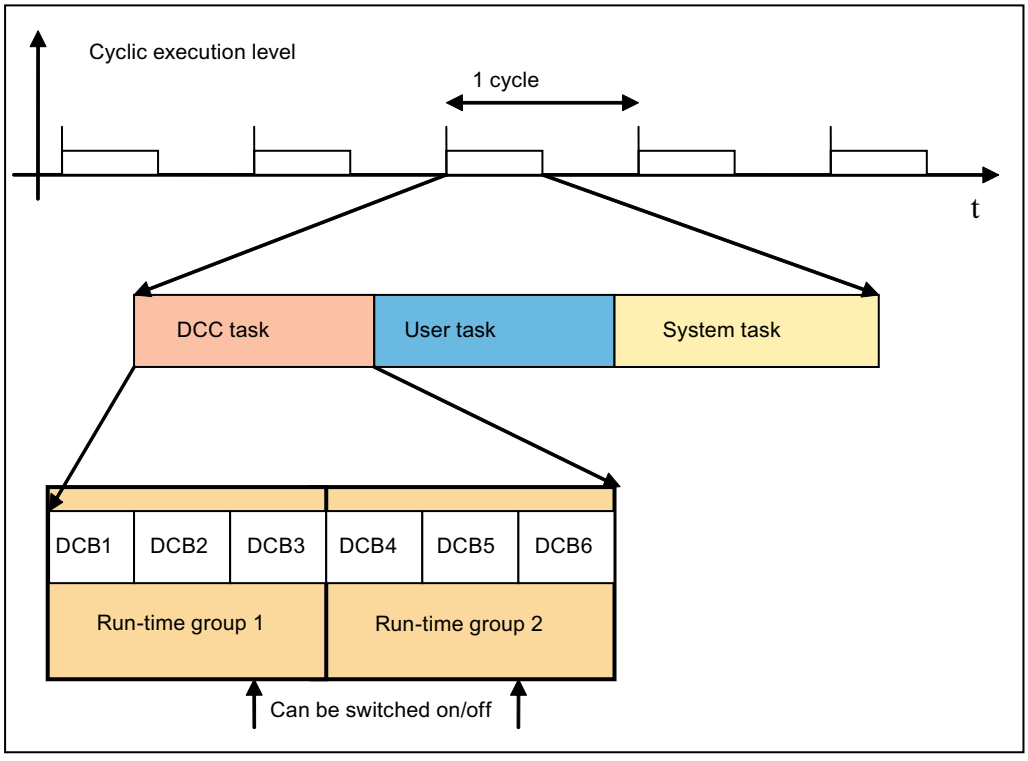


Figure 7-667 Sequence model for blocks

Tasks

A DCC task is available for the user programming in each of the five cyclic execution levels (in descending order of priority):

- T1: servodcc in the servo level
- T2: ipodcc in the IPO level
- T3: ipodcc_2 in the IPO_2 level
- T4: dccaux in the DCCAUX level
- T5: dccaux_2 in the DCCAUX_2 level

Further information on the execution system of SIMOTION can be found in Section *Execution system / tasks / system cycle clocks* of the *SIMOTION SCOUT Basic Functions* Function Manual.

Execution groups

The available tasks (sampling time and execution sequence in the system: T1 to T5) are assigned execution groups. The blocks are embedded into these execution groups. The execution groups therefore allow the task to be structured or divided as required by the user; e.g. dancer control, setpoint processing, etc. The execution groups are embedded sequentially into the blocks. **Only** blocks **from one** basic chart may be contained in an execution group.

You can use the "Enable" attribute to activate and deactivate an execution group. Normally, execution groups are processed in cycles; however, the enable attribute can be used for switching individual execution groups on and off.

A BOOL-type block output can be connected to enable an execution group or a block group. To do this, highlight the connection to be interconnected and select the **Interconnection with execution group**... command in the context menu.

Note

The **Interconnection with execution group**... command is only supported with DCC SIMOTION.

Execution sequence

For tasks: The execution sequence corresponds to the sequence in which execution groups and blocks are inserted within a task.

The same applies to groups: The sequence in which the individual blocks are inserted is the execution sequence within the execution group.

You can change the execution sequence.

Setting the system cycle clocks

The properties that determine the execution behavior of a task are configured in SIMOTION. You can carry out parameterization in the Properties dialog box of the CPU under **Expert execution system > Set system cycle clocks**.

No multiple insertion of blocks in different tasks

Inserting a block several times in different tasks is not permissible.

Changing the execution sequence

- Click **Edit > Execution sequence** or the  button.

The **Execution Sequence** window appears: In the left-hand side of the window, you will see the structure of the task. In the right-hand window, the content is displayed. The default integration position of the blocks is in cyclic task **T2**, in the default execution group that has the same name as the chart.

To move blocks that have already been inserted from task **T2** to higher-priority task **T1**:

- Double-click the **T2** symbol.
- The execution groups are displayed in the right-hand window:
  - When you highlight the execution group, the blocks it contains are displayed (including the chart/block name, comment and position).
  - Highlight the execution group that you want to move in task **T2** and use drag-and-drop to drag it to the task **T1** symbol in the left-hand window. The right-hand window is now empty.
  - When you double-click task **T1** to open it, the execution group that you have moved will be displayed there.
- Close the window via **Chart > Close** and proceed to the next step.

HMI variables (publishing of variables and @ variables)

HMI variables

You can declare block inputs and outputs as **HMI variables** and therefore generate a static interface for these for use in your system visualization. This interface is largely static from an HMI point of view, i.e. not every change made to the DCC configuration demands that the address information of the HMI system be reimported.

The memory image with the HMI variables is **not** automatically deleted during compilation. It is only deleted on request.

HMI variables that are removed from the DCC when the block that defined them is deleted remain in the memory image until **reorganization** is requested during compilation in the DCC.

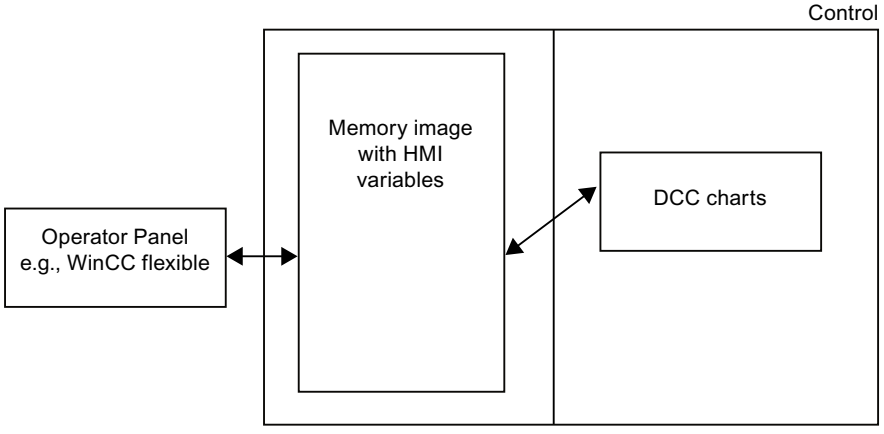


Figure 7-668 HMI variables

@ variables (SIMOTION)

Structure of the @ variables

In the comment, block inputs and outputs are published by the variable name preceded by @.

Table 7-615 Structure of comments in SIMOTION

Comment	Meaning
SIMOTION: @name <<variable comment>>	The connection is entered as an HMI variable in SIMOTION. Name is the part before the first blank (whereby you must adhere to the ST conventions (Page 5609)).

The text which follows the variable name (separated by a blank), is transferred as a variable comment. It is then displayed in the symbol browser. The variable identifier results from the variable name.

The data type of the published input/output is taken over by the block connection. Where necessary, it is mapped onto the appropriate data type of the engineering system.

No help function can be created for @ variables.

Exporting to WinCC

Procedure

A created DCC chart can be exported to WinCC, whereby the export specifications of the SIMOTION SCOUT also apply for DCC charts. However, for this to happen the OPC-XML export must be activated in the Properties dialog box of the chart in SIMOTION SCOUT.

In the engineering system, right-click the chart and select **Properties** in the context menu. The export is activated via **Enable OPC-XML** on the **Compiler** tab.

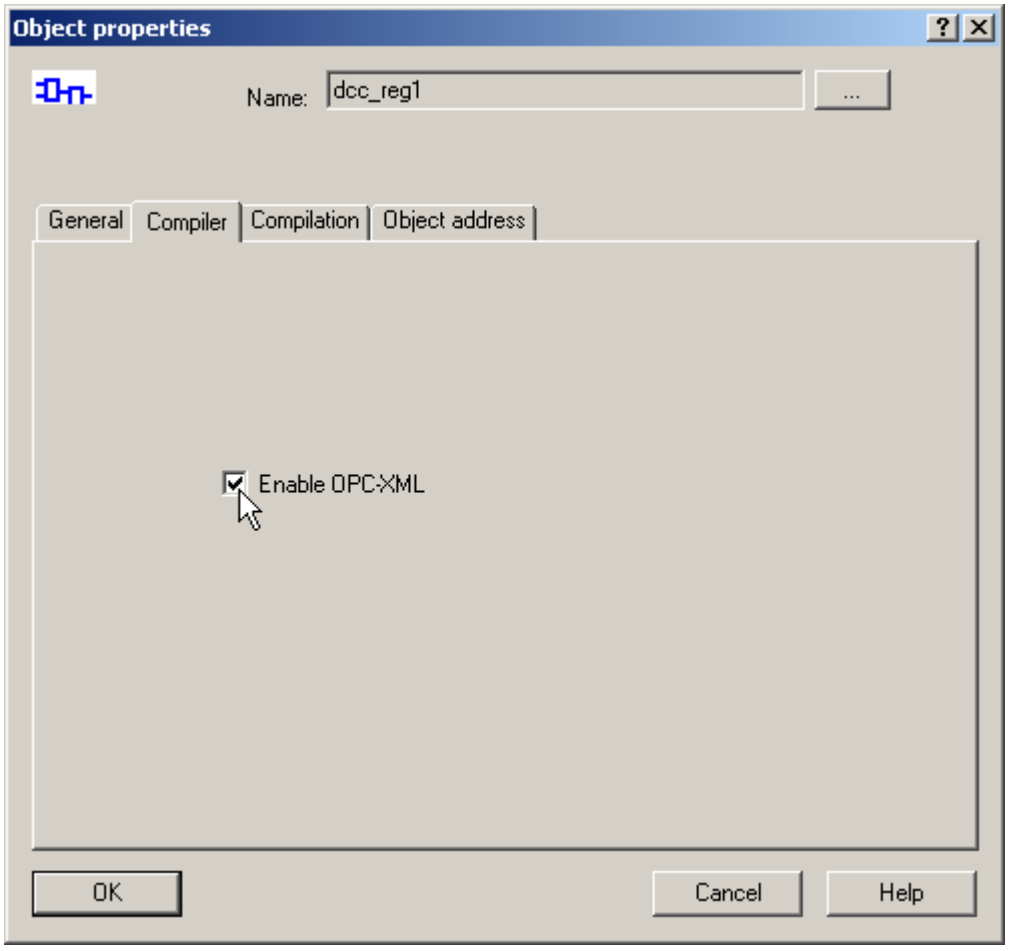


Figure 7-669 DCC chart properties - OPC-XML export

The export can be performed in SIMOTION SCOUT via **Options > Export OPC data**.

Interconnecting with SIMOTION variables

Interconnecting with SIMOTION variables

You can interconnect with global operands (SIMOTION variables) in the DCC editor.

Note

Please observe the Specifications for assigning names in DCC SIMOTION (Page 5609).

Procedure

Interconnection can be performed as follows:

1. Select the block connection to be interconnected.
2. Execute the **Insert > Interconnection to operand** command in the menu. The interconnection to a global operand can also be performed via the context menu of the block. The **Symbol Input Help** window appears.

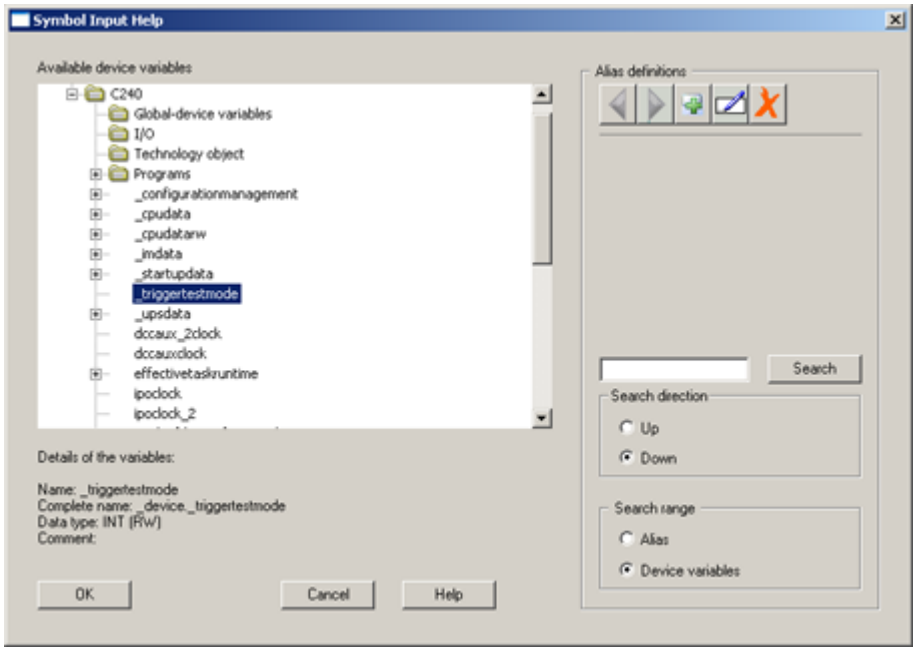


Figure 7-670 **Symbol Input Help** window

3. You can now navigate in the CPU or technology object assigned to the chart and select the device variable to be interconnected – the application highlights compatible device variables in bold-face type.
4. Click **OK** to close the window.

From this point on, the selected block connection is interconnected with the global operand and the global operand is represented by a sheet bar variable in the DCC editor.

Note

The global variables and the I/O variables must first be created and then the interconnection performed.

Note

You can search for a variable name or a parameter text in the search field of the **Symbol Input Help** dialog box.

Interconnection to array elements

You can interconnect array elements of ST programs in the DCC editor.

Procedure

Interconnection can be performed as follows:

1. Create a new ST program with an array variable in the SIMOTION SCOUT engineering system using the **Insert ST program** command.
2. Right-click and select **Interconnection with operand** from the context menu. The **Symbol Input Help** window appears.

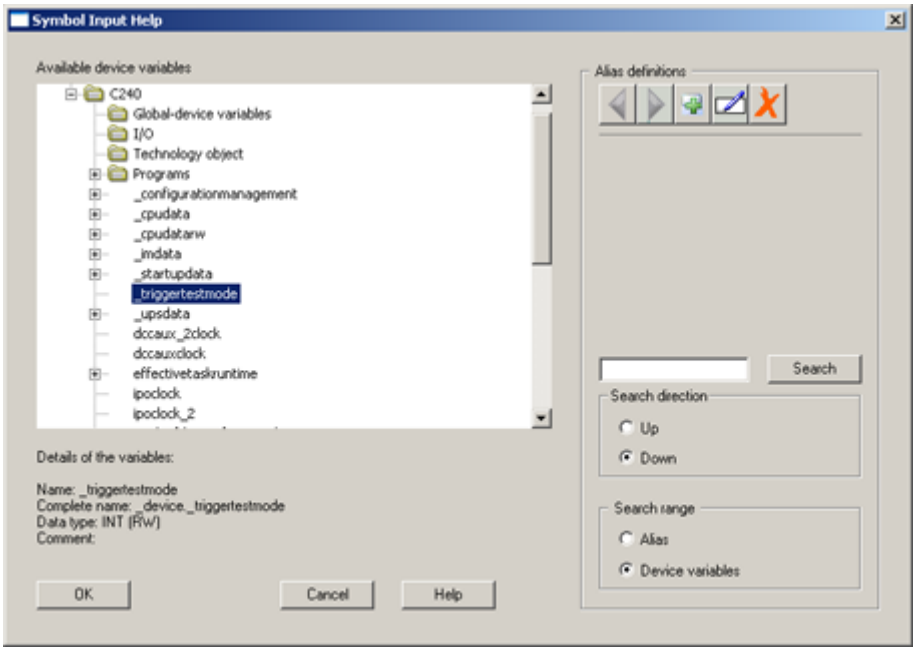


Figure 7-671 Symbol input help window

3. Select the array variable to be interconnected.
4. Enter the number of the array element to be interconnected.
5. Click **OK** to close the window.

From this point on, the selected block connection is interconnected with the array element and the array element is represented by a sheet bar variable in the DCC editor.

Alias parameter

You can assign alias identifiers in the DCC editor for variables. The reason for assigning an alias may be that a name is too long - the DCC editor only accepts a maximum of 49 characters for identifiers.

Procedure

An alias can be created as follows:

1. Right-click the block connection and select **Interconnection with operand** from the context menu. The **Symbol Input Help** window appears.
2. Select the connection that is to be defined by the alias.

3. Click the button from the alias definition area.

4. Enter the alias name in the entry field.
5. Close the window.

The alias has been successfully created and can now be used - the computer now resolves the alias as required, e.g. within the framework of the compilation.

DCC and SIMOTION trace

As accustomed with SIMOTION-based devices, you can trace each block connection as a variable or use it in the trigger condition in SCOUT.

7.4.5.2 Working with DCC SIMOTION

Preliminary remarks on configuration

Preliminary remarks

The following is a brief explanation of what you will be configuring in this chart.

Configuration example

The configuration example deals with a straightforward oscillating circuit that creates sinusoidal oscillation at its output.

It will only take you a few minutes to create the chart; then you can execute it in test mode as a demonstration.

The following blocks are used:

- Two integrators (**INT**)
- One inverter (**SII**)

As indicated by the differential equation **f"(x) = - f(x)**, the oscillating circuit is comprised of two integrator blocks that are linked by negation.

The frequency of the oscillating circuit is determined by the integrating time constant at the integrators.

The oscillation amplitude is specified by the initial value at the integrator output.

The DCC editor can be started via the SIMOTION SCOUT engineering system.

Configuration example structure

The configuration example is divided into the following steps:

1. Creating a new project
2. Inserting the DCC in the project
3. Inserting blocks in the DCC
4. Interconnecting blocks in the DCC
5. Compiling the DCC in the DCC editor
6. Downloading the compiled DCC program to a CPU
7. Starting the CPU
8. Switching between process and laboratory mode
9. Setting test mode and the test mode types
10. Monitoring values in laboratory and process modes
11. Logging on and logging off connections for monitoring
12. Using the "Enable" attribute for execution group control
13. Creating complete documentation for the example
14. Information on the chart reference data function

Creating a project

- Create a new project in the SIMOTION SCOUT engineering system, e.g. **dcc_ex**, see Creating a project (Page 5424).
- Create a new device, e.g. a D445, using **Create new device**.
- Select the package **TPdcblib_SIMOTION_4_2.3.0 [x.y]** and close the window.
- You can now insert a chart.

Inserting a chart

- In the project navigator, double-click **Insert DCC** under **Programs**.
- Now give your chart another new name. The text field for the DCC name is selected automatically and the cursor is activated.
- Edit the new name here: **dcc_reg1**.

- If the **Open editor automatically** option is activated, the DCC will be started automatically. Alternatively, you can open the DCC by double-clicking **dcc_reg1**.
- If a library has not yet been imported, at this point you will be prompted to import one: Select **TPdcblib_SIMOTION_4_2.3.0 [x.y]** in the left-hand window, then click **>>** and finally **Close**.

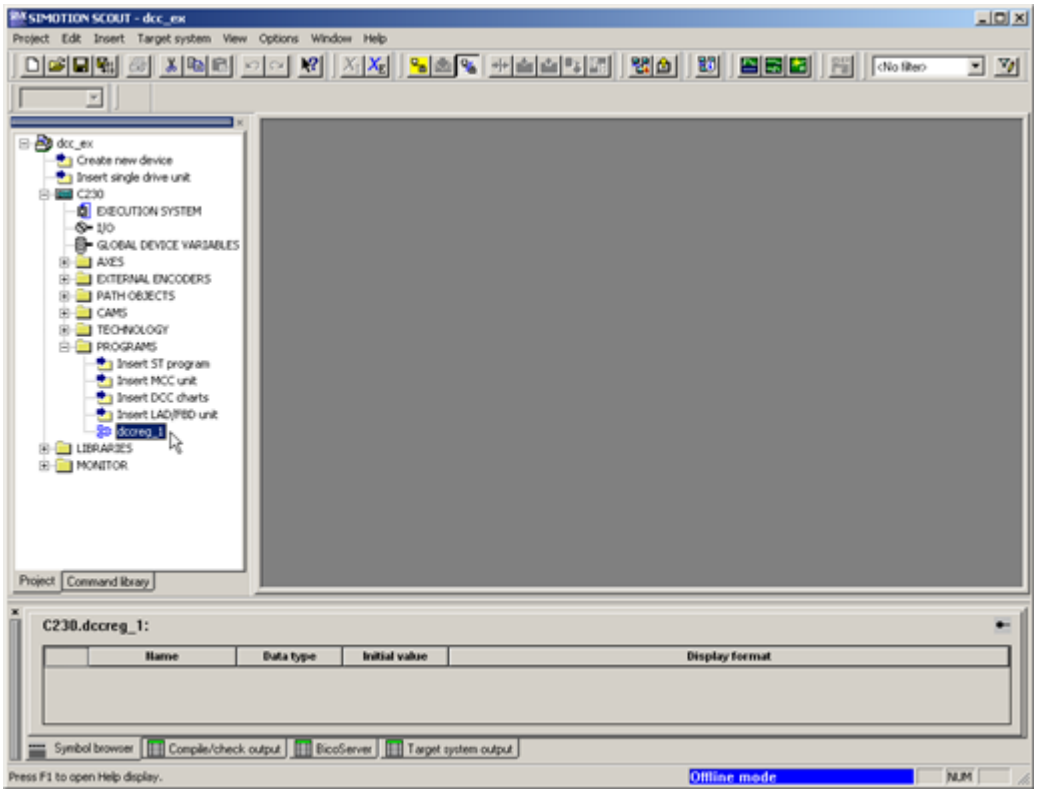


Figure 7-672 Inserting a chart

This creates the project structure and a chart. All that remains is to create some activity within the chart, i.e. by inserting blocks and interconnecting them.

Inserting blocks

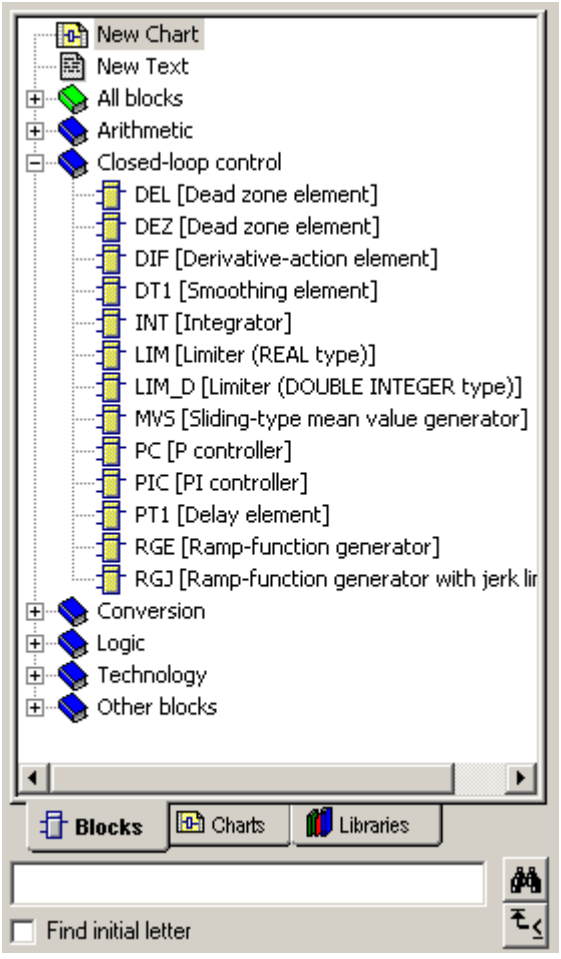


Figure 7-673 Inserting a block

- Open a block family in the **Closed-loop control** family with the closed-loop control blocks.
- Select the required block and insert it in the chart using drag-and-drop. Only the outline of the block in dashed lines is displayed during the copying procedure. Release the mouse button at the required point.
- To search for a block, enter its name in the input field of the block catalog and click the **binoculars** button. The search process begins. Once the block has been found, insert it in an empty space on the chart using drag-and-drop.

Note

If blocks are superimposed on the chart with other elements, such as other blocks or the sheet bar, the superimposed block will be displayed in gray and its connections will not be visible. You must reposition the blocks to ensure that all block information can be viewed.

Changing to the page view

To change to the page view from the overview representation, right-click an empty space in the chart and select **Display this page** in the context menu that appears. The names of the block connections are displayed in this enlarged view.

You can also switch to the page view and back to the overview again by double-clicking an empty area on a page.

Configuring the block display

You can change the display of the blocks. You can change the block width using **Options > Settings > Blocks / sheet bars width....** You can change the designation of the inputs and outputs of the blocks using **Options > Settings > Display** in the **Connections** submenu.

The block type can be displayed in the form of both text (name) and graphics (control symbol). This can be configured via **Options > Settings > Display** in the submenu **Block headers**.

Interconnecting blocks

Procedure

- Select the **Y** output of the first **INT**egrator, followed by the **X** input of the second **INT**egrator.
- Select the **Y** output of the second **INT**egrator, followed by the **X** input of the inverter (**SII**).
- Select the **Y** output of the inverter (**SII**), followed by the **X** input of the first **INT**egrator.

The autorouter creates the connecting lines from the outputs to the inputs and they are then interconnected.

Parameterizing block connections in the chart

For the first integrator INT 1/1, the following initial values should be assigned to the connections: LL = -10.0, LU = 10.0, SV = 2.0, Ti = 100 ms

For the second integrator INT 1/2, the following initial values should be assigned to the connections: LL = -10.0, LU = 10.0, Ti = 100 ms

To do this, open the **Properties - Connection** window of the associated block connection with a double-click. Enter the initial value for **Value** and click **OK** to close the window. Note that for the input of the values for Ti, the "ms" unit must follow the numeric value 100 without any spaces.

As the connections mentioned above are not interconnected, the entered values also remain valid during the cyclic processing of the chart.

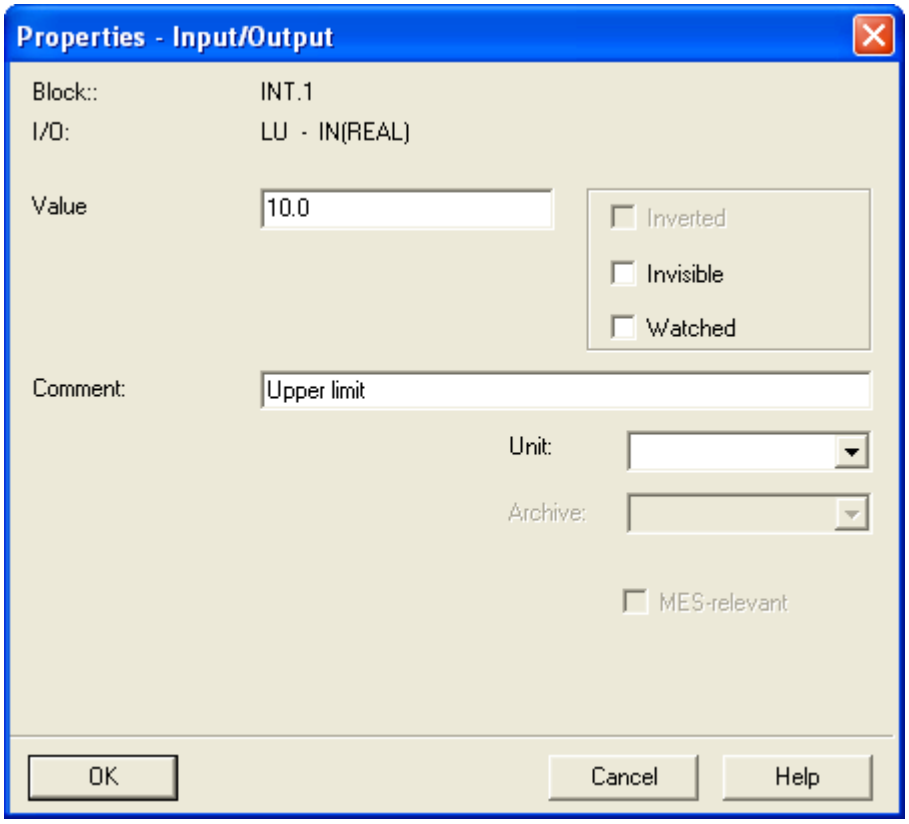


Figure 7-674 Properties window of the "LU" block connection

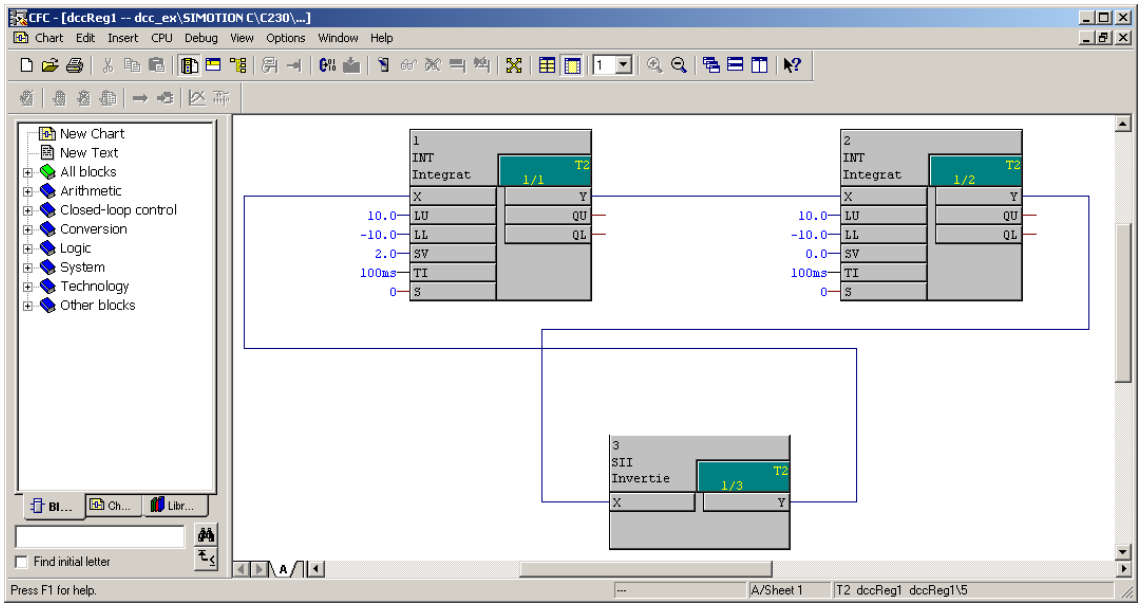


Figure 7-675 Chart "dccReg1" with interconnected blocks. Some connections have been assigned values that differ from the default values.

Structures for DCB block connections

As of DCC 2.1/SIMOTION RT 4.2, user-defined data types structured for DCC SIMOTION can be used for block connections. The structure definitions are taken from the DCC block library.

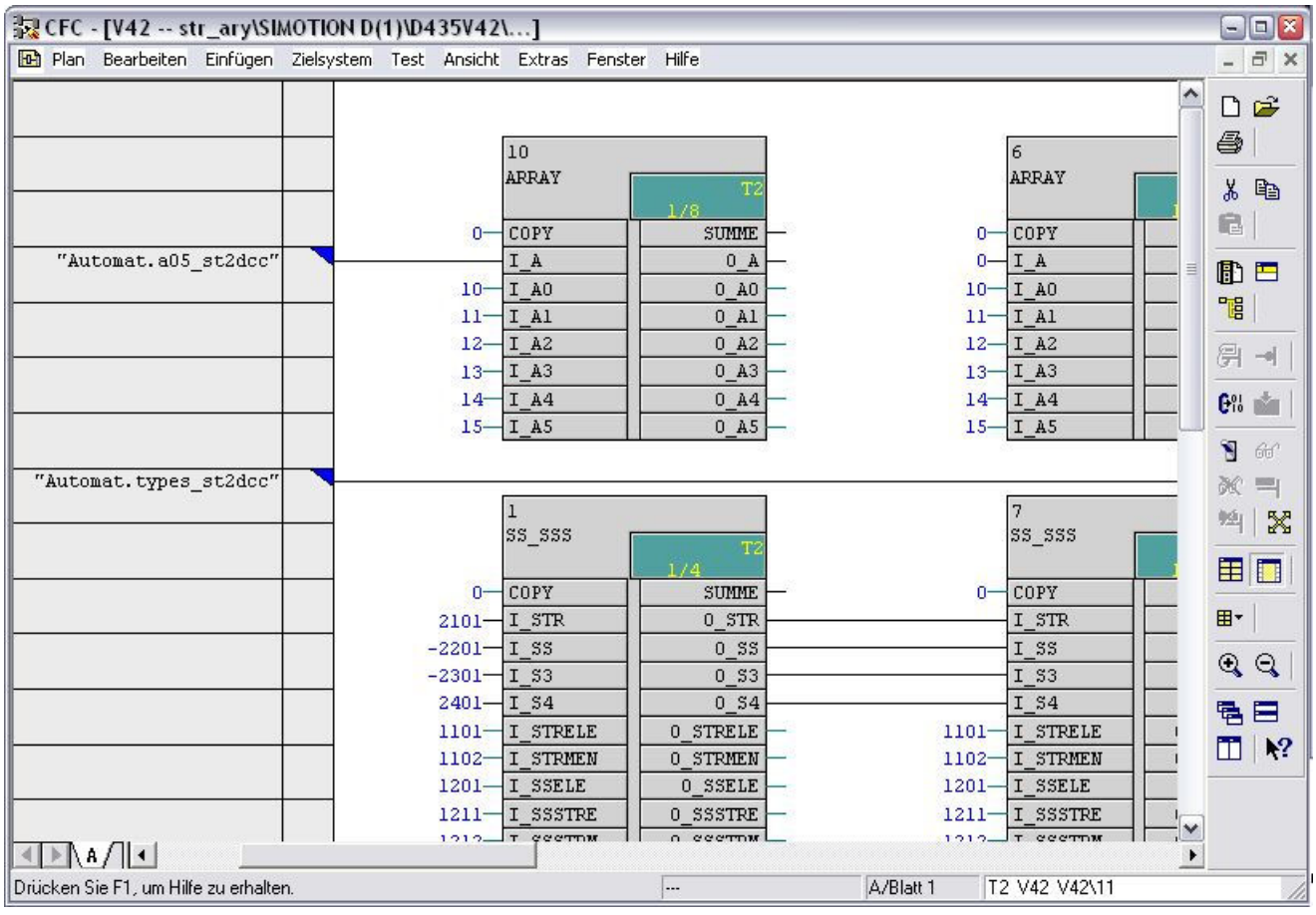


Figure 7-676 Interconnected structures

Purpose

Signals associated with one another from a technological perspective can be combined to form structures.

For example, this means that a setpoint, factor, and control signal can be transferred as related components of the same structure. The structure can be extended at block level; this means there is no need to make any changes in the DCC chart.

Creating structures

New structures can only be generated by blocks that have been created with the SIMOTION CLib Studio. Structures cannot be defined in the DCC editor.

Transferring structured block connections to/from TO variables

The structures must be the same in the DCB library and in the TP definition.

It is not possible to accept structures from TPs. The structure must be defined in the DCB block.

Assigning parameters for structured block inputs

An initialization value is pre-assigned to structured block inputs.

Call up the object properties for a structure or an elementary data type in the structure in order to edit them. Double-click the block connection to open the **Select structure element** dialog. Highlight the structure element you wish to edit and open the **Properties** dialog by either clicking the **Properties** button or double-clicking the structure element.

The **Properties** dialog can be called up in both edit mode and test mode.

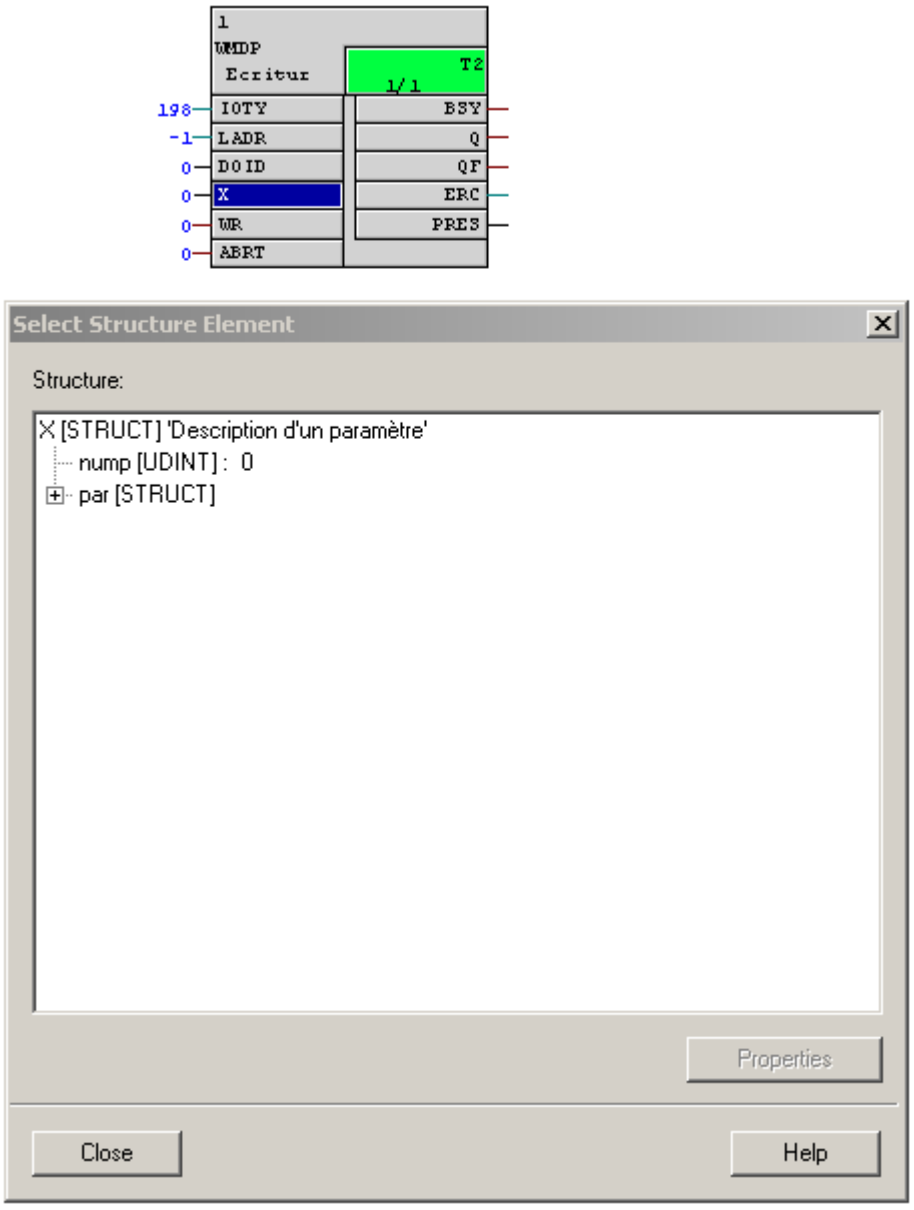


Figure 7-677 Assigning parameters for structured block connections

Arrays are represented by individual symbols for display in the CFC editor.

Example: FIELD|2|.

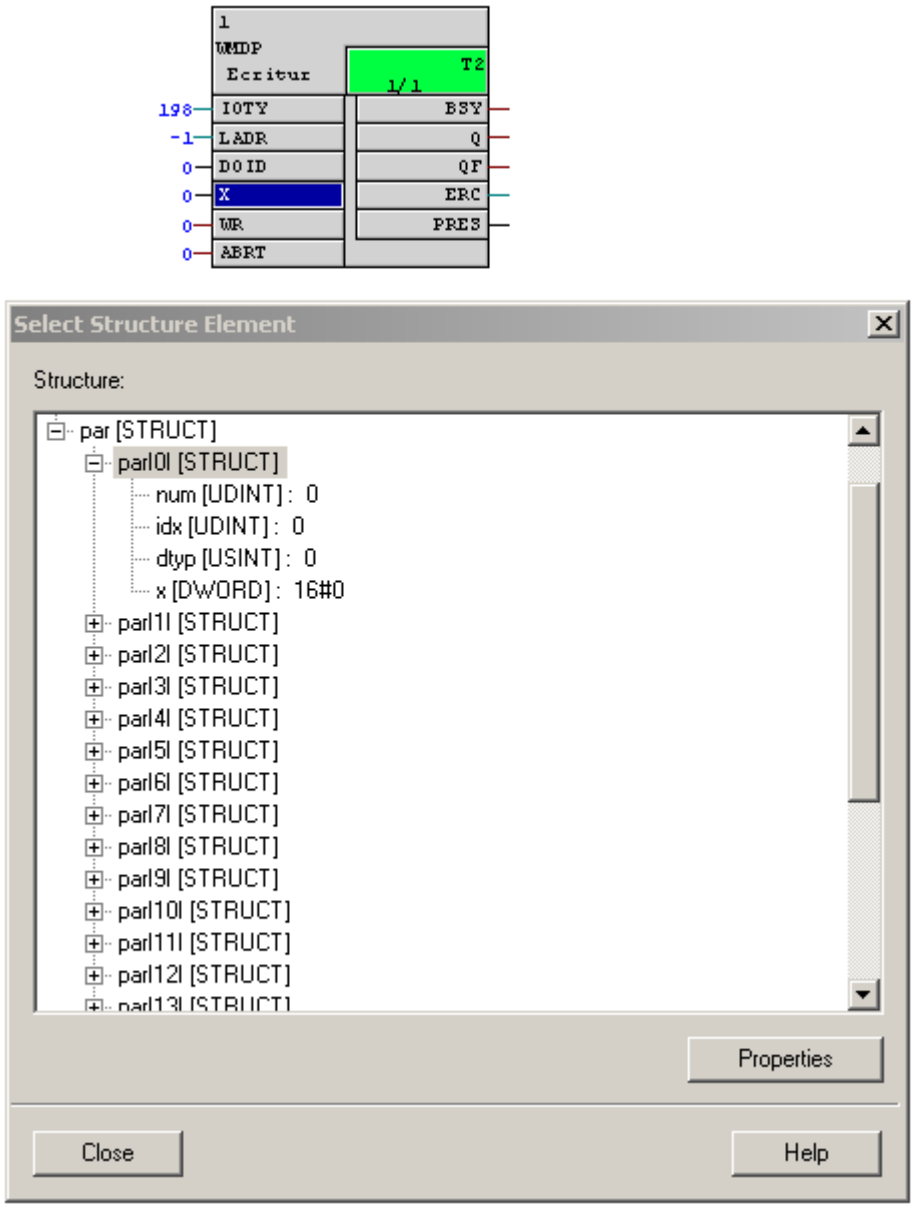


Figure 7-678 Assigning parameters for structured block connections

Double-clicking a structure element or clicking the **Properties** button opens a dialog box that is similar to the one for a block connection of an elementary data type.

Interconnecting structured block connections

Structured block connections are interconnected with the following in the editor:

- Connections of the same type from other CFC blocks
- Global variables of the same type
- TO variables of the same type
- IO arrays of the same length

Structured connections can only be interconnected as an entire unit.

It is not possible to interconnect individual structure elements.

You can monitor values using the dynamic display.

Structures can be interconnected when the name of the data type and the basic data type of the individual elements match. The interconnection rules of the DCC apply: DINT and DWORD are considered to be the same type.

Arrays can be interconnected when the basic data type and the dimension match.

With arrays of structures, the complete definition of the structure must be identical so that the connections can be interconnected.

Structures can only be monitored in the values table.

Interconnection of structured block connections to external variables

You can also interconnect structured block connections to external variables, e.g. a global variable of an ST block. However, as only structures from a DCBLib are valid, this library must be imported within the ST source file and the structure type from the DCBLib used for the global variable. The use of an identical reproduction within ST is not possible for an interconnection.

Example of an ST source file for the interconnection of input X of the RMDP DCC block

```
INTERFACE
    USEPACKAGE dcblib_simotion_4_3;
    VAR_GLOBAL
        x: DCB_RMDP_PAR;
    END_VAR
END_INTERFACE
```

An input or output parameter of a block (in this case RMDP) with the appropriate structure type can then be interconnected to the global ST variable via **Interconnection to operand...**. For subcharts, the input of the block must already have been interconnected previously to an input or output parameter of the subchart.

Publishing structured block connections

You can publish a structure element by entering a comment beginning with an @ sign at the structure element.

You can publish a block connection by entering a comment beginning with an @ sign at the block connection (top level). A structured variable is then created. The first identifier in the comment determines the variable identifier.

Note

DCB blocks with user-defined structures may only be edited as of SIMOTION 4.2.

Libraries with structures may not be generated for older SIMOTION versions.

Block types with user-defined structures may not be used in DCC SINAMICS.

Default connection values for delta downloads

Changes made to the default values of the block connections at a later date are only transferred to SIMOTION if the option highlighted in the following image is activated:

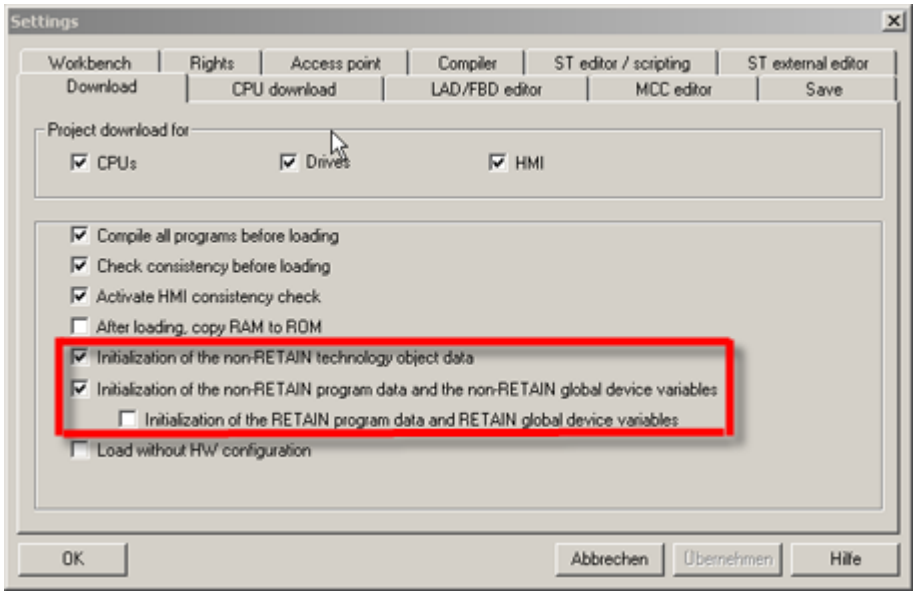


Figure 7-679 Settings for SIMOTION data transfer

Compiling the DCC in the DCC editor

Compiling

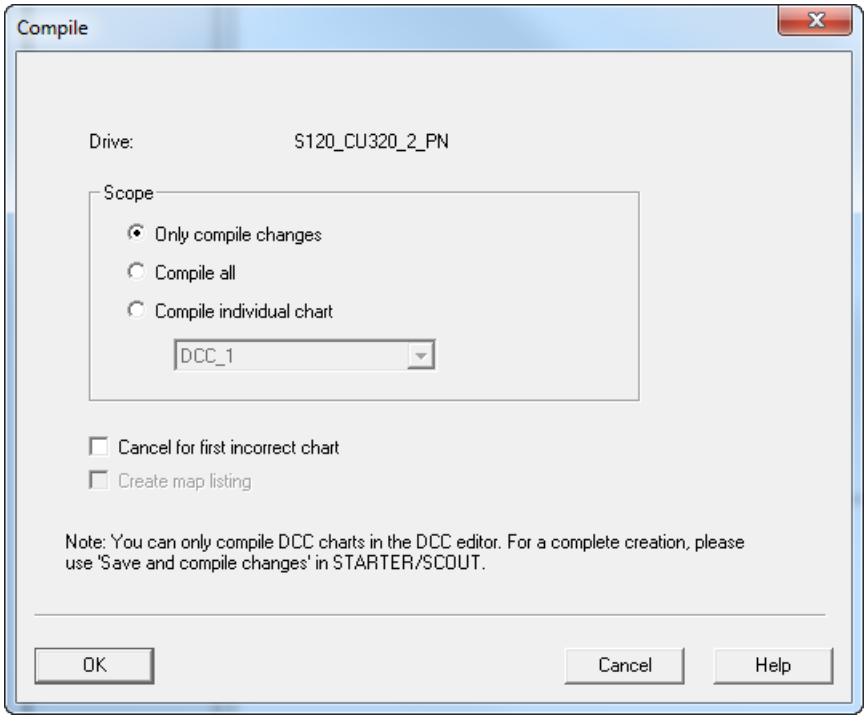


Figure 7-680 Compiling a dialog in the DCC editor

You can begin compiling with **Chart > Compile > Charts as program...** or via the button.

If errors occur during compilation, the **Logs** dialog box will automatically be displayed at the end of the procedure (just as in the case of the consistency check).

Note

You can check whether all the required block libraries and technology packages have been activated via the menu item **Options -> Block types**.

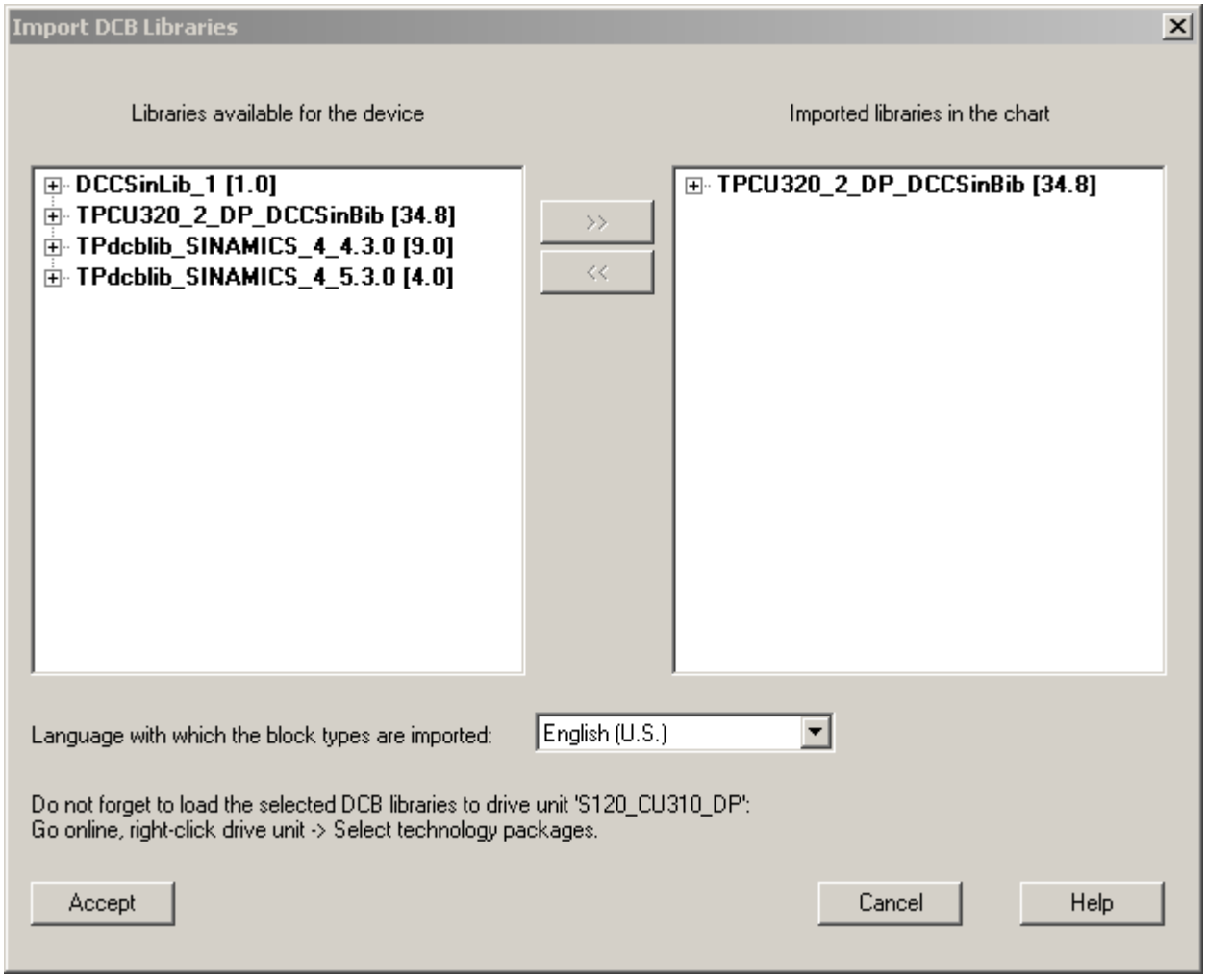


Figure 7-681 Updating the block library dialog

Compilation options

For detailed information about the compilation options, refer to Compiling (Page 5450).

Note

The **Reorganize HMI variable interface** option enables you to reassign the addresses for all @ variables currently defined in the chart. @ variables that are still available in the interface, but are no longer used in the charts, are deleted.

After compilation

On completion of the compilation procedure, a detailed compilation log appears.

- To access the relevant block that is causing an error, select the error line in the log and click **Go to**.
- The log can be called again at a later point via the menu **Options > Logs** and, if necessary, can also be printed out.

Loading the compiled DCC

To be able to operate the DCC program on a CPU, it must have first been downloaded to the CPU.

Downloading can only take place in SIMOTION SCOUT.

Note

The DCC editor assigns the configured DCBs to tasks T1 to T5 in the execution system.

Starting the CPU

To execute the DCC program, the CPU must be switched to RUN.

Enable attribute, execution groups

The "Enable" attribute enables or disables an execution group (enable = 1; disable = 0).

The "Enable" attribute is set to 1 by default. However, it can also be set dynamically. The output value of a block determines whether the group is to be enabled or disabled. To this end, you can interconnect the digital output of a block with the execution group.

Creating documentation

Complete documentation

The example has been successfully configured and tested. You can now create the complete documentation for your example.

Chart reference data

In this case, the chart reference data is the **block type cross references** and the **execution sequence**. In the toolbar, click: 

You can open the **Chart Ref: Display Chart Reference Data** window via **Options > Reference data** in the menu. In this window, you can use the buttons in the toolbar to create and display the relevant lists. To begin printing, click: 

Then close the dialog box with **Reference data > Exit**.

Note

The **Chart Ref: Display Chart Reference Data** window can only be opened when the window of the execution editor is not open.

Printing a chart

You also want to print the chart. Since only one page has been used on this chart, a single page display is recommended for printing. The chart is now set to **Page view**. To begin printing, click: 

If your chart consists of several pages, we recommend that you print them individually in the page view. Click the  button to change to the page view.

Click  or  to customize the display so that all DCBs are shown on a single page on the PC monitor.

See also

Compiling the DCC in the DCC editor (Page 5606)

7.4.5.3 DCC SIMOTION specifications

Rules for assigning names in the DCC editor

Variables are used for data exchange between SCOUT and the DCC editor. The variable names used are subject to the same restrictions regarding names in SIMOTION.

Note

In the DCC editor, the length of the name is restricted to **a maximum of 49 characters**. In SIMOTION, however, longer names are supported. Where this arises, it is necessary to either use aliases or interpose ST variables.

The names in the DCC editor must therefore abide by the following rules:

- Basic chart:
  - No keyword or previously defined name permitted
  - Must start with a letter
  - May contain numbers, letters, and underscores
  - "_" must be followed by a number
- Subchart:
  - Must start with a letter
  - May contain numbers and letters
  - Underscores are not permitted
  - Keyword or previously defined name permitted (during compilation, a check is performed to determine whether this is unique)
  - During compilation in the DCC editor, the name is checked to determine whether it is unique
- Block instance:
  - Can start with a number
  - May contain numbers and letters
  - May not start with an underscore
  - Keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined
- Execution group:
  - Keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined
- Chart connection:
  - No keyword or previously defined name permitted
  - In the DCC editor, a check is performed to determine whether the name has previously been defined

Field/name lengths and conventions

Field/name lengths and conventions

Object	Length	Remark
Chart name	22 *)	May not contain the following characters: \ . : / * ? " < > | # % ( ) Use of the "_" character is subject to particular specifications.
Chart comment	255	All ANSI characters are permitted.
Execution group	22	Only letters, numbers and underscores may be used.
Block type	6	Determined by DCC.
Block instance comment	80	All ANSI characters are permitted.
Parameter comment	80	All ANSI characters are permitted.
Block name (instance name)	16 *)	May not contain the following characters: \ . : / * ? " < > | # % Use of the "_" character is subject to particular specifications.
Global operand	Max. 49	If names from SIMOTION are too long, an ALIAS must be defined.

*) The chart name and block name must jointly consist of a maximum of 24 characters, including separating characters.

See also

Rules for assigning names in the DCC editor (Page 5609)

Representation of the dynamic value display

The values are output next to the connections, according to their data type. They are displayed on the screen with a colored background.

Table 7-616 Representation of the dynamic value display

Representation	Meaning
Blue on white	Representation of the values in edit mode (offline)
Black asterisks on yellow	Values during transfer to the dynamic display
Black value on yellow	Representation of the values read from the SIMOTION RT in test mode
#### on a red background	Representation of values while the dynamized values required from the SIMOTION RT are missing (fault, overload)

Displaying operands

When integrating OpenCFC into SIMOTION DCC, only symbolic operand names rather than absolute operand names may be used. The settings for displaying operands in the CFC Editor under **Options -> Settings -> Display ...** are ignored.

7.4.5.4 Faults and warnings

Notes on error message display

Notes on error message display

If an error message or warning appears in the DCC, there will be special characters that draw your attention to names or parts of names. You can double-click these characters to proceed directly to the relevant site of the error.

The characters have the following meanings:

- **Block instance or block input/block output**
  - Represented as:
    "\." <instance name> ["." <pin name>]
  - Example:
    Interconnection of "_device\ST_1.i" with "._DCC_6_CFC1_1.x_1" cannot be achieved; invalid type
- **Interconnection to chart connections**
  - Represented as:
    "<" <variable name> ">"
  - Example:
    The input connection DCC_5\<interconn_add_1_x_1> from the chart 'DCC_5' is not interconnected with a sink.
- **Execution level/execution group**
  - Represented as:
    "{" <group name> "}"
  - Example:
    Empty execution group {EG}

Note

Error analysis when a DCB block crashes

If a DCB block crashes during operation, information about the cause and the block instance can be found in the device diagnostics buffer in SIMOTION. With this information, you can contact the creator of the block library, remove the relevant block instance or change the parameterization.

7.4.6 Appendix

7.4.6.1 List of abbreviations

Abbreviation	Description
(G)UI	(**G**rapical) **U**ser **I**nterface
Connection	General term for block input or block output
BI parameter	**B**inector **I**nput parameter. The parameter is used for the interconnection of a binector to a sink signal that can only have the states 0 or 1
BICO	**Bi**nector-**Co**nnector that designates an interconnectable parameter in the drive
BO parameter	Binector parameter (also **B**inector **O**utput parameter). The parameter can be used as a binary signal source (0 or 1)
CFC	**D**rive **C**ontrol **C**hart
CI parameter	**C**onnector **I**nput parameter. The parameter is used for the interconnection of a connector to a sink signal
CO parameter	Connector parameter (also **C**onnector **O**utput parameter). The parameter can be used as a signal source
CSV	**C**omma **S**eparated **V**alue, text format for column-oriented data
DCB	**D**rive **C**ontrol **B**lock
DCC	**D**rive **C**ontrol **C**hart
DO	**D**rive **O**bject
ELF file	File coded in **E**xecutable **L**inkable **F**ormat
FEAT	**Feat**ure document
ITCP	**I**nstance, **T**ime slice, **C**onnection and **P**arameter/Variable
MBCS	**M**ulti-**B**yte **C**haracter **S**et, corresponds to UTF-8
MDI	**M**ultiple-**D**ocument **I**nterface – an application with several windows
OEM	**O**riginal **E**quipment **M**anufacturer
TP	**T**echnology **P**ackage

7.4.6.2 Glossary

Execution level	Interface to the execution system used for the execution of blocks.
Execution group	Execution groups are used for structuring or subdividing execution levels. The blocks and/or charts are integrated sequentially into the execution groups. SIMOTION only: Execution groups, for example, can be switched on and off separately through a connection to a block output (data type BOOL). When an execution group is switched off, all blocks/charts contained therein are no longer calculated.
Execution sequence	Sequence in which the blocks should be calculated in an execution level or execution group

User library	A user library which contains block libraries. As the blocks are based on charts, they can be handled individually in this case.
Output connections	The output connections can be interconnected to other inputs or assigned an initialization value. The value is then active at this connection when the block is calculated in the INIT mode for the first time. This is useful, for example, when a specific value is to be assigned to the output connection of a flip-flop block or a controller block.
HMI variables	You can declare block inputs and outputs as **HMI variables** and therefore generate a static interface for these for use in your system visualization.
Basic library	A library that has been created in C, e.g. with the aid of a DCB tool. Such a library is a closed unit, i.e. a DCB cannot, for example, be separated from a library in order to transfer it to another block. The supplied DCBLIB is an example.
Basic chart	The chart that is visible and can be managed in STEP 7 or SCOUT/STARTER. All other charts, i.e. chart partitions or sub-charts can only be managed in the DCC editor. Only the term chart is used in the following.
Block help	Detailed information about a certain block type can be obtained by selecting the block and pressing the F1 key.
Block type library	Consists of the description of the block types contained in the library and the object file that implements the blocks, e.g. ELF.
INIT mode	Initialization, ramp-up
RUN mode	Cyclic operation
DCB library	A user-defined DCB is created from a chart.
DCB studio	Development environment for the programming of C block libraries.
DCB viewer	You can read the descriptions of the individual block types in the DCB viewer during the configuration in the DCC editor. The DCB viewer is opened by selecting a block and pressing the F1 key.
Input connections	The input connections can be parameterized with constants by the project engineer or interconnected to other block outputs. When the DCB blocks are called, the inputs and outputs have already been assigned default values that can be changed when required.
Creation mode	DCCs can be edited in creation mode.
Freely defined parameters	@ parameters and @ technology connectors. Interface defined for the parameterization by the user in the form of parameter numbers that can be defined as ALIAS for block inputs and block outputs.
Global operands	Global operands are connection partners for block inputs/outputs with which DCB blocks can read and write information from the system environment.

Hierarchical chart or also subchart	A chart (subchart) can be inserted in another chart (chart-in-chart-technique). Hierarchical structures can be formed here. Each inserted chart can be opened and, like every other chart, edited further and therefore changed individually. A chart can be encapsulated for further use, i.e. chart connections added. Which block connections are provided at the chart connections can also be specified individually.
Comments	Each block connection of the CFC page can have a comment added.
Connector	If no more lines can be drawn because the page is too full, the CFC inserts a connector at the block/chart connection and a number in the sheet bar. The corresponding connectors are assigned the same reference number. If several interconnections that cannot be displayed all start at one output, they are all assigned the same reference number. You can recognize the connection point location through the varying representation of the connector. For details, refer to the CFC help.
Laboratory mode	The blocks are automatically monitored in laboratory mode (display of the current values of the block/chart connections logged on for testing).
Offline	Project is processed without a connection to the device.
Online	Project is processed with a connection to the device.
Online RUN, or pulse enable at the drive	Project is processed with a connection to the device and the device is in open-loop or closed-loop control.
Parameterization	Instead of an interconnection, a constant different than the default can be parameterized at each input or output.
Process mode	The blocks are not automatically monitored in process mode (display of the current values of the block/chart connections logged on for testing) so that the additional load is very small.
Sheet bars	The sheet bars on the left and right sides of a CFC page contain the references to the interconnected objects, e.g. other blocks or execution groups, that are not on the current page. They also contain the number of the connector (breakpoint) when the connection line to the sheet bar cannot be drawn because the page is too full.
Technology package	A technology package (the DCBLIB) contains technology objects (DCB) that are capable of being instantiated.
Chart partition	Each chart comprises up to 26 chart partitions ("A".."Z"). Each chart partition consists of up to six DIN A4 pages. A newly created chart only contains chart partition "A".
Test mode	Test mode can be used to debug the DCCs.
Text interconnections	Are used to split projects into separate, configurable units and define an "open" interconnection between charts. A text interconnection can only be at a block/chart input and always references a block or chart output in CFC. The text interconnection is an "open" interconnection until it becomes a "real" interconnection upon closing. A closable text interconnection is the addressing of an input by means of a character string, which identifies a specific interconnection source (output) via the path specification (chart/block.connection).

Trace	The signals of the block outputs can be recorded with the trace function.
Trend display	In test mode with SIMOTION, you can use the value and trend display to analyze the input and output values of blocks.
Typical	DCB library
Overflow pages	Overflow pages are created automatically when more sheet bar entries are generated than can be displayed on a page. An overflow page consists solely of the sheet bars and contains no further objects.
Interconnecting	The term interconnecting means: The connection of a block output to another DCB block input on the same device Interconnections of a block output to an execution group. SIMOTION only Interconnection of a block output to a global operand or a global operand to a DCB block input. A global operand can be the following: The name of a SIMOTION variable. The name of a SINAMICS parameter (can also be performed implicitly through the interconnection to a SB).
Workbench	The workbench is the navigation center for the individual engineering steps and is used for the centralized creation and management of projects. It provides a uniform, function-oriented, consistent, filterable view of all data and programs, even in distributed systems.

Programming - References

8

8.1 Description of the DCC standard blocks

Preface

SIMOTION documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.3.1:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

SINAMICS documentation

The SINAMICS documentation is organized into two parts:

- General documentation/catalogs
- Manufacturer/service documentation

An overview of the current documentation in the available languages can be found on the Internet:

http://www.siemens.com/motioncontrol

Select the menu items "Support" --> "Technical Documentation" --> "Overview of Publications."

Information on the range of training courses and FAQs (Frequently Asked Questions) is available on the Internet:

http://www.siemens.com/motioncontrol

Select the menu item "Support".

Further documentation for the DCC editor

- SINAMICS/SIMOTION DCC Editor Description

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (finding and searching in manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and on the Service&Support pages under **Product Support**:

http://support.automation.siemens.com

Technical Support

Country-specific telephone numbers for technical support are available on the internet under **Contact**:

http://www.siemens.com/automation/service&support

Compliance with the General Data Protection Regulation

Siemens respects the principles of data privacy, in particular the data minimization rules (privacy by design).

For the product SINAMICS DCC this means:

The product stores personal data only in the project. The user is therefore responsible for ensuring compliance with the statutory data protection provisions. This applies in particular to the transfer of projects. If the user links this data with other data (e.g. shift plans) or if he/she stores person-related data on the same data medium (e.g. hard disk), thus personalizing this data, he/she has to ensure compliance with the applicable data protection stipulations.

The following data must be taken into account.

- Windows login
  In the standard configuration, the product saves the login details of the Windows user together with technical function data (e.g. time stamp) in the project. The specified data is saved in order to trace changes in large configurations.
  With SINAMICS DCC, a personal reference can be established via the project and all elements contained in it (e.g. charts).
  The specified data can be viewed in the properties of the project and the elements in SINAMICS DCC (property "Author") and subsequently changed, with the exception of the last modification of the project.

For the above-mentioned point, the details of the functions mentioned in the respective chapter in the information system of SINAMICS DCC must be observed.

By generating the login or username, personal data can be pseudonymized for the functions. Deleting the project will cause all personal data saved within it to be deleted too.

8.1.1 Safety instructions and industrial security

8.1.1.1 Fundamental safety instructions

General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. Application examples do not represent specific customer solutions, but are only intended to provide support for typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit:

Industrial security (http://www.siemens.com/industrialsecurity)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at:

Industrial security (http://www.siemens.com/industrialsecurity)

Further information is provided on the Internet:

Industrial Security Configuration Manual (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ WARNING

Unsafe operating states resulting from software manipulation

Software manipulations (e.g. viruses, trojans, malware or worms) can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
- Make sure that you include all installed products into the holistic industrial security concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- Protect the drive against unauthorized changes by activating the "know-how protection" drive function.

8.1.1.2 Use write and know-how protection

Prevent unauthorized changes by means of know-how protection

⚠ WARNING

Danger to life through manipulation of DCC charts and DCC libraries

The use of unprotected DCCs and DCC libraries entails a higher risk of manipulation of DCCs, DCC libraries and backup files.

- Protect important DCC charts and DCC libraries by using **know-how protection programs** or via the **know-how protection for drive units** in the SCOUT/STARTER. You can prevent manipulation by assigning a strong password.
- For **know-how protection programs** and **the know-how protection of drive units**, use passwords which include at least eight characters, upper and lower cases, numbers and special characters.
- Make sure that only authorized personnel can access the passwords.
- Protect the backup files on your file system using a write protection.

8.1.1.3 Industrial Security Configuration Manual

Manual on industrial security

Further information can be found in the configuration manual "Industrial Security" for SINAMICS, SINUMERIK and SIMOTION at Security Manual (https://support.industry.siemens.com/cs/document/108862708/sinumerik-simotion-sinamics-industrial-security?dti=0&lc=en-DE).

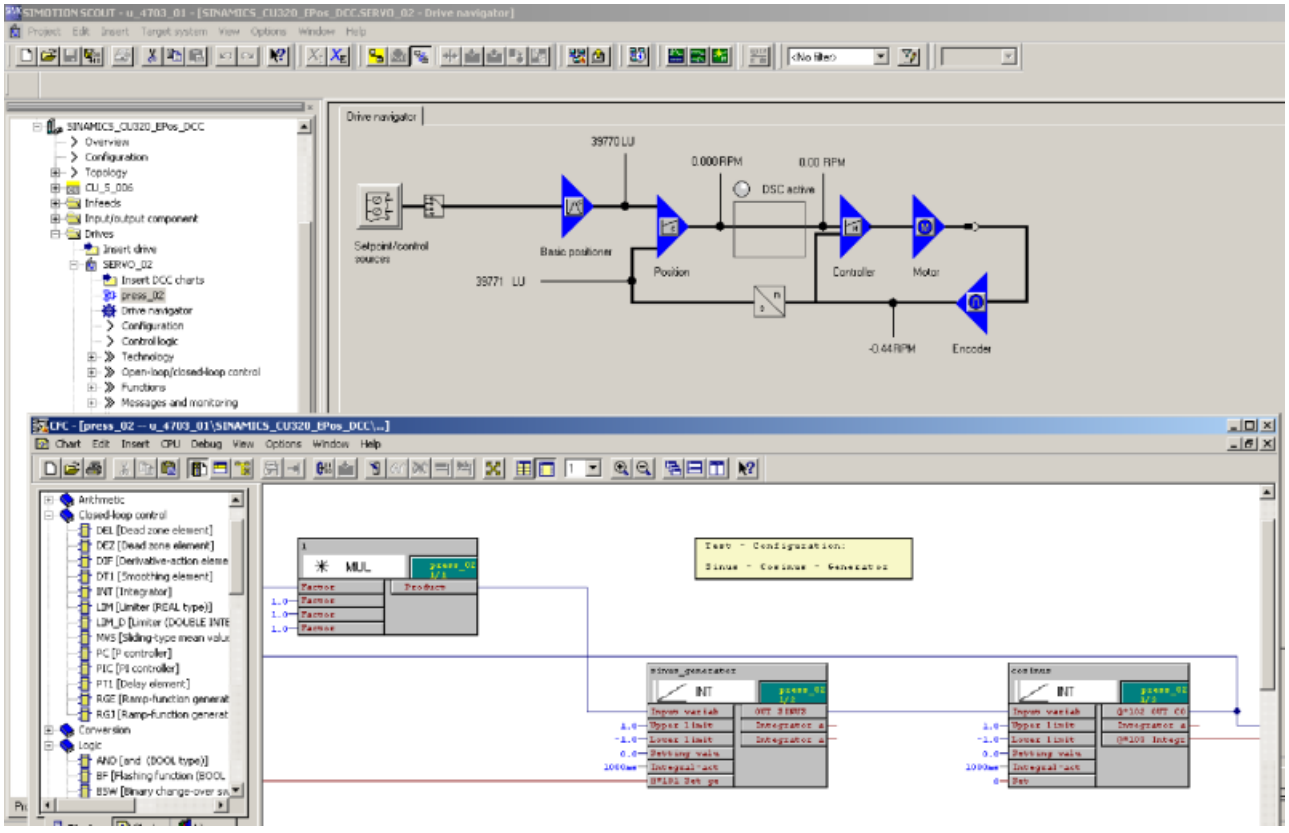
Note in particular the explanations on the cell protection concept in Section "General security measures - Network segmentation".

8.1.2 Introduction

8.1.2.1 Introduction to the Drive Control Chart (DCC)

Drive Control Chart (DCC) for SINAMICS and SIMOTION means graphic configuration and expansion of the device functionality by means of freely available control, calculation and logic blocks.

Drive Control Chart (DCC) expands the facility for the simplest possible configuring of technological functions both for the SIMOTION motion control system and the SINAMICS drive system. This opens up a new dimension for users for adapting the specified systems to the specific functions of their machines. DCC has no restriction with regard to the number of usable functions; this is only limited by the performance capability of the target platform.



The user-friendly DCC Editor enables easy graphics-based configuration, allows control loop structures to be clearly represented and provides a high degree of reusability of diagrams that have already been created.

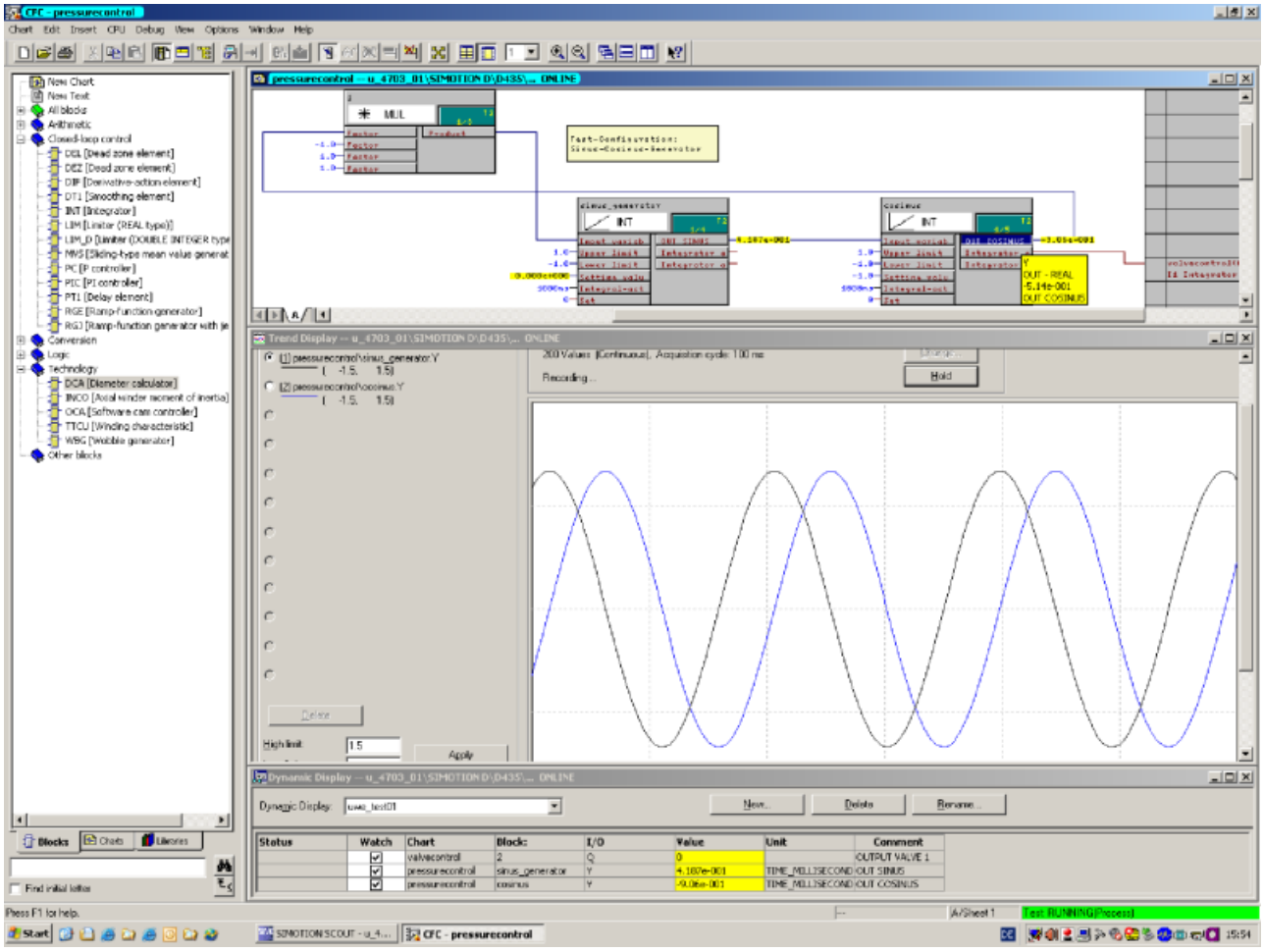
The open-loop and closed-loop control functionality is defined by using multi-instance-capable blocks (Drive Control Blocks - DCBs) from a predefined library (DCB library) that are selected and graphically linked by dragging and dropping. Test and diagnostic functions allow the program behavior to be verified and, in the case of a fault, the cause identified.

The block library contains a large selection of control, arithmetic and logic blocks as well as extensive open-loop and closed-loop control functions.

All commonly used logic functions are available for selection (AND, XOR, On/Off delay, RS flip-flop, counters, etc.) for the logic operation, evaluation and acquisition of binary signals. Numerous calculation functions, such as summation, division and minimum/maximum evaluation are available for monitoring and evaluating numeric variables. In addition to the drive control, axial winder functions, PI controllers, ramp-function generators or sweep generators can be configured simply and without problem.

Almost unlimited programming of control structures is possible in conjunction with the SIMOTION motion control system. These can then be combined with other program sections to form an overall program.

Drive Control Chart for SINAMICS drives also provides a convenient basis for resolving drive-related open-loop and closed-loop control tasks directly in the converter. This results in further adaptability of SINAMICS for the tasks set. Local data processing in the drive supports the implementation of modular machine concepts and results in an increase in the overall machine performance.



8.1.2.2 Libraries

Blocks are located in libraries that are imported as technology packages in the DCC editor.

There are two different libraries:

1. The SIMOTION library contains the SIMOTION blocks identified in this document.
2. The SINAMICS library contains the SINAMICS blocks identified in this document.

To find out which of the blocks described here are available within SIMOTION and/or SINAMICS, you can use both the overview in Appendix A3 and the sections on block descriptions.

Compatibility

SIMOTION

The compatibility of the standard library for the SIMOTION devices is described on the SIMOTION SCOUT DVD and at the following link on the Internet:

SIMOTION compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317)

SINAMICS

The following standard libraries are executable with SINAMICS V5.1:

- SINAMICS V5.1 (dcblibV3_0_sinamics5_1)
- SINAMICS V4.8 (dcblibV3_0_sinamics4_8)
- SINAMICS V4.7 (dcblibV3_0_sinamics4_7)
- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.8:

- SINAMICS V4.8 (dcblibV3_0_sinamics4_8)
- SINAMICS V4.7 (dcblibV3_0_sinamics4_7)
- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.7:

- SINAMICS V4.7 (dcblibV3_0_sinamics4_7)
- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

The following standard libraries are executable with SINAMICS V4.6:

- SINAMICS V4.6 (dcblibV3_0_sinamics4_6)
- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V4.5, the following standard libraries are executable:

- SINAMICS V4.5 (dcblibV3_0_sinamics4_5)
- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V4.4, the following standard libraries are executable:

- SINAMICS V4.4 (dcblibV3_0_sinamics4_4)
- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V4.3.x, the following standard libraries are executable:

- SINAMICS V4.3 (dcblibV2_0_sinamics4_3)

With SINAMICS V2.6.x, the following standard libraries are executable:

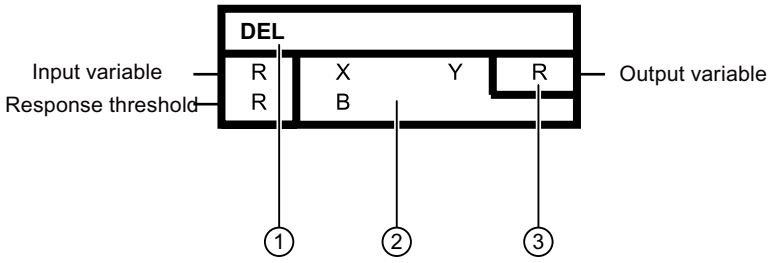
- SINAMICS V2.6 (dcblibV2_0_sinamics2_6)

With SINAMICS V2.5.SP1, the following standard libraries are executable:

- SINAMICS V2.5 (dcblibV2_0_sinamics2_5_1)

8.1.2.3 Nomenclature of the blocks

A block is displayed as follows:



① Block designator
② Connection designator
③ Connection data type

It is identified using the following attributes:

Block designator

Each data type has its own block type. To simplify differentiation between the blocks for various data types with the same functionality, these are provided with a postfix corresponding to the data type, whereby postfix is not usually used for the Real and Bool data types (e.g. MUL_I: Integer-type multiplier, MUL: Real-type multiplier). The following table lists commonly used extensions:

Table 8-1 Block designator

Postfix for block designator	Data type of the input/output variable
_I	Integer
_D	Double_Integer
_W	Word
_R	Real (optional)
_B	Bool (optional)
_SI	Short Integer
_M	Modulo
_BY	Byte
_UI	Unsigned Integer
_US	Unsigned Short Integer
_UD	Unsigned Double Integer

Postfix for block designator	Data type of the input/output variable
_DW	Double Word
_LR	Long Real

Connection designator

- To identify a field from the input or output variable, the designator is extended by an index beginning with 1 (e.g. X1, X2, X3, etc.).
- With a generic number of inputs (e.g. ADD), the connection name is indexed beginning with 1 (e.g. X1, X2, X3, etc.).

The following table shows an overview of common connection designators:

Table 8-2 Connection designator

Connection designator	Use
X, X1, X2...	Numeric input variable
Y, Y1, Y2...	Numeric output variable
I, I1, I2...	Binary input variable
Q, Q1, Q2...	Binary output variable
IS	Bit string input (Word)
QS	Bit string output (Word)

If further inputs and outputs are used along with the primary input and output variables (e.g. limit values, time data, substitute values, status displays), the designators from the pool of the primary input/output variables are not used. The following table shows the preferred designators for secondary variables:

Connection designator	Use
LU	Input: Upper limit
LL	Input: Lower limit
SV	Input: Setting value
S	Input: Setting the setting value
R	Input: Resetting the setting value
QU	Output: Upper limit reached
QL	Output: Lower limit reached
QF	Output: Fault flag
QE	Output Y equals input X
QN	Inverted binary variable

Connection data type

The abbreviated designators of the data types are listed in the following table:

Abbreviation	Bit width	Data type according to IEC 61131-3	Description
BO	1	BOOL	BOOLEAN
BY	8	BYTE	Bit string, Unsigned Integer

Abbreviation	Bit width	Data type according to IEC 61131-3	Description
SI	8	SINT	Signed Short Integer
DI	32	DINT	Signed Double Integer
DW	32	DWORD	Bit string, Unsigned Integer
I	16	INT	Signed Integer
R	32	REAL	Floating Point Single Precision acc. to IEEE 754
LR	64	LREAL	Floating Point Double Precision according to IEEE 754
T	32	SDTIME	Floating Point Single Precision according to IEEE754
W	16	WORD	Bit string, Unsigned Integer
AID	32	-	Alarm ID

8.1.2.4 Block connections

Block connections display the interface of the DCBs, via which interconnection between the blocks can be performed. A distinction is made between

- Block output
- Block input

here, and these have the following properties:

- Inputs are positioned on the left of the block and are the target of an interconnection
- Outputs are positioned on the right of the block and are the source of an interconnection

8.1.2.5 Byte ordering

When interconnecting the blocks, the byte ordering of the data does not have to be taken into consideration. During data type conversions and arithmetic operations, the byte ordering of the target system is implicitly taken into consideration. Any byte swapping required for handling data beyond the system boundaries is carried out by the system (e.g. byte swapping may have to be carried out in Big Endian format before transferring data via PROFIBUS).

8.1.2.6 Direct interconnection of different data types

When interconnecting blocks, the target and source must be of the same data type. If the data types are different, there are special conversion blocks available which allow the data type to be converted.

The following permissible implicit conversions are an exception. The table below lists the permissible conversions.

The following data types, which can be interconnected without a conversion block, are another exception. In this case, the binary value of the output variable is transferred unchanged as the input variable.

Table 8-3 Conversions

Input	Output	Description
WORD	INT	Interconnection of a word variable to an integer variable
INT	WORD	Interconnection of an integer variable to a word variable
DWORD	DINT	Interconnection of a double word variable to a double integer variable
DINT	DWORD	Interconnection of a double integer variable to a double word variable
BYTE	SINT	Interconnection of a byte variable to a short integer variable
SINT	BYTE	Interconnection of a short integer variable to a byte variable
USINT	BYTE	Interconnection of an unsigned short integer variable to a byte variable
BYTE	USINT	Interconnection of a byte variable to an unsigned short integer variable
USINT	SINT	Interconnection of an unsigned short integer variable to a short integer variable
SINT	USINT	Interconnection of a short integer variable to an unsigned short integer variable
UINT	WORD	Interconnection of an unsigned integer variable to a word variable
WORD	UINT	Interconnection of a word variable to an unsigned integer variable
UINT	INT	Interconnection of an unsigned integer variable to an integer variable
INT	UINT	Interconnection of an integer variable to an unsigned integer variable
UDINT	DWORD	Interconnection of an unsigned double integer variable to a double word variable
DWORD	UDINT	Interconnection of a double word variable to an unsigned double integer variable
UDINT	DINT	Interconnection of an unsigned double integer variable to a double integer variable
DINT	UDINT	Interconnection of a double integer variable to an unsigned double integer variable
SDTIME	REAL	Interconnection of an SDTime variable to a real variable

8.1.2.7 Initialization of the blocks

Initialization determines the starting condition of the block. It is carried out by the system before the cyclical processing[1)] of the block. The sequence for initializing the individual blocks is implemented according to the configured priority and process sequence. At the time of initialization, the configured interconnections and constants for a block are already active. At this point, the values from the interconnection source are already available in a block. Should a block behave in a special way during initialization, this is described in the respective block description under "Initialization". In the case of initialization, the blocks must be assigned in a time slice (SINAMICS) or to a task (SIMOTION).

[1)] As of SIMOTION 4.1 SP2, initialization is performed during a STOP/RUN transition (SIMOTION) or during the transition to cyclical operation (SINAMICS).

8.1.2.8 Implementing complex functions in a sample configuration

Sample configurations are available for the "convenient ramp-function generator" and "technology controller". These are based on the "free blocks" of the SIMOVERT MASTERDRIVES frequency converter series.

The functionality of the convenient ramp-function generator is implemented by interconnecting individual DCBs. They are provided in the form of a sample configuration.

Notes relating to the technology controller

- The smoothing filters cannot be deactivated using the time constant T = 0, as the time constant is limited to the sampling time of the block. The filters must be deactivated explicitly by a signal. The corresponding binary input must be provided in the sample configuration.
- The D component cannot be deactivated using the delay time Tv = 0. This must be deactivated explicitly by a binary signal. The corresponding binary input must be provided in the sample configuration.
- The I component cannot be deactivated using Tn = 0. For this, the I component of PIC must be reset explicitly using SV = 0 and S = 1.

Note

Here, you should now also use the advantage of DCC and configure/transfer only the blocks that are required, i.e. not fall back on the sample configurations from the outset and start with the basic blocks, such as RGJ (ramp-function generator with rounding) or PIC (PI controller) and expand, if necessary.

Import sample configuration

The sample configuration is archived in the form of exported DCC charts as of SCOUT/STARTER version V4.3.

SIMOTION

In order to import the DCC chart in SCOUT, a SIMOTION device needs to be present in the project. In the Programs folder, select the path of the exported DCC chart via the shortcut menu Expert -> Import object. The files can usually be found under the following path:

C:\Program Files\Siemens\Step7\Examples\dcc\SIMOTION

SINAMICS

To import the DCC chart in SCOUT/STARTER, an S120 drive unit with a drive object (DO) must be present in the project. On the drive object, select the path of the exported DCC chart via the shortcut menu Expert -> Import object. The files can usually be found under the following path:

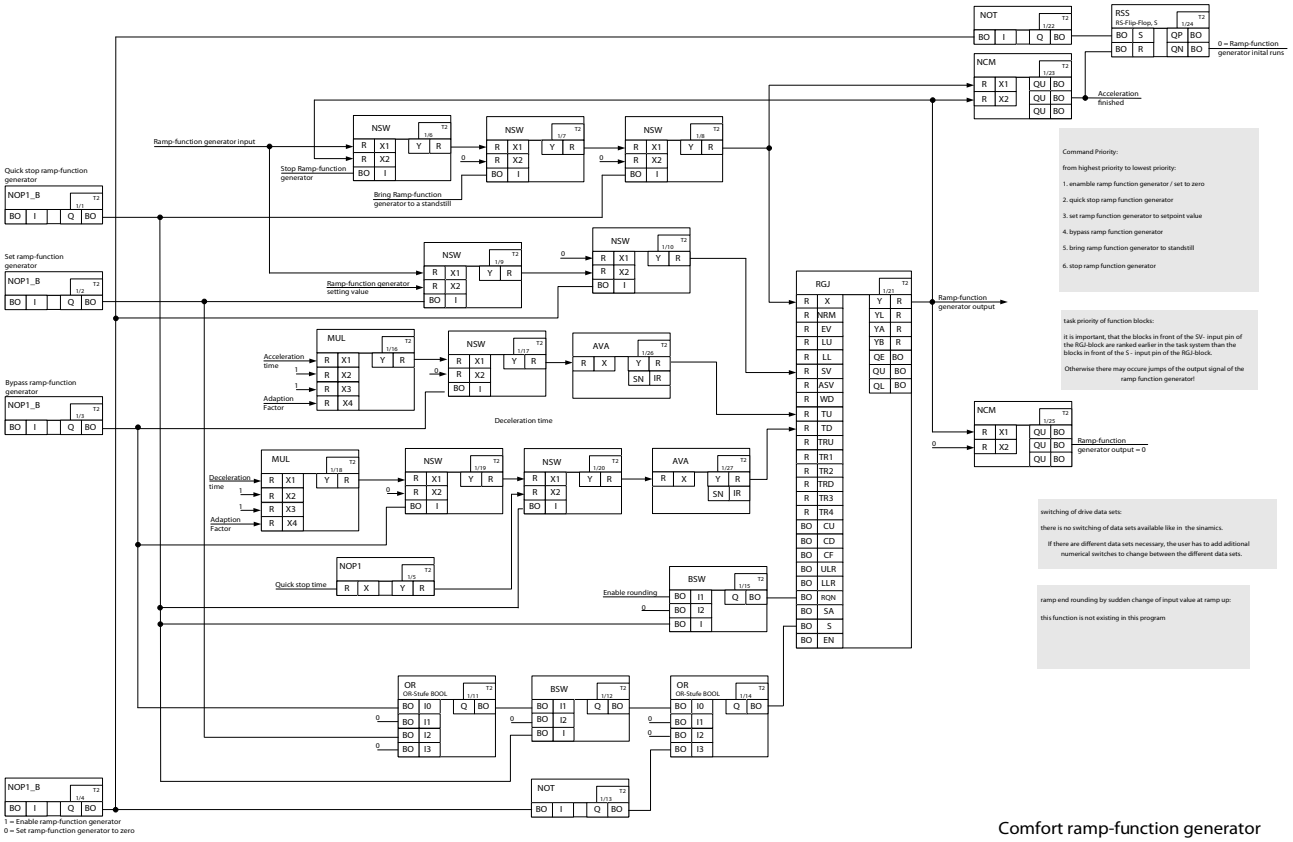
C:\Program Files\Siemens\Step7\Examples\dcc\SINAMICS

Up to SCOUT/STARTER version V4.3, the sample configuration was archived as a project export under the following path:

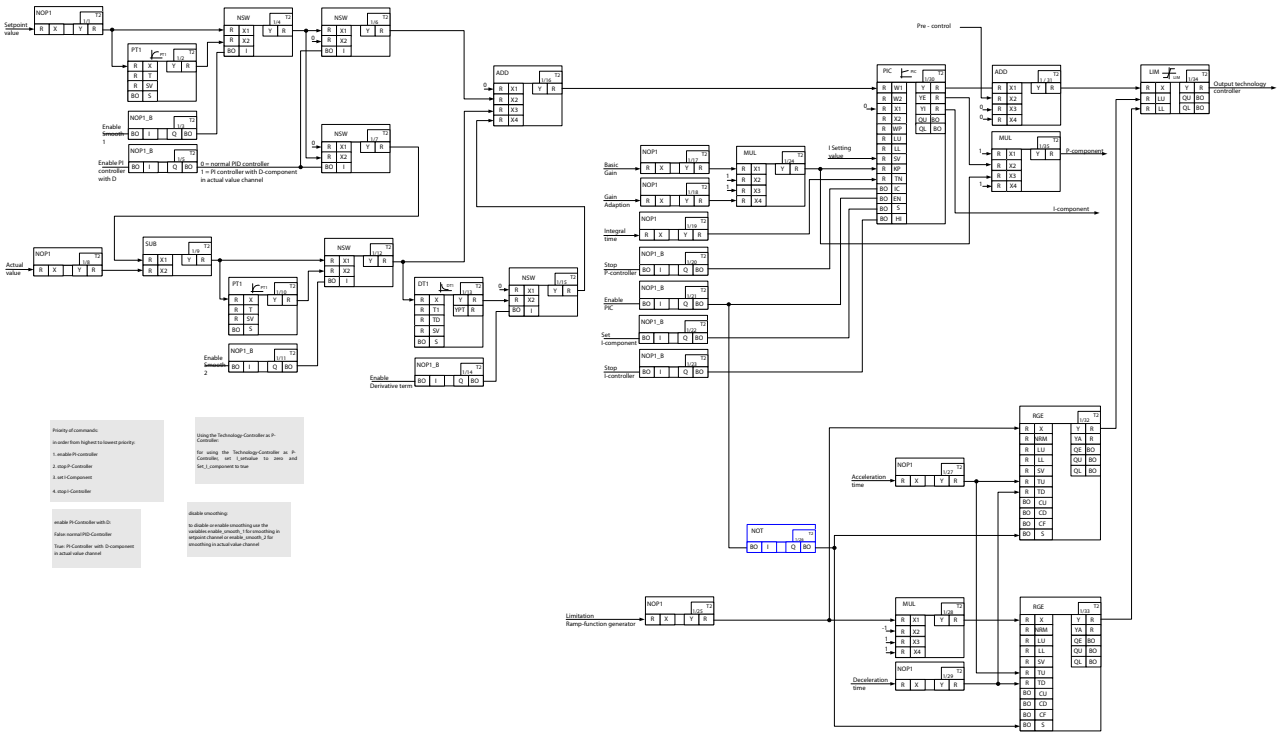
C:\Program Files\Siemens\Step7\Examples\dcc\Examples_CRGE_TCLR.xml

Note

The sample project has been created on the basis of a SCOUT project and thus contains both the configurations for SINAMICS_Integrated and SINAMICS standalone / CU320. When importing the sample project with STARTER (stand-alone), the SIMOTION components are naturally rejected; the CU320 components, however, continue to be accurately imported and are reproducible.



Comfort ramp-function generator



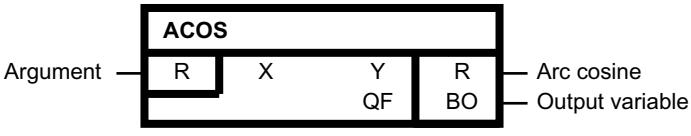
Technology Controller

8.1.3 Arithmetic

8.1.3.1 ACOS

Arc cosine function

Symbol



Brief description

Determination of the arc cosine value for an argument

Method of operation

The block determines the associated arc cosine value in radian measure for an argument to be entered at input X and outputs the result at output Y.

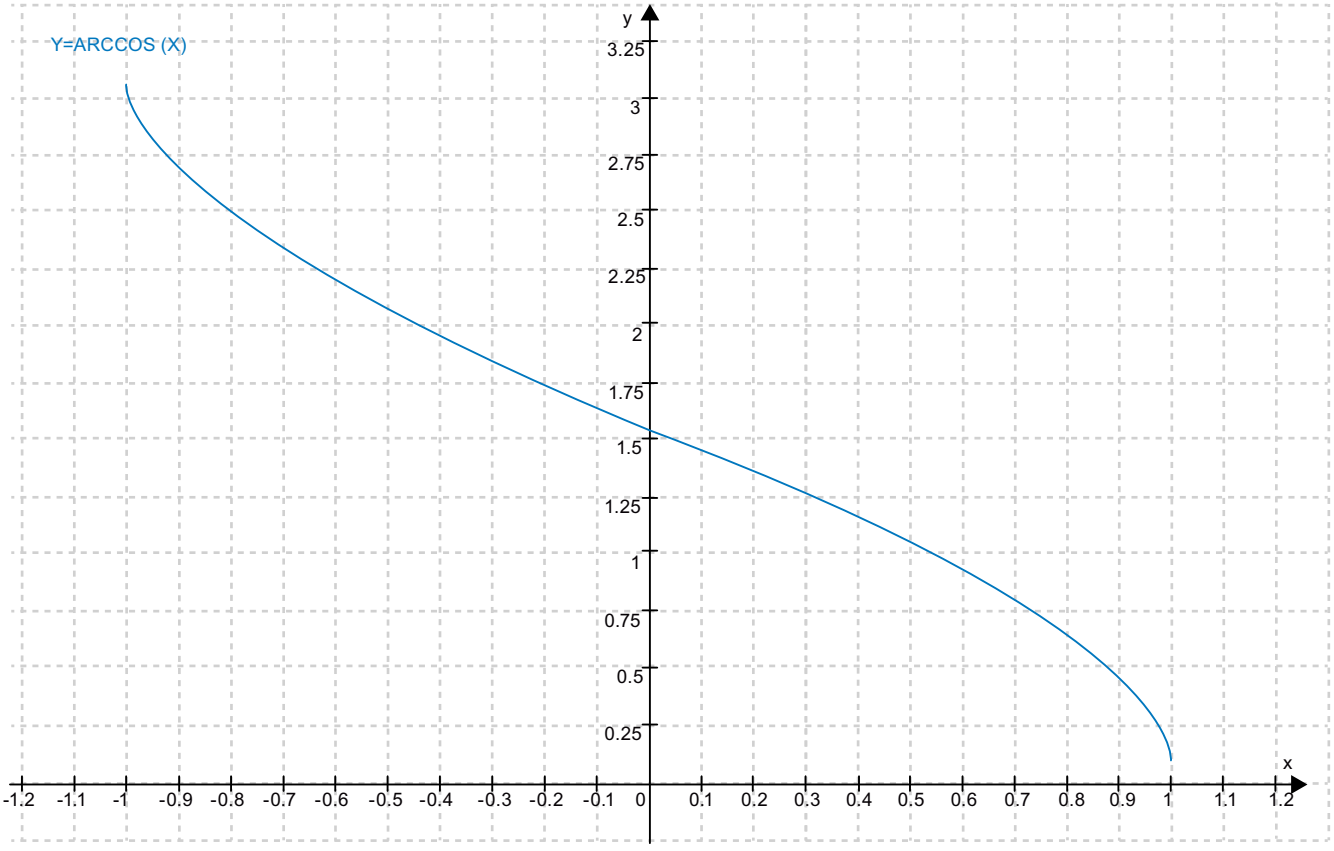
Y = arccos X

Permitted input range: -1.0 <= X <= +1.0

Output range: 0.0 <= Y <= ∏

If the argument lies outside of the permitted input range, output Y is limited to ∏ (when X < -1.0) or 0.0 (when X > +1.0) and binary output QF = 1 is set at the same time.

Transfer function



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Argument	0.0	REAL	
Y	Arc cosine	∏/2	REAL	
QF	Output variable	0	0/1	

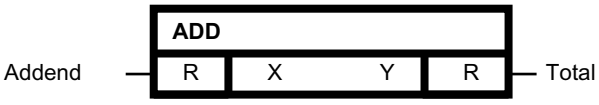
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.3.2 ADD

Adder (REAL type)

Symbol



Brief description

Adder with up to four inputs of the REAL type

Method of operation

The block adds the values entered at the X inputs, taking account of the sign. The result, limited to the range of -3.402823 E38 to 3.402823 E38, is output at output Y.

Algorithm:

Y = X1 + X2 + X3 + X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Addend	0.0	REAL	
Y	Total	0.0	REAL	

Configuration data

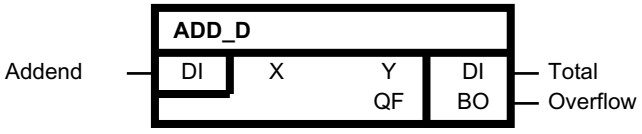
SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes

Can be configured in	Cyclic tasks
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.3 ADD_D

Adder (double integer type)

Symbol



Brief description

Adder with up to four inputs of the double integer type

Method of operation

The block adds the values entered at the X inputs, taking account of the sign. The result, limited to a range of approximately -2147483648 (2^{31}) to +2147483647 (2^{31}-1), is output at output Y.

Algorithm:

Y = X1 + X2 + X3 + X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Addend	0	DINT	
Y	Total	0	DINT	
QF	Overflow	0	0/1	

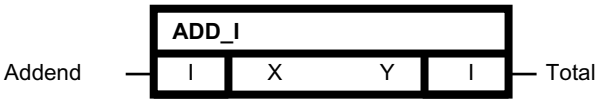
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.4 ADD_I

Adder (integer type)

Symbol



Brief description

Adder with up to four inputs of the integer type

Method of operation

The block adds the values entered at the X inputs, taking account of the sign. The result, limited to the range of -32768 to +32767, is output at output Y.

Algorithm:

Y = X1 + X2 + X3 + X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Addend	0	INT	
Y	Total	0	INT	

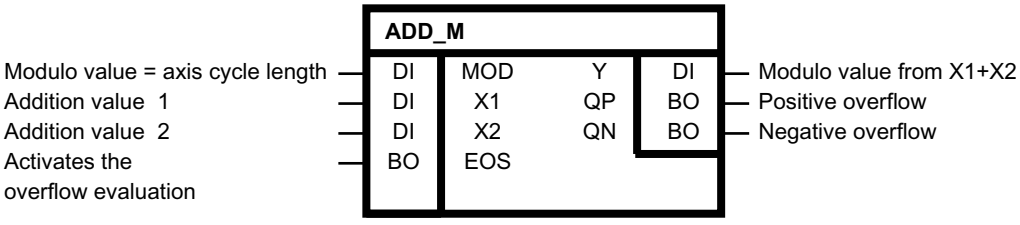
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.5 ADD_M

Modulo adder for addition in correct axis cycle

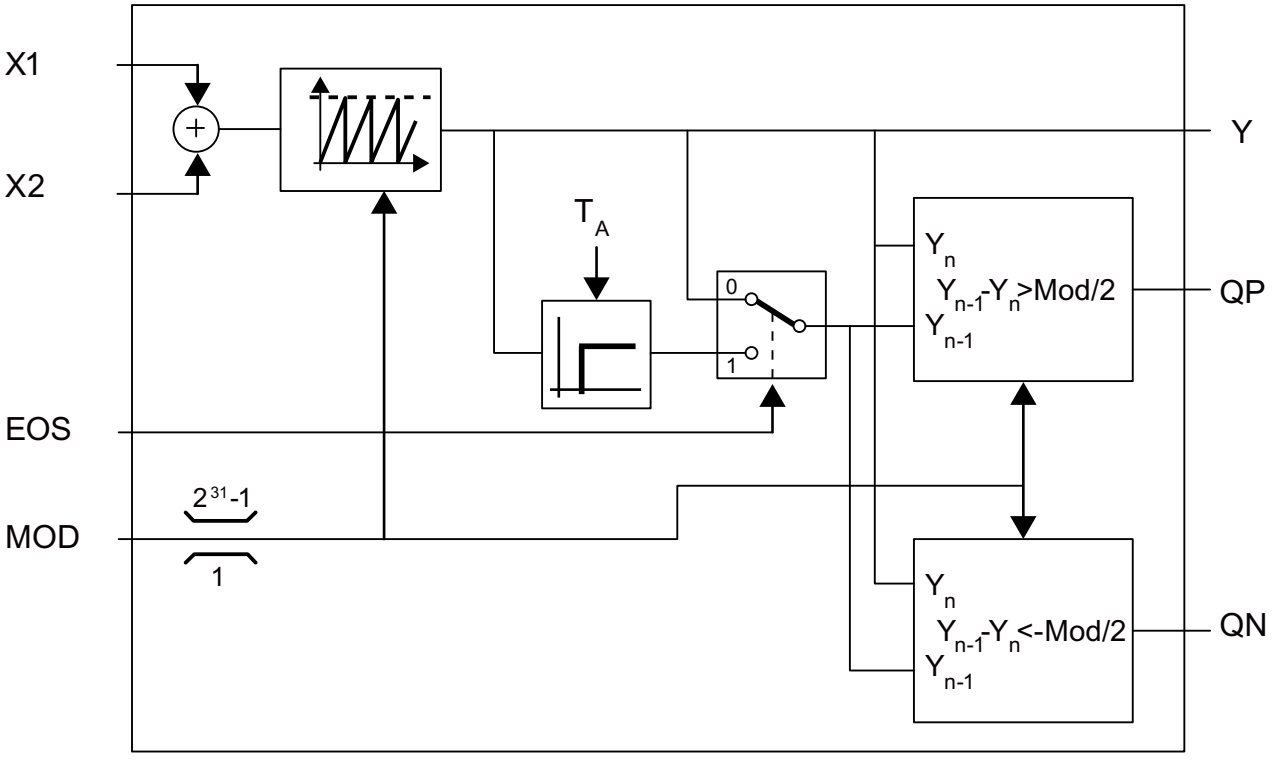
Symbol



Brief description

The ADD_M block is used to add position values. It can be used to "add up" offsets for position setpoints or for dead time compensation in the real master.

Block diagram



Method of operation

This block adds the input values X1 and X2. A modulo value can be specified at the MOD input. The module value must be in the range of 1 to 2^{31}-1 and is applied to the sum of X1 and X2. Thus, the result Y of the modulo operation is always in the band from 0 to MOD.

The EOS input can be used to activate an overflow evaluation. When EOS = 1:

Positive overflow: QP = Y_{n-1}-Y_n> MOD/2

Negative overflow: QN = $Y_{n-1} - Y_n < -MOD/2$

When EOS = 0: QP = 0; QN = 0

This enables the overflow evaluation to be deactivated when setting offsets. When the modulo value is changed, the overflow evaluation is switched off for one cycle.

Block connections

Block connection	Description	Default	Value range	Attributes
MOD	Modulo value = axis cycle length	1	DINT	
X1	Addition value 1	0	DINT	
X2	Addition value 2	0	DINT	
EOS	Activates the overflow evaluation	0	0/1	
Y	Modulo value from X1+X2	0	DINT	
QP	Positive overflow	0	0/1	
QN	Negative overflow	0	0/1	

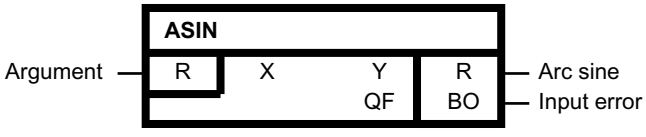
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.3.6 ASIN

Arc sine function

Symbol



Brief description

Determination of the arc sine value for an argument

Method of operation

The block determines the associated arc sine value in radian measure for an argument to be entered at input X and outputs the result at output Y.

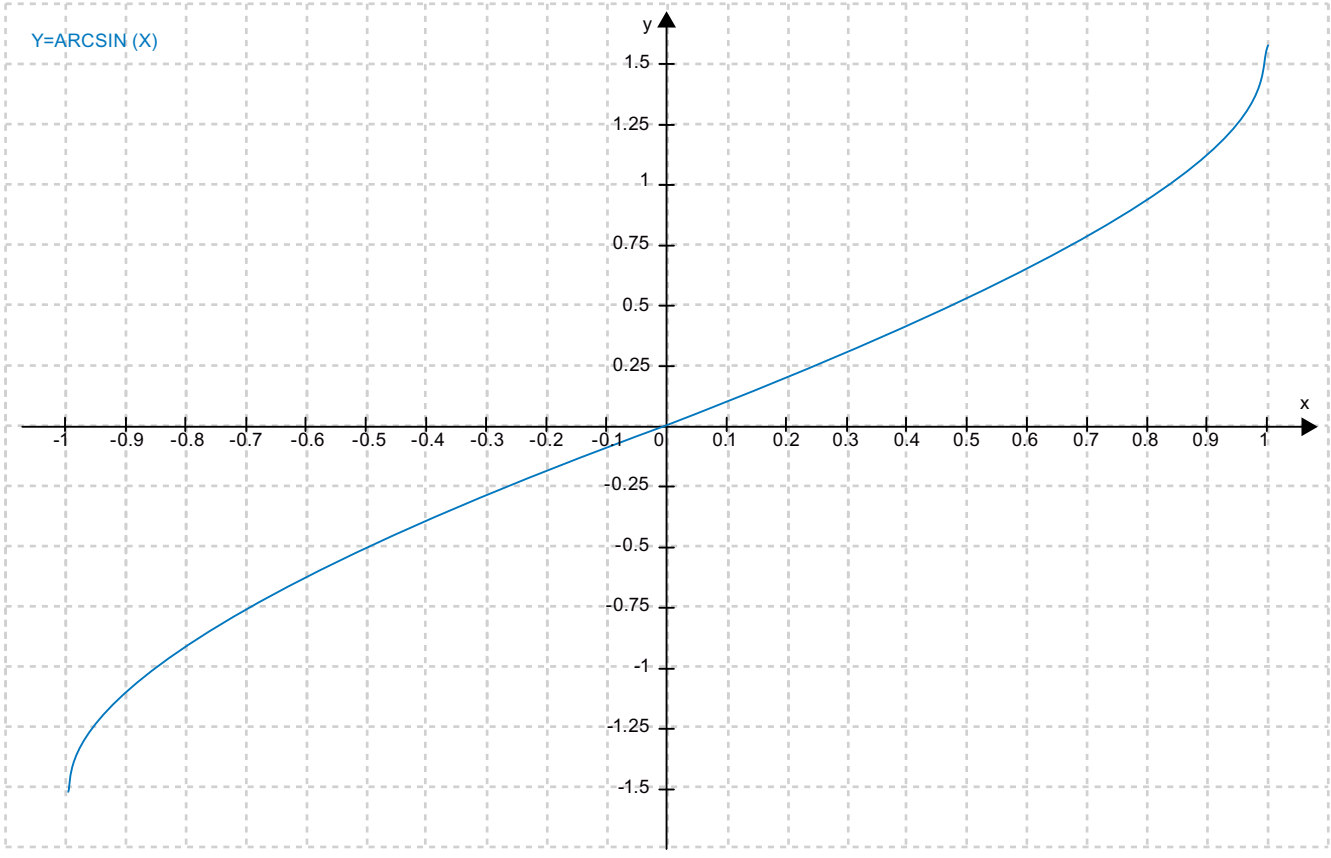
Y = arcsin X

Permitted input range: -1.0 <= X <= +1.0

Output range: -∏/2 <= Y <= ∏/2

If the argument lies outside of the permitted input range of |X| <= 1.0, output Y is limited to -∏/2 (when X < -1.0) or ∏/2 (when X > +1.0) and binary output QF = 1 is set at the same time.

XY diagram



Block connections

Block connection	Description	Default	Value range	Attributes
X	Argument	0.0	REAL	
Y	Arc sine	0.0	REAL	
QF	Input error	0	0/1	

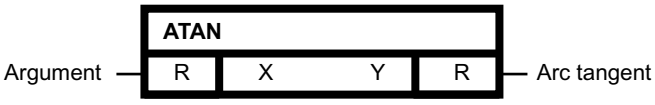
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted on-line	Yes
Special characteristics	-

8.1.3.7 ATAN

Arc tangent function

Symbol



Brief description

Determination of the arc tangent value for an argument

Method of operation

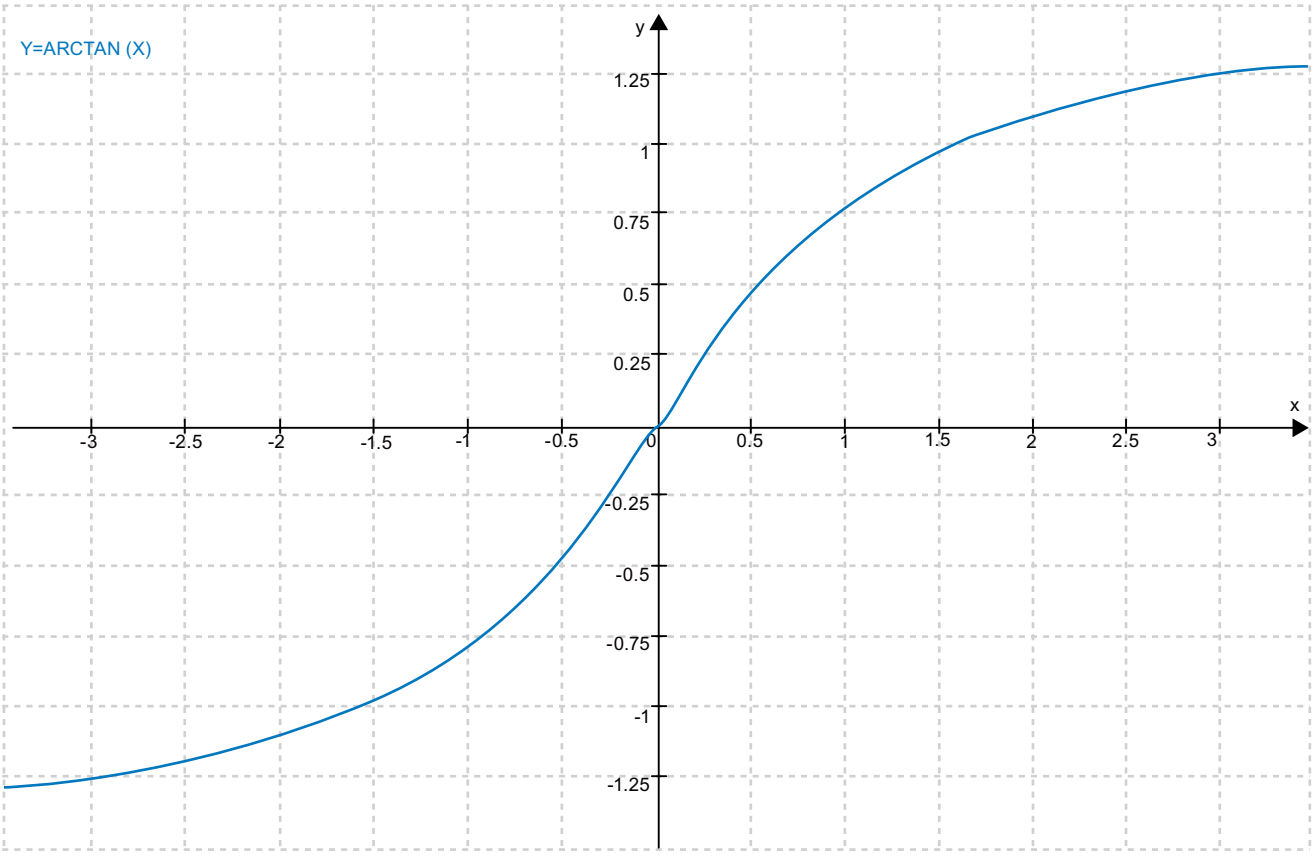
The block determines the associated arc tangent value in radian measure for an argument to be entered at input X and outputs the result at output Y.

Y = arctan X

Permitted input range: -3.402823 E38 to 3.402823 E38

Output range: -∏/2 <= Y <= ∏/2

XY diagram



Block connections

Block connection	Description	Default	Value range	Attributes
X	Argument	0.0	REAL	
Y	Arc tangent	0.0	REAL	

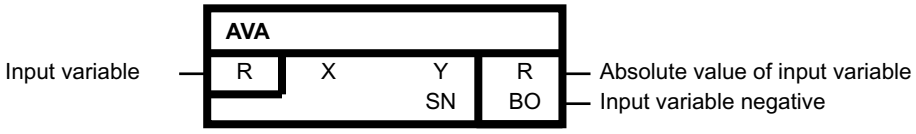
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.3.8 AVA

Absolute value generator, with sign evaluation

Symbol



Brief description

Arithmetic function block for absolute value generation of type real

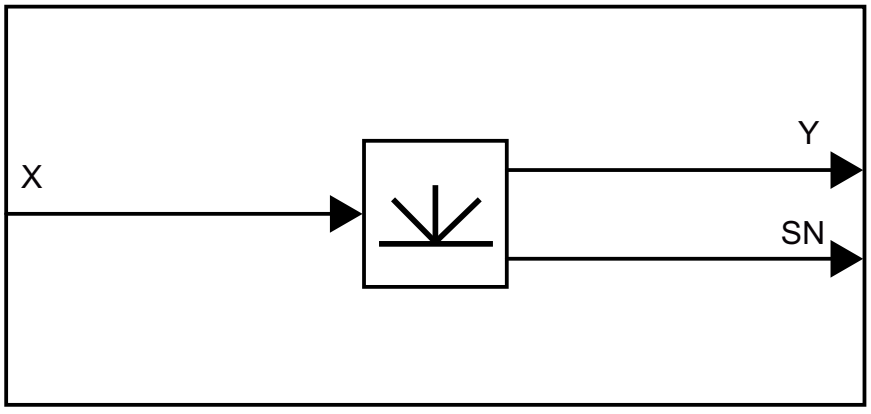
Method of operation

The block generates the absolute value of the value at input X (input variable). The result is output at output Y.

Y = |X|

If the input variable is negative, binary output SN = 1 is set at the same time.

Block diagram



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
Y	Absolute value of input variable	0.0	REAL	
SN	Input variable negative	0	0/1	

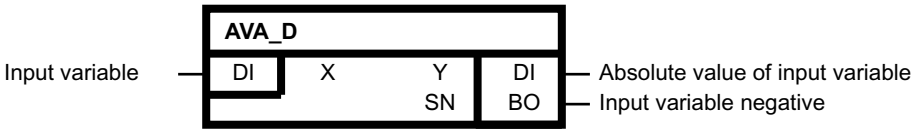
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.3.9 AVA_D

Absolute value generator (double integer)

Symbol



Brief description

Arithmetic function block for absolute value generation of type DOUBLE INTEGER

Method of operation

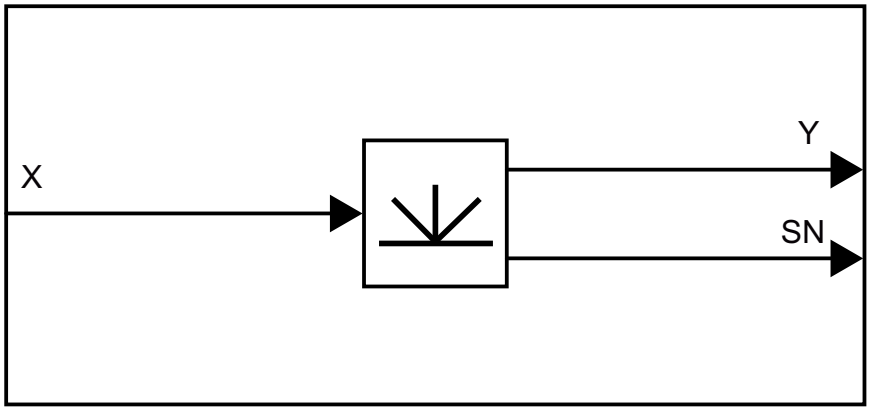
The block generates the absolute value of the value at input X (input variable). The result is output at output Y.

Y = |X|

If the input variable is negative, binary output SN = 1 is set at the same time.

Output values Y -2147483648 and SN 1 are set for input value -2147483648.

Block diagram



Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Absolute value of input variable	0	DINT	
SN	Input variable negative	0	0/1	

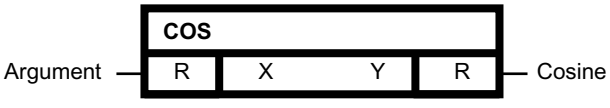
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.3.10 COS

Cosine function

Symbol



Brief description

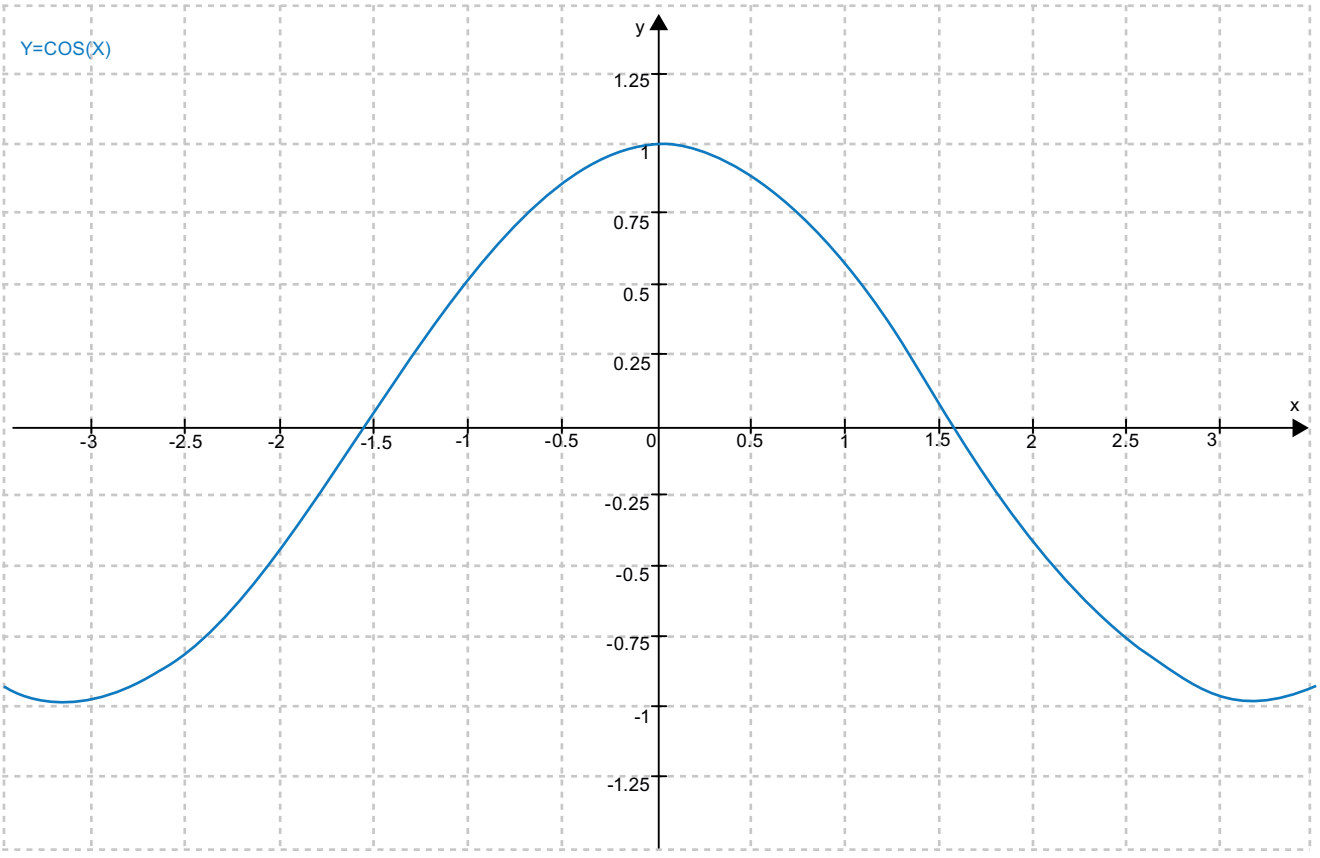
Determination of the cosine value for an argument

Method of operation

The block determines the associated cosine value in radian measure for an argument to be entered at input X and outputs the result at output Y.

Y = cos X

XY diagram



X is modular ∏

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Argument	0.0	REAL	
Y	Cosine	1	REAL	

Configuration data

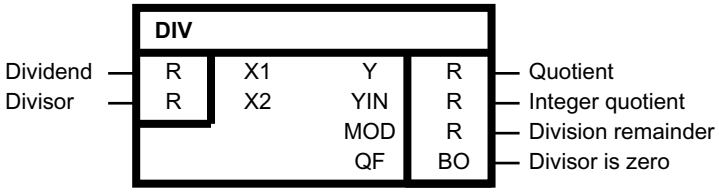
SIMOTION	✓ (as of V4.1)
SINAMICS	✓ (as of V4.4)

Can be loaded online	Yes
Special characteristics	-

8.1.3.11 DIV

Divider (REAL type)

Symbol



Brief description

Divider with two real-type inputs

Method of operation

The block divides the value entered at connection X1 by the value entered at connection X2.

The result is output at outputs Y, YIN, and MOD:

- The Y output contains the quotient with integer places and decimal places
- The YIN output contains the integer quotient
- The MOD output contains the division remainder (absolute residual value)

The Y output is limited to a range of approximately -3.4 E38 to +3.4 E38.

$$Y = \frac{X1}{X2}$$

MOD = (Y – YIN) * X2

If the output value Y violates the permissible value range of -3.402823 E38 to 3.402823 E38 (because the divisor X2 is very small or zero), then the limit value of the output range is output at connection Y with the correct sign. The binary output QF=1 is set at the same time. If X2 is zero, then the outputs YIN and MOD retain their last values.

With division of 0/0, the block output Y remains unchanged. The binary output QF is set to 1. With a division by zero, output MOD retains its last value.

Truth table(s)

The following truth table lists the block responses in the cases specified above.

X1/X2	Y	YIN	MOD	OF
X/0	Limit value with correct sign	YIN n-1	MOD n-1	1
0/0	Y n-1	YIN n-1	MOD n-1	1
0/X	0	0	0	0

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Dividend	0.0	REAL	
X2	Divisor	1	REAL	
Y	Quotient	0.0	REAL	
YIN	Integer quotient	0.0	REAL	
MOD	Division remainder	0.0	REAL	
QF	Divisor is zero	0	0/1	

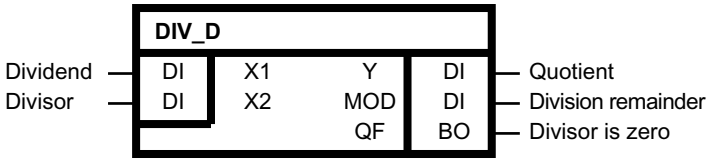
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.3.12 DIV_D

Divider (double integer type)

Symbol



Brief description

Divider with two inputs of the double integer type

Method of operation

The block divides the value entered at connection X1 by the value entered at connection X2 taking account of the sign. The quotient, limited to the range of -2147483648 (2^{31}) to 2147483647 (2^{31} -1), is output at connection Y.

$$Y = \frac{X1}{X2}$$

The division remainder is output at connection MOD. The sign of the division remainder MOD matches that of dividend X1.

MOD = X1 MOD X2

When output value Y exceeds the permissible range of -2147483648 (2^{31}) to +2147483647 (2^{31}-1) (because divisor X2 is zero), then the limit value of the output range with the correct sign is output at connection Y. The binary output QF = 1 is set at the same time.

With division of 0/0, the block output Y remains unchanged. The binary output QF is set to 1. With a division by zero, output MOD retains its last value.

Truth table(s)

The following truth table lists the block responses in the cases specified above.

X1/X2	Y	MOD	OF
X/0	Limit value with correct sign	MOD n-1	1
0/0	Y n-1	MOD n-1	1
0/X	0	0	0

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Dividend	0	DINT	
X2	Divisor	1	DINT	
Y	Quotient	0	DINT	
MOD	Division remainder	0	DINT	
QF	Divisor is zero	0	0/1	

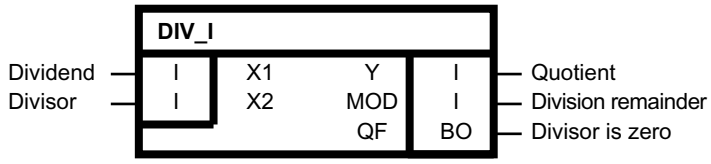
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.3.13 DIV_I

Divider (integer type)

Symbol



Brief description

Divider with two integer-type inputs

Method of operation

The block divides the value entered at connection X1 by the value entered at connection X2 taking account of the sign. The quotient is limited to the range of -32768 to +32767 and output at connection Y.

$$Y = \frac{X1}{X2}$$

The division remainder is output at connection MOD. The sign of the division remainder MOD matches that of dividend X1.

MOD = X1 MOD X2

When output value Y exceeds the permissible range of -32768 to +32767 (because the divisor is zero), then the limit value of the output range with the correct sign is output at connection Y. The binary output QF = 1 is set at the same time.

With division of 0/0, the block output Y remains unchanged. The binary output QF is set to 1. With a division by zero, output MOD retains its last value.

Truth table(s)

The following truth table lists the block responses in the cases specified above.

X1/X2	Y	MOD	OF
X/0	Limit value with correct sign	MOD n-1	1
0/0	Y n-1	MOD n-1	1
0/X	0	0	0

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Dividend	0	INT	
X2	Divisor	1	INT	
Y	Quotient	0	INT	
MOD	Division remainder	0	INT	
QF	Divisor is zero	0	0/1	

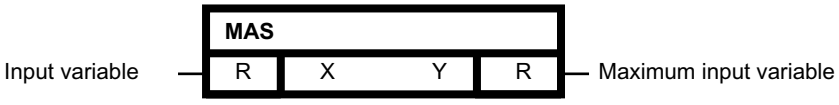
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.3.14 MAS

Maximum evaluator

Symbol



Brief description

Comparison block with up to four inputs of the REAL type to determine the largest input value present at the time of processing

Method of operation

The block determines the largest of the values present at inputs X 1-4.

The result is output at output Y.

Y = max. {X1, X2, X3, X4}

If the same value is present at all inputs, this value is output as the maximum input variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	-3.402823 E38	REAL	
Y	Maximum input variable	0.0	REAL	

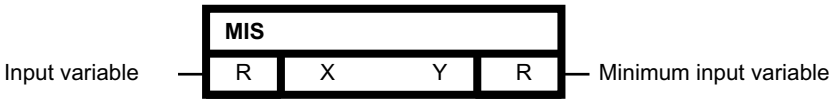
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.15 MIS

Minimum evaluator

Symbol



Brief description

Comparison block with up to four REAL-type inputs to determine the smallest input value present at the time of processing.

Method of operation

The block determines the smallest of the values present at inputs X 1-4.

The result is output at output Y.

Y = min. {X1, X2, X3, X4}

If the same value is present at all inputs, this value is output as the minimum input variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	3.402823 E38	REAL	
Y	Minimum input variable	0.0	REAL	

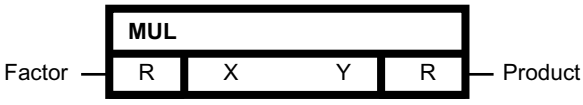
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.16 MUL

Multiplier (REAL type)

Symbol



Brief description

Multiplier with up to four real-type inputs

Method of operation

The block multiplies the values entered at the generic inputs X 1-4 taking account of the sign. The result, limited to the range of -3.402823 E38 to +3.402823 E38, is output at output Y.

Y = X1 x X2 x X3 x X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Factor	1.0	REAL	
Y	Product	0.0	REAL	

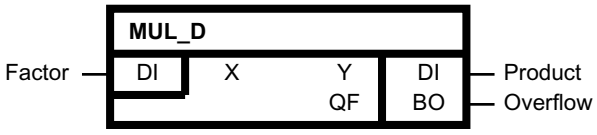
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.17 MUL_D

Multiplier (double integer type)

Symbol



Brief description

Multiplier with up to four double integer-type inputs

Method of operation

The block multiplies the values entered at the generic inputs X 1-4 taking account of the sign. The result, limited to a range of approximately -2147483648 (2^{31}) to +2147483647 (2^{31}-1), is output at output Y.

Y = X1 x X2 x X3 x X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Factor	1	DINT	
Y	Product	0	DINT	
QF	Overflow	0	0/1	

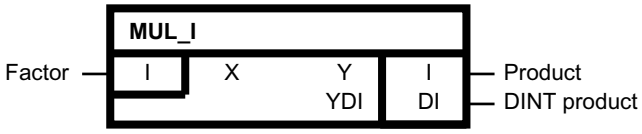
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.18 MUL_I

Multiplier (integer type)

Symbol



Brief description

Multiplier with up to four integer-type inputs

Method of operation

The block multiplies the values entered at the generic inputs X 1-4 taking account of the sign. The result, limited to the range of -32768 to +32767, is output at output Y. In addition, the result, limited to a range of -2147483648 (2^{31}) to +2147483647 (2^{31}-1), is output at output YDI.

Y = X1 x X2 x X3 x X4

Block connections

Block connection	Description	Default	Value range	Attributes
X	Factor	1	INT	
Y	Product	0	INT	
YDI	DINT product	0	DINT	

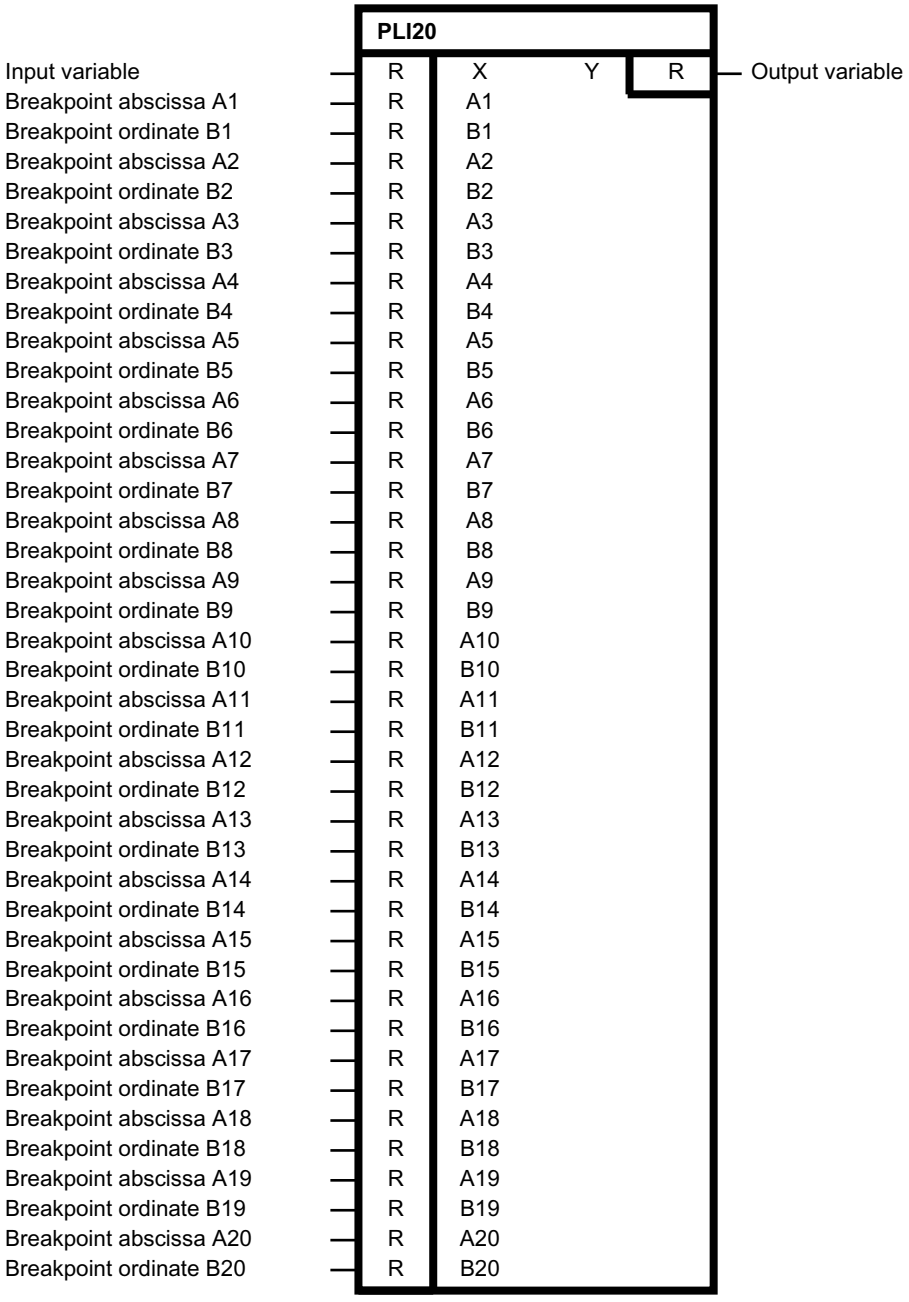
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	X comprises up to four inputs (X1 to X4)

8.1.3.19 PLI20

Polyline, 20 breakpoints

Symbol



Brief description

Block of the REAL type

- For linearization of characteristic curves
- For simulation of non-linear transfer elements
- For controller gain defined in sections

Method of operation

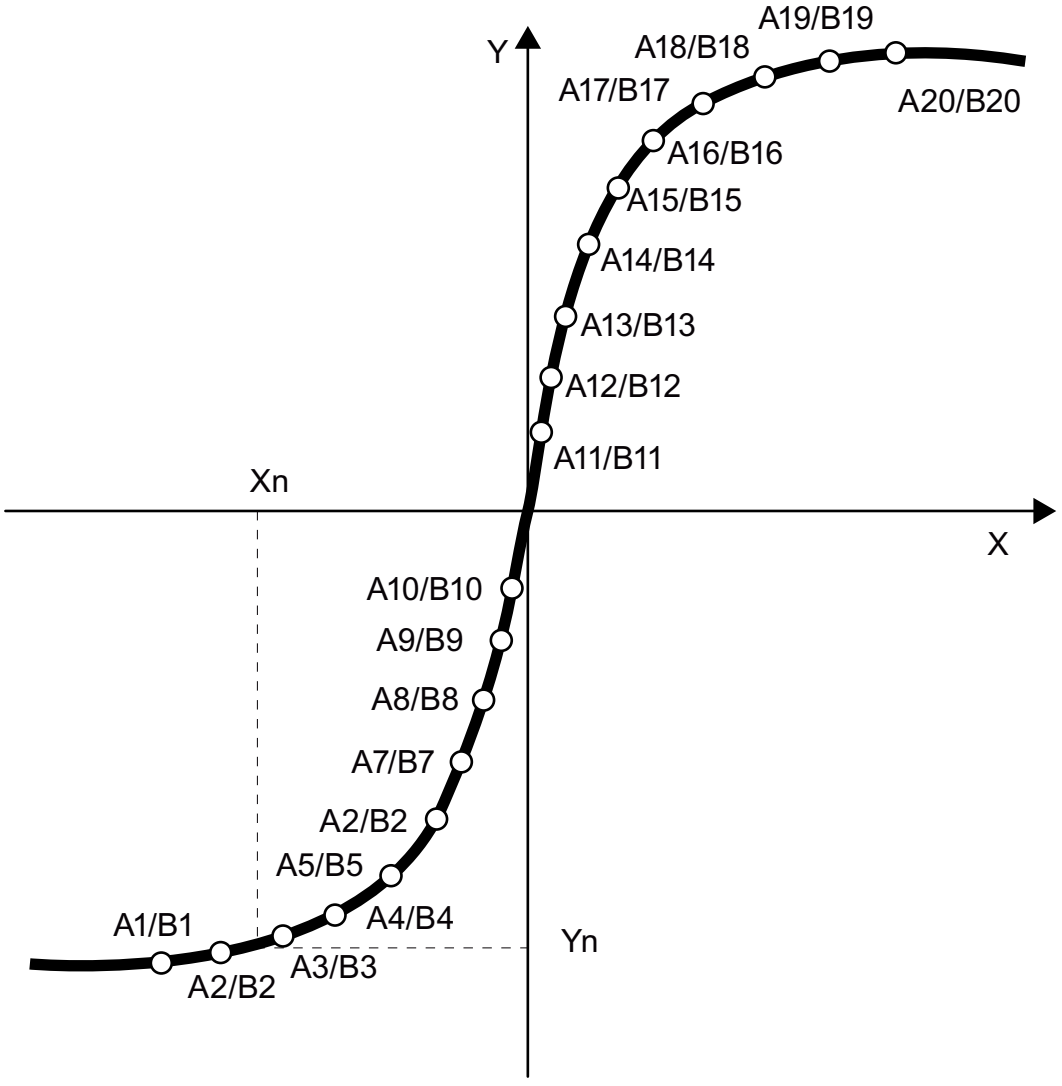
- The block adapts output variable Y arbitrarily to input variable X by means of up to 20 breakpoints in four quadrants.
- Interpolation is carried out linearly between the breakpoints. Outside of A1 and A20, the characteristic curve runs horizontally.

Configuration notes

During configuration, you must ensure that the values of A1 to A20 are sorted in ascending order otherwise incorrect values are output. The ordinate values B1 to B20 can be selected arbitrarily, i.e. irrespective of the preceding value.

If breakpoints are not needed (e.g. as of A16/B16), the following abscissas and ordinates (A16/B16 to A20/B20) must be assigned the same values as A15/B15.

Example



Simulation of the magnetization characteristic curve

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
A1	Breakpoint abscissa A1	0.0	REAL	
B1	Breakpoint ordinate B1	0.0	REAL	
O2	Breakpoint abscissa A2	0.0	REAL	
B2	Breakpoint ordinate B2	0.0	REAL	
O3	Breakpoint abscissa A3	0.0	REAL	
B3	Breakpoint ordinate B3	0.0	REAL	

Block connection	Description	Default	Value range	Attrib-utes
A4	Breakpoint abscissa A4	0.0	REAL	
B4	Breakpoint ordinate B4	0.0	REAL	
A5	Breakpoint abscissa A5	0.0	REAL	
B5	Breakpoint ordinate B5	0.0	REAL	
A6	Breakpoint abscissa A6	0.0	REAL	
B6	Breakpoint ordinate B6	0.0	REAL	
A7	Breakpoint abscissa A7	0.0	REAL	
B7	Breakpoint ordinate B7	0.0	REAL	
A8	Breakpoint abscissa A8	0.0	REAL	
B8	Breakpoint ordinate B8	0.0	REAL	
A9	Breakpoint abscissa A9	0.0	REAL	
B9	Breakpoint ordinate B9	0.0	REAL	
A10	Breakpoint abscissa A10	0.0	REAL	
B10	Breakpoint ordinate B10	0.0	REAL	
A11	Breakpoint abscissa A11	0.0	REAL	
B11	Breakpoint ordinate B11	0.0	REAL	
A12	Breakpoint abscissa A12	0.0	REAL	
B12	Breakpoint ordinate B12	0.0	REAL	
A13	Breakpoint abscissa A13	0.0	REAL	
B13	Breakpoint ordinate B13	0.0	REAL	
A14	Breakpoint abscissa A14	0.0	REAL	
B14	Breakpoint ordinate B14	0.0	REAL	
A15	Breakpoint abscissa A15	0.0	REAL	
B15	Breakpoint ordinate B15	0.0	REAL	
A16	Breakpoint abscissa A16	0.0	REAL	
B16	Breakpoint ordinate B16	0.0	REAL	
A17	Breakpoint abscissa A17	0.0	REAL	
B17	Breakpoint ordinate B17	0.0	REAL	
A18	Breakpoint abscissa A18	0.0	REAL	
B18	Breakpoint ordinate B18	0.0	REAL	
A19	Breakpoint abscissa A19	0.0	REAL	
B19	Breakpoint ordinate B19	0.0	REAL	
A20	Breakpoint abscissa A20	0.0	REAL	
B20	Breakpoint ordinate B20	0.0	REAL	
Y	Output variable	0.0	REAL	

Configuration data

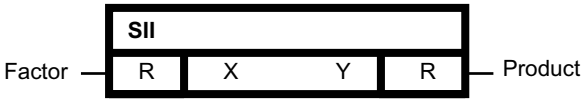
SIMOTION	✓
SINAMICS	✓

Can be loaded online	Yes
Special characteristics	-

8.1.3.20 SII

Inverter

Symbol



Brief description

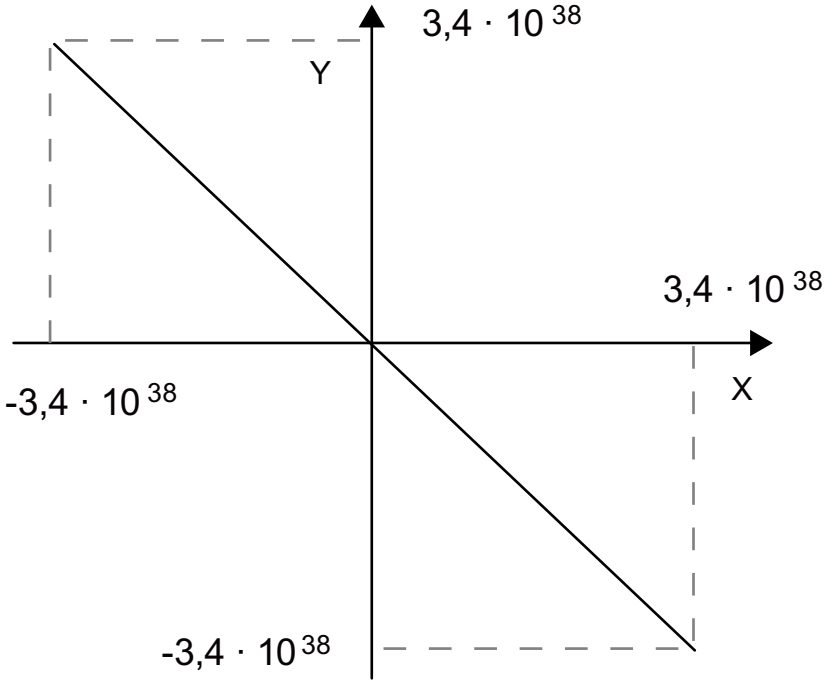
- Inverter with one Real-type input
- Arithmetic function block for sign reversal

Method of operation

The block inverts input variable X and outputs the result at block output Y (in accordance with the following transmission characteristic).

Y=-X

Transfer function



Block connections

Block connection	Description	Default	Value range	Attributes
X	Factor	0.0	REAL	
Y	Product	0.0	REAL	

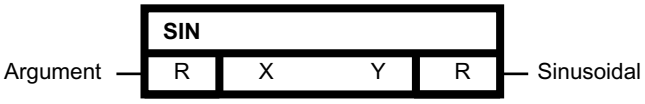
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.3.21 SIN

Sine function

Symbol



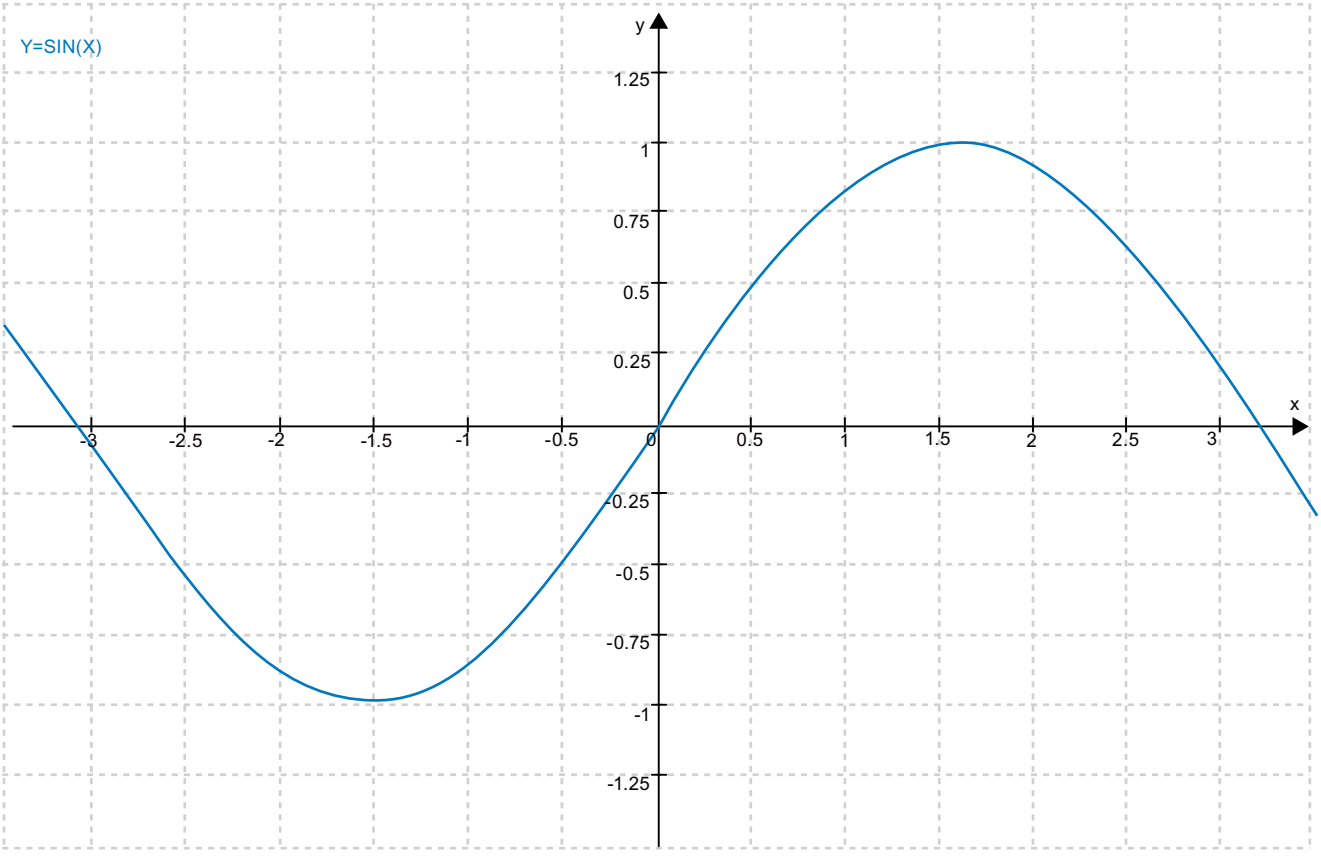
Brief description

Determination of the sine value for an argument

Method of operation

- The block determines the associated sine value in radian measure for an argument to be entered at input X and outputs the result at output Y.
- Y = sin X

XY diagram



X is modular ∏

Block connections

Block connection	Description	Default	Value range	Attributes
X	Argument	0.0	REAL	
Y	Sinusoidal	0.0	REAL	

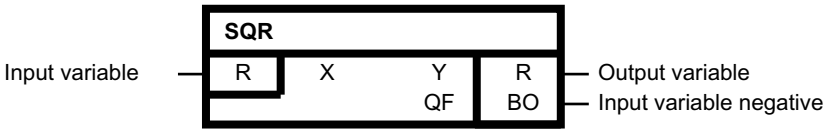
Configuration data

SIMOTION	✓ (as of V4.1)
SINAMICS	✓ (as of V4.4)
Can be loaded online	Yes
Special characteristics	-

8.1.3.22 SQR

Square-root extractor

Symbol



Brief description

Arithmetic function block for determining the square root

Method of operation

The block calculates the square root of the value entered at connection X. The result is output at connection Y.

$$Y = \sqrt{X}$$

If the input variable is negative, the value zero is output at connection Y. The binary output QF = 1 is set at the same time.

Truth table(s)

Condition	Y	QF
X > 0	SQR(X)	0
X = 0	0	0
X < 0	0	1

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0.0	REAL	
QF	Input variable negative	0	0/1	

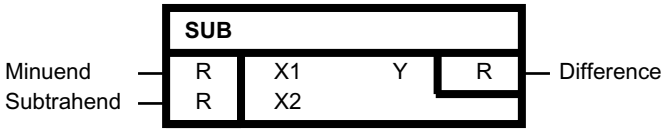
Configuration data

SIMOTION	✓ (as of V4.1)
SINAMICS	✓ (as of V4.4)
Can be inserted online	Yes
Special characteristics	-

8.1.3.23 SUB

Subtractor (REAL type)

Symbol



Brief description

Subtractor with two Real-type inputs

Method of operation

- The block subtracts the value entered at connection X2 from the value entered at connection X1, taking account of the sign. The result is limited to the range of -3.402823 E38 to 3.402823 E38 and output at output Y.
- Y= X1-X2

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Minuend	0.0	REAL	
X2	Subtrahend	0.0	REAL	
Y	Difference	0.0	REAL	

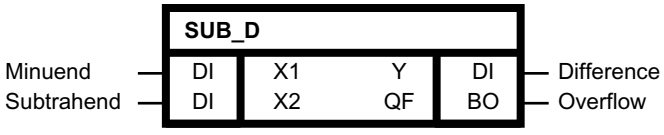
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.3.24 SUB_D

Subtractor (DOUBLE INTEGER type)

Symbol



Brief description

Subtractor with two inputs of the DOUBLE INTEGER type

Method of operation

The block subtracts the value entered at connection X2 from the value entered at connection X1, taking account of the sign. The result, limited to a range of -2147483648 (2^{31}) to +2147483647 (2^{31}-1), is output at output Y. An overflow is indicated at the binary output with QF=1.

Y= X1-X2

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Minuend	0	DINT	
X2	Subtrahend	0	DINT	

Block connection	Description	Default	Value range	Attributes
Y	Difference	0	DINT	
QF	Overflow	0	0/1	

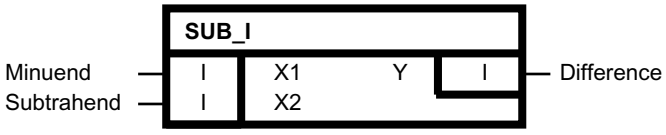
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.3.25 SUB_I

Subtractor (INTEGER type)

Symbol



Brief description

Subtractor with two inputs of the INTEGER type

Method of operation

- The block subtracts the value entered at connection X2 from the value entered at connection X1, taking account of the sign. The result, limited to a range of approximately -32768 to 32767, is output at output Y.
- Y= X1-X2

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Minuend	0	INT	
X2	Subtrahend	0	INT	
Y	Difference	0	INT	

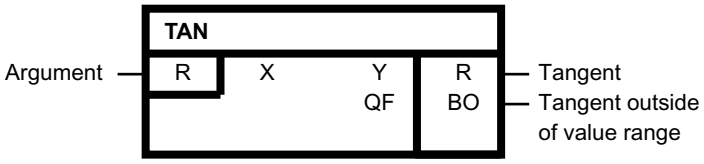
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.3.26 TAN

Tangent

Symbol



Brief description

Determination of the tangent value for an angle

Method of operation

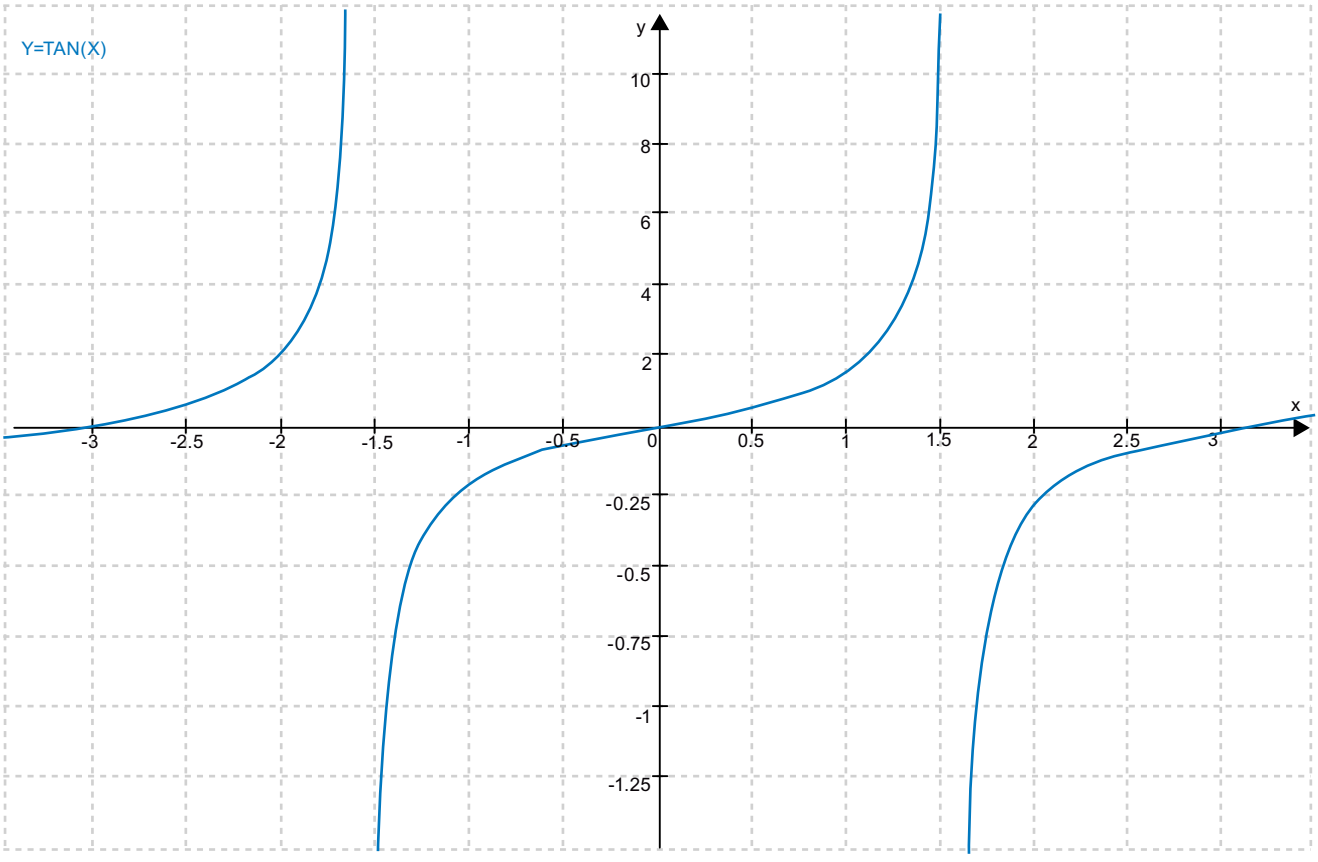
The block determines the associated tangent value in radian measure for an argument to be entered at input X and outputs the result at output Y.

Y = tan X

Output range: -3.402823 E38 to 3.402823 E38

If the tangent value determined lies outside of the range of -3.402823 E38 to 3.402823 E38, the block output Y is limited to -3.402823 E38 or +3.402823 E38, and the binary output QF = 1 is set at the same time.

Transfer function



Block connections

Block connection	Description	Default	Value range	Attributes
X	Argument	0.0	REAL	
Y	Tangent	0.0	REAL	
QF	Tangent outside the value range	0	0/1	

Configuration data

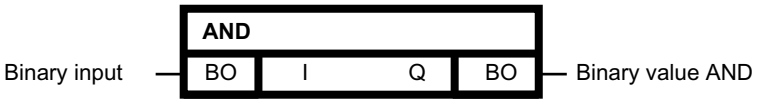
SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.4 Logic

8.1.4.1 AND

Logic AND operation (BOOL type)

Symbol



Brief description

AND block with up to four inputs of the BOOL type

Method of operation

The block combines the binary values at the inputs I 1-4 to a logic AND and outputs the result at its binary output Q.

$Q = I_{01} \wedge ... \wedge I_{04}$

Output Q = 1, when the value 1 is present at all generic inputs I1 to I4. In all other cases, output Q = 0.

Truth table(s)

Input				Output
I01	I02	I03	I04	Q
0	*	*	*	0
*	0	*	*	0
*	*	0	*	0
*	*	*	0	0
1	1	1	1	1

* Arbitrary

Block connections

Block connection	Description	Default	Value range	Attributes
I	Binary input	1	0/1	
Q	Binary value AND	0	0/1	

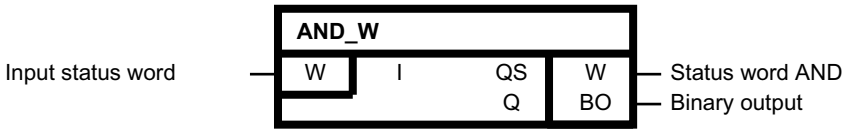
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	I comprises up to four connections (I1 to I4)

8.1.4.2 AND_W

Logic AND operation (WORD type)

Symbol



Brief description

AND_W block with up to four inputs of the WORD type

Method of operation

16 binary states are combined in a status word.

This function block links the status words I_{01} to I_{16} bit-by-bit according to the logic AND function. The corresponding bits of status word AND are then set at block output QS.

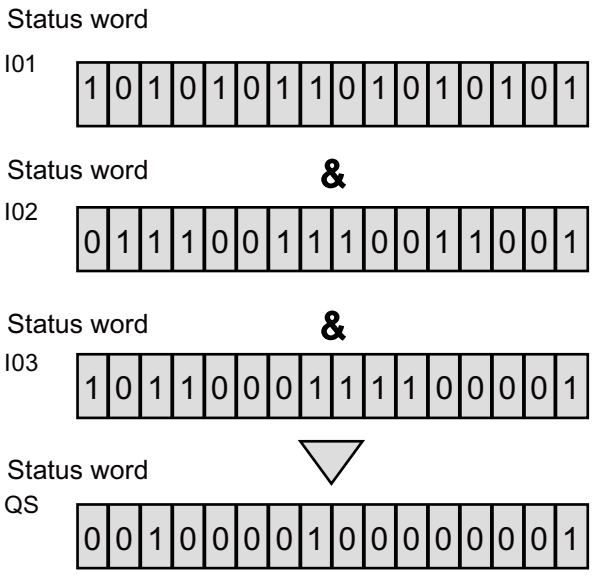
The following applies for bit k of status word AND:

$$QS_k = I_{01k} \wedge .. \wedge I_{nnk}, k = 1...16$$

A bit of the AND status word is equal to 0 when at least one of the equivalent bits on the block inputs I1 to I4 is equal to 0.

The binary output Q is 1 if at least one bit of the status word AND is equal to 1.

Following state diagram (for 3 inputs)



Block connections

Block connection	Description	Default	Value range	Attributes
I	Input status word	16#FFFF	WORD	
QS	Status word AND	16#0000	WORD	
Q	Binary output	0	0/1	

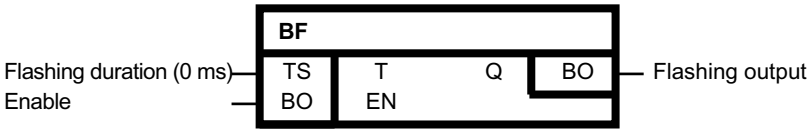
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	I comprises up to four inputs (I1 to I4)

8.1.4.3 BF

Flashing function (BOOL type)

Symbol



Brief description

Block of the BOOL type

- For controlling signal encoders
- As clock generator

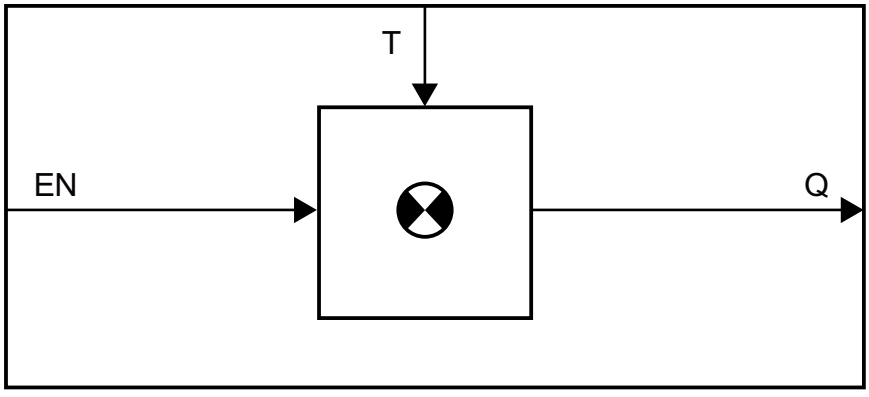
Method of operation

This block sets its output Q alternately to 1 and 0 at a frequency of interval T, as long as input EN = 1.

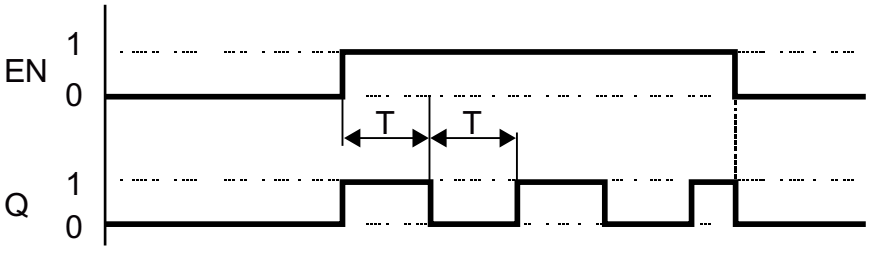
If enable input EN = 0, then output Q = 0.

In this case, T is both the light duration and the dark duration.

Block diagram



Time diagram



Flashing pulse Q subject to flashing duration T and the enable EN

Block connections

Block connection	Description	Default	Value range	Attributes
T	Flashing duration (0 ms)	0	SDTIME	
EN	Enable	0	0/1	
Q	Flashing output	0	0/1	

Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.4 BF_W

Flashing function for status word (WORD type)

Symbol

Brief description

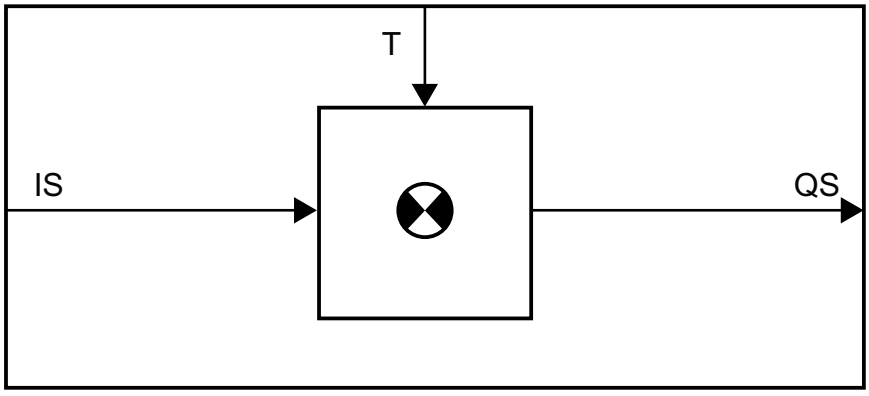
Block of the WORD type for controlling signal encoder combinations

Method of operation

This block sets all bits of the input status word IS that have a logic value of 1 alternately to 1 and 0 in the output status word QS at a frequency of interval T.

In this case, T is both the light duration and the dark duration.

Block diagram



Block connections

Block connection	Description	Default	Value range	Attributes
IS	Status word	16#0000	WORD	
T	Flashing duration (0 ms)	0	WORD	
QS	Flashing output status word	16#0000	WORD	

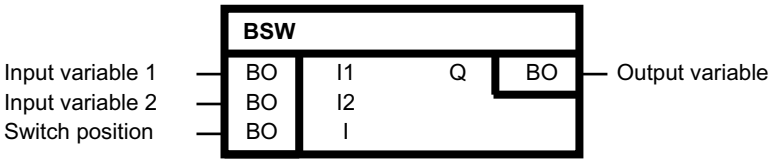
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.4.5 BSW

Binary change-over switch (BOOL type)

Symbol



Brief description

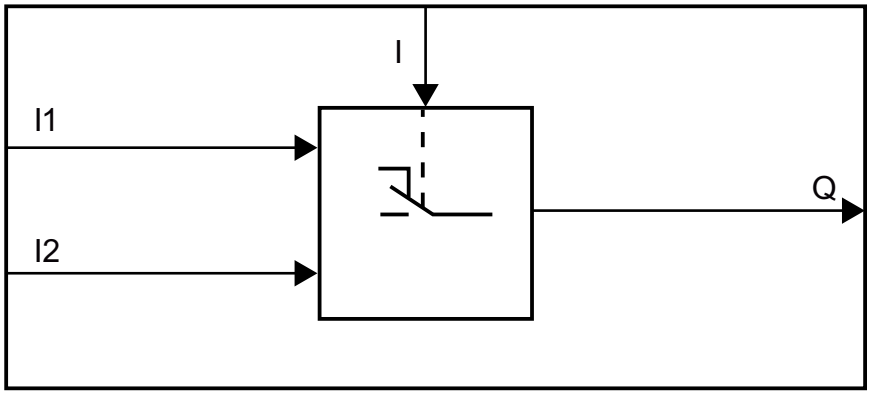
The block switches one of two binary input variables to the output.

Method of operation

If input I = 0, then I1 is given to output Q.

If input I = 1, then I2 is given to output Q.

Block diagram



Truth table(s)

Switch position 1	Output variable Q
0	Q = I1
1	Q = I1

Initialization

If input I = 0, then I1 is given to output Q.

If input I = 1, then I2 is given to output Q.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
I1	Input variable 1	0	0/1	
I2	Input variable 2	0	0/1	
I	Switch position	0	0/1	
Q	Output variable	0	0/1	

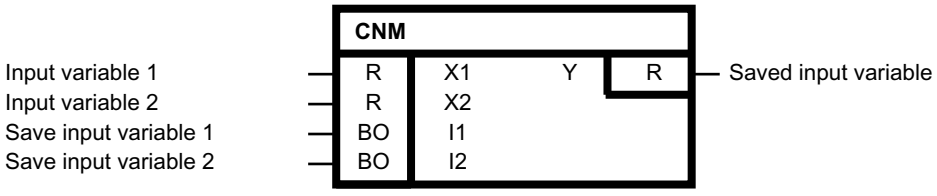
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.4.6 CNM

Controllable numeric memory (REAL type)

Symbol



Brief description

Block of the REAL type for saving an instantaneous input value (sample-and-hold function) with

- Selectable input
- Selectable save time
- Rising edge-initiated triggering

The blocks CNM_I and CNM_ D fulfill the same function. They only differ in the data type used.

Method of operation

On a rising edge at I1, X1 is switched through to output Y.

On a rising edge at I2, X2 is switched through to output Y.

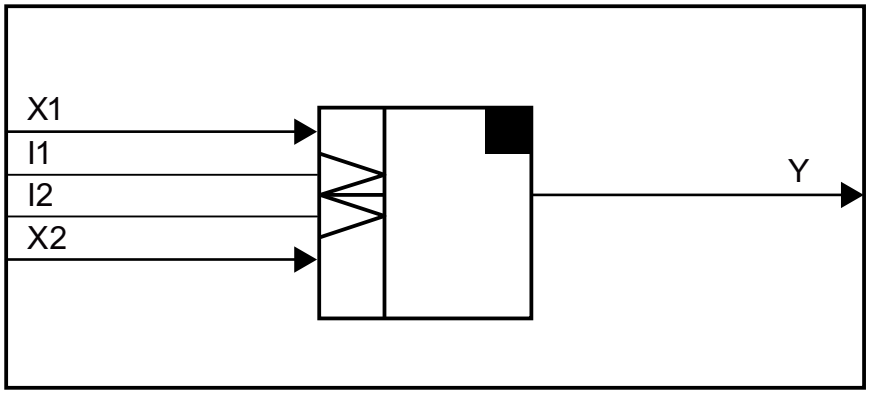
The saved input variable remains pending on Y until the next rising edge at I1 or I2 switches through the next instantaneous value.

In the case of a simultaneous rising edge at I1 and I2, I1 receives priority, and X1 is switched through to Y.

Initialization

If input I1 or I2 receives the value 1 during initialization of an upstream output, the block does not detect a positive edge during the first cyclic pass. Otherwise, the block detects a positive edge during the first cyclic pass. In START mode (edge memory bit), the values for I1 and I2 are stored temporarily.

Block diagram



Truth table(s)

Input		Output Y at the time of triggering
I1	I2	
*	*	$Y_n = Y_{n-1}$
*	0 -> 1	$Y_n = X2_n$
0 -> 1	*	$Y_n = X1_n$
0 -> 1	0 -> 1	$Y_n = X1_n$

*: No rising edge, 0 -> 1: Rising edge

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0.0	REAL	
X2	Input variable 2	0.0	REAL	
I1	Save input variable 1	0.0	0/1	
I2	Save input variable 2	0.0	0/1	
Y	Saved input variable	0.0	REAL	

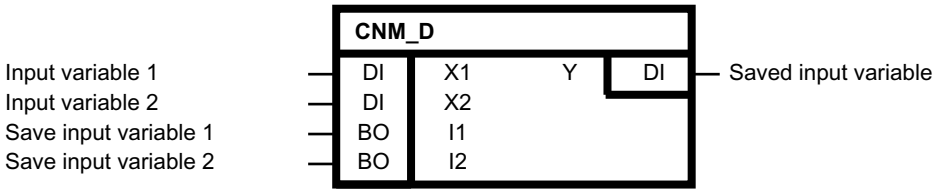
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.7 CNM_D

Controllable numeric memory (DOUBLE INTEGER type)

Symbol



Brief description

Block of the DOUBLE INTEGER type for saving a current input value (sample and hold function) with

- Selectable input
- Selectable save time
- Rising edge-initiated triggering

The blocks CNM and CNM_ i fulfill the same function. They only differ in the data type used.

Method of operation

On a rising edge at I1, X1 is switched through to output Y.

On a rising edge at I2, X2 is switched through to output Y.

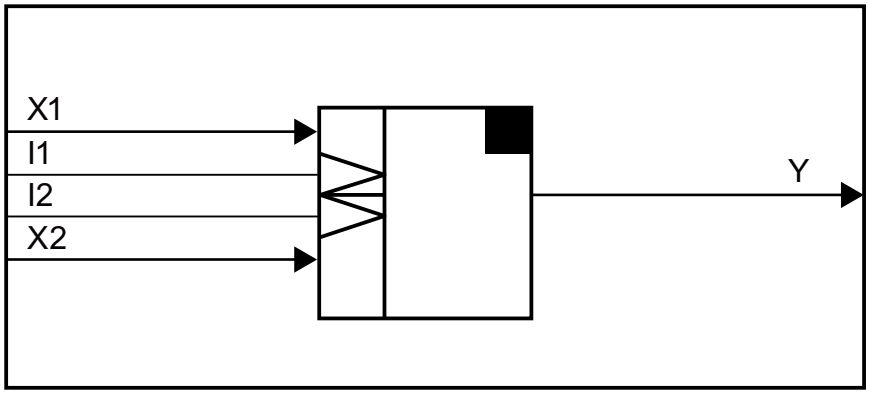
The saved input variable remains pending on Y until the next rising edge at I1 or I2 switches through the next momentary value.

In the case of a simultaneous rising edge at I1 and I2, I1 receives priority, and X1 is switched through to Y.

Initialization

If input I1 or I2 receives the value 1 during initialization of an upstream output, the block does not detect a positive edge during the first cyclic pass. The block detects a positive edge during the first cyclic pass. In START mode, the values for I1 and I2 are stored temporarily.

Block diagram



Truth table(s)

Input		Output Y at the time of triggering
I1	I2	
*	*	$Y_n = Y_{n-1}$
*	0 -> 1	$Y_n = X2_n$
0 -> 1	*	$Y_n = X1_n$
0 -> 1	0 -> 1	$Y_n = X1_n$

*: No rising edge, 0 -> 1: Rising edge

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	DINT	
X2	Input variable 2	0	DINT	
I1	Save input variable 1	0	0/1	
I2	Save input variable 2	0	0/1	
Y	Saved input variable	0	DINT	

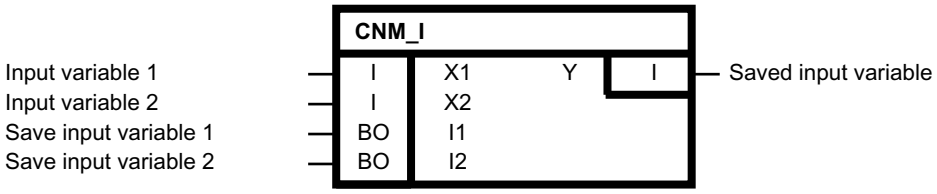
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.8 CNM_I

Controllable numeric memory (INTEGER type)

Symbol



Brief description

Block of the INTEGER type for saving a current input value (sample and hold function) with

- Selectable input
- Selectable save time
- Rising edge-initiated triggering

The CNM and CNM_D blocks have the same function. They only differ in the data type used.

Method of operation

On a rising edge at I1, X1 is switched through to output Y.

On a rising edge at I2, X2 is switched through to output Y.

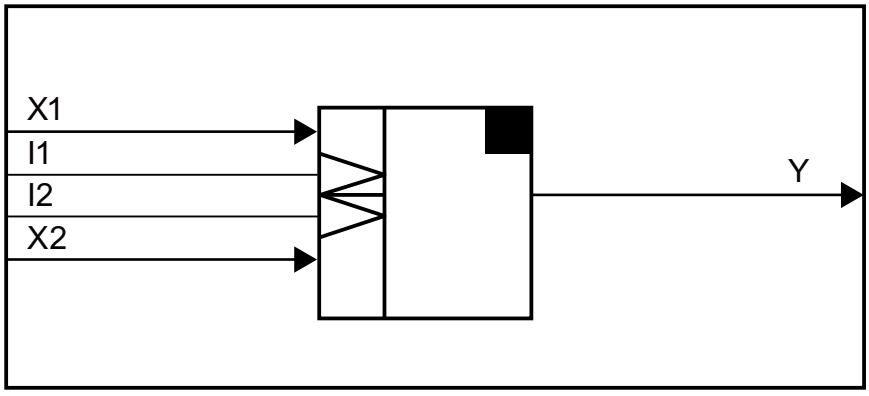
The saved input variable remains pending on Y until the next rising edge at I1 or I2 switches through the next momentary value.

In the case of a simultaneous rising edge at I1 and I2, I1 receives priority, and X1 is switched through to Y.

Initialization

If input I1 or I2 receives the value 1 during initialization of an upstream output, the block does not detect a positive edge during the first cyclic pass. The block detects a positive edge during the first cyclic pass. In START mode, the values for I1 and I2 are stored temporarily.

Block diagram



Truth table(s)

Input		Output Y at the time of triggering
I1	I2	
*	*	$Y_n = Y_{n-1}$
*	0 -> 1	$Y_n = X2_n$
0 -> 1	*	$Y_n = X1_n$
0 -> 1	0 -> 1	$Y_n = X1_n$

*: No rising edge, 0 -> 1: Rising edge

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	INT	
X2	Input variable 2	0	INT	
I1	Save input variable 1	0	0/1	
I2	Save input variable 2	0	0/1	
Y	Saved input variable	0	INT	

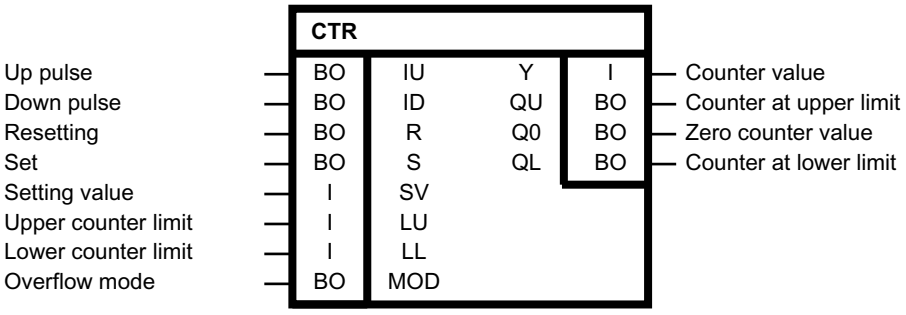
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.9 CTR

Counter (BOOL type)

Symbol



Brief description

Block for counting up and down with the following counter functions:

- Set counter to zero
- Hold counter at zero (disable)
- Set counter to initial value

Independent setting of upper and lower counter limit.

Method of operation

This block forms an edge-triggered up-down counter. With a rising edge of a pulse at input IU, the counter value is incremented.

With a rising edge of a pulse at input ID, the counter value is decremented. The counter value is present at output Y. Controlling the counter (see also truth table). With S=1, the counter value Y can be preset with the set value SV.

However, the reset input R has priority over the set input. As long as R is logic 1, Y is held at 0. The counter is disabled. If Y is not in the counting range between LL and LU, the output is set to the active limit value when R = 1.

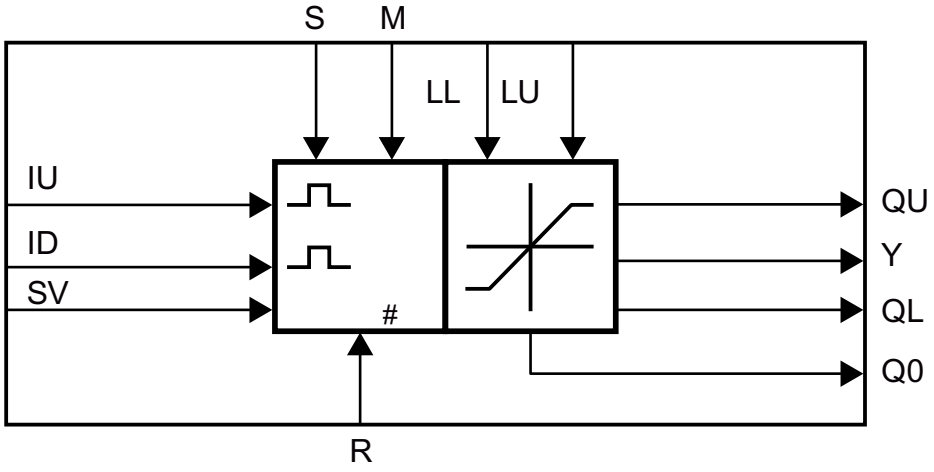
The working area of the counter can be specified via LU (upper counter limit value) or LL (lower counter limit value).

The setting value (SV) is in the range from LL >= SV >= LU.

MOD=0	When these limits are reached, the counter stops counting and the display QU (counter at upper limit) or QL (counter at lower limit) is set.
MOD=1	When the upper limit (LU) is reached, the counter value is set to the lower limit when the next Up pulse occurs; QU = 1 indicates positive overflow for a cycle.
	When the lower limit (LL) is reached, the counter value is set to the upper limit when the next Down pulse occurs; QL = 1 indicates negative overflow for a cycle.

When the counter value is zero, the output Q0 is set to 1.

Block diagram



Truth table(s)

Binary command	Binary command	Counter value Y
S	R	
0	0	Y is retained
0	1	Y is reset
1	0	Y = SV (set value)
1	1	Y is reset

Counter value when set/reset command is given

Initialization

The initialization defines the start value for the first cyclic pass. If input ID or IU is preset with 1, the block cannot detect a positive edge during the first cyclic pass.

Boundary conditions:

- LL <= Y <= LU for LL <LU
- Y = LU for LL >= LU

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IU	Up pulse	0	0/1	
ID	Down pulse	0	0/1	
R	Resetting	0	0/1	
S	Set	0	0/1	
SV	Setting value	0	INT	

Block connection	Description	Default	Value range	Attributes
LU	Upper counter limit	0	INT	
LL	Lower counter limit	0	INT	
MOD	Overflow mode	0	0/1	
Y	Counter value	0	INT	
QU	Counter at upper limit	0	0/1	
Q0	Zero counter value	0	0/1	
QL	Counter at lower limit	0	0/1	

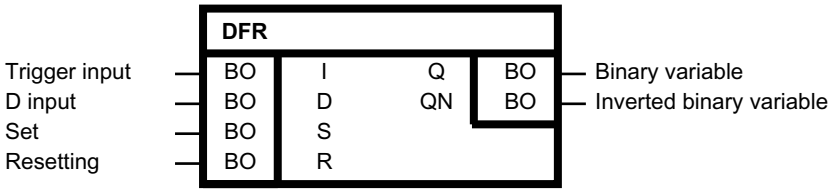
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.10 DFR

Reset-dominant D-type flip-flop (BOOL type)

Symbol



Brief description

Block of the BOOL type for use as reset-dominant D-type flip-flop

Method of operation

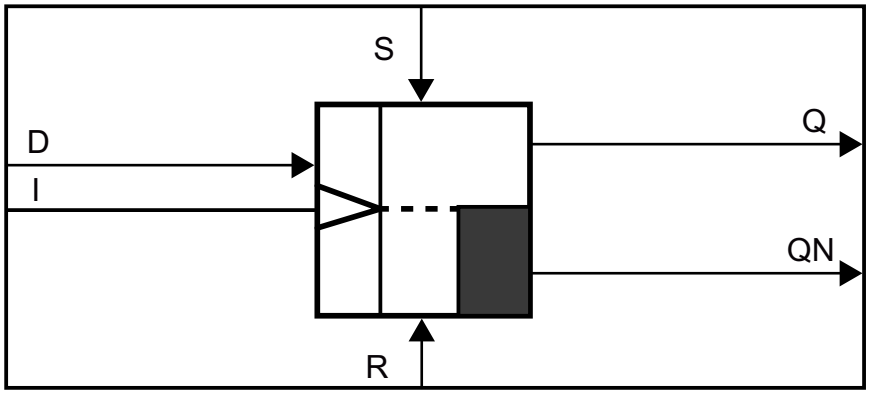
If the two inputs S and R are logic 0, the D input information is switched through to output Q on a rising edge at trigger input I. Output QN always has the value inverse to Q. With logic 1 at input S, output Q is set to logic 1. If input R is set to logic 1, then output Q is set to logic 0. If both inputs are logic 0, then Q does not change. However, if the two inputs S and R are logic 1, then Q is logic 0, since the reset input dominates.

Initialization

If input I receives the value 1 during initialization from an upstream output, the block does not detect a positive edge during the first cyclic pass.

Otherwise, the block detects a positive edge during the first cyclic pass. In START mode, the value for I is stored temporarily.

Block diagram

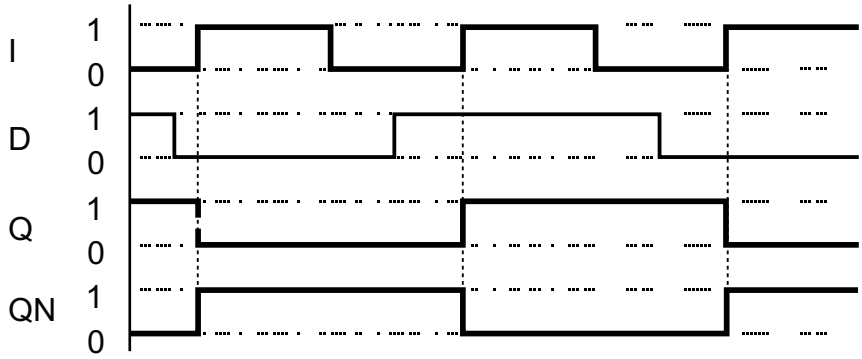


Truth table(s)

D	I	Binary command		Output states	
		S	R	Q	QN
0	0 -> 1	0	0	0	1
1	0 -> 1	0	0	1	0
*	1 -> 0	0	0	Q_{n-1}	Q_{n-1}
*	*	0	1	0	1
*	*	1	0	1	0
*	*	1	1	0	1

Time diagram

With D and I



Output pulse Q subject to the D input and input pulse I for S = R = 0

Block connections

Block connection	Description	Default	Value range	Attributes
I	Trigger input	0	0/1	
D	D input	0	0/1	
S	Set	0	0/1	
R	Resetting	0	0/1	
Q	Binary variable	0	0/1	
QN	Inverted binary variable	1	0/1	

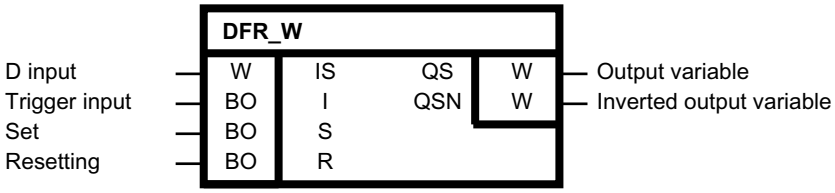
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.11 DFR_W

Reset-dominant D-type flip-flop (WORD type)

Symbol



Brief description

Block of the WORD type for use as reset-dominant D-type flip-flop

Method of operation

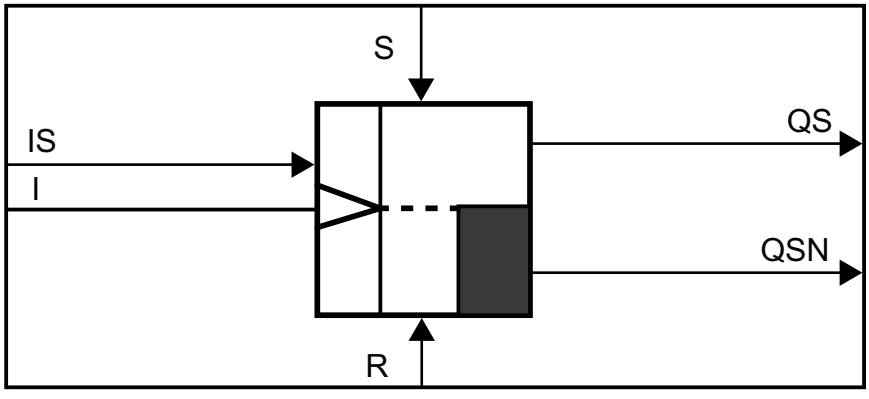
If the two inputs S and R = 0, the D input information is switched through to output QS on a rising edge at trigger input I. Output QSN always has the value inverse to QS. If S = 1, all bits of output variable QS are set to 1. If R = 1, all bits of output variable QS are set to 0. If both inputs S and R = 0, then QS does not change. If the two inputs S and R = 1, all bits of output variable QS are set to 0, since the reset input R dominates.

Initialization

If input I receives the value 1 during initialization from an upstream output, the block does not detect a positive edge during the first cyclic pass.

Otherwise, the block detects a positive edge during the first cyclic pass. In START mode, the value for I is stored temporarily.

Block diagram



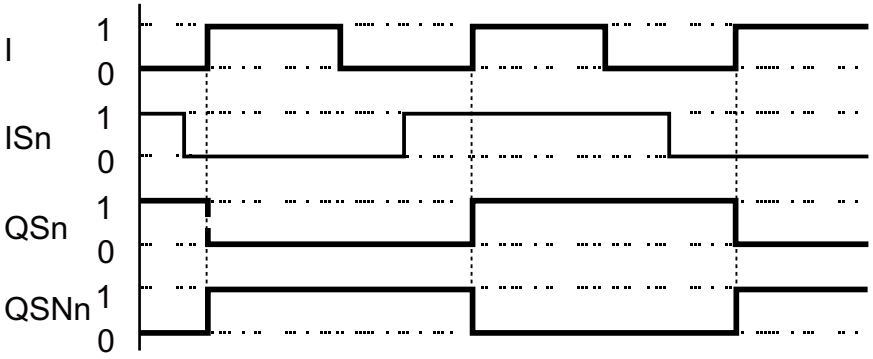
Truth table(s)

I	Binary command		Output states	
	S	R	QS	QSN
0 -> 1	0	0	IS	IS inverted
*	0	1	0	1
*	1	0	1	0
*	1	1	0	1

* Arbitrary

Time diagram

With I and IS



Output variables QS and QSN depending on trigger input I and D-input IS for S = R = 0 (n is the bit number)

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IS	D input	16#0000	WORD	
I	Trigger input	0	0/1	
S	Set	0	0/1	
R	Resetting	0	0/1	
QS	Output variable	16#0000	WORD	
QSN	Inverted output variable	16#FFFF	WORD	

Configuration data

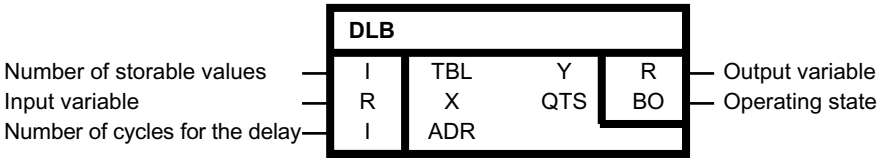
SIMOTION	✓
SINAMICS	-

Can be loaded online	Yes
Special characteristics	-

8.1.4.12 DLB

Delay element (REAL type)

Symbol



Brief description

Block of the REAL type for the output of an input variable which is delayed by a specifiable number of sampling times.

Method of operation

If the operating state is QTS = 1, the block contains a delay memory of the TBL variable. The input variable specified at input X is output after a delay as output variable Y. The delay is specified by the integer multiple ADR of the sampling time (time slice in which the block is calculated). When operating mode QTS = 0, the delay memory is not activated. In this case, the input variable specified at input X is output immediately as output variable Y.

Initialization

During initialization, the delay memory is requested for the purpose of acquiring TBL input variables. The delay memory can contain a maximum of 1000 values. If TBL < 0, TBL is limited to 0. QTS = 1 indicates that the delay memory requested in TBL is available. QTS = 0 indicates that the system was not able to make the memory available, due to a lack of resources, or a TBL value > 1000 has been defined. In this case, output Y is corrected to input X during cyclic operation.

Block connections

Block connection	Description	Default	Value range	Attributes
TBL	Number of storable values	100	0...1000	
X	Input variable	0.0	REAL	
ADR	Number of cycles for the delay	0	0...1000	
Y	Output variable	0.0	INT	
QTS	Operating state	0	0/1	

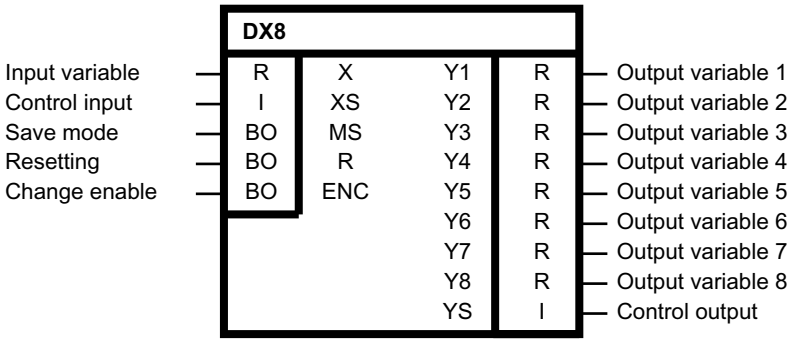
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.4.13 DX8

Demultiplexer, eight outputs, cascadable (REAL type)

Symbol



Brief description

Block of the REAL type for demultiplex operation. This block is cascadable.

Method of operation

Depending on ENC, R, MS and XS = 1 to 8, the block switches through its input X to one of the eight selectable outputs Y1 to Y8 (example: XS = 3 means Y3 = X).

When XS = 0 or XS >= 9, none of the block inputs Y1 to Y8 is selected. Non-selected outputs are either set to zero or retain their previous value until the next change.

The following priority sequence applies for the control inputs:

ENC before R before MS

When ENC = 0, all outputs Y1 to Y8 remain unchanged, regardless of R and MS.

When ENC = 1, outputs Y1 to Y8 are enabled for change.

When R = 1, all outputs Y1 to Y8 receive the value 0, irrespective of MS.

When MS = 0 (non-latching mode), all outputs Y1 to Y8 not selected by XS receive the value 0.

When MS = 1 (storing mode), all outputs not selected by XS remain unchanged.

Truth table(s)

ENC	R	MS	XS	Outputs Y1 to Y8
0	*	*	*	The previous values are retained
1	1	*	*	Y1 to Y8 = 0
1	0	0	1 <= XS <= 8	• Selected output = X • Output not selected = 0
1	0	0	XS = 0 or XS >= 9	Y1 to Y8 = 0
1	0	1	1 <= XS <= 8	• Selected output = X • Outputs not selected remain unchanged
1	0	1	XS = 0 or XS >= 9	All previous values remain unchanged

Cascading

The block output YS must be connected to the block input XS of the following block.

For XS = 0 to 8, YS = 0

When XS > 8: YS = XS-8

(use for cascading)

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
XS	Control input	0	INT	
MS	Save mode	0	0/1	
R	Resetting	0	0/1	
ENC	Change enable	0	0/1	
Y1	Output variable 1	0.0	REAL	
Y2	Output variable 2	0.0	REAL	
Y3	Output variable 3	0.0	REAL	
Y4	Output variable 4	0.0	REAL	
Y5	Output variable 5	0.0	REAL	
Y6	Output variable 6	0.0	REAL	
Y7	Output variable 7	0.0	REAL	
Y8	Output variable 8	0.0	REAL	
YS	Control output	0	INT	

Configuration data

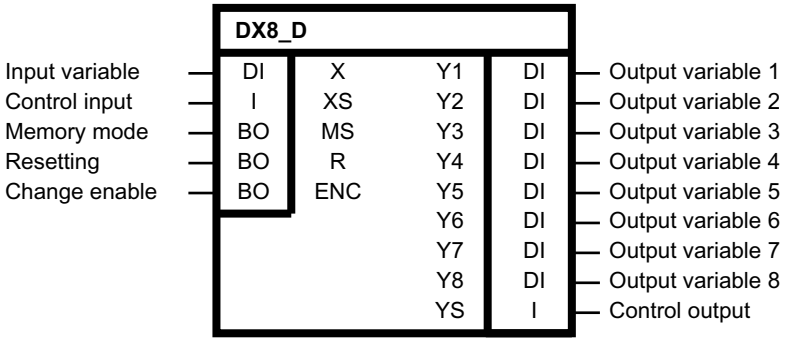
SIMOTION	✓
SINAMICS	✓

Can be loaded on-line	Yes
Special characteristics	-

8.1.4.14 DX8_D

Demultiplexer, eight outputs, cascadable (DOUBLE INTEGER type)

Symbol



Brief description

Block of the DOUBLE INTEGER type for demultiplex operation. This block is cascadable.

Method of operation

Depending on ENC, R, MS and XS = 1 to 8, the block switches through its input X to one of the eight selectable outputs Y1 to Y8 (example: XS = 3 means Y3 = X).

When XS = 0 or XS ≥ 9, none of the block inputs Y1 to Y8 is selected. Non-selected outputs are either set to zero or retain their previous value until the next change.

The following priority sequence applies for the control inputs:

ENC before R before MS

When ENC = 0, all outputs Y1 to Y8 remain unchanged, regardless of R and MS. When ENC = 1, outputs Y1 to Y8 are enabled for change. When R = 1, all outputs Y1 to Y8 receive the value 0, irrespective of MS. When MS = 0 (non-latching mode), all outputs Y1 to Y8 not selected by XS receive the value 0. When MS = 1 (storing mode), all outputs not selected by XS remain unchanged.

Truth table(s)

ENC	R	MS	XS	Outputs Y1 to Y8
0	*	*	*	The previous values are retained
1	1	*	*	Y1 to Y8 = 0

ENC	R	MS	XS	Outputs Y1 to Y8
1	0	0	1 <= XS <= 8	• Selected output = X • Output not selected = 0
1	0	0	XS = 0 or XS >= 9	Y1 to Y8 = 0
1	0	1	1 <= XS <= 8	• Selected output = X • Outputs not selected remain unchanged
1	0	1	XS = 0 or XS >= 9	All previous values remain unchanged

* Arbitrary

For XS = 0 to 8, YS = 0.When XS > 8: YS = XS - 8 (use for cascading).

Cascading

The block output YS must be connected to the block input XS of the following block.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
XS	Control input	0	INT	
MS	Memory mode	0	0/1	
R	Resetting	0	0/1	
ENC	Change enable	0	0/1	
Y1	Output variable 1	0	DINT	
Y2	Output variable 2	0	DINT	
Y3	Output variable 3	0	DINT	
Y4	Output variable 4	0	DINT	
Y5	Output variable 5	0	DINT	
Y6	Output variable 6	0	DINT	
Y7	Output variable 7	0	DINT	
Y8	Output variable 8	0	DINT	
YS	Control output	0	INT	

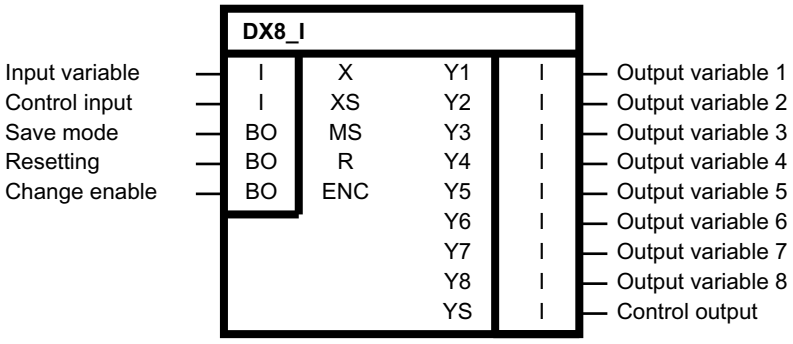
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.4.15 DX8_I

Demultiplexer, eight outputs, cascadable (INTEGER type)

Symbol



Brief description

Block of the INTEGER type for demultiplex operation. This block is cascadable.

Method of operation

Depending on ENC, R, MS and XS = 1 to 8, the block switches through its input X to one of the eight selectable outputs Y1 to Y8 (example: XS = 3 means Y3 = X).

When XS = 0 or XS >= 9, none of the block inputs Y1 to Y8 is selected. Non-selected outputs are either set to zero or retain their previous value until the next change.

The following priority sequence applies for the control inputs:

ENC before R before MS

When ENC = 0, all outputs Y1 to Y8 remain unchanged, regardless of R and MS.

When ENC = 1, outputs Y1 to Y8 are enabled for change.

When R = 1, all outputs Y1 to Y8 receive the value 0, irrespective of MS.

When MS = 0 (non-latching mode), all outputs Y1 to Y8 not selected by XS receive the value 0.

When MS = 1 (storing mode), all outputs not selected by XS remain unchanged.

Truth table(s)

ENC	R	MS	XS	Outputs Y1 to Y8
0	*	*	*	The previous values are retained
1	1	*	*	Y1 to Y8 = 0
1	0	0	1 <= XS <= 8	• Selected output = X • Output not selected = 0
1	0	0	XS = 0 or XS >= 9	Y1 to Y8 = 0

ENC	R	MS	XS	Outputs Y1 to Y8
1	0	1	1 <= XS <= 8	• Selected output = X • Outputs not selected remain unchanged
1	0	1	XS = 0 or XS >= 9	All previous values remain unchanged

* Arbitrary

For XS = 0 to 8, YS = 0.When XS > 8: YS = XS - 8 (use for cascading)

Cascading

The block output YS must be connected to the block input XS of the following block.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
XS	Control input	0	INT	
MS	Save mode	0	0/1	
R	Resetting	0	0/1	
ENC	Change enable	0	0/1	
Y1	Output variable 1	0	INT	
Y2	Output variable 2	0	INT	
Y3	Output variable 3	0	INT	
Y4	Output variable 4	0	INT	
Y5	Output variable 5	0	INT	
Y6	Output variable 6	0	INT	
Y7	Output variable 7	0	INT	
Y8	Output variable 8	0	INT	
YS	Control output	0	INT	

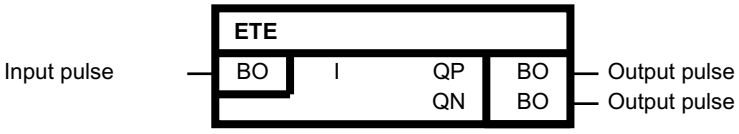
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.16 ETE

Edge evaluator (BOOL type)

Symbol



Brief description

Edge evaluation

Method of operation

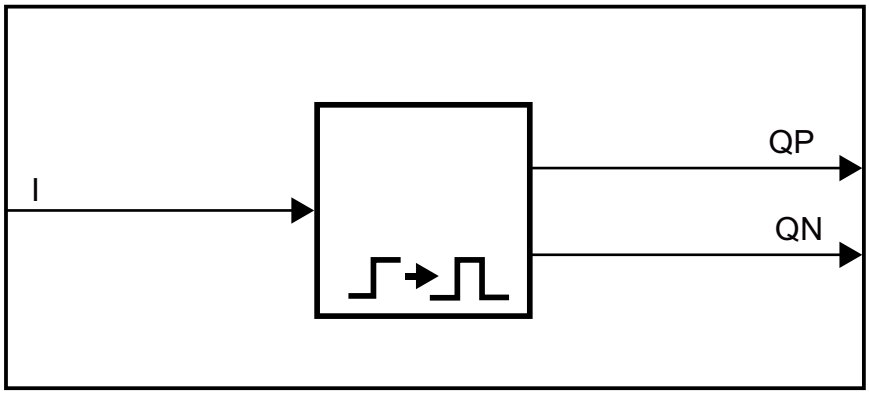
This block detects a signal change at input I. With a positive edge (0→1) at input I, output QP = 1 is set for scan time TA.

With a negative edge (1→0) at input I, output QN = 1 is set for scan time TA.

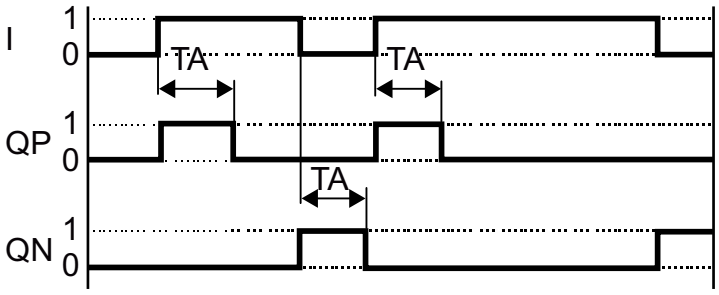
Initialization

The initialization defines the start value for the first cyclic pass. If input I receives the value 1 during initialization of an upstream block, the block cannot detect a positive edge during the first cyclic pass. If input I receives the value 0 during initialization of an upstream block, the block cannot detect a negative edge during the first cyclic pass.

Block diagram



Time diagram



Output pulses QP and QN as a function of scan time TA and input pulse I

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
QP	Output pulse	0	0/1	
QN	Output pulse	0	0/1	

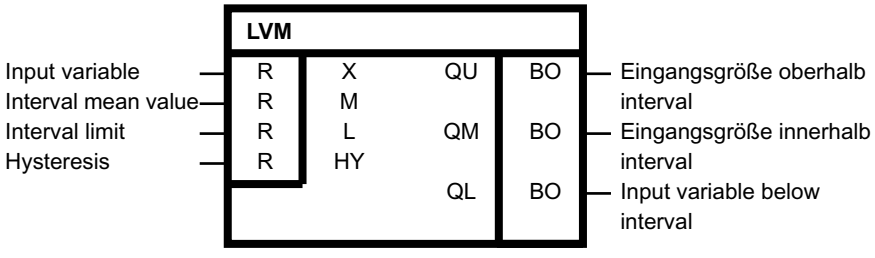
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.17 LVM

Double-sided limit monitor with hysteresis (BOOL type)

Symbol



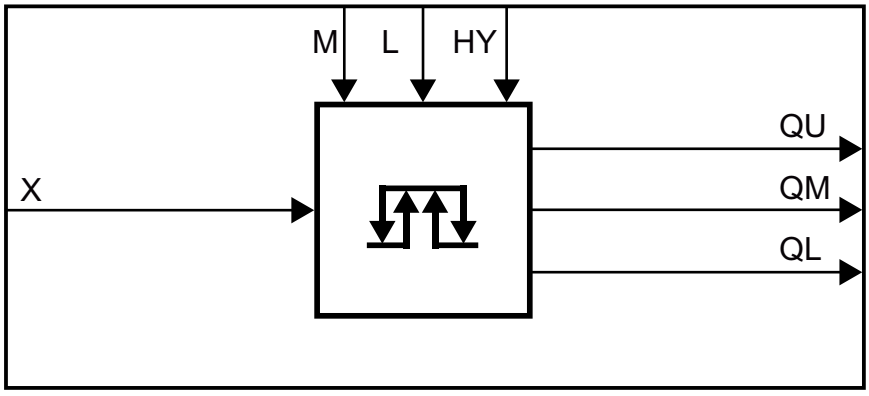
Brief description

- The block of the BOOL type monitors an input variable through comparison with selectable reference variables.
- Can be used for monitoring setpoints, actual values and measured values, as well as for the suppression of frequent switching (chatter).
- The block provides a window discriminator function.

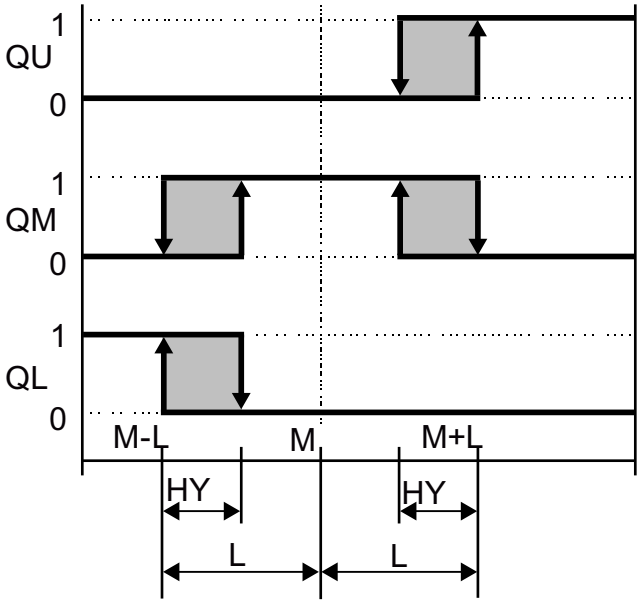
Method of operation

The block calculates an intermediate value based on a transmission characteristic (see Transmission characteristic) with hysteresis. This intermediate value is compared with the interval limits, and the result is output at outputs QU, QM, and QL. The transfer characteristic is configured with the values for the mean value M, the interval limit L and the hysteresis HY.

Block diagram



Transfer function



Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
M	Interval mean value	0.0	REAL	
L	Interval limit	0.0	REAL	
HY	Hysteresis	0.0	REAL	
QU	Input variable above the interval	0	0/1	
QM	Input variable above the interval	0	0/1	
QL	Input variable above the interval	0	0/1	

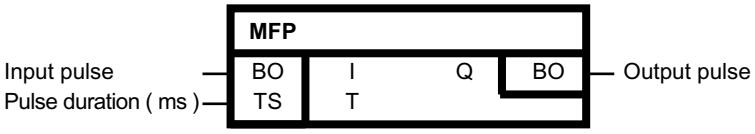
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.18 MFP

Pulse generator (BOOL type)

Symbol



Brief description

- Timer for generating a pulse with a fixed duration.
- Used as a pulse-contracting or pulse-stretching monoflop.

Method of operation

The rising edge of a pulse at input I sets output Q to 1 for the pulse duration T. The pulse generator cannot be retriggered. When T=0, a pulse duration of 1 cycle is active.

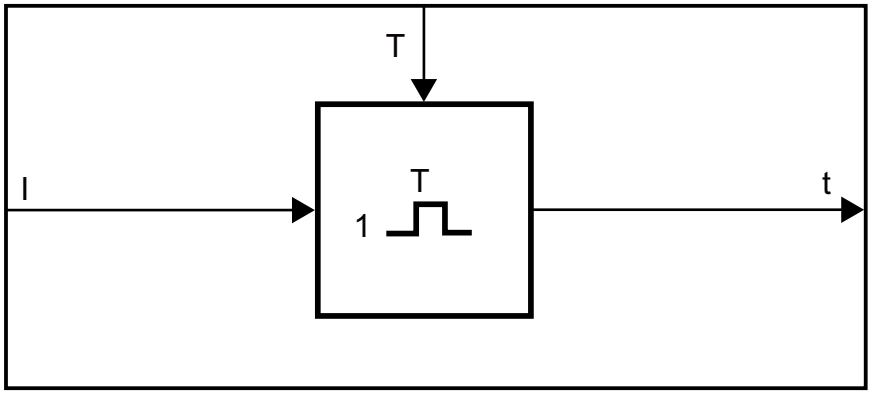
Initialization

The initialization defines the start value for the first cyclic pass.

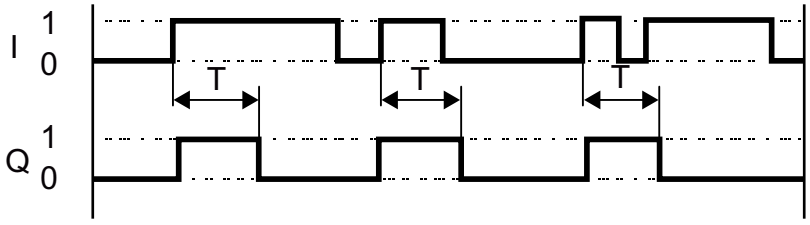
If input I receives the value 1 during initialization from the upstream block output, the block cannot detect a positive edge during the first cyclic pass.

If output Q receives the default value 1, output Q = 1 is set after initialization for the pulse duration T.

Block diagram



Time diagram



Output pulse Q as a function of pulse duration T and input pulse I

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
T	Pulse duration (ms)	0	SDTIME	
Q	Output pulse	0	0/1	

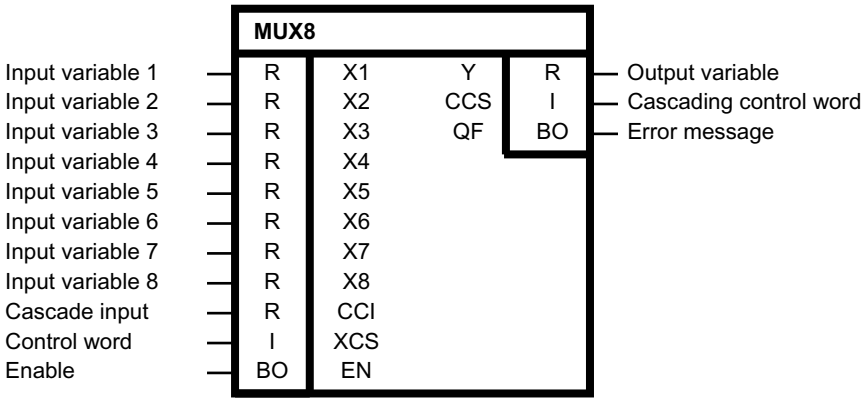
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.19 MUX8

Multiplexer, cascadable (REAL type)

Symbol



Brief description

Block of the REAL type for 8-fold multiplex operation. This block is cascadable.

Method of operation

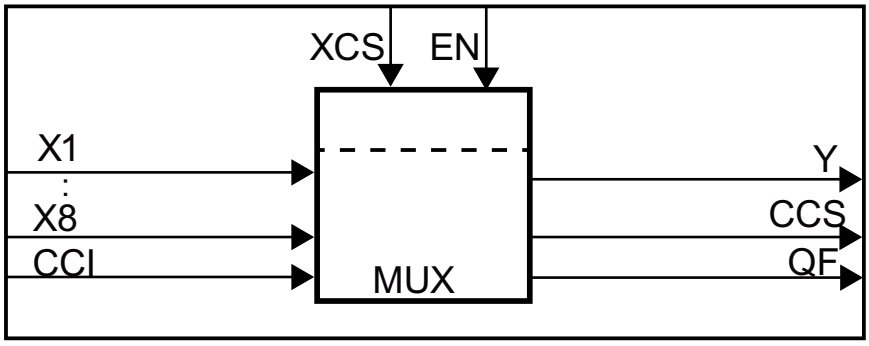
This block outputs the value of the cascading input CCI at output Y as long as the enable input EN is logic 0.

When EN is logic 1, one of the input variables X1 to X8 is switched through to output Y as long as the 16-bit control word XCS assumes a value between 1 and 8.

If the value of the input XCS > 8, output Y assumes the value 0, and output QF becomes logic 1. The cascading control word assumes the value CCS = XCS-8, see truth table.

The outputs Y, CCS, and QF can be used to cascade the blocks. In this case, output Y of the first block is connected to input CCI of the downstream multiplexer, output CCS to the following XCS, and output QF to the following input EN.

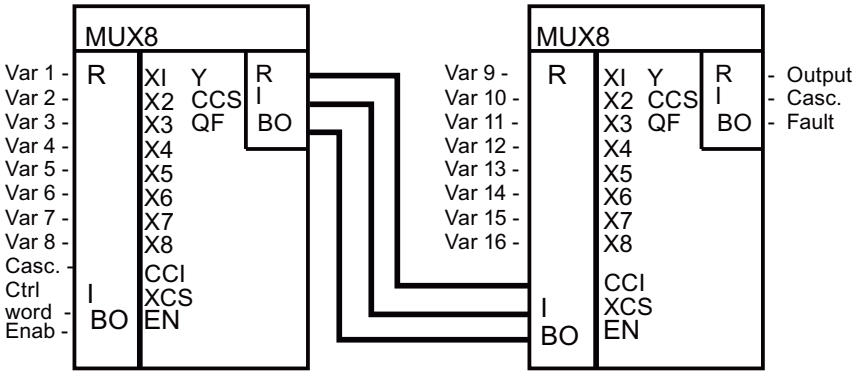
Block diagram



Truth table(s)

EN	XCS	Y	CSS	QF
0	Any	CCI	0	0
1	0	0	0	1
1	1	X1	0	0
1	2	X2	0	0
1	3	X3	0	0
1	4	X4	0	0
1	5	X5	0	0
1	6	X6	0	0
1	7	X7	0	0
1	8	X8	0	0
1	>8	0	XCS-8	1

Cascading



Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0.0	REAL	
X2	Input variable 2	0.0	REAL	
X3	Input variable 3	0.0	REAL	
X4	Input variable 4	0.0	REAL	
X5	Input variable 5	0.0	REAL	
X6	Input variable 6	0.0	REAL	
X7	Input variable 7	0.0	REAL	
X8	Input variable 8	0.0	REAL	
CCI	Cascade input	0.0	REAL	
XCS	Control word	0	0...32767	
EN	Enable	0	0/1	
Y	Output variable	0.0	REAL	
CCS	Cascading control word	0	0...32767	
QF	Error message	0	0/1	

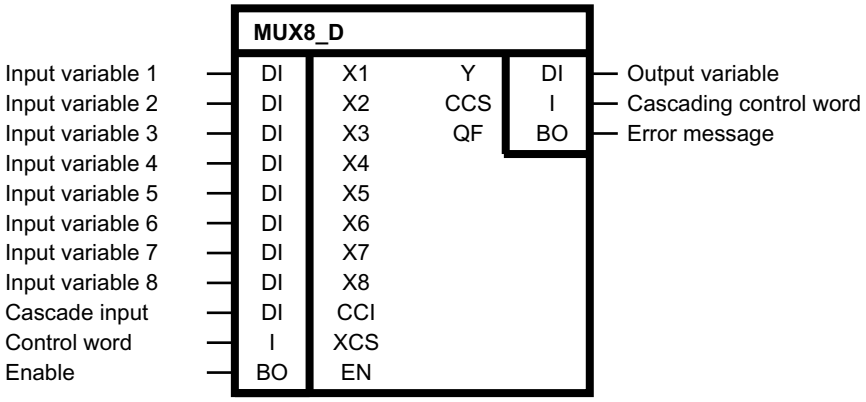
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.20 MUX8_D

Multiplexer, cascadable (DOUBLE-INTEGER type)

Symbol



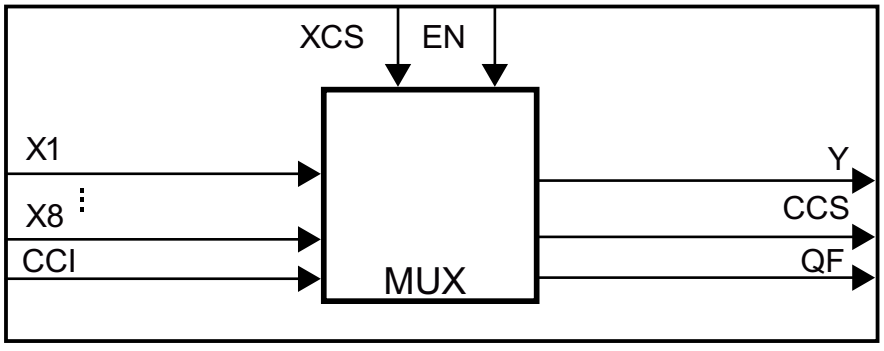
Brief description

Block of the DOUBLE INTEGER type for 8-fold multiplex operation. This block is cascadable.

Method of operation

This block outputs the value of the cascading input CCI at output Y as long as the enable input EN is logic 0. When EN is logic 1, one of the input variables X1 to X8 is switched through to output Y as long as the 16-bit control word XCS assumes a value between 1 and 8. If the value of the input XCS > 8, output Y assumes the value 0, and output QF becomes logic 1. The cascading control word assumes the value CCS = XCS-8, see truth table. The outputs Y, CCS, and QF can be used to cascade the blocks. In this case, output Y of the first block is connected to input CCI of the downstream multiplexer, output CCS to the following XCS, and output QF to the following input EN.

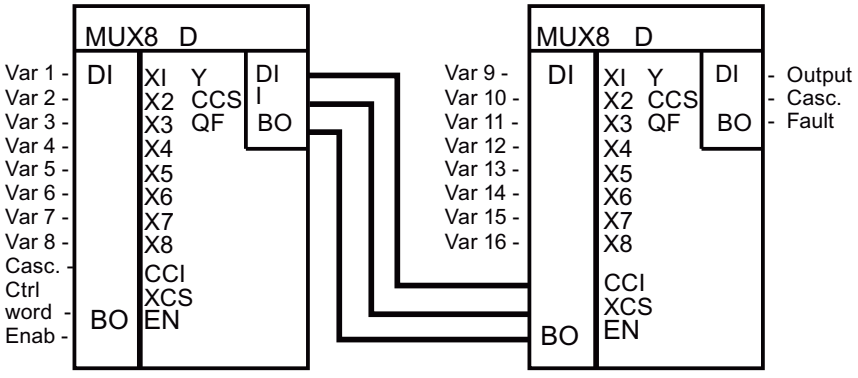
Block diagram



Truth table(s)

EN	XCS	Y	CSS	QF
0	Any	CCI	0	0
1	0	0	0	1
1	1	X1	0	0
1	2	X2	0	0
1	3	X3	0	0
1	4	X4	0	0
1	5	X5	0	0
1	6	X6	0	0
1	7	X7	0	0
1	8	X8	0	0
1	>8	0	XCS-8	1

Cascading



Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	DINT	
X2	Input variable 2	0	DINT	
X3	Input variable 3	0	DINT	
X4	Input variable 4	0	DINT	
X5	Input variable 5	0	DINT	
X6	Input variable 6	0	DINT	
X7	Input variable 7	0	DINT	
X8	Input variable 8	0	DINT	
CCI	Cascade input	0	DINT	
XCS	Control word	0	0...32767	
EN	Enable	0	0/1	
Y	Output variable	0	DINT	

Block connection	Description	Default	Value range	Attrib-utes
CCS	Cascading control word	0	0...32767	
QF	Error message	0	0/1	

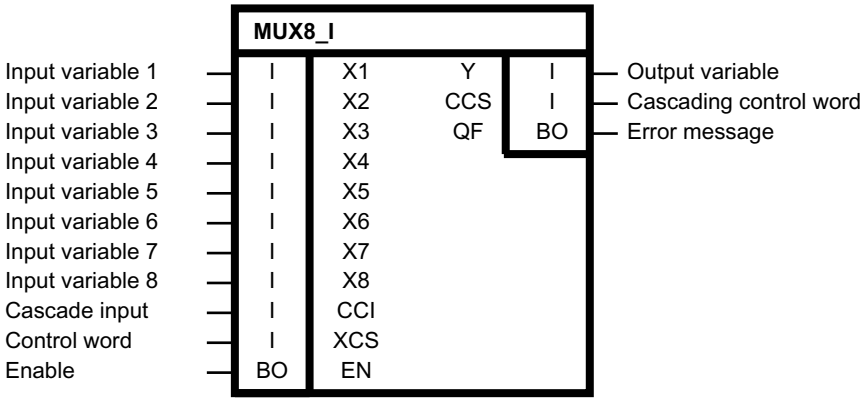
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	Yes
Special character-istics	-

8.1.4.21 MUX8_I

Multiplexer, cascadable (INTEGER type)

Symbol



Brief description

Block of the INTEGER type for 8-fold multiplex operation. This block is cascadable.

Method of operation

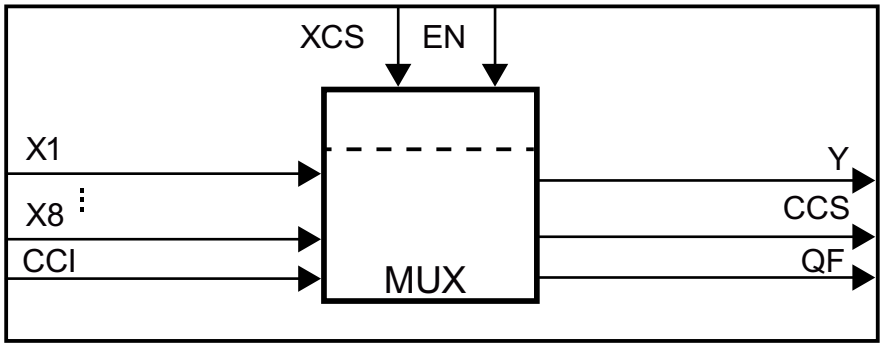
This block outputs the value of the cascading input CCI at output Y as long as the enable input EN is logic 0.

When EN is logic 1, one of the input variables X1 to X8 is switched through to output Y as long as the 16-bit control word XCS assumes a value between 1 and 8.

If the value of the input XCS > 8, output Y assumes the value 0, and output QF becomes logic 1. The cascading control word assumes the value CCS = XCS-8, see truth table.

The outputs Y, CCS, and QF can be used to cascade the blocks. In this case, output Y of the first block is connected to input CCI of the downstream multiplexer, output CCS to the following XCS, and output QF to the following input EN.

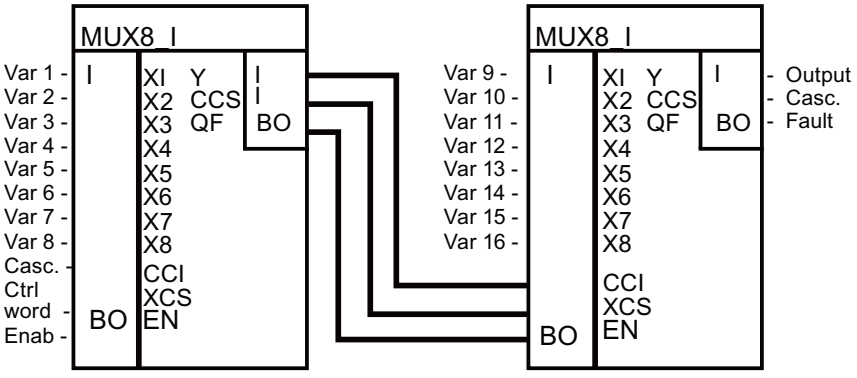
Block diagram



Truth table(s)

EN	XCS	Y	CSS	QF
0	Any	CCI	0	0
1	0	0	0	1
1	1	X1	0	0
1	2	X2	0	0
1	3	X3	0	0
1	4	X4	0	0
1	5	X5	0	0
1	6	X6	0	0
1	7	X7	0	0
1	8	X8	0	0
1	>8	0	XCS-8	1

Cascading



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0	INT	
X2	Input variable 2	0	INT	
X3	Input variable 3	0	INT	
X4	Input variable 4	0	INT	
X5	Input variable 5	0	INT	
X6	Input variable 6	0	INT	
X7	Input variable 7	0	INT	
X8	Input variable 8	0	INT	
CCI	Cascade input	0	INT	
XCS	Control word	0	0...32767	
EN	Enable	0	0/1	
Y	Output variable	0	INT	
CCS	Cascading control word	0	0...32767	
QF	Error message	0	0/1	

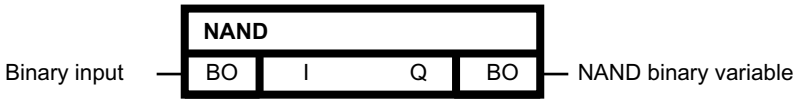
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.4.22 NAND

Logic AND operation (BOOL type)

Symbol



Brief description

NAND block with up to four inputs of the BOOL type

Method of operation

The block combines the binary values at the inputs I 1-4 to a logic AND, inverts the result and outputs it at binary output Q.

$$Q = \overline{I_{01} \wedge \ldots \wedge I_{04}}$$

Output Q = 0, when the value 1 is present at all generic inputs I1 to I4. In all other cases, output Q = 1.

Truth table(s)

Input				Output
I01	I02	I03	I04	Q
0	*	*	*	1
*	0	*	*	1
*	*	0	*	1
*	*	*	0	1
1	1	1	1	0

* Arbitrary

Block connections

Block connection	Description	Default	Value range	Attributes
I	Binary input	1	0/1	
Q	NAND binary variable	0	0/1	

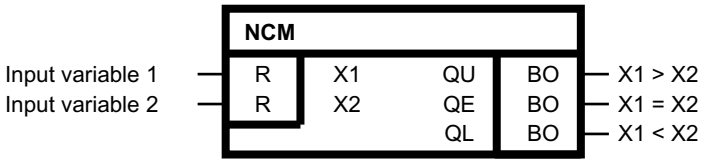
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	I comprises up to four inputs (I1 to I4)

8.1.4.23 NCM

Numeric comparator (REAL type)

Symbol



Brief description

Block for comparison operations of two numeric variables of the REAL type

Method of operation

The input variables X1 and X2 are compared and one of binary outputs QU, QE, or QL is set depending on the result of the comparison operation.

Truth table(s)

Comparison of input variables	Output signals	Output signals Y	Output signals Y
	QU	QE	QL
X1 > X2	1	0	0
X1 = X2	0	1	0
X1 < X2	0	0	1

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0	REAL	
X2	Input variable 2	0	REAL	
QU	X1 > X2	0	0/1	
QE	X1 = X2	1	0/1	
QL	X1 < X2	0	0/1	

Configuration data

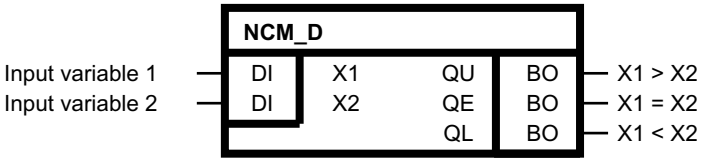
SIMOTION	✓
SINAMICS	✓

Can be loaded online	Yes
Special characteristics	If the execution group of the DCC chart is set to "Do not calculate", the default value is set at the output.

8.1.4.24 NCM_D

Numeric comparator (DOUBLE INTEGER type)

Symbol



Brief description

Block for comparison operations of two numeric variables of the DOUBLE INTEGER type

Method of operation

The input variables X1 and X2 are compared and one of binary outputs QU, QE, or QL is set depending on the result of the comparison operation.

Truth table(s)

Comparison of input variables	Description	Default	Value range
	QU	QE	QL
X1 > X2	1	0	0
X1 = X2	0	1	0
X1 < X2	0	0	1

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	DINT	
X2	Input variable 2	0	DINT	
QU	X1 > X2	0	0/1	
QE	X1 = X2	1	0/1	
QL	X1 < X2	0	0/1	

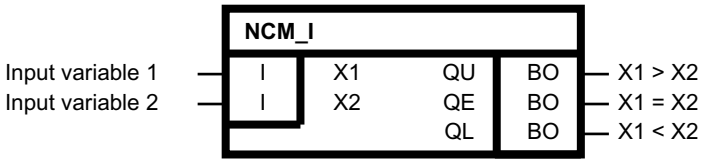
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	If the execution group of the DCC chart is set to "Do not calculate", the default value is set at the output.

8.1.4.25 NCM_I

Numeric comparator (INTEGER type)

Symbol



Brief description

Block for comparison operations of two numeric variables of the INTEGER type

Method of operation

The input variables X1 and X2 are compared and one of binary outputs QU, QE, or QL is set depending on the result of the comparison operation.

Truth table(s)

Comparison of input variables	Output signals	Output signals Y	Output signals Y
	QU	QE	QL
X1 > X2	1	0	0
X1 = X2	0	1	0
X1 < X2	0	0	1

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	INT	
X2	Input variable 2	0	INT	
QU	X1 > X2	0	0/1	

Block connection	Description	Default	Value range	Attributes
QE	X1 = X2	1	0/1	
QL	X1 < X2	0	0/1	

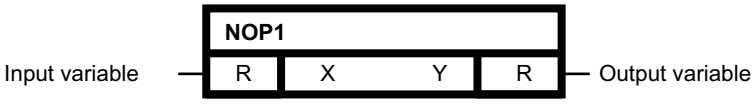
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	If the execution group of the DCC chart is set to "Do not calculate", the default value is set at the output.

8.1.4.26 NOP1

Dummy block (REAL type)

Symbol



Brief description

The block of the REAL type is used as dummy block (No Operation). It is used to provide a constant value for several blocks.

Method of operation

The block outputs the value present at input X without change at output Y. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0.0	REAL	

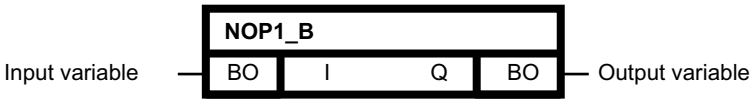
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.27 NOP1_B

Dummy block (BOOL type)

Symbol



Brief description

The block of the BOOL type is used as dummy block (No Operation). It is used to provide a constant value for several blocks.

Method of operation

The block outputs the value present at input I without change at output Q. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input variable	0	0/1	
Q	Output variable	0	0/1	

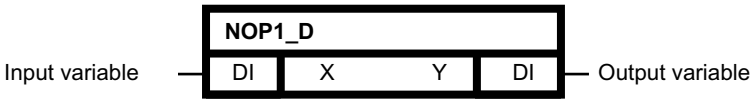
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.28 NOP1_D

Dummy block (DOUBLE INTEGER type)

Symbol



Brief description

The block of the DOUBLE INTEGER type is used as dummy block (No Operation). It is used to provide a constant value for several blocks.

Method of operation

The block outputs the value present at input X without change at output Y. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Output variable	0	DINT	

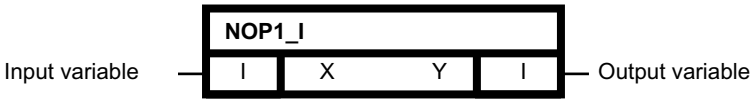
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.29 NOP1_I

Dummy block (INT type)

Symbol



Brief description

The block of the INT type is used as dummy block (No Operation). It is used to provide a constant value for several blocks.

Method of operation

The block outputs the value present at input X without change at output Y. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
Y	Output variable	0	INT	

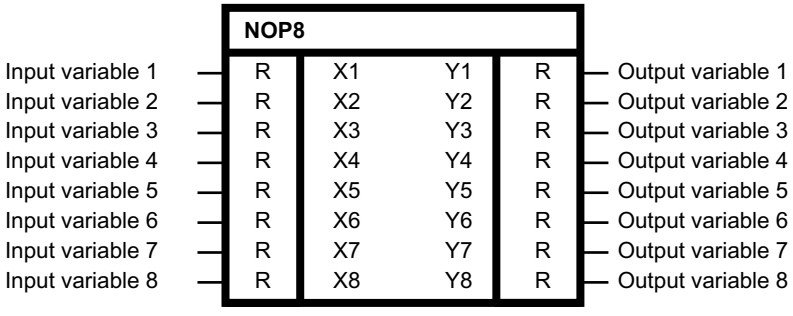
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.30 NOP8

Dummy block (REAL type)

Symbol



Brief description

The block of the REAL type is used as dummy block (No Operation). It is used to provide up to eight constant values for several blocks.

Method of operation

The block outputs the values present at inputs X1-X8 without change at outputs Y1 to Y8. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0.0	REAL	
X2	Input variable 2	0.0	REAL	
X3	Input variable 3	0.0	REAL	
X4	Input variable 4	0.0	REAL	
X5	Input variable 5	0.0	REAL	
X6	Input variable 6	0.0	REAL	
X7	Input variable 7	0.0	REAL	
X8	Input variable 8	0.0	REAL	
Y1	Output variable 1	0.0	REAL	
Y2	Output variable 2	0.0	REAL	
Y3	Output variable 3	0.0	REAL	
Y4	Output variable 4	0.0	REAL	
Y5	Output variable 5	0.0	REAL	
Y6	Output variable 6	0.0	REAL	
Y7	Output variable 7	0.0	REAL	
Y8	Output variable 8	0.0	REAL	

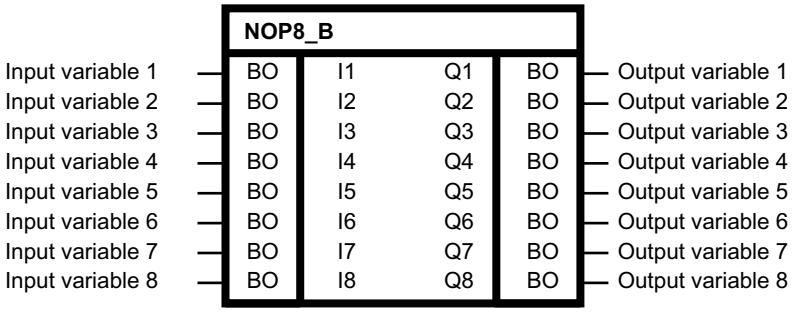
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.4.31 NOP8_B

Dummy block (BOOL type)

Symbol



Brief description

The block of the BOOL type is used as dummy block (No Operation). It is used to provide up to eight constant values for several blocks.

Method of operation

The block outputs the values present at inputs I1-I8 without change at outputs Q1 to Q8. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
I1	Input variable 1	0	0/1	
I2	Input variable 2	0	0/1	
I3	Input variable 3	0	0/1	
I4	Input variable 4	0	0/1	
I5	Input variable 5	0	0/1	
I6	Input variable 6	0	0/1	
I7	Input variable 7	0	0/1	
I8	Input variable 8	0	0/1	
Q1	Output variable 1	0	0/1	
Q2	Output variable 2	0	0/1	
Q3	Output variable 3	0	0/1	
Q4	Output variable 4	0	0/1	
Q5	Output variable 5	0	0/1	
Q6	Output variable 6	0	0/1	
Q7	Output variable 7	0	0/1	
Q8	Output variable 8	0	0/1	

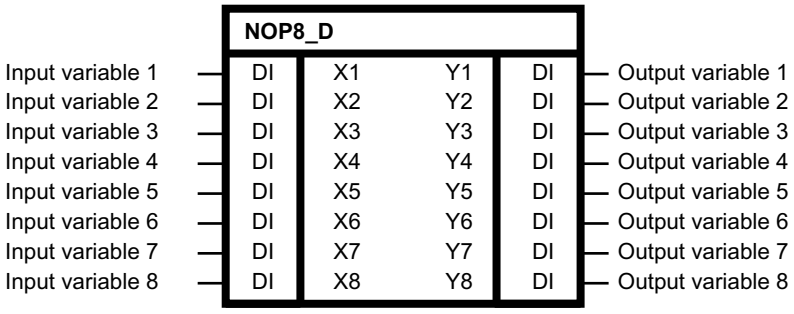
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.32 NOP8_D

Dummy block (DOUBLE INTEGER type)

Symbol



Brief description

The block of the DOUBLE INTEGER type is used as dummy block (No Operation). It is used to provide up to eight constant values for several blocks.

Method of operation

The block outputs the values present at inputs X1-X8 without change at outputs Y1 to Y8. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	DINT	
X2	Input variable 2	0	DINT	
X3	Input variable 3	0	DINT	
X4	Input variable 4	0	DINT	
X5	Input variable 5	0	DINT	
X6	Input variable 6	0	DINT	
X7	Input variable 7	0	DINT	

Block connection	Description	Default	Value range	Attributes
X8	Input variable 8	0	DINT	
Y1	Output variable 1	0	DINT	
Y2	Output variable 2	0	DINT	
Y3	Output variable 3	0	DINT	
Y4	Output variable 4	0	DINT	
Y5	Output variable 5	0	DINT	
Y6	Output variable 6	0	DINT	
Y7	Output variable 7	0	DINT	
Y8	Output variable 8	0	DINT	

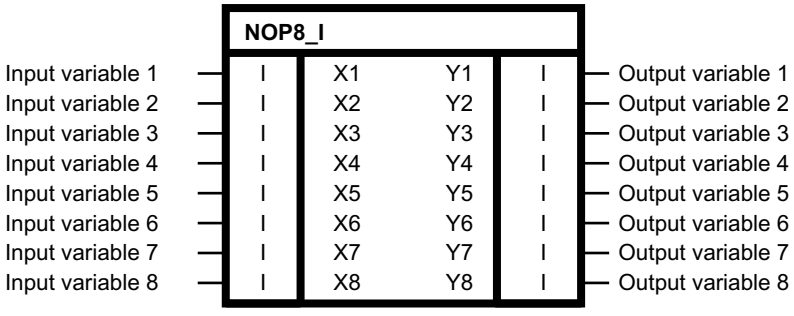
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.33 NOP8_I

Dummy block (INTEGER type)

Symbol



Brief description

The block of the INTEGER type is used as dummy block (No Operation). It is used to provide up to eight constant values for several blocks.

Method of operation

The block outputs the values present at inputs X1-X8 without change at outputs Y1 to Y8. This is a so-called DUMMY or No Operation block.

Block connections

Block connection	Description	Default	Value range	Attributes
X1	Input variable 1	0	INT	
X2	Input variable 2	0	INT	
X3	Input variable 3	0	INT	
X4	Input variable 4	0	INT	
X5	Input variable 5	0	INT	
X6	Input variable 6	0	INT	
X7	Input variable 7	0	INT	
X8	Input variable 8	0	INT	
Y1	Output variable 1	0	INT	
Y2	Output variable 2	0	INT	
Y3	Output variable 3	0	INT	
Y4	Output variable 4	0	INT	
Y5	Output variable 5	0	INT	
Y6	Output variable 6	0	INT	
Y7	Output variable 7	0	INT	
Y8	Output variable 8	0	INT	

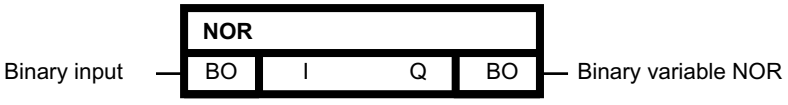
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.34 NOR

Logic OR operation (BOOL type)

Symbol



Brief description

NOR block with up to four inputs of the BOOL type

Method of operation

The block combines the binary values at the inputs I 1-4 to a logic OR, inverts the result and outputs it at binary output Q.

$$Q = \overline{I_{01} \vee \ldots \vee I_{04}}$$

Output Q = 1, when the value 0 is present at all inputs I1 to I4. In all other cases, output Q = 0.

Truth table(s)

Input				Output
I01	I02	I03	I04	Q
1	*	*	*	0
*	1	*	*	0
*	*	1	*	0
*	*	*	1	0
0	0	0	0	1

* Arbitrary

Block connections

Block connection	Description	Default	Value range	Attrib-utes
I	Binary input	0	0/1	
Q	Binary variable NOR	1	0/1	

Configuration data

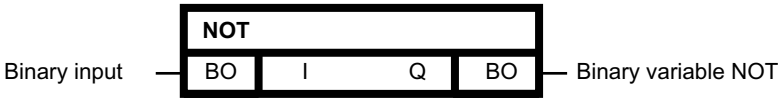
SIMOTION	✓
SINAMICS	✓

Can be loaded online	Yes
Special characteristics	I comprises up to four connections (I1 to I4)

8.1.4.35 NOT

Inverter (BOOL type)

Symbol



Brief description

Inverter of the BOOL type

Method of operation

The block inverts the binary variable at input I and outputs the result at output Q.

$$Q = \overline{I}$$

Truth table(s)

Input 1	Output Q
1	0

Block connections

Block connection	Description	Default	Value range	Attributes
I	Binary input	0	0/1	
Q	Binary variable NOT	1	0/1	

Configuration data

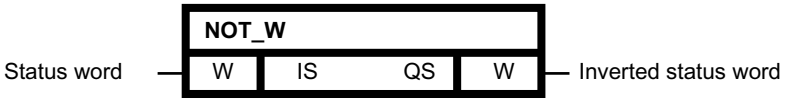
SIMOTION	✓
SINAMICS	✓

Can be loaded online	Yes
Special characteristics	-

8.1.4.36 NOT_W

Status word inverter (WORD type)

Symbol



Brief description

- Inverter for WORD-type status word
- One's complement formation of IS

Method of operation

16 binary states are combined in a status word.

The block inverts the status word IS bit-by-bit and outputs it at output QS.

The following applies for the bit k of the inverted status word:

$$QS_k = \overline{IS_k}$$

Complement formation

Example: IS = 15 -> QS = -16

Status word

IS

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Inverted status word

QS

1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IS	Status word	16#0000	WORD	
QS	Inverted status word	16#FFFF	WORD	

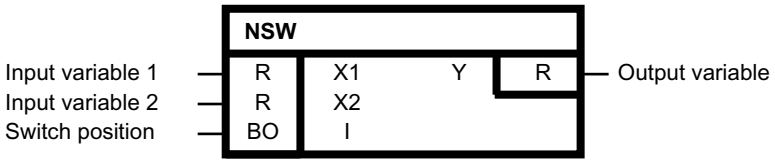
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded on-line	Yes
Special character-istics	-

8.1.4.37 NSW

Numeric change-over switch (REAL type)

Symbol



Brief description

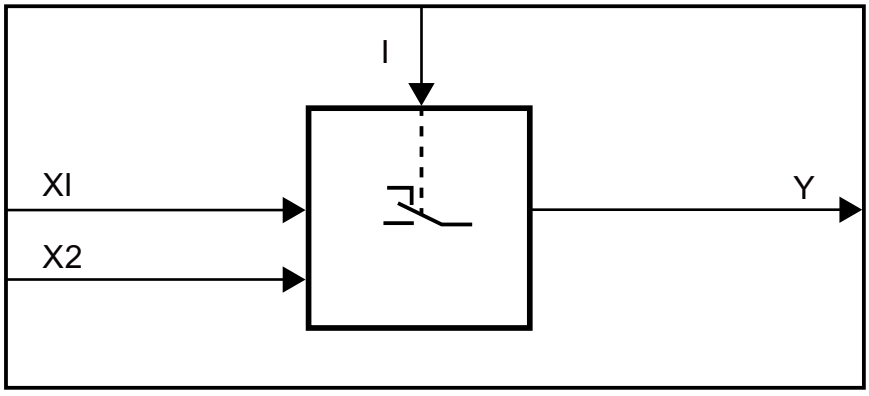
The block switches one of two numeric input variables (REAL type) to the output.

Method of operation

If input I = 0, then X1 is given to output Y.

If input I = 1, then X2 is given to output Y.

Block diagram



Truth table(s)

Switch position 1	Output variable Y
0	Y = X1
1	Y = X2

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0	REAL	
X2	Input variable 2	0	REAL	
I	Switch position	0	0/1	
Y	Output variable	0	REAL	

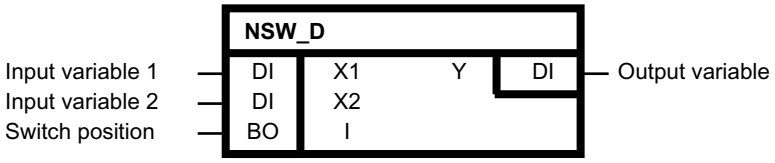
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.4.38 NSW_D

Numeric change-over switch (DOUBLE INTEGER type)

Symbol



Brief description

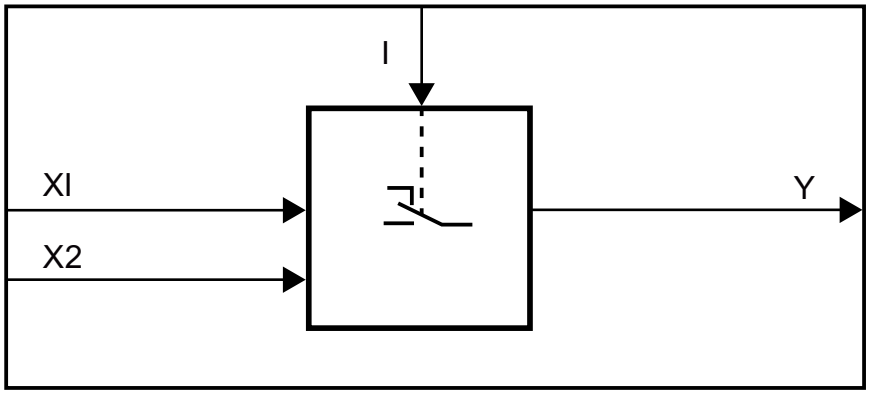
The block switches one of two numeric input variables (DOUBLE INTEGER type) to the output

Method of operation

If input I = 0, then X1 is given to output Y.

If input I = 1, then X2 is given to output Y.

Block diagram



Truth table(s)

Switch position 1	Output variable Y
0	Y = X1
1	Y = X2

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0	DINT	
X2	Input variable 2	0	DINT	
I	Switch position	0	0/1	
Y	Output variable	0	DINT	

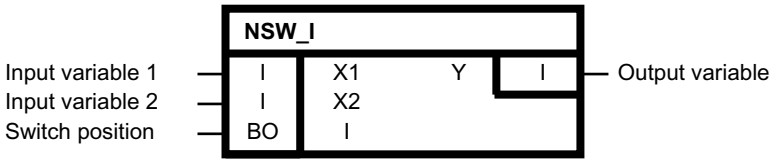
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.4.39 NSW_I

Numeric change-over switch (INTEGER type)

Symbol



Brief description

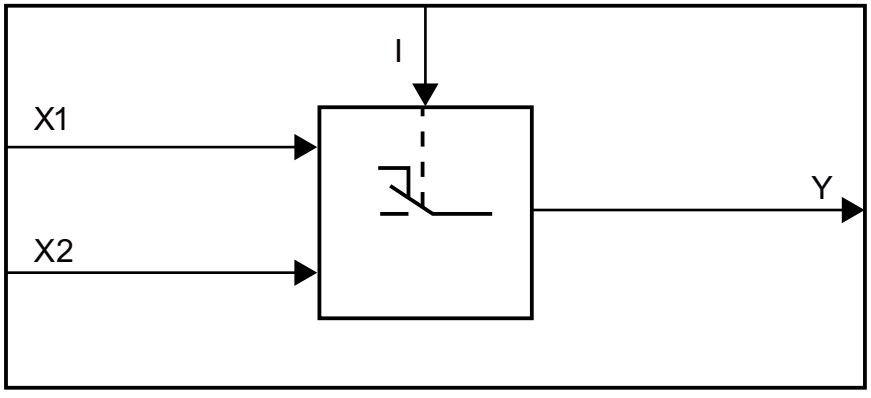
The block switches one of two numeric input variables (INTEGER type) to the output.

Method of operation

If input I = 0, then X1 is given to output Y.

If input I = 1, then X2 is given to output Y.

Block diagram



Truth table(s)

Switch position 1	Output variable Y
0	Y = X1
1	Y = X2

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X1	Input variable 1	0	INT	
X2	Input variable 2	0	INT	
I	Switch position	0	0/1	
Y	Output variable	0	INT	

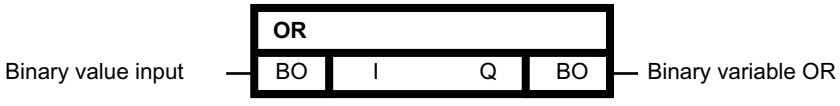
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.4.40 OR

Logic OR operation (BOOL type)

Symbol



Brief description

OR block with up to four inputs of the BOOL type

Method of operation

The block combines the binary values at the inputs I 1-4 to a logic OR (disjunction) and outputs the result at its binary output Q.

$Q = I_{01} \vee \ldots \vee I_{04}$

Output Q = 0, when the value 0 is present at all inputs I1 to I4. In all other cases, output Q = 1.

Truth table(s)

Input				Output
I01	I02	I03	I04	Q
1	*	*	*	1
*	1	*	*	1
*	*	1	*	1
*	*	*	1	1
0	0	0	0	0

* Arbitrary

Block connections

Block connection	Description	Default	Value range	Attributes
I	Binary value input	0	0/1	
Q	Binary variable OR	0	0/1	

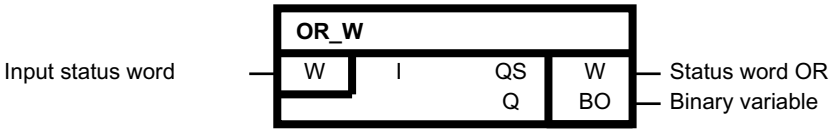
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	I comprises up to four connections (I1 to I4)

8.1.4.41 OR_W

Logic OR operation (WORD type)

Symbol



Brief description

OR block with up to four inputs of the WORD type

Method of operation

16 binary states are combined in a status word.

The block combines the status words I1 to I4 bit-by-bit according to the logic OR function.

The result is given to the block output QS (status word OR).

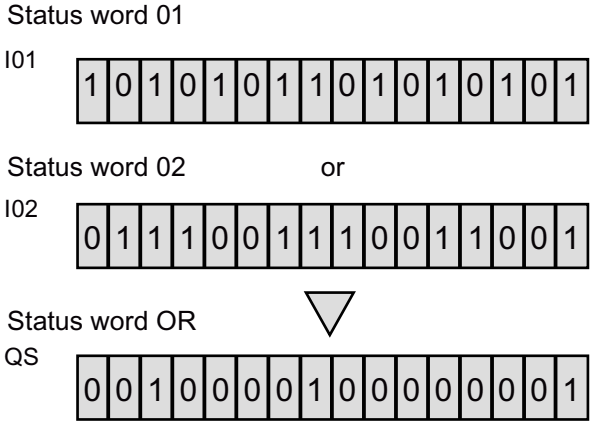
The following applies for bit k of status word OR:

$QS_k = I02_k \vee I02_k, \quad k = 1...16$

A bit of the OR status word is equal to 1 when at least one of the equivalent bits on the block inputs I1 to I4 is equal to 1.

The binary output Q is 1 if at least one bit of the status word OR is equal to 1.

Following state diagram (for 3 inputs)



Block connections

Block connection	Description	Default	Value range	Attrib-utes
I	Input status word	16#0000	WORD	
QS	Status word OR	16#0000	WORD	
Q	Binary variable	0	0/1	

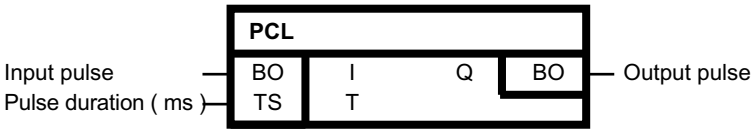
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded on-line	Yes
Special characteristics	I comprises up to four connections (I1 to I4)

8.1.4.42 PCL

Pulse shortener (BOOL type)

Symbol



Brief description

Timer for limiting the pulse duration

Method of operation

The rising edge of a pulse at input I sets output Q to 1. Output Q becomes 0 when input I = 0 or pulse duration T has expired. When T=0, a pulse duration of 1 cycle is active.

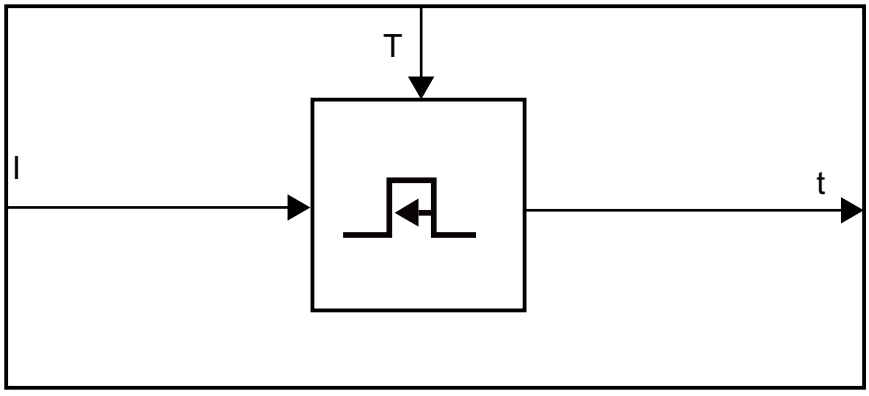
Initialization

The initialization defines the start value for the first cyclic pass.

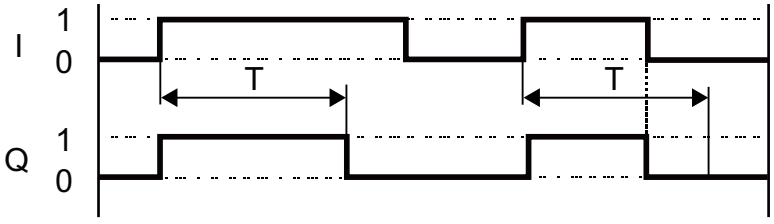
If input I receives the value 1 during initialization from the upstream block output, the block cannot detect a positive edge during the first cyclic pass.

If output Q receives the default value 1, output Q = 1 is set after initialization for the pulse duration T.

Block diagram



Time diagram



Output pulse Q as a function of pulse duration T and input pulse I

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
T	Pulse duration (ms)	0	SDTIME	
Q	Output pulse	0	0/1	

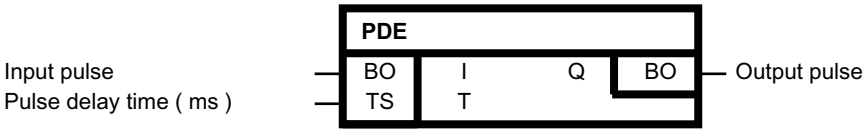
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.43 PDE

Switch-on delay (BOOL type)

Symbol



Brief description

BOOL-type timer with on-delay

Method of operation

The pulse delay time at the input T is taken over with the rising edge at input I. After this time has elapsed, output Q is set to 1.

Output Q becomes 0 when I = 0.

If the duration of input pulse I is less than pulse delay time T, then Q remains at 0.

If time T is so long that the maximum value that can be displayed internally (T/ta as 32-bit value, where ta = sampling time) is exceeded, the maximum value is set (e.g. when ta = 1 ms, approx. 50 days).

When T=0, a pulse delay time of 1 cycle is active.

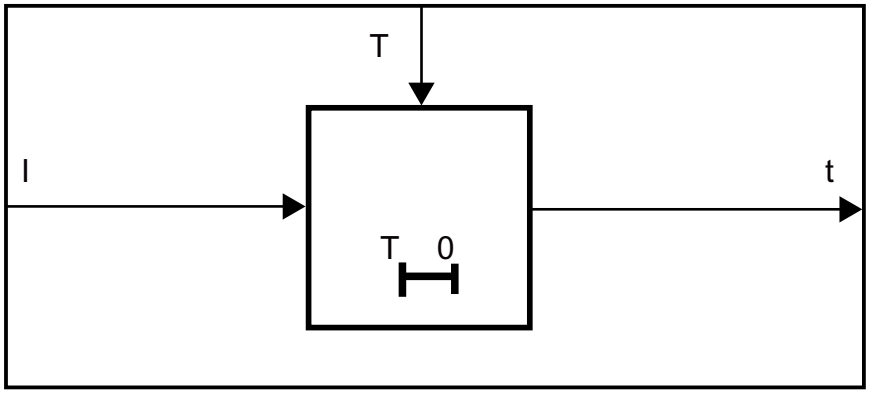
Method of operation

The initialization defines the start value for the first cyclic pass.

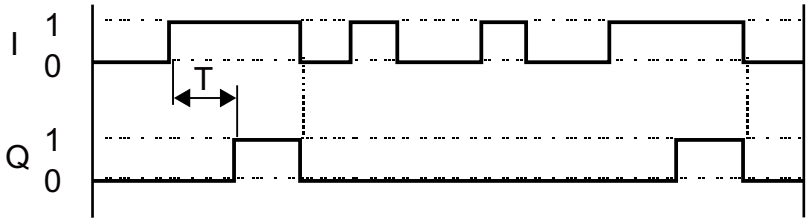
If input I receives the value 1 during initialization from the upstream block output, the block cannot detect a positive edge during the first cyclic pass. The pulse delay time T is therefore not taken over in the first cyclic pass with I = 1, the specified time from the initialization remains effective.

If output Q receives a value of 1 during initialization, then output Q = 1 is set immediately after initialization when I = 1.

Block diagram



Time diagram



Output pulse Q as a function of pulse duration T and input pulse I

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
T	Pulse delay time (ms)	0	SDTIME	
Q	Output pulse	0	0/1	

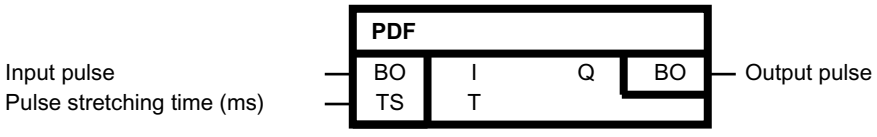
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.44 PDF

Switch-off delay (BOOL type)

Symbol



Brief description

BOOL-type timer with off-delay

Method of operation

The falling edge of a pulse at block input I resets output Q to 0 after pulse stretching time T.

Output Q becomes 1 when I = 1.

Output Q becomes 0 when input pulse I = 0 and the off-delay time T has expired.

If input I is reset to 1 before time T expires, then output Q remains at 1.

When T=0, a pulse stretching time of 1 cycle is active.

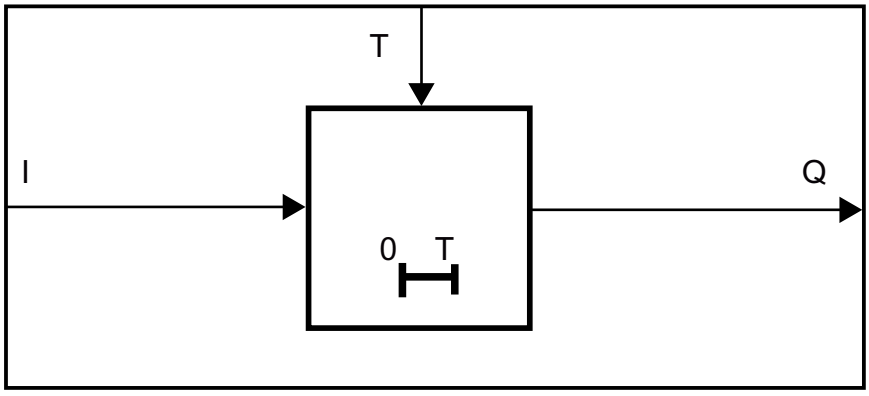
Initialization

The initialization defines the start value for the first cyclic pass.

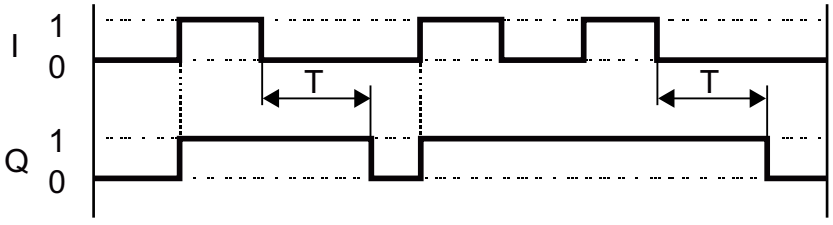
If input I receives the value 1 during initialization from the upstream block output, the block cannot detect a negative edge during the first cyclic pass.

If output Q receives the value 1 during initialization, output Q = 1 is set after initialization for the pulse stretching time T.

Block diagram



Time diagram



Output pulse Q as a function of pulse duration T and input pulse I

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
T	Pulse stretching time (ms)	0	SDTIME	
Q	Output pulse	0	0/1	

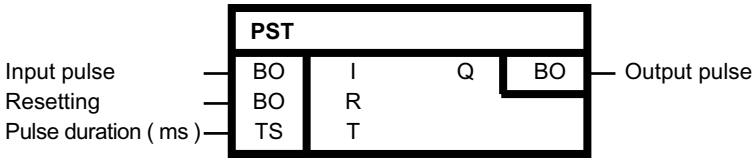
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.45 PST

Pulse stretching block (BOOL type)

Symbol



Brief description

Block for the generation of a pulse with a minimum duration and with additional reset input.

Method of operation

The rising edge of a pulse at input I sets output Q to 1.

Output Q does not fall back to 0 until input pulse I = 0 and the pulse duration T has expired.

Output Q can be set to zero at any time by means of the reset input R with R = 1.

When T=0, a pulse duration of 1 cycle is active.

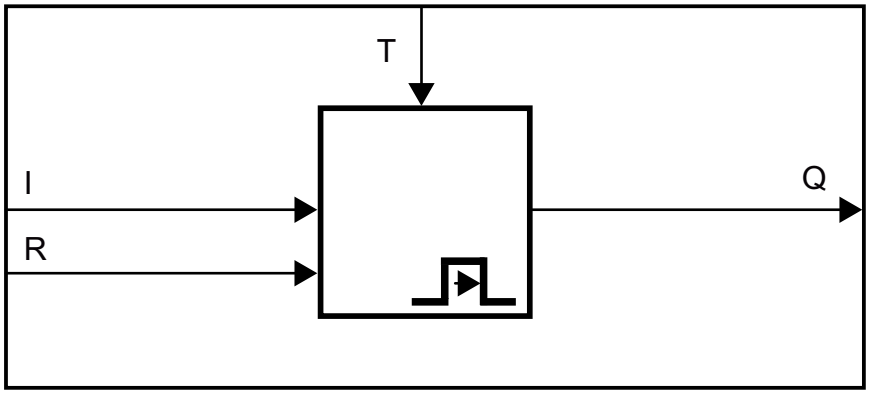
Initialization

The initialization defines the start value for the first cyclic pass.

If input I receives the value 1 during initialization from the upstream block output, the block cannot detect a positive edge during the first cyclic pass.

If output Q receives the value 1 during initialization, output Q = 1 is set after initialization for the pulse duration T.

Block diagram



Time diagram

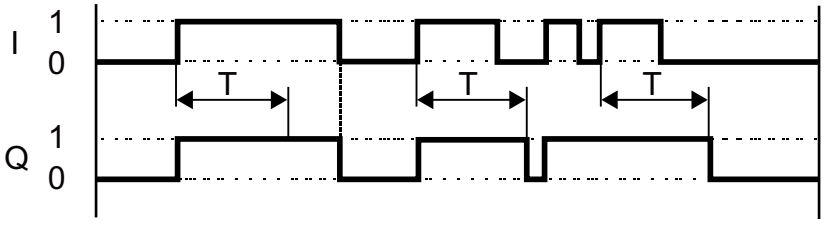


Figure 8-1 Output pulse Q as a function of pulse duration T and input pulse I (with R = 0)

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input pulse	0	0/1	
R	Resetting	0	0/1	
T	Pulse duration (ms)	0	SDTIME	
Q	Output pulse	0	0/1	

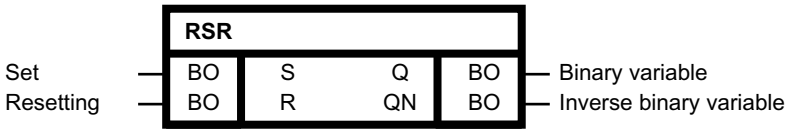
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.46 RSR

RS flip-flop, R-dominant (BOOL type)

Symbol



Brief description

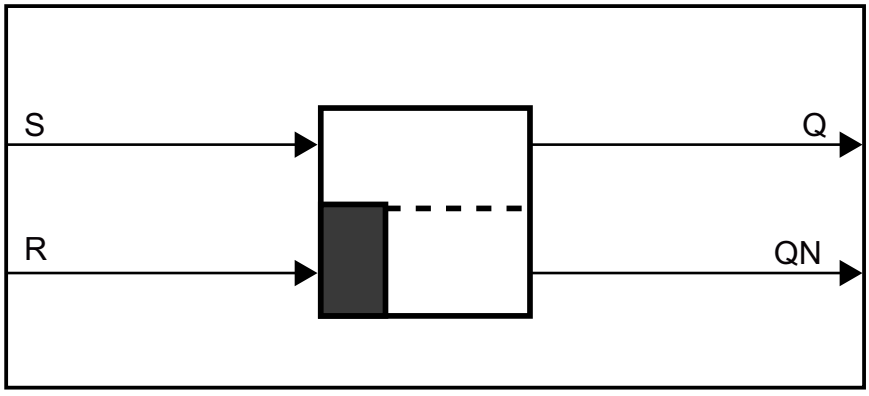
Used as static binary value memory

Method of operation

With logic 1 at input S, output Q is set to logic 1. If input R is set to logic 1, then output Q is set to logic 0. If both inputs are logic 0, then Q does not change. However, if the two inputs are logic 1, then Q is logic 0, since the reset input dominates.

Output QN always has the value inverse to Q.

Block diagram



Truth table(s)

Binary values when set/reset command is given

Binary command		Output status Q
S	R	
0	0	Q does not change
0	1	Q = 0
1	0	Q = 1
1	1	Q = 0

Block connections

Block connection	Description	Default	Value range	Attributes
S	Set	0	0/1	
R	Resetting	0	0/1	
Q	Binary variable	0	0/1	
QN	Inverse binary variable	1	0/1	

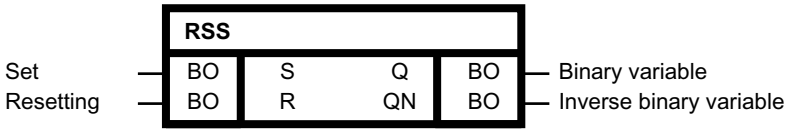
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.47 RSS

RS flip-flop, S-dominant (BOOL type)

Symbol



Brief description

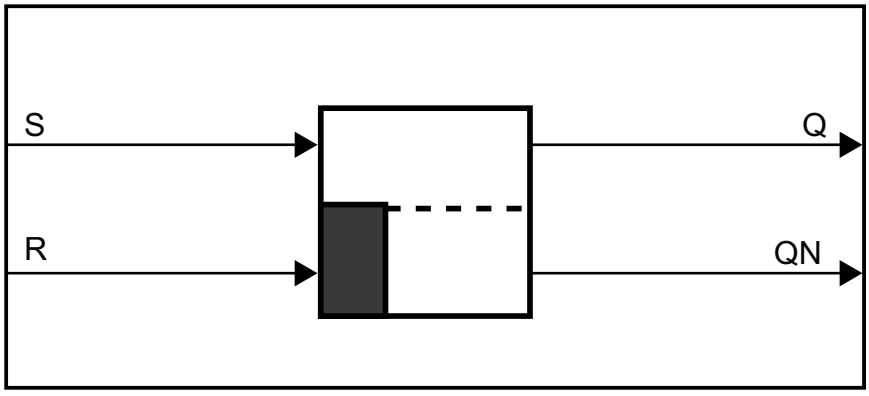
Block of the BOOL type for use as a static binary value memory

Method of operation

With logic 1 at input S, output Q is set to logic 1. If input R is set to logic 1, then output Q is set to logic 0. If both inputs are logic 0, then Q does not change. However, if the two inputs are logic 1, then Q is also logic 1, since the setting input dominates.

Output QN always has the value inverse to Q.

Block diagram



Truth table(s)

Binary values when set/reset command is given

Binary command		Output status Q
S	R	
0	0	Q does not change
0	1	Q = 0
1	0	Q = 1
1	1	Q = 0

Block connections

Block connection	Description	Default	Value range	Attributes
S	Set	0	0/1	
R	Resetting	0	0/1	
Q	Binary variable	0	0/1	
QN	Inverse binary variable	1	0/1	

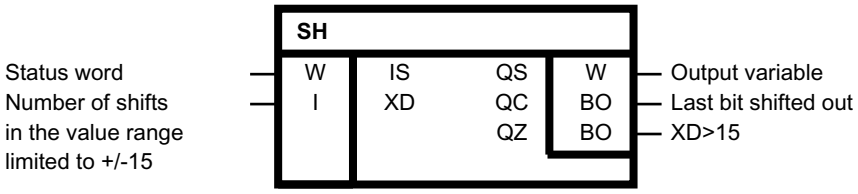
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.48 SH

Shift block (WORD type)

Symbol



Brief description

The block of the WORD type shifts a status word bit-by-bit to the left or right.

Method of operation

The block shifts the status word present at input IS bit-by-bit by the number of positions specified at input XD.

New positions created by the shifting are filled with zeros irrespective of the shift direction.

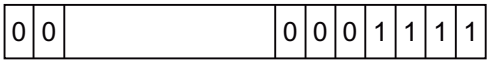
The last bit shifted out is output on output QC. When XD = 0, QC = 0 is always true. When |XD| > 15, QC = 0, QS = 0 and QZ = 1 are always true.

Shift to the left - example:

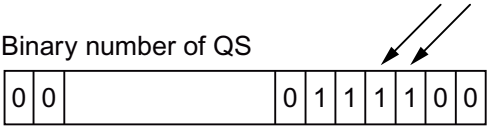
XD = 2; IS = 15

-> QS = 60; QC = 0

Binary number of IS



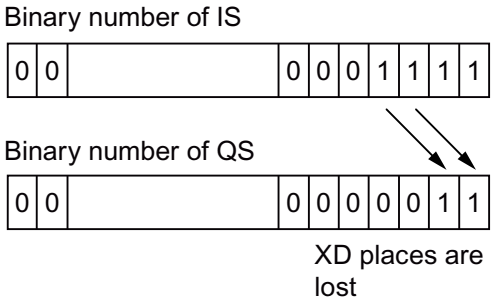
Binary number of QS



Shift to the right - example:

XD = -2; IS = 15

-> QS = 3 (remainder is omitted); QC = 1



Block connections

Block connection	Description	Default	Value range	Attributes
IS	Status word	16#0000	WORD	
XD	Number of shifts in the value range limited to +/-15	0	INT	
QS	Output variable	16#0000	WORD	
QC	Last bit shifted out	0	0/1	
QZ	XD>15	0	0/1	

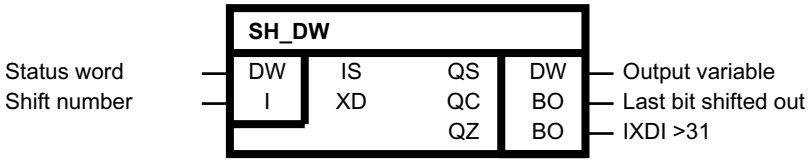
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.4.49 SH_DW

Shift block (DWORD type)

Symbol



Brief description

The block of the DWORD type shifts a status word bit-by-bit to the left or right.

Method of operation

The block shifts the status word present at input IS bit-by-bit by the number of positions specified at input XD.

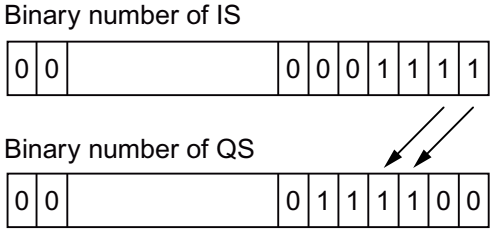
New positions created by the shifting are filled with zeros irrespective of the shift direction.

The last bit shifted out is output on output QC. When XD = 0, QC = 0 is always true. When |XD| > 31, QC = 0, QS = 0 and QZ = 1 are always true.

Shift to the left - example:

XD = 2; IS = 15

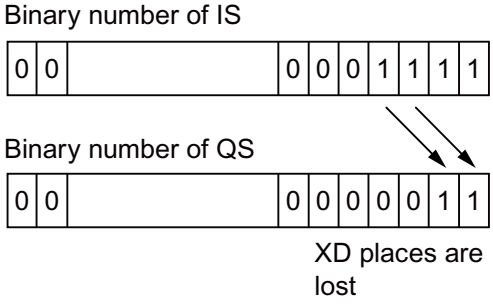
-> QS = 60; QC = 0



Shift to the right - example:

XD = -2; IS = 15

-> QS = 3 (remainder is omitted); QC = 1



Block connections

Block connection	Description	Default	Value range	Attributes
IS	Status word	16#00000000	DWORD	
XD	Shift number	0	+/-31	
QS	Output variable	16#00000000	DWORD	
QC	Last bit shifted out	0	0/1	
QZ	IXDI > 31	0	0/1	

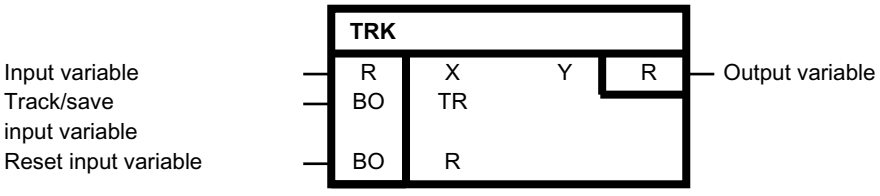
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.50 TRK

Tracking/memory element (REAL type)

Symbol



Brief description

Block of the REAL type for saving a current input value with the following properties:

- Edge-controlled latch functions for the input value
- Level-controlled correction of the output value

Method of operation

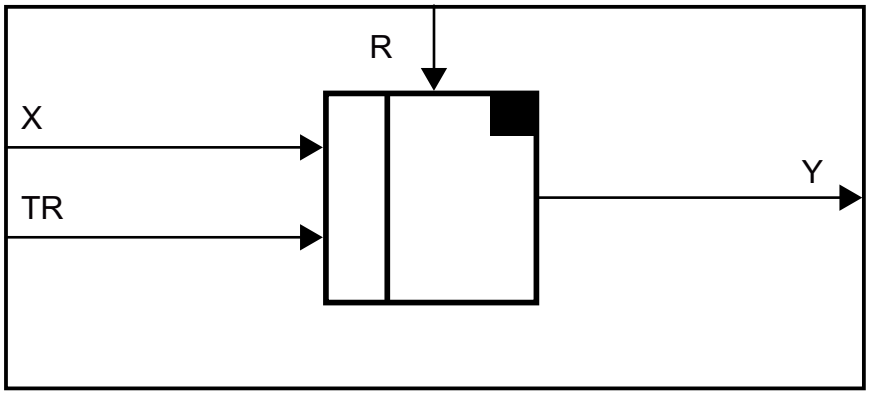
TRACK	TR = 1	Direct tracking of output value Y = X.
	TR = 1-> 0	With a negative edge at TR, the current input variable is saved and output on output Y.
	TR = 0	The value at output Y does not change.
RESET	R = 1	Output Y is reset to 0. The reset input is dominant.

Initialization

If input TR receives the value 1 during initialization of an upstream block output, a negative edge can be detected during the first cyclic pass. In START mode, the value for TR is stored temporarily.

If input TR receives the value 0 during initialization of the upstream block output, the block cannot detect a negative edge during the first cyclic pass.

Block diagram



Truth table(s)

Input		Output Y at the time of triggering
TR	R	
0	0	$Y_n = Y_{n-1}$
1	0	$Y_n = X_n$
1	1	$Y_n = 0$
1->0	0	$Y_n = X_n$
1->0	1	$Y_n = 0$

1 -> 0: Fall

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
TR	Track/save input variable	0	0/1	
R	Reset input variable	0	0/1	
Y	Output variable	0.0	REAL	

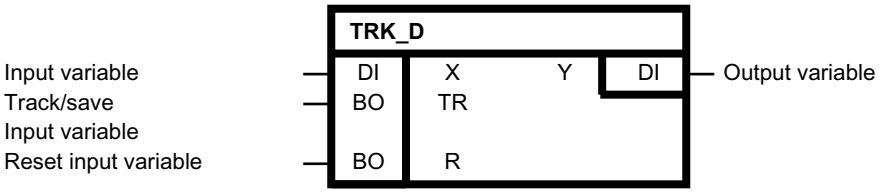
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.4.51 TRK_D

Tracking/memory element (DOUBLE INTEGER type)

Symbol



Brief description

Block of the DOUBLE INTEGER type for saving a current input value with the following properties:

- Edge-controlled latch functions for the input value
- Level-controlled correction of the output value

Method of operation

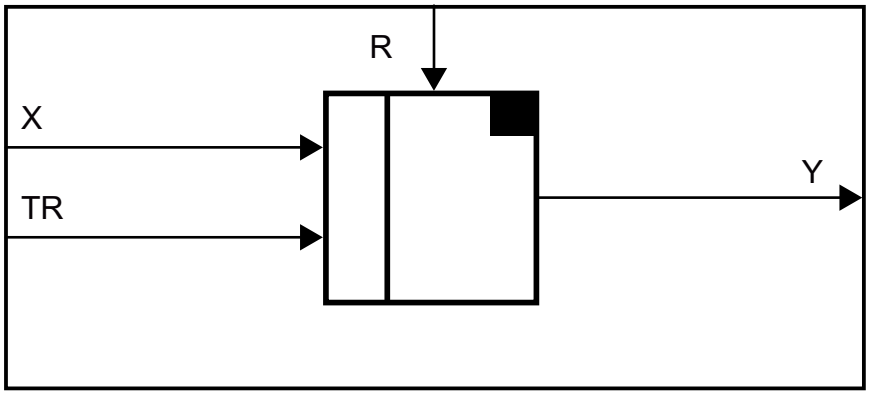
TRACK	TR = 1	Direct tracking of output value Y = X.
	TR = 1-> 0	With a negative edge at TR, the current input variable is saved and output on output Y.
	TR = 0	The value at output Y does not change.
RESET	R = 1	Output Y is reset to 0. The reset input is dominant.

Initialization

If input TR receives the value 1 during initialization of an upstream block output, a negative edge can be detected during the first cyclic pass. In START mode, the value for TR is stored temporarily.

If input TR receives the value 0 during initialization of the upstream block output, the block cannot detect a negative edge during the first cyclic pass.

Block diagram



Truth table(s)

Input		Output Y at the time of triggering
TR	R	
0	0	$Y_n = Y_{n-1}$
1	0	$Y_n = X_n$
1	1	$Y_n = 0$
1->0	0	$Y_n = X_n$
1->0	1	$Y_n = 0$

1 -> 0: Fall

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
TR	Track/save input variable	0	0/1	
R	Reset input variable	0	0/1	
Y	Output variable	0	DINT	

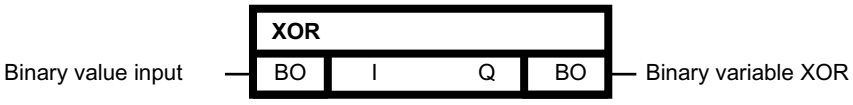
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.4.52 XOR

Logic exclusive OR operation (BOOL type)

Symbol



Brief description

XOR block with up to four inputs of the BOOL type

Method of operation

The block combines the binary values at the inputs I 1-4 according to the logic exclusive OR function and outputs the result at its binary output Q.

Output Q is 0, when a 0 is present at all inputs I1 to I4 or when a 1 is present at an even number of the inputs I1 to I4.

Output Q is 1, when a 1 is present at an odd number of the inputs I1 to I4.

Block connections

Block connection	Description	Default	Value range	Attributes
I	Binary value input	0	0/1	
Q	Binary variable XOR	0	0/1	

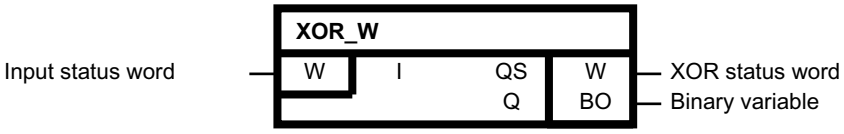
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	I comprises up to four connections (I1 to I4)

8.1.4.53 XOR_W

Logic exclusive OR operation (WORD type)

Symbol



Brief description

XOR block with up to four inputs of the WORD type

Method of operation

The block combines the status words I1 to I4 bit-by-bit according to the logic exclusive OR function.

The result is given to the block output QS (status word XOR).

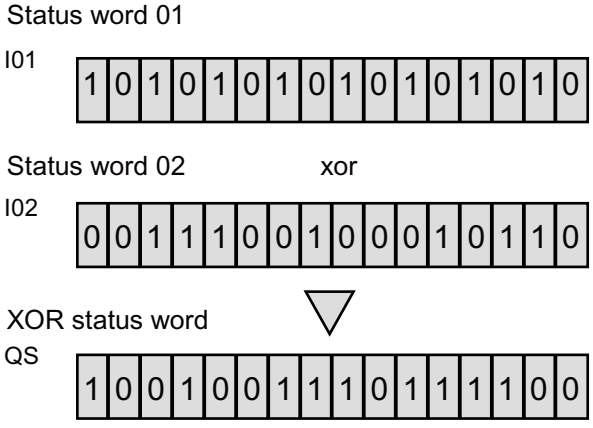
The following applies for bit k of status word XOR:

$$QS_k = (\overline{I01_k} \wedge I02_k) \vee (I01_k \wedge \overline{I02_k}), k = 1...16$$

A bit of status word XOR equals 1 if an odd number of the equivalent bits at the generic block inputs I1 to I4 equals 1.

The binary output Q is 1 if at least one bit of the status word XOR is equal to 1.

Following state diagram (for 3 inputs)



Block connections

Block connection	Description	Default	Value range	Attributes
I	Input status word	16#0000	WORD	
QS	XOR status word	16#0000	WORD	
Q	Binary variable	0	0/1	

Configuration data

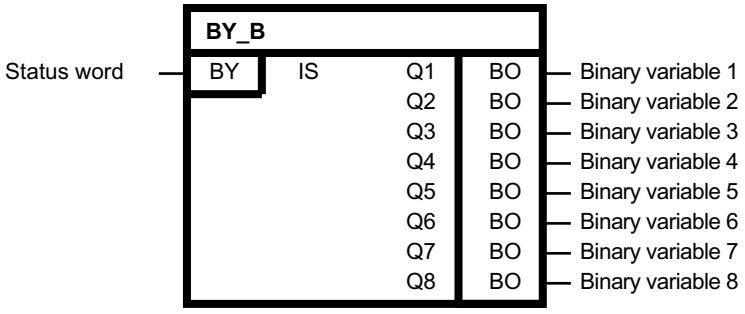
SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	Up to four connections (I1 to I4)

8.1.5 Conversion

8.1.5.1 BY_B

Status byte to eight binary variables converter

Symbol



Brief description

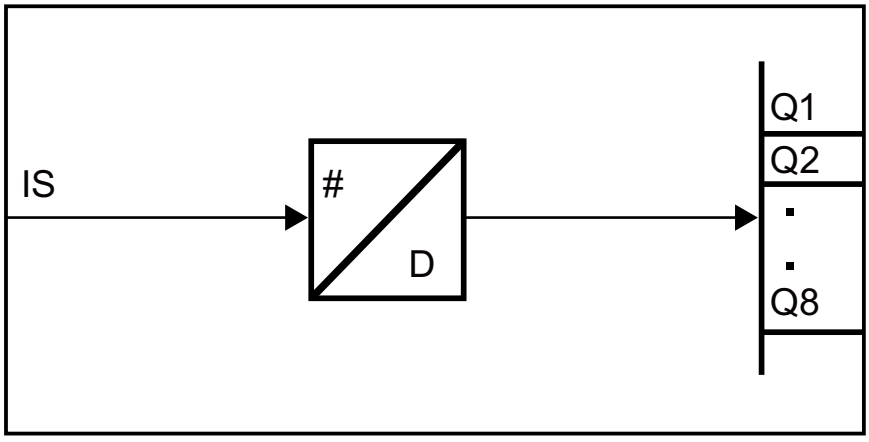
Status word decoding to eight binary variables

Method of operation

This block decodes the status word IS to eight binary variables and gives the result to its outputs Q1 to Q8.

The binary variable of outputs Q1 to Q8 is assigned to each dual equivalent 2^0 to 2^7 of the status byte.

Block diagram



Mapping scheme

Bit position (dual equivalent) of status byte IS	Output variable
0 (2^0)	Q1
1 (2^1)	Q2
2 (2^2)	Q3
3 (2^3)	Q4
4 (2^4)	Q5
5 (2^5)	Q6
6 (2^6)	Q7
7 (2^7)	Q8

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IS	Status word	16#00	BYTE	
Q1	Binary variable 1	0	0/1	
Q2	Binary variable 2	0	0/1	
Q3	Binary variable 3	0	0/1	
Q4	Binary variable 4	0	0/1	
Q5	Binary variable 5	0	0/1	
Q6	Binary variable 6	0	0/1	
Q7	Binary variable 7	0	0/1	
Q8	Binary variable 8	0	0/1	

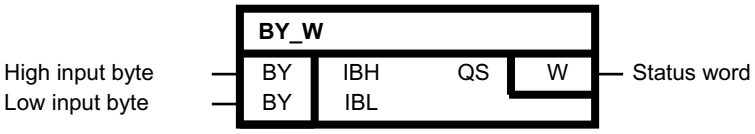
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.2 BY_W

Status byte to status word converter

Symbol



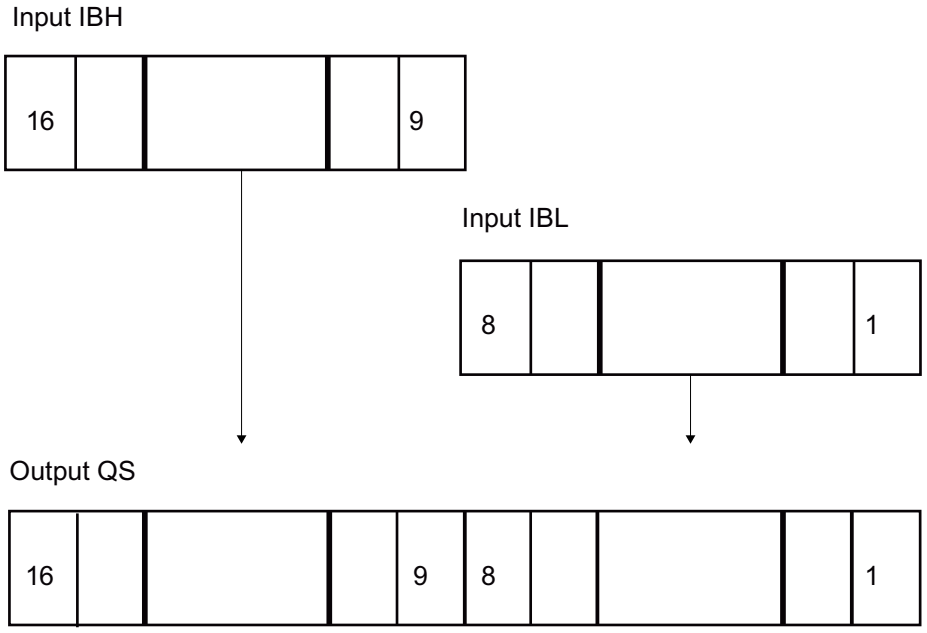
Brief description

Combining two bytes into one word

Method of operation

The block combines two bytes into one word. The low byte of the output word is assigned to the input byte IBL and the high byte of the output word is assigned to the input byte IBH. The output word is present on QS according to the following conversion scheme.

Conversion scheme



Block connections

Block connection	Description	Default	Value range	Attributes
IBH	High input byte	16#00	BYTE	
IBL	Low input byte	16#00	BYTE	
QS	Status word	16#0000	WORD	

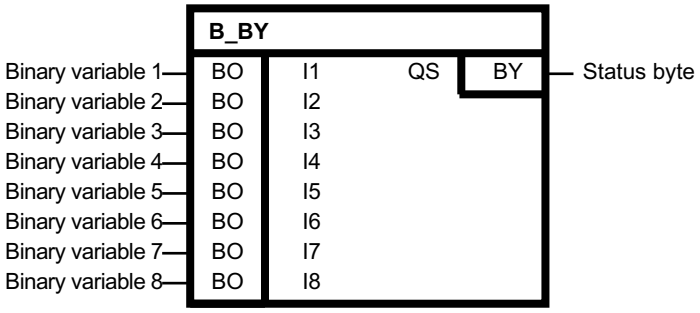
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.3 B_BY

Eight binary variables to status byte converter

Symbol



Brief description

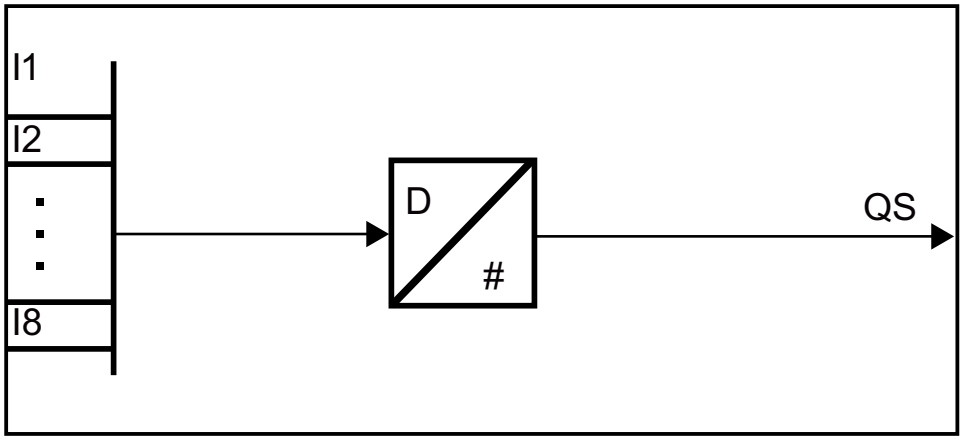
Status byte generation from eight binary variables

Method of operation

This block combines the binary variables from I1 to I8 into a status byte and gives the result to its output QS.

Each binary variable of inputs I1 to I8 is assigned the dual equivalent 2^0 to 2^7 from which the status word is generated.

Block diagram



Mapping scheme

Input variable	Bit position (dual equivalent) of status byte QS
I1	0 (2^0)
I2	1 (2^1)
I3	2 (2^2)
I4	3 (2^3)

Input variable	Bit position (dual equivalent) of status byte QS
I5	4 (2^4)
I6	5 (2^5)
I7	6 (2^6)
I8	7 (2^7)

Block connections

Block connection	Description	Default	Value range	Attributes
I1	Binary variable 1	0	0/1	
I2	Binary variable 2	0	0/1	
I3	Binary variable 3	0	0/1	
I4	Binary variable 4	0	0/1	
I5	Binary variable 5	0	0/1	
I6	Binary variable 6	0	0/1	
I7	Binary variable 7	0	0/1	
I8	Binary variable 8	0	0/1	
QS	Status byte	16#00	BYTE	

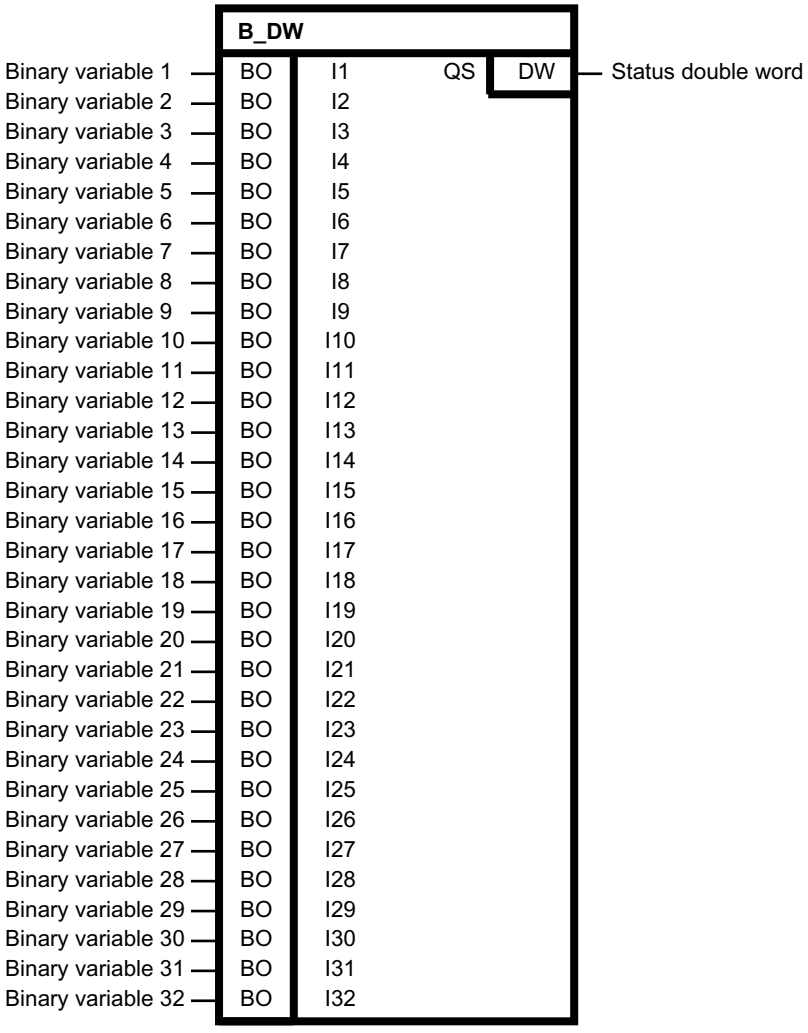
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.4 B_DW

32 binary variables to status double word converter

Symbol



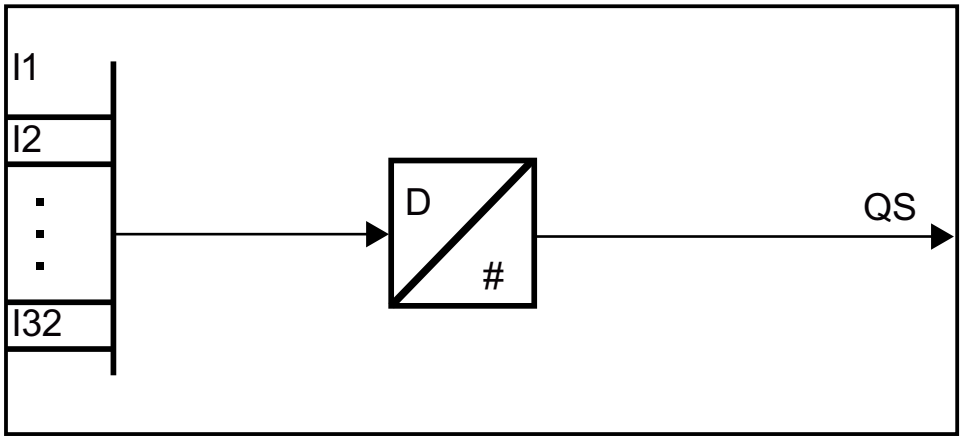
Brief description

Status double word generation from 32 binary variables

Method of operation

The block combines the binary variables of I1 to I32 into a status double word and outputs the result at output QS. Each binary variable of inputs I1 to I32 is assigned the dual equivalent 2^0 to 2^{31} from which the status double word is generated.

Block diagram



Mapping scheme

Input parameters	Bit position (dual equivalent) of status byte QS
I1	0 (2^0)
I2	1 (2^1)
I3	2 (2^2)
...	...
I32	31 (2^{31})

Block connections

Block connection	Description	Default	Value range	Attributes
I1	Binary variable 1	0	0/1	
I2	Binary variable 2	0	0/1	
I3	Binary variable 3	0	0/1	
I4	Binary variable 4	0	0/1	
I5	Binary variable 5	0	0/1	
I6	Binary variable 6	0	0/1	
I7	Binary variable 7	0	0/1	
I8	Binary variable 8	0	0/1	
I9	Binary variable 9	0	0/1	
I10	Binary variable 10	0	0/1	
I11	Binary variable 11	0	0/1	
I12	Binary variable 12	0	0/1	
I13	Binary variable 13	0	0/1	
I14	Binary variable 14	0	0/1	
I15	Binary variable 15	0	0/1	
I16	Binary variable 16	0	0/1	

Block connection	Description	Default	Value range	Attributes
I17	Binary variable 17	0	0/1	
I18	Binary variable 18	0	0/1	
I19	Binary variable 19	0	0/1	
I20	Binary variable 20	0	0/1	
I21	Binary variable 21	0	0/1	
I22	Binary variable 22	0	0/1	
I23	Binary variable 23	0	0/1	
I24	Binary variable 24	0	0/1	
I25	Binary variable 25	0	0/1	
I26	Binary variable 26	0	0/1	
I27	Binary variable 27	0	0/1	
I28	Binary variable 28	0	0/1	
I29	Binary variable 29	0	0/1	
I30	Binary variable 30	0	0/1	
I31	Binary variable 31	0	0/1	
I32	Binary variable 32	0	0/1	
QS	Status double word	16#00000000	DWORD	

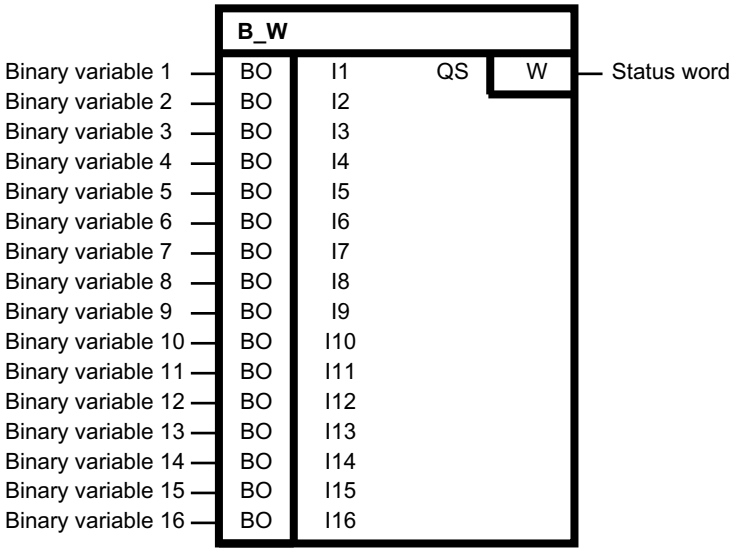
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.5 B_W

16 binary variables to status word converter

Symbol



Brief description

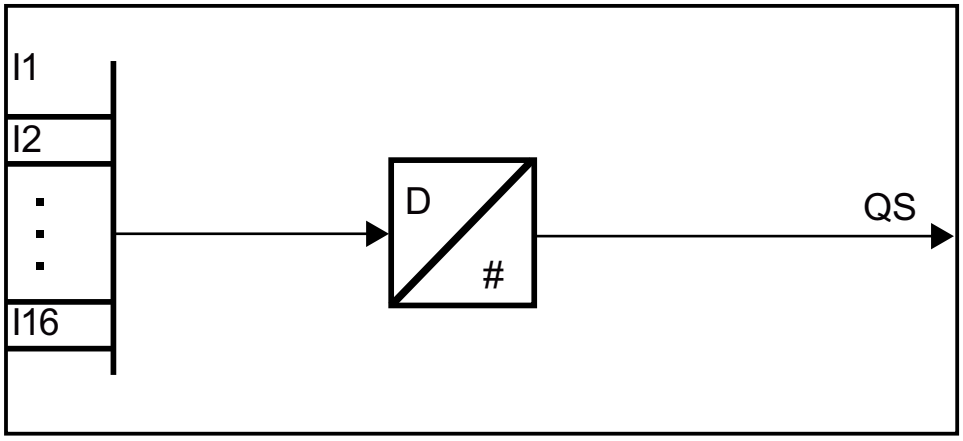
Status word generation from 16 binary variables

Method of operation

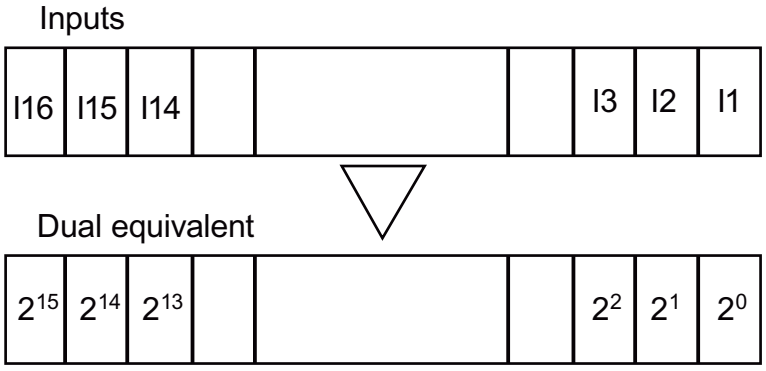
This block combines the binary variables from I1 to I16 into a status word and gives the result to its output QS.

Each binary variable of inputs I1 to I16 is assigned the dual equivalent 2^0 to 2^{15} from which the status word is generated.

Block diagram



Conversion scheme



Block connections

Block connection	Description	Default	Value range	Attributes
I1	Binary variable 1	0	0/1	
I2	Binary variable 2	0	0/1	
I3	Binary variable 3	0	0/1	
I4	Binary variable 4	0	0/1	
I5	Binary variable 5	0	0/1	
I6	Binary variable 6	0	0/1	
I7	Binary variable 7	0	0/1	
I8	Binary variable 8	0	0/1	
I9	Binary variable 9	0	0/1	
I10	Binary variable 10	0	0/1	
I11	Binary variable 11	0	0/1	
I12	Binary variable 12	0	0/1	
I13	Binary variable 13	0	0/1	
I14	Binary variable 14	0	0/1	
I15	Binary variable 15	0	0/1	
I16	Binary variable 16	0	0/1	
QS	Status word	16#0000	WORD	

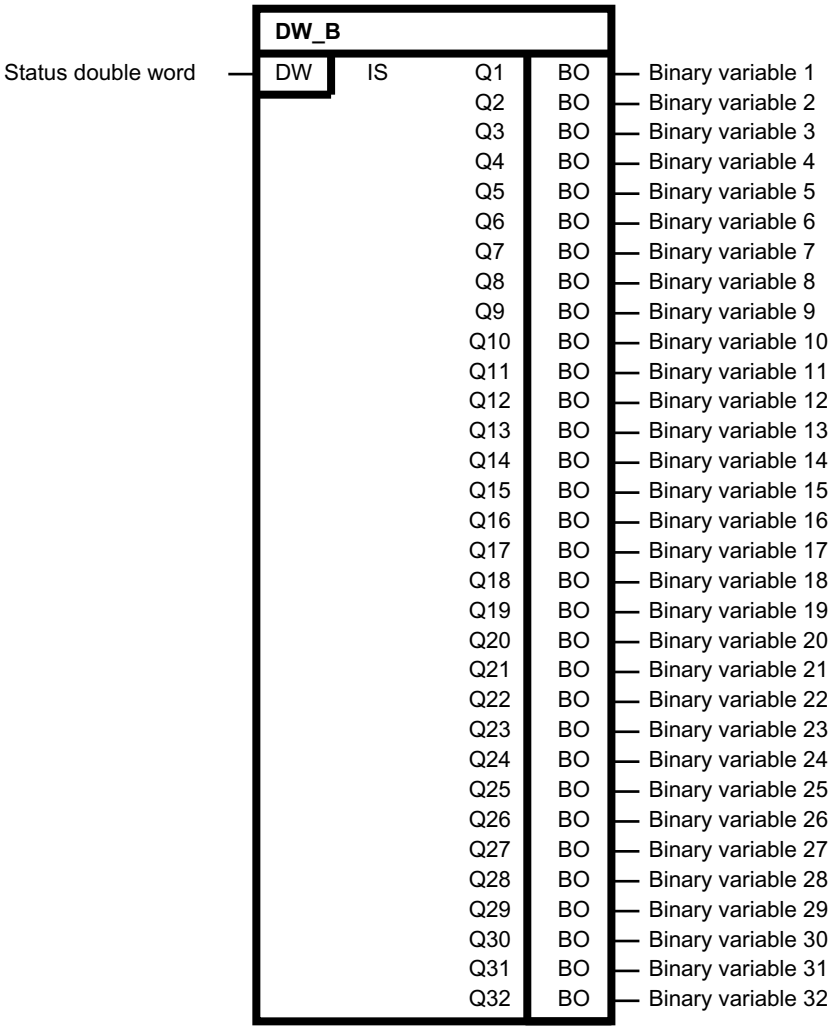
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.6 DW_B

Status double word to 32 binary variables converter

Symbol



Brief description

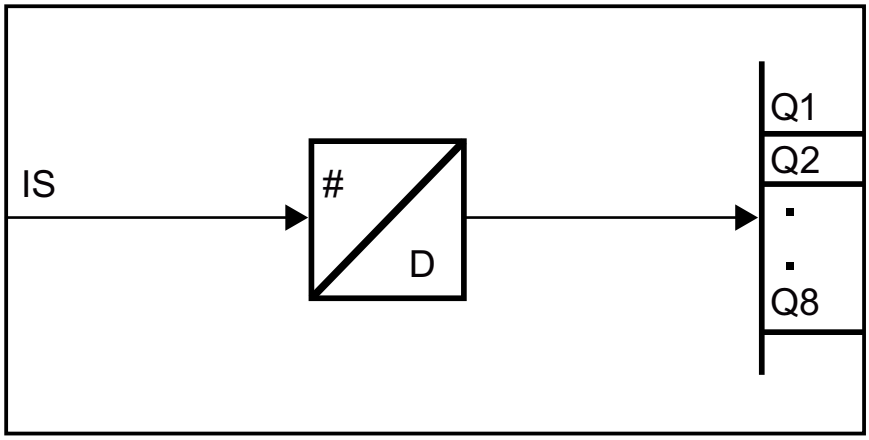
Status double word decryption to 32 binary variables

Method of operation

This block decodes the status double word IS to 32 binary variables and gives the result to its outputs Q1 to Q32.

The binary variable of outputs Q1 to Q32 is assigned to each dual equivalent 2^0 to 2^{31} of the status word.

Block diagram



Mapping scheme

Bit position (dual equivalent) of status double word IS	Output variable
0 (2^0)	Q1
1 (2^1)	Q2
2 (2^2)	Q3
...	...
31 (2^{31})	Q32

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IS	Status double word	16#00000000	DWORD	
Q1	Binary variable 1	0	0/1	
Q2	Binary variable 2	0	0/1	
Q3	Binary variable 3	0	0/1	
Q4	Binary variable 4	0	0/1	
Q5	Binary variable 5	0	0/1	
Q6	Binary variable 6	0	0/1	
Q7	Binary variable 7	0	0/1	
Q8	Binary variable 8	0	0/1	
Q9	Binary variable 9	0	0/1	
Q10	Binary variable 10	0	0/1	
Q11	Binary variable 11	0	0/1	
Q12	Binary variable 12	0	0/1	
Q13	Binary variable 13	0	0/1	
Q14	Binary variable 14	0	0/1	

Block connection	Description	Default	Value range	Attributes
Q15	Binary variable 15	0	0/1	
Q16	Binary variable 16	0	0/1	
Q17	Binary variable 17	0	0/1	
Q18	Binary variable 18	0	0/1	
Q19	Binary variable 19	0	0/1	
Q20	Binary variable 20	0	0/1	
Q21	Binary variable 21	0	0/1	
Q22	Binary variable 22	0	0/1	
Q23	Binary variable 23	0	0/1	
Q24	Binary variable 24	0	0/1	
Q25	Binary variable 25	0	0/1	
Q26	Binary variable 26	0	0/1	
Q27	Binary variable 27	0	0/1	
Q28	Binary variable 28	0	0/1	
Q29	Binary variable 29	0	0/1	
Q30	Binary variable 30	0	0/1	
Q31	Binary variable 31	0	0/1	
Q32	Binary variable 32	0	0/1	

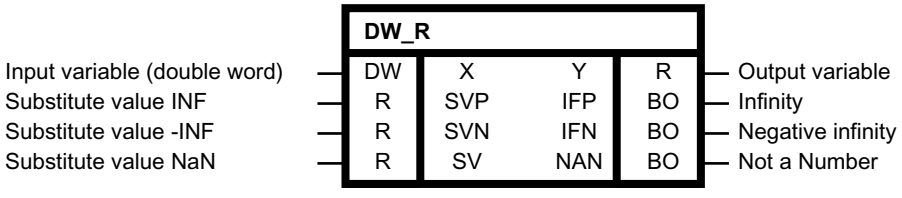
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.7 DW_R

Acceptance of bit string as REAL value

Symbol



Brief description

This block accepts the bit string at the input as a REAL variable and checks the value for validity

Method of operation

The DW_R block accepts the bit string at the input as a REAL variable and supplies it at output Y.

The bit pattern of input variable X is checked. If the bit pattern corresponds to the representation for +/-infinite or NaN according to IEEE 754, the corresponding binary outputs IFP, IFN, and NAN are set to 1, and the substitute value predefined for each are applied at output Y.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable (double word)	16#00000000	DWORD	
SVP	Substitute value INF	3.402823 E38	REAL	
SVN	Substitute value -INF	-3.402823 E38	REAL	
SV	Substitute value NaN	0.0	REAL	
IFP	Infinity	0	0/1	
IFN	Negative infinity	0	0/1	
NAN	Not a Number	0	0/1	
Y	Output variable	0.0	REAL	

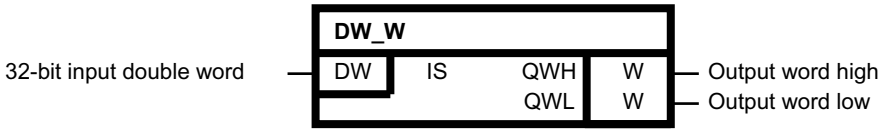
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.8 DW_W

Status double word to status word converter

Symbol



Brief description

A 32-bit double word is divided into two 16-bit words.

Method of operation

Output variables are calculated according to the following regulation:

QWL = IS mod 2^{16}

QWH = IS / 2^{16}

Block connections

Block connection	Description	Default	Value range	Attributes
IS	32-bit input double word	16#00000000	DWORD	
QWH	Output word high	16#0000	WORD	
QWL	Output word low	16#0000	WORD	

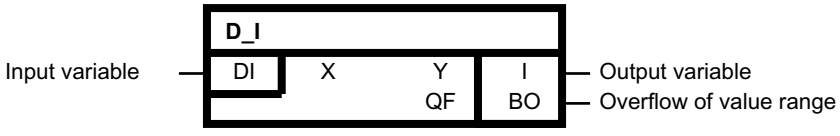
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.9 D_I

DOUBLE INTEGER to INTEGER converter

Symbol



Brief description

Conversion of a DOUBLE INTEGER variable to an INTEGER variable

Method of operation

This block converts a DOUBLE INTEGER variable to an INTEGER variable, i.e. the least significant word of the DOUBLE INTEGER input variable is applied to the output variable Y.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Output variable	0	INT	
QF	Overflow of value range	0	0/1	

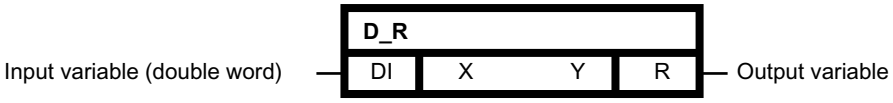
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.10 D_R

DOUBLE INTEGER to REAL converter

Symbol



Brief description

Conversion of a DOUBLE INTEGER variable to a REAL variable

Method of operation

This block converts a DOUBLE INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable (double word)	0	DINT	
Y	Output variable	0.0	REAL	

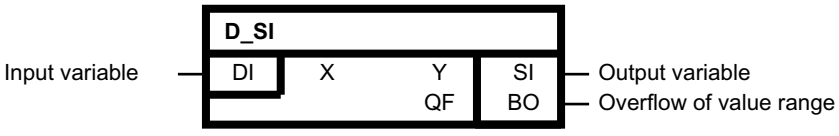
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.11 D_SI

DOUBLE INTEGER to SHORT INTEGER converter

Symbol



Brief description

Conversion of a DOUBLE INTEGER variable to a SHORT INTEGER variable

Method of operation

This block converts a DOUBLE INTEGER variable to a SHORT INTEGER variable, i.e. the least significant byte of the DOUBLE INTEGER input variable is applied to output variable Y.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Output variable	0	SINT	
QF	Overflow of value range	0	0/1	

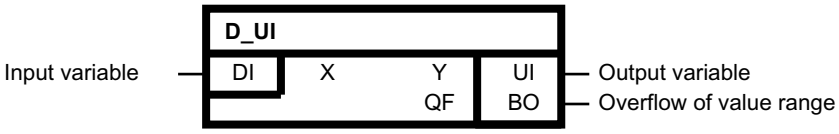
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.12 D_UI

DOUBLE INTEGER to UNSIGNED INTEGER converter

Symbol



Brief description

Conversion of a DOUBLE INTEGER variable to an UNSIGNED INTEGER variable

Method of operation

This block converts a DOUBLE INTEGER variable to an UNSIGNED INTEGER variable, i.e. the least significant word of the DOUBLE INTEGER input variable is applied to the output variable.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Output variable	0	UINT	
QF	Overflow of value range	0	0/1	

Configuration data

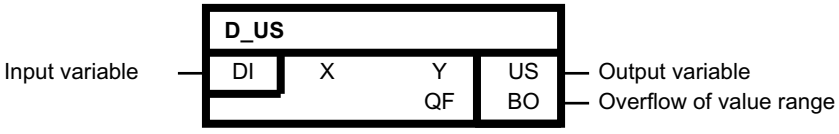
SIMOTION	✓
SINAMICS	✓

Can be inserted online	Yes
Special characteristics	-

8.1.5.13 D_US

DOUBLE INTEGER to UNSIGNED SHORT INTEGER converter

Symbol



Brief description

Conversion of a DOUBLE INTEGER variable to an UNSIGNED SHORT INTEGER variable

Method of operation

This block converts a DOUBLE INTEGER variable to an UNSIGNED SHORT INTEGER variable, i.e. the least significant word of the DOUBLE INTEGER input variable is applied to the output variable.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
Y	Output variable	0	USINT	
QF	Overflow of value range	0	0/1	

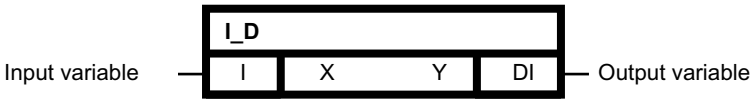
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.14 I_D

INTEGER to DOUBLE INTEGER converter

Symbol



Brief description

Conversion of an INTEGER variable to a DOUBLE INTEGER variable

Method of operation

This block converts an INTEGER variable to a DOUBLE INTEGER variable.

The input variable of data type INTEGER is copied to the least significant word of the output variable. If the input variable has a positive sign, the most significant word of the output variable is filled with 16#0000. If, on the other hand, the sign is negative, the most significant word receives the value 16#FFFF.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0	INT	
Y	Output variable	0	DINT	

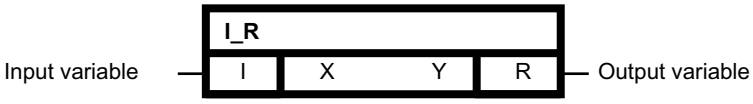
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	Yes
Special character-istics	-

8.1.5.15 I_R

INTEGER to REAL converter

Symbol



Brief description

Conversion of an INTEGER variable to a REAL variable

Method of operation

This block converts an INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
Y	Output variable	0.0	REAL	

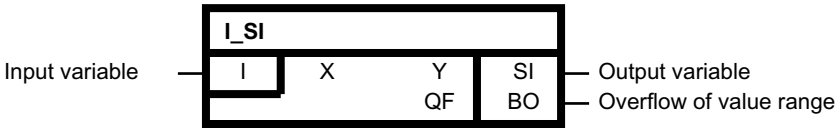
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.16 I_SI

INTEGER to SHORT INTEGER converter

Symbol



Brief description

Conversion of an INTEGER variable to a SHORT INTEGER variable

Method of operation

This block converts an INTEGER variable to a SHORT INTEGER variable, i.e. the least significant byte of the INTEGER input variable is applied to output variable Y.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
Y	Output variable	0	SINT	
QF	Overflow of value range	0	0/1	

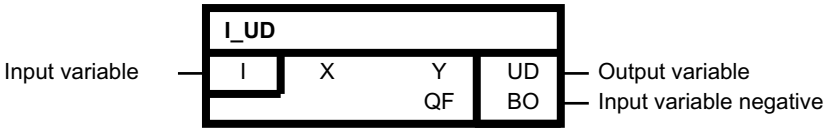
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.17 I_UD

INTEGER to UNSIGNED DOUBLE INTEGER converter

Symbol



Brief description

Conversion of an INTEGER variable to an UNSIGNED DOUBLE INTEGER variable

Method of operation

This block converts an INTEGER variable to an UNSIGNED DOUBLE INTEGER variable.

The input variable of data type INTEGER is copied to the least significant word of the output variable.

The most significant word of the output variable is filled with 16#0000.

If the value of the input variable is negative, QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
Y	Output variable	0	UDINT	
QF	Input variable negative	0	0/1	

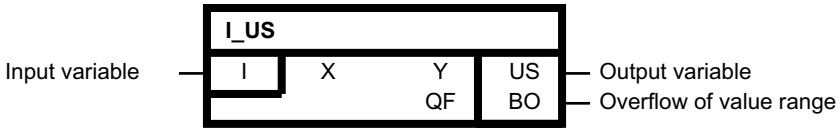
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.18 I_US

INTEGER to UNSIGNED SHORT INTEGER converter

Symbol



Brief description

Conversion of an INTEGER variable to an UNSIGNED SHORT INTEGER variable

Method of operation

This block converts an INTEGER variable to an UNSIGNED SHORT INTEGER variable, i.e. the least significant byte of the INTEGER input variable is applied to output variable Y.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
Y	Output variable	0	USINT	
QF	Overflow of value range	0	0/1	

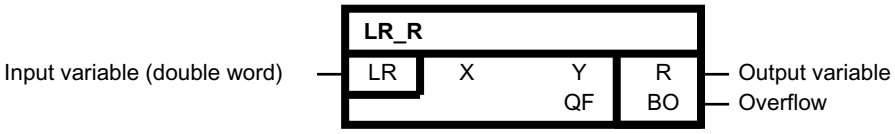
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.19 LR_R

LONG REAL to REAL converter

Symbol



Brief description

Conversion of a LONG REAL variable to a REAL variable

Method of operation

This block converts a LONG REAL variable to a REAL variable. The result is limited to the maximum range of data type REAL. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable (double word)	0	LREAL	
Y	Output variable	0.0	REAL	
QF	Overflow	0	0/1	

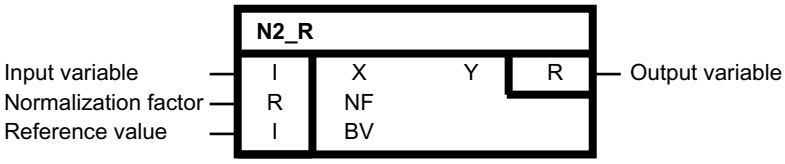
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.5.20 N2_R

16-bit fixed-point format (N2) to REAL converter

Symbol



Brief description

Conversion of a 16-bit fixed-point variable to a REAL variable. For the case X and BV= 16384 (corresponds to 100% in normalized PROFIdrive representation), output Y assumes the value at input NF.

Method of operation

Input variable X is mapped to output Y according to the following formula:

$$Y = \frac{(X \cdot NF)}{BV}$$

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
NF	Normalization factor	1.0	REAL	
BV	Reference value	16384	INT	
Y	Output variable	0.0	REAL	

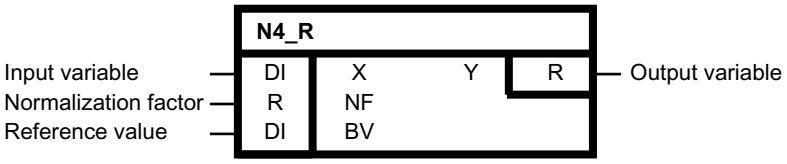
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.21 N4_R

32-bit fixed-point format (N4) to REAL converter

Symbol



Brief description

Conversion of a 32-bit fixed-point variable to a REAL variable. For the case X and BV= 1073741824 (corresponds to 100% in normalized PROFIdrive representation), output Y assumes the value at input NF.

Method of operation

Input variable X is mapped to output Y according to the following formula:

$$Y = \frac{(X \cdot NF)}{BV}$$

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
NF	Normalization factor	1.0	REAL	
BV	Reference value	1073741824	DINT	
Y	Output variable	0.0	REAL	

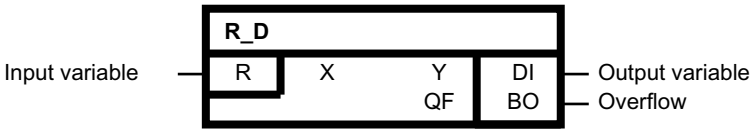
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.22 R_D

REAL to DOUBLE INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to a DOUBLE INTEGER variable

Method of operation

This block converts a REAL variable to a DOUBLE INTEGER variable. During the conversion, decimal places of the input variable are truncated.

Note

The number is not rounded off.

The result is limited to the data type of the output variable corresponding to -2^{31} or $2^{31}-1$. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	DINT	
QF	Overflow	0	0/1	

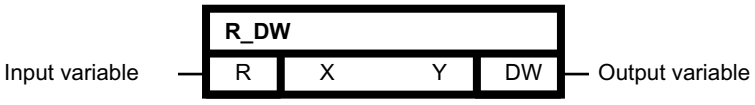
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.23 R_DW

Acceptance of bit string as DWORD

Symbol



Brief description

This block copies the bit string of the input variable to the output variable.

Method of operation

This block copies the bit string of input variable X to the output variable Y.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	16#00000000	DWORD	

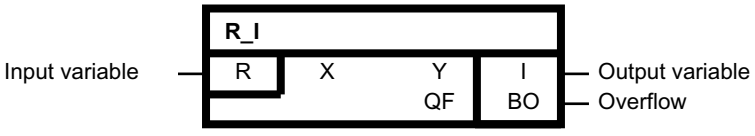
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.24 R_I

REAL to INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to an INTEGER variable

Method of operation

This block converts a REAL variable to an INTEGER variable. During the conversion, decimal places of the input variable are truncated. The number is not rounded off. The result is limited to the data type of the output variable corresponding to +32767 or -32768. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	INT	
QF	Overflow	0	0/1	

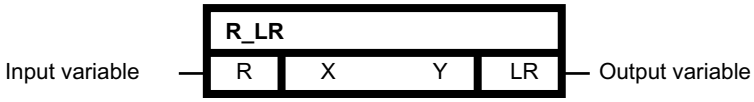
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.25 R_LR

REAL to LONG REAL converter

Symbol



Brief description

Conversion of a REAL variable to a LONG REAL variable

Method of operation

This block converts a REAL variable to a LONG REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0.0	LREAL	

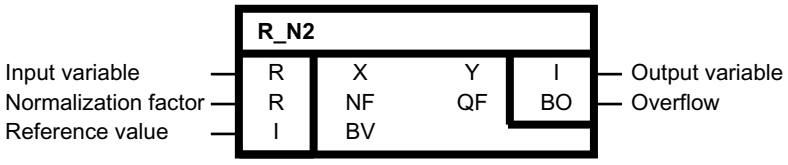
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.26 R_N2

REAL to 16-bit fixed-point format (N2) converter

Symbol



Brief description

Conversion of a REAL variable to a 16-bit fixed-point variable. For the case X = NF and BV = 16384 (default), output Y assumes the value 16384 (corresponds to 100% in normalized PROFIdrive representation).

Method of operation

Input variable X is mapped to output Y according to the following formula (result is rounded):

$$Y = \frac{X \cdot BV}{NF}$$

Y is limited to the range -32768 ≤ Y ≤ 32767 (corresponds to -200% ≤ Y < 200%).

Output QF (overflow) is set to "1" if X cannot be mapped on Y because of a range violation, or if NF = 0 has been set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
NF	Normalization factor	1.0	REAL	
BV	Reference value	16384	INT	
Y	Output variable	0	INT	
QF	Overflow	0	0/1	

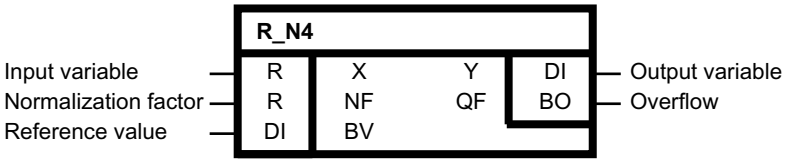
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.27 R_N4

REAL to 32-bit fixed-point format (N4) converter

Symbol



Brief description

Conversion of a REAL variable to a 32-bit fixed-point variable. For the case X = NF and BV = 1073741824 (default), output Y assumes the value 1073741824 (corresponds to 100%).

Method of operation

Input variable X is mapped to output Y according to the following formula (result is rounded):

$$Y = \frac{X \cdot BV}{NF}$$

Y is limited to the range -2147483648 ≤ Y ≤ 2147483647 (decimal) or 16#8000000 ≤ Y ≤ 16#7FFFFFF (hexadecimal) (corresponds to -200% ≤ Y < 200%).

Output QF (overflow) is set to "1" if X cannot be mapped on Y because of a range violation, or if NF = 0 has been set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
NF	Normalization factor	1.0	REAL	
BV	Reference value	1073741824	DINT	
Y	Output variable	0	DINT	
QF	Overflow	0	0/1	

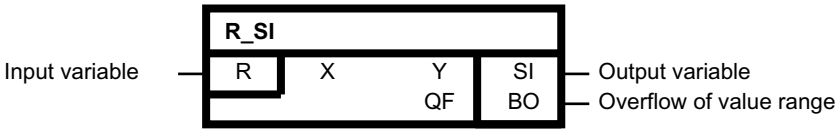
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.28 R_SI

REAL to SHORT INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to a SHORT INTEGER variable

Method of operation

This block converts a REAL variable to a SHORT INTEGER variable. During the conversion, decimal places of the input variable are truncated. The number is not rounded off. The result is limited to the data type of the output variable corresponding to -128 or 127. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	SINT	
QF	Overflow of value range	0	0/1	

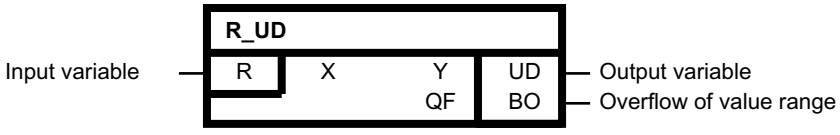
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.29 R_UD

REAL to UNSIGNED DOUBLE INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to an UNSIGNED DOUBLE INTEGER variable

Method of operation

This block converts a REAL variable to an UNSIGNED DOUBLE INTEGER variable. During the conversion, decimal places of the input variable are truncated. The number is not rounded off. The result is limited to the data type of the output variable corresponding to 0 or 2^{32}-1. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	UDINT	
QF	Overflow of value range	0	0/1	

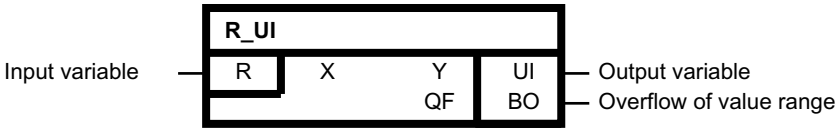
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.30 R_UI

REAL to UNSIGNED INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to an UNSIGNED INTEGER variable

Method of operation

This block converts a REAL variable to an UNSIGNED INTEGER variable. During the conversion, decimal places of the input variable are truncated. The number is not rounded off. The result is limited to the data type of the output variable corresponding to 0 or 2^{16}-1. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	UINT	
QF	Overflow of value range	0	0/1	

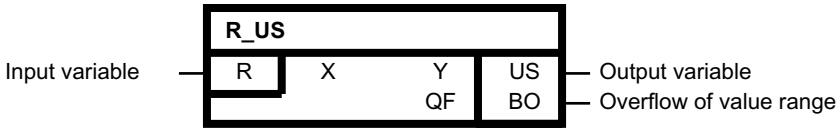
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.31 R_US

REAL to UNSIGNED SHORT INTEGER converter

Symbol



Brief description

Conversion of a REAL variable to an UNSIGNED SHORT INTEGER variable

Method of operation

This block converts a REAL variable to an UNSIGNED SHORT INTEGER variable.

During the conversion, decimal places of the input variable are truncated. The number is not rounded off. The result is limited to the data type of the output variable corresponding to 0 or 2^8 -1. If the output variable has been limited, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
Y	Output variable	0	USINT	
QF	Overflow of value range	0	0/1	

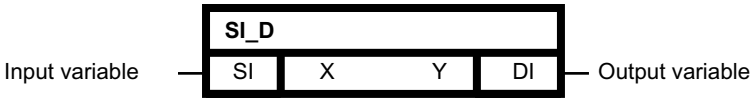
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.32 SI_D

SHORT INTEGER to DOUBLE INTEGER converter

Symbol



Brief description

Conversion of a SHORT INTEGER variable to a DOUBLE INTEGER variable

Method of operation

This block converts a SHORT INTEGER variable to a DOUBLE INTEGER variable.

The input variable of data type SHORT INTEGER is copied to the least significant byte of the output variable. If the input variable has a positive sign, the most significant bytes of the output variable are filled with 16#00. If, on the other hand, the sign is negative, the most significant bytes receive the value 16#FF.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	SINT	
Y	Output variable	0	DINT	

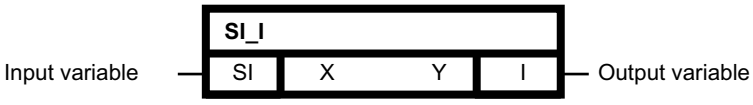
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.33 SI_I

SHORT INTEGER to INTEGER converter

Symbol



Brief description

Conversion of a SHORT INTEGER variable to an INTEGER variable

Method of operation

This block converts a SHORT INTEGER variable to an INTEGER variable.

The input variable of data type SHORT INTEGER is copied to the least significant byte of the output variable. If the input variable has a positive sign, the most significant byte of the output variable is filled with 16#00. If, on the other hand, the sign is negative, the most significant byte receives the value 16#FF.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	SINT	
Y	Output variable	0	INT	

Configuration data

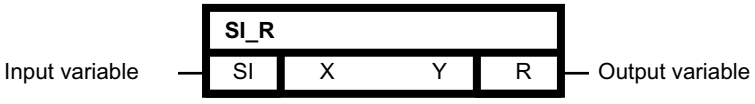
SIMOTION	✓
SINAMICS	-

Can be inserted online	Yes
Special characteristics	-

8.1.5.34 SI_R

SHORT INTEGER to REAL converter

Symbol



Brief description

Conversion of a SHORT INTEGER variable to a REAL variable

Method of operation

This block converts a SHORT INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	SINT	
Y	Output variable	0.0	REAL	

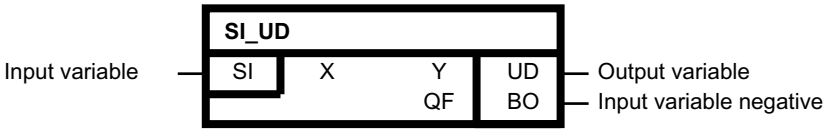
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.35 SI_UD

SHORT INTEGER to UNSIGNED DOUBLE INTEGER converter

Symbol



Brief description

Conversion of a SHORT INTEGER variable to an UNSIGNED DOUBLE INTEGER variable

Method of operation

This block converts a SHORT INTEGER variable to an UNSIGNED DOUBLE INTEGER variable.

The input variable of data type SHORT INTEGER is copied to the least significant byte of the output variable. The most significant bytes of the output variable are filled with 16#00. If the value of the input variable is negative, QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	SINT	
Y	Output variable	0	UDINT	
QF	Input variable negative	0	0/1	

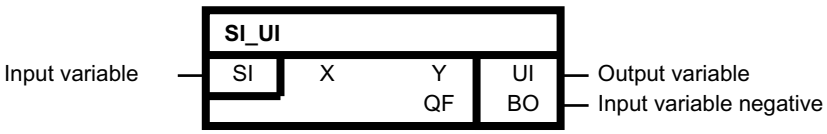
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.36 SI_UI

SHORT INTEGER to UNSIGNED INTEGER converter

Symbol



Brief description

Conversion of a SHORT INTEGER variable to an UNSIGNED INTEGER variable

Method of operation

This block converts a SHORT INTEGER variable to an UNSIGNED INTEGER variable.

The input variable of data type SHORT INTEGER is copied to the least significant byte of the output variable. The most significant byte of the output variable is filled with 16#00. If the value of the input variable is negative, QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	SINT	
Y	Output variable	0	UDINT	
QF	Input variable negative	0	0/1	

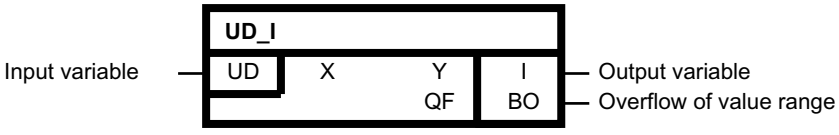
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.37 UD_I

UNSIGNED DOUBLE INTEGER to INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED DOUBLE INTEGER variable to an INTEGER variable

Method of operation

This block converts an UNSIGNED DOUBLE INTEGER variable to an INTEGER variable, i.e. the least significant word of the UNSIGNED DOUBLE INTEGER input variable is applied to output variable Y.

If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UDINT	
Y	Output variable	0	INT	
QF	Overflow of value range	0	0/1	

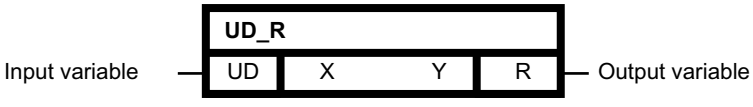
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.38 UD_R

UNSIGNED DOUBLE INTEGER to REAL converter

Symbol



Brief description

Conversion of an UNSIGNED DOUBLE INTEGER variable to a REAL variable

Method of operation

This block converts an UNSIGNED DOUBLE INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UDINT	
Y	Output variable	0.0	REAL	

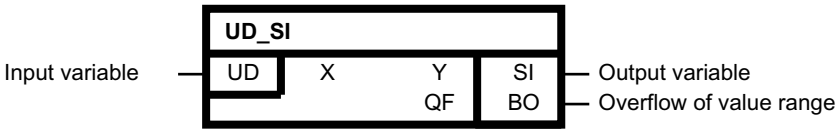
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.39 UD_SI

UNSIGNED DOUBLE INTEGER to SHORT INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED DOUBLE INTEGER variable to a SHORT INTEGER variable

Method of operation

This block converts an UNSIGNED DOUBLE INTEGER variable to a SHORT INTEGER variable, i.e. the least significant byte of the UNSIGNED DOUBLE INTEGER input variable is applied to output variable Y. If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UDINT	
Y	Output variable	0	SINT	
QF	Overflow of value range	0	0/1	

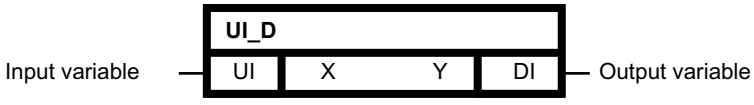
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.40 UI_D

UNSIGNED INTEGER to DOUBLE INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED INTEGER variable to a DOUBLE INTEGER variable

Method of operation

This block converts an UNSIGNED INTEGER variable to a DOUBLE INTEGER variable.

The input variable of data type UNSIGNED INTEGER is copied to the least significant word of output variable Y. The most significant word is filled with 16#0000.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UINT	
Y	Output variable	0	DINT	

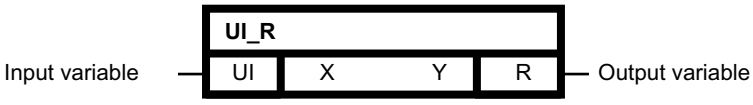
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.41 UI_R

UNSIGNED INTEGER to REAL converter

Symbol



Brief description

Conversion of an UNSIGNED INTEGER variable to a REAL variable

Method of operation

This block converts an UNSIGNED INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UINT	
Y	Output variable	0	DINT	

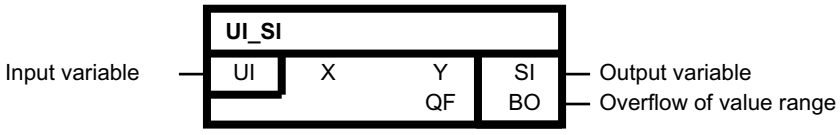
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.42 UI_SI

UNSIGNED INTEGER to SHORT INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED INTEGER variable to a SHORT INTEGER variable

Method of operation

This block converts an UNSIGNED INTEGER variable to a SHORT INTEGER variable, i.e. the least significant byte of the UNSIGNED INTEGER input variable is applied to output variable Y. If the value of input variable X exceeds the value range of output variable Y, then QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	UINT	
Y	Output variable	0	SINT	
QF	Overflow of value range	0	0/1	

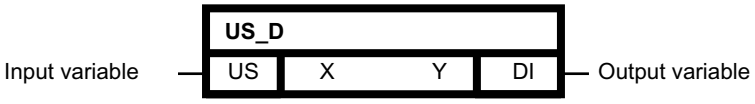
Configuration data

SIMOTION	✓
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.5.43 US_D

UNSIGNED SHORT INTEGER to DOUBLE INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED SHORT INTEGER variable to a DOUBLE INTEGER variable

Method of operation

This block converts an UNSIGNED SHORT INTEGER variable to a DOUBLE INTEGER variable.

The input variable of data type UNSIGNED SHORT INTEGER is copied to the least significant byte of output variable Y. The remaining most significant bytes are filled with 16#00.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	USINT	
Y	Output variable	0	DINT	

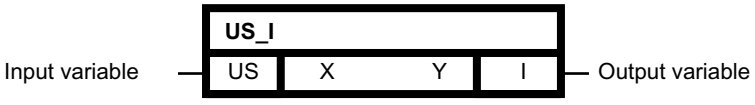
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.44 US_I

UNSIGNED SHORT INTEGER to INTEGER converter

Symbol



Brief description

Conversion of an UNSIGNED SHORT INTEGER variable to an INTEGER variable

Method of operation

This block converts an UNSIGNED SHORT INTEGER variable to an INTEGER variable.

The input variable of data type UNSIGNED SHORT INTEGER is copied to the least significant byte of output variable Y. The remaining most significant bytes are filled with 16#00.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	USINT	
Y	Output variable	0	INT	

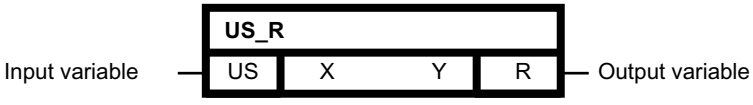
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.45 US_R

UNSIGNED SHORT INTEGER to REAL converter

Symbol



Brief description

Conversion of an UNSIGNED SHORT INTEGER variable to a REAL variable

Method of operation

This block converts an UNSIGNED SHORT INTEGER variable to a REAL variable.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	USINT	
Y	Output variable	0.0	REAL	

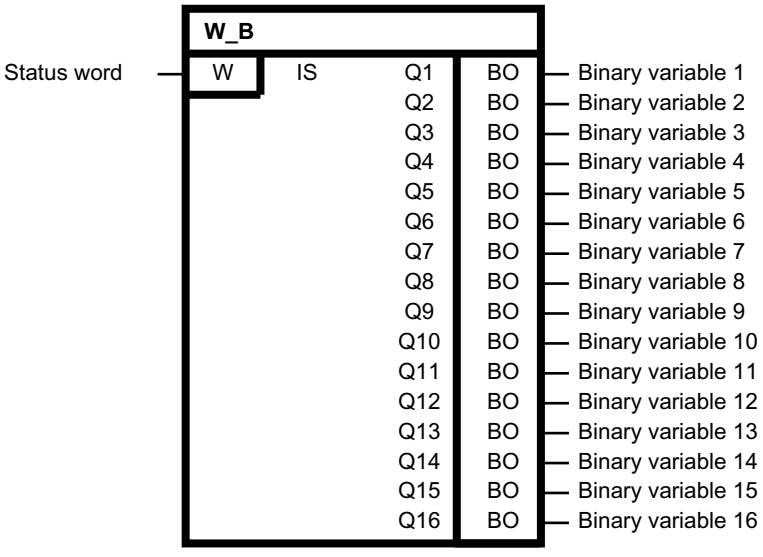
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.46 W_B

Status word to 16 binary variables converter

Symbol



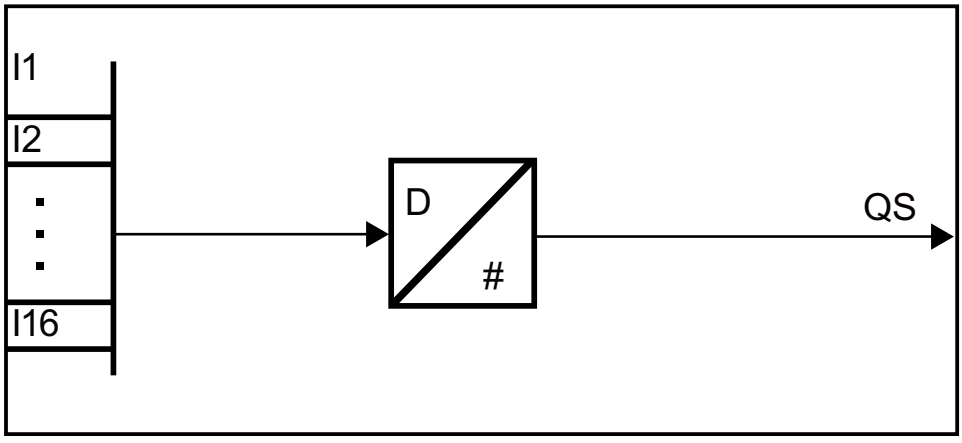
Brief description

Status word decoding to 16 binary variables

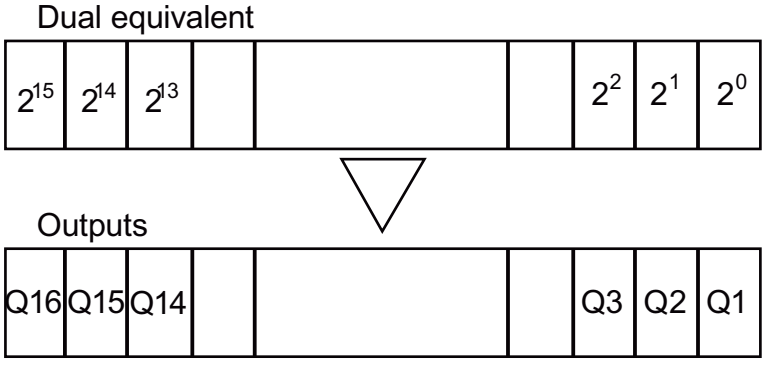
Method of operation

This block decodes the status word IS to 16 binary variables and gives the result to its outputs Q1 to Q16.

The binary variable of outputs Q1 to Q16 is assigned to each dual equivalent 2^0 to 2^{15} of the status word.



Conversion scheme



Block connections

Block connection	Description	Default	Value range	Attributes
IS	Status word	16#0000	WORD	
I1	Binary variable 1	0	0/1	
I2	Binary variable 2	0	0/1	
I3	Binary variable 3	0	0/1	
I4	Binary variable 4	0	0/1	
I5	Binary variable 5	0	0/1	
I6	Binary variable 6	0	0/1	
I7	Binary variable 7	0	0/1	

Block connection	Description	Default	Value range	Attrib-utes
I8	Binary variable 8	0	0/1	
I9	Binary variable 9	0	0/1	
I10	Binary variable 10	0	0/1	
I11	Binary variable 11	0	0/1	
I12	Binary variable 12	0	0/1	
I13	Binary variable 13	0	0/1	
I14	Binary variable 14	0	0/1	
I15	Binary variable 15	0	0/1	
I16	Binary variable 16	0	0/1	

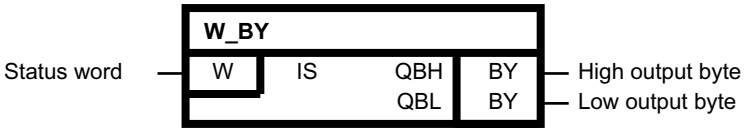
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	Yes
Special character-istics	-

8.1.5.47 W_BY

Status word to status byte converter

Symbol



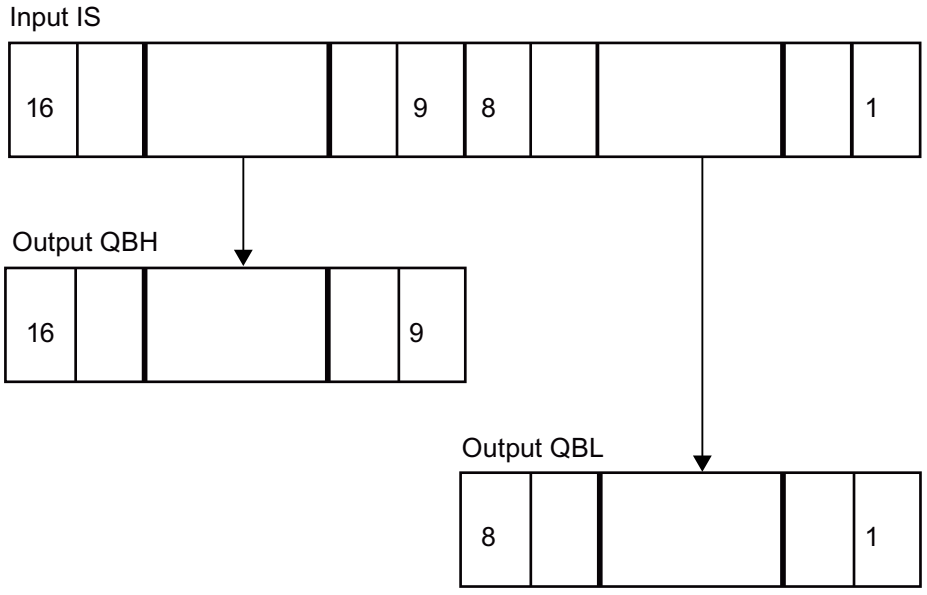
Brief description

Conversion of a word to 2 bytes

Method of operation

This block splits the input word at IS into two bytes. These can be output to the I/O via the SBQ block. The most significant byte of the word at input IS is output at output QBH, and the least significant byte of the word at input IS is output at output QBL (see conversion scheme below):

Conversion scheme



Block connections

Block connection	Description	Default	Value range	Attributes
IS	Status word	16#0000	WORD	
QBH	High output byte	16#00	BYTE	
QBL	Low output byte	16#00	BYTE	

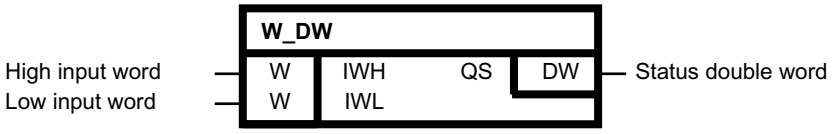
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	Yes
Special characteristics	-

8.1.5.48 W_DW

Status word to status double word converter

Symbol



Brief description

Two 16-bit words are copied to one 32-bit double word.

Method of operation

The input variables are mapped according to the formula

$QS = (IWL + IWH) * 2^{16}$

to output QS.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
IWH	High input word	16#0000	WORD	
IWH	Low input word	16#0000	WORD	
QS	Status double word	16#00000000	DWORD	

Configuration data

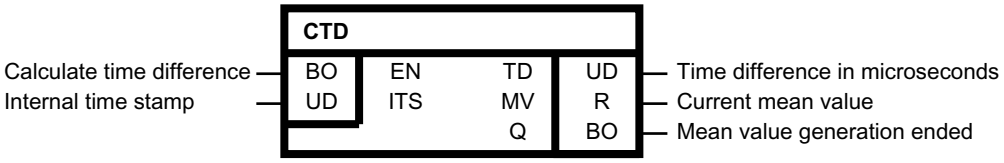
SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	Yes
Special character-istics	-

8.1.6 System

8.1.6.1 CTD

Time difference determination from an internal time stamp

Symbol



Brief description

Block for determining a time difference in microseconds.

Method of operation

If EN = 1, the time difference relative to time stamp ITS is determined and output at output TD. Time stamp ITS must be determined beforehand with block GTS. A positive edge at EN starts the mean value generation of TD, and the output is output at MV. After 10000 mean value determinations, the mean value generation ends and output Q is set to 1. If input EN = 0 is set, the mean value generation and output Q is reset. Outputs TD and MV retain their last value.

Block connections

Block connection	Description	Default	Value range	Attributes
EN	Calculate time difference	0	0/1	
ITS	Internal time stamp	0	UDINT	
TD	Time difference in microseconds	0	UDINT	
MV	Current mean value	0	REAL	
Q	Mean value generation ended	0	0/1	

Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.6.2 GTS

Reads out a time stamp

Symbol

Brief description

Blocks for reading out an internal time stamp for determination of runtimes. The determined time stamp can then be indicated at the CTD block for calculating a time difference in microseconds.

Method of operation

If EN = 1, an internal time stamp is determined and output at output TS. If EN = 0 is predefined, the last determined time stamp is output at TS.

Block connections

Block connection	Description	Default	Value range	Attributes
EN	Output time stamp	0	0/1	
ITS	Internal time stamp	0	UDINT	

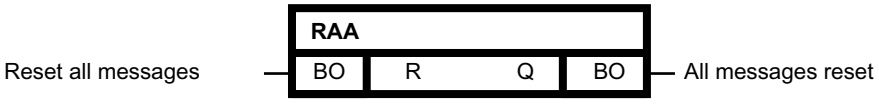
Configuration data

SIMOTION	✓
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.6.3 RAA

Reset all messages

Symbol



Brief description

All active messages are reset with the RAA (Reset all Alarms) block.

Method of operation

As long as input R = 1, all active messages are reset. Output Q indicates that the reset has been performed.

Block connections

Block connection	Description	Default	Value range	Attributes
R	Reset all messages	0	0/1	
Q	All messages reset	0	0/1	

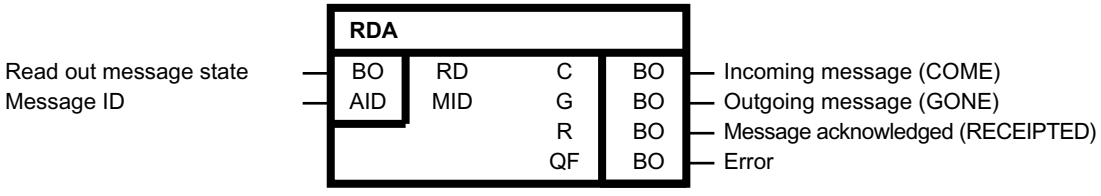
Configuration data

SIMOTION	✓ (as of V4.3)
SINAMICS	-
Can be inserted online	Yes
Special characteristics	-

8.1.6.4 RDA

Reads out a message

Symbol



Brief description

The block reads the state of a message and its acknowledgement state.

Method of operation

- The message is configured in SIMOTION SCOUT and referenced via a project-wide unique ID.
- Input MID contains the message ID, e.g. _alarm.Message.

- The state of the message is determined as long as input RD=1.
- A change of the message ID is possible. The state of the message ID specified at input MID is read out in each cycle.
- The outputs display the state of the message. The following combinations are possible:

C (incoming message)	G (outgoing message)	R (message acknowledged)	Meaning
0	1	0	Outgoing message, not acknowledged
1	0	0	Incoming message, not acknowledged
1	0	1	Incoming message, acknowledged
0	0	0	Message not in the message buffer *)

*) Message not in the message buffer - there are three options:

- Message never triggered.
- Message triggered via AlarmS, but also sent.
- Message triggered via _AlarmSq, sent and acknowledged at the display device.

The outputs are refreshed as long as RD=1. With RD=0, the last state of the message buffer is retained. Output Q is set when an error occurs, e.g. message ID not configured.

Block connections

Block connection	Description	Default	Value range	Attributes
RD	Read out message state	0	0/1	
MID	Message ID	STRUCTALARMID#NIL	StructAlarmId	
C	Incoming message (COME)	0	0/1	
G	Outgoing message (GONE)	0	0/1	
R	Message acknowledged (RECEIPTED)	0	0/1	
QF	Error	0	0/1	

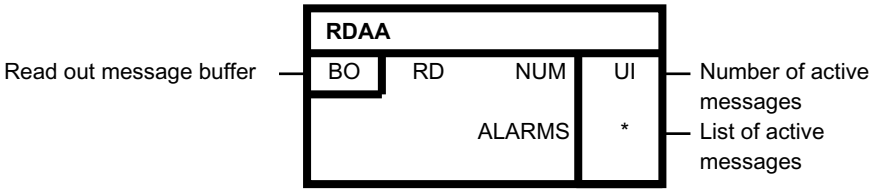
Configuration data

SIMOTION	✓ (as of V4.3)
SINAMICS	-
Can be loaded online	Yes
Special characteristics	-

8.1.6.5 RDAA

Reads out all messages

Symbol



Brief description

The list of active messages in the SIMOTION target device is read out.

Method of operation

- The reading out of all active messages is initiated with a rising edge at input RD.
- The number of active messages is returned at NUM output.
- A field of up to 40 active messages is displayed at the ALARMS output.
  The following is displayed for each alarm:
  - The message ID
  - The identifier for message not acknowledgeable (0), acknowledgeable message (1)
  - The state of the alarm: OUTGOING_ALARM (0), INCOMING_ALARM (1)
- The outputs are refreshed as long as RD=1. With RD=0, the last state of the message buffer is retained.

Block connections

Block connection	Description	Default	Value range	Attributes
RD	Read out message buffer	0	0/1	
NUM	Number of active messages	0	0..40	
ALARMS	List of active messages			
ALARMS[]	Up to 40 messages can be active	0		
ALARMS[].Id	Message ID	STRUCTALAR-MID#NIL	StructAlarmId	
ALARMS[].type	Corresponds to enumAlarmIdType (0: ALARM_S, 1: ALARM_SQ)	0	0/1	
ALARMS[].InOut	Corresponds to enumAlarmState OUTGO-ING_ALARM (0), INCOMING_ALARM (1)	0	0/1	

Configuration data

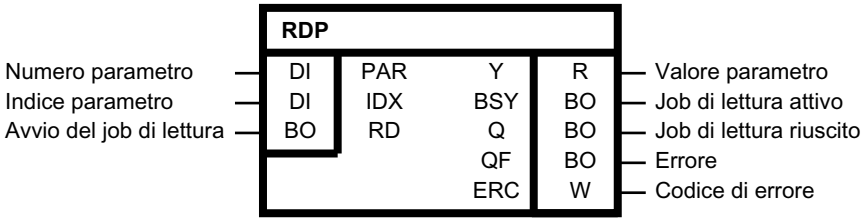
SIMOTION	✓ (as of V4.3)
SINAMICS	-

Can be inserted online	Yes
Special characteristics	-

8.1.6.6 RDP

Reads drive parameters (REAL type)

Symbol



Brief description

The block enables the asynchronous reading of drive parameters of the REAL type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

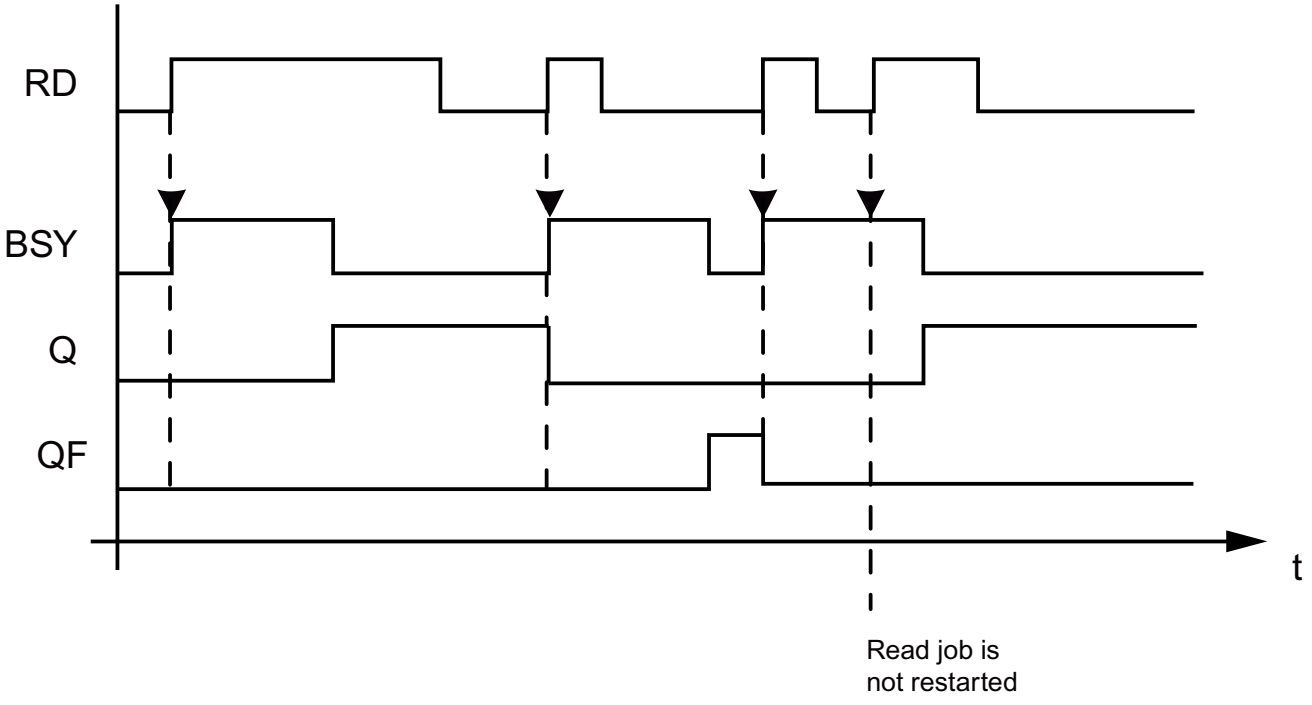
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0.0	REAL	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	DWORD	

Configuration data

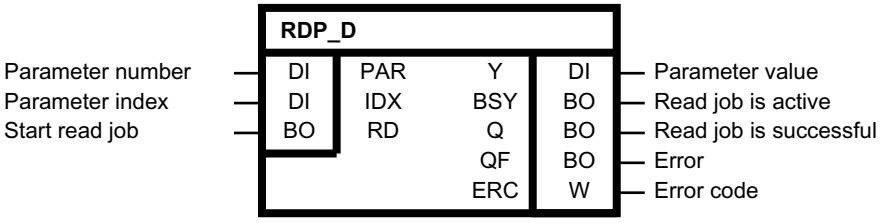
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.7 RDP_D

Reads drive parameters (DOUBLE INTEGER type)

Symbol



Brief description

The block enables the asynchronous reading of drive parameters of the DOUBLE INTEGER type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

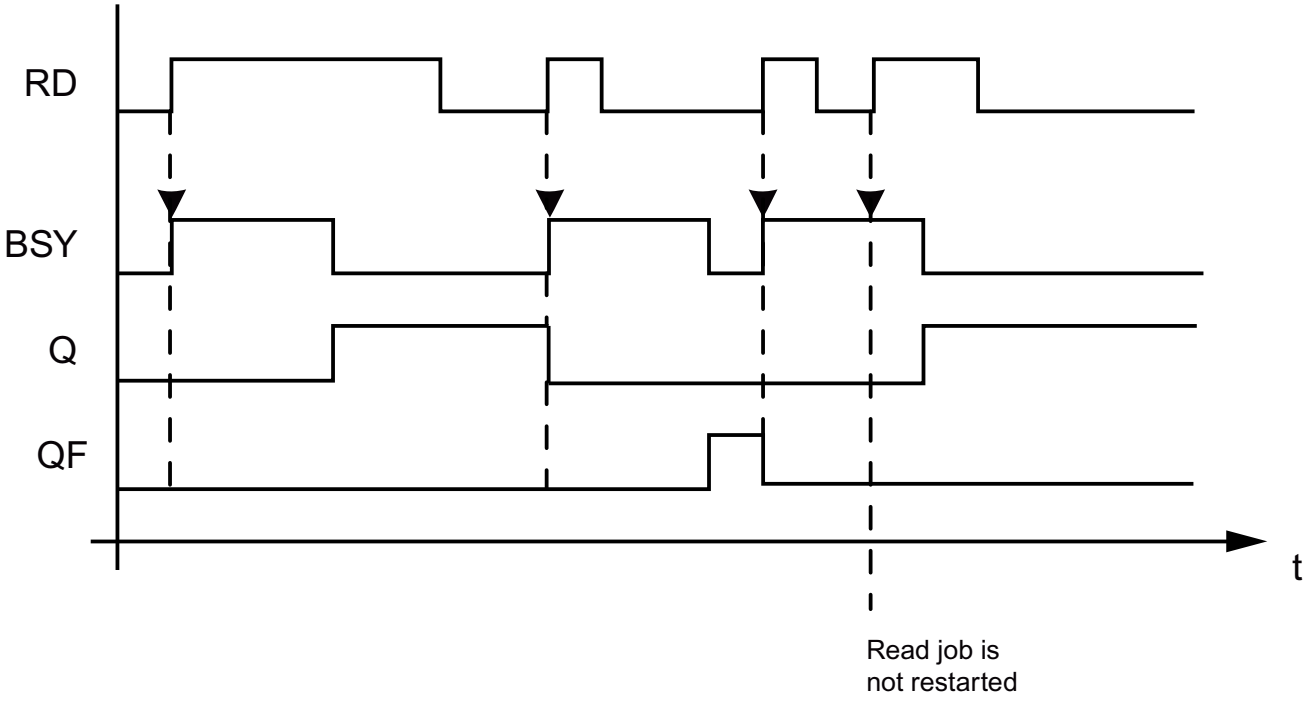
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0	DINT	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

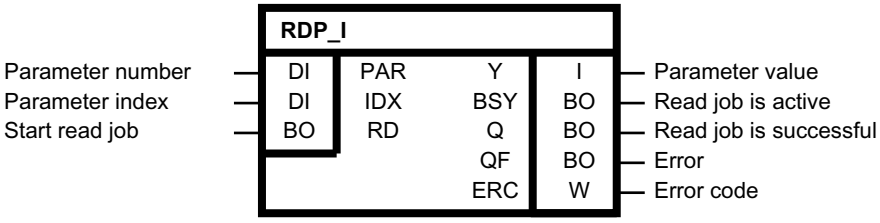
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.8 RDP_I

Reads drive parameters (INTEGER type)

Symbol



Brief description

The block enables the asynchronous reading of drive parameters of the INTEGER type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

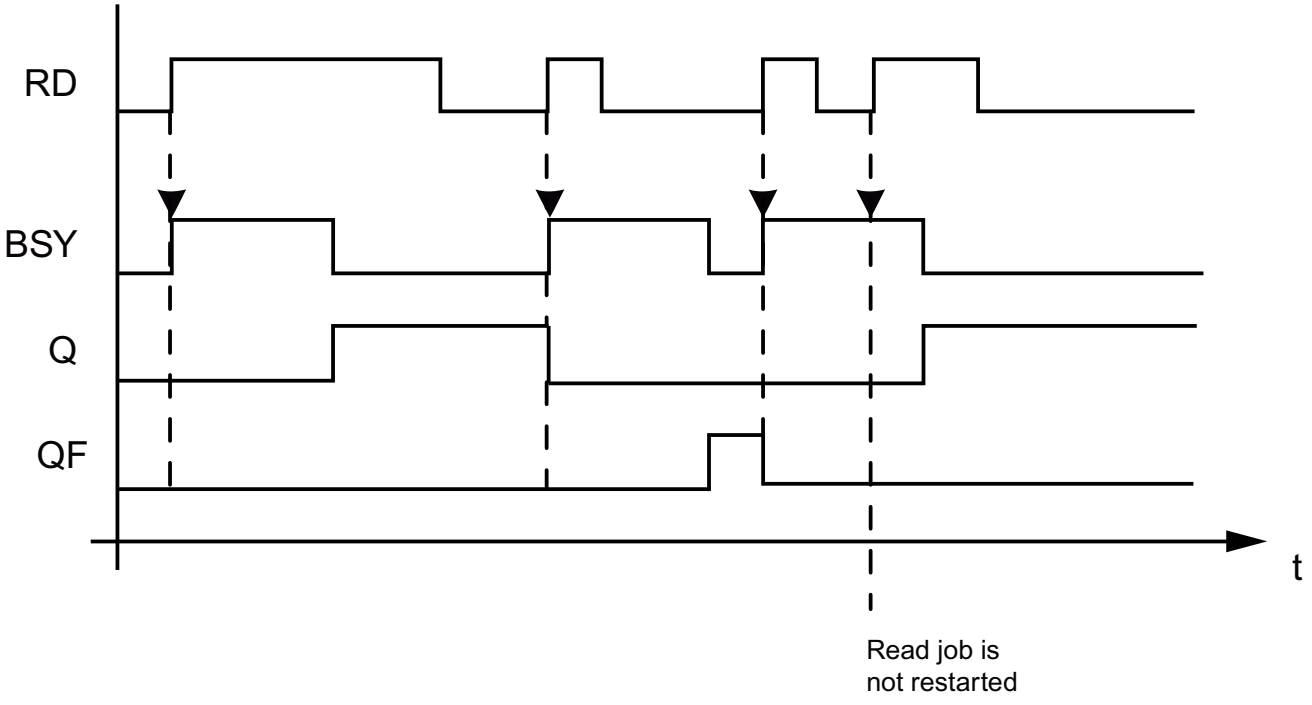
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0	INT	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

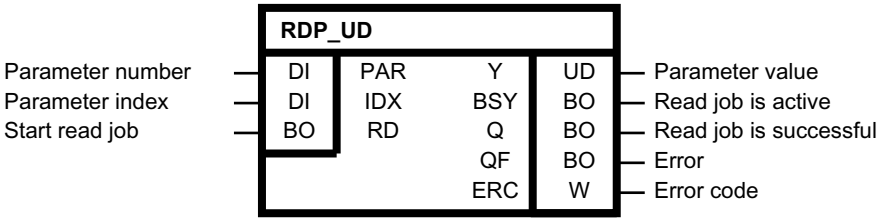
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.9 RDP_UD

Reads drive parameters (UNSIGNED DOUBLE INTEGER type)

Symbol



Brief description

RDP_UD (Read Parameter) enables the asynchronous reading of drive parameters of the UNSIGNED DOUBLE INTEGER type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

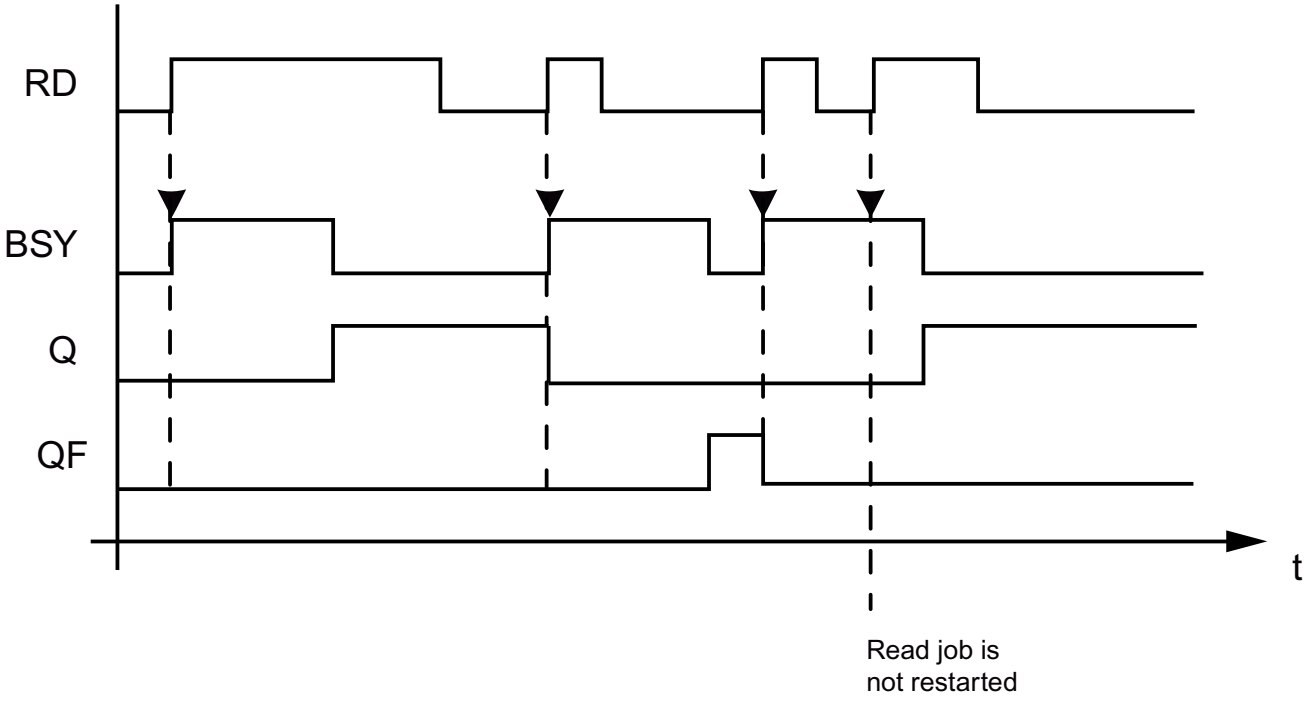
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0	UDINT	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

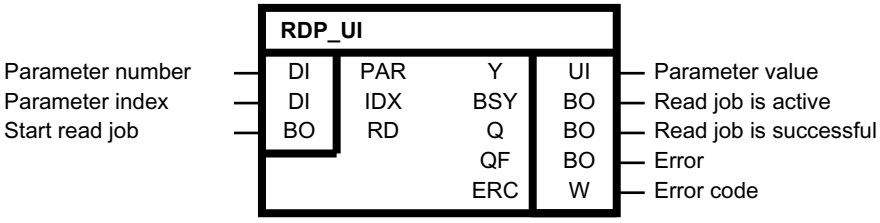
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.10 RDP_UI

Reads drive parameters (UNSIGNED INTEGER type)

Symbol



Brief description

RDP_UI (Read Parameter) enables the asynchronous reading of drive parameters of the UNSIGNED INTEGER type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

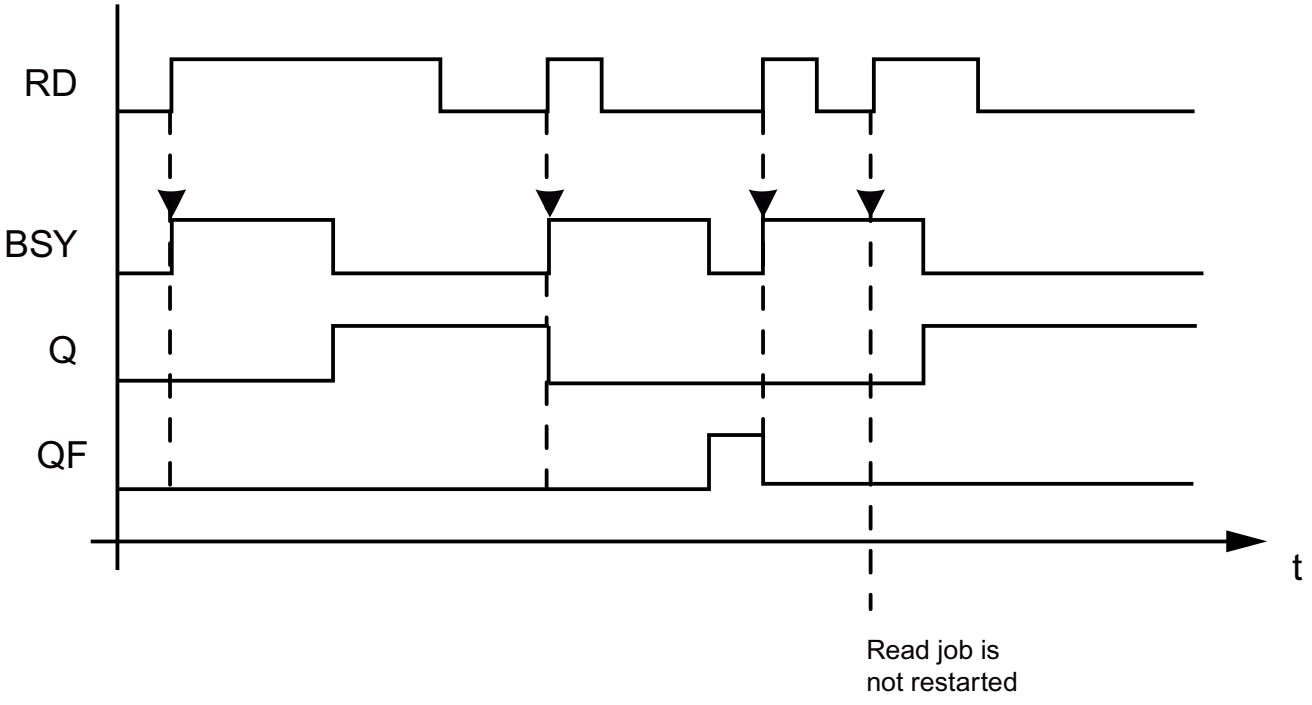
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0	UINT	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

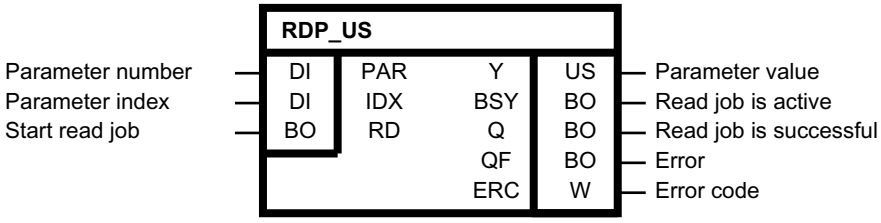
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.11 RDP_US

Reads drive parameters (UNSIGNED SHORT INTEGER type)

Symbol



Brief description

RDP_US (Read Parameter) enables the asynchronous reading of drive parameters of the UNSIGNED SHORT INTEGER type on the local drive object.

Method of operation

The parameter number and the index of the parameter to be read must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always read on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

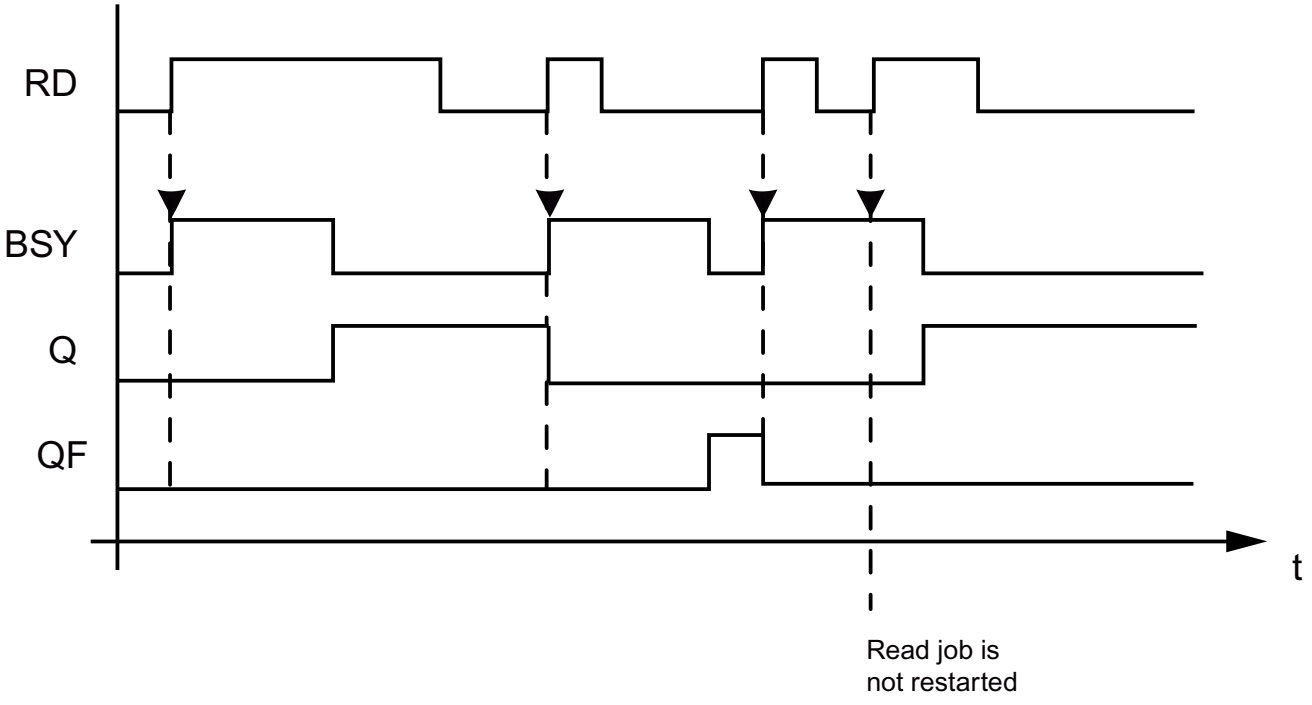
The asynchronous read job is started on a positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored.

Output Q = 1 indicates that the parameter has been read successfully and the value is available at output Y. Y holds its value until a new value has been read. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
RD	Start read job	0	0/1	
Y	Parameter value	0	USINT	
BSY	Read job is active	0	0/1	
Q	Read job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

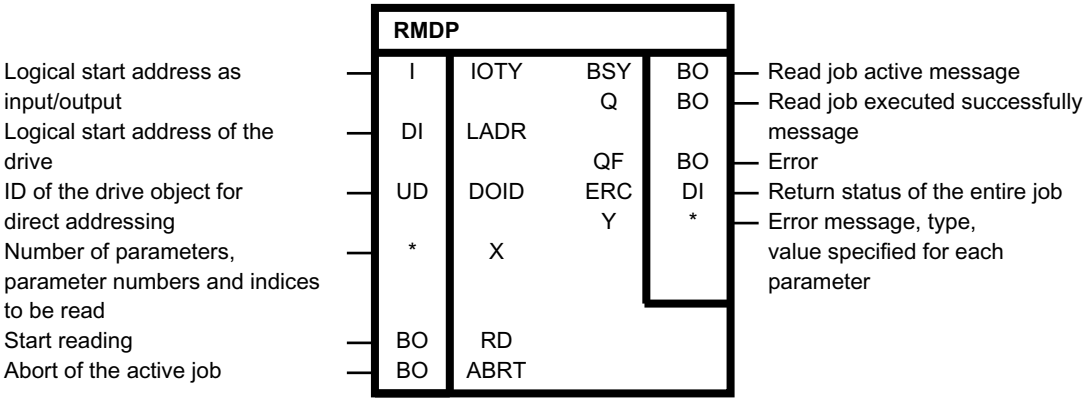
SIMOTION	-
SINAMICS	✓

Can be loaded online	No
Special characteristics	-

8.1.6.12 RMDP

Reads drive parameters from the controller

Symbol



Brief description

The RMDP block enables up to 39 SINAMICS parameters to be read from the DCC SIMOTION program. Only SINAMICS drives are supported. For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SIMOTION Communication System Manual in Section PROFIdrive and there in the Subsection Acyclic communication (Base Mode Parameter Access) → Error evaluation in table Error codes in Base Mode Parameter Access responses.

The RMDP block is available as of SIMOTION V4.2.

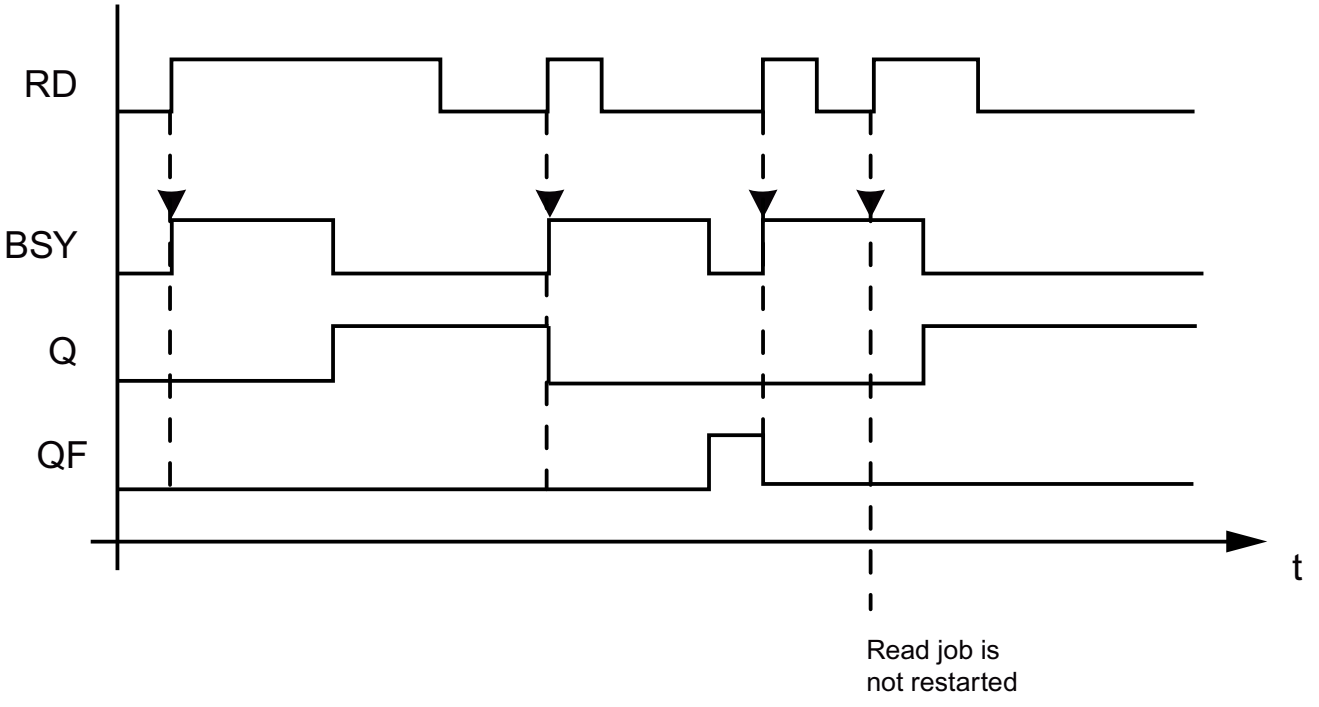
Method of operation

First the block inputs for addressing the drive are entered as well as the selection of the parameters to be read. The asynchronous read job is started by the positive edge at input RD. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and communication load and can vary from job to job. During an active read job, any additional positive edges at input RD are ignored. The actual reading/writing of the parameters is not performed in the DCC task. The block instance only controls the communication command. The results of the read/write job must be polled at the block outputs in the following task cycles. The evaluation is performed via global variables or user-defined block types. Output Q = 1 indicates that the parameters have been read successfully and the values are available at output Y. Y retains its value until a new job has been completed successfully. If an error occurs during an access, this is signaled with QF = 1. Output Y retains its last value. For error diagnostics, the error code ERC can be evaluated.

If the entire job is successful (output Q=1) and individual jobs are not successful (ERC not equal to 0), the values that are read from the drive are displayed. The other parameters with ERC = 0 have been read error-free.

The error status of the individual read jobs can be evaluated on the parameter-specific return value PRES. An active job is aborted with the positive edge on the ABRT input. The ABRT signal must have the value 1 for at least one cycle.

Time diagram



Brief description

The block enables up to 39 SINAMICS parameters to be read.

Data set 47 is always read out for PROFIBUS (external or integrated) irrespective of whether the function is called with a valid (0>=doId<=254) or invalid 'doId' (doId=255).

Two data sets are available for PROFINET:

- Base Mode Parameter Access - local (data set 0xB02E)
- Base Mode Parameter Access - global (data set 0xB02F)

Data set 0xB02E is used for SIMOTION if either no 'dold' or an invalid 'dold' (dold=255) is specified in the function. Access to the appropriate DO is then performed via the Parameter Access Point (PAP). Either the PAP address can be specified directly or the log. address of the cyclic data (e.g. 256 for a DO axis) can be specified. SIMOTION then determines the corresponding PAP from this address before accessing the correct address. PAP must always be at subslot 1 (HW Config configuration).

Data set 0xB02F is used if a valid 'dold' (0>=doId<=254) is entered. Any valid PAP or address can be specified because the assignment is only performed via the 'doId'.

Description of the block inputs

'IOTY' input/output assignment of the logical start address of the drive.
With 198: INPUT, the logical address of the drive is in the input range.
With 199: OUTPUT, the logical address of the drive is in the output range.
Diagnostic addresses are always of type INPUT.

'LADR': Specification of the logical start address of the drive. If the optional parameter DOID is also used, any arbitrary address of the station (preferably the diagnostics address of the station) can be specified.
With PROFINET, parameter access is via the Parameter Access Point (PAP) of a drive object.
As an alternative to the logical start address of the drive, specification of the diagnostic address of the associated PAP is recommended.

'DOID': For the direct addressing of a drive object. The DOID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DOID is not supported by the DP slave / IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to be via the parameter access point of a DO (PROFINET only).

'X': The parameters to be read are specified under input X.

'X.NUMP': Number of parameters to be read.

'X.PAR[].NUM': Specifies the parameter numbers from which the values are to be read.

'X.PAR[].IDX': Parameter index; for indexed values, 0 means index 0. For non-indexed values, parameter index 0 must be specified.

'RD': Start read job

'ABRT': Abort active job

Description of the block inputs

'Q': Job completed without errors.

'QF': Job completed with errors.

'ERC': Corresponds to the values of the return value 'functionResult' of the _readDriveMultiParameter function.

'Y': Description of the parameter values. An error code, the data type and the value are read out for each parameter. Further information on the return value parameterResult can be found in the SIMOTION List Manual System Functions/Variables Devices → System Functions - Devices 1 → _readDriveMultiParameterDescription

'Y[].PRES': Corresponds to the parameter-specific return value. Coding corresponds to the return parameter parameterResult of the _readDriveMultiParameter ST function. Further information on the return parameter parameterResult can be found in the SIMOTION List Manual System Functions/Variables Devices → System Functions - Devices 1 → _readDriveMultiParameterDescription

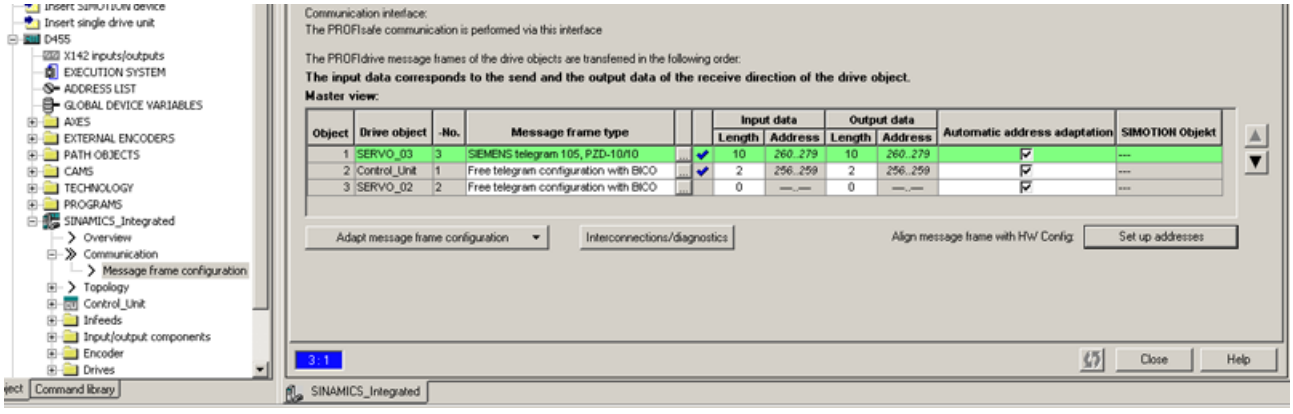
'Y[].DTYP': Returns the data type of the parameter (for the coding, see PROFIdrive profile).

'Y[].VAL': Parameter values read from the drive; the data type results from the returned data type. A conversion block must be called for different data types. When accessing REAL parameters, the conversion is performed via the conversion block DW_R.

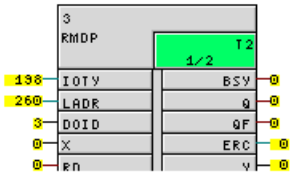
Parameterization example

In order to be able to write certain parameters of a drive object (in the example: SERVO_03), proceed as follows:

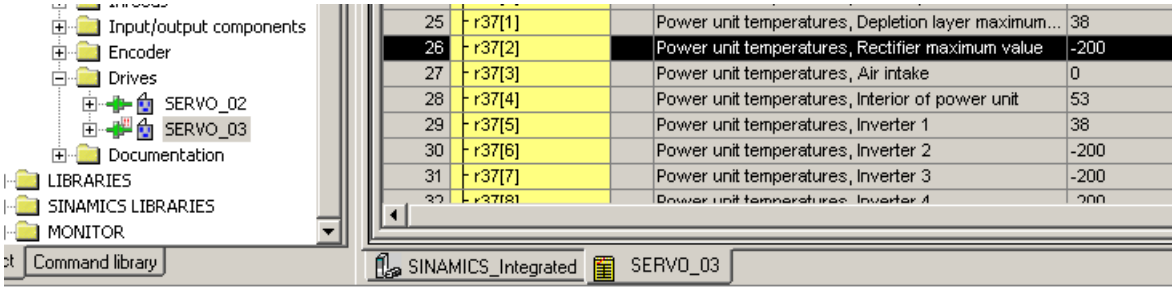
First set the correct telegram configuration.



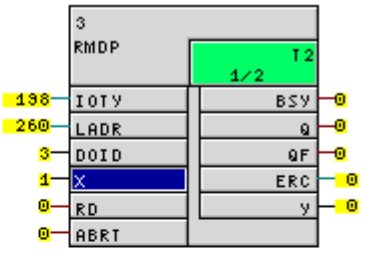
Then set the desired DO address in the RMDP block. To do this, set the block input 'LADR' to the address (260) set in the telegram and the block input 'DOID' to the number (3) set in the telegram.

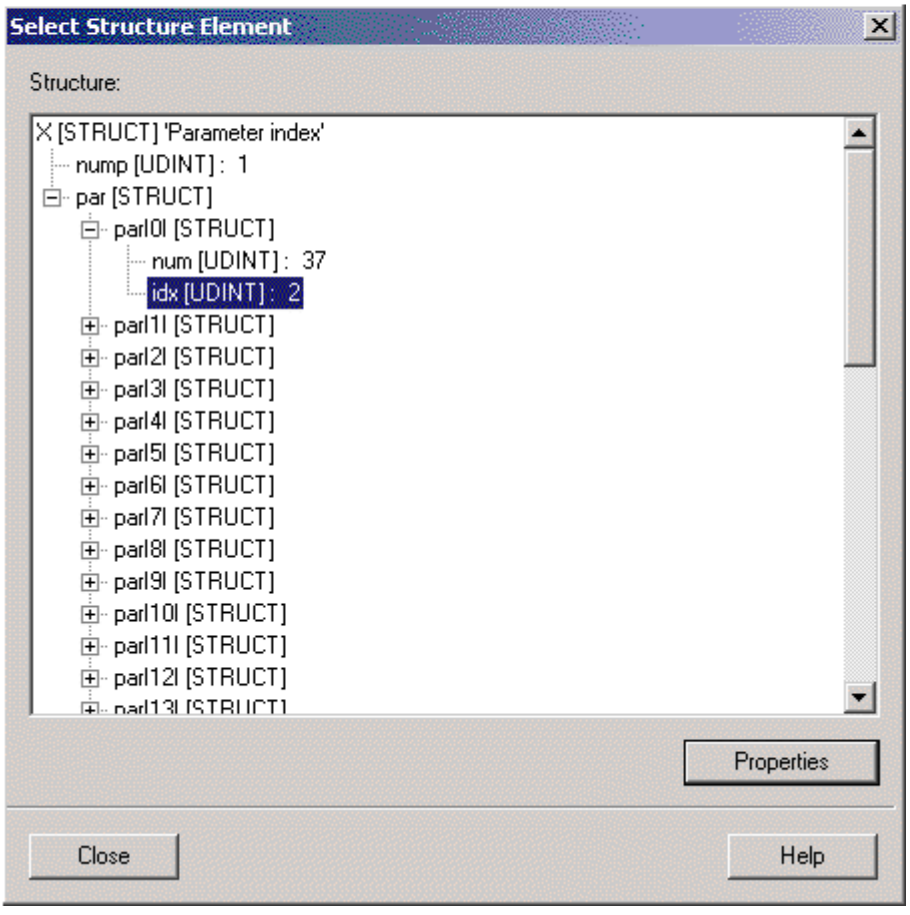


Select the parameters to be read at block input 'X', e.g. r37(2) power unit temperatures, rectifier maximum value in the expert list.

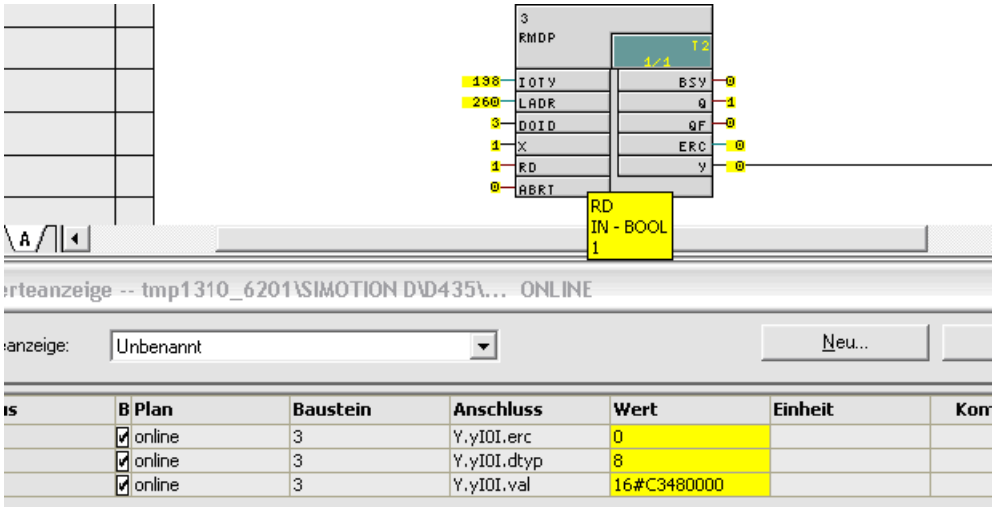


To do this, double-click block input 'X', select the first structure element and enter the parameter number (37) at 'num' and the index (2) at 'idx'.





Then set the block input 'RD' to 1 to start the reading.



Block connections

Block connection	Description	Default	Value range	Attributes
IOTY	Logical start address as input/ output	0	0: Invalid 198: Input address 199: Output address	
LADR	Logical start address of the drive	-1	DINT	
DOID	ID of the drive object for direct addressing	255	0 .. 254, 255: Invalid	
X	Number of parameters, parameter numbers and indices to be read			
X.NUMP	Number of parameters to be read	1	1..39	
X.PAR	Description of a parameter			
X.PAR[].NUM	Parameter number	1	1..65535	
X.PAR[].IDX	Parameter index	0	0..65535	
RD	Start reading	0	0/1	
ABRT	Abort of the active job	0	0/1	
BSY	Read job active message	0	0/1	
Q	Read job executed successfully message	0	0/1	
QF	Error	0	0/1	
ERC	Return status of the entire job	16#0000	DWORD	
Y	Error message, type, value specified for each parameter			
Y[].PRES	Parameter-specific return value	0	DINT	
Y[].DTYP	Data type of the read parameter	0	USINT	
Y[].VAL	Parameter value read from the drive	0	DWORD	

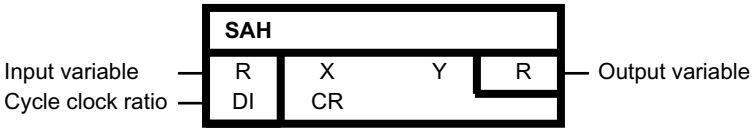
Configuration data

SIMOTION	✓ (as of V4.2)
SINAMICS	-
Can be loaded on-line	No
Process context	Cyclic, equidistant
Special characteristics	-

8.1.6.13 SAH

Sample & hold (REAL type)

Symbol



Brief description

Sample & hold block for equidistant value transfer (REAL type) between blocks with different scanning procedures.

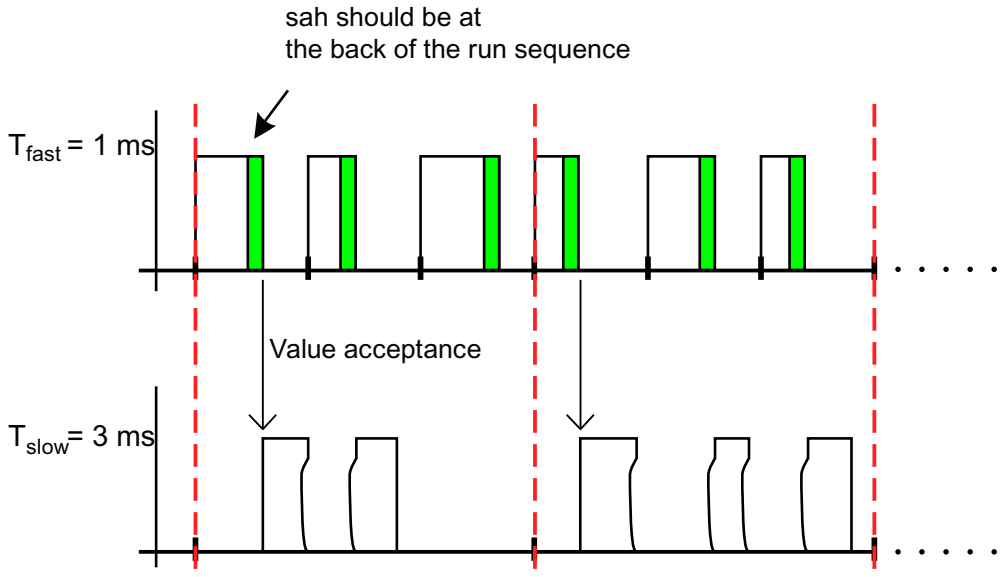
Method of operation

The value of the input variable X is taken over in the output variable Y in every CR cycle. The value transfer cycle clock is synchronized with the cycle control point of the execution system. The cycle control point defines the cycle clock in which all sampling times of the execution system are restarted.

A value transfer takes place every CR cycle clock relative to the cycle control point. The absolute value of CR is always generated for the cycle clock ratio. In the special case of CR = 0, the block behaves as for CR = 1. The block must always be configured in the faster sampling time. If the value from the slower scan time is transferred, it should be at the very beginning of the run sequence. If the value is to taken from the faster into the slower sampling time, the block should be computed last in the execution sequence.

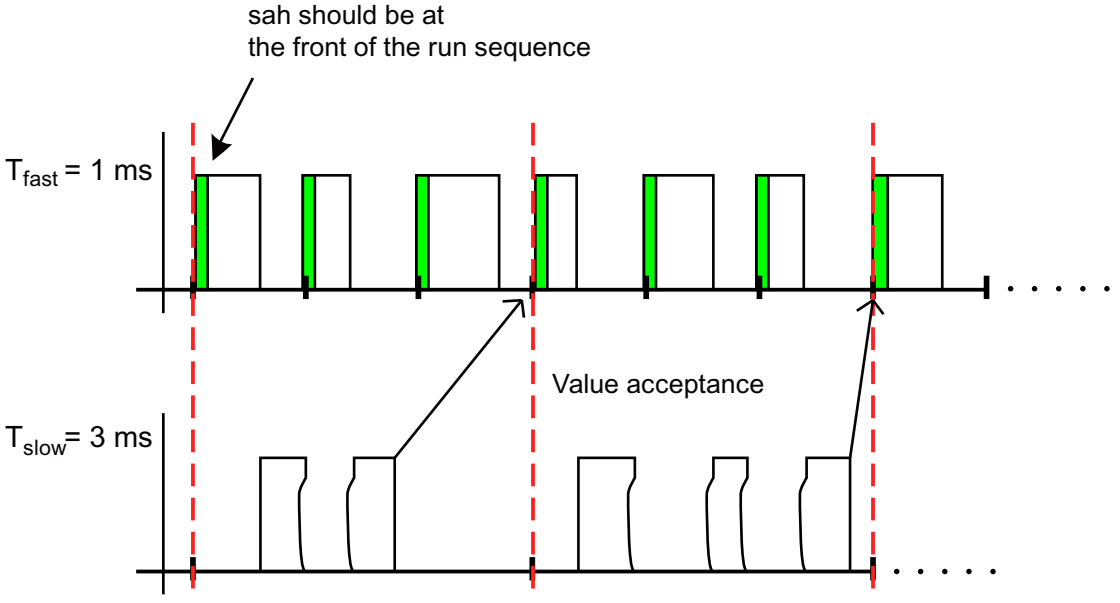
The following figure shows the transfer of values from a 1 ms level to a 3 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



The following figure shows the transfer of values from a 3 ms level to a 1 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



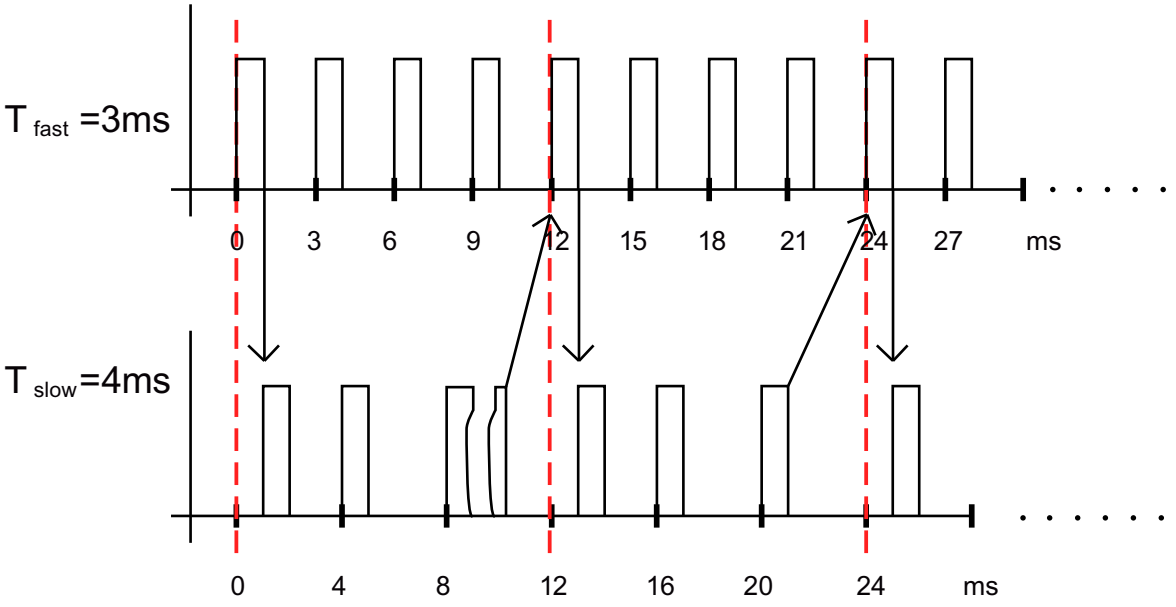
If the slower scan time is not a multiple of the faster scan time, the value can only be transferred consistently if both scanning procedures are restarted synchronously after CR cycles. This corresponds to the least common multiple of both scan times. CR is then calculated as follows:

$$CR = \frac{g\left(T_{fast}, T_{slow}\right)}{T_{fast}}$$

$g(T_{fast}, T_{slow})$: least common multiple

The following shows the value transfer for $T_{fast} = 3$ ms and $T_{slow} = 4$ ms. The value transfer is made in both directions.

$$CR = \frac{g(3ms, 4ms)}{3ms} = \frac{12ms}{3ms} = 4$$



To enable values to be transferred at an optimal speed, it is recommended that the slower scan time is always a multiple of the faster scan time.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
CR	Cycle clock ratio	1	0 - (2^{31}-1)	
Y	Output variable	0.0	REAL	

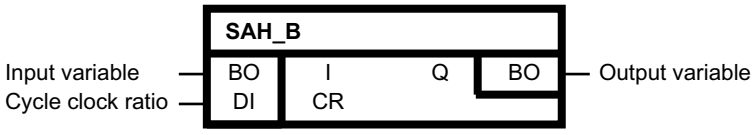
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.6.14 SAH_B

Sample & hold (BOOL type)

Symbol



Brief description

Sample & hold block for equidistant value transfer (BOOL type) between blocks with different scanning procedures.

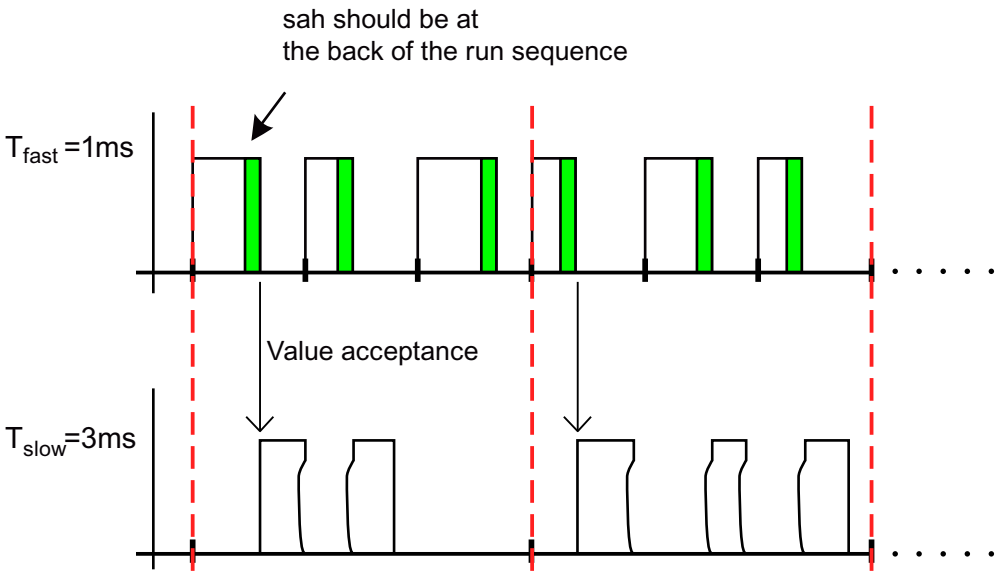
Method of operation

The value of the input variable I is taken over in the output variable Q in every CR cycle. The value transfer cycle clock is synchronized with the cycle control point of the execution system. The cycle control point defines the cycle clock in which all sampling times of the execution system are restarted.

A value transfer takes place every CR cycle clock relative to the cycle control point. The absolute value of CR is always generated for the cycle clock ratio. In the special case of CR = 0, the block behaves as for CR = 1. The block must always be configured in the faster sampling time. If the value from the slower scan time is transferred, it should be at the very beginning of the run sequence. If the value is to taken from the faster into the slower sampling time, the block should be computed last in the execution sequence.

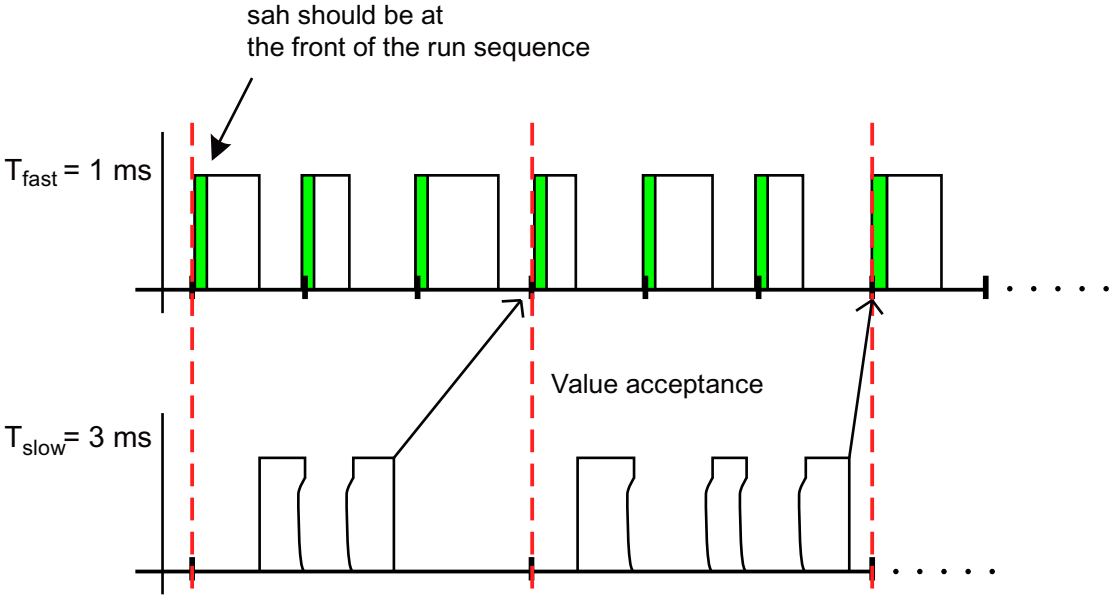
The following figure shows the transfer of values from a 1 ms level to a 3 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



The following figure shows the transfer of values from a 3 ms level to a 1 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



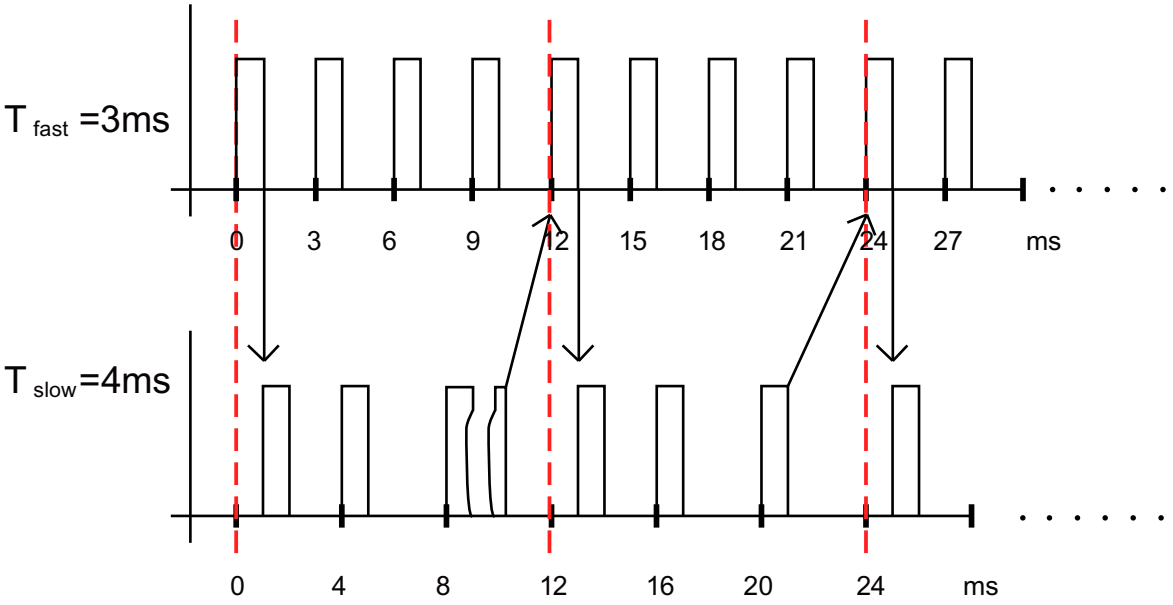
If the slower scan time is not a multiple of the faster scan time, the value can only be transferred consistently if both scanning procedures are restarted synchronously after CR cycles. This corresponds to the least common multiple of both scan times. CR is then calculated as follows:

$$CR = \frac{g\left(T_{fast}, T_{slow}\right)}{T_{fast}}$$

g(T_{fast},T_{slow}): least common multiple

The following shows the value transfer for T_{fast} = 3 ms and T_{slow} = 4 ms. The value transfer is made in both directions.

$$CR = \frac{g(3ms, 4ms)}{3ms} = \frac{12ms}{3ms} = 4$$



To enable values to be transferred at an optimal speed, it is recommended that the slower scan time is always a multiple of the faster scan time.

Block connections

Block connection	Description	Default	Value range	Attributes
I	Input variable	0	0/1	
CR	Cycle clock ratio	1	0 - (2^{31}-1)	
Q	Output variable	0	0/1	

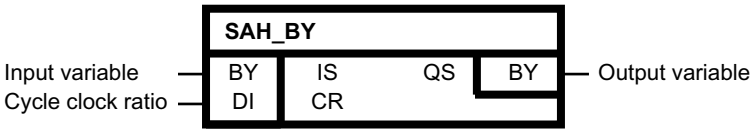
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.6.15 SAH_BY

Sample & hold (BYTE type)

Symbol



Brief description

Sample & hold block for the equidistant value transfer (BYTE type) between blocks with different sampling times.

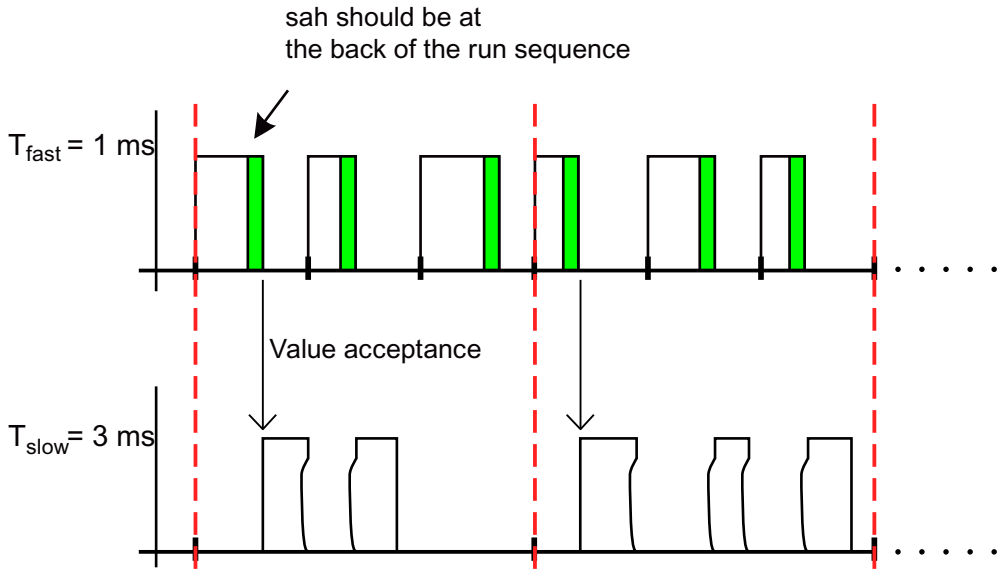
Method of operation

The value of the input variable IS is taken over in the output variable QS in every CR cycle. The value transfer cycle clock is synchronized with the cycle control point of the execution system. The cycle control point defines the cycle clock in which all sampling times of the execution system are restarted.

A value transfer takes place every CR cycle clock relative to the cycle control point. The absolute value of CR is always generated for the cycle clock ratio. In the special case of CR = 0, the block behaves as for CR = 1. The block must always be configured in the faster sampling time. If the value from the slower scan time is transferred, it should be at the very beginning of the run sequence. If the value is to taken from the faster into the slower sampling time, the block should be computed last in the execution sequence.

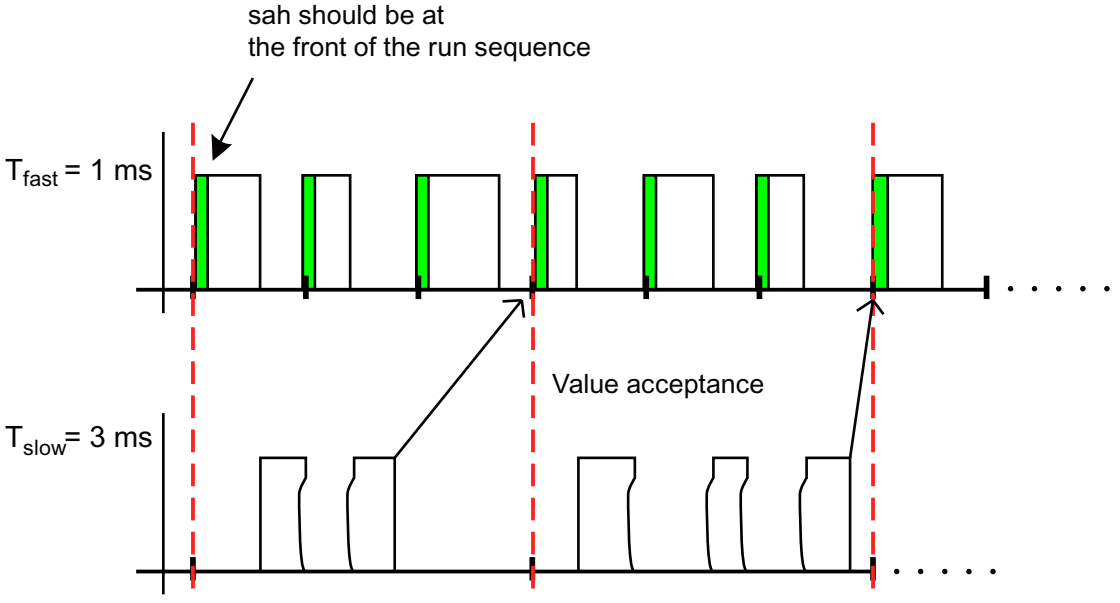
The following figure shows the transfer of values from a 1 ms level to a 3 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



The following figure shows the transfer of values from a 3 ms level to a 1 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



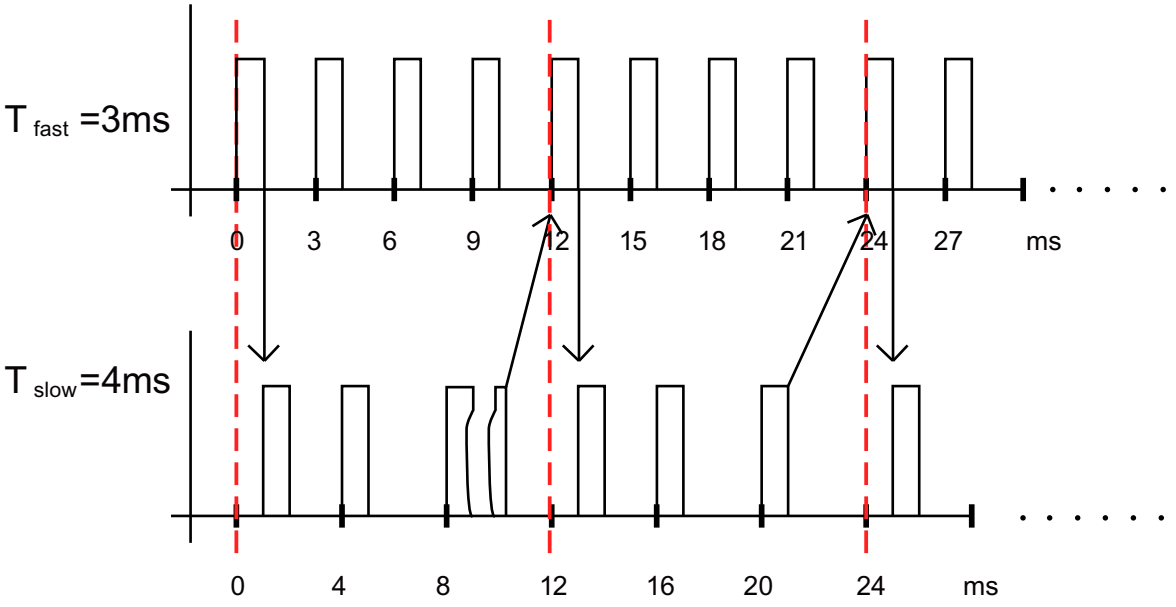
If the slower scan time is not a multiple of the faster scan time, the value can only be transferred consistently if both scanning procedures are restarted synchronously after CR cycles. This corresponds to the least common multiple of both scan times. CR is then calculated as follows:

$$CR = \frac{g\left(T_{fast}, T_{slow}\right)}{T_{fast}}$$

$g(T_{fast},T_{slow})$: least common multiple

The following figure shows the value transfer for T_{fast} = 3 ms and T_{slow} = 4 ms. The value transfer is made in both directions.

$$CR = \frac{g(3ms, 4ms)}{3ms} = \frac{12ms}{3ms} = 4$$



To enable values to be transferred at an optimal speed, it is recommended that the slower scan time is always a multiple of the faster scan time.

Block connections

Block connection	Description	Default	Value range	Attributes
IS	Input variable	16#00	BYTE	
CR	Cycle clock ratio	1	0 - (2^{31}-1)	
QS	Output variable	16#00	BYTE	

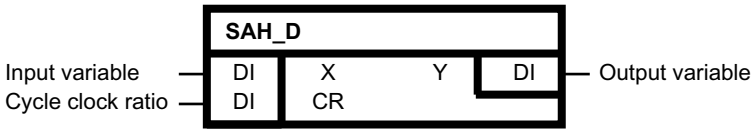
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.6.16 SAH_D

Sample & hold (DOUBLE INTEGER type)

Symbol



Brief description

Sample & hold block for equidistant value transfer (DOUBLE INTEGER type) between blocks with different scanning procedures.

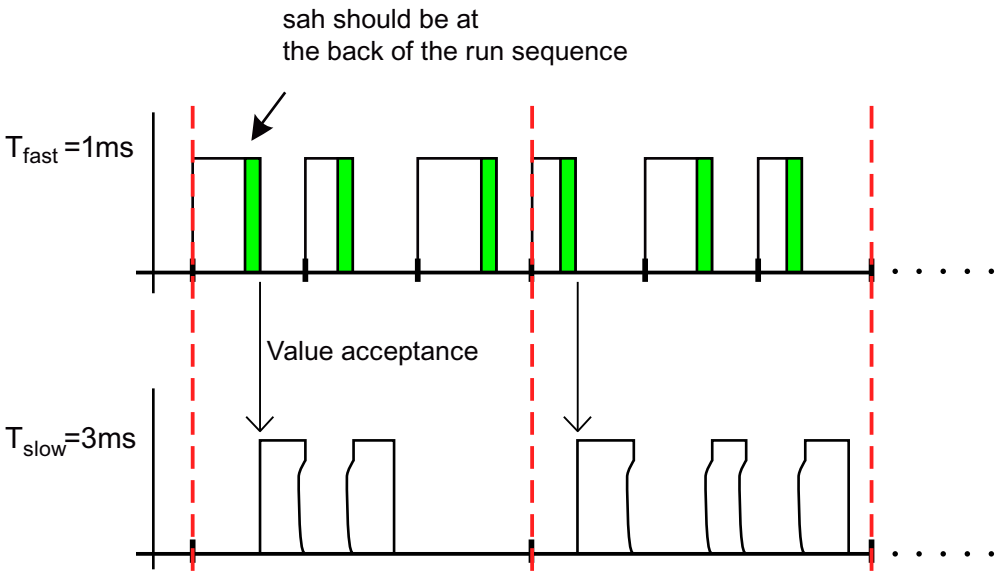
Method of operation

The value of the input variable X is taken over in the output variable Y in every CR cycle. The value transfer cycle clock is synchronized with the cycle control point of the execution system. The cycle control point defines the cycle clock in which all sampling times of the execution system are restarted.

A value transfer takes place every CR cycle clock relative to the cycle control point. The absolute value of CR is always generated for the cycle clock ratio. In the special case of CR = 0, the block behaves as for CR = 1. The block must always be configured in the faster sampling time. If the value from the slower scan time is transferred, it should be at the very beginning of the run sequence. If the value is to taken from the faster into the slower sampling time, the block should be computed last in the execution sequence.

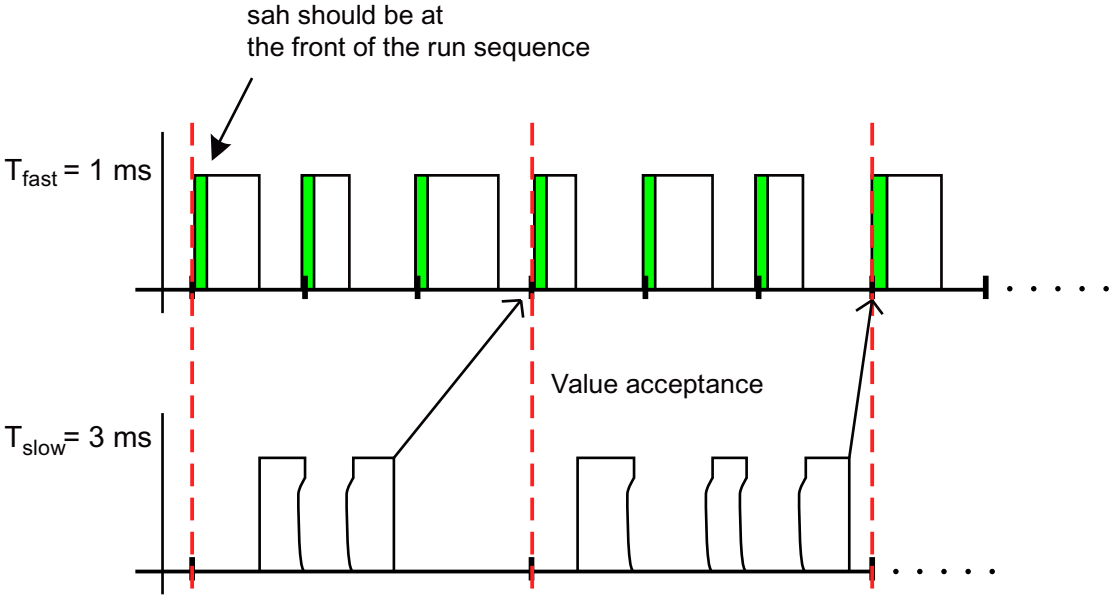
The following figure shows the transfer of values from a 1 ms level to a 3 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



The following figure shows the transfer of values from a 3 ms level to a 1 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



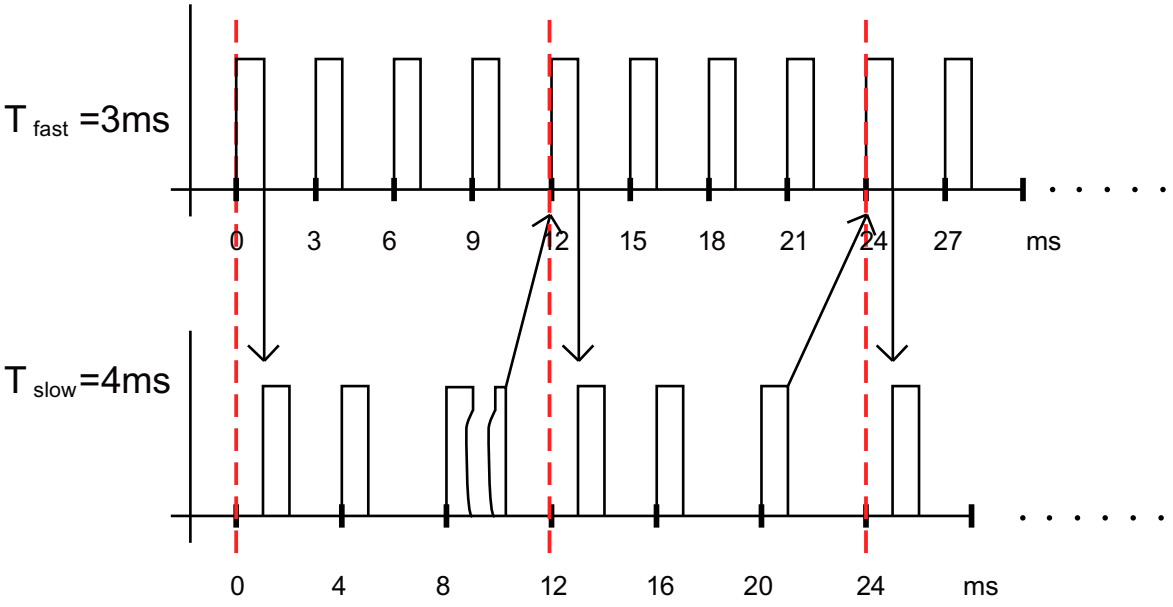
If the slower scan time is not a multiple of the faster scan time, the value can only be transferred consistently if both scanning procedures are restarted synchronously after CR cycles. This corresponds to the least common multiple of both scan times. CR is then calculated as follows:

$$CR = \frac{g\left(T_{fast}, T_{slow}\right)}{T_{fast}}$$

$g(T_{fast},T_{slow})$: least common multiple

The following figure shows the value transfer for T_{fast} = 3 ms and T_{slow} = 4 ms. The value transfer is made in both directions.

$$CR = \frac{g(3ms, 4ms)}{3ms} = \frac{12ms}{3ms} = 4$$



To enable values to be transferred at an optimal speed, it is recommended that the slower scan time is always a multiple of the faster scan time.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
CR	Cycle clock ratio	1	0 - (2^{31}-1)	
Y	Output variable	0	DINT	

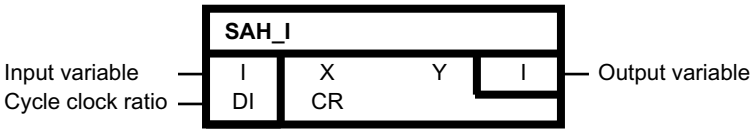
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.6.17 SAH_I

Sample & hold (INTEGER type)

Symbol



Brief description

Sample & hold block for the equidistant value transfer (INTEGER type) between blocks with different sampling times.

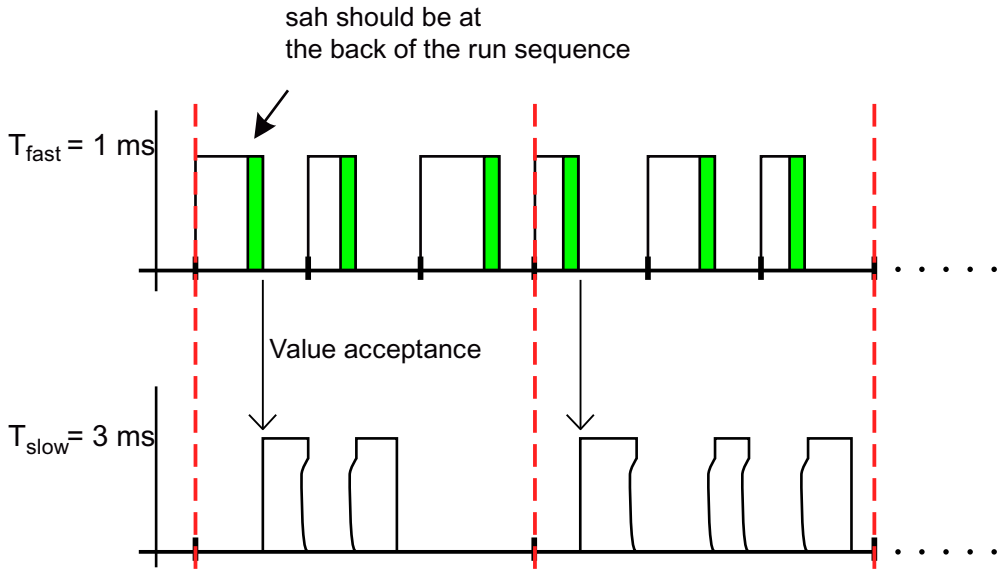
Method of operation

The value of the input variable X is taken over in the output variable Y in every CR cycle. The value transfer cycle clock is synchronized with the cycle control point of the execution system. The cycle control point defines the cycle clock in which all sampling times of the execution system are restarted.

A value transfer takes place every CR cycle clock relative to the cycle control point. The absolute value of CR is always generated for the cycle clock ratio. In the special case of CR = 0, the block behaves as for CR = 1. The block must always be configured in the faster sampling time. If the value from the slower scan time is transferred, it should be at the very beginning of the run sequence. If the value is to taken from the faster into the slower sampling time, the block should be computed last in the execution sequence.

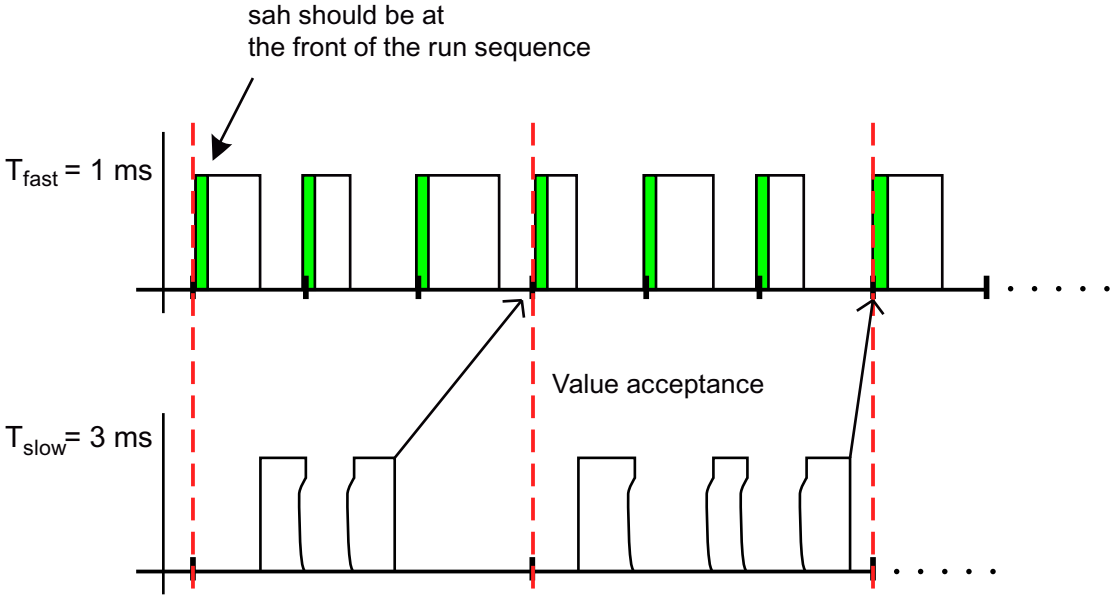
The following figure shows the transfer of values from a 1 ms level to a 3 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



The following figure shows the transfer of values from a 3 ms level to a 1 ms level. The time diagram is shown for the calculation of the execution group.

$$CR = \frac{3ms}{1ms} = 3$$



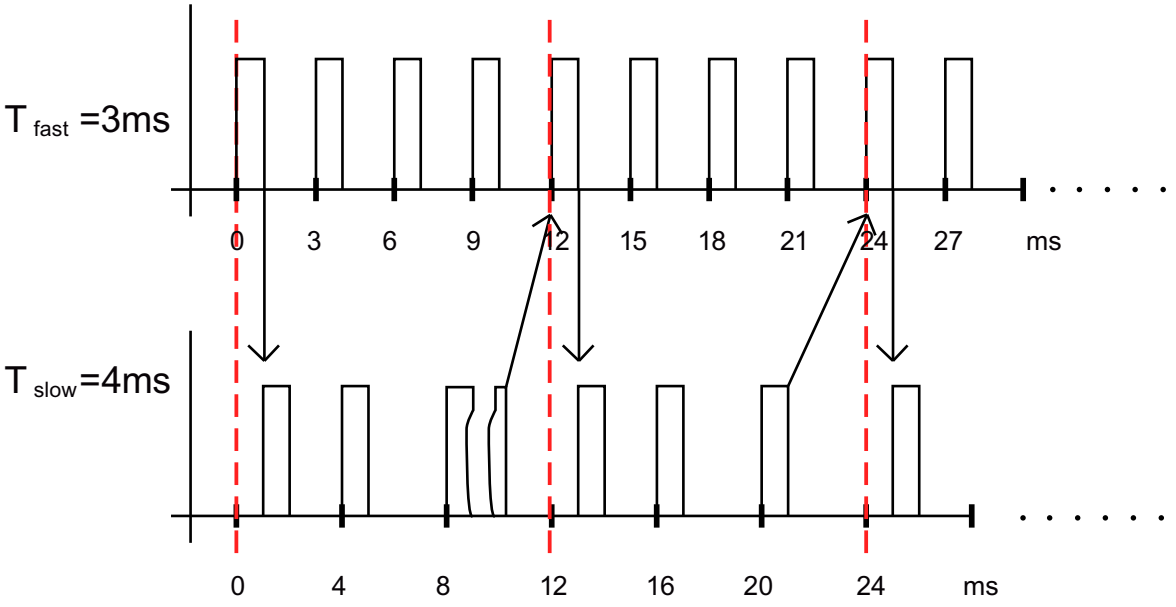
If the slower scan time is not a multiple of the faster scan time, the value can only be transferred consistently if both scanning procedures are restarted synchronously after CR cycles. This corresponds to the least common multiple of both scan times. CR is then calculated as follows:

$$CR = \frac{g\left(T_{fast}, T_{slow}\right)}{T_{fast}}$$

$g(T_{fast},T_{slow})$: least common multiple

The following figure shows the value transfer for T_{fast} = 3 ms and T_{slow} = 4 ms. The value transfer is made in both directions.

$$CR = \frac{g(3ms, 4ms)}{3ms} = \frac{12ms}{3ms} = 4$$



To enable values to be transferred at an optimal speed, it is recommended that the slower scan time is always a multiple of the faster scan time.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
CR	Cycle clock ratio	1	0 - (2^{31}-1)	
Y	Output variable	0	INT	

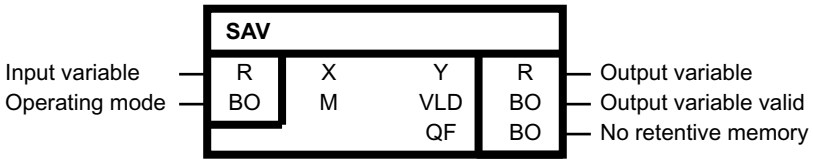
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.6.18 SAV

Value buffering (REAL type)

Symbol



Brief description

SAV (Save) enables retentive storage of a REAL-type input variable.

Method of operation

The block is a retentive read/write memory for a REAL value.

The saved value of a SAV block is not retained when:

- The retentive memory on the target device has been cleared through a user action.
- The chart on which the block was configured, has been deleted and the change transferred to the target device.
- The block has been deleted and the change transferred to the target system.
- The instance name of a block has been changed and transferred to the target system.

The value is retained when:

- The instance name does not change during a download.
- The target device ramps up without configuration data on the memory card. The memory of the missing SAV blocks is only released after a download. In this way, the data is also retained when the firmware is updated.
- Another SAV block has been added or removed.
- A download of the configuration is performed after an update of the DCBLIB.
- Another DO has been added or removed and downloaded to the target device.
- Another chart has been added or removed and downloaded to the target device.
- The target device ramps up with the same configuration as before the power failure.

The block is only active when a 0 on output QF indicates that retentive memory space is available on the target device for storing the input values.

The block mode is set at input M:

Write mode (M =1)

- Input variable X is written cyclically to output Y.
- In addition, input variable X is transferred to the system for retentive storage. In so doing, an already saved value is overwritten.

Read mode (M = 0)

- The current saved value is output at output Y. The values at input X are not saved.
- Output VLD = 1 indicates the validity of Y. If the retentive memory of the system was recreated when the block is initialized, VLD = 0. In this case, Y is invalid and contains its default value. The status of VLD changes to 1 the first time a value is written (M = 1).

Initialization

The assignment between the SAV block and the value in the retentive memory is performed via the instance name of the block. The unique instance name is automatically generated by the DCC editor when the block is inserted in a chart. The instance name is made up of the call path of the block as follows:

(Chart name)/(Name of subchart 1)/(Name of subchart 2)/../(Name of the block)

The instance name could look like this:

DCC_1/CFC1/CFC2/CFC3/SAV1

Chart name	DCC_1
Name of subchart 1	CFC1
Name of subchart 2	CFC2
Name of subchart 3	CFC3
Name of the block	SAV1

This instance name controls whether output Y is initialized with its default value or outputs the last saved value in the INIT mode. A check is made on the target device whether a retentive value has been saved for this instance name of the block. If this is not the case, the system recreates the memory space, the default value of output variable Y is transferred to the system for retentive storage and VLD is set to 0. If a retentive value has been saved for the instance name, it is read and written to output Y and status VLD = 1 is output.

If no retentive memory is available for the block, output QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
M	Operating mode	0	0/1	
Y	Output variable	0.0	REAL	
VLD	Output variable valid	0	0/1	
QF	No retentive memory	0	0/1	

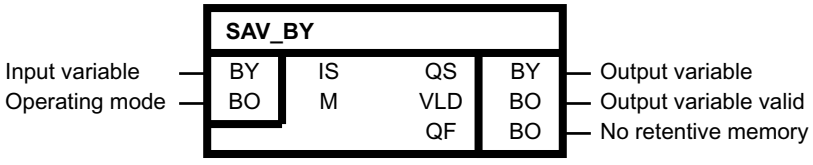
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	No
Special characteristics	A maximum of ten blocks can be used for the retentive storage (SAV, SAV_BY, SAV_I, SAV_D) for each SINAMICS or SINAMICS INTEGRATED drive unit. Retentive memory is available for a maximum of 40 bytes of user data. As of STARTER/SCOUT V4.2, the number of blocks is only checked in the consistency check (menu "Project -> Check consistency" or "Check consistency" in the shortcut menu of the drive unit).

8.1.6.19 SAV_BY

Value buffering (BYTE type)

Symbol



Brief description

SAV_BY (Save) enables retentive storage of a BYTE-type input variable.

Method of operation

The block is a retentive read/write memory for a BYTE value.

The saved value of a SAV block is not retained when:

- The retentive memory on the target device has been cleared through a user action.
- The chart on which the block was configured, has been deleted and the change transferred to the target device.
- The block has been deleted and the change transferred to the target system.
- The instance name of a block has been changed and transferred to the target system.

The value is retained when:

- The instance name does not change during a download.
- The target device ramps up without configuration data on the memory card. The memory of the missing SAV blocks is only released after a download. In this way, the data is also retained when the firmware is updated.
- Another SAV block has been added or removed.

- A download of the configuration is performed after an update of the DCBLIB.
- Another DO has been added or removed and downloaded to the target device.
- Another chart has been added or removed and downloaded to the target device.
- The target device ramps up with the same configuration as before the power failure.

The block is only active when a 0 on output QF indicates that retentive memory space is available on the target device for storing the input values.

The block mode is set at input M:

Write mode (M =1)

- Input variable IS is written cyclically to output QS.
- Input variable IS is also transferred to the system for retentive storage. In so doing, an already saved value is overwritten.

Read mode (M = 0)

- The currently saved value is output at output QS. The values at input IS are not saved
- Output VLD = 1 displays the validity of QS. If the retentive memory of the system was recreated when the block is initialized, VLD = 0. In this case, QS is invalid and contains its default value. The status of VLD changes to 1 the first time a value is written (M = 1).

Initialization

The assignment between the SAV block and the value in the retentive memory is performed via the instance name of the block. The unique instance name is automatically generated by the DCC editor when the block is inserted in a chart. The instance name is made up of the call path of the block as follows:

(Chart name)/(Name of subchart 1)/(Name of subchart 2)/../(Name of the block)

The instance name could look like this:

DCC_1/CFC1/CFC2/CFC3/SAV1

Chart name	DCC_1
Name of subchart 1	CFC1
Name of subchart 2	CFC2
Name of subchart 3	CFC3
Name of the block	SAV1

This instance name controls whether output QS is initialized with its default value or outputs the last saved value in the INIT mode. A check is made on the target device whether a retentive value has been saved for this instance name of the block. If not, the memory space is recreated by the system, the default value of the output variable QS transferred to the system for retentive storage, and VLD = 0 set. If a retentive value has been saved for the instance name, this is read, written to output QS, and the status VLD = 1 output.

If no retentive memory is available for the block, output QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
IS	Input variable	16#00	BYTE	
M	Operating mode	0	0/1	
QS	Output variable	16#00	BYTE	
VLD	Output variable valid	0	0/1	
QF	No retentive memory	0	0/1	

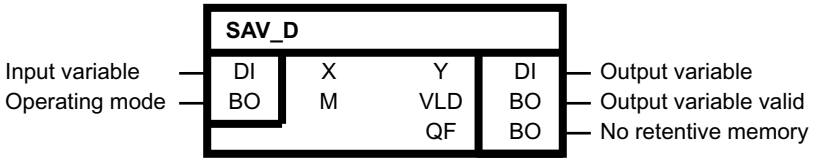
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	No
Special characteristics	A maximum of ten blocks can be used for the retentive storage (SAV, SAV_BY, SAV_I, SAV_D) for each SINAMICS or SINAMICS INTEGRATED drive unit. Retentive memory is available for a maximum of 40 bytes of user data. As of STARTER/SCOUT V4.2, the number of blocks is only checked in the consistency check (menu "Project -> Check consistency" or "Check consistency" in the shortcut menu of the drive unit).

8.1.6.20 SAV_D

Value buffering (DOUBLE INTEGER type)

Symbol



Brief description

SAV_D (Save) enables retentive storage of a DOUBLE INTEGER-type input variable.

Method of operation

The block is a retentive read/write memory for a DOUBLE INTEGER value.

The saved value of a SAV block is not retained when:

- The retentive memory on the target device has been cleared through a user action.
- The chart on which the block was configured, has been deleted and the change transferred to the target device.

- The block has been deleted and the change transferred to the target system.
- The instance name of a block has been changed and transferred to the target system.

The value is retained when:

- The instance name does not change during a download.
- The target device ramps up without configuration data on the memory card. The memory of the missing SAV blocks is only released after a download. In this way, the data is also retained when the firmware is updated.
- Another SAV block has been added or removed.
- A download of the configuration is performed after an update of the DCBLIB.
- Another DO has been added or removed and downloaded to the target device.
- Another chart has been added or removed and downloaded to the target device.
- The target device ramps up with the same configuration as before the power failure.

The block is only active when a 0 on output QF indicates that retentive memory space is available on the target device for storing the input values.

The block mode is set at input M:

Write mode (M =1)

- Input variable X is written cyclically to output Y.
- In addition, input variable X is transferred to the system for retentive storage. In so doing, an already saved value is overwritten.

Read mode (M = 0)

- The current saved value is output at output Y. The values at input X are not saved.
- Output VLD = 1 indicates the validity of Y. If the retentive memory of the system was recreated when the block is initialized, VLD = 0. In this case, Y is invalid and contains its default value. The status of VLD changes to 1 the first time a value is written (M = 1).

Initialization

The assignment between the SAV block and the value in the retentive memory is performed via the instance name of the block. The unique instance name is automatically generated by the DCC editor when the block is inserted in a chart. The instance name is made up of the call path of the block as follows:

(Chart name)/(Name of subchart 1)/(Name of subchart 2)/../(Name of the block)

The instance name could look like this:

DCC_1/CFC1/CFC2/CFC3/SAV1

Chart name	DCC_1
Name of subchart 1	CFC1
Name of subchart 2	CFC2

Name of subchart 3	CFC3
Name of the block	SAV1

This instance name controls whether output Y is initialized with its default value or outputs the last saved value in the INIT mode. A check is made on the target device whether a retentive value has been saved for this instance name of the block. If this is not the case, the system recreates the memory space, the default value of output variable Y is transferred to the system for retentive storage and VLD is set to 0. If a retentive value has been saved for the instance name, it is read and written to output Y and status VLD = 1 is output.

If no retentive memory is available for the block, output QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	DINT	
M	Operating mode	0	0/1	
Y	Output variable	0	DINT	
VLD	Output variable valid	0	0/1	
QF	No retentive memory	0	0/1	

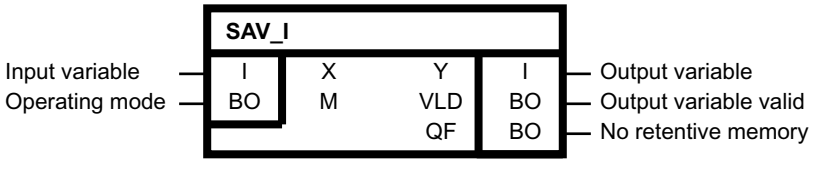
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	No
Special characteristics	A maximum of ten blocks can be used for the retentive storage (SAV, SAV_BY, SAV_I, SAV_D) for each SINAMICS or SINAMICS INTEGRATED drive unit. Retentive memory is available for a maximum of 40 bytes of user data. As of STARTER/SCOUT V4.2, the number of blocks is only checked in the consistency check (menu "Project -> Check consistency" or "Check consistency" in the shortcut menu of the drive unit).

8.1.6.21 SAV_I

Value buffering (INTEGER type)

Symbol



Brief description

SAV_I (Save) enables retentive storage of an INTEGER-type input variable.

Method of operation

The block is a retentive read/write memory for an INTEGER value.

The saved value of a SAV block is not retained when:

- The retentive memory on the target device has been cleared through a user action.
- The chart on which the block was configured, has been deleted and the change transferred to the target device.
- The block has been deleted and the change transferred to the target system.
- The instance name of a block has been changed and transferred to the target system.

The value is retained when:

- The instance name does not change during a download.
- The target device ramps up without configuration data on the memory card. The memory of the missing SAV blocks is only released after a download. In this way, the data is also retained when the firmware is updated.
- Another SAV block has been added or removed.
- A download of the configuration is performed after an update of the DCBLIB.
- Another DO has been added or removed and downloaded to the target device.
- Another chart has been added or removed and downloaded to the target device.
- The target device ramps up with the same configuration as before the power failure.

The block is only active when a 0 on output QF indicates that retentive memory space is available on the target device for storing the input values.

The block mode is set at input M:

Write mode (M =1)

- Input variable X is written cyclically to output Y.
- In addition, input variable X is transferred to the system for retentive storage. In so doing, an already saved value is overwritten.

Read mode (M = 0)

- The current saved value is output at output Y. The values at input X are not saved.
- Output VLD = 1 indicates the validity of Y. If the retentive memory of the system was recreated when the block is initialized, VLD = 0. In this case, Y is invalid and contains its default value. The status of VLD changes to 1 the first time a value is written (M = 1).

Initialization

The assignment between the SAV block and the value in the retentive memory is performed via the instance name of the block. The unique instance name is automatically generated by the DCC editor when the block is inserted in a chart. The instance name is made up of the call path of the block as follows:

(Chart name)/(Name of subchart 1)/(Name of subchart 2)/../(Name of the block)

The instance name could look like this:

DCC_1/CFC1/CFC2/CFC3/SAV1

Chart name	DCC_1
Name of subchart 1	CFC1
Name of subchart 2	CFC2
Name of subchart 3	CFC3
Name of the block	SAV1

This instance name controls whether output Y is initialized with its default value or outputs the last saved value in the INIT mode. A check is made on the target device whether a retentive value has been saved for this instance name of the block. If this is not the case, the system recreates the memory space, the default value of output variable Y is transferred to the system for retentive storage and VLD is set to 0. If a retentive value has been saved for the instance name, it is read and written to output Y and status VLD = 1 is output.

If no retentive memory is available for the block, output QF = 1 is set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0	INT	
M	Operating mode	0	0/1	
Y	Output variable	0	INT	
VLD	Output variable valid	0	0/1	
QF	No retentive memory	0	0/1	

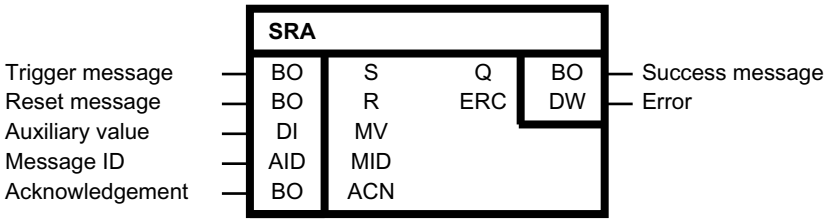
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted online	No
Special characteristics	A maximum of ten blocks can be used for the retentive storage (SAV, SAV_BY, SAV_I, SAV_D) for each SINAMICS or SINAMICS INTEGRATED drive unit. Retentive memory is available for a maximum of 40 bytes of user data. As of STARTER/SCOUT V4.2, the number of blocks is only checked in the consistency check (menu "Project -> Check consistency" or "Check consistency" in the shortcut menu of the drive unit).

8.1.6.22 SRA

Triggers/resets a message

Symbol



Brief description

A message configured in SCOUT can be triggered or reset with the SRA (Set Alarm) block. The message is sent to the HMI and entered in the message buffer in the SIMOTION target device. The message buffer contains all active messages. The SRA block is configured for multiple instantiation. There can be several block instances that trigger the same message number.

Method of operation

- The message configured in SCOUT is entered at the MID message number.
- Input ACN =1 indicates that it is an acknowledgeable message. In this case, the message also only disappears after a reset when it has been acknowledged by the user on the HMI. ACN = 0 is configured for messages that cannot be acknowledged.
- A process value / auxiliary value must be entered at parameter MV if this has been specified in the configuration in SCOUT. A numerical value of the DINT type can be configured.
  The auxiliary value is inserted in the message text with a special syntax during the configuration of the messages: The call of a process value starts with @ and ends with @. The parameters in between indicate the output of the value and its format.
  Only auxiliary values of the DINT type are possible for messages triggered with the SRA block. For detailed information on the syntax of auxiliary values in the message configuration, refer to the SIMOTION SCOUT online help.
- The message is triggered with a rising edge at input S. If the block is called with a new message number with 1 at input S, then the message is also triggered. Up to 40 messages can be entered in the message buffer. If the block SRA is activated with rising edge at input S at full message buffer, the block will be acknowledged with the error message. The message is not entered.
- The message is reset with a rising edge at input R.
- If a rising edge is set at both inputs S and R during a call, R takes priority, i.e. the message is reset.

- Output Q = 1 indicates that the parameter has been successfully set or reset. The outputs are set again with a rising edge at input S or R.
- With Q = 0, output ERC displays an error code that specifies the reason why the message could not be issued:
  The specified values can be displayed as OR logic operation between the constants.

Error code	Meaning
16#00 00	No error. For an incoming message, an entry is made in the message list. The entry is deleted from the message list for an outgoing message.
16#80 01	Message name not permitted.
16#80 02	Message loss through overflow. All 40 entries of the message list are occupied. An entry has not been made in the message list.
16#80 03	Message loss through overflow (signal not sent yet, signal overflow). Send buffer for notification of the clients is still occupied by the last event. An entry has not been made in the message list. An error might also occur if function calls with rising and falling edge follow in quick succession.
16#80 04	Double message, message rejected (call with message came or went two times in succession). An entry has not been made in the message list.
16#80 05	No display device reported. A message is still entered in the list.
16#80 07	No job has been started yet with this message name (first call with S = FALSE). Falling edge (outgoing message) arrived without prior rising edge (incoming message) An entry has not been made in the message list.
16#80 08	A message with this ID is already active in the message buffer. The message occurs when a message with the ID is in the message buffer and the same ID is set again. A new message buffer entry is not generated.
16#80 09	Internal error
16#80 10	Entry was rejected; message acknowledgment memory full.

Block connections

Block connection	Description	Default	Value range	Attributes
S	Trigger message	0	0/1	
R	Reset message	0	0/1	
MV	Auxiliary value	0	DINT	
MID	Message ID	STRUCTALAR-MID#NIL	StructAlarmId	
IFP	Acknowledgement	0	0/1	
Q	Success message	0	0/1	
ERC	Error	0	0- 0x80FF	

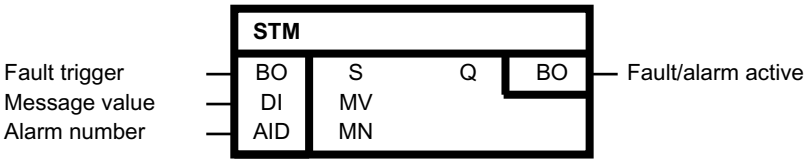
Configuration data

SIMOTION	✓ (as of V4.3)
SINAMICS	-
Can be inserted on-line	Yes
Special characteristics	-

8.1.6.23 STM

Fault/alarm trigger

Symbol



Brief description

A predefined message (fault or alarm) can be triggered on the DO with the STM (Set Message) block. The fault is displayed (e.g. STARTER, AOP) and entered in the fault buffer or alarm buffer of the DO. The following specifications apply to this block type:

- The message number (fault/alarm number) assigned to an instance must be in the range 51050 to 51069 (default value is 51050).
- A message number can be repeated at multiple instances in the DO (message can be issued from different instances). However, for performance reasons, the STM block is not designed for multiple instantiation. The figure below shows the resulting behavior when there is multiple instantiation with the same message number for a fault on the same DO. Without additional RC circuitry, the block instances with the same message number are not coordinated (in any case, this would not be possible if the instances were running in different scan times). For this reason, we recommend assigning a unique message number in the DO for each instance.
- The message text is predefined and cannot be changed (see table below).
- The message type cannot be changed (a fault cannot be redefined to an alarm, or vice versa).

- The default setting for the fault response is OFF2. This can be changed in the SINAMICS basic system parameter:
  - p2100[0..19] "Set fault number for fault response" and
  - p2101[0..19] "Fault response setting"
- The default setting for the acknowledgement mode is IMMEDIATE. This can be changed in the SINAMICS basic system parameter:
  - p2126[0..19] "Setting fault number for acknowledge mode" and
  - p2127[0..19] "Acknowledgement mode setting"

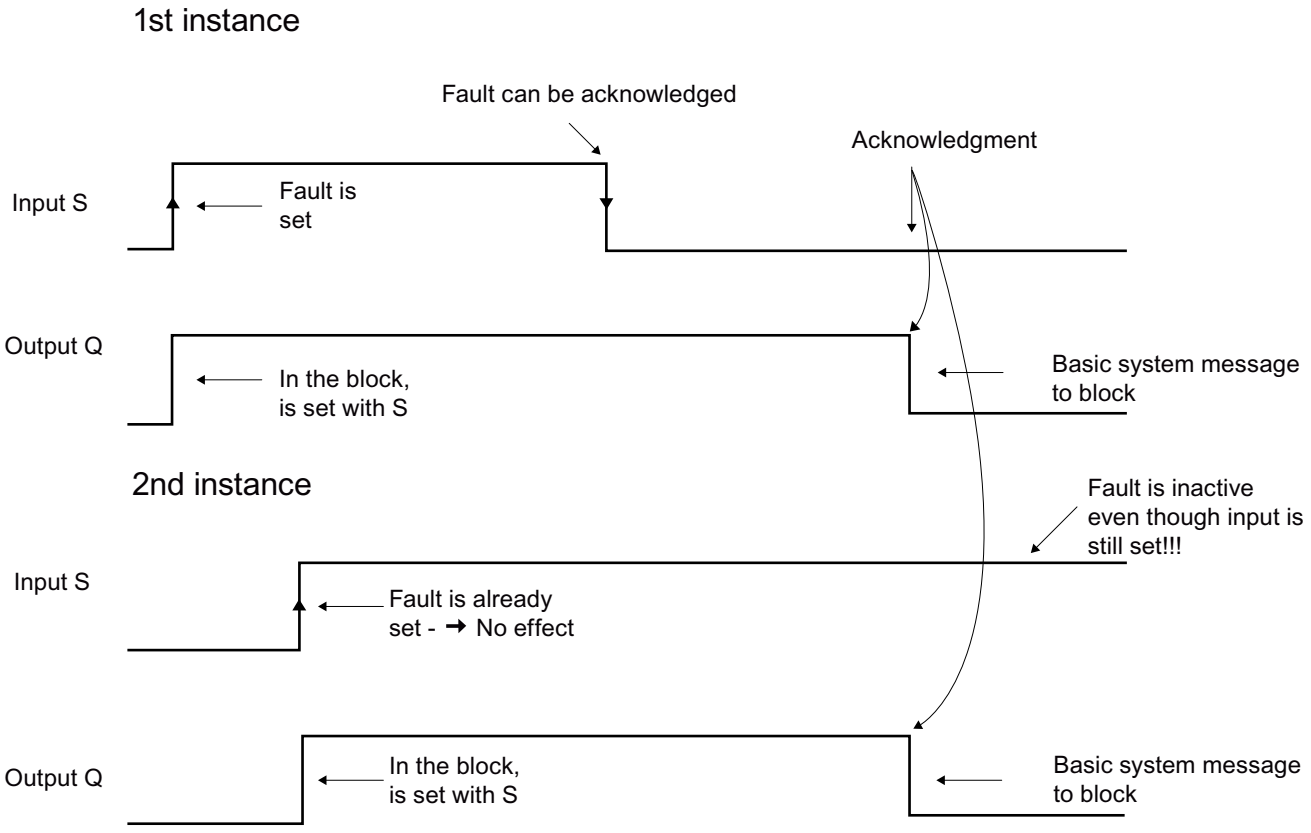
The table below specifies the default settings for the attributes. Possible options for the various settings can be found in the user documentation:

Message type	Alarm number	Reaction	Acknowledgment	Message text
Fault (cannot be changed)	F51050 - F51059	OFF2 (can be changed via p2100/p2101)	IMMEDIATE (can be changed via p2126/ p2127)	DCC: Fault F5105x Additional value: %d(x:= 0 to 9)
Alarm (cannot be changed)	A51060 - A51069			DCC: Alarm A5106x Additional value: %d(x:= 0 to 9)

Method of operation

The number of the fault to be triggered (F51050 - F51059) must be specified at input MN. A positive edge at input S triggers a fault at the DO. This is entered in the fault buffer of the DO and the specified response at the DO is executed. By doing this, output Q is set by the block. Output Q remains set as long as the fault is active. After a negative edge at input S, the fault can be acknowledged according to the acknowledgement attribute of the message (analog system faults: see first instance in figure below).

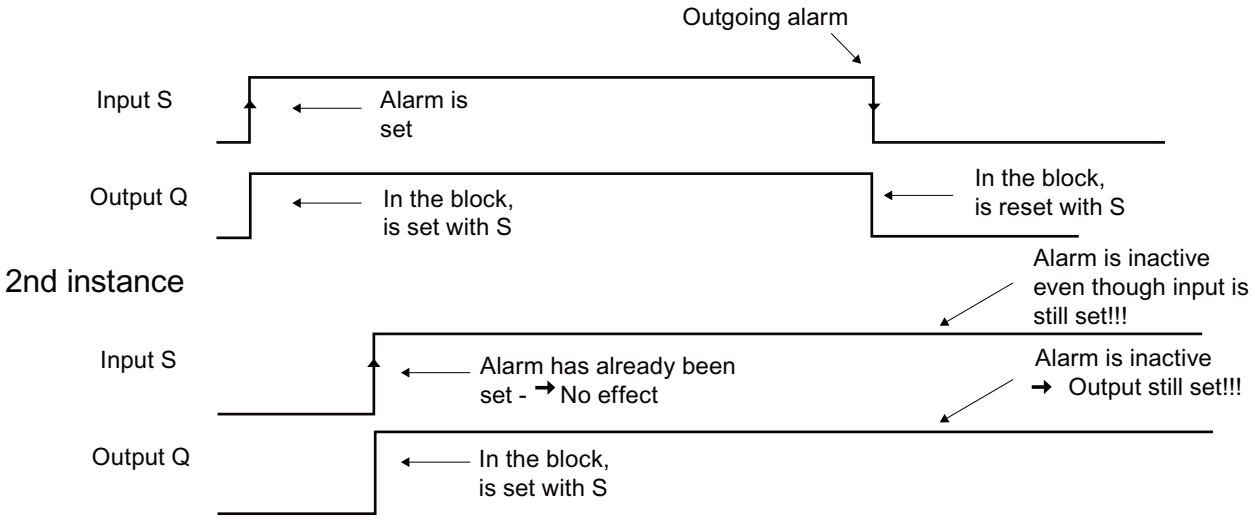
Input MV can be used to add additional information (fault value) for the fault. The value is transferred to input S when the fault is triggered on a positive edge and is entered in the fault buffer of the DO.



Example of two-fold instantiation with the same fault number on one DO (without additional RC circuitry)

Method of operation

The number of the alarm to be triggered (A51060 - A51069) must be specified at input MN. A positive edge at input S triggers the alarm assigned to the block. This is entered in the alarm buffer of the DO. In so doing, output Q is set. The output remains set as long as the alarm is active. Alarms are self-acknowledging and are acknowledged when input S is reset (see figure below). Input MV can be used to provide additional information (alarm value) for the alarm, which is also entered in the alarm buffer.



Example of two-fold instantiation with the same alarm number at one DO (without additional RC circuitry)

Block connections

Block connection	Description	Default	Value range	Attributes
S	Fault trigger	0	0/1	
MV	Message value	0	DINT	
MN	Alarm number	F51050	F51050 - F51059 A51060 - A51069	
Q	Fault/alarm active	0	0/1	

Configuration data

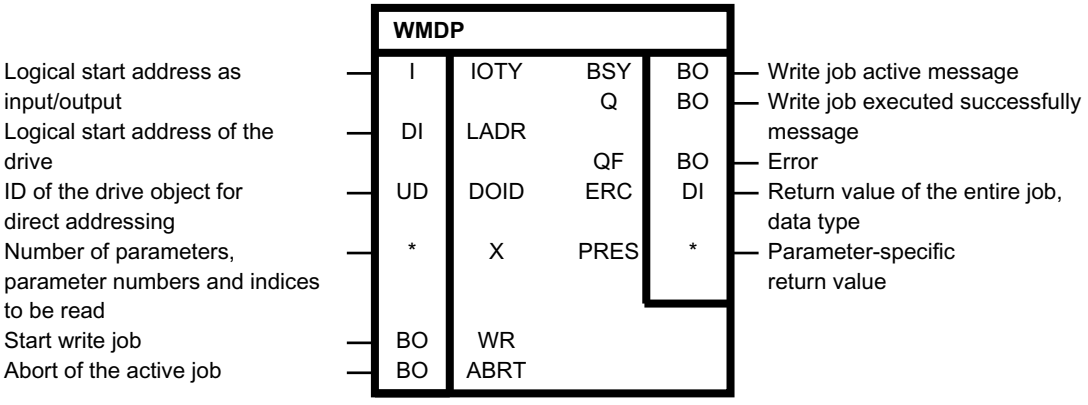
SIMOTION	-
SINAMICS	✓

Can be inserted online	No
Special characteristics	-

8.1.6.24 WMDP

Writes drive parameters from the controller

Symbol



Brief description

The WMDP block enables up to 23 SINAMICS parameters to be written from the DCC SIMOTION program. Only SINAMICS drives are supported. For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter access according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SIMOTION Communication System Manual in Section PROFIdrive and there in the Subsection Acyclic communication (Base Mode Parameter Access) → Error evaluation in table Error codes in Base Mode Parameter Access responses.

The WMDP block is available as of SIMOTION V4.2.

Method of operation

First the block inputs for addressing the drive as well as the selection of the parameters and values to be written are entered. If a parameter is not indexed, IDX = 0 must be set. The asynchronous write job is started with the positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and communication load and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored. The actual reading/writing of the parameters is not performed in the DCC task. The block instance only controls the communication command. The results of the read/write job must be polled at the block outputs in the following task cycles. The evaluation is performed via global variables or user-defined block types. Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1. For error diagnostics, the error code ERC can be evaluated.

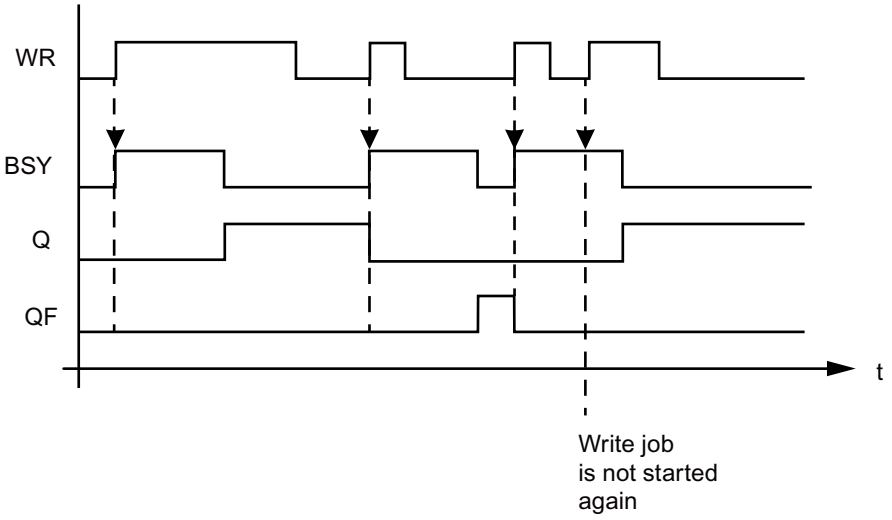
Despite the overall status Q =1, the writing of individual parameters may have been aborted with error. Write jobs for which there is no error in PRES, have been executed.

The error status of the individual write jobs can be evaluated on the parameter-specific return value PRES. An active job is aborted with the positive edge on the ABRT input. The ABRT signal must have the value 1 for at least one cycle.

Brief description

The block enables up to 23 SINAMICS parameters to be written.

Time diagram



Data set 47 is always read out for PROFIBUS (external or integrated) irrespective of whether the function is called with a valid (0>=doid<=254) or invalid 'doid' (doid=255).

Two data sets are available for PROFINET:

- Base Mode Parameter Access - local (data set 0xB02E)
- Base Mode Parameter Access - global (data set 0xB02F)

Data set 0xB02E is used for SIMOTION if either no 'doid' or an invalid 'doid' (doid=255) is specified in the function. Access to the appropriate DO is then performed via the Parameter Access Point (PAP). Either the PAP address can be specified directly or the log. address of the cyclic data (e.g. 256 for a DO axis) can be specified. SIMOTION then determines the corresponding PAP from this address before accessing the correct address. PAP must always be at subslot 1 (HW Config configuration).

Data set 0xB02F is used if a valid 'doid' (0>=doid<=254) is entered. Any valid PAP or address can be specified because the assignment is only performed via the 'doid'.

Description of the block inputs

'IOTY' input/output assignment of the logical start address of the drive.
With 198: INPUT, the logical address of the drive is in the input range.
With 199: OUTPUT, the logical address of the drive is in the output range.
Diagnostics addresses are always of the INPUT type.

'LADR': Specification of the logical start address of the drive. If the optional parameter DOID is also used, any arbitrary address of the station (preferably the diagnostics address of the station) can be specified.
With PROFINET, parameter access is via the Parameter Access Point (PAP) of a drive object.
As an alternative to the logical start address of the drive, specification of the diagnostic address of the associated PAP is recommended.

'DOID': For the direct addressing of a drive object. The DOID can be unspecified or specified as invalid (>254) under the following conditions:

- Access via the DOID is not supported by the DP slave / IO device (P978 is not implemented).
- Data set 0xB02F is not supported (PROFINET only).
- Access is to be via the parameter access point of a DO (PROFINET only).

'X.NUMP': Number of parameters to be written.

'X.PAR.NUM': Specifies the parameter numbers from which the values are to be written.

'X.PAR.IDX': Parameter index; for indexed values, 0 means index 0. For non-indexed values, parameter index 0 must be specified.

'X.PAR.DTYP' specifies the parameter data type (for the coding, see PROFIdrive profile). The data type must match the type of the parameter in the drive. The block performs the data type-specific transfer. If the specified data type does not match the actual data type of the parameter in SINAMICS, an error status is returned.

'X.PAR.X': Data to be written to the drive, DWORD. Conversion blocks are required for different data types. Conversion block R_DW should be used to write a REAL parameter. Conversion block B_DW should be used to write a BYTE parameter.

'WR': Start write job

'ABRT': Abort active job

Description of the block outputs

'Q': Job completed without errors.

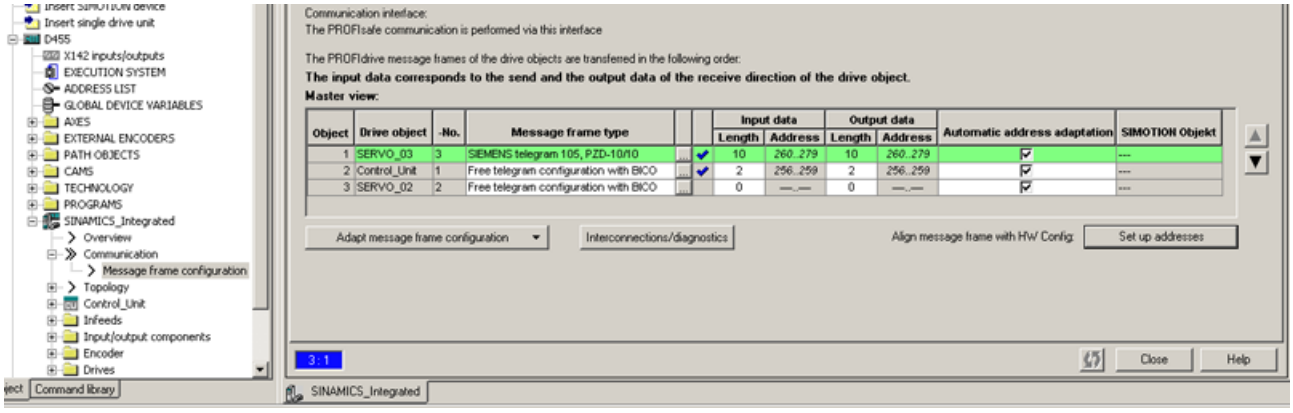
'QF': Job completed with errors.

'ERC': Corresponds to the values of the return value 'functionResult' of the _writeDriveMultiParameter function.

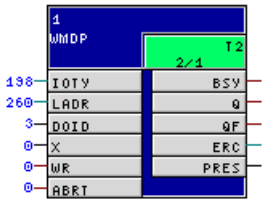
Parameterization example

In order to be able to write certain parameters of a drive object (in the example: SERVO_03), proceed as follows:

First set the correct telegram configuration.

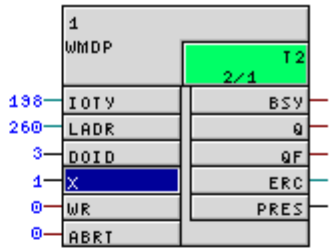


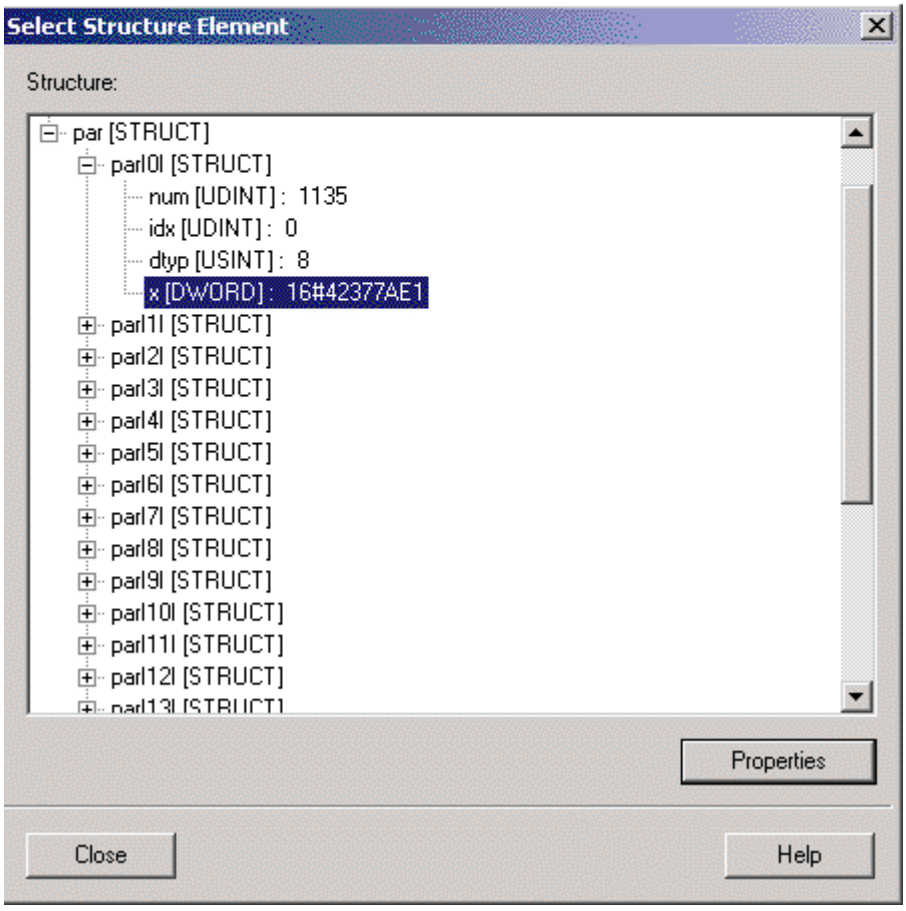
Then set the desired DO address in the WMDP block. To do this, set the block input 'LADR' to the address (260) set in the telegram and the block input 'DOID' to the number (3) set in the telegram.



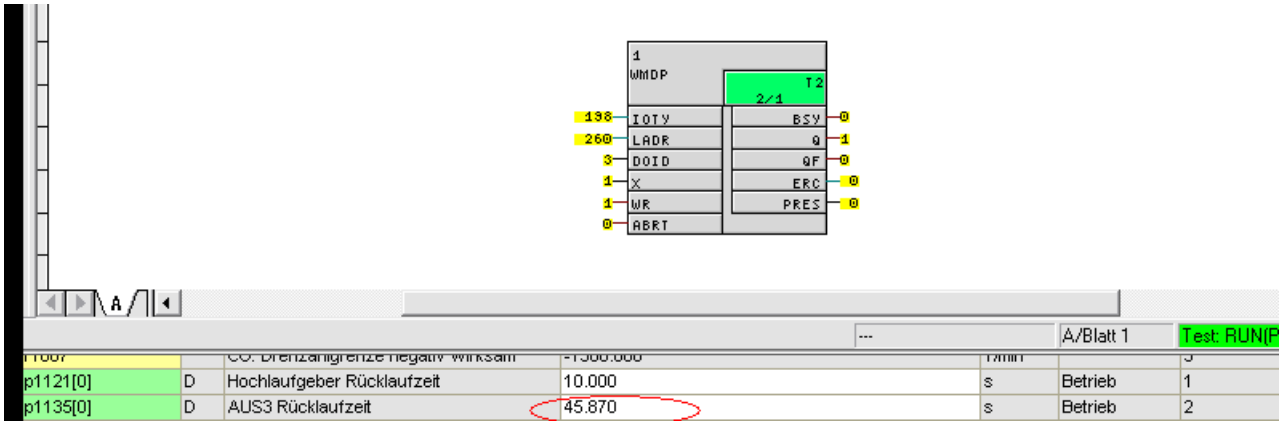
Select the parameters to be written at block input 'X', e.g. P1135(0) OFF3 ramp-down time.

To do this, double-click block input 'X', select the first structure element and enter the parameter number (1135) at 'num' and the index (0) at 'idx'. Enter the data type at 'dtyp'. It must correspond to the data type of the parameter, in our case 8 (floating-point). Enter the value 16#42377AE1 as DWORD at 'x'. The coding of the data types can be found in the SIMOTION List Manual 'System Functions/Variables Devices → System Functions - Devices 1 → _readDriveMultiParameterDescription'.





Then set the block input 'WR' to 1 to start the writing. The result can be viewed in the expert list.



Block connections

Block connection	Description	Default	Value range	Attributes
IOTY	Logical start address as input/ output	0	0: Invalid 198: Input address 199: Output address	
LADR	Logical start address of the drive	-1	DINT	
DOID	ID of the drive object for direct addressing	255	0 .. 254, 255: Invalid	
X	Number of parameters, parameter numbers and indices to be read	1		
X.NUMP	Number of parameters to be written	0	1..23	
X.PAR	Description of a parameter	0		
X.PAR.NUM	Number of the parameter	0	1..65535	
X.PAR.IDX	Index of the parameter	0	1..65535	
X.PAR.DTYP	Data type of the parameter to be written	0	USINT	
X.PAR.X	Value of the parameter	0	DWORD	
WR	Start write job	0	0/1	
ABRT	Abort of the active job	0	0/1	
BSY	Write job active message	0	0/1	
Q	Write job executed successfully message	0	0/1	
QF	Error	0	0/1	
ERC	Return value of the entire job, data type	16#0000	DWORD	
PRES	Parameter-specific return value	0	DWORD	

Configuration data

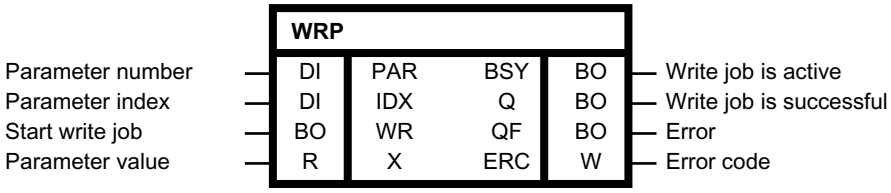
SIMOTION	✓ (as of V4.2)
SINAMICS	-

Can be loaded online	No
Process context	Cyclic, equidistant
Special characteristics	-

8.1.6.25 WRP

Writes drive parameters (REAL type)

Symbol



Brief description

The block enables the asynchronous writing of drive parameters of the REAL type on the local drive object.

NOTICE
Do not use with Safety Integrated The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter that is to be written are indicated at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

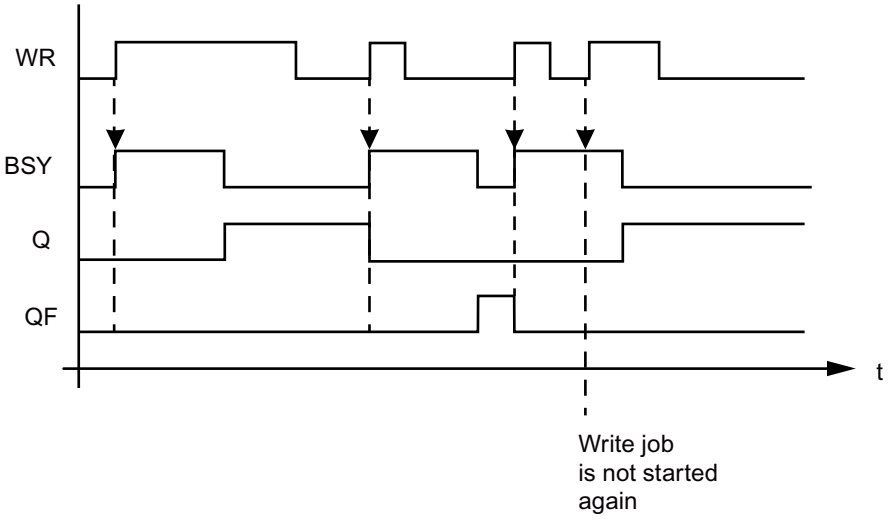
Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in

Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0.0	REAL	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

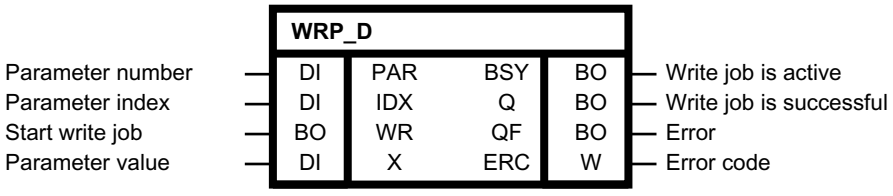
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	No
Special characteristics	-

8.1.6.26 WRP_D

Writes drive parameters (DOUBLE INTEGER type)

Symbol



Brief description

The block enables the asynchronous writing of drive parameters of the DOUBLE INTEGER type on the local drive object.

NOTICE
Do not use with Safety Integrated The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter to be written must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

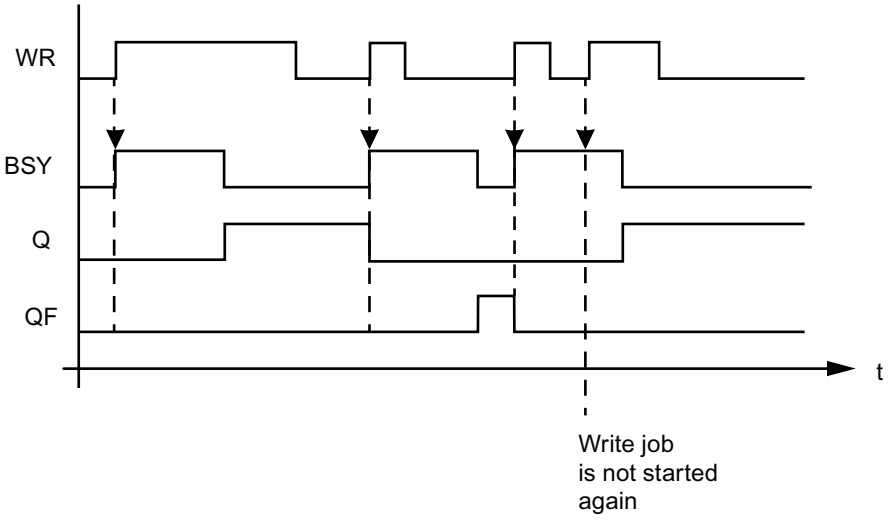
The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0	DINT	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

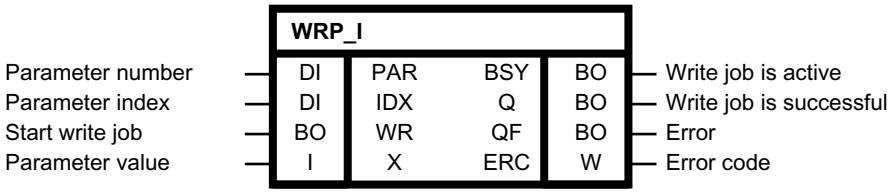
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	No
Special characteristics	-

8.1.6.27 WRP_I

Writes drive parameters (INTEGER type)

Symbol



Brief description

The block allows asynchronous writing of drive parameters of the INTEGER type on the local drive object

NOTICE
Do not use with Safety Integrated
The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter to be written must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

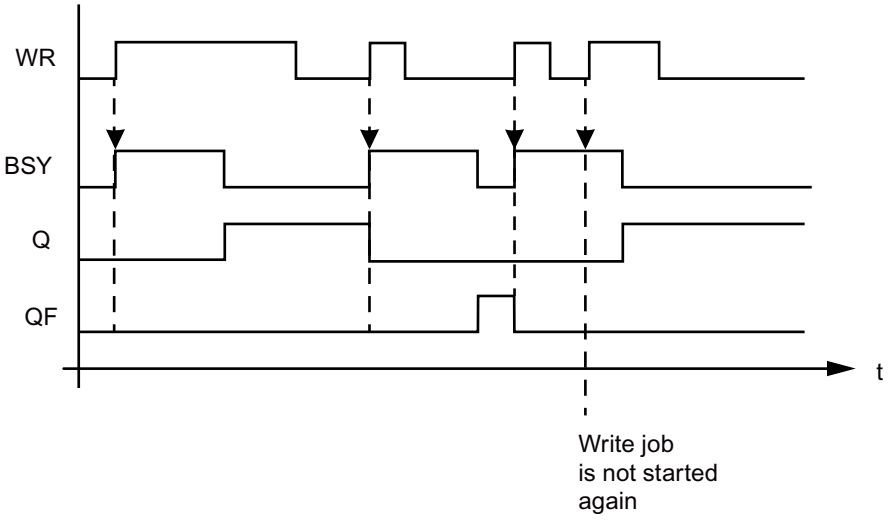
The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0	INT	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

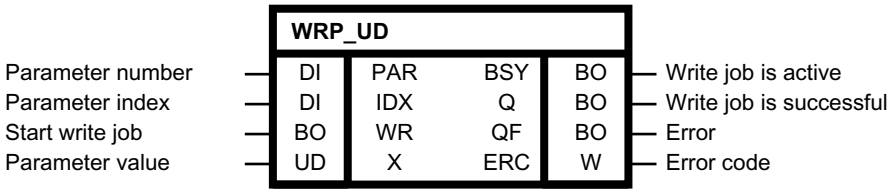
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	No
Special characteristics	-

8.1.6.28 WRP_UD

Writes drive parameters (UNSIGNED DOUBLE INTEGER type)

Symbol



Brief description

WRP_UD (Write Parameter) enables the asynchronous writing of drive parameters of the UNSIGNED DOUBLE INTEGER type on the local drive object.

NOTICE
Do not use with Safety Integrated
The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter to be written must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

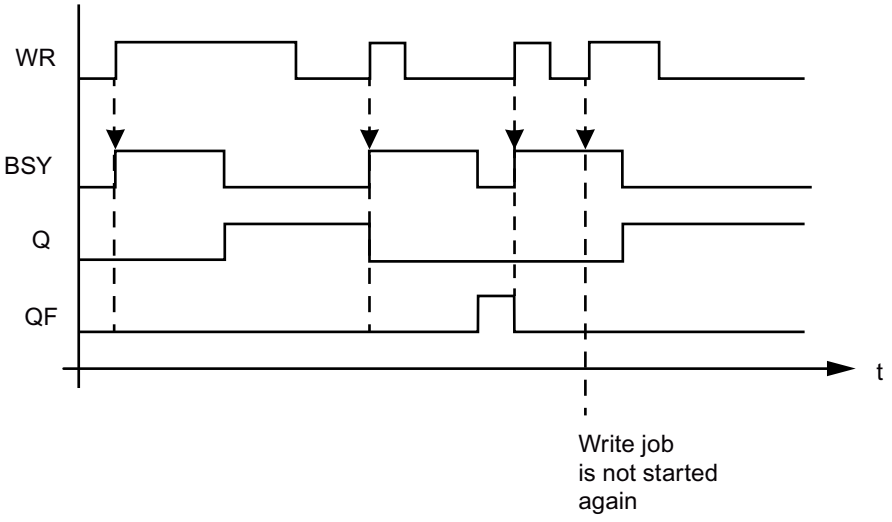
The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0	UDINT	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

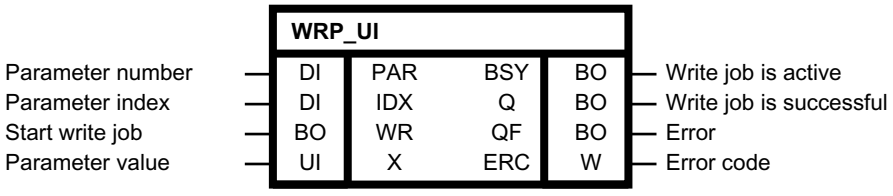
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	No
Special characteristics	-

8.1.6.29 WRP_UI

Writes drive parameters (UNSIGNED INTEGER type)

Symbol



Brief description

WRP_UI (Write Parameter) enables the asynchronous writing of drive parameters of the UNSIGNED INTEGER type on the local drive object.

NOTICE
Do not use with Safety Integrated The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter to be written must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

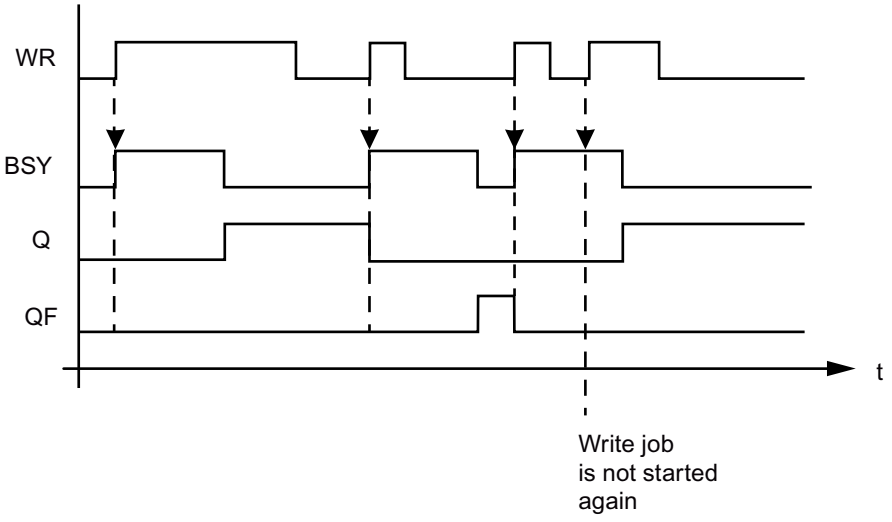
The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0	UINT	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

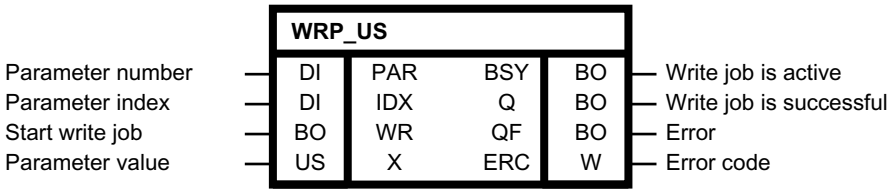
Configuration data

SIMOTION	-
SINAMICS	✓
Can be loaded online	No
Special characteristics	-

8.1.6.30 WRP_US

Writes drive parameters (UNSIGNED SHORT INTEGER type)

Symbol



Brief description

WRP_US (Write Parameter) enables the asynchronous writing of drive parameters of the UNSIGNED SHORT INTEGER type on the local drive object.

NOTICE
Do not use with Safety Integrated The block for programming a drive parameter must not be used for safety reasons to change parameters of the SINAMICS Safety Integrated Functions. DCC is not considered suitable for safety applications in the context of functional safety (Safety Integrated).

Method of operation

The parameter number and the index of the parameter to be written must be specified at inputs PAR and IDX, respectively. If a parameter is not indexed, IDX = 0 must be set. The parameter is always written on the drive object on which the chart with the block is calculated. It is not possible to access parameters on several drive objects.

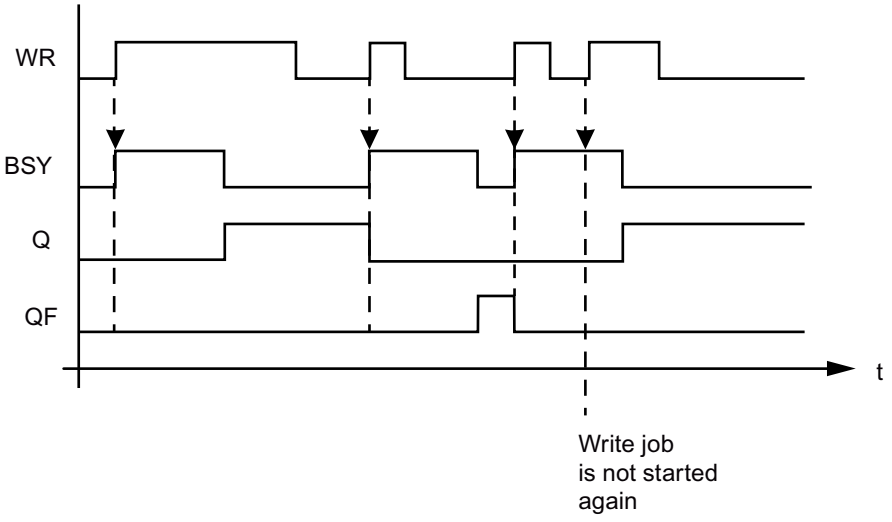
The parameter value is specified via input X. The asynchronous write job can be started on a positive edge at input WR. As long as the job is active, the BSY flag is set. The number of cycles for a parameter access is dependent on the system utilization and can vary from job to job. During an active write job, any additional positive edges at input WR are ignored.

Output Q = 1 indicates that the parameter has been successfully written. If an error occurs during an access, this is signaled with QF = 1.

For error diagnostics, the error code ERC can be evaluated. ERC corresponds to the error code for parameter accesses according to PROVIdrive DPV1. The possible error codes can be found in Appendix A.2 of this document or in the SINAMICS Function Manual FH1 in Section PROFIBUS DP / PROFINET IO Communication and there in the Subsection Communication according to PROFIdrive → Acyclic communication → Configuration of the jobs and responses in Table Error values in DPV1 parameter responses.

ERC is only valid as long as QF = 1.

Time diagram



Quantity framework

Any number of asynchronous jobs of different block instances can be issued in parallel. Each block instance can only process one job.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
PAR	Parameter number	0	$0..2^{16}$	
IDX	Parameter index	0	$0..2^{16}$	
WR	Start write job	0	0/1	
X	Parameter value	0	USINT	
BSY	Write job is active	0	0/1	
Q	Write job is successful	0	0/1	
QF	Error	0	0/1	
ERC	Error code	16#0000	WORD	

Configuration data

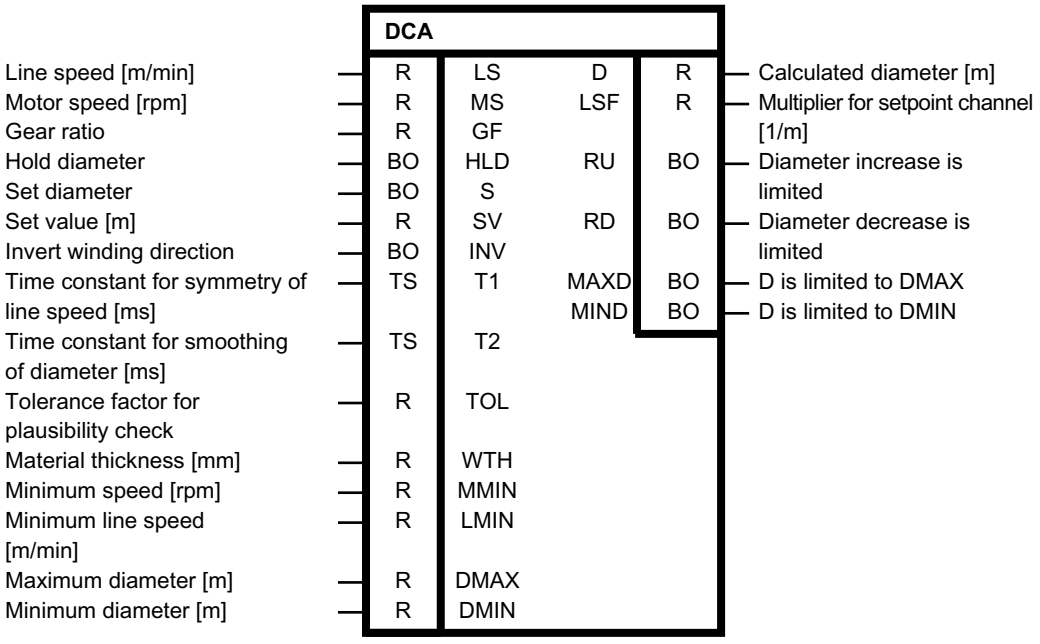
SIMOTION	-
SINAMICS	✓
Can be loaded on-line	No
Special characteristics	-

8.1.7 Technology

8.1.7.1 DCA

Diameter calculator

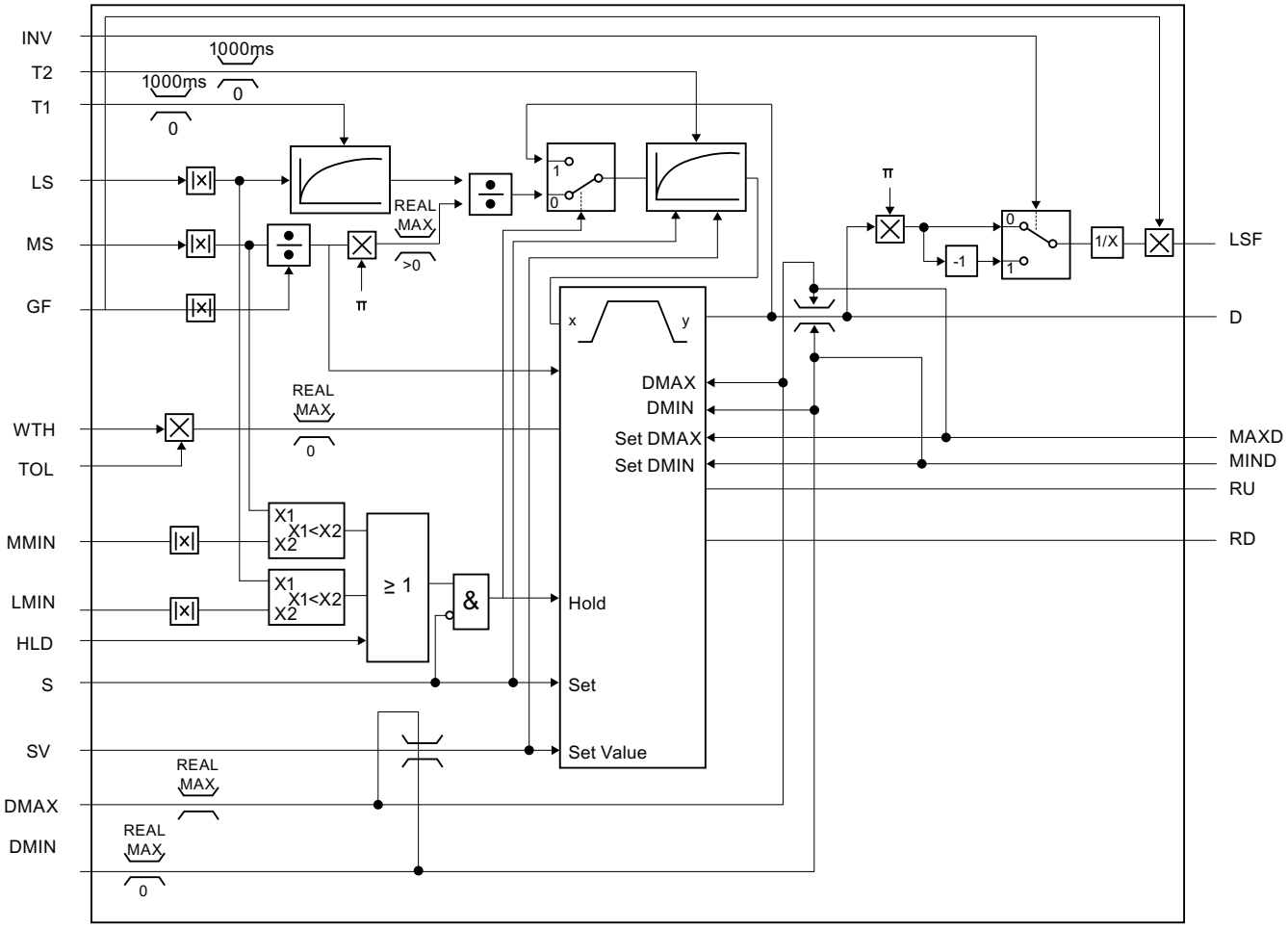
Symbol



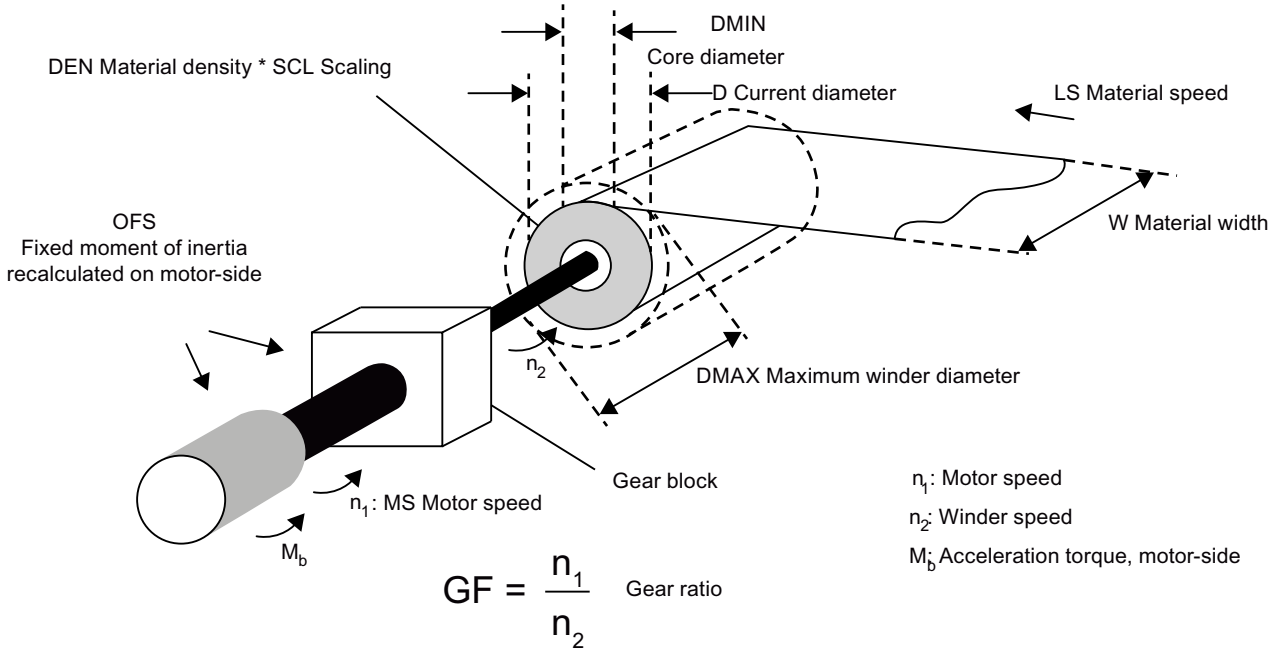
Brief description

Technological block for axial winder applications. It is used to determine the current diameter of an axial winder based on the line speed and the motor speed. The calculated diameter is checked for plausibility.

Block diagram



Method of operation



The block cyclically calculates the diameter of an axial winder on the basis of the current line speed and the motor speed, which have to be supplied via the LS and MS inputs. The current motor speed is indicated at input MS. With deceleration time T1, the path velocity can be decelerated relative to the motor speed.

The current diameter is then calculated using the following formula:

$$\text{Diameter} = \frac{\text{Line speed} \cdot \text{Gear ratio}}{\text{Motor speed} \cdot \pi}$$

The result can then be smoothed again using a smoothing element with time constant T2. The smoothing filters T1 and T2 have PT1 behavior. If time constant T1 or T2 = 0, the input value of the smoothing is written directly to the output. The diameter is only calculated if the path velocity LS or motor speed MS is greater than the threshold value LMIN or MMIN, respectively. Otherwise, the last calculated diameter value is held. In this case, smoothing T2 switches over to the fed-back diameter D. It is also possible to trigger holding of diameter D directly by setting input HLD = 1. Input SV can be used to assign a preset value to the diameter; this diameter is applied when S =1. Smoothing element T2 is also initialized with this value. When S = 0, the diameter calculation and smoothing T2 is enabled again. Setting the diameter has precedence over holding.

After smoothing element T2, the calculated diameter is checked for plausibility and corrected if a violation is identified. This test function is equivalent to that of a single ramp-function generator. The ramp-up time or ramp-down time is calculated dynamically from the material thickness WTH, tolerance factor TOL, and the winding speed. When material thickness WTH = 0, the plausibility check has no effect.

The maximum diameter change ΔD_{max} per scan interval is determined as follows:

$$\Delta D_{max} = TOL \cdot 2 \cdot \frac{MS}{60 \cdot GF} \cdot \frac{WTH}{1000} \cdot T_A$$

with:

ΔD_{max}	Maximum diameter change [m] per scan interval
TOL	Tolerance factor
MS	Motor speed [rpm]
GF	Gear ratio
WTH	Material thickness [mm]
T_A	Block sampling time [s]

The resulting diameter D is limited as follows:

$D_n \leq D_{n-1} + \Delta Dmax_n$; for D_n(unlimited)$\geq D_{n-1}$ (ramp-up limiting)

$D_n \geq D_{n-1} - \Delta Dmax_n$; for D_n(unlimited)$\leq D_{n-1}$ (ramp-down limiting)

Output RU (ramp-up limiting) or RD (ramp-down limiting) is set in order to signal externally that limiting is in effect. If limiting is no longer in effect, the corresponding output is reset to zero. When Hold = 1 or Set = 1, both outputs are reset. When the diameter is set, the ramp-function generator has no effect. The plausibility check is a downstream limiter. If the current diameter is limited to DMAX, output MAXD = 1 is set. If the current diameter is limited to DMIN, this is signaled at output MIND. If limiting is active, the ramp-function generator is corrected with the active limit value in order to avoid anti-windup. In this case, the following applies to the next ramp-function generator cycle:

$D_{n-1} = DMAX_{n-1}$ if diameter is limited to DMAX

$D_{n-1} = DMIN_{n-1}$ if diameter is limited to DMIN

Output LSF cyclically supplies a multiplication factor for the setpoint channel in order to calculate the speed setpoint of the motor from the current path velocity. If the INV input is set to the value 1, the winding direction is inverted.

Block connections

Block connection	Description	Default	Value range	Attributes
LS	Line speed [m/min]	0.0	0..REAL MAX	
MS	Motor speed [rpm]	1.0	0..REAL MAX	
GF	Gear ratio	1.0	0..REAL MAX	
HLD	Hold diameter	0	0/1	
S	Set diameter	0	0/1	
SV	Set value [m]	0.0	0..REAL MAX	
INV	Invert winding direction	0	0/1	
T1	Time constant for symmetry of line speed [ms]	0.0	0..REAL MAX	
T2	Time constant for smoothing of diameter [ms]	0.0	0..REAL MAX	

Block connection	Description	Default	Value range	Attributes
TOL	Tolerance factor for plausibility check	1.5	0..REAL MAX	
WTH	Material thickness [mm]	0.0	0..REAL MAX	
MMIN	Minimum speed [rpm]	1.0	0..REAL MAX	
LMIN	Minimum line speed [m/min]	0.1	0..REAL MAX	
DMAX	Maximum diameter [m]	0.1	0..REAL MAX	
DMIN	Minimum diameter [m]	0.01	0..REAL MAX	
D	Calculated diameter [m]	0.0	0..REAL MAX	
LSF	Multiplier for setpoint channel [1/m]	1.0	0..REAL MAX	
RU	Diameter increase is limited	0	0/1	
RD	Diameter decrease is limited	0	0/1	
MAXD	D is limited to DMAX	0	0/1	
MIND	D is limited to DMIN	0	0/1	

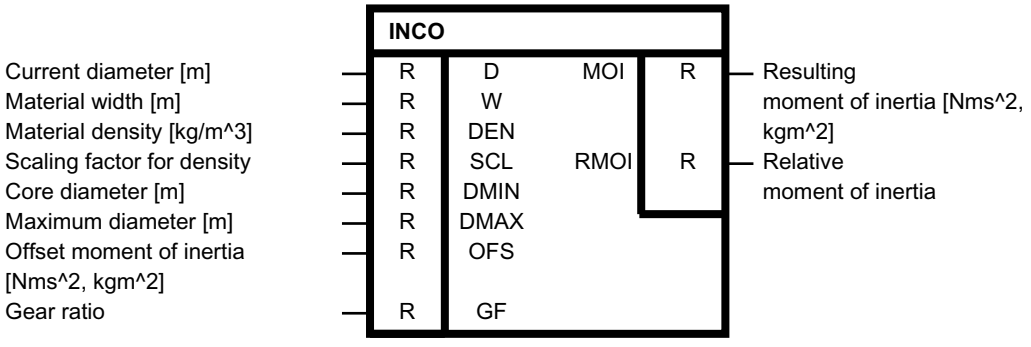
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.7.2 INCO

Axial winder moment of inertia

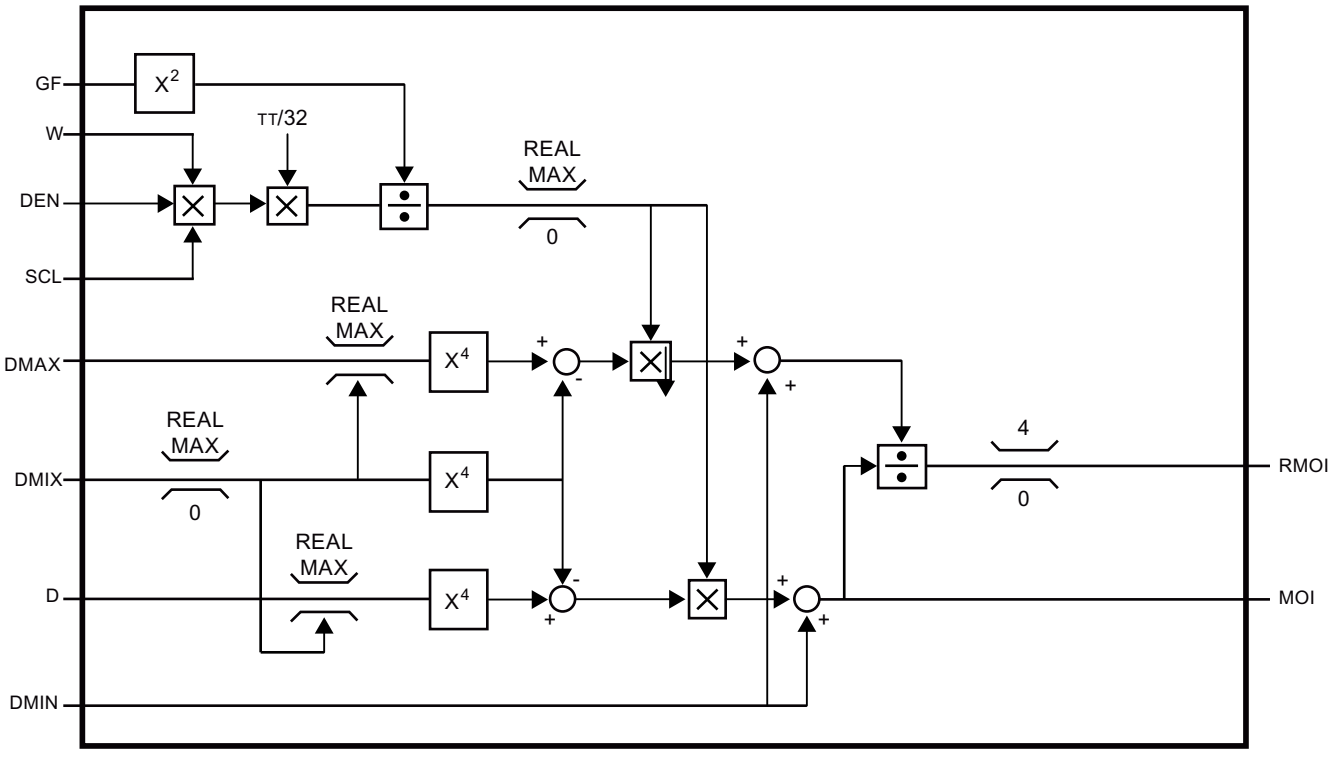
Symbol



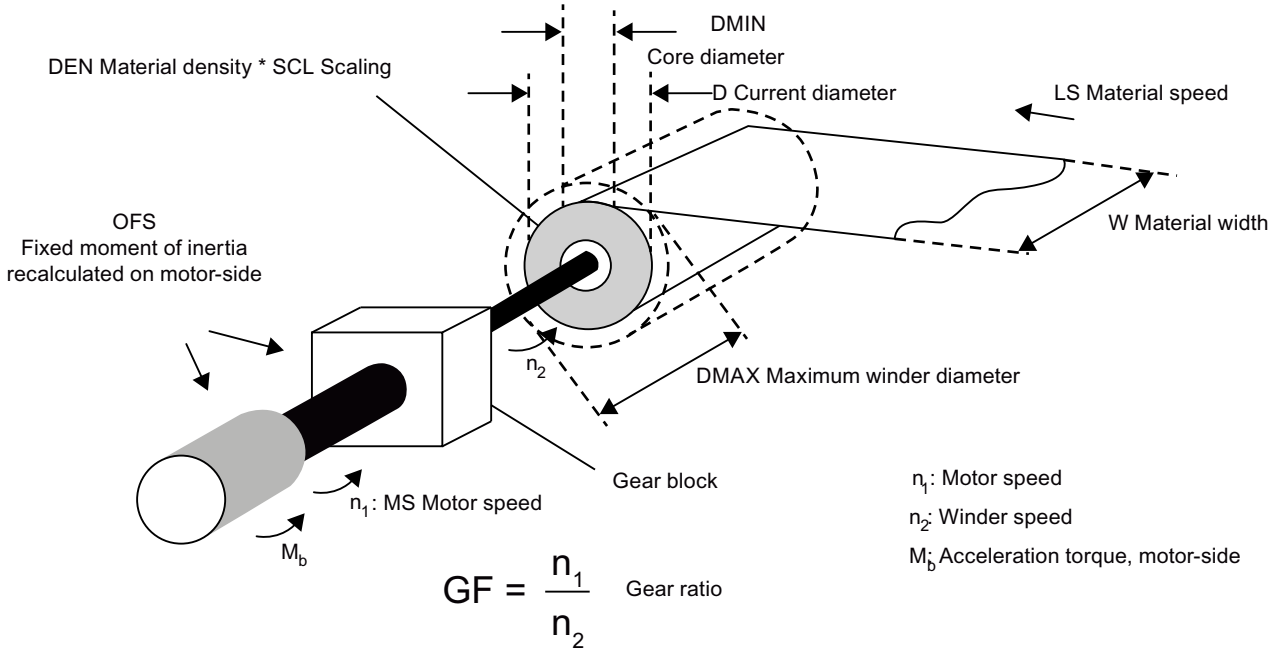
Brief description

Technological block for axial winder applications. It is used for determining the moment of inertia of a winder, which is used to derive a torque feedforward control.

Block diagram



Method of operation



The block calculates the motor-side moment of inertia of an axial winder. Input variable D specifies the current diameter [m] of the winding. The density [kg/m3] of the winder can be specified via DEN, and a correction factor for the density can be specified via SCL. Input variable DMIN [m] is used to specify the diameter of the winding core or the minimum diameter of the wound material. In order to calculate the relative moment of inertia RMOI for an adaptation Kp of the speed controller, the block requires the maximum moment of inertia of the layout. To calculate this, the maximum winding diameter must be specified at input DMAX [m]. The total static moment of inertia (motor, empty winder and, if required, gearbox) relative to the motor side can be specified via input OFS [Nms2, kgm2]. The transmission ratio is specified at input GF. The current moment of inertia of the entire winder layout relative to the motor side is output at output MOI.

Block connections

Block connection	Description	Default	Value range	Attributes
D	Current diameter [m]	0.0	0..REAL MAX	
W	Material width [m]	0.0	0..REAL MAX	
DEN	Material density [kg/m^3]	0.0	0..REAL MAX	
SCL	Scaling factor for density	1.0	0..REAL MAX	
DMIN	Core diameter [m]	0.01	0..REAL MAX	
DMAX	Maximum diameter [m]	0.1	0..REAL MAX	
OFS	Offset moment of inertia [Nms^2, kgm^2]	0.0	0..REAL MAX	
GF	Gear ratio	1.0	0..REAL MAX	

Block connection	Description	Default	Value range	Attrib-utes
MOI	Resulting moment of inertia [Nms^2, kgm^2]	0.0	0..REAL MAX	
RMOI	Relative moment of inertia	0.0	0..REAL MAX	

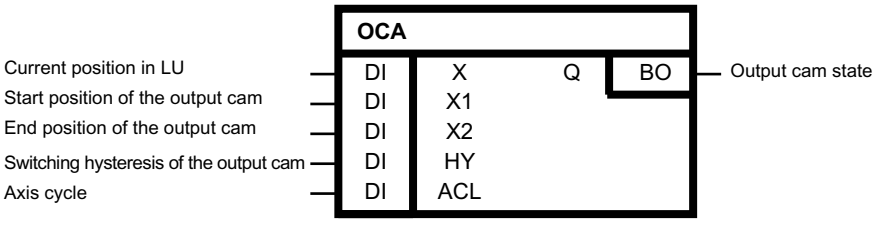
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.7.3 OCA

Software cam controller

Symbol

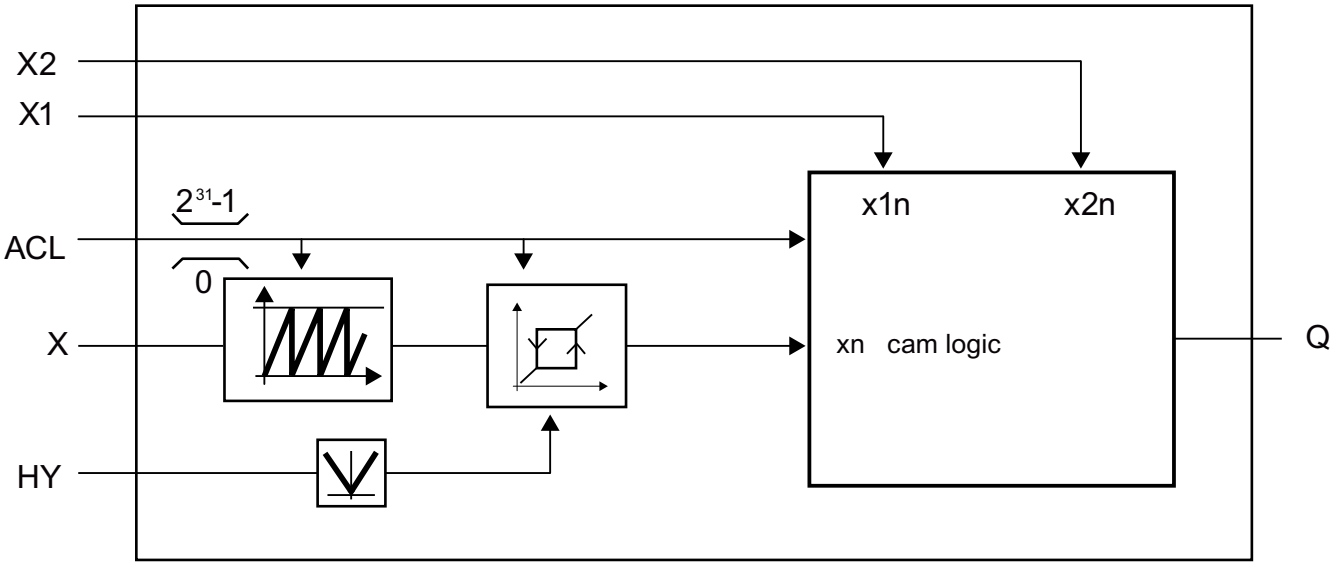


Brief description

Software cam controller with the following features:

- Position-based cam.
- Switch-on/switch-off positions can be changed dynamically.
- Adjustable hysteresis for actual-value-related output cam.

Block diagram



Method of operation

The switch-on position of the position-based cam in the positive direction and the switch-off position in the negative direction is specified via input X1 [LU]. X2 [LU] specifies the switch-off position in the positive direction or the switch-on position in the negative direction. In order to be able to drive the cam controller with modulo axes, the axis cycle can be specified at input ACL. If ACL = 0, there is no internal modulo correction. A hysteresis band for input X can be set via HY. This means that switching operations do not occur when actual value-related output cams are at a standstill.

The hysteresis is used to avoid unwanted switch-on and switch-off operations of the output cam during actual value noise. After a switching operation, switching is only possible again for a direction reversal when the hysteresis range is exited.

The cam logic makes the following evaluation:

Non-modulo axis (ACL = 0)

x1n < x2n	Q = (x1n ≤ xn) AND (x2n > xn)
x1n ≥ x2n	Q = 0

Modulo axis (ACL <> 0):

x1n < x2n	Q = (x1n ≤ xn) AND (x2n > xn)
x1n > x2n	Q = (x1n ≤ xn) OR (x2n > xn)
x1n = x2n	Q = 0

Block connections

Block connection	Description	Default	Value range	Attributes
X	Current position in LU	0	DINT	
X1	Start position of the output cam	0	DINT	
X2	End position of the output cam	0	DINT	
HY	Switching hysteresis of the output cam	0	DINT	
ACL	Axis cycle	0	0...2^{31}-1	
Q	Output cam state	0	0/1	

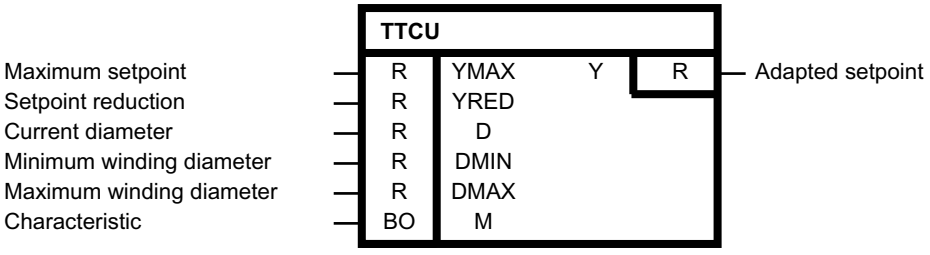
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.7.4 TTCU

Winding harshness characteristic

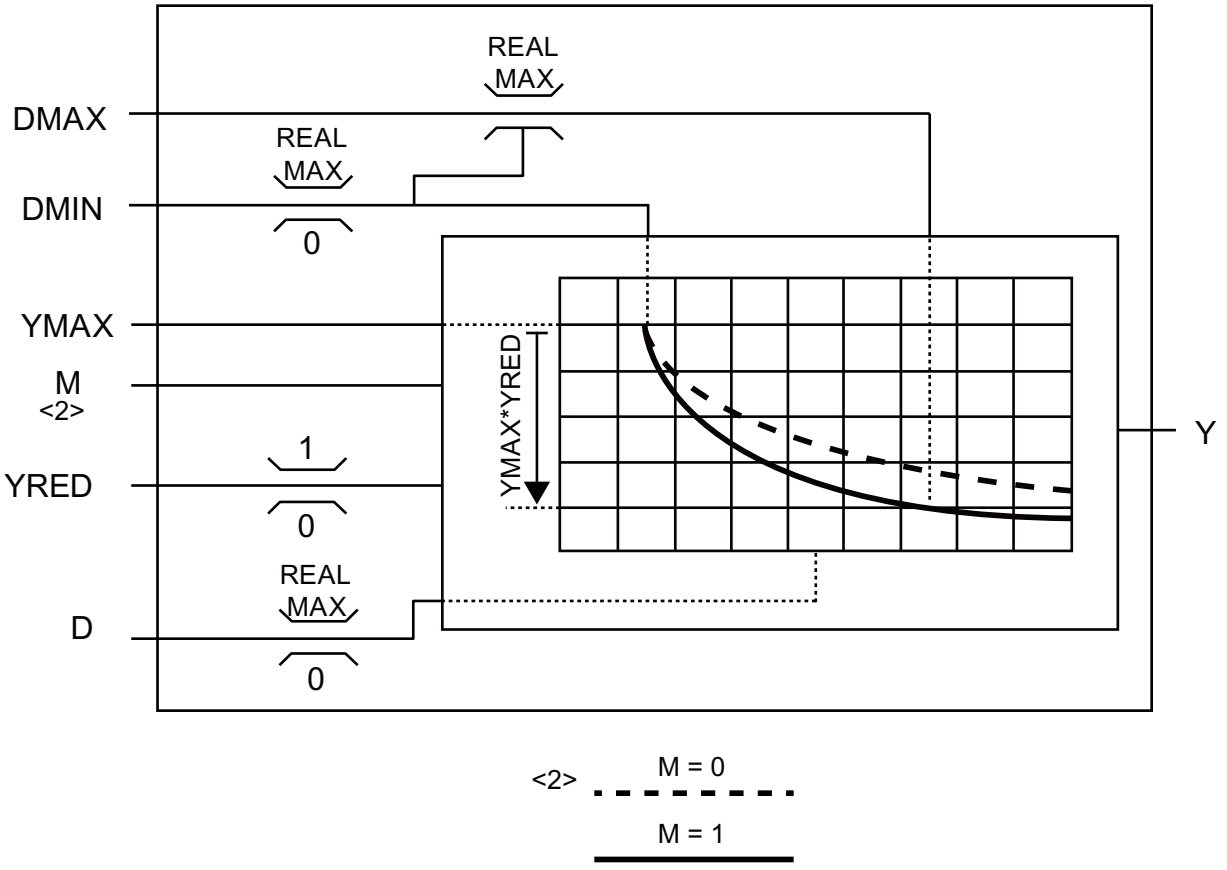
Symbol



Brief description

Adaptation of a setpoint according to the specified characteristic. The block used for winder applications to determine the tension setpoint depending on the current winder diameter.

Block diagram



Method of operation

The reduction of the characteristic starts when D > DMIN is true. Input variable YRED specifies the degree of reduction relative to input variable YMAX. Input M can be used to preselect a characteristic that defines the reduction behavior of the output variable as the input variable increases. If M = 0 has been preselected, the characteristic is reduced asymptotically by the factor YMAX*YRED. In this case, input variable DMAX is not taken into account. If M =1 has been preselected, input variable DMAX can be used to specify at which input variable D = DMAX the characteristic runs through YMAX- YMAX*YRED.

The calculation of the characteristic is specified as follows:

D ≤ DMIN is true

$$Y = YMAX$$

D > DMIN and M = 0 (reaching of the reduction factor for D→∞)

$$Y = YMAX\left(1 - YRED\left(1 - \frac{DMIN}{D}\right)\right)$$

D > DMIN and M = 1 (attainment of reduction factor for D = DMAX)

$$DMAX > DMIN : Y = YMAX \left(1 - YRED \frac{DMAX}{DMAX - DMIN} \left(1 - \frac{DMIN}{D}\right)\right)$$

$$DMAX = DMIN : Y = YMAX \left(1 - YRED\right)$$

Block connections

Block connection	Description	Default	Value range	Attributes
YMAX	Maximum setpoint	0.0	0..REAL MAX	
YRED	Setpoint reduction	0.0	0..1	
D	Current diameter	0.0	0..REAL MAX	
DMIN	Minimum winding diameter	1.0e-2	0..REAL MAX	
DMAX	Maximum winding diameter	0.1	0..REAL MAX	
M	Characteristic	1	0/1	
Y	Adapted setpoint	0.0	0..REAL MAX	

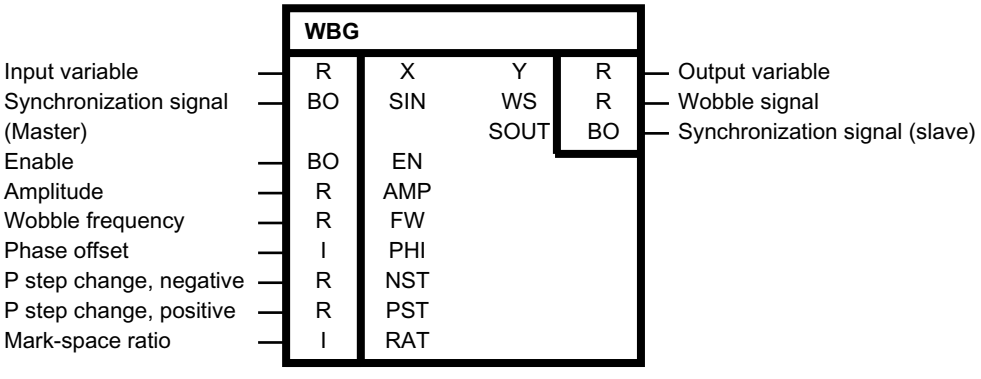
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.7.5 WBG

Wobble generator

Symbol

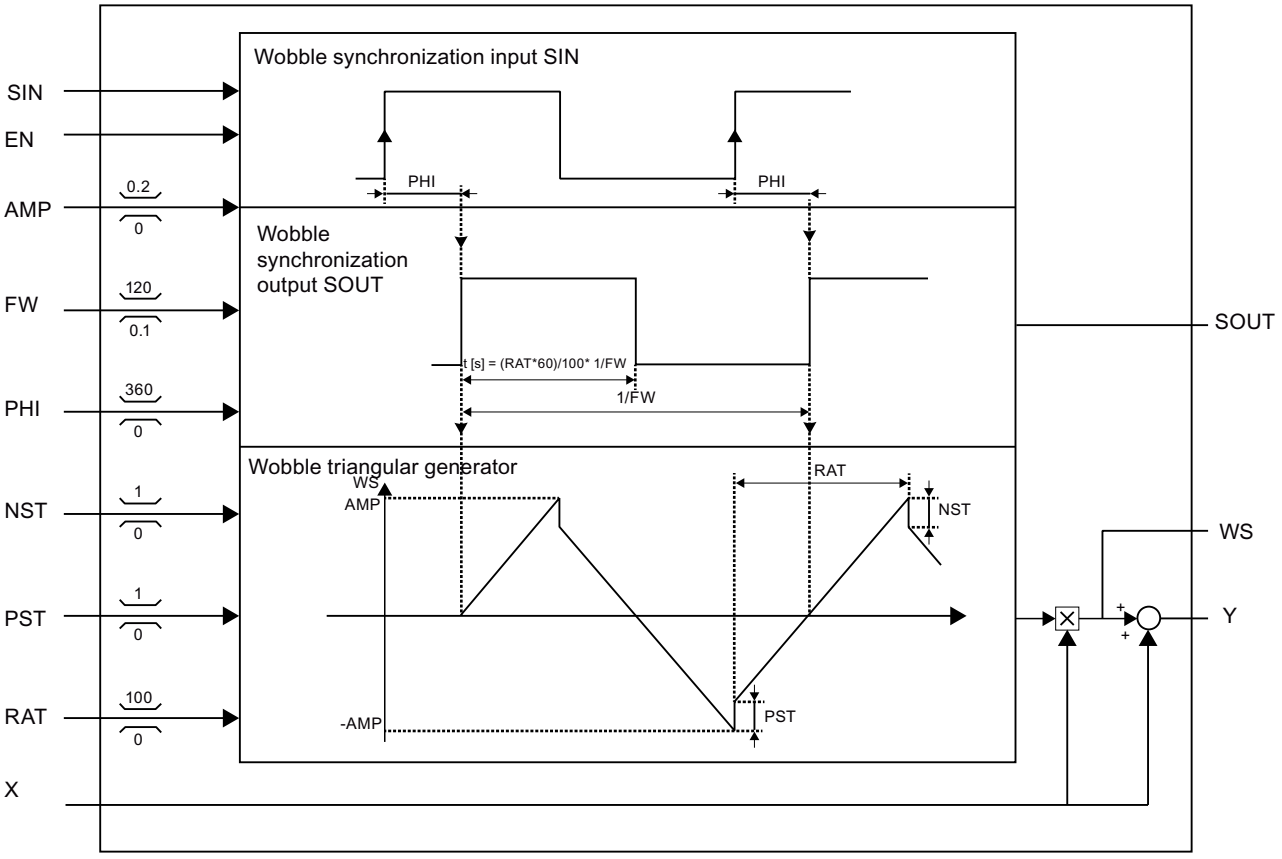


Brief description

Triangular generator with adjustable frequency and amplitude for imprinting of "faults" on traversing drives for winding up textile threads. This generator has the following features:

- Positive and negative P step change, can be adjusted separately.
- Synchronization to a master drive with an adjustable phase shift.
- Enable input.

Block diagram



Method of operation

The wobble generator is enabled with EN = 1. This triggers the output of wobble signal WS and synchronization signal SOUT. The signal generation always starts with a positive zero crossover or with a positive edge of synchronization output SOUT. If EN is reset, wobble generation continues up to the next zero crossover of WS. The generator is then disabled and SOUT = 0 is set. Input PHI (0-360°) enables a phase shift between the positive edge of the synchronization input SIN to be set along with the start of the wobble signal. The signal is then generated for a signal period. For continuous signal generation, SIN must be used periodically as a trigger. If the generation of the preceding signal period is still running at a new start time, this generation is canceled. Special case PHI = 360 enables free-running wobble generation to be activated. The signal generation runs periodically and is decoupled from synchronization input SIN. The wobble signal is injected into input x and output at output Y.

Attributes of the wobble signal

Input	Value range	Description
AMP	0..0.2	Relative amplitude of the wobble signal
FW	0.1..120 rpm	Frequency of the wobble signal
PHI	0..360°	Phase shift of wobble signal relative to a positive edge at synchronization input SIN
NST	0.0..1.0	Relative, negative step change of wobble signal at the end of the positive signal edge
PST	0.0..1.0	Relative, positive step change of wobble signal at the end of the negative signal edge
RAT	0..100%	Ratio of rising signal edge / signal period

Effective amplitude(WS) = ABS(X) * AMP

Effective negative step change = -ABS(X) * AMP * NST

Effective positive step change = ABS(X) * AMP * PST

Ratio of rising edge / falling edge = RAT/(100-RAT)

If the attributes of the wobble signal are changed dynamically, the changed attributes take effect at the start of a new signal period (positive zero crossover).

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
SIN	Synchronization signal (master)	0	0/1	
EN	Enable	0	0/1	
AMP	Amplitude	0.0	0..0.2	
FW	Wobble frequency	60	0.1..120	
PHI	Phase offset	360	0..360	
NST	P step change, negative	0.0	0.0..1.0	
PST	P step change, positive	0.0	0.0..1.0	

Block connection	Description	Default	Value range	Attrib-utes
RAT	Mark-space ratio	50	0..100	
Y	Output variable	0.0	REAL	
WS	Wobble signal	0.0	REAL	
SOUT	Synchronization signal (slave)	0	0/1	

Configuration data

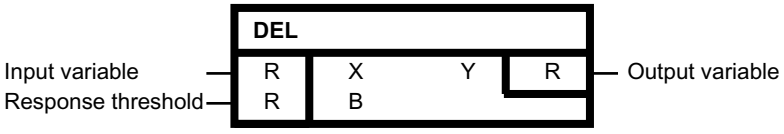
SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.8 Closed-loop control

8.1.8.1 DEL

Dead zone element

Symbol



Brief description

- Adjustable dead band
- Set zero-point symmetric value range to zero

Method of operation

- If the absolute value of X is less than B, then Y = 0
- If X is greater than or equal to B, then Y = X - B
- If X is less than or equal to B, then Y = X + B

The zero-point symmetric dead band can be set with operating value B.

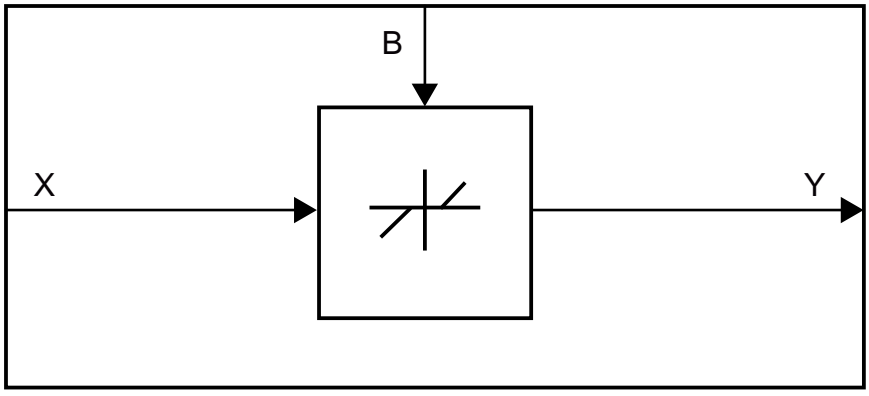
Algorithm:

with boundary condition B >=0

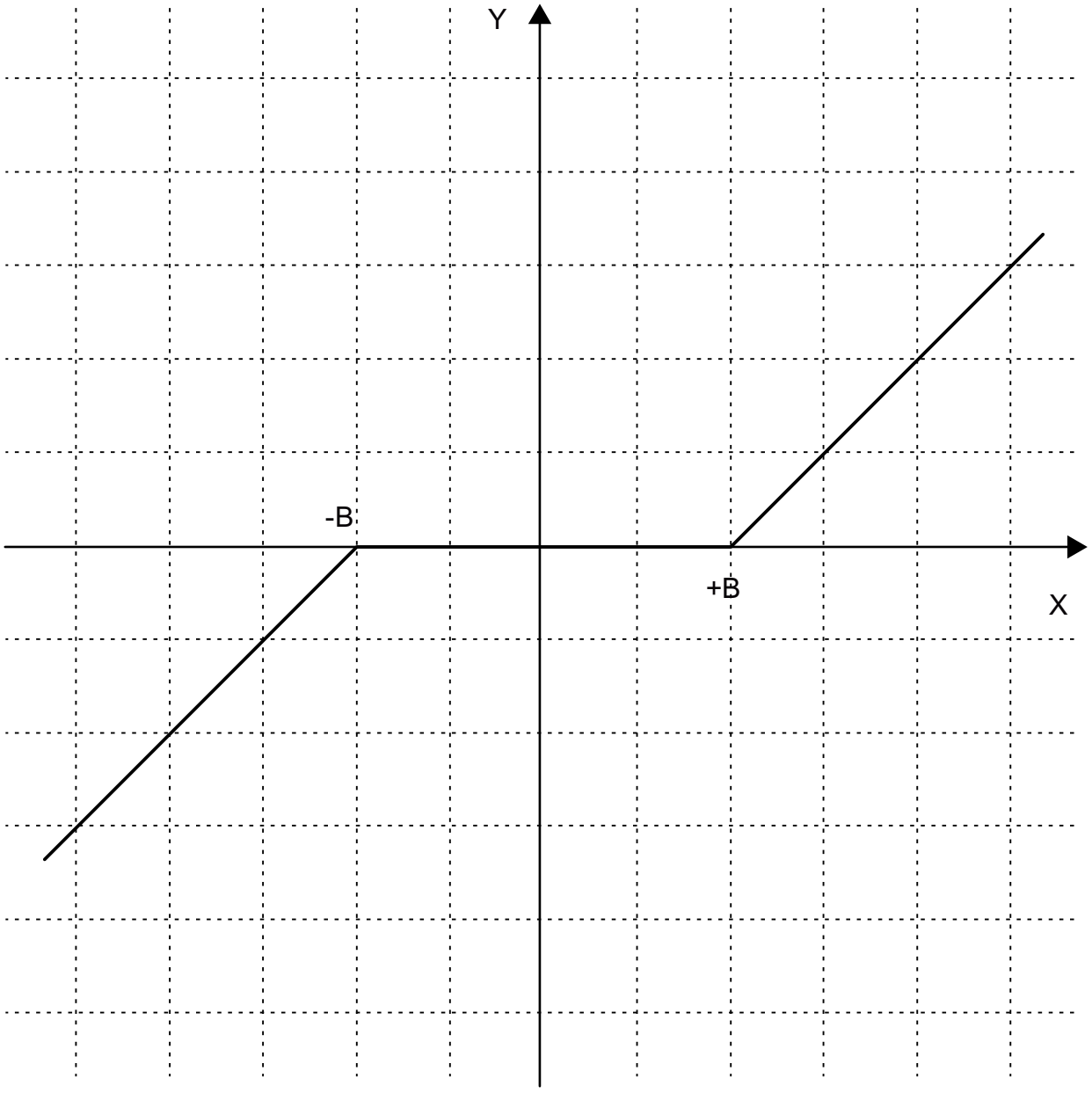
For B < 0, the following applies for all X: Y = X.

$$Y = \begin{cases} X + B \text{ for } X <= -B \\ 0 \text{ for } -B < X < B \\ X - B \text{ for } X >= B \end{cases}$$

Block diagram



XY diagram



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
B	Response threshold	0.0	REAL	
Y	Output variable	0.0	REAL	

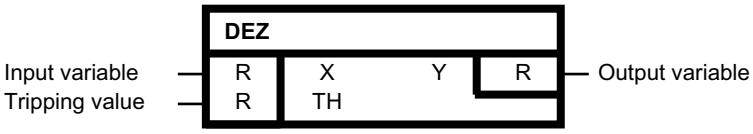
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.8.2 DEZ

Dead zone element

Symbol



Brief description

- Adjustable dead zone
- Set zero-point symmetric value range to zero

Method of operation

If the absolute value of X is less than TH, then Y = 0.

If the absolute value of X is greater than or equal to TH, then Y = X.

The zero-point symmetric dead zone can be set with operating value TH.

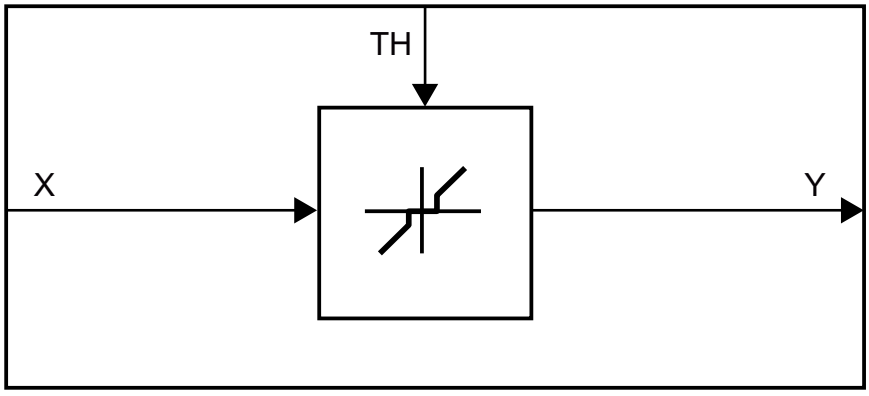
Algorithm:

$$Y = \begin{cases} X \text{ for } X <= -TH \\ 0 \text{ for } -TH < X < TH \\ X \text{ for } X >= TH \end{cases}$$

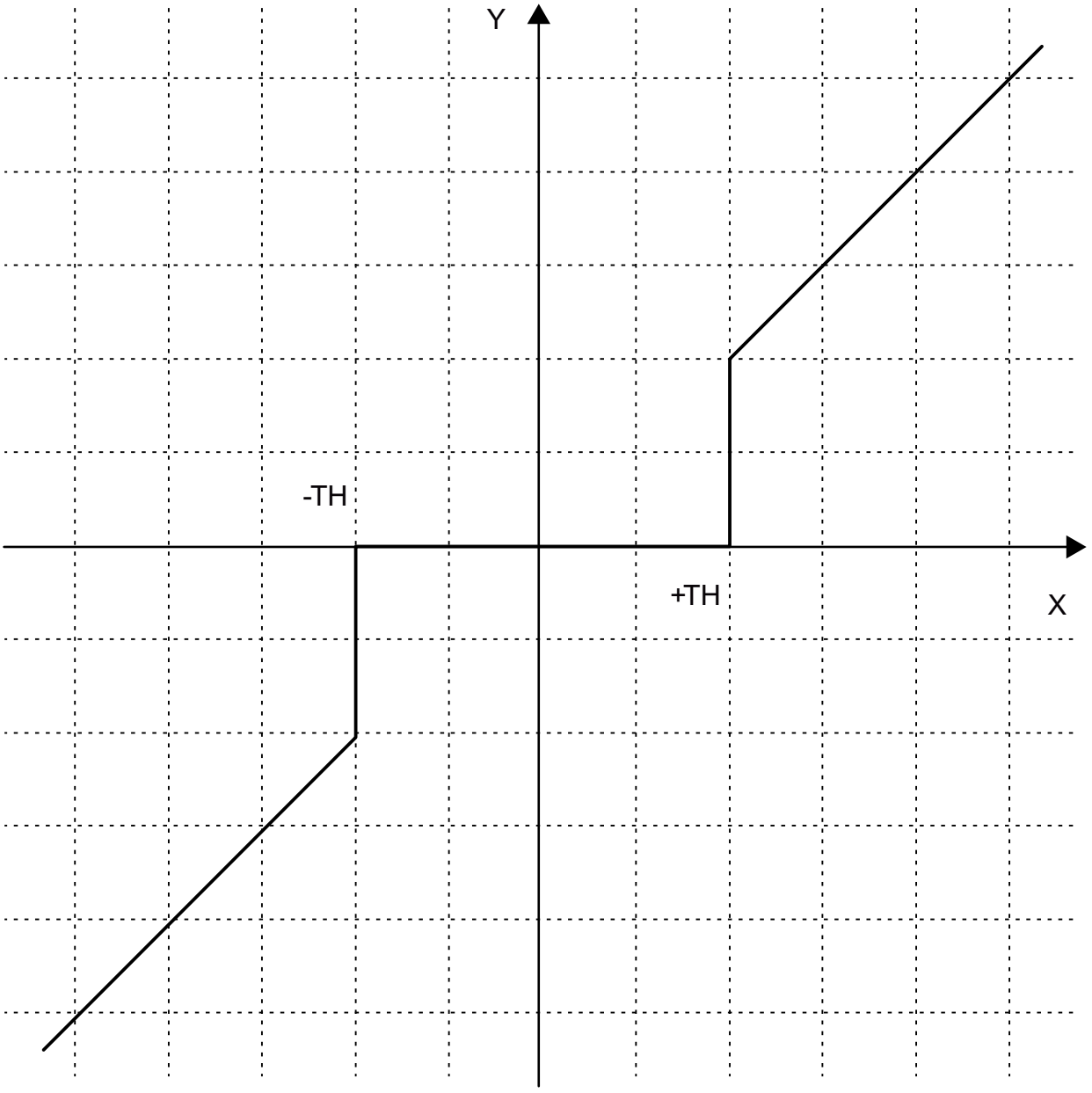
with boundary condition TH >=0

For TH< 0, the following applies for all X: Y = X.

Block diagram



XY diagram



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
TH	Tripping value	0.0	REAL	
Y	Output variable	0.0	REAL	

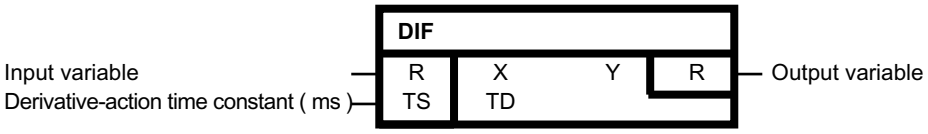
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.8.3 DIF

Derivative action element

Symbol



Brief description

Block with derivative-action response

Method of operation

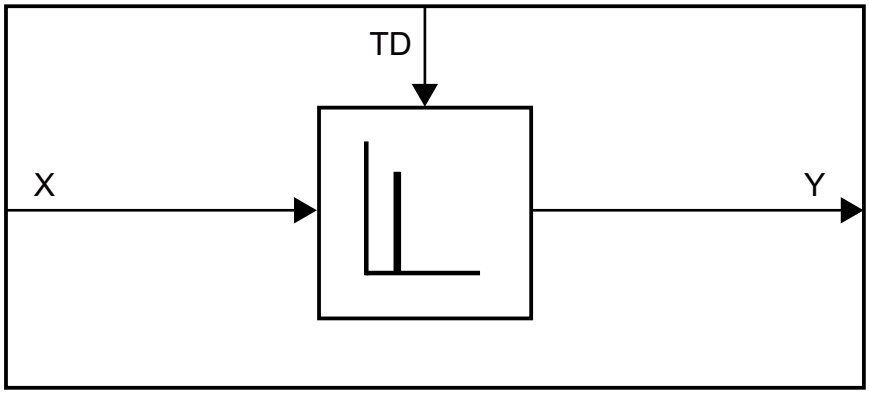
- Output variable Y is proportional to the change velocity of input variable X, multiplied by the derivative-action time constant TD.
- Discrete values are calculated according to the algorithm:

Algorithm:

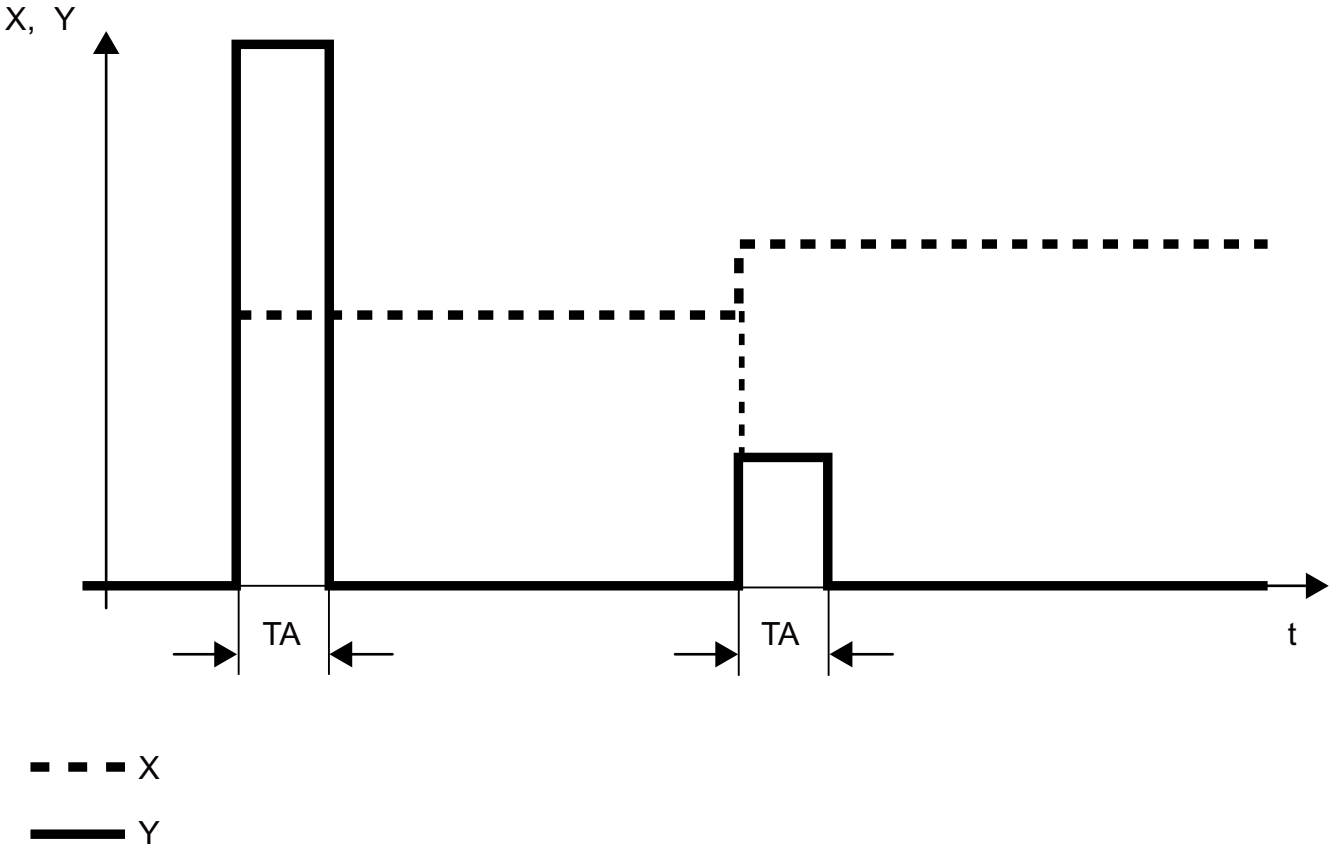
$$Y_n = (X_n - X_{n-1}) \cdot \frac{TD}{TA}$$

Y_n	Value of Y in scan interval n
X_n	Value of X in scan interval n
X_{n-1}	Value of X in scan interval n-1

Block diagram



XY diagram



Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
TD	Derivative-action time constant (ms)	0	SDTIME	
Y	Output variable	0.0	REAL	

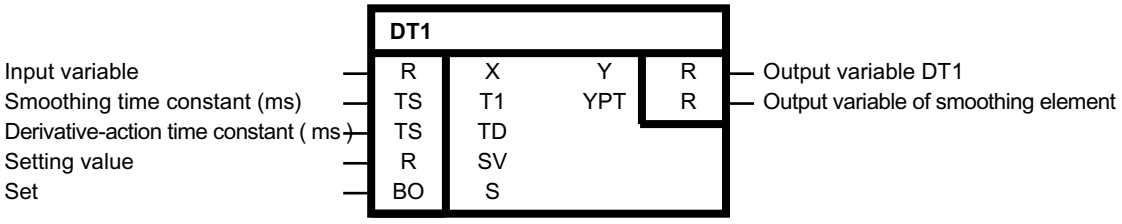
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.4 DT1

Smoothing element

Symbol



Brief description

Block with derivative-action response and smoothing. The DT1 output can be set.

Method of operation

Setting function not active (S = 0)

Input variable X (dynamically delayed by smoothing time constant T1), is given to a derivative-action element and block output YPT.

Output variable Y of the entire DT1 element is proportional to the change velocity of YPT (differential quotient), multiplied by the derivative-action time constant TD.

T1 determines the steepness of the decline of the output variable. It specifies the time at which the transfer function fell to 37% of X-TD/T1 after smoothing and differentiation. If T1/TA is sufficiently large (T1/TA>10), the transfer function corresponds to the characteristic curve of

$Y(t) = X \cdot (TD/T1) \cdot e^{-t/T1}$

with $t = n \cdot TA$

Algorithm:

$$Y_n = \frac{TD}{T1} \cdot (X_n - YPT_{n-1})$$

$$YPT_n = YPT_{n-1} + \frac{TA}{T1} \cdot (X_n - YPT_{n-1})$$

YPT_n	Value of YPT in scan interval n
Y_n	Value of Y in scan interval n
X_n	Value of X in scan interval n
YPT_{n-1}	Value of YPT in scan interval n-1

The larger T1/TA is, the smaller is the amplitude change on Y and YPT from one sampling time to the next. TA is the sampling time in which the block is configured. The larger that TD/TA is, the larger the amplitude change on Y from one sampling time to the next. TD and T1 are limited internally: TD >= 0, T1 >= TA.

Setting function active (S = 1)

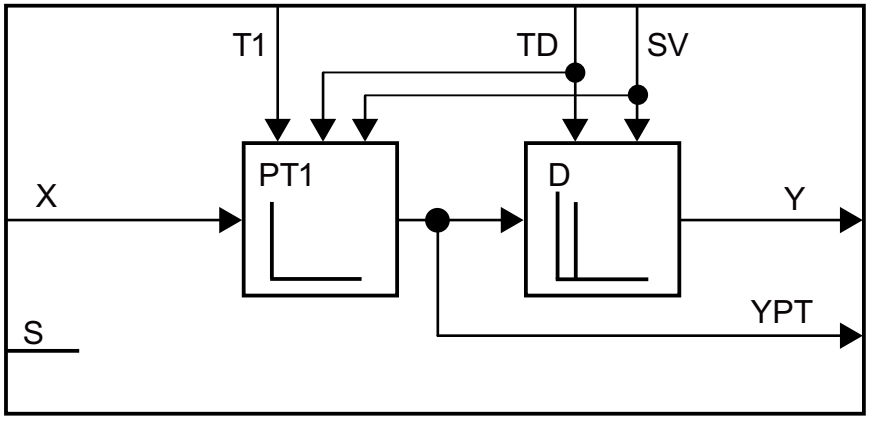
With active setting function, the setting value SV is applied at the dt1 output Y (Y=SV), the following results for the output of the smoothing element:

$$YPT_n = X_n - \frac{T1}{TD} \cdot SV_n$$

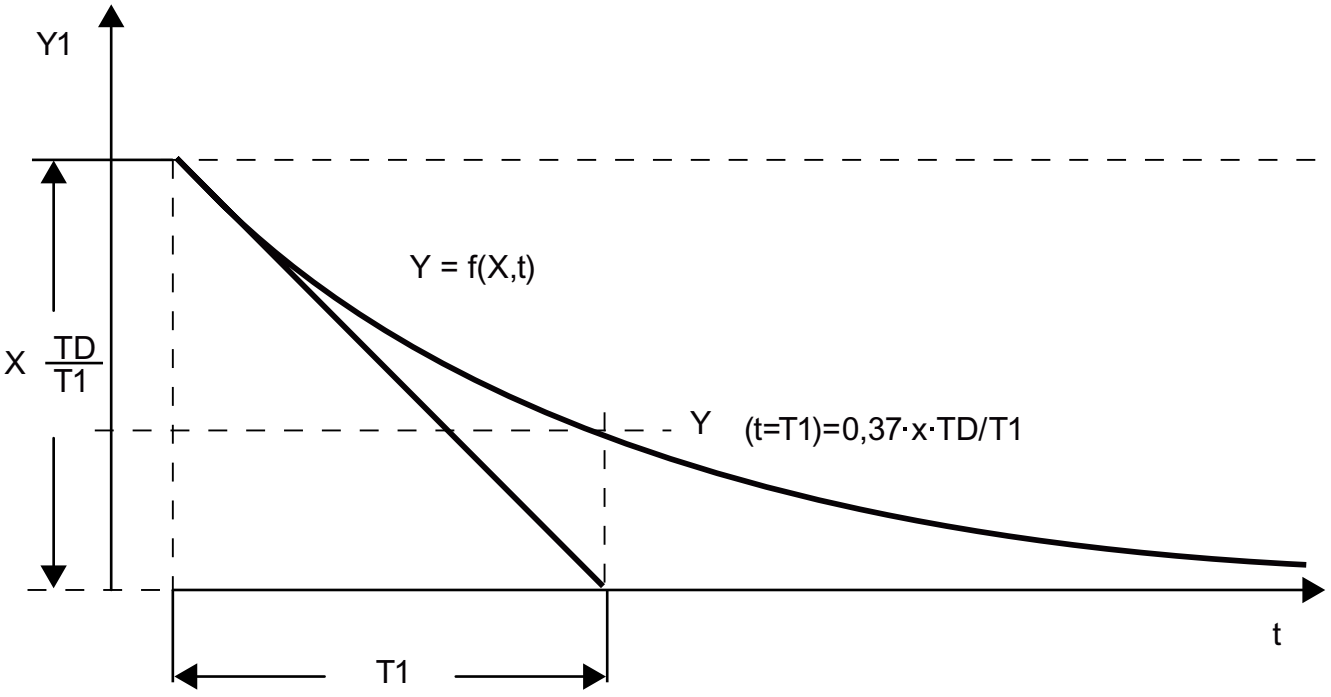
for $TD \neq 0$

The internal limitations for T1 and TD apply in this case. When TD=0, the output variables remain unchanged, as long as S=1.

Block diagram



XY diagram



Initialization

If input S is logic 1 at the initialization, the setting value SV is applied at output Y and YPT = T1/TD*(X-SV) set.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
T1	Smoothing time constant (ms)	0.0	SDTIME	
TD	Derivative-action time constant (ms)	0.0	SDTIME	
SV	Setting value	0.0	REAL	
S	Set	0	BOOL	
Y	Output variable DT1	0.0	REAL	
YPT	Output variable of smoothing element	0.0	REAL	

Configuration data

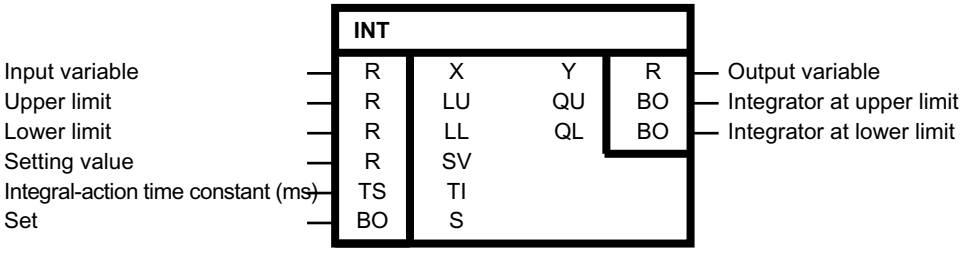
SIMOTION	✓
SINAMICS	✓

Can be loaded on-line	Yes
Special characteristics	-

8.1.8.5 INT

Integrator

Symbol



Brief description

- Block with integral behavior
- Set initial value
- Adjustable integral-action time constant
- Adjustable limits
- For normal integrator mode, a positive limit value must be specified for LU and a negative limit value for LL

Method of operation

The change in output variable Y is proportional to input variable X and inversely proportional to the integral-action time constant TI.

The output Y of the integrator can be limited via the inputs LU and LL. If the output reaches one of the two limits, a message is sent via the outputs QU or QL. If LL >= LU, then output Y = LU.

The calculation of the discrete values (TA is the sampling time in which the block is configured) is performed according to the following algorithm:

Algorithm:

$$Y_n = Y_{n-1} + \frac{TA}{TI} \cdot X_n$$

Y_n	Value of Y in scan interval n
Y_{n-1}	Value of Y in scan interval n-1
X_n	Value of X in scan interval n

When S = 1, the output variable Y is set to the setting value SV. Two functions can be realized via S:

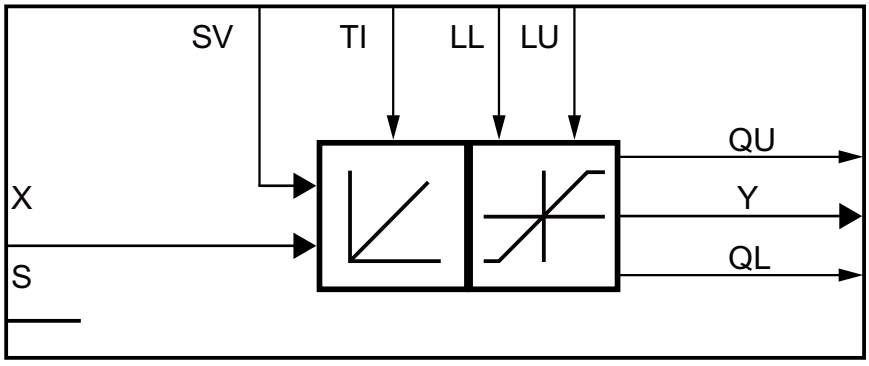
Track integrator (Y = SV)

The binary input is S = 1 and the setting value SV is changed. If applicable, the output makes a jump to the setting value immediately after the setting operation.

Set integrator to initial value SV

S is switched to 1. S is then set to 0, and the integrator starts from SV in the direction specified by the polarity of input variable X.

Block diagram



Truth table(s)

S	Condition	Y	QU	QL	Operating mode
0	LL<Yn-1+X×TA/TI<LU	Yn	0	0	Integration
0	Yn-1+X×TA/TI >= LU	LU	1	0	INT at upper limit
0	Yn-1+X×TA/TI <= LL	LL	0	1	INT at lower limit
1	LL<SV<LU	SVn	0	0	Set
1	SV >= LU	LU	1	0	INT at upper limit
1	SV <= LL	LL	0	1	INT at lower limit

Truth table for LL>= LU

S	Condition	Y	QU	QL	Operating mode
(any)	LL >= LU	LU	1	1	INT at upper limit

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
SV	Setting value	0.0	REAL	
TI	Integral-action time constant (ms)	0.0	SDTIME	
S	Set	0	0/1	
Y	Output variable	0.0	REAL	
QU	Integrator at upper limit	0	0/1	
QL	Integrator at lower limit	0	0/1	

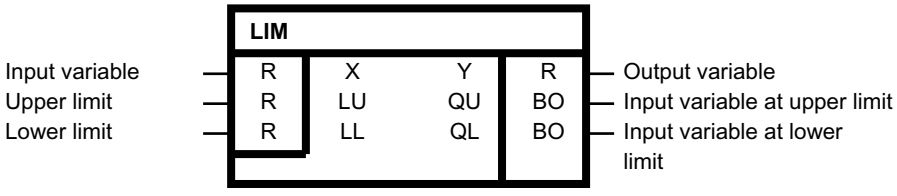
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.6 LIM

Limiter (REAL type)

Symbol



Brief description

- Block for the limitation
- Adjustable upper and lower limit
- Indication when set limits are reached

Method of operation

This block transfers the input variable X to its output Y, during which the input variable is limited depending on LU and LL.

If the input variable reaches the upper limit LU, then output QU = 1 is set.

If the input variable reaches the lower limit LL, then output QL = 1 is set.

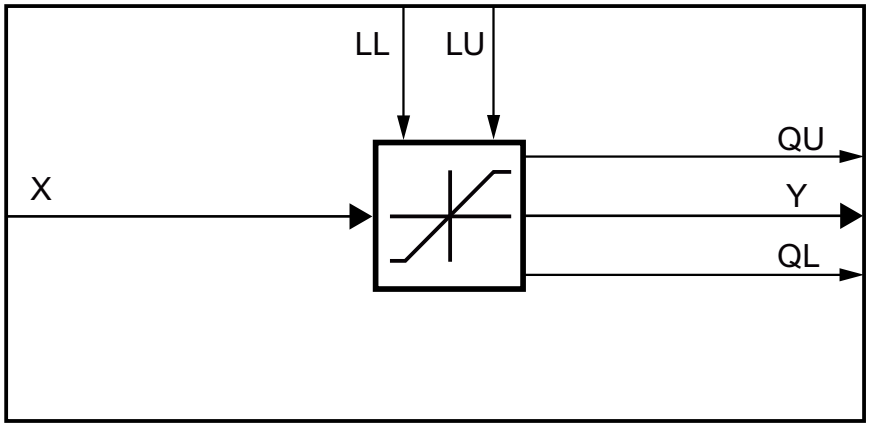
If the lower limit is greater than or equal to the upper limit, then output Y is set to the upper limit LU.

Algorithm:

$$Y = \begin{cases} LU \text{ for } X \geq LU \\ X \text{ for } LL < X < LU \\ LL \text{ for } X \leq LL \end{cases}$$

With the boundary condition: LL<LU

Block diagram



Truth table(s)

Condition	Y	QU	QL	Operating mode
LL < X < LU	X	0	0	
X >= LU	LU	1	0	Input variable at upper limit
X <= LL	LL	0	1	Input variable at lower limit

Truth table for LL>= LU

Condition	Y	QU	QL	Operating mode
LL >= LU	LU	1	1	Input variable at upper limit

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
Y	Output variable	0.0	REAL	
QU	Input variable at upper limit	1	0/1	
QL	Input variable at lower limit	1	0/1	

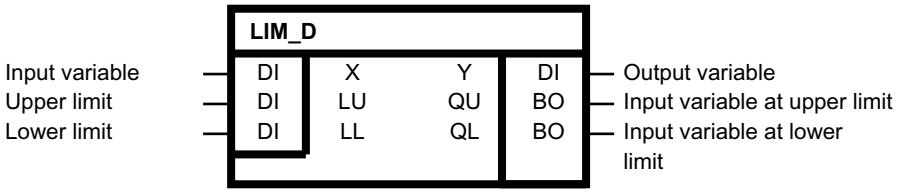
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.7 LIM_D

Limiter (DOUBLE INTEGER type)

Symbol



Brief description

- Block for the limitation of the DOUBLE INTEGER type
- Adjustable upper and lower limit
- Indication when set limits are reached

Method of operation

This block transfers the input variable X to its output Y, during which the input variable is limited depending on LU and LL.

If the input variable reaches the upper limit LU, then output QU = 1 is set.

If the input variable reaches the lower limit LL, then output QL = 1 is set.

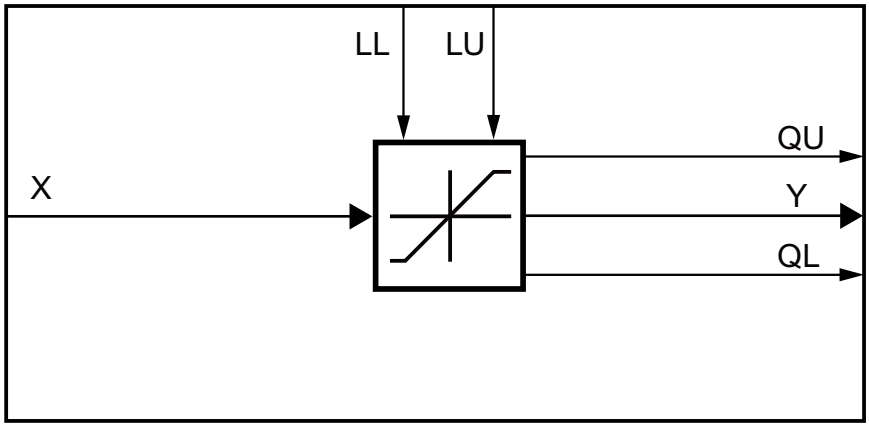
If the lower limit is greater than or equal to the upper limit, then output Y is set to the upper limit LU.

Algorithm:

$$Y = \begin{cases} LU \text{ for } X \geq LU \\ X \text{ for } LL < X < LU \\ LL \text{ for } X \leq LL \end{cases}$$

With the boundary condition: LL<LU

Block diagram



Truth table(s)

Condition	Y	QU	QL	Operating mode
LL < X < LU	X	0	0	
X >= LU	LU	1	0	Input variable at upper limit
X <= LL	LL	0	1	Input variable at lower limit

Truth table for LL >= LU

Condition	Y	QU	QL	Operating mode
LL >= LU	LU	1	1	Input variable at upper limit

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0	DINT	
LU	Upper limit	0	DINT	
LL	Lower limit	0	DINT	

Block connection	Description	Default	Value range	Attrib-utes
Y	Output variable	0	DINT	
QU	Input variable at upper limit	1	0/1	
QL	Input variable at lower limit	1	0/1	

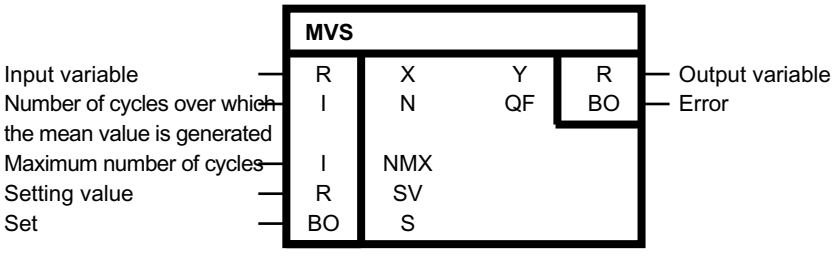
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.8.8 MVS

Sliding-type mean value generator

Symbol



Brief description

The block calculates a sliding-type mean value via the input variable X.

Method of operation

The mean value is generated over the last N cycles.

$$Y_k = \frac{1}{N} \cdot \sum_{i=k-(N-1)}^{k} X_i$$

X_k = X in cycle k

k = 0 is the current cycle

The number of cycles can be changed in the range $1 \leq N \leq NMX$. The maximum number of cycles is specified through NMX and cannot be changed during operation. The block limits input N to

the range of 1 ≤ N ≤ NMX. The buffer for the input values is always filled up to NMAX, irrespective of N. In this way, the block can re-determine the current mean value via all variables when there is a change in the window length.

The mean value is set to set value SV as long as S = 1.

Initialization

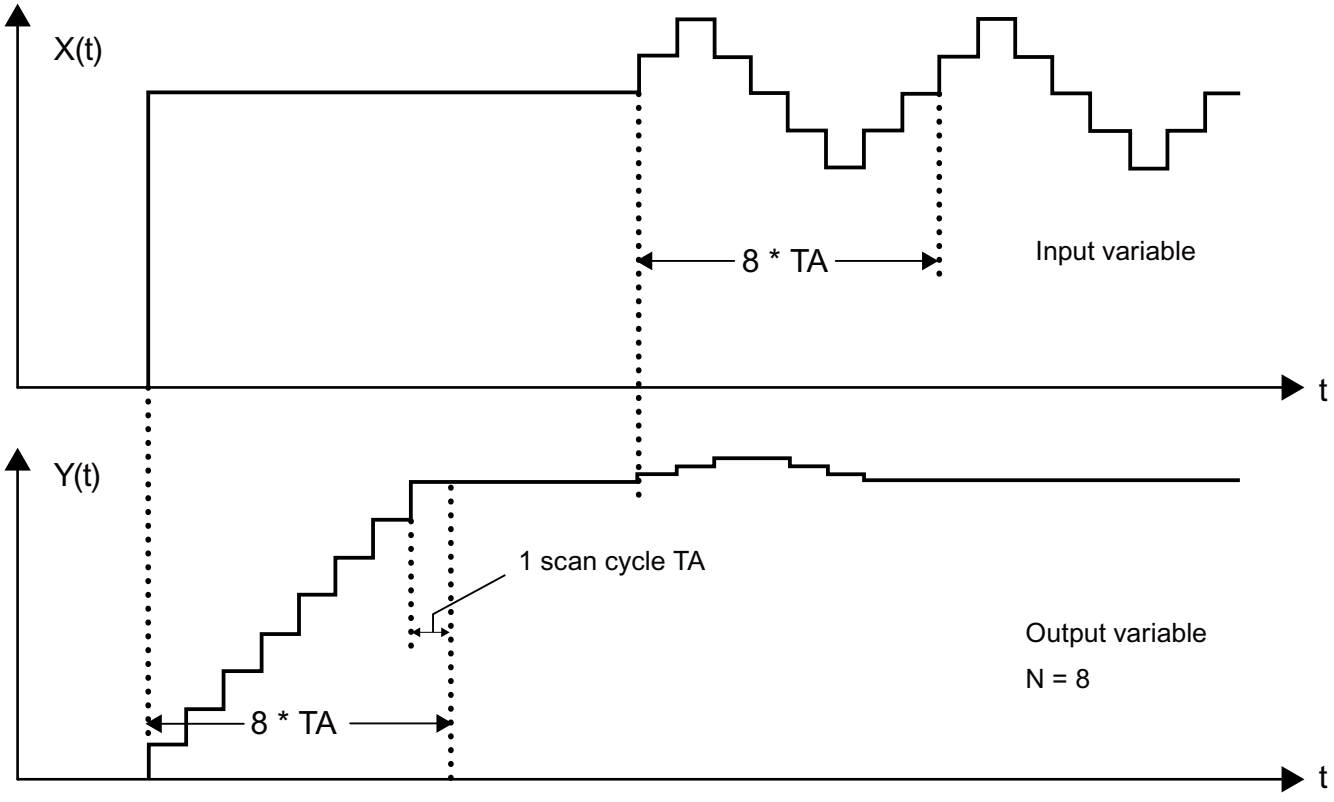
At the initialization, NMX is used to specify the maximum size of the mean value buffer for the X values. For this reason, NMX should be set to the maximum value of N required under operating conditions. The value of NMX is limited to the range of values between 1 and 1,000. If there is not enough memory for NMX on the target device, or if NMX is limited, output QF is set to 1 and output Y retains its default value during cyclic operation. As NMX cannot be changed dynamically during operation, NMX should be specified as a constant.

Application areas

The block can be used as ramp-function generator or filter block for the mean value generation. It acts as a low pass and band-stop filter for frequencies f_k.

$$f_k = \frac{k}{N \cdot T_A}$$

k = 1, 2, ...



Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
N	Number of cycles over which the mean value is generated	10	1...1000	
NMX	Maximum number of cycles	100	1...1000	
SV	Setting value	0.0	REAL	
S	Set	0	0/1	
Y	Output variable	0.0	REAL	
QF	Error	0	0/1	

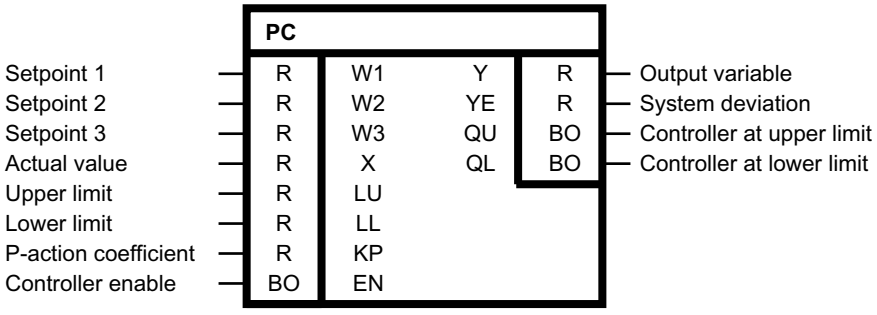
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be inserted on-line	Yes
Special character-istics	-

8.1.8.9 PC

P controller

Symbol



Brief description

- P controller with 3 setpoint inputs and 1 actual value input
- Sign reversal of actual value in block
- Indication when set limits are reached
- For normal controller operation, a positive limit value must be specified for LU and a negative limit value for LL.

Method of operation

The three setpoints W1, W2, and W3 are added and the actual value X is subtracted from the setpoint total. The result YE is multiplied by the proportional coefficient KP and given to output Y.

Algorithm:

$Y = KP \cdot YE = KP \cdot (W1 + W2 + W3 - X)$

$YE = W1 + W2 + W3 - X$

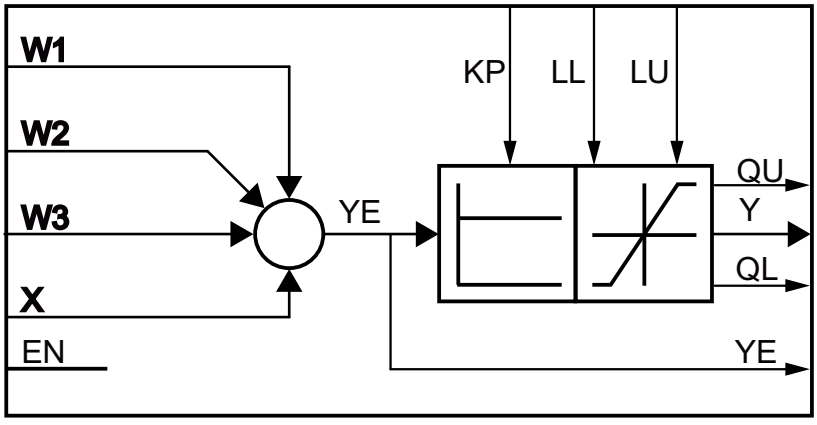
The system deviation YE is always calculated, irrespective of the operating mode, and is lead out separately.

The output Y of the controller can be limited via the inputs LU and LL. If the output Y reaches one of the two limits, a message is sent via the outputs QU and QL. If LL >= LU, then output Y = LU.

The controller is enabled with EN = 1. If EN = 0, the output variable Y is set to zero. The controller is disabled. The binary outputs QU and QL are treated in this case as if KP*YE equaled zero.

The controller operates inverted when a negative KP value is selected (inversion amplifier).

Block diagram



Truth table(s)

EN	Condition	Y	QU	QL	Operating mode
0	LL< 0 <LU	0	0	0	Controller disable
0	LU <= 0	0	1	0	Controller disable
0	LL >= 0	0	0	1	Controller disable
1	LL<YE * KP<LU	KP × YE	0	0	Controller enable
1	YE * KP >= LU	LU	1	0	Controller at upper limit
1	YE * KP <= LL	LL	0	1	Controller at upper limit

Truth table for LL >= LU

EN	Condition	Y	QU	QL	Operating mode
0	None	0	1	1	Controller disable
0	LL >= LU	LU	1	1	Controller at upper limit

Block connections

Block connection	Description	Default	Value range	Attributes
W1	Setpoint 1	0.0	REAL	
W2	Setpoint 2	0.0	REAL	
W3	Setpoint 3	0.0	REAL	
X	Actual value	0.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
KP	P-action coefficient	0.0	REAL	
EN	Controller enable	0	0/1	
Y	Output variable	0.0	REAL	
YE	System deviation	0.0	REAL	
QU	Controller at upper limit	1	0/1	
QL	Controller at lower limit	1	0/1	

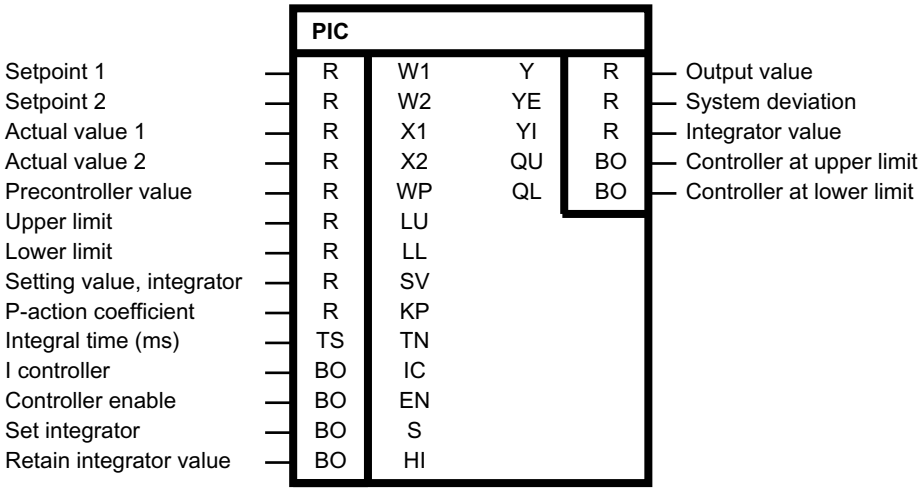
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.10 PIC

PI controller

Symbol



Brief description

- Universal PI controller, can be switched to P controller or I controller modes. Can be used as a speed controller or a primary controller. Suitable for dynamic override control

Flexible integrator functions:

- Set initial value ⇒ Load SV to integrator
- Retain current value of integrator ⇒ P controller
- Integrator control by SV
- Integrator control by controller limiting
- Gain shutdown ⇒ I controller

Overall controller functions:

- Independent setting and modification of the following variables during operation:
  - Proportional coefficient KP
  - Integral action time TN
  - Controller limits LU and LL
  - Precontroller value WP, e.g. for acceleration injection
  - Second actual value input X2, e.g. for droop injection
- Indication when set limits are reached

Method of operation

The actual value total (X1+X2) is subtracted from the setpoint total (W1+W2) according to the equation:

YE = (W1 + W2) - (X1 + X2)

The result, system deviation YE, is then multiplied by the adjustable proportional coefficient KP. The product is carried to the output summation device and the integrator. The adjustable integral time TN determines the integration behavior of the controller. The change in output variable YI is proportional to input variable KP*YE and inversely proportional to the integral-action time TN. The integrator value YI is also given to the output summation device. Another value with the correct sign can be added to output value Y via input WP.

Discrete values are calculated according to the algorithm:

Algorithm:

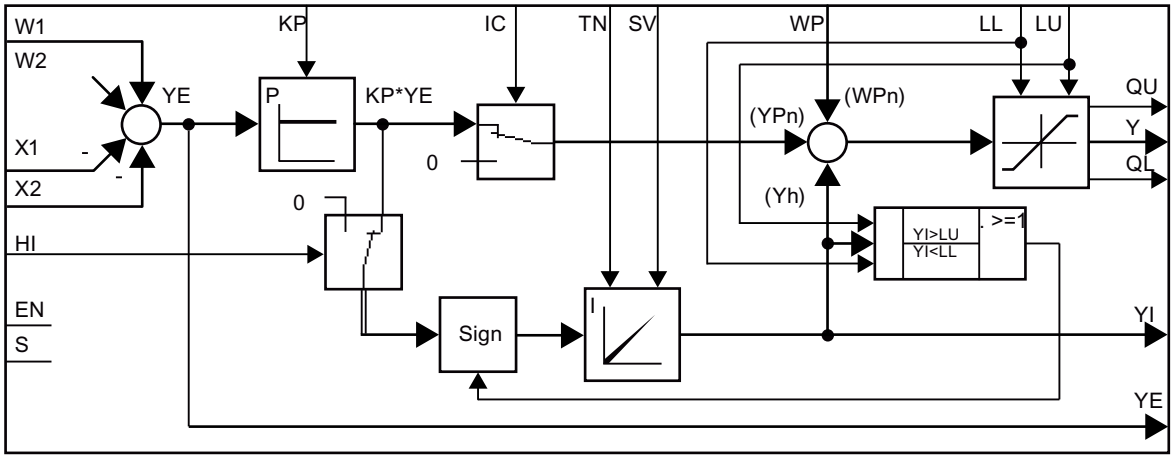
$$Y_n = Y_{n-1} + KP \cdot \left[\left(1 + \frac{TA}{TN} \right) \cdot YE_n - YE_{n-1} \right]$$

With the boundary conditions: LL<Y<LU and LL<LU

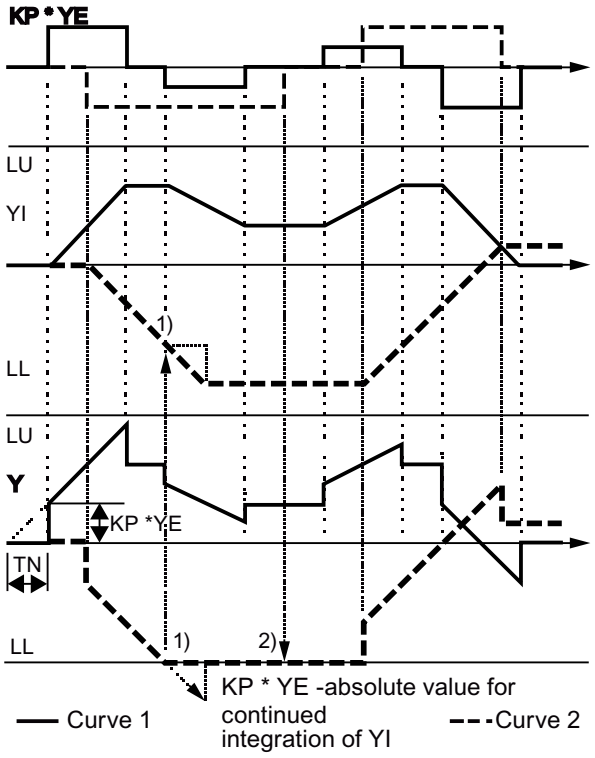
Y_n	Value of Y in scan interval n
Y_{n-1}	Value of Y in scan interval n-1

TA is the sampling time in which the block is configured.

Block diagram



XY diagram



Curves 1 and 2 show the characteristic of Y and YI during YE jumps:

- Curve 1, normal operation, no limiting
- Curve 2, with use of limiting (e.g. LL)

For 2) a decay of YE*KP is expected, but this is canceled by the continued integration in 1).

Operating modes and control of the controller

Output value Y and integrator value YI of the controller can be limited via the inputs LU and LL. When the set limits are reached by output variable Y, a message is issued with QU = 1 of QL = 1.

The following priority sequence applies for the control inputs:

EN before IC before S before HI.

Command input at the control inputs:

Control input	Value	Functions
EN	1	Controller enable
IC	1	Change-over from PI controller to I controller
S	1	Accept integrator setting value, do not integrate
HI	1	Retain integrator output YI, do not integrate

The combination of commands at the control inputs and the possible operating modes can be found in the truth tables.

In normal controller operation, LL <= 0 <= LU and LL < Yn < LU. However, other settings, explained below, are possible. To this end, the algorithm is converted appropriately:

$Y_n = KP \cdot YE_n + YI_n + WP_n$

There are 5 different operating conditions in conjunction with LU and LL:

No.	Condition	Y_n
LL < LU		
1	$LL < KP*YE_n + YI_n + WP_n < LU$	$KP*YE_n+YI_n+WP_n$
2	$KP*YE_n + YI_n + WP_n >= LU$	LU
3	$KP*YE_n + Y_n + WP_n <= LL$	LL
LL=LU		
4	None	LU
LL > LU		
5	None	LU

Integrator control by own limiting

If output Y comes up against one of the set limitations LL or LU during the control process, integrator YI will continue to run if applicable, until it comes up against the limitation itself and is retained there.

If the controller is at the limit and the limit value is changed, output Y momentarily assumes the new value as long as an override is defined. However, the integrator is updated to the new limiting value at change velocity YIn.

Truth table(s)

Operating condition 1

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
0	*	*	*	*	0	0	Controller disable	KP, RN, WP, LU, LL, YE any value

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
1	0	0	0	$KP*YE_n \times TA/TN$	$YI_{n-1}+\Delta YI_n$	$KP*YE_n+YI_n+WP_n$	PI controller	Controller enable, normal operation
1	1	0	0	$KP*YE_n*TA/TN$	$YI_{n-1}+\Delta YI_n$	YI_n+WP_n	I controller	P component = 0
1	0	1	*	*	SV_n	$KP*YE_n+YI_n+WP_n$	P controller, integrator guidance	$YI_n = SV_n$
1	1	1	*	*	SV_n	YI_n+WP_n	I controller, integrator guidance	$YI_n = SV_n$
1	0	0	0	0	YI_{n-1}	$KP*YE_n+YI_n+WP$	P controller, integrator = constant	$YI_n = YI_{n-1}$
1	1	0	0	0	YI_{n-1}	YI_n+WP_n	I controller, integrator = constant	$YI_n = YI_{n-1}$

*= any value

Operating condition 2

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
1	0	0	0	KP*YE_n×TA/TN	$YI_{n-1}+\Delta YI_n$ for YI_{n-1} < $LUYI_{n-1}$- ΔYI_n for YI_{n-1}> LULU for YI_{n-1} = LU	LU	PI controller at upper limit	YI_n integrated -> LU, possibly with (-)
1	1	0	0	KP*YE_n*TA/TN	$YI_{n-1}+\Delta YI_n$ for YI_{n-1} < $LUYI_{n-1}$- ΔYI_n for YI_{n-1}> LULU for YI_{n-1} = LU	LU	I controller at upper limit	YI_n integrated -> LU, possibly with (-)
1	0	1	*	*	SV_n for SV_n < LU-LU for SV_n >= LU	LU	P controller at upper limit	$YI_n = SV_n$ or YI_n = LU
1	1	1	*	*	SV_n for SV_n < LU-LU for SV_n >= LU	LU	I controller at upper limit	$YI_n = SV_n$ or YI_n = LU, P component = 0
1	0	0	1	0	YI_{n-1}	LU	P controller, integrator = constant	$YI_n = YI_{n-1}$ or YI_{n-1} = LU
1	1	0	1	0	YI_{n-1}	LU	I controller, integrator = constant	$YI_n = YI_{n-1}$ or YI_{n-1} = LU, P component = 0

*= any value

Operating condition 3

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
1	0	0	0	KP*YE_n×TA/TN	$YI_{n-1}+\Delta YI_n$ for YI_{n-1} < $LLYI_{n-1}$- ΔYI_n for YI_{n-1} > LLLL for YI_{n-1} = LL	LL	PI controller at lower limit	YI_n integrated -> LL, possibly with (-)
1	1	0	0	KP*YE_n*TA/TN	$YI_{n-1}+\Delta YI_n$ for YI_{n-1} < $LLYI_{n-1}$- ΔYI_n for YI_{n-1} > LLLL for YI_{n-1} = LL	LL	I controller at lower limit	YI_n integrated -> LL, possibly with (-)
1	0	1	*	*	SV_n for SV_n < LLLL for SV_n >= LL	LL	P controller at lower limit	$YI_n = SV_n$ or YI_n = LL
1	1	1	*	*	SV_n for SV_n < LLLL for SV_n >= LL	LL	I controller at lower limit	$YI_n = SV_n$ or YI_n = LL, P component = 0
1	0	0	1	0	YI_{n-1}	LL	P controller, integrator = constant	$YI_n = YI_{n-1}$ or YI_{n-1} = LL
1	1	0	1	0	YI_{n-1}	LL	I controller, integrator = constant	$YI_n = YI_{n-1}$ or YI_{n-1} = LL, P component = 0

*= any value

Operating condition 4

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
1	*	*	*	*	*	LL=LU	See operating condition 2 or 3	-

*= any value

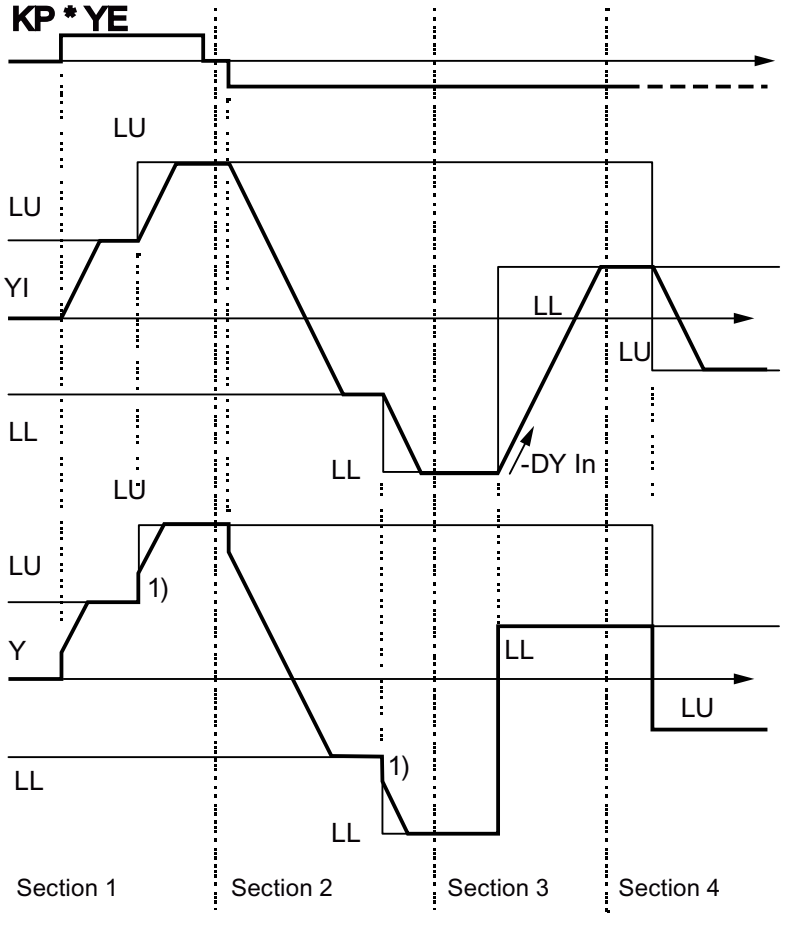
Operating condition 5

EN	IC	S	HI	ΔYI_n	YI_n	Y_n	Operating mode	Remark
1	*	*	*	$KP*YE_nTA/TN$	$YI_{n-1}+\Delta YIn$ for $YI_{n-1} < LU-YI_{n-1}-\Delta YI_n$ for $YI_{n-1} > LU-LU$ for $YI_{n-1} = LU$	LU	PI controller at upper limit	-

Depending on the direction of the limit value change, the sign of the integration is inverted if necessary.

Transfer functions

Transfer function during controller override for operating conditions 2, 3 and 5:



Section 1: Characteristic with LUn > LUn-1 according to operating condition 2

Section 2: Characteristic with LLn < LLn-1 according to operating condition 3

Section 3: Characteristic with LLn > LLn-1 according to operating condition 3, for

limit shift relative to the control direction with

sign inversion at the integrator input

Section 4: Characteristic with LLn > LUn according to operating condition 5

1) Jump by KP*YE, because the integrator was run up to the limit.

Change-over from PI mode to I mode

When EN = 1 and IC = 1, the P component is retained at 0, and the controller is switched from PI mode to I mode. Output Y assumes integrator value YI. If this occurs during the control process, then a jump by -KP*YE will occur at output Y During a reset to IC = 0, the P component is reset to the current value of KP*YE. The controller again exhibits PI behavior. If this occurs during the control process, then a jump by KP * YE will occur at output Y.

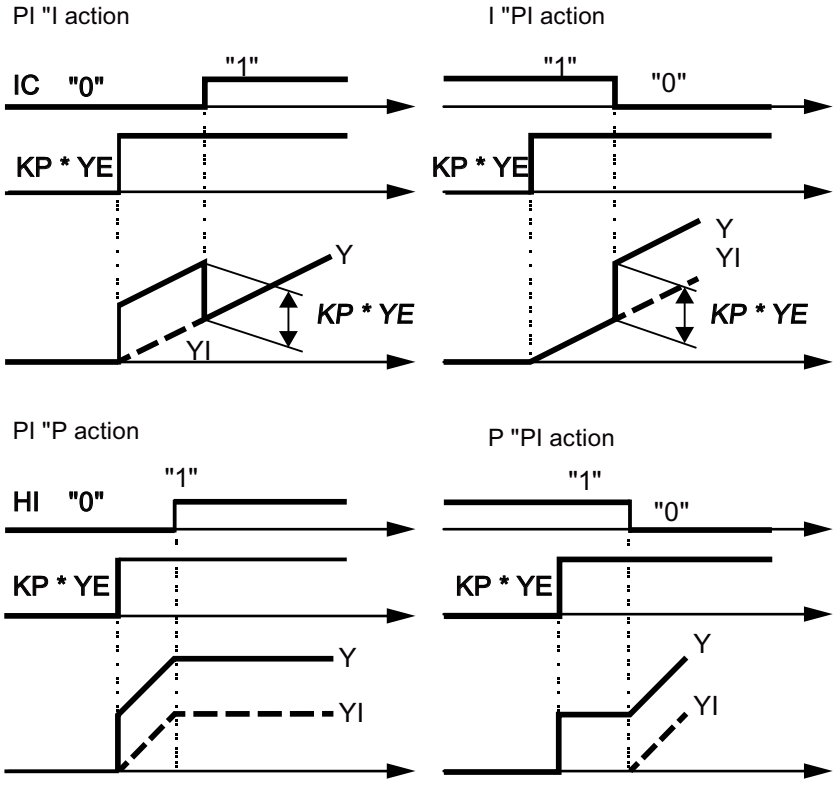
Change-over from PI mode to P mode

If block inputs EN = 1 and HI = 1, the integrator YI is retained, and a bumpless controller change-over takes place from PI mode to P mode. YI continues to act as an addend on output Y.

During a reset to HI = 0, the integrator is enabled again. The controller again exhibits PI behavior.

Transfer functions

Transfer functions during changeover without controller override: Examples for EN=1∧S=0



Block connections

Block connection	Description	Default	Value range	Attributes
W1	Setpoint 1	0.0	REAL	
W2	Setpoint 2	0.0	REAL	
X1	Actual value 1	0.0	REAL	
X2	Actual value 2	0.0	REAL	
WP	Precontroller value	0.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
SV	Setting value, integrator	0.0	REAL	
KP	P-action coefficient	0.0	REAL	
TN	Integral time (ms)	0.0	SDTIME	
IC	I controller	0	0/1	
EN	Controller enable	0	0/1	
S	Set integrator	0	0/1	
HI	Retain integrator value	0	0/1	
Y	Output value	0.0	REAL	
YE	System deviation	0.0	REAL	
YI	Integrator value	0.0	REAL	
QU	Controller at upper limit	1	0/1	
QL	Controller at lower limit	1	0/1	

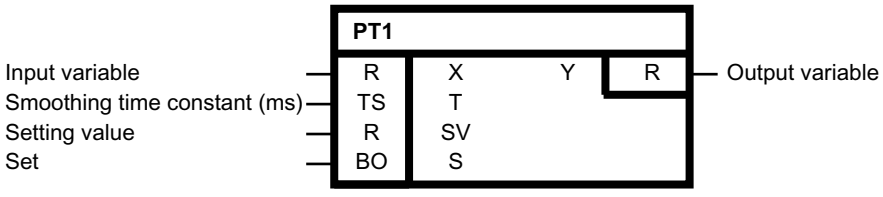
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.11 PT1

Delay element

Symbol



Brief description

- First-order delay element with setting function
- Use as smoothing element

Method of operation

Setting function not active (S = 0)

Input variable X, dynamically delayed by smoothing time constant T, is given to output Y.

T determines the steepness of the rise of the output variable. It indicates the time at which the transfer function has risen to 63% of its end value.

After t = 3T, the transfer function reaches approximately 95% of its end value.

The internally fixed proportional gain is 1 and does not vary.

If T/TA (T/TA > 10) is sufficiently large, the transfer function corresponds to the characteristic of

$Y(t) = X \cdot (1-e^{-t/T})$

with t = n * TA.

Discrete values are calculated according to the algorithm:

Algorithm:

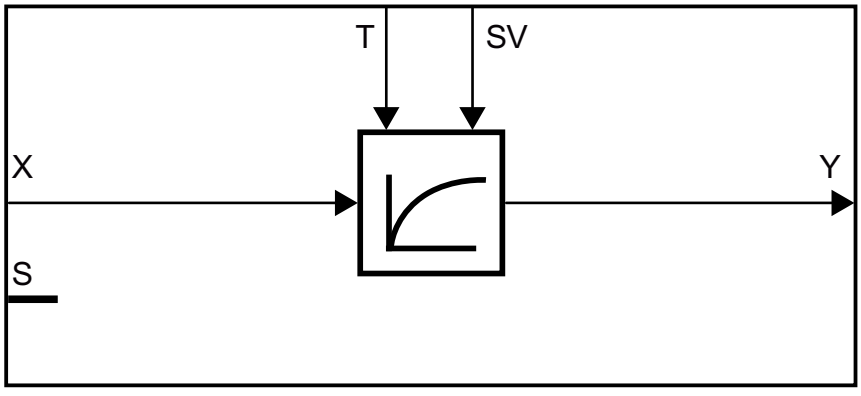
$$Y_n = Y_{n-1} + \frac{TA}{T} \cdot (X_n - Y_{n-1})$$

Y_n	Value of Y in scan interval n
Y_{n-1}	Value of Y in scan interval n-1
X_n	Value of X in scan interval n

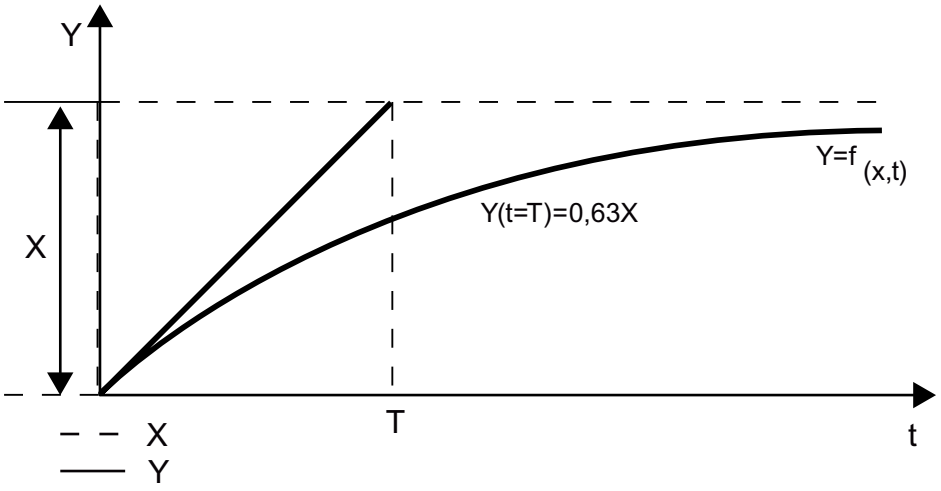
Setting function active (S = 1)

When the setting function is active, the actual setting value SVn is accepted at the output variable: Yn = SVn

Block diagram



Time diagram



Initialization

If input S is logic 1 at the initialization, the setting value SV is applied at output Y.

Block connections

Block connection	Description	Default	Value range	Attributes
X	Input variable	0.0	REAL	
T	Smoothing time constant (ms)	0.0	SDTIME	
SV	Setting value	0.0	REAL	
S	Set	0	0/1	
Y	Output variable	0.0	REAL	

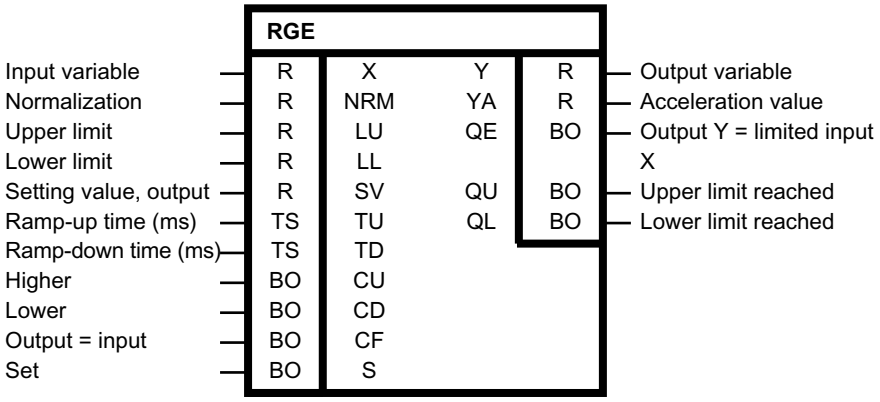
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded online	Yes
Special characteristics	-

8.1.8.12 RGE

Ramp-function generator

Symbol



Brief description

- Ramp-function generator for limiting the change velocity of input variable X
- Output variable can be limited

Independent setting and modification of the following variables during operation:

- Ramp-up and ramp-down time
- Output limits LU and LL
- Setting value

Flexible ramp-function generator functions:

- Integrating correction to setpoint X
- Setting of initial value for ramp-function generator output (-> load SV to integrator)
- Integrating increase and decrease of ramp-function generator output

Method of operation

The block contains an integrator with two integration time constants that can be set separately. Output Y changes according to the algorithm:

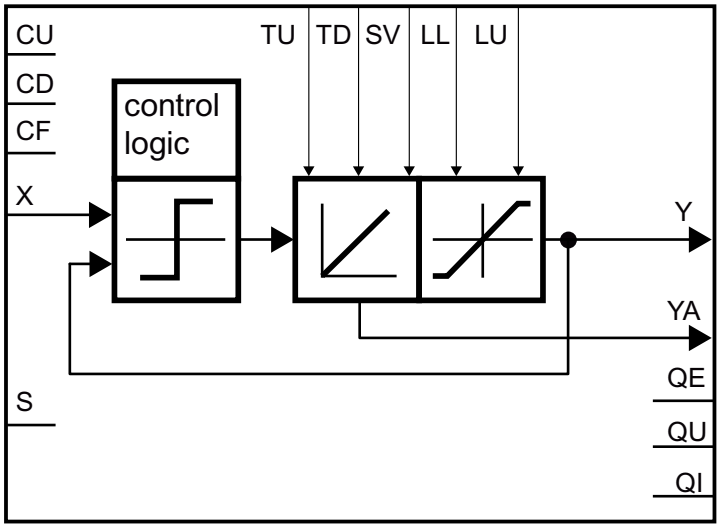
$Y_n = Y_{n-1}+YA_n$

The acceleration value YA is calculated separately for the ramp up and ramp down and is output on an output.

The process in which output value Y moves away from zero is called ramp up.

The process in which output value Y moves toward zero is called ramp down.

Block diagram



The following applies for the ramp up acceleration value:

$$YA = \frac{TA}{TU} \cdot NRM \text{ for } Y > 0$$

$$YA = -\frac{TA}{TU} \cdot NRM \text{ for } Y < 0$$

The following applies for the ramp down acceleration value:

$$YA = -\frac{TA}{TD} \cdot NRM \text{ for } Y > 0$$

$$YA = \frac{TA}{TD} \cdot NRM \text{ for } Y < 0$$

The change-over between ramp up time and ramp down time takes place during a direction change or at the zero crossover of the transfer function.

The operating mode is predefined by means of control logic, depending on the logic states of the control inputs S, CF, CU, and CD.

The output variable can be limited by means of the inputs LU and LL. When the set limits are reached by Y, the binary outputs QU or QL are set to 1. The binary output QE becomes 1 when Y = X.

Ramp-up time and ramp-down time

The ramp-up time TU is the time in which the absolute value of the output variable increases by NRM.

The ramp-down time TD is the time in which the absolute value of the output variable decreases by NRM. Ramp up time and ramp down time can be selected differently.

The smaller TA/TU or TA/TD is, the smaller is the amplitude change on Y from one scan time to the next. TA is the scan time in which the block is processed.

The following priority sequence applies for the control inputs:

S before CF before CU and CD.

Function of control inputs:

S=1	Load setting value SV in integrator; do not integrate.
CF=1	Correct output Y to setpoint X with integration.
CU=1	Correct output Y in the direction LU with integration
CD=1	Correct output Y in the direction LL with integration

Operating modes and control of the ramp-function generator

The combination of commands at the control inputs and the possible operating modes can be found in the truth tables.

In normal ramp-up mode, LL <= 0 <= LU and LL< Yn < LU. However, other settings, explained below, are possible.

The following applies to the setting LL >= LU: The LU limit is dominant over the LL limit.

Behavior of the integrator at the limitation

If output Y comes up against one of the set limits LL or LU during the control process, the integrator value is retained. The output value Y is then kept constant until the integrator value leaves the limit due to changes in the input variables.

If the integrator is at the limit and the limit value is changed, the integrator behaves differently depending on the direction of the limit value change.

If the absolute value of a limit value is increased and it has been defined in the control logic that the ramp-function generator should run in the same direction, the integrator continues to integrate from the previously held value according to the set ramp up time, until the output once again comes up against the limit value.

If the absolute value of the limit value is reduced, the integrator integrates from the previously held value according to the set ramp down time, until the output again reaches the limit value.

TU and TD are limited internally: TU >= TA, TD >= TA

Truth table(s)

S	CF	CU	CD	Y_{An}	Y_n	Operating mode	Remark
0	0	0	0	0	Y_{n-1}	Stop	Y is constant

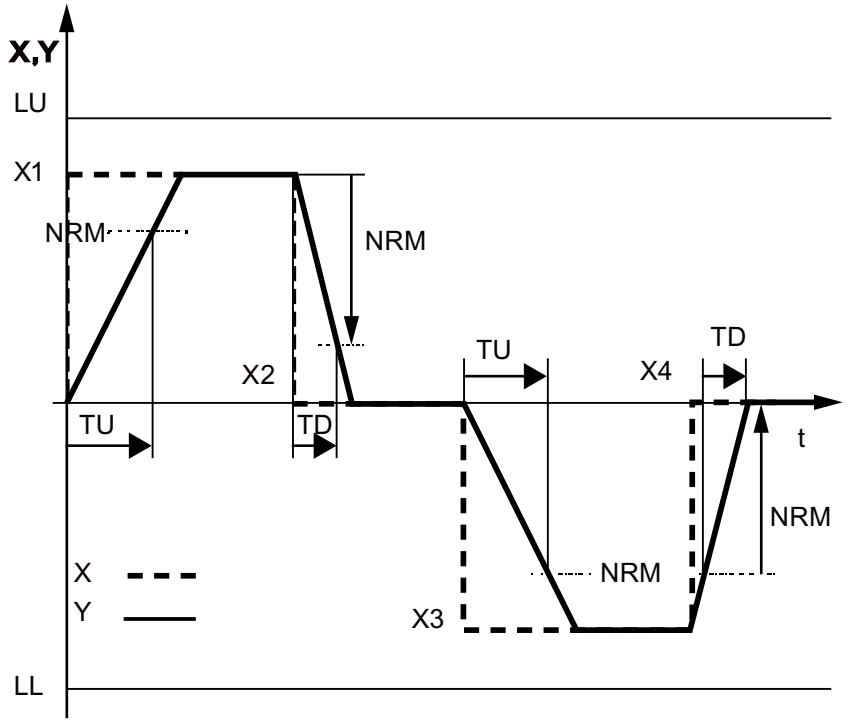
LL < LU and LL < actual value Y_{n-1} < LU

S	CF	CU	CD	Y_{An}	Y_n	Operating mode	Remark
1	*	*	*	Jump	SV_n	Set output to SV	Any SV, fixed or variable
0	1	*	*	TA/ TU;TA/TD	$Y_{n-1}+YA_n$	Normal mode Y -> X	TU for [X > Y ∧ Y >= 0] ∨ [X <Y ∧ Y <= 0] TD for [X > Y ∧ Y <0] ∨ [X < Y ∧ Y > 0]

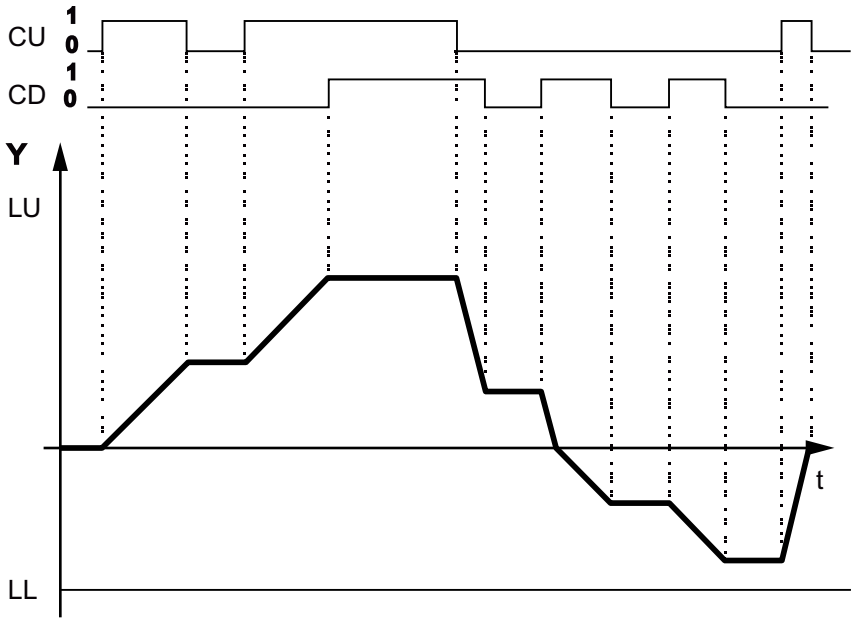
S	CF	CU	CD	Y_{An}	Y_n	Operating mode	Remark
0	0	1	0	TA/ TU(TA/TD)	$Y_{n-1}+YA_n$	Touch upper limit value Y -> LU	TU, TD as before, depending on start position
0	0	0	1	TA/ TD(TA/TU)	$Y_{n-1}+YA_n$	Touch lower limit value Y -> LL	TU, TD as before, depending on start position

* Arbitrary

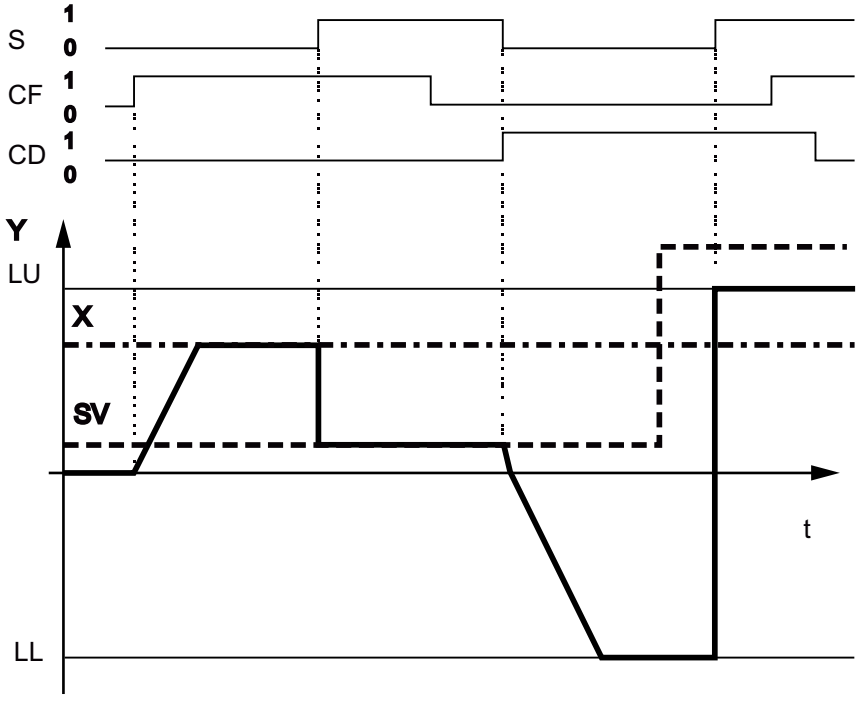
Transfer function



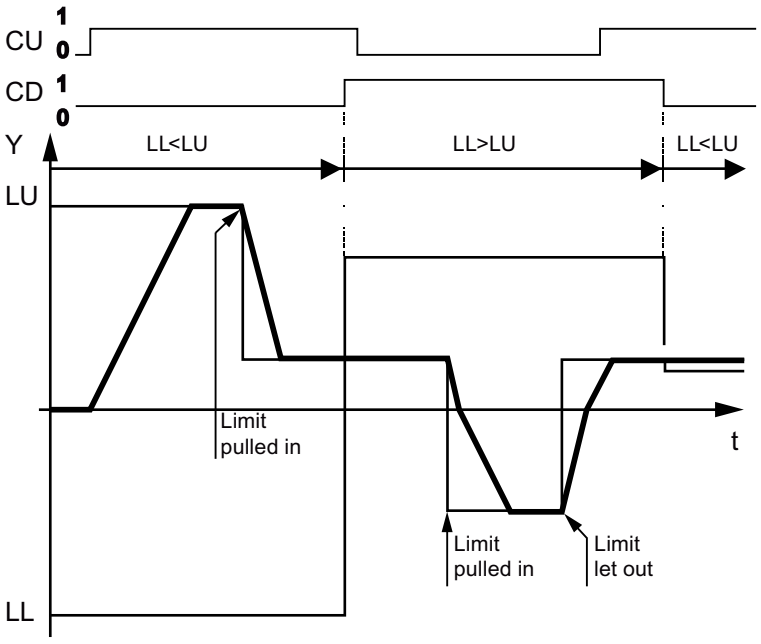
Example 1: CF = 1 with LL < LU and LL < X < LU, and X_1=1.5, X_2=X_4 =0.0, X_3=–1.5, LU=2.0, LL=–2.0, TU > TD



Example 2: Motor potentiometer function with CU and CD and with LL < LU



Example 3: Set integrator with LL < LU



Example 4: Change and swap limits

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
NRM	Normalization	1.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
SV	Setting value, output	0.0	REAL	
TU	Ramp-up time (ms)	0.0	SDTIME	
TD	Ramp-down time (ms)	0.0	SDTIME	
CU	Higher	0	0/1	
CD	Lower	0	0/1	
CF	Output = input	0	0/1	
S	Set	0	0/1	
Y	Output variable	0.0	REAL	
YA	Acceleration value	0.0	REAL	
QE	Output Y = limited input X	0	0/1	
QU	Upper limit reached	0	0/1	
QL	Lower limit reached	0	0/1	

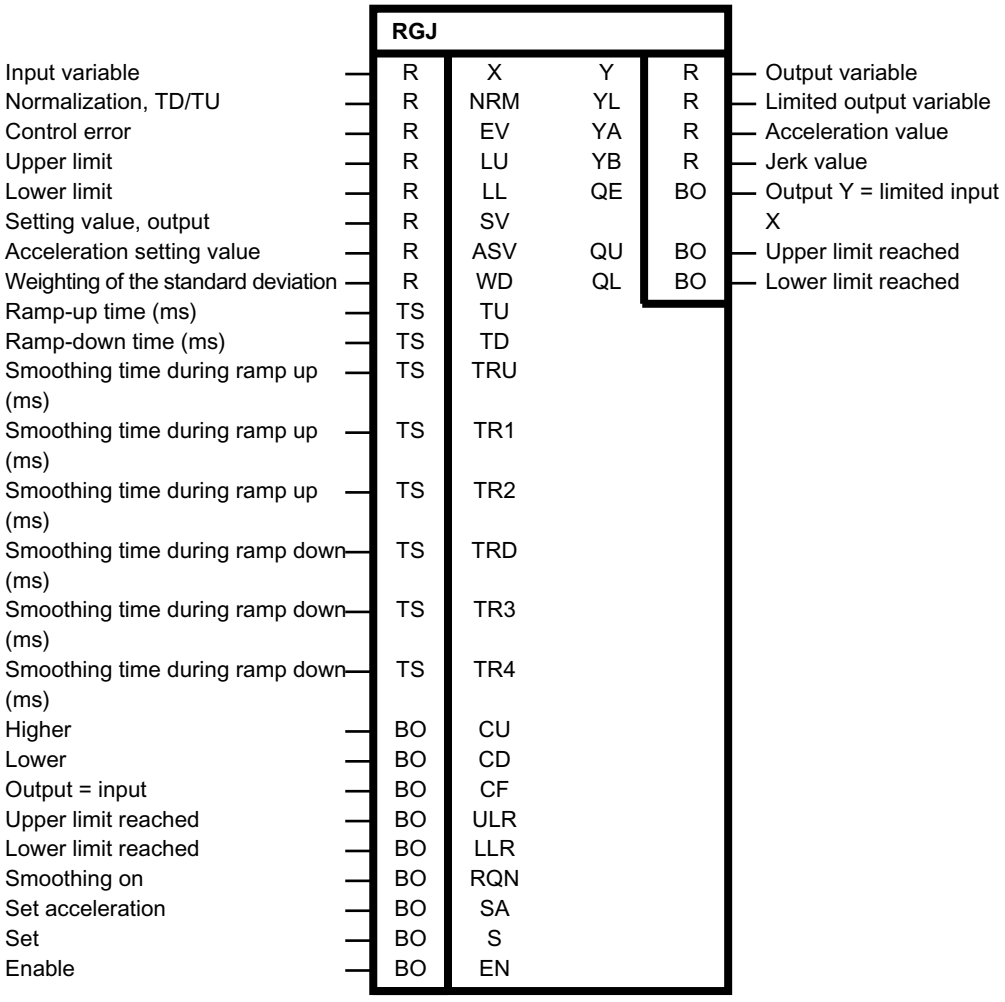
Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special characteristics	-

8.1.8.13 RGJ

Ramp-function generator with jerk limiting

Symbol



Brief description

- Ramp-function generator with jerk limiting and correction
- Ramp-function generator functions
- Set output Y or acceleration YA
- Correct ramp-function generator output to setpoint X with integration and jerk limiting
- Integrating increase and decrease of ramp-function generator output
- Correction of the ramp-function generator according to the system deviation of a lower-level controller during limiting

Method of operation

The block limits the acceleration (change in velocity) and the jerk (change in acceleration) of setpoints.

The following algorithms apply:

$$Y_n = Y_{n-1} + YA_n$$

$$YA_n = YA_{n-1} + YB_n$$

Acceleration value YA and jerk YB are calculated separately for ramp up and ramp down. This requires configuration of the time values ramp up time TU and smoothing time during ramp up TRU as well as ramp down TD and smoothing time during ramp down TRD.

The following applies for the acceleration value YA outside of the smoothing time during ramp up:

$$YA = YA_{max} = \frac{TA}{TU} \cdot NRM \text{ for } Y > 0$$

$$YA = YA_{max} = -\frac{TA}{TU} \cdot NRM \text{ for } Y < 0$$

The following applies for the acceleration value YA outside the smoothing time during ramp-down:

$$YA = YA_{max} = -\frac{TA}{TD} \cdot NRM \text{ for } Y > 0$$

$$YA = YA_{max} = \frac{TA}{TD} \cdot NRM \text{ for } Y < 0$$

The following applies for the jerk value YB during the smoothing time during ramp up:

$$YB = \frac{TA \cdot YA_{max}}{TRU}$$

or

$$YB = \frac{TA \cdot YA_{max}}{TR1} \quad YB = \frac{TA \cdot YA_{max}}{TR2}$$

The following applies for the jerk value YB during the smoothing time during ramp down:

$$YB = \frac{TA \cdot YA_{max}}{TRD}$$

or

$$YB = \frac{TA \cdot YA_{max}}{TR3} \quad YB = \frac{TA \cdot YA_{max}}{TR4}$$

The operating mode is predefined by means of control logic, depending on the logic states of the binary variables EN, S, SA, CF, CU, and CD.

Input variable X and thus indirectly output variable Y are limited by means of the block inputs LU and LL. When the set limits are reached by Y, a message is issued to the binary outputs with QU = 1 or QL = 1.

Binary output QE becomes 1 when output variable Y equals the limited value of input variable X.

A ramp up process is subdivided into three phases:

Phase 1

When setpoint X increases, the maximum jerk YB (depending on TRU or TR1) is defined in the first part. Thus, the acceleration increases proportionally over time; in this smoothing phase, output Y rises quadratically over time.

Phase 2

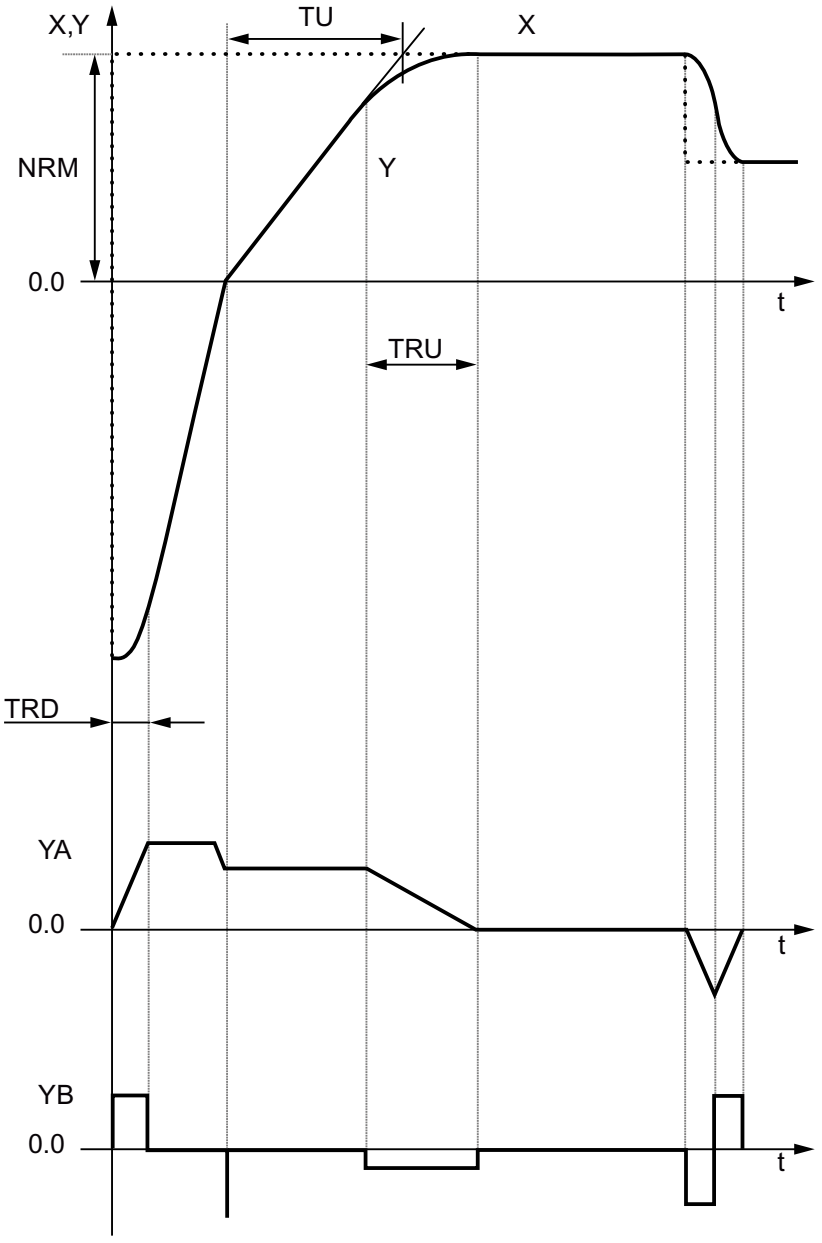
Once the maximum acceleration YA has been reached according to the defined ramp-up time TU, the acceleration is constant. Output variable Y rises proportionally over time.

Phase 3

In the third part, the acceleration is decreased proportionally over time. In this smoothing phase, output variable Y approaches input variable X on YB quadratically over time (depending on TRU or TR2).

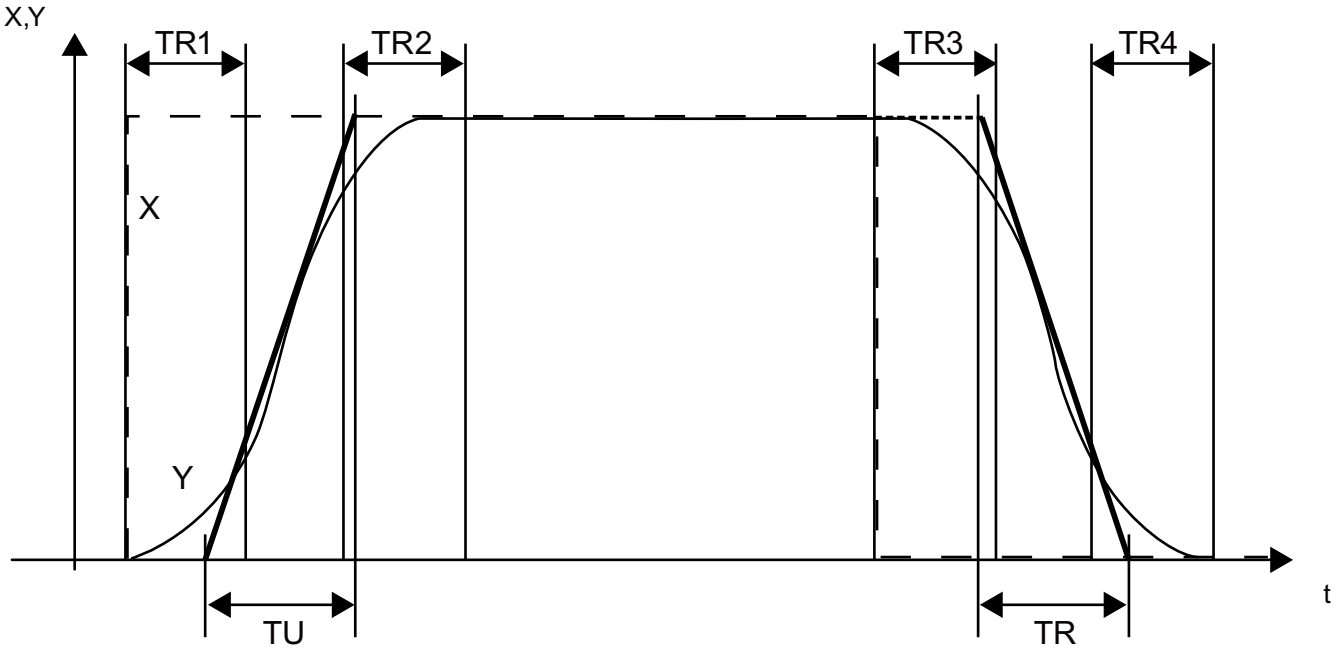
A ramp-down process proceeds analogously.

Transfer function



Ramp-up and ramp-down (not true to scale)

Rounding times if TRU = 00.0 ms and TRD = 0.0 ms:



Ramp-up and ramp-down (not true to scale)

Ramp-up time and ramp-down time

The ramp-up time TU is defined as the time in which the value of the output variable increases proportionally over time by the value NRM.

The ramp-down time TD is defined as the time in which the value of the output variable decreases proportionally over time by the value NRM.

Ramp up time and ramp down time can be selected differently.

Smoothing time during ramp up and ramp down

The smoothing time is defined as the time in which the output variable reaches the maximum acceleration value starting from a constant initial value. During this time, the jerk value is constant and not equal to zero (compare with phase 1).

The smoothing time is also defined as the time in which the output variable reaches a constant final value starting from its maximum acceleration value (compare with phase 3). The smoothing time is defined with TRU or TR1 and TR2 during a ramp-up process and with TRD or TR3 and TR4 during a ramp-down process.

Each time the setpoint changes direction, the system switches from ramp-up to ramp-down or from ramp-down to ramp-up with the associated smoothing processes, depending on the initial position. The same applies accordingly when the ramp-up or ramp-down time is changes during operation.

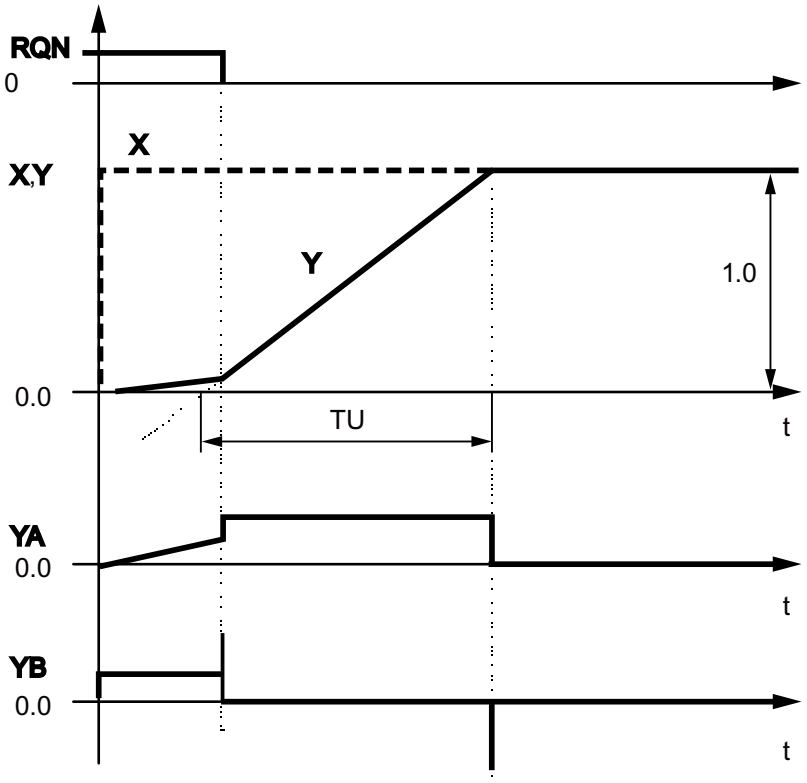
If a ramp-up follows a ramp-down whereby TRD and TD are small and TRU and TU are large, YA is reduced during the ramp-down to the extent that no overshooting occurs in the following ramp-up as long as the target value (X, LL, or LU) and the ramp-function generator times (TU, TD, TRU, TRD) do not change.

If the smoothing (RQN=0) and the correction (ULR=LLR=0) are switched off, then the RGJ block behaves the same as the RGE block.

Enabling smoothing (jerk limitation)

Smoothing is active during ramp up and ramp down when RQN=1.

Transfer function: Switching off smoothing during ramp up



Smoothing is off when RQN = 0. Ramp up/down takes place according to the ramp-up/ramp-down time specified in TU or TD.

When the jerk limiting is switched off during the smoothing time, the remaining ramp up/down also occurs with the ramp-up/ramp-down time specified in TU or TD.

"Smoothing off" mode

If you want to operate the block in this mode, proceed as follows:

- Set the connections TRU, TR1, TR2, TRD, TR3, and TR4 to "0" (all smoothing times are "0").
- Set the RQN connection to "1" ("Smoothing on" mode).

With these settings, the RGJ block behaves as described in the "Smoothing off" mode (RQN = 0).

Operating modes and control of the ramp-function generator

The control inputs are defined as follows:

EN=1	Enable ramp-function generator
S=1	Set output Y to setting value SV; do not integrate

SA=1	Set acceleration YA to setting value ASV; do not integrate
CF=1	Correct output Y to setpoint X with integration.
CU=1	Correct output Y in the direction LU with integration
CD=1	Correct output Y in the direction LL with integration

Truth table(s)

EN	S	SA	CF	CU	CD	Y_{An}	Y_n	Operating mode	Remark
0	*	*	*	*	*	0	0	Inhibit	Y=0
1	0	0	0	0	0	0	Y_{n-1}	Inhibit	Y = constant

*= any value

LL < LU and LL < actual value Y_{n-1} < LU

EN	S	SA	CF	CU	CD	YA_n	Y_n	Operating mode	Remark
1	1	*	*	*	*	Jump	SV_n	Set output to SV	Any SV, fixed or variable
1	0	1	*	*	*	ASV_n	$Y_{n-1}+YA_n$	Set output to integrator 1 on ASV	Any ASV, fixed or variable
1	0	0	1	*	*	TA/ TU(TA/TD)	$Y_{n-1}+YA_n$	Normal mode Y -> X	TU for [X>Y ∧ Y ≥ 0] ∨ [X<Y ∧ Y<=0] TD for [X>Y ∧ Y<0] ∨ [X<Y ∧ Y>0] QE=1 is set when Y=X is reached.
1	0	0	0	1	0	TA/ TU(TA/TD)	$Y_{n-1}+YA_n$	Touch upper limit value Y -> LU	TU, TD as above, depending on the start position QU=1 and QE=1 are set when Y=LU is reached.
1	0	0	0	0	1	TA/ TD(TA/TU)	$Y_{n-1}+YA_n$	Touch lower limit value Y -> LL	TU, TD as above, depending on the start position QL=1 and QE=1 are set when Y=LL is reached.

Correction of the ramp-function generator

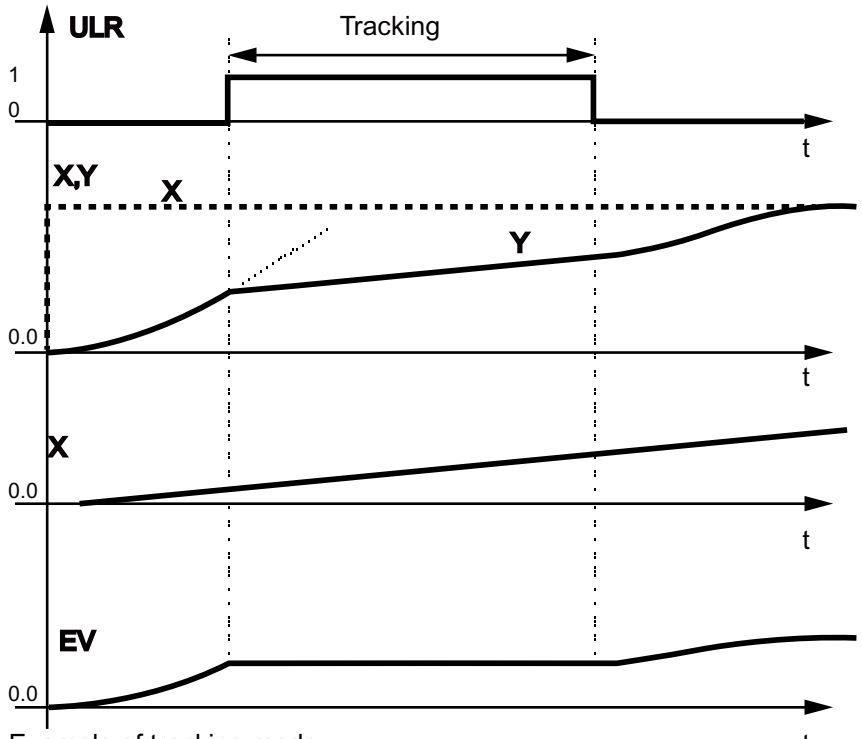
In general, output Y of the ramp-function generator is carried as a setpoint to a lower-level control loop (e.g. speed controller).

If this controller reaches the limit because of a change (e.g. during a ramp-up), the ramp-function generator may not increase the output in accordance to the ramp-up times. In this case, output Y is corrected using the system deviation EV and the weighting factor WD:

$Y_n = Y_{n-1}-EV_n+WD \cdot EV_k$

$_n$= scan interval n

$_k$= the time at which the controller first reaches the limit (0 -> 1 edge on ULR or LLR)



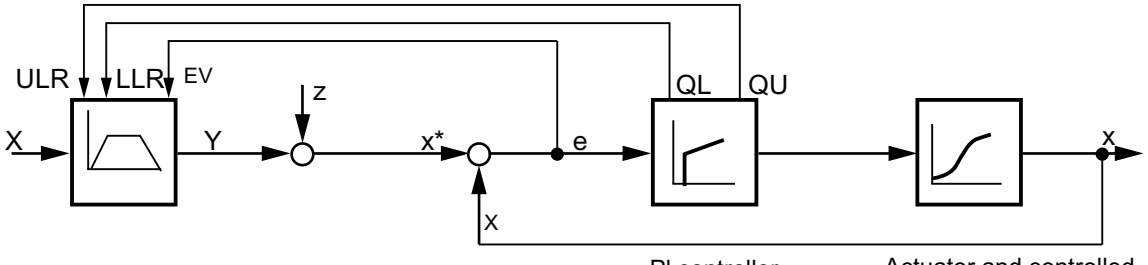
Example of tracking mode

In general, this correction can only be used for "classic control loops" (e.g. PI speed controllers). The controller limits must be set correctly (e.g. equal to the current limits).

Generally, WD is 1.01 to 1.1 (> 1.0!). Jerk limiting is not active during the correction.

The binary outputs of the controller ("Upper/lower limit reached") are returned to the binary inputs ULR or LLR. When the limit is reached, one of the two binary inputs ULR = 1 or LLR = 1 is set via the feedback on the RGJ block, and therefore the correction activated.

If the correction is not to be used, ULR and LLR must be set to 0.



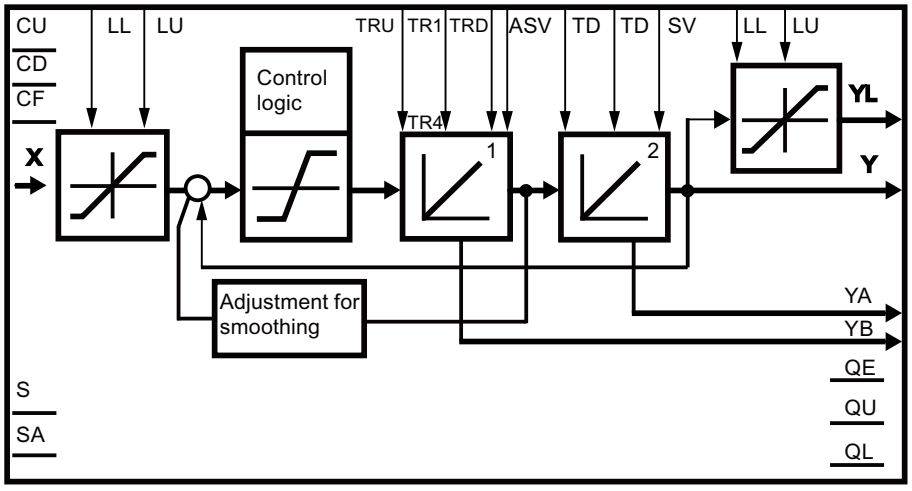
Ramp-function generator

PI controller

Actuator and controlled system

Example configuration of "Tracking"

Block diagram



The input value NRM is set internally to 1.0 when NRM < 1.0e-18.

Block connections

Block connection	Description	Default	Value range	Attrib-utes
X	Input variable	0.0	REAL	
NRM	Normalization, TD/TU	1.0	REAL	
EV	Control error	0.0	REAL	
LU	Upper limit	0.0	REAL	
LL	Lower limit	0.0	REAL	
SV	Setting value, output	0.0	REAL	
ASV	Acceleration setting value	0.0	REAL	
WD	Weighting of the standard deviation	0.0	REAL	
TU	Ramp-up time (ms)	0.0	SDTIME	
TD	Ramp-down time (ms)	0.0	SDTIME	
TRU	Smoothing time during ramp up (ms)	0.0	SDTIME	
TR1	Smoothing time during ramp up (ms)	0.0	SDTIME	
TR2	Smoothing time during ramp up (ms)	0.0	SDTIME	
TRD	Smoothing time during ramp down (ms)	0.0	SDTIME	
TR3	Smoothing time during ramp down (ms)	0.0	SDTIME	
TR4	Smoothing time during ramp down (ms)	0.0	SDTIME	
CU	Higher	0	0/1	
CD	Lower	0	0/1	

Block connection	Description	Default	Value range	Attrib-utes
CF	Output = input	0	0/1	
ULR	Upper limit reached	0	0/1	
LLR	Lower limit reached	0	0/1	
RQN	Smoothing on	0	0/1	
SA	Set acceleration	0	0/1	
S	Set	0	0/1	
EN	Enable	0	0/1	
Y	Output variable	0.0	REAL	
YL	Limited output variable	0.0	REAL	
YA	Acceleration value	0.0	REAL	
YB	Jerk value	0.0	REAL	
QE	Output Y = limited input X	0	0/1	
QU	Upper limit reached	0	0/1	
QL	Lower limit reached	0	0/1	

Configuration data

SIMOTION	✓
SINAMICS	✓
Can be loaded on-line	Yes
Special character-istics	-

8.1.9 Messages and parameters

8.1.9.1 Messages

All objects: DCC, DCC_DC

F51000 DCC: Logon of the run-time group with sampling time management rejected

Message value: -
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: IMMEDIATELY
Cause: The OA application "Drive Control Chart" (DCC) attempted to log on a sampling time that cannot be implemented with the sampling time management of the basic SINAMICS system. The logon was rejected.
Remedy: Try to assign this run-time group another fixed or free run-time group.
The assignment is set in STARTER in the context menu of the DCC chart via sampling times.
Then compile the chart and download it again into the drive unit.

F51001 DCC: No further hardware sampling times available

Message value: %1

Message class: General drive fault (19)

Drive object: All objects

Reaction: NONE

Acknowledge: IMMEDIATELY

Cause: The drive unit can no longer provide any additional hardware sampling times, whose sampling time deviates from the sampling times.already logged on.

Remedy: The fault can be immediately acknowledged, as the system run-time group 0 (corresponds to "Do not calculate") was assigned in p21000[x].

Fault value (r0949, interpret hexadecimal):

yyyyxxxx hex

yyyy: The upper 16 bits of the fault value specify the number of the drive object.

xxxx: The lower 16 bits specify the index of the run-time group in p21000.

Note:

In window "Set run-time groups" in the context menu of the chart, p21000[0] is the topmost entry and p21000[9] the lowest entry.

The current assignment of hardware sampling times can be read-out in r21008.

F51004 DCC: Sampling time of the free run-time group differs at download

Message value: %1

Message class: General drive fault (19)

Drive object: All objects

Reaction: NONE

Acknowledge: IMMEDIATELY

Cause: In the STARTER/SCOUT project that was downloaded, the hardware sampling time of a free run-time group (1 <= p21000[i] <= 256) was set to a value that was either too low or too high. The sampling time must be between 1 ms and the value (r21003 - r21002).

If the sampling time of the selected free run-time group is < 1 ms, the equivalent value of 1 ms is used.

If the value >= r21003, then the sampling time is set to the next higher or the same software sampling time >= r21003. In order to prevent the fault the calculated software sampling time can be set in the run-time group (1001 <= p21000[i] <= 1096).

The free run-time group involved is assigned as a minimum to one block.

If this fault still occurs during download after the selection in p21000[i] in the project has been corrected, please check which run-time group is involved on the basis of the fault value (r0949). Only one F51004 fault is signaled at a time, even if several run-time groups have been incorrectly parameterized in p21000[].

Fault value (r0949, decimal interpretation):

Number of the p21000 index of the run-time group where the sampling time was incorrectly set.

Number of the run-time group = fault value + 1

Note:

With SIMOTION D410, r21003 (unlike all the other Control Units) is automatically set the same as the PROFIBUS sampling time.

Remedy: Correctly set the sampling time of the run-time group or remove all of the blocks from the run-time group.

F51005 DCC: Sampling time of the fixed run-time group differs online

Message value: %1

Message class: General drive fault (19)

Drive object: All objects

Reaction: NONE

Acknowledge: IMMEDIATELY

Cause: Generally, the sampling times of the fixed run-time groups correspond to the sampling times of the associated system function (e.g. the sampling time of the fixed run-time group "BEFORE speed controller" generally corresponds to the sampling of the speed controller p0115[1]).
The sampling time of a system function online was set to a lower value (e.g. with p0112, p0115, p0799, p4099) than the smallest permissible sampling time that is allowed for the fixed run-time group belonging to this system function (1 ms). The sampling time is set to 1 ms. The fixed run-time group involved is assigned as a minimum to one block.
Fault value (r0949, decimal interpretation):
Number of the p21000 index of the run-time group where the sampling time was incorrectly set.
Number of the run-time group = fault value + 1

Remedy: Using parameter p0112 or p0115, increase the sampling time of the system function to the minimum permissible sampling time for the run-time groups of 1 ms or remove all of the blocks from the run-time group.

F51006 DCC: Sampling time of the fixed run-time group differs at download

Message value: %1

Message class: General drive fault (19)

Drive object: All objects

Reaction: NONE

Acknowledge: IMMEDIATELY

Cause: Generally, the sampling times of the fixed run-time groups correspond to the sampling times of the associated system function (e.g. the sampling time of the fixed run-time group "BEFORE speed controller" generally corresponds to the sampling of the speed controller p0115[1]).
During a download, the sampling time of a system function was set to a lower value (p0112, p0115) than the smallest permissible sampling time that is allowed for the fixed run-time group belonging to this system function (1 ms). The sampling time is set to the smallest possible value (r21002 on the drive object).
Fault value (r0949, decimal interpretation):
Number of the p21000 index of the run-time group where the sampling time was incorrectly set.
Number of the run-time group = fault value + 1

Remedy: Using parameter p0112 or p0115, increase the sampling time of the system function to the minimum permissible sampling time for the run-time groups of 1 ms or remove all of the blocks from the run-time group.

F51008 DCC: No NVRAM available

Message value: %1

Message class: General drive fault (19)

Drive object: All objects

Reaction: OFF2

Acknowledge: IMMEDIATELY

Cause: The DCC project contains at least one block that requires remanent memory from the basic SINAMICS system (e.g. SAV, SAV_BY, SAV_D, SAV_I). The request for remanent memory was rejected by the basic SINAMICS system.
Fault value (r0949, decimal interpretation):
0: There is no more free remanent memory available on the drive unit.
1: The EPROM data of the drive unit indicates that there is no remanent memory on the module.

Remedy: For fault value = 0:
- Deactivate other applications on the drive unit that use remanent memory.
- Do not use blocks that require remanent memory in your DCC charts.
For fault value = 1:
- For modules D425 or D435, use hardware version D or higher.
Note:
You can read out the hardware version using SCOUT in online mode under Target system --> Device diagnostics --> tab "General" in the lower window, 3rd column in the line of the CPU.

F51009 DCC: Project data and block library are incompatible

Message value: -

Message class: General drive fault (19)
Drive object: All objects
Reaction: OFF2
Acknowledge: IMMEDIATELY
Cause: The block library and the saved or downloaded project data are incompatible.
Remedy: Make sure that the block library and project data match.
- Update the block library in SINAMICS by downloading the technology package.
or
- Update the project data in the DCC Editor by importing the correct block library.

A51032 DCC: Internal measurement active

Message value: -
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: A Siemens internal measurement has been activated.
Remedy: Carry out a POWER ON (switch off/on) for the Control Unit involved.

F51033 Licensing DCC application not sufficient

Message value: -
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: IMMEDIATELY
Cause: There is a license error in a DCB block.
Remedy: -Obtain the necessary license.
-Later licensing is not possible online via p9920, 9921.

F51034 DCC: block runtimes are not measured

Message value: -
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: IMMEDIATELY
Cause: A block library created with DCB Studio contains blocks for which the runtime has not been measured. Contact the person that created the library.
Remedy:

F51035 DCC: DCC configuration error

Message value: -
Message class: General drive fault (19)
Drive object: All objects
Reaction: OFF2
Acknowledge: IMMEDIATELY
Cause: An error has occurred when powering up from a DCC configuration.

Remedy: - evaluate fault buffer (r0945).
- carry out a POWER ON for all components
(switch-off/switch-on)
- if required, check the data on the non-volatile memory (e.g. memory card).
- upgrade firmware to later version.
- contact the Hotline.

F51050 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51051 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51052 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)

Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51053 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51054 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51055 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects

Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51056 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51057 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51058 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)

Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

F51059 DCC: Fault initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: Infeed: OFF2 (NONE, OFF1)
Servo: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Vector: OFF2 (ENCODER, IASC/DCBRK, NONE, OFF1, OFF3, STOP2)
Chopper: OFF2 (NONE)
Hla: OFF2 (ENCODER, NONE, OFF1, OFF3, STOP2)
Acknowledge: IMMEDIATELY (POWER ON)
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Fault value (r0949, decimal interpretation):
The configured message value is displayed in r0949.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51060 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51061 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE

Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51062 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51063 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51064 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51065 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects

Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51066 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51067 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51068 DCC: alarm initiated by "Drive Control Chart"

Message value: %1
Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

A51069 DCC: alarm initiated by "Drive Control Chart"

Message value: %1

Message class: General drive fault (19)
Drive object: All objects
Reaction: NONE
Acknowledge: NONE
Cause: "Drive Control Chart" (DCC) has initiated this message via the block "Set Message" (STM) or via a block (SINAMICS DCB Extension) generated using SINAMICS DCB Studio.
Alarm value (r2124, interpret decimal):
The configured message value is displayed in r2124.
Remedy: This message was configured with "Drive Control Chart" (DCC).
The cause and remedy depend on the project and should be described in the corresponding project documentation.

8.1.9.2 Parameters

Version: 5200300

All objects: DCC, DCC_DC

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4005	0

Description: Allocates properties to run-time groups 1 to 10.
This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.
The index x + 1 of p21000 corresponds to the number of the run-time group:
- p21000[0] is used to set the property of the run-time group 1
...
- p21000[9] is used to set the property of the run-time group 10

Value:
0: Do not calculate run-time group
1: T = 1 * r21002
2: T = 2 * r21002
3: T = 3 * r21002
4: T = 4 * r21002
5: T = 5 * r21002
6: T = 6 * r21002
7: T = 7 * r21002
8: T = 8 * r21002
9: T = 9 * r21002
10: T = 10 * r21002
11: T = 11 * r21002
12: T = 12 * r21002
13: T = 13 * r21002
14: T = 14 * r21002
15: T = 15 * r21002
16: T = 16 * r21002
17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002

20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002
66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002

69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002
115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002

118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002
164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002

167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002
213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002

216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003
1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003

1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
2000: Read-in AFTER digital inputs
2001: Output BEFORE digital outputs
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4002: Receive AFTER IF2 PZD
4003: Send BEFORE IF2 PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD
4005: Receive AFTER IF2 flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

For value = 4002, 4003, 4005 (IF2 run-time group):

On devices where IF2 does not exist (D4xx, CU310), when selecting the run-time groups that involve IF2, the corresponding run-time group for IF1 is automatically logged on.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4004	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002

17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002

66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002

115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002

164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002

213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003

1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
2000: Read-in AFTER digital inputs
2001: Output BEFORE digital outputs
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

p21000[0...9] Run-time group properties / RTG property

All objects	**Can be changed:** T	**Calculated:** -	**Access level:** 1
	Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	0	4005	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002
17:	T = 17 * r21002
18:	T = 18 * r21002
19:	T = 19 * r21002
20:	T = 20 * r21002

21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002
66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002

70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002
115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002

119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002
164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002

168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002
213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002

217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003
1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003

1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4002: Receive AFTER IF2 PZD
4003: Send BEFORE IF2 PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD
4005: Receive AFTER IF2 flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

For value = 4002, 4003, 4005 (IF2 run-time group):

On devices where IF2 does not exist (D4xx, CU310), when selecting the run-time groups that involve IF2, the corresponding run-time group for IF1 is automatically logged on.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4005	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002

17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002

66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002

115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002

164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002

213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003

1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
3001: BEFORE speed ctrl
3003: BEFORE speed setpoint channel
3004: BEFORE pos ctrl
3005: BEFORE basic positioner
3006: BEFORE standard technology controller
3007: BEFORE act p v
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4002: Receive AFTER IF2 PZD
4003: Send BEFORE IF2 PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD
4005: Receive AFTER IF2 flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION
The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

For value = 4002, 4003, 4005 (IF2 run-time group):

On devices where IF2 does not exist (D4xx, CU310), when selecting the run-time groups that involve IF2, the corresponding run-time group for IF1 is automatically logged on.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4005	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002

17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002

66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002

115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002

164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002

213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003

1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
3001: BEFORE speed ctrl
3003: BEFORE speed setpoint channel
3006: BEFORE standard technology controller
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4002: Receive AFTER IF2 PZD
4003: Send BEFORE IF2 PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD
4005: Receive AFTER IF2 flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

For value = 4002, 4003, 4005 (IF2 run-time group):

On devices where IF2 does not exist (D4xx, CU310), when selecting the run-time groups that involve IF2, the corresponding run-time group for IF1 is automatically logged on.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4005	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0: Do not calculate run-time group
1: T = 1 * r21002
2: T = 2 * r21002
3: T = 3 * r21002
4: T = 4 * r21002
5: T = 5 * r21002
6: T = 6 * r21002
7: T = 7 * r21002
8: T = 8 * r21002
9: T = 9 * r21002
10: T = 10 * r21002
11: T = 11 * r21002
12: T = 12 * r21002
13: T = 13 * r21002
14: T = 14 * r21002
15: T = 15 * r21002
16: T = 16 * r21002

17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002

66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002

115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002

164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002

213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003

1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
3001: BEFORE speed ctrl
3003: BEFORE speed setpoint channel
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4002: Receive AFTER IF2 PZD
4003: Send BEFORE IF2 PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD
4005: Receive AFTER IF2 flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

For value = 4002, 4003, 4005 (IF2 run-time group):

On devices where IF2 does not exist (D4xx, CU310), when selecting the run-time groups that involve IF2, the corresponding run-time group for IF1 is automatically logged on.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4004	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002

17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002

66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002

115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002

164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002

213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003

1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
3003: BEFORE speed setpoint channel
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):
This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:
1 ms <= T_sample < r21003.
At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.
For value > 2000 (fixed run-time group):
The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.
Example:
"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

p21000[0...9] Run-time group properties / RTG property

DCC			
	Can be changed: T	**Calculated:** -	**Access level:** 1
	Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	0	4004	0

Description: Allocates properties to run-time groups 1 to 10.
This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.
The index x + 1 of p21000 corresponds to the number of the run-time group:
- p21000[0] is used to set the property of the run-time group 1
...
- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002
17:	T = 17 * r21002
18:	T = 18 * r21002
19:	T = 19 * r21002
20:	T = 20 * r21002
21:	T = 21 * r21002
22:	T = 22 * r21002
23:	T = 23 * r21002
24:	T = 24 * r21002
25:	T = 25 * r21002
26:	T = 26 * r21002
27:	T = 27 * r21002
28:	T = 28 * r21002
29:	T = 29 * r21002
30:	T = 30 * r21002
31:	T = 31 * r21002
32:	T = 32 * r21002
33:	T = 33 * r21002
34:	T = 34 * r21002
35:	T = 35 * r21002

36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002
66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002

85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002
115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002

134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002
164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002

183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002
213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002

232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003
1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):
This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:
1 ms <= T_sample < r21003.
At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.
For value > 2000 (fixed run-time group):
The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.
Example:
"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

p21000[0...9] Run-time group properties / RTG property

DCC

Can be changed: T	**Calculated:** -	**Access level:** 1
Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4004	0

Description: Allocates properties to run-time groups 1 to 10.
This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.
The index x + 1 of p21000 corresponds to the number of the run-time group:
- p21000[0] is used to set the property of the run-time group 1
...
- p21000[9] is used to set the property of the run-time group 10

Value:
0: Do not calculate run-time group
1: T = 1 * r21002
2: T = 2 * r21002
3: T = 3 * r21002

4: T = 4 * r21002
5: T = 5 * r21002
6: T = 6 * r21002
7: T = 7 * r21002
8: T = 8 * r21002
9: T = 9 * r21002
10: T = 10 * r21002
11: T = 11 * r21002
12: T = 12 * r21002
13: T = 13 * r21002
14: T = 14 * r21002
15: T = 15 * r21002
16: T = 16 * r21002
17: T = 17 * r21002
18: T = 18 * r21002
19: T = 19 * r21002
20: T = 20 * r21002
21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002

53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002
66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002
70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002

102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002
115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002
119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002

151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002
164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002
168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002

200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002
213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002
217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002

249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003
1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003
1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
2000: Read-in AFTER digital inputs
2001: Output BEFORE digital outputs
2002: Read-in AFTER analog inputs
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):

This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:

1 ms <= T_sample < r21003.

At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.

For value > 2000 (fixed run-time group):

The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.

Example:

"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

p21000[0...9] Run-time group properties / RTG property

DCC			
	Can be changed: T	**Calculated:** -	**Access level:** 1
	Data type: Integer16	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	0	4004	0

Description: Allocates properties to run-time groups 1 to 10.

This property comprises the sampling time and, for p21000[x] >= 2000, the instant of the call within the sampling time.

The index x + 1 of p21000 corresponds to the number of the run-time group:

- p21000[0] is used to set the property of the run-time group 1

...

- p21000[9] is used to set the property of the run-time group 10

Value:

0:	Do not calculate run-time group
1:	T = 1 * r21002
2:	T = 2 * r21002
3:	T = 3 * r21002
4:	T = 4 * r21002
5:	T = 5 * r21002
6:	T = 6 * r21002
7:	T = 7 * r21002
8:	T = 8 * r21002
9:	T = 9 * r21002
10:	T = 10 * r21002
11:	T = 11 * r21002
12:	T = 12 * r21002
13:	T = 13 * r21002
14:	T = 14 * r21002
15:	T = 15 * r21002
16:	T = 16 * r21002
17:	T = 17 * r21002
18:	T = 18 * r21002
19:	T = 19 * r21002
20:	T = 20 * r21002

21: T = 21 * r21002
22: T = 22 * r21002
23: T = 23 * r21002
24: T = 24 * r21002
25: T = 25 * r21002
26: T = 26 * r21002
27: T = 27 * r21002
28: T = 28 * r21002
29: T = 29 * r21002
30: T = 30 * r21002
31: T = 31 * r21002
32: T = 32 * r21002
33: T = 33 * r21002
34: T = 34 * r21002
35: T = 35 * r21002
36: T = 36 * r21002
37: T = 37 * r21002
38: T = 38 * r21002
39: T = 39 * r21002
40: T = 40 * r21002
41: T = 41 * r21002
42: T = 42 * r21002
43: T = 43 * r21002
44: T = 44 * r21002
45: T = 45 * r21002
46: T = 46 * r21002
47: T = 47 * r21002
48: T = 48 * r21002
49: T = 49 * r21002
50: T = 50 * r21002
51: T = 51 * r21002
52: T = 52 * r21002
53: T = 53 * r21002
54: T = 54 * r21002
55: T = 55 * r21002
56: T = 56 * r21002
57: T = 57 * r21002
58: T = 58 * r21002
59: T = 59 * r21002
60: T = 60 * r21002
61: T = 61 * r21002
62: T = 62 * r21002
63: T = 63 * r21002
64: T = 64 * r21002
65: T = 65 * r21002
66: T = 66 * r21002
67: T = 67 * r21002
68: T = 68 * r21002
69: T = 69 * r21002

70: T = 70 * r21002
71: T = 71 * r21002
72: T = 72 * r21002
73: T = 73 * r21002
74: T = 74 * r21002
75: T = 75 * r21002
76: T = 76 * r21002
77: T = 77 * r21002
78: T = 78 * r21002
79: T = 79 * r21002
80: T = 80 * r21002
81: T = 81 * r21002
82: T = 82 * r21002
83: T = 83 * r21002
84: T = 84 * r21002
85: T = 85 * r21002
86: T = 86 * r21002
87: T = 87 * r21002
88: T = 88 * r21002
89: T = 89 * r21002
90: T = 90 * r21002
91: T = 91 * r21002
92: T = 92 * r21002
93: T = 93 * r21002
94: T = 94 * r21002
95: T = 95 * r21002
96: T = 96 * r21002
97: T = 97 * r21002
98: T = 98 * r21002
99: T = 99 * r21002
100: T = 100 * r21002
101: T = 101 * r21002
102: T = 102 * r21002
103: T = 103 * r21002
104: T = 104 * r21002
105: T = 105 * r21002
106: T = 106 * r21002
107: T = 107 * r21002
108: T = 108 * r21002
109: T = 109 * r21002
110: T = 110 * r21002
111: T = 111 * r21002
112: T = 112 * r21002
113: T = 113 * r21002
114: T = 114 * r21002
115: T = 115 * r21002
116: T = 116 * r21002
117: T = 117 * r21002
118: T = 118 * r21002

119: T = 119 * r21002
120: T = 120 * r21002
121: T = 121 * r21002
122: T = 122 * r21002
123: T = 123 * r21002
124: T = 124 * r21002
125: T = 125 * r21002
126: T = 126 * r21002
127: T = 127 * r21002
128: T = 128 * r21002
129: T = 129 * r21002
130: T = 130 * r21002
131: T = 131 * r21002
132: T = 132 * r21002
133: T = 133 * r21002
134: T = 134 * r21002
135: T = 135 * r21002
136: T = 136 * r21002
137: T = 137 * r21002
138: T = 138 * r21002
139: T = 139 * r21002
140: T = 140 * r21002
141: T = 141 * r21002
142: T = 142 * r21002
143: T = 143 * r21002
144: T = 144 * r21002
145: T = 145 * r21002
146: T = 146 * r21002
147: T = 147 * r21002
148: T = 148 * r21002
149: T = 149 * r21002
150: T = 150 * r21002
151: T = 151 * r21002
152: T = 152 * r21002
153: T = 153 * r21002
154: T = 154 * r21002
155: T = 155 * r21002
156: T = 156 * r21002
157: T = 157 * r21002
158: T = 158 * r21002
159: T = 159 * r21002
160: T = 160 * r21002
161: T = 161 * r21002
162: T = 162 * r21002
163: T = 163 * r21002
164: T = 164 * r21002
165: T = 165 * r21002
166: T = 166 * r21002
167: T = 167 * r21002

168: T = 168 * r21002
169: T = 169 * r21002
170: T = 170 * r21002
171: T = 171 * r21002
172: T = 172 * r21002
173: T = 173 * r21002
174: T = 174 * r21002
175: T = 175 * r21002
176: T = 176 * r21002
177: T = 177 * r21002
178: T = 178 * r21002
179: T = 179 * r21002
180: T = 180 * r21002
181: T = 181 * r21002
182: T = 182 * r21002
183: T = 183 * r21002
184: T = 184 * r21002
185: T = 185 * r21002
186: T = 186 * r21002
187: T = 187 * r21002
188: T = 188 * r21002
189: T = 189 * r21002
190: T = 190 * r21002
191: T = 191 * r21002
192: T = 192 * r21002
193: T = 193 * r21002
194: T = 194 * r21002
195: T = 195 * r21002
196: T = 196 * r21002
197: T = 197 * r21002
198: T = 198 * r21002
199: T = 199 * r21002
200: T = 200 * r21002
201: T = 201 * r21002
202: T = 202 * r21002
203: T = 203 * r21002
204: T = 204 * r21002
205: T = 205 * r21002
206: T = 206 * r21002
207: T = 207 * r21002
208: T = 208 * r21002
209: T = 209 * r21002
210: T = 210 * r21002
211: T = 211 * r21002
212: T = 212 * r21002
213: T = 213 * r21002
214: T = 214 * r21002
215: T = 215 * r21002
216: T = 216 * r21002

217: T = 217 * r21002
218: T = 218 * r21002
219: T = 219 * r21002
220: T = 220 * r21002
221: T = 221 * r21002
222: T = 222 * r21002
223: T = 223 * r21002
224: T = 224 * r21002
225: T = 225 * r21002
226: T = 226 * r21002
227: T = 227 * r21002
228: T = 228 * r21002
229: T = 229 * r21002
230: T = 230 * r21002
231: T = 231 * r21002
232: T = 232 * r21002
233: T = 233 * r21002
234: T = 234 * r21002
235: T = 235 * r21002
236: T = 236 * r21002
237: T = 237 * r21002
238: T = 238 * r21002
239: T = 239 * r21002
240: T = 240 * r21002
241: T = 241 * r21002
242: T = 242 * r21002
243: T = 243 * r21002
244: T = 244 * r21002
245: T = 245 * r21002
246: T = 246 * r21002
247: T = 247 * r21002
248: T = 248 * r21002
249: T = 249 * r21002
250: T = 250 * r21002
251: T = 251 * r21002
252: T = 252 * r21002
253: T = 253 * r21002
254: T = 254 * r21002
255: T = 255 * r21002
256: T = 256 * r21002
1001: T = 1 * r21003
1002: T = 2 * r21003
1003: T = 3 * r21003
1004: T = 4 * r21003
1005: T = 5 * r21003
1006: T = 6 * r21003
1008: T = 8 * r21003
1010: T = 10 * r21003
1012: T = 12 * r21003

1016: T = 16 * r21003
1020: T = 20 * r21003
1024: T = 24 * r21003
1032: T = 32 * r21003
1040: T = 40 * r21003
1048: T = 48 * r21003
1064: T = 64 * r21003
1080: T = 80 * r21003
1096: T = 96 * r21003
2000: Read-in AFTER digital inputs
2001: Output BEFORE digital outputs
2002: Read-in AFTER analog inputs
2003: Output BEFORE analog outputs
4000: Receive AFTER IF1 PROFIdrive PZD
4001: Send BEFORE IF1 PROFIdrive PZD
4004: Receive AFTER IF1 PROFIdrive flexible PZD

Recommendation: On the drive objects of CU, TB30, TM15DI_DO, TM31, TM41, TM120, the sampling time for supplementary functions is preset to p0115[0] = 4 ms. If you wish to configure a DCC run-time group with a shorter sampling time on these drive objects, then you should first set the sampling time for supplementary functions p0115[0] on this drive object to the value of the shortest sampling time required.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: r7903, r21008

⚠ CAUTION

The properties of the run-time groups must not be changed during operation as this could result in discontinuous signal transitions.

Note

For value = 1 ... 256 (free run-time group):
This selection value can only be selected online if the following applies for sampling time T_sample of this run-time group:
1 ms <= T_sample < r21003.
At download, a value that violates this condition is not rejected, but a permissible equivalent value is set automatically and fault F51004 is output.
For value > 2000 (fixed run-time group):
The fixed run-time groups p21000[x] >= 2000 log on with the sampling time of the associated basic system function, subject to a minimum sampling time of 1 ms. If, as a result of this limit, the actual sampling time deviates from the sampling time of the basic system function, then fault F51005 (during F51006 download) is output. In this case, another run-time group with a sampling time >= 1 ms should be selected. When selecting the fixed run-time groups, a check is not made as to whether the associated system block exists.
Example:
"BEFORE speed setpoint channel" means before function charts 3010, 3020, 3030, 3040, etc. are calculated, if the setpoint channel is activated. If, e.g. for SERVO, a setpoint channel has not been configured (p0108.8 = 0), the calculation is made before function chart 3095.

r21001[0...9] **Run-time group sampling time / RTG sampling time**

All objects

Can be changed: -	**Calculated:** -	**Access level:** 1
Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
- [ms]	- [ms]	- [ms]

Description: Displays the current sampling time of the run-time groups.

Index:
[0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

r21002 **Basis sampling time, hardware / Basis samp time HW**

All objects

Can be changed: -	**Calculated:** -	**Access level:** 1
Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
- [ms]	- [ms]	- [ms]

Description: Displays the basis sampling time effective at this drive object for values 1 to 256 of p21000.
Sampling time T = p21000 * r21002

r21003 **Basis sampling time, software / Basis samp time SW**

All objects

Can be changed: -	**Calculated:** -	**Access level:** 1
Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
- [ms]	- [ms]	- [ms]

Description: Displays the basis sampling time effective at this drive object for p21000 = 1002 to 1096 as factor.
Sampling time T = (p21000 - 1000) * r21003

Dependency: Ensure that the basis sampling time on the SIMOTION D410 for the software time slices is always the same as the configured PROFIBUS/PROFINET clock cycle.

r21005[0...9] **Computing time load of the run-time group / RTG load**

All objects

Can be changed: -	**Calculated:** -	**Access level:** 3
Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
- [%]	- [%]	- [%]

Description: Share of the computing time load with which the DCC run-time group contributes to the utilization of the sampling time during which it is called.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Note
The computing time load can only be displayed for the run-time groups which are logged on (p21000[x] > 0). The value for the computing time load is calculated in the drive unit based on the project loaded plus DCC chart. Therefore, the values r21005[x] are not available in the offline mode of the SCOUT/STARTER.

In r21005 the computing time load is displayed, with which the DCC runtime group utilizes the sampling time in which it is called. The runtime groups "Receive AFTER IF1 PROFIdrive PZD" (p21000 = 4000), "Send BEFORE IF1 PROFIdrive PZD" (p21000 = 4001), "Receive BEFORE IF2 PZD" (p21000 = 4002) and "Send BEFORE IF2 PZD" (p21000 = 4003) are called in the isochronous mode and in the non-isochronous mode, in different sampling times.
In the non-isochronous mode, these are IF1 / IF2 PZD sampling time (p2048 for p21000 = 4000 or 4001, p8848 for p21000 = 4002 or 4003). In the isochronous mode, this is the current controller sampling time (p115[0]) which is periodically called with the isochronous bus cycle time. The computing time load displayed in r21005 is always calculated for the (more unfavorable) case of isochronous operation. This is why this value does not (always) act to the full amount on the computing time load of the complete system.

r21008[0...31] Hardware sampling times available / HW t_samp

All objects

Can be changed: -	**Calculated:** -	**Access level:** 3
Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
- [ms]	- [ms]	- [ms]

Description: Displays the assignment of the available hardware sampling times of the drive unit.
The designated sampling times are those created as a multiple of the hardware basis sampling time (r21002) and which are always < r21003.

Index:
[0] = Hardware 1
[1] = Hardware 2
[2] = Hardware 3
[3] = Hardware 4
[4] = Hardware 5
[5] = Hardware 6
[6] = Hardware 7
[7] = Hardware 8
[8] = Hardware 9
[9] = Hardware 10
[10] = Hardware 11
[11] = Hardware 12
[12] = Hardware 13
[13] = Hardware 14
[14] = Hardware 15
[15] = Hardware 16
[16] = Hardware 17
[17] = Hardware 18
[18] = Hardware 19
[19] = Hardware 20
[20] = Hardware 21
[21] = Hardware 22
[22] = Hardware 23
[23] = Hardware 24
[24] = Hardware 25
[25] = Hardware 26
[26] = Hardware 27
[27] = Hardware 28
[28] = Hardware 29
[29] = Hardware 30
[30] = Hardware 31
[31] = Hardware 32

Dependency:
See also: r7903, p21000
See also: F51001

NOTICE
For internal purposes, the drive unit always requires several free hardware sampling times. Therefore the current number of free hardware sampling times can be read out in r7903. If r7903=0, no additional sampling time different from r21008[0...31] may be provided from the Control Unit. When selecting in this state, if a run-time group with a sampling time < r21003 (p21000 <= 255) is selected in p21000, only run-time groups whose sampling time is already provided in r21008[0...31] may be selected.

Note
A sampling time that is provided can be simultaneously used by system functions, several FBLOCK run-time groups and several DCC run-time groups.
The sampling time of run-time groups that have been assigned to the PROFIBUS run-time groups (p21000 = 4000 ... 4004) is not displayed in r21008. For this sampling time, one of the internally and permanently assigned hardware sampling times is used.
If the value of r21008[x] != 0 (not equal to 0), then the sampling time is specified in ms.
If the value of r21008[x] = 0, this sampling time can still be freely assigned. It should be noted that the basic system requires several freely assignable hardware sampling times for internal functions. The number of hardware sampling times that can still be freely assigned can be read out in r7903.
If the value r21008[x] = 99999.00000, this hardware sampling time is not supported.

p21030 Run-time group, computing time measurement / RTG comp_ti_meas

All objects

Can be changed: T, U	**Calculated:** -	**Access level:** 4
Data type: Unsigned16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	65535	0

Description: Only for internal Siemens service purposes.

Dependency: See also: p21032, r21035, r21036, r21037

p21031 Computing time measurement, blocks / Comp_ti_meas block

All objects

Can be changed: T, U	**Calculated:** -	**Access level:** 4
Data type: Unsigned32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
0	4294967295	0

Description: Sets the number of the run-time group to be measured 1 ... 10.
This starts measurement of the entire run-time group and the individual blocks.
At the end of the measurement the parameter is reset automatically to 0.

⚠ DANGER

During this measurement, the run-time group n is logged off and on again twice for the purpose of time slice management. Since this procedure runs in a background task, the run-time group is not calculated for an indefinite period on two occasions. This can lead to freezing or jumps in signal values. Logical signals can assume unexpected values. This measurement must therefore never be carried out when an application is running!

Note
The run-time group to be measured has to be logged on.
Only one measurement can take place at a time.
The results are displayed in parameters r21035, r21036, r21037.

p21032 Computing time measurement, duration / Comp_ti_meas dur

All objects

Can be changed: T, U	**Calculated:** -	**Access level:** 4
Data type: Unsigned16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
60 [s]	10000 [s]	60 [s]

Description: Only for internal Siemens service purposes.

Dependency: See also: p21030, r21035, r21036, r21037

p21033 Computing time measurement, number of individual measurements / Comp_ti_meas qty

All objects

Can be changed: T, U	**Calculated:** -	**Access level:** 4
Data type: Unsigned32	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
1	4294967295	10000

Description: Setting for the number of calls during the measurement of the individual blocks.

Dependency: See also: p21031

r21035[0...9] **Computing time, minimum value / Computing time min**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Only for internal Siemens service purposes.

Index:
[0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: p21030, p21032, r21036, r21037

r21036[0...9] **Computing time, mean value / Computing tim av**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Only for internal Siemens service purposes.

Index:
[0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

r21037[0...9] **Computing time, maximum value / Computing time max**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Only for internal Siemens service purposes.

Index: [0] = Run-time group 1
[1] = Run-time group 2
[2] = Run-time group 3
[3] = Run-time group 4
[4] = Run-time group 5
[5] = Run-time group 6
[6] = Run-time group 7
[7] = Run-time group 8
[8] = Run-time group 9
[9] = Run-time group 10

Dependency: See also: p21030, p21032, r21035, r21036

r21041[0...49] Block ID of the measured block / Block ID

All objects

Can be changed: -	**Calculated:** -	**Access level:** 4
Data type: Unsigned16	**Dynamic index:** -	**Function diagram:** -
P-Group: -	**Unit group:** -	**Unit selection:** -
Not for motor type: -	**Scaling:** -	**Expert list:** 1
Min:	**Max:**	**Factory setting:**
-	-	-

Description: Block ID of the measured block (block run-time measurement via parameter p21031. The blocks are measured in the same sequence as they have been programmed in the execution sequence.
The parameter is designed for the measurement of 50 block instances

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

r21042[0...49] **First run / subsequent run identifiers / First**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: Unsigned16	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	-	-	-

Description: In the block run-time measurements, the block run-times are measured. R21039 indicates whether the measurement is the first or a subsequent call.

If the block type occurs only once in the run-time group, only the measured value for the first run will be supplied.

The parameter is designed for the measurement of 50 block instances

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

r21043[0...49] **Minimum measured block run-time in us / Computing time min**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Minimum measured run-time of the measured block (block run-time measurement via parameter p21031. The blocks are measured in the same sequence as they have been programmed in the execution sequence.

The parameter is designed for the measurement of 50 block instances

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

r21044[0...49] **Average measured block run-time in us / Computing tim av**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Average measured run-time of the measured block (block run-time measurement via parameter p21031. The blocks are measured in the same sequence as they have been programmed in the execution sequence.

The parameter is designed for the measurement of 50 block instances

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

r21045[0...49] **Maximum measured block run-time in us / Computing time max**

All objects	**Can be changed:** -	**Calculated:** -	**Access level:** 4
	Data type: FloatingPoint32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	- [µs]	- [µs]	- [µs]

Description: Average measured run-time of the measured block (block run-time measurement via parameter p21031. The blocks are measured in the same sequence as they have been programmed in the execution sequence.

The parameter is designed for the measurement of 50 block instances

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

r21046[0...49] **Library IDs of the measured blocks / Lib ID measured**

All objects			
	Can be changed: -	**Calculated:** -	**Access level:** 4
	Data type: Unsigned32	**Dynamic index:** -	**Function diagram:** -
	P-Group: -	**Unit group:** -	**Unit selection:** -
	Not for motor type: -	**Scaling:** -	**Expert list:** 1
	Min:	**Max:**	**Factory setting:**
	-	-	-

Description:
Library ID of the measured block (block run-time measurement via parameter p21031).
Measurements with blocks from different libraries can thereby be carried out in one run-time group.
The blocks are measured in the same sequence as they have been programmed in the execution sequence.
The parameter is designed for the measurement of 50 block instances
Indices 0..49

Index:
[0] = Block 1
[1] = Block 2
[2] = Block 3
[3] = Block 4
[4] = Block 5
[5] = Block 6
[6] = Block 7
[7] = Block 8
[8] = Block 9
[9] = Block 10
[10] = Block 11
[11] = Block 12
[12] = Block 13
[13] = Block 14
[14] = Block 15
[15] = Block 16
[16] = Block 17
[17] = Block 18
[18] = Block 19
[19] = Block 20
[20] = Block 21
[21] = Block 22
[22] = Block 23
[23] = Block 24
[24] = Block 25
[25] = Block 26
[26] = Block 27
[27] = Block 28
[28] = Block 29
[29] = Block 30
[30] = Block 31
[31] = Block 32
[32] = Block 33
[33] = Block 34
[34] = Block 35
[35] = Block 36
[36] = Block 37
[37] = Block 38
[38] = Block 39
[39] = Block 40
[40] = Block 41
[41] = Block 42
[42] = Block 43
[43] = Block 44
[44] = Block 45
[45] = Block 46
[46] = Block 47
[47] = Block 48
[48] = Block 49
[49] = Block 50

8.1.10 Appendix

8.1.10.1 Data types

The table lists the data types relevant for the DCBLIB.

Table 8-4 Overview: Data types of the block connections

Abbreviation	Data width	Data type according to IEC 61131-3	Postfix for DCB identifier	PIN designator ○ Input ○ Output	Can be connected with data type	Description
BO/B	1 bit	BOOL	_B*	I, I1, I2,.... Q, Q1, Q2;....	BOOL	Bool
BY	8 bit	BYTE	_BY	IS QS	BY, SINT, USINT	Bit string
W	16 bit	WORD	_W		WORD, INT, UINT	Bit string
DW	32 bit	DWORD	_DW		DWORD, DINT, UDINT	Bit string
SI	8 bit	SINT	_SI	X, X1, X2, ... Y, Y1, Y2, ...	SINT, USINT, BY	Signed Short Integer
I	16 bit	INT	_I		INT, UINT, WORD	Signed Integer
DI/D	32 bit	DINT	_D		DINT, UDINT, DWORD	Signed Double Integer
US	8 bit	USINT	_US		SINT, USINT, BY	Unsigned Short Integer
UI	16 bit	UINT	_UI		INT, UINT, WORD	Unsigned Integer
UD	32 bit	UDINT	_UD		DINT, UDINT, DWORD	Unsigned Double Integer
R	32 bit	REAL	_R*		REAL, SDTIME	Floating Point Single Precision according to IEEE 754
LR	64 bit	LREAL	_LR		LREAL	Floating Point Double Precision according to IEEE 754
TS	32 bit	(SDTIME)	-	-	SDTIME, REAL	The SDTIME data type is derived from the REAL data type, 1.0 corresponds to 1.0 ms. Negative values are not defined.
AID	32 bit	-	-	-	DINT, UDINT, DWORD	Alarm ID
*	-	Block-defined	-	-	See DCC editor description	See DCC editor description

The table lists the fields for DPV1 parameter job and response.

Table 8-5 Overview: Fields in DPV1 parameter job and response

Field	Data type	Values	Remark
Job reference	Unsigned8	0x01 to 0xFF	
	Unique identification of the job/response pair for the master. The master changes the job reference with each new job. The slave mirrors the job reference in its response.		
Job identifier	Unsigned8	0x010x02	Read job Write job
	Specifies the type of job. In the case of a write job, the changes are made in the volatile memory (RAM). A save operation is needed in order to transfer the modified data to the non-volatile memory (p0971, p0977).		
Response ID	Unsigned8	0x010x020x810x82	Read job (+) write job (+) read job (-) write job (-)
	Mirrors the job identifier and specifies whether job execution was positive or negative. Negative means: The job could not be carried out totally or partly. For each subresponse, the error values are transferred instead of the values.		
Drive object number	Unsigned8	0x01 to 0x27	Number 1 ... 39 Limited by DPV1 telegram length
	Defines the number of adjoining areas for the parameter address and/or parameter value for multi-parameter jobs. The number of parameters = 1 for single jobs.		
Attribute	Unsigned8	0x100x200x30	Value Description Text (not implemented)
	Type of parameter element accessed		
Number of elements	Unsigned8	0x000x01 ... 0x75	Special function Number 1 ... 117 Limited by DPV1 telegram length
	Number of array elements accessed		
Parameter number	Unsigned16	0x0001 to 0xFFFF	No. 1 to 65535
	Addresses the parameter accessed		
Subindex	Unsigned16	0x0000 to 0xFFFF	No. 0 to 65535
	Addresses the first array element of the parameter accessed		

Field	Data type	Values	Remark
Format	Unsigned8	0x020x030x040x050x060x070x08 Other values 0x400x410x420x430x44	Integer8 data type Integer16 data type Integer32 data type Unsigned8 data type Unsigned16 data type Unsigned32 data type Floating-point data type See PROFIdrive PROFILE v3.1 Zero (without values as a positive subresponse of a write request) Byte Word Double Word Error
	The format and number specify the adjoining space containing values in the telegram. For write access, it is preferable to specify data types according to the PROFIdrive profile. Bytes, words, and double words are also possible as a substitute.		
Number of values	Unsigned8	0x00 to 0xEA	Number 0 ... 234 Limited by DPV1 telegram length
	Specifies the number of subsequent values.		
Error values	Unsigned16	0x0000 to 0x00FF	Meaning of the error values, see Appendix A.2
	The error values in the event of a negative response. If the values make up an odd number of bytes, a zero byte is attached. This ensures the integrity of the word structure of the telegram.		
Values	Unsigned16	0x0000 to 0x00FF	
	The values of the parameter for read or write access. If the values make up an odd number of bytes, a zero byte is attached. This ensures the integrity of the word structure of the telegram.		

8.1.10.2 Error values in PROFIdrive parameter responses, data types

Table 8-6 Error values in DPV1 parameter responses

Error value	Meaning	Remark	Additional info
0x00	Illegal parameter number.	Access to a parameter that does not exist.	-
0x01	Parameter value cannot be changed.	Modification access to a parameter value that cannot be changed.	Subindex
0x02	Lower or upper value limit exceeded.	Modification access with value outside value limits.	Subindex
0x03	Invalid subindex.	Access to a subindex that does not exist.	Subindex
0x04	No array.	Access with subindex to an unindexed parameter.	-
0x05	Wrong data type.	Modification access with a value that does not match the data type of the parameter.	-

Error value	Meaning	Remark	Additional info
0x06	Illegal set operation (only reset permitted).	Modification access with a value not equal to 0 in a case where this is not permitted.	Subindex
0x07	Description element cannot be changed.	Modification access to a description element that cannot be changed.	Subindex
0x09	No description data available	Access to a description that does not exist (the parameter value exists).	-
0x0B	No parameter change rights.	Modification access with no parameter change rights.	-
0x0F	No text array available.	Access to a text array that does not exist (the parameter value exists).	-
0x11	Request cannot be executed due to operating status.	Access is temporarily not possible for unspecified reasons.	-
0x14	Illegal value.	Modification access with a value that is within the limits but is illegal for other permanent reasons (parameter with defined individual values).	Subindex
0x15	Response too long.	The length of the present response exceeds the maximum transfer length.	-
0x16	Illegal parameter address.	Illegal or unsupported value for attribute, number of elements, parameter number, subindex or a combination of these.	-
0x17	Illegal format.	Write request: illegal or unsupported parameter data format.	-
0x18	Number of values inconsistent.	Write request: a mismatch exists between the number of values in the parameter data and the number of elements in the parameter address.	-
0x19	Drive object does not exist.	You have attempted to access a drive object that does not exist.	-
0x20	The text element of the parameter cannot be changed.	-	-
0x21	BMP service is not supported; invalid request ID.	-	-
0x22	Multi-parameter accessing is not supported.	-	-
0x65	Parameter presently deactivated.	Access to a parameter that, although available, is currently inactive (e.g. n control set and access to parameter from V/f control).	-
0x6B	Parameter %s [%s]: No write access for the enabled controller.	-	-
0x6C	Parameter %s [%s]: Unit unknown.	-	-
0x6D	Parameter %s [%s]: Write access only in the commissioning state, encoder (p0010 = 4).	-	-
0x6E	Parameter %s [%s]: Write access only in the commissioning state, motor (p0010 = 3).	-	-
0x6F	Parameter %s [%s]: Write access only in the commissioning state, power unit (p0010 = 2).	-	-
0x70	Parameter %s [%s]: Write access only in the quick commissioning mode (p0010 = 1).	-	-

Error value	Meaning	Remark	Additional info
0x71	Parameter %s [%s]: Write access only in the ready mode (p0010 = 0).	-	-
0x72	Parameter %s [%s]: Write access only in the commissioning state, parameter reset (p0010 = 30).	-	-
0x73	Parameter %s [%s]: Write access only in the commissioning state, Safety (p0010 = 95).	-	-
0x74	Parameter %s [%s]: Write access only in the commissioning state, tech. application/units (p0010 = 5).	-	-
0x75	Parameter %s [%s]: Write access only in the commissioning state (p0010 not equal to 0).	-	-
0x76	Parameter %s [%s]: Write access only in the commissioning state, download (p0010 = 29).	-	-
0x77	Parameter %s [%s] must not be written during download.	-	-
0x78	Parameter %s [%s]: Write access only in the commissioning state, drive configuration (device: p0009 = 3).	-	-
0x79	Parameter %s [%s]: Write access only in the commissioning state, define drive type (device: p0009 = 2).	-	-
0x7A	Parameter %s [%s]: Write access only in the commissioning state, data set basis configuration (device: p0009 = 4).	-	-
0x7B	Parameter %s [%s]: Write access only in the commissioning state, device configuration (device: p0009 = 1).	-	-
0x7C	Parameter %s [%s]: Write access only in the commissioning state, device download (device: p0009 = 29).	-	-
0x7D	Parameter %s [%s]: Write access only in the commissioning state, device parameter reset (device: p0009 = 30).	-	-
0x7E	Parameter %s [%s]: Write access only in the commissioning state, device ready (device: p0009 = 0).	-	-
0x7F	Parameter %s [%s]: Write access only in the commissioning state, device (device: p0009 not equal to 0).	-	-
0x81	Parameter %s [%s] must not be written during download.	-	-
0x82	Transfer of the control authority (master) is inhibited by BI: p0806.	-	-
0x83	Parameter %s [%s]: Requested BICO interconnection not possible.	BICO output does not supply float values. The BICO input, however, requires a float value.	-
0x84	Parameter %s [%s]: Parameter change inhibited (refer to p0300, p0400, p0922).	-	-
0x85	Parameter %s [%s]: Access method not defined.	-	-

Error value	Meaning	Remark	Additional info
0xC8	Below currently valid limit.	Modification request for a value that, although within "absolute" limits, is below the currently valid lower limit.	-
0xC9	Above currently valid limit.	Modification request for a value that, although within "absolute" limits, is above the currently valid lower limit (e.g. governed by the current converter rating).	-
0xCC	Write access not permitted.	Write access is not permitted because an access code is not available.	-
0xFF	Successful read/write operation.	The value has been successfully read or written.	-

8.1.10.3 Block overview

The version as of which they are available must be specified for new blocks.

Block	Description	SIMOTION	SINAMICS
ACOS	Arc cosine function	x	
ADD	Adder (REAL type)	x	x
ADD_D	Adder (double integer type)	x	x
ADD_I	Adder (integer type)	x	x
ADD_M	Modulo adder for addition in correct axis cycle	x	x
AND	Logic AND operation (BOOL type)	x	x
AND_W	Logic AND operation (WORD type)	x	
ASIN	Arc sine function	x	
ATAN	Arc tangent function	x	
AVA	Absolute value generator, with sign evaluation	x	x
AVA_D	Absolute value generator (double integer)	x	x
BF	Flashing function (BOOL type)	x	x
BF_W	Flashing function for status word (WORD type)	x	
BSW	Binary change-over switch (BOOL type)	x	x
BY_B	Status byte to eight binary variables converter	x	
BY_W	Status byte to status word converter	x	x
B_BY	Eight binary variables to status byte converter	x	
B_DW	32 binary variables to status double word converter	x	x
B_W	16 binary variables to status word converter	x	x
CNM	Controllable numeric memory (REAL type)	x	x
CNM_D	Controllable numeric memory (DOUBLE INTEGER type)	x	x
CNM_I	Controllable numeric memory (INTEGER type)	x	x
COS	Cosine function	x	V4.4
CTD	Time difference determination from an internal time stamp	x	
CTR	Counter (BOOL type)	x	x
DCA	Diameter calculator	x	x
DEL	Dead zone element	x	x

Block	Description	SIMOTION	SINAMICS
DEZ	Dead zone element	x	x
DFR	Reset-dominant D-type flip-flop (BOOL type)	x	x
DFR_W	Reset-dominant D-type flip-flop (WORD type)	x	
DIF	Derivative action element	x	x
DIV	Divider (REAL type)	x	x
DIV_D	Divider (double integer type)	x	x
DIV_I	Divider (integer type)	x	x
DLB	Delay element (REAL type)	x	x
DT1	Smoothing element	x	x
DW_B	Status double word to 32 binary variables converter	x	x
DW_R	Acceptance of bit string as REAL value	x	x
DW_W	Status double word to status word converter	x	x
DX8	Demultiplexer, eight outputs, cascadable (REAL type)	x	x
DX8_D	Demultiplexer, eight outputs, cascadable (DOUBLE INTEGER type)	x	x
DX8_I	Demultiplexer, eight outputs, cascadable (INTEGER type)	x	x
D_I	DOUBLE INTEGER to INTEGER converter	x	x
D_R	DOUBLE INTEGER to REAL converter	x	x
D_SI	DOUBLE INTEGER to SHORT INTEGER converter	x	
D_UI	DOUBLE INTEGER to UNSIGNED INTEGER converter	x	x
D_US	DOUBLE INTEGER to UNSIGNED SHORT INTEGER converter	x	x
ETE	Edge evaluator (BOOL type)	x	x
GTS	Reads out a time stamp	x	
INCO	Axial winder moment of inertia	x	x
INT	Integrator	x	x
I_D	INTEGER to DOUBLE INTEGER converter	x	x
I_R	INTEGER to REAL converter	x	x
I_SI	INTEGER to SHORT INTEGER converter	x	
I_UD	INTEGER to UNSIGNED DOUBLE INTEGER converter	x	x
I_US	INTEGER to UNSIGNED SHORT INTEGER converter	x	x
LIM	Limiter (REAL type)	x	x
LIM_D	Limiter (DOUBLE INTEGER type)	x	x
LR_R	LONG REAL to REAL converter	x	
LVM	Double-sided limit monitor with hysteresis (BOOL type)	x	x
MAS	Maximum evaluator	x	x
MFP	Pulse generator (BOOL type)	x	x
MIS	Minimum evaluator	x	x
MUL	Multiplier (REAL type)	x	x
MUL_D	Multiplier (double integer type)	x	x
MUL_I	Multiplier (integer type)	x	x
MUX8	Multiplexer, cascadable (REAL type)	x	x
MUX8_D	Multiplexer, cascadable (DOUBLE-INTEGER type)	x	x
MUX8_I	Multiplexer, cascadable (INTEGER type)	x	x
MVS	Sliding-type mean value generator	x	x

Block	Description	SIMOTION	SINAMICS
N2_R	16-bit fixed-point format (N2) to REAL converter	x	x
N4_R	32-bit fixed-point format (N4) to REAL converter	x	x
NAND	Logic AND operation (BOOL type)	x	x
NCM	Numeric comparator (REAL type)	x	x
NCM_D	Numeric comparator (DOUBLE INTEGER type)	x	x
NCM_I	Numeric comparator (INTEGER type)	x	x
NOP1	Dummy block (REAL type)	x	x
NOP1_B	Dummy block (BOOL type)	x	x
NOP1_D	Dummy block (DOUBLE INTEGER type)	x	x
NOP1_I	Dummy block (INT type)	x	x
NOP8	Dummy block (REAL type)	x	x
NOP8_B	Dummy block (BOOL type)	x	x
NOP8_D	Dummy block (DOUBLE INTEGER type)	x	x
NOP8_I	Dummy block (INTEGER type)	x	x
NOR	Logic OR operation (BOOL type)	x	x
NOT	Inverter (BOOL type)	x	x
NOT_W	Status word inverter (WORD type)	x	
NSW	Numeric change-over switch (REAL type)	x	x
NSW_D	Numeric change-over switch (DOUBLE INTEGER type)	x	x
NSW_I	Numeric change-over switch (INTEGER type)	x	x
OCA	Software cam controller	x	x
OR	Logic OR operation (BOOL type)	x	x
OR_W	Logic OR operation (WORD type)	x	
PC	P controller	x	x
PCL	Pulse shortener (BOOL type)	x	x
PDE	Switch-on delay (BOOL type)	x	x
PDF	Switch-off delay (BOOL type)	x	x
PIC	PI controller	x	x
PLI20	Polyline, 20 breakpoints	x	x
PST	Pulse stretching block (BOOL type)	x	x
PT1	Delay element	x	x
RAA	Reset all messages	V4.3	
RDA	Reads out a message	V4.3	
RDAA	Reads out all messages	V4.3	
RDP	Reads drive parameters (REAL type)		x
RDP_D	Reads drive parameters (DOUBLE INTEGER type)		x
RDP_I	Reads drive parameters (INTEGER type)		x
RDP_UD	Reads drive parameters (UNSIGNED DOUBLE INTEGER type)		x
RDP_UI	Reads drive parameters (UNSIGNED INTEGER type)		x
RDP_US	Reads drive parameters (UNSIGNED SHORT INTEGER type)		x
RGE	Ramp-function generator	x	x
RGJ	Ramp-function generator with jerk limiting	x	x
RMDP	Reads drive parameters from the controller	V4.2	

Block	Description	SIMOTION	SINAMICS
RSR	RS flip-flop, R-dominant (BOOL type)	x	x
RSS	RS flip-flop, S-dominant (BOOL type)	x	x
R_D	REAL to DOUBLE INTEGER converter	x	x
R_DW	Acceptance of bit string as DWORD	x	x
R_I	REAL to INTEGER converter	x	x
R_LR	REAL to LONG REAL converter	x	
R_N2	REAL to 16-bit fixed-point format (N2) converter	x	x
R_N4	REAL to 32-bit fixed-point format (N4) converter	x	x
R_SI	REAL to SHORT INTEGER converter	x	
R_UD	REAL to UNSIGNED DOUBLE INTEGER converter	x	x
R_UI	REAL to UNSIGNED INTEGER converter	x	x
R_US	REAL to UNSIGNED SHORT INTEGER converter	x	x
SAH	Sample & hold (REAL type)		x
SAH_B	Sample & hold (BOOL type)		x
SAH_BY	Sample & hold (BYTE type)		x
SAH_D	Sample & hold (DOUBLE INTEGER type)		x
SAH_I	Sample & hold (INTEGER type)		x
SAV	Value buffering (REAL type)	x	x
SAV_BY	Value buffering (BYTE type)	x	x
SAV_D	Value buffering (DOUBLE INTEGER type)	x	x
SAV_I	Value buffering (INTEGER type)	x	x
SH	Shift block (WORD type)	x	
SH_DW	Shift block (DWORD type)	x	x
SII	Inverter	x	x
SIN	Sine function	x	V4.4
SI_D	SHORT INTEGER to DOUBLE INTEGER converter	x	
SI_I	SHORT INTEGER to INTEGER converter	x	
SI_R	SHORT INTEGER to REAL converter	x	
SI_UD	SHORT INTEGER to UNSIGNED DOUBLE INTEGER converter	x	
SI_UI	SHORT INTEGER to UNSIGNED INTEGER converter	x	
SQR	Square-root extractor	x	
SRA	Triggers/resets a message	V4.3	
STM	Fault/alarm trigger		x
SUB	Subtractor (REAL type)	x	x
SUB_D	Subtractor (DOUBLE INTEGER type)	x	x
SUB_I	Subtractor (INTEGER type)	x	x
TAN	Tangent	x	
TRK	Tracking/memory element (REAL type)	x	x
TRK_D	Tracking/memory element (DOUBLE INTEGER type)	x	x
TTCU	Winding harshness characteristic	x	x
UD_I	UNSIGNED DOUBLE INTEGER to INTEGER converter	x	x
UD_R	UNSIGNED DOUBLE INTEGER to REAL converter	x	x
UD_SI	UNSIGNED DOUBLE INTEGER to SHORT INTEGER converter	x	

Block	Description	SIMOTION	SINAMICS
UI_D	UNSIGNED INTEGER to DOUBLE INTEGER converter	x	x
UI_R	UNSIGNED INTEGER to REAL converter	x	x
UI_SI	UNSIGNED INTEGER to SHORT INTEGER converter	x	
US_D	UNSIGNED SHORT INTEGER to DOUBLE INTEGER converter	x	x
US_I	UNSIGNED SHORT INTEGER to INTEGER converter	x	x
US_R	UNSIGNED SHORT INTEGER to REAL converter	x	x
WBG	Wobble generator	x	x
WMDP	Writes drive parameters from the controller	V4.2	
WRP	Writes drive parameters (REAL type)		x
WRP_D	Writes drive parameters (DOUBLE INTEGER type)		x
WRP_I	Writes drive parameters (INTEGER type)		x
WRP_UD	Writes drive parameters (UNSIGNED DOUBLE INTEGER type)		x
WRP_UI	Writes drive parameters (UNSIGNED INTEGER type)		x
WRP_US	Writes drive parameters (UNSIGNED SHORT INTEGER type)		x
W_B	Status word to 16 binary variables converter	x	x
W_BY	Status word to status byte converter	x	x
W_DW	Status word to status double word converter	x	x
XOR	Logic exclusive OR operation (BOOL type)	x	x
XOR_W	Logic exclusive OR operation (WORD type)	x	

8.2 Supplement to SIMODRIVE POSMO A Positioning Motor

Introduction

Contents of the function manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

This documentation serves as a supplement to the documentation on SIMODRIVE POSMO A in the *Distributed Positioning Motor on PROFIBUS DP* user manual.

This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

This manual describes how you can use function blocks to control and assign parameters for a POSMO A drive from a SIMOTION program.

This manual describes differences in handling that arise when controlling and assigning parameters for a POSMO A drive from the SIMOTION system as compared to the SIMATIC system.

Function block

The function blocks for communication between the SIMOTION system and the distributed SIMODRIVE POSMO A positioning motor are part of the program library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.2.1 Fundamental safety instructions

8.2.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.2.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.2.2 Description

8.2.2.1 General

Overview

SIMODRIVE POSMO A is an intelligent distributed positioning drive on the PROFIBUS DP field bus (DP standard slave).

The power unit and all of the motion control are located in the motor.

All of the signals and data for commissioning and operating the drive are transferred via the PROFIBUS DP.

The operating energy is supplied by a 24 VDC connection (for a 75 W motor) or a 48 VDC connection (for a 300 W motor).

The integrated positioning functionality is suitable for a variety of simple single-axis applications, such as adjusting endstops and formats.

Note

Hardware and software requirements

The following requirements apply for the functionalities described in this manual:

- Hardware release POSMO A 75 W: As of O
- Software release POSMO A 75 W: As of V3.0
- Hardware release POSMO A 300 W: As of G
- Software release POSMO A 300 W: As of V3.0

POSMO A positioning motors with different hardware and software requirements can be controlled with function blocks integrated in SIMOTION SCOUT V4.1. The functionality is restricted by the hardware/software release of the POSMO A positioning motor used.

Requirement

The following software versions are required for the standard functions described in this documentation:

- SIMOTION SCOUT V4.1 or higher
- SIMOTION Kernel V4.1 or higher
- SIMOTION technology packages V4.1 or higher

Communication

The PROFIBUS DP field bus allows rapid cyclical data exchange between the DP slave (POSMO A) and the higher-level DP master (SIMOTION hardware platform, such as SIMOTION C2xx).

Further information

Note

For more information, refer to the "Product brief" section of the *Distributed Positioning Motor on PROFIBUS DP* user manual.

This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Installation and connection

For a description of how to install and connect a SIMODRIVE POSMO A and points you must be aware of when doing so, refer to the "Installation and connection" section of the *Distributed Positioning Motor on PROFIBUS DP* user manual.

On the SIMOTION device (hardware platform), connect SIMODRIVE POSMO A to one of the PROFIBUS DP interfaces.

The following figure shows how to connect a SIMODRIVE POSMO A drive to a SIMOTION hardware platform (such as SIMOTION C2xx).

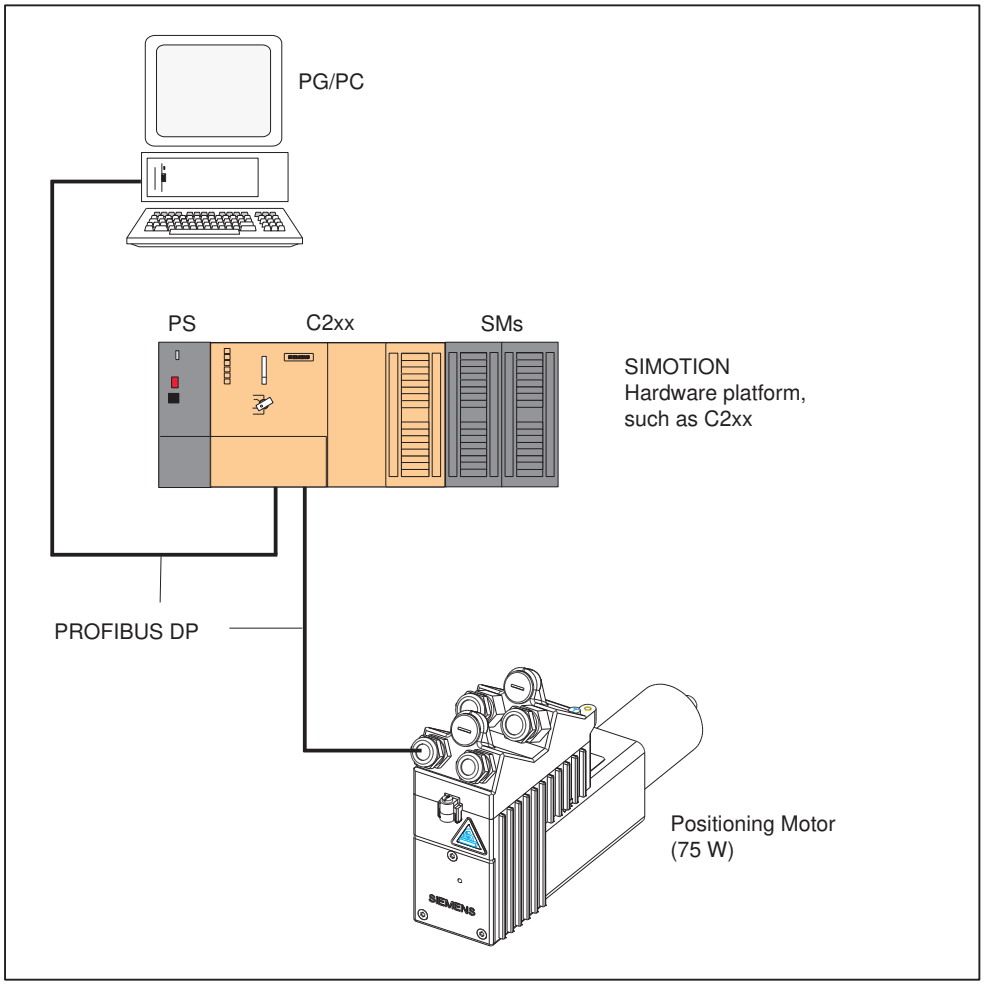


Figure 8-2 Connection of SIMODRIVE POSMO A to the SIMOTION C2xx hardware platform

8.2.2.2 Installation and startup

Overview

You must perform the following steps to commission the SIMODRIVE POSMO A and control it from the SIMOTION system:

1. Mount and wire the SIMODRIVE POSMO A positioning motor.
2. Set the PROFIBUS DP node address on the connection cover of the SIMODRIVE POSMO A.
3. Switch on the terminating resistor at the first and last bus node.

Note

For steps 1 to 3, refer to *Section "Installation and Connection"* of the *Distributed Positioning Motor on PROFIBUS DP* user manual.

4. You can use any of the following to commission the SIMODRIVE POSMO A:
   - C1 master "SIMODRIVE POSMO A PROFIBUS MASTER"
   - Commissioning tool "SimoCom A"

     Note

     Refer to the *Distributed Positioning Motor on PROFIBUS DP* user manual, *Section "Commissioning of DP Master"*.

   - "Drive ES" tool. This tool includes "SimoCom A"

     Note

     Refer to the *Drive ES Basic* function description.

5. Insert the SIMODRIVE POSMO A into the SIMOTION project (refer to Section Inserting a SIMODRIVE POSMO A positioning motor into a SIMOTION project (Page 6047)).
6. Control the SIMODRIVE POSMO A from the SIMOTION system using function blocks, see Section Function blocks (Page 6049).

Note

Refer to the *Distributed Positioning Motor on PROFIBUS DP* user manual for more information on the following topics:

- Axis commissioning
- Communication via PROFIBUS DP
- Description of functions
- Error handling and diagnostics
- Assembly and service

8.2.2.3 Inserting a SIMODRIVE POSMO A positioning motor into a SIMOTION project

Requirement

The following requirements must be met:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet.

Note

For information on creating a project and configuring a PROFIBUS subnet, refer to the online help for SIMOTION SCOUT.

Inserting SIMODRIVE POSMO A

To integrate the SIMODRIVE POSMO A into the PROFIBUS subnet of your project, proceed as follows:

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config** (by double-clicking the SIMOTION device in the project navigator of SIMOTION SCOUT).
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.
4. In the hardware catalog, open the **PROFIBUS DP** folder and the **SIMODRIVE** subfolder and select **SIMODRIVE POSMO A**.
5. Use a drag-and-drop operation to move the SIMODRIVE POSMO A onto the PROFIBUS subnet of your project.
   The **Properties - PROFIBUS Interface SIMODRIVE POSMO A** dialog box is displayed. In this dialog box, you select the address you set in the connection cover of POSMO A (see *Section "Installation and Connection" of the Distributed Positioning Motor on PROFIBUS DP* user manual) and confirm with **OK**.
   The SIMODRIVE POSMO A positioning motor you have selected is inserted into the project.
6. Input and output addresses of the POSMO A.
   When you insert the POSMO A into your SIMOTION project, the input and output addresses are assigned default values. You can see these values when you select the inserted POSMO A. You can read the input and output addresses in the lower part of the **HW Config** window. You must create these addresses as I/O variables in the symbol browser before calling the function blocks; see Section Creating I/O Variables (Page 6048).

8.2.2.4 Integrating the function blocks in the user project

Creating the instance of the FBs in the user project

The function blocks are part of the program library of the SIMOTION SCOUT engineering system. For working with the blocks, an instance has to be created in the user project for each function block used and if using the **_POSMOA_rwAllParameter** function block, a variable of type **Struct_POSMOA_params**.

Example:

```
VAR_GLOBAL
...
 myPosmoAControl        : _POSMOA_control;         // FB for controlling of POSMO A
 myPosmoArwParameter    : _POSMOA_rwParameter;     // FB for handling single parameter
 myPosmoArwAllParameter : _POSMOA_rwAllParameter;  // FB for handling parameterset
 myAllParaPosmoA        : Struct_POSMOA_params;    // Variable for structure of all
                                                   // parameters POSMO A
...
END_VAR
```

Call (LAD representation)

The LAD representation of the individual function blocks can be found in the respective function block descriptions.

Example of an application

The application example is contained on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

8.2.2.5 Creating I/O Variables

Overview

Communication between the SIMOTION hardware platform and the SIMODRIVE POSMO A takes place through direct access to the I/O. I/O variables are used to address the direct read/write access to the I/O.

You can freely assign the names of I/O variables in SIMOTION SCOUT. You must define the I/O variables as ARRAY [0..7] and [0..3] of BYTE. You assign the address settings in the hardware configuration to these I/O variables.

The names of the I/O inputs must be transferred to the function blocks as call parameters. The prepared data for the I/O outputs are provided by the function block as in/out parameters. The

in/out parameters must be be supplied with variables of type ARRAY [0..7] of BYTE and [0..3] of BYTE. Once the block has been called, these variables must be assigned to the I/O variables for the I/O outputs; see call example in Section Calling function blocks (Page 6071).

Note

The variable for supplying the in/out parameters must not be created as a temporary variable (VAR_TEMP or local variable of a function).

The following example shows how to assign the module addresses to the I/O variables in SIMOTION SCOUT.

	Name	I/O address	Read only	Data type	Field length
1	⊞ mypkwin	PIB 256		Array	8
2	⊞ mypzdin	PIB 264		Array	4
3	⊞ mypkwout	PQW 256	☐	Array	8
4	⊞ mypzdout	PQW 264	☐	Array	4

Figure 8-3 Address assignment in SIMOTION SCOUT

Each input and output address has a range of 8 bytes (which corresponds to the parameter identifier value (PKW) range of POSMO A) and a range of 4 bytes (which corresponds to the process data (PZD) range of POSMO A).

Note

For additional information, refer to:

- *SIMOTION SCOUT* online help
- Programming Manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* Programming Manual
  - *SIMOTION MCC, Motion Control Chart Programming Manual*
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram Programming Manual*

These documents are shipped with SIMOTION SCOUT in electronic form!

8.2.3 Function blocks

8.2.3.1 Overview of function blocks

This section contains a description of all of the function blocks (FBs) and the data structure you need for communication between a SIMOTION hardware platform and the SIMODRIVE POSMO A.

The function blocks form the software interface between the SIMOTION system and the SIMODRIVE POSMO A positioning motor.

These function blocks make it easier to control and assign parameters for a SIMODRIVE POSMO A positioning motor from the SIMOTION program.

For example, you can assign parameters for a POSMO A without being familiar with PROFIBUS parameter formats and request specifiers.

The function blocks must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- Function block _POSMOA_control (Page 6050)
- Function block _POSMOA_nControl (Page 6057) (V4.1 and higher):
- Function block _POSMOA_rwParameter (Page 6062)
- Function block _POSMOA_rwAllParameter (Page 6065)

Note

For the complete control and communication of the SIMODRIVE POSMO A from the SIMOTION program, an instance must be created for each **_POSMOA_rwParameter** and **_POSMOA_rwAllParameter** function block and, depending on the parameterized operating mode (speed or position control mode), an instance of the **_POSMOA_control** or **_POSMOA_nControl** function block.

Note

If the SIMODRIVE POSMO A is disconnected and then reconnected to the power system, any MDI traversing block data (see the table titled "Parameters of the _POSMOA_control function block") that had been transferred previously must be transferred to the POSMO A again.

8.2.3.2 Function block _POSMOA_control

Task

You can control the connected SIMODRIVE POSMO A with the **_POSMOA_control** function block.

The functions are as follows:

- Initialize
  Sets the drive in "ready to operate" mode.
  Requirements:
  - A drive fault has not been signaled (**driveError** = FALSE)
  - Fault acknowledgement is not active (**resetError** = FALSE)
- Referencing
  Sets the home position for the drive.
- Tippen
  The drive travels at a controlled speed in a plus or minus direction.
- Program execution
  Starts, stops, or aborts a single block addressed by **blockNumber** or a block within the program.

- MDI
  The drive travels at the assigned speed and acceleration to an assigned position.
  The MDI parameters are transferred in block 3.
  The MDI block can be started with **blockNumber** = 3 and **start** = TRUE.
- Fault acknowledgement
  Acknowledges a fault in the drive.

  Note

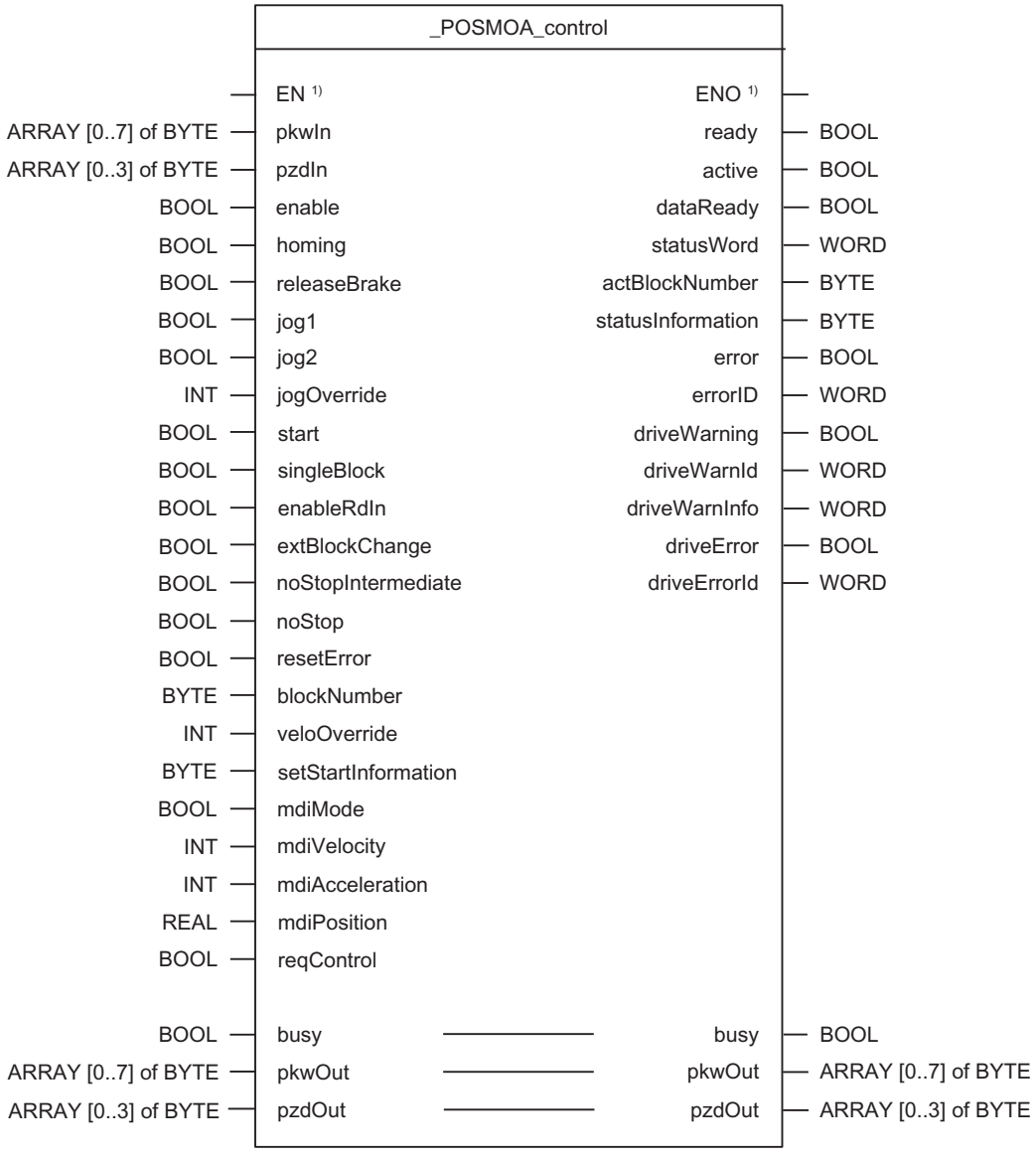
  A fault must be acknowledged before the drive can move. This requires that parameter **enable** = TRUE.

- Automatic single-block operation/automatic control

Checkback signals are as follows:

- Current traversing block
- Data Set Ready
- Warning and fault information
- Complete status (status word and checkback signal byte)
- Data transfer status

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Note

The SIMOTION identifiers have changed as of V4.0.

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6081).

The **busy** parameter must **not** be overwritten by the user. It is supplied and checked by the function block, and must be supplied with a global variable created by the user only when the respective function block is called. This parameter coordinates the individual function blocks for the POSMO A. This ensures that no more than one function block can access a POSMO A at the same time.

Table 8-7 Parameters of the _POSMOA_control function block

Name	P type [1]	Data type	Default	Meaning
pkwIn	IN	ARRAY [0..7] of BYTE	8(16#00)	Transfer I/O inputs of POSMO A to FB
pzdIn	IN	ARRAY [0..3] of BYTE	4(16#00)	Transfer I/O inputs of POSMO A to FB
enable	IN	BOOL	FALSE	Sets drive to ready for operation The drive is now ready for operation provided there are no faults.
homing	IN	BOOL	FALSE	Sets the home position This signal must be present for at least 50 ms.
releaseBrake [4]	IN	BOOL	FALSE	= TRUE: Release holding brake = FALSE: Brake sequence control effective
jog1	IN	BOOL	FALSE	Selection of jog 1 If jog 1 and 2 are set simultaneously, a warning is issued and the drive does not move.
jog2	IN	BOOL	FALSE	Selection of jog 2 If jog 1 and 2 are set simultaneously, a warning is issued and the drive does not move.
jogOverride [3]	IN	INT	20	Speed override of jogging (0 to 100%) The override can also be changed during travel.
veloOverride [3]	IN	INT	20	Velocity override (0 to 100%) This override can also be changed during travel.
start	IN	BOOL	FALSE	= edge FALSE → TRUE: The traversing block specified in **blockNumber** is started. Once a block has been selected in **blockNumber**, the **start** parameter cannot be set until the next block call.
singleBlock[4]	IN	BOOL	FALSE	= TRUE: Automatic single block. Each block has to be re-started. = FALSE: AUTOMATIC mode
enableRdIn[4]	IN	BOOL	TRUE	= TRUE: Read-in enable. Next block is enabled for execution. = FALSE: Read-in disable

Name	P type [1]	Data type	Default	Meaning
extBlockChange [4]	IN	BOOL	FALSE	= Edge FALSE → TRUE: The active block is interrupted and the next block is selected. = FALSE: No external block change
noStopIntermediate	IN	BOOL	FALSE	= TRUE: No intermediate stop or block in intermediate stop is resumed = FALSE: Intermediate stop Interruption of current travel request**start** is not accepted
noStop	IN	BOOL	FALSE	= TRUE: No stop = FALSE: Stop Abort of current travel request If **start** parameter is set at the same time, **start** is not accepted.
blockNumber	IN	BYTE	16#00	Traversing block numbers 3 to 27 Single block or program **blockNumber** = 3 → MDI operation
resetError	IN	BOOL	FALSE	Acknowledges fault 1. Remedy cause of fault. 2. FALSE → TRUE edge 3. Parameter must remain set to TRUE until **driveError** = FALSE.
setStartInformation	IN	BYTE	16#00	Start byte Bit combination transmitted to the drive as an additional start requirement. [2]
mdiMode [3]	IN	BOOL	FALSE	= TRUE: MDI relative The value in the **mdiPosition** parameter is evaluated relative to the current position = FALSE: MDI absolute The value in the **mdiPosition** parameter is evaluated in absolute terms relative to the drive zero position set by homing.
mdiVelocity [3]	IN	INT	0	Velocity of MDI travel (0 to 100%)
mdiAcceleration [3]	IN	INT	0	Acceleration of MDI travel (0 to 100%)
mdiPosition [3]	IN	REAL	0	Target position of MDI travel Value range: $-2 \cdot 10^5$ to $2 \cdot 10^5$
reqControl [4]	IN	BOOL	FALSE	Control requested by the open-loop control p701 = 1: Message frame substitution active = TRUE: PROFIBUS data are taken over by the POSMO A = FALSE: Data from the PROFIBUS are frozen; the data last received are used p701 = 0: Message frame substitution inactive. Behavior as for POSMO A before software version V3.0
pkwOut	IN/OUT	ARRAY [0..7] of BYTE	-	Prepared FB data for I/O outputs of the POSMO A
pzdOut	IN/OUT	ARRAY [0..3] of BYTE	-	Prepared FB data for I/O outputs of the POSMO A
busy	IN/OUT	BOOL	-	Coordination of the FBs
ready	OUT	BOOL	FALSE	Drive ready for operation Logic operation: Status word bits 2 to 0 [2]

Name	P type [1]	Data type	Default	Meaning
active	OUT	BOOL	FALSE	= TRUE: Axis in motion
dataReady [4]	OUT	BOOL	FALSE	Several cycles are required for the transfer , for example, of **mdiPosition** and **jogOverride** = TRUE: Data transfer completed (e.g. **mdiPosition**, **jogOverride**,...) = FALSE: Data transfer in progress (ramp-up time) or data transfer not yet started
statusWord	OUT	WORD	16#0000	Read out of status word [2]
actBlockNumber	OUT	BYTE	16#00	Readout of current block number
statusInformation	OUT	BYTE	16#00	Checkback signal byte Bit combination as additional status signal. [2]
driveWarning	OUT	BOOL	FALSE	A drive warning is pending (refer to parameter **driveWarnId**).
driveWarnId	OUT	WORD	16#0000	Reason for the warning Bit format Value corresponds to parameter 953 (warnings) [2]
driveWarnInfo [4]	OUT	WORD	16#0000	Warnings or supplementary information, corresponds to p954 of the POSMO A
driveError	OUT	BOOL	FALSE	A drive error is pending (refer to the **driveErrorId** parameter)
driveErrorId	OUT	WORD	16#0000	Reason for the error Bit format Value corresponds to parameter 947 (errors) [2]
error	OUT	BOOL	FALSE	= TRUE: Request completed with error (refer to the **errorID** parameter)
errorID	OUT	WORD	16#0000	Number of the parameter assignment error signaled by the drive (parameter identifier value (PKW) range) [2]

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] See *Distributed Positioning Motor on PROFIBUS DP* user manual

[3] This parameter is only transferred when the value of the parameter changes.

[4] As of SIMOTION V4.1, this parameter is part of the **_POSMOA_control** FB and can only be operated with POSMO A as of software version V3.0.

Message frame substitution (POSMO A as of software version V3.0)

For specific applications it is necessary that under no circumstances the drive comes undesirably to a standstill or the drive state can be configured to "freeze" to run-down the master (SIMOTION device).

The "Message frame substitution" function can be activated with the **reqControl** input parameter using software version V3.0 or higher of the POSMO A, with parameter **p701** = **TRUE** set.

The process data sent by the SIMOTION device are taken over by the POSMO A with **reqControl** = **TRUE**. When the transition from **TRUE** to **FALSE** occurs on the **reqControl** input parameter, the

POSMO A uses the process data received most recently (control word, block selection and start byte). If parameter **P701** = **FALSE**, the status of the **reqControl** input parameter is not evaluated.

Note

The "Message frame substitution" function takes immediate effect when p701 = 1.

Make sure that it is possible to shut down the drive at any time using an EMERGENCY STOP.

For additional information, refer to the SIMODRIVE POSMO A user manual, *Distributed Positioning Motor on PROFIBUS DP*.

Task integration (call)

The **_POSMOA_control** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in synchronous tasks (e.g. **IPOSynchronousTask**) is not recommended for runtime reasons.

Note

The functionality of the **_POSMOA_control** FB has been expanded with V4.1. To enable you to use the newly implemented functions, you must add the new input parameters when calling the **_POSMOA_control** FB.

If you want to work with the previous functions (< V4.1), you can leave out the new input parameters with a detailed notation when calling the FB.

Error messages, faults and warnings

The **TRUE** value at the **error** output parameter indicates a parameterization error. The **errorID** output parameter provides more detailed information on the parameterization error that has occurred or has been signaled by POSMO A. Parameterization errors do not need to be acknowledged. Changed parameters (e.g. ramp-up time) can be transferred again.

Faults on the POSMO A are signaled in the **driveError** output parameter with the value **TRUE**. The reason for the fault can be read out in the **driveErrorId** output parameter (value corresponds to P947). Drive faults have to be acknowledged and must be reset in the **resetError** input parameter with the rising edge!

Warnings pending from the POSMO A and the associated information are output in the **driveWarning**, **driveWarnId** (value corresponds to P953) and **driveWarnInfo** (value corresponds to P954) output parameters.

8.2.3.3 Function block _POSMOA_nControl

Task

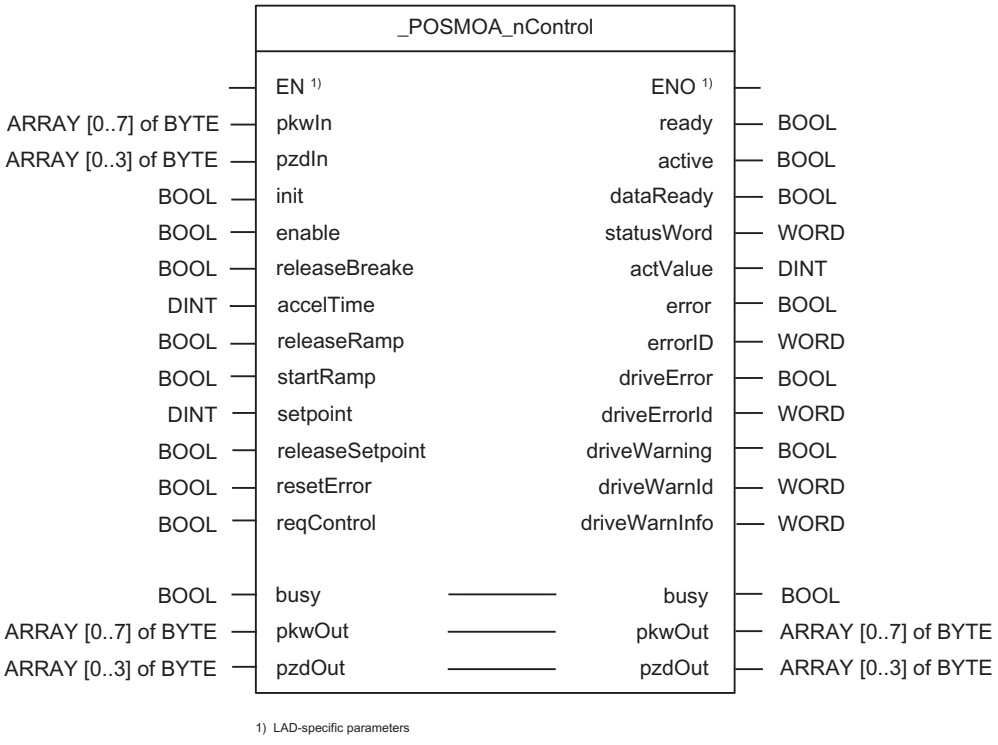
You can control the connected SIMODRIVE POSMO A in speed-controlled mode with the **_POSMOA_nControl** function block.

Note

The **_POSMOA_nControl** function block is contained in SIMOTION SCOUT as of V4.1.

The technology objects (TOs) cannot be used to operate the POSMO A via the speed setpoint interface.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Note

The **busy** parameter must **not** be overwritten by the user. It is supplied and checked by the function block, and must be supplied with a global variable created by the user only when the respective function block is called. This parameter coordinates the individual function blocks for the POSMO A. This ensures that no more than one function block can access a POSMO A at the same time.

Table 8-8 Parameters of the _POSMOA_nControl function block

Name	P type [1)]	Data type	Default	Meaning
pkwIn	IN	ARRAY[0..7] of BYTE	8(16#00)	Transfer I/O inputs of POSMO A to **_POSMOA_nControl** FB
pzdIn	IN	ARRAY[0..3] of BYTE	4(16#00)	Transfer I/O inputs of POSMO A to FB
init	IN	BOOL	FALSE	= TRUE: Sets the drive to "Ready to start" STW = 0x040E
enable	IN	BOOL	FALSE	= TRUE: Sets the drive to "Ready for operation" The drive is now ready for operation (provided there are no errors).
releaseBrake	IN	BOOL	FALSE	= TRUE: Release holding brake = FALSE: Brake sequence control effective
accelTime	IN	DINT	0	Ramp-up/ramp-down time [ms] During this time, the setpoint is adjusted in speed-controlled mode as follows: • Ramp-up: From zero to the maximum permissible actual speed • Ramp-down: From the maximum permissible actual speed to zero
releaseRamp	IN	BOOL	FALSE	= TRUE: Release ramp-function generator output
startRamp	IN	BOOL	FALSE	= Edge FALSE → TRUE: Start ramp-function generator
setpoint	IN	INT	0	Speed setpoint
releaseSetpoint	IN	BOOL	FALSE	Setpoint release **=** TRUE: Setpoint released
resetError	IN	BOOL	FALSE	Acknowledge error 1. Remedy cause of error 2. FALSE → TRUE edge 3. Parameter must remain set to TRUE until **driveError** = FALSE.
reqControl	IN	BOOL	FALSE	Control requested by the open-loop control p701 = 1: Message frame substitution active = TRUE: PROFIBUS data are taken over by the POSMO A = FALSE: Data from the PROFIBUS are frozen; the most recent data received are used p701 = 0: Message frame substitution inactive. Behavior as with POSMO A before software version V3.0
busy	IN/OUT	BOOL	-	Coordination of the function blocks

Name	P type [1]	Data type	Default	Meaning
pkwOut	IN/OUT	ARRAY[0..7] of BYTE	-	Prepared FB data for I/O outputs of the POSMO A (parameter identifier value interface)
pzdOut	IN/OUT	ARRAY[0..3] of BYTE	-	Prepared FB data for I/O outputs of the POSMO A (process data interface)
ready	OUT	BOOL	FALSE	Drive ready for operation, AND operation: Status word bit 2, bit 1, bit 0
active	OUT	BOOL	FALSE	= TRUE: Drive traveling (n > 0)
dataReady	OUT	BOOL	FALSE	Several cycles are required for transferring the ramp-up time, for example. Completion of the data transfer is indicated by a rising edge. = TRUE: Data transfer finished, data have been transferred = FALSE: Data transfer in progress (e.g. ramp-up time)
statusWord	OUT	WORD	16#0000	Display of status word
actValue	OUT	DINT	0	Actual speed
error	OUT	BOOL	FALSE	= TRUE: Request completed with error (refer to the **errorID** parameter)
errorID	OUT	WORD	16#0000	Number of the parameter assignment error signaled by the drive (parameter identifier value (PKW) range) [2]
driveError	OUT	BOOL	FALSE	Drive error is pending
driveErrorId	OUT	WORD	16#0000	Reason for the error Bit format Value corresponds to parameter 947 (errors) [2]
driveWarning	OUT	BOOL	FALSE	A drive warning is pending (refer to parameter **driveWarnId**).
driveWarnId	OUT	WORD	16#0000	Reason for the warning Bit format Value corresponds to parameter 953 (warnings) [2]
driveWarnInfo	OUT	WORD	16#0000	Supplementary information for warnings (for POSMO A, firmware version 1.4 and higher) Bit format The value corresponds to P954, (supplementary information for warnings)

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] See *Distributed Positioning Motor on PROFIBUS DP* user manual

Function description

The POSMO A is set to the "ready to start" state (control word 0x040E) with the **TRUE** level on the **init** input parameter. The "ready to start" state is displayed in the output parameter **statusWord**, bit 0 = **TRUE**. When the transition from **FALSE** to **TRUE** level occurs at the input parameter **enable**, the drive is set to "ready for operation". The POSMO A changes to the "ready for operation" state, which can be read out on the output parameter **ready** = **TRUE**. Traversing is started when the input parameters **enableSetpoint** = **TRUE**, **enableRamp** = **TRUE**, (enable ramp-function generator) **execRamp** = **TRUE** (start ramp-function generator) with a positive edge, and **setpoint** > 0.

The order in which the input parameters **enable**, **enableSetpoint**, **enableRamp**, and **execRamp** are set for starting traversing is at the user's discretion. The input parameters

specified above have equal status. Traversing stops when the input parameters **enable**, **enableSetpoint**, **enableRamp**, and **execRamp** are reset.

Transferring parameters (e.g. ramp-up time - input parameter **accelTime**) requires several task cycles. If a new value is parameterized at the **accelTime** input parameter, the output parameter **dataReady** is set to **FALSE**. Any pending parameter errors (output parameter **error** = **TRUE**) are reset. Data transfer is performed as soon as the in/out parameter **busy** = **FALSE**. If data transmission was active when the **accelTime** input parameter was newly parameterized (e.g. read parameter with **_POSMOA_rwParameter** FB), data transfer is suspended until **busy** = **FALSE**. The value which was parameterized when **busy** = **TRUE** changed to **busy** = **FALSE** at the input parameter **accelTime** is transferred. Completion of data transfer is displayed with a rising edge at the output parameter **dataReady** and stays at **TRUE** until the next data transfer is started.

Message frame substitution (POSMO A, software version V3.0 and higher)

With certain applications, it is essential that the drive be prevented from coming to an undesirable standstill under all circumstances, or that "freezing" of the drive status can be configured for the purpose of shutting down the master (SIMOTION device).

The "Message frame substitution" function can be activated with the **reqControl** input parameter as of software version V3.0 of the POSMO A with set parameter **p701** = **TRUE**.

The process data sent by the SIMOTION device are taken over by the POSMO A with **reqControl** = **TRUE**. With the transition from **TRUE** to **FALSE** on the **reqControl** input parameter, the POSMO A uses the process data received last (control word, block selection and start byte). If parameter **P701** = **FALSE**, the status of the **reqControl** input parameter is not evaluated.

Note

The "Message frame substitution" function takes effect immediately with p701 = 1!

Make sure that the drive can be shut down at any time by an EMERGENCY STOP.

For more information, refer to the SIMODRIVE POSMO A user manual, *Distributed Positioning Motor on PROFIBUS DP*.

Graphical overview of the functionality

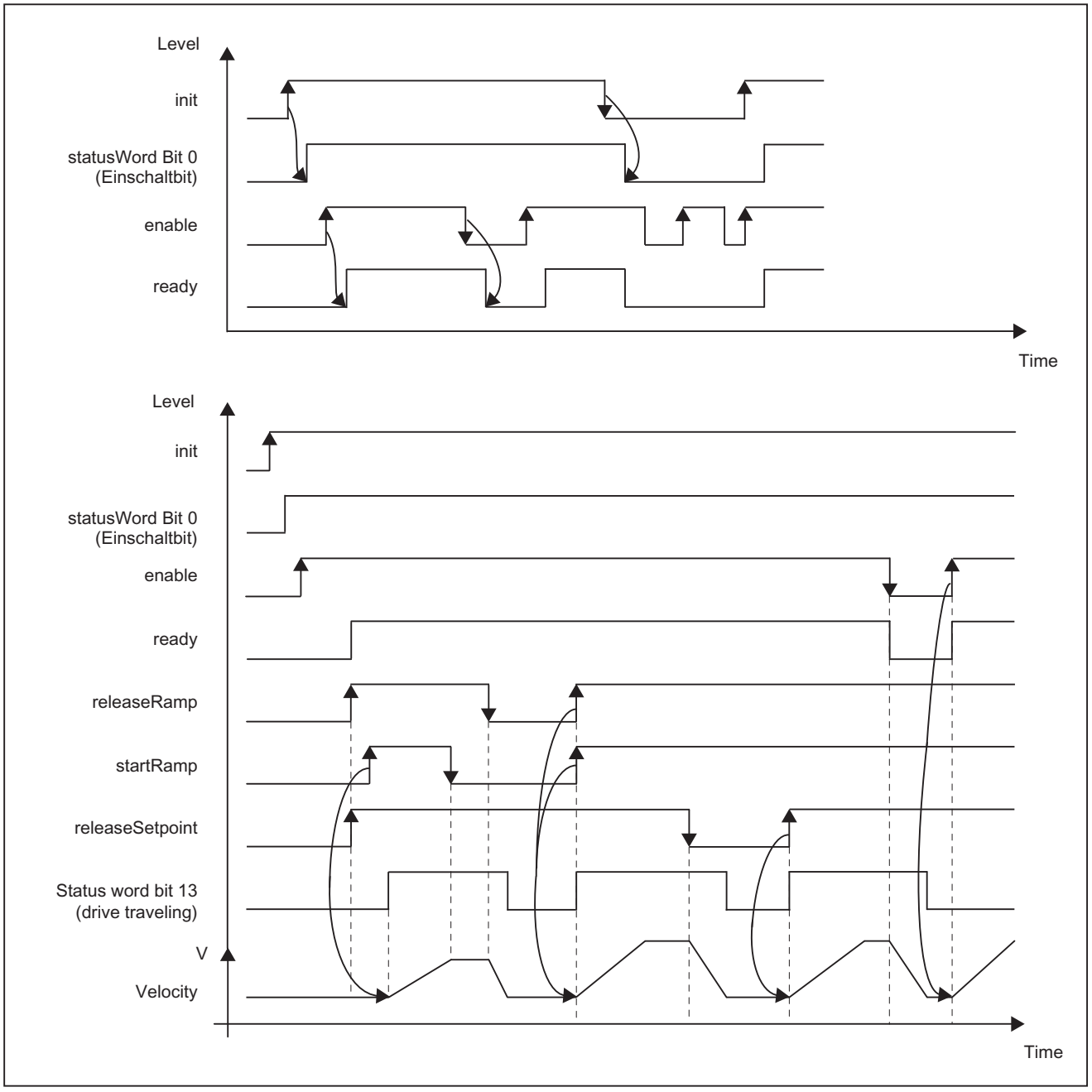


Figure 8-4 Signal propagation diagram

Task integration (call)

The **_POSMOA_nControl** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in synchronous tasks (e.g. **IPOSynchronousTask**) is not recommended for runtime reasons.

Error messages, errors, and warnings

The **TRUE** value at the **error** output parameter indicates a parameterization error. The **errorID** output parameter provides more detailed information on the parameterization error that has occurred. Parameterization errors do not need to be acknowledged. Parameters that have been changed (e.g. ramp-up time) can be transferred again.

Errors in the POSMO A are signaled in the **driveError** output parameter with the value **TRUE**. The reason for the error can be read out in the **driveErrorId** output parameter (value corresponds to P947). Drive errors have to be acknowledged and must be reset in the **resetError** input parameter with a rising edge.

Warnings from the POSMO A that are pending, and their associated information, are output in the **driveWarning**, **driveWarnId** (value corresponds to P953), and **driveWarnInfo** (value corresponds to P954) output parameters.

8.2.3.4 Function block _POSMOA_rwParameter

Task

The **_POSMOA_rwParameter** function block enables parameters to be assigned for the connected SIMODRIVE POSMO A.

The functions are as follows:

- Reading a parameter: Provides the value of the specified parameter.
- Writing a parameter value: Sets the specified parameter to the specified value.
- Loading factory settings: Resets the parameter configuration to the factory settings.
- Saving a parameter: Saves the current parameter configuration in non-volatile memory.

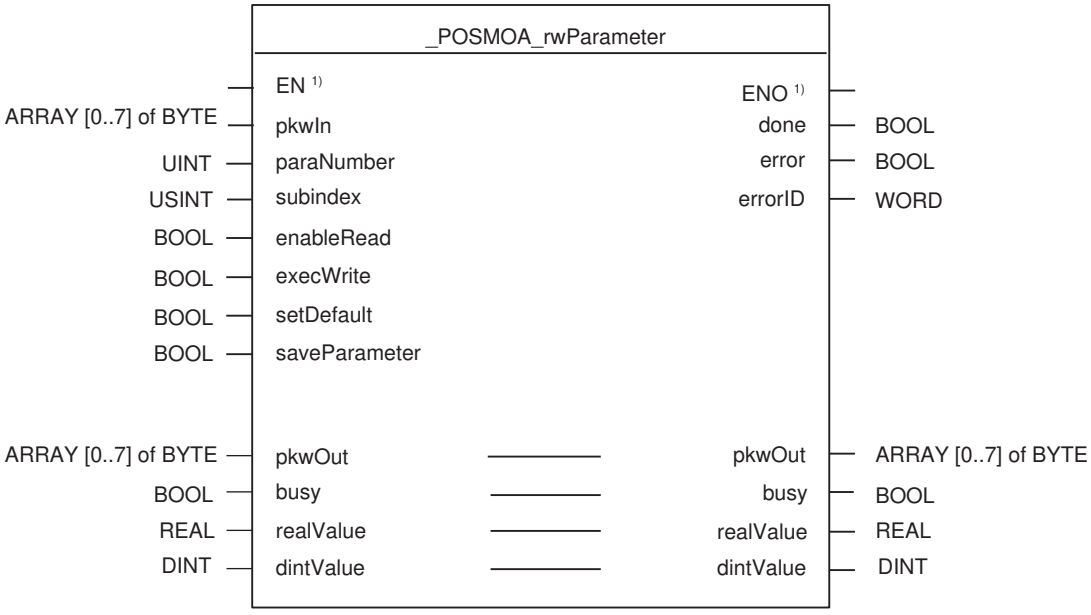
The following parameters can be read/written with this function block:

Parameter numbers	Read from POSMO A	Write to POSMO A (positioning mode)	Write to POSMO A (speed-controlled mode)
1...5	Yes	Yes	Yes
6...7	Yes	Yes	No
9...23	Yes	Yes	Yes
24	Yes	Yes	No
25...38	Yes	Yes	Yes
39...53	Yes	No	No
54	Yes	Yes	Yes
55	Yes	No	No
56...61	Yes	Yes	Yes
62	Yes	No	No
80:28...87:28	Yes	Yes	Yes
99:21	Yes	Yes	Yes
100	Yes	Yes	No
101:11	Yes	Yes	No
700 [1)]	Yes	Yes	Yes

Parameter numbers	Read from POSMO A	Write to POSMO A (positioning mode)	Write to POSMO A (speed-controlled mode)
701 [1]	Yes	Yes	Yes
880 [1]	Yes	Yes	Yes
918...928	No	No	No
930	Yes	No	No
947...954	No	No	No
964:8	Yes	No	No
967...990:78	No	No	No
1426 [1]	Yes	Yes	Yes
1427 [1]	Yes	Yes	Yes

[1] This parameter is new or extended with SIMOTION V4.1.

Call (LAD representation)



[1] LAD-specific parameters

Parameter description

Note

The SIMOTION identifiers have changed as of V4.0.

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6081).

The **busy** parameter must **not** be overwritten by the user. It is supplied and checked by the function block, and must be supplied with a global variable created by the user only when the respective function block is called. This parameter coordinates the individual function blocks for the POSMO A. This ensures that no more than one function block can access a POSMO A at the same time.

Table 8-9 Parameters of the _POSMOA_rwParameter function block

Name	P type [1]	Data type	Default	Meaning
pkwIn	IN	ARRAY [0..7] of BYTE	8(16#00)	Transfer I/O inputs of POSMO A to FB
paraNumber	IN	UINT	0	Parameter number to be read or written
subindex	IN	USINT	0	Subindex = 0 for parameters with no index This value is the array index for parameters with an array. [2]
enableRead	IN	BOOL	FALSE	= TRUE: Reads parameter cyclically = edge FALSE → TRUE: Reads parameter one time
execWrite	IN	BOOL	FALSE	= edge FALSE → TRUE: Writes parameter When set simultaneously with **enableRead**, read is executed.
setDefault	IN	BOOL	FALSE	= edge FALSE → TRUE: Loads factory settings When set simultaneously with **enableRead**, read is executed.
saveParameter	IN	BOOL	FALSE	= edge FALSE → TRUE: Saves parameter When set simultaneously with **enableRead**, read is executed.
pkwOut	IN/OUT	ARRAY [0..7] of BYTE	-	Prepared FB data transferred to the I/O outputs of the POSMO A
busy	IN/OUT	BOOL	-	Coordination of the FBs
realValue	IN/OUT	REAL	-	Write → value to be written (data types C4 and N2) [2] Read → value to be read (data types C4 and N2) [2]
dintValue	IN/OUT	DINT	-	Write → value to be written (data types I2, T2, V2 and T4) [2] Read → value to be read (data types I2, T2, V2 and T4) [2]
done	OUT	BOOL	FALSE	= TRUE: When current request has been completed = FALSE: There is no request pending, or a request is being executed.
error	OUT	BOOL	FALSE	= TRUE: Request completed with error (refer to the **errorID** parameter)
errorID	OUT	WORD	16#0000	Number of the parameter assignment error signaled by the drive (parameter identifier value (PKW) range) [2]

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] See *Distributed Positioning Motor on PROFIBUS DP* user manual

Task integration (call)

The **_POSMOA_rwParameter** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in synchronous tasks (e.g. **IPOSynchronousTask**) is not recommended for runtime reasons.

Fault messages

The **TRUE** value at the **error** output parameter indicates a parameterization error. The **errorID** output parameter provides more detailed information on the parameterization error that has occurred or has been signaled by POSMO A. Parameterization errors do not need to be acknowledged. Changed parameters (e.g. ramp-up time) can be transferred again.

8.2.3.5 Function block _POSMOA_rwAllParameter

Task

The **_POSMOA_rwAllParameter** function block enables reading and writing of the parameter block of the connected SIMODRIVE POSMO A.

The data to be read or written are saved in a variable created by the user with the **Struct_POSMOA_params** data structure when the associated function block instance is called.

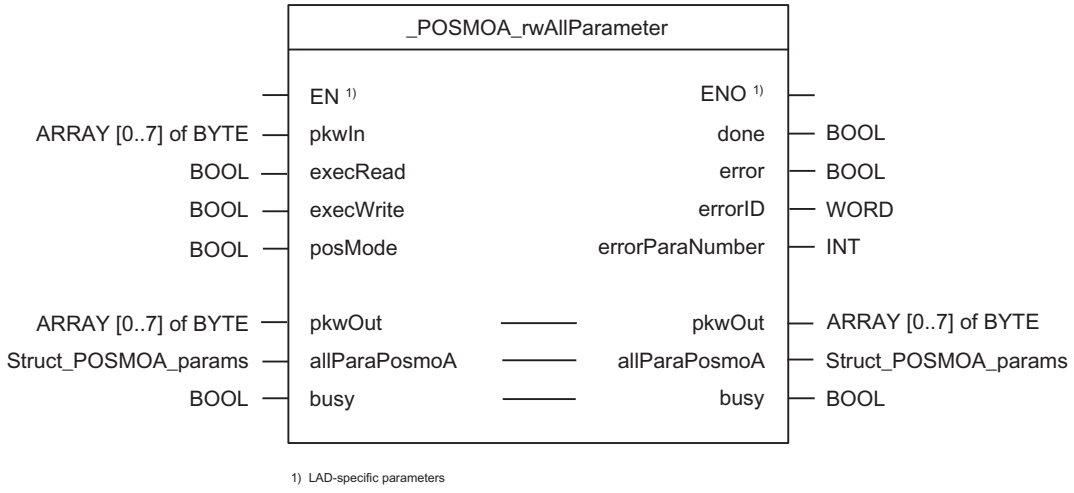
The following parameters are read/written with this function block:

Parameter numbers	Read from POSMO A	Write to POSMO A (positioning mode)	Write to POSMO A (speed-controlled mode)
1...5	Yes	Yes	Yes
6...7	Yes	Yes	No
9...23	Yes	Yes	Yes
24	Yes	Yes	No
25...38	Yes	Yes	Yes
39...53	Yes	No	No
54	Yes	Yes	Yes
55	Yes	No	No
56...61	Yes	Yes	Yes
62	Yes	No	No
80:28...87:28	Yes	Yes	Yes
99:21	Yes	Yes	Yes
100	Yes	Yes	No
101:11	Yes	Yes	No
700 [1)]	Yes	Yes	Yes
701 [1)]	Yes	Yes	Yes
880 [1)]	Yes	Yes	Yes
918...928	No	No	No
930	Yes	No	No

Parameter numbers	Read from POSMO A	Write to POSMO A (positioning mode)	Write to POSMO A (speed-controlled mode)
947...954	No	No	No
964:8	Yes	No	No
967...990:116	No	No	No
1426 [1)]	Yes	Yes	Yes
1427 [1)]	Yes	Yes	Yes

[1)] This parameter is new or extended with SIMOTION V4.1.

Call (LAD representation)



Parameter description

Note

The SIMOTION identifiers have changed as of V4.0.

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6081).

The **busy** parameter must **not** be overwritten by the user. It is supplied and checked by the function block, and must be supplied with a global variable created by the user only when the respective function block is called. This parameter coordinates the individual function blocks for the POSMO A. This ensures that no more than one function block can access a POSMO A at the same time.

Table 8-10 Parameters of the _POSMOA_rwAllParameter function block

Name	P type [1]	Data type	Default	Meaning
pkwIn	IN	ARRAY [0..7] of BYTE	8(16#00)	Transfer I/O inputs of POSMO A to FB
execRead	IN	BOOL	FALSE	= edge FALSE → TRUE: Reads all data one time The start takes place on a positive edge.
execWrite	IN	BOOL	FALSE	= edge FALSE → TRUE: Writes all data one time The start takes place on a positive edge.
posMode [3]	IN	BOOL	TRUE	= TRUE: Positioning mode of the POSMO A = FALSE: Speed-controlled mode
pkwOut	IN/OUT	ARRAY [0..7] of BYTE	-	Prepared FB data for I/O outputs of the POSMO A
allParaPosmoA	IN/OUT	Struct_POS-MOA_params	-	Data structure for all parameters of the POSMO A
busy	IN/OUT	BOOL	-	Coordination of the FBs
done	OUT	BOOL	FALSE	= TRUE: When current request has been completed = FALSE: There is no request pending, or a request is being executed.
error	OUT	BOOL	FALSE	= TRUE: Request completed with error (refer to the **errorID** parameter)
errorID	OUT	WORD	16#0000	Number of the parameter assignment error signaled by the drive (parameter identifier value (PKW) range) [2]
errorParaNumber	OUT	INT	0	Number of the parameter that caused the error [2]

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] See *Distributed Positioning Motor on PROFIBUS DP* user manual

[3] As of SIMOTION V4.1, this parameter is part of the **_POSMOA_rwAllParameter** FB and can only be operated with POSMO A software version 3.0 and higher.

Data structure of Struct_POSMOA_params

The data structure of type **Struct_POSMOA_params** contains all of the parameters for controlling the SIMODRIVE POSMO A.

This data structure is used by the **_POSMOA_rwAllParameter** function block. Self-defined variables of data type **Struct_POSMOA_params** are used to access data structure elements.

The following table contains the **Struct_POSMOA_params** data structure.

Note

The SIMOTION identifiers have changed as of V4.0.

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6081).

Table 8-11 Data structure of Struct_POSMOA_params

Name	Type	Initial value	Comment	r/w [1)]
p1	REAL	0.0	Linear/rotary axis	r/w
p2	REAL	10.0	Travel per gear revolution	r/w
p3	REAL	1.0	Gear reduction factor	r/w
p4	INT	0	Unit of measure	r/w
p5	REAL	0.0	Position at home position	r/w
p6	REAL	-200000.0	Start of software limit switch	r/w
p7	REAL	200000.0	End of software limit switch	r/w
p8	REAL	3000.0	Maximum rotation speed	r/w
p9	INT	10	Rampup time	r/w
p10	DINT	30000	Maximum velocity	r/w
p11	REAL	2.0	Target area	r/w
p12	REAL	20000.0	Maximum following error	r/w
p13	DINT	50	Monitoring time	r/w
p14	REAL	20000.0	Zero speed area	r/w
p15	REAL	0.0	Backlash on reversal compensation	r/w
p16	REAL	9.0	Maximum overcurrent	r/w
p17	DINT	20	P-gain of speed controller	r/w
p18	INT	22	Integral time of speed controller	r/w
p19	REAL	1.0	K_V factor	r/w
p20	REAL	30.0	Current setpoint smoothing	r/w
p21	REAL	2.0	Rotation speed setpoint smoothing	r/w
p22	REAL	1000.0	Maximum acceleration	r/w
p23	DINT	0	Jerk time constant	r/w
p24	INT	100	Override	r/w
p25	INT	100	Acceleration override	r/w
p26	INT	20	Rotation speed override for jogging	r/w
p27	INT	50	Acceleration override for jogging	r/w
p28	REAL	9.0	Maximum current	r/w
p29	DINT	12000	Electronics temperature tolerance time	r/w
p30	INT	0	Interference suppression	r/w
p31	INT	0	Terminal 1 function	r/w
p32	INT	0	Terminal 2 function	r/w
p33	DINT	0	Address for measurement output 1	r/w
p34	INT	7	Shift factor for measurement output 1	r/w
p35	INT	128	Offset for measurement output 1	r/w
p36	DINT	0	Address for measurement output 2	r/w
p37	INT	0	Shift factor for measurement output 2	r/w
p38	INT	128	Offset for measurement output 2	r/w
p39	REAL	0.0	Position reference value	r
p40	REAL	0.0	Actual position value	r
p41	REAL	0.0	Speed setpoint	r

Name	Type	Initial value	Comment	r/w [1]
p42	REAL	0.0	Actual speed	r
p43	REAL	0.0	Current setpoint	r
p44	REAL	0.0	Actual current value	r
p45	DINT	0	Timer status	r
p46	REAL	0.0	Following error	r
p47	REAL	0.0	Electronics temperature	r
p48	INT	0	Current traversing block number	r
p49	INT	0	Subsequent block number	r
p50	DINT	0	Speed setpoint	r
p51	DINT	0	Actual velocity value	r
p52	DINT	0	HW version	r
p53	DINT	0	Firmware version	r
p54	DINT	5	P-gain of speed controller zero speed	r/w
p55	REAL	0.0	Signal position	r
p56	INT	0	Operating position	r/w
p57	DINT	20	P-gain of stop controller zero speed (HW Version F and higher)	r/w
p58	DINT	100	Holding brake release time	r/w
p59	REAL	10.0	Holding brake closure speed	r/w
p60	DINT	400	Holding brake deceleration time	r/w
p61	DINT	100	Holding brake controller disable time	r/w
p62	REAL	0.0	Measuring position	r
p80	ARRAY[0..27] of Array_POSMOA_prgCtrlInfo		Traversing blocks 1 to 27 see Table "Structure of Array_POSMOA_prgCtrlInfo"	r/w
p81	ARRAY[0..27] of REAL	28(0.0)	Target position for traversing blocks 1 to 27	r/w
p82	ARRAY[0..27] of INT	28(100)	Velocity or speed for traversing blocks 1 to 27	r/w
p83	ARRAY[0..27] of INT	28(100)	Acceleration for traversing blocks 1 to 27	r/w
p84	ARRAY[0..27] of DINT	28(0)	Timer value for traversing blocks 1 to 27	r/w
p85	ARRAY[0..27] of REAL	28(0.0)	Signaling position for traversing blocks 1 to 27	r/w
p86	ARRAY[0..27] of INT	28(0)	SMStart MMStart for traversing blocks 1 to 27	r/w
p87	ARRAY[0..27] of INT	28(0)	MMStop MMPos for traversing blocks 1 to 27	r/w
p99	ARRAY[0..20] of INT	13,18,23,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	Program management (see *Distributed Positioning Motor on PROFIBUS DP* user manual)	r/w
p100	INT	0	Control word simulation	r/w
p101	ARRAY[0..10] of INT	10(0)	Blocks 1 to 10 of the data structure for the POSMO A parameters	r/w
p700 [2]	INT	2	Operating mode 1 = speed-controlled mode 2 = positioning mode	r/w
p701 [2]	INT	0	Message frame substitution	r/w
p880 [2]	REAL	4096	Normalizing of the speed at the gear output when a setpoint of 4096 decimal is specified via the control word (STW)	r/w

Name	Type	Initial value	Comment	r/w [1]
p930	INT	0	Current mode 1 = speed-controlled mode 2 = positioning mode	r
p964	ARRAY[0..7] of INT	8(0)	Drive identification	r
p1426 [2]	REAL	100	Tolerance band for actual speed value	r/w
p1427 [2]	INT	0	Delay time for "Ramp-up completed" signal	r/w

[1] r - read, w - write

[2] This parameter is new or extended with SIMOTION V4.1.

Structure of "Array_POSMOA_prgCtrlInfo"

"Array_POSMOA_prgCtrlInfo" contains the program control word. Here, you can define the behavior of a traversing block (see *Distributed Positioning Motor on PROFIBUS DP* user manual).

Table 8-12 Structure of Array_POSMOA_prgCtrlInfo

Array element	Data type	Initial value	Comment
0	BOOL	TRUE	Type of motion
1	BOOL	TRUE	Type of positioning
2	BOOL	FALSE	Type of timer
3	BOOL	FALSE	Connection between timer and start byte
4	BOOL	FALSE	Program return
5	BOOL	FALSE	Type of traversing
6	BOOL	FALSE	Invert start byte condition
7	BOOL	FALSE	SM start type
8	BOOL	FALSE	Program stop
9	BOOL	FALSE	Set actual value

Task integration (call)

The **_POSMOA_rwAllParameter** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in synchronous tasks (e.g. **IPOSynchronousTask**) is not recommended for runtime reasons.

Note

The functionality of the **_POSMOA_rwAllParameter** FB has been expanded with V4.1. To enable you to use the newly implemented functionality, you must add the new **posMode** input parameter when calling the **_POSMOA_rwAllParameter** FB.

If you want to work with the previous functionality (< V4.1), you can leave out the new input parameter with a detailed notation when calling the FB.

Fault messages

The **TRUE** value at the **error** output parameter indicates a parameterization error. The **errorID** output parameter contains more detailed information on the parameterization error that has occurred or has been signaled by POSMO A. The **errorParaNumber** output parameter supplies the number of the parameter that has caused the error.

Parameterization errors do not need to be acknowledged. Changed parameters (e.g. ramp-up time) can be transferred again.

8.2.3.6 Calling function blocks

In order to be able to work with the function blocks in your user program, proceed as follows (The numbers shown in the following program segment correspond to the steps below.):

1. Create the function block instances (see the following program segment, e.g. create instance for the **_POSMOA_control** function block).
2. Create a variable for the data structure (for FB **_POSMOA_rwAllParameter** only).
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the FB are accessed with <instance name of FB>. <name of output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the I/O variables from the array created in step 3 (see the following program segment).

Note

If you want to control more than one SIMODRIVE POSMO A, you must create a new variable for the data structure (FB **_POSMOA_rwAllParameter**) and FB instances with new names for each POSMO A you use.

Call example

```
UNIT E_posmoA;

INTERFACE

// Definition of global variables for demo program
 VAR_GLOBAL
```

```
  myPosmoAControl  : _POSMOA_control;   // create "_POSMOA_control" instance                    (1)
  myEnable         : BOOL;              // enable posmoA
  myHoming         : BOOL;              // homing posmoA
  myJogPositive    : BOOL;              // jog positive posmoA
  myJogNegative    : BOOL;              // jog negative posmoA
  myBusy           : BOOL;              // coordination bit
  myError          : BOOL;              // variable created by user for accessing
                                        // an output variable of the function block
 END_VAR

 PROGRAM ExamplePosmoA;                 // Program of BackgroundTask
END_INTERFACE

IMPLEMENTATION

PROGRAM ExamplePosmoA                   // Program of BackgroundTask
  VAR
  //    temporary array for outputs of FBs                                                    (3)
        tmpPkwOutput : ARRAY[0..7] of BYTE;
        tmpPzdOutput : ARRAY[0..3] of BYTE;
  END_VAR

// INSTANCE CALL of FB _POSMOA_control                                                       (4)
    myPosmoAControl  (    pkwIn    := myPkwIn,                                            (5)
                          pzdIn    := myPzdIn,
                          enable   := myEnable,
                          homing   := myHoming,
                          jog1     := myJogNegative,
                          jog2     := myJogPositive,
                          busy     := myBusy,
                          pkwOut   := tmpPkwOutput,
                          pzdOut   := tmpPzdOutput
                     );
// an output variable in the "_POSMOA_control" function block is assigned to a              (6)
// "myError" variable created by the user.
    myError := myPosmoAControl.error;

// Assignment of intermediate buffer byte arrays to I/O addresses
    myPkwOut := tmpPkwOutput;                                                              (7)
    myPzdOut := tmpPzdOutput;

END_PROGRAM             // ExamplePosmoA
END_IMPLEMENTATION
```

Note

The ExamplePosmoA program must be assigned in the execution system.

8.2.4 Application example

8.2.4.1 General information on the application example

Task

The application example shows how POSMO A can be controlled with the help of the function blocks and how POSMO A drive parameters can be read and written.

Ther is a command interface **enumCommands** for starting the desired action, e.g. jogging.

The **Struct_checkbacks** data structure shows the status of the actions and additional information.

The following operating modes and functionalities are implemented:

- Homing
  Option: "Approach using visual axis marking and assign actual value"
- Jogging
  Move in positive or negative direction
- MDI
  Travel to the required position
- Parameter handling
  - Write or read individual parameters
  - Save all parameters in the EEPROM of the POSMO A
- Read out current actual position
- The current actual position of the POSMO A is read cyclically in jog mode and in MDI and stored in a variable.

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your SIMOTION hardware platform, you must adapt the hardware configuration.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. The configuration in the example can be adapted to suit the available hardware (commission drive, PROFIBUS DP address).
2. The hardware configuration can be adapted to the example (commission drive, PROFIBUS DP address).

Note

Please observe the drive documentation when commissioning the drive.
This documentation is included in the SIMOTION SCOUT scope of delivery in electronic format.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. De-archive and open the project containing the application example.
2. Check the hardware configuration: PROFIBUS DP addresses.
3. Save and compile the example project. You can then download the example to the SIMOTION device and switch to **RUN** mode.

Additional handling steps for the example are carried out using the Enums in the symbol browser within the **myCommand** structure. This requires that the "E_posmoA" element be selected from the **Programs** container in the project navigator.

Values from the "Control value" column are assigned to the corresponding variables by clicking **Immediate control**.

Error messages

Pending errors and warnings (for example, reading during jogging or of individual parameter) are displayed in the following variables:

- myCheckbacks.error = TRUE
  An error has occurred (request canceled; a POSMO A error is pending).
- myCheckbacks.ctrlErrorID
  Error specification of the **_POSMOA_control** function block. Number of the parameter assignment error signaled by POSMO A.
- myCheckbacks.driveErrorID
  Error specification of POSMO A
  Reason for an error signaled by POSMO A
- myCheckbacks.rwErrorID
  Error specification of the **_POSMOA_rwParameter** function block. Error has occurred during reading or writing.

- myCheckbacks.driveWarning = TRUE
  POSMO A warning is pending.
- myCheckbacks.driveWarningID = TRUE
  Warning number of an alarm signaled by POSMO A.

8.2.4.2 Operator control and monitoring of the application example in the detail view

POSMO A will be automatically initialized when your SIMOTION hardware platform changes from STOP to RUN mode. The POSMO A provides feedback that the drive is ready to operate with the following variable:

- myCheckbacks.driveReady = TRUE

Select operating mode

You can choose between "Jog", "Homing", "MDI", or "Parameter handling" modes. This is done via the **myCommand** variable.

"Jog" mode

In the "Jog" mode, the POSMO A can be moved in positive and negative direction.

"Jogging" is implemented with the following parameter settings on the instance created of the **_POSMOA_control** function block:

- jogOverride = 100
- veloOverride = 100
- noStopIntermediate = TRUE — No intermediate stop
- noStop = TRUE — No stop

In the "Control value" column of the symbol browser, select the check boxes for the following variables and select the values to be assigned.

- myCommand = START_JOG_POSITIVE — Jogging in positive direction
- myCommand = START_JOG_NEGATIVE — Jogging in negative direction

Clicking on **Immediate control** assigns the value to the variable, and the POSMO A is moved in the respective direction.

The current "Jog" state can be read in the symbol browser as follows:

- myCheckbacks.actCommand = JOG_POSITIVE_ACTIVE
  Jogging in positive direction activated
- myCheckbacks.actCommand = JOG_NEGATIVE_ACTIVE
  Jogging in negative direction activated
- myCheckbacks.jogPositiveBusy = TRUE
  POSMO A moves in position direction
- myCheckbacks.jogNegativeBusy = TRUE
  POSMO A moves in negative direction

The current actual position of POSMO A can be read in the symbol browser in the **myCheckbacks.actPosition** variable.

Note

The "Jog" mode may be terminated only after the POSMO A is stopped (myCommand = STOP)!

"Homing" mode

The "approach using visual axis marking and assign actual value" options have been implemented. For homing, POSMO A must be switched to closed-loop control and zero speed. The actual value (parameter 40 of POSMO A) can be set via the "Parameter handling", write individual parameter operating mode.

In the "Control value" column of the symbol browser, select the check boxes for the following variables and select the values to be assigned.

- myCommand = START_HOMING

Clicking **Immediate control** assigns the value to the variable, and the POSMO A is homed to the value set in parameter 40 of POSMO A.

Note

For more information on the homing of the POSMO A, refer to the *Distributed Positioning Motor on PROFIBUS DP* user manual.

This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

The current "Homing" state can be read in the symbol browser as follows:

- myCheckbacks.actCommand = HOMING_ACTIVE
  Homing activated
- myCheckbacks.actCommand = NO_COMMAND_ACTIVE
  myCheckbacks.done = TRUE
  Homing completed without error

"MDI" mode

Requirement: The POSMO A is homed!

In the "MDI" mode, one MDI block absolute can be moved.

"Jogging" of the MDI block is implemented with the following parameter settings on the instance created of the **_POSMOA_control** function block:

- mdiMode = FALSE MDI absolute
- mdiVelocity = 100
- mdiAcceleration = 100
- veloOverride = 100

- noStopIntermediate = TRUE — No intermediate stop
- noStop = TRUE — No stop

The target position of the MDI block is specified in the **myAbsolutePosition** variable.

In the "Control value" column of the symbol browser, select the check boxes for the following variables and select the values to be assigned.

- myCommand = START_MDI_BLOCK_ABSOLUTE
- myAbsolutePosition = ... The target position of the MDI block is specified here. (Default = 0.0)

Clicking on **Immediate control** assigns the values to the variables, and the POSMO A moves the MDI block absolute.

The current state of the "MDI" mode can be read in the symbol browser as follows:

- myCheckbacks.actCommand = MDI_BLOCK_ACTIVE
  Move MDI block absolute activated
- myCheckbacks.actCommand = NO_COMMAND_ACTIVE
- myCheckbacks.done = TRUE
- myCheckbacks.positionReached = TRUE
  MDI block traversed without error

The current actual position of POSMO A can be read in the symbol browser, from the **myCheckbacks.actPosition** variable.

Note

The "MDI" mode may be terminated only after the POSMO A is stopped (myCommand = STOP)!

"Parameter handling" mode

In the "Parameter handling" mode you can read and write individual parameters and save all parameters in the EEPROM of POSMO A.

Read individual parameter

In the **myRdParaNumber** variable you state the parameter you want to read. In the **myRdSubIndex** variable, you state the subindex for the parameter you wish to read (indexed parameters only).

In the "Control value" column of the symbol browser, select the check boxes for the following variables and select the values to be assigned.

- myCommand = READ_ONE_PARAMETER
  Read individual parameter
- myRdParaNumber = ...
  You state the number of the parameter to be read here.
- myRdSubIndex = ...
  You state the subindex of the parameter to be read here (for indexed parameters only)

The value read is saved in the **myReadValue** variable.

Write individual parameter

In the **myWrParaNumber** variable you state the parameter you want to write. In the **myWrSubIndex** variable, you state the subindex for the parameter you wish to write (indexed parameters only).

In the **myWrRealValue** variable (data types C4 and N2) [2)] or **myWrDintValue** (data types I2, T2, V2, and T4) [2)], you state the value for the parameter to be written.

In the "Control value" column of the symbol browser, select both the check boxes for the following variables and the values to be assigned.

- myCommand = WRITE_ONE_PARAMETER
  Write individual parameter
- myWrParaNumber = ...
  You state the number of the parameter to be written here.
- myWrSubIndex = ...
  You state the subindex of the parameter to be written here (for indexed parameters only)
- myWrRealValue = ...
  You state the value of the parameter to be written here (data types C4 and N2) [2)].
- myWrDintValue = ...
  Here, you state the value of the parameter to be written (data types I2, T2, V2, and T4) [2)].

[2)] Refer to the *Distributed Positioning Motor on PROFIBUS DP* user manual.
This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Saving all parameters in the EEPROM

In the "Control value" column of the symbol browser, select the check boxes for the following variables and select the values to be assigned.

Clicking **Immediate control** assigns the value to the variable; this activates saving of all parameters in the EEPROM.

- myCommand = SAVE_PARAMETER Saving all parameters in the EEPROM

The current "Parameter handling" state can be read in the symbol browser as follows:

- myCheckbacks.actCommand = READ_PARA_ACTIVE
  Read individual parameter activated
- myCheckbacks.actCommand = WRITE_PARA_ACTIVE
  Write individual parameter activated

- myCheckbacks.actCommand = SAVE_PARAMETER_ACTIVE
  Save all parameters of the POSMO A activated
- myCheckbacks.actCommand = NO_COMMAND_ACTIVE
  myCheckbacks.done = TRUE
  Parameter handling completed without error

Acknowledging faults on POSMO A

Faults on POSMO A are acknowledged as follows:

- myCommand = RESET_ERRORS

The current fault acknowledgement state can be read in the symbol browser as follows:

- myCheckbacks.actCommand = RESET_ERRORS_ACTIVE
  Fault acknowledgement active
- myCheckbacks.actCommand = NO_COMMAND_ACTIVE
  myCheckbacks.done = TRUE
  Fault acknowledgement completed

Note

A POSMO A fault can only be acknowledged successfully when the cause of the error no longer exists.

8.2.4.3 Variables used in application example

Table 8-13 Overview of the variables used

Symbol	Data type	Initial value	Meaning
myCommand	enumCommands	NO_COMMAND	Command interface
myCheckbacks	Struct_Checkbacks	NO_COMMAND_ACTIVE	Command status Additional information

Table 8-14 Overview of the enums enumCommands

Symbol	Enum Value	Meaning
START_HOMING	0	Start homing of POSMO A
START_JOG_NEGATIVE	1	Start jogging in negative direction
START_JOG_POSITIVE	2	Start jogging in positive direction
START_MDI_BLOCK_ABSOLUTE	3	Start moving MDI block absolute
STOP	4	Stop all actions
RESET_ERRORS	5	Acknowledge faults on POSMO A
READ_ONE_PARAMETER	6	Start reading individual parameter
WRITE_ONE_PARAMETER	7	Start writing individual parameter

Symbol	Enum Value	Meaning
SAVE_PARAMETER	8	Save all parameters in the EEPROM of POSMO A
NO_COMMAND	9	No action to be carried out

Table 8-15 Overview of the enums enumActCommand

Symbol	Enum Value	Meaning
HOMING_ACTIVE	0	Homing of POSMO A activated
JOG_NEGATIVE_ACTIVE	1	Jogging in negative direction activated
JOG_POSITIVE_ACTIVE	2	Jogging in positive direction activated
MDI_BLOCK_ACTIVE	3	Moving MDI block absolute activated
STOP_ACTIVE	4	Stop all actions activated
RESET_ERRORS_ACTIVE	5	Acknowledge faults on POSMO A active
READ_PARA_ACTIVE	6	Read individual parameter active
WRITE_PARA_ACTIVE	7	Write individual parameter active
SAVE_PARAMETER_ACTIVE	8	Save all parameters in the EEPROM of POSMO A active
NO_COMMAND_ACTIVE	9	No action active

Table 8-16 Data structure Struct_checkbacks

Symbol	Data type	Meaning
actCommand	enumActCommand	Enums active actions
done	BOOL	Action completed
driveReady	BOOL	Drive (POSMO A) ready for operation
jogPositiveBusy	BOOL	Jogging positive active
jogNegativeBusy	BOOL	Jogging negative active
actPosition	REAL	Current actual position of POSMO A
positionReached	BOOL	Position when moving MDI block absolute reached
error	BOOL	Error has occurred (request canceled; a POSMO A error is pending)
ctrlErrorID	WORD	Error specification of the **_POSMOA_control** block. Number of the parameter assignment error signaled by POSMO A
driveErrorID	WORD	Error specification of the POSMO A Reason for an error signaled by POSMO A
rwErrorID	WORD	Error specification of the **_POSMOA_rwParameter** block. Error during reading or writing has occurred
driveWarning	BOOL	POSMO A warning is pending
driveWarningID	WORD	Warning number

8.2.5 Appendix

8.2.5.1 SIMOTION and SIMATIC names

The table below contains a comparison of SIMOTION and SIMATIC names.

Table 8-17 SIMOTION and SIMATIC names for SIMODRIVE POSMO A

Name in the SIMOTION system as of V4.1 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
Function block parameters		
_POSMOA_control	**FB 10**	**_FB_posmoA_control**
pkwIn	I_O_address	PKWInputInterface
pzdIn	I_O_address	PZDInputInterface
enable	Initialization	initialize
homing	Referencing	homing
releaseBrake	Brake_release	-
jog1	Jogging_1	jog1
jog2	Jogging_2	jog2
jogOverride	Jogging_override	jogOverride
start	Start	start
singleBlock	Automatic_operation	-
enableRdIn	Read_in_enable	-
extBlockChange	External_blockchange	-
noStopIntermediate	No_intermediate_stop	intermediateStop
noStop	No_stop	stop
resetError	Fault_acknowledgement	resetError
blockNumber	Block_number	blockNumber
veloOverride	Override	velocityOverride
setStartInformation	Start_byte	setStartInformation
mdiMode	MDI_type	MDIMode
mdiVelocity	MDI_velocity	MDIVelocity
mdiAcceleration	MDI_acceleration	MDIAcceleration
mdiPosition	MDI_position	MDIPosition
reqControl	Ctrl_Req	-
pkwOut	I_O_address	PKWOutputInterface
pzdOut	I_O_address	PZDOutputInterface
busy	FB_coordination	busy
ready	Ready	ready
active	-	-
dataReady	Data_transfer_ready	-
statusWord	Status_word	statusWord
actBlockNumber	Actual_block	actualBlockNumber
statusInformation	Checkback_signal_byte	statusInformation

Name in the SIMOTION system as of V4.1 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
driveWarning	Warning	driveWarning
driveWarnId	Warn_number	driveWarningNumber
driveWarnInfo	Warn_info	-
driveError	-	driveError
driveErrorId	-	driveErrorNumber
error	Fault	error
errorID	Fault_number	errorNumber
_POSMOA_nControl	**FB 9**	-
enable	-	-
pkwIn	-	-
pzdIn	-	-
init	Initialization	-
releaseBrake	Brake_release	-
accelTime	Acc_Time	-
releaseRamp	Ramp_en	-
startRamp	Ramp_on	-
setpoint	Sp	-
releaseSetpoint	Sp_en	-
resetError	Fault_acknowledgement	-
reqControl	Ctrl_req	-
busy	FB_coordination	-
pkwOut	-	-
pzdOut	-	-
ready	Ready	-
active	-	-
dataReady	Data_transfer_ready	-
statusWord	Status_word	-
actValue	Pv	-
error	-	-
errorID	-	-
driveError	Fault	-
driveErrorId	Fault_number	-
driveWarning	Warning	-
driveWarnId	Warn_number	-
driveWarnInfo	Warn_info	
_POSMOA_rwParameter	**FB 11**	**_FB_posmoA_readWriteParameter**
pkwIn	I_O_address	PKWInputInterface
paraNumber	Number	parameterNumber
subindex	Index	subindex

Name in the SIMOTION system as of V4.1 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
enableRead	Read	read
execWrite	Write	write
setDefault	Factory_default	setFactorySettings
saveParameter	Parameter_save	saveParameter
pkwOut	I_O_address	PKWOutputInterface
busy	FB_coordination	busy
realValue	Value	REALValue
dintValue	Value	DINTValue
done	Task_completed	done
error	Fault_present	error
errorID	Fault_number	errorNumber
_POSMOA_rwAllParameter	**FB 12**	**_FB_posmoA_readWriteAllParameter**
pkwIn	I_O_address	PKWInputInterface
execRead	Read_all	read
execWrite	Write_all	write
posMode	Pos_en	-
pkwOut	I_O_address	PKWOutputInterface
allParaPosmoA	-	allPosmoAParameter
busy	FB_active	busy
done	Task_complete	done
error	Fault_present	error
errorID	Fault_number	errorNumber
errorParaNumber	Fault_parameter_number	errorParameterNumber
Data structure elements		
Struct_POSMOA_params		**Struct_posmoA_parameter**
p1	p1	parameter1
p2	p2	parameter2
p3	p3	parameter3
p4	p4	parameter4
p5	p5	parameter5
p6	p6	parameter6
p7	p7	parameter7
p8	p8	parameter8
p9	p9	parameter9
p10	p10	parameter10
p11	p11	parameter11
p12	p12	parameter12
p13	p13	parameter13
p14	p14	parameter14

Name in the SIMOTION system as of V4.1 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
p15	p15	parameter15
p16	p16	parameter16
p17	p17	parameter17
p18	p18	parameter18
p19	p19	parameter19
p20	p20	parameter20
p21	p21	parameter21
p22	p22	parameter22
p23	p23	parameter23
p24	p24	parameter24
p25	p25	parameter25
p26	p26	parameter26
p27	p27	parameter27
p28	p28	parameter28
p29	p29	parameter29
p30	p30	parameter30
p31	p31	parameter31
p32	p32	parameter32
p33	p33	parameter33
p34	p34	parameter34
p35	p35	parameter35
p36	p36	parameter36
p37	p37	parameter37
p38	p38	parameter38
p39	p39	parameter39
p40	p40	parameter40
p41	p41	parameter41
p42	p42	parameter42
p43	p43	parameter43
p44	p44	parameter44
p45	p45	parameter45
p46	p46	parameter46
p47	p47	parameter47
p48	p48	parameter48
p49	p49	parameter49
p50	p50	parameter50
p51	p51	parameter51
p52	p52	parameter52
p53	p53	parameter53
p54	p54	parameter54
p55	p55	parameter55

Name in the SIMOTION system as of V4.1 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
p56	p56	parameter56
p57	p57	parameter57
p58	p58	parameter58
p59	p59	parameter59
p60	p60	parameter60
p61	p61	parameter61
p62	p62	parameter62
p80	p80	parameter80
p81	p81	parameter81
p82	p82	parameter82
p83	p83	parameter83
p84	p84	parameter84
p85	p85	parameter85
p86	p86	parameter86
p87	p87	parameter87
p99	p99	parameter99
p100	p100	parameter100
p101	p101	parameter101
p700	p700	-
p701	p701	-
p880	p880	-
p930	p930	parameter930
p964	p964	parameter964
p1426	p1426	-
p1427	p1427	-
Program control word		
Array_POSMOA_prgCtrlInfo		Array_posmoA_programControlInformation

8.2.5.2 List of abbreviations

Table 8-18 Abbreviations

Abbreviation	Meaning
DC	Direct current
DP	Distributed I/O
EEPROM	Electrically Erasable Programmable Read-Only Memory
ES	SIMOTION SCOUT
FB	Function block
FW	Firmware
HW	Hardware

Abbreviation	Meaning
IN	Input parameters
IN/OUT	In/out parameter
LAD	Ladder diagram
MDI	**M**anual **D**ata **I**nput
OUT	Output parameters
PIV	Parameter identification value: Parameter part of a PPO
POSMO A	**Pos**itioning **Mo**tor **A**ctuator
PPO	**P**arameter **P**rocess data **O**bject: Cyclic data message frame when transferring data with PROFIBUS DP and the "variable-speed drives" profile
PZD	Process data: Process data part of a PPO
ST	Structured text
STW	Control word
SW	Software
TO	Technology object

8.3 Reading and Writing Drive Data

Preface

Contents of Function Manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

The manual describes how you can use a function block to read and write drive parameters using the PIV (Parameter Identification Value) interface.

Function block

The function block for the "Reading and Writing of Drive Data" for SIMOTION is part of the command library of the "SIMOTION SCOUT" engineering system.

Information in this manual

The following describes the purpose and objectives of the manual:

- General
  This chapter presents information on commissioning the drives, assigning parameters to the PROFIBUS DP interface, and integrating the function block into the SIMOTION system.
- Function blocks
  This chapter describes the function block for reading and writing drive data in connection with the SIMOTION system.
- Index to locate information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.3.1 Fundamental safety instructions

8.3.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.3.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.3.2 Description

8.3.2.1 General

Overview

Siemens offers a wide spectrum of drives with a graduated range of power ratings.

Of this extensive selection of drives, the SIMOTION Motion Control System supports the SIMODRIVE 611 universal, SIMOVERT MASTERDRIVES MC and POSMO CA/CD/SI drives, as well as other drives.

Coupling via PROFIBUS DP enables the drive system and the SIMOTION system to exchange all of their data over this link.

Requirement

The following software versions are required for the standard functions described in this documentation:

- SIMOTION SCOUT V4.0 or higher
- SIMOTION Kernel V4.0 or higher
- SIMOTION technology packages V4.0 or higher

Communication

The PROFIBUS DP field bus allows rapid cyclical data exchange between the DP slave (drive) and the higher-level DP master (SIMOTION hardware platform, such as SIMOTION C2xx).

Note

The **_RWPAR_cyclic** function block described in this documentation enables you to read or write the individual parameters of a drive. In so doing, the parameter channel is transferred in the cyclic message frame. This requires that the relevant drive supports data exchange via the parameter identifier value (PIV) interface, as defined in the **PROFIdrive V2 Drive Profile** (1997).

For drives that support the parameter channel according to the **PROFIdrive V3.1 Drive Profile** (parameter requests are transferred with acyclic DP-V1 message frames), the corresponding system functions are available in SIMOTION:

- **_readDriveParameterDescription**: This function enables the descriptive data of a drive parameter to be read out.
- **_readDriveParameter**: This function enables a drive parameter to be read.
- **_writeDriveParameter**: This function enables a drive parameter to be written.
- **_readDriveFaults**: This function enables the current fault buffer entry in the drive to be read.
- **_readDriveMultiParameterDescription**: This function enables multiple parameter descriptions to be read.
- **_readDriveMultiParameter**: This function enables multiple parameters to be read.
- **_writeDriveMultiParameter**: This function enables multiple parameters to be written.

Further information is contained in the **SIMOTION C2xx / P350 / D4xx System Function/ Variables** lists manual. These documents are shipped with SIMOTION SCOUT in electronic form!

The SIMODRIVE 611 universal, SIMOVERT MASTERDRIVES, MICROMASTER MM4, and POSMO CA/ CD/SI drives support both the PIV interface as well as the acyclic parameter channel. SINAMICS devices support only the acyclic PROFIdrive parameter channel.

8.3.2.2 Start-up and Parameterization of PROFIBUS DP Interface

Requirement

The following requirements must be met:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet.
3. You have commissioned the drive(s) and inserted them in your SIMOTION project.
4. You have configured the drive(s) as a PROFIBUS slave and have assigned parameters for the PROFIBUS DP interface.
5. PIV slot
   If the message frame set during configuration of the drive as a PROFIBUS slave (HW Config > DP slave properties > Configuration tab) does not contain a PIV component, then one PIV slot must be configured for each axis (slot 4 for single-axis drives; slots 4 and 8 for double-axis drives).
   Refer to the figure: "Addresses (PIV slot) in SIMOTION SCOUT"
   Note:
   For SIMOVERT MASTERDRIVES MC, a message frame with a PIV component must be set.

Note

For instructions on creating a project, configuring a PROFIBUS subnet, and inserting a drive in the project, refer to the online help for *SIMOTION SCOUT*.

For additional information, e.g. about commissioning, refer to the user manuals of the installed drives.

8.3.2.3 Integrating the function block in the user project

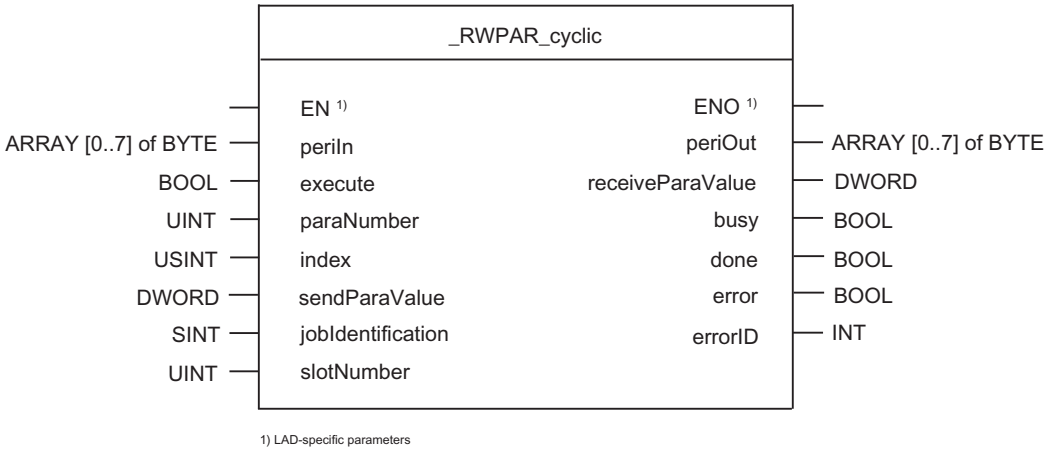
Creating the instance of the FBs in the user project

The function block is part of the command library of the "SIMOTION SCOUT" engineering system. To work with the block, an instance must first be created in the user project.

Example:

```
VAR_GLOBAL
...
  myInstRWParCyclicAxis1: _RWPAR_cyclic ;   // create "_RWPAR_cyclic"
                                             // instance
...
END_VAR
```

Call (LAD representation)



Application example

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

8.3.2.4 Creating I/O Variables

Overview

To read and write drive parameters by means of the PIV interface, you must define I/O variables in SIMOTION SCOUT.

Communication between the SIMOTION device and the drive takes place by means of direct I/O access when data are exchanged over the PIV interface. I/O variables are used to address the direct read/write access to the I/O.

You can freely assign the names of I/O variables in SIMOTION SCOUT. I/O variables must be specified as ARRAY [0..7] of BYTE. You assign addresses set in the hardware configuration (PIV slot) to these I/O variables.

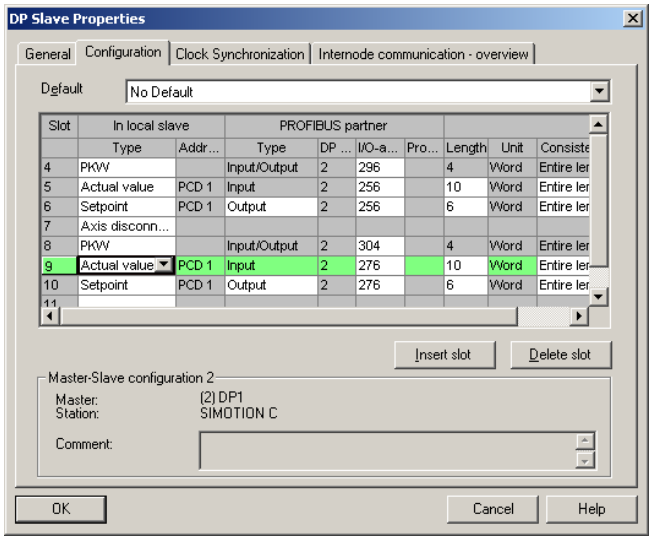


Figure 8-5 Addresses (PIV slot) in SIMOTION SCOUT

The names of the I/O inputs must be transferred to the function block as a call parameter (inputInterface). The data prepared for the I/O outputs are made available as output parameters by the FB and must be assigned to the corresponding I/O variables (see call example in the "Calling the Function Block" section).

The following example shows how to assign the module addresses to the I/O variables in SIMOTION SCOUT.

	Name	I/O address	Read only	Data type	Field length
1	myperipheralinputaxis1	PIB 296		Array	8
2	myperipheraloutputaxis1	PQB 296	☐	Array	8
3	myperipheralinputaxis2	PIB 304		Array	8
4	myperipheraloutputaxis2	PQB 304	☐	Array	8

Figure 8-6 Address assignment in SIMOTION SCOUT

Input addresses and output addresses have a range of 8 bytes each (corresponding to the PIV range of the drive).

Note

For additional information, refer to:

- *SIMOTION SCOUT* online help
- Programming manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* programming manual
  - *SIMOTION MCC, Motion Control Chart* programming manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* programming manual

These documents are shipped with SIMOTION SCOUT in electronic form!

8.3.3 Function block

8.3.3.1 Overview

This section contains a description of the function block (FB) required to read and write drive data between a SIMOTION device and the drive.

This function block makes it easier to read and write drive parameters from a SIMOTION program over the PIV interface of a drive.

The **_RWPAR_cyclic** function block must be called repeatedly (cyclically) from the user program.

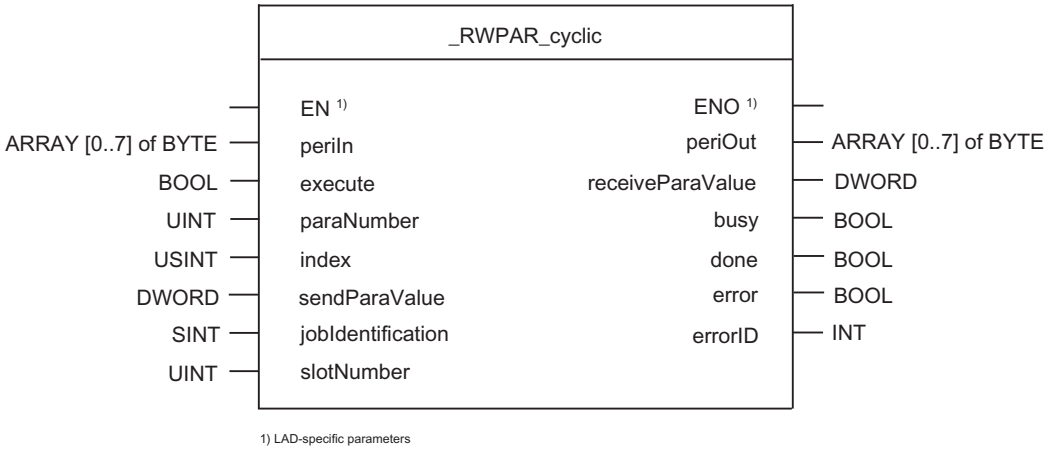
8.3.3.2 _RWPAR_cyclic function block

Task

The **_RWPAR_cyclic** function block enables you to read or write the individual parameters of a drive.

For this purpose, the relevant drive must support the mechanism of data exchange via the PIV interface.

Call (LAD representation)



_RWPAR_cyclic FB parameters

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the designations up to V3.2 / from V4.0 is contained in the "List of parameters" table in Appendix A.

Table 8-19 _RWPAR_cyclic FB parameters

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
periIn	IN	ARRAY [0 to 7] of BYTE	Transfers I/O inputs of drive to FB	I/O variable of the I/O inputs of the drive transferred to the FB	Checked
execute	IN	BOOL	FALSE → TRUE edge Start request	Entered	Checked
paraNumber	IN	UINT	Parameter to be read or written [2]	Entered	Checked
index	IN	USINT	Subindex for parameter to be read or written	Entered	Checked
sendParaValue	IN	DWORD	Value of parameter to be written	Entered	Checked
jobIdentification	IN	SINT	Request identifier [2]	Entered	Checked
slotNumber	IN	UINT	PIV slot number [2]	Entered	Checked
periOut	OUT	ARRAY [0 to 7] of BYTE	Prepared FB data for I/O outputs of drive	Checked and entered on the I/O variable for the I/O outputs	Entered
receiveParaValue	OUT	DWORD	Value of read parameter	Checked	Entered
busy	OUT	BOOL	= TRUE: A request is being processed = FALSE: No request has been received	Checked	Entered
done	OUT	BOOL	= TRUE: If current request is done, only one cycle is active = FALSE: There is no request pending, or a request is being executed.	Checked	Entered
error	OUT	BOOL	= TRUE: Request terminated with error (see **errorID** parameter) - only one cycle is active	Checked	Entered
errorID	OUT	INT	Error specification [2] only one cycle is active	Checked	Entered

[1] Parameter types: IN = input parameter, OUT = output parameter

[2)]See user manuals for relevant drives (PIV range)

Note

If **errorID** = 201, an incorrect slot was configured
Remedy: Check and modify slot configuration!

If **errorID** = 301, transfer of the request was aborted because no correct response data has been received from the drive after 1000 function block call cycles.
Remedy: Repeat request, check communication link if necessary!

If **errorID** = 302, parameter change rights do not exist for the relevant drive
Remedy: Set the parameter change rights and repeat the request.

If you want to read and write indexed parameters for SIMODRIVE 611 universal and SIMODRIVE POSMO CD, CA and SI, you must set bit 11 in the P879 parameter to "1".

For information about drive parameters and their formats and data types, refer to the user manuals for the installed drives.

Function description

This function block processes the PIV interface (4 words wide) in a cyclic message frame.

A complete request (**paraNumber**, **index**, **sendParaValue**, **jobIdentification**) is processed. Consistent data transfer is ensured by the byte ARRAY mechanism for I/O inputs and outputs. A new request is accepted through a positive edge at the **execute** parameter of the block. A **busy** parameter at the block indicates that request processing is in progress. A new request cannot be started until processing of the previous request is finished (**done** = **TRUE**). Any errors are indicated in the **error** and **errorID** parameters.

Signal sequence diagram

The following figure shows the behavior of the parameters of the **_RWPAR_cyclic** function block

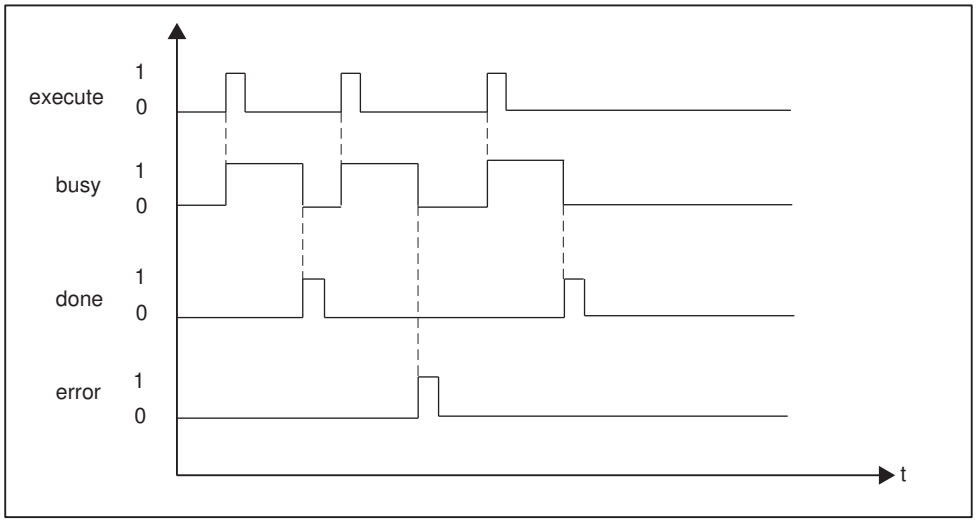


Figure 8-7 _RWPAR_cyclic FB signal sequence diagram

Task integration (call)

The **_RWPAR_cyclic** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

8.3.3.3 Calling the function block

In order to work with the function block in your user program, proceed as follows (the numbers shown in the following program snippet correspond to the steps below):

1. Create an instance of the function block.
2. Call instance of the function block.
3. Transfer input parameters.
4. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.
5. The data for the I/O outputs prepared by the FB are assigned to the I/O variables.

Note

If you want to read or write parameters from multiple drives, you must create an FB instance with a new name for each drive.

Call example

```
UNIT E_rwPar;
```

```
INTERFACE                                                                    (1)
  VAR_GLOBAL
    myInstRWParCyclicAxis1: _RWPAR_cyclic;    // create "_RWPAR_cyclic" instance
    myTransferError        : WORD :=16#0000;   // displays error number of
                                               // data transfer
    myWrOneParaToAxis1     : BOOL :=FALSE;     // starts the transmission of
                                               // proportional gain
                                               // (non indexed parameter) to Axis1
  END_VAR

  PROGRAM ExampleRWParameter;

END_INTERFACE

IMPLEMENTATION

  PROGRAM ExampleRWParameter                   // program of BackgroundTask

  VAR
    mySlotNumberAxis1        : UINT ;
    myStartParamAxis1        : BOOL :=FALSE;   // start flag for data transmission
                                               //of axis 1
    myParaNumAxis1           : UINT  :=0;
    myIndexAxis1             : USINT := 0;
    mySendParaValueAxis1     : DWORD :=16#0000;
    myJobIdentificationAxis1 : SINT := 0;
  END_VAR


    // call instance from standard FB "_RWPAR_cyclic" to proceed data transfer  (2)
    myInstRWParCyclicAxis1( periIn            := myPeripheralInputAxis1,
                            slotNumber        := mySlotNumberAxis1,            (3)
                            execute           := myStartParamAxis1,
                            paraNumber        := myParaNumAxis1,
                            index             := myIndexAxis1,
                            sendParaValue     := mySendParaValueAxis1,
                            jobIdentification := myJobIdentificationAxis1 );
    // copy error code                                                          (4)
    myTransferError:= myInstRWParCyclicAxis1.errorID;


    // write peripheral outputs                                                 (5)
    myPeripheralOutputAxis1 := myInstRWParCyclicAxis1.periOut;

END_PROGRAM

END_IMPLEMENTATION
```

Note

The ExampleRWParameter program must be assigned in the execution system.

8.3.4 Example of an application

8.3.4.1 General

The call example above shows how to read and write a variety of drive parameters from a SIMOTION program in a system with a SIMODRIVE 611 universal double-axis module. The parameters are written either individually or as a group to the 1st and 2nd axis on the double-axis module. Drive parameters are read:

- once from the 1st axis of the double-axis module
- cyclically from the 2nd axis of the double-axis module

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you have to adapt the hardware configuration.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. The configuration in the example can be adapted to the available hardware (commission drive, PROFIBUS DP address).
2. The hardware configuration can be adapted to the example (commission drive, PROFIBUS DP address).

   Note

   For commissioning the drive, please observe the drive documentation!
   This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration: PROFIBUS DP addresses.
3. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Variables used in application example

Symbol	Data type	Meaning	Actions performed by user	Actions performed by example
myWrOneParaToAxis1	BOOL	Transfers the P-gain of the speed controller to Axis 1 (edge-triggered)	Set Reset (on error)	Reset
myWrIndexedParaToAxis1	BOOL	Transfers the ramp-up time of parameter block 1 of Axis 2 - indexed parameter (edge-triggered)	Set Reset (on error)	Reset
myWrThreeParaToAxis1	BOOL	Transfers a set of 3 parameters to Axis 1 (edge-triggered)	Set Reset (on error)	Reset
myWrOneParaToAxis2	BOOL	Transfers the P-gain of the speed controller to Axis 2 (edge-triggered)	Set Reset (on error)	Reset
myRdThreeParaAxis1	BOOL	Reads 3 parameters from axis 1 once	Set Reset (on error)	Reset
myRdIndexedParaAxis1	BOOL	Reads the ramp-up time of parameter block 1 of Axis 1 - indexed parameter (edge-triggered)	Set Reset (on error)	Reset
myRdThreeParaCyclicAxis2	BOOL	Reads 3 parameters from axis 2 cyclically	Set Reset	Read Reset (on error)
myTransferError	INT	Contains the error identifier (**errorID**) of the **_RWPAR_cyclic** FB	Read	Written
myReadCyclicActive	BOOL	= TRUE: cyclical reading of parameters is active	Read	Written

8.3.4.2 Sequence of the application example

This application example illustrates how to read and write the drive parameters for a SIMODRIVE 611 universal double-axis module using the **_RWPAR_cyclic**function block. On the basis of five different parameterized calls of the **_RWPAR_cyclic** FB, drive parameters will be written to the 2 axes of the double-axis module or will be read discretely or cyclically.

Start the transfer operation by setting the global variables (e.g., "myWrOneParaToAxis1" to transfer the P-gain of the speed controller to Axis 1) in the symbol browser and pressing the "Immediate control" button. Once the parameters have been transferred successfully, the variables will be reset by the application example. However, if any transfer errors have occurred, the global variables will not be reset automatically by the application example. In this case, you must reset the global variables (to "FALSE") in the symbol browser yourself. The error ID is saved in the "myTransferError" global variable and displayed in the symbol browser of SIMOTION SCOUT.

The drive parameters are stored in the following structures:

Table 8-20 Overview of the data structures for the drive parameters

Data structure	Meaning
structParameterAxis1	Drive parameters for axis 1
writeParameter	Substructure for the drive parameters to be written for axis 1
readParameter	Substructure for the drive parameters of axis 1 to be read
structParameterAxis2	Drive parameters for axis 2
writeParameter	Substructure for the drive parameters to be written for axis 2
readParameter	Substructure for the drive parameters of axis 2 to be read

Before initiating the transfer, you can modify the values of the drive parameters to be written in the symbol browser and activate them using the "Immediate control" button.

8.3.4.3 Error messages

Any transfer errors that occur are stored in the "myTransferError" global variable and displayed in the symbol browser of SIMOTION SCOUT.

Note

If **myTransferError** = 201, an incorrect slot was configured
Remedy: Check and modify slot configuration!

If **myTransferError** = 301, transfer of the request was aborted because no correct response data have been received from the drive after 1,000 function block call cycles.
Remedy: Repeat request, check communication link if necessary!

If **myTransferError** = 302, parameter change rights do not exist for the relevant drive
Remedy: Set the parameter change rights and repeat the request.

If you want to read and write indexed parameters for SIMODRIVE 611 universal and SIMODRIVE POSMO CD, CA and SI, you must set bit 11 in the P879 parameter to "1".

For information about drive parameters and their formats and data types, as well as a detailed description of the error ID, refer the the *SIMODRIVE 611 universal (PIV range)* function description.
This documentation is included as electronic documentation with the supplied SIMOTION SCOUT!

8.3.5 Appendix

8.3.5.1 List of parameters

A comparison of the SIMOTION identifiers up to V3.2/as of V4.0 is shown in the table below.

Table 8-21 List of parameters

Name in the SIMOTION system as of V4.0 (program library in SCOUT)	**Name in the SIMOTION system up to V3.2 (SIMOTION function library)**
Function block parameters	
_RWPAR_cyclic	**_FB_rwPar_cyclic**
periIn	inputInterface
execute	execute
paraNumber	parameterNumber
index	index
sendParaValue	sendParameterValue
jobIdentification	jobIdentification
slotNumber	slotNumber
periOut	outputInterface
receiveParaValue	receiveParameterValue
busy	busy
done	done
error	error
errorID	errorID

8.4 Standard function for SINAMICS S120 line modules

Foreword

Contents of the function manual

This **document** is part of the **SIMOTION programming references documentation package**.

Function block

The **_LineModule_control** function block is part of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.4.1 Fundamental safety instructions

8.4.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.4.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.4.2 Description

8.4.2.1 General

The **_LineModule_control** function block can only be used in the SIMOTION system

- For the SIMOTION D4x5/D4x5-2 hardware platform or
- For the SIMOTION C, P, and D hardware platforms with a subordinate SINAMICS S120 CU320/CU320-2

in conjunction with a SINAMICS S120 infeed (Line Module) with a DRIVE-CLiQ connection.

From SIMOTION V4.2 onward, the symbolic assignment is activated as standard for newly created projects. For this purpose, no telegrams must be configured for the infeed. If the symbolic assignment is deactivated, SIEMENS telegram 370 must be configured.

The following software versions are required for the standard function described in this documentation:

- SIMOTION SCOUT V4.2 or higher
- SIMOTION Kernel V4.2 or higher

Note

For versions earlier than V4.2, a function block can be found in "SIMOTION Utilities & Applications" for controlling SINAMICS S120 Line Modules (under **Applications > Cross-Sector Applications > Function Block for Controlling Line Modules**).

"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of delivery.

8.4.2.2 Product description

You can use the **_LineModule_control** function block to switch on and off all SINAMICS S120 Line Modules with DRIVE-CLiQ connections from SIMOTION, as well as to perform straightforward diagnostics tasks.

The following infeeds are supported by the **_LineModule_control** function block:

- Active Line Modules (ALM)
- Basic Line Modules (BLM)
- Smart Line Modules (SLM)

The **_LineModule_control** function block is provided for communication between the higher-level SIMOTION control and SINAMICS S120 Line Modules using the command library of the SIMOTION SCOUT engineering system.

The function block is described in this manual.

Note

Line Modules (Active Line Modules, Basic Line Modules, Smart Line Modules) of different types must not be operated simultaneously on the same DC link.

Note

For explanations relating to the status and control word of SINAMICS S120 Line Modules, refer to the

- *SINAMICS S120 Drive Functions* Function Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual

These documents are provided as part of the SIMOTION SCOUT scope of delivery in electronic format.

8.4.3 Parameter assignment / addressing

8.4.3.1 Overview

Communication between the SIMOTION device and the Line Module takes place via the internal PROFIBUS DP (for SIMOTION D4x5/D4x5-2) or the external PROFIBUS DP or PROFINET (for SIMOTION C, P, D with subordinate SINAMICS S120 CU320/CU320-2).

From SIMOTION V4.2 onward, the symbolic assignment is activated as standard for newly created projects. For this purpose, no message frames must be configured for the infeed.

If the symbolic assignment is deactivated, SIEMENS message frame 370 must be configured for use of the **_LineModule_control** function block function. Addresses from 256 are recommended for configuring the Line Modules, where I/O variables are used for the write and read I/O access procedures.

8.4.3.2 Addressing the Line Module for SINAMICS S120

Creating I/O variables

In the SIMOTION user project, two I/O variables with the configured Line Module I/O addresses must be created for the purpose of reading the status word (ZSW) and writing the control word (STW) of the configured Line Module.

The I/O variables must be created in the symbol browser in SIMOTION SCOUT. You are free to choose any names for them. The following shows an example of an Active Line Module created as I/O variable **myperiInAlm** and **myperiOutAlm** . These are also used in the application example: refer to the chapter titled Example of an application (Page 6122).

Activated symbolic assignment

In the case of **activated symbolic assignment,** create the I/O variables in the address list as follows:

1. Create two I/O variables in the address list (e. g. **myperiInAlm** and **myperiOutAlm**), see Figure "Assigning an I/O variable".
2. The status word must be assigned to the I/O variable **myperiInALM**. For this purpose, select the entry "IN" as the I/O address and under data type select the entry "empty" or WORD. In the "Assignment" column, click on the "**...**" button.

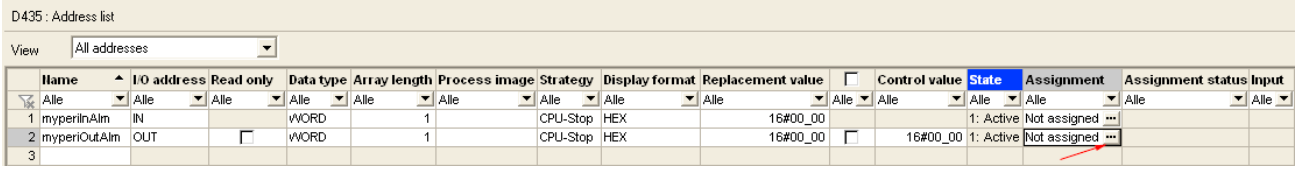


Figure 8-8 Assigning an I/O variable

The "Assigning myperiInALM" window opens.

3. Highlight the status word under infeed and confirm by pressing "OK".

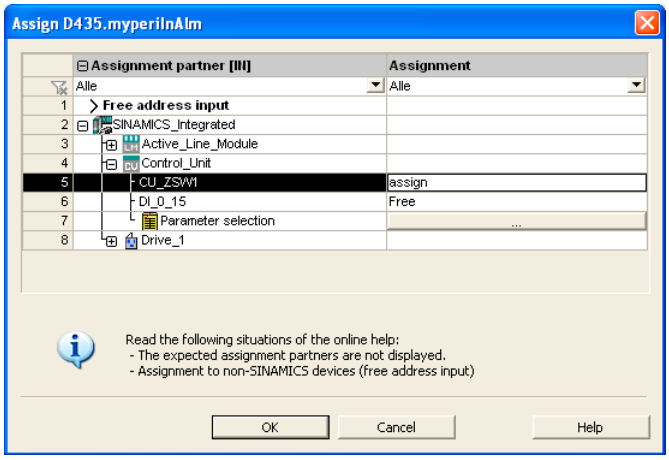


Figure 8-9 Highlighting the status word

The I/O variable is assigned to the status word.

4. Repeat steps 2 to 3 for assigning the control word with the I/O variable **myperiOutAlm**. The I/O variables are now assigned to the status and control word.

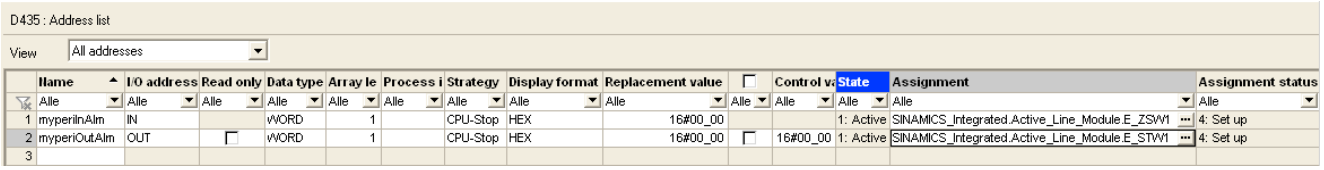


Figure 8-10 Assigned I/O variables in the address list

Continue with the chapter entitled Parameter transfer at FB _LineModule_control (Page 6109).

Deactivated symbolic assignment

In the case of **deactivated symbolic assignment,** proceed as follows to determine the addresses of the Line Module:

1. In the project navigator, double-click "Communication" followed by "Message frame configuration" underneath the SINAMICS drive unit.
2. The "PZD message frames" window opens. Check whether message frame 370 is selected for the infeed. Click on "Set up address". The assigned addresses of the message frames are then displayed. Select the start address for the input data (see figure below).

Example

Determining the addresses from the message frame configuration of the SINAMICS drive unit (using an ALM as an example):

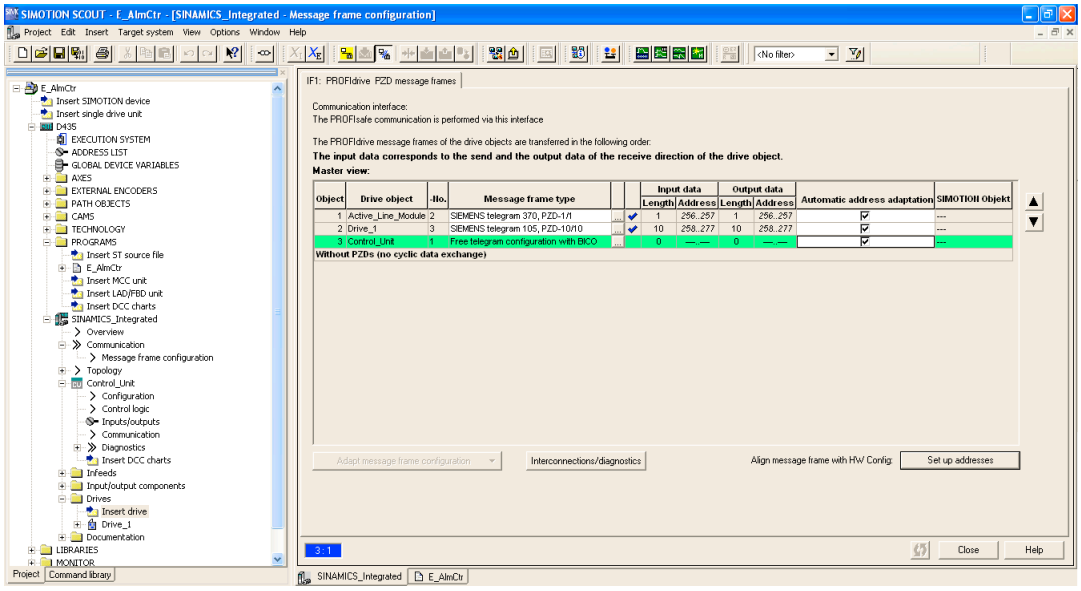
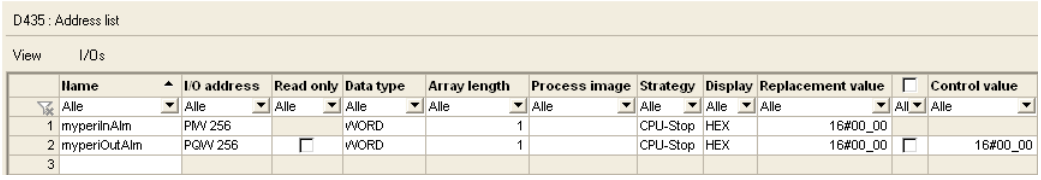


Figure 8-11 Message frame configuration addresses

Creating the I/O variables in the symbol browser:



	Name	I/O address	Read only	Data type	Array length	Process image	Strategy	Display	Replacement value		Control value
	Alle	Alle	Alle	Alle	Alle	Alle	Alle	Alle	Alle	All	Alle
1	myperiInAlm	PIW 256		WORD	1		CPU-Stop	HEX	16#00_00		
2	myperiOutAlm	PQW 256		WORD	1		CPU-Stop	HEX	16#00_00		16#00_00
3											

Figure 8-12 Addressing with I/O variables (example)

Note

Make sure that the addresses are accepted correctly from the message frame configuration. In your user project, when the **_LineModule_control** FB is called you must assign these two I/O variables to the **periIn** and **periOut** input/output parameters. Only once you have done this will the Line Module status word that has been read be transferred to the FB, and the output data prepared by the FB for the control word be transferred to the Line Module.

8.4.3.3 Parameter transfer at FB _LineModule_control

The I/O variable for the I/O inputs (status word) must be transferred to the **periIn** input parameter.

The prepared data for the I/O outputs (control word) are supplied by the **_LineModule_control** FB at the **periOut** output parameter. The **periOut** output parameter must be assigned to the I/O variable for the I/O outputs.

At the **typeLM** input parameter, specify the type of Line Module to be controlled.

8.4.4 Programming

8.4.4.1 _LineModule_control function block

Overview of _LineModule_control FB

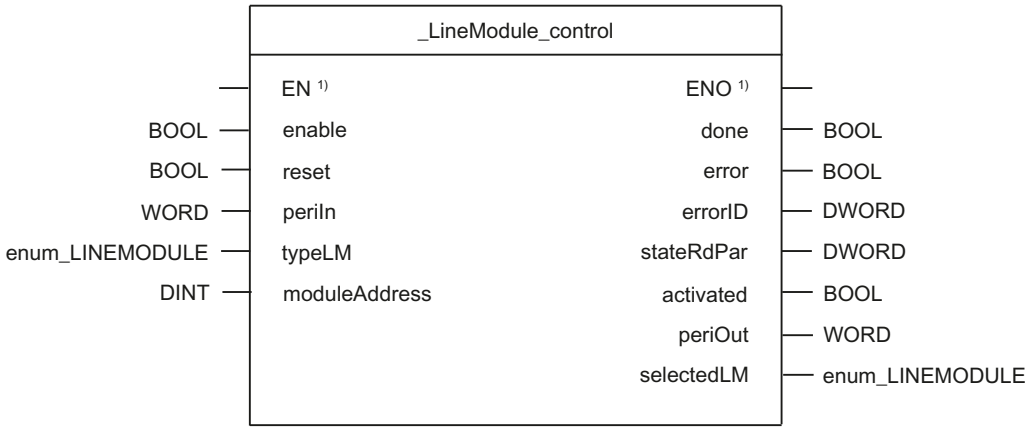
Task

You can use the **_LineModule_control** FB to switch on and off infeeds (Line Modules) for SINAMICS S120 with a DRIVE-CLiQ connection via your user program.
The **_LineModule_control** FB transfers the commands to the selected Line Module, reads the response data provided, and monitors the status signals from the Line Module.

Note

From SIMOTION V4.2 onward, the symbolic assignment is activated as standard for newly created projects. For this purpose, no message frames must be configured for the infeed. If the symbolic assignment is deactivated, SIEMENS message frame 370 must be configured.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

The table below contains all the parameters of the **_LineModule_control** FB.

Table 8-22 Parameters of the _LineModule_control FB

Name	P type [1)]	Data type	Default	Meaning
enable	IN	BOOL	FALSE	Switch line module on/off **TRUE** = Depending on the **typeLM** input parameter, the line module type is determined and the line module is then switched on (FALSE → TRUE status change) **FALSE** = Switch off
reset	IN	BOOL	FALSE	Acknowledge pending errors with rising edge
periIn	IN	WORD	16#0000	The I/O variable for the status word read from the line module is assigned to this input parameter, see Section Parameter assignment / addressing (Page 6107)
typeLM	IN	enum_ LINEMODULE	NOT_DEFINED	• Selection of line module to be controlled: ACTIVE_LINE_MODULE = 1 SMART_LINE_MODULE = 2 BASIC_LINE_MODULE = 3 • Automatic detection of line module type: AUTO_DETECT = 4 (only recommended in exceptional cases) The typeLM input parameter can only be changed when the line module is switched off. If the **typeLM** input parameter is changed in the activated state, there is no reaction. Only after deactivation (enable=FALSE) and subsequent activation (enable=TRUE) is the value in the **typeLM** input parameter taken over.
moduleAddress	IN	DINT	0	Logical address of the line module Only required if **typeLM** = AUTO_DETECT.
done	OUT	BOOL	FALSE	Job completed without errors. Signal present for just one cycle.
error	OUT	BOOL	FALSE	Line module error status **TRUE** = error during the request processing
errorID	OUT	DWORD	16#00000000	Specification of the error For **error** = **TRUE**, the **errorID** parameter contains the error information (refer to the "Error messages" table)
stateRdPar	OUT	DWORD	16#00000000	Error messages or status messages of system function _readDriveParameter() Only relevant if **typeLM** = AUTO_DETECT.
activated	OUT	BOOL	FALSE	Line module operating state: **TRUE** = line module switched on **FALSE** = line module switched off

Name	P type [1)]	Data type	Default	Meaning
periOut	OUT	WORD	16#0000	This output parameter must be linked with the I/O variable for the line module control word; refer to Section Parameter assignment / addressing (Page 6107).
selectedLM	OUT	enum_ LINEMODULE	NOT_DEFINED	Selected or automatically detected line module: • NOT_DEFINED = 0: – If the readDriveParameter() system function was not completed without errors. – If no line module type was selected at the **typeLM** input parameter on a rising edge at the **enable** input parameter. The **activated** output parameter is also set to FALSE in this status. • ACTIVE_LINE_MODULE = 1 • SMART_LINE_MODULE = 2 • BASIC_LINE_MODULE = 3

[1)] Parameter types: IN = input parameter, OUT = output parameter

Function description

At the **typeLM** input parameter, specify the type of Line Module to be controlled. In exceptional cases, e.g. generic blocks for modular machines, automatic detection of the Line Module can also be configured.

At the **selectedLM** output parameter, the selected Line Module or the Line Module determined in the **_LineModule_control** FB is displayed.

A rising edge at the **enable** input parameter switches on the Line Module.

The switch from TRUE to FALSE at the **enable** input parameter switches off the Line Module. Errors present at the **error** output parameter must be acknowledged with a rising edge at the **reset** input parameter. Only the error is acknowledged. Once the error has been successfully acknowledged, the Line Module must be switched on again using a rising edge on the **enable** input parameter. The Line Module will not be switched on if errors are present (**error** = TRUE) while there is a rising edge at the **enable** input parameter. The error must be acknowledged first.

The error reset and the enable can also be executed together. Setting **enable** and **reset** from FALSE to TRUE simultaneously first acknowledges a pending error and then switches on the Line Module.

- If an error has been acknowledged with **reset** = TRUE and no others errors are pending, **error** = FALSE and **errorID** = 0.
- If an error is still present, **error** = TRUE remains and the **errorID** will be updated.

Manual selection of the Line Module

Select the Line Module to be controlled via the parameter **typeLM** :

- **typeLM** = ACTIVE_LINE_MODULE
- **typeLM** = SMART_LINE_MODULE
- **typeLM** = BASIC_LINE_MODULE

If no errors are present, the Line Module selected at the **typeLM** input parameter will be switched on when there is a rising edge at the **enable** input parameter.
The **_LineModule_control** FB does not check the selected Line Module. The _readDriveParameter() system function is not called in the **_LineModule_control** FB. The FB does not evaluate the **moduleAddress** input parameter in this case.

Automatic detection of the Line Module (only in exceptional cases)

General information

In exceptional cases, e.g. generic blocks for modular machines, automatic detection of the Line Module type can be configured:

typeLM = AUTO_DETECT

Note

When using this function, you must take potential programming conflicts with other parallel DPV1 jobs into account, e.g. from the library functions used. Manual selection is recommended for familiar Line Module types.

If the user sets the **typeLM** input parameter to AUTO_DETECT, the **_LineModule_control** FB automatically detects the type of Line Module once when there is a rising edge at the **enable** input parameter. In this mode, the _readDriveParameter() system function is called internally in the FB and the type of Line Module is determined by means of a parameter request. For this function, the configured Line Module address must also be specified at the **moduleAddress** input parameter (see Section "Determining the Logical Address of the Line Module"). Otherwise, the _readDriveParameter() system function called by the **_LineModule_control** FB will signal an error. In the case of error "50001" at the **errorID** output parameter, the **stateRdPar** output parameter will contain specific information on the error from the _readDriveParameter() system function.

Note

You must ensure that only one parameter request is ever active for each drive unit (e.g. SINAMICS Integrated, CU320, CX32). Otherwise, conflicts with other DPV1 jobs may arise, e.g. from the library functions used. Any additional requests sent to the same drive unit will be rejected with the error 16#FFFF81C7 (on output parameter functionResult of the parameter job or on output parameter **stateRdPar** of the **_LineModule_control** FB).

In the case of automatic Line Module detection, no parameter requests from the user program may be active at the drive unit of the Line Module when there is a rising edge at the **enable** input

parameter. If the Line Module type cannot be determined automatically, an error is generated and the Line Module is not switched on.

You can recognize an active parameter request in the **_LineModule_control** FB from the values 0x00007001 or 0x00007002 in the **stateRdPar** output parameter.

The values 0x0000000 or FFFF8xxx (parameter request aborted with an error) in the **stateRdPar** output parameter indicate that the **_LineModule_control** FB is not processing any parameter requests internally.

For additional information on acyclic reading and writing of parameters (using DP-V1 services) with the _readDriveParameter() system function, please refer to the *SIMOTION Communication* System Manual, as well as the *SIMOTION D4x5* Commissioning and Hardware Installation Manual, Section "Acyclic communication with the drive".

Effect of the _readDriveParameter() system function in the _LineModule_control FB

The _readDriveParameter() system function enables the **_LineModule_control** FB to determine the type of Line Module when **typeLM** is set to AUTO_DETECT. For this purpose, the system function reads out parameter r107 of the Line Module. Parameter r107 indicates the Line Module type.

The parameter is read out once before the start of the Line Module switch-on sequence.

Once the Line Module type has been successfully read out, the switch-on sequence is executed.

If the Line Module type is read out incorrectly, the **error** output parameter is set to TRUE, the **selectedLM** output parameter is set to NOT_DEFINED, and the switch-on sequence is not started. Additionally, more detailed information on the error is specified in the **errorID** and **stateRdPar** output parameters.

Requirement for reading out the Line Module type/call:

The _readDriveParameter() system function call starts when there is a rising edge at the **enable** input parameter and **typeLM** is set to AUTO_DETECT for the **_LineModule_control** FB. The system function is called repeatedly until either 0x00000000 (OK) or 0xFFFF8xxx (parameter request aborted with error) is signaled at its output parameter, functionResult.

During the read process, the values 0x00007001 (first call) or 0x00007002 (intermediate call) are output (parameter request active).

All values of the functionResult output parameter of system function _readDriveParameter() are output in exactly the same format at the **stateRdPar** output parameter of the **_LineModule_control** FB. The error functionResult = 0xFFFF81C7 (parameter request active) is an exception to this rule. This error is only output once a timeout has expired at the output parameters of the **_LineModule_control FB** (see the "Timeout behavior" description below).

You can use the **stateRdPar** output parameter to execute (start/end) reading of the internal FB parameters.

Timeout behavior in relation to the _readDriveParameter() system function:

If a parameter request from a user or from the SIMOTION device is already active at the Line Module (functionResult = 0xFFFF81C7 error message), the parameter is read out repeatedly within a timeout period of 1 s.

The **stateRdPar** output parameter of the **_LineModule_control** FB remains at value 0x00007002 during the repeated readout process taking place within the timeout.

The parameter is only read again if there is a functionResult = 0xFFFF81C7 error. If the system function signals any other errors, an error is output immediately at the **_LineModule_control** FB and the parameter reading process is aborted with the **_readDriveParameter** system function.

If the functionResult = 0xFFFF81C7 error is still present once the timeout period has elapsed, the following error is output at the output parameters of the **_LineModule_control** FB:

error = TRUE
errorID = 50001
stateRdPar = 0xFFFF81C7
selectedLM = NOT_DEFINED

Output parameters in relation to the _readDriveParameter() system function:

Table 8-23 Output parameters in relation to _readDriveParameter():

Parameters of the _LineModule_control FB	Task
stateRdPar	Output the functionResult output parameter of the _readDriveParameter() system function
selectedLM	Output the Line Module type determined If a parameter has been assigned incorrectly (e.g. the wrong module address) or an error occurs during parameter reading, this output parameter will be set to NOT_DEFINED.

Determining the logical address of the Line Module

If you are using the automatic detection method, you need to assign a value for the module address (logical address) of the Line Module at the **moduleAddress** input parameter of the **_LineModule_control** FB (see Chapter Determining the module address (Page 6115)).

Determining the module address

To select a diagnostic interrupt in the PeripheralFaultTask, the module address (logical address) of the Line Module must be determined and made readily available in a variable.

In the case of activated symbolic assignment, determine the module address using the system function **_getLogicalAddressOfIoVariable** from the name of the I/O variable for the status or control word of the Line Module.

In the case of deactivated symbolic assignment, proceed in the same way as for determining the addresses of I/O variables, but this time determine the module address for the status and control word of the Line Module. Use the same address value (start address of input data) in the same way as for I/O variables for the status and control word of the Line Module, see Addressing the Line Module for SINAMICS S120 (Page 6107).

Graphical overview of the functionality

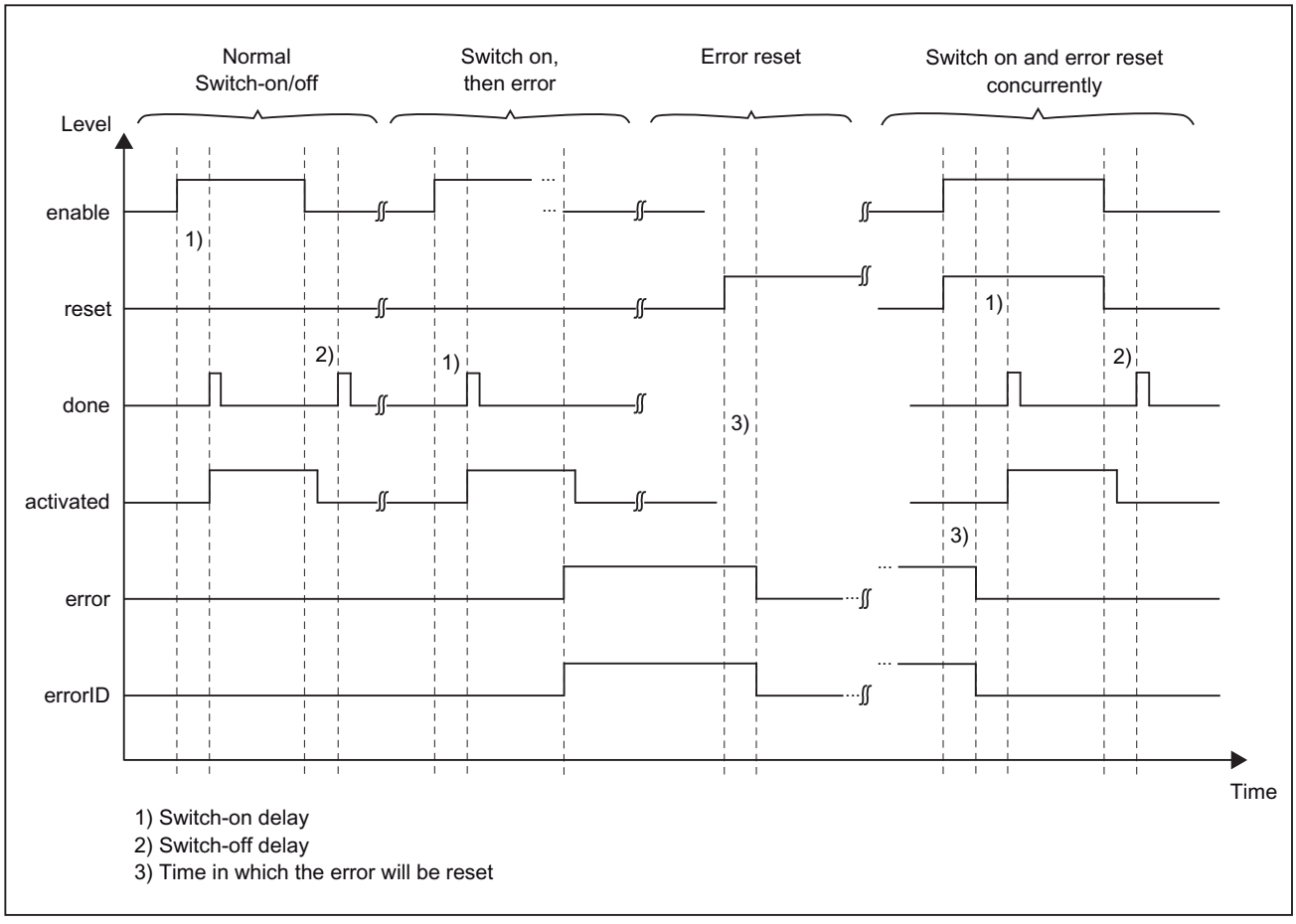


Figure 8-13 Signal propagation diagram

Task integration (call)

The **_LineModule_control** FB is designed to be called in a cyclic task and must be called in this task during each task pass. Processing a job can take several cycles. The user decides which cyclic task of the **_LineModule_control** FB the call is made in. No restrictions are applied with respect to the functionality of the **_LineModule_control** FB.

8.4.4.2 Calling the function block

Procedure

Proceed as follows to work with the **_LineModule_control** function block in your user project (the numbers shown in the program segment below correspond to the steps listed):

1. Create an instance of the **_LineModule_control** function block.
2. Call the function block instance and transfer input parameters.

3. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.
4. The data for the I/O outputs (control word of the Active Line Module) prepared by the FB must be assigned by the user program to the I/O variables for the purpose of writing the control word. It is recommended that you use the _setSafeValue system function for this assignment task.

Note

The program segment is an extract from the application example supplied. The application example is included on the "SIMOTION Utilities & Applications" DVD and is available for various SIMOTION hardware platforms.

"SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of delivery.

Note

For additional information, see the following sources:

- *SIMOTION SCOUT* online help
- Programming manual of the corresponding programming language, e.g.:
  - *SIMOTION ST*, *Structured Text* programming manual
  - *SIMOTION MCC*, *Motion Control Chart* programming manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* Programming Manual

These documents are provided as part of the SIMOTION SCOUT scope of delivery in electronic format.

Program segment from the sample program:

```
UNIT E_AlmCtr;

INTERFACE

VAR_GLOBAL
  myFbAlmCtrl : _LineModule_control;    // Instance of FB _LineModule_control                    (1)
END_VAR

PROGRAM StartUpAlm;
PROGRAM BackGrndAlm;
PROGRAM PeripheralFaultAlm;

END_INTERFACE

IMPLEMENTATION
```

```
PROGRAM BackGrndAlm
// ********************************************************************
// call instance of FB _LineModule_control
// ********************************************************************

  myFbAlmCtrl ( enable    := myAlmCtrl_In.enable,                                    (2)
                reset     := myAlmCtrl_In.reset,
                typeLM    := ACTIVE_LINE_MODULE,
                periIn    := myAlmCtrl_In.periIn
              );
  myAlmCtrl_Out.error     := myFbAlmCtrl.error;                                      (3)
  myAlmCtrl_Out.errorID   := myFbAlmCtrl.errorID;
  myAlmCtrl_Out.done      := myFbAlmCtrl.done;
  myAlmCtrl_Out.activated := myFbAlmCtrl.activated;
  myAlmCtrl_Out.periOut   := myFbAlmCtrl.periOut;

//**************************************************************************
// write the output parameters of FB _LineModule_control - control WORD ALM -
// to i/o-variable myperiOutAlm
//**************************************************************************
  s_eRetVal := _setSafeValue (                                                       (4)
               variable   := myperiOutAlm,
               value      := myAlmCtrl_Out.periOut,
               accessmode := default_value,
               setvalue   := s_setValue
              );

END_PROGRAM

END_IMPLEMENTATION
```

8.4.4.3 Error messages

The value **TRUE** at the **error** output parameter indicates an error status in the Line Module. More details of the error are provided at the **errorID** output parameter. The assignment of the error cause is made in this output parameter.

If an internal Line Module (200xx group) fault occurs, the user must read out the exact cause using the appropriate SIMOTION system functions, e.g. **_readDriveFaults**.

For a detailed description of the system functions, refer to the *SIMOTION System Functions/ Variables Device* Parameter Manual. This document is provided as part of the SIMOTION SCOUT scope of delivery in electronic format.

Error groups

The errors signaled in the **errorID** output parameter may be allocated to the following error groups:

Group 100xx:	A time-out is present. The Line Module does not respond to the **_LineModule_control** FB commands, or responds too late.
Group 200xx:	An internal Line Module fault is present. "Fault active" bit of the Line Module status word = **TRUE** The exact cause of the error must be read out using the appropriate SIMOTION system functions (e.g. **_readDriveFaults**).
Group 300xx:	The Line Module has withdrawn the "Controlled by PLC" bit in the status word during operation.
Group 400xx:	An error occurred during a **reset**.
Group 500xx:	An error occurred during the execution of the _readDriveParameter() system function used internally by the **_LineModule_control** FB. More detailed error information can be obtained from the **statusRdPar** output parameter.

Error messages

Note

Statuses S1 to S4 are contained in the **errorID** output parameter while the Line Module is in the process of being switched on or off.

You can find the meanings of the statuses (S1 to S4) described in the table below in the Appendix Flow diagrams for switching the Line Modules on and off (Page 6125).

For explanations relating to the status and control word of SINAMICS S120 Line Modules, refer to the

- *SINAMICS S120 Drive Functions* Function Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual

These documents are provided as part of the SIMOTION SCOUT scope of delivery in electronic format.

The error groups and error messages listed below are to be viewed as being in decimal format.

Table 8-24 Error messages

Error no. (errorID), decimal	**Meaning**
100xx	Timeout during the status transition of the Line Module. • 1000x: Time-out when switching on: Status x (S1...S4) was not attained. • 100x0: Time-out when switching off: Status x (S1...S4) was not attained. **Example:** 10030: Time-out error when switching off. Status S3 was either not attained at all or attained too late.
200xx	The Line Module reports an internal fault using the "Fault active" bit of the status word. • 2000x: Internal fault when switching on: Status x (S1...S4) was not attained. • 200x0: Internal fault when switching off: Status x (S1...S4) was not attained. **Example:** 20002: Internal fault when switching on: In status S2. The exact cause of the error must be read out using the appropriate SIMOTION system functions (e.g. _readDriveFaults).
30000	While there was a rising edge on the **enable** input parameter, the "Controlled by PLC" bit in the status word of the Line Module was set to **FALSE**. This means that the command could not be performed.
300xx	The Line Module has reset the "Controlled by PLC" bit in the status word. The FB resets all bits in the Line Module control word and waits for new commands. • 3000x: The "Controlled by PLC" bit in the status word was reset to Status x (S1...S4) when switching on. • 300x0: The "Controlled by PLC" bit in the status word was reset during operation or when switching off to Status x (S1...S4). **Example:** 30002: The "Controlled by PLC" bit in the status word was reset during switching on to Status S2.
40001	While there was a rising edge on the **reset**input parameter, the "Controlled by PLC" bit in the status word of the Line Module was set to **FALSE**. This means that the command could not be performed.
40003	While there was a rising edge on the **reset** input parameter, the Line Module did not reset the "Fault active" bit in the status word, even though the "Acknowledge error" bit was set in the control word.
50001	Only relevant if input parameter **typeLM** = AUTO_DETECT An error occurred during the execution of the _readDriveParameter() system function. Evaluate the **statusRdPar** output parameter. Information on the errors can be found in the *SIMOTION System Functions/Variables Device* List Manual.

Error correction

Use the methods outlined below to correct pending errors.

Table 8-25 Information on correcting pending errors

Check the diagnostics LEDs on the Line Module and all SINAMICS components. • Check the RDY LED. • Check the DC-LINK LED. For descriptions of all the LEDs, please refer to the *SINAMICS S120 Booksize Power Units* Manual.
Check the communication between SIMOTION and the connected Line Module. Has SIEMENS message frame 370 been configured for the connected Line Module? See Section Parameter assignment / addressing (Page 6107)
Check the programming for the _LineModule_control FB. • Does the symbol browser in your project contain one I/O variable for reading the status word and one for writing the control word? • Is the I/O variable for reading the status word assigned to input parameter **periIn**of the **_LineModule_control** FB? • Is output parameter **periOut** assigned to the I/O variable for writing the control word after the **_LineModule_control** FB is called? • Is the **_LineModule_control** FB called in a cyclic task and run during each cycle? See Section Parameter assignment / addressing (Page 6107)
Check whether the connected Line Module has signaled any errors. • Check the error messages in the "Alarms" window of SIMOTION SCOUT. • In the Line Module expert list, check the error messages in the following parameters: p945 (fault code) p947 (fault number) p2131 (current fault code) • Check the error messages via the user program, using the _readDriveFaults() system function. For information on the causes of the errors read and how to remedy them, please refer to the *SINAMICS S* List Manual and the *SIMOTION SCOUT* online help. On SIMOTION Utilities & Applications, you will also find a DPV1 library containing a function block for reading drive fault and warning messages. SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.
Check the entries in the SINAMICS diagnostics buffer (D4x5 SINAMICS Integrated/CX32: SIMOTION V4.1 SP2 and higher; CU320: SINAMICS Firmware V2.6 and higher) 1. Right-click the SINAMICS device in the SIMOTION SCOUT project navigator. 2. Select "Target device" > "Device Diagnostics" from the context menu and open the "Diagnostics buffer" tab.

Check the status of the SINAMICS drive. **Line Module:** 1. Open the Line Module expert list in SIMOTION SCOUT. 2. Check the following parameters: r2 (status display) p10 = 0 (commissioning parameter filter) r46 (missing enables) **Control Unit:** 1. Open the Control Unit expert list in SIMOTION SCOUT. 2. Check the following parameters: r2 (status display) p9 = 0 (device commissioning parameter filter)
Check the Line Module wiring. • Check the supply voltage for the connected Line Module. • Check the supply voltage at the EP terminals. • Check the DRIVE-CLiQ topology. • Check the wiring of the Active Interface Module (if present).

See also

_LineModule_control function block (Page 6110)

8.4.5 Example of an application

8.4.5.1 General

Task

The application example shows:

- How you can use the **_LineModule_control** FB to switch a Line Module on and off. This is shown using the example of an ALM.
- How error statuses are signaled by the **_LineModule_control** FB, and how you can reset the errors.
- Additional ALM diagnostic information.

The application example contains the following programs:

Table 8-26 Application example programs

Program	Task	Meaning
StartUpALM	StartupTask	Program for start-up
PeripheralFaultALM	PeripheralFaultTask	Program for handling diagnostic alarms
BackGrndALM	BackgroundTask	Program for switching the ALM on and off

Structure of the example

In the application example, an Active Line Module (ALM) is used as the infeed. The application example is structured as follows:

- Type declarations
- Variable definitions
- Variable initialization
- Switch-on sequence
- Switch-off sequence
- Evaluation of diagnostic alarms in the **PeripheralFaultTask**

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" DVD. "SIMOTION Utilities & Applications" is provided free of charge and as part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration:
3. If necessary, modify the example project.
4. Save and compile the example project. You can then download the example to the SIMOTION device and switch to RUN mode.

Adapting the application example

The address of the Line Module in the frame configuration must match the addresses of the I/O variables in the application example. To enable selection of a diagnostic interrupt in the PeripheralFaultTask, the address of the Line Module must be adapted in a subsequent variable.

You must adapt the following settings where necessary:

- Address of I/O variables **myperiInAlm** and **myperiOutAlm** (default: 256), see Chapter Addressing the Line Module for SINAMICS S120 (Page 6107)
- **myAlmModuleAddress**(default: 256)
  In the case of activated symbolic assignment, determine the module address via the system function **_getLogicalAddressOfIoVariable**, see Chapter Determining the module address (Page 6115).

8.4.5.2 Sequence of the application example

Relevant variables in the application example

Table 8-27 Overview of the relevant variables

Variable	Data type	Initial value	Meaning
mySwitchOn	BOOL	FALSE	**TRUE** = Start the switch-on sequence
mySwitchOff	BOOL	FALSE	**TRUE** = Start the switch-off sequence
myFirstRun	BOOL	TRUE	**TRUE** = Initialize the variables
myError	BOOL	FALSE	**TRUE** = Error on ALM or error during switching on/off
myErrorId	DWORD	16#00000000	Error specification
myDiagnosticAlarm	BOOL	FALSE	**TRUE** = Diagnostic alarm present on ALM
myProcessAlarm	BOOL	FALSE	**TRUE** = process alarm present
myAlmCtrl_In	Struct_AlmControlIn	-	Structure for input parameter of the **_LineModule_control** FB
myAlmCtrl_Out	Struct_AlmControlOut	-	Structure for output parameter of the **_LineModule_control** FB
myperiInAlm	WORD	16#0000	I/O variable with ALM address for status word
myperiOutAlm	WORD	16#0000	I/O variable with ALM address for control word
myAlmModuleAddress	DINT	256	ALM address for selection of diagnostic alarm in **PeripheralFaultTask**
myPftTsi	Struct_PeripheralFaultTaskTsi	-	Task start information for **PeripheralFaultTask**

StartUpALM program

In the StartUpAlm program, a flag for the initial run is set and then evaluated in the BackgroundTask. This allows for the implementation of a standard start-up sequence for not only the STOP – RUN transition, but also an appropriate user request.

BackGrndALM program

The BackGrndALM program contains 2 program sequences for switching the Active Line Module on and off. In both sequences, the switch-on/switch-off procedure is monitored for errors and tested to ensure it has completed successfully.

The switch-on procedure for the ALM is initiated using a positive edge on the **mySwitchOn** variable. The steps that follow involve testing the switch-on procedure to ensure it has completed successfully and checking it for errors. The **mySwitchOn** variable is set to FALSE once the switch-on procedure has begun.

The switch-off procedure for the ALM is initiated using a positive edge on the **mySwitchOff** variable. The value of the **mySwitchOff** variable is then set to FALSE. The step that follows involves testing the procedure to ensure it has completed successfully and checking it for errors.

Any errors that occurred during processing are displayed in the **myError** and **myErrorId** global variables and can be reset using the **myAlmCtrl_In.reset** variable.

PeripheralFaultALM program

If the module triggering an error is the ALM, the start information for the **PeripheralFaultTask** is written to the **myPftTsi** global variable. If the start information evaluation produces a diagnostic or process alarm signaled by the ALM, this is displayed in the **myDiagnosticAlarm** or **myProcessAlarm** global variable.

8.4.6 Appendix

8.4.6.1 Flow diagrams for switching the Line Modules on and off

Overview

The flow diagrams below describe the statuses (S1 to S4) present while the Line Module is ramping up.

The Line Modules are switched off using the same procedure used to switch them on, but in the reverse order.

Note

For explanations relating to the status and control word of SINAMICS S120 Line Modules, refer to the

- *SINAMICS S120 Drive Functions* Function Manual
- *SIMOTION D4x5* Commissioning and Hardware Installation Manual

These documents are provided as part of the SIMOTION SCOUT scope of delivery in electronic format.

Flow diagram for switching on the Active Line Module (ALM)

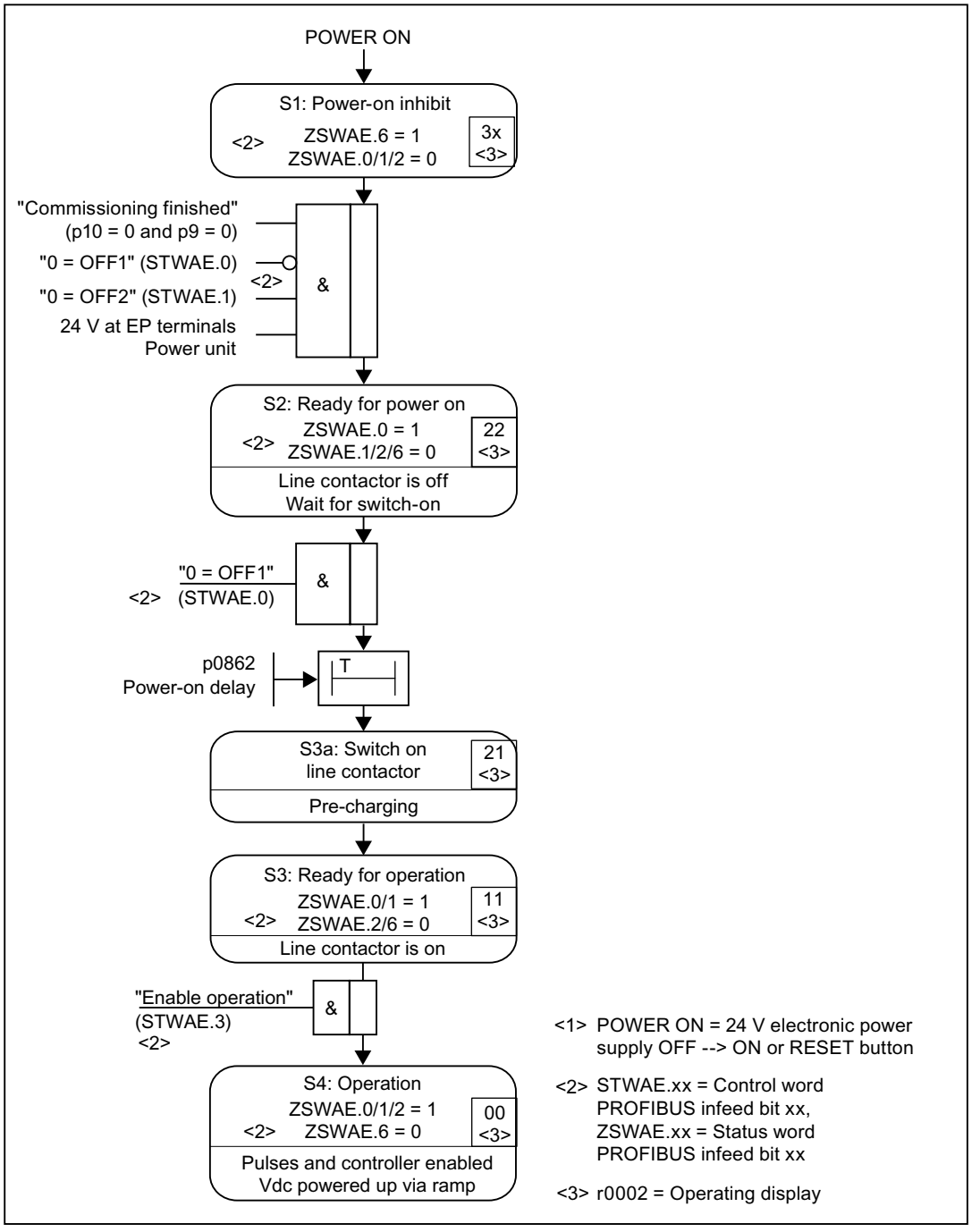


Figure 8-14 Flow diagram: Switching on the ALM

Flow diagram for switching on the Basic Line Module (BLM)

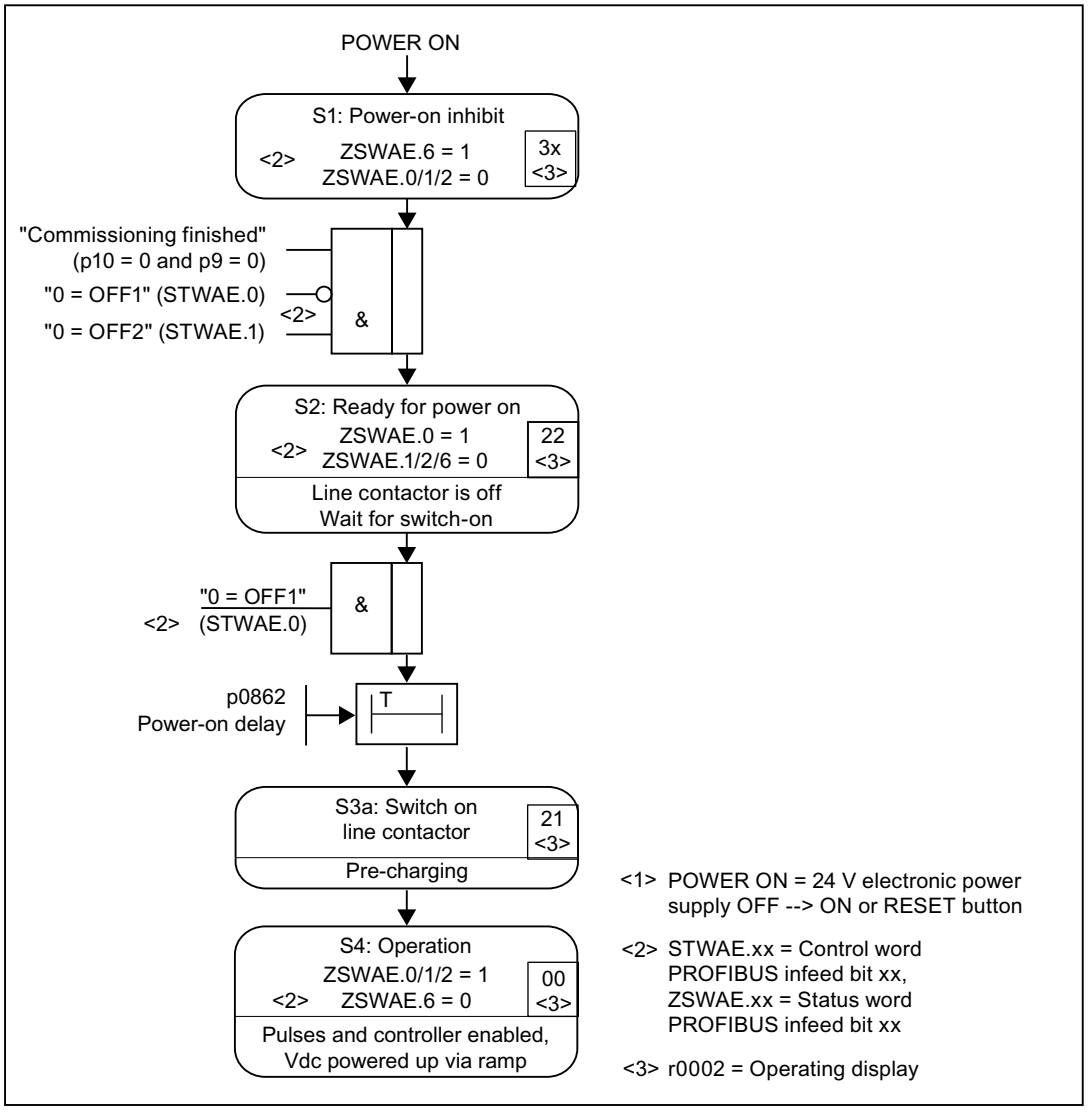


Figure 8-15 Flow diagram: Switching on the BLM

Flow diagram for switching on the Smart Line Module (SLM)

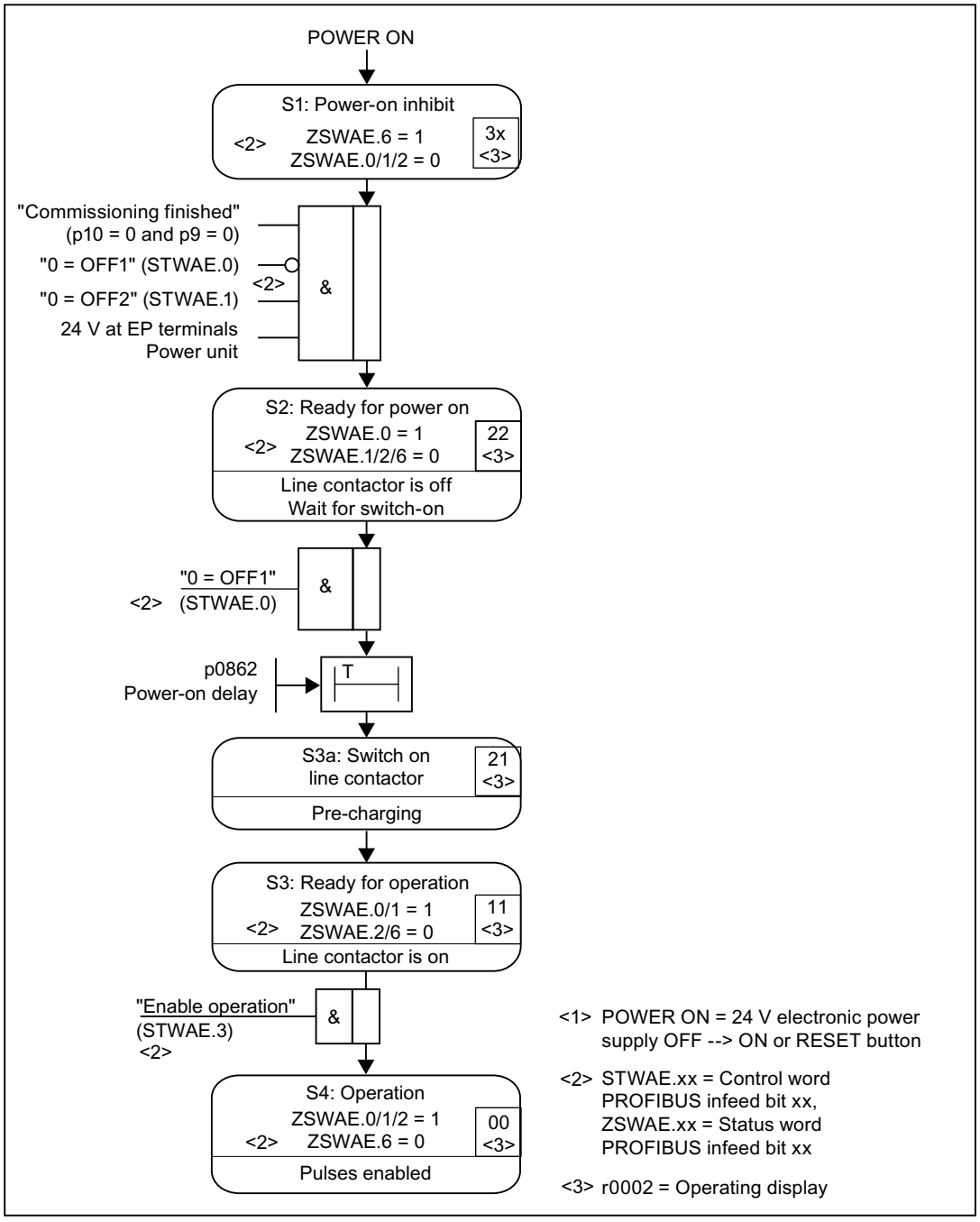


Figure 8-16 Flow diagram: Switching on the SLM

8.4.6.2 List of abbreviations / acronyms

Table 8-28 Abbreviations

Abbreviation	Meaning
ALM	Active Line Module
BLM	Basic Line Module
DRIVE-CLiQ	DRIVE Component Link with IQ
FB	Function block
IN	Input parameters
I/O	Input/output
LAD	Ladder diagram
LED	Light Emitting Diode
OUT	Output parameter
PLC	Programmable Logic Controller
PZD	Process data
SLM	Smart Line Module
STW	Control word
ZSW	Status word

8.5 Supplement to the CP 340 and CP 341 Modules

Preface

Contents of the function manual

This **document** is part of the **SIMOTION Programming References documentation package**.

This manual is a supplement to the following SIMATIC manuals:

- CP 340 Point-to-Point Communication, *Installation and Parameter Assignment*
- CP 341 Point-to-Point Communication, *Installation and Parameter Assignment*

These documents are included in the SIMOTION SCOUT scope of supply as electronic documentation!

This manual supplement will help you to integrate and start up the CP 340 and CP 341 communication processors in a SIMOTION system.

Differences in handling which result from the software architecture of a SIMOTION system as compared to the software architecture of a SIMATIC system will be described.

Function blocks

The function blocks for communication between the SIMOTION system and the CP 340 and CP 341 modules are part of the program library of the "SIMOTION SCOUT" engineering system.

Sections in this manual

The following chapters of this manual describe the function blocks (FBs) and data structures used in a SIMOTION system.

- General
  This chapter describes the differences and similarities in operation of the various CP modules.
- CP 340 function blocks
  This chapter describes the function blocks required for communication between a SIMOTION system and a CP 340.
- CP 341 function blocks
  This chapter describes the function blocks required for communication between a SIMOTION system and a CP 341.
- Alarm processing
  This chapter describes the differences in alarm processing in the SIMOTION system compared to the SIMATIC system.
- SIMATIC and SIMOTION Names
  This appendix contains a comparison of SIMATIC and SIMOTION names.
- The index allows you to locate information quickly.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.5.1 Fundamental safety instructions

8.5.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.5.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.5.2 Description

8.5.2.1 General

This chapter describes the general differences between SIMOTION and SIMATIC systems in terms of the operation of the CP 340 / CP 341 communications processors and in terms of data transfer.

Note

This manual is a supplement to SIMATIC manuals *CP 340 Point-to-Point Connection, Installation and Parameter Assignment/CP 341 Point-to-Point Connection, Installation and Parameter Assignment*.

These documents are shipped with SIMOTION SCOUT in electronic form!

The following software versions are required for the standard functions described in this documentation:

- SIMOTION SCOUT V4.2 or higher
- SIMOTION Kernel V4.2 or higher

8.5.2.2 Product description

The communications processor (CP) enables data to be exchanged between your SIMOTION system and another communications partner.

Function blocks are required for data exchange between the SIMOTION device and the communications processors. The function blocks for the SIMOTION system are described in this manual; these function blocks are handled differently than the function blocks for SIMATIC S7.

Functionality of the CP 340 / CP 341

The functionality of the function blocks and the CPs in a SIMOTION system is the same as in the SIMATIC S7 with the exception of special protocols for the CP 341.

The special protocols for "Modbus Slave", "Modbus Master" and "Data Highway" for the CP 341 are **not** currently supported by SIMOTION function blocks. For detailed information, see the "CP 340 function blocks" and "CP 341 function blocks" chapters.

Possible applications

In addition to the possible applications described in SIMATIC manuals *CP 340 Point-to-Point Connection, Installation and Parameter Assignment and CP 341 Point-to-Point Connection, Installation and Parameter Assignment*, these communications processors (CPs) can also be used in a SIMOTION system. The communications processors can be used as centralized modules (on the SIMOTION C2xx only) or as distributed modules (SIMOTION C2xx, SIMOTION P350 and SIMOTION D4xx).

More than one CP 340/CP 341 can be used on one SIMOTION device.

The figure below shows the connection of an ET 200M distributed I/O device with IM 153-1 and CP 340 or CP 341 to a SIMOTION device (e.g. SIMOTION C2xx).

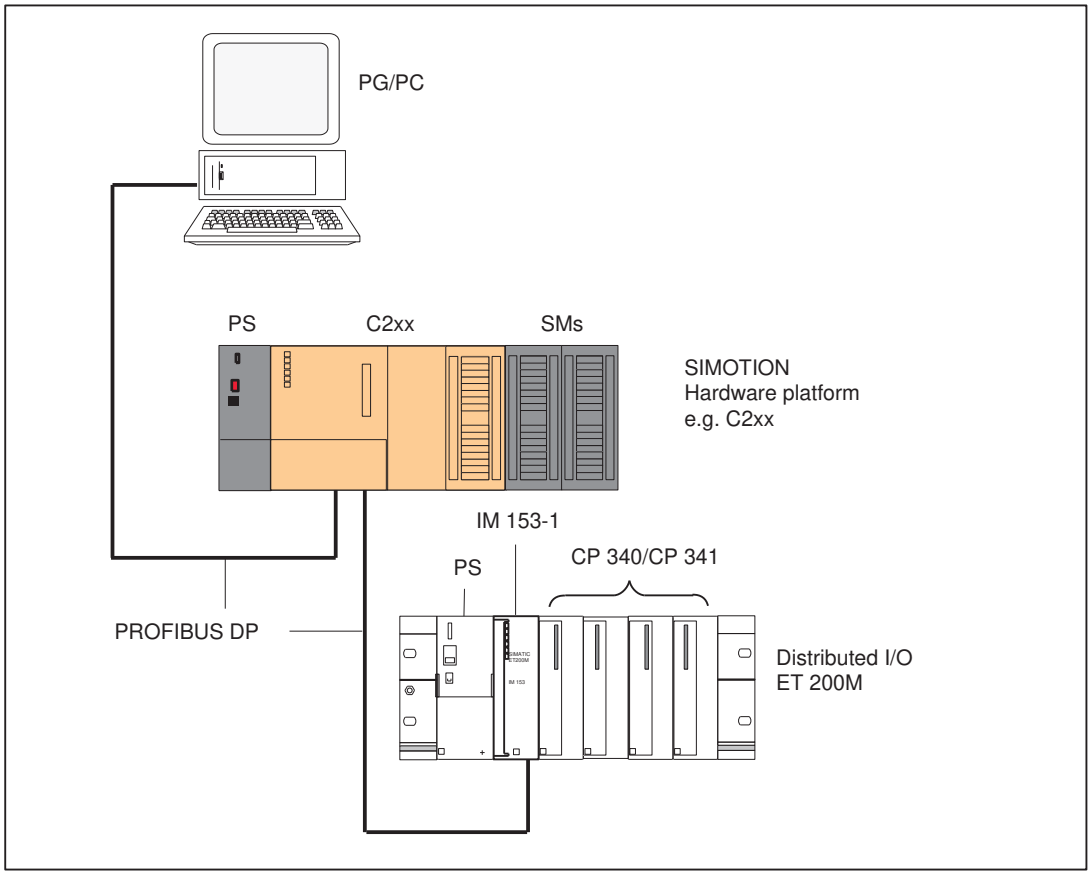


Figure 8-17 Connection of an ET 200M distributed I/O device with IM 153-1 and CP 340 or CP 341 to a SIMOTION C2xx (example of distributed application)

8.5.2.3 Setup and connection

Overview

The following sequence of operations is required to commission the CP 340/CP 341 and operate it under the control of the SIMOTION system:

Distributed application (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx)

1. Assemble and install the cables for the ET 200M distributed I/O device complete with power supply (PS), interface module (IM) and communications processor (CP).
2. Establish the PROFIBUS connection between the ET 200M and the SIMOTION device.
3. Set the PROFIBUS DP node address on the IM.
4. Switch on the terminating resistor at the first and last bus node.

Note

For steps 1 to 4, refer to the *ET 200M Distributed I/O* manual.

This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

5. For inserting the communications processors CP 340 or CP 341 into the SIMOTION project, see Chapter Integrating the communications processors in the SIMOTION project (Page 6136).
6. Assign parameters for the CP 340/CP 341 communications processors.
   The SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment and CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manuals contain a description of how to install the parameter assignment interface for the CP 340 and 341 and how to assign parameters for the communications processors.
7. For integrating function blocks into the SIMOTION project, see Chapter Integrating the function blocks in the user project (Page 6137).

Centralized application (SIMOTION C2xx only)

1. For information on planning the mechanical installation and preparing and mounting the SIMOTION components, refer to the *SIMOTION C2xx* operating instructions and the *SIMATIC S7-300 Automation System, Software Installation* manual.
   These documents are shipped with SIMOTION SCOUT in electronic form!
2. To continue, refer to steps 5 to 7 for distributed application.

8.5.2.4 Integrating the communications processors in the SIMOTION project

Requirement

The following requirements must be met when networking with PROFIBUS:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet (for distributed use only).

Note

For how to create a project and configure a PROFIBUS subnet, refer to the online help of *SIMOTION SCOUT*.

The following requirements must be met when with PROFINET:

1. You have created a project in SIMOTION SCOUT and have inserted and configured a rack with a PROFINET-compatible SIMOTION device in the hardware configuration.
2. You have configured a PROFINET IO System (for distributed use only).

Note

For how to create a project and configure a PROFINET IO system, refer to the online help of *SIMOTION SCOUT*.

Inserting the CP 340/CP 341 (distributed application)

The description below is an example of networking via PROFIBUS.

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.
4. Open the **PROFIBUS DP** folder and the **ET 200M** subfolder in the hardware catalog. Select, for example, the **IM 153-1 interface module** (Article No.: 6ES7 153-1AA03-0XB0 or a successor module).
5. Use a drag-and-drop operation to place the IM 153-1 I/O device on the PROFIBUS subnet of your project. The **Properties - PROFIBUS IM 153-1 Interface** dialog box opens. In this dialog box, select the address you set on the IM 153-1 (see *ET 200M Distributed I/O Device* manual) and confirm with **OK**.
   The selected IM 153-1 I/O device is inserted in the project.

6. The inserted I/O device must now be fitted with your project modules. To do this, open the **CP 300** subfolder below the selected I/O device in the hardware catalog and select the relevant **CP modules**.

   Note

   Diagnostic alarms are not enabled by default. Activate the alarms for each module in the **Properties** dialog box.

7. **Save** and **compile** your project.

8.5.2.5 Integrating the function blocks in the user project

Creating an instance of the FBs in the user project

The function blocks are part of the program library of the SIMOTION SCOUT engineering system. For working with the blocks, an instance has to be created in the user project for each function block used.

Example:

```
VAR_GLOBAL
...
  myInstCP340Send  : _CP340_send;  // create FB instance
  myInstCP341Send  : _CP341_send;  // create FB instance
...
END_VAR
```

Call (LAD representation)

The LAD representation of the individual function blocks can be found in the respective function block descriptions.

Example of an application

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

8.5.2.6 Creating I/O variables

Overview

Communication between the SIMOTION device and the CP 340 and CP 341 takes place by means of direct I/O access and data set transfer. For data set transfer, the module address is transferred to the FB as an input parameter. I/O variables are used to address the direct read/write access to the I/O.

You can freely assign the names of I/O variables in SIMOTION SCOUT. I/O variables must be specified as ARRAY [0..15] of BYTE. You assign the address settings in the hardware configuration to these I/O variables.

The names of the I/O inputs must be transferred to the function blocks as call parameters (**periIn**). The prepared data for the I/O outputs are provided by the FB as in/out parameters (**periOut**). The in/out parameter must be supplied with a variable of type ARRAY [0..15] of BYTE. After the block is called, this variable must be assigned to the I/O variables for the I/O outputs (see Chapter Calling the CP 340 function blocks (Page 6167)).

Note

The variable for supplying the in/out parameters must not be created as a temporary variable (VAR_TEMP or local variable of a function).

The following example shows how to assign the module addresses to the I/O variables in SIMOTION SCOUT.

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperipheralinputcp340	PIB 256		Array	16
2	⊞ myperipheraloutputcp340	PQB 256	☐	Array	16
3	⊞ myperipheralinputcp341	PIB 272		Array	16
4	⊞ myperipheraloutputcp341	PQB 272	☐	Array	16

Figure 8-18 Address assignment in SIMOTION SCOUT for two CP modules

Note

For additional information, see the following sources:

- *SIMOTION SCOUT* online help
- Programming Manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* programming manual
  - *SIMOTION MCC, Motion Control Chart* programming manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* programming manual

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation.

8.5.3 CP 340 function blocks

8.5.3.1 Overview of the function blocks of the CP 340

This chapter contains a description of all of the function blocks (FBs) and the data structure required for communication between a SIMOTION device and a CP 340.

The function blocks form the software interface between the SIMOTION device and the CPs. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- _CP340_send function block (Page 6139)
- _CP340_receive function block (Page 6143)
- _CP340_printer function block (Page 6147)
- _CP340_getV24Signals function block (Page 6164)
- _CP340_setV24Signals function block (Page 6166)

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6231) in the table "SIMOTION and SIMATIC CP 340 identifiers".

SIMOTION SCOUT contains all of the required FBs and the **Struct_CP340_printData** data structure (for **_CP340_printer** function block only) of the CP 340. The function blocks can be used to control one or more CP 340 modules.

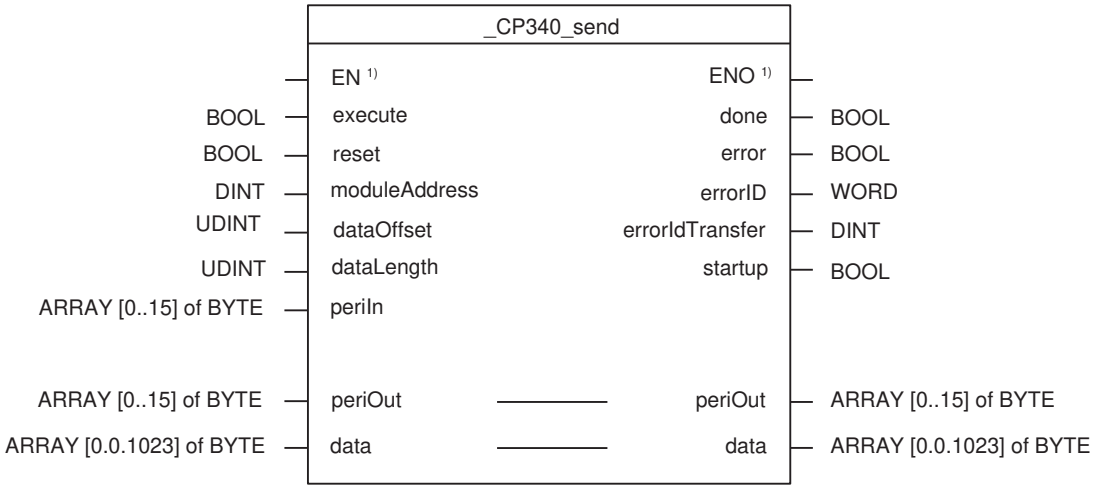
8.5.3.2 _CP340_send function block

Function

The **_CP340_send** function block enables you to send data from the **data** send array to a communications partner. You have 1,024 bytes available for this.

For the transfer, you can use the 3964 (R) protocol or ASCII driver.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-29 Parameters of the _CP340_send FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job on positive edge	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
dataOffset	IN	UDINT	Offset of the first element to be sent	Entered	Checked
dataLength	IN	UDINT	Number of elements to be sent	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP 2)	Checked and transferred to the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..1023] of BYTE	Send data array	Entered	Checked
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. 3)	Checked	Entered

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
errorIdTransfer	OUT	DINT	Error during data transfer between the CP and the SIMOTION device (detailed error diagnostics if 16#1E0F is present in the **errorID** parameter [4])	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 340.

[3] For error information, refer to the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 340"

[4] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

Signal sequence diagram of the _CP340_send FB

The following figure illustrates the behavior of the **done** and **error** parameters according to the input circuit of **execute** and **reset**.

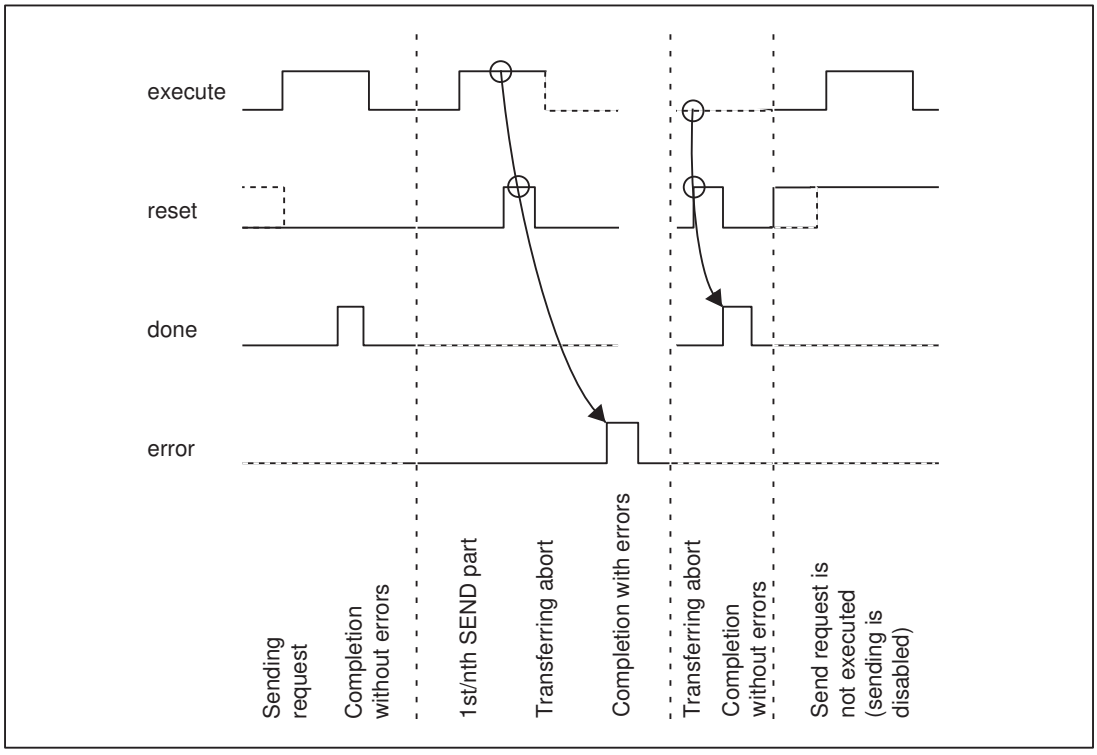


Figure 8-19 Signal sequence diagram of the _CP340_send FB

Note

The **execute** input is edge-triggered. The send job starts when there is a positive edge at the **execute** input.

Task integration (call)

The **_CP340_send** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

The SIMOTION device sends data to a communications partner

The **_CP340_send** function block transfers a data block that is specified by the following parameters to the CP 340:

- **data** corresponds to the data array containing the send data
- **dataOffset** corresponds to the array index containing the first send byte
- **dataLength** corresponds to the amount of data to be sent in bytes

The **_CP340_send** FB must be called repeatedly by a program. The send job can only be executed by cyclically calling the send FB.

A positive edge at the **execute** input initiates the transfer. A data transfer operation can run over several calls, depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP340_send** FB to its initial state. The send operation will remain disabled as long as the signal state at the **reset** parameter is "TRUE".

The **moduleAddress** parameter specifies the module address of the CP 340 being addressed.

Status and error display on the _CP340_send FB

The **done** output indicates that the job has been completed without errors. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/number is displayed in the **errorID** output (see "Parameters of the _CP340_send FB" table). If no errors have occurred, **errorID** has a value of 0. **Done** and **error**/**errorID** are also displayed for **reset** of the **_CP340_send** FB. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **done**, **error, errorID**, and **errorIdTransfer** parameters are available for one block call only.

Note

There is no parameter check for the **_CP340_send** function block. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode.

Before the CP 340 can process an initiated job following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP340_send** FB must be complete. Any jobs initiated in the meantime will not be lost. They are transferred to the CP 340 once the startup coordination has finished.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

Assignment in the data area

The data to be sent is transferred to the **_CP340_send** FB in the **data** parameter (VAR_IN_OUT) as ARRAY of BYTE. At the beginning of the send operation, the data is copied to a local variable of the FB and transferred from there to the CP 340.

Note

Once the data have been entered in the static memory of the send FB, you can modify the variable created in the **data** parameter. This does not affect the data to be sent.

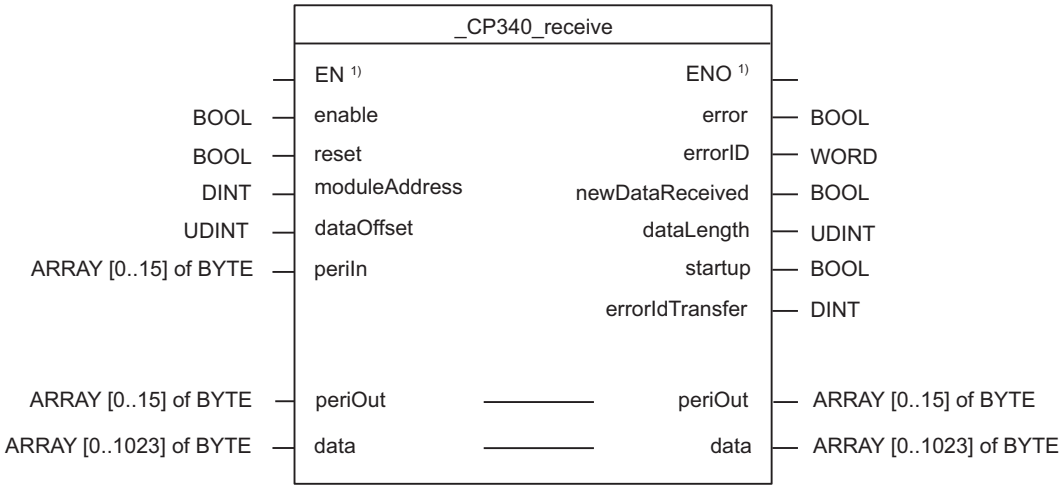
8.5.3.3 _CP340_receive function block

Function

The **_CP340_receive** function block enables you to receive data from a communications partner in the **data** receive field. You have 1,024 bytes available for this.

For the transfer, you can use the 3964 (R) protocol or ASCII driver.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-30 Parameters of the _CP340_receive FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Receive enable	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
dataOffset	IN	UDINT	Offset of the first element to be received	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP 2)	Checked and transferred to the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..1023] of BYTE	Receive data array	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.3)	Checked	Entered
newDataReceived	OUT	BOOL	Receive new data	Checked	Entered
dataLength	OUT	UDINT	Quantity of data received	Checked	Entered

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
startup	OUT	BOOL	Indicates CP startup	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer between the CP and the SIMOTION device (precise error diagnostics if 16#1E0F is present in the **errorID** parameter [4])	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 340.

[3] For error information, refer to the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 340".

[4] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

Signal sequence diagram of the _CP340_receive FB

The following figure illustrates the behavior of the **newDataReceived**, **dataLength**, and **error** parameters according to the input circuit of **enable** and **reset**.

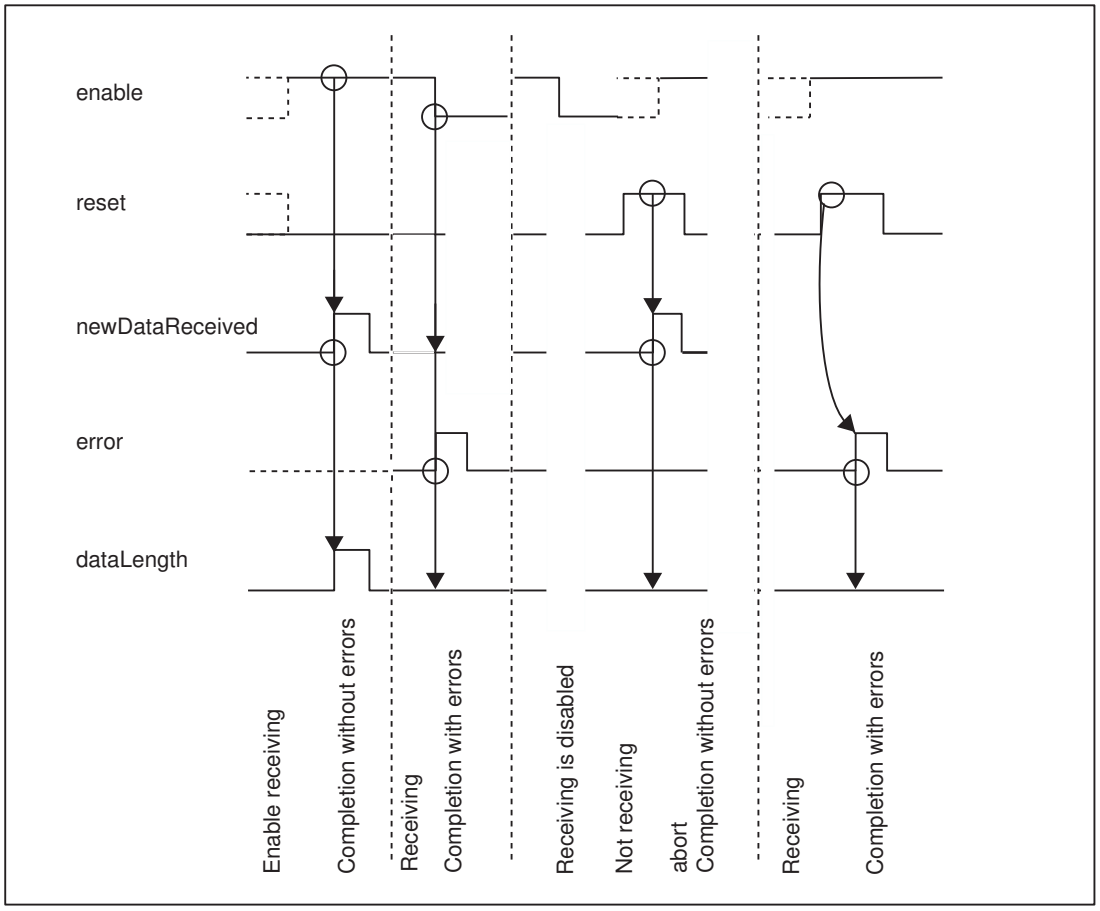


Figure 8-20 Signal sequence diagram of the _CP340_receive FB

Task integration (call)

The **_CP340_receive** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

SIMOTION device receives data from a communications partner

The **_CP340_receive** FB transfers a data block that is specified by the **data** and **dataOffset** parameters from the CP 340 to a SIMOTION device. The **_CP340_receive** function block must be called repeatedly by a program. The receive job can only be executed by cyclically calling the receive FB.

Receiving of data is enabled with static signal state "TRUE" in the **enable** parameter. An active data transfer can be canceled with signal state "FALSE" in the **enable** parameter. The canceled receive job is terminated with an error message (**errorID** output). The receive operation will

remain disabled as long as the signal state at the **enable** parameter is "FALSE". A data transfer operation can run over several calls, depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP340_receive** FB to its initial state. The receive operation will remain disabled as long as the signal state at the **reset** parameter is "TRUE".

The **moduleAddress** parameter specifies the module address of the CP 340 being addressed for the data set transfer.

Status and error display on the _CP340_receive FB

The **newDataReceived** output indicates that new data have been received without errors. The amount of data received is indicated in the **dataLength** parameter. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/number is displayed in the **errorID** output (see "Parameters of the _CP340_receive FB" table). If no error has occurred, **errorID** has the value "0". **newDataReceived** and **error**/**errorID** will also be output when the **_CP340_receive** FB is **reset**. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **newDataReceived**, **dataLength, error, errorID**, and **errorIdTransfer** parameters are available for one block passage only.

Note

There is no parameter check for the **_CP340_receive** function block. Incorrect parameterization of this block may cause the SIMOTION device to switch to STOP mode.

Before a job from the CP 340 can be received following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP SIMOTION startup mechanism of the **_CP340_receive**FB must be complete.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

Assignment in the data area

During the receive operation, the data to be received is stored temporarily in the FB. Once data transmission from the CP 340 to the SIMOTION device has been completed, the data is made available in the **data** parameter (VAR_IN_OUT) of the **_CP340_receive** FB.

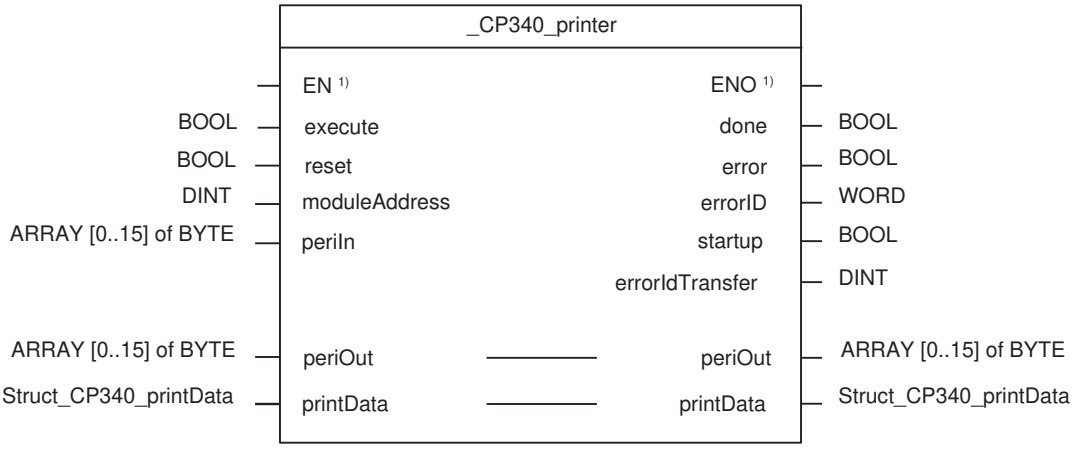
8.5.3.4 _CP340_printer function block

Description of the _CP340_printer FB

Function

The **_CP340_printer** function block is used to send data of type **Struct_CP340_printData** from the printer memory area to a serial printer. For example, the **_CP340_printer** function block might send a process message to the CP 340. The CP 340 prints out the process message on the connected printer.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-31 Parameters of the _CP340_printer FB

Name	P type 1)	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job on positive edge	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP 2)	Checked and transferred to the I/O variable for the I/O outputs	Entered
printData	IN/OUT	**Struct_CP340_printData**	Send data array (data to be printed)	Entered	Checked
Struct_CP340_printData (data structure) 5)					
variable		ARRAY[0..3] of Struct_CP340_dataRecord	Variable to be printed	Entered	Checked
format		ARRAY[0..150] of BYTE	Format string	Entered	Checked
Struct_CP340_dataRecord (data structure) 5)					
dataLength		UDINT	Quantity of data	Entered	Checked
data		ARRAY[0..31] of BYTE	Print data	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.[3]	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [4])	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 340.

[3] For error information, refer to the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 340".

[4] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

[5] See "Message text structure", print storage area structure example

Signal sequence diagram of the _CP340_printer FB

The following figure illustrates the behavior of the **done** and **error** parameters according to the input circuit of **execute** and **reset**.

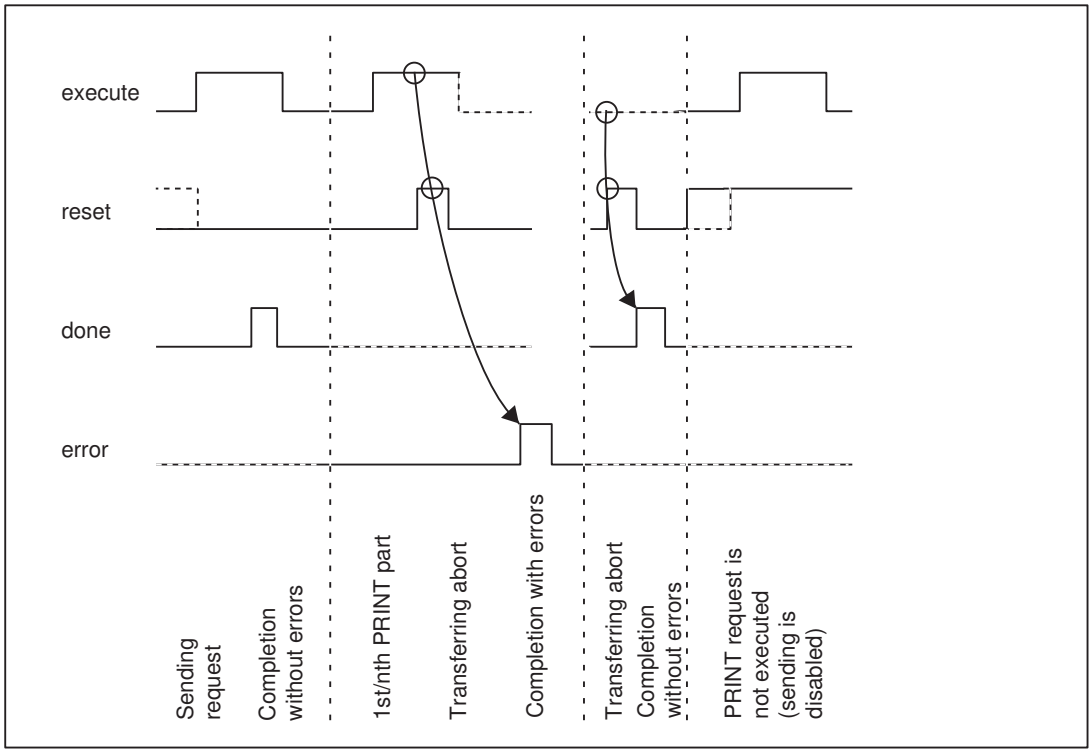


Figure 8-21 Signal sequence diagram of the _CP340_printer FB

Note

The **execute** input is edge-triggered. The send job starts when there is a positive edge at the **execute** input.

Task integration (call)

The **_CP340_printer** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

The **_CP340_printer** function block is called repeatedly by a program. The print job can only be executed by cyclically calling the print FB.

A positive edge at the **execute** input initiates the transfer of the message text. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP340_printer** FB to its initial state. The sending of print jobs will remain disabled as long as the signal state at the **reset** parameter is TRUE.

The **moduleAddress** parameter specifies the module address of the CP 340 being addressed for the data set transfer.

For call examples for the **_CP340_printer** FB, see Chapter CP 340 print call examples (Page 6160).

Status and error display on the _CP340_printer FB

The **done** output parameter indicates that the job has been completed without errors. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/ number is displayed in the **errorID** output parameter (see "Parameters of the CP340_printer FB" table). If no errors have occurred, **errorID** has a value of 0. **Done** and **error/errorID** are also displayed for **reset** of the **_CP340_printer** FB. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **done**, **error, errorID**, and **errorIdTransfer** parameters are available for one block call only.

Note

There is no parameter check for the **_CP340_printer** function block. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode. Before the CP 340 can process an initiated job following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP340_printer** FB must be complete. Any jobs initiated in the meantime will not be lost. They are transferred to the CP 340 once the startup coordination has finished.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

Assignment in the data area

The print data is transferred to the **_CP340_printer** FB as a data structure of type **Struct_CP340_printData** and copied to a local variable of the FB at the beginning of the print operation.

Structure of printer memory of type Struct_CP340_printData

The four variables to be printed and the format string must be entered in a variable with the following data type:

Example:

```
Struct_CP340_dataRecord     : STRUCT
     dataLength : UDINT;                        // Data quantity
     data       : ARRAY [0..31] of BYTE;        // Data field
END_STRUCT

Struct_CP340_printData    : STRUCT
     variable   : ARRAY [0..3] of Struct_CP340_dataRecord;  // 1st to 4th variable
     format     : ARRAY [0..150] of BYTE;                   // Format string
END_STRUCT
```

The first variable to be printed corresponds to the **variable [0]** element, the second variable to be printed corresponds to the **variable [1]** element, etc. The number of bytes to be printed per variable is limited to 32. The data for variable **i** must be placed in variable [**i-1**].data[0..31]. The number of bytes to be printed must be entered in the **variable [i-1].dataLength** element.

The format string corresponds to the **format** element. The format string must be structured as follows (refer to the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual):

- Specification of string length in **format [0]**
- Specification of individual characters in **format [1 to 150]**

Note

If the maximum length is exceeded, the print job is canceled and event number 16#1E41 is indicated at the **errorID** parameter output of the **_CP340_printer** FB.

Entering variables and message texts in the printer memory area

Before the data transfer to the CP 340 begins, the variable values to be printed must be entered **byte by byte and in the proper format** in the **variable[].data** element of the data structure of type **Struct_CP340_printData** (see item 2 in the example below). The number of bytes for each variable (variable length) must be assigned to the **variable.datalength** element (e.g. WORD type variable - variable.datalength:=2). An entry corresponding to the data type of the value must be made in the **format** element for each value entered in the **variable** element. (e.g. WORD type variable - %I). The total length of the entries in the **format** element must be assigned to the **format[0]** element.

You configure message texts with the CP 340 "point-to-point connection" parameter assignment interface. Once the hardware configuration has been downloaded to the SIMOTION device, the message texts are stored in the CP 340. The message texts that have been saved can be selected with corresponding entries in the **variable** and **format** elements.

Note

You can use supplemental function blocks (see Chapter supplemental function blocks (Page 6154)) to enter values into the printer memory area and to select message texts.

Example:

- Print message text no. 3 (stored in CP 340).
  Configured message text: "This is message text no. 3"

```
myPrintData.variable[0].datalength := 1;
myPrintData.variable[0].data[0]    := 3;   // Message text no. 3
```

```
myPrintData.format[0]     :=2;      // Format string length
myPrintData.format[1]     :=16#25;  // "%" Format specification for message
text
myPrintData.format[1]     :=16#4E   // "N" Format specification for message
text
```

- Print message text no. 4 with a WORD-type variable.

```
Configured message text   : "This is message text no. %I"
Printed text          : "This is message text no. 4"

myPrintData.variable[0].datalength := 1;
myPrintData.variable[0].data[0]    := 4;  // Message text no.4

myPrintData.variable[1].datalength := 2;  // 2 Byte data type WORD
myPrintData.variable[1].data[0]    := 0   // High - Byte
myPrintData.variable[1].data[1]    := 4   // Low - Byte
myPrintData.format[0]              := 4;  // Format string length
myPrintData.format[1]          := 16#25;  // ASCII code "%" format
specification
myPrintData.format[2]          := 16#4E;  // ASCII code "N" format
specification
                                          // for message text
myPrintData.format[3]          := 16#25;  // ASCII code "%" format
specification
myPrintData.format[4]          := 16#49;  // ASCII code "I" format
specification
                                          // for integer
```

Notes on handling

The format string entry in the "format" field must be hexadecimal.

Example:
% corresponds to 25 hex in the IBM character set,
N (message text output) corresponds to 4E hex in the IBM character set.
(see Hardware configuration > Character set)

Data types DATE, TIME, DATE_AND_TIME_OF_DAY and TIME_OF_DAY are not supported. The date information must be entered as a DWORD or WORD in the printer data structure.

Representation type "A" (German date format):
//datefrg:=4018 (01.01.2001) and 4199 (01.07.2001)
printData.variable[0].datalength:=2;
printData.variable[0].data[0]:=WORD_TO_BYTE(SHR(datefrg,8));
printData.variable[0].data[1]:=WORD_TO_BYTE(SHR(datefrg,0));

Representation type "F":
The value to be printed must be in floating point format (mantissa/exponent) (see call example 2, Chapter CP 340 print call examples (Page 6160))

Representation type "C":
If variable[ ].datalength:=1 in the printer data structure, the characters will be printed horizontally.
If variable[ ].datalength:=2 (3,4) in the printer data structure, the characters will be printed vertically.

Representation type "X":
For CP 340 RS232, product version E08 and higher, representation type "X" (binary) outputs the values correctly on a serial printer.

Disconnected printer
The communications link is not monitored for printers even if alarm generation is enabled in HW Config of STEP 7.

Example:

- A break in the connection between the printer and the CP 340 triggers neither an error nor a diagnostic alarm.
- Nor are they triggered if a print job is started but no printer is connected.

Note

The code in examples 1, 2, and 3 (see Chapter CP 340 print call examples (Page 6160)) can be transferred to the SIMOTION SCOUT editor with **Copy** and **Paste**.

supplemental function blocks

Function

Supplemental function blocks are provided for entering variables of various data types as well as for entering message texts into data structure **Struct_CP340_printData**. You enter the value of the variables **byte by byte** and **in the proper format** in the **variable** element of the printer memory area and, optionally, in the **format** element. When numbers are entered for message texts, one entry is made in each of the **format** and **variable** elements. The method of representation for the variables in printed text and the method of entry in the format string can be selected by means of parameters.

The following supplemental function blocks are available:

- **_CP340_realToPrintData**
  Entry of a number of data type REAL into data structure **Struct_CP340_printData**
- **_CP340_dwordToPrintData**
  Entry of a number of data type DWORD into data structure **Struct_CP340_printData**
- **_CP340_wordToPrintData**
  Entry of a number of data type WORD into data structure **Struct_CP340_printData**
- **_CP340_byteToPrintData**
  Entry of a number of data type BYTE into data structure **Struct_CP340_printData**
- **_CP340_dintToPrintData**
  Entry of a number of data type DINT into data structure **Struct_CP340_printData**

- **_CP340_intToPrintData**
  Entry of a number of data type INT into data structure **Struct_CP340_printData**
- **_CP340_printMsgText**
  Selection of message texts stored in the CP 340

Note

SINT and USINT data types can be entered into data structure **Struct_CP340_printData** with the **_CP340_intToPrintData** function block. Type conversion is implicit.

Parameter description

Table 8-32 _CP340_byteToPrintData, _CP340_wordToPrintData, _CP340_dwordToPrintData parameters

Name	P type [1)]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	BYTE/WORD/ DWORD	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1≤ **numVariable** ≤ 4	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_REPLACE_WITH_SIGN	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_REPLACE_WITHOUT_SIGN	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: unsigned integer		
		CP_REPLACE_BINARY	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: binary		
		CP_ADD_WITHOUT_SIGN	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: unsigned integer		
		CP_ADD_WITH_SIGN	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP340_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-33 _CP340_realToPrintData

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	REAL	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_REPLACE_WITHOUT_EXPONENT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: floating-point		
		CP_REPLACE_WITH_EXPONENT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: with exponent		
		CP_ADD_WITHOUT_EXPONENT	Entry is added in the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: floating-point		
		CP_ADD_WITH_EXPONENT	Entry is added in the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: with exponent		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP340_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-34 _CP340_dintToPrintData, _CP340_intToPrintData

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	INT/DINT	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_DEFAULT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_ADD_TO_STRING	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP340_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-35 _CP340_printMsgText

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
numMsgText	IN	USINT	Number of the message text (stored in the CP 340)	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_DEFAULT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_ADD_TO_STRING	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP340_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Task integration (call)

The supplemental blocks must be called in the **BackgroundTask** or the **TimerInterruptTask**.

For call examples for the **_CP340_printer** FB, see Chapter CP 340 print call examples (Page 6160).

Status and error indicators

The **done** output indicates that the job has been completed without errors. The **error** output indicates that an error has occurred.

The **done** and **error** parameters are available for one block call only.

CP 340 print call examples

Call example 1

```
UNIT E1CP340p;

INTERFACE
VAR_GLOBAL
  myPreparePrintData          : BOOL;               // Initiate prepare print request
  myRequestPrint              : BOOL;               // Initiate transfer to printer
  myReset                     : BOOL;               // Abort print
  myActualLevel               : REAL := 5.67;       // actual value "level"
  myModuleAddress_1           : DINT:= 256;         // address of 1st CP340 module
  myPrintData                 : Struct_CP340_printData; // instance of datastruct
  myFB_CP340_printMessageText : _CP340_printMsgText;    // instances of function blocks
  myFB_CP340_realToPrintData  : _CP340_realToPrintData; // instances of function blocks
  myFB_CP340_print            : _CP340_printer;         // instance of function block
  myOutputArrayCP340_1        : ARRAY[0..15] of BYTE;   // field for CP340 output data
END_VAR
PROGRAM Example_print_1;                            // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_1
// BackgroundTask program

// Example to print out messagetext 3 with one variable
// ("actualLevel") of type REAL: "The actual level (l) is: <actualLevel>"
// The message-text 3 must be specified in the hardwareconfiguration of CP340
// like this: "The actual level (l) is:"

// The following I/O-variable for CP340 module are required:
// peripheralInputCP340_1: input address of CP340 module; type Array; length 16
// peripheralOutputCP340_1: output address of CP340 module; type Array; length 16

// entry to printDatastruct myprintData
myFB_CP340_printMessageText ( execute           := myPreparePrintData,
                              printData         := myPrintData,
                              numMsgText        := 3,        // number of message text
                              numVariable       := 1,        // number of variable
                              entryFormatString := CP_DEFAULT);
```

```
myFB_CP340_realToprintData  ( execute           := myPreparePrintData,
                              printData         := myPrintData,
                              data              := myActualLevel,
                              numVariable       := 2,
                              entryFormatString := CP_ADD_WITH_EXPONENT );
// call instance of _CP340_printer
// use requestPrint to start datatransfer to serial printer

myFB_CP340_print      ( execute       := myRequestPrint,           // initiate request
                        reset         := myReset,                  // abort request
                        moduleAddress := myModuleAddress_1,        // module address
                        periIn        := myPeripheralinputcp340_1, // peripheral input
                        periOut       := myOutputArrayCP340_1,     // output data field
                        printData     := myPrintData);             // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP340_1 := myOutputArrayCP340_1;

END_PROGRAM //Example_print_1
END_IMPLEMENTATION
```

Call example 2

```
UNIT E2CP340p;

INTERFACE
VAR_GLOBAL
  myRequestPrint        : BOOL;                    // Initiate transfer to printer
  myReset               : BOOL;                    // abort print
  myPrintDword          : DWORD;                   // value to print out
  myModuleAddress_1     : DINT := 256;             // address of 1st CP340 module
  myPrintData           : Struct_CP340_printData;  // instance of datastruct
  myFB_CP340_print      : _CP340_printer;          // instance of function block
  myOutputArrayCP340_1  : ARRAY[0..15] OF BYTE;    // field for CP340 output data
END_VAR
PROGRAM Example_print_2;                           // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_2                            // BackgroundTask program
// example with notation "F" and variable of type DWORD
// (containing mantissa and exponent)
```

```
// The following I/O-variable for CP340 module are required:
// peripheralInputCP340_1: input address of CP340 module; type Array; length 16
// peripheralOutputCP340_1: output address of CP340 module; type Array; length 16

// formatstring (length 2 Byte, notation "F"):
myPrintData.format[0]  := 2 ;
myPrintData.format[1]  := 16#25 ;            // "%"
myPrintData.format[2]  := 16#46 ;            // "F"

// assignment for variable (type DWORD), to print out with notation "F"
myPrintDword := REAL_TO_DWORD(10000.0);  // variable (DWORD) with mantissa and exponent

// !!!
// ATTENTION! wrong example for this case is an assingnment with an integer value e.g.:
// myprintDword := 10000;        // because the format is WITHOUT mantissa and exponent
// !!!

// fill out printData interface manually with DWORD-variable
myPrintData.variable[0].dataLength := 4 ;    // 1st variable, lenth 4 Byte
myPrintData.variable[0].data[0] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,24)));
myPrintData.variable[0].data[1] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,16)));
myPrintData.variable[0].data[2] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,8)));
myPrintData.variable[0].data[3] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,0)));

// call instance of _CP340_printer
// use requestPrint to start datatransfer to serial printer
myFB_CP340_print ( execute        := myRequestPrint,            // initiate request
                   reset          := myReset,                   // abort request
                   moduleAddress  := myModuleAddress_1,         // module address
                   periIn         := myPeripheralinputcp340_1,  // peripheral input
                   periOut        := myOutputArrayCP340_1,      // output data field
                   printData      := myPrintData);              // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP340_1 := myOutputArrayCP340_1;

END_PROGRAM //Example_print_2
END_IMPLEMENTATION
```

Call example 3

```
UNIT E3CP340p;
```

```
INTERFACE
VAR_GLOBAL
  myRequestPrint              : BOOL;                   // Initiate transfer to printer
  myReset                     : BOOL;                   // Abort print
  myPrintReal1                : REAL;                   // 1st value
  myPrintReal2                : REAL;                   // 2nd value
  myPrintReal3                : REAL;                   // 3rd value
  myModuleAddress_1           : DINT := 256;            // address of 1st CP340 module
  myPrintData                 : Struct_CP340_printData; // instance of datastruct
  myFB_CP340_print            : _CP340_printer;         // instance of function block
  myFB_CP340_realToprintData1 : _CP340_realToPrintData; // instances of function blocks
  myFB_CP340_realToprintData2 : _CP340_realToPrintData; // instances of function blocks
  myFB_CP340_realToprintData3 : _CP340_realToPrintData; // instances of function blocks
  myOutputArrayCP340_1        : ARRAY[0..15] OF BYTE;
END_VAR
PROGRAM Example_print_3;                                // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_3                                 // BackgroundTask program
// The following example program demonstrates usage of _CP340_realToPrintData()

// The following I/O-variable for CP340 module are required:
// peripheralInputCP340_1: input address of CP340 module; type Array; length 16
// peripheralOutputCP340_1: output address of CP340 module; type Array; length 16

// preset variables with user-values
myPrintReal1 := 1.11; myPrintReal2 := 2.22; myPrintReal3 := 3.33;
```

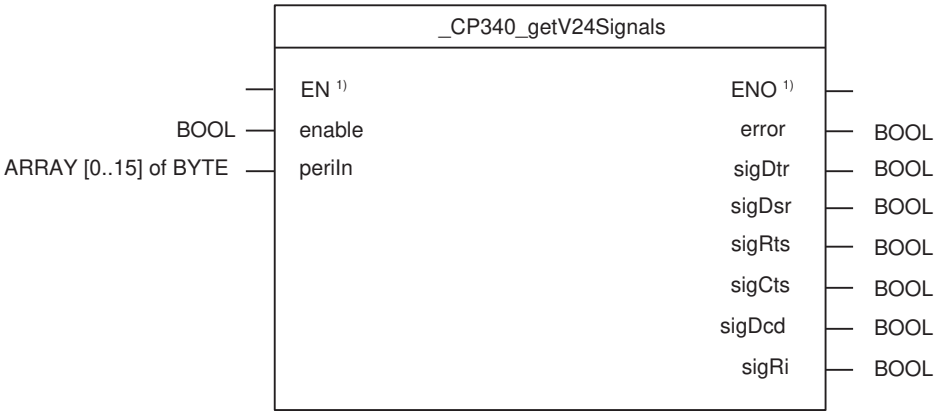
```
// write variables (type REAL) to printDatastruct with _CP340_realToPrintData
myFB_CP340_realToprintData1 (execute           := TRUE,
                             data              := myPrintReal1,        // 1st variable
                             numVariable       := 1,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);
myFB_CP340_realToprintData2 (execute           := TRUE,
                             data              := myPrintReal2;        // 2nd variable
                             numVariable       := 2,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);
myFB_CP340_realToprintData3 (execute           := TRUE,
                             data              := myPrintReal3,        // 3rd variable
                             numVariable       := 3,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);

// call instance of _CP340_printer, "requestPrint" starts data transfer
// to serial printer
myFB_CP340_print (  execute        := myRequestPrint,            // initiate request
                    reset          := myReset,                   // abort request
                    moduleAddress  := myModuleAddress_1,         // module address
                    periIn         := myPeripheralinputcp340_1,  // peripheral input
                    periOut        := myOutputArrayCP340_1,      // output data field
                    printData      := myPrintData);              // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP340_1 := myOutputArrayCP340_1;

END_PROGRAM                       //Example_print_3
END_IMPLEMENTATION
```

8.5.3.5 _CP340_getV24Signals function block

Function

The **_CP340_getV24Signals** function block reads the RS-232-C accompanying signals from the CP 340 and makes them available to the user in the block parameters. The functionality of the **_CP340_getV24Signals** FB can only be used if a parameterized ASCII driver is specified.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-36 Parameters of the _CP340_getV24Signals FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Block enable	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
error	OUT	BOOL	Job completed with errors	Checked	Entered
sigDtr	OUT	BOOL	Data terminal ready	Checked	Entered
sigDsr	OUT	BOOL	Data set ready	Checked	Entered
sigRts	OUT	BOOL	Request to send	Checked	Entered
sigCts	OUT	BOOL	Clear to send	Checked	Entered
sigDcd	OUT	BOOL	Data carrier detected	Checked	Entered
sigRi	OUT	BOOL	Ring Indicator	Checked	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Task integration (call)

The **_CP340_getV24Signals** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. For runtime reasons, calling the FB in the **IPOSynchronousTask** is not recommended.
The RS 232C accompanying signals will be updated with each FB call (cyclical polling). The CP 340 updates the status of the inputs/outputs in a time base of 20 ms.

Note

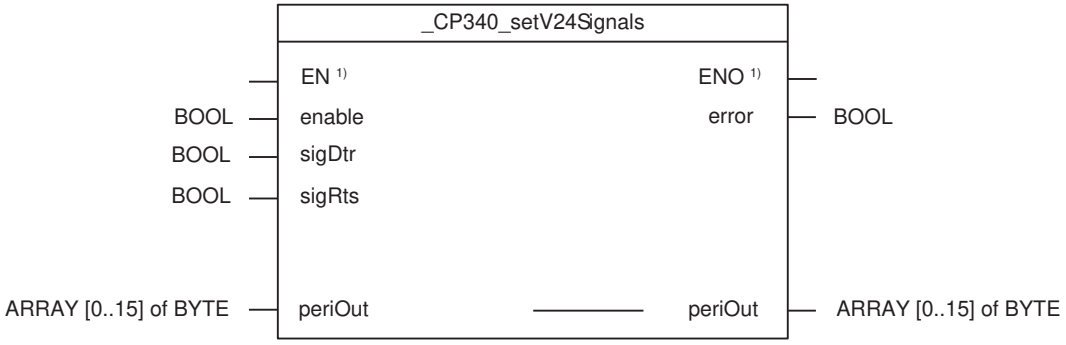
A minimum pulse duration is necessary to detect a signal change. Determining factors are the cycle time (SIMOTION device), the update time on the CP 340, and the response time of the communications partner.

8.5.3.6 _CP340_setV24Signals function block

Function

The **_CP340_setV24Signals** function block can be used to set or reset RS 232C accompanying signals. The functionality of the **_CP340_setV24Signals** FB can only be used if a parameterized ASCII driver is specified.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-37 Parameters of the _CP340_setV24Signals FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Block enable	Entered	Checked
sigDtr	IN	BOOL	Data terminal ready	Entered	Checked
sigRts	IN	BOOL	Request to send	Entered	Checked

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP [2]	Checked and transferred to the I/O variable for the I/O outputs	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 340.

Task integration (call)

The **_CP340_setV24Signals** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

8.5.3.7 Calling the CP 340 function blocks

In order to be able to work with the function blocks in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for the **_CP340_send** FB).
2. Create a variable for the data structure (for FB **_CP340_printer** only).
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the FB are accessed with <instance name of FB>. <name of the output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variable created in step 3.

Note

The CP 340 call example is an extract from the supplied E_CP340 application example, which is included on the "SIMOTION Utilities & Applications" CD-ROM.

If you wish to control multiple CP 340 devices, you must create a new variable for the data structure and FB instances with a new name for each CP 340 you implement.

Call example for CP 340

```
UNIT E_CP340;


INTERFACE
```

```
  VAR_GLOBAL
   myExecSendCP340      : BOOL;                    // Trigger send task
   myModuleAddrCP340    : DINT:=256;               // module address
   mySendDataArrayCP340 : ARRAY [0..1023] OF BYTE; // Send data array 1024 bytes
   myInstCP340Send      : _CP340_send;             // Create instance of FB                    (1)
  END_VAR

  PROGRAM ExampleCP340;          // Program in BackgroundTask

END_INTERFACE

IMPLEMENTATION
  VAR_GLOBAL
   MyResetSend          : BOOL;                    // Cancel send order
  END_VAR

  VAR
   MyCPOutputArray      : ARRAY [0..15] OF BYTE;   // Array for CP output data                 (3)
  END_VAR

  VAR_TEMP
   MyDataLengthSend     : UDINT;             // Length of data to be sent
   MyDataOffsetSend     : UDINT;             // Offset of first byte to be sent
  END_VAR

   // CALL FB INSTANCE TO SEND                                                                 (4)
   myInstCP340Send (                                                                           (5)
          execute       := myExecSendCP340,         // Trigger order
          reset         := myResetSend,             // Order cancellation
          moduleAddress := myModuleAddrCP340,       // Module address
          dataOffset    := myDataOffsetSend,        // Data offset
          dataLength    := myDataLengthSend,        // Number of data to be sent
          periIn        := myPeripheralInputCP340,  // I/O variable of I/O inputs
          periOut       := myCPOutputArray,         // Output data array
          data          := mySendDataArrayCP340     // Send data array
          );

   myStateStartUpCP340  := myInstCP340Send.startUp; // Module start-up status                (6)

   // TRANSFER DATA TO CP340
   myPeripheralOutputCP340 := myCPOutputArray;  // Assign array for CP output data

END_PROGRAM                                     // ExampleCP340                             (7)
```

```
END_IMPLEMENTATION
```

Note

The PROGRAM "ExampleCP340" must be assigned in the execution system.

8.5.3.8 Data consistency

When sending data

Once a job is initiated by a positive edge at the **execute** input, the data to be sent is copied to the static memory area of the send FB. This means that once the FB call has ended, the send data array can be written to again for the next send request on a positive edge. The data are retained as consistent data within the send FB.

Note

During the copy operation to the static memory area of the FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

When receiving data

Once the receive request is complete, the data are copied over to the receive buffer in a block from the static memory area of the receive FB. This means that once the FB call has ended, either all data are entered in the receive buffer (**newDataReceived** = TRUE) or no data are entered in the receive buffer (**newDataReceived** = FALSE).

Note

During the copy operation from the static memory area of the receive FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

When printing

The data in the **printData** parameter in the static memory area of the FB are copied when a positive edge occurs at the **execute** input of the **_CP340_printer** FB. This means that once the FB call has ended, you can write to the variable and the format string for the next print request.

8.5.3.9 Application Examples

sending and receiving with CP 340

Function

This example shows how to:

- use the **_CP340_send** function block to send data from the Send data array to a communications partner.
- use the **_CP340_receive** function block to receive data in the receive data array.

In the example program, the CP 340 is used both as sender and as receiver. This requires the jumpering of the send and receive lines (PIN 2 and PIN 3 on the RS232 interface) and the "ASCII" setting in the parameter assignment tool. The **_CP340_send** FB is used to transfer the send data to the CP module. This sends the data using the RS232 interface. The jumpered send and receive line means that the data to be sent is read immediately by the CP module. The **_CP340_receive** FB reads the received data from the CP module and copies these data to the receive data array.

This example requires proper installation of the parameter assignment tool, as described in the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual.

Hardware platform

The application example is available for various SIMOTION hardware platforms. You must adapt the example for centralized applications with SIMOTION C.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station with a CP 340 module.
4. Double-click the **CP 340** to open the **Properties** dialog box for this module. Click the **Parameter** button to launch the parameter assignment tool of the CP 340 module.
5. The "ASCII" protocol must be selected in the protocol selection box.
   The standard settings of the ASCII protocol suffices for the application example.
6. Jumper the send and receive line (PIN 2/PIN 3) of the RS232 interface.
7. Apply your settings in the parameter assignment interface with the **File > Save** menu command, and close the interface with the **File > Exit** menu command. Close the Properties window for the CP 340 module by clicking the **OK** button.

8. Save the hardware configuration with the **Station > Save and compile** menu command.
9. Download the hardware configuration with the **Target system > Download to module** menu command.
   The red "SF" LED on the IM 153 turns on and then off if the assigned module parameters have been downloaded without errors.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (e.g. PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (e.g. PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is included in the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and module address in the program (myModuleAddrCP340).
4. **Save** and **compile** the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Sequence of the application example

Table 8-38 Input icons used

Symbol	Data type	Designation
mySelectPointToPointCP340	BOOL	Select point-to-point connection
myExecSendCP340	BOOL	Start send job
mySendOrder1	BOOL	Select send: Job 1
mySendOrder2	BOOL	Select send: Job 2
mySendOrder3	BOOL	Select send: Job 3
myResetSend	BOOL	Cancel send job
myEnableToReceive	BOOL	Receive enable
myReceiveOrder1	BOOL	Select receive: Job 1
myReceiveOrder2	BOOL	Select receive: Job 2
myReceiveOrder3	BOOL	Select receive: Job 3
myResetReceive	BOOL	Cancel receive job
myModuleAddrCP340	DINT	CP 340 module address, default 256

Symbol	Data type	Designation
mySendDataArrayCP340	ARRAY[0..1023] of BYTE	Send data array
myReceiveDataArrayCP340	ARRAY[0..1023] of BYTE	Receive data array

Table 8-39 Output symbols used

Symbol	Data type	Designation
mySendDone	BOOL	Send: Completed
mySendError	BOOL	Send: Error display
mySendErrorNumber	WORD	Send: Error status
mySendTransErrorNumber	DINT	Send: Error status transfer
myNewDataReceived	BOOL	Receive: New data have been received
myReceiveError	BOOL	Receive: Error display
myReceiveErrorNumber	WORD	Receive: Error status
myReceiveTransErrorNumber	DINT	Receive: Error status transfer
myStateStartupCP340	BOOL	CP 340 startup status FALSE = startup completed
myDiagnosticAlarm	BOOL	TRUE = diagnostic alarm present on the CP 340
myProcessAlarm	BOOL	TRUE = process alarm present on the CP340
myAlarmInterrupt	UDINT	Type of the alarm (process, diagnostic alarm)
myLogBaseAddrIn	DINT	Module address
myLogBaseAddrOut	DINT	
myLogAddress	DINT	Diagnostic address
myAlarmDetails	DWORD	Alarm information

Note

You can either monitor and control the input and output variables used in the programming example in the INTERFACE area of the unit (under VAR_GLOBAL) using the symbol browser, or you can assign real inputs and outputs to the input and output variables in your user program.

For the "point-to-point connection" application example, set the "mySelectPointToPointCP340" input to "TRUE". This requires that the "E_CP340p" element be selected in the "PROGRAMS" folder in the project navigator. This will call function blocks contained in the application example.

Receiving data:

To receive data, you must set the "myEnableToReceive" variable to "TRUE" (static signal). If receive jobs 1 and 3 are enabled ("myReceiveOrder1" = TRUE and "myReceiveOrder3" = TRUE), the data is stored in the "myReceiveDataArrayCP340" data array starting with the "myReceiveDataArrayCP340[0]" array element (data offset is 0). If job 2 is enabled ("myReceiveOrder2" = TRUE), the data is stored in the "receiveDataArray" data array starting with the "myReceiveDataArrayCP340[20]" array element (data offset is 20).
If "myNewDataReceived" = TRUE, this indicates that new data has been received. This signal is present for one cycle only.
If an error occurred during the transfer ("myReceiveError" = TRUE), the error code is stored in the "myReceiveErrorNumber" variable. If error code 16#1E0F is present in "myReceiveErrorNumber", an error occurred during the data transfer. The transfer error code is stored in the

"myReceiveTransErrorNumber" variable. The error signals are deleted when you set input "myResetReceive" = TRUE.

Sending data:

You can use the "mySendOrder1", "mySendOrder2" and "mySendOrder3" inputs to select between three send jobs:

- Job 1 sends 10 bytes of data from the "mySendDataArrayCP340" data array from array element "mySendDataArrayCP340[0]" to "mySendDataArrayCP340[9]"
- Job 2 sends 20 bytes of data from the "mySendDataArrayCP340" data array from array element "mySendDataArrayCP340[20]" to "mySendDataArrayCP340[39]"
- Job 3 sends 1024 bytes of data from the "mySendDataArrayCP340" data array.

The data is sent to the communications partner if the "myExecSendCP340" input detects a signal change from "FALSE" to "TRUE" (positive edge).
If output signal "mySendDone" = TRUE, the send job has been completed. A new job can be sent if the "myExecSendCP340" input signal detects another signal change from FALSE to TRUE. If an error occurred during the transfer ("mySendError" = TRUE), the error code is stored in the "mySendErrorNumber" variable. If error code 16#1E0F is present in "mySendErrorNumber" , an error occurred during the data transfer. The transfer error code is stored in the "mySendTransErrorNumber" variable. The error signals are deleted when you set input "myExecSendCP340" = FALSE.

When the signal state at the "myResetSend" or "myResetReceive" input is set to "TRUE", the send job or receive job is canceled, respectively. If the signal state remains "TRUE", sending and receiving of data is disabled.

Note

Proper data transfer can be observed as follows:

- The "TxD" and "RxD" LEDs on the CP module illuminate.
- Output parameter (_CP_send FBs) **done** = TRUE or **NewDataReceived** = TRUE

Printing with CP 340

Function

The example shows how to use the **_CP340_printer** function block to send **Struct_CP340_printData** type data from the print memory area to a serial printer. For more information, see Chapter _CP340_printer function block (Page 6147).

In the example program, text examples with and without variables, date specifications, and line and page feeds are output to the printer by means of the CP 340.

This example requires proper installation of the parameter assignment tool, as described in the SIMATIC *CP 340 Point-to-Point Connection, Installation and Parameter Assignment* manual.

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station with a CP 340 module.
4. Double-click the **CP 340** to open the **Properties** dialog box for this module. Click the **Parameter** button to launch the parameter assignment tool of the CP 340 module.
5. The "PRINTER" protocol must be selected in the protocol selection box.
6. Open the protocol properties by selecting the **Edit > Go to protocol** menu command and the **Edit > Open object** menu command.
7. Select the following setting in the **Protocol** dialog box (all other settings remain unchanged):
   - **Baud rate: 9600 bits/s**
   - **Data flow control: none**
   - **Data bits: 8**
   - **Stop bits: 1**
   - **Parity: even**

   Confirm your selection with **OK**.
8. To edit the message texts, double-click the **Messages** icon to open the relevant dialog box. In the next dialog box, click the **SDB** button.
9. Under "Edit Message", enter a number on the left and the text on the right, as follows:
   - 1 System level log
   - 2 Date:
   - 3 Level %I I reached at %Z (time)

   Confirm your selection with **OK**.
10. To edit the page layout, double-click the **Page Layout** icon to open the relevant dialog box.

11. Select the following settings in this dialog box:
    - **Left-hand margin: 3**
    - **Lines per page: 20**
    - **Separators / end of line: CR LF**
    - **Header lines: Example: Printing with the CP 340 for SIMOTION**
    - **Page footers: page %P**

    Confirm your selection with **OK**.
12. The default settings for the "IBM" character set remain selected under "Font" and "Control Characters".
13. Apply your settings in the parameter assignment interface with the **File > Save** menu command, and close the interface with the **File > Exit** menu command. Click **OK** to close the Properties window for the CP 340 module.
14. Save the hardware configuration with the **Station > Save and compile** menu command.
15. Download the hardware configuration with the **Target system > Download to module** menu command.
    The red "SF" LED on the IM 153 turns on and then off if the assigned module parameters have been downloaded without errors.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (e.g. PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (e.g. PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example
2. Check the axis configuration: PROFIBUS DP addresses
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and the module addresses in the program (myModuleAddr_1)
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Sequence of the application example

Further steps for executing the example are performed in the symbol browser using the "myPrintSelect" variable. This requires that the "E_CP340p" element be selected in the "PROGRAMS" folder in the project navigator.

The program resets the "myPrintSelect" variable autonomously; this can be seen in the "Status Value" column in the symbol browser.

This application example shows you how to use the _FB_CP340.print function block to send data from the send data array to a serial printer.

The following texts can be printed with the example program:

1. Print new page with header:
   "Example: Printing with the CP 340 for SIMOTION"
2. Print new line and five line feeds
3. Print message text without variables and two line feeds:
   "System level log"
4. Message text, two tabs and WORD variable for German date format.
   "Date:"
   "Date: 01.01.2001"
5. Message text with time of day and quantity (DWORD variable and DINT variable) and six line feeds
   "Level %I I was reached at %Z (time)"
   "Level 3000 I was reached at 08:45:04.018 (time)"
6. Print footer with conversion instruction for page number
   "Page: %P"
   "Page: 1"

To print out all the lines from the example, select the check box for "myPrintSelect" and set ALL_PRINT in the "Control Value" column in the symbol browser.

Click "Immediate control" to assign the value ALL_PRINT to the variable and to print the example.

Other settings can be assigned to the "myPrintSelect" variable:

"DATE_PRINT"	Output "Date: 01.01.2001"
"REPORT_TEXT_PRINT"	Output "Level 3000 reached at 08:45:04.018"
"NO_PRINT"	No output made

The date, time, and level quantity can be changed using the following variables:

- myDateGermany
- myTimeValue,
- myQuantity

Utilized control variables

Table 8-40 Utilized control variables

Symbol	Data type	Initial value	Designation
myPrintSelect	EnumPrintSelect	NO_PRINT	Selection of printer output
myDateGermany	WORD	4018	Date: 01.01.2001
myTimeValue	DWORD	31504018	Time: 08:45:04.018
myQuantity	DINT	3000	Quantity: 3000
myModuleAddr_1	DINT	256	Module address

Pending errors are indicated by the following variables:

Table 8-41 Variables used for displaying errors

Symbol	Data type	Initial value	Designation
myDiagnosticAlarm	BOOL		Diagnostic interrupt
myProcessAlarm	BOOL		Process interrupt
myErrorNumberprinter	WORD		Error code
myTransErrorNumberprinter	DINT		Transfer error number
myStopPrinter	BOOL	FALSE	Cancel print

Note

The active transfer job can be canceled by setting the "myStopPrinter" variable to "TRUE". The **_CP340_printer** FB is reset to its initial state.

The sending of print jobs will remain disabled as long as the signal state of the "myStopPrinter" variable is "TRUE".

8.5.4 CP 341 function blocks

8.5.4.1 Overview of the function blocks of the CP 341

This chapter contains a description of all of the function blocks (FBs) and the data structure required for communication between a SIMOTION device and a CP 341.

The function blocks form the software interface between the SIMOTION device and the CPs. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- _CP341_send function block (Page 6178)
- _CP341_receive function block (Page 6190)
- _CP341_printer function block (Page 6200)

- _CP341_getV24Signals function block (Page 6217)
- _CP341_setV24Signals function block (Page 6219)

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6231) in the table "SIMOTION and SIMATIC identifiers CP 341".

SIMOTION SCOUT contains all of the required function blocks and the **Struct_CP341_CI512Data** data structure (for RK 512 computer link only) of the CP 341. The function blocks can be used to control one or more CP 341 modules.

8.5.4.2 _CP341_send function block

Description of the _CP341_send FB

Function

The **_CP341_send** function block enables you to send data from the send data array to a communications partner (3964(R) protocol, RK 512 protocol, ASCII driver) or to fetch data from a communications partner and store it there (RK 512). You have 4,096 bytes available for this.

Task integration (call)

The **_CP341_send** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

Application with 3964(R) protocol or ASCII driver

The SIMOTION device sends data to a communications partner

The **_CP341_send** function block transfers a data block that is specified by the following parameters to the CP 341:

- **data** corresponds to the data array containing the send data
- **dataOffset** corresponds to the array index containing the first send byte
- **dataLength** corresponds to the amount of data to be sent in bytes

The **_CP341_send** function block is called repeatedly by a program. The send job can only be executed by cyclically calling the send FB.

A positive edge at the **execute** input initiates the transfer. A data transfer operation can run over several calls, depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP341_send** FB to its initial state. The send operation will remain disabled as long as the signal state at the **reset** parameter is "TRUE".

The **moduleAddress** parameter specifies the module address of the CP 341 being addressed.

Status and error display on the _CP341_send FB

The **done** output indicates that the job has been completed without errors. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/number is displayed in the **errorID** output (see "Parameters of the _CP341_send FB" table (3964 (R) procedure or ASCII driver application)"). If no errors have occurred, **errorID** has a value of 0. **Done** and **error**/**errorID** are also displayed on **reset** of the **_CP341_send** function block. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

Parameters **done**, **error,errorID**, and **errorIdTransfer** are present for one block call only.

Note

There is no parameter check for the **_CP341_send** function block. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode.

Before the CP 341 can process an initiated job following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP341_send** FB must be complete. Any jobs initiated in the meantime will not be lost. They are transferred to the CP 341 once the startup coordination has finished.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

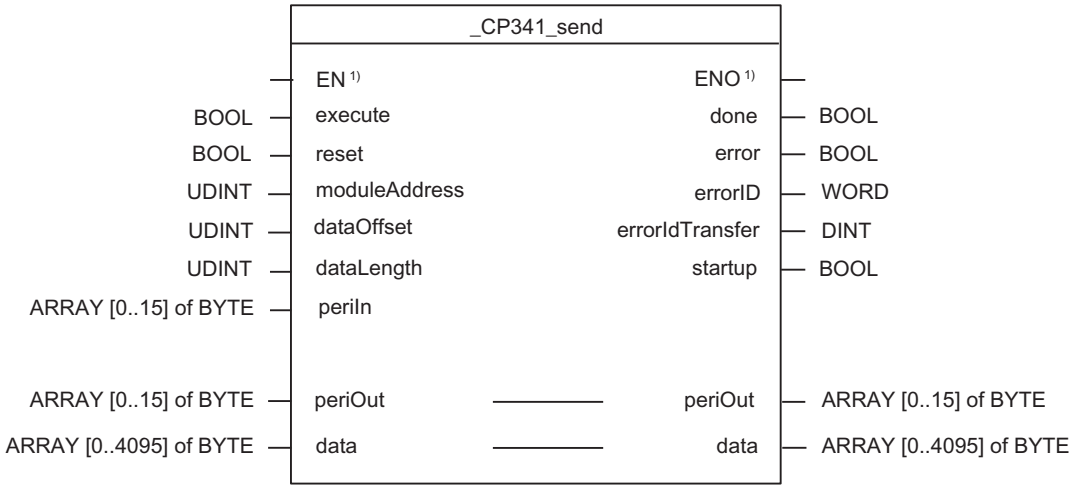
Assignment in the data area

The data to be sent are transferred to the **_CP341_send** FB in the **data** parameter (VAR_IN_OUT) as ARRAY of BYTE. At the beginning of the send operation, the data is copied to the FB and transferred from there to the CP 341.

Note

Once the data have been entered in the static memory of the send FB, you can modify the variable created in the **data** parameter. This does not affect the data to be sent.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-42 Parameters of the _CP341_send FB (application with 3964(R) protocol or ASCII driver)

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job on positive edge	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
dataOffset	IN	UDINT	First element to be sent	Entered	Checked
dataLength 2)	IN	UDINT	Number of elements to be sent	Entered	Checked
periIn	IN	ARRAY [0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY [0..15] of BYTE	Prepared FB data for the I/O outputs of the CP 3)	Checked and transferred to the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..4095] of BYTE	Send data array	Entered and checked	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. 4)	Checked	Entered

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [5])	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] If an application uses earlier CP 341 versions with a maximum telegram length of 1 kB, only the maximum value of 1,024 can be set at the **dataLength** input parameter.

[3] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY [0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

[4] For error information, refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 341".

[5] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

Signal sequence diagram of the _CP341_send FB

The following figure illustrates the behavior of the **done** and **error** parameters according to the input circuit of **execute** and **reset**.

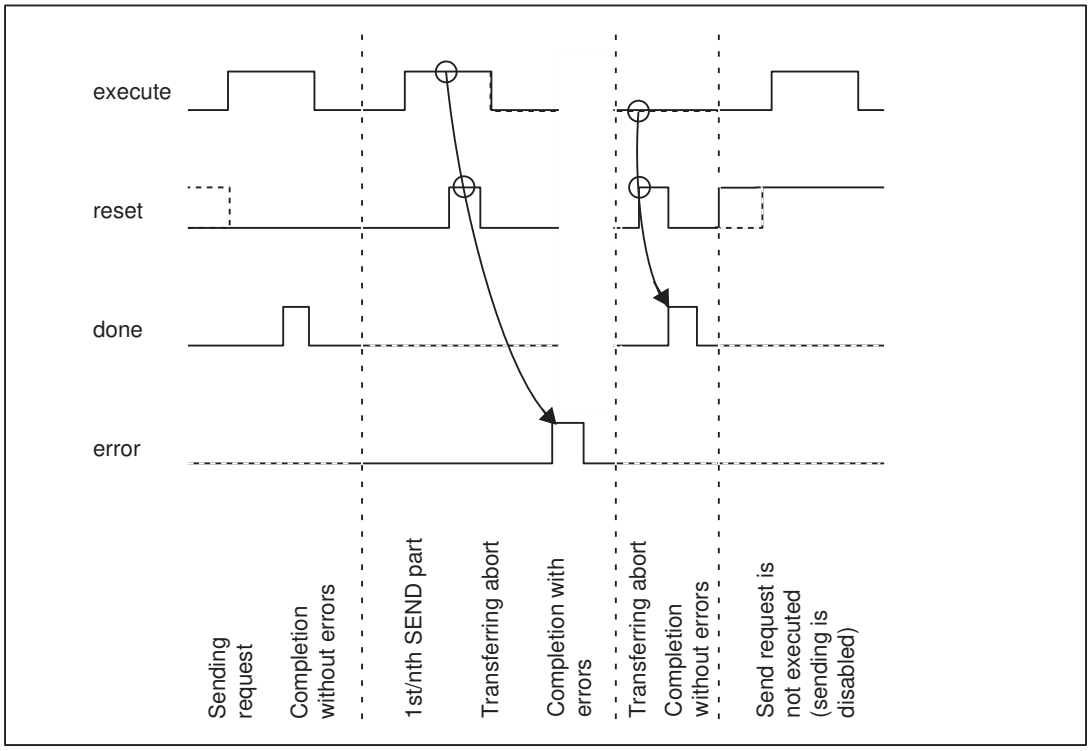


Figure 8-22 Signal sequence diagram of the _CP341_send FB

Note

The **execute** input is edge-triggered. The send job starts when there is a positive edge at the **execute** input.

Application with RK 512 computer interfacing

Sending data with the _CP341_send FB: SIMOTION with SIMOTION

Note

The handling of the **_CP341_send** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMOTION** communication.

You can send data to a communications partner by specifying 'SEND_CP' in the **mode** parameter.

A positive edge at the **execute** input initiates the data transfer. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

The data to be sent is specified in the following parameters:

- **data** indicates the data array containing the send data.
- **dataOffset** corresponds to the array index containing the first send byte.
- **dataLength** corresponds to the amount of data to be sent in bytes.

You must also indicate where the data is to be entered on the receiver.

- The **remoteCpuId** parameter specifies the number of the receiver (relevant for multiprocessor communication only).
- **remoteDataType** specifies the memory area where the data will be entered on the receiver (if the communications partner is a SIMOTION device, 68 must always be entered).

- **remoteMemIndex** specifies the memory index where the receive data will be entered. **RemoteMemIndex** = 0 is not permissible. An error message will be generated.
- **remoteDataOffset** specifies the array index in which the first receive byte is to be entered.

Note

The above parameters relate to a variable addressed specifically by the receiver. The receiver specifies the variable in which the data will be entered.

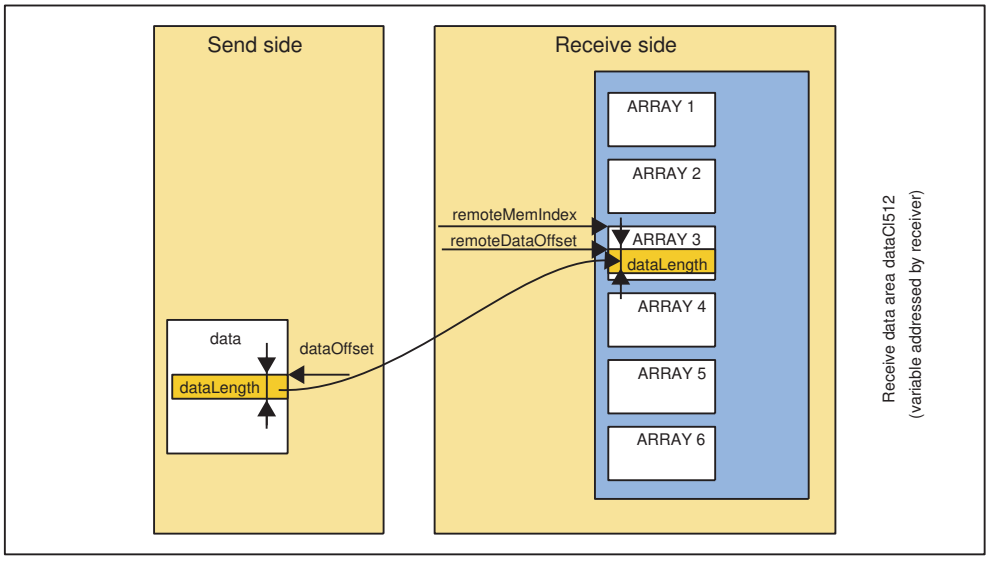


Figure 8-23 Send data _FB_P_Send: SIMOTION with SIMOTION

The communication flag bit (see Chapter Special features related to data transfer (Page 6223)), which is checked in the receiver before the send job is initiated and set once the send job is complete, is specified with parameters **remoteComFlagByte** (array index containing the communication flag bit) and **remoteComFlagBit** (bit position of the communication flag bit).

Special features for sending data

Assigning 255 to the **remoteComFlagByte** parameter will disable the communication flag functionality.

Sending data with the _CP341_send FB: SIMOTION with SIMATIC

Note

The handling of the **_CP341_send** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMATIC** communication.

You can send data to a communications partner by specifying 'SEND_CP' in the **mode** parameter.

A positive edge at the **execute** input initiates the data transfer. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

The data to be sent is specified in the following parameters:

- **data** indicates the data array containing the send data.
- **dataOffset** corresponds to the array index containing the first send byte.
- **dataLength** corresponds to the amount of data to be sent in bytes.

You must also indicate where the data is to be entered on the receiver.

- The **remoteCpuId** parameter specifies the number of the receiver (relevant for multiprocessor communication only).
- **remoteDataType** indicates the memory area where data will be entered on the receiver.
  - Data block = 68
  - Extended data block = 88
  - Flag = 77
  - Input = 69
  - Output = 65
  - Counter = 90
  - Timer = 84
- **remoteMemIndex** specifies the number of the data block or extended data block where the receive data will be entered. It has no relevance for all other receive areas. **RemoteMemIndex** = 0 is not permissible. An error message will be generated.
- **remoteDataOffset** specifies the array index in which the first receive byte is to be entered.

The communication flag bit (see Chapter Special features related to data transfer (Page 6223)), which is checked in the receiver before the send job is initiated and set once the send job is complete, is specified with parameters **remoteComFlagByte** (array index containing the communication flag bit) and **remoteComFlagBit** (bit position of the communication flag bit) in the SIMATIC flag area.

Special features for sending data

Assigning 255 to the **remoteComFlagByte** parameter will disable the communication flag functionality.

The signal sequence diagram of the **_CP341_send** FB can be found in Chapter Description of the _CP341_send FB (Page 6178).

Fetching data with the _CP341_send FB: SIMOTION with SIMOTION

Note

The handling of the **_CP341_send** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMOTION** communication.

You can fetch data from a communications partner by specifying "FETCH_CP" in the **mode** parameter.

A positive edge at the **execute** input initiates the data transfer. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

You must also indicate where the data are to be fetched on the receiver.

- The **remoteCpuId** parameter specifies the number of the receiver (relevant for multiprocessor communication only).
- **remoteDataType** specifies the memory area where the data will be fetched on the receiver (if the communications partner is a SIMOTION device, 68 must always be entered).
- **remoteMemIndex** specifies the memory index where the data will be fetched. **RemoteMemIndex** = 0 is not permissible. An error message will be generated.
- **remoteDataOffset** specifies the array index where the first byte is to be fetched.

Note

The above parameters relate to a variable addressed specifically by the communications partner. They specify the variable from which the data will be fetched.

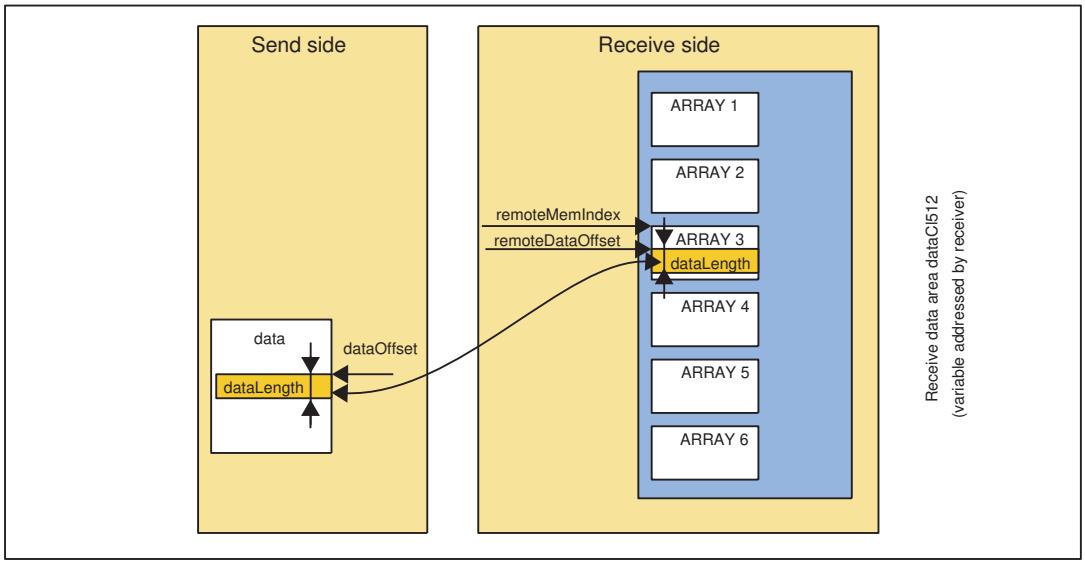


Figure 8-24 Fetch data _P_Send FB: SIMOTION with SIMOTION

The following parameters are used to specify where the fetched data will be saved:

- **data** corresponds to the data array in which the requested data will be entered.
- **dataOffset** corresponds to the array index in which the first fetched data byte will be entered.
- **dataLength** corresponds to the amount of data to be fetched in bytes.

The communication flag bit (see Chapter Special features related to data transfer (Page 6223)), which is checked in the receiver before the send job is initiated and set once the fetch job is complete, is specified with parameters **remoteComFlagByte** (array index containing the communication flag bit) and **remoteComFlagBit** (bit position of the communication flag bit).

Special features for fetching data

Assigning 255 to the **remoteComFlagByte** parameter will disable the communication flag functionality.

Fetching data with the _CP341_send FB: SIMOTION with SIMATIC

Note

The handling of the **_CP341_send** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMATIC** communication.

You can fetch data from a communications partner by specifying "FETCH_CP" in the **mode** parameter.

A positive edge at the **execute** input initiates the data transfer. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

You must also indicate where the data is to be fetched on the communications partner.

- **remoteCpuId** specifies the number of the communications partner (relevant for multiprocessor communication only).
- **remoteDataType** specifies the memory area where the data will be fetched on the communications partner.
  - Data block = 68
  - Extended data block = 88
  - Flag = 77
  - Input = 69
  - Output = 65
  - Counter = 90
  - Timer = 84

- **remoteMemIndex** specifies the number of the data block or extended data block where the receive data will be fetched.
  RemoteMemIndex = 0 is not permissible. An error message will be generated.
- **remoteDataOffset** specifies the array index where the first byte is to be fetched.

The following parameters are used to specify where the fetched data will be saved:

- **data** corresponds to the data array in which the requested data will be entered.
- **dataOffset** corresponds to the array index in which the first fetched data byte will be entered.
- **dataLength** corresponds to the amount of data to be fetched in bytes.

The communication flag bit (see Chapter Special features related to data transfer (Page 6223)), which is checked in the communications partner before the send job is initiated and set once the send job is complete, is specified with parameters **remoteComFlagByte** (array index containing the communication flag bit) and **remoteComFlagBit** (bit position of the communication flag bit) in the SIMATIC communication flag area.

Special features for fetching data

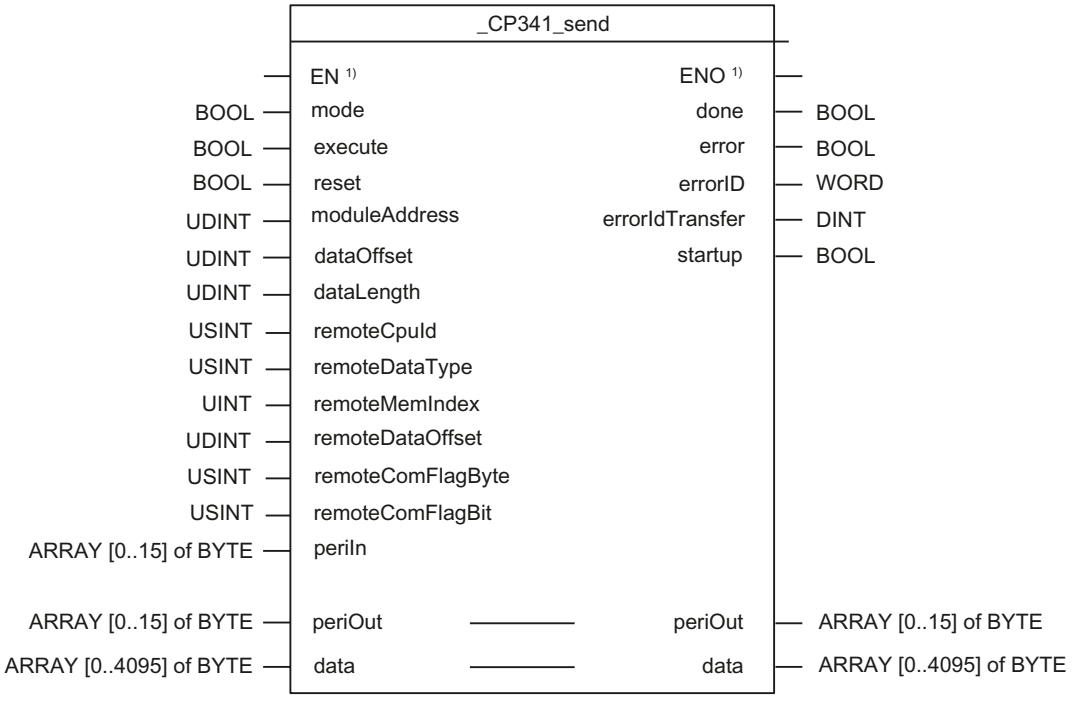
Assigning 255 to the **remoteComFlagByte** parameter will disable the communication flag functionality.

The signal sequence diagram of the **_CP341_send** FB can be found in Chapter Description of the _CP341_send FB (Page 6178).

Assignment in the data area

During the receive operation, the data to be received are stored temporarily in a local ARRAY. Once the data transfer from the CP 341 to the SIMOTION device is complete, the data is made available in the **data** (VAR_IN_OUT) parameter of the **_CP341_send** FB.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-43 Parameters of the _CP341_send FB (application with RK 512 computer link)

Name	P type 1)	Data type	Comment	Actions performed by user	Actions performed by block
mode	IN	EnumSendFetch	SEND_CP: Sends data FETCH_CP: Fetches data from the communications partner	Entered	Checked
execute	IN	BOOL	Initiates job on positive edge	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (found in **HW Config**)	Entered	Checked
dataOffset	IN	UDINT	First element to be sent	Entered	Checked
dataLength	IN	UDINT	Number of elements to be sent	Entered	Checked
remoteCpuId	IN	USINT	Number of the remote communications partner	Entered	Checked
remoteDataType	IN	USINT	Area type in the remote communications partner	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
remoteMemIndex	IN	UINT	Memory index in the remote communications partner (1 ≤ remoteMemIndex ≤ 6) [2]	Entered	Checked
remoteDataOffset	IN	UDINT	First element in the remote communications partner	Entered	Checked
remoteComFlag-Byte	IN	USINT	Communication flag index in the local communications partner	Entered	Checked
remoteComFlagBit	IN	USINT	Communication flag bit no. in the local communications partner	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP [3]	Checked and transferred to the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..4095] of BYTE	Send data array Receive data for parameter **mode =** FETCH_CP	Entered and checked	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.[4]	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [5])	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **remoteMemIndex** = 0 is not permissible. An error message will be generated.

[3] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

[4] For error information, refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 341"

[5] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

8.5.4.3 _CP341_receive function block

Description of the _CP341_receive FB

Function

The **_CP341_receive** function block enables you to receive data from a communications partner in the receive data array (3964(R) protocol, ASCII driver, RK 512) or to make data available for the communications partner (RK 512). Each data array depends on the type of protocol used. If you are using the 3964(R) protocol or ASCII driver, the receive data array is **dataCI3964**. For an RK 512 computer link, the data array is **dataCI512**. In both cases, 4,096 bytes are available. From the **dataCI512** data array, data is also made available for the communications partner.

Task integration (call)

The **_CP341_receive** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

Application with 3964(R) protocol or ASCII driver

SIMOTION device receives data from a communications partner

The **_CP341_receive** function block transfers a data block that is specified by the **dataCI3964** and **dataOffset** parameters from the CP 341 to a SIMOTION device. The **_CP341_receive** function block is called repeatedly by a program. The receive job can only be executed by cyclically calling the receive FB.

Receiving of data is enabled with static signal state "TRUE" in the **enable** parameter. An active data transfer can be canceled with signal state "FALSE" in the **enable** parameter. The canceled receive job is terminated with an error message at the **errorID** output. The receive operation will remain disabled as long as the signal state at the **enable** parameter is "FALSE". A data transfer operation can run over several calls, depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP341_receive** FB to its initial state. The receive operation will remain disabled as long as the signal state at the **reset** parameter is "TRUE".

The **moduleAddress** parameter specifies the module address of the CP 341 being addressed.

Status and error display on the _CP341_receive FB

The **newDataReceived** output indicates that the job has been completed without errors. The amount of data received is indicated in the **dataLength** parameter. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/number is displayed in the **errorID** output (see "Parameters of the _CP341_receive FB" table (3964 (R) procedure or ASCII driver)"). If no error has occurred, **errorID** has the value "0".
newDataReceived and **error**/**errorID** will also be output when the **_CP341_receive** FB is **reset**. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **newDataReceived**, **dataLength, error, errorID**, and **errorIdTransfer** parameters are available for one block call only.

Note

There is no parameter check for the **_CP341_receive** FB. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode.

Before a job from the CP 341 can be received following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP341_receive** FB must be complete.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

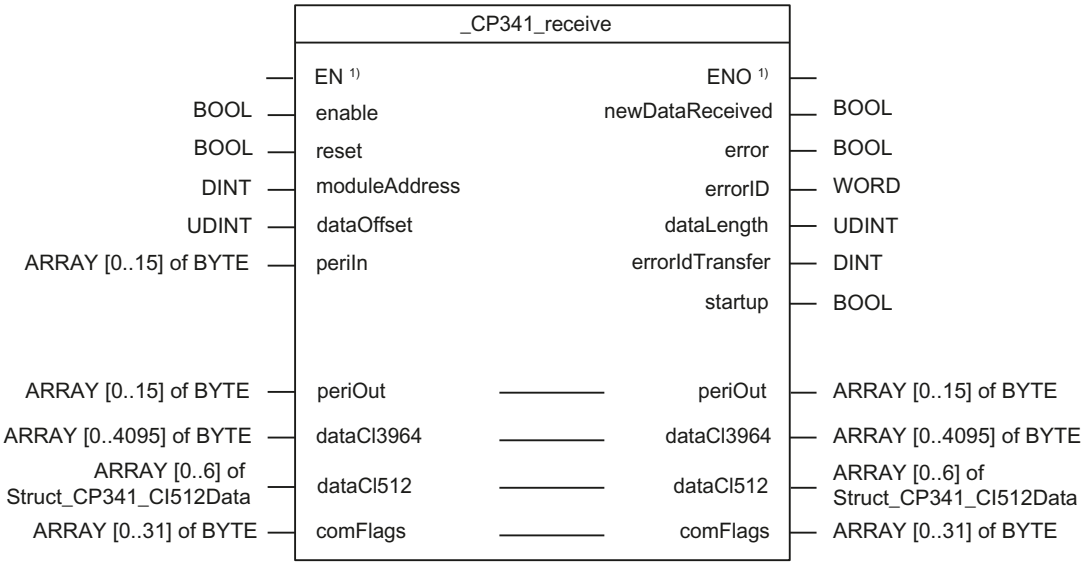
Assignment in the data area

During the receive operation, the data to be received are stored temporarily in a local ARRAY. Once the data transfer from the CP 341 to the SIMOTION device is complete, the data are made available in the **dataCI3964** (VAR_IN_OUT) parameter of the **_CP341_receive** FB.

Note

To ensure data consistency, do not access the receive data array until all the data has been received (**newDataReceived** = TRUE).

Call (LAD representation)



1) LAD-specific parameters

Parameters of the _CP341_receive FB (3964(R) protocol or ASCII driver)

Table 8-44 Parameters of the _CP341_receive FB (3964(R) protocol or ASCII driver)

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Receive enable	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
dataOffset	IN	UDINT	First element to be received	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP [2]	Checked and transferred to the I/O variable for the I/O outputs	Entered
dataCl3964	IN/OUT	ARRAY[0..4095] of BYTE	Receive data array	Entered and checked	Entered
dataCl512	IN/OUT	ARRAY[0..6] of 'Struct_CP341_Cl512Data'	Data area for RK 512 two-dimensional array [3]	Entered and checked	Checked
comFlags	IN/OUT	ARRAY[0..31] of BYTE	Communication flag area for RK 512	Entered and checked	Entered
newDataReceived	OUT	BOOL	New data have been received	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
dataLength	OUT	UDINT	Quantity of data received	Checked	Entered
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **status** parameter.[4]	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [5])	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

2) **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

3) No data can be exchanged in array index **dataCl512[0]**.

4) For error information, refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 341"

5) For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

Signal sequence diagram of the _CP341_receive FB

The following figure illustrates the behavior of the **newDataReceived**, **dataLength**, and **error** parameters according to the input circuit of **enable** and **reset**.

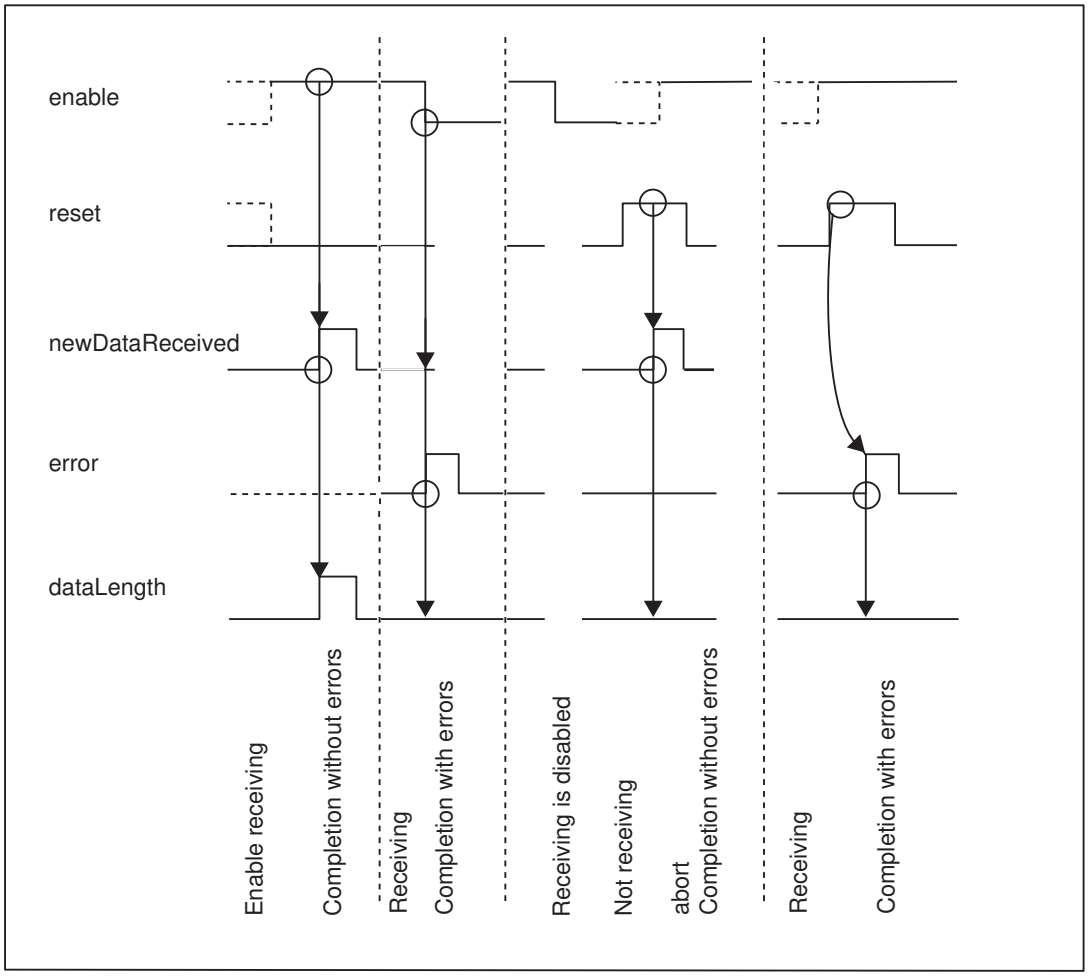


Figure 8-25 Signal sequence diagram of the _CP341_receive FB

Application with RK 512 computer interfacing

Receiving data with the _CP341_receive FB: SIMOTION with SIMOTION

Note

The handling of the **_CP341_receive** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMOTION** communication.

The data to be received will be entered in a variable created in the **dataCl512** parameter. The communications partner specifies where the receive data will be entered in the variable. The information about the entry location is output at the following output parameters.

- **localCpuId** specifies the number of the receiver (relevant for multiprocessor communication only).
- **localDataType** specifies the memory area where the receive data have been entered (irrelevant for SIMOTION with SIMOTION communication).
- **localMemIndex** specifies the memory index where the receive data have been entered.
- **localDataOffset** specifies the array index where the first receive byte has been entered.
- **dataLength** corresponds to the amount of data received in bytes.

A static "TRUE" signal state in the **enable** parameter enables a check to determine whether data are to be read from the CP 341. An active data transfer can be canceled with signal state "FALSE" in the **enable** parameter. The canceled receive job is terminated with an error message at the **errorID** output. The receive operation will remain disabled as long as the signal state at the **enable** parameter is "FALSE". A data transfer operation can run over several calls, depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

The communication flag area (see Chapter Special features related to data transfer (Page 6223)) is specified by a variable created in the **comFlags** parameter. The communication flag bit, which was checked in the receiver before the send job was initiated and was set after the send job was finished, is indicated at output parameters **localComFlagByte** (array index containing the communication flag bit) and **localComFlagBit** (bit position of communication flag bit).

Special features for receiving data

Assigning 255 to the **localComFlagByte** parameter will disable the communication flag functionality.

Receiving data with the _CP341_receive FB: SIMOTION with SIMATIC

Note

The handling of the **_CP341_receive** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMATIC** communication.

The communications partner on the SIMATIC side must set the input parameters according to the SIMOTION description.

Signal sequence diagram of the _CP341_receive FB

The following figure illustrates the behavior of the **newDataReceived, dataLength**, and **error** parameters according to the input circuit of **enable** and **reset**.

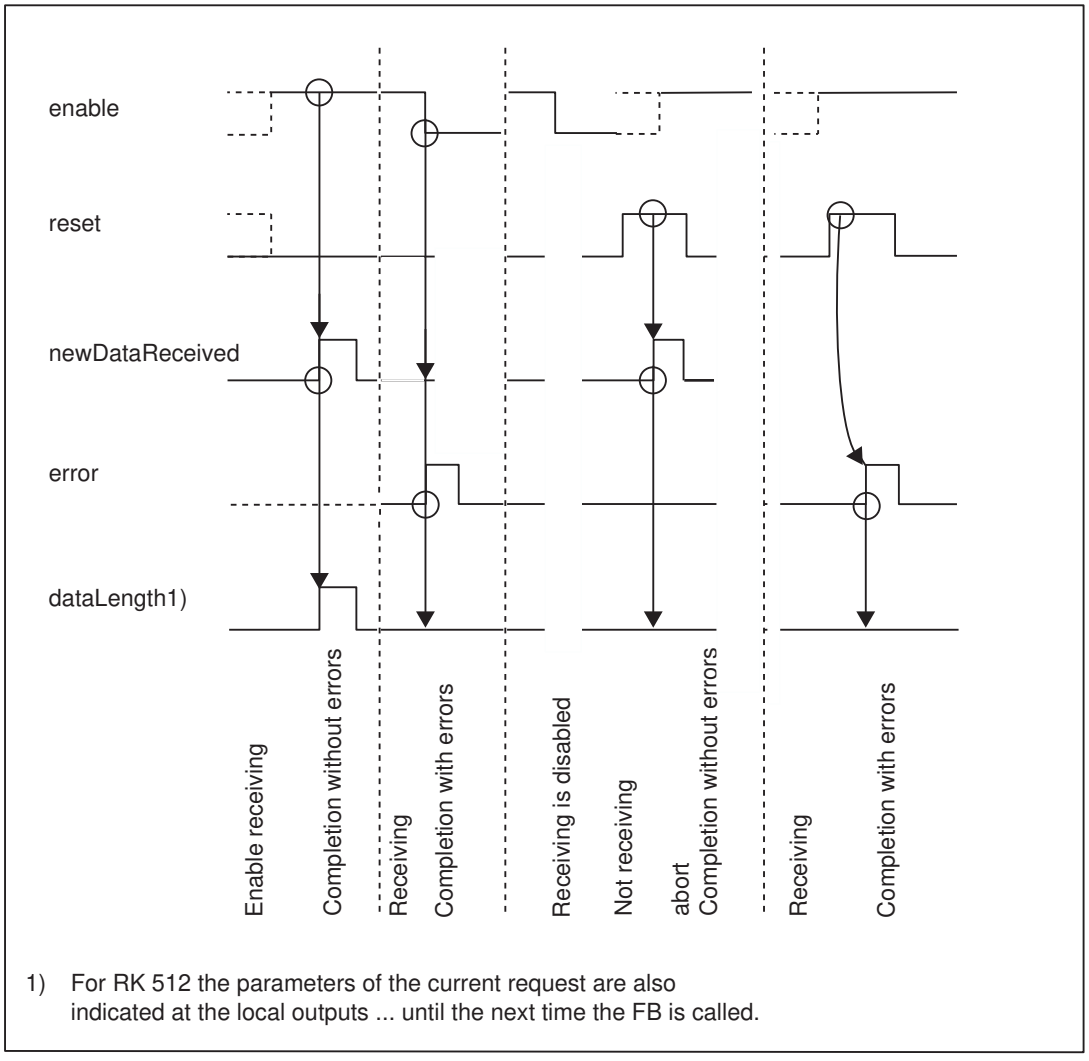


Figure 8-26 Signal sequence diagram of the _CP341_receive - receive data FB

Providing data with the _CP341_receive FB: SIMOTION with SIMOTION

Note

The handling of the **_CP341_receive** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMOTION** communication.

Function block processing is enabled with static signal state "TRUE" in the **enable** parameter. An active data transfer can be canceled with signal state "FALSE" in the **enable** parameter. The

canceled fetch job is terminated with an error message at the **errorID** output. Data cannot be provided as long as the signal state at the **enable** parameter is "FALSE". A data transfer operation can run over several calls, depending on the amount of data involved.

The **moduleAddress** parameter specifies the module address of the CP that is to be used for the data transfer.

The data to be provided are entered in a variable created in the **dataCl512** parameter. The communications partner specifies which data is to be fetched from the variable. The information about the fetched data is output in the following output parameters.

- **localCpuId** specifies the number of the communications partner providing the data (relevant for multiprocessor communication only).
- **localDataType** specifies the memory area where the data was fetched (irrelevant for SIMOTION).
- **localMemIndex** specifies the memory index where the data were fetched.
- **localDataOffset** specifies the array index where the first data byte was fetched.
- **dataLength** corresponds to the amount of data sent in bytes.

Special features for providing data

Assigning 255 to the **localComFlagByte** parameter will disable the communication flag functionality.

Providing data with the _CP341_receive FB: SIMOTION with SIMATIC

Note

The handling of the **_CP341_receive** FB differs according to whether data is to be exchanged with a **SIMOTION System** or a **SIMATIC System**.

The parameter assignment is described below for a function block for **SIMOTION with SIMATIC** communication.

The communications partner on the SIMATIC side must specify the input parameters according to the SIMOTION description.

Signal sequence diagram of the _CP341_receive FB

The following figure illustrates the behavior of the **newDataReceived, dataLength**, and **error** parameters according to the input circuit of **enable** and **reset**.

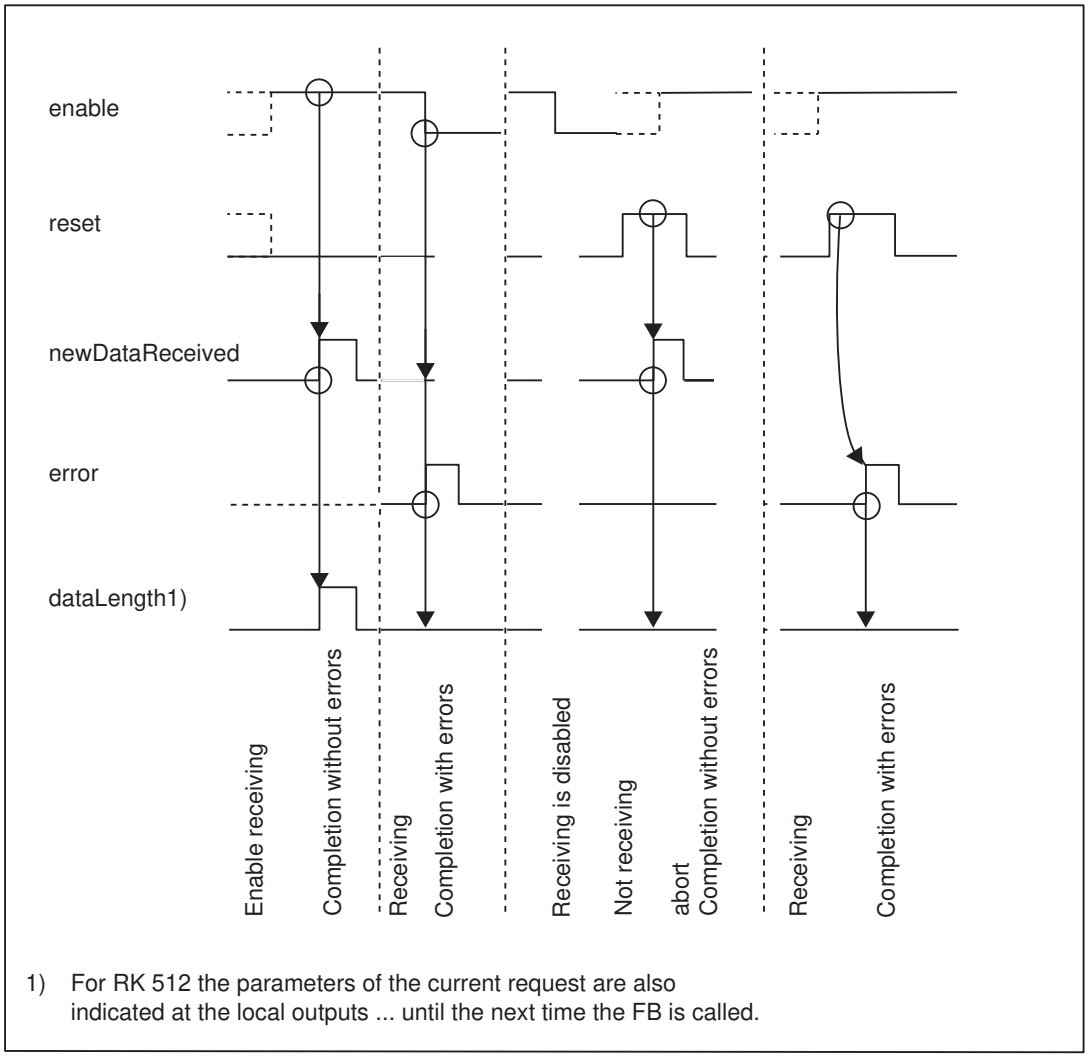


Figure 8-27 Signal sequence diagram of the _CP341_receive - providing data FB

Error display on the _CP341_receive FB

The **newDataReceived** output parameter indicates that the job has been completed without errors. The amount of data received is indicated in the **dataLength** parameter. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/number is displayed in the **errorID** output parameter (see "Parameters of the _CP341_receive FB" table (application with RK 512 computer link)"). If no errors have occurred, **errorID** has a value of 0. The **newDataReceived** and **error**/**errorID** parameters are also displayed on **reset** of the **_CP341_receive** FB. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **newDataReceived**, **dataLength, error, errorID**, and **errorIdTransfer** parameters are available for one block passage only.

Note

There is no parameter check for the **_CP341_receive** FB. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode.

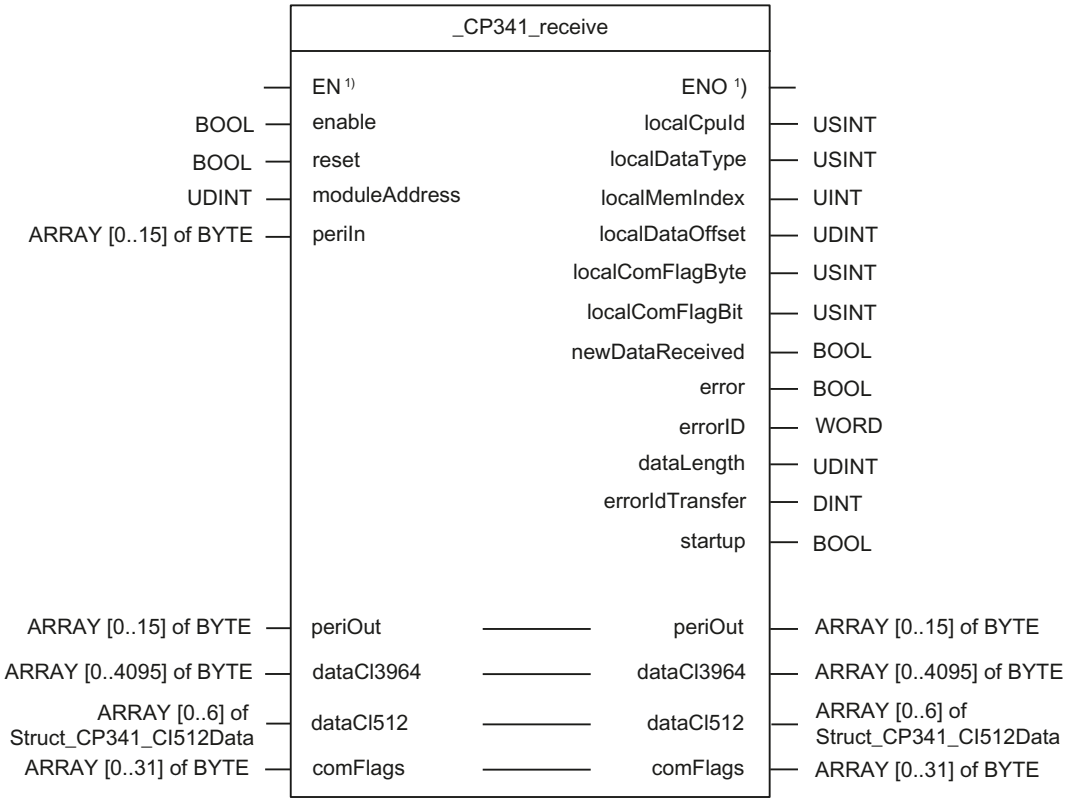
Before a job from the CP 341 can be received following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP341_receive** FB must be complete.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

Assignment in the data area

During the receive operation, the data to be received is stored temporarily in the **_CP341_receive** FB. Once the data transfer from the CP 341 to the SIMOTION device is complete, the data are made available in the **dataCI512** (VAR_IN_OUT) parameter of the **_CP341_receive** FB.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-45 Parameters of the _CP341_receive FB (application with RK 512 computer link)

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Receive enable	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	UDINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP [3]	Checked and transferred to the I/O variable for the I/O outputs	Entered
dataCl3964	IN/OUT	ARRAY[0..4095] of BYTE	Receive data array	Entered and checked	Entered
dataCl512	IN/OUT	ARRAY[0..6] of 'Struct_CP341_Cl512Data'	Data area for RK 512 two-dimensional array [2]	Entered and checked	Checked
Struct_CP341_Cl512Data (data structure)					
data		ARRAY[0..4095] of BYTE	Data array for RK 512 computer link (send and fetch job)	SEND_CP: checked FETCH_CP: Entered	SEND_CP: entered FETCH_CP: Checked
comFlags	IN/OUT	ARRAY[0..31] of BYTE	Communication flag area for RK 512	Entered and checked	Entered
localCpuId	OUT	USINT	Number of the local communications partner	Checked	Entered
localDataType	OUT	USINT	Area type in the local communications partner	Checked	Entered
localMemIndex	OUT	UINT	Memory index in the local communications partner	Checked	Entered
localDataOffset	OUT	UDINT	First element in the local communications partner	Checked	Entered
localComFlagByte	OUT	USINT	Communication flag index in the local communications partner	Checked	Entered
localComFlagBit	OUT	USINT	Communication flag bit no. in the local communications partner	Checked	Entered
newDataReceived	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.[4]	Checked	Entered
dataLength	OUT	UDINT	Quantity of data received	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [5])	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] No data can be exchanged in array index **dataCl512[0]**.

[3] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

[4] For error information, refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 341".

[5] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

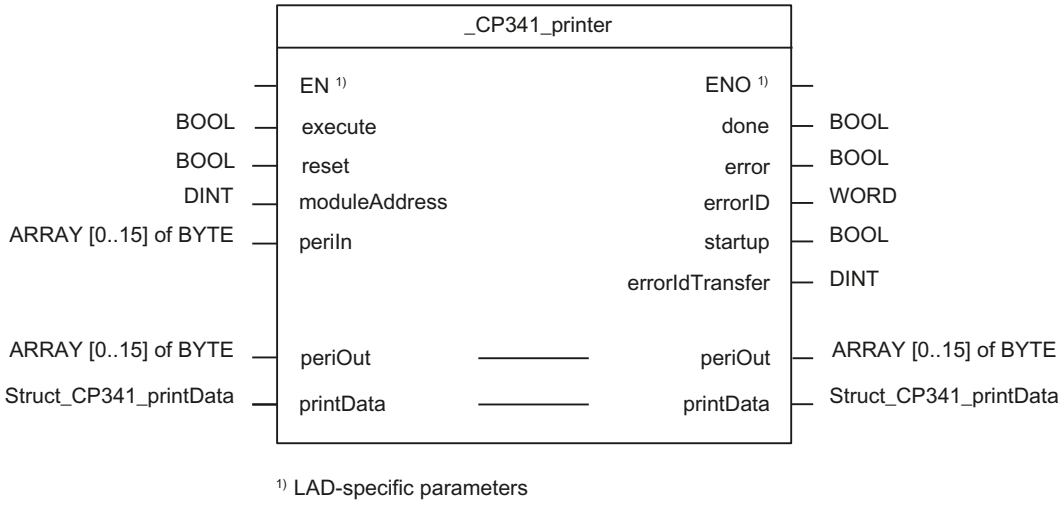
8.5.4.4 _CP341_printer function block

Description of the _CP341_printer FB

Function

The **_CP341_printer** function block is used to send data of type **Struct_CP341_printData** from the printer memory area to a serial printer. For example, the **_CP341_printer** function block might send a process message to the CP 341. The CP 341 prints out the process message on the connected printer.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-46 Parameters of the _CP341_printer FB

Name	P type 1)	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job on positive edge	Entered	Checked
reset	IN	BOOL	Cancels job	Entered	Checked
moduleAddress	IN	DINT	Module address of the CP for data set transfer (from HW Config)	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
periOut	IN/OUT	ARRAY[0..15] of BYTE	Prepared FB data for the I/O outputs of the CP 2)	Checked and transferred to the I/O variable for the I/O outputs	Entered
printData	IN/OUT	**Struct_CP341_printData**	Send data array (data to be printed)	Entered	Checked
Struct_CP341_printData (data structure) 5)					
variable		ARRAY[0..3] of Struct_CP341_dataRecord	Variable to be printed	Entered	Checked
format		ARRAY[0..150] of BYTE	Format string	Entered	Checked
Struct_CP341_dataRecord (data structure) 5)					
dataLength		UDINT	Quantity of data	Entered	Checked
data		ARRAY[0..31] of BYTE	Print data	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.[3]	Checked	Entered
startup	OUT	BOOL	Indicates CP startup	Checked	Entered
errorIdTransfer	OUT	DINT	Error during data transfer to the CP (precise error diagnostics if 16#1E0F is present at the **errorID** parameter [4])	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

[3] For error information, refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual, Chapter "Diagnostics with the CP 341".

[4] For a more detailed description (_readRecord and _writeRecord), see *SIMOTION System Function/Variable Devices* Parameter Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation.

[5] See "Message text structure", print storage area structure example

Signal sequence diagram of the _CP341_printer FB

The following figure illustrates the behavior of the **done** and **error** parameters according to the input circuit of **execute** and **reset**.

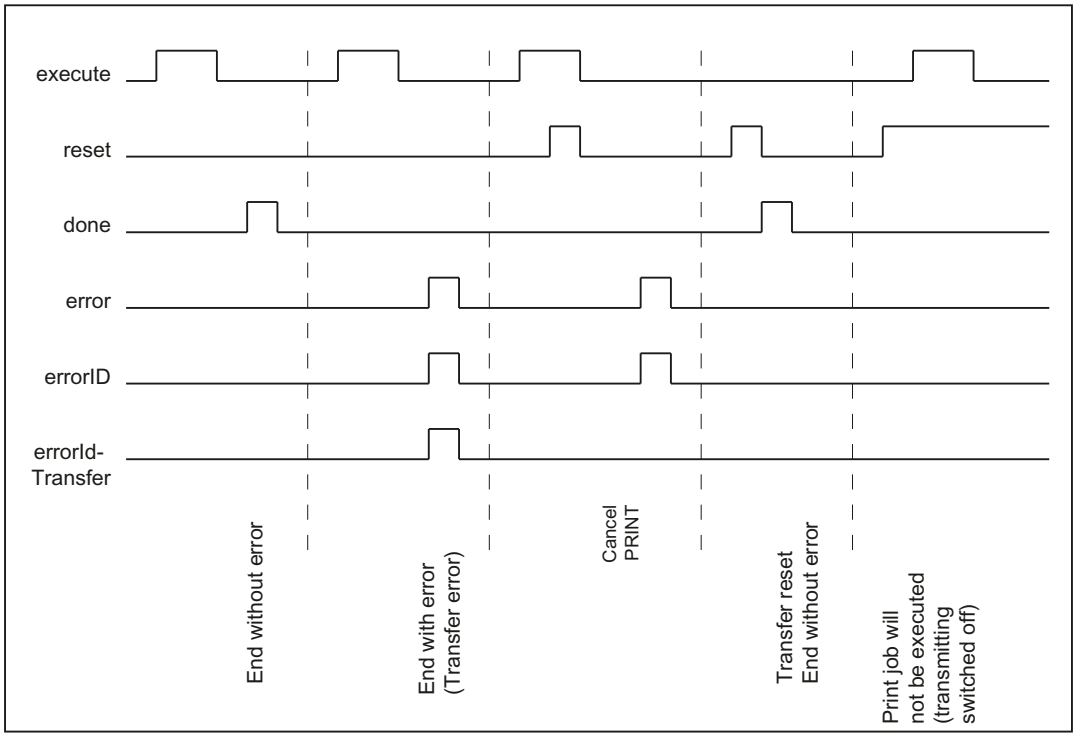


Figure 8-28 Signal sequence diagram of the _CP341_printer FB

Note

The **execute** input is edge-triggered. The send job starts when there is a positive edge at the **execute** input.

Task integration (call)

The **_CP341_printer** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

The **_CP341_printer** function block is called repeatedly by a program. The print job can only be executed by cyclically calling the print FB.

A positive edge at the **execute** input initiates the transfer of the message text. A data transfer operation can run over several calls (program cycles), depending on the amount of data involved.

The active transfer job can be canceled by setting the **reset** parameter to "TRUE". This will reset the **_CP341_printer** FB to its initial state. The sending of print jobs will remain disabled as long as the signal state at the **reset** parameter is TRUE.

The **moduleAddress** parameter specifies the module address of the CP 341 being addressed for the data set transfer.

For call examples for the **_CP341_printer** FB, see Chapter CP 341 print call examples (Page 6213).

Status and error display on the _CP341_printer FB

The **done** output parameter indicates that the job has been completed without errors. The **error** output indicates that an error has occurred. If an error occurs, the corresponding event class/ number is displayed in the **errorID** output parameter (see "Parameters of the CP341_printer FB" table). If no errors have occurred, **errorID** has a value of 0. **Done** and **error/errorID** are also output for **reset** of the **_CP341_printer** FB. When 16#1E0F is displayed in the **errorID** parameter, a detailed error description is also output via the **errorIdTransfer** parameter.

The **done**, **error, errorID**, and **errorIdTransfer** parameters are available for one block call only.

Note

There is no parameter check for the **_CP341_printer** function block. Incorrect parameterization of this block may cause the SIMOTION device to switch to "STOP" mode. Before the CP 341 can process an initiated job following a transition of the SIMOTION device from "STOP" to "RUN" mode, the CP-SIMOTION startup mechanism of the **_CP341_printer** FB must be complete. Any jobs initiated in the meantime will not be lost. They are transferred to the CP 341 once the startup coordination has finished.

The end of the startup coordination is indicated in output parameter **startup** = FALSE.

Assignment in the data area

The print data is transferred to the **_CP341_printer** FB as a data structure of type **Struct_CP341_printData** and copied to a local variable of the FB at the beginning of the print operation.

Structure of printer memory of type Struct_CP341_printData

The four variables to be printed and the format string must be entered in a variable with the following data type:

Example:

```
Struct_CP341_dataRecord     : STRUCT
     dataLength : UDINT;                        // Data quantity
     data       : ARRAY [0..31] of BYTE;        // Data field
END_STRUCT

Struct_CP341_printData    : STRUCT
     variable   : ARRAY [0..3] of Struct_CP341_dataRecord;  // 1st to 4th variable
     format     : ARRAY [0..150] of BYTE;                   // Format string
END_STRUCT
```

The first variable to be printed corresponds to the **variable [0]** element, the second variable to be printed corresponds to the **variable [1]** element, etc. The number of bytes to be printed per variable is limited to 32. The data for variable **i** must be placed in **variable [i-1].data[0..31]**. The number of bytes to be printed must be entered in the **variable [i-1].dataLength** element.

The format string corresponds to the format element. The format string must be structured as follows (refer to the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* Manual):

- Specification of string length in **format [0]**
- Specification of individual characters in **format [1 to 150]**

Note

If the maximum length is exceeded, the print job is canceled and event number 16#1E41 is indicated at the **errorID** parameter output of the **_CP341_printer** FB.

Entering variables and message texts in the printer memory area

Before the data transfer to the CP 341 begins, the variable values to be printed must be entered **byte by byte and in the proper format** in the **variable[].data** element of the data structure of type **Struct_CP341_printData** (see item 2 in the example below). The number of bytes for each variable (variable length) must be assigned to the **variable.datalength** element (e.g. WORD type variable - variable.datalength:=2). An entry corresponding to the data type of the value must be made in the **format** element for each value entered in the **variable** element. (e.g. WORD type variable - %I). The total length of the entries in the **format** element must be assigned to the **format[0]** element.

You configure message texts with the CP 341 "point-to-point connection" parameter assignment interface. Once the hardware configuration has been downloaded to the SIMOTION device, the message texts are stored in the CP 341. The message texts that have been saved can be selected with corresponding entries in the **variable** and **format** elements.

Note

You can use supplemental function blocks (see Chapter supplemental function blocks (Page 6207)) to enter values into the printer memory area and to select message texts.

Example:

- Print message text no. 3 (stored in CP 341).
  Configured message text: "This is message text no. 3"

```
myPrintData.variable[0].datalength := 2;   // Data type WORD
myPrintData.variable[0].data[0]    := 0;
myPrintData.variable[1].data[0]    := 3;   // Message text no. 3
```

```
myPrintData.format[0]     :=2;       // Format string length
myPrintData.format[1]     :=16#25;  // "%" Format specification for message
text
myPrintData.format[1]     :=16#4E   // "N" Format specification for message
text
```

- Print message text no. 4 with a WORD-type variable.

```
Configured message text   : "This is message text no. %I"
Printed text          : "This is message text no. 4"

myPrintData.variable[0].datalength := 2;  // Data type WORD
myPrintData.variable[0].data[0]    := 0;
myPrintData.variable[0].data[0]    := 4;  // Message text no.4

myPrintData.variable[1].datalength := 2;  // 2 Byte data type WORD
myPrintData.variable[1].data[0]    := 0   // High - Byte
myPrintData.variable[1].data[1]    := 4   // Low - Byte
myPrintData.format[0]              := 4;  // Format string length
myPrintData.format[1]          := 16#25;  // ASCII code "%" format
specification
myPrintData.format[2]          := 16#4E;  // ASCII code "N" format
specification
                                          // for message text
myPrintData.format[3]          := 16#25;  // ASCII code "%" format
specification
myPrintData.format[4]          := 16#49;  // ASCII code "I" format
specification
                                          // for integer
```

Notes on handling

The format string entry in the "format" field must be hexadecimal.

Example:
% corresponds to 25 hex in the IBM character set,
N (message text output) corresponds to 4E hex in the IBM character set.
(see Hardware configuration > Character set)

Data types DATE, TIME, DATE_AND_TIME_OF_DAY and TIME_OF_DAY are not supported. The date information must be entered as a DWORD or WORD in the printer data structure.

Representation type "A" (German date format):
//datefrg:=4018 (01.01.2001) and 4199 (01.07.2001)
printData.variable[0].datalength:=2;
printData.variable[0].data[0]:=WORD_TO_BYTE(SHR(datefrg,8));
printData.variable[0].data[1]:=WORD_TO_BYTE(SHR(datefrg,0));

Representation type "F":
The value to be printed must be in floating point format (mantissa/exponent) (see call example 2, Chapter CP 341 print call examples (Page 6213))

Representation type "C":
If variable[ ].datalength:=1 in the printer data structure, the characters will be printed horizontally.
If variable[ ].datalength:=2 (3,4) in the printer data structure, the characters will be printed vertically.

Representation type "X":
For CP 341 RS232, product version E08 and higher, representation type "X" (binary) outputs the values correctly on a serial printer.

Disconnected printer
The communications link is not monitored for printers even if alarm generation is enabled in HW Config of STEP 7.

Example:

- A break in the connection between the printer and the CP 341 triggers neither an error nor a diagnostic alarm.
- Nor are they triggered if a print job is started but no printer is connected.

Note

The code in examples 1, 2, and 3 (see Chapter CP 341 print call examples (Page 6213)) can be transferred to the SIMOTION SCOUT editor with **Copy** and **Paste**.

supplemental function blocks

Function

Supplemental function blocks are provided for entering variables of various data types as well as for entering message texts into data structure **Struct_CP341_printData**. You enter the value of the variables **byte by byte** and **in the proper format** in the **variable** element of the printer memory area and, optionally, in the **format** element. When numbers are entered for message texts, one entry is made in each of the **format** and **variable** elements. The method of representation for the variables in printed text and the method of entry in the format string can be selected by means of parameters.

The following supplemental function blocks are available:

- **_CP341_realToPrintData**
  Entry of a number of data type REAL into data structure **Struct_CP341_printData**
- **_CP341_dwordToPrintData**
  Entry of a number of data type DWORD into data structure **Struct_CP341_printData**
- **_CP341_wordToPrintData**
  Entry of a number of data type WORD into data structure **Struct_CP341_printData**
- **_CP341_byteToPrintData**
  Entry of a number of data type BYTE into data structure **Struct_CP341_printData**
- **_CP341_dintToPrintData**
  Entry of a number of data type DINT into data structure **Struct_CP341_printData**

- **_CP341_intToPrintData**
  Entry of a number of data type INT into data structure **Struct_CP341_printData**
- **_CP341_printMsgText**
  Selection of message texts stored in CP 341

Note

SINT and USINT data types can be entered into data structure **Struct_CP341_printData** with the **_CP341_intToPrintData** function block. Type conversion is implicit.

Parameter description

Table 8-47 _CP341_byteToPrintData, _CP341_wordToPrintData, _CP341_dwordToPrintData parameters

Name	P type [1)]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	BYTE/WORD/ DWORD	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1≤ **numVariable** ≤ 4	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_RE-PLACE_WITH_SIGN	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_RE-PLACE_WITH-OUT_SIGN	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: unsigned integer		
		CP_REPLACE_BINA-RY	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: binary		
		CP_ADD_WITH-OUT_SIGN	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: unsigned integer		
		CP_ADD_WITH_SIG N	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNum-ber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP341_prin tData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-48 _CP341_realToPrintData

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	REAL	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_REPLACE_WITHOUT_EXPONENT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: floating-point		
		CP_REPLACE_WITH_EXPONENT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: with exponent		
		CP_ADD_WITHOUT_EXPONENT	Entry is added in the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: floating-point		
		CP_ADD_WITH_EXPONENT	Entry is added in the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: with exponent		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP341_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-49 _CP341_dintToPrintData, _CP341_intToPrintData

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
data	IN	INT/DINT	Variable/value to be entered in the data structure.	Entered	Checked
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_DEFAULT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_ADD_TO_STRING	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string. **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP341_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Table 8-50 _CP341_printMsgText

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
execute	IN	BOOL	Edge-triggered job initiation	Entered	Checked
numMsgText	IN	USINT	Number of the message text (stored in the CP 341)	Entered	Checked

Name	P type [1]	Data type	Comment	Actions performed by user	Actions performed by block
numVariable	IN	INT	Number of the variable in which the entry is to be made. 1 ≤ **numVariable** ≤ 4	Entered	Checked
entryFormatString	IN	ENUM	Method of entry in the format string and method of representation of the value in the **data** parameter.	Entered	Checked
		CP_DEFAULT	Entry starting at byte 1 in the substructure of the format string; existing entries are overwritten. Method of representation: signed integer		
		CP_ADD_TO_STRING	Entry is added to the substructure of the format string, **entryAtByteNumber** parameter is evaluated. Method of representation: signed integer		
		CP_NO_ENTRY	No entry in the substructure of the format string		
entryAtByteNumber	IN	USINT	Specifies the byte at which the entry is to begin in the substructure of the format string **entryAtByteNumber** = 0 Entry is made after the last entry found	Entered	Checked
printData	IN/OUT	Struct_CP341_printData	Data structure for the printer data	No actions	Enters values
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors (permissible value range exceeded)	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Task integration (call)

The supplemental blocks must be called in the **BackgroundTask** or the **TimerInterruptTask**.

For call examples for the **_CP341_printer** FB, see Chapter CP 341 print call examples (Page 6213).

Status and error indicators

The **done** output indicates that the job has been completed without errors. The **error** output indicates that an error has occurred.

The **done** and **error** parameters are available for one block call only.

CP 341 print call examples

Call example 1

```
UNIT E1CP341p;

INTERFACE
VAR_GLOBAL
  myPreparePrintData          : BOOL;               // Initiate prepare print request
  myRequestPrint              : BOOL;               // Initiate transfer to printer
  myReset                     : BOOL;               // Abort print
  myActualLevel               : REAL := 5.67;       // actual value "level"
  myModuleAddress_1           : DINT:= 256;         // address of 1st CP341 module
  myPrintData                 : Struct_CP341_printData; // instance of datastruct
  myFB_CP341_printMessageText : _CP341_printMsgText;    // instances of function blocks
  myFB_CP341_realToPrintData  : _CP341_realToPrintData; // instances of function blocks
  myFB_CP341_print            : _CP341_printer;         // instance of function block
  myOutputArrayCP341_1        : ARRAY[0..15] of BYTE;   // field for CP341 output data
END_VAR
PROGRAM Example_print_1;                            // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_1
// BackgroundTask program

// Example to print out messagetext 3 with one variable
// ("actualLevel") of type REAL: "The actual level (l) is: <actualLevel>"
// The message-text 3 must be specified in the hardwareconfiguration of CP341
// like this: "The actual level (l) is:"

// The following I/O-variable for CP341 module are required:
// peripheralInputCP341_1: input address of CP341 module; type Array; length 16
// peripheralOutputCP341_1: output address of CP341 module; type Array; length 16

// entry to printDatastruct myprintData
myFB_CP341_printMessageText ( execute           := myPreparePrintData,
                              printData         := myPrintData,
                              numMsgText        := 3,        // number of message text
                              numVariable       := 1,        // number of variable
                              entryFormatString := CP_DEFAULT);
```

```
myFB_CP341_realToprintData  ( execute           := myPreparePrintData,
                              printData         := myPrintData,
                              data              := myActualLevel,
                              numVariable       := 2,
                              entryFormatString := CP_ADD_WITH_EXPONENT );
// call instance of _CP341_printer
// use requestPrint to start datatransfer to serial printer

myFB_CP341_print      ( execute       := myRequestPrint,           // initiate request
                        reset         := myReset,                  // abort request
                        moduleAddress := myModuleAddress_1,        // module address
                        periIn        := myPeripheralinputcp341_1, // peripheral input
                        periOut       := myOutputArrayCP341_1,     // output data field
                        printData     := myPrintData);             // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP341_1 := myOutputArrayCP341_1;

END_PROGRAM //Example_print_1
END_IMPLEMENTATION
```

Call example 2

```
UNIT E2CP341p;

INTERFACE
VAR_GLOBAL
  myRequestPrint        : BOOL;                    // Initiate transfer to printer
  myReset               : BOOL;                    // abort print
  myPrintDword          : DWORD;                   // value to print out
  myModuleAddress_1     : DINT := 256;             // address of 1st CP341 module
  myPrintData           : Struct_CP341_printData;  // instance of datastruct
  myFB_CP341_print      : _CP341_printer;          // instance of function block
  myOutputArrayCP341_1  : ARRAY[0..15] OF BYTE;    // field for CP341 output data
END_VAR
PROGRAM Example_print_2;                           // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_2                            // BackgroundTask program
// example with notation "F" and variable of type DWORD
// (containing mantissa and exponent)
```

```
// The following I/O-variable for CP341 module are required:
// peripheralInputCP341_1: input address of CP341 module; type Array; length 16
// peripheralOutputCP341_1: output address of CP341 module; type Array; length 16

// formatstring (length 2 Byte, notation "F"):
myPrintData.format[0]  := 2 ;
myPrintData.format[1]  := 16#25 ;            // "%"
myPrintData.format[2]  := 16#46 ;            // "F"

// assignment for variable (type DWORD), to print out with notation "F"
myPrintDword := REAL_TO_DWORD(10000.0);  // variable (DWORD) with mantissa and exponent

// !!!
// ATTENTION! wrong example for this case is an assingnment with an integer value e.g.:
// myprintDword := 10000;        // because the format is WITHOUT mantissa and exponent
// !!!

// fill out printData interface manually with DWORD-variable
myPrintData.variable[0].dataLength := 4 ;    // 1st variable, lenth 4 Byte
myPrintData.variable[0].data[0] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,24)));
myPrintData.variable[0].data[1] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,16)));
myPrintData.variable[0].data[2] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,8)));
myPrintData.variable[0].data[3] := WORD_TO_BYTE(DWORD_TO_WORD(SHR(printDword,0)));

// call instance of _CP341_printer
// use requestPrint to start datatransfer to serial printer
myFB_CP341_print ( execute        := myRequestPrint,            // initiate request
                   reset          := myReset,                   // abort request
                   moduleAddress  := myModuleAddress_1,         // module address
                   periIn         := myPeripheralinputcp341_1,  // peripheral input
                   periOut        := myOutputArrayCP341_1,      // output data field
                   printData      := myPrintData);              // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP341_1 := myOutputArrayCP341_1;

END_PROGRAM //Example_print_2
END_IMPLEMENTATION
```

Call example 3

```
UNIT E3CP341p;
```

```
INTERFACE
VAR_GLOBAL
  myRequestPrint              : BOOL;                  // Initiate transfer to printer
  myReset                     : BOOL;                  // Abort print
  myPrintReal1                : REAL;                  // 1st value
  myPrintReal2                : REAL;                  // 2nd value
  myPrintReal3                : REAL;                  // 3rd value
  myModuleAddress_1           : DINT := 256;           // address of 1st CP341 module
  myPrintData                 : Struct_CP341_printData; // instance of datastruct
  myFB_CP341_print            : _CP341_printer;        // instance of function block
  myFB_CP341_realToprintData1 : _CP341_realToPrintData; // instances of function blocks
  myFB_CP341_realToprintData2 : _CP341_realToPrintData; // instances of function blocks
  myFB_CP341_realToprintData3 : _CP341_realToPrintData; // instances of function blocks
  myOutputArrayCP341_1        : ARRAY[0..15] OF BYTE;
END_VAR
PROGRAM Example_print_3;                               // program for BackgroundTask
END_INTERFACE

IMPLEMENTATION
PROGRAM Example_print_3                                // BackgroundTask program
// The following example program demonstrates usage of _CP341_realToPrintData()

// The following I/O-variable for CP341 module are required:
// peripheralInputCP341_1: input address of CP341 module; type Array; length 16
// peripheralOutputCP341_1: output address of CP341 module; type Array; length 16

// preset variables with user-values
myPrintReal1 := 1.11; myPrintReal2 := 2.22; myPrintReal3 := 3.33;
```

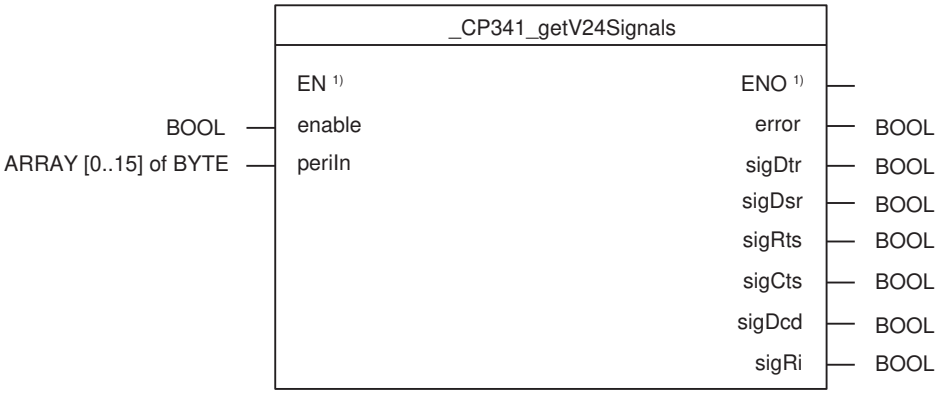
```
// write variables (type REAL) to printDatastruct with _CP341_realToPrintData
myFB_CP341_realToprintData1 (execute           := TRUE,
                             data              := myPrintReal1,        // 1st variable
                             numVariable       := 1,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);
myFB_CP341_realToprintData2 (execute           := TRUE,
                             data              := myPrintReal2;        // 2nd variable
                             numVariable       := 2,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);
myFB_CP341_realToprintData3 (execute           := TRUE,
                             data              := myPrintReal3,        // 3rd variable
                             numVariable       := 3,
                             entryFormatString := CP_ADD_WITHOUT_EXPONENT,
                             entryAtByteNumber := 0,
                             printData         := myPrintData);

// call instance of _CP341_printer, "requestPrint" starts data transfer
// to serial printer
myFB_CP341_print (  execute        := myRequestPrint,            // initiate request
                    reset          := myReset,                   // abort request
                    moduleAddress  := myModuleAddress_1,         // module address
                    periIn         := myPeripheralinputcp341_1,  // peripheral input
                    periOut        := myOutputArrayCP341_1,      // output data field
                    printData      := myPrintData);              // data to print out

// transfer output data field to peripheral output
myPeripheralOutputCP341_1 := myOutputArrayCP341_1;

END_PROGRAM                       //Example_print_3
END_IMPLEMENTATION
```

8.5.4.5 _CP341_getV24Signals function block

Function

The **_CP341_getV24Signals** function block reads the RS-232-C accompanying signals from the CP 341 and makes them available to the user in the block parameters. The functionality of the **_CP341_getV24Signals** FB can only be used if a parameterized ASCII driver is specified.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-51 Parameters of the _CP341_getV24Signals FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Block enable	Entered	Checked
periIn	IN	ARRAY[0..15] of BYTE	I/O inputs of the CP transferred to the FB	I/O variable of the I/O inputs of the CP transferred to the FB	Checked
error	OUT	BOOL	Job completed with errors	Checked	Entered
sigDtr	OUT	BOOL	Data terminal ready	Checked	Entered
sigDsr	OUT	BOOL	Data set ready	Checked	Entered
sigRts	OUT	BOOL	Request to send	Checked	Entered
sigCts	OUT	BOOL	Clear to send	Checked	Entered
sigCcd	OUT	BOOL	Data carrier detected	Checked	Entered
sigRi	OUT	BOOL	Ring Indicator	Checked	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Task integration (call)

The **_CP341_getV24Signals** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

The RS 232C accompanying signals are updated each time the function is called (cyclic polling). The CP 341 updates the status of the inputs/outputs in a time base of 20 ms.

Note

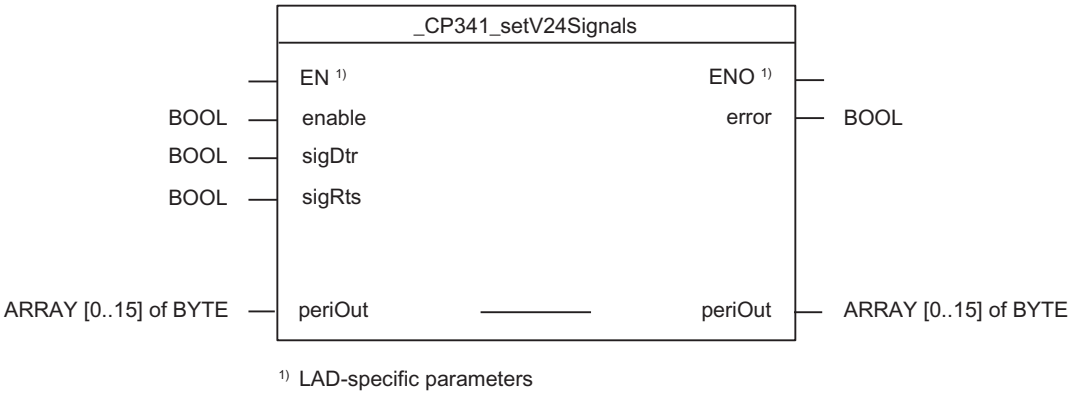
A minimum pulse duration is necessary to detect a signal change. Determining factors are the cycle time (SIMOTION device), the update time on the CP 341, and the response time of the communications partner.

8.5.4.6 _CP341_setV24Signals function block

Function

The **_CP341_setV24Signals** function block can be used to set or reset RS-232-C accompanying signals. The functionality of the **_CP341_setV24Signals** FB can only be used if a parameterized ASCII driver is specified.

Call (LAD representation)



Parameter description

Table 8-52 Parameters of the _CP341_setV24Signals FB

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Block enable	Entered	Checked
sigDtr	IN	BOOL	Data terminal ready	Entered	Checked
sigRts	IN	BOOL	Request to send	Entered	Checked

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
peri-Out	IN/OUT	ARRAY [0..15] of BYTE	Prepared FB data for the I/O outputs of the CP [2]	Checked and transferred to the I/O variables for the I/O outputs	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with a variable of type **ARRAY[0..15] of BYTE**. Create a local or global variable in your program under **VAR** (do not create a temporary variable under VAR_TEMP). After the FB has been called, this variable must be assigned to the I/O variable for the I/O outputs of the module. See call example for CP 341.

Task integration (call)

The **_CP341_getV24Signals** function block must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

8.5.4.7 Calling the CP 341 function blocks

In order to be able to work with the function blocks in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for the **_CP341_send** FB).
2. Create a variable for the data structure (for RK 512 computer interfacing only).
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the FB are accessed with <instance name of FB>.<name of output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variables created in step 3.

Note

The CP 341 call example is an extract from the supplied E_CP341 application example, which is included on the "SIMOTION Utilities & Applications" CD-ROM.

If you wish to control multiple CP 341 devices, you must create a new variable for the data structure and FB instances with a new name for each CP 341 you implement.

Call example for CP 341

```
UNIT E_CP341;

INTERFACE
```

```
VAR_GLOBAL
      myExecSendCP341      : BOOL;                    // Trigger send task
      myModuleAddrCP341    : DINT:=256;               // module address
      mySendDataArrayCP341 : ARRAY [0..4095] OF BYTE; // Send data array 4,096 bytes
      myInstCP341Send      : _CP341_send;             // Create instance of FB                    (1)
END_VAR

PROGRAM ExampleCP341;                                // Program in BackgroundTask

END_INTERFACE

IMPLEMENTATION
    VAR_GLOBAL
      MyResetSend          : BOOL;                    // Cancel send order
    END_VAR

PROGRAM ExampleCP341                                 // Program in BackgroundTask

// Variables used: see interface area under VAR_GLOBAL
VAR
    MyCPOutputArray       : ARRAY [0..15] OF BYTE;   // Array for CP output data                 (3)
END_VAR

VAR_TEMP
    MyDataLengthSend      : UDINT;            // Length of data to be sent
    MyDataOffsetSend      : UDINT;            // Offset of first byte to be sent
END_VAR

// CALL FB INSTANCE TO SEND
    myInstCP341Send (                                                                            (4)
        mode             := SEND_CP,                // Send data                                 (5)
        execute          := myExecSendCP341,        // Trigger order
        reset            := myResetSend,            // Order book
        moduleAddress    := myModuleAddrCP341,      // Module address
        dataOffset       := myDataOffsetSend,       // Offset
        dataLength       := myDataLengthSend,       // Number of data to be sent
        periIn           := myPeripheralInputCP341, // I/O variable of I/O inputs
        periOut          := myCPOutputArray,        // Data for I/O outputs
        data             := mySendDataArrayCP341    // Send data array
        );

    myStateStartUpCP341 := myInstCP341Send.startUp;  // Start-up status                          (6)

// TRANSFER DATA TO CP341
```

```
    myPeripheralOutputCP341  := myCPOutputArray;    // Array for output variables                (7)
                                                    // of I/O variables

END_PROGRAM               // ExampleCP341

END_IMPLEMENTATION
```

Note

The "ExampleCP341" program must be assigned in the execution system.

8.5.4.8 Data consistency

When sending data

Once a job is initiated by a positive edge at the **execute** input, the data to be sent is copied to the static memory area of the send FB. This means that once the FB call has ended, the send data array can be written to again for the next send request on a positive edge. The data are retained as consistent data within the send FB.

Note

During the copy operation to the static memory area of the FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

When receiving data

Once the receive request is complete, the data are copied over to the receive buffer in a block from the static memory area of the receive FB. This means that when the FB call is complete, either all data (**newDataReceived** = TRUE) are entered in the receive buffer or no data (**newDataReceived** = FALSE) have been entered.

Note

During the copy operation from the static memory area of the receive FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

8.5.4.9 Special features related to data transfer

Communication flag function with the CP 341

The communication flag functionality with respect to RK 512 computer interfacing is similar to that in the SIMATIC. Once data has been sent or fetched, a bit is set in the communications partner. This bit is checked when a new job is initiated. If it is set (= TRUE), the job is not processed. The communications partner must first reset this bit (= FALSE). The bit to be set or checked is specified by the requester FB (**_CP341_send**) in the **remoteComFlagByte** and **remoteComFlagBit** parameters.

SIMOTION ↔ SIMOTION

In the variable (ARRAY [0..31] of BYTE) created in the **comFlags** parameter, the bit position of the communication flag bit to be checked or set is specified as follows:

- **remoteComFlagByte** corresponds to the array index of the byte in which the bit is to be set/ checked.
- **remoteComFlagBit** corresponds to the bit position in the byte specified by the **remoteComFlagByte** parameter.

Note

The same variable must always be transferred at the **comFlags** parameter during the block call to ensure consistency of the communication flag bit.

SIMOTION ↔ SIMATIC

Communication flags are always stored in the flag area MB0 to MB254 in SIMATIC. The bit position of the communication flag bit to be checked or set is specified as follows:

- **remoteComFlagByte** corresponds to the flag byte in which the bit is to be set or checked (e.g. **remoteComFlagByte**=1 corresponds to MB1).
- **remoteComFlagBit** corresponds to the bit position in the byte specified by the **remoteComFlagByte** parameter.

Requests that can be processed simultaneously with the CP 341

The following function blocks may only be programmed once in your user program for each CP 341 communication processor used:

- _CP341_send FB
- _CP341_receive FB

Data transfer with the RK 512 computer interfacing

Data transfer options

Active jobs:

The **_CP341_send** function block enables you to issue active jobs to the CP 341 in the user program of the SIMOTION system. You can

- Send data from your automation system to a remote communications partner.
- Fetch data from a remote communications partner and store it in the send data array.

Note

If you fetch data from a CP 341, a **_CP341_receive** function block must be programmed for the communications partner.

Passive jobs:

The **_CP341_receive** function block enables you to use passive jobs to coordinate the reading of data on the CP 341 and make the data available. The communications partner is active. You can

- Enter data sent by the communications partner in the **dataCI512** receive data array.
- Make data available from **dataCI512** for a remote communications partner.

Special feature related to sending data

Note the following special features related to "sending" data:

- With RK 512, the amount of sent data must be an even number. If an odd value is specified for the length in the **dataLength** parameter, an additional filler byte with a value of "FALSE" will be transferred at the end of the data.
- With RK 512, any specified offset must be an even number. If an odd offset is specified, the data will be stored for the partner starting with the next smallest even offset.

8.5.4.10 Application example of the CP 341

Function

This example shows how to:

- use the **_CP341_send** function block to send data from the Send data array to a communications partner.
- use the **_CP341_receive** function block to receive data in the receive data array.

In the example program, the CP 341 is used both as the sender and receiver. This requires the jumpering of the send and receive lines (PIN 2 and PIN 3 on the RS232 interface) and the "ASCII" setting in the parameter assignment tool. The **_CP341_send** FB is used to transfer the send data to the CP module. This sends the data using the RS232 interface. The jumpered send and receive line means that the data to be sent is read immediately by the CP module. The **_CP341_receive** FB reads the received data from the CP module and copies these data to the receive data array.

This example requires proper installation of the parameter assignment tool, as described in the SIMATIC *CP 341 Point-to-Point Connection, Installation and Parameter Assignment* manual.

Hardware platform

The application example is available for various SIMOTION hardware platforms. You must adapt the example for centralized applications with SIMOTION C.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station with a CP 341 module.
4. Double-click the **CP 341** to open the "Properties" dialog box for this module. Click the "Parameters" button to launch the parameter assignment tool of the CP 341 module.
5. The "ASCII" protocol must be selected in the protocol selection box.
   The standard settings of the ASCII protocol suffices for the application example.
6. Jumper the send and receive line (PIN 2/PIN 3) of the RS232 interface.
7. Apply your settings in the parameter assignment interface with the "File" > "Save" menu command, and close the interface with the "File" > "Exit" menu command. Click "OK" to close the Properties window for the CP 341 module.
8. Save the hardware configuration with the "Station" > "Save and compile" menu command.
9. Download the hardware configuration with the "Target system" > "Download to module" menu command.
   The red "SF" LED on the IM 153 turns on and then off if the assigned module parameters have been downloaded without errors.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (e.g. PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (e.g. PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and module address in the program (myModuleAddrCP341).
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Sequence of the application example

Table 8-53 Input icons used

Symbol	Data type	Designation
mySelectPointToPointCP341	BOOL	Select point-to-point connection
myExecSendCP341	BOOL	Initiate send job
mySendOrder1	BOOL	Select send: Job 1
mySendOrder2	BOOL	Select send: Job 2
mySendOrder3	BOOL	Select send: Job 3
myResetSend	BOOL	Cancel send job
myEnableToReceive	BOOL	Receive enable
myReceiveOrder1	BOOL	Select receive: Job 1
myReceiveOrder2	BOOL	Select receive: Job 2
myReceiveOrder3	BOOL	Select receive: Job 3
myResetReceive	BOOL	Cancel receive job
myModuleAddrCP341	DINT	CP 341 module address, default 256
mySendDataArrayCP341	ARRAY[0..4095] of BYTE	Send data array
myReceiveDataArrayCP341	ARRAY[0..4095] of BYTE	Receive data array

Table 8-54 Output symbols used

Symbol	Data type	Designation
mySendDone	BOOL	Send: Completed
mySendError	BOOL	Send: Error display
mySendErrorNumber	WORD	Send: Error status
mySendTransErrorNumber	DINT	Send: Error status transfer
myNewDataReceived	BOOL	Receive: New data have been received
myReceiveError	BOOL	Receive: Error display
myReceiveErrorNumber	WORD	Receive: Error status
myReceiveTransErrorNumber	DINT	Receive: Error status transfer

Symbol	Data type	Designation
myStateStartupCP341	BOOL	CP 341 startup status FALSE = startup completed
myDiagnosticAlarm	BOOL	TRUE = diagnostic alarm present on the CP 341
myProcessAlarm	BOOL	TRUE = process alarm present on the CP341
myAlarmInterrupt	UDINT	Type of the alarm (process, diagnostic alarm)
myLogBaseAddrIn	DINT	Module address
myLogBaseAddrOut	DINT	
myLogAddress	DINT	Diagnostic address
myAlarmDetails	DWORD	Alarm information

Note

You can monitor and control the input and output variables used in the programming example in the INTERFACE area of the unit (under VAR_GLOBAL); alternatively, you can assign real inputs and outputs to the input and output variables in your user program.

For the "point-to-point connection" application example, set the "myselectPointToPointCP341" input to "TRUE". This will call function blocks contained in the application example.

Receiving data:

To receive data, you must set the "myEnableToReceive" variable to "TRUE" (static signal). If receive jobs 1 and 3 are enabled (myReceiveOrder1 = TRUE and "myReceiveOrder3" = TRUE), the data is stored in the "myReceiveDataArrayCP341" data array starting with the "myReceiveDataArrayCP341[0]" array element (data offset is 0). If job 2 is enabled ("myReceiveOrder2" = TRUE), the data is stored in the "myReceiveDataArrayCP341" data array starting with the "myReceiveDataArrayCP341[20]" array element (data offset is 20).
If "myNewDataReceived" = TRUE, this indicates that new data has been received. This signal is present for one cycle only.

If an error occurred during the transfer ("myReceiveError" = TRUE) the error code is stored in the "mySendErrorNumber" variable. If error code 16#1E0F is present in "myReceiveErrorNumber", an error occurred during the data transfer. The transfer error code is stored in the "myReceiveTransErrorNumber" variable. The error signals are reset when you set input "myResetReceive" = TRUE.

Sending data:

You can use the "mySendOrder1", "mySendOrder2" and "mySendOrder3" inputs to select between three send jobs:

- Job 1 sends 10 bytes of data from the "mySendDataArrayCP341" data array from array element "mySendDataArrayCP341[0]" to "mySendDataArrayCP341[9]"
- Job 2 sends 20 bytes of data from the "mySendDataArrayCP341" data array from array element "mySendDataArrayCP341[20]" to "mySendDataArrayCP341[39]"
- Job 3 sends 4,096 bytes of data from the "mySendDataArrayCP341" data array.

The data is sent to the communications partner if the "myExecSendCP341" input detects a signal change from "FALSE" to "TRUE" (positive edge).
If output signal "mySendDone" = TRUE, the send job has been completed. A new job can be sent if the "myExecSendCP341" input signal detects another signal change from FALSE to TRUE.

If an error occurred during the transfer ("mySendError" = TRUE), the error code is stored in the "mySendErrorNumber" variable. If error code 16#1E0F is present in "mySendErrorNumber", an error occurred during the data transfer. The transfer error code is stored in the "mySendTransErrorNumber" variable. The error signals are reset when you set input "myExecSendCP341" = FALSE.

When the signal state at the "myResetSend" or "myResetReceive" input is set to "TRUE", the send job or receive job is canceled, respectively. If the signal state remains "TRUE", sending and receiving of data is disabled.

Note

Proper data transfer can be observed as follows:

- The "TxD" and "RxD" LEDs on the CP module illuminate.
- Output parameter (_CP341_send FB) **done** = TRUE or **NewDataReceived** = TRUE

8.5.5 Alarm processing

Pending error messages are processed and evaluated differently in a SIMOTION system than in a SIMATIC system. Diagnostic alarms are not enabled by default. Enable the alarms for each module in the hardware configuration, see Chapter Integrating the communications processors in the SIMOTION project (Page 6136).

If you have parameterized diagnostic alarms, then you should program the alarm processing sequence according to the principle presented below.

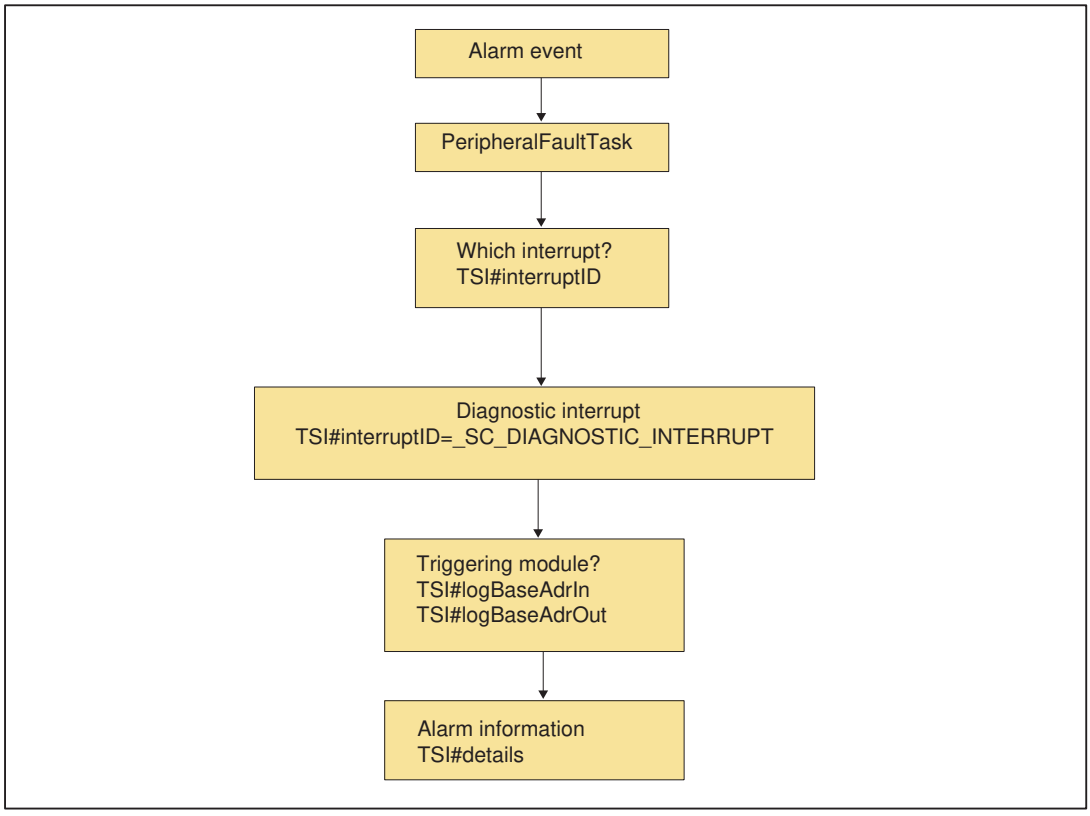


Figure 8-29 Alarm processing for CP 340 or CP 341

Alarm evaluation

Alarms originating from the I/O are evaluated in the **PeripheralFaultTask**. When the **PeripheralFaultTask** is started, the **Taskstartinfo** is made available, which you can evaluate in the user program.

The **Taskstartinfo** of **PeripheralFaultTask** is comparable to the local data of OB82 in the SIMATIC system.

Table 8-55 Meaning of the Taskstartinfo

Task	TSI		Remarks
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an input area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an output area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the alarm was caused by a station failure or station recovery of an associated DP slave, otherwise _SC_INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

Definition of a diagnostic alarm

If the user program is to respond to an internal or external error, you can set the parameters for a diagnostic alarm that will interrupt the cyclical program of the SIMOTION device.

Events triggering a diagnostic alarm

The criteria (events) that trigger diagnostic alarms in a SIMOTION system are the same as in a SIMATIC system.

More detailed information is available in the following SIMATIC manuals: *CP 340 Point-to-Point Connection, Installation and Parameter Assignment and CP 341 Point-to-Point Connection, Installation and Parameter Assignment*.

Responses to a diagnostic alarm

If a diagnostic alarm occurs, the following take place:

- Diagnostic data are written to **TSI#details** variable in the Taskstartinfo of **PeripheralFaultTask**.
- In the **PeripheralFaultTask**, you can read out and save the 4 bytes of diagnostic data.
- The group error LED (SF) of the CP module lights up. The group error LED (SF) is extinguished as soon as the error has been remedied.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

More information is available in the following SIMATIC manuals:

- *CP 340 Point-to-Point Connection, Installation and Parameter Assignment*
- *CP 341 Point-to-Point Connection, Installation and Parameter Assignment*

8.5.6 Appendices

8.5.6.1 SIMOTION and SIMATIC names

The tables below contain a comparison of SIMOTION and SIMATIC names.

Table 8-56 SIMOTION and SIMATIC names for CP 340

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
Function block parameters		
_CP340_send	**FB P_SEND (FB 3)**	**_FB_CP340_send**
execute	REQ	request
reset	R	abort
moduleAddress	LADDR	moduleAddress
dataOffset	DBB_NO	dataOffset
dataLength	LEN	dataLength
periIn	-	inputInterface
periOut	-	outputInterface
data	DB_NO	data
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber
errorIdTransfer	-	transferErrorNumber
startup	-	startup
_CP340_receive	**FB P_RCV (FB 2)**	**_FB_CP340_receive**
enable	EN_R	enable
reset	R	abort
moduleAddress	LADDR	moduleAddress
dataOffset	DBB_NO	dataOffset
periIn	-	inputInterface
periOut	-	outputInterface

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
data	DB_NO	data
newDataReceived	NDR	newDataReceived
error	ERROR	error
dataLength	LEN	dataLength
errorID	STATUS	errorNumber
errorIdTransfer	-	transferErrorNumber
startup	-	startup
_CP340_printer	**FB P_PRINT (FB 4)**	**_FB_CP340_print**
execute	REQ	request
reset	R	abort
moduleAddress	LADDR	moduleAddress
periIn	-	inputInterface
periOut	-	outputInterface
printData	(DB_NO/DBB_NO)	printerData
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber
errorIdTransfer	-	transferErrorNumber
startup	-	startup
_CP340_getV24Signals	**FC V24_STAT (FC 5)**	**_FB_CP340_getV24State**
enable	-	-
periIn	-	inputInterface
error	-	-
sigDtr	DTR_OUT	signalDTR
sigDsr	DSR_IN	signalDSR
sigRts	RTS_OUT	signalRTS
sigCts	CTS_IN	signalCTS
sigDcd	DCD_IN	signalDCD
sigRi	RI_IN	signalRI
_CP340_setV24Signals	**FC V24_SET (FC 6)**	**_FB_CP340_setV24Signals**
enable	-	-
sigDtr	DTR	signalDTR
sigRts	RTS	signalRTS
periOut	-	outputInterface
error	-	-
Data structure elements	**Data structure elements**	
Struct_CP340_printData	No direct reference to SIMATIC	**Struct_CP340_printerData**

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	**Name in the SIMATIC system**	**Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)**
variable	-	variable
format	-	format
Struct_CP340_dataRecord	No direct reference to SIMATIC	**Struct_CP340_dataRecord**
dataLength	-	dataLength
data	-	data
Module parameters (supplemental function blocks)	**Module parameters (supplemental function blocks)**	
_CP340_byteToPrintData, _CP340_wordToPrintData, _CP340_dwordToPrintData,	-	**_FB_CP340_byteToPrinterdata, _FB_CP340_wordToPrinterdata, _FB_CP340_dwordToPrinterdata,**
execute	-	execute
data	-	data
numVariable	-	numberVariable
entryFormatString	-	entryFormatstring
entryAtByteNumber	-	entryAtByteNumber
printData	-	printerData
done	-	done
error	-	error
_CP340_realToPrintData	-	**_FB_CP340_realToPrinterdata**
execute	-	execute
data	-	data
numVariable	-	numberVariable
entryFormatString	-	entryFormatstring
entryAtByteNumber	-	entryAtByteNumber
printData	-	printerData
done	-	done
error	-	error
_CP340_dintToPrintData, _CP340_intToPrintData	-	**_FB_CP340_dintToPrinterdata, _FB_CP340_intToPrinterdata**
execute	-	execute
data	-	data
numVariable	-	numberVariable
entryFormatString	-	entryFormatstring
entryAtByteNumber	-	entryAtByteNumber
printData	-	printerData
done	-	done
error	-	error
_CP340_printMsgText	-	**_FB_CP340_printMessageText**
execute	-	execute
numMsgText	-	numberMsgText
numVariable		numberVariable

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
entryFormatString	-	entryFormatstring
entryAtByteNumber	-	entryAtByteNumber
printData	-	printerData
done	-	done
error	-	error

Table 8-57 SIMOTION and SIMATIC names for CP 341

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
Function block parameters		
_CP341_send	**FB P_SND_RK (FB 8)**	**_FB_CP341_send**
mode	SF	operatingMode
execute	REQ	request
reset	R	abort
moduleAddress	LADDR	moduleAddress
dataOffset	DBB_NO	dataOffset
dataLength	LEN	dataLength
periIn	-	inputInterface
periOut	-	outputInterface
data	DB_NO	data
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber
errorIdTransfer	-	transferErrorNumber
remoteCpuId	R_CPU_NO (computer interfacing)	remoteCPUNumber
remoteDataType	R_Typ (computer interfacing)	remoteDataType
remoteMemIndex	R_NO (computer interfacing)	remoteMemoryIndex
remoteDataOffset	R_OFFSET (computer interfacing)	remoteDataOffset
remoteComFlagByte	R_CF_BYT (computer interfacing)	remoteComFlagByte
remoteComFlagBit	R_CF_BIT (computer interfacing)	remoteComFlagBit
startup	_	startup
_CP341_receive	**FB P_RCV_RK (FB 7)**	**_FB_CP341_receive**
enable	EN_R	enable
reset	R	abort
moduleAddress	LADDR	moduleAddress
dataOffset	DBB_NO	dataOffset
periIn	-	inputInterface
periOut	-	outputInterface

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	**Name in the SIMATIC system**	**Name in the SIMOTION system up to V3.2 (SIMOTION function library)**
dataCl3964	DB_NO (procedure/ASCII driver)	dataCL3964
newDataReceived	NDR	newDataReceived
error	ERROR	error
dataLength	LEN	dataLength
errorID	STATUS	errorNumber
errorIdTransfer	-	transferErrorNumber
dataCl512	-	dataCL512
comFlags	-	communicationFlags
localCpuId	-	localCPUNumber
localDataType	L_TYP (computer interfacing)	localDataType
localMemIndex	L_NO (computer interfacing)	localMemoryIndex
localDataOffset	L_OFFSET (computer interfacing)	localDataOffset
localComFlagByte	L_CF_BYT (computer interfacing)	localComFlagByte
localComFlagBit	L_CF_BIT (computer interfacing)	localComFlagBit
startup	_	startup
_CP341_printer	**FB P_PRINT_RK (FB13)**	No equivalent
execute	REQ	-
reset	R	-
moduleAddress	LADDR	-
periIn	-	-
periOut	-	-
printData	DB_NO/DBB_NO	-
done	DONE	-
error	ERROR	-
errorID	STATUS	-
errorIdTransfer	-	-
startup	-	-
_CP341_byteToPrintData, _CP341_wordToPrintData, _CP341_dwordToPrintData, _CP341_realToPrintData, _CP341_dintToPrintData, _CP341_intToPrintData, _CP341_printMsgText	-	No equivalent
_CP341_getV24Signals	**FC V24_STAT (FC 5)**	**_FB_CP341_getV24State**
enable	-	-
periIn	-	inputInterface
error		-
sigDtr	DTR_OUT	signalDTR

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION function library)
sigDsr	DSR_IN	signalDSR
sigRts	RTS_OUT	signalRTS
sigCts	CTS_IN	signalCTS
sigDcd	DCD_IN	signalDCD
sigRi	RI_IN	signalRI
_CP341_setV24Signals	**FC V24_SET (FC 6)**	**_FB_CP341_setV24Signals**
enable		-
periIn	-	inputInterface
error		-
sigDtr	DTR	signalDTR
sigRts	RTS	signalRTS
periOut	-	outputInterface
Data structure elements	**Data structure elements**	
Struct_CP341_CI512CpData	No direct reference to SIMATIC	**Struct_CP341_CL512Data**
data	-	data

8.5.6.2 List of abbreviations

Table 8-58 Abbreviations

Abbreviation	Meaning
CP	Communications processor
DP	Distributed I/O
FB	Function block
HW	Hardware
IM	Interface Modul (SIMATIC S7-300 interface module)
IN	Input parameters
IN/OUT	In/out parameters
I/O	Input/Output
LAD	Ladder Logic
LED	Light Emitting Diode (Light emitting diodes)
OUT	Output parameter
RK	Computer link

8.6 Supplement to the FM 350-1, FM 350-2, and FM 352 Modules

Preface

Contents of the function manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

This documentation is a supplement to the following SIMATIC manuals:

- FM 350-1 Function Module, *Installation and Parameter Assignment*
- FM 350-2 Counter Module, *Installation and Parameter Assignment*
- FM 352 Electronic Cam Controller, *Installation and Parameter Assignment*

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation.

This manual supplement will help you to correctly integrate and commission the FM 350-1, FM 350-2 and FM 352 modules in a SIMOTION system.

Differences in handling which result from the software architecture of a SIMOTION system as compared to the software architecture of a SIMATIC system will be described.

Function blocks

The function blocks required for communication between the SIMOTION system and the FM 350-1, FM 350-2 and FM 352 modules are included in the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P

- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.6.1 Fundamental safety instructions

8.6.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.6.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.6.2 Description

8.6.2.1 General part

This section describes the general similarities and differences in the operation of the function modules (FMs) in a SIMOTION system as compared to a SIMATIC system.

Note

This manual is a supplement to the following SIMATIC manuals:

- *FM 350–1 Function Module Installation and Parameter Assignment*
- *FM 350–2 Counter Function Module Installation and Parameter Assignment*
- *FM 352 Electronic Cam Controller, Installation and Parameter Assignment*

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation!

The following software versions are required for the standard functions described in this documentation:

- SIMOTION SCOUT V4.2 or higher
- SIMOTION Kernel V4.2 or higher

8.6.2.2 Product description

FM 350-1 counter module

The FM 350-1 is a single-channel, high-speed counter module. The module can function in the following counter ranges:

- 0 to 2^{32} - 1
- -2^{31} to 2^{31} - 1

The maximum input frequency of the counter signals is up to 500 kHz depending on the encoder signal.

The FM 350-1 can be used for the following counting and measuring tasks:

- Continuous counting
- Single counting
- Periodic counting
- Frequency measurement
- Period measurement
- Speed measurement

You can start and stop the count or measurement process either via the user program (software gate) or via external signals (hardware gate).

From release 3 (Article No.: 6AG1350-1AH03-2AE0), you can also operate the FM 350-1 in a clock synchronized manner with a suitable interface module (IM) in the ET 200M system.

FM 350-2 counter module

The FM 350-2 is an 8-channel counter module with dosing functions. The module can function in the following counter ranges:

- -2^{31} to 2^{31} - 1

The maximum input frequency of the counter signals is up to 10 kHz per counter channel depending on the encoder signal.

The FM 350-2 can be used for the following tasks:

- Continuous counting up/down
- Single counting up/down
- Periodic counting up/down
- Frequency measurement
- Speed measurement
- Period measurement
- Dosing

You can start and stop the count either via the user program (software gate) or via external signals (hardware gate).

The counter, gate and direction signals can be connected directly to the module.

FM 352 electronic cam controller

The FM 352 is a single-channel electronic cam controller. It supports both rotary axes and linear axes. Initiators, incremental encoders or absolute encoders (SSI) can be connected for position sensing. As a slave, the FM 352 can also listen in on the SSI message frame of an absolute encoder.

Up to a maximum of 128 position-based or time-based cams can be parameterized. The position-based and time-based cams can be assigned to 32 cam tracks. The first 13 cam tracks are output via the digital outputs on the module.

Function blocks

Function blocks are required for controlling the function modules. The function blocks for the SIMOTION system are described in this manual.

Functionality of the function modules

The function blocks (FBs) and the function modules (FMs) have the same functionality in a SIMOTION system as in a SIMATIC S7 automation system. However, the execution of data transfers and the handling of FBs have been adapted to the given SIMOTION boundary conditions.

Possible applications

In addition to the possible applications described in the SIMATIC manuals, the FM 350-1, FM 350-2 and FM 352 function modules can also be used in a SIMOTION system. The function modules can be used as centralized modules (on the SIMOTION C2xx only) or as distributed modules (SIMOTION C2xx, SIMOTION P35x and SIMOTION D4xx).

Several FM 350-1, FM 350-2 and FM 352 modules can be operated on one SIMOTION device.

The following figure shows you how to connect an ET 200M distributed I/O device with IM 153-1 and FM 350-1 or FM 350-2 or FM 352 to a SIMOTION device (e.g. SIMOTION C2xx).

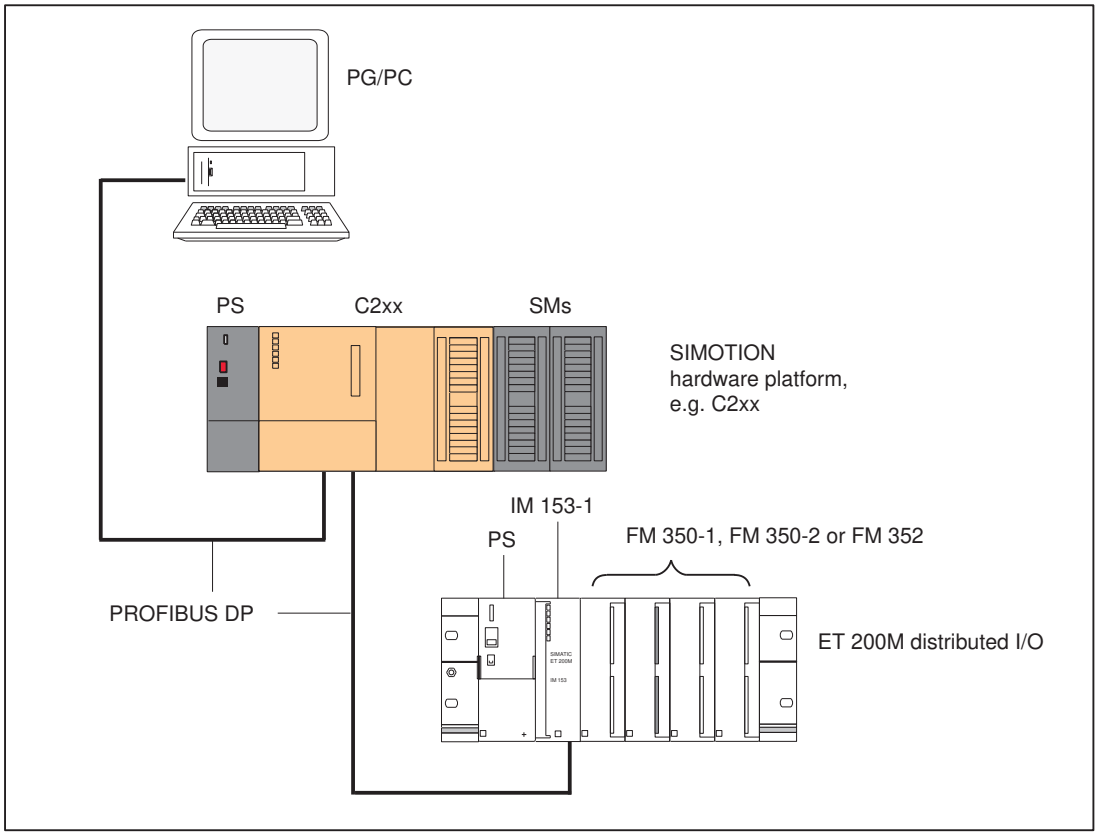


Figure 8-30 Connection of the FMs in an ET 200M to the SIMOTION C2xx device (example of distributed application)

8.6.2.3 Installation and connection

Overview

The following steps need to be carried out in order to commission the FM 350-1, FM 350-2 or FM 352 function modules and to control them from the SIMOTION system:

Distributed application (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx)

1. Assemble and wire the ET 200M distributed I/O device complete with power supply (PS), interface module (IM) and function module (FM).
2. Establish the PROFIBUS connection between the ET 200M and the SIMOTION device.
3. Set the PROFIBUS DP node address on the IM.

4. Switch on the terminating resistor at the first and last bus node.

Note

For steps 1 to 4, refer to the *ET 200M Distributed I/O* manual.

This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

5. Insert the function modules FM 350-1 / FM 350-2 / FM 352 into the SIMOTION project, see chapter Inserting function modules into the SIMOTION project (Page 6244).
6. Assign parameters for the FM 350-1, FM 350-2 or FM 352 function module.
   The following SIMATIC manuals describe how to install the parameter assignment interfaces for the FMs and how to assign the module parameters:
   – For FM 350-1, refer to the *FM 350-1 Function Module, Installation and Parameter Assignment* manual.
   – For FM 350-2, refer to the *FM 350-2 Counter Module, Installation and Parameter Assignment* manual.
   – For FM 352, refer to the *FM 352 Electronic Cam Controller, Installation and Parameter Assignment* manual.
7. Link the function blocks to the SIMOTION project (refer to chapter Integrating the function blocks in the user project (Page 6245)).

Centralized application (SIMOTION C2xx only)

1. For information on planning the mechanical installation and preparing and mounting the SIMOTION components, refer to the *SIMOTION C2xx* operating instructions and the *SIMATIC S7-300 Automation System, software installation manual*.
   These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation!
2. To continue, refer to steps 5 to 7 for distributed application.

8.6.2.4 Inserting function modules into the SIMOTION project

Requirement

The following requirements must be met when networking with PROFIBUS:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet (for distributed use only).

Note

For how to create a project and configure a PROFIBUS subnet, refer to the online help of *SIMOTION SCOUT*.

The following requirements must be met when with PROFINET:

1. You have created a project in SIMOTION SCOUT and have inserted and configured a rack with a PROFINET-compatible SIMOTION device in the hardware configuration.
2. You have configured a PROFINET IO System (for distributed use only).

Note

For how to create a project and configure a PROFINET IO system, refer to the online help of *SIMOTION SCOUT*.

Inserting FM 350-1, FM 350-2 or FM 352 (distributed application)

The description below is an example of networking via PROFIBUS.

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.
4. Open the **PROFIBUS DP** folder and the **ET 200M** subfolder in the hardware catalog. Select, for example, the **IM 153-1 interface module** (Article No.: 6ES7 153-1AA03-0XB0 or a successor module).
5. Use a drag-and-drop operation to place the IM 153-1 I/O device on the PROFIBUS subnet of your project. The **Properties - PROFIBUS IM 153-1 Interface** dialog box opens. In this dialog box, select the address you set on the IM 153-1 (see *ET 200M Distributed I/O Device* manual) and confirm with **OK**.
   The selected IM 153-1 I/O device is inserted in the project.
6. The inserted I/O device must now be fitted with your project modules. To do this, open the **FM300** subfolder below the selected I/O device in the hardware catalog and select the relevant **FM modules**.

   Note

   By default, the diagnostic alarms and process alarms are not enabled. Activate the alarms for each module in the **Properties** dialog box.

7. **Save** and **compile** your project.

8.6.2.5 Integrating the function blocks in the user project

Creating the FBs instance in the user project

The function blocks are part of the command library of the SIMOTION SCOUT engineering system. For working with the blocks, an instance has to be created in the user project for each function block used.

Example:

```
VAR_GLOBAL
...
    myInstFM3502Ctrl    : _FM3502_control;    // create FB instance
    myInstFM3501Ctrl    : _FM3501_control2;   // create FB instance
    myInstFM352Ctrl     : _FM352_control;     // create FB instance
...
END_VAR
```

Call (LAD representation)

The LAD representation of the individual function blocks can be found in the respective function block descriptions.

Application example

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

8.6.2.6 Creating I/O variables

Overview

Communication between the SIMOTION device and the FM 350-1, FM 350-2 and FM 352 takes place via direct I/O access and data set transfer. For data set transfer, the module address is transferred to the FB as an input parameter. I/O variables are used to address the direct read/write access to the I/O.

You can freely assign the names of I/O variables in SIMOTION SCOUT. I/O variables must be specified as ARRAY [0..15] of BYTE. You assign the address settings in the hardware configuration to these I/O variables.

The names of the I/O inputs must be transferred to the function blocks as call parameters (**periIn**). The prepared data for the I/O outputs are provided by the FB as in/out parameters (**periOut**). The in/out parameter must be supplied with a variable of type ARRAY [0..15] of BYTE. After the block is called, this variable must be assigned to the I/O variables for the I/O outputs (see call example in chapter "Calling the function blocks").

Note

The variable for supplying the in/out parameters **must not** be created as a temporary variable (VAR_TEMP or local variable of a function).

The following example shows how to assign the module addresses to the I/O variables in SIMOTION SCOUT.

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperipheralinputfm3501	PIB 256		Array	16
2	⊞ myperipheraloutputfm3501	PQB 256	☐	Array	16

Figure 8-31 Address assignment in SIMOTION SCOUT

Note

For additional information, see the following sources:

- *SIMOTION SCOUT* online help
- Programming manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* programming manual
  - *SIMOTION MCC, Motion Control Chart* programming manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* programming manual

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation!

8.6.3 Function blocks of the FM 350-1

8.6.3.1 Overview of the FM 350-1 function blocks

This section describes the function blocks (FBs) and the data structure required for parameter assignment, control and commissioning of the FM 350-1 module.

The function blocks form the software interface between the SIMOTION device and the FMs. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- Function block _FM3501_control2 (Page 6248)
- Function block _FM3501_diagnostic (Page 6252)

SIMOTION SCOUT contains all of the required FBs and data structure **Struct_FM3501_fmData** of the FM 350-1. The function blocks can be used to control one or more FM 350-1 modules.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC identifiers FM 350-1".

8.6.3.2 Function block _FM3501_control2

Introduction

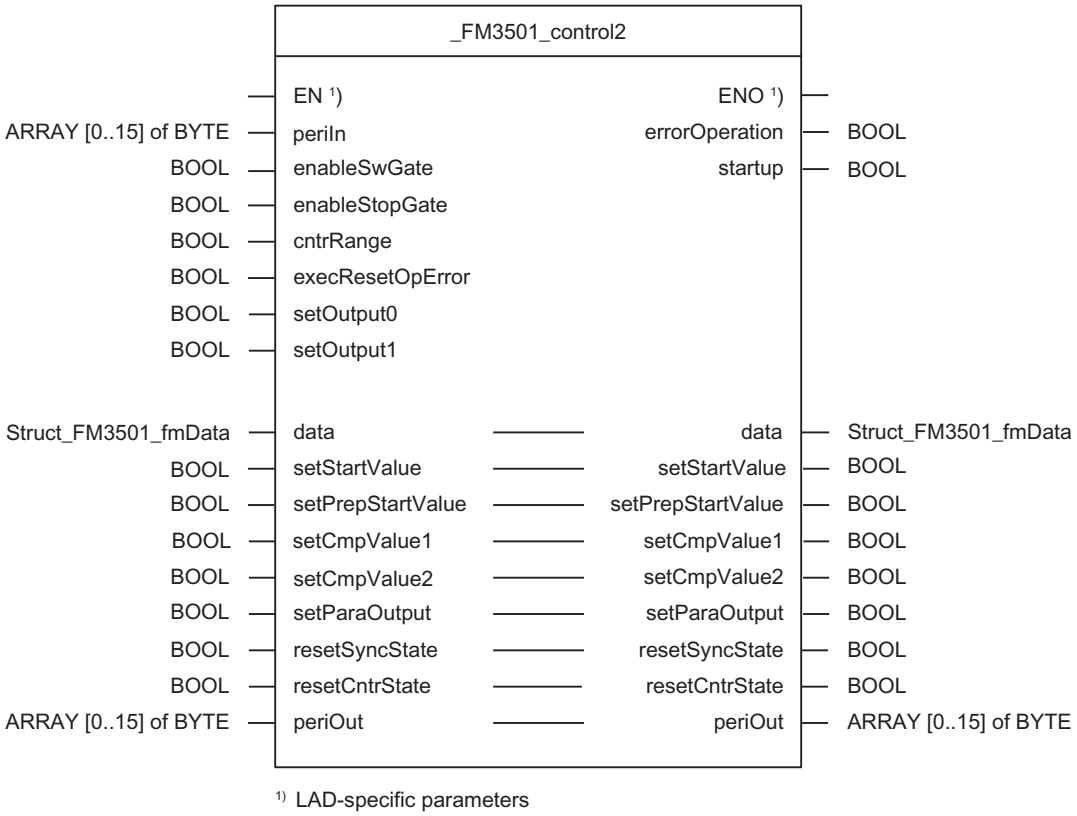
The **_FM3501_control2** function block can be used to control the module and to query the status of the FM 350-1.

Note

With SIMOTION V4.2 and higher, the **_FM3501_control2** function block with enhanced functions is available.

The **_FM3501_control** function block is still supported for compatibility reasons, although it does not enable the full scope of functions offered by the FM 350-1 to be activated.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-59 Parameters of the _FM3501_control2 function block

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
periIn	IN	ARRAY [0 to 15] of BYTE	Transfers I/O inputs of the FM to the FB	I/O variable of the I/O inputs of the FM transferred to the FB	Queried
enableSwGate	IN	BOOL	"SW gate (start/stop)" counter control bit	Sets and resets this	Queried
enableStopGate	IN	BOOL	"Stop gate" counter control bit	Sets and resets this	Queried
cntrRange	IN	BOOL	Counter range limit of the FM FALSE: $-2^{31} \leq$ count value $< 2^{31}-1$ TRUE: $0 \leq$ count value $< 2^{32}-1$	Sets and resets this	Queried
execResetOpError	IN	BOOL	Acknowledges operator error in the case of a rising edge	Sets and resets this	Queried
setOutput0	IN	BOOL	Setting/resetting of digital output DO0 of the FM 350-1	Sets and resets this	Queried
setOutput1	IN	BOOL	Setting/resetting of digital output DO1 of the FM 350-1	Sets and resets this	Queried
data	IN/OUT	Struct_FM3501 _fmData	Data structure	Entered and queried	Queried and entered
setStartValue	IN/OUT	BOOL	Count: Transfers "direct loading" trigger bit [2] Measuring: may not be used.	set	Queries and resets this
setPrepStartValue	IN/OUT	BOOL	Count: Transfers "preparatory loading" trigger bit [3] Measuring: Transmission of the lower limit	set	Queries and resets this
setCmpValue1	IN/OUT	BOOL	Count: Transfers "comparison value 1" trigger bit Measuring: Transmission of the upper limit	set	Queries and resets this
setCmpValue2	IN/OUT	BOOL	Count: Transfers "comparison value 2" trigger bit Measuring: Transmission of the update time	set	Queries and resets this

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
setParaOutput	IN/OUT	BOOL	Reassignment of parameters for behavior of digital outputs DO0 and DO1 of the FM 350-1, specification in following elements of data structure **Struct_FM3501_fmData**: configMeasOut0 configCntrOut0/ configCntrOut1 setCntrHyst setCntrPulse	set	Queries and resets this
resetSyncState	IN/OUT	BOOL	Deletes "synchronization" status bit	set	Queries and resets this
resetCntrState	IN/OUT	BOOL	Deletes "zero crossing" status bit	set	Queries and resets this
periOut	IN/OUT	ARRAY [0 to 15] of BYTE	Prepared data of the FB for the I/O outputs of the FM [4]	Queried and entered on the I/O variable for the I/O outputs	Entered
errorOperation	OUT	BOOL	An operator error has occurred	Queried	Sets and resets this
startup	OUT	BOOL	Indicates the startup of the FM	Queried	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] The **setStartValue** parameter specifies that the load value will be transferred to the load register and directly to the counter.

[3] The **setPrepStartValue** parameter specifies that the load value will be stored in the load register only. The load value in the load register will then be transferred at the next trigger (FM input "DI set" - set counter). The following must be satisfied:
- **enableReverseSetting = TRUE** (element of data structure **Struct_FM3501_fmData**) or
- **enableForwardSetting = TRUE** (element of data structure **Struct_FM3501_fmData**)

[4] **Note:** The **periOut** parameter must be supplied with an array of type **ARRAY [0 to 15] of BYTE**. Create a local or global array in your program under **VAR** (do not create a temporary array under VAR_TEMP). After the FB has been called, this array must be assigned to the I/O variable for the I/O outputs of the module. See FM 350-1 call example.

Functionality

The **_FM3501_control2** function block transfers data cyclically from a data structure of type **Struct_FM3501_fmData** to the FM 350-1. It also reads data from the FM 350-1 and enters this data in the elements of the data structure.

Note

Count:

The **cntrRange** input parameter must be set according to the assigned count range limits of the FM 350-1.

- **cntrRange** := **FALSE**, count range -231 ≤ count value < 231 - 1
  - loadValue1, cmpValue1_1, cmpValue2_1 are written from the FB to the FM
  - actValue1, actCntrValue1 are read from the FM
- **cntrRange** := **TRUE**, count range 0 ≤ count value < 232 - 1
  - loadValue2, cmpValue1_2, cmpValue2_2 are written from the FB to the FM
  - actValue2, actCntrValue2 are read from the FM

The same count ranges must be selected in the parameterization tools and in the data structure (**cntrRange**).

Measuring:

In measuring operating modes (frequency measurement, period measurement, speed measurement) the input parameter **cntrRange** := **TRUE** must be set.

Task integration (call)

The **_FM3501_control2** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

You can start a job for the FM 350-1 via the appropriate parameters **setStartValue**, **setPrepStartValue**, **setCmpValue1**, **setCmpValue2**, **setOutput0, setOutput1, setParaOutput**, **resetSyncState**, **resetCntrState**, **execResetOpError**, **enableStopGate**, or **enableSwGate**.

Depending on the job, the following values must be entered in the data structure before each call:

- during counting: the load value or the comparison value
- during measuring: the lower limit, the upper limit, or the update time
- when reassigning parameters for digital outputs: the specification of the behavior of the digital outputs

Once the job is carried out, the **_FM3501_control2** function block deletes any set in/out parameter (**setStartValue**, **setPrepStartValue**, **setCmpValue1**, **setCmpValue2**, **setParaOutput**, **resetSyncState**, or **resetCntrState**). This enables you to recognize that the job has been executed by the FM 350-1.

Startup behavior

As soon as the **_FM3501_control2** function block detects that the FM 350-1 is starting up, any pending job is deferred until after the startup is acknowledged. A startup of the FM 350-1 is indicated by output parameter **startup**=**TRUE**. Any deferred jobs are carried out once the startup is finished and are therefore not lost.

Error message during an FB call

If an error occurs during an FB call, it is indicated at the **errorOperation** block parameter. You can read out the error information in the **errorIdOperation** element of the data structure. You can acknowledge the error using the **execResetOpError** parameter.

Note

No new errors can be signaled until you have acknowledged the error.

Error numbers

The following error numbers can be displayed in the **errorIdOperation** element in the data structure.

Table 8-60 Error number assignment

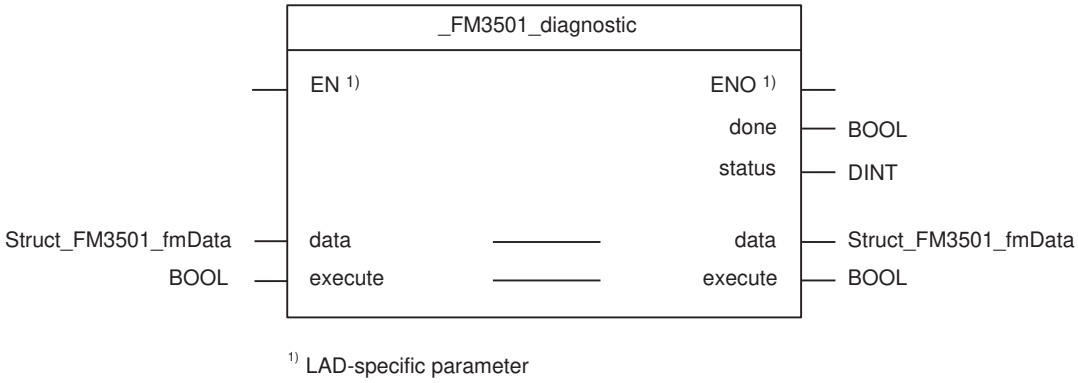
Error number	Meaning
0	No error
1	Operating mode cannot be started using the SW gate
2	Operating mode cannot be aborted
4	Only permitted if there is a pending output disable (OD)

8.6.3.3 Function block _FM3501_diagnostic

Introduction

The **_FM3501_diagnostic** function block enables you to read out the complete diagnostic data from the FM 350-1.

Call (LAD representation)



Parameter description

Table 8-61 Parameters of the _FM3501_diagnostic function block

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
data	IN/OUT	Struct_FM3501 _fmData	Data structure with counter data and diagnostic data	Entered and checked	Checked and entered
execute	IN/OUT	BOOL	Trigger bit for diagnostic data set	set	Checks and resets this
done	OUT	BOOL	Job completed without errors	Checked	Entered
status	OUT	DINT	Return value (error ID) 2) **_readRecord**	Checked	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

2) A detailed description is contained in the *SIMOTION System Function/Variables* parameter manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Functional description

Diagnostic data is read out from the **_FM3501_diagnostic** function block and made available in the associated **Struct_FM3501_fmData** data structure. The return value (error ID) can be read out at the **status** output parameter of the function block.

Sequence

Data is transferred as follows:

1. If in/out parameter **execute** = **TRUE** is set, the diagnostic data is read out from the FM 350-1.
2. The data are entered in data structure **data** of the **_FM3501_diagnostic** function block.

3. The return value (error ID) is copied to the **status** parameter of FB instance **_FM3501_diagnostic**.

Note

The return value (error ID) in the **status** parameter is present for one cycle only. The values 0x7001 and 0x7002 indicate that a data transfer has been initiated and is active.

4. As soon as the function has been executed, in/out parameter**execute** is reset.

Note

For the diagnostic sequence to be correct, the module address must be entered in the **moduleAddress** element of the data structure of type **Struct_FM3501_fmData**.

Task integration (call)

The **_FM3501_diagnostic** function block can be called in the **PeripheralFaultTask**, **BackgroundTask** or **TimerInterruptTask**. For performance reasons, the function block should not be called in the **PeripheralFaultTask**.

8.6.3.4 Data structure of the FM 350-1

Overview

The data structure of type **Struct_FM3501_fmData** contains the control and checkback signals of the FM 350-1 and the diagnostic data.

The data structure is used by the **_FM3501_control2** and **_FM3501_diagnostic** function blocks. Elements of the data structure are accessed using a variable of data type **Struct_FM3501_fmData**, which you must define yourself.

The **Struct_FM3501_fmData** data structure is shown in the table below.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC identifiers FM 350-1".

Table 8-62 Data structure of Struct_FM3501_fmData

Name	Type	Initial value	Counting	Measuring
General data				
xxxReserved1 [1)]	BYTE	16#00	Reserved	Reserved
xxxReserved2 [1)]	BYTE	16#00	Reserved	Reserved
moduleAddress	INT	256	Module address (see hardware configuration)	Module address (see hardware configuration)
xxxReserved3 [1)]	BYTE	16#00	Reserved	Reserved

Name	Type	Initial value	Counting	Measuring
Control signals				
loadValue1	DINT	0	New load value (**cntrRange** := **FALSE**)	-
cmpValue1_1	DINT	0	New comparison value 1 (**cntrRange** := **FALSE**)	-
cmpValue2_1	DINT	0	New comparison value 2 (**cntrRange** := **FALSE**)	-
loadValue2	UDINT	0	New load value (**cntrRange** := **TRUE**)	Lower limit
cmpValue1_2	UDINT	0	New comparison value 1 (**cntrRange** := **TRUE**)	Upper limit
cmpValue2_2	UDINT	0	New comparison value 2 (**cntrRange** := **TRUE**)	Update time
configMeasOut0	BYTE	16#00	-	"Measuring" operating mode Determining behavior of digital output DO0 when **setParaOut = TRUE** (see table titled "configMeas-Out0 structure")
configCntrOut0/ configCntrOut1	BYTE	16#00	"Counting" operating mode Determining behavior of digital outputs DO0 and DO1 when **setParaOut = TRUE** (see table titled "configCntrOut0/ configCntrOut1 structure")	-
setCntrHyst	USINT	0	Hysteresis (value range 0 to 255) in counts	-
setCntrPulse	USINT	0	Pulse duration (value range 0 to 250) in [ms]	-
xxxReserved4 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved5 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved6 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved7 [1)]	BOOL	FALSE	Reserved	Reserved
enableForwardSetting	BOOL	FALSE	Enable setting in forward direction	-
enableReverseSetting	BOOL	FALSE	Enable setting in reverse direction	-
xxxReserved8 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved9 [1)]	BOOL	FALSE	Reserved	Reserved
enableOutput0	BOOL	FALSE	Enable output 0	Enable output 0
enableOutput1	BOOL	FALSE	Enable output 1	Enable output 1
xxxReserved10 to 15 [1)]	BOOL	FALSE	Reserved	Reserved
Checkback signals				
actValue1	DINT	0	Current load or latch value (**cntrRange** := **FALSE**)	-
actCntrValue1	DINT	0	Current count value (**cntrRange** := **FALSE**)	-

Name	Type	Initial value	Counting	Measuring
actValue2	UDINT	0	Current load or latch value (**cntrRange** := **TRUE**)	Current measured value
actCntrValue2	UDINT	0	Current count value (**cntrRange** := **TRUE**)	Actual count value
errorIdData	WORD	16#0000	Specification of the data error	Specification of the data error
errorIdOperation	BYTE	16#00	Operator error (error number)	Operator error (error number)
xxxReserved16 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved17 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved18 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved19 [1)]	BOOL	FALSE	Reserved	Reserved
dataError	BOOL	FALSE	Data error (can be read out via the parameterization tool or in the "errorIdData" element)	Data error (can be read out via the parameterization tool or in the "errorIdData" element)
xxxReserved20 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved21 [1)]	BOOL	FALSE	Reserved	Reserved
parameterized	BOOL	FALSE	Module parameterized	Module parameterized
opState	BOOL	FALSE	Counter operation status	Counter operation status
opDirection	BOOL	FALSE	Count direction status	Count direction status
zeroCrossing	BOOL	FALSE	Zero crossing status	Full scale value
overflow	BOOL	FALSE	Status overflow	Status overflow
underflow	BOOL	FALSE	Status underflow	Status underflow
synchronized	BOOL	FALSE	Counter synchronization status	-
stateGate	BOOL	FALSE	Status of the internal gate	Status of the internal gate
stateSwGate	BOOL	FALSE	SW gate status	SW gate status
stateSetInput	BOOL	FALSE	Status of digital input "DI set"	Status of digital input "DI set"
xxxReserved22 [1)]	BOOL	FALSE	Reserved	Reserved
stateOfDiStart	BOOL	FALSE	Status of digital input "DI start"	Status of digital input "DI start"
stateOfDiStop	BOOL	FALSE	Status of digital input "DI stop"	Status of digital input "DI stop"
stateCompValue1	BOOL	FALSE	Status of output of comparison value 1	Status of output of comparison value 1
stateCompValue2	BOOL	FALSE	Status of output of comparison value 2	Status of output of comparison value 2
remStateCompValue1	BOOL	FALSE	Saved (retentive) status of comparator 1	-
remStateCompValue2	BOOL	FALSE	Saved (retentive) status of comparator 2	-
xxxReserved23..28 [1)]	BOOL	FALSE	Reserved	Reserved
xxxReserved29 [1)]	DINT	0	Reserved	Reserved
xxxReserved30 [1)]	DINT	0	Reserved	Reserved
Diagnostic data				
faultModule	BOOL	FALSE	Module fault	
internFault	BOOL	FALSE	Internal fault	
extFault	BOOL	FALSE	External fault	

Name	Type	Initial value	Counting	Measuring
faultChannel	BOOL	FALSE	Channel fault (for decoding, refer to elements starting with **chType**)	
faultExtVoltage	BOOL	FALSE	Auxiliary voltage fault	
faultConnector	BOOL	FALSE	Front connector	
invalidConfig	BOOL	FALSE	Parameter assignment missing	
invalidPara	BOOL	FALSE	Parameter assignment faulty	
moduleType	BYTE	16#00	Module type	
faultSubModule	BOOL	FALSE	Incorrect/missing interface module	
faultCommunication	BOOL	FALSE	Communication error	
moduleStop	BOOL	FALSE	RUN/STOP mode display	
faultWatchdog	BOOL	FALSE	Watchdog (FM)	
faultIntPower	BOOL	FALSE	Internal power supply fault	
xxxReserved47 [1)]	BOOL	FALSE	Reserved	
xxxReserved48 [1)]	BOOL	FALSE	Reserved	
xxxReserved31 [1)]	BOOL	FALSE	Reserved	
faultRack	BOOL	FALSE	Rack fault	
faultDevice	BOOL	FALSE	SIMOTION device fault	
faultEprom	BOOL	FALSE	EPROM fault	
faultRam	BOOL	FALSE	RAM fault	
faultAdc	BOOL	FALSE	ADC fault	
faultFuse	BOOL	FALSE	Fuse fault	
lostProcessAlarm	BOOL	FALSE	Process alarm lost	
xxxReserved32 [1)]	BOOL	FALSE	Reserved	
chType	BYTE	16#00	Channel type	
lenDiagData	BYTE	16#00	Length of diagnostic data per channel	
chNumber	BYTE	16#00	Channel number	
groupErrorChannel1	BOOL	FALSE	Group error channel 1	
xxxGroupErrorChannel2 [1)]	BOOL	FALSE	Group error channel 2	
xxxReserved33...38 [1)]	BOOL	FALSE	Reserved	
faultCh1SignalA	BOOL	FALSE	Channel 1, signal A fault	
faultCh1SignalB	BOOL	FALSE	Channel 1, signal B fault	
faultCh1SigZero	BOOL	FALSE	Channel 1, signal zero fault	
faultChannel1	BOOL	FALSE	Channel 1, fault between channels	
faultCh1EncSupply	BOOL	FALSE	Channel 1, 5.2 V encoder supply fault	
xxxReserved39..41 [1)]	BOOL	FALSE	Reserved	
xxxReserved42 [1)]	BYTE	16#00	Reserved	
faultCh2SignalA	BOOL	FALSE	Channel 2, signal A fault	
faultCh2SignalB	BOOL	FALSE	Channel 2, signal B fault	
faultCh2SigZero	BOOL	FALSE	Channel 2, signal zero fault	
faultChannel2	BOOL	FALSE	Channel 2, fault between channels	
faultCh2EncSupply	BOOL	FALSE	Channel 2, 5.2 V encoder supply fault	
xxxReserved43..45 [1)]	BOOL	FALSE	Reserved	
xxxReserved46 [1)]	BYTE	16#00	Reserved	

[1] Variable for internal FB use (not relevant to users)

configMeasOut0 structure

In "Measuring" operating mode, the behavior of digital output DO0 for **setParaOutput = TRUE** is defined as follows in parameter **configMeasOut0**:

Table 8-63 configMeasOut0 structure

Bits 2 to 7	Bit 1	Bit 0	Behavior of digital output DO0
x	0	0	Inactive
x	0	1	Outside the limits
x	1	0	Below the lower limit
x	1	1	Above the upper limit

configCntrOut0/configCntrOut1 structure

In "Counting" operating mode, the behavior of digital outputs DO0 and DO1 for **setParaOutput = TRUE** is defined as follows in parameter **configCntrOut0/configCntrOut1**:

Table 8-64 configCntrOut0/configCntrOut1 structure

Bits 3 to 7	Bit 2	Bit 1	Bit 0	Behavior of digital outputs DO0/DO1
x	0	0	0	Inactive
x	0	0	1	Active within the range from comparison value to overflow
x	0	1	0	Active within the range from comparison value to underflow
x	0	1	1	Active on reaching the comparison value for pulse duration (up/down)
x	1	0	0	Active on reaching the comparison value for pulse duration up
x	1	0	1	Active on reaching the comparison value of pulse duration down

8.6.3.5 Calling function blocks

In order to be able to work with the function blocks in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for FB **_FM3501_control2**).
2. Set up variables for the data structure.
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.

6. The output parameters of the FB are accessed with <instance name of FB>. <name of output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variables created in step 3.

Note

The call example is an extract from the E_FM3501 application example supplied, which can be found on the "SIMOTION Utilities & Applications" CD-ROM.

If you wish to control more than one FM 350-1, you must create a new variable for the data structure and FB instances with a new name for each FM 350-1 used.

Call example

```
UNIT E_FM3501;
INTERFACE
    VAR_GLOBAL
    myDataFM3501 : Struct_FM3501_fmData;      // Create variable of data structure              (2)
    // Following variables are - set by application to activate function;
    // - reset by FB to signal completion of function.

    MyLoadStartValue         : BOOL;             // Load load value directly
    MyLoadPrepareStartValue  : BOOL;             // Load load value in preparation
    ...

    // INPUT VARIABLES
    MySetSoftwareGate        : BOOL;             // Software gate
    MyStopGate               : BOOL;             // Stop gate
    ...

    // OUTPUT VARIABLES
    MyOperationError         : BOOL;             // Error in FB _FB_FM3501_control2
    MyStateFMStartup         : BOOL;             // Start-up status

    myInstFM3501Ctrl         : _FM3501_control2; // create FB instance                         (1)

    END_VAR

END_INTERFACE

IMPLEMENTATION

PROGRAM ExampleFM3501                            // Program in BackroundTask

VAR
```

```
    FMOutputArray      : ARRAY [0..15] of BYTE;  // Array for FM output data                    (3)
END_VAR

// CALL INSTANCE of _FM3501_control2                                                            (4)
  myInstFM3501Ctrl(
    EnableSwGate       := mySetSoftwareGate,        // Control software gate                    (5)
    EnableStopGate     := myStopGate,               // Control internal gate
    ExecResetOpError   := myResetError,             // Acknowledge operator error
    CntrRange          := TRUE,                     // Counter range
    setOutput0         := mySetOutput0,             // Control digital otput
    setOutput1         := mySetOutput1,             // Control digital otput
    PeriIn             := myPeripheralInputFM3501,  // input address
    Data               := myDataFM3501,             // Transfer data structure
    PeriOut            := FMOutputArray,            // FM output data array

  // following IN_OUT parameters are set by the application
  // and reset by the FB! (shake-hand-effect)
    SetStartValue      := myLoadStartValue,         // Load counter
    SetPrepStartValue  := myLoadPrepareStartValue,  // Load counter in preparation
    setCmpValue1       := myLoadComparisonValue1,   // Load new comparison value 1
    setCmpValue2       := myLoadComparisonValue2,   // Load new comparison value 2
    resetSyncState     := myResetSyncState,         // Reset synchronization bit
    resetCntrState     := myResetCounterState,      // Reset status bit
    setParaOutput      := mySetParaOutput           // set new configuration
                                                    // data of digital outputs
    );

// TRANSFER DATA TO FM
    myPeripheralOutputFM3501 := FMOutputArray;   // Assign array of FM output data             (7)
                                                 // to I/O variables

// EVALUATE AND DISPLAY STATUS MESSAGES
    MyStateFMStartup   := myInstFM3501Ctrl.startup;          // Start-up status               (6)
    MyOperationError   := myInstFM3501Ctrl.errorOperation;

END_PROGRAM           // ExampleFM3501
END_IMPLEMENTATION
```

Note

The PROGRAM ExampleFM3501 must be assigned in the execution system.

8.6.3.6 Application example for FM 350-1

Introduction

The following example uses the "Transfer load value to FM 350-1" and "Start counter" functions to show how the **_FM3501_control2** function block can be applied. This example is representative for all of the functions of this module. A call example for the **_FM3501_diagnostic** function block is located in the diagnostic section (PeripheralFaultTask).

Table 8-65 Input symbols used

Symbol	Data type	Description
myLoadStartValue	BOOL	Directly load the load value
myLoadPrepareStartValue	BOOL	Load the load value in preparation
myLoadComparisonValue1	BOOL	Transfer comparison value 1
myLoadComparisonValue2	BOOL	Transfer comparison value 2
myResetSyncState	BOOL	Reset the synchronization status bit
myResetCntrState	BOOL	Reset the status bit for zero crossing/overflow/underflow
mySetSoftwareGate	BOOL	Software gate
myEnableStopGate	BOOL	Stop gate
myResetError	BOOL	Acknowledge error
myResetDiagnosticAlarm	BOOL	Acknowledge diagnostic alarm
myResetProcessAlarm	BOOL	Acknowledge process alarm

Table 8-66 Output symbols used

Symbol	Data type	Description
myErrorOperation	BOOL	Error in the **_FM3501_control2** function block
myStateFMStartup	BOOL	Startup status
myLoadStartValueActive	BOOL	Load count value active
myLoadPrepareStartValueActive	BOOL	Load count value in preparation active
myLoadComparisonValue1Active	BOOL	Load comparison value 1 active
myLoadComparisonValue2Active	BOOL	Load comparison value 2 active
myResetSyncActive	BOOL	Reset synchronization status bit active
myResetCounterStateActive	BOOL	Reset zero crossing/overflow/underflow status bit active
myDiagnosticAlarm	BOOL	Receive diagnostic alarm
myProcessAlarm	BOOL	Receive process alarm
myStateCounter	BOOL	Counter running (opState)
myStateDirection	BOOL	Direction bit (opDirection)
myStateZeroCrossing	BOOL	Zero crossing
myCounterOverflow	BOOL	Counter overflow
myCounterUnterflow	BOOL	Counter underflow
myStateSwGate	BOOL	Software gate (stateSwGate)
myStateGate	BOOL	Internal gate (stateGate)

Note

Depending on the type of signal used, you should pay attention to the coding plug of the FM 350-1 (for example, position D for 24 V signals).

You can either monitor and modify the input and output variables used in the programming example in the INTERFACE section of the unit (under VAR_GLOBAL) using the symbol browser, or you can assign real inputs and outputs to the input and output variables in your unit.

Contents of example

When the **_FM3501_control2** function block is called, the control and checkback signals are exchanged cyclically between the SIMOTION device (C230-2, P350, D435) and the FM 350-1. All of the data that are relevant to the module are located in data structure "dataFM3501".

Depending on the configuration of the FM 350-1 (operating mode, use of gates, alarm configuration, etc.), the FM 350-1 counts the pulses at the wired encoder signal input if, for example, the value of the "myStateSwGate" input is "TRUE". The counting operation stops if input "myStateSwGate" = FALSE or if input "myEnableStopGate" = TRUE.

Transferring the load values

Two parameters are available for transferring the load value to the FM 350-1. When the **_FM3501_control2** FB is called, either the "myLoadStartValue" or "myLoadPrepareStartValue" parameter is selected on a rising edge. The "myLoadStartValue" parameter specifies that the load value will be transferred to the load register and directly to the counter (you must set input "myLoadStartValue" = TRUE).
The "myLoadPrepareStartValue" parameter specifies that the load value will be stored in the load register only (you must set trigger bit "myLoadPrepareStartValue" = TRUE in your user program). The load value in the load register is then transferred at the next event (FM input "DI set") that sets the counter.
The FB must be called until the FB has reset the selected trigger bit ("myLoadStartValue" or "myLoadPrepareStartValue").
The in/out parameter remains set while the transfer is active. If the trigger bit you set has been reset by the **_FM3501_control2** function block, the FM 350-1 has received the load value.

Loading comparison values

New comparison values are transferred to the FM by setting the "myLoadComparisonValue1" or "myLoadComparisonValue2" inputs (rising edge).

Deleting status bits

The synchronization status bit is reset by setting the "myResetSyncState" input (rising edge) and the zero crossing/overflow/underflow status bit is reset by setting the "myResetCntrState" input.

Process alarm / diagnostic alarm

If a process alarm or diagnostic alarm is triggered by the FM 350-1, this is indicated by the "myProcessAlarm" and "myDiagnosticAlarm" variable, respectively. If the "PeripheralFaultFM3501" program is integrated in the PeripheralFaultTask, this task is started and the most important task start information is stored temporarily in the "myAlarmDetails", "myIogAddressIn" and "myAlarmInterrupt" variables. If a diagnostic alarm has been signaled, the **_FM3501_diagnostic** FB is started, which reads out detailed diagnostic information from the module. These diagnostic data are then located in data structure "dataFM3501". Setting the

"myResetDiagnosticAlarm" input variable (to acknowledge a diagnostic alarm) or the "myResetProcessAlarm" input variable (to acknowledge a process alarm) acknowledges the respective alarms.

The respective states are signaled by the output variables used (see Table "Output symbols used").

Hardware platform

The application example is available for various SIMOTION hardware platforms and is intended for distributed use of the FM 350-1.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the hardware configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and module address in the program (dataFM3501.moduleAddress).
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

8.6.4 Function blocks of the FM 350-2

8.6.4.1 Overview of the FM 350-2 function blocks

This section describes the function blocks (FBs) and the data structure required for parameter assignment, control and commissioning of the FM 350-2 module.

The function blocks form the software interface between the SIMOTION device and the FMs. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- Function block _FM3502_control (Page 6264)
- Function block _FM3502_write (Page 6266)
- Function block _FM3502_read (Page 6267)
- Function block _FM3502_diagnostic (Page 6269)

SIMOTION SCOUT contains all of the required FBs and data structure **Struct_FM3502_fmData** of the FM 350-2. The function blocks can be used to control one or more FM 350-2 modules.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC identifiers FM 350-2".

Note

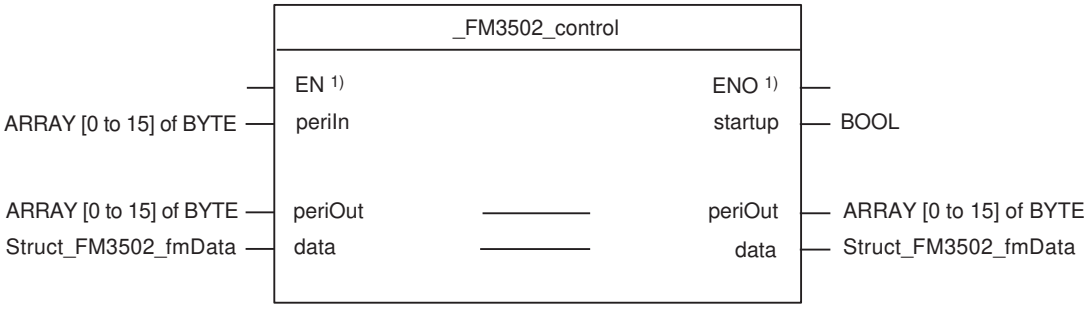
The online functions of the parameter assignment tool in STEP 7 **HW Config** can only be used for diagnostic purposes (read-only access to the module). Write access (control function) has no effect. The parameters set by the program can be read out using the parameter assignment tool.

8.6.4.2 Function block _FM3502_control

Introduction

The **_FM3502_control** function block can be used to control the module and to query the status of the FM 350-2.

Call (LAD representation)



1) LAD-specific parameter

Parameter description

Table 8-67 Parameters of the _FM3502_control function block

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
periIn	IN	ARRAY [0 to 15] of BYTE	Transfers I/O inputs of the FM to the FB	I/O variable of the I/O inputs of the FM transferred to the FB	Queried
periOut	IN/OUT	ARRAY [0 to 15] of BYTE	Prepared data of the FB for the I/O outputs of the FM [2]	Queried and entered on the I/O variable for the I/O outputs	Entered
data	IN/OUT	Struct_FM3502_fmData	Data structure with channel-specific data	Entered and queried	Queried and entered
startup	OUT	BOOL	Indicates the startup of the FM	Queried	Entered

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] **Note:** The **periOut** parameter must be supplied with an array of type **ARRAY [0 to 15] of BYTE**. Create a local or global array in your program under **VAR** (do not create a temporary array under VAR_TEMP). After the FB has been called, this array must be assigned to the I/O variable for the I/O outputs of the module. See FM 350-2 call example!

Functional description

The **_FM3502_control** function block cyclically transfers the control signals from the **control** substructure of the data structure of type **Struct_FM3502_fmData** to the FM 350-2. In addition, it reads the checkback signals from the FM 350-2 and enters these into the **checkback** substructure of the data structure of type **Struct_FM3502_fmData**.

The **_FM3502_control** function block is absolutely essential for operation of the FM 350-2.

Task integration (call)

The**_FM3502_control** function block must be called cyclically via the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

Before the call, you enter the current control signals in the **control** substructure of the **Struct_FM3502_fmData** data structure. After the call, the checkback signals are updated in the **checkback** substructure of the data structure. They can then be processed further.

The **_FM3502_control** function block must be called cyclically for **each** FM 350-2 integrated in the project.

Startup behavior

The **_FM3502_control** function block performs startup coordination with the FM 350-2. A startup of the FM 350-2 is indicated by output parameter **startup** = **TRUE**.
After startup has been acknowledged, the control and checkback signals are exchanged with the FM 350-2.

8.6.4.3 Function block _FM3502_write

Introduction

The **_FM3502_write** function block executes write jobs (for example, loading count values and comparison values) to the FM 350-2.

Call (LAD representation)

1) LAD-specific parameter

Parameter description

Table 8-68 Parameters of the _FM3502_write function block

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
periIn	IN	ARRAY [0 to 15] of BYTE	Transfers I/O inputs of the FM to the FB	I/O variable of the I/O inputs of the FM transferred to the FB	Queried
periOut	IN/OUT	ARRAY [0 to 15] of BYTE	Prepared data of the FB for the I/O outputs of the FM 2)	Queried and entered on the I/O variable for the I/O outputs	Entered
data	IN/OUT	Struct_FM3502_fmData	Data structure with channel-specific data	Entered and queried	Queried and entered
error	OUT	BOOL	Job completed with errors	Queried	Entered
status	OUT	DINT	Error ID 3) **_writeRecord**	Queried	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

2) **Note:** The **periOut** parameter must be supplied with an array of type **ARRAY [0 to 15] of BYTE**. Create a local or global array in your program under **VAR** (do not create a temporary array under VAR_TEMP). After the FB has been called, this array must be assigned to the I/O variable for the I/O outputs of the module. See FM 350-2 call example!

3) A detailed description is contained in the *SIMOTION System Function/Variables* List Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Functional description

The **_FM3502_write** function block loads the counters and comparators of the FM 350-2 from the data structure of type **Struct_FM3502_fmData** by means of a write job.

The **_FM3502_write** function block should only be called when executing write jobs.

Task integration (call)

The **_FM3502_write** function block can be called via the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

Before executing a write job, you must supply the appropriate values in the data range associated with the write job. A write job is triggered by assigning the job number in the **write.execJobNumber** element. The **_FM3502_write** function block must continue to be called cyclically until the **write.execJobNumber** element is zero. The last write job must be complete before a new write job can be executed, i.e. **write.execJobNumber** is deleted.

Note

To ensure the correct sequence, the module address must be entered (in "general data") in the **moduleAddress** element of the data structure of type **Struct_FM3502_fmData**.

Startup behavior

The **_FM3502_write** function block does not perform startup coordination with the FM 350-2. During the startup phase, job execution is disabled. Any pending jobs are not lost, but they are not executed until the startup has been acknowledged.

Error message during a call

If an error occurs during a call, it is reported in the **status** output parameter.

Note

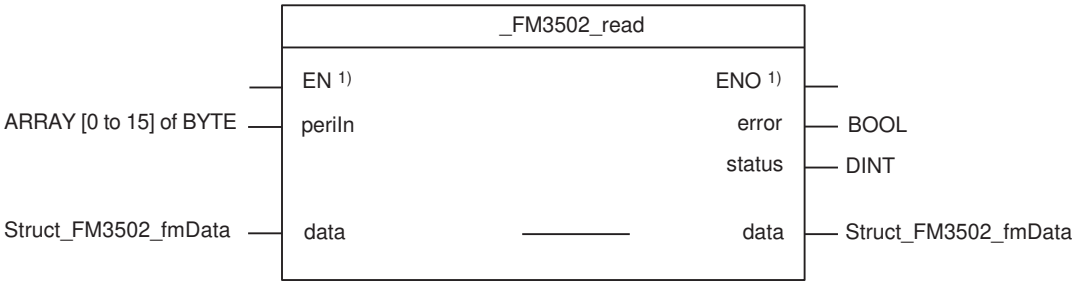
The return value (error ID) in the **status** parameter is present for one cycle only. The values 0x7001 and 0x7002 indicate that data transfer has been initiated and is active.

8.6.4.4 Function block _FM3502_read

Introduction

The **_FM3502_read** function block is used to read out the count values and measured values of the FM 350-2.

Call (LAD representation)



1) LAD-specific parameter

Parameter description

Table 8-69 Parameters of the _FM3502_read function block

Name	P type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
periIn	IN	ARRAY [0 to 15] of BYTE	Transfers I/O inputs of the FM to the FB	I/O variable of the I/O inputs of the FM transferred to the FB	Queried
data	IN/OUT	Struct_FM3502_fmData	Data structure with channel-specific data	Entered and queried	Queried and entered
error	OUT	BOOL	Job completed with errors	Queried	Entered
status	OUT	DINT	Error ID 3) **_readRecord**	Queried	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

2) **Note:** The **periOut** parameter must be supplied with an array of type **ARRAY [0 to 15] of BYTE**. Create a local or global array in your program under **VAR** (do not create a temporary array under VAR_TEMP). After the FB has been called, this array must be assigned to the I/O variable for the I/O outputs of the module. See FM 350-2 call example!

3) A detailed description is contained in the *SIMOTION System Function/Variables list manual*. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Functional description

The **_FM3502_read** function block executes the read jobs entered in the **read.execJobNumber** element and transfers the read data to the data structure of type **Struct_FM3502_fmData**.

The **_FM3502_read** function block should only be called when executing read jobs.

Task integration (call)

The **_FM3502_read** function block can be called via the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

A read job is triggered by assigning the job number in the **read.execJobNumber** element. The **_FM3502_read** function block must continue to be called cyclically until the

read.execJobNumber element is zero. The current read job must be complete before a new read job can be executed, i.e. **read.execJobNumber** is deleted.

Note

To ensure the correct sequence, the module address must be entered (in "general data") in the **moduleAddress** element of the data structure of type **Struct_FM3502_fmData**.

Startup behavior

The **_FM3502_read** function block does not perform startup coordination with the FM 350-2. During the startup phase, job execution is disabled. Any pending jobs are not lost, but they are not executed until the startup has been acknowledged.

Error message during a call

If an error occurs during a call, it is reported in the **status** output parameter.

Note

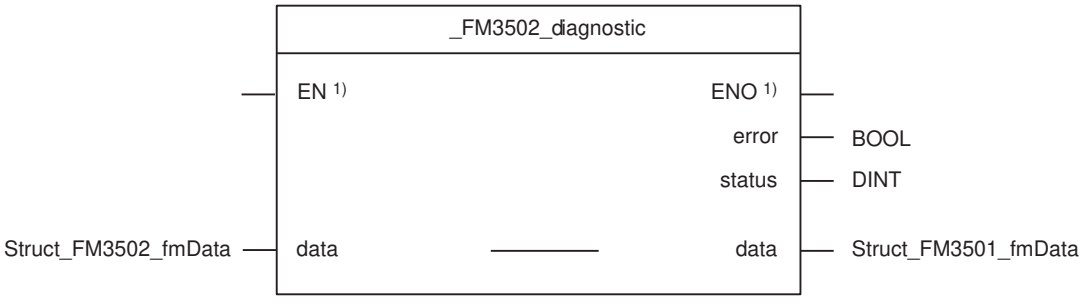
An error ID in **status** is present for one cycle only. The values 0x7001 and 0x7002 indicate that data transfer has been initiated and is active.

8.6.4.5 Function block _FM3502_diagnostic

Introduction

The **_FM3502_diagnostic** function block enables you to read out the complete diagnostic data from the FM 350-2.

Call (LAD representation)



1) LAD-specific parameter

Parameter description

Table 8-70 Parameters of the _FM3502_diagnostic function block

Name	P type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
data	IN/OUT	Struct_FM3502_fmData	Data structure with channel-specific data	Entered and checked	Checked and entered
error	OUT	BOOL	Request completed with errors	Checked	Entered
status	OUT	DINT	Return value (error ID) [2] **_readRecord**	Checked	Entered

[1] Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

[2] A detailed description is contained in the *SIMOTION System Function/Variables* parameter manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Functional description

The entire diagnostic data are read out by the **_FM3502_diagnostic** function block and made available in the **diagnostic** substructure of the **Struct_FM3502_fmData** data structure.

The return value (error ID) can be read out at the **status** output parameter of the function block.

Sequence

Data is transferred as follows:

1. When the **_FM3502_diagnostic** function block is called, data transfer is enabled and the data are transferred. You can view the error ID at the **status** output parameter.

   Note

   The return value (error ID) in the **status** parameter is present for one cycle only. The values 0x7001 and 0x7002 indicate that a data transfer has been initiated and is active.

2. The data are entered in the data structure of type **Struct_FM3502_fmData**.
3. The return value (error ID) is provided at the **status** parameter of the **_FM3502_diagnostic** function block.
4. The reading out of diagnostic data is finished when **status** = 0 is signaled.

   Note

   To ensure the correct sequence, the module address must be entered (in "general data") in the **moduleAddress** element of the data structure of type **Struct_FM3502_fmData**.

Task integration (call)

The **_FM3502_diagnostic** function block can be called in the **PeripheralFaultTask**, **BackgroundTask** or **TimerInterruptTask**. For performance reasons, the function block should only be called in the **PeripheralFaultTask**.

8.6.4.6 Data structures of the FM 350-2

Overview

The data structure of type **Struct_FM3502_fmData** contains all data of the FM 350-2 relevant for operation, as well as diagnostic data.

The data structures are used by the the following function blocks: **_FM3502_control**, **_FM3502_write**, **_FM3502_read** and **_FM3502_diagnostic**. Elements in the data structure are accessed using a variable of data type **Struct_FM3502_fmData**, which you must define yourself.

The **Struct_FM3502_fmData** data structure is shown in the table below.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC identifiers FM 350-2".

Table 8-71 Data structure of Struct_FM3502_fmData

Struct_FM3502_fmData			
Name	**Type**	**Initial value**	**Comment**
Write job (data structure)			
write	**Struct_FM3502_wrJob**		Write job elements
execJobNumber	BYTE	16#00	Number
busy	BOOL	FALSE	Write job in progress
done	BOOL	FALSE	Write job finished
invalid	BOOL	FALSE	Write job not possible
unknown	BOOL	FALSE	Write job unknown
Read job (data structure)			
read	**Struct_FM3502_rdJob**		Read job elements
execJobNumber	BYTE	16#00	Number
busy	BOOL	FALSE	Read job in progress
done	BOOL	FALSE	Read job finished
invalid	BOOL	FALSE	Read job not possible
unknown	BOOL	FALSE	Read job unknown
General data			
xxxReserved1 [1)]	ARRAY [1..3] of WORD		Reserved
xxxReserved2 [1)]	WORD	16#0000	Reserved
moduleAddress	INT	256	Module address
xxxReserved3 [1)]	BYTE	16#00	Reserved
Control signals (data structure)			

Struct_FM3502_fmData			
Name	**Type**	**Initial value**	**Comment**
control	**Struct_FM3502_control**		Elements for control signals
xxxReserved4..11 [1)]	BOOL	FALSE	Reserved
enableOutput0	BOOL	FALSE	Output 0 enabled
enableOutput1	BOOL	FALSE	Output 1 enabled
enableOutput2	BOOL	FALSE	Output 2 enabled
enableOutput3	BOOL	FALSE	Output 3 enabled
enableOutput4	BOOL	FALSE	Output 4 enabled
enableOutput5	BOOL	FALSE	Output 5 enabled
enableOutput6	BOOL	FALSE	Output 6 enabled
enableOutput7	BOOL	FALSE	Output 7 enabled
setOutput0	BOOL	FALSE	Set output 0
setOutput1	BOOL	FALSE	Set output 1
setOutput2	BOOL	FALSE	Set output 2
setOutput3	BOOL	FALSE	Set output 3
setOutput4	BOOL	FALSE	Set output 4
setOutput5	BOOL	FALSE	Set output 5
setOutput6	BOOL	FALSE	Set output 6
setOutput7	BOOL	FALSE	Set output 7
enableSwGate0	BOOL	FALSE	Open SW gate counter 0
enableSwGate1	BOOL	FALSE	Open SW gate counter 1
enableSwGate2	BOOL	FALSE	Open SW gate counter 2
enableSwGate3	BOOL	FALSE	Open SW gate counter 3
enableSwGate4	BOOL	FALSE	Open SW gate counter 4
enableSwGate5	BOOL	FALSE	Open SW gate counter 5
enableSwGate6	BOOL	FALSE	Open SW gate counter 6
enableSwGate7	BOOL	FALSE	Open SW gate counter 7
xxxReserved12 [1)]	DWORD	16#0000 0000	Reserved
xxxReserved13 [1)]	DWORD	16#0000 0000	Reserved
xxxReserved14 [1)]	DWORD	16#0000 0000	Reserved
Checkback signals (data structure)			
checkback	**Struct_FM3502_checkback**		Elements for checkback signals
xxxReserved15 [1)]	BOOL	FALSE	Reserved
testModePg	BOOL	FALSE	Test mode is selected on the parameter assignment tool
xxxReserved16..17 [1)]	BOOL	FALSE	Reserved
dataError	BOOL	FALSE	Data error (can be read out via parameterization tool)
xxxReserved18..19 [1)]	BOOL	FALSE	Reserved
parameterized	BOOL	FALSE	Module parameterized
stateCmpValue0	BOOL	FALSE	Comparator 0 addressed
stateCmpValue1	BOOL	FALSE	Comparator 1 addressed

Struct_FM3502_fmData			
Name	**Type**	**Initial value**	**Comment**
stateCmpValue2	BOOL	FALSE	Comparator 2 addressed
stateCmpValue3	BOOL	FALSE	Comparator 3 addressed
stateCmpValue4	BOOL	FALSE	Comparator 4 addressed
stateCmpValue5	BOOL	FALSE	Comparator 5 addressed
stateCmpValue6	BOOL	FALSE	Comparator 6 addressed
stateCmpValue7	BOOL	FALSE	Comparator 7 addressed
cntr0Underflow	BOOL	FALSE	Underflow counter 0
cntr1Underflow	BOOL	FALSE	Underflow counter 1
cntr2Underflow	BOOL	FALSE	Underflow counter 2
cntr3Underflow	BOOL	FALSE	Underflow counter 3
cntr4Underflow	BOOL	FALSE	Underflow counter 4
cntr5Underflow	BOOL	FALSE	Underflow counter 5
cntr6Underflow	BOOL	FALSE	Underflow counter 6
cntr7Underflow	BOOL	FALSE	Underflow counter 7
cntr0Overflow	BOOL	FALSE	Overflow counter 0
cntr1Overflow	BOOL	FALSE	Overflow counter 1
cntr2Overflow	BOOL	FALSE	Overflow counter 2
cntr3Overflow	BOOL	FALSE	Overflow counter 3
cntr4Overflow	BOOL	FALSE	Overflow counter 4
cntr5Overflow	BOOL	FALSE	Overflow counter 5
cntr6Overflow	BOOL	FALSE	Overflow counter 6
cntr7Overflow	BOOL	FALSE	Overflow counter 7
cntr0Reverse	BOOL	FALSE	Reverse counting direction for counter 0
cntr1Reverse	BOOL	FALSE	Reverse counting direction for counter 1
cntr2Reverse	BOOL	FALSE	Reverse counting direction for counter 2
cntr3Reverse	BOOL	FALSE	Reverse counting direction for counter 3
cntr4Reverse	BOOL	FALSE	Reverse counting direction for counter 4
cntr5Reverse	BOOL	FALSE	Reverse counting direction for counter 5
cntr6Reverse	BOOL	FALSE	Reverse counting direction for counter 6
cntr7Reverse	BOOL	FALSE	Reverse counting direction for counter 7
input0	BOOL	FALSE	Digital input 0 active/not active
input1	BOOL	FALSE	Digital input 1 active/not active
input2	BOOL	FALSE	Digital input 2 active/not active
input3	BOOL	FALSE	Digital input 3 active/not active
input4	BOOL	FALSE	Digital input 4 active/not active
input5	BOOL	FALSE	Digital input 5 active/not active
input6	BOOL	FALSE	Digital input 6 active/not active
input7	BOOL	FALSE	Digital input 7 active/not active
output0	BOOL	FALSE	Digital output 0 active/not active
output1	BOOL	FALSE	Digital output 1 active/not active
output2	BOOL	FALSE	Digital output 2 active/not active
output3	BOOL	FALSE	Digital output 3 active/not active

Struct_FM3502_fmData			
Name	**Type**	**Initial value**	**Comment**
output4	BOOL	FALSE	Digital output 4 active/not active
output5	BOOL	FALSE	Digital output 5 active/not active
output6	BOOL	FALSE	Digital output 6 active/not active
output7	BOOL	FALSE	Digital output 7 active/not active
gate0	BOOL	FALSE	Internal gate counter 0 open/closed
gate1	BOOL	FALSE	Internal gate counter 1 open/closed
gate2	BOOL	FALSE	Internal gate counter 2 open/closed
gate3	BOOL	FALSE	Internal gate counter 3 open/closed
gate4	BOOL	FALSE	Internal gate counter 4 open/closed
gate5	BOOL	FALSE	Internal gate counter 5 open/closed
gate6	BOOL	FALSE	Internal gate counter 6 open/closed
gate7	BOOL	FALSE	Internal gate counter 7 open/closed
opValue0	WORD	16#0000	Depending on assigned count value/measured value 0...3 (is updated each time the **_FM3502_control** FB is called)
opValue1	WORD	16#0000	
opValue2	WORD	16#0000	
opValue3	WORD	16#0000	
loadValue0	DINT	0	Load counter 0 directly
loadValue1	DINT	0	Load counter 1 directly
loadValue2	DINT	0	Load counter 2 directly
loadValue3	DINT	0	Load counter 3 directly
loadValue4	DINT	0	Load counter 4 directly
loadValue5	DINT	0	Load counter 5 directly
loadValue6	DINT	0	Load counter 6 directly
loadValue7	DINT	0	Load counter 7 directly
prepValue0	DINT	0	Load counter 0 in preparation
prepValue1	DINT	0	Load counter 1 in preparation
prepValue2	DINT	0	Load counter 2 in preparation
prepValue3	DINT	0	Load counter 3 in preparation
prepValue4	DINT	0	Load counter 4 in preparation
prepValue5	DINT	0	Load counter 5 in preparation
prepValue6	DINT	0	Load counter 6 in preparation
prepValue7	DINT	0	Load counter 7 in preparation
cmpValue0	DINT	0	Load comparison value 0
cmpValue1	DINT	0	Load comparison value 1
cmpValue2	DINT	0	Load comparison value 2
cmpValue3	DINT	0	Load comparison value 3
cmpValue4	DINT	0	Load comparison value 4
cmpValue5	DINT	0	Load comparison value 5

Struct_FM3502_fmData			
Name	**Type**	**Initial value**	**Comment**
cmpValue6	DINT	0	Load comparison value 6
cmpValue7	DINT	0	Load comparison value 7
actCntrValue0	DINT	0	Actual counter value 0
actMeasValue0	DINT	0	Measured value result 0
actCntrValue1	DINT	0	Actual counter value 1
actMeasValue1	DINT	0	Measured value result 1
actCntrValue2	DINT	0	Actual counter value 2
actMeasValue2	DINT	0	Measured value result 2
actCntrValue3	DINT	0	Actual counter value 3
actMeasValue3	DINT	0	Measured value result 3
actCntrValue4	DINT	0	Actual counter value 4
actMeasValue4	DINT	0	Measured value result 4
actCntrValue5	DINT	0	Actual counter value 5
actMeasValue5	DINT	0	Measured value result 5
actCntrValue6	DINT	0	Actual counter value 6
actMeasValue6	DINT	0	Measured value result 6
actCntrValue7	DINT	0	Actual counter value 7
actMeasValue7	DINT	0	Measured value result 7
Diagnostic data (data structure)			
diagnostic	**Struct_FM3502_diagInfo**		Elements for diagnostic data
xxxReserved20..23 [1)]	BYTE	16#00	Reserved
chType	BYTE	16#00	Channel type
chInfoLength	BYTE	16#00	Length of channel info
numOfChannel	BYTE	16#00	Number of channels
chFault	BYTE	16#00	Channel fault vector
cntr0Fault	BYTE	16#00	Counter 0 fault
cntr1Fault	BYTE	16#00	Counter 1 fault
cntr2Fault	BYTE	16#00	Counter 2 fault
cntr3Fault	BYTE	16#00	Counter 3 fault
cntr4Fault	BYTE	16#00	Counter 4 fault
cntr5Fault	BYTE	16#00	Counter 5 fault
cntr6Fault	BYTE	16#00	Counter 6 fault
cntr7Fault	BYTE	16#00	Counter 7 fault

[1)] Variable for internal FB use (not relevant to users)

8.6.4.7 Calling function blocks

In order to be able to work with the function blocks in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for FB **_FM3502_control**).
2. Set up variables for the data structure.
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the FB are accessed with <instance name of FB>.<name of output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variables created in step 3.

Note

The call example is an extract from the supplied E_FM3502 application example, which is contained on the "SIMOTION Utilities & Applications" CD-ROM.

If you wish to control more than one FM 350-2, you must create a new variable for the data structure and FB instances with a new name for each FM 350-2 used.

Call example

```
UNIT E_FM3502;

INTERFACE
VAR_GLOBAL

    myDataFM3502        : Struct_FM3502_fmData;   // variable for data structure                (2)

    // OUTPUT VARIABLES
    myStateFMStartup    : BOOL;                   // Start-up status
    myInstFM3502Ctrl    : _FM3502_control;        // create FB instance                         (1)

END_VAR

END_INTERFACE

IMPLEMENTATION

PROGRAM ExampleFM3502                           // Program in BackgroundTask

// Variables used: see interface area under VAR_GLOBAL
VAR
```

```
    FMOutputArray      : ARRAY [0..15] of BYTE;    // Array for FM output data                (3)
END_VAR

// CALL INSTANCE of FB _FB_FM3502_control                                                      (4)
    myInstFM3502Ctrl
        (
        periIn         := myPeripheralInputFM3502,   // I/O variable of I/O inputs           (5)
        periOut        := FMOutputArray,             // FM output data array
        data           := myDataFM3502               // Data structure
        );

// TRANSFER DATA TO FM
    myPeripheralOutputFM3502   := FMOutputArray;     // Assign array of FM output           (7)
                                                     // data to I/O variable

    myStateFMStartup  := myInstFM3502Ctrl.startup;   // Start-up status                     (6)

END_PROGRAM                  // ExampleFM3502

END_IMPLEMENTATION
```

Note

The PROGRAM ExampleFM3502 must be assigned in the execution system.

8.6.4.8 Application example for FM 350-2

Introduction

In this example, the FM 350-2 counter module is used to solve two different tasks. Counter channels 0 and 1 will be used to control a filling unit. At the same time, a frequency measurement with limit value check will be performed using counter channel 4.

Filling unit

A box will be filled with a certain number of parts from a collection bin. Counter channel 0 counts 10 (0...9) parts and controls the filling valve. Counter channel 1 is used to control the motor that transports the boxes and to count the number of boxes. When the box is in the correct position, the valve is opened and the parts are filled into the box. When the specified number has been reached, the valve is closed and the box transport is initiated. Arriving parts are counted until the next box arrives.

While the box is being transported, a new number of parts can be specified.
The number of filled parts and the number of boxes can be monitored (in the symbol browser).

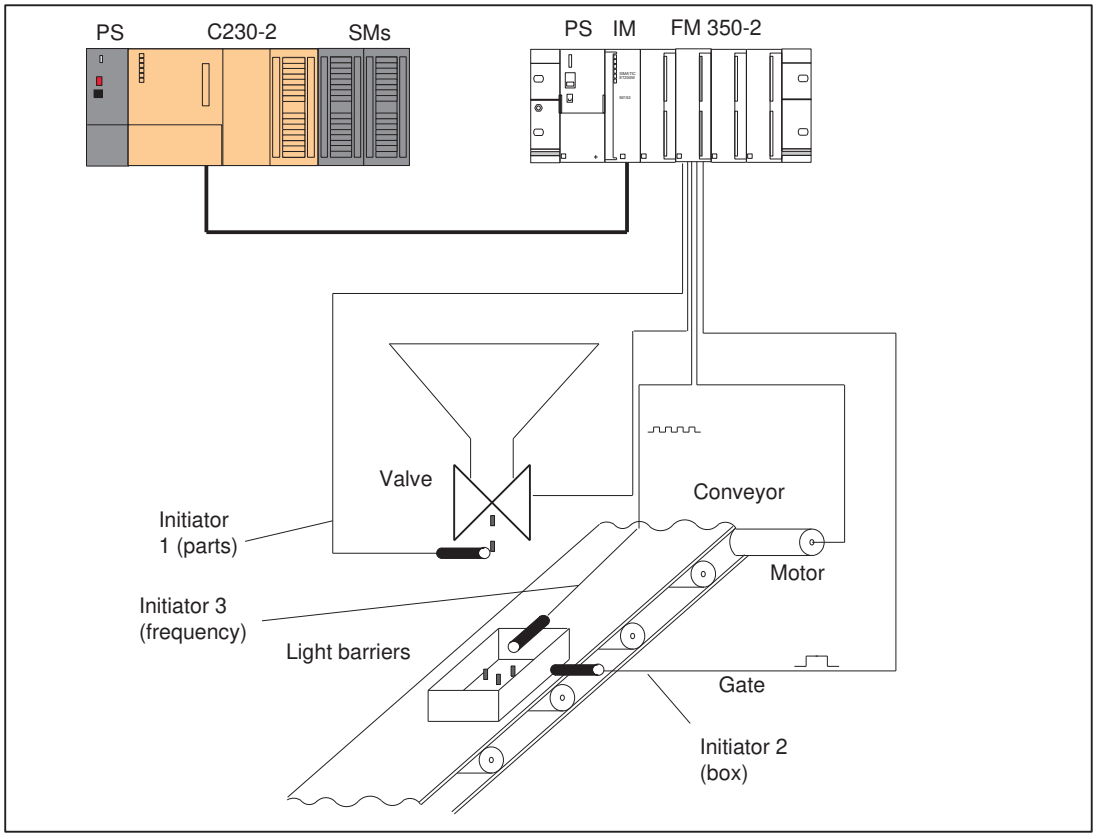


Figure 8-32 Example application for the FM 350-2

Frequency measurement

A frequency measurement (speed of falling parts) is made at counter channel 4 for frequencies up to 10 kHz. The measured frequency is subjected to a limit value check for the lower limit of 1 kHz and upper limit of 9 kHz. The status of the limit values, the measured frequency and the continuously counted pulses can be monitored (in the symbol browser).

FM 350-2 installation and wiring

Proceed as follows to install and wire the FM 350-2:

1. Insert the bus connector supplied with the FM 350-2 onto the bus plug.
2. Hook the FM 350-2 onto the rail, pivot the module downwards and bolt it into place (more detailed instructions can be found in SIMATIC manual *FM 350-2 Counter Module Installation and Parameter Assignment*).

3. Wire the front connector as shown below (a complete pin assignment for the front connector can be found in SIMATIC manual *FM 350-2 Installation and Parameter Assignment*).

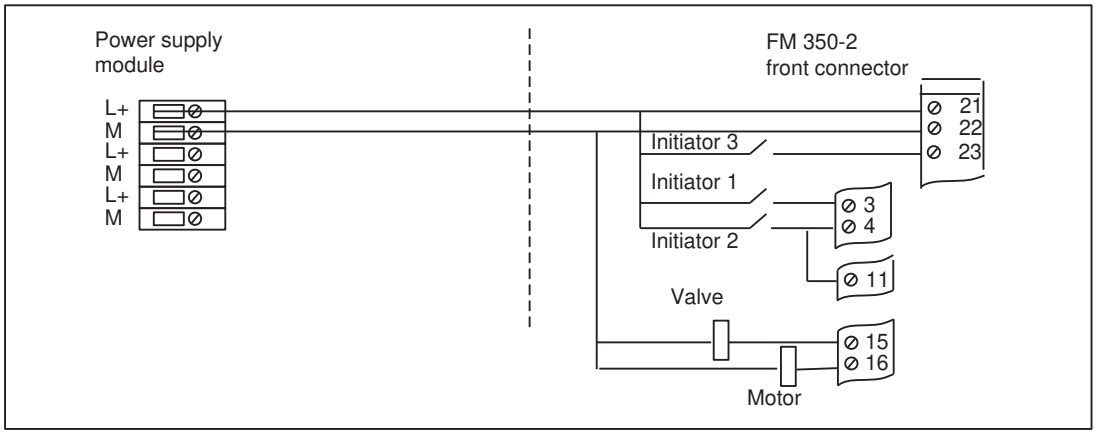


Figure 8-33 FM 350-2 installation and wiring

Terminal	Name	Meaning
21	L+	24 V power supply
22	M	Ground
23	A4	Frequency input from 24 V initiator 3
3	A0	Counter pulses for parts from 24 V initiator 1
4	A1	Counter pulses for boxes from 24 V initiator 2
11	I0	Box in position (HW gate) from terminal 4
15	Q0	Actuation of valve for filling parts
16	Q1	Actuation of the motor for box transport

4. Plug the front connector into the FM 350-2 and screw tightly.

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station.
4. Double-click **FM350-2** to open the **Properties** dialog box for the module. There you will see the "General", "Addresses" and "Basic parameters" tabs for the module.
5. Click the Parameters button. The parameter assignment screen forms of the FM 350-2 will open. The parameters for encoders, operating modes, alarm enabling, and outputs are stored for every channel in these screen forms.
   Under the **Edit > Specify channels** menu command, you will find the global settings for all of the channels of the FM 350-2.

6. Change the following parameters:
   Specify channels:
   - Specify channels 0...7 as single counters!
   - For USER_TYP2, select data type DWORD and for channel 4, select the "Measured value" setting instead of "Counter value".

   Afterwards, click **OK** to confirm.
   Channel 0:
   - Mode: "Single counting" and activation of "Use hardware gate"
   - Under alarm enable, enter the comparison value "9" (10 parts) (all alarm enables alarms are deactivated).

   Channel 4:
   - Mode: "Frequency measurement" and x10 ms for time window "1"
   - Under alarm enable, enter "1000000" for the range underflow limit and "9000000" for the range overflow limit (all alarm enables are deactivated).
7. Transfer the parameter assignment for the FM 350-2 to the hardware configuration using the **File > Save** menu command and close the "FM350-2 Counter" window using the **File > Exit** menu command.
8. Save the hardware configuration with the **Station > Save and compile** menu command.
9. Download the hardware configuration with the **Target system > Download to module** menu command.
   The red "SF" LED of the FM 350-2 turns on and off after the module parameter assignment is downloaded without errors.

Hardware platform

The application example is available for various SIMOTION hardware platforms and is intended for distributed use of the FM 350-2.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the hardware configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and module address in the program (myDataFM3502.moduleAddress).
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Input/output symbols

Table 8-72 Input symbols used

Symbol	Data type	Description
myStartFillup	BOOL	Start the filling unit
myMeasurementFrequency	BOOL	Start the frequency measurement
mySetNewCounterValue	BOOL	Start to load new quantity
myReadActualCounterValue	BOOL	Start to read current values
myChangeChannelActualValue	BOOL	Read selection of current values FALSE: Channels 0..3 TRUE: Channels 4..7
myNewQuantity	UINT	New quantity
myResetProcessAlarm	BOOL	Acknowledgement of process alarm
myResetDiagnosticAlarm	BOOL	Acknowledgement of diagnostic alarm

Table 8-73 Output symbols used

Symbol	Data type	Description
myStateLoad	BOOL	New quantity loaded
myErrorWrite	BOOL	Error when loading quantity
myErrorRead	BOOL	Error when reading current values
myCounterOverflow	BOOL	Upper frequency limit exceeded
myCounterUnderflow	BOOL	Lower frequency limit fallen below
myStateFMStartup	UINT	Startup status
myProcessAlarm	BOOL	Process alarm
myDiagnosticAlarm	BOOL	Diagnostic alarm

Note

You can either monitor and modify the input and output variables used in the programming example in the INTERFACE section of the unit (under VAR_GLOBAL) using the symbol browser, or you can assign real inputs and outputs to the input and output variables in your unit.

Filling unit application sequence

The sequence for the "Filling unit" application is reproduced below.

1. Start the filling unit application by setting the "myStartFillup" input.
   Output Q1 of the FM 350-2 is set to move the box into position.
2. Actuate 24 V initiator 2 (box in position/box counting pulses) when the box is in position.
   A "1" is displayed in data structure "myDataFM3502.checkback.opValue1" (number of boxes).
   Then the valve is opened via output Q0 of the FM 350-2, and the parts are counted. When you actuate 24 V initiator 1, the number of filled parts is incremented in "myDataFM3502.checkback.opValue0" (number of parts). When 10 parts are reached, the valve is closed and the box transport is activated. The process is repeated when the next box arrives.
   Proceed as follows to change the number of parts:
3. Enter the new quantity in the "myNewQuantity" input parameter as a control value and activate it. The new quantity is applied with "Immediate control".
4. Set the "mySetNewCounterValue" input to load the new quantity. The "mySetNewCounterValue" input evaluates the rising edge and only then starts a new write job (transfer of new parts is resumed only if the "mySetNewCounterValue" input was set to "FALSE" beforehand). If the new quantity is successfully loaded, the "myStateLoad" output is briefly set.

Frequency measurement application sequence

The sequence for the "Frequency measurement" application is reproduced below.

1. Start the frequency measurement application by setting the "myMeasurementFrequency" input.
2. Actuate 24 V initiator 3 (frequency input), for example, by connecting a frequency generator to it.
3. Set the "myChangeChannelActualValue" input and the "myReadActualCounterValue" input.
   As long as these inputs are set, the current values are displayed in data structures "myDataFM3502".counterValue4 to "myDataFM3502".measuringValue7.
   You have the option of reading the current values of counter channels 0 to 3 by deleting the "myChangeChannelActualValue" input.
   If the value drops below the lower frequency limit of 1 kHz, this is indicated at the"myCounterUnderflow" output.
   If the value exceeds the upper frequency limit of 9 kHz, this is indicated at the "myCounterOverflow" output.
   In addition, you can also read the current values (counter values and measured values) of counter channels 4 to 7.

Diagnostic alarm / process alarm

Incorrect wiring can cause errors, which the FM 350-2 displays using the "SF" group error LED and in the "myProcessAlarm" and "myDiagnosticAlarm" variables. In these cases, the FM 350-2 triggers a diagnostic alarm/process alarm, provided the basic parameters are set accordingly (alarm generation: YES; and alarm selection: diagnostic or diagnostic+process). If the "PeripheralFaultFM3502" program is integrated in the PeripheralFaultTask, this task is started and the most important task start information is stored temporarily in the "myAlarmDetails", "myLogBaseAddressIn" and "myAlarmInterrupt" variables.

If a diagnostic alarm has been signaled, the **_FM3502_diagnostic** FB is started, which reads out detailed diagnostic information from the module. These diagnostic data are then located in data structure "myDataFM3502". You can acknowledge the respective alarm by setting the "myResetDiagnosticAlarm" input variable (when acknowledging a diagnostic alarm) or "myResetProcessAlarm" input variable (when acknowledging a process alarm).

8.6.5 Function Blocks of the FM 352

8.6.5.1 Overview of the FM 352 function blocks

This section describes the function blocks (FBs) and the data structures required for parameter assignment, control and commissioning of the FM 352 module.

The function blocks form the software interface between the SIMOTION device and the FMs. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available:

- Function block _FM352_initialize (Page 6284)
- Function block _FM352_control (Page 6285)
- Function block _FM352_diagnostic (Page 6287)

SIMOTION SCOUT contains all of the required FBs and data structures **Struct_FM352_ctrlData**, **Struct_FM352_diagData** and **Struct_FM352_paraData** of the FM 352. The function blocks can be used to control one or more FM 352 modules.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC identifiers FM 352".

Note

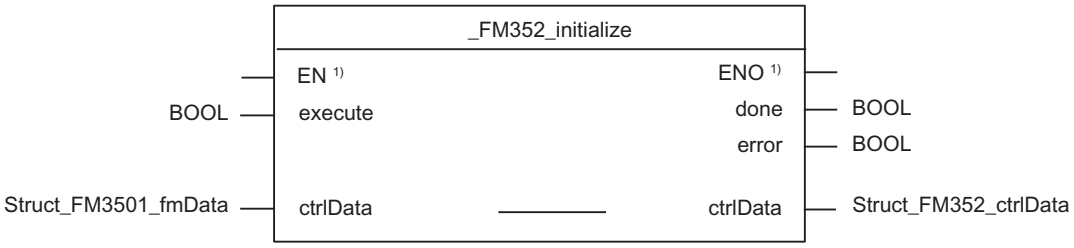
The online functions of the parameter assignment tool in STEP 7 **HW Config** can only be used for diagnostic purposes (read-only access to the module). Write access (control function) has no effect. The parameters set by the program can be read out using the parameter assignment tool.

8.6.5.2 Function block _FM352_initialize

Introduction

The **_FM352_initialize** function block allows you to initialize the channel data after startup of the FM 352 module.

Call (LAD representation)



Parameter description

Table 8-74 Parameters of the _FM352_initialize function block

Name	P type 1)	Data type	Meaning
execute	IN	BOOL	Activation
ctrlData	IN/OUT	Struct_FM352_ctrlData	Data structure with channel-specific data
done	OUT	BOOL	Function or job executed completely without errors
error	OUT	BOOL	Request completed with errors

1) Parameter types: IN/OUT = in/out parameter

Functional description

The _FM352_initialize function block initializes the channel data structure:

- Control signals
- Checkback signals
- Trigger bits, done bits and error bits of jobs
- Function switches and their done bits and error bits
- Job management and internal buffers for the **_FM352_control** function block

The data required for **_FM352_initialize** function block are transferred in variables of the data structure of type **Struct_FM352_ctrlData**.

Task integration (call)

The **_FB_FM352_initialize** function block must be run through after startup of the SIMOTION system. Therefore, you should call it in the **StartupTask** or in a self-programmed initialization phase of your user program. In this way you can ensure that your user program does not access outdated data.

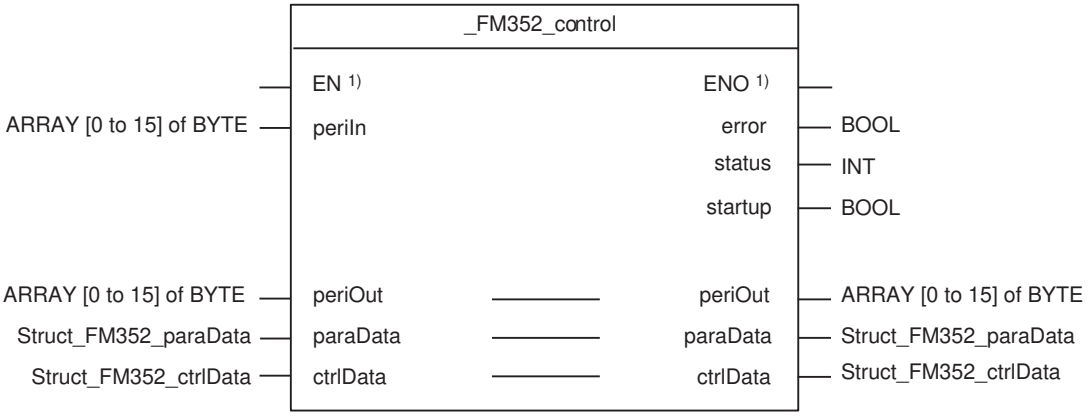
8.6.5.3 Function block _FM352_control

Introduction

The **_FM352_control** function block enables you to do the following:

- Write control signals and read out checkback signals from the FM 352
- Read and write parameters of the FM 352

Call (LAD representation)



1) LAD-specific parameter

Parameter description

Table 8-75 Parameters of the _FM352_control function block

Name	P type 1)	Data type	Meaning
periIn	IN	ARRAY [0 to 15] of BYTE	Transfers I/O variable of the I/O inputs of the FM to the FB
periOut	IN/OUT	ARRAY [0 to 15] of BYTE	Prepared data of the FB for the I/O outputs of the FM 2)
ctrlData	IN/OUT	Struct_FM352_ctrlData	Data structure with channel-specific data
paraData	IN/OUT	Struct_FM352_paraData	Data structure with machine data and output cam data
error	OUT	BOOL	Job completed with errors

Name	P type [1]	Data type	Meaning
status	OUT	INT	Status 0: FB inactive 1: FB active -1: Error
startup	OUT	BOOL	Indicates the startup of the FM

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be supplied with an array of type **ARRAY [0 to 15] of BYTE**. Create a local or global array in your program under **VAR** (do not create a temporary array under VAR_TEMP). After the FB has been called, this array must be assigned to the I/O variable for the I/O outputs of the module. Call example for FM 352!

Functional description

The **_FM352_control** function block performs the following activities in the **BackgroundTask**:

- Reading checkback signals
  The FB reads all of the checkback signals of the FM 352 and enters them into the channel data structure. Because the control signals and jobs are not executed until after this step, the checkback signals reflect the status of the module before the block was called.
- Writing control signals
  The control signals entered in the channel data structure are transferred to the module. Enabling of output cam processing, however, is delayed as long as the trigger for a "Set reference point" job or "Write output cam data" job is set.
- Executing read/write jobs
  The jobs to be executed are specified in channel data structure **Struct_FM352_ctrlData** using "trigger bits". More than one job can be activated at the same time. The jobs are executed by the **_FM352_control** function block in the order received.

Task integration (call)

The **_FM352_control** function block must be called cyclically in the **BackgroundTask** or in the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Read and write jobs are triggered by setting the corresponding trigger bits in the data structure. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

To ensure the proper sequence, the module address must be entered (under general data / version switches) in the **moduleAddress** element of the data structure of type **Struct_FM352_ctrlData**.

Startup behavior

The **_FM352_control** function block acknowledges the module startup. During this time, **status** and **jobBusy** = 1. During the startup phase, job execution is disabled. Any pending jobs are not lost and they will be executed after startup has been acknowledged.

Error message during a call

You can read out error information in the data structure in the **jobErrorId** element of the data structure of type **Struct_FM352_ctrlData**.

- The programming device or PC enables you to read out the diagnostic buffer via the parameter assignment interface using the **Test > Error evaluation** menu command.
  - You will find the error class and the error number along with plain text.
- You can evaluate errors in your program. The following means are available for this purpose:
  - Return values (**status**) of the integrated FBs as a group display for errors that occurred during execution of the FBs.
  - Error bits of the jobs as a group display for errors that occurred during execution of a job.
  - **dataError** error bit as a group display for an error that has been detected by the FM 352 during a write job.
  - Error ID in **jobErrorId** for the cause of the error during communication between FB and FM 352. The return values of system functions **_writeRecord** and **_readRecord** are entered in **jobErrorId**.
    Further information on the system functions is contained in the *SIMOTION System Function/Variables* List Manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!
  - **_FM352_diagnostic** function block for reading out the diagnostic buffer of the FM 352. Here you can obtain the causes for errors, for jobs and asynchronous events (operating errors, diagnostic errors).

Note

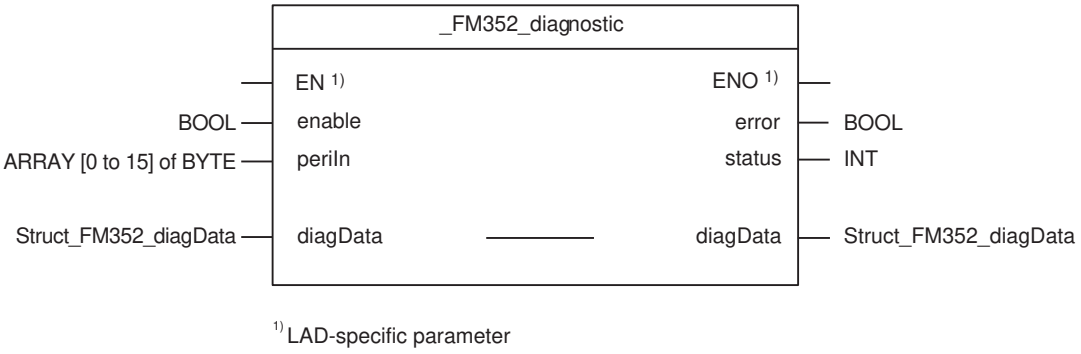
The return value (error ID) in the **status** parameter is present for one cycle only. The values 0x7001 and 0x7002 indicate that data transfer has been initiated and is active.

8.6.5.4 Function block _FM352_diagnostic

Introduction

The **_FM352_diagnostic** function block enables you to read out complete diagnostic data from the FM 352.

Call (LAD representation)



1) LAD-specific parameter

Parameter description

Table 8-76 Parameters of _FM352_diagnostic function block

Name	P type 1)	Data type	Comment
enable	IN	BOOL	Enable
periIn	IN	ARRAY [0..15] of BYTE	I/O variable for access to I/O inputs from the FM 352
diagData	IN/OUT	Struct_FM352_diagData	Data structure for diagnostic data
error	OUT	BOOL	Request completed with errors
status	OUT	INT	Return value 0 : FB inactive 1: FB reading data -1 : Error

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

Functional description

The diagnostic data are read out by the **_FM352_diagnostic** function block and made available to you in the associated **Struct_FM352_diagData** data structure. The return value (error ID) can be read out at the **status** output parameter of the function block.

The **_FM352_diagnostic** function block reads the diagnostic data when checkback signal **diagDataChanged** = **TRUE** either automatically or per job (**diagInformation** = **TRUE**).

Sequence

The data are transferred as follows:

1. If the set trigger parameter **diagInformation** = TRUE or **diagDataChanged** = TRUE in the data structure of type **Struct_FM352_diagData**, the diagnostic data are read out from the FM 352.
2. The data are entered in the data structure of the **_FM352_diagnostic** function block.

3. The return value (error ID) is provided at the **status** output parameter of the **_FM352_diagnostic** function block.

Note

The return value (error ID) in the **status** parameter is present for one cycle only. The values 0x7001 and 0x7002 indicate that a data transfer has been initiated and is active.

4. As soon as the function has been executed, the trigger parameter is signaled as finished by the transfer.

Note

To ensure the proper sequence, the module address must be entered in the **moduleAddress** element of the data structure of type **Struct_FM352_diagData**.

Task integration (call)

The **_FM352_diagnostic** function block must be called in the **BackgroundTask** or **TimerInterruptTask**. An additional call in the **SystemInterruptTask** is not permitted. At least two calls (cycles) are required for complete execution of the function. For performance reasons, the function block should only be called in the **PeripheralFaultTask**.

Job

You can read the diagnostic buffer independent of a new entry by setting the **diagInformation** trigger bit. After the diagnostic buffer is read, the trigger bit is set to **FALSE**.

Error message during a call

If an error occurs during a call, it is reported in the **status** output parameter (= -1). The return value of the **_readRecord** system function is entered in the **jobErrorId** element of the data structure of type **Struct_FM352_diagData**.

Further information on the system function is contained in the *SIMOTION System Function/ Variables* parameter manual. This documentation is included in the SIMOTION SCOUT scope of delivery as electronic documentation!

Note

The error ID in the jobErrorId element and the **status** output parameter is present for one cycle.

8.6.5.5 Data structures of the FM 352

Overview of the FM 352 data structures

The data structures contain all data of the FM 352 relevant for operation, as well as diagnostic data. Three different data structures are provided as type declarations for the FM 352.

- Data structure of type Struct_FM352_ctrlData (Page 6290)
- Data structure of type Struct_FM352_diagData (Page 6298)
- Data structure of type Struct_FM352_paraData (Page 6296)

These data structures are used by the following function blocks: **_FM352_initialize**, **_FM352_control** and **_FM352_diagnostic**. The elements of the data structure are accessed using a variable of the appropriate data type, which is to be defined by the user.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC names of the "Data structure of Struct_FM352_ctrlData", "Data structure of Struct_FM352_paraData" and "Data structure of Struct_FM352_diagData" tables can be found in the appendix SIMOTION and SIMATIC names (Page 6310) in the table "SIMOTION and SIMATIC names for FM 352".

Struct_FM352_ctrlData

The **Struct_FM352_ctrlData** data structure is shown in the table below.

Table 8-77 Data structure of Struct_FM352_ctrlData

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
General data / version switches			
moduleAddress	INT	256	Module address
FmType	BOOL	TRUE	FM 352 V5.0 and higher
Control signals			
enableSimPositive	BOOL	FALSE	Simulation in positive direction
enableSimNegative	BOOL	FALSE	Simulation in negative direction
enableOutputCam	BOOL	FALSE	Output cam processing enabled
enableTrack0Counter	BOOL	FALSE	Counter function of counter output cam track 0 enabled
enableTrack1Counter	BOOL	FALSE	Counter function of counter output cam track 1 enabled
enableTrack	WORD	16#0000	Enable output cam tracks 0 to 15 Bit 0: Track 0
Checkback signals			

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
diagDataChanged	BOOL	FALSE	New entry in the diagnostic buffer of the FM 352 (can be read out with **_FM352_diagnostic**)
dataError	BOOL	FALSE	Data error (can be read out using the parameterization tool)
parameterized	BOOL	FALSE	Module parameterized
outputCamActive	BOOL	FALSE	Output cam processing running
synchronized	BOOL	FALSE	Axis is synchronized
measDone	BOOL	FALSE	Length measurement or edge detection is finished
dirNegative	BOOL	FALSE	Axis moving in a negative direction
dirPositive	BOOL	FALSE	Axis moving in a positive direction
hystZone	BOOL	FALSE	Axis is within the hysteresis range
floatActValue	BOOL	FALSE	Set actual value on-the-fly executed
actPosition	DINT	0	Current position of axis (cyclic updating)
trackSignals	DWORD	16#0000 0000	Current track signals of tracks 0 to 31 Bit 0: Track 0
Function switches			
enableEdgeDetection	BOOL	FALSE	Edge detection ON
enableSimulation	BOOL	FALSE	Simulation ON
enableLenMeasuring	BOOL	FALSE	Length measuring ON
execRetrigRefPoint	BOOL	FALSE	Retrigger reference point
switchOffSwLimit	BOOL	FALSE	Software limit switch disabled
Trigger bits for write jobs			
execWrMachineData	BOOL	FALSE	Write machine data
execWrActivateMData	BOOL	FALSE	Activate machine data
execWrActValueRevoke	BOOL	FALSE	Set actual value, undo on-the-fly actual value setting
execWrOutputCamData1	BOOL	FALSE	Write output cam data 1 (output cams 1 to 15)
execWrOutputCamData2	BOOL	FALSE	Write output cam data 2 (output cams 16 to 31)
execWrOutputCamData3	BOOL	FALSE	Write output cam data 3 (output cams 32 to 47)
execWrOutputCamData4	BOOL	FALSE	Write output cam data 4 (output cams 48 to 63)
execWrOutputCamData5	BOOL	FALSE	Write output cam data 5 (output cams 64 to 79)
execWrOutputCamData6	BOOL	FALSE	Write output cam data 6 (output cams 80 to 95)
execWrOutputCamData7	BOOL	FALSE	Write output cam data 7 (output cams 96 to 111)
execWrOutputCamData8	BOOL	FALSE	Write output cam data 8 (output cams 112 to 127)
execWrSetRefPoint	BOOL	FALSE	Set reference point coordinates
execWrActValue	BOOL	FALSE	Set actual value
execWrActValSetOnTheFly	BOOL	FALSE	Set actual value on-the-fly
execWrZeroOffset	BOOL	FALSE	Set zero point offset
execWrOutputCamEdge1	BOOL	FALSE	Write output cam edge setting (1 cam)
execWrOutputCamEdge16	BOOL	FALSE	Write settings for fast output cam change (16 output cams)

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
Trigger bits for read jobs			
execRdMachineData	BOOL	FALSE	Read machine data
execRdOutputCamData1	BOOL	FALSE	Read output cam data 1
execRdOutputCamData2	BOOL	FALSE	Read output cam data 2
execRdOutputCamData3	BOOL	FALSE	Read output cam data 3
execRdOutputCamData4	BOOL	FALSE	Read output cam data 4
execRdOutputCamData5	BOOL	FALSE	Read output cam data 5
execRdOutputCamData6	BOOL	FALSE	Read output cam data 6
execRdOutputCamData7	BOOL	FALSE	Read output cam data 7
execRdOutputCamData8	BOOL	FALSE	Read output cam data 8
execRdMeasValue	BOOL	FALSE	Read measured values
execRdCntrValueTrack	BOOL	FALSE	Read the count values of the counter cam tracks
execRdActPosition	BOOL	FALSE	Read position data and track data
execRdEncValue	BOOL	FALSE	Read encoder values
execRdOutputCamData	BOOL	FALSE	Read output cam data and track data
Done bits for function switches			
enableEdgeDetectDone	BOOL	FALSE	"Switch edge detection ON or OFF" completed
enableSimDone	BOOL	FALSE	"Switch simulation ON or OFF" completed
enableLenMeasDone	BOOL	FALSE	"Switch length measurement ON or OFF" completed
retrigRefPointDone	BOOL	FALSE	"Switch retrigger reference point ON or OFF" completed
switchOffSwLimDone	BOOL	FALSE	"Switch software limit switch ON or OFF" completed
Done bits for write jobs			
wrMdDone	BOOL	FALSE	"Write machine data" job completed
wrActivateMdDone	BOOL	FALSE	"Activate machine data" job completed
wrActValueRevokeDone	BOOL	FALSE	"Set actual value, undo on-the-fly actual value setting" completed
wrOutputCamData1Done	BOOL	FALSE	"Write output cam data 1" job completed
wrOutputCamData2Done	BOOL	FALSE	"Write output cam data 2" job completed
wrOutputCamData3Done	BOOL	FALSE	"Write output cam data 3" job completed
wrOutputCamData4Done	BOOL	FALSE	"Write output cam data 4" job completed
wrOutputCamData5Done	BOOL	FALSE	"Write output cam data 5" job completed
wrOutputCamData6Done	BOOL	FALSE	"Write output cam data 6" job completed
wrOutputCamData7Done	BOOL	FALSE	"Write output cam data 7" job completed
wrOutputCamData8Done	BOOL	FALSE	"Write output cam data 8" job completed
wrRefPointDone	BOOL	FALSE	"Set reference point coordinates" job completed
wrActValueDone	BOOL	FALSE	"Set actual value" job completed
wrActValSetOnTheFlyDone	BOOL	FALSE	"Set actual value on-the-fly" job completed
wrZeroOffsetDone	BOOL	FALSE	"Set zero point offset" job completed

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
wrOutputCamEdge1Done	BOOL	FALSE	"Change 1 output cam" job completed
wrOutputCamEdge16Done	BOOL	FALSE	"Change 16 output cams" job completed
Done bits for read jobs			
rdMdDone	BOOL	FALSE	"Read machine data" job completed
rdOutputCamData1Done	BOOL	FALSE	"Read output cam data 1" job completed
rdOutputCamData2Done	BOOL	FALSE	"Read output cam data 2" job completed
rdOutputCamData3Done	BOOL	FALSE	"Read output cam data 3" job completed
rdOutputCamData4Done	BOOL	FALSE	"Read output cam data 4" job completed
rdOutputCamData5Done	BOOL	FALSE	"Read output cam data 5" job completed
rdOutputCamData6Done	BOOL	FALSE	"Read output cam data 6" job completed
rdOutputCamData7Done	BOOL	FALSE	"Read output cam data 7" job completed
rdOutputCamData8Done	BOOL	FALSE	"Read output cam data 8" job completed
rdMeasValueDone	BOOL	FALSE	"Read measured values" job completed
rdCntrValueTrackDone	BOOL	FALSE	"Read count values of the counter output cam tracks" job completed
rdActPosDone	BOOL	FALSE	"Read position data and track data" job completed
rdEncValueDone	BOOL	FALSE	"Read current encoder values" job completed
rdOutputCamDataDone	BOOL	FALSE	"Read output cam data and track data" job completed
Error bits for function switches			
enableEdgeDetectError	BOOL	FALSE	Error during "Switch edge detection ON or OFF"
enableSimError	BOOL	FALSE	Error during "Switch simulation ON or OFF"
enableLenMeasError	BOOL	FALSE	Error during "Switch length measurement ON or OFF"
retrigRefPointError	BOOL	FALSE	Error during "Switch retrigger reference point ON or OFF"
switchOffSwLimitError	BOOL	FALSE	Error during "Switch software limit switch ON or OFF"
Error bits for write jobs			
wrMdError	BOOL	FALSE	Error during "Write machine data" job
wrActivateMdError	BOOL	FALSE	Error during "Activate machine data" job
wrActValueRevokeError	BOOL	FALSE	Error during "Set actual value, undo on-the-fly actual value setting" job
wrOutputCamData1Error	BOOL	FALSE	Error during "Write output cam data 1" job
wrOutputCamData2Error	BOOL	FALSE	Error during "Write output cam data 2" job
wrOutputCamData3Error	BOOL	FALSE	Error during "Write output cam data 3" job
wrOutputCamData4Error	BOOL	FALSE	Error during "Write output cam data 4" job
wrOutputCamData5Error	BOOL	FALSE	Error during "Write output cam data 5" job
wrOutputCamData6Error	BOOL	FALSE	Error during "Write output cam data 6" job
wrOutputCamData7Error	BOOL	FALSE	Error during "Write output cam data 7" job
wrOutputCamData8Error	BOOL	FALSE	Error during "Write output cam data 8" job
wrSetRefPointError	BOOL	FALSE	Error during "Set reference point coordinates" job

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
wrActValueError	BOOL	FALSE	Error during "Set actual value" job
wrActValSetOnTheFlyError	BOOL	FALSE	Error during "Set actual value on-the-fly" job
wrZeroOffsetError	BOOL	FALSE	Error during "Set zero point offset" job
wrOutputCamEdge1Error	BOOL	FALSE	Error during "Change 1 output cam" job
wrOutputCamEdge16Error	BOOL	FALSE	Error during "Change 16 output cams" job
Error bits for read jobs			
rdMdError	BOOL	FALSE	Error during "Read machine data" job
rdOutputCamData1Error	BOOL	FALSE	Error during "Read output cam data 1" job
rdOutputCamData2Error	BOOL	FALSE	Error during "Read output cam data 2" job
rdOutputCamData3Error	BOOL	FALSE	Error during "Read output cam data 3" job
rdOutputCamData4Error	BOOL	FALSE	Error during "Read output cam data 4" job
rdOutputCamData5Error	BOOL	FALSE	Error during "Read output cam data 5" job
rdOutputCamData6Error	BOOL	FALSE	Error during "Read output cam data 6" job
rdOutputCamData7Error	BOOL	FALSE	Error during "Read output cam data 7" job
rdOutputCamData8Error	BOOL	FALSE	Error during "Read output cam data 8" job
rdMeasValueError	BOOL	FALSE	Error during "Read measured values" job
rdCntrValueTrackError	BOOL	FALSE	Error during "Read count values of the counter cam tracks" job
rdActPosError	BOOL	FALSE	Error during "Read position data and track data" job
rdEncValueError	BOOL	FALSE	Error during "Read encoder values" job
rdOutputCamDataError	BOOL	FALSE	Error during "Read output cam data and track data" job
Job management for _FM352_control function block			
jobErrorId	INT	0	Communication error
jobBusy	BOOL	FALSE	At least one job running
execJobReset	BOOL	FALSE	Reset all errors and error bits
Data for jobs			
Zero point offset			
zeroOffset	DINT	0	Zero point offset
Set actual value			
actValue	DINT	0	Coordinate for "Set actual value"
Set actual value on-the-fly			
actValueSetOnTheFly	DINT	0	Coordinate for "Set actual value on-the-fly"
Set reference point			
refPoint	DINT	0	Coordinate for "Set reference point"
Set output cam edge			
outputCamNumber	INT	0	Output cam numbers for "Change output cam edges"
beginOutputCam	DINT	0	Beginning of output cam for "Change output cam edges"
endOutputCam	DINT	0	End of output cam for "Change output cam edges"

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
Length/edge measurement			
beginLenMeasuring	DINT	0	Starting value for length/edge measurement
endLenMeasuring	DINT	0	End value for "length/edge measurement"
measRange	DINT	0	Length for "length/edge measurement"
Read count values			
cntrValueTrack0	INT	0	Current count value for counter output cam track 0
cntrValueTrack1	INT	0	Current count value for counter output cam track 1
Read position data and track data			
actPosition1	DINT	0	Current position of axis (read with "execRdActPosition" job)
actSpeed	DINT	0	Current velocity
trackState1	DWORD	16#0000 0000	Track ID bits for tracks 0..31
Read encoder data			
encValue	DINT	0	Read encoder values
cntrValueByZero	DINT	0	Count value at the last zero mark
absEncOffset	DINT	0	Absolute encoder adjustment
Read output cam data and track data			
outputCamData00_31	DWORD	16#0000 0000	Output cam data 0 to 31
outputCamData32_63	DWORD	16#0000 0000	Output cam data 32 to 63
outputCamData64_95	DWORD	16#0000 0000	Output cam data 64 to 95
outputCamData96_127	DWORD	16#0000 0000	Output cam data 96 to 127
trackState2	DWORD	16#0000 0000	Track ID bits for tracks 0...31
actPosition2	DINT	0	Current position of axis (read with "execRdOutputCamData" job)
Fast output cam change			
numOfOutputCamsToSet	BYTE	16#00	Volume of project data: 0, 1, 2, 3 = max. 16, 32, 64, 128 cams
disableDataCheck	BOOL	FALSE	Disable data check
Output cam data (data structure)			
outputCam	**ARRAY [0..15] of outputCamType**		Elements for output cam data
number	BYTE	16#00	Output cam number
setForceDirection	BOOL	FALSE	"Change force direction of output cam" job
setBegin	BOOL	FALSE	"Change beginning of output cam" job
setEnd	BOOL	FALSE	"Change end of output cam" job
setActuationTime	BOOL	FALSE	"Change actuation time" job
deactivate	BOOL	FALSE	Switch off the output cam during the output cam change
posForceDirection	BOOL	FALSE	Force direction positive (plus)
negForceDirection	BOOL	FALSE	Force direction negative (minus)
beginOutputCam	DINT	0	Beginning of output cam
endOutputCam	DINT	0	End of output cam / ON duration
actuationTime	UINT	0	Actuation time

Struct_FM352_ctrlData			
Name	**Type**	**Initial value**	**Comment**
Variables for FB-internal use (not relevant to users)			
xxxEnableScom	BOOL	FALSE	Internal use
xxxEnableSfct	BOOL	FALSE	Internal use
xxxScomDone	BOOL	FALSE	Internal use
xxxSfctDone	BOOL	FALSE	Internal use
xxxErrorScom	BOOL	FALSE	Internal use
xxxErrorSfct	BOOL	FALSE	Internal use
xxxmeasJobErrorId	INT	0	Internal use
xxxMeasJobBusy	BOOL	FALSE	Internal use
xxxActOrder	INT	0	Internal use
xxxNextOrder	INT	0	Internal use
xxxDataSetNumber	DINT	0	Internal use
xxxDataSetLength	INT	0	Internal use
xxxDataSetStart	DINT	0	Internal use
xxxEdgeOnOld	BOOL	FALSE	Internal use
xxxSimOnOld	BOOL	FALSE	Internal use
xxxMeasOnOld	BOOL	FALSE	Internal use
xxxReftrOnOld	BOOL	FALSE	Internal use
xxxSswOffOld	BOOL	FALSE	Internal use
xxxRead	BOOL	FALSE	Internal use
xxxError	BOOL	FALSE	Internal use
xxxEnableEdgeDetectOld	BOOL	FALSE	Internal use
xxxEnableSimOld	BOOL	FALSE	Internal use
xxxEnableLenMeasOld	BOOL	FALSE	Internal use
xxxRetrigRefPointOld	BOOL	FALSE	Internal use
xxxSwitchOffSwLimOld	BOOL	FALSE	Internal use
xxxDataSet11	WORD	16#0000	Internal use
xxxDataSet12	WORD	16#0000	Internal use
xxxControl	DWORD	16#00000000	Internal use
xxxFeedback0	DWORD	16#00000000	Internal use

Struct_FM352_paraData

The **Struct_FM352_paraData** data structure is shown in the table below.

Table 8-78 Data structure Struct_FM352_paraData

Struct_FM352_paraData			
Name	**Type**	**Initial value**	**Comment**
Machine data			
xxxModuleType1 [1)]	BOOL	FALSE	0 for FM 352

Struct_FM352_paraData			
Name	**Type**	**Initial value**	**Comment**
enableProcessAlarm	BOOL	FALSE	Enable process alarm: Output cam ON/OFF
xxxModuleType2 [1)]	BOOL	FALSE	0 for FM 352
minEdgeDistance	DINT	0	Minimum edge distance for edge detection
unitDimension	DINT	1	Dimension system
axisType	DINT	0	Linear axis = 0 Rotary axis = 1
endRotAxis	DINT	100000	End of rotary axis
encType	DINT	1	Encoder type, message frame length
lenPerRevolution	DINT	80000	Length per encoder revolution
incPerRevolution	DINT	500	Increments per encoder revolution
cntOfRevolutions	DINT	1024	Number of encoder revolutions
baudRate	DINT	0	Transmission rate
refPoint	DINT	0	Home position coordinates
absEncOffset	DINT	0	Absolute encoder adjustment
refPointTrigMode	DINT	0	Reference point retrigger mode
cntrDirection	BOOL	FALSE	Counting direction: 0 = normal, 1 = inverted
openCircuit	BOOL	TRUE	Wire break monitoring
transmissionError	BOOL	TRUE	Message frame error monitoring
missingPulse	BOOL	TRUE	Missing pulse monitoring
swLimitStart	DINT	-100000000	Start of software limit switch
swLimitEnd	DINT	100000000	End of software limit switch
numOfOutputCamsToSet	DINT	0	Volume of project data: 0,1,2,3 = max. 16,32,64,128 output cams
hysteresis	DINT	0	Hysteresis
simSpeed	DINT	0	Simulation speed
ctrlTrackOutputs	WORD	16#0000	Track output actuation: 0 = electronic cam controller, 1 = SIMOTION device; Bit number = track number
enableInput3	BOOL	FALSE	Enable input I3
xxxEnableInput4..10 [1)]	BOOL	FALSE	Reserved
track0CntrOutputCam	BOOL	FALSE	Track 0 is the counter cam track
track1CntrOutputCam	BOOL	FALSE	Track 1 is the counter cam track
track2CntrOutputCam	BOOL	FALSE	Track 2 is the counter cam track
track0CntrLimit	DINT	2	Upper count value for counter cam track 0
track1CntrLimit	DINT	2	Upper count value for counter cam track 1
Output cam data for output cams 0 to 15 / 0 to 31 / 0 to 63 / 0 to 127 (data structure)			
outputCam	**ARRAY [0..127] of outputCamData**		Elements for output cam data
valid	BOOL	FALSE	Output cam valid
posForceDirection	BOOL	FALSE	Force direction positive (plus)
negForceDirection	BOOL	FALSE	Force direction negative (minus)

Struct_FM352_paraData			
Name	**Type**	**Initial value**	**Comment**
outputCamType	BOOL	FALSE	FALSE: Position-based cams, TRUE: Time-based output cam
switchOnAlarm	BOOL	FALSE	Process alarm while switching on
switchOffAlarm	BOOL	FALSE	Process alarm while switching off
trackNumber	BYTE	16#00	Track number
beginOutputCam	DINT	0	Beginning of output cam
endOutputCam	DINT	0	End of output cam / ON duration
actuationTime	UINT	0	Actuation time

[1)] Variable for FB-internal use (not relevant to users)

Struct_FM352_diagData

The **Struct_FM352_diagData** data structure is shown in the table below. This data structure contains the diagnostic data of the FM 352.

Table 8-79 Data structure Struct_FM352_diagData

Struct_FM352_diagData			
Name	**Type**	**Initial value**	**Comment**
moduleAddress	INT	256	Module address
jobErrorId	INT	0	Description of errors
jobBusy	BOOL	FALSE	At least one job running
diagInformation	BOOL	FALSE	Essential to read the diagnostic buffer
numOfValidEntries	INT	0	Number of valid entries in the list
Diagnostic data (data structure)			
diagnosticEntry	**ARRAY [1..4] of diagType**		Elements for diagnostic data
incomingAlarm	BOOL	FALSE	Event coming
internFault	BOOL	FALSE	Internal fault
extFault	BOOL	FALSE	External fault
faultClass	INT	0	Fault class
faultNumber	INT	0	Error code
chNumber	INT	0	Channel number
outputCamNumber	INT	0	Output cam numbers 0 to 127 with fault class = output cam data error
Variables for FB-internal use (not relevant to users)			
xxxDataSet	ARRAY [0..227] of BYTE		Internal use
xxxDiagnosis0	Struct_FM352_diagType	-	Internal use
xxxFeedback0	DWORD	16#00000000	Internal use
xxxNextOrder	INT	0	Internal use
xxxDataSetNumber	BYTE	16#00	Internal use
xxxDataSetLength	INT	0	Internal use

Struct_FM352_diagData			
Name	**Type**	**Initial value**	**Comment**
xxxCntOfBuffers	INT	0	Internal use
xxxEnableDataSet236	BOOL	FALSE	Internal use
xxxDataSet	BOOL	FALSE	Internal use

8.6.5.6 Calling function blocks

In order to be able to work with the function blocks in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for **FB_FM352_control**).
2. Set up variables for the data structure.
3. Create an array for the in/out parameters of the FB.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the FB are accessed with <instance name of FB>. <name of output parameter>.
7. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variables created in step 3.

Note

The call example is an extract from the supplied E_FM352 application example, which is contained on the "SIMOTION Utilities & Applications" CD-ROM.

If you wish to control more than one FM 352, you must create a new variable for the data structure and FB instances with a new name for every FM 352 used.

Call example

```
UNIT E_FM352;

INTERFACE
VAR_GLOBAL
  myDataFM352Ctrl       : Struct_FM352_ctrlData ;    // variable of data structure              (2)
  myDataFM352Parameter  : Struct_FM352_paraData ;    // variable of data structure
  myInstFM352Ctrl       : FM352_control ;            // create FB instance                      (1)
END_VAR

PROGRAM ExampleFM352;                                // program in background task

END_INTERFACE

IMPLEMENTATION
```

```
PROGRAM ExampleFM352

VAR
    FMOutputArray        : ARRAY [0..15] of BYTE;    // Array for FM output data                (3)
END_VAR

// CALL INSTANCE of FB _FB_FM352_control                                                        (4)

  myInstFM352Ctrl
    (
    periIn      := myPeripheralInputFM352,  // variable of I/O inputs                        (5)
    ctrlData    := myDataFM352Ctrl,         // variable with channel-specific data
    paraData    := myDataFM352Parameter,    // variable with machine and output
                                            // cam data
    periOut     := FMOutputArray            // FM output data array
    );

   // TRANSFER DATA TO FM
  myPeripheralOutputFM352  := FMOutputArray;    // Copy array of FM output                   (7)
                                                // data to I/O variables

  myStateFMStartup   := myInstFM352Ctrl. startup; // Start-up status                         (6)

END_PROGRAM                                       // ExampleFM352

END_IMPLEMENTATION
```

Note

The PROGRAM ExampleFM352 must be assigned in the execution system.

8.6.5.7 Application example for FM 352

Introduction

In this example you use a user program to control an output cam controller.

Once startup of the FM 352 is complete, the example program transfers stored machine data to the FM 352. The FM 352 is now parameterized for the application example. It then executes a series of steps in response to events.

Using the variable tables, you can specify events, monitor the reactions of the module and evaluate the diagnostic buffer.

This example will familiarize you with the following block options:

- Submitting several jobs at once
- Mixing read and write jobs
- Reading using a continuous job without waiting for the end of the job
- Evaluating the checkback signals of the block
- Evaluating the checkback signals for an individual job
- Centralized error evaluation by the **_FM352_diagnostic** function block at the end of the user program
- Evaluating the diagnostic buffer in conjunction with "dataError"

FM 352 installation and wiring

Connect the power supply to the front connector of the FM 352 at terminal 1 "L+" and terminal 2 "M" (24 V).

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station.
4. You will find the FM 352 module under the corresponding ET 200M.
5. Save the hardware configuration with the **Station > Save and compile** menu command.
6. Download the hardware configuration with the **Target system > Download to module** menu command.
   The red "SF" LED on the FM 352 turns on and then off if the assigned module parameters have been downloaded without errors.

Hardware platform

The application example is available for various SIMOTION hardware platforms and is intended for distributed use of the FM 352.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Adapting the application example

The configuration in the example and your available hardware must be adapted.

The following options are available:

1. You can adapt the configuration in the example to the available hardware (PROFIBUS DP address).
2. You can adapt the configuration of the hardware to the example (PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the hardware configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and module address in the program (myDataFM352Ctrl.moduleAddress).
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Default settings for machine and output cam data

The following FM 352 data are preset in the program and transferred to the FM at the start of the example (see Starting the example).

- Measuring system: Degrees (4 decimal places)
- Axis: Rotary axis:
  - End of rotary axis: 360.0000 degrees
  - Simulation speed: 360.0000 degrees/min
- Encoder: Monitoring: Switch off wire break and missing pulse
- Cams:

No.	Valid	Track	Type	Start [degrees]	End [degrees]	Time [ms]	Actuation time [ms]	Effective direction	Process alarm
0	x	0	Distance	0.0000	90.0000	-	0.0	Both	None
1	x	0	Distance	180.0000	270.0000	-	0.0	Both	None
2	x	1	Distance	0.0000	90.0000	-	2000.0	Both	None
3	x	1	Distance	180.0000	270.0000	-	2000.0	Both	None
4	x	2	Distance	130.0000	330.0000	-	0.0	Both	None
5	x	3	Distance	130.0000	330.0000	-	2000.0	Both	None

Application sequence

The SIMOTION device (C230-2, P350, D435) is in "RUN" mode. Create a watch table in your project and insert the values from tables "Current cam tracks and output cams", "Switch", "Current actual value" and "Error displays".

Default settings in the program

The following constants are defined in the **E_FM352** unit:

Table 8-80 Constants

Name	Data type	Control value	Meaning
myRefPoint	DINT	1000	Reference coordinate
myBeginOutputCam0	DINT	1470000	Beginning of output cam 0
myEndOutputCam0	DINT	1920000	End of output cam 0
myBeginOutputCam1	DINT	2000000	Beginning of output cam 1
myEndOutputCam1	DINT	3000000	End of output cam 1

Table 8-81 Current cam tracks and output cams

Name	Data type	Meaning
myDataFM352Ctrl.synchronized	BOOL	= TRUE if the axis is synchronized
myDataFM352Ctrl.outputCamActive	BOOL	Output cam processing running
myDataFM352Ctrl.trackSignalsCamActive	DWORD	Current track signals (tracks 0...31)
myDataFM352Ctrl.outputCamData00_31	DWORD	Output cam identifier bits for output cams 0 to 31
myDataFM352Parameter.outputCam[0].beginOutputCam	DINT	Beginning of output cam 0
myDataFM352Parameter. outputCam[0].endOutputCam	DINT	End of output cam 0
myDataFM352Parameter. outputCam[1].beginOutputCam	DINT	Beginning of output cam 1
myDataFM352Parameter. outputCam[1].endOutputCam	DINT	End of output cam 1

Table 8-82 Switch

Name	Data type	Control value	Meaning
mySwitch	BOOL	TRUE	Input for simulated switch

Table 8-83 Current actual value

Name	Data type	Meaning
myDataFM352Ctrl.actPosition	DINT	Current axis position
myStepNumber	INT	Step number

Table 8-84 Error displays

Name	Data type	Meaning
myError	BOOL	Group errors
myOutputCamError	BOOL	Output cam errors
myDataFM352Ctrl.jobErrorId	INT	Error code

Starting the example

Start the example by setting input **myExampleStart** = **TRUE**. Once the program has detected a rising edge at this input, all of the machine data and output cam data required for the example are transferred to the FM and an axis is then started in simulation. You can observe the changes in the actual position (myDataFM352Ctrl.actPosition), output cam data (myDataFM352Ctrl.outputCamData00_31) and track signals (myDataFM352Ctrl.trackSignals). You can also observe the step number of the step sequence (myStepNumber).

When output cam 4 is set (130 degrees), parameters for output cams 0 and 1 are reset to the default values (see Table "Constants"). You can see the change in the watch table.

The program then waits for an external event. Switch the simulated switch by setting **mySwitch** = **TRUE**. The output cam data revert back to the previous values.

After this pass, the sequence of steps is executed, the step number is -2 and the simulation is stopped.

Error assessment

If an error occurs during execution, the step sequence is stopped and the simulation is switched off. Step number -1 is entered.

User program (sequence steps)

The user program executes a sequence of steps as follows:

- **Step** 99: The program waits in cyclic processing mode for the example to start (StartupTask has been executed but "exampleStart" = FALSE).
- **Step** 0: The output cam controller is initialized. Associated data is set to the jobs that are to be executed when the module is restarted. Module restart can be triggered, for example, by a restart of the C230-2.
- **Step** 1: The program waits for the set jobs to be executed.
- **Step** 2: The program continuously reads the output cam ID bits and waits until output cam 4 is set.
- **Step** 3: Parameters for output cams 0 and 1 are reassigned. To enable you to observe the change, the output cam data are read out before and after the change and displayed in the watch table.
- **Step** 4: The program waits for the set jobs to be executed.
- **Step** 5: Here the program waits for "external" event "Switch ON" (mySwitch = TRUE), which you can set via the watch table.

- **Step** 6: When the event occurs, output cams 0 and 1 are reset to the value that was read out in the initialization step.
- **Step** 7: The program waits for the set jobs to be executed.

At the end of the sequence of steps, the instance of FB **_FM352_control** and the instance of FB **_FM352_diagnostic** are called.

If the diagnostics detected a message about invalid output cam data, the **myOutputCamError** output is set.

8.6.6 Alarm processing

8.6.6.1 Overview of alarm processing

Pending error messages are processed and evaluated differently in a SIMOTION system than in a SIMATIC system. By default, the diagnostic alarms and process alarms are not enabled. Activate the alarms for the relevant module in the hardware configuration (refer to Chapter Inserting function modules into the SIMOTION project (Page 6244)).

If you have parameterized process alarms and/or diagnostic alarms, program the alarm processing sequence in accordance with the flow diagram below.

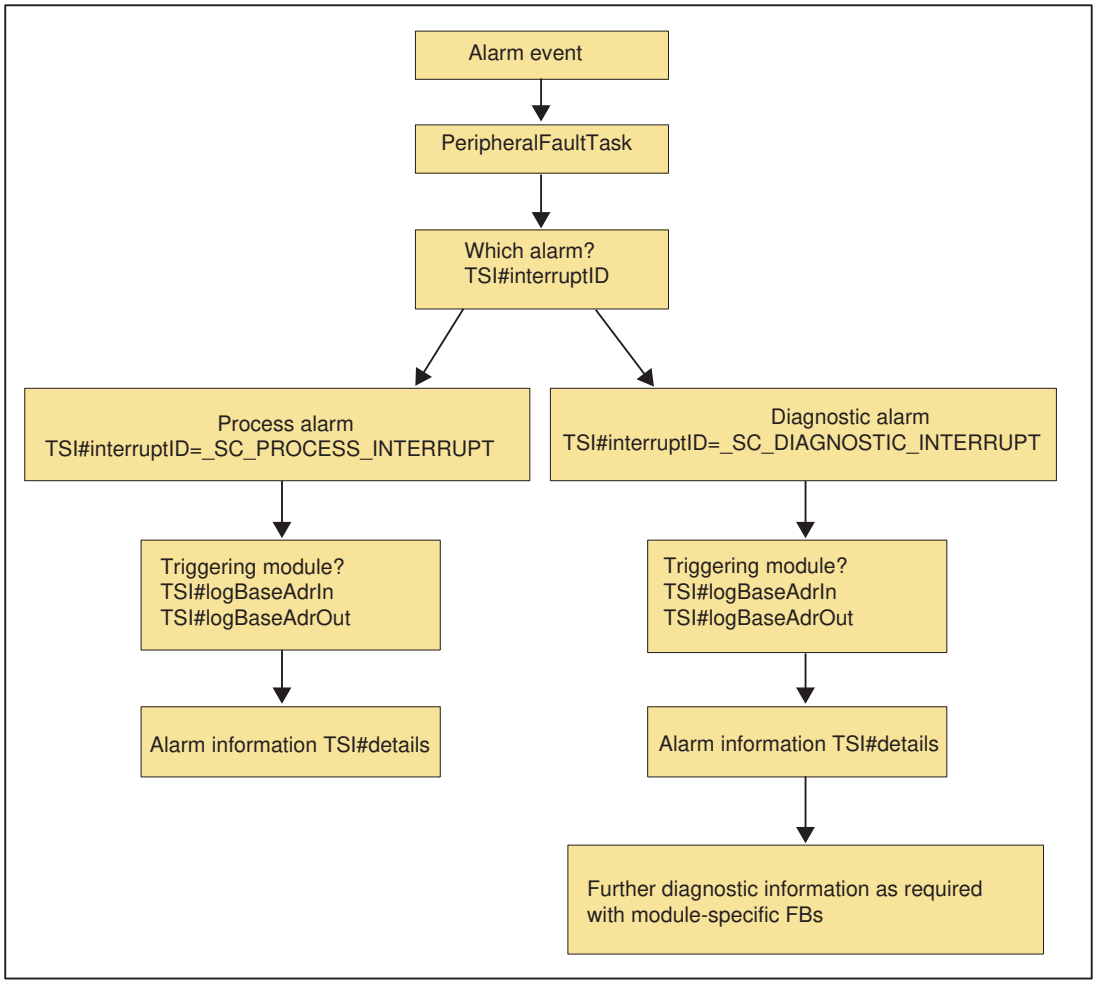


Figure 8-34 Alarm processing in the FM modules

Alarm evaluation

Alarms originating from the I/O are evaluated in the **PeripheralFaultTask**. When the **PeripheralFaultTask** is started, the **Taskstartinfo** is made available, which you can evaluate in the user program.

The **Taskstartinfo** of the **PeripheralFaultTask** is comparable to the local data of OB40 and OB82 in the SIMATIC system.

Table 8-85 Meaning of the Taskstartinfo

Task	TSI		Note
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an input area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an output area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the alarm was caused by a station failure or station recovery of an associated DP slave, otherwise _SC_INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

8.6.6.2 Process alarms

For the FM modules, you can select which events will trigger a process alarm. The assignment can be made in the parameter assignment screen forms of the configuration package for each module.

Definition of a process alarm

If an event needs a response irrespective of the cycle of the SIMOTION device, the relevant FM module can trigger a process alarm.

Events that trigger a process alarm

The criteria (events) that trigger process alarms in a SIMOTION system are the same as those in a SIMATIC system.

For a more detailed description, refer to the
FM 350-1/FM 350-2/FM 352 Installation and Parameter SIMATIC manuals.

Reactions to a process alarm

A process alarm causes the following to occur:

- Process data are written in the **TSI#details** variable in the Taskstartinfo of **PeripheralFaultTask**.
- The SIMOTION device goes into STOP mode if a program has not been assigned in the **PeripheralFaultTask**.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

More information is available in the following SIMATIC manuals:

- *FM 350-1 Function Module Installation and Parameter Assignment*
- *FM 350-2 Counter Function Module Installation and Parameter Assignment*
- *FM 352 Electronic Cam Controller Installation and Parameter Assignment*

Evaluation of process alarms in the FM

If a process alarm is triggered, the data in the Taskstartinfo of **PeripheralFaultTask** are written to the **TSI#details** variable and made available there.

Note

If an event occurs that should trigger a process alarm, but a previous identical event has not yet been acknowledged, a new alarm is not triggered. The new process alarm is lost.

Subject to parameter assignment, this can result in a diagnostic alarm ("Process alarm lost").

If < 2 ms elapse between two events that should normally both trigger a process alarm, the second process alarm is lost and no diagnostic alarm can be triggered (for FM 350-2 only).

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

8.6.6.3 Diagnostic alarms

Definition of a diagnostic alarm

If the user program is to respond to an internal or external error, you can set the parameters for a diagnostic alarm that will interrupt the cyclical program of the SIMOTION device.

Events that trigger a diagnostic alarm

The criteria (events) that trigger diagnostic alarms in a SIMOTION system are the same as in a SIMATIC system.

For a more detailed description, refer to the *FM 350-1/FM 350-2/FM 352, Installation and Parameter Assignment* SIMATIC manuals.

Responses to a diagnostic alarm

If a diagnostic alarm is issued, the following occurs:

- Diagnostic data are written to **TSI#details** variable in the Taskstartinfo of **PeripheralFaultTask**.
- The SIMOTION device goes into STOP mode if a program has not been assigned in the **PeripheralFaultTask**.
- You can use module-specific FBs to read additional diagnostic data in the **PeripheralFaultTask** or **BackgroundTask** according to FM type.
- The group error LED (SF) of the FM module lights up. The group error LED (SF) is extinguished as soon as the error has been remedied.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

More information is available in the following SIMATIC manuals:

- *FM 350-1 Function Module Installation and Parameter Assignment*
- *FM 350-2 Counter Function Module Installation and Parameter Assignment*
- *FM 352 Electronic Cam Controller, Installation and Parameter Assignment*

Evaluation of diagnostic alarms

If a diagnostic alarm is triggered, the data in the Taskstartinfo of the **PeripheralFaultTask** are written to the **TSI#details** variable. The **PeripheralFaultTask** is called automatically and provides 4 bytes of diagnostic information for fast analysis. The contents of the diagnostic information are the same as in a SIMATIC system.

Additional information can be read out using module-specific FBs.

FM 350-1

FB **_FM3501_diagnostic** reads out 16 bytes of diagnostic data and makes them available in the data structure of type **Struct_FM3501_data**. The function block can be called in the **PeripheralFaultTask**, **BackgroundTask** or **TimerInterruptTask**.

FM 350-2

FB **_FM3502_diagnostic** reads out 16 bytes of diagnostic data and enters them in the **diagnostic** substructure of the data structure of type **Struct_FM3502_data**. The function block can be called in the **PeripheralFaultTask**, **BackgroundTask** or **TimerInterruptTask**.

FM 352

FB **_FM352_diagnostic** reads out all diagnostic data and enters them into the data structure of type **Struct_FM352_diagnosticData**. The FB must be called in the **BackgroundTask** or **TimerInterruptTask**.

8.6.7 Appendix

8.6.7.1 SIMOTION and SIMATIC names

The tables below contain a comparison of SIMOTION and SIMATIC names.

Table 8-86 SIMOTION and SIMATIC names for FM 350-1

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
Function block parameters		
_FM3501_control2	**FC CNT_CTRL (FC 0)**	**_FB_FM3501_control**
periIn	-	inputInterface
enableSwGate	SW_GATE	softwareGate
enableStopGate	GATE_STP	stopGate
cntrRange	-	counterRange
execResetOpError	OT_ERR_A	resetOperationError
setOutput0/setOutput1	SET_DO0/SET_DO1	-
data	DB_NO	data
setStartValue	L_DIRECT	setStartValue
setPrepStartValue	L_PREPARE	setPrepareStartValue
setCmpValue1	T_CMP_V1	setComparisonValue1
setCmpValue2	T_CMP_V2	setComparisonValue2
setParaOutput	C_DOPARA	-
resetSyncState	RES_SYNC	resetSyncState
resetCntrState	RES_ZERO	resetCounterState
periOut	-	outputInterface
errorOperation	OT_ERR	operationError
startup	-	startup
_FM3501_diagnostic	**FC DIAG_INF (FC 1)**	**_FB_FM3501_diagnostic**
data	DB_NO	data
execute	IN_DIAG	requestDiagnosticData
done	-	-
status	-	returnValue
Data structure elements		
Struct_FM3501_fmData		**Struct_FM3501_data**
xxxReserved1 [1)]	FP	dedicatedData1

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
xxxReserved2 [1)]	RESERVED	dedicatedData2
moduleAddress	MOD_ADR	moduleAddress
xxxReserved3 [1)]	A_BYTE	dedicatedData3
loadValue1	LOAD_VAL	loadValue_1
configMeasOut0	LOAD_VAL (Byte0, Measuring operating mode)	
configCntrOut0/configCntrOut1	LOAD_VAL (Byte0, Counting operating mode)	
setCntrHyst	LOAD_VAL (Byte1, Counting operating mode)	
setCntrPulse	LOAD_VAL (Byte2, Counting operating mode)	
cmpValue1_1	CMP_V1	comparisonValue1_1
cmpValue2_1	CMP_V2	comparisonValue2_1
loadValue2	-	loadValue_2
cmpValue1_2	-	comparisonValue1_2
cmpValue2_2	-	comparisonValue2_2
xxxReserved4 [1)]	A_BIT0_0	dedicatedData4
xxxReserved5 [1)]	TFB	enableTestMode
xxxReserved6 [1)]	A_BIT0_3	dedicatedData5
xxxReserved7 [1)]	A_BIT0_6	dedicatedData6
enableForwardSetting	ENSET_UP	enableSettingForward
enableReverseSetting	ENSET_DN	enableSettingReverse
xxxReserved8 [1)]	A_BIT1_2	dedicatedData7
xxxReserved9 [1)]	A_BIT1_3	dedicatedData8
enableOutput0	CTRL_DQ0	enableOutput0
enableOutput1	CTRL_DQ1	enableOutput1
xxxReserved10 to 15 [1)]	A_BIT3_0...5	dedicatedData9..14
actValue1	ACT_LOAD	actualValue_1
actCntrValue1	ACT_CNTV	actualCounterValue_1
actValue2	-	actualValue_2
actCntrValue2	-	actualCounterValue_2
errorIdData	DA_ERR_W	dataErrorNumber
errorIdOperation	OT_ERR_B	operationErrorNumber
xxxReserved16 [1)]	E_BIT0_0	dedicatedData15
xxxReserved17 [1)]	STS_TFB	testMode
xxxReserved18 [1)]	E_BIT0_2	dedicatedData16
xxxReserved19 [1)]	E_BIT0_3	dedicatedData17
dataError	DATA_ERR	dataError
xxxReserved20 [1)]	E_BIT0_5	dedicatedData18
xxxReserved21 [1)]	E_BIT0_6	dedicatedData19
parameterized	PARA	parameterized

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
opState	STS_RUN	operationState
opDirection	STS_DIR	operationDirection
zeroCrossing	STS_ZERO	zeroCrossing
overflow	STS_OFLW	overflow
underflow	STS_UFLW	underflow
synchronized	STS_SYNC	synchronized
stateGate	STS_GATE	stateGate
stateSwGate	STS_SW_G	stateSoftwareGate
stateSetInput	STS_SET	stateInputSet
xxxReserved22 [1)]	E_BIT2_1	dedicatedData20
stateOfDiStart	STS_STA	stateInputStart
stateOfDiStop	STS_STP	stateInputStop
stateCompValue1	STS_CMP1	comparisonValue1Reached
stateCompValue2	STS_CMP2	comparisonValue2Reached
remStateCompValue 1	STS_COMP1	-
remStateCompValue 2	STS_COMP2	-
xxxReserved23..28 [1)]	E_BIT3_0...5	dedicatedData21...26
xxxReserved29 [1)]	ACT_CMP1	dedicatedData27
xxxReserved30 [1)]	ACT_CMP2	dedicatedData28
faultModule	MDL_DEFECT	moduleFault
internFault	INT_FAULT	internalFault
extFault	EXT_FAULT	externalFault
faultChannel	PNT_INFO	channelFault
faultExtVoltage	EXT_VOLTAGE	externalVoltageFault
faultConnector	FLD_CONNCTR	connectorFault
invalidConfig	NO_CONFIG	configFault
invalidPara	CONFIG_ERR	configInvalid
moduleType	MDL_TYPE	moduleType
faultSubModule	SUB_MDL_ERR	submoduleFault
faultCommunication	COMM_FAULT	communicationFault
moduleStop	MDL_STOP	moduleStop
faultWatchdog	WTCH_DOG_FLT	watchdogFault
faultIntPower	INT_PS_FLT	intPowerSupplyFault
xxxReserved47 [1)]	PRIM_BATT_FLT	dedicatedData45
xxxReserved48 [1)]	BCKUP_BATT_FLT	dedicatedData46
xxxReserved31 [1)]	RESERVED_2	dedicatedData29
faultRack	RACK_FLT	rackFault
faultDevice	PROC_FLT	deviceFault
faultEprom	EPROM_FLT	EPROMFault
faultRam	RAM_FLT	RAMFault
faultAdc	ADU_FLT	ADUFault

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
faultFuse	FUSE_FLT	fuseFault
lostProcessAlarm	HW_INTR_FLT	processAlarmFault
xxxReserved32 [1]	RESERVED_3	dedicatedData30
chType	CH_TYPE	channelType
lenDiagData	LGTH_DIA	lengthDiagnosticData
chNumber	CH_NO	channelNumber
groupErrorChannel1	GRP_ERR1	groupFault1
xxxGroupErrorChannel2 [1]	GRP_ERR2	groupFault2
xxxReserved33..38 [1]	D_BIT7_2...7	dedicatedData31..36
faultCh1SignalA	CH1_SIGA	channel1SignalAFault
faultCh1SignalB	CH1_SIGB	channel1SignalBFault
faultCh1SigZero	CH1_SIGZ	channel1SignalZeroFault
faultChannel1	CH1_BETW	channel1ChannelFault
faultCh1EncSupply	CH1_5V2	channel1EncoderSupplyFault
xxxReserved39..41 [1]	D_BIT8_5...7	dedicatedData37..39
xxxReserved40 [1]	D_BYTE9	dedicatedData40
faultCh2SignalA	CH2_SIGA	channel2SignalAFault
faultCh2SignalB	CH2_SIGB	channel2SignalBFault
faultCh2SigZero	CH2_SIGZ	channel2SignalZeroFault
faultChannel2	CH2_BETW	channel2ChannelFault
faultCh2EncSupply	CH2_5V2	channel2EncoderSupplyFault
xxxReserved43..45 [1]	D_BIT10_5...7	dedicatedData41..43
xxxReserved46 [1]	D_BYTE11	dedicatedData44

[1] Variable for internal FB use (not relevant to users)

Table 8-87 SIMOTION and SIMATIC names for FM 350-2

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
Function block parameters		
_FM3502_control	**FC CNT2_CTR (FC 2)**	**_FB_FM3502_control**
periIn	-	inputInterface
data	DB_NO	data
periOut	-	outputInterface
startup	-	startup
_FM3502_write	**FC CNT2_WR (FC 3)**	**_FB_FM3502_write**
periIn	-	inputInterface
data	DB_NO	data
periOut	-	outputInterface

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
error	-	-
status	-	returnValue
_FM3502_read	**FC CNT2_RD (FC 4)**	**_FB_FM3502_read**
periIn	-	inputInterface
data	DB_NO	data
error	-	-
status	-	returnValue
_FM3502_diagnostic	**FC DIAG_RD (FC 5)**	**_FB_FM3502_diagnostic**
enable	-	-
data	DB_NO	data
error	-	-
status	-	returnValue
Data structure elements		
Struct_FM3502_fmData		**Struct_FM3502_data**
write (Struct_FM3502_wrJob)	**JOB_WR**	**write (Struct_FM3502_writeJob)**
execJobNumber	NO	jobNumber
busy	BUSY	busy
done	DONE	done
invalid	IMPOSS	invalid
unknown	UNKNOWN	unknown
read (Struct_FM3502_rdJob)	**JOB_RD**	**read (Struct_FM3502_readJob)**
execJobNumber	NO	jobNumber
busy	BUSY	busy
done	DONE	done
invalid	IMPOSS	invalid
unknown	UNKNOWN	unknown
xxxReserved1 [1)]	RESERV_0	dedicatedData1
xxxReserved2 [1)]	RESERV_1	dedicatedData2
moduleAddress	MOD_ADR	moduleAddress
xxxReserved3 [1)]	RESERV_2	dedicatedData3
control (Struct_FM3502_control)	**CONTROL_SIGNALS**	**control (Struct_FM3502_controlSignals)**
xxxReserved4..11 [1)]	BIT0_0...BIT0_7	dedicatedData4..11
enableOutput0	CTRL_DQ0	enableOutput0
enableOutput1	CTRL_DQ1	enableOutput1
enableOutput2	CTRL_DQ2	enableOutput2
enableOutput3	CTRL_DQ3	enableOutput3
enableOutput4	CTRL_DQ4	enableOutput4
enableOutput5	CTRL_DQ5	enableOutput5

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
enableOutput6	CTRL_DQ6	enableOutput6
enableOutput7	CTRL_DQ7	enableOutput7
setOutput0	SET_DQ0	setOutput0
setOutput1	SET_DQ1	setOutput1
setOutput2	SET_DQ2	setOutput2
setOutput3	SET_DQ3	setOutput3
setOutput4	SET_DQ4	setOutput4
setOutput5	SET_DQ5	setOutput5
setOutput6	SET_DQ6	setOutput6
setOutput7	SET_DQ7	setOutput7
enableSwGate0	SW_GATE0	softwareGate0
enableSwGate1	SW_GATE1	softwareGate1
enableSwGate2	SW_GATE2	softwareGate2
enableSwGate3	SW_GATE3	softwareGate3
enableSwGate4	SW_GATE4	softwareGate4
enableSwGate5	SW_GATE5	softwareGate5
enableSwGate6	SW_GATE6	softwareGate6
enableSwGate7	SW_GATE7	softwareGate7
xxxReserved12 [1)]	CTRL_DWORD1	dedicatedData12
xxxReserved13 [1)]	CTRL_DWORD2	dedicatedData13
xxxReserved14 [1)]	CTRL_DWORD3	dedicatedData14
checkback (Struct_FM3502_checkback)	**CHECKBACK_SIGNALS**	**checkback (Struct_FM3502_checkback)**
xxxReserved15 [1)]	BIT0_0	dedicatedData15
testModePg	STS_TFB	testMode
xxxReserved16..17 [1)]	BIT0_2...BIT0_3	dedicatedData16...17
dataError	DATA_ERR	dataError
xxxReserved18..19 [1)]	BIT0_5...BIT0_6	dedicatedData18...19
parameterized	PARA	parameterized
stateCmpValue0	STS_CMP0	stateComparisonValue0
stateCmpValue1	STS_CMP1	stateComparisonValue1
stateCmpValue2	STS_CMP2	stateComparisonValue2
stateCmpValue3	STS_CMP3	stateComparisonValue3
stateCmpValue4	STS_CMP4	stateComparisonValue4
stateCmpValue5	STS_CMP5	stateComparisonValue5
stateCmpValue6	STS_CMP6	stateComparisonValue6
stateCmpValue7	STS_CMP7	stateComparisonValue7
cntr0Underflow	STS_UFLW0	underflow0
cntr1Underflow	STS_UFLW1	underflow1
cntr2Underflow	STS_UFLW2	underflow2
cntr3Underflow	STS_UFLW3	underflow3

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
cntr4Underflow	STS_UFLW4	underflow4
cntr5Underflow	STS_UFLW5	underflow5
cntr6Underflow	STS_UFLW6	underflow6
cntr7Underflow	STS_UFLW7	underflow7
cntr0Overflow	STS_OFLW0	overflow0
cntr1Overflow	STS_OFLW1	overflow1
cntr2Overflow	STS_OFLW2	overflow2
cntr3Overflow	STS_OFLW3	overflow3
cntr4Overflow	STS_OFLW4	overflow4
cntr5Overflow	STS_OFLW5	overflow5
cntr6Overflow	STS_OFLW6	overflow6
cntr7Overflow	STS_OFLW7	overflow7
cntr0Reverse	STS_DIR0	counter0Reverse
cntr1Reverse	STS_DIR1	counter1Reverse
cntr2Reverse	STS_DIR2	counter2Reverse
cntr3Reverse	STS_DIR3	counter3Reverse
cntr4Reverse	STS_DIR4	counter4Reverse
cntr5Reverse	STS_DIR5	counter5Reverse
cntr6Reverse	STS_DIR6	counter6Reverse
cntr7Reverse	STS_DIR7	counter7Reverse
input0	STS_DI0	input0
input1	STS_DI1	input1
input2	STS_DI2	input2
input3	STS_DI3	input3
input4	STS_DI4	input4
input5	STS_DI5	input5
input6	STS_DI6	input6
input7	STS_DI07	input7
output0	STS_DQ0	output0
output1	STS_DQ1	output1
output2	STS_DQ2	output2
output3	STS_DQ3	output3
output4	STS_DQ4	output4
output5	STS_DQ5	output5
output6	STS_DQ6	output6
output7	STS_DQ7	output7
gate0	STS_GATE0	gate0
gate1	STS_GATE1	gate1
gate2	STS_GATE2	gate2
gate3	STS_GATE3	gate3
gate4	STS_GATE4	gate4

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
gate5	STS_GATE5	gate5
gate6	STS_GATE6	gate6
gate7	STS_GATE7	gate7
opValue0	USER_STAT_WORD0	operatingValue0
opValue1	USER_STAT_WORD1	operatingValue1
opValue2	USER_STAT_WORD2	operatingValue2
opValue3	USER_STAT_WORD3	operatingValue3
loadValue0	LOAD_VAL0	loadValue0
loadValue1	LOAD_VAL1	loadValue1
loadValue2	LOAD_VAL2	loadValue2
loadValue3	LOAD_VAL3	loadValue3
loadValue4	LOAD_VAL4	loadValue4
loadValue5	LOAD_VAL5	loadValue5
loadValue6	LOAD_VAL6	loadValue6
loadValue7	LOAD_VAL7	loadValue7
prepValue0	LOAD_PREPARE_VAL0	preparedValue0
prepValue1	LOAD_PREPARE_VAL1	preparedValue1
prepValue2	LOAD_PREPARE_VAL2	preparedValue2
prepValue3	LOAD_PREPARE_VAL3	preparedValue3
prepValue4	LOAD_PREPARE_VAL4	preparedValue4
prepValue5	LOAD_PREPARE_VAL5	preparedValue5
prepValue6	LOAD_PREPARE_VAL6	preparedValue6
prepValue7	LOAD_PREPARE_VAL7	preparedValue7
cmpValue0	CMP_VAL0	comparisonValue0
cmpValue1	CMP_VAL1	comparisonValue1
cmpValue2	CMP_VAL2	comparisonValue2
cmpValue3	CMP_VAL3	comparisonValue3
cmpValue4	CMP_VAL4	comparisonValue4
cmpValue5	CMP_VAL5	comparisonValue5
cmpValue6	CMP_VAL6	comparisonValue6
cmpValue7	CMP_VAL7	comparisonValue7
actCntrValue0	ACT_CNTV0	actualCounterValue0
actMeasValue0	ACT_MSRV0	actualMeasuringValue0
actCntrValue1	ACT_CNTV1	actualCounterValue1
actMeasValue1	ACT_MSRV1	actualMeasuringValue1
actCntrValue2	ACT_CNTV2	actualCounterValue2
actMeasValue2	ACT_MSRV2	actualMeasuringValue2
actCntrValue3	ACT_CNTV3	actualCounterValue3
actMeasValue3	ACT_MSRV3	actualMeasuringValue3
actCntrValue4	ACT_CNTV4	actualCounterValue4
actMeasValue4	ACT_MSRV4	actualMeasuringValue4

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
actCntrValue5	ACT_CNTV5	actualCounterValue5
actMeasValue5	ACT_MSRV5	actualMeasuringValue5
actCntrValue6	ACT_CNTV6	actualCounterValue6
actMeasValue6	ACT_MSRV6	actualMeasuringValue7
actCntrValue7	ACT_CNTV7	actualCounterValue7
actMeasValue7	ACT_MSRV7	actualMeasuringValue7
diagnostic (Struct_FM3502_diagInfo)	**DIAGNOSTIC_INT_INFO**	**diagnostic (DIAGNOSTIC_INT_INFO_TYPE)**
xxxReserved20..23 [1)]	BYTE0...BYTE3	dedicatedData20..23
chType	BYTE4	channelType
chInfoLength	BYTE5	lengthChannelInfo
numOfChannel	BYTE6	numberOfChannel
chFault	BYTE7	channelFault
cntr0Fault	BYTE8	counter0Fault
cntr1Fault	BYTE9	counter1Fault
cntr2Fault	BYTE10	counter2Fault
cntr3Fault	BYTE11	counter3Fault
cntr4Fault	BYTE12	counter4Fault
cntr5Fault	BYTE13	counter5Fault
cntr6Fault	BYTE14	counter6Fault
cntr7Fault	BYTE15	counter7Fault

[1)] Variable for internal FB use (not relevant to users)

Table 8-88 SIMOTION and SIMATIC names for FM 352

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
Function block parameters		
_FM352_initialize	**FC CAM_INIT (FC 0)**	**_FB_FM352_initialize**
execute	-	-
ctrlData	DB_NO	controlData
done	-	-
error	-	-
_FM352_control	**FC CAM_CTRL (FC 1)**	**_FB_FM352_control**
periIn	-	inputInterface
ctrlData	DB_NO	controlData
paraData	DB_NO	parameterData
periOut	-	outputInterface
error	-	-

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
status	-	returnValue
startup	-	startup
_FM352_diagnostic	**FC CAM_DIAG (FC 2)**	**_FB_FM352_diagnostic**
enable	-	-
periIn	-	inputInterface
diagData	DB_NO	diagnosticData
error	-	-
status	-	returnValue
Data structure elements		
Struct_FM352_ctrlData	-	**Struct_FM352_controlData**
moduleAddress	MOD_ADDR	moduleAddress
FmType	FM_TYPE	FMType
enableSimNegative	DIR_M	simulateNegative
enableSimPositive	DIR_P	simulatePositive
enableOutputCam	CAM_EN	enableOutputCam
enableTrack0Counter	CNTC0_EN	enableCounterTrack0
enableTrack1Counter	CNTC1_EN	enableCounterTrack1
enableTrack	TRACK_EN	enableTrack
diagDataChanged	DIAG	diagnosticDataModify
dataError	DATA_ERR	dataError
parameterized	PARA	parameterized
outputCamActive	CAM_ACT	outputCamActive
synchronized	SYNC	synchronized
measDone	MSR_DONE	measuringDone
dirNegative	GO_M	negativeDirection
dirPositive	GO_P	positiveDirection
hystZone	HYS	hysteresisZone
floatActValue	FVAL_DONE	floatingActualValue
actPosition	ACT_POS	actualPosition
trackSignals	TRACK_OUT	trackSignals
enableEdgeDetection	EDGE_ON	enableEdgeDetection
enableSimulation	SIM_ON	enableSimulation
enableLenMeasuring	MSR_ON	enableLengthMeasuring
execRetrigRefPoint	REFTR_ON	retriggerReferencePoint
switchOffSwLimit	SSW_OFF	switchOffSoftwareLimit
execWrMachineData	MDWR_EN	writeMachineData
execWrActivateMData	MD_EN	writeActivateMachineData
execWrActValueRevoke	AVALREM_EN	writeActualValueRevoke
execWrOutputCamData1	CAM1WR_EN	writeOutputCamData1

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
execWrOutputCamData2	CAM2WR_EN	writeOutputCamData2
execWrOutputCamData3	CAM3WR_EN	writeOutputCamData3
execWrOutputCamData4	CAM4WR_EN	writeOutputCamData4
execWrOutputCamData5	CAM5WR_EN	writeOutputCamData5
execWrOutputCamData6	CAM6WR_EN	writeOutputCamData6
execWrOutputCamData7	CAM7WR_EN	writeOutputCamData7
execWrOutputCamData8	CAM8WR_EN	writeOutputCamData8
execWrSetRefPoint	REFPT_EN	writeReferencePoint
execWrActValue	AVAL_EN	writeActualValue
execWrActValSetOnTheFly	FVAL_EN	writeActualValueSettingOnTheFly
execWrZeroOffset	ZOFF_EN	writeZeroOffset
execWrOutputCamEdge1	CH01CAM_EN	writeOutputCamEdge1
execWrOutputCamEdge16	CH16CAM_EN	writeOutputCamEdge16
execRdMachineData	MDRD_EN	readMachineData
execRdOutputCamData1	CAM1RD_EN	readOutputCamData1
execRdOutputCamData2	CAM2RD_EN	readOutputCamData2
execRdOutputCamData3	CAM3RD_EN	readOutputCamData3
execRdOutputCamData4	CAM4RD_EN	readOutputCamData4
execRdOutputCamData5	CAM5RD_EN	readOutputCamData5
execRdOutputCamData6	CAM6RD_EN	readOutputCamData6
execRdOutputCamData7	CAM7RD_EN	readOutputCamData7
execRdOutputCamData8	CAM8RD_EN	readOutputCamData8
execRdMeasValue	MSRRD_EN	readMeasuringValue
execRdCntrValueTrack	CNTTRC_EN	readCounterValueTrack
execRdActPosition	ACTPOS_EN	readActualPosition
execRdEncValue	ENCVAL_EN	readEncoderValue
execRdOutputCamData	CAMOUT_EN	readOutputCamData
xxxEnableScom [1)]	-	scom_en
xxxEnableSfct [1)]	-	sfct_en
enableEdgeDetectDone	EDGE_D	enableEdgeDetectionDone
enableSimDone	SIM_D	enableSimulationDone
enableLenMeasDone	MSR_D	enableLengthMeasuringDone
retrigRefPointDone	REFTR_D	retriggerReferencePointDone
switchOffSwLimDone	SSW_D	switchOffSoftwareLimitDone
wrMdDone	MDWR_D	writeMachineDataDone
wrActivateMdDone	MD_D	writeActivateMachineDataDone
wrActValueRevokeDone	AVALREM_D	writeActualValueRevokeDone
wrOutputCamData1Done	CAM1WR_D	writeOutputCamData1Done
wrOutputCamData2Done	CAM2WR_D	writeOutputCamData2Done
wrOutputCamData3Done	CAM3WR_D	writeOutputCamData3Done
wrOutputCamData4Done	CAM4WR_D	writeOutputCamData4Done

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
wrOutputCamData5Done	CAM5WR_D	writeOutputCamData5Done
wrOutputCamData6Done	CAM6WR_D	writeOutputCamData6Done
wrOutputCamData7Done	CAM7WR_D	writeOutputCamData7Done
wrOutputCamData8Done	CAM8WR_D	writeOutputCamData8Done
wrRefPointDone	REFPT_D	writeReferencePointDone
wrActValueDone	AVAL_D	writeActualValueDone
wrActValSetOnTheFlyDone	FVAL_D	writeActualValueSettingOnTheFlyDone
wrZeroOffsetDone	ZOFF_D	writeZeroOffsetDone
wrOutputCamEdge1Done	CH01CAM_D	writeOutputCamEdge1Done
wrOutputCamEdge16Done	CH16CAM_D	writeOutputCamEdge16Done
rdMdDone	MDRD_D	readMachineDataDone
rdOutputCamData1Done	CAM1RD_D	1readOutputCamDataDone
rdOutputCamData2Done	CAM2RD_D	readOutputCamData2Done
rdOutputCamData3Done	CAM3RD_D	readOutputCamData3Done
rdOutputCamData4Done	CAM4RD_D	readOutputCamData4Done
rdOutputCamData5Done	CAM5RD_D	readOutputCamData5Done
rdOutputCamData6Done	CAM6RD_D	readOutputCamData6Done
rdOutputCamData7Done	CAM7RD_D	readOutputCamData7Done
rdOutputCamData8Done	CAM8RD_D	readOutputCamData8Done
rdMeasValueDone	MSRRD_D	readMeasuringValueDone
rdCntrValueTrackDone	CNTTRC_D	readCounterValueTrackDone
rdActPosDone	ACTPOS_D	readActualPositionDone
rdEncValueDone	ENCVAL_D	readEncoderValueDone
rdOutputCamDataDone	CAMOUT_D	readOutputCamDataDone
xxxScomDone [1)]	-	scom_D
xxxSfctDone [1)]	-	sfct_D
enableEdgeDetectError	EDGE_ERR	enableEdgeDetectionError
enableSimError	SIM_ERR	enableSimulationError
enableLenMeasError	MSR_ERR	enableLengthMeasuringError
retrigRefPointError	REFTR_ERR	retriggerReferencePointError
switchOffSwLimitError	SSW_ERR	switchOffSoftwareLimitError
wrMdError	MDWR_ERR	writeMachineDataError
wrActivateMdError	MD_ERR	writeActivateMachineDataError
wrActValueRevokeError	AVALREM_ERR	writeActualValueRevokeError
wrOutputCamData1Error	CAM1WR_ERR	writeOutputCamData1Error
wrOutputCamData2Error	CAM2WR_ERR	writeOutputCamData2Error
wrOutputCamData3Error	CAM3WR_ERR	writeOutputCamData3Error
wrOutputCamData4Error	CAM4WR_ERR	writeOutputCamData4Error
wrOutputCamData5Error	CAM5WR_ERR	writeOutputCamData5Error
wrOutputCamData6Error	CAM6WR_ERR	writeOutputCamData6Error
wrOutputCamData7Error	CAM7WR_ERR	writeOutputCamData7Error

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
wrOutputCamData8Error	CAM8WR_ERR	writeOutputCamData8Error
wrSetRefPointError	REFPT_ERR	writeReferencePointError
wrActValueError	AVAL_ERR	writeActualValueError
wrActValSetOnTheFlyError	FVAL_ERR	writeActualValueSettingOnTheFlyError
wrZeroOffsetError	ZOFF_ERR	writeZeroOffsetError
wrOutputCamEdge1Error	CH01CAM_ERR	writeOutputCamEdge1Error
wrOutputCamEdge16Error	CH16CAM_ERR	writeOutputCamEdge16Error
rdMdError	MDRD_ERR	readMachineDataError
rdOutputCamData1Error	CAM1RD_ERR	readOutputCamData1Error
rdOutputCamData2Error	CAM2RD_ERR	readOutputCamData2Error
rdOutputCamData3Error	CAM3RD_ERR	readOutputCamData3Error
rdOutputCamData4Error	CAM4RD_ERR	readOutputCamData4Error
rdOutputCamData5Error	CAM5RD_ERR	readOutputCamData5Error
rdOutputCamData6Error	CAM6RD_ERR	readOutputCamData6Error
rdOutputCamData7Error	CAM7RD_ERR	readOutputCamData7Error
rdOutputCamData8Error	CAM8RD_ERR	readOutputCamData8Error
rdMeasValueError	MSRRD_ERR	readMeasuringValueError
rdCntrValueTrackError	CNTTRC_ERR	readCounterValueTrackError
rdActPosError	ACTPOS_ERR	readActualPositionError
rdEncValueError	ENCVAL_ERR	readEncoderValueError
rdOutputCamDataError	CAMOUT_ERR	readOutputCamDataError
xxxErrorScom [1)]	-	scom_ERR
xxxErrorSfct [1)]	-	sfct_ERR
jobErrorId	JOB_ERR	jobError
jobBusy	JOBBUSY	jobBusy
execJobReset	JOBRESET	jobReset
xxxmeasJobErrorId [1)]	-	jobError_M
xxxMeasJobBusy [1)]	-	jobBusy_M
xxxActOrder [1)]	-	aktAuftrag
xxxNextOrder [1)]	-	naechsterAuftrag
xxxDataSetNumber [1)]	-	DS_nummer
xxxDataSetLength [1)]	-	DS_laenge
xxxDataSetStart [1)]	-	DS_anfang
xxxEdgeOnOld [1)]	-	edge_on_alt
xxxSimOnOld [1)]	-	sim_on_alt
xxxMeasOnOld [1)]	-	msr_on_alt
xxxReftrOnOld [1)]	-	reftr_on_alt
xxxSswOffOld [1)]	-	ssw_off_alt
xxxRead [1)]	-	read
xxxError [1)]	-	error
xxxEnableEdgeDetectOld [1)]	-	enableEdgeDetectionDone_OLD

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
xxxEnableSimOld [1)]	-	enableSimulationDone_OLD
xxxEnableLenMeasOld [1)]	-	enableLengthMeasuringDone_OLD
xxxRetrigRefPointOld [1)]	-	retriggerReferencePointDone_OLD
xxxSwitchOffSwLimOld [1)]	-	switchOffSoftwareLimit_D_OLD
xxxDataSet11 [1)]	-	ds11
xxxDataSet12 [1)]	-	ds12
xxxControl [1)]	-	steuer
xxxFeedback0 [1)]	-	rueck0
zeroOffset	ZOFF	zeroOffset
actValue	AVAL	actualValue
actValueSetOnTheFly	FVAL	actualValueSettingOnTheFly
refPoint	REFPT	referencePoint
outputCamNumber	CAM_NO	outputCamNumber
beginOutputCam	CAM_START	beginOfOutputCam
endOutputCam	CAM_END	endOfOutputCam
beginLenMeasuring	BEG_VAL	beginOfLengthMeasuring
endLenMeasuring	END_VAL	endOfLengthMeasuring
measRange	LEN_VAL	measuringRange
cntrValueTrack0	CNT_TRC0	counterValueTrack0
cntrValueTrack1	CNT_TRC1	counterValueTrack1
actPosition1	ACTPOS	actualPosition1
actSpeed	ACTSPD	actualSpeed
trackState1	TRACK_ID	trackState1
encValue	ENCVAL	encoderValue
cntrValueByZero	ZEROVAL	counterValueByZeroCrossing
absEncOffset	ENC_ADJ	offsetOfAbsoluteEncoder
outputCamData00_31	CAM_00_31	outputCamData00_31
outputCamData32_63	CAM_32_63	outputCamData32_63
outputCamData64_95	CAM_64_95	outputCamData64_95
outputCamData96_127	CAM_96_127	outputCamData96_127
trackState2	TRACK_ID1	trackState2
actPosition2	ACTPOS1	actualPosition2
numOfOutputCamsToSet	C_QTY	activatedOutputCam
disableDataCheck	DIS_CHECK	disableDataCheck
outputCam (Struct_FM352_outCamCtrl)	**CAM**	**outputCam (outputCamTypeFM352)**
number	CAM_NO	number
setForceDirection	C_EFFDIR	changeForceDirection
setBegin	C_CBEGIN	changeBeginOfOutputCam
setEnd	C_CEND	changeEndOfOutputCam
setActuationTime	C_LTIME	changeActuationTime
deactivate	CAM_OFF	deactivate

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
posForceDirection	EFFDIR_P	forceDirectionPositive
negForceDirection	EFFDIR_M	forceDirectionNegative
beginOutputCam	CBEGIN	beginOfOutputCam
endOutputCam	CEND	endOfOutputCam
actuationTime	LTIME	actuationTime
Struct_FM352_paraData	-	**Struct_FM352_parameterData**
xxxModuleType1 [1)]	PI_MEND	moduleType1
enableProcessAlarm	PI_CAM	enableProcessAlarm
xxxModuleType2 [1)]	PI_MSTRT	moduleType2
minEdgeDistance	EDGEDIST	minimumEdgeDistance
unitDimension	UNITS	dimensionUnit
axisType	AXIS_TYPE	axisType
endRotAxis	ENDROTAX	endOfRotaryAxis
encType	ENC_TYPE	encoderType
lenPerRevolution	DISP_REV	lengthPerRevolution
incPerRevolution	INC_REV	incrementsPerRevolution
cntOfRevolutions	NO_REV	numberOfRevolution
baudRate	BAUDRATE	baudrate
refPoint	REFPT	referencePoint
absEncOffset	ENC_ADJ	offsetOfAbsoluteEncoder
refPointTrigMode	RETR_TYPE	referencePointTriggerMode
cntrDirection	CNT_DIR	counterDirection
openCircuit	MON_WIRE	openCircuit
transmissionError	MON_FRAME	transmissionError
missingPulse	MON_PULSE	missingPulse
swLimitStart	SSW_STRT	softwareLimitSwitchBegin
swLimitEnd	SSW_END	softwareLimitSwitchEnd
numOfOutputCamsToSet	C_QTY	activatedOutputCam
hysteresis	HYS	hysteresis
simSpeed	SIM_SPD	simulationSpeed
ctrlTrackOutputs	TRACK_OUT	trackOutControl
enableInput3	EN_IN_I3	enableInput3
xxxEnableInput4..10 [1)]	EN_IN_I4	enableInput4...10
track0CntrOutputCam	SPEC_TRC0	counterOutputCamTrack0
track1CntrOutputCam	SPEC_TRC1	counterOutputCamTrack1
track2CntrOutputCam	SPEC_TRC2	counterOutputCamTrack2
track0CntrLimit	CNT_LIM0	counterLimitTrack0
track1CntrLimit	CNT_LIM1	counterLimitTrack1
outputCam (Struct_FM352_outCam-Para)	**CAM**	**outputCam (outputCamData)**
valid	CAMVALID	valid

Name in SIMOTION System, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
posForceDirection	EFFDIR_P	forceDirectionPositive
negForceDirection	EFFDIR_M	forceDirectionNegative
outputCamType	CAM_TYPE	typeOfOutputCam
switchOnAlarm	PI_SW_ON	switchOnAlarm
switchOffAlarm	PI_SW_OFF	switchOffAlarm
trackNumber	TRACK_NO	trackNumber
beginOutputCam	CBEGIN	beginOfOutputCam
endOutputCam	CEND	endOfOutputCam
actuationTime	LTIME	actuationTime
Struct_FM352_diagData	-	**Struct_FM352_diagnosticData**
moduleAddress	MOD_ADDR	moduleAddress
xxxDataSet [1)]	-	ds
xxxDiagnosis0 [1)]	-	DIAG0
xxxFeedback0 [1)]	-	rueck0
xxxActOrder [1)]	-	aktAuftrag
xxxNextOrder [1)]	-	naechsterAuftrag
xxxDataSetNumber [1)]	-	DS_nummer
xxxDataSetLength [1)]	-	DS_laenge
xxxCntOfBuffers [1)]	-	anzPuffer
xxxEnableDataSet236 [1)]	-	ds236_en
xxxDataSet [1)]	-	ds237_en
jobErrorId	JOB_ERR	jobError
jobBusy	JOBBUSY	jobBusy
diagInformation	DIAGRD_EN	diagnosticInformation
numOfValidEntries	DIAG_CNT	numberOfValidEntries
diagEntry (Struct_FM352_diagType)	**DIAG**	**diagnosticEntry (diagnosticData)**
incomingAlarm	STATE	alarmIncoming
internFault	INTF	internalFault
extFault	EXTF	externalFault
faultClass	FCL	faultClass
faultNumber	FNO	faultNumber
chNumber	CH_NO	chanNumber
outputCamNumber	CAM_NO	outputCamNumber

[1)] Variable for internal FB use (not relevant to users)

8.6.7.2 List of abbreviations

Table 8-89 Abbreviations

Abbreviation	Meaning
ADC	Analog-digital converter
DI	Digital Input
DP	Distributed I/O
EPROM	Erasable Programmable Read-Only Memory (Erasable programmable read-only memory)
FB	Function block
FM	Function module
IM	Interface Module (SIMATIC S7-300 interface module)
IN	Input parameter
IN/OUT	In/out parameter
LAD	Ladder Logic
LED	Light Emitting Diode (LED displays)
OB	Organization block
OUT	Output parameter
Programming device/PC	Programming device / Personal computer
PS	Power Supply (SIMATIC S7-300 power supply)
RAM	Random Access Memory
SF	System Fault (System error)
SW	Software

8.7 Supplement for the ET 200S frequency converter

Preface

Contents of this Function Manual

This **document** is a component of the **SIMOTION Programming - References** documentation package.

This documentation is a supplement to the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions *(03/2005 Edition)*.

This documentation is supplied with the SIMOTION SCOUT in electronic form!

This manual describes how you can control the ET 200S frequency converter with the aid of the **_ET200S_FC_control** function block (Article No.: 6SL3244-0SA00-1AA**0**) from a SIMOTION device.

Function block

The **_ET200S_FC_control** function block is used for the communication between the SIMOTION system and the ET 200S frequency converter and is a component of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.7.1 Fundamental safety instructions

8.7.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.7.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.7.2 Description

8.7.2.1 General

Overview

This manual describes the differences in the operation of the ET 200S FC (FC = Frequency Converter) in the SIMOTION system compared with operation in the SIMATIC system.

Note

This manual is a supplement to the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).

This documentation is supplied with the SIMOTION SCOUT in electronic form!

Requirements

The following are required for the standard function described in this document:

- SIMOTION SCOUT, V4.1 SP1 or higher
- SIMOTION Kernel, V4.1 SP1 or higher
- ET 200S frequency converter with ICU24, software release V1.02 or higher

Note

The functions associated with the **_ET200S_FC_control** function block can only be used on the ET 200S frequency converter with ICU (Article No: 6SL3244-0SA00-1AA**0**).

8.7.2.2 Product description

ET 200S frequency converter

The ET 200S FC is a modular frequency converter that is completely embedded in the distributed I/O system of the ET 200S.

During generator operation of the connected motor, the ET 200S FC feeds back energy into the supply system.

Function block

The **_ET200S_FC_control** function block is provided for the cyclic communication and the parameter transfer of the ET 200S FC with SIMOTION. This function block is described in this manual.

Note

The **_ET200S_FC_control** function block **cannot** be used to control the ET 200S FC (Article No.: 6SL3244-0SA00-1AA**1**). Standard telegram 1 should be used for this frequency converter.

Functionality

The functionality of the **_ET200S_FC_control** function block and the ET 200S FC is the same as when used in the SIMATIC S7 automation system. There are differences in the execution of the data transfer and the handling of the FB.

Possible applications

The ET 200S FC can be operated in a distributed ET 200S station via PROFIBUS DP on a SIMOTION controller.

Several ET 200S frequency converters can be operated in one ET 200S station. The ET 200S interface module IM 151-1 is used for the connection to PROFIBUS DP.

The following figure shows a distributed configuration with SIMOTION C2xx.

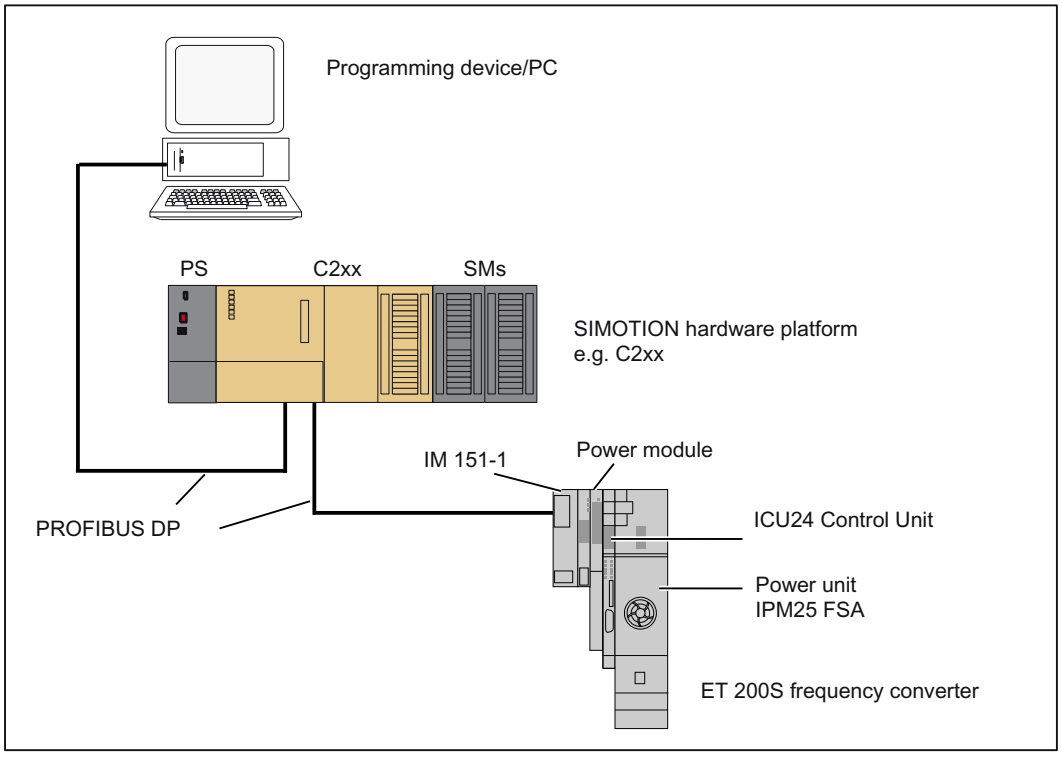


Figure 8-35 Connection of the ET 200S frequency converter to the SIMOTION C2xx device (example)

8.7.2.3 Installation and connection

You must perform the following steps to be able to operate the ET 200S frequency converter using the SIMOTION system:

1. Install and connect the ET 200S interface module (IM) with the components of the ET 200S FC.
2. Establish the PROFIBUS connection between the IM and the SIMOTION device.
3. Set the PROFIBUS DP node address on the IM.
4. Switch on the terminating resistor at the first and last bus node.

Note

The commissioning of the ET 200S frequency converter can be performed either via

- the STARTER functionality integrated in the SIMOTION SCOUT engineering system (via "Insert single drive"), or
- or the STARTER engineering software.

For a description of commissioning and steps 1 to 4, see the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).

5. Insert the ET 200S FC into the SIMOTION project (refer to Chapter Inserting the ET 200S frequency converter into a SIMOTION project (Page 6332)).
6. Parameterizing the ET 200S FC.
   To parameterize the module, go to **HW Config**. Double-click the relevant module to open the **Properties** dialog box. Select the **Parameters** tab to access the parameterization interface. There you can assign parameters for your module.
   See the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).
7. Link the function block to the SIMOTION project (refer to Chapter Integrating the function block in the user project (Page 6333)).

8.7.2.4 Inserting the ET 200S frequency converter into a SIMOTION project

Requirement

The following requirements must be met:

1. You have created a project in SIMOTION SCOUT and inserted a rack with a SIMOTION device in the hardware configuration.
2. You have configured a PROFIBUS subnet.

Note

For how to create a project and configure a PROFIBUS subnet, refer to the online help of *SIMOTION SCOUT*.

Inserting the frequency converter

1. In SIMOTION SCOUT, open the "User Projects" dialog box with the "Project" > "Open" menu command. In this dialog box, select your project and confirm with "OK".
2. Open **HW Config**.
3. In the "HW Config" window, open the **hardware catalog** with the "View" > "Catalog" menu command.
4. In the hardware catalog, open the "PROFIBUS DP" folder and the "ET 200S" subfolder and select, for example, the **IM 151-1 Standard** interface module (Article No.:6ES7 151-1AA04-0AB0 or follow-on module).
5. Use a drag-and-drop operation to place the "IM 151-1 Standard" interface module on the PROFIBUS subnet of your SIMOTION device.
   The "Properties - PROFIBUS IM 151-1 Standard Interface" dialog box opens. In this dialog box, select the address you set on the IM 151-1 (see *ET 200S Distributed I/O Device* Manual) and confirm your selection with "OK".
   The selected IM 151-1 Standard I/O device is inserted into the project.
6. The inserted I/O device still has to be equipped with the following modules: PM-D power module and ET 200S FC frequency converter, consisting of ICU24 closed-loop control module.
   Open the "Motor starter" folder and the "PM" subfolder under "IM 151-1 Standard" in the hardware catalog. Select the **PM-D 24VDC** power module.
   You insert the frequency converter by selecting in the hardware catalog under "IM151-1 Standard" (Article No.: 6ES7 151-1AA04-0AB0) in folder "Frequency converters" the closed-loop control module **ICU24** (Article No.: 6SL3244-0SA00-1AA0).

   Note

   You can disable or enable diagnostic alarms via parameter p8452 in the ET 200S frequency converter.
7. Save and compile your project.

8.7.3 Programming

8.7.3.1 Integrating the function block in the user project

Creating the FB instance in the user project

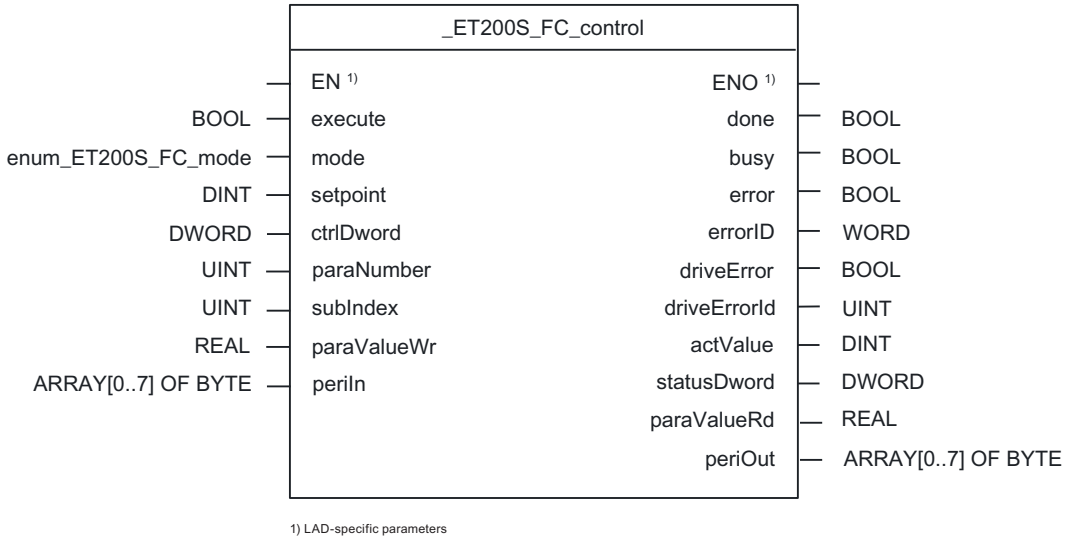
The **_ET200S_FC_control** function block is a component of the command library of the SIMOTION SCOUT engineering system. To work with the block, an instance of the function block must first be created in the user project.

Example:

```
VAR_GLOBAL
...
```

```
  myFC_control    : _ET200S_FC_control;   // create instance of FB _ET200S_FC_control
...
END_VAR
```

Call (LAD representation)



Application example

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

8.7.3.2 Addressing the ET 200S frequency converter

Overview

Communication between the SIMOTION device and the ET 200S FC takes place through direct access to the I/O. I/O variables are used to address the direct read/write access to the I/O.

Creating I/O variables

For the cyclic data transfer with the ET 200S frequency converter, you must create an I/O variable in the symbol browser for each I/O input and output. When creating the I/O variable, enter the configured address from the hardware configuration in the "I/O address" column (see example below). Enter a field length of 8 in the "Field length" column.

With these specifications, an I/O variable of type ARRAY[0..7] of BYTE is created in the SIMOTION project for cyclic data transfer.

Example:

Determining addresses from **HW Config**:

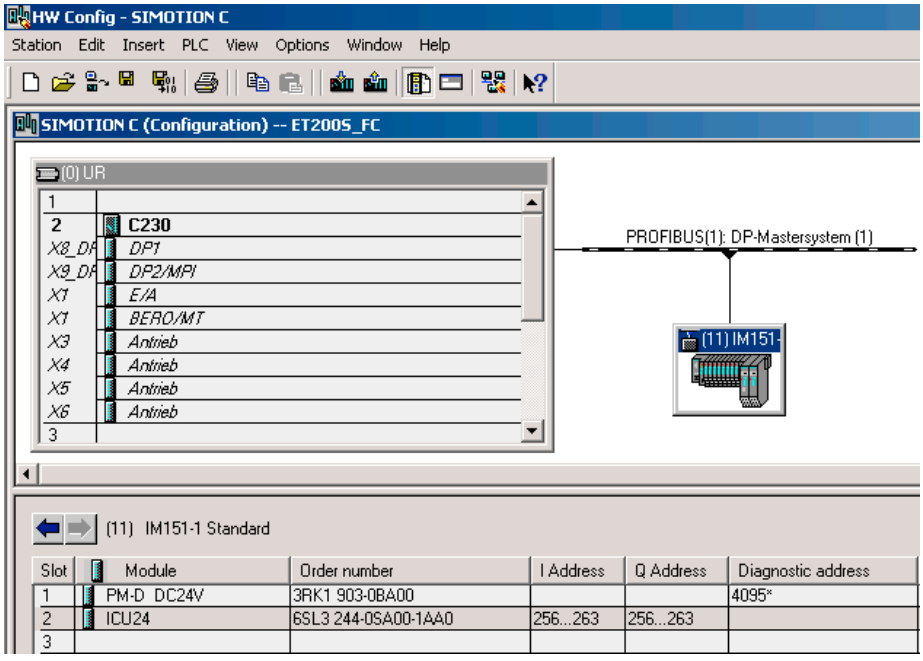


Figure 8-36 Addresses in HW Config

Creating the I/O variables in the symbol browser:

	Name	I/O address	Read only	Data type	Field length
1	myperiin	PIB 256		Array	8
2	myperiout	PQB 256	☐	Array	8

Figure 8-37 Addressing with I/O variable

The I/O addresses in the symbol browser shown in the figure have the following meanings:

- "PIB" = Peripheral Input Byte
- "PQB" = Peripheral Output Byte

Parameter transfer

The I/O variable for the I/O inputs is transferred to the **periIn** input parameter.

The prepared data for the I/O outputs are supplied by the function block at the **periOut** output parameter. The **periOut** output parameter must be assigned to the I/O variable for the I/O outputs.

Note

When booting the firmware of the ET 200S frequency converter from the MMC card, it may take more than 500 ms for the ET 200S frequency converter to ramp up.

To avoid I/O area access errors when the frequency converter has not yet ramped up, accesses to the I/O inputs and outputs of the frequency converter must be performed with the **_getSafeValue** and **_setSafeValue** system functions (see sample program **UNIT E_FC_ctr**).

Note

For additional information, refer to:

- SIMOTION SCOUT online help
- Programming Manuals for the relevant programming languages, such as:
  - SIMOTION ST, Structured Text Programming Manual
  - SIMOTION MCC, Motion Control Chart Programming Manual
  - SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

8.7.4 Parameterization

8.7.4.1 Function block _ET200S_FC_control

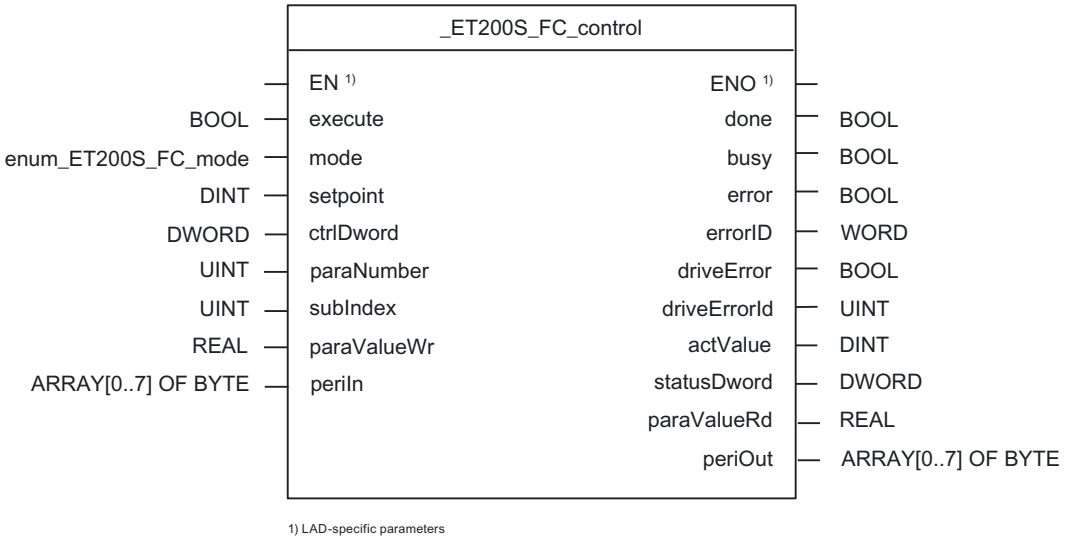
Task

You can control the ET 200S FC via your user program with the FB **_ET200S_FC_control**.

The FB **_ET200S_FC_control** supports the following functions:

- Drive control:
  - Sending control words
  - Receiving status words
- Diagnostics:
  - Displaying drive faults
  - Acknowledging drive errors
  - Displaying parameterization errors in the data exchange
- Reading and writing of drive parameters

Call (LAD representation)



Parameter description

The following table contains all the parameters of the function block **_ET200S_FC_control**.

Table 8-90 Parameters of the FB _ET200S_FC_control

Name	Type [1]	Data type	Default	Meaning
execute	IN	BOOL	FALSE	Start of a data transfer from/to the ET 200S FC with rising edge
mode	IN	enum_ET200S_FC_mode (READ, WRITE, DATA1, DATA2, RESET_ERR)	READ	Selection of data transfer type • READ: Read individual parameter • WRITE: Write individual parameter • DATA1: Data set 1 is set active (p0810 to 0.0) with rising edge on parameter **execute**. • DATA2: Data set 2 is set active (p0810 to 1.0) with rising edge on parameter **execute**. • RESET_ERR: Acknowledgement of parameterization and drive errors
periIn	IN	ARRAY [0 to 7] of BYTE	8 (16#00)	Transfer I/O inputs of ET 200S FC to FB
setpoint	IN	DINT	0	Normalized setpoint without units for the ET 200S FC, transferred cyclically [4]
ctrlDword	IN	DWORD	16#00000000	Control words (STW 1 and STW 2) of the ET 200S FC, transferred cyclically [5]
paraNumber	IN	UINT	0	Parameter number to be read or written
subIndex	IN	UINT	0	Index of an indexed parameter for the ET 200S FC
paraValueWr	IN	REAL	0.0	Parameter value to be written
done	OUT	BOOL	FALSE	**TRUE** = Read/write parameter was processed without errors

Name	Type [1]	Data type	Default	Meaning
busy	OUT	BOOL	FALSE	**TRUE** = Read/write parameter active
error	OUT	BOOL	FALSE	**TRUE** = parameterization error
errorID	OUT	WORD	16#0000	Error number of the parameterization error (error number 0..108, or 16#0000..16#006C) [3] **Note**: If **error** = **TRUE** and **errorID** = **16#0000**, the following error is pending: "Invalid parameter number"
driveError	OUT	BOOL	FALSE	**TRUE** = Drive error is pending (refer to the **driveErrorId** parameter)
driveErrorId	OUT	UINT	0	Reason for the error (decimal format) Value corresponds to parameter r0947 (errors) [2]
periOut	OUT	ARRAY [0 to 7] of BYTE	8 (16#00)	Prepared FB data for I/O outputs of the ET 200S FC
actValue	OUT	DINT	0	Display of the parameter value of p2051.1 (process data on field bus) [2]
statusDword	OUT	DWORD	16#00000000	Status words (ZSW 1 and ZSW 2) of the ET 200S FC [2]
paraValueRd	OUT	REAL	0.0	Read parameter value

[1] Parameter types: IN = input parameter, OUT = output parameter

[2] See the *SIMATIC ET 200S FC Frequency Converter* Parameter Manual (03/2005 Edition).

[3] See the *SIMATIC ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).

[4] Calculation in the converter:

$$\text{Converter set frequency} = \frac{\text{Value in setpoint} \cdot \text{reference frequency in drive parameter p2000}}{\text{Normalizing value}}$$

Example 1: - Normalizing value = 16,384 (4000 hex)
- p2000 = 50 Hz
- Desired set frequency = 20 Hz

$$\text{Setpoint} = \frac{20\ \text{Hz} \cdot 16{,}384}{50\ \text{Hz}} = 6{,}553$$

Example 2: - Normalizing value = 16,384 (4000 hex)
- Speed (50 Hz ≙ max. speed ≙ 1,350 rpm)
- Desired speed = 1,000 rpm

$$\text{Setpoint} = \frac{1{,}000\ \text{rpm} \cdot 16{,}384}{1{,}350\ \text{rpm}} = 12{,}136$$

Note

You can find the normalizing value in the *SIMATIC ET 200S FC Frequency Converter* Parameter Manual (03/2005 Edition) under parameter p2000.

The **execute** parameter is only used to start acyclic data transfer from/to the frequency converter.

When the frequency converter is switched on, the enables must be set in the control word and the setpoint (**setpoint**) takes immediate effect.

The effect of the control word depends on the parameterization in the frequency converter.

[5] Bit assignment of **ctrlDword** (control words 1 and 2 for the ET 200S FC):

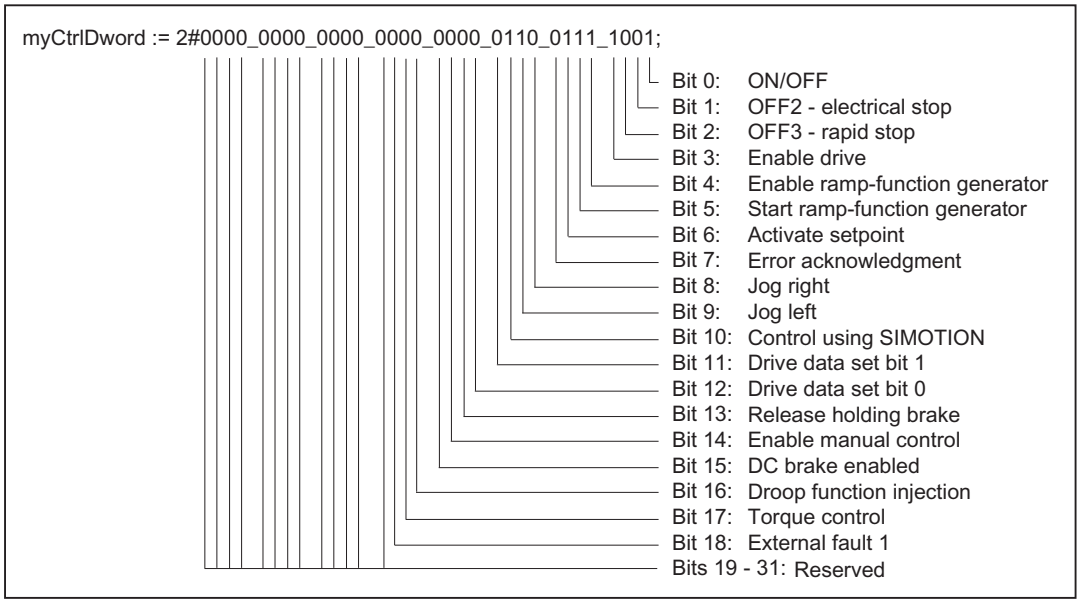


Figure 8-38 Meaning of control bits (in STW 1 and STW 2 for the ET 200S FC)

Description of functions

When the **_ET200S_FC_control** function block is called, the cyclic data is written and updated and jobs (such as reading/writing parameter) can be carried out.

The values in the input parameters **periIn**, **setpoint**, and **ctrlDword** are transmitted cyclically - regardless of the **busy**, **done**, **execute**, and **mode** parameters - and the output parameters **periOut**, **actValue**, and **statusDword** are updated cyclically.

Jobs are started with a rising edge on the input parameter **execute**. The type of data transfer is selected in the input parameter **mode**.

You can choose from the following options in the input parameter **mode**:

- READ: Individual parameters are read from the ET 200S FC (factory setting).
- WRITE: Individual parameters are written to the ET 200S FC.
- DATA1: The drive parameter p0810 is set to 0.0 (command data set 1 of the drive active).
- DATA2: The drive parameter p0810 is set to 1.0 (command data set 2 of the drive active).
- RESET_ERR: Resetting of parameterization and drive errors.

A job is started with a rising edge on the input parameter **execute** and the output parameters set to **done** = **FALSE** and **busy** = **TRUE**.

If **mode** = **WRITE** is selected, the parameters **paraNumber**, **subIndex**, and **paraValueWr** are accepted with a rising edge on the input parameter **execute**. An active data transfer is indicated by **busy** = **TRUE**. When a job is completed without errors, the following parameters are displayed: **busy** = **FALSE**, **done** = **TRUE**, and **error** = **FALSE**. When a job is completed with errors (parameterization errors), the following parameters are displayed: **busy** = **FALSE**, **done** = **FALSE**, and **error** = **TRUE**. The error number of the parameterization error is displayed in output parameter **errorID**. Parameterization errors are acknowledged with a rising edge on the input parameter **execute** and **mode** = **RESET_ERR** selected. If acknowledgement is successful, the output parameter is set to **done** = **TRUE** and the **error** and **errorID** parameters are reset.

The drive parameter p0810 is set to 0.0 (command data set 1 of the drive active) with a rising edge on the input parameter **execute** and **mode** = **DATA1** selected. This drive parameter is set to 1.0 (command data set 2 of the drive active) with a rising edge on the input parameter **execute** and **mode** = **DATA2** selected. This means that control command acceptance can be switched from "USS on RS232" to "Fieldbus", if the relevant P0700 setting is made.

Note

If another job (e.g. reading parameters when a write job is active) is started when **busy** = **TRUE**, this is disregarded. A new job is only executed when **busy** = **FALSE**.

Note

When booting the firmware of the ET 200S frequency converter from the MMC card, it may take more than 500 ms for the ET 200S frequency converter to ramp up.

To avoid I/O area access errors when the frequency converter has not yet ramped up, accesses to the I/O inputs and outputs of the frequency converter must be performed with the **_getSafeValue** and **_setSafeValue** system functions (see sample program **UNIT E_FC_ctr** on the "SIMOTION Utilities & Applications" CD-ROM).

Graphic overview of the functionality

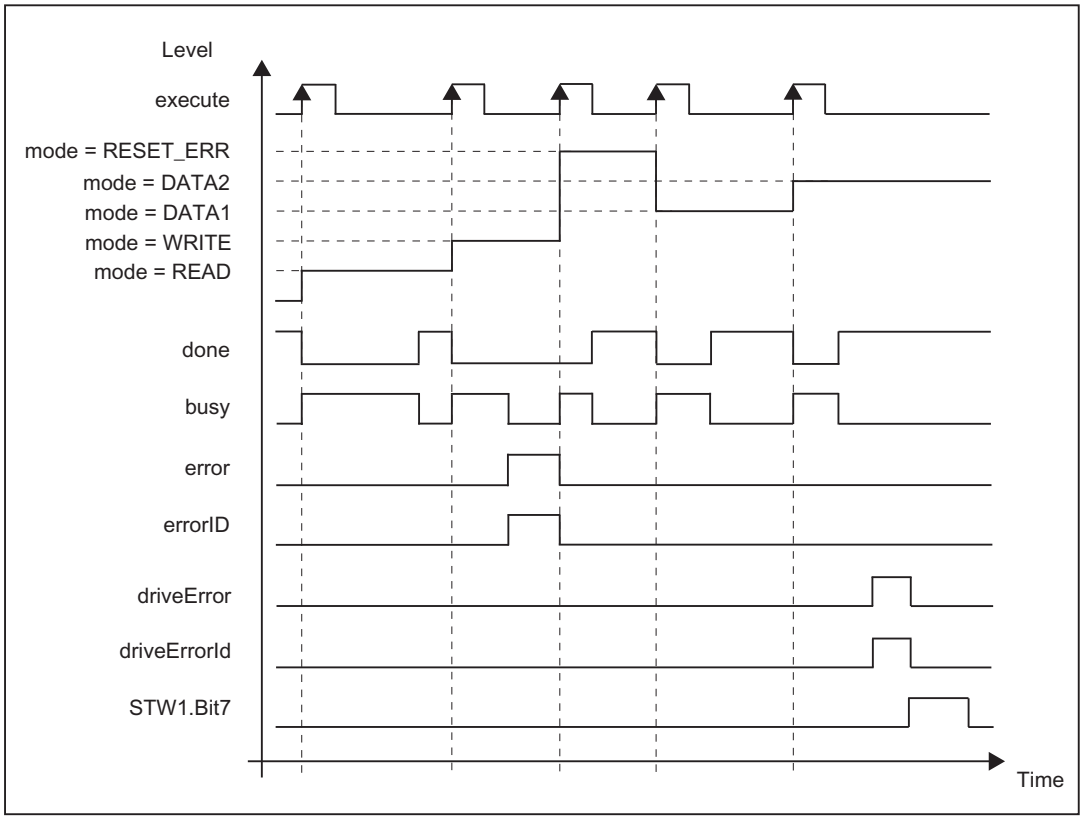


Figure 8-39 Signal propagation diagram

Task integration (call)

The **_ET200S_FC_control** FB is designed to be called in a cyclic task and must be called in this task during each task cycle. The execution of a job can take several cycles. The user decides which cyclical task the **_ET200S_FC_control** FB is called in. There are no restrictions with respect to this in the **_ET200S_FC_control** FB. The call can be performed in all cyclic tasks. A fixed time frame (e.g. **IPOSynchronousTask**) is not required for processing.

A separate instance of the **_ET200S_FC_control** function block must be created for the control and communication tasks of each frequency converter used.

Error messages

The drive errors of the ET 200S FC are output at the output parameters **driveError** and **driveErrorId** of the **_ET200S_FC_control** FB. The output parameter **driveErrorId** displays the associated error number (from r0947). For a description of the error numbers, see the *SIMATIC ET 200S FC Frequency Converter* Parameter Manual (03/2005 Edition).

Rectify the cause of the drive error. Drive errors are acknowledged with a rising edge on the input parameter **execute** and **mode** = **RESET_ERR** selected.

Parameterization errors are signaled at the output parameters **error** and **errorID** , and reset with a rising edge on input parameter **execute** and **mode** = **RESET_ERR** selected.

8.7.4.2 Calling the function block

Procedure

Proceed as follows to work with the **_ET200S_FC_control** function block in your user project (the numbers shown in the following program segment correspond to the steps listed below):

1. Create an instance of the function block.
2. Call instance of the function block.
3. Transfer input parameters.
4. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.
5. The data for the I/O outputs prepared by the FB are assigned to the I/O variables.

Note

The program segment is an extract from the supplied application example. The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

If you want to control several frequency converters, you must create a new variable for the FB instances with a new name for each module used.

Separate data structures have been created for the input and output parameters in the example to provide a better overview.

Call example

```
UNIT E_FC_ctrl;

INTERFACE

TYPE
   ...                                    // see example
END_TYPE

VAR_GLOBAL
```

```
 myFC_control     : _ET200S_FC_control;     // create instance of FB _ET200S_FC_control          (1)
 myFC_control_In  : Struct_FC_control_In;   // input parameter of FB _ET200S_FC_control
 myFC_control_Out : Struct_FC_control_Out;  // output parameter of the FB _ET200S_FC_control
 myRetValSetSafe  : enumSetAndGetSafeValue; // return value from system function _setsafevalue

 ...                                        // see example

END_VAR

PROGRAM ExampleET200S_FC;                  // Program in BackgroundTask

END_INTERFACE

IMPLEMENTATION

PROGRAM ExampleET200S_FC
// Program in BackgroundTask
   // call _getSafeValue -> Test I/O-variables after powerOn Reset
   myRetValGetSafe := _getSafeValue (
    variable   := myPeriIn,
    accessMode := DEFAULT_VALUE,
    getValue   := myTmpPeriIn
   );
 IF (myRetValGetSafe = OK)                      // powerOn ready
 THEN
   // CALL FB INSTANCE
   myFC_control (                                                                             (2)
    periIn      := myTmpPeriIn,                // I/O peripheral input                       (3)
    execute     := myFC_control_In.execute,    // execute / start the data transfer
    mode        := myFC_control_In.mode,       // select the mode
    paraNumber  := myFC_control_In.paraNumber, // parameter number to read or write
    subIndex    := myFC_control_In.subIndex,   // index of the parameter
    paraValueWr := myFC_control_In.paraValueWr,// value of the parameter to write
    setpoint    := myFC_control_In.setpoint,   // set the setpoint
    ctrlDword   := myFC_control_In.ctrlDword   // set the control word
   );

  // get output parameter of the FB _ET200S_FC_control                                         (4)
```

```
  myFC_control_Out.done         := myFC_control.done;         // ready
  myFC_control_Out.busy         := myFC_control.busy;         // at work
  myFC_control_Out.error        := myFC_control.error;        // error occured
  myFC_control_Out.errorID      := myFC_control.errorID;      // number of error
  myFC_control_Out.driveError   := myFC_control.driveError;   // drive error
  myFC_control_Out.driveErrorId := myFC_control.driveErrorId; // number of the drive error
  myFC_control_Out.actValue     := myFC_control.actValue;     // per default the
                                                              // actual speed value
  myFC_control_Out.statusDword  := myFC_control.statusDword;  // the status word
  myFC_control_Out.paraValueRd  := myFC_control.paraValueRd;  // value of the
                                                              // parameter to read
 END_IF

  // write peripheral output                                                    (5
  myRetValSetSafe := _setSafeValue (                                            )
    variable   := myPeriOut,                                // I/O peripheral output
    value      := myFC_control.periOut,
    accessMode := NO_CHANGE
   );

END_PROGRAM                          // end Program in BackgroundTask

END_IMPLEMENTATION
```

8.7.5 Application example

8.7.5.1 General

Task

The application example shows how you can control the ET 200S frequency converter with the aid of the **_ET200S_FC_control** function block. You can carry out the following functions with this example:

- Reset parameterization and drive errors
- Read parameter
- Write parameter
- Activate command data set 1
- Activate command data set 2
- Start the motor at a specified speed
- Stop a running motor

Requirements

The following requirements must be met:

- You have commissioned the ET 200S frequency converter (Article No.: 6SL3244-0SA00-1AA**0**) with the connected motor.
  For information on commissioning the frequency converter, see the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).
  This document is supplied with SIMOTION SCOUT in electronic form.
- You have completed the addressing of the ET 200S frequency converter (refer to chapter Addressing the ET 200S frequency converter (Page 6334)).

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your SIMOTION hardware platform, you must adapt the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

1. Dearchive and open the project containing the application example.
2. Check the hardware configuration, e.g. the PROFIBUS DP addresses.
3. **Save** and **compile** the example project. Then, you can download the example to the SIMOTION device and switch to RUN mode.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration of the example to the available hardware.
2. You can adapt the configuration of the hardware to the example.

You can adapt the configuration in the example to the available hardware.

You have assigned a module address <> 256 in **HW Config**.

1. In the symbol browser, change the I/O address to the value you assigned in **HW Config**:

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperiin	PIB 256		Array	8
2	⊞ myperiout	PQB 256	☐	Array	8

Figure 8-40 Adapting the I/O address

2. Enter the same value in the **myModuleAddr** variable of the application example.

Note

When booting the firmware of the ET 200S frequency converter from the MMC card, it may take more than 500 ms for the ET 200S frequency converter to ramp up.

To avoid I/O area access errors when the frequency converter has not yet ramped up, accesses to the I/O inputs and outputs of the frequency converter must be performed with the **_getSafeValue** and **_setSafeValue** system functions (see sample program **UNIT E_FC_ctrl** on the "SIMOTION Utilities & Applications" CD-ROM).

8.7.5.2 Sequence of the application example

Relevant variables in the application example

Table 8-91 Overview of the relevant variables

Symbol	Data type	Initial value	Meaning
myCommand	enumCommand (CMD_READ, CMD_WRITE, CMD_RESET_ERR, CMD_USE_DATA1, CMD_USE_DATA2, CMD_START, CMD_STOP)	CMD_READ	Selection of command to be executed
myCmdExecute	BOOL	FALSE	TRUE = Start selected command
myCmdDone	BOOL	FALSE	TRUE = Error-free command processing
myCmdError	INT	0	Error specification
myParaNumber	UINT	0	Number of parameter to be written
mySubIndex	UINT	0	Index of the parameter number
myParaValueWr	REAL	0.0	Parameter value to be written
myParaValueRd	REAL	0.0	Read parameter value
mySetpoint	INT	500	Speed specification in [rpm]
myMaxRevolution	INT	1,350	Maximum speed of the motor in [rpm]

Command processing

Proceed as follows to activate command processing in the sample program:

1. In the project navigator, under **Programs** select the **E_FC_ctrl Unit**.
2. Open the symbol browser in SIMOTION SCOUT.
3. Select a command in the symbol browser by setting the **myCommand** variable to the command to be executed.
4. Parameterize the command-specific variable.
5. Start the selected command by setting the **myCmdExecute** variable to **TRUE**.
6. Wait until error-free command execution is indicated by **myCmdDone** = **TRUE**. An error during command execution is specified in the **myCmdError** parameter.
7. You can now call additional commands.

The functions contained in the application example are described below.

Example: Resetting drive and parameterization errors

The function block displays drive errors at the output parameters **myFC_control.driveError** = **TRUE** and **myFC_control.driveErrorId** ≠ **16#0000**; these must be acknowledged before the next command is called. Drive errors are reset with bit 7 in the control word **ctrlDword**.

Parameterization errors are indicated at the output parameters **myFC_control.error** = **TRUE** and **myFC_control.errorID** = **16#0000**.

Example: You have set an invalid parameter number (**myParaNumber** = 9999).

To acknowledge the parameterization error, you must set the variables **myCommand** and **myCmdExecute** to **CMD_RESET_ERR** and **TRUE** respectively.
The output parameters are then reset with **myFC_control.error** = **FALSE** and **myFC_control.errorID** = **16#0000**.

Example: Read parameter

Proceed as follows to read a parameter:

1. Set the **myCommand** variable to **CMD_READ** in the symbol browser.
2. Assign the **myParaNumber** variable the parameter number to be read (e.g. "1120", startup time).
3. Set the **mySubIndex** variable to "0".
4. Start the command with **myCmdExecute** = **TRUE**.
   The parameter value read from the ET 200S FC is output in the **myParaValueRd** variable.

Example: Write parameter

Proceed as follows to write a parameter:

1. Set the **myCommand** variable to **CMD_WRITE** in the symbol browser.
2. Assign the **myParaNumber** variable the parameter number to be written (e.g. "1120", startup time).
3. Set the **mySubIndex** variable to "0".
4. Enter the parameter value to be written (e.g. "50") in the **myParaValueWr** variable.
5. Start the command with **myCmdExecute** = **TRUE**.
   The value entered in the **myParaValueWr** variable is transmitted to the ET 200S FC.

Example: Activate command data set 1

Proceed as follows to activate command data set 1:

1. Set the **myCommand** variable to **CMD_USE_DATA1** in the symbol browser.
2. Start the command with **myCmdExecute** = **TRUE**.

Example: Activate command data set 2

Proceed as follows to activate command data set 2:

1. Set the **myCommand** variable to **CMD_USE_DATA2** in the symbol browser.
2. Start the command with **myCmdExecute** = **TRUE**.

Example: Start the motor at a specified speed

Proceed as follows to get the connected motor to turn at a speed of 500 rpm:

1. Enter the value "500" in the **mySetpoint** variable.
2. Set the **myCommand** variable to **CMD_START** in the symbol browser.
3. Start the command with **myCmdExecute** = **TRUE**.
   The input parameter **myFC_Control_In.setpoint** is set to the speed parameterized in the **mySetpoint** variable accordingly and transferred to the input parameter **setpoint**.
   All enables in control word 1 are set. The motor starts to turn.
   After the speed parameterized in the **mySetpoint** variable is reached, the command is ended with **myCmdDone** = **TRUE**.

Note

When starting the motor with a rising edge on bit 0 in STW 1, bit 8 and bit 9 in STW 1 (jog right/left) must have the signal level **FALSE**.

Assignment of **ctrlDword** (STW 1 and STW 2 of ET 200S FC) when starting the motor:

`CMD_START : // start motor`	16#0000 047F hex or binary: 0000 01**00** 0111 111**1**
`CMD_JOG_RIGHT : // motor jog. Right`	16#0000 057E hex or binary: 0000 010**1** 0111 111**0**
`CMD_JOG_LEFT : // motor jog. Left`	16#0000 067E hex or binary: 0000 01**1**0 0111 111**0**

Example: Stop a running motor

Proceed as follows to stop a running motor:

1. Set the **myCommand** variable to **CMD_STOP** in the symbol browser.
2. Start the command with **myCmdExecute** = **TRUE**.
   Bit 0 in control word 1 (On/Off) is reset. The motor begins to stop.
   After reaching the actual speed < 1 rpm, the command is terminated with **myCmdDone** = **TRUE**.

Note

Bit 0, bit 8, and bit 9 in STW 1 must have the signal level **FALSE** when stopping a motor.

Assignment of **ctrlDword** (STW 1 and STW 2 of ET 200S FC) when stopping the motor:

`CMD_STOP : // stop motor`	16#0000 047E hex or binary: 0000 01**00** 0111 111**0**

8.7.6 Alarm processing

8.7.6.1 Overview

Sequence of the alarm processing

Pending error messages are processed and evaluated differently than in a SIMATIC system. Diagnostic alarms are not enabled by default. Activate the alarms for the relevant module in the hardware configuration (refer to Chapter Inserting the ET 200S frequency converter into a SIMOTION project (Page 6332)).

If you have parameterized diagnostic alarms, then you should program the alarm processing sequence according to the principle presented below.

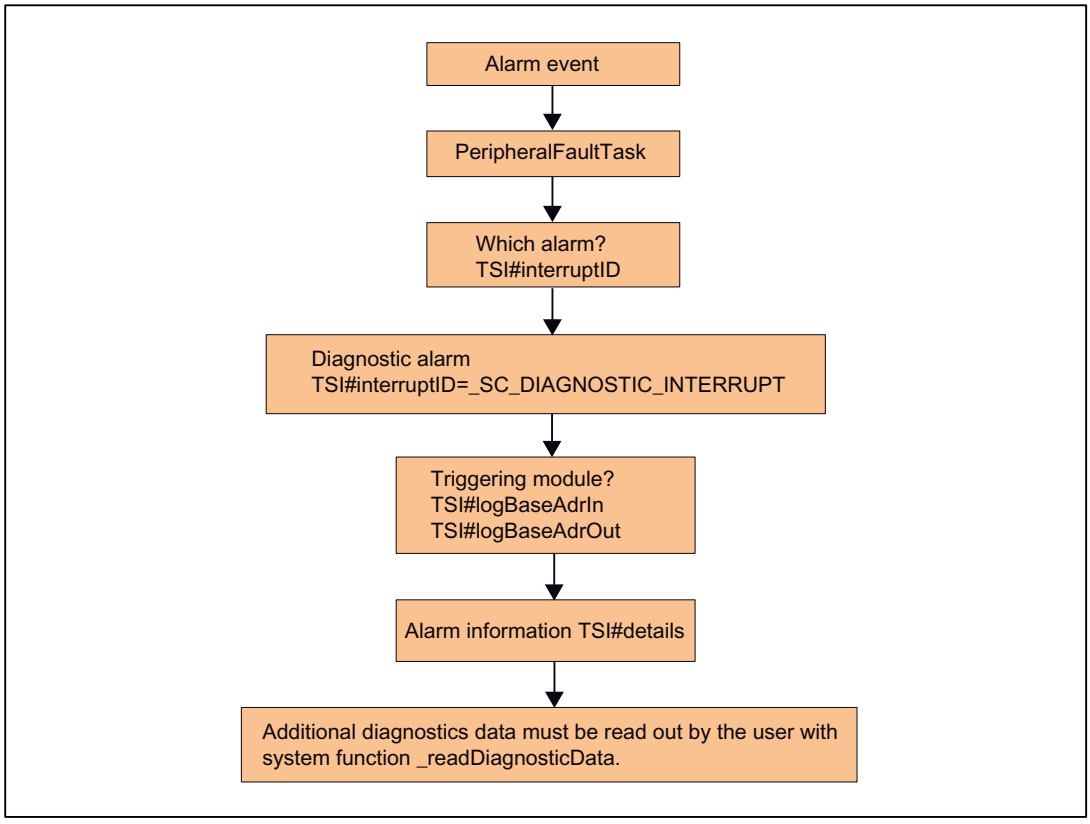


Figure 8-41 Alarm processing for the frequency converter

Additional references

A detailed description of the system functions can be found in the *SIMOTION System Function/ Variable Devices* Parameter Manual. This document is shipped with SIMOTION SCOUT in electronic form.

Detailed information on DP slave diagnostics can be found in the SIMATIC Manual titled *ET 200S Distributed I/O System*.

Alarm evaluation

Alarms sent by the I/O are evaluated in the **PeripheralFaultTask**. When the **PeripheralFaultTask** is started, the **Taskstartinfo** is made available; you must evaluate this in the user program. To do this, assign a program to the **PeripheralFaultTask** in the execution system (see sample program **Unit_E_FC_ctr**).

The **Taskstartinfo** of **PeripheralFaultTask** is comparable to the local data of OB 82 in the SIMATIC system.

Table 8-92 Meaning of the Taskstartinfo

Task	TSI		Meaning
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical start address if a hardware interrupt (PRAL) or a diagnostic interrupt (DAL) was caused by an input range on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical start address if a hardware interrupt (PRAL) or a diagnostic interrupt (DAL) was caused by an output range on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the interrupt was caused by a CPU stop or recovery of the associated DP slave, otherwise _SC_INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

8.7.6.2 Diagnostic alarms

Diagnostic alarms enable operational messages (hardware faults) of the ET 200S FC to be detected in the SIMOTION device.

Definition of a diagnostic alarm

If the user program is to respond to an internal or external error, you can set the parameters for a diagnostic alarm that will interrupt program execution in the SIMOTION device.

Events that trigger a diagnostic alarm

The criteria (events) for triggering diagnostic alarms in a SIMOTION system correspond to those in a SIMATIC system.

For a more detailed description, refer to the section titled "Diagnostics and monitoring functions" in the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).

Responses to a diagnostic alarm

If a diagnostic alarm is issued, the following occurs:

1. The group error LED (SF) lights up on the ET 200S frequency converter.
2. Diagnostic data is written to the **TSI#details** variable in the Taskstartinfo of **PeripheralFaultTask**. The program assigned to the **PeripheralFaultTask** is executed.

3. The SIMOTION device goes into STOP mode if a program has not been assigned to the **PeripheralFaultTask**.
4. The group error LED (SF) goes out as soon as the error has been remedied.

Bit assignment

The **TS#details** variable is assigned in the same way as in a SIMATIC system.

Note

For a more detailed description, refer to the section titled "Diagnostics and monitoring functions" in the SIMATIC *ET 200S FC Frequency Converter* Operating Instructions (03/2005 Edition).

8.7.7 Appendix

8.7.7.1 SIMOTION and SIMATIC names

The table below contains a comparison of SIMOTION and SIMATIC names.

Name in the SIMOTION system, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
Function block parameter	
_ET200S_FC_control	**ET200S_FC_DRIV**
execute	REQ
mode	-
periIn	-
setpoint	FREQ_SET
ctrlDword	CTRL
paraNumber	PAR_ADDRESS
subIndex	PAR_INDEX
paraValueWr	WRITE_DATA_VALUE
done	-
busy	BUSY
error	PAR_ERR
errorID	PAR_ERR_NO
driveError	ERR_NO_VALID
driveErrorId	ERR_NO
periOut	-
actValue	FREQ
statusDword	STAT
paraValueRd	READ_DATA_VALUE

8.7.7.2 List of abbreviations

Table 8-93 Abbreviations

Abbreviation	Meaning
DP	Distributed I/O
FB	Function Block
FC	**F**requency **C**onverter
IN	Input parameters
IM	Interface Module
I/O	Input/Output
LAD	Ladder diagram
LED	Light Emitting Diode
OUT	Output parameters
PG/PC	Programming Device / Personal Computer
PIB	Peripheral Input Byte
PM	Power Module
PQB	Peripheral Output Byte
PS	Power Supply
ST	Structured Text
STW	Control word
ZSW	Status word

8.8 Supplement to SIWAREX FTA Weighing Module

Preface

Contents of this Function Manual

This **document** is a component of the **SIMOTION Programming - References** documentation package.

This document is a supplement to the
"SIWAREX FTA Weighing Electronics for Automatic Scale" Manual.

This document is supplied in electronic format with SIMOTION SCOUT.

This manual tells you how you can control and assign parameters for a SIWAREX FTA weighing module from a SIMOTION device using the **_FTA_control** function block.

Function block

The **_FTA_control** function block for communication between the SIMOTION system and the SIWAREX FTA weighing module is a component of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.8.1 Fundamental safety instructions

8.8.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.8.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.8.2 Description

8.8.2.1 General

Overview

This manual describes the general differences and similarities of the SIWAREX FTA (**F**lexible **T**echnology, **A**utomatic Weighing Instrument) weighing module that result for data transfer and module diagnostics when the module is operated in a SIMOTION system as compared to a SIMATIC system.

The "Weight display with calibration capability" section also describes how the weight display with calibration capability is integrated in SIMOTION.

Note

This manual is a supplement to the *"SIWAREX FTA Electronic Weighing System for Automatic Scale"* Manual.

This document is supplied in electronic format with SIMOTION SCOUT.

Requirements

The following are required for the standard function described in this document:

- SIMOTION SCOUT, V4.1 or higher
- SIMOTION Kernel, V4.1 or higher
- SIWAREX FTA weighing module, firmware version V2.1.8 or higher

To first commission and test the SIWAREX FTA weighing module, you need the "SIWAREX FTA for SIMATIC S7 Configuration Package" CD-ROM. This CD-ROM is **not** supplied with SIMOTION SCOUT.

8.8.2.2 Product description

SIWAREX FTA is a versatile and flexible weighing module.

The primary task of the SIWAREX FTA weighing module consists of high-precision measurement of actual weight with up to three measurement ranges and precise control of weighing operations.

The SIWAREX FTA weighing module can be assigned parameters for the following operating modes:

- **N**on **A**utomatic **W**eighing **I**nstrument = NAWI
- **A**utomatic **W**eighing **I**nstrument = AWI
  - For gravimetric filling
  - For catch-weighing
  - For totalizing

Function block

The **_FTA_control** function block is provided for communication between the SIMOTION device and the SIWAREX FTA weighing module. This function block is described in this manual.

Functionality

The **_FTA_control** function block reads/writes cyclic data from/to the SIWAREX FTA weighing module. Data are also transferred acyclically from/to the weighing module upon request.

The functionality of the **_FTA_control** function block and the SIWAREX FTA weighing module is the same as when used in a SIMATIC S7 automation system. However, the execution of data transmission and the handling of the FB have been adapted to the given general SIMOTION conditions.

Possible applications

In addition to the possible applications described in the SIMATIC manuals, you can also use the SIWAREX FTA weighing module in a SIMOTION system. The SIWAREX FTA weighing module can be used for centralized applications (only on the SIMOTION C2xx) or distributed applications (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx).

The following figure illustrates the connection of a distributed ET 200M I/O device with IM 153-1 and SIWAREX FTA to a SIMOTION device (e.g., C2xx).

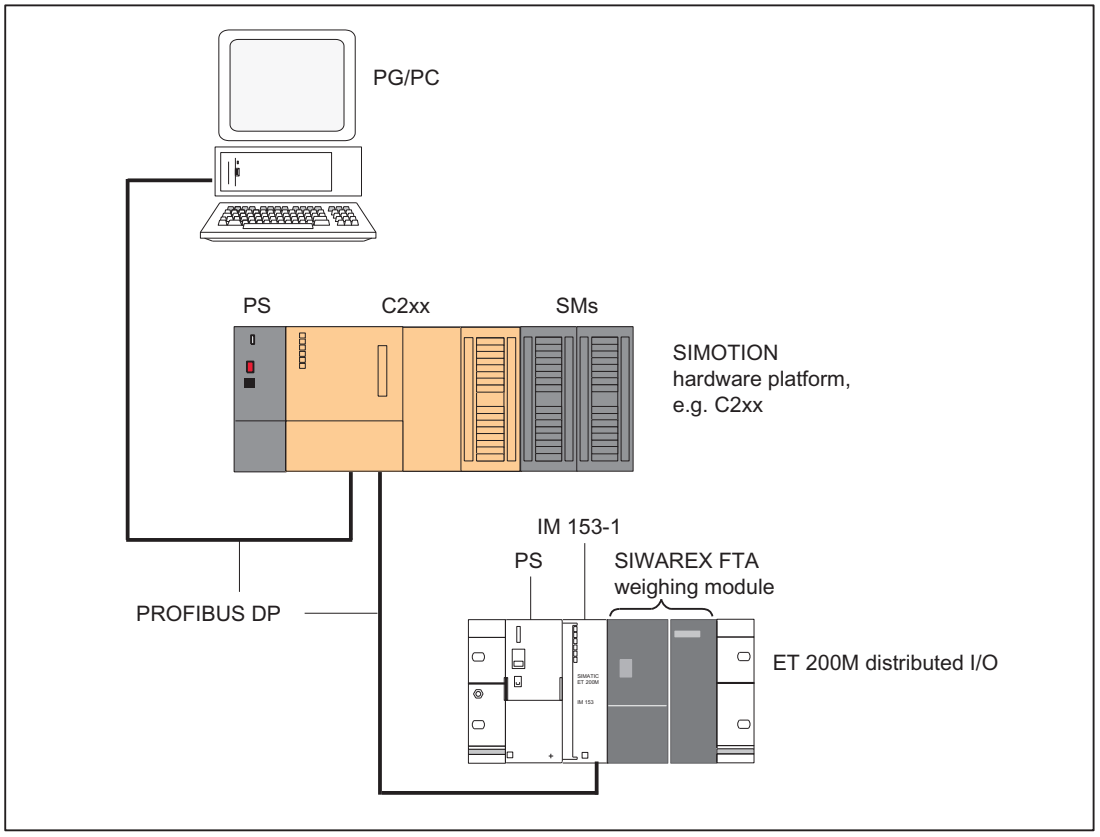


Figure 8-42 Connection of weighing module with an ET 200M to a SIMOTION C2xx device (example of distributed application)

8.8.2.3 Setup and connection

Overview

You must perform the following steps to commission the SIWAREX FTA weighing module and control it from the SIMOTION system.

Distributed application (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx)

1. Assembly and wiring of ET 200M distributed I/O device with power supply (PS), interface module (IM), and weighing module.
2. Establish the PROFIBUS connection between the ET 200 M and the SIMOTION device.
3. Set the PROFIBUS DP node address on the IM.
4. Switch on the terminating resistor at the first and last bus node.
5. Insert the weigh beam in the SIMOTION project, see section Integrating the weighing module in a SIMOTION project (Page 6359).
6. Assign parameters to the weighing module using the SIWATOOL FTA program.
   Consult the *SIWAREX FTA Electronic Weighing System for Automatic Scale* Manual to learn how to install the SIWATOOL FTA parameterization tool and assign parameters to the weighing module.
7. Integrate the function block in the SIMOTION project, see section Integrating the function block in the user project (Page 6361).

Centralized application (SIMOTION C2xx only)

1. To learn how to configure the mechanical design and prepare and install the SIMOTION components, refer to *SIMOTION C* Operating Instructions and *SIMATIC Automation System S7-300, Software Installation* Manual .
   These documents are shipped with SIMOTION SCOUT in electronic form.
2. For the remaining steps, refer to steps 5 to 7 of the distributed application.

8.8.2.4 Integrating the weighing module in a SIMOTION project

Requirements

You have run the Hardware Support Package for installing the weighing module in the hardware catalog of SIMATIC Manager.

Note

This Hardware Support Package is included on the "SIWAREX FTA configuration package for SIMATIC S7" CD-ROM, which you need to configure the weighing module. This CD-ROM is **not** supplied with the SIWAREX FTA and must be ordered separately.

The following requirements must be met when networking with PROFIBUS:

1. You have created a project in SIMOTION SCOUT and inserted a rack with a SIMOTION device in the hardware configuration.
2. You have configured a PROFIBUS subnet (for distributed use only).

Note

For how to create a project and configure a PROFIBUS subnet, refer to the online help of *SIMOTION SCOUT*.

The following requirements must be met when with PROFINET:

1. You have created a project in SIMOTION SCOUT and inserted and configured a rack with a SIMOTION device capable of PROFINET in the hardware configuration.
2. You have configured a PROFINET IO System (for distributed use only).

Note

For how to create a project and configure a PROFINET IO system, refer to the online help of *SIMOTION SCOUT*.

Inserting the weighing module (distributed application)

The description below is an example of networking via PROFIBUS.

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, use the **View > Catalog** menu command to open the **Hardware Catalog**.
4. In the hardware catalog, open the **PROFIBUS DP** folder and the **ET 200M** subfolder and select, for example, the **IM 153-1 interface module** (Article No.: 6ES7 153-1AA03-0XB0 or a successor module).
5. Use a drag-and-drop operation to move the IM 153-1 I/O device to the PROFIBUS subnet of your project.
   The **Properties - PROFIBUS IM 153-1 Interface** dialog box opens. In this dialog box, select the address you set on the IM 153-1 (see *Distributed I/O Device ET 200M* Manual) and confirm your selection with **OK**.
   The selected IM 153-1 I/O device is inserted in the project.
6. The inserted I/O device must now be fitted with the module from your project. To do so, open the **Weighing Modules** subfolder under the selected I/O device in the hardware catalog, and select **SIWAREX FTA**.

Note

By default, the diagnostic alarms and process alarms are not enabled. Enable the alarms for the weighing module in the **Properties** dialog box.

7. **Save** and **compile** your project.

Inserting the weighing module (centralized application)

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, use the **View > Catalog** menu command to open the **Hardware Catalog**.
4. In the hardware catalog, open the **SIMATIC 300 > FM 300** folder followed by the **Weighing Modules** subfolder and select **SIWAREX FTA**.
5. Use a drag-and-drop operation to place the SIWAREX FTA weighing module into the SIMOTION device's rack.

   Note

   By default, the diagnostic alarms and process alarms are not enabled. Enable the alarms for the weighing module in the **Properties** dialog box.

6. **Save** and **compile** your project.

8.8.3 Programming

8.8.3.1 Integrating the function block in the user project

Creating the FB instance in the user project

The **_FTA_control** function block is a component of the command library of the SIMOTION SCOUT engineering system. In order to work with the block, you must create an instance of the function block and a **struct_FTA_scaleData** type variable in the user project.

Example:

```
VAR_GLOBAL
  myFTA_control  : _FTA_control;           // create instance of FB _FTA_control
  myScaleData    : struct_FTA_scaleData;   // structure with all scale data
END_VAR
```

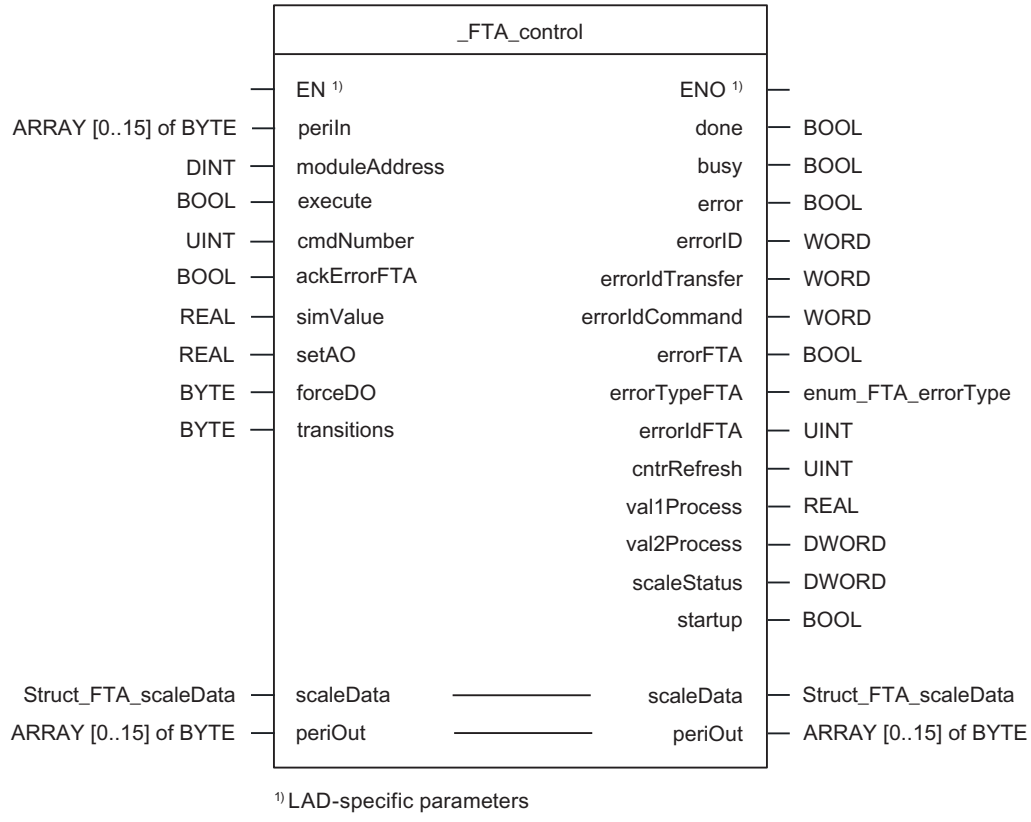
Note

If you want to control more than one weighing module, you must create

- a new variable for the data structure and
- an instance of the function block with a new name

in the user program for each weighing module used.

Call (LAD representation)



Application example

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

8.8.3.2 Addressing the SIWAREX FTA weighing module

Overview

Communication between the SIMOTION device and the SIWAREX FTA weighing module takes place by means of direct I/O access and data record transfer.

Creating I/O variables

For cyclic data transfer, you must create one I/O variable for the I/O inputs and one I/O variable for the I/O outputs in the symbol browser. This I/O variable must have the address of the weighing module of data type BYTE and a field length of 16.

When creating the I/O variable, enter the configured address from the hardware configuration in the "I/O address" parameter (see example below).

Example

Determining addresses from **HW Config**:

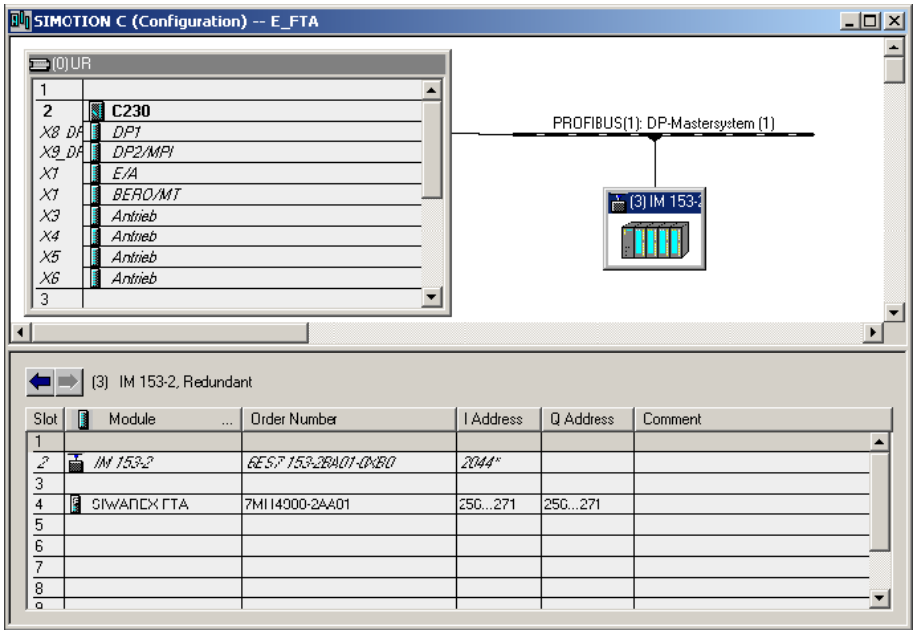


Figure 8-43 Addresses in HW Config

Creating the I/O variables in the symbol browser:

	Name	I/O address	Read only	Data type	Field length
1	myperiin	PIB 256		Array	16
2	myperiout	PQB 256		Array	16

Figure 8-44 Addressing with I/O variables

Parameter transfer

The I/O variable for the I/O inputs is transferred to the **periIn** input parameter.

The prepared data for the I/O outputs are supplied by the function block at the **periOut** output parameter. These data are assigned to the I/O variable for the I/O outputs.

Note

For additional information, refer to:

- SIMOTION SCOUT online help
- Programming Manuals for the relevant programming language, such as:
  - SIMOTION ST, Structured Text Programming Manual
  - SIMOTION MCC, Motion Control Chart Programming Manual
  - SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

8.8.4 Parameter assignment

8.8.4.1 _FTA_control function block

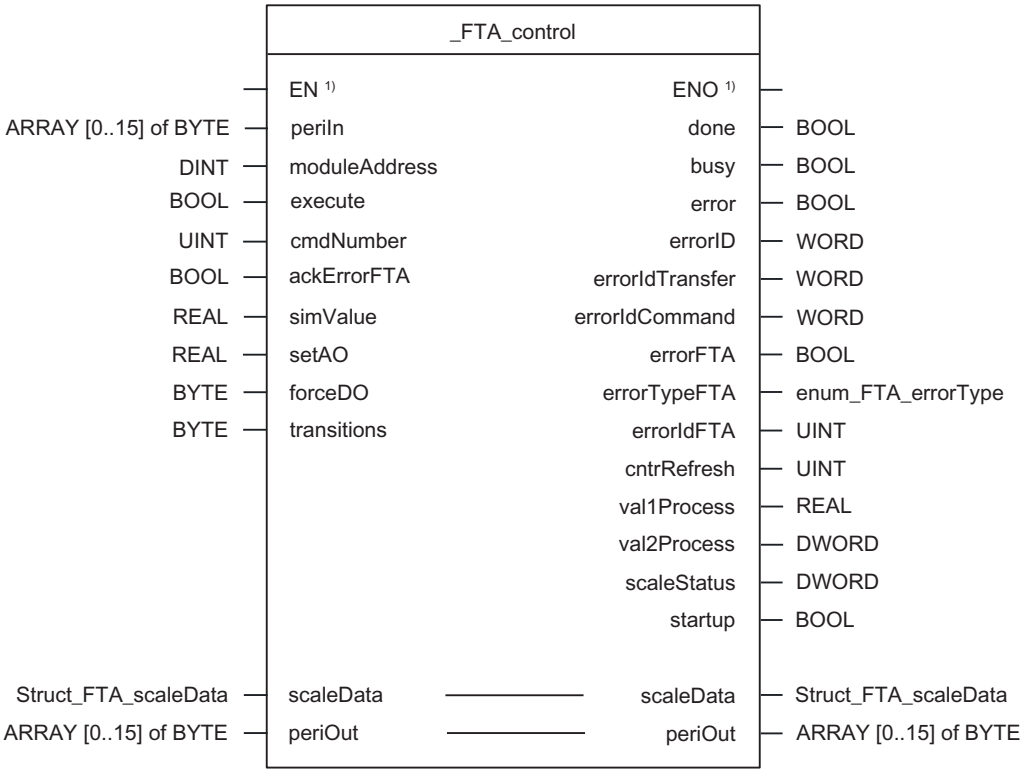
Task

The **_FTA_control** function block (FB) enables you to control the SIWAREX FTA weighing module in the SIMOTION automation system.

The **_FTA_control** FB writes process data cyclically to the weighing module and reads the status of the weighing module cyclically. The status is made available at the output parameters of the FB.

On request, the FB writes scale data to the weighing module and reads scale data and process status from the weighing module. The scale data and process status are stored by the function block in the **Struct_FTA_scaleData** data structure and read by the function block from this data structure.

Call (LAD representation)



Parameter description

The table below contains all parameters of the function block **_FTA_control**.

Table 8-94 Parameters of the _FTA_control function block

Name	Type[1)]	Data type	Default	Definition
periIn	IN	ARRAY [0 to 15] of BYTE	16 (16#00)	Transfers I/O inputs to **_FTA_control** function block
moduleAddress	IN	DINT	0	Module address from **HW Config**
execute	IN	BOOL	FALSE	Rising edge: the command entered in **cmdNumber** is executed.
cmdNumber	IN	UINT	0	Provides the command number The command is started using the **execute** parameter.
ackErrorFTA	IN	BOOL	FALSE	**TRUE** = acknowledgment of error of SIWAREX FTA weighing module waiting to be dealt with
simValue	IN	REAL	0.0	Value for weight simulation when simulation is enabled; selection is made in DR7
setAO	IN	REAL	0.0	Value specification for analog output of weighing module with master control of higher-level control; selection is made in DR7

Name	Type[1)]	Data type	Default	Definition
forceDO	IN	BYTE	16#00	Specifies digital outputs of weighing module with parameterized forced control; selection is made in DR7
transitions	IN	BYTE	16#00	Disables and enables weighing steps The bit numbers in the **transitions** byte correspond to the numbers of the weighing steps. Bit = **FALSE**: weighing step is executed Bit = **TRUE**: weighing step is not executed, and the system waits for the bit to be reset
scaleData	IN/OUT	**Struct_FTA_scaleData**		Data structure for the parameters of the SIWAREX FTA weighing module
periOut	IN/OUT	ARRAY [0 to 15] of BYTE	16 (16#00)	Prepared FB data for I/O outputs of SIWAREX FTA weighing module
done	OUT	BOOL	FALSE	**TRUE** = command execution completed without errors
busy	OUT	BOOL	FALSE	**TRUE** = command in progress
error	OUT	BOOL	FALSE	**TRUE** = **_FTA_control** FB has detected an error; error specified in **errorID**.
errorID	OUT	WORD	16#0000	Error specification For **error** = **TRUE**, the **errorID** output parameter contains the error information, see "Error messages"
errorIdTransfer	OUT	WORD	16#0000	Error specification for faults during data transfer with system functions **_writeRecord** / **_readRecord** [2)]
errorIdCommand	OUT	WORD	16#0000	Error number of data or operation error that has occurred. For a description of errors, see the SIWAREX FTA Manual *Electronic Weighing System for Automatic Scale*
errorFTA	OUT	BOOL	FALSE	**TRUE** = new error message in message buffer of SIWAREX FTA weighing module, error specification in **errorTypeFTA**
errorTypeFTA	OUT	Enum_FTA_ errorType (OP_ERROR, TECHNO_ERROR, DATA_ERROR)	0	Error specification When **errorFTA** = **TRUE**, parameter **errorTypeFTA** contains the error information type: • OP_ERROR: Operating message (error) • TECHNO_ERROR: technological error • DATA_ERROR: data or operation error
errorIdFTA	OUT	UINT	0	Message number, for a description of the signal, see the SIWAREX FTA Manual *Electronic Weighing System for Automatic Scale*

Name	Type[1)]	Data type	Default	Definition
cntrRefresh	OUT	UINT	0	Outputs the update number for the values in output parameters **val1Process** / **val2Process**. The SIWAREX FTA weighing module updates the values for the **val1Process** / **val2Process** parameters internally in a 10 ms cycle. Each time the values are updated, the weighing module increments the value for the **cntrRefresh** parameter. The **_FTA_control** function block reads the values in cycles from the weighing module in the time scale of the calling task (e.g. BackgroundTask) and updates the values in the output parameters **cntrRefresh**, **val1Process** / **val2Process**. The value in **cntrRefresh** can be used like a time stamp in the SIMOTION device.
val1Process	OUT	REAL	0.0	Outputs the selected process parameter (e.g. net weight); selection is made in DR7
val2Process	OUT	DWORD	16#00000000	Outputs the selected process parameter (e.g. AWI status); selection is made in DR7
scaleStatus	OUT	DWORD	16#00000000	Outputs the status of the non-automatic scale
startup	OUT	BOOL	FALSE	**TRUE** = restart of the SIWAREX FTA weighing module Communication and command execution are not possible

1) Parameter types: IN = input parameter, IN/OUT = in/out parameter, OUT = output parameter

2) Description of system functions **_writeRecord** and **_readRecord** see Parameter Manual *SIMOTION system functions/ variables for devices*. This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

Function description

When the **_FTA_control** FB is called, the cyclic data of the FB are read and written.

Command execution is started on a rising edge on input parameter **execute**. The command is selected by specifying the command number in input parameter **cmdNumber**. For the command groups with numbers 203 - 699, one or more data records (DR) are transferred to or read by the SIWAREX FTA weighing module. When a command is started, the output parameters **busy** = **TRUE** and **done** = **FALSE** are set. Output parameters **error**, **errorID**, **errorIdCommand**, and **errrorIdTransfer** are reset. When the command has been executed without errors, the following parameters are displayed: **busy** = **FALSE**, **done** = **TRUE**, and **error** = **FALSE**.

If an error occurs during command execution, the following output parameters are set **busy** = **FALSE**, **done** = **FALSE**, and **error** = **TRUE**. The error is specified in output parameters **errorID**, **errorIdTransfer**, and **errorIdCommand**.

A new command can only be started once execution of the commands is complete (**busy** = **FALSE**). If a new command is started while commands are being executed (**busy** = **TRUE**), the new command is ignored.

Note

You can find an overview of the command groups and commands in the appendix.

If the SIWAREX FTA weighing module generates a new error message, the following output parameter is set: **errorFTA** = **TRUE**. The error message is specified in output parameters **errorTypeFTA** and **errorIdFTA**. After the error message has been evaluated, it must be acknowledged with a rising edge on input parameter **ackErrorFTA** . The output parameters **errorFTA**, **errorTypeFTA** and **errorIdFTA** are reset by the **_FTA_control** function block in the process.

Graphic overview of the functionality

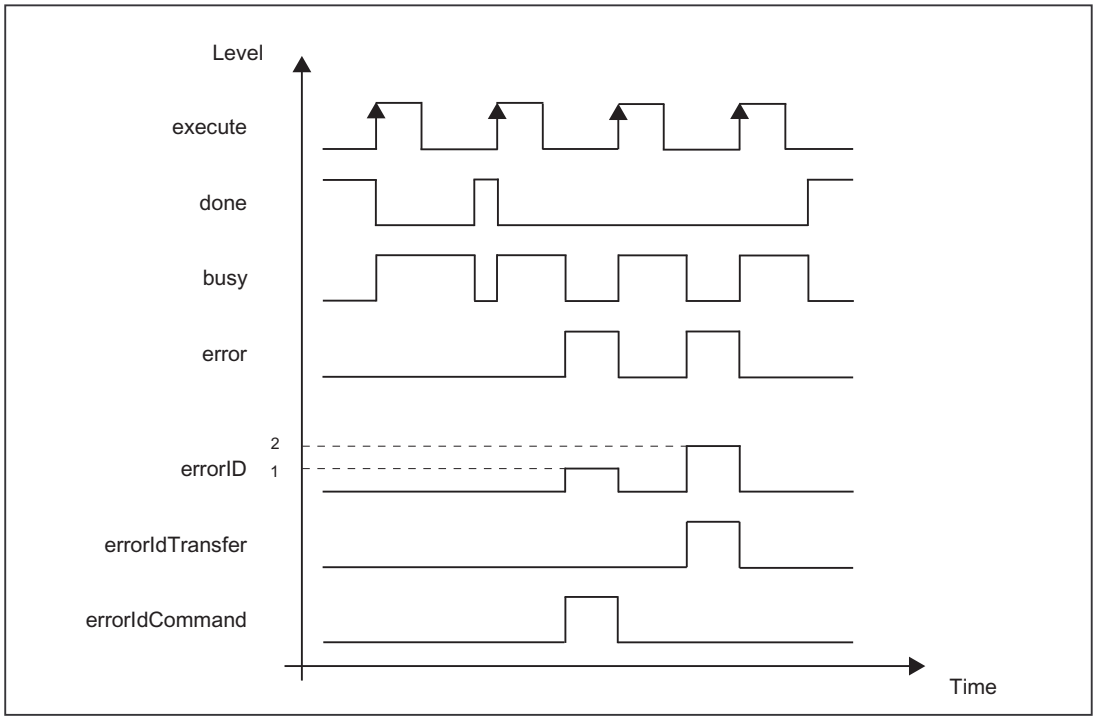


Figure 8-45 Command execution signal sequence diagram

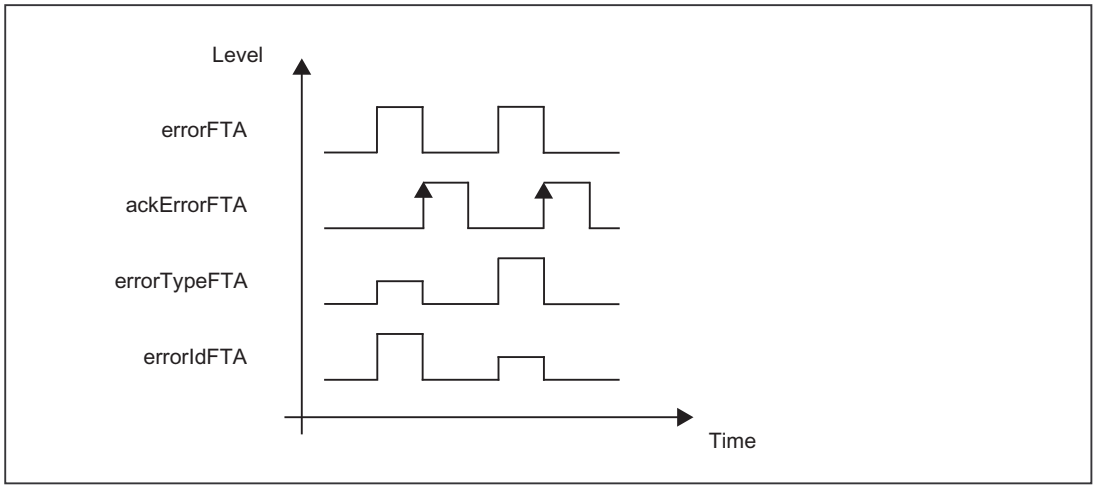


Figure 8-46 Messages signal sequence diagram

Task integration (call)

The **_FTA_control** function block is designed to be called in a cyclic task and must be called in this task during each task cycle. It may take several cycles for a task to be executed. The user decides which cyclic task of the **_FTA_control** function block the call is made in. No limitations are placed on the functionality of the **_FTA_control** function block. The call can be performed in all cyclic tasks. A fixed time scale (e.g. **IPOSynchronousTask**) is not required for the processing.

Error messages

When an error occurs, an error specification is displayed in output parameter **errorID**. If an error has not occurred, **errorID** has the value "16#0000". The **errorID** output parameter remains present until the error is corrected.

The **errorID** output parameter is reset with a rising edge on input parameter **execute**.

Table 8-95 Error numbers in output parameter errorID

Error no. errorID	Definition
Bit 0	Command terminated with error; data or operation error has occurred For error specification, see output parameter **errorIdCommand**
Bit 1	Error has occurred during call of system functions **_writeRecord** / **_readRecord** [1)] For error specification, see output parameter **errorIdTransfer**
Bit 2	Error during data record/command interpretation The specified data record or command number is incorrect.
Bit 3	Life bit error, SIWAREX FTA is not responding
Bit 4	I/O data could not be read out in this cycle
Bit 5	Active command aborted during restart of SIWAREX FTA weighing module
Bit 6	Reserve (use not permitted)
Bit 7	Reserve (use not permitted)
Bit 8	Reserve (use not permitted)
Bit 9	Reserve (use not permitted)
Bit 10	Reserve (use not permitted)
Bit 11	Reserve (use not permitted)
Bit 12	Reserve (use not permitted)
Bit 13	Reserve (use not permitted)
Bit 14	Reserve (use not permitted)
Bit 15	Reserve (use not permitted)

1) Description of system functions **_writeRecord** and **_readRecord** see Parameter Manual *SIMOTION system functions/variables for devices*. This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

When communication errors and command-execution faults occur, an error number is output in output parameters **errorIdTransfer** and **errorIdCommand**. Refer to the *SIWAREX FTA Electronic Weighing System for Automatic Scale* Manual for the meaning of the error numbers.

8.8.4.2 Data structures

Overview of data structures

Overview

The data structure of type **Struct_FTA_scaleData** contains all relevant data for operating the SIWAREX FTA weighing module. This data structure is used by the **_FTA_control** function block. The elements of the data structure are accessed by means of a user-defined variable of data type **Struct_FTA_scaleData** (see example below):

```
VAR
    myScaleData : Struct_FTA_scaleData;   // data structure with all scale data
END_VAR

PROGRAM
//...
    myScaleData.DR3.filtSequence := 1;   // write one element in DR3
//...
END_PROGRAM
```

Data records of the data structure

The table below contains all data records (DR) of the **Struct_FTA_scaleData** data structure which can be read and written by the SIMOTION device and the winccOcxWrite variable for the data transfer of the weight display with calibration capability.

Table 8-96 Data structure Struct_FTA_scaleData

Name	Data type	Definition
DR3	**Struct_FTA_DR3**	Data structure for DR3: Calibration parameter
DR4	**Struct_FTA_DR4**	Data structure for DR4: Basic parameter
DR7	**Struct_FTA_DR7**	Data structure for DR7: Interfaces
DR8	**Struct_FTA_DR8**	Data structure for DR8: Date/time
DR9	**Struct_FTA_DR9**	Data structure for DR9: information about the weighing module
DR15	**Struct_FTA_DR15**	Data structure for DR15: Tare input
DR16	**Struct_FTA_DR16**	Data structure for DR16: Weight simulation input
DR17	**Struct_FTA_DR17**	Data structure for DR17: Analog output control
DR18	**Struct_FTA_DR18**	Data structure for DR18: Remote display control
DR20	**Struct_FTA_DR20**	Data structure for DR20: Setpoint weight
DR21	**Struct_FTA_DR21**	Data structure for DR21: Load weight
DR22	**Struct_FTA_DR22**	Data structure for DR22: Weighing parameter 1
DR23	**Struct_FTA_DR23**	Data structure for DR23: Weighing parameter 2
DR26	**Struct_FTA_DR26**	Data structure for DR26: Internal process values 1

Name	Data type	Definition
DR30	**Struct_FTA_DR30**	Data structure for DR30: Process values 1
DR31	**Struct_FTA_DR31**	Data structure for DR31: Process values 2
DR32	**Struct_FTA_DR32**	Data structure for DR32: Statistical data
DR34	**Struct_FTA_DR34**	Data structure for DR34: ASCII weight value
DR35	**Struct_FTA_DR35**	Data structure for DR35: Value for display with calibration capability is encrypted
DR39	**Struct_FTA_DR39**	Data structure for DR39: Version ID of "SecureOCX" weight display with calibration capability
DR44	**Struct_FTA_DR44**	Data structure for DR44: Last log
DR45	**Struct_FTA_DR45**	Data structure for DR45: Variable for log
DR46	**Struct_FTA_DR46**	Data structure for DR46: Parameter for readout of logs stored on the micro memory card of the weighing module
DR47	**Struct_FTA_DR47**	Data structure for DR47: Log requested via DR46
DR123	**Struct_FTA_DR123**	Data structure for DR123: DR123 can be read out with the SIWATOOL FTA to determine which data are stored on the micro memory card. Based on this information, the user can read out specific recordings and logs.
winccOcxWrite	**DWORD**	Variable for the data transfer between the "SecureOCX" weight display with calibration capability and the SIWAREX FTA weighing module, see section Weight display with calibration capability (Page 6400).

The data records listed in the table are defined in greater detail below.

Struct_FTA_DR3

Description

Data structure **Struct_FTA_DR3** contains the calibration parameter of the SIWAREX FTA weighing module.

To learn how to perform the calibration, refer to the *"SIWAREX FTA Weighing Electronics for Automatic Scale"* Manual.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR3**.

Table 8-97 Struct_FTA_DR3

Name	Data type	Default	Definition
Calibration			
calibDigits0	DINT	1.398.101	Calibration digits for the zero point
calibDigits1	DINT	15.379.113	Calibration digits for calibration weight 1
calibDigits2	DINT	0	Calibration digits for calibration weight 2
calibDigits3	DINT	0	Calibration digits for calibration weight 3
calibDigits4	DINT	0	Calibration digits for calibration weight 4

Name	Data type	Default	Definition
calibWeight1	REAL	100.0	Calibration weight 1
calibWeight2	REAL	0.0	Calibration weight 2
calibWeight3	REAL	0.0	Calibration weight 3
calibWeight4	REAL	0.0	Calibration weight 4
sigRange	BYTE	16#02	Characteristic value range 16#01 = characteristic value to 1 mV/V 16#02 = characteristic value to 2 mV/V 16#04 = characteristic value to 4 mV/V
Filter			
filtSequence	BYTE	16#00	Signal filter sequence The 8 bits in this byte have the following meaning: Bit 0 = 0: Averaging filter before digital filter Bit 0 = 1: Digital filter before averaging filter Bit 1-7: Not assigned
filtType	BYTE	16#00	Low-pass filter type 16#00 = critically damped 16#01 = Bessel filter 16#02 = Butterworth filter Other definitions not permitted.
filtCutOffFreq	USINT	16#04	Filter limit frequency (Hz) Specifies the response time of the scale to a measured value change 16#00 = no filter 16#01 = 20 Hz 16#02 = 10 Hz 16#03 = 5 Hz 16#04 = 2 Hz 16#05 = 1 Hz 16#06 = 0.5 Hz 16#07 = 0.2 Hz 16#08 = 0.1 Hz 16#09 = 0.05 Hz Other definitions not permitted.
filtDepth	INT	10	Depth of averaging filter Number of values for averaging [0 to 250] x 2.5 ms 0 = averaging filter deactivated
Calibration parameter 1			
scaleId	STRING[10]	'SIWAREX xx'	Scale name
numRanges	USINT	1	Number of scale ranges 1 = 1 range 2 = 2 ranges 3 = 3 ranges Other definitions not permitted.
scaleType	BOOL	FALSE	Scale type **FALSE** = multi-range scale **TRUE** = multi-resolution scale
setZeroPwrOn	BOOL	FALSE	**TRUE** = zero setting activated upon weighing module startup (power ON)

Name	Data type	Default	Definition
setZeroTare	BOOL	FALSE	**TRUE** = zero setting activated upon weighing module startup (power ON) and tare <> 0
setZeroAutomatic	BOOL	FALSE	**TRUE** = automatic zero correction switched on
setTareType	BOOL	FALSE	Tare selection **FALSE** = use subtractive tare equipment **TRUE** = use additive tare equipment
Calibration parameter 2			
minWeightRange1	REAL	1	Minimum weight for weighing range 1
maxWeightRange1	REAL	100.0	Maximum weight for weighing range 1
incRange1	REAL	0.02	Numerical increment for weighing range 1 Numerical increment for weighing range 1 ($1*10^k$, $2*10^k$, $5*10^k$, $-4 \leq k \leq 1$)
minWeightRange2	REAL	0.0	Minimum weight for weighing range 2
maxWeightRange2	REAL	0.0	Maximum weight for weighing range 2
incRange2	REAL	0.0	Numerical increment for weighing range 2 Numerical increment for weighing range 2 ($1*10^k$, $2*10^k$, $5*10^k$, $-4 \leq k \leq 1$)
minWeightRange3	REAL	0.0	Minimum weight for weighing range 3
maxWeightRange3	REAL	0.0	Maximum weight for weighing range 3
incRange3	REAL	0.0	Numerical increment for weighing range 3 Numerical increment for weighing range 3 ($1*10^k$, $2*10^k$, $5*10^k$, $-4 \leq k \leq 1$)
Calibration parameter 3			
standStillTime1	TIME	T#1s	Downtime If the fluctuation range of the weight during the downtime is less than the fluctuation range defined in the **standStillWeight1**parameter, the scale standstill signal is set to **TRUE**.
standStillWeight1	REAL	0.02	Fluctuation range of the weight for detection/generation of the scale standstill signal
timeOutStandStill1	TIME	T#2s	Maximum wait time for scale standstill If a scale standstill is not detected within this time, a technology message (output parameter **errorFTA**, **errorTypeFTA**, **errorIdFTA**) is output.
maxPosWeightPwrOn	BYTE	16#0A	Maximum positive weight in [% of WBmax] for zero setting after Power On, 16#0A ≙ 10 %
minNegWeightPwrOn	BYTE	16#0A	Maximum negative weight in [% of WBmax] for zero setting after Power On, 16#0A ≙ 10 %
maxPosWeightZero	BYTE	16#01	Maximum positive weight in [% of WBmax] for zero setting, 16#01 ≙ 1 %
minNegWeightZero	BYTE	16#03	Maximum negative weight in [% of WBmax] for zero setting, 16#03 ≙ 3 %
maxTare	BYTE	16#64	Maximum tare load in [% of WBmax] with respect to the maximum weighing range, 16#64 ≙ 100 %
setLCType	BYTE	16#00	Selection of load cell type

Name	Data type	Default	Definition
timeOutDigLC	INT	80	Monitoring time in [ms] for digital load cell If a weight message frame is not received within the monitoring time, the SIWAREX FTA weighing module generates an operating error.
setRestriction	STRING[4]	'----'	The entry 'OIML' activates restrictions/regulations for scales that require official calibration
weightUnit	STRING[4]	'kg--'	Unit of mass (for example, t, kg, g)
Calibration parameter 4			
standStillTime2	TIME	T#1s	Downtime If the fluctuation range of the weight during the downtime is less than the fluctuation range defined in the **standStillWeight2**parameter, the scale standstill signal is set to **TRUE**.
standStillWeight2	REAL	0.02	Fluctuation range of the weight for detection/generation of the scale standstill signal
timeOutStandStill2	TIME	T#500ms	Maximum wait time for scale standstill If a scale standstill is not detected within this time, a technology message (output parameter **errorFTA**, **errorTypeFTA**, **errorIdFTA**) is output.
standStillWeight3	REAL	0.02	Fluctuation range of the weight for detection/generation of the scale standstill signal
standStillTime3	TIME	T#1s	Downtime If the fluctuation range of the weight during the downtime is less than the fluctuation range defined in the **standStillWeight3**parameter, the scale standstill signal is set to **TRUE**.
timeOutStandStill3	TIME	T#500ms	Maximum wait time for scale standstill If a scale standstill is not detected within this time, a technology message (output parameter **errorFTA**, **errorTypeFTA**, **errorIdFTA**) is output.
minDosingValTot	REAL	100	Smallest set weight for weight totalizing
incTotalizing	REAL	0.1	Scale interval for weight totalizing
xxxReserve0	Real	0.0	Reserve (use not permitted)
xxxReserve1	Byte	16#00	Reserve (use not permitted)
xxxReserve2	Byte	16#00	Reserve (use not permitted)

Struct_FTA_DR4

Overview

Data structure **Struct_FTA_DR4** contains the basic parameters of the SIWAREX FTA weighing module. These basic parameters determine additional characteristics that are responsible for scale performance.

Parameter description

The table below contains all parameters of the data structure **Struct_FTA_DR4**.

Table 8-98 Struct_FTA_DR4

Name	Data type	Default	Definition
Operating mode / logs			
scaleMode	BYTE	16#03	Scale mode, selection of weighing program 16#00 = weighing (additive) 16#01 = weighing (subtractive) 16#02 = dosing (additive), subsequent emptying 16#03 = dosing (additive), without subsequent emptying 16#04 = dosing (subtractive), without subsequent emptying 16#05 = check 16#06 = dosing (tare reweighing), cumulating 16#07 = dosing (additive)/(big bag) 16#08 = dosing (subtractive)/(big bag) 16#09 = dosing (subtractive), subsequent emptying Other definitions not permitted.
xxxReserve0	BYTE	16#00	Reserve (use not permitted)
xxxReserve1	WORD	16#0000	Reserve (use not permitted)
timeOutLogOutput	TIME	T#2s	Monitoring time in [ms] for output of log data to a connected printer.
selectLogOutput	BYTE	16#01	Selection of device for log output Bit 0 = 0: Log output to RS232 interface Bit 0 = 1: Log output to micro memory card (MMC)
xxxReserve2	BYTE	16#00	Reserve (use not permitted)
Limit values			
baseWeightLimit1	BOOL	FALSE	Base weight for limit value 1 **TRUE** = basis for limit value 1 is gross weight of scale **FALSE** = basis for limit value 1 is net weight of scale
baseWeightLimit2	BOOL	FALSE	Base weight for limit value 2 **TRUE** = basis for limit value 2 is gross weight of scale **FALSE** = basis for limit value 2 is net weight of scale
xxxReserve3	BOOL	FALSE	Reserve (use not permitted)
baseEmptyRange	BOOL	FALSE	Base weight for empty range monitoring **FALSE** = basis for empty range is gross weight of scale **TRUE** = basis for empty range is net weight of scale
xxxReserve4	BYTE	16#00	Reserve (use not permitted)
emptyRange	REAL	1.0	Empty range
limit1On	REAL	1.0	Start-up weight limit value 1
limit1Off	REAL	1.1	Shut-down weight limit value 1
limit2On	REAL	50.0	Start-up weight limit value 2
limit2Off	REAL	49.0	Shut-down weight limit value 2
limit3On	REAL	99.0	Start-up weight limit value 3
limit3Off	REAL	98.0	Shut-down weight limit value 3
Flow			

Name	Data type	Default	Definition
minFlowLimit1	REAL	0.0	Limit value 1 for minimum flow rate 0.0 = no limit value in effect >0.0 = minimum flow rate in [1/s]
minFlowLimit2	REAL	0.0	Limit value 2 for minimum flow rate 0.0 = no limit value in effect >0.0 = minimum flow rate in [1/s]
filtDepthMinFlow	BYTE	16#05	Filter depth of mean value filter for flow rate calculation [0..255] * 10 ms, 16#05 ≙ 5 * 10 ms = 50 ms
xxxReserve5	BYTE	16#00	Reserve (use not permitted)

Struct_FTA_DR7

Overview

Data structure **Struct_FTA_DR7** contains the parameters that determine the behavior of the SIWAREX FTA weighing module at the interfaces.

Parameter description

The table below contains all parameters of the data structure **Struct_FTA_DR7**.

Table 8-99 Struct_FTA_DR7

Name	Data type	Default	Definition
Interface to SIMOTION device			
xxxReserve0	BYTE	16#00	Reserve (use not permitted)
srcWeightSim	BYTE	16#00	Weight simulation source 16#00 = weight simulation is inactive 16#01 = weight simulation value from SIMOTION (defined externally via DR 16) 16#02 = weight simulation value from RS232
deciDigitsProcVal	BYTE	16#03	Number of decimal places/decade used for rounding the decimal places of process values (REAL weight values) 16#00 = rounding to 0 decimal places 16#01 = rounding to 1 decimal place 16#02 = rounding to 2 decimal places 16#03 = rounding to 3 decimal places 16#04 = rounding to 4 decimal places 16#05 = rounding to 5 decimal places 16#06 = rounding to 6 decimal places
xxxReserve1	BYTE	16#00	Reserve (use not permitted)
enableForceDO	BOOL	FALSE	**TRUE** = forced control of outputs possible in service mode
indexProcessValue1	BYTE	16#02	Selection of process value 1 from list at output parameter **val1Process** of function block **_FTA_control** for fast output to the SIMOTION device, 16#02 ≙ "Net weight (process value)" see AppendixSelection list for process values (Page 6412)

Name	Data type	Default	Definition
indexProcessValue2	BYTE	16#1E	Selection of process value 2 from list at output parameter **val2Process** of function block **_FTA_control** for fast output to the SIMOTION device, 16#1E ≙ "AWI status" see AppendixSelection list for process values (Page 6412)
xxxReserve2	BYTE	16#00	Reserve (use not permitted)
SIMOTION alarms			
defProcessAlarm0	WORD	16#0000	Definition of process alarm 0 16#0000 = no triggering of process alarms Range of values 16#0001 - 16#00FF: Number of technology error Range of values 16#0100 - 16#013F: 16#0100...16#011F ≙ bit no. NAWI status bit coming 16#0120...16#013F ≙ bit no. AWI status bit coming Range of values 16#0200 - 16#023F: 16#0200...16#021F ≙ bit no. NAWI status bit going 16#0220...16#023F ≙ bit no. AWI status bit going
defProcessAlarm1	WORD	16#0000	Definition of process alarm 1 (see **defProcessAlarm0**)
defProcessAlarm2	WORD	16#0000	Definition of process alarm 2 (see **defProcessAlarm0**)
defProcessAlarm3	WORD	16#0000	Definition of process alarm 3 (see **defProcessAlarm0**)
defProcessAlarm4	WORD	16#0000	Definition of process alarm 4 (see **defProcessAlarm0**)
defProcessAlarm5	WORD	16#0000	Definition of process alarm 5 (see **defProcessAlarm0**)
defProcessAlarm6	WORD	16#0000	Definition of process alarm 6 (see **defProcessAlarm0**)
defProcessAlarm7	WORD	16#0000	Definition of process alarm 7 (see **defProcessAlarm0**)
timeOutLifeBit	TIME	T#0ms	Sign-of-life monitoring in higher-level control 0ms = sign-of-life monitoring switched off
Analog output			
weightAOZero	REAL	0.0	Weight for zero point (0 or 4 mA)
weightAOEnd	REAL	0.0	Weight for end value (20 mA)
weightAOonOD	REAL	0.0	Substitute value for analog output for CPU-STOP (OS, output disable)
srcAO	BYTE	16#00	Source for the analog output 16#00 = SIMOTION control signals 16#01 = external default value via DR17 16#02 = gross 16#03 = net 16#04 = coarse/fine default values
rangeAO	BOOL	FALSE	Current range for the analog output **TRUE** = 0 to 20 mA **FALSE** = 4 to 20 mA
RS232			
printerBdRate	BYTE	16#03	RS232 - printer baud rate 16#00 = 1200 bit/s 16#01 = 2400 bit/s 16#02 = 4800 bit/s 16#03 = 9600 bit/s

Name	Data type	Default	Definition
setXonXoff	BOOL	TRUE	Transfer control for the RS232 interfaces **TRUE** = XON/XOFF transfer control ON **FALSE** = XON/XOFF transfer control OFF
setRtsCts	BOOL	FALSE	Transfer control for the RS232 interfaces **TRUE** = CTS/RTS transfer control ON **FALSE** = CTS/RTS transfer control OFF
RS485			
setRS485Prot	BYTE	16#00	R485 log selection 16#00 = no device 16#01 = SIEBERT display S11 16#02 = reserve 16#03 = SIEBERT display S102
digitsRemDisplay	BYTE	16#00	Decimal place for remote display (0 to 4)
rs485BdRate	BYTE	16#03	RS485 baud rate 16#00 = 1200 bit/s 16#01 = 2400 bit/s 16#02 = 4800 bit/s 16#03 = 9600 bit/s 16#04 = 19200 bit/s 16#05 = 38400 bit/s
rs485Parity	BOOL	FALSE	RS485 bit parity **TRUE** = odd **FALSE** = even
rs485NumDataBits	BOOL	TRUE	RS485- data bits **TRUE** = 8 data bits **FALSE** = 7 data bits
rs485NumStopBits	BOOL	FALSE	RS485- stop bits **TRUE** = 2 stop bits **FALSE** = 1 stop bit
Digital outputs			
defDO1	BYTE	16#FF	Definition of digital output 1 Range of values 16#00 - 16#3F 16#00...16#1F = bit no. NAWI status bit 16#20...16#3F = bit no. AWI status bit 16#FD = start of cyclic transfer of measurement, digital load cell 16#FE = stop of cyclic transfer of measurement, digital load cell 16#FF = output always inactive Range of values 16#40... 16#FC not permitted!
defDO2	BYTE	16#FF	Definition of digital output 2 (see **defDO1**)
defDO3	BYTE	16#FF	Definition of digital output 3 (see **defDO1**)
defDO4	BYTE	16#FF	Definition of digital output 4 (see **defDO1**)
defDO5	BYTE	16#FF	Definition of digital output 5 (see **defDO1**)
defDO6	BYTE	16#FF	Definition of digital output 6 (see **defDO1**)
defDO7	BYTE	16#FF	Definition of digital output 7 (see **defDO1**)
defDO8	BYTE	16#FF	Definition of digital output 8 (see **defDO1**)

Name	Data type	Default	Definition
lowActiveDO1	BOOL	FALSE	Level definition of digital output 1 **TRUE** = output low active **FALSE** = output high active
lowActiveDO2	BOOL	FALSE	Level definition of digital output 2 **TRUE** = output low active **FALSE** = output high active
lowActiveDO3	BOOL	FALSE	Level definition of digital output 3 **TRUE** = output low active **FALSE** = output high active
lowActiveDO4	BOOL	FALSE	Level definition of digital output 4 **TRUE** = output low active **FALSE** = output high active
lowActiveDO5	BOOL	FALSE	Level definition of digital output 5 **TRUE** = output low active **FALSE** = output high active
lowActiveDO6	BOOL	FALSE	Level definition of digital output 6 **TRUE** = output low active **FALSE** = output high active
lowActiveDO7	BOOL	FALSE	Level definition of digital output 7 **TRUE** = output low active **FALSE** = output high active
lowActiveDO8	BOOL	FALSE	Level definition of digital output 8 **TRUE** = output low active **FALSE** = output high active
DO1onOD	BOOL	FALSE	Substitute value for digital output 1 for CPU STOP (OD) or error
DO2onOD	BOOL	FALSE	Substitute value for digital output 2 for CPU STOP (OD) or error
DO3onOD	BOOL	FALSE	Substitute value for digital output 3 for CPU STOP (OD) or error
DO4onOD	BOOL	FALSE	Substitute value for digital output 4 for CPU STOP (OD) or error
DO5onOD	BOOL	FALSE	Substitute value for digital output 5 for CPU STOP (OD) or error
DO6onOD	BOOL	FALSE	Substitute value for digital output 6 for CPU STOP (OD) or error
DO7onOD	BOOL	FALSE	Substitute value for digital output 7 for CPU STOP (OD) or error
DO8onOD	BOOL	FALSE	Substitute value for digital output 8 for CPU STOP (OD) or error
enableDOonError	BOOL	FALSE	Activate substitute value output for fault condition **TRUE** = substitute value for fault condition enabled **FALSE** = substitute value for fault condition disabled
xxxReserve3	BYTE	16#00	Reserve (use not permitted)
Digital inputs			
defDI1	BYTE	16#00	Definition of digital input 1 16#00 = no command 16#01... 16#FE = command code 16#FF = step enabling (see DR23, step control weighing parameters)
defDI2	BYTE	16#00	Definition of digital input 2 (see **defDI1**)
defDI3	BYTE	16#00	Definition of digital input 3 (see **defDI1**)
defDI4	BYTE	16#00	Definition of digital input 4 (see **defDI1**)
defDI5	BYTE	16#00	Definition of digital input 5 (see **defDI1**)

Name	Data type	Default	Definition
defDI6	BYTE	16#00	Definition of digital input 6 (see **defDI1**)
defDI7	BYTE	16#00	Definition of digital input 7 (see **defDI1**)
lowActiveDI1	BOOL	FALSE	Level definition of digital input 1 **TRUE** = input low active **FALSE** = input high active
lowActiveDI2	BOOL	FALSE	Level definition of digital input 2 (see **lowActiveDI1**)
lowActiveDI3	BOOL	FALSE	Level definition of digital input 3 (see **lowActiveDI1**)
lowActiveDI4	BOOL	FALSE	Level definition of digital input 4 (see **lowActiveDI1**)
lowActiveDI5	BOOL	FALSE	Level definition of digital input 5 (see **lowActiveDI1**)
lowActiveDI6	BOOL	FALSE	Level definition of digital input 6 (see **lowActiveDI1**)
lowActiveDI7	BOOL	FALSE	Level definition of digital input 7 (see **lowActiveDI1**)
measTimeForCntr	TIME	T#999ms	Measuring time for pulse input/counter input
xxxReserve4	DWORD	16#0000 0000	Reserve (use not permitted)
MMC parameters			
modeLogOverflow	BOOL	TRUE	Behavior during log overflow (MMC card full) **TRUE** = overwrite oldest entries when MMC memory is full **FALSE** = stop logging with MMC memory is full
modeTraceOverflow	BOOL	TRUE	Behavior during trace overflow (MMC card full) **TRUE** = overwrite oldest trace data when MMC card is full **FALSE** = trace data cannot be overwritten
traceDataToMMC	BOOL	FALSE	Trace data memory location **TRUE** = store trace data on MMC **FALSE** = store trace data in RAM
traceSizeMMC	BYTE	16#32	Memory segment on MMC for trace function in [%], 16#32 ≙ 50 % Up to 100%; however, the sum of the trace function and logs cannot exceed 100%
logSizeMMC	BYTE	16#32	Memory segment on MMC for logs in [%], 16#32 ≙ 50 % Up to 100%; however, the sum of the trace function and logs cannot exceed 100%
traceCycle	BYTE	16#01	Recording cycle for trace function in [ms] 1 to n x 10 ms

Struct_FTA_DR8

Overview

You can set the date and time on the module using the **Struct_FTA_DR8** data structure.

Note

If necessary, the current date and time can be read from data record DR31, **actDataAndTime** element.

Parameter description

Table 8-100 Struct_FTA_DR8

Name	Data type	Default	Definition
dateAndTime	DATE_AND_TIME	DT#1992-01-01-0:0:0.0	Date and time for the SIWAREX FTA weighing module (write)

Struct_FTA_DR9

Overview

Data structure **Struct_FTA_DR9** contains information about the SIWAREX FTA weighing module. These parameters are read-only.

Parameter description

The table below contains all parameters of the data structure **Struct_FTA_DR9**.

Table 8-101 Struct_FTA_DR9

Name	Data type	Default	Definition
Module information			
crcCheckSumFw	DWORD	16#00000000	Firmware checksum
lenFw	DWORD	16#00000000	Firmware length in bytes
moduleInfo	STRING[26]	' '	Siemens AG module serial number
moduleName	STRING[10]	' '	Module name
application	ARRAY[1..8] OF STRING[4]	[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']	Application identification
fileName	STRING[20]	'NAWI'	File name
typeVersion	BYTE	16#00	Version type 16#42 = 'B' = Lab status 16#50 = 'P' = Pilot 16#52 = 'R' = Release 16#53 = 'S' = Special status 16#56 = 'V' = Version 16#4B = 'K' = Revision status
fctVersion	USINT	0	Function status Major function changes or calibration-related changes (0 to 99)
dataStructVersion	USINT	0	Data record structure version Identifies changes in the data record structure (0 to 99)
corrVersion	USINT	0	Revision status Small changes or error corrections (0 to 99)
dateCreation	STRING[10]	' '	Creation date
timeCreation	STRING[8]	' '	Creation time

Name	Data type	Default	Definition
bootVersion	UINT	16#00	Bootloader version
scaleType	STRING[4]	'AWI' or 'NAWI'	AWI - Automatic Weighing Instrument NAWI - Non Automatic Weighing Instrument
xxxReserve1	WORD	16#0000	Reserve (use not permitted)

Struct_FTA_DR15

Description

Data structure **Struct_FTA_DR15** is used for external tare weight definition.

Parameters

Table 8-102 Struct_FTA_DR15

Name	Data type	Default	Definition
tareSetValue	REAL	0.0	External tare weight definition

Struct_FTA_DR16

Description

Data structure **Struct_FTA_DR16** is used for entering a weight simulation. The requirement for this is that the data structure **Struct_FTA_DR16** has been defined as the source for the weight simulation.

Parameters

Table 8-103 Struct_FTA_DR16

Name	Data type	Default	Definition
weightSimValue	REAL	0.0	Default value for weight simulation This value is used in place of the gross weight, for example, during testing.

Struct_FTA_DR17

Description

Data structure **Struct_FTA_DR17** is used with a predefined weight value to control the analog output. The requirement for this is that the data structure **Struct_FTA_DR17** has been defined in the data structure **Struct_FTA_DR7** as the source for controlling the analog output.

Parameters

Table 8-104 Struct_FTA_DR17

Name	Data type	Default	Definition
setAO	REAL	0.0	External definition for the analog output

Struct_FTA_DR18

Description

Data structure **Struct_FTA_DR18** is a structure for direct control of the remote display.

Parameters

Table 8-105 Struct_FTA_DR18

Name	Data type	Default	Definition
setValRemDisplay	REAL	0.0	External definition for the remote display

Struct_FTA_DR20

Description

Data structure **Struct_FTA_DR20** sends the setpoint weight for a weighing procedure to the SIWAREX FTA.

Parameters

Table 8-106 Struct_FTA_DR20

Name	Data type	Default	Definition
dosingSetpoint	REAL	50.0	Setpoint weight for the weighing operation

Struct_FTA_DR21

Description

The total amount of the material to be loaded is specified in loading mode. Data structure **Struct_FTA_DR21** sends the load setpoint weight for a weighing operation to the SIWAREX FTA.

Parameters

Table 8-107 Struct_FTA_DR21

Name	Data type	Default	Definition
loadingSetpoint	REAL	1000.0	Load quantity/total setpoint weight for loading mode

Struct_FTA_DR22

Description

Data structure **Struct_FTA_DR22** contains the weighing parameters. In general, the weighing parameters change when the material is changed and must be resent to the SIWAREX FTA weighing module.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR22**.

Table 8-108 Struct_FTA_DR22

Name	Data type	Default	Definition
Weighing parameter 1			
maxDosingTime	TIME	T#0ms	Maximum dosing time 0ms = deactivated
inFlightWeight	REAL	1	In-flight weight Amount that is still trailing (in-flight) after the fine signal has been switched off
fineWeight	REAL	20	Fine weight Amount to be dosed during the fine signal
switchOffCorr	REAL	0.0	Switch-off compensation value Additional positive/negative offset of the fine-signal switch-off point
timePreDosing	TIME	T#0ms	Pre-dosing time 0 = deactivated > 0 = pre-dosing time
upperTolValTO1	REAL	0.2	Upper tolerance limit 1 (value of the permissible positive deviation from the setpoint weight)
lowerTolValTU1	REAL	0.2	Lower tolerance limit 1 (value of the permissible negative deviation from the setpoint weight)
upperTolValTO2	REAL	0.5	Value of upper tolerance limit 2 **upperTolValTO2** must be larger than **upperTolValTO1**
lowerTolValTU2	REAL	0.5	Value of lower tolerance limit 2 **lowerTolValTU2** must be larger than **lowerTolValTU1**

Struct_FTA_DR23

Description

The weighing parameters contained in data structure **Struct_FTA_DR23** are typical scale parameters that do not depend greatly on material properties.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR23**.

Table 8-109 Struct_FTA_DR23

Name	Data type	Default	Definition
Weighing parameter 2			
selectText	BYTE	16#01	Text selection for automatic logging 16#00 = no automatic logging after weighing 16#01 = automatic logging with text 1 16#02 = automatic logging with text 2 16#03 = automatic logging with text 3 16#04 = automatic logging with text 4
xxxReserve1	BYTE	16#00	Reserve (use not permitted)
xxxReserve2	WORD	16#0000	Reserve (use not permitted)
maxSetpointDosing	REAL	90.0	Maximum setpoint weight for single dosing
disableTimeCoarse	TIME	T#500ms	Inhibit time coarse 0ms = deactivated After the coarse signal is switched on, weight evaluation is suspended for the specified time.
disableTimeFine	TIME	T#500ms	Inhibit time fine 0ms = deactivated After the coarse signal is switched off, weight evaluation is suspended for the specified time.
disableTimeCompare	TIME	T#0ms	Inhibit time for setpoint-actual comparison After the inhibit time is initiated by command, the current weight monitoring is suspended for the specified time during the weighing procedure.
valAOCoarse	BYTE	16#3C	Default value in [%] for analog output when coarse signal is active, 16#3C ≙ 60 %
valAOFine	BYTE	16#14	Default value in [%] for analog output when fine signal is active, 16#14 ≙ 20 %
filtTypeDosing	BYTE	16#00	Filter type for dosage control 16#00 = critically damped 16#01 = Bessel filter 16#02 = Butterworth filter Other definitions not permitted.

Name	Data type	Default	Definition
filtCutOffFreq	BYTE	16#04	Dosing filter limit frequency 16#00: No filter 16#01: fg = 20 Hz 16#02: fg = 10 Hz 16#03: fg = 5 Hz 16#04: fg = 2 Hz 16#05: fg = 1 Hz 16#06: fg = 0.5 Hz 16#07: fg = 0.2 Hz 16#08: fg = 0.1 Hz 16#09: fg = 0.05 Hz Other definitions not permitted.
Taring/zeroing			
modeZeroTare	BYTE	16#02	Tare/zeroing mode 16#00 = do not tare or zero when starting scale 16#01 = set to zero 16#02 = tare 16#03 = tare with mean value 16#04 = tare with external tare input Other definitions not permitted.
cycleZeroTare	BYTE	16#00	Tare/zeroing cycle 16#00 = each weighing procedure is zeroed or tared 16#01 = one fill is not zeroed or tared 16#02... 16#63 = 2...99 fills are not zeroed or tared Other definitions not permitted.
xxxReserve3	WORD	16#0000	Reserve (use not permitted)
minTareValue	REAL	0	Minimum tare weight Taring or external tare input is performed only if gross > minimum tare weight 0 = minimum tare weight is not monitored
maxTareValue	REAL	0	Maximum tare value Taring or external tare input is performed only if gross < maximum tare weight 0 = maximum tare weight is not monitored
timeAutoZeroing	TIME	T#5m	Cycle time for zeroing = 0ms: No time-controlled neutral position ≠ 0ms: Time between two neutral positions **Note:** For weighing mode AWI and country code "OIML", zeroing/taring is performed after 15 min., at the latest.
Step control / check stop			

Name	Data type	Default	Definition
waitDI1InStepX	BYTE	16#00	Step control via digital input 1 Instead of using parameterizable command codes for the digital inputs, step enabling for weighing control can also be controlled via the inputs. The requirement for this is that DR7 (interface parameter) must contain 16#FF. 16#00 = weighing waits in step 0, if DI1 is active 16#01 = weighing waits in step 1, if DI1 is active 16#02 = weighing waits in step 2, if DI1 is active ... 16#07 = weighing waits in step 7, if DI1 is active Other definitions not permitted.
waitDI2InStepX	BYTE	16#00	Step control via digital input 2 (see **waitDI1InStepX**)
waitDI3InStepX	BYTE	16#00	Step control via digital input 3 (see **waitDI1InStepX**)
waitDI4InStepX	BYTE	16#00	Step control via digital input 4 (see **waitDI1InStepX**)
waitDI5InStepX	BYTE	16#00	Step control via digital input 5 (see **waitDI1InStepX**)
waitDI6InStepX	BYTE	16#00	Step control via digital input 6 (see **waitDI1InStepX**)
waitDI7InStepX	BYTE	16#00	Step control via digital input 7 (see **waitDI1InStepX**)
xxxReserve4	BYTE	16#00	Reserve (use not permitted)
timeOutOneStep	TIME	T#0ms	Monitoring time for step control = 0ms: No monitoring > 0ms: Time target for monitoring If a transition to the next step does not occur within the defined time, the "transition timeout" technology error is issued.
stopAfterStep1	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 1 **FALSE** = no check stop
stopAfterStep2	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 2 **FALSE** = no check stop
stopAfterStep3	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 3 **FALSE** = no check stop
stopAfterStep4	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 4 **FALSE** = no check stop
stopAfterStep5	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 5 **FALSE** = no check stop
stopAfterStep6	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 6 **FALSE** = no check stop
stopAfterStep7	BOOL	FALSE	Check stop definition **TRUE** = weighing goes to check stop after step 7 **FALSE** = no check stop
xxxReserve5	BYTE	16#00	Reserve (use not permitted)
Follow-up dosing tolerance check			

Name	Data type	Default	Definition
autoPostDosing	BOOL	FALSE	Automatic follow-up dosing **FALSE** = no automatic follow-up dosing **TRUE** = automatic follow-up dosing for deviation from tolerance limit
modePostDosing	BOOL	FALSE	Follow-up dosing type **FALSE** = follow-up dosing with continuous fine signal **TRUE** = follow-up dosing in JOG mode
stopTO1Limit	BOOL	FALSE	Stop when upper tolerance limit 1 (TO1) violated **FALSE** = weighing is not stopped due to tolerance error **TRUE** = weighing is stopped due to tolerance error (weight above TO1)
stopTO2Limit	BOOL	FALSE	Stop when upper tolerance limit 2 (TO2) violated **FALSE** = weighing is not stopped due to tolerance error **TRUE** = weighing is stopped due to tolerance error (weight above TO2)
stopTU1Limit	BOOL	FALSE	Stop when lower tolerance limit 1 (TU1) violated **FALSE** = weighing is not stopped due to tolerance error **TRUE** = weighing is stopped due to tolerance error (weight below TU1)
stopTU2Limit	BOOL	FALSE	Stop when lower tolerance limit 2 (TU2) violated **FALSE** = weighing is not stopped due to tolerance error **TRUE** = weighing is stopped due to tolerance error (weight below TU2)
continueTolStop	BOOL	FALSE	Continue after stop due to tolerance error **FALSE** = cycle cannot be continued if tolerance error exists **TRUE** = cycle can be continued despite existing tolerance error
numNoTolCheck	BYTE	16#00	Check for tolerance deviations 16#00 = all weighing operations are checked for tolerance deviations 16#01 = a weighing operation is not checked for tolerance deviation 16#02... 16#62 = 2...98 weighing operations are not checked for tolerance deviations 16#63 = tolerance deviation check deactivated Other definitions not permitted.
timePulseInching	TIME	T#1s	Pulse duration of fine signal
Controller			
ctrlErrReaction	BYTE	16#00	Controller response to weighing error Bit 0 = 0: Reset controller in response to technology error (weighing error) Bit 0 = 1: Limit controller to maximum control action Bits 1 to 7 not used
typeController	BYTE	16#00	Selection of controller type 16#00 = no control for switching off coarse/fine signal 16#01 = proportional controller without fine signal time controller 16#02 = proportional controller with fine signal time controller 16#03 = fine signal time controller without proportional controller Other definitions not permitted.
factorController	BYTE	16#1E	Control factor for proportional controller [0 to 100%], 16#1E ≙ 30 %
xxxReserve6	BYTE	16#00	Reserve (use not permitted)
limitController	REAL	1.0	Maximum one-time control action Limitation of maximum one-time control action of proportional controller
optiPlusCtrl	REAL	0.0	Optimum plus controller
optiMinusCtrl	REAL	0.0	Optimum minus controller

Name	Data type	Default	Definition
setFineTime	TIME	T#3s	Fine time setpoint
factorFineTime	BYTE	16#14	Control factor for fine-time controller [0 to 100 %], 16#14 ≙ 20 %
xxxReserve7	BYTE	16#00	Reserve (use not permitted)
Emptying			
xxxReserve8	WORD	16#0000	Reserve (use not permitted)
timeOverlap	TIME	T#0ms	Bridging time The overlap time **timeOverlap** must be less than the emptying time **timeEmptying**. The next weighing operation can be advanced by the bridging time and can begin during the emptying operation, after a predefined time has elapsed.
timeEmptying	TIME	T#0ms	Emptying time = 0ms: Emptying depends on the emptying range > 0ms: Emptying occurs after a predefined time
timeOutEmptying	TIME	T#0ms	Maximum emptying time = 0ms: Monitoring is switched off > 0ms: A technology error is issued if the empty range has not been reached by the emptying timeout.
Loading			
modeLoading	BYTE	16#00	Loading with coarse 16#00 = all weighing operations during loading mode are controlled with coarse and fine signals 16#01 = only coarse feed is used for weighing; however, the last 5 weighing operations are performed with coarse and fine signals 16#02 = only coarse feed is used for weighing; however, the last 4 weighing operations are performed with coarse and fine signals 16#04 = only coarse feed is used for weighing; however, the last 3 weighing operations are performed with coarse and fine signals 16#08 = only coarse feed is used for weighing; however, the last 2 weighing operations are performed with coarse and fine signals 16#10 = only coarse feed is used for weighing; however, the last weighing operation is performed with coarse and fine signals Other entries not permitted.
xxxReserve9	BYTE	16#00	Reserve (use not permitted)

Struct_FTA_DR26

Description

Data structure **Struct_FTA_DR26** contains the internal process values. These internal process values can be used to read out the current internal statuses and data of the scale in the event of servicing.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR26**.

Table 8-110 Struct_FTA_DR26

Name	Data type	Default	Definition
presetTareActive	BOOL	FALSE	**TRUE** = tare memory is assigned and activated by an external default value
xxxReserve1	BYTE	16#00	Reserve (use not permitted)
xxxReserve2	BYTE	16#00	Reserve (use not permitted)
internState	BYTE	16#00	Internal status Bit 0 = 0: SIMOTION mode activated Bit 0 = 1: Stand-alone mode activated Bit 1 = 0: Recording of digital load cell stopped Bit 1 = 1: Recording of digital load cell activated Bits 2 to 7 not used (use not permitted)
actTareWeight	REAL	0.0	Actual tare weight (process value)
actAverTareWeight	REAL	0.0	Actual tare mean (when taring with mean value)
pwrOnZeroValue	REAL	0.0	Zero value Value is set during power-up if "power-on zero value" is activated.
zeroValue	REAL	0.0	Zero value Value is set during zeroing
zeroCorrValue	REAL	0.0	Zero compensation value Value is controlled by automatic zero correction
impedRefValue	INT	0	Impedance reference value
actImpedValue	INT	0	Actual impedance value from last measurement
lastMaxWeight	REAL	0.0	Last maximum weight value
numOpMinutes	UDINT	0	Operating minutes counter
maxTemperature	INT	0	Previous maximum internal temperature measured since the last loading of default values [0.1 °C] , (**maxTemperature** := 300 corresponds to 30.0 °C) If a temperature value is not available: -100.0 °C.
xxxReserve3	BYTE	16#00	Reserve (use not permitted)
xxxReserve4	BYTE	16#00	Reserve (use not permitted)
sigLevel	INT	0	Signal level at measuring input
crc	WORD	16#0000	Checksum

Struct_FTA_DR30

Description

Data structure **Struct_FTA_DR30** contains process values that can be used to observe the current statuses and data in the scale. These observations can then be used as a basis for optimizing the parameters in test mode.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR30**.

Table 8-111 Struct_FTA_DR30

Name	Data type	Default	Definition
NAWI status bits			
stateNAWI	DWORD	16#0000 0000	NAWI status bits, 32 status displays for NAWI The 32 NAWI status bits are listed below.
weightInRange1	BOOL	FALSE	**TRUE** = Weight is within weighing range 1
weightInRange2	BOOL	FALSE	**TRUE** = Weight is within weighing range 2
weightInRange3	BOOL	FALSE	**TRUE** = Weight is within weighing range 3
limit1On	BOOL	FALSE	**TRUE** = Limit value 1 has been activated
limit2On	BOOL	FALSE	**TRUE** = Limit value 2 has been activated
limit3On	BOOL	FALSE	**TRUE** = Limit value 3 has been activated
scaleTared	BOOL	FALSE	**TRUE** = if the scale is tared
scaleTaredManual	BOOL	FALSE	**TRUE** = if the scale has been tared with a manual tare entry
weightMax9e	BOOL	FALSE	Max. weight for weighing range + 9*e (e = verification interval) **TRUE** = if the maximum load has been exceeded by 9 e
weight025dZero	BOOL	FALSE	**TRUE** = if the weight does not exceed ¼ d (d = numerical increment)
waitOfStandStill1	BOOL	FALSE	**TRUE** = if the scale is waiting for standstill after startup
standStill1On	BOOL	FALSE	**TRUE** = standstill 1 is on.
scaleCalibrated	BOOL	FALSE	**TRUE** = if the scale is calibrated
cmdErrOnDI	BOOL	FALSE	**TRUE** = if a command could not be executed on a digital input
simWeighingOn	BOOL	FALSE	**TRUE** = if weight simulation has been activated
serviceModeOn	BOOL	FALSE	**TRUE** = if service mode has been activated
printingOn	BOOL	FALSE	**TRUE** = The log is being printed.
printImpossible	BOOL	FALSE	**TRUE** = The log cannot be printed.
MMCconnected	BOOL	FALSE	**TRUE** = The MMC is connected.
MMCready	BOOL	FALSE	**TRUE** = The MMC is formatted and ready for recording.
MMCreadyForTrace	BOOL	FALSE	**TRUE** = The MMC is ready for the trace function.
MMCreadyForLog	BOOL	FALSE	**TRUE** = The MMC is ready for logging.
MMCTraceActive	BOOL	FALSE	**TRUE** = The trace function is activated.
minFlowCtrl1On	BOOL	FALSE	**TRUE** = Flow control 1 has been activated.
minFlowCtrl2On	BOOL	FALSE	**TRUE** = Flow control 2 has been activated.
scaleEmptyRange	BOOL	FALSE	**TRUE** = The scale is in empty range.
protectionOn	BOOL	FALSE	**TRUE** = The switch for protecting the calibration data is on.
xxxReserve1	BOOL	FALSE	Reserve (use not permitted)
MMCDataPrepared	BOOL	FALSE	**TRUE** = Preparation of the MMC data with the values defined in DR46 is complete. The data can be read out with DR47.
digLCactive	BOOL	FALSE	**FALSE** = digital load cell sensing is not active **TRUE** = digital load cell sensing is in progress
standAloneActive	BOOL	FALSE	**FALSE** = operation with programmable controller is activated **TRUE** = stand-alone mode is activated

Name	Data type	Default	Definition
errorFTAoccurred	BOOL	FALSE	**TRUE** = At least one fault condition (fault) has occurred.
AWI status bits			
stateAWI	DWORD	16#0000 0000	AWI status bits, 32 status displays for AWI The 32 AWI status bits are listed below.
dosingStep0	BOOL	FALSE	Current step 0 of weighing control
dosingStep1	BOOL	FALSE	Current step 1 of weighing control
dosingStep2	BOOL	FALSE	Current step 2 of weighing control
dosingStep3	BOOL	FALSE	Current step 3 of weighing control
dosingStep4	BOOL	FALSE	Current step 4 of weighing control
dosingStep5	BOOL	FALSE	Current step 5 of weighing control
dosingStep6	BOOL	FALSE	Current step 6 of weighing control
dosingStep7	BOOL	FALSE	Current step 7 of weighing control
postDosingActive	BOOL	FALSE	**TRUE** = Follow-up dosing is active.
coarseSigOn	BOOL	FALSE	**TRUE** = The coarse signal is switched on.
fineSigOn	BOOL	FALSE	**TRUE** = The fine signal is switched on.
timerPreDosingOn	BOOL	FALSE	**TRUE** = The timer for pre-dosing is active.
emtyingSigOn	BOOL	FALSE	**TRUE** = The emptying signal is switched on.
weighingStopped	BOOL	FALSE	**TRUE** = The weighing cycle has been stopped.
stoppedForCheck	BOOL	FALSE	**TRUE** = The weighing cycle has been stopped by the check stop command.
checkStopFollow	BOOL	FALSE	**TRUE** = The weighing cycle will be stopped by a check stop (set with check stop command and reset when check stop is reached).
dosingAborted	BOOL	FALSE	**TRUE** = last weighing operation aborted by "residual weighing" or "weighing control reset".
nextStepWaiting	BOOL	FALSE	**TRUE** = if the step enabling to the next step in the weighing cycle has been blocked due to a missing step enable.
upperLimit2On	BOOL	FALSE	**TRUE** = net weight above the TO2 limit
upperLimit1On	BOOL	FALSE	**TRUE** = net weight above the TO1 limit
toleranceOK	BOOL	FALSE	**TRUE** = net weight within tolerance range TU1 to TO1
lowerLimit1On	BOOL	FALSE	**TRUE** = net weight below TU1 but above TU2
lowerLimit2On	BOOL	FALSE	**TRUE** = net weight below the TU2 limit
toleranceBad	BOOL	FALSE	**TRUE** = net weight below TU2 or above TO2
standStill2On	BOOL	FALSE	**TRUE** = standstill 2 is on
standStill3On	BOOL	FALSE	**TRUE** = standstill 3 is on
checkFollows	BOOL	FALSE	**TRUE** = A check weighing operation is performed at the end of the cycle.
comparatorDisable	BOOL	FALSE	**TRUE** = The setpoint/actual comparison is disabled and the weighing operation is being executed without a weight evaluation!
continueModeOn	BOOL	FALSE	**TRUE** = Continuous start is activated for the cycle sequence.
xxxReserve2	BOOL	FALSE	Reserve (use not permitted)
endOfDosingCycle	BOOL	FALSE	**TRUE** = The weighing cycle is complete.
endOfCharge	BOOL	FALSE	**TRUE** = The emptying operation is complete.
Process and weight values			
actGrossWeight	REAL	0.0	Actual gross weight (process value)
actNetWeight	REAL	0.0	Actual net weight (process value)
actTareWeight	REAL	0.0	Actual tare weight (process value)

Name	Data type	Default	Definition
weight	REAL	0.0	Actual weight *1 (numerical increment from DR3)
weightX10	REAL	0.0	Actual weight *10 (numerical increment from DR3)
tareWeight	REAL	0.0	Actual tare weight (numerical increment from DR3)
lastCheckedWeight	REAL	0.0	Net weight from last checked weighing operation (numerical increment from DR3)
pulseCntrValue	UDINT	0	Current value from pulse counter (counter input)
sumMem1	LREAL	0.0	Totalizing memory 1 (with calibration capability) Current value in totalizing memory 1 (numerical increment from DR3)
sumMem2	REAL	0.0	Totalizing memory 2 Current value in totalizing memory 2 (numerical increment from DR3)

Struct_FTA_DR31

Description

Data structure **Struct_FTA_DR31** contains process values that can be used to observe additional current statuses and data in the scale. These observations can then be used as a basis for optimizing the parameters in test mode.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR31**.

Table 8-112 Struct_FTA_DR31

Name	Data type	Default	Definition
Extended process values			
actFlow	REAL	0.0	Current flow rate [amount per second]
actInFlightWeight	REAL	0.0	Current in-flight weight The in-flight weight is initialized with a default value when the filling parameters are received.
actFineWeight	REAL	0.0	Current fine weight The fine weight is initialized with a default value when the filling parameters are received.
actAduValue	DINT	0	Unfiltered ADC value Direct value from the analog/digital converter
actAduValueFilt1	DINT	0	Direct value from the analog/digital converter, after filter 1 (DR3)
actAduValueFilt2	DINT	0	Direct value from the analog/digital converter, after filter 2 (DR3)
actRestLoading	REAL	0.0	Current residual amount in loading mode
actSetpointLoading	REAL	0.0	Current setpoint of an individual weighing operation in loading mode

Name	Data type	Default	Definition
actStateError	DWORD	16#00000000	Operating error (bit-coded) Current status of the 32 operating errors (faults) Bit 0 = TRUE: Fault condition 1 has occurred Bit 1 = TRUE: Fault condition 2 has occurred Bit 2 = TRUE: Fault condition 3 has occurred ... Bit 30 = TRUE: Fault condition 31 has occurred Bit 31 = TRUE: Fault condition 32 has occurred
actDateAndTime	DATE_AND_TIME	DT#1992-01-01-0:0:0.0	Actual date and time in SIWAREX FTA weighing module
actTemperature	INT	0	Current temperature of SIWAREX FTA [0.1 °C], (**actTemperature** := 300 corresponds to 30.0 °C)
actStateDI	BYTE	16#00	Current status of digital inputs Bit 0 = status of digital input 1 Bit 1 = status of digital input 2 Bit 2 = status of digital input 3 Bit 3 = status of digital input 4 Bit 4 = status of digital input 5 Bit 5 = status of digital input 6 Bit 6 = status of digital input 7 Bit 7 = not used
actStateLC	BYTE	16#00	Status of digital load cell Bit 0 = TRUE: Overload Bit 1 = TRUE: Underload Bit 2 = TRUE: Unknown command Bit 3 = TRUE: Command cannot be executed Bit 4 = TRUE: Standstill Bit 5 = TRUE: Hardware fault Bit 6 = TRUE: Not used Bit 7 = TRUE: Communication fault
measImpedance	INT	0	Impedance reference value Measured impedance value of load cells [0.1 Ω], (**measImpedance** := 100 corresponds to 10.0 Ω)
actImpedance	INT	0	Actual impedance value Current impedance value of load cells [0.1 Ω], (**actImpedance** := 100 corresponds to 10.0 Ω)

Struct_FTA_DR32

Description

Data structure **Struct_FTA_DR32** contains the statistical data of the SIWAREX FTA weighing module. The statistical data provide information about the quality of the weighing operations.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR32**.

Table 8-113 Struct_FTA_DR32

Name	Data type	De-fault	Definition
totalNumWeighing	DINT	0	Total number of weighing operations (with and without tolerance check)
numCheckedWeigh-ing	DINT	0	Number of weighing operations with tolerance check
numTO2Weighing	DINT	0	Number of weighing operations with tolerance check above tolerance limit TO2
numTO1Weighing	DINT	0	Number of weighing operations with tolerance check above tolerance limit TO1
numGoodWeighing	DINT	0	Number of weighing operations with tolerance check within the range TU1 to TO1
numTU1Weighing	DINT	0	Number of weighing operations with tolerance check below tolerance limit TU1
numTU2Weighing	DINT	0	Number of weighing operations with tolerance check below tolerance limit TU2
numBadWeighing	DINT	0	Number of weighing operations with tolerance check below TU2 or above TO2
xxxReserve1	DINT	0	Reserve (use not permitted)
xxxReserve2	DINT	0	Reserve (use not permitted)
actSetpointWeight	REAL	0.0	Current setpoint weight (numerical increment with calibration capability, rounded to the nearest numerical increment for AWI applications with country code "OIML")
averValue	REAL	0.0	Mean value of net weights checked for tolerance errors
standardDeviation	REAL	0.0	Standard deviation of net weights checked for tolerance errors
thruputPerHour	REAL	0.0	Throughput per hour [e.g. g/h, kg/h or t/h] is calculated based on the net weight of the last weighing operation.
weighingPerHour	INT	0	Weighing operations per hour are calculated based on the last weighing operation (time for one weighing cycle).

Struct_FTA_DR34

Description

Data structure Struct_FTA_DR34 is used to output the current weight in ASCII format according to the main scale display.

Parameters

Table 8-114 Struct_FTA_DR34

Name	Data type	Default	Definition
actWeightASCII	STRING[16]	' '	Actual weight in ASCII format (as output on the display)

Struct_FTA_DR35

Description

Data structure **Struct_FTA_DR35** contains encrypted data for calibration-capable display via an operator panel that can handle the encryption (for example, SIMATIC HMI).

Parameters

Table 8-115 Struct_FTA_DR35

Name	Data type	Default	Definition
cryptoData	ARRAY [0 to 31] of BYTE	32 (16#00)	Encrypted data for weight display with calibration capability

Struct_FTA_DR39

Description

The **Struct_FTA_DR39** data structure stores the version ID of the "SecureOCX" weight display with calibration capability. For a weight display with calibration capability, the content of DR39 must be identical to the version of the "SecureOCX", which is used in the WinCC flexible configuration.

The version is entered during commissioning.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR39**.

Table 8-116 Struct_FTA_DR39

Name	Data type	Default	Definition
typeVersion	STRING[1]	' '	"V" for product version
xxxReserve1	BYTE	16#00	Reserve 1 (use not permitted)
versionPrimary	UINT	0	Sequential number between 0 and 15
versionSecondary	UINT	0	Sequential number between 0 and 15

Example

The following figure clearly shows the relation between the **Struct_FTA_DR39** data structure and the version ID of the "SecureOCX".

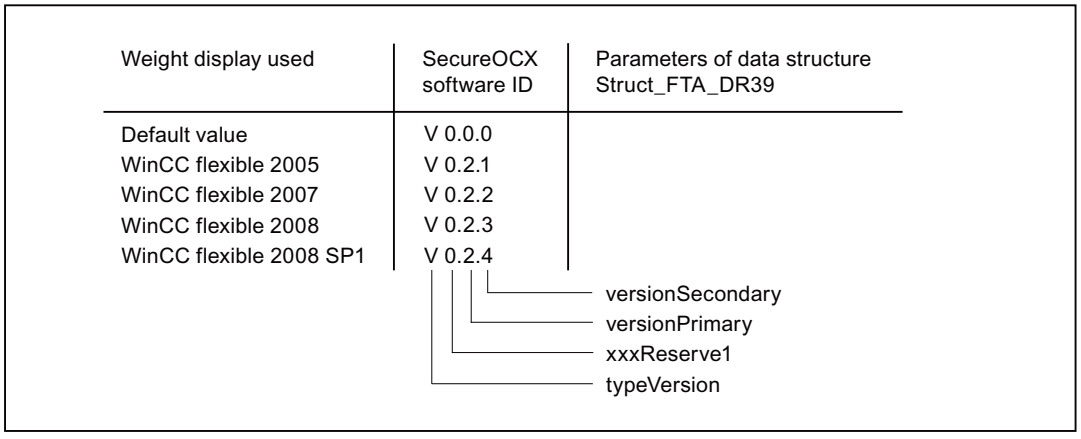


Figure 8-47 SecureOCX software ID

See also

Weight display with calibration capability (Page 6400)

Struct_FTA_DR44

Description

Data structure **Struct_FTA_DR44** stores the log data until the next logging operation. This content can be output again, if necessary.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR44**.

Table 8-117 Struct_FTA_DR44

Name	Data type	Default	Definition
MMCId	ARRAY [0 to 4] of BYTE	5 (16#00)	Identification of micro memory card 1 byte for manufacturer's ID + 4 bytes for serial number; always 0 for log output on RS232 interface
xxxReserve1	BYTE	16#00	Reserve 1 (use not permitted)
xxxReserve2	WORD	16#0000	Reserve 2 (use not permitted)
logId	DINT	0	Log ID ID of stored log text; number is incremented for each logging procedure
lastLogData	STRING[160]	STRING[160] = ' '	Log text from last logging procedure

Struct_FTA_DR45

Description

Data structure **Struct_FTA_DR45** contains variable text that can be inserted in the logs as strings. The content of these strings can be defined via the SIMOTION controller.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR45**.

Table 8-118 Struct_FTA_DR45

Name	Data type	Default	Definition
varText1	STRING[16]	' '	Variable text (string 1)
varText2	STRING[16]	' '	Variable text (string 2)
varText3	STRING[16]	' '	Variable text (string 3)
varText4	STRING[16]	' '	Variable text (string 4)

Struct_FTA_DR46

Description

Data structure **Struct_FTA_DR46** contains the necessary parameters for reading out the logs stored on the MMC of the SIWAREX FTA weighing module. Data structures **Struct_FTA_DR46** and **Struct_FTA_DR47** can be used to read out any log in the the SIMOTION device.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR46**.

Table 8-119 Struct_FTA_DR46

Name	Data type	Default	Definition
selectLogId	DINT	0	ID number for read-out With this setting, the log is read out with its ID number when DR47 is read out. If request last data record (reqLastLog = 1) is activated, the ID number is ignored.
reqLastLog	BYTE	16#00	Request for last data record If this parameter setting = 1, the last log is output via data record DR47.
xxxReserve1	BYTE	16#00	Reserve (use not permitted)

Struct_FTA_DR47

Description

Data structure **Struct_FTA_DR47** provides the log data that were requested via data record DR46.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR47**.

Table 8-120 Struct_FTA_DR47

Name	Data type	Default	Definition
MMCId	ARRAY [0 to 4] of BYTE	5 (16#00)	Identification of the MMC card 1 byte for manufacturer's ID + 4 bytes for serial number; always 0 for log output on RS232 interface
xxxReserve1	BYTE	16#00	Reserve (use not permitted)
xxxReserve2	WORD	16#0000	Reserve (use not permitted)
logId	DINT	0	Log ID ID of the stored log text
logData	STRING[160]	´ ´	Log text

Struct_FTA_DR123

Description

Data structure **Struct_FTA_DR123** contains an overview of the data stored on the MMC card. Based on this information, the user can read out specific recordings and logs.

Parameters

The table below contains all parameters of the data structure **Struct_FTA_DR123**.

Table 8-121 Struct_FTA_DR123

Name	Data type	Default	Definition
logId	DWORD	16#00000000	Log ID Number assigned to each log output; incremented each time a log is output
MMCId	ARRAY [0 to 4] of BYTE	5 (16#00)	Identification of the MMC card 1 byte for manufacturer's ID + 4 bytes for serial number
xxxReserve1	BYTE	16#00	Reserve (use not permitted)
xxxReserve2	WORD	16#0000	Reserve (use not permitted)
MMCCapacity	DINT	0	Total memory capacity of the MMC card in [bytes]
MMCLogCapacity	DINT	0	Total memory capacity of the MMC card in [bytes] for log data

Name	Data type	Default	Definition
traceCapacity	DINT	0	Available capacity for trace data in [bytes] Display depends on the defined trace destination: RAM or MMC in the interface parameters.
oldMMCLogId	DINT	0	Oldest MMC log ID
newMMCLogId	DINT	0	Newest MMC log ID
oldMMCTraceId	DINT	0	Oldest MMC trace ID
newMMCTraceId	DINT	0	Newest MMC trace ID
oldRAMTraceId	DINT	0	Oldest RAM trace ID
newRAMTraceId	DINT	0	Newest RAM trace ID

8.8.4.3 Weight display with calibration capability

Overview

The value for the weight display with calibration capability is formed internally by SIWAREX FTA, encrypted and made available to the user in the DR35 data record.

Function description

The DR35 data record is read using the **_FTA_control** function block and stored in the **Struct_FTA_scaleData** data structure in the DR35 element.

The DR35 and DR39 data records and the winccOcxWrite element of the **Struct_FTA_scaleData** data structure are used to ensure reciprocal monitoring of the "SecureOCX" add-on and the SIWAREX firmware.

The version number (ID) of the weight display with calibration capability used has to be entered in the DR39 data record (see table below).

Table 8-122 Version numbers of weight displays with calibration capability

Display used	Version number
WinCC flexible 2005	V0.2.1
WinCC flexible 2007	V0.2.2
WinCC flexible 2008	V0.2.3
WinCC flexible 2008 SP1	V0.2.4

The "SecureOCX" add-on for WinCC flexible, a special function which is able to decipher the content of the DR35 and display it in a special output field is used for evaluating. "SecureOCX" has to be installed in addition to WinCC flexible.

Note

Refer to the *SIWAREX FTA Electronic Weighing System for Automatic Scale* Manual for details of how to install and configure the weight display with calibration capability in WinCC flexible and which visualization devices can be used.

This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

Error messages

If the data received are incorrect, "ERROR 1" is output. If the DR35 data record is not updated (monitoring time approx. 2 s), the text "ERROR 2" appears on the display instead of the weight value. Errors in the reciprocal recognition of the SIWAREX firmware and "SecureOCX" produce ERROR 3, if regulations = OIML is set. When regulations = ----, the current weight value is displayed.

Create variable

An ARRAY[0..1] OF WORD type variable must be created as VAR_GLOBAL in the interface area in the SIMOTION device for data transfer between the "SecureOCX" add-on and SIWAREX FTA. The name of these variables must be configured for write-only access in the "SecureOCX", e.g. myWinccOcxWrite (see following diagram). The further processing of these variables in the SIMOTION device is explained in the call example for the **_FTA_control** function block, see Calling the function block (Page 6402).

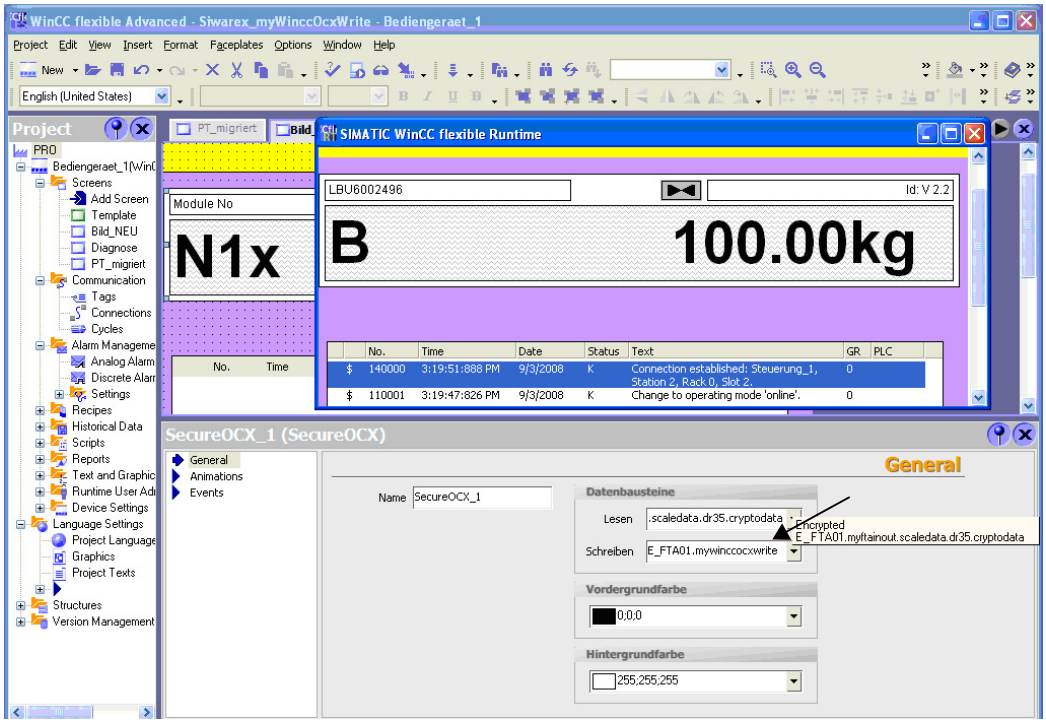


Figure 8-48 Configuration of reading and writing access in the "SecureOCX" add-on

8.8.4.4 Calling the function block

Procedure

In order to work with the **_FTA_control** function block in your user program, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Set up variables for the **Struct_FTA_scaleData** data structure.
2. Create an instance of the **_FTA_control** function block.
3. Create variable for module address from **HW config** and assign module address.
4. Create a field for in/out parameters of the function block.
5. Variable for the data transfer between the "SecureOCX" weight display with calibration capability and SIWAREX FTA.
6. Assignment of module address.

7. Assign ARRAY OF WORD type variable (e.g. myWinccOcxWrite) to winccOcxWrite element of **Struct_FTA_scaleData** data structure.
8. Call the function block instance.
9. Transfer input parameters.
10. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.
11. The data prepared by the function block for the I/O outputs are assigned to the I/O variable by the field created in step 3.

Note

This call example is excerpted from the "E_FTA01" application example, which is included on the "SIMOTION Utilities & Applications" CD-ROM.

If you would like to control multiple weighing modules, you must create a new variable for the data structure and FB instances with a new name for each weighing module.

Call example

```
UNIT E_FTA01;

INTERFACE

    TYPE
        // input - parameters
        StructFTAIn       : STRUCT
            execute       : BOOL;              // execute the selected command
            cmdNumber     : UINT;              // command number of the command
            ackErrorFTA   : BOOL;              // acknowledge errors
            moduleAddress : DINT;              // address of the FTA
            simValue      : REAL;              // simulation weighing value
            setAO         : REAL;              // set analog output
            forceDO       : BYTE;              // force digital output
            transitions   : BYTE;              // set transitions
        END_STRUCT

        // in/out - parameters                                                    (1)
        StructFTAInOut  : STRUCT
           ScaleData    : Struct_FTA_scaleData; // structure with the data records
        END_STRUCT
```

```
        // output - parameters
        StructFTAOut    : STRUCT
        done            : BOOL;                 // command done
        busy            : BOOL;                 // command at working
        error           : BOOL;                 // error occurred
        errorID         : WORD;                 // error-ID of the error
        errorIdTransfer : WORD;                 // error during readrecord or writerecord
        errorIdCommand  : WORD;                 // error by the command
        errorFTA        : BOOL;                 // new error generated
        errorTypeFTA    : enum_FTA_ErrorType;   // type of error
        errorIdFTA      : UINT;                 // error-ID of the new error
        cntrRefresh     : UINT;                 // signalized the refresh of the output values
                                                // val1Process and val2Process
        val1Process     : REAL;                 // parameterized output value 1
        val2process     : DWORD;                // parameterized output value 2
        scaleStatus     : DWORD;                // status of the fta
        startup         : BOOL;                 // fta at startup
        END_STRUCT
 END_TYPE

    VAR_GLOBAL                                                                          (2)
        myFTA_control   : _FTA_control;         // create instance of FB
        myFTAIn         : StructFTAIn;          // input parameters of the fta
        myFTAInOut      : StructFTAInOut;       // in-out parameters of the fta
        myFTAOut        : StructFTAOut;         // output parameters of the fta
        myModuleAddr    : DINT := 256;          // module address of SIWAREX FTA           (3)
        myIOPeriOut     : ARRAY[0..15] OF BYTE;                                         (4)
        myWinccOcxWrite : ARRAY[0..1] OF WORD;  // variable for data transfer              (5)
                                                // SecureOCX - SIWAREX FTA
    END_VAR

    PROGRAM ExampleFTA ;                        // Program in BackgroundTask

END_INTERFACE

IMPLEMENTATION

    // Program in BackgroundTask
    PROGRAM ExampleFTA

      myFTAIn.moduleAddress := myModuleAddr;        // address of the fta                (6)
```

```
    myFTAInOut.scaleData.winccOcxWrite                                          (7)
                    := _DWORD_FROM_2WORD (myWinccOcxWrite[1], myWinccOcxWrite[0]);
                    // assign variable from weighing display of
                    // AddOn WinCC flexible "SecureOCX"

    // CALL FB INSTANCE                                                         (8)
    myFTA_control (                                                             (9)
      periIn         := myperiIn,                // I/O-Input
      execute        := myFTAIn.execute,         // execute selected command
      cmdNumber      := myFTAIn.cmdNumber,       // select command
      ackErrorFTA    := myFTAIn.ackErrorFTA,     // acknowledge error
      moduleAddress  := myFTAIn.moduleAddress,   // address of the FTA
      simValue       := myFTAIn.simValue,        // simulation value
      setAO          := myFTAIn.setAO,           // set analog output
      forceDO        := myFTAIn.forceDO,         // force digital output
      transitions    := myFTAIn.transitions,     // transitions
      // IN/OUT
      scaleData      := myFTAInOut.scaleData,    // datarecords
      periOut        := myIOperiOut              // I/O-Output
    );

    // return from FB _FTA_control                                              (10)
     myFTAOut.done             := myFTA_control.done;            // command done
     myFTAOut.busy             := myFTA_control.busy;            // FB is working
     myFTAOut.error            := myFTA_control.error;           // error occured
     myFTAOut.errorID          := myFTA_control.errorID;         // information of the error
     myFTAOut.errorIdTransfer := myFTA_control.errorIdTransfer; // transfer error
     myFTAOut.errorIdCommand  := myFTA_control.errorIdCommand;  // command error
     myFTAOut.errorFTA         := myFTA_control.errorFTA;        // new error
     myFTAOut.errorTypeFTA     := myFTA_control.errorTypeFTA;    // type of new error
     myFTAOut.errorIdFTA       := myFTA_control.errorIdFTA;      // errorId of new error
     myFTAOut.cntrRefresh      := myFTA_control.cntrRefresh;     // signalized refresh
     myFTAOut.val1Process      := myFTA_control.val1Process;     // parametrized output 1
     myFTAOut.val2process      := myFTA_control.val2process;     // parametrized output 2
     myFTAOut.scaleStatus      := myFTA_control.scaleStatus;     // status
     myFTAOut.startup          := myFTA_control.startup;         // at startup

     myperiOut                 := myIOPeriOut;                                  (11)

   END_PROGRAM

END_IMPLEMENTATION
```

8.8.5 Application example

8.8.5.1 General information about the application example

Task

This application example shows you how you can use the **_FTA_control** function block to control the SIWAREX FTA weighing module using 3 commands:

- Read data from the SIWAREX FTA weighing module
- Write data to the SIWAREX FTA weighing module
- Start weighing function

Hardware platform

The application example is available for various SIMOTION hardware platforms and is designed for the distributed application of the SIWAREX FTA weighing module.

Note

If the application example is not available for your hardware platform, you must change the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is supplied with SIMOTION SCOUT.

1. Retrieve and open the project containing the application example.
2. Check the hardware configuration: for example, PROFIBUS DP addresses and module addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and the module address in the program.
   If necessary, adapt your example project to the available hardware (see "Adapting the application example").
4. **Save** and **compile** your example project. You can now download the example to the SIMOTION device and switch to **RUN** mode.

Adapting the application example

The configuration in the example and your available hardware must be compatible.

The following options are available:

1. You can adapt the configuration of the example to the available hardware.
2. You can adapt the configuration of the hardware to the example.

Configuration of the example

You have assigned a module address <> 256 in **HW Config**.

1. In the symbol browser, change the I/O address to the value you assigned in **HW Config**:

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperiin	PIB 272		Array	16
2	⊞ myperiout	PQB 272	☐	Array	16

Figure 8-49 Adapting the I/O address

2. Enter the same value in the **myModuleAddr** variable of the application example.

8.8.5.2 Sequence of the application example

Command processing

Proceed as follows to start the command processing in the example program:

1. Open the symbol browser in SIMOTION SCOUT.
2. To select a command, set the **myCommand** parameter in the symbol browser to the command to be executed.
3. Define the command-specific parameters.
4. Start the command by setting the **myCmdExecute** parameter to **TRUE**.
5. Wait until an error-free command execution is indicated with **TRUE** on the **done** output parameter of the function block. An error in the command execution is indicated with **error** = **TRUE**.
6. If the command is executed successfully, the value of data record x (x = number of corresponding data record) is displayed on the **myFTAInOut.scaledata.DRx**in-out parameter.

Reading a data record

Proceed as follows to read a data record:

1. Set the **myCommand** parameter to **CMD_READ_SINGLE_DR**.
2. To select the data record to be read, set the **myFTACommand** parameter to **2xx** (read single data record, xx = data record number), e.g. **myFTACommand := 220** (read setpoint weight).
3. Set the **myCmdExecute** parameter to **TRUE**.
   This initiates the reading of the data record from the SIWAREX FTA. You can find the read data in the **scaleData.DRx** data structure (xx = data record number).

Writing a data record

Proceed as follows to write a data record:

1. Write the values to be transferred to the **scaleData.DRx** data structure (xx = data record number).
2. Set the **myCommand** parameter to **CMD_WRITE_SINGLE_DR**.
3. To select the data record to be written, set the**myFTACommand** parameter to **4xx** (read single data record, xx = data record number), e.g. **myFTACommand := 420** (write setpoint weight).
4. Set the **myCmdExecute** parameter to **TRUE**.
   This initiates the transfer of the data record to the SIWAREX FTA.

Start weighing function

Proceed as follows to start the weighing function:

1. Set the **myCommand** parameter to **CMD_START_WEIGHING**.
2. Start the weighing function by setting the **myCmdExecute** parameter to **TRUE** .

8.8.6 Alarm processing

8.8.6.1 Overview of alarm processing

Pending error messages are processed and evaluated differently than in a SIMATIC system. By default, diagnostic and hardware interrupts are not activated.
Activate the alarms for the weighing module in the hardware configuration (see section Integrating the weighing module in a SIMOTION project (Page 6359)).

Once you have defined parameters for the process and/or diagnostic alarms, you can refer to the flow chart below to program the alarm processing sequence.

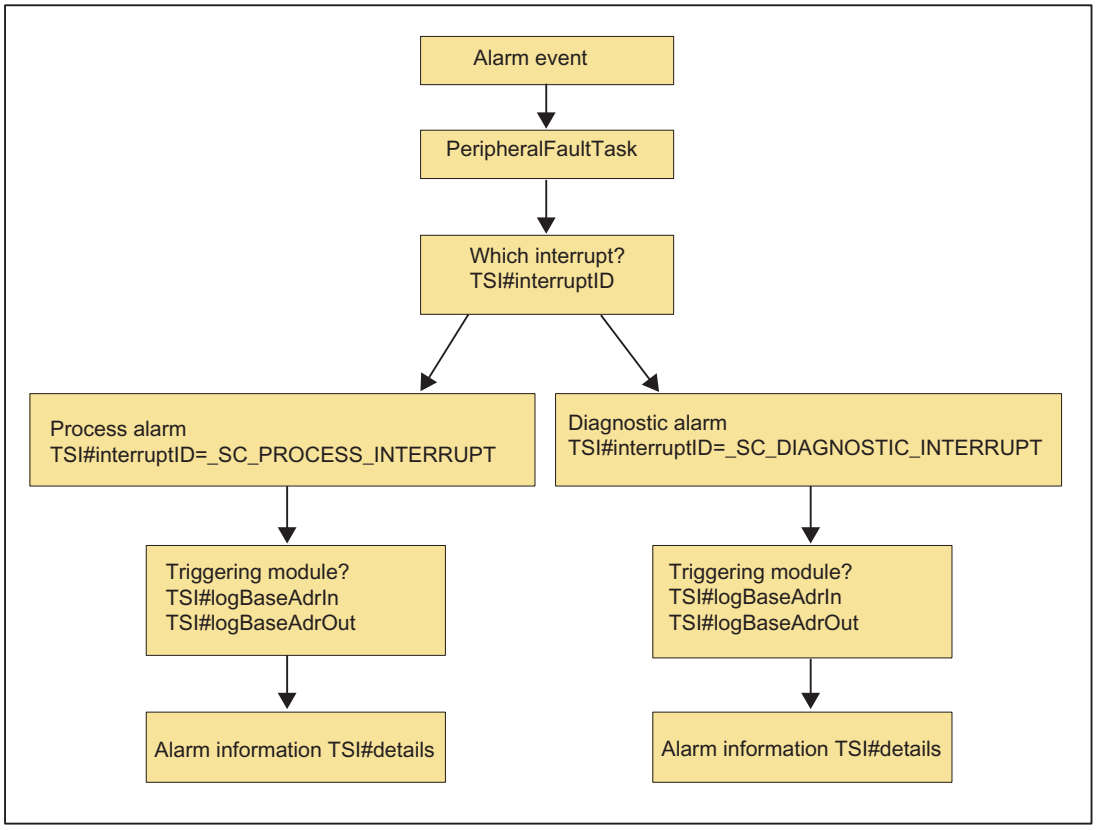


Figure 8-50 Alarm processing for the weighing module

Note

The application example contains an example for programming the alarm processing.

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

Alarm evaluation

Alarms reported by the I/O are evaluated in the **PeripheralFaultTask**. When the **PeripheralFaultTask** is started, the **Taskstartinfo** is made available, which you can evaluate in the user program.

The **Taskstartinfo** (TSI) of the **PeripheralFaultTask** is comparable to the local data of OB 40 and OB 82 in the SIMATIC system.

Table 8-123 Meaning of the Taskstartinfo

Task	TSI		Definition
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical base address if a process alarm (PRAL) or a diagnostic interrupt (DAL) was caused by an input area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical base address if a process alarm (PRAL) or a diagnostic interrupt (DAL) was caused by an output area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the alarm was caused by a station failure or recovery of the associated DP slave, otherwise _SC_INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

8.8.6.2 Process alarms

On the SIWAREX FTA weighing module, you can specify which events should trigger a process alarm. You can use the process alarm to very flexibly respond to technological events and the process status of the SIWAREX FTA weighing module.

Parameters for hardware interrupts are assigned in the **Struct_FTA_DR7** data structure, see section Data structures (Page 6370).

Defining a process alarm

If an event outside of the SIMOTION device cycle requires a response, the weighing module can trigger a process alarm.

Events triggering a process alarm

The criteria (events) for triggering process alarms in a SIMOTION system correspond to those in a SIMATIC system. For more information, refer to the *SIWAREX FTA Weighing Electronics for Automatic Scale* Manual.

Responses to a process alarm

If a process alarm is issued, the following occurs:

- Process data are written to the **TSI#details** variables in the Taskstartinfo of **PeripheralFaultTask**.
- The SIMOTION device goes into STOP mode if a program has not been assigned in the **PeripheralFaultTask**.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

For more information, refer to the *SIWAREX FTA Weighing Electronics for Automatic Scale* Manual.

Evaluating process alarms

When a process alarm is triggered, the data in the **PeriperalFaultTask** Taskstartinfo are written to the **TSI#details** variables and made available to you.

Note

If an event that should trigger a process alarm occurs, and the same previous event is not yet acknowledged, a new alarm is not triggered. The new process alarm is lost.

Depending on the parameter definition, this can lead to the "Lost process alarm" diagnostic alarm.

8.8.6.3 Diagnostic alarms

Diagnostic alarms enable operating messages (hardware faults) of the SIWAREX FTA to be detected in the SIMOTION device.

Defining a diagnostic alarm

If the user program should respond to an internal or external error, you can define a diagnostic alarm that will interrupt the cyclic program of the SIMOTION device.

Events triggering a diagnostic alarm

The criteria (events) for triggering diagnostic alarms in a SIMOTION system correspond to those in a SIMATIC system. For more information, refer to the *SIWAREX FTA Weighing Electronics for Automatic Scale* Manual.

Responses to a diagnostic alarm

If a diagnostic alarm is issued, the following occurs:

- Diagnostic data is written to the **TSI#details** variables in the Taskstartinfo of **PeripheralFaultTask**.
- All programs which are assigned in the **PeripheralFaultTask** are called one after another. The SIMOTION device goes into STOP mode if a program has not been assigned in the **PeripheralFaultTask**.
- The group error LED (SF) lights up on the weighing module. The group error LED (SF) is extinguished as soon as the error has been corrected.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

For more information, refer to the *SIWAREX FTA Weighing Electronics for Automatic Scale* Manual.

Evaluating diagnostic alarms

If a diagnostic alarm is triggered, the data in the Taskstartinfo of the **PeripheralFaultTask** are written to the **TSI#details** variables. The **PeripheralFaultTask** is called automatically and provides 4 bytes of diagnostic information for fast analysis. The diagnostic information content is the same as in a SIMATIC system.

8.8.7 Appendix

8.8.7.1 Selection list for process values

The I/O interface of the SIWAREX FTA weighing module is available for fast output of a process value to the SIMOTION device. The process value selection is defined in the **indexProcessValue1** and **indexProcessValue2** parameters of the **Struct_FTA_DR7** data structure.

Process values are selected from the following list:

Table 8-124 Selection list for process values

Selection number (hexadecimal)		Process value
0	16#00	NAWI status
1	16#01	Gross weight (process value)
2	16#02	Net weight (process value)
3	16#03	Tare (process value)
4	16#04	Gross/net weight (numerical increment with calibration capability)

Selection number (hexadecimal)		Process value
5	16#05	Gross/net weight (numerical increment with calibration capability x 10)
6	16#06	Tare (with calibration capability)
7	16#07	Pulse counter value
8	16#08	Temperature
9	16#09	Fault condition (32-bit information)
10	16#0A	Unfiltered ADC value
11	16#0B	Filtered ADC value (for process value)
12	16#0C	Flow rate per second
30	16#1E	AWI status
31	16#1F	Totalizing memory 1
32	16#20	Totalizing memory 2
33	16#21	Total number of dosages
34	16#22	Number of check weighing operations
35	16#23	Number of TO2-class dosages
36	16#24	Number of TO1-class dosages
37	16#25	Number of dosages in the "good" class (within the range TU1 - TO1)
38	16#26	Number of TU1-class dosages
39	16#27	Number of TU2-class dosages
40	16#28	Number of incorrect dosages
41	16#29	Reserve0
42	16#2A	Setpoint
43	16#2B	Average value of actual weight
44	16#2C	Standard deviation
45	16#2D	Last actual weight
46	16#2E	Weighing operations per hour
47	16#2F	Throughput per hour
48	16#30	Actual in-flight weight
49	16#31	Actual fine weight
50	16#32	Filtered ADC value (for coarse and fine signals)
51	16#33	Residual amount
52	16#34	Current load setpoint

See also

Struct_FTA_DR7 (Page 6376)

8.8.7.2 Command groups and commands

Command groups

The commands of the SIWAREX FTA weighing module are divided into groups. Commands are assigned to a group based on their functional similarity.

The table below provides an overview of the command groups.

Table 8-125 Command group overview

Command group	Definition
1...199	These commands are passed on to the module without reading or writing data records (scale commands, weighing commands, logging commands). Refer to the "Command list" for the meaning of the command numbers.
203...247...399	Read a data record 3 to 47. The numbers 248 to 399 are reserved (use not permitted).
403...447...599	Write a data record 3 to 47. The numbers 448 to 599 are reserved (use not permitted).
601...699	Range of composite commands. The **_FTA_control** function block can transfer multiple data records in succession. Refer to the "Command list" for the meaning of the command numbers.

Command list

The table below lists the commands and their meaning.

Table 8-126 Command list

Command	Meaning of command
Service and calibration commands	
1	Switch on service mode The SIWAREX must be switched to service mode for calibration to be performed. A non-calibrated scale cannot switch out of service mode.
2	Switch off service mode Service mode can be switched off when calibration is complete. Only then can the scale accept weighing commands.
3	Zero point valid calibration command The start of the characteristic curve – zero point of the scale – is defined with the current dead load.
4	Calibration weight 1 valid calibration command The first calibration weight is assigned to the current weight.
5	Calibration weight 2 valid calibration command The second calibration weight is assigned to the current weight.
6	Calibration weight 3 valid calibration command The third calibration weight is assigned to the current weight.
7	Calibration weight 4 valid calibration command The fourth calibration weight is assigned to the current weight.
8	Assign default values to all data records All parameters are set to the delivery condition.

Com-mand	Meaning of command
9	Acknowledge error Fault conditions and fatal system errors that have caused a system restart are acknowledged. The error status is abandoned if no other operating errors are pending.
10	Run impedance check The resistance of the load cells is measured and compared with the stored impedance reference value.
11	Determine impedance reference value The resistance of the load cells is determined and stored as a reference value for future impedance checks.
12	Switch on stand-alone mode This command enables stand-alone mode. This command has a storing effect, i.e., the module remains in stand-alone mode after the power supply is cycled off and back on. Stand-alone mode is deactivated automatically as soon as the module is operated with a higher-level programmable controller.
13	Switch off stand-alone mode This command disables stand-alone mode. Stand-alone mode is automatically deactivated as soon as the module is operated with a higher-level programmable controller.
14	Min/max pointer reset This command resets the **lastMaxWeight** parameter to 0 (see Struct_FTA_DR26 (Page 6389)).
15	Characteristic curve shift This command shifts the entire characteristic curve (calibration coordinates) such that the currently filtered digital value from the ADC becomes a new zero point. Only permitted if the calibration switch is not set. If this shift results in an illegal digital value for a calibration coordinate, the command is rejected.
Scale commands	
21	Zero scale The current weight is set to 0. Subject to restrictions (-1 %, +3 %) during operation with calibration capability ("OIML"). The tare is deleted at the same time.
22	Tare The current weight is set to 0 and the weight display is designated at the same time as "Net" and "Tare".
23	Delete tare The tare is deleted. The current weight is displayed, the designation "Net" is changed to "Gross", and the designation "Tare" or "Preset tare" is reset.
24	Accept tare entry The tare entered is accepted as tare, and at the same time, "Preset tare" is designated along with the weight display.
25	Switch on increased resolution Activates the output/display of the weight value with calibration capability at an increased resolution for 5 s.
26	Display tare weight Activates the output/display of the tare value for 5 s.
29	Display firmware Activates output/display of the firmware version in data record 34 for 5 s.

Com-mand	Meaning of command
Log commands	
31	Output log text 1 The log is output with text layout 1.
32	Output log text 2 The log is output with text layout 2.
33	Output log text 3 The log is output with text layout 3.
34	Output log text 4 The log is output with text layout 4.
35	Repeat last logging operation The last log output is repeated.
Digital weighing commands	
40	"Send digital load cell" On A command is output to the digital load cell to send the weight values.
41	"Send digital load cell" Off Command to the digital load cell to stop sending weight values
Micro memory commands	
70	Start recording Starts the recording (trace function).
71	End current recording Ends the activated recording (trace function).
72	Delete logs in MMC Deletes logs stored on the micro memory card.
73	Delete recording (trace) in MMC Deletes recordings (trace function) stored on the micro memory card.
74	Delete recording (trace) in RAM Deletes recordings (trace function) stored in the RAM memory.
75	Format MMC Formats the micro memory card according to the parameters specified in the module data. Deletes all content stored on the MMC.
76	Delete all MMC data (log data, measurement data, etc.) Deletes data stored on the micro memory card.
77	Trace single recording This command initiates a single recording of a trace element. The time between two trace recordings can be defined in any interval. With trace to MMC, new trace commands initiate a new trace recording only after 50 ms. With trace to RAM, the recording is performed within 10 ms, maximum.
Scale commands	

Command	Meaning of command
100	Start weighing operation with tare/zeroing mode A weighing operation with zeroing or taring is started according to the tare/zeroing mode setting.
101	Start weighing operation without tare/zeroing mode A weighing operation is started without zeroing or taring. This command is only permitted when operation does not require calibration (country code is not "OIML").
102	Start weighing operation in continuous mode Continuous start of sequential weighing cycles. This command is only available for AWI mode.
103	Continue weighing cycle This command is used to continue weighing as well as emptying operations.
104	Continue weighing in JOG mode Weighing is continued in JOG mode (pulse/pause) only.
105	Stop during weighing cycle The weighing or emptying cycle is stopped immediately, the scale stops and remains in "stopped" mode. Possible subsequent commands: 103 – Continue weighing cycle 104 – Continue weighing in pulse mode 108 – Abort 110 – Residual weighing
106	End continuous operation The current weighing operation is completed, after which continuous operation is ended.
107	Activate check stop The scale is stopped at the next defined step for check stop. Continued with: 103 – Continue weighing cycle 104 – Continue weighing in pulse mode 108 – Abort 110 – Residual weighing
108	Abort The stopped weighing cycle is ended with no further activity. Automatic emptying is not performed. The previous weight is not taken into account in the totalizing calculation.
109	Empty On This command activates the emptying signal in idle state (no active weighing). If the emptying time is 0, the emptying signal remains activated until the weight is in the empty range. Independently of this, the emptying operation can be terminated with command 118.
110	Residual weighing A weighing cycle in progress is stopped if necessary, and the emptying operation is started immediately. The current weight is balanced before emptying. In certain cases, continuous operation is ended.
111	Activate inhibit time for setpoint-actual comparison The setpoint-actual comparison is not performed for a defined period of time. Command 112 can be used to shorten this defined time, thus immediately ending the inhibit time for the setpoint-actual comparison.
112	Stop inhibit time for setpoint-actual comparison The activated inhibit time is stopped early.
113	Logging and deleting calibration-capable totalizing memory 1 This command is only executed if the log text contains total 1.

Com-mand	Meaning of command
114	Delete totalizing memory 1 Permitted only if the country code is not "OIML".
115	Delete totalizing memory 2 Totalizing memory 2 can be deleted at any time.
116	Delete statistical data Deletes statistical data, with the exception of the totalizing memories.
117	Output totalizing memory 1 for 5 s The content of totalizing memory 1 is output instead of the calibration-capable weight value. When in SWT operating mode, the totalizing memory 1 is displayed permanently. In the other operating modes, it automatically switches back to the weight with calibration capability after 5 s. Command 125 also re-activates the weight with calibration capability.
118	Empty Off The emptying operation initiated with command 109 is terminated immediately.
121	Start weighing operation with/without taring/zeroing A single start with/without tare/check is to be performed. The counters for the tare and check cycle run in the background (as in continuous mode) and determine whether the scale should be tared or, in certain cases, a check should be performed.
122	Execute taring/zeroing The scale is to be tared/zeroed the next time it is started. After this is performed, the taring-cycle counter is reset.
123	Check follows After the check is performed, the counter is reset, but only restarted after being adjusted to the control optimum.
124	Residual emptying Continuous operation is ended and residual emptying performed. The emptied residual amount is not logged.
125	Display weight with calibration capability The content of weight with calibration capability is output instead of the totalizing memory 1.
126	Display log values with calibration capability Display the log values with calibration capability (log ID, gross or net, tare, total 1 and setpoint) one after another in a 4s cycle and then switches back to the original value. (cannot currently only be used via SIWATOOL)
127	Residual weighing without emptying Emptying is stopped. The amount emptied so far is added to the current total 1. The cycle is emptied without any more emptying.
Composite commands	
601	Read DR30 and DR31
602	Read DR34 and DR35
610	Read DR20 and DR22
649	Read all data records from the SIWAREX FTA: DR3, DR4, DR7, DR8, DR9, DR15, DR16, DR17, DR18, DR20, DR21, DR22, DR23, DR30, DR31, DR32, DR34, DR35, DR44, DR45, DR46, DR47)
651	Write scale data 1 (DR22) and setpoint weight (DR20) to the SIWAREX FTA and the start the weighing operation with command 100 (Start weighing operation with tare/zeroing mode)

Com-mand	Meaning of command
652	Write scale data 1 (DR22) and load quantity (DR21) to the SIWAREX FTA and the start the weighing operation with command 100 (Start weighing operation with tare/zeroing mode)
653	Write scale data 1 (DR22) and setpoint weight (DR20) to the SIWAREX FTA and the start the weighing operation with command 102 (Start weighing operation in continuous mode with tare/zeroing mode)
654	Write scale data 1 (DR22) and load quantity (DR21) to the SIWAREX FTA and the start the weighing operation with command 102 (Start weighing operation in continuous mode with tare/zeroing mode)
660	Write DR20 and DR22
699	Write the following data records to the SIWAREX FTA: DR3, DR4, DR7, DR8, DR15, DR18, DR21, DR22, DR23, DR45, DR46

8.8.7.3 SIMOTION and SIMATIC names

The following table contains a comparison of the SIMOTION and the SIMATIC names.

Table 8-127 SIMOTION and SIMATIC names for the SIWAREX FTA weighing module

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	Name in the SIMATIC system
Function block parameter	
execute	AUT_CMDEN
cmdNumber	AUT_CMD
ackErrorFTA	ERR_MSG_Q
periIn	-
moduleAddress	LADDR
simValue	SIM_VAL
setAO	ANA_OUT
forceDO	DO_FORCE
transitions	TRANSITION
scaleData	UDT12
periOut	-
done	CMD_FOK
busy	CMD_INPR
error	FB_ERR
errorID	FB_ERR_C
errorIdTransfer	-
errorIdCommand	CMD_ERR_C
errorFTA	ERR_MSG
errorTypeFTA	ERR_MSG_TYPE
errorIdFTA	ERR_MSG_C
cntrRefresh	REF_COUNT
val1Process	PROC_VAL1

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
val2Process	PROC_VAL2
scaleStatus	SC_STATUS
startup	START_UP
Data structure elements	
Struct_FTA_DR3	**DR3**
calibDigits0	CAL_D0_M
calibDigits1	CAL_D1_M
calibDigits2	CAL_D2_M
calibDigits3	CAL_D3_M
calibDigits4	CAL_D4_M
calibWeight1	CAL_W1_M
calibWeight2	CAL_W2_M
calibWeight3	CAL_W3_M
calibWeight4	CAL_W4_M
sigRange	SI_RNG_M
filtSequence	F_PARA_M
filtType	F_TYPS_M
filtCutOffFreq	F_FRQS_M
filtDepth	F_DEPTH_M
scaleId	SC_ID_M
numRanges	RNG_M
scaleType	TYPE_RNG_M
setZeroPwrOn	Z_P_ON_M
setZeroTare	Z_P_ON_TARA_M
setZeroAutomatic	Z_AUTO_M
setTareType	bo_ADD_TARE
minWeightRange1	MIN_WR1_M
maxWeightRange1	MAX_WR1_M
incRange1	INC_WR1_M
minWeightRange2	MIN_WR2_M
maxWeightRange2	MAX_WR2_M
incRange2	INC_WR2_M
minWeightRange3	MIN_WR3_M
maxWeightRange3	MAX_WR3_M
incRange3	INC_WR3_M
standStillTime1	T_STILL1_M
standStillWeight1	W_STILL1_M
timeOutStandStill1	T_WAIT_STILL1_M
maxPosWeightPwrOn	PON_Z_POS_M
minNegWeightPwrOn	PON_Z_NEG_M
maxPosWeightZero	Z_POS_V_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
minNegWeightZero	Z_NEG_V_M
maxTare	TARA_MAX_M
setLCType	TYP_LC_M
timeOutDigLC	MON_T_M
setRestriction	LEG_TRADE_M
weightUnit	W_UNIT_M
standStillWeight2	W_STILL2_M
standStillTime2	T_STILL2_M
timeOutStandStill2	MAX_T_STILL2_M
standStillWeight3	W_STILL3_M
standStillTime3	T_STILL3_M
timeOutStandStill3	MIN_T_STILL3_M
minDosingValTot	MIN_V_TOT_M
incTotalizing	INC_TOT_M
xxxReserve0	Res303_M
xxxReserve1	Res403_M
xxxReserve2	Res503_M
Struct_FTA_DR4	
scaleMode	SC_TYPE_M04
xxxReserve0	Res104_M
xxxReserve1	Res204_M
timeOutLogOutput	T_OUT_PR_M
selectLogOutput	PROT_PARA_M
xxxReserve2	Res304_M
baseWeightLimit1	LIMIT1_M
baseWeightLimit2	LIMIT2_M
xxxReserve3	Res404_M
baseEmptyRange	EMPTY_GN_M
xxxReserve4	b_Reserve4
emptyRange	EMPTY_RNG_M
limit1On	LIM1_ON_M
limit1Off	LIM1_OFF_M
limit2On	LIM2_ON_M
limit2Off	LIM2_OFF_M
limit3On	LIM3_ON_M
limit3Off	LIM3_OFF_M
minFlowLimit1	MIN_FL1_M
minFlowLimit2	MIN_FL2_M
filtDepthMinFlow	MIN_F_D_FL_M
xxxReserve5	b_Reserve5

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	Name in the SIMATIC system
Struct_FTA_DR7	
xxxReserve0	-
srcWeightSim	SIM_SRC_W_M
deciDigitsProcVal	DECPNT_M
xxxReserve1	Res107_M
enableForceDO	FRC_SERV_EN_M
indexProcessValue1	PROC_V1_M
indexProcessValue2	PROC_V2_M
xxxReserve2	Res207_M
defProcessAlarm0	PR_AL0_M
defProcessAlarm1	PR_AL1_M
defProcessAlarm2	PR_AL2_M
defProcessAlarm3	PR_AL3_M
defProcessAlarm4	PR_AL4_M
defProcessAlarm5	PR_AL5_M
defProcessAlarm6	PR_AL6_M
defProcessAlarm7	PR_AL7_M
timeOutLifeBit	S7_LB_M
weightAnaOutZero	AO_ZERO_M
weightAnaOutEnd	AO_END_M
weightAnaOutOD	AO_CST_M
srcAnaOut	AO_SRC_M
rangeAnaOut	AO4_20_M
printerBdRate	PRT_BD_M
setXonXoff	RS232XONOFF_M
setRtsCts	RS232RTSCTS_M
setRS485Prot	RS485_PROT_M
digitsRemDisplay	DECPNT_D_M
rs485BdRate	RS485_BD_M
rs485Parity	RS485_PAR_M
rs485NumDataBits	RS485_DATA_M
rs485NumStopBits	RS485_STOP_M
defDO1	DOF1_M
defDO2	DOF2_M
defDO3	DOF3_M
defDO4	DOF4_M
defDO5	DOF5_M
defDO6	DOF6_M
defDO7	DOF7_M
defDO8	DOF8_M
lowActiveDO1	DO_HL_A1_M
lowActiveDO2	DO_HL_A2_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
lowActiveDO3	DO_HL_A3_M
lowActiveDO4	DO_HL_A4_M
lowActiveDO5	DO_HL_A5_M
lowActiveDO6	DO_HL_A6_M
lowActiveDO7	DO_HL_A7_M
lowActiveDO8	DO_HL_A8_M
DO1onOD	DO_BY_E1_M
DO2onOD	DO_BY_E2_M
DO3onOD	DO_BY_E3_M
DO4onOD	DO_BY_E4_M
DO5onOD	DO_BY_E5_M
DO6onOD	DO_BY_E6_M
DO7onOD	DO_BY_E7_M
DO8onOD	DO_BY_E8_M
enableDOonError	DO_BY_E_EN_M
xxxReserve3	Res407_M
defDI1	DIF1_M
defDI2	DIF2_M
defDI3	DIF3_M
defDI4	DIF4_M
defDI5	DIF5_M
defDI6	DIF6_M
defDI7	DIF7_M
lowActiveDI1	DI_HL_A1_M
lowActiveDI2	DI_HL_A2_M
lowActiveDI3	DI_HL_A3_M
lowActiveDI4	DI_HL_A4_M
lowActiveDI5	DI_HL_A5_M
lowActiveDI6	DI_HL_A6_M
lowActiveDI7	DI_HL_A7_M
measTimeForCntr	CNT_T_M
xxxReserve4	Res507_M
modeLogOverflow	MMC_PR_OWR_M
modeTraceOverflow	MMC_TR_OWR_M
traceDataToMMC	MMC_RAM_TR_M
traceSizeMMC	MMC_TR_S_M
logSizeMMC	MMC_PR_S_M
traceCycle	MMC_TR_CYC_M
Struct_FTA_DR8	
dateAndTime	DT_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	Name in the SIMATIC system
Struct_FTA_DR9	
crcCheckSumFw	CRC_CH_M
lenFw	LENGTH_M
moduleInfo	COPYRT_M
moduleName	MOD_NAME_M
application	APPL_ID_M
fileName	FILE_NAME_M
typeVersion	A_VER_M
fctVersion	A_F_VER_M
dataStructVersion	A_DR_VER_M
corrVersion	A_VER_NO_M
dateCreation	CREAT_D_M
timeCreation	CREAT_T_M
bootVersion	VER_BOOT_M
scaleType	SC_TYPE_M9
xxxReserve1	w_Reserve
Struct_FTA_DR15	
tareSetValue	TARE_V_M
Struct_FTA_DR16	
weightSimValue	SIM_V_M
Struct_FTA_DR17	
setAO	AO_V_M17
Struct_FTA_DR18	
setValRemDisplay	DISP_V_ADD_M
Struct_FTA_DR20	
dosingSetpoint	SP_V_M
Struct_FTA_DR21	
loadingSetpoint	SP_LOAD_V_M
Struct_FTA_DR22	
maxDosingTime	MAX_DOS_T_M
inFlightWeight	IN_FL_V_M
fineWeight	FINE_V_M
switchOffCorr	COMP_V_M
timePreDosing	T_PREDOS_M
upperTolValTO1	TO1_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
lowerTolValTU1	TU1_M
upperTolValTO2	TO2_M
lowerTolValTU2	TU2_M
Struct_FTA_DR23	
selectText	TXTNO_A_M
xxxReserve1	Res123_M
xxxReserve2	Res223_M
maxSetpointDosing	MAX_SP_UNLD_M
disableTimeCoarse	DIS_COARSE_M
disableTimeFine	DIS_FINE_M
disableTimeCompare	DIS_COMPARE_M
valAnaOutCoarse	COARSE_AO_V_M
valueAnaOutFine	FINE_AO_V_M
filtTypeDosing	F_TYPE_D_M
filtCutOffFreq	F_FREQ_D_M
modeZeroTare	TARA_Z_PROG_M
cycleZeroTara	TARA_Z_CYC_M
xxxReserve3	Res323_M
minTareValue	TARA_MIN_V_M
maxTareValue	TARA_MAX_V_M
timeAutoZeroing	T_FOR_Z_M
waitDI1InStepX	W_DI0_STEP_N_M
waitDI2InStepX	W_DI1_STEP_N_M
waitDI3InStepX	W_DI2_STEP_N_M
waitDI4InStepX	W_DI3_STEP_N_M
waitDI5InStepX	W_DI4_STEP_N_M
waitDI6InStepX	W_DI5_STEP_N_M
waitDI7InStepX	W_DI6_STEP_N_M
xxxReserve4	Res423_M
timeOutOneStep	T_ONE_STEP_M
stopAfterStep1	CH_STOP_STEP1_M
stopAfterStep2	CH_STOP_STEP2_M
stopAfterStep3	CH_STOP_STEP3_M
stopAfterStep4	CH_STOP_STEP4_M
stopAfterStep5	CH_STOP_STEP5_M
stopAfterStep6	CH_STOP_STEP6_M
stopAfterStep7	CH_STOP_STEP7_M
xxxReserve5	Res523_M
autoPostDosing	AUTO_AFTER_DOS_M
modePostDosing	AFTER_DOS_METH_M
stopTO1Limit	TO1_STOP_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
stopTO2Limit	TO2_STOP_M
stopTU1Limit	TU1_STOP_M
stopTU2Limit	TU2_STOP_M
continueTolStop	CONT_TOL_M
numNoTolCheck	PER_NOTOL_CH_M
timePulseInching	T_INCH_P_M
ctrlErrReaction	CNTR_R_ERR_M
typeController	CNTR_TYPE_M
factorController	PR_CNTR_F_M
xxxReserve6	Res623_M
limitController	PR_CNTR_LIM_M
optiPlusCtrl	PR_CNTR_OPP_M
optiMinusCtrl	PR_CNTR_OPM_M
setFineTime	MIN_FINE_T_M
factorFineTime	F_T_CNTR_M
xxxReserve7	Res723_M
xxxReserve8	Res823_M
timeOverlap	T_OVLAP_M
timeEmptying	T_EMPTY_M
timeOutEmptying	MAX_T_EMPTY_M
modeLoading	UNLD_COARSE_M
xxxReserve9	Res923_M
Struct_FTA_DR26	
presetTareActive	PR_TARA_M
xxxReserve1	Res126_M
xxxReserve2	Res226_M
internState	-
actTareWeight	TARE_W_P_M
actAverTareWeight	TARE_W_AV_M
pwrOnZeroValue	PWRON_ZV_M
zeroValue	ZV_M
zeroCorrValue	ZV_AUTO_M
impedRefValue	SEN_R_CH_M
actImpedValue	SEN_R_REF_M
lastMaxWeight	MAX_W_MEM_M
numOpMinutes	ON_TIME_M
maxTemperature	TEMP_MAX_M
xxxReserve3	Res326_M
xxxReserve4	Res426_M
sigLevel	i_SIGNAL_LEVEL
crc	CRC_M

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
Struct_FTA_DR30	
stateNAWI	-
weightInRange1	SWR1_O
weightInRange2	SWR2_O
weightInRange3	SWR3_O
limit1On	SLIM1_ON_O
limit2On	SLIM2_ON_O
limit3On	SLIM3_ON_O
scaleTared	STARED_O
scaleTaredManual	STARED_BY_M_O
weightMax9e	SMAX_9E_O
weight025dZero	S025D_Z_O
waitOfStandStill1	SWAIT_STILL1_O
standStill1On	SSTILL1_ON_O
scaleCalibrated	SSC_CAL_O
cmdErrOnDI	SCMDERR_DI_O
simWeighingOn	SSIM_ON_O
serviceModeOn	SSERV_MODE_ON_O
printingOn	SPRT_O
printImpossible	SRS232_BUSY_O
MMCconnected	SMMC_CON_O
MMCready	SMMC_RDY_O
MMCreadyForTrace	SMMC_RDY_F_TR_O
MMCreadyForLog	SMMC_RDY_W_O
MMCTraceActive	SMMC_TR_A_O
minFlowCtrl1On	SMIN_FLOW1_O
minFlowCtrl2On	SMIN_FLOW2_O
scaleEmptyRange	SEMPTY_O
protectionOn	SL_DATA_PROT_O
xxxReserve1	SRes130_O
MMCDataPrepared	SMMC_REA_O
digLCactive	SDIGIT_LC_O
standAloneActive	SST_ALONE_O
errorFTAoccurred	SERR_OC_O
stateAWI	-
dosingStep0	SDOS_STEP0_O
dosingStep1	SDOS_STEP1_O
dosingStep2	SDOS_STEP2_O
dosingStep3	SDOS_STEP3_O
dosingStep4	SDOS_STEP4_O
dosingStep5	SDOS_STEP5_O

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
dosingStep6	SDOS_STEP6_O
dosingStep7	SDOS_STEP7_O
postDosingActive	SAFTER_DOS_O
coarseSigOn	SCOARSE_ON_O
fineSigOn	SFINE_ON_O
timerPreDosingOn	ST_PREDOS_O
emtyingSigOn	SEMPTY_ON_O
weighingStopped	SSTOPPED_O
stoppedForCheck	SCH_STPD_O
checkStopFollow	SCH_STP_FOL_O
dosingAborted	SDOS_CY_ABO_O
nextStepWaiting	SN_STEP_W_O
upperLimit2On	STO2_O
upperLimit1On	STO1_O
toleranceOK	STOL_OK_O
lowerLimit1On	STU1_O
lowerLimit2On	STU2_O
toleranceBad	STOL_BAD_O
standStill2On	SSTILL2_ON_O
standStill3On	SSTILL3_ON_O
checkFollows	SCHECK_F_O
comparatorDisable	SDIS_COMPARA_O
continueModeOn	SCONTI_MODE_DOS_O
xxxReserve2	SRes630_O
endOfDosingCycle	SEND_DOS_CYC_O
endOfCharge	SEND_CHARGE_O
actGrossWeight	SGROS_WGT_O
actNetWeight	SNET_WGT_O
actTareWeight	STARE_WGT_O
weight	SGROS_NET_V_O
weightx10	SGROS_NET_V_10X_O
tareWeight	STARE_V_O
lastCheckedWeight	SLAST_DOS_V_O
pulseCntrValue	SCOUNTER_V_O
sumMem1	STOT_V1_O
sumMem2	STOT_V2_O
Struct_FTA_DR31	
actFlow	M_FLOW_O
actInFlightWeight	ACT_AFTERRUN_V_O
actFineWeight	ACT_FINE_V_O
actAduValue	AO_V_O31

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	Name in the SIMATIC system
actAduValueFilt1	ACT_DIG_FS_O
actAduValueFilt2	ACT_DIG_FD_O
actRestLoading	REST_WGT_O
actSetpointLoading	ACT_SP_UNLD_O
actStateError	ACT_ERR_SERV_O
actDateAndTime	ACT_DT_O
actTemperature	ACT_TEMP_O
actStateDI	ACT_DI_O
actStateLC	STAT_DI_LC_O
measImpedance	SEN_RES_REF_O
actImpedance	SEN_RES_CH_O
Struct_FTA_DR32	
totalNumWeighing	CNT_CYC_TOT_O
numCheckedWeighing	CNT_CH_CYC_O
numTO2Weighing	CNT_TO2_EX_O
numTO1Weighing	CNT_TO1_BAND_O
numGoodWeighing	CNT_TOL_OK_O
numTU1Weighing	CNT_TU1_BAND_O
numTU2Weighing	CNT_TU2_BAND_O
numBadWeighing	CNT_TOL_BAD_O
xxxReserve1	Res132_O
xxxReserve2	Res133_O
actSetpointWeight	ACT_SP_O
averValue	ACT_AV_V_O
standardDeviation	STD_DEV_O
thruputPerHour	THRU_PER_H_O
weighingPerHour	CYC_PER_H_O
Struct_FTA_DR34	
actWeightASCII	ASCII_WGT_O
Struct_FTA_DR35	
cryptoData	DATAx_O
Struct_FTA_DR39	
typeVersion	c_OCX_VERSION_DESIG
xxxReserve1	b_Reserve_39_1
versionPrimary	i_OCX_VERS_NUMBER_MAIN
versionSecondary	i_OCX_VERS_NUMBER_SUB
Struct_FTA_DR44	

Name in SIMOTION System, V4.1 and higher (command library in SCOUT)	**Name in the SIMATIC system**
MMCId	MMC_ID1_O
xxxReserve1	Res144_O
xxxReserve2	Res244_O
logId	PROT_ID_O
lastLogData	L_PROT_O
Struct_FTA_DR45	
varText1	ADD_TXT1_M
varText2	ADD_TXT2_M
varText3	ADD_TXT3_M
varText4	ADD_TXT4_M
Struct_FTA_DR46	
selectLogId	ACC_ID_PROT_M
reqLastLog	LAST_PROT_SEL_M
xxxReserve1	Res146_M
Struct_FTA_DR47	
MMCId	MMCID1_O
xxxReserve1	Res147_O
xxxReserve2	Res247_O
logId	P_ID_O
logData1	P_DATA1_O
logData2	P_DATA2_O
logData3	P_DATA3_O
logData4	P_DATA4_O
Struct_FTA_DR123	
logId	PRT_PROT_ID_O
MMCId	MMCID1_O
xxxReserve1	Res1123_O
xxxReserve2	Res2123_O
MMCCapacity	MMC_CAP_O
MMCLogCapacity	MMC_CAP_P_O
traceCapacity	CAP_TRACE_O
oldMMCLogId	OID_MMC_P_O
newMMCLogId	NID_MMC_P_O
oldMMCTraceId	OID_MMC_T_O
newMMCTraceId	NID_MMC_T_O
oldRAMTraceId	OID_RAM_T_O
newRAMTraceId	NID_RAM_T_O

8.8.7.4 List of abbreviations

Table 8-128 Abbreviations

Abbreviation	Definition
ADC	Analog-Digital Converter
ADC	Analog-digital converter
AWI	**A**utomatic **W**eighing **I**nstrument
DAL	Diagnostic alarm
DR	Data record
FB	Function block
FTA	**F**lexible **T**echnology, **A**utomatic Weighing Instrument
HW	Hardware
IM	Interface module
IN	Input parameter
IN/OUT	In/out parameter
I/O	Input/Output
LAD	Ladder diagram
LC	Load Cell
MMC	Micro Memory Card
NAWI	**N**on **A**utomatic **W**eighing **I**nstrument
OD	Output disable (response to CPU-STOP)
OIML	International Organization of Legal Metrology
OUT	Output parameter
PRAL	Process alarm
PS	Power supply
RAM	Random Access Memory
SecureOCX	Add-on for WinCC flexible for configuration of the weight display with calibration capability
SW	Software
AWI	Automatic Weighing Instrument (weighing operating mode)
SWE	Automatic scale for single weighing machine (weighing operating mode)
SWT	Automatic scale for loading - totalizing (weighing operating mode)
TO	Upper tolerance limit
TSI	Taskstartinfo
TU	Lower tolerance limit
WB	Weighing range
WinCC flexible	Software for configuring the weight display with calibration capability
Tool	Load cell

8.9 Extension to the command interface for AS-Interface master modules

Introduction

Contents of the function manual

This **document** is part of the **SIMOTION Programming References documentation package**.

Function block

The **_ASI_cmdInterface** function block for the AS-Interface master modules CP 343-2 P, DP/AS-Interface Link 20E, DP/AS-Interface Link Advanced, and IE/AS-Interface Link PN IO is part of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.9.1 Fundamental safety instructions

8.9.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.9.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.9.2 Description

8.9.2.1 General

This chapter describes the general differences and the common features of the AS-Interface master modules CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced, and IE/AS-Interface Link PN IO that result from operating a SIMOTION system (as opposed to a SIMATIC system) for data transfer and command interface operation purposes.

Note

This manual is a supplement to the following SIMATIC manuals:

- *SIMATIC NET CP 343-2/CP 343-2 P AS-Interface Master*
- *SIMATIC NET DP/AS-Interface Link 20E*
- *SIMATIC NET DP/AS-Interface Link Advanced*
- *SIMATIC NET IE/AS-Interface Link PN IO*

These documents are shipped with SIMOTION SCOUT in electronic form.

The following software versions are required for the standard function described in this documentation:

- For AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced):
  - SIMOTION SCOUT V4.0 or higher
  - SIMOTION Kernel V4.0 or higher
- For AS-Interface master IE/AS-Interface Link PN IO:
  - SIMOTION SCOUT, V4.1 SP1 or higher
  - SIMOTION Kernel, V4.1 SP1 or higher

8.9.2.2 Product description

The following modules can be operated using automation systems from both the SIMATIC and SIMOTION ranges:

- CP 343-2 P
- DP/AS-Interface Link 20E
- DP/AS-Interface Link Advanced
- IE/AS-Interface Link PN IO

They permit the connection of an AS-Interface line to the above mentioned automation systems.

You can use the modules to access from the SIMOTION motion control system to the input/output of the AS-Interface slaves. You can access binary or analog values depending on the slave type.

The following AS-Interface slaves can be used:

- Standard slaves
- Slaves with extended address area (extended addressing mode)
- Analog slaves to 7.3/7.4 slave profile

Additionally, for IE/AS-Interface Link PN IO it is possible to use slaves with data transfer mechanisms in accordance with the AS-Interface Specification V3.0-Combined Transaction Type (CTT) 1-5.

Function block

The **_ASI_cmdInterface** function block is provided for operating the command interface of the AS-Interface master CP 343-2 P and DP/AS-Interface Link 20E/Link Advanced with SIMOTION. The function block is described in this manual.

Note

No specific function blocks are required for operating the command interface of the IE/AS-Interface Link PN IO AS-Interface master. Data transfer is performed using the SIMOTION system functions **_readRecord** and **_writeRecord**.

Functionality of the function block and the AS-Interface master CP 343-2 P, DP/AS-Interface Link 20E, and DP/AS-Interface Link Advanced

Although the functionality of the **_ASI_cmdInterface** function block and the AS-Interface master is unchanged when compared with their use in a SIMATIC S7 automation system, the execution of the data transmission and the handling of the FB has been adapted to the given SIMOTION general conditions.

Possible applications

In addition to the applications described in the SIMATIC manuals, you can use the AS-Interface master modules with a SIMOTION system. The CP 343-2 P can be used centrally (on the SIMOTION C2xx only), or the CP 343-2 P and DP/AS-Interface Link 20E/Link Advanced can be used on a distributed basis (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx). The IE/AS-Interface Link PN IO can be operated with all SIMOTION devices that feature a PROFINET interface (e.g. D435 with CBE30).

You can operate several AS-Interface masters with the associated AS-Interface slaves on a single SIMOTION device.

The following figures show how the AS-Interface master can be connected to various SIMOTION devices.

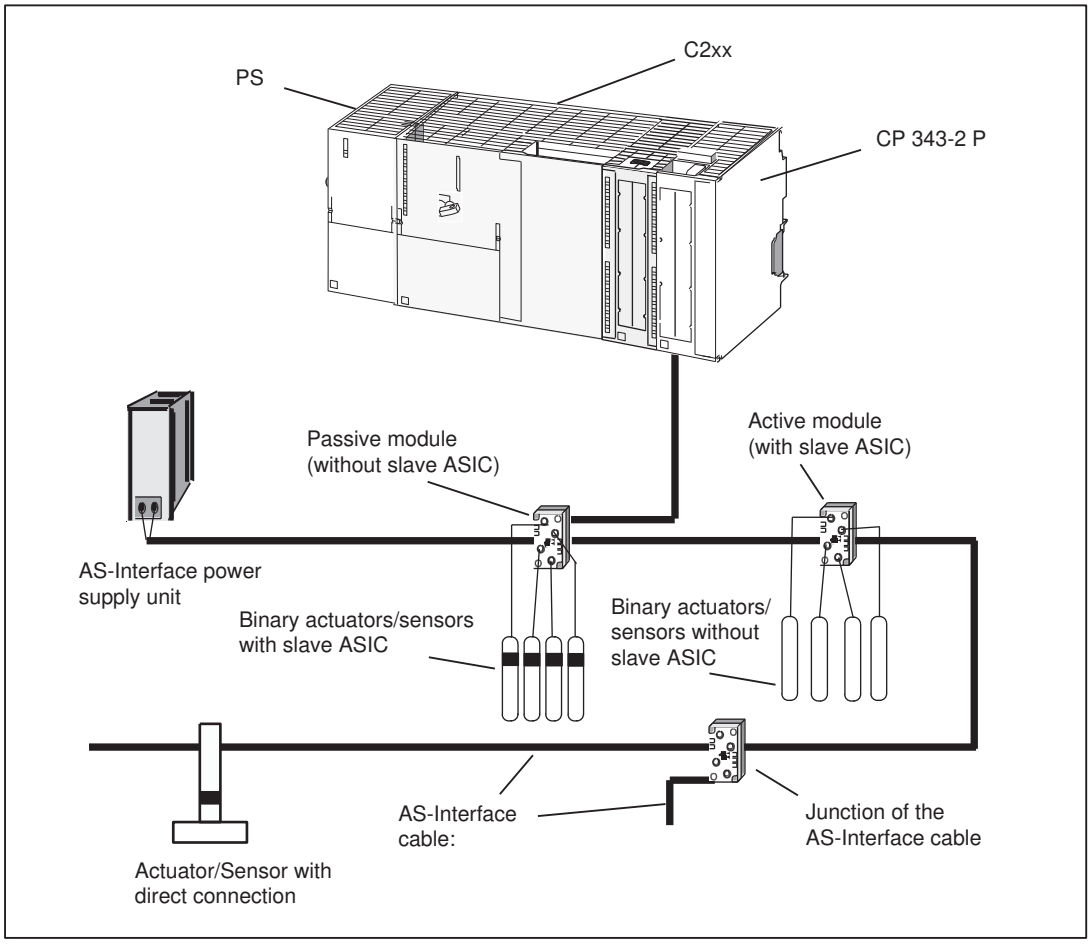


Figure 8-51 Centralized use of the CP 343-2 P AS-Interface master module on the SIMOTION C2xx device

Note

For AS-Interface modules, a differentiation is made between active and passive modules.

- Active modules with slave ASIC:
  It is possible to connect actuators (using a digital or analog output signal) or sensors (using a digital or analog input signal) to active modules.
  This enables actuators and sensors to be interconnected using the AS-Interface.
- Passive modules without slave ASIC:
  These do not have any electronics of their own, but act as distributors and enable connection of the AS-Interface actuators and sensors with integrated slave ASIC (e.g. AS-Interface LED displays).

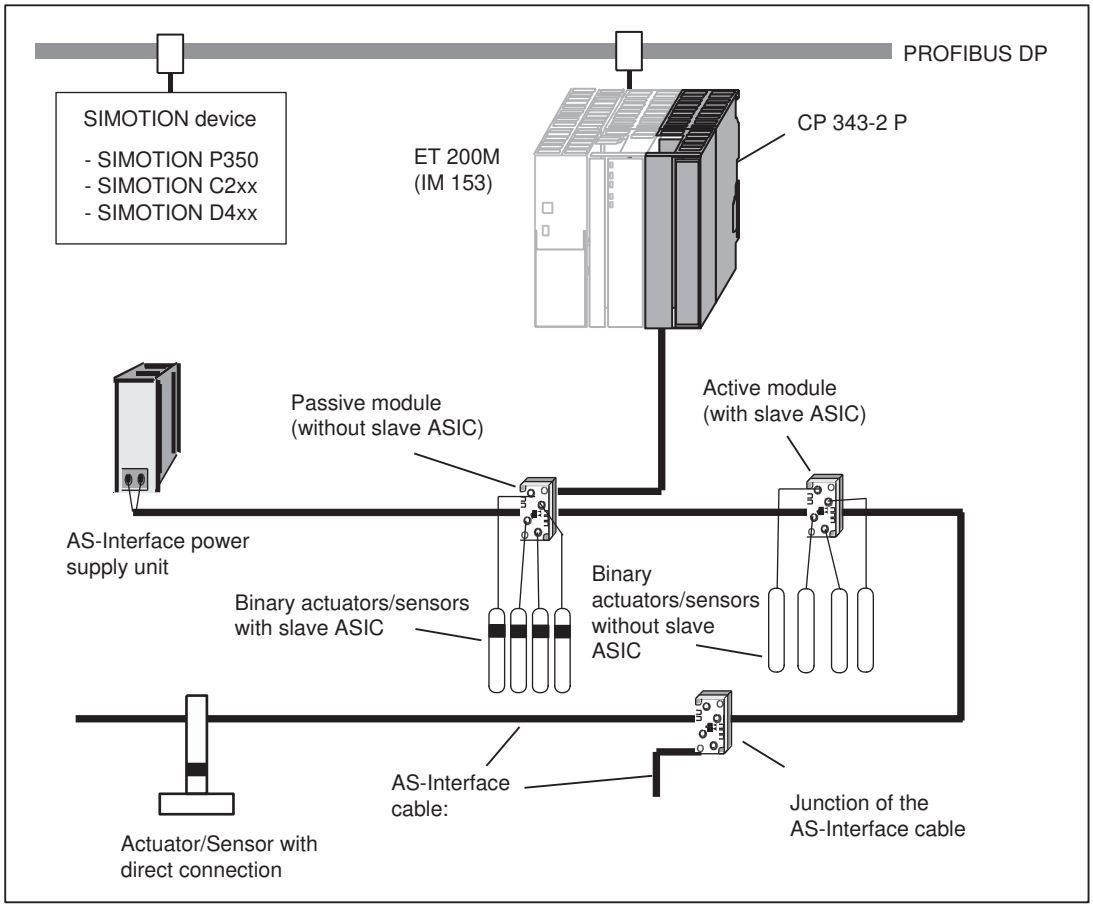


Figure 8-52 Distributed use of the CP 343-2 P AS-Interface master module with ET 200M on a SIMOTION device

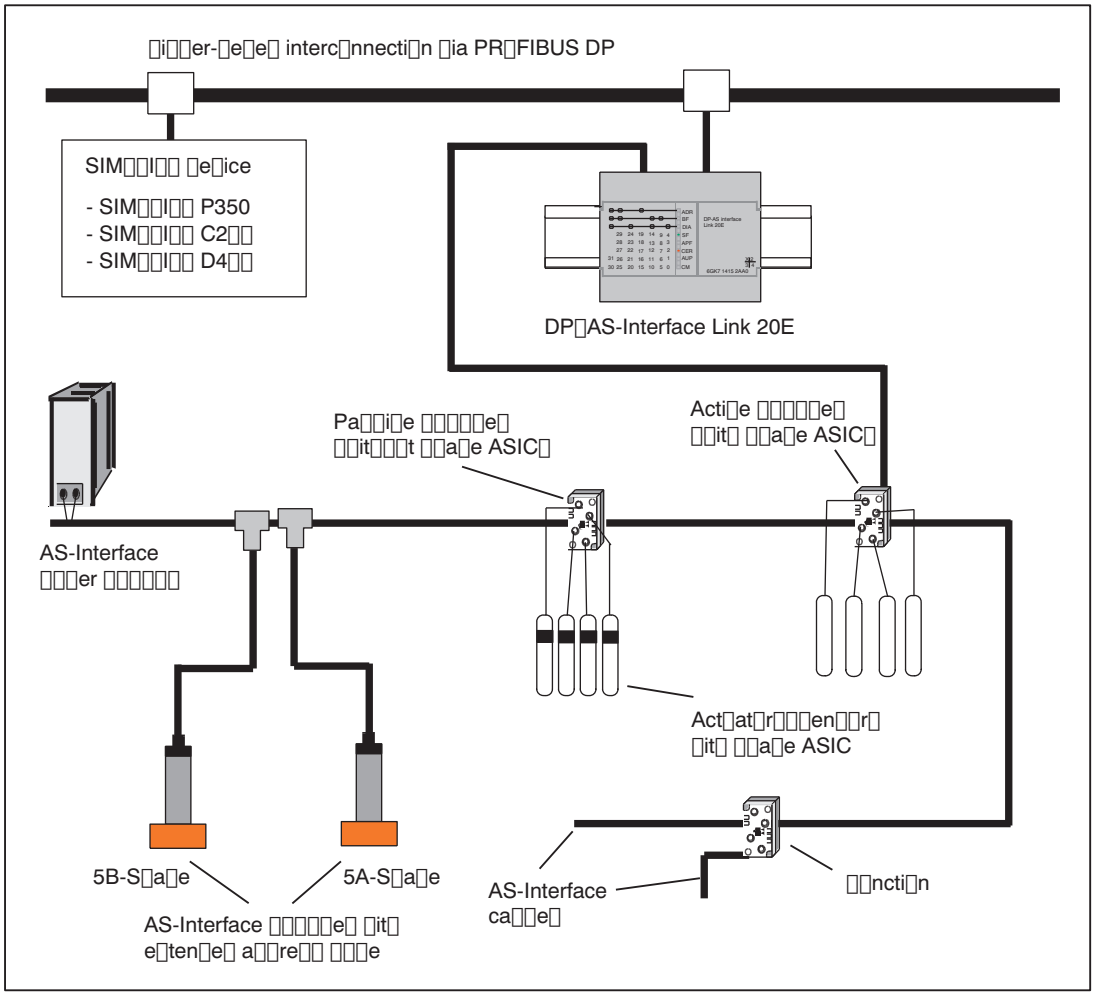


Figure 8-53 Distributed use of the DP/AS-Interface Link 20E AS-Interface master module on a SIMOTION device

Note

AS-Interface A/B slaves use an extended address area. This allows two A/B slaves to be assigned as a pair to an address on the AS-Interface.

The address organization thus allows up to 62 A/B slaves (rather than 31 slaves) to be connected to the AS-Interface.

Addressing is performed on a module-specific basis. For a more detailed description of addressing, refer to the SIMATIC Manual of the AS-Interface master module being used.

8.9.3 Program

8.9.3.1 Operating the command interface of the IE/AS-Interface Link PN IO AS-Interface master

Overview

The command interface of the IE/AS-Interface Link PN IO AS-Interface master is operated by means of data record transfer.

Data record transfer

Data record transfer is performed in the SIMOTION system using the functions **_readRecord** and **_writeRecord**. In terms of their functionality, these correspond to system function blocks SFB 52 and SFB 53 in the SIMATIC system (see the table below).

Table 8-129 Parameter assignment

Parameters for SFB 52/ SFB 53 (SIMATIC)	Parameter _readRecord (SIMOTION)	Parameter _writeRecord (SIMOTION)	Meaning
ID	logAddress	logAddress	I/O address or diagnostic address of the IE/AS-Interface Link slot
-	ioId	ioId	Input/output assignment of logical base address of the drive [1]
INDEX	recordNumber	recordNumber	Data record number
MLEN/LEN	dataLength	dataLength	Length of the data record
RECORD	StructRetReadRecord.data	Data	Interface for net data

[1] Parameter **ioId** is parameterized as type Input (for addresses in the I/O input range) or Output (for addresses in the I/O output range). Diagnostic addresses are always of type Input. Two identical ID-logic addresses may be in both the input and output range.

Note

Data record transfer using **_readRecord** or **_writeRecord** is shown in the application example for FB **_ASI_cmdInterface**.

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

Addressing the AS-Interface master (IE/AS-Interface Link PN IO)

Communication between the SIMOTION device and the AS-Interface master takes place via direct access to the I/O (cyclic data transfer) and data record transfer (acyclic data transfer). Addresses starting at 256 are recommended for the AS-Interface master.

For data record transfer, the module address is transferred as input parameter **logAddress** to the **_readRecord**/**_writeRecord** system functions. Communication via direct access to the I/O takes place using I/O variables in the recommended address area (addresses starting at 256).

There are no fixed specifications in terms of how the AS-Interface slave addresses are assigned to the IE/AS-Interface Link PN IO address area; this can be changed in the Step7 HW configuration.

Note

When using the IE/AS-Interface Link PN IO AS-Interface master, a distinction is made between parameterizing AS-Interface line calls and AS-Interface slave calls.

In the case of AS-Interface line calls, the parameters are implemented using the system functions **_readRecord** and **_writeRecord** as follows:

- **logAddress** with the "diagnostic address" of the IE/AS-Interface Link PN IO AS-Interface master
- **recordNumber** with the data record index of the AS-Interface line command
- **datalength** with the max. data record length of 240 bytes

In the case of AS-Interface slave calls, the parameters are implemented using the system functions **_readRecord** and **_writeRecord** as follows:

- **logAddress** with the "I/O address" of the AS-Interface slave modules
- **recordNumber** with the data record index of the AS-Interface slave command
- **datalength** with the max. data record length of 240 bytes
- **ioId** with type Input (for addresses in the I/O input range) or Output (for addresses in the I/O output range)

Additional information on the AS-Interface line calls and AS-Interface slave calls can be found in the *SIMATIC NET IE/AS-Interface Link PN IO* Manual. This document is shipped with SIMOTION SCOUT in electronic form.

8.9.3.2 Operating the command interface of the AS-Interface master CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced

Integrating the _ASI_cmdInterface function block and the application examples in the user project

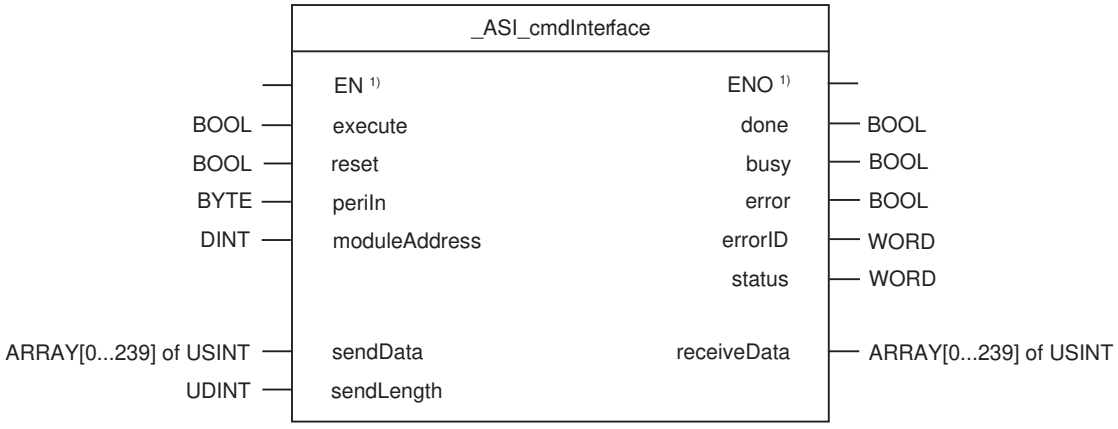
Creating the FB instance in the user project

The **_ASI_cmdInterface** FB is part of the command library of the “SIMOTION SCOUT” engineering system. For the operation of the command interface, an instance of the **_ASI_cmdInterface** FB for the corresponding AS-Interface master must be created in the user project. If several AS-Interface masters are operated, an instance of the FB for each master must be created in the user project.

Example:

```
VAR
   myFB_ASI_cmdInterface : _ASI_cmdInterface;  // instance of FB _ASI_cmdInterface for
                                               // one AS-Interface Master
END_VAR
```

Call (LAD representation)



1) LAD-specific parameters

Example of an application

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

Addressing the AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced)

Overview

Communication between the SIMOTION device and the AS-Interface master takes place via direct access to the I/O (cyclic data transfer) and data record transfer (acyclic data transfer). Addresses starting at 256 are recommended for the AS-Interface master.

For data record transfer, the module address is transferred as **moduleAddress** input parameter to the **_ASI_cmdInterface** FB. Communication via direct access to the I/O takes place using I/O variables in the recommended address area (addresses starting at 256).

Creating I/O variables

You must create an I/O variable in the symbol browser for each read or write access to addresses outside the process image (addresses ≥ 64).

When you create the I/O variable, you must enter the parameterized address of the AS-Interface master from the HW configuration in the "I/O address" parameter. The value in the "field length" parameter depends on the associated AS-Interface master.
Specify the value 16 (= 16 bytes) as the field length for the CP 343-2 P.

The field length (xx) to be parameterized for the DP/AS-Interface Link 20E/Link Advanced must be entered as appropriate for the assigned address space (from HW Config). These values are used to create an I/O variable of types ARRAY[0..15] of byte (CP 343-2 P) and ARRAY[0..(xx-1)] of byte (DP/AS-Interface Link 20E 20E/Link Advanced) in the SIMOTION project for direct write/ read access to the I/O.

Example

I/O variables for access to the AS-Interface slaves with CP 343-2 P:

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperiincp3432	PIB 256		Array	16
2	⊞ myperioutcp3432	PQB 256	☐	Array	16

Figure 8-54 Addressing with I/O variable

Parameter transfer

- Addressing with I/O variables
  The **periIn** call parameter is used to transfer the control information to the **_ASI_cmdInterface** FB. These are contained in array element 0 of the I/O variable created.
- Addressing using addresses within the process image
  For addressing using address within the process image, the input byte of the AS-Interface master start address must be transferred to the **_ASI_cmdInterface** FB.

Example: Calling FB _ASI_cmdInterface

Addressing using addresses outside the process image by means of I/O variables

```
myFBcmdInterface (execute    := ..,
                  reset      := ..,
                  periIn     := myPeriInCP3432[0], // control information
                  moduleAddress := 256,
                  sendData   := ..,
                  sendLength := .. );
```

Addressing with addresses of the process image

```
myFBcmdInterface (execute       := ..,
                  reset         := ..,
                  periIn        := %IB0,       // control information
                  moduleAddress := 0,
                  sendData      := ..,
                  sendLength    := ..);
```

Note

For additional information, see the following sources:

- *SIMOTION SCOUT* online help
- Programming Manuals for the relevant programming languages, such as:
  - *SIMOTION ST, Structured Text* Programming Manual
  - *SIMOTION MCC, Motion Control Chart* Programming Manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

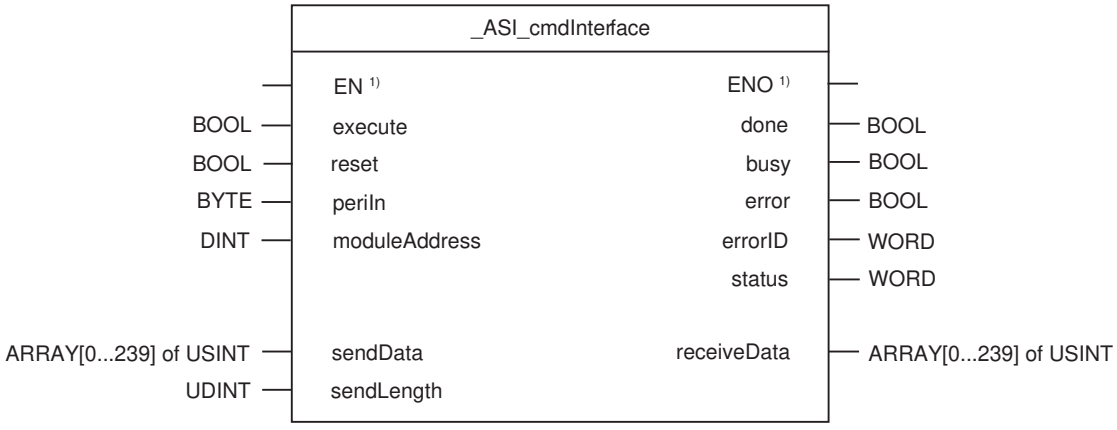
8.9.4 Parameter assignment,

8.9.4.1 _ASI_cmdInterface function block

Task

You can use the **_ASI_cmdInterface** FB to easily and completely control the AS-Interface master behavior from the user program. The **_ASI_cmdInterface** FB transfers the commands and data to the AS-Interface master, receives the response data provided by the module, and transfers this data. The calls for the reading and writing of data records are managed in the **_ASI_cmdInterface** FB.

Call (LAD representation)



1) LAD-specific parameters

parameter description

The following table contains all parameters of the **_ASI_cmdInterface. function block**

Note

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6465).

Table 8-130 FBs _ASI_cmdInterface parameters

Name	Type	Data type	Default	Meaning
execute	IN	BOOL	FALSE	Starting a new request with a rising edge
reset	IN	BOOL	FALSE	**TRUE** = Error acknowledgement/reinitialization (takes priority over command processing)
periIn	IN	BYTE	16#00	Control information from AS-Interface master for FB **_ASI_cmdInterface** (see Chapter Addressing the AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced) (Page 6442))
moduleAddress	IN	DINT	0	Module address from **HW Config**
sendData	IN	ARRAY[0...239] of USINT	0	Data to be sent (command number, possibly parameters)
sendLength	IN	UDINT	0	Length of the data to be sent (depends on the command to be sent)
done	OUT	BOOL	FALSE	**TRUE** = request was processed successfully
busy	OUT	BOOL	FALSE	**TRUE** = request in progress
error	OUT	BOOL	FALSE	**TRUE** = error during the request processing

Name	Type	Data type	Default	Meaning
errorID	OUT	WORD	16#0000	Error specification of the error For **error** = **TRUE**, the **errorID** parameter contains the error information (see table titled "Error and status messages", Chap. Error and status messages of the FB _ASI_cmdInterface (Page 6460)).
status	OUT	WORD	16#0000	Status of the command processing (see table titled "Status messages", Chap. Error and status messages of the FB _ASI_cmdInterface (Page 6460))
receiveData	OUT	ARRAY[0...239] of USINT	0	Data field for the response data of a request

Function description

A rising edge at the **execute** input initiates a request. The **done** output parameter will be reset. The completion of a request without error will be signaled with falling edge at the **busy** output parameter and with rising edge at the **done** output parameter. The completion of a request with error will be signaled with falling edge at the **busy** output parameter and with rising edge at the **error** output parameter. The error is specified in the **errorID** output parameter.

Errors present at the **error** output parameter must be reset with a rising edge on the **reset** input parameter before a new request is initiated. This resets the **error** and **errorID** output parameters.

The **reset** input parameter has priority over the command processing. A request initiated at the same time will be stored internally and executed once error acknowledgement/reinitialization is complete.

Prior to initiating the request with rising edge at the **execute** input parameter; the user must enter the request number or the data that belongs to the request in the **sendData** input parameter and set as parameter the length of the data to be sent in the **sendLength** input parameter.

If the AS-Interface master passes response data for a command (e.g. read version identification), the **_ASI_cmdInterface** FB will provide this data in the **receiveData** output parameter. The corresponding data for the AS-Interface slave commands and the meaning of the received data are described in the documentation for the CP 343-2 P and the DP/AS-Interface Link 20E/Link Advanced.

The **status** output parameter indicates the processing status of the command and the /**reset** reinitialization (see the table titled "Error and status messages", Chapter Error and status messages of the FB _ASI_cmdInterface (Page 6460)).

Note

A rising edge at the **execute** input parameter will not be detected when a command or a reinitialization is being processed (**busy = TRUE** output parameter).

Example: read parameter value

```
mySendData[0] := 16#03;                              // command number for read parameter
mySendData[1] := 16#01;                              // address ASI-Slave
mySendLength  := 2;
myExecute     := TRUE;

myFbAsiCmdInterface (execute    := myExecute,
                     sendData   := mySendData,
                     sendLength := mySendLength,
                     ...);

myParaValue   := myFbAsiCmdInterface.receiveData[0];  // read parameter value
```

Graphical overview of the functionality

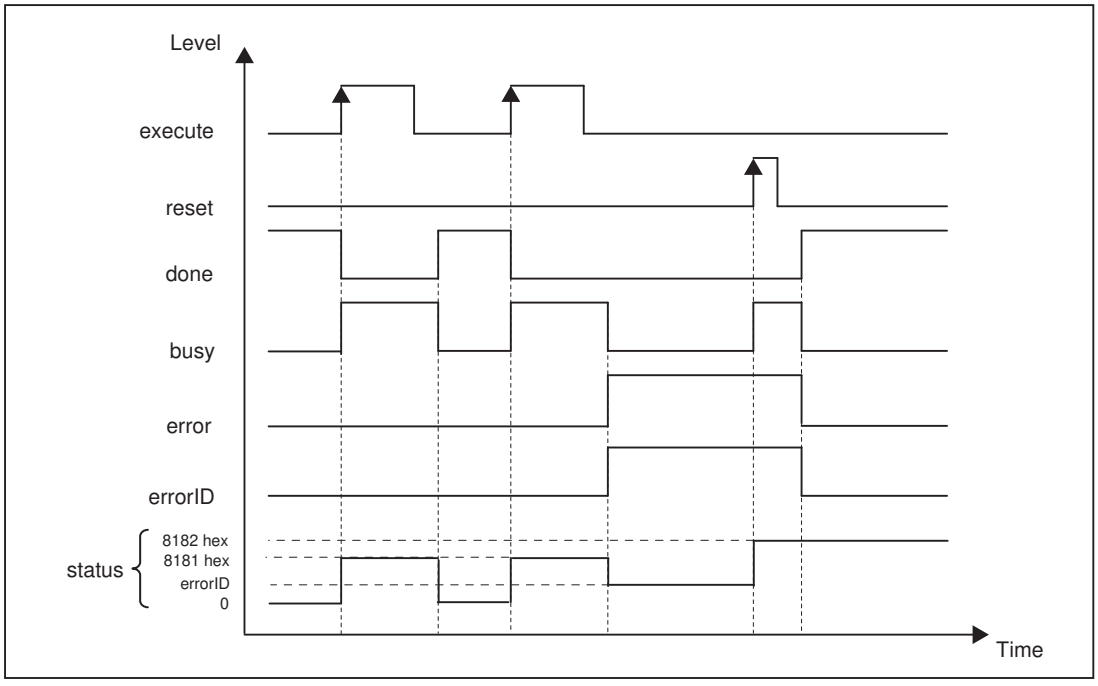


Figure 8-55 Signal sequence diagram

Task integration (call)

The **_ASI_cmdInterface** function block is provided for being called in a cyclical task and must be called in this task for each pass. A request is processed over several cycles. The user decides in which cyclical task the **_ASI_cmdInterface** FB is called. The **_ASI_cmdInterface** FB functionality does not place any restrictions. The call can be made in all cyclical tasks. A fixed time frame (e.g. **IPOSynchronousTask**) is not required for processing.

Error messages

You will find the error and status messages in the chapter titled Error messages / diagnosis (Page 6460).

8.9.4.2 Call example for the FB _ASI_cmdInterface

Calling the function block

In order to work with the function block in your user project, proceed as follows (the numbers shown in the following program snippet correspond to the steps below):

1. Create an instance of the function block
2. Create variable for the send and receive data array
3. Call instance of the function block
4. Transfer input parameters
5. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.

Note

The call example is a fragment from the provided application example contained on the "SIMOTION Utilities & Applications" CD-ROM.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

If you want to trigger several AS-Interface master modules, you must create a new FB instance with a new name for each used AS-Interface master.

Program fragment from the program example:

```
UNIT E_AsiCmd;

INTERFACE

VAR_GLOBAL
  myAsiCmdInterface : _ASI_cmdInterface;  // Instance of FB _ASI_commandInterface                (1)
  mySendData    : ARRAY[0..239] OF USINT; // send buffer with ASI commands and data              (2)
  myReceiveData : ARRAY[0..239] OF USINT; // receive buffer
  mySendLength  : UDINT;                  // send length
  myResetAsi    : BOOL;                   // reset ASI master
END_VAR

PROGRAM ExampleAsiCommand;
```

```
END_INTERFACE

IMPLEMENTATION

// ***********************************************************************
// program to handle ASI commands
// ***********************************************************************
PROGRAM ExampleAsiCommand

  // prepare send command
  IF (FP_executeAsiCommand.Q=TRUE)      // test new command
  AND (myAsiCmdInterface.busy=FALSE)    // test FB _ASI_cmdInterface is not running
  THEN
    mySendData[0]   := 16#14;           // code of ASI command "read version"
    mySendLength    := 1;               // send length
    myRequestAsiCmd := TRUE;            // set request for new ASI command
  END_IF

  // call FB _ASI_commandInterface                                                    (3)
  myAsiCmdInterface
  ( execute       := myRequestAsiCmd,   // request ASI command
    reset         := myResetAsi,        // reset ASI master
    periIn        := myPeriInDataByte0, // Byte0 of peripheral data ASI master      (4)
    moduleAddress := myModuleAddress,   // module address of ASI master
    sendData      := mySendData,        // send buffer with ASI commands and data
    sendLength    := mySendLength       // send length
  );
  //
  IF(myAsiCmdInterface.done=TRUE)                                                  (5)
  THEN
     // finish without error
     myDone             := TRUE;
     myError            := FALSE;
     myRequestAsiCmd    := FALSE;       // reset request
     myResetAsi         := FALSE;
     // copy received data
     myReceiveData      := myAsiCmdInterface.receiveData;                         (5)

  ELSIF(myAsiCmdInterface.error=TRUE)                                              (5)
  THEN
     // finish with error
     myDone             := FALSE;
```

```
    myError            := TRUE;
    myRequestAsiCmd    := FALSE;       // reset request
    myResetAsi         := FALSE;
    // copy error code
    myErrorID          := myAsiCmdInterface.errorID;                    (5)
  END_IF;

END_PROGRAM

END_IMPLEMENTATION
```

8.9.5 Configuring

8.9.5.1 Setup and connection

Overview

You must perform the following steps in order to commission the AS-Interface master modules and control them from the SIMOTION system:

Distributed use of the CP 343-2 P AS-Interface master module (SIMOTION C2xx, SIMOTION P350 and SIMOTION D4xx)

1. Assemble and install the cables for the ET 200M distributed I/O device complete with power supply (PS), interface module (IM), and the CP 343-2 P AS-Interface master module. Install the cables and address the AS-Interface slaves.
2. Establish the PROFIBUS connection between the ET 200M and the SIMOTION device.
3. Set the PROFIBUS DP node address on the IM.
4. Switch on the terminating resistor at the first and last bus node.

   Note

   Steps 1. to 4. are described in detail in the *"ET 200M Distributed I/O Device"* Manual.

5. Add the CP 343-2 P AS-Interface master module to the SIMOTION project (see the following section).

6. Parameterize the CP 343-2 P AS-Interface master module and the AS-Interface slaves in **HW Config**. For a detailed description of parameterizing the modules, see:
   - *SIMATIC NET CP 343-2/CP 343-2 P AS-Interface Master* Manual.
   - Documentation for the corresponding AS-Interface slaves

   You can use the "Load into the PG" function to import the configuration of the AS-Interface slaves into your hardware configuration:

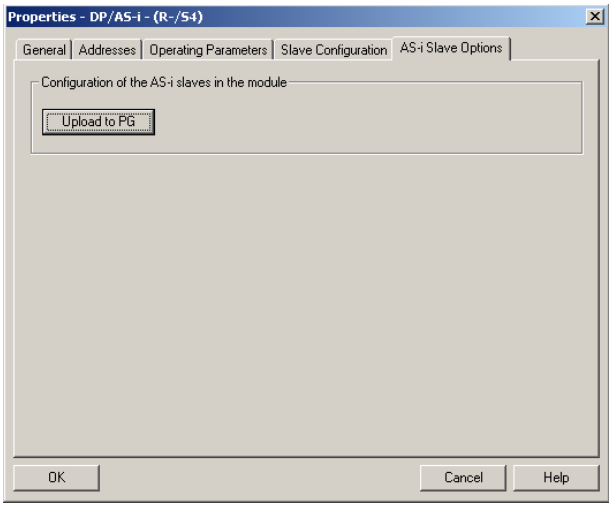


Figure 8-56 Loading the AS-Interface slave configuration into the PG

7. Link the function block to the SIMOTION project (refer to Chapter Integrating the _ASI_cmdInterface function block and the application examples in the user project (Page 6441)).

Distributed use of the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module (SIMOTION C2xx, SIMOTION P350, and SIMOTION D4xx)

1. Assemble and install the cables for the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module. Install the cables and address the AS-Interface slaves.
2. Establish the PROFIBUS connection between the DP/AS-Interface Link 20E/Link Advanced and the SIMOTION device.
3. Set the PROFIBUS DP node address on the DP/AS-Interface Link 20E/Link Advanced.
4. Switch on the terminating resistor at the first and last bus node.

   Note

   Steps 1. to 4. are described in detail in the *SIMATIC NET DP/AS-Interface Link 20E* and *SIMATIC NET DP/AS-Interface Link Advanced* Manuals.

5. Add the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module to the SIMOTION project (see Chapter Adding the AS-Interface master modules to the SIMOTION project (Page 6453)).

6. Parameterize the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module and the AS-Interface slaves in **HW Config**. For a detailed description of parameterizing the modules, see:
   - *SIMATIC NET DP/AS Interface Link 20E* Manual
   - *SIMATIC NET DP/AS-Interface Link Advanced* Manual.
   - Documentation for the corresponding AS-Interface slaves
     You can use the "Load into the PG" function to import the configuration of the AS-Interface slaves into your hardware configuration:

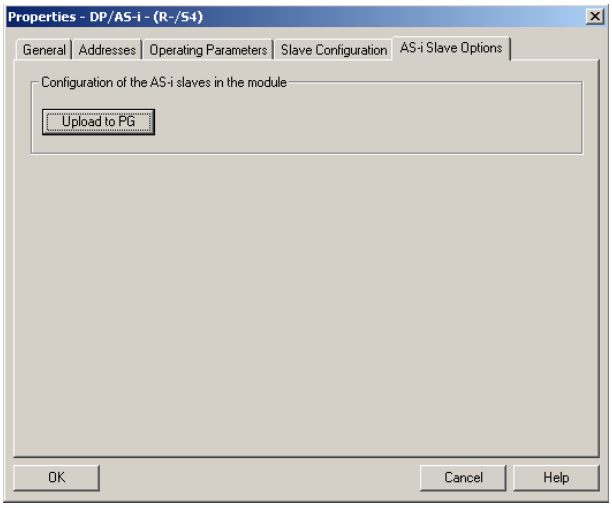


Figure 8-57 Loading the AS-Interface slave configuration into the PG

7. Link the function block to the SIMOTION project (refer to Chapter Integrating the _ASI_cmdInterface function block and the application examples in the user project (Page 6441)).

Distributed use of the AS-Interface master module (IE/AS-Interface Link PN IO)

1. Assemble and install the cables for the IE/AS-Interface Link PN IO AS-Interface master module. Install the cables and address the AS-Interface slaves.
2. Establish the PROFINET connection between the IE/AS-Interface Link PN IO and the SIMOTION device.
3. Add the IE/AS-Interface Link PN IO AS-Interface master module to the SIMOTION project (see Chapter Adding the AS-Interface master modules to the SIMOTION project (Page 6453)).
4. Assign device names according to the rules for address and name assignment in the SIMOTION system (see: *SIMOTION SCOUT Communication* System Manual.
5. Parameterize the IE/AS-Interface Link PN IO AS-Interface master module and the AS-Interface slaves in **HW Config**. For a detailed description of parameterizing the module, see:
   - *SIMATIC NET IE/AS-Interface Link PN IO* Manual.
   - Documentation for the corresponding AS-Interface slaves
     You can use the "Upload to PG" function to import the configuration of the AS-Interface slaves into your hardware configuration.

Centralized use of the CP 343-2 P AS-Interface master module (for SIMOTION C2xx only)

1. To configure the mechanical design, and prepare and install the SIMOTION components for assembly, see:
   - *SIMOTION C* Operating Instructions
   - *SIMATIC S7-300 Automation System, Hardware and Installation* Software Installation Manual.
2. To continue, refer to steps 5. to 7. for distributed use with PROFIBUS.

Note

The listed documents are included in the SIMOTION SCOUT scope of supply as electronic documentation

8.9.5.2 Adding the AS-Interface master modules to the SIMOTION project

Requirements

The following requirements must be met when networking with PROFIBUS:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet (for distributed use only).

Note

For how to create a project and configure a PROFIBUS subnet, refer to the online help of *SIMOTION SCOUT*.

The following requirements must be met when with PROFINET:

1. You have created a project in SIMOTION SCOUT and have inserted and configured a rack with a PROFINET-compatible SIMOTION device in the hardware configuration.
2. You have configured a PROFINET IO System (for distributed use only).

Note

For how to create a project and configure a PROFINET IO system, refer to the online help of *SIMOTION SCOUT*.

Inserting CP 343-2 P AS-Interface master modules (distributed use)

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.

4. Open the **PROFIBUS DP** folder and the **ET 200M** subfolder in the hardware catalog. Select, for example, the IM 153-1 interface module (Article No.: 6ES7 153-1AA03-0XB0 or a successor module).
5. Use **drag-and-drop** to place the IM 153-1 I/O device on the PROFIBUS subnet of your project. The **Properties - PROFIBUS IM 153-1 Interface** dialog box opens. In this dialog box, select the address you set on the IM 153-1 (see ET 200M Distributed I/O Device Manual) and confirm with **OK**. The selected IM 153-1 I/O device is inserted in the project.
6. The inserted I/O device must now be fitted with your project modules. Open the **CP-300 > AS-Interface** subfolder under the selected I/O device in the hardware catalog. Select the appropriate CP modules there.

   Note

   Diagnostic alarms are not enabled by default. Activate the alarms for each module in the **Properties** dialog box.

7. **Save** and **compile** your project.

Inserting DP/AS-Interface Link 20E/DP/AS-Interface Link Advanced AS-Interface master modules (distributed use)

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.
4. Open the **PROFIBUS DP** folder and the **DP/AS-i** subfolder in the hardware catalog. Here, select **DP/AS-i Link 20E** or **DP/AS-i Link Advanced**.
5. Use **drag-and-drop** to move the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module to the PROFIBUS subnet of your project. The **Properties - PROFIBUS Interface DP/AS-i** dialog box is displayed. Here, select the address you set on the DP/AS-Interface Link 20E/Link Advanced AS-Interface master module (see the *"SIMATIC NET DP/AS-Interface Link 20E"* or *"SIMATIC NET DP/AS-Interface Link Advanced"* Manual) and confirm with **OK**. The selected DP/AS-Interface Link 20E/Link Advanced AS-Interface master module is inserted in the project.

   Note

   Diagnostic alarms are not enabled by default. Activate the alarms for each module in the **Properties** dialog box.

6. **Save** and **compile** your project.

Inserting an IE/AS-Interface Link PN IO AS-Interface master module (distributed use)

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.

3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.
4. Open the **PROFINET IO** folder and the **Gateway** subfolder in the hardware catalog. Here, select **IE/AS-i Link PN IO**.
5. Use **drag-and-drop** to move the IE/AS-Interface Link PN IO AS-Interface master module to the PROFINET IO System of your project.
6. Assign a device name according to the rules for address and device name assignment in the SIMOTION system.
7. If necessary, adjust how the ASI slave addresses are assigned to the input and output addresses of the AS-Interface master module.
8. **Save** and **compile** your project.

8.9.6 Example of an application

8.9.6.1 General information on the application example

Overview

The application example for the **_ASI_cmdInterface** FB is available for the following AS-Interface master modules:

- CP 343-2 P
- DP/AS-Interface Link 20E
- DP/AS-Interface Link Advanced

Task

The example program for the **_ASI_cmdInterface** FB shows the call and the parameter settings for the **_ASI_cmdInterface** FB. Five AS-Interface commands have been programmed. Further AS-Interface commands can be added to the program.

In addition, the sample project contains 2 programs for reading and writing analog and digital values.

You can choose from 2 sample projects:

- Sample project for **_ASI_cmdInterface** FB with the CP 343-2 P
- Sample project for **_ASI_cmdInterface** FB with the DP/AS-Interface Link 20E/Link Advanced

The two projects contain the following example programs:

Program	Task	Meaning
StartUpAsiCommand	StartupTask	Program for start-up
PeripheralFaultAsi	PeripheralFaultTask	Program for handling diagnostic alarms
ExampleAsiCommand	BackgroundTask	Example for serving the command interface

Program	Task	Meaning
ExampleAsiAnalog	BackgroundTask	Example for reading and writing analog values
ExampleAsiDigital	BackgroundTask	Example for reading and writing digital values

The following addresses are set by default for the example projects:

Sample project	Address
Sample project with CP 343-2 P	256
Sample project with DP/AS-Interface Link 20E/Link Advanced	0 (process image)

The following AS-Interface slaves are used on the AS-Interface addresses 3..5 in both example projects:

Address	AS-Interface slave
Address 3	AS-Interface slave with 2 digital inputs and outputs
Address 4	AS-Interface slave with 2 analog outputs
Address 5	AS-Interface slave with 2 analog inputs

Requirement

The addressing of the AS-Interface slaves and the configuring of the AS-Interface must be completed.

Hardware platform

The application example is available for various SIMOTION hardware platforms. A sample project with CP 343-2 P and DP/AS-Interface Link 20E/Link Advanced is provided for each hardware platform.

Note

If the application example is not available for your hardware platform, you have to adapt the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration.

3. If necessary, adapt the example project.
4. Save and compile the example project. Following this, you can download the sample project to the SIMOTION device and switch to **RUN** mode.

Adapting the application example

The configuration in the example and its available hardware must be adapted to each other. The following options are available:

- You can adapt the configuration in the example to the available hardware.
- Adapt the configuration of the hardware to the example.

You can adapt the configuration in the example to the available hardware.

AS-Interface master:

Check the address in the "moduleAddress" variable and, if necessary, change the parameterized addresses of the I/O variables in the example project.

AS-Interface analog slaves:

If addresses other than the default addresses 4 and 5 are used for the analog AS-Interface slaves, the values of the indexes for the send or receive data array of the **_writeRecord/_readRecord** system functions must be adapted in the following variables.

Default indexes of the send data array based on the analog AS-Interface slave with address 4:

```
highByteAnaOutCh1       :=24;
lowByteAnaOutCh1        :=25;
highByteAnaOutCh2       :=26;
lowByteAnaOutCh2        :=27;
```

Default indexes of the receive data array based on the analog AS-Interface slave with address 5:

```
highByteAnaInCh1        :=32;
lowByteAnaInCh1         :=33;
highByteAnaInCh2        :=34;
lowByteAnaInCh2         :=35;
```

In addition, the value for the bytes to be transferred and the data record number must be adapted for addresses larger than 5 of the analog AS-Interface slaves and for addresses larger than 16, respectively. Data record 140 with a length of 40 bytes is set by default.

Both values are contained in the following variables:

```
dsNumber                :=140;
dsLength                :=40;
```

AS-Interface digital slaves:

The write and read accesses and the bit shift operations must be modified appropriately for digital AS-Interface slaves with addresses other than the default address 3.

Note

For a more detailed description of addressing, refer to the SIMATIC Manual of the AS-Interface master module being used.

8.9.6.2 Execution of the example project programs

Example program for serving the command interface

You can use the **ExampleAsiCommand** example program to transfer AS-Interface commands to the AS-Interface master with the **_ASI_cmdInterface** FB. You use the following variables to control the program execution:

asiCommands : Selection of the AS-Interface command (default: READ_VERSION)
executeAsiCommand : Start the transfer

The data read by the AS-Interface master are contained in the **receiveData** variable.

The status of the command processing is indicated in the following variables:

done : Command processing completed without error
error : Command processing completed with error
errorID : Error specification / error codes

The following AS-Interface commands have been programmed:

AS-Interface command	Meaning
READ_VERSION (default)	Fetch the version identification
CHANGE_ASI_SLAVE_ADDRESS	Change an AS-Interface slave address
READ_EXTENDED_CONFIGURATION	Import the configuration of the AS-Interface slaves connected to the AS-Bus
GET_LAS	Read extended lists and flags
RESET_ASI	Reset/initialize AS-Interface master
WRITE_ANA_PARAMETER	Transfer parameters to the analog AS-Interface slave with AS-Interface address 4 or 5

Example program for reading and writing analog values

The **ExampleAsiAnalog** example program illustrates not only the preparation, but also the reading and writing of analog values to and from analog AS-Interface slaves. Analog AS-Interface slaves (each with 2 channels) with AS-Interface address 4 (analog output) and AS-Interface address 5 (analog input) are used here.

You use the following variables to control the program execution:

wrAnaData : Write analog values
rdAnaData : Read analog values
dataAnaOut1 : Analog value for channel 1 to be written
dataAnaOut2 : Analog value for channel 2 to be written
dataAnaIn1 : Read analog value for channel 1
dataAnaIn2 : Read analog value for channel 2

The following variables indicate the status of the read/write of the analog values:

doneAnaWrite : Write analog value (data record transfer) completed without error
errorAnaWrite : Write analog value (data record transfer) completed with error
errorIdAnaWrite : Error specification of the data transfer/error code (see **_readRecord** and **_writeRecord** system functions, and Chap. Error messages / diagnosis (Page 6460))
doneAnaRead : Read analog value (data record transfer) completed without error
errorAnaRead : Read analog value (data record transfer) completed with error
errorIdAnaRead : Error specification of the data transfer/error code (see **_readRecord** and **_writeRecord** system functions, and Chap. Error messages / diagnosis (Page 6460))

Example program for reading and writing digital values

The **ExampleAsiDigital** example program writes and reads binary values from and to an AS-Interface slave with digital inputs and outputs. A slave with 2 digital inputs, 2 digital outputs, and AS-Interface slave address 3 is used.

Write or read the digital values in the following variables:

dataDigOut1 : Value to be written for digital output 1
dataDigOut2 : Value to be written for digital output 2
dataDigIn1 : Digital value read from input 1
dataDigIn2 : Digital value read from input 2

8.9.7 Error messages / diagnosis

8.9.7.1 Error and status messages of the FB _ASI_cmdInterface

The **error = TRUE** output parameter will be set if an error occurred. In this case, the AS-Interface master will not provide any receive data for requests with response data. The error is specified in the **errorID** output parameter. This output parameter contains the errors from the data record transfer (**_readRecord** and **_writeRecord** system functions) and the errors from the AS-Interface. The *SIMOTION System Functions/Variables Device*.
Parameter Manual contains a detailed description of the system functions.
This document is shipped with SIMOTION SCOUT in electronic form.

Table 8-131 Error messages

Error no.: hex (dec)	Meaning
Error messages (errorID parameter)	
0000 (0)	Request completed without errors
8090 (32912)	Error during data record transfer, request aborted. Specified logical base address invalid.
8091 (32913)	Error during data record transfer, request aborted. The **_readRecord** function cannot reach the specified logical base address.
809E (32926)	Error, request aborted. Attempt to interrupt a non-active function.
809F (32927)	Error, request aborted. Function cannot be executed.
80A0 (32928)	Error during data record transfer, request aborted. Negative acknowledgment when reading from the module: • Module was removed during the read operation • Defective module
80A1 (32929)	Error during data record transfer, request aborted. Negative acknowledgment when writing to module: • Module removed during write operation • Defective module
80A2 (32930)	Error during data record transfer, request aborted. PROFIBUS DP protocol error in layer 2.
80A3 (32931)	Error during data record transfer, request aborted. PROFIBUS DP protocol error in user interface/user.
80B0 (32944)	Error during data record transfer, request aborted. • System function not supported for this module type. • Module does not recognize the data record
80B1 (32944)	Error during data record transfer, request aborted. The length specified in the **sendLength** parameter is incorrect.
8184 (33156)	Invalid data type for the **receiveData** formal operand.
8381 (33665)	The AS-Interface slave address is incorrect.
8382 (33666)	The AS-Interface slave is not activated (not in LAS).
8383 (33667)	Error on the AS-Interface
8384 (33668)	The command is not permitted in the current state of the AS-Interface master.
8385 (33669)	An AS-Interface slave with address 0 exists.
8386 (33670)	The AS-Interface slave has invalid configuration data (I/O or ID codes).
83A1 (33697)	The requested AS-Interface slave was not found on the AS-Interface.

Error no.: hex (dec)	Meaning
83A2 (33698)	An AS-Interface slave with address 0 exists.
83A3 (33699)	An AS-Interface slave with the new address already exists on the AS-Interface.
83A4 (33700)	The AS-Interface slave address cannot be deleted.
83A5 (33701)	The AS-Interface slave address cannot be set.
83A6 (33702)	The AS-Interface slave address cannot be stored permanently.
83A7 (33703)	Error while reading the Extended ID1 code.
83A8 (33704)	The target address is not plausible (e.g. a B-Slave address is used for a Standard slave).
83B1 (33713)	A length error occurred during the string transfer to profile 7.4.
83B2 (33714)	A protocol error occurred during the string transfer to profile 7.4.
83F8 (33784)	The request number or the request parameter is not known.
83F9 (33785)	The AS-Interface master has detected an EEPROM error.
9999 (39321)	The number of repetitions for transient errors has been exceeded. Retry request.

Table 8-132 Status messages

Error no.: hex (dec)	Meaning
Status messages (status parameter)	
0000 (0)	Request successfully completed
8181 (33153)	Command being processed
8182 (33154)	Status after **reset**

8.9.7.2 Diagnosis

Definition of a diagnostic alarm

If the user program is to respond to an internal or external error, you can set the parameters for a diagnostic alarm that will interrupt the cyclical program of the SIMOTION device.

Events that trigger a diagnostic alarm

The criteria (events) that trigger diagnostic alarms in a SIMOTION system are the same as in a SIMATIC system.

More information is available in the following manuals:

- SIMATIC NET CP 343-2/CP 343-2 P, AS-Interface Master
- SIMATIC NET DP/AS-Interface Link 20E
- SIMATIC NET DP/AS-Interface Link Advanced
- SIMATIC NET IE/AS-Interface Link PN IO

These documents are shipped with SIMOTION SCOUT in electronic form.

Responses to a diagnostic alarm

If a diagnostic alarm occurs, the following take place:

- The diagnostic data is written to the TSI#details variable in the Taskstartinfo of **PeripheralFaultTask**. The SIMOTION hardware platform goes into STOP mode if a program has not been assigned in the **PeripheralFaultTask**.
- You can call the **_readRecord** or the **_readDiagnosticData** system function to read additional diagnostic data in the **BackgroundTask** or another cyclic task.
- The group error LED (SF) of the AS-Interface master module illuminates and clears once the error has been corrected.

Error diagnosis of the AS-Interface master

Pending error messages are processed and evaluated differently in a SIMOTION system than in a SIMATIC system. As default, the diagnosis is not activated. Activate the alarms for the AS-Interface master in the hardware configuration.

Once you have parameterized the diagnostic alarms, program the execution of the alarm processing using the principle shown in the following figure:

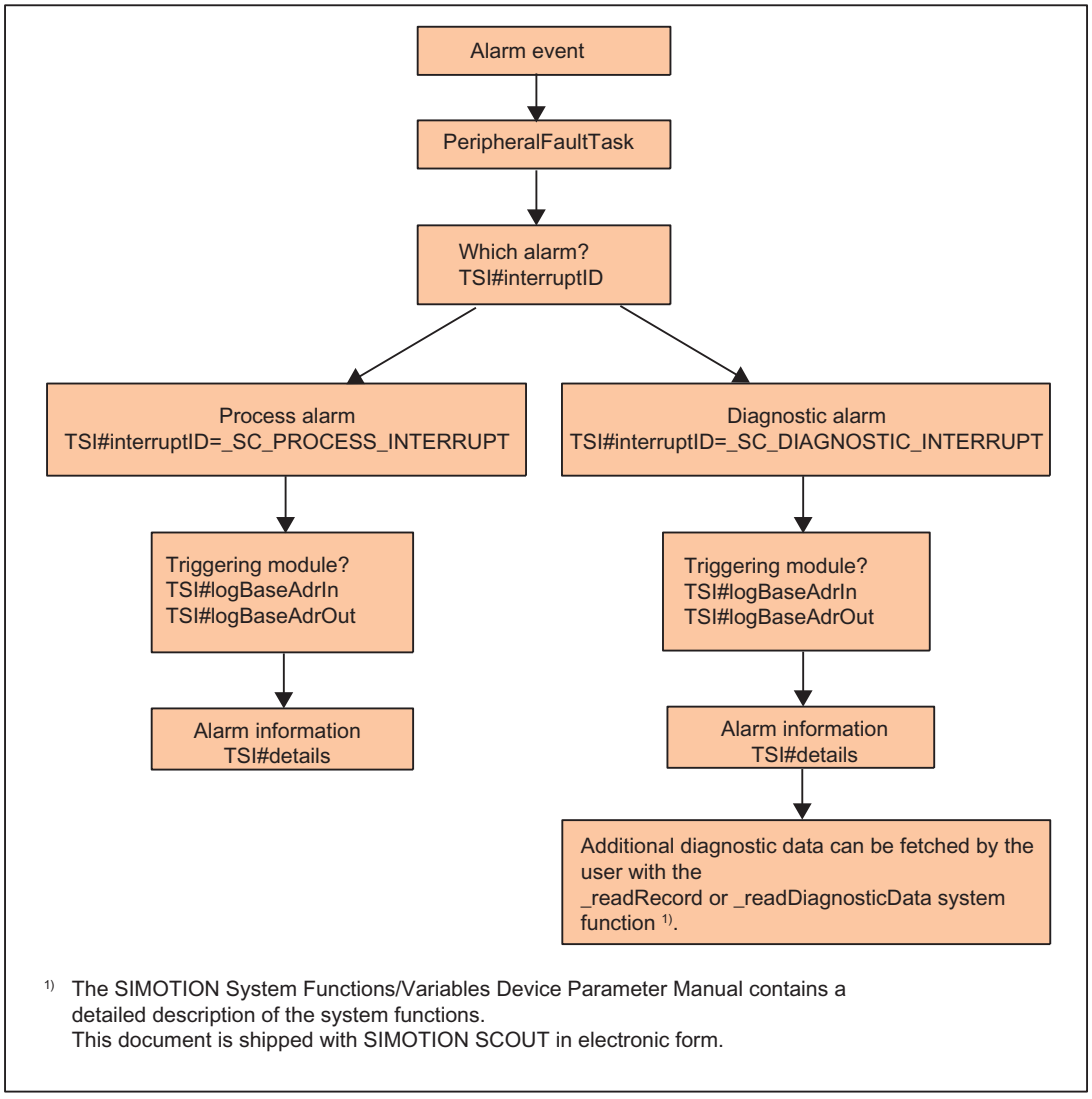


Figure 8-58 Alarm processing

Alarm evaluation

In order to evaluate diagnostic information when a diagnostic alarm occurs, you must integrate an appropriate evaluation program into the **PeripheralFaultTask**. The task start information contains the standard diagnostic data.

You can fetch additional diagnostic data using the **_readRecord** or **_readDiagnosticData** system functions.

Note

CP 343-2 P: The **_readRecord** (data record 1 with length 16) function is called with the parameter logAdress := module address from **HW Config**.

DP/AS-Interface Link 20E/Link Advanced: The **_readDiagnosticData** function is called with the parameter logAddress := diagnostic address from **HW Config**.

IE/AS-Interface Link PN IO: The **_readDiagnosticData** function is called with the parameter logAddress := diagnostic address from **HW Config**.

The **Taskstartinfo** of **PeripheralFaultTask** is comparable to the local data of OB82 in the SIMATIC system.

Table 8-133 Meaning of the Taskstartinfo

Task	TSI		Comment
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an input area on the module, otherwise INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an output area on the module, otherwise INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the alarm was caused by a station failure or station recovery of an associated DP slave, otherwise INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

8.9.8 Appendix

8.9.8.1 SIMOTION and SIMATIC names

The table below contains a comparison of SIMOTION and SIMATIC names.

Table 8-134 SIMOTION and SIMATIC names

Name in the SIMOTION system as of V4.0 (program library in SCOUT)	Name in the SIMATIC system
Function block parameters	
INPUT	-
execute	ACT
reset	-
periIn	-
moduleAddress	LADDR
sendData	SEND
sendLength	-
OUTPUT	
done	DONE
busy	-
error	ERROR
errorID	STATUS (error message)
status	STATUS (status message)
receiveData	RECV

8.9.8.2 List of abbreviations

Table 8-135 Abbreviations

Abbreviation	Meaning
AS	Actor Sensor
ASI	Actor Sensor Interface
CP	Communications Processor
DB	Data block
DP	Decentralized peripherals
DS	Data set
FB	Function block
ID	Identifier (designates a unique coding of a product)
IE	Industrial Ethernet
IM	Interface Module (SIMATIC S7-300 interface module)
IN	Input parameters
IN/OUT	In/out parameter

Abbreviation	Meaning
LAD	Ladder Diagram
LAS	List of active slaves
OUT	Output parameter
PN	PROFINET
PS	Power Supply (SIMATIC S7-300)
SF	System Fault
SFB	System Function Block (SIMATIC)

8.10 Supplement to the ET 200S 1SI serial interface module

Preface

Contents of the function manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

This manual is a supplement to SIMATIC manual *ET 200S 1Sl Serial Interface Modules*.

This documentation is supplied with the SIMOTION SCOUT in electronic form!

Note

The function blocks have been developed for the ET 200S 1Sl serial interface module (3964R, ASCII) with Article No. 6ES7 138-4DF0☐-0AB0.

This supplement will enable you to correctly integrate and commission the ET 200S 1Sl serial interface module in a SIMOTION system.

Differences in handling which result from the software architecture of a SIMOTION system as compared to the software architecture of a SIMATIC system will be described.

Function block

The function blocks for communication between the SIMOTION system and the ET 200S 1Sl serial interface module are part of the program library of the "SIMOTION SCOUT" engineering system.

Sections in this manual

The following sections of the manual describe the function blocks (FBs) used in a SIMOTION system.

- General
  This section describes the similarities and differences in handling the ET 200S 1Sl serial interface module.
- Function blocks of the ET 200S 1Sl serial interface module
  This section describes the function blocks required for communication between a SIMOTION system and an ET 200S 1Sl serial interface module.
- Alarm processing
  This section describes the differences in alarm processing in the SIMOTION system compared to the SIMATIC system.
- SIMATIC and SIMOTION names
  This section contains a comparison of SIMATIC and SIMOTION names.
- The index allows you to locate information quickly.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.3.1:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.10.1 Fundamental safety instructions

8.10.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.10.1.2 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that can be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

8.10.1.3 Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

8.10.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

8.10.2 Description

8.10.2.1 General

This section describes the general differences for the serial interface module and for data exchange when operated in a SIMOTION system as compared to a SIMATIC system.

Note

This manual is a supplement to SIMATIC manual *ET 200S 1SI Serial Interface Modules*.

This documentation is supplied with the SIMOTION SCOUT in electronic form!

The function blocks have been developed for the ET 200S 1SI serial interface module (3964R, ASCII) with Article No. 6ES7 138-4DF0□-0AB0.

The following software versions are required for the standard functions described in this documentation:

- SIMOTION SCOUT V4.0 or higher
- SIMOTION Kernel V4.0 or higher
- SIMOTION technology packages V4.0 or higher

8.10.2.2 Product description

The ET 200S 1SI serial interface module enables you to exchange data between your SIMOTION system and another communication partner.

You can select the mode of communication for data exchange in the hardware configuration of SIMOTION SCOUT. The serial interface module is displayed in the hardware catalog in the following versions:

- 1SI 3964(R) (4 bytes)
- 1SI 3964(R) (8 bytes)
- 1SI 3964(R) (32 bytes)
- 1SI ASCII (4 bytes)
- 1SI ASCII (8 bytes)
- 1SI ASCII (32 bytes)

8-byte or 32-byte data transfers increase the throughput rate, but require more I/O memory on the ET 200S rack, whereas 4-byte data transfers require less I/O memory on the ET 200S rack, but provide a lower throughput rate. The module variant you choose depends on your application requirements.

Note

The 32-byte interface is only supported by the serial interface modules with Article No. 6ES7 138-4DF01-0AB0!

The serial interface modules with Article No. 6ES7 138-4DF00-0AB0 only support 4-byte and 8-byte interfaces!

Function blocks are required to manage communication. The function blocks for the SIMOTION system are described in this manual; these function blocks are handled differently than the function blocks for SIMATIC S7.

Functionality of the ET 200S 1SI serial interface module

The functionality of the function blocks and the ET 200S 1SI serial interface module in a SIMOTION system is the same as that in SIMATIC S7.

The following ET 200S 1SI serial interface modules are **not** supported by the function blocks:

- 1SI Modbus Master (4, 8, 32 bytes)
- 1SI USS Master (4, 8, 32 bytes)

Possible applications

You can use the ET 200S 1SI serial interface module without restrictions in a SIMOTION system. For this to be possible, the module must be operated as a distributed module via an ET 200S I/O system and a PROFIBUS connection to the SIMOTION system.

You can operate multiple ET 200S 1SI serial interface modules on one SIMOTION device.

The following figure illustrates how to connect an ET 200S distributed I/O system with an IM 151-1 and two serial interface modules on a SIMOTION device (e.g. SIMOTION C2xx).

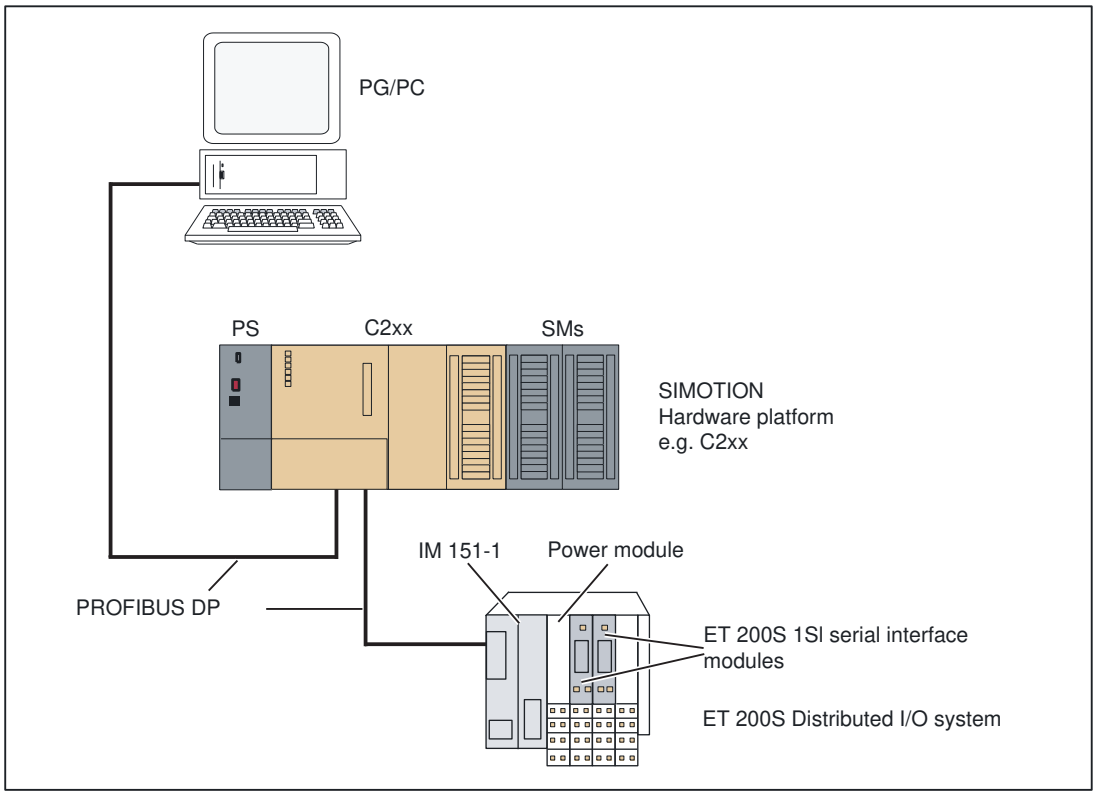


Figure 8-59 Connection of the ET 200S serial interface modules to the SIMOTION C2xx (example)

8.10.2.3 Setup and connection

Overview

You must perform the following steps to commission the serial interface module and control it from the SIMOTION system:

1. Assemble and wire the ET 200S distributed I/O system complete with interface module (IM) and serial interface module.
2. Set up the PROFIBUS connection between the ET 200S and the SIMOTION hardware platform.
3. Set the PROFIBUS DP node address on the IM.
4. Switch on the terminating resistor at the first and last bus node.

Note

For steps 1 to 4, refer to the *ET 200S Distributed I/O System* manual.

This documentation is included in the SIMOTION SCOUT scope of supply as electronic documentation!

5. Insert the serial interface module into the SIMOTION project (refer to Chapter Integrating the ET 200S 1SI serial interface module into the SIMOTION project (Page 6473)).
6. Parameter assignment to the serial interface module.
   To assign parameters for the module, go to **HW Config** and double-click the appropriate module. The **Properties** dialog box will open. Select the **Parameters** tab to access the parameter assignment interface. There you can assign parameters for your module.
   Refer to SIMATIC manual *ET 200S 1SI Serial Interface Modules*.
7. Link the function blocks to the SIMOTION project (refer to Chapter Integrating the function blocks in the user project (Page 6474)).

8.10.2.4 Integrating the ET 200S 1SI serial interface module into the SIMOTION project

Requirement

The following requirements must be met when networking with PROFIBUS:

1. You have created a project in SIMOTION SCOUT and have inserted a rack with a SIMOTION hardware platform in the hardware configuration.
2. You have configured a PROFIBUS subnet.

Note

For information on creating a project and configuring a PROFIBUS subnet, refer to the online help for *SIMOTION SCOUT*.

The following requirements must be met when with PROFINET:

1. You have created a project in SIMOTION SCOUT and have inserted and configured a rack with a PROFINET-compatible SIMOTION device in the hardware configuration.
2. You have configured a PROFINET IO system.

Note

Consult the *SIMOTION SCOUT* online help to learn how to create a project and configure a PROFINET IO system.

Inserting a serial interface module

The description below is an example of networking via PROFIBUS.

1. In SIMOTION SCOUT, open the **User Projects** dialog box with the **Project > Open** menu command. In this dialog box, select your project and confirm your choice with **OK**.
2. Open **HW Config**.
3. In the **HW Config** window, open the **hardware catalog** with the **View > Catalog** menu command.

4. In the hardware catalog, open the **PROFIBUS DP** folder and the **ET 200S** subfolder and select, for example, the **IM 151-1 interface module** (Article No.: 6ES7 151-1AA02-0AB0 or follow-up module).
5. Use a drag-and-drop operation to place the IM 151-1 I/O device on the PROFIBUS subnet of your project. The **Properties - PROFIBUS IM 151-1 Interface** dialog box opens. In this dialog box, select the address that you have set on the IM 151-1 (see the *ET 200S Distributed I/O System*) manual) and confirm your selection with **OK**.
   The selected IM 151-1 I/O device is inserted in the project.
6. The inserted I/O device must now be fitted with your project modules. To do this, open the **IM 151-1** subfolder under the selected I/O device in the hardware catalog, and select the appropriate **modules**, beginning with a power module in slot 1. The serial interface modules with Article No. 6ES7 138-4DF0☐-0AB0 are located in the **CP** subfolder.
   In this subfolder, select the appropriate module for your application. Use a drag-and-drop operation to place these modules in the ET 200S.

   Note

   Diagnostic alarms are not enabled by default. Activate the alarms for each module in the **Properties** dialog box.

7. **Save** and **compile** your project.

8.10.2.5 Integrating the function blocks in the user project

Creating the FBs instance in the user project

The function blocks are part of the program library of the SIMOTION SCOUT engineering system. For working with the blocks, an instance has to be created in the user project for each function block used.

Example:

```
VAR_GLOBAL
...
  myInstET200S_SISend      : _ET200S_SI08_send;        // create FB instance
  myInstET200S_SIReceive   : _ET200S_SI08_receive;     // create FB instance
  myInstET200S_SIGetV24Sig : _ET200S_SI08_getV24Sig;   // create FB instance
  myInstET200S_SISetV24Sig : _ET200S_SI08_setV24Sig;   // create FB instance
...
END_VAR
```

Call (LAD representation)

The LAD representation of the individual function blocks can be found in the respective function block descriptions.

Application example

The application example is contained on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

8.10.2.6 Creating I/O variables

Overview

Communication between the SIMOTION hardware platform and the ET 200S 1SI serial interface module takes place by means of direct access to the I/O. The data interface of the serial interface module is specified in the hardware configuration at a 4-byte length, an 8-byte length or a 32-byte length.

I/O variables are used to address the direct read/write access to the I/O. You can freely assign the names of I/O variables in SIMOTION SCOUT. The I/O variables must be specified as type ARRAY [0..3] of BYTE, ARRAY [0..7] of BYTE or ARRAY [0..31] of BYTE. Assign the addresses set in the **HW Config** to them.

The names of the I/O inputs must be transferred to the function blocks as call parameters (**periIn**). The prepared data for the I/O outputs are provided by the FB as in/out parameters (**periOut**). The in/out parameter must be supplied with a variable of type ARRAY [0..3] of BYTE, ARRAY [0..7] of BYTE or ARRAY [0..31] of BYTE. After the block is called, this variable must be assigned to the I/O variables for the I/O outputs (see call example in chapter Calling function blocks (Page 6494).

Note

The variable for supplying the in/out parameters **must not** be created as a temporary variable (VAR_TEMP or local variable of a function).

The following example shows how to assign the module addresses to the I/O variables in SIMOTION SCOUT.

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperipheralinputet200s_1	PIB 256		Array	32
2	⊞ myperipheraloutputet200s_1	PQB 256	☐	Array	32
3	⊞ myperipheralinputet200s_2	PIB 288		Array	32
4	⊞ myperipheraloutputet200s_2	PQB 288	☐	Array	32

Figure 8-60 Address assignment in SIMOTION SCOUT for two ET 200S 1SI serial interface modules with 32-byte data interface

Note

For additional information, see the following sources:

- *SIMOTION SCOUT* online help
- Programming manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* programming manual
  - *SIMOTION MCC, Motion Control Chart* programming manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* programming manual

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation!

8.10.3 Function blocks of the ET 200S 1SI serial interface module

8.10.3.1 Overview of function blocks

This section describes all of the function blocks (FBs) required for communication between a SIMOTION hardware platform and a serial interface module.

The function blocks form the software interface between the SIMOTION hardware platform and the serial interface module. They must be called repeatedly (in cycles) from the user program.

The following function blocks are available for the 4-, 8- and 32-byte data interface of the ET 200S 1SI serial interface module:

- Function blocks _ET200S_SIxx_send (Page 6477)
- Function blocks _ET200S_SIxx_receive (Page 6481)
- Function blocks _ET200S_SIxx_getV24Sig (Page 6485)
- Function blocks _ET200S_SIxx_setV24Sig (Page 6487)
- Function blocks _ET200S_SIxx_flowXon (Page 6488)

- Function blocks _ET200S_SIxx_flowRts (Page 6490)
- Function blocks _ET200S_SIxx_flowV24 (Page 6492)

Note

xx stands for 04, 08 and 32, e.g.:

- Function block **_ET200S_SI04_send**
- Function block **_ET200S_SI08_send**
- Function block **_ET200S_SI32_send**

The designation with **..._SIxx_...** is retained in the following.

Note

The SIMOTION identifiers have changed as of V4.0. A comparison of the SIMOTION and SIMATIC identifiers can be found in the Appendix SIMOTION and SIMATIC names (Page 6502) in the table "List of parameters".

The provided function blocks can be used to control one or more serial interfaced modules.

8.10.3.2 Function blocks _ET200S_SIxx_send

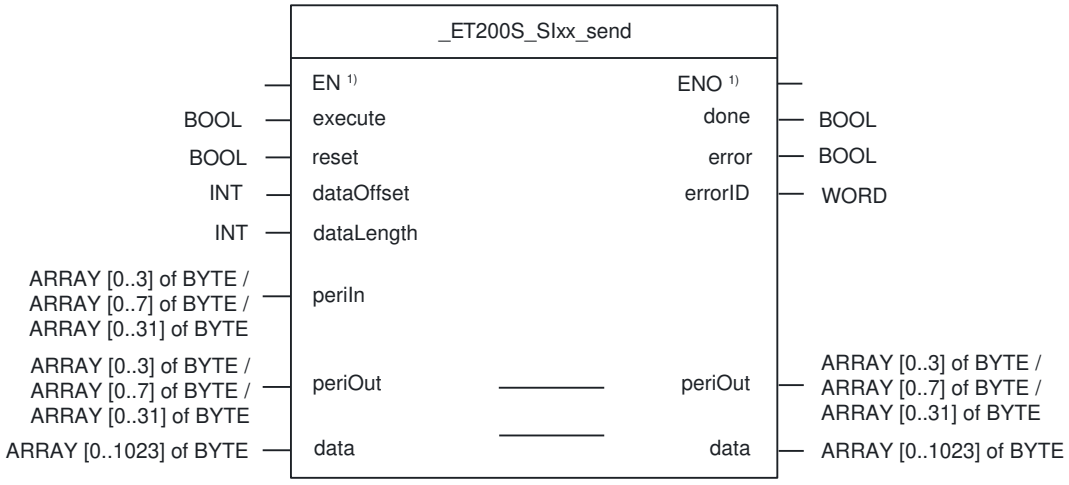
Task

The **_ET200S_SIxx_send** function block can be used to send data from the **data** send data array to a communication partner. 1024 bytes are available.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_send** function block for the 4-byte data interface, the **_ET200S_SI08_send** function block for the 8-byte data interface, and the **_ET200S_SI32_send** function block for the 32-byte data interface of the serial interface module.

For the transfer, you can use the 3964 (R) protocol procedure or ASCII driver.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-136 _ET200S_SIxx_send parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
dataOffset	IN	INT	Offset of the first element to be sent	Entered	Checked
dataLength	IN	INT	Number of elements to be sent 1 ≤ dataLength ≤ 224	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..1023] of BYTE	Send data array	Entered	Checked
done	OUT	BOOL	Job completed without errors	Checked	Entered

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The **periOut** parameter must be provided with an array of type **ARRAY [0..3] of BYTE** or **ARRAY [0..7] of BYTE**. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System*, chapter "Commissioning and diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Allocation in the data storage area

The data to be sent is transferred to FB **_ET200S_SIxx_send** in the **data** parameter (VAR_IN_OUT) as ARRAY of BYTE. At the beginning of the send operation, the data is copied to a local ARRAY and transferred from there to the serial module.

Note

Once the data has been entered in the static memory of the send FB, you can modify the variable created in the **data** parameter. This does not affect the data to be sent.

_ET200S_SIxx_send time sequence diagram

The following figure illustrates the behavior of the **done** and **error** parameters according to the input circuit of **execute** and **reset**.

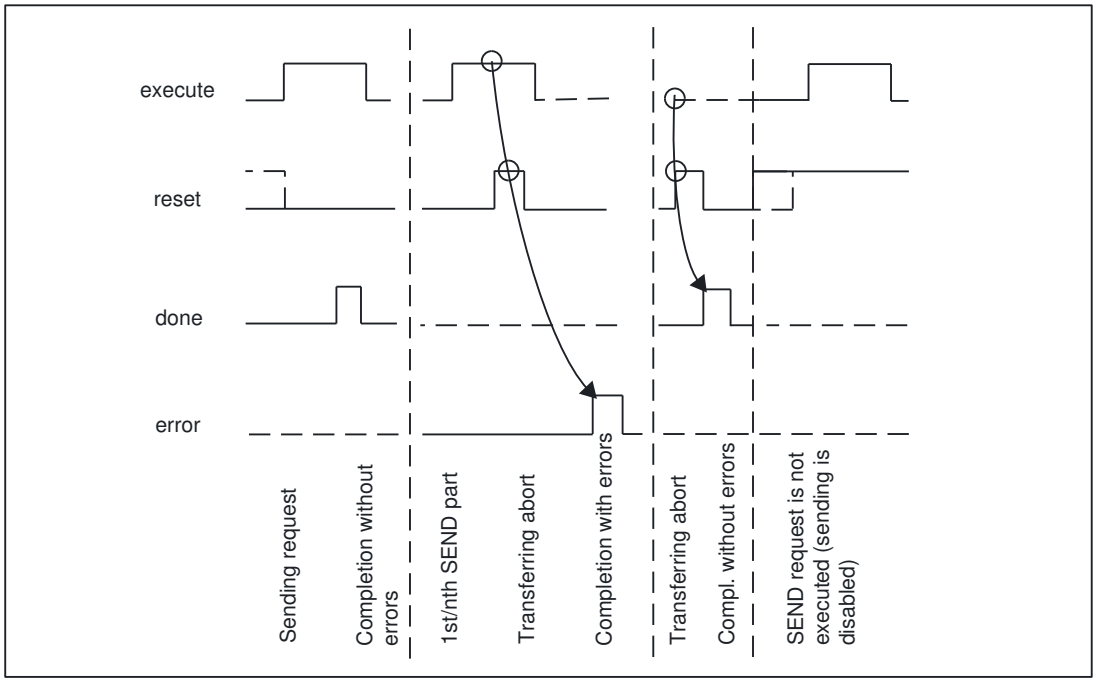


Figure 8-61 _ET200S_SIxx_send time sequence diagram

Note

The **execute** input parameter is edge-triggered. A positive edge at the **execute** input parameter is sufficient.

Task integration (call)

The **_ET200S_SIxx_send** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

The SIMOTION device sends data to a communications partner

The **_ET200S_SIxx_send** FB transfers a data block, which is specified by the following parameters for the serial interface module:

- **data** corresponds to the data array containing the send data
- **dataOffset** corresponds to the array index containing the first send byte
- **dataLength** corresponds to the amount of data to be sent in bytes (maximum of 224 bytes)

The **_ET200S_SIxx_send** FB must be called repeatedly by a program. The send job can only be executed by cyclically calling the send FB.

A positive edge at the **execute** input parameter initiates the transfer. Depending on the data volume, data may be transferred over several calls.

Note

A subsequent transfer at the **execute** input parameter can only be successfully triggered when the previous send job is completed with the **done** output signal ("Job completed without errors").

The active transfer request can be aborted by setting the **reset** parameter to TRUE. This will reset the **_ET200S_SIxx_send** FB to its initial state. The send operation will remain disabled as long as the signal state at the **reset** parameter is TRUE.

Status and error display on _ET200S_SIxx_send

The **done** output parameter indicates that the job has been completed without errors. The **error** output parameter indicates that an error has occurred. When an error occurs, the corresponding event class/number is displayed in the **errorID** output parameter (see table "Parameter _ET200S_SIxx_send"). If no errors have occurred, **errorID** has a value of "0". The parameters **Done** and **error**/**errorID** are also displayed on **reset** of the **_ET200S_SIxx_send** FB.

Parameters **done**, **error**, and **errorID** are present for one block pass only.

8.10.3.3 Function blocks _ET200S_SIxx_receive

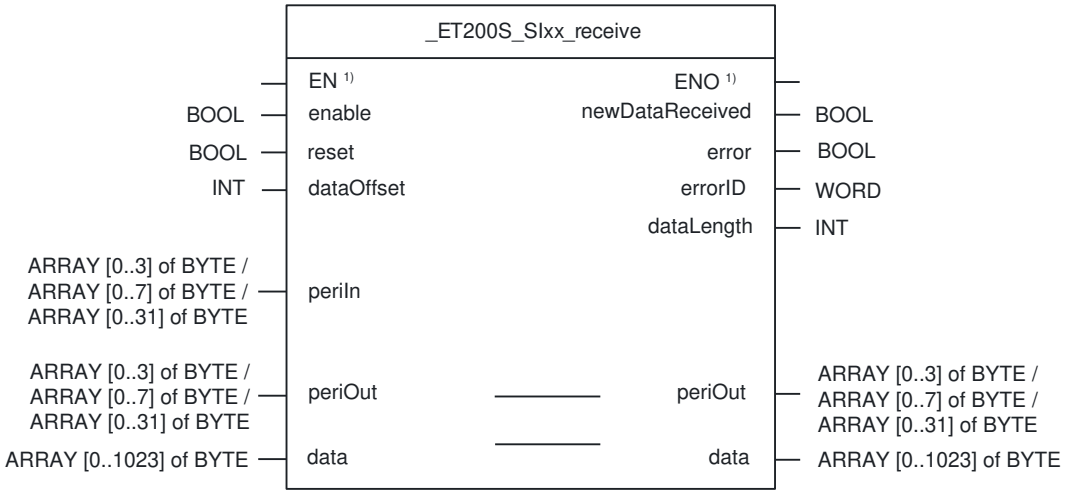
Task

The **_ET200S_SIxx_receive** function block can be used to receive data from a communication partner in the **data** receive data array. 1024 bytes are available.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_receive** function block for the 4-byte data interface, the **_ET200S_SI08_receive** function block for the 8-byte data interface, and the **_ET200S_SI32_receive** function block for the 32-byte data interface of the serial interface module.

For the transfer, you can use the 3964 (R) protocol procedure or ASCII driver.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-137 _ET200S_SIxx_receive parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
enable	IN	BOOL	Receive enable	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
dataOffset	IN	INT	Offset of the first element to be received	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
data	IN/OUT	ARRAY[0..1023] of BYTE	Receive data array	Checked	Entered
newDataReceived	OUT	BOOL	New data has been received	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered

Name	P-Type [1)]	Data type	Meaning	Actions performed by user	Actions performed by block
errorID	OUT	WORD	Error specification For **error**=TRUE, the error information (event class and number) is displayed in the **errorID** parameter.[3)]	Checked	Entered
dataLength	OUT	INT	Data volume received in bytes	Checked	Entered

1) Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

2) **Note:** The **periOut** parameter must be provided with an array of type **ARRAY [0..3] of BYTE** or **ARRAY [0..7] of BYTE**. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

3) For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System*, chapter "Commissioning and diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Allocation in the data storage area

During the receive operation, the data to be received is stored temporarily in a local ARRAY. After completion of the data transfer from the serial interface module to the SIMOTION device, the data is made available in the **data** parameter (VAR_IN_OUT) of the **_ET200S_SIxx_receive** FB.

_ET200S_SIxx_receive time sequence diagram

The following figure illustrates the behavior of the **newDataReceived**, **dataLength**, and **error** parameters according to the input circuit of **enable** and **reset**.

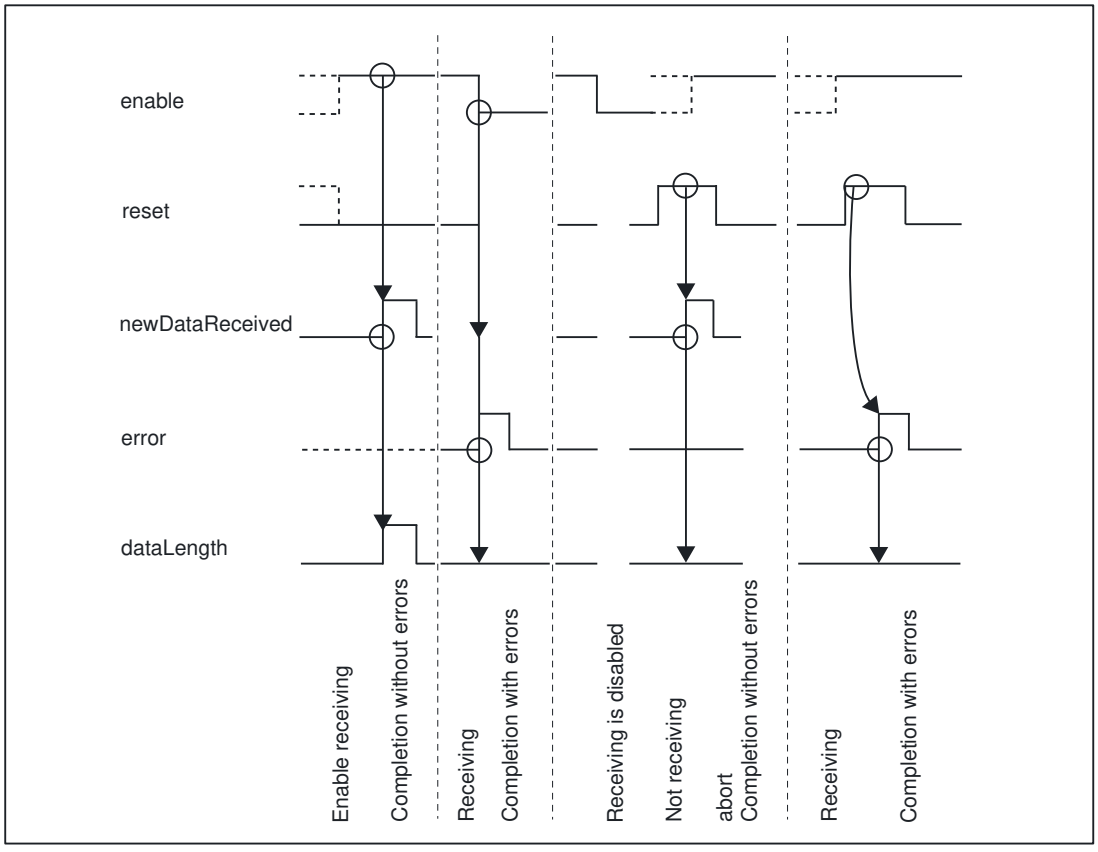


Figure 8-62 _ET200S_SIxx_receive time sequence diagram

Task integration (call)

The **_ET200S_SIxx_receive** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

SIMOTION device receives data from a communications partner

The **_ET200S_SIxx_receive** FB transfers a data block, which is specified by the **data** and **dataOffset** parameters, from the serial interface module to a SIMOTION hardware platform. The **_ET200S_SIxx_receive** FB must be called repeatedly by a program. The receive job can only be executed by cyclically calling the receive FB.

Receiving of data is enabled with static signal state TRUE in the **enable** parameter. An active data transfer is aborted by a FALSE signal state in the **enable** parameter. The aborted receive request is terminated with an error message at the **errorID** output parameter. The receive operation will remain disabled as long as the signal state at the **enable** parameter is FALSE. Depending on the data volume, data may be transferred over several calls.

The active transfer request can be aborted by setting the **reset** parameter to TRUE. This will reset the **_ET200S_SIxx_receive** FB to its initial state. The receive operation will remain disabled as long as the signal state at the **reset** parameter is TRUE.

Status and error display on _ET200S_SIxx_receive

The **newDataReceived** output parameter indicates that the job has been completed without errors. The amount of data received is indicated in the **dataLength** parameter. The **error** output indicates that an error has occurred. When an error occurs, the corresponding event class/ number is displayed in the **errorID** output parameter (see table "Parameter _ET200S_SIxx_receive"). If no errors have occurred, **errorID** has a value of "0". The parameters **newDataReceived** and **error**/**errorID** are also displayed on **reset** of the **_ET200S_SIxx_receive** FB.

Parameters **newDataReceived, dataLength**, **error**, and **errorID** are present for one block pass only.

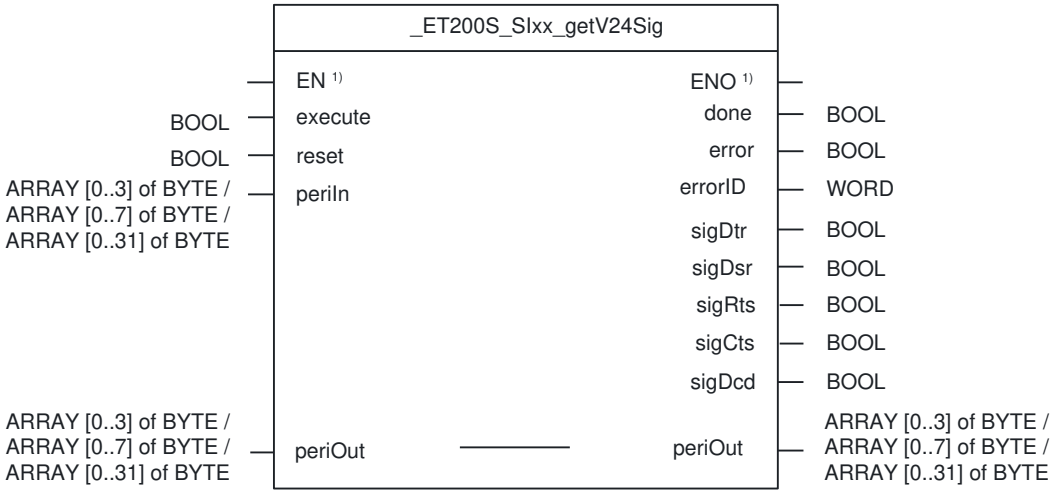
8.10.3.4 Function blocks _ET200S_SIxx_getV24Sig

Task

When an **ASCII driver** is specified, the **_ET200S_SIxx_getV24Sig** function block reads the RS 232C secondary signals and provides them to the user as output parameters.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_getV24Sig** function block for the 4-byte data interface, the **_ET200S_SI08_getV24Sig** function block for the 8-byte data interface, and the **_ET200S_SI32_getV24Sig** function block for the 32-byte data interface of the serial interface module.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-138 _ET200S_SIxx_getV24Sig parameters

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module [2]	Checked and entered on the I/O variable for the I/O outputs	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered
sigDtr	OUT	BOOL	Data terminal ready	Checked	Entered
sigDsr	OUT	BOOL	Data set ready	Checked	Entered
sigRts	OUT	BOOL	Request to send	Checked	Entered
sigCts	OUT	BOOL	Clear to send	Checked	Entered
sigDcd	OUT	BOOL	Data carrier detected	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The periOut parameter must be provided with an array of type ARRAY [0..3] of BYTE or ARRAY [0..7] of BYTE. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System*, chapter "Commissioning and Diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Task integration (call)

The **_ET200S_SIxx_getV24Sig** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

RS-232-C secondary signals are updated each time the FB is called (cyclic polling).

Note

A minimum pulse time is necessary for a signal change to be identified. Determining factors are the cycle time (SIMOTION device), the update time on the serial interface module and the response time of the communications partner.

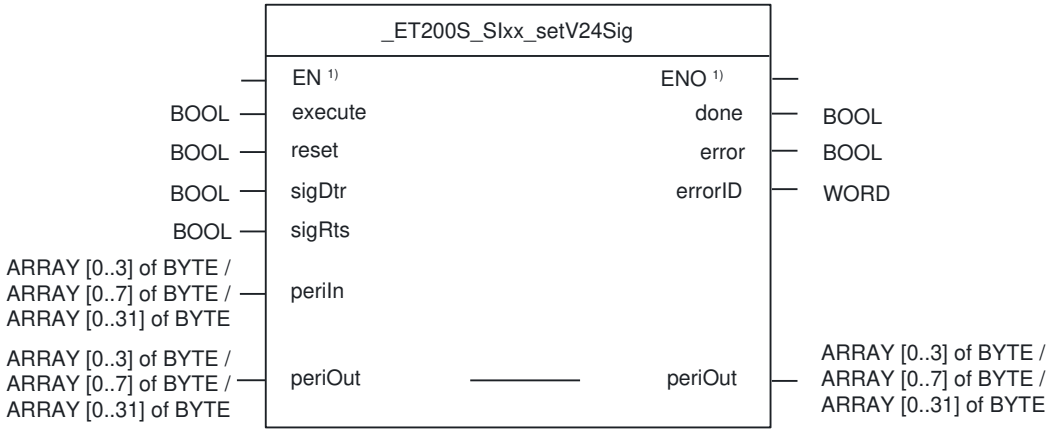
8.10.3.5 Function blocks _ET200S_SIxx_setV24Sig

Task

The **_ET200S_SIxx_setV24Sig** function block enables you to set or reset the RS 232C secondary signals when an **ASCII driver** is specified.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_setV24Sig** function block for the 4-byte data interface, the **_ET200S_SI08_setV24Sig** function block for the 8-byte data interface, and the **_ET200S_SI32_setV24Sig** function block for the 32-byte data interface of the serial interface module.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-139 _ET200S_SIxx_setV24Sig parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
sigDtr	IN	BOOL	Data terminal ready	Entered	Checked
sigRts	IN	BOOL	Request to send	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The periOut parameter must be provided with an array of type ARRAY [0..3] of BYTE or ARRAY [0..7] of BYTE. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System,* section "Commissioning and Diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Task integration (call)

The **_ET200S_SIxx_setV24Sig** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

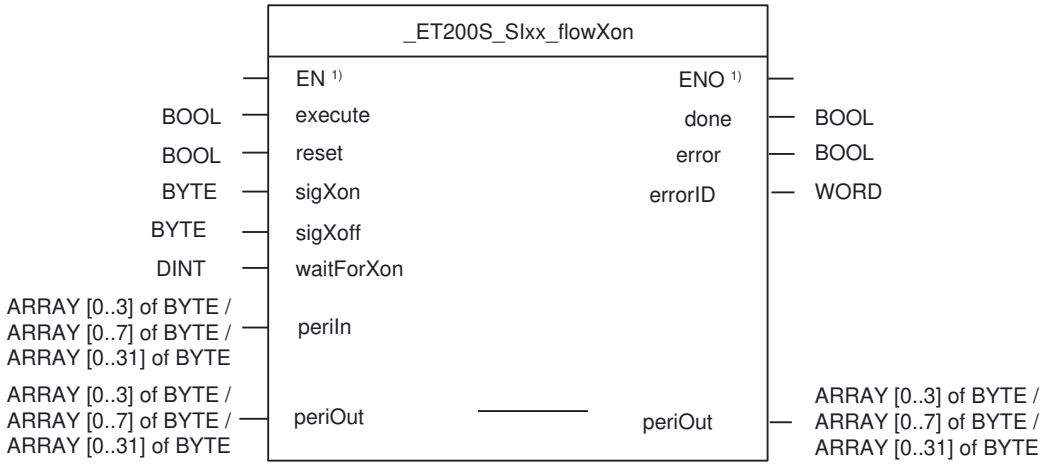
8.10.3.6 Function blocks _ET200S_SIxx_flowXon

Task

The **_ET200S_SIxx_flowXon** function block enables you to modify parameters on the module independently of the parameter assignment interface if **"XON/XOFF" data flow control** has been specified for the module.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_flowXon** function block for the 4-byte data interface, the **_ET200S_SI08_flowXon** function block for the 8-byte data interface, and the **_ET200S_SI32_flowXon** function block for the 32-byte data interface of the serial interface module.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-140 _ET200S_SIxx_flowXon parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
sigXon	IN	BYTE	Code for XON character	Entered	Checked
sigXoff	IN	BYTE	Code for XOFF character	Entered	Checked
waitFor-Xon	IN	DINT	Delay time for XON after XOFF Value range: 20 ms ≤ waitForXon ≤ 655350 ms	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The periOut parameter must be provided with an array of type ARRAY [0..3] of BYTE or ARRAY [0..7] of BYTE. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System,* section "Commissioning and Diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Task integration (call)

The **_ET200S_SIxx_flowXon** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

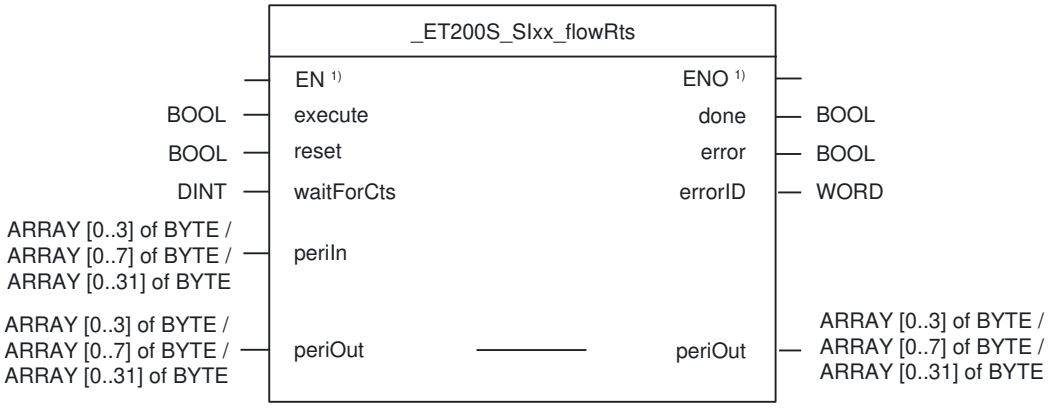
8.10.3.7 Function blocks _ET200S_SIxx_flowRts

Task

The **_ET200S_SIxx_flowRts** function block enables you to modify parameters on the module independently of the parameter assignment interface if **"RTS/CTS" data flow control** has been specified for the module.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_flowRts** function block for the 4-byte data interface, the **_ET200S_SI08_flowRts** function block for the 8-byte data interface, and the **_ET200S_SI32_flowRts** function block for the 32-byte data interface of the serial interface module.

Schematic LAD representation



1) LAD-specific parameters

Parameter description

Table 8-141 _ET200S_SIxx_flowRts parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
waitForCts	IN	DINT	Delay time for CTS=ON Value range: 20 ms ≤ waitForXon ≤ 655350 ms	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The periOut parameter must be provided with an array of type ARRAY [0..3] of BYTE or ARRAY [0..7] of BYTE. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System,* chapter "Commissioning and Diagnostics", and *ET 200S 1SI Serial Interface Modules*

Task integration (call)

The **_ET200S_SIxx_flowRts** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

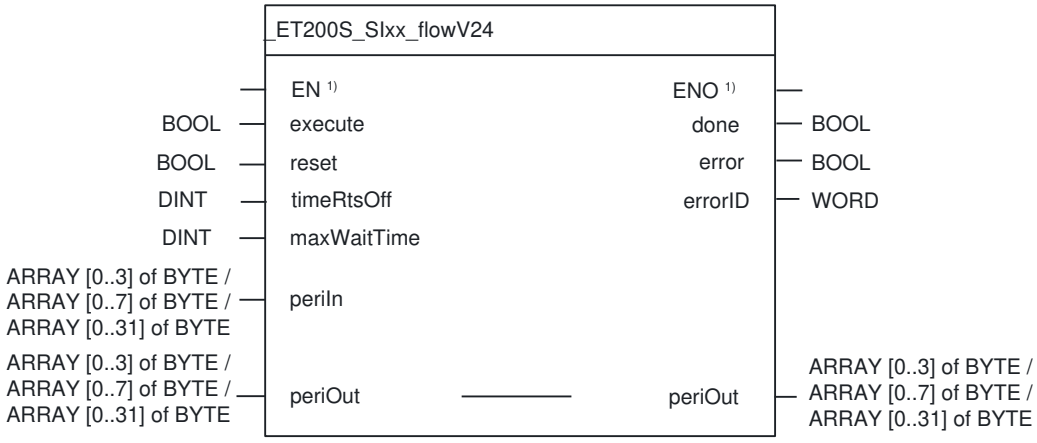
8.10.3.8 Function blocks _ET200S_SIxx_flowV24

Task

The **_ET200S_SIxx_flowV24** function block enables you to modify parameters on the module independently of the parameter assignment interface if the **"Automatic use of V24 signal" data flow control** has been specified for the module.

Depending on which data interface you have chosen in the hardware configuration, you use the **_ET200S_SI04_flowV24** function block for the 4-byte data interface, the **_ET200S_SI08_flowV24** function block for the 8-byte data interface, and the **_ET200S_SI32_flowV24** function block for the 32-byte data interface of the serial interface module.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

Table 8-142 _ET200S_SIxx_flowV24 parameters

Name	P-Type 1)	Data type	Meaning	Actions performed by user	Actions performed by block
execute	IN	BOOL	Initiates job with positive edge	Entered	Checked
reset	IN	BOOL	Job cancellation	Entered	Checked
timeRtsOff	IN	DINT	Time that must elapse after the transfer before RTS is disabled Value range: 0 ≤ timeRtsOff ≤ 655350 ms	Entered	Checked
maxWait-Time	IN	DINT	Wait time for CTS=ON by partner after RTS=ON has been set Value range: 0 ≤ maxWaitTime ≤ 655350 ms	Entered	Checked
periIn	IN	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Transfer I/O inputs of the serial interface module to FB	Transfer I/O variables of the I/O inputs of the serial module to the FB	Checked
periOut	IN/OUT	ARRAY [0..3] of BYTE / ARRAY [0..7] of BYTE / ARRAY [0..31] of BYTE	Prepared data of the FB for the I/O outputs of the serial interface module 2)	Checked and entered on the I/O variable for the I/O outputs	Entered
done	OUT	BOOL	Job completed without errors	Checked	Entered

Name	P-Type [1]	Data type	Meaning	Actions performed by user	Actions performed by block
error	OUT	BOOL	Job completed with errors	Checked	Entered
errorID	OUT	WORD	Error specification For **error** = TRUE, the error information (event class and number) is displayed in the **errorID** parameter. [3]	Checked	Entered

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] **Note:** The periOut parameter must be provided with an array of type ARRAY [0..3] of BYTE or ARRAY [0..7] of BYTE. This variable must be static and may be initialized only once (e.g. after switching the operating mode from STOP to RUN). We recommend that a global array is created for VAR_GLOBAL (do not create any temporary array for VAR_TEMP). After calling the FB, this array must be assigned the I/O variables for the I/O outputs of the module. See call example in section "Calling function blocks". See also chapter *Integration of ST in SIMOTION* in the *SIMOTION ST Structured Text* Programming Manual.

[3] For error information, refer to SIMATIC manuals *ET 200S Distributed I/O System,* chapter "Commissioning and diagnostics", and *ET 200S 1SI Serial Interface Modules*.

Task integration (call)

The **_ET200S_SIxx_flowV24** FB must be called cyclically in the **BackgroundTask** or the **TimerInterruptTask**. Calling in the **SystemInterruptTask** is not permitted. Calling the function block in the **IPOSynchronousTask** is not recommended for runtime reasons.

8.10.3.9 Calling function blocks

In order to be able to work with the function blocks in your user program, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see the following program segment, e.g. create instance for the **_ET200S_SI08_send** function block).
2. Create an array for the in/out parameters of the FB.
3. Call instance of the function block.
4. Transfer input parameters.
5. The output parameters of the FB are accessed with <instance name of FB>. <name of output parameter>.
6. Data prepared by the FB for the I/O outputs are assigned to the array of the I/O variables created in step 2.

Note

The call example is an extract from the supplied E_SI_8By application example, which is contained on the "SIMOTION Utilities & Applications" CD-ROM.

If you want to control more than one serial interface module, you must create a new variable for the FB instances with a new name for each additional module used.

Call example

```
UNIT E_SI_8BY;

INTERFACE

VAR_GLOBAL

    myExecSend            : BOOL;                     // trigger send task
    myAbortSend           : BOOL;                     // cancel send order
    mySendDone            : BOOL;                     // send: completed

    myInstET200S_SISend   : _ET200S_SI08_send;       // create FB instance                    (1)
    mySendDataArray       : ARRAY [0..1023] OF BYTE;  // send data array 1024 bytes
END_VAR

    PROGRAM ExampleET200S_SI;                         // program for BackgroundTask

END_INTERFACE

IMPLEMENTATION

PROGRAM ExampleET200S_SI
// program for backgroundTask
VAR
    myOutputArrayET200S_1 : ARRAY [0..7] OF BYTE;  // array for ET200S output data
END_VAR                                                                                   (2)

// CALL FB INSTANCE TO SEND

    myInstET200S_SISend (                                                                 (3)

    execute          := myExecSend,                  // trigger order
    reset            := myAbortSend,                 // order cancellation                 (4)
    dataOffset      := myDataOffsetSend,            // data offset
    dataLength       := myDataLengthSend,            // amount of data to be sent
    periIn           := myPeripheralInputET200s_1,   // I/O variable of I/O inputs
    periOut          := myOutputArrayET200S_1,       // output data array
    data             := mySendDataArray              // send data array
    );

    mySendDone       := myInstET200S_SISend.done;
```

```
// TRANSFER DATA TO ET200S                                                    (5)
   myPeripheralOutputET200s_1 := myOutputArrayET200S_1; // array for output data

END_PROGRAM            // end program for backgroundTask

END_IMPLEMENTATION                                                            (6)
```

Note

The ExampleET200S_SI program must be assigned in the execution system.

8.10.3.10 Data consistency

When sending data

Once a request is initiated by a positive edge at the **execute** input, the data to be sent are copied to the static memory area of the send FB. This means that once the FB call has ended, the send data array can be written to again for the next send request on a positive edge. The data are retained as consistent data within the send FB.

Note

During the copy operation to the static memory area of the FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

When receiving data

Once the receive request is complete, the data are copied over to the receive buffer in a block from the static memory area of the receive FB. This means that once the FB call has ended, either all data are entered in the receive buffer (**newDataReceived** = TRUE) or no data are entered in the receive buffer (**newDataReceived** = FALSE).

Note

During the copy operation from the static memory area of the receive FB, data consistency cannot be guaranteed if the send/receive data areas are accessed in a higher priority task.

8.10.3.11 Application example for the ET 200S 1SI serial interface module

Task

This example shows how to:

- use the **_ET200S_SI08_send** function block to send data from the send data array to a communication partner (in the example, the second serial interface module).
- use the **_ET200S_SI08_receive** function block to receive data in the receive data array.

In the example program, the first serial interface module is used as the sender, and the second is used as the receiver. The **_ET200S_SI08_send** FB is used to transfer the send data to the second interface module.

The **_ET200S_SI08_receive** FB reads out the data received from the first serial interface module and copies it to the receive data array.

In this example, two serial interface modules with 8-byte data interface are used.

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you have to adapt the hardware configuration.

Assigning module parameters

Proceed as follows:

1. Open your project in SIMOTION SCOUT.
2. Open the hardware configuration in SIMOTION SCOUT.
3. Configure your hardware station with the two ET 200S 1SI serial interface modules (8-byte data interface), see chapter Integrating the ET 200S 1SI serial interface module into the SIMOTION project (Page 6473).
4. The serial interface modules are located below the IM 151-1 PROFIBUS interface module. Double-click the first or second **1SI ASCII (8 byte)** to open the **Properties** dialog for this module. Select the **Parameters** tab to access the parameter assignment interface.
5. Select the following setting (leave all other settings as they are): **Diagnostic alarm: activate.** Confirm your selection with **OK**.
6. Save the hardware configuration with the **Station > Save and compile** menu command.
7. Download the hardware configuration with the **Target system > Download to module** menu command.
   The red "SF" LED of the IM 151-1 turns on and off after the module parameter assignment is downloaded without errors.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

- You can adapt the configuration in the example to the available hardware (PROFIBUS DP address).
- You can adapt the configuration of the hardware to the example (PROFIBUS DP address).

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration: PROFIBUS DP addresses.
3. Check the module addresses (hardware configuration) against the I/O addresses of the controller in SIMOTION SCOUT and the module addresses in the program (myModuleAddr_1, myModuleAddr_2).
4. Save and compile the example project. Then, you can download the example to the SIMOTION device and switch to **RUN** mode.

Sequence of the application example

Input icons used

Table 8-143 Input icons used

Symbol	Data type	Description
mySelectPointToPoint	BOOL	Select point-to-point communication
myExecSend	BOOL	Initiate send request
mySendOrder1	BOOL	Select send: Request 1
mySendOrder2	BOOL	Select send: Request 2
myAbortSend	BOOL	Abort send request
myEnableToReceive	BOOL	Receive enable
myReceiveOrder1	BOOL	Select receive: Request 1
myReceiveOrder2	BOOL	Select receive: Request 2
myAbortReceive	BOOL	Abort receive request
myDataLengthSend	INT	Length/number of data to be sent
myDataOffsetSend	INT	1. byte in the send data array
myDataOffsetReceive	INT	1. byte in the receive data array
myModuleAddr_1	DINT	Module address first ET 200S 1SI
myModuleAddr_2	DINT	Module address second ET 200S 1SI
mySendDataArray	ARRAY [0..1023] of BYTE	Send data array

Output icons used

Table 8-144 Output icons used

Symbol	Data type	Description
mySendDone	BOOL	Send: Complete
mySendError	BOOL	Send: Error display
mySendErrorId	WORD	Send: Error number
myNewDataReceived	BOOL	Receive: New data have been received
myReceiveError	BOOL	Receive: Error display
myReceiveErrorId	WORD	Receive: Error number
myDiagAlarm	BOOL	Diagnostic alarm occurred
myProcessAlarm	BOOL	Process alarm occurred
myReceiveDataArray	ARRAY [0..1023] of BYTE	Receive data array

Note

You can monitor and modify the input and output variables used in the programming example in the INTERFACE area of the unit (under VAR_GLOBAL); alternatively, you can assign real inputs and outputs to the input and output variables in your unit.

For the "point-to-point communication" application example, set the "mySelectPointToPoint" input to TRUE. This will call the function blocks contained in the application example.

Receiving data:

To receive data, you must set the "myEnableToReceive" input to TRUE (static signal). If receive request 1 is enabled ("myReceiveOrder1" = TRUE), the data is stored in the "receiveDataArray" data array starting with the "receiveDataArray[0]" array element (data offset is 0). If request 2 is enabled ("myReceiveOrder2" = TRUE), the data is stored in the "receiveDataArray" data array starting with the "receiveDataArray[20]" array element (data offset is 20).
If "myNewDataReceived" = TRUE, this indicates that new data has been received. This signal is present for one cycle only.
If an error occurred during the transfer ("myReceiveError" = TRUE), the error code is stored in the "myReceiveErrorId" variable. The error signals are deleted when you set input "myAbortReceive" = TRUE.

Sending data:

You can use the "mySendOrder1" and "mySendOrder2" inputs to select between two send requests:

- Request 1 sends 10 bytes of data from the "mySendDataArray" data array starting from the "mySendDataArray[0]" array element up to the "mySendDataArray[9]" array element
- Request 2 sends 20 bytes of data from the "mySendDataArray" data array starting from the "mySendDataArray[20]" array element up to the "mySendDataArray[39]" array element

The data is sent to the communication partner if the "myExecSend" input detects a signal change from FALSE to TRUE (positive edge).
If output signal "mySendDone" = TRUE, the send request has been completed. A new request can be sent if the "myExecSend" input signal detects a signal change from FALSE to TRUE.

If an error occurred during the transfer ("mySendError" = TRUE), the error code is stored in the "mySendErrorId" variable. The error signals are deleted when you set input "myExecSend" = FALSE.

If the signal state at the "myAbortSend" or "myAbortReceive" input is TRUE, the send request or receive request, respectively, is aborted. If the signal state remains TRUE, sending and receiving of data is disabled.

8.10.4 Alarm processing

Overview

Pending error messages are processed and evaluated differently in a SIMOTION system than in a SIMATIC system. Diagnostic alarms are not enabled by default. Activate the alarms for the relevant module in the hardware configuration (refer to chapter Integrating the ET 200S 1SI serial interface module into the SIMOTION project (Page 6473)).

If you have parameterized diagnostic alarms, then you should program the alarm processing sequence according to the principle presented below.

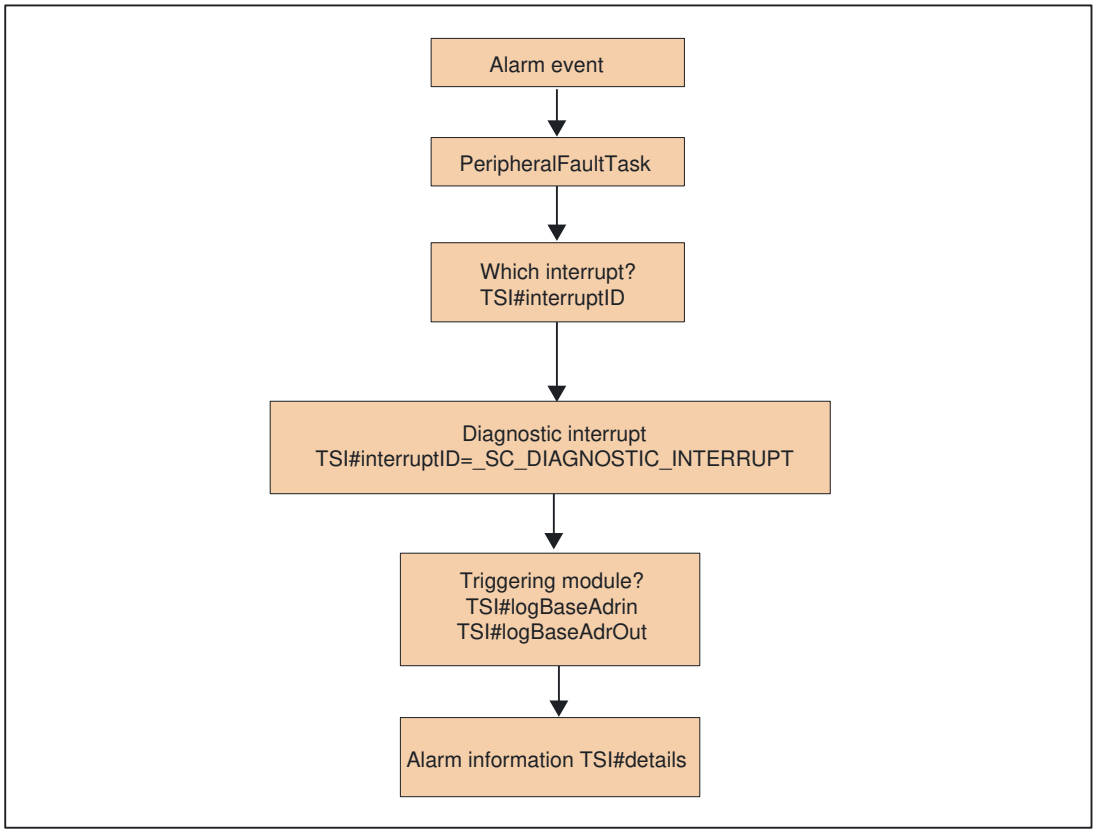


Figure 8-63 Alarm processing for the serial interface module

Alarm evaluation

Alarms originating from the I/O are evaluated in the **PeripheralFaultTask**. When the **PeripheralFaultTask** is started, the **Taskstartinfo** is made available, which you can evaluate in the user program.

The **Taskstartinfo** of **PeripheralFaultTask** is comparable to the local data of OB82 in the SIMATIC system.

Table 8-145 Meaning of the Taskstartinfo

Task	TSI		Remarks
PeripheralFaultTask	DT	TSI#startTime	Start time of the task
	UDINT	TSI#interruptID	Identifies the triggering event: • _SC_PROCESS_INTERRUPT • _SC_DIAGNOSTIC_INTERRUPT • _SC_STATION_DISCONNECTED • _SC_STATION_RECONNECTED
	DINT	TSI#logBaseAdrIn	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an input area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logBaseAdrOut	Logical base address if a process alarm (PRAL) or a diagnostic alarm (DAL) was caused by an output area on the module, otherwise _SC_INVALID_ADDRESS
	DINT	TSI#logDiagAdr	Diagnostic address of a DP slave if the alarm was caused by a station failure or station recovery of an associated DP slave, otherwise _SC_INVALID_ADDRESS
	DWORD	TSI#details	Detail information (bit fields)

Definition of a diagnostic alarm

If the user program is to respond to an internal or external error, you can set the parameters for a diagnostic alarm that will interrupt the cyclical program of the SIMOTION device.

Events triggering a diagnostic alarm

The criteria (events) that trigger diagnostic alarms in a SIMOTION system are the same as in a SIMATIC system.

For a more detailed description, refer to SIMATIC manual *ET 200S 1SI Serial Interface Module*, "Diagnostics" chapter.

Responses to a diagnostic alarm

If a diagnostic alarm occurs, the following take place:

- Diagnostic data are written to **TSI#details** variable in the Taskstartinfo of **PeripheralFaultTask**.
- In the **PeripheralFaultTask**, you can read out and save the 4 bytes of diagnostic data.
- The group error LED (SF) of the serial interface module illuminates. The group error LED (SF) is extinguished as soon as the error has been remedied.

Bit assignment

The **TSI#details** variable is assigned in the same way as in a SIMATIC system.

Note

For a more detailed description, refer to the SIMATIC manual *ET 200S 1SI Serial Interface Module,* "Diagnostics" chapter.

8.10.5 Appendix

8.10.5.1 SIMOTION and SIMATIC names

The following table contains a comparison of the SIMOTION and the SIMATIC names.

Table 8-146 List of parameters

Name in SIMOTION system, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
Function block parameters		
_ET200S_SI04_send, _ET200S_SI08_send, _ET200S_SI32_send	**FB P_SEND (FB 3)**	**_FB_ET200S_SI4_send, _FB_ET200S_SI8_send, -**
execute	REQ	request
reset	R	abort
dataOffset	DBB_NO	dataOffset
dataLength	LEN	dataLength
periIn	-	inputInterface
periOut	-	outputInterface
data	DB_NO	data
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber

Name in SIMOTION system, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
_ET200S_SI04_receive, _ET200S_SI08_receive, _ET200S_SI32_receive	**FB P_RCV (FB 2)**	**_FB_ET200S_SI4_receive, _FB_ET200S_SI8_receive, -**
enable	EN_R	enable
reset	R	abort
dataOffset	DBB_NO	dataOffset
periIn	-	inputInterface
periOut	-	outputInterface
data	DB_NO	data
newDataReceived	NDR	newDataReceived
error	ERROR	error
dataLength	LEN	dataLength
errorID	STATUS	errorNumber
_ET200S_SI04_getV24Sig, _ET200S_SI08_getV24Sig, _ET200S_SI32_getV24Sig	**FB V24_STAT (FB 4)**	**_FB_ET200S_SI4_getV24State, _FB_ET200S_SI8_getV24State, -**
execute	REQ	request
reset	R	abort
periIn	-	inputInterface
periOut	-	outputInterface
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber
sigDtr	DTR_OUT	signalDTR
sigDsr	DSR_IN	signalDSR
sigRts	RTS_OUT	signalRTS
sigCts	CTS_IN	signalCTS
sigDcd	DCD_IN	signalDCD
_ET200S_SI04_setV24Sig, _ET200S_SI08_setV24Sig, _ET200S_SI32_setV24Sig	**FB V24_SET (FB 5)**	**_FB_ET200S_SI4_setV24signals, _FB_ET200S_SI8_setV24signals, -**
execute	REQ	request
reset	R	abort
sigDtr	DTR	signalDTR
sigRts	RTS	signalRTS
periIn	-	inputInterface
periOut	-	outputInterface
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber

Name in SIMOTION system, V4.0 and higher (command library in SCOUT)	Name in the SIMATIC system	Name in the SIMOTION system up to V3.2 (SIMOTION Function Library)
_ET200S_SI04_flowXon, _ET200S_SI08_flowXon, _ET200S_SI32_flowXon	**FB F_XON (FB 6)**	**_FB_ET200S_SI4_FXON, _FB_ET200S_SI8_FXON, -**
execute	REQ	request
reset	R	abort
sigXon	XON	signalXON
sigXoff	XOFF	signalXOFF
waitForXon	WAIT_FOR_XON	waitForXON
periIn	-	inputInterface
periOut	-	outputInterface
done	DONE	done
error	ERROR	error
errorID	STATUS	errorNumber
_ET200S_SI04_flowRts, _ET200S_SI08_flowRts, _ET200S_SI32_flowRts	**FB F_RTS (FB 7)**	**_FB_ET200S_SI4_FRTS, _FB_ET200S_SI8_FRTS, -**
execute	REQ	request
reset	R	abort
waitForCts	WAIT_FOR_CTS	waitForCTS
periIn	-	inputInterface
periOut	-	outputInterface
done	DONE	done
error	ERROR	error
errorID	STATE	errorNumber
_ET200S_SI04_flowV24, _ET200S_SI08_flowV24, _ET200S_SI32_flowV24	**FB F_V24 (FB 8)**	**_FB_ET200S_SI4_FV24, _FB_ET200S_SI8_FV24, -**
execute	REQ	request
reset	R	abort
timeRtsOff	TIME_RTS_OFF	timeRTSOff
maxWaitTime	DATA_WAIT_TIME	maxWaitTime
periIn	-	inputInterface
periOut	-	outputInterface
done	DONE	done
error	ERROR	error
errorID	STATE	errorNumber

8.10.5.2 List of abbreviations

Table 8-147 Abbreviations

Abbreviation	Meaning
DP	Distributed I/O
FB	Function block
HW	Hardware
IM	Interface Module (SIMATIC S7-300 interface module)
IN	Input parameter
IN/OUT	In/out parameter
I/O	Input/Output
LAD	Ladder Logic
LED	Light-emitting diode (LED display)
MCC	Motion Control Chart
OB	Organization block
OUT	Output parameter
SF	System Fault (System error)
ST	Structured Text

8.11 Standard Function for ASIsafe Safety Monitors

Preface

Contents of the function manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

Function block

The **_ASI_rdAsiMonDiagnostic** function block is part of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.11.1 Fundamental safety instructions

8.11.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.11.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.11.2 Description

8.11.2.1 General

This documentation describes the handling when reading out the diagnostic information of the ASIsafe safety monitors with the **_ASI_rdAsiMonDiagnostic** function block.

Note

The safety function runs independently in the AS-Interface network. The safety monitor and the safety-oriented AS-Interface slaves are self-contained, tested and certified hardware modules. The **_ASI_rdAsiMonDiagnostic** function block on the SIMOTION side does not perform any safety-relevant functions, but only reads out the status of the enabling circuits on the safety monitor.

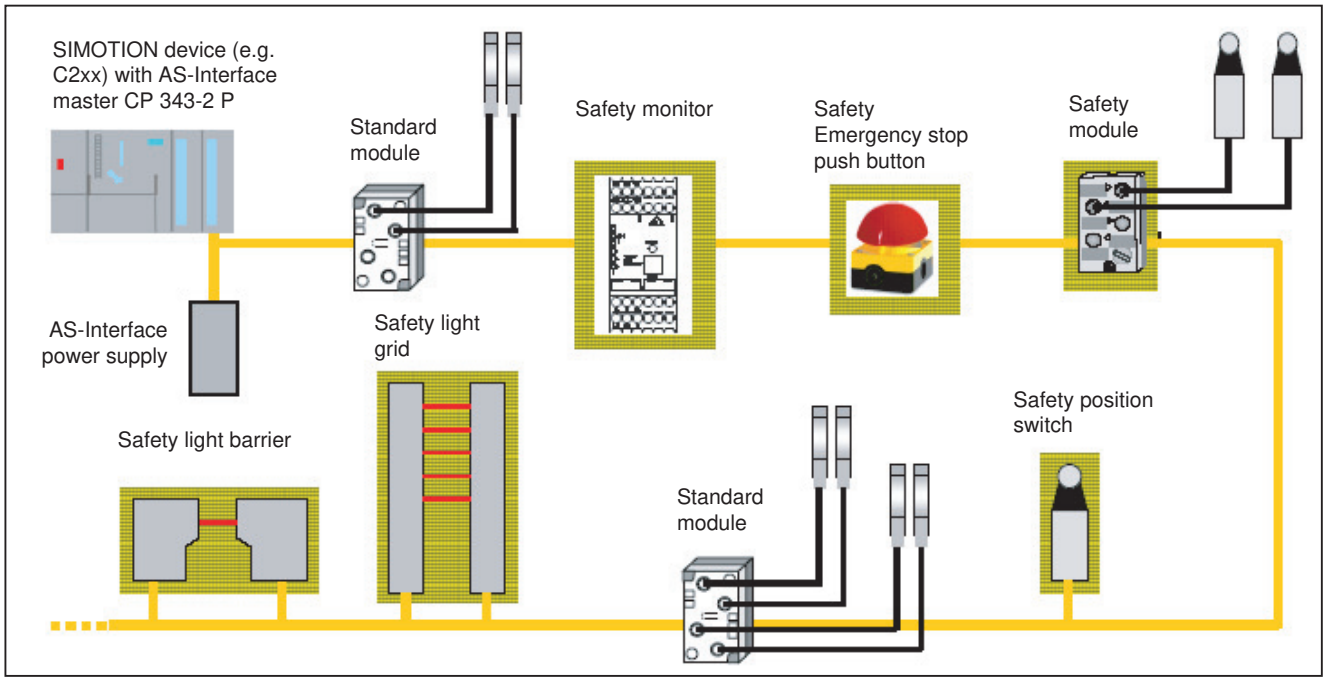


Figure 8-64 System overview

Note

The handling of the **_ASI_rdAsiMonDiagnostic** function block is described in this document. Please refer to the following documents for further information on the commissioning, configuration of the ASIsafe safety monitor and the integration in the AS-Interface:

- *"AS-Interface Safety Monitor"* operating instructions
- *"AS-Interface Safety Monitor IEC61508/EN 954-1* operating instructions
- *"asimon AS-Interface Safety Monitor Configuration Software for Microsoft® - Windows®"* manual
- *"SIMATIC NET CP 343-2/CP 343-2 P AS-Interface Master"* manual
- *"SIMATIC NET DP/AS-Interface Link 20E"* manual
- *"SIMATIC NET DP/AS-Interface Link Advanced"* Manual
- *"SIMATIC NET IE/AS-Interface Link PN IO"* Manual

The following software versions are required for the standard function described in this documentation:

- For AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced):
  - SIMOTION SCOUT V4.0 or higher
  - SIMOTION Kernel V4.0 or higher
- For AS-Interface master IE/AS-Interface Link PN IO:
  - SIMOTION SCOUT, V4.1 SP1 or higher
  - SIMOTION Kernel, V4.1 SP1 or higher

8.11.2.2 Product description

The **_ASI_rdAsiMonDiagnostic** function block is used as a communication interface between the ASIsafe safety monitor, Version 2 and higher (in the Basic/Advanced development stages), and a SIMOTION device. The FB **_ASI_rdAsiMonDiagnostic** reads the diagnostics data provided by the ASIsafe safety monitor, evaluates this and makes the evaluated data available in a data structure.

The **_ASI_rdAsiMonDiagnostic** FB can be operated with the following AS-Interface masters:

- CP 343-2 P
- DP/AS-Interface Link 20E
- DP/AS-Interface Link Advanced
- IE/AS-Interface Link PN IO

8.11.3 Programming

8.11.3.1 Integrating the function block in the user project

Creating the FB instance in the user project

The function block is part of the command library of the "SIMOTION SCOUT" engineering system. To work with the block, an instance must first be created in the user project (see following example).

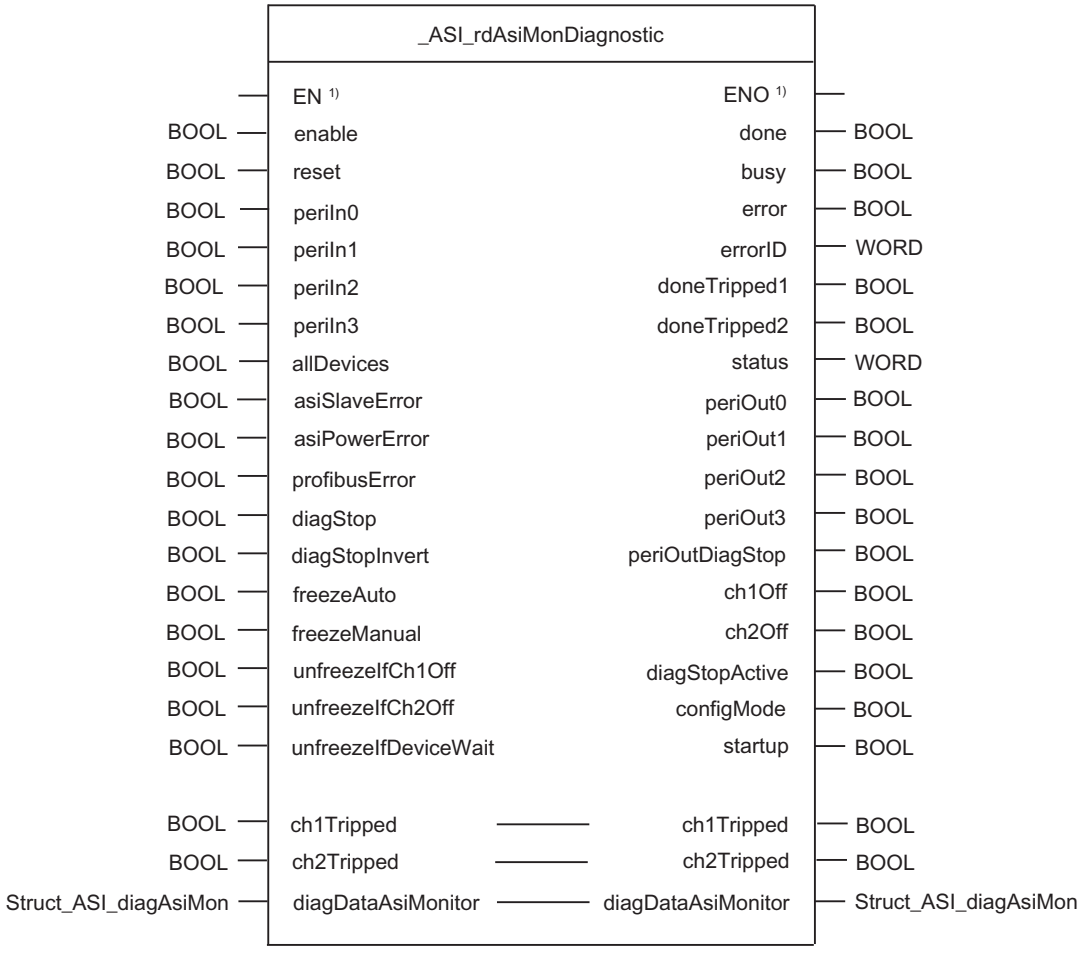
Note

If you wish to control several ASIsafe safety monitors, you must create a variable for the **Struct_ASI_diagAsiMon** data structure and an FB instance with a new name for each safety monitor used.

Example

```
VAR_GLOBAL
...
    myRdAsiMonDiagnostic   : _ASI_rdAsiMonDiagnostic;    // instance of FB
                                                         // _ASI_rdAsiMonDiagnostic
...
END_VAR
```

Call (LAD representation)



1) LAD-specific parameters

Application example

The application example is included on the "SIMOTION Utilities & Applications" DVD and is available for various SIMOTION hardware platforms.

The "SIMOTION Utilities & Applications" DVD is provided free of charge and as part of the SIMOTION SCOUT scope of supply.

8.11.3.2 Addressing the AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/ Link Advanced)

Integrating the ASIsafe safety monitor in the address area of the AS-Interface master

The ASIsafe safety monitor is addressed in the same way as an AS-Interface slave. The diagnostic information is read in via a 4-bit wide interface and control information is output on the ASIsafe safety monitor via a 4-bit wide interface.

This 4-bit input and output data is contained in the address area of the AS-Interface master (CP 343-2 P, Link 20E or Link Advanced) as the high or low part of a byte. Which byte in the address area contains the 4-bit input and output data and whether this data is classified as high or low part, depends on the specified AS-Interface address of the ASIsafe safety monitor.

The following table shows an example of how the AS-Interface slave addresses are assigned to the address area of the AS-Interface master CP 343-2 P (where n is the start address of the I/O address area assigned to the CP 343-2 P).

The assignment of the AS-Interface slave addresses to the address area of the AS-Interface master Link 20E/Link Advanced is the same as that for the CP 343-2 P. The address area of the Link 20E/Link Advanced covers 32 bytes.

Table 8-148 Binary addressing of the AS-Interface slaves in the AS-Interface master CP 343-2 P

Master								
Starting address (n)	**Bits 7 - 4**				**Bits 3 -0**			
	Slave				Slave			
	Bit 3	Bit 2	Bit 1	Bit 0	Bit 3	Bit 2	Bit 1	Bit 0
n+0	Reserved				Slave 1 or 1A			
n+1	Slave 2 or 2A				Slave 3 or 3A			
n+2	Slave 4 or 4A				Slave 5 or 5A			
n+3	Slave 6 or 6A				Slave 7 or 7A			
n+4	Slave 8 or 8A				Slave 9 or 9A			
n+5	Slave 10 or 10A				Slave 11 or 11A			
n+6	Slave 12 or 12A				Slave 13 or 13A			
n+7	Slave 14 or 14A				Slave 15 or 15A			
n+8	Slave 16 or 16A				Slave 17 or 17A			
n+9	Slave 18 or 18A				Slave 19 or 19A			
n+10	Slave 20 or 20A				Slave 21 or 21A			
n+11	Slave 22 or 22A				Slave 23 or 23A			
n+12	Slave 24 or 24A				Slave 25 or 25A			
n+13	Slave 26 or 26A				Slave 27 or 27A			
n+14	Slave 28 or 28A				Slave 29 or 29A			
n+15	Slave 30 or 30A				Slave 31 or 31A			

Addressing the AS-Interface master in SIMOTION

Communication between the SIMOTION device and the AS-Interface master takes place via direct access to the I/O and data record transfer. Addresses as of 256 are recommended for the AS-Interface master. In this address area, communication takes place via direct access to the I/O with the aid of I/O variables.

Creating I/O variables

One I/O variable each must be created in the symbol browser for the read or write access. When you create the I/O variable, you must enter the parameterized address of the AS-Interface master from the HW configuration in the "I/O address" column. The value in the "Array length" column depends on the used AS-Interface master. Enter the value 16 bytes as field length for the CP 343-2 P.

The field length (xx) to be parameterized for the Link 20E/Link Advanced should be entered in relation to the address area assigned (from **HW Config**).

With these specifications, one I/O variable of type ARRAY[0..15] of byte (CP 343-2 P) or of type ARRAY[0..xx] of byte (Link 20E/Link Advanced) is created respectively in the SIMOTION project for direct read/write access to the I/O.

Parameter transfer

The 4-bit wide input data of the ASIsafe safety monitor are transferred bit-by-bit to the FB **_ASI_rdAsiMonDiagnostic** in the **periIn0..3** input parameters.

The output data (periOut0..3) provided by the FB must be assigned to the I/O variable for write access to the AS-Interface master in accordance with the address assignment of the ASIsafe safety monitor.

Example of parameter transfer to FB _ASI_rdAsiMonDiagnostic

An ASIsafe safety monitor with AS-Interface slave address 1 and an AS-Interface master with address 256 are used in the example.

I/O variable for access to the AS-Interface slaves in the CP 343-2 P address area:

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperiincp3432	PIB 256		Array	16
2	⊞ myperioutcp3432	PQB 256	☐	Array	16

Figure 8-65 Addressing with I/O variable

Assignment of the ASIsafe safety monitor with the AS-Interface slave address 1 in the CP 343-2 P address area (corresponds to the table "Binary addressing of the AS-Interface slaves in the AS-Interface master CP 343-2 P"):

Bits 0..3 in input byte 0 with address 256 → periInCP3432[0]

Bits 0..3 in output byte 0 with address 256 → periOutCP3432[0]

Transfer of the input and output data to the FB **_ASI_rdAsiMonDiagnostic**:

```
myRdAsiMonDiagnostic (.......
            periIn0 := ((periInCP3432[0] AND 16#01)<>0),                    // Bit 0
            periIn1 := ((periInCP3432[0] AND 16#02)<>0),                    // Bit 1
            periIn2 := ((periInCP3432[0] AND 16#04)<>0),                    // Bit 2
            periIn3 := ((periInCP3432[0] AND 16#08)<>0),                    // Bit 3
            .......
            );
  // selective deletion of bits 0..3 in output byte 0
  periOut[0]:= periOut[0] AND 16#F0;
  periOut[0]:= periOut[0] OR myRdAsiMonDiagnostic.periOut0;                 // Bit 0
  periOut[0]:= periOut[0] OR SHL(myRdAsiMonDiagnostic.periOut1,1);          // Bit 1
  periOut[0]:= periOut[0] OR SHL(myRdAsiMonDiagnostic.periOut2,2);          // Bit 2
  periOut[0]:= periOut[0] OR SHL(myRdAsiMonDiagnostic.periOut3,3);          // Bit 3
```

Note

For additional information, refer to:

- *SIMOTION SCOUT* online help
- Programming Manual for the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* Programming Manual
  - *SIMOTION MCC, Motion Control Chart* Programming Manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

8.11.3.3 Addressing the AS-Interface master IE/AS-Interface Link PN IO

Integrating the ASIsafe safety monitor in the address area of the AS-Interface master

The ASIsafe safety monitor is addressed the same as an AS-Interface slave. The diagnostic information is read in via a 4-bit wide interface and control information is output on the ASIsafe safety monitor via a 4-bit wide interface.

For each AS-Interface slave, 1 byte is reserved in the IE/AS-Interface Link address area. The 4-bit wide input and output data of the ASIsafe safety monitor are transferred in the low part (bits 0 - 3) of this reserved byte.

There are no fixed specifications in terms of how the AS-Interface slave addresses are assigned to the IE/AS-Interface Link PN IO address area; this can be changed in the Step7 HW configuration.

Example of assigning AS-Interface slave addresses to the input/output addresses of the IE/AS-Interface Link PN IO:

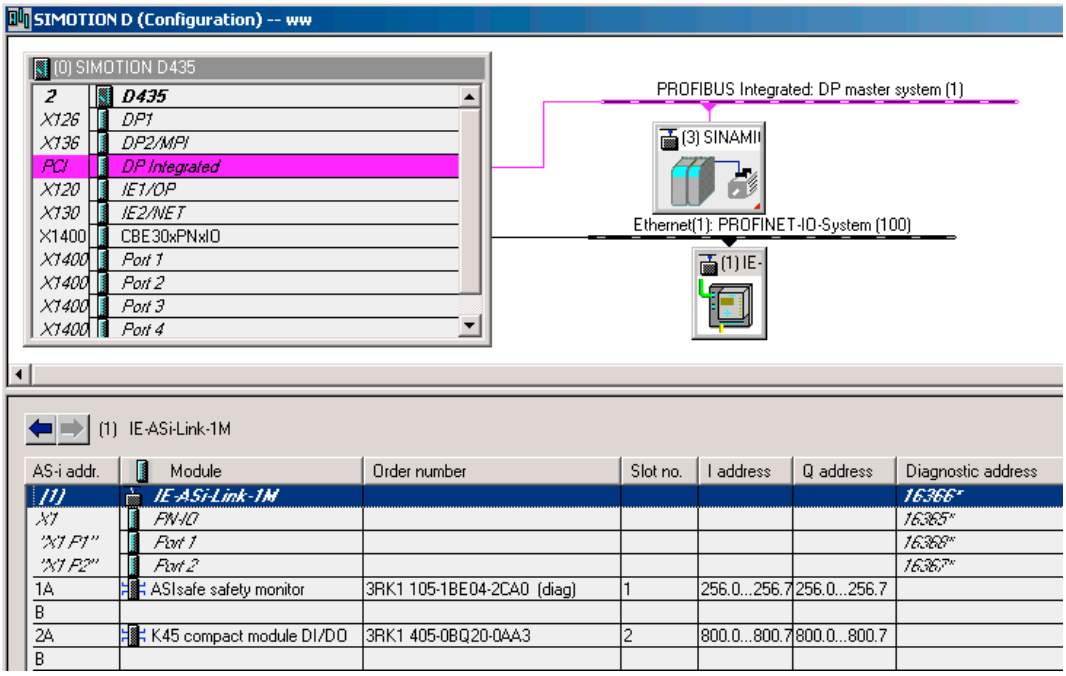


Figure 8-66 Example of address assignment for the IE/AS-Interface Link PN IO

Addressing the AS-Interface master in SIMOTION

Communication between the SIMOTION device and the AS-Interface master is performed through direct access to the I/O and data set transfer. Addresses from 256 are recommended for the AS-Interface master. In this address area, communication takes place via direct access to the I/O with the aid of I/O variables.

Creating I/O variables

One I/O variable of type BYTE must be created in the symbol browser for each instance of read or write access. When you create the I/O variable, you must enter the parameterized address of the AS-Interface master from the HW configuration in the "I/O address" column. Enter the value "1" in the "Field length" column.

Parameter transfer

The 4-bit wide input data of the ASIsafe safety monitor are transferred bit-by-bit to the FB **_ASI_rdAsiMonDiagnostic** in the **periIn0..3** input parameters.

The output data (periOut0..3) provided by the FB must be assigned to the I/O variable for write access to the AS-Interface master in accordance with the address assignment of the ASIsafe safety monitor.

Example of parameter transfer to FB _ASI_rdAsiMonDiagnostic

An ASIsafe safety monitor with AS-Interface slave address 1 and an AS-Interface master with address 256 are used in the example.

I/O variable for access to the ASIsafe safety monitor:

	Name	I/O address	Read only	Data type	Field length
1	myperiinieasilink	PIB 256		BYTE	1
2	myperioutieasilink	PQB 256		BYTE	1

Figure 8-67 Addressing with I/O variable

The 4-bit wide information is transferred in the LOW part (bits 0 - 3).

Transfer of the input and output data to the FB **_ASI_rdAsiMonDiagnostic**:

```
myRdAsiMonDiagnostic (.......
            periIn0 := ((myperiInIEASILink AND 16#01)<>0),              // Bit 0
            periIn1 := ((myperiInIEASILink AND 16#02)<>0),              // Bit 1
            periIn2 := ((myperiInIEASILink AND 16#04)<>0),              // Bit 2
            periIn3 := ((myperiInIEASILink AND 16#08)<>0),              // Bit 3
            .......
            );
 // selective deletion of bits 0..3 in output byte 0
 myPeriOutIEASILink:= myPeriOutIEASILink AND 16#F0;
 myPeriOutIEASILink:= myPeriOutIEASILink OR myRdAsiMonDiagnostic.periOut0;        // Bit 0
 myPeriOutIEASILink:= myPeriOutIEASILink OR SHL(myRdAsiMonDiagnostic.periOut1,1); // Bit 1
 myPeriOutIEASILink:= myPeriOutIEASILink OR SHL(myRdAsiMonDiagnostic.periOut2,2); // Bit 2
 myPeriOutIEASILink:= myPeriOutIEASILink OR SHL(myRdAsiMonDiagnostic.periOut3,3); // Bit 3
```

Note

For additional information, refer to:

- *SIMOTION SCOUT* online help
- Programming Manual for the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* Programming Manual
  - *SIMOTION MCC, Motion Control Chart* Programming Manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

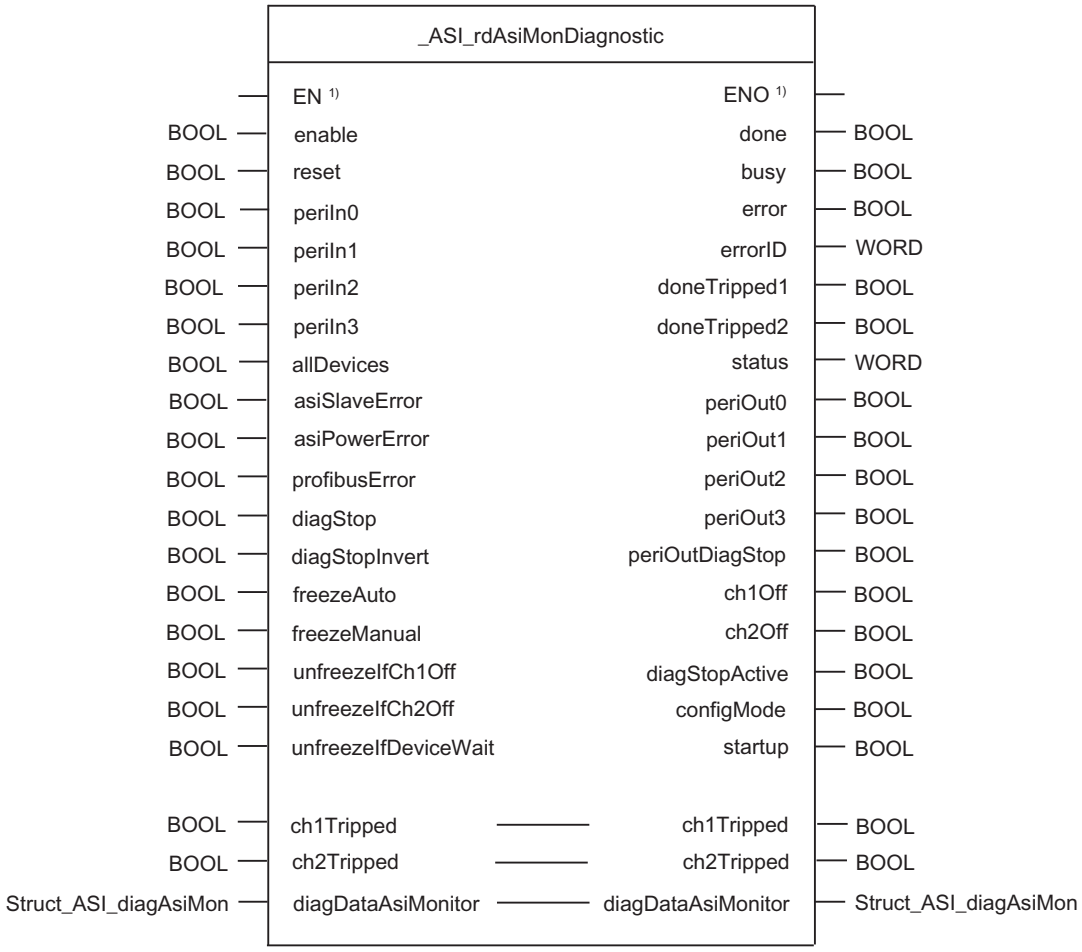
8.11.4 Parameterizing

8.11.4.1 Function block _ASI_rdAsiMonDiagnostic

Task

The FB **_ASI_rdAsiMonDiagnostic** is a function block for reading out the diagnostic information of the ASIsafe safety monitor. The diagnostic information can be used to determine the status of the enabling circuits of the safety monitor and the safety-oriented slaves. For this purpose, during a cyclic call of the FB **_ASI_rdAsiMonDiagnostic** the diagnostic information provided by the safety monitor is continuously read out, evaluated, and made available in the data structure **Struct_ASI_diagAsiMon**.

Call (LAD representation)



1) LAD-specific parameters

Parameter description

The following table contains all parameters of the **_ASI_rdAsiMonDiagnostic** function block.

Note

You can find a comparison of SIMOTION and SIMATIC names in the Appendix SIMOTION and SIMATIC names (Page 6539).

Table 8-149 Parameters of the FB _ASI_rdAsiMonDiagnostic

Name	Type	Data type	Default	Meaning
enable	IN	BOOL	FALSE	**TRUE** = continuous read out of the diagnostic information **FALSE** = the initialization of the ASIsafe safety monitor is run through once
reset	IN	BOOL	FALSE	Rising edge = reset of FB **_ASI_rdAsiMonDiagnostic**
periIn0	IN	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periIn1	IN	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periIn2	IN	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periIn3	IN	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
allDevices	IN	BOOL	FALSE	Data format of the diagnostic data (depends on the ASIsafe safety monitor version and the asimon configuration program); see the section titled "Settings in the asimon conifguration program" in this chapter.
asiSlaveError [1)]	IN	BOOL	FALSE	The parameter must be set (**TRUE**) when an AS-Interface slave has failed.
asiPowerError [1)]	IN	BOOL	FALSE	The parameter must be set (**TRUE**) when the AS-Interface power supply has failed.
profibusError [1)]	IN	BOOL	FALSE	The parameter must be set (**TRUE**) when the communication with the AS-Interface master has been interrupted.
diagStop [1)]	IN	BOOL	FALSE	**TRUE** = activation of the diagnosis stop functionality in FB **_ASI_rdAsiMonDiagnostic** (corresponding to the asimon configuration program)
diagStopInvert [1)]	IN	BOOL	FALSE	**TRUE** = value on output parameter **periOutDiagStop** is output inverted (corresponding to the asimon configuration program)
freezeAuto [1)]	IN	BOOL	FALSE	**TRUE** = control of the diagnosis stop condition via FB **_ASI_rdAsiMonDiagnostic**
freezeManual [1)]	IN	BOOL	FALSE	**TRUE** = manual activation of the diagnosis stop condition
unfreezeIfCh1Off [1)]	IN	BOOL	FALSE	**TRUE** = diagnosis stop condition is also deactivated (unfreeze) when the enabling circuit 1 has shut down (independent of the status in enabling circuit 2)
unfreezeIfCh2Off [1)]	IN	BOOL	FALSE	**TRUE** = diagnosis stop condition is also deactivated (unfreeze) when the enabling circuit 2 has shut down (independent of the status in enabling circuit 1)

Name	Type	Data type	Default	Meaning
unfreezeIfDeviceWait [1]	IN	BOOL	FALSE	**TRUE** = the diagnosis stop condition is also deactivated periodically (unfreeze) when it is detected in the running diagnostic sequence that at least one device is in the "Wait" state (status = 2)
done	OUT	BOOL	FALSE	**TRUE** = diagnostic sequence completed, present for one cycle.
busy	OUT	BOOL	FALSE	**TRUE** = block reading diagnostic information, data not consistent **FALSE** = diagnostic sequence completed, data consistent
error	OUT	BOOL	FALSE	**TRUE** = diagnostic sequence aborted with error, error specification in **errorID**
errorID	OUT	WORD	16#0000	Error specification With **error** = **TRUE**, the error information is in parameter **errorID**, see "Error and status messages".
doneTripped1 [1]	OUT	BOOL	FALSE	**TRUE** = diagnostic sequence completed and shutdown in enabling circuit 1, present for one cycle.
doneTripped2 [1]	OUT	BOOL	FALSE	**TRUE** = diagnostic sequence completed and shutdown in enabling circuit 2, present for one cycle.
status	OUT	WORD	16#0000	0x7A01 = **reset** active 0x00xx = number of cycles for a diagnostic sequence
periOut0	OUT	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periOut1	OUT	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periOut2	OUT	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periOut3	OUT	BOOL	FALSE	I/O variable with bit address of the ASIsafe safety monitor
periOutDiagStop	OUT	BOOL	FALSE	I/O variable with bit address for diagnosis stop (must match the configured address in the asimon configuration program)
ch1Off	OUT	BOOL	FALSE	**TRUE** = enabling circuit 1 will be configured and shut down
ch2Off	OUT	BOOL	FALSE	**TRUE** = enabling circuit 2 will be configured and shut down
diagStopActive	OUT	BOOL	FALSE	**TRUE** = diagnosis stop condition active A device will be "frozen" by the safety monitor
configMode	OUT	BOOL	FALSE	**TRUE** = safety monitor is in configuration mode
startup	OUT	BOOL	FALSE	**TRUE** = safety monitor is starting up
ch1Tripped	IN/OUT	BOOL	FALSE	**TRUE** = status change from ON to OFF in enabling circuit 1
ch2Tripped	IN/OUT	BOOL	FALSE	**TRUE** = status change from ON to OFF in enabling circuit 2
diagDataAsiMonitor	IN/OUT	**Struct_ASI_diagAsiMon**		Data structure for storing the diagnostic information of the ASIsafe safety monitor

[1] The function of this parameter is described in detail in the following.

asiSlaveError, asiPowerError, profibusError

These input parameters inform the FB **_ASI_rdAsiMonDiagnostic** of error statuses of the ASI slaves and the ASI power supply, as well as errors on the PROFIBUS DP with level **TRUE**.

- The following table contains the input parameters in which the error messages with the listed meanings are transferred.

Parameter	Meaning of the error message
asiSlaveError	**TRUE** = failure of the AS-Interface slave safety monitor
asiPowerError	**TRUE** = failure of the AS-Interface power supply (or other general AS-Interface failure)
profibusError	**TRUE** = failure of the communication with the AS-Interface master

Note

PROFIBUS errors are signaled by alarms. The cause of the error can be determined by evaluating the DP slave diagnostics. The AS-Interface master signals faults and errors at the AS-Interface (e.g. failure of the ASI slave, failure of the ASI power supply, etc.) to the higher-level control using alarms.

You can obtain detailed information on how the AS-Interface master responds, as well as the relevant error information, from the following documents:

- For CP 342-2 P from DS1 (see *"SIMATIC NET CP 343-2/CP 343-2 P AS-Interface Master"* Manual)
- For Link 20E from the PROFIBUS slave diagnostics (see *"SIMATIC NET DP/AS-Interface Link 20E"* Manual)
- For Link Advanced from the PROFIBUS slave diagnostics (see *"SIMATIC NET DP/AS-Interface Link Advanced"* Manual)
- For IE/AS Link PNI IO from the alarms and diagnostic data records (see *"SIMATIC NET IE/AS-Interface PN IO"* Manual)

These documents are shipped with SIMOTION SCOUT in electronic form.

- Requirements for transferring the diagnostic data are that the AS-Interface slave address of the safety monitor can be addressed by the AS-Interface master, and that the SIMOTION device is correctly connected to the AS-Interface master.
  The diagnostic sequence must be aborted in the event of a communication failure. The diagnostic sequence must be re-initialized when the communication is restored.
- The diagnostic sequence is aborted if any of the **asiSlaveError**, **asiPowerError**, **profibusError**, or **reset** parameters is set to **TRUE**.
  In this status, no input data are read and no output data are written.
- The function block runs through an initialization phase once as soon as all 4 parameters are **FALSE**:

asiSlaveError := **FALSE**
asiPowerError := **FALSE**
profibusError := **FALSE**
reset := **FALSE**

Diagnostic data is then continuously read in.

- The reset and initialization phase are also run through continuously for as long as parameter **reset** := **TRUE**.
- The initialization phase is also run through once when the block-internal watchdog determines that no data has been received from the safety monitor for some time (2,000 ms) during a running diagnostic sequence.

diagStop, diagStopInvert, freezeAuto, freezeManual, unfreezeIfCh1Off, unfreezeIfCh2Off, unfreezeIfDeviceWait

These input parameters control the "diagnosis stop" function of the safety monitor as of Version V2.x. A safety monitor in Version V1.x does not support this function and ignores these input parameters.

- If a device has registered a shutdown, the "diagnosis stop" function "freezes" the device in its current status. This means that the device remains in the "Off" status even when it is switched on again at a later point, and is frozen in the "Wait" status (YELLOW color display in the diagnostics mode of the asimon configuration program, corresponding to status code 2 in the diagnostics data, as with waiting for local acknowledgement).
- "Freezing" is only possible when the stop condition of the diagnosis stop function is activated.
- A "frozen" device can be enabled again (unfreeze) by deactivating the stop condition of the diagnosis stop function.
- With the "diagnosis stop" function, the diagnostics can register a transient device shutdown, which would otherwise result in a shutdown of the enabling circuit but would not be recognizable in the diagnostics.

Note

The "diagnosis stop" function can delay the reactivation of an enabling circuit of the safety monitor, as the devices must first be enabled (unfreeze). In connection with OR gates, the "diagnosis stop" function can even result in the unwanted shutdown of an enabling circuit, as each individual status of a gate input is frozen and the resulting sum of the possibly frozen inputs can result in the shutdown of the gate output.

- The "diagnosis stop" function can be activated in the asimon configuration program under **Edit > Monitor/bus information > Diagnosis/service > Diagnosis stop** (see "asimon AS-Interface Safety Monitor Configuration Software for Microsoft®-Windows®" manual). It is recommended that this function only be activated for troubleshooting. The function should not be activated especially when using OR gates (as described above).
- When the diagnosis stop function is activated, an AS-Interface slave address including the bit address must be specified in the asimon configuration program. An output must be specified as a bit address for use with the diagnostics function block (output parameter **periOutDiagStop**), so that this can be controlled by the SIMOTION device. The option "Inverted" should remain deactivated. Reason: During the operating mode transition from RUN to STOP, all outputs are in status 0, which means that the diagnosis stop is also deactivated. The safety monitor cannot be blocked by "frozen" devices. If required, the system can then be operated in manual mode.

Note

Connection of output parameter periOutDiagStop with setting in the asimon configuration program

The diagnosis stop condition is controlled with the output parameter **periOutDiagStop**. For the correct sequence, the output parameter **periOutDiagStop** must be linked to a bit output of an arbitrary AS-Interface slave. The bit address of this AS-Interface slave must match the set stop condition in the asimon configuration program (see following example).

Example

The first output is used for controlling the diagnosis stop condition for an AS-Interface standard slave with 2 inputs and outputs and slave address 3.

1. Settings in the asimon configuration program in the **Monitor/Bus Information** window:

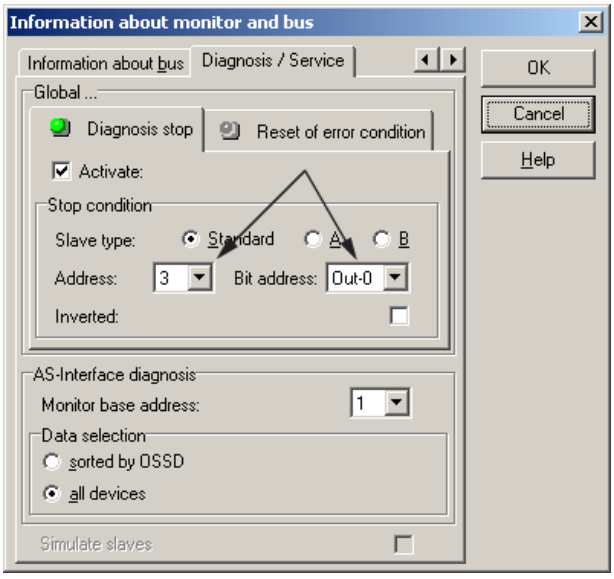


Figure 8-68 Stop condition setting for diagnosis stop (asimon V2.0.3)

2. Linking of the output parameter **periOutDiagStop** to the first output of the AS-Interface standard slave with slave address 3 (see Chapter Addressing the AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced) (Page 6512) or Addressing the AS-Interface master IE/AS-Interface Link PN IO (Page 6515)):

```
myPeriout_cp343_2[1] := (myPeriout_cp343_2[1] AND 16#FE) OR
myRdAsiMonDiagnostic.periOutDiagStop;

 // Explanation
 // myPeriOutCP3432                          : I/O variable with length of 16 bytes for
 //                                          the address area of the AS-Interface master
 // myPeriOutCP3432[1]                      : Byte 1 bits 0..3 of AS-Interface address space
 //                                          Standard slave with address 3
 // (myPeriOutCP3432[1] AND 16#FE)           : Selective resetting of output 1
 // OR myRdAsiMonDiagnostic.periOutDiagStop : OR operation output parameter
 //                                          periOutDiagStop with first output of the
 //                                          AS-Interface standard slave with address 3
```

- in order that the function block can control the diagnosis stop condition, the input parameter **diagStop** := **TRUE**. With **diagStop** := **FALSE**, the diagnosis stop function is not processed; the output bit defined with output parameter **periOutDiagStop** is not changed.
- The input parameter **diagStopInvert** must match the setting in the asimon configuration program:
  - If the option "Inverted" is deactivated (recommended), then **diagStopInvert** := **FALSE** must be set.
  - If the option "Inverted" is selected, then **diagStopInvert** := **TRUE** must be set.
- If both input parameters are **freezeAuto** := **FALSE** and **freezeManual** := **FALSE**, then the diagnosis stop condition is permanently deactivated, i.e. the safety monitor does not "freeze" any devices.

- If the input parameter **freezeAuto** = **FALSE**, then the diagnosis stop condition can be manually activated with **freezeManual** := **TRUE**, so that the safety monitor "freezes" devices. The re-enabling (unfreeze) must also be performed manually! In this case, the input parameters **unfreezeIfCh1Off**, **unfreezeIfCh2Off** and **unfreezeIfDeviceWait** are not taken into account.
- If input parameter **freezeAuto** = **TRUE**, then the function block manages control of the diagnosis stop condition automatically. In this case, the input parameter **freezeManual** is not taken into account. The following rules then apply:
  - At first, the diagnosis stop condition is activated, i.e. the safety monitor can "freeze" devices.
  - After the diagnostics have detected that both enabling circuits have shut down, the diagnosis stop condition is deactivated (unfreeze), i.e. the safety monitor can be reactivated.
  - If the input parameter **unfreezeIfCh1Off** = **TRUE**, the diagnosis stop condition is also deactivated (unfreeze) when enabling circuit 1 has shut down (regardless of the status in enabling circuit 2).
  - If the input parameter **unfreezeIfCh2Off** = **TRUE**, the diagnosis stop condition is also deactivated (unfreeze) when enabling circuit 2 has shut down (regardless of the status in enabling circuit 1).
  - If the input parameter **unfreezeIfDeviceWait** := **TRUE**, the diagnosis stop condition is also deactivated periodically (unfreeze) in the event that the running diagnostic sequence has detected that at least one device is in the "Wait" status (**status** = 2) In this case, periodic deactivation means that the diagnosis stop condition is deactivated for 100 ms and then reactivated for 100 ms. Although it is no longer possible to register transient device shutdowns, "frozen" devices that do not directly lead to the shutdown of the enabling circuit (e.g. gate inputs) may be enabled again during running operation (unfreeze).

doneTripped1, doneTripped2

The parameter assumes the state **TRUE** for one cycle, when a diagnostic sequence has been successfully completed or aborted with error, and a shutdown (transition **ON** to **OFF**) has been registered in enabling circuit 1 or enabling circuit 2 in this diagnostic sequence. This enables, for example, diagnostic data archiving to be triggered.

Functional description

After the safety monitor is ramped up, the diagnostic information of the safety monitor is continuously read out with level **TRUE** on the input parameter **enable** of FB **_rdAsiMonDiagnostic**. During the read out of the diagnostic sequence, output parameter **busy** = **TRUE**. The successful completion of the read out of the diagnostic information is indicated by output parameters **done** = **TRUE** and **busy = FALSE**. The data evaluated by FB **_ASI_rdAsiMonDiagnostic** have been copied to the data structure **Struct_ASI_diagAsiMon** and are up-to-date. The diagnostics data of the safety monitor are then read out again from FB **_ASI_rdAsiMonDiagnostic** (**busy** = **TRUE**).

A faulty completion of the read out is indicated with **TRUE** on the output parameter **error**. The error is specified in output parameter **errorID**.

In the input parameters **asiSlaveError**, **asiPowerError**, and **profibusError**, the FB **_ASI_rdAsiMonDiagnostic** is informed of error statuses of the AS-Interface slaves (e.g. safety monitor), AS-Interface power supply, and communication to the AS-Interface master with level **TRUE**. The read out of the diagnostic information is then aborted.

No diagnostic data is read in if input parameter **enable** = **FALSE** . A running read out of the diagnostic information and an internal reset/initialization is run through once.

With a rising edge on input parameter **reset**, the FB **_ASI_rdAsiMonDiagnostic** is reset to the basic state. This ensures that the function block can start correctly with the readout of the diagnostic data, irrespective of the previous actions (e.g. separate shutdown of the safety monitor).

With level **FALSE** on input parameter **enable** or rising edge on input parameter **reset**, the levels of the output parameters **busy**, **done** and **error** are set to **FALSE**. The value in output parameter **status** is 0x7A02. The levels and states of all further output parameters as well as the data of the elements of the data structure **Struct_Asi_diagAsiMon** are "frozen".

A detected shutdown is output immediately on the in/out parameters **ch1Tripped** and **ch2Tripped**, without waiting for the end of the diagnostic sequence (**done** = **TRUE**). The in/out parameters must be reset in the user program after their evaluation.

Graphic overview of the functionality

Safety operation

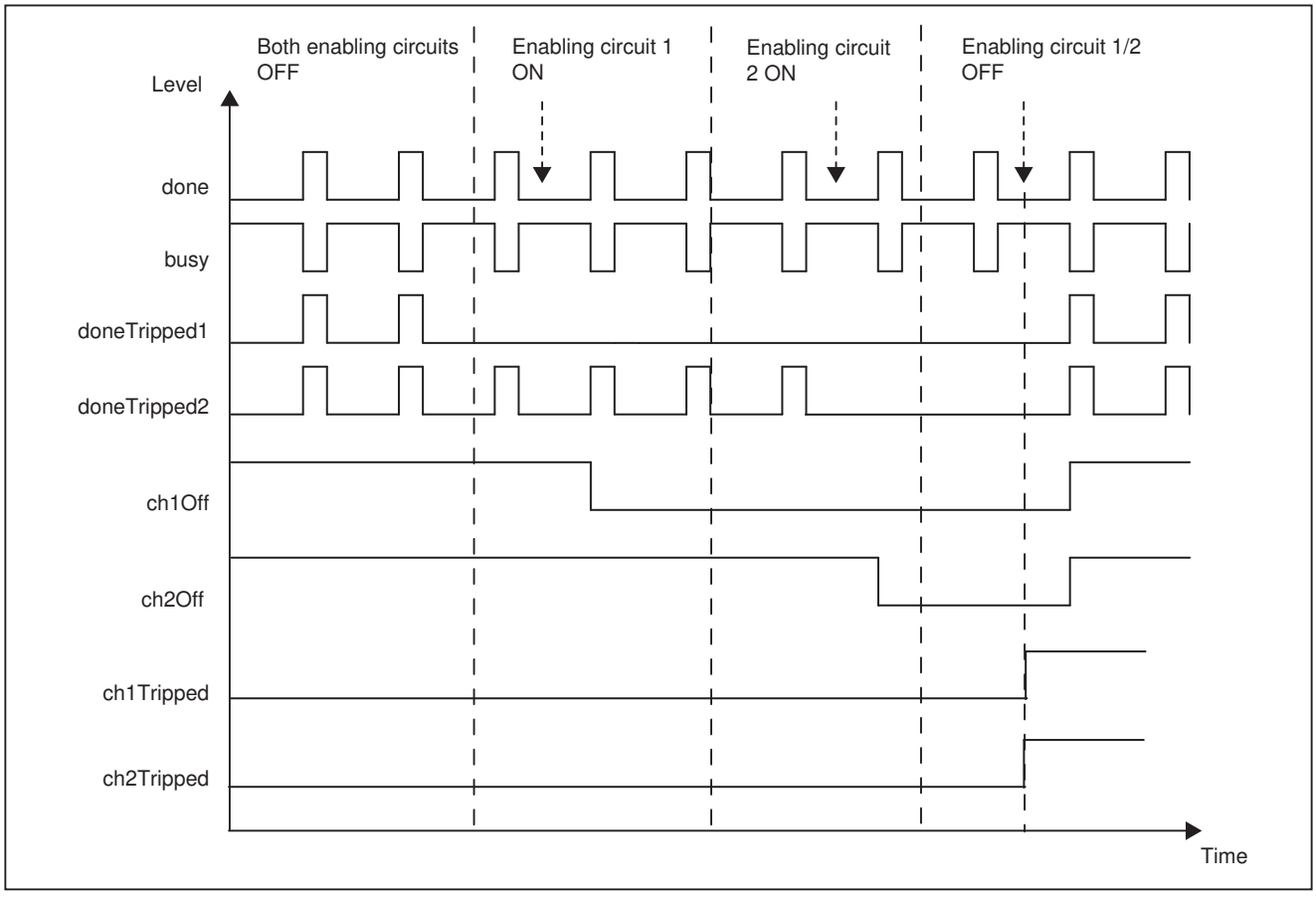


Figure 8-69 Signal flowchart of safety operation

reset := TRUE, enable := FALSE, asiSlaveError := TRUE

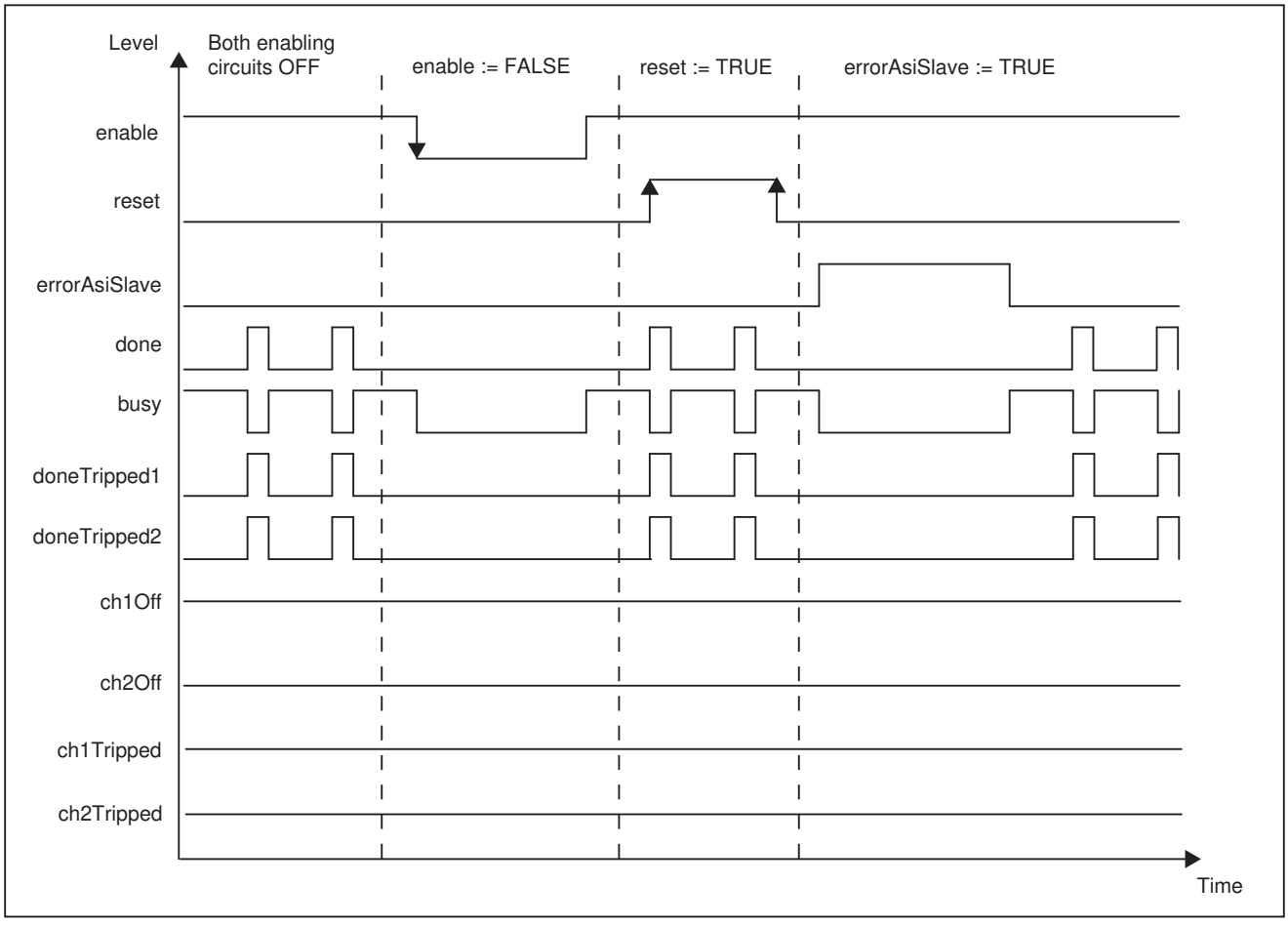


Figure 8-70 Signal flowchart of both enabling circuits OFF

Settings in the asimon configuration program

Data selection

The asimon program is used for the configuration and commissioning of the ASIsafe safety monitor via a PC.

The input parameter **allDevices** on FB **_ASI_rdAsiMonDiagnostic** must be set with level **TRUE** or **FALSE** in accordance with the configuration of the safety monitor.

The selection can be made in the asimon configuration program of the safety monitor. To do this, select the menu item **Monitor/bus information** in the **Edit** menu and open the **Diagnosis/ Service** window:

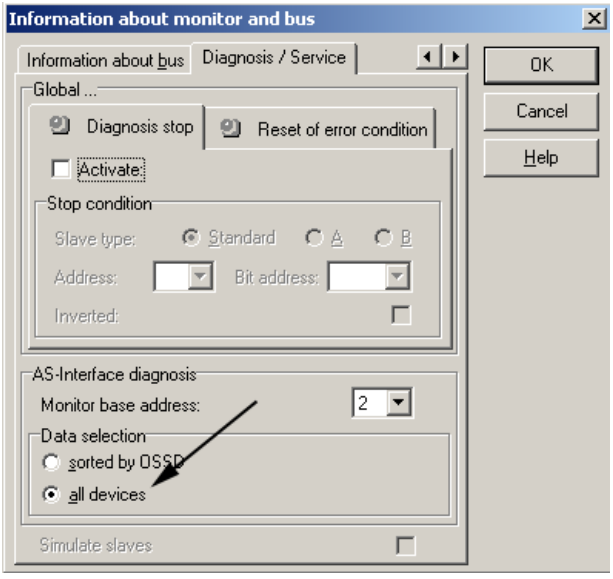


Figure 8-71 Activating the "all devices" data selection (asimon V2.0.3)

With the data selection "all devices" in the asimon configuration program, the input parameter **allDevices** must be set to **TRUE**. If the "all devices" setting is made, the diagnostic data of the devices are also provided during safety monitor preprocessing. With the selection "sorted by OSSD", the input parameter **allDevices** must be set to **FALSE**.

Diagnosis stop

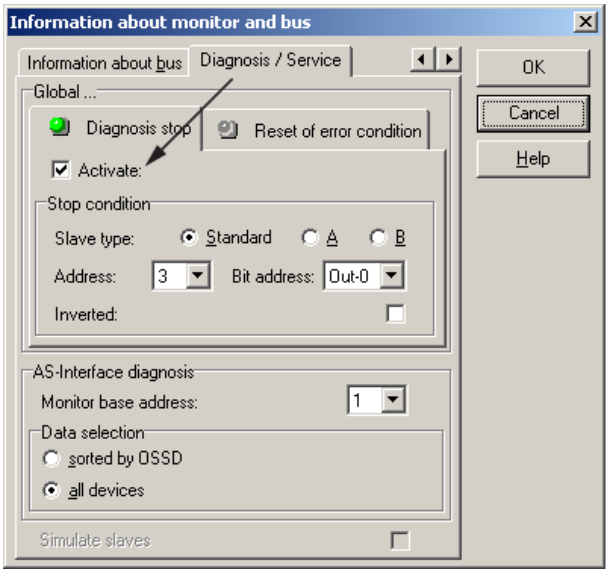


Figure 8-72 Activation of diagnosis stop (asimon V2.0.3)

The following input/output parameters are only relevant for the activation of the "diagnosis stop" functionality in the asimon configuration program of the safety monitor:

Input parameters:	**diagStop, diagStopInvert, freezeAuto, freezeManual, unfreezeIfCh1Off, unfreezeIfCh2Off, unfreezeIfDeviceWait**
Output parameters:	**periOutDiagStop, diagStopActive**

Note

The diagnosis stop is explained in more detail in the description of the input parameters (see "Parameter description" in this section).

Task integration (call)

The **_ASI_rdAsiMonDiagnostic** function block is intended for the call in a cyclic task and must be run through cyclically. The execution of a job can take several cycles. The user decides in which cyclic task the FB **_ASI_rdAsiMonDiagnostic** is called. No restrictions are applied with respect to the functionality of the FB **_ASI_rdAsiMonDiagnostic**. The call can be performed in all cyclic tasks. A fixed time frame (e.g. **IPOSynchronousTask**) is not required for processing.

Error and status messages

Errors that occur are indicated at the **error** output parameter. This includes errors in the FB, communication faults and errors of the safety monitor. In the event of an error, the corresponding error number is displayed at the **errorID** output parameter. If no errors have occurred, **errorID** has a value of "16#0000". The parameter **errorID** is present until the error is corrected.

Table 8-150 Error codes in the output parameter errorID

Error no. errorID	Meaning
0x8010	Watchdog has triggered, no communication to the safety monitor
0x8020	Internal error / no request active
0x8701	Diagnostic sequence aborted, parameter **asiSlaveError = TRUE**
0x8801	Diagnostic sequence aborted, parameter **asiPowerError = TRUE**
0x8901	Diagnostic sequence aborted, parameter **profibusError = TRUE**

The pending errors are acknowledged with a rising edge in input parameter **reset**.

Table 8-151 Status messages

Status messages status	Meaning
0x7A01	The FB is in reset or in the initialization phase.
0x7A02	The FB is inactive, no update of the diagnostics data.
0x00xx	Number of cycles for a diagnostic sequence

After the error-free completion of the read in of the diagnostic sequences, the number of required cycles of the FB **_ASI_rdAsiMonDiagnostic** is displayed in the output parameter **status**.

8.11.4.2 Data structure

Struct_ASI_diagAsiMon for diagnostics data

The data structure **Struct_ASI_diagAsiMon** is a data structure for storing the diagnostic information.

Table 8-152 Struct_ASI_diagAsiMon

Name	Data type	Default	Meaning
stateFb 1)	BYTE	16#00	Status of **_ASI_rdAsiMonDiagnostic** FB
stateAsiMonitor 1)	BYTE	16#00	Status of safety monitor
stateChannel1 1)	BYTE	16#00	Status of enabling circuit 1
stateChannel2 1)	BYTE	16#00	Status of enabling circuit 2
sumChannel1	INT	0	Number of triggered devices in enabling circuit 1
sumChannel2	INT	0	Number of triggered devices in enabling circuit 2
device[32...79]	**Struct_ASI_deviceInfo**		Information on the device
Struct_ASI_deviceInfo			
chInfo 1)	BYTE	16#00	Specification of which enabling circuit contains the device
state 1)	BYTE	16#00	Status of the device
infoMaxDeviceChannel1	USINT	0	Number of the highest device signaled in channel 1 of the ASIsafe safety monitor
stateMaxDeviceChannel1 1)	BYTE	16#00	Status of this device
infoMaxDeviceChannel2	USINT	0	Number of the highest device signaled in channel 2 of the ASIsafe safety monitor
stateMaxDeviceChannel2 1)	BYTE	16#00	Status of this device

Name	Data type	Default	Meaning
infoBits	BYTE	16#00	Additional information • Bit0: TRUE - safety monitor in configuration mode • Bit1: TRUE - safety monitor is ramping up • Bit2: TRUE - safety monitor error
actTime	DT	DT#0001-01-01-0:0:0.0	Current time in FB **_ASI_rdAsiMonDiagnostic**
timeCh1Off	DT	DT#0001-01-01-0:0:0.0	Time of the last shutdown of enabling circuit 1
timeCh1On	DT	DT#0001-01-01-0:0:0.0	Time of the last switch on of enabling circuit 1
timeCh2Off	DT	DT#0001-01-01-0:0:0.0	Time of the last shutdown of enabling circuit 2
timeCh2On	DT	DT#0001-01-01-0:0:0.0	Time of the last switch on of enabling circuit 2
timeConfig	DT	DT#0001-01-01-0:0:0.0	Time of the last switchover to the configuration mode
timeStartupAsiMonitor	DT	DT#0001-01-01-0:0:0.0	Time of the last ramp-up of the safety monitor
timeError	DT	DT#0001-01-01-0:0:0.0	Time of the last communication failure
timeStartupFB	DT	DT#0001-01-01-0:0:0.0	Time of the last ramp-up of the FB **_ASI_rdAsiMonDiagnostic**

[1] The meaning of the data bytes is described in detail below.

stateFb

The 8 bits in this byte have the following meaning:

Bit	Name	Meaning
0	allDevices	Copy of the parameter **allDevices**
1	doublecheck	The diagnostic sequence is run through once again, the data may not be consistent.
2	ch1Tripped	The diagnostic sequence has been completely terminated. A shutdown (transition from ON to OFF) was registered in enabling circuit 1 in this diagnostic sequence.
3	ch2Tripped	The diagnostic sequence has been completely terminated. A shutdown (transition from ON to OFF) was registered in enabling circuit 2 in this diagnostic sequence.
4	done	The diagnostic sequence has been completed successfully.
5	error	The diagnostic sequence has been aborted.
6	ch1Off	Enabling circuit 1 is switched off.
7	ch2Off	Enabling circuit 2 is switched off.

stateAsiMonitor

The data byte **stateAsiMonitor** describes the overall status of the ASIsafe safety monitor. It can be a value between 8 and 15. The meaning of these values is described in the following table.

Decimal value	Binary value (b3 b2 b1 b0)	Meaning
8	1 0 0 0	Both circuits switched on
9	1 0 0 1	Circuit 1 off / circuit 2 on
10	1 0 1 0	Circuit 1 on / circuit 2 off
11	1 0 1 1	Both circuits switched off
12	1 1 0 0	Configuration mode (power on / reset on the safety monitor)
13	1 1 0 1	Configuration mode (stop state, editing possible with asimon configuration program)
14	1 1 1 0	Configuration mode (reserved)
15	1 1 1 1	Configuration mode (fatal device error, reset on the safety monitor or replacement required)

Both circuits are switched off in configuration mode.

stateChannel1, stateChannel2

The data byte of **stateChannel1** describes the status of enabling circuit 1. The data byte of **stateChannel2** describes the status of enabling circuit 2. The values can be between 0 and 7. The meaning of the values is described in the following table.

Decimal value	Binary value (b3 b2 b1 b0)	Meaning	LED status on the monitor
0	0 0 0 0	Circuit is switched on	Green
1	0 0 0 1	Circuit is ready to be switched on, waiting for start condition	Yellow + red
2	0 0 1 0	Circuit is switched off	Red
3	0 0 1 1	Circuit is switched off, Service key required	Flashing red
4	0 1 0 0	Reserved	
5	0 1 0 1	Reserved	
6	0 1 1 0	Reserved	
7	0 1 1 1	Reserved	

chInfo (Struct_ASI_deviceInfo)

These 48 data bytes specify in which enabling circuit each device is configured. The meaning of the values is described in the following table.

Decimal value	Binary value (b3 b2 b1 b0)	Meaning
0	0 0 0 0	Device is contained in the preprocessing or not configured.
1	0 0 0 1	Device is contained in enabling circuit 1.

Decimal value	Binary value (b3 b2 b1 b0)	Meaning
2	0 0 1 0	Device is contained in enabling circuit 2.
3	0 0 1 1	Device is contained in both enabling circuits.

state (Struct_ASI_deviceInfo), stateMaxDeviceChannel1, stateMaxDeviceChannel2

These 48 data bytes describe the status of the individual devices in the enabling circuits. One **state** data byte exists for each device (m = 32..79).

It can be a value between 0 and 7. The meaning of the values is described in the following table.

Note

The value **status** = 2 is specified even when a device has been "frozen" because of the diagnosis stop function.

Decimal value	Binary value (b3 b2 b1 b0)	Meaning	Color display in the asi-mon configuration program
0	0 0 0 0	Device is switched on.	Green
1	0 0 0 1	Device is ON, shutdown timer started.	Flashing green
2	0 0 1 0	Device waiting for local acknowledgement or start condition.	Yellow
3	0 0 1 1	Device (two-channel dependent) has been actuated with single channel; test (OFF -> ON) required, also for "startup test".	Flashing yellow
4	0 1 0 0	Device has shut down (regular shutdown).	Red
5	0 1 0 1	Device (positive-action) has switched on with one channel, or error during contactor check. Service key required.	Flashing red
6	0 1 1 0	Device communication error between AS-Interface module and safety monitor.	Gray
7	0 1 1 1	ASIsafe safety monitor is in configuration mode	-

8.11.4.3 Call example for FB _ASI_rdAsiMonDiagnostic

Calling the function block

In order to be able to work with the function block in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create the function block instance (see program segment).
2. Create variable of type **struct_ASI_diagAsiMon** and use in/out parameter **diagDataAsiMon** to transfer this to FB **_ASI_rdAsiMonDiagnostic**.

3. Select input bits 1..4 of the ASIsafe safety monitor from the input address area of the AS-Interface master.
4. Call the function block instance (see program segment).
5. Transfer the input parameters (see program segment).
6. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter> (see program segment).
7. The **periOut 0..3** data prepared by the FB is copied to the output address area of the AS-Interface master (see program segment).

Note

The program segment is an extract from the supplied application example.

The application example is contained on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge and as part of the SIMOTION SCOUT scope of delivery.

If you wish to control several ASIsafe safety monitors, you must create a variable for the **Struct_ASI_diagAsiMon** data structure and an FB instance with a new name for each safety monitor used.

Program segment from the sample program:

```
UNIT E_AsiMon;

INTERFACE

VAR_GLOBAL
// ****************************************************
// Variables for FB_ASI_rdAsiMonDiagnostic
// ****************************************************

 myDataDiagAsiMon     : struct_ASI_diagAsiMon;    // variable of data structure        (2)
                                                  // struct_ASI_diagAsiMon
 myRdAsiMonDiagnostic : _ASI_rdAsiMonDiagnostic;  // instance of FB                    (1)
                                                  // _ASI_rdAsiMonDiagnostic

END_VAR
```

```
PROGRAM ExampleAsiMon;

END_INTERFACE

IMPLEMENTATION

PROGRAM ExampleAsiMon

  // copy input data byte 0 of Asi master CP343-2P
  myPeriInDataByte := myPeriInCP3432[0];

  // ASI address of monitor : 1 -> input byte 0 (bits 0..3)
  // select and copy bit 0..3 of input byte 0
  myInputBit1AsiMon:= ((myPeriInDataByte AND 16#01)<>0); // select/copy bit 0
  myInputBit2AsiMon:= ((myperiInDataByte AND 16#02)<>0); // select/copy bit 1          (3)
  myInputBit3AsiMon:= ((myPeriInDataByte AND 16#04)<>0); // select/copy bit 2
  myInputBit4AsiMon:= ((myPeriInDataByte AND 16#08)<>0); // select/copy bit 3


  myRdAsiMonDiagnostic
  (     enable               := myEnable,                                        (4)
        reset                := myReset,
        periIn0              := myInputBit1AsiMon,
        periIn1              := myInputBit2AsiMon,
        periIn2              := myInputBit3AsiMon,
        periIn3              := myInputBit4AsiMon,
        allDevices           := myAllDevices,
        asiSlaveError        := myAsiSlaveError,
        asiPowerError        := myAsiPowerError,
        profibusError        := myProfibusError,                                 (5)
        diagStop             := myDiagStop,
        diagStopInvert       := myDiagStopInvert,
        freezeAuto           := myFreezeAuto,
        freezeManual         := myFreezeManual,
        unfreezeIfCh1Off     := myUnfreezeIfCh1Off,
        un freezeIfCh2Off    := myUnfreezeIfCh2Off,
        unfreezeIfDeviceWait := myUnfreezeIfDeviceWait,
        diagDataAsiMonitor   := mydiagDataAsiMonitor,
        ch1Tripped           := myCh1Tripped,
        ch2Tripped           := myCh2Tripped
  );

  // copy output bits 1..4 to temporary data byte (bit0..3)                        (6)

                                                                                  (7)
  myPeriOutDataByte := BYTE FROM 8BOOL( bit0  := myRdAsiMonDiagnostic.periOut0,
                                        bit1  := myRdAsiMonDiagnostic.periOut1,
                                        bit2  := myRdAsiMonDiagnostic.periOut2,
                                        bit3  := myRdAsiMonDiagnostic.periOut3,
                                        bit4  := FALSE,
                                        bit5  := FALSE,
                                        bit6  := FALSE,
                                        bit7  := FALSE );
```

```
  // delete selective bit 0..3 in peripheral output byte 0
  // "(myPeriOutCP3432[0] AND 16#F0)" and copy output data for Asi monitor
  myPeriOutCP3432[0] := ((myPeriOutCP3432[0] AND 16#F0) OR myPeriOutDataByte);

END_PROGRAM

END_IMPLEMENTATION
```

8.11.5 Application examples

8.11.5.1 General

Task

The sample program for FB **_ASI_rdAsiMonDiagnostic** shows the call and the parameter supply of FB **_ASI_rdAsiMonDiagnostic**.

The application example contains the following preconfigured addresses:

- 256 for the AS-Interface master
- AS-Interface slave address 1 for the safety monitor
- Addresses 2 and 3 for the two safety-oriented slaves (K45F).

You can choose between two sample projects:

- Sample project for **FB _ASI_rdAsiMonDiagnostic** with the CP 343-2 P
- Sample project for **FB _ASI_rdAsiMonDiagnostic** with the Link 20E/Link Advanced

Note

For the IE/AS-Interface Link PN IO, you must make the following replacements in the sample project:

1. Replace "myPeriInDataByte := myPeriInCP3432[0];" with "myPeriInDateByte := myPeriInIEASILink;"
2. Replace "myPeriOutCP3432[0] := ((myPeriOutCP3432[0] AND 16#F0) OR myPeriOutDataByte);" with "myPeriOutIEASILink := ((myPeriOutIEASILink AND 16#F0) OR myPeriOutDataByte);"

The following addresses are preset for the sample projects:

Sample project with AS-Interface master CP 343-2 P:	256
Sample project with AS-Interface Master Link 20E/Link Advanced:	0 (process image)

The following slave modules are used on the AS-Interface slave addresses 1..3 in the sample projects:

AS-Interface slave address	AS-Interface slave
1	Safety monitor
2	K45F (safety-oriented slave)
3	K45F (safety-oriented slave)

Requirement

The addressing of the AS-Interface slaves and the configuration of the AS-Interface must have been completed. The safety monitor must have a valid configuration.

A sample configuration is contained in the scope of delivery for the configuration of the safety monitor. This must be loaded with the asimon configuration program to the safety monitor.

Note

You can find out how to load a configuration to the safety monitor in the *"asimon AS-Interface Safety Monitor Configuration Software for Microsoft® Windows®"* software manual.

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you have to adapt the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. This CD-ROM is provided free of charge and is part of the SIMOTION SCOUT scope of delivery.

1. Dearchive and open the project containing the application example.
2. Check the hardware configuration.
3. If required, adapt the sample project.
4. Save and compile the example project. You can then download the sample to the SIMOTION device and switch to **RUN** mode.

Adapting the application example

The configuration in the example must be adapted to your existing hardware. The following options are available:

- You can adapt the configuration of the example to the available hardware.
- You can adapt the configuration of the hardware to the example.

Adapting the configuration of the example to the available hardware:

The assigned AS-Interface slave address of the safety monitor or the address of the AS-Interface master is decisive for the correct sequence of the read out of the diagnostic information from the safety monitor. The AS-Interface slave address of the safety monitor determines the assignment in the address area of the AS-Interface master. You can adapt the address of the AS-Interface master in the **HW Config**.

Information on addressing for a safety monitor with AS-Interface slave address 1 and address 256 of the AS-Interface master can be found in Chapter Programming (Page 6511).

You must make the following changes in the I/O address for a safety monitor with, for example, AS-Interface slave address 2 and address 272 of the AS-Interface master:

	Name	I/O address	Read only	Data type	Field length
1	⊞ myperiincp3432	PIB 272		Array	16
2	⊞ myperioutcp3432	PQB 272	☐	Array	16

Figure 8-73 Changing the address of the I/O variable for access to the AS-Interface slaves in the CP 343-2 P address area at address 272

This results in the following assignment for the ASIsafe safety monitor with AS-Interface slave address 2 in the CP 343-2 P address area (corresponds to the table titled "Binary addressing of the AS-Interface slaves in the AS-Interface master CP 343-2 P", in Chapter Addressing the AS-Interface master (CP 343-2 P, DP/AS-Interface Link 20E/Link Advanced) (Page 6512)):

Bits **4..7** in input byte **1** with address 272 → myPeriInCP3432[**1**]
Bits **4..7** in output byte **1** with address 272 → myPeriOutCP3432[**1**]

Adaptation of the transfer of the input and output data to FB **_ASI_rdAsiMonDiagnostic**:

```
myRdAsiMonDiagnostic (.......
        periIn0 := ((myPeriInCP3432[1] AND 16#10)<>0),        // Bit 4
        periIn1 := ((myPeriInCP3432[1] AND 16#20)<>0),        // Bit 5
        periIn2 := ((myPeriInCP3432[1] AND 16#40)<>0),        // Bit 6
        periIn3 := ((myPeriInCP3432[1] AND 16#80)<>0),        // Bit 7
        .......
        );
```

```
// Selective deletion of bits 4..7 in output byte 1
myPeriOut_cp343_2[1]:= myPeriOut_cp343_2[1] AND 16#0F;
myPeriOut_cp343_2[1]:= myPeriOut_cp343_2[1] OR
SHL(myRdAsiMonDiagnostic.periOut0,4);                    // Bit 4
myPeriOut_cp343_2[1]:= myPeriOut_cp343_2[1] OR
SHL(myRdAsiMonDiagnostic.periOut1,5);                    // Bit 5
myPeriOut_cp343_2[1]:= myPeriOut_cp343_2[1] OR
SHL(myRdAsiMonDiagnostic.periOut2,6);                    // Bit 6
myPeriOut_cp343_2[1]:= myPeriOut_cp343_2[1] OR
SHL(myRdAsiMonDiagnostic.periOut3,7);                    // Bit 7
```

If AS-Interface slave addresses other than the preset addresses 2 and 3 are used for the safety-oriented AS-Interface slaves, you must adapt the configuration of the safety monitors with the asimon configuration program and reload it to the safety monitor.

8.11.5.2 Sequence of the sample program

The sample program is ready to start with the preset addresses or after an adaptation of the addresses. You can check this on the cyclic change of the output parameters **done**/**busy**. The data in the data structure **Struct_ASI_diagAsiMon** must indicate the status of the enabling circuits and the status of the configured devices.

8.11.6 Appendix

8.11.6.1 SIMOTION and SIMATIC names

The following table contains a comparison of the SIMOTION and the SIMATIC names.

Table 8-153 SIMOTION and SIMATIC names

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	Name in the SIMATIC system
Function block parameters	
IN	
enable	-
reset	reset
periIn0	-
periIn1	-
periIn2	-
periIn3	-
allDevices	DataSelect_All_Devices
asiSlaveError	ASi_Slave_Fault
asiPowerError	ASi_Power_Fault
profibusError	Profibus_Fault

Name in the SIMOTION system as of V4.0 (command library in SCOUT)	**Name in the SIMATIC system**
diagStop	ASIMON_diag_stop_active
diagStopInvert	ASIMON_diag_stop_invert
freezeAuto	freeze_ASIMON_automat
freezeManual	freeze_ASIMON_manual
unfreezeIfCh1Off	unfreeze_if_chann_1_off
unfreezeIfCh2Off	unfreeze_if_chann_2_off
unfreezeIfDeviceWait	unfreeze_if_device_wait
OUT	
busy	-
done	diagnosis_done
doneTripped1	diag_done_puls_tripped_1
doneTripped2	diag_done_puls_tripped_2
error	diagnosis_error
errorID	retval
status	-
periOut0	-
periOut1	-
periOut2	-
periOut3	-
periOutDiagStop	ASIMON_diag_stop_address
ch1Off	Channel_1_off
ch2Off	Channel_2_off
diagStopActive	ASIMON_diag_stop_Tracing
configMode	Config_Mode
startup	Startup_Phase
IN/OUT	
diagDataAsiMonitor	ASIMON2_UDT
ch1Tripped	Channel_1_tripped
ch2Tripped	Channel_2_tripped
Data structure	
stateFb	status_function_block
stateAsiMonitor	status_monitor
stateChannel1	status_channel [1]
stateChannel2	status_channel [2]
sumChannel1	quantity [1]
sumChannel2	quantity [2]
device	

chInfo	device [xx].channel_info

state	device [xx].status
infoMaxDeviceChannel1	max_device_in_channel [1].index
stateMaxDeviceChannel1	max_device_in_channel [1].status
infoMaxDeviceChannel2	max_device_in_channel [2].index
stateMaxDeviceChannel2	max_device_in_channel [2].status
infoBits	info_bits
actTime	DT: actual_time Current time in the function block
timeCh1Off	DT: OSSD_1_off Time of the last shutdown of enabling circuit 1
timeCh1On	DT: OSSD_1_on Time of the last switch on of enabling circuit 1
timeCh2Off	DT: OSSD_2_off Time of the last shutdown of enabling circuit 2
timeCh2On	DT: OSSD_2_on Time of the last switch on of enabling circuit 2
timeConfig	DT: Config_Mode_asimon Time of the last switchover to the configuration mode (no ramp-up)
timeStartupAsiMonitor	DT: Startup_Phase_asimon Time of the last ramp-up of the safety monitor
timeError	DT: Fault_or_Error_FB Time of the last communication failure
timeStartupFb	DT: Startup_Diagnosis_FB Time of the last restart of the function block

8.11.6.2 List of abbreviations

Table 8-154 Abbreviations

Abbreviation	Meaning
AS	Actor sensor
ASI	Actor sensor interface
CP	Communications processor
DP	Decentralized peripherals
FB	Function block
IE	Industrial Ethernet
IN	Input parameter
IN/OUT	In/out parameter
LAD	Ladder diagram
OUT	Output parameter
PN	PROFINET

8.12 Standard Functions for RFID Systems

Preface

Contents of the function manual

This **document** is part of the **SIMOTION Programming - References documentation package**.

Function block

The function blocks **_PIB_001KB**, **_PIB_016KB** , and **_PIB_032KB** are part of the command library of the "SIMOTION SCOUT" engineering system.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.12.1 Fundamental safety instructions

8.12.1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

The non-observance of the safety instructions and residual risks stated in the associated hardware documentation can result in accidents with severe injuries or death.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.12.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.12.1.3 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

8.12.2 Description

8.12.2.1 General

This documentation describes data exchange between the SIMOTION system and standard profile RFID systems (PIB = Proxy Ident Block) using the function blocks (FB):

- **_PIB_001KB**
- **_PIB_016KB**
- **_PIB_032KB**

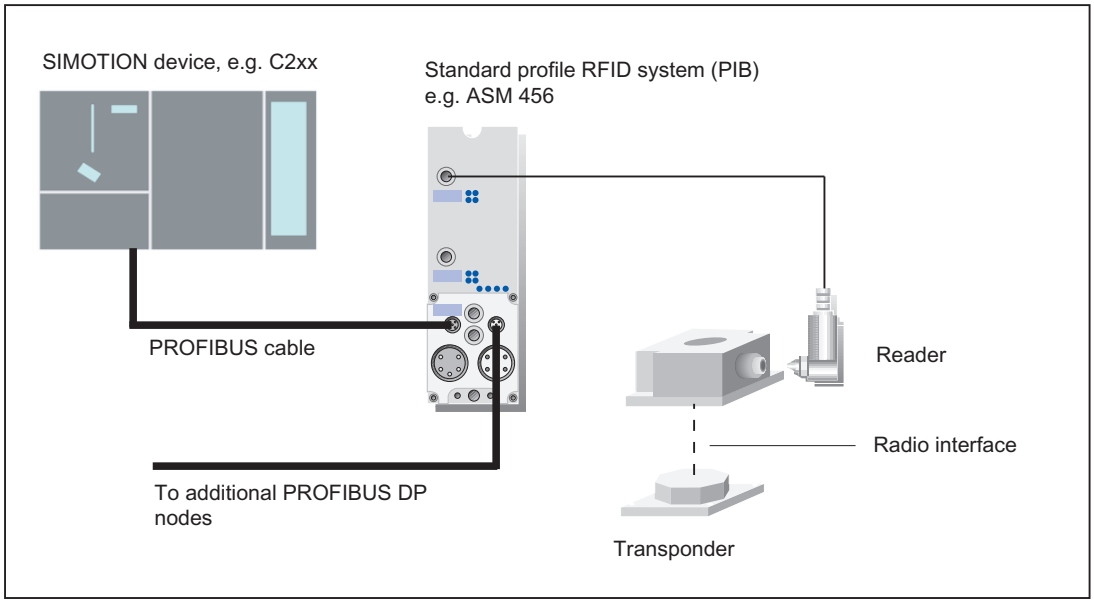


Figure 8-74 System overview

Components

The most important components of a SIMOTION application for writing to and reading from transponders are listed below along with their functions.

Table 8-155 Components

Component	Function
SIMOTION device	Motion control module with integrated function block for data exchange with standard profile RFID systems.
Standard profile RFID systems	Communication modules acting as an interface between the reader and the SIMOTION device.
Reader	Device for communication with transponders and for serial interfacing with standard profile RFID systems.
Transponder	Data memory mounted on the product or its transport or packaging unit that can be written, modified, and read using a contactless reader.

Note

This document describes only the operation of function blocks **_PIB_001KB**, **_PIB_016KB**, and **PIB_032KB**. For data exchange between the SIMOTION System and the standard profile RFID systems (PIB), you need the following documentation:

- SIMOTION SCOUT configuration manual
  This document is shipped with SIMOTION SCOUT in electronic form.
- Documentation of the corresponding standard profile RFID system (e.g. ASM 456, available on CD, Article No. 6GT2080-2AA10)
  - Hardware Description
  - Software Description

Requirements

The requirements for the standard functions described in this documentation are as follows:

- SIMOTION SCOUT V4.0 or higher
- SIMOTION Kernel V4.0 or higher
- Standard profile RFID system (PIB), e.g. ASM 456 V3.0 or higher, Article No. 6GT2002-0ED00

Note

The ASM 456 is integrated in the SIMOTION Motion Control System by means of a GSD file. The GSD file (device master file) contains all the properties specific to a slave. The format of a GSD file is defined in IEC 61784-1:2002 Ed1 CP 3/1.

For the current GSD file, go to:

http://support.automation.siemens.com/WW/view/en/22511087

Installing GSD files

1. In the STEP 7 "Hardware catalog", select the menu command **Options > Install GSD Files**.
2. In the dialog box that appears, open the drive/directory with the corresponding GSD file.
   Result: The DP slave is listed in the "Hardware catalog" window (only in the "Standard" catalog profile!) under "PROFIBUS DP\Additional Field Devices\Identsystems\MOBY\ASM456" and is available there for configuration.

Note

For further information about installing GSD files, refer to the STEP 7 Online Help.

8.12.2.2 Product description

The function blocks act as the communication interface between a standard profile RFID system (e.g. ASM 456) and the user program. The function blocks (FBs) support the following functions:

- Configuration
- Command processing
- Reading and writing of data
- Diagnostics

8.12.3 Programming

8.12.3.1 Integrating the function blocks in the user project

Creating the instance of the FBs in the user project

The function blocks are part of the command library of the "SIMOTION SCOUT" engineering system. To work with the Ident Unit, an instance of the function block must first be created in the user project.

The Ident Unit comprises the following components:

- Standard profile RFID system
- Reader

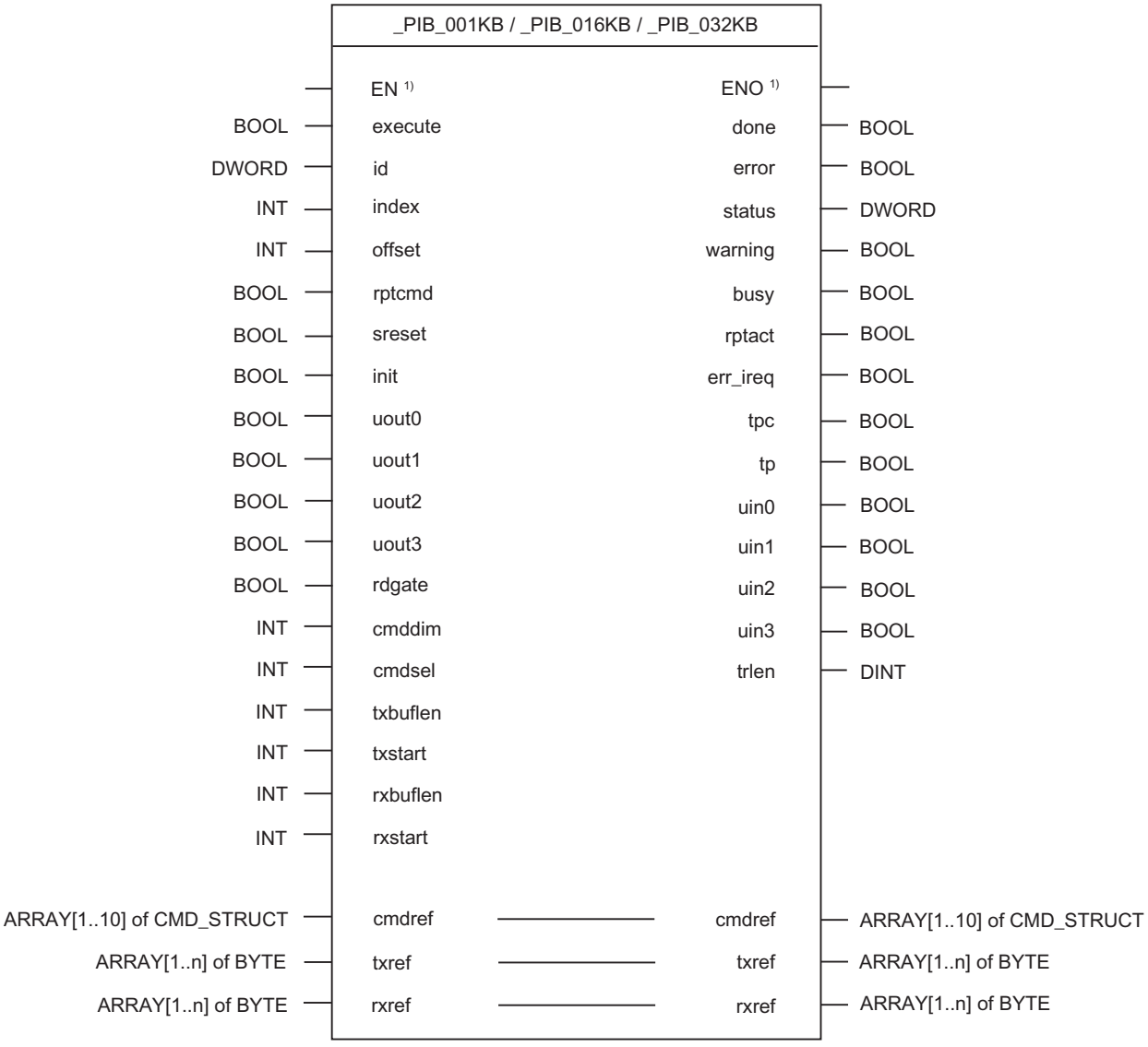
Note

For multi-channel Ident Units, one instance per channel must be created.

Example:

```
VAR_GLOBAL
    myFbPib001KB    : _PIB_001KB;    // instance of FB _PIB_001KB
END_VAR
```

Call (LAD representation)



The following values apply for n:

Function block	Value (in bytes)
_PIB_001KB	n := 1024
_PIB_016KB	n := 16384
_PIB_032KB	n := 32767

1) LAD-specific parameters

Application example

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION hardware platforms. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

8.12.3.2 Addressing the field devices

Overview

The SIMOTION device and the Ident Unit communicate via the PROFIBUS DP. Addresses starting at 256 are recommended for the standard profile RFID system (e.g. ASM 456), whereby I/O variables are used for read- and write-accessing the I/O.

Creating I/O variables

In the SIMOTION user project, two I/O variables with the addresses taken from **HW Config** must be created for reading and writing of cyclic data.

The parameters must be assigned in the Symbol Browser (refer to the following example).

Example for the ASM 456

Determining the addresses from **HW Config**:

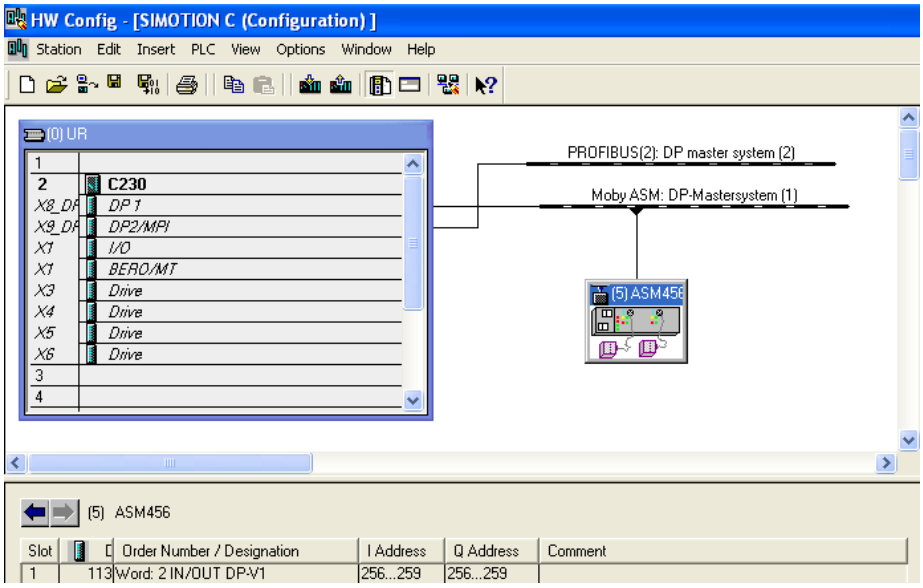


Figure 8-75 Addresses in HW Config

The ASM 456 provides two read/write channels. Each channel communicates cyclically with the controller by means of a control word (direction: controller to ASM 456) and a status word (direction: ASM 456 to the controller). The addresses of both the status words (ZSW) and the control word (STW) are specified via the start addresses of the ZSW/STW of the first channel. The start addresses of the ZSW/STW of the second channel are connected directly.

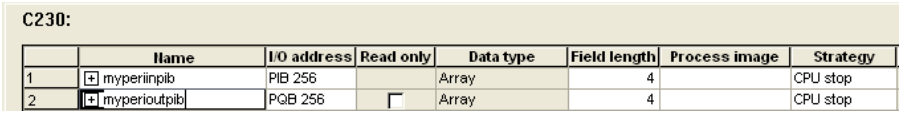
The start addresses of the input and output address range must be identical. In the example, the start address is 256.

This results in the following addresses for the ZSW/STW:

- ZSW/STW of ASM 456 channel X1 are located on addresses 256/257
- ZSW/STW of ASM 456 channel X2 are located on addresses 258/259

The I/O variables must be created for each module.

Creating the I/O Variables in the Symbol Browser:


C230:

	Name	I/O address	Read only	Data type	Field length	Process image	Strategy
1	myperiinpib	PIB 256		Array	4		CPU stop
2	myperioutpib	PQB 256		Array	4		CPU stop

Figure 8-76 Addressing with I/O variables

Notice: The I/O addresses in the Symbol Browser shown in the figure have the following meanings:

- "PIB" = Peripheral Input: Byte
- "PQB" = Peripheral Output: Byte

Parameter transfer

The above-indicated addresses of the status and command words are passed to the input parameter **id** of the corresponding function block instance.

Note

For additional information, refer to:

- *SIMOTION SCOUT* online help
- Programming Manual of the corresponding programming language, e.g.:
  - *SIMOTION ST, Structured Text* Programming Manual
  - *SIMOTION MCC, Motion Control Chart* Programming Manual
  - *SIMOTION LAD/FBD, Ladder Diagram and Function Block Diagram* Programming Manual

These documents are shipped with SIMOTION SCOUT in electronic form.

8.12.4 Parameter assignment

8.12.4.1 Function blocks _PIB_001KB / _PIB_016KB / _PIB_032KB

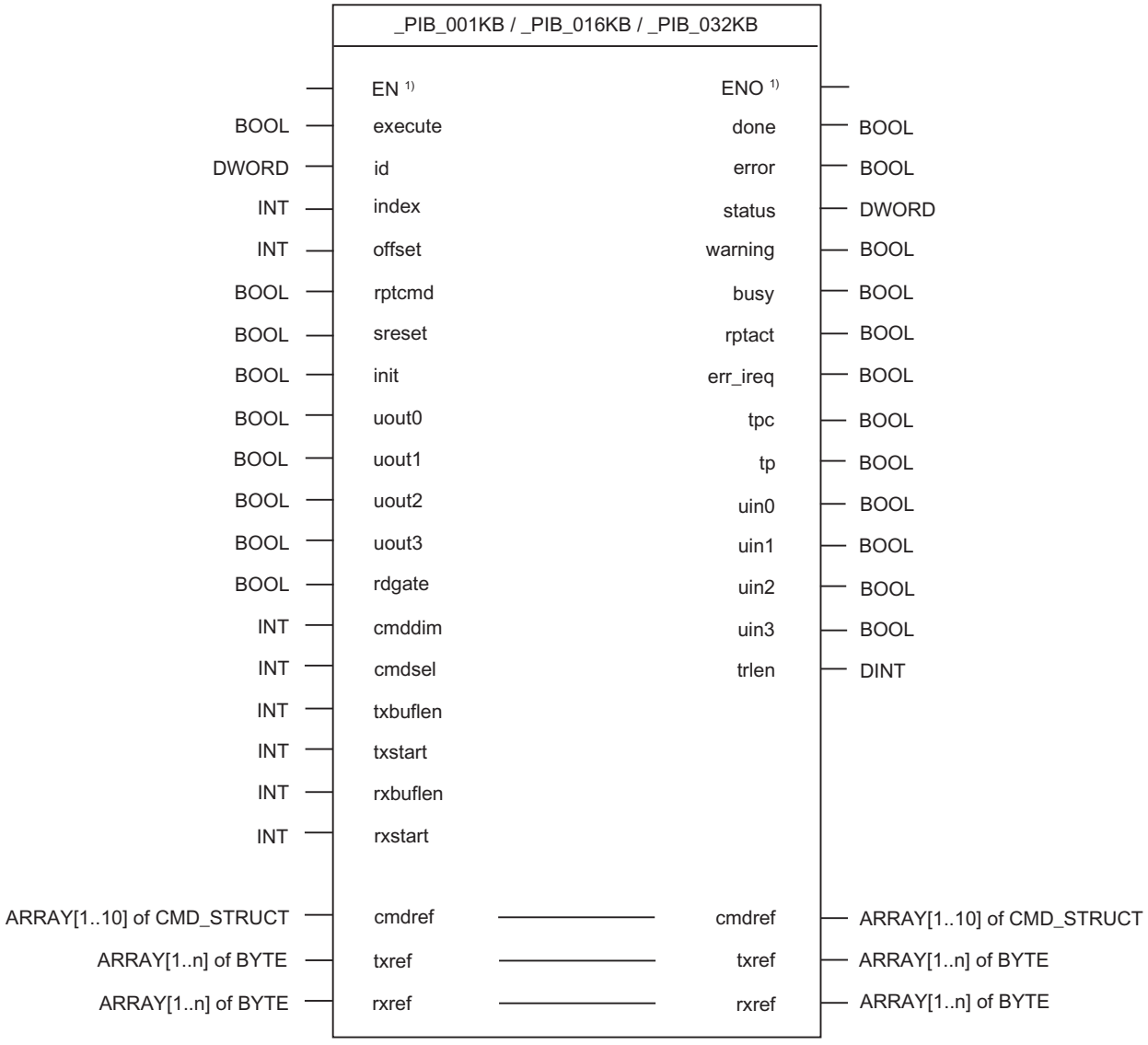
Introduction

With the function blocks **_PIB_001KB**, **_PIB_016KB** , and **_PIB_032KB** , data can be exchanged between SIMOTION and standard profile RFID systems.

Use of the FBs depends on the amount of data to be transferred. The following table contains an overview of the maximum amount of data that can be transferred with the FBs during a read/ write operation.

Function block	Maximum data amount per command call:
_PIB_001KB	1 KB
_PIB_016KB	16 KB
_PIB_032KB	32 KB

Call (LAD representation)



The following values apply for n:

Function block	Value (in bytes)
_PIB_001KB	n := 1024
_PIB_016KB	n := 16384
_PIB_032KB	n := 32767

1) LAD-specific parameters

Parameter description

The following table contains the parameters of FBs **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB**.

Note

For detailed information about the use of parameters, see the documentation for the standard profile RFID system, e.g. ASM 456.

Table 8-156 Parameters of function blocks _PIB_001KB / _PIB_016KB / _PIB_032KB

Name	Type	Data type	Default	Meaning
execute	IN	BOOL	FALSE	**TRUE** = Initiating a new request (command) with rising edge Before starting the block operation, the command and the corresponding parameters must be set by the user in the entry in the **cmdref** field selected with **cmdsel**.
id	IN	DWORD	0	Start address of the I/O device **Notice:** The start addresses of the in/out area must be identical. See figure "Addresses in HW Config".
index	IN	INT	0	Identification of the data of a channel: • Channel 1: 101/111 • Channel 2: 102/112 • Channel 3: 103/113 • Channel 4: 104/114 • Channel 5: 105/115 • Channel 6: 106/116 • Channel 7: 107/117 • Channel 8: 108/118 The 10th position of the **index** parameter has the following meanings: • 0 = Parameter assignment of channel • 1 = Data transfer for the channel
offset	IN	INT	0	Address offset from the base address (input parameter **id**) of the channel-related cyclic I/O data. • Channel 1: 0 • Channel 2: 2 • Channel 3: 4 • Channel 4: 6 • Channel 5: 8 • Channel 6: 10 • Channel 7: 12 • Channel 8: 14
rptcmd	IN	BOOL	FALSE	**TRUE** = Repeating the command currently being executed or the next command to be executed by the Ident Unit.

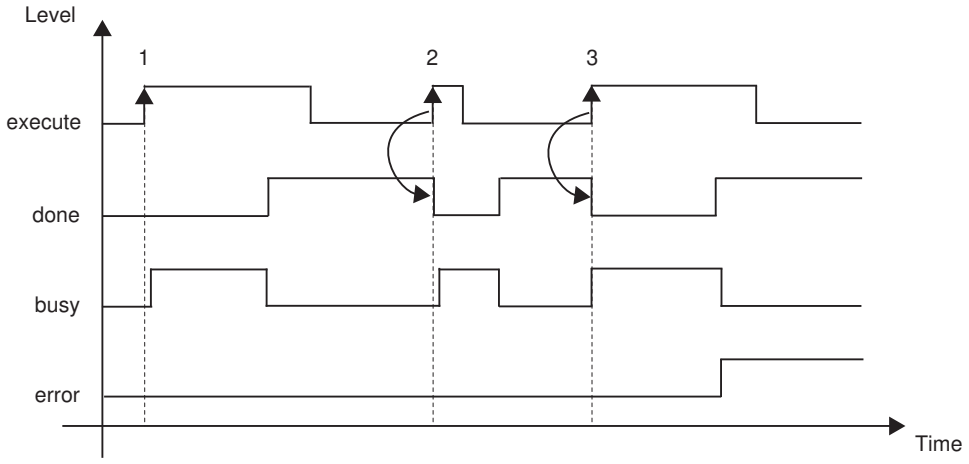
Name	Type	Data type	Default	Meaning
sreset	IN	BOOL	FALSE	**TRUE** = Aborting the command currently being processed in the Ident Unit.
init	IN	BOOL	FALSE	**TRUE** = Ident Unit is restarting operation
uout0	IN	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uout1	IN	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uout2	IN	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uout3	IN	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
rdgate	IN	BOOL	FALSE	**TRUE** = Activating read gate
cmddim	IN	INT	0	Number of commands in parameter **cmdref**
cmdsel	IN	INT	0	Selection of command to be executed
txbuflen	IN	INT	0	Number of bytes used by this instance of the FB to store data to be sent.
txstart	IN	INT	0	Relative position of the send data buffer (TXBUF) within the global memory area to which the **txref** parameter refers.
rxbuflen	IN	INT	0	Number of bytes used by this instance of the FB to store data received.
rxstart	IN	INT	0	Relative position of the receive data buffer (RXBUF) within the global memory area to which the **rxref** parameter refers.
cmdref	IN/OUT	ARRAY[1..10] of CMD_STRUCT		Array that can accept 10 commands The commands are complex variables of type **CMD_STRUCT** (for a detailed description, see section "Commands of function blocks").
txref	IN/OUT	ARRAY[1..n] of BYTE		Transmit data Reference to a global memory area used by several blocks. The instance of the FB can share the memory with several other instances. The following values apply for n, according to the FBs: • n := 1024 (FB **_PIB_001KB**) • n := 16384 (FB **_PIB_016KB**) • n := 32767 (FB **_PIB_032KB**)
rxref	IN/OUT	ARRAY[1..n] of BYTE		Receive data Reference to a global memory area used by several blocks. The instance of the FB can share the memory with several other instances. The following values apply for n, according to the FBs: • n := 1024 (FB **_PIB_001KB**) • n := 16384 (FB **_PIB_016KB**) • n := 32767 (FB **_PIB_032KB**)
done	OUT	BOOL	FALSE	**TRUE** = Command was executed successfully
error	OUT	BOOL	FALSE	**TRUE** = Error has been detected, specification of error in parameter **status**
status	OUT	DWORD	16#00	Warning and error specification For **error** = **TRUE** or **warning** = **TRUE**, the **status** parameter contains the error or warning information (see "Error messages")
warning	OUT	BOOL	FALSE	**TRUE** = Warning has been detected, specification of warning in parameter **status**
busy	OUT	BOOL	FALSE	**TRUE** = FB is in use, and other commands are ignored (except for **init** and **sreset**)

Name	Type	Data type	Default	Meaning
rptact	OUT	BOOL	FALSE	**TRUE** = Request to repeat the current command was accepted by the RFID system. This parameter is identical to the "Repeat_Command_Active" signal bit from the cyclic control word of the RFID system and remains **TRUE** the same as this bit. As long as parameter **rptact** is set to **TRUE**, the RFID system supplies result data of the command execution, which the user must read.
err_ireq	OUT	BOOL	FALSE	**TRUE** = A fatal error has been detected by the RFID system. This parameter is identical to the "Error_Flag" signal bit from the cyclic control word of the RFID system and remains **TRUE** the same as this bit. The RFID system remains in the current state until a device status command is issued or the **err_ireq** parameter is reset by the user with **init** = **TRUE**.
tpc	OUT	BOOL	FALSE	**TRUE** = A new transponder is within range of Ident Unit or a transponder has left the range. The parameter is set to **FALSE** when the next "Inventory" command is successfully executed.
tp	OUT	BOOL	FALSE	**TRUE** = A transponder is within the range of the Ident Unit **Note**: This parameter is not used for barcode readers.
uin0	OUT	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uin1	OUT	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uin2	OUT	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
uin3	OUT	BOOL	FALSE	This bit can be assigned by the vendor of the RFID system.
trlen	OUT	DINT	0	Display of number of data elements received after successful execution of the command.

Description of functions

The function blocks **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB** implemented according to the PNO standard provide an interface by means of which the SIMOTION application can communicate with the corresponding Ident Units. Communication with the transponders is command-controlled.

Graphic overview of the functionality



Case 1: The execute input parameter remains TRUE until the FB call is complete. The execute input parameter must be reset after the done parameter has reached TRUE.

Case 2: The execute input parameter is only clocked for one call. The query is not aborted.

Case 3: Same as case 1; however, an error occurs.

Figure 8-77 Signal sequence diagram

Task integration (call)

The FBs **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB** are provided for calling in a cyclical task and must be called in this task.during each task cycle. The execution of a job can take several cycles. The FBs **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB** can be called in any cyclic tasks. There are no restrictions placed on the functionality of the FBs **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB**. A fixed time frame (e.g. **IPOSynchronousTask**) is not required for the processing.

Error messages and warnings

The error and warning concept can be found in section "Error messages".

See also

Error and warning concept (Page 6583)

8.12.4.2 Calling function blocks

Procedure

In order to be able to work with function blocks **_PIB_001KB**, **_PIB_016KB** and **_PIB_032KB** in your user project, proceed as follows (the numbers shown in the following program segment correspond to the steps below):

1. Create an array for the in/out parameters of the FB.
2. Create the function block instance (e.g. instance for FB **_PIB_032KB**).
3. Initialize function block.
4. Call instance of the function block.
5. Transfer input parameters.
6. The output parameters of the function block are accessed with <instance name of FB>.<name of output parameter>.

Note

The program segment is an extract from the supplied application example.

The application example is included on the "SIMOTION Utilities & Applications" CD-ROM and is available for various SIMOTION platforms. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

For multi-channel Ident Units, one instance per channel must be created.

Call example

```
UNIT E_PIB032;

INTERFACE

  VAR_GLOBAL CONSTANT

    PIB_COMMAND_BUFFER_DIM  : INT:= 10;             // size of the command buffer
                                                    // is maximal 10
     PIB_TX_BUFFER_LEN_32   : INT:= 1024 * 32 - 1;  // size of the receive buffer
                                                    // for a tag of 32k
    PIB_RX_BUFFER_LEN_32    : INT:= 1024 * 32 - 1;  // size of the transmit buffer
                                                    // for a tag of 32k
```

```
  PIB_CMD_WRITE_CONFIG  : INT := 16#78;   // command INIT - send device-specific
                                          // data and initialize
  PIB_CMD_FORMAT        : INT := 16#66;   // command FORMAT - delete all data
  PIB_CMD_CREATE        : INT := 16#68;   // command CREATE - create a read-only file
  PIB_CMD_GET_DIRECTORY : INT := 16#6d;   // command GET_DIRECTORY - read the
                                          // filesystem
  PIB_CMD_WRITE         : INT := 16#77;   // command WRITE - write data to the file

  // ...

  PIB_I_FH_ADR_IO       : DINT := 256;    // address of the ident unit
  PIB_I_FH_ADR_DIAG     : DINT := 2044;   // diagnostic address of the ident unit


  // The following data is manufacturer-specific and must be got from the
  // manufacturer. In this example it is the data for the ASM456 and can be got
  // from the documentation of the ASM456.

  // These are the config data for WRITE_CONFIG and FORMAT of the ASM456
  // manufacturer-specific data for the command "WRITE_CONFIG"
  ASM456_I_FH_WC_LEN      : INT := 14;
  ASM456_I_FH_WC_DAT      : ARRAY [1..ASM456_I_FH_WC_LEN] OF BYTE
                          := [4,0,0,0,0,8,16#42,16#34,16#30,16#FF,16#FF,0,0,1];

  // manufacturer-specific data for the command "FORMAT"
  ASM456_I_FH_FM_LEN      : INT := 21;
  ASM456_I_FH_FM_DAT_8    : ARRAY [1..ASM456_I_FH_FM_LEN] OF BYTE
                          := [8(0),16#0C,4(16#41,16#31),05,01,16#4C,0];
  ASM456_I_FH_FM_DAT_32   : ARRAY [1..ASM456_I_FH_FM_LEN] OF BYTE
                          := [8(0),16#0C,4(16#41,16#31),06,01,16#4C,0];

END_VAR


TYPE
// here are my commands
Enum_Commands: (
          cmdInit,    // command INIT - send device-specific data and initialize
          cmdFormat,       // command FORMAT - delete all data
          cmdCreate,       // command CREATE - create a read-only file
          cmdGetDirectory, // command GET_DIRECTORY - read the filesystem
          cmdWrite         // command WRITE - write data to the file
);

// ...
```

```
// these are the input-parameters for the FB
Struct_PIB_In  : STRUCT
  execute      : BOOL;        // execute a command
  id: DWORD;       // address of the peripheral device
  index        : INT;         // identifier for a single ident channel
  offset       : INT;         // offset of the buffer
  rptcmd       : BOOL;        // repeat a command
  sreset       : BOOL;        // software reset
  init         : BOOL;        // initialization
  uout0        : BOOL;        // represents a user specific bit 0
  uout1        : BOOL;        // represents a user specific bit 1
  uout2        : BOOL;        // represents a user specific bit 2
  uout3        : BOOL;        // represents a user specific bit 3
  rdgate       : BOOL;        // optional bit 8 of the cyclic control word
  cmddim       : INT;         // max. size of commands
  cmdsel       : INT;         // command to be selected
  txbuflen     : DINT;        // length of the transmit buffer
  txstart      : DINT;        // start of the transmit buffer
  rxbuflen     : DINT;        // length of the receive buffer
  rxstart      : DINT;        // start of the receive buffer
END_STRUCT

// these are the output-parameters for the FB
Struct_PIB_Out : STRUCT
  done         : BOOL   := TRUE;     // command successful
  error        : BOOL   := FALSE;    // error occurred
  status       : DWORD  := 0;        // status
  warning      : BOOL   := FALSE;    // warning
  rptact       : BOOL   := FALSE;    // command repetition active
  err_ireq     : BOOL   := FALSE;    // init request
  tpc          : BOOL   := FALSE;    // count present tag
  tp           : BOOL   := FALSE;    // tag present
  uin0         : BOOL   := FALSE;    // user specific bit 0
  uin1         : BOOL   := FALSE;    // user specific bit 1
  uin2         : BOOL   := FALSE;    // user specific bit 2
  uin3         : BOOL   := FALSE;    // user specific bit 3
  trlen        : DINT   := 0;        // length of the transmit buffer
END_STRUCT

// ...

END_TYPE


VAR_GLOBAL
  myCmdBuf : ARRAY [1..PIB_COMMAND_BUFFER_DIM] OF _PIB_COMMAND; // the command      (1)
                                                                // buffer
  myTxBuf : ARRAY [1..PIB_TX_BUFFER_LEN_32] OF BYTE;  // the transmit buffer
  myRxBuf : ARRAY [1..PIB_RX_BUFFER_LEN_32] OF BYTE;  // the receive buffer
```

```
    myIn     : Struct_PIB_In;     // the input parameters                          (2)
    myOut    : Struct_PIB_Out;    // the output parameters
    myFbPIB  : _PIB_032KB;        // the FB

    my_TX_BUFFER_LEN    : INT := PIB_TX_BUFFER_LEN_32;  // the size of the
                                                        // transmit buffer
    my_RX_BUFFER_LEN    : INT := PIB_RX_BUFFER_LEN_32;  // the size of the
                                                        // receive buffer

    myPIBAdrDiag        : DINT := PIB_I_FH_ADR_DIAG;    // the diagnostic address
    myPIBAdrIO          : DINT := PIB_I_FH_ADR_IO;      // the io address

    // ...

    END_VAR


    PROGRAM StartUpPIB;             // program for StartupTask
    PROGRAM ShutDownPIB;            // program for ShutdownTask
    PROGRAM PeripheralFaultPIB;     // program for PeripheralFaultTask
    PROGRAM TechnologicalFaultPIB;  // program for TechnologicalFaultTask
    PROGRAM ExamplePIB;         // program for BackgroundTask with call FB _PIB_32k

END_INTERFACE

IMPLEMENTATION

// ...

// StartUpTask

PROGRAM StartUpPIB

  // ...

  // set default values
```

```
myIn.execute  := FALSE;                      // execute a command                    (3)
myIn.id       := DINT_TO_DWORD(myPIBAdrIO); // address of the peripheral device
myIn.index    := 111;                     // identifier for a single ident channel
                                            // (channel X1 and data transfer)
myIn.offset   := 0;                         // offset of the buffer
myIn.rptcmd   := FALSE;                     // repeat a command
myIn.sreset   := FALSE;                     // software reset
myIn.init     := FALSE;                     // initialization
myIn.uout0    := FALSE;                     // represents a user specific bit 0
myIn.uout1    := FALSE;                     // represents a user specific bit 1
myIn.uout2    := FALSE;                     // represents a user specific bit 2
myIn.uout3    := FALSE;                     // represents a user specific bit 3
myIn.rdgate   := FALSE;             // optional bit 8 of the cyclic control word
myIn.cmddim   := PIB_COMMAND_BUFFER_DIM;   // dimension of the command buffer
myIn.cmdsel   := 1;                         // command to be selected
myIn.txbuflen := my_TX_BUFFER_LEN;          // length of the transmit buffer
myIn.txstart  := 1;                         // start of the transmit buffer
myIn.rxbuflen := my_RX_BUFFER_LEN;          // length of the receive buffer
myIn.rxstart  := 1;                         // start of the receive buffer

END_PROGRAM                                 // End StartUpPIB


// this is the example program for the use of the PIB
// it must be in the backgroundtask
PROGRAM ExamplePIB

// ...

  // call the FB
  myFbPIB (                                                                      (4)
      execute     := myIn.execute    // execute the selected command (CMDSEL)   (5)
    , id          := myIn.id         // address of the peripheral device
    , index       := myIn.index      // identifier for a single ident channel
    , offset      := myIn.offset     // offset of the buffer
    , rptcmd      := myIn.rptcmd     // repeat a command
    , sreset      := myIn.sreset     // software reset
    , init        := myIn.init       // initialization
    , uout0       := myIn.uout0      // represents a user specific bit 0
    , uout1       := myIn.uout1      // represents a user specific bit 1
    , uout2       := myIn.uout2      // represents a user specific bit 2
    , uout3       := myIn.uout3      // represents a user specific bit 3
    , rdgate      := myIn.rdgate     // optional bit 8 of the cyclic control word
    , cmddim      := myIn.cmddim     // max. size of commands
    , cmdsel      := myIn.cmdsel     // command to be selected
    , txbuflen    := myIn.txbuflen   // length of the transmit buffer
    , txstart     := myIn.txstart    // start of the transmit buffer
    , rxbuflen    := myIn.rxbuflen   // length of the receive buffer
    , rxstart     := myIn.rxstart    // start of the receive buffer
    , cmdref      := myCmdBuf        // command buffer
    , txref       := myTxBuf         // transmit buffer
    , rxref       := myRxBuf         // receive buffer
  );
// end call FB
```

```
  // get FB-Output parameter
    myOut.done      := myFbPIB.done;   // command successful
    myOut.error     := myFbPIB.error;  // error occurred

    IF ( myOut.error = TRUE )              // only to prevent misunderstandings      (6)
    THEN
        myOut.status   := myFbPIB.status;   // get status
    ELSE
        myOut.status   := 0;                // reset status
    END_IF;

    myOut.warning     := myFbPIB.warning;     // warning
    myOut.rptact      := myFbPIB.rptact       // command repetition active
    myOut.ireq        := myFbPIB.ireq         // init request
    myOut.tpc         := myFbPIB.tpc;         // count present tag
    myOut.tp          := myFbPIB.tp;          // tag present
    myOut.uin0        := myFbPIB.uin0;        // user specific bit 0
    myOut.uin1        := myFbPIB.uin1;        // user specific bit 1
    myOut.uin2        := myFbPIB.uin2;        // user specific bit 2
    myOut.uin3        := myFbPIB.uin3;        // user specific bit 3
    myOut.trlen       := myFbPIB.trlen;       // length of the transmit buffer
    // end get FB-Output parameter

  END_PROGRAM

END_IMPLEMENTATION
```

8.12.4.3 Commands of function blocks

Overview of commands

This section describes the commands that are supported by the function blocks **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB** along with the associated parameters.

Note that the commands must also be supported by the corresponding Ident Unit. For a list of commands supported by the RFID system, consult the standard profile RFID system (e.g. ASM 456) documentation.

Commands

The following overview shows all commands supported by the standard profile. Different commands are available, depending on the connected reader and the settings in **HW Config** via the GSD file.

Table 8-157 Overview of commands

Command	Command code		Parameters used
Clear	16#63	'c'	UID, FileName
Create	16#68	'h'	UID, FileName, length, attributes, FileType
Delete	16#64	'd'	UID, FileName

Command	Command code		Parameters used
Dev status	16#74	't'	Attributes, OffsetBuffer
Format	16#66	'f'	OffsetBuffer, UID, Length
Get	16#62	'b'	OffsetBuffer, Length
Get attribute	16#6B	'k'	UID, FileName
Get directory	16#6D	'm'	OffsetBuffer, UID, FileType
Inventory	16#69	'i'	Attributes, OffsetBuffer
Mem status	16#73	's'	UID, attributes, OffsetBuffer
Next	16#6E	'n'	UID, NextMode
Physical read	16#70	'p'	OffsetBuffer, UID, length, StartAddress
Physical write	16#71	'q'	OffsetBuffer, UID, length, StartAddress
Put	16#65	'e'	OffsetBuffer, Length
Read	16#72	'r'	OffsetBuffer, UID, FileName, offset, length
Read BarCode	16#76	'v'	OffsetBuffer, TimeOut, ObjectNumber
Read config	16#61	'a'	OffsetBuffer
Set attribute	16#6F	'o'	UID, FileName, attributes, FileType
Update	16#75	'u'	OffsetBuffer, UID, FileName, length
Write	16#77	'w'	OffsetBuffer, UID, FileName, offset, length
Write config	16#78	'x'	OffsetBuffer, length, config

Table 8-158 Overview of command codes

Command code		Command	Parameters used
16#61	'a'	Read config	OffsetBuffer
16#62	'b'	Get	OffsetBuffer, Length
16#63	'c'	Clear	UID, FileName
16#64	'd'	Delete	UID, FileName
16#65	'e'	Put	OffsetBuffer, Length
16#66	'f'	Format	OffsetBuffer, UID, Length
16#68	'h'	Create	UID, FileName, length, attributes, FileType
16#69	'i'	Inventory	Attributes, OffsetBuffer
16#6B	'k'	Get attribute	UID, FileName
16#6D	'm'	Get directory	OffsetBuffer, UID, FileType
16#6E	'n'	Next	UID, NextMode
16#6F	'o'	Set attribute	UID, FileName, attributes, FileType
16#70	'p'	Physical read	OffsetBuffer, UID, length, StartAddress
16#71	'q'	Physical write	OffsetBuffer, UID, length, StartAddress
16#72	'r'	Read	OffsetBuffer, UID, FileName, offset, length
16#73	's'	Mem status	UID, attributes, OffsetBuffer
16#74	't'	Dev status	Attributes, OffsetBuffer
16#75	'u'	Update	OffsetBuffer, UID, FileName, length
16#76	'v'	Read BarCode	OffsetBuffer, TimeOut, ObjectNumber

Command code		Command	Parameters used
16#77	'w'	Write	OffsetBuffer, UID, FileName, offset, length
16#78	'x'	Write config	OffsetBuffer, length, config

General information

The following restrictions relate to the use of commands:

- The input parameters **init** and **sreset** interrupt the command execution within the Ident Unit.
- After the commands **init** and **sreset** are sent, the subsequent change of the output parameter **done** relates to the commands **init** or **sreset** and not to the command that was interrupted by the input parameters **init** or **sreset**.
- The input parameter **init** resets the communication (cyclic control and status flow, acyclic commands) between FB and Ident Unit. Afterwards, the FB automatically executes the "Write config" command. This assumes that the user enters the "Write config" command in the first element of the in/out parameter **cmdref** and sets the input parameter **cmdsel** to "1". Any configuration data to be written must be in place at the location in the send data buffer indicated by the input parameter **txref**, to which the command parameters **OffsetBuffer** and **Length** refer.
- The "Write config" command resets all functions within the Ident Unit, except the communication.
- The **sreset** parameter cancels the last command. The execution of **sreset** is considerably faster than that of **init**.

Data structure CMD_STRUCT

The data structure CMD_STRUCT contains the parameters of the command. The data types of the parameters are defined in the following program excerpt.

```
TYPE
    CMD_STRUCT
        STRUCT
            CMD          : BYTE;
            Config       : BYTE;
            OffsetBuffer : INT;
            UID          : ARRAY[1..8] OF BYTE;
            FileName     : ARRAY[1..8] OF BYTE;
            Offset       : DINT;
            Length       : INT;
            StartAddress : DINT;
            Attributes   : BYTE;
            NextMode     : BYTE;
            Timeout      : INT;
            ObjectNumber : INT;
            FileType     : WORD;
        END_STRUCT;
END_TYPE
```

Note

Not every command occupies every structure element.

Clear

Description

The "Clear" command resets the content of a file. All elements are set to "16#00".

Command code of the command

```
CMD : BYTE := 16#63;
```

Parameters

The following table lists the parameters of the "Clear" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed.

Create

Description

The "Create" command creates a new file on a formatted transponder.

Command code of the command

```
CMD : BYTE := 16#68;
```

Parameters

The following table lists the parameters of the "Create" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be generated
Length	This parameter indicates the length of the file to be created. **Length** = 0: A file with dynamic length is created.
Attributes	Valid values: • Unlimited read/write = bit 0,1 not set • Read only = bit 0 set • Single read = bit 1 set • Defined size = bit 2 set (the file length cannot be changed by a command)
FileType	Classification/grouping of the files according to user-specific criteria. Files with the same characteristics are assigned to the same group. Accordingly, these files are of the same file type that is displayed with the same value of the **FileType** parameter. If the parameter is not used, all bytes must be set to "16#20".

Delete

Description

The "Delete" command deletes a file from the transponder. Depending on the supported file system, the file disappears from the directory or the file length and the used length are set to zero.

Command code of the command

```
CMD : BYTE := 16#64;
```

Parameters

The following table lists the parameters of the "Delete" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed

Dev status

Description

The "Dev status" command is used to read out the status of an Ident Unit. The status data are stored in RXBUF. Status data are vendor-specific. The output parameter **trlen** of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#74;
```

Parameters

The following table lists the parameters of the "Dev status" command.

Parameters	Description
Attributes	Information class to be read Valid values: • 16#00 → reserved • 16#01 → warning detail (vendor-specific details) • 16#02 → error history (vendor-specific details) • 16#03 → command history (vendor-specific details) • 16#04 → channel-specific identification and maintenance information (I&M information) (data recording I&M0) • 16#05 → channel-specific I&M information (data recording I&M1) • 16#06 → channel-specific I&M information (data recording I&M2) • 16#07 → channel-specific I&M information (data recording I&M3) • 16#08 → channel-specific I&M information (data recording I&M4) • 16#09 to 16#7F → reserved • 16#80 to 16#FF → vendor-specific
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the data to be read is to be stored.

Format

Description

The "Format" command initializes the transponder. After formatting, the transponder is ready for use.

A Security_Code (STRING[8]) must be set at the beginning of the TXBUF. This code is a vendor-specific code for defining the transponder name and protecting against formatting with

incorrect parameters and arbitrary formatting. TXBUF is used as a vendor-specific area for parameter data that are required for formatting. For more information, see the Ident Unit documentation.

Command code of the command

```
CMD : BYTE := 16#66;
```

Parameters

The following table lists the parameters of the "Format" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the Security_Code to be sent is to be stored. Other data follow consecutively.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
Length	Number of bytes to be sent to the Ident Unit The start address is designated by the command parameter **OffsetBuffer**. This parameter contains the 8 bytes of the Security_Code and a variable number of formatting parameters. The range of values is: 8...226.

Get

Description

The "Get" command reads out vendor-specific data located on the Ident Unit. TXBUF is used as the vendor-specific area for parameter data (optional send data). Received data are stored from the start of RXBUF. The output parameter **trlen** of the FB indicates the number of received bytes.

For the precise meaning of the data, refer to the Ident Unit documentation.

Command code of the command

```
CMD : BYTE := 16#62;
```

Parameters

The following table lists the parameters of the "Get" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the parameter data to be sent is to be stored. Other parameter data follow consecutively.
Length	Number of bytes to be sent to the Ident Unit The start address is designated by the command parameter **OffsetBuffer**. The range of values is: 0...226.

Get attribute

Description

The "Get attribute" command reads out attributes associated with a file. The attributes (attributes and file type) are stored in the directory.

Command code of the command

```
CMD : BYTE := 16#6B;
```

Parameters

The following table lists the parameters of the "Get attribute" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed.

The following table lists all subparameters that are passed with the response:

Parameter	Description
Attributes	Valid values: • Unlimited read/write = bit 0,1 not set • Read only = bit 0 set • Single read = bit 1 set • Defined size = bit 2 set (the file length cannot be changed with a command)
FileType	Classification/grouping of the files according to user-specific criteria. Files with the same characteristics are assigned to the same group. Accordingly, these files are of the same file type that is displayed with the same value of the **FileType** command parameter. If the parameter is not used, all bytes must be set to "16#20".

Get directory

Description

The "Get directory" command reads out the directory of the transponder. File names and associated attributes are passed. The **FileType** command parameter is used to indicate files of a specific type to be passed. The response data of this command are supplied in a structure within the RXBUF as follows:

```
TYPE
    DIRELEMENTS_STRUCT
        STRUCT
            FileName : ARRAY [1..8] OF BYTE;
            UsedLength   : DINT;
            Attributes : BYTE;
            FileLength   : DINT;
            FileType : WORD;
        END_STRUCT;
END_TYPE

TYPE
    DIRLIST_STRUCT
        STRUCT
            UID1         : ARRAY [1..8] OF BYTE;
            TagName      : ARRAY [1..8] OF BYTE;
            FreeUserMem DINT;
            CheckSum     : WORD;
            FileCount    : INT;
            FileList     : ARRAY [1..FileCount] OF DIRELEMENTS_STRUCT;
        END_STRUCT;
END_TYPE
```

The **trlen** parameter of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#6D;
```

Parameters

The following table lists the parameters of the "Get directory" command.

Parameter	Description
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the received data is to be stored.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileType	Classification/grouping of the files according to user-specific criteria. Files with the same characteristics are assigned to the same group. Accordingly, these files are of the same file type that is displayed with the same value of the **FileType** command parameter. If the parameter is not used, all bytes must be set to "16#20". "?" (16#3F) is used as a placeholder. "?,?" (16#3F, 16#3F) means all files.

The following table lists all subparameters that are passed with the response:

Parameter	Description
UID1	Identification of an individual transponder **UID1** = 0 indicates that the transponder does not contain a UID. If **UID** = 0 was to have been passed to the transponder within the command, then **UID1** supplies the **UID** of the present transponder.
TagName	This parameter indicates the name of the transponder.
FreeUserMem	This parameter indicates the unused memory on the transponder.
CheckSum	Checksum of the directory on the transponder If not used, all bytes are returned with "16#00".
FileCount	Number of directory entries that are available on the transponder and supplied by the **Get directory** command.
FileList	This parameter supplies the list of directory entries.
FileName	File name
UsedLength	Memory occupied by command parameter **FileLength**
Attributes	Valid values: • Unlimited read/write = bit 0,1 not set • Read only = bit 0 set • Single read = bit 1 set • Defined size = bit 2 set (the file length cannot be changed by a command)

Parameter	Description
FileLength	Maximum file length In the case of a dynamic file system, **FileLength** and **UsedLength** are identical. If **FileLength** = 0, then the file has been deleted.
FileType	Classification/grouping of the files according to user-specific criteria. Files with the same characteristics are assigned to the same group. Accordingly, these files are of the same file type that is displayed with the same value of the **FileType** command parameter. If the parameter is not used, all bytes must be set to "16#20".

Inventory

Description

The "Inventory" command is used to request a list of all currently accessible transponders within the antenna range. Additional information can be passed, depending on the specific vendor. Thus, in addition to the UIDs, user data can be read at the same time.

Note

The "Inventory" command causes the Ident Unit to reset the **tpc** output parameter. As soon as a transponder leaves the antenna range or another transponder enters into the antenna range, the output parameter **tpc** is set again. The "Inventory" command must be executed in order to determine how many transponders are located within the antenna range.

RXBUF is structured as follows:

```
// RXBUF:
//
// |     |     | | | ... | |  | | | ... | |  | | | ... | |  | | | ... | |  |
// | ON  | OL  |    UID       |     Data     |     UID      |    Data      | ...
// | INT | INT | ARRAY 8 Byte | ARRAY n Byte | ARRAY 8 Byte | ARRAY n Byte |
//               \_____________________________/   \_____________________________/
//                ObjectLength of Object 1      ObjectLength of Object 2     ...
// ON : ObjectNumber
// OL : ObjectLength, OL >= 8

TYPE
    ObjectNumber    : INT;
    ObjectLength     : INT;                                                    (1)
END_TYPE

TYPE
    UID_STRUCT
        STRUCT
            UID1    : ARRAY [1..8] OF BYTE;
            Data    : ARRAY [1..(ObjectLength-8)] OF BYTE;                    (2)
        END_STRUCT;
END_TYPE
```

```
TYPE
    UidList    : ARRAY [1..ObjectNumber] OF UID_STRUCT;
END_TYPE
```

Note

If the object length signalized by the RFID system under **(1)** is >8, a data array of length (ObjectLength - 8) must be declared in **(2)**. This length is dependent on the hardware you are using and can also assume the value "0".

Command code of the command

```
CMD : BYTE := 16#69;
```

Parameters

The following table lists the parameters of the "Inventory" command.

Parameter	Description
Attributes	Specification of the information to be read Valid values: • 16#00 → all UIDs are read • 16#01 to 16#7F → reserved • 16#80 to 16#FF → vendor-specific
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the data read is to be stored.

The following table lists all subparameters that are passed with the response:

Parameter	Description
ObjectNumber	Number of UIDs passed within the acknowledgement.
ObjectLength	Number of bytes connected to a single UID (size of UID and additional data). If **Attributes** = 16#00, then **ObjectLength** = 8 is set.
UidList	Data field with elements of the type **UID_Struct**. If **Attributes** = 16#00, only UIDs are stored, and no vendor-specific information is stored.

Mem status

Description

The "Mem status" command is used to read out the status of a transponder (battery status, memory, type of transponder, available capacity). The status data are stored in RXBUF. Status data are vendor-specific. The output parameter **trlen** of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#73;
```

Parameters

The following table lists the parameters of the "Mem status" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
Attributes	Specification of the information class to be read. Valid values: • 16#00 → reserved • 16#01 → warning detail • 16#02 → reserved • 16#03 → reserved • 16#04 → physical status information (vendor-specific details) • 16#05 → status information relating to the file system (vendor-specific details) • 16#06 to 16#7F → reserved • 16#80 to 16#FF → vendor-specific
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the data to be read is to be stored.

Next

Description

The "Next" command does not permit any further operations on a transponder. The subsequent command is not executed until the next transponder is detected/displayed.

Command code of the command

```
CMD : BYTE := 16#6E;
```

Parameters

The following table lists the parameters of the "Next" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is locked when this command is processed.
NextMode	Valid values: • **NextMode** = 0 (each transponder, irrespective of the current transponder, is processed) • **NextMode** = 1 (only another transponder is processed)

Physical read

Description

The "Physical read" command reads out data from a transponder by using the physical start address and the length of the data to be read. The output parameter **trlen** of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#70;
```

Parameters

The following table lists the parameters of the "Physical read" command.

Parameter	Description
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the received data must be saved. All subsequent bytes must be stored in ascending addresses.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.

Parameter	Description
Length	Number of bytes to be read by the transponder, beginning with the address identified by the **StartAddress** parameter.
StartAddress	This parameter indicates a physical address within the transponder.

Physical write

Description

The "Physical write" command writes data to a transponder using the physical start address and the length of the data to be written.

Command code of the command

```
CMD : BYTE := 16#71;
```

Parameters

The following table lists the parameters of the "Physical write" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the first address within the memory area where the first byte of the data to be sent is to be stored. Other data follow consecutively.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
Length	Number of bytes to be sent to the Ident Unit The target address is designated by the command parameter **StartAddress**.
StartAddress	Physical address within the transponder starting from which the data are written.

Put

Description

The "Put" command writes vendor-specific data to the Ident Unit.

For the precise meaning of the data, refer to the Ident Unit documentation.

Command code of the command

```
CMD : BYTE := 16#65;
```

Parameters

The following table lists the parameters of the "Put" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the data to be sent is to be stored. Other data follow consecutively.
Length	Number of bytes to be sent to the Ident Unit The start address is designated by the command parameter **OffsetBuffer**.

Read

Description

The "Read" command reads data of a file from the Ident Unit. After successful execution of the command, the data are saved in RXBUF. The output parameter **trlen** of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#72;
```

Parameters

The following table lists the parameters of the "Read" command.

Parameter	Description
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the received data must be saved. All subsequent bytes must be stored in ascending addresses.
UID	Identification of a transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	This parameter indicates the file to be accessed.
Offset	Relative offset within the indicated file from which the data are to be read. This parameter indicates the start address within the file from which the data are to be read.
Length	Number of bytes to be read. Length = -1: complete file is read

Read BarCode

Description

The "Read BarCode" command reads the barcode data.

After this command is sent, the barcode reader waits for a reading gate. Depending on the operating mode of the barcode reader, the reading gate is activated by a local input on the barcode reader or by the **rdgate** input parameter of the FB.

Likewise, the end of the read operation depends on the operating mode of the barcode reader. It can be initiated by the gate signal, a local timer, or the receipt of the expected barcode data (timer).

The output parameter **done** is activated by the function block once the barcode reader has finished reading and the result of the read operation has been stored in RXBUF.

The command parameter **ObjectNumber** can be used to send an object identifier to the barcode reader. The barcode reader can place this identifier in a specific position in the data string of the read result, which makes the link between the object and the barcode data more easily recognized by the user.

If the "Read BarCode" parameter is repeated by activation of the input parameter **rptcmd** of the FB, the command parameter **ObjectNumber** is incremented.

Command code of the command

```
CMD : BYTE := 16#76;
```

Parameters

The following table lists the parameters of the "Read BarCode" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the data to be sent is to be stored.
Timeout	Maximum time for a read operation before a timeout is generated. If the value of the command parameter **Timeout** is set to "0", time monitoring does not occur. The basic setting for this parameter is 10 ms.
ObjectNumber	Linking of an object with its barcode information The barcode reader can set this number in its read result message frame. If the input parameter **rptcmd** of the FB is active, the **ObjectNumber** is incremented with each read operation. Value range of the parameter: 0...1023 After the value 1023 has been exceeded, the counting begins with "0".

Read config

Description

The "Read Config" command is used to read out configuration data from the Ident Unit. RXBUF is used as an area for configuration data. Configuration data are vendor-specific. The output parameter **trlen** of the FB indicates the number of received bytes.

Command code of the command

```
CMD : BYTE := 16#61;
```

Parameters

The following table lists the parameters of the "Read Config" command.

Parameter	Description
OffsetBuffer	Relative offset within the RXBUF This parameter indicates the address within the memory area where the first byte of the data to be read is to be stored.

Set attribute

Description

The "Set attribute" command sets/changes the attributes associated with a file. These attributes are stored in the directory on the transponder.

Command code of the command

```
CMD : BYTE := 16#6F;
```

Parameters

The following table lists the parameters of the "Set attribute" command.

Parameter	Description
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed.

Parameter	Description
Attributes	Valid values: • Unlimited read/write = bit 0,1 not set • Read only = bit 0 set • Single read = bit 1 set • Defined size = bit 2 set (the file length cannot be changed with a command)
FileType	Classification/grouping of the files according to user-specific criteria. Files with the same characteristics are assigned to the same group. Accordingly, these files are of the same file type that is displayed with the same value of the **FileType** command parameter. If the parameter is not used, all bytes must be set to "16#20".

Update

Description

The "Update" command writes data to a file located on the transponder. The file length is updated exactly to the number of written data. This command always refers to the entire file.

Command code of the command

```
CMD : BYTE := 16#75;
```

Parameters

The following table lists the parameters of the "Update" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the data to be sent is to be stored.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed.
Length	Number of bytes to be written to the file

Write

Description

The "Write" command writes data to a file located on the transponder. If the number of bytes to be written is smaller than the file length, bytes that are not overwritten remain unchanged.

Command code of the command

```
CMD : BYTE := 16#77;
```

Parameters

The following table lists the parameters of the "Write" command.

Parameters	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the data to be sent is to be stored.
UID	Identification of an individual transponder **UID** = 0: Any random transponder The transponder currently located in the Ident Unit is read.
FileName	File to be accessed.
Offset	Relative offset within the indicated file to which the data are to be written. • **Offset** = 0: data begin at file start • **Offset** = -1: data are added to the file
Length	Number of bytes to be written to the file

Write config

Description

The "Write config" command is used to modify operation of the Ident Unit, except for interruption of the communication. It is possible to send new parameters to the Ident Unit (configuration data). A reset can also be initiated to prompt the Ident Unit to restart operation ("Restart"). TXBUF is used as an area for configuration data. Configuration data are vendor-specific.

Refer to the documentation of the relevant standard profile RFID system to learn which values the **Config** parameter can accept.

Note

The "Write config" command must always be in the first position in the command buffer (**cmdref** [1]). When the input parameter **init** is executed, the "Write config" command is automatically executed. The data are automatically retrieved from the first buffer. **Cmdsel** must be set to 1. If the "Write config" command is started with **execute**, the input parameter **cmdsel** must be set accordingly.

Command code of the command

```
CMD : BYTE := 16#78;
```

Parameters

The following table lists the parameters of the "Write config" command.

Parameter	Description
OffsetBuffer	Relative offset within the TXBUF This parameter indicates the address within the memory area where the first byte of the data to be sent is to be stored.
Length	Number of configuration data bytes to be written to the Ident Unit
Config	• **Config** = 0 → not permitted • **Config** = 1 → reset, no configuration data • **Config** = 2 → no reset, configuration data to be sent • **Config** = 3 → reset, configuration data to be sent • **Config** > 3 → reserved

8.12.4.4 Error messages

Overview

This chapter covers the error and warning concept of the function blocks **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB**.

If an error occurs, the output parameter **error** = **TRUE** is set. If a warning occurs, the output parameter **warning** = **TRUE** is set. More details about the error or warning are provided in output parameter **status**. In the event of an error, the relevant error number is displayed in this output parameter. If no errors or warnings have occurred, **status** has a value of "16#00".

Note

The **status** parameter is **not** reset when the next command is called and is available until the next error message. Evaluate the **status** parameter only when **error** = **TRUE**.

The **status** output parameter consists of a compact array (ARRAY) of 4 bytes. Bit 4 of byte 0 distinguishes between an error message and a warning:

- Bit 4 = 0 → error message pending
- Bit 4 = 1 → warning pending

Error and warning concept

The function blocks **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB** indicate that the requested command has been executed successfully. The error messages have the following functions:

1. Controlling of program execution
2. Output of error messages via the user program or the control system

status

In the event of an error, the corresponding error number is displayed in the **status** output parameter. If no errors or warnings have occurred, **status** has a value of "16#00". The **status** parameter is not reset when the next command is called and is available until the next error message.

The error number in the **status** output parameter consists of a compact array (ARRAY) of 4 bytes, which are described in the table below.

Table 8-159 Structure of 'status' output parameter

Byte	Name	Type	Definition
0	**Function_Num**	Byte	Grouping in error messages and warnings, see "Function_Num"
1	**Error_Decode**	Byte	Definition of the meaning of **Function_Num**, **Error_Code_**1, and **Error_Code_2**, see "Error_Decode"
2	**Error_Code_1**	Byte	Contains warning and error numbers, see "Error_Code_1"
3	**Error_Code_2**	Byte	Warnings or vendor-specific errors, see "Error_Code_2"

The following figure shows the structure of the **status** output parameter.

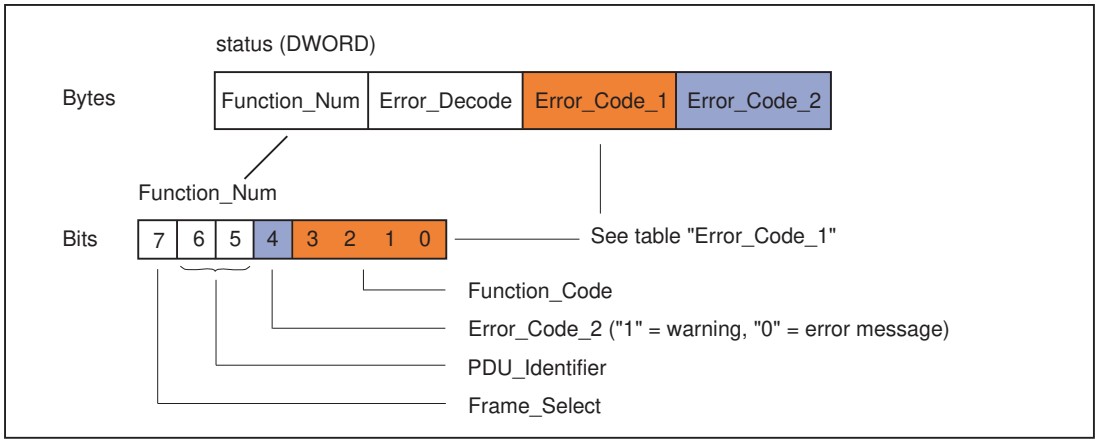


Figure 8-78 Output parameter 'status'

These bytes are described in detail below.

Note

See section "Examples for reading out error numbers" to learn how to read out error messages and warnings from the **status** parameter.

Function_Num

The **Function_Num** byte of the **status** output parameter consists of 8 bits. They are defined as follows:

Bit 0 to 3	Function_Code / Error_Code
Bit 4	Error_Code_2 (error/warning)
Bit 5 to 6	PDU_Identifier
Bit 7	Frame_Select

The following table contains the values that can be assumed by the **Function_Num** byte.

Table 8-160 Function_Num

Frame_Select (Bit 7)	PDU_Identifier (Bits 6 to 5)	Error_Code_2 (Bit 4)	Function_Code / Error_Code (Bits 3 to 0)	Description according to profile
0	0..3	0/1	0..15	No error
1	0, 1	0/1	0..15	Error does not relate to the PROFIBUS DP protocol and is not defined for this profile.
1	2	0/1	0..15	Error message relating to PROFIBUS DP protocol
1	3	0/1	0	Vendor-specific code of Error_Code_1 and Error_Code_2
1	3	0/1	1	Error_Code_1 provides error information relating to the transponder Vendor-specific code of Error_Code_2
1	3	0/1	2	Error_Code_1 provides error information relating to the radio interface Vendor-specific code of Error_Code_2
1	3	0/1	3	Error_Code_1 provides error information relating to the file system Vendor-specific code of Error_Code_2
1	3	0/1	4	Error_Code_1 provides error information relating to the Ident Unit (interrogator/barcode reader) Vendor-specific code of Error_Code_2
1	3	0/1	5	Error_Code_1 provides error information relating to communication between FB and Ident Unit (except for PROFIBUS DP errors) Vendor-specific code of Error_Code_2

Frame_Select (Bit 7)	PDU_Identifier (Bits 6 to 5)	Error_Code_2 (Bit 4)	Function_Code / Error_Code (Bits 3 to 0)	Description according to profile
1	3	0/1	6	Error_Code_1 provides command-specific error information Vendor-specific code of Error_Code_2
1	3	0/1	7	Error_Code_1 provides error information generated internally by the FB
1	3	0/1	8..15	Not defined

Error_Decode

The **Error_Decode** byte contains the meaning of the bytes **Function_Num**, **Error_Code_1**, and **Error_Code_2**.

The following table contains the values that can be assumed by the **Error_Decode** byte.

Table 8-161 Error_Decode

Error_Decode	Source	Meaning
16#00	Control	Not a warning, not an error
16#01 to 16#7F	Control	Warning (not used for this profile)
16#80	DPV1	Error report in accordance with IEC 61158-6
16#81 to 16#8F	Control	Error report matching the nth parameter of the call of the communication FB. The exact definition of the communication FB is provided in the following documentation: PROFIBUS Communication and Proxy Function Blocks acc. to IEC 61131-3, Version 1.2, July 2001
16#90 to 16#FD	-	Reserved
16#FE	Profile (FB, Ident Unit)	Profile-specific error
16#FF	Profile (FB, Ident Unit)	Reserved for future use

Error_Code_1

The **Error_Code_1** byte supplies a number that assigns the error message or warning. In the following table, the value "16#FE" is specified for the **Error_Decode** byte. Bits 0 to 3 correspond to bits 0 to 3 of the **Function_Num** byte.

Table 8-162 Error_Code_1

Function_Code / Error_Code	Error_Code_1 (decimal)	Indexed by	Meaning
Errors and warnings relating to the transponder			
1	1	Ident Unit	Memory error of the transponder
1	2	Ident Unit	Presence error - transponder is no longer within the transmission window

Function_Code / Error_Code	Error_Code_1 (decimal)	Indexed by	Meaning
1	3	Ident Unit	Address or command does not match the transponder characteristics (memory capacity).
1	4	Ident Unit	Transponder defective Transponder or batteries must be replaced.
1	5	Ident Unit	Transponder capacity is exceeded.
1	6	Ident Unit	Unformatted transponder
1	7	Ident Unit	Incompatible data structure of transponder The transponder must be reformatted.
1	8	Ident Unit	The transponder in the transmission window does not have the expected UID.
1	9	Ident Unit	Command is not supported by the transponder
1	10	Ident Unit	Access error (e.g. block locked), see ISO 18000-x
1	11...127	Ident Unit	Reserved for future profile use
1	128...255	Ident Unit	Vendor-specific
Errors and warnings relating to the radio interface			
2	1	Ident Unit	Time limit exceeded for communication via radio interface
2	2	Ident Unit	There are more transponders in the transmission window than is permitted.
2	3...127	Ident Unit	Reserved for future profile use
2	128...255	Ident Unit	Vendor-specific
Errors and warnings relating to the file system			
3	1	Ident Unit	Error in file name
3	2	Ident Unit	File not available
3	3	Ident Unit	The type of transponder is incorrect or inappropriate for the selected operating mode. A file system cannot be accessed on the transponder.
3	4	Ident Unit	Command is being created No further directory entries are available.
3	5	Ident Unit	Command is being created The directory already contains this file.
3	6	Ident Unit	Error relating to access rights
3	7	Ident Unit	File length exceeded
3	8	Ident Unit	File cannot be accessed or is damaged
3	9...127	Ident Unit	Reserved for future profile use
3	128...255	Ident Unit	Vendor-specific
Errors and warnings relating to the Ident Unit			
4	1	Ident Unit	Power supply failure
4	2	Ident Unit	Hardware fault in the Ident Unit
4	3	Ident Unit	Antenna not operating; e.g., switched off or disconnected from system
4	4	Ident Unit	Capacity of command buffer in the Ident Unit is exceeded.
4	5	Ident Unit	Capacity of data buffer in the Ident Unit is exceeded.

Function_Code / Error_Code	Error_Code_1 (decimal)	Indexed by	Meaning
4	6	Ident Unit	Command is not supported by the Ident Unit in this mode.
4	7	Ident Unit	Ident Unit is returning a non-specific error message that was indexed by the cyclic status word (e.g. antenna does not work, etc.). This error message does not relate to a specific command.
4	8...127	Ident Unit	Reserved for future profile use
4	128...255	Ident Unit	Vendor-specific
Errors and warning relating to the communication between the FB and the Ident Unit			
5	1	Ident Unit	Incorrect sequence number (SN)
5	2	FB	Incorrect sequence number (SN)
5	4	Ident Unit	Invalid data block number (DBN)
5	5	FB	Invalid DBN
5	6	Ident Unit	Invalid data block length (DBL)
5	7	FB	Invalid DBL
5	8	Ident Unit	Command of another user is being processed
5	9	FB	The Ident Unit is performing a hardware reset (Init_Active is set to "1"), and the **init** input parameter (bit 15 in cyclic control word) is expected by the FB.
5	10	FB	The "CMD" command signal and the relevant acknowledgement do not match. This is a software or synchronization error that does not occur during normal operation.
5	11	FB	Incorrect sequence of acknowledgement message frame (TDB/DBN)
5	12	FB	Synchronization error (increment of AC_H/AC_L and CC_H/CC_L in cyclic control word is incorrect). The **init** input parameter of the FB is required for execution.
5	13...127	Ident Unit	Reserved for future profile use
5	128...255	Ident Unit	Vendor-specific
Command-specific errors and warnings			
6	1	Ident Unit	Invalid "CMD" command signal
6	2	Ident Unit	Invalid command index (CI)
6	3	Ident Unit	Invalid command parameter (e.g. data area)
6	4	Ident Unit	Object detection error Error in synchronization between user program and transponder An expected command is missing.
6	5	Ident Unit	In this state, only the "Write config" command is permitted.
6	6	FB	Starting time expired
6	7...127	Ident Unit	Reserved for future profile use
6	128...255	FB	Vendor-specific
Error information generated internally by the FB			
7	1	FB	In this state, only the **init** parameter of the FB is permitted.
7	2	FB	"CMD" command signal is not permitted

Function_Code / Error_Code	Error_Code_1 (decimal)	Indexed by	Meaning
7	3	FB	The **Length** parameter of the command is too long for the global data reserved in RXBUF.
7	4	FB	Capacity of RXBUF exceeded (more data were received than memory available in the RXBUF).
7	5	FB	Only the **init** parameter of the FB is permitted as the next command. All other commands are rejected.
7	6	FB	Wrong index (outside of permissible range of 101 to 108)
7	7	FB	Ident Unit is not responding to the **init** input parameter (Init_Active is expected in the cyclic status communication).
7	8	FB	Time exceeded during **init**
7	9...127	FB	Reserved for future profile use
7	128...255	FB	Vendor-specific

Error_Code_2

If bit 4 of the **Function_Num** byte is set to "1", then the **Error_Code_2** byte contains more information.

The figure below shows the coding of warnings. Here, the value "16#FE" is specified for the **Error_Decode** byte, and the source of the warnings is always the Ident Unit.

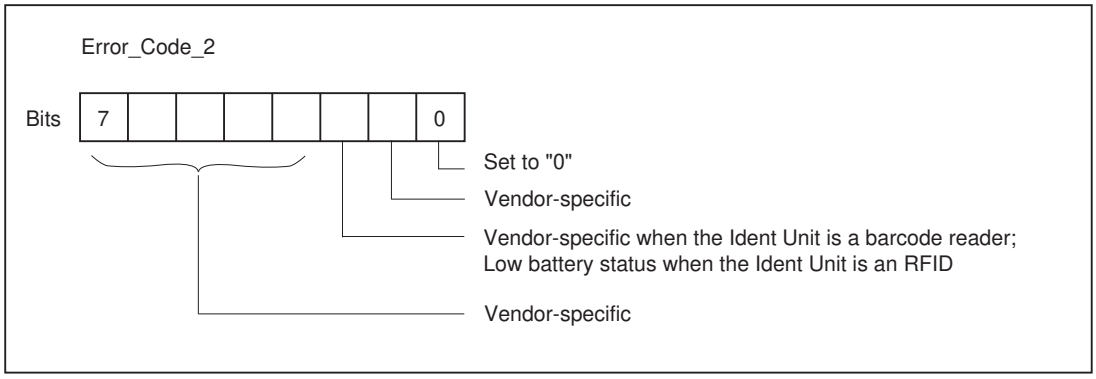


Figure 8-79 Error_Code_2

If the warning bit is set to "0", the **Error_Code_2** byte can contain vendor-specific information.

The warnings are displayed by the acyclic acknowledgement message frame in byte 5.

Error in acyclic and cyclic data transfer

In addition to the error messages supplied by the Ident Unit and the function blocks, the following error messages and warnings are specified by the SIMOTION System in the **status** output parameter:

Table 8-163 Error messages and warnings during acyclic data transfer: Writing data

Error number	Meaning
16#C0845700	Error: Data length
16#C0835700	Error: Index value
16#D080D1xx	Warning: Data transfer aborted by user: "ABORT_CURRENT_COMMAND"
16#C080D3xx	Error: too many job repetitions
16#C080D5xx	Error: indexed by return value of **_writeRecord** system function
16#C080D7xx	Error: unknown return value of**_writeRecord** system function

xx: least significant byte of return value of **_writeRecord** system function

Table 8-164 Error messages and warnings during acyclic data transfer: Reading data

Error number	Meaning
16#C0845200	Error: Data length
16#C0835200	Error: Index value
16#D080D0xx	Warning: Data transfer aborted by user: "ABORT_CURRENT_COMMAND"
16#C080D2xx	Error: too many job repetitions
16#C080D4xx	Error: indexed by return value of **_readRecord** system function
16#C080D6xx	Error: unknown return value of **_readRecord** system function

xx: least significant byte of return value of **_readRecord** system function

Note

For a description of the system functions **_writeRecord** and **_readRecord**, see

SIMOTION C2xx / P350 / D4xx System Function/Variables Parameter Manual.

These documents are included in the SIMOTION SCOUT scope of delivery as electronic documentation.

Table 8-165 Error messages during cyclic data transfer Reading data

Error number	Meaning
16#DF80B200	Invalid slot (**id**), or I/O variables for the RFID device do not exist or are invalid (data type: BYTE)
16#DF80B600	I/O access failed
16#DF80B700	Invalid area (**offset**)

Table 8-166 Error messages during cyclic data transfer Writing data

Error number	Meaning
16#DE80B200	Invalid slot (**id**), or I/O variables for the RFID device do not exist or are invalid (data type: BYTE)
16#DE80B600	I/O access failed
16#DE80B700	Invalid area (**offset**)

Examples of reading out error numbers

The following examples show you how to read out the meaning of the warnings and error messages from an error number of the **status** output parameter.

Example 1

The **error** output parameter is set to **TRUE** by the function block **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB**. The error number **16#E3FE0700** is displayed in the **status** output parameter.

This error number can be broken down as follows:

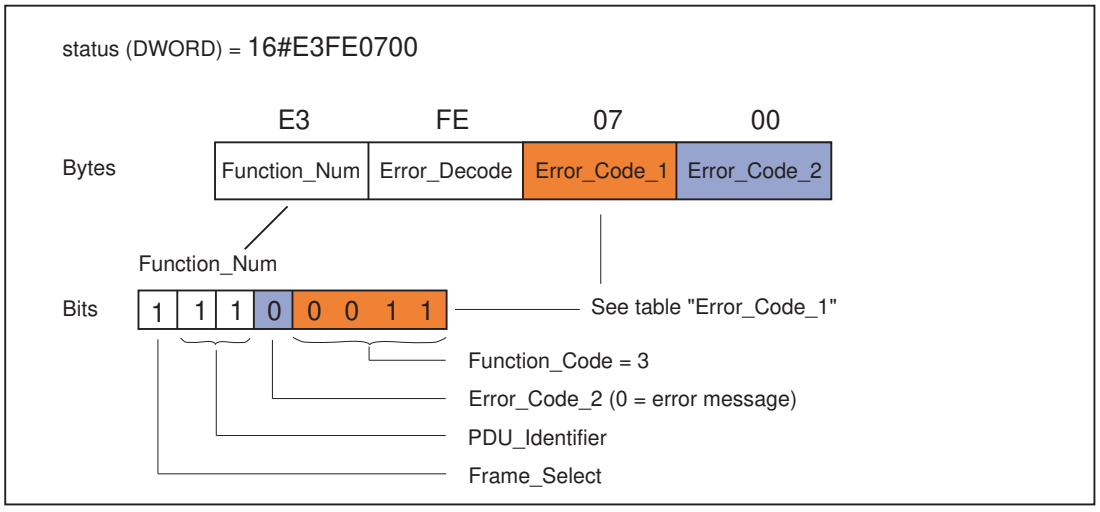


Figure 8-80 Error message: Example 1

From the **Function_Num** byte, **Bit 4** = 0 indicates that an error message is present. From the table "Function_Num", you receive the following meaning for **Function_Code** = 3:

Frame_Select (bit 7)	PDU_Identifier (bits 6 to 5)	Error_Code_2 (bit 4)	Function_Code / Error_Code (Bit 3 to 0)	Description according to profile
1	3	0/1	**3**	Error_Code_1 provides error information relating to the file system; Vendor-specific code of Error_Code_2

You can find the meaning for **Error_Code_1** = 7 in the table "Error_Code_1":

Function_Code / Error_Code	Error_Code_1 (decimal)	Indexed by	Meaning
3	**7**	Ident Unit	File length exceeded

For error number **status** = 16#E3FE0700, this results in the error message that the previous command has attempted to transfer too large a file.

Example 2

The **error** output parameter is set to **TRUE** by the function block **_PIB_001KB** / **_PIB_016KB** / **_PIB_032KB**. The error number **16#C080D5A1** is displayed in the **status** output parameter.

This error number can be broken down as follows:

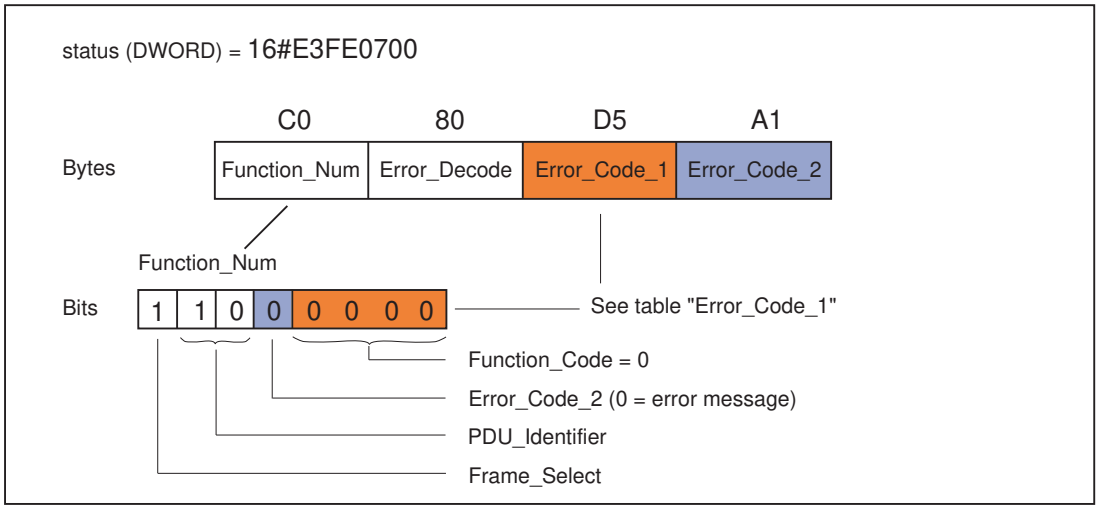


Figure 8-81 Error message: Example 2

From the **Function_Num** byte, **Bit 4** = 0 indicates that an error message is present. From the table "Function_Num", you receive the following meaning for **Function_Code** = 0:

Frame_Select (bit 7)	PDU_Identifier (bits 6 to 5)	Error_Code_2 (bit 4)	Function_Code / Error_Code (Bit 3 to 0)	Description according to profile
1	2	0/1	**0**	Error message relating to PROFIBUS DP protocol

Errors of this category are not profile-specific errors and are therefore addressed in section "Errors during acyclic and cyclic data transfer". In table "Error messages and warnings during acyclic data transfer: writing data", you will find the meaning of error number **1616#C080D5A1**:

Error code	Meaning
16#C080D5xx	Error: indexed by return value of **_writeRecord** system function

xx: least significant byte of return value of **_writeRecord** system function

In the *SIMOTION C230-2 System Functions/Variables* list manual, you will find the following entry under **_writeRecord** :

16#FFFF80A1 Error during data set transfer, job aborted. Negative acknowledgment when writing to module:
- Module removed during write operation
- Module defective

For error number **status** = 16#C080D5A1, this results in the error message that the error is module-specific and the module may have to be replaced.

8.12.5 Application example

8.12.5.1 General information on the application example

Introduction

The "E_PIB032" application example shows you how to exchange data between the SIMOTION system and the RFID system ASM 456 using the **_PIB_001KB**, **_PIB_016KB**, and **_PIB_032KB** function blocks. The following commands are provided in the application example for data exchange:

- Initialize the Ident Unit (Write config)
- Format the transponder (Format)
- Create a new file on a formatted transponder (Create)
- Write data to a transponder file (Write)
- Read out the transponder directory (Get directory)

Requirements

The application example is available for the RFID system ASM 456 (type: MOBY I-Filehandler) and a transponder for a maximum data amount of 32 KB.

The following requirements must be met:

- You have installed the GSD file in your language. You need the GSD file in order to insert the ASM 456 into the device catalog where you can select it in **HW Config**.
- You need a transponder for a maximum data amount of 32 KB.
- Vendor-specific data are needed in the example for the "Write config" and "Format" commands. The vendor-specific data can be found in the user documentation for the RFID system ASM 456.

Hardware platform

The application example is available for various SIMOTION hardware platforms.

Note

If the application example is not available for your hardware platform, you must adapt the hardware configuration.

Calling the application example

The application example can be found on the "SIMOTION Utilities & Applications" CD-ROM. The "SIMOTION Utilities & Applications" CD-ROM is provided free of charge with SIMOTION SCOUT.

1. Dearchive and open the project containing the application example.
2. Check the axis configuration: PROFIBUS DP addresses.
3. **Save** and **compile** the example project. Then, you can download the example to the SIMOTION device and switch to RUN mode.

Adapting the application example

The configuration in the example and its available hardware must be adapted.

The following options are available:

1. You can adapt the configuration of the example to the available hardware.
2. You can adapt the configuration of the hardware to the example.

Adapting the configuration of the example to the available hardware

You have assigned a module address <> 256 in **HW Config**.

1. Open the **DP Slave Properties** dialog box in **HW Config** by double-clicking the ASM456 module. Then select the **Parameter Assignment** tab.
2. In the **Parameter Assignment** tab, open the **Station parameters** folder and the **Device-specific parameters** subfolder. Set the device-specific parameters as follows:
   - USER-Mode = RFID standard profile.
   - MOBY-Mode = MOBY I-Filehandler

   Acknowledge the entry with **OK**.

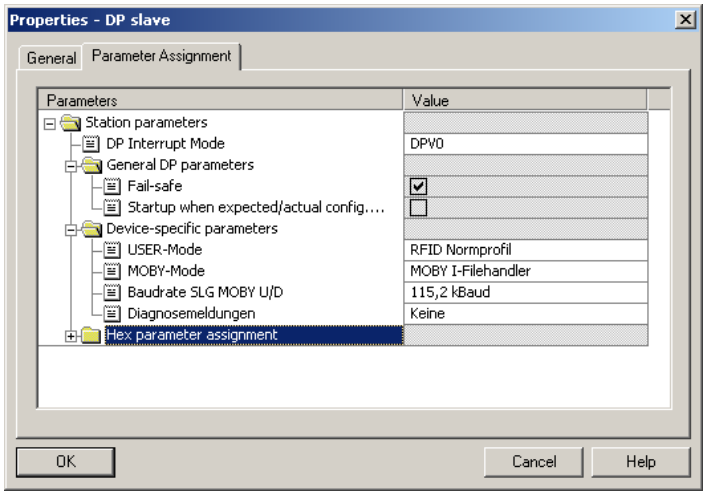


Figure 8-82 Setting device-specific parameters

3. Set the start address of the input and output address area to the value that you have assigned in **HW Config** (e.g. 256).
4. This results in the following addresses for the ZSW/STW, for example:
   - pibfhzswin : PIB 256 : Array 4 ("PIB" = Peripheral Input: Byte)
   - pibfhstwout : PQB 256 : Array 4 ("PQB" = Peripheral Output: Byte)

   Create these I/O addresses in the Symbol Browser.

In the application example, you need vendor-specific data for the "Write config" and "Format" commands. You obtain the following settings from the user documentation of the ASM 456, which you must adapt in the application example.

- Vendor-specific data for the "Write config" command:

```
ASM456_I_FH_WC_LEN : INT := 14;
ASM456_I_FH_WC_DAT : ARRAY [1..ASM456_I_FH_WC_LEN] OF BYTE
                   := [4,0,0,0,0,8,16#42,16#34,16#30,16#FF,16#FF,0,0,1];
```

- Vendor-specific data for the "Format" command:

```
ASM456_I_FH_FM_LEN      : INT := 21;
ASM456_I_FH_FM_DAT_8    : ARRAY [1..ASM456_I_FH_FM_LEN] OF BYTE
                        := [8(0),16#0C,4(16#41,16#31),05,01,16#4C,0];
ASM456_I_FH_FM_DAT_32   : ARRAY [1..ASM456_I_FH_FM_LEN] OF BYTE
                        := [8(0),16#0C,4(16#41,16#31),06,01,16#4C,0];
```

Note

When another transponder is used (e.g. for maximum data amount of 1 KB), you must adapt the following in the application example:

- Adapt the **myMdsSize** parameter as well as the CONFIG and FORMAT arrays to your transponder as appropriate.
- Set the **myFbPIB** parameter: **_PIB_001KB**.
- Adapt the array length of in/out parameters **txref** and **rxref** to the utilized FB as appropriate.

8.12.5.2 Sequence of the application example

Command processing

Proceed as follows to initiate the command processing in the example program:

1. Open the Symbol Browser in SIMOTION SCOUT.
2. To select a command, set the **myCommand** parameter in the Symbol Browser to the command to be executed.

   Note

   You must always set the **myCommand** parameter to the **cmdInit** command to initialize the Ident Unit at the beginning of command processing.

3. Start the selected command by setting the **myExecuteCmd** parameter to **TRUE**.
4. Wait for the indication that the command processing was completed without errors, i.e. **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE** at the output parameters of the utilized FB. An error or a warning during command processing is indicated with **error** = **TRUE** or **warning** = **TRUE**.
5. Once the command has been processed successfully, you can start additional commands.

Example: Initializing Ident Unit

Vendor-specific configuration data of the ASM456 must be transferred to the transmit data buffer (TXBUF) of the Ident Unit for initialization of the Ident Unit. The "Write config" command initializes the Ident Unit and must always be first in the command buffer.

In the example, the "Init" command (cmdInit) is implemented such that all relevant parameters are set to the required values in Step 1. Proceed as follows to execute the "Init" command (as well as the implicitly associated "Write-Config" command:

1. In the Symbol Browser, set parameter **myCommand** = **cmdInit**.
2. Start the command with **myExecuteCmd** = **TRUE**.
3. Wait for the indication that the command processing was completed without errors, i.e. **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE** at the output parameters of the utilized FB.
   Once the command has been processed successfully, you can start additional commands.

Example: Formatting the transponder

The "Format" command initializes the transponder. After formatting, the transponder is ready for use. Vendor-specific data of the ASM456 must be transferred to the transmit data buffer (TXBUF) of the Ident Unit for definition of the transponder name. Proceed as follows to execute the "Format" command:

1. In the Symbol Browser, set parameter **myCommand** = **cmdFormat**.
2. Start the command with **myExecuteCmd** = **TRUE**.
3. Error-free command execution is indicated in the utilized FB with **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE**. The transponder is formatted and ready for use in the application.

Example: Creating a (read-only) file on the formatted transponder

The "Create" command is used to create a new file on the formatted transponder. This file is read-only. Proceed as follows to execute the "Create" command:

1. In the Symbol Browser, set parameter **myCommand** = **cmdCreate**.
2. Start the command with **myExecuteCmd** = **TRUE**.
3. Error-free command execution is indicated in the utilized FB with **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE**. The file has been created on the transponder.

Example: Writing data to a transponder file

You can use the "Write" command to write data to a file that you previously created on the transponder. Proceed as follows to execute the "Write" command:

1. In the Symbol Browser, set parameter **myCommand** = **cmdWrite**.
2. Assign the data that are to be written to the transponder file to the **FileName** command parameter.
3. Start the command with **myExecuteCmd** = **TRUE**.
4. Error-free command execution is indicated in the utilized FB with **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE**. The data have been transferred to the transponder file.

Example: Reading out the transponder directory

You can use the "Get directory" command to read out the directory of the transponder. File names and associated attributes are passed by the transponder. Proceed as follows to execute the "Get directory" command:

1. In the Symbol Browser, set parameter **myCommand** = **cmdGetDirectory**.
2. Start the command with **myExecuteCmd** = **TRUE**.
3. Error-free command execution is indicated in the utilized FB with **done** = **TRUE**, **error** = **TRUE** and **warning** = **TRUE**.
   The read out data are displayed in the output parameters of the "Get directory" command (see parameter description in "Get Directory").

8.12.6 Appendix

8.12.6.1 List of abbreviations

Table 8-167 Abbreviations

Abbreviation	Meaning
ASM	Interface modules
CI	Command index
DBL	Data block length
DBN	Data block number
DP	Distributed I/O
DPV1	Short term for expanded DP functions in PROFIBUS-DP (IEC 61158)
FB	Function block
GSD	Device data file
IN	Input parameters
I&M	Identification and maintenance functions
IN/OUT	In/out parameter
MOBY	Mobile identification system
OUT	Output parameters
PIB	Proxy Ident Block
PNO	PROFIBUS User Organization
RFID	Radio Frequency Identification
RXBUF	Receive data buffer indicated by the **rxref** input parameter of the FB
SN	Sequence number
ST	Structured text
STW	Control word
TDB	Total number of data blocks
TXBUF	Transmit data buffer indicated by the **txref** input parameter of the FB
UID	Unique Identifier (identification of a transponder)
ZSW	Status word

8.13 PLCopen Blocks

Preface

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

8.13.1 Fundamental safety instructions

8.13.1.1 General safety instructions

⚠ WARNING

Risk of death if the safety instructions and remaining risks are not carefully observed

If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

8.13.1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

8.13.2 Introduction

The PLCopen blocks are designed for use in cyclic programs/tasks and enable motion control programming in a PLC environment. If preferred, they can be used in the LAD/FBD programming language.

PLCopen blocks are available as standard functions (directly from the command library).

The motion control functions meet the PLCopen specifications in terms of interfaces, functions and sequence, and are certified according to "PLCopen Compliance Procedure for Motion Control Library V1.1".

References

More detailed information on technology objects and the effect of PLCopen block input parameters on these TOs can be found in the following Function Manuals:

- TO Axis Electric / Hydraulic, External Encoder
- Technology Objects Synchronous Operation, Cam

8.13.3 Description

8.13.3.1 Description of PLCopen blocks

In SIMOTION, the following list of blocks, which are certified according to "PLCopen Compliance Procedure for Motion Control Library V1.1", can be used in cyclic programs/tasks.

If preferred, they can be used in the LAD/FBD programming language. PLCopen blocks are available as standard functions (directly from the command library).

Table 8-168 SingleAxis functions for the axis

Function	Description
_MC_Power()	Enabling/disabling axis
_MC_Stop()	Stopping the axis
_MC_Home()	Homing axis/clearing absolute value encoder offset
_MC_MoveAbsolute()	Absolutely positioning axis
_MC_MoveRelative()	Relatively positioning axis
_MC_MoveVelocity()	Traversing axis at defined velocity
_MC_MoveAdditive()	Positioning relative to current target position (traversing axis using an additional, defined path, relative to current position setpoint)
_MC_MoveSuperimposed()	Superimposed positioning (traversing axis relative to current motion)
_MC_PositionProfile()	Traveling through position/time profile (traversing axis along a predefined, fixed position/time profile)
_MC_VelocityProfile()	Traveling through velocity/time profile (traversing axis along a predefined, fixed velocity/time profile)
Basic functions	
_MC_Reset()	Resetting errors/alarms on the axis or triggering a restart
_MC_ReadActualPosition()	Reading the actual position of axis
_MC_ReadStatus()	Reading the status of an axis
_MC_ReadAxisError()	Reading the error of an axis
_MC_ReadParameter()	Reading axis parameter and outputting in data type LREAL
_MC_ReadBoolParameter()	Reading axis parameter and outputting in data type BOOL
_MC_WriteParameter()	Writing axis parameter of data type LREAL
_MC_WriteBoolParameter()	Writing axis parameter of data type BOOL

Function	Description
Apart from the standard PLCopen functions, the following additional standard axis function is included:	
_MC_Jog()	Continuous or incremental jogging

Table 8-169 MultiAxis functions for the axis

Function	Description
_MC_GearIn()	Starting gearing (synchronizing master and slave axis while taking into account a positional relationship described by a fixed gear ratio)
_MC_GearOut()	Terminating gearing (desynchronizing master and slave axis)
_MC_CamIn()	Starting camming (synchronizing master and slave axis while taking into account a positional relationship described by a cam)
_MC_CamOut()	Terminating camming (desynchronizing master and slave axis)
_MC_Phasing()	Changing phase shift between the leading axis and following axis

Table 8-170 Functions for external encoder

Function	Description
_MC_Power()	Enabling external encoder
_MC_Reset()	Resetting external encoder
_MC_Home()	Homing external encoder
_MC_ReadActualPosition()	Reading actual position of external encoder
_MC_ReadStatus()	Reading external encoder status
_MC_ReadAxisError()	Reading external encoder error
_MC_ReadParameter()	Reading external encoder parameter and outputting in data type LREAL
_MC_ReadBoolParameter()	Reading external encoder parameter and outputting in data type BOOL

Other blocks

On the "Utilities & Applications" CD (included in SIMOTION SCOUT scope of delivery) you will also find a versatile block that can be used for axis control purposes (under Applications > Converting > Versatile Block for Axis Control).

The **FBLineAxis** function block serves a wide range of application cases and is, therefore, classified as versatile. It enables the user to not only control individual axis motions, but also

define axis groups using master/slave relationships, e.g. as a setpoint cascade with closed-loop position-controlled axes.

Note

No liability assumed when using FBLineAxis

Siemens AG assumes no liability for any problems or errors which may arise from using FBLineAxis.

8.13.3.2 General rules for the PLCopen FB interface

Description

The following applies in accordance with the PLCopen specification "Technical Specification, PLCopen - Technical Committee 2 – Task Force, Function blocks for motion control, Version 1.1":

Exclusiveness of output parameters

The output parameters "Busy", "Done", "Error" and "CommandAborted" are mutually exclusive: At each FB, only one of these output parameters can be set to TRUE. If "Execute" is TRUE, one of these output parameters must also be TRUE.

Only one of the output parameters "Active", "Error", "Done" and "CommandAborted" is set.

Status of the output parameters

The output parameters "Done", "InGear", "InSync", "InVelocity", "Error", "ErrorID" and "CommandAborted" are reset with the falling edge of "Execute". The falling edge of "Execute" stops, but does not however influence the execution of the actual FB. The corresponding output parameters are set for a minimum of one cycle if this situation occurs, even if "Execute" has been reset prior to completing the execution of the FB.

If an FB instance receives a new "Execute" parameter prior to the execution being completed (as a series of commands for the same instance), the FB does not output any feedback for the previous action such as "Done" or "CommandAborted".

Input parameters

The parameters are used with the rising edge of the "Execute" input parameter. For all parameters to be modified, it is essential that the input parameter(s) are changed and the processing/traversing is re-triggered.

Missing input parameters

In accordance with IEC 61131-3, if any parameters of a function block input are missing (open), the value from the previous call of this instance is used. The initialization value is used during the first call.

"Position" in comparison with "Distance"

"Position" is a value that is defined within a coordinate system. "Distance" is a relative value that relates to technical units. "Distance" is the difference between two positions.

Sign rules

The "Velocity", "Acceleration", "Deceleration" and "Jerk" parameters always have positive values. "Position" and "Distance" can be both positive and negative.

Error processing response

All blocks contain two output parameters, which process errors that may occur during the execution of the respective function block. These output parameters are defined as follows:

- "Error"
  The rising edge of "Error" indicates that an error occurred during the execution of a function block.
- "ErrorID"
  Error code

"Done", "InVelocity", "InGear" and "InSync" indicate that the execution has been completed successfully, these signals and "Error" are therefore logically mutually exclusive.

Error types:

- Function blocks (e.g. parameters outside the range, fault of the sequence control)
- Communication
- Drive

Instance errors do not always lead to an axis error (which brings the axis to a "standstill").

The error outputs of the respective FB are reset with the falling edge of "Execute".

Response of the "Done" output parameter

The "Done" output parameter (as well as the "InGear", "InSync", output parameters, among others) are set if the required action has been completed successfully.

The following applies for several function blocks, which work in a sequence on the same axis:

If a motion on an axis is interrupted by another motion on the same axis, without the final target being reached, the "Done" output parameter of the first FB is not set.

Response of the "CommandAborted" output parameter

"CommandAborted" is activated if a requested traversing is interrupted by another traversing command.

The Reset response of "CommandAborted" corresponds to that of "Done". If "CommandAborted" occurs, other output signals are reset such as "InVelocity".

Input parameters that exceed application limits

1. The parameter violates the parameter-related absolute upper/lower limits (the limits in place on the related limit data):
   If an FB is controlled with parameters that violate the application limits, the instance of the FB generates an error. The consequences of this error for the axis are application-specific and should therefore be processed by the application program.
2. The parameter is located in these upper/lower limits, but not in the "application limit", i.e. it exceeds the TO limit values:
   depending on the input parameter and the general "SIMOTION" rules.
   As a general rule, only the parameter is limited; these limits lead to an alarm/warning, resulting in a stop response -> the FB is then also switched to the error state with ErrorID: Abort by pending error response.

8.13.4 Blocks

8.13.4.1 Handling PLCopen blocks

Block instances

In order to use PLCopen blocks, you must create one block instance for each being used.

The data type of the instance corresponds to the block name.

Example

Below you can see how a block instance is structured, using the _MC_MoveAbsolute block in an ST program by way of example. Optional parameters are converted to comments.

Table 8-171 Block instance for _MC_MoveAbsolute

```
VAR
myinst_moveAbsolute : _mc_moveabsolute;
END_VAR

myinst_moveAbsolute(
axis := Hour
// ,execute := 0
// ,position := 0.0
// ,velocity := -1.0
// ,acceleration := -1.0
// ,deceleration := -1.0
// ,jerk := -1.0
// ,direction := USER_DEFAULT
// ,done =>
```

```
// ,busy =>
// ,active =>
// ,commandaborted =>
// ,error =>
// ,errorid =>
);
```

Upgrading a PLCopen library from V3.2 to V4.2

Up until SIMOTION SCOUT V3.2 SP1, the PLCopen blocks formed part of the SIMOTION Function Library (provided on a separate CD).

With SIMOTION SCOUT V4.0 and higher, the PLCopen blocks are included in the SIMOTION SCOUT command library as standard functions ("PLCopen" command group). As part of this, they have been modified and considerably extended (to include PLCopen multiaxis functions, for example).

Note

Incompatibility errors during compilation

After upgrading a PLCopen library from V3.2 to V4.2, an LAD/FBD incompatibility error occurs if you have integrated the new library using a namespace. The operations with the parameters can no longer be compiled without errors.

As an example, the namespace in this case should be called "L_SAxis".
During compilation, the relevant PLCopen blocks are downloaded to the namespace using the USELIB L_SAxis AS ns_L_SAxis operation. Therefore, all of the corresponding data types are also in this namespace. Versions of SIMOTION SCOUT up to V4.0 were fault-tolerant in the event of a faulty operation occurring without a namespace having been specified (e.g. Direction := POSITIVE_DIRECTION). However, SIMOTION SCOUT V4.1 and higher is no longer fault-tolerant and produces a compilation error.

Manual intervention is required when this occurs: **all** incomplete operations must be completed using the namespace extension "ns_L_SAxis".
From V4.1, the correct operation for the example above must be as follows:
Direction := ns_L_SAxis.POSITIVE_DIRECTION or
Direction := ns_L_SAxis.mc_direction#POSITIVE_DIRECTION

8.13.4.2 SingleAxis

_MC_Power - Enabling/disabling axis

Overview

Schematic diagram

Schematic diagram

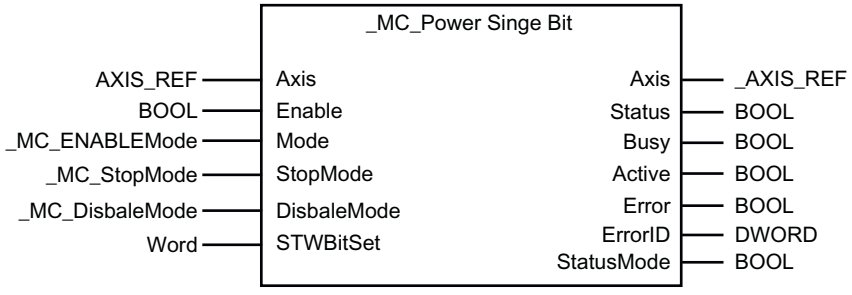


Figure 8-83 _MC_Power Schematic diagram

Purpose

Purpose

The **_MC_Power** function block enables or disables an Axis technology object or an External Encoder technology object.

If an active braking is possible before an axis is disabled, it is stopped with the set stop mode. The parameter **StopMode** specifies a stop mode for the axis. The stop mode is taken over with a falling edge at the **Enable** input.

_MC_Power with individual bits (as of V4.2)

The bits of the PROFIdrive protocol can be set and reset with _enableAxis and _disableAxis. This functionality is also enabled in the **_MC_Power** function block. For this purpose, it is extended by the STWBitSet input parameter. The STWBitSet parameter is only evaluated if the value BY_STW_BIT is transferred in parameter mode.

In the STWBitSet parameter, the values of Bit 0 - Bit 6 that are set in the PROFIdrive protocol are specified. In the case of enable=TRUE, the bits specified in the control word are set, whereas in the case of enable=FALSE, the specified bits are reset.

See also AUTOHOTSPOT.

Applicable for

Applications

Drive axes

Positioning axes

Following axes

Path axes

External Encoders

Note

This block can only enable or disable electrical standard axes. The block does not function for axes with force or pressure control.

Requirements

Requirements

No alarms preventing the enable may be present on the TO.

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The axis is enabled with a rising edge at this input. If this is not possible, the attempt to set the enables is repeated as long as **Enable** is set. The axis is stopped with a falling edge at this input. The axis is disabled after reaching standstill.
Mode	_MC_EnableMode	ALL	Specification of the axis enables to be set Not effective for external encoders. ALL: Set all enables and deactivate follow-up mode DRIVE: Only set drive enable BY_STW_BIT setting of the bits specified in the STWBitSet parameter in the PROFIdrive protocol

Parameter	Data type	Initial value	Description
StopMode	_MC_StopMode	WITH_COMMAND_VALUE_ZERO	Specification of the stop mode Not effective for external encoders. WITH_COMMAND_VALUE_ZERO: The axis is stopped with emergency stop in the STOP_WITH_COMMAND_VALUE_ZERO mode. It is stopped via the emergency stop ramp in the controller. The ramp is set during axis configuration. WITH_MAXIMAL_DECELERATION: The axis is stopped with emergency stop in the STOP_WITH_MAXIMAL_DECELERATION mode. It is stopped according to interpolation with the maximum dynamic values of the axis. IN_DEFINED_TIME: The axis is stopped with emergency stop in the STOP_IN_DEFINED_TIME mode. The default setting for the stop time defined in the system variable userDefaultDynamics.stopTime is used. The specified time is adhered to irrespective of the starting velocity. DISABLE_DRIVE_IMMEDIATELY: The POWER enable is removed directly from the axis. The drive coasts to a standstill.
DisableMode	_MC_DisableMode	ALL	Operating mode of axis ALL; Remove all enables and Follow-up operation on CONTROL; Remove axis enable in the control and set axis in follow-up mode, the drive remains active, it is not switched off; BY_STW_BIT; reset of the bits specified in the STWBit-Set parameter in the PROFIdrive protocol
STWBitSet	WORD	0	Bit0 - ON Bit1 - No coast stop (no OFF2) Bit2 - No quick stop (no OFF3) Bit3 - Enable operation Bit4 - Enable ramp generator Bit5 - Unfreeze ramp generator Bit6 - Enable setpoint See also AUTOHOTSPOT.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Status	BOOL	FALSE	Display of the enable status of the axis With TRUE, the axis is activated and motion commands can be executed. With FALSE, the axis is not (completely) activated, no motion commands can be executed on the axis.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Section ErrorIDs (Page 6611)).
StatusMode	Bool	FALSE	State of the actual enables in comparison with the setpoint enables; indicates that the enables specified in 'mode' or 'DisableMode' have been reached.

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The axis is traversed with velocity 50 after the enable. After an error has occurred resulting in the removal of the axis enable, the error is corrected with **_MC_Reset**. The axis is enabled again. The axis is then traversed again with velocity 50 and the enable is removed. The axis brakes with the ramp set in the **StopMode** parameter.

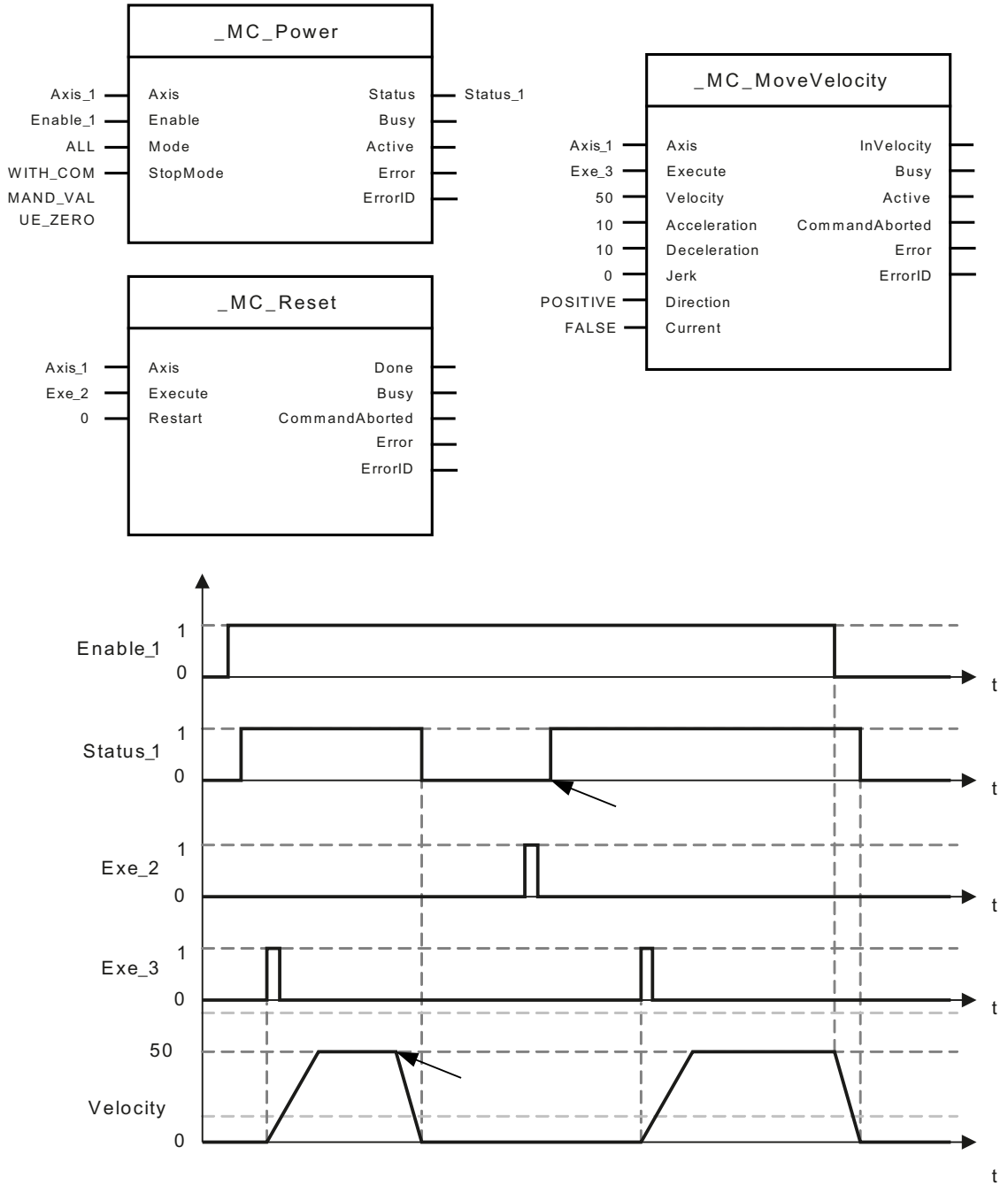


Figure 8-84 _MC_Power Example

_MC_Stop - Stopping axis

Overview

Schematic diagram

Schematic diagram

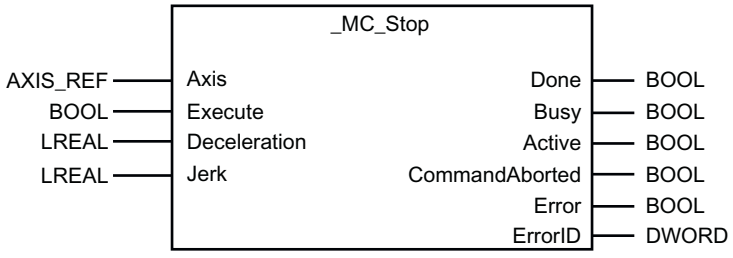


Figure 8-85 _MC_Stop Schematic diagram

Purpose

Purpose

The function block **_MC_Stop** terminates all active motion commands on an axis and decelerates it down to standstill. The function block can be overrided or aborted by another motion command, i.e. it is not possible to start a motion command.

The function block is terminated (**Done** equals **TRUE**) when the axis is stationary and the input **Execute** is reset to **FALSE**. It is then possible again to start a motion command on the axis.

The input parameters **Deceleration** and **Jerk** define the dynamic response of the stop procedure.

Applicable for

Applications

Drive axes

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled and not in follow-up mode

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The axis stops with a rising edge at this input.

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

⚠ WARNING

If the function block **_MC_Stop** is started with a jerk specification not equal to zero during the acceleration phase of an axis, its velocity can increase to the configured maximum velocity of the drive in an extreme situation. The axis is only decelerated once acceleration has been reduced by the jerk.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block The axis is at standstill and the disable for the motion commands has been removed (**Execute** equals **FALSE**). This output is only set for one cycle.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.

Parameter	Data type	Initial value	Description
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6616)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The axis is started with velocity 50 and then decelerated to velocity 0 with **_MC_Stop**. A further call of **_MC_MoveVelocity** results in an error as the **Execute** input is set on the **_MC_Stop**. After the **Execute** input on the **_MC_Stop** is reset, the **_MC_MoveVelocity** can be executed again.

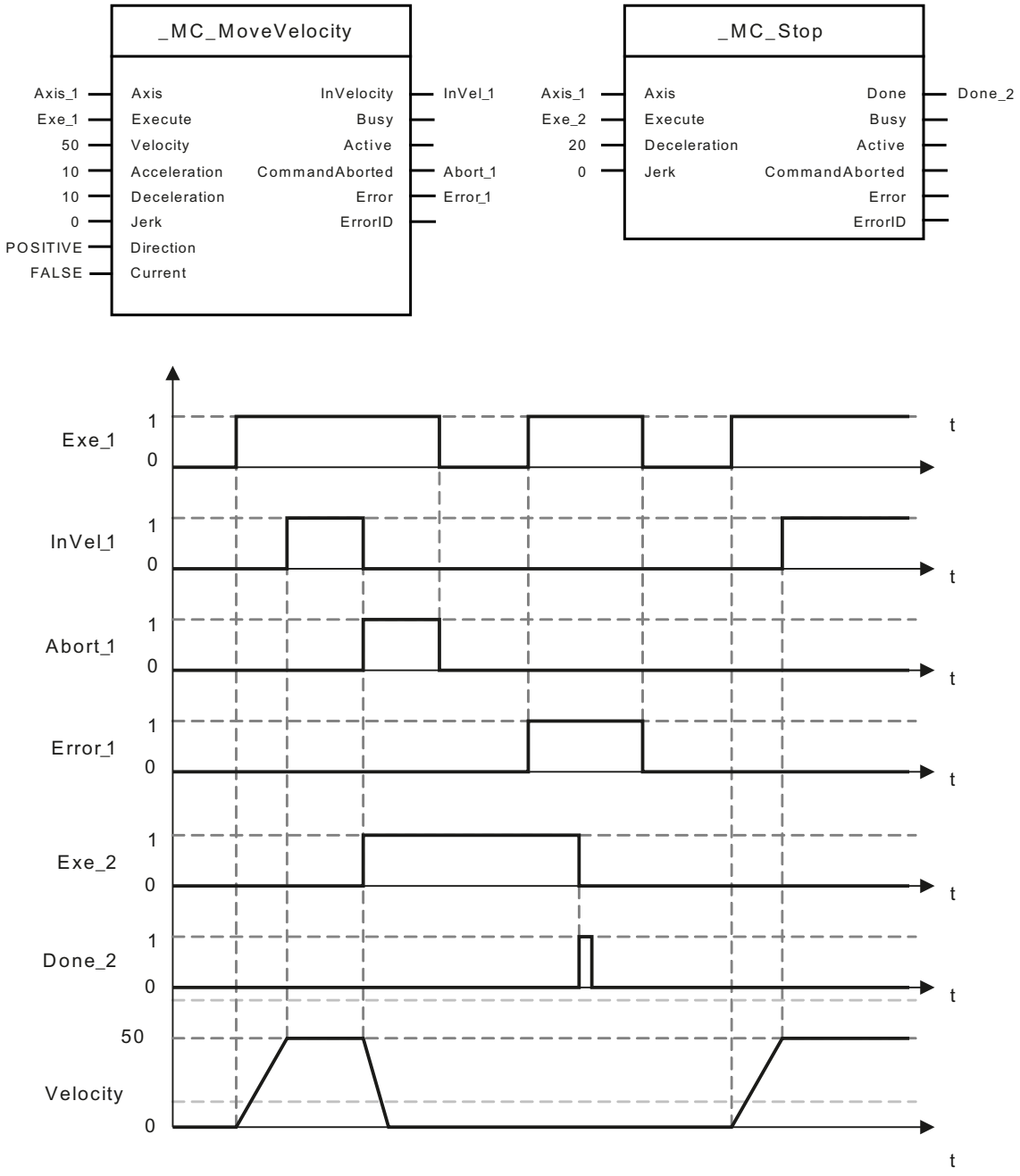


Figure 8-86 _MC_Stop Example

_MC_Home - Homing axis/clearing absolute value encoder offset

Overview

Schematic diagram

Schematic diagram

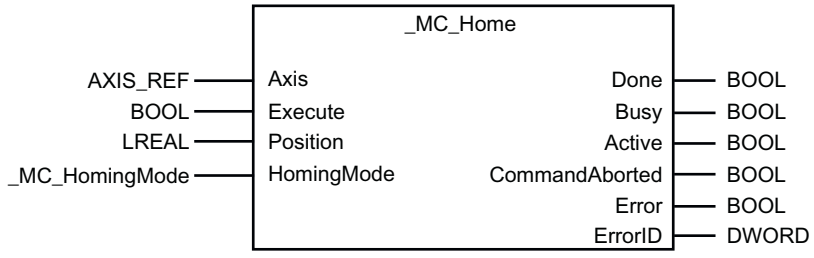


Figure 8-87 _MC_Home Schematic diagram

Purpose

Purpose

The function block **_MC_Home** establishes a positional relationship between the control and the mechanical system via a measuring system.

The following are supported:

- Active and passive homing of an axis
  The homing mode is defined at the function block. Parameters for the other data are set in the "Homing" axis dialog.
- Setting a position value
- Relative correction/shift of the actual value
- Absolute encoder adjustment

Applicable for

Applications

Positioning axes

Following axes

Path axes

External encoders

Restrictions

The parameter **HomingMode** of the function block **_MC_Home** only defines the homing mode. The homing procedure itself is performed in accordance with the configuration of the encoder on the axis.

Homing mode	Virtual axis	Real axis with incremental encoder	Real axis with absolute encoder	External Encoders
ACTIVE_HOMING		X [1)]		
PASSIVE_HOMING		X		X
DIRECT_HOMING	X	X	X	X
DIRECT_HOMING_ RELATIVE	X	X	X	X
ENABLE_OFFSET_OF_ ABSOLUTE_ENCODER			X [2)]	X [2)]

1) If homing mode **MODE_NO_REFERENCE** is set in the configuration of the **TypeOfAxis.NumberOfEncoders.Encoder_<n>.IncHomingEncoder.HomingMode** encoder, then the homing mode **ACTIVE_HOMING** is not possible at the function block.

2) In homing mode **ENABLE_OFFSET_OF_ABSOLUTE_ENCODER**, the value transferred with input parameter **Position** is not effective.
For this mode you must enter the required offset in the encoder configuration before calling the **_MC_Home** function block. Enter the offset value in the configuration data under **TypeOfAxis.NumberOfEncoders.Encoder_<n>.absHomingEncoder.absShift** and select **absolute** or **relative**.
If ENABLE_OFFSET_OF_ABSOLUTE_ENCODER (**absolute**) is selected, the axis is set to the specified offset value. If DIRECT_HOMING_RELATIVE (**relative**)is selected, the specified offset value is added to the current axis position and the axis is set to this "total" value.

Requirements

Requirements

Enabling axes or external encoders

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoder-Type data type)
Execute	BOOL	FALSE	Function block enable The homing procedure starts with a rising edge at this input.
Position	LREAL	0.0	Specification of the position at the reference point or the position setting value or the position offset value
HomingMode	_MC_HomingMode	ACTIVE_HOMING	Specification of the homing mode: ACTIVE_HOMING: Active homing PASSIVE_HOMING: On-the-fly homing DIRECT_HOMING: Setting current position as reference point DIRECT_HOMING_RELATIVE: Direct homing (the actual position of the axis is added to the value specified in the parameter **Position** as position difference). ENABLE_OFFSET_OF_ABSOLUTE_ENCODER: Absolute encoder adjustment

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the programmed target position has been reached.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6621)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Examples

Example: Active homing

Sequence for active homing:

- Phase 1:
  Approach of the home position switch (BERO).
  The axis traverses with the homing approach velocity V_{app} (approach velocity).
- Phase 2:
  Synchronization with the zero mark.
  The axis traverses with the homing reduced velocity V_{red} (reduced velocity).
- Phase 3:
  Travel to the home position coordinate.
  The axis traverses with the homing entry velocity V_{ent} (entry velocity).

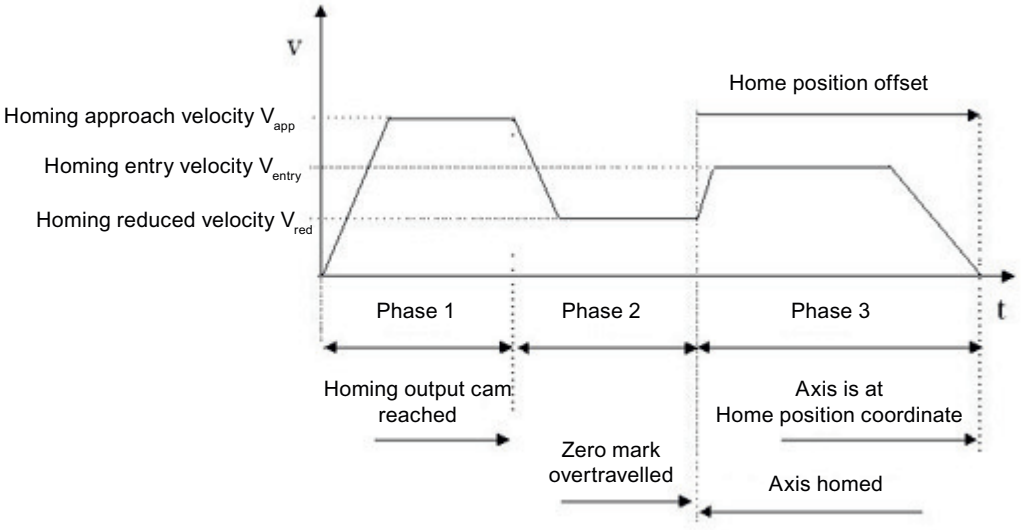


Figure 8-88 _MC_Home Example: Active homing

Example: Direct homing

The new absolute position is set in the next interpolator cycle after the call of the function block with the set Execute input. When calling the function block from a task which is not synchronous with the interpolator, the setting procedure is only recommended for axes at standstill.

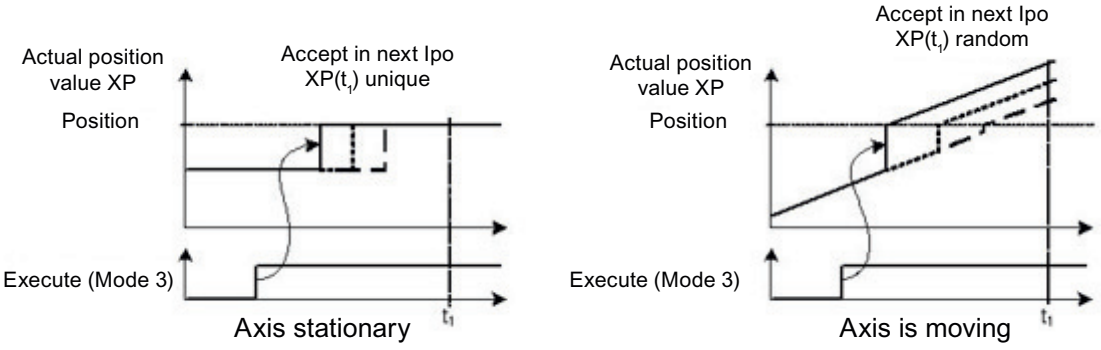


Figure 8-89 _MC_Home Example: Direct homing

Example: Passive homing

The axis is traversed at the velocity 50. Then the passive homing with zero mark is triggered. At the next zero pulse, the actual position of the axis is set to 90.

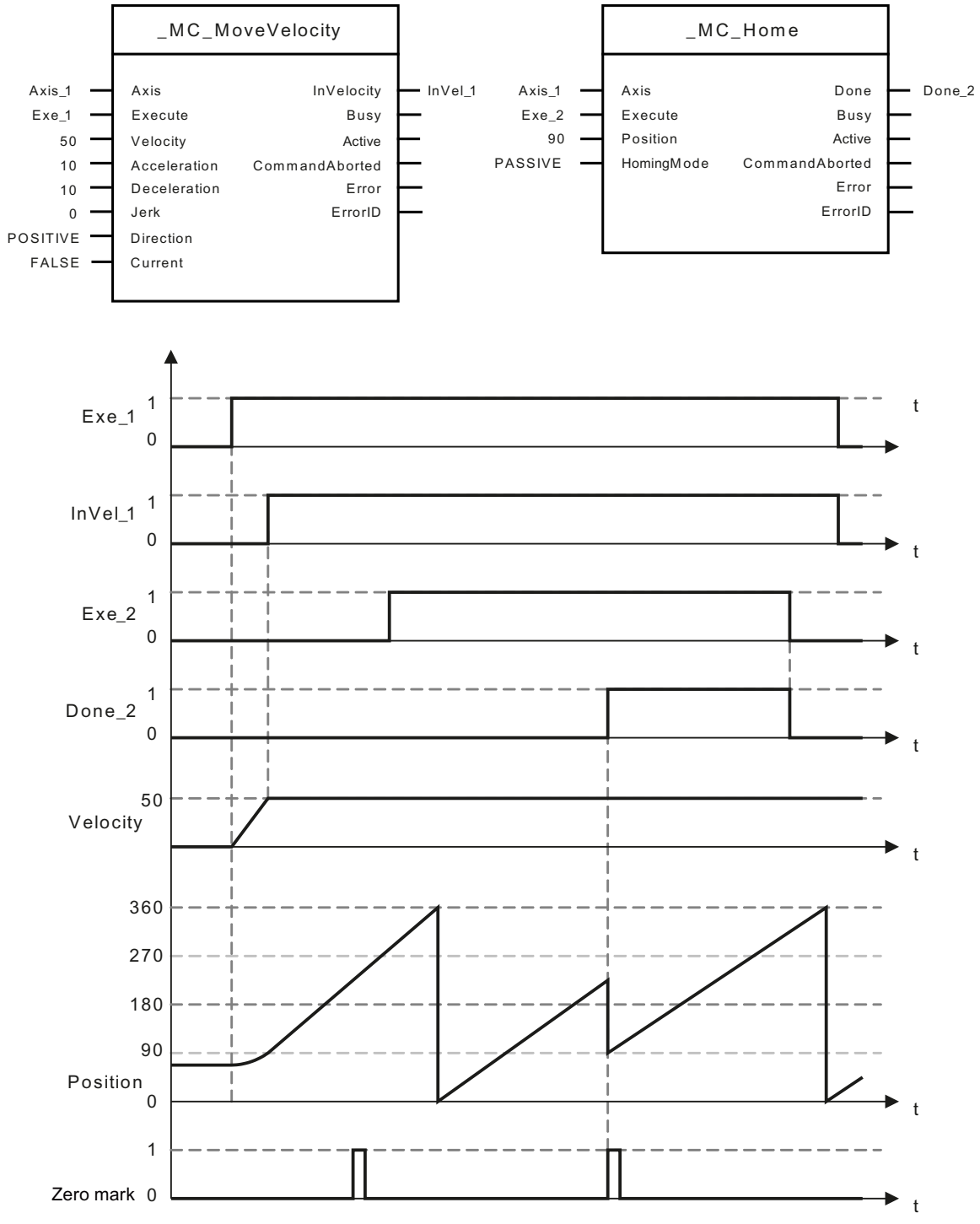


Figure 8-90 _MC_Home Example: Passive homing

_MC_MoveAbsolute - Absolute positioning of axis

Overview

Schematic diagram

Schematic diagram

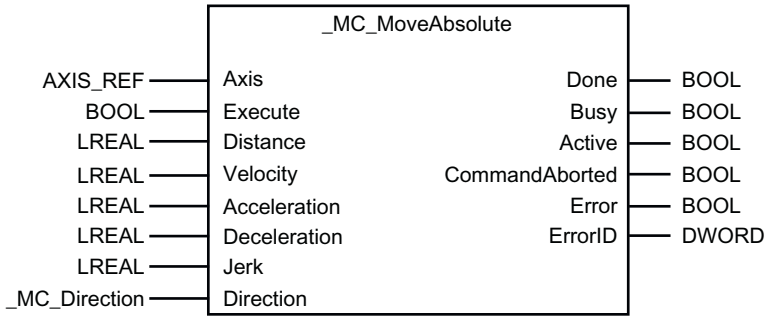


Figure 8-91 _MC_MoveAbsolute Schematic diagram

Purpose

Purpose

The function block **_MC_MoveAbsolute** starts a positioning motion of an axis to an absolute position.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

The axis stops after completion of the positioning motion.

An active motion command is overridden by the function block.

With modulo axes, **SHORTEST_WAY** can be specified as an additional direction of motion.

Note

The state of the "DONE" output depends on the system variable motionstatedata.motionstate.

Applicable for

Applications

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

Axis homed if the axis configuration data item **TypeOfAxis.Homing.referencingNecessary** was set to **YES** (homing required).

No **_MC_Stop** active

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The positioning operation starts with a rising edge at this input.
Position	LREAL	0.0	Specification of the absolute target position of the motion

Parameter	Data type	Initial value	Description
Velocity	LREAL	-1.0	Specification of the maximum velocity The velocity is reached depending on the set values for traversing distance, acceleration and jerk. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.
Direction	_MC_Direction	USER_DEFAULT	Specification of direction of motion for modulo axes: USER_DEFAULT: Default value from axis configuration POSITIVE: Direction of rotation/motion positive SHORTEST_WAY: In direction of shortest distance [1)] NEGATIVE: Direction of rotation/motion negative EFFECTIVE: Last programmed direction of rotation/motion

[1)] For modulo axes only

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the programmed target position has been reached.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6629)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

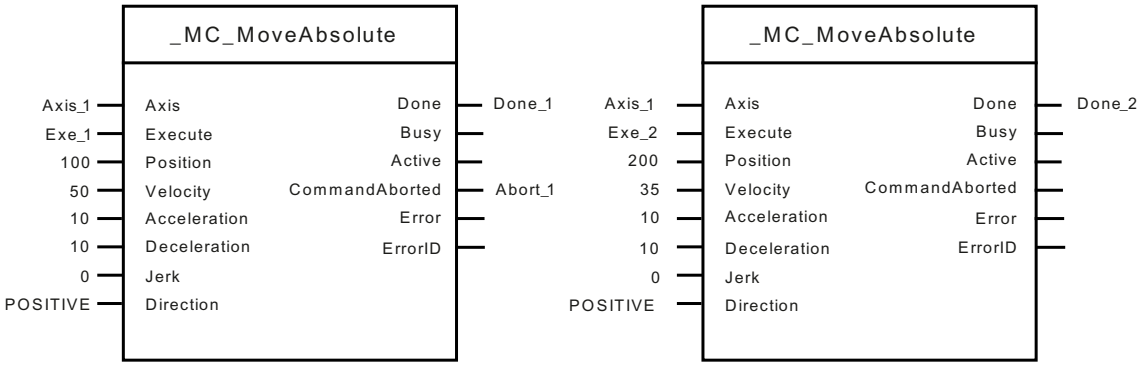
Example

Case A:

Two _MC_MoveAbsolute blocks are started in succession.

Case B:

The second _MC_MoveAbsolute cancels the first _MC_MoveAbsolute block. The target position results relatively from the position at the start of the second block.



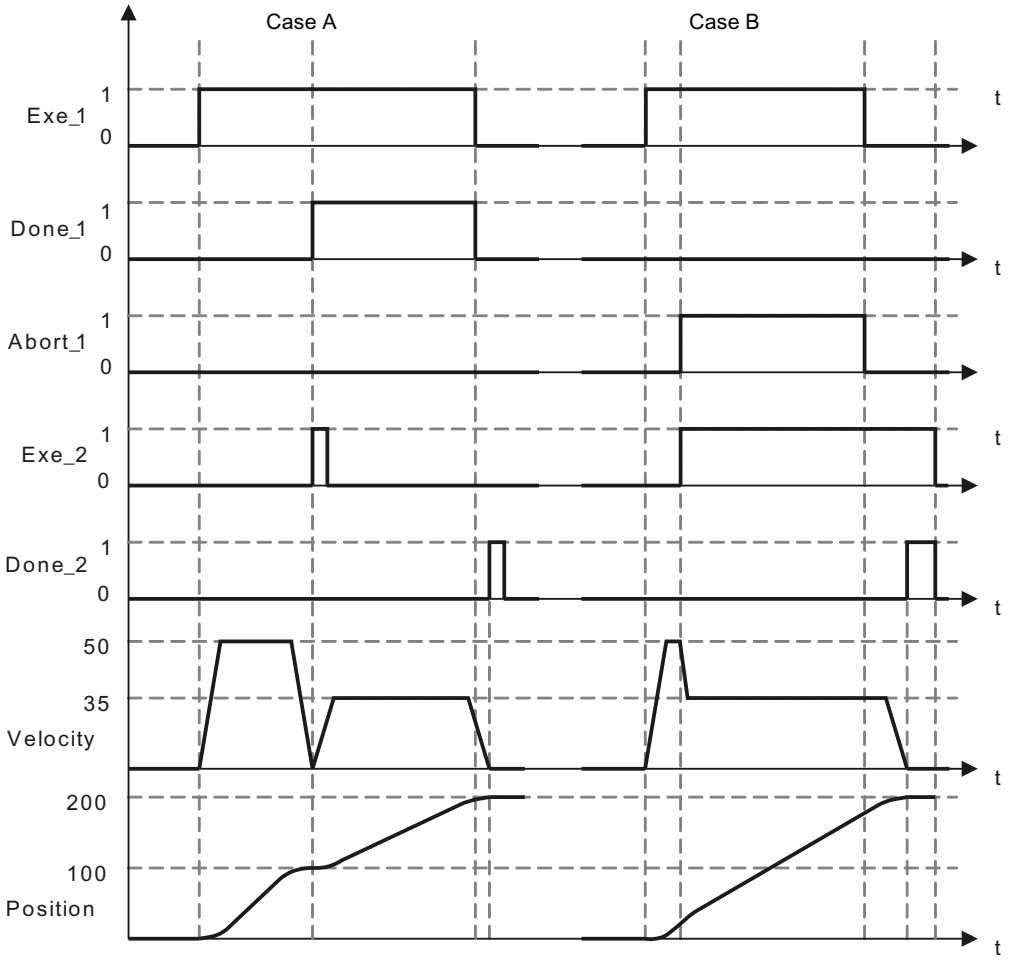


Figure 8-92 _MC_MoveAbsolute Example

_MC_MoveRelative - Relative positioning of axis

Overview

Schematic diagram

Schematic diagram

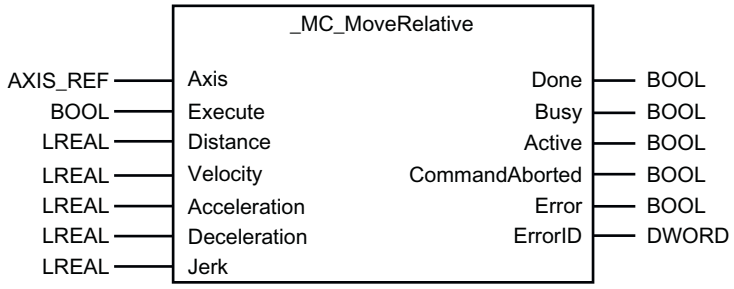


Figure 8-93 _MC_MoveRelative Schematic diagram

Purpose

Purpose

The function block **_MC_MoveRelative** positions an axis relative to the actual position of the axis. If the axis is already in motion when the job is started, the position that is present in the system at the start of the job processing is used as the start position. It must be taken into account that there is a response time between the processing of the function block and the execution of the motion, which depends on the user task in which the function block was programmed and on the set interpolation cycle clock.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

The axis stops after completion of the positioning motion.

An active motion command is overridden by the function block.

Note

The state of the "DONE" output depends on the system variable motionstatedata.motionstate.

Applicable for

Application

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

No **_MC_Stop** active

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The positioning operation starts with a rising edge at this input.
Distance	LREAL	0.0	Specification of the distance difference to be traversed
Velocity	LREAL	-1.0	Specification of the maximum velocity The velocity is reached depending on the set values for traversing distance, acceleration and jerk. Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameters	Data type	Initial value	Description
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the programmed target position has been reached.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.

Parameter	Data type	Initial value	Description
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6634)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

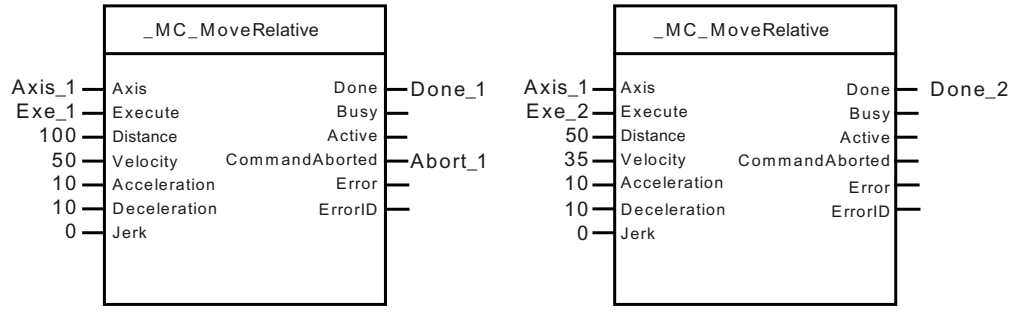
Example

Case A:

Two _MC_MoveRelative blocks are started in succession.

Case B:

The second _MC_MoveRelative cancels the first _MC_MoveRelative block. The target position results relatively from the position at the start of the second block.



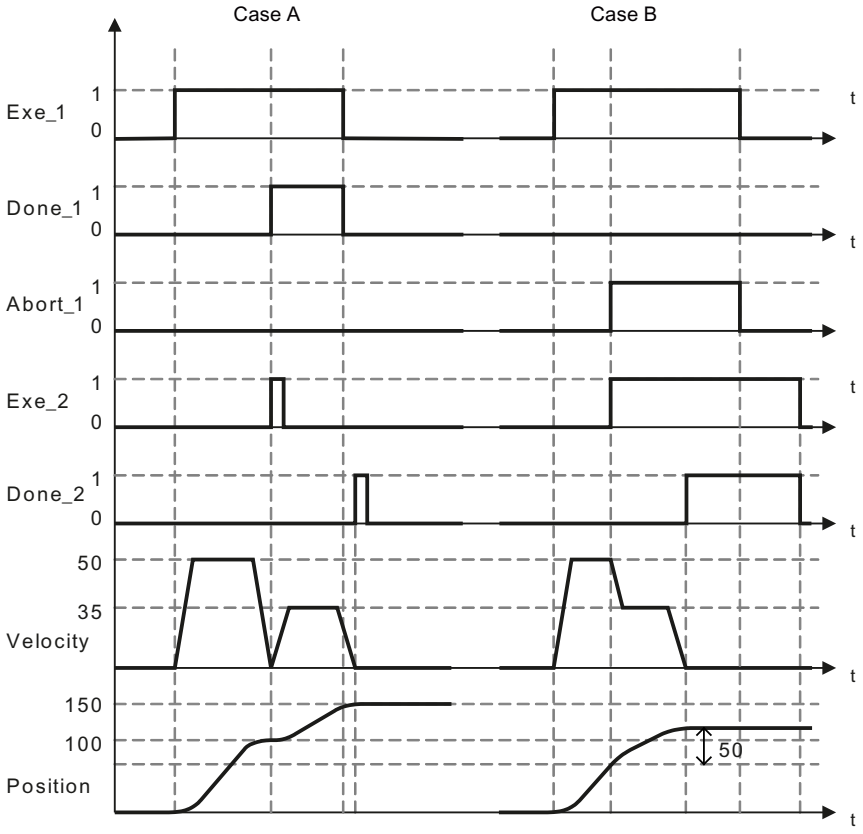


Figure 8-94 _MC_MoveRelative Example

_MC_MoveVelocity - Traversing axis at a specified velocity

Overview

Schematic diagram

Purpose

Applicable for

Requirements

Input parameters

Output parameters

ErrorIDs

Example

Schematic diagram

Schematic diagram

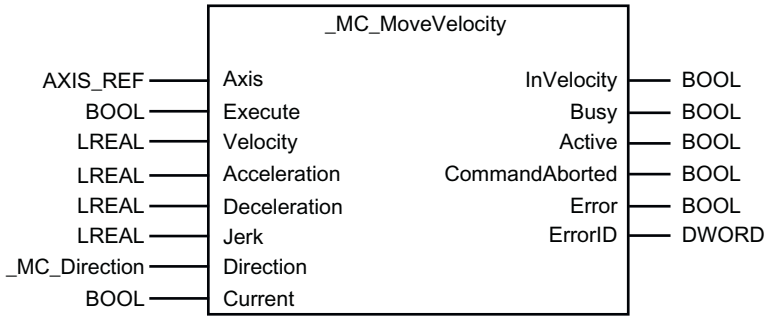


Figure 8-95 _MC_MoveAdditive Schematic diagram

Purpose

Purpose

The technology function **_MC_MoveVelocity** accelerates or decelerates an axis to a set velocity.

The dynamic response parameters **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

If a velocity override is in effect, then the end velocity is calculated under consideration of the override. You must take this behavior into account in the user program.

Note

The state of the "DONE" output depends on the system variable motionstatedata.motionstate.

Note

With **_MC_MoveVelocity**, the axis is traversed in position-controlled mode instead of speed-controlled mode

Applicable for

Applications

Drive axes

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

No **_MC_Stop** active

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The axis accelerates or decelerates to the programmed set velocity with a rising edge at this input.
Velocity	LREAL	-1.0	Specification of the set velocity Value ≥ 0: The specified value is used. Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.
Direction	_MC_Direction	USER_DEFAULT	Specification of the direction of motion: USER_DEFAULT: Default value from axis configuration POSITIVE: Direction of rotation/motion positive NEGATIVE: Direction of rotation/motion negative EFFECTIVE: Last programmed direction of rotation/motion
Current	BOOL	FALSE	Type of velocity specification With TRUE, the actual velocity of the axis is taken over as programmed set velocity. With FALSE, the set velocity programmed on the **Velocity** input is used.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
InVelocity	BOOL	FALSE	Indicates termination of the function block. With TRUE, the axis has reached the programmed setpoint velocity. Until the abort of the function block or until the input Execute = FALSE, the output remains unchanged irrespective of the subsequent characteristic of the axis velocity.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6639)).

Note

When speed setpoint = zero (Velocity = 0.0), the following applies:

InVelocity is set when the axis has reached a standstill, and remains set as long as **Execute = 1**.

As soon as **InVelocity** is set, the command is terminated, i.e. **Busy = FALSE**. This means that neither **CommandAborted** nor **Error** can be signaled by the function block.

Note

The behavior of the outputs depends on the task in which the block is processed. The display is only refreshed when the task, in which the block is to be processed, is executed. If the block is called in a non-cyclic task (e.g. MotionTask), then the BUSY parameter may be set to TRUE, but the ACTIVE/INVELOCITY parameters may still display FALSE, because the block, e.g. on the axis, may not be effective yet. Only when the MotionTask is executed does the display change from ACTIVE to TRUE. In a cyclic task such as the BackgroundTask, the blocks are processed cyclically and therefore the BUSY and ACTIVE parameters both display TRUE.

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

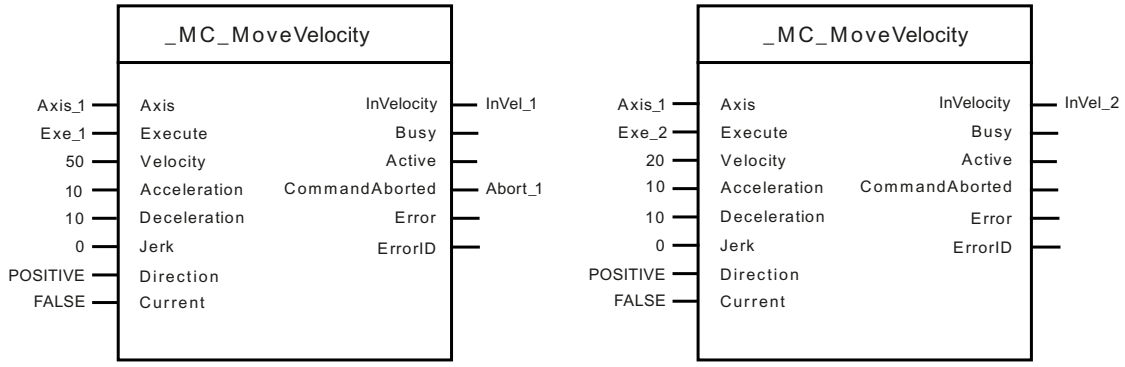
Example

Case A:

An _MC_MoveVelocity block is overridden by another _MC_MoveVelocity block after the end velocity has been reached (**InVelocity = TRUE**).

Case B:

An _MC_MoveVelocity block is overridden by another _MC_MoveVelocity block before the end velocity has been reached (**InVelocity = FALSE**).



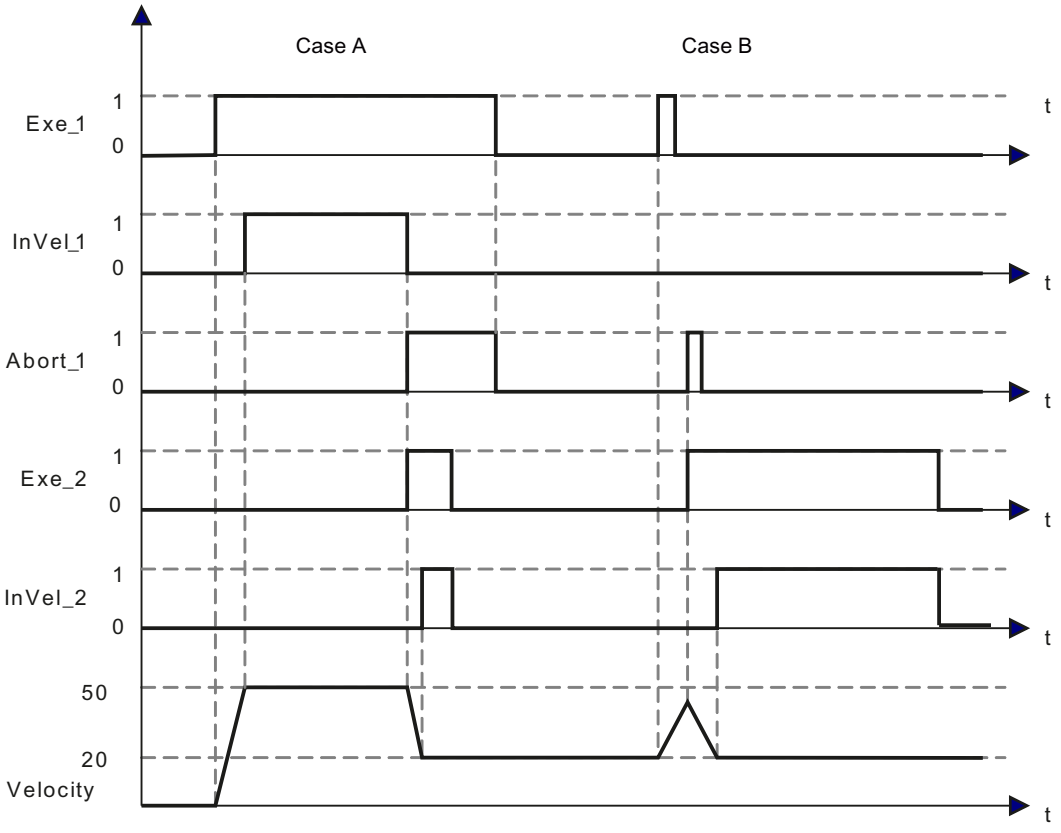


Figure 8-96 _MC_MoveVelocity Example

_MC_MoveAdditive - Relative positioning to current target position

Overview

Schematic diagram

Schematic diagram

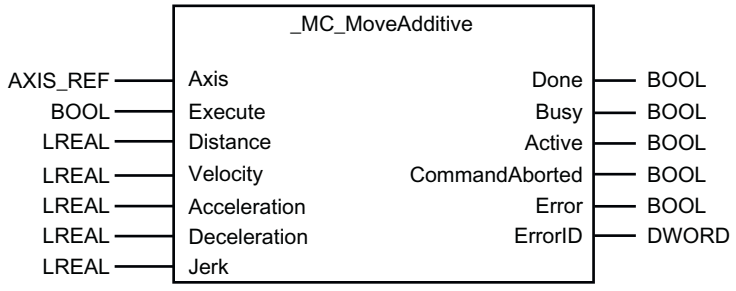


Figure 8-97 _MC_MoveAdditive Schematic diagram

Purpose

Purpose

The function block **_MC_MoveAdditive** positions an axis relative to the target position of the active positioning command. The function block enables a correction of the target position of the previous positioning command by a distance specified at the Distance input.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

The axis stops after completion of the positioning motion.

An active motion command is overridden by the function block.

Note

The state of the "DONE" output depends on the system variable motionstatedata.motionstate.

Applicable for

Applications

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

No **_MC_Stop** active

The axis must be homed if the TypeOfAxis.Homing.referencingNecessary axis configuration data item has been set to YES (homing required), and

- the axis is in motion, or
- **_MC_MoveAdditive** overrides an active motion command (exception: **_MC_MoveVelocity**).

Note

The function block **_MC_MoveAdditive** behaves like a **_MC_MoveRelative** function block, if

- The axis is stationary at the start of the job or
- An active motion command without defined target position is overridden by the function block. The target position then depends on the position of the axis at the time of the override and the additional distance to be traversed.

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The positioning operation starts with a rising edge at this input.
Distance	LREAL	0.0	Specification of the additional distance difference to be traversed
Velocity	LREAL	-1.0	Specification of the maximum velocity The velocity is reached depending on the set values for traversing distance, acceleration and jerk. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameters	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the axis is at the resulting position setpoint.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6646)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

Case A:

Two _MC_MoveAdditive blocks are started in succession.

Case B:

The second _MC_MoveAdditive cancels the first _MC_MoveAdditive block. The target position results from the target position of the first block.

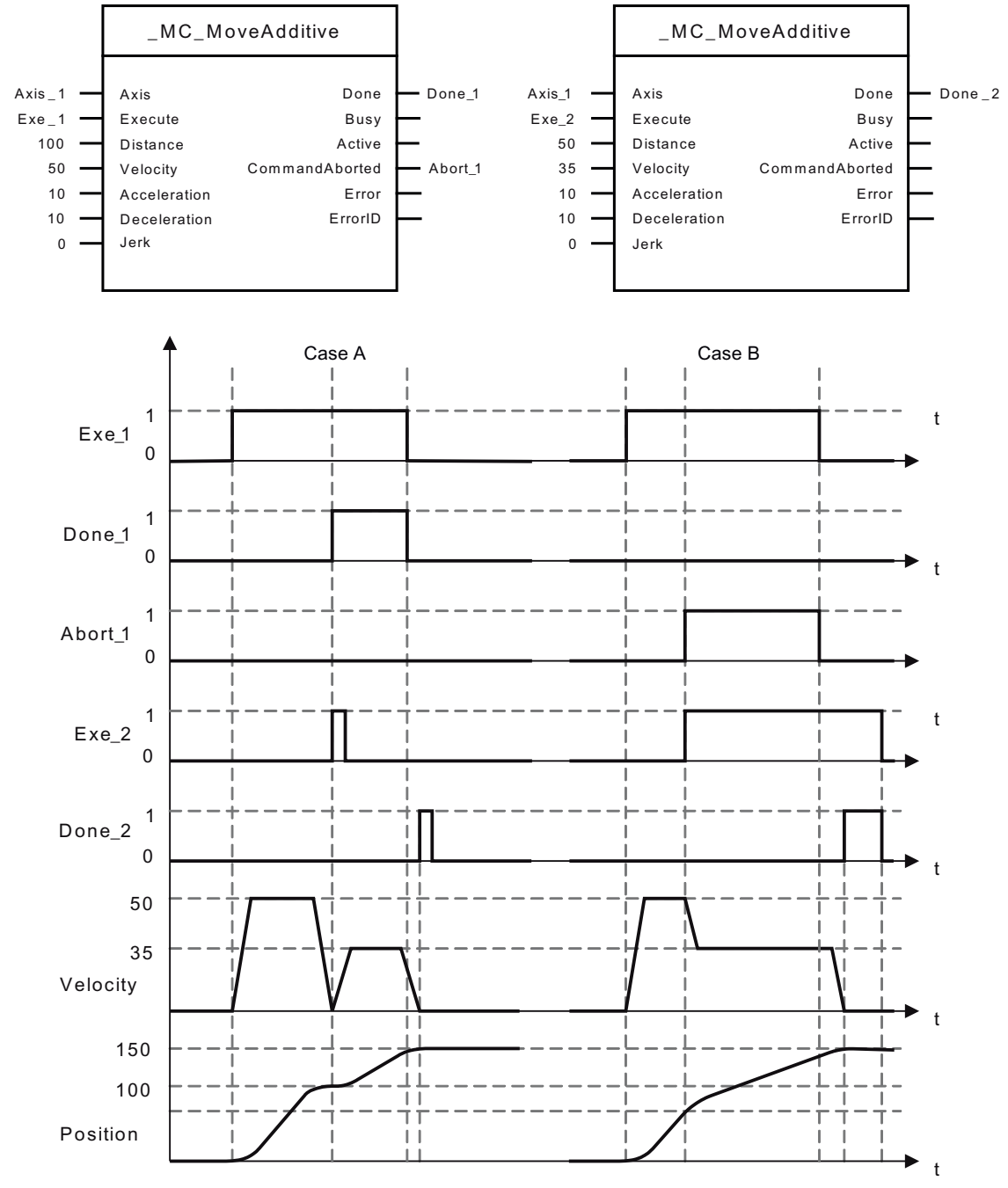


Figure 8-98 _MC_MoveAdditive Example

_MC_MoveSuperimposed - Superimposed positioning

Overview

Schematic diagram

Schematic diagram

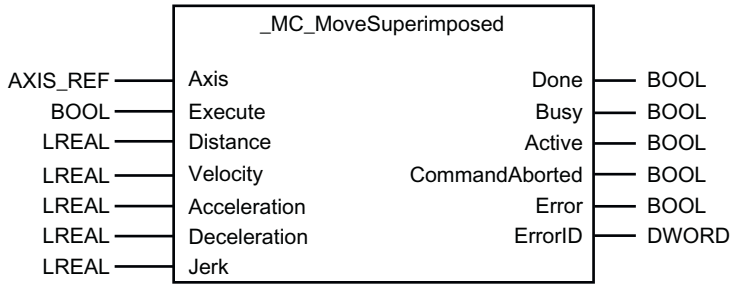


Figure 8-99 _MC_MoveSuperimposed Schematic diagram

Purpose

Purpose

The function block **_MC_MoveSuperImposed** starts a positioning motion relative to the active positioning motion of an axis. This enables a superimposed positioning of an axis, e.g. for the print-mark correction.

The dynamic response parameters **VelocityDiff**, **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

An active motion command (main motion) is not overridden by the function block.

An active superimposed positioning motion is overridden by a restart of the function block **_MC_MoveSuperImposed**. The remaining distance-to-go of the overridden superimposed positioning motion is lost.

Note

The state of the "DONE" output depends on the system variable motionstatedata.motionstate.

Applicable for

Applications

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

No **_MC_Stop** active

The axis velocity is increased for the superimposed positioning operation. Therefore, the basic motion of the axis should not be performed with the maximum permissible velocity.

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The positioning operation starts with a rising edge at this input.
Distance	LREAL	0.0	Specification of the distance difference to be traversed

Parameters	Data type	Initial value	Description
Velocity	LREAL	-1.0	Specification of the maximum velocity The velocity is reached in relation to the set values for the traversing distance, acceleration, and jerk. Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the programmed target position has been reached.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6651)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

Case A:

An _MC_MoveSuperImposed is started during a relative positioning.

Case B:

MC_MoveSuperImposed is started again before _MC_MoveSuperImposed is completed.

Case C:

Start _MC_MoveSuperImposed with a stationary axis.

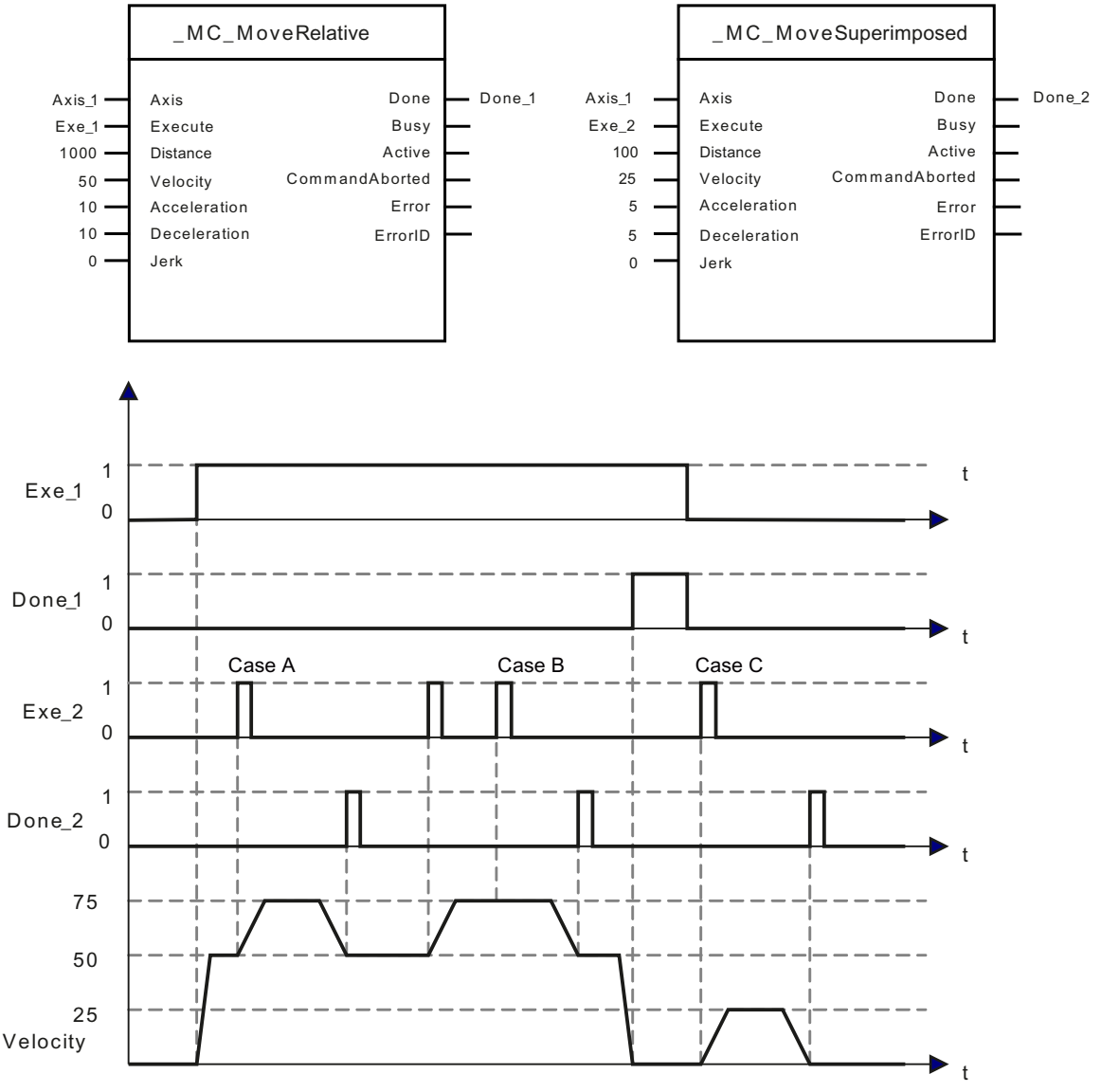


Figure 8-100 _MC_MoveSuperimposed Example

_MC_PositionProfile - Traveling through position/time profile

Overview

Schematic diagram

Purpose

Applicable for

Requirements

Schematic diagram

Schematic diagram

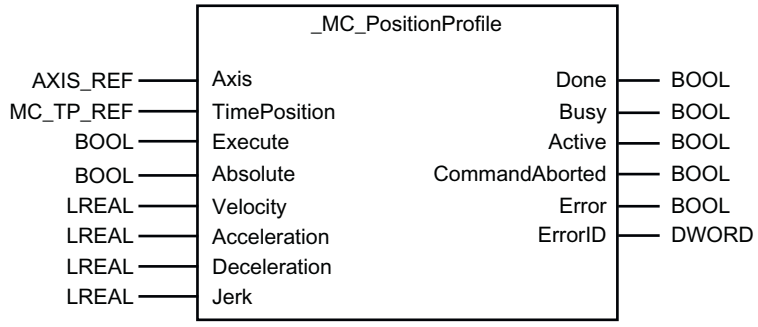


Figure 8-101 _MC_PositionProfile Schematic diagram

Purpose

Purpose

The function block **_MC_PositionProfile** traverses an axis along a position profile which is specified as an s(t) function.

Applicable for

Applications

Positioning axes
Following axes
Path axes

Requirements

Requirements

Axis enabled

Axis homed if the axis configuration data item **TypeOfAxis.Homing.referencingNecessary** was set to **YES** (homing required).

Position profiles are specified using cams and must be assigned to the axis (in the "Profiles" axis dialog).

No **_MC_Stop** active

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
TimePosition	MC_TP_REF	0	Specification of cam profile (name) that describes the position as a function of the time: • Cam (CamType data type)
Execute	BOOL	FALSE	Function block enable The positioning motion starts with a rising edge at this input.
Absolute	BOOL	TRUE	Specification of the traversing method With TRUE, the cam positions are approached according to the absolute values. With FALSE, the position profile of the cam is set at the actual position of the axis.
Velocity	LREAL	-1.0	Specification of the maximum velocity The velocity is reached depending on the set values for traversing distance, acceleration and jerk. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used. Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the axis has completed the specified position profile.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6656)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

Case A:

Two _MC_PositionProfile blocks are started in succession.

Case B:

The second _MC_PositionProfile block aborts the motion of the first block.

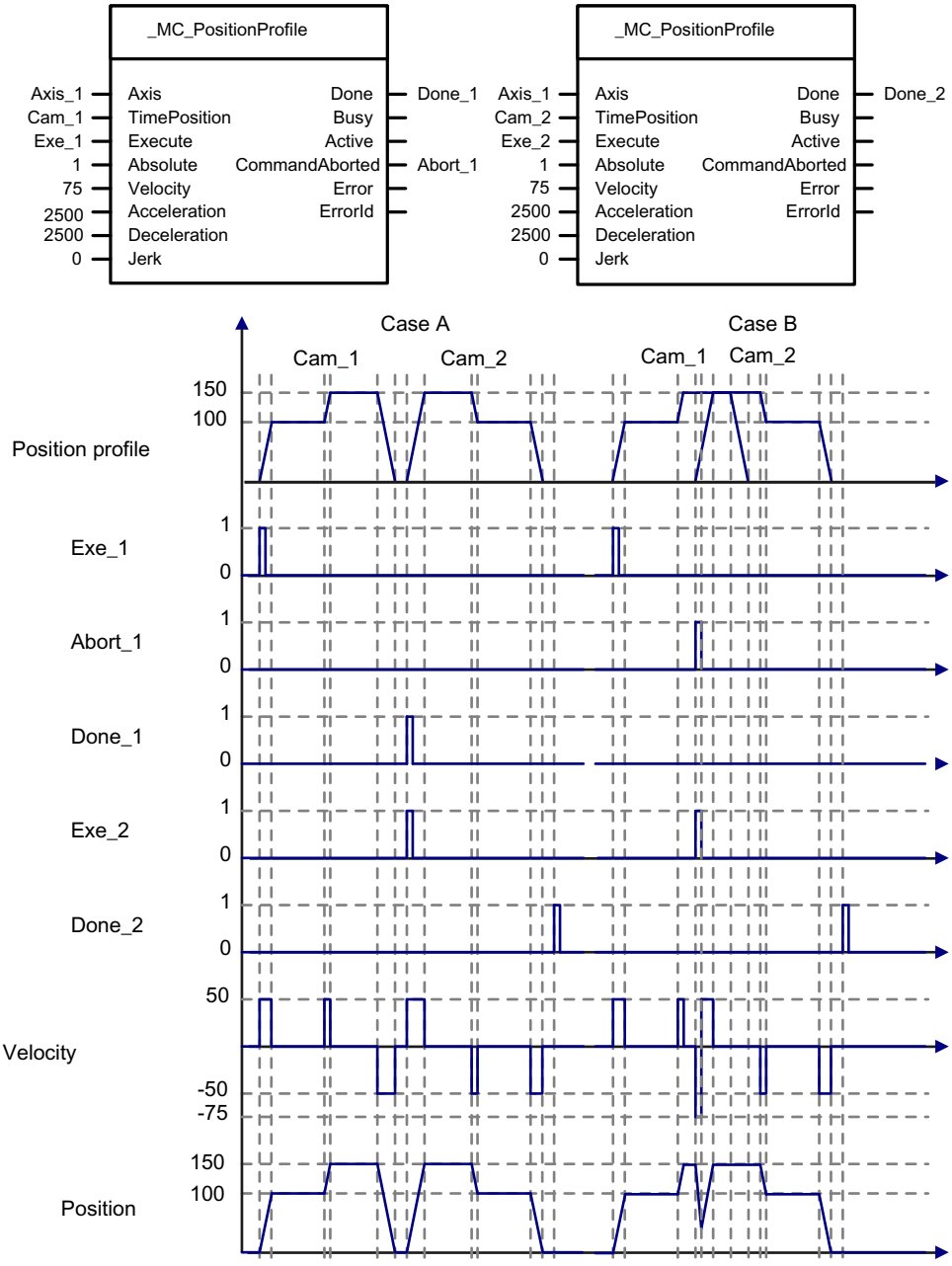


Figure 8-102 _MC_PositionProfile example

_MC_VelocityProfile - Traveling through velocity/time profile

Overview

Schematic diagram

Schematic diagram

Schematic diagram

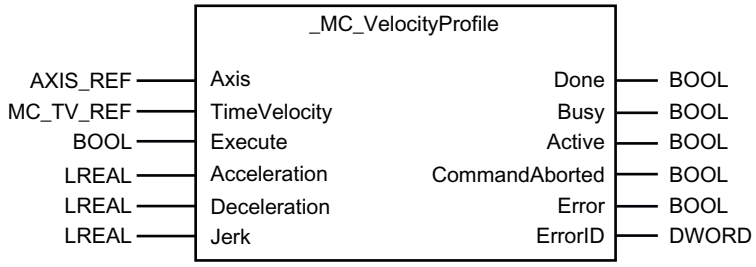


Figure 8-103 _MC_VelocityProfile Schematic diagram

Purpose

Purpose

The function block **_MC_VelocityProfile** traverses an axis along a velocity profile, which is specified as a v(t) function.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes

Requirements

Requirements

Axis enabled

Velocity profiles are specified using cams and must be assigned to the axis (in the "Profiles" axis dialog).

No **_MC_Stop** active

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
TimeVelocity	MC_TV_REF	0	Specification of cam profile (name) that describes the velocity as a function of the time: • Cam (CamType data type)
Execute	BOOL	FALSE	Function block enable The motion starts with a rising edge at this input.
Acceleration	LREAL	-1.0	Specification of the maximum acceleration (increasing energy in the motor) Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration (decreasing energy in the motor) Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the axis has traversed the specified velocity profile.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6661)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

Case A:

Two _MC_VelocityProfile blocks are started in succession.

Case B:

The second _MC_VelocityProfile block aborts the motion of the first block.

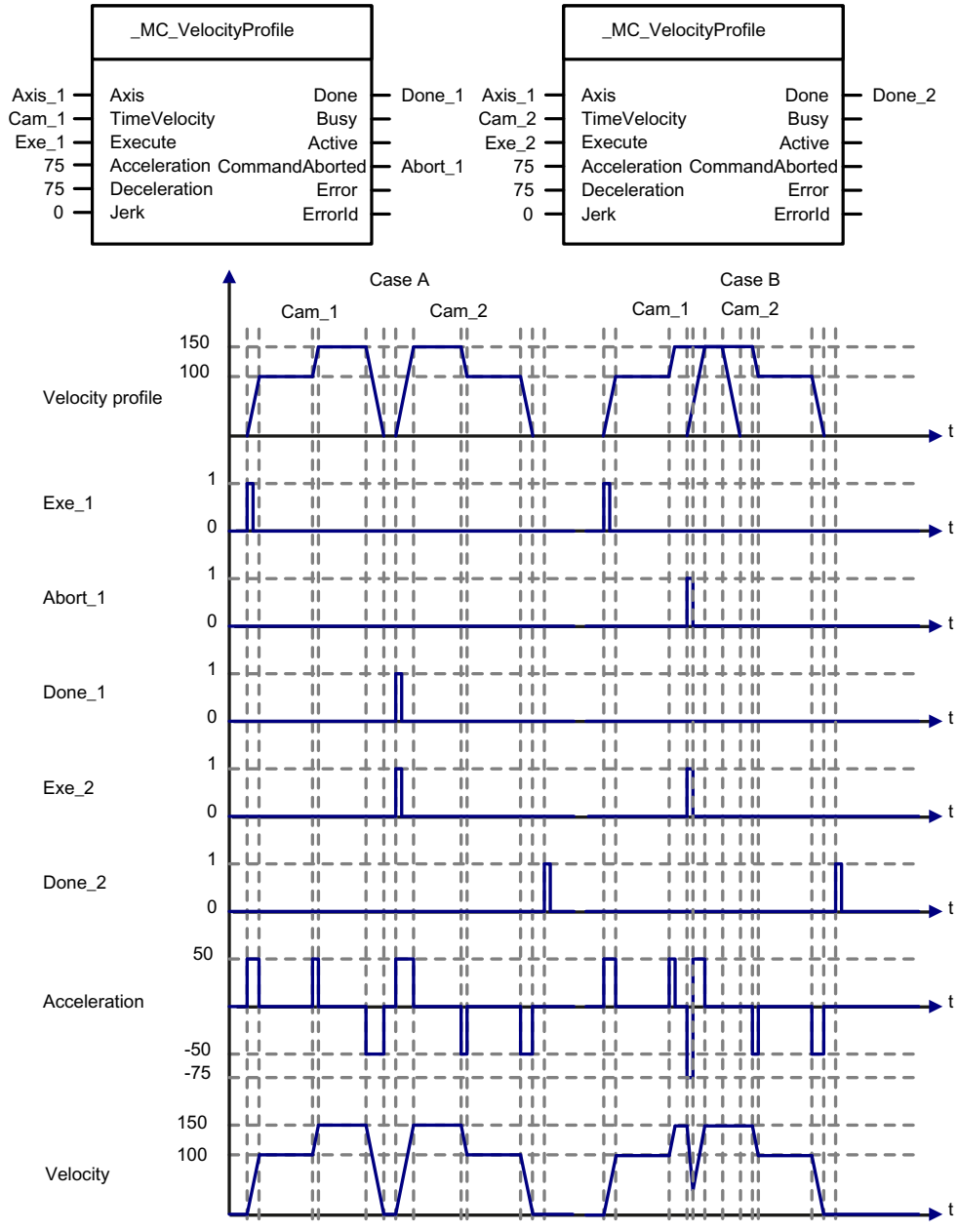


Figure 8-104 _MC_VelocityProfile example

_MC_Reset - Resetting errors/alarms on the axis or triggering a restart

Overview

Schematic diagram

Schematic diagram

Schematic diagram

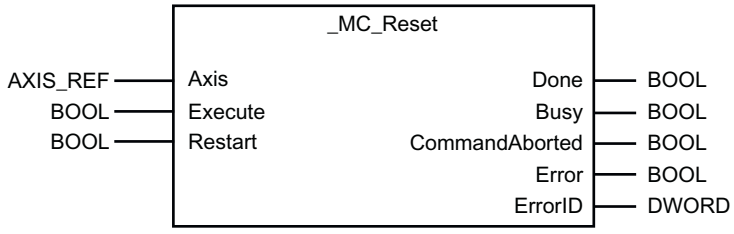


Figure 8-105 _MC_Reset Schematic diagram

Purpose

Purpose

The function block **_MC_Reset** resets all errors on an axis or external encoder that can be acknowledged by means of the software. Fatal errors can be acknowledged via Power off/on or reloading of the project to the module.

If the **Restart** input is set, the transferred technology object is re-initialized via **_MC_Reset**. Axes that are operated with incremental encoders return to the **Not homed** mode.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes
External Encoders

Requirements

Requirements

The restart of an axis depends on the condition set with configuration data item **restartAxisCondition**.

Note

- Set the input **Restart** to **FALSE**, only when the errors present on the transferred technology object are to be acknowledged.
- No variables of the technology object are updated during the restart operation!

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Execute	BOOL	FALSE	Function block enable With a rising edge at this input, the function block acknowledges the errors present on the transferred technology object or optionally triggers a restart.
Restart	BOOL	FALSE	Specification of the reset type With TRUE, the transferred technology object is restarted and modified configuration data are accepted. With FALSE, the alarms pending on the transferred technology object are acknowledged.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block The alarms have been acknowledged or a restart performed.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6665)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

Case A:

The _MC_Reset block resets active errors/alarms at the axis transferred on input parameter **Axis**.

Case B:

The _MC_Reset block triggers a restart at the axis transferred on input parameter **Axis**.

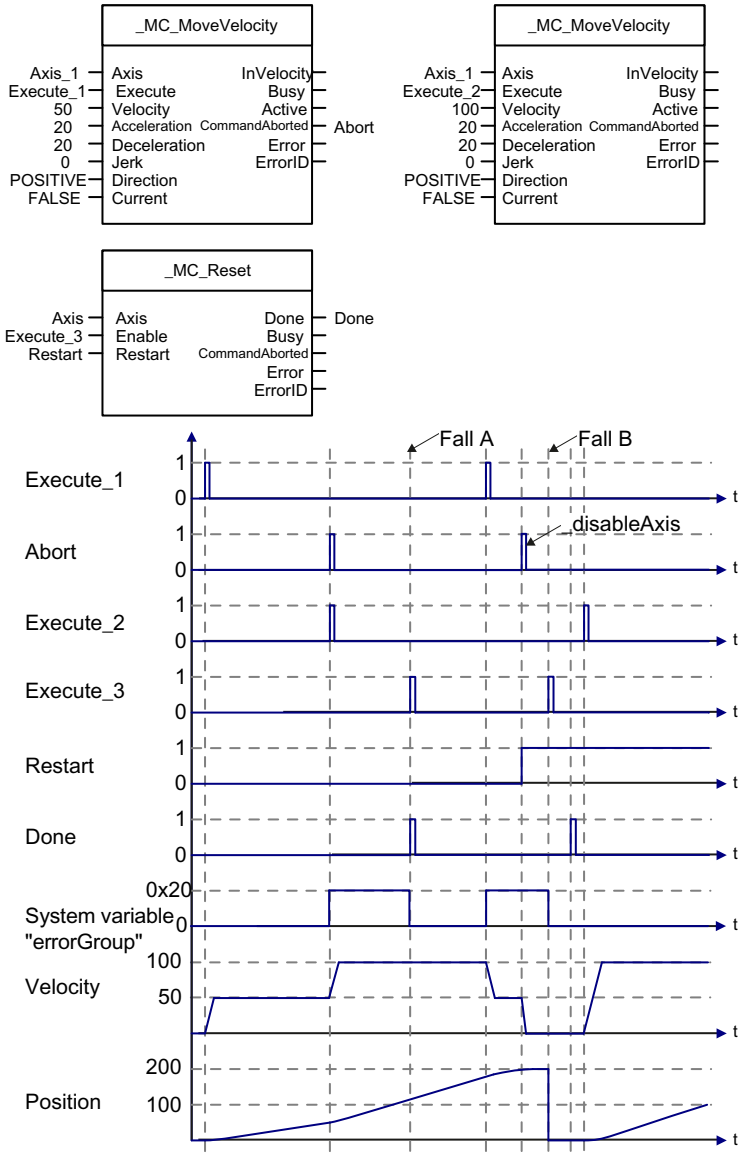


Figure 8-106 _MC_Reset example

_MC_ReadActualPosition - Reading actual position of axis

Overview

Schematic diagram

Purpose

Applicable for

Schematic diagram

Schematic diagram

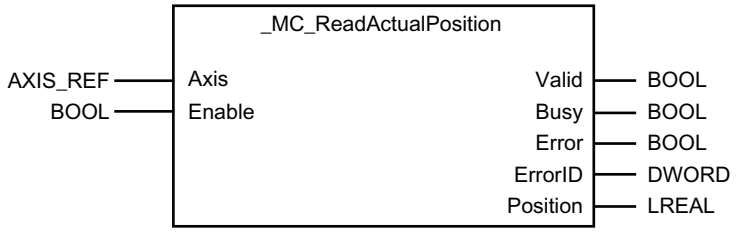


Figure 8-107 _MC_ReadActualPosition Schematic diagram

Purpose

Purpose

The function block **_MC_ReadActualPosition** reads the actual position of an axis or an external encoder.

As an output value, the block supplies:

- The value for the **positioningState.actualPosition** system variable, in the case of axes
- The value for the **motionState.position** system variable, in the case of external encoders

Applicable for

Applications

Positioning axes
Following axes
Path axes
External encoders

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The value at output **Position** is updated as long as **Enable** equals **TRUE**.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Valid	BOOL	FALSE	Display of the validity of the value which can be read at output **Position**.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6668)).
Position	LREAL	0.0	Displays the actual position of the axis or external encoder

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_ReadActualPosition block reads the actual position of the axis transferred on the **Axis** input parameter.

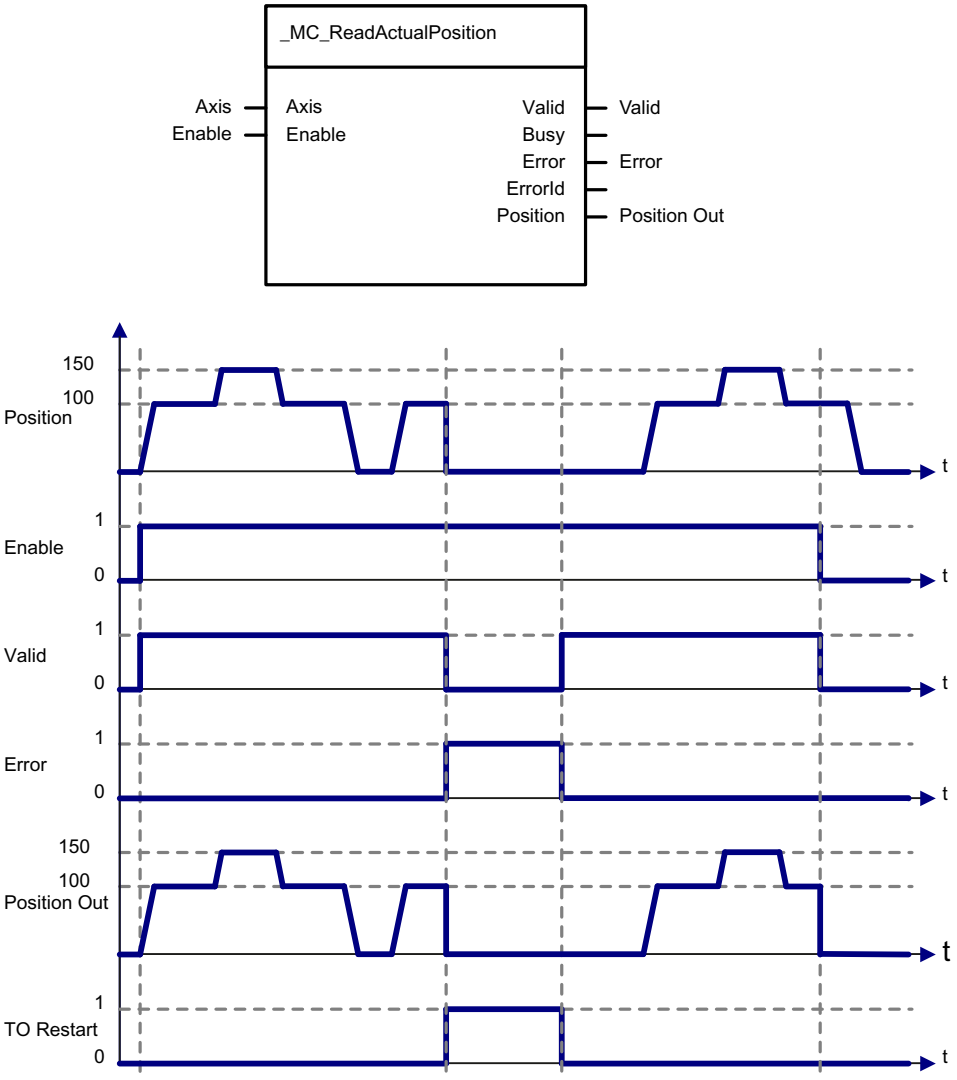


Figure 8-108 _MC_ReadActualPosition example

_MC_ReadStatus - Reading an axis status

Overview

Schematic diagram

Purpose

Applicable for

Input parameters

Output parameters

ErrorIDs

Example

Schematic diagram

Schematic diagram

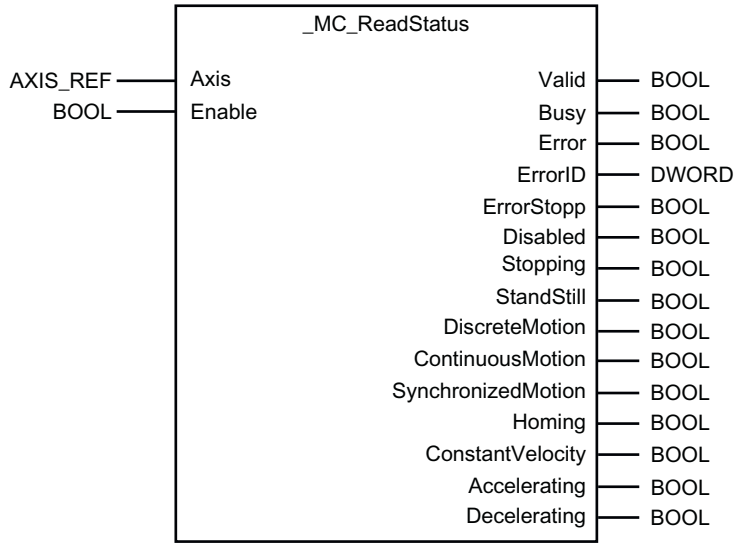


Figure 8-109 _MC_ReadStatus Schematic diagram

Purpose

Purpose

The function block **_MC_ReadStatus** reads various states of an axis or an external encoder.

Applicable for

Applications

Drive axes

Positioning axes

Following axes

Path axes

External encoders

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The values at the status outputs are updated as long as **Enable** equals **TRUE**.

Output parameter

Output parameter

Table 8-172 Generally applicable output parameter

Parameter	Data type	Initial value	Description
Valid	BOOL	FALSE	Display of the validity of the values that can be read at the status outputs
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6672)).

Table 8-173 Output parameters for axes

Parameter	Data type	Initial value	Description
ErrorStop	BOOL	FALSE	Display of an active axis stop triggered by an error
Disabled	BOOL	FALSE	Display of no enable of the axis
Stopping	BOOL	FALSE	Display of an active regular axis stop
StandStill	BOOL	FALSE	Display of the standstill signal of the axis
DiscreteMotion	BOOL	FALSE	Display of an active single axis motion with discrete target position
ContinuousMotion	BOOL	FALSE	Display of an active single axis motion without a discrete target position

Parameter	Data type	Initial value	Description
SynchronizedMotion	BOOL	FALSE	Display of an active synchronous operation on the axis
Homing	BOOL	FALSE	Display of an active homing procedure on the axis
ConstantVelocity	BOOL	FALSE	Display of traversing the axis with a constant velocity
Accelerating	BOOL	FALSE	Display of traversing the axis with an accelerating velocity
Decelerating	BOOL	FALSE	Display of traversing the axis with a decelerating velocity

Table 8-174 Output parameter for external encoders

Parameter	Data type	Initial value	Description
ErrorStop	BOOL	FALSE	Display of an error on the external encoder
Disabled	BOOL	FALSE	Display of no enable of the external encoder
StandStill	BOOL	FALSE	Display of the standstill signal of the external encoder
Homing	BOOL	FALSE	Display of an active homing procedure on the external encoder
ConstantVelocity	BOOL	FALSE	Display of a constant velocity that is measured/determined by the external encoder
Accelerating	BOOL	FALSE	Display of an accelerating velocity that is measured/determined by the external encoder
Decelerating	BOOL	FALSE	Display of a decelerating velocity that is measured/determined by the external encoder

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_ReadStatus block reads the motion statuses of the axis transferred on the **Axis** input parameter during an active positioning command.

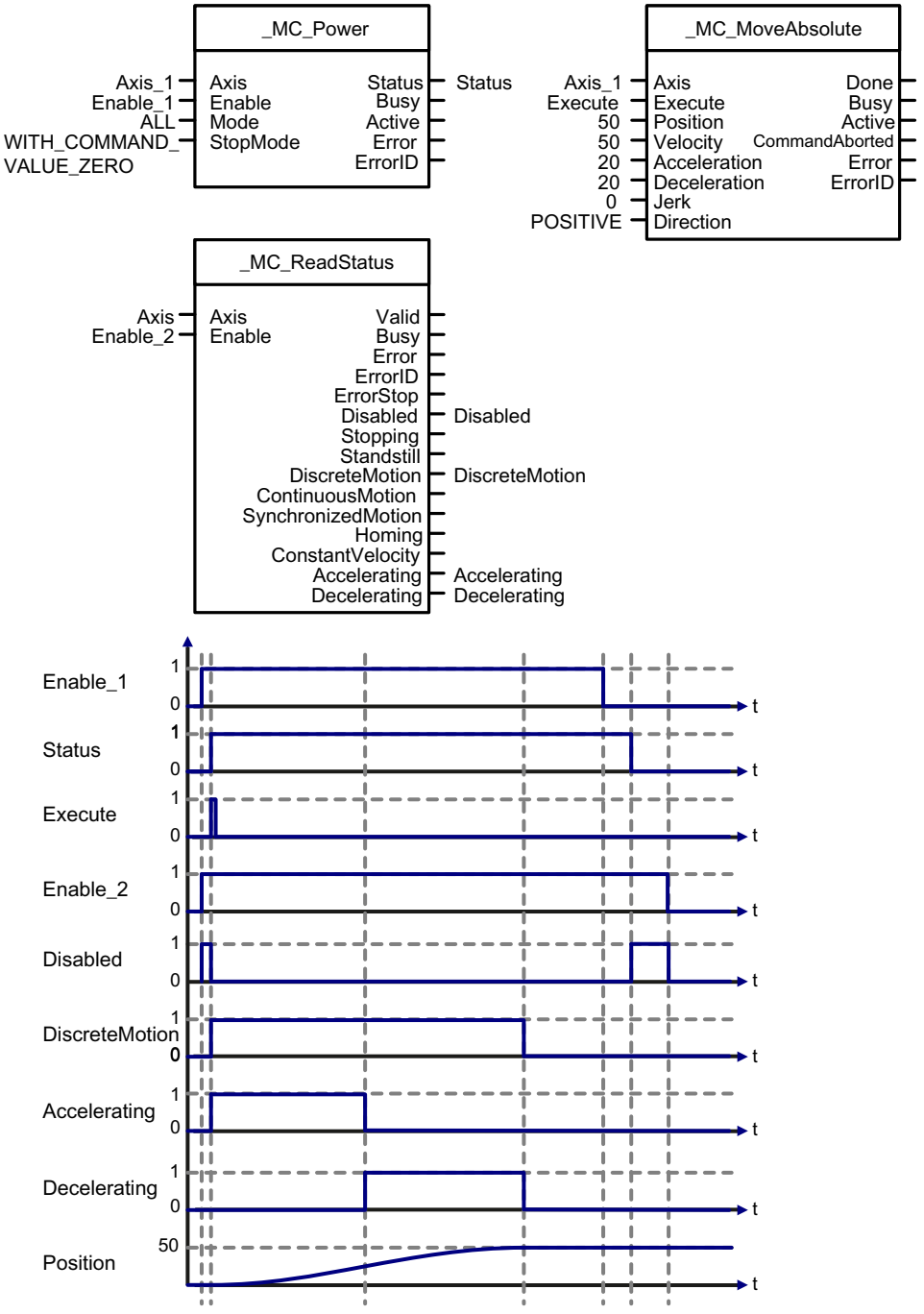


Figure 8-110 _MC_ReadStatus example

_MC_ReadAxisError - Reading an axis error

Overview

Schematic diagram

Schematic diagram

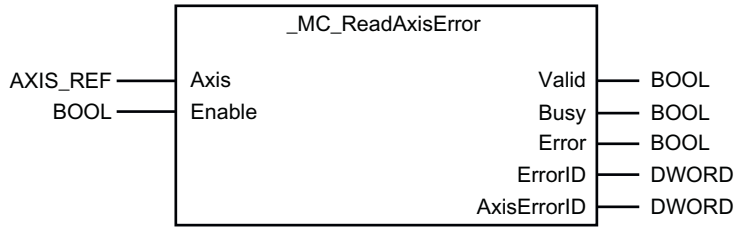


Figure 8-111 _MC_ReadAxisError Schematic diagram

Purpose

Purpose

The function block **_MC_ReadAxisError** reads the error status of an axis or an external encoder. The error status is a 32-bit representation of all alarms present on the technology object.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes
External encoders

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The value at output **AxisErrorID** is updated as long as **Enable** equals **TRUE**.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Valid	BOOL	FALSE	Display of the validity of the value which can be read at output **AxisErrorID**.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6675)).
AxisErrorID	DWORD	0	Displays the error status of the axis or external encoder For more information, refer to the section Query of general errors with the _MC_ReadAxisError function block (Page 6737).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

Generally, the following is valid:

- The **ErrorID** on the **_MC_...** function blocks refer to special block-related errors and to errors on the axis
- The **AxisErrorID** on the **_MC_ReadAxisError** function block refers to errors on the axis

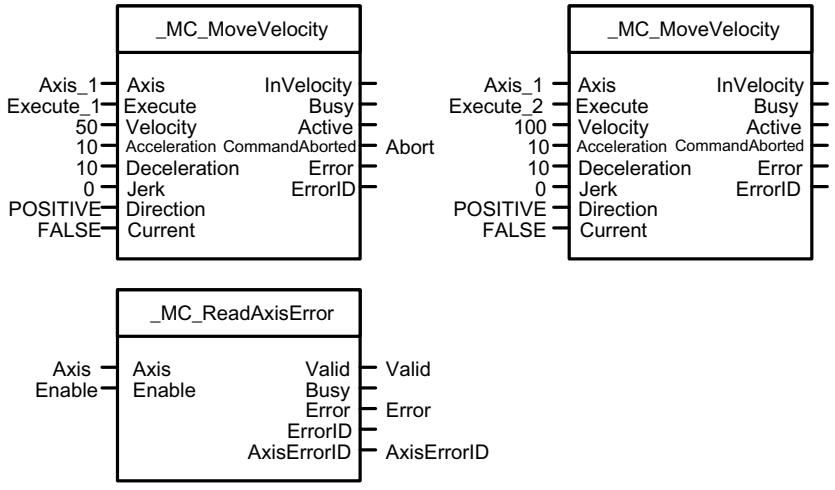
See also

Troubleshooting - PLCopen Blocks (Page 6733)

Query of general errors with the _MC_ReadAxisError function block (Page 6737)

Example

The _MC_ReadAxisError block reads the group error bit string ("errorGroup" system variable) value of the axis transferred on the **Axis** input parameter.



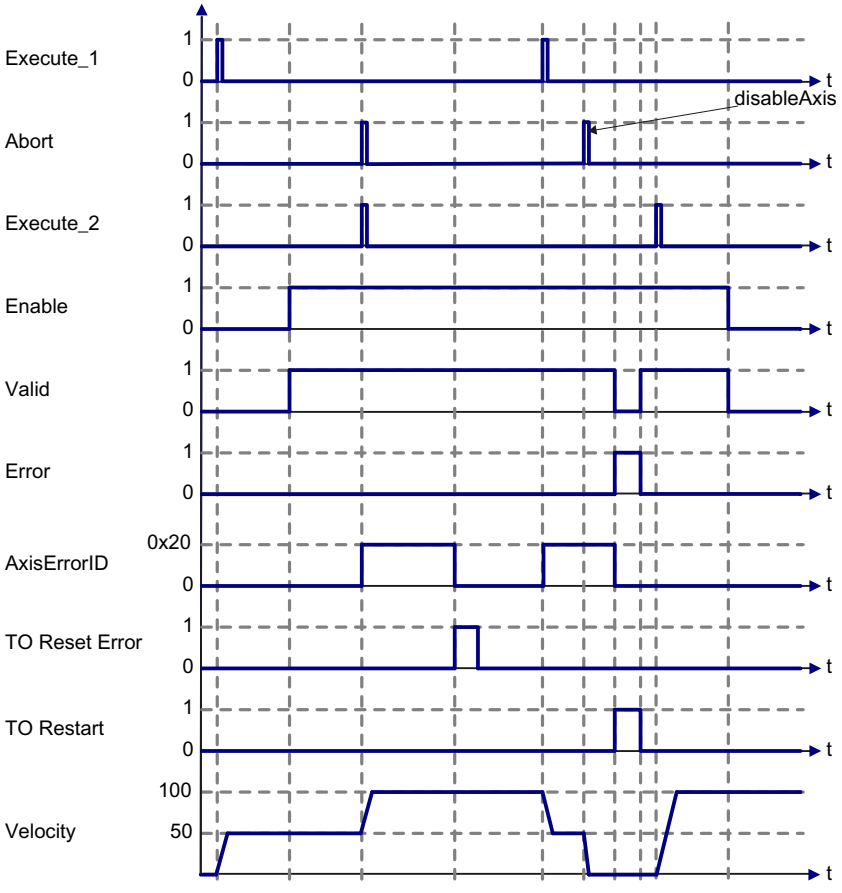


Figure 8-112 _MC_ReadAxisError example

_MC_ReadParameter - Reading axis parameter and outputting in data type LREAL

Overview

Schematic diagram

Schematic diagram

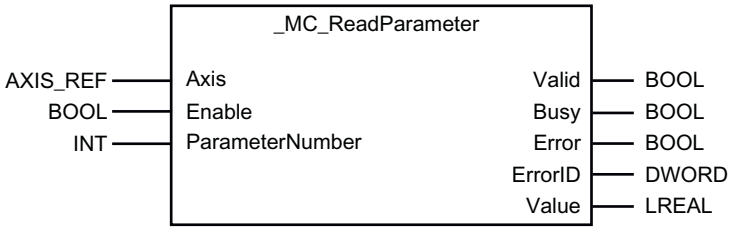


Figure 8-113 _MC_ReadParameter Schematic diagram

Purpose

Purpose

The function block **_MC_ReadParameter** reads values of various parameters of an axis or external encoder. Each parameter is specified by a number.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes
External encoders

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The value at output **Value** is updated as long as **Enable** equals **TRUE**.
ParameterNumber	INT	0	Specification of the number of the parameter to be read (see table below).

Parameter description for input ParameterNumber

Table 8-175 Input ParameterNumber for axes

Parameter	Data type	Access	Description
1	LREAL	Read	**positioningstate.commandposition**: Axis position setpoint
2	LREAL	Read/Write	**swlimit.plusposition**: Position of the positive software limit switch (10^12 is output with inactive limit switch.)
3	LREAL	Read/Write	**swlimit.minusposition**: Position of the negative software limit switch (-10^12 is output with inactive limit switch.)
7 [1)]	LREAL	Read/Write	**typeofaxis.numberofdatasets.dataset_x.dynamicfollowing.minpositiontolerance**: Minimum permissible following error
8	LREAL	Read/Write	**typeofaxis.maxvelocity.maximum**: Maximum permissible velocity
10	LREAL	Read	**motionstatedata.actualvelocity:** Actual axis velocity (A value is only displayed if an encoder was configured on the axis. For drive axes, the encoder must be configured for the setpoint acceptance if applicable.)
11	LREAL	Read	**motionstatedata.commandvelocity**: Axis velocity setpoint
12	LREAL	Read/Write	**typeofaxis.maxacceleration.maximum**: Maximum permissible acceleration
16	LREAL	Read/Write	**typeofaxis.maxjerk.maximum**: Maximum permissible jerk

[1)] The "x" in the name "dataset_x" stands for the active data set.

Table 8-176 Input ParameterNumber for external encoders

Parameter	Data type	Access	Description
10	LREAL	Read	**motionState.velocity:** Actual velocity measured by the external encoder

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Valid	BOOL	FALSE	Indicates the validity of the value that can be read at output **Value**
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6680)).
Value	LREAL	0.0	Display of the parameter value

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_ReadParameter block reads the actual velocity of the axis transferred on the **Axis** input parameter.

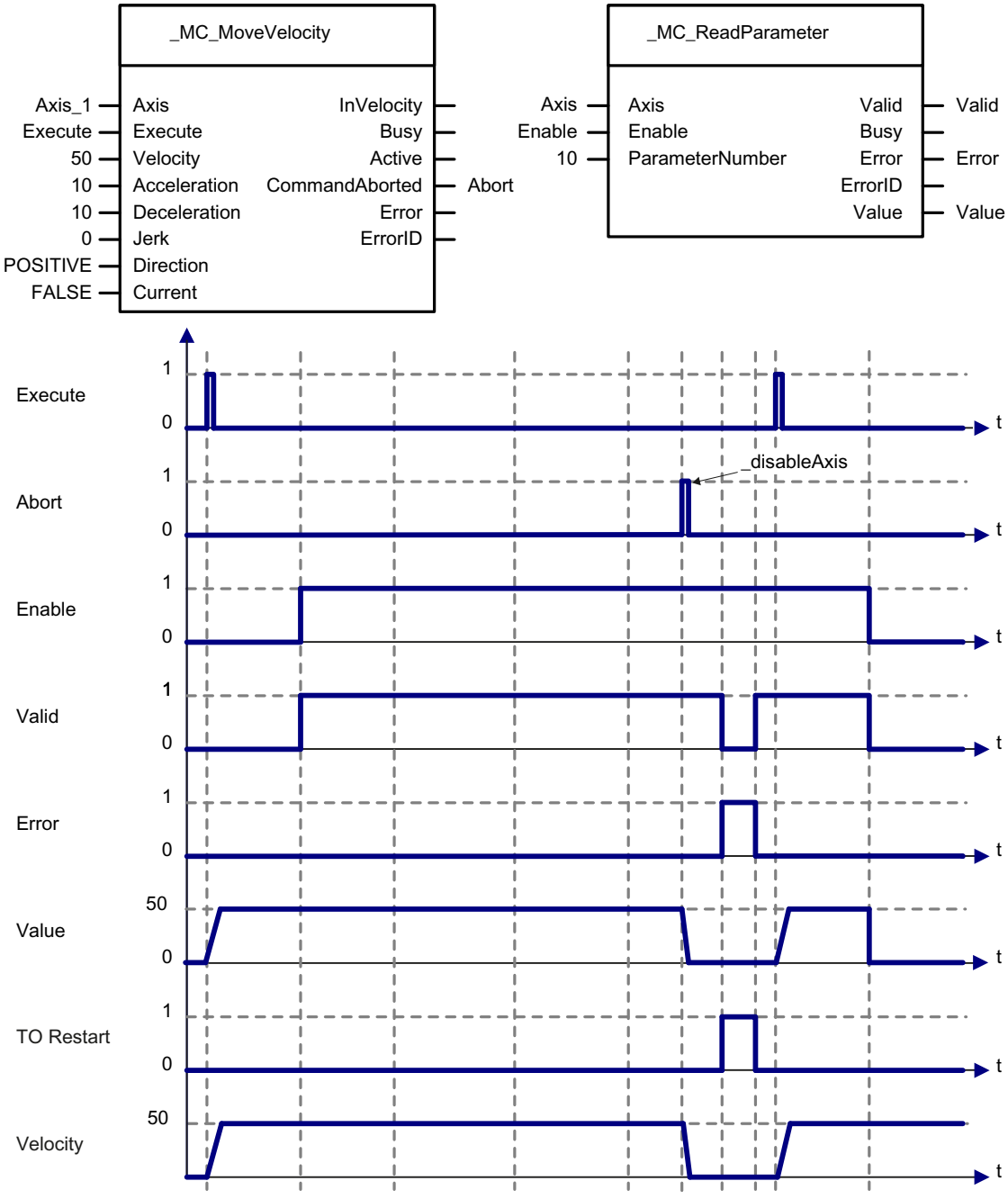


Figure 8-114 _MC_ReadParameter example

_MC_ReadBoolParameter - Reading axis parameter and outputting in data type BOOL

Overview

Schematic diagram

Schematic diagram

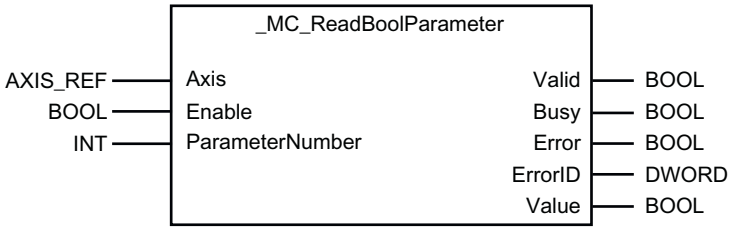


Figure 8-115 _MC_ReadBoolParameter Schematic diagram

Purpose

Purpose

The function block **_MC_ReadBoolParameter** reads values of various parameters of an axis or external encoder. Each parameter is specified by a number. The return value of the parameter is converted to the data type of the function block (BOOL), i.e. values that are not of the data type BOOL are converted to FALSE when they are exactly 0 and to TRUE in all other cases.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Enable	BOOL	FALSE	Function block enable The value at output **Value** is updated as long as **Enable** equals **TRUE**.
ParameterNumber	INT	0	Specification of the number of the parameter to be read (see table below).

Parameter description for input ParameterNumber

Table 8-177 Input ParameterNumber for axes

Parameter	Data type	Description
4	BOOL	**swlimit.state**: Activation state of the two software limit switches (The limit switches are always activated or deactivated together.)
6 [1)]	BOOL	**typeofaxis.numberofdatasets.dataset_x.dynamicfollowing.enable**: Activation state of the following error monitoring

[1)] The "x" in the name "dataset_x" stands for the active data set.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Valid	BOOL	FALSE	Indicates the validity of the value that can be read at output **Value**
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.

Parameter	Data type	Initial value	Description
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6684)).
Value	BOOL	FALSE	Display of the parameter value

ErrorIDs

ErrorIDs

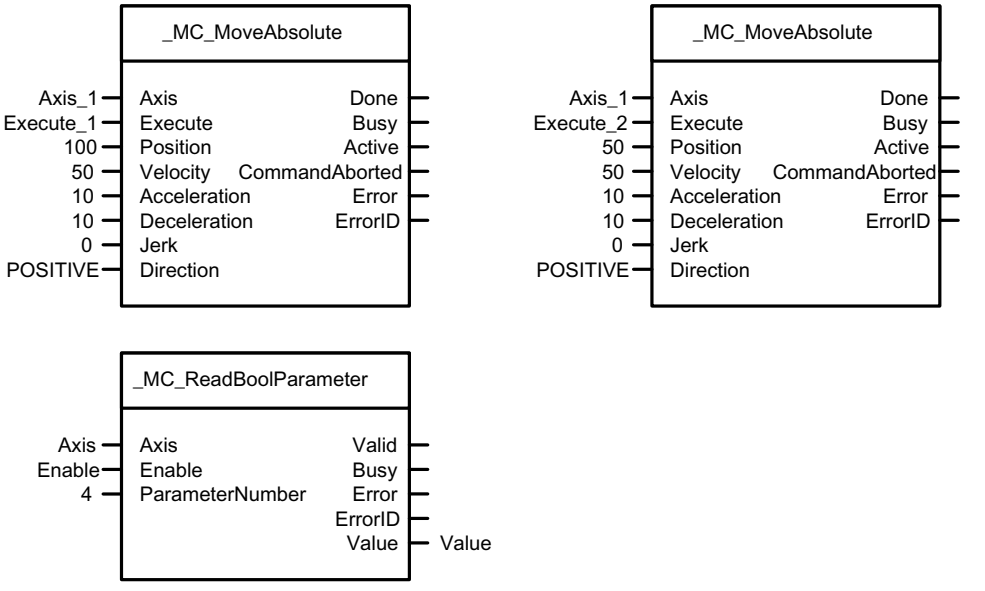
The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_ReadBoolParameter block reads the monitoring status of the software limit switches and converts the value read into the Boolean output value displayed.



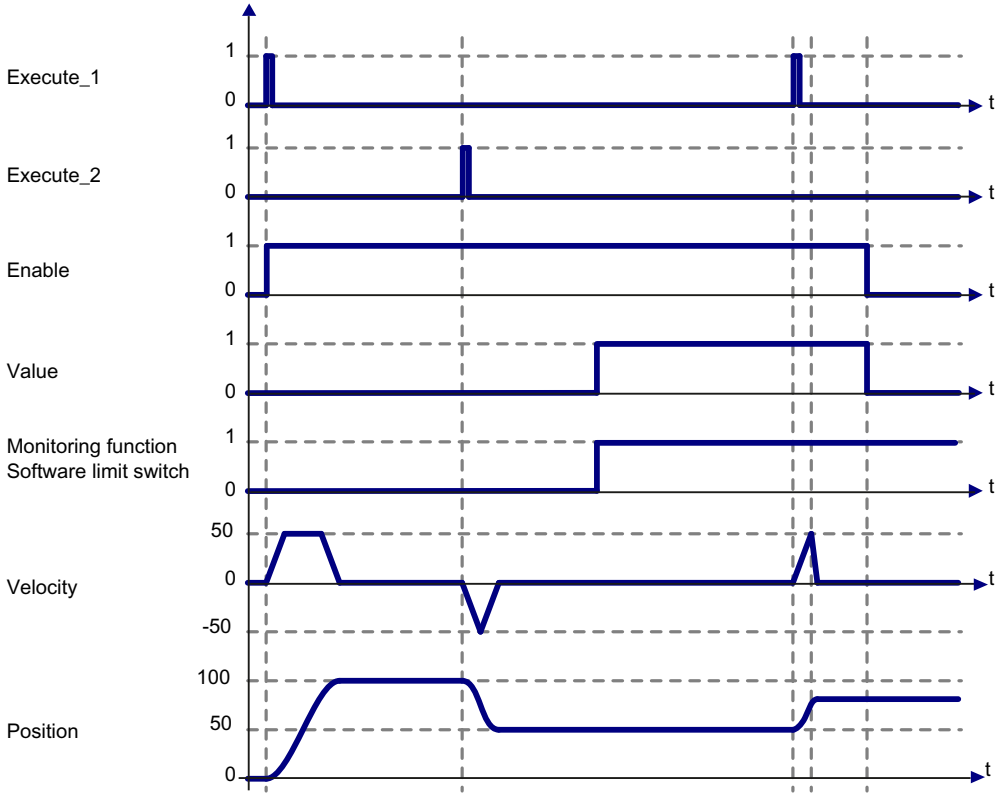


Figure 8-116 _MC_ReadBoolParameter example

_MC_WriteParameter - Writing axis parameter of data type LREAL

Overview

Schematic diagram

Schematic diagram

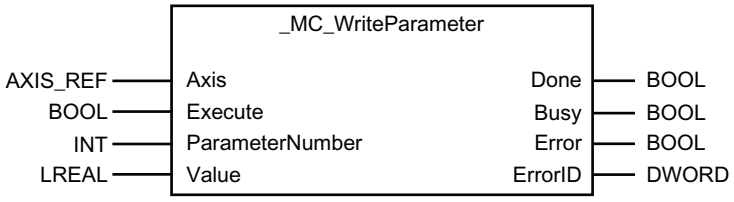


Figure 8-117 _MC_WriteParameter Schematic diagram

Purpose

Purpose

The function block **_MC_WriteParameter** writes values of various parameters of an axis. Each axis parameter is specified by a number.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis) • Positioning axis (posAxis) • Following axis (followingAxis) • Path axis (pathAxis)
Execute	BOOL	FALSE	Function block enable The parameter at input **Value** is written with a rising edge at this input.
ParameterNumber	INT	0	Specification of the number of the parameter to be written (see table below)
Value	LREAL	0.0	Specification of the parameter value to be written

Parameter description for input ParameterNumber

Parameter number	Data type	Description
2	LREAL	**swlimit.plusposition**: Position of the positive software limit switch
3	LREAL	**swlimit.minusposition**: Position of the negative software limit switch
7 [1) 2)]	LREAL	**typeofaxis.numberofdatasets.dataset_x.dynamicfollowing.minpositiontolerance; typeofaxis.numberofdatasets.dataset_x.dynamicfollowing.maxpositiontolerance**: Minimum or maximum permissible following error
8	LREAL	**typeofaxis.maxvelocity.maximum**: Maximum permissible velocity (the change only takes effect after a restart)
12	LREAL	**typeofaxis.maxacceleration.maximum**: Maximum permissible acceleration
16	LREAL	**typeofaxis.maxjerk.maximum**: Maximum permissible jerk

1) The "x" in the name "dataset_x" stands for the active data set, i.e. the data item that is combined with parameter 7 is only changed at this data set.

2) Not available at virtual axes

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the parameter was able to be written successfully.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6688)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_WriteParameter block changes the maximum permissible acceleration for the axis transferred on the **Axis** input parameter.

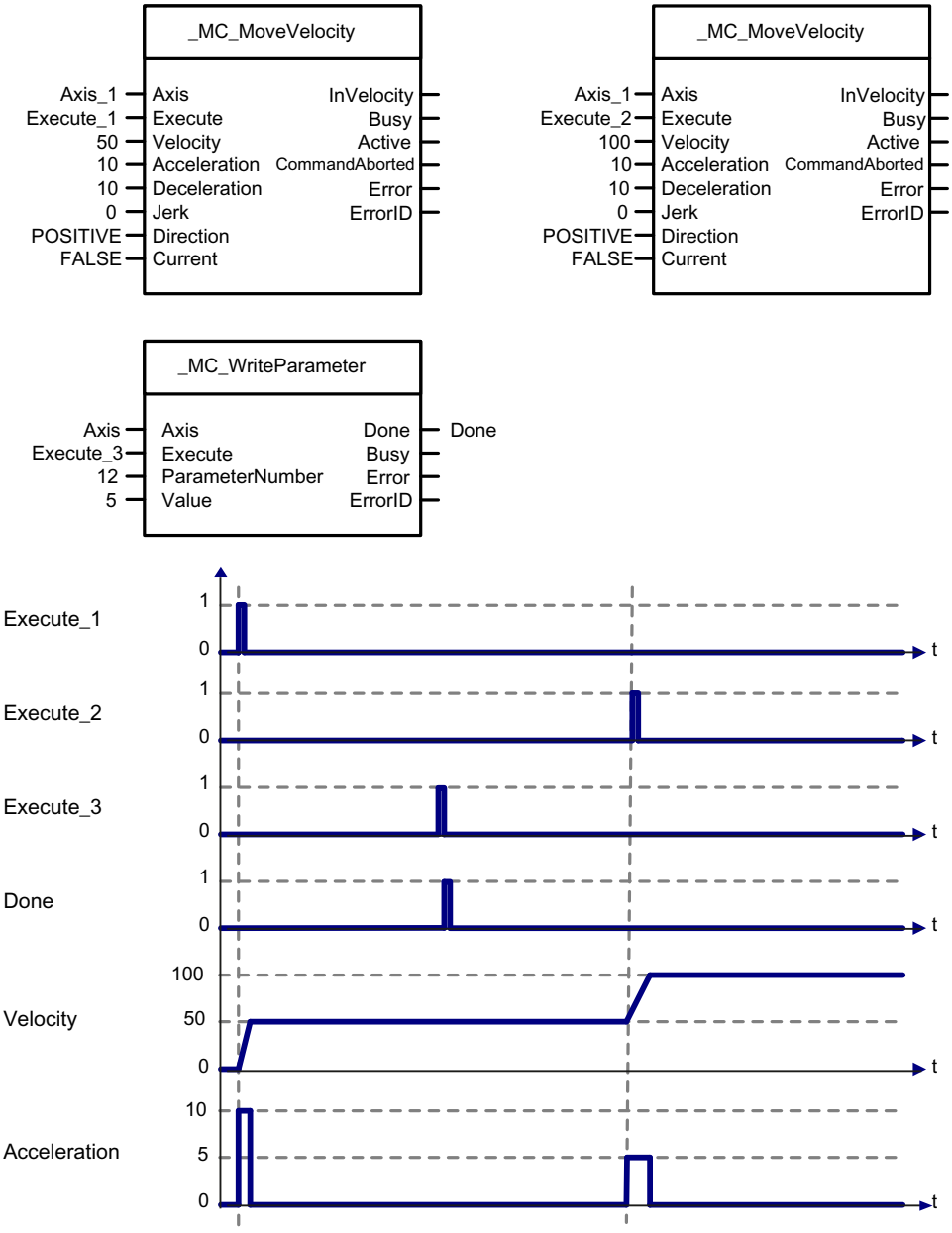


Figure 8-118 _MC_WriteParameter example

_MC_WriteBoolParameter - Writing axis parameter of data type BOOL

Overview

Schematic diagram

Schematic diagram

Schematic diagram

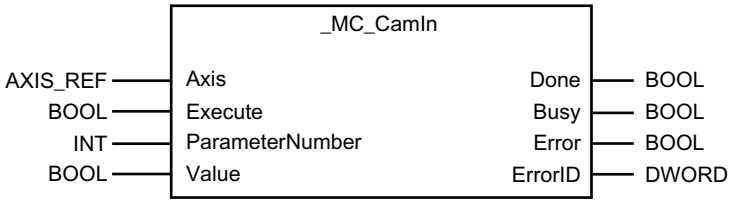


Figure 8-119 _MC_WriteBoolParameter Schematic diagram

Purpose

Purpose

The function block **_MC_WriteBoolParameter** writes values of various parameters of an axis. Each axis parameter is specified by a number.

Applicable for

Applications

Drive axes
Positioning axes
Following axes
Path axes

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Drive axis (driveAxis data type) • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The parameter at input **Value** is written with a rising edge at this input.
ParameterNumber	INT	0	Specification of the number of the parameter to be written (see table below)
Value	BOOL	FALSE	Specification of the parameter value to be written

Parameter description for input ParameterNumber

Parameter number	Data type	Description
4	BOOL	**swlimit.state**: Activation of the positive and negative software limit switches (the limit switches are always activated or deactivated together)
6 [1) 2)]	BOOL	**typeofaxis.numberofdatasets.dataset_x.dynamicfollowing.enable**: Activation of following error monitoring (the change only takes effect after a restart) The data sets are configured at the corresponding technology object. Several data sets can be used, in order to (for example): • Change over controller data during operation • Change over the encoder used during operation (motor encoder, machine encoder, etc.) For more details, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual.

1) The "x" in the name "dataset_x" stands for all existing data sets, i.e. the data item that is combined with parameter 6 is changed in all data sets at the same time.

2) Not available at virtual axes

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the parameter was able to be written successfully.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.

Parameter	Data type	Initial value	Description
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6692)).

ErrorIDs

ErrorIDs

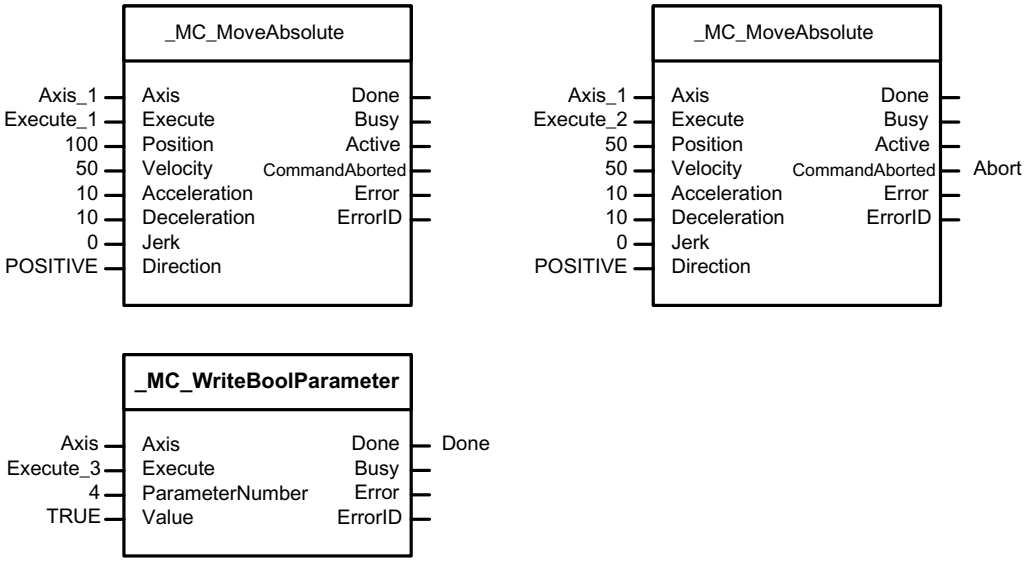
The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

The _MC_ WriteBoolParameter block activates software limit switch monitoring for the axis transferred on the **Axis** input parameter.



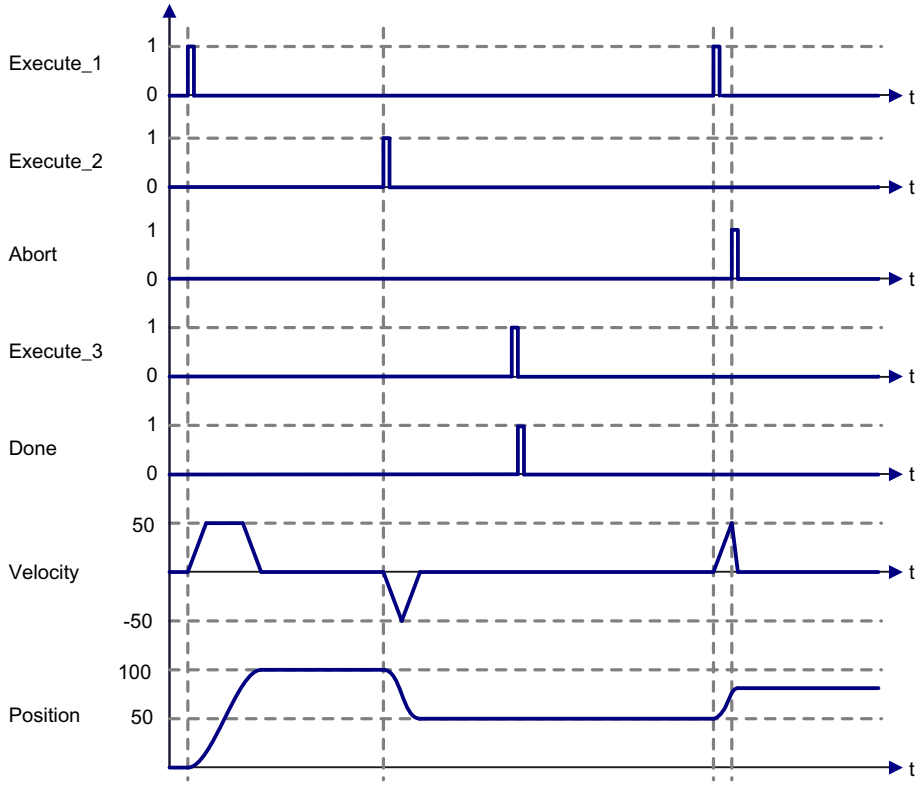


Figure 8-120 _MC_WriteBoolParameter example

8.13.4.3 MultiAxis

_MC_GearIn - Starting gearing

Overview

Schematic diagram

Purpose

Applicable for

Requirements

Input parameters

Output parameters

ErrorIDs

Examples

Schematic diagram

Schematic diagram

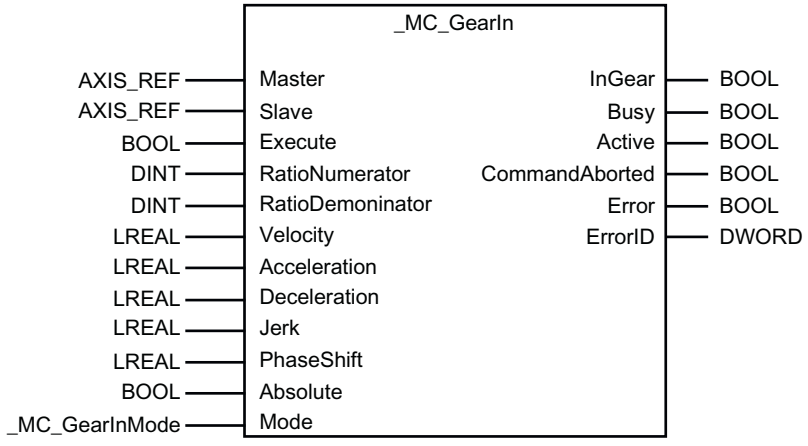


Figure 8-121 _MC_GearIn Schematic diagram

Purpose

Purpose

The function block **_MC_GearIn** starts a gearing between a master and a slave axis.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration**, and **Jerk** define the dynamic response of the slave axis during synchronization in accordance with time.

The gear ratio is specified as a fraction.

Synchronous operation may be:

- relative to the start position or synchronization position
- absolute, in relation to the axis zero

By transmitting a new _MC_GearIn command, the gear ratio can be changed during operation. This operation does not require the master or slave axis to be stopped. Transitions are executed according to specified acceleration or deceleration values.

The function block can be started when the leading axis is at a standstill, or when it is in motion.

Applicable for

Applications

Master:

- Positioning axes
- Following axes
- Path axes
- External Encoders
- ...

Slave:

- Following axes
- Path axes with synchronous operation activated

Requirements

Requirements

The master is set for use as a potential master value at the synchronous object of the slave axis.

Master and slave axes are enabled.

No **_MC_Stop** active on the slave axis

Input parameters

Input parameters

Parameters	Data type	Initial value	Description
Master	AXIS_REF	0	Specification of reference to the master (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Slave	AXIS_REF	0	Specification of reference to the slave axis (name of TO) The following technology objects can be homed: • Following axis (followingAxis data type) • Path axis with synchronous operation activated (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The slave axis is synchronized with the interconnected master with a rising edge on this input.
RatioNumerator	DINT	1	Specification of the numerator of the gear ratio
RatioDenominator	DINT	1	Specification of the denominator of the gear ratio
Velocity	LREAL	-1.0	Specification of the maximum synchronization velocity The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.velocity system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Acceleration	LREAL	-1.0	Specification of the maximum synchronization acceleration The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.positiveaccel system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).

Parameters	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum synchronization deceleration The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.neagtiveaccel system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Jerk	LREAL	-1.0	Specification of the maximum synchronization jerk The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. To activate the jerk limitation, the configuration data **SyncingMotion.smoothAbsoluteSynchronization** on the interconnected synchronous object must be set to **YES**. Otherwise the parameter specification for **Jerk** is ignored and a trapezoidal velocity profile is always used. Value > 0: The specified value is used Value = 0: Use trapezoidal velocity profile Value < 0: The preset values in the system variables userdefault.syncdynamics.positiveaccelstartjerk, userdefault.syncdynamics.positiveaccelendjerk, userdefault.syncdynamics.negativeaccelstartjerk, and userdefault.syncdynamics.negativeaccelendjerk of the interconnected synchronous object are used (see "Preassignment > Dynamic Response" dialog at synchronous object).
PhaseShift	LREAL	0.0	Specification of the master value-related phase shift during synchronous operation The phase shift is absolute when synchronous operation is reached if **Absolute** = TRUE. The specified phase shift is added to the phase offset determined by the relative relationship if **Absolute** = FALSE. Value ≠ 0: The specified value is used Value = 0: No phase shift Slave value = gear ratio x (master value − PhaseShift)

Parameters	Data type	Initial value	Description
Absolute	BOOL	TRUE	Specification of the gearing type With TRUE, gearing is absolute in relation to the axis zero for the relevant axes. A phase shift can be set via the parameter **PhaseShift**. With FALSE, gearing is relative to the start position or synchronization position. A phase shift at the **PhaseShift** input during synchronization does not result in any additional shifting of the slave position in relation to the master. However, the phase shift is accepted as an absolute shift between the master and slave (system variable gearingadjustments.master.offset).
Mode	_MC_GearInMode	USER_DEFAULT	Specification of the synchronization mode / engage mode USER_DEFAULT: With the exception of the values set at the block inputs • Absolute (userDefault.gearingSettings.typeOfGearing), • RatioNumerator (userDefault.gearingSettings.defineMode, userDefault.gearingSettings.numerator), and • RatioDenominator (userDefault.gearingSettings.defineMode, userDefault.gearingSettings.denominator) the synchronization parameters preset at the synchronous object in the "Preassignment > Gearing" (→ system variables under userdefault.gearingsettings...) and "Preassignment > Dynamic Response" (→ system variables under userdefault.syncdynamics...) dialogs are accepted. IMMEDIATELY_BY_TIME_PROFILE: Synchronization is performed immediately by time, taking account of the dynamic response values set on the function block. Synchronous operation is operated using the synchronous object settings • "Time-related synchronization" profile specification from the "Preassignment > Dynamic Response" dialog (userDefault.syncprofile.syncprofilereference parameter equals RELATE_SYNC_PROFILE_TO_TIME), • Synchronization "with immediate effect" from the "Preassignment > Gear Synchronization" dialog (parameter userDefault.gearingsettings.synchronizingmode equals IMMEDIATELY), and • "Compatibility mode" synchronization direction (userDefault.gearingsettings.synchronizingdirection parameter equals SYSTEM_DEFINED [1]).

[1] SYSTEM_DEFINED corresponds to the **shortest distance** setting, although the direction of motion is maintained while the axis is in motion.

Note

The gearing direction is preassigned by userDefault.GearingSettings.direction (Preassignment > Gearing > Direction). The direction can also be inverted, e.g. for synchronous operation in the opposite direction.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
InGear	BOOL	FALSE	Display of the synchronism of the master and slave axis With TRUE, the slave axis is in synchronous operation with the master.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6699)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

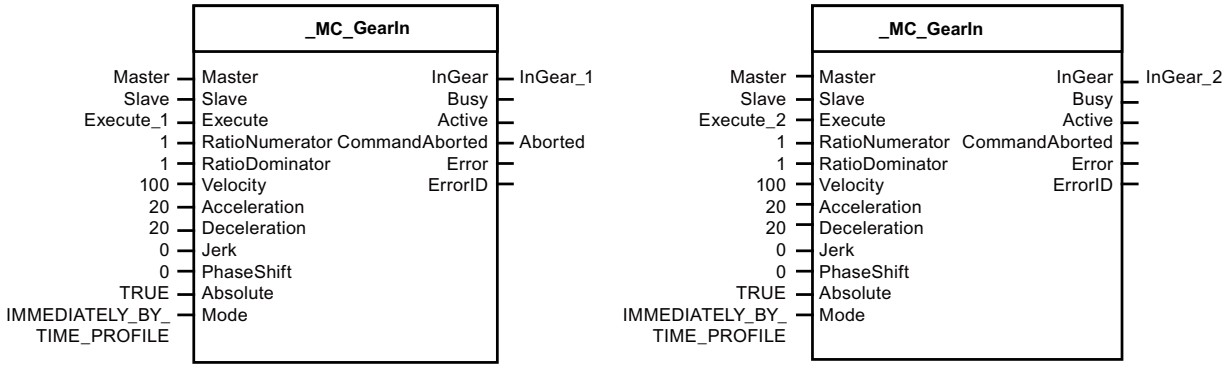
See also

Troubleshooting - PLCopen Blocks (Page 6733)

Examples

Example: Gearing

At the master axis, a positioning command is active at 100 mm. Two mutually overriding _MC_GearIn blocks with different gear ratios establish the gearing between the axes transferred on the **master** and **slave** input parameters.



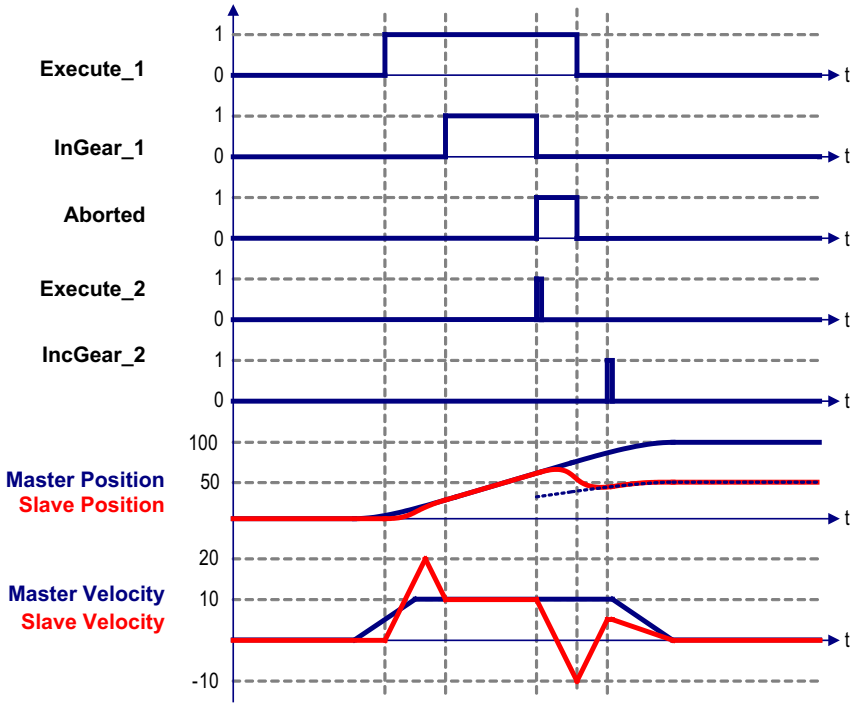
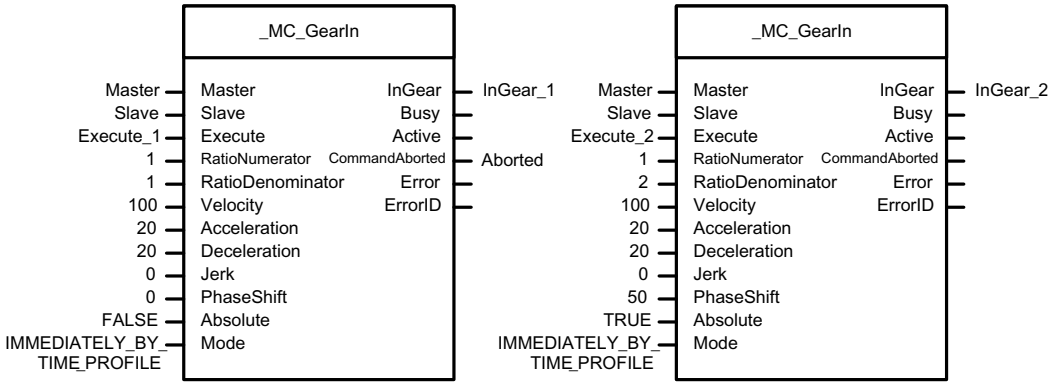


Figure 8-122 _MC_GearIn example: Gearing

Example: Gearing with gear ratio

At the master axis, a positioning command is active at 100 mm. Two mutually overriding _MC_GearIn blocks with different gear ratios establish the gearing between the axes transferred on the **master** and **slave** input parameters. While the first block is being executed, the slave position changes relative to the master position. While the second block is being executed, the slave position changes absolutely in relation to the master position, i.e. the slave position is the same as the master position evaluated using the gear ratio (RatioNumerator = 1, RatioDenominator = 2).



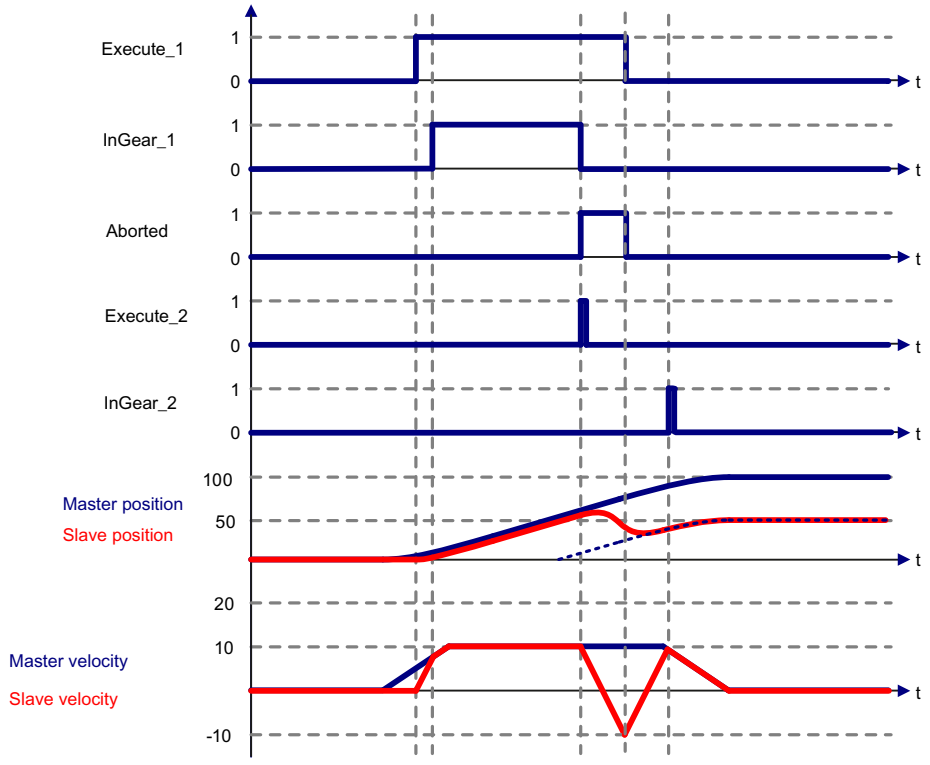


Figure 8-123 _MC_GearIn example: Gearing with gear ratio

_MC_GearOut - Terminating gearing

Overview

Schematic diagram

Schematic diagram

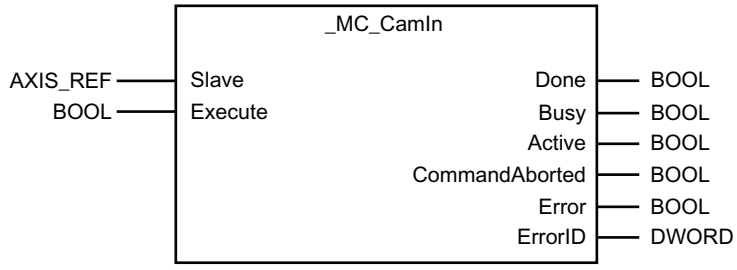


Figure 8-124 _MC_GearOut Schematic diagram

Purpose

Purpose

The function block **_MC_GearOut** terminates a gearing and stops the slave axis. The slave axis is desynchronized in accordance with the presettings at the synchronous object in the "Preassignment > Gearing" (→ system variables under **userdefault.gearingsettings...**) and "Preassignment > Dynamic Response" (→ system variables under **userdefault.syncdynamics...**) dialogs.

Recommendation

Use the function block when the shutdown procedure is to depend on the position of the master and/or the slave axis. You can also remove the slave axis from the synchronous operation with the technology functions **_MC_Stop**, **_MC_MoveRelative**, **_MC_MoveAdditive**, **_MC_MoveAbsolute** or **_MC_MoveVelocity**.

Applicable for

Applications

Following axes

Path axes with synchronous operation activated

Requirements

Requirements

A gearing must be active on the slave axis. If no synchronous operation is active, the function block is aborted.

No **_MC_Stop** active on the slave axis.

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Slave	AXIS_REF	0	Specification of reference to the slave axis (name of TO) The following technology objects can be homed: • Following axis (followingAxis data type) • Path axis with synchronous operation activated (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The synchronous operation of the slave axis with the interconnected master is terminated with a rising edge on this input.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the slave axis has been desynchronized from the interconnected master.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.

Parameter	Data type	Initial value	Description
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6704)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

At the master axis, a positioning command is active at 100 mm. The _MC_GearOut block terminates gearing of the axis transferred on the **slave** input parameter.

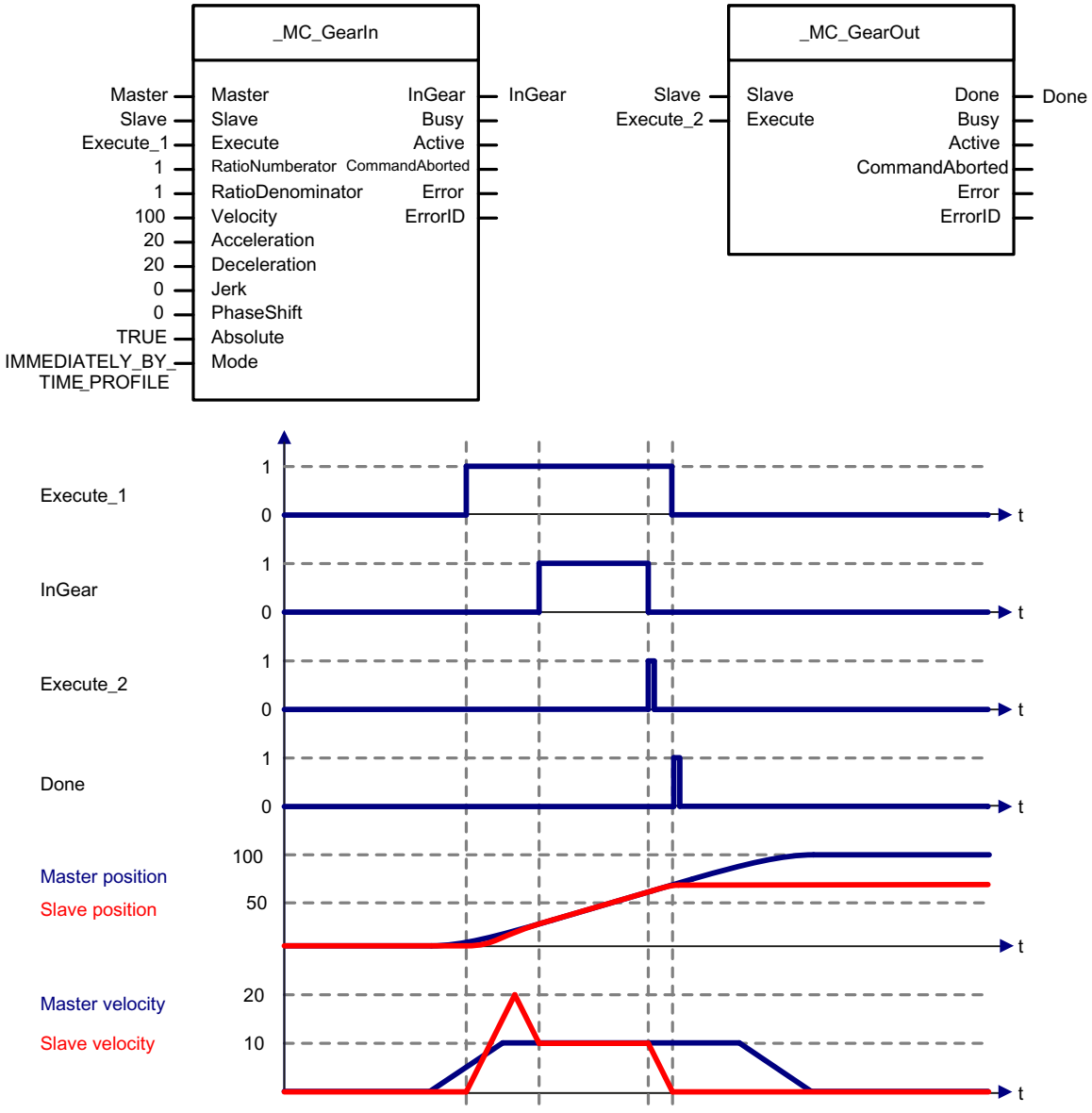


Figure 8-125 _MC_GearOut example

_MC_CamIn - Starting camming

Overview

Schematic diagram

Purpose

Applicable for

Requirements

Input parameters

Output parameters

ErrorIDs

Examples

Schematic diagram

Schematic diagram

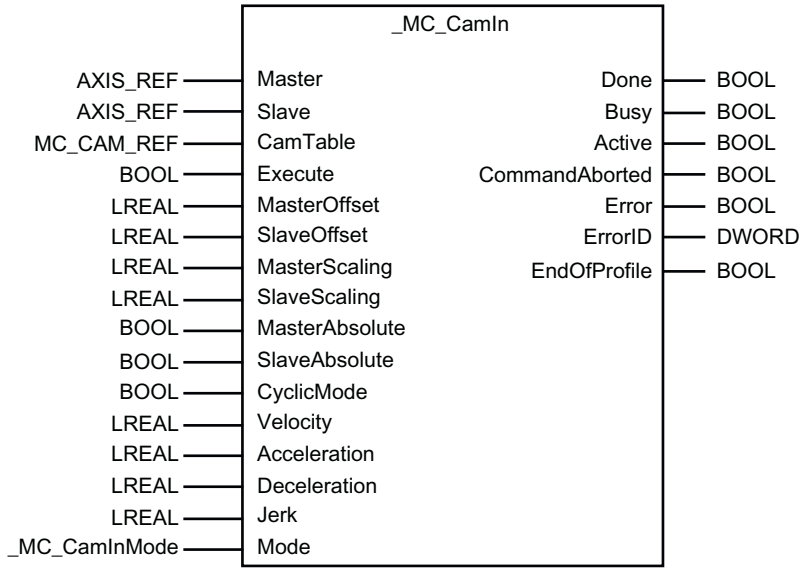


Figure 8-126 _MC_CamIn Schematic diagram

Purpose

Purpose

The function block **_MC_CamIn** starts a camming between a master and a slave axis.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration**, and **Jerk** define the dynamic response of the slave axis during synchronization in accordance with time.

The cam profile can be scaled and/or the position offset.

The specified cam can optionally be run through once or periodically.

Camming can be relative or absolute.

Applicable for

Applications

Master:

- Positioning axes
- Following axes
- Path axes
- External Encoders
- ...

Slave:

- Following axes
- Path axes with synchronous operation activated

Requirements

Requirements

When configuring the synchronous object of the slave axis, you must have selected the required cam and master as potential interconnection objects.

Master and slave axes are enabled.

No **_MC_Stop** active on the slave axis

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Master	AXIS_REF	0	Specification of reference to the master (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type) • External encoder (externalEncoderType data type)
Slave	AXIS_REF	0	Specification of reference to the slave axis (name of TO) The following technology objects can be homed: • Following axis (followingAxis data type) • Path axis with synchronous operation activated (_pathAxis data type)
CamTable	MC_CAM_REF	0	Specification of the cam (name)

Parameter	Data type	Initial value	Description
Execute	BOOL	FALSE	Function block enable The slave axis is synchronized with the interconnected master with a rising edge on this input.
MasterOffset	LREAL	0.0	Specification of the offset of the master values in the master coordinates.
SlaveOffset	LREAL	0.0	Specification of the offset of the slave values in the slave coordinates.
MasterScaling	LREAL	1.0	Specification of the scaling for the master values in the master coordinates.
SlaveScaling	LREAL	1.0	Specification of the scaling for the slave values in the slave coordinates.
MasterAbsolute	BOOL	TRUE	Specification of the evaluation method of the master values With TRUE, the master values are applied as absolute values in the domain of the cam. With FALSE, the master values are evaluated relative to the start value of the cam.
SlaveAbsolute	BOOL	TRUE	Specification of the evaluation method of the slave values With TRUE, the slave values are applied as absolute values in the range of the cam. With FALSE, the slave values are applied relative to the start value of the cam. During synchronization, the slave axis also travels the path difference between the start of the cam and the cam start value.
CyclicMode	BOOL	TRUE	Specification of the cam mode With TRUE, the cam repeats after reaching its end point. With FALSE, the function block is terminated after one cycle of the cam.
Velocity	LREAL	-1.0	Specification of the maximum synchronization velocity The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.velocity system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Acceleration	LREAL	-1.0	Specification of the maximum synchronization acceleration The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.positiveaccel system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum synchronization deceleration The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.negativeaccel system variable of the interconnected synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).

Parameter	Data type	Initial value	Description
Jerk	LREAL	-1.0	Specification of the maximum synchronization jerk The parameter is only taken into account with **Mode** equals **IMMEDIATELY_BY_TIME_PROFILE**. To activate the jerk limitation, the configuration data **SyncingMotion.smoothAbsoluteSynchronization** on the interconnected synchronous object must be set to **YES**. Otherwise the parameter specification for **Jerk** is ignored and a trapezoidal velocity profile is always used. Value > 0: The specified value is used Value = 0: Use trapezoidal velocity profile Value < 0: The preset values in the system variables userdefault.syncdynamics.positiveaccelstartjerk, userdefault.syncdynamics.positiveaccelendjerk, userdefault.syncdynamics.negativeaccelstartjerk, and userdefault.syncdynamics.negativeaccelendjerk of the interconnected synchronous object are used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Mode	_MC_CamInMode	USER_DEFAULT	Specification of the synchronization mode / engage mode USER_DEFAULT: With the exception of the values set at the block inputs • MasterAbsolute (userDefault.cammingSettings.masterMode), • SlaveAbsolute (userDefault.cammingSettings.slaveMode) and • CyclicMode (userDefault.cammingSettings.cammingMode) the synchronization parameters preset at the synchronous object in the "Preassignment > Camming" (→ system variables under userdefault.cammingsettings...) and "Preassignment > Dynamic Response" (→ system variables under userdefault.syncdynamics...) dialogs are accepted. IMMEDIATELY_BY_TIME_PROFILE: Synchronization is performed immediately according to time taking into account the dynamic response values set on the function block. Synchronous operation is performed using the synchronous object settings • "Time-related synchronization" profile specification from the "Preassignment > Dynamic Response" dialog (userDefault.syncprofile.syncprofilereference parameter equals RELATE_SYNC_PROFILE_TO_TIME), • Synchronization "with immediate effect" from the "Preassignment > Cam Synchronization" dialog (parameter userDefault.cammingsettings.synchronizingmode equals IMMEDIATELY), and • "Compatibility mode" synchronization direction (userDefault.cammingsettings.synchronizingdirection parameter equals SYSTEM_DEFINED [1]).

[1] SYSTEM_DEFINED corresponds to the **shortest distance** setting, although the direction of motion is maintained while the axis is in motion.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
InSync	BOOL	FALSE	Display of the synchronism of the master and slave axis With TRUE, the slave axis is in synchronous operation with the master.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Section ErrorIDs (Page 6711)).
EndOfProfile	Bool	FALSE	Indicates the end of the cam profile. The parameter is reset after being read out once.

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Examples

Example: Camming

At the master axis, a positioning command is active at 100 mm. Two mutually overriding _MC_CamIn blocks with different cam profiles establish the camming between the axes transferred on the **master** and **slave** input parameters.

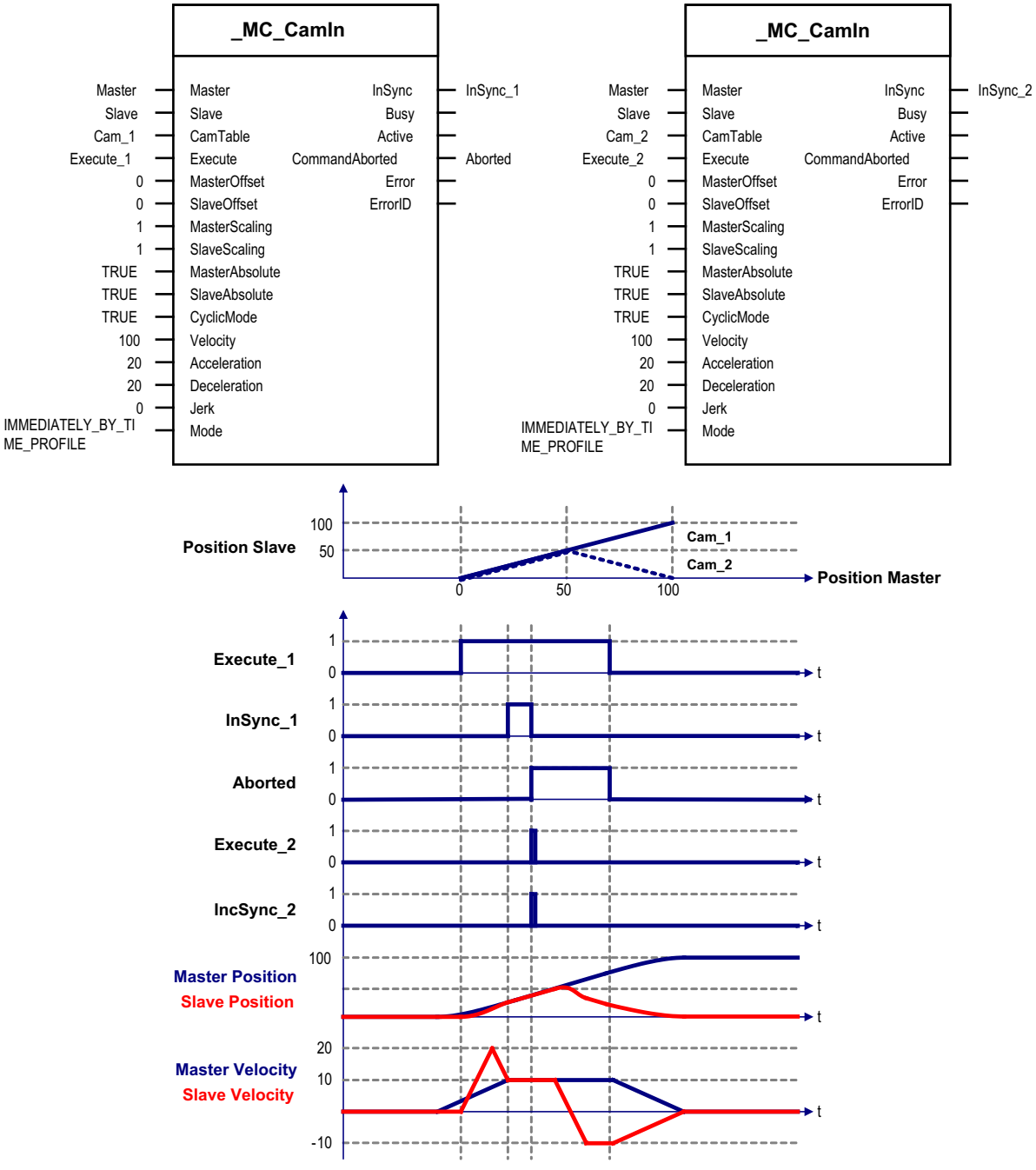


Figure 8-127 _MC_CamIn example: Camming

Example: Camming with offset

At the master axis, a positioning command is active at 100 mm. Two mutually overriding _MC_CamIn blocks with different cam profiles establish the camming between the axes transferred on the **master** and **slave** input parameters. While the first block is being executed, the master position shifts by 10 mm in relation to the slave position (MasterOffset). While the second block is being executed, the slave position shifts by 10 mm in relation to the master position (SlaveOffset).

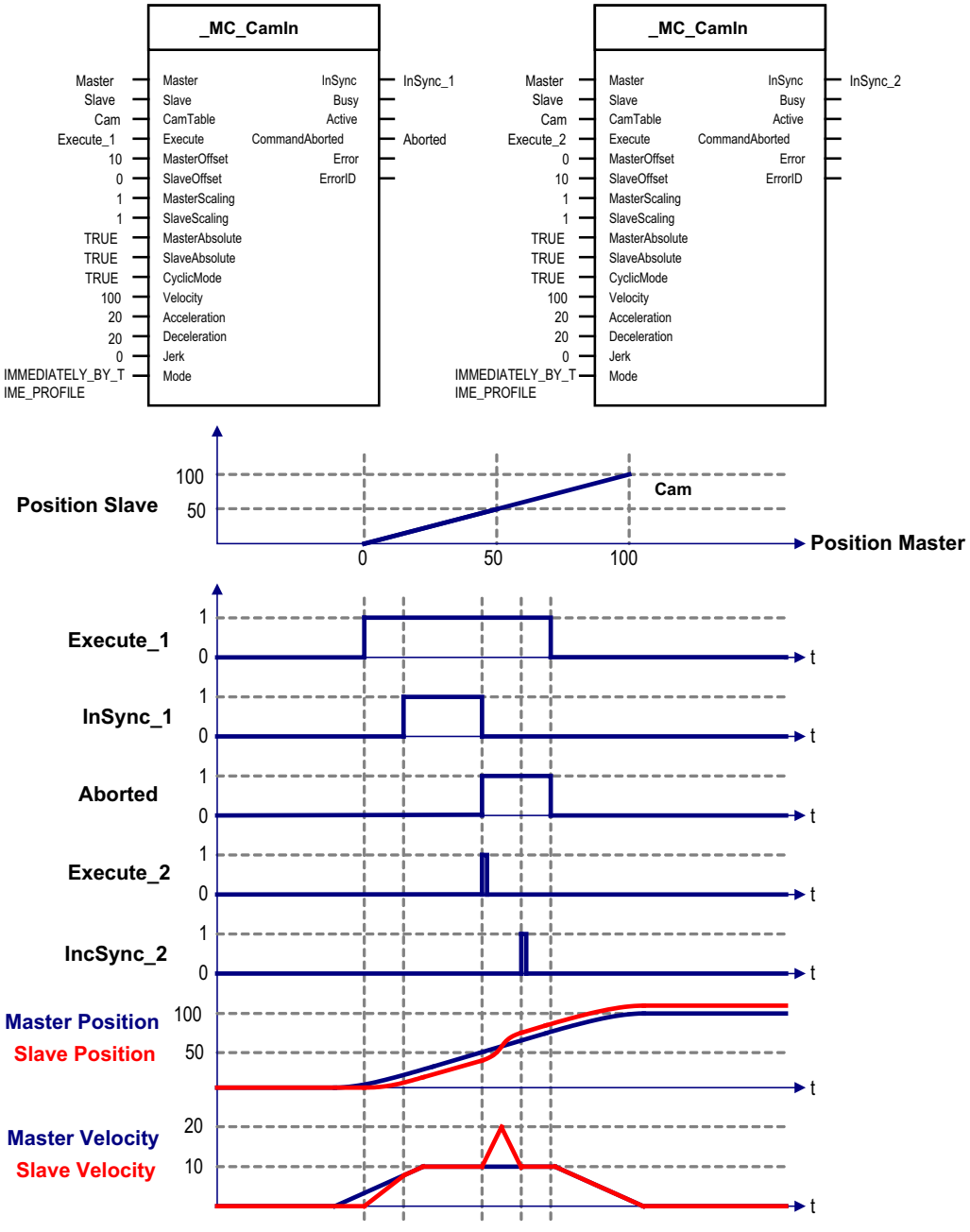


Figure 8-128 _MC_CamIn example: Camming with offset

Example: Camming with scaling

At the master axis, a positioning command is active at 100 mm. Two mutually overriding _MC_CamIn blocks with different cam profiles establish the camming between the axes transferred on the **master** and **slave** input parameters. While the first block is being executed, the cam is scaled by a factor of 2 in relation to the master position (MasterScaling). While the second block is being executed, the cam is scaled by a factor of 2 in relation to the slave position (SlaveScaling).

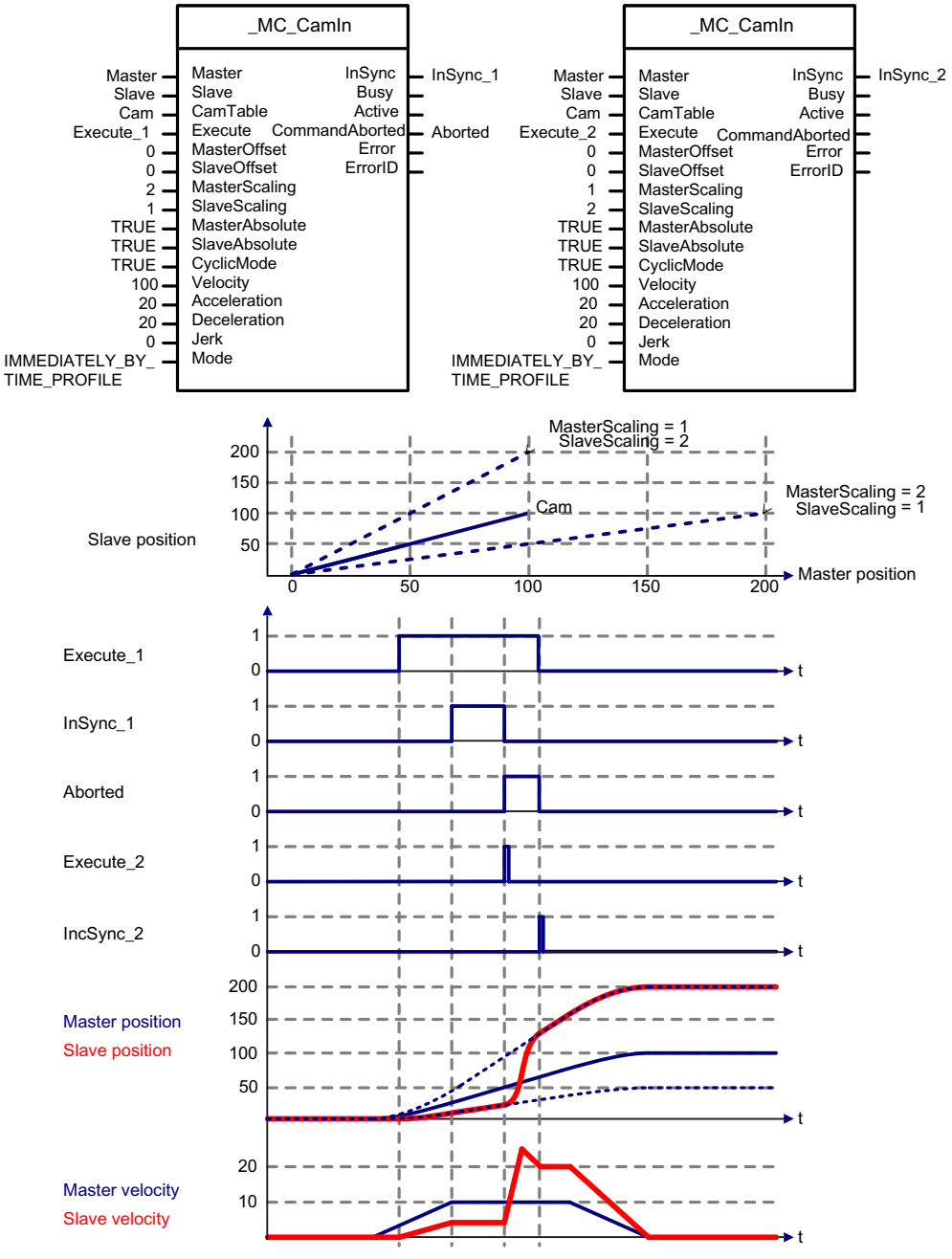


Figure 8-129 _MC_CamIn example: Camming with scaling

Example: Camming with starting point shift to actual position of slave axis

At the master axis, a positioning command is active at 100 mm. When the first block starts (Execute_1 = TRUE), the slave axis synchronizes to the master axis in accordance with the positional relationship $Pos_{Slave} = Pos_{Master}$ and then follows this axis.

When the second block starts (Execute_2 = TRUE), the cam starting point is shifted to the actual position of the slave axis (Pos_{Slave} = 48 mm) (→ interpretation of slave position relative to cam starting point). The cam does not change in relation to the master position (→ interpretation of master position is absolute in the cam's range of values). Therefore, for the second block the equation $Pos_{Slave} = Pos_{Master}$ + 48 mm applies as the positional relationship.

In the figure below, in accordance with this positional relationship the position setpoint of the slave axis is shifted by 48 mm in relation to the master position. Once the master position setpoint of 100 mm has been reached, the result is a position setpoint of 148 mm (100 mm + 48 mm = 148 mm) for the slave axis.

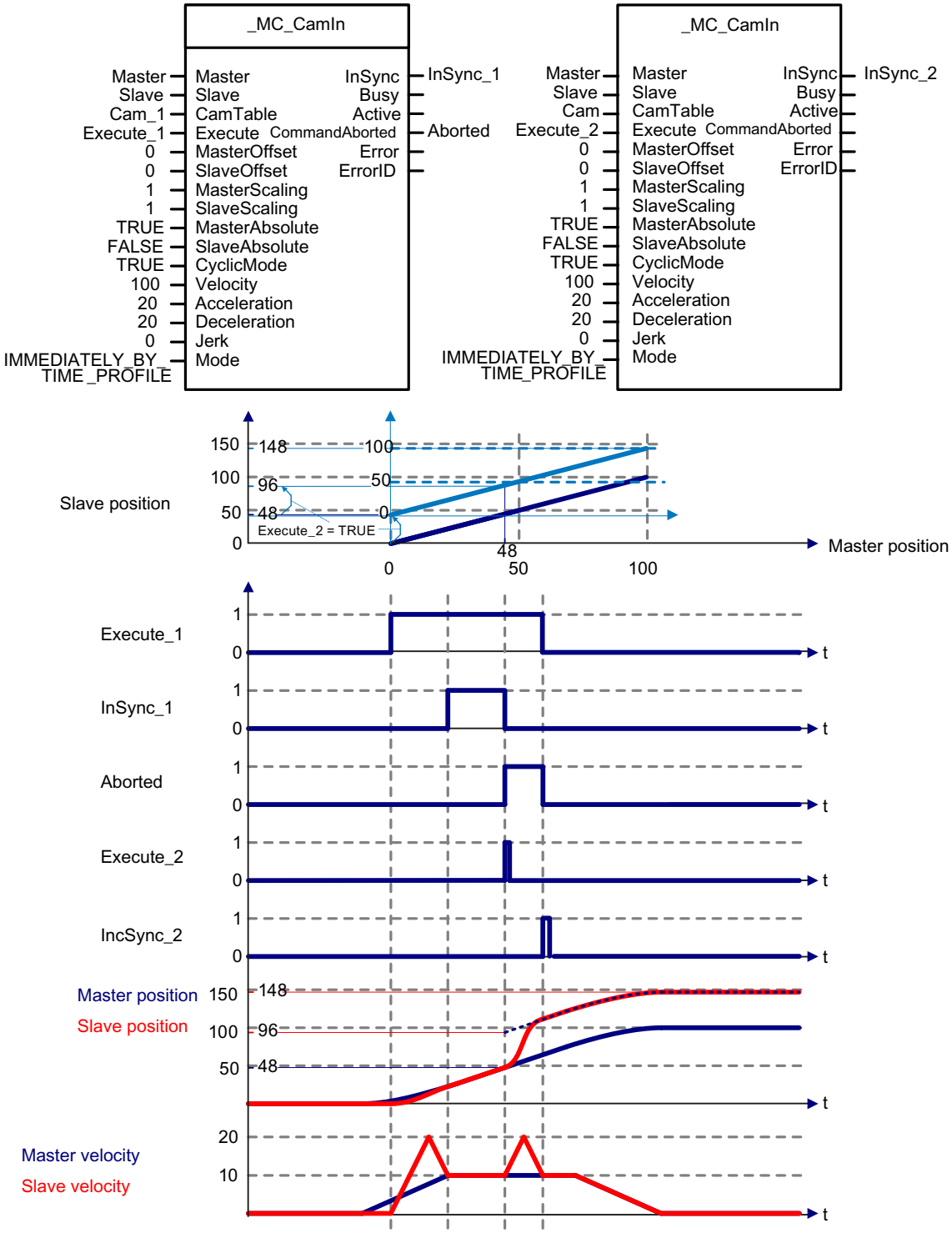


Figure 8-130 _MC_CamIn example: Camming with starting point shift to actual position of slave axis

Example: Camming with starting point shift to actual position of master axis

At the master axis, a positioning command is active at 100 mm. When the first block starts (Execute_1 = TRUE), the cam starting point is shifted to the actual position of the master axis (Pos_{Master} = 2 mm) (→ interpretation of master position relative to cam starting point). For this reason, the positional relationship between the slave and master positions must be corrected as follows:
(PosSlave = PosMaster) ⇒ $Pos_{Slave} = Pos_{Master} – 2$ mm.

This results in a 2 mm shift of the slave position in relation to the master position that is permanent pending further notice.

When the second block is started (Execute_2 = TRUE), the cam starting point is shifted repeatedly to the actual position of the master axis (Pos_{Master} = 48 mm). Therefore, the following equation applies for the second block:
($Pos_{Slave} = Pos_{Master} – 2$ mm) ⇒ $Pos_{Slave} = Pos_{Master} – 48$ mm

In accordance with the positional relationship the position setpoint of the slave axis reset to zero, and in the figure below, is shifted by 48 mm in relation to the master position. Once the master position setpoint of 100 mm has been reached, the result is a position setpoint of 52 mm (100 mm – 48 mm = 52 mm) for the slave axis.

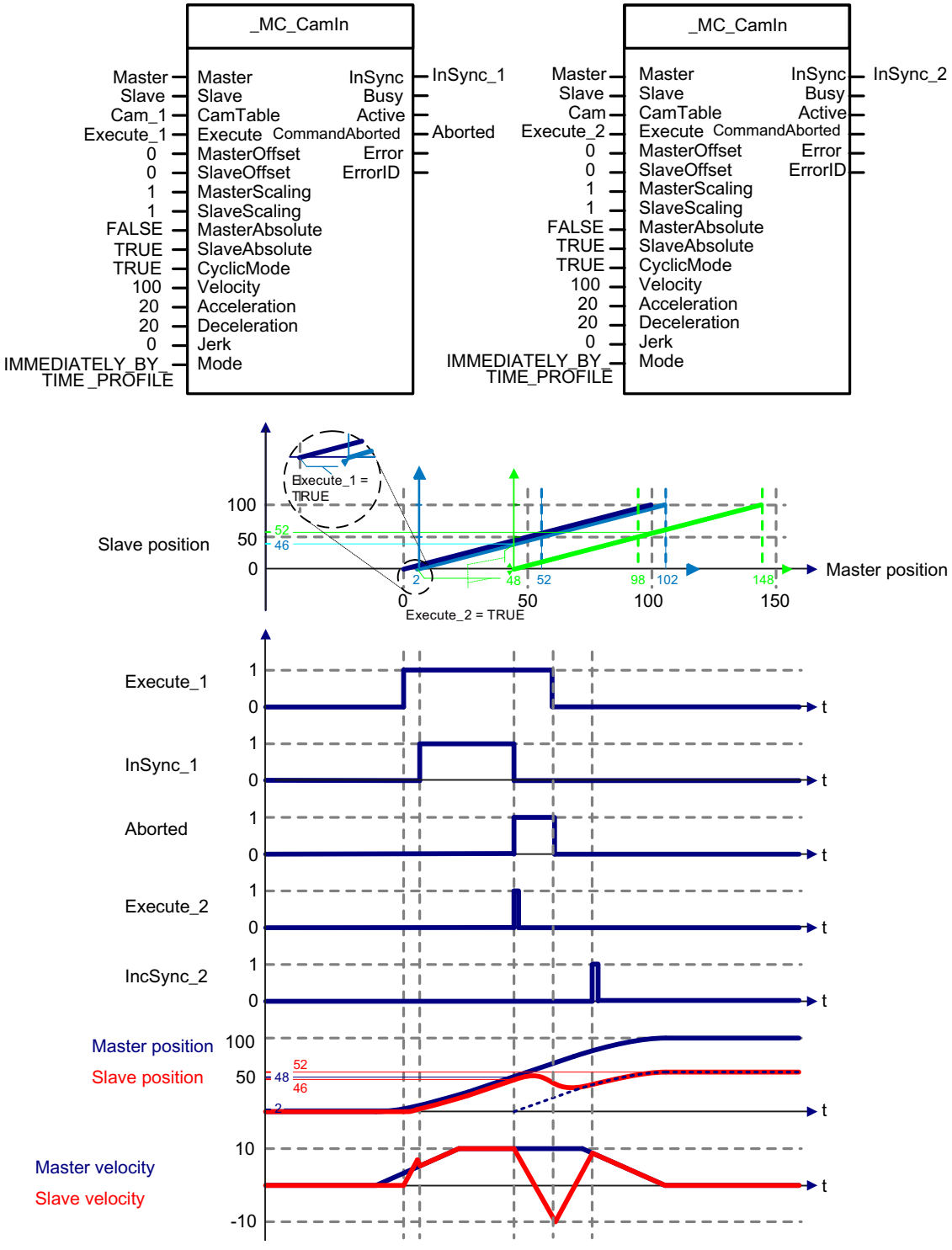


Figure 8-131 _MC_CamIn example: Camming with starting point shift to actual position of master axis

_MC_CamOut - Terminating camming

Overview

Schematic diagram

Schematic diagram

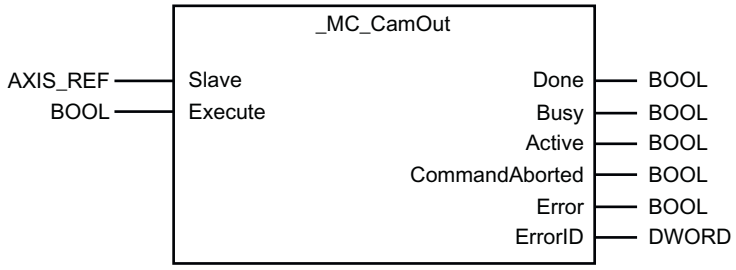


Figure 8-132 _MC_CamOut Schematic diagram

Purpose

Purpose

The function block **_MC_CamOut** terminates a camming and stops the slave axis. The slave axis is desynchronized in accordance with the presettings at the synchronous object in the "Preassignment > Camming" (→ system variables under **userdefault.cammingsettings...**) and "Preassignment > Dynamic Response" (→ system variables under **userdefault.syncdynamics...**) dialogs.

Recommendation

Use the function block when the shutdown procedure is to depend on the position of the master and/or the slave axis. You can also remove the slave axis from the synchronous operation with the technology functions **_MC_Stop**, **_MC_MoveRelative**, **_MC_MoveAdditive**, **_MC_MoveAbsolute** or **_MC_MoveVelocity**.

Applicable for

Applications

Following axes

Path axes with synchronous operation activated

Requirements

Requirements

A camming must be active on the slave axis. If no synchronous operation is active, the function block is aborted.

No **_MC_Stop** active on the slave axis

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Slave	AXIS_REF	0	Specification of reference to the slave axis (name of TO) The following technology objects can be homed: • Following axis (followingAxis data type) • Path axis with synchronous operation activated (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable The synchronous operation of the slave axis with the interconnected master is terminated with a rising edge on this input.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Display of the completion of the function block With TRUE, the slave axis has been desynchronized from the interconnected master.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.

Parameter	Data type	Initial value	Description
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6721)).

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

At the master axis, a positioning command is active at 100 mm. The _MC_CamOut block terminates camming of the axis transferred on the **slave** input parameter.

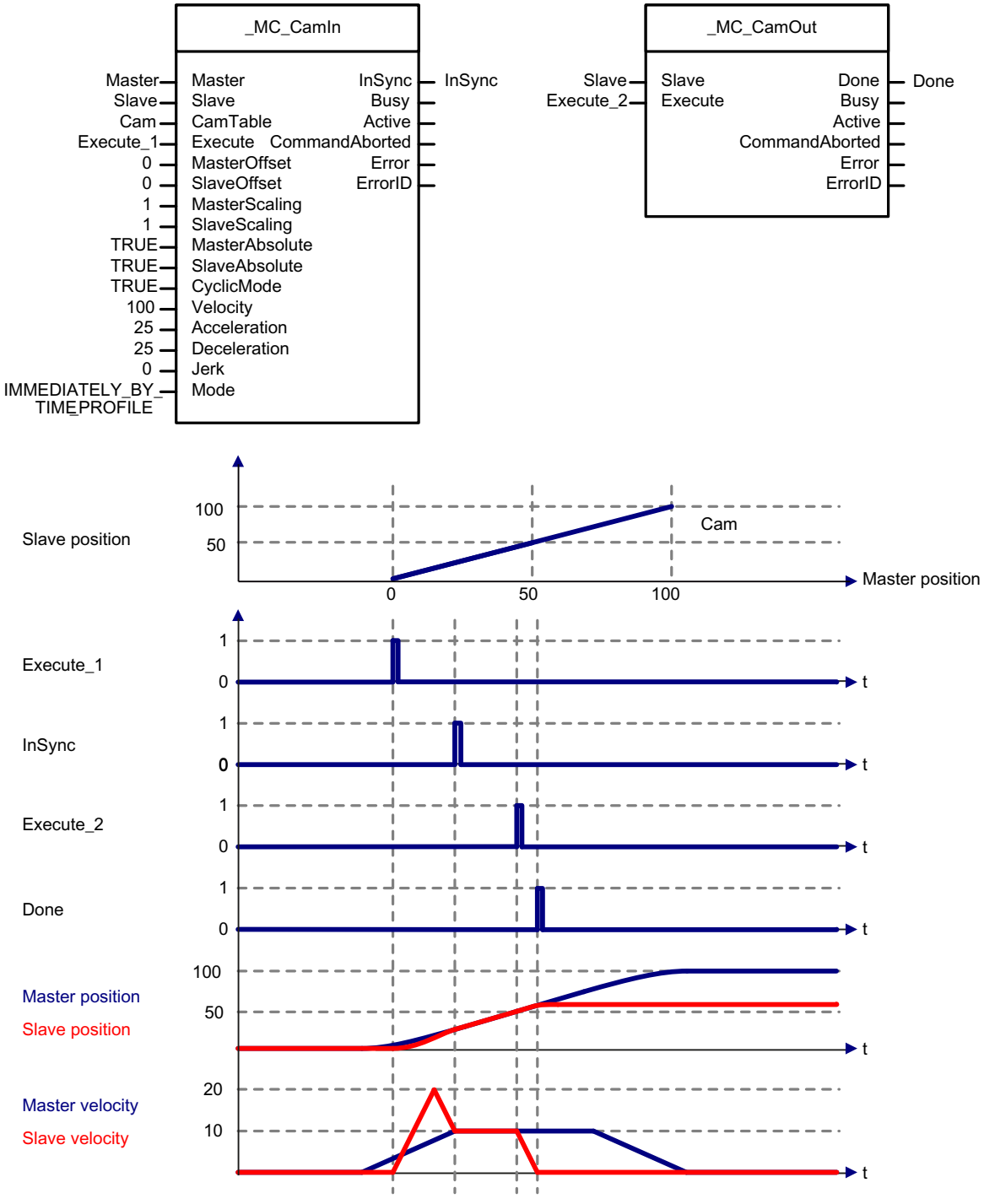


Figure 8-133 _MC_CamOut example:

_MC_Phasing - Changing phase shift between the leading axis and following axis

Overview

Schematic diagram

Schematic diagram

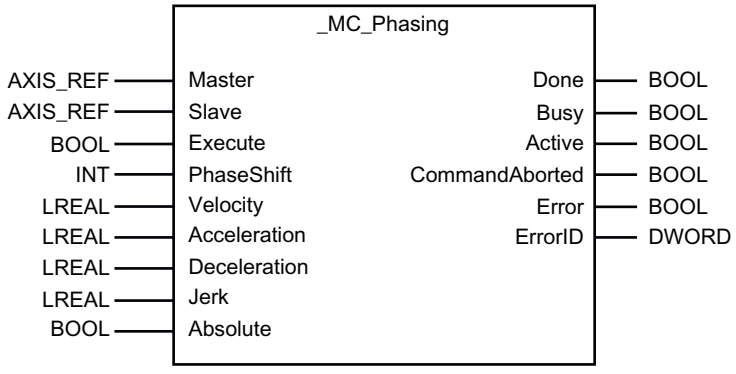


Figure 8-134 _MC_Phasing Schematic diagram

Purpose

Purpose

The function block **_MC_Phasing** shifts the position of the slave axis with respect to the master.

The effect on a following axis during camming equates to a horizontal cam shift.

The shift can be absolute or relative to the existing offsets.

The dynamic response parameters **Velocity**, **Acceleration**, **Deceleration** and **Jerk** define the dynamic response of the motion procedure.

Applicable for

Applications

Following axes operating during camming or gearing

Path axes with synchronous operation activated during camming or gearing

Requirements

Requirements

No **_MC_Stop** active on the slave axis

A camming or gearing must be active on the slave axis.

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Master	AXIS_REF	0	Specification of reference to the master (name of TO) This parameter presently has no function. The slave axis is always offset with respect to the current master of the active synchronous operation.
Slave	AXIS_REF	0	Specification of reference to the slave axis (name of TO) The following technology objects can be homed: • Following axis (followingAxis data type) • Path axis with synchronous operation activated (_pathAxis data type)
Execute	BOOL	FALSE	Function block enable With a rising edge on this input, the position of the slave axis is offset with respect to the position of the master.
PhaseShift	LREAL	1.0	Specification of the phase shift
Velocity	LREAL	-1.0	Specification of the maximum velocity Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.velocity system variable of the synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Acceleration	LREAL	-1.0	Specification of the maximum acceleration Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.positiveaccel system variable of the synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).

Parameter	Data type	Initial value	Description
Deceleration	LREAL	-1.0	Specification of the maximum deceleration Value > 0: The specified value is used Value = 0: Not permissible Value < 0: The preset value in the userdefault.syncdynamics.negativeaccel system variable of the synchronous object is used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Jerk	LREAL	-1.0	Specification of the maximum jerk Value > 0: The specified value is used Value = 0: Use trapezoidal velocity profile Value < 0: The preset values in the system variables userdefault.syncdynamics.positiveaccelstartjerk, userdefault.syncdynamics.positiveaccelendjerk, userdefault.syncdynamics.negativeaccelstartjerk, and userdefault.syncdynamics.negativeaccelendjerk of the synchronous object are used (see "Preassignment > Dynamic Response" dialog at synchronous object).
Absolute	BOOL	TRUE	Indicates the type of phase shift With TRUE, the phase shift is taken over as an absolute value. With FALSE, the phase shift is added to an existing offset.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Indicates termination of the function block. With TRUE, the master and slave axis have been displaced with respect to one another by the absolute value of the phase shift.
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started.
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is processed by the command execution, i.e. the function block has active control over the axis / technology object. The command is processed in the interpolator.
CommandAborted	BOOL	FALSE	Display of the abort of the function block With TRUE, the function block has been aborted because of another overriding function block or TO command.
Error	BOOL	FALSE	Display of an error in the function block In the case of TRUE, an error occurred during parameterization or execution of the function block. The function block is terminated. The error description can be read at the ErrorID output.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **Error** output (see Chapter ErrorIDs (Page 6726)).

ErrorIDs

ErrorIDs

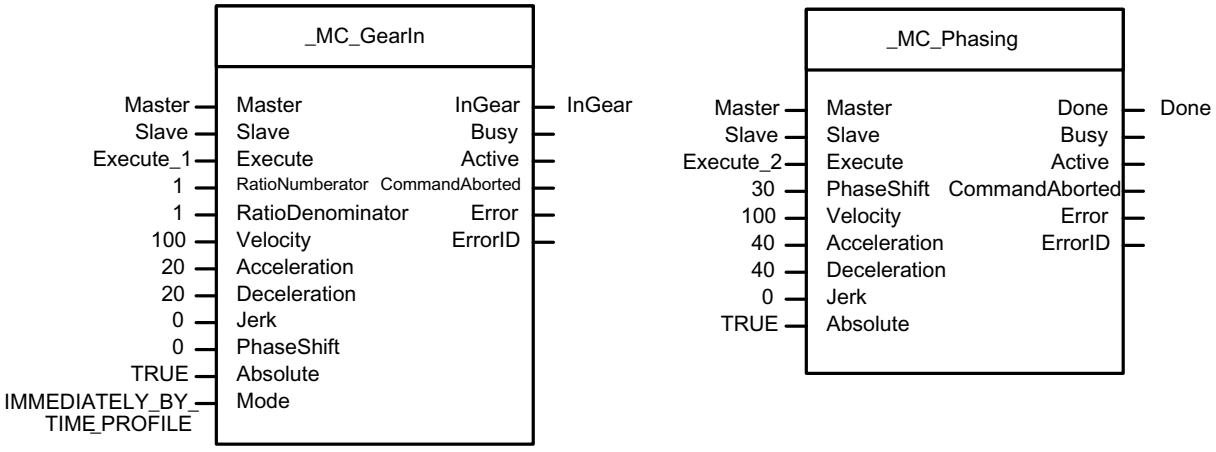
The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

See also

Troubleshooting - PLCopen Blocks (Page 6733)

Example

At the master axis, a positioning command is active at 100 mm. The _MC_Phasing block shifts the position of the slave following axis transferred on the **slave** input parameter.



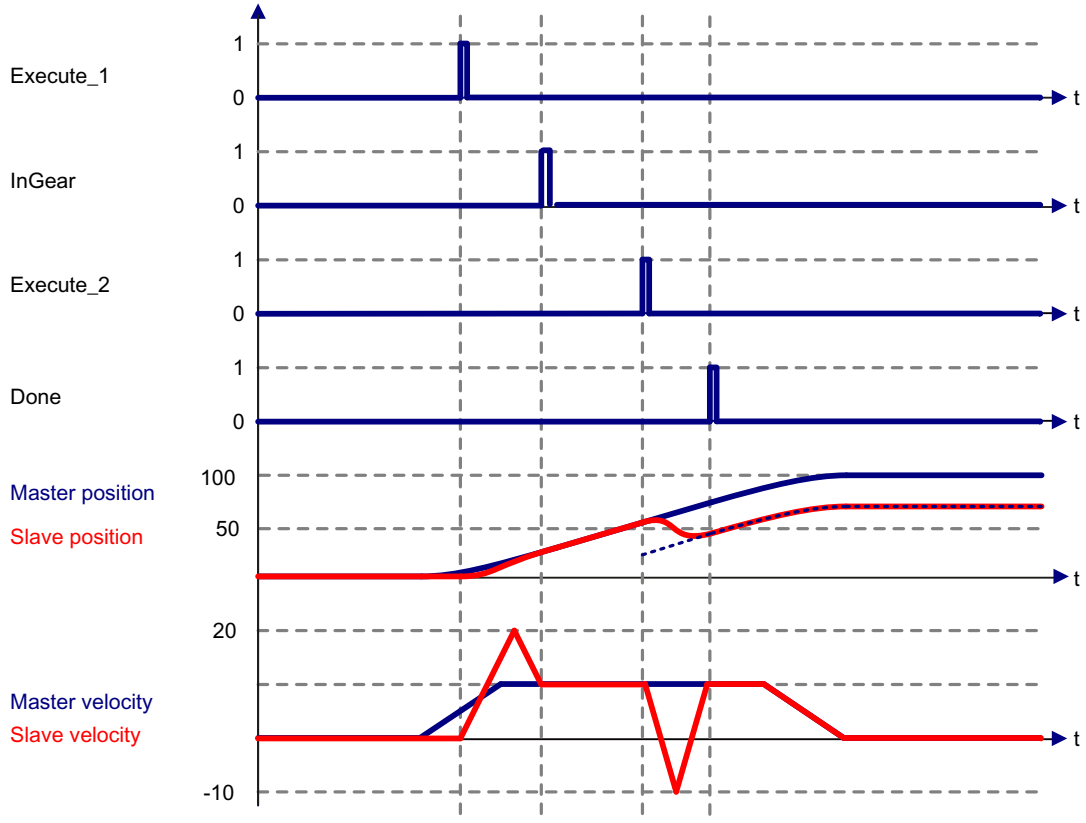


Figure 8-135 _MC_Phasing example

8.13.4.4 Advanced functions

_MC_Jog - Continuous or incremental jogging

Overview

Schematic diagram

Schematic diagram

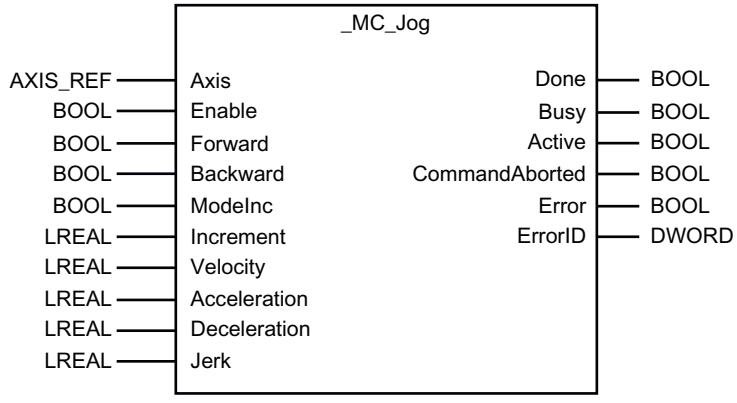


Figure 8-136 _MC_Jog Schematic diagram

Purpose

Purpose

The function block **_MC_Jog** implements a continuous or incremental jogging of an axis.

Applicable for

Applications

Positioning axes

Following axes

Path axes

Requirements

Requirements

Axis enabled

No **_MC_Stop** active

Input parameters

Input parameters

Parameter	Data type	Initial value	Description
Axis	AXIS_REF	0	Specification of axis reference (name of TO) The following technology objects can be homed: • Positioning axis (posAxis data type) • Following axis (followingAxis data type) • Path axis (_pathAxis data type)
Enable	BOOL	FALSE	Function block enable
Forward	BOOL	FALSE	Start of the traversing motion in positive direction with positive edge Stop of the traversing motion with negative edge TRUE = motion in positive direction as long as **Forward** is set
Backward	BOOL	FALSE	Start of the traversing motion in negative direction with positive edge Stop of the traversing motion with negative edge TRUE = motion in negative direction as long as **Backward** is set
ModeInc	BOOL	FALSE	FALSE = continuous jogging TRUE = incremental jogging Each status change on the **ModeInc** results in an axis stop and deletion of the distance-to-go
Increment	LREAL	0.0	Value for incremental jogging
Velocity	LREAL	-1.0	Maximum velocity that is reached depending on the set traversing distance, acceleration and jerk (is not always reached) Value > 0: Use the specified value Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.velocity system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).

Parameter	Data type	Initial value	Description
Acceleration	LREAL	-1.0	Acceleration (increasing energy in the motor): Value > 0: Use the specified value Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.positiveaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Deceleration	LREAL	-1.0	Deceleration (decreasing energy in the motor): Value > 0: Use the specified value Value = 0: Not permissible Value < 0: The preset value in the userdefaultdynamics.negativeaccel system variable of the axis is used (see "Preassignment > Dynamic Response" axis dialog).
Jerk	LREAL	-1.0	Specification of maximum jerk in conjunction with velocity profile definition Value > 0: Acceleration-constant velocity profile (SMOOTH); specified jerk is used Value = 0: Trapezoidal velocity profile (TRAPEZOIDAL) Value < 0: Type of velocity profile results from setting of axis system variable userdefaultdynamics.profile In the case of an effective, acceleration-constant velocity profile (SMOOTH), the preset values in the userdefaultdynamics.positiveaccelstartjerk, userdefaultdynamics.positiveaccelendjerk, userdefaultdynamics.negativeaccelstartjerk, and userdefaultdynamics.negativeaccelendjerk system variables are used. For information on the effects these have, refer to the "TO Axis Electric / Hydraulic, External Encoder" Function Manual; Command variable calculation > Velocity profiles.

Output parameter

Output parameter

Parameter	Data type	Initial value	Description
Done	BOOL	FALSE	Incremental or continuous jogging finished, the axis has been stopped
Busy	BOOL	FALSE	Display of the activity of the function block With TRUE, the function block has been started, distance-to-go pending
Active	BOOL	FALSE	Display of the command activity in the function block With TRUE, the command is being processed by the command processing, i.e., the function block has active control of the axis.
CommandAborted	BOOL	FALSE	Command has been aborted by another command or by an error during the command processing.
Error	BOOL	FALSE	Indicates an error **An error** has occurred during the command execution. The cause can be found in the **ErrorID**.
ErrorID	DWORD	0	Display of a function block error code The error code is always output in conjunction with the **CommandAborted** or **Error** output (see Chapter ErrorIDs (Page 6733)).

Function

Function

The function block traverses the axis with the parameterized dynamic response values either continuously or incrementally in a positive or negative direction.

A positive edge at the input parameters **Forward** or **Backward** starts the traversing motion; it is stopped with the negative edge.

When incremental jogging is selected, the axis is stopped by the function block after traversing the parameterized distance. If the axle is stopped before reaching the parameterized distance, a restart of the motion completes the residual distance.

When incremental jogging is selected, the direction can only be changed when there is no remaining distance to be traversed.

- **Enable**
  - **Enable = TRUE**: Functionality of the block is processed and traversing motions are possible; with the transition FALSE/TRUE at Enable, all output parameters are reset and the distance-to-go is deleted.
    Note: Busy is only set with an action.
  - **Enable = FALSE**: Stop of the traversing motion - traversing motions no longer possible; with the transition from Enable = TRUE to Enable = FALSE, the **_MC_Stop** is always called internally, irrespective of whether the axis is in motion or not.
- **Busy/Active**
  - Incremental jogging:
    Busy = TRUE / Active = TRUE: Axis moves, distance-to-go != 0
    Busy = TRUE / Active = FALSE: Intermediate stop - the axis is stationary, distance-to-go != 0
  - Continuous jogging:
    After a positive edge at the input parameter **Forward** or **Backward**, **Busy = TRUE** is output in the same cycle. **Active** is only output in the interpolator after activation of the motion command. This may occur several cycles later.
- **Continuous jogging**
  - The jogging is started with a positive edge at **Forward** or **Backward**. The axis traverses with continuous jogging as long as the level on one of the two parameters is TRUE. The motion is stopped with the falling edge.
  - If the two inputs **Forward** and **Backward** are set to TRUE at the same time, no axis motion is started (**Active** = FALSE, **Busy** = FALSE, **Done** = FALSE), the distance-to-go is deleted and no error message is generated.
  - If the input parameter for the opposite direction is also set during the traversing motion, the traversing motion is stopped, the distance-to-go is deleted and also no error message is generated.
  - A positive edge at **Forward** and a negative edge at **Backward** in the same cycle results in a change of direction. The same applies for the reverse situation.

- **Incremental jogging**
  - With incremental jogging, the traversing motion is stopped when the distance parameterized in the input parameter **Increment** is reached. If a falling edge is detected at the input parameter **Forward** (or **Backward**) before this, the traversing motion is also stopped.
  - Behavior with distance-to-go present:
    If a distance-to-go is present, i.e., the traversing motion was stopped before reaching the distance parameterized at the input parameter **Increment** by a falling edge at **Forward** or **Backward**, this state is signaled with **Busy** = TRUE, **Done** = FALSE, **Active** = FALSE.
    With another positive edge in the same direction of motion, the present distance-to-go is traversed.
    Incremental jogging in the opposite direction with a stopped axis and distance-to-go present is not possible. To do this, the distance-to-go present must be deleted with selection of continuous jogging or **Enable** = 0/1 and incremental jogging selected again.
  - Behavior when distance-to-go is zero:
    The transition **Busy** = TRUE to FALSE, **Active** = TRUE to FALSE and **Done** = FALSE to TRUE signals the complete traversing of the selected increment. The distance-to-go is 0.
- **Switchover from incremental / continuous jogging**
  A change of **ModeInc** during traversing with an active traversing motion results in an axis stop without error message. If the input parameter **ModeInc** is changed to continuous jogging (**ModeInc**=FALSE), the distance-to-go is deleted.
- **Override**
  If the **Override** = 0 with active, started traversing motion, then there is no traversing motion and **Active** is TRUE.
- **Parameter acceptance**
  The parameter acceptance of the distances/**Increment** and the dynamic response parameters (**Velocity**, **Acceleration**, ...) is performed with a rising edge of **Forward**/**Backward**, i.e., a change of **Increment**, **Velocity**, **Acceleration**, **Deceleration** or **Jerk** has no effect during traversing.
- **Axis not in position control**
  If jogging is issued and the axis is not in position control, the corresponding error message of the used PLCopen FB is output.

ErrorIDs

ErrorIDs

The error code contains the number and, when available, the associated reason for the error that has occurred in the function block. The error number occupies the lower 16 bits of the error code (see Error codes of the errorID (LOW word) (Page 6734)). The error reason, when available, is also coded as a number and occupies the upper 16 bits of the error code (see Command abort reason of the errorID (HIGH word) (Page 6736)).

Note

ErrorIDs from other blocks possible

During execution of the function block **_MC_Jog**, the function blocks **_MC_MoveVelocity**, **_MC_MoveRelative** and **_MC_Stop** are called internally.
If the **_MC_Jog** is aborted, errorIDs can therefore also be reported from these three function blocks.

See also

Troubleshooting - PLCopen Blocks (Page 6733)

8.13.5 Troubleshooting PLCopen blocks

8.13.5.1 Troubleshooting - PLCopen Blocks

The function blocks display errors that occurred at the output parameters Error and ErrorID , in the case of CommandAborted the execution was terminated prematurely.

Errors at output parameters Error or ErrorID

If output parameter **Error** = TRUE, this means that the function block was unable to initiate the command. The cause of the error is indicated by the value at output parameter **ErrorID**.

The error codes of the function blocks (errorID) have been taken over from the return values of the ST commands in SIMOTION and from the command abort reason. The return value of the ST commands is written to the LOW word of the errorID. The command abort reason is in the HIGH word of the errorID.

If the function block indicates an **Error due to incorrect parameterization**, you need to call it either using the correct parameters or at a different time (provided that this function is permitted).

It is neither required nor possible to acknowledge the errors or warnings. The error remains active until the input parameters Enable or Execute have been reset.

If **Error with abort function due to an error in the technology object** is displayed at the function block, the TO error must be acknowledged prior to calling the function block again.

The error remains active until the Execute input parameters have been reset.

Warnings and notes at the technology object, which do not cause the function to abort are not displayed at the Error and ErrorID output.

Errors at the technology object can be determined via the TO system variables error and errorGroup. In addition, the error status of the technology object can be queried via the _MC_ReadAxisError function block (see section "Query of general errors with the _MC_ReadAxisError function block").

Example - error handing in the case of following error:

Axis motion is started by means of the _MC_MoveAbsolute function block.

- The Busy and Active output parameters are TRUE

A following error is detected in the technology object.

- The TO error 50102 is output (default error response RELEASE_DISABLE)
- The Busy and Active output parameters are FALSE
- The Error output parameter is TRUE
- Output parameter ErrorID = 16#00050003
  displays "Command aborted" with the cause "Abort by a pending error response"
- The Error system variable is TRUE and Bit 8 is set in ErrorGroup (FollowingError)

Indications at output parameter CommandAborted

If the function block was aborted by another overriding command (function block or TO command), this is displayed at the CommandAborted output parameter.

Acknowledging errors at the technology object

1. Acknowledge all errors.
   To do this, first remove the causes of the errors and then acknowledge them using **_MC_Reset** (**Restart** = FALSE).
2. You can then re-enable the technology object using the **_MC_Power** function.

8.13.5.2 Error codes of the errorID (LOW word)

Error codes of the errorID (LOW word)

Error code		Description
Hex	dec	
0	0	No error
1	1	Illegal command parameter

Error code		Description
2	2	Illegal range specification in command parameters
3	3	Command aborted
4	4	Unknown command
5	5	Command cannot be executed due to current object state
6	6	Command aborted due to termination of user task
7	7	Command rejected due to suspension of command interpretation of the addressed technology object
8	8	Command aborted due to full command buffer
9	9	Insufficient memory
A	10	A connection to a technology object required for this operation does not exist
B	11	No object configuration
C	12	The error to be reset cannot be reset due to its configuration
D	13	Axis is not homed
E	14	Measuring job on virtual axis not possible
F	15	Ambiguous **commandId**
10	16	Command not implemented
11	17	Read access denied
12	18	Write access denied
13	19	Command argument not supported
14	20	The cam has already been interpolated and cannot be manipulated
15	21	The interpolation condition was violated
16	22	The programmed jerk is 0
17	23	The alarm to be deleted is not pending
18	24	Command not possible on a virtual axis
19	25	A synchronized start of this command is not possible
1a	26	Superimposed command has been aborted as it is not permitted by the active command
1b	27	Time-out during communication with the drive
1c	28	Actual values are not valid
1d	29	The command cannot be executed when velocity control is active
1e	30	The command cannot be executed when position control is active
1f	31	The command cannot be executed in torque-reduced operation or during travel to fixed end stop
20	32	The command can only be executed when force or pressure control is active
21	33	The command cannot be executed when force/pressure control is active
22	34	The command can only be executed when pressure limiting is active
23	35	Master values are not valid
24	36	Slave values are not valid
25	37	No slave value can be defined for a master value
26	38	No master value can be defined for a slave value
27	39	The command cannot be executed when synchronous operation is inactive
28	40	The command cannot be executed with non-synchronous operation
29	41	The command cannot be executed when gearing or camming is active

Error code		Description
2a	42	The command cannot be executed when camming is inactive
2b	43	This command can only be used for an interpolated cam
2c	44	The command can only not be executed when pressure limiting is active
2d	45	Insufficient interpolation points are available to interpolate the cam
2e	46	The specified path point cannot be reached due to restrictions of the kinematics
2f	47	Path axis values are not valid
30	48	Reserved memory is full
2710	10000	(Greater than or equal to) internal error

See also

Troubleshooting - PLCopen Blocks (Page 6733)

8.13.5.3 Command abort reason of the errorID (HIGH word)

Command abort reason of the errorID (HIGH word)

Command abort reason		Description
Hex	dec	
0	0	No abort reason
1	1	Reset of the command buffer
2	2	Abort by another command
3	3	Abort by a stop
4	4	Abort by a higher-order stop
5	5	Abort by a pending error response
6	6	Abort due to ambiguous **commandId**
7	7	Acknowledgement delay
8	8	No actual value for axis (e.g., encoder or data bus not ready)
9	9	Abort due to abort of a dependent command
a	10	Abort due to active synchronous operation
b	11	Abort due to active superimposed motion
c	12	Abort due to active speed-controlled controller mode
d	13	Abort due to active position-controlled controller mode
e	14	Abort due to active travel to fixed end stop
f	15	Axis is not in pressure-limiting operation
10	16	Abort due to active pressure-controlled operation
11	17	Abort due to inactive pressure-controlled operation
12	18	Superimposed command is not permitted
13	19	Abort due to error during cam access
14	20	Slave is not ready for operation

Command abort reason		Description
15	21	Error in the slave synchronization
16	22	Abort by command on the slave
17	23	Abort by stop on the slave
18	24	Abort by a pending error response on the slave
19	25	No actual values for slave (e.g., encoder or data bus not ready)
1a	26	Abort by reset on the slave
1b	27	Master values are not valid
1c	28	Active command in recursive TO interconnection
1d	29	Abort due to error during synchronization
1e	30	Axis is in pressure-limiting operation
1f	31	Maximum number of active commands exceeded
20	32	Abort due to active correction command
21	33	Action only permissible in standstill
22	34	Path axis is not ready for operation
23	35	Error during synchronization with a path axis
24	36	Abort due to command on a path axis
25	37	Abort due to stop on a path axis
26	38	Abort due to a pending error response on a path axis
27	39	No actual values for path axis (e.g., encoder or data bus not ready)
28	40	Abort due to reset on a path axis
29	41	Command parameters became invalid during processing
2a	42	A required connection to a technology object does not exist
2b	43	Abort through user program

See also

Troubleshooting - PLCopen Blocks (Page 6733)

8.13.5.4 Query of general errors with the _MC_ReadAxisError function block

General errors on the **axis**, **external encoder**, **synchronous object**, and **cam** can be queried using the **_MC_ReadAxisError** function block. The function block converts the alarms pending on the technology object into a 32-bit representation, which is output at the AxisErrorID parameter.
The alarm numbers of each technology object are assigned to specific alarm groups. In this respect, some alarm groups are defined for all technology objects, while others are TO-specific. A bit in the "errorGroup" system variable is assigned to each alarm group. If a bit is set here, an alarm from this group is pending at the technology object.

Note

Which alarm is assigned to which alarm group for a technology object is fixed and cannot be changed (reconfigured).

It is possible to use SIMOTION SCOUT in the execution system of a device in order to read which alarm group an individual alarm is assigned to. Pressing the "Alarm configuration" button produces the relevant information for each technology object that has been created, under "ExceptionFaultTask" (last column in window that appears).

The tables below indicate which bit numbers are assigned to which alarm groups for the individual technology objects.

Meaning of the bit numbers for the axis

Bit number	Meaning (alarm group)
0	System fault
1	Config fault
2	User fault
3	Peripheral fault
4	Function fault
5	Function aborted
6	Reset/restart fault
7	Distributed motion fault
8	Following error
9	Standstill positioning error
10	Dynamic limit
11	Clamping error
12	Software limit
13	Limit switch
14	Sensor fault
15	Reference not found
16	Output limit
17	Force dynamic limit
18	Additional sensor fault
19	Synchronous motion fault
20	Following object
21	Path synchronous motion fault
22	Path motion fault
23	Path object

Meaning of the bit numbers for the external encoder

Bit number	Meaning (alarm group)
0	System fault
1	Config fault
2	User fault
3	Peripheral fault
4	Function fault
5	Function aborted

Bit number	Meaning (alarm group)
6	Reset/restart fault
7	Distributed motion fault
9	Standstill positioning error
14	Sensor fault
15	Reference not found
19	Synchronous motion fault

Meaning of the bit numbers for the following object

Bit number	Meaning (alarm group)
0	System fault
1	Config fault
2	User fault
3	Peripheral fault
4	Function fault
5	Function aborted
6	Reset/restart fault
7	Distributed motion fault
8	Direct sync
9	Stability
10	Dynamic limit
11	Value ambiguous

Meaning of the bit numbers for the cam

Bit number	Meaning (alarm group)
0	System fault
1	Config fault
2	User fault
3	Peripheral fault
4	Function fault
5	Function aborted
6	Reset/restart fault
8	Value not valid
9	Interpolation fault

Alarms and, therefore, error states, which switch the controller to STOP mode (e.g. 20001 **Internal error**), can be evaluated by the **_MC_ReadAxisError** block in the shutdown task. In addition to the 32-bit error representation, the TO alarms are also output.

See also

Troubleshooting - PLCopen Blocks (Page 6733)

SIMOTION C 9

9.1 SIMOTION C

Preface

Contents of manual

This **document** is part of the **SIMOTION C documentation package**.

Sections in this manual

The following is a list of sections included in this manual along with a description of the information presented in each section.

- Description
  This section explains the purpose and potential applications of the module.
- Operation (hardware)
  This section describes the operator controls and how they are operated.
- Interfaces
  This section includes a description of the interfaces and their functions.
- Installing
  This section explains how to configure the mechanical design and how to install the SIMOTION C components.
- Connecting
  This section describes the wiring and networking of SIMOTION C.
- Addressing
  This section contains the information you will require to define the module start addresses for the modules you are using.
- Commissioning (hardware)
  This section describes how you commission the hardware components and what you must take into account.
- Service and maintenance
  This section describes how you replace the module and update the SIMOTION Kernel.
- Interrupt, error and system alarms
  This section provides information about the causes of alarms and the corrective actions you must take.
- Technical data
  This section describes the properties and features of the SIMOTION C.
- Dimension drawing
  This section contains the dimension drawing of SIMOTION C.

- Spare parts/accessories
  This section provides information on accessories and spare parts for SIMOTION C.
- Appendices with factual information for reference (for example, Standards and approvals, ESD, etc.)
- Index to locate information

Standards and approvals

Our products meet the requirements of EU Directive 89/336/EEC Electromagnetic Compatibility and the harmonized European Standards (EN) listed there.

You can find detailed information on approvals and standards in the appendix.

The current Declaration of Conformity is on the Internet at

http://support.automation.siemens.com/WW/view/de/15257461

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Product disposal

SIMOTION C is an environmentally friendly product. It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Marking of the plastic materials as per ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The product is to be disposed of in accordance with national regulations.

The product described in this manual can be recycled owing to its low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

9.1.1 Fundamental safety instructions

9.1.1.1 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

9.1.1.2 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

9.1.1.3 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

9.1.1.4 Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

9.1.1.5 Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

9.1.2 Description

9.1.2.1 System overview

	C230-2	C240	C240 PN
Mode selector	Rotary switch	Toggle switch	Toggle switch
Error and status displays	X	X	X
Memory module slot for micro memory card	32 MB	64 MB	64 MB
I/O interface X1	X	X	X
Drive interface X2	X	X	-
Measuring system interface X3 - X6	X	X	-
Ethernet interface X7	X	X	X
PROFIBUS DP1 interface X8	X	X	X
PROFIBUS DP2/MPI interface X9	X	X	X
Power supply X10	X	X	X
PROFINET interface X11	-	-	X11 P1 X11 P2 X11 P3
Bus connector for I/O BUS	X	X	X
Diagnostics LED for Ethernet	-	-	X
Diagnostics LED for PROFINET	-	-	X

What capabilities does the SIMOTION C have?

The SIMOTION C is a motion control module for the control of servo drives.

As of V3.2, stepper drives with a pulse-direction interface can be connected to the onboard drive interfaces.

The following configurations can be selected:

- C230-2 and C240: Up to four axes via the onboard drive interface or
- C230-2, C240 and C240 PN: Axes via PROFIBUS DP or
- C240 PN: Axes via PROFINET IO or
- Mixed operation with a maximum of four axes via the onboard drive interface (C230-2, C240) and additional axes via PROFIBUS DP

The number of operable axes on the PROFIBUS DP or PROFINET IO depends on the system cycle clock settings, i.e. more axes can be operated with longer cycle times.

The SIMOTION C controllers are powerful modules for positioning independent single axes or motions in an axis grouping.

Both rotary axes and linear axes can be operated.

The SIMOTION SCOUT engineering system (ES) is used to configure, parameterize, commission, program perform diagnostics for the SIMOTION C.

Where can the SIMOTION C be used?

The C230-2, C240 and C240 PN can be used both for positioning and synchronous operation (gearing, camming and path interpolation).

Typical areas in which the motion control module can be used are:

- Packaging industry
- Plastics industry
- Presses
- Textiles
- Printing industry
- Wood, glass, ceramic, stone

System integration

SIMOTION offers an optimized system platform for automation and drive solutions where priority is given to motion control applications and technological tasks.

The SIMOTION modular system consists of the SIMOTION SCOUT engineering system and a common runtime system for various hardware platforms.

The truly innovative aspect of SIMOTION is that it does away with the traditional separation between pure automation functions and motion functions.

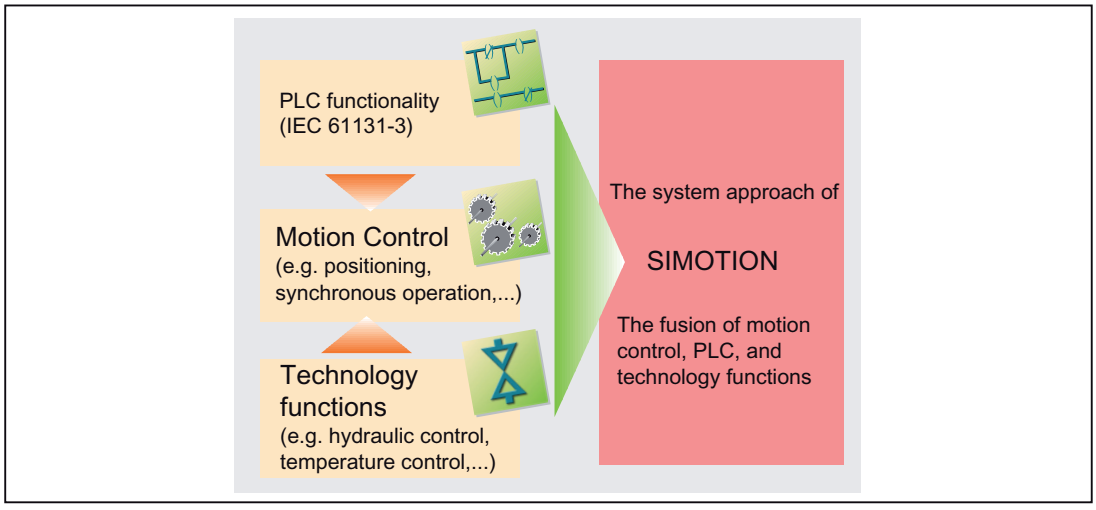


Figure 9-1 System solution for production machines

SIMOTION can be used with all machines with motion control tasks. The focus is on a simple and flexible solution to a wide variety of motion control tasks. In order to achieve this in the best way possible, a new system approach has been introduced:

Motion control has been combined with two other open-loop control functions found in most machines: PLC and technology functions.

This approach enables the motion control of axes and machine control within the same system. The same applies to technology functions, such as pressure control of a hydraulic axis. A seamless switch can be made from position-controlled positioning mode to pressure control.

Combining the three open-loop control functions of motion control, PLC and technology functions has the following benefits:

- Reduced engineering overhead and increased machine performance
- Time-critical interfaces between the individual components are no longer required
- Simple, uniform and transparent programming and diagnostics of the entire machine

The following figures show system configurations with SIMOTION C230-2, C240 and C240 PN.

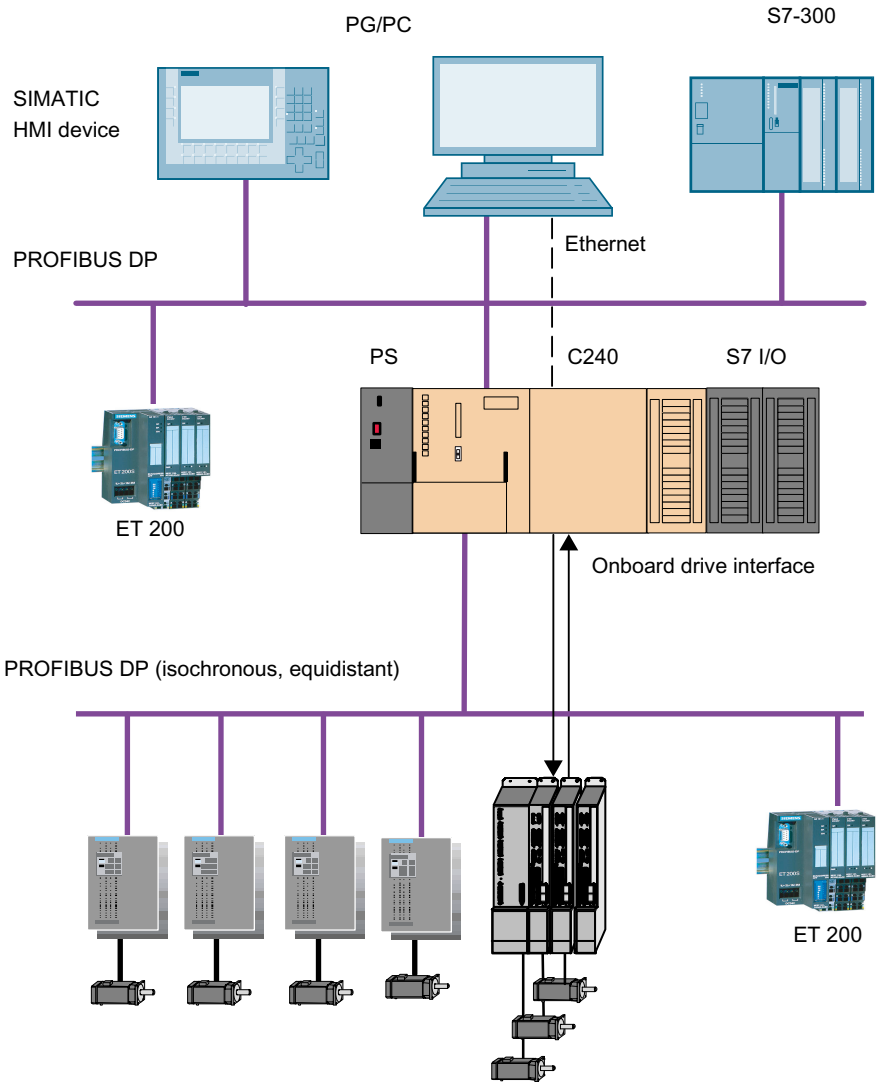


Figure 9-2 System overview of C240

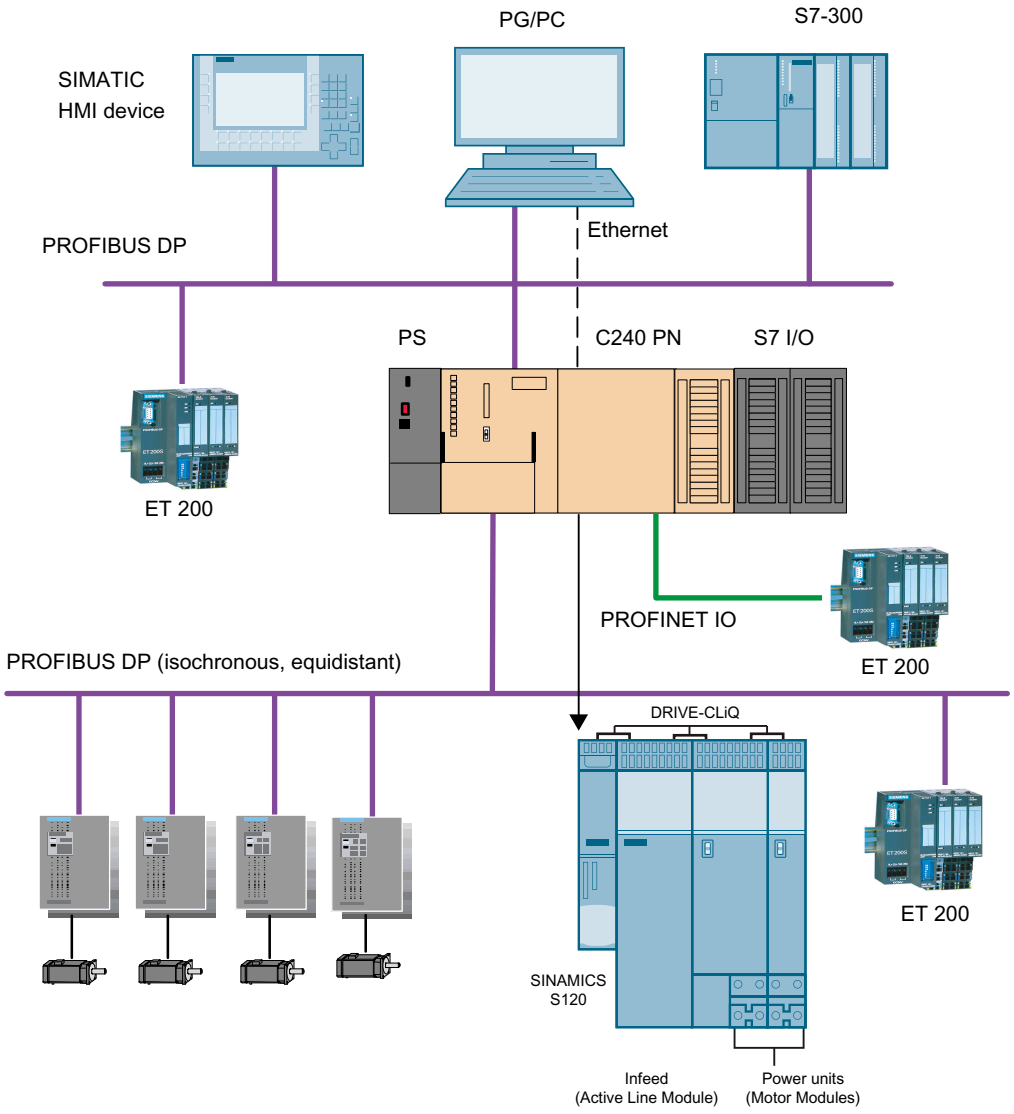


Figure 9-3 System overview of C240 PN

Components

The most important components of a SIMOTION application with SIMOTION C and their functions are listed below.

Note

The modules and devices approved for the SIMOTION C are listed in the *SIMOTION PM 21 Catalog*. This document is supplied in electronic format with SIMOTION SCOUT.

Centralized I/O

Table 9-1 SIMOTION C with centralized I/O

Component	Function
SIMOTION C	... is the motion control module. The SIMOTION C supplies the I/O bus with 5 V; communicates over the I/O bus (backplane bus) with the I/O modules. You can use the integrated fast digital inputs of the SIMOTION C as: • B1 to B4: As homing inputs (C230-2, C240) • M1 and M2: As inputs for measuring inputs - local measuring (C230-2, C240) • B1 to B4: As inputs for global measuring (C240, C240 PN) • Freely addressable process inputs You can use the integrated fast digital outputs as: • Fast output cams • Freely addressable process outputs Drive units with a ±10 V analog interface or stepper drives with a pulse/direction interface can be connected to the onboard drive interface (C230-2, C240). With the C240, the analog interfaces can also be used as available process outputs. Incremental encoders or absolute encoders can be connected to the measuring system interface (C230-2, C240). With the C240, free remaining encoder inputs of an axis channel can be used as the input for a 16-bit up/down counter.
Power supply (PS)	... converts the line voltage (120/230 V AC) into an operating voltage of 24 V DC to supply the SIMOTION C.
Signal modules (SM)	... adapt various process signal levels to the SIMOTION C (digital input/digital output modules and analog input/analog output modules). **Note:** All approved modules are listed in the *PM 21* Catalog and in chapter I/O modules approved for SIMOTION (Page 6753).
Function modules (FM)	... relieve the CPU of computation-intensive tasks, for example, counting. **Note:** All approved modules are listed in the *PM 21* Catalog and in chapter I/O modules approved for SIMOTION (Page 6753).
Communications processors (CP)	... for data exchange **Note:** All approved modules are listed in the *PM 21* Catalog and in chapter I/O modules approved for SIMOTION (Page 6753).

Distributed I/O

Table 9-2 SIMOTION C with distributed I/O

Component	Function
SIMOTION C	... is the motion control module. • Communicates via two PROFIBUS DP interfaces with: – Programming device (PG/PC) – SIMATIC HMI devices – SIMATIC S7 controllers with PROFIBUS DP interface – SIMATIC ET 200 I/O systems – Drive units – Other SIMOTION controllers • Communicates via an Ethernet interface with the programming device (PG/PC) • The C240 PN also communicates via the PROFINET IO interface with: – Programming device (PG/PC) via Ethernet – SIMATIC HMI devices – SIMATIC S7 controllers with PROFINET IO interface – SIMATIC ET 200 I/O systems with PROFINET IO interface – Drive units – Other SIMOTION controllers **Note:** All approved devices are listed in the *PM 21* Catalog and in chapter I/O modules approved for SIMOTION (Page 6753).

Table 9-3 Further components that can be connected to the SIMOTION C

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs and tests SIMOTION.
SIMATIC HMI device	... is used for operator control and monitoring functions. It is not an essential requirement for the operation of a SIMOTION C. **Note:** All approved devices are listed in the *PM 21* Catalog.
Drive units	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors. **Note:** All approved devices are listed in the *PM 21* Catalog.

9.1.2.2 I/O modules approved for SIMOTION

Approved I/O modules

Preferred peripherals for use with SIMOTION C:

- Centralized peripherals:
  SIMATIC S7-300 I/O modules
- Distributed I/O systems for PROFIBUS and PROFINET:
  - **SIMATIC ET 200M** (distributed implementation of SIMATIC S7-300 I/O modules):
    The modular I/O system for control cabinet installation and high channel density.
  - **SIMATIC ET 200S**:
    The bit-modular I/O system for control cabinet installation including motor starters, safety technology and individual grouping of load groups.
  - **SIMATIC ET 200eco (only for PROFIBUS)**:
    The compact, economical I/O system in IP 67 degree of protection for machine-level use without a control cabinet with flexible and fast ECOFAST or M12 connection system.
  - **SIMATIC ET 200pro**
    New modular I/O system in IP65/66/67 degree of protection for machine-level use without a control cabinet; with new features such as small frame size, integrated PROFIsafe safety technology, PROFINET connection and hot swapping of modules.
- Other PROFIBUS DP I/O:
  - **ADI 4 (Analog Drive Interface)**
    For the connection of drives with analog ±10 V setpoint interface.
  - **IM174 (interface module for 4 axes)**
    For the connection of drives with analog ±10 V setpoint interface, external encoders or the connection of stepper drives with pulse-direction interface.

Note

Please note that not all modules of the above-mentioned I/O or I/O systems are approved for SIMOTION. Moreover, system-related functional differences can come into play when these I/O or I/O systems are used on SIMOTION vs. on SIMATIC. For example, special process-control functions (e.g. insertion and removal under voltage, etc.) are not supported by SIMOTION for the SIMATIC ET 200M distributed I/O system.

A detailed, regularly updated list of the I/O modules approved for use with SIMOTION, as well as notes on their use, can be found on the Internet at:

http://support.automation.siemens.com/WW/view/de/11886029

In addition to the I/O modules approved for SIMOTION, all certified standard slaves can, in principle, be connected to SIMOTION if they support the following:

- Cyclic data traffic (DP-V0) and, possibly,
- Acyclic data traffic (DP-V1) or
- Isochronous data traffic (DP-V2)

These modules are integrated via the GSD file from the device manufacturer.

Note

Please note that in individual cases further boundary conditions must be fulfilled in order to integrate a standard slave into SIMOTION. Some modules need "driver blocks", e.g. in the form of function blocks, which make integration possible or much easier.

For modules approved for use with SIMOTION (e.g. S7-300 module FM 350-1, etc.), these driver blocks:

- Up to V3.2 SP1: SIMOTION Function Library
- As of V4.0: SCOUT command library

9.1.2.3 The fundamentals of motion control

Position-controlled motion control for servo axes via the onboard drive interface (C230-2, C240)

These enable position-controlled motion control of up to four axes. The C230-2/C240 provides one analog output per axis for the speed setpoint and one encoder input per axis for the cyclic measurement of the actual position value.

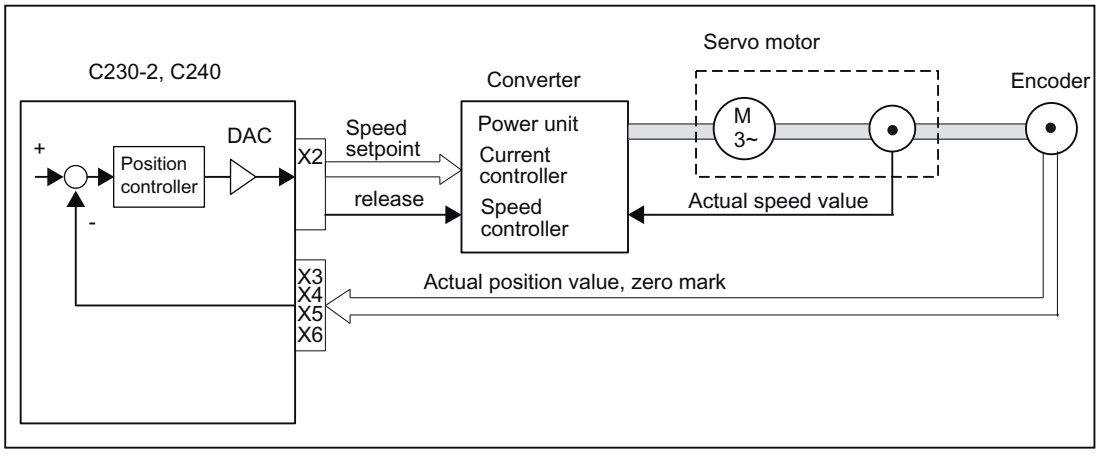


Figure 9-4 Servo system with converter, e.g. SIMODRIVE 611 universal, with built-in incremental encoder

Incremental encoder (C230-2, C240)

For position measurement, the encoders usually connected supply counting pulses according to their resolution for the distances traveled. These can be rotary encoders or linear scales. Homing is necessary to determine the absolute position reference.

Absolute encoder (SSI, C230-2, C240)

Instead of conventional incremental encoders, which supply only a relative dimension for the distance traveled, absolute encoders with a serial interface can be connected. No homing operation need be performed for these encoders as they always supply the absolute position as an actual value.

A one-time adjustment is required for an axis with an absolute encoder when the machine is initially commissioned.

Encoder emulation (C230-2, C240)

Modern motor/converter systems often have a high-resolution motor measuring system (rotor position encoder) connected at the converter. Here the converter provides the position information via an interface that emulates an incremental encoder (e.g. incremental shaft encoder (WSG) interface with SIMODRIVE). A separate position measuring system is not required in this case.

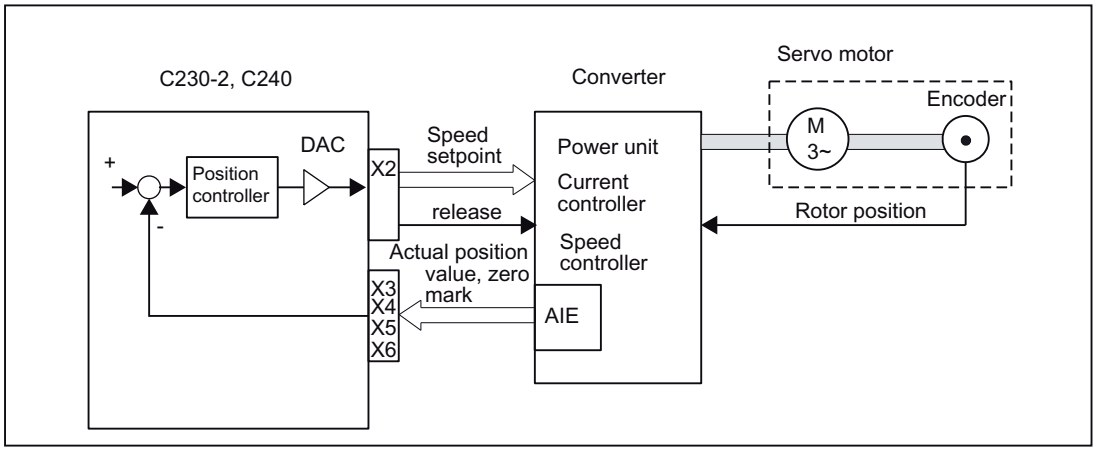


Figure 9-5 Servo system with converter, e.g. SIMODRIVE 611 universal, with incremental encoder

Position-controlled motion control for servo axes (PROFIBUS DP)

SIMOTION C enables position-controlled motion control of axes via PROFIBUS DP.

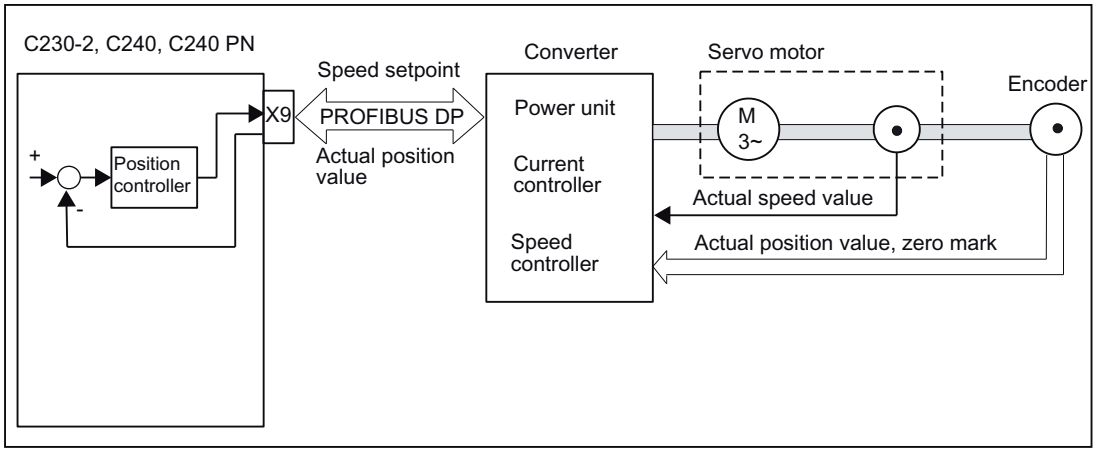


Figure 9-6 Servo system with converter, for example SIMODRIVE 611 Universal

Stepper motor control (C230-2, C240)

In addition to the analog setpoint outputs, the C230-2/C240 has pulse outputs for up to four stepper motor axes. The stepper motor is controlled by means of cycle clocks, the number of which determines the position and the frequency of which determines the rotational speed (velocity). The actual position value is not measured during controlled operation; the position controller takes the number of pulses output (position setpoint) as the actual value. With this type of control circuit, the motor cannot lose any steps in order to enable exact positioning.

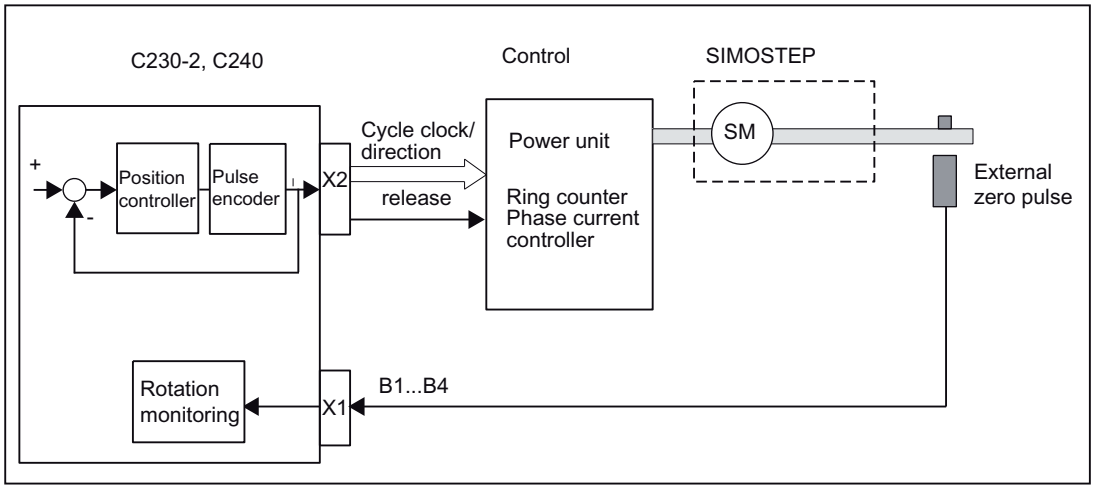


Figure 9-7 Controlled stepper motor system with control circuit

Position-controlled stepper motor control (C230-2, C240)

The C230-2, C240 also provide the option of using one encoder input per axis to operate stepper motors in closed-loop position control as a servo axis.

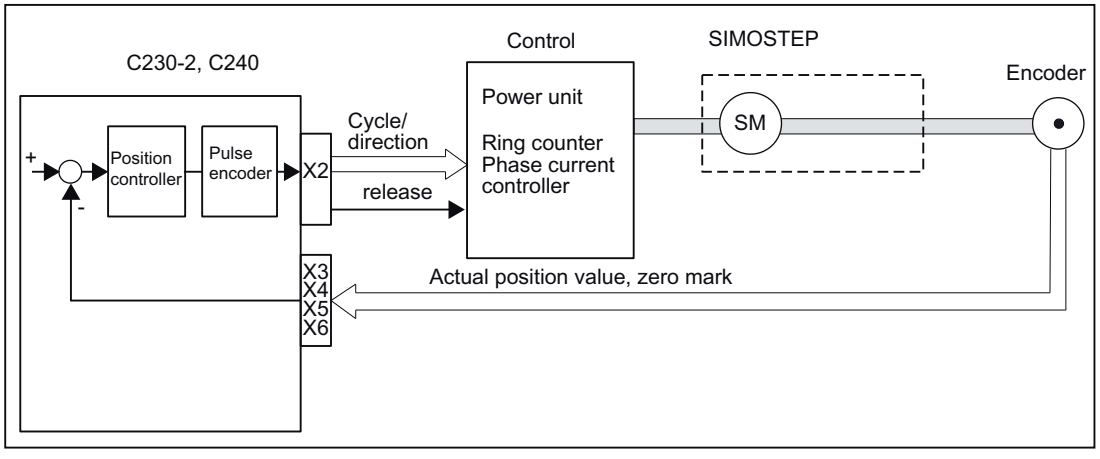


Figure 9-8 Position-controlled stepper motor system with control circuit

Position-controlled motion control for servo axes (C240 PN, PROFINET IO)

The C240 PN enables position-controlled motion control of axes via PROFINET IO.

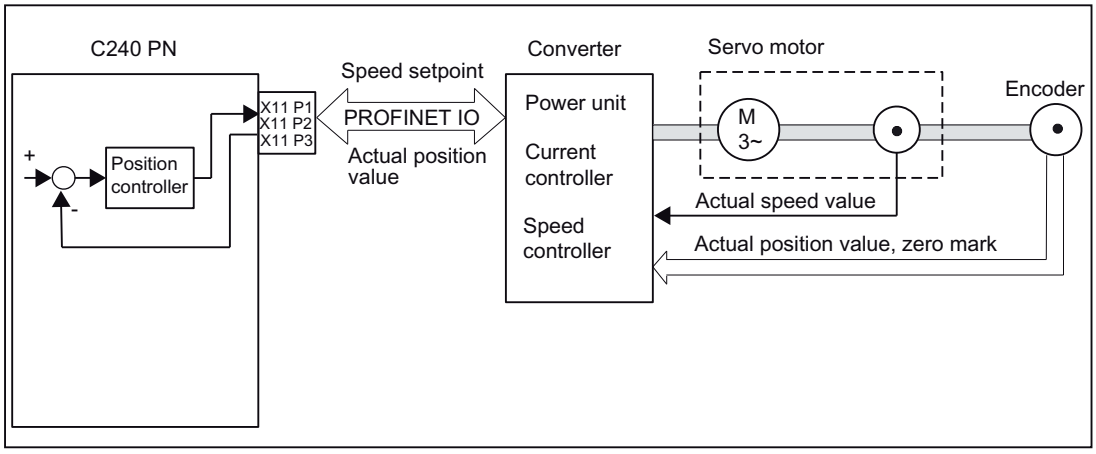


Figure 9-9 Servo system with converter, e.g. SINAMICS S120 with CBE20

9.1.2.4 Layout of the module

View of the C230-2

The following figure shows the C230-2 module, indicating the interfaces and components on the front panel (fault and status displays).

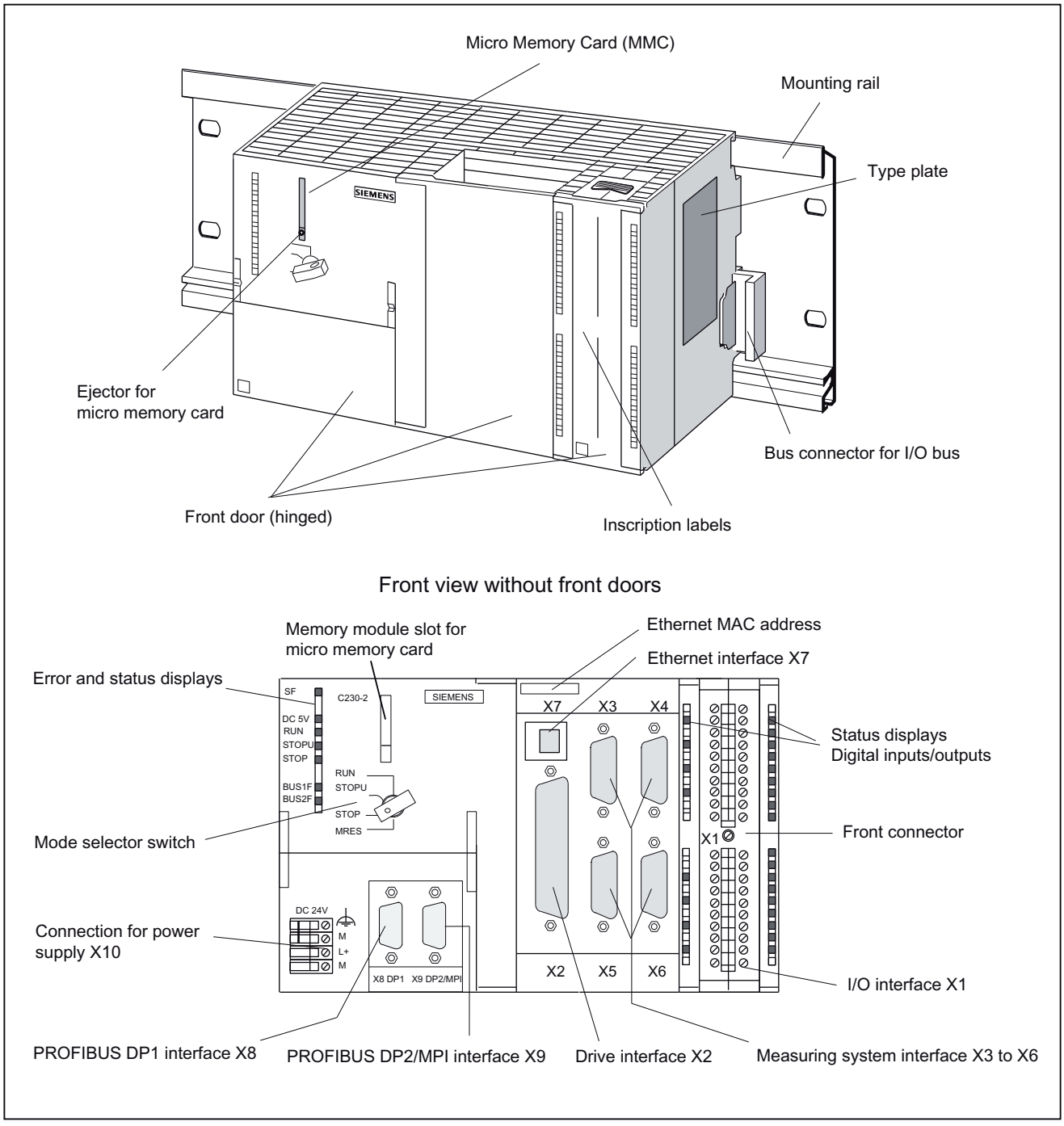


Figure 9-10 Position of the C230-2 interfaces and front panel elements

View of the C240

The following figure shows the C240 module, indicating the interfaces and components on the front panel (fault and status displays).

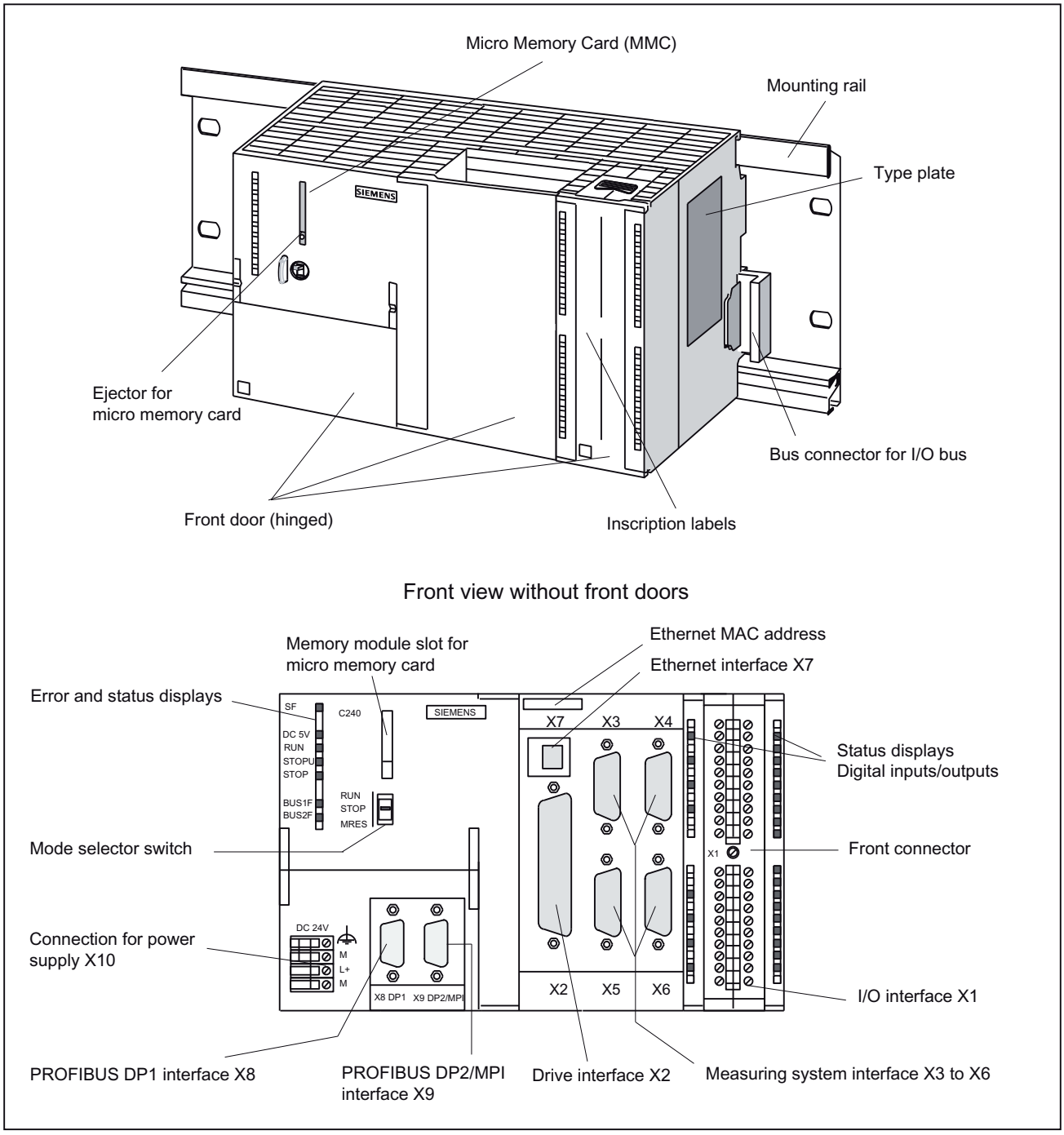


Figure 9-11 Position of the C240 interfaces and front panel elements

View of the C240 PN

The following figure shows the C240 PN module, indicating the interfaces and components on the front panel (fault and status displays).

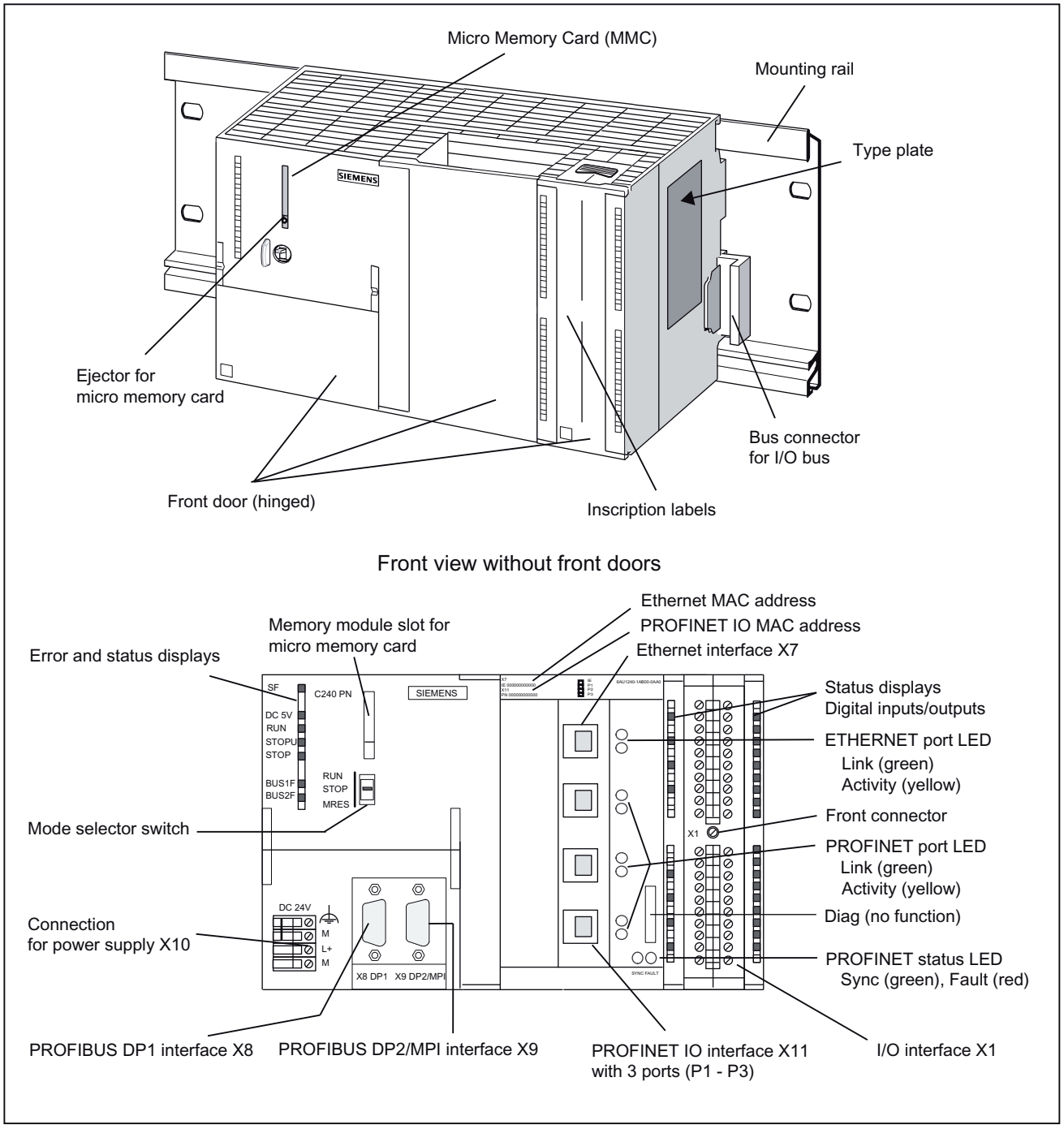


Figure 9-12 Position of the C240 PN interfaces and front panel elements

9.1.2.5 Nameplate

SIMOTION C type plates

The following figures explain the information on the type plate.

Note

The contents of the individual type plate fields on the current module may differ from those described in this manual (e.g. updated product status, approvals and markings not yet issued, etc.).

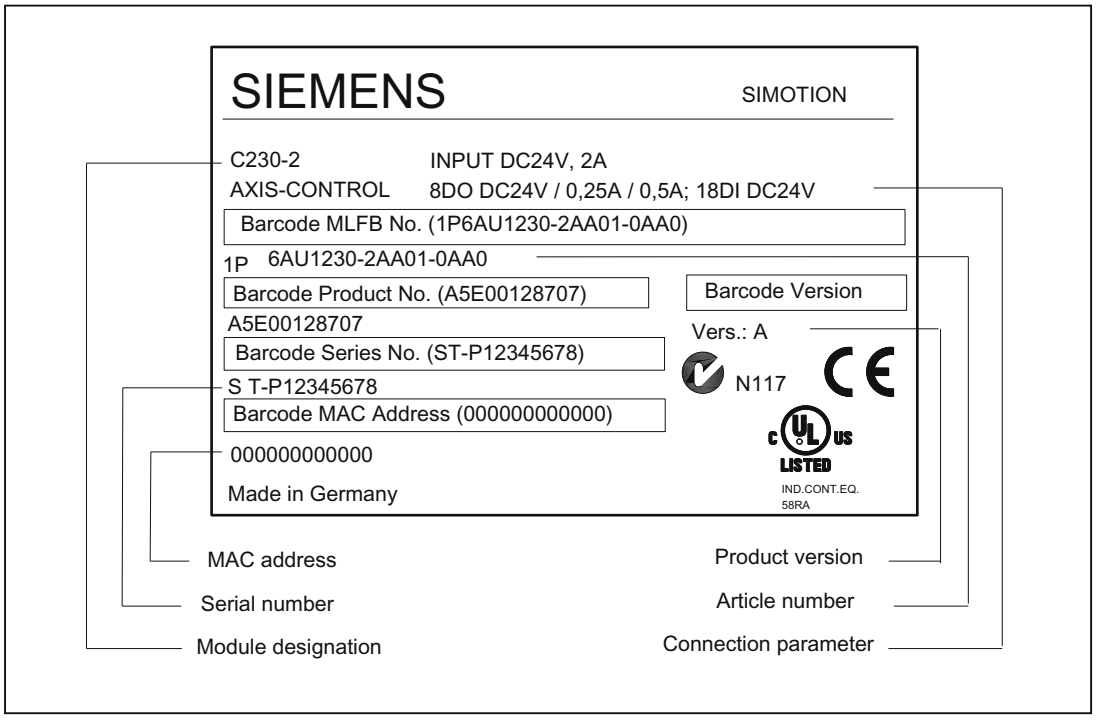


Figure 9-13 Type plate of the C230-2

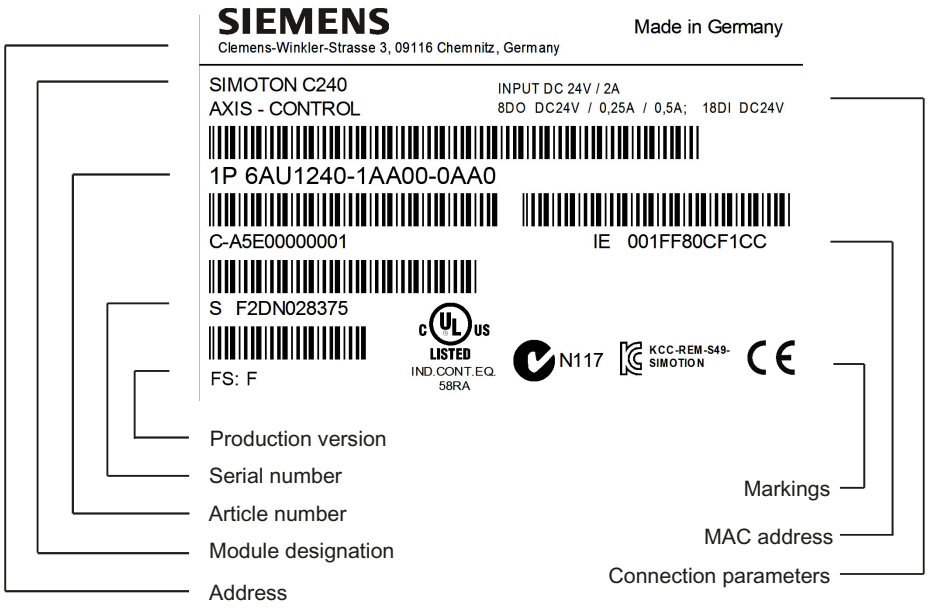


Figure 9-14 Type plate of the C240 (production version F)

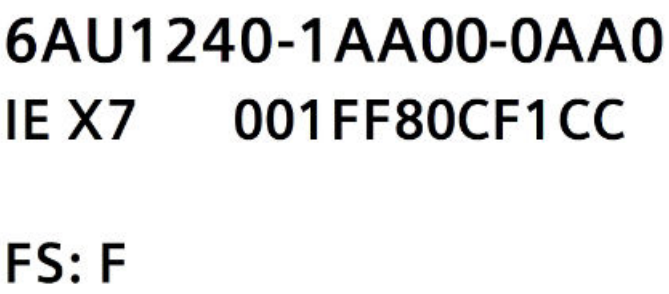


Figure 9-15 Additional plate of the C240 (production version F)

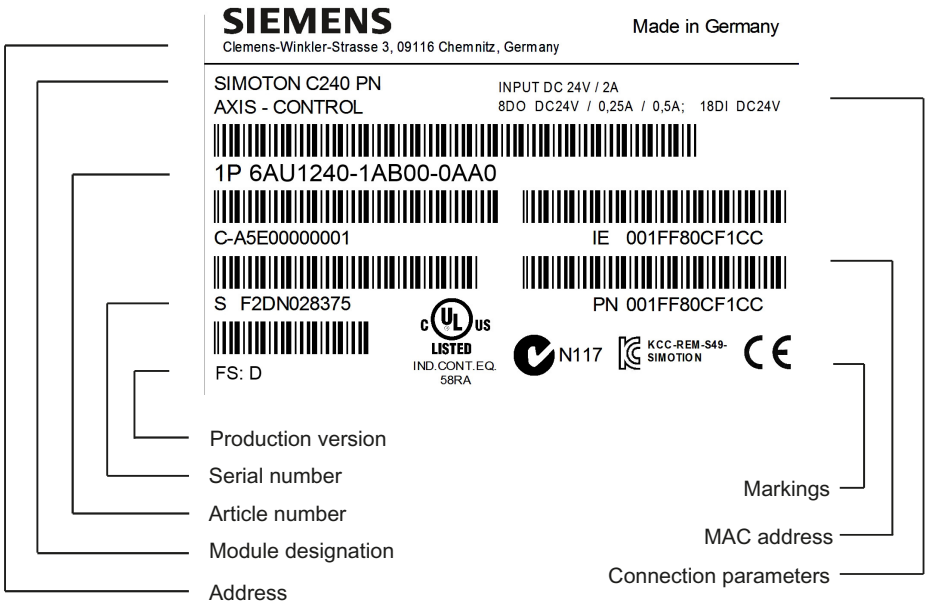


Figure 9-16 Type plate of the C240 PN (production version D)

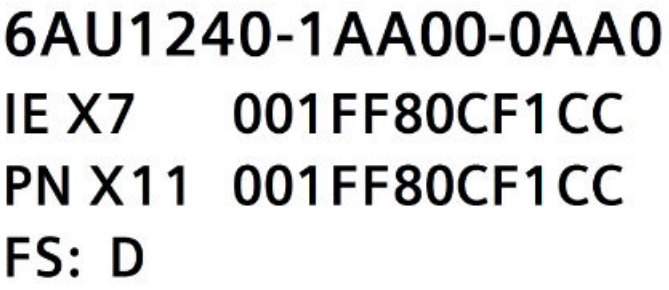


Figure 9-17 Additional plate of the C240 PN (production version D)

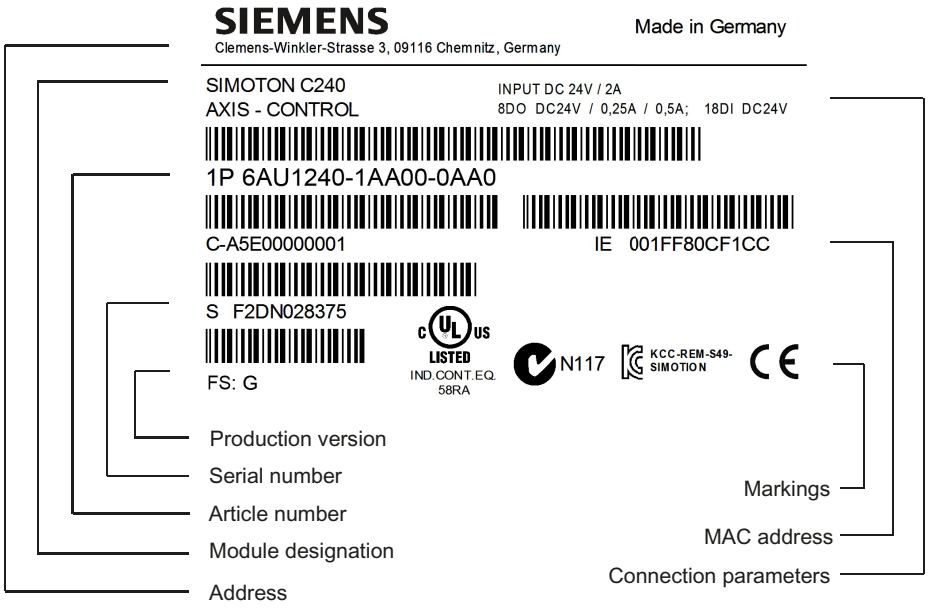


Figure 9-18 Type plate of the C240 (production version G)

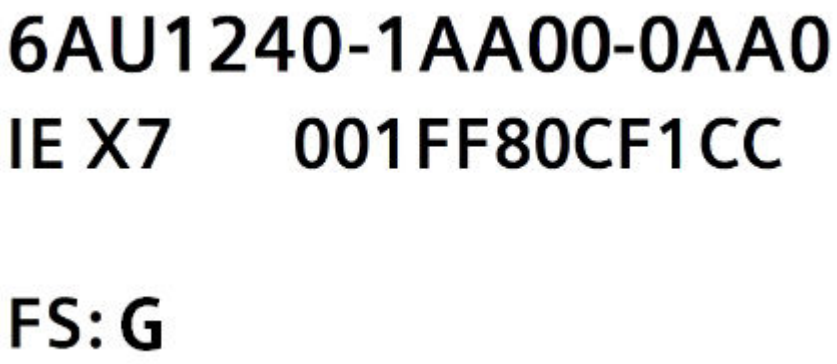


Figure 9-19 Additional plate of the C240 (production version G)

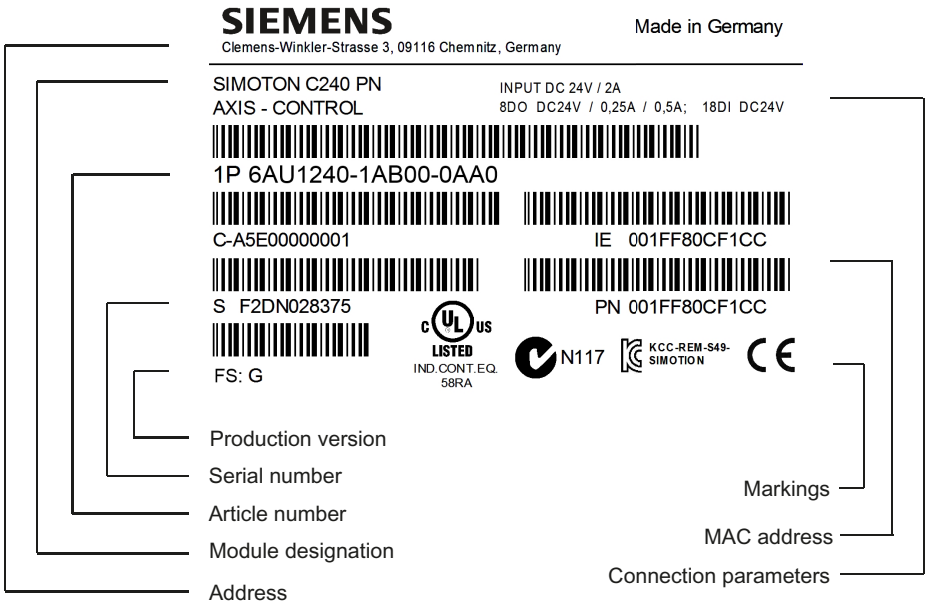


Figure 9-20 Type plate of the C240 PN (production version G)

6AU1240-1AA00-0AA0
IE X7 001FF80CF1CC
PN X11 001FF80CF1CC
FS: G

Figure 9-21 Additional plate of the C240 PN (production version G)

9.1.2.6 Versions of SIMOTION C

There are three different versions of the SIMOTION C: The previous C230-2, a more powerful C240 and a C240 PN with PROFINET interface. The C240 and C240 PN also have functional improvements, as compared to the C230-2.

Compared to the C230-2, the C240 and C240 PN have:

- A larger working memory
- A larger memory for retain data
- Higher performance

Instead of the onboard drive and measuring system interfaces, the C240 PN has a PROFINET interface with three ports (X11 P1, X11 P2, X11 P3).

The following table lists the differences between the individual motion controllers:

Table 9-4 Differences between C230-2, C240 and C240 PN

	C230-2	**C240**	**C240 PN**
Article No. of the module	6AU1 230-2AA01-0AA0	6AU1 240-1AA00-0AA0	6AU1 240-1AB00-0AA0
Article No. of the Micro Memory Card (MMC)	6AU1 700-0AA02-0AA0	6AU1 720-1KA00-0AA0 **Note**: The MMC of the C240 and the C240 PN are identical.	
	Note: The MMC of the C240 / C240 PN cannot be operated in the C230-2 and vice-versa.		
Uses of the analog drive interface (X2)	• For analog drives • For stepper drives	• For analog drives • For stepper drives • For standard outputs (analog/digital outputs)	• Not available
Filter time for analog outputs	• With filter (C230-2-compatible)	• With filter (C230-2-compatible) • Without filter	• Not available
Repeatability when using the outputs as fast output cams (X1)	140 µs	70 µs	70 µs
Use of the inputs (X1: B1... B4)	• As digital inputs • For external zero mark signals	• As digital inputs • For external zero mark signals • Measuring pulses for global measuring (in addition to inputs M1 and M2 for local measuring)	• As digital inputs • Measuring pulses for global measuring
Use of measuring system interface (X3 to X6)	• Encoder connection	• Encoder connection • Counter input	• Not available
Mode selector (hardware)	Key-operated switch Switch positions: • RUN • STOPU • STOP • MRES	Toggle switch Switch positions: • RUN • STOP • MRES	Toggle switch Switch positions: • RUN • STOP • MRES
PROFINET interface (X11)	Not available	Not available	3x PROFINET X11 P1, X11 P2, X11 P3

Note

In order to take the different versions into account, the product will be referred to in this manual as "SIMOTION C". Specific product designations will be used for information that applies only to one product version, e.g. for SIMOTION C230-2, C240 or C240 PN.

9.1.3 Operator control (hardware)

9.1.3.1 Control Elements

Mode selector

Certain operating modes can be selected using the mode selector.

Mode selector positions

The mode selector positions are explained in the order in which they are arranged on the SIMOTION C.

Table 9-5 Modes and switch settings

Operating mode	Explanations
RUN	SIMOTION C executes the user program (UP) and the associated system functions: • Reading process image of inputs • Execution of the user programs assigned to the execution system • Writing process image of outputs The technology packages are active in this state. They can execute commands from the user program. **Note:** With the C230-2, the key cannot be removed in this position.
STOPU	SIMOTION C does not execute a user program. • The technology packages are active. Test and commissioning functions can be executed. The user program is not active. • The I/O modules (SMs) are in a safe state. With the C230-2, the key can be removed in this position so that no unauthorized person can change the operating modes. **Note:** The toggle switch of the C240 / C240 PN does **not** have the "STOPU" switch setting. You can switch to "STOPU" mode **only** via the SIMOTION SCOUT engineering system. In SIMOTION SCOUT, you can switch from the hardware settings "STOP" and "RUN" to "STOPU" mode.

Operating mode	Explanations
STOP	SIMOTION C does not execute a user program. • It is possible to load a complete user program. • All system services (communications, etc.) are active. • The I/O modules (SMs) are in a safe state. • The technology packages are inactive, i.e. all enables are deleted. No axis motions can be executed. **Note:** With the C230-2, the key can be removed in this position so that no unauthorized person can change the operating modes.
MRES (memory reset)	Pushbutton position for memory reset on the SIMOTION C. A memory reset by means of a mode selector requires a specific sequence of operation, see chapter SIMOTION C memory reset (Page 6869).

Micro Memory Card (MMC)

SIMOTION C230-2 micro memory card

The following micro memory card is available:

Article No.: 6AU1 700-0AA02-0AA0

The micro memory card for SIMOTION C230-2 can be used to save the SIMOTION Kernel in order to run an update. See chapter SIMOTION kernel update (Page 6872).

The SIMOTION Kernel should always be stored on the micro memory card.

As of SIMOTION V2.1, the SIMOTION Kernel is automatically copied to the micro memory card during power-up if it does not already contain one.

The micro memory card is also needed to save the technology packages and user data (programs, configuration data, parameterizations).

⚠ CAUTION
The micro memory card may only be inserted or removed when the control unit is disconnected from the power supply.

Micro memory card for SIMOTION C240 / C240 PN

The following micro memory card is available:

Article No.: 6AU1 720-1KA00-0AA0

The micro memory card is **mandatory** for operating the SIMOTION C240 / C240 PN.

The micro memory card is supplied in a bootable format with the latest SIMOTION Kernel. It is **not** supplied with the SIMOTION C240 / C240 PN and must be ordered as a separate component.

Note: The SIMOTION C240 / C240 PN does not contain any firmware.

The micro memory card for the SIMOTION C240 / C240 PN can be used to save the SIMOTION Kernel.

The micro memory card is also needed to save the technology packages and user data (programs, configuration data, parameterizations).

⚠ CAUTION
The micro memory card may only be inserted or removed when the control unit is disconnected from the power supply.

Memory module slot

The micro memory card is inserted in the memory module slot.

9.1.3.2 Display elements

LED displays

The following LED displays are on the front panel of the SIMOTION C. This table describes the LEDs and their function.

Table 9-6 Status and error displays

LED	Meaning
SF (red)	This LED indicates a fault on the SIMOTION C.
5 VDC (green)	This LED indicates that the power supply for the electronics is ready.
RUN (green) - SIMOTION C in RUN mode	This LED indicates that the user program is running.
STOPU (yellow) - SIMOTION C in STOP user program mode	This LED indicates that the technology packages (for example, synchronous operation and cam) are active. The user program is not active.
STOP (yellow) - SIMOTION C in STOP mode	This LED indicates that no user program is running. The technology packages are not active.
BUS1F (red) - Group fault	This LED indicates a fault on the SIMOTION C PROFIBUS DP1 interface **(X8)**.
BUS2F (red) - Group fault	This LED indicates a fault on the SIMOTION C PROFIBUS DP2/MPI interface **(X9)**.
Q0 to Q7 , I0 to I11 , B1 to B4, M1, M2 (green) - Digital inputs/digital outputs	These LEDs show the status of the digital inputs/outputs.

The following LED displays are arranged the front cover of the SIMOTION C240 PN. This table describes the LEDs and their function.

Table 9-7 Status and fault displays behind the front cover (C240 PN)

LED	Meaning
Ethernet link (X7) (green)	This LED indicates a physical connection of the Ethernet interface.
Ethernet activity (X7) (yellow)	This LED indicates a data transfer via the Ethernet interface.
PROFINET link (X11 Px) (green)	This LED indicates a physical connection of the PROFINET interface at port x.
PROFINET activity (X11 Px) (yellow)	This LED indicates a data transfer via the PROFINET interface at port x.
PROFINET fault (X11) (red)	This LED indicates a fault at the PROFINET interface.
PROFINET sync (X11) (green)	This LED indicates synchronization status of the PROFINET interface.

See also

Diagnosis using the LEDs (Page 6877)

9.1.4 Interfaces

9.1.4.1 SIMOTION C interfaces

The interfaces and their meaning are described in following table.

Table 9-8 Interfaces

Interfaces	Description
Bus connector	Rear-mounted connector for connecting to other S7 modules via the I/O bus
Drive interface (onboard) (C230-2, C240)	50-pin Sub-D connector **X2** for the analog connection of analog and stepper drives (max. 4 axes)
Measuring system interface (onboard) (C230-2, C240)	15-pin Sub-D sockets **X3** to **X6** for connection of encoders (max. 4)
I/O interface	40-pin front connectors **X1** for connecting the fast digital inputs/digital outputs including measuring inputs and external zero mark and for wiring the READY relay
Power supply connection	4-pin screw-type terminal connection **X10** for connecting the 24 V load power supply
Memory module slot	Slot for micro memory card (MMC)
PROFIBUS DP1 interface	9-pin Sub-D socket **X8** for connection to the PROFIBUS DP. This interface can be used for isochronous operation
PROFIBUS DP2/MPI interface	9-pin Sub-D socket **X9** for connection to PROFIBUS DP (factory setting) or an MPI bus This interface can be used for isochronous operation.

Interfaces	Description
Ethernet interface	8-pin RJ45 socket **X7** for connection to an Industrial Ethernet
PROFINET interface with three ports (C240 PN)	3 x 8-pin RJ45 sockets **X11 P1, X11 P2** and **X11 P3** for connecting to PROFINET subnets

9.1.4.2 Ethernet interface

Definition

Interface for connecting an Industrial Ethernet.

Industrial Ethernet is a communication network with a transmission rate of 10/100 Mbit/s.

The SIMOTION C provides the following functions via the Ethernet interface:

- Communication with STEP 7 and SIMOTION SCOUT
- Communication with distributed I/O (e.g. SIMATIC HMI)
- Communication between SIMOTION and SIMATIC NET OPC
  The "SIMATIC NET SOFTNET S7 (S7 OPC server)"
  software must be installed on the PG/PC for this function.
  Note:
  SOFTNET-S7 is a superset of SOFTNET-PG, i.e. SOFTNET-S7 contains Protocol TPC/IP RFC 1006 as well.

For more information about the software packages, see *SIMOTION Motion Control System*, PM 21 Catalog. This documentation is supplied in electronic format with SIMOTION SCOUT.

Connectable devices

A PG/PC can be connected to the Ethernet interface via a Fast Ethernet network. The PG must be equipped with an Ethernet card and the corresponding software must be available.

Interface position

The following figures show the installation position and the designation of the interface on the C230-2, C240 and C240 PN modules.

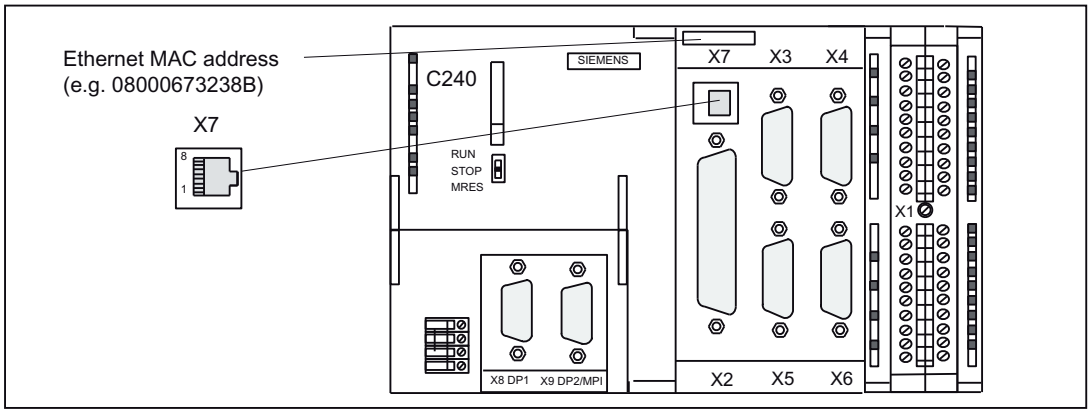


Figure 9-22 X7 interface position (C230-2, C240)

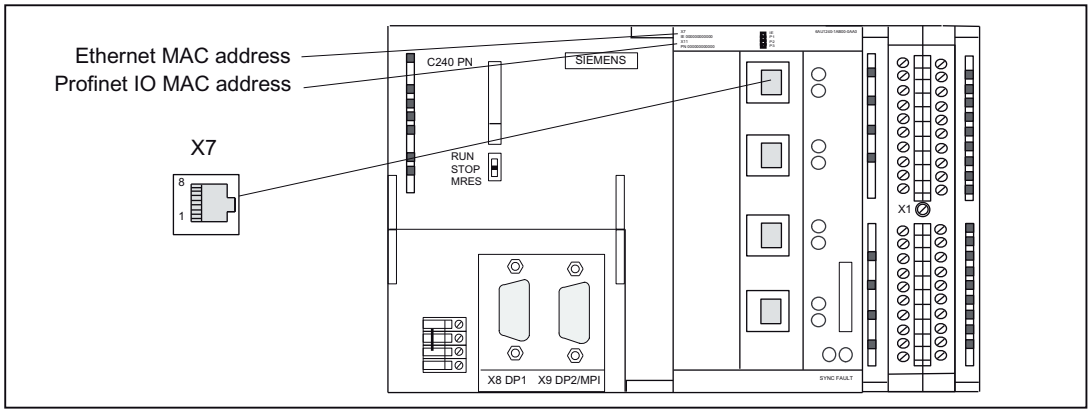


Figure 9-23 X7 interface position (C240 PN)

Interface assignment

Designation: **X7 (Ethernet)**
Type: 8-pin RJ45 socket

Table 9-9 X7 interface assignment

Pin	Name	Type	Pin	Name	Type
1	TDP	O	5	Not assigned	
2	TDM	O	6	RDM	I
3	RDP	I	7	Not assigned	
4	Not assigned		8	Not assigned	

Signal names

RDP, RDM - Receive Data +/-
TDP, TDM - Transmit Data +/-

Signal type

I - signal input
O - signal output

Note

You will find additional information about Ethernet in the *SIMOTION SCOUT Communication* System Manual.

9.1.4.3 PROFINET interface (C240 PN)

Definition

A SIMOTION C240 PN provides an interface for connecting to PROFINET IO with three ports (X11 P1, X11 P2, X11 P3) with a transmission rate of 100 Mbit/s. The PROFINET interface supports the parallel operation of:

- IRT with the "high flexibility" and "high performance" options - isochronous real-time communication:
  - Equidistant transmission of input/output data between an IO controller and its IO devices with high stability for time-critical applications (e.g. motion control). The required bandwidth is in the bandwidth reserved for cyclic data. For "high flexibility", a fixed bandwidth of the transmission resources is reserved for the real-time communication. The "high flexibility" enables simple planning and expansion of the system. For "high performance", a fixed bandwidth of the transmission resources is also reserved for the real-time communication. The data traffic is further optimized and accelerated through an additional topology planning. The "high performance" option always requires a configuration of the topology.
  - With the aid of IRT "high performance", IO devices (I/O modules and drive units) that support IRT can be operated isochronously on PROFINET IO and data can be exchanged between the SIMOTION devices via the controller-controller data exchange broadcast.
- RT - real-time communication: Transmission of input/output data between an IO controller and its IO devices in prioritized Ethernet message frames, but not isochronously. The required bandwidth is in the bandwidth of PROFINET IO reserved for cyclic data.
- Standard Ethernet communication such as TCP/IP, UDP, HTTP, FTP, etc. as well as communication with STEP 7 / SIMOTION SCOUT and communication with SIMATIC NET OPC: The required bandwidth is in the free bandwidth of PROFINET IO.

A SIMOTION C240 PN can be used as IO controller or I device. PROFINET IO differentiates between an IO controller and the IO devices assigned to it. IO controller and IO devices form a PROFINET IO system, comparable to a master-slave system on PROFIBUS DP.

Connectable devices

The following devices can be connected to the PROFINET interface:

- PG/PC programming device
- SIMATIC HMI devices
- SIMATIC S7 controllers with PROFINET interface
- Distributed I/O (e.g. SIMATIC ET 200M) with PROFINET interface
- Drive units with PROFINET interface
- SIMOTION devices with PROFINET interface according to PROFIdrive profiles / IEC61800-7

- Teleservice adapter
- Gateways

Note

All released modules and devices are listed in the PM 21 Catalog and in chapter I/O modules approved for SIMOTION (Page 6753). This documentation is supplied in electronic format with SIMOTION SCOUT. Take note of the documentation on the individual modules or devices!

Interface position

The following figure shows the mounting position and the designation of the PROFINET IO interface on the module.

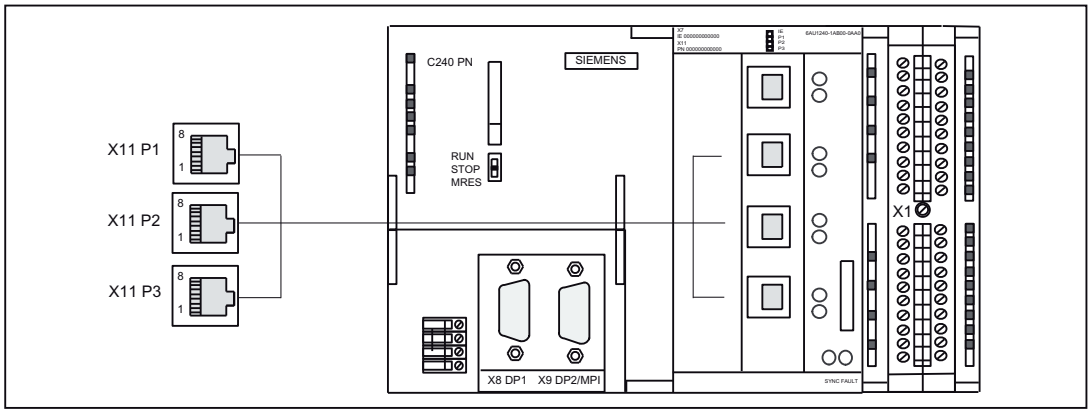


Figure 9-24 X11 interface position (C240 PN), three ports P1, P2, P3

The following LED displays are arranged behind the front cover of the SIMOTION C240 PN. For a description of the LED displays, see section Display elements (Page 6769).

Interface assignment

Designation: X11 (PROFINET), three ports P1, P2, P3

Type: 8-pin RJ45 socket

Table 9-10 X11 interface assignment (ports P1, P2, P3)

Pin	Name	Type	Pin	Name	Type
1	TDP	O	5	Not assigned	
2	TDM	O	6	RDM	I
3	RDP	I	7	Not assigned	
4	Not assigned		8	Not assigned	

Signal names

RDP, RDM - Receive Data +/-

TDP, TDM - Transmit Data +/-

Signal type

I - signal input

O - signal output

Note

You will find additional information about PROFINET in the *SIMOTION SCOUT Communication* System Manual.

9.1.4.4 PROFIBUS DP interfaces

PROFIBUS DP interfaces (X8, X9)

The SIMOTION C provides two interfaces for connection to the PROFIBUS DP. Baud rates up to 12 Mbits/s are possible. Both interfaces can be operated isochronously.

If both interfaces (X8, X9) are to be operated isochronously, then they must both be configured with the same DP cycle clock.

Alternatively, the X9 interface can be used as an MPI interface with a transmission rate up to 12 Mbits/s.

Connectable devices

The following devices can be connected to the PROFIBUS DP interfaces:

- PG/PC
- SIMATIC HMI devices
- SIMATIC S7 controllers with PROFIBUS DP interface
- Distributed I/O (e.g. SIMATIC ET 200M)
  The digital inputs/digital outputs are updated in the position control cycle clock.
- SIMOTION controller
- Teleservice adapter
- Drive units with PROFIBUS DP interface (e.g. SIMODRIVE 611 universal) according to PROFIdrive profiles / IEC61800-7.

Note

A teleservice adapter can only be connected to **one** of the two interfaces.

All released modules and devices are listed in the *PM 21* Catalog and in chapter I/O modules approved for SIMOTION (Page 6753). This document is supplied in electronic format with SIMOTION SCOUT.

Take note of the documentation on the individual modules or devices!

Interface positions

The following figure shows the mounting position and the designation of the interfaces on the module.

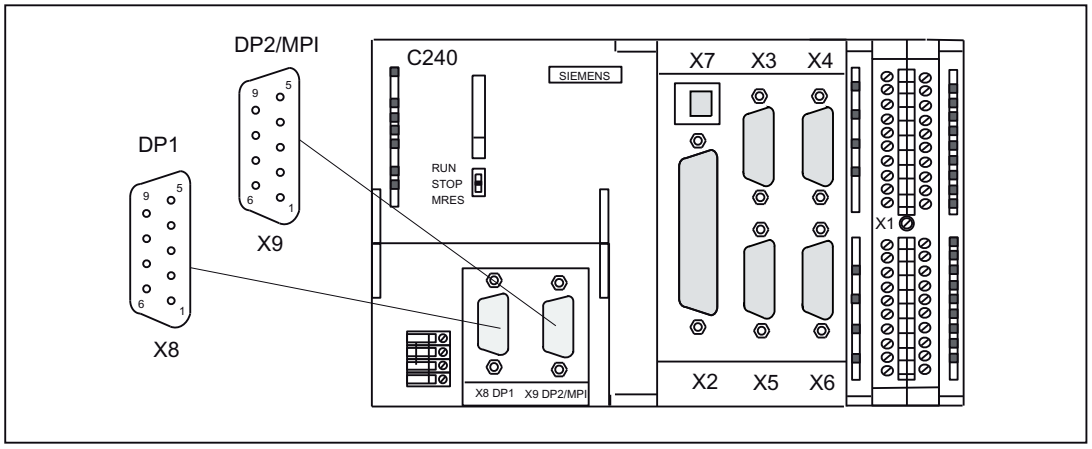


Figure 9-25 Position of connectors X8, X9

Interface assignments

Designation: **X8, X9** DP1, DP2/MPI Type: 9-pin Sub-D socket connector

Table 9-11 X8, X9 interface assignments

Pin	Name	Type	Pin	Name	Type
1	Not assigned		6	P5	VO
2	M24	VO	7	P24	VO
3	B	I/O	8	A	I/O
4	RTS	O	9	Not assigned	
5	M5	VO			

Signal names

Table 9-12 Signal names

Signal name	Meaning
A, B	Data input/output (RS485)
RTS	Transmission request
P5	5 V power supply 60 mA, short-circuit-proof
M5	5 V reference potential
P24	24 V supply 150 mA, short-circuit-proof, not isolated
M24	24 V reference potential

Signal type

O - signal output
I/O - signal input/output
VO - voltage output

9.1.4.5 Onboard drive interface (C230-2, C240)

Connector to drive unit

Drive units with an analog interface (±10 V) or stepper motor power units with at least one clock pulse input and direction input can be connected to the 50-pin Sub-D socket X2. Any hybrid configuration can be used for up to four drives.

In addition, the C230-2 and C240 provide one enable signal per axis.

With the C240, this interface (X2) can also be used for standard outputs:

- 4 analog outputs
- 4 digital outputs

Position of the connector

The following figure shows the installation position and the designation of the connector on the module.

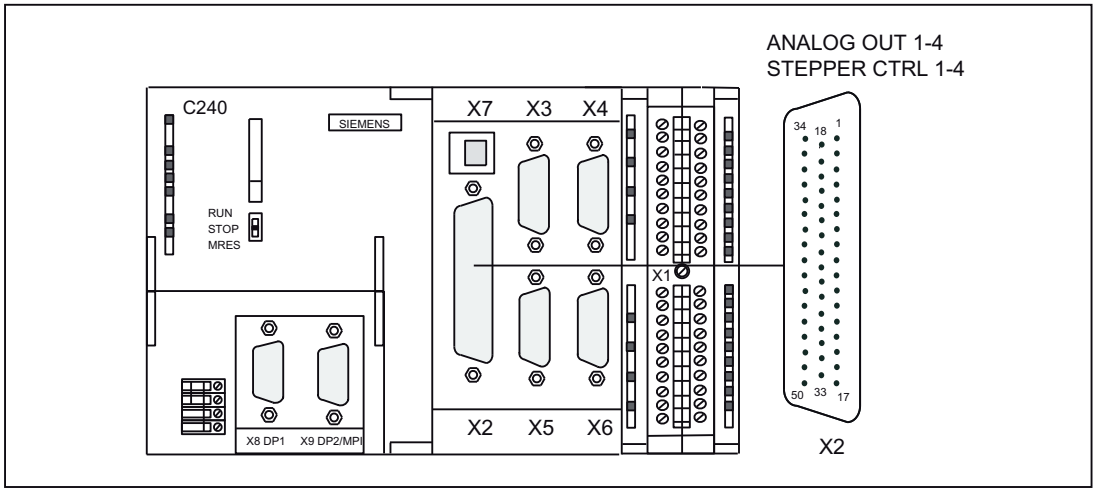


Figure 9-26 Position of the X2 connector

Connector pin assignment

Onboard drive interface (servo interface, 4 axes)

Connector designation: X2 ANALOG OUT 1-4/STEPPER CTRL 1-4
Connector type: 50-pin Sub-D plug connector

Table 9-13 X2 connector pin assignment

Pin	Name	Type	Pin	Name	Type	Pin	Name	Type
1	SETP1	VO	18	ENABLE1	O	34	REFPOT1	VO
2	REFPOT2	VO	19	ENABLE1_N	O	35	SETP2	VO
3	SETP3	VO	20	ENABLE2	O	36	REFPOT3	VO
4	REFPOT4	VO	21	ENABLE2_N	O	37	SETP4	VO
5	PULSE1	O	22	GND	O	38	PULSE1_N	O
6	DIR1	O	23	GND	O	39	DIR1_N	O
7	PULSE2_N	O	24	GND	O	40	PULSE2	O
8	DIR2_N	O	25	GND	O	41	DIR2	O
9	PULSE3	O	26	ENABLE3	O	42	PULSE3_N	O
10	DIR3	O	27	ENABLE3_N	O	43	DIR3_N	O
11	PULSE4_N	O	28	ENABLE4	O	44	PULSE4	O
12	DIR4_N	O	29	ENABLE4_N	O	45	DIR4	O
13	Not assigned		30	Not assigned		46	Not assigned	
14	CTREN1.1	C	31	Not assigned		47	CTREN1.2	C
15	CTREN2.1	C	32	Not assigned		48	CTREN2.2	C
16	CTREN3.1	C	33	Not assigned		49	CTREN3.2	C
17	CTREN4.1	C				50	CTREN4.2	C

Signal names

Table 9-14 Signal names for drives with analog interface

Signal name	Meaning
SETP[1 to 4]	Setpoint
REFPOT[1 to 4]	Reference potential for setpoint (analog ground)
CTREN[1.1 to 4.1], CTREN[1.2 to 4.2]	Controller-enable contact

Table 9-15 Signal names for stepper drives:

Signal name	Meaning
PULS[1 to 4], PULS[1 to 4]_N	Clock pulse is inverted and not inverted
DIR[1 to 4], DIR[1 to 4]_N	Clock pulse inverted and non-inverted
ENABLE[1 to 4], ENABLE[1 to 4]_N	Controller enable inverted and non-inverted
GND	Signal ground

Signal type

O - signal output
VO - voltage output
K - switching contact

Drives with analog interface

Signals:

One voltage signal and one enable signal are provided per axis.

- **Setpoint (SETP)**
  Analog voltage signal in the ±10 V range for the output of a speed setpoint.
- **Reference potential (REFPOT**)
  Reference potential (analog ground) for the setpoint signal, connected internally to logic ground.
- **Controller enable (CTREN)**
  Contact assembly mated set (NO contact), used for axis-specific enabling of the drive, e.g. a SIMODRIVE drive unit. The RF signal to the drive is set as soon as the controller enable is signaled by the user program.

Note: When used as a digital output, the contact must be supplied with voltage.

⚠ **WARNING**
Brief voltage peaks may occur at the analog outputs when the supply voltage is switched on or off. For this reason, it is important to make sure that the enable signals are wired correctly and that the necessary safety regulations are met.

Stepper drives

Signals:

One clock pulse signal, direction signal, and enable signal is provided as a true and negated signal.

- **Setpoint (PULS)**
  The clock pulses control the motor. The motor performs one step for each rising pulse edge. Thus, the number of pulses output determines the angle of rotation, i.e. the distance to be traversed.
  The pulse frequency determines the rotational speed, i.e. the traversing speed.

  ⚠ CAUTION

  If your drive unit responds to falling pulse edges, you must replace the true pulse signal with the negated pulse signal when performing the wiring; failure to do so can cause deviations to occur between the position calculated by the controller and the actual position.

- **Direction signal (DIR)**
  The output signal level determines the direction of rotation of the motor.
  Signal ON: "Rotation to the left"
  Signal OFF: "Rotation to the right"

  Note

  If the direction of rotation of the motor is different, you can change the direction of rotation by means of the "Reversal of traversing direction" configuration data element. Refer to the technical documentation of your drive unit for the correct assignment of the signal levels to the direction of rotation.

- **Enable signal (ENABLE)**
  This signal is activated when the axis enable is set in RUN mode by the user program.
  Signal ON: Power control circuit enabled
  Signal OFF: Depending on the power unit, one or more of the following responses can occur:
  - Disable pulse input
  - Deenergize motor
  - Reset ring counter
  - Delete error messages

  Note

  The ENABLE signal is output at the same time with controller enable contact RF. Alternatively, you can also use the relay contacts.

Signal parameters

All signals for stepper drives are output by means of differential signal line drivers in accordance with the RS422 standard. For optimal reliability, the power unit should have differential signal

receivers or optical coupler inputs to enable symmetrical signal transmission. An asymmetrical transmission is also possible, however, the maximum cable length in this case is limited to 10 m.

Note

Because of the wide range of non-standardized input circuits of the drive units during asymmetrical transmission, no responsibility can be taken for this function. In particular, cable lengths and the limit frequency depend on the properties of the input circuit and the cable being used. In addition, the GND reference potential should be isolated to prevent electrical interference.

All outputs are electronically protected against short-circuit and thermal overload.

The following figure shows different possibilities for protective signal circuits.

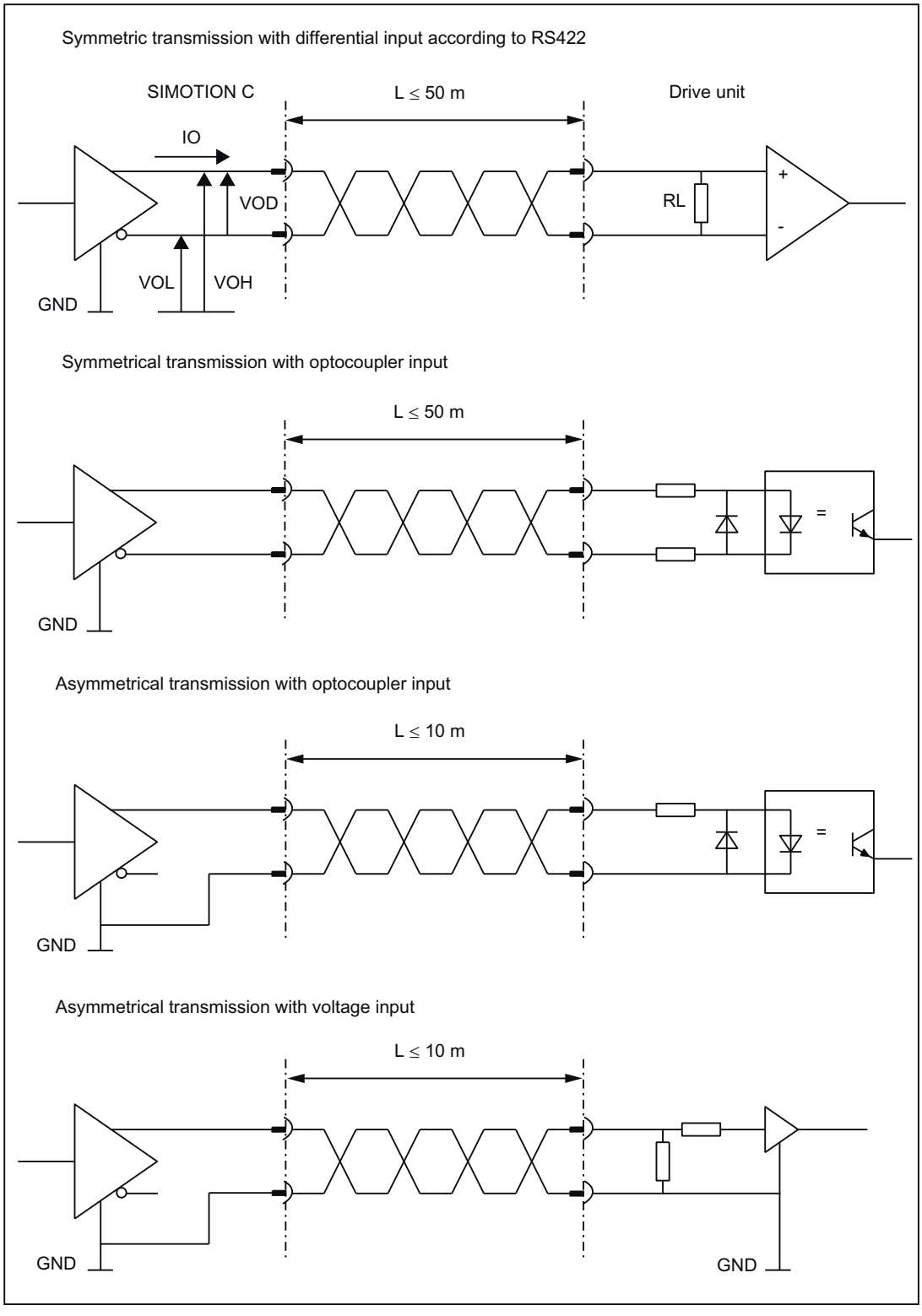


Figure 9-27 Possible protective signal circuits for the stepper motor interface

Use as standard outputs (analog/digital output) for C240

The setpoint signals (connector X2, SETP1 to 4) can also be used on the C240 as four analog outputs. The analog output (X2) has a filter that can be switched off.

The default setting is "with filter (C230-2-compatible)". This setting is active regardless of whether the four outputs are used (actuating signal for axis of analog output). Output values are interpolated linearly via the servo cycle clock.

Procedure:

1. Select the **C240** in the rack.
2. Select the menu command **Edit > Object Properties** to open the **Properties C240 - (R0/S2)** dialog.
3. You can set the filter time in the **Onboard I/O** tab.

The controller enable contacts (connector X2, CTREN1 to 4) can also be used on the C240 as four digital outputs. These digital outputs are isolated relay contacts (NO).

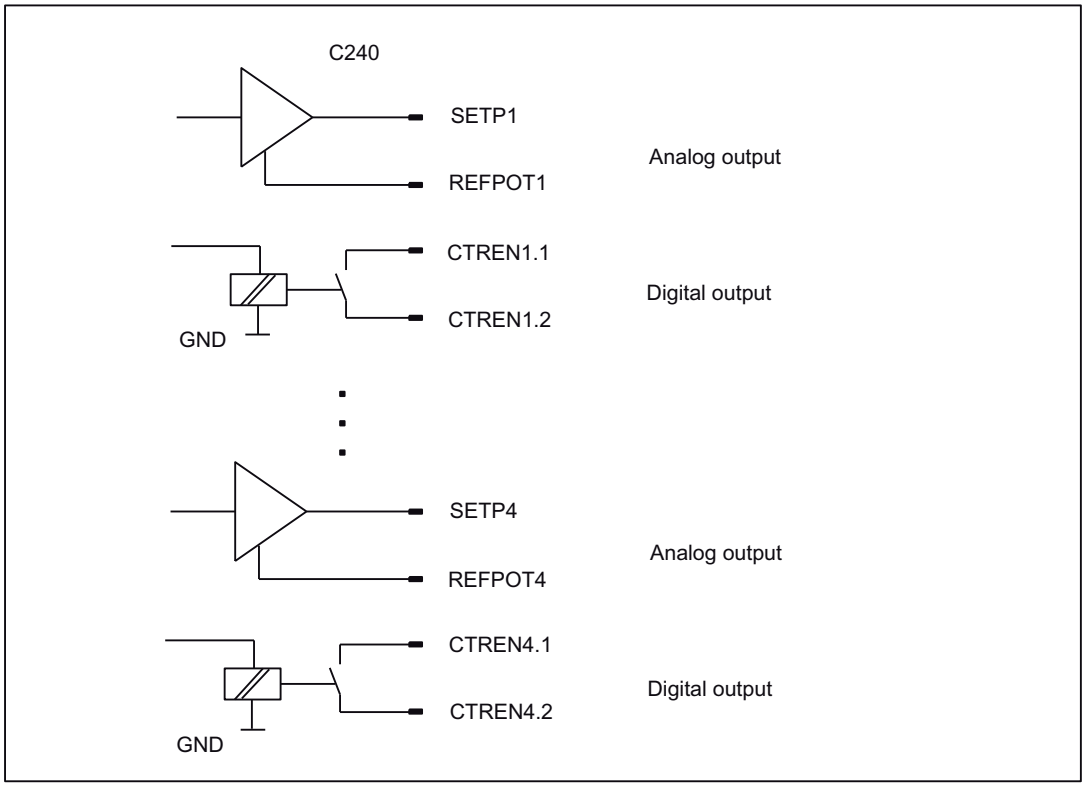


Figure 9-28 Use as standard outputs

See also

Overview (Page 6793)

9.1.4.6 Onboard measuring system interface (C230-2, C240)

Connectors to the encoder

A 15-pin Sub-D socket for the connection of incremental or absolute encoders (SSI) is provided for each axis.

With the C240, this interface (X3 to X6) can also be used as a counter input.

Position of connectors

This figure shows the mounting position and the designation of the connector on the module.

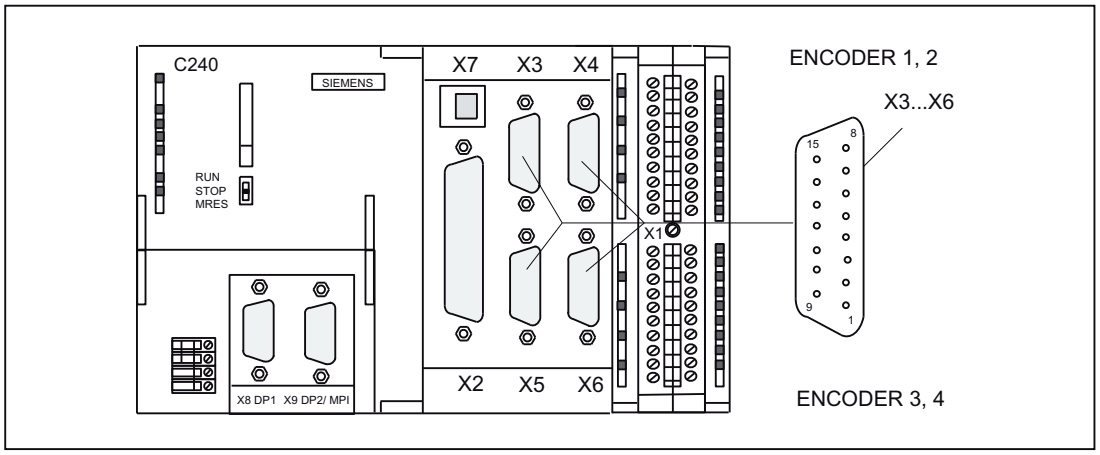


Figure 9-29 Position of connectors X3 to X6

Connector pin assignment

Designation:

X3, X4, X5, X6 - ENCODER 1 to 4
Assignment of ENCODER - axis channel:

X3 - axis channel 1
X4 - axis channel 2
X5 - axis channel 3
X6 - axis channel 4

Type: 15-pin Sub-D socket connector

Table 9-16 Assignment of connectors X3 to X6

Pin	Encoder		Type	Pin	Encoder		Type
	Incremental	Absolute			Incremental	Absolute	
1	Not assigned			9	MEXT		VO
2		CLS	O	10	Z		I
3		CLS_N	O	11	Z_N		I
4	P5EXT		VO	12	B_N		I

Pin	Encoder	Type	Pin	Encoder		Type
5	P24EXT	VO	13	B		I
6	P5EXT	VO	14	A_N	DATA_N	I
7	MEXT	VO	15	A	DATA	I
8	Not assigned					

Signal names

Table 9-17 Measuring system interface signal names

Signal name	Meaning
A, A_N	Track A non-inverted and inverted (incremental encoder)
B, B_N	Track B non-inverted and inverted (incremental encoder)
Z, Z_N	Zero mark non-inverted and inverted (incremental encoder)
CLS, CLS_N	SSI clock shift non-inverted and inverted (absolute encoder)
DATA, DATA_N	SSI data non-inverted and inverted (absolute encoder)
P5EXT	+5 V supply
P24EXT	+24 V supply
MEXT	Supply ground

Signal type

VO - voltage output (supply)
O - output (5 V signal)
I - input (5 V signal)

Types of encoder that can be connected

Both rotary (shaft encoders, angle measurement systems) and linear (linear encoder, linear measurement systems) measuring systems can be used. These can be built on the machine/system (incremental encoder) or integrated in the motor (rotor shaft angle encoder).

The following table gives you an overview of the encoders that can be connected and what you should take into account here.

Table 9-18 Encoders which can be connected

Encoder	Connection to X3 to X6 of the C230-2/C240
Rotary encoders • Incremental encoders with 5 V encoder power supply and TTL/RS422 interface • Incremental encoders with 24 V encoder power supply and TTL/RS422 interface • Absolute encoders (single/multi-turn) with 24 V encoder power supply and SSI interface • Shaft position encoders with 5 V encoder power supply and TTL/RS422 interface • Rotor position encoder with SINE signals • Resolver	• Direct • Direct • Direct • Direct • Via SIMODRIVE drive control using incremental shaft encoder (WSG) interface • Via SIMODRIVE resolver control using incremental shaft encoder (WSG) interface
Linear encoders • Length measurement systems with 5 V encoder power supply and TTL/RS422 interface • Linear encoders with SINUSOIDAL signals	• Direct • Via EXE (external pulse shaper electronics)

Note

If you operate drives via the PROFIBUS DP, you do not need to connect any encoders to this interface. The encoders are connected directly to the drive.

Encoder emulation (incremental shaft encoder (WSG) interface)

If the drive unit is equipped with encoder emulation, this can be connected instead of an encoder. The drive control analyzes the information from a rotor shaft angle encoder and provides information on the actual position to this interface by emulating the signals of an incremental encoder.

Note

Please note the drive manufacturer's wiring specifications. As the encoder interface of the C230-2, C240 is non-isolated, it may be necessary to take special measures for EMC on a case-by-case basis.

Encoder properties

Incremental encoder

Table 9-19 Incremental encoder properties

Property	Condition
Transmission procedure:	Differential transmission using 5 V rectangular signals (as in RS422 standard)
Output signals:	Track A as non-inverted and inverted signal (U_{a1}, $\overline{U_{a1}}$)
	Track B as non-inverted and inverted signal (U_{a2}, $\overline{U_{a2}}$)
	Zero signal Z as non-inverted and inverted signal (U_{a0}, $\overline{U_{a0}}$)
Max. output frequency:	1 MHz
Phase shift of Track A to Track B:	90° ± 30°
Current consumption:	max. 300 mA

Signal forms of incremental encoders

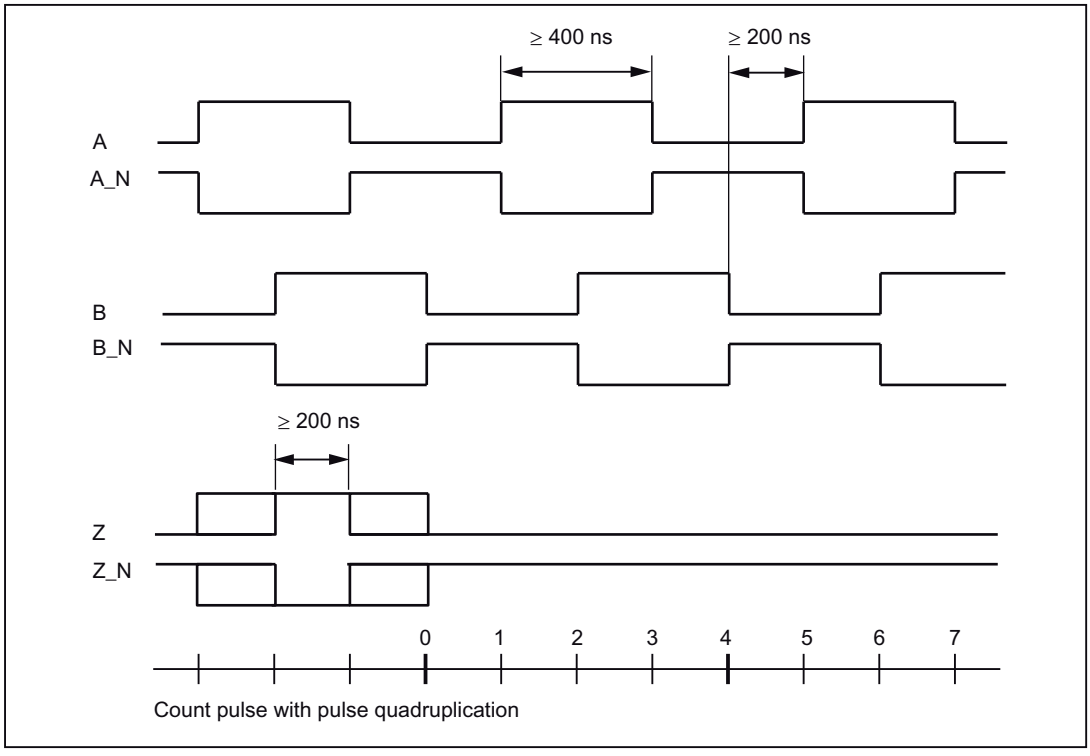


Figure 9-30 Signal forms of incremental encoders

Note

If you want to connect incremental encoders without zero signals, then you must connect the corresponding pins in the connector with the encoder power supply:

Pin 10 with Pin 7 or 9 (MEXT)
Pin 11 with Pin 4 or 6 (P5EXT)

Encoder connection via EXE

Encoders or EXEs (external pulse shaper electronics - for the connection of linear position encoders) that can be connected directly must fulfill the above conditions:

Absolute encoder (SSI)

Table 9-20 Properties of absolute encoders (SSI)

Property	Features
Transmission procedure:	Synchronous serial interface (SSI) with 5 V differential signal transmission (as in RS422 standard)
Output signal:	Data as non-inverted and inverted signal
Input signal:	Clock shift as non-inverted and inverted signal
Format:	Single/multi-turn
Resolution:	max. 25 bits
Max. transmission rate:	1.5 Mbits/s
Current consumption:	max. 300 mA

Configuration of absolute value encoders (SSI)

The configuration data of the TO axis and/or TO external encoder must perfectly match the parameters of the SSI encoder. Encoders which transmit a greater number of data bits than are to be read in the set SIMOTION C message length cannot be connected. The configuration data can be set within the range of the maximum values specified in the following message profiles (fir tree or right-justified).

Fir tree profile

The encoder always transmits the number of revolutions with the first 12 cycles. The configured encoder pulses per revolution is monitored by the encoder driver for the configured message length less these 12 cycles. The maximum configurable data width is monitored for the number of bit digits needed for representation of the configured encoder pulses per revolution in the telegram plus these 12 cycles.

Configuration data:

Message length:	13/21/25 cycles can be selected (asymmetric fir tree profile)
Encoder pulses per revolution:	≤ $2^{(\text{message length } -12)}$
Data width:	≤ 12 + (log encoder pulses per revolution / log 2)
	Note: The number of bit digits needed for representation of the encoder pulses per revolution is calculated using (log encoder pulses per revolution / log 2).

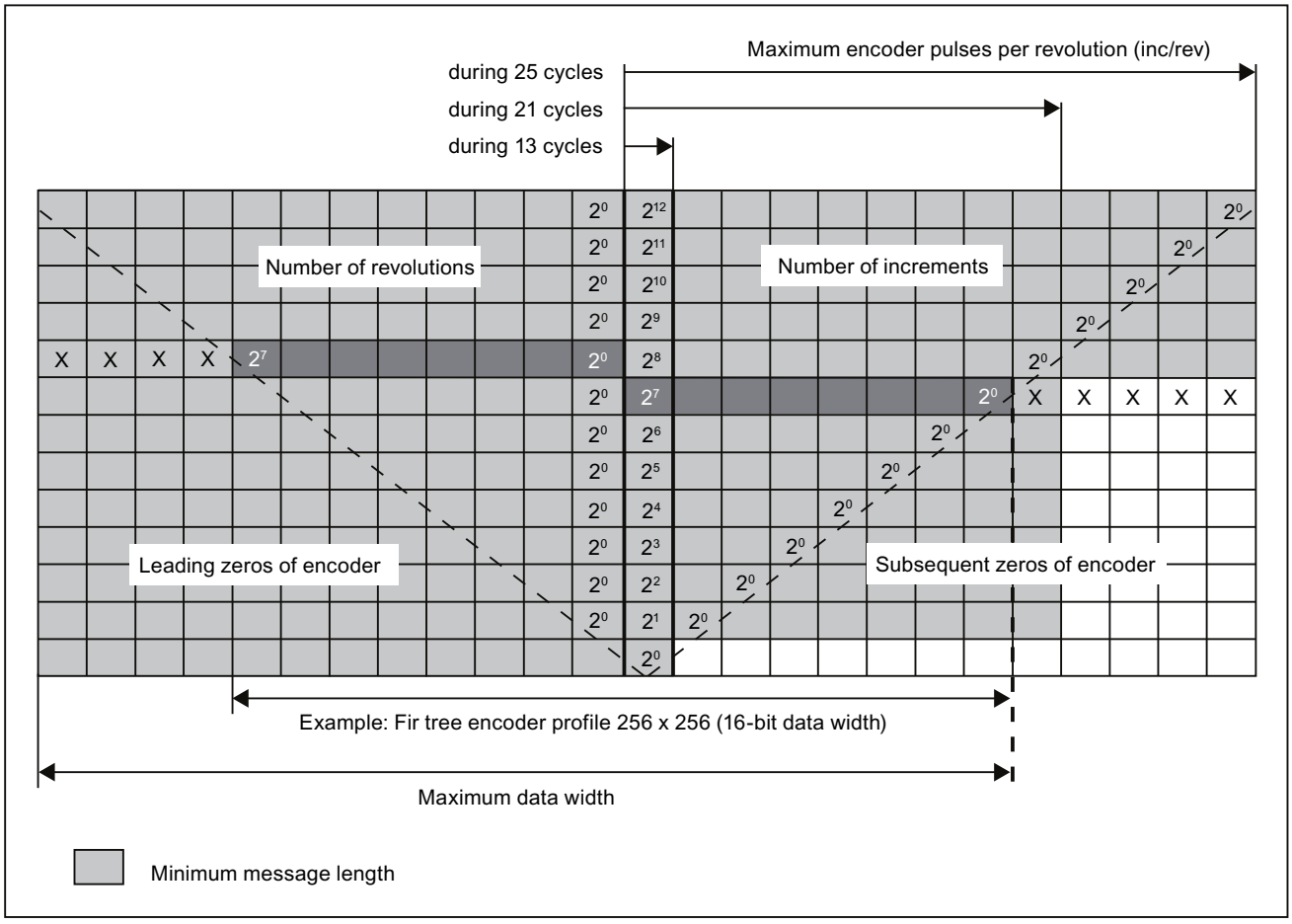


Figure 9-31 Fir tree profile

Note

Please note the following:

- Bit 2^0 of the number of revolutions is fixed on the 12th cycle, followed on the 13th cycle by the most significant bit of the number of increments/revolution.
- The leading and subsequent data bits transmitted by the encoder (or zeros) are not evaluated by SIMOTION C.
- The encoder used in the example should be configured as follows:
  - Encoder pulses per revolution: 256
  - Data width: 16
  - Message length: 21 bits (or 25 bits)
  - Message profile: Fir tree

Right-justified profile

The most significant data bit is transmitted by the encoder with the first cycle. The configured data width is monitored by the encoder driver depending on the configured message length. The maximum encoder pulses per revolution is monitored depending on the data width configured.

Configuration data:

Message length:	13/21/25 cycles can be selected
Encoder pulses per revolution:	$\leq 2^{(\text{Data width})}$
Data width:	≤ message length

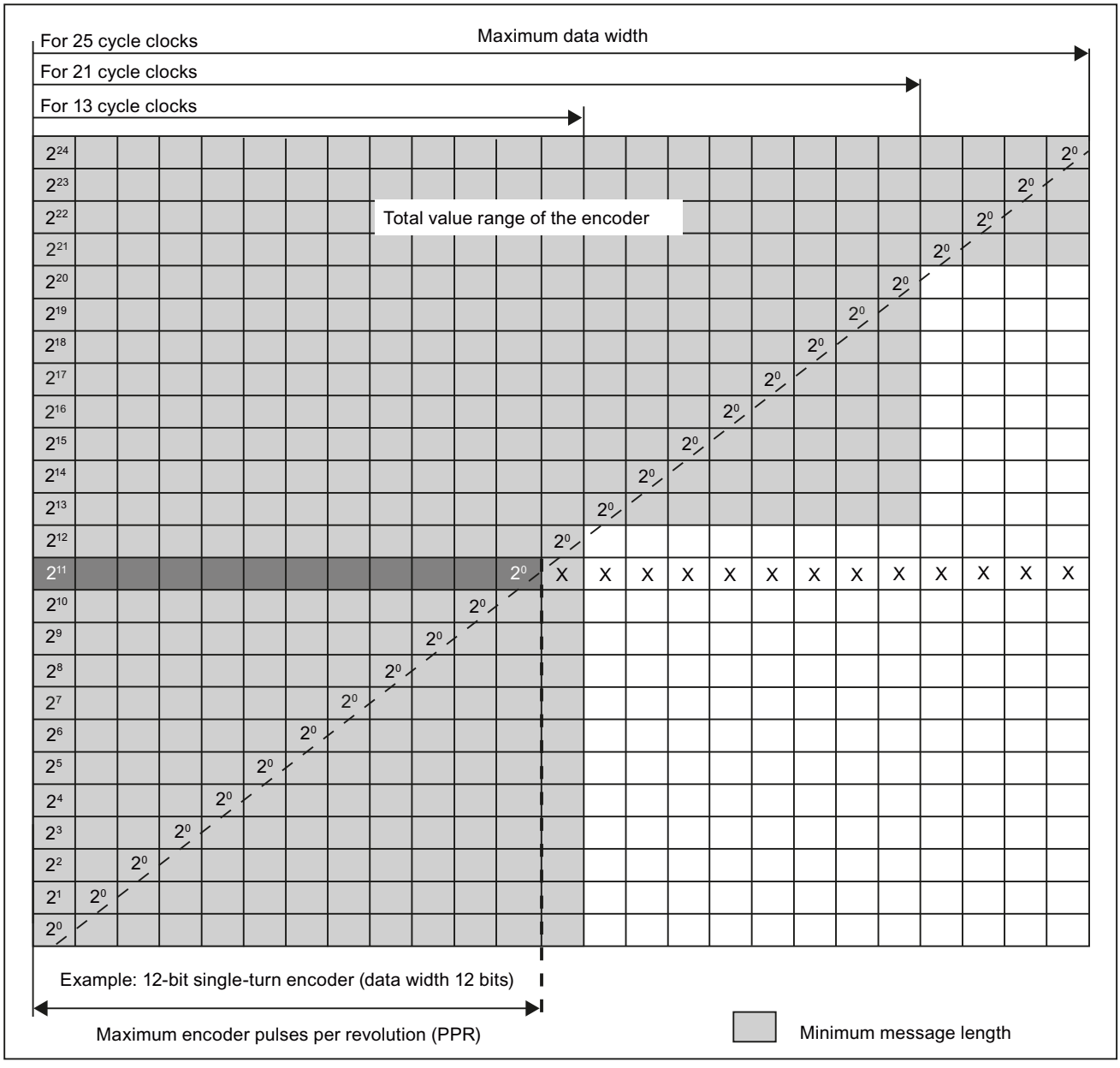


Figure 9-32 Right-justified profile

Note

Please note the following:

- The most significant data bit for the encoder's entire range of values is transmitted by the encoder with the first cycle.
- Subsequent data bits transmitted by the encoder (or zeros) are not evaluated by SIMOTION C.
- The encoder used in the example should be configured as follows:
  - Encoder pulses per revolution: 4096
  - Data width: 12
  - Message length: 13 bits (or 21 bits or 25 bits)
  - Message profile: Right-justified

Encoder supply 5 V

The 5 V supply voltage for the encoders is generated inside the module and is therefore present at the Sub-D socket. This means you can supply the encoders via the connecting cable without the need for additional wiring. The voltage supplied is protected electronically against short circuits and thermal overload, and it is monitored. The encoder supply is not isolated from the load power supply to the module.

Encoder supply 24 V

For encoders with an operating voltage of 24 V, the 24 VDC power is supplied to the Sub-D sockets. This means you can supply the encoders via the connecting cable without the need for additional wiring. The voltage supplied is protected electronically against short circuits and thermal overload, and it is monitored. The encoder supply is not isolated from the load power supply to the module.

Behavior of the integrated measurement electronics

Pulses from connected incremental encoders are quadrupled (for pulse quadruplication, see figure "Signal shapes of incremental encoders").

The actual encoder values are latched servo-synchronously in the C230-2/C240.

Note

With certain selected set values for the position control cycle clock, the measured value sampling in the C240 takes place at time Ti before the position control cycle clock time, see "Actual value latch time" table, in chapter Technical data (Page 6882). This behavior is similar to that of PROFIBUS DP encoders.

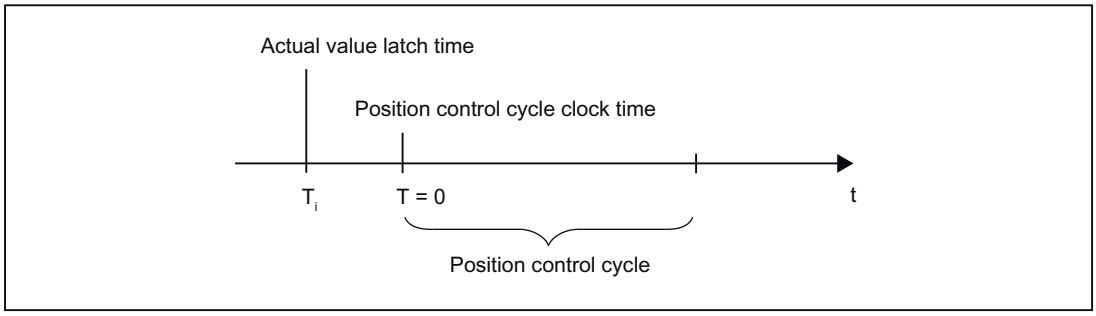


Figure 9-33 Position control cycle

Connecting cable to the encoder

The maximum cable length depends on the specification for the encoder supply and the baud rate. For problem-free operation, you must not exceed the following values when using SIEMENS preassembled connecting cables (see *Catalogs PM 21/NC 60/ST 70*):

Table 9-21 Maximum cable lengths, depending on the encoder power supply

Supply voltage	Encoder supply voltage range	Current consumption	Max. cable length
5 VDC	4.75 V to 5.25 V	< 300 mA	25 m
5 VDC	4.75 V to 5.25 V	< 210 mA	35 m
24 VDC	20.4 V to 28.8 V	< 300 mA	100 m
24 VDC	10 V to 30 V	< 300 mA	250 m

Table 9-22 Maximum cable lengths, depending on the baud rate

Encoder type	Frequency°	Max. cable length
Incremental encoder	1 MHz	10 m
	500 kHz	35 m
	300 kHz	100 m
Absolute encoder (SSI)	1.5 Mbits/s	10 m
	187.5 Kbyte/s	250 m

9.1.4.7 Possible uses of onboard drive and measuring system interface in the application (C230-2, C240)

Overview

The onboard drive interface and the measuring system interface can be used in the application for the technology objects **TO axis** and **TO externalEncoder**, and for **I/O variable**.

The table below lists the combination possibilities for the C230-2/C240.

Table 9-23 Combination possibilities for the C230-2/C240

Application	Axis channel	
	Output (X2)	Input (X3 to X6)
Position axis	x	x
Drive axis	x	-
External encoder	-	x
Also for C240:		
Standard output (I/O variables) • Analog output (PQW) • Digital output (PQ)	 x x	 - -
Standard input (I/O variables) [1] • Counter input (PIW)	 -	 x

[1] The incremental encoder used must not have a zero pulse

Note

A standard output and a drive **cannot** be used simultaneously on one axis channel.

The input (X3 to X6) used as a counter input cannot be used simultaneously as an encoder input.

When a project is created and when a project is downloaded, a consistency check of the permitted combinations per axis channel is automatically performed.

This results in the following possibilities in the application for the C240, for example:

- Configuration of hydraulic axes on the onboard resources of the C240
- Use of unassigned axis channels of the C240 as I/O variable for the user program
- Use of the analog outputs of the C240 as unassigned process outputs
  Note: The resolution and the characteristic curve of the analog output of the C240 differ from those of a SIMATIC S7 controller. The following table lists the digital values and their associated analog values (characteristic curve).

Digital value (WORD data type)	Analog value
16#7FFF ≙ 32767	+10 V
16#0000 ≙ 0	0 V
16#8000 ≙ 32768	-10 V

See also

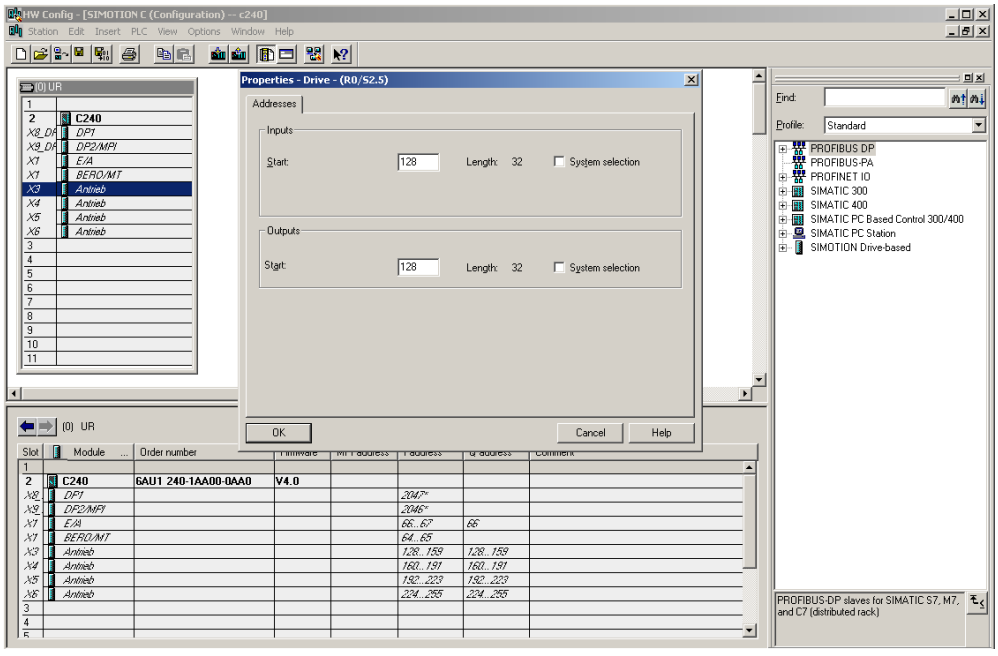
Connecting the drive units (Page 6827)

Configuration examples for C240:

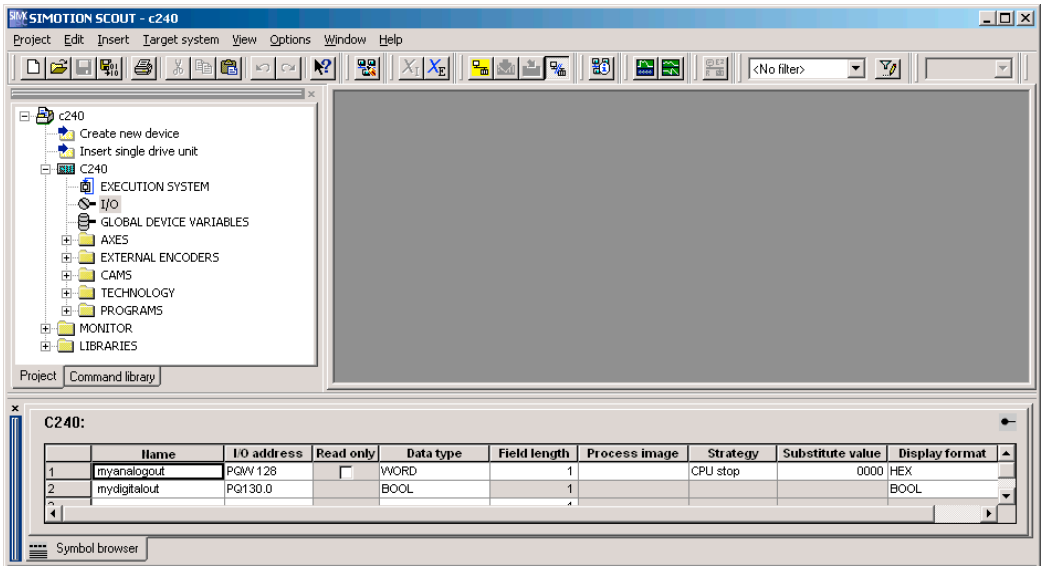
Use as standard output on X2

Configuration procedure in SIMOTION SCOUT:

1. Start address is specified in the **hardware configuration**



2. Specification of standard output (16-bit analog and 1-bit digital)

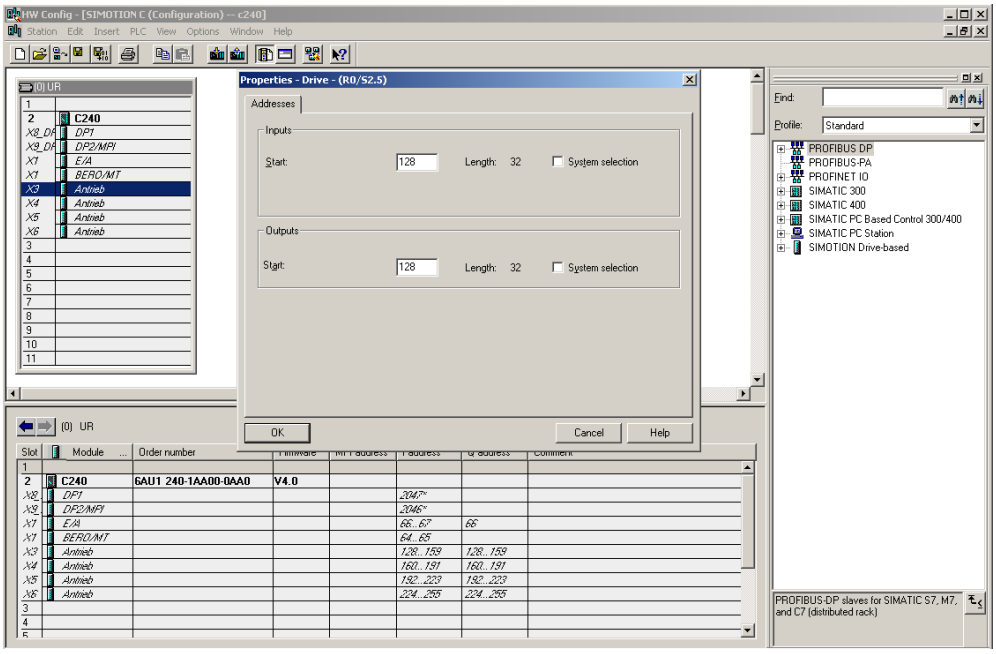


Use as counter input on X3 to X6

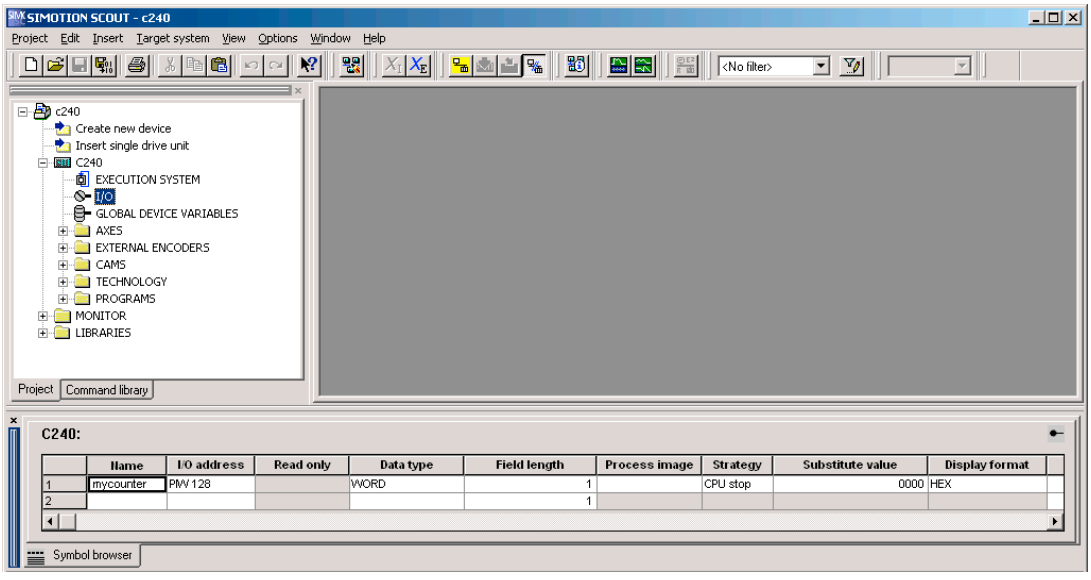
An available encoder input of an axis channel (e.g. for a speed-controlled axis) can be used as an input for a 16-bit up/down counter (90-degree pulse train of a connected TTL encoder, zero pulse not required). The counter value can be accessed by means of an I/O variable.

Configuration procedure in SIMOTION SCOUT:

1. Specification of start address in the **hardware configuration**.



2. Use of the available encoder input as an additional counter using an incremental encoder.



Note

Use of the drive and encoder interfaces as standard outputs and counter inputs is only possible via their direct I/O address. Symbolic assignment is not available for this application.

9.1.4.8 I/O interface

Front connector

Various encoders and actuators can be connected via digital inputs/outputs to the 40-pin front connector X1 with single-wire connection.

Position of the connector

The following figure shows the position of the front connector.

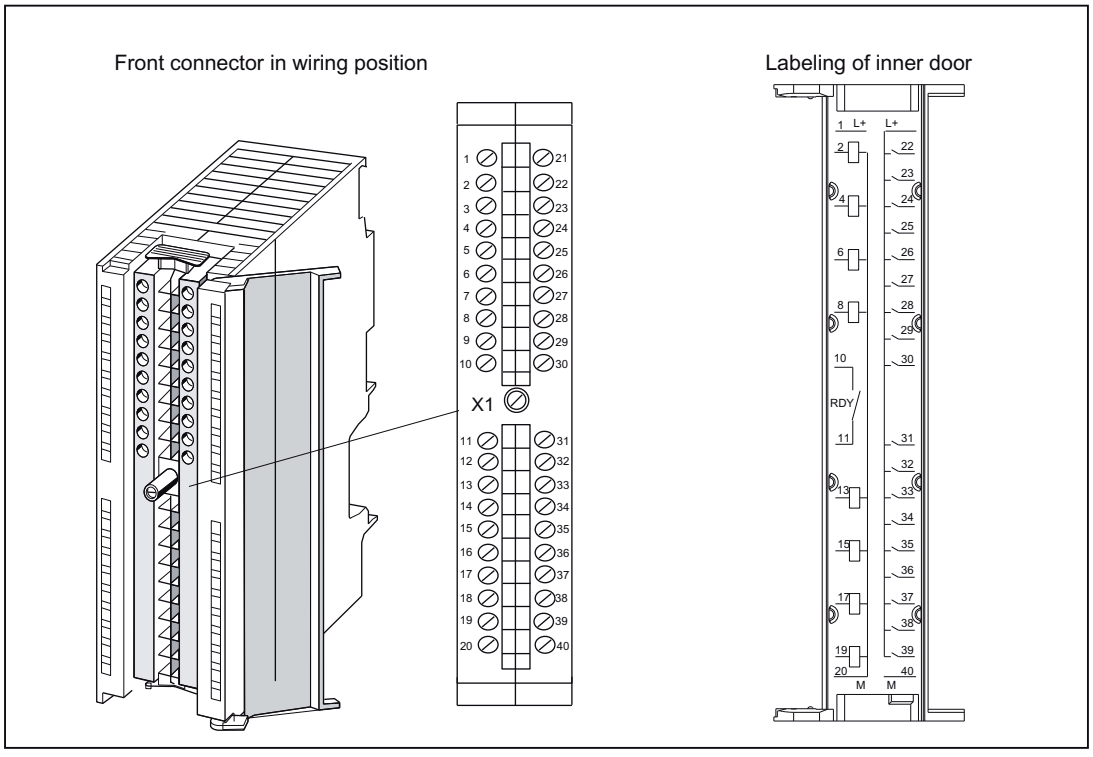


Figure 9-34 Position of connector X1

Wiring diagram and block diagram

The following figure shows the terminal diagram and block diagram for digital inputs/outputs on the SIMOTION C.

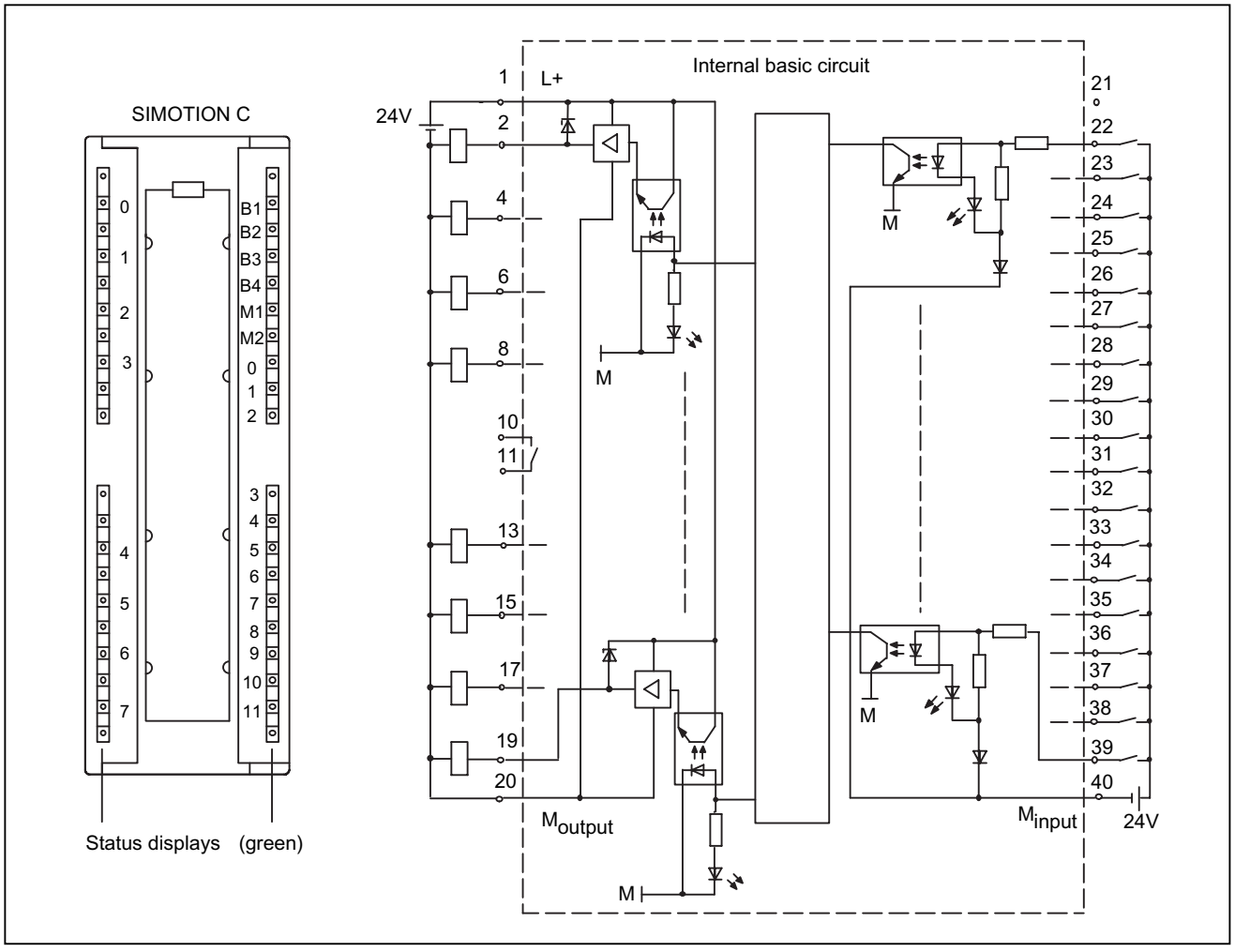


Figure 9-35 Terminal and block diagram of digital inputs/outputs on the SIMOTION C

Connector pin assignment

Connector designation: X1

Connector type: 40-pin S7 front connector for single-wire connection

Table 9-24 Front connector X1 pin assignment

Pin	Name	Type		Pin	Name	Type
1	L+	VI		21	Unassigned	
2	Q0	DO		22	B1	DI
3	Unassigned			23	B2	DI
4	Q1	DO		24	B3	DI

Pin	Name	Type		Pin	Name	Type
5	Unassigned			25	B4	DI
6	Q2	DO		26	M1	DI
7	Unassigned			27	M2	DI
8	Q3	DO		28	I0	DI
9	Unassigned			29	I1	DI
10	RDY.1	C		30	I2	DI
11	RDY.2	C		31	I3	DI
12	Unassigned			32	I4	DI
13	Q4	DO		33	I5	DI
14	Unassigned			34	I6	DI
15	Q5	DO		35	I7	DI
16	Unassigned			36	I8	DI
17	Q6	DO		37	I9	DI
18	Unassigned			38	I10	DI
19	Q7	DO		39	I11	DI
20	M_{output}	VI		40	M_{input}	VI

Signal names

Table 9-25 Signal names of the I/O interface

Signal name	Meaning
RDY.1 to 2	Ready (READY contact 1 to 2)
B1 to B4	• Inputs B1 to B4 for external zero mark signals (C230-2/C240) or • Measuring pulse inputs B1 to B4 for global measuring (C240/C240 PN) or • Digital inputs B1 to B4
M1, M2	• Measuring pulse inputs M1 and M2 for local measuring (C230-2/C240) or • Digital inputs M1 and M2
I0 to I11	Digital inputs 0 to 11
Q0 to Q7	Digital outputs 0 to 7
L+	Supply for digital outputs
M_{output}	Reference potential for digital outputs
M_{input}	Reference potential for digital inputs

Signal type

DI - digital input (24 V signal)
DO - digital output (24 V signal)
K - switching contact
VI - voltage input

⚠ DANGER

The 24 V power supply is to be designed as functional extra-low voltage with protective separation in accordance with EN60204-1, Section 6.4, PELV (with G ground).

Note

The connecting cable between the voltage source and the load current supply connector L+ and the associated reference potential M should not exceed a maximum length of 10 m.

Digital inputs (onboard)

The SIMOTION C has 18 digital inputs. These can also be interconnected via symbolic assignment using variables as of SIMOTION V4.2 for C240 and C240 PN.

Switches or proximity encoders (2- or 3-wire encoders) can be connected. Addresses are allocated in the hardware configuration.

The digital inputs can be used as:

- User-addressable process inputs (I0 to I11). The inputs are subject to a signal delay (see Technical data (Page 6882)) and are sampled in a cycle of 125 µs.

The following inputs can be used for special functions:

- C230-2, C240: As inputs for measuring pulses for local measuring (M1, M2)
  With a signal edge at the relevant input, the current actual values of one or more encoders connected to X3 to X6 are measured with positioning accuracy to determine lengths or distances.
  The assignment of inputs is **not** fixed; the special use is activated in the SCOUT engineering system during configuration of the Measuring Input TO via the measuring input number.
- C230-2, C240: As inputs for external zero mark signals (B1...B4)
  With a signal edge at the relevant input, the current encoder value of the associated axis is acquired with positioning accuracy in order to record the reference coordinates.
  The assignment of the inputs to the encoders is fixed, and the special use is activated in the SIMOTION SCOUT engineering system.
  - B1 → axis 1 (encoder at X3)
  - B2 → axis 2 (encoder at X4)
  - B3 → axis 3 (Encoder at X5)
  - B4 → axis 4 (encoder at X6)
- As inputs for measuring pulses for global measuring (B1 to B4) on the C240/C240 PN
  Alternatively to the "external zero mark signals" function, these inputs can be used on the C240 / C240 PN for global measuring.
  Note: The inputs for global measurement (B1 to B4) can be used in addition to the inputs for local measurement (M1, M2).
  With a signal edge at the relevant input, the current actual values of one or more encoders are measured with positioning accuracy in order to provide information for determining lengths or distances (possible with any encoders included in the project).
  The assignment of inputs is **not** fixed; the special use is activated in the SCOUT engineering system during configuration of the Measuring Input TO (see the configuration example below for global measuring).
  Up to two edges can be measured for each position-control cycle clock of the Measuring Input TO.
  The measured values must be read from the user program before they can be overwritten by a new measurement.
  Global measuring can be used for the following applications:
  - Several Measuring Input TOs on one axis/encoder where they can be active simultaneously
  - Several Measuring Input TOs are assigned to one measuring input (where one Measuring Input TO is interconnected to one measuring input and the remaining Measuring Input TOs are configured as listening measuring inputs)
    This functionality enables one measuring input to act on several Measuring Input TOs and thus on several axes / external encoders.
  - In addition to **single measuring**, **cyclic measuring** is also supported

– Measuring on virtual axes

Note

Local measuring (onboard) on SIMOTION C

A measurement is only undertaken when there are at least two servo cycles between the _enableMeasuringInput command being called and the real switching signal expected on the measuring input (positive/negative flank).

In order to reliably guarantee time-critical processes, the programming should be undertaken in a synchronous task, e.g. IPO-synchronous task. It may take up to two servo cycles for the measurement to be available in Messtaster.counterMeasuredValue1 after the status change in Messtaster.actualInputState (POSITIVE => NEGATIVE or vice versa).

When the measurement result is received, the measurement position is stored. Once the measurement has been taken, the state variable is set to TRIGGER_OCCURED, and the measured values can be evaluated using the measuredValue1 and measuredValue2 variables for two measured edges.

The counterMeasuredValue1 and counterMeasuredValue2 counter variables are defined for the measuredValue1 and measuredValue2 system variables and are automatically incremented by a value of one for each measurement input. New results can be traced immediately and can also be read from non-IPO-synchronous tasks.

You will find detailed information on this subject in the SIMOTION Function Manual *Output Cams and Measuring Inputs*.

Configuration example for global measuring

Configuration procedure in SIMOTION SCOUT:

The measuring input must be configured as follows:

1. Activate global measuring.
2. Assign to the input e.g. PI 64.0 or use symbolic assignment (e.g. C240.B1 [B1, X1.22]).

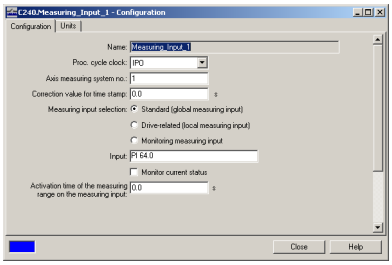


Figure 9-36 Configuration example for global measuring (without symbolic assignment)

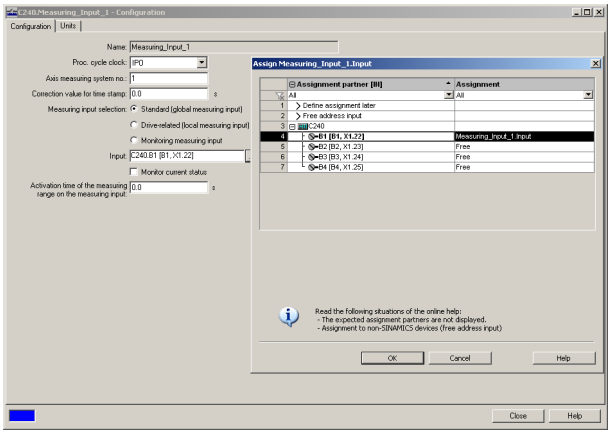


Figure 9-37 Configuration example for global measuring (symbolic assignment)

With a signal edge at the measurement input, the current actual value of the assigned encoder is recorded with an exact position.

Digital outputs (onboard)

Eight digital outputs (Q0 to Q7) are provided on the SIMOTION C. These can also be interconnected via symbolic assignment using variables as of SIMOTION V4.2 for C240 and C240 PN.

These fast outputs (onboard) can be used as freely addressable process outputs or as "fast output cams" (position switching signals). Addresses are allocated in the hardware configuration. The outputs are subject to a signal delay, see Technical data (Page 6882).

For repeat accuracy when using the outputs as fast output cams, see Versions of SIMOTION C (Page 6765).

READY output

The ready signal (RDY.1, RDY.2) is an isolated contact assembly (make contact).

The contact can be used for the safe shutdown of parts of the system, for example, through integration in the EMERGENCY STOP circuit.

The following table describes the states of the enables and outputs in the respective SIMOTION C operating states with an open or closed READY contact.

Table 9-26 States of enable signals and outputs:

Status of the READY contact	SIMOTION C	Status of enable signals and outputs:
Open	• During ramp-up • During memory reset • With fault condition • In STOP state • In STOPU state	• Controller enable deactivated • Analog outputs at 0 V • Digital outputs deactivated
Closed	• in RUN state	Enable signals and outputs controlled by user program and technology

Additional references

You can find information about TO outputCam and TO measuringInput in the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

9.1.5 Configuring and installing

9.1.5.1 General requirements

Overview

In this chapter we will explain how to design the mechanical configuration, prepare the SIMOTION components for installation and install them.

During the installation of SIMOTION C modules, you must pay attention to the electrical configuration. Therefore, also refer to the chapter Wiring (Page 6815).

Open equipment

These modules are open equipment. This means they may only be installed in housings, cabinets or in electrical service rooms that can be entered or accessed exclusively by means of a key or tool. Housings, cabinets or electrical service rooms may only be accessed by trained or authorized personnel.

9.1.5.2 Configuring an installation using SIMOTION C modules

Horizontal and vertical configuration

Design

You can install the rack either horizontally or vertically. The horizontal configuration should be used if possible.

Permissible ambient temperature

- Horizontal installation: 0 °C to 55 °C
- Vertical installation: 0 °C to 40 °C

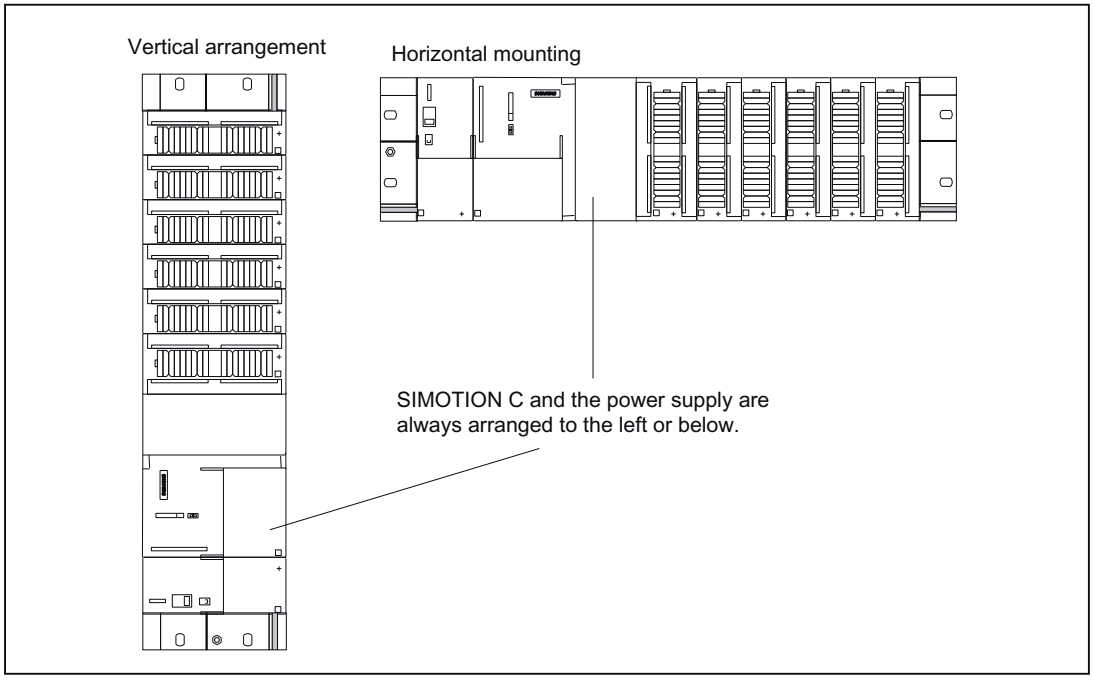


Figure 9-38 Horizontal and vertical configuration

Clearances

Rules

If you comply with the minimum clearances, you will:

- Ensure heat is dissipated from the modules
- Provide space to fit and remove modules

- Provide space to lay wiring
- Increase the mounting height of the rack to 205 mm!
  To guarantee the functionality, clearances of 40 mm must be maintained.

Note

If you use a shield connecting element, the dimensions stated are measured from the lower edge of the shield connecting element.

Clearances

The following figure shows the clearances between the individual racks and the clearance to adjacent equipment, cable ducts, cabinet walls, etc.

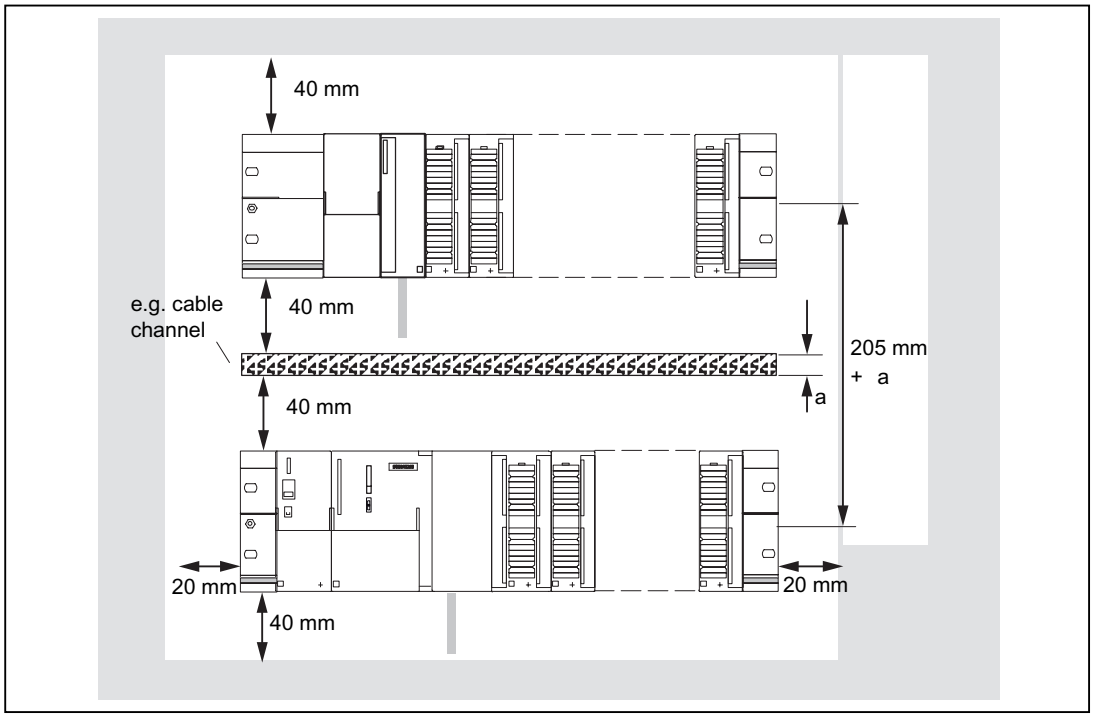


Figure 9-39 Clearances

Mounting dimensions of modules

Overview of mounting dimensions

The following table shows the mounting dimensions of modules.

Table 9-27 Mounting dimensions of modules

Modules	Module width	Module height	Maximum mounting depth
Power supply PS 307, 2 A	50 mm	125 mm, 185 mm with shield connecting element	130 mm or 180 mm with an open front cover of the SIMOTION C
Power supply PS 307, 5 A	80 mm		
Power supply PS 307, 10 A	200 mm		
SIMOTION C	200 mm		
Signal modules (SMs)	40 mm		
Function modules (FM)	40 mm or 80 mm		
Communications processors (CP)	40 mm		

Mounting rail lengths

Depending on the configuration you have chosen, you can use the following mounting rails:

Table 9-28 Mounting rails

Mounting rail	Usable length for modules	Comments
160 mm	120 mm	Mounting holes are provided
482.6 mm	450 mm	Mounting holes are provided
530 mm	480 mm	Mounting holes are provided
830 mm	780 mm	Mounting holes are provided
2,000 mm	Cut to required length	Mounting holes must be drilled

Layout of modules on a rack

Rules

The following rules apply with respect to the layout of the modules on a rack:

- Up to eight modules can be inserted to the right of the SIMOTION C.
- The number of plug-in modules is also limited by their power consumption from the backplane bus (see Technical data table for the individual modules in the *S7-300 Automation Systems, M7-300 Module Data* Manual).
  The total power consumption from the backplane bus of all modules that are mounted on a rack must not exceed 1.2 A.

This figure shows the order of the modules in an installation with eight I/O modules.

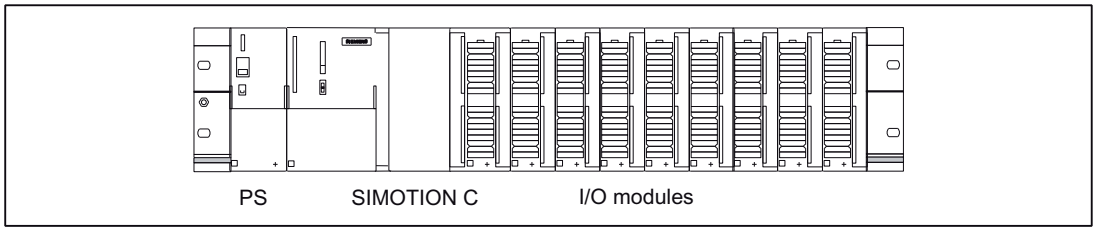


Figure 9-40 Layout of modules on a rack

Installation of FM STEPDRIVE

The FM STEPDRIVE modules can be installed, in addition to the eight SMs. They have no connection to the backplane bus and must therefore only be taken into consideration with regard to the module width. In order to prevent the backplane bus from being interrupted, the FM STEPDRIVE modules must always be configured as the last modules on the module rack.

Layout of modules on several racks

Overview

With the SIMOTION C, a 2-tier layout is possible.

Interface modules

Interface modules are required for the 2-tier layout which route the backplane bus from one rack to the other. The SIMOTION C is always located on rack 0.

Table 9-29 Interface modules

Interface module	Usable for ...	Article number
IM 365 SEND	Rack 0	6ES7 365-0BA01-0AA0
IM 365 RECEIVE	Rack 1	

IM 365 interface modules

The two IM 365 interface modules have a fixed connection via a 1-meter long connecting cable.

The total power consumption of the inserted I/O modules of both racks must not exceed 1.2 A; the power consumption from rack 1 is limited to 800 mA.

Rules

The following rules apply with respect to the layout of the modules on two racks:

- The interface module always occupies Slot 3 and is always left of the first signal module.
- No more than eight modules may be inserted per rack. These modules are always to the right of the interface modules.
- The number of inserted modules is limited by the permissible power consumption from the backplane bus. Total power consumption must not exceed 1.2 A (see Technical data table for each module in the manual *S7-300 Automation System, M7-300 Module Data*).

2-tier layout

The figure below shows the 2-tier layout with SIMOTION C.

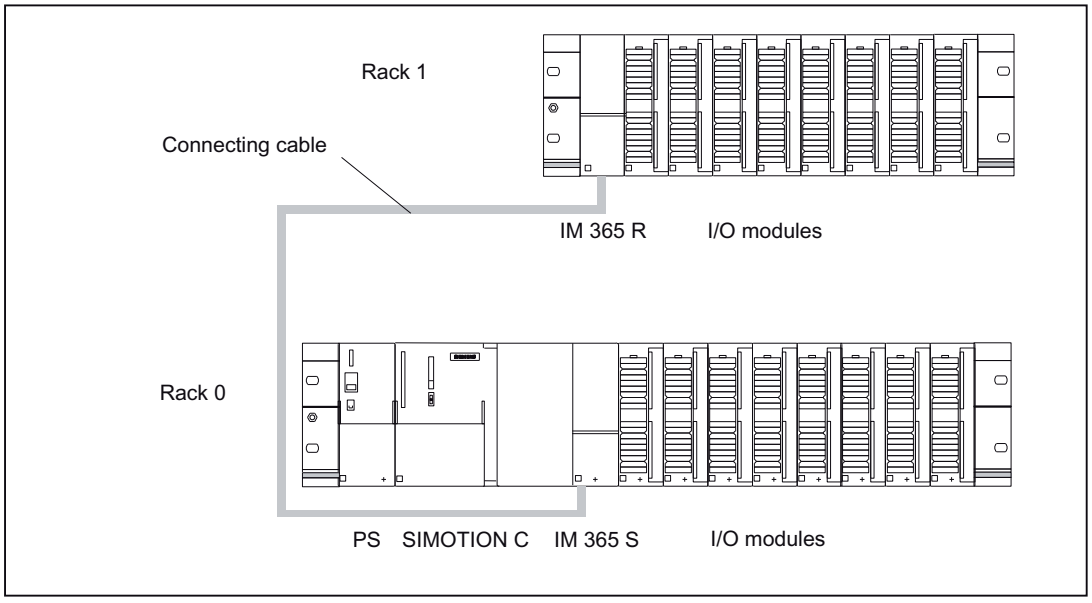


Figure 9-41 Layout of modules on two racks

9.1.5.3 Installing

Installing mounting rails

Installing a 2-meter mounting rail

You must prepare the 2-meter mounting rail for installation. Proceed as follows:

1. Shorten the 2-meter mounting rail to the required dimension.
2. Mark in
   - Four holes for mounting screws (dimensions: see table below)
   - One hole for a protective conductor mounting screw.

3. Is the mounting rail longer than 830 mm?
   If so: you must drill additional holes for more mounting screws to stabilize the mounting rail.
   Mark out these holes along the groove in the middle section of the rail (see the figure below).
   These additional holes should be spaced approximately every 500 mm.
   If not: no additional work required.
4. Drill the $6.5^{+0.2}$mm-diameter holes where marked for M6-size screws.
5. Fit an M6 screw to secure the protective conductor.

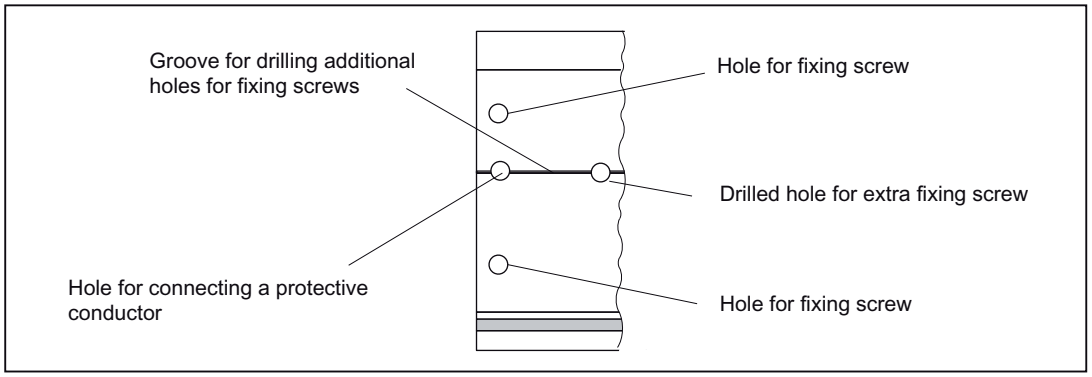
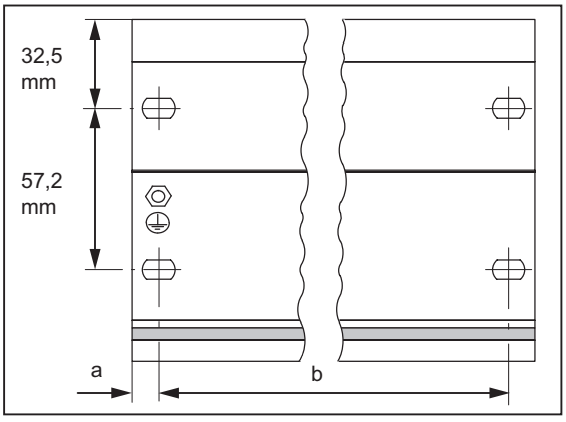
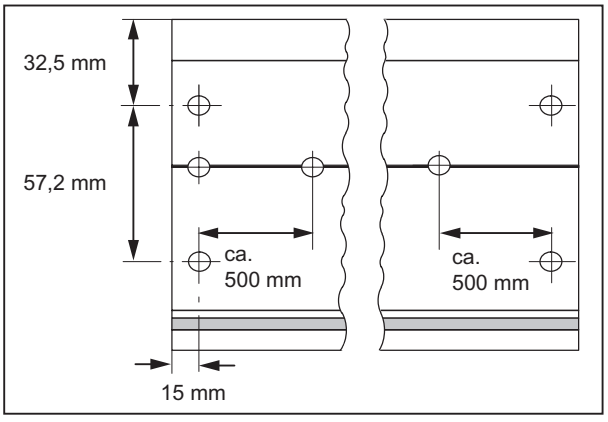


Figure 9-42 Mounting holes in a 2-meter mounting rail

Dimension drawing for mounting holes

The mounting hole dimensions for the mounting rail are shown in the table below.

Table 9-30 Mounting holes for rails

"Standard" mounting rail			2-meter mounting rail
			
Mounting rail length	**Distance a**	**Distance b**	-
160 mm	10 mm	140 mm	
482.6 mm	8.3 mm	466 mm	
530 mm	15 mm	500 mm	
830 mm	15 mm	800 mm	

Mounting screws

Choose one of the following types of screw to mount the mounting rail:

Table 9-31 Mounting screws

For	you can use...	Explanation
Outer mounting screws	M6 cylinder-head screw per ISO 1207/ISO 1580 (DIN 84/DIN 85)	Select a screw length that is appropriate to your configuration You also need 6.4 mm washers per ISO 7092 (DIN 433)
	M6 hexagonal screw per ISO 4017 (DIN 4017)	
Additional mounting screw (2-meter mounting rail only)	M6 cylinder-head screw to ISO 1207/ISO 1580 (DIN 84/DIN 85)	

Installing mounting rails

Install the mounting rails as follows:

1. Fit the mounting rail in a position that will allow enough room for the modules to be installed and for the heat to dissipate (a minimum of 40 mm above and below the mounting rail; see Figure "Clearance distances").
2. Screw the mounting rail onto the surface where it is to be affixed (screw size: M6). Is this substrate a grounded metal plate or a grounded equipment mounting plate?
   If so: make sure that there is a low-resistance connection between the mounting rail and the substrate. Use suitable electro-lubricant or contact washers with painted and anodized metals, for example.
   If not: no special action required.
3. Connect the mounting rail with the protective conductor. An M6 protective conductor screw is provided on the mounting rail for this purpose.
   Minimum cross-section of protective conductor line: 10 mm^2.

Note

Always make sure that there is a low-resistance connection to the protective conductor (see figure below). If the rack is mounted on a movable frame, for example, make sure that the line to the protective conductor is flexible.

PE connection

The figure below shows the proper way of connecting the protective conductor to the mounting rail.

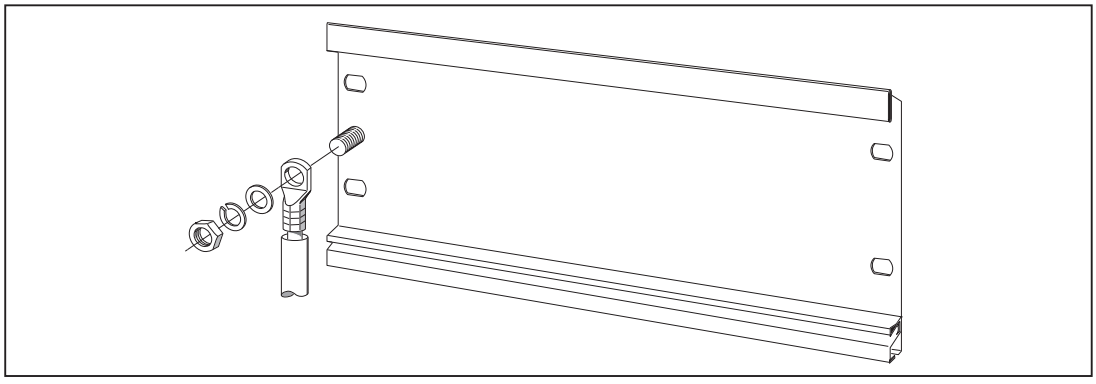


Figure 9-43 Protective conductor connection on the mounting rail

Fitting modules on the mounting rail

Accessories

Any accessories you need for installation are in the pack with the modules. Chapter Spare parts and accessories (Page 6892) contains a list of accessories and spare parts together with the corresponding Article Nos.

Table 9-32 Module accessories

Module	Accessories supplied	Explanation
SIMOTION C	One slot number plate	For the assignment of slot numbers
	Two keys (C230-2)	The key is used to operate the mode selector for the C230-2.
	One labeling plate	For labeling of integrated inputs and outputs of the SIMOTION C
Signal module (SM)	One bus connector	To provide the electrical connections between the modules
	One labeling plate	To label the inputs and outputs on the module

Sequence in which modules are affixed to the mounting rail

1. Power supply module
2. SIMOTION C
3. Signal module(s)

Installation sequence

The individual steps for the installation of the modules are described below:

1. Except for the SIMOTION C, each signal module is supplied with a bus connector. When plugging in the bus connectors, always start with the SIMOTION C.
   Take the bus connector from the next module and plug it into the bus connector of the SIMOTION C. (The bus connector is located on the rear side, see Figure "Position of interfaces and front panel elements").
   You must not plug a bus connector into the "last" module in the row.
2. Fit the modules by hooking them into position, push them against the left-hand module and lower them down into position.
3. Screw down the modules, applying a torque of 0.8 to 1.1 Nm.

After installation

Inserting the key (C230-2)

Once the C230-2 has been mounted on the mounting rail, you can insert the key in the STOP or STOPU position on the C230-2.

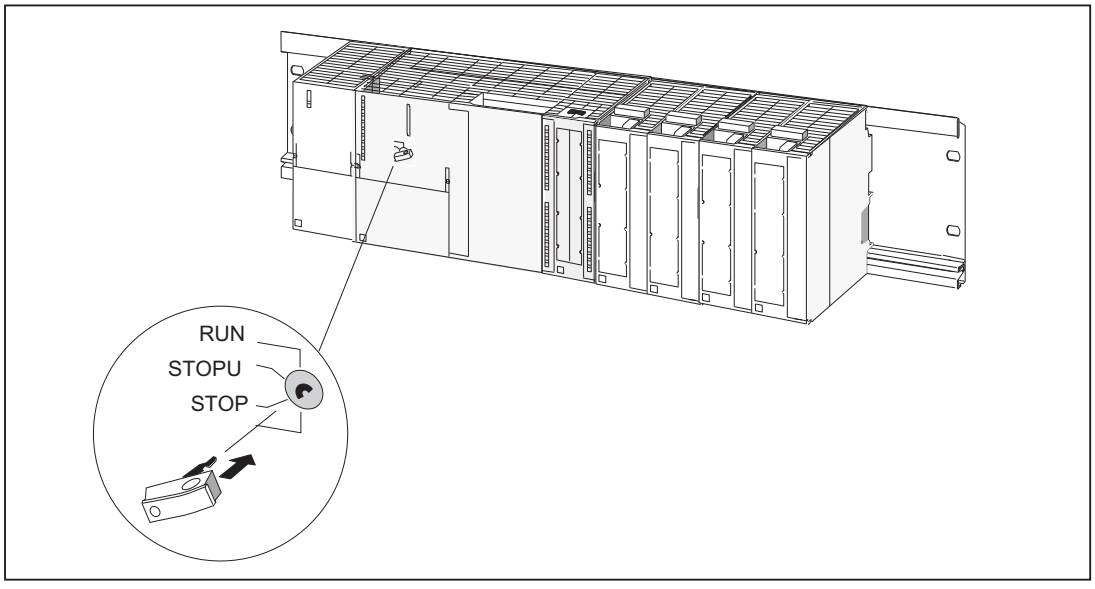


Figure 9-44 Inserting the key in the C230-2

Assigning the slot numbers

After mounting, you can assign each module a slot number. This eases the assignment of the modules to the configuration table in the *Engineering System*. The table below shows the slot number assignment.

Table 9-33 Slot numbers for S7 modules and SIMOTION C

Slot no.	Module	Comments
1	Power supply (PS)	-
2	SIMOTION C	-
3	Reserved	-
4	1. I/O module	To the right of SIMOTION C
5	2. I/O module	-
6	3. I/O module	-
7	4. I/O module	-
8	5. I/O module	-
9	6. I/O module	-
10	7. I/O module	-
11	8. I/O module	-

Inserting slot numbers

This figure shows the proper way of inserting the slot numbers. The slot number labels are included with the SIMOTION C.

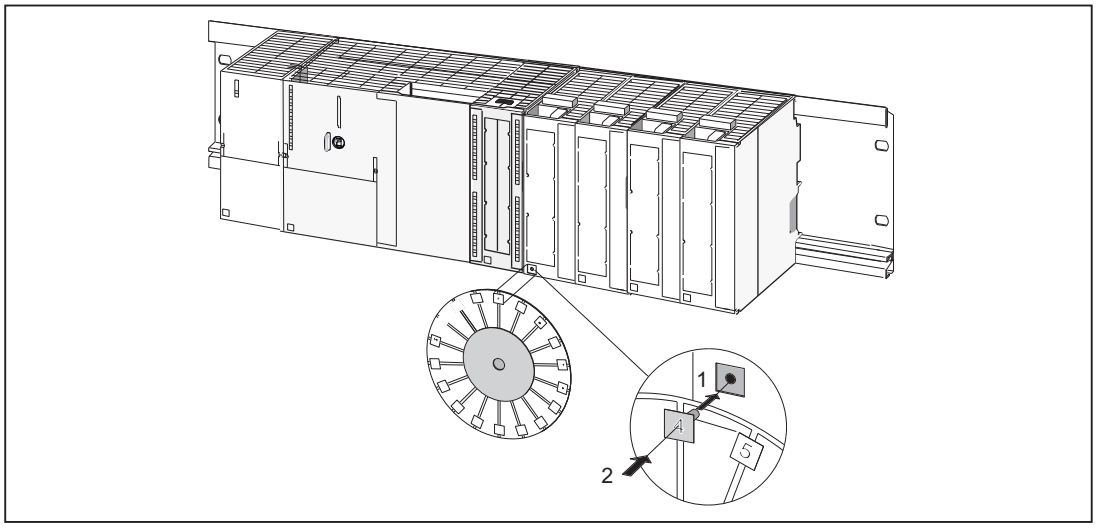


Figure 9-45 Inserting slot numbers on the modules

9.1.6 Connecting

9.1.6.1 Wiring

General requirements for wiring

Basic rules

Because of the wide range of uses of a SIMOTION C, only the basic rules for electrical installation can be included in this section. At a minimum, you must comply with these basic rules to ensure problem-free operation.

Safety regulations

In order to ensure safe operation of your equipment, implement the following measures, adapting them to suit your conditions:

- An EMERGENCY OFF concept in accordance with the generally accepted rules of current engineering practice (e.g. European standards EN 60204, EN 418, and similar).
- Additional measures for end position limiting of axes (e.g. hardware limit switches).
- Equipment and measures for protection of motors and power electronics in accordance with the installation guidelines of the used drive units.

In addition, in order to identify hazards, we recommend that a risk analysis be conducted on the entire system in accordance with the basic safety requirements set out in Appendix 1 of EU Machinery Directive 89/392/EEC.

Additional references

Please also note the information in the following sections of this manual:

- Guidelines on Handling Electrostatic Sensitive Devices (ESD guidelines)
- For additional information on installing a system containing S7-300 process I/O, refer to the "Wiring" section of the *S7-300 Automation System, Hardware and Installation, CPU Data* Manual.

Standards and specifications

When wiring the SIMOTION C, you must observe the appropriate VDE guidelines, in particular VDE 0100 and VDE 0113 for tripping devices and short-circuit and overload protection.

Configuring the electrical installation

General rules for operating a SIMOTION C

You must observe the following primary rules for integrating a SIMOTION C in an automation system or plant.

System startup after certain events

The following table identifies considerations required for startup of a system following certain events.

Table 9-34 System startup

If there is ...	Then ...
Startup after voltage drop or power failure	All hazardous operating conditions must be avoided. If necessary, force an EMERGENCY STOP.
Startup after releasing the EMERGENCY STOP apparatus	must always be controlled and defined.

Supply voltage

The following table identifies the considerations required for the supply voltage.

Table 9-35 Supply voltage

Scope	Requirement
Stationary systems without all-pole power disconnect switches	A power disconnect switch or a fuse must be provided in the electrical installation for the building.
Load power supply, power supply modules	The set rated voltage range corresponds to the local supply voltage.
All electric circuits	The fluctuation/deviation of the supply voltage from the rated value must be within the permissible tolerance (see Technical data for S7-300 modules).

24 VDC supply

The following table identifies the considerations required for the 24 V supply.

Table 9-36 24 V supply

Scope	Requirement	
buildings	external lightning protection.	Take lightning protection measures
24 V DC supply lines, signal lines	internal lightning protection	(e.g. lightning conductors).
24 V supply	Safe (electrical) isolation of extra-low voltage	

Protection from external electrical influences

The following table identifies the considerations required for protection against external electrical phenomena or faults.

Table 9-37 Protection against external electrical phenomena

Scope	Requirement
All plants or systems in which the SIMOTION C or S7-300 is installed	The equipment or system is connected to a protective conductor to discharge electromagnetic interference.
Supply, signal, and bus lines	The wiring arrangement and installation complies with EMC regulations.
Signal and bus lines	A wire or conductor strand break must not cause any undefined installation or system states.

Rules for current consumption and heat loss in a central configuration

The I/O modules on the I/O bus take the current required for their operation from the I/O bus as well as, when necessary, from an external load power supply.

- The current consumption of all I/O modules from the I/O bus must not exceed 1.2 A. This is the current that the SIMOTION C can supply on the I/O bus.
- PS 307 power supply modules are available for different loads (2 A, 5 A, 10 A). The selection of the appropriate power supply depends on the sum of the current consumption of the SIMOTION C, the connected I/O modules and any other connected loads fed by the load power supply.
- The power loss of all components used in a cabinet must not exceed the maximum amount that can be dissipated from the cabinet.
  Tip: When selecting the cabinet design, make sure that the temperature in the cabinet will not exceed the permissible ambient temperature for the installed components, even if the outside temperatures are high.

You will find information on the current consumption and heat loss of a module in the technical data of the corresponding modules.

⚠ DANGER

All interfaces of the SIMOTION C must operate only on circuits with safety extra-low voltage (SELV).

Pulling out and inserting I/O-bus I/O

I/O-bus I/O must not be pulled out or inserted during operation.

Note

The I/O bus will not work if I/O-bus I/O are pulled out during operation. The SIMOTION must be switched off and back on again when these are reinserted.

Overview of wiring diagram

C230-2/C240 with servo drive (analog connection)

The figure below shows how the individual components are connected to the C230-2/C240 and the servo drive (analog connection).

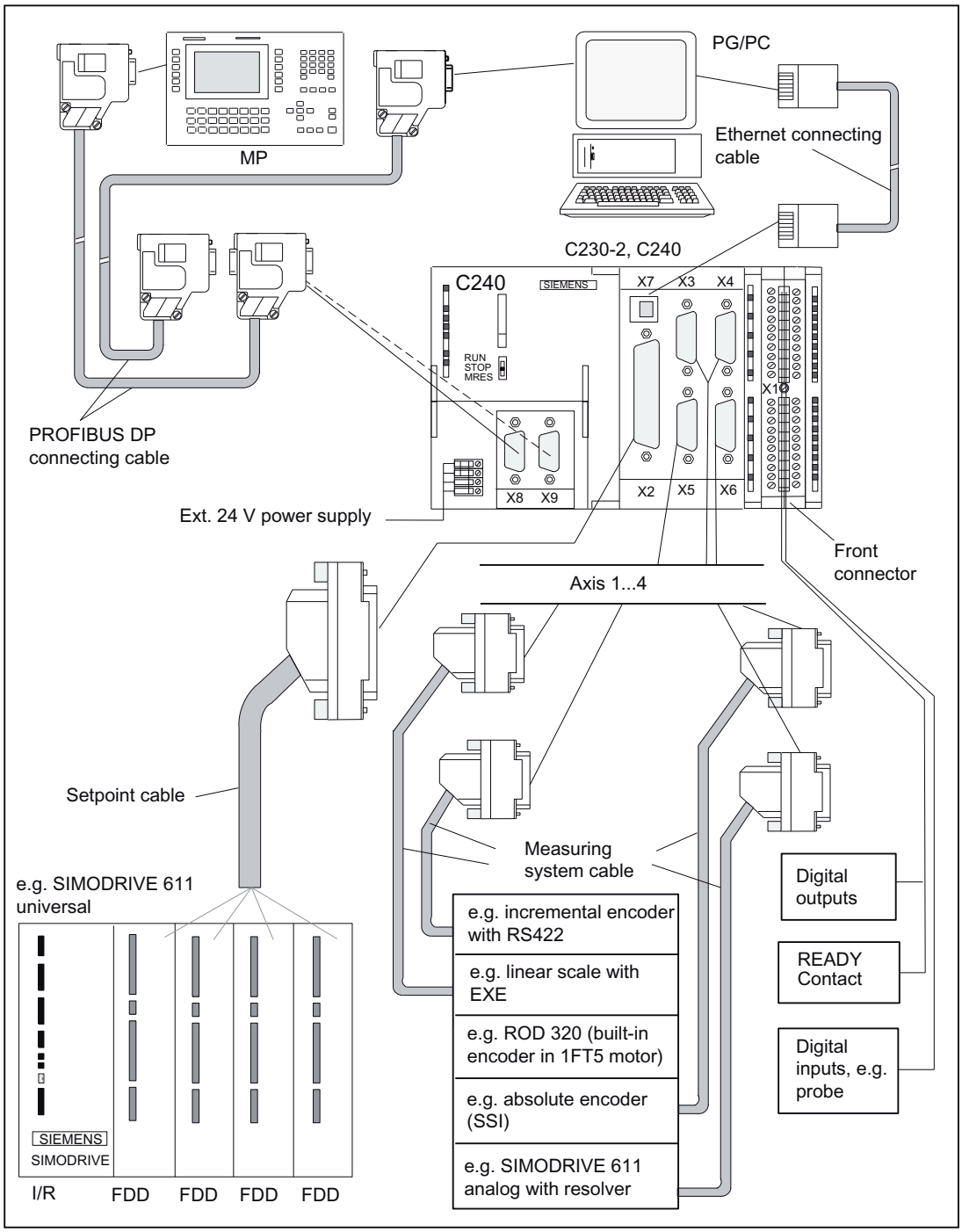


Figure 9-46 Overview of the cable connecting the C230-2/C240 to the servo drive (analog coupling) - example

SIMOTION C with servo drive (digital connection) via PROFIBUS DP

The figure below shows how the individual components are connected to the SIMOTION C and the servo drive (digital connection).

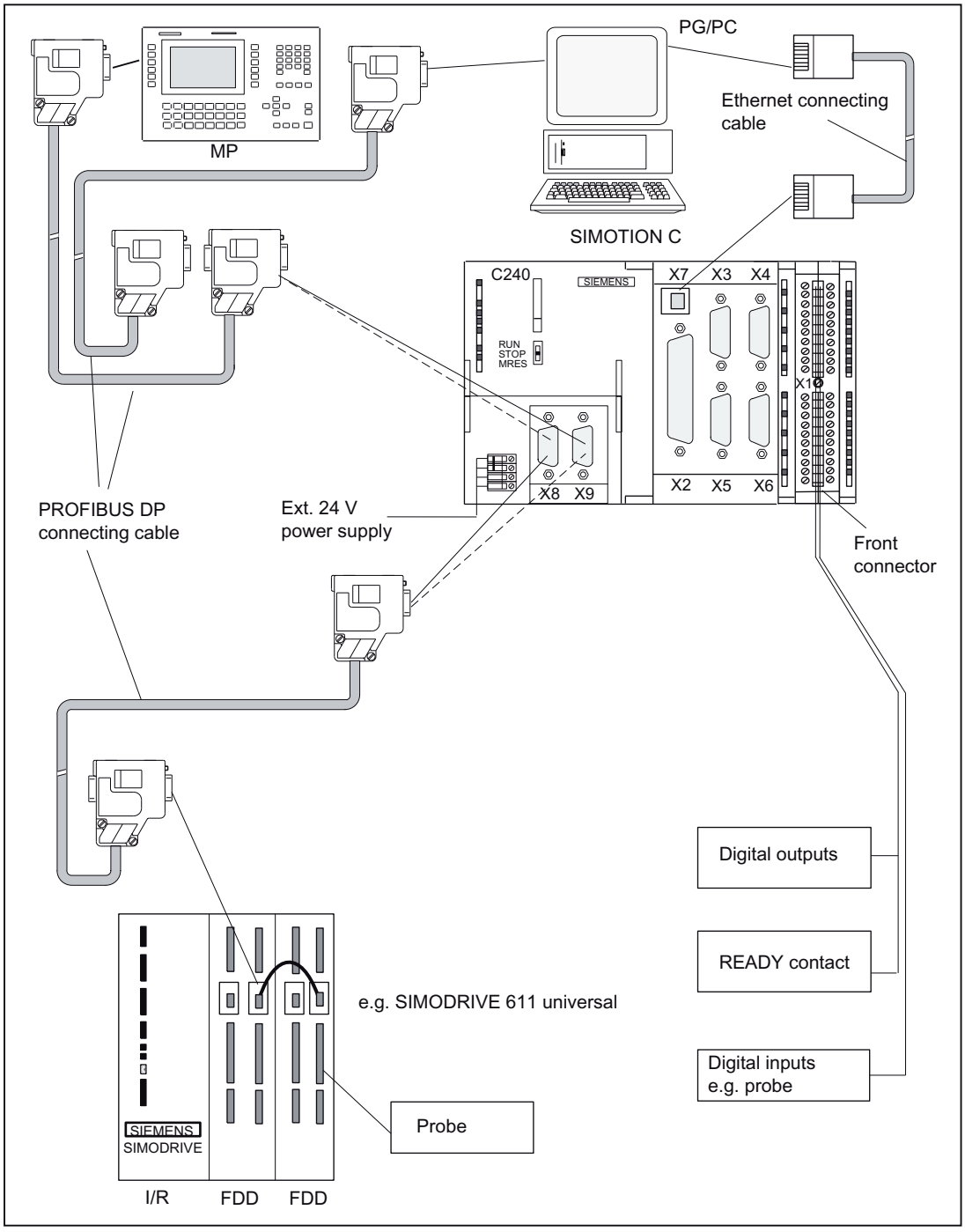


Figure 9-47 Overview of the cable connecting the SIMOTION C to the servo drive (digital connection) - example

C240 PN with servo drive (digital connection) via PROFINET IO

The figure below shows how the individual components are connected to the C240 PN and the servo drive (digital connection).

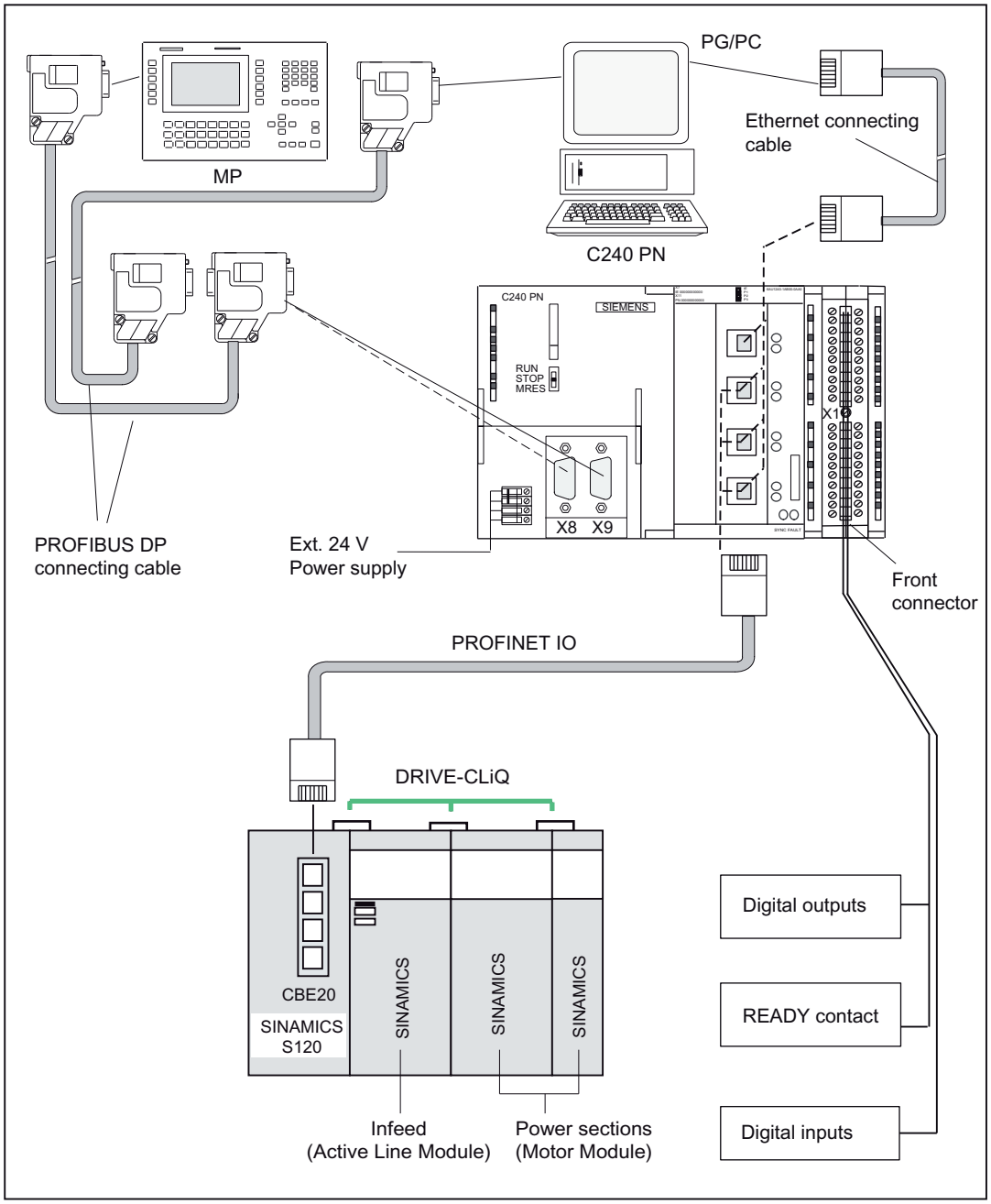


Figure 9-48 Overview of the cable connecting the C240 PN to the servo drive (digital connection) - example

C230-2, C240 with stepper drive

The figure below shows how the individual components of the multi-axis controller are connected to the C230-2, C240 and stepper drive.

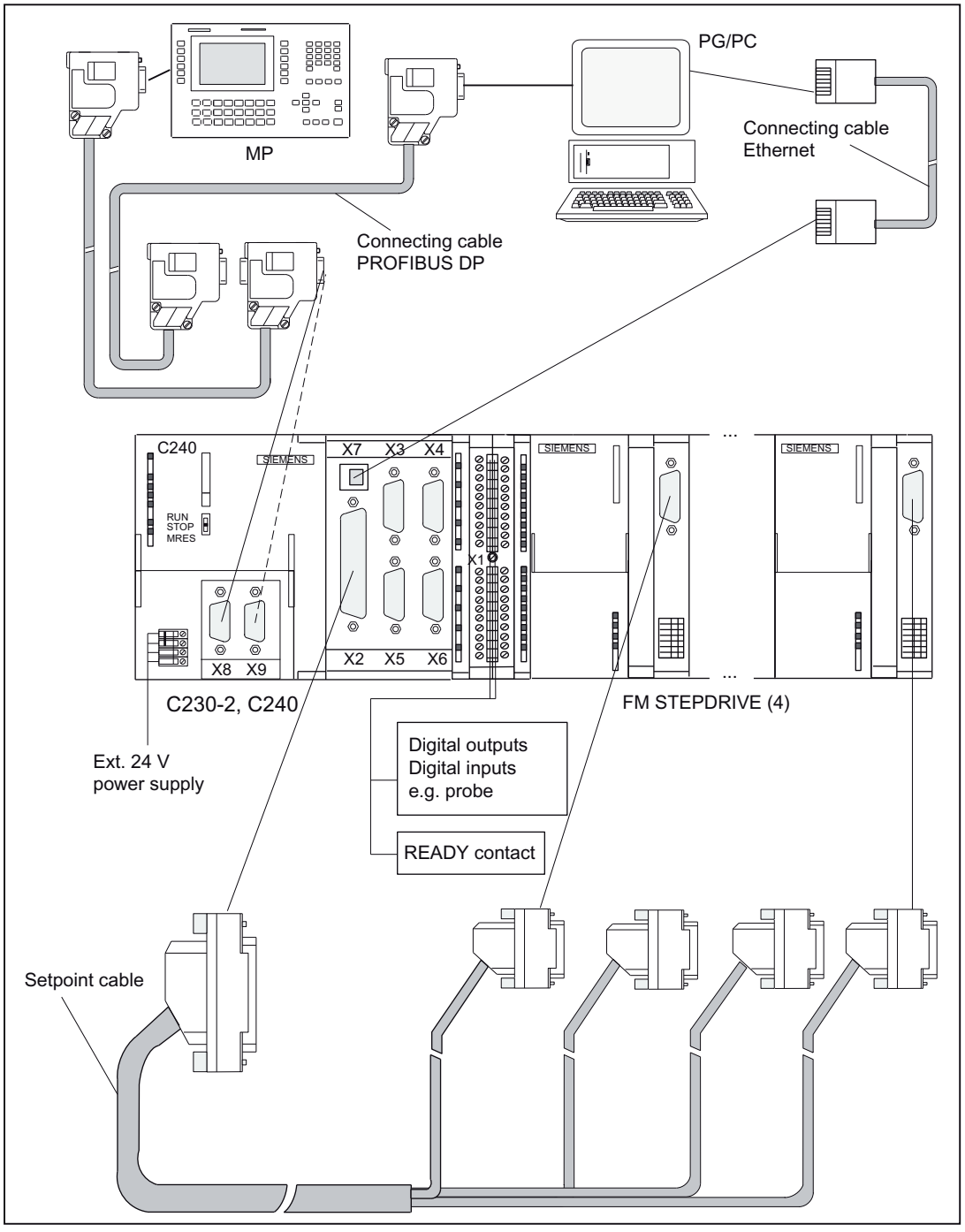


Figure 9-49 Overview of the cable connecting the C230-2/C240 to the stepper drive - example

Overview of connections - connecting cable

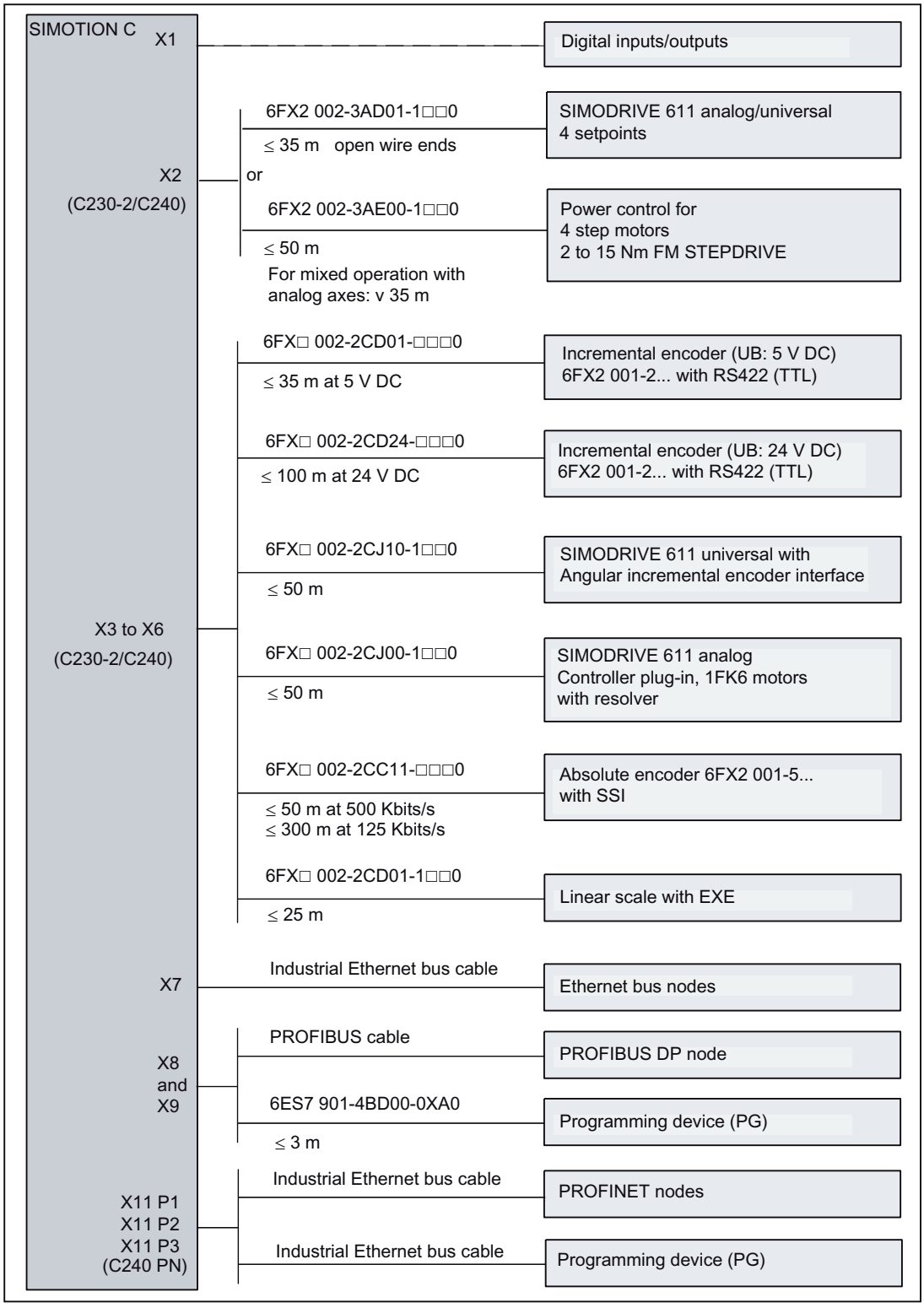


Figure 9-50 Overview of SIMOTION C connections

The setpoint and measuring system cables (see figure "Overview of connections") are available in various lengths.

See *PM 21 Catalog* / *NC 60 Catalog* / *ST 70 Catalog* / *CA 01 Catalog*

For information about the PROFIBUS cable or Ethernet bus cable, see chapter Networking (Page 6842).

For additional information about PROFIBUS DP, PROFINET IO and Ethernet, see *IK PI Catalog*

Front connector

For the wiring of the digital inputs/digital outputs, you need a 40-pole front connector (screw type or spring tension type). This must be ordered separately.

Article No.: Screw type 6ES7 392-1AM00-0AA0
Spring type 6ES7 392-1BM01-0AA0

See *ST 70 catalog/NC 60 catalog*

Connecting the power supply

Screw-type terminal block

The required 24 VDC load power supply is connected at the screw-type terminal block.

Properties of the load power supply

⚠ DANGER
The 24 VDC should be configured as functional extra-low voltage with safe isolation.

Note

The connecting cable between the voltage source and the load current supply connector L+ and the associated reference potential M should **not** exceed a maximum length of 10 m.

Table 9-38 Electrical parameters of load power supply

Parameters	min	max	Unit	Conditions
Voltage range - mean value	20,4	28,8	V	
Ripple		3,6	Vpp	
Transient overvoltage	35	35	V	500 ms duration 50 s recovery time
Rated current consumption		2,2	A	see "Connection values" table in chapter Technical data (Page 6882)
Starting current		8	A	

Pin assignment

The table below shows the pin connections on the screw-type terminal block.

Table 9-39 Assignment of the screw-type terminal block

Terminal	Pin assignment
⏚	Functional ground
M	Ground
L+	24 V DC
M	Ground

If you want to ground the reference potential, you must **not** remove the jumper between terminals M and functional ground on the SIMOTION C.

Line buffering

The PS 307 load current supplies from the S7-300 system guarantee mains buffering for 20 ms.

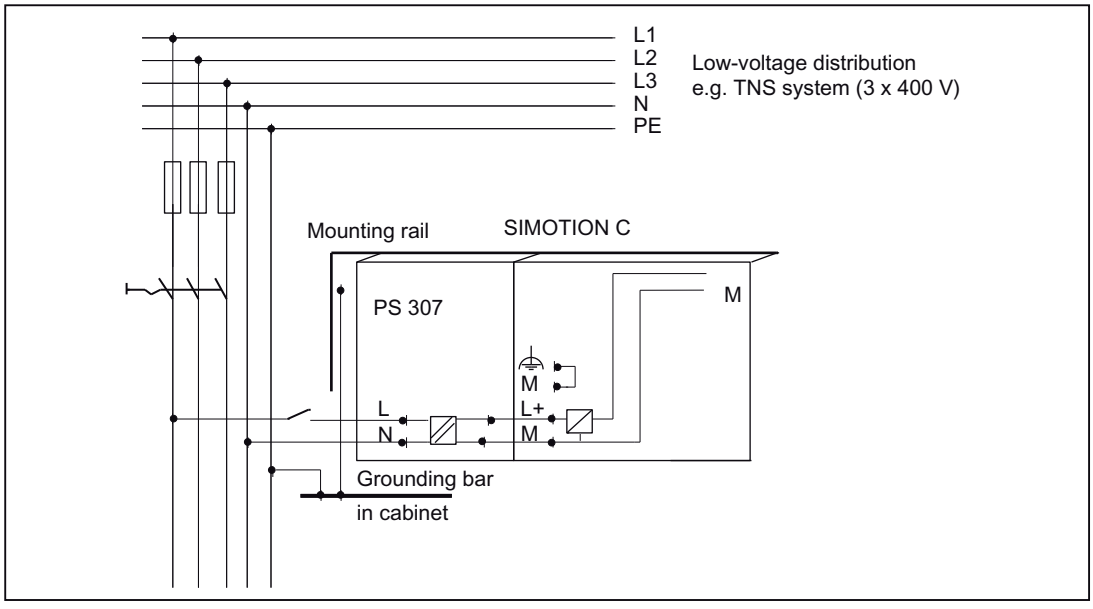


Figure 9-51 Module supply options

Supply system lines

Use flexible cables with a cross-section of 0.25 to 2.5 mm^2 (or AWG 18 to AWG 14) for wiring the power supply.

If you only use one wire per connection, a ferrule is not required.

You can use ferrules without an insulating collar in accordance with DIN 46228, Form A long version.

Connecting comb

You can also use a connecting comb to connect the PS 307 power supply module to the SIMOTION C. You will find the Article Nos. of the connecting comb required in chapter Spare parts and accessories (Page 6892).

Other 24 V connections

On the PS 307 power supply, there are 24 V connections available via the connecting comb for connecting the supply of the I/O modules.

Wiring with the connecting comb

Proceed as follows to wire the PS 307 power supply module and the SIMOTION C.

⚠ WARNING

You could come into contact with live wires if the power supply module and any additional load power supplies are switched on.

Ensure that the system is de-energized when you wire the system!

1. Open the front doors of the PS 307 and the SIMOTION C.
2. Release the clip for the cable strain relief on the PS 307.
3. Strip the power supply cable (230 V/120 V) (stripped length of 12 mm) and connect it to the PS 307.
4. Tighten the clip for the cable strain relief.

5. Insert the connecting comb and tighten it.
6. Close the front doors

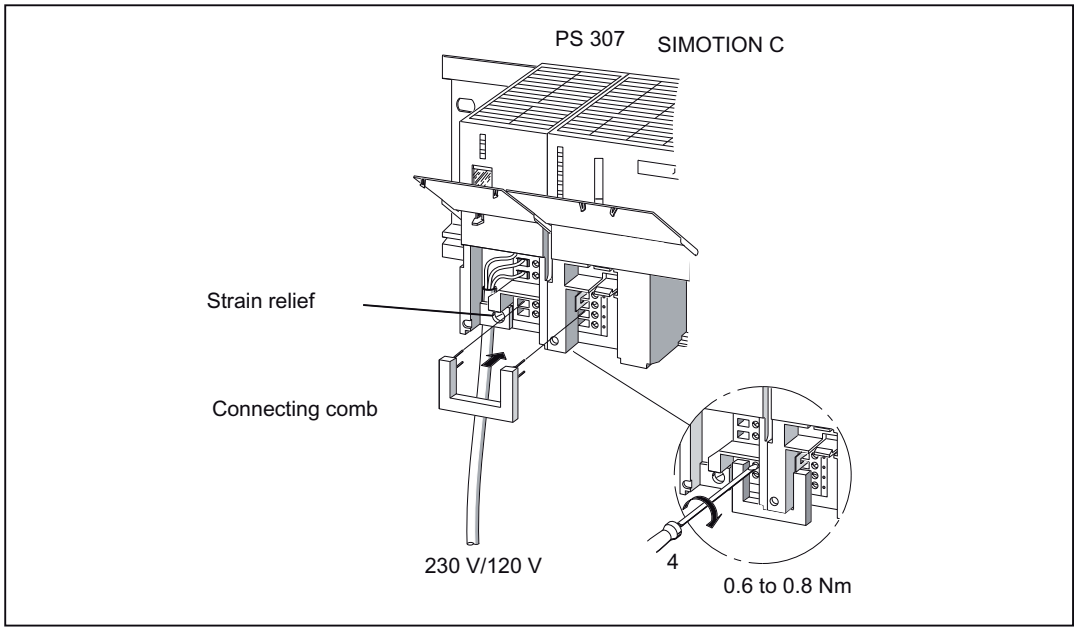


Figure 9-52 Connect power supply module and SIMOTION C with connecting comb

Setting power supply to the required supply voltage

Check whether the supply voltage selection switch is set correctly for your supply voltage. The basic setting on the PS 307 is always 230 V. Proceed as follows to change the setting for the supply voltage:

1. Remove the covering cap using a screwdriver.
2. Set the switch to the available supply voltage.
3. Refit the covering cap over the switch opening.

Reverse polarity protection

If the connection is correct and the power supply is switched on, the "5 VDC" LED is illuminated green.

Note

Your module will not work in the case of reverse polarity. However, a built-in reverse polarity protection will protect the electronics against damage.

Fuse

If a fault is present on the module, a built-in fuse protects the electronics against consequential damage (e.g. fire). In this case, the module must be replaced.

Connecting the drive units

Connecting the connecting cable

Note the following:

Note

Only use shielded, twisted pair cables; the shield must be connected to the metallic or metalized connector housing on the controller side. We recommend that you do not ground the shield on the drive side. This is to separate low-frequency interferences from the analog setpoint signal.

The pre-assembled cable available as an accessory provides the best possible immunity to interference.

Connection of drives (e.g. SIMODRIVE 611 universal with analog setpoint interface) to the onboard drive interface (C230-2, C240)

The figure below shows the connection of the C230-2, C240 to SIMODRIVE 611 universal drive units.

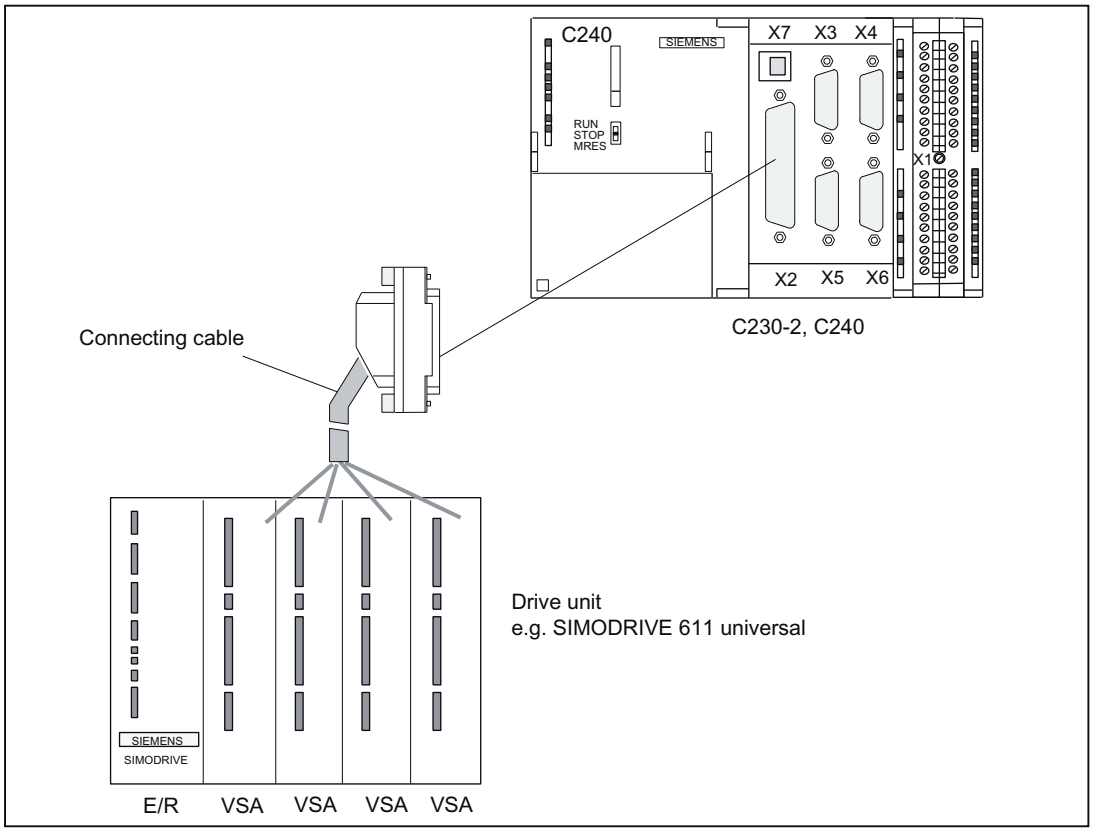


Figure 9-53 Connection of a SIMODRIVE 611 universal drive unit

Procedure:

1. Wire the free cable end of the connecting cable to the terminals on the drive unit. (The terminal markings on the cable ends indicate the corresponding terminals for SIMODRIVE devices.)
2. Open the front cover of the C230-2/C240 and connect the Sub-D socket (50-pin) to the X2 connector.
3. Lock the connector using the finger screws. Close the front cover.

Connecting cable

The connecting cable is a pre-assembled cable for four axes with an analog interface and terminal designation for SIMODRIVE drive units.

The connecting cable is available in a choice of lengths.

See *Catalog PM 21, NC 60, or ST 70*

Wiring diagram

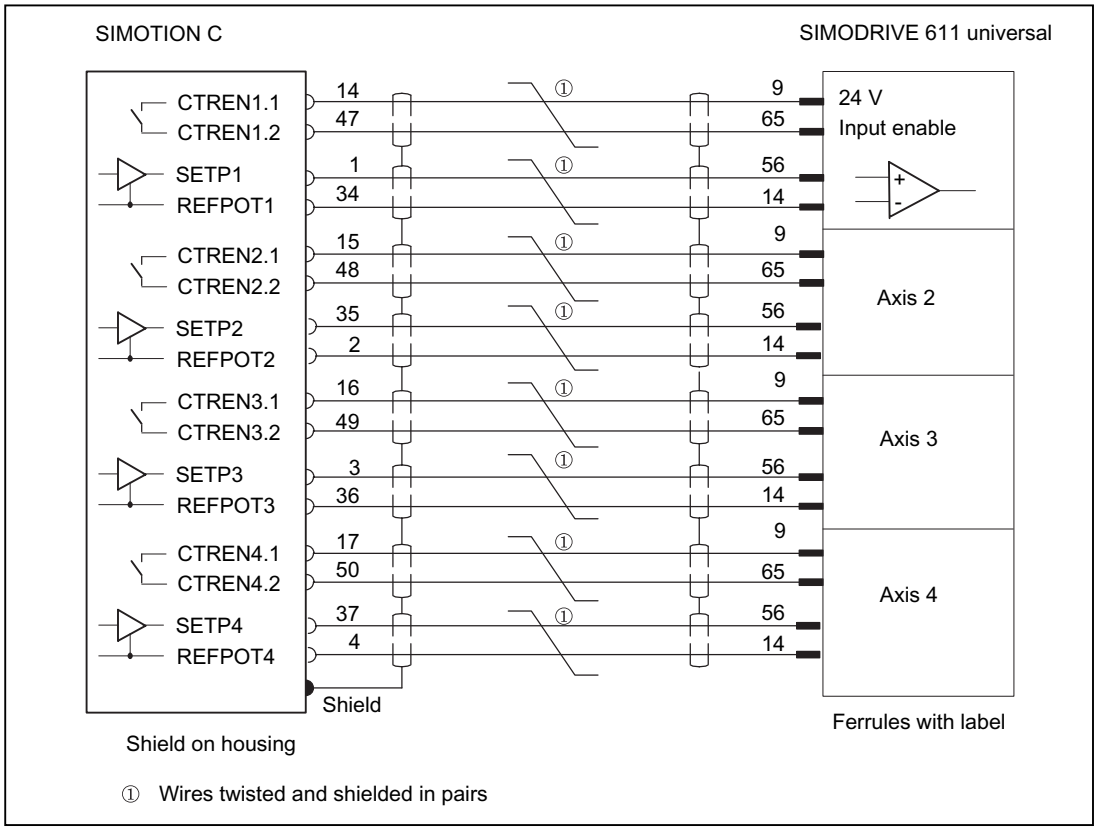


Figure 9-54 Terminal diagram for C230-2/C240 and SIMODRIVE 611 universal with analog setpoint interface

Setpoint assignment

The assignment of the setpoints for axes 1 to 4 is fixed.

Setpoint output signals (X2) for **drives with analog interface:**

- SW1, BS1, RF1.1, RF1.2 for axis 1
- SW2, BS2, RF2.1, RF2.2 for axis 2
- SW3, BS3, RF3.1, RF3.2 for axis 3
- SW4, BS4, RF4.1, RF4.2 for axis 4

Note

When using the drive interface for standard outputs (C240 only), proceed as described in chapter Onboard drive interface (C230-2, C240) (Page 6777). Wire the free ends of the connecting cable according to your application.

Connection of stepper drives (e.g. FM STEPDRIVE) to the onboard drive interface (C230-2, C240)

The figure below shows the connection of the C230-2, C240 to FM STEPDRIVE drive units.

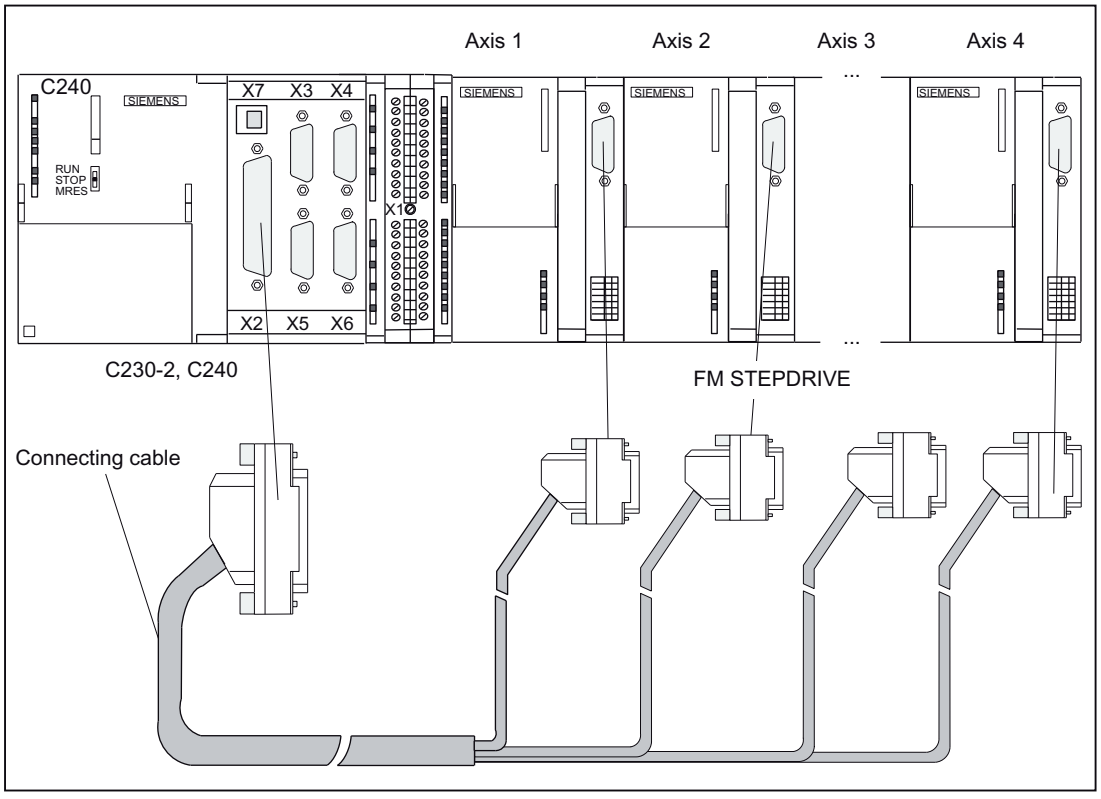


Figure 9-55 Connection of FM STEPDRIVE drive units

Procedure:

1. Insert the Sub-D socket (15-pin) in the FM STEPDRIVE module.
2. Open the front cover of the C230-2/C240 and connect the Sub-D socket (50-pin) to the X2 connector.
3. Lock the connector using the finger screws. Close the front cover.

Connecting cable

The connecting cable is a preassembled cable for four FM STEPDRIVE stepper motor drive units.

See *Catalog PM 21, NC 60.1, or ST 70*

Wiring diagram

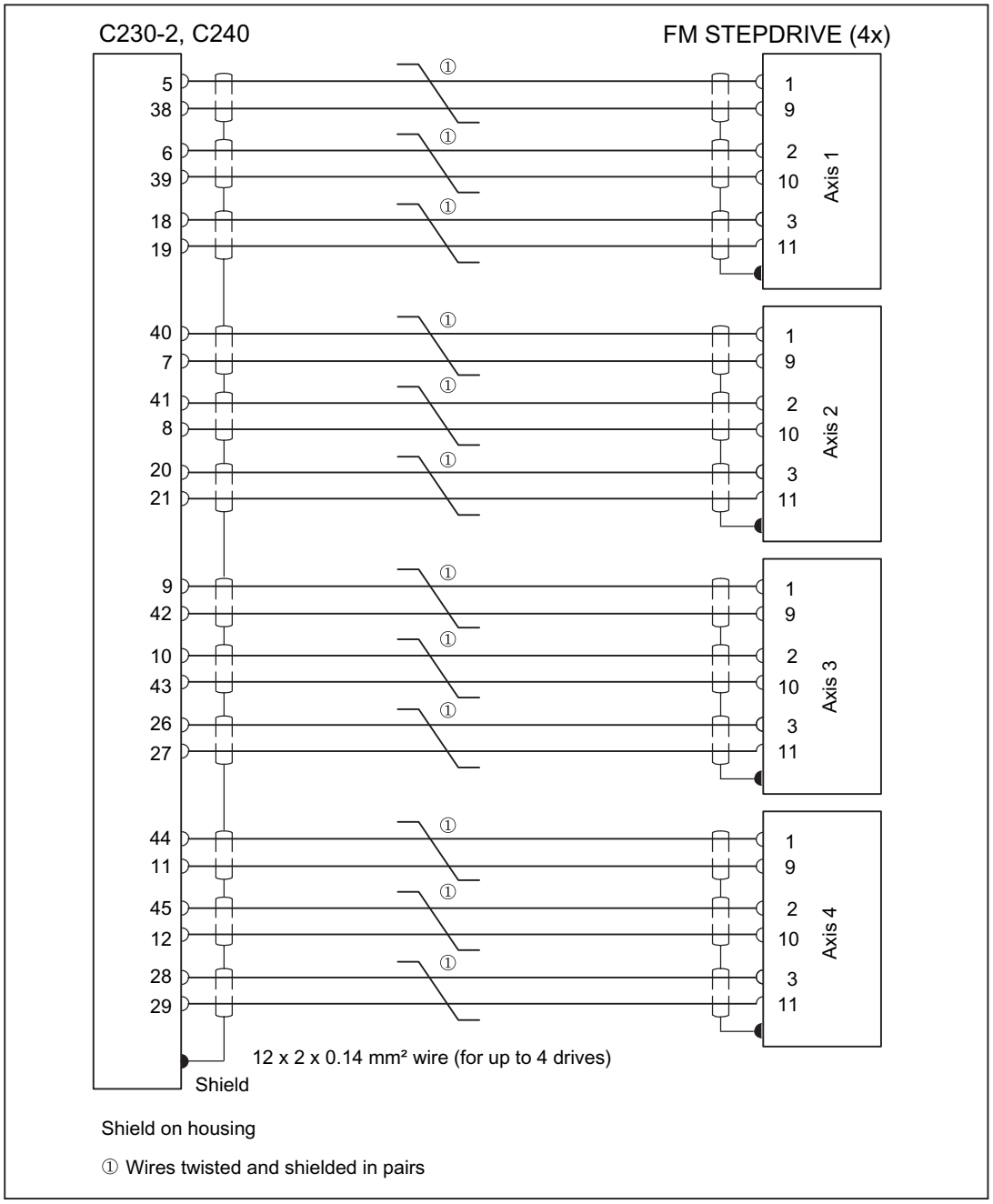


Figure 9-56 Wiring diagram for C230-2/C240 and FM STEPDRIVE

Setpoint assignment

The assignment of the setpoints for axes 1 to 4 is fixed.

Setpoint output signals (X2) for **stepper drive**:

- PULSE1, PULSE1_N, DIR1, DIR1_N, ENABLE1, ENABLE1_N for axis 1
- PULSE2, PULSE2_N, DIR2, DIR2_N, ENABLE2, ENABLE2_N for axis 2
- PULSE3, PULSE3_N, DIR3, DIR3_N, ENABLE3, ENABLE3_N for axis 3
- PULSE4, PULSE4_N, DIR4, DIR4_N, ENABLE4, ENABLE4_N for axis 4

Connection of drives (e.g. SIMODRIVE 611 universal) to the PROFIBUS DP

The figure below shows the connection of the SIMOTION C to a SIMODRIVE 611 universal drive unit.

Note that the "Motion Control with PROFIBUS DP" module must be fitted on the control unit of your drive unit.

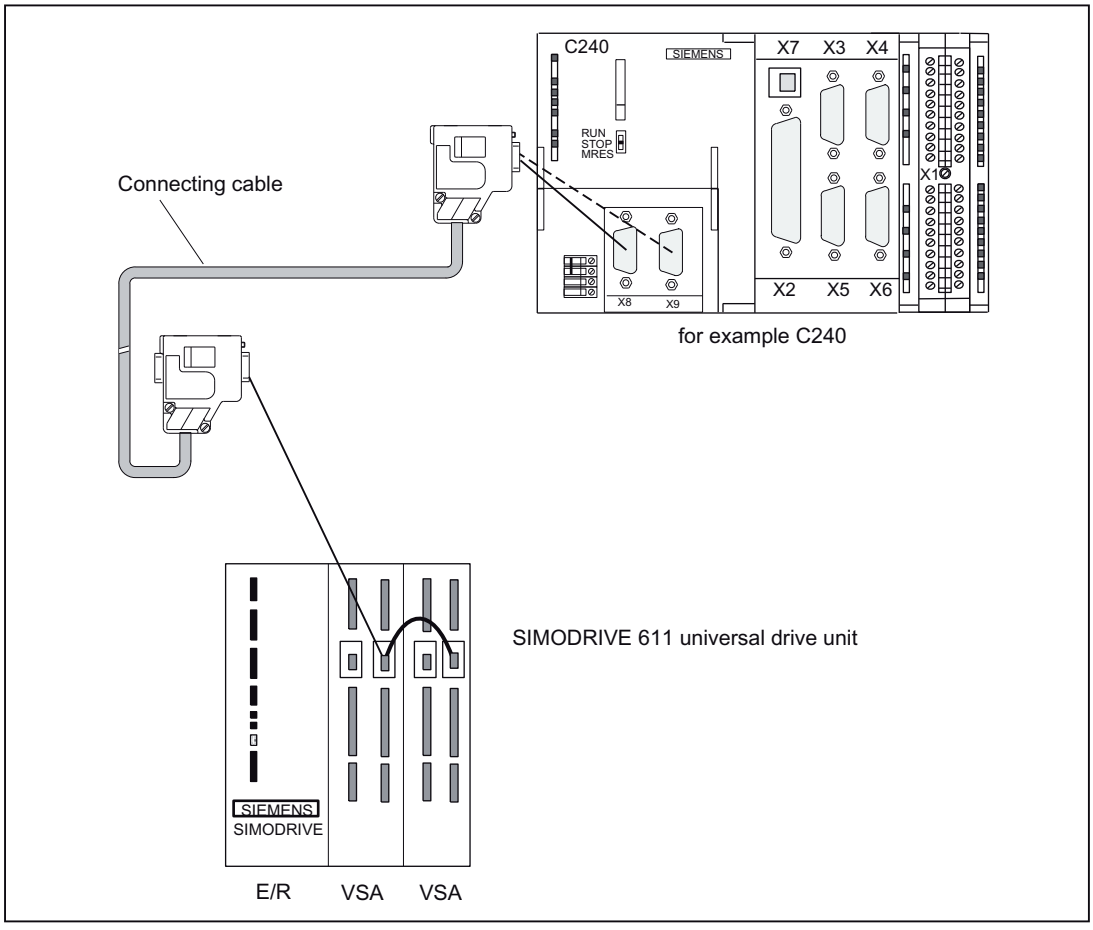


Figure 9-57 Connection of SIMODRIVE 611 universal drive unit to PROFIBUS DP

Procedure:

1. Insert the Sub-D connector (9-pin) into the drive unit.
2. Open the front door of the SIMOTION C and insert the Sub-D connector (9-pin) into the X8/X9 socket.
3. Lock the connector using the finger screws. Close the front cover.

Connecting cable

Details about the bus cable, bus connector and installation can be found in chapter Networking (Page 6842).

Note

The maximum cable length is 100 m.

The terminating resistor must be switched in (switch position "On") at the beginning and end of the bus segment. See figure "Bus connector (6ES7 ...): Terminating resistor connected and disconnected" in chapter Network components for a PROFIBUS subnet (Page 6845)

Connection of drives (e.g. SINAMICS S120) to PROFINET IO (C240 PN)

The figure below shows the connection of the C240 PN to a SINAMICS S120 drive unit.

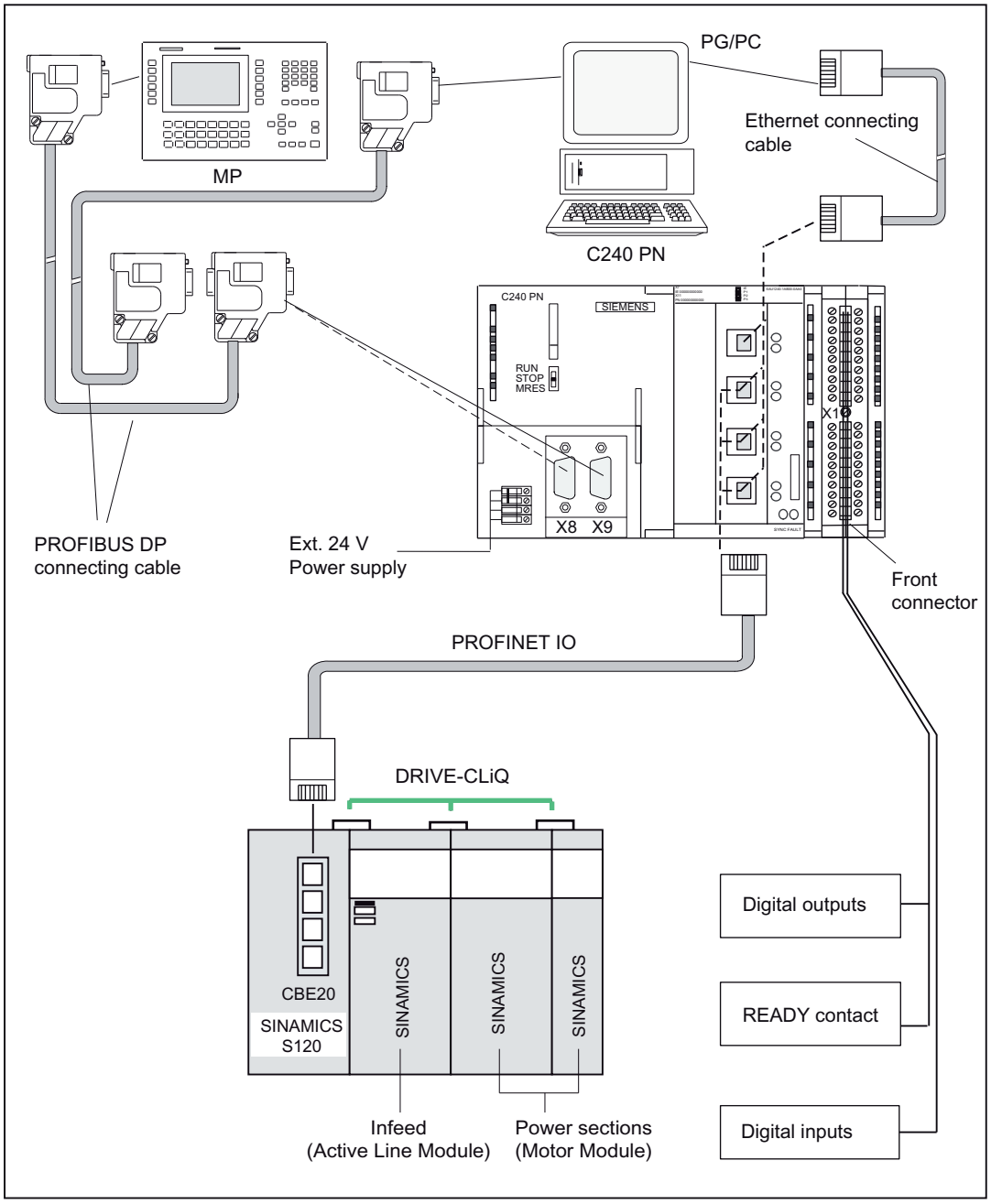


Figure 9-58 Connection of a SINAMICS S120 drive unit on the PROFINET IO

Procedure:

1. Insert the PROFINET connector into the drive unit.
2. Open the front door of the C240 PN and insert the PROFINET connector into the X11 P1, X11 P2 or X11 P3 socket.

Note

The strain relief provided for the PROFINET cable must be used in particular for vibrating machines. The cables should be secured close to the controller to achieve high vibration strength.

Figure 9-59 Strain relief

Connecting cable

Details about the bus cable, bus connector and installation can be found in chapter Networking (Page 6842).

Note

The maximum cable length is 100 m.

Mixed operation of analog drives and stepper drives (C230-2, C240)

Connecting cables for your configuration are available on request.

Follow the procedure outlined for connecting analog drives or stepper drives. The design conditions determine whether you install a terminal block or perform the wiring directly with pre-assembled cables.

Note

Ensure that the polarity assignment of the signals is correct. Refer to the technical documentation for your drive unit (e. g. *FM STEPDRIVE, Function Description* manual) and chapter Onboard drive interface (C230-2, C240) (Page 6777) of these operating instructions to ensure that the interconnection is correct.

Mixed operation of drives on the onboard drive interface and PROFIBUS DP

Drives with analog and digital connections and stepper drives can be operated together.

Proceed as described for the connection of drives to the onboard drive interface or PROFIBUS DP.

Mixed operation of drives on the onboard drive interface and use as standard output (C240 only)

Follow the procedure outlined for connecting analog drives.

Mixed operation of drives on the PROFIBUS DP and drives on the PROFINET IO (C240 PN)

Drives can be used in mixed operation.

Proceed as described for the connection of drives on the PROFIBUS DP or on the PROFINET IO.

See also

Overview (Page 6793)

Connecting the encoders (C230-2, C240)

Connecting the connecting cable

Note the following:

Note

Always use shielded data cables. The shielding must be connected with the metallic or metallized connector housing.

The pre-assembled cable available as an accessory provides the best possible immunity to interference and adequately dimensioned cross-sections for the power supply to the encoders.

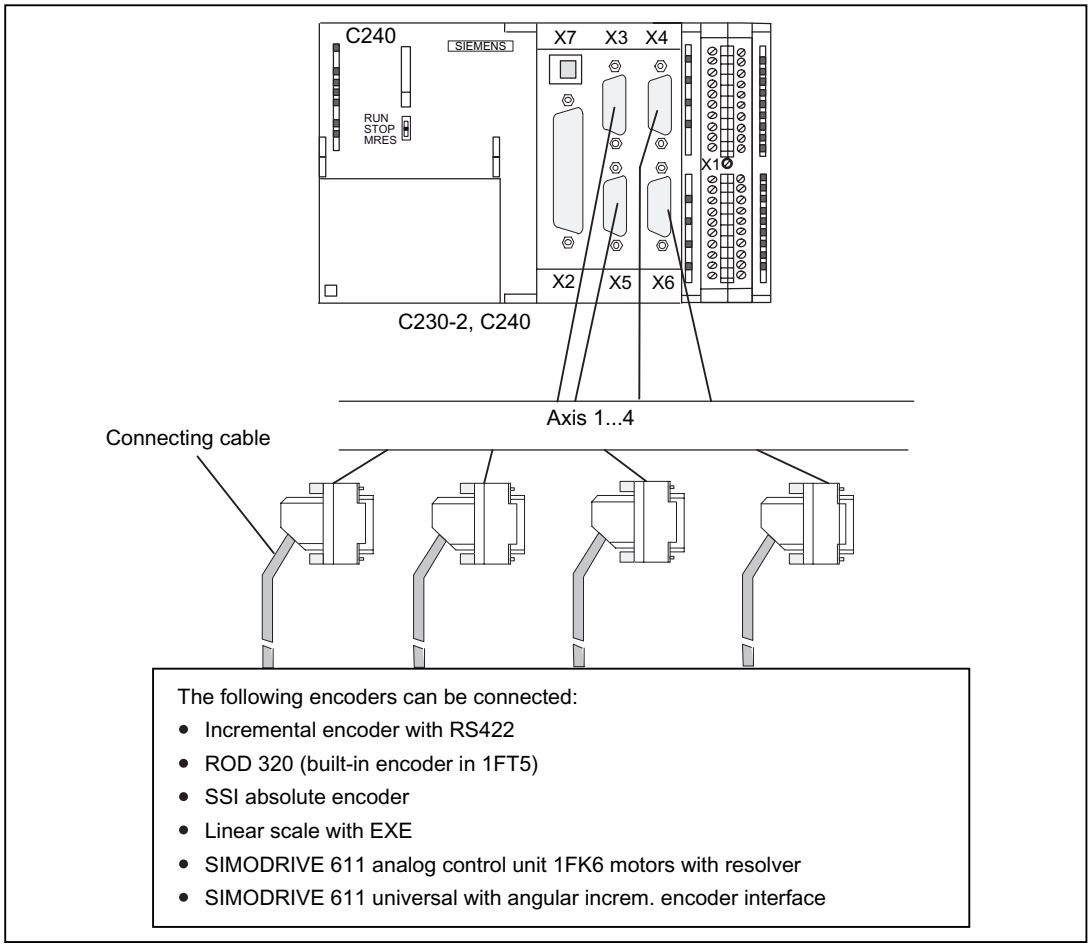


Figure 9-60 Encoder connections

Procedure for connecting encoders

Proceed as follows to connect the encoders:

1. Connect the cable to the encoders.
2. Open the front cover of the C230-2, C240 and insert the Sub-D connectors (15-pin) into sockets X3 to X6.
3. Lock the connector using the finger screws. Close the front cover.

Connecting cables available for encoders

The following connecting cables are available (see figure "Overview of connections"):

- Pre-assembled cable for external encoder or EXEs (for the connection of linear position encoders)
- Pre-assembled cable for built-in encoder with 17-pin round connector
- Pre-assembled cable for absolute encoders (SSI)

- Pre-assembled cable for SIMODRIVE 611 analog closed-loop control module 1FK6 motors with resolver
- Pre-assembled cable for SIMODRIVE 611 universal with incremental shaft encoder interface
  The incremental shaft encoder interface is used to emulate an incremental encoder. For this, the actual position value is measured by the encoder connected to the drive unit and transmitted to the C230-2, C240 as an incremental counting pulse.
  The encoder supply provided on the C230-2 and C240 is not used. Make sure there is a good ground connection between the controller and drive unit.

The connecting cables are available in a choice of lengths.

See *Catalog PM 21, NC 60, or ST 70*

Actual value assignment

Actual value assignment in the SIMOTION SCOUT axis wizard:

The assignment of actual values is only permanent for position axes with an encoderless stepper motor. The corresponding encoder channel is required for actual value simulation in the hardware.

Example:

For a position axis with an encoderless stepper motor on drive 2, encoder channel 2 is used within the hardware for actual value simulation.

For stepper motors with an encoder or for analog axes, the assignment of encoder channels is subject to no restrictions.

Connection options for encoders on the C230 or C240:

Encoders must be connected to actual value inputs X3 to X6.

The assignment between the drive and encoder channel is subject to no restrictions.

Connection example:

- The encoder for axis 1 is connected to actual value input X3
- The encoder for axis 2 is connected to actual value input X4
- The encoder for axis 3 is connected to actual value input X5
- The encoder for axis 4 is connected to actual value input X6

Note

The assignment between the drive and encoder channel is permanent for position axes with an encoderless stepper motor as a result of the actual value simulation required in the hardware.

Wiring the front connector

The following figure shows how the cables are routed to the front connector and how to suppress line interference through the use of the shield connecting element.

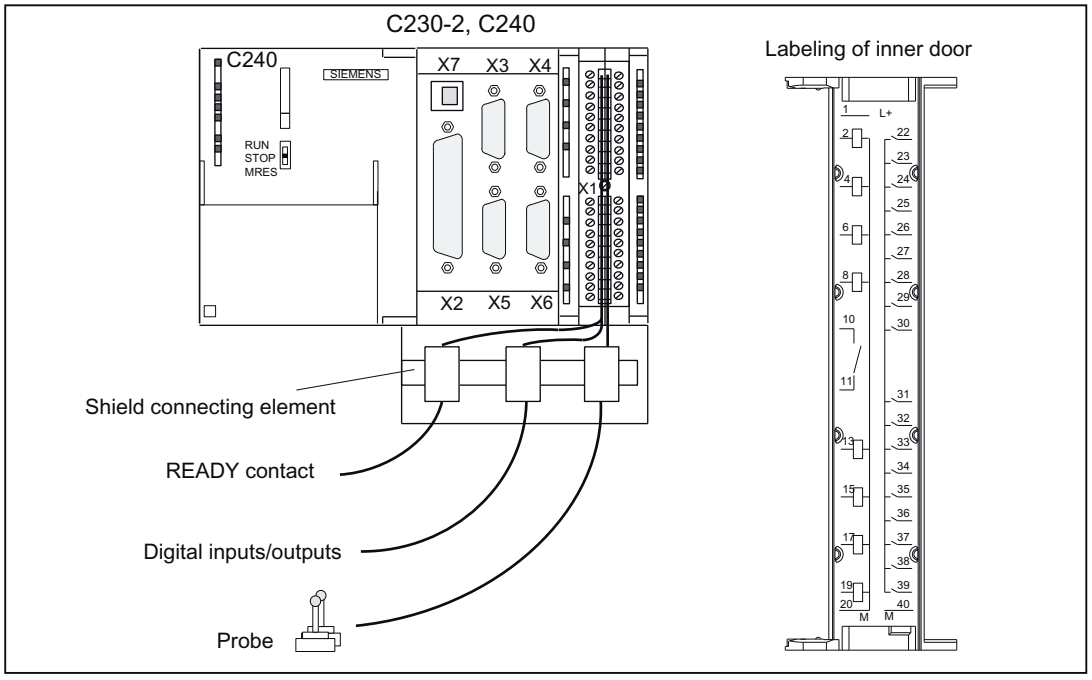


Figure 9-61 Wiring the front connector

Connecting cables

Flexible cable, cross-section 0.25 to 1.5 mm^2

Ferrules are not required.

You can use ferrules without an insulating collar in accordance with DIN 46228, Form A long version.

You can connect two cables of 0.25 to 0.75 mm^2 each in one ferrule.

Note

To achieve optimum interference suppression, a shielded cable must be used for connection of measuring inputs or external zero mark.

Required Tools

3.5-mm screwdriver or power screwdriver

Procedure for front connector wiring

Proceed as follows for the terminal strip:

1. Strip 6 mm of insulation off the cable. It may be necessary to fit a connector sleeve.
2. Open the front cover. Move the front connector into position for wiring.
   To do this, push the front connector into the module until it locks into position. In this position the front connector will still protrude from the module.
   The connector is locked in position, without electrical contact to the module.
3. If you route the wires out downwards, start the wiring at the bottom. If this is not the case, start at the top. Also screw in terminals that are not assigned.
   The tightening torque is 0.4 to 0.7 Nm.
4. Fit the cable strain relief provided around the wiring loom and the front connector.
5. Tighten the strain relief for the cable harness. Push in the strain relief to the left to increase cable space.
6. Tighten the mounting screw to move the front connector into operating position.
   Note: When the front connector is moved into operating position, a front connector keying engages in the front connector. The front connector will then fit only this module type.
7. Close the front cover.
8. You can fill out the labeling field provided and insert it in the front cover.

Shielded cables

If a shielded cable is used, the following additional actions are required:

1. Attach the cable shield to a grounded shielding bus immediately after the cable entry point in the cabinet (strip the insulation off the cable for this purpose).
   You can use the shield connecting element. This is mounted on the rail and can accommodate up to eight shielding terminals.
2. Continue routing the shielded cable as far as the module but do not make a connection to the shield there.

Shield connecting element

This element can be inserted in the mounting rail to provide screening for shielded cables. It can accommodate up to eight shielding terminals.

See chapter Connecting shielded cables via a shield connecting element (Page 6840).

Connection of measuring inputs or proximity sensors (external zero mark)

Procedure:

1. Wire the power supply for the encoders. This must meet the same criteria as the load power supply for the SIMOTION C.
2. Connect the shielded signal line to the encoders.

3. Remove a sufficient length of the cable sheath at the control end so that you can connect the shield to the shield connecting element and the free cable ends to the front connector.
4. Wire the signal line to the front connector.

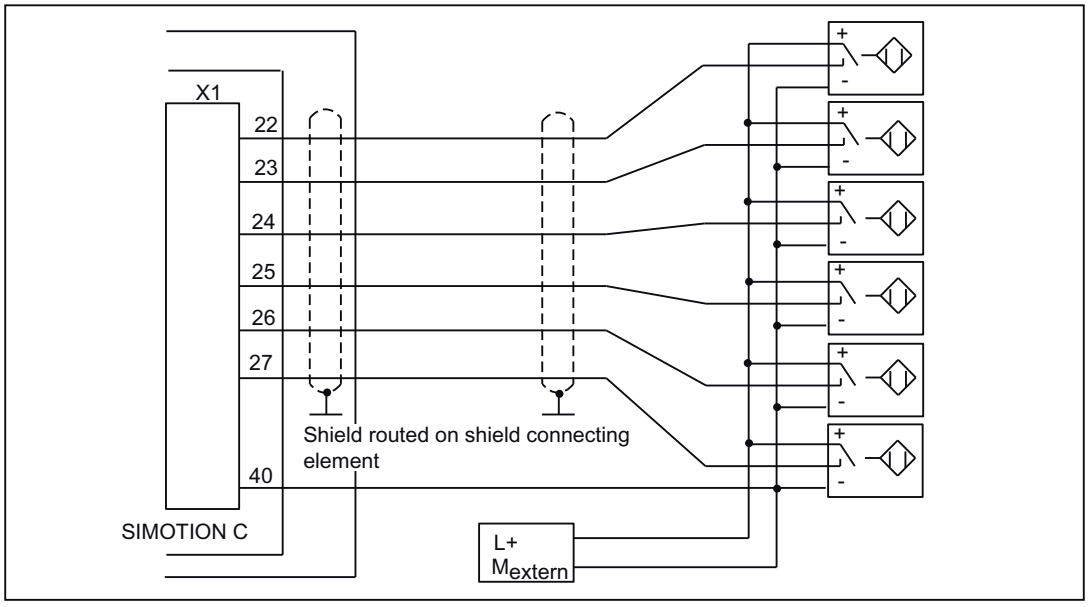


Figure 9-62 Overview of connections for measuring inputs or proximity encoders

Note

For the assignment of the front connector and the description of the I/O interface, refer to chapter I/O interface (Page 6797).

Connection of additional actuators/encoders

If you wish to connect additional actuators/encoders to the SMs on the I/O bus, proceed in the same way as for connecting digital inputs/digital outputs to the SIMATIC S7-300.

See the *S7-300, M7-300 Automation Systems, Module Data* Manual.

The digital inputs/digital outputs on the central I/O system are recorded/output at a refresh rate of approx. 1 ms.

Connecting shielded cables via a shield connecting element

Application

With the shield connecting element, you can easily connect all the shielded wires of the SIMOTION C or S7 modules to ground by directly connecting the shield connecting element with the mounting rail.

Design of the shield connecting element

The shield connecting element consists of:

- Retaining bracket with two screw bolts to mount the shield connecting element on the mounting rail (Article No.: 6ES5 390-5AA00-0AA0)
- The shielding terminals

You must use the following shielding terminal, depending on the cable cross-section used:

Table 9-40 Assignment of cable cross-sections and shielding terminals

Wire with shield diameter	Shielding terminal Article No. :
2 cables, each with 2 to 6 mm shield diameter	6ES7 390-5AB00-0AA0
1 cable with 3 to 8 mm shield diameter	6ES7 390-5BA00-0AA0
1 cable with 4 to 13 mm shield diameter	6ES7 390-5CA00-0AA0

The shield connecting element is 80 mm wide and provides space for two rows, each with four shielding terminals.

Fitting the shield connecting element

Fit the shield connecting element as follows:

1. Push the two threaded studs for the retaining bracket into the guide on the underside of the rail. Position the retaining bracket under the modules to be wired.
2. Screw the retaining bracket tight on the mounting rail.
3. The bottom of the shield connection terminal consists of a web interrupted by a slot. Place this part of the shielding terminal on edge a of the retaining bracket (see the following figure). Press the shielding terminals down and pivot them into the required position. You can fit a maximum of four shield terminals on each of the two rows.

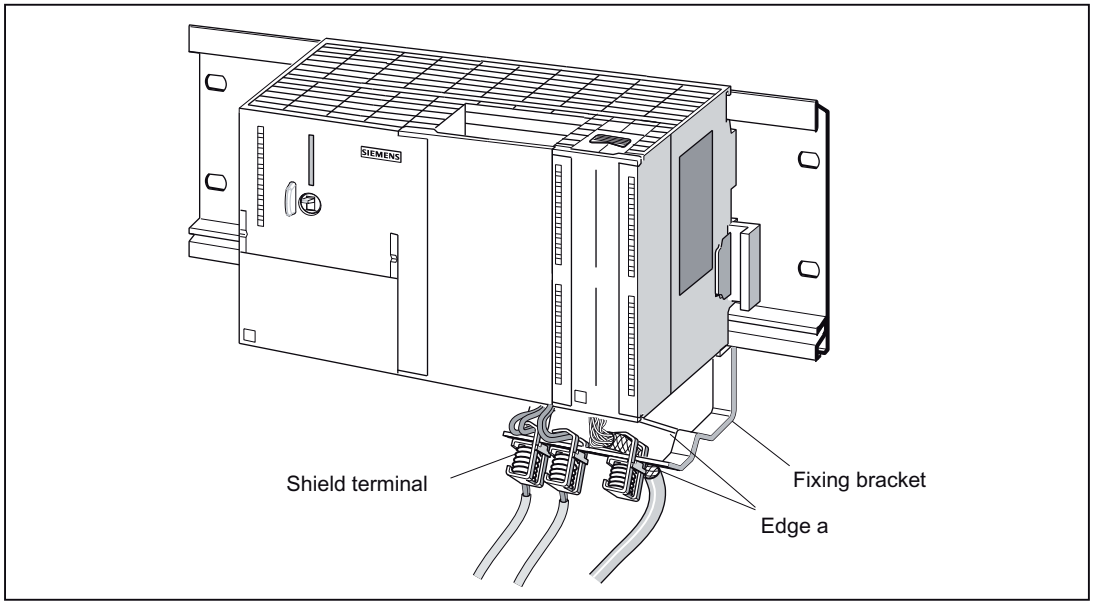


Figure 9-63 Fitting the shield connecting element

Laying the cables

Only one or two shielded cables can be attached per shielding terminal (see table "Assignment of cable cross-sections and shielding terminals"). Attach the cable to the stripped cable shield. The stripped section of cable shield must have a minimum length of 20 mm. If you require more than four shielding terminals, start the wiring on the back row of the shield connecting element.

Tip: Make sure you allow a sufficient length of cable between the shielding terminal and the front connector. This will, for example, ensure that if a repair is required, you can unplug the front connector without also having to remove the shield terminal.

9.1.6.2 Networking

Configuring

The *SIMOTION SCOUT* online help shows you how to design and configure a PROFIBUS or Ethernet subnet for your application.

Configuring a PROFIBUS subnet

device = node

Convention: All devices connected in a subnet will be referred to hereafter as nodes.

PROFIBUS addresses

In order for all nodes to communicate with each other, you must assign a 'PROFIBUS-address" to each node **before** connecting them:

You set the PROFIBUS addresses for each node individually using the programming device or PC (also by a switch on the slave for some PROFIBUS DP slaves).

The factory settings on the SIMOTION C are **address 2** and **baud rate 1.5 Mbit/s** for both PROFIBUS DP interfaces X8 and X9.

Tip: Mark the set address on the housing of all nodes in a subnet. You can then always see which address is assigned to which node in your plant.

The "highest PROFIBUS addresses" are preset for each PROFIBUS subnet. You can change these default settings.

Rules for PROFIBUS addresses

Before assigning PROFIBUS addresses, note the following rules:

- All PROFIBUS addresses in a subnet must be unique.
- The highest PROFIBUS address in a subnet must be ≥ the largest actual PROFIBUS node address in this subnet.

Recommendation for PROFIBUS addresses

Reserve PROFIBUS address "0" for a service programming device and "1" for a service SIMATIC HMI device, which will be connected to the subnet if required.

Recommendation for the PROFIBUS address of the SIMOTION C in case of replacement or service:

Reserve address "2" for a SIMOTION C. This prevents duplicate addresses when installing a SIMOTION C with factory setting on the subnet (e.g. when a SIMOTION C is replaced). You should therefore assign addresses greater than "2" to additional nodes in the subnet.

Segment

A segment is a bus cable between two terminating resistors. A segment with SIMOTION C as the master can contain up to 64 slaves. In addition, a segment is limited by the permitted cable length according to the baud rate (see chapter Onboard measuring system interface (C230-2, C240) (Page 6784)).

Rules for connecting nodes in a subnet

- Connect all nodes in a subnet "in series". In addition, integrate the programming devices and SIMATIC HMI devices for commissioning or servicing in the subnet in series.
- If you are operating more than 32 nodes in one subnet, you must use RS 485 repeaters to connect the bus segments (see also the description of the RS 485 repeater in the *S7-300 Automation Systems, M7-300, Module Data* Manual).
  In a PROFIBUS subnet, all bus segments combined must have at least one DP master and one DP slave.
- Use RS 485 repeaters to connect ungrounded bus segments and grounded bus segments.
- The maximum number of nodes per bus segment decreases with each RS 485 repeater. That is, if a bus segment contains one RS 485 repeater, the bus segment can contain no more than 31 additional nodes. However, the number of RS 485 repeaters does not affect the maximum number of nodes on the bus.
  Up to ten segments can be connected in series.
- Switch on the terminating resistor at the first and last node of a segment.

Components

Connect the individual nodes using bus connectors and the PROFIBUS cable, also refer to chapter Network components for a PROFIBUS subnet (Page 6845). Remember to provide a bus connector with a programming device socket at the ends of the subnet. This will give you the option of expanding the subnet if required (for example, for a programming device or SIMATIC HMI device).

Use RS 485 repeaters for the connection between segments and for extending the cable.

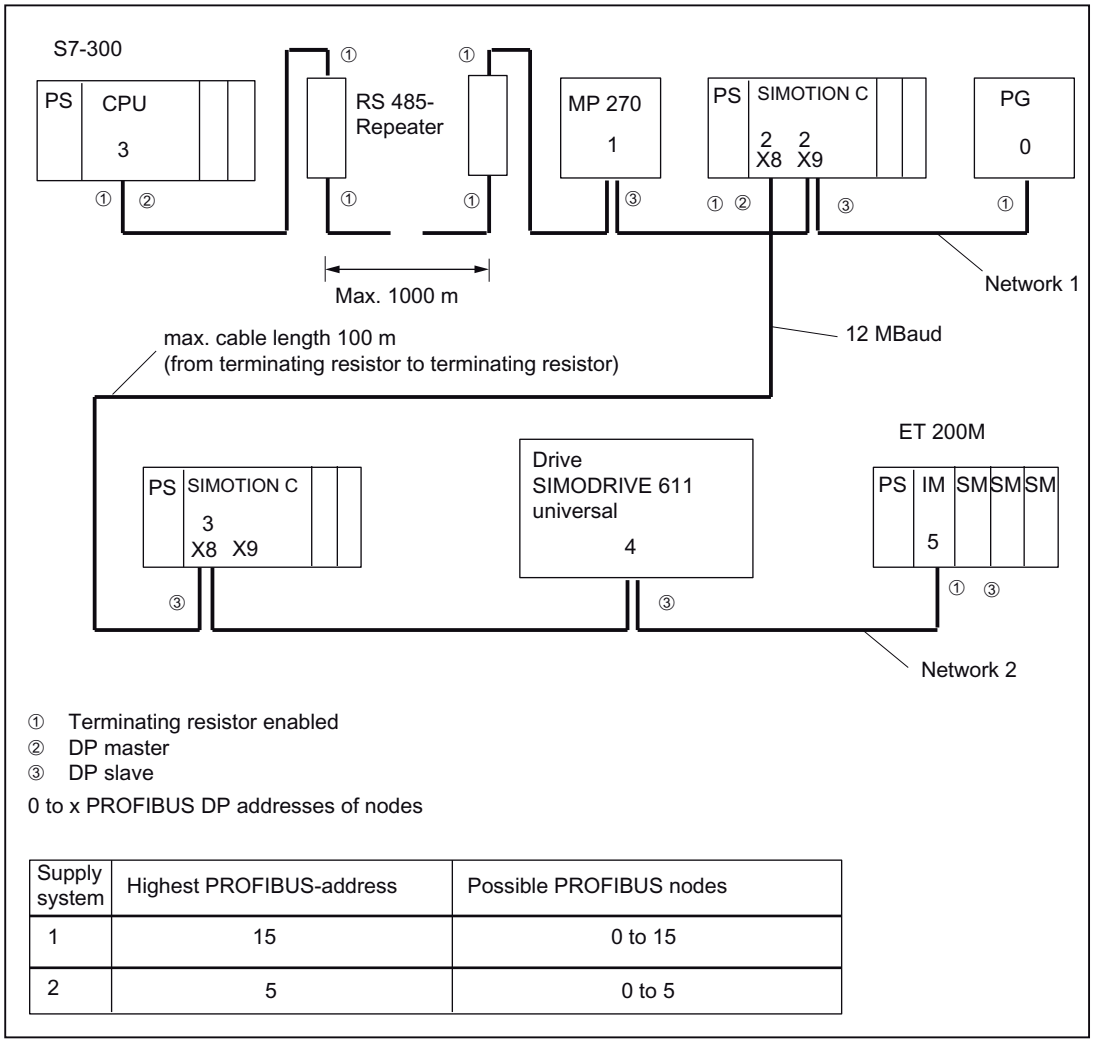
Terminating resistor

A cable must be terminated with its own surge impedance to prevent line disturbances caused by reflections. To this end, activate the terminating resistor at the first and last node of a subnet or segment (see figure "Bus connector (6ES7...): Terminating resistor connected and disconnected" in chapter Network components for a PROFIBUS subnet (Page 6845)

Make sure that the nodes to which the terminating resistor is connected are always supplied with voltage during power-up and operation.

Example

The following figure shows an example configuration of a subnet with SIMOTION C.



Supply system	Highest PROFIBUS-address	Possible PROFIBUS nodes
1	15	0 to 15
2	5	0 to 5

Figure 9-64 Networking example

Segment in subnet

The baud rate determines the cable length of a subnet segment (see the following table).

Table 9-41 Permitted cable lengths of a subnet segment for specific baud rates

Baud rate	Max. cable length of a segment (in m)
9.6 to 187.5 bits/s	1000[1)]
500 Kbits/s	400
1.5 Mbit/s	200
3 to 12 Mbits/s	100

[1)] With isolated interface

Greater cable lengths

You need to install RS485 repeaters for segments requiring cable lengths longer than the allowed length. The maximum possible cable length between two RS 485 repeaters corresponds to the cable length of a segment (see the previous table). You can connect up to nine RS 485 repeaters in series.

Note that an RS 485 repeater must be counted as a subnet node when determining the total number of nodes to be connected. This is true even if the RS 485 repeater is not assigned its own PROFIBUS address.

Network components for a PROFIBUS subnet

PROFIBUS cable

We can offer you e.g. the following PROFIBUS cables:

Table 9-42 PROFIBUS cable

Cables	Article No.
PROFIBUS cable	6XV1 830-0EH10
PROFIBUS direct-buried cable	6XV1 830-3FH10
PROFIBUS drum cable	6XV1 830-3EH10

Properties of the PROFIBUS cable

The PROFIBUS cable is a two-core, twisted, and shielded cable with the following features:

Table 9-43 Properties of the PROFIBUS cable

Features	Values
Characteristic impedance	Approx. 135 to 160 Ω (f = 3 to 20 MHz)
Loop resistance	≦ 115 Ω/km
Effective capacitance	30 nF/km
Attenuation	0.9 dB/100 m (f = 200 kHz)

Features	Values
Permissible conductor cross-section	0.3 mm^2 to 0.5 mm^2
Permitted cable diameter	8 mm ± 0.5 mm

Rules for cable installation

PROFIBUS cables must not be twisted, stretched, or compressed.

When installing the indoor bus cable, you must also keep within the following supplementary conditions (d_A = outer diameter of the cable):

Table 9-44 Supplementary conditions with indoor routing of bus cable

Features	Supplementary conditions
Bending radius for a single bend	≥ 80 mm (10 x d_A)
Bend radius for multiple bends	≥ 160 mm (20 x d_A)
Permissible temperature range for installation	- 40° C to + 60° C
Shelf and static operating temperature range	- 40° C to + 60° C

Bus connector

The bus connector is used to connect the PROFIBUS cable to the PROFIBUS DP interfaces (X8, X9), thus establishing a connection to additional nodes.

The following bus connectors are available:

- Up to 12 Mbits/s, cable outlet 90°
  - Without PG socket (6ES7 972-0BA12-0XA0 or 6ES7 972-0BA50-0XA0)
  - With PG socket (6ES7 972-0BB12-0XA0 or 6ES7 972-0BB50-0XA0)
- Up to 12 Mbits/s, canted cable turn
  - Without PG socket (6ES7 972-0BA41-0XA0)
  - With PG socket (6ES7 972-0BB41-0XA0)

Inserting a bus connector in a module

Proceed as follows to connect the bus connector:

1. Plug the bus connector into the module.
2. Screw the bus connector tightly onto the module.
3. If the bus connector is located at the start or the end of a segment, you must connect the terminating resistor ("ON" switch setting) (refer to the following figure).

Make sure that the nodes at which the terminating resistor is located are always supplied with voltage during startup and operation.

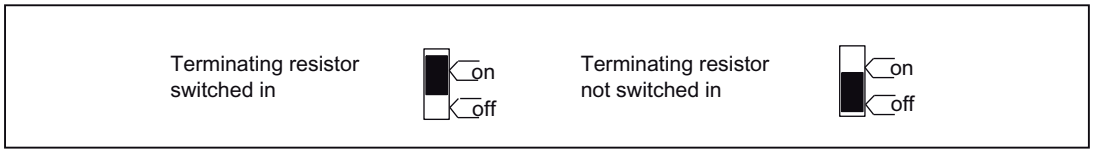


Figure 9-65 Bus connector (6ES7 ...): terminating resistor switched on and off

Unplugging a bus connector

You can unplug a bus connector with a looped-through bus cable at any time from the PROFIBUS DP interface without interrupting data exchange on the bus.

⚠ WARNING

Data traffic error might occur on the bus!

A bus segment must always be terminated at both ends with the terminating resistor. This is not the case if the last bus connector node is de-energized, for example. Because the bus connector takes its voltage from the station, this terminating resistor is ineffective.

Please make sure that power is always supplied to stations on which the terminating resistor is active.

Configuring an Ethernet subnet on the Ethernet interface

Overview

You can connect an Industrial Ethernet to the 8-pin RJ45 **X7** socket, see Figure "Overview of the cable connecting the SIMOTION C to the servo drive (digital connection) - example" in chapter Overview of wiring diagram (Page 6818).

Industrial Ethernet is a communication network with a transmission rate of 10/100 Mbits/s.

Communication between SIMOTION C, the PG/PC (e.g. STEP 7, SIMOTION SCOUT, SIMATIC NET OPC), and the I/O devices provided for this purpose is supported by the Ethernet.

A shielded twisted pair cable is used for the networking in this case. For additional information, refer to the *SIMATIC NET, Industrial Twisted Pair and Fiber Optic Networks* Manual. This document is supplied with SIMOTION SCOUT in electronic form.

The following connecting cables are recommended:

- SIMATIC NET, Ind. Ethernet TP XP CORD RJ45/RJ45, TP cable assembled with 2 RJ45 plugs, send and receive cables crossed
  Article No.: 6XV1870-3R□□□ (□□□ - length code)
- SIMATIC NET, Ind. Ethernet TP CORD RJ45/RJ45, TP cable assembled with 2 RJ45 plugs
  Article No.: 6XV1870-3Q□□□ (□□□ - length code)

You can obtain additional information about the different cable systems for Ethernet from your SIEMENS contact.

Note

A crossover cable should be used for the direct connection of the controller to a PG/PC.

Loading the Ethernet configuration via PROFIBUS DP (loading of the IP address)

For configuration using Industrial Ethernet, the SIMOTION C must be provided with an IP address, the subnet mask and the router address.

To configure the Ethernet addresses and transfer them to the SIMOTION C, proceed as follows:

1. Open your project.
2. Open HW Config. Double-click the SIMOTION C module to open the "Properties - C2xx" dialog box.
3. On the "General" tab, click the "Properties" button of the Ethernet interface. The "Properties - C2xx Ethernet Interface" dialog is displayed.
4. In this dialog, click "New". The "New Industrial Ethernet" subnet dialog is displayed. In this dialog box, you can change the name of the new subnet or confirm the factory setting with "OK".
5. The newly created Ethernet subnet is now shown under Subnet in the "Properties - C2xx Ethernet Interface" dialog and must be selected.
6. In this dialog box, enter the required addresses for IP address and subnet dialog box. Under Router, choose whether a router is to be used. If using a router, enter the router address.
7. Confirm this dialog box with "OK".
8. Close the "Properties - C2xx" dialog with "OK".
9. Save and compile the modified hardware configuration.
10. Load the new hardware configuration to the SIMOTION C via PROFIBUS DP.

Loading the Ethernet configuration via Ethernet (loading of the IP address)

If a PROFIBUS DP is not available for the initial loading of the IP address, the following procedure starting from Windows 2000 can be used: The "Automatically Assign IP Address" setting must be enabled in the TCP/IP configuration of the PC.

1. Connecting a Windows PC and C2xx directly via RJ45 crossover cable.
2. Boot the Windows PC. The PC does not find a DHCP server and automatically selects an IP address from the APIPA subnet (Automatic Private IP Addressing) 169.254.0.0.
3. Load the new hardware configuration with the new IP address via Ethernet to IP address 169.254.11.22 (default IP address for the C2xx upon delivery).

Note

You can also write the user project and/or configuration with SIMOTION SCOUT and the menu command "Load to file system" directly from the PC to the memory card (micro memory card). To do this, select the device in the project navigator and execute the "Load to file system" command in the context menu.

Configuring an Ethernet subnet on the PROFINET interface (C240 PN)

Overview

You can connect a PROFINET node to the 8-pin RJ45 X11 P1, X11 P2 and X11 P3 sockets, see Figure "Overview of the cable connecting the C240 PN to the servo drive (digital connection) - example" in chapter Overview of wiring diagram (Page 6818).

PROFINET is a communication network with a transmission rate of 100 Mbit/s.

You can use a PG/PC to communicate with STEP 7, SIMOTION SCOUT, and SIMATIC NET OPC.

A shielded twisted pair cable is used for the networking in this case. For additional information, refer to the *SIMATIC NET, Industrial Twisted Pair and Fiber Optic Networks* Manual. This document is supplied with SIMOTION SCOUT in electronic form.

All ports of the PROFINET interface support auto-MDI(X) in contrast to the Ethernet interface. A crossover cable is therefore not required for a direct connection to the PG/PC.

The following connecting cables are recommended:

- SIMATIC NET, Ind. Ethernet FC TP standard cable GP
  Article No.: 6XV1840-2AH10 (sold by the meter)
- SIMATIC NET, Ind. Ethernet FC TP trailing cable GP
  Article No.: 6XV1870-2D (sold by the meter)
- SIMATIC NET, Ind. Ethernet TP XP CORD RJ45/RJ45, TP cable assembled with two RJ45 plugs, send and receive cables crossed
  Article No.: 6XV1870-3R□□□ (□□□ - length code)
- SIMATIC NET, Ind. Ethernet TP CORD RJ45/RJ45, TP cable assembled with two RJ45 plugs
  Article No.: 6XV1870-3Q□□□ (□□□ - length code)

The following bus connectors can be used with the C240 PN:

- SIMATIC NET, Ind. Ethernet FC RJ45 plug 145
  Article No.: 6GK1901-1BB30-0AA0 (1 item)
  Article No.: 6GK1901-1BB30-0AB0 (10 items)

You can obtain additional information about the different cable systems for PROFINET and Ethernet from your SIEMENS contact.

Loading the PROFINET configuration via PROFIBUS DP (loading of the IP address)

For configuration using PROFINET, the SIMOTION C must be provided with an IP address, the subnet mask and the router address.

To configure and transfer PROFINET addresses to the C240 PN, proceed as follows:

1. Open your project.
2. Open HW Config. Double-click the X11 "PNxIO" slot of the C240 PN module to open the "Properties - PNxIO" dialog box.
3. Click the "Properties" button of the Ethernet interface. The "Properties - PNxIO Ethernet Interface" dialog box is displayed.
4. In this dialog, click "New". The "New Industrial Ethernet" subnet dialog is displayed. In this dialog box, you can change the name of the new subnet or confirm the factory setting with "OK".
5. The newly created Ethernet subnet is now shown under Subnet in the "Properties - Ethernet Interface" dialog box and must be selected.
6. In this dialog box, enter the required addresses for IP address and subnet dialog box. Under Router, choose whether a router is to be used. If using a router, enter the router address.
7. Confirm this dialog box with "OK".
8. Close the "Properties - PNxIO" dialog box with "OK".
9. Save and compile the modified hardware configuration.
10. Load the new hardware configuration to the C240 PN via PROFIBUS DP.

Loading the PROFINET configuration via Ethernet (loading of the IP address)

If a PROFIBUS DP is not available for the initial loading of the IP address, the following procedure starting from Windows 2000 can be used: The "Automatically Assign IP Address" setting must be enabled in the TCP/IP configuration of the PC.

1. Connecting a Windows PC and C240 PN directly via RJ45 crossover cable
2. Boot the Windows PC. The PC does not find a DHCP server and automatically selects an IP address from the APIPA subnet (Automatic Private IP Addressing) 169.254.0.0.
3. Load the new hardware configuration with the new IP address via Ethernet to IP address 169.254.11.22 (default IP address of the SIMOTION C as delivered)

Note

You can also write the user project and/or configuration with SIMOTION SCOUT and the menu command "Load to file system" directly from the PC to the memory card (micro memory card). To do this, select the device in the project navigator and execute the "Load to file system" command in the context menu.

Factory setting

After resetting the SIMOTION C to the factory setting (see chapter Setting SIMOTION C to factory settings (Page 6872)) or when the module is delivered, the following addresses are set:

- Baud rate = 1.5 Mbit/s
- PROFIBUS address for the interfaces X8 = 2 and X9 = 2
- Ethernet interface:
  IP address = 169.254.11.22
  Subnet mask = 255.255.0.0
  Router address = do not use a router
- PROFINET interface X11 (C240 PN)
  IP address = no valid address
  Subnet mask = no valid address
  Router: Do not use a router

MPI subnet

An MPI subnet has the same basic configuration as a PROFIBUS subnet. The installation rules, stated in the chapter Configuring a PROFIBUS subnet (Page 6842), therefore apply.

9.1.7 Addressing

9.1.7.1 Slot-oriented address allocation for modules (default addresses for centralized I/O)

Introduction

In the case of addressing based on the slot (default addressing), a module start address is assigned to every slot number. Depending on the type of module, these are different addresses for digital, analog, FM and CP modules (see table below). In this section we will show you which module start address is assigned to which slot number. You need this information to determine the module start addresses of the modules used.

Maximum configuration

The following figure shows a configuration of a rack and the possible slots. A 2-tier layout with IM 365 is possible with SIMOTION C.

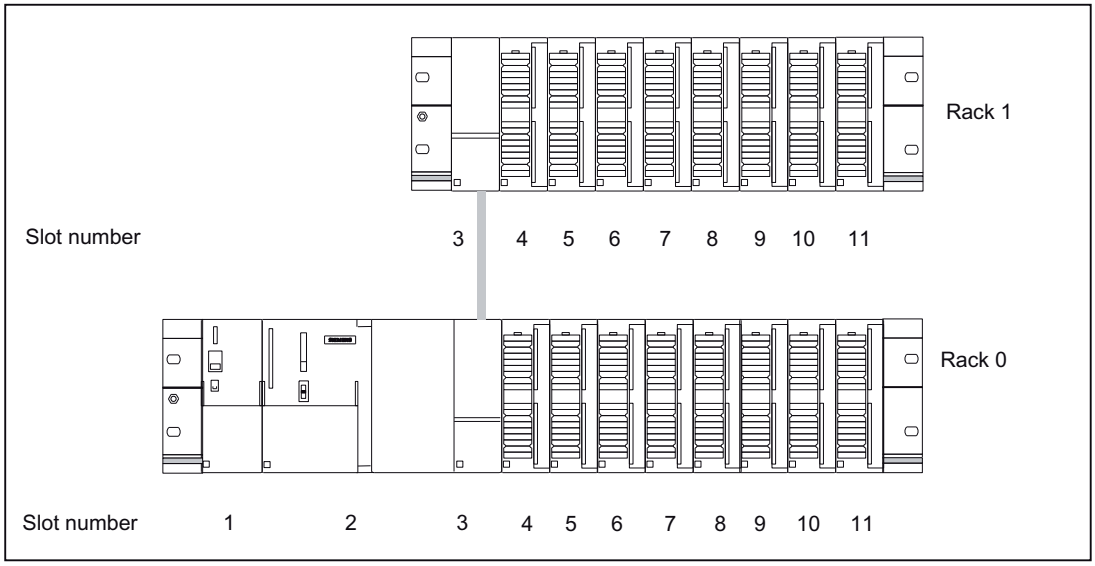


Figure 9-66 Slots for modules on rack (centralized I/O)

Module start addresses

The following table shows the assignment of module start addresses to slot numbers and racks.

In the case of input/output modules, the input addresses and output addresses start from the same module start address.

Table 9-45 Module start addresses of signal modules

Module carrier	Module start addresses	Slot number										
		1	2	3	4	5	6	7	8	9	10	11

0	Digital Analog [1]	PS	SIMOTION C	IM 365 S	0 256	4 272	8 288	12 304	16 320	20 336	24 352	28 368
1	Digital Analog [1]	-	-	IM 365 R	32 384	36 400	40 416	44 432	48 448	52 464	56 480	60 496

[1] The FM and CP modules are assigned to the analog address range.

Note

Do not insert any non-configured modules centrally. Modules that are installed but not configured are repeatedly addressed via the I/O bus. This requires additional computing time.

9.1.7.2 User-assignable addressing on the SIMOTION C (centralized and distributed I/O)

User-assignable addressing

User-assignable addressing means you can assign an address of your choice to each module or slot, for example, integrated inputs/outputs, drives. You make this assignment in the **hardware configuration** (see *SIMOTION SCOUT* online help). Here you specify the module start address on which all further module addresses will then be based.

advantages

Advantages of user-definable addressing:

- You can make the best possible use of the available address spaces because there are no "address gaps" between the modules.
- When creating standard software, you can specify addresses which are independent of the respective configuration of the SIMOTION modules.

9.1.7.3 Addressing signal modules

Introduction

The following section describes signal module addressing in the default setting. You need the information so that you can address the signal module channels in the user program.

Addresses of digital modules

The address of an input or output of a digital module is made up of the byte address and the bit address.

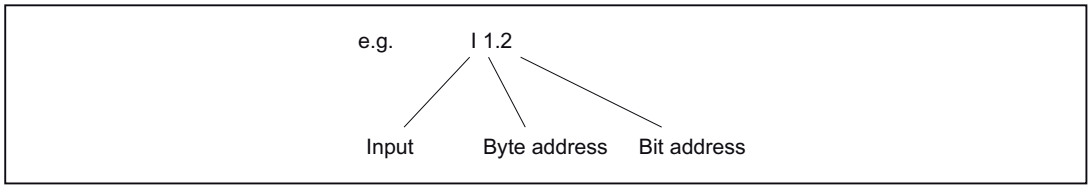


Figure 9-67 Address of an input of a digital module - example

The byte address is governed by the module start address.

You can note the bit address on the module.

The figure below shows you the scheme by which the addresses of the individual channels of the digital module are obtained.

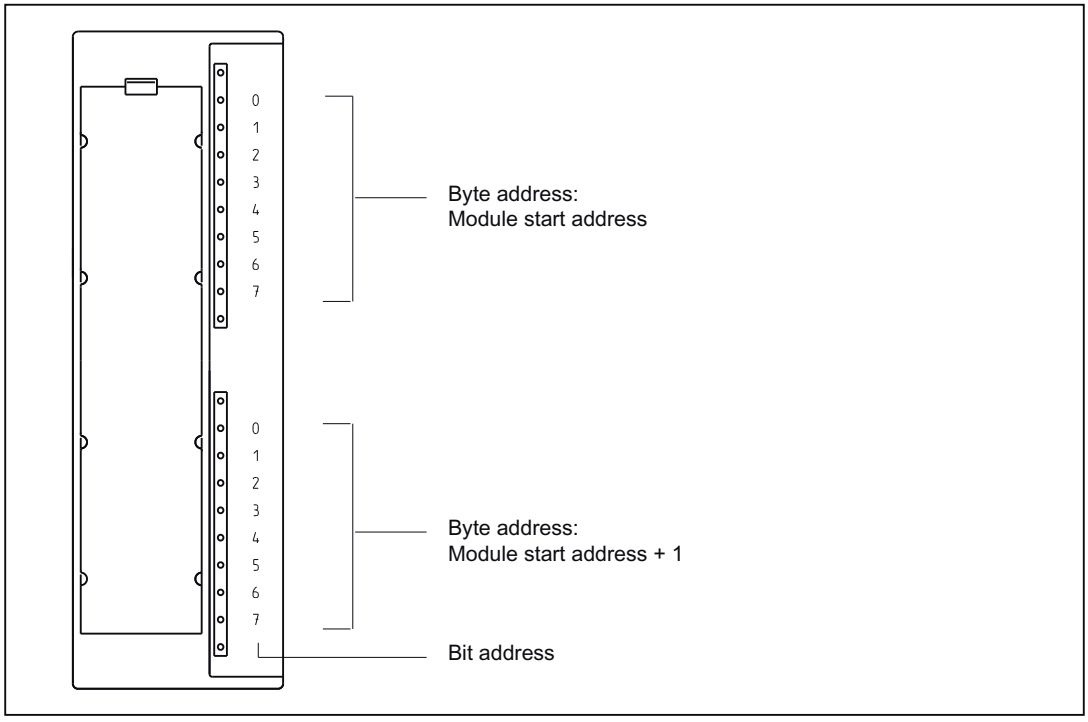


Figure 9-68 Addresses for the inputs and outputs of digital modules

Example of digital modules

The example in the figure below shows which default addresses are derived when a digital module is located in slot 4 (that is, when the module start address is 0).

Slot number 3 is reserved as no interconnection module is present in the example.

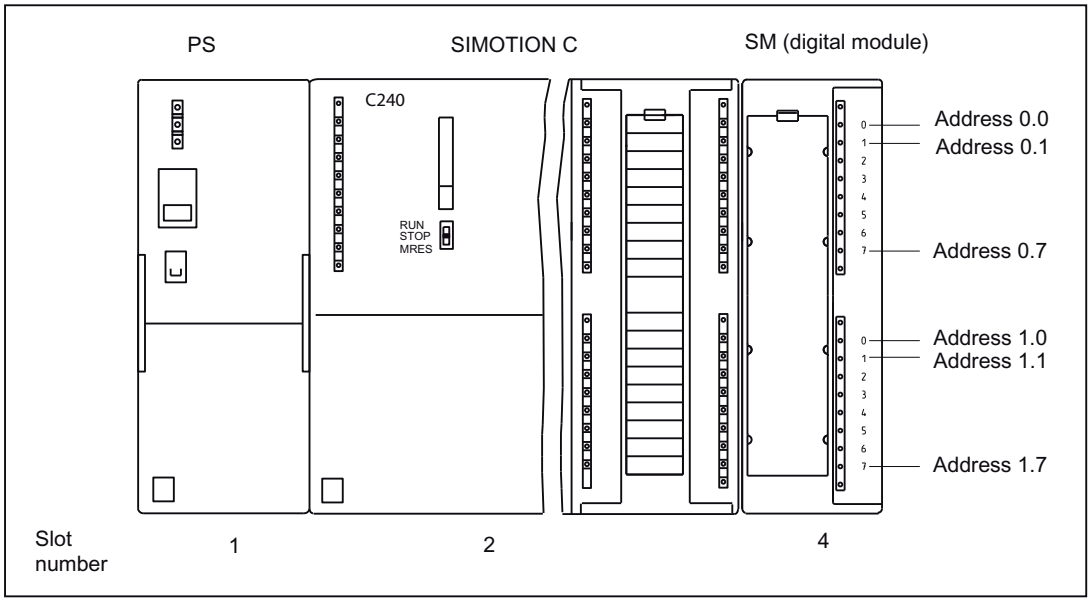


Figure 9-69 Addresses for the inputs and outputs of the digital module in slot 4

Addresses of analog modules

The address of an analog input or analog output channel is always an even address.

The channel address is based on the module start address.

If the first analog module is in slot 4, then it has the default start address 256. The start address of each additional analog module is raised per slot by 16 (see table "Slots for modules on rack (centralized I/O)").

An analog input/analog output module has the same start addresses for the analog input and analog output channels.

Example for analog modules

The example in the figure below shows you which default channel addresses are obtained for an analog module located at slot 4. You will see that in the case of an analog input/analog output module, the analog input and analog output channels are addressed from the same address, the module start address.

Slot number 3 is reserved as no interconnection module is present in the example.

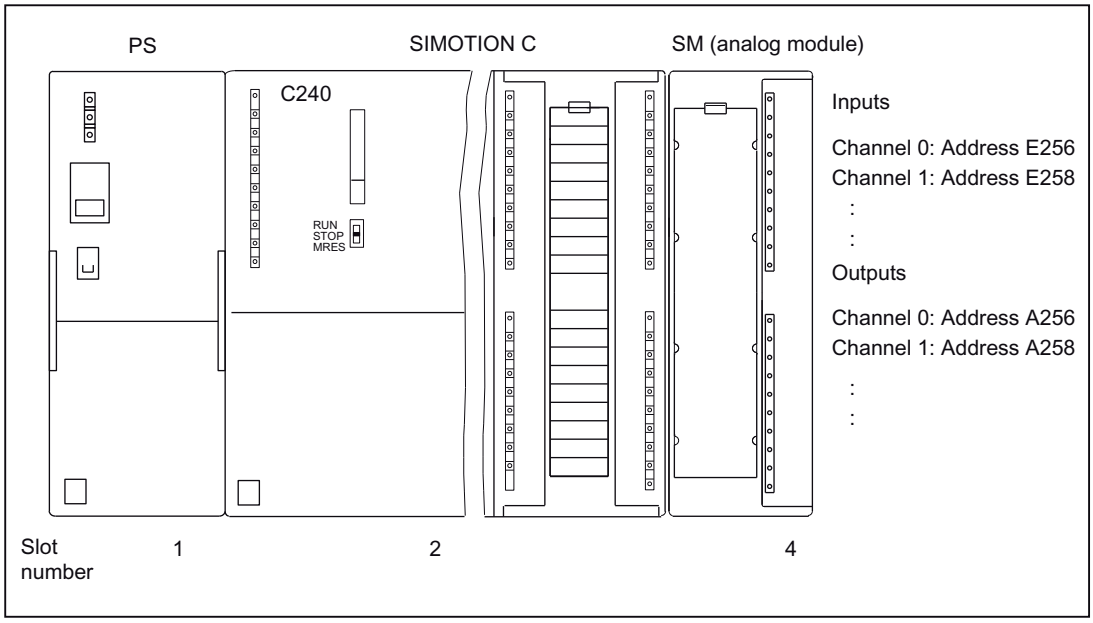


Figure 9-70 Addresses for the inputs and outputs of the analog module in slot 4

Addresses of the FM and CP modules

The FM and CP modules are assigned to the analog address range. In addition, the FM and CP modules have extended interfaces (data sets). For a detailed description, see the corresponding module's manual.

9.1.7.4 Addressing the onboard digital inputs and outputs of the SIMOTION C

The following figure shows the default start addresses of the onboard digital inputs/outputs.

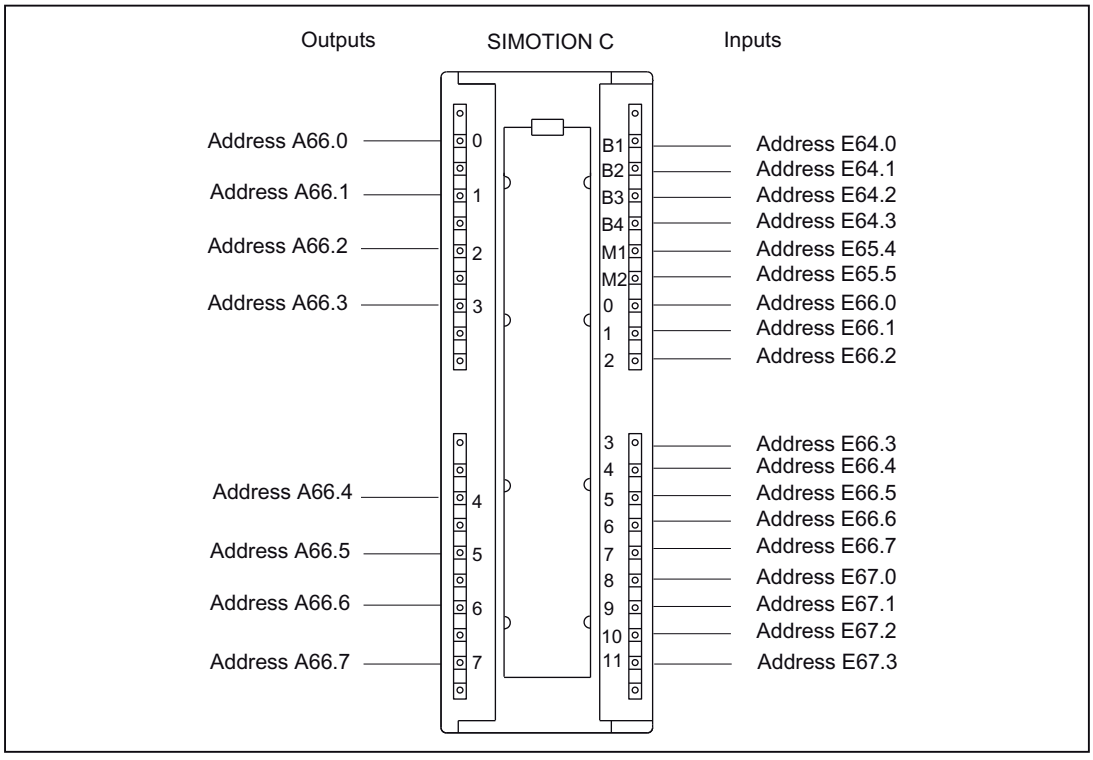


Figure 9-71 Addressing the onboard digital inputs/outputs

Note

These addresses can be modified by the user in the *hardware configuration* (see **SIMOTION SCOUT** online help).

When used by a TO (e.g. measuring input/output cam), the address must be ≥ 64.

All start addresses/signal bits (64.4...64.7, 65.0...65.3, 65.6, 65.7, 67.4...67.7) not listed do not have any defined values and consequently may not be used for evaluation.

As of SIMOTION V4.2, the onboard digital inputs/outputs of the C240 and C240 PN cannot only be interconnected with variables via their direct addresses, by also by means of symbolic assignment.

9.1.7.5 Addressing the onboard drive and measuring system interface of the C230-2, C240

Use as a standard output is only possible by means of an I/O variable. The I/O variables must be created on the addresses of the corresponding axis channel specified via the **hardware configuration** (relating to the default start address).

Symbolic assignment is not available for this application.

The following figure shows the default start addresses of the onboard drive interface of the C230-2/C240.

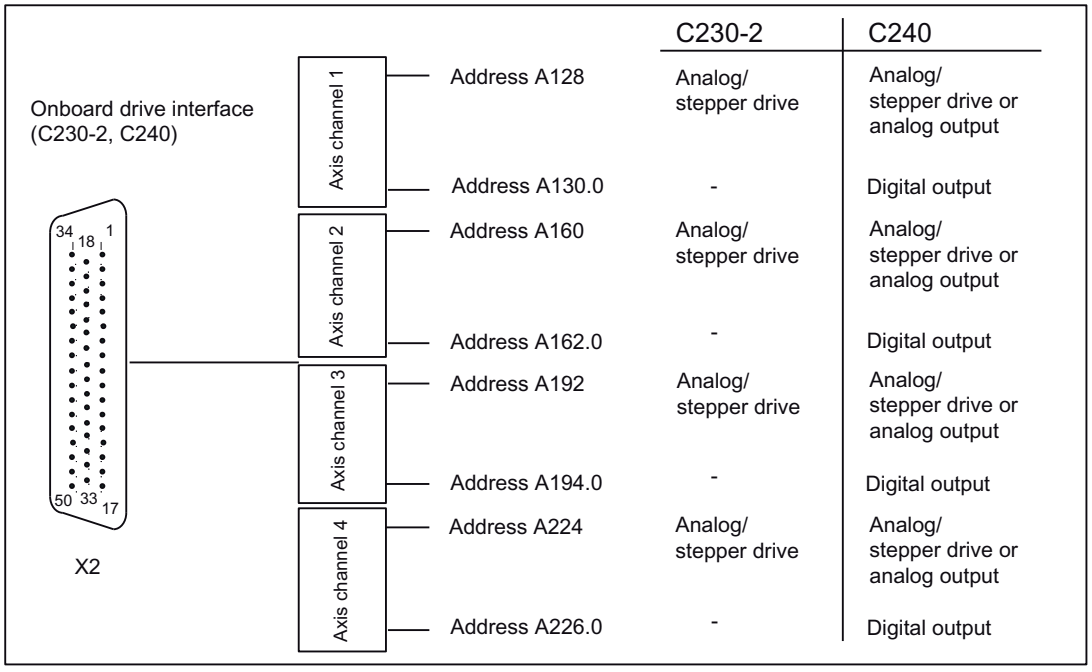


Figure 9-72 Addressing the onboard drive interface

An available encoder input of an axis channel (e.g. for a speed-controlled axis) can be used as an input for a 16-bit up/down counter (90-degree pulse train of a connected TTL encoder, zero pulse not required). The counter value can be accessed by means of an I/O variable (default start address of the axis channel). Symbolic assignment is not available for this application.

The following figure shows the default start addresses of the onboard measuring system interface of the C230-2/C240.

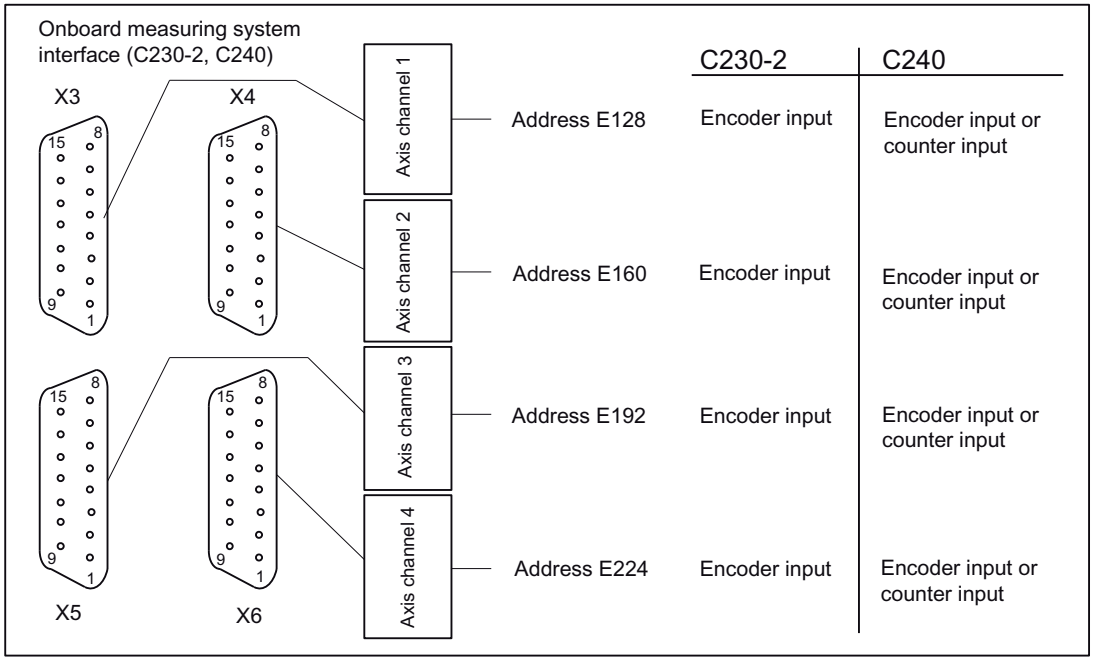


Figure 9-73 Addressing the onboard measuring system interface

9.1.8 Commissioning

9.1.8.1 Requirements for commissioning

Requirements

Table 9-46 Requirements

Prerequisite action	See ...
Your system with SIMOTION C is installed.	ChapterInstalling (Page 6809)
Your system with SIMOTION C is wired.	ChapterWiring (Page 6815)
For networking via PROFIBUS DP • The PROFIBUS addresses of the bus nodes with which the SIMOTION C communicates, are set. • The terminating resistors are switched on (at segment ends).	ChapterNetworking (Page 6842)
When networking via the Ethernet, the IP address / subnet mask of the bus nodes with which the SIMOTION C communicates, are set.	ChapterConfiguring an Ethernet subnet on the Ethernet interface (Page 6847)
When networking via PROFINET IO, the PROFINET device names of the bus nodes with which the C240 PN communicates, are set	ChapterConfiguring an Ethernet subnet on the PROFINET interface (C240 PN) (Page 6849)

System requirements

Note

The readme file for the software version you are using provides information about the hardware and software requirements.

Please take note of the information on the current CD for "SIMOTION SCOUT"!

For online operation, a connection must be established between the PG/PC and the SIMOTION C via PROFIBUS DP, Ethernet or PROFINET (C240 PN), see chapter Overview of wiring diagram (Page 6818).

The programming device must be equipped with a PROFIBUS or Ethernet card.

You need an MMC adapter to write to the micro memory card (SIMOTION Kernel update) on the PG/PC, see chapter Writing, formatting and erasing the Micro Memory Card (Page 6862).

Connecting a PG/PC to a SIMOTION C

You can interconnect the programming device / PC

- with the PROFIBUS of SIMOTION C (connector X8 and/or X9) using a connecting cable (see chapter Network components for a PROFIBUS subnet (Page 6845)).
  Information on the respective cable lengths for PROFIBUS DP can be found in chapter Onboard measuring system interface (C230-2, C240) (Page 6784).
- to the Ethernet of the SIMOTION C (connector X7) with a shielded twisted pair cable.
  For information about cabling the Ethernet subnet, refer to chapter Configuring an Ethernet subnet on the Ethernet interface (Page 6847).
- to the PROFINET interface X11 (port P1, P2 or P3) of the C240 PN with a shielded twisted pair cable.
  For information about cabling the PROFINET subnet, refer to chapter Configuring an Ethernet subnet on the PROFINET interface (C240 PN) (Page 6849).

9.1.8.2 Inserting and changing the Micro Memory Card

Procedure:

1. Switch off the power supply module.
2. Is a micro memory card inserted? If yes: remove the memory card.
   An ejector is located on the frame of the module receptacle to enable you to remove the micro memory card.
   Press the ejector and remove the micro memory card.

3. Applying slight pressure, insert the ("new") micro memory card into the slot of the SIMOTION C until it snaps into place. Make sure the beveled edge of the micro memory card faces the ejector (see the following figure).
4. Switch the power supply module on again.

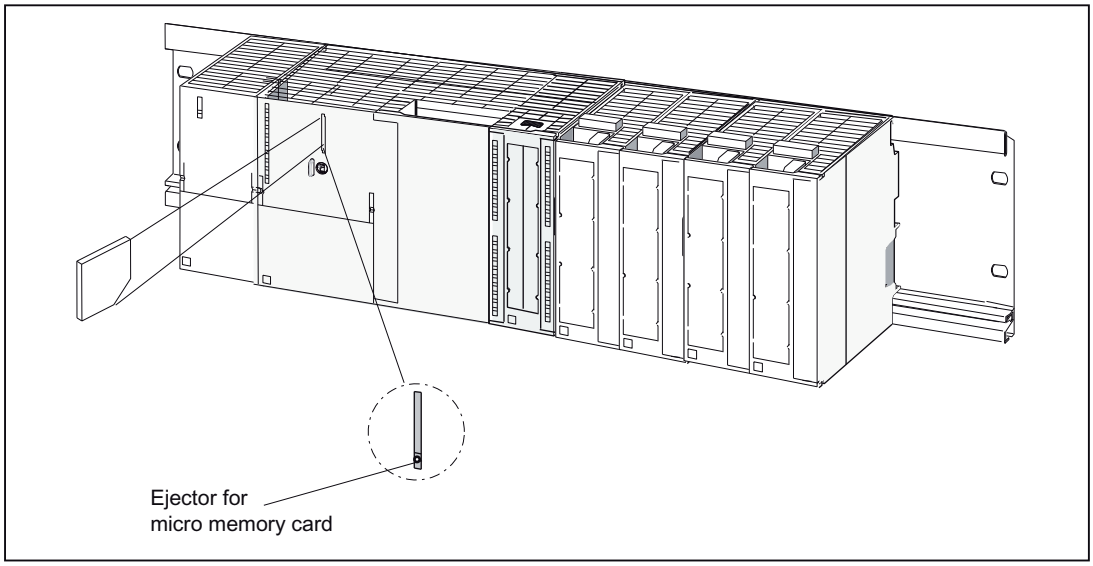


Figure 9-74 Inserting the micro memory card into the SIMOTION C

Note

With the **SIMOTION C240 / C240 PN**, the micro memory card must **always** be inserted for operation.

For the purpose of data backup, the contents of the micro memory card can be copied to the hard drive of a PG/PC with an MMC card reader.

9.1.8.3 Initial Power ON

Requirements

- You have completed installation and the wiring of your system with SIMOTION C.
- The micro memory card is inserted.
- The mode selector must be set to STOP!

Initial Power ON

Switch on the power supply module.

- The 24 VDC LED on the power supply module is illuminated.
- On the SIMOTION C:
  - The 5 VDC LED illuminates.
  - All other LEDs light up briefly (approx. 2 seconds).

Note

For C230-2

With an empty micro memory card, the SIMOTION Kernel on the C230-2 is copied to the micro memory card during power-up. This increases the duration of the power-up by about 1.5 minutes.

For C240 / C240 PN

If a micro memory card is not inserted in the C240 or if the micro memory card does not contain a SIMOTION Kernel, all LEDs except the RUN LED are illuminated on the SIMOTION C240.

9.1.8.4 Writing, formatting and erasing the Micro Memory Card

Writing the micro memory card of SIMOTION C

You can write to the micro memory card as follows:

- Micro memory card is inserted in the SIMOTION C and is written with the "Copy RAM to ROM" menu command (there must be a connection between the programming device and SIMOTION C).
  You can store the technology packages and user data (programs, configuration data, parameter assignments) on the micro memory card (see *SIMOTION SCOUT* online help).
- Write to the micro memory card on the PG or PC.
  You can write to the micro memory card directly via a PC using a suitable memory card adapter.
  This function is required for updating the SIMOTION C.

You can write the user project with SIMOTION SCOUT and the menu command "Load to file system" directly from the PC to the memory card (micro memory card). To do this, select the device in the project navigator and execute the "Load to file system" command in the context menu.

Note the following information when handling a micro memory card.

Note

The micro memory card always comes formatted! With the **C240 / C240 PN**, the SIMOTION Kernel is located on the micro memory card.

In order to ensure error-free functioning of the micro memory card in the SIMOTION C, the card must **not** be repartitioned.

Using Windows media to modify or delete files on the micro memory card that were written with "Copy RAM to ROM" can destroy the project.

With the **C230-2**, the following information must also be noted.

Note

The micro memory card for the C230-2 can only be formatted in the "SIMOTION SCOUT" engineering system.

As of Kernel Version 3.1, you will **no longer** be able to format the micro memory card by means of the C230-2 mode selector.

The card must **not** be formatted with the PC.

Any saved license keys are also deleted during formatting. The license key must then be reentered via SIMOTION SCOUT.

The micro memory card can be formatted by calling this function in "SIMOTION SCOUT" (see *SIMOTION SCOUT* online help).

Note

As of Kernel Version 2.1 in the C230-2 and micro memory card 6AU1 700-0AA02-0AA0, the C230-2 transfers its stored kernel to the micro memory card after formatting **only** after the C230-2 has been restarted. When the kernel is transferred, all LEDs on the C230-2 flash. If a specific kernel version is required, it must be placed on the micro memory card manually.

If the C230-2 is switched off or de-energized during commissioning, this could damage the file system on the micro memory card (data on the card can no longer be read). If this occurs, the micro memory card **must** be reformatted.

The micro memory card must **not** be removed when energized.

Repairing the micro memory card

You can repair the micro memory card, for example, if it is faulty. The MMC card can be inserted in a USB Flash card reader and formatted by means of Windows (FAT file system). After it is formatted, the boot sector of the card must be rewritten to by means of SCOUT ("Options" > "Write to Boot Sector...").

With the micro memory card for the C240, the SIMOTION Kernel must be copied to the micro memory card again, see chapter Kernel update for SIMOTION C240 / C240 PN (Page 6874).

9.1.8.5 User memory concept

SIMOTION C memory model

The following figure provides an overview of the SIMOTION C memory model.

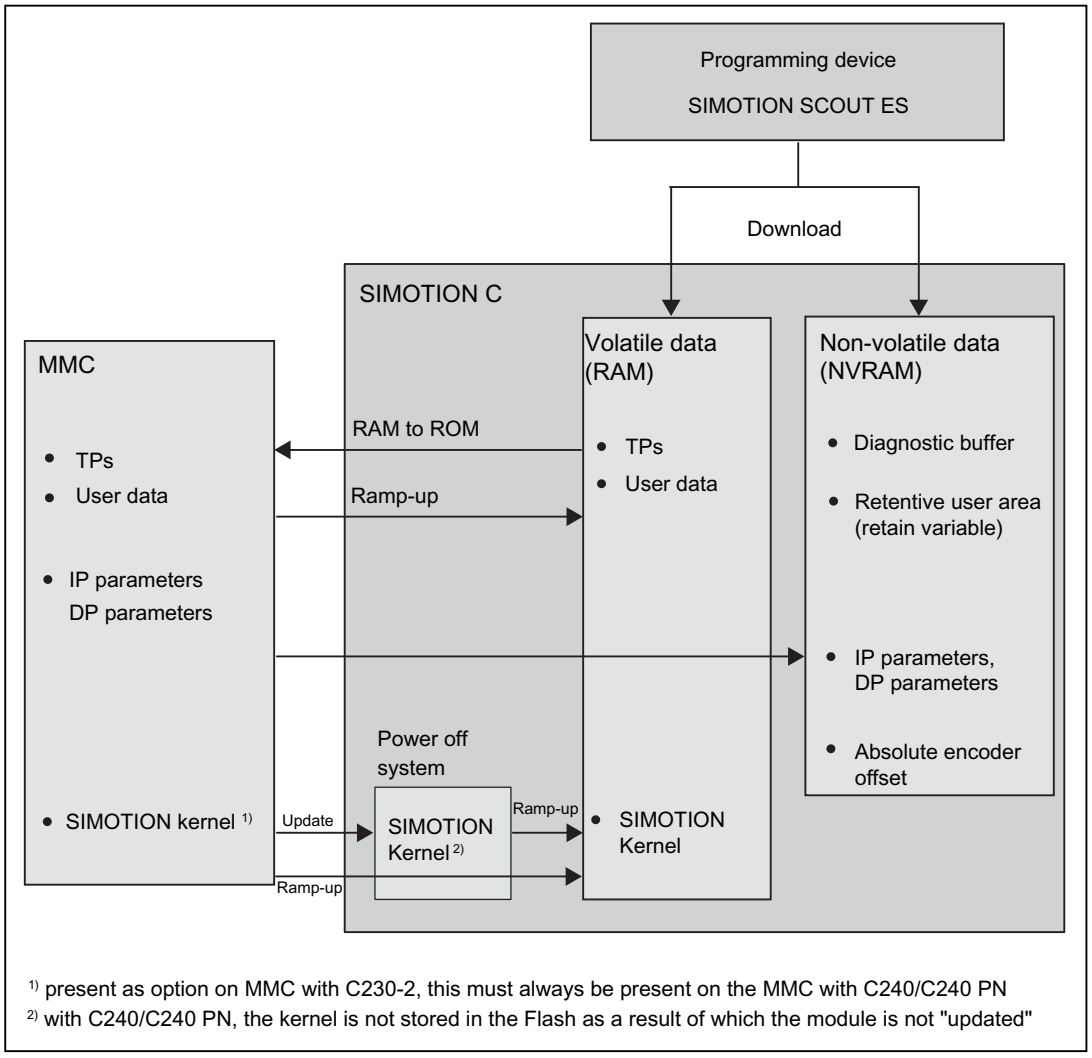


Figure 9-75 SIMOTION C memory model

In the following sections, you will learn information about the user memories and the steps involved in certain operations.

See also

Overview of data deletion (Page 6868)

Properties of the user memory

Non-volatile data (NVRAM)

Non-volatile data is used with the objective of retaining user- and system-relevant data when the SIMOTION C is de-energized. You will find information about the area that can be used for non-volatile data in the SIMOTION SCOUT Configuration Manual.

The following non-volatile data is available in a SIMOTION C:

Table 9-47 Content of non-volatile data

Non-volatile data	Content
Kernel data	• Last operating state • IP parameters (IP address, subnet mask, router address) • DP parameters (DP addresses, baud rate) • Diagnostic buffer
Retain variables	• Variables in the interface or implementation section of a UNIT declared with **VAR_GLOBAL RETAIN** • Global device variables set with the "**RETAIN**" attribute
Retain TO	• Absolute encoder offset

The non-volatile data of a SIMOTION C has the following properties:

Table 9-48 Properties of non-volatile data

Property	Meaning
Location	In the NVRAM of the SIMOTION C, no back-up battery required
Backup time	Unlimited

Note

IP and DP parameters in non-volatile data

If there is a configuration on the MMC, the IP and DP parameters are loaded from the MMC during ramp-up, written to the non-volatile data and used by the SIMOTION C. The SIMOTION C uses the addresses defined in these parameters to go online. The IP and DP parameters in the non-volatile data are retained and used by the SIMOTION C if a ramp-up is performed with an MMC that does not contain a configuration.

A SIMOTION C can therefore always go online if a configuration has been loaded at least once with the SIMOTION SCOUT, or if the SIMOTION C has been ramped up once with an MMC.

Volatile data (RAM)

Definition of the properties of volatile data:

- The volatile data is located in the RAM memory of the SIMOTION C.
- The download data of SIMOTION SCOUT are written to this memory.
- This data is lost when the SIMOTION C is shut down.
- Contents of the "volatile data" area:
  - SIMOTION Kernel
  - Technology packages (TP)
  - User data (programs, configuration data, parameterizations, task configuration)

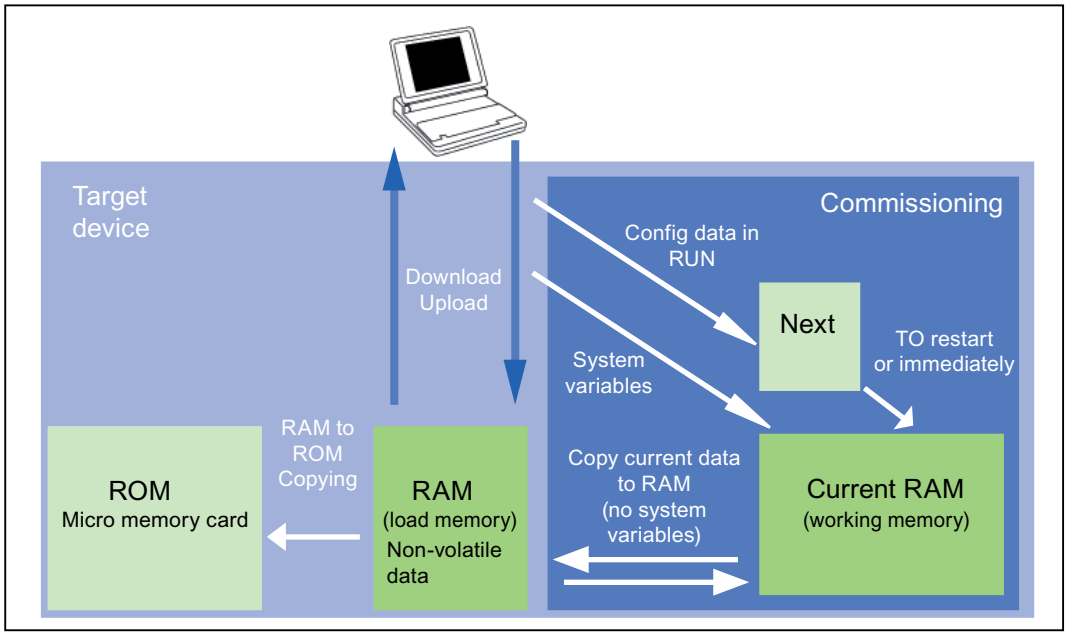


Figure 9-76 Configuration data and system variables on volatile memory

Non-volatile system

With the SIMOTION C230-2, the SIMOTION Kernel is stored as non-volatile data in a flash memory. A C230-2 can therefore also ramp up without a memory card.

MMC memory card

The MMC contains the following data:

- SIMOTION Kernel
- Technology packages (TP)
- User data (programs, configuration data, parameterizations, task configuration)
- IP parameters (IP address, subnet mask, router address)
- DP parameters (DP addresses, baud rate)

Operations and their effect on the user memory

The next section describes the operations identified in the "SIMOTION C memory model" figure by arrows and their effect on the user memory.

SIMOTION SCOUT download

The "Download" command transfers the following data from the engineering system to the "volatile data" area:

- User data (programs, configuration data, parameterizations, task configuration)
- Technology packages

The IP and DP parameters are also saved to the "non-volatile data". Depending on the setting in SIMOTION SCOUT, the retain variables are set to their initial values.

The "volatile data" is lost if the SIMOTION C is switched off after a download without having performed a "Copy RAM to ROM".

Copy RAM to ROM

The "Copy RAM to ROM" command is used on the engineering system to save the following data to the MMC:

- Technology packages and user data (programs, configuration data, parameterizations, task configuration) of the "volatile data" area
- Depending on the setting in SIMOTION SCOUT, actual values can be copied to the "volatile data" area before copying the data from RAM to ROM.

Note

The "Copy RAM to ROM" command does not save the actual values of the retain variables to the MMC.

SIMOTION C ramp-up

During ramp-up, the SIMOTION Kernel is loaded to the "volatile data" area. A C230-2 loads the kernel from the "non-volatile system" (integrated flash in the device), a C240/C240PN loads the kernel from the MMC.

The following data is also loaded from the MMC during ramp-up:

- Technology packages and user data to the "volatile data"
- IP and DP parameters to the "non-volatile data"

SIMOTION Kernel update (only valid for SIMOTION C230-2)

An update writes the SIMOTION Kernel from the micro memory card to the "non-volatile system". If a newer or an older version of the SIMOTION Kernel is to be installed, an update can be performed with a micro memory card that contains the appropriate SIMOTION Kernel.

Backup of non-volatile data

The "_savePersistentMemoryData" system function is used to save the contents of "non-volatile data" to the MMC. This backup prevents the retain variables and the absolute encoder position from being lost if a component is replaced.

The backup copy is saved to the "PMEMORY.XML" backup file in the "USER/SIMOTION" folder. On the system side, this system function ensures that a consistent overall image of the non-volatile data is always available the next time the unit is powered on, even if there is a power failure during backup. An already existing backup file is renamed to "PMEMORY.BAK" before a new backup file is generated. If the backup to this new file fails (for example, due to insufficient storage capacity of the memory card), the existing backup file is used in the next attempt to restore the content of the "non-volatile data". The backup file is deleted if the new file was successfully created.

Note

If you do not save the "non-volatile data" to the MMC, it is lost when a spare part is used (in the event of a module defect).

If an absolute encoder overflow occurs after **_savePersistentMemoryData**, the actual position value is no longer correct after the non-volatile data is restored. In this case, homing (absolute encoder adjustment) must be repeated.

Power failure

The "non-volatile data" is saved to the NVRAM of the SIMOTION C during a power failure. The "non-volatile data" is available again at the next ramp-up. Thus, the SIMOTION C is immediately ready for operation without data loss.

9.1.8.6 Deleting data

Overview of data deletion

You can define the scope of data to be deleted from SIMOTION C memory described in the "user memory concept". This enables you to determine whether data in your system should be deleted completely or partially.

You have the following options for deleting SIMOTION C data:

- SIMOTION C memory reset
- Deleting user data from the micro memory card
- Setting SIMOTION C to factory settings

See also

SIMOTION C memory model (Page 6864)

SIMOTION C memory reset

Introduction

During the memory reset, the "volatile data" in the RAM of the SIMOTION C and the "non-volatile data" in the NVRAM, except for the communication configuration (baud rates, network addresses, etc.), is deleted. The data on the MMC is retained during the memory reset.

You must perform a memory reset of the SIMOTION C:

- When you want to undo changes you have made to your user data (programs, configuration data, parameter assignments) that you have not backed up with the "Copy RAM to ROM" menu command.
- If the SIMOTION C requests a memory reset with a flashing STOP LED (slow flashing) (e.g. micro memory card has been removed).
- The "non-volatile data" does not match the project on the MMC and therefore an error occurs.

You can perform the memory reset online via SIMOTION SCOUT or offline via the mode selector on the SIMOTION C.

Data deleted on memory reset

The following data is deleted during a memory reset:

- User data (programs, configuration data, parameterizations, task configuration)
- Technology packages
- Retain TO (absolute encoder adjustment)
- Retain variables
  Retain variables are variables in the interface or implementation section of a UNIT that are declared with VAR_GLOBAL RETAIN, or global device variables with the RETAIN attribute.

Note

Absolute encoder data is deleted during a memory reset operation and must therefore be readjusted after the memory reset.

Reset-proof data

The following data is retained during a memory reset:

- TCP/IP and DP parameters
- Diagnostic buffer
- Data saved with the_savePersistentMemoryData, _saveUnitDataSet, exportUnitDataSet and RAMtoROM commands
  If backup files (PMEMORY.XML/PMEMORY.BAK) have been backed up with _savePersistentMemoryData, the data in these files is backed up again to the non-volatile data after the memory reset. Users can therefore force the restoration of non-volatile data by means of memory reset.
- Licenses

The technology packages and user data (configuration data, programs, parameterizations) that were previously backed up to the micro memory card using the "Copy RAM to ROM" menu command will be transferred to the "non-volatile data" area of the SIMOTION C during the next ramp-up. **Thus, an existing configuration on the MMC is loaded to the SIMOTION device following the memory reset.**

Memory reset by means of SIMOTION SCOUT

The SIMOTION device for which overall reset is to be carried out must be online.

1. In SIMOTION SCOUT, open the Control Operating State dialog: Select the Target system > Control Operating State menu command. Or click on Control Operating State in the toolbar.
2. Switch the SIMOTION device for which overall reset is to be carried out to STOP in the Control Operating State dialog.
3. Select the SIMOTION device under Overall reset (MRES). Click the Execute switch. Confirm the command by clicking "Yes".
   The memory reset will now be performed.

Memory reset with the mode selector

Note

The operation for the memory reset with the toggle switch is described below. The operation with the key-operated switch of the C230-2 is similar to the operation with the toggle switch.

Proceed as follows (see following figure):

1. Set the mode selector to STOP.
2. Move the switch to MRES and hold the switch in this position until the STOP LED changes from flashing to steady illumination.
3. Within 3 seconds, you must release the switch and return it to the MRES position. The memory reset will now be performed. The SIMOTION C completed the memory reset when the STOP LED lights up permanently.

The SIMOTION C has reset the memory.

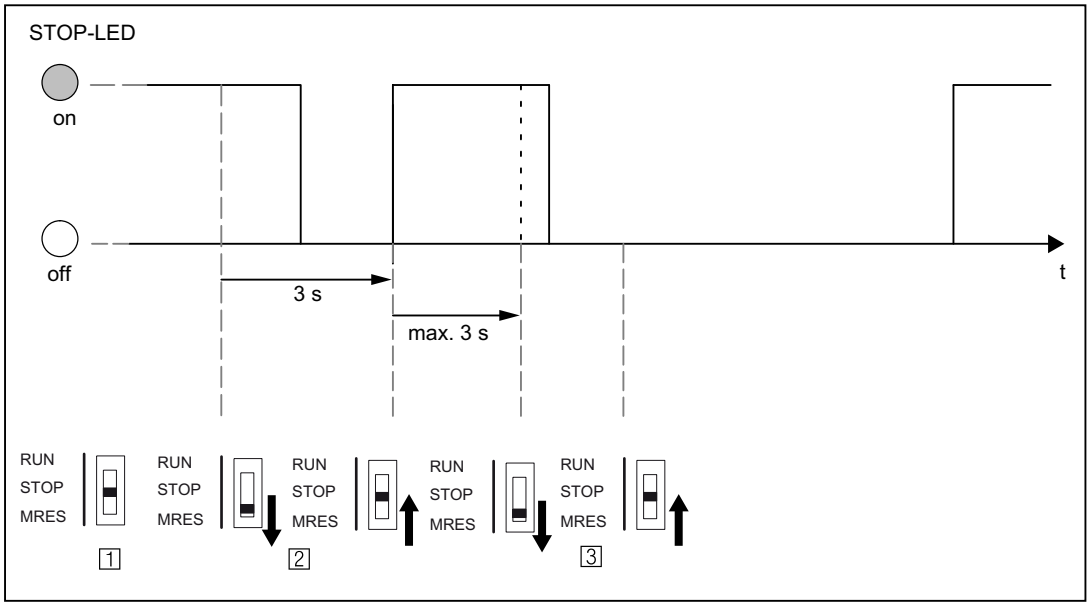


Figure 9-77 Mode selector operation sequence for the memory reset

Deleting user data from Micro Memory Card

Requirement

You can delete user data with SIMOTION SCOUT. To do so, you must be online on the SIMOTION C. The following data is deleted during this operation:

- User data in the volatile data
- Non-volatile data, except for the IP and DP parameters
- User data on the MMC (user directories)

You can thus continue to go online to the SIMOTION C with your PG/PC. The licenses on the MMC are retained.

Deleting user data

1. In SIMOTION SCOUT, open the project you want to modify.
2. Go online with the SIMOTION C.
3. Select the "Delete user data on card" option in the "Target system" menu.
4. Confirm the "Delete User Data from Card" prompt with "OK".
   The user data are deleted.

Setting SIMOTION C to factory settings

Overview

SIMOTION C is supplied with preset parameters, such as the transmission rate or PROFIBUS addresses. You can restore the factory settings with the mode selector. The following data is deleted during this operation:

- "Non-volatile data" in the SIMOTION device
- The backup (PMEMORY.XML/PMEMORY.BAK) of the non-volatile data on the MMC
- User data in the "volatile data" and on the MMC
- The communication configuration (IP and DP parameters) is set to the factory settings

The licenses on the micro memory card are retained.

Operations on the mode selector

Proceed as follows:

1. Switch off the power supply for SIMOTION C.
2. The micro memory card is inserted in the SIMOTION C.
3. Switch the mode selector to the MRES position. Keep this position selected and switch the power supply for the SIMOTION C on. When the STOP LED display lights up statically, switch the mode selector switch back to the STOP position.

Note

After switching on the power supply, it takes approx. one minute until the SIMOTION C deletes the "permanent power off" data, performs a restart and has reached the STOP state.

Note

The communication configuration is now reset to the factory settings. The communication configuration for the SIMOTION C must be repeated.

See also

Factory setting (Page 6851)

9.1.9 Maintenance and servicing

9.1.9.1 SIMOTION kernel update

Kernel update for SIMOTION C230-2

Each C230-2 always contains the latest version of the SIMOTION Kernel when it is delivered. The user can install newer or older versions via an update.

The corresponding SIMOTION Kernel version is included on the respective SIMOTION SCOUT Add-On CD-ROM. (Proceed in the same way with a SIMOTION SCOUT DVD)

Note:
The SIMOTION Kernel and the technology packages must always have the same software version. To avoid incompatibilities when replacing the C230-2, it is recommended that the SIMOTION Kernel version of the current configuration is stored on the micro memory card.

You can use a PG/PC with the appropriate micro memory card adapter to copy the SIMOTION Kernel to the micro memory card.

Proceed as follows:

1. Connect the SC card / MMC adapter to your PG/PC and insert the micro memory card in the module slot of the adapter.
2. Insert the CD ROM containing the SIMOTION Kernel in the CD drive of your PG/PC.
3. Open Windows Explorer.
   Note
   The micro memory card must be visible as a drive with an arbitrary letter in the Windows Explorer.
4. Delete all files and folders from the MMC except for the complete KEYS folder. The KEYS.TXT file in this folder contains the license key and must be retained.
5. Unpack the firmware contained in the ZIP file (\3_C2xx\Firmware_C230\V....\c230_2fw.zip) on the CD-ROM and copy the c230_2fw.bin file using the Windows Explorer to the root directory of the micro memory card.
6. Remove the micro memory card from the PG/PC.
7. Switch off the power supply for the C230-2.
8. Insert the prepared micro memory card into the C230-2.
9. Switch on the power supply for the C230-2.
   The C230-2 compares the version of the SIMOTION Kernel on the micro memory card with that on the C230-2 and automatically carries out an update.
   - First, the version of the SIMOTION Kernel stored in the C230-2 is deleted. The "RUN", "STOP" and "BUS2F" LEDs flash.
   - Then the new version of the SIMOTION Kernel is transferred from the micro memory card to the C230-2. During the data transfer, the "SF", "RUN", "STOPU", "STOP", "BUS1F" and "BUS2F" LEDs illuminate in sequence.

   Note: The C230-2 must **not** be switched off during this phase.
   - After completing the SIMOTION Kernel update, the C230-2 performs a restart and goes into the STOP mode.

The C230-2 can then be operated with the new SIMOTION Kernel.

Note

Following an update, if problems occur during restart, the "SF" LED will illuminate or the "STOPU", "STOP", "BUS1F", and "BUS2F" LEDs will flicker. This means that the update was not completed correctly.

To remedy this error, proceed as follows:

- Power OFF/ON
- Check whether the restart is being performed correctly

If the error occurs again, repeat the update or replace the module.

Kernel update for SIMOTION C240 / C240 PN

The C240 / C240 PN does not contain any firmware.

The Micro Memory Card of the C240 / C240 PN always contains the latest version of the SIMOTION Kernel on delivery. The user can install newer or older versions via an update.

The corresponding SIMOTION Kernel version is included on the respective SIMOTION SCOUT Add-On CD-ROM. (Proceed in the same way with a SIMOTION SCOUT DVD)

Note:
The SIMOTION Kernel and the technology packages must always have the same software version.

You can use a PG/PC with the appropriate Micro Memory Card adapter to copy the SIMOTION Kernel to the Micro Memory Card.

Proceed as follows:

1. Connect the SD card / MMC adapter to your PG/PC and insert the Micro Memory Card in the slot of the adapter.
2. Insert the CD ROM containing the SIMOTION Kernel in the CD drive of your PG/PC.
3. Open Windows Explorer.
   Note
   The Micro Memory Card must be visible as a removable data carrier with an arbitrary drive letter in the Windows Explorer.
4. Delete all files and folders from the MMC except for the complete KEYS folder. The KEYS.TXT file in this folder contains the license key and must be retained.

5. Unpack the firmware contained in the ZIP file (\3_C2xx\Firmware_C240\V...\c240_fw.zip) on the CD-ROM and copy the following files and folders to the root directory of the Micro Memory Card using Windows Explorer.
   - **c240_fw1.bin**
   - **c240_fw2.bin**
   - **c240_fw.bin**
   - **startup.txt**
   - **toc.txt**
   - **SIEMENS\SIMOTION\cbe30.ufw**
6. Remove the Micro Memory Card from the PG/PC.
7. Switch off the power supply to the C240 / C240 PN.
8. Insert the prepared Micro Memory Card into the C240 / C240 PN.
9. Switch on the power supply to the C240 / C240 PN.

The C240 / C240 PN can then be operated with the new SIMOTION Kernel.

Note

If the Micro Memory Card does not contain a SIMOTION Kernel, all LEDs on the SIMOTION C240 / C240 PN except the RUN LED are illuminated.

Firmware compatibility as of V4.5

A SIMOTION Kernel as of V4.5 can only be used in conjunction with a SIMOTION C240 / C240 PN controller as of production version G.

9.1.9.2 Removal and replacement of the SIMOTION C

Overview

You can only replace SIMOTION C as a complete unit.

⚠ WARNING

The SIMOTION C can only be replaced when the load power supply is switched off.
You must therefore switch off the power supply, e.g. by means of the on/off switch on the PS module.

Removing a faulty module

To remove the SIMOTION C, proceed as follows:

1. Switch off the power supply.
2. Remove the micro memory card.
3. Open the front door panels. If necessary, remove the labeling strips.
4. Undo the connections on the terminal strip for the power supply.
5. Depending on the controller used, disconnect the encoders (X3...X6), the drive unit (X2), the PROFIBUS DP interfaces (X8, X9) as well as the Ethernet (X7) and PROFINET interfaces (X11 P1 to X11 P3).
6. Loosen the mounting screw in the middle of the front connector (X1) and then pull out the front connector, holding it at the gripping points provided.
7. Unscrew the module's mounting screws and swing it upwards and out.

Installing a new module

Procedure:

1. Remove the upper part of the front connector coding from the new module.
2. Insert a module of the same type, swing it down and screw it in tightly.
3. Insert the front connector and then tighten the mounting screw.
4. Depending on the controller used, connect the encoders (X3...X6), the drive unit (X2), the PROFIBUS DP interfaces (X8, X9) as well as the Ethernet (X7) and PROFINET interfaces (X11 P1 to X11 P3).
5. Connect the load power supply on the terminal strip.
6. Close the front panel doors and put in the labeling strips.
   The controller is once again ready for operation and can be commissioned.
7. Insert the micro memory card.
8. Switch on the power supply.

9.1.9.3 Module replacement without programming device or PC

Overview

If a defective SIMOTION C has to be replaced without using a PG/PC, you can transfer the data of the defective module to the new module by simply inserting the micro memory card of the defective module in the new module.

Requirement

When you commissioned your project, you must have used the menu command "Copy RAM to ROM" of the SIMOTION SCOUT to save your project to the micro memory card, as described in the chapter "User memory concept".

The following data are on the micro memory card and can be transferred to the new module:

- Technology packages (TP)
- User data (programs, configuration data, parameterizations, task configuration)
- DP parameters (DP addresses, baud rate)
- IP parameters (IP address, subnet mask, router address)

As the non-volatile data is in a memory within the module, it is lost when the module is replaced.

If the non-volatile data is to be transferred to the new module, it must first be backed up on the micro memory card of the module to be replaced. The restoration of the non-volatile data must be triggered via a memory reset on the new module.

Note

If the non-volatile data is not restored, the retain variables are set to their initial values and the diagnostics buffer with the history from the old module is no longer available.

9.1.10 Alarm, error, and system messages

9.1.10.1 Diagnosis using the LEDs

Diagnostic LEDs

The LEDs are explained in the order in which they are positioned on the SIMOTION C.

Table 9-49 Diagnostics LEDs of the SIMOTION C

Display	Meaning	Explanations
SF (red)	**System fault**	**This LED indicates a fault on the SIMOTION C.**
LED - ON		An event which can be acknowledged is present (alarm, message, note). (see documentation package *SIMOTION System and Function Descriptions*)
LED - flashing (0.5 Hz)		No license exists for technology/optional objects under license. Technology/option objects under license include, for example: • Cam (synchronous axes with connected cam) • Ethernet (Ethernet interface) • TControl (Temperature Control)
LED - OFF		SIMOTION C is operating without error.
5 VDC (green)	**Power supply for the electronics**	**This LED indicates that the power supply is ready.**
LED - ON		The power supply of the SIMOTION C is operating without error.

Display	Meaning	Explanations
LED - OFF		If this is not illuminated, the reason may be: • No connected or switched-on network • No specified load power supply connected • Module not connected correctly • Module defective
RUN (green)	**SIMOTION C in RUN mode**	**This LED indicates that the user program is running.**
LED - ON		See RUN (green)
LED - flashing (2 Hz)		The time between selection of "RUN" mode until this mode has been attained is indicated by the LED flashing.
STOPU (yellow)	**SIMOTION C in STOP user program mode**	**This LED indicates that the technology packages are active. A user program is not being executed.**
LED - ON		See STOPU (yellow)
LED - flashing (2 Hz)		The time from when the "STOPU" operating mode is selected until this operating mode has been attained is indicated by the LED flashing.
LED - "flickering"		Formatting the micro memory card
STOP (yellow)	**SIMOTION C in STOP mode**	**This LED indicates that a user program is not running. The technology packages are inactive.**
LED - ON		See STOP (yellow)
LED - flashing (2 Hz)		The time from when the "STOP" operating mode is selected until this operating mode has been attained is indicated by the LED flashing.
LED - flashing (0.5 Hz)		A memory reset request is indicated by slow flashing, see chapter SIMOTION C memory reset (Page 6869).
BUS1F (X8) (red) or BUS2F (X9) (red)	**Fault on interface**	**This LED indicates a fault on the PROFIBUS DP interface.**
LED - ON		*Interface configured as slave:* Bus fault Search for baud rate
LED - flashing (0.5 Hz)		Parameterization error No cyclic data exchange
LED - OFF		Cyclic data exchange
LED - ON		*Interface configured as master:* Bus fault (bus short circuit)
LED - flashing (0.5 Hz)		Bus fault (slaves have failed or cable faulty)
LED - OFF		No fault or no interface configured

Table 9-50 Status and fault displays behind the front cover

Display	Meaning	Explanations
Link X7 (green)	**Link status of Ethernet interface**	**This LED indicates a physical connection of the Ethernet interface X7.**
LED - ON		A device is connected and there is a physical connection to this device.
LED - OFF		A device is not connected and there is no physical connection.

Display	Meaning	Explanations
Activity X7 (yellow)	**Activity status of Ethernet interface**	**This LED indicates a data transfer via the Ethernet interface X7.**
LED - flashing		Data is being received or transmitted.
LED - OFF		Data is not being received or transmitted.
Link X11 Px (green)	**Link status of PROFINET interface**	**This LED indicates a physical connection of the PROFINET interface X11 at port x (1 to 3).**
LED - ON		A device is connected and there is a physical connection to this device.
LED - OFF		A device is not connected and there is no physical connection.
Activity X11 Px (yellow)	**Activity status of PROFINET interface**	**This LED indicates a data transfer via the PROFINET interface X11 at port x (1 to 3).**
LED flashing or LED ON		Data is being received or transmitted.
LED - OFF		Data is not being received or transmitted.
Fault X11 (red)	**Fault status of PROFINET interface**	**This LED indicates a fault at the PROFINET interface X11.**
LED - OFF		PROFINET interface is operating without error; the data exchange to all configured I/O devices is running.
LED - flashing (2 Hz)[1)]		Bus fault • Failure of a connected I/O device. • At least one of the assigned I/O devices cannot be addressed • Incorrect or no configuration
Sync X11 (green)[2)]	**Sync status of PROFINET interface**	**This LED indicates the synchronization status of the PROFINET interface X11.**
LED - OFF		The PROFINET interface has not synchronized yet to the send cycle of IRT or PROFINET with IRT has not been configured (e.g. only RT or TCP/IP operation). If no PROFINET IO with IRT has been configured, then generally there is no synchronization of the PROFINET interface. If isochronous data has been configured for the SIMOTION device, the PROFINET interface generates a local substitute cycle of the same size as the configured send cycle of IRT, as long as no synchronization has been performed to the send cycle of IRT: • The task system of SIMOTION has synchronized to the local substitute cycle of the PROFINET interface. • DP interfaces, if configured as isochronous, are synchronized to the local substitute cycle of the PROFINET interface.

Display	Meaning	Explanations
LED - flashing (2 Hz)		The PROFINET interface has synchronized to the send cycle of IRT. If isochronous data has been configured for SIMOTION, then the following applies: • The task system of SIMOTION has synchronized to the send cycle of IRT. • DP interfaces, if configured as isochronous, have not synchronized yet to the send cycle of IRT.
LED - ON		The PROFINET interface has synchronized to the send cycle of IRT. If isochronous data has been configured, then the following applies: • The task system of SIMOTION has synchronized to the send cycle of IRT. • DP interfaces, if configured as isochronous, are synchronized to the send cycle of IRT. If no isochronous data has been configured for SIMOTION, this state indicates that only the PROFINET interface has synchronized to the send cycle of IRT. The PROFINET interface is then used to forward the data. Isochronous data means that data is exchanged with a further device on PROFINET with configured RT class IRT and isochronous application/device.

1) Minimum flashing duration 3 s

2) If no PROFINET IO with IRT has been configured, then generally no synchronization will be made to the send cycle.

Note

During ramp-up of the SIMOTION C, the LEDs on the housing front are briefly illuminated.

You can carry out a detailed diagnosis with a programming device or PC and the engineering aystem.

Projects cannot be downloaded in **STOPU** mode.

9.1.10.2 Combinations of LED displays

Combination of LED displays

The following table provides an overview of all permissible and/or required LED display combinations.

The meaning of the displays used in the table is as follows:

1	LED on
0	LED off
0.5/1	LED flashing (0.5 Hz)
2/1	LED flashing (2 Hz)
☆	LED flickers
→	Running light
x	LED may light up

Table 9-51 Summary of LED displays

Meaning	LED displays						
	SF (red)	5 VDC (green)	RUN (green)	STOPU (yellow)	STOP (yellow)	BUS1F (red)	BUS2F (red)
Ramp-up (C230-2)	1 0 0	1 1 1	1 0 0	1 ☆ 0	1 0 2/1	1 0 0	1 0 0
Ramp-up (C240 / C240 PN) Ramp-up (C240 product version < G as of V4.5) SF (rot) flickers (5 Hz)	1 0 0 0 0	1 1 1 1 1	1 ☆ 0 0 0	1 0 ☆ → 0	1 0 0 → 2/1	1 0 0 0 0	1 0 0 0 0
STOPU → RUN	x	1	2/1	1	0	x	x
RUN	x	1	1	0	0	x	x
RUN → STOPU	x	1	1	2/1	0	x	x
STOPU	x	1	0	1	0	x	x
STOPU → STOP	x	1	0	1	2/1	x	x
STOP	x	1	0	0	1	x	x
STOP → STOPU	x	1	0	2/1	1	x	x
Defective operating state Remedy: • Switch SIMOTION C Off/On • Check diagnostics buffer	0	1	1	☆	☆	☆	☆
Power supply is ready for operation	x	1	x	x	x	x	x
Writing to micro memory card (Copy RAM to ROM)	x	1	0	0	☆	x	x
Formatting the micro memory card	x	1	0	☆	2/1	x	x
Request for overall reset by the SIMOTION C or via the mode selector of the SIMOTION C	x	1	0	0	0.5/1	x	x
Overall reset in progress	1	1	1	1	1	1	1
Overall reset completed	x	1	0	0	1	x	x
SIMOTION Kernel update in progress (only for SIMOTION C230-2)	→	1	→	→	→	→	→
SIMOTION Kernel update completed (only for SIMOTION C230-2)	x	1	0	0	1	x	x
SIMOTION C240 without micro memory card	1	1	0	1	1	1	1
Copy SIMOTION Kernel to micro memory card	0.5/1	1	0.5/1	0.5/1	0.5/1	0.5/1	0.5/1
SIMOTION C is operating without error	0	1	x	x	x	0	0
An event that can be acknowledged (alarm, message, note) is present	1	1	x	x	x	x	x

Meaning	LED displays						
	SF (red)	5 VDC (green)	RUN (green)	STOPU (yellow)	STOP (yellow)	BUS1F (red)	BUS2F (red)
Fault to which the user program cannot respond The following actions may be required to rectify the fault: • Switch Off/On • Check micro memory card • Re-commissioning • Replacement of SIMOTION C	x	1	x	☆	☆	☆	☆
No license exists for technology/option objects which require a license	0.5/1	1	x	x	x	x	x
SIMOTION - **STOP** state User program is running at a breakpoint	x	1	0.5/1	1	1	x	x
PROFIBUS DP interface as slave							
Bus fault Searching for baud rate	x	1	x	x	x	1	1
Parameterization error No cyclic data exchange	x	1	x	x	x	0.5/1	0.5/1
Cyclic data exchange	x	1	x	x	x	0	0
PROFIBUS DP interface as master							
Bus fault (bus short-circuit)	x	1	x	x	x	1	1
Bus fault (slaves have failed or cable break)	x	1	x	x	x	0.5/1	0.5/1
No error or no interface configured	x	1	x	x	x	0	0

9.1.11 Technical data

9.1.11.1 Technical data

Memory for system data

Table 9-52 Memory for system data

Data	Memory size		
	C230-2	C240	C240 PN
Diagnostic buffer	200 messages		
RAM (Random Access Memory)	20 MB	40 MB	
RAM disk (load memory)	23 MB	23 MB	
Retentive memory	13 KB	107 KB	

For information about user data, refer to chapter Properties of the user memory (Page 6865).

Table 9-53 Memory for system data (as of hardware version or FS "G")

Data	Memory size	
	C240	C240 PN
Diagnostic buffer	200 messages	
RAM (Random Access Memory)	67 MB	
RAM disk (load memory)	29 MB	
Retentive memory	107 KB	

Note

If the C240/C240 PN is used as of software version V4.4 and as of a hardware version or FS "G", you will have a RAM disk (load memory) of 29 MB and a RAM (Random Access Memory) of 67 MB.

If the C240/C240 PN as of a hardware version or FS "G" is operated as a spare part with older SW versions, the previous values set for the corresponding SW version shall apply.

System clocks

Table 9-54 System clocks

System clocks	C230-2	C240	C240 PN
Basic cycle clock (isochronous bus cycle not activated)	1.5 ms to 8 ms	0.5 to 8 ms	
DP cycle clock (isochronous bus cycle activated)	1.5 ms to 8 ms	1.0 to 8 ms	
Position control cycle clock (servo cycle clock)	≥ 1.5 ms	≥ 0.5 ms	
Interpolator cycle clock (IPO cycle clock)	≥ 1.5 ms	≥ 0.5 ms	
Send cycle of PROFINET (C240 PN)	-	-	0.5 to 4 ms

A modification can be made in the engineering system.

For system cycle clock settings, refer to the SIMOTION SCOUT online help.

Note

With the SIMOTION C, errors can occur during **Download** or during the transition from **STOP** to **RUN** if I/O bus modules and a system cycle clock > 8 ms (only with equidistant bus cycle) are used. If an error occurs, set the system cycle clock ≤ 8 ms.

DP slave connections

Table 9-55 Number of DP slave connections

Number of DP slave connections	Max. 64

PROFINET device connections

Table 9-56 Number of PROFINET device connections

Number of PROFINET device connections	Max. 64

Address areas

Table 9-57 Address areas

Input address area, total	• 2 Kbytes for C230-2 • 4 Kbytes for C240 / C240 PN
Output address area, total	• 2 Kbytes for C230-2 • 4 Kbytes for C240 / C240 PN
Process image inputs	64 bytes
Process image outputs	64 bytes

Connection values

Table 9-58 Connection values

Supply voltage	24 V DC (permissible range: 20.4 to 28.8 V)
Power consumption from 24 V	• typically 0.9 A (inputs/outputs open) • typically 1.2 A (with 4 encoders, 5 V) • typically 1.9 A (with 4 encoders, 24 V)
Power loss	15 W
Starting current	8 A
Encoder supply 5 V max. output current	1.2 A
Encoder supply 24 V max. output current	1.2 A

Dimensions and weight

Table 9-59 Dimensions and weight

Dimensions W x H x D [mm]	200 x 125 x 118
Weight [g]	Approximately 1 150

Onboard drive interface (C230-2, C240)

Analog

Table 9-60 Onboard drive interface

Setpoint signal (analog output)	
Number	4
Rated voltage range	-10.5 to 10.5 V
Output current	-3 to 3 mA
Electrically isolated	No
Load impedance	≥ 3 kOhm
Resolution	• 16-bit, including sign (with filter) • 12-bit, including sign (without filter)
Filter time • C230-2 • C240	 • With filter (C230-2-compatible) • With filter (C230-2 compatible) or without filter (can be switched in **HW Config**)
Basic error limit (at 25° C) • 0 V to 10 V • -10 V to 0 V	 • -2 to +6 % (typically +3 %) • -6 to +2 % (typically -3 %)

Relay contact controller enable	
Number	4
Switching voltage	max. 50 V
Switching current	Max. 1 A
Switching capacity	Max. 30 VA
Mechanical service life Electrical service life	Usually 10^9 switching cycles Typically $3 \cdot 10^6$ switching cycles at 24 V / 1 A

Cable length: Max. 35 m

Stepper drive (C230-2, C240)

Table 9-61 Stepper drive

5 V output signals according to RS422 standard		
Differential output voltage	V_{OD}	Min. 2 V (R_L = 100 Ω)
Output voltage "1"	V_{OH}	3.7 V (I_O = -20 mA) 4.5 V (I_O = -100 µA)
Output voltage "0"	V_{OL}	Max. 1 V (I_O = 20 mA)
Load impedance	R_L:	min. 55 Ω
Output current	I_O	max. ±60 mA
Pulse frequency	f_P	Max. 750 kHz

Cable length: Max. 50 m

- For mixed operation with analog axes, cable length: 35 m
- For asymmetrical transmission, cable length: 10 m

Actual value latch time

The following table shows the different latch times Ti of the actual encoder values depending on the position control cycle clock.

Table 9-62 Actual value latch time

Position control cycle clock [µs]	Actual value latch time Ti [µs]	
	C230-2	C240
1500	0	0
1750	0	0
2000	0	0
2250	0	0
2500	0	0
2750	0	2
3000	0	0
3250	0	64
3500	0	0
3750	0	127
4000	0	0
4250	0	189
4500	0	0
4750	0	252
5000	0	0
5250	0	314
5500	0	2
5750	0	377
6000	0	0
6250	0	439
6500	0	64
6750	0	502
7000	0	0
7250	0	564
7500	0	127
7750	0	627
8000	0	0

Onboard measuring system interface (C230-2, C240)

Table 9-63 Onboard measuring system interface

Position measuring	• Incremental • Absolute (SSI)
Signal voltages	Inputs: 5 V as per RS422
Supply voltage and current consumption per encoder	• 5 V/300 mA • 24 V/300 mA
Input frequency for incremental encoder	Max. 1 MHz
Baud rate for absolute encoder (SSI)	• 187.5 / 375 / 750 Kbits/s • 1.5 Mbit/s
Cable length for incremental encoder • 5 V encoder supply (tolerance 4.75 to 5.25 V) • 24 V encoder supply (tolerance 10 to 30 V)	• Max. 10 m at 1 MHz • Max. 25 m at 500 kHz and max. 300 mA • Max. 35 m at 500 kHz and max. 210 mA • Max. 100 m at 300 kHz and max. 300 mA
Cable length for absolute encoder (SSI) 24 V encoder supply (tolerance 10 to 30 V)	• Max. 250 m at 187.5 Kbits/s • Max. 10 m at 1.5 Mbit/s
Electrically isolated	No
Monitoring	Short-circuit in encoder supply and faulty wire

Digital inputs

Table 9-64 Digital inputs

Number of inputs	18
Supply voltage	24 V DC (permissible range: 20.4 to 28.8 V)
Input voltage	• 0 signal: -3...5 V • 1 signal: 11 to 30 V
Input current	• 0 signal: 15 mA • 1 signal: 6 to 30 mA (typically 8 mA)
Input delay (I0 to I11, B1 to B4, M1 to M2)	• 0 → 1 signal: 15 µs (typically 6 µs) • 1 → 0 signal: 150 µs (typically 40 µs)
Connection of a 2-wire encoder Permissible quiescent current	Supported 2 mA
Electrical isolation between inputs	No
Electrical isolation between inputs and logic	Yes
Insulation	500 V DC
Length of cable	Max. 30 m

Digital outputs

Table 9-65 Digital outputs

Number of outputs	8
Supply voltage	24 V DC (permissible range: V_L = 20.4 to 28.8 V)
Output voltage	1 signal: From V_L [1)] - 0.8 V to V_L [1)] V
Max. output current	1 signal: 5 mA. to 0.6 A (via supply voltage)
Total current of the outputs	• max. 4 A (at 0 to 40° C) • max. 2 A (at 40 to 55° C)
Extinguishing energy per output	400 mJ (not simultaneous)
Lamp load	5 W
Switching rate	• 100 Hz (with resistive load) • 2 Hz (with inductive load)
Short-circuit protection Max. leakage current	Yes 0 signal: 2 mA
Output delay (Q0...Q7)	• 0 → 1 signal: 500 µs (typically 150 µs) with RL = 60 Ohm • 1 → 0 signal: 500 µs (typically 150 µs) with RL = 60 Ohm
Electrical isolation between outputs Electrical isolation between outputs and logic Insulation	No Yes 500 V DC
Length of cable	Max. 30 m
1) V_L - Supply voltage of outputs	

Note

The connecting cable between the voltage source and the load current supply connector L+ and the associated reference potential M should **not** exceed a maximum length of 10 m.

READY output (RDY)

Table 9-66 Electrical parameters of RDY relay contact

RDY relay contact	
Switching voltage DC	max. 50 V
Switching current	Max. 1 A
Switching capacity	Max. 30 VA
Mechanical service life	Usually 10^9 switching cycles
Electrical service life	Typically $3 \cdot 10^6$ switching cycles at 24 V / 1 A

9.1.11.2 Real-time clock

Properties and functions of the clock

This table lists the properties and functions of the SIMOTION C clock.

Table 9-67 Properties of the SIMOTION C clock

Properties	C230-2	C240 / C240 PN
Type	Hardware clock (integrated "realtime clock")	
Factory setting upon delivery	DT#1992-01-01-00:00:00	
Backup	Permanently installed accumulator (maintenance-free goldcap)	
Accuracy • With supply voltage on 0 to 55° C • With supply voltage off 25° C -20° C to 70° C	Max. deviation per day: ±9 s ±2 s +2 s to -9 s	 -12 s to +4 s -1 s to +4 s -9 s to +4 s
Backup time	Typically 4 weeks (at 0 to 25° C)	
Charging time	1 h	

With POWER OFF

The SIMOTION C clock continues to operate after POWER OFF for the battery backup time (excluding software clock). The battery is recharged during POWER ON.

No error message is output if the backup function is defective. With POWER ON, the clock resumes at the time at which POWER OFF occurred.

When the SIMOTION C is reset to the factory setting, the clock is also reset to the "factory setting as delivered".

9.1.11.3 Transportation and storage conditions for SIMOTION C

With regard to transportation and storage conditions, the SIMOTION C surpasses the requirements specified in IEC 1131, Part 2. The following conditions apply to modules that are transported and stored in the original packaging.

Table 9-68 Transportation and storage conditions for SIMOTION C

Type of condition	Permissible range
Free fall	≤ 1 m
Temperature (transport)	From - 40° C to + 70° C
Atmospheric pressure	1,060 to 700 hPa (corresponds to an altitude of up to 3000 m)
Relative humidity (transport)	5% to 95%, without condensation

9.1.11.4 Mechanical and climatic environmental conditions for operation of the SIMOTION C

Use conditions

The SIMOTION C is designed for use in a stationary, weather-protected installation.

The SIMOTION C satisfies the operating conditions for Class 3C2 in accordance with DIN EN 60721 3-3 (operating locations with high traffic densities and in the immediate vicinity of industrial equipment with chemical emissions).

The SIMOTION C must not be used in the following locations without additional measures being taken:

- Locations with a high percentage of ionizing radiation
- Locations with severe operating conditions, e.g. due to:
  - Dust accumulation
  - Corrosive vapors or gases
- Installations requiring special monitoring, such as:
  - Elevator installations
  - Electrical installations in highly sensitive areas

An additional measure for the use of the SIMOTION C could be installation in a cabinet, for example.

Climatic environmental conditions

The SIMOTION C may be used under the following climatic environmental conditions:

Table 9-69 Climatic environmental conditions

Environmental conditions	Range of application	Comments
Temperature: Horizontal mounting: Vertical mounting	 From 0 to 55° C From 0 to 40° C	-
Relative humidity	From 5% to 95%	Without condensation, corresponds to relative humidity (RH) severity level 2 in accordance with IEC 1131-2
Atmospheric pressure	1,060 to 700 hPa	Corresponds to an altitude of mean sea level to 3,000 m
Contaminant concentration	 SO_2: < 0.5 ppm; Relative humidity <60%, no condensation H_2S: < 0.1 ppm; Relative humidity <60%, no condensation	Test: 10 ppm; 4 days 1 ppm; 4 days
Moisture condensation and ice formation	Not permitted	

Mechanical environmental conditions

The mechanical environmental conditions for the SIMOTION C are specified in the following table in terms of sinusoidal vibrations.

Table 9-70 Mechanical environmental conditions

Mechanical environmental conditions	Operation	Transport (in packaging)
Vibration tested in accordance with DIN EN 60068-2-68	10 to 58 Hz: 0.35 mm 58 to 200 Hz: 50 m/s²	5 to 9 Hz: 3.5 mm 9 to 200 Hz: 10 m/s²
Shock resistance tested in accordance with DIN EN 60068-2-27	10 g peak value, 6 ms duration 100 shocks in each of the 3 axes vertical to one another	10 g peak value, 6 ms duration 100 shocks in each of the 3 axes vertical to one another

Reduction of vibration

If the SIMOTION C is subjected to greater shocks or vibrations, you must take appropriate measures to reduce the acceleration or the amplitude.
Installation on damping material (e.g. rubber-bonded metals) is recommended.

9.1.11.5 Specifications for dielectric tests, safety class and degree of protection

Test Voltages

During the routine test, the insulation resistance is tested at the following test voltage in accordance with IEC 1131 Part 2:

Table 9-71 Test Voltages

Circuits with rated voltage U_e relative to other circuits or ground	Test voltage
$0 V < U_e \leq 50 V$	500 VDC

Safety class

Safety class I in accordance with IEC 536 (VDE 0106, Part 1), i.e. a protective-conductor terminal is required on the mounting rail!

Protection against the ingress of foreign matter and water

IP 20 degree of protection in accordance with IEC 529, i. e. protection against contact with standard probes.

Also: Protection against ingress of solid foreign bodies with diameters greater than 12.5 mm.

No special protection against ingress of water.

9.1.12 Dimension drawing, spare parts, and accessories

9.1.12.1 Dimension drawing

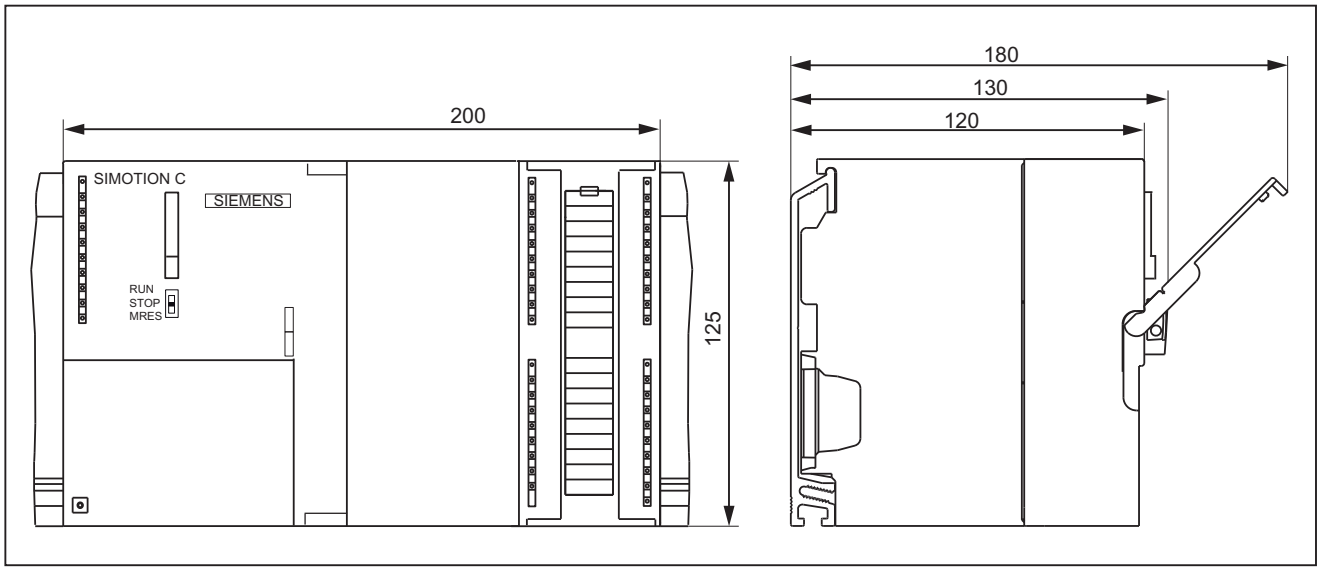


Figure 9-78 SIMOTION C dimension drawing

9.1.12.2 Spare parts and accessories

Table 9-72 Spare parts and accessories

Parts for SIMOTION C	Article number	Accesso-ries	Spare parts
Bus connector	6ES7 390-0AA00-0AA0	-	X
Connecting comb between power supply and SIMOTION C	6ES7 390-7BA00-0AA0	X	-
2 keys for C230-2 (for mode selector)	6ES7 911-0AA00-0AA0	-	X
SIMOTION C230-2 micro memory card	6AU1 700-0AA02-0AA0	X	-
Micro memory card for SIMOTION C240 / C240 PN	6AU1 720-1KA00-0AA0	X	-
Labeling plate (10 items)	6ES7 392-2XX00-0AA0	-	X
Slot number plate	6ES7 912-0AA00-0AA0	-	X
Front connector, 40-pin • Screw-type • Spring-tension type	 6ES7 392-1AM00-0AA0 6ES7 392-1BM01-0AA0	X	-
Shield connecting element	6ES7 390-5AA00-0AA0	X	-
Shield connection terminals for • 2 cables, each with 2 to 6 mm shield diameter • 1 cable with 3 to 8 mm shield diameter • 1 cable with 4 to 13 mm shield diameter	 6ES7 390-5AB00-0AA0 6ES7 390-5BA00-0AA0 6ES7 390-5CA00-0AA0	X	-

9.1.13 Standards, Certificates and Approvals

9.1.13.1 General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

Korea certification

KC	KC registration number: KCC-REM-S49-SIMOTION Note that this device complies with limit class A with regard to the emission of radio frequency interference. This device can be used in all areas except residential areas. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert).

RCM (C-Tick)

AUSTRALIA	
	This product meets the requirements of the AS/NZS CISPR 22.

9.1.13.2 Residual risks of power drive systems

Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public grids requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are used.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the controller and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example:
   - Hardware faults and/or software errors in sensors, controllers, actuators, and connection systems
   - Response times of the controller and drive
   - Operating and/or ambient conditions not within the scope of the specification
   - Condensation / conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of radio devices / cellular phones in the immediate vicinity of the controller
   - External influences/damage
2. In the event of a fault, exceptionally high temperatures, including open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the converter, e.g.:
   - Component malfunctions
   - Software errors
   - Operating and/or ambient conditions not within the scope of the specification
   - External influences/damage

   Inverters of the Open Type/IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that the contact with fire inside and outside the inverter is not possible.
3. Hazardous shock voltages caused by, for example:
   - Component malfunctions
   - Influence of electrostatic charging
   - Induction of voltages in moving motors
   - Operating and/or ambient conditions not within the scope of the specification
   - Condensation / conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc. if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

Note

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant chapters in the technical user documentation.

9.1.14 ESD guidelines

9.1.14.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

9.1.14.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

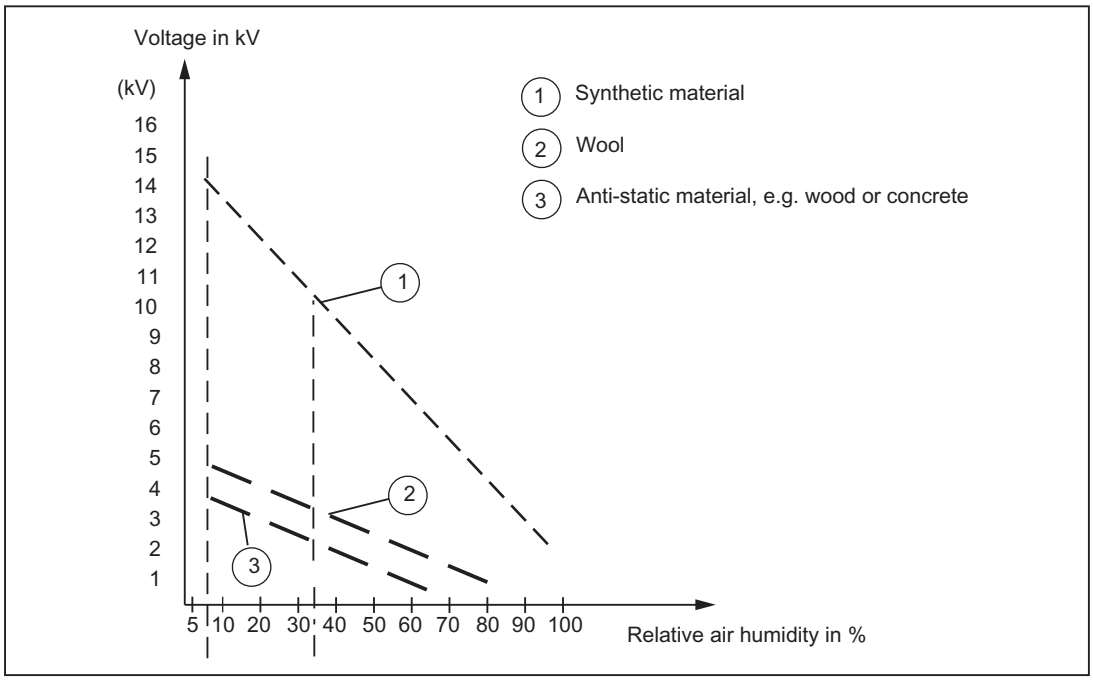


Figure 9-79 Electrostatic voltage that can accumulate on operating personnel

9.1.14.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

SIMOTION D 10

10.1 Commissioning Manual

10.1.1 SIMOTION D4x5-2

Preface

Contents of this commissioning and hardware installation Manual

This **document** is part of the **SIMOTION D documentation package**.

Scope

The SIMOTION D4x5-2 Commissioning and Hardware Installation Manual is valid for the SIMOTION D4x5-2 control units as well as for the CX32-2, CBE30-2 and TB30 supplementary system components.

A separate *SIMOTION D4x5* Commissioning and Hardware Installation Manual is available for the SIMOTION D425, SIMOTION D435 and SIMOTION D445/D445-1 devices including the CX32, CBE30 and TB30 system components.

The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 Version V5.x.

Information of configuration of the SIMOTION D Control Units in the Engineering Framework Totally Integrated Automation Portal (SCOUT in the TIA Portal), you will find in the configuration manual *SIMOTION SCOUT TIA*.

The TIA Portal requires at least SIMOTION SCOUT V4.4 and SIMOTION D4xx-2 Control Units as of firmware V4.3.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Information in this manual

The following is a description of the purpose and use of this Commissioning and Hardware Installation Manual:

- Description
  Provides information about the SIMOTION system and its integration in the automation environment.
- Installing
  Provides information on the various installation options for the device.
- Connecting
  Provides information about connecting and cabling the various devices, and about communications interfaces.
- Commissioning (hardware)
  Provides information on commissioning the device.
- Parameter assignment / addressing
  Provides information on configuring and parameterizing the various bus systems.
- Commissioning (software)
  Provides information on configuring and commissioning the system.
- Service and maintenance
  Provides information about service and maintenance procedures that must be performed on the device.
- Diagnostics
  Provides information about the available diagnostic information, how to interpret it, and its meaning.
- Appendices with factual information for reference (e.g. standards, approvals, and ESD guidelines).

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P

- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Disposal and recycling of the device

SIMOTION D is an environmentally friendly product. It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner/index.asp

Further information / FAQs

You can find further information on this manual at the following FAQ:

https://support.industry.siemens.com/cs/ww/de/view/27585482

The following information sources are also available:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling)
- The latest SIMOTION FAQs at https://support.industry.siemens.com/cs/ww/de/ps/14505/faq
- SIMOTION SCOUT online help
- For additional documentation, see the *Overview of SIMOTION documentation* (separate document)

Open source software

Third-party software - License conditions and copyright notes

Copyright notes for the third-party software contained in this product, in particular the open source software, as well as the applicable license conditions, can be found in the READ_OSS.ZIP file on the SIMOTION D CF card or in the corresponding firmware files.

Special note for resellers

The notes and the license conditions contained in the READ_OSS.ZIP file must be passed on to the purchaser in order to avoid the reseller and purchaser from violating the license conditions.

Source code availability

Some license terms of third-party software components used in this product may require us to provide you with the source code and other information for those components. You can find this information directly on or with the product (e.g. on mass storage devices, DVD). If this is not possible for technical reasons, Siemens will be happy to send you this OSS source code in exchange for reimbursement of the processing costs. Please contact the address provided at the end of this section.

Siemens AG

Digital Factory Customer Services

DI CS SD CCC TS

Gleiwitzer Str. 555

D-90475 Nuremberg, Germany

Internet (https://support.industry.siemens.com/cs/ww/en/ps)

Tel.: +49 911 895 7222

10.1.1.1 Safety instructions

Fundamental safety instructions

General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety information for SIMOTION D4x5-2

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra low voltage in accordance with EN/IEC 60950-1 may be connected at all connectors and terminals.

⚠ WARNING

Danger to life from electric shock due to insufficient safety isolation

Employing protection against direct contact using DVC A (PELV) is only permissible in areas with equipotential bonding and in dry rooms indoors.

Use other protective measures against electric shock, such as touch protection, if the specified conditions are not met.

⚠ WARNING

Danger to life from unexpected movement of machines on automatic restart

An automatic restart can be programmed for SIMOTION controllers. When the power returns, the axes start automatically.

Make sure this presents no hazard to personnel or property.

NOTICE

Damage to option boards caused by electric fields or electrostatic discharge

Option boards are ESD-sensitive components.

De-energize the SIMOTION D4x5-2 device before inserting or removing the option board. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.

Comply with the ESD rules.

NOTICE

Damage to the CompactFlash card from electrical fields or electrostatic discharge

The CompactFlash card is an ESD-sensitive component.

De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.

Comply with the ESD rules.

NOTICE

Higher operating temperature if ventilation clearances are too small

The 80 mm clearances above and below the components must be observed.

The unit protects itself from overheating by shutting down.

The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

10.1.1.2 Description

System overview

Overview

SIMOTION D is a drive-based version of SIMOTION based on the SINAMICS S120 drive family.

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S120 drive software run on shared control hardware.

SIMOTION D is available in two versions:

- SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications. The Control Unit is snapped directly on to the SINAMICS Power Module in blocksize format and has an integrated drive control for either one servo, one vector or one *V/f* axis.
- SIMOTION D4x5-2 is a Control Unit for multi-axis applications in SINAMICS S120 booksize format. The following performance versions are offered:
  - SIMOTION D425-2 (BASIC performance) Control Unit for up to 16 axes
  - SIMOTION D435-2 (STANDARD performance) Control Unit for up to 32 axes
  - SIMOTION D445-2 (HIGH performance) Control Unit for up to 64 axes
  - SIMOTION D455-2 (ULTRA-HIGH performance) Control Unit for up to 128 axes or applications with very short control cycles

The SIMOTION D4x5 2 is described in this manual. Separate manuals are available for the SIMOTION D410-2 and the D4x5/D410 predecessor modules.

Like SINAMICS S120, SIMOTION D also follows the Totally Integrated Automation (TIA) concept. TIA is characterized by integrated data management, configuration, and communication for all products and systems. Thus, an extensive toolbox of automation modules is also available for SIMOTION D.

Note

In order to cover all versions of SIMOTION D for multi-axis applications, the product will be referred to as "D4x5-2". Specific product designations will be used for information that applies only to one product version, e.g. D445-2 DP/PN.

SIMOTION D4x5-2 DP describes all PROFIBUS versions, and SIMOTION D4x5-2 DP/PN all PROFIBUS/PROFINET versions of the SIMOTION D4x5-2 Control Units.

Application

The SIMOTION D4x5-2 is ideally suited to applications with many coordinated axes with high clock-pulse rates.

Typical applications include:

- Compact multiple-axis machines
- High-performance applications with short machine cycles
- Compact machines
  - Including the complete machine control in the drive
  - With extensive connection possibilities for communication, HMI and I/O
- Distributed drive concepts
  - Applications with many axes
  - Synchronization of several SIMOTION D Control Units using distributed synchronous operation

Versions

The Control Units are available in the versions SIMOTION D425-2 (BASIC performance), SIMOTION D435-2 (STANDARD performance), SIMOTION D445-2 DP/PN (HIGH performance) and SIMOTION D455-2 DP/PN (ULTRA-HIGH performance). The versions differ in their PLC performance and in their motion control performance. The main distinguishing features are:

Table 10-1 Device versions and features

	SIMOTION D425-2	**SIMOTION D435-2**	**SIMOTION D445-2**	**SIMOTION D455-2**
Maximum number of axes	16	32	64	128
Minimum servo/interpolator cycle clock	0.5 ms	D435-2 DP: 0.5 ms D435-2 DP/PN: 0.25 ms	0.25 ms	0.25 ms 0.125 ms [1]
DRIVE-CLiQ interfaces	4	6	6	6

[1] Only with ET 200SP, SCOUT TIA and Servo_fast / IPO_fast

The Control Units feature PLC and motion control performance (open-loop control and motion control) for up to 16, 32, 64 or 128 axes, as required.

The integrated drive computing performance of the Control Units allows up to 6 servo, 6 vector or 12 *V/f* axes on each D4x5-2 Control Unit (drive control based on CU320-2, firmware version ≥ V4.x).

The drive control supports servo control (for a highly dynamic response), vector control (for maximum torque accuracy) and *V/f* control.

SIMOTION D435-2 DP/PN and D455-2 DP/PN are also available as SIPLUS version for use under extremely harsh environmental conditions, e.g. in toxic atmospheres (for details refer to technical specifications). As BasedOn products, the SIPLUS versions have the same functionality as the standard modules and are configured in the same way.

The SIMOTION D4x5-2 Control Units and their CX32-2, CBE30-2 and TB30 supplementary system components are described in the following.

Note

With the SIZER configuration tool, you can easily configure the SINAMICS S110/120 drive family including SIMOTION.

It provides you with support for selecting and dimensioning the components required for a Motion Control task.

You can also determine the possible number of axes and the resulting load with SIZER in accordance with your performance requirements.

Hardware components

As the central hardware, SIMOTION D uses the SIMOTION D4x5-2 as Control Unit consisting of the SIMOTION runtime system and the SINAMICS drive control. The Control Unit uses the SINAMICS Integrated drive with various SINAMICS S120 drive modules (Line and Motor Modules) to perform open-loop and closed-loop control of the axis grouping. A range of additional SINAMICS S120 components, such as SMx encoder systems or Terminal Modules can also be connected via DRIVE-CLiQ. With a few exceptions (e.g. no basic positioner EPOS, no Basic Operator Panel BOP20, etc.), the drive control integrated in SIMOTION D has the same control characteristics and performance features as the SINAMICS S120 CU320-2 Control Unit. The EPOS functionality is provided by the SIMOTION technology functions. The functionality of SIMOTION D can be expanded with distributed I/O via PROFIBUS or PROFINET IO.
The following figure shows a typical SIMOTION D axis grouping.

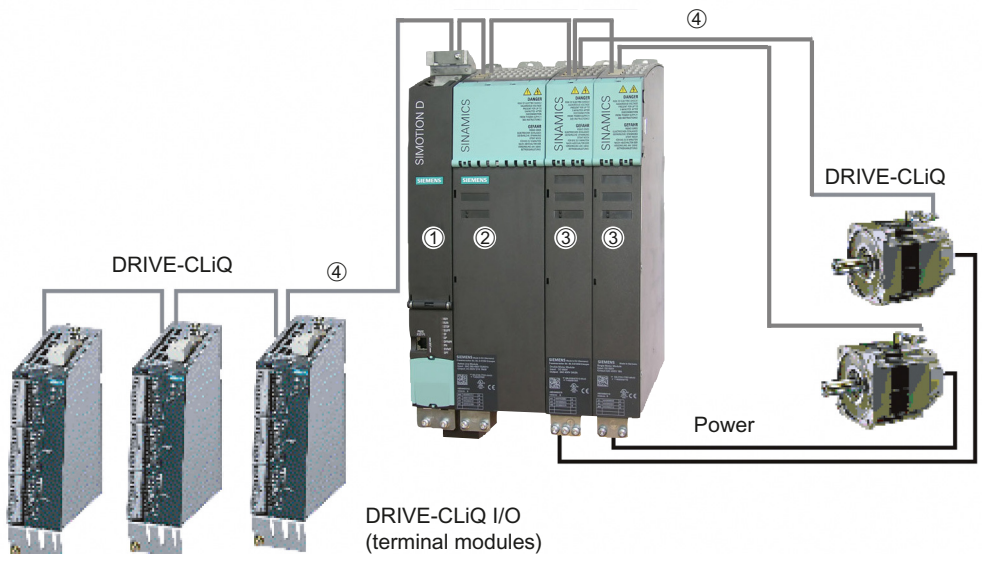


Figure 10-1 Example of an axis grouping with SIMOTION D4x5-2

A SIMOTION D axis grouping generally consists of the following elements:

- **SIMOTION D** (Control Unit) (1)
  This unit contains the programmable runtime system of SIMOTION and the drive software of SINAMICS S120. In principle, SIMOTION D is capable of controlling multiple axes/drives.
- One **SINAMICS infeed** (Line Module) (2)
  This module generates a DC link from the supply system.
- **SINAMICS power units** (Motor Modules) (3)
  These modules are used to control motors.
  It is also possible to operate SINAMICS Power Modules in blocksize format with the SINAMICS Control Unit Adapter (CUA). A separate infeed is then unnecessary.
- **DRIVE-CLiQ components** (4)
  In SINAMICS S120 / SIMOTION D, the individual components of the drive system communicate with each other via DRIVE-CLiQ. In addition to power components, it is also possible to link encoder systems and special DRIVE-CLiQ I/O devices via DRIVE-CLiQ.

Extension of the drive computing performance

The motion control performance of a SIMOTION D4x5-2 can be utilized in full by expanding the computing performance at the drive in two different ways:

- SINAMICS S/G Control Units (e.g. CU320-2, CU310-2, CU305, CU250S-2, etc.) together with further drive components can be connected via PROFIBUS or PROFINET.
- With SIMOTION D4x5-2, the CX32-2 Controller Extension can be connected via DRIVE-CLiQ. This module is extremely compact, does not require a separate CompactFlash card and can control up to 6 servo, 6 vector or 12 *V/f* axes.

Software components

The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following:

The SIMOTION runtime system with the following functions:

- Freely programmable runtime system (IEC 61131)
- Various runtime levels (tasks)
- PLC and arithmetic functionality
- Motion control functions
- Communication functions

The SINAMICS S120 drive control with the following functions:

- Closed-loop current and torque control
- Closed-loop speed control
- Closed-loop infeed

System components

Central components

SIMOTION D4x5-2 communicates with automation components via the following interfaces:

- PROFIBUS DP
- Ethernet
- PROFINET IO
- DRIVE-CLiQ (DRIVE Component Link with IQ)

SIMOTION D features a SINAMICS Integrated drive element. The communication with the SINAMICS Integrated is performed via PROFIBUS mechanisms (DP Integrated), i.e. the communication is handled, for example, via PROFIdrive telegrams.

Shorter cycle times and greater numbers of addresses for each node are achieved with the "DP Integrated" compared to the "external PROFIBUS DP".

The most important components of the system and their functions are shown below.

Table 10-2 Central components

Component	Function
SIMOTION D4x5-2 controller	... is the central motion control module. This module contains the programmable SIMOTION runtime for the SIMOTION D4x5-2 and the SINAMICS S120 drive software. You can use the integrated high-speed digital I/Os as: • User-addressable process I/Os • Homing inputs • Inputs for measuring inputs • Outputs for fast output cams The measuring sockets can output any analog signals.
System software	The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following: • SIMOTION runtime (kernel) • Drive software of SINAMICS S120 - implements all drive functions
Power supply	... provides the electronics power supply for SIMOTION D, e.g. via the SITOP power supply.

PROFIBUS DP

The Control Unit can communicate with the following components via the PROFIBUS DP interfaces:

Table 10-3 Components on PROFIBUS DP

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs, and tests with the "SIMOTION SCOUT" engineering system (ES)
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.

Component	Function
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET connection and live module replacement.
SIMATIC ET 200eco	I/O system with IP65/67 degree of protection for machine-related applications with no cabinet, and with flexible and fast ECOFAST or M12 connection methods.
Other PROFIBUS I/O	
Gateways	• DP/AS-Interface link 20E and DP/AS-Interface link Advanced for the PROFIBUS DP gateway to AS-Interface • DP/DP coupler for connecting two PROFIBUS DP networks
Drive interfaces	• ADI4 (Analog Drive Interface for 4 axes) for connection of drives with analog ± 10 V setpoint interface or for external encoders • IM174 (Interface Module for 4 axes) for connection of drives with analog ± 10 V setpoint interface, for external sensors, or for connection of stepper drives with pulse-direction interface
Drive units with PROFIBUS DP interface (e.g. SINAMICS S120)	... convert speed setpoints into signals for controlling the motors and supply the power required to operate the motors. Also can be operated as an isochronous slave on PROFIBUS DP.
Teleservice adapter	Remote diagnostics

Ethernet

The Control Unit can communicate with the following components via the Ethernet interfaces or be embedded in an automation environment:

Table 10-4 Components on the Ethernet

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs, and tests with the "SIMOTION SCOUT" engineering system (ES)
Master computer	... communicates with other devices via UDP, TCP/IP
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.

PROFINET IO

The D4x5-2 DP/PN can communicate with the following components via the onboard PROFINET IO interface or via the Ethernet communication board (CBE30-2).

Table 10-5 Components on the PROFINET IO

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs and tests with the "SIMOTION SCOUT" engineering system (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Master computer	... communicates with other devices via UDP, TCP/IP
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data. With the time-based I/Os, signals can be recorded or output to the precise µs.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety. With the time-based I/Os, signals can be recorded or output to the precise µs.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET IO connection and live module replacement.
SIMATIC ET 200eco PN	Compact block I/O with IP65/66/67 degree of protection for machine-related applications with no cabinet, and with M12 connection method. Very rugged and resistant encapsulated metal enclosure.
Other PROFINET IO I/O devices	
Drive units with PROFINET IO interface	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors.
Gateways	• IE/AS-Interface link PN IO for the PROFINET IO gateway to AS-Interface • PN/PN coupler for connecting two PROFINET IO networks

DRIVE-CLiQ

The DRIVE-CLiQ interfaces permit a fast connection to the SINAMICS drive components.

DRIVE-CLiQ offers the following advantages within the DRIVE-CLiQ topology rules:

- Expandability of components
- Automatic detection of components by the Control Unit
- Standardized interfaces to all components
- Uniform diagnostics down to the components
- Uniform service through to the components
- Simple mechanical handling

The controller can communicate with the following components via DRIVE-CLiQ:

Table 10-6 Components connected to DRIVE-CLiQ

Component	Function
Control Unit (SINAMICS S110/ S120)	Central control module in which the open- and closed-loop control functions for the drive are implemented.
Line Module (SINAMICS S120)	... generates a DC link from the supply system.
Motor Module (SINAMICS S120)	... controls motors (DC/AC inverters, booksize).
Power Module (SINAMICS S110/ S120)	... controls motors (AC/AC converters, blocksize).
CX32-2 Controller Extension	... enables additional axes to be connected for SIMOTION D4x5-2.
CUA31/CUA32 Control Unit Adapter	... enables a Power Module in blocksize format to be connected to a booksize D4x5-2, CX32-2 or CU320-2 Control Unit.
TM15, TM17 High Feature Terminal Modules	The TM15 and TM17 High Feature Terminal Modules are used to implement inputs of measuring inputs and outputs of output cams. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times.
TM31 Terminal Module	... enables a terminal expansion via DRIVE-CLiQ (additional analog and digital I/Os).
TM41 Terminal Module	... enables a terminal expansion (analog and digital I/Os) and encoder emulation.
TM54F Terminal Module	... enables terminal expansion (fail-safe digital I/Os) for controlling the safe motion monitoring functions of the integrated drives.
TM120 Terminal Module	Four temperature sensors (KTY84-130 or PTC) can be evaluated via the TM120 Terminal Module. The temperature sensor inputs are safely electrically separated from the evaluation electronics in the TM120 Temperature Module and are suitable for evaluating the temperature of special motors, e.g. 1FN linear motors and 1FW6 built-in torque motors.
TM150 Terminal Module	The TM150 Terminal Module can be used to evaluate temperature sensors (KTY, PT100, PT1000, PTC, and bimetal normally closed contact). This means, for example, that other temperatures from the process can be measured in addition to the motor temperature. Temperature sensors can be evaluated using a 2, 3 or 4-wire system. Twelve temperature sensors can be evaluated with 2-wire evaluation and six temperature sensors with 3 and 4-wire evaluation.

Component	Function
SMx Sensor Modules	... enable acquisition of encoder data from connected motors via DRIVE-CLiQ.
Motors with DRIVE-CLiQ interface	... enable simplified commissioning and diagnostics, as the motor and encoder type are identified automatically.
DMC20/DME20 DRIVE-CLiQ hub	... enables the number of DRIVE-CLiQ interfaces to be increased and the creation of a point-to-point topology.

Power Modules via CUA31/32

The following Power Modules are supported:

- PM340
- PM240-2 (as of SIMOTION V4.4 / SINAMICS V4.7)

The mixed operation of a PM240-2 with booksize modules and/or PM340 blocksize modules on a CU320-2/D4x5-2/CX32-2 is possible as of SINAMICS V4.7 HF12 or SIMOTION V4.4 HF6 (SINAMICS Integrated V4.7 HF12).

Note

You will find detailed information on components of the SINAMICS S110/S120 product family in the SINAMICS S110/S120 manuals.

It is possible that older DRIVE-CLiQ components can no longer be used with SIMOTION D4x5-2/CX32-2. You will find detailed information on this topic in the SIMOTION D4x5-2 Commissioning and Hardware Installation Manual at "Migration of D4x5 to D4x5-2" in Section "Permissible combinations".

Optional components

The functionality of the D4x5-2 Control Unit can be expanded with the following components:

Table 10-7 Optional components

Component	Function	D4x5-2 DP	D4x5-2 DP/PN
CBE30-2 Ethernet communication board	Communication via PROFINET IO with IRT and PROFINET IO with RT	No	Yes
TB30 Terminal Board	Terminal expansion, i.e. additional analog and digital I/Os	Yes	Yes

The components are plugged into the option slot of the Control Unit.

I/O integration

Note

Note that not all modules in the ET 200 I/O family are approved for SIMOTION. Moreover, system-related functional differences can come into play when these I/Os or I/O systems are used on SIMOTION vs. on SIMATIC. For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

A detailed, regularly updated list of the I/O modules approved for use with SIMOTION, as well as notes on their use, can be found on the Internet at: (https://support.industry.siemens.com/cs/ww/en/view/11886029)

In addition to the I/O modules enabled for SIMOTION, in principle all certified standard PROFIBUS slaves (DP-V0/DP-V1/DP-V2) and PROFINET IO devices with RT and IRT real-time classes may be connected to SIMOTION D4x5-2. These modules are integrated using the GSD file (PROFIBUS) or GSDML file (PROFINET) provided by the relevant device manufacturer.

Note

Please note that in individual cases further boundary conditions must be fulfilled in order to integrate a standard slave/standard device into SIMOTION. Thus, a few modules require "driver blocks" , e.g. in the form of function blocks, that permit (or simplify) integration.

For modules released with SIMOTION (e.g. SIMATIC S7-300 module FM 350-1, etc.), these driver blocks are part of the SIMOTION SCOUT engineering system command library.

Commissioning software

Requirement

To create and edit projects on your PG/PC, you need the SIMOTION SCOUT commissioning and configuration tool. For information on how to install SIMOTION SCOUT, see the *SIMOTION SCOUT* Configuration Manual.

Note

SIMOTION SCOUT contains the functionality of STARTER and SIMATIC S7-Technology.

Simultaneous operation of SIMOTION SCOUT, STARTER, and SIMATIC S7-Technology as a single installation on one PC/PG is not possible.

Integrated STARTER

You can insert a standalone drive (e.g. SINAMICS S120) with the "Insert single drive unit" element in the project navigator. It is commissioned using wizards in the working area of the workbench that contains the STARTER functionality.

SINAMICS Support Package (SSP)

The following SSPs are relevant for SIMOTION SCOUT:

- SSP "SINAMICS" for single drive units (e.g. CU3xx)
- "SIMOTION SINAMICS Integrated" SSP for the SINAMICS drives integrated into SIMOTION D.

You will find detailed information on the SSPs in the readme files and in the software compatibility list at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Upgrading SIMOTION D4x5-2 projects and hardware

Projects that you have created for one SIMOTION D4x5-2 firmware version can also be converted for other firmware versions. You can also change the version of the SIMOTION D4x5-2. For example, a D445-2 DP/PN can be converted to a D455-2 DP/PN (and vice versa, to the extent that the performance and quantities allow this). See, for example, Section Adapting a project (Upgrading the project / Replacing the SIMOTION controller) (Page 7221).

SIMOTION IT web server

The SIMOTION D4x5-2 features an integrated Web server.

The web server supports the display of diagnostics and system data in standard Internet browsers, even in the absence of an engineering system, and the carrying out of project/ firmware updates.

As of firmware version V4.2, it is no longer necessary to purchase licenses for the web server (IT DIAG license) and for SIMOTION IT OPC XML-DA.

Additional references

You will find detailed information on working with projects in the *SIMOTION SCOUT* Configuration Manual.

You will find detailed information on SIMOTION IT web server in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

10.1.1.3 Installing

Installation notes

Open components

These modules are open components. This means they may only be installed in housings, cabinets, or in electrical equipment rooms that can only be entered or accessed with a key or tool. Housings, cabinets, or electrical equipment rooms may only be accessed by trained or authorized personnel. An external fire-protection housing is required.

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

Permitted mounting positions

The following mounting positions are permitted:

- vertical installation
  (preferred standard mounting position)
  For details on this, see
  - Section Installing the SIMOTION D4x5-2 (Page 6923)
  - Section Installing the CX32-2 (Page 6932)
- lying on back
  (e.g. for applications in which the installation situation makes a low installation height necessary)
  For details, see Section Alternative mounting position (Page 6939).

Installing the SIMOTION D4x5-2

Installing the SIMOTION D4x5-2

Requirement

The Control Unit is installed in a control cabinet along with the SINAMICS components.

The following requirements must be met for installing a Control Unit:

- The control cabinet has been installed and wired.
- The SINAMICS components should already have been installed and wired.
- Components and tools are available.

Note

The components must be protected against conductive contamination, e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12.

If conductive contamination can be excluded at the installation site, a lower degree of cabinet protection may be permitted.

Designs

A D4x5-2 Control Unit is designed similar to a SINAMICS S120 booksize unit. It is mounted directly on the wall of the control cabinet, whereby a distinction is made between the following designs.

- Mounting the SIMOTION D4x5-2 with spacers
- Mounting the SIMOTION D425-2 and D435-2 without spacers
- Mounting the SIMOTION D445-2 and D455-2 without spacers (external air cooling)

The D425-2 and D435-2 Control Units do not have any cooling fins on the rear side of the module so that the spacers can be removed if required.

As the D445-2 and D455-2 Control Units have cooling fins, the spacers can be removed when external air cooling is used.

With external air cooling, the cooling fins are outside of the control cabinet. A seal is required so that the Control Unit can be hermetically mounted in the rear cabinet panel.

The fan/battery module can be mounted or replaced without having to disassemble the Control Unit. As the Control Unit is similar to the SINAMICS S120 family with regard to installation, please also refer to the information in the SINAMICS manuals.

The same installation instructions apply for the SIMOTION D4x5-2/CX32-2 Control Units as for the SINAMICS S120 CU320-2 Control Units with regard to EMC.
For further information, see the SINAMICS manuals.

Mounting and installation aids

The Control Unit is designed for mounting in a control cabinet (IP20 degree of protection). All versions of SIMOTION D4x5-2 are supplied with pre-assembled spacers for installation on the wall of the control cabinet. The spacers can be removed if necessary.

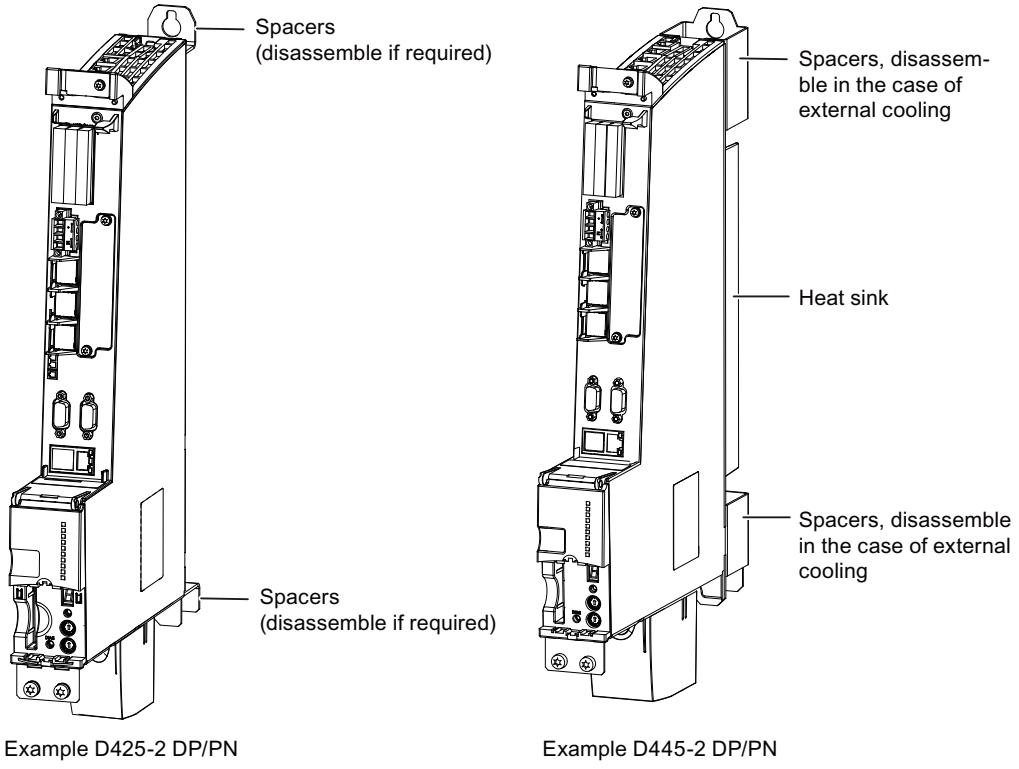


Figure 10-2 Mounting aids on the D425-2 DP/PN and D445-2 DP/PN

See also

Mounting the SIMOTION D4x5-2 with spacers (Page 6925)

Mounting the SIMOTION D4x5-2 with spacers

By using spacers, you can attach the control unit to a bare, metallic rear wall of a control cabinet with good electrical conductivity with two M6 screws.

Purpose of this type of installation

This type of mounting is needed if:

- The heat is to be dissipate within the control cabinet
- The mounting depth of the SINAMICS S120 booksize grouping is to be achieved

The spacers are included in the scope of delivery of a control unit and are preassembled.

Procedure

1. Mount the spacers on the control unit, if they have previously been removed.
2. Fasten the Control Unit to the control cabinet wall with two M6 screws (tightening torque 6 Nm or 53.1 lbf in).

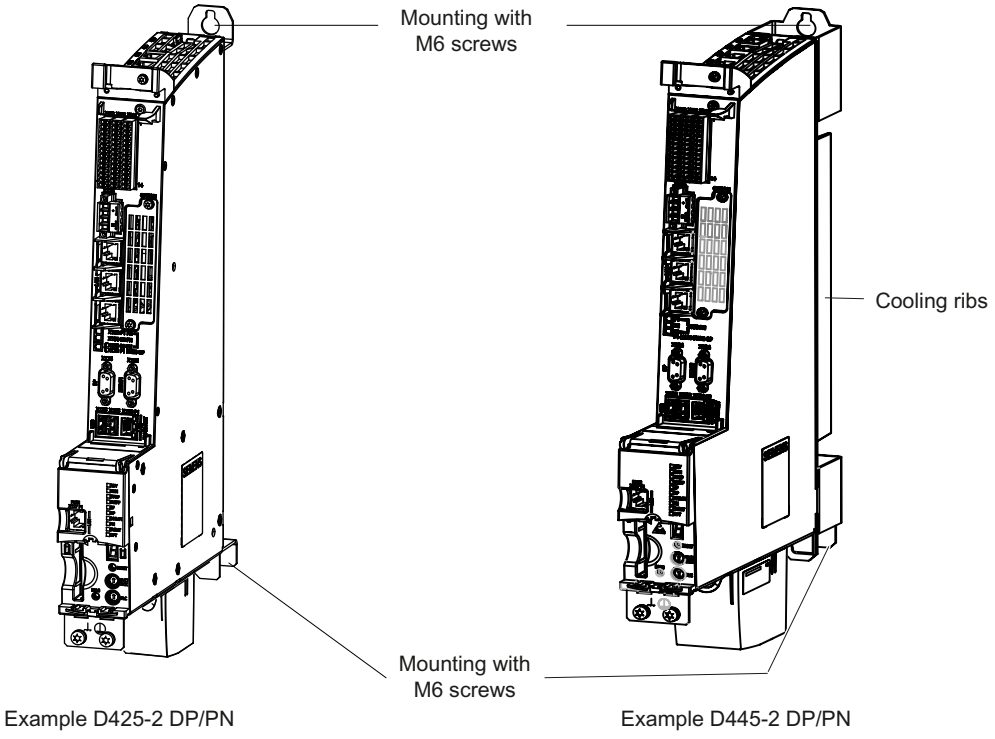


Figure 10-3 Mounting of the D425-2 DP/PN and D445-2 DP/PN with spacers

Result

The control unit is arranged flush with the grouping on the wall of the control cabinet.

Mounting the SIMOTION D425-2 and D435-2 without spacers

The control units are supplied with pre-assembled spacers. For mounting without spacers, proceed as follows:

1. Remove the upper and lower spacers (screws: M3, Torx T10)
2. There is a metal clip under the upper spacer; when shipped, the clip is pushed in and secured with three screws (M3, Torx T10) fixed. Loosen the screws and push the clip up until the upper hole extends beyond the housing.

3. Tighten the 3 screws on the clip again (tightening torque 0.8 Nm or 7.1 lbf in).
4. Mount the top and bottom of the control unit directly on the cabinet wall with two M6 screws (tightening torque 6 Nm or 53.1 lbf in).

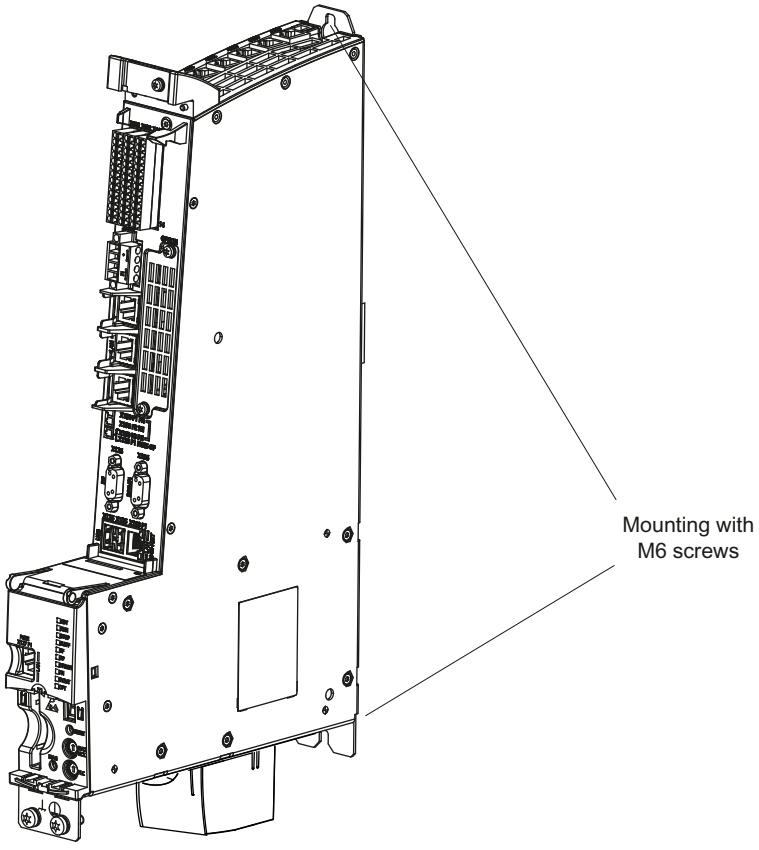


Figure 10-4 Mounting the control unit without spacers

Mounting the SIMOTION D445-2 and D455-2 without spacers (external air cooling)

Purpose of this type of installation

If you want to externally cool a Control Unit with cooling fins (D445-2 DP/PN and D455-2 DP/PN), it can be mounted directly on the rear wall of the control cabinet without spacers.

Requirements

- The bushing for the cooling fins (external heat sink) has been fitted in the rear wall of the control cabinet.
- Please ensure the area around the seal is both smooth and clean (② see "Panel cutout" figure).
- Seal available (for the article number, see the *SIMOTION D4x5-2* Manual, Section *Spare parts and accessories*)

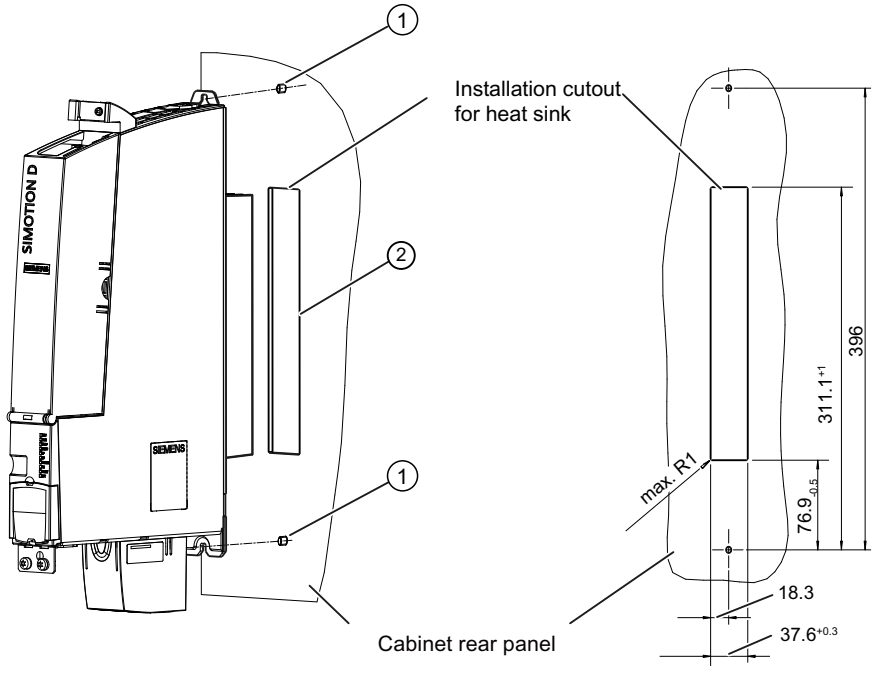


Figure 10-5 Panel cutout

Procedure

1. Remove the spacers (screws: M3, Torx T10).
2. Fit the seal around the cooling fins of the Control Unit.

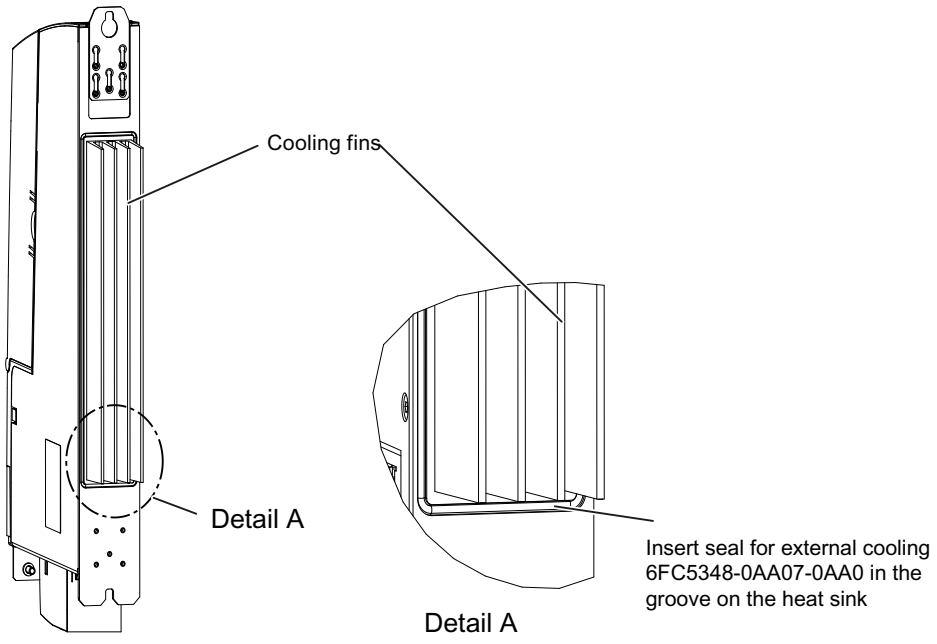


Figure 10-6 Mounting the seal

3. Loosen the three M3 screws (screws: M3, Torx T10) on the upper clip and push the clip up until the upper hole protrudes beyond the housing.
4. Tighten the 3 screws on the clip again (tightening torque 0.8 Nm or 7.1 lbf in).
5. Push the Control Unit with the heat sink through the panel cutout.
6. Fasten the Control Unit directly on the rear wall of the control cabinet with an M6 screw at the top and bottom, tightening torque: 6 Nm or 53.1 lbf in (see ① in the "Panel cutout" figure).

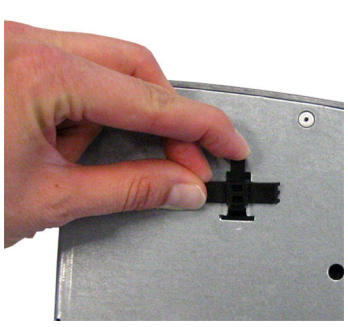
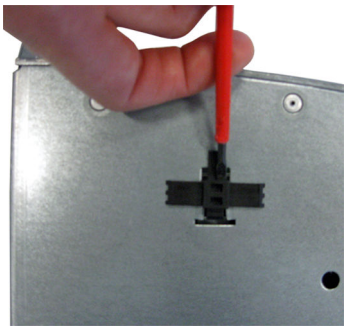
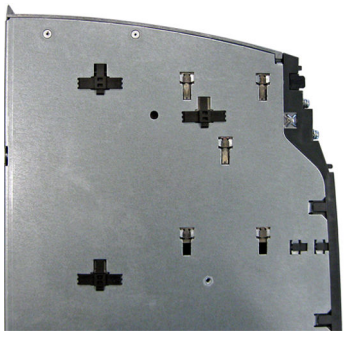
Fixing elements

The Controller Extension CX32-2 and the Control Unit CU320-2 can be mounted on the side wall of a SINAMICS S120 Line Module. The mounting elements required for this are included with the SINAMICS S120 Line Module and can be plugged if necessary.

- CU320-2: 3 mounting elements required
- CX32-2: 5 mounting elements required

Installation of the mounting elements

Table 10-8 Mounting the support brackets onto a Line Module in the booksize format

		
Place the mount in the installation opening provided.	Use a suitable tool (such as a screwdriver) to push the mount in as far as it will go.	Installed mounts (3) (booksize compact format).

Removing the mounting elements

It is not possible to mount the D4x5-2 Control Units with lateral mounting elements.

If the mounting elements are already plugged in on the Line Module, we recommend removing these for flush mounting with the D4x5-2 Control Units.

Installing supplementary system components

Installing the TB30

Requirement

Note

The TB30 terminal board may only be inserted and removed when the D4x5-2 control unit and the terminal board are de-energized.

Procedure

The TB30 is installed in the option slot of the control unit.

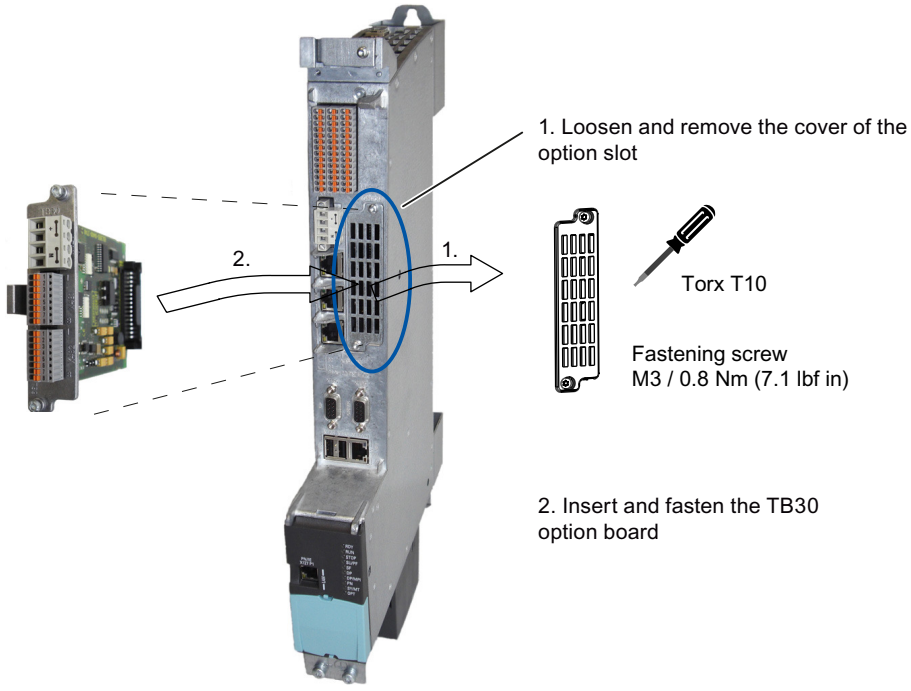


Figure 10-7 Installation of the TB30 terminal board using the D445-2 DP/PN as an example

Installing the CBE30-2

A second PROFINET interface can be implemented for the SIMOTION D4x5-2 DP/PN with the CBE30-2 Ethernet communication board.

The CBE30-2 cannot be used with the SIMOTION D4x5-2 DP.

Requirement

Note

The CBE30-2 Ethernet communication board may only be inserted and removed when the D4x5-2 DP/PN control unit and the CBE30-2 are de-energized.

Procedure

A CBE30-2 is installed in the option slot of the control unit.

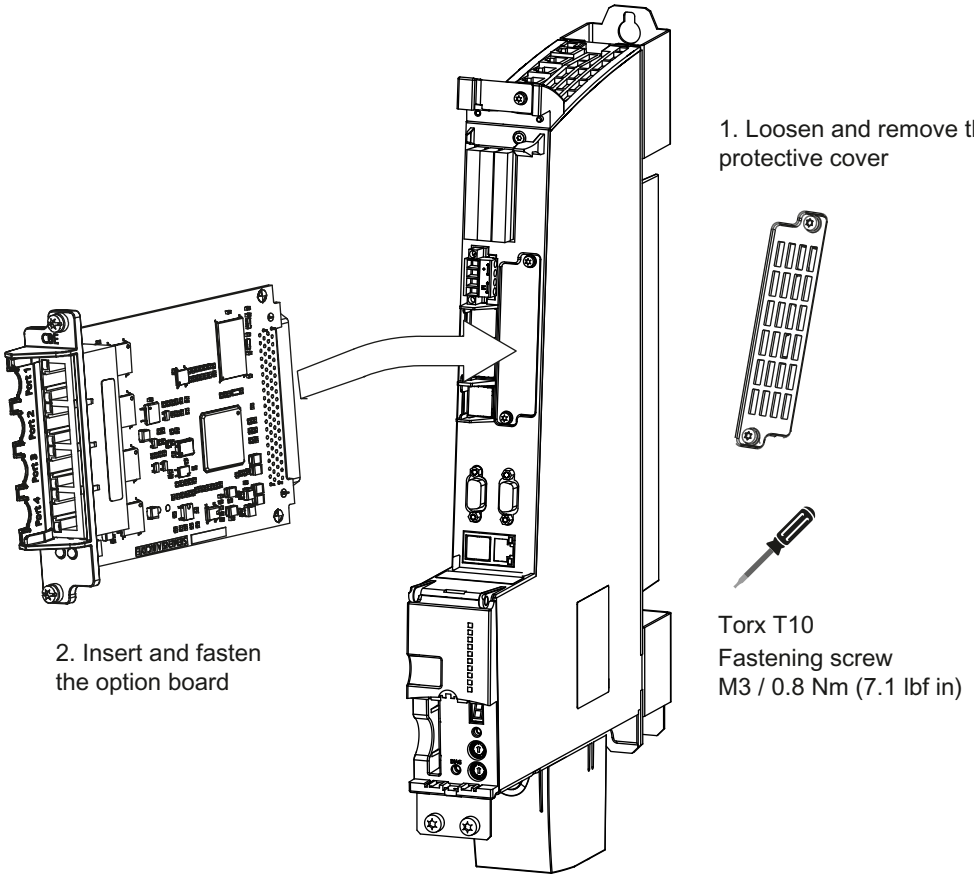


Figure 10-8 Installing the CBE30-2

Installing the CX32-2

Overview

The CX32-2 can be installed in several ways.

Mounting the CX32-2 directly on a Line Module in booksize format

The CX32-2 can be mounted on the side wall of a SINAMICS S120 Line Module. The mounting fixtures required for this are supplied with the SINAMICS S120 Line Module.

For more information on mounting the mounting elements, see also Section Fixing elements (Page 6929).

The SINAMICS S120 Line Module has five mounting fixtures on the left side. To mount the control unit, proceed as follows (the steps are shown in the figure):

1. Attach the CX32-2 to the left side of the SINAMICS S120 Line Module. Position the CX32-2 a few millimeters higher than the Line Module. The mounting fixtures fit exactly in the five cutouts on the module.
2. Push the two units together.
3. Press down on the module until the unit engages and is securely connected to the SINAMICS S120 Line Module.
   The module is now flush with the Line Module at the top and front.

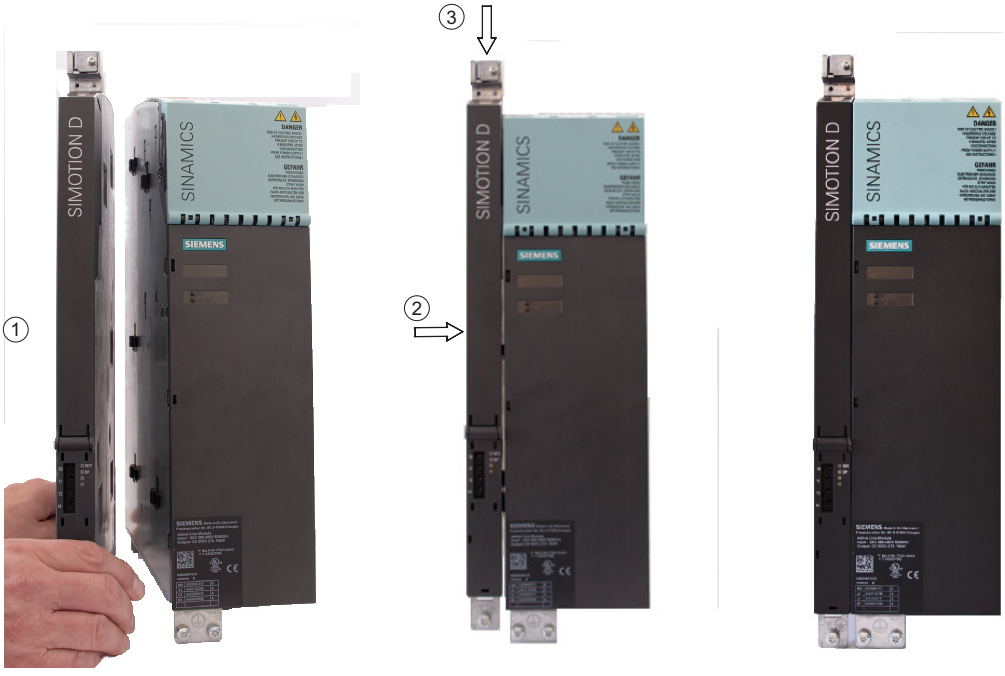


Figure 10-9 Steps in mounting the CX32-2 on a booksize Line Module

Mounting the CX32-2 on another CX32-2

A description with photos of opening the cover can be found in Section Removing the cover of the CX32-2 (Page 6938).

Table 10-9 Mounting a CX32-2 on another CX32-2

		
Opening the covers.	Unscrewing the screw (M3, Torx T10)	The bracket must be moved so that the pin is in the opening of the bracket. Tighten the screw (tightening torque 0.8 Nm or 7.1 lbf in).
		
Mounted bracket.	Closing the cover.	

Mounting the CX32-2 next to D4x5-2

If the CX32-2 is mounted next to a D4x5-2, the two modules can be mechanically connected with the bracket.

1. Loosen the screw (M3, Torx T10).
2. Disassemble the bracket at its original position.
3. Mount the bracket at the shield clamp, as shown in the following figure (tightening torque 0.8 Nm or 7.1 lbf in).

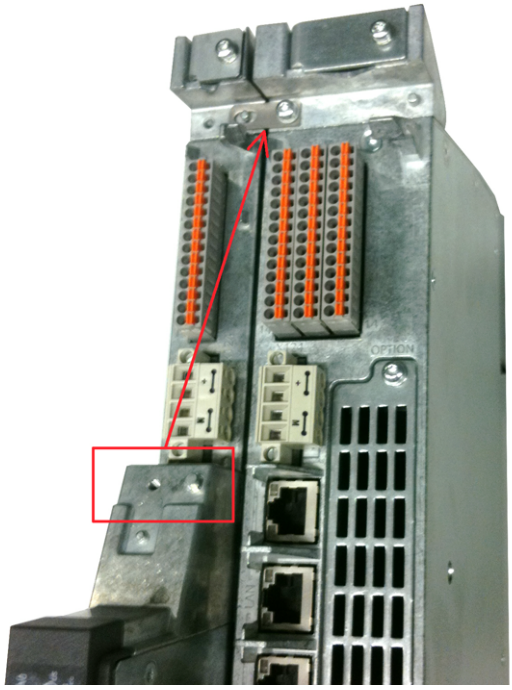
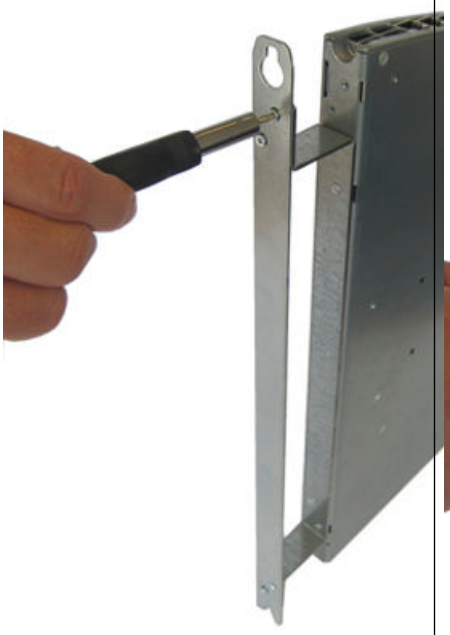


Figure 10-10 Mechanical connection CX32-2 with D4x5-2

Mounting the CX32-2 directly on a mounting surface

A spacer is pre-assembled on the CX32-2. To mount the CX32-2 directly on a mounting surface, you must first disassemble the spacer as follows:

Table 10-10 Disassembling the CX32-2 spacer

		
Loosen the screws on the mounting rail (M3, Torx T10) and remove the rail.	Loosen the screws of the spacer (M3, Torx T10) and remove the spacer.	Fasten the mounting rail to the CX32-2 again without spacer (tightening torque 0.8 Nm or 7.1 lbf in).

The CX32-2 can now be mounted directly on a mounting surface, e.g. a wall of the control cabinet. Fasten it with 2 M6 screws / tightening torque 6 Nm or 53.1 lbf in.

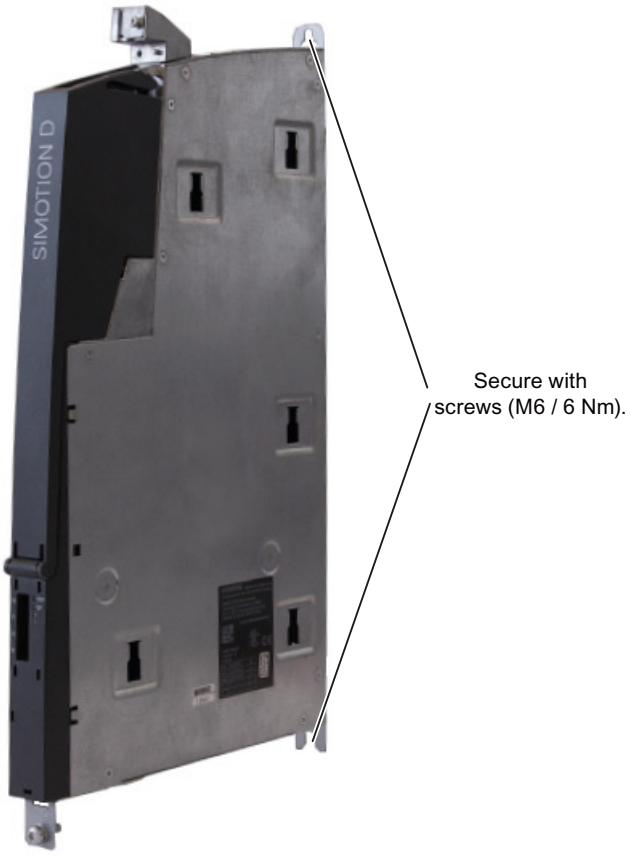


Figure 10-11 Mounting the CX32-2 on a mounting surface

CX32-2 with spacers on a mounting surface

To achieve the correct mounting depth for a booksize line-up with internal air cooling, a spacer is pre-assembled on the CX32-2. The spacer must then be fixed to the mounting surface.

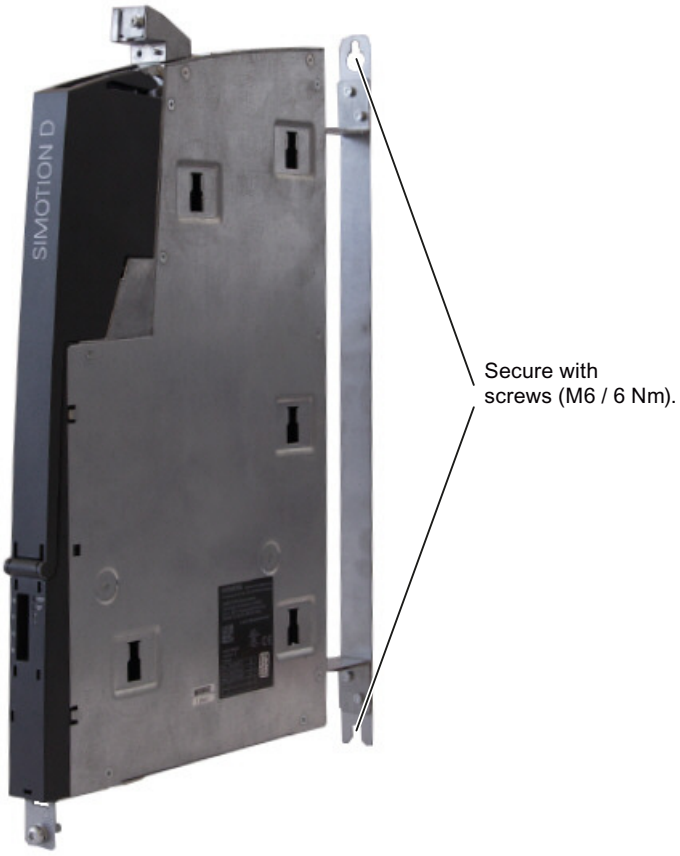


Figure 10-12 Mounting a CX32-2 with spacers

Removing the cover of the CX32-2

Actions to remove the cover of the CX32-2

Several steps are required to remove the cover:

- Release
- Swing open through approx. 45°
- Remove

Note that the cover can only be removed when it has been swung open through approx. 45°.

See also

Opening the front cover (Page 6949)

Alternative mounting position

Fundamentals

In applications in which a low overall height has to be achieved because of the mounting situation, the SIMOTION D4x5-2 control units together with the SIMOTION CX32-2 can be installed horizontally.

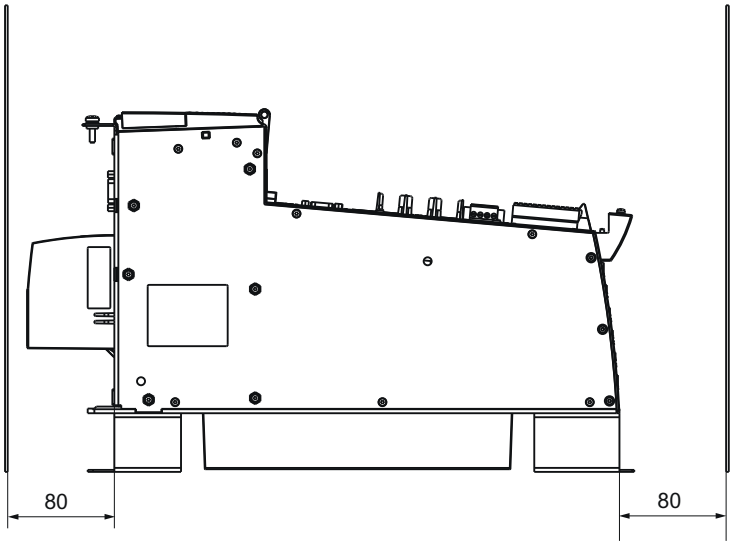


Figure 10-13 Installation of the SIMOTION D4x5-2 in a horizontal position

Requirements

Supported Control Units

The "horizontal" installation type has been released exclusively for the following Control Units:

Control Unit	Article number
SIMOTION D425-2 DP	6AU1425-2AA00-0AA0
SIMOTION D425-2 DP/PN	6AU1425-2AD00-0AA0
SIMOTION D435-2 DP	6AU1435-2AA00-0AA0
SIMOTION D435-2 DP/PN	6AU1435-2AD00-0AA0
SIMOTION D445-2 DP/PN	6AU1445-2AD00-0AA1 6AU1445-2AD00-0AA0
SIMOTION D455-2 DP/PN	6AU1455-2AD00-0AA0
SIMOTION CX32-2	6AU1432-2AA00-0AA0

Note

Mounting positions other than those described in this document are not permissible!

To achieve a low overall height with SINAMICS S120 power units, use SINAMICS S120 booksize compact power units or Power Modules in blocksize format connected via the CUA31/CUA32 Control Unit Adapter.

Heat dissipation

If SIMOTION D4x5-2 Control Units are mounted horizontally, a double fan / battery module is required as in the standard mounting position.

The double fan / battery module is always contained in the scope of delivery of the Control Unit for the SIMOTION D4x5-2.

⚠ CAUTION

Danger of injury or damage to property due to high surface temperature

SIMOTION D4x5-2 Control Units can have a high surface temperature particularly on the aluminum front and the top of the module especially with the "horizontal" installation type.

Perform a module and cable replacement only when the module has cooled down.

The ventilation spaces of 80 mm must be complied with, see Fig. "Installation of the SIMOTION D4x5-2 in a horizontal position". The control cabinet cooling concept must also ensure that the maximum air inlet temperature and the ambient temperature do not exceed 55° C.

Table 10-11 Requirements for the heat dissipation

Requirement	SIMOTION D4x5-2	SIMOTION CX32-2
Double fan/battery module	Always required	-
Clearances	80 mm (measured from the top or bottom edge of the module)	80 mm (measured from the top or bottom edge of the module)
Max. ambient temperature	55° C	55° C
Max. air inlet temperature	55° C	55° C

Mounting the D4x5-2 / CX32-2

The D4x5-2 / CX32-2 are fixed in the same way as in the standard "vertical installation" mounting position.

Requirements

If CX32-2 controller extensions are used, then to ensure mechanical stability

- the CX32-2 controller extensions and
- the CX32-2 controller extension with SIMOTION D4x5-2 control unit

must be connected to each other via a bracket.

Procedure

Like the standard "vertical installation on the control cabinet wall" mounting position, each module must be attached to a mechanically stable base via two M6 screws.

For the mechanical connection between two CX32-2, the screw (Torx 10) on the connection bracket must be loosened and the bracket turned so that the pin is in the opening of the bracket. The screw must then be tightened again with a torque of 0.8 Nm or 7.1 lbf in.

For the mechanical connection between the CX32-2 and the D4x5-2, the bracket must be disassembled from its original position and attached to the shield terminal.

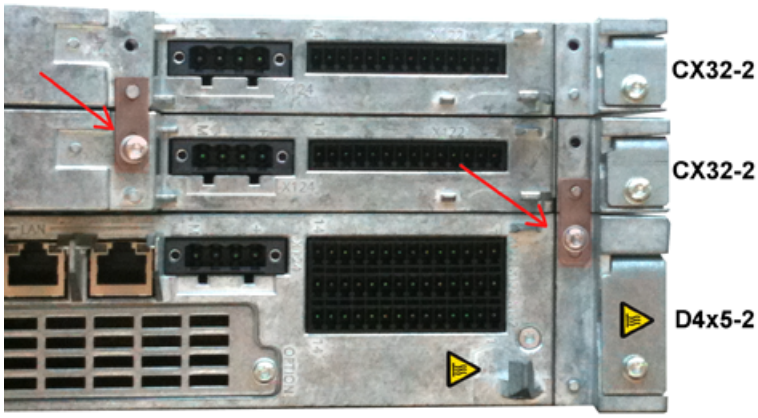


Figure 10-14 Mechanical connection of the CX32-2

Mounting without spacers

Like the standard "vertical installation" mounting position, mounting without spacers is also possible for the "horizontal" mounting position.

How spacers are removed is described in Section Installing the SIMOTION D4x5-2 (Page 6923).

Technical data / approvals / certifications

The same technical data, approvals and certifications apply for the "horizontal" mounting position as for the standard "vertical installation" mounting position.

The technical data can be found in the *SIMOTION D4x5-2* Manual.

10.1.1.4 Connecting

General Overview

Overview

The SIMOTION D4x5-2 has a number of interfaces that can be used for connecting the power supply and for communication with the other components of the system. To make these connections, the front cover of the SIMOTION D4x5-2 must be opened.

- The different SINAMICS components are interconnected via DRIVE-CLiQ.
- Actuators and sensors can be connected to the digital inputs/outputs.
- For communication purposes, the SIMOTION D4x5-2 can be connected to PROFIBUS DP, PROFINET IO with IRT/RT, MPI and Ethernet.

Overview of connections

The following overview shows examples of the various interfaces and their connection possibilities.

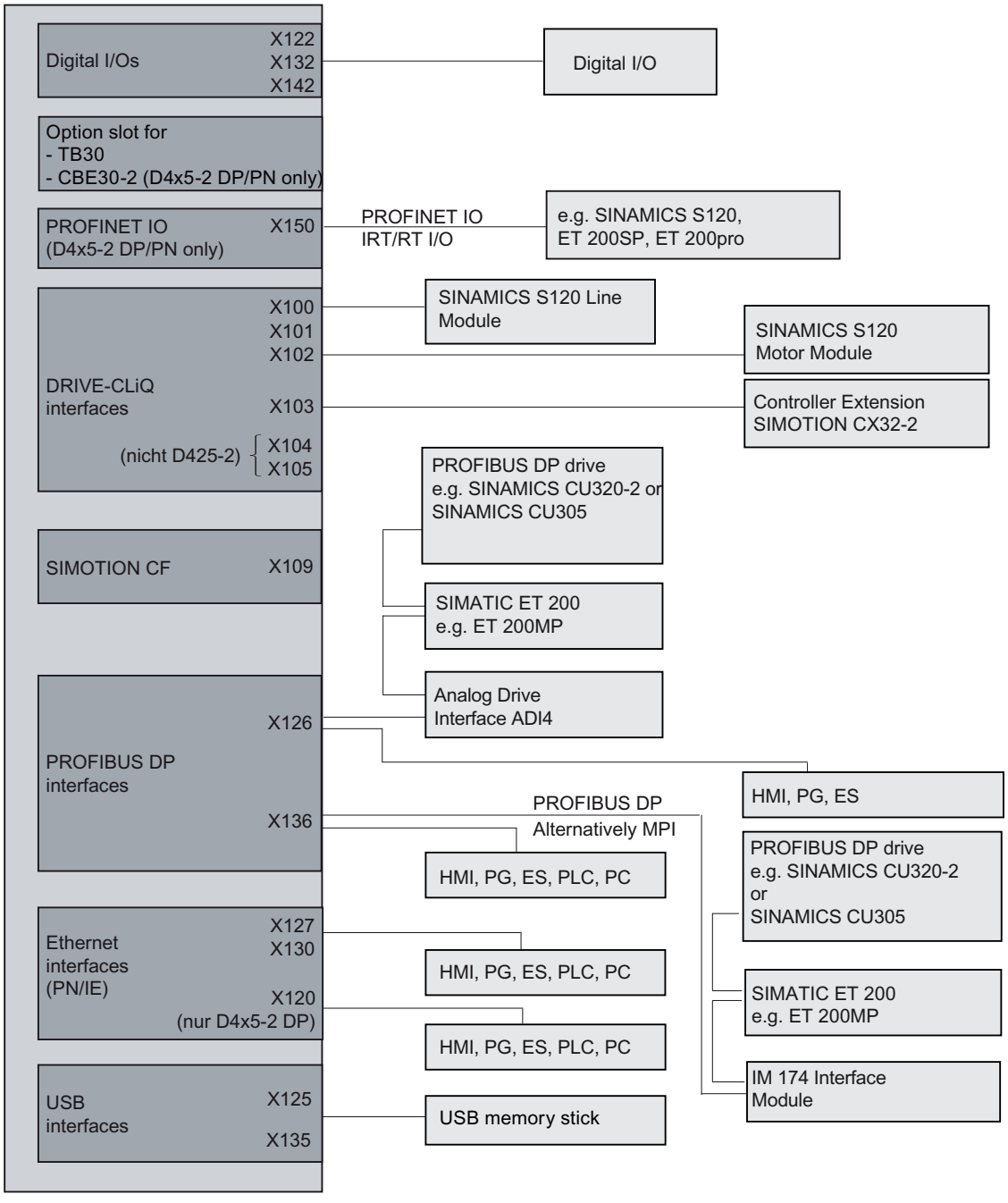


Figure 10-15 SIMOTION D4x5-2 DP and D4x5-2 DP/PN connection options

Electromagnetic compatibility

General

Electromagnetic compatibility (EMC) means that the devices function satisfactorily, without interfering with other devices and without being disrupted by other devices. The EMC requirements for "Variable-speed drive systems" (PDS, Power Drive System) are described in product standard IEC/EN 61800-3. A variable-speed drive system consists of a Control Unit, Line Module and Motor Module - as well as the associated electric motors, encoders and connecting cables. The driven machine is not part of the drive system.

Note

PDS as component of machines or systems

For the integration of PDS into machines or systems, additional measures may be required so that the product standards of these machines or systems are complied with. The machine or system builder is responsible for taking these measures.

Environments and categories

The EMC environments and categories are defined in the EMC Product Standard EN 61800-3, as follows:

Environments

IEC/EN 61800-3 differentiates between the first and second environments - and defines different requirements for these environments.

First environment

Residential buildings or locations at which the drive system is directly connected to a public low-voltage line supply without intermediate transformer.

Second environment

All locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

Categories

IEC/EN 61800-3 differentiates between four drive system categories:

Category C1

Drive systems for rated voltages < 1000 V for unrestricted use in the first environment.

Category C2

Stationary drive systems for rated voltages < 1000 V for operation in the second environment.

The drive system must be installed by appropriately qualified and trained personnel. Additional measures are required for operation in the first environment.

Category C3

Drive systems for rated voltages < 1000 V - only for operation in the second environment.

Category C4

Drive systems for IT line supplies for operation in complex systems in the second environment. An EMC plan must be drawn up.

SIMOTION

SIMOTION D is designed for industrial use in accordance with Category C2.

Interference immunity

SIMOTION D and SINAMICS S120 are suitable for operation in the second environment. With regard to interference immunity, SIMOTION D and SINAMICS S120 can be used in the first and second environments.

Emitted interference

With regard to interference emission, to comply with the limit values in accordance with IEC/EN 61800-3 second environment, Category C2; in particular for the drive system, certain measures must be observed (EMC-conform installation by appropriately qualified and trained personnel, possibly required radio interference suppression filters, etc.) Observe the information on electromagnetic compatibility (EMC).

EMC notes

Note

Further notes can be found in the SINAMICS S120 Equipment Manuals for the Booksize and Blocksize power units in chapter "Control cabinet installation and EMC".

Note

Requirements for implementing EMC are listed in EN 61000-6-2, EN 61000-6-4, EN 61800-3, EN 60204-1 and in the "EMC Installation Guidelines" Configuration Manual. (https://support.industry.siemens.com/cs/ww/de/view/60612658)

Note

Conformance with the EMC Directive of the EC is ensured by following the measures described in the "EMC Installation Guideline" Configuration Manual.

Protective conductor connection and potential equalization

Requirement

You have mounted the Control Unit in the control cabinet.

The SIMOTION D4x5-2 Control Unit has a connection for the protective conductor and a connection for the equipotential bonding conductor:

- Screw M5, Torx T25
- Tightening torque: 3 Nm (26.6 lbf in)

Only one connection is available for the CX32-2 - this must be used both for the protective conductor and the equipotential bonding conductor.

Note

Requirements for functional safety for machines and systems; reliability and EMC are only guaranteed with original SIEMENS cables.

Note the following safety information:

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

Protective conductor connection

SIMOTION D and the SINAMICS S120 drive system are designed for use in cabinets with a protective conductor connection.

All system components and machine parts must be incorporated in the protection concept. The drive line-up must be arranged on a common bright mounting plate ① in order to comply with the EMC limit values. The connection ② establishes a low-impedance connection to the mounting plate.

The mounting plate must be connected to the protective conductor connection of the control cabinet. For this purpose, a connection ③ must be established to the protective conductor bar ④. The protective conductor bar ④ must be connected to the protective conductor ⑤.

The protective connection (PE connection) that is used for the motors ⑥ must be established through the motor cable.

For EMC reasons, the motor cable shield must be laid flat at both the Motor Module (MM) / Power Module (PM) and motor.

A protective connection ⑧ must also be established for components that are **not** connected with low impedance (e.g. control cabinet door via hinges ⑦).

Example: Booksize axis grouping comprising Control Unit (CU), Line Module (LM), and Motor Modules (MM) as well as drive in blocksize format comprising a Power Module (PM) with snapped on Control Unit (CU)

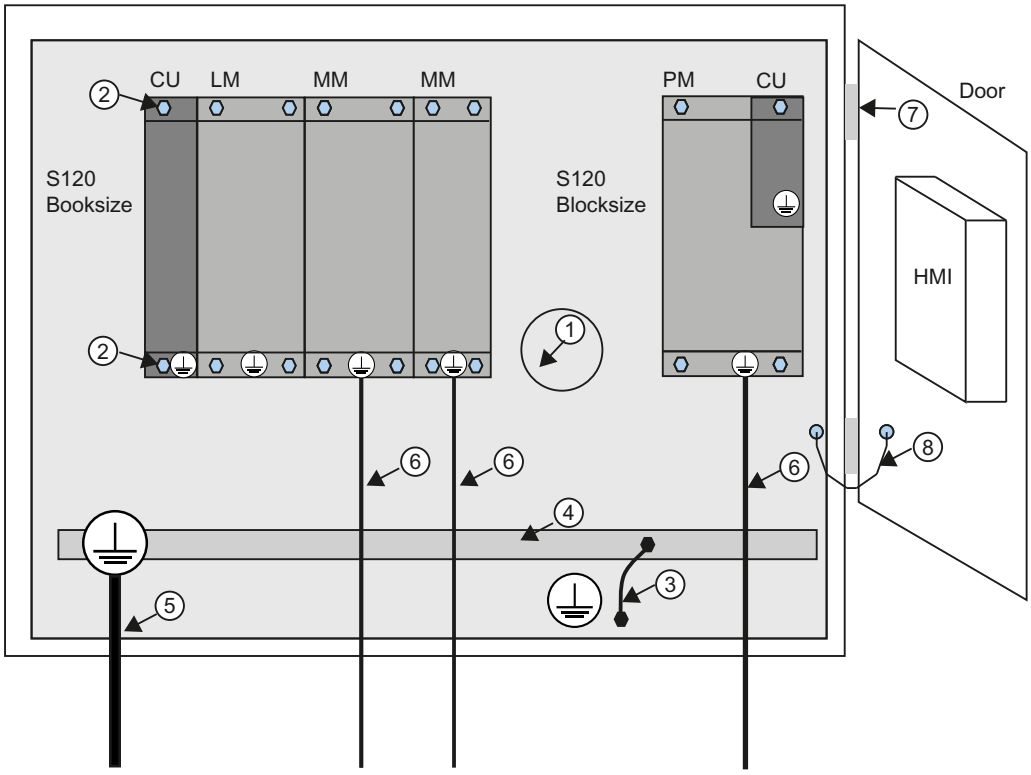


Figure 10-16 Protective conductor connection, cabinet with mounting plate / equipotential bonding surface

The protective conductor connections must be dimensioned as follows:

Table 10-12 Conductor cross-section for copper protective connections

Line supply cable in mm²	Copper protective connections in mm²
Up to 16 mm²	The same as the line supply cable
From 16 mm² to 35 mm²	16 mm²
From 35 mm²	0.5 x line supply cable

For materials other than copper, the cross-section should be increased so that as a minimum, the same conductivity is attained.

Equipotential bonding

A mounting plate serves simultaneously as an equipotential bonding surface. This means that no additional equipotential bonding is required within the drive line-up. If a common bright mounting plate is not available, then equally good equipotential bonding ⑨ must be established using cable cross-sections as listed in the table above or, as a minimum, with the same conductivity.

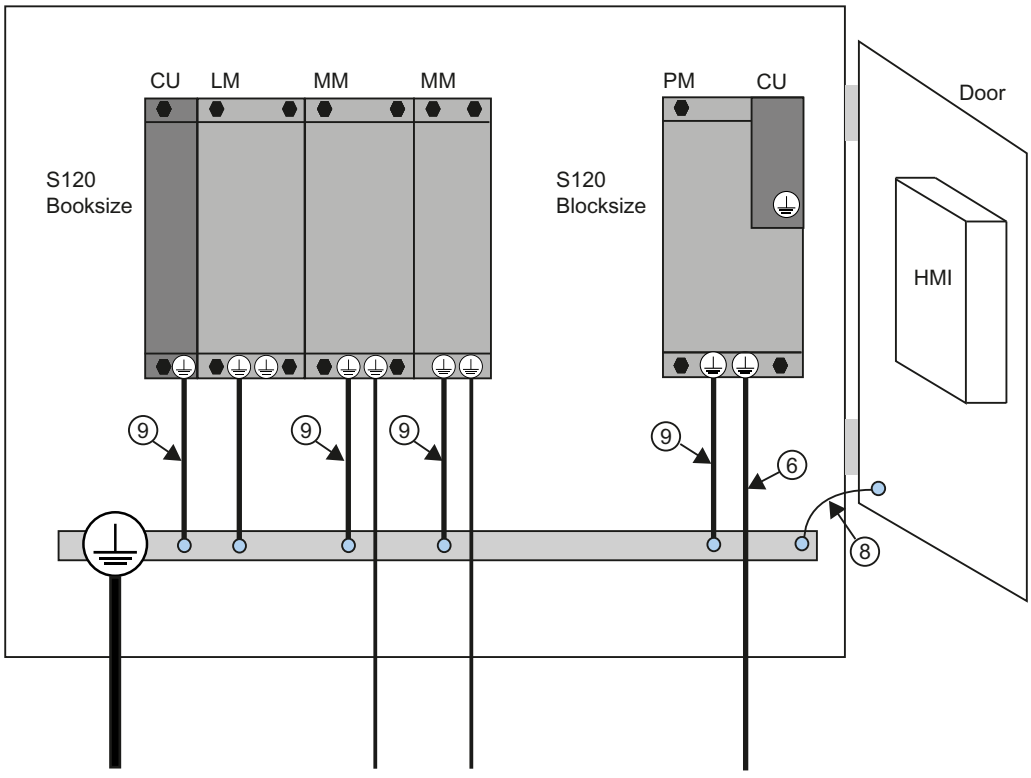


Figure 10-17 Protective conductor connection without equipotential bonding

Note

Ground potential and housing (PE) for the SIMOTION D4x5-2 and CX32-2 are connected internally with low impedance.

Communication links

Within the same control cabinet, no equipotential bonding conductor is required for fieldbus components if they installed as described above.

You have to ensure equipotential bonding for communication connections between remote components in a system (e.g. devices in different control cabinets) as well as between buildings or building sections.

If, for example, data cables (PROFIBUS, PROFINET, Ethernet or DRIVE-CLiQ) are routed through several control cabinets, equipotential bonding must be established with an equipotential bonding conductor. Install the equipotential bonding conductor together with the data cable.

The following minimum cross-sections are required according to IEC 60364-5-54:

- For copper, at least 6 mm²
- For aluminum, at least 16 mm²
- For steel, at least 50 mm²

NOTICE
Fault of the data link or device defect with missing equipotential bonding
Considerable leakage currents can flow via the data cable if equipotential bonding is not provided. Disturbance of the data link or device faults may result.
Install an equipotential bonding conductor together with the data cable.

Due to the maximum length of 100 m for PROFIBUS copper cable at 12 Mbit/s or for PROFINET copper cable and due to the electrical isolation, EMC protection and equipotential bonding aspects, it is recommended that fiber-optic cables be used for connections between buildings.

Additional information

Further information on the protective connection and equipotential bonding can be found in the following references:

- SINAMICS drive system: See the SINAMICS manuals
- PROFIBUS and PROFINET: See the following Internet address (http://www.profibus.com) (at Downloads)

Opening the front cover

Introduction

The interfaces on the front side of the D4x5-2 and the CX32-2 have a cover. This is opened by swinging the front cover down; the interface connections are then accessible and can be wired.

A hinge connects the front cover to the front of the housing. Once opened, the cover can be removed. The front cover is closed by swinging it up; it automatically locks by means of a hook at the top of the device housing.

Note

The cover of the D4x5-2 and CX32-2 can only be removed when it has been swung open through approx. 45°.

Procedure for D4x5-2 and CX32-2

1. Release the latch at the top on the inside of the front cover by pushing down the hook (see the following figure).
2. Remove the front cover with a forward motion.

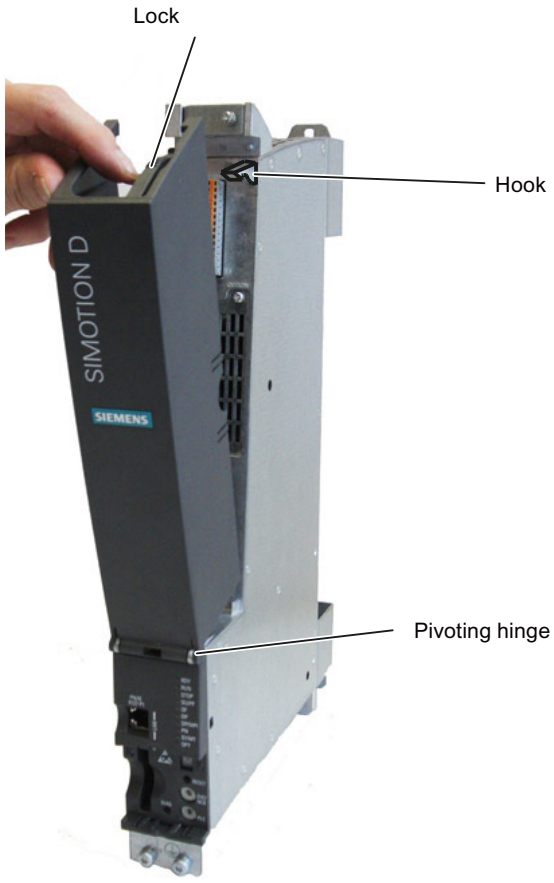


Figure 10-18 Opening the front cover using the D4x5-2 as an example

Note

All cables must be routed vertically upwards to the fullest extent possible so that the front cover can be closed. The front cover has an outlet for the upward routing of these cables.

See also

Removing the cover of the CX32-2 (Page 6938)

Power supply

Safety rules

Basic rules

Because of the wide range of possible applications, only the basic rules for electrical installation are described in this section. At a minimum, you must comply with these basic rules to ensure problem-free operation.

Rules for safe operation

In order to ensure safe operation of your equipment, implement the following measures, adapting them to suit your conditions:

- An EMERGENCY STOP concept in accordance with the generally accepted rules of current engineering practice (e.g. European standards EN 60204, EN 418, and similar).
- Additional measures for end position limiting of axes (e.g. hardware limit switches).
- Equipment and measures for protection of motors and power electronics in accordance with the SINAMICS installation guidelines.
  To identify hazards, we also recommend that a risk analysis be conducted on the entire system in accordance with the basic safety requirements set out in Appendix 1 of the EU Machinery Directive.

Additional references

- Guidelines on Handling Electrostatically Sensitive Devices (ESD), see Appendix.
- For installing a system with SIMATIC ET 200 I/O (e.g. ET 200SP, ET 200MP, etc.), see the manuals for the relevant ET 200 I/O systems.
- As a further source of information on EMC, we recommend the *EMC Installation Guidelines / Basic System Requirements*Configuration Manual.
  Download: EMC (https://support.industry.siemens.com/cs/ww/en/view/60612658)

Standards and regulations

VDE guideline compliance

During wiring, you must observe the appropriate VDE guidelines, in particular VDE 0100 and VDE 0113 for tripping devices and short-circuit and overload protection.

System startup after certain events:

The following list identifies considerations required for startup of a system following certain events.

- If the system starts up again following a voltage drop or power failure, all hazardous operating states must be prevented from occurring. If necessary, force an EMERGENCY OFF.
- If the system starts up again after the EMERGENCY OFF apparatus is released, the startup must not be unchecked or undefined.

Mains voltage

Rules for the line voltage

The following list indicates what you must take into account for the line voltage:

- For stationary installations or systems that do not have all-pole line disconnect switches, the building installation must include a line disconnect switch or a fuse.
- For load power supplies and power supply modules, the rated voltage range set must correspond to the local line voltage.
- For all circuits, the fluctuation/deviation of the line voltage from the rated value must be within the permitted tolerance (see the technical data for the SIMOTION D and SINAMICS modules).

24 VDC supply

For...	Requirement	
Buildings	External lightning protection	Install lightning protection (e.g. lightning conductors).
24 VDC supply lines, signal lines	Internal lightning protection	
24 V supply	safe (electrical) isolation of the extra-low voltage	

Protection against external electrical interference

The table below shows how you must protect your system against electrical interference or faults.

Table 10-13 External electrical phenomena

For ...	Requirement
All plant or systems in which the component is installed	The plant or system is connected to a protective conductor for the discharge of electromagnetic interference.
Supply, signal, and bus lines	The wiring arrangement and installation complies with EMC regulations.
Signal and bus lines	A cable or wire break cannot lead to undefined states in the plant or system.

Additional references

The same installation instructions apply for the SIMOTION D4x5-2/CX32-2 control units as for the SINAMICS S120 CU320-2 control units with regard to EMC.
See the SINAMICS manuals.

Connecting the power supply

24 V DC power supply

Power is supplied by an external 24 V DC power supply. Power supplies from the SITOP family are suitable, for example.

The required 24 V DC load power supply is connected at the screw-type terminal block. The terminal block must be screwed on tightly using a flat-bladed screwdriver.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Implement the 24 V DC supply with safe separation as protective extra-low voltage.

When connecting an external 24 V DC power supply to the interfaces, it must meet the requirements of protective extra low voltage (PELV) according to UL 61010. A series fuse must also be used, which reliably trips within 120 seconds when a short-circuit occurs at an ambient temperature of 0 °C.

The contact gap of the fuse or the individual error-proof circuit must be ensured with 3.0 mm according to UL 61010 for a primary power supply from OVC III circuits up to 600 V AC (line-to-neutral voltage).

When using an external power supply, make sure that the fuse used has a tripping rating that corresponds to the maximum possible short-term short-circuit current of the power supply unit used.

Note

Ground potential and housing (PE) are connected internally with low impedance. Therefore, note the following:

- Insulation monitors in the 24 V power supply are not permitted.
- When using external power supplies (e.g. SITOP), you need to connect the ground potential to the protective conductor terminal (PELV).
- Sufficiently dimensioned equipotential bonding connections must be provided between the 24 V power supply and all grounded, locally separated loads. You can find information about equipotential bonding in the section Protective conductor connection and equipotential bonding (Page 6945).

Disconnecting 24 V plug-in connections during operation

Observe the following safety instructions when using SIMOTION D4x5-2/CX32-2:

⚠ WARNING

Personal injury and damage to property may occur

Personal injury and damage to property may occur if you disconnect 24 V plug-in connections during operation. Disconnecting 24 V plug-in connections is only permitted in a de-energized state.

Connecting the power supply

For the power supply wiring, use rigid or flexible cables with a conductor cross-section as stated in the *SIMOTION D4x5-2* Manual, Section "Interfaces... Power supply."

If you only use one wire per connection, a ferrule is not required.

You can use ferrules without an insulating collar in accordance with DIN 46228, Form A long version.

The following special requirements apply for the connection cables:

- The 24 V DC cable must be approved for temperatures up to 75 °C.
- The maximum permissible cable length is 10 m.
- Select the permitted conductor cross-section in accordance with the national regulations (NEC, VDE, etc.). The basis for this can be the output current of the 24 V DC supply or the overcurrent protection device used in the 24 V circuit. If the 24 V power supply unit that is used has a short-circuit current greater than 50 A, a corresponding overcurrent protection device, which limits to this value, must be used upstream of the product.
- The protective conductor must have a minimum cross-section according to EN 60204-1.
- Strip 7 mm of the cable for connection to the 24 V DC plug.
- Observe the permissible bending radius of the cables.
- Route all the cables so that they cannot be crushed or pinched.
- Route all of the cables in such a way that they do not come into contact with chafing edges.

Note

If power supply units with primary-side supplies up to 600 V AC (line-to-neutral voltage) have to be deployed, the transient voltages on the primary side of the power supply unit must be limited to 4000 V. Connect the device only to a 24 V DC power supply that is compliant with protective extra-low voltage (PELV) requirements according to UL 61010.

You can find information on the current consumption of SIMOTION D4x5-2 in the Power Supply section of the SIMOTION D4x5-2 Manual.

Pin assignments

For the pin assignment of the screw-type terminal block, see the *SIMOTION D4x5-2* Manual.

Connecting DRIVE-CLiQ components

DRIVE-CLiQ wiring

Introduction

The components of the SINAMICS S120 drive family and the SIMOTION D4x5-2 with CX32-2 are wired together by means of DRIVE-CLiQ. DRIVE-CLiQ is a communication system that enables the SIMOTION D4x5-2 to detect the connected components automatically. It provides a wiring tree whose topology can be visualized in SIMOTION SCOUT.

Note

For information on the number of DRIVE-CLiQ interfaces and their properties, refer to the *SIMOTION D4x5-2* Manual.

Rules for the DRIVE-CLiQ wiring

The following rules must be followed for DRIVE-CLiQ wiring:

- Ring wiring is not permitted
- Components must not be double-wired
- For a motor module, the power line for the motor and the associated motor encoder must be connected.

Further information on DRIVE-CLiQ wiring can be found

- In the *SINAMICS S120* Function Manual, Section *Rules for wiring with DRIVE-CLiQ*
- In the *SINAMICS S120 Control Units and Supplementary System Components* Manual

Example

The example in the following figure shows the rules for DRIVE-CLiQ wiring.

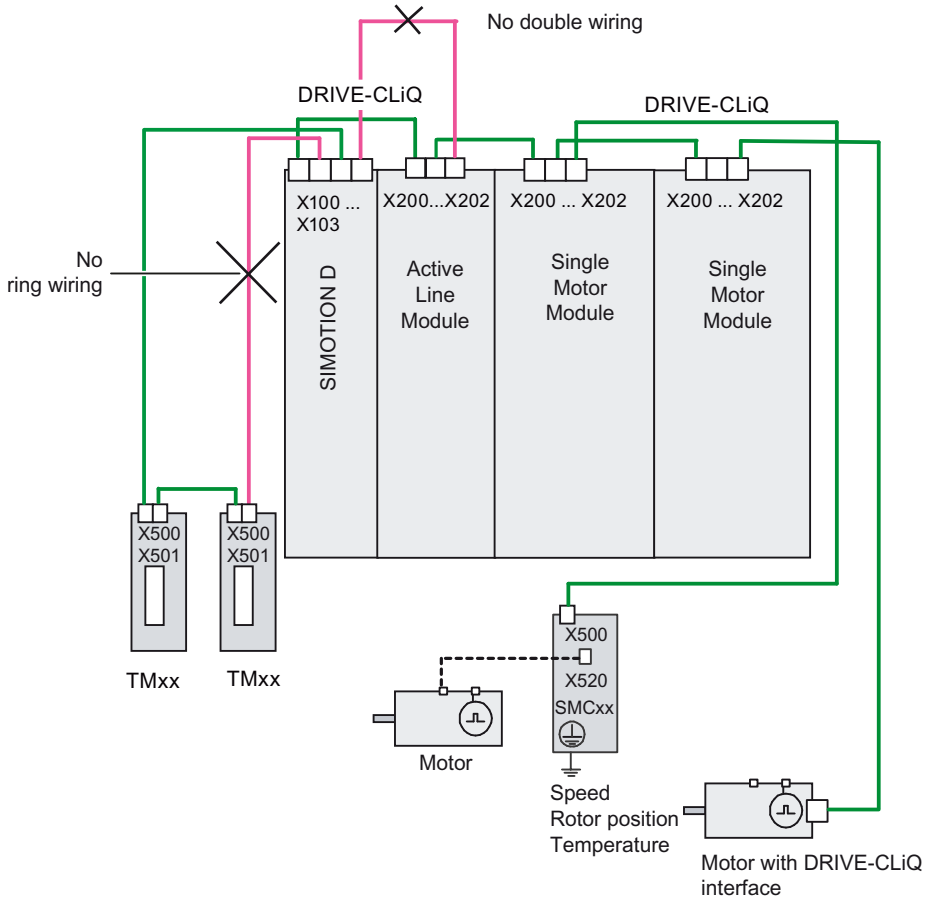


Figure 10-19 DRIVE-CLiQ wiring using the D425-2 as an example

Connectable DRIVE-CLiQ components

Components

As a general principle, all SINAMICS components approved for SIMOTION D can be connected directly to SIMOTION D or another DRIVE-CLiQ component using the DRIVE-CLiQ interface.

Table 10-14 DRIVE-CLiQ

Component	Description
CX32-2 controller extension	The CX32-2 enables scaling of the drive-side computing performance of the SIMOTION D4x5-2. Each CX32-2 can operate up to 6 additional servo, 6 vector or 12 *V/f* axes.
Line Module	Line modules (e.g. Active Line Modules) provide the DC-link voltage and can be connected via DRIVE-CLiQ depending on the module type.

Component	Description
Motor Module	Motor Modules are used to control motors. SMC modules for processing encoder signals, for example, can be connected to Motor Modules. Motor Modules are available in the version: • Single Motor Module (SMM) for connection of a motor • Double Motor Module (DMM) for connection of 2 motors
Motors with DRIVE-CLiQ interface	Motors with a DRIVE-CLiQ interface allow simplified commissioning and diagnostics, because the motor and encoder type are identified automatically.
SMx modules	SMx Sensor Modules allow the acquisition of encoder data from the connected motors via DRIVE-CLiQ.
TM15 and TM17 High Feature Terminal Modules	The TM15 and TM17 High Feature Terminal Modules are used to implement inputs for measuring inputs and outputs for cam outputs. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times. • TM15: 24 isolated bidirectional digital I/Os, with measuring input and output cam functionality • TM17 High Feature: 16 non-floating, bidirectional digital I/Os with measuring input and output cam functionality for the highest demands with respect to resolution, accuracy and short input delay times.
TM31 Terminal Module	TM31 provides 8 DI, 4 bidirectional digital I/Os, 2 relay outputs, 2 AI, 2 AO and 1 temperature sensor input (KTY84-130 or PTC).
TM41 Terminal Module	TM41 provides 4 DI, 4 bidirectional digital I/Os, 1 AI and 1 TTL encoder output.
TM54F Terminal Module	TM54F provides the following interfaces: 4 fail-safe DO (F-DO), 10 fail-safe DI (F-DI), 2 sensor power supplies with dynamic capability, 1 sensor power supply without dynamic capability and 4 DI to check the F-DO during a test stop.
DMC20/DME20	DRIVE-CLiQ hubs provide 4 more DRIVE-CLiQ interfaces. They can be used, for example, to establish point-to-point topologies. • DMC20 is the hub for the control cabinet configuration • DME20 is the hub for use without a control cabinet (IP67 degree of protection)
SINAMICS S120 Power Module (with CUA31/CUA32)	A SINAMICS Power Module in blocksize format can be connected using the CUA31/CUA32 control unit adapter. You will find more information on the CUA31/CUA32 in the *SINAMICS S120 AC Drive* Manual.

Example of axis grouping

The following figure shows a possible DRIVE-CLiQ wiring scheme for an axis grouping.

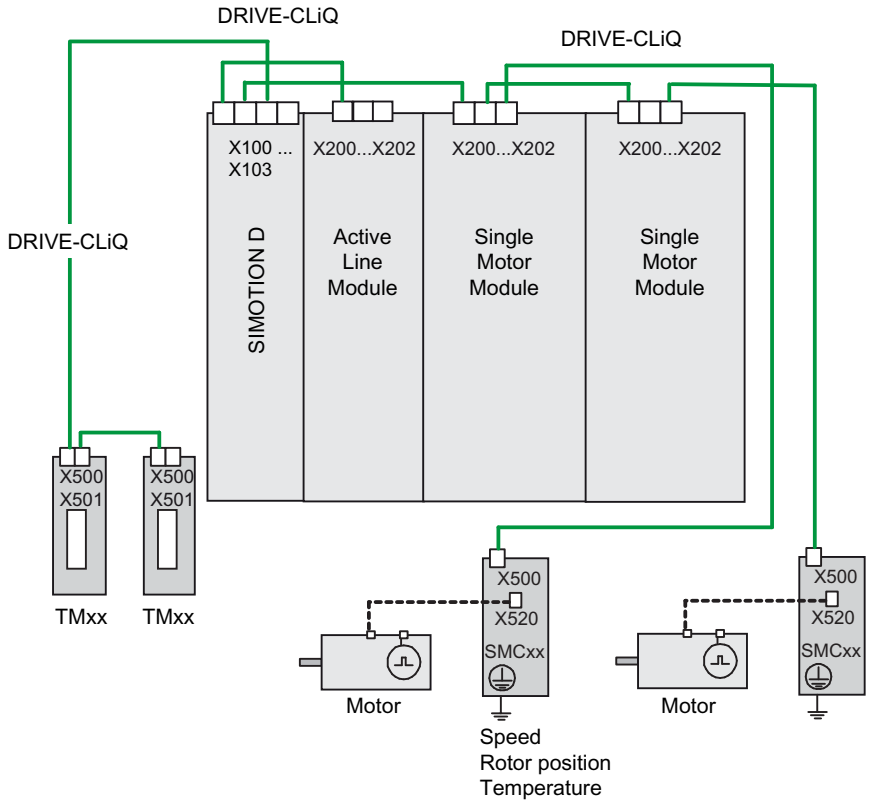


Figure 10-20 Axis grouping with DRIVE-CLiQ using the D425-2 as an example

Restrictions

Some "older" DRIVE-CLiQ components can no longer be used with the SIMOTION D4x5-2/ CX32-2. Refer to the Permissible combinations (Page 7206) for details.

Connecting the CX32-2

Installation/Mounting/Wiring

Information on installing, mounting and wiring the CX32-2 can be found in the *SIMOTION D4x5-2* Manual.

CX32-2 DRIVE-CLiQ topology

In contrast to other DRIVE-CLiQ components (e.g. terminal modules), special rules apply when wiring the CX32-2.

- Only a point-to-point topology is possible between the CX32-2 and SIMOTION D. Each CX32-2 needs its own DRIVE-CLiQ port on the SIMOTION D control unit.
- When a CX32-2 is inserted in an existing DRIVE-CLiQ connection (target port for CX32-2 is occupied, e.g. by a TM31), this connection is disconnected and replaced by the CX32-2 connection. The component that is freed up is moved to the component archive of the SINAMICS topology overview. A notice is displayed indicating that the component has been moved to the archive. The components must then be reassigned.
- A CX32-2 is inserted via **HW Config** (see Section Configuration of a CX32-2 (Page 7088)). Whereby the selection made for the PROFIBUS address automatically and permanently assigns the DRIVE-CLiQ port for connecting the CX32-2. Given that this assignment is permanent, the following points must be noted:
  - An inserted and configured CX32-2 cannot be connected to another DRIVE-CLiQ port without taking additional measures. Reconnecting a CX32-2 results in a discrepancy between the specified and actual topologies of the DRIVE-CLiQ components.
  - A cross-exchange of two occupied DRIVE-CLiQ ports is not permitted. Such an exchange results in inconsistencies in the specified-actual topologies.
  - Once it has been created in **HW Config**, the connecting port of a CX32-2 cannot be changed.
  - To change the connecting port of a CX32-2, the CX32-2 must be deleted from **HW Config** and recreated with another address.
  - To delete a CX32-2, it must be deleted from HW Config and then the configuration saved and compiled.
  - It is not possible to replace a control unit with six DRIVE-CLiQ ports (D435-2, D445-2 or D455-2) with a control unit with four DRIVE-CLiQ ports (D425-2) when a CX32-2 has been configured with address 14 or 15 (a control unit with four DRIVE-CLiQ ports does not have the DRIVE-CLiQ ports X104/X105 and consequently not the addresses 14 and 15).

– The previous CX32 version cannot be used on a D4x5-2, see Section Permissible combinations (Page 7206) in Migration of D4x5 to D4x5-2.

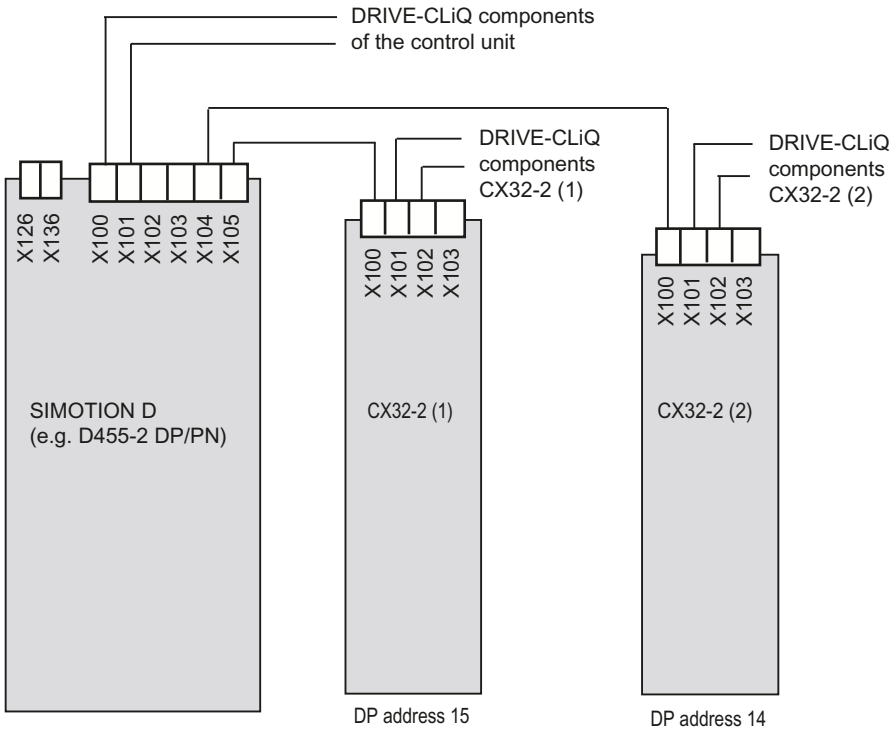


Figure 10-21 CX32-2 topology

See also

Quantity structures (Page 7199)

Additional references

Further information on the CX32-2 can be found in the *SIMOTION D4x5-2* Manual.

Connecting I/Os

Connection cables for the inputs/outputs of the D4x5-2, CX32-2, and TB30

For the input/output wiring, use rigid or flexible cables with a conductor cross-section as stated in the manual.

Ferrules are not mandatory.

You can use ferrules without an insulating collar in accordance with DIN 46228, Form A long version.

Note

To achieve optimum interference immunity, shielded cables must be used for connecting analog signals, measuring inputs or external zero marks.

Tools required

Screwdriver 0.4 x 2.0 mm

Wiring inputs/outputs

1. Strip back the cable insulation as stated in the manual, if necessary, press on a ferrule.
2. Wire the digital inputs and outputs for connection of the sensors and actuators.
3. Insert the cable into the appropriate spring-loaded terminal. This is easier when you use the screwdriver to push back the spring.

Additional references

For further information on cabling (e.g. use of rigid cables) and the pin assignment of the I/O -interfaces, see:

- *SIMOTION D4x5-2* Manual, Section "Digital inputs/outputs."
- *SIMOTION D4x5-2* Manual, Section "Controller Extension CX32-2."
- *SIMOTION D4x5-2* Manual, Section "Digital inputs/outputs."

Creating a shield connection

Using shielded cables

The following options are available for the shield connection when using shielded cables:

- A shield connection using a shielding bus supplied separately
- Shield connection via shield connecting element on the top of the SIMOTION D4x5-2/CX32-2 housing

Using a shielding bus

If a shielding bus is used, proceed as follows:

1. Attach the cable shield to a grounded shielding bus after the cable entry point in the cabinet. To do this, expose the cable shield of the cable.
2. Continue routing the shielded cable as far as the module, but do not make a connection to the shield there.

Using the shield connection on the D4x5-2/CX32-2

1. Unscrew the holding clamp of the shield connection at the top of the D4x5-2/CX32-2 until there is a space below the clamp (M3 screw, Torx T10).
2. Insert the cable. To do this, expose the cable shield first.
3. Tighten the fixing bracket so that the cable shield and cable are pressed against the shield connection (tightening torque 0.8 Nm or 7.1 lbf in).

This figure shows where to attach the cables to the front panel connector and where to apply the cable interference suppression using the shield connecting element.

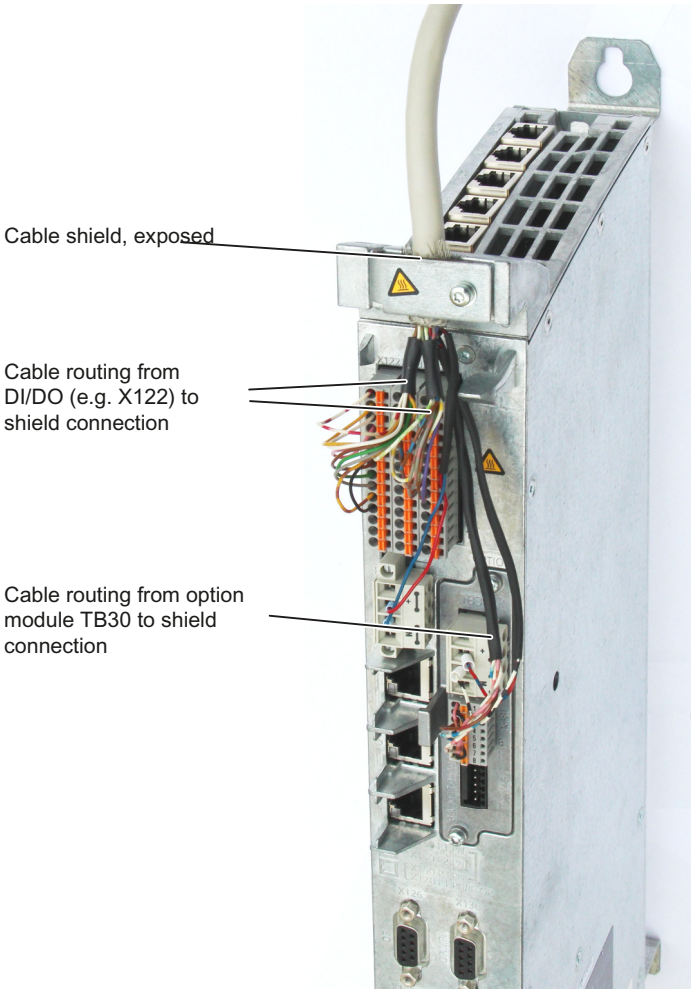


Figure 10-22 Shield connection using the D4x5-2 DP/PN as an example

Note

With a CX32-2, the shield is connected in the same way as the D4x5-2. A CX32-2 only has the X122 terminal block and the shield connection element is narrower on a CX32-2 than on a D4x5-2.

Connecting I/Os of the TB30

Pin assignment

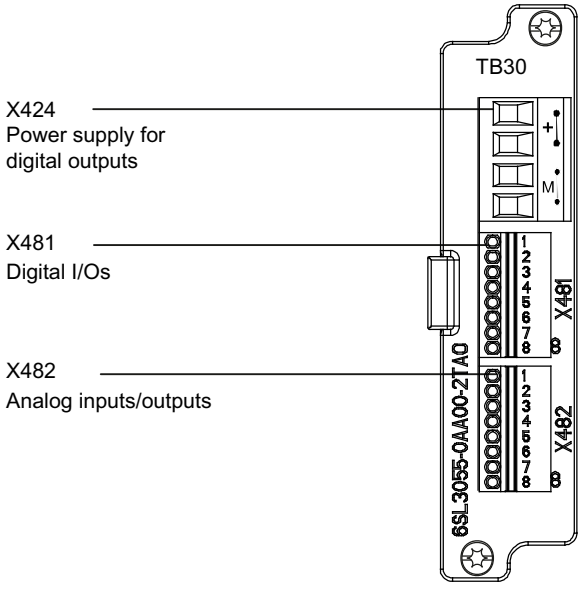


Figure 10-23 Interface arrangement on the TB30

Additional references

Detailed information about the pin assignment of the X424, X481 and X482 interfaces can be found in the *SIMOTION D4x5-2* Manual, Section "Supplementary system components".

Shield connection for TB30

The following figure shows the cable routing, cable connection and the connection of the I/Os on the TB30. The TB30 is inserted in a CU320-2 here.

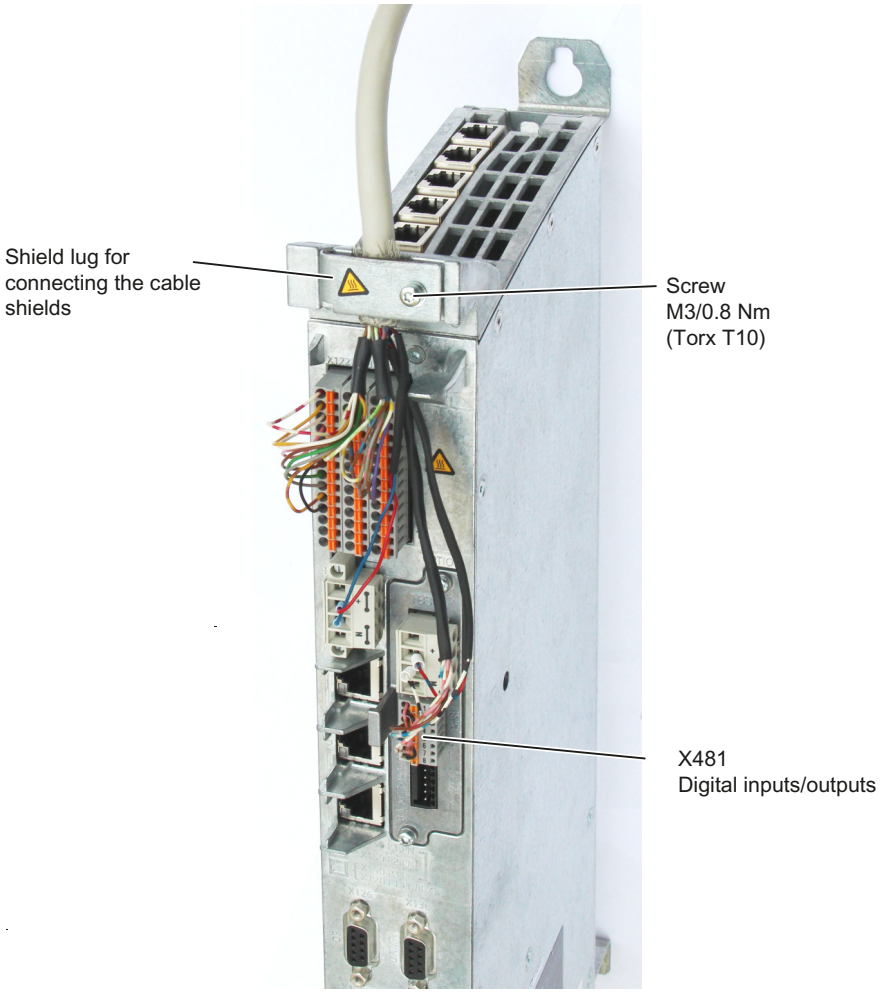


Figure 10-24 Shield connection for TB30 using the digital I/Os as an example

The arrangement is similar when inserted in a D4x5-2, see also Section Creating a shield connection (Page 6961) for more details.

The permissible bending radii for the cables must be maintained when the cables are being laid.

Connecting PROFIBUS/MPI

PROFIBUS connection components

Connection components

Individual nodes are connected by means of bus connectors and PROFIBUS cable. Remember to provide a bus connector with a programming port at the ends of the subnet. This will give you the option of expanding the subnet if required (for example, for a programming device or SIMATIC HMI device).

Use RS 485 repeaters for the connection between segments and to extend the cable.

Segments

A segment is a bus cable between two terminating resistors. A segment can contain up to 32 nodes. In addition, a segment is limited by the permissible cable length, which varies according to the transmission rate.

Terminating resistor

A cable must be terminated with its own surge impedance to prevent line disturbances caused by reflections. Activate the terminating resistor at the first and last node of a subnet or segment.

Make sure that the nodes to which the terminating resistor is connected are always supplied with voltage during power-up and operation.

PROFIBUS cables and connectors

Properties of PROFIBUS cables

The PROFIBUS cable is a two-core, twisted and shielded cable with defined properties:

Properties of the cables

Table 10-15 Properties of PROFIBUS cables

Characteristics	Values
Characteristic impedance	Approximately 135 to 160 Ω (f = 3 to 20 MHz)
Loop resistance	≤ 115 Ω/km
Effective capacitance	30 nF/km
Damping	0.9 dB/100 m (f = 200 kHz)
Permissible conductor cross-section	0.3 mm^2 to 0.5 mm^2
Permissible cable diameter	8 mm + 0.5 mm

Connector features

The bus connector is used to connect the PROFIBUS cable to the PROFIBUS DP interfaces (X126, X136), thus establishing a connection to additional nodes.

Only bus connectors with a 35° cable outlet should be used in order to ensure that the front cover can be closed.

See also

SIMOTION D4x5-2 Manual, Section "Spare parts and accessories which can be ordered"

Length of PROFIBUS cables

Cable lengths and baud rate

The baud rate determines the cable length of a subnet segment.

Table 10-16 Permitted cable length of a subnet segment for specific baud rates

Baud rate	Max. cable length of a segment (in m)
19.6 to 187.5 kbit/s	1000 [1]
500 kbit/s	400
1.5 Mbit/s	200
3 to 12 Mbit/s	100

[1] With isolated interface

Longer cable lengths

If you must implement longer cable lengths than permitted in one segment, you must use RS 485 repeaters. The maximum possible cable lengths between two RS 485 repeaters correspond to the cable length of a segment. You can connect up to nine RS 485 repeaters in series.

Note that an RS 485 repeater must be counted as a subnet node when determining the total number of nodes to be connected, even if it does not have its own PROFIBUS address.

Rules for the laying of PROFIBUS cables

Laying of bus cable

During laying of the PROFIBUS cable, you must:

- Not twist the cable
- Not stretch the cable
- Not squeeze the cable

Supplementary conditions

In addition, when laying a bus cable for indoor use, you must take into account the following supplementary conditions (dA = external cable diameter):

Table 10-17 Supplementary conditions for the laying of PROFIBUS cables

Characteristics	Supplementary conditions
Bending radius for a single bend	80 mm (10xdA)
Bending radius for multiple bends	160 mm (20xdA)
Permissible temperature range for cable routing	-5° C to +50° C
Temperature range for storage and stationary operation	-30° C to +65° C

Additional references

Length codes for prefabricated cables can be found in:

- Catalog *SIMOTION PM 21*
- *IK PI Industrial Communication* Catalog

Connecting PROFIBUS DP (interfaces X126 and X136)

PROFIBUS cables are connected to the corresponding interface via a bus connector.

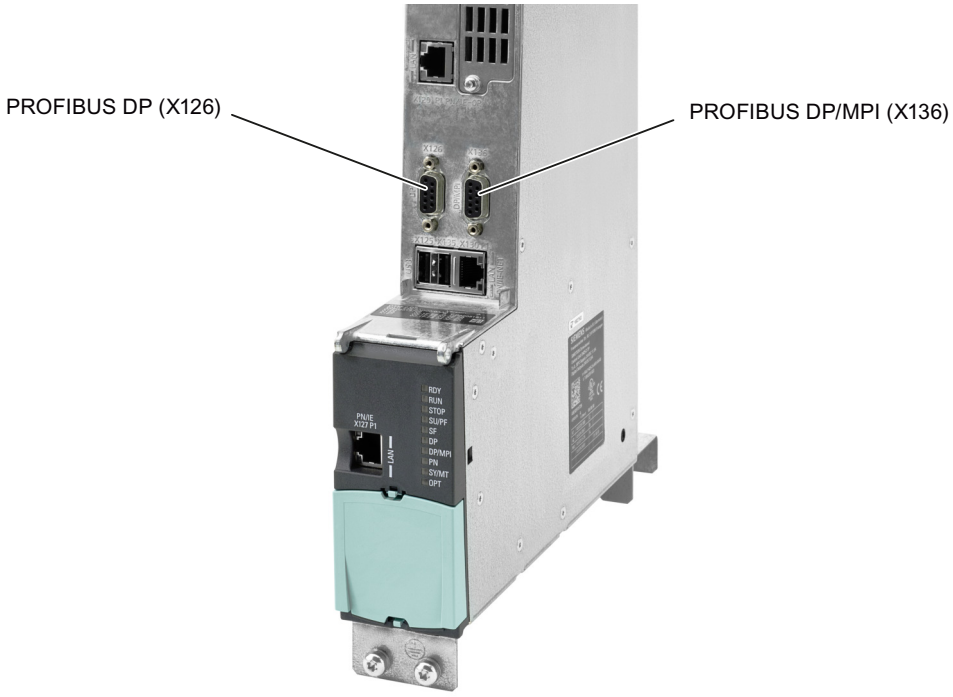


Figure 10-25 Connecting PROFIBUS DP

Connecting the bus connector

Proceed as follows to connect the bus connector:

1. Plug the bus connector into the corresponding interface of the Control Unit.
2. Screw the bus connector into place.
   If the Control Unit is located at the start or end of a segment, you must switch on the terminating resistor ("ON" switch setting).

Figure 10-26 Terminating resistor "switched on and off"

Note

Make sure that the nodes at which the terminating resistor is located are always supplied with voltage during startup and operation.

Removing the bus connector

You can remove the bus connector with a looped-through bus cable from the PROFIBUS DP interface at any time without interrupting data traffic on the bus.

NOTICE
Data communication disturbed because bus terminator missing
A bus segment must be terminated at both ends with a terminating resistor. This is not the case if the last bus connector node is de-energized, for example. Because the bus connector takes its voltage from the station, this terminating resistor is ineffective.
Make sure that the stations at which the terminating resistor is connected are always energized.

Adapter plug

An adapter plug (Article No. 6FX2003-0BB00) is required when the bus cable has to be looped through the PROFIBUS interface (X126; two PROFIBUS cables wired to the plug) and also

- Ethernet interface X120, in the case of D4x5-2 DP or
- PROFINET interface X150 for D4x5-2 DP/PN Port 3

has to be wired to a FastConnect plug.

When using the adapter plug, the PROFIBUS connector is higher, which creates extra wiring space.

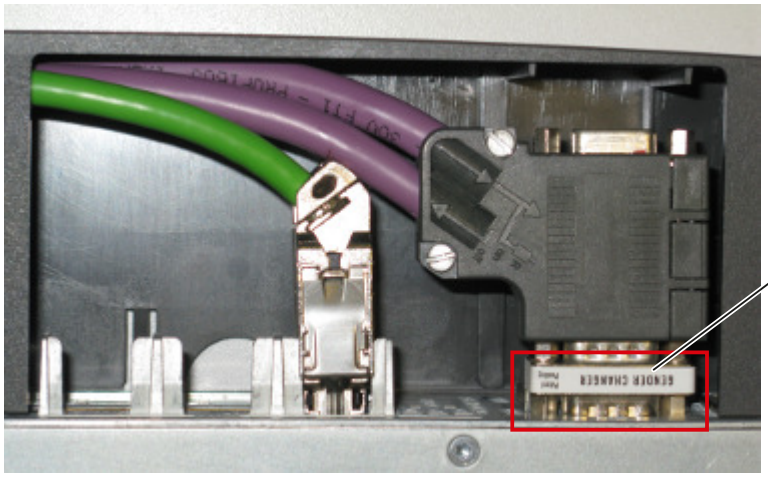


Figure 10-27 SIMOTION D4x5-2 DP/PN with adapter plug

See also

PROFIBUS cables and connectors (Page 6965)

Length of PROFIBUS cables (Page 6966)

Connection rules in the PROFIBUS subnet

Introduction

There are a number of rules for configuring and installing cables for PROFIBUS networks to ensure seamless communication over PROFIBUS. These rules apply to both configuring and cabling as well as address assignment for the different network nodes.

Connection rules

- **Before** you interconnect individual nodes in a subnet, you must assign a unique PROFIBUS address to each node.
- Narrow down the number of nodes by limiting the PROFIBUS addresses to the highest address in the network.
  Tip: Mark the address on the housing of all nodes in a subnet. Then you can always see which address is assigned to which node in your system.
- Connect all nodes in a subnet "in series". No spur lines may be routed to the PROFIBUS DP. In addition, integrate the programming devices and SIMATIC HMI devices for commissioning or servicing in the subnet in series.
- If you operate more than 32 nodes on a subnet, you must use RS 485 repeaters to connect the bus segments. More detailed information can be found in the description of the RS 485 repeater, see the *S7-300 Automation Systems, Module Data* Manual.
  In a PROFIBUS subnet, all segments combined must have at least one DP master and one DP slave.
- Use RS 485 repeaters to connect ungrounded bus segments and grounded bus segments.

- The maximum number of nodes per bus segment decreases with each RS 485 repeater. This means that when a bus segment contains an RS 485 repeater, the bus segment can contain no more than 31 additional nodes. However, the number of RS 485 repeaters does not affect the maximum number of nodes on the bus.
- Up to 10 segments can be connected in a row (max. 9 repeaters).
- **At least** one terminator must be supplied with **5 V**.
  To accomplish this, the PROFIBUS DP connector with an activated terminating resistor must be connected to a device that is switched on.
- Before inserting a new node on the subnet, you must switch off its supply voltage.
  The station must be inserted **first** and then switched on.
  When a station is disconnected, the connection must **first** be deactivated and then the connector withdrawn.
- The bus line of a segment must be terminated at **both ends**. This is achieved by switching on the terminating resistor in the PROFIBUS DP connector at the first and last node and switching off the other terminating resistors.

Example

This illustration below shows an example configuration of a subnet with D4x5-2.

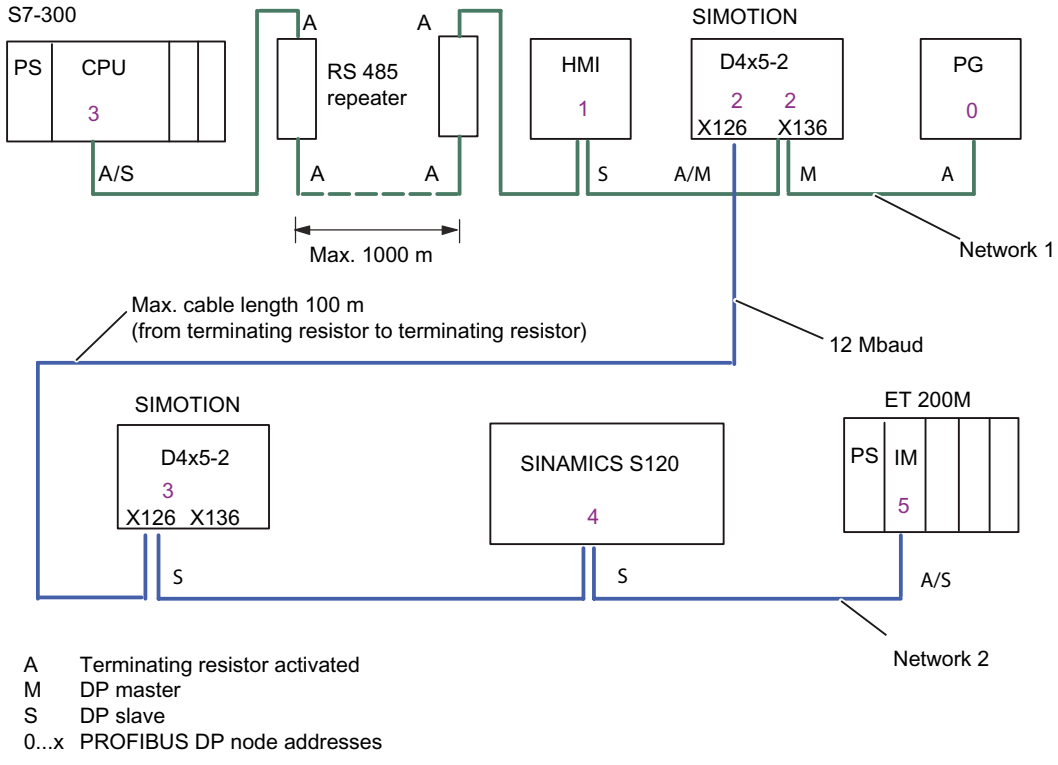


Figure 10-28 Networking example for a D4x5-2

Operating the X136 interface as MPI

Applications

The X136 interface can also be operated as an MPI interface instead of a PROFIBUS DP interface. The typical (default) baud rate is 187.5 Kbaud. A baud rate of up to 12 MBaud can be set for communication with other CPUs. It should be noted, however, that a rate of 12 MBaud is not supported by all CPUs (e.g. smaller SIMATIC S7 CPUs).

The following list provides examples of when using MPI (multi-point interface) may prove effective:

- If a PC/PG is being used with an MPI interface
- If an OP/TP only has an MPI interface
  (newer devices have PROFIBUS or PROFINET interfaces)
- If SIMOTION and SIMATIC CPUs are coupled via XSEND / XRECEIVE

When communicating with XSEND/XRECEIVE, there is no need to configure the connection in **NetPro**. XSEND/XRECEIVE can be used via PROFIBUS or MPI.

- Via PROFIBUS: For communication between SIMOTION devices
- Via MPI: For communication between SIMOTION and SIMATIC S7 devices
  The SIMOTION interface must be connected to the MPI interface of the SIMATIC S7 devices. Connection via PROFIBUS is not possible.
  The baud rate of the SIMATIC S7 device must be set at the SIMOTION interface (see documentation for the relevant SIMATIC S7 devices).

Operate MPI like PROFIBUS

The information on wiring the connector (terminating resistors) and the rules for routing of cables for PROFIBUS apply to this interface as well. When carrying out this procedure, consult the relevant references.

Connector features

The bus connector is used to connect the MPI bus cable to the MPI interface (X136). This enables you to establish connections to additional nodes (e.g. PG or SIMATIC S7-CPU). Only bus connectors with a 35° cable outlet should be used in order to ensure that the front cover can be closed.

Additional information

See *SIMOTION D4x5-2* Manual, Chapter *Available spare parts and accessories*.

MPI bus cable

The PROFIBUS cable specifications apply here as well;

Please note the relevant information on setting up an MPI network.

Setting up an MPI network

Keep in mind the following basic rules when setting up an MPI network:

- When using the interface as an MPI interface, it is not possible to arrange additional control for a drive in isochronous mode or to connect distributed I/Os to this interface.
- An MPI bus line must be terminated at both ends. This is achieved by activating the terminating resistor in the MPI connector in the first and last station and deactivating the other terminating resistors.
- At least one terminator must be supplied with 5 V.
  This means that an MPI connector with an activated terminating resistor must be connected to a device that is switched on.
- Spur lines (cables leading from the bus segment to the station) should be as short as possible, that is, < 5 m in length. Unused spur lines should be removed wherever possible.
- Every MPI station must be connected to the bus first and then activated.
  To disconnect the station, it must first be deactivated. Then, the station can be removed from the bus.
- Maximum cable lengths:
  - 200 m per bus segment
  - 2,000 m total length with RS 485 repeaters

Note

You can also use intelligent DP slave functionality for PROFIBUS communication between CPUs.

Connecting PROFINET IO components

Wiring PROFINET

Procedure

As standard, a SIMOTION D4x5-2 DP/PN has a PROFINET IO interface with three ports.

The SIMOTION D4x5-2 DP/PN control units can be expanded by an additional PROFINET IO interface with four ports. For this purpose, a CBE30-2 must be inserted in the option slot of the control unit.

Suitable PROFINET cables and connectors must be used for the PROFINET connection. The autocrossing functionality of the PROFINET interfaces means crossed as well as uncrossed cables can be used.

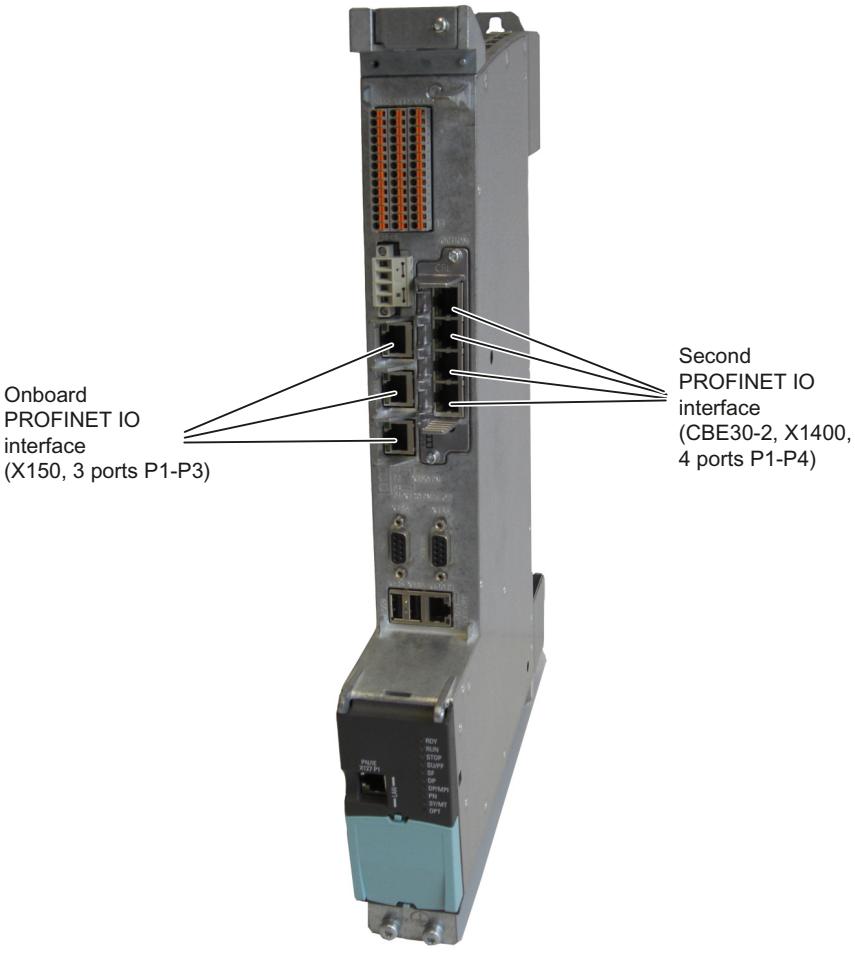


Figure 10-29 Connecting PROFINET IO

Mixed operation of IRT and RT

For mixed operation of IRT and RT, note that the IRT-capable devices must form a so-called IRT domain, i.e. there must not be any non-IRT devices on the data transmission link between the IRT devices.

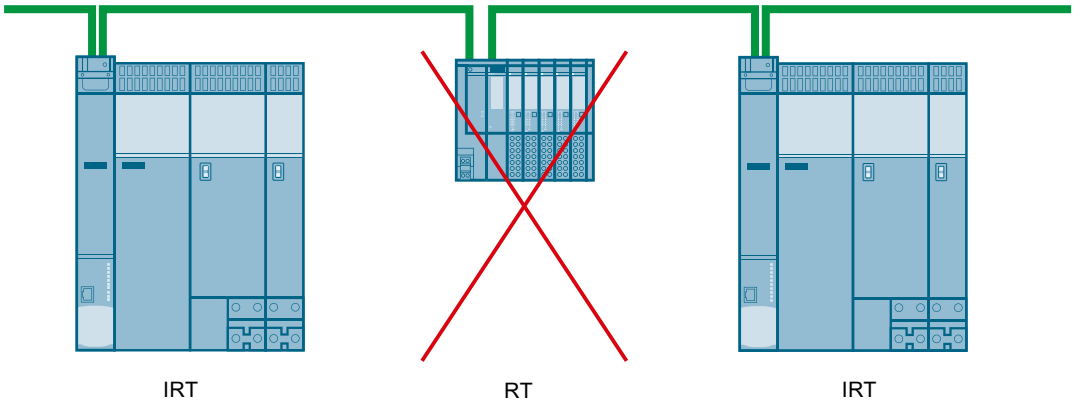


Figure 10-30 Mixed operation of IRT and RT

PROFINET cables and connectors

Cable and connector types

Note

For connecting PROFINET IO to D4x5-2, a connector with a 145° cable outlet is recommended (IE FC RJ45 plug 145).

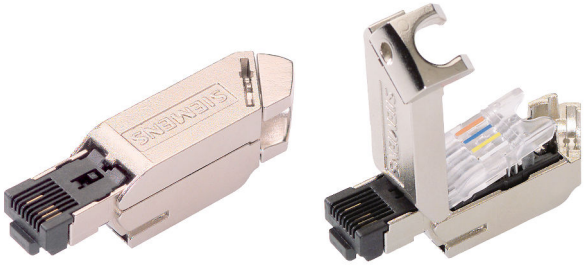


Figure 10-31 RJ45 PN connector with a 145° cable outlet

Table 10-18 Connector types for PROFINET

Connectors	Designation	Article number
IE FC RJ45 plug 145	RJ45 FastConnect connector for Industrial Ethernet/PROFINET with 145° cable outlet	
	• 1 pack = 1 unit • 1 pack = 10 units	6GK1 901-1BB30-0AA0 6GK1 901-1BB30-0AB0

Table 10-19 Cable types for PROFINET

Cable	Designation	Article number
IE FC Cable GP 2x2 (Type A)	4-wire, shielded TP installation cable for IE FC RJ45	6XV1 840-2AH10
IE FC Flexible Cable GP 2x2 (Type B)	4-wire, shielded flexible TP installation cable for IE FC RJ45	6XV1 870-2B
IE FC Trailing Cable GP 2x2 (Type C)	4-wire TP installation cable for trailing cable use	6XV1 870-2D

Cable	Designation	Article number
IE FC Trailing Cable 2x2 (Type C)	4-wire shielded TP installation cable for connection to FC OUTLET RJ45, for ground cable use	6XV1 840-3AH10
IE FC Marine Cable 2x2	4-wire, shielded marine-certified TP installation cable for connection to FC OUTLET RJ45	6XV1 840-4AH10

Table 10-20 Stripping tool for Industrial Ethernet / PROFINET

Tool	Designation	Article number
IE FC Stripping Tool	Stripping tool for Industrial Ethernet / PROFINET	6GK1 901-1GA00

Additional references

For further information about the cables, connectors, and stripping tool, see

- Section "Spare parts and accessories" in the *SIMOTION D4x5-2* Manual
- Catalog *IK PI Industrial Communication*.

Connecting Ethernet

Wiring Ethernet

Procedure

To the 8-pin RJ45 sockets

- X127 P1
- X130 P1
- X120 P1 (only for D4x5-2 DP)

you can connect Industrial Ethernet.

The interfaces support a transmission rate of 10/100/1000 Mbit/s. Suitable Ethernet cables and connectors must be used for the Ethernet connection.

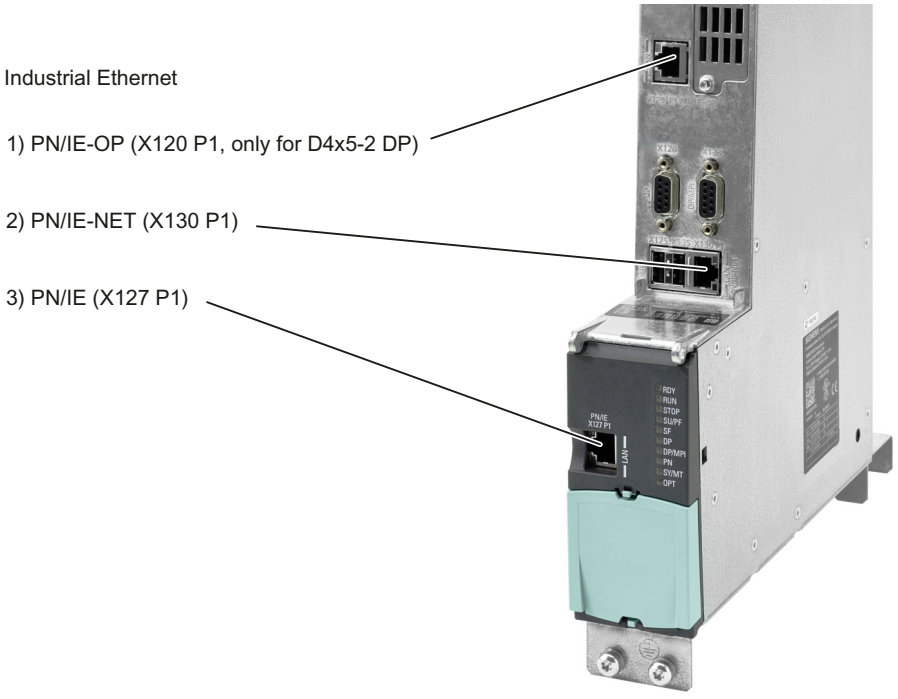


Figure 10-32 Connecting Industrial Ethernet

Note

As of V4.3, the three Ethernet interfaces support the PROFINET basic services - they therefore have the designation PN/IE-NET, PN/IE-OP or PN/IE.

These PROFINET basic services (e.g. DCP, LLDP, SNMP) provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives, I/O modules, etc.

Ethernet cables and connectors

Characteristics of the cables

A shielded twisted pair cable is used for the networking.
The transmission rate of the cable is determined as follows by the number of core pairs:

- 2 x 2 cores support 10/100 Mbit/s
- 4 x 2 cores support 10/100/1000 Mbit/s

Recommended connecting cables

The following cables are available:

- SIMATIC NET, Industrial Ethernet TP XP CORD RJ45/RJ45
  - TP cable prefabricated with 2xRJ45 connectors
  - Crossed send and receive cable
  - Article No.: 6XV1870-3R□□□ (□□□ - length code)
- SIMATIC NET, Industrial Ethernet TP CORD RJ45/RJ45
  - TP cable prefabricated with 2xRJ45 connectors
  - Uncrossed send and receive cable
  - Article No.: 6XV1870-3Q□□□ (□□□ - length code)

The autocrossing functionality of the Ethernet interfaces means crossed as well as uncrossed cables can be used.

Autocrossing

Per default, the Ethernet interfaces are set to "Automatic setting" in HW Config, so that autocrossing is available.

The settings can be found in the port properties (e.g. X127 P1) in the module rack of HW Config (SIMOTION < V4.3: Setting via HW Config "D4x5-2 Properties" > "Ethernet extended").

Autocrossing is deactivated with "Manual setting". As the Ethernet interfaces of the D4x5-2 are wired as Ethernet terminal, in this case you must use crossed cables for the networking with a PG/PC or another D4x5-2.

If the communication peer has autocrossing (e.g. PC, switch or hub), crossed and uncrossed cables can be used.

Additional references

For further information about the cables and connectors, see the *Industrial Communication IK PI* Catalog.

Routing

Routing describes the cross-network transfer of information from Network x to Network y.

Routing on SIMOTION D

Routing between the different interfaces

The Ethernet interfaces of the D4x5-2

- X127 P1 and X130 P1 (for D4x5-2 DP/PN) or
- X127 P1, X120 P1 and X130 P1 (for D4x5-2 DP)

each form a separate IP subnet.

The D4x5-2 DP/PN onboard PROFINET IO interface (X150, P1-P3) and the optional second PROFINET IO interface with CBE30-2 (X1400, P1-P4) also each form a separate IP subnet. All ports of a PROFINET IO interface always belong to the same IP subnet.

- IP routing from IP subnet to IP subnet is not supported. You can use an external IP router for this
- The S7 routing between PROFINET/Ethernet subnets and also to PROFIBUS is possible.

There are therefore the following options for connecting a PG/PC or HMI device to a SIMOTION D using S7 routing.

Engineering system / HMI to PROFINET

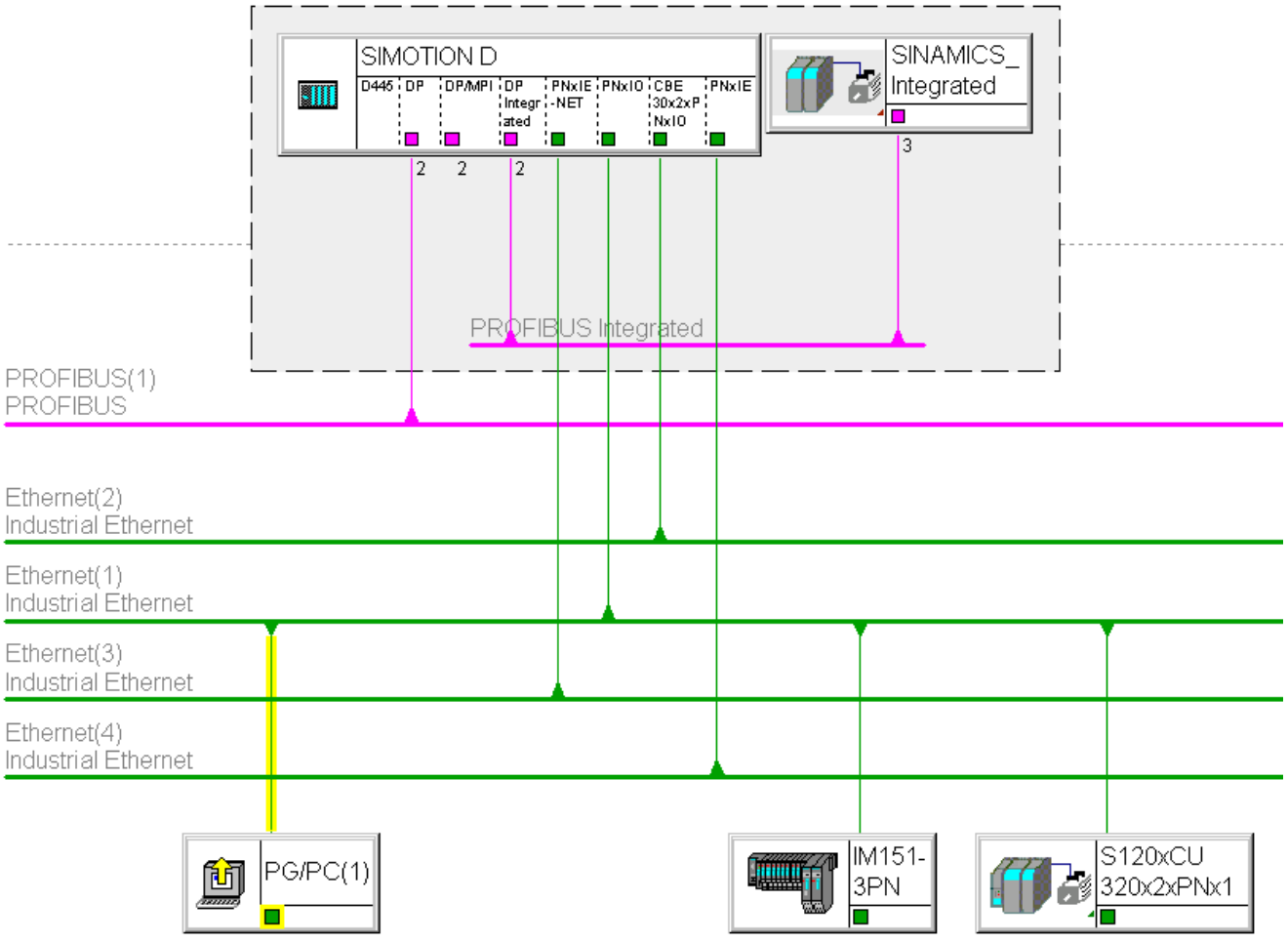


Figure 10-33 Example of PG/PC to PROFINET interface (PNxIO, X150)

- S7 routing to the (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1) (for D4x5-2 DP also to PN/IE-OP (X120 P1))

- S7 routing to the second PROFINET IO interface with CBE30-2 (X1400, P1-P4)
- Access to the components on the same subnet via the switch functionality of the PROFINET IO interface

Engineering system / HMI to PROFIBUS

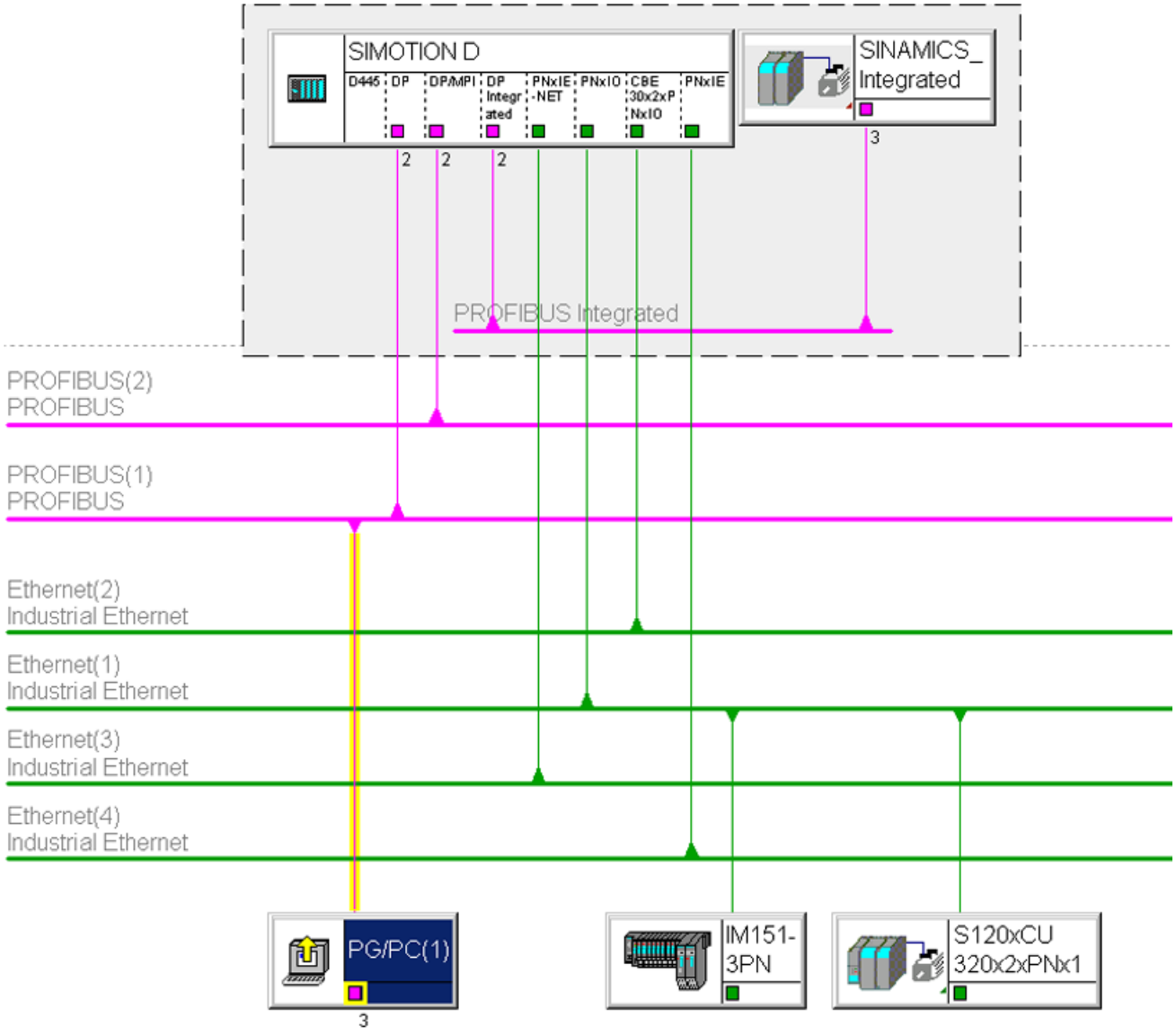


Figure 10-34 Example of PG/PC to PROFIBUS interface (DP, X126)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)

- S7 routing to the second PROFINET IO interface with CBE30-2 (X1400, P1-P4)
- S7 routing to the Ethernet interfaces PN/IE (X127 P1) and PN/IE-NET (X130 P1) (for D4x5-2 DP also to PN/IE-OP (X120 P1))

Engineering system / HMI to Ethernet

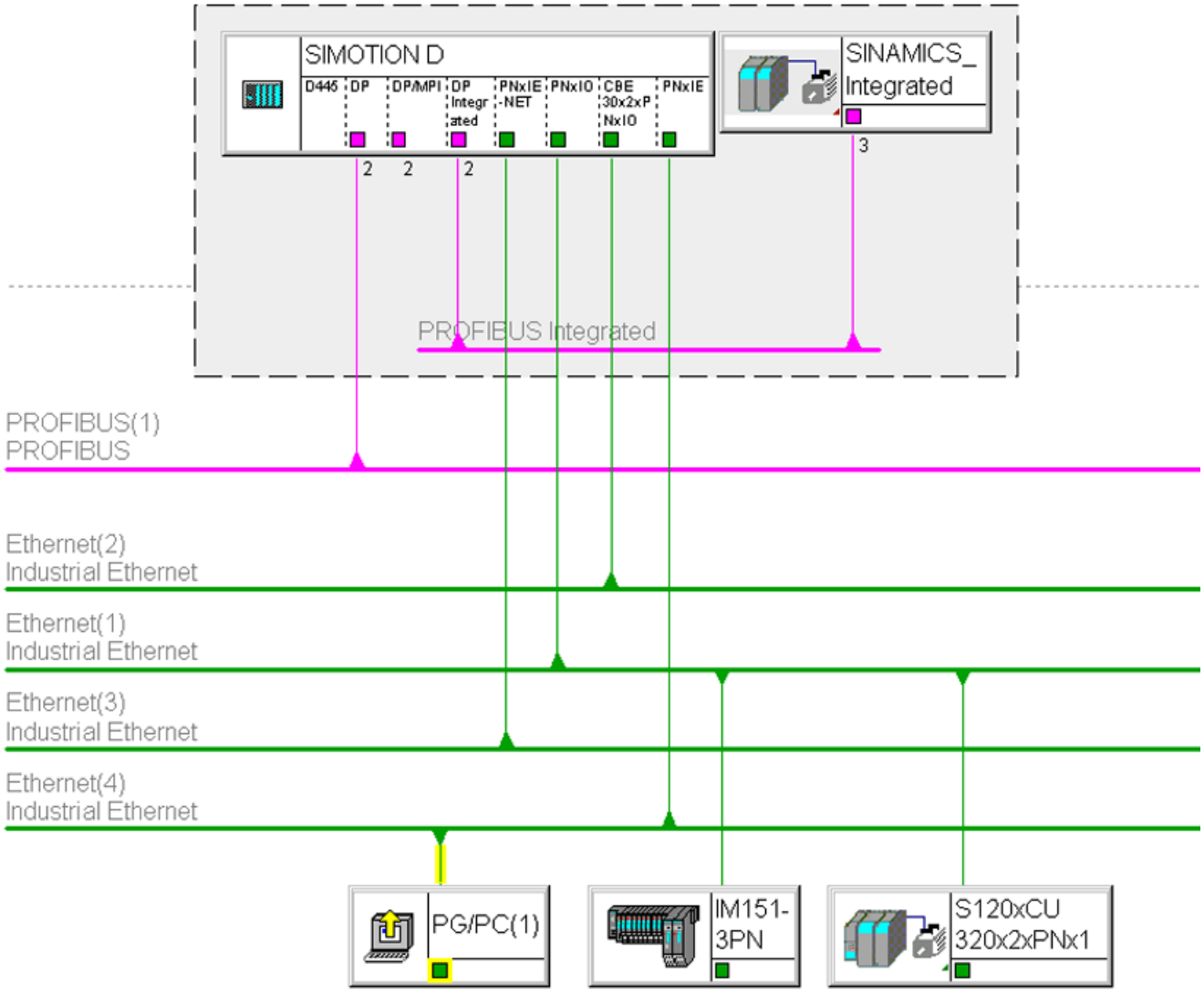


Figure 10-35 Example of PG/PC to Ethernet interface (PNxIE, X127)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the onboard PROFINET IO interface (X150, P1-P3)

- S7 routing to the second PROFINET IO interface with CBE30-2 (X1400, P1-P4)
- S7 routing between the Ethernet interfaces

Routing on SIMOTION D (SINAMICS integrated)

S7 routing to the internal PROFIBUS on SINAMICS Integrated

All SIMOTION D have an integrated SINAMICS drive control. To be able to access drive parameters, the telegrams must be routed from the external SIMOTION D interfaces to the internal PROFIBUS DP. S7 routing can be used to access the integrated PROFIBUS. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.

Additional references

You will find more information on routing and the differences between IP and S7 routing in the *SIMOTION Communication* System Manual.

10.1.1.5 Commissioning (hardware)

Requirements for commissioning

Requirements

The following requirements must be satisfied for the initial commissioning of the SIMOTION D4x5-2 and the SINAMICS S120 modules required for operation (SINAMICS S120 line modules and SINAMICS S120 motor modules):

- Your system with SIMOTION D4x5-2 has been installed and wired.
- Your PG/PC has been connected to the SIMOTION D4x5-2 via the PROFIBUS, Ethernet or PROFINET IO interface.

Commissioning steps

Commissioning the hardware involves the following steps:

1. Inserting the CompactFlash card (Page 6982)
2. Checking the system (Page 6984)
3. Switching on the power supply (Page 6984).

Additional references

For information on installing/mounting and commissioning the SINAMICS S120 components, refer to the *SINAMICS S120* Commissioning Manual.

Inserting the CompactFlash card

Characteristics of the CF card

The CF card is essential for operation of the SIMOTION D4x5-2. The SIMOTION Kernel (SIMOTION D firmware) and the software used to control the drives (SINAMICS firmware) are contained on the CF card.

To load the SIMOTION Kernel, the CF card must be inserted when the SIMOTION D4x5-2 is powered up.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Comply with the ESD rules.

Note

The ramp-up time of the SIMOTION D4x5-2 as well as the time behavior during CF card accesses depends on the type of memory card used. Take this into account in your application, as different memory card types are used for availability reasons.

Procedure

To insert the CF card, perform the following steps:

1. The direction of insertion of the CF card is indicated by an arrow located on both the plug-in slot and the CF card. Align the CF card with the arrows.
2. Gently insert the CF card into the empty plug-in slot of the SIMOTION D4x5-2 until it clicks into place.
   If correctly inserted, the CF card is flush with the housing.

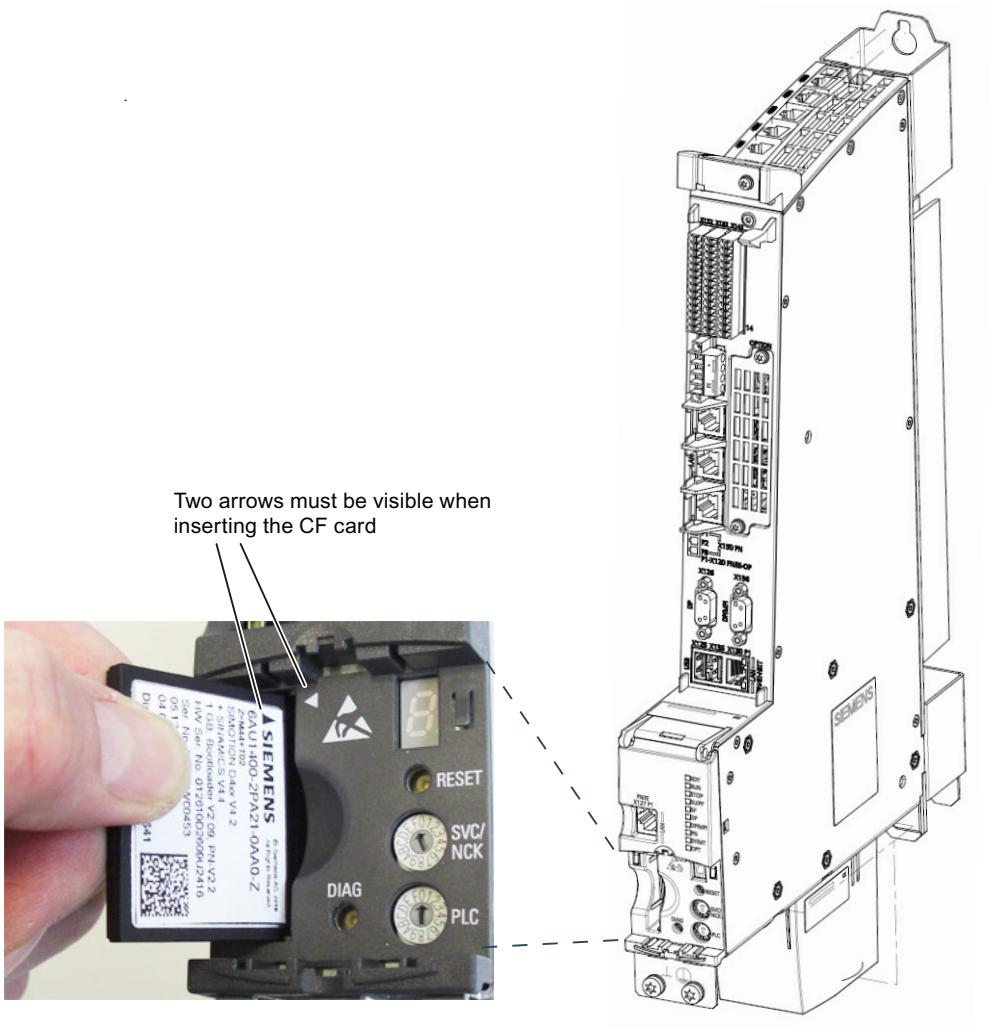


Figure 10-36 Inserting the CF card

Checking the system

Procedure

Check the final installed and wired system one more time before it is switched on, keeping in mind the safety points in the following check list:

- Have you observed all ESD measures when handling the components?
- Have all screws been tightened to their specified torque?
- Have all connectors been properly inserted and locked/screwed?
- Are all components grounded and have all shields been attached?
- Have you taken the load capacity of the central power supply into consideration?

Note

The SIMOTION D4x5-2 Control Units cannot be operated without a fan/battery module. The SIMOTION D4x5-2 Control Units will not power up without a fan/battery module.

Switching on the power supply

Switching on the external power supply

Power is supplied to the SIMOTION D4x5-2 via an external power supply unit, e.g. via a SITOP power supply.

Switch on this power supply.

NOTICE
SIMOTION D4x5-2 shuts down if the power supply is interrupted
It is essential to ensure that the external 24 VDC power supply to the D4x5-2 is not interrupted for longer than 3 ms. After a longer interruption, the SIMOTION D4x5-2 shuts down and can only be restarted with OFF/ON.
You will find further information in Section Properties of the user memory (Page 6987).

Power-up of Control Unit

Once the power supply has been switched on, the SIMOTION D4x5-2 begins to power up:

1. At the start of the power-up, all LEDs are briefly illuminated in yellow for a short LED test. The LEDs on the SIMOTION D4x5-2 enable you to track the progress of the power-up. Any errors are displayed.
2. Startup of the SIMOTION Kernel.

3. All DRIVE-CLiQ connections (e.g. with the SINAMICS S120 Active Line Module) are also detected automatically.

Note

As long as the RDY LED continues to flicker, power-up is not complete and it is not possible to go online.

During commissioning the firmware of the components is upgraded or downgraded automatically based on the firmware version on the CF card and the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, CBE30-2, Power Modules, etc.).

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

SIMOTION D4x5-2/CX32-2/DRIVE-CLiQ components:

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2, and indicate that a firmware update is being performed on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

CBE30-2 option board:

During the firmware update, the OPT LED of the SIMOTION D module and the SYNC LED of the CBE30-2 flash green.

4. The first time it is switched on, the SIMOTION D4x5-2 goes to STOP operating state following power-up.

Following power-up, the SIMOTION D4x5-2 is in a state in which it can be configured.

Fan/battery module defective or not mounted properly on a SIMOTION D4x5-2

While the SIMOTION D4x5-2 is powering up, a test is performed to check whether the fan/battery module is functioning properly. If the fan/battery module is missing or faulty, the kernel is not loaded and the RDY LED flashes red/yellow (2 Hz). Switch off the power supply and correct the fault before switching on the power supply again.

NOTICE
Module enters the reset state If the SIMOTION D4x5-2 Control Unit needs to be cooled (thermostatically controlled fan) and this is not possible due to a fan/battery module having been removed or being faulty, the module state will change to reset after approximately 1 minute. This state can only be reset with a switch-off.

RESET button

Layout

The RESET button is located behind the blanking cover on the SIMOTION D4x5-2.

Performing a reset

A reset causes the entire system to be reset and forces the system to power up again. This operation is comparable with a "Power on Reset" but does not require the disconnection of the 24 V power supply (e.g. in order to exit the F state).

User memory concept

SIMOTION D4x5-2 memory model

The following figure provides an overview of the memory model of SIMOTION D4x5-2.

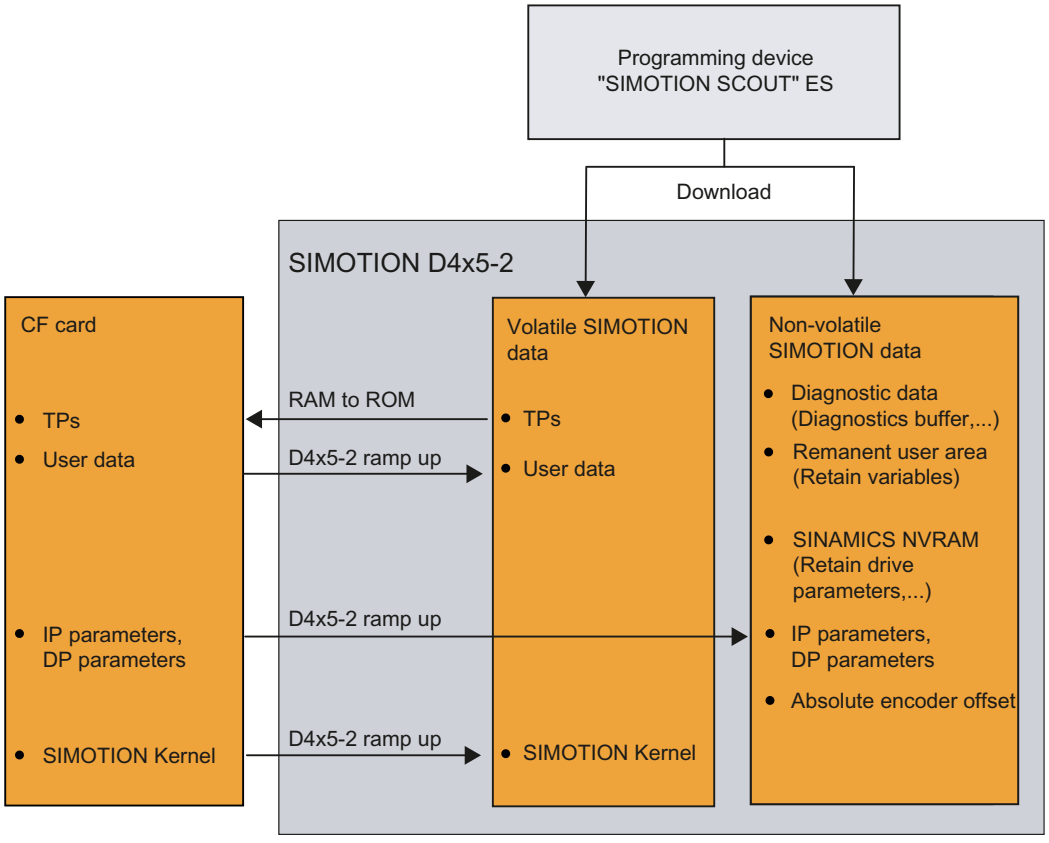


Figure 10-37 SIMOTION D4x5-2 memory model

The SIMOTION Kernel (SIMOTION D firmware, including SINAMICS Integrated Firmware) contains the functions required for virtually all applications, and essentially acts as a PLC with the command set in accordance with IEC 61131-3 as well as system functions for controlling various components, such as inputs and outputs.

The SIMOTION Kernel can be expanded by loading technology packages (TPs), e.g. for motion control or temperature controllers.

In the following sections, you will find information about the user memories and the steps involved in certain operations.

Properties of the user memory

Non-volatile data

Non-volatile data make it possible to retain relevant data for the user and the system even when the SIMOTION D4x5-2 has been switched off. Information about the area which can be used for non-volatile data is available in the *SIMOTION D4x5-2, Technical Data* Manual.

A SIMOTION device has the following non-volatile data:

Table 10-21 Non-volatile data contents

Non-volatile data	Content
Kernel data	• Last operating state • IP parameters (IP address, subnet mask, router address) • DP parameters (PROFIBUS DP address, baud rate) • Diagnostic data (diagnostics buffer,...)
Retain variables	• Variables in the interface or implementation section of a unit declared with VAR_GLOBAL RETAIN • Global device variables set with the "RETAIN" attribute
Retain TO	Absolute encoder offset
DCC blocks	SAV blocks and user-defined blocks with retain behavior ("SAV = SAVE", blocks for the non-volatile data backup).
NVRAM (SINAMICS)	For the SINAMICS Integrated, CX32-2 and SINAMICS S120 CU310-2/CU320-2, data protected against loss at power failure is called NVRAM data or non-volatile data.

Note

DCC SIMOTION blocks with retain behavior act like retain variables in terms of copying RAM to ROM, resetting memory, downloading, backing up non-volatile SIMOTION data (**_savePersistentMemoryData**) and backing up data.

For the SINAMICS Integrated and CX32-2, the data backup of the SINAMICS retain data (and therefore also for SINAMICS DCC) is performed via the NVRAM (D4x5-2) or the FRAM (CX32-2). SINAMICS data is not saved with **_savePersistentMemoryData**. The non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

For further information on DCC, see the *DCC Programming* Programming Manual.

The non-volatile data of the SIMOTION D4x5-2 has the following properties:

Table 10-22 Non-volatile data and real-time clock properties

Properties	Non-volatile data	Real-time clock (RTC)
Location	The non-volatile data is located • For a D4x5-2 in an NVRAM • For a CX32-2 in an FRAM	The real-time clock is backed up maintenance-free via a SuperCap or a battery.
Back-up battery	No	Optional (for longer backup time)
Backup time	There is no maximum backup time	• SuperCap: At least 4 days • Battery: At least 3 years

If the buffer time is exceeded on the real-time clock, the time is reset.

Fan/battery module

If the backup time for the real-time clock (RTC) is not sufficient, a battery can be connected by means of an external fan/battery module. The backup time when a battery is used is at least 3 years.

A 3 V lithium SN: 575332 battery type (with cable tail and connector) is used. The battery can be replaced without data loss, because the real-time clock is backed up internally via the SuperCap.

A fan/battery module is always required for SIMOTION D4x5-2 and therefore is always supplied with the module (see the *SIMOTION D4x5-2* Manual).

CF card

The CF card contains the following data:

- SIMOTION Kernel (SIMOTION D firmware)
- Technology packages (TP)
- User data (units, configuration data, parameter settings, task configuration)
- IP parameters (IP address, subnet mask, router address)
- DP parameters (PROFIBUS DP address, baud rate)

It may also contain:

- User data saved with **_savePersistentMemoryData** and **_export/_saveUnitDataSet**
- SINAMICS non-volatile data (NVRAM data) of the SINAMICS Integrated or a CX32-2 backed up with CU parameter p7775 = 1
- Data from SIMOTION IT
- Archived SCOUT project

With the **_savePersistentMemoryData** system function, the user program can back up the contents of the non-volatile SIMOTION data to the CF card. This ensures that the retain variables and the absolute encoder position are backed up in the event that a spare part is used.

For the SINAMICS Integrated, CX32-2 and SINAMICS S120 CU310-2/CU320-2, the non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

Note

IP and DP parameters in the non-volatile data

If the CF card contains a configuration, the IP and DP parameters are loaded from the CF card during ramp-up and used by the SIMOTION device. The SIMOTION D4x5-2 uses the addresses defined in these parameters to go online. During ramp-up, the IP and DP parameters on the CF card are also written to the non-volatile data. If the SIMOTION device is then powered up with a CF card with no configuration, the IP and DP parameters are retained in the non-volatile data and are used by the device. Thus, the SIMOTION device can continue to go online if a configuration was loaded with SIMOTION SCOUT at least once or if the SIMOTION device is powered up with a CF card containing a configuration.

Volatile SIMOTION data (RAM / current data RAM)

The volatile SIMOTION data is defined by the following properties:

- The volatile SIMOTION data is located in the RAM memory of the SIMOTION device.
- The download data of SIMOTION SCOUT is written to this memory.

- This data is lost when the SIMOTION D4x5-2 is switched off.
- The "volatile SIMOTION data" area contains the following data:
  - SIMOTION Kernel (D4x5-2 firmware)
  - Technology packages (TP)
  - User data (programs, configuration data, parameter settings)

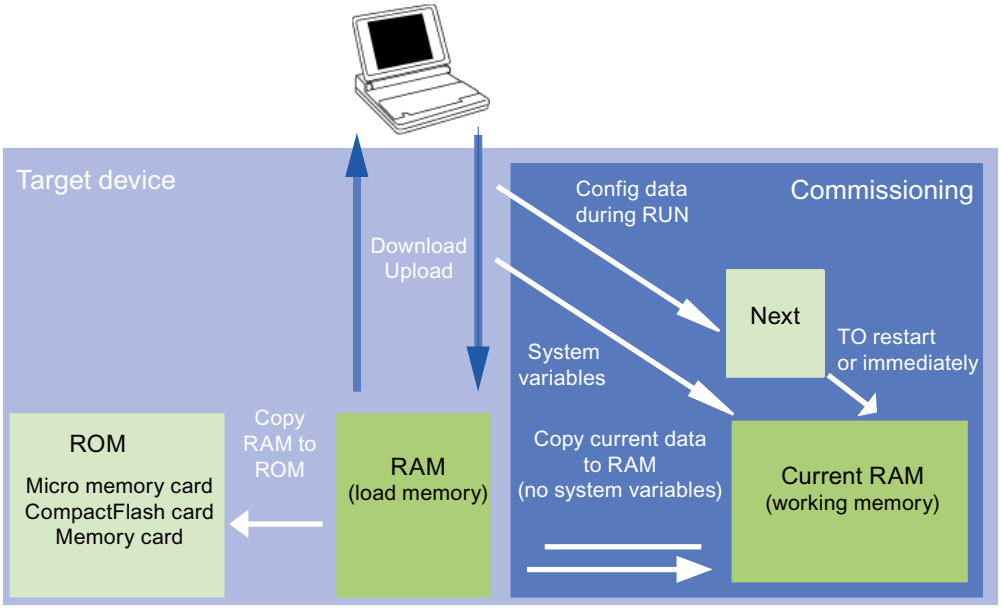


Figure 10-38 Configuration data and system variables in the volatile memory (memory concept, principle)

You can find additional information about memory management in SIMOTION in the *SIMOTION Runtime Basic Functions* Function Manual.

Operations and their effect on the user memory

The operations marked with arrows in the figures entitled "SIMOTION D4x5-2 memory model" and "Configuration data and system variables in the volatile memory" and their effects on the user memory are described below.

SIMOTION SCOUT download

The following data is transferred with "Download project to target system" or "Download CPU / drive unit to target device" from the engineering system to the "volatile SIMOTION data" area:

- User data (units, configuration data, parameter settings, task configuration)
- Technology packages (TPs)

In addition, the IP and DP parameters are saved to the "non-volatile SIMOTION data" area. The retain variables are set to their initial values, but this depends on the settings in SIMOTION SCOUT. If the SIMOTION D4x5-2 is switched off following the download, the volatile SIMOTION data is lost.

Copy RAM to ROM

The Copy RAM to ROM menu command saves the following data via the engineering system to the CF card:

- Technology packages and user data (units, configuration data, parameter assignments, task configuration) from the "volatile SIMOTION data" area
- Current values are copied to the "volatile SIMOTION data" area, depending on the settings in SIMOTION SCOUT.

Note

The "Copy RAM to ROM" menu command does not save the current values of the retain variables to the CF card. Use the system function **_savePersistentMemoryData** for this purpose.

Note

The "Copy RAM to ROM" function is also available for drive units and saves the volatile SINAMICS data to the non-volatile memory (CF card).

Current RAM

If you change the system variable values, these take immediate effect in the current RAM. New configuration data values are initially stored in the Next memory. Configuration data that takes immediate effect is automatically transferred to the current RAM. Configuration data that will only become active following a RESTART on the technology object (set the **restartactivation** system variable to value ACTIVATE_RESTART) is only written to the current RAM once the RESTART has taken place.

To save the configuration data changed online to the offline project, you must first transfer the content of the current data RAM to the RAM using the menu command "Target system" > "Copy current data to RAM".

Once you have done this, the configuration in SCOUT will no longer be consistent with the configuration in the target device, as a consistency check is performed on the RAM data. Read the data from the RAM using the menu command "Target system" > "Load" > "Load CPU / drive unit to PG" (for the configuration data only) to re-establish a consistent system state.

Use the "Target system" > "Copy RAM to ROM" menu command to save the configuration to the non-volatile memory on the CF card.

Note

The "Copy current data to RAM" command does not transfer the values of the system variables to the RAM memory. This means that "Save to memory card (Copy RAM to ROM)" or "Save in the engineering project (Load CPU / drive unit to PG)" is not possible.

In order to ensure that system variable values can also be saved to the engineering project and memory card, the system variable values must be changed OFFLINE and then downloaded to the target device and saved.

SIMOTION D4x5-2 power-up

During power-up of the SIMOTION D4x5-2, the SIMOTION Kernel is loaded from the CF card to the "volatile SIMOTION data" area.

When the SIMOTION D4x5-2 is switched off, the contents of the "volatile SIMOTION data" area are lost. When the unit is powered up again, the following data is loaded from the CF card:

- Technology packages and user data to the "volatile SIMOTION data" area
- IP and DP parameters to the "non-volatile SIMOTION data" area

Backing up non-volatile SIMOTION data

You have the following options for backing up non-volatile SIMOTION data on the CF card:

- In the user program:
  With the **_savePersistentMemoryData** system function, the user program can back up the non-volatile SIMOTION data content to the CF card. This ensures that the retain variables and the absolute encoder position are backed up in the event that a spare part is used. The contents are saved to the "PMEMORY.XML" backup file in the "USER\SIMOTION" directory.
- Via switch/button (service selector switch or DIAG button of the SIMOTION D4x5-2) or SIMOTION IT web server. See Section Backing up diagnostic data and non-volatile SIMOTION data (Page 7266). The contents are saved to the "PMEMORY.XML" backup file in the "USER\SIMOTION\HMI\SYSLOG\DIAG" directory.

On the system side, this system function ensures that a consistent overall image of the non-volatile SIMOTION data is always available the next time the unit is switched on, even if there is a power failure during backup. An already existing backup file is renamed to "PMEMORY.BAK" before a new backup file is generated. If the save operation to the new backup file fails (e.g. because the capacity of the CF card is insufficient), this backup copy of the backup file is used the next time an attempt is made to restore the non-volatile SIMOTION data content.

NOTICE
Loss of data due to failure to make backup copies Non-backed-up non-volatile SIMOTION data can be lost on hardware replacement (module defect), for example, if the current values of the retain variables have not been backed up and replaced by their initial values again. Back up the non-volatile SIMOTION data on the CF card.

NOTICE
Homing required again after absolute encoder overflow If an absolute encoder overflow occurs after **_savePersistentMemoryData**, the actual position value is no longer correct after the non-volatile SIMOTION data is restored. In this case, homing (absolute encoder adjustment) must be repeated.

With the SCOUT functions "Save variables" and "Restore variables," you also have the option to back up and restore data to your PC and restoring data that was changed during operation and only stored in the runtime system.

Restoring non-volatile SIMOTION data

SIMOTION data backed up on a CF card with **_savePersistentMemoryData** is restored in the following cases:

1. After a module replacement (spare part scenario), see Section Replacing modules in the spare part scenario (Page 6998)
2. After a general reset, see Section SIMOTION D4x5-2 general reset (Page 7186)
3. Via switch position, see Section Deleting/restoring non-volatile SIMOTION data (Page 7274).

Backing up / restoring non-volatile SINAMICS data

Requirement: As of SIMOTION V4.3 / SINAMICS V4.5

For the SINAMICS Integrated, CX32-2 and SINAMICS S120 CU310-2/CU320-2, as of SINAMICS firmware version V4.5, the non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

Restoring

- Is performed automatically in the event of a module replacement
  A module replacement is detected on the basis of the serial number.
- Can be performed manually
  Restoring can be initiated manually by setting the CU parameter p7775 to 2.

For more information, see Section Backing up / restoring / deleting SINAMICS NVRAM data (Page 7110).

Power failure

With a power failure, the real-time clock is backed up by an internal SuperCap and, if available, a battery in the fan/battery module.

The non-volatile data is backed up on the D4x5-2 permanently and maintenance-free in an NVRAM. In this way, the Control Unit is immediately ready for operation without data loss after a power failure.

Power-up and non-volatile SIMOTION data

The table below lists the cases that can arise during power-up in conjunction with the non-volatile SIMOTION data and explains how they are handled.

Table 10-23 Power-up scenarios for non-volatile SIMOTION data

Case	Initial condition	Result
1	The non-volatile SIMOTION data is valid.	SIMOTION D4x5-2 powers up with the non-volatile SIMOTION data, i.e. with the PROFIBUS address in the non-volatile data.
2	The non-volatile SIMOTION data is invalid and there is no backup file (PMEMORY.XML) and no backup copy of the backup file (PMEMORY.BAK).	SIMOTION D4x5-2 copies the default settings to the non-volatile SIMOTION data and powers up with this data. In this case, for example, the default PROFIBUS address is used.

Case	Initial condition	Result
3	The non-volatile SIMOTION data is invalid, backup file (PMEMORY.XML) exists and is valid.	SIMOTION D4x5-2 copies the backup file contents to the non-volatile SIMOTION data and powers up with this data.
4	The non-volatile SIMOTION data is invalid, the backup file is invalid and there is no backup copy of the backup file (PMEMORY.BAK).	SIMOTION D4x5-2 copies the default settings to the non-volatile SIMOTION data and powers up with this data, in which case, for example, the default PROFIBUS address is used.
5	The non-volatile SIMOTION data is invalid; a backup file exists, but it is invalid; a backup copy of the backup file exists and is valid.	SIMOTION D4x5-2 copies the backup file contents to the non-volatile SIMOTION data and powers up with this data.

Non-volatile SIMOTION data diagnostics

The user can determine the state of the non-volatile SIMOTION data and the battery using the diagnostic buffer, system variables, and PeripheralFaultTask.

Evaluating via the diagnostic buffer

When they are issued, the following messages are entered once in the diagnostic buffer:

Table 10-24 Messages of the diagnostic buffer

Entry	Meaning	Remedy
Level 1 battery voltage warning [1)]	Battery voltage below prewarning level.	Replace battery in the fan/battery module
Level 2 battery voltage warning [1)]	The battery voltage is below the warning level, backup of real-time clock (RTC) can no longer be guaranteed.	Replace battery in the fan/battery module
Battery voltage for data backup in permissible range	The battery voltage is in the permissible range for the backup of the real-time clock (RTC).	
Non-volatile data memory voltage error	The buffer voltage of the SuperCap or the battery was too low after switching on. Possible data loss in the real-time clock.	Replace battery in the fan/battery module or charge the SuperCap over a longer period.
Non-volatile data loaded from a file (Persistent Data File Loading done)	Non-volatile SIMOTION data has been successfully restored from the backup file on the CF card.	-
Non-volatile data loaded from the backup file (Persistent Data Backup File Loading done)	Non-volatile SIMOTION data has been successfully restored from the backup copy of the backup file on the CF card.	-
Error while loading non-volatile data from a file (Persistent Data File Loading failure)	Backup file or backup copy could not be loaded. Possible causes: • Backup file or backup copy not available • Invalid data in backup file	Use the **_savePersistentMemoryData** system function to generate a backup file with valid contents.
Device with battery module	Fan/battery module is present.	-

Entry	Meaning	Remedy
Device without battery module	Fan/battery module is not present.	Connect fan/battery module
Module replacement detected - NVRAM has been initialized	A module replacement has been detected on the basis of the serial number. The non-volatile SIMOTION data on the controller will be deleted and the data from the CF card transferred to the controller.	-
Module replacement not detected - NVRAM has not been initialized	An error has occurred. The non-volatile SIMOTION data on the controller will not be deleted.	Possible causes: • Incorrect controller type • File system of the CF card corrupt

[1] These warnings are only signaled when the fan/battery module has been inserted.

Refer to the *SIMOTION SCOUT* Configuration Manual for how to read out the contents of the diagnostic buffer.

Evaluating via PeripheralFaultTask

Battery state changes in RUN are reported to the user program by calling PeripheralFaultTask. Changes can be evaluated there using Taskstartinfo:

- TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205)
- TSI#details = 16#00000040

If no battery is used during the transition of the operating state to RUN, no PeripheralFaultTask will be triggered (application case: D4x5-2 should generally be operated without a battery).

References

You will find detailed information on setting up Taskstartinfo (#TSI) in the *SIMOTION Runtime Basic Functions* Function Manual.

Evaluating via system variables

The system variables in the **device.persistentDataPowerMonitoring** structure indicate the state of the non-volatile SIMOTION data and the battery.

Table 10-25 State of the non-volatile SIMOTION data and battery

System variable	Designation	State	Updating
powerFailure	Buffer voltage (SuperCap or battery) too low, possible data loss of the real-time clock	NO (91) YES (173)	"YES" was set at too low a buffer voltage when powering up; status needs to be reset to "NO" via the application; the state remains even after power off/on.
rtcFailure (as of V4.3)	Indicates that the clock contents (RTC) are invalid (clock must be set again)	NO (91) YES (173)	Is updated once during power-up; status must be reset to "NO" via the application; the state is retained even after power off/on.
retainDataFailure (as of V4.3)	Indicates a checksum error of the non-volatile SIMOTION data; can be an indication of defective HW	NO (91) YES (173)	Is updated once during power-up; status must be reset to "NO" via the application; the state is retained even after power off/on.
persistentDataState	Reading the persistent data	See the table below, "State of non-volatile data following power-up."	During power-up
warningBatteryVoltage Level1 [1)]	Battery voltage below the pre-warning level	NO (91) YES (173)	During a state change, remains set if Level 2 is reached
warningBatteryVoltage Level2 [1)]	Battery voltage below the warning level	NO (91) YES (173)	During a state change

1) **Both** battery warning levels are set under the following conditions:
- The fan/battery module is present and the battery has been inserted, but the battery voltage is below warning level 2
- The fan/battery module is present, but no battery has been inserted
- The fan/battery module is not present (not permissible on D4x5-2)

The system variable **device.persistentDataPowerMonitoring.powerFailure = YES** indicates that the buffer voltage of the SuperCap or the battery was too low after switching on.
With .**powerFailure = YES** and **.persistentDataState = FROM_RAM**, only a possible data loss of the real-time clock displayed for the SIMOTION D4x5-2 (and not also a possible loss of the non-volatile SIMOTION data as for the SIMOTION D4x5). The non-volatile SIMOTION data is not lost, but the data from the NVRAM will still be used.

As of SIMOTION V4.3, a data loss of the real-time clock is signaled via the system variable **device.persistentDataPowerMonitoring.rtcFailure = YES**.

The system variable **device.persistentDataPowerMonitoring.persistentDataState** indicates the state of the non-volatile SIMOTION data after power-up.

Table 10-26 State of the non-volatile SIMOTION data after powering up (**persistentDataState** system variable)

State	Meaning
FROM_RAM (1)	Non-volatile SIMOTION data in the SIMOTION device is used
FROM_FILE (2)	Non-volatile SIMOTION data is restored from the backup file
FROM_BACKUP (3)	Non-volatile SIMOTION data is restored from the backup copy of the backup file
INVALID (4)	Data in the non-volatile SIMOTION data and in the backup file / backup copy of backup file is invalid or non-existent/deleted. The SIMOTION device has copied the default settings to the non-volatile SIMOTION data and used this data to power up.

Requirement/availability of the battery

System variables can be used to evaluate:

- Whether a battery is required for the operation of the device (or not)
- Whether a battery is available (or not)

Table 10-27 System variable batterynecessary/batteryexisting

System variable on the device	States	Description
fanbattery of data type StructDeviceFanBattery (the system variables are of the data type EnumFanBattery)		
.batterynecessary	MANDATORY	Battery is required for the backup of the non-volatile data and for the real-time clock (RTC) of the device. **.batteryexisting** can be used to query whether a battery is installed.
	OPTIONAL	The non-volatile data and the real-time clock (RTC) are backed up via SuperCap. A battery can be used as an option to extend the backup time. **.batteryexisting** can be used to query whether a battery is installed. **Example: D4x5**
	OPTIONAL_RTC [1)]	A battery is not required for backing up the non-volatile data. Only the real-time clock (RTC) is backed up via SuperCap. A battery can be used as an option to extend the backup time of the real-time clock. **.batteryexisting** can be used to query whether a battery is installed. **Example: D4x5-2**
	NOT_MANDATORY [1)]	A battery is not required for backing up the non-volatile data. The real-time clock (RTC) is backed up via SuperCap. **Example: D410-2**

System variable on the device	States	Description
.batteryexisting [2]	EXISTING	EXISTING is only displayed when **.batterynecessary** is set to: • MANDATORY or • OPTIONAL or • OPTIONAL_RTC and a battery is installed.
	NOT_EXISTING	Battery is not available.

[1] If the SuperCap is discharged, the contents of the real-time clock (RTC) are lost.

[2] The value for SIMOTION D4x5-2 is updated dynamically.

Replacing modules in the spare part scenario

SIMOTION module replacement

A module replacement is detected automatically as of SIMOTION V4.3.

During a module replacement, a CF card that contains the non-volatile SIMOTION data backed-up with **_savePersistentMemoryData**, is inserted in a new device of the same type.

A module replacement is detected by the SIMOTION D4x5-2 on the basis of the serial number. The data backed up on the CF card with **_savePersistentMemoryData** is then automatically transferred to the new device.

Note

As an additional option, you can back up the non-volatile data by setting the service selector switch, with the DIAG button or via SIMOTION IT web server, see Section diagnostic data and non-volatile SIMOTION data (Page 7266).

Initial power-up with the CF card

Requirement: CF card (no device serial number stored)

If the SIMOTION D4x5 2 powers up successfully with a new CF card, the serial number of the device is stored on the CF card. In this case, a module replacement cannot be detected.

Note

The serial number stored on the CF card remains unaffected by the following actions:

- Copy RAM to ROM
- Project download
- Writing the CF card via the SCOUT function "Load to file system"
- FW/project update via device update tool

If the CF card contents are copied to another CF card, the serial number is also copied.

A serial number stored on the CF card can only by removed by deleting the CF card contents.

SIMOTION D4x5-2 power-up with the CF card (no module replacement)

Requirement:

- The same CF card
- The same controller

If the device powers up successfully, the serial number stored on the CF card is compared with the serial number of the device.

If the **serial numbers are identical**, there has been no module replacement.

The device powers up. In the non-volatile SIMOTION data present in the device is valid, then this is used (for details, see Table 10-23 Power-up scenarios for non-volatile SIMOTION data (Page 6993)).

SIMOTION D4x5-2 power-up with the CF card (module replacement)

Requirement:

- The same CF card
- Different device (e.g. replacement because of fault)

If the device powers up successfully, the serial number stored on the CF card is compared with the serial number of the device.

If the **serial numbers are not identical**, there has been a module replacement.

This means:

- The serial number of the new module is stored on the CF card.
- The non-volatile SIMOTION data are deleted in the device.
- A diagnostic buffer entry is issued which signals that a module has been replaced.
- The non-volatile SIMOTION data stored "on the CF card" is transferred to the device (for details, see Table 10-23 Power-up scenarios for non-volatile SIMOTION data (Page 6993)).

NOTICE
Irrevocable data deletion due to incorrect CF card with stored device serial number
A module replacement is detected only on the basis of the changed serial number.
Inserting the wrong CF card with saved device serial number has the following consequences: • The non-volatile data on the device is permanently deleted. • The IP/DP address set in the device is deleted. You can no longer go online via the IP/DP address set originally.
Make sure that the correct CF card is inserted into the SIMOTION D4x5-2.

Error scenarios

In the event of an error, a diagnostic buffer entry signals that it was not possible to determine whether a module has been replaced.

Possible reasons are:

- The serial number of the device cannot be determined.
- The serial number saved on the CF card cannot be determined, for example, due to a corrupt file system.
- The controller has not powered up.
- The new serial number could not be transferred to the CF card, for example, due to a corrupt file system.

Restart after reloading

The system variable **device.startupData.operationMode** is used to define whether the SIMOTION D4x5-2 control unit goes into the RUN state or the last operating state after a power-on/restart.

Possible values of **device.startupData.operationMode**:
LAST_OPERATION_MODE [0] (default setting)
RUN [1]

With LAST_OPERATION_MODE [0], the module remains in the STOP state after reloading the non-volatile SIMOTION data and must be switched manually to the RUN state with SCOUT, the web server, or the mode selector.

With RUN [1], the module goes automatically into the RUN state after reloading.

SINAMICS module replacement

A module replacement is detected as of SINAMICS V4.5.

For the SINAMICS Integrated, CX32-2 and SINAMICS S120 CU310-2/CU320-2, a module replacement is also identified via the serial number. Non-volatile SINAMICS data (NVRAM data) backed up previously via the CU parameter p7775 on the CF card, is then automatically transferred to the control unit.

Fan

Cooling the SIMOTION D4x5-2

Overview

A fan/battery module is always required for cooling the SIMOTION D4x5-2 Control Unit.

Table 10-28 Fan/battery module for SIMOTION D4x5-2

Property	SIMOTION D425-2 DP SIMOTION D425-2 DP/PN	SIMOTION D435-2 DP SIMOTION D435-2 DP/PN	SIMOTION D445-2 DP/PN SIMOTION D455-2 DP/PN
Fan/battery module	Always required (double fan/ battery module included in the D425-2's scope of delivery)	Always required (double fan/ battery module included in the D435-2's scope of delivery)	Always required (double fan/ battery module included in the scope of delivery of the D445-2 DP/PN and D455-2 DP/PN)
Fan/battery module included in the D4x5-2 scope of delivery	Double fan/battery module 6FC5348-0AA02-0AA0	Double fan/battery module 6FC5348-0AA02-0AA0	Double fan/battery module 6FC5348-0AA02-0AA0
Usable fan/battery modules	Only type 6FC5348-0AA02-0AA0 (double fan)	Only type 6FC5348-0AA02-0AA0 (double fan)	Only type 6FC5348-0AA02-0AA0 (double fan)
Max. permissible supply air temperature	55° C	55℃	55° C
Fan control	Temperature-controlled fan unit will be switched on depending on supply air temperature and CPU load		

The fan/battery modules are usually supplied with a backup battery.

Fan faults

Fan faults are indicated as follows:

- Entry in diagnostics buffer
- In the event of fan failure (both fans in the double fan/battery module failed) the RDY LED flashes red/yellow at 2 Hz
- Indicated via system variable
- Call to the PeripheralFaultTask

For further information on the evaluation of fan faults, see Section Overview of the fan/battery module states (Page 7002).

Fan/battery module

A fan/battery module is always required for operation of the SIMOTION D4x5-2 Control Unit. A corresponding module including a backup battery is therefore included in the scope of delivery of the SIMOTION D4x5-2 Control Unit.

For increased availability a double fan/battery module is used as standard on SIMOTION D4x5-2.

The double fan/battery module also guarantees sufficient cooling with just one functional fan.

If one of the fans fails, the remaining fan continues under full load. The fan failure is signaled by the generation of an event in the PeripheralFaultTask. In this case it is strongly recommended that the double fan/battery module is replaced at the next available opportunity.

Checks are performed during power-up to determine whether the double fan/battery module on the SIMOTION D4x5-2 Control Unit is functioning correctly (i.e. at least one of the two fans is working). If a double fan/battery module is missing or faulty, the kernel is not downloaded and the RDY LED flashes red/yellow (2 Hz). Switch off the power supply and eliminate the fault. Then switch the power supply back on.

Note

If the SIMOTION D4x5-2 needs to be cooled (fan switches on with temperature control) and this is not possible due to a fan/battery module having been removed or being faulty, the module state will change to the RESET state after approximately 1 minute (7-segment display shows "8"). This state can only be reset with a switch-off.

If only one fan in the double fan/battery module fails, the module is still regarded as functional, although it should be replaced as soon as possible. If neither of the fans is working, the double fan/battery module is identified as defective.

Fan test

Fan faults can only be detected by the control unit if the double fan is switched on (fan not turning or turning too slowly).

To detect failure of a fan in good time, the Control Unit performs a fan test:

- The Control Unit runs a fan test on power-on (short switch-on of the double fan)
- During operation, the double fan is switched on briefly at cyclic intervals (even if not necessitated by the temperature).

Overview of states, fan/battery module

The states that can occur during operation are described in the following.

Table 10-29 Overview of states

State	Fan [4]	PeripheralFaultTask Diagnostic buffer entry	System variables [3]	
			_cpuDataRW. fanWarning	_cpuDataRW. redundantfanWarning [2]
Failure of one fan during STOP, then RUN	Single fan	PeripheralFaultTask: Is not called Diagnostic buffer entry: Fan on the module is defective	= YES	= NO
	Double fan	PeripheralFaultTask: Is not called Diagnostic buffer entry: There is no redundant fan available	= NO	= YES

State	Fan [4]	PeripheralFaultTask Diagnostic buffer entry	System variables [3]	
			_cpuDataRW. fanWarning	_cpuDataRW. redundantfanWarning [2]
Failure of both fans during STOP, then RUN	Double fan	PeripheralFaultTask: Is not called Diagnostic buffer entry: Fan on the module is defective	= YES [1]	= NO
Failure of one fan during RUN	Single fan	PeripheralFaultTask: TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) TSI#details = 16#00000080 Diagnostic buffer entry: Fan on the module is defective	= YES	= NO
	Double fan	PeripheralFaultTask: TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) TSI#details = 16#00004000 Diagnostic buffer entry: There is no redundant fan available	= NO	= YES
Failure of both fans during RUN	Double fan	PeripheralFaultTask: TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) TSI#details = 16#00000080 Diagnostic buffer entry: Fan on the module is defective	= YES [1]	= NO

1) SIMOTION D4x5-2 switches to the RESET state after approximately 1 minute.

2) Maintenance: Replace double fan/battery module at the next available opportunity.

3) The "YES" value must be reset to "NO" by the application.

4) Single fans are only supported by SIMOTION D4x5 and D410-2.

Fan faults are detected if through

- A cyclic fan test
- Or when the fan is switched on

a malfunction is detected (fan does not turn or fan turns at too low a speed).

Note

If the "redundant fan" is already defective in a double fan/battery module when the system is powered up, this is detected as a single fan/battery module depending on the type of defect (e.g. a wire breakage).

You should therefore also set the **.fanexisting** system variable to the "SINGLE" state to alert the personnel operating the machine to the fact that **no is redundancy available**.

Diagnostic buffer entry

The diagnostic buffer entries have the following meanings:

- Fan on the module is defective
  All of the fans in a fan/battery module have failed
- There is no redundant fan available
  This message only occurs in the case of modules that support double fan/battery modules with a redundant fan.
  The message occurs:
  - If it is only the redundant fan that has failed on a double fan/battery module or
  - If a single fan/battery module is connected but the module also supports a double fan/battery module in principle (this is only the case for D445-1; with D4x5-2, only double fan/battery modules are permitted)

Requirement for/presence of a fan

System variables can be used to evaluate:

- Whether a fan is required for the operation of the device (or not)
- Whether a fan is installed (or not)

Table 10-30 System variable fannecessary/fanexisting

System variable on the device	States	Description
fanbattery of data type StructDeviceFanBattery (the system variables are of the data type EnumFanBattery)		
.fannecessary	**MANDATORY**	Fan is required for operation of the device. **.fanexisting** can be used to query whether a fan is installed. **Examples: D410-2, D4x5-2**
	OPTIONAL	Fan can be used optionally. **.fanexisting** can be used to query whether a fan is installed. **Examples: D425, D435**
	NOT_MANDATORY	Fan is not required for operation of the device.
.fanexisting [1]	**SINGLE**	Single fan is available. **Examples: D410-2**
	REDUNDANT	Double fan is available. **Examples: D4x5-2**
	NOT_EXISTING	No fan available. **Example: D425 and D435 without optional fan**

[1] Value for SIMOTION D4x5-2 is updated dynamically

References

You will find detailed information on setting up Taskstartinfo (#TSI) in the *SIMOTION Runtime Basic Functions* Function Manual.

Response to overtemperature

Operation at overtemperature reduces the module service life and can result in damage to the module.

Causes

Cause of problems in the heat dissipation of the module can be, for example:

- Violation of the maximum permissible air intake temperature
- Free convection is not ensured (clearances are not maintained, pollution, convection is prevented by cables)
- Impermissible mounting position of the module

Temperature thresholds

The internal module temperature is monitored via two module-specific temperature thresholds:

- Overtemperature is signaled when the first (lower) temperature threshold is exceeded.
- When the temperature falls below the first temperature threshold again (minus a hysteresis of approx. 5° C), "Normal temperature" is signaled.
- When the second (higher) temperature threshold is exceeded, the module shuts down in order to protect itself.

The internal module temperature is available in system variable **_cpuData.moduletemperature** as of V4.4.

Response to overtemperature

Table 10-31 Response of the temperature monitoring

Temperature...	Response
... exceeds the 1st temperature threshold (overtemperature)	Call of the PeripheralFaultTask: • TSI#InterruptI = _SC_PC_INTERNAL_FAILURE (= 205) • TSI#details = 16#00000002 Diagnostic buffer entry: "Temperature exceeded in the housing"
... falls below the 1st temperature threshold minus a hysteresis of approx. 5° C	Call of the PeripheralFaultTask: • TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) • TSI#details = 16#00000004 Diagnostic buffer entry: "Temperature in the housing has returned to normal"
... exceeds the 2nd temperature threshold	Module goes into permanent RESET to protect itself (7-segment display shows "8", RDY LED flashes red/yellow at 2 Hz). Diagnostic buffer entry (as of V4.3): "Temperature in the housing too high, self-protection function of the module activated"

10.1.1.6 Parameter assignment / addressing

Software requirements

SIMOTION SCOUT engineering system

The following SCOUT Version must be must be installed on your PG/PC for the commissioning of the SIMOTION D4x5-2:

- As of V4.2 for the SIMOTION D445-2/D455-2/CX32-2
- As of V4.3 for the SIMOTION D425-2/D435-2/CBE30-2

Please note the information on the latest *SIMOTION SCOUT* DVD.

For information on how to install SIMOTION SCOUT on your PG/PC, see the *SIMOTION SCOUT* Configuration Manual.

Note

The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 Version V5.x.

Information of configuration of the SIMOTION D Control Units in the Engineering Framework Totally Integrated Automation Portal (SCOUT in the TIA Portal), you will find in the configuration manual *SIMOTION SCOUT TIA*.

The TIA Portal requires at least SIMOTION SCOUT V4.4 and SIMOTION D4xx-2 Control Units as of firmware V4.3.

Creating a project and configuring the communication

Creating a SIMOTION project and inserting a D4x5-2

Procedure

Proceed as follows to create a new project in SIMOTION SCOUT and insert a SIMOTION D4x5-2:

1. Select the Project > New... menu command.
2. In the "New Project" dialog box, assign a name and confirm with "OK."
   A new folder with the name of the project will be created in the project navigator.
3. In the project navigator, double-click "Insert SIMOTION device". The "Insert SIMOTION Device" dialog box is opened.

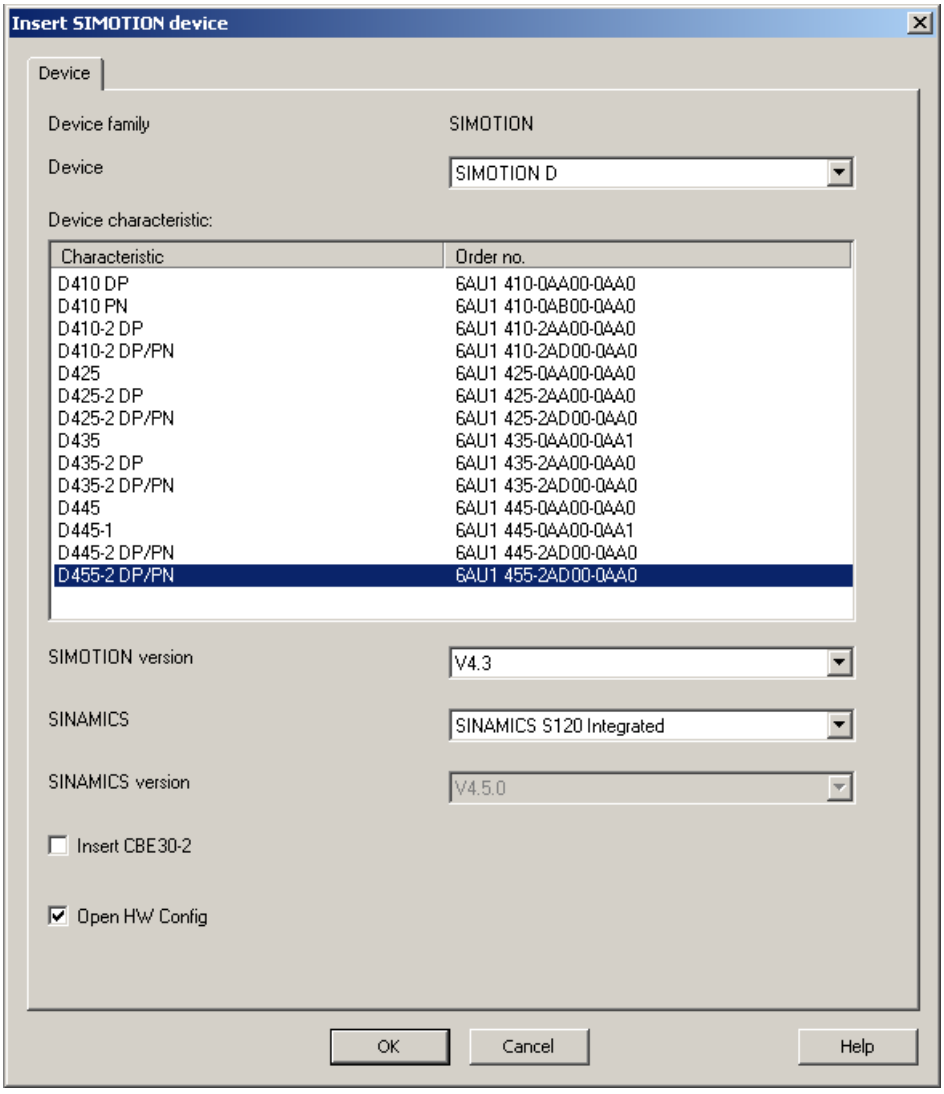


Figure 10-39 Insert SIMOTION device

4. In the "Insert SIMOTION Device" dialog box, select the device, its version and the SIMOTION version.
5. If required, make further settings:
   - SINAMICS: Select either a "SINAMICS S120 Integrated" or a "SINAMICS SM150 Integrated" (only for D445/D445-1/D455-2)
   - SINAMICS version: Select the SINAMICS Integrated version if several drive versions are available for a SIMOTION version
   - Insert a CBE30-2: Selection of a CBE30-2 (only for D4x5-2 DP/PN)
6. The "Open HW Config" option can be used to select whether HW Config should be opened in the next step (e.g. in order to insert a CX32-2 controller extension).
7. Confirm the "Insert SIMOTION Device" dialog with "OK".

SINAMICS Integrated type

The following selection options are available for SIMOTION D445/D445-1/D455-2:

- SINAMICS S120 Integrated
- SINAMICS SM150 Integrated for applications with medium-voltage converters

A separate SIMOTION D firmware is available for each of the two versions.

Version of the SINAMICS Integrated

Depending on the selected SIMOTION version, several versions are available for the SINAMICS Integrated. Please note that a separate SIMOTION D firmware is available for each version of the SINAMICS Integrated.

Configuring the PROFINET interface

After you have acknowledged the "Insert SIMOTION Device" dialog with "OK", the "Properties - Ethernet Interface" dialog box opens in the case of a D4x5-2 DP/PN.

If you are using the PROFINET interface, set the interface properties in the "Properties - Ethernet Interface" dialog box.

To this end, proceed as follows:

1. Click the "New" button.
   The "New Subnet Industrial Ethernet" dialog box opens. Rename the new subnet, or accept the default name by clicking "OK".
2. Select the new Ethernet subnet which is now displayed in the "Properties - Ethernet Interface" dialog box.
3. Enter the required addresses in the "IP address" and "Subnet mask" fields of the "Properties - Ethernet Interface" dialog box. Change to the "Network node" field and define whether you are going to use a router and, if yes, enter the router address. Confirm with "OK".

Result

If you have not yet configured a PG/PC in your project, you can select the interface for the PG/PC connection now.

Configuring the PROFIBUS PG/PC interface

Requirements

The following requirements must be satisfied in order to configure the PG/PC interface:

- You have completed the "Insert SIMOTION Device" dialog box with "OK".
- A PG/PC has not yet been configured in the project.

If these requirements have been satisfied, you can configure the interface for the PG/PC connection in the "Interface Selection - D4x5" dialog box.

Proceed as follows to configure the PROFIBUS DP interface:

Procedure

1. In the "Interface Selection - D4x5" dialog box, select the entry "PROFIBUS DP/MPI (X136)".

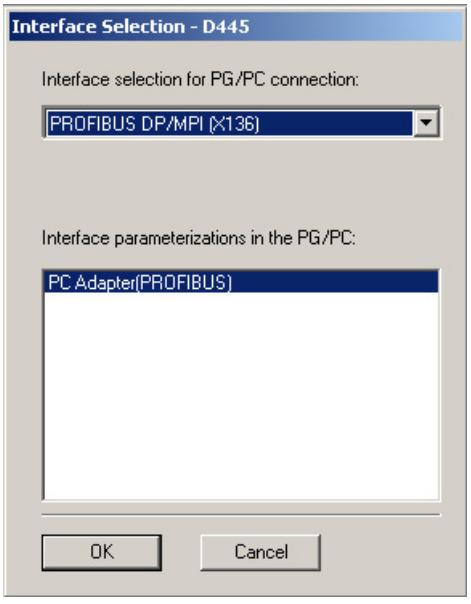


Figure 10-40 Selecting a PROFIBUS interface

2. Select the interface parameter assignment that you would like to use to go online, and confirm with "OK".

The dialog box is closed, the SIMOTION D4x5-2 is created in the project navigator and **HW Config** is started automatically (if parameterized).

A PROFIBUS subnet with factory settings (1.5 Mbit/s transmission rate) is created automatically. The PG/PC is now connected to the SIMOTION D4x5-2 via PROFIBUS. You can configure and parameterize your system.

Note

If you do not use the factory settings, you must configure the PROFIBUS interfaces in **HW Config**. Please make sure that the S7 online access has been activated (PG/PC connection must be yellow and bold in NetPro).

Inserting a further SIMOTION device

If you insert a further SIMOTION device using "Insert SIMOTION device", the PG/PC interface selection dialog box is not displayed. A further SIMOTION device is automatically connected to the PG/PC via PROFIBUS and a new unique IP address (address 4, 5, etc. until 125 is reached) is calculated.

Additional references

Further information on the topic of "Going online" can be found:

- In the online help via the "Contents" tab at
  - "Diagnostics" > "Overview of service and diagnostics options" > "Part III" > "Go online"
  - "Insert device and connect to target system" > "Go online/offline"
- On the Internet at (https://support.industry.siemens.com/cs/ww/en/view/22016709)
- In SIMOTION Utilities & Applications, FAQ "Online connections to SIMOTION devices"

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

See also

Establishing a PG/PC assignment (Page 7022)

Configuring the Ethernet PG/PC interface

Requirement

The following requirements must be satisfied in order to configure the PG/PC interface:

- You have completed the "Insert SIMOTION Device" dialog box with "OK".
- A PG/PC has not yet been configured in the project.

If these requirements have been satisfied, you can configure the interface for the PG/PC connection in the "Interface Selection - D4x5" dialog box.

Proceed as follows to configure the Ethernet interface:

Procedure

1. In the "Interface Selection - D4x5" dialog box, select the entry "Ethernet PNxIE (X127)".

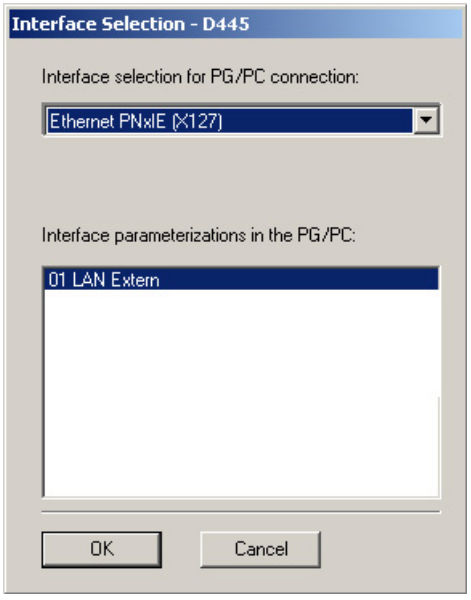


Figure 10-41 Configuring the Ethernet interface

2. Select the interface parameter assignment that you would like to use to go online, and confirm with "OK".

The dialog box is closed, the SIMOTION D4x5-2 is created in the project navigator and **HW Config** is started automatically (if parameterized).

An Ethernet subnet with factory settings is created automatically. (Factory settings, see Section Properties of the Ethernet interfaces (Page 7040).)

The PG/PC is now connected to the SIMOTION D4x5-2 via Ethernet. You can configure and parameterize your system.

Note

If you want to change the default settings for IP addresses and the transmission rate, you must configure the Ethernet interfaces in **HW Config** and **NetPro**. Please make sure that the S7 online access has been activated (PG/PC connection must be yellow and bold in NetPro).

Inserting a further SIMOTION device

If you insert a further SIMOTION device using "Insert SIMOTION device", the PG/PC interface selection dialog box is not displayed. The second SIMOTION device is automatically connected to the PG/PC via Ethernet and a new unique IP address (last digit + 1, up to 255) is calculated.

Additional references

Further information on the topic of "Going online" can be found:

- In the online help via the "Contents" tab at
  - "Diagnostics" > "Overview of service and diagnostics options" > "Part III" > "Go online"
  - "Insert device and connect to target system" > "Go online/offline"
- On the Internet at (https://support.industry.siemens.com/cs/ww/en/view/22016709)
- In SIMOTION Utilities & Applications, FAQ "Online connections to SIMOTION devices".

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Representation of SIMOTION D4x5-2 in HW Config

Once you have created a project and inserted a SIMOTION D4x5-2 as module, **HW Config** opens automatically (if parameterized).

In **HW Config** the SIMOTION D4x5-2 units are shown with the SINAMICS Integrated and the interfaces.

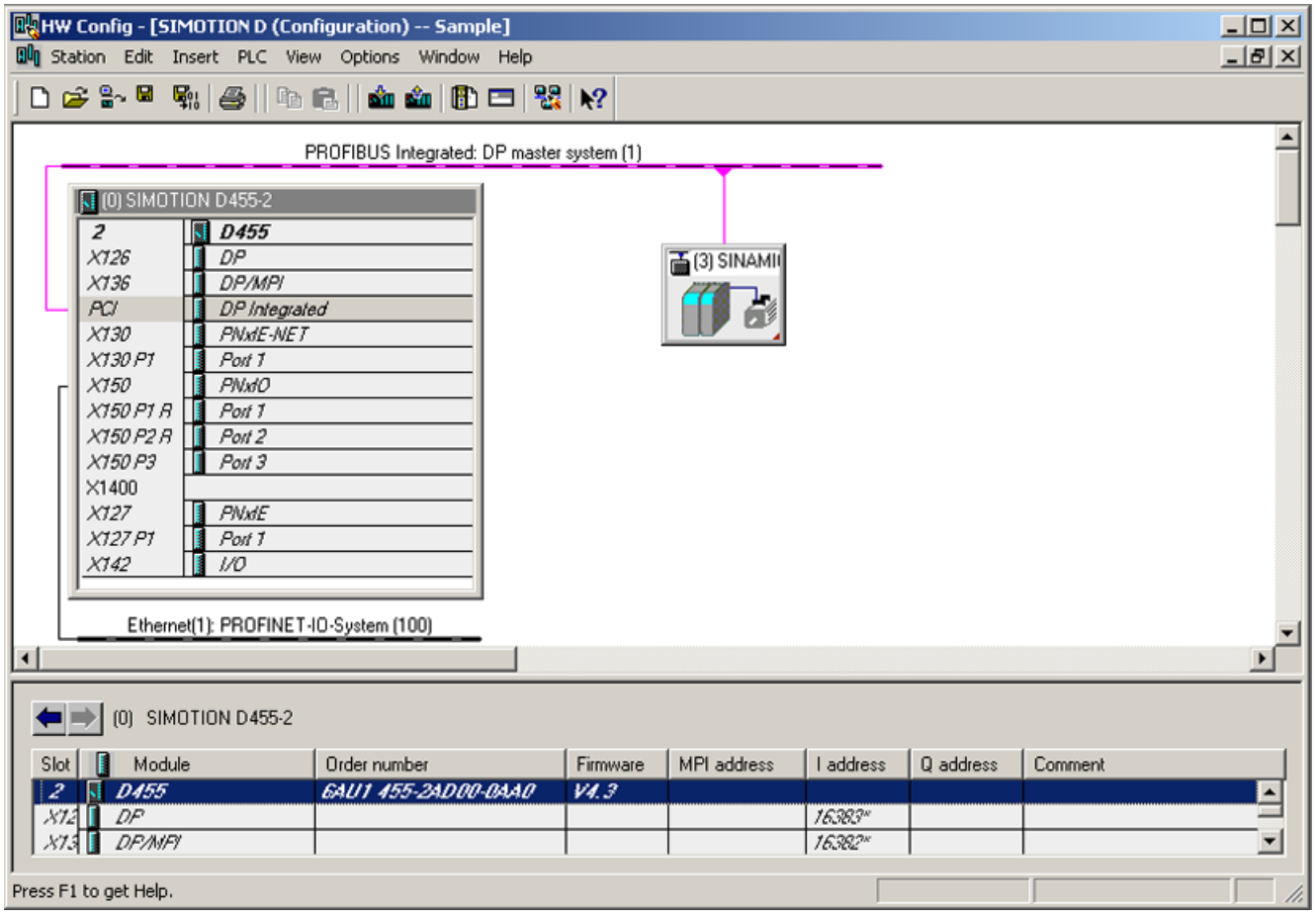


Figure 10-42 Representation of a D4x5-2 in HW Config

Note

According to the DNS conventions, "/" is a permissible character. For this reason, the Ethernet and PROFINET interfaces have a different name in the engineering software than on the module lettering ("/" is replaced by "x")

Example:
PN/IE (lettering on module) → PNxIE (as shown in SCOUT, HW Config, NetPro)

Configuring PROFIBUS DP

General information about communication via PROFIBUS DP

Definition of PROFIBUS DP

PROFIBUS DP is an international, open field bus standard specified in the European field bus Standard EN 50170 Part 2. PROFIBUS DP is optimized for high-speed, time-sensitive data transfer at field level.

Components communicating by means of PROFIBUS DP are classified as master or slave components.

- Master (active bus node):
  Components that represent a master on the bus define data transfer along the bus, and are therefore known as active bus nodes.
  Masters components are divided into two classes:
  - DP master class 1 (DPMC1):
    Central master devices are thus designated, which exchange information with the slaves in specified message cycles.
    Examples: SIMOTION D445-2 DP/PN, C240, P350, SIMATIC S7, etc.
  - DP master class 2 (DPMC2):
    These are devices for configuration, commissioning, and operator control and monitoring while the bus is in operation.
    Examples: Programming devices, operator control/monitoring devices
- Slaves (passive bus nodes):
  These devices may only receive, acknowledge and transfer messages to a master when so requested.
  Examples: SINAMICS drives, I/O modules

Functions on PROFIBUS DP

The functional scope can differ between DP masters and DP slaves. The functional scope is different for DP-V0, DP-V1 and DP-V2.

These functions on the PROFIBUS DP are characterized by:

- Configurable, equidistant, isochronous PROFIBUS DP cycle
- Synchronization of slaves by the master by means of a global control message frame in each cycle clock
- Independent maintenance of the isochronous cycle clock by the slaves in the event of a short-term communication failure.

Additional references

You will find additional information about PROFIBUS DP in the *SIMOTION Communication* System Manual.

Operating the SIMOTION D4x5-2 on PROFIBUS DP

PROFIBUS DP interface (X126, X136)

SIMOTION D4x5-2 has two interfaces for connection to the PROFIBUS DP. Transmission rates up to 12 Mbit/s are possible. Both interfaces can be operated isochronously.

The X136 interface can also be used as an MPI interface.

As supplied, both PROFIBUS DP interfaces are preset as a master with address 2 and a transmission rate of 1.5 Mbit/s. The PROFIBUS DP network is automatically created for this setting.

However, other settings can also be configured. This requires that you configure the network manually using **HW Config** and **NetPro**.

Note

Communication with the SINAMICS Integrated of a D4x5-2 or CX32-2 is always isochronous. Here, SIMOTION D4x5-2 is the master and the SINAMICS Integrated drives are slaves.

Master-slave configuration

The master/slave configuration can be used, for example, to establish hierarchical PROFIBUS networks that can be used to implement a modular machine concept.

Table 10-32 Master-slave configuration

X126 DP	X136 DP/MPI	Remark	Actions in the application
DP slave, isochronous	DP master, isochronous	Application synchronized to DP master (X136), application controls synchronization to DP slave (X126) Internal drive is synchronous with external cycle clock Cycle clock X136 = cycle clock DP Integrated	DP master / DP slave synchronization mechanisms
DP master, isochronous	DP slave, isochronous	Application synchronized to DP master (X126), application controls synchronization to DP slave (X136) Internal drive is synchronous with external cycle clock Cycle clock X126 = cycle clock DP Integrated	DP master / DP slave synchronization mechanisms
DP slave, isochronous	DP master, not isochronous	Application synchronized to DP slave (X126) (can be monitored by the application) Internal drive is synchronous with X126	DP slave synchronization mechanisms
DP master, not isochronous	DP slave, isochronous	Application synchronized to DP slave (X136) (can be monitored by the application) Internal drive is synchronous with X136	DP slave synchronization mechanisms
DP master, isochronous	DP master, isochronous	Application synchronized to DP master (X126, X136) Internal drive is synchronous with external cycle clock Cycle clock X126 = cycle clock X136 = cycle clock DP Integrated	None
DP master, isochronous	DP master, not isochronous	Application synchronized to DP master (X126) Internal drive is synchronous with X126 Cycle clock X126 = cycle clock DP Integrated	None
DP master, isochronous	DP slave, not isochronous	Application synchronized to DP master (X126) Internal drive is synchronous with X126 Cycle clock X126 = cycle clock DP Integrated	None
DP master, not isochronous	DP master, isochronous	Application synchronized to DP master (X136) Internal drive is synchronous with X136 Cycle clock X136 = cycle clock DP Integrated	None
DP slave, not isochronous	DP master, isochronous	Application synchronized to DP master (X136) Internal drive is synchronous with X136 Cycle clock X136 = cycle clock DP Integrated	None
DP master, not isochronous	DP master, not isochronous	Application synchronized to internal drive cycle clock	None
DP slave, not isochronous	DP master, not isochronous	Application synchronized to internal drive cycle clock	None
DP master, not isochronous	DP slave, not isochronous	Application synchronized to internal drive cycle clock	None
DP slave, not isochronous	DP slave, not isochronous	Application synchronized to internal drive cycle clock	None

X126 DP	X136 DP/MPI	Remark	Actions in the application
DP slave, isochronous	DP slave, not isochronous	Application synchronized to DP slave (X126) (can be monitored by the application) Internal drive is synchronous with X126	DP slave synchronization mechanisms
DP slave, not isochronous	DP slave, isochronous	Application synchronized to DP slave (X136) (can be monitored by the application) Internal drive is synchronous with X136	DP slave synchronization mechanisms

For detailed information about controlling synchronization across the application, see the *Basic Functions for Modular Machines* Description of Functions.

Alternatively, the X136 interface can be used as an MPI interface with a transmission rate of 19.2 kbit/s up to 12 Mbit/s.

Assignment of the PROFIBUS addresses in HW Config

Assigning PROFIBUS addresses

In order for all devices to communicate with each other, you must assign a PROFIBUS address to each device before connecting them:

Note

Before you assign any PROFIBUS addresses, please remember that all addresses must be unique on the PROFIBUS subnet.

You set these PROFIBUS addresses individually for each device with the PG/PC using **HW Config**. Some PROFIBUS DP slaves have a switch for this purpose.

Note

The PROFIBUS addresses set at the devices using these switches must correspond with the address settings in **HW Config**.

Recommendation for PROFIBUS addresses

Reserve PROFIBUS address "0" for a service programming device and "1" for a service HMI device, which will be connected to the subnet if required.

Recommendation for the PROFIBUS address of the SIMOTION D4x5-2 in case of replacement or service:

Reserve address "2" for a SIMOTION D4x5-2. This prevents duplicate addresses when installing a SIMOTION D4x5-2 with a default setting on the subnet (e.g. when a SIMOTION D4x5-2 is replaced). You should therefore assign addresses greater than "2" to additional units on the subnet.

Setting the DP cycle and system cycle clocks

All cycle clocks for the SIMOTION D4x5-2 are based on the DP cycle of SINAMICS Integrated, which must be set in **HW Config**.

To do so, click the SINAMICS block on the integrated PROFIBUS. The "DP Slave Properties" dialog box opens. You can adjust the DP cycle of the SINAMICS Integrated on the "Isochronous mode" tab.

Table 10-33 SIMOTION D4x5-2 value range

	D425-2 DP D425-2 DP/PN D435-2 DP	**D435-2 DP/PN D445-2 DP/PN D455-2 DP/PN**
DP cycle	≥ 0.5 ms (DP internal)≥ 1.0 ms (DP external)	≥ 0.25 ms (DP internal) ≥ 0.5 ms (DP internal, < V4.5) ≥ 1.0 ms (DP external)
Grid	0.125 ms	0.125 ms
Min. IPO cycle clock	≥ 0.5 ms	≥ 0.25 ms ≥ 0.5 ms (< V4.5)

External DP interfaces can only be operated with a DP cycle of ≥ 1 ms.

SINAMICS Integrated always runs in isochronous mode. The DP cycle setting of the SINAMICS Integrated is displayed as the "Bus data cycle" in the "System Cycle Clocks" dialog box. In SIMOTION SCOUT, select the SIMOTION D Control Unit and then select the "Set system cycle clocks" option in the "Target system" > "Expert" menu.

The table below shows the possible ratio settings for the SIMOTION D4x5-2 system cycle clocks based on the bus cycle clock.

Table 10-34 Ratios of system cycle clocks

Servo cycle clock: Bus cycle clock	**IPO cycle clock: Servo cycle clock**	**IPO2 cycle clock: IPO cycle clock**
As of V4.3: 1 ...4, 8 V4.2: 1	1 ... 6	2 ... 64

As of V4.3, a reduction ratio for the servo cycle clock to the DP cycle clock is possible. The reduction ratio is only permitted when PROFINET IO with IRT has not been configured.

If the DP interfaces (X126/X136) are configured as isochronous master interfaces, you must set both DP cycles equal to the bus cycle clock of the SINAMICS Integrated in **HW Config**.

If the DP interfaces (X126/X136) are operated as the master, the system cycle clocks are obtained from an internal cycle clock of the module. Of the two DP interfaces (X126/X136), no more than one can also be operated as an isochronous slave interface. In this case, the system cycle clocks are obtained from the cycle clock of the slave interface.

As a result, the task system of SIMOTION and SINAMICS Integrated runs synchronously to the slave cycle clock. This assumes that a slave cycle clock exists and synchronization with the slave cycle clock has been achieved. If this is not the case, the system cycle clocks are acquired from an internal replacement cycle clock.

When the project is downloaded, the cycle clock configuration is downloaded to the SIMOTION D4x5-2 and automatically set according to the specifications.

Note

The deployment of TM15 (SIMOTION) and TM17 (SIMOTION) is possible only with DP Integrated, PN and servo cycle clocks ≥ 0.5 ms. This applies to TM modules connected directly to D4x5-2 and indirectly via CX32-2/CU320-2.

Cycle clock scaling of external PROFIBUS interface to internal PROFIBUS interface

Definition

Cycle clock scaling means that an external PROFIBUS interface of the SIMOTION D4x5-2 (X126/X136) can be operated in an integer multiple of the internal PROFIBUS interface. This reduces the CPU load, thereby allowing you to operate more axes, for example. The settings of the scaled cycle clocks for the external DP interfaces are made in **HW Config**.

Supplementary conditions

The following supplementary conditions are applicable to cycle clock scaling:

- An external DP interface of the D4x5-2 is used as an isochronous slave interface. Only in this case can an **integer** cycle clock scaling of isochronous external DP slave interface to internal interface be specified. This is checked during compilation and an error message is output in the event of noncompliance. If the external DP interfaces are configured as equidistant interfaces but none are configured as slaves and cycle clock scaling is specified for these interfaces, an error is output during compilation.
- For SERVO, IPO, and IPO2, settings can also be made for all permissible cycle clocks. Master and slave axes can run in different IPO levels. Different cycle clocks and phase offsets are tolerated by the system.

  Note

  The IPO cycle clock of the IPO in which the synchronous operation technology object runs must be set equal to the cycle clock of the isochronous external DP slave interface.

- The second external DP interface can be operated as an isochronous master (while the other is an isochronous slave) in order to operate external drives, for example. In this case, the cycle clock must be the same as the cycle clock of the internal PROFIBUS DP. If this condition is not satisfied, an error message is output during compilation.
- One or both external DP interfaces can also be operated as non-isochronous, free-running interfaces. In this case, there is no effect on the cycle clock settings.

Application example

The system consists of a synchronous master (DP master) and at least one SIMOTION D4x5-2 synchronous slave (DP slave). The synchronous master contains the master axis; the synchronous slave contains the following axes:

- The axes in the SINAMICS Integrated of the D4x5-2 synchronous slave must exhibit high performance with a servo cycle clock of 1 ms and an internal DP cycle of 1 ms. This requires that the internal fast PROFIBUS DP be decoupled from the slower external PROFIBUS DP.
- The PROFIBUS DP has, for example, a cycle time of 4 ms due to the quantity framework on the bus; in all cases, its cycle time exceeds that of the cycle clock of the internal DP interface.
- The master values are transmitted via the DP bus. Further nodes can also be connected to the DP bus, e.g. DP drives, distributed I/Os, etc.

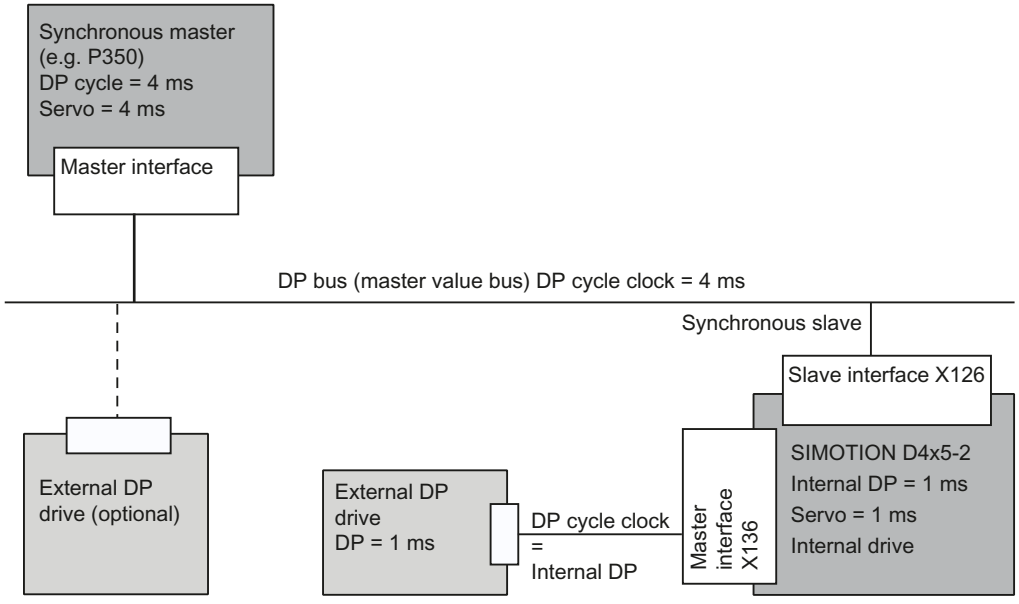


Figure 10-43 Example of a cycle clock scaling for PROFIBUS DP

Internal PROFIBUS interface with DP Integrated < 0.5 ms

Requirement

The minimum settable cycle clock for the internal PROFIBUS interface (DP Integrated) is 0.5 ms.

As of SIMOTION version V4.5, the following SIMOTION D Control Units can also be operated with a DP Integrated with 250 µs and 375 µs:

- SIMOTION D435-2 DP/PN
- SIMOTION D445-2 DP/PN
- SIMOTION D455-2 DP/PN

Configuration rules

The following configuration rules must be taken into account when using DP Integrated cycle clocks < 0.5 ms:

- Simultaneous use of Servo_fast and IPO_fast with DP Integrated cycle clocks < 0.5 ms is not possible because the resulting cycle clock ratios violate the rules for the use of Servo_fast and IPO_fast.
  See also Section Setting of the cycle clocks (two servos activated) (Page 7034)
- If the SINAMICS Integrated is operated with a DP Integrated cycle clock of 250 µs or 375 µs, this results in a higher utilization on the drive side in comparison to a cycle clock of ≥ 500 µs. Depending on the utilization situation (e.g. through ALM, CXs, TMs, etc.), this can result in a reduction in the number of axes (e.g. from six to five servo drives). It can also result in a reduction of the quantity structure for vector and V/f-controlled drives.
- CBE30 supports a minimum send cycle clock of 500 µs. The SINAMICS Integrated can therefore only be operated with 250 µs or 375 µs if isochronous communication has not been configured for the CBE30.

Creating a new PROFIBUS subnet

Introduction

SIMOTION SCOUT is used to network the SIMOTION D4x5-2. During the configuration process, the desired bus parameters can be set for the PROFIBUS DP interfaces.

Note

If a hardware configuration is loaded without a PROFIBUS network (X126 or X136) being configured on the CPU, a new PROFIBUS address that was previously set in HW Config or **NETPro** will **not** be accepted by the CPU.

Requirement

You have created a project and have inserted a SIMOTION D4x5-2.

Procedure

To create a new subnet, proceed as follows:

1. In the project navigator, double-click the SIMOTION D control unit to call **HW Config**.
2. In the SIMOTION D4x5-2 representation, double-click the interface for which you want to create a PROFIBUS subnet.
   The "DPx Properties" dialog box is opened.
3. Click "Properties" to show the "PROFIBUS Interface DPx" dialog box.
4. Click "New" to call the "Properties - New PROFIBUS Subnet" dialog.
5. Name the new subnet and enter the properties of the new subnet, such as transmission rate, on the "Network settings" tab.

6. If the PROFIBUS interface is to be operated isochronously, click "Options". Activate the "Activate isochronous bus cycle" option in the "Options" dialog box and set the DP cycle. Confirm with "OK" to exit the "Options" dialog box.
7. Confirm with "OK" again to exit the "Properties - New PROFIBUS Subnet" dialog box and accept the settings.

The new subnet is now displayed in the "Properties - PROFIBUS Interface DPx" dialog. You can now connect the new subnet to the corresponding PROFIBUS interface.

Follow the same steps to configure the second PROFIBUS interface.

A graphical representation of the PROFIBUS subnet you have created is shown in **HW Config**.

Note

PROFIBUS DP functionality is both equidistant and isochronous in nature. As such, it can guarantee that bus cycles will have exactly the same length and ensures deterministic behavior. Applications: Connecting drives or synchronized I/O devices.

Modifying the data transmission rate

Introduction

You can modify the transmission rate in a PROFIBUS subnet in **HW Config** according to your requirements.

Procedure

1. Open the project in SIMOTION SCOUT.
2. Double-click the device whose PROFIBUS subnet you want to configure.
   HW Config is displayed showing the settings for this device.
3. In **HW Config**, double-click in the graphical display on the PROFIBUS network whose transmission rate you want to configure.
   The "Properties - DP Master System" dialog is displayed.
4. Click on "Properties" to display the "PROFIBUS Properties" dialog.
5. Select the required transmission rate on the "Network settings" tab.
   If you wish to activate an isochronous bus cycle, the setting can be made under "Options."
6. Confirm with "OK".
7. Save and compile the new hardware configuration, and load it on the SIMOTION D.

Note

If you modify the transmission rate of the subnet over which you are operating the PG/PC, the PG/PC loses its active designation. You must then reconfigure it manually in **NetPro** or else you will no longer be able to go online by means of this PG/PC.

Note

PROFIBUS DP functionality is both equidistant and isochronous in nature. As such, it can guarantee that bus cycles will have exactly the same length and ensures deterministic behavior. Applications: Connecting drives or synchronized I/O devices.

See also

SIMOTION Runtime Basic Functions Function Manual, Chapter "Isochronous I/O processing on fieldbus systems."

Establishing a PG/PC assignment

Introduction

A PG/PC is required to create projects for a SIMOTION D4x5-2 and download them to the target device. The interface that can be connected via the PG/PC will be automatically queried during the communications configuring. If you change these settings, you must reestablish the active designation of the PG/PC in **NetPro** (the PG/PC connection must appear yellow and bold in **NetPro**).

Procedure

1. Open the project in SIMOTION SCOUT.
2. Click the "Open NetPro" button.
   NetPro is accessed, and the configured network is graphically displayed. The PG/PC connection to the configured network is shown in bold in a color other than yellow.
3. Double-click the PG/PC you would like to configure.
   The "Properties - PG/PC" dialog will be displayed with the "Assignment" tab in foreground.
4. Select the interface in the "Assigned" field and activate S7ONLINE access by clicking the appropriate checkbox.
5. Click "OK" to accept the settings.
   The PG/PC connection to the configured network is displayed again in bold and yellow.
6. Save and compile the changes and download them to the SIMOTION D4x5-2.

You can now go online via the PG/PC once again.

Alternatively, you can make the assignment in SIMOTION SCOUT by clicking the "Assign PG/PC" button. This calls the properties window for PG/PC assignment, where you can modify the assignment and "activate" it (S7ONLINE access).

Configuring the MPI bus

Operating the X136 interface as MPI

The X136 interface can also be used as an MPI interface, for example, to connect to an external PG/PC.

When the X136 interface is used as an MPI bus, additional activation of a drive on this interface is not possible.

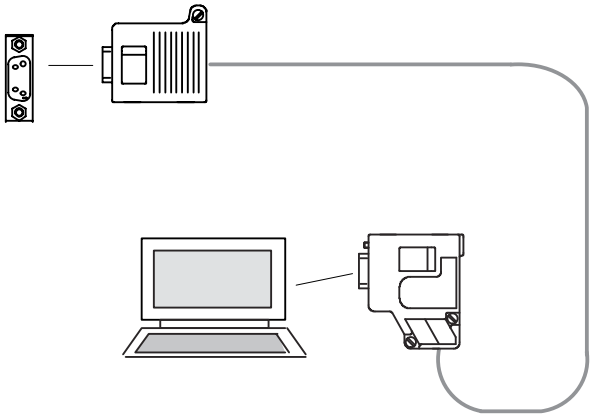


Figure 10-44 Connection of the D4x5-2 to MPI bus components

MPI parameters

MPI bus addresses and data transmission rate

Every node on the MPI bus must have a bus address in the range 0 to 31.

The data transmission rate on the MPI bus can be set to any value for the SIMOTION D4x5-2.

Communication attempt unsuccessful

If communication cannot be established at all, or if it cannot be established with individual nodes on the MPI bus, check the following:

- Is the transmission rate setting for the D4x5-2 used for all nodes?
- Are there any loose plug connections?
- Are all bus segments terminated properly?
  Bus segments that are not terminated properly will disrupt communication on the MPI bus.

Configuring PROFINET IO

General information about communication via PROFINET IO

Communication cycle

In PROFINET, the communication cycle is subdivided into different, time-specific intervals. The first interval is used for isochronous real-time communication (IRT), followed by real-time communication (RT) and standard TCP/IP communication. The bandwidth reservation for IRT ensures that RT communication and standard communication have no effect on the transmission of IRT telegrams, which are important for motion control applications.

The following figure shows how the PROFINET communication cycle is divided into isochronous real-time communication (IRT), real-time communication (RT), and standard TCP/IP communication.

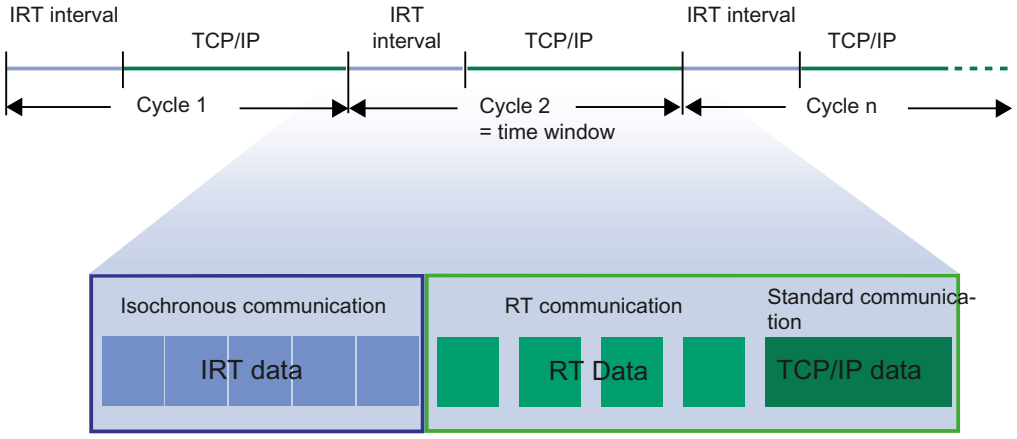


Figure 10-45 PROFINET communication cycle

Isochronous real-time Ethernet

STEP 7 can be used to configure PROFINET devices supporting data exchange via isochronous real-time Ethernet (IRT). IRT telegrams are transferred deterministically via planned communication paths in a defined sequence to achieve the best possible synchronism and performance.

IRT requires special network components supporting a planned data transfer.

Isochronous operation and mode

Equidistant mode and isochronous mode function similarly in PROFINET IO to the way they function in PROFIBUS DP.

For PROFIBUS DP, in isochronous operation all nodes are synchronized using a Global Control Signal created by the DP master.

In PROFINET IO with IRT, a sync master generates a signal to which sync slaves synchronize themselves. Sync master and sync slaves belong to a sync domain which is assigned a name via configuration. The role of the sync master can in principle be played by an I/O controller as well as an I/O device. A sync domain has exactly one sync master.

Context: Sync domain and I/O systems

An important fact is that sync domains do not need to be limited to one PROFINET IO system: The devices of several I/O systems can be synchronized by a single sync master, provided they are connected to the same Ethernet subnet.

The following applies the other way around: An I/O system must only belong to a single sync domain.

Signal propagation delays not negligible

For the extremely exact synchronization interval, line lengths, namely the associated delay times, must be taken into consideration. You can use a topology editor to enter the properties of the lines among the ports of the switches. STEP 7 uses this data and the other configuration data to calculate the optimized process of the IRT communication and the resulting updating time.

IRT runs in parallel to real-time and TCP/IP communication

Apart from IRT communication, for which a defined bandwidth is reserved within the update time, RT communication and TCP/IP communication are also permitted within the update time.

With RT communication (real-time communication), the cyclic data is transferred between I/O controller and I/O device, but without "best possible synchronism".

With non-synchronized I/O devices, data is exchanged automatically via RT communication.

Due to the fact that TCP/IP communication is also possible, other data, e.g. non-real-time data, configuration data or diagnostic data, can be transported.

PROFINET IO controller

Typically, the function of a PROFINET IO controller is taken on by controllers (e.g. SIMOTION C/P/D, SIMATIC S7 CPUs, ...).

The PROFINET IO controller takes on the master function for I/O data communication of the distributed field devices. The function is comparable to a PROFIBUS DP master class 1.

PROFINET IO device

Distributed field devices such as I/Os, drives (e.g. SINAMICS S120) or operator terminals are designated as I/O devices. The function is comparable to a PROFIBUS DP slave.

Addressing

In the delivery condition, the onboard PROFINET IO interface does not have an IP address or a subnet mask.

Note

The IP addresses 192.168.215.240 to 192.168.215.255 are reserved for internal communication in the SIMOTION D4x5-2 (subnet mask 255.255.255.240). When configuring the PROFINET interface (X150), make sure that the internal addresses are not located within the network of this interface. In IP, the network is defined as an AND link of IP address and subnet mask.

Media redundancy (MRP), as of V4.3

It is possible to establish redundant networks via the Media Redundancy Protocol (MRP). Redundant transmission links (ring topology) ensure that an alternative communication path is made available when a transmission link fails. The PROFINET devices that are part of this redundant network form an MRP domain.

MRP guarantees media redundancy in the event of a problem in the ring. The switchover of the ring is performed by the Redundancy Manager.

The switchover times depend on:

- The actual topology
- The devices used and
- The network load in the relevant network

The typical reconfiguration time of the communication paths for TCP/IP and RT frames in the event of a fault is < 200 ms.

In most systems, the switchover time of MRP is far above the PROFINET update time for cyclic data, so that a failure for cyclic data is detected. The PROFINET connection therefore fails and is reestablished after the switchover by the Redundancy Manager. In this way, an error can be corrected in the network, while the system continues to run **with bumps**.

Note

During the interruption of the ring, as well as when correcting the interruption (e.g. repair of the defective cable), there is a brief failure of the communication.

Ring ports

A SIMOTION/SINAMICS device may only be inserted in an MRP ring as a node with MRP-capable ports. For SIMOTION D, the first two ports of the PROFINET IO interfaces are designed as ring ports.

These two ports are marked with an "R" in the module rack in HW Config.

Note

Only devices with MRP-capable ports may be inserted in an MRP ring. If MRP-capable ports are not used, the reconfiguration times can be in the seconds range.

Bumpless media redundancy (MRPD)

Requirement for SIMOTION D4x5-2:

- SIMOTION SCOUT as of V4.3
- SIMOTION SCOUT TIA as of V4.5

MRPD is a procedure for bumpless media redundancy for PROFINET IO with IRT. MRPD also requires MRP.

The combination of MRP with MRPD provides bumpless PROFINET operation for short cycle times in the event of a fault in the ring. MRPD is based on IRT and ensures bumpless operation by the provider sending the cyclic data in both directions which the consumers then receive twice. If the ring is interrupted somewhere (e.g. through the failure of a ring node), receipt of the cyclic data via the fault-free side of the ring is still guaranteed.

Bumpless media redundancy MRPD always requires the activation of MRP in the individual rings.

A maximum of two Ethernet nodes may be between the sync master and the redundant sync master. If the redundant sync master is used together with MRPD, it is recommended that the redundant sync master be connected directly to the sync master and the two nodes located in a shared cabinet so that the cable connection between both nodes is protected.

If there is an interruption in the link between the sync master and the redundant sync master, the system continues to run without bumps, but after switching off and on again faults can occur.

Too high a network load or too-rapid coming and going of faults can also result in unfavorable cases in the failure of the PROFINET connection with activated MRPD because of delayed or incomplete switchover actions of MRP/MRPD.

For example, with two consecutive faults at different points in the ring, bumpless operation is only ensured when there is approx. three seconds between the two faults.

Additional information

You will find additional information about media redundancy in the *SIMOTION Communication* System Manual.

Setting a send cycle clock and system cycle clocks

Basic principles

Basic cycle clock

For the SIMOTION D4x5-2, the basic cycle clock depends on:

- Whether a second servo task has been activated for the runtime system.
- Whether a second PROFINET interface (=CBE30-2) is used for the SIMOTION D4x5-2 DP/PN.

By default, one servo task and two IPO tasks (IPO, IPO_2) are used. The IPO_2 cycle clock can be set as a multiple of the IPO cycle clock. In this way, the IPO_2 cycle clock can be used for lower priority technology objects (e.g. axes with less dynamic response).

Second servo task

A second servo task is only required for applications with special requirements, e.g.:

- For fast I/O processing via PROFINET IO (for particularly short sampling times and response times).
- If in addition to the electric axes (e.g. servo drives), hydraulic axes with particularly high-performance pressure/position control are implemented.
- If the electric axes have to be split into two performance classes on the servo level (fast servo - slow servo).

Note

If the axes have to be split into two performance classes, usually a split on the IPO level (IPO, IPO_2) is sufficient, a split on the servo level (Servo, Servo_fast) is only required in exceptional circumstances.

- The reference variable calculation / motion profile calculation of the axes is performed in the IPO level.

- The position control and monitoring of the axes is performed in the servo level.

Note that an additional Servo_fast/IPO_fast places an extra load on the SIMOTION runtime system.

In addition, with DSC (Dynamic Servo Control) for servo axes, the dynamically effective part of the position controller in the drive is performed in the frequency of the speed control loop (i.e. usually with 125 µs in the case of SINAMICS S120). If symbolic assignment with automatic telegram determination is used, DSC is activated automatically.

Table 10-35 Comparison of the task system with one or two servos

Characteristic	One servo (default)	Two servos	
Available servo tasks	- Servo	- Servo_fast - Servo	
Available IPO tasks	- IPO - IPO_2	- IPO_fast - IPO - IPO_2	
Basic cycle clock	• PROFINET cycle clock for servo, IPO, IPO_2, if PROFINET is isochronous • DP cycle of the SINAMICS Integrated for servo, IPO, IPO_2, if PROFINET is not isochronous	DP cycle of the SINAMICS Integrated for -Servo, - IPO, - IPO_2	PROFINET cycle clock (interface X150) for - Servo_fast - IPO_fast
Basic cycle clock when using a second PROFINET interface (= CBE30-2)	Both PROFINET interfaces use the same basic cycle clock	DP cycle of the SINAMICS Integrated = PROFINET cycle clock (X1400 interface, CBE30-2) for - Servo, - IPO, - IPO_2	PROFINET cycle clock (X150 interface) for - Servo_fast - IPO_fast
DP cycle	0.25/0.5 ... 0.8 ms (DP internal)[4] 1 ... 8 ms (DP external)	0.25/0.5 ... 8 ms (DP internal)[4] 1 ... 8 ms (DP external)	
PN cycle	0.25 ... 4 ms	0.25 ... 4 ms (D455-2 as of V4.5: Minimum 0.125 ms)[5]	
Grid [1]	125 µs	125 µs	
Servo cycle clock - D425-2 DP - D425-2 DP/PN - D435-2 DP - D435-2 DP/PN - D445-2 DP/PN - D455-2 DP/PN	Servo ≥ 0.5 ms ≥ 0.5 ms ≥ 0.5 ms ≥ 0.25 ms (≥ 0.5 ms for < V4.5) ≥ 0.25 ms (≥ 0.5 ms for < V4.5) ≥ 0.25 ms (≥ 0.5 ms for < V4.5)	Servo_fast/Servo[6] --- [2] --- [2] --- [2] ≥ 0.25 ms / ≥ 0.75 ms ≥ 0.25 ms / ≥ 0.75 ms ≥ 0.125 ms / ≥ 0.5 ms (< V4.5: ≥ 0.25 ms / ≥ 0.75 ms)	
IPO cycle clock - D425-2 DP - D425-2 DP/PN - D435-2 DP - D435-2 DP/PN - D445-2 DP/PN - D455-2 DP/PN	IPO ≥ 0.5 ms ≥ 0.5 ms ≥ 0.5 ms ≥ 0.25 ms (≥ 0.5 ms for < V4.5) ≥ 0.25 ms (≥ 0.5 ms for < V4.5) ≥ 0.25 ms (≥ 0.5 ms for < V4.5)	IPO_fast/IPO[6] --- [2] --- [2] --- [2] ≥ 0.25 ms / ≥ 0.75 ms ≥ 0.25 ms / ≥ 0.75 ms ≥ 0.125 ms / ≥ 0.5 ms (< V4.5: ≥ 0.25 ms / ≥ 0.75 ms)	
Notes regarding the drives		In Servo_fast/IPO_fast, only drives that are connected via PROFINET IO can be operated. [3]	

[1] The following applies for < V4.4: If there are IO devices with RT class "RT" in a sync domain, it is only possible to set the send cycle clocks 0.25 ms (for two servos), 0.5 ms, 1 ms, 2 ms and 4 ms.

[2] Servo_fast and IPO_fast are only provided on modules with PROFINET interface and as of performance class D435-2.

[3] CU310-2 PN/CU320-2 PN: Minimum PROFINET send cycle clock: 250 µs (as of SINAMICS version V4.5)

[4] 0.25 ms as of V4.5 for D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN

[5] Only available for SIMOTION D455-2 DP/PN in the TIA Portal (SCOUT TIA as of V4.5) Additional cycle clock setting: 125 µs

[6] Note the rules for the cycle clock ratio "PROFIBUS bus cycle clock" to "PROFINET cycle clock" when using two servos.

See also SINAMICS manuals.

Note

The DCC tasks T1...T3 are always assigned to Servo, IPO, or IPO_2.

You can also display the basic cycle clock used in **HW Config**. You can also activate the Servo_fast/ IPO_fast there, when required.

Proceed as follows to do this:

1. Open **HW Config**. Double-click the D4x5-2 module to open the "Properties - D4x5-2" dialog box.
2. You can activate use of the Servo_fast/IPO_fast in the "Isochronous tasks" tab. The basic cycle clock for the Servo and (if parameterized) the basic cycle clock for the Servo_fast is displayed.

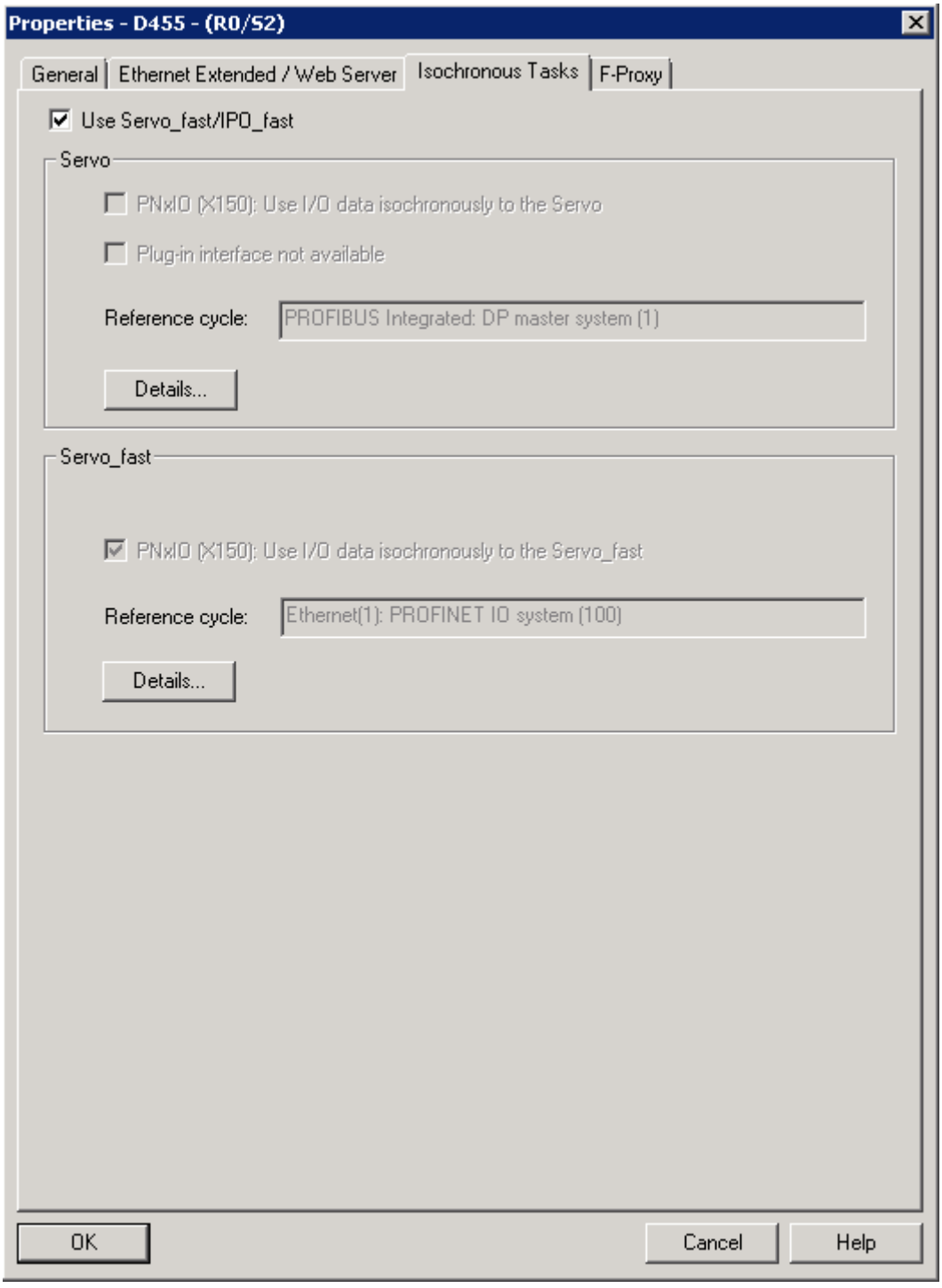


Figure 10-46 Properties D4x5-2 - Isochronous Tasks dialog box

3. Close the "Properties - D4x5-2" dialog box with "OK".
4. Save and compile the modified hardware configuration.
5. Load the new hardware configuration to the SIMOTION D4x5-2 via PROFIBUS DP / PROFINET IO / Ethernet.

The setting of the cycle clocks is described in the following.

Note

The deployment of TM15 (SIMOTION) and TM17 (SIMOTION) is possible only with DP Integrated, PN and servo cycle clocks ≥ 0.5 ms. This applies to TM modules connected directly to D4x5-2 and indirectly via CX32-2/CU320-2.

See also

You will find detailed information on the system cycle clocks in the *SIMOTION Runtime Basic Functions* Function Manual and in the *SIMOTION Communication* System Manual.

You will find detailed information on the configuration of axes in the *SIMOTION TO Axis, Electrical/Hydraulic, External Encoder* Function Manual.

Setting of the cycle clocks (one servo activated = default)

Setting the DP cycle in HW Config

To set the DP cycle of the SINAMICS Integrated, double-click the SINAMICS block on the integrated PROFIBUS in HW Config.

The "DP Slave Properties" dialog box opens. You can adjust the DP cycle of the SINAMICS Integrated on the "Isochronous mode" tab. See also Settings for DP slave properties (Page 7101).

Table 10-36 SIMOTION D4x5-2 value range

	D425-2 DP **D425-2 DP/PN** **D435-2 DP**	**D435-2 DP/PN** **D445-2 DP/PN** **D455-2 DP/PN**
DP cycle	≥ 0.5 ms (DP internal) ≥ 1.0 ms (DP external)	≥ 0.25 ms (DP internal) ≥ 0.5 ms (DP internal, < V4.5) ≥ 1.0 ms (DP external)
Grid	0.125 ms	0.125 ms

If, in addition to the drives on the SINAMICS Integrated / CX32-2, external drives are also to be connected via isochronous PROFIBUS, the DP cycle must be ≥ 1 ms.

Setting the send cycle clock in HW Config

The send cycle clock for PROFINET IO must be set in the "Domain Management" dialog in **HW Config**. To do this, select the "Edit" > "PROFINET IO" > "Domain management ..." menu command in **HW Config** and set the desired cycle clock.

The PROFINET interface can be operated with a send cycle clock in the range of 0.25 ms ≤ send cycle clock ≤ 4 ms when one servo is used. The smallest configurable time base is 0.125 ms.

Note

Information for version < V4.4

If there are IO devices with RT class "RT" in a sync domain, it is only possible to set the send cycle clocks 0.25 ms, 0.5 ms, 1 ms, 2 ms and 4 ms.

Cycle clock reduction ratio

As of V4.3, a reduction ratio for the PROFIBUS cycle clock to the Servo cycle clock is possible. The reduction ratio is only permitted when PROFINET IO with IRT has not been configured. A reduction ratio for the PROFINET send cycle clock to the PROFIBUS cycle clock is also possible.

Example:
PROFINET send cycle clock = 0.5 ms
PROFIBUS cycle clock = servo cycle clock = 1 ms

The PROFIBUS cycle clock can be operated relative to the PROFINET send cycle clock at a ratio of 1:1 to 16:1.

The table below shows the possible ratio settings for the SIMOTION D4x5-2 system cycle clocks based on the DP cycle of the SINAMICS Integrated or PROFINET send cycle clock.

Table 10-37 Ratios of the system cycle clock (when one servo is used)

Cycle clock name	Settable factors	Reference cycle clock
PROFIBUS DP bus cycle clock	1, 2, 3, 4, 6, 8, 10, 12, 14, 16	PROFINET IO send cycle clock
Servo	As of V4.3: 1, 2, 3, 4, 8[1)] V4.2: 1	PROFIBUS DP bus cycle clock
IPO	1, 2, 3, 4, 5, 6	Servo
IPO_2	2, 3, 4, 5, ..., 64	IPO

[1)] Always "1" if PROFINET with IRT has been configured

Setting the cycle clock ratios

The set bus cycle clock is displayed in SIMOTION SCOUT as the "Bus data cycle" in the "System Cycle Clocks - D4x5-2" dialog box. Select SIMOTION D4x5-2 and then select the "Set system cycle clocks" command from the "Target system" > "Expert" menu.

Set the required cycle clock ratios for the servo, IPO and IPO_2 in the "System Cycle Clocks - D4x5-2" dialog box.

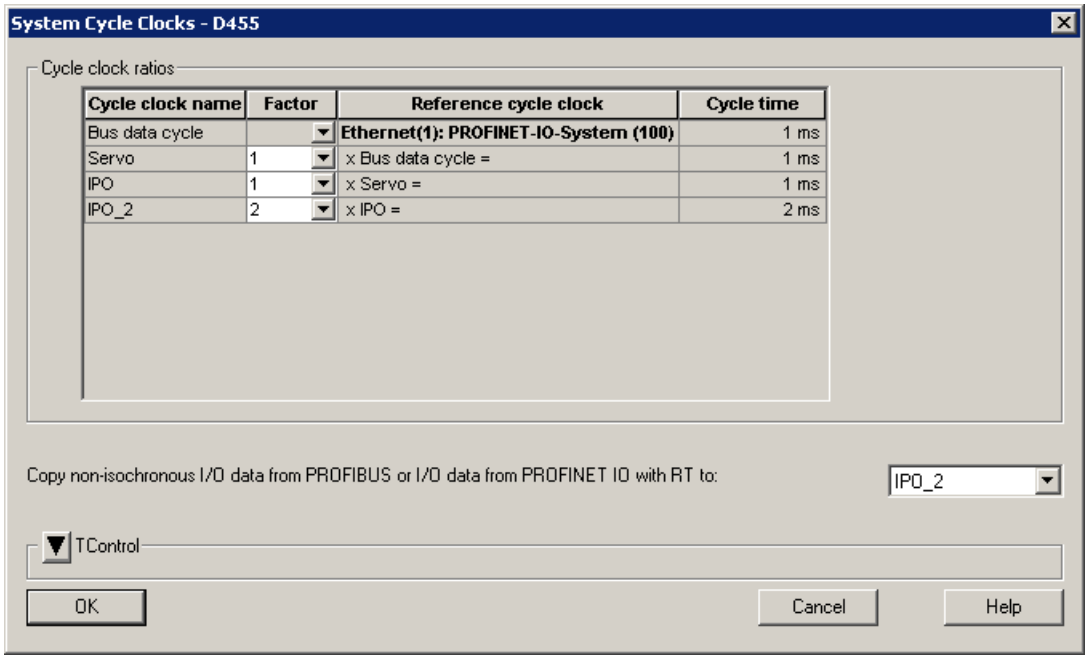


Figure 10-47 D4x5-2 system cycle clocks (one servo activated)

Setting of the cycle clocks (two servos activated)

Requirement

Supported control units: SIMOTION D435-2 DP/PN, D445-2 DP/PN and D455-2 DP/PN

If Servo_fast and IPO_fast are used, the PROFINET must be operated isochronously.

A reduction ratio for the PROFINET cycle clock to the Servo_fast-cycle and the PROFIBUS cycle clock to Servo cycle clock is not possible.

Cycle clock ratios and conditions

The table below shows the possible ratio settings for the SIMOTION D4x5-2 system cycle clocks based on the DP cycle of the SINAMICS Integrated or PROFINET send cycle clock.

Table 10-38 Ratios of system cycle clocks (two servos)

Cycle clock name	Settable factors	Reference cycle clock
Servo_fast	1	Send cycle clock of the onboard PROFINET IO interface (X150)
IPO_fast	1, 2, 4	Servo_fast
Servo	1	PROFIBUS DP bus cycle clock and send cycle clock of the optional second PROFINET IO interface (CBE30-2)
IPO	1, 2, 4	Servo
IPO_2	2, 3, 4, 5, ... 64	IPO

Note

- If illegal settings are made in the "D4x5-2 System Cycle Clocks" dialog box, an error is signaled during the consistency check.
- If the required isochronous buses have not be configured, "???" is displayed under cycle time in the "D4x5-2 System Cycle Clocks" dialog box.

The following conditions must also be met when using 2 servos:

- Bus cycle clock PROFIBUS DP = N x send cycle clock of the onboard PROFINET IO interface X150
  N = 2, 4, 8, 16, 32, 64 (N = 2 only for send cycle clock > 250 µs)
  (for < V4.4: N = 2, 4, 8, 16 for all send cycle clocks)
- Send cycle clock of an optional second PROFINET IO interface (CBE30-2) = PROFIBUS DP bus cycle clock
- IPO ≥ IPO_fast
- Servo_fast and IPO_fast are always assigned to the PROFINET interface
- The onboard PROFINET interface can only be operated as sync master (consequently with distributed synchronous operation via PROFINET, the D4x5-2 cannot participate as sync slave when Servo_fast/IPO_fast is used)
  If the PROFINET interface is configured as sync slave, the faulty configuration is indicated via a diagnostic buffer entry and the controller goes into starting inhibit.

Setting the bus cycle clocks

See Section Setting of the cycle clocks (one servo activated = default) (Page 7032). This procedure is described in:

- Setting the DP cycle in HW Config or
- Setting the send cycle clock in HW Config

Setting the cycle clock ratios

The set bus cycle clocks are displayed in SIMOTION SCOUT as the "Bus data cycle 1" and "Bus data cycle 2" in the "System Cycle Clocks - D4x5-2" dialog box. Select SIMOTION D4x5-2 and then select the "Set system cycle clocks" command from the "Target system" > "Expert" menu.

Set the required cycle clock ratios for the Servo_fast/IPO_fast as well as the Servo/IPO/IPO_2 in the "System Cycle Clocks - D4x5-2" dialog box.

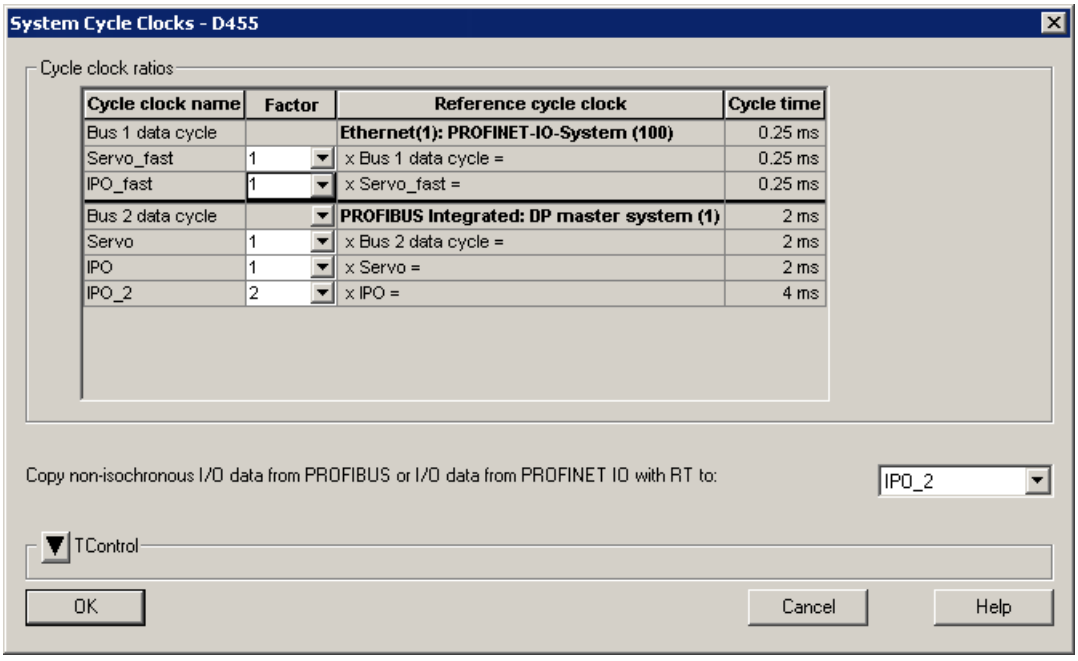


Figure 10-48 D4x5-2 system cycle clocks (two servos activated)

Using a second PROFINET interface

With the SIMOTION D4x5-2 DP/PN Control Units, optionally a second PROFINET interface is available as of V4.3 in addition to the onboard PROFINET interface (X150) with the Ethernet communication board (CBE30-2, X1400).

The CBE30-2 can only be operated in the D4x5-2 DP/PN Control Units, operation in the D4x5-2 DP Control Units is not possible.

Applications

A CBE30-2 can be used, for example, for the following applications:

- To use the second PROFINET interface.
  - To increase the maximum number of connectable devices.
  - To increase the available I/O address space.
- Devices are to be operated with different send cycle clocks and assigned different isochronous tasks (Servo and Servo_fast).
- Devices are to receive an independent IP address area or NameOfStation (e.g. higher-level network as plant network; local network for machine module; nodes in the "local network" can be addressed separately from the nodes in the "plant network").
- I-device and controller are to be operated isochronously at the same time.
  If one PROFINET interface is operated simultaneously as I-device and controller, either only the I-device or only the controller can be operated isochronously (i.e. with IRT). With two PROFINET interfaces, it is possible to operate a PROFINET IO controller and an I-device isochronously on a D4x5-2 DP/PN at the same time.

Rules for two PROFINET interfaces

The following rules must be taken into account when using two PROFINET interfaces, if both interfaces are to be operated isochronously (an isochronous device or an isochronous I-device or data exchange broadcast is configured at the interface):

- Both interfaces can be configured as a PROFINET IO I-device and/or as a controller.
- A higher-level controller must always be connected isochronously via the CBE30-2.
  This applies both for the controller-controller communication and for the I-device communication to a higher-level controller. Lower-level isochronous drives/I/O should be connected via the onboard PROFINET interface.
- The onboard PROFINET interface must be configured as sync master.
- The CBE30-2 can be configured as sync master or sync slave.
- An F-CPU can only be connected to one of the two PROFINET interfaces, because fail-safe I/O transfer areas can only be configured either on the I-device on the onboard PROFINET interface or on the CBE30-2.
  The PROFIsafe telegrams are routed to the following drives:
  - Drives on the SINAMICS Integrated and CX32-2
  - Drives on the onboard PROFINET interface (X150 interface)
  - Drives on the CBE30-2 (X1400 interface)

  The maximum quantity structure for PROFIsafe on PROFINET is therefore not increased by using a second PROFINET interface.
- If Servo_fast is not used, the servo isochronous devices can be assigned to both PROFINET interfaces.
- A redundant sync master is not permitted in the SYNC domain of the onboard PROFINET interface if both PROFINET interfaces are operated isochronously and the CBE30-2 is configured as a sync slave or IRT I-device.
- CBE30 supports a minimum send cycle clock of 500 µs. The SINAMICS Integrated can therefore only be operated with 250 µs or 375 µs if isochronous communication has not been configured for the CBE30.

Rules for Servo_fast

The following rules apply for the onboard PROFINET interface (X150) assigned to the Servo_fast:

- The interfaces are permanently assigned:
  - Servo/IPO/IPO_2 to the CBE30-2 (or DP and DP Integrated)
  - Servo_fast and IPO_fast to the onboard PROFINET interface
- If the Servo_fast is used, only one CACF (Controller Application Cycle Factor) = 1 is permitted for Servo and Servo_fast, i.e. it must be set:
  - Servo cycle clock = send cycle clock of the PROFINET interface X1400 (CBE30-2)
  - Servo_fast cycle clock = send cycle clock of the PROFINET interface X150 (onboard interface)
- The Servo_fast must have a reduction ratio to the Servo of 2, 4, 8, or 16.

Further information

For further information, see the *SIMOTION Communication* System Manual.

PROFINET V2.3: Optimized data transfer / 125 µs PROFINET send cycle clock

All SIMOTION D4x5-2 DP/PN Control Units support PROFINET according to the PI specification PN V2.3. For SIMOTION D455-2 DP/PN, the optional functionality **Performance Upgrade** is available, which provides the following functional scope:

- Reduction of the minimum PROFINET send cycle clock from 250 µs to 125 µs (isochronous)
- Optimization of data transfer (e.g. by Dynamic Frame Packing - DFP)
  By optimized data transfer
  - more devices can be operated with the same cycle time or
  - the cycle time can be reduced for the same number of devices

Requirements

The following configuration rules must be taken into account when using the 125 µs send cycle clocks:

- The 125 µs send cycle clock is only available for the SIMOTION D455-2 DP/PN
- Configuration only with SCOUT TIA as of V4.5 (SIMOTION in the TIA Portal)
- The 125 µs send cycle clock is only supported by the onboard PROFINET interface X150
- X150 can only be used as a PROFINET IO controller (not as an I-device)
- Only one port can be used on X150; the other two ports must therefore be deactivated in the interface properties of X150
  The following then applies:
  - The deactivated ports also cannot be used for TCP/IP and UDP communication
  - No MRP/MRPD media redundancy
  - Devices **that support a send cycle clock of 125 µs** must be connected via "one" active port of X150 (assign device to the controller and configure the Ethernet topology in the topology view)
- For isochronous mode, the clock synchronization must be activated on the controller (sync master) and on the device (sync slave) (RT class: IRT)
- The 125 µs send cycle clock is only supported in conjunction with Servo_fast/IPO_fast; activate the "Use Servo_fast" option at Isochronous mode in the CPU properties in the TIA Portal.
- In addition to the send cycle clock, the "Enable High Performance" option must be activated for the sync domain.

- Note the cycle clock dependencies for the DP Integrated and set a suitable cycle clock at PROFIBUS Integrated:
  - For a 125 µs PN send cycle clock: Only DP Integrated cycle clock 0.5 / 1 / 2 / 4 ms permitted
- For the ET 200SP, the Isochronous mode option must also be set for the I/O module in the device properties at Isochronous mode in the detail view; in addition, "Servo_fast" must be set as process image at I/O addresses in the properties of the I/O terminal.

Components that can be used

The 125 µs send cycle clock is only supported by selected PROFINET nodes (e.g. by the ET 200SP I/O system). No SCALANCE switches are currently available for a 125 µs send cycle clock.

For the PROFINET nodes that support a send cycle clock of 125 µs and the conditions that have to be met, see the relevant product documentation.

Further information

You will find further information on configuration in the *SIMOTION Communication* System Manual.

Properties of PROFINET

Properties

The onboard PROFINET IO interface supports the parallel operation of:

- IRT - isochronous real-time Ethernet
  - Operation of IRT I/O (e.g. ET 200SP)
  - Operation of a SINAMICS S120 as an IRT device
- RT - real-time Ethernet
  - Operation of RT I/O (e.g. ET 200SP, ET 200pro, etc.)
  - Operation of a SINAMICS S120 as an RT device
- TCP/IP, UDP, HTTP, ... standard Ethernet services

Note

For mixed operation of IRT and RT, make sure that the IRT-compatible devices form what is referred to as an IRT domain; i.e. there must not be any non-IRT devices on the data transmission link between the IRT devices.

Additional references

You will find an overview of the specific properties of PROFINET IO on SIMOTION D in the *SIMOTION Communication* System Manual.

Configuration tasks

Configuration of PROFINET involves the following steps:

1. Insert the SIMOTION D4x5-2.
2. Configure the onboard PROFINET IO interface in **HW Config**.
3. Create a topology: Here, you specify how the individual ports of the PROFINET IO devices are interconnected.
4. Configure the sync domain: Here, you specify which PROFINET nodes are sync masters (clock generators) and sync slaves.
5. Specify the send clock: Describes the time during which a PROFINET IO device exchanges user data with the PROFINET IO controller.
6. Configure the direct data exchange: The direct data exchange specifies which address areas are to be used for sending and receiving respectively.

Additional references

You will find a detailed description of each configuration step in Section "Configuring PROFINET IO with SIMOTION" of the *SIMOTION Communication* System Manual.

Configuring an Ethernet subnet

Properties of the Ethernet interfaces

Characteristics

SIMOTION D4x5-2 has several onboard Ethernet interfaces:

- X127 P1 PN/IE (on the front of the module)
- X120 P1 PN/IE-OP (under the front flap, only for D4x5-2 DP)
- X130 P1 PN/IE-NET (under the front flap)

You can connect an Industrial Ethernet with a transmission rate of 10/100/1000 Mbit/s to the 8-pin **X127 P1**, **X120 P1** and **X130 P1** RJ45 sockets.

As the X127 is easily accessible on the front of the module, this should preferably be used for the PG/PC connection.

Note

As of V4.3, the three Ethernet interfaces support the PROFINET basic services - they therefore have the designation PN/IE-NET, PN/IE-OP or PN/IE.

The interfaces do not have any HUB/switch functionality, i.e. telegrams are not forwarded from one interface to the other. As a rule, the interfaces belong to separate S7 subnets.
SIMOTION D4x5-2 does not have any IP router functionality; it does not forward the telegrams from one IP subnet to another.

The features of the Ethernet interfaces are as follows:

- TCP/IP timeout parameters can be set jointly for all Ethernet/PROFINET interfaces (HW Config: Double-click the D4x5 module, "Ethernet extended" tab).
- The transmission rate / duplex can be set separately for all interfaces
- The interfaces have autocrossing functionality.

Ethernet communication

SIMOTION D4x5-2 offers the following functions via Industrial Ethernet:

- Communication with STEP 7, SIMOTION SCOUT and SIMATIC NET OPC via a PG/PC
- Communication via UDP (user datagram protocol) with other components, e.g. other D4x5-2 devices
- Communication with other devices via TCP/IP
- Connection of SIMATIC HMI devices, such as MP27x, MP37x or PC-based HMIs
- IT communication (e.g. via SIMOTION IT OPC XML-DA)
- Communication based on OPC UA (Unified Architecture)
- PROFINET basic services (e.g. DCP, LLDP, SNMP)
  These PROFINET basic services provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives, I/O modules, etc.

Setting/reading the IP address

As of V4.4, new system functions (_setPnIpConfig/_getPnIpConfig) for setting/reading the IP address are available. The new system functions have a larger functional scope (e.g. temporary or permanent setting of the IP address) and support all Ethernet-based interfaces of a SIMOTION D4x5-2.

The previous system functions _setIpConfig/_getIpConfig only support the interfaces X127 PN/IE (EnumInterfaceID = IE_01(0)) and X130 PN/IE-NET (EnumInterfaceID = IE_02(1)). The interfaces X120 PN/IE-OP, X150 PN, and X1400 PN (CBE30-2) cannot be used. We recommend only using the new system functions in the future.

Routing

"Services via TCP" are supported for all Ethernet interfaces.
From the two Ethernet interfaces, S7 routing is possible to the

- PROFIBUS interfaces and
- PROFINET interfaces
- As well as between the Ethernet interfaces

IP routing from one Ethernet interface to another, as well as from the PROFINET interface to the Ethernet interface and vice versa, is not possible.

You can find the MAC addresses on the rating plate located on the front of the SIMOTION D4x5-2.

See also

Further information about routing and the differences between IP and S7 routing can be found in the *SIMOTION Communication* System Manual.

Default Ethernet addresses

The following addresses are assigned by default:

Table 10-39 IP address assignment for a SIMOTION D4x5-2

Interface	Use case	Default address	
X127 P1 PN/IE	Insert SIMOTION device or HW Config	IP address: Subnet mask: Router address: Automatic private IP address	169.254.11.22 255.255.0.0 0.0.0.0
	D4x5-2 as delivered	IP address: Subnet mask: Router address: Automatic private IP address	169.254.11.22 255.255.0.0 0.0.0.0
X120 P1 PN/IE-OP (only for D4x5-2 DP)	Insert SIMOTION device or HW Config	IP address: Subnet mask: Router address:	192.168.213.1 255.255.255.0 0.0.0.0
	D4x5-2 DP as delivered	IP address: Subnet mask: Router address: Automatic private IP address	0.0.0.0 0.0.0.0 0.0.0.0
X130 P1 PN/IE-NET	Insert SIMOTION device or HW Config	IP address: Subnet mask: Router address:	192.168.2.1 255.255.255.0 0.0.0.0
	D4x5-2 as delivered	IP address: Subnet mask: Router address:	0.0.0.0 0.0.0.0 0.0.0.0

Note

The IP addresses 192.168.215.240 to 192.168.215.255 are reserved for internal communication in the SIMOTION D4x5-2 (subnet mask 255.255.255.240). When configuring the external Ethernet interfaces (X127 P1, X120 P1 and X130 P1), make sure that the internal addresses are not inside their network. (In IP, the network is defined as an AND link of IP address and subnet mask.)

Note

If you want to go online via Ethernet, you have to make sure that the connection from PG/PC to SIMOTION D4x5-2 is active. You can check this in **NetPro**. A description of how to switch a connection to active again can be found in the Section Establishing the PG/PC assignment (Page 7022).

Automatic Private IP Addressing

The Ethernet interfaces support the automatic IP assignment according to the Automatic Private IP Addressing (APIPA) process.

Interface X127 P1 already has IP address 169.254.11.22 in the delivery condition. Further information on Automatic Private IP Addressing can be found on the Internet, e.g. in WIKIPEDIA with the keywords "APIPA" or "Zeroconf".

Separate Ethernet interfaces

The Ethernet interfaces are entirely separate and are generally connected to different networks.

The Ethernet interfaces support:

- S7 routing
- No IP routing

SIMOTION D4x5-2 can therefore be connected to the machine user's internal company network with one interface and the second interface can be used, for example, by the machine manufacturer for the purpose of remote maintenance.

The Ethernet connection of the D4x5-2 can be configured in HW Config. Proceed as follows to do this:

1. Open your project.
2. Open HW Config. Double-click the Ethernet port (e.g. X127 P1) to open the "Properties PNxIE ..." dialog box.
3. You can configure the Ethernet connection in the "Options" tab.
   Recommendation: Use the default setting "Automatic setting". With Automatic setting, the baud rate and duplex operating mode are automatically aligned with the connection partner. The autocrossing functionality is also available so that you can use crossed and uncrossed cables.
   If transmission is to be set manually, not only must the connection (e.g. 10 Mbit/s half duplex) be set manually but autonegotiation must also be deactivated.
4. Close the "Properties PNxIE ..." dialog box with "OK".
5. Save and compile the modified hardware configuration.
6. Load the new hardware configuration to the SIMOTION D4x5-2 via PROFIBUS DP / PROFINET IO / Ethernet.

Shielded twisted pair cables are used for the networking.

- 4- and 8-wire cables can be used for 10/100 Mbit/s
- 8-wire cables must be used for 1000 Mbit/s

Note

With SIMOTION < V4.3, the settings are made via HW Config by double-clicking the D4x5-2 module. The "Properties - D4x5 2" dialog box opens. You can configure the Ethernet connection in the "Extended Ethernet" tab.

For additional information, see the *SIMATIC NET, Industrial Twisted Pair, and Fiber Optic Networks* Manual.

For further information about the cabling spectrum for Ethernet, see the *Industrial Communication IK PI* Catalog.

Configuring Ethernet addresses in HW Config

Requirement

For configuration using Industrial Ethernet, SIMOTION D4x5-2 must be provided with an IP address, the subnet mask, and the router address.

Note

Only one router may be configured.

Procedure

To configure and transfer Ethernet addresses to the D4x5-2, proceed as follows:

1. Open your project.
2. Open **HW Config**. Double-click the interface to be configured (e.g. X127) to open the "Properties" dialog box.
3. On the "General" tab, click the "Properties" button of the Ethernet interface. The "Properties - Ethernet Interface" dialog is displayed.
4. Click the "New" button. The "New Industrial Ethernet" subnet dialog is displayed. In this dialog box, you can change the name of the new subnet or confirm the default setting with "OK".
5. The newly created Ethernet subnet is now displayed under "Subnet" in the "Properties - Ethernet Interface" dialog box and must be selected.
6. In this dialog box, enter the required addresses for "IP Address" and "Subnet". Under "Router", choose whether a router is to be used. If using a router, enter the router address.
7. Confirm this dialog box with "OK".
8. Close the "Properties" dialog by clicking "OK".
9. To configure another Ethernet interface, open the "Properties" dialog box of the other interface and repeat Steps 3 to 7.
10. Save and compile the modified hardware configuration.
11. Load the new hardware configuration to the SIMOTION D4x5-2.

Assigning the Ethernet address later

It is possible to assign the IP address later on (e.g. on modular machines). In this case, an IP address is not assigned in HW Config, but activated differently with the "Set IP address using different method" option. The address will then be assigned later on the machine, for example, by a the user program or by commissioning tools, such as PRONETA.

Further information on PRONETA can be found on the Internet:

- PRONETA: See Internet address (http://support.automation.siemens.com/WW/view/en/67460624)

Reading out IP and MAC address

Requirement

To read out the IP and MAC addresses, the following requirements must be met:

- SIMOTION D4x5-2 is wired.
- You have assigned the communication parameters.
- You are online.

Procedure

The IP addresses and MAC addresses of SIMOTION D4x5-2 can be displayed as follows via SIMOTION SCOUT.

1. Right-click the module.
2. Select "Target device" > "Device diagnostics" in the context menu.

The addresses are displayed as follows, e.g. for SIMOTION D4x5-2 DP/PN:

X127 P1 (PN/IE)

- Active MAC Address: 08-00-06-73-25-3E
- IP address: 169.254.11.22
- Subnet mask: 255.255.0.0
- Standard gateway: No router used

X130 P1 (PN/IE-NET)

- Active MAC address: 08-00-06-73-25-3F
- IP address: 192.168.2.1
- Subnet mask: 255.255.255.0
- Standard gateway: No router used

As an alternative, you can determine the IP address as follows:

- By selecting "Project" > "Accessible nodes" in SIMOTION SCOUT or
- By calling "Target system" > "Ethernet" > "Edit Ethernet node..." in HW Config and browsing to "Online accessible nodes"
- Using the system function _getIpConfig

Note

The MAC address is listed on the nameplate on the front of the module.

10.1.1.7 Commissioning (software)

Overview of commissioning

Requirements

The following requirements must be fulfilled in order to commission the SIMOTION D:

- The system has been connected and wired.
- The SIMOTION D has been switched on and powered up (STOP mode).
- SIMOTION SCOUT (with integrated STARTER) has been installed and powered up on the PG/PC.
- The communication and networks have been configured.
- You have created a project and inserted a SIMOTION D in the project.

Note

The article numbers of the SINAMICS S120 components must be available.

You need these order numbers when setting up a SIMOTION project to verify that the components selected from the hardware catalog in the **HW Config** application correspond to the ones used in the system.

Symbolic assignment / adaptation

Symbolic assignment

As of V4.2, SIMOTION supports the symbolic assignment on SINAMICS drive objects (DOs) during the configuration of technology objects (TO) and I/Os.

This simplifies the configuration of the technological relationships including the communication between controller and drive.

With the symbolic assignment:

- Only suitable assignment partners are offered in an assignment dialog box.
- Communication between axis and drive is set up automatically by the engineering system and the required PROFIdrive axis telegrams as well as the used addresses set up.
- Telegrams are extended and interconnections created automatically in the drive depending on the selected TO technology (e.g. SINAMICS Safety Integrated).
- Axis and drive configuration can initially be performed independently of one another.
- Communication connections are established automatically during the configuration of I/O variables on SINAMICS I/Os (telegrams are set up automatically, the I/Os interconnected to the telegram and the addresses set up).

Apart from the symbolic assignment, no further configuration is required for the communication. As addresses no longer have to be configured, the connection is retained even with address offsets.

Note

During the configuration of drive objects (DO drive, DO encoder, ...) as well as in the Telegram Configuration dialog box (see Section Telegram configuration (Page 7148)), you can deactivate the **automatic telegram configuration** and the **automatic telegram adaptation**.

As many of the previously described advantages are lost through a deactivation, we recommend that a deactivation only be performed in exceptional circumstances.

The symbolic assignment enables an independent configuration of the axes on the SIMOTION side and the drives on the SINAMICS side.

This enables, for example:

- The PLC and motion control functions to be completely configured by a programmer even without drive know-how using technology objects (e.g. TO axis) and loaded to the device.
- The drives to be separately configured and optimized by a drive expert.
- The technology objects to be symbolically assigned later to the drive objects via an interconnection dialog box.

Note

The previous methods of drive, axis and I/O configuration are still available.
Symbolic assignment must be deactivated for these methods.

For newly created projects, the symbolic assignment is used by default.

If projects < V4.2 are upgraded, the symbolic assignment is deactivated by default and must be activated when required.

Symbolic assignment can be activated/deactivated in SIMOTION SCOUT via the "Project" > "Use symbolic assignment" menu.

Activating symbolic assignment later

Symbolic assignment simplifies the configuration of the technological relationships including the communication between controller and drive significantly. The names of the symbolic assignments in plain text is also advantageous for project maintenance.
Symbolic assignment is therefore recommended for new projects as of V4.2 and is automatically active.

If symbolic assignment is used in a project, telegrams, interconnections and addresses are automatically created by the engineering system by default.

The engineering system sets the "optimal" PROFIdrive telegrams and telegram extensions for the system, makes the required BICO interconnections and determines the addresses.

It is also possible to change upgraded projects to symbolic assignment and must be decided individually, as follow-up work may be required.

Extensive follow-up work is usually required particularly for free telegram configurations (e.g. for TB30, TM15 DI/DO, TM31).

Note

If the symbolic assignment is activated later, previously existing telegram settings and BICO interconnections for all SINAMICS telegrams that are set to Standard/Automatic (Communication -> Telegram configuration) are replaced at the next compilation. This may change telegrams and BICO interconnections and delete telegram extensions previously set up manually.

For this reason, make a backup copy of your project before activating the symbolic assignment.

To retain the settings that have been made, after the selection of "Use symbolic assignment," the setting "User-defined" must be selected and the check box cleared for automatic telegram setting/address adaptation for the respective message before compilation.

For further details, see the *SIMOTION Runtime Basic Functions* Function Manual.

Assigning a drive later

You can create an axis in SIMOTION SCOUT and assign it later to a drive. You can thus load your user program to the controller and (with the exception of the non-existent drives) test it.

Compared to a procedure with temporarily created "virtual axes", "axes without assigned drive" have the advantage that the configuration data is completely available and do not require a "virtual axis -> real axis" reconfiguration.

Simulation of axes

The axis simulation can also be used to test the user program.

You can find a script for switching the axis simulation on and off in
SIMOTION Utilities & Applications, which is part of the scope of delivery of SIMOTION SCOUT.

For further details, see also *TO Axis Electrical/Hydraulic, External Encoder* Function Manual.

Adaptation

In addition to the symbolic assignment, the **automatic adaptation** facilitates the configuration of SINAMICS S110/S120 data as of SIMOTION V4.2. During power-up of the SIMOTION devices, reference variables as well as drive and encoder data of the SINAMICS S110/S120 are automatically taken over for the configuration data of the SIMOTION technology objects "TO axis" and "TO externalEncoder". This data no longer has to be entered in SIMOTION.

For additional information, see the following sources:

- *SIMOTION Runtime Basic Functions* Function Manual
- *TO Axis Electrical/Hydraulic, External Encoder* Function Manual

Requirement

Symbolic assignment is supported by the TO axis, TO externalEncoder and the TO outputCam, TO camTrack and TO measuringInput. The onboard I/Os of a SIMOTION D, of a SINAMICS S110/S120 control unit as well as from a TB30 and selected Terminal Modules can be interconnected symbolically.

Table 10-40 Control units that support a symbolic assignment

Module	Supports symbolic assignment
SIMOTION D410-2	As of SIMOTION V4.3
SIMOTION D410	As of SIMOTION V4.2
SIMOTION D4x5-2	As of SIMOTION V4.2
SIMOTION D4x5	As of SIMOTION V4.2
Controller extension • CX32-2 • CX32	As of SIMOTION V4.2
SINAMICS S110 CU305	As of SINAMICS V4.3
SINAMICS S120 • CU310-2 • CU310 • CU320-2 • CU320	 • As of SINAMICS V4.4 • As of SINAMICS V2.6.2 • As of SINAMICS V4.3 • As of SINAMICS V2.6.2

See also

Only the drive configuration by means of symbolic assignment is described in this documentation.

On the Internet, you can find documentation of older SIMOTION versions at: (https://support.industry.siemens.com/cs/ww/en/view/40211807)

For further information on the configuration of the TO axis and TO externalEncoder, see the *TO Axis Electrical/Hydraulic, External Encoder* Function Manual.

Procedure when commissioning

Commissioning steps

This section shows you how to configure a system and test the configured drives and axes. The commissioning steps are listed below. The sequence is only an example and can vary depending on the supplementary conditions.

1. Configure the SINAMICS Integrated:
   You can configure the integrated drives (SINAMICS Integrated) offline or online:
   - Offline configuration:
     Performing D4x5-2 offline configuration (Page 7052) and CX32-2 offline configuration (Page 7093)
     For offline configuration, all of the components and their article numbers must be known.
   - Online configuration:
     Performing D4x5-2 online configuration (Page 7080) and
     Performing CX32-2 online configuration (Page 7096)
     During online configuration, you can download all of the information from the connected DRIVE-CLiQ components to your user project.
     Note:
     The next two sections describe how a SIMOTION D4x5-2 including SINAMICS Integrated is configured. The described procedure also applies when CX32-2 controller extensions are used.
     The Section *Configuring a CX32-2* also describes the step-by-step commissioning with a CX32-2 controller extension. The advantage of a step-by-step configuration is that the location and cause of an error are easier to identify.
2. Test a drive with the drive control panel (Page 7132)
3. Create an axis with the axis wizard (Page 7135)
4. Test an axis with the axis control panel (Page 7142)
5. Activate the infeed (Line Module) (Page 7144)
6. Set up addresses and telegrams (Page 7147)
7. Link an additional encoder (optional) (Page 7152)
8. Configure drive-related I/Os (with symbolic assignment) (Page 7158)
9. Configure the technology (Page 7163)
10. Optimize the drive and controller (Page 7176)

This section also contains additional configuration information, e.g. for vector drives, Safety Integrated, etc.

Important functions for the project handling during commissioning

The following functions are very important for the project handling and commissioning:

Table 10-41 Buttons for operator control

Symbol	Function	Effect
	Save project and compile changes	Select Save project and compile changes to save the entire project and compile the project data (e.g. programs) in an executable code. Only the changes are compiled. Unchanged code is retained.
	Connect to selected target devices	The online connection is established to the selected target devices. Under "Target system" > "Select target devices", you can set which target devices are to go online.
	Download project to target system	Downloading programs to the SIMOTION device and performing configuration for the SINAMICS Integrated and any connected CX32-2 modules.
	Download CPU / drive unit to target device	The configuration is only loaded to the device selected in the project tree (which means that the function needs to be performed separately for each D4x5-2/CX32-2 and each SINAMICS Integrated).
	Load CPU / drive unit to PG	The unit's configuration is only loaded to the PG selected in the project tree (which means that the function needs to be performed separately for each D4x5-2/CX32-2 and each SINAMICS Integrated).
	Copy RAM to ROM	Copying from RAM to ROM is only performed for the device selected in the project tree (which means that the function needs to be performed separately for each D4x5-2/CX32-2 and each SINAMICS Integrated).

Note

Tips for going ONLINE:

In online operation, SCOUT attempts to conduct online operation with all hardware components contained in the project. This means that the time needed for going online increases.
We recommend that you make settings for SCOUT so that online operation is made only with those components currently needed. The setting can be found at "Target system > Select target devices..." in the menu. The selection and deselection of the devices in the online state can be made via the "Connect target device" context menu on the device.

This procedure is also advantageous when the configuration of the drive unit is completed. Without going completely offline, the connection can be simply deselected via the context menu on the drive unit.

Performing the configuration for the D4x5-2 offline

Overview

Introduction

In the case of offline configuration, the project is created while not all of the hardware components (in particular drives) are available. In this way, a SIMOTION project in the office environment can be created up to a point where a basic project specification including a program exists. You can then load the finished project to the SIMOTION D4x5-2 later and test it with the drives.

Requirements

- For offline configuration, all of the components and their article numbers must be known.
- You have created a project in SIMOTION SCOUT and inserted a SIMOTION D4x5-2 into the project in the hardware configuration.
- You have configured the communication between the SIMOTION D4x5-2 and the PG/PC. See Section Creating a project and configuring the communication (Page 7007).

Procedure

The offline configuration involves the following steps:

- Accessing the drive wizard (Page 7053)
- Configuring components (Page 7054)
- Downloading the project to the SIMOTION D4x5-2 by means of one of the following options:
  - Downloading to the target system (Page 7074)
  - Downloading to the CF card (Page 7077)
  - Downloading incl. sources and additional data (Page 7078)
  - Archiving on the CF card (zip file) (Page 7079)

Note

During offline configuration, you can also configure the option board (TB30) and Terminal Modules, e.g. TM41.

Accessing the drive wizard

Integrated drive

The SIMOTION D4x5-2 contains an integrated SINAMICS S120 drive unit, which is automatically inserted together with the SIMOTION D4x5-2 control unit in the project navigator. The integrated drive must be operated in equidistant, isochronous mode using PROFIdrive-compliant message frame types.

The drive wizard for the integrated STARTER is available in SIMOTION SCOUT for configuring the integrated drive and its associated modules (e.g. SINAMICS S120 active line modules and SINAMICS S120 motor modules).

Note

To configure the drive components of a CX32-2 controller extension, the same procedure should be used as for the integrated drive of the SIMOTION D4x5-2 control unit.

Note

Take note of all the necessary safety precautions and rules governing connections, which can be found in the latest SINAMICS S120 documentation on the SIMOTION SCOUT DVD.

Requirement

You have already created a project and have inserted a SIMOTION D4x5-2.

Procedure

To call the drive wizard for the configuration of your drive unit, double-click "Configure drive unit" below "D4x5-2" > "SINAMICS Integrated" in the project navigator.

You can configure the following components:

- Infeed (e.g. SINAMICS S120 active line module)
- Drive
- Power unit (e.g. SINAMICS S120 motor module)
- Motor
- Encoder
- Option module

Configuring components

Requirement

You have inserted a SIMOTION D4x5-2 into the project, configured the communication and called the drive wizard by double-clicking "Configure drive unit" in the project navigator.

Note

An overview of permissible configurations, quantity structures and DRIVE-CLiQ topologies can be found in the *SINAMICS S120* Commissioning Manual and in Section Quantity structures (Page 7199).

It should be noted, for example, that mixed operation of servo and vector **is not possible**, although mixed operation of servo and vector *V/f* **is possible**.

Failure to comply with the rules listed in this manual will result in errors that are not output until the download is performed, rather than at the configuration stage.

Procedure

While executing the wizard you will be prompted to perform, for example, the following configuration steps:

1. In the "Option Module" dialog box, select whether you want to use a TB30 as option module.

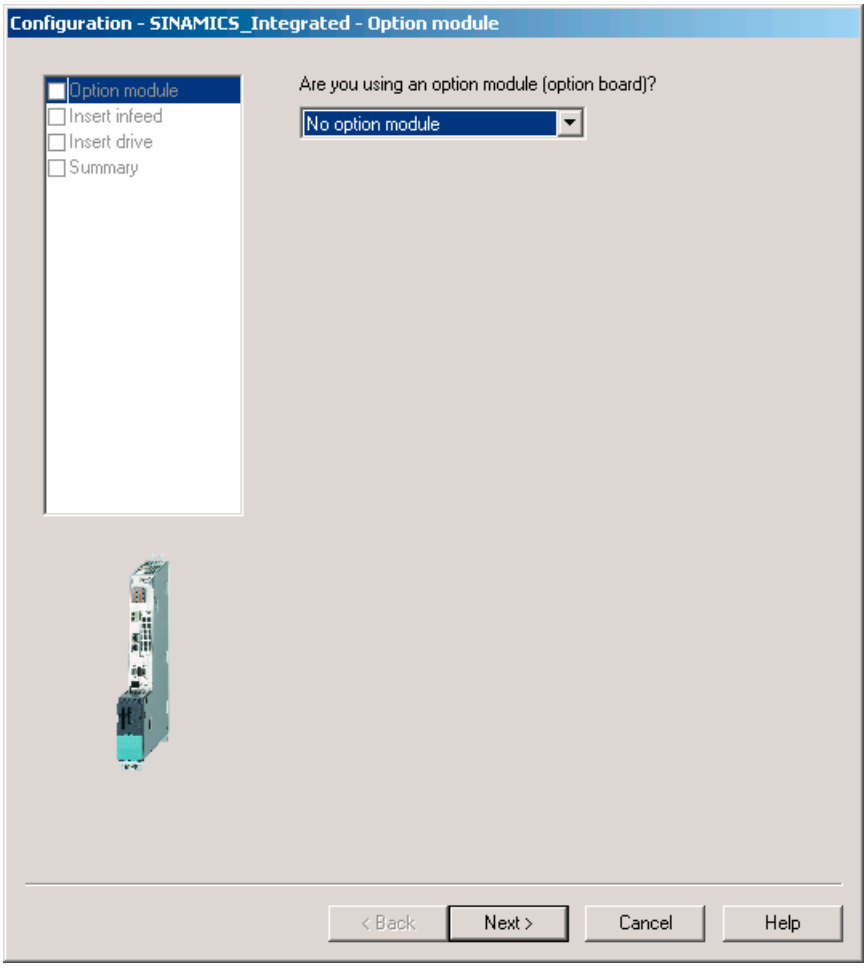


Figure 10-49 Selecting an option module

Note

The TB30 is displayed as drive object below the Control Unit in the project navigator and can be configured there.

2. In the "Insert Infeed" dialog box", select whether you want to use an infeed with or without a DRIVE-CLiQ connection. If the DC link has an external supply, select "No" (no DRIVE-CLiQ connection).

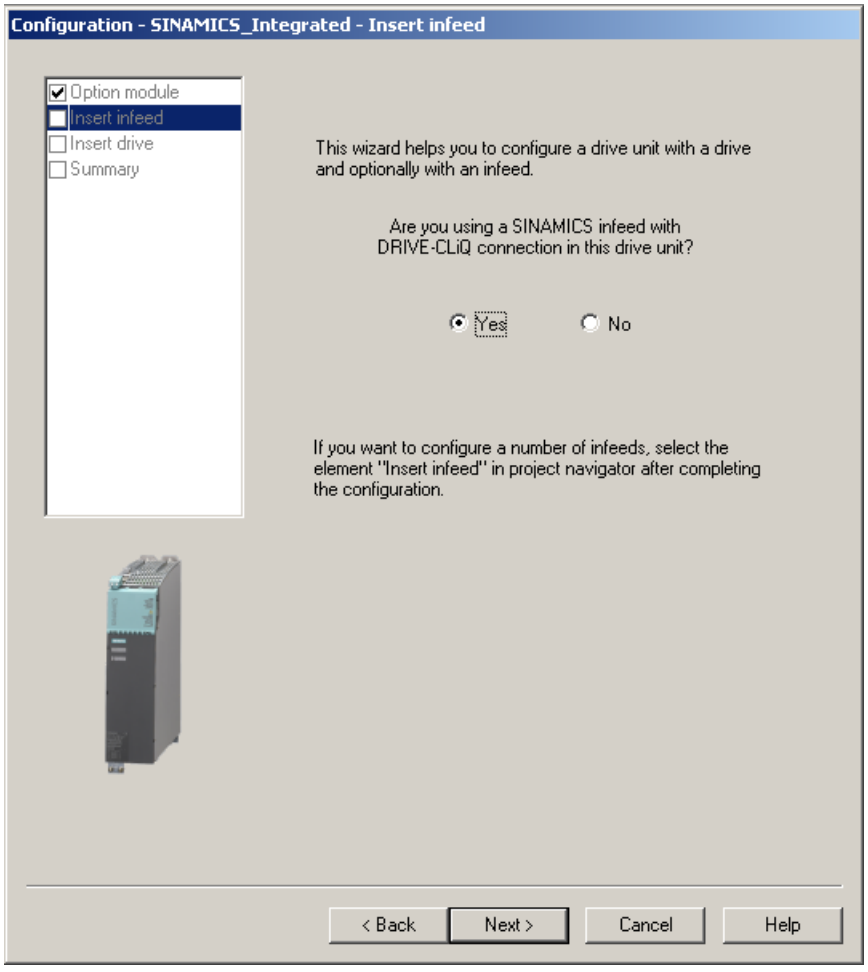


Figure 10-50 Selecting an infeed with DRIVE-CLiQ connection

Note

If you have selected an uncontrolled infeed without a DRIVE-CLiQ connection, omit steps 3 to 6 below.

3. In the "Infeed Configuration" dialog box, enter a name for the drive object and select a type for your infeed (e.g. Active Infeed).

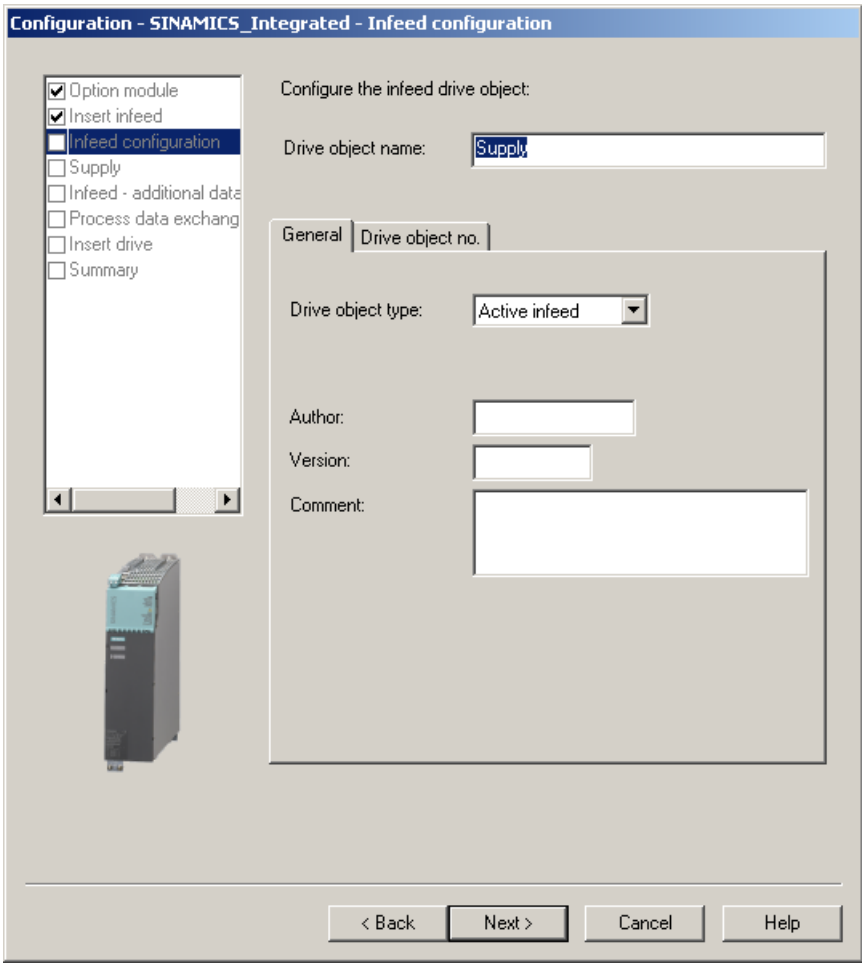


Figure 10-51 Selecting the infeed type

4. Using the article number, select an infeed from the list. You can filter the information to limit the number of infeeds displayed, for example, using "Type".

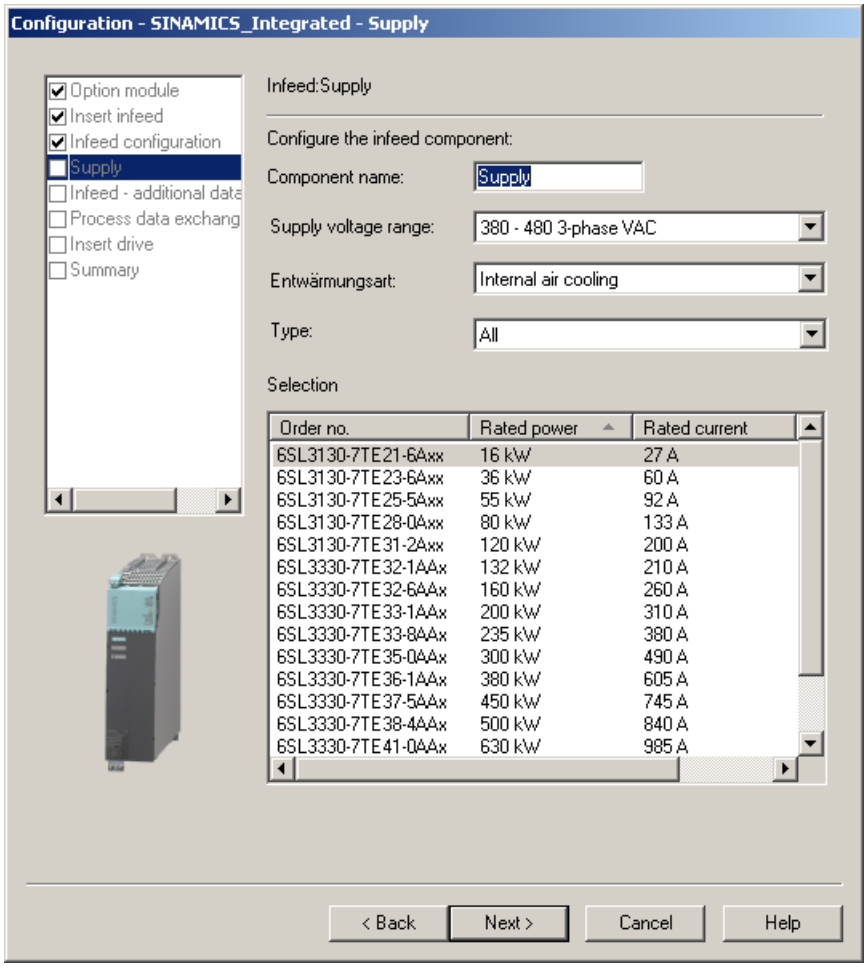


Figure 10-52 Selecting an infeed

5. You can make additional settings for the infeed in the "Infeed Drive Object - Additional Data" dialog box.

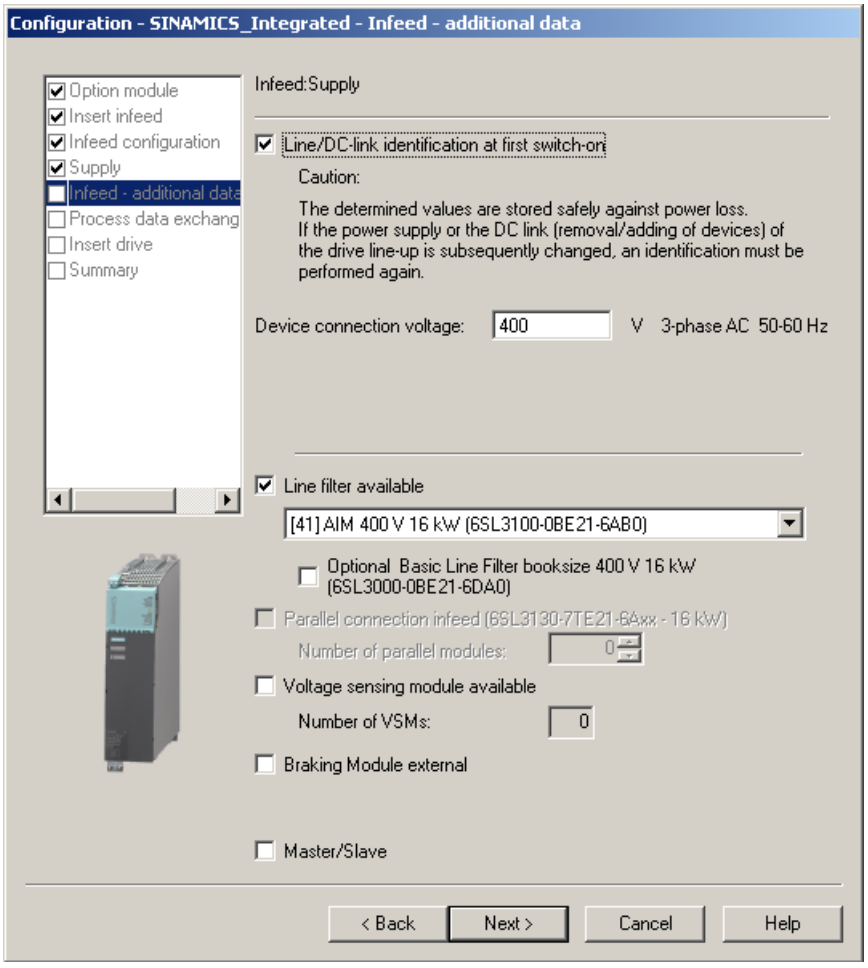


Figure 10-53 Making settings for the infeed

6. The communication for the control of the infeed is configured in the following dialog box. It is recommended that these settings be made automatically by the engineering system. You can also make the settings manually for the process data exchange by selecting "User-defined".

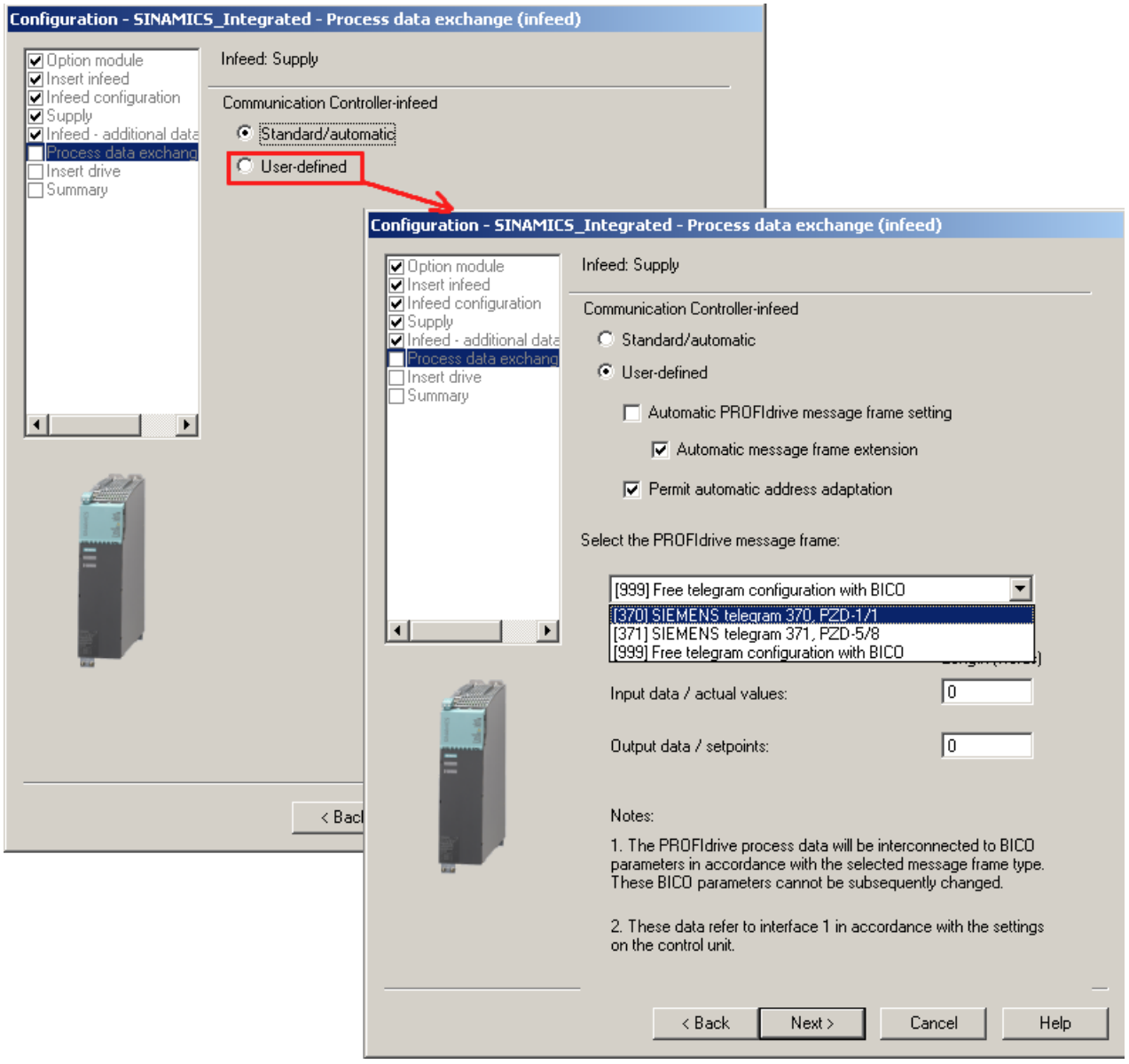


Figure 10-54 Configuring the process data exchange for the infeed

If the automatic communication setting has been selected, SIMOTION SCOUT uses PROFIdrive telegram 370 by default. This telegram is also used by the system function **_LineModule_control** to control the infeed.
You can find further information on control of the infeed in Section Activating the infeed (Line Module) (Page 7144).
If you are using a CX32-2 controller extension, also see the information in Section Interconnecting the infeed "Operation" signal on the CX32-2 (Page 7098).

7. Configure the drive.

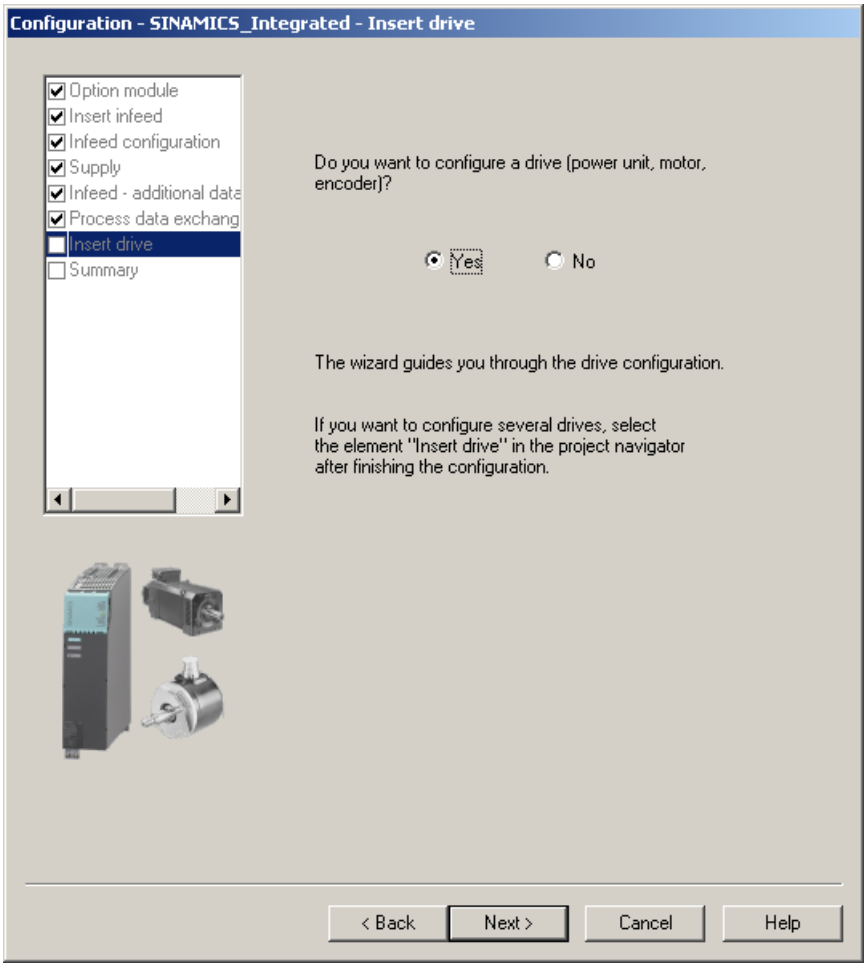


Figure 10-55 Configuring a drive

8. Enter a name for the drive and select the type of drive object (servo or vector).

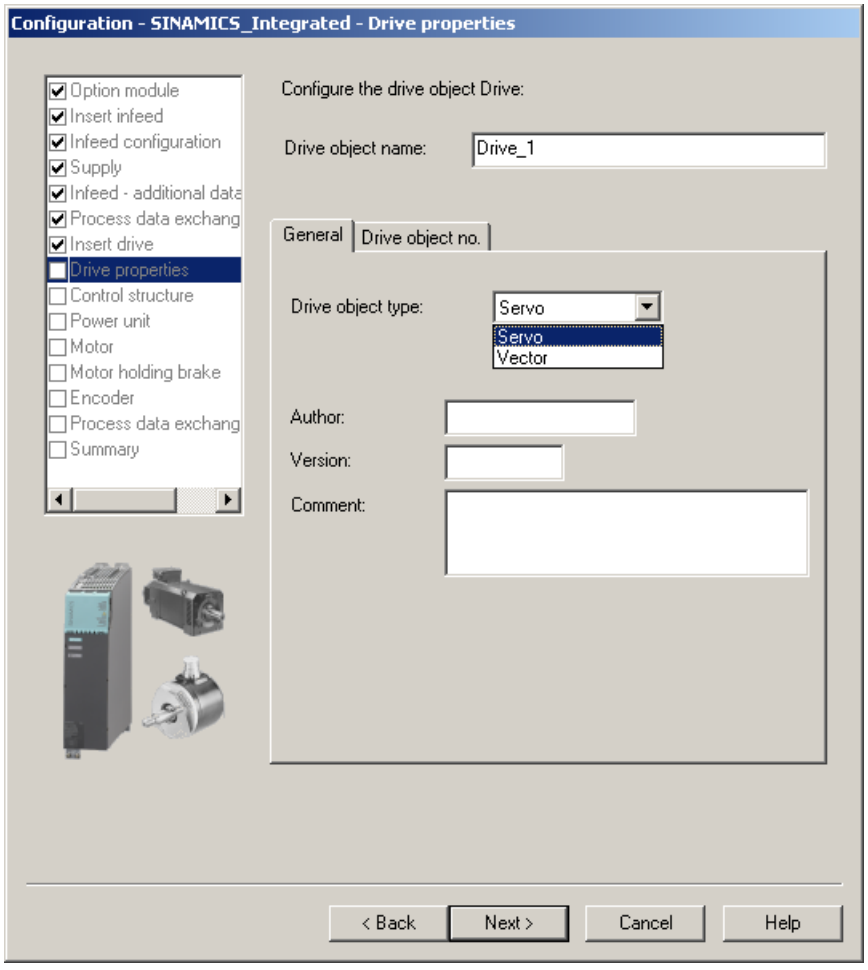


Figure 10-56 Drive properties

9. In the "Control Structure" dialog, you can select the function modules and the control type. Here, you can select the *VIf* control under drive objects type "vector".

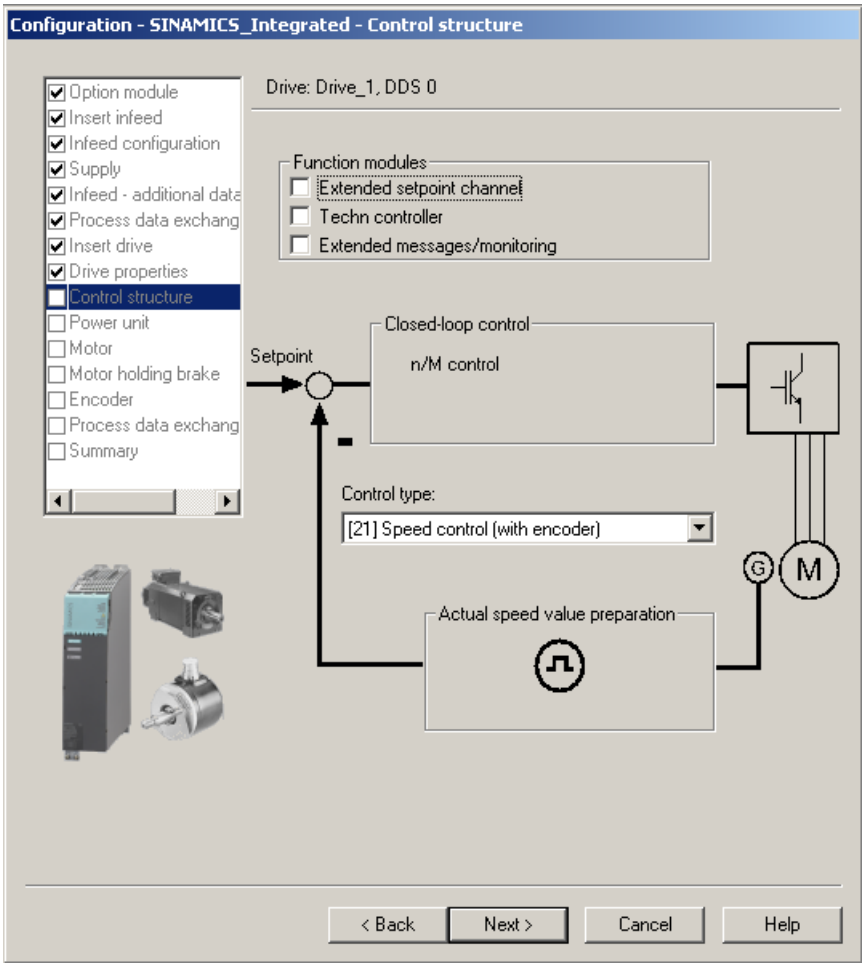


Figure 10-57 Control structure

10. In the "Power Unit" dialog box, use the article number to select your Motor Module from the list.

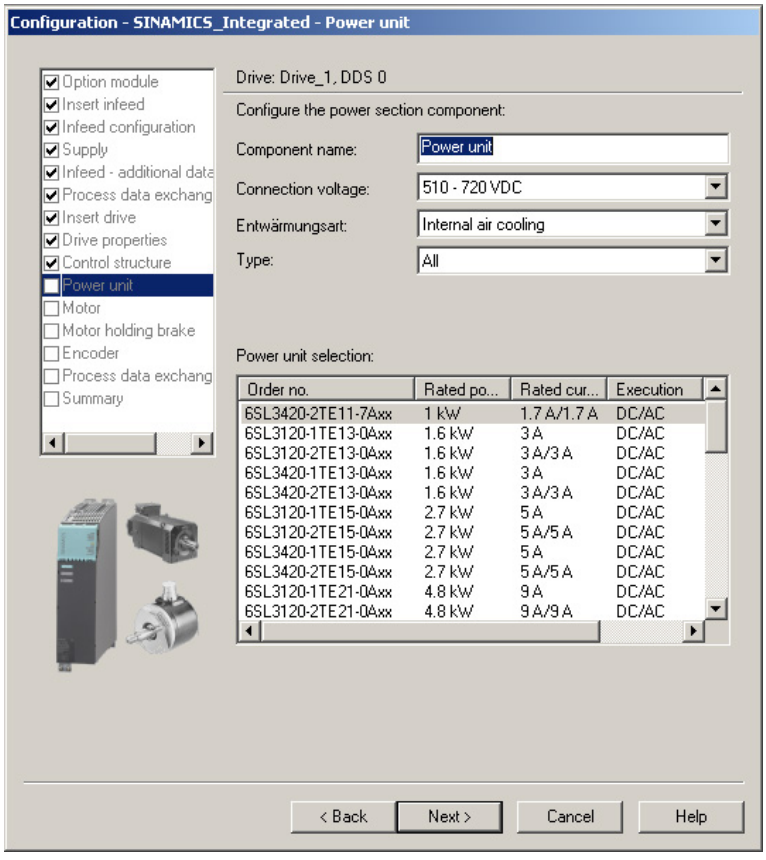


Figure 10-58 Selecting a power unit

11. If you have selected an infeed without a DRIVE-CLiQ connection in step 2, a message prompting you to wire the operation signal will appear. The next dialog box allows you to select the source of the infeed's operation signal.

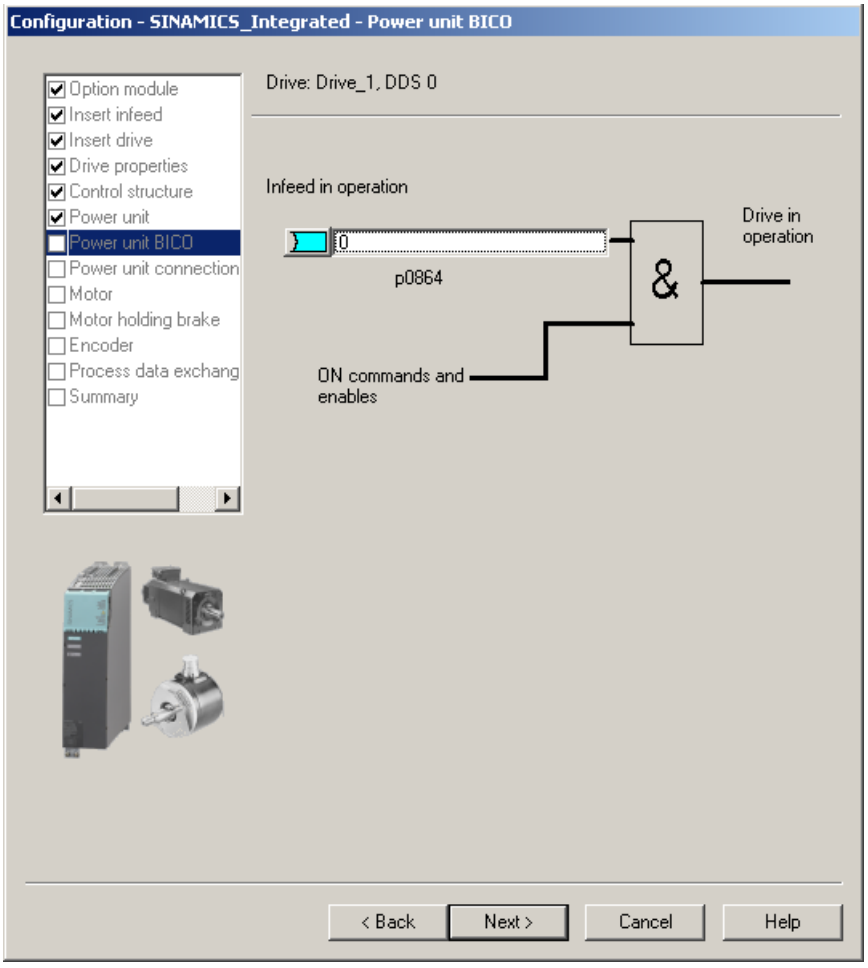


Figure 10-59 Selecting the infeed operation signal

12. In the case of Double Motor Modules, you need to specify the terminal to which the motor is connected.
    You specify the motor in the following dialog boxes:
    - Either by selecting a standard motor from the list
    - Or by entering the motor data

– Or by automatically identifying the motor (motor with DRIVE-CLiQ interface)

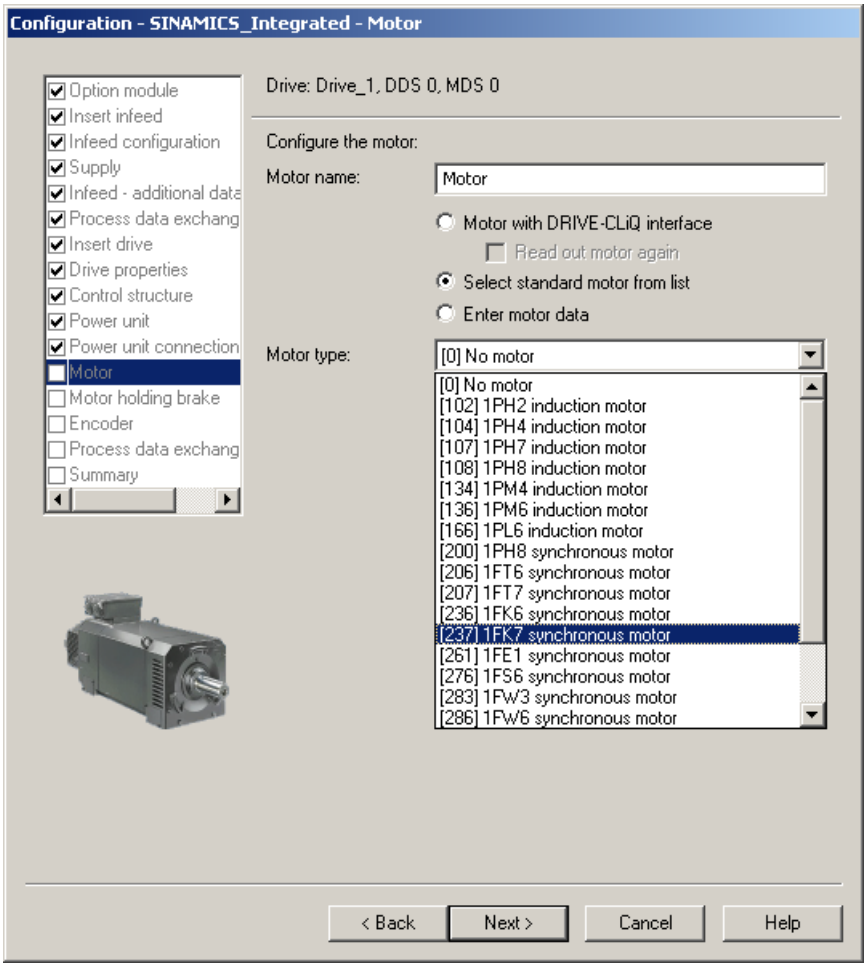


Figure 10-60 Selecting a motor (1)

Note

Motors with DRIVE-CLiQ interface have an integrated encoder evaluation that is connected to the Motor Module via a fully digital communication interface (DRIVE-CLiQ).

In this way, motor encoder and temperature signals as well as electronic rating plate data, such as unique article numbers, rated data (voltage, current, torque) can be transferred directly to the Control Unit.

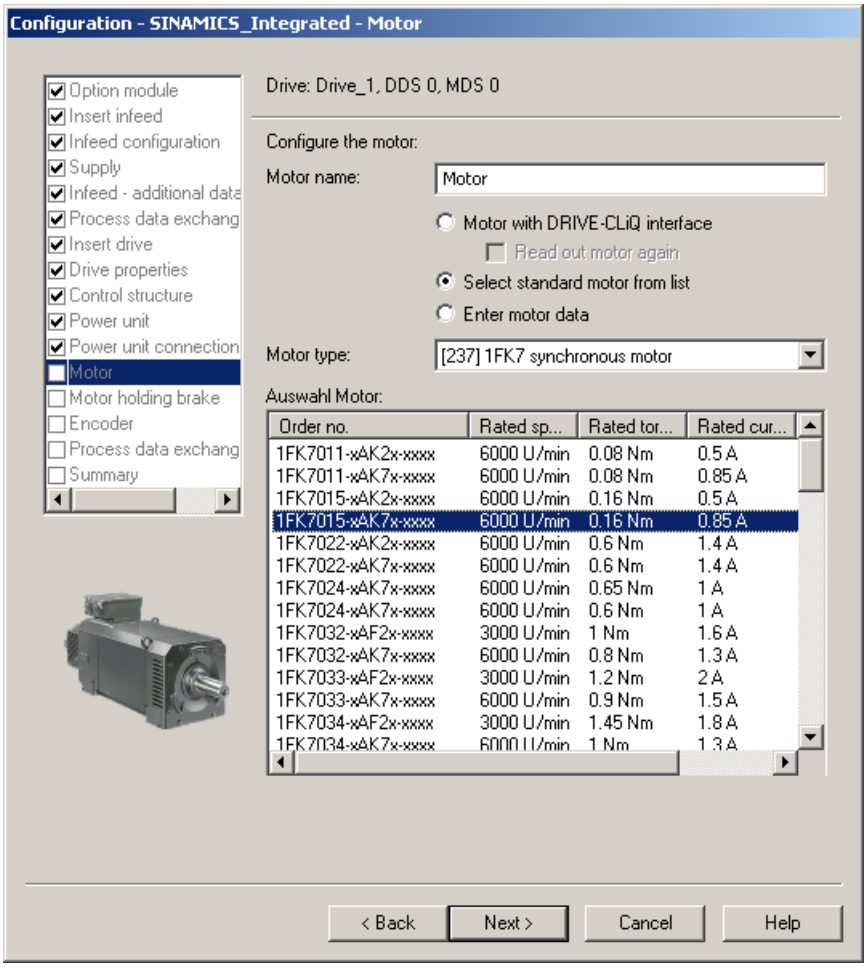


Figure 10-61 Selecting a motor (2)

13. Select a motor holding brake (if installed).

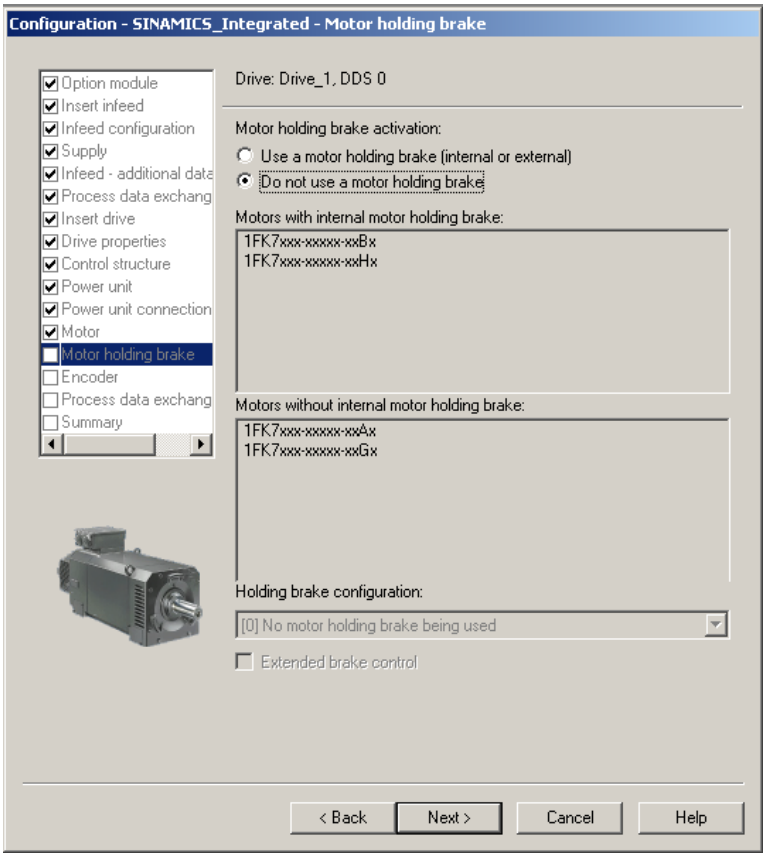


Figure 10-62 Motor holding brake

14. If you are using a motor that is not equipped with a DRIVE-CLiQ interface, select the encoder order number in the "Encoder Selection via Motor Order Number" dialog box.

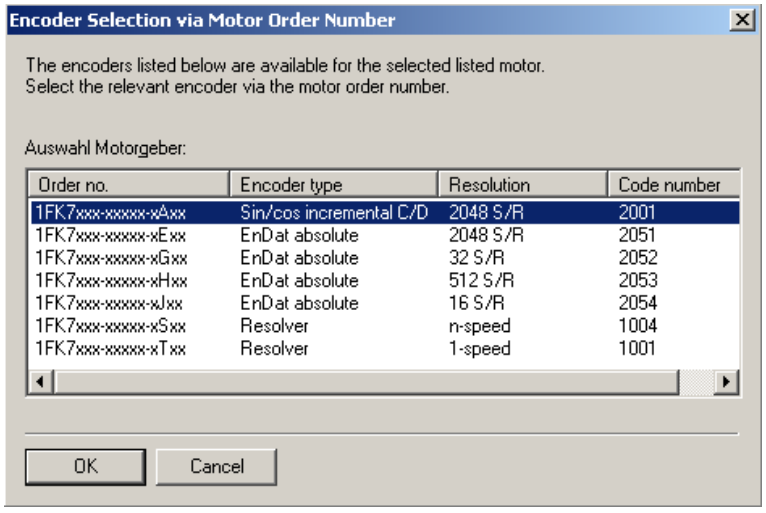


Figure 10-63 Selecting a motor encoder (1)

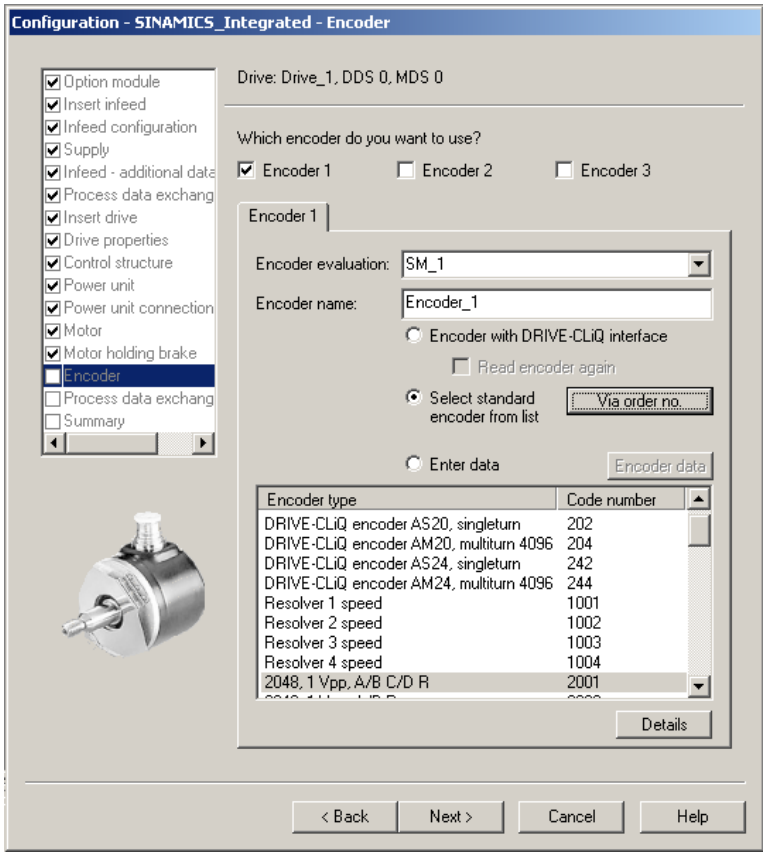


Figure 10-64 Selecting a motor encoder (2)

Note

If required, you can configure a second or third encoder in the "Encoder" dialog box. You can transfer a maximum of two encoder values to SIMOTION via the axis telegram.

In the case of motors with a DRIVE-CLiQ interface, the motor encoder is identified automatically. It is not necessary to enter encoder data in such cases (the dialog for selecting Encoder 1 is grayed out and, therefore, inactive).

15. The communication for the control of the SINAMICS drive is configured in the following dialog box.
It is recommended that these settings be made automatically by the engineering system.
You can also make the settings manually for the process data exchange by selecting "User-defined".
Information about the manual setting options can be found in the online help and in the manuals for the SINAMICS S120 drive system.

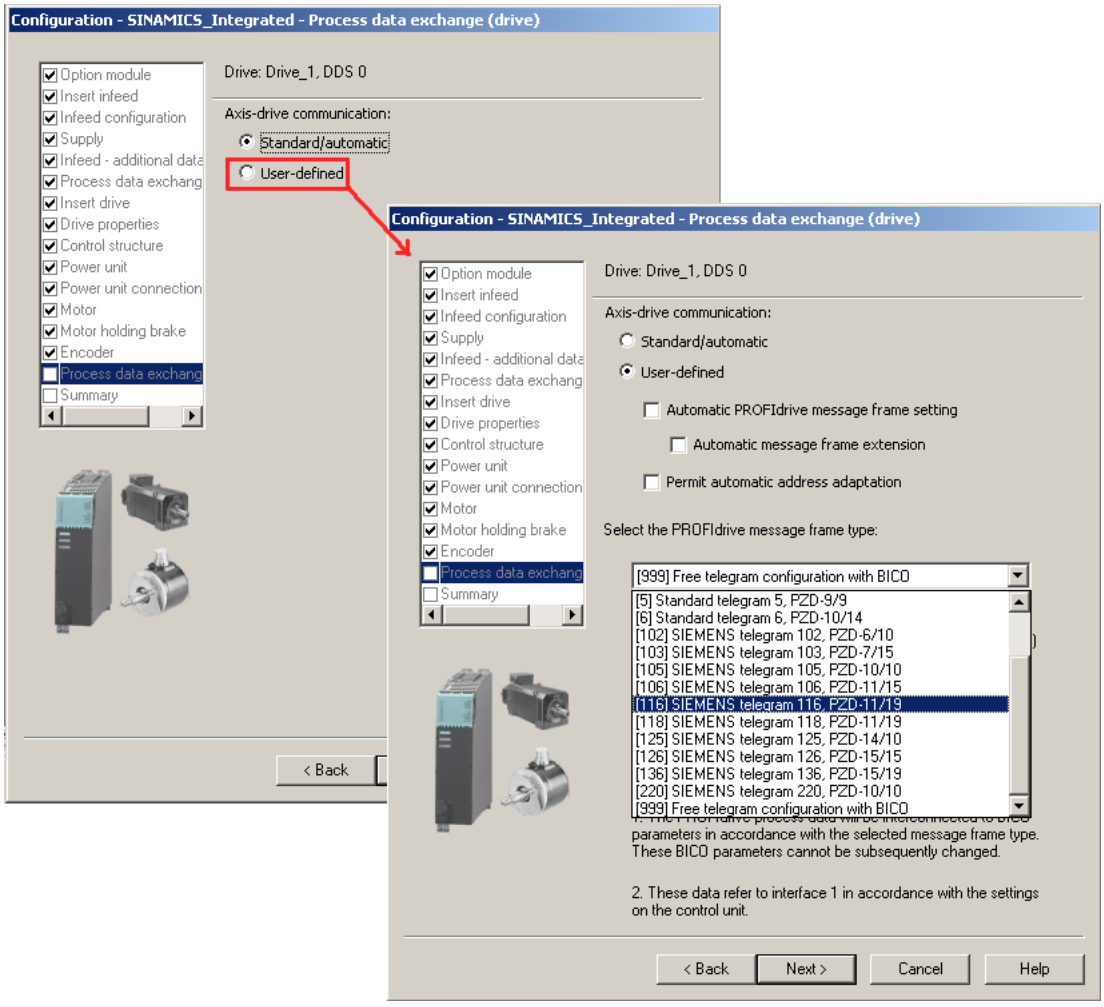


Figure 10-65 Configuring the process data exchange

After you have configured all of the settings in the drive wizard, the "Summary" dialog box displays a list of all settings. You can accept these settings with "Finish" or edit the configuration of individual components using the "Back" button.

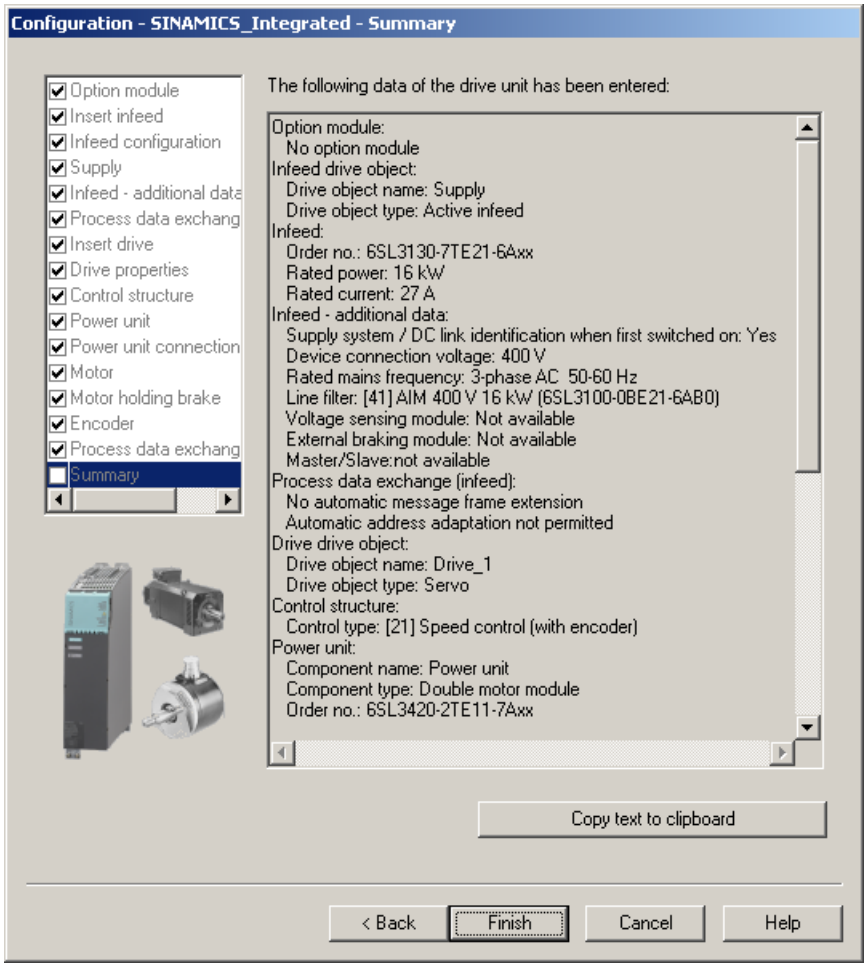


Figure 10-66 Finishing the drive

Result

The configured drive is displayed in the project navigator.

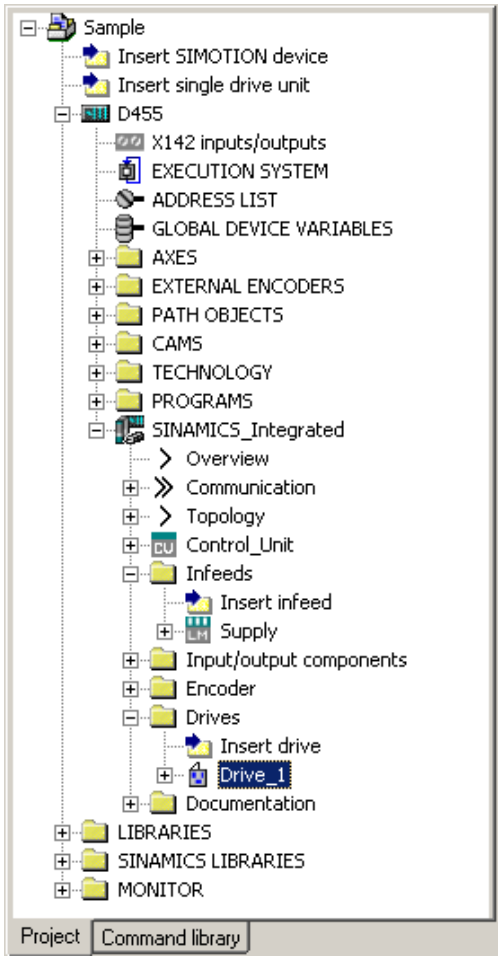


Figure 10-67 Representation in the project navigator

You will find an overview of the configured SINAMICS components under "SINAMICS_Integrated" > "Topology".

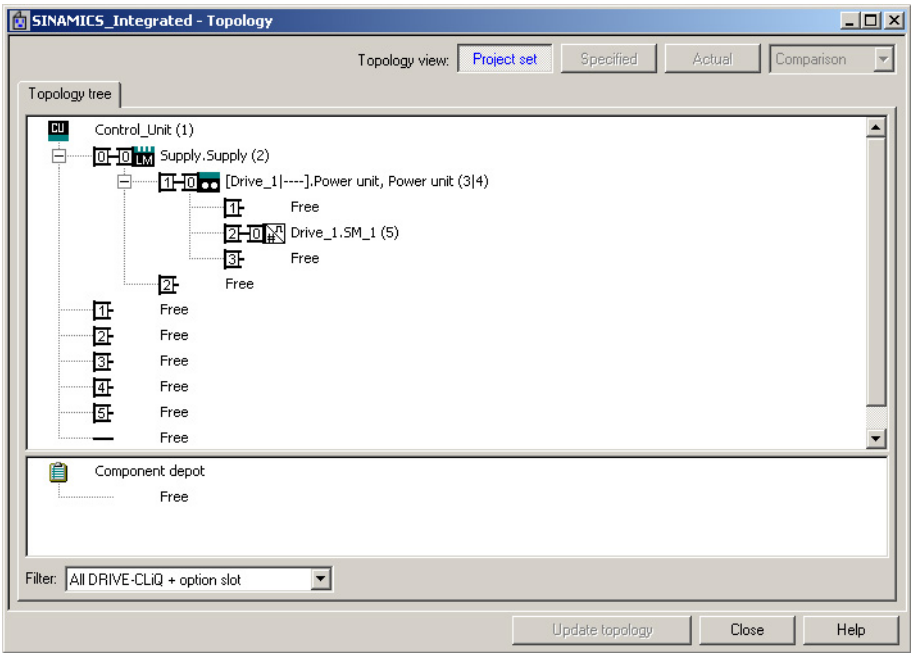


Figure 10-68 Displaying the topology

See also

If you configure the telegrams for the infeed and drive manually, you can find detailed information on the respective telegram types in the following:

- *Motion Control, TO Axis Electrical/Hydraulic*, *External Encoder* Function Manual
- *SINAMICS S120* Function Manual

Downloading a project into the target system

Requirement

You have configured the hardware. You now need to download the hardware configuration and the entire SIMOTION project to the target system.

If you have not yet configured your SIMOTION project (i.e. created ST programs, assigned execution levels, etc.), complete this step first.

Note

You can only perform a project download in the STOP operating mode and for all target devices with which you are ONLINE. You cannot download to drives that cannot be configured in SCOUT, e.g MASTERDRIVES. The project data is loaded to all the devices connected ONLINE and their subordinate drive units (provided these are selected in the Target Device Selection dialog box). This can only be done in the STOP operating mode.

Procedure

1. Save and compile the project.
2. Go online.

3. To load the project, perform "Download project to target system".
   The data must also be saved on the CompactFlash card to ensure that the project is retained in the event of a power failure. The following options are available:
   - Perform the "Copy RAM to ROM..." function manually on the D4x5-2 and all drives (SINAMICS Integrated, CX32-2, etc.).
   - In the "Download to Target System" dialog box, select the option "After loading, copy RAM to ROM". You can change the default setting for this dialog box in **"Options" > "Settings" > "Download"**.

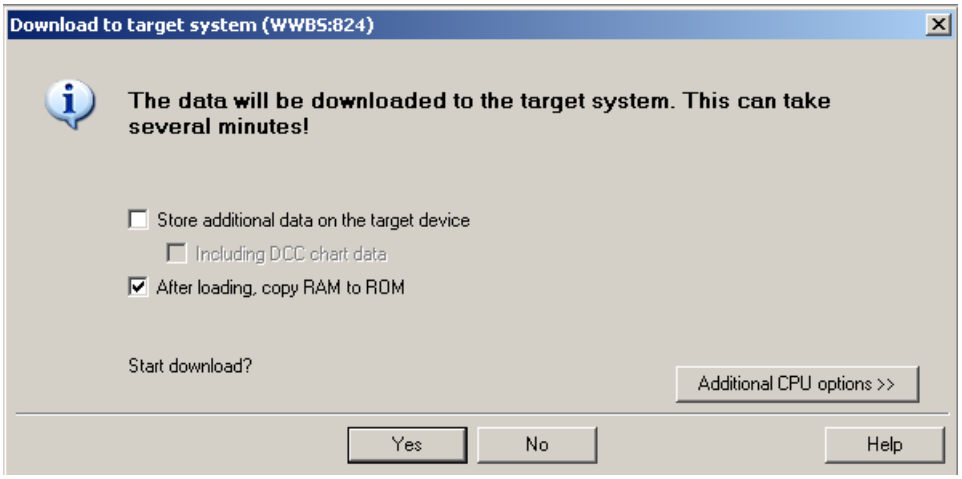


Figure 10-69 Download project to target system

The SIMOTION project is downloaded to the target system and the online connection to the drives established automatically.

Note

Online access to the drives is not possible if HW Config is not loaded at the time you initially connect to the target system.

Download HW Config first to enable online access to the drives. During the "Download to target system" process, SIMOTION SCOUT automatically attempts to establish an online connection to SINAMICS Integrated.

Note

If you have deselected the "Drives" option under "Tools" > "Settings" > "Download" in SIMOTION SCOUT, you must download the configuration separately to each drive (SINAMICS Integrated, CX32-2, etc.).

To do this, select the drive (e.g. SINAMICS Integrated) in the project navigator and perform "Download CPU / drive unit to target device".
For fast operation, we recommend that, in principle, the drives be deselected and a download only performed when required.

4. To save the parameter calculations of the drive in the project, perform "Target device" > "Load CPU / drive unit to PG" for each drive.

Downloading a project to the CompactFlash card

Requirement

Loading a project created offline to the CF card has the advantage of being faster than a download.

It should be noted, however, that the first system power-up will take longer, as the SINAMICS Integrated and CX32-2 controller extension will perform one-off parameter calculations. These are automatically backed up on the CF card.

Note

In order to save the project data via the PG/PC, you need a CF card adapter in which you can insert the CF card. The CF card must be displayed as a removable storage device with any drive letter in Windows Explorer.

If the CF card is not displayed, check the CF card adapter and contact the hotline if necessary.

Procedure

You can use a card adapter to write the entire project to the CF card, even in offline mode. In SIMOTION SCOUT, you can call the "Load to file system" function in the context menu of the SIMOTION device.

1. Save and compile the project.
2. Switch the SIMOTION D4x5-2 off.
3. Remove the CF card and insert it into a card adapter. The card adapter must be connected to a PG/PC.
4. In the SCOUT project, select the SIMOTION D4x5-2 device that you want to download to the CF card.
5. Click "Load to file system" in the context menu. A dialog box opens.
6. In the "Load to File System" dialog box, select the "Save normally" option and click the "Select target" button.
7. Select the target drive.
8. Confirm your entries with "OK". The data is written to the CF card.

Note

With "Load to file system", the project data of the controller (including SINAMICS Integrated / CX32-2) is written directly to the CF card. Separate execution of "Load to file system" for SINAMICS Integrated / CX32-2 is not possible.

9. Remove the CF card and insert it into the slot on the D4x5-2.
10. Switch the D4x5-2 on. The D4x5-2 powers up with the downloaded project.

Note

The components' firmware is upgraded or downgraded automatically based on the FW version on the CF card and the FW version on the SINAMICS components (DRIVE-CLiQ components, CBE30-2, TB30, Power Modules, etc.).

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red/green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are displayed additionally via a yellow RDY LED on the SIMOTION D/CX32-2, indicating that components connected to SIMOTION D/CX32-2 are carrying out a firmware update or that all components have finished their firmware update.

Components that require a POWER ON after the firmware update will indicate this through a rapidly flashing RDY-LED. Go offline with SCOUT and switch the 24 V supply off/on (POWER ON) at the respective components for initialization.

CBE30-2 option board:

During the firmware update, the OPT LED of the SIMOTION D module and the SYNC LED of the CBE30-2 flash green.

Loading a project, including sources and additional data

Overview

It is possible to download additional data (e.g. sources) to the target device when saving a project to the CF card or downloading to the D4x5-2.

These data are required for:

- Online object comparison (e.g. additional properties)
- Various detailed comparisons (e.g. ST source file comparison)
- Synchronization with online objects

In order to be able to load a project's sources and additional data to the PG, this must be specified in the project under "Options" > "Settings" > "Download" > "Store additional data on the target device". Alternatively, this setting can also be made when the data is loaded to the target device/target system.

If the sources and additional data have been saved on the CF card, the option described in the sections that follow becomes available.

Project comparison

You intend to perform servicing work on a commissioned system and have placed a project on your PG/PC. This project is not consistent with the project on the D4x5-2 in the system. In order to analyze the differences, perform an object comparison via "Start object comparison".

You have the following options in terms of re-establishing consistency:

- In the object comparison, you can establish consistency for sources and technology objects on an object-granular basis.
- Consistency can be established for the entire Control Unit by loading from the CF card via "Target system" > "Load" > "Load CPU / drive unit to PG...."

Additional references

Detailed information on loading data to the target device can be found in the *SIMOTION Runtime Basic Functions* Function Manual.

Archiving a project to the CompactFlash card (zip file)

Procedure

In SIMOTION SCOUT, you can save the project as a zip file to the CF card.

Proceed as follows to archive the SIMOTION project on the CF card:

1. Open SIMOTION SCOUT and select the "Project" > "Archive" menu command.
2. In the "Archive" dialog box, select the SIMOTION project and save it to your drive (PG/PC).
3. Open the project.
4. Go online with the SIMOTION D4x5-2.
5. In the project navigator, select the SIMOTION D4x5-2 and select the "Target system" > "Load" > "Save archive project on card ..." menu command.
6. In the dialog that is displayed, select the project and click "Open".
   This saves the project to the CF card as Project.zip in the following directory: USER\SIMOTION \HMI\PRJLOG

Note

If you want to load the current project from the card, select the "Target system" > "Copy archived project from card to PG/PC ..." menu command.

Prerequisite is that you have backed up the project with "Save archive project on card..." each time a change was made.

Additional references

Detailed information on loading data to the target device can be found in the *SIMOTION Runtime Basic Functions* Function Manual.

Performing an online configuration for the D4x5-2

Overview

Introduction

You can configure the plant in online mode after having completed its wiring. You can load the data of SINAMICS components connected via DRIVE-CLiQ to your PG/PC using the "Automatic configuration" function. However, this is only possible for initial commissioning.

Note

Components without DRIVE-CLiQ connection must be edited in offline mode. You may need to edit DRIVE-CLiQ components which were detected in the course of automatic configuration (for example, adding encoder data if using SMC modules).

Requirements

- Your system has been mounted and wired
- You have created a project in SIMOTION SCOUT and inserted a SIMOTION D4x5-2 in the project
- You have configured the communication between the SIMOTION D4x5-2 and the PG/PC. See Section Creating a project and configuring the communication (Page 7007).

Procedure

The online configuration involves the following steps:

- Establish the online connection (Page 7080)
- Starting automatic configuration (Page 7081)
- Reconfiguring SINAMICS components (Page 7085)
- Downloading a project to the SIMOTION D4x5-2 (Page 7086)

Establish an online connection

Requirement

You have created a project.

Procedure

This section outlines the procedure for initial commissioning.

To perform an online configuration, you must establish an online connection to the SIMOTION D4x5-2. In this case, no connection can yet be established to the SINAMICS

Integrated. An appropriate message is output. Once the hardware configuration has been loaded to the target device, an online connection to the SINAMICS Integrated is established automatically. Proceed as follows:

1. Save and compile the project.
2. Establish an online connection.
3. Select the SIMOTION D4x5-2 device in the project navigator.
4. Use the "Download CPU / drive unit to target device" function to download the SIMOTION D4x5-2 to the target device. The connection to the SINAMICS Integrated is established automatically.

Result

You can now run **automatic configuration** on the SINAMICS Integrated.
See Section Starting the automatic configuration (Page 7081).

Additional references

Further information about establishing an online connection to the programming device/PC can be found in the following documentation:

- *SIMOTION SCOUT* Configuration Manual
- *SIMOTION SCOUT* Online Help
- SIMOTION Utilities & Applications FAQs
  SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.
- FAQOnline connection to SIMOTION (https://support.industry.siemens.com/cs/ww/en/view/22016709)

Starting automatic configuration

Requirements

You have activated the online connection with the SINAMICS Integrated.

Procedure

1. In the project navigator, open the "Automatic Configuration" dialog box by selecting "SINAMICS Integrated" > "Automatic configuration".

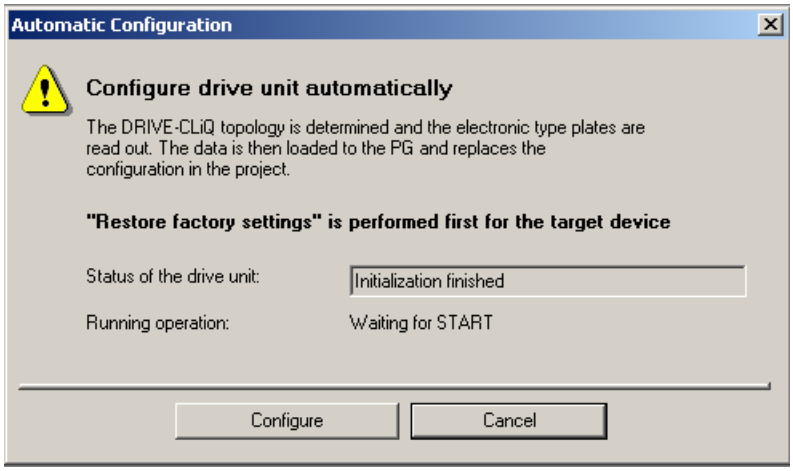


Figure 10-70 Starting automatic configuration

2. Click the "Configure" button.
3. If the drive unit is not in the "First commissioning" state, the factory settings are restored after acknowledging a prompt.
4. The drive object types can now be selected via a further dialog box.

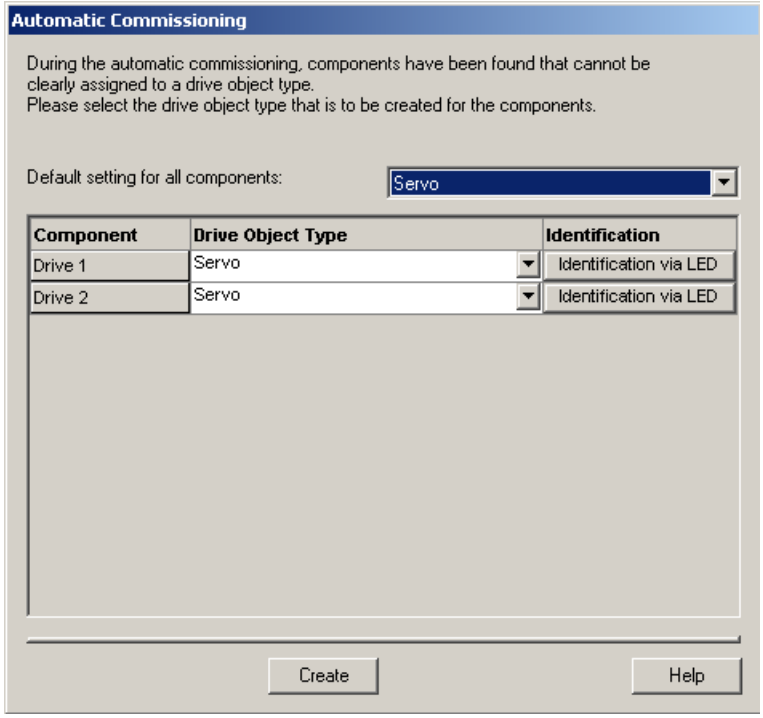


Figure 10-71 Selecting the drive object type

5. Select whether a servo- or vector-type drive object is to be used.

6. Click "Create" to start the automatic configuration. As soon as automatic commissioning has been run through, an upload operation (Load to PG) is performed automatically.

Note

The components' firmware is upgraded or downgraded automatically based on the firmware version on the CF card and the firmware version on the SINAMICS components (DRIVE-CLiQ components, TB30, CBE30-2, Power Modules, etc.).

The update procedure can take several minutes and is indicated in the "Automatic Configuration" dialog box by the following message:

State of the drive unit: Automatic FW update of DRIVE-CLiQ components

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2 and indicate that a firmware update is being carried out on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

CBE30-2 option board:

During the firmware update, the OPT LED of the SIMOTION D module and the SYNC LED of the CBE30-2 flash green.

7. At the end of the automatic configuration, you are prompted whether you want to "Go OFFLINE" or "Stay ONLINE" with the drive unit.

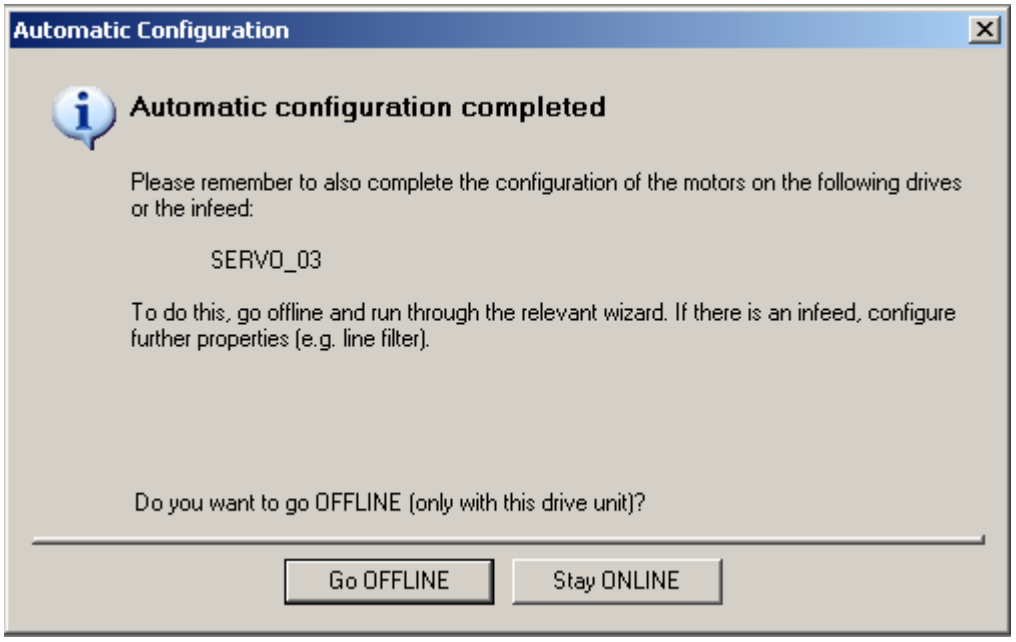


Figure 10-72 Automatic configuration completed

8. Execute the "Copy RAM to ROM ..." function on the D4x5-2 and on the SINAMICS Integrated. This saves the project on the CF card so that it does not need to be reloaded after switching off and on.

Result

The DRIVE-CLiQ components loaded to the user project by means of automatic configuration are displayed in the project navigator.

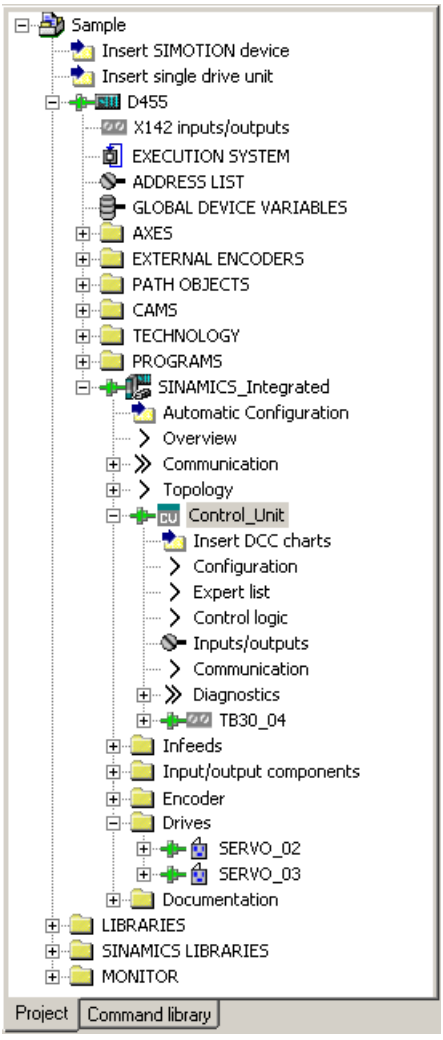


Figure 10-73 Project navigator with real configuration

You must then

- If required, reconfigure SINAMICS components (e.g. components without a DRIVE-CLiQ interface, such as an encoder connected via SMCxx).
- Assign the "TO axis" to "Drive"

Editing SINAMICS components

Requirement

- You have loaded all connected DRIVE-CLiQ components to your user project.
- You have shut down the connection to the target system (offline mode).

Procedure

You can now adapt your components to the application.

Run through the wizard for all the DRIVE-CLiQ components to be adapted and perform the required reconfigurations.

This procedure corresponds with the description in Performing the configuration for the D4x5-2 offline (Page 7052).

The amount of editing work involved depends on the components used. For example, in the case of a motor with a DRIVE-CLiQ interface, the motor and encoder type are identified automatically.

Downloading a project to the SIMOTION D4x5-2

After you have performed the reconfigurations, you must download the configuration to the SIMOTION D4x5-2 (incl. SINAMICS Integrated).

1. Save and compile the project.
2. Go online.

3. To load the project, perform "Download project to target system".
   The data must also be saved on the CompactFlash card to ensure that the project is retained in the event of a power failure. The following options are available:
   - Perform the "Copy RAM to ROM..." function manually on the D4x5-2 and all drives (SINAMICS Integrated, CX32-2, etc.).
   - In the "Download to Target System" dialog box, select the option "After loading, copy RAM to ROM". You can change the default setting for this dialog box in **"Options" > "Settings" > "Download"**.

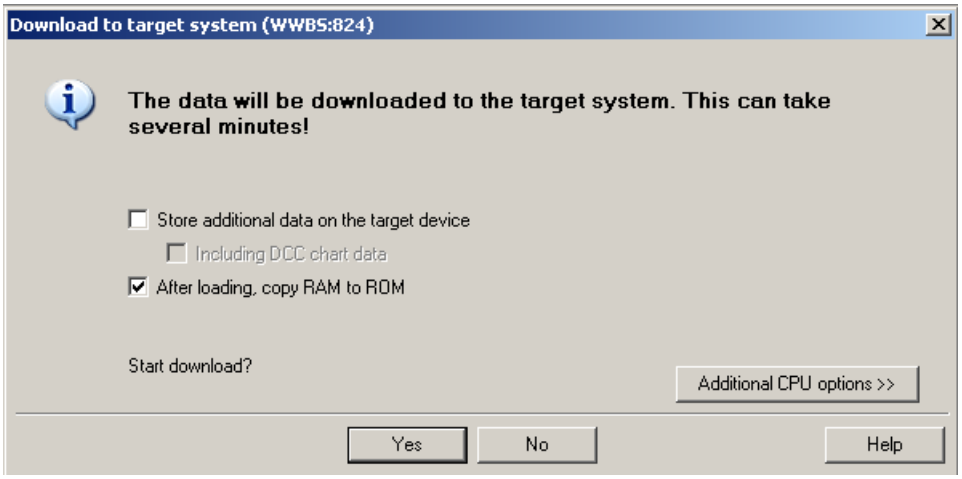


Figure 10-74 Download project to target system

The SIMOTION project is downloaded to the target system and the online connection to the drives established automatically.

Note

Online access to the drives is not possible if **HW Config** is not loaded at the time you initially connect to the target system.

Download the data to **HW Config** in order to enable online access to SINAMICS Integrated. During the "Download to target system" process, SIMOTION SCOUT automatically attempts to establish an online connection to SINAMICS Integrated.

Note

If you have deselected the "Drives" option under "Tools" > "Settings" > "Download" in SIMOTION SCOUT, you must download the configuration separately to each drive (SINAMICS Integrated, CX32-2, etc.).

To do this, select the drive (e.g. SINAMICS Integrated) in the project navigator and perform "Download CPU / drive unit to target device".
For fast operation, we recommend that, in principle, the drives be deselected and a download only performed when required.

4. To save the parameter calculations of the drive in the project, perform "Target device" > "Load CPU / drive unit to PG" for each drive.

The drive has been assigned parameters and commissioned. You can now test the drive via the drive control panel.

Configuring a CX32-2

Overview

The SIMOTION CX32-2 controller extension is a component in SINAMICS S120 booksize format and enables scaling of the drive-side computing performance of the SIMOTION D4x5-2 control units. Each CX32-2 can operate up to six additional servo, six vector or twelve *V/f* drives.

The CX32-2 controller extension offers the following advantages:

- With a width of only 25 mm, the CX32-2 is extremely compact.
- The CX32-2 is connected to the SIMOTION D4x5-2 via DRIVE-CLiQ, so that high-performance, isochronous closed-loop control of the drives is possible without the need for additional modules.
- The communication interfaces on the SIMOTION D4x5-2 remain available for other connections.
- The data for the CX32-2 is stored exclusively on the SIMOTION D4x5-2 CompactFlash card, which means that no action needs to be taken when the module is replaced.
- The "Control operation" signal from an infeed connected to the SIMOTION D4x5-2 is particularly easy to interconnect to the drives of a CX32-2.
- SIMOTION CX32-2 is automatically also upgraded when the SIMOTION D4x5-2 is upgraded

Basic principles of the CX32-2

Basic principles of the CX32-2

Each CX32-2 is entered as a separate DRIVE object in the project navigator on the level of SINAMICS Integrated. In principle, a CX32-2 can be configured the same way as SINAMICS Integrated.

Communication

With CX32-2 (as with the SINAMICS Integrated), communication takes place via the PROFIBUS Integrated, but is still routed by the SINAMICS Integrated to the relevant CX32-2 via DRIVE-CLiQ.

Communication with the CX32-2 runs in the same cycle (same DP cycle) as communication with the SINAMICS Integrated. Direct communication between two CX32-2 modules or the SINAMICS Integrated and CX32-2 is not possible. Exception: Interconnection of the "Operation" signal of the infeed for the CX32-2. For a description of how to interconnect to the "Operation" signal, refer to Section Interconnecting the infeed "Operation" signal on the CX32-2 (Page 7098).

Power-up

The CX32-2 does not require its own CompactFlash card. The firmware and the parameterization are stored centrally on the SIMOTION D CompactFlash card. After a firmware change, the CX32-2 firmware from the CompactFlash card is copied to the CX32-2 at the first power-up and saved there as non-volatile data.
For this reason, the first power-up with new firmware takes longer than subsequent power-ups.

Copying from RAM to ROM also takes longer, as the devices perform this function one after the other.

How to detect if a power-up has been completed is described in Section *CX32-2 configuration information* in CX32-2 power-up (Page 7097).

Online functions

Since the SINAMICS Integrated routes online functions to the CX32-2 via DRIVE-CLiQ, operational performance is reduced in the case of online functions (e.g. parameter changes or downloads) compared to a similar installation with CU320-2.

Commissioning requirements

To commission a SIMOTION D with CX32-2, all drive components which were configured offline must also be actually available and connected to the correct DRIVE-CLiQ port.

Only when there are no topology errors

- Will the download be successful (where a project created offline is loaded to the target system)
- Will the system startup be successful

To be able to go online on a CX32-2, at least the **HW Config** configuration must be loaded to the D4x5-2 and the CX32-2 connected "correctly" via the configured DRIVE-CLiQ ports on the D4x5-2.

Loading the SINAMICS Integrated

If the SINAMICS Integrated of a SIMOTION D is loaded, the connected CX32-2 controller extensions go offline as a result of reinitialization.

Preparing for configuration

Preparing for configuration

The following measures are required to enable online or offline configuration of a CX32-2.

1. Create a project and insert a SIMOTION D4x5-2 (in this example, D455-2).
2. In the project navigator, double-click the D4x5-2. The **HW Config** appears.
3. In the hardware catalog, open the "PROFIBUS DP" entry and select "SINAMICS".

4. Drag a CX32-2 to the PROFIBUS Integrated master system of the SIMOTION D module. The mouse pointer permits the CX32-2 to be inserted on the master system only. The "DP Slave Properties" screen form is displayed with the PROFIBUS address. The PROFIBUS address is assigned automatically. See the table below.
   Select the relevant DRIVE-CLiQ port by selecting the relevant PROFIBUS address (in this example, PROFIBUS address 15).

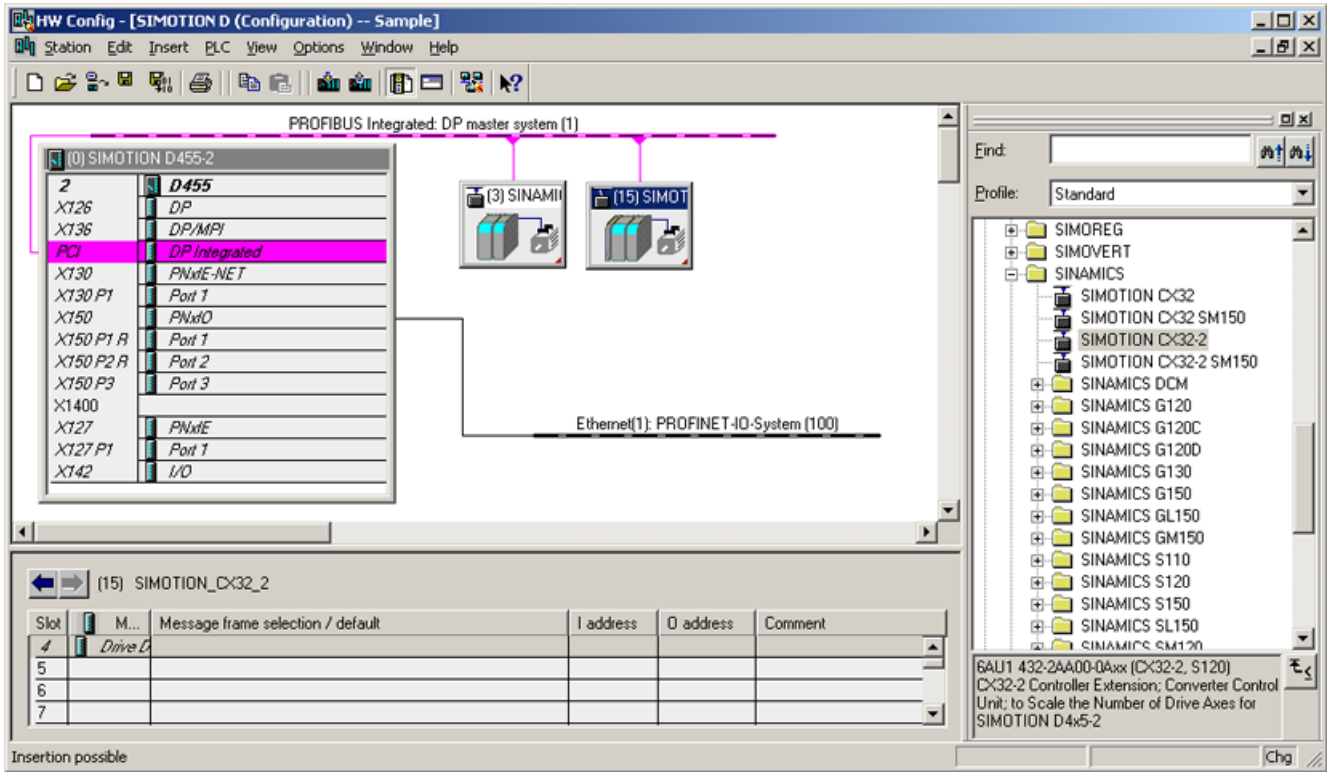


Figure 10-75 Inserting the CX32-2 in HW Config

5. Click "OK" to confirm your settings.
   The CX32-2 is displayed in the project navigator and can be configured there in the same way as a SINAMICS Integrated.
   In **HW Config**, the respective PROFIBUS address of the CX32-2 is displayed in parentheses in the module icon.

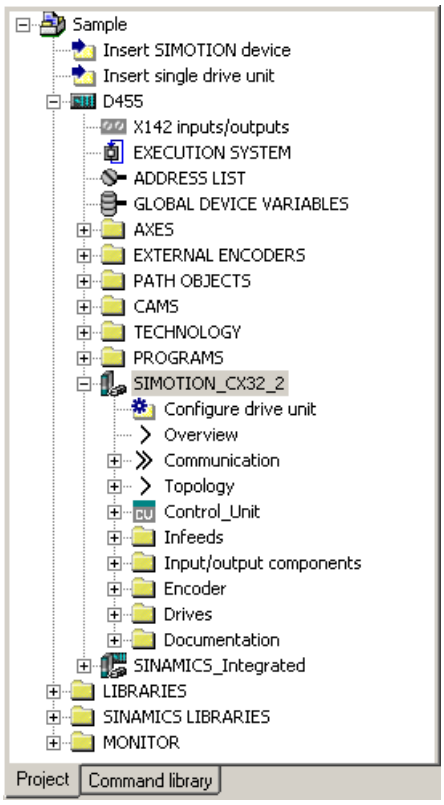


Figure 10-76 CX32-2 in the project navigator

6. Save the project.

7. A CX32-2 is configured in the same way as the SINAMICS Integrated on a SIMOTION D4x5-2.
   Using an offline configuration
   Where configuration is being performed offline, configure the drive components (infeed, motor modules, motor, encoder, terminal modules, etc.) by running through the drive wizards for the SINAMICS Integrated or CX32-2. After you have done this, carry out commissioning.
   Automatic commissioning
   During the automatic commissioning, the drive components that are physically connected to a SIMOTION D are determined. I.e. you only require a D4x5-2 project with the CX32-2 modules configured in **HW Config**.

Assigning DRIVE-CLiQ ports to PROFIBUS addresses

Table 10-42 CX32-2 PROFIBUS addresses (PROFIBUS Integrated)

DRIVE-CLiQ port	PROFIBUS address (PROFIBUS Integrated)
X105 (not for D425-2)	15
X104 (not for D425-2)	14
X103	13
X102	12
X101	11
X100	10

Displaying the topology

Topology of the SINAMICS Integrated

Because the CX32-2 is connected to the SINAMICS Integrated of a SIMOTION D4x5-2 via DRIVE-CLiQ, it is also displayed in the topology tree of the SINAMICS Integrated.

All inserted CX32-2s are displayed in the SINAMICS Integrated topology without their subtopology.

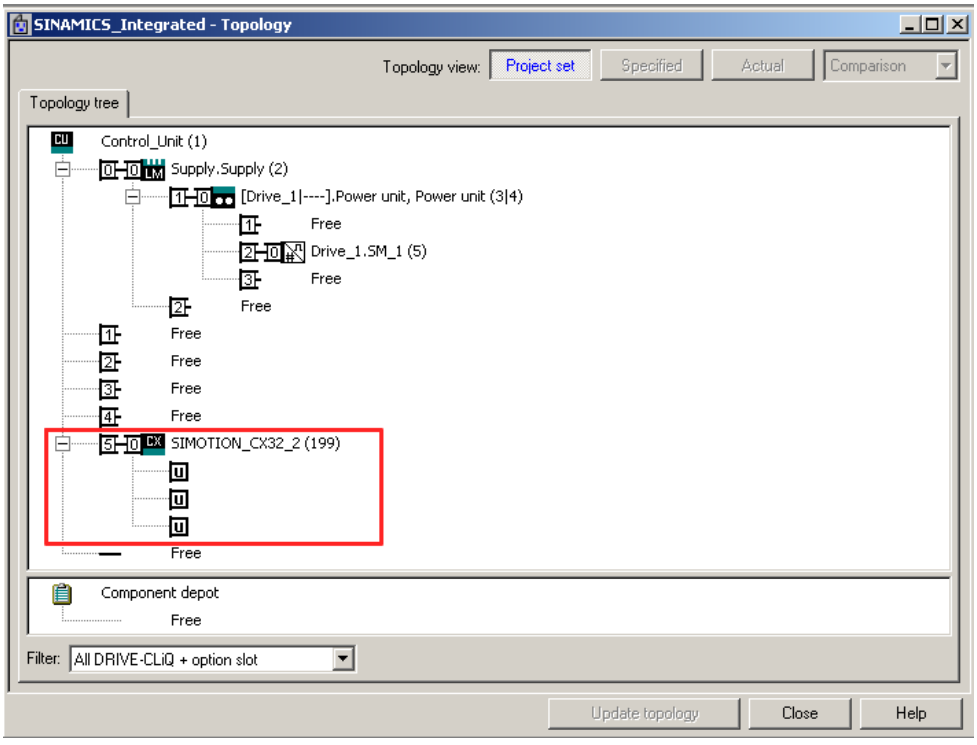


Figure 10-77 SINAMICS Integrated topology (example of D455-2 DP/PN)

CX32-2 topology

The CX32-2 topology displays the DRIVE-CLiQ port required for connection to the SINAMICS Integrated. The drive objects connected to the CX32-2 are also displayed.

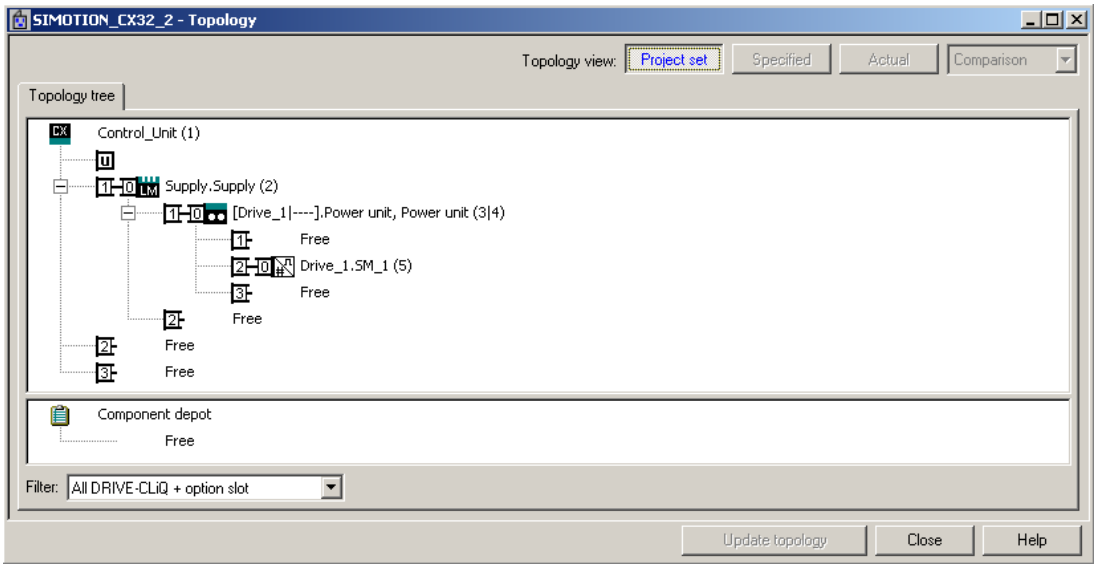


Figure 10-78 CX32-2 topology

Note

Topology errors can result in a download or system power-up not being possible.

CX32-2 offline configuration

Options

The following options are available for commissioning a project created offline:

- Loading a project created offline to the target system
- Loading a project created offline to the CompactFlash Card

Loading a project created offline to the target system

Requirement

- Your project, including all SINAMICS Integrated and CX32-2 drive components, must have been configured already (see Preparing for configuration (Page 7089)).
- The actual topology must correspond with the reference topology.

Note

A step-by-step commissioning with the CX32-2 controller extension is described below.

In a step-by-step procedure, first the D4x5-2, then the SINAMICS Integrated and finally each CX32-2 is commissioned. This step-by-step procedure is not mandatory, but has the advantage that the location and cause of an error are easier to identify.

Procedure

1. Deselect all target devices, with the exception of the SIMOTION D4x5-2, using the "Target system" > "Select target devices" menu command in SCOUT.
2. Go online with the D4x5-2 by clicking "Connect to selected target devices".
3. Select the D4x5-2 in the project tree. Load the configuration to the SIMOTION D4x5-2 with "Download CPU / drive unit to target device".
4. Copy the configuration from RAM to ROM with "Copy RAM to ROM".
5. Select "Connect target device" in the SINAMICS Integrated context menu. You will then be connected to the SINAMICS Integrated online.
6. Load the parameterization by selecting "Target device" > "Download" in the SINAMICS Integrated context menu.
7. Copy the parameterization from RAM to ROM with "Copy RAM to ROM".
8. Then select "Load CPU / drive unit to PG" to load the parameterization from the SINAMICS Integrated back to the programming device in order to back up the SINAMICS parameter calculations.

The SINAMICS Integrated is now ready for operation (RDY LED flashes green at 0.5 Hz). The CX32-2 can be accessed online, although it has not been configured yet (RDY LED flashes green at 0.5 Hz, DP LED is off).

1. Select "Connect online" in the CX32-2 context menu. You will then be connected to the CX32-2 online.
2. Load the parameterization to the CX32-2 and copy the parameterization from RAM to ROM.
3. Then load the parameterization from the CX32-2 back to the programming device in order to back up the SINAMICS parameter calculations.

4. Repeat steps 1 to 3 for each CX32-2.
5. Save the project.

Note

Topology errors can result in a download or system power-up not being possible.

Result

The CX32-2 is now also ready for operation (RDY LED is green, DP LED is green (in RUN mode) or flashes green at 0.5 Hz (in STOP mode)).

Loading a project created offline to the CF card

Loading a project created offline to the CF card has the advantage of being faster than a download.

It should be noted, however, that the first system power-up will take longer, as the SINAMICS Integrated and CX32-2 will perform one-off parameter calculations. These are automatically backed up on the CF card.

Requirement

- You will need a card reader for the SIMOTION D4x5-2 CF card.
- Your project, including all SINAMICS Integrated and CX32-2 drive components, must have been configured already (see Preparing for configuration (Page 7089)).
- The actual topology must correspond with the reference topology.

Procedure

1. Select "Load to file system" in the SIMOTION D4x5-2 context menu.
2. Select "Normal save" and click the "Select target" button. Select your card reader's drive and confirm with "OK".
3. If a project has already been saved to the CF card, the message "Memory card file already exists. Do you want overwrite this file?" appears. Confirm the message with "Yes".
4. Once the project has been subsequently transferred to the CF card, insert the card into the disconnected SIMOTION D4x5-2. Switch on the power supply of the D4x5-2 and the connected components.
5. After a successful power-up, the RDY LED of the D4x5-2 and the CX32-2 will be green, as will the DP LED of the CX32-2.

The components' firmware is automatically updated, depending on the FW version on the SINAMICS components and on the CF card. Note the information in Section Upgrading the D4x5-2/CX32-2 (Page 7100).

Note

Topology errors can result in a download or system power-up not being possible.

Performing an online configuration for the CX32-2

Requirement

You need a project which has already been created and which contains the SIMOTION D4x5-2 with SINAMICS Integrated, as well as the CX32-2 controller extension(s).

Additional drive components (Line Module, Motor Modules, Terminal Modules, etc.) are configured using "Automatic commissioning" and must not, therefore, be created offline.

Procedure

1. Deselect all target devices, with the exception of the SIMOTION D4x5-2, using the "Target system" > "Select target devices" menu command in SCOUT.
2. Go online with the D4x5-2 by clicking "Connect to selected target devices".
3. Select the D4x5-2 in the project tree. Load the configuration to the SIMOTION D4x5-2 with "Download CPU / drive unit to target device".
4. Copy the configuration from RAM to ROM with "Copy RAM to ROM".
5. Select "Connect target device" in the SINAMICS Integrated context menu. You will then be connected to the SINAMICS Integrated online.
6. Perform "Automatic Configuration" on the SINAMICS Integrated. This may take several minutes, depending on the number of components connected.
   The SINAMICS Integrated is now ready for operation (RDY LED flashes green at 0.5 Hz). The CX32-2 can be accessed online, although it has not been configured yet (RDY LED flashes green at 0.5 Hz, DP LED is off).
7. Select "Connect target device" in the CX32-2 context menu. You will then be connected to the CX32-2 online.
8. Perform the automatic configuration on the CX32-2. The CX32-2 is now also ready for operation (RDY LED is green, DP LED is green (in RUN state) or flashes green at 0.5 Hz (in STOP state)).
9. Repeat steps 7 to 8 for each CX32-2.
10. Copy the parameterization from RAM to ROM with "Copy RAM to ROM".
11. Save the project.

If you run the "automatic configuration" in Step 6 and 8, a check will be made to see if the firmware on the SINAMICS components differs from the firmware on the CF card. If it does, you

will be informed accordingly and the firmware on the SINAMICS components will be upgraded/downgraded automatically.

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2 and indicate that a firmware update is being carried out on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

Note

Topology errors can result in a download or system power-up not being possible.

CX32-2 configuration information

This section contains configuration tips relating to a number of features which are specific to operating the CX32-2 together with the D4x5-2.

CX32-2 power-up

Powerup times

The CX32-2 does not require its own CF card. The firmware and the parameterization are stored centrally on the SIMOTION D CF card. After a firmware change, the CX32-2 firmware from the CF card is copied to the CX32-2 at the first power-up and saved there as non-volatile data. For this reason, the first power-up with new firmware takes longer than subsequent power-ups.

Detecting when the CX32-2 powers up in the user program

The user must wait for the CX32-2 to power up before the drive objects can be accessed by the user program.

One way of checking is to query the status of system variables of the drives on the CX32-2.

- <axis name>.actormonitoring.cyclicinterface = ACTIVE AND
- <axis name>.sensordata[1].state = VALID

This procedure can also be used to establish when the SINAMICS Integrated drives power up.

Interconnecting the infeed "Operation" signal on the CX32-2

Requirement

- Drive line-up with SIMOTION D4x5-2 control unit and Line Module with DRIVE-CLiQ interface
- One or more CX32-2 controller extensions without a Line Module of their own (i.e. the Motor Modules of the D4x5-2 and the CX32-2 modules are supplied by the same DC link)

In this example, the Line Module must to be connected to the SIMOTION D4x5-2 (not to the CX32-2).
For all other cases, see Section Using one infeed for several CUs (Page 7146).

Procedure

When drives are created in Drive system 1, the signal "Closed-loop control operation" r0863.0 from the Line Module is automatically interconnected with the "Infeed operation" signal from the drives (Drive 1...n) via p0864.

There is an internal communication channel available for the CX32-2 controller extension (drive system 2), via which the following signals from the SIMOTION D4x5-2 control unit can be automatically interconnected to the CX32-2 controller extensions:

- "Closed-loop control operation" of the Line Module connected to the SIMOTION D4x5-2
- Status of the SIMOTION D4x5-2 onboard digital inputs (X122/X132, DI 0 ... 7 and DI/DO 8 ... 15)

These signals are available on the CX32-2 controller extension; since there is no need to perform any additional configuration steps, it is very easy to make additional interconnections with them within the CX32-2.

Interconnecting the infeed operation signal

The "Closed-loop control operation" signal of the Line Module connected to the SIMOTION D4x5-2 is available in parameter r8510.0 of the CX32-2. Using the expert list or the drive wizard, interconnect parameter p0864 (drives on CX32-2) to parameter r8510.0 of the CX32-2 controller extension.

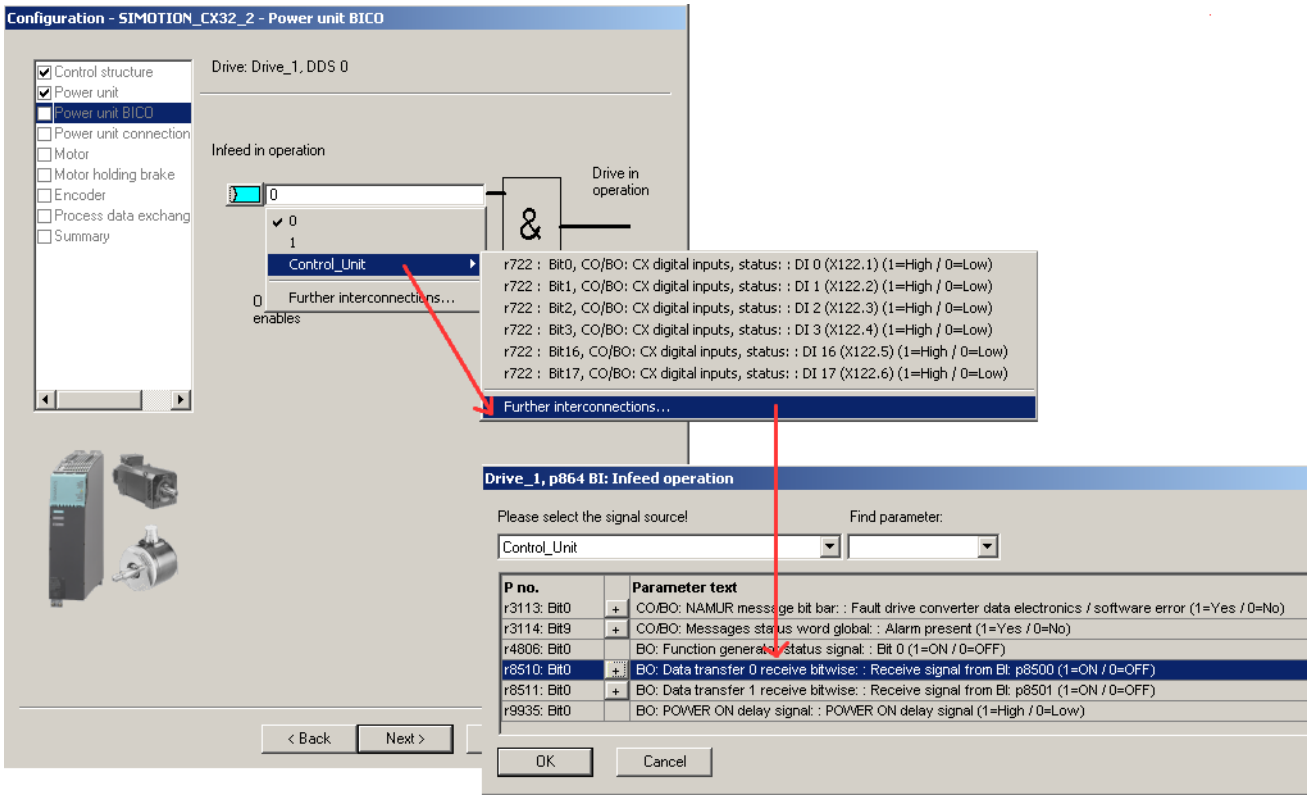


Figure 10-79 Interconnecting the infeed operation signal

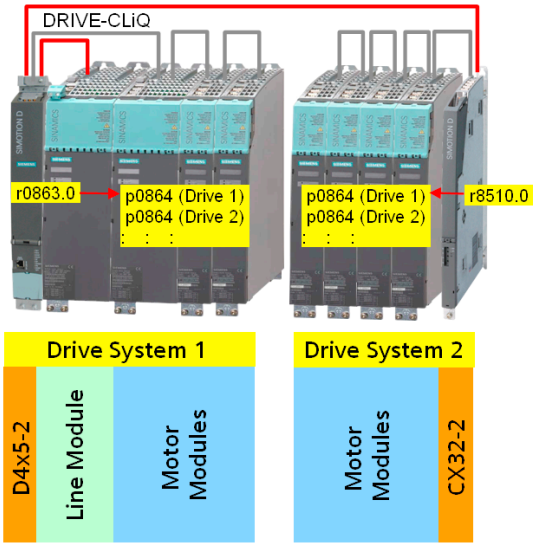


Figure 10-80 Operation signal interconnection (explanatory illustration)

Infeed to CX32-2 / CU320-2

If the infeed is connected to a CX32-2 / CU320-2 and the ready signal is to be connected to the SINAMICS Integrated, or another CX32-2 or CU320-2, proceed as described in Section Using one infeed for several CUs (Page 7146).

Terminal state of D4x5-2 onboard digital inputs

The terminal states of the SIMOTION D4x5-2 onboard digital inputs (X122/X132, DI 0 ... 7 and DI/DO 8 ... 15) are available in parameter r8511[0 ... 15] of the CX32-2. These can be interconnected further on the CX32-2.

Upgrading the CX32-2

CX32-2 modules are always operated with the same firmware version as the SINAMICS Integrated of a SIMOTION D.

This means no additional measures need to be taken for CX32-2 modules.

If the firmware version of the SIMOTION D is upgraded via HW Config, the SINAMICS Integrated and the connected CX32-2 controller extension are also automatically upgraded.

For instructions on upgrading a SIMOTION D, see also Section Adapting a project (Upgrading the project / Replacing the SIMOTION controller) (Page 7221).

FW update

The components' firmware is upgraded or downgraded automatically based on the firmware version on the CF card and the firmware version on the SINAMICS components (DRIVE-CLiQ components, CBE30-2, TB30, Power Modules, etc.).

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2 and indicate that a firmware update is in progress on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

See also

Also refer to the information in Section General information on SINAMICS firmware update (Page 7244).

Replacing the CX32-2

Replacing a module

When used as a spare part, a CX32-2 behaves like other DRIVE-CLiQ components.

A module replacement is detected automatically as of SINAMICS V4.5.
See Section

- Replacing modules in the spare part scenario (Page 6998)
- Backing up / restoring / deleting SINAMICS NVRAM data (Page 7110)

D435-2/D445-2/D455-2 ↔ D425-2 replacement

Drag-and-drop operations can be used to perform a D435-2/D445-2/D455-2 ↔ D425-2 module replacement in HW Config.

Please note that it will not be possible to replace a D435-2/D445-2/D455-2 with a D425-2 if a CX32-2 has been configured with the address 14 or 15. (A D425-2 does not have DRIVE-CLiQ ports X104/X105 and therefore does not have addresses 14 and 15.)

Additional information on configuring the SINAMICS Integrated

Settings for DP slave properties

Settings in HW Config

Depending on the cycle clock ratios (bus cycle clock, servo cycle clock) and the drives used, it may be necessary to change the properties of the DP slave (SINAMICS Integrated) on the PROFIBUS Integrated.

Open HW Config. Double-clicking the SINAMICS Integrated enables you to display and, if required, change the properties of the DP slave on the "Isochronous mode" tab. Example:

- Synchronizing a drive to the isochronous DP cycle
  SINAMICS Integrated and CX32-2 can only be operated isochronously. For this reason, this option cannot be deactivated.
- Changing the master application cycle (T_{MAPC})
  The master application cycle must always be the same as the servo cycle clock set (setting: "Shortcut menu of the D4x5-2" > "Set system cycle clocks" in the project tree). Provided that the DP cycle is not reduced to the servo cycle clock, the master application cycle will always be the same as the DP cycle.

Note

As of V4.3, a reduction ratio for the DP cycle to the Servo cycle clock is possible. The reduction ratio is only permitted when PROFINET IO with IRT has not been configured.

- Changing the DP cycle (T_{DP})
  Depending on the requirements in terms of the quantity structures and response times, the DP cycle may need to be changed (see also *SIMOTION Runtime Basic Functions* Function Manual).
  In addition, the minimum DP cycle for vector drives also depends on the speed controller cycle clock, which in turn depends on the drive quantity structure and the device type used. This means that, particularly in the case of vector drives, the DP cycle must be checked and changed if necessary (see Section Use of vector drives (Page 7104)).

  Note

  After T_{DP} has been changed on the PROFIBUS master, the drive system must be switched on (POWER ON).

- Changing the T_I and T_O times
  A change to T_I/T_O is required in the case of vector drives, for example, where the T_I/T_O time depends on the number of vector drives or, with chassis devices, on the device type used.

If you change the settings for the SINAMICS Integrated, you must also change the settings on the CX32-2 modules accordingly. This is easily done by using the Align button to transfer the settings for all the CX32-2 modules.

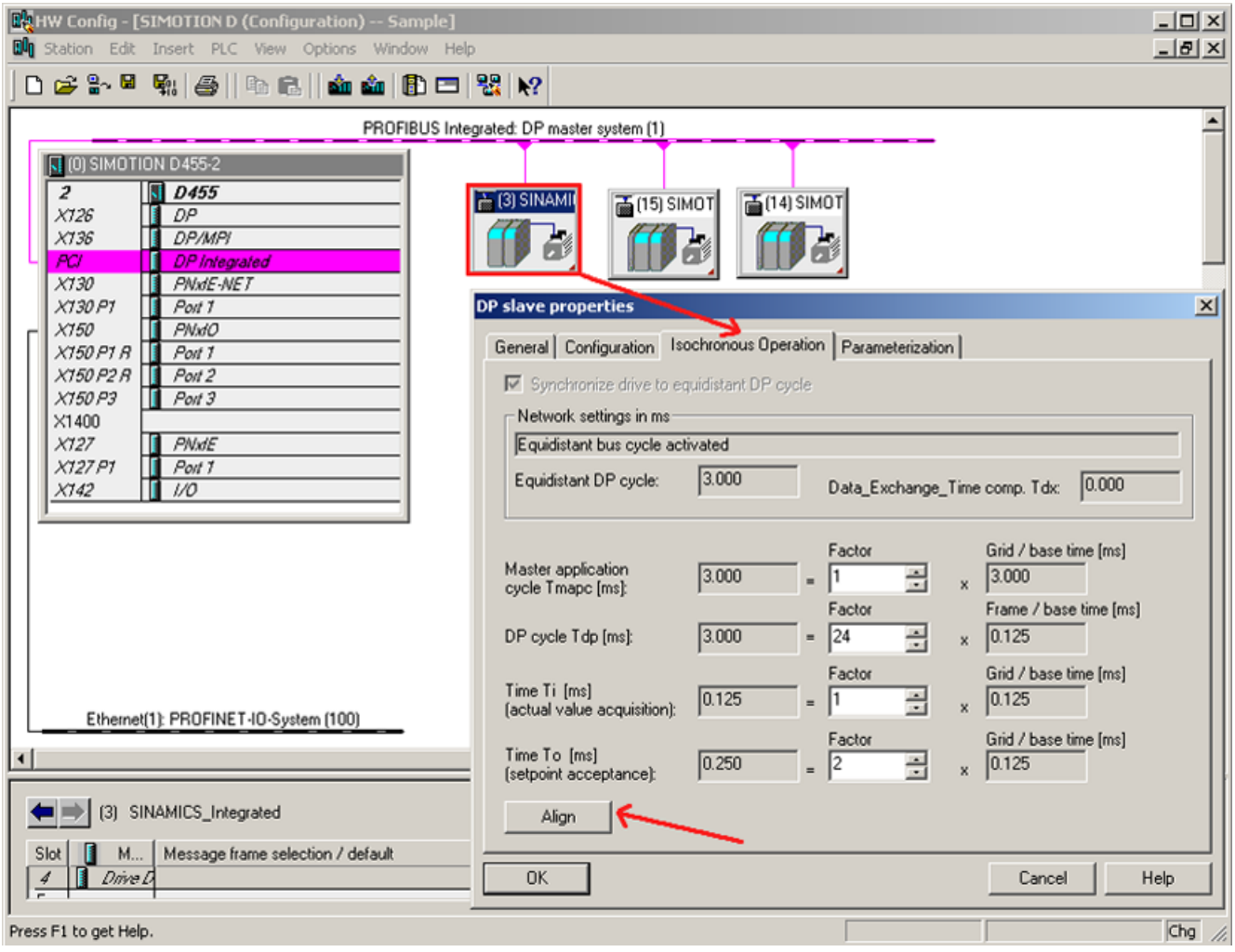


Figure 10-81 HW Config - settings

The times are modified by changing the value in the "Factor" field.

SCOUT TIA

In order to parameterize the isochronous cycle clock for the PROFIBUS Integrated in the TIA Portal, proceed as follows:

1. Select the PROFIBUS Integrated in the network view.
2. Select the "General" tab in the Inspector window and click "Equidistance" to display the parameters for the isochronous PROFIBUS. The "Activate isochronous bus cycle" setting is permanently selected and cannot be edited.

The other parameters are set to default settings and can be edited.

Additional references

Further information is available in the following references:

- *SINAMICS S120* Function Manual
- *SIMOTION Runtime Basic Functions* Function Manual
- *SIMOTION SCOUT TIA* Configuration Manual

Using vector drives

Changes need to be made in HW Config when using SINAMICS vector drives. This means, for example, that the T_I/T_O time and the minimum DP cycle depend on the number of vector drives or, with chassis drives, also on the device type used.

Therefore, we recommend adopting the following procedure when using vector drives with the SIMOTION D4x5-2.

Scenario 1: Current and speed controller cycle is known

As long as you know the current and speed controller cycle, you will be able to determine times T_{MAPC}, T_{DP}, T_I, and T_O. Where there are several drives with different cycle clocks, the largest cycle clock must be used for the current and speed controller cycle.

Procedure

1. Open HW Config. By "double clicking" the SINAMICS Integrated, you can change the properties of the DP slave in the "Isochronous operation" tab.
2. For $T_I = T_O =$, enter an integer multiple of the current controller cycle.
3. For T_{DP}, enter an integer multiple of the speed controller cycle. For drives on the SINAMICS Integrated, T_{DP} must always be $\geq T_O$.
4. Enter $T_{MAPC} = T_{DP}$ (exception: you are working with cycle clock scaling; i.e. position control cycle clock > DP cycle).
5. Use the "Download project to target system" menu command to load the configuration to the SIMOTION D4x5-2.
6. After the download has successfully completed, you should determine the current and speed controller cycles of the drives from the drives' expert lists, as the cycle clocks are set in the SINAMICS drive unit after a project has been downloaded.
   p0115[0] current controller cycle clock
   p0115[1] speed controller cycle clock
7. If the current and speed controller cycles from the expert lists are different to the cycle clocks used in steps 2 and 3, you will have to repeat the steps with the current values for the current and speed controller cycles.

Note

The described procedure is also required for an online configuration with an automatic configuration.

Table 10-43 Example for power units in booksize format (vector drives)

Example	Settings
1 to 3 vector drives Current controller cycle = 250 µs Speed controller cycle = 1 ms	$T_I = T_O$ = at least 250 µs T_{DP} = 1 ms (... or 2 ms, 3 ms, ...) $T_{MAPC} = T_{DP}$
4 to 6 vector drives Current controller cycle = 500 µs Speed controller cycle = 2 ms	$T_I = T_O$ = at least 500 µs T_{DP} = 2 ms (... or 4 ms, 8 ms, ...) $T_{MAPC} = T_{DP}$

Note

Vector drives in chassis format can also be operated with a current controller sampling time of 400 µs (amongst other settings).

In the SIMOTION context, the following should be considered:

- A current controller sampling time of 400 µs is only possible if control is via a SINAMICS S120 control unit, which is not operated isochronously via PROFIBUS/PROFINET on SIMOTION D.
- If the bus is operated isochronously, only cycle clocks with an integer multiple of 125 µs are possible (i.e. 375 µs or 500 µs instead of 400 µs, for example).
- The PROFIBUS Integrated of a D4x5-2/CX32-2 is always isochronous! This means that a current controller sampling time of 400 µs is not possible.
- With CU parameter p0092 = 1, the sampling times are pre-assigned so that isochronous operation with a controller is possible.

Scenario 2: Current and speed controller cycle is not known

This procedure is especially suitable for devices in chassis format, as the current and speed controller cycle also depends on the relevant device type.

1. To determine which cycle clocks are set in the SINAMICS drive unit after a project has been downloaded, first set reliable values for the cycle clocks in HW Config (see table "Recommended cycle clock settings in HW Config").
2. Use the "Download project to target system" menu command to download the parameterization.
3. After the download has successfully completed, you will be able to determine all the current and speed controller clock cycles of the drives from the drives' expert lists.
   p0115[0] current controller cycle clock
   p0115[1] speed controller cycle clock
4. If required, it is now possible to optimize the T_{MAPC}, T_{DP}, T_I, and T_O times in HW Config (see the procedure in Scenario 1).

Table 10-44 Recommended cycle clock settings in HW Config

Setting	Explanation
T_{DP} = 3.0 ms	T_{DP} = DP cycle time
$T_I = T_O$ = 1.5 ms	T_I = time of actual value acquisition T_O = time of setpoint transfer
T_{MAPC} = 3.0 ms	T_{MAPC} = master application cycle time

Setting the sampling times

Automatic setting

On SIMOTION D4x5-2, if p0112 = 3 (default) is set, the sampling times in the p0115 (current controller, speed controller, ...) are automatically set after a project download and may therefore differ from the offline values. The sample times depend, e.g. on the number of vector drives and the Power Module used (e.g. chassis). If the sampling times have been adapted by the system, p0112 = 0 (expert) is set automatically.

Manual setting

With p0112 = 1...5, the sampling times (current controller, speed controller, ...) can be selected.

If sampling times are required which cannot be set using p0112 > 0, then you can directly set the sampling times using p0115. To do so, p0112 must be set to "0" (Expert).

p0112 = 0 enables the individual sampling times to be adjusted in p0115. Moreover, with this setting, the system does not change the current controller sampling time automatically.

Output cams / measuring inputs with vector drives

With vector drives, the cycle clock ratios (current controller cycle clock, speed controller cycle clock, sampling time of the I/Os, etc.) depend on the number of vector drives, or with chassis units also on the device type used.

Note the information in Section Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs (Page 7116).

Additional references

Additional information on quantity structures and cycle clock settings can be found in the *SINAMICS S120* Function Manual.

Setting the time of day

Time on SIMOTION (real-time clock)

SIMOTION D4x5-2 has an integrated real-time clock. All events on a module (alarms, messages, etc.) are "time-stamped" based on the time shown by this real-time clock.

Various options are available to set the clock of the SIMOTION D4x5-2:

- Setting the clock via the engineering system
  - SIMOTION SCOUT: Select the SIMOTION D4x5-2 in the project tree and then "Target system" > "Set time" in the menu.
  - SIMOTION SCOUT TIA: Call the "Set time" function at "Online & diagnostics" in the TIA Portal in the project tree.
- Setting the clock using the "rtc" system function block
- Synchronization of the local time with an NTP time server (as of V4.5)

The status of the NTP time synchronization can be determined with the **_getStateOfNTPClockSynchronisation** system function.

For further information on the NTP process, see the *SIMOTION Communication* System Manual and the *SIMOTION SCOUT TIA* Configuration Manual.

SINAMICS system runtime (operating hours counter)

Faults and alarms are "time-stamped" based on the system runtime on SINAMICS Integrated of a SIMOTION D, Controller Extensions and SINAMICS S120 Control Units. This means that events are recorded by default on the basis of operating hours rather than a particular time of day or date.

System runtime

The entire system runtime is displayed in CU parameter p2114.

- p2114[0] indicates the system runtime in milliseconds. After reaching 86,400,000 ms (24 hours), the value is reset.
- p2114[1] indicates the system runtime in days.

The counter value is saved when the power is switched off. After the drive unit has been switched on, the counter continues to run with the value stored when the power was last switched off.

As a result, the drive displays the system runtime from 00:00:00 on 01/01/1992 in both the alarm window in SIMOTION SCOUT and the diagnostic buffer for entries.

If faults and alarms need to be "time-stamped" based on a time of day, "Time stamp operating hours" needs to be changed to "Time stamp UTC format" as described below.

Requirements

A telegram 39x is required for the time synchronization. If the automatic PROFIdrive telegram setting is selected for the Control Unit, this telegram is used automatically (see Section Calling the drive wizard (Page 7053), Standard/Automatic setting).

If the telegrams are specified manually, telegram 39x must be set up (see Section Telegram configuration (Page 7148)).

So that drive units can be synchronized with the SIMOTION time, they must support telegram 39x and the UTC time format.

For precise time synchronization, the drive unit must also be operated via an isochronous bus on SIMOTION. The SINAMICS Integrated for SIMOTION D and the CX32/CX32-2 Controller Extension always have an isochronous connection.

The following Control Units support time synchronization:

- SINAMICS Integrated of the SIMOTION D
- CX32/CX32-2 Controller Extensions
- SINAMICS S120 CU310, CU310-2, CU320, CU320-2 Control Units connected via PROFIBUS or PROFINET
- SINAMICS S110 Control Units CU305, connected via PROFIBUS or PROFINET (precondition: as of SCOUT V4.4)

Synchronizing the SINAMICS clock

Proceed as follows to convert the SINAMICS clock to UTC format and to synchronize this with the SIMOTION clock:

1. Call the D4x5-2 context menu in the project navigator.
2. Select the "Properties..." entry in the context menu.
3. Select the "Perform time synchronization with SINAMICS drive units" option in the "Settings" tab of the "Properties - D4x5-2" dialog box.

Note

This setting is automatically activated for new projects as of V4.2 and applies for all drive units connected to the D4x5-2. The SINAMICS clock is automatically synchronized with the SIMOTION clock for all drive units with configured telegram 39x.

The first time synchronization is performed after the SIMOTION D Control Unit has reached the RUN operating state.

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically resynchronized at regular intervals.

Via the system variable **_driveStates.allClocksSynchronized** on the D4x5-2 device, the user program can query whether the automatic time synchronization is activated (=YES) or deactivated (=NO).

Before the first synchronization, alarms and messages are stored with the time stamp valid in the SINAMICS at this time, all subsequent alarms and messages with the synchronized time.

The first time synchronization after switching on is entered with the status of the operating hours counter and the time (UTC time, synchronized with SIMOTION) in the diagnostic buffer of the drive (e.g. SINAMICS Integrated).

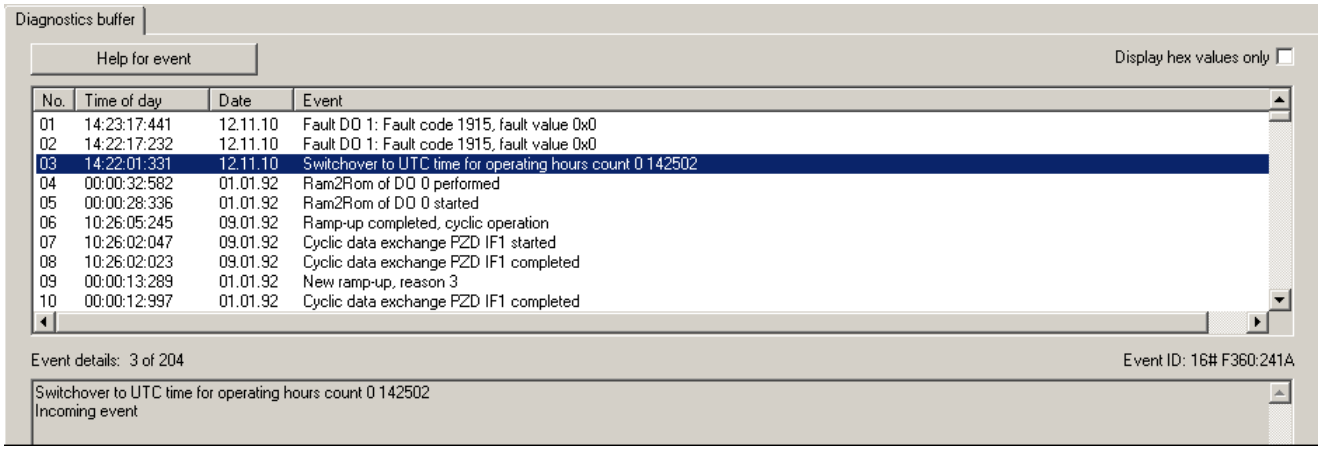


Figure 10-82 SINAMICS time synchronization diagnostic buffer entry

Error correction

If problems occur with time synchronization, this may be due to deactivated symbolic assignment to incorrect configuration information (Fast IO configuration).

For more information, see Section Message frame configuration (Page 7148).

Compensation of runtime deviations

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically resynchronized at regular intervals.

The following behavior must be taken into consideration when setting the SIMOTION time:

- "Time/date to be set" is later than "Time/date on SINAMICS": Time and date are corrected on the SINAMICS.
- "Time/date to be set" is earlier than "Time/date on SINAMICS": The SINAMICS clock must be stopped until the SINAMICS "Time/date" has caught up with "Time/date to be set".

Adopting this procedure ensures that the sequence of SINAMICS diagnostic buffer entries remains the same, even when the runtime differences are aligned.

The SINAMICS clock operates with a resolution of 1 ms. A synchronization accuracy of 1 ms can be achieved for all bus cycle clocks that can be divided exactly by 1 ms (e.g. 1 ms, 2 ms, 3 ms, etc.).
Due to system considerations, a slightly lower synchronization accuracy is achieved for all bus cycle clocks that cannot be divided exactly by 1 ms (e.g. 1.25 ms).

If the drive unit does not have an isochronous connection on SIMOTION, this can result in runtime differences of milliseconds (depending on the set cycle clock ratios).

Resetting the time

Requirement:

- SIMOTION D4x5-2: As of SIMOTION V4.3
- SINAMICS S120: As of SINAMICS V4.5
- SINAMICS S110: Not available

A threshold value is defined via parameter cu.p3109 and is effective as follows:

- In the case of negative time jumps less than the threshold value (p3109), the time is halted (for details, see "Compensation of runtime deviations").
- In the case of negative time jumps greater than the threshold value (p3109), the time is reset.
  Default setting: cu.p3109 = 100 ms
  This means that in the case of negative time jumps greater than 100 ms, the time is reset. The default value is configured so that normal runtime deviations (drift of the quartzes) are below the threshold value.
  If the SIMOTION clock is reset by more than 100 ms, this is interpreted as a "specific reset of the time" and the time of the drives is also immediately reset.

If the real-time clock is reset by more than 60 seconds, a diagnostic buffer entry is also made in the drive:

Time correction (reset) by <correction value> seconds

After a resynchronization (negative time jump greater than the threshold value)

cu p3107[0..1] the UTC time after synchronization

cu p3107[2..3] the UTC time before synchronization

is displayed in the parameter, whereby [0] and [2] are milliseconds and [1] and [3] are days.

Note

The diagnostic buffer entries are not converted to the new time when the time is changed.

Backing up / restoring / deleting SINAMICS NVRAM data

Requirement

- SIMOTION D: SIMOTION D4xx-2/CX32-2 as of SIMOTION V4.3 / SINAMICS V4.5
- SINAMICS S120 CU310-2/CU320-2: SINAMICS V4.5
- Other supported SINAMICS CUs: See the SINAMICS manuals

Saving the NVRAM data

The backup of the SINAMICS NVRAM data is performed by setting parameter p7775 to 1.

The following applies:

- Parameter p7775 can also be set to 1 during pulse enable.
- A warm restart is not required.

Note

The consistent backup of the NVRAM data must be ensured by the application.

Make sure that there is no change of the NVRAM contents during the backup.

The greatest possible consistency is achieved when

- Backup is performed during pulse disable or
- The drives are run down to 0 Hz.

The data is stored on the CF card:

- "....\USER\SINAMICS\NVRAM" for S120 CUs and SINAMICS Integrated
- "....\USER\SINAMICS\\NVRAM\xx" for CX32-2 (xx corresponds to the DRIVE-CLiQ port)

The data itself is stored in the PMEMORY.ACX backup file.

For the S120 Control Units, the data is stored on the CF card of the S120 Control Unit.

Similar to the non-volatile SIMOTION data, an already existing PMEMORY.ACX file is first renamed as PMEMORY.BAK and then the PMEMORY.ACX file is generated when you save the SINAMICS data.

If an error occurs during backup, the BAK file can be used to restore the data.

Parameter p7775 is set to 0 at the end of the backup.

Restoring the NVRAM data

The SINAMICS NVRAM data can only be restored when none of the connected drives has a pulse enable.

When restoring, the PMEMORY.ACX file is accessed first. If this is available and is error-free, it is loaded. If no PMEMORY.ACX file is available or is corrupt, the PMEMORY.BAK file is loaded (if available and error-free).

Restoration of the NVRAM data can be performed manually and automatically:

Automatic restoration during module replacement

- SINAMICS detects whether the control unit has been replaced on the basis of the serial number.
- In this case, the NVRAM of the used control unit is deleted first after POWER ON.
  The following are not deleted:
  - CU operating hours counter
  - CU temperature and
  - Safety logbook
  - Crash diagnostic data

If there is an error-free PMEMORY.ACX or PMEMORY.BAK file on the CF card, the data is then loaded to the NVRAM.

If no error-free backup file is available, ramp-up is as for SINAMICS < V4.5. A fault is not issued and any existing "corrupt" backup files are not deleted.

Manual restoration

The manual restoration of the NVRAM data is performed by setting parameter p7775 to 2.

If the backup file (ACX or BAK) is error-free is, a warm restart is performed. The contents of the NVRAM are deleted first and then the data is loaded from the backup file to the NVRAM.

The following are not loaded:

- CU operating hours counter
- CU temperature and
- Safety logbook
- Crash diagnostic data

Parameter p7775 is set to 0 at the end of the restoration.

If the backup files are corrupt or no backup exists, the job is acknowledged with an error value in parameter p7775.

Deleting the NVRAM data

The SINAMICS NVRAM data can only be deleted when none of the connected drives has a pulse enable.

The deletion of the NVRAM data is performed by setting parameter p7775 to 3.

After setting p7775 to 3, a warm restart is performed automatically.

During the following ramp-up, first all NVRAM data is deleted on the control unit.

The following are not deleted:

- CU operating hours counter
- CU temperature and

- Safety logbook
- Crash diagnostic data

At the end of the deletion, the initialization data of the applications is in the NVRAM, similar to as after a device automatic commissioning (with the exception of the above-mentioned data).

Parameter p7775 is set to 0 at the end of the deletion.

Explanations for p7775

The following jobs can be issued via p7775:

1: Backup NVRAM data (on CF card)

2: Restore NVRAM data (from CF card)

3: Delete NVRAM data on CU

The acknowledgement of a job is delayed.

- If the job cannot be executed, it is acknowledged negatively.
- If the job can be executed, it is acknowledged with the value 255.

Acknowledgements of the parameter jobs

Table 10-45 Acknowledgements of the parameter jobs

Value	Cause
17	Job cannot be executed because of the operating state
20	Illegal value
107	Write access is not permitted (cause: At least one DO has a pulse enable). As the SINAMICS has to perform a warm restart when restoring and deleting the data, none of the connected drives must have a pulse enable
132	Parameter change blocked (see p0300, p0400, p0922, p7760, macro being executed)
204	Write access not permitted
255	"OK": Job being executed

If an error is detected during the execution of a parameter job, the error cause is signaled via the parameter itself (no message via the fault buffer).

Causes of errors for parameter jobs

Table 10-46 Causes of errors of the parameter jobs

Value	Cause
10	Error occurred while deleting
11	No backup possible: Memory card is not inserted
12	No backup possible: Memory card is full
13	Backup could not be completed (e.g. memory card removed during the backup)
14	Restoration not possible: Memory card is not inserted

Value	Cause
15	Restoration not possible: Checksum of the NVRAM data backup file is faulty
16	Restoration not possible: No backup available

Note

The memory card must be always remain inserted for SIMOTION D; please refer to the SINAMICS manuals for which SINAMICS Control Units allow the memory card to be removed.

Parameter p7775 is only reset automatically to 0 after the successful completion of a job.

Know-how and write protection

- Parameter p7775 is not part of the know-how protection.
  I.e. the parameter can be read and written irrespective of the know-how.
- Parameter p7775 is part of the write protection. This means the parameter can only be written by the controller that configured a PZD cyclic communication with the SINAMICS when write protection is activated (p7761 = 1).
  All other write jobs are acknowledged negatively, e.g.
  - SCOUT/STARTER
  - For SINAMICS CUs, BOP, IOP, AOP
  - Other masters that only communicate acyclically with the SINAMICS

SINAMICS diagnostic buffer

The diagnostic buffer of the SINAMICS Integrated, the CX32-2 and the SINAMICS S120 control unit can be displayed in SIMOTION SCOUT.

To do this, select the SINAMICS Integrated, the CX32-2 or the SINAMICS S120 control unit in the project tree. Then select "Target system" > "Device diagnostics" in the menu.

In addition, the SINAMICS diagnostic buffer entries are also displayed in the D4x5-2 device diagnostics. All D4x5-2 diagnostic buffer entries are displayed first, followed by those for the SINAMICS Integrated and the CX32-2. The start of the SINAMICS diagnostic buffer entries is identified by the following entry:

>>>>>> Start of SINAMICS Integrated diagnostic buffer, station address = x <<<<<<

You also have the option of viewing the SIMOTION D4x5-2 and SINAMICS Integrated diagnostic buffers via SIMOTION IT web server.

Acyclic communication with the drive

Overview

PROFIdrive drive units are supplied with control signals and setpoints by the controller and return status signals and actual values. These signals are normally transferred cyclically (i.e. continuously) between the controller and the drive.

For the SINAMICS S110/S120, configure the axis telegrams for data exchange (see Section Performing the configuration for the D4x5-2 offline (Page 7052)).

As well as offering cyclic data exchange, PROFIdrive drive units have an acyclic communication channel. In particular, this is used for reading and writing drive parameters (e.g. error codes, alarms, controller parameters, motor data, etc.).

As a result, data can be transferred on an "acyclic" as opposed to "cyclic" basis when required. Acyclic reading and writing of parameters for PROFIdrive drives is based on the DP V1 services "Read data set" and "Write data set".

The acyclic DP-V1 services are transferred in parallel to the cyclic communication via PROFIBUS or PROFINET. The PROFIdrive profile specifies precisely how these basic mechanisms are used for read/write access to parameters of a PROFIdrive-compliant drive.

The PROFIdrive standard states that "pipelining" of jobs on PROFIdrive drives is not supported. This means:

- Only one "Write/read data set" can be performed at any one time on a drive unit (e.g. SINAMICS S110/S120 control unit or the SINAMICS Integrated of a SIMOTION D).
- However, if several PROFIdrive drive units are connected to a controller, a job can be processed for each of these drive units at the same time. In this case, the maximum total number of jobs will depend on the controller (maximum of eight jobs simultaneously with SIMOTION).

For acyclic data exchange with SINAMICS drives, this means you will have to coordinate the write/read jobs with each other (buffer management). An interlock must be set to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive unit.

Additional references

Additional information on how to use DP V1 services can be found in the *SIMOTION Communication* System Manual.

SIMOTION Utilities & Applications also has a DP V1 library with functions that are capable of performing coordination tasks commonly associated with acyclic communication. The library not only coordinates access to the system functions (_readRecord/_writeRecord/ _readDriveParameter/_writeDriveParameter/, etc.), but also expands the range of functions for frequently required tasks, e.g. the reading of faults and alarms from the drive unit.

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

The following functions are available in the DP V1 library:

- Buffer management (coordination of a number of parallel DP V1 services)
- StartUp (function for coordinating the power-up of the SINAMICS drive with SIMOTION)
- TimeSync (applicative time synchronization: Transfer of SIMOTION time of day to the SINAMICS drives)
- SetActIn (activating/deactivating objects in SIMOTION and SINAMICS)
- RwnPar (reading and writing of drive parameters)
- GetFault (reading faults and alarms from the drive)

Control properties and performance features

With a few exceptions, the integrated drive control of SIMOTION D4x5-2 and CX32-2 has the same control properties and performance features as the SINAMICS S120 CU320 control unit.

However, the following points must be particularly observed:

- SINAMICS Integrated and CX32-2 have no basic positioner (EPOS). EPOS functionality is provided by SIMOTION technology functions.
- A BOP20 basic operator panel cannot be connected to the SIMOTION D4x5-2.
  The following alternative options are available:
  - Use of SIMATIC HMI devices (e.g. TP177B, configurable with WinCC flexible)
  - Use of the SIMOTION IT web server: You can use a Web browser to access the standard diagnostics pages of the SIMOTION D4x5-2 (diagnostics and alarm buffers, Watch table, read/write SIMOTION variables and drive parameters, access protection, trace function, etc.)
    You also have the option of creating your own Web pages, in order for example to visualize machine states and implement service functions. The SIMOTION D4x5-2 web pages can be accessed, for example, with a PC or PDA via Ethernet. Wireless access is also possible in conjunction with WLAN.
- Not all SINAMICS function modules are supported (e.g. the function modules free function blocks, spindle diagnostics, and CAN are not supported).
- The SINAMICS Integrated of the SIMOTION D4x5-2 does not support a reset via parameter p0972. A reset via p0972, is only supported by CX32-2 and SINAMICS Control Units. You can use this method, for example, in the context of the implementation of modular machine concepts to force a restart of the CX32-2 controller extension (only required for versions < V4.3). See also the *Motion Control basic functions for modular machines* Function Manual.

Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs

If current controller cycle clocks <> 125 µs are used, the parameter calculations of the drive must be transferred to the PG when using cam outputs on the TM15 / TM17 High Feature or for global measuring inputs, and FastIO configuration generated again.

A change of the current controller cycle clock may have effects on the sampling times of the inputs/outputs on the drive side (e.g. TM15/TM17 High Feature, p4099 Sampling time of the inputs/outputs).

Sampling times <> 125 µs occur in the following cases:

- For servo drives with manual change of the current controller sampling time (drive parameters p0112 and p0115[0])
- For vector drives depending on the number of vector drives and with chassis units depending on the device type used
- If only an infeed and **no drives** are connected to the drive unit, then the sampling time is 250 µs

For the cam outputs and the measuring input inputs (only for global measuring inputs) to function correctly, the sampling times must be known to the engineering system.

Table 10-47 Influence of the current controller cycle clock on the dead time compensation

	Current controller cycle clock has no effect on the function	**Current controller cycle clock has an effect on the function**
Cam outputs	• SIMOTION D • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	TM15 / TM17 High Feature
Measuring input inputs (global measuring inputs)	• D4x5-2 (terminal X142) • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	• TM15 / TM17 High Feature • SIMOTION D (except D4x5-2, terminal X142) • Controller Extension (CX) • SINAMICS S110/S120 Control Units
Measuring input inputs (local measuring inputs)	---	---

In order that the changed cycle clock ratios are taken into account by the engineering system, proceed as follows:

1. Go online and perform a project download. The SINAMICS performs parameter calculations once.
2. Perform an upload to the PG ("Target system" > Load" > "Load CPU / drive unit to PG").
3. This transfers the parameter calculations of the drive to the PG. The cycle clock ratios are then known in the engineering system.
4. Go offline.
5. Generate the configuration information (Fast IO configuration) again. To do this, select the SIMOTION CPU in the project tree and right-click to open the context menu "Fast IO" > "Create new configuration".
6. Execute "Save project and recompile all".
7. Go online and download the project to the target system.
8. Save the data on the CF card.

SCOUT uses the described procedure to calculate internal system data that is required for outputting/detecting signals with a high level of position accuracy.

Note

If the cycle clock ratios are not set correctly, an appropriate message is output in the diagnostic buffer.

CU-Link (data transfer via DRIVE-CLiQ)

Terms

The following table provides an overview of the terms used in this section.

Table 10-48 Terms used

Designation	Explanation
CU-Link communication	Communication between Controller Extension CX32-2 and SINAMICS Integrated of a SIMOTION D4x5-2 based on DRIVE-CLiQ communication
CU_I (also DO1)	CU drive object of the SINAMICS Integrated
CU_CX32 (also DO1)	CU drive object of the Controller Extension CX32-2
CU_LINK	Each Controller Extension is represented on the SINAMICS Integrated of a D4x5-2 by a drive object CU_LINK
CU_NX_CX	CU drive object for CX/NX Term usually used in SINAMICS manuals for drive extensions for SIMOTION (CX32-2) and SINUMERIK (NX1x.3)
DO	Drive object (DO)

CU-Link

For communication between Controller Extension CX32-2 and Control Unit SIMOTION D4x5-2, not only PROFIBUS-Integrated but also a further fast communication channel is available for communication with the drives. This communication channel is named CU-Link and can only be used via BiCo interconnections.

Communication via the CU-Link is performed in the very fast DRIVE-CLiQ basic cycle clock 0 (r0110[0]) with 125 µs. The CU-Link is therefore especially suitable for transmission of the signal "Infeed ready" between the SINAMICS Integrated of the SIMOTION D4x5-2 and the connected Controller Extensions CX32-2. CU-Link cannot be used for communication with a CU320-2.

For every CX32-2 that is connected via the DRIVE-CLiQ Port X100 to X105 on a SIMOTION D4x5-2, a separate CU-Link communication channel exists.

The CU-Link communication channel consists of

- two data transfer channels for **bit-by-bit transmission and receipt** of information (p8500[0..7], p8501[0..21], and r8510.0-7, r8511.0-21)
- four data transfer channels for **word-by-word transmission and receipt** of information (p8502 to p8505, and r8512 to r8515). These data transfer channels also ahve a scaling factor that can be set in parameter p8520[0..3].
  Observe the notes below in this section under "Word-by-word data transfer via p850x/r851x (CU-Link)".

At the CX32-2 end, it is possible to access the CU-Link via the parameters on the DO CU_CX32. At the SINAMICS Integrated end, the parameters are accessed via the DO CU_LINK.

Note

There are only CU-Link communication channels between the SINAMICS Integrated and the CX32-2 Controller Extensions. Such a communication channel does not exist between the individual Controller Extensions CX32-2. If data are to be exchanged between Controller Extensions CX32-2, this is only indirectly possible via the SINAMICS Integrated.

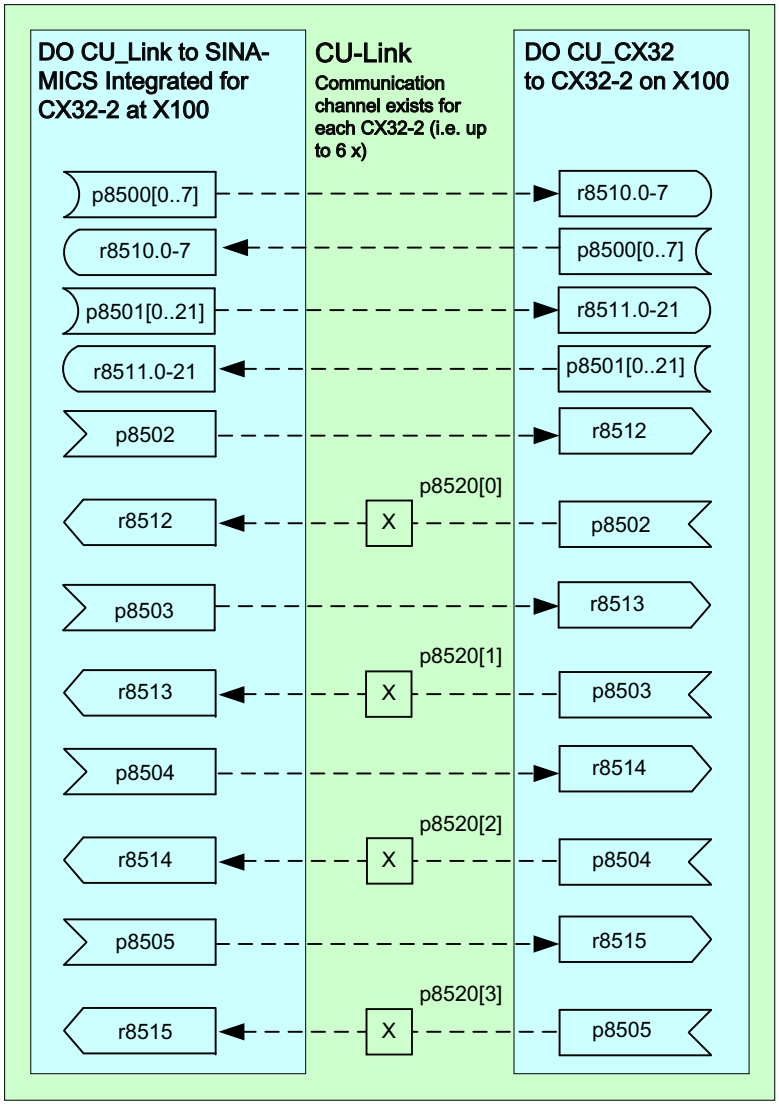


Figure 10-83 CU-Link communication channel

Interconnections on the CX32-2

At the CX32-2 end, the CU-Link communication channel can only be interconnected via the expert list of the CU.

The following interconnections are already set by default

- p8501[0..3] with cu.r0722.0-3 (status DI 0 to 3 on CX32-2)
- p8501[8..11] with cu.r0722.8-11 (status DI 8 to 11 on CX32-2)
- p8501[16..17] with cu.r0722.16-17 (status DI 16 to 17 on CX32-2)

The interconnections are visible via the expert list and also in the parameter screens for the CX32-2 Onboard I/Os (see the following figure).

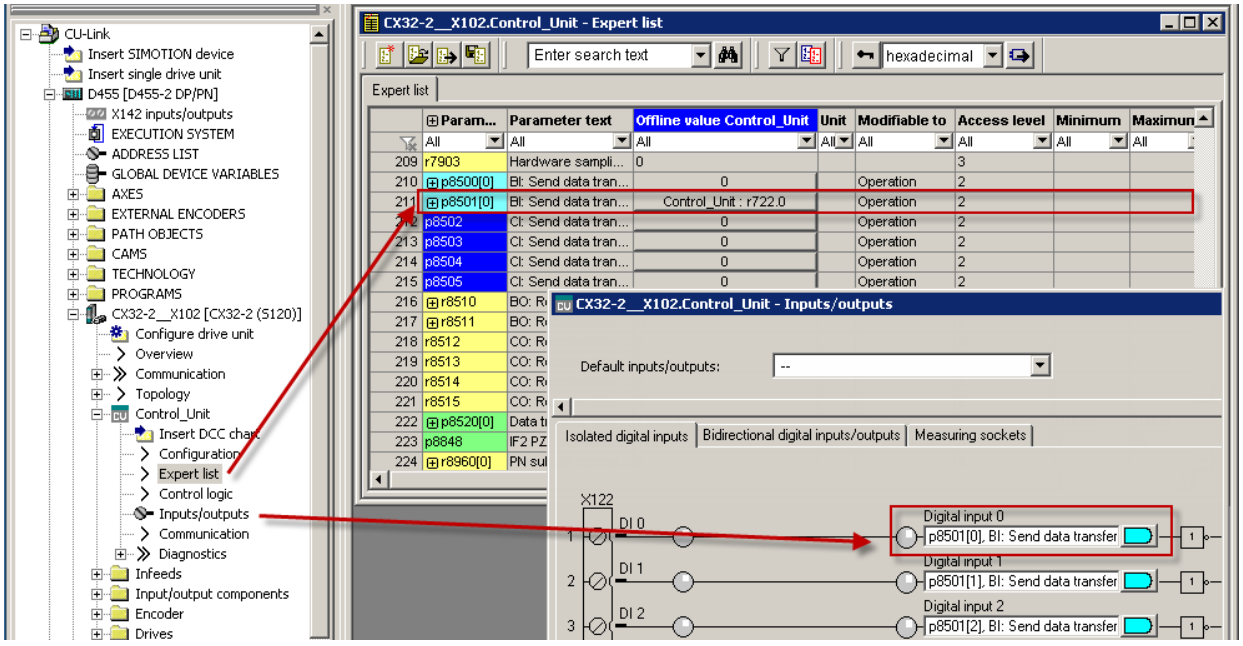


Figure 10-84 Interconnection of the DO CU_CX32 at the CX32-2 end

Interconnections on the SINAMICS Integrated

For each configured Controller Extension CX32-2, there is a separate CU-Link communication channel, including DO CU_LINK.

It must be considered that the CU_LINK DOs can **not directly** be configured, i.e. it is not possible to access the DO CU_LINK via an expert list to interconnect, for example, parameters.

Access to the DO CU_LINK is **only indirectly** possible via other DOs (e.g. via the drive DO or the DO or the infeed). Direct interconnection between two CU_Link DOs is therefore not possible (red arrow in the following figure).

To interconnect two CU_Link DOs, the CU drive object of the SINAMICS Integrated (DO CU_I) has parameters p8500-p8505 and r8510-r8515. These are visible in the expert list of the Control Unit of the SINAMICS Integrated and can be used for interconnection (blue arrow in the following figure).

Remember that the 6 CU_Link DOs have to share this one parameter set in the DO CU_I.

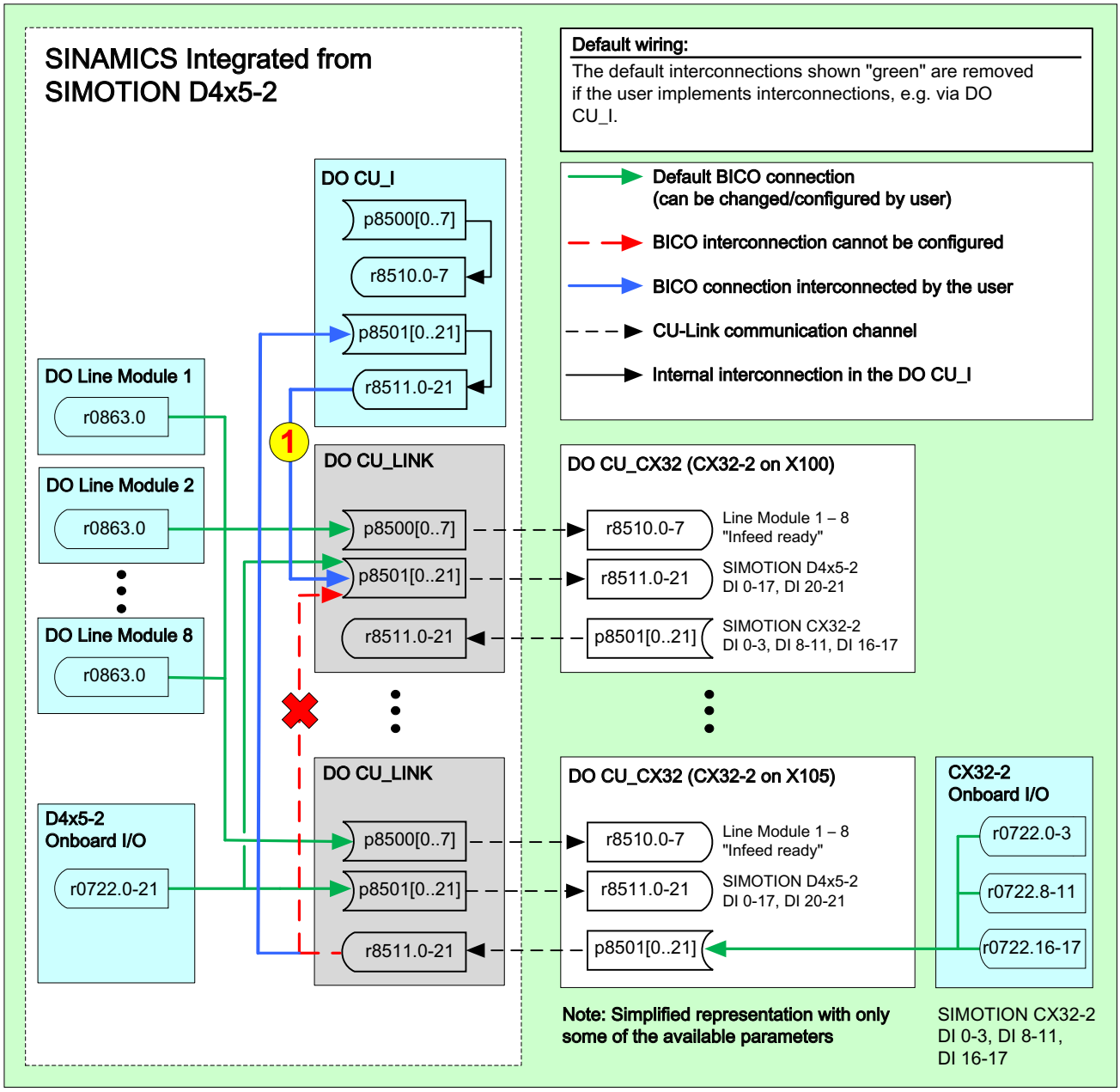


Figure 10-85 Interconnection on the SINAMICS Integrated

Transfer parameters p850x/r851x (DO CU_I)

Parameters p8500-p8505 and r8510-r8515 of the Control Unit (DO CU_I) are so-called transfer parameters. This means that a signal, for example, at the binector input p8500[0] is made available at the binector output p8510[0] and can be interconnected further from there.

With this property, the transfer parameters are ideal for the following applications:

- With the transfer parameters, CU_LINK DOs can be interconnected. This is only possible using transfer parameters because it is only possible to access a DO CU_LINK indirectly.
- With the transfer parameters, cross-DO interconnections can be created that are independent of the source DO. For example, if multiple signal sinks are interconnected to one signal source and the signal source is to be changed, it becomes necessary to change the interconnection of all signal sources. On the other hand, if the signal sources are interconnected to a transfer parameter, it is only necessary to change the signal sources entered in the signal sink of the transfer parameters.

Word-by-word data transfer via p850x/r851x (CU-Link)

The CU-Link data transfer channels for word-by-word transmission and receipt of information (p8502 to p8505 and r8512 to r8515) do not have normalization. Therefore they should only be used for the transmission of drive parameters that are normalized to SI units (V, A, m, s, etc.) or their derivatives (N, °C, Hz, W, V, etc.) or that are normalized to PERCENT. For information on normalization, see the Parameter List in the SINAMICS S120/S150 List Manual under "Normalization".

Example:
Voltage (V, normalized acc. to p2001); current (A, normalized acc. to p2002); torque (Nm, normalized acc. to p2003); temperature (°C, normalized acc. to p2006);
the speed (normalized acc. to p2000) is normalized to rpm and therefore not normalized to an SI unit. The same applies to power in kW.

For the data transfer via CU-Link, the data source and data sink should have the same normalization.

If this is not adhered to, the transferred values may be assigned a factor.

CU-Link data consistency

Data transfer via p8500[0..7] and p8502 to p8505 is consistently performed in the set DRIVE-CLiQ basic cycle clock 0 (r0110[0]) at 125 µs.

With p8501[0..21], on the other hand, transmission is performed bit-sequentially, that is, in each cycle, only one of the 22 bits is transmitted so that all data in parameter r8511 have only been updated after 2.75 ms (22 bits x 0.125 µs). The individual bits are then not consistent among themselves.

This behavior only applies to bidirectional data transfer via the CU-Link (DO CU_Link, DO CU_CX32). The transfer parameters p850x/r851x in the DO CU_I are always consistent.

Preset interconnections

Some interconnections are already preset in the system (green arrows in the figure above).

The following interconnections are available by default:

- On the SINAMICS Integrated, the "infeed ready" signal of all connected infeeds is provided on the CU-Link. This means the "infeed ready" signal is on all Controller Extensions CX32-2
  - can be picked off via r8510.0 for the first infeed
  - via r8510.1 for the second infeed etc.

  The default interconnections are only created if an infeed has be configured with a DRIVE-CLiQ connection (CX32-2 must **first** have been configured).
- The status of the onboard I/Os of the D4x5-2 is forwarded to all CX32-2, where it can be picked off via r8511.0-21.
- The status of the onboard I/Os of the CX32-2 is forwarded to the relevant DO CU_LINK.

Default interconnections can be deleted by appropriate user configurations (e.g. by configuration of the interconnection, see the previous item ①). If an existing interconnection is disconnected, an information dialog box will be displayed to indicate this.

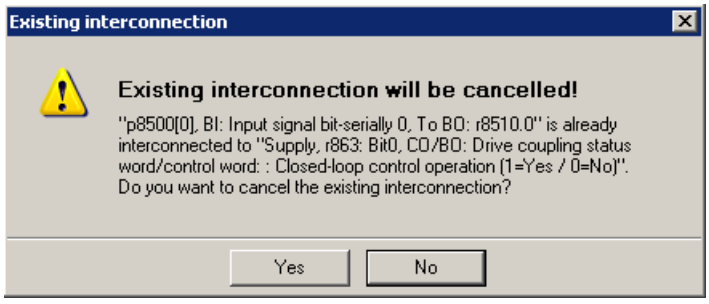


Figure 10-86 Information dialog box when an existing interconnection is disconnected

The interconnection options via CU-Link are explained below using an example.

Configuring example:

Task definition (example)

The ready signal of an infeed r0863.0 on CX32-2 (X102) is to be interconnected to three other CX32-2 Controller Extensions and to the drives of the SINAMICS Integrated via CU-Link. The items indicated the necessary interconnection measures, which are explained in detail below.

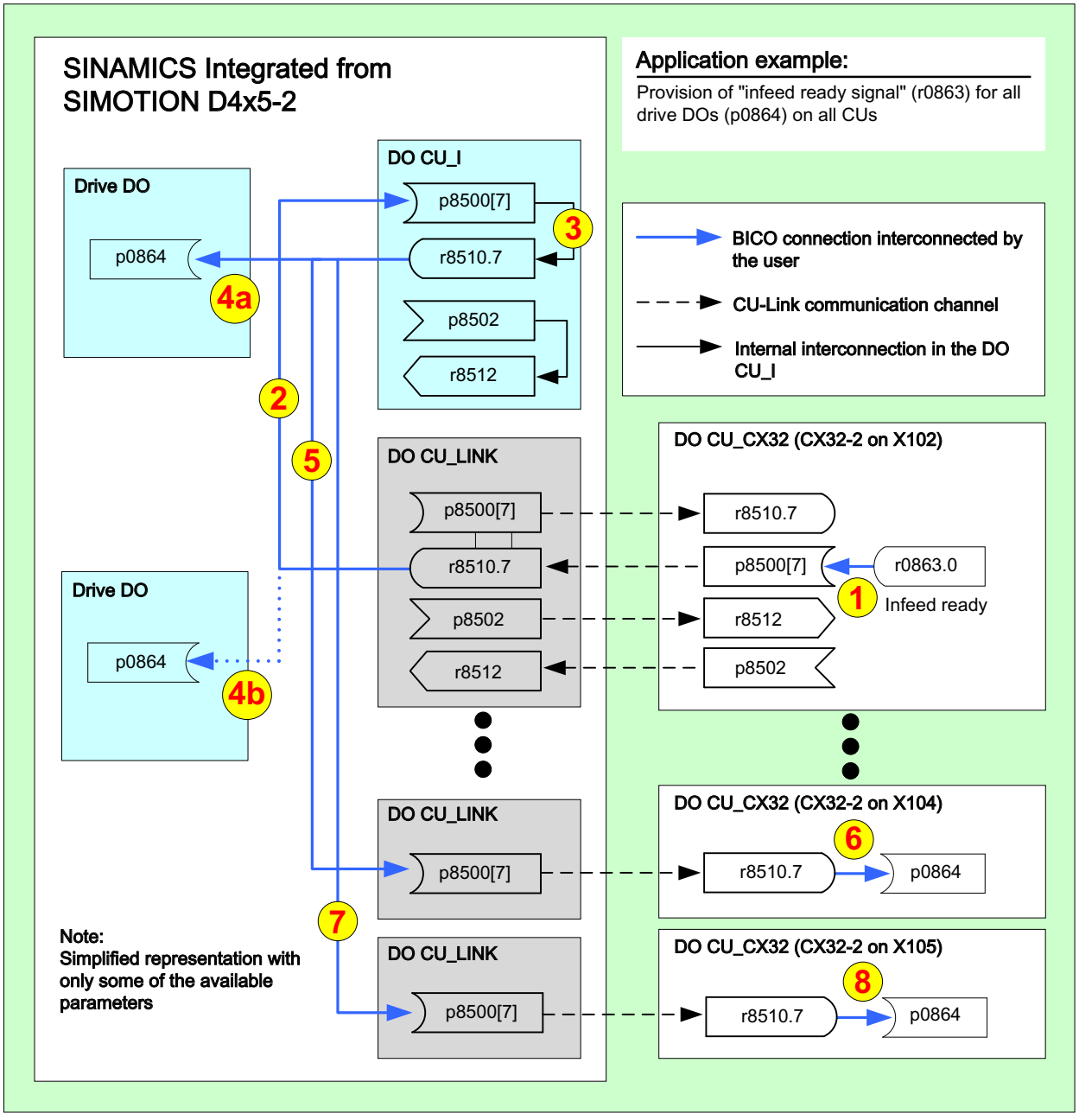
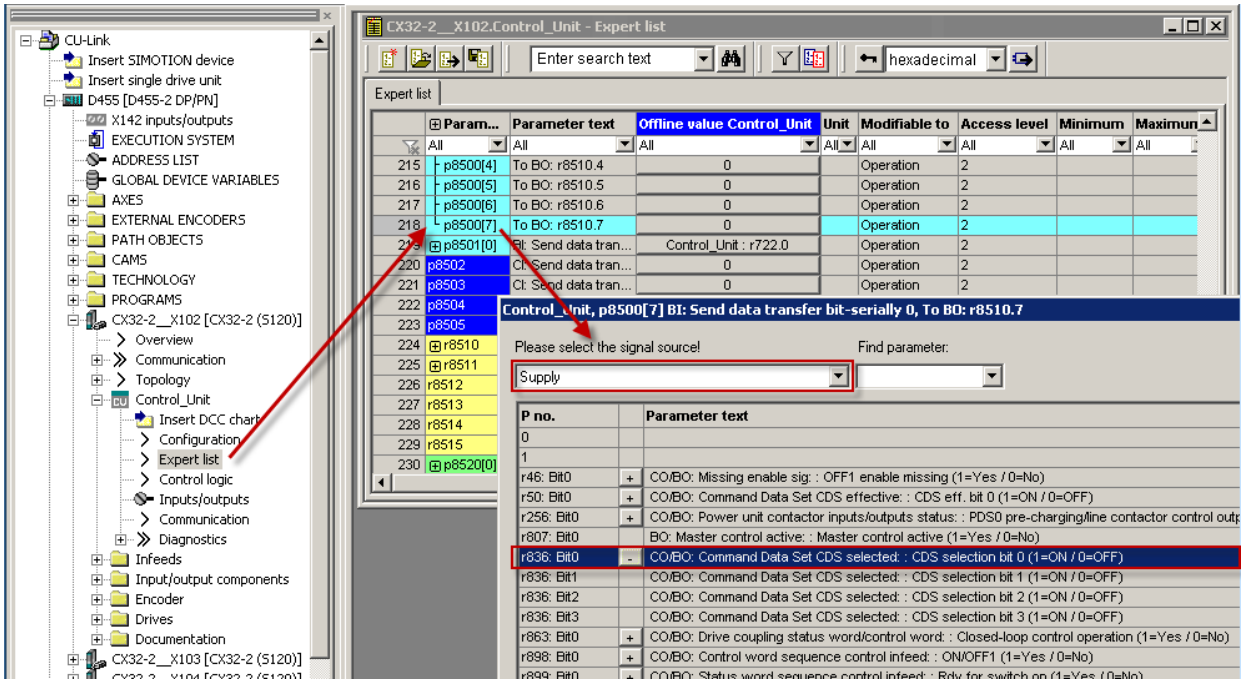


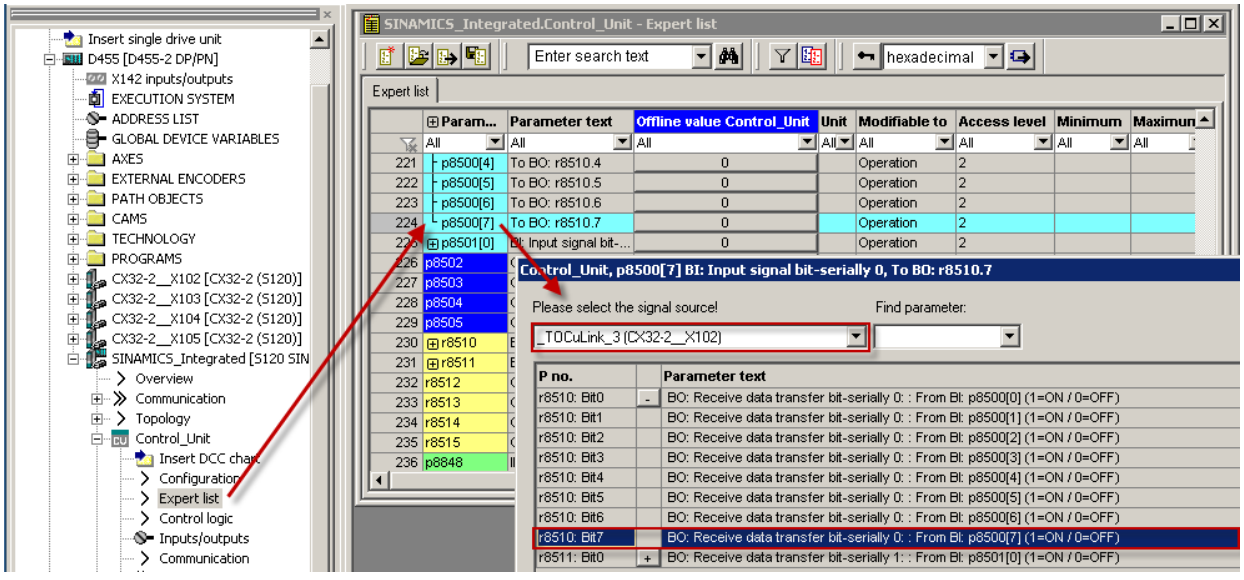
Figure 10-87 Configuration example

Step 1: In the expert list of the CX32-2 (X102), interconnect the "infeed ready" signal (r0863.0) of the infeed on the DO CU_CX32 (p8500[7]).

The status of the signal "infeed ready" is available on the CU-Link communication channel and can be further interconnected via the DO CU_LINK.



Step 2: In the expert list of the SINAMICS Integrated, interconnect the signal from the DO CU_LINK (r8510.7) of the CX32-2 (X102) to DO CU_I (cu.p8500[7]).



Step 3: In the DO CU_I of the SINAMICS Integrated, the parameter cu.p8500.7 "internal" is forwarded to the read parameter cu.r8510.7 (use of the transfer parameters).

Step 4: There are 2 ways of interconnecting the "infeed ready" drive signal (p0864) to drives of the SINAMICS Integrated:

- 4a: direct interconnection to the DO CU_LINK (preferred solution)
- 4b: Interconnection via the DO CU_I of the SINAMICS Integrated

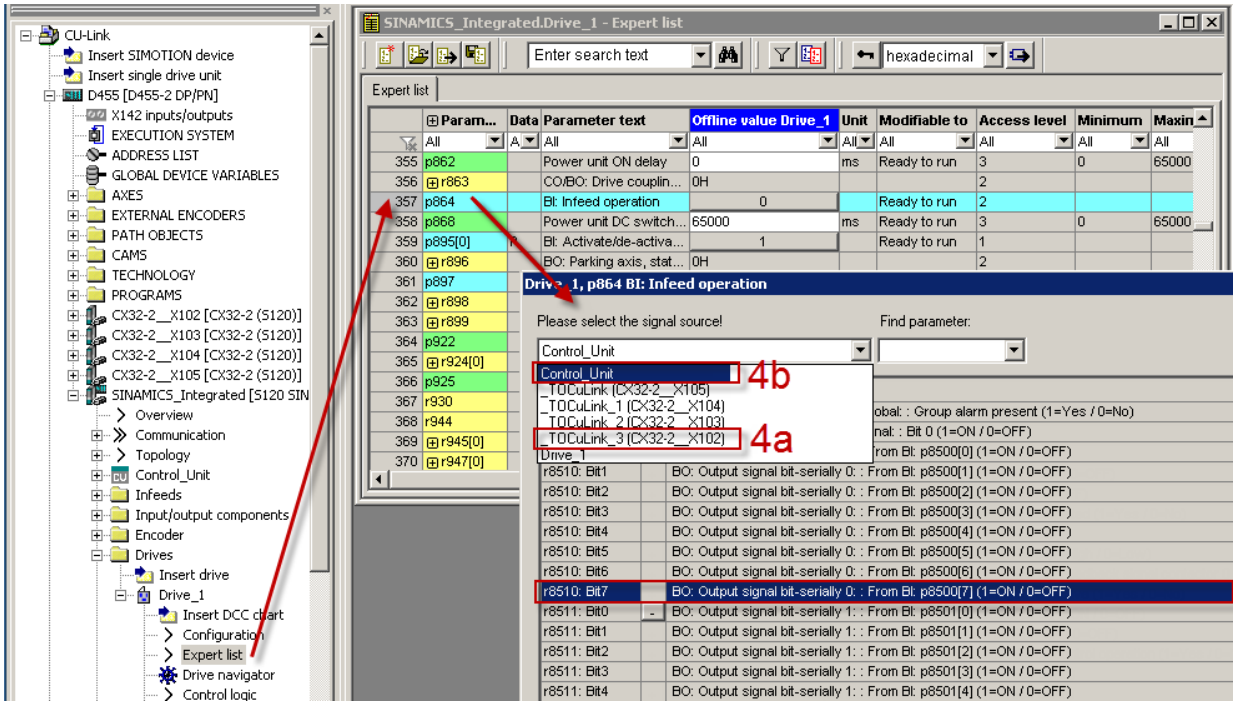


Figure 10-88 Step 4 - interconnect "infeed ready" drive signal (p0864)

Step 5: p8500[7] from the DO CU_LINK (X104) is interconnected with the parameter cu.r8510.7 of the DO CU_I of the SINAMICS Integrated.

This interconnection can only be made via "Diagnostics > Interconnections > Binector output (BO)" because there is no expert list for the CU_LINK DOs.

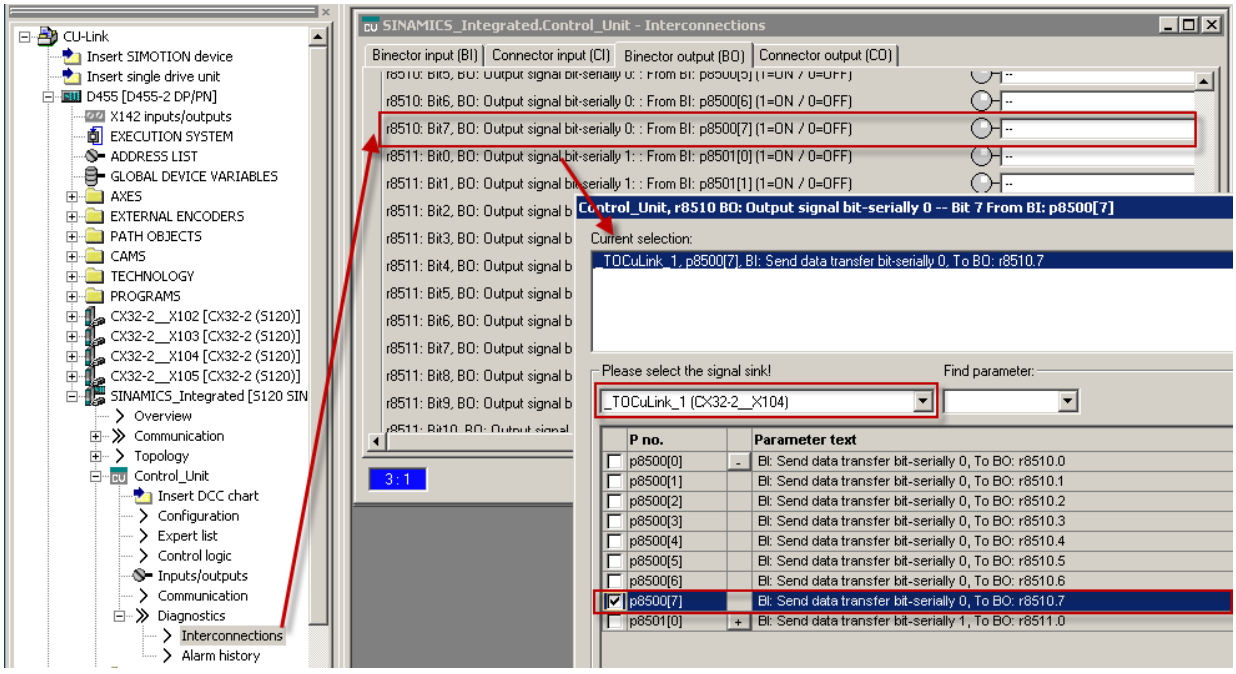


Figure 10-89 Step 5 - interconnect p8500[7] from the CU_LINK DO (X104) with parameter cu.r8510.7 of the DO CU_I

Step 6: On the drives of the CX32-2, the "infeed ready" drive signal (p0864) must be interconnected with the r8510.7 of the DO CU_CX32 of the CX32-2.

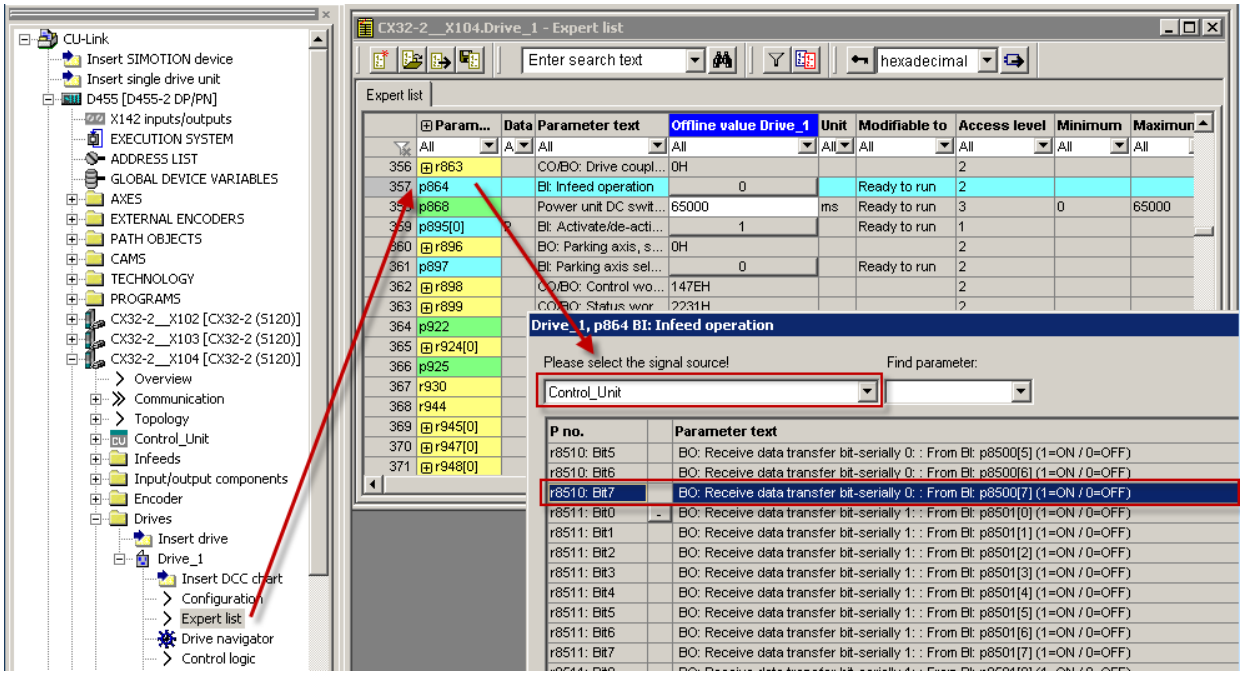


Figure 10-90 Step 6 - on CX32-2, interconnect the drive signal (p0864) with the r8510.7 of the DO CU_CX32 of the CX32-2

Step 7+8: To interconnect the "infeed ready" signal to further Controller Extensions CX32-2, repeat steps 5 and 6 for the CU-Link in question.

See also

For more information, see the List Manual *SINAMICS S120/S150,* function block diagrams 2179 to 2195.

Licenses for the SINAMICS Integrated / CX32-2

The following SINAMICS functions are licensed via SIMOTION:
- SINAMICS Safety Integrated Extended Functions for SIMOTION D

SINAMICS Safety Integrated Advanced Functions for SIMOTION D (as of V5.2)

- SINAMICS high output frequency for SIMOTION D

SINAMICS Integrated / CX32-2

SINAMICS Safety Integrated Extended Functions for SIMOTION D

Highly effective protection of personnel and machinery can be implemented with SIMOTION D thanks to the integrated safety functions of SINAMICS S120. Whereas the Safety Integrated Basic Functions are license-free, the Safety Integrated Extended Functions and Advanced Functions require a license for each drive with safety functions.

- SINAMICS Safety Integrated Extended Functions for SIMOTION D or

- SINAMICS Safety Integrated Advanced Functions for SIMOTION D

With the licenses, the Safety Integrated Extended and Advanced Functions are licensed for drives on the SINAMICS Integrated of a SIMOTION D Control Unit as well as on the CX32-2 Controller Extensions.

Licensing method: License for each drive with Safety Integrated Extended Functions or Advanced Functions. The Safety Integrated Advanced Functions also include the Safety Integrated Extended Functions.

Note

Instead of for each drive, the licensing can be performed for all integrated SINAMICS drives and CX32-2 Controller Extensions with the **MultiAxes and Safety Extended Package** for SIMOTION D.

SINAMICS high output frequency for SIMOTION D

Because of legal specifications, the output frequency for converter systems is limited to 550 Hz per default.

- SINAMICS S120 Control Units as of firmware V4.7 HF7

- SIMOTION D Control Units as of firmware V4.4 HF6

The **SINAMICS high output frequency for SIMOTION D** license allows users to cancel the output frequency limit if, as a result of the application, output frequencies of more than 550 Hz

are required for drives. The license cancels the limitation **for all drives** that are connected to the SINAMICS Integrated of a SIMOTION D Control Unit as well as to the CX32-2 Controller Extensions.

Licensing method: License for each SIMOTION D Control Unit

Note

The display of the Runtime licenses in the SCOUT Licensing dialog is updated after the power-up of the SINAMICS Integrated and the CX32-2 Controller Extensions.

SINAMICS Control Units

SINAMICS Control Units connected via PROFINET or PROFIBUS (e.g. CU310-2, CU320-2) are licensed via the SINAMICS memory card. Licensing via SIMOTION is not possible.

Further information

- Changes to the EC Dual Use regulation and how it impacts SIMOTION
  https://support.industry.siemens.com/cs/ww/en/view/104020777
- SINAMICS S120/S150: The "High output frequency" license delivery release
  https://support.industry.siemens.com/cs/en/en/view/104020669

Support of motors with PT1000 temperature sensor

Because of the discontinuation of the KTY84-130 temperature sensor, SIMOTICS motors will be equipped with PT1000 temperature sensor in the future.

- Motors with DRIVE-CLiQ interface remain compatible and can be operated on any SW version.
- Motors without DRIVE-CLiQ interface require SIMOTION SCOUT as of V4.5 for the configuration of the PT1000 temperature sensor.

Support of Motor Modules (C/D type)

The SINAMICS S120 booksize C/D type Motor Modules are a further development of the previous Motor Module and can be used compatibly with regard to the configuration.

The new Motor Modules partly extend the previous spectrum or the functionality. In this case, there are special conditions of use.

Table 10-49 Conditions of use for Motor Modules C/D type

Motor Module	Article number	Conditions of use
Single Motor Module 18 A C type	6SL3120-1TE21-8A C0	These Motor Modules extend the previous spectrum and can only be used with: • **SIMOTION SCOUT V4.5** – With SIMOTION Runtime/firmware: V4.4 as of HF11 (SINAMICS Integrated V4.7 as of HF17) or – With SIMOTION Runtime/firmware V4.5 (SINAMICS Integrated V4.8) and installed SSP SINAMICS V4.8 HF1 • **SIMOTION SCOUT TIA V4.5** – With SIMOTION Runtime/firmware: V4.4 as of HF11 (SINAMICS Integrated V4.7 as of HF17) – NOT with SIMOTION Runtime/firmware V4.5 (SINAMICS Integrated V4.8) -> SSPs cannot be installed in the TIA Portal and therefore cannot be used with SCOUT TIA) • **SIMOTION SCOUT / SCOUT TIA as of V5.1** – With SIMOTION Runtime/firmware: As of V4.4 HF11 (SINAMICS Integrated as of V4.7 HF17) **SCOUT V4.4 HFx** Does not contain an integrated STARTER V4.4 with Service Pack 1, which is required for the configuration of these three Motor Modules. For details, see also: Link (https://support.industry.siemens.com/cs/ww/en/view/109480497)
Double Motor Module 18 A D type	6SL3120-2TE21-8A D0	
Single Motor Module 30 A D type	6SL3120-1TE23-0A D0	
Single Motor Module 24 A C type	6SL3120-1TE22-4A C0	**Engineering: SCOUT / SCOUT TIA as of V5.1** and SIMOTION Runtime/firmware as of SIMOTION V4.5 (SINAMICS Integrated as of V4.8) The 24 A Motor Modules require a new firmware in each case. The 45 A and 60 A Motor Modules require a current firmware if the maximum current of 90 A or 120 A is to be utilized. For details, see also: Link (https://support.industry.siemens.com/cs/ww/en/view/109751266)
Single Motor Module 24 A D type	6SL3120-1TE22-4A D0	
Single Motor Module 45 A C type	6SL3120-1TE24-5A C0	
Single Motor Module 60 A C type	6SL3120-1TE26-0A C0	

The latest information on the SINAMICS components that can be used is available at:

FAQ (https://support.industry.siemens.com/cs/de/en/view/11886029)

Testing a drive with the drive control panel

Introduction

You can test a configured drive with the drive control panel, where you can specify a speed and adjust it with a scaling factor. The drive control panel should only be used for commissioning.

Requirements

- The project has been downloaded to the target system.
- SIMOTION SCOUT is in online mode.
- The drive is not being used by a current project in RUN mode.

⚠ WARNING
Danger to life from unexpected movement of driven machine parts
Make sure this presents no hazard to personnel or property.

Testing a drive with the drive control panel

1. Change to the configured drive in the Project Navigator and open the drive control panel by selecting "Commissioning" > "Control panel". The drive control panel opens in the detail view.

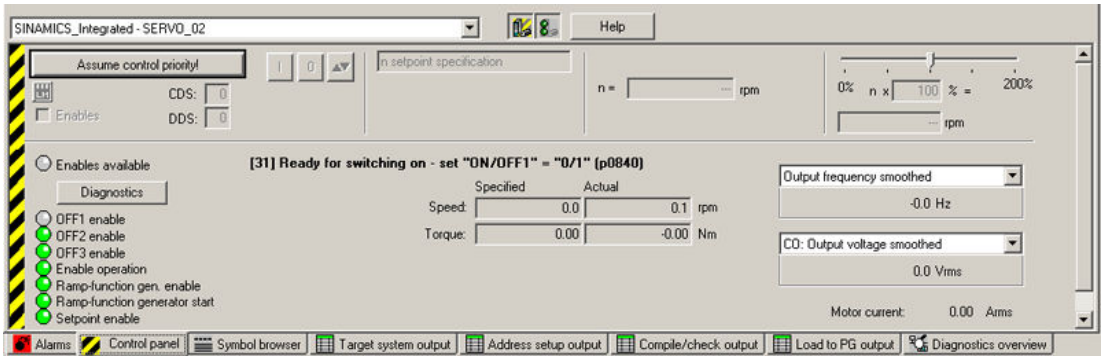


Figure 10-91 Drive control panel

2. To display the control area and axis diagnostics, click the "Show/hide control area" and "Show/hide diagnostics area" buttons.

3. Click "Assume control priority". The "Assume Control Priority" dialog box is opened.

Note

If you are using an infeed without a DRIVE-CLiQ interface, you will have to interconnect the "Infeed operation" signal (drive parameter p0864) yourself. If you are using an infeed with a DRIVE-CLiQ interface, select the infeed for which the control priority is to be assumed under "Infeed" in the "Assume Control Priority" dialog box.

If the infeed signal "Closed-loop control operation" already has a BICO interconnection to the drive, the infeed is permanently specified (infeed selection and checkbox are grayed out).

The infeed must be switched on before the drive can move ("LM" button, switch infeed on/off).

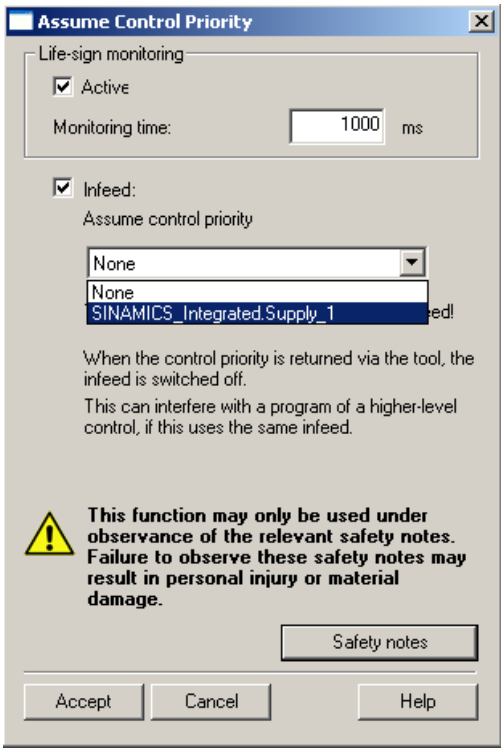


Figure 10-92 Assume_control_priority

4. Read the notes and confirm these with "Accept."
5. Activate the "Enables" checkbox to enable the drive.
   All enables are now set with the exception of ON/OFF1.

6. Enter the desired setpoint in the entry field, and, as a safety setting, slide the scaling to 0%.

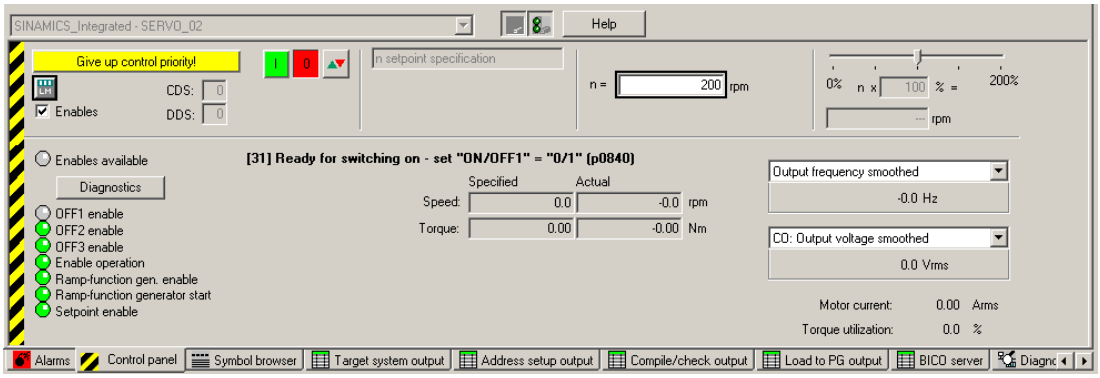


Figure 10-93 Entering a setpoint

7. Click the "Drive On" button. The "Enable is set" LED lights up in green color. If you move the slider to the right, the drive rotates. The current motor speed is displayed in "Actual."
8. Click "Drive Off" to stop the drive after you completed the test.
9. Deactivate the enable and click the "Give up control priority" button to deactivate control from the PG/PC. In this state, you can no longer make the drive rotate.

Creating and testing axes

Overview of SIMOTION engineering

Performing engineering with SIMOTION SCOUT

You use the engineering software to configure the individual axes and define the project sequence by means of programs.

1. First, run through the axis wizard to configure the axes and interconnect to the real drive (e.g. SINAMICS Integrated).
2. Provided you have completed the configuration at the drive end, we strongly recommend for faster working that the SINAMICS Integrated is deactivated via "Target system" ⇒ "Select target device".
3. Complete your SIMOTION application, for example, by creating axis functions and SIMOTION execution programs.
4. Compile the project and download it to the SIMOTION D4x5-2.

Creating an axis with the axis wizard

Overview

The TO axis provides the user with the technological functionality and the interface to the drive/actuator. The TO axis processes the motion control commands from the user program (e.g. MCC) and coordinates the interface to the drives. It executes control and motion commands and indicates statuses and actual values. The TO axis communicates with an actuator (drive or hydraulic valve) via a fieldbus system (PROFIBUS or PROFINET via PROFIdrive protocol) or via a direct setpoint interface (analog ±10 V or pulse/direction).

When running through the axis wizard, the basic settings are made for the axis and the TO axis interconnected to a drive (e.g. SINAMICS Integrated). Further options are available when "Use symbolic assignment" has been activated:

- A real axis is interconnected to an already configured drive
- A real axis including drive is created via the axis wizard and the drive interconnected to the axis
- A real axis is created without assigning this to a drive (assignment is made later)

Inserting an axis

1. In the Project Navigator, double-click the entry "Axis" > "Insert Axis."
   This will access the axis wizard. Set the required technology and then click "OK."

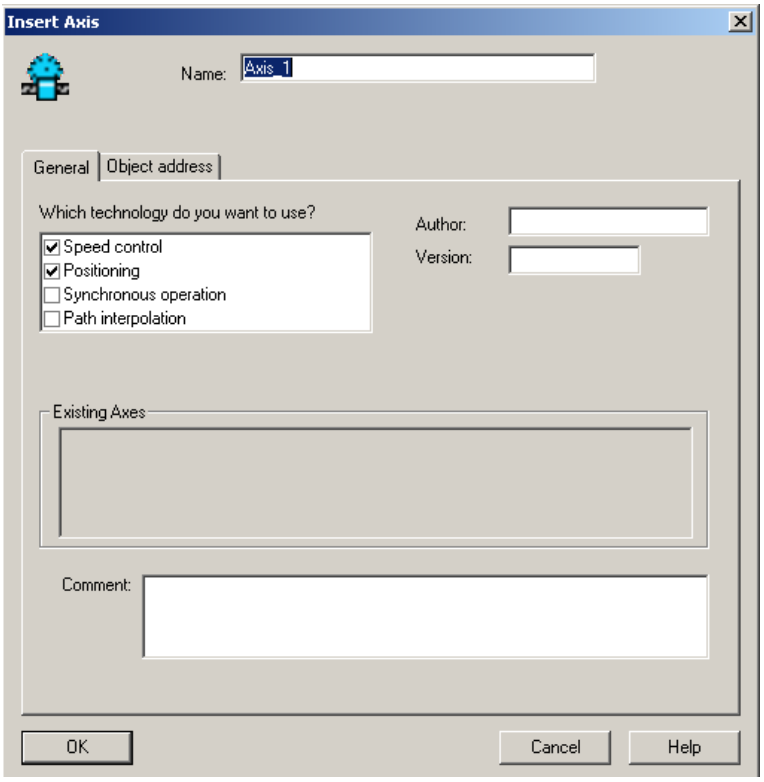


Figure 10-94 Inserting an axis

2. Set an axis type and, if required, configure the units.

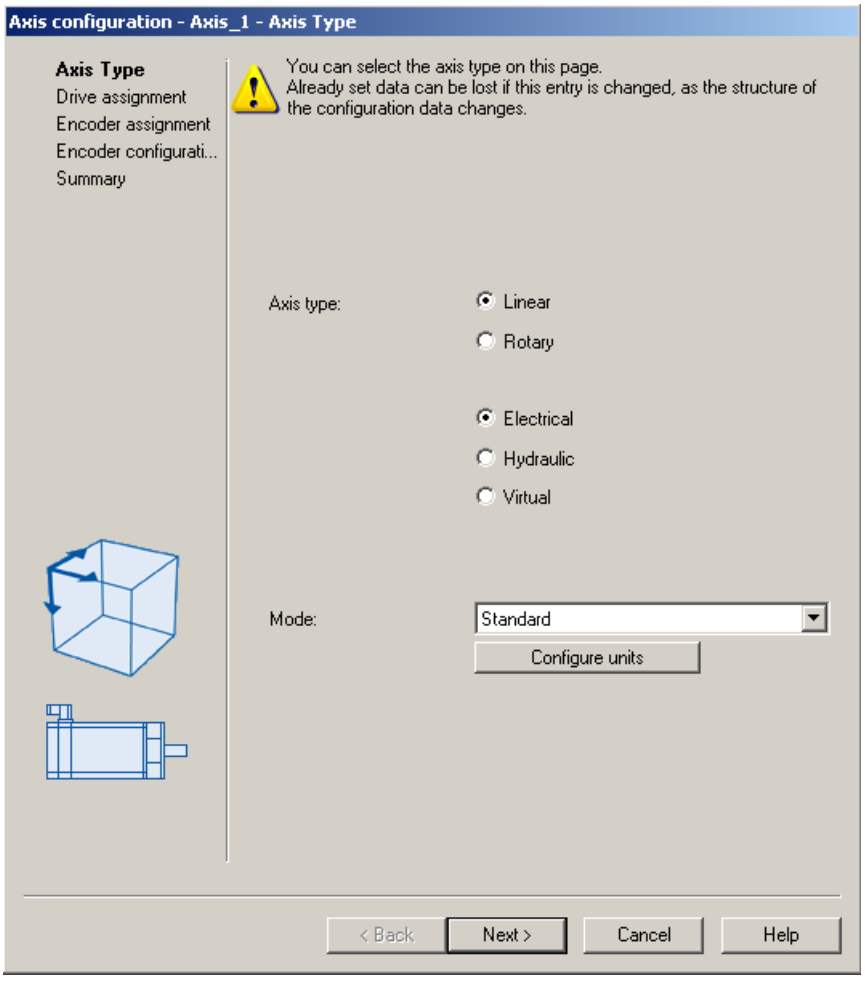


Figure 10-95 Defining the axis type

3. Create a new drive or make the assignment to an existing drive.

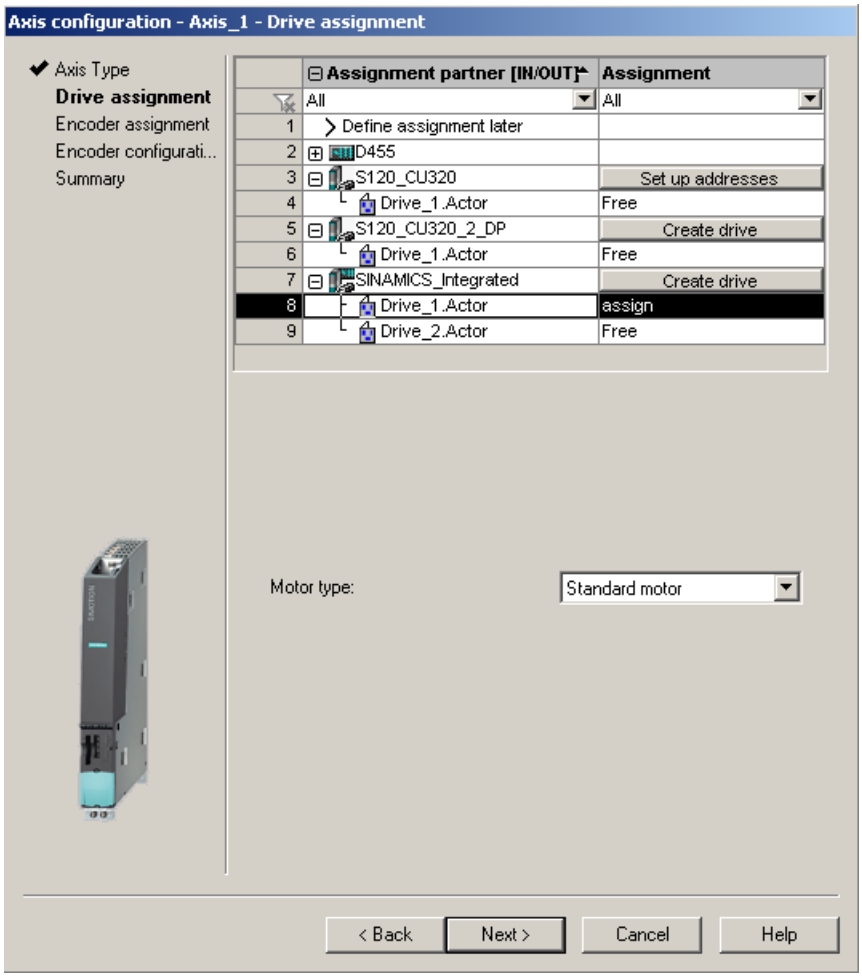


Figure 10-96 Assigning a drive

The following setting options are available for the drive assignment:

– Assign drive
  Assigning a previously configured drive

– Define a subsequent assignment
  The axis should not be assigned to a drive until a later point in time.
  In this way, the PLC and motion control functions can be completely configured by a programmer even without drive know-how using technology objects (e.g. TO axis) and loaded to the device, the drives can be configured and optimized separately by a drive expert, and the technology objects can be assigned symbolically later to the drive objects via an interconnection dialog box.

– Create drive
  From the assignment dialog box, a new drive can be created on an existing drive unit (e.g. S120 CU320-2 or SINAMICS Integrated) and assigned to the axis. This allows the axis, including the drive, to be created in one operation. It is not necessary to configure a drive before creating an axis.

– Set up addresses
  The addresses must be set up manually if "Use symbolic assignment" has been deactivated. This corresponds to the previous methods of the drive and axis configuration, available in SCOUT versions < V4.2. This is required, for example, for drive units that do not support symbolic assignment (e.g. SINAMICS S120 with FW version < 2.6.2, MASTERDRIVES, SIMODRIVE, etc.).

The address list in the "All Addresses" view provides an overview of the assignments to all interfaces of the TO axis. From this view, the assignments can also be changed via the Interconnection dialog box (... button).

Note

The methods of the drive and axis configuration, previously available in SCOUT versions < V4.2, are still available. Symbolic assignment must be deactivated for these methods.

4. Run through the wizard and enter the settings of your system. The required axis telegrams as well as the addresses used are automatically specified by the engineering system. Telegrams are also extended and interconnections created automatically in the drive depending on the selected TO technology (e.g. SINAMICS Safety Integrated).
5. Click **Finish** to confirm the "Summary" window.
   The configured real axis is displayed in the project navigator.

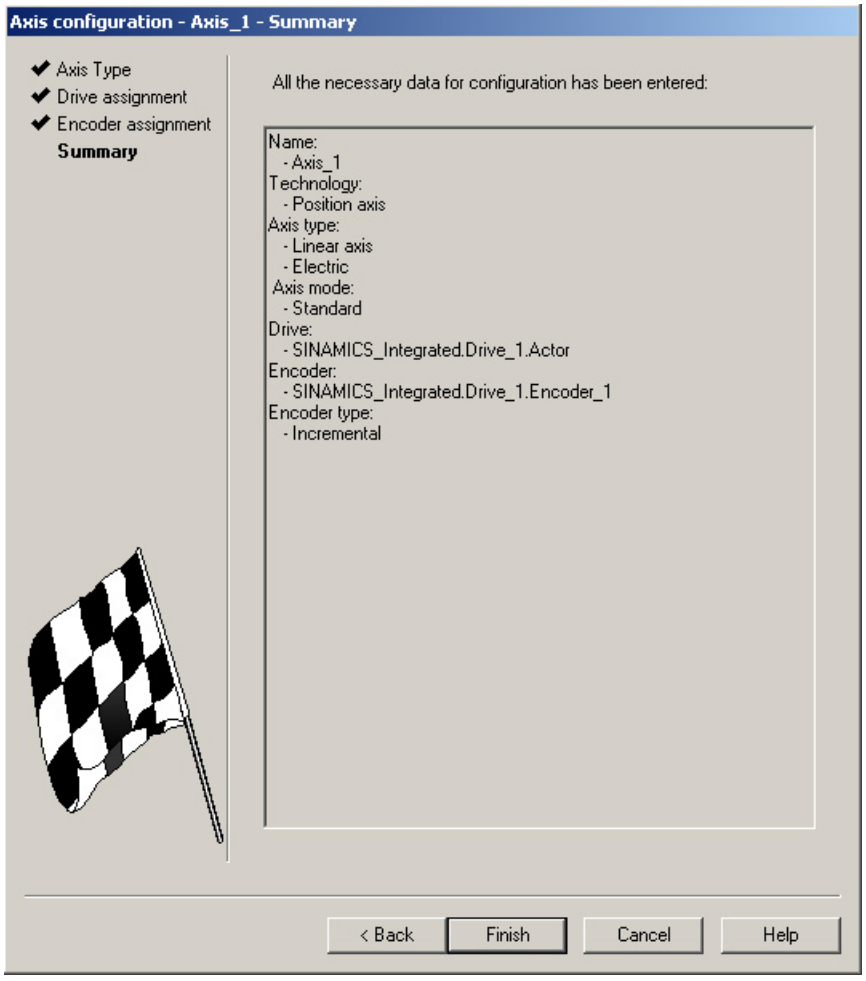


Figure 10-97 Axis wizard summary

Note

During system power-up, reference variables as well as drive and encoder data of the SINAMICS are automatically taken over for the SIMOTION configuration data of the SIMOTION technology objects "TO axis" and "TO externalEncoder".

Encoder assignment

With a position axis, encoder 1 is also created on the TO axis (motor encoder) and automatically assigned to the first encoder on the drive.

If encoder 2 (direct encoder) is created at the TO axis, it is assigned to the second encoder of the drive control.

Result

The configured axis will appear in the project navigator.

Save and compile the project and download it to the target system.

On completion of the axis wizard, the symbolic drive assignment is visible:

- Via the "configuration" of the axis
- Via the address list (view all addresses)

The Assignment dialog box can be called again from these dialog boxes using the ... button.

Instead of calling the Assignment dialog box, it is also possible to edit the input field SINAMICS_Integrated.SERVO_02.Actor ... containing the symbolic name directly.

TBD, DSDB, and SIDB

In the "Configuration" dialog box of the TO axis, you can activate the following functions from "Functions" > "Change":

- Technology data block (TDB): for the cyclic exchange of technology data, e.g. actual torque value
- Drive Safety data block (DSDB), V4.4 and higher: to support the SINAMICS Safety Integrated Functions by means of the TO
- Safety Info Data Block (SIDB): Predecessor of the DSDB (for compatibility only)

The assignment is always made to the drive DO of the actuator of the axis. The system automatically generates a telegram extension and the BICO interconnection of the relevant SINAMICS parameters.

Note

The safety data blocks (DSDB or SIDB) are automatically configured by the engineering system and interconnected in the drive.

The PROFIsafe telegram must be configured by the user.

If the activation of the safety functions is to be made using PROFIsafe, configure the PROFIsafe communication to the higher-level SIMATIC F-CPU (see *SINAMICS S120 Safety Integrated* Function Manual).

I/O signals at the TO axis

For the assignment of I/O signals on the TO axis (e.g. the inputs for the homing output cam or hardware limit switches), call the assignment dialog box from the parameterization dialog boxes of the TOs created or from the address list (view of all addresses) by clicking the ... button.

See also

Downloading a project into the target system (Page 7074).

For more information on symbolic assignment, see the *SIMOTION Runtime Basic Functions* Function Manual.

Testing the axis with the axis control panel

Axis control panel

The axis control panel is used exclusively for testing axes. You can use the axis control panel for the following tasks, for example:

- To test each part of the system individually before program-driven axis motions are initiated.
- In the event of an error, to test whether the individual axes and drives can be traversed from the axis control panel
- To traverse the axes for optimization purposes (controller optimization)
- To perform active homing
- To set and remove the axis enable
- To test an axis that has been created

Requirement

The following requirements must be fulfilled for testing:

- The project has been downloaded to the target system.
- SIMOTION SCOUT is in online mode.

Axis test

1. Open the AXES folder in the project navigator and click the "Control Panel" entry below the axis (for example, Axis_1).
   The axis control panel is displayed.

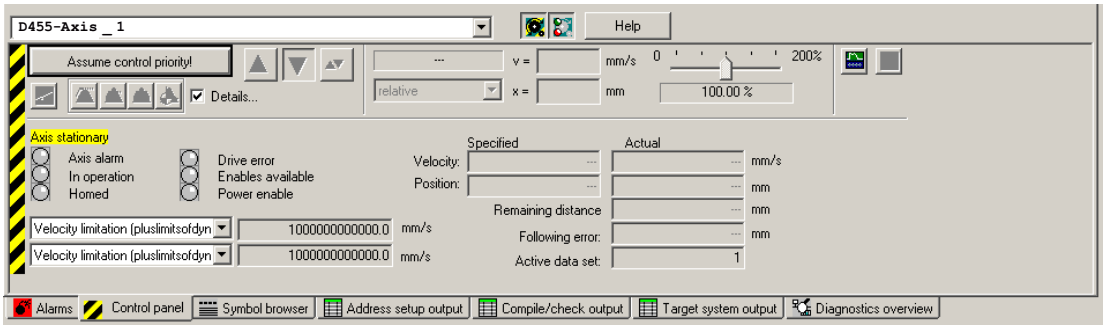


Figure 10-98 Axis control panel

2. To display the control range and axis diagnostics, click the "Show/hide control range" and "Show/hide diagnostics area" buttons.
3. Click "Assume control priority".

   Note

   In order to move the axis from the PG/PC, you must assume control priority. However, by pressing the SPACER bar, you can stop the axis at any time.

4. The further procedure depends on the CPU status:

   – Case 1: CPU in the STOP/STOPU state
   If the CPU is in the STOP state, a message will appear, indicating that the CPU has been put in the STOPU state.
   A safety message appears in a further dialog box and must be accepted.
   The activated Service function is then displayed via the LEDs (RUN flashes green at 2 Hz and SU/PF flashes yellow at 2 Hz).

   – Case 2: CPU in the RUN state (as of SCOUT/Kernel V4.4)
   Control priority can only be assumed if the axis is not in motion.
   After accepting the safety message, a message is shown on the control panel that the CPU is in the RUN state, that the user program is running, and further axes may be moving (LED display: RUN).
   If the control priority for a TO is active, commands for the TO from the user program are rejected with an error code. Alarm 30009: reason 0x04 is output.

5. To enable the axis, click the "Set/remove enable" button. Confirm the "Switch Axis Enable" dialog box with "OK".

   Note

   If the control panel is operated in the RUN state, switching the power supply on/off and setting/canceling the axis enable can alternatively be controlled via the user program.

   Note

   If you are using **an infeed without a DRIVE-CLiQ interface**, you will have to interconnect the "Infeed operation" signal (drive parameter p0864) yourself.

   If you are using **an infeed with a DRIVE-CLiQ interface**, select the infeed for which the control priority is to be assumed under "Infeed" in the "Switch Axis Enable" dialog box. Select this checkbox when the control priority is to be fetched and activated. If the infeed signal "Closed-loop control operation" already has a BICO interconnection to the drive, the infeed is permanently specified (infeed selection and checkbox are grayed out).

6. To traverse the axis, click the "Position-controlled traversing of the axis" button.
7. Enter a velocity and close the dialog box by clicking "OK".
8. Click the "Start motion" button to traverse the axis. You can monitor the traversing motion under velocity and position. Use "Stop motion" to stop axis movement again.
9. To cancel the enable and switch off the infeed, click the "Set/remove enable" button. Confirm the "Remove Axis Enable" dialog box with "OK".
10. Click the "Give up control priority" button to deactivate axis control from the PG/PC. In this operating state, the axes can no longer be controlled from the PG/PC.

    Note

    For commissioning kinematics transformations, a path control panel is available as of SIMOTION V4.4.

See also

Please refer to the additional information in the SCOUT Online Help (Axis control panel index).

Activating the infeed (line module)

Requirement

Before a drive can be traversed, the infeed (line module) must be switched on and the "Closed-loop control operation" signal of the infeed must be present at the drives.

If this is not taken into account, this can result in a fault state, and in the worst case, damage to the infeed.

How the infeed is controlled and how the drives receive the Ready signal depends on the type of infeed used.

A distinction is made between:

- Infeeds without DRIVE-CLiQ connection
- Infeeds with DRIVE-CLiQ connection

Infeed without DRIVE-CLiQ connection

The infeed is controlled via terminals on the line module, e.g. DO Ready (DC link is precharged and ´pulse enable is set).

These terminals must be connected to the SIMOTION D4x5-2 (e.g. wire "DO Ready of the infeed" to "X122, DI 0 of the D4x5-2").

The enable signal is queried when the drive is configured and must be interconnected accordingly.

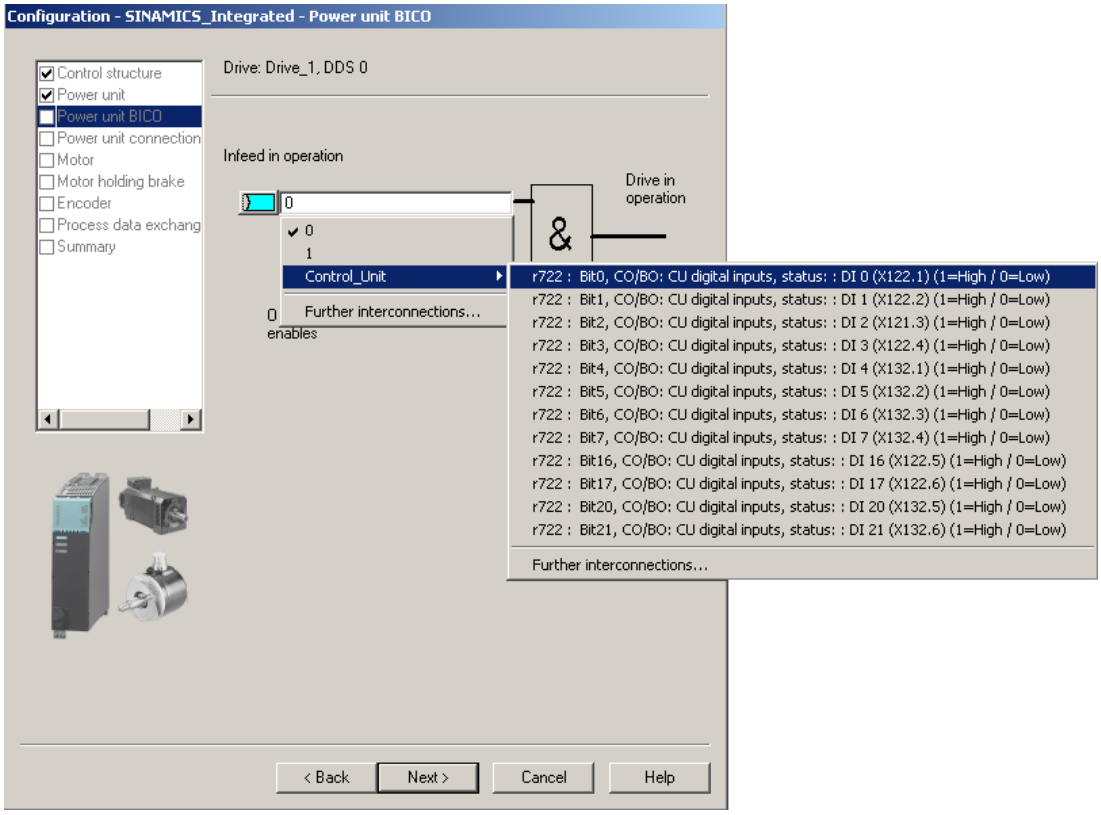


Figure 10-99 Interconnection of the "Infeed in operation" signal

Infeeds with DRIVE-CLiQ connection

The infeed is controlled via DRIVE-CLiQ. The infeed is switched on or off by the SIMOTION D4x5-2 via PROFIdrive message frame 370.

If "Use symbolic assignment" has been activated, message frame 370 is set automatically when setting standard/automatic (see Section Calling the drive wizard (Page 7053)).

With message frame 370, the required BICO interconnections to the message frame are created automatically on the drive unit.

If an infeed with DRIVE-CLiQ connection has already been created, the infeed signal "Closed-loop control operation, r863.0" is automatically interconnected to "Infeed operation, p0864" of the drive when drives are inserted (only applies to drives that are attached to the same drive unit as the infeed).

The **FB _LineModule_control** function block is available on the controller for control and diagnostics of the infeed.

Note

Always create the infeed first so that the infeed "Closed-loop control operation" signal is automatically interconnected to a drive when this is inserted.
Recommendation: If the drive unit has not been configured, use "Configure drive unit" in the project navigator. "Configure drive unit" has the advantage that all basic settings for the drive unit are queried via one wizard.

FB _LineModule_control

The **_LineModule_control** function block (FB) can be used to switch the infeed on and off and also perform simple diagnostics.

The following infeeds are supported by the **_LineModule_control** function block:

- Basic Line Modules (BLM)
- Smart Line Modules (SLM)
- Active Line Modules (ALM)

The **_LineModule_control** FB is part of the command library of the "SIMOTION SCOUT" engineering system. You can find the FB at "Drives" > "SINAMICS".

For detailed information on the **_LineModule_control** FB, refer to the SIMOTION SCOUT online help or the *Standard Function for SINAMICS S120 Line Modules* Function Manual.

CX32-2

Further information on how to use line modules with the CX32-2 can be found in Section Interconnecting the infeed "Operation" signal on the CX32-2 (Page 7098).

Additional references

For a detailed description of the control words and status words of the PROFIdrive message frames for SINAMICS S120 drives, see the *SINAMICS S120 Commissioning Manual*.

Using one infeed for several CUs

If the infeed is controlled from another control unit, then the Ready for operation signal of the infeed (parameter r0863.0) must be connected to parameter p0864 "Infeed operation" of the drive via a digital input/output. If this is not taken into account, the infeed may be damaged.

The following table shows which interconnection options are available, depending on the topology.

Table 10-50 Interconnection options for the ready signal of an infeed

Infeed is controlled by ...	Ready signal of the infeed to be interconnected to ...	Recommended interconnection option for the ready signal of the infeed
SINAMICS Integrated of D4x5-2	CX32-2	① Default CU-Link interconnection
	CU320-2	③ Wiring via terminals
CX32-2	SINAMICS Integrated of D4x5-2	② Individual CU-Link interconnection
	CX32-2	
	CU320-2	③ Wiring via terminals
CU320-2	SINAMICS Integrated of D4x5-2	③ Wiring via terminals
	CX32-2	
	CU320-2	

① Use of the CU-Link (DRIVE-CLiQ data transfer channel); by factory setting, the ready signal of an infeed on the SINAMICS Integrated is automatically communicated to the connected Controller Extensions and only has to be interconnecgted to the drives there.
For details, see Section Interconnecting the infeed "Operation" signal on the CX32-2 (Page 7098).

② For the interconnection, the CU-Link (DRIVE-CLiQ data transfer channel) can be used. For this purpose, it is necessary to configure the CU-Link according to the requirements.
For details, see Section CU-Link (data transfer via DRIVE-CLiQ) (Page 7118).

③ Wiring via terminals is recommended.
An interconnection of the Ready for operation signal via PROFIBUS/PROFINET is not recommended because of the delay times (communication, evaluation in controller, etc.) or a possible CPU stop (no execution of the user program).

Setting up addresses and message frames

Overview

After all SINAMICS components have been configured, addresses must be determined for the process data exchange between the drive and the controller.

This procedure depends on whether **symbolic assignments** are used:

- With **symbolic assignment**, the addresses are determined automatically by the engineering system, see Section Setting up communication for symbolic assignment (Page 7147).
- Without symbolic assignment, the determination of the addresses must be started manually, see Section Telegram configuration (Page 7148).

Setting up communication for symbolic assignment

The communication for symbolic assignment is set up with the following actions:

- In the SCOUT menu (call "Project" > "Set up communication for symbolic assignment")
- At "Download project to target system"
- At "Save project and compile changes"

When setting up the communication, the message frames, BICO interconnections and addresses are set up for the entire project.

See also

Message frame configuration (Page 7148)

Message frame configuration

Requirement

You have configured the drive unit.

On the basis of this configuration, one or more of the following actions should be performed:

- The automatic PROFIdrive telegram setting for a drive object should be activated/deactivated
- The automatic telegram extension for a drive object should be activated/deactivated
- The automatic address adaptation for a drive object should be activated/deactivated
- PROFIdrive telegrams should be configured for drive objects
- The addresses should be set up
- Telegrams should be extended manually

Procedure

Proceed as follows:

In the project navigator, open the "Communication" > "Telegram configuration" entry under "SINAMICS_Integrated."
The "SINAMICS Integrated - telegram Configuration" dialog box is displayed with the PROFIdrive PZD telegrams tab.

The dialog box lists all the available drive objects. The possible setting options are described in the following.

When using the symbolic assignment, the default setting does not have to be changed and no configuration performed.

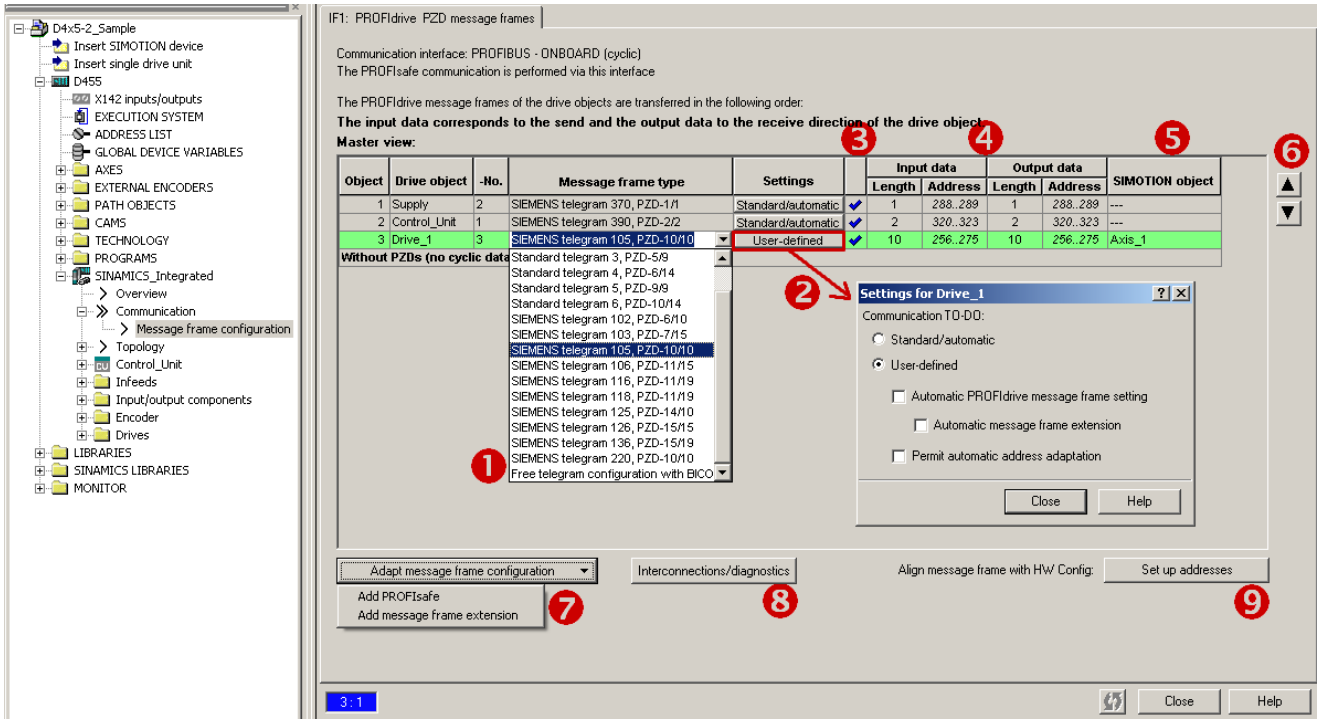


Figure 10-100 Telegram configuration

Table 10-51 Explanation of the figure

Number	Meaning
1	Selection of a telegram • The drive telegrams (telegrams 1 ... 6 and telegram 1xx) are defined in accordance with the PROFIdrive specification and can be selected based on the required functional scope. • You can transfer the signals of the I/Os or the global measuring inputs, for example, via the telegrams 39x. Telegram 39x is also required for the time synchronization between SIMOTION and SINAMICS. • Free telegram configuration with BICO allows you to define your own telegram. • Free telegram configuration with p0915/p0916 (for TM15/17). • Telegrams 37x for control of the infeed.
2	The "Standard/automatic" and "User-defined" settings are only visible if "Use symbolic assignment" is activated. Using the setting "Standard/automatic" is generally recommended. The "User-defined" setting allows the automatic telegram setting, telegram extension, and address adaptation to be deactivated or activated. • "Automatic PROFIdrive telegram setting" allows the telegram to be set by the system depending on the configured technology (telegram selection, e.g. for infeed, drive, and control unit incl. onboard I/O). • "Automatic telegram extension" allows the telegram to be extended by the system depending on the configured technology (e.g. if the technology data block is activated in the axis configuration). • "Permit automatic address adaptation" allows addresses to be adapted by the system in the case of address offsets, for example. Address offsets can occur, for example, if a telegram is extended and the adjacent addresses are already occupied by other telegrams. With TM15/TM17 High Feature, "Automatic PROFIdrive telegram setting", "Automatic telegram extension", and "Automatic address adaptation" cannot be deactivated by design, since for these drive objects the telegram is always set up in accordance with the parameterized terminal functionality (DI, DO, output cam, measuring input) and cannot be extended. "Automatic PROFIdrive telegram setting" and "Automatic telegram extension" must be deactivated if the telegrams are to be configured manually for TM15 DI/DO, TM31, and TB30 and interconnected with BICO. See Section Setting up communication for symbolic assignment (Page 7147).
3	Telegram status The icons in the status column show the following information: The message frame is configured differently in HW Config. You must align it with HW Config. You are using a predefined standard message frame or free BICO interconnection. You are using a modified standard message frame, which you have extended to include additional data. You are using a message frame for which one of the two message frame lengths is too long. The drive object cannot process this entry.
4	Length: Displays the size of the telegram component. Address: Address area in HW Config. The addresses will be displayed only after they have been set up.
5	Displays the SIMOTION object that is interconnected to the SINAMICS object (e.g. axis or encoder).
6	Changing the telegram order **Note**: Before the alignment, all drive objects without I/O addresses ("---..---") must be moved behind the objects with valid I/O addresses or those still to be aligned ("???..???").

Number	Meaning
7	"Manual" adaptation of the telegram configuration (e.g. when additional data, such as a motor temperature, is to be transferred via the telegram).
8	Display of the individual control and status words of the associated telegram.
9	Setting up the addresses (alignment of the addresses with HW Config) Only the addresses for the respective drive unit are determined (no automatic determination of telegrams / BICO interconnections).

Note

If symbolic assignment is deactivated, the following applies:

If the telegrams for drive objects (drives, Terminal Modules, etc.) change, you must set up the addresses again. The addresses are not updated automatically.

Error correction (symbolic assignment deactivated)

Based on the 39x telegram, SIMOTION generates further configuration information (FastIO configuration) for the following functions:

- Time-of-day synchronization SIMOTION ⇔ SINAMICS
- Use of onboard I/Os of SIMOTION D, CU, or CX
- Use of cams and global measuring inputs
- System function _setDriveObjectSTW

If the telegrams are defined manually (symbolic assignment is deactivated), a telegram 39x must be set up in the telegram configuration. The telegram then has to be aligned with HW Config via "Set up addresses."

If use of the functions stated above is not possible, generate the FastIO configuration anew. For this purpose select in the project tree the affected SIMOTION D Control Unit, the SINAMICS CU or Controller Extension CX and open the "FastIO" > "Create new configuration" shortcut menu with the right mouse button. Then compile the project and load it into the CPU. Perform a restart.

The FastIO configuration is also used for the telegram of the Terminal Modules TM15 and TM17 High Feature. Proceed in the same way if problems occur.

Linking an additional encoder (optional)

Basic principles

Using encoders

In addition to a motor encoder, additional encoders can be linked and configured:

- Additional encoders on the drive
  - Encoders with DRIVE-CLiQ interface
  - Encoders connected to a CU310/CU310-2 or CUA32 via the onboard encoder interface
  - Encoders connected to SIMOTION D via an SMx module
- Additional encoders on SIMOTION D
  - Encoders connected via PROFIBUS
  - Encoders connected via PROFINET

Configuring additional encoders

The additional encoder can be used in SIMOTION, for example, as:

- Machine encoder (second encoder = direct measuring system)
  A direct measuring system measures the technological parameter directly, i.e. without the interference of influences such as torsion, backlash, slip, etc. This can facilitate improved smoothing of mechanical influences by means of the closed-loop control. If you use a second encoder as a machine encoder, you can work with the encoder changeover function.
- External encoder
  You can use the external encoder to record an external master value, for example.
- Encoder for hydraulic axes
- Encoders for the implementation of cam controllers

Additional encoders on the drive

The following options are available for configuring additional encoders on the drive:

- Configuration of a second encoder on the drive
- Configuration of an encoder as drive object (as of SINAMICS firmware V4.3)

The following options are available for configuring additional encoders on the drive.

Second encoder on the drive

The configuration of a second encoder on the drive is useful when the second encoder value is also to be used for this drive (e.g. motor or machine encoder). Note that only a maximum of two encoder values can be transferred via PROFIdrive message frames.

In principle, the second encoder value can be freely used (e.g. for acquisition of an external master value), however the use of an encoder as a separate drive object (drive object DO encoder) is preferable because of the clear functional separation.

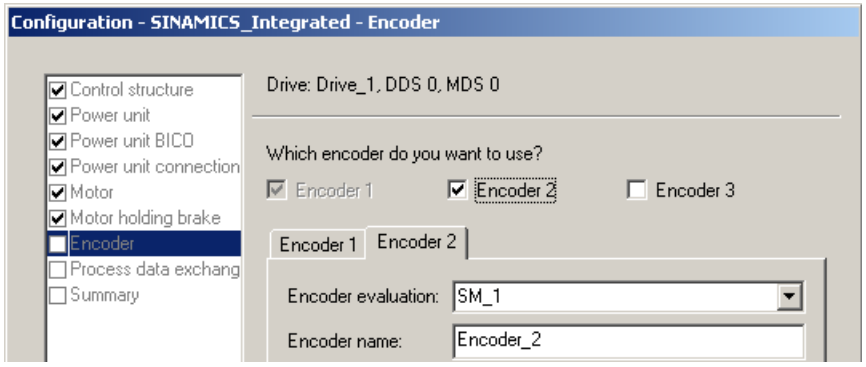


Figure 10-101 Configuration of a second encoder on the drive

Encoder as drive object

The configuration of an encoder as drive object (drive object DO encoder) has the advantage that this encoder can be used independently of a configured drive (e.g. for acquisition of a master value).

The configuration is performed by inserting an encoder via the project navigator.

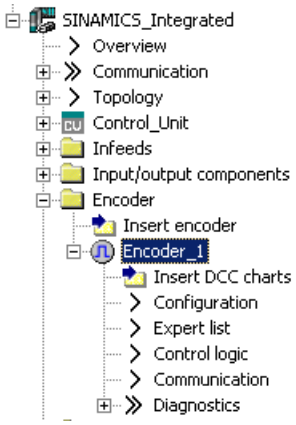


Figure 10-102 Configuration of an encoder as drive object

Note

In the same way as for axes, a "DO encoder" can also be symbolically interconnected to a "TO externalEncoder".

Additional encoders via PROFIBUS/PROFINET

Additional encoders can be connected via PROFIBUS or PROFINET. The following options are available for the encoder integration.

- Encoder interconnection using a PROFIdrive message frame (encoder with message frame type 81 and 83)
- Encoder interface as a direct value in the I/O area.

Additional references

Detailed information is contained in the
SIMOTION TO Axis, Electric/Hydraulic, External Encoder Function Manual

Symbolic assignment of I/O variables (PROFIdrive message frame / drive parameters)

Symbolic assignment of I/O variables to the PROFIdrive message frame of the TO axis

You can assign I/O variables from the address list which you require for display and diagnostic purposes, for example, to individual components (status word, for example) of the PROFIdrive message frame using the assignment dialog. Only components suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, this is determined via the assignment partner after the selection.

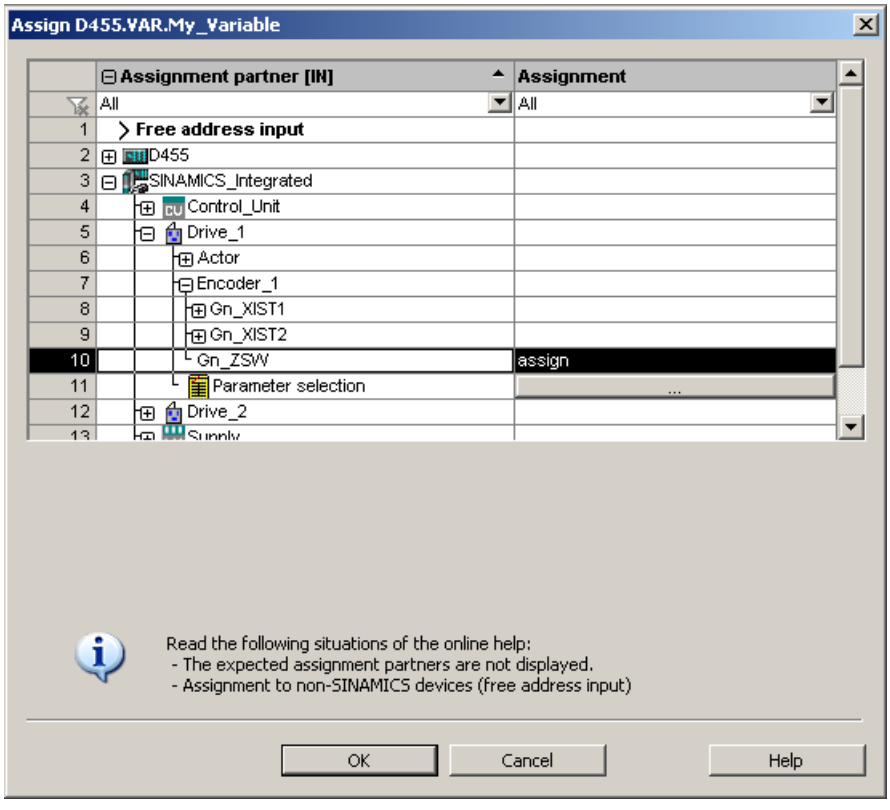


Figure 10-103 Assignment of I/O variables to the PROFIdrive message frame

Symbolic assignment of I/O variables to drive parameters

I/O variables from the address list can be assigned to drive parameters using the assignment dialog. Only parameters suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, this is determined by the parameter selection.

An extension of the standard message frame is created automatically for the transfer of the parameters to/from the drive.

Procedure

1. Open the assignment dialog box from the address list (view of all addresses). The assignment dialog opens with the corresponding assignment partners.
2. Click the ... button in the parameter selection line to open the parameter list.

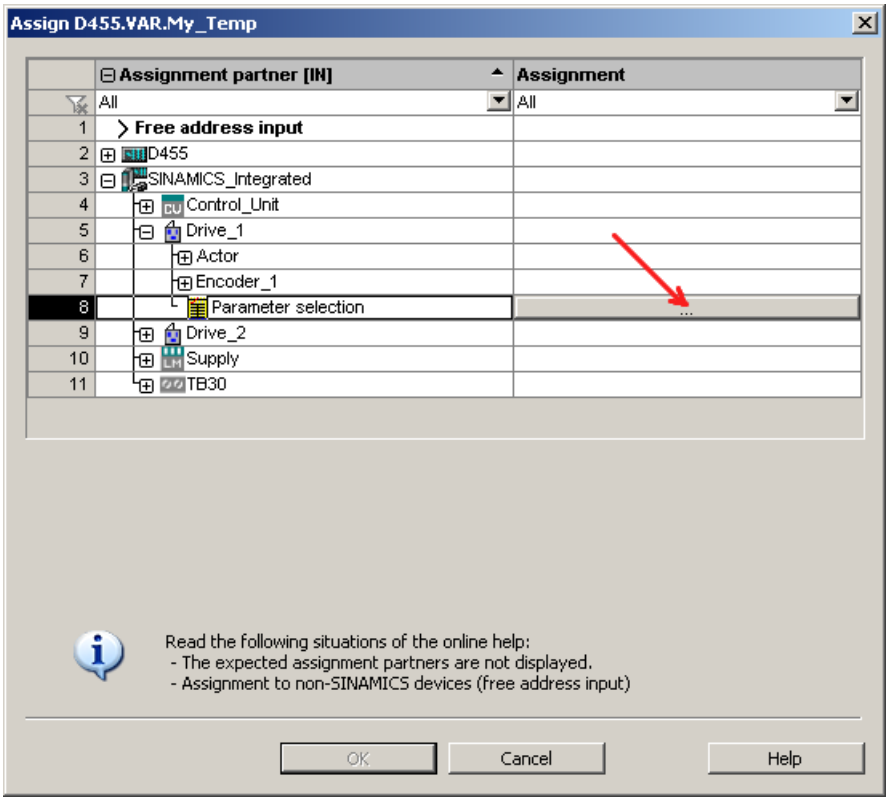


Figure 10-104 Assignment dialog for drive parameters

3. Select the desired signal source (e.g. DO drive). Then select the required parameter.

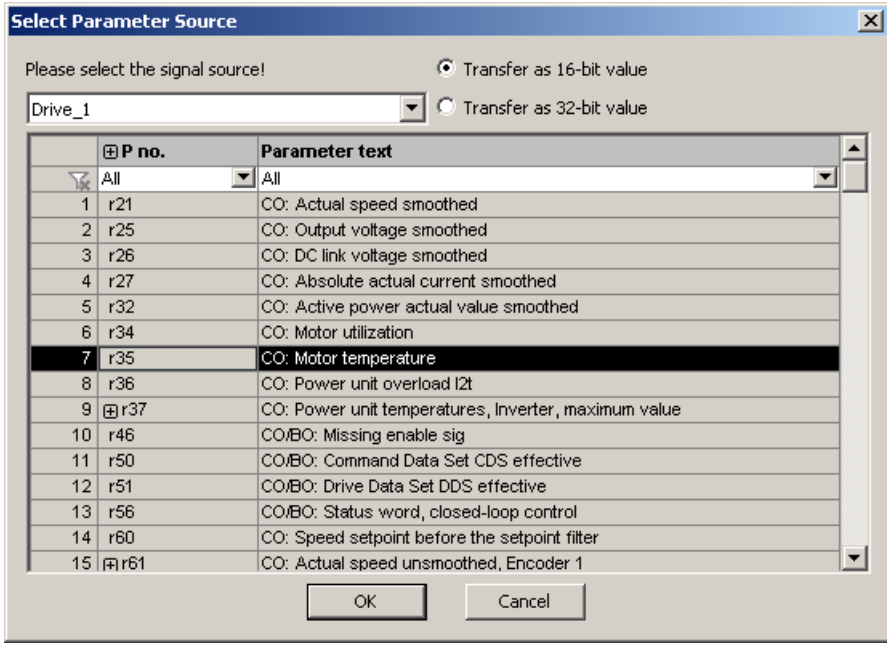


Figure 10-105 Dialog box for the DO and parameter selection

4. Click "OK" to accept the selection.

5. The desired SINAMICS parameter is assigned to the I/O variable in the interconnection dialog box.

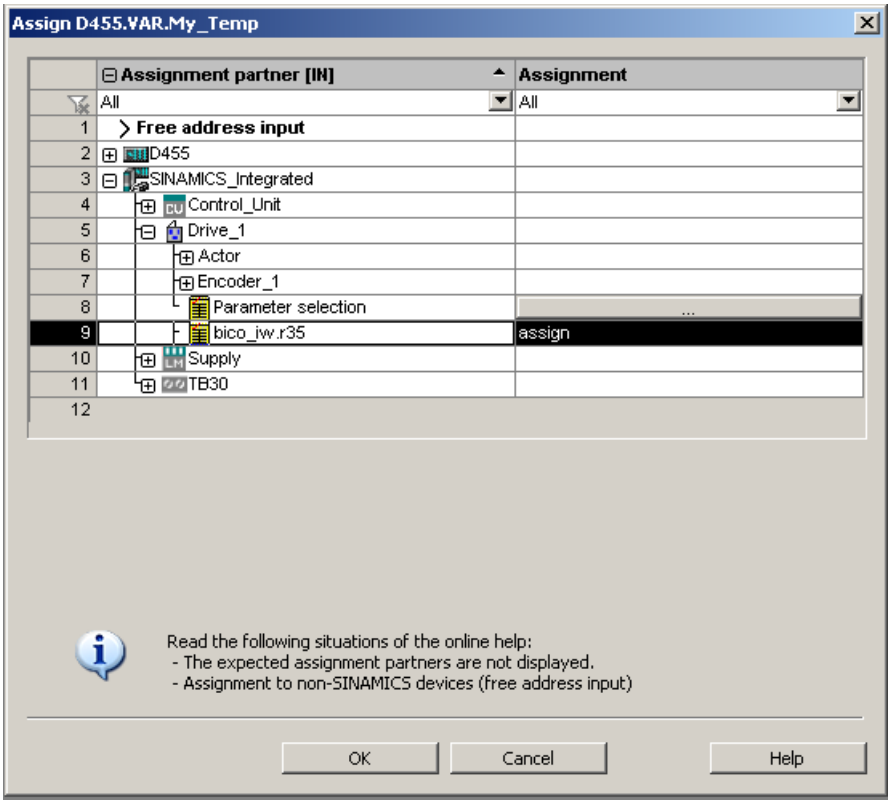


Figure 10-106 Assigned drive parameters

6. Click "OK" to accept the assignment.

The following table shows the possible types of assignment:

Name of the assignment	Data type	Direction	Transferrable BICO parameters
BICO_IW.<parameter number>	WORD	Input	All CO parameters (BICO source)
BICO_QW.<parameter number>	WORD	Output	All CI parameters (BICO sink)
BICO_ID.<parameter number>	DWORD	Input	All CO parameters (BICO source)
BICO_QD.<parameter number>	DWORD	Output	All CI parameters (BICO sink)

Syntax of the assignment names

- A number of parameters (separated by periods) are specified for outputs (SINAMICS side = received data) which can be interconnected with a number of BICO sinks.
- If the transferred parameter is on another drive object (DO), the DO name precedes the parameter. "#" is used as a separator between the DO name and the parameter.
- Individually transferred bits of a parameter appear in brackets [x].

Configuring drive-related I/Os (with symbolic assignment)

Overview

SIMOTION D4x5-2, the CX32-2 controller extension as well as the SINAMICS S110/S120 control units and additional components (TB30, TMs) have I/Os that can be used by the drive unit and SIMOTION.

So that I/Os, which were originally assigned to SINAMICS, can be used by SIMOTION, they must be interconnected to a message frame.

Symbolic assignment

As of version V4.2, SIMOTION SCOUT supports the symbolic configuration of I/Os. "Use symbolic assignment" must be activated for this purpose (see also Section Symbolic assignment / adaptation (Page 7046)).

The symbolic assignment simplifies the configuration significantly:

Table 10-52 Comparison of configuration with/without symbolic assignment

	With symbolic assignment	**Without symbolic assignment**
Configuring message frames	So that SIMOTION can use SINAMICS I/Os, the required message frames are created automatically	Message frames must be set manually (either predefined message frame (e.g. 39x) or free message frame configuration)
BICO interconnections	The required BICO interconnections are made automatically (I/Os are interconnected to message frame)	With predefined message frames (e.g. 39x), the BICO interconnections are made automatically With free message frame configuration with BICO, the interconnection must be made by the user
Parameterization of the I/O functionality (e.g. measuring input)	Parameterization via screen forms	Parameterization via screen forms and partly via parameters in the expert list
Handling of I/O addresses	Handling of addresses is not required because of symbolic assignment	I/O addresses must be determined
Setting up addresses	Addresses are set up automatically, see also Section Setting up addresses (Page 7147)	Addresses must be set up manually, see also Section Message frame configuration (Page 7148)

Only the configuration with symbolic assignment is described in the following. For further information on the configuration of drive-related I/Os without symbolic assignment, see Configuration of drive-related I/Os (without symbolic assignment) (Page 7280).

Procedure

The configuration of the I/Os is divided into two basic steps:

1. Configuration of the I/O terminals (Page 7159). The functionality of an I/O channel is configured (e.g. configuration of a DI/DO as digital output).
2. Configuration of the technology objects and I/O variables. (Page 7163)
   The access of technology objects and I/O variables to I/Os is configured.
   The configuration is performed symbolically, whereby only "function-compatible" I/O channels are offered for selection.
   Example:
   For the TO measuringInput, only symbolic assignments of the type MI (measuring input) are offered for selection.

The required message frames and the interconnection are then set automatically by the engineering system.

Configuration of the I/O terminals

The following table provides an overview of the configuration options for the I/O terminals of various modules.

Table 10-53 Overview of the configuration of I/O terminals

Module	Use of the I/Os by		Configuration of the I/O terminals	Supports symbolic assignment
	SIMOTION	SINAMICS		
SIMOTION D4x5-2 • Terminal X122/X132 • Terminal X142	 X [1] X	 X -	 On the drive unit (CU) On the D4x5-2 (HW Config)	As of SIMOTION V4.2
SIMOTION D4x5	X [1]	X	On the drive unit (CU)	As of SIMOTION V4.2
CX32-2, CX32	X [1]	X	On the drive unit (CU)	As of SIMOTION V4.2
SINAMICS S110 CU305	X [1]	X	On the drive unit (CU)	As of SINAMICS V4.3
SINAMICS S120 • CU310 • CU310-2 • CU320 • CU320-2	X [1]	X	On the drive unit (CU)	 • As of SINAMICS V2.6.2 • As of SINAMICS V4.4 • As of SINAMICS V2.6.2 • As of SINAMICS V4.3
TB30, TM15 DI/DO, TM31	X [1]	X	On the drive unit (TB30 or TM)	Yes
TM41	X [1]	X	On the drive unit (TM41)	Yes [2]
TM15, TM17 High Feature	X	-	On the drive unit (TM15 or TM17)	Yes
Time-based I/O • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	X	-	On the ET 200 I/O system	Yes, as of SIMOTION V4.4 HF6

[1] I/Os are originally assigned to a SINAMICS drive unit and can be assigned to SIMOTION via configuration

[2] TM41 supports symbolic assignment only for encoder interfaces (not symbolic assignment for DI, DO and AI)

Note

The module hardware for TM15 and TM15 DI/DO is identical. A distinction is only made by the addition of the component in the SIMOTION SCOUT project navigator using "Insert input/output component".

I/Os that are originally assigned to the SINAMICS drive unit can also be used by SIMOTION via configuration:

- An output is always only exclusively available for the SINAMICS drive unit or SIMOTION.
- An input used by SIMOTION can also be interconnected on the drive side.

The configuration of the I/O terminals is described in detail in the following:

- Configuration of the D4x5-2 I/Os (terminal X122/X132) (Page 7160)
- Configuration of the D4x5-2 I/Os (terminal X142) (Page 7161)
- Configuration of the CX32-2/CU3xx/TB30/TMxx I/O terminals (Page 7162)

Configuration of the D4x5-2 I/Os (terminal X122/X132)

Procedure

The I/Os of terminals X122 and X132 are originally assigned to the drive unit. The configuration is therefore performed via the drive unit ("SINAMICS_Integrated" > "Control_Unit" > "Inputs/outputs).

Using the default settings as basis for further parameterizations, you can assign SIMOTION to all I/Os or SINAMICS to all I/Os.

The properties of the I/O channel can be configured in the parameterization dialog box. With the bidirectional digital I/Os, for example, an I/O channel can be:

- Parameterized as input or output.
- Inverted.
- BICO-interconnected (use as drive I/O).
- Used as digital input for SIMOTION with "DI (SIMOTION)".

- Used as digital output for SIMOTION with "DO (SIMOTION)".
- Used as a global measuring input input for SIMOTION with "Measuring input (SIMOTION)".

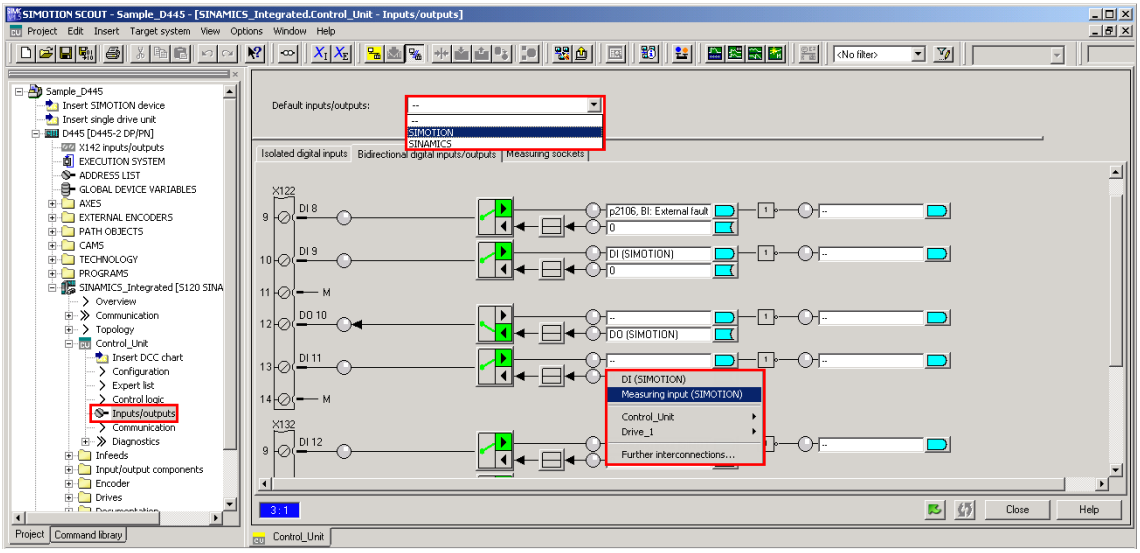


Figure 10-107 Configuration of the D4x5-2 I/Os (terminal X122/X132)

Note

The properties of the I/O channel must be configured offline

Online changes only take effect after a restart (power off/on), provided that the setting has been previously saved with "Copy RAM to ROM" power off.

Data transfer

If the X122 and X132 I/Os are interconnected using symbols (or if telegram 39x is used for the X122/X132 I/Os), the status information of the DI and DO will be transmitted to cu.p2048 at the PROFIdrive PZD sampling rate. Sampling of the inputs and outputs is also performed in the sampling time parameterized according to cu.p0799.

The same applies if the I/Os are manually interconnected to a drive telegram via a BICO converter.

Transfer of the output values and feedback of the input values is therefore subject to dead times and jitter.

For time-critical applications, use of measuring probes or cams is recommended. Alternately, the isosynchronous X142 onboard I/Os, I/Os of TM15, TM17, or isosynchronous ET 200 I/Os can be used.

Configuration of the D4x5-2 I/Os (terminal X142)

Procedure

The I/Os of terminal X142 are permanently assigned to the SIMOTION D4x5-2. The configuration is therefore performed via the D4x5-2 ("D4x5-2" > "X142 inputs/outputs).

Alternatively, the dialog box can also be called in HW Config by double-clicking the X142 interface.

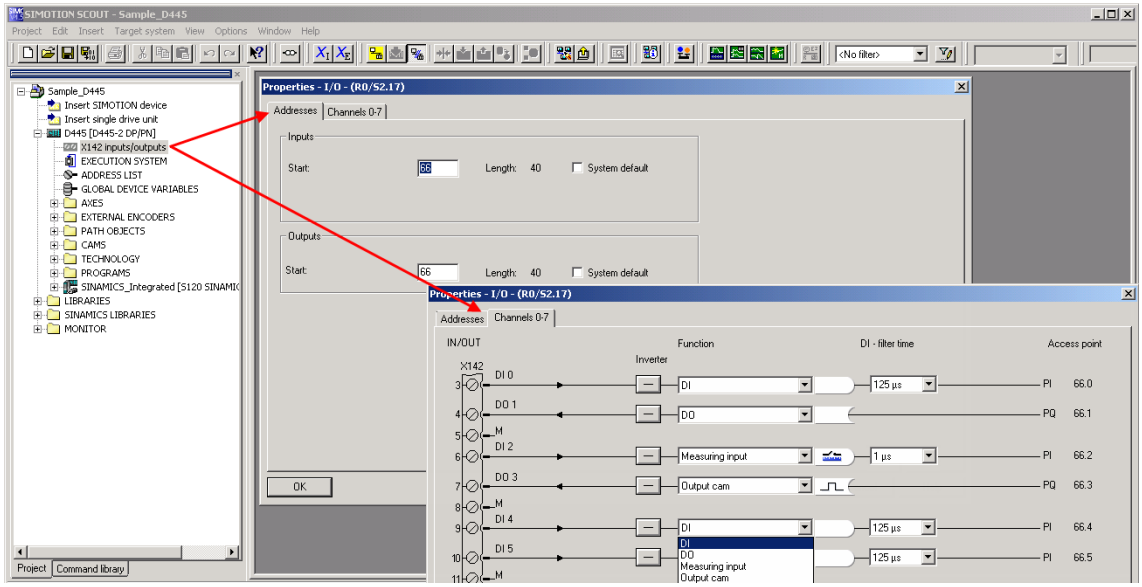


Figure 10-108 Configuration of the D4x5-2 I/Os (terminal X142)

Addresses tab

The start address for the X142 I/Os can be set in the "Addresses" tab. The address can be specified or automatically assigned by the system.

Channel tab

The function of each individual I/O channel is set in the "Channel 0-7" tab. Possible functions:

- DI
- DO
- Output cam
- Measuring input

All I/O channels can be inverted. A filter time of 1 µs or 125 µs can also be set for digital inputs and measuring inputs.

The displayed logical address of a channel is only required when symbolic assignments are not used.

Configuration of the CX32-2/CU3xx/TB30/TMxx I/O terminals

The configuration is performed in a similar way as for the onboard I/Os X122/X132 for the SIMOTION D4x5-2, i.e. the I/Os can be

- BICO-interconnected (use as drive I/O)
- Used by SIMOTION

See also Section Configuration of the D4x5-2 I/Os (terminal X122/X132) (Page 7160)

Note

If symbolic assignment is activated retrospectively for a project in which message frames have already been configured and interconnected, these can be changed together with the BICO interconnections.

For this reason, make a backup copy of your project before activating the symbolic assignment.

TB30, TM15 DI/DO and TM31 are especially affected.

See also Section Symbolic assignment / adaptation (Page 7046)

Configuration of the ET 200SP/MP timer DIDQ

The I/O terminals of the ET 200SP and ET 200MP timer DIDQ can be used as drive-related I/Os for measuring input and output cam applications.

The DI/DQ timers are configured via the hardware configuration of STEP 7 or TIA Portal. The properties of the I/O channel can be configured in the parameterization dialog box, for example

- Timer DI for use of the I/O as measuring input input or
- Timer DQ for use of the I/O as cam output

Note

For the technological use of the timer DIDQ, you must operate the I/O station isochronously on PROFINET:

- ET 200SP timer DIDQ: With SCOUT or SCOUT TIA, IM 155-6 PN HF required
- ET 200MP timer DIDQ: Only with SCOUT TIA, IM 155-5 PN ST or HF required

Operation via PROFIBUS is not possible.

For detailed information on the configuration, see *Technology Modules TM Timer DIDQ for SIMOTION SCOUT* and *SIMOTION SCOUT TIA* Commissioning Manual

Configuration of the technology objects and I/O variables

Configuration of global measuring inputs

Overview

The type of measuring input must be selected during the configuration of the TO measuringInput.

The following measuring input types are available:

Table 10-54 Measuring input types

Measuring input types	Explanation
Standard (global measuring input)	Compared with the drive-related local measuring inputs, global measuring inputs have extended functionality and also support a symbolic configuration. They are therefore set as standard.
Drive-related (local measuring input)	The drive-related local measuring inputs are configured via drive parameters, see Section Configuration of drive-related I/Os (without symbolic assignment) (Page 7280) (in the appendix).
Listening measuring input	Through the configuration of a listening measuring input, measuring can be performed simultaneously on several axes or external encoders with one measuring input. Detailed information can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs* Function Manual.

A detailed comparison of "local" and "global" measuring inputs as well as an overview of which modules support local or global measuring inputs can be found in the appendix in Section Configuration of drive-related I/Os (without symbolic assignment) (Page 7280).

Procedure

If a global measuring input is selected, it must be assigned a hardware input.

To do this, open the assignment dialog box via ... and select a free (i.e. not yet used) I/O.

Note

Only those I/Os are displayed that have the appropriate measuring input functionality (MI_xx [channel name, terminal number]). If no suitable I/Os are displayed, you must first configure the I/Os (I/O must be configured as "measuring input")

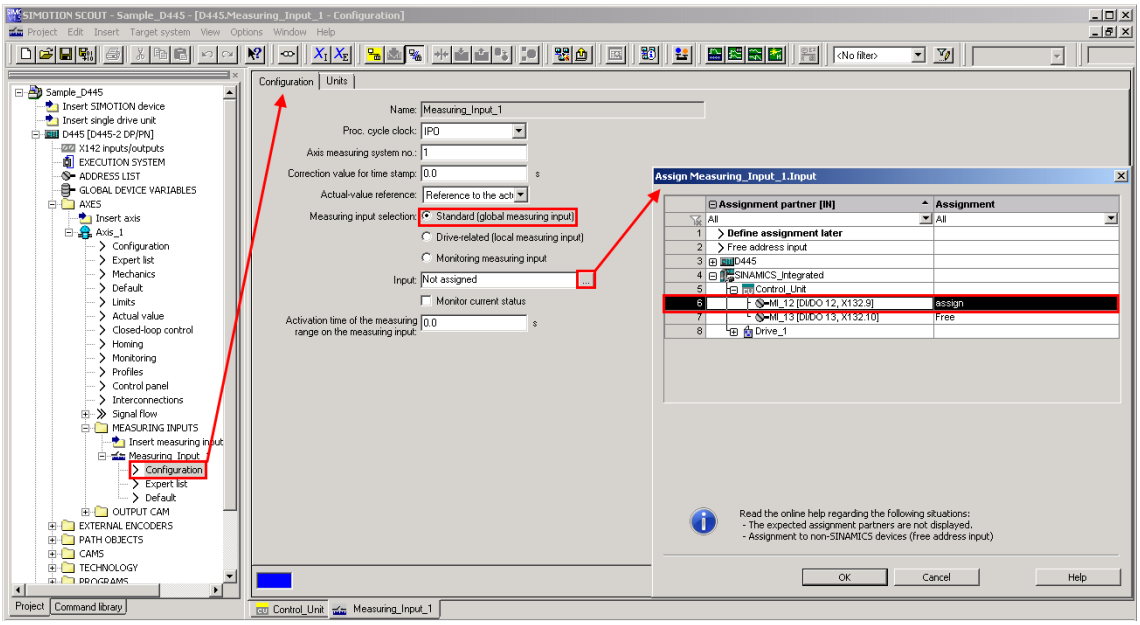


Figure 10-109 Configuration of a global measuring input for the D4x5-2

Detailed information on the configuration of the TO measuringInput can be found in the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Configuration of local measuring inputs

Local measuring inputs are drive-related measuring inputs. The configuration is performed via drive parameters.

For further details, see:

- Section Configuration of drive-related I/Os (without symbolic assignment) (Page 7280) in the appendix
- *SIMOTION Output Cams and Measuring Inputs* Function Manual

Configuration of output cams / cam tracks

Overview

The type of cam output must be selected during the configuration of the TO outputCam and TO camTrack.

The following output types are available:

Table 10-55 TO outputCam / TO camTrack output types

Cam output on ...	Explanation
Output cam output (CAM)	The cam output is performed on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. Supported hardware: • SIMOTION D4x5-2 (terminal X142): Resolution 1 µs • TM17 High Feature: Resolution 1 µs • TM15: Typical resolution 125 µs (DRIVE-CLiQ cycle clock) • D410-2 (DI/DO 8 to 15): Typical resolution 125 µs • ET 200MP TM timer DIDQ 16x24V: Resolution 1 µs • ET 200SP TM timer DIDQ 10x24V: Resolution 1 µs
High-speed digital output (DO)	The cam output is performed via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle. Supported hardware: • SIMOTION D4x5 (terminal X122, X132) • SIMOTION D410 (terminal X121) • SIMOTION C240, C240 PN (terminal X1)
Standard digital output (DO)	The output cam calculations are performed in the processing cycle clock (IPO or IPO2 cycle clock or in the servo cycle clock). Actual cam output is performed in the servo cycle clock. The resolution with respect to time of the cam output is generally reduced by the output cycle of the I/O used. The resolution is therefore dependent as follows: • For the standard I/O (e.g. ET 200), on the cycle time of the bus system (PROFIBUS DP / PROFINET I/O) • For the TM15 / TM17, on the cycle time of the bus system (PROFIBUS Integrated / PROFIBUS DP / PROFINET IO) • For the TM15 DI/DO, TM31, TM41, TB30, on the configured sampling time: – cu.p0799 (CU inputs/outputs sampling time) for the onboard outputs – p4099 (TMxx inputs/outputs sampling time) for TB30, TM15 DI/DO, TM31, and TM41 Supported hardware: • Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx) • Centralized I/O (SIMOTION C) • Distributed I/O via PROFIBUS DP/PROFINET I/O (e.g. ET 200, ...) • Drive-related I/O (TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30)

Note

The deployment of TM15 (SIMOTION) and TM17 (SIMOTION) is possible only with DP Integrated, PN and servo cycle clocks ≥ 0.5 ms. This applies to TM modules connected directly to D4x5-2 and indirectly via CX32-2/CU320-2.

Procedure

To achieve the best possible output cam resolution on the onboard I/Os of a SIMOTION D4x5-2, activate the output and select "Cam output on output cam output (CAM)."

Then assign a hardware output. To do this, open the Assignment dialog box via ... and select a free (i.e. not yet used) I/O.

Note

Only those I/Os are displayed that have the appropriate output cam functionality (CAM_xx [channel name, terminal number]). If no suitable I/Os are displayed, you must first configure the I/Os (I/O must be configured as "output cam (CAM)")

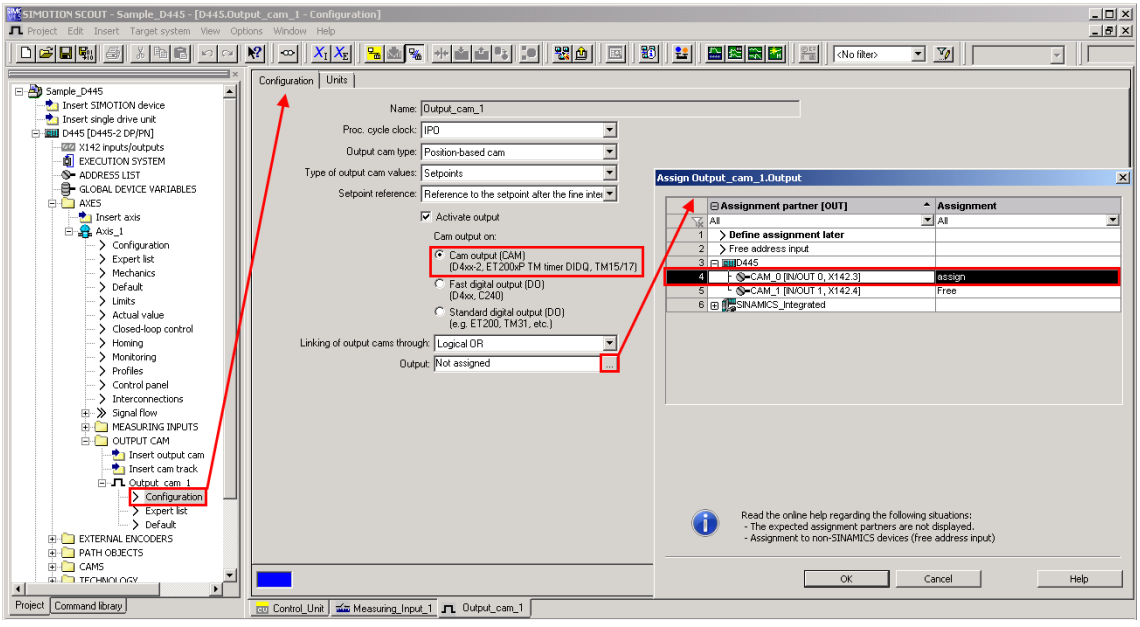


Figure 10-110 Configuration of an output cam for the D4x5-2

A maximum of 2 edges can be output per processing cycle clock of TO outputCam or TO camTrack.

Detailed information on configuring the output cam and cam track technology objects can be found in the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Supplementary interconnection options for X142 I/Os

Supplementary interconnection options for X142 I/Os

If the X142 I/Os are symbolically interconnected, further interconnection options are offered for selection under DI, DO, MI, and CAM, depending on the chosen data type.

With the exception of ACTVAL and LEC, these interconnection options are only intended for diagnostics by the Siemens Hotline.

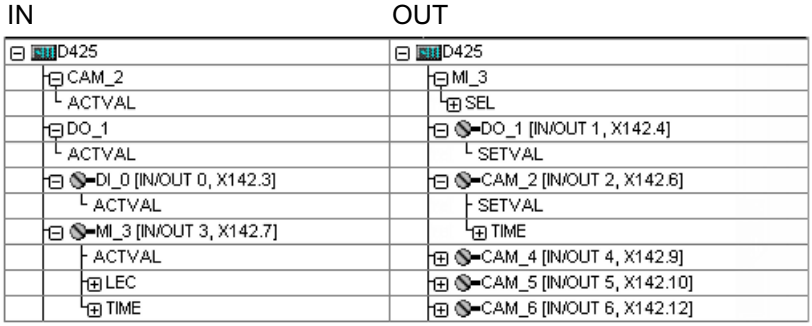


Figure 10-111 Supplementary interconnection options for X142 I/Os

ACTVAL, terminal status

ACTVAL represents the logical channel status, considering an configured inversion.

- in the case of a DI, ACTVAL corresponds to the logic state of the DI
- in the case of an MI, ACTVAL corresponds to the logic state of the measuring input
- in the case of a DO / CAM, ACTVAL corresponds to the actual terminal status of the digital output or output cam output. If the terminal status differs from the controlled status, there may be a short-circuit or fault of the output driver.

LEC, lost edge counter

The LEC (Lost Edge Counter) counts the lost edges at a measuring input input. Lost edges are edges that must be measured according to the measurement job, i.e.

- the measurement job is active and
- the edge corresponds to the edge selection

For example, if only rising edges are to be acquired according to the measurement job, only lost "rising" edges will be acquired.

Note

The precondition for the use of the LEC is that "servo" is set as the processing cycle clock for the measuring input TO.

If the capacity of the measuring input input (2 edges per servo cycle or per measurement job) is exceeded, the edges will be lost.

Up to 7 lost edges

- can be acquired per servo cycle (cyclic measurement) or
- per measurement job (single measurement)

.

If more than 7 edges are lost, the counter will stop at the value 7.

Therefore: if the value is 7, 7 or more edges have been lost.

Lost edges can indciate that measurement is faulty (contact bounce, disturbances, etc.).

Example 1:

- Measuring input input is only to measure positive edges.
- 6 negative and 5 positive edges occur in cycle n.
- 2 positive edges are measured

Result: The number of lost positive edges is 3.

Example 2:

- Measuring input input is to measure any edges.
- 11 edges occur in cycle n.
- 2 edges are measured

Result: 7 lost edges are signaled; 2 further edges cannot be recorded due to the limitation of the counter to 7 in the module.

Evaluation in the user program

In eveluation of the LEC, ensure that the edge counter only fills 3 bits of the LEC byte in each case. The non-relevant bits have to be masked in this way.

Lost edge counter	**Relevant bits in the LEC byte**							
	7	6	5	4	3	2	1	0
for I/O channel 1 / 3 / 5 / 7		x	x	x				
for I/O channel 0 / 2 / 4 / 6						x	x	x

Program example

```
// lec_1 : IO variable in the SCOUT,  symbolically interconnected
with LEC measuring input 1 (type byte)
// lec_2 : IO variable in the SCOUT, symbolically interconnected with
LEC measuring input 2 (type byte)


VAR
   LecMeasuringInput1 : BYTE;
   LecMeasuringInput2 : BYTE;
END_VAR


LecMeasuringInput1:= (lec_1 AND 16#07);           //  even channel
LecMeasuringInput2:= SHR((lec_2 AND 16#70),4);    //  odd channel
```

Configuration of the I/Os (variables / TO axis)

You have two ways to assign an I/O variable to I/O terminals:

- Assignment via preferred interconnection (e.g. DI_0 [DI 0, X122.1])
  To do this, you must use the SIMOTION preferred interconnection for the corresponding input/outputs of the SINAMICS DOs. The BICO interconnection is performed automatically.
- Assignment via PZD (e.g. via DI_0_15 or DO_0_15).
  Note that for these signals a message frame of the appropriate length is generated, but the BICO interconnection is not performed.

Interconnection via preferred interconnection

The I/O variables are configured via the address list. Components that support a symbolic assignment can be configured without I/O addresses.

Preferred interconnections are displayed as assignment targets in the Assignment dialog box (e.g. DI_0 [DI 0, X122.1]). The assignment is made by direct selection of the corresponding terminal signal.

Components that do not support symbolic assignment (e.g. standard PROFIBUS I/O) are configured via I/O addresses.

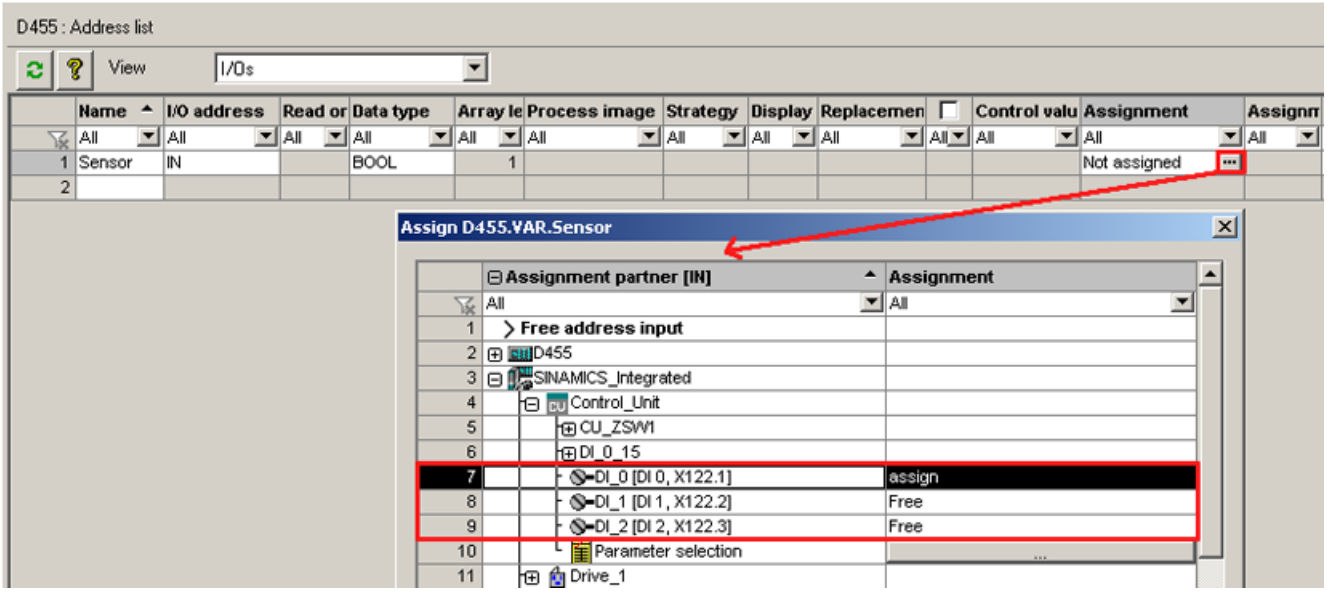


Figure 10-112 Address list

Interconnection via PZD

In principle, an assignment via the PZD is also possible (e.g. via DI_0_15 or DO_0_15). Note that for these signals a message frame of the appropriate length is generated, but the BICO interconnection is not performed.

For this purpose, switch to the "Communication" dialog box of the corresponding SINAMICS DO. You can now see the individual bits of the PZD (e.g. I_Digital or O_Digital) listed there. Interconnect the appropriate bit of the PZD with a signal.

Alternatively, you can assign SIMOTION to an I/O channel during the terminal configuration (e.g. by selecting "DI (SIMOTION)", see Section Configuration of the D4x5-2 I/Os (terminal X122/ X132) (Page 7160)).

TO axis

The symbolic assignment of I/Os is also supported by the TO axis (e.g. for a HW limit switch).

Substitute values for I/O variables

Substitute values cannot be specified for input variables of the BOOL data type. If you do require substitute values however, proceed as follows:

1. Assign a digital input (e.g. SINAMICS_Integrated.Control_Unit.DI_0 [DI 0, X122.1]) to an input variable of the BOOL type (e.g. sensor).
2. Create a global variable (e.g. all_inputs) (at least data type WORD, e.g. SINAMICS_Integrated.Control_Unit.DI_0_15).
3. Configure the substitute value.
   The appropriate bit of the substitute value must then contain the substitute value for the BOOL variable.
   In the same way, you can assign a substitute value to a BICO parameter.

For various SINAMICS drive objects, higher-level types are available for the assignment of substitute values.

D455 : Address list

View: I/Os

	Name	I/O address	Read only	Data type	Array leng	Process image	Strategy	Display	Substitute v		Control valu	Assignment	Assignment
	All	All	All	All	All	All	All	All	All	All	All	All	All
1	all_inputs	IN		WORD	1		Substitute value	HEX	16#00_00			SINAMICS_Integrated.Control_Unit.DI_0_15	4: Set up
2	sensor	IN		BOOL	1							SINAMICS_Integrated.Control_Unit.DI_0 [DI 0, X122.1]	4: Set up
3													

Figure 10-113 Configuration of substitute values

DMC20/DME20 DRIVE-CLiQ hub

Hub properties

DRIVE-CLiQ hub characteristics

The DMC20 and DME20 DRIVE-CLiQ hub modules are used to implement point-to-point distribution of a DRIVE-CLiQ line. With the DMC20/DME20, an axis grouping can be expanded with 4 DRIVE-CLiQ sockets for additional subgroups.

- DMC20 is the hub for the control cabinet configuration
- DME20 is the hub for use without a control cabinet (IP67 degree of protection).

The modules are especially suitable for applications which require DRIVE-CLiQ nodes to be removed in groups, without interrupting the DRIVE-CLiQ line and therefore the data exchange.

Application examples

Typical applications of DRIVE-CLiQ hubs are encoder expansion and hot plugging.

- In an encoder expansion, direct measuring systems are connected. For example, these are attached directly to the machine in the control cabinet. Several encoders can be connected to one hub in the cabinet.
- Hot plugging is the option for changing Motor Modules while in operation. To do so, the Motor Modules are connected via a DRIVE-CLiQ hub in the form of a point-to-point topology. This means they can be deactivated without impairing downstream components (via cascading).

Note

Cascading is permitted one time only (from hub to hub).

Connection conditions for ENCODER drive objects

- All encoders that can be assigned to a drive can be used.
- ENCODER drive objects may be connected to all DRIVE-CLiQ ports.
- Up to 4 DRIVE-CLiQ HUBs (DMC20 or DME20) can be used to establish a star-shaped wiring of the ENCODER drive objects. This means that a maximum of 19 possible ENCODER drive objects can be connected to one Control Unit. This means that the number of possible ENCODER drive objects is restricted so that a total maximum of 24 drive objects can be connected to one Control Unit.
- The DRIVE-CLiQ HUBs must be directly connected to the Control Unit.

Additional references

Additional information on the DMC20/DME20 DRIVE-CLiQ is contained in the

- *SIMOTION D4x5-2* Manual
- *SINAMICS S120 Control Units and Additional System Components* Manual

Creating a DMC20/DME20 DRIVE-CLiQ hub

Introduction

A DMC20/DME20 can be inserted directly in the project navigator. The hub is not wired when you insert the DMC20/DME20 and is displayed in the topology tree in the component storage. The hub has to be wired manually.

Procedure

1. Right-click "Topology" in the project navigator.
2. Select the "Insert new object" > "DRIVE-CLiQ hub" command from the context menu and confirm with "OK".

3. Double-click "Topology" to display the topology tree.
   In the topology tree, the hub is stored in the component archive.
4. Drag-and-drop the hub to the required DRIVE-CLiQ interface.
   The components connected to the hub are displayed in the topology tree.

Result

The added hub is displayed for the "Topology" entry in the project navigator. All components connected to a hub are also displayed during an automatic configuration.

TM41 terminal module

Overview

The TM41 terminal module can be used to expand the number of digital I/O and of analog inputs within a drive system. TM41 also returns TTL signals which emulate an incremental encoder, for example, for a master control system.

The emulated encoder signal has the signal characteristic of an incremental TTL encoder (A track, B track, R track). The resolution of the encoder signal can be specified in the configuration.

Note

The digital inputs/outputs and the analog input can be interconnected via BICO configuration.

The TM41 encoder interface (incremental encoder representation) can

- Be interconnected with an encoder signal of the control unit by means of parameterization, e.g. sin/cos incremental encoders. For detailed information, see the SINAMICS manuals.
- From the SIMOTION viewpoint, be accessed as axis. This allows you to return the axis position (a master value) as an encoder signal to a second controller, for example. The configuration can be performed symbolically and is described in the following.

  Note

  TM41 only supports the symbolic assignment for the encoder interface (no symbolic assignment for the DI, DO and AI).

Configuring the TM41 involves the following steps:

- Configuring the TM41 at SINAMICS Integrated
- Configuring the TM41 using the axis wizard

Configuring TM41 at SINAMICS Integrated

The TM41 can be configured after you completed configuration of the SINAMICS Integrated. Proceed as follows:

1. Double-click "Insert input/output component" at "Input/output component" in the project navigator.
2. Select TM41 from the "Drive object type" field of the "Insert Input/Output Component" dialog box and assign a unique name to the module.
3. Confirm your entry with "OK".
   The TM41 is inserted in the project navigator by the name you entered.
4. Double-click "Inputs/outputs". The "Inputs/Outputs" property dialog box of TM41 opens.
5. Set "SIMOTION" as the operating mode on the "Pulse encoder emulation" tab, and make the settings for the emulated sensor signal.
6. Double-click "Configuration" under "SINAMICS_Integrated" in the project navigator and select the "PROFIdrive telegram" tab.
7. Click "Close" to exit the dialog box.

You have now programmed the TM41 and aligned it with HW Config.

Configuring the TM41 using the axis wizard

After configuring the TM41 for a SINAMICS Integrated device in the project navigator, you can interconnect it with an axis using the Axis Wizard. The wizard implements the TM41 as a drive device.

1. Open the Axis Wizard and create a position or following axis (electrical).
2. Step the Axis Wizard forward until the "Drive Assignment" dialog box opens.

3. Select "SINAMICS_Integrated" as the drive device and "TM41" as the drive. TM41 operates as setpoint sink of the axis with this setup.

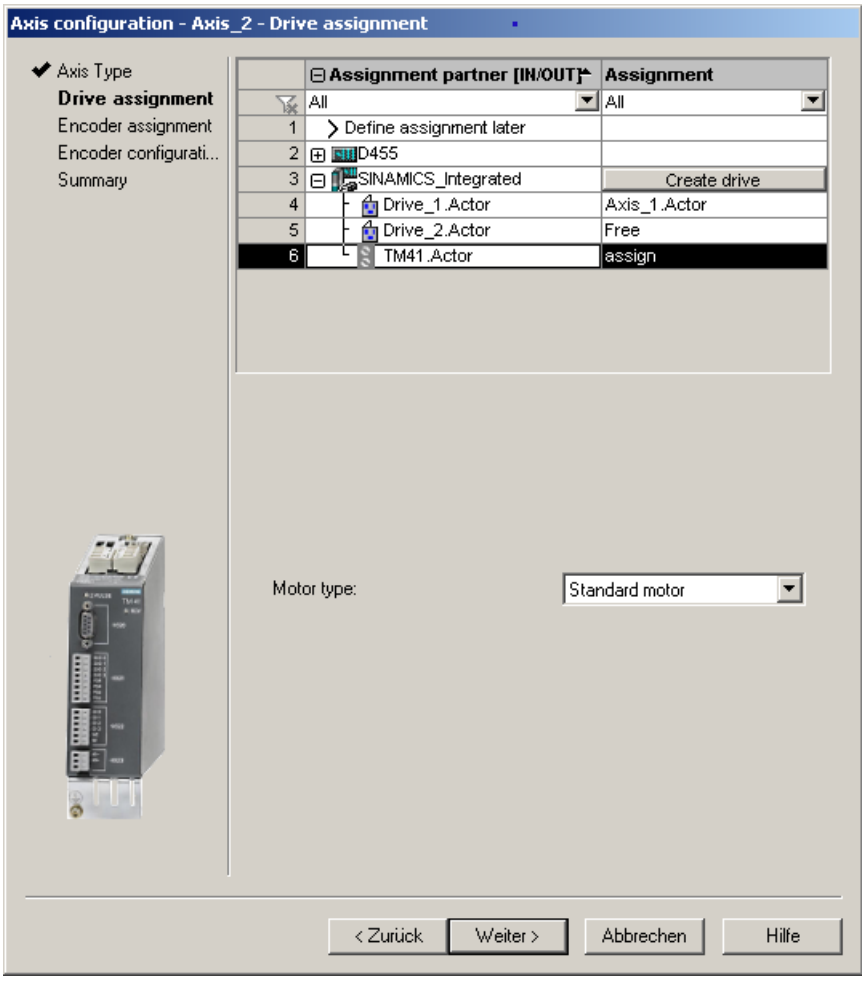


Figure 10-114 Drive assignment

4. Run through the wizard to the end.

Detailed information on configuring incremental encoder emulation with the TM41 can be found in:

- FAQs at: (https://support.industry.siemens.com/cs/ww/en/view/27554028)
- *SIMOTION Utilities & Applications*
  SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Optimizing the drive and controller

Overview of automatic controller setting

Overview

For the controller optimization of the drive, SIMOTION SCOUT offers a wizard for the automatic controller setting.

In the "Automatic Controller Setting" screen form, you can configure an automatic setting for the speed controller and the DSC (dynamic servo control) position controller for SINAMICS drive units. The necessary steps for this calculation can be controlled from this screen form. The parameter values calculated for the speed controller or position controller are displayed; these can then be transferred online to the drive or axis on the controller.
You can set the controller automatically using the menu command "Target system" > "Automatic controller setting".

For a detailed description of the parameters that can be defined, to the *SIMOTION SCOUT* Online Help.

Requirements

- You have configured a SINAMICS drive.
- The configured drive is operated in the "Servo" drive object type.
- Closed-loop control takes place with the motor encoder.
- There is an online connection to the relevant drive unit.

Procedure

Automatic controller setting involves the following steps:

1. Setting the speed controller
2. Setting the position controller

Note

You can cancel the automatic controller setting by pressing the SPACEBAR.

- The step currently being executed is aborted.
- The drive enable is canceled.

Additional references

Information on the controller structure can be found in the *SIMOTION TO Axis, Electric/ Hydraulic, External Encoder* Function Manual.

In addition to automatic controller setting, SIMOTION SCOUT also offers the option of optimizing the drive and controller manually by means of measuring functions, trace and function generator (see Sections Measuring functions, trace, and function generator (Page 7179) and Manual speed controller optimization (Page 7180)).

Automatic speed controller setting

Characteristics

The automatic speed controller setting has the following features:

- Attenuation of resonances in the speed-controlled system
- Automatic setting of the Kp gain factor and the Tn reset time of the speed controller
- The speed setpoint filter and the reference model are not changed.

Procedure

To perform an automatic setting of the speed controller, proceed as follows:

1. Select the "Target system" > "Automatic controller setting" menu command.
2. Select the drive unit and the drive.
3. Select the "Speed controller" from the "Controller Selection".
4. Click "Assume control priority" to assume control priority.
5. Click the "Drive on" button to enable the drive.
   Perform these steps (1 to 4) in automatic mode or as individual steps.
6. Click "Transfer" to transfer the calculated parameter values for the speed controller to the drive.
7. Disable the drive by clicking the "Drive Off" button.
8. Click "Give up control priority" to give up control priority of the PG/PC.
9. Save the online parameters.

You can now transfer the automatically set parameters to the project.

Backing up parameters

Proceed as follows to back up the parameters:

1. In the project navigator, select the SINAMICS unit with the drive for which you want to perform the automatic setting.
2. Select "Target device" > "Copy RAM to ROM" in the context menu.
3. Select "Target device" > "Load CPU / drive unit to PG" in the context menu.

If required, the automatic controller settings can be checked using the measuring functions.

Automatic position controller setting

Introduction

In the "Automatic Controller Setting" screen form, you can select the SINAMICS drive unit and the drive for which you want to carry out an automatic DSC position controller setting. The necessary steps for this calculation can be performed from this screen form. The calculated Kv value is displayed and can then be accepted online in the configuration data of the axis that is assigned to the drive.

Requirements

In addition to the General requirements for the automatic controller setting, the following boundary conditions apply for setting the position controller:

- DSC is required for the position controller setting.
  Tip:
  Activate the project setting "Use symbolic assignment" and select the Standard/Automatic option for the axis-drive communication when configuring the drives. You automatically use DSC for the servo drives with these settings.
- The speed controller has already been configured (e.g. with the automatic speed controller setting).
- At least one axis is connected to the SINAMICS drive (servo).
- An online connection to the SIMOTION device must be established to transfer the results of the automatic position controller setting.
- The balancing filter is not changed.
- For operation without precontrol, the equivalent time constant of the position controller must be adjusted manually by the user (PositionTimeConstant = 1/Kv).
- Vibration on the load side is not taken into account for the position controller setting.

Procedure

To perform an automatic setting of the position controller, proceed as follows:

1. Select the "Target system" > "Automatic controller setting" menu command.
2. Select the drive unit and the drive (axis).
3. Select the "Position controller (DSC)" from "Controller selection".
4. Click "Assume control priority" to assume control priority.
5. Click the "Drive on" button to enable the drive.
   Perform the steps either in automatic mode or as individual steps.
6. Select the axis data sets to which the Kv factor is to be transferred.
7. Click "Accept values" to transfer the calculated Kv factor to the axis data sets.
8. Disable the drive by clicking the "Drive off" button.

9. Give up the control priority of the PG/PC.
10. Save the online parameters.

You can now transfer the automatically set parameters to the project.

Backing up parameters

Proceed as follows to back up the parameters:

1. In the project navigator, select the SIMOTION unit with the axis for which you want to perform the automatic setting.
2. Select "Target device" > "Copy current data to RAM" in the context menu.
3. Select "Target device" > "Copy RAM to ROM" in the context menu.
4. Select "Target device" > "Load CPU / drive unit to PG" in the context menu.

If required, the automatic controller settings can be checked using the measuring functions.

Measuring functions, trace, and function generator

Drive optimization

Drive optimization is part of commissioning and can be performed with SIMOTION SCOUT.

Note

Controller optimization may only be performed by skilled personnel with control engineering knowledge.

Controller optimization

Various measuring functions are available for controller optimization of the drive. These measuring functions enable the control of the higher-level control loop to be selectively switched off and the dynamic response of individual drives to be analyzed through simple parameter assignment. The function generator and the trace recorder are used.

The control loop is supplied with the ramp-function generator signal at a specific point (e.g. speed setpoint), and the signal from the trace recorder is recorded at another point (e.g. speed actual value).

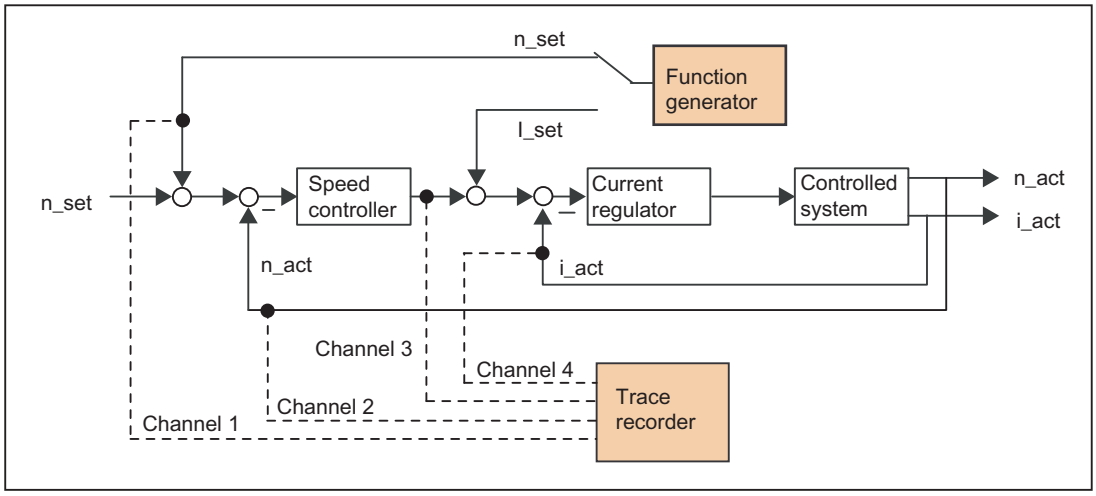


Figure 10-115 Optimizing a controller

Depending on the form of controller optimization to be performed, it is possible to define the quality (e.g. signal form, amplitude, transient recovery time) of the disabled signal, the measuring duration for step functions in the time range, or the bandwidth and number of averaging operations in the frequency range for the trace. The analytical and graphical evaluation can then be performed accordingly (FFT diagram, Bode diagram).

The following measuring functions are available:

- Setpoint jump at current controller
- Reference frequency response at current controller
- Setpoint jump at speed controller
- Disturbance variable jump at speed controller
- Reference frequency response at speed controller
- Disturbance frequency response at speed controller
- Speed-controlled system (input at current setpoint filter)

Additional references

For additional information about drive optimization, consult the *SINAMICS S120* Commissioning Manual.

Additional information on trace and measuring functions, as well as on the function generator, can be found in the *SIMOTION SCOUT Online Help*.

Manual speed controller optimization

Requirement

You have already created a project and configured an axis and a drive. You can now optimize the speed controller.

Procedure

1. Open the project and go to online mode.
2. Click  to call the "Measuring Functions" dialog box.
3. Select the drive unit and the drive.
4. Select "Speed controller setpoint jump".
   You can change the values in the following fields: "Settling time", "Amplitude", "Offset", "Ramp-up time" and "Measuring time".

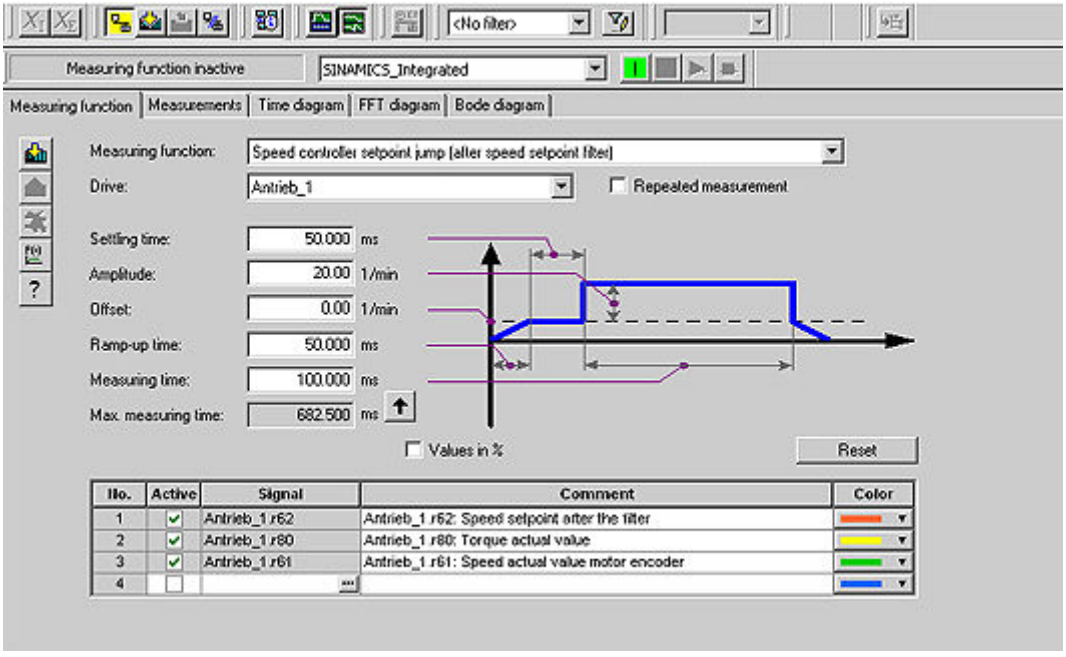


Figure 10-116 Speed controller measuring function

   Four channels can be traced. Certain channels are preassigned, depending on the measuring function.
5. Download the changes to the drive by clicking  (Download parameter assignment).

Starting the measuring function

1. Click "Assume control priority" to assume control priority.
   Read the notice that appears and click "Accept" to confirm. The activated Service function is displayed via the LEDs (RUN flashes green at 2 Hz and SU/PF flashes yellow at 2 Hz).
2. Click the "Drive on" button to enable the drive.

3. Click (Start measuring function) to start the measuring functions.
   The axis is moved during the measurement. For this reason, a safety message that allows the process to be aborted is displayed.
4. The traced signals are represented on the "Time diagram" tab.

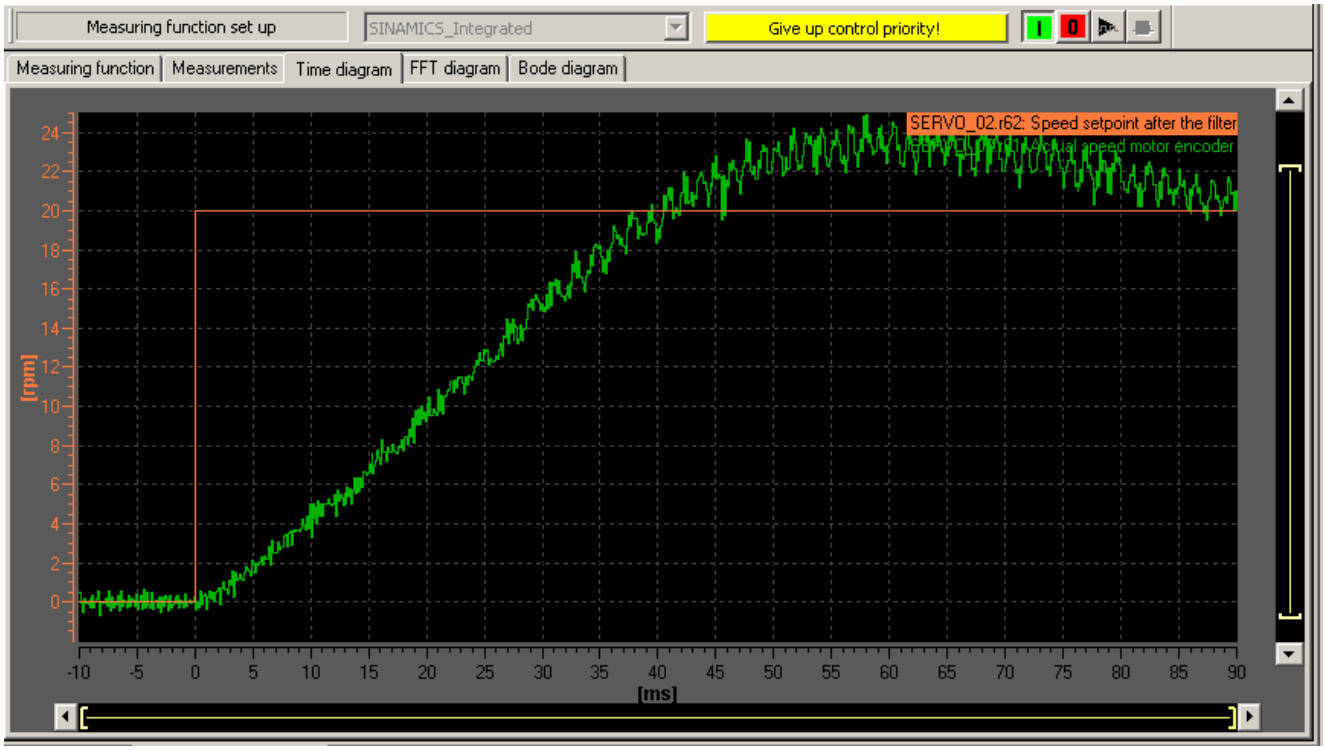


Figure 10-117 Time diagram before parameter change

Adjusting the P-gain

You can adjust the P-gain of the controller to optimize the transient response.

1. In the project navigator under the corresponding drive, for example, Servo_1, use the menu command "Open-loop/closed-loop control" > "Speed controller" to display the "Speed Controller with Encoder" dialog box.
2. Enter an appropriate value in the "P-gain" field and the "Integral time" field.

Note

The values entered take immediate effect.

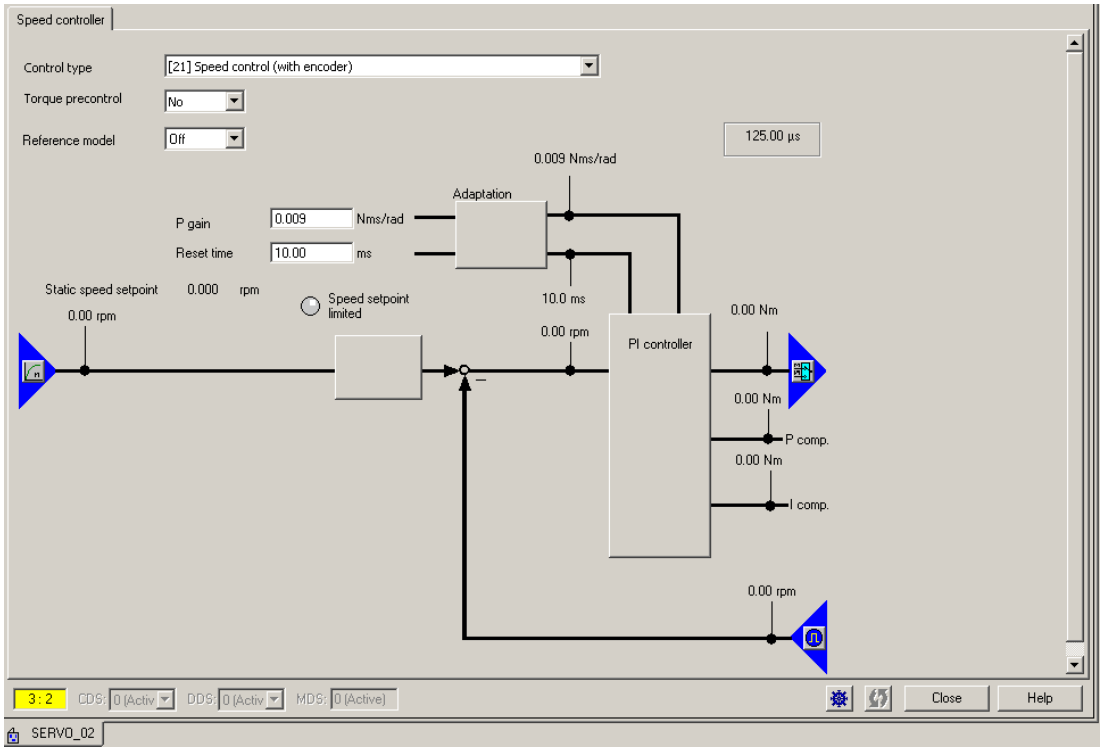


Figure 10-118 Entering P-gain

3. For verification purposes, perform the measurement again.
4. With the modified parameters, the controller displays a much better transient response. If necessary, you can continue changing the value until the transient response is optimal.

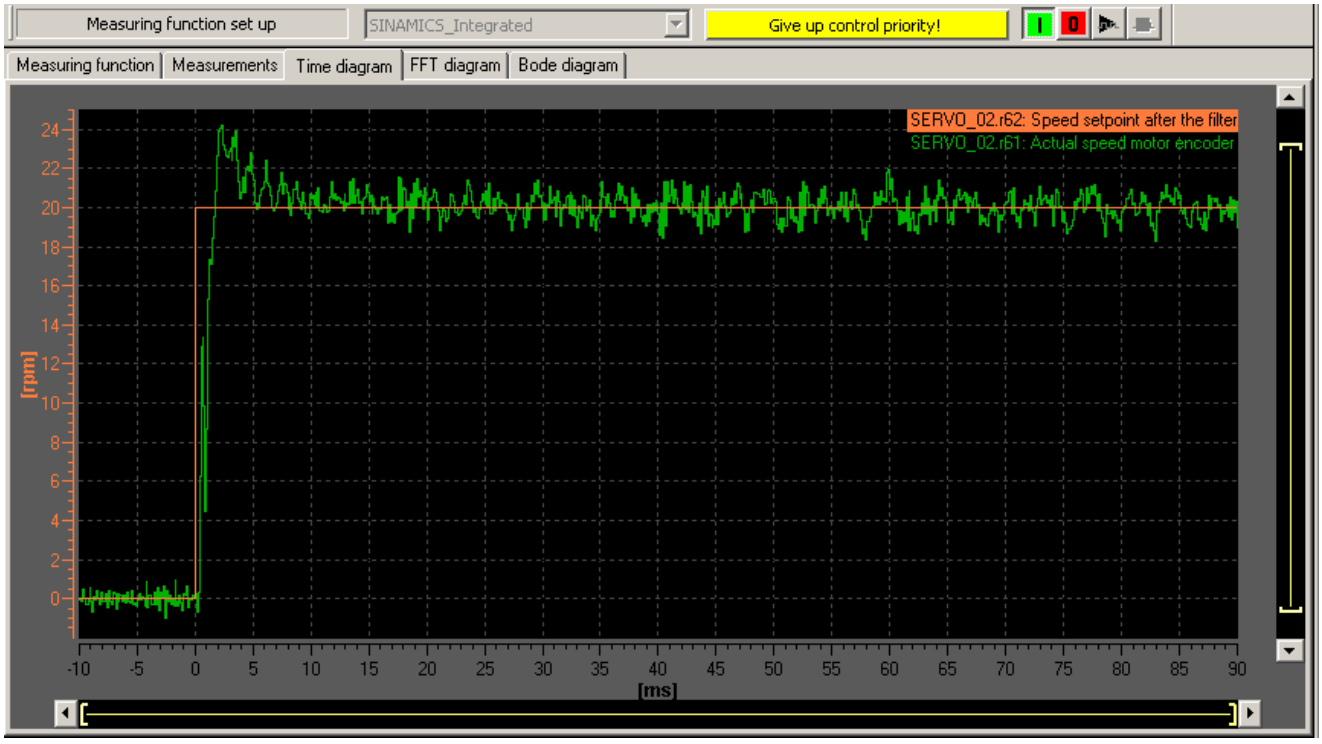


Figure 10-119 Measurement with modified P-gain

Loading and saving SIMOTION user data

Overview

After commissioning the SIMOTION D4x5-2, we recommend that you back up the user data (programs, configuration data and parameter assignments) on the CF card.

Loading user data

The menu command "Target system" > "Load" > "Download project to target system" is used to transfer the following data from the SIMOTION SCOUT engineering system to the "non-volatile SIMOTION data" area of the SIMOTION D4x5-2:

- Configuration data
- Programs
- Parameterization
- Technology packages

The hardware configuration of the SIMOTION D4x5-2 and the retain variables are also stored in the "non-volatile SIMOTION data" area.

Note

Using the menu:

- "Target system" > "Load" > "Load project to target system" downloads all of the project data to the target system.
- "Target system" > "Load" > "Load CPU / drive unit to target device" only loads the data of the selected device / drive element to the target device.

After the SIMOTION D4x5-2 is switched off, the contents of the "volatile SIMOTION data" area are lost.

Additional information on the SIMOTION SCOUT Engineering System can be found in the *SIMOTION SCOUT* Configuration Manual.

Saving user data

The "Copy RAM to ROM" function is used in SIMOTION SCOUT to save the following data from RAM to the CF card:

- Technology packages and user data (units, configuration data, parameter assignments, task configuration) from the "volatile SIMOTION data" area
- Current data values are copied to the "volatile SIMOTION data" area, depending on the settings in SIMOTION SCOUT.

  Note

  The "Copy RAM to ROM" command **cannot** be used to save the current values of retain variables to the CF card.

You have the following options for backing up the current values of retain variables to the CF card:

- User program
  Use the **_savePersistentMemoryData** system function in the user program.
- Back up with the service selector switch or DIAG button on the SIMOTION D4x5-2 or via SIMOTION IT web server: See Section Back up diagnostic data and non-volatile SIMOTION data via the web server (Page 7276)

With the SCOUT functions "Save variables" and "Restore variables," you also have the option of backing up to your PC and restoring data that were changed during operation and only stored in the runtime system.

Execute the "Copy RAM to ROM" function separately for the SINAMICS Integrated. This requires that the drive element has been selected in the Project Navigator.

See also

Properties of the user memory (Page 6987)

Deleting data

Overview of data deletion

The SIMOTION D4x5-2 memory described in the "user memory concept" can be deleted in various graduations. This enables you to determine whether data in your system should be deleted completely or partially.

You have the following options for deleting SIMOTION D4x5-2 data:

- Memory reset of SIMOTION D4x5-2 (Page 7186)
- Deleting user data on CompactFlash card (Page 7189)
- Setting SINAMICS Integrated to the factory settings (Page 7189)
- Setting SIMOTION D4x5-2 to the factory settings (Page 7190)
- Deleting/restoring non-volatile SIMOTION data (Page 7274)
- Backing up / restoring / deleting SINAMICS NVRAM data (Page 7110)

Memory reset of SIMOTION D4x5-2

Introduction

During the memory reset, the memory of the SIMOTION D4x5-2 and the non-volatile SIMOTION data in the NVRAM with the exception of the communication configuration (baud rates, network addresses, etc.), are deleted. The data on the CF card is retained during the memory reset.

You must perform a memory reset of SIMOTION SIMOTION D4x5-2:

- If you want to undo changes made to user data (programs, configuration data, parameter assignments) which you have not backed up by means of the "Copy RAM to ROM" command.
- If the STOP LED is flashing (slow flashing, yellow) to indicate that the SIMOTION D4x5-2 is requesting a memory reset.
- If the non-volatile SIMOTION data and the project on the CF card do not match and an error occurs (diagnostic buffer entry).

You can perform the memory reset online via SIMOTION SCOUT or offline via the mode selector on the SIMOTION D4x5-2.

Data deleted on memory reset

The following data are deleted during a memory reset:

- User data (units, configuration data, parameter settings, task configuration)
- Technology packages

- Retain TO (absolute encoder adjustment)
- Retain variables
  Retain variables are variables in the interface or implementation section of a UNIT that are declared with VAR_GLOBAL RETAIN or global device variables with the RETAIN attribute.

Note

Absolute encoder data are deleted during a memory reset and must therefore be readjusted after the memory reset.

Data retained during memory reset

The following data are retained during a memory reset:

- TCP/IP parameters and DP parameters
- Diagnostic buffer
- Data saved with the **_savePersistentMemoryData**, **_saveUnitDataSet** or **_exportUnitDataSet** system functions and the "Copy RAM to ROM" function
  If backup files (PMEMORY.XML/PMEMORY.BAK) backed up with **_savePersistentMemoryData** exist, the data in these files is restored to the non-volatile SIMOTION data after the memory reset. The user can therefore perform a memory reset to force the backed up non-volatile SIMOTION data to be restored. This also includes the absolute encoder position.
- Licenses
- SINAMICS NVRAM data

The technology packages and user data (configuration data, programs, parameter assignments) that were previously backed up on the CF card using the "Copy RAM to ROM" menu command will be transferred to the "non-volatile SIMOTION data" area of the SIMOTION D4x5-2 during the next power-up. Thus, an existing configuration on the CF card is loaded to the SIMOTION device following the memory reset.

Memory reset via SIMOTION SCOUT

The SIMOTION device for which overall reset is to be performed must be online.

1. Open the "Control Operating State" dialog box in SIMOTION SCOUT. Select the "Target system" > "Control operating state" menu command. Or click on "Control Operating State" on the toolbar.
2. Switch the SIMOTION device for which memory reset is to be performed to STOP in the "Control Operating State" dialog box.
3. Select the SIMOTION device under "Memory reset (MRES)." Click the "Run" button. Confirm the command by clicking "Yes."
   The memory reset will now be performed.

Memory reset with the mode switch

You can perform a memory reset with the mode switch when you are offline with the SIMOTION D4x5-2.

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Comply with the ESD rules.

Proceed as follows to reset the memory:

1. Place the mode switch in the STOP position (switch setting 2, see figure below).

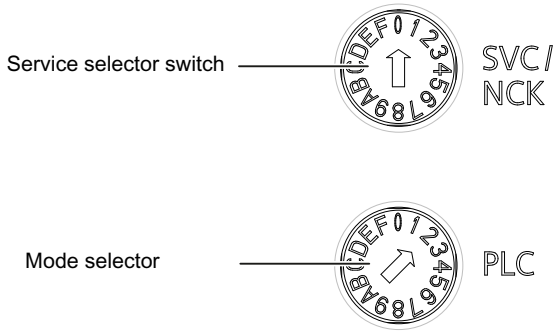


Figure 10-120 Memory reset with the mode switch (position 2 = STOP)

2. When the STOP LED is steadily yellow, turn the switch to the MRES position (switch position 3). The STOP LED starts to flash slowly (slow yellow flashing).
   Wait until the STOP LED stops flashing.
3. Turn the selector back to the STOP setting.
4. You must turn the selector back to the MRES position again within 3 seconds. The memory reset will now be performed.
   The SIMOTION D4x5-2 has completed the memory reset when the STOP LED is steadily yellow.

   Note

   If you do not return the mode switch to the MRES position (switch position 3) within 3 seconds, but take longer, then the memory reset may not be performed. You must then repeat the procedure.

5. Now move the mode switch back to the required operating mode.

NOTICE
Unintentional restoration of the default settings instead of memory reset
Note that the MRES position (switch position 3) during power-up causes the default settings to be restored. See Section Setting SIMOTION D4x5-2 to the default settings (Page 7190).
Make sure you do not accidentally switch the power supply OFF/ON in the MRES selector setting, as this restores the default settings instead of performing the desired memory reset.

Deleting user data on CompactFlash card

Overview

Deletion of the user data from the CompactFlash card is necessary, for instance, if you wish to load another (new) project to the CompactFlash card and therefore find it necessary to delete the existing user data for an old project (e.g. unit data sets) from the CompactFlash card.

You can delete the user data with SIMOTION SCOUT. This requires you to go online on the SIMOTION D4x5-2. The following data is deleted:

- User data from the "volatile data" area
- Non-volatile data, with the exception of IP and DP parameters
- User data on the CompactFlash card (user directories), including the SINAMICS configuration

You can thus continue to go online to the SIMOTION D4x5-2 with your PG/PC. The licenses on the CompactFlash card are retained.

Deleting user data

1. Open the project you want to edit in SIMOTION SCOUT.
2. Go online with the SIMOTION D4x5-2.
3. Select the SIMOTION D4x5-2 in the project navigator and select the "Delete user data on card" option in the "Target system" menu.
4. Confirm the "Delete user data on card" prompt with "OK".
   The user data is deleted and the SINAMICS Integrated goes offline.

Restoring the default settings of SINAMICS Integrated

Requirement

You must be online to SINAMICS Integrated in order to restore its factory settings.

Restoring the factory settings

1. Right-click "SINAMICS_Integrated" in the project navigator.
2. Select the "Target device > Restore factory settings" command from the context menu.

This restores the delivery state of SINAMICS Integrated.

Note

Use the same procedure for the CX32-2 controller extension and the SINAMIC S120 drive units.

Note

Note if a CX32-2 is configured

If the factory settings are restored on a SINAMICS Integrated with an already configured CX32-2, and the SINAMICS Integrated is subsequently uploaded, the CU-Link is deleted and thus also the entry of the CX32-2 in the topology. The project can then no longer be corrected by the user.

Setting SIMOTION D4x5-2 to the factory settings

Overview

SIMOTION D4x5-2 is supplied with preset parameters, such as the transmission rate or PROFIBUS addresses. You can restore the default settings with the mode selector. This will delete the following data:

- Non-volatile SIMOTION data in the SIMOTION device
- The backup copy of non-volatile SIMOTION data on the CF card (PMEMORY.XML/ PMEMORY.BAK)
- User data in the "volatile SIMOTION data" area and on the CF card
- The communication configuration (IP and DP parameters) on the CF card is set to the factory settings.

The licenses on the CF card are retained.

Restoring default settings with the mode selector

1. The power supply is switched off.
2. Set the mode selector on the SIMOTION D4x5-2 to MRES (switch position 3).

3. Switch on the power supply.
   The NVRAM and the SIMOTION user data are deleted. The default settings are loaded. SIMOTION D4x5-2 remains in STOP mode.
4. Now use the mode selector to change to the desired operating state.

Note

The communication parameters are now reset to the default settings (PROFIBUS address 2, baud rate of 1.5 Mbit/s). The communication configuration for SIMOTION D4x5-2 must be performed again. The existing IP settings for the CBE30-2 are restored to the default settings.

Powering down the system

If you wish to shut the system down, you must ensure that all axes and system parts are set to a safe state. You can set this up by providing a separate motion task, for example.

Once the system is at a standstill, you can switch off the power supply.

Note

You must observe the safety notes for SINAMICS components, which you will find in the corresponding SINAMICS manuals.

Configuring Safety Integrated functions

Overview

Integrated safety functions

When used in conjunction with SIMOTION D, the integrated safety functions of SINAMICS S120 provide highly effective practical protection for personnel and machinery.

- Safety Integrated Basic Functions
- Safety Integrated Extended Functions
- Safety Integrated Advanced Functions

Note

As of SIMOTION V5.2 / SINAMICS Integrated V5.1, a distinction is made between Safety Integrated Extended and Advanced Functions. The Safety Integrated Advanced Functions also include the Safety Integrated Extended Functions.

The safety functions listed here conform to:

- Safety Integrity Level (SIL) 2 according to IEC 61508
- Category 3 according to DIN EN ISO 13849-1
- Performance Level (PL) d according to DIN EN ISO 13849-1

The safety functions correspond to the functions according to DIN EN 61800-5-2 (if they are defined there).

Safety Integrated Functions overview

If motors without a (safety-capable) encoder are being used, not all Safety Integrated Functions can be used.

Note

Definition: "Without encoder"

When "without encoder" is used in this manual, then this always means that either no encoder or no safety-capable encoder is being used.

In operation without encoder, the actual velocity values are calculated from the measured electrical actual values. Therefore, velocity monitoring is also possible during operation without encoder.

Table 10-56 Overview of the Safety Integrated Functions

	Functions	Abbreviation	With encoder	Without encoder	Brief description
Basic Functions	Safe Torque Off	STO	Yes	Yes	Safe torque off
	Safe Stop 1	SS1 [1)]	Yes	Yes	Safe stop according to stop category 1
	Safe Brake Control	SBC [2)]	Yes	Yes	Safe brake control

	Functions	Abbreviation	With encoder	Without encoder	Brief description
Extended Functions	Safe Torque Off	STO	Yes	Yes[4]	Safe torque off
	Safe Stop 1	SS1[1]	Yes	Yes[4]	Safe stop according to stop category 1
	Safe Brake Control	SBC	Yes	Yes[4]	Safe brake control
	Safe Operating Stop	SOS	Yes	No	Safe monitoring of the standstill position
	Safe Stop 2	SS2	Yes	No	Safe stop according to stop category 2
	Safely-Limited Speed	SLS	Yes	Yes[5][6]	Safe monitoring of the maximum velocity
	Safe Speed Monitor	SSM	Yes	Yes[5][4]	Safe monitoring of the minimum velocity
	Safe Direction	SDI	Yes	Yes[5][4]	Safe monitoring of the direction of motion
	Sicheres Referenzieren	SR	Yes	No	Safe homing
	Safe Acceleration Monitor	SAM	Yes	Yes[4]	Safe monitoring of the drive acceleration
	Safe Brake Ramp	SBR	Yes	Yes[4]	Safe braking ramp
	Safe gear change	–	Yes	No	–
	Safely-Limited Acceleration	SLA[3]	Yes	No	Safely-limited acceleration
	Safe Brake Test	SBT	Yes[5]	No	Safe test of the required holding torque of a brake
Advanced Functions	Safely-Limited Position	SLP	Yes[5]	No	Safely-limited position
	Transfer of safe position values	SP	Yes[5]	Yes[4]	Transfer of safe position values
	Transfer of Safe Cam position values	SCA[3]	Yes	No	Safe cams

1) Incl. SS1E (SS1 with external brake). In this variant of SS1, SIMOTION is responsible for the stop response.

2) For SBC you require a Safe Brake Relay in addition to Power Modules in blocksize format connected via CUA31/32. A Safe Brake Adapter is required for Power Modules in chassis format.

3) As of SIMOTION V5.2 / SINAMICS Integrated V5.1

4) The use of this safety function without encoder is permitted only for induction motors or synchronous motors of the SIMOTICS A-1FU (previously: SIEMOSYN).

5) As of SIMOTION V4.4 / SINAMICS Integrated V4.7. (SBT: As of V4.4 HF4)

Note

Details on the Safety Integrated functions, such as information on the configuration of safety functions as well as operating conditions for the encoderless operation can be found in the *SINAMICS S120 Safety Integrated* Function Manual.

Notes

Parameterize the desired Safety Integrated functions and the monitoring with or without encoder and activate them in the safety screen forms of the SIMOTION SCOUT engineering system.

If motors are used without an encoder or with an encoder that is not suitable for Safety Integrated Extended Functions, not all Safety Integrated functions can be used (see previous table, column "without encoder").

The safe speed monitoring without encoder also functions at standstill as long as the drive is switched on.

PROFIsafe telegrams

SIMOTION D4x5-2 supports PROFIsafe telegrams 30, 31, 901, 902 and 903. (902 as of SIMOTION V4.4; only in conjunction with the TIA Portal; 903 as of SIMOTION V5.1 for Safe Cam, SCA).

With telegrams 901, 902 and 903, it is possible to apply a factor to the SLS limit value 1 parameterized in the drive, allowing the SLS monitoring limit to be changed during operation. Moreover, in telegrams 901 and 902, the safe position actual values communicated in the SINAMICS S120 are transferred via PROFIsafe to the F-CPU (function SP). With telegram 903, the state of up to 30 cams can be transferred.

Note

The "Transfer safe position values via PROFIsafe" function is supported as of SIMOTION V4.4.

Activate Safety Integrated functions

Control

The Safety Integrated functions are completely integrated into the drive system. They can be activated as follows:

- Safety Integrated Basic Functions are activated via terminals on the device or via a PROFIsafe telegram by means of PROFIBUS or PROFINET.
- Safety Integrated Extended/Advanced Functions are activated via the TM54F Terminal Module or via a PROFIsafe telegram by means of PROFIBUS or PROFINET.
- The Safety Integrated Basic Functions via terminals and the Safety Integrated Extended/ Advanced Functions (via TM54F or PROFIsafe) can be used simultaneously.
- As of SIMOTION V4.5, the Safety Integrated Basic Functions STO, SS1 and SBC can also be controlled license-free via the TM54F. (Requirement: STARTER V4.4 SP1; STARTER V4.4 SP1 is not part of SCOUT V4.4 - the functionality is therefore only available as of SCOUT V4.5)
- The SLS and SDI functions can also be activated permanently via parameter assignment.
- The SLA Extended Function can only be activated via PROFIsafe.

The output signals of the SCA Advanced Function can only be output via PROFIsafe.

Note

PROFIsafe or TM54F

Using a Control Unit, control is possible either via PROFIsafe or TM54F. Mixed operation is not permitted.

The Safety Integrated functions are implemented electronically and therefore offer short response times in comparison to solutions with externally implemented monitoring functions.

Note

Although SIMOTION does not contain any safety-related functionality, it provides support for SINAMICS drives that can perform safety-related functions.

The purpose of this SIMOTION support, depending on the activated the safety-related monitoring functions, is to prevent fault reactions at the drive end by ensuring that the drive does not exit the monitored operating state.

Further information on support of the SINAMICS Safety Integrated functions on the TO axis can be found in the *TO Axis Electric/Hydraulic, External Encoder* Function Manual.

Required hardware

When safety functions are controlled via the TM54F or via PROFIBUS/PROFINET with PROFIsafe, at the very least the following hardware versions must be used:

Table 10-57 Required hardware versions

Module	Article number	Required product version
SIMOTION D425-2 DP	6AU1425-2AA00-0AA0	E
SIMOTION D425-2 DP/PN	6AU1425-2AD00-0AA0	E
SIMOTION D435-2 DP	6AU1435-2AA00-0AA0	E
SIMOTION D435-2 DP/PN	6AU1435-2AD00-0AA0	E
SIMOTION D445-2 DP/PN	6AU1445-2AD00-0AA0	C
	6AU1445-2AD00-0AA1	C
SIMOTION D455-2 DP/PN	6AU1455-2AD00-0AA0	C
SIMOTION CX32-2	6AU1432-2AA00-0AA0	B
CBE30-2	6FC5312-0FA00-2AA0	A

The hardware requirements of the drive components can be found in the *SINAMICS S120 Safety Integrated* Function Manual.

Quantity structures

The maximum number of servo drives with Safety Integrated functions for each Control Unit is:

Table 10-58 SIMOTION D4x5-2 quantity structures

Maximum number of drives when safety function is activated (servo / vector / V/f) [3]	**SIMOTION D4x5-2** [5]	**CX32-2**	**CU320-2**
• Via CU/EP terminals (STO, SBC, SS1 only)	6 / 6 / 12	6 / 6 / 12	6 / 6 / 12
• Motion monitoring without selection	6 / 6 / 12 [4]	6 / 6 / 12 [4]	6 / 6 / 12 [4]
• Via TM54F terminals	6 / 6 / 6	6 / 6 / 6	6 / 6 / 6
• Via PROFIBUS with PROFIsafe [1]	6 / 6 / 11	6 / 6 / 11	6 / 6 / 11
• Via PROFINET with PROFIsafe [2] (D4x5-2 DP/PN only)	6 / 6 / 11	6 / 6 / 11	6 / 6 / 11

1) PROFIBUS with PROFIsafe:
If SIMOTION D is operated as an intelligent DP slave on an F-CPU, a maximum of 32 safety slots can be configured for each intelligent DP slave in **HW Config**. If the Safety Integrated functions are controlled via PROFIBUS with PROFIsafe, one slot in the input area and one slot in the output area are used for each safety axis. This results in a maximum of 16 drives with the Safety Integrated function for each PROFIBUS interface (intelligent DP slave) of the D4x5-2. If with SIMOTION D both PROFIBUS interfaces are operated as an intelligent DP slave on a F-CPU, the quantity structure increases to 2 x 16 drives (sum of the drives on the SINAMICS Integrated, CX32-2 and SINAMICS Control Units). In this case however, no PROFIBUS master interface is available for the connection of I/Os, for example.

2) PROFINET with PROFIsafe:
The maximum permissible quantity structure for the D4x5-2 is max. 128 drives depending on the Control Unit (< V4.2 SP1 HF1: 42 drives; V4.2 SP1 HF1 to V4.4: 64 drives) with Safety Integrated function (sum of drives on SINAMICS Integrated, CX32-2 and SINAMICS Control Units).

3) Data for V/f refers to versions as of SIMOTION V4.3 / SINAMICS V4.5

4) All axes V/f control, 500 µs, Safety Integrated with encoder

5) Also with CX32-2 connected to D4x5-2

The maximum drive quantity structure depends on other factors. For example with T_i= 125 µs, maximum of four Motor Modules may be operated with Safety Integrated Extended/Advanced Functions on a DRIVE-CLiQ line. Additional drive components (e.g. Line Modules, Terminal Modules, encoders, etc.) can also reduce the maximum drive quantity structure.

Recommendation:
With the SIZER configuration tool, you can easily configure the SINAMICS S110/120 drive family including SIMOTION.

For further information see

- *SINAMICS S120* Function Manual (FH1), Section *Rules for wiring with DRIVE-CLiQ*
- *SINAMICS S120 Safety Integrated* Function Manual

Use of two PROFINET interfaces

An F-CPU can only be connected to one of the two PROFINET interfaces, because fail-safe I/O transfer areas can only be configured either on the I-device on the onboard PROFINET interface or on the CBE30-2.

The PROFIsafe telegrams are routed to the following drives:

- Drives on the SINAMICS Integrated and CX32-2
- Drives on the onboard PROFINET interface (X150 interface)
- Drives on the CBE30-2 (X1400 interface)

The maximum quantity structure for PROFIsafe on PROFINET is therefore not increased by using a second PROFINET interface.

Safety Integrated functions with TM54F

Safety Integrated functions are activated via fail-safe digital inputs on the TM54F. This means that every drive control (SINAMICS Integrated of SIMOTION D4x5-2, CX32-2, CU320-2, etc.) requires its own TM54F (assuming appropriate safety functions are to be used on the respective Control Unit). Only one TM54F can be connected for each SIMOTION D4x5-2, CX32-2 or SINAMICS S120 Control Unit.

Safety Integrated functions with PROFIsafe (PROFINET example)

Safety Integrated functions are activated via "PROFIsafe on PROFINET" safe communication. Control (F logic) is via a SIMATIC F-CPU which is connected to PROFIsafe via PROFINET, e.g. a SIMATIC S7-1500 F-CPU.

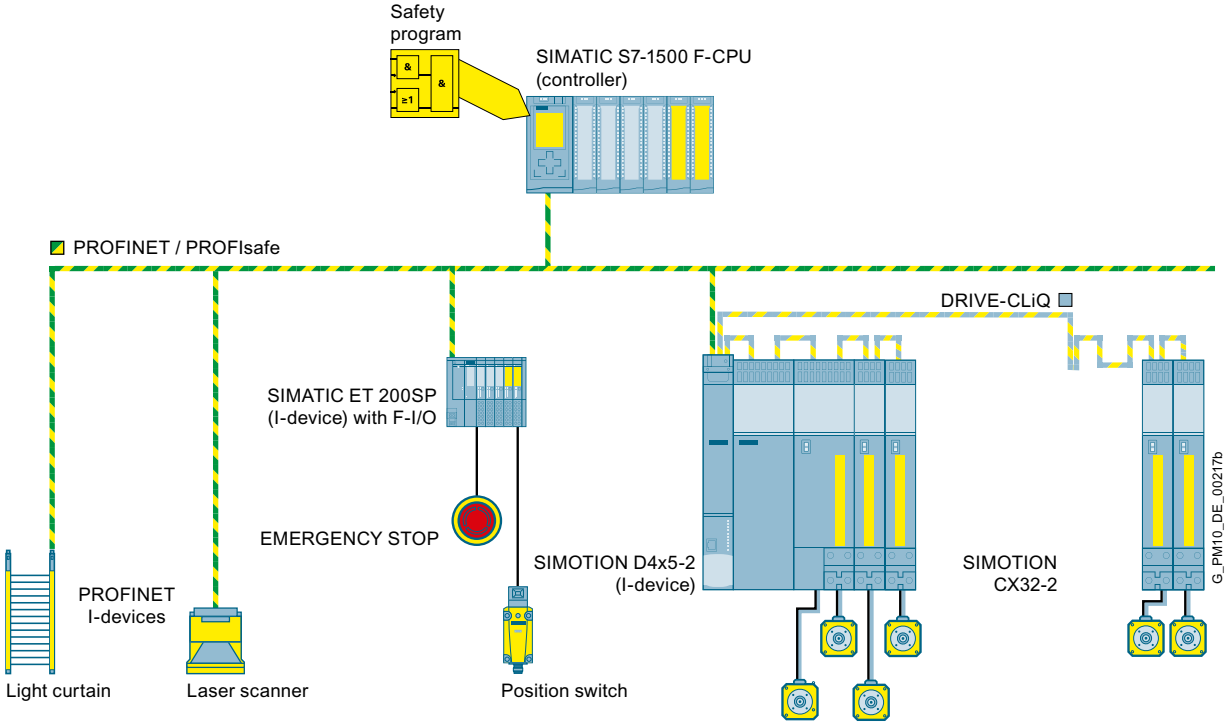


Figure 10-121 SIMOTION D, control of F functions via PROFINET with PROFIsafe

You will find an example function in Internet (https://support.industry.siemens.com/cs/ww/en/view/36489289).

Topologies (F proxy)

The F proxy functionality enables transparent routing of safety telegrams from the SIMOTION I-device (or slave) to the SIMOTION controller (or master).

The topologies that may be used with the SIMOTION D4x5-2 and CX32-2 are listed below. In each case, the description specifies whether the control of Safety Integrated functions is routed through to the drives.

- SIMATIC F-CPU (master), connected via PROFIBUS with PROFIsafe to SIMOTION D4x5-2 (I-slave)
  Note: PROFIBUS not possible configuration via SCOUT TIA
  - Routing to SINAMICS Integrated drives of the D4x5-2 and the CX32-2.
  - Routing to the drives of a SINAMICS S110/S120 CU connected to the D4x5-2.
    The CU is connected as a slave via PROFIBUS to the DP master interface of the D4x5-2.
    Routing to a CU connected via PROFINET is not possible.
- SIMATIC F-CPU (controller), connected via PROFINET with PROFIsafe to SIMOTION D4x5-2 (I-device)
  - Routing to SINAMICS Integrated drives of the D4x5-2 and the CX32-2.
  - Routing to the drives of a SINAMICS S110/S120 CU connected to the D4x5-2.
    The CU is connected as device via PROFINET to the PROFINET interface of the D4x5-2 (=controller) or the CU is connected as a slave via PROFIBUS to the DP master interface of the D4x5-2.
- The SIMOTION D4x5-2 is the PROFIBUS master for a direct fail-safe slave-to-slave communication, e.g. between a SIMATIC F-CPU (e.g. ET 200S F-CPU) and a SINAMICS S110/S120 CU.
  Direct fail-safe slave-to-slave communication with telegram 901: As of V4.3 SP1 HF9
  Not for SCOUT TIA (SIMOTION in the TIA Portal)
  See also application example in Internet (https://support.industry.siemens.com/cs/ww/en/view/38701812).

Note

Control for the Safety Integrated functions cannot be routed to the SINAMICS Integrated of the D4x5-2 or a CX32-2 in this constellation.

Topologies (Shared Device)

With the Shared Device functionality, you can configure access to an IO device with several IO controllers using PROFINET. This enables channels/modules to be flexibly assigned to different IO controllers. This option is available for inputs and outputs. You can use this mechanism to access the fail-safe data of a drive configured below a SIMOTION CPU via the F-CPU (e.g. access of an S7-1500 F-CPU to a SINAMICS S120 CU320-2 which is subordinate to a SIMOTION D4x5-2).

Requirement:

SIMOTION SCOUT (not possible with SCOUT TIA)

Note

When configuring PROFIsafe, it is recommended that you use the I-device F proxy functionality instead of the Shared Device.

Additional references

Additional information on configuring Safety Integrated functions can be found in the

- *SINAMICS S120 Safety Integrated* Function Manual
- *TO Axis Electrical/Hydraulic, External Encoder* Function Manual
- *SIMOTION Communication* System Manual
- Internet at the following address (https://www.automation.siemens.com/safety)

Quantity structures

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S 120 drive software run on a shared control hardware. The IEC 61131-3-compliant PLC integrated in SIMOTION D means that the system is not just capable of controlling sequences of motions, but the entire machine as well.

- The **scaling of the PLC and motion control functionality** is performed via Control Units of various performance classes for SIMOTION D
  - SIMOTION D410-2 is predestined for single-axis solutions and small multi-axis solutions with typically 2-3 axes (max. 8 axes)
  - SIMOTION D425-2 (BASIC performance) for up to 16 axes
  - SIMOTION D435-2 (STANDARD performance) for up to 32 axes
  - SIMOTION D445-2 (HIGH performance) for up to 64 axes
  - SIMOTION D455-2 DP/PN (ULTRA-HIGH performance) for up to 128 axes or applications with very short control cycle clocks

The possible axis quantity structures depend on the required servo and interpolator cycle clocks and apply for electric, hydraulic and virtual axes.

- The **scaling of the drive computing performance** is performed via SINAMICS drive controls for the SIMOTION D4x5-2. With the SIMOTION D4x5-2, the drive control of a CU320-2 is already integrated (SINAMICS Integrated). It can be extended as follows:
  - SINAMICS S110/S120 Control Units can be connected via PROFIBUS or PROFINET
  - The CX32-2 Controller Extension can be connected via DRIVE-CLiQ

SIZER

With the SIZER configuration tool, you can easily configure the SINAMICS S120 drive family including SIMOTION. It provides you with support for selecting and dimensioning the components required for a Motion Control task.

- Dimensioning of the PLC and motion control functionality --> selection of the SIMOTION D Control Unit
- Dimensioning of the drive computing performance and the required drive components.

Depending on your performance requirements, SIZER determines the possible number of axes and the resulting utilization on the SIMOTION and SINAMICS side.

Drive quantity structure

The following table can be used for a rough estimation of the maximum possible quantity structures on the drive side.

Depending on the functionality used, more or less performance resources are used on the drive control.

The performance points specified in the table can be used for a rough estimation. Maximum 15.5 performance points are available for each drive control (SINAMICS Integrated, CX32-2, CU320-2).

As well as the maximum number of available performance points, the maximum quantity structure of a functionality must also be taken into account (e.g. max. 6 servo drives with 125 µs current controller cycle clock).

Note

The following method just serves as a rough estimation of the possible quantity structures. It is not a substitute for dimensioning with the SIZER configuration tool.

Higher utilization can result through the use of SINAMICS function modules (e.g. extended messages / monitoring functions) or with a DP Integrated cycle clock < 0.5 ms.

Table 10-59 Estimation of the possible quantity structure

Information on dimensioning					**Your dimensioning (example)**	
Functionality	Characteristic [2]	max. number	Performance points		Quantity	SUM of the performance points
Line Module [5]	ALM/SLM with DRIVE-CLiQ	1	1.3		1	1.3
	BLM	1	0.4			
Servo[1]	125 µs	6	2	For each drive	6	12
	250 µs	6	1.2	For each drive		
2nd encoder	Not relevant	6	2 encoder/ drive	0.1		
DO encoder	1 ms	19	0.4	For each DO		

Information on dimensioning					Your dimensioning (example)	
Vector [1]	250 µs	3	4	For each drive		
	500 µs	6	2	For each drive		
V/f	500 µs	12	1	For each drive		
Safety Extended/ Advanced Function	12 ms	Depending on the drive	0.25	For each drive	6	1.5
TB30	Not relevant	1	0.1			
TM15 DI/DO	250 µs 1 ms 4 ms	8 [3]	0.5 0.2 0.1	For each TM		
TM31	250 µs 1 ms 4 ms	8 [3]	0.5 0.25 0.2	For each TM		
TM41 (SIMOTION operating mode)	125 µs / 2 ms [4] 125 µs / 4 ms [4] 250 µs / 4 ms [4]	8 [3]	1 1 0.5	For each TM		
TM54F	---	1 [3]	0.5	For each TM		
TM15/TM17	Not relevant	8 [3]	1	For each TM		
CX32-2	Not relevant	6	0.3	For each CX32-2		
CBE20 (for CU320-2)	Not relevant	1	0.5			
DCC SINAMICS	2 ms	75 DCC	2			
		50 DCC	1.5			
			SUM (max. 15.5)			14.8

[1] No mixed servo-vector operation possible

[2] Used time slice (for drives: Speed controller cycle clock)

[3] Absolute maximum: 16 Terminal Modules; these quantities can only be achieved with considerable restrictions (e.g. not with TM15/TM17).

[4] Sampling time for encoder emulation / sampling time for digital and analog I/Os

[5] Max. one Line Module, SLM without DRIVE-CLiQ (5 kW and 10 kW) not relevant

The required performance resources depend on further factors. The influence of the functions listed below can be neglected for an estimation using performance points:

- Use of DSC (linear)
- Connection of blocksize Power Modules via CUA31/32 ⇒ counts as booksize Motor Modules.

During the drive configuration, note that each CX32-2 on the SINAMICS Integrated generates an additional load. The transfer of a CX32-2/CU320-2 configuration on the SINAMICS Integrated is

therefore only possible with a connected CX32-2 when the CX32-2/CU320-2 were not fully utilized.

Note

Note also the topology rules of the SINAMICS S120, see *SINAMICS S120* Function Manual, Section *Rules for wiring with DRIVE-CLiQ*.

Mixed operation

Note the following when mixing control types on a drive control:

- A mixture of servo-controlled and vector-controlled drives is not possible
- A mixture of servo-controlled and V/f-controlled drives is possible
- A mixture of vector-controlled and V/f-controlled drives is possible

With regard to the drive quantity structure, the maximum number of servo or vector drives may not be exceeded by drives with V/f control in mixed operation, whereby two drives with V/f control (500 µs) count as one servo or vector drive.

In mixed operation of vector control and V/f control, the following axis constellations and current controller sampling times are not possible:

- 1 axis (vector control, 250 µs) and 8 axes (V/f control, 500 µs)
- 2 axes (vector control, 250 µs) and 4 axes (V/f control, 500 µs)

Migration of SIMOTION D4x5 to SIMOTION D4x5-2

Changeover from SIMOTION D4x5 to SIMOTION D4x5-2

A SIMOTION D4x5-2/CX32-2 not only has a different structure, but also a different functionality to a SIMOTION D4x5/CX32.

This has effects that must be taken into account during a changeover.

Change from D4x5 to D4x5-2 (upgrade)

The change from D4x5 to D4x5-2 is performed via a module replacement in **HW Config**. That means the change is similar to that of a D425 to D435.

The module replacement is started by dragging the new module to the frame of the module rack with the existing module in **HW Config**, see Section Device replacement in HW Config (Page 7228).

The module replacement is performed automatically. During this operation, the technology packages and the device versions are updated.

Existing configurations of the D4x5 are mostly taken over by the D4x5-2 during the module replacement.

Information on when project adaptations are required or not required when changing from a D4x5 to a D4x5-2 can be found in the following sections (see Table *Transfer of existing projects from D4x5 to D4x5-2*).

For further information on measures generally required for project adaptations, see Section Service and maintenance (Page 7209).

Change from D4x5-2 to D4x5 (downgrade)

As a SINAMICS drive cannot be downgraded, a module replacement of a SIMOTION D4x5-2 (SINAMICS Integrated firmware V4.x/V5.x) with a SIMOTION D4x5 (SINAMICS Integrated firmware V2.x) is not possible.

However, project data can be transferred by means of an XML export/import.

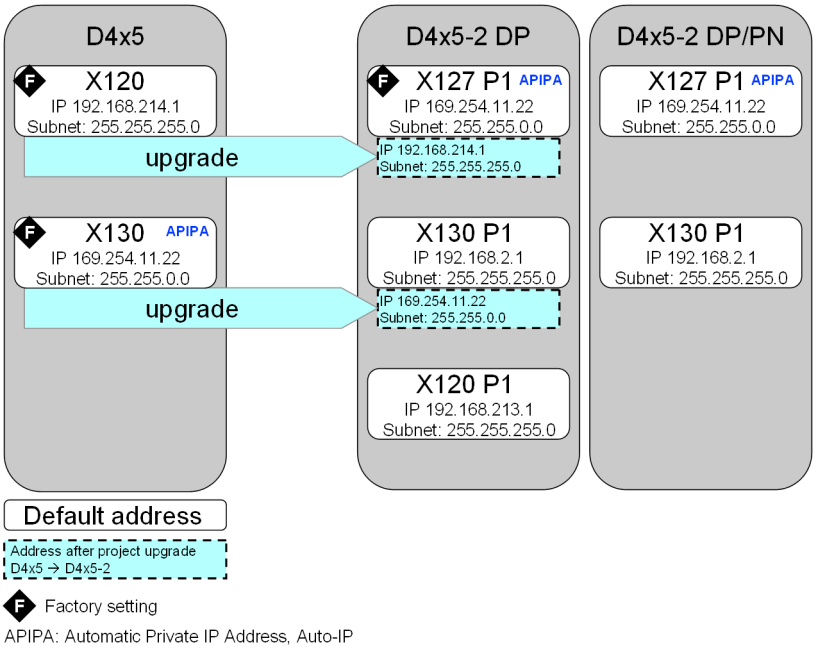
Transfer of existing projects

Existing configurations of the D4x5 are mostly taken over by the D4x5-2 during the module replacement.

The following table provides information on where project adaptations are required or not required:

Table 10-60 Transfer of existing projects from D4x5 to D4x5-2

Keyword	Explanation
PROFINET interface	D4x5 with CBE30 (4 ports) is mapped on D4x5-2 DP/PN onboard PROFINET interface (3 ports): • Only the wiring of ports 1 to 3 are taken over • Components on port 4 have to be reconnected • Communication relations, I/O data, synchronous operation connections, data exchange broadcast data, module addresses and diagnostics addresses are retained • Any existing PG/PC assignment is taken over Mapping of D4x5 on D4x5-2: X1400 (P1, P2, P3, not P4) → X150 (P1, P2, P3)
PROFIBUS interfaces	The configurations on the PROFIBUS interfaces including any existing PG/PC assignment are retained. Mapping of D4x5 on D4x5-2: X126 DP1 → X126 DP X136 DP2/MPI → X136 DP/MPI
Reduction ratio of PROFIBUS cycle clock to servo cycle clock	As of SIMOTION V4.3, the SIMOTION D4x5-2 supports a reduction ratio of PROFIBUS cycle clock to servo cycle clock. The following applies for V4.2: **PROFIBUS cycle clock: servo cycle clock = 1 : 1** If SIMOTION D4x5 projects have been transferred in which a reduction ratio factor was set, • a SIMOTION version ≥ V4.3 must be used, or • the cycle clock setting must be adapted for V4.2 projects

Keyword	Explanation
Ethernet interfaces	The configurations on the Ethernet interfaces including any existing PG/PC assignment are retained. **Mapping of D4x5 on D4x5-2:** X120 IE1/OP → X127 P1 PN/IE (interface on the front of the module) X130 IE2/NET → X130 P1 PN/IE-NET (interface under the front flap) The X120 P1 PN/IE-OP interface is only available on the D4x5-2 DP. Mapping is never performed on this interface. The following figure provides an overview of the default addresses as well as the behavior when upgrading a D4x5 to D4x5-2 (assumption: The default addresses have been used for the D4x5).  With SIMOTION D4x5, the Ethernet interfaces of the D4x5 have default addresses in the delivery condition. These addresses are also offered as default addresses in the configuration of SIMOTION SCOUT. With SIMOTION D4x5-2, only interface X127 P1 has a default address in the delivery condition. Interface X127 P1 is preferable for the PG/PC connection because it is easier to access. The default address is an Automatic Private IP Address (APIPA, Automatic Private IP Addressing). In this way, a PG/PC can automatically configure the network interface. Interfaces X120 P1 and X130 P1 do not have an address in the delivery condition. When configuring the interfaces, the addresses shown in the figure are offered as default addresses.
Telegram configurations	Telegram configurations on the PROFIBUS and PROFINET interfaces are retained. The same applies for the addresses of the slots/subslots.
SINAMICS Integrated / Controller Extension	The SINAMICS Integrated and Controller Extension are changed to the new type and its corresponding version. The drive configuration and the assignments of the DRIVE-CLiQ interfaces are retained.

Keyword	Explanation
Configuration of the onboard I/Os	The configuration of the onboard I/Os is retained. Functionality, quantity structure and terminal designations of the onboard I/Os for the D4x5-2/CX32-2 differ from the previous modules because of innovations. Example: • D4x5-2 has additional technology I/Os on terminal block X142 (e.g. for high-precision cam outputs); a cam output on X122/X132 is no longer supported • Terminal blocks X122/X132 have additional digital inputs and therefore the pin assignments on the terminal block have been moved.
CF card / firmware	SIMOTION D4x5 and D4x5-2 have different CompactFlash cards and different card mappers (firmware). 1 GB CF cards of the D4x5 can also be used for the D4x5-2 when a new boot loader is loaded. For details, see Section Boot loader on the CompactFlash card (Page 7247).
Mixture of Control Units of different generations	The firmware version of a Controller Extension (CX) is always identical to the SINAMICS Integrated. Therefore • The CX32 can only be operated on the D4x5 • The CX32-2 can only be operated on the D4x5-2 These dependencies do not apply for SINAMICS S120 Control Units. A CU320 can therefore be operated on a D4x5-2.
Connectable drive components	As for the CU320-2, some older drive components are also no longer supported by the D4x5-2/CX32-2. For details, see Section Permissible combinations (Page 7206)
User program	In principle, a D4x5 user program can run on a D4x5-2, however, adaptations may be required because of innovations in the hardware. Examples: • The non-volatile data is backed up permanently and maintenance-free for the D4x5-2 (backed-up SRAM for D4x5); a buffer failure signaled via **persistentDataPowerMonitoring.powerFailure = "YES"** therefore only applies to the real-time clock, not for the non-volatile data. • The runtime behavior of the D4x5-2 modules changes because of the increase in performance. As long as the user program does not contain any "runtime-dependent code", adaptations should not be necessary. • The D4x5-2 has a double fan module. The user program must be adapted in order to profit from the advantages of a double fan compared with a single fan (identification of required maintenance). See Section Overview of the fan/battery module states (Page 7002). • The user program will have to be modified with respect to detection of the maintenance case for D425-2/D435-2 if D425/D435 was used without a fan/battery module. See Section Overview of states, fan/battery module (Page 7002). • The **batterynecessary** and **fannecessary** system variables have different states depending on the device. See – Section Operations and their effect on the user memory (Page 6990) and – Section Cooling the SIMOTION D4x5-2 (Page 7001) • As of SIMOTION V4.3, the following additional functions are available for the D4x5-2: – System variables **rtcFailure** and **retainDataFailure** – Backup of the non-volatile SINAMICS data (NVRAM data) via CU parameter p7775 – Automatic detection of a module replacement on the basis of the serial number of the Control Unit

Keyword	Explanation
Hardware	The following changes have been made because of the hardware innovations: • CF card under a cover (previously open access) • Additional and slightly differently arranged operator control and display elements • Modified connectors (larger cable cross-section, screw connection on the 24 V terminal block, etc.) • D445-2 DP/PN and D455-2 DP/PN have a heat sink so that the spacers (with the exception of external heat dissipation) cannot be disassembled as for the D445-1. See also Section Mounting a SIMOTION D4x5-2 (Page 6923). • A double fan/battery module is always required for D4x5-2 Control Units. See Section Cooling the SIMOTION D4x5-2 (Page 7001). • Controller Extensions (CX) for D4x5 can only be used for the D435 and D445/D445-1; with the D4x5-2, all Control Units support the use of CX32-2 Controller Extensions.
Installation options	For the SIMOTION D4x5 Control Units, it was possible to mount the Control Unit directly on the Line Module via lateral mounting elements. This mounting option is no longer offered for D4x5-2 Control Units because of their higher weight. This mounting option is now only available for the CX32-2 Controller Extension and the CU320-2 Control Unit.

Permissible combinations

DRIVE-CLiQ components

Some "older" DRIVE-CLiQ components can no longer be used with the SIMOTION D4x5-2/ CX32-2.

Table 10-61 DRIVE-CLiQ components that cannot be used with the D4x5-2/CX32-2

DRIVE-CLiQ components	Notes	Ending of the Article No.
Single Motor Modules (SMM) Double Motor Modules (DMM)	6SL31* (all Motor Modules Booksize Compact 6SL34* can be used)	< 3
Line Modules (ALM/SLM)	6SL313* The following can generally be used: • All BLM (6SL313*) • SLM (6SL313*) 5 and 10 kW	< 3
Chassis power units (Line Modules, Motor Modules, Power Modules)		< 3
SMC30 Sensor Module Cabinet		< 2
TM31/TM41 Terminal Modules		< 1
SME20/25 Sensor Modules External		< 3
CUA31 Control Unit Adapter		< 1
Power Modules (chassis)		< 3

A detailed, regularly updated list of the DRIVE-CLiQ components approved for use with SIMOTION, as well as notes on their use, can be found on the Internet at (https://support.industry.siemens.com/cs/ww/en/view/11886029).

If incorrect components are used, a topology error is signaled
F01360 Topology: Actual topology not permitted.

CX32-2

The tables below provide an overview of the possible combinations:

Table 10-62 Controller extension combinations

Controller extension	Article number	Use in D4x5	Use in D4x5-2
CX32	6SL3040-0NA00-0AA0	Possible	Not possible
CX32-2	6AU1432-2AA00-0AA0	Not possible	Possible

Only approved combinations can be configured. If physically an incorrect controller extension is used, a topology error is signaled.
F01360 Topology: Actual topology not permitted

CBE30-2

The tables below provide an overview of the possible combinations:

Table 10-63 Communication Board Ethernet combinations

Communication Board Ethernet	Article number	Use in D4x5	Use in D4x5-2
CBE30	6FC5312-0FA00-0AA0	Possible	Not possible
CBE30-2	6FC5312-0FA00-2AA0	Not possible	• D4x5-2 DP/PN: As of V4.3 • D4x5-2 DP: Not possible

Only approved combinations can be configured. If physically an incorrect Communication Board Ethernet is used, the CBE cannot be fully inserted into the option slot.

CF card

A CF card with D4x5 firmware / drive software cannot run on a D4x5-2. The same applies for the reverse situation.

In case of an error, the 6 LED displays (RUN to DP/MPI) flash yellow at 2 Hz.

Licenses

Licenses purchased for SIMOTION D can be used for both the D4x5 and D4x5-2.
Previously used platform-specific "MultiAxes Package for D445/D445-1" licenses can also be used for the D445-2 DP/PN and D455-2 DP/PN.

A new license has not been introduced for the D455-2 DP/PN performance class.

Migration of SIMOTION D445-2 (-0AA0) to D445-2 (-0AA1)

SIMOTION D445-2 (article number 6AU1445-2AD00-0AA1) is the successor type of SIMOTION D445-2 (article number 6AU1445-2AD00-0AA0). Availability: from approx. mid/end 2020.

Compatibility

SIMOTION D445-2 (-0AA1) is compatible with SIMOTION D445-2 (-0AA0). (Compatibility regarding installation, electrical connections, project and engineering).

Configuration with older software versions

SIMOTION D445-2 (6AU1445-2AD00-0AA1) is available as of SCOUT/SCOUT TIA Version V5.4.x in the module catalog.

For SCOUT/SCOUT TIA versions (≤ V5.3 SP1), the new article number (6AU1445-2AD00-0AA1) is not yet included in the module catalog. SIMOTION D445-2 (6AU1445-2AD00-0AA1) can be configured due to the compatibility with the article number (6AU1445-2AD00-0AA0).

Spare part scenario

For a spare part scenario, you can swap the CF card from the predecessor (-0AA0) to the successor (-0AA1). The SCOUT/SCOUT TIA project does not have to be changed when exchanging SIMOTION D445-2 (-0AA0) for SIMOTION D445-2 (-0AA1). You also do not need to upgrade the SCOUT/SCOUT TIA project to the successor type (-0AA1).

Note

Depending on the application, SIMOTION D445-2 (-0AA1) provides higher performance than SIMOTION D445-2 (-0AA0). A configured CF card can be operated with SIMOTION D445-2 (-0AA0) and D445-2 (-0AA1) under the following conditions:

- When changing from SIMOTION D445-2 (-0AA0) to SIMOTION D445-2 (-0AA1), the application is able to cope with the increased PLC and motion control performance; i.e. the application is programmed in such a way that shorter running times do not result in a deviating/incorrect program sequence.
- When changing from SIMOTION D445-2 (-0AA1) to SIMOTION D445-2 (-0AA0), the lower PLC and Motion Control performance is sufficient for the application. If this is not the case, adjustments need to be made (e.g. in the cycle clock settings).

It is recommended to ensure there is no effect on the application before changing the module.

10.1.1.8 Service and maintenance

Overview

Introduction

It is possible to distinguish between the following scenarios when replacing and updating components:

- Replacing modules (spare part scenario)
  - Parts replacement for SIMOTION D4x5-2 (Page 7213)
  - Removal and replacement of the SIMOTION D4x5-2 (Page 7214)
  - Replacing DRIVE-CLiQ components (Page 7215)
  - For information on replacing SIMOTION D4x5-2 fans, see Chapter "Replacing the battery in the fan/battery module" in the *SIMOTION D4x5-2* Manual
  - Replacing the CompactFlash card (Page 7217)
- Adapting a project (new device type / new device version)
  The project needs to be adapted if you want to change the type (e.g. D445 ⇒ D445-2 DP/PN) or the version of the SIMOTION device in your existing project.
  - Creating backup copies (project/CF) (Page 7222)
  - Backing up user data (Save variables) (Page 7222)
  - Upgrading a user project to the new SCOUT version (Page 7224)
  - Platform replacement via XML export/import (Page 7225)
  - Preparing the device replacement (Page 7226)
  - Device replacement in HW Config (Page 7228)
  - Upgrading technology packages (Page 7228)
  - Upgrading the device version of SINAMICS S120 control units (Page 7231)
  - Upgrading a library (Page 7232)
  - Save project, compile and check consistency (Page 7233)
- Performing a firmware and project update
  - Upgrading the boot loader on the CF card (Page 7233)
  - Update - preparatory measures (Page 7234)
  - Update via SIMOTION IT web server (Page 7235)
  - Upgrade via device update tool (upgrading SIMOTION devices) (Page 7235)

– Update via CF card
  Backup of the CF card data (Page 7239)
  Firmware update via CF card (Page 7240)
  Upgrading SINAMICS (Page 7241)
  Download project to target system (Page 7243)
  General information on SINAMICS firmware update (Page 7244)

Note

Upgrading using the device update tool offers a number of advantages (keeping retain data, option of downgrading, no license key handling, etc.). We would therefore recommend using this method for firmware and project updates.

Please also observe the information on handling the CompactFlash Card.

- Changing the CompactFlash card (Page 7245)
- Writing to a CompactFlash card (Page 7246)
- Formatting the CompactFlash card (Page 7246)
- Boot loader on the CompactFlash card (Page 7247)
- Recommended method of handling CF cards (Page 7248)
- Card reader for CF cards (Page 7249)

Note

This document uses the following terms:

- Upgrade: Denotes upgrading to a higher version of a component/software
- Downgrade: Denotes reverting to a previous version of a component/software
- Update: In general terms, denotes the act of bringing a component/software up to date (or, more specifically, an upgrade or downgrade)

Overview

The exact procedure when replacing or updating components depends on various factors.

When you update a project, how you proceed will depend on the which parts are affected by the version change.

- Change of the SIMOTION main version
- Change of the SIMOTION service pack or hotfix version
- Change of the PROFINET version
- Change of the SINAMICS version
  (There are SIMOTION versions that contain several SINAMICS versions for a device.)

If another SIMOTION controller is to be used, then the procedure depends on whether a device or a platform replacement is required.

Examples of upgrade scenarios are listed in the following overview table. They are shown in the columns. The lines list the principle measures that have to be performed. Whether the measure

has to be performed in a specific case, must be decided project-specifically.
Grayed-out cells mean that a measure is not required.

Note

If the version is changed and the SIMOTION controller replaced at the same time, then all measures apply; the measures must be performed in the TOP-DOWN sequence according to the table.

Action/Measure	Upgrade project				Replacement of SIMOTION controller		Activity relates to
	Main version	Service pack or hotfix version	PROFINET version	SINAMICS version	Device replacement via HW Config	Platform replacement via XML export/import	
Customize project							
Examples	V4.1 ⇨ V4.2	V4.1 SP2 ⇨ V4.1 SP4	PN2.1 ⇨ PN2.2	V2.5 ⇨ V2.6.2	D445-2 ⇨ D455-2 D4x5 ⇨ D4x5-2 D410-2 DP ⇨ D410-2 DP/PN	C240 ⇨ D445-2 D410-2 ⇨ D445-2 D445 ⇨ D445 (S120) (SM150)	Project/SIMOTION SCOUT
Create backup copy (project/CF)							
Back up user data (back up variables)	Only if required				Only if required	Only if required	
Upgrade user project to new SCOUT version							
Platform replacement via XML export/import	During platform replacement, the target version is specified (= version of the device that is the recipient of the import)						
Prepare device replacement							
Device replacement in HW Config							
Upgrade technology packages (TP)	[4]	[4]			[4]		
Upgrade device version of SINAMICS S120 CUs [3]				For external CUs, only if necessary			
Upgrade library	Libraries are version-dependent				Libraries may be device-dependent	Libraries may be device-dependent	
Save/compile project; check consistency							
Perform a firmware and/or project update							
Upgrade the boot loader on the CompactFlash card	Check whether new version requires a new boot loader [2]				Check whether the new [2] device requires a new boot loader		Memory card/Target hardware
Update - preparatory measures							
Upgrade via IT DIAG	Selection of one of the 3 update methods				Only possible if the CF card contains valid firmware		
Update via device update tool							
Update via CompactFlash Card							
Backup of the CompactFlash card data							
Firmware update using a CompactFlash card							
Upgrade SINAMICS [5]							
Download the project to the target system [1]							

Not relevant Relevant

1) Alternative: Load the project to the CF card using a card reader

2) See SW compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317)

3) The versions of SINAMICS Integrated and Controller Extensions are upgraded automatically in the HW Config during device replacement

4) The technology packages are updated automatically. The user can directly specify a TP

5) SINAMICS components are upgraded/downgraded to the component version of the CF card. Ensure that LED codes are observed. After the upgrade it is necessary to switch the device off and on.

Figure 10-122 Overview of upgrade options

Replacing modules (spare part scenario)

Parts replacement for SIMOTION D4x5-2

Kernel/firmware ≥ V4.3

As of SIMOTION V4.3, a module replacement is detected by the controller on the basis of the serial number. This automatically deletes the non-volatile data and transfers the data backed up on the CF card to the controller. See Section Replacing modules in the spare part scenario (Page 6998).

Kernel/firmware V4.2

If a used D4x5-2 is used for the module replacement, we recommend that the non-volatile SIMOTION data be deleted on the used D4x5-2.

Two options are available for this.

Table 10-64 Options for deleting non-volatile SIMOTION data

Option	Reference to description	Notes
Module memory reset	Memory reset of SIMOTION D4x5-2 (Page 7186)	Controller via mode selector (position "3" MRES). Is available on all SIMOTION platforms.
Deleting and restoring non-volatile SIMOTION data	Overview (Page 7274)	Controller via Service switch (position "1"). Is only available on SIMOTION D.

NOTICE
Unintentional restoration of the default settings instead of memory reset
With a Power OFF/ON, "Restore default settings" is performed instead of "Memory reset" in switch position "3" (MRES) of the mode switch.
To avoid and unintentional/accidental deletion of the project data on the CF card, we recommend that the non-volatile SIMOTION data be deleted using the service selector switch. Restore default settings deletes all the project data on the CF card!

See also

Operations and their effect on the user memory (Page 6990)

Removal and replacement of the SIMOTION D4x5-2

Removing the Control Unit

NOTICE
Damage to property can occur Installing or removing SIMOTION D410 with power connected can result in undefined states in your system. Damage to property can occur in your automation solution as a result. For this reason, only ever install or remove SIMOTION D410 with the power source disconnected. When configuring a system therefore always comply with the relevant standards and safety regulations, which are mandatory in any case.

Proceed as follows to remove the SIMOTION D4x5-2:

1. Switch off the power supply.
2. Remove the front cover and remove the CompactFlash card from the plug-in slot.
3. Disconnect the terminal block for the power supply.
4. Disconnect the DRIVE-CLiQ connectors to the SINAMICS S120 modules, the connectors of the PROFIBUS DP interfaces (X126 and X136), the Ethernet connectors to the X127, X120 and X130 interfaces (X120 only for D4x5-2 DP) and, if necessary, the connectors on the PROFINET X150 interface (X150 only for D4x5-2 DP/PN).
5. Remove the connectors to the digital inputs/outputs at interfaces X122, X132 and X142.
6. Disconnect any option module which may be inserted (CBE30-2, TB30).
7. Remove the mounting screws of the control unit.

Installing a new module

Proceed as follows to install a new SIMOTION D4x5-2:

1. Remove the front cover from the new Control Unit.
2. Mount the new Control Unit.
3. Reconnect all connectors that were removed previously.
4. Connect the terminal block for the power supply.
5. Reattach the designated shielding for all cables.
6. Reinsert the original CompactFlash card into the plug-in slot.
7. Reattach the front cover and close it.
8. Switch on the power supply. The Control Unit is immediately ready to operate.

SIMOTION D4x5-2 module replacement without PG/PC

To enable a module replacement without a PG/PC, you must back up the current non-volatile SIMOTION and SINAMICS data on the CF card during operation.

NOTICE
Loss of data due to failure to make backup copies
Non-backed-up non-volatile SIMOTION data can be lost on hardware replacement (module defect), for example, if the current value of the retain variables have not been backed up and replaced by their initial values again.
Back up the non-volatile SIMOTION data on the CF card.

NOTICE
Repeat referencing necessary after absolute encoder overflow
If an absolute encoder overflow occurs after **_savePersistentMemoryData**, the actual position value is no longer correct after the non-volatile SIMOTION data are restored.
In this case, homing (absolute encoder adjustment) must be repeated.

See also

Parts replacement for SIMOTION D4x5-2 (Page 7213)

Operations and their effect on the user memory (Page 6990)

Replacing modules in the spare part scenario (Page 6998)

Replacing DRIVE-CLiQ components

Module replacement

DRIVE-CLiQ components not only support replacement in the switched-off state of the machine/system (Power Off), but also a replacement during operation. For this, the component to be replaced must be at the end of the DRIVE-CLiQ line.

Requirement

- The affected components are located at the end of the line.
- If an infeed is affected, the power units supplied by it do not function.

Procedure for "Removing a component"

1. Deactivate the affected component or the drive object.
2. Remove the DRIVE-CLiQ connector.
3. Remove the supply voltage of the component and uninstall the component.

Procedure for "Installing a component"

1. Install the component and reconnect the supply voltage.
2. Reconnect the DRIVE-CLiQ cable at the same location (port). The cable must have the same length as the old one.
3. Activate the affected component or drive object.

Parameters for topology comparator and component replacement

In the expert list, you can use CU parameter p9906 to specify how the electronic rating plates are compared for all the components of a Control Unit. The type of comparison can be changed subsequently for each individual component by using p9907/p9908 or right-clicking in the topology. All data on the electronic rating plate is compared by default.

- If p9909 = 1, the serial number and the hardware version of the new replaced component will be transferred automatically from the actual topology to the target topology and then saved to non-volatile memory.
- When p9909 = 0, serial numbers and hardware versions are not automatically transferred.

The setting p9901 = 1 enables the **spare parts / components replacement without tool support** to be carried out. The new serial number of the spare part is automatically transferred from the actual topology to the target topology and saved in non-volatile memory. Prerequisite is that the replaced components are of the same type and have the same article number, e.g. "6SL3055-0AA00-5BA2". The last or last two digits of the article number (depending on the component type) are not checked, as the HW version, for example, is coded in these. This mechanism is also applied when several components are replaced.

Modified wiring following module replacement

In the default setting for the topology comparator, modified wiring configurations of DRIVE-CLiQ components (e.g. cross-exchange) cannot be accepted for safety reasons and a fault is generated.

If a cross-exchange of components is required (i.e. existing components are replaced with other existing components, and no spare parts are used), e.g. for troubleshooting purposes, the topology comparator must be reduced via parameter p9906, or preferably via p9907/p9908; alternatively, by right-clicking in the topology.

Note

In this case, incorrect insertion of components is no longer monitored.

Automatic upgrading/downgrading (firmware update)

When starting up, the system automatically upgrades or downgrades DRIVE-CLiQ components to the version of the component firmware on the CF card. Components that cannot be downgraded to the component firmware version on the CF card (e.g. old firmware on the CF card and new components to which the old firmware cannot be loaded) retain their firmware version. The resulting firmware version combinations are always functional.

Note

During an automatic firmware update, please read the messages and alarms in the SIMOTION SCOUT detail window.

Note

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are displayed additionally via a yellow RDY LED on the SIMOTION D/CX32-2, indicating that components connected to SIMOTION D/CX32-2 are carrying out a firmware update or that all components have finished their firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

The upgrade/downgrade function can be deactivated using the p7826 CU parameter in the expert list.

The component version can be obtained from the CONTENT.TXT file in the main directory of the CF card.

Additional references

You can find further information on this topic:

- In the *SINAMICS S120* Commissioning Manual
- In the *SINAMICS S120* Function Manual

Replacing the CompactFlash Card

If spare parts are being used, you must contact the Technical Support Center to convert the license key of the defective CompactFlash card to the new CompactFlash card. Proceed as described in the following sections to write your project to the new CompactFlash card:

- Changing the CompactFlash card (Page 7245)
- Writing to a CompactFlash card (Page 7246)

Detailed information about licensing can be found:

- In the *SIMOTION SCOUT* Configuration Manual
- As well as in the FAQ (https://support.industry.siemens.com/cs/ww/en/view/73501913)

Replacing the fan/battery module

Overview

The procedure for replacing the double-fan/battery module is described below.

Procedure

Proceed as follows to replace the double-fan/battery module:

1. Press the latch. This detaches the module from its front latching device.

Figure 10-123 Unlatching the fan/battery housing

2. Tilt the double-fan/battery module forwards at an angle and pull out the plastic lug from the Control Unit cutout.
3. Now take the double-fan/battery module and hold at an angle to the front with the open side facing up (battery visible).
4. Push the plastic lug into the cutout on the lower side of the Control Unit.
5. Tilt the double-fan/battery module up until the front latch snaps into place.
   The electrical connection between the double-fan/battery module and the Control Unit is made automatically.

Note

The fan/battery module should preferably be replaced in the POWER OFF state.

Replacement during POWER ON

The fan/battery module should preferably be replaced in the POWER OFF state. In principle, replacement during POWER ON is possible, but the following aspects must be taken into consideration:

- If the fan/battery module is disconnected and heat dissipation is required because of the temperature conditions, a fan fault is signaled (fan fault is signaled via system variable, PeripheralFaultTask and diagnostic buffer entry; if no PeripheralFaultTask is configured, the Control Unit switches to the STOP state)
- For every SIMOTION D4x5-2 with a fan fault, the Control Unit switches to RESET after approximately 1 minute to protect itself
- During the replacement, the missing fan and battery is signaled via the fanbattery.fanexisting and fanbattery.batteryexisting system variables

NOTICE
Unintentional failure of the machine/system Only replace the fan/battery module when the CPU is in the STOP state so that there is no risk of an unintentional failure of the machine/system.

The causes of an unintentional failure can be, for example:

- No PeripheralFaultTask has been configured; if no PeripheralFaultTask is configured, the Control Unit switches to the STOP state when a fan fault occurs
- A delayed replacement can result in a RESET state
- Cables may be unintentionally disconnected during the replacement

Fan service life

Fans are parts subject to wear. The fans are monitored in the SIMOTION D4x5-2. A failure is signaled in the diagnostic buffer and can be evaluated by the user program (e.g. via the PeripheralFaultTask).

Contamination is the main cause of fan failure. A visual inspection should be made as a first criterion for replacement. If the fan is highly contaminated, it must be replaced.

If no contamination is present, we recommend that the service period is used as criterion. Because the service life of the fans depends greatly on the operating conditions (temperature, humidity, number of operating hours per day, contamination caused by dust, etc.), no fixed limits for all applications are can be specified. In the field, under average industrial conditions, a replacement interval of 5 to 7 years has proven itself.

We recommend that the maintenance intervals are adapted for the specific application based on empirical values.

Replace battery in the fan/battery module

Overview

The procedure for replacing the double-fan/battery module's battery is described below.

Procedure

Proceed as follows to replace the battery:

1. To disassemble the double-fan/battery module, proceed as described in section Replacing fan/battery module (Page 7218).
2. Remove the battery using a screwdriver (at the side) and disconnect the battery from the module by unplugging the connector.
3. Connect the cable connector of the new battery to the mating connector in the fan/battery module and push the battery in.

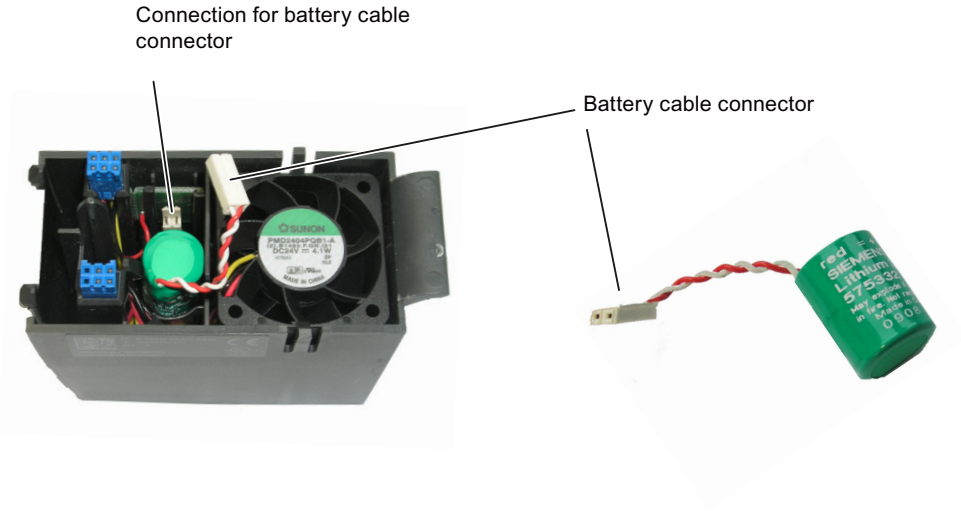


Figure 10-124 Battery replacement

4. To reassemble the double-fan/battery module, proceed as described in section Replacing fan/battery module (Page 7218).

Note

The fan/battery module should preferably be replaced in the POWER OFF state.

The battery should be replaced at least once every 3 years, at the latest when the prewarning level is reached (see SIMOTION D4x5-2 Commissioning and Hardware Installation Manual).

Later replacement of the battery can result in a backup failure of the RTC (real-time clock) and therefore loss of the date and time!

Rules for handling backup batteries

⚠ WARNING

Risk of ignition, explosion, and combustion

Incorrect handling of backup batteries can result in a risk of ignition, explosion, and combustion.

The regulations of DIN EN 60086-4, in particular regarding avoidance of mechanical or electrical tampering of any kind, must be complied with.

- Do not recharge
- Do not throw into a fire
- Do not solder on the body of the cell
- Do not open
- Only replace with identical types
- Only obtain replacements through Siemens (see *SIMOTION D4x5-2 Equipment Manual, section: Available spare parts and accessories*)

For information on transport and storage of backup batteries, see section Transport and storage conditions (see *SIMOTION D4x5-2 Equipment Manual, section: Transport and storage conditions*).

Notes on disposal

Dispose of used batteries in the specially provided collection points on site. This will ensure that the batteries are recycled in the correct manner or treated as special waste.

Adapting a project (Upgrading the project / Replacing the SIMOTION controller)

Overview

The project needs to be adapted if you want to replace the type (e.g. D445-2 DP/PN ⇒ D455-2 DP/PN) or version of the SIMOTION device in your existing project.

Procedure

The exact procedure for project adaptations depends on the scope of the target hardware and version changes.

An overview of the different applications can be found in Figure 10-122 Overview of upgrade options (Page 7212).

See also

Overview (Page 7209)

Creating backup copies (project/CF)

Requirement

Before adapting the project, it is essential that you create the following backup copies:

- A backup copy of the project and
- A backup copy of the contents of the CF card, see Backup of the CompactFlash card data (Page 7239)

Backing up user data (backup variables)

Overview

With the SCOUT functions "Save variables" and "Restore variables", you have the option of backing up and restoring data that were changed during operation and only stored in the runtime system. This is necessary if a SIMOTION platform is changed or a version upgraded, for example.

The "Save variables" function creates XML files which are stored in a folder of your choice.

The following types of data can be backed up:

- Retentive global device variables and unit variables, as well as TO retain data (as of V4.1), located in the SRAM or NVRAM depending on the controller
- Data saved with **_saveUnitDataSet** or **_exportUnitDataSet** and located on the CF card.

Note

When performing an upgrade with a firmware version that is ≥ V4.1, this function is only required for backing up and restoring unit data sets that have been created using **_saveUnitDataSet**.

Retain and unit data (saved with **_exportUnitDataSet**) remain valid even after a version upgrade.

SIMOTION retain data can also be easily backed up to a memory card without the use of SIMOTION SCOUT. For this purpose, use:

- The **_savePersistentMemoryData** function or
- The service selector switch or DIAG button on the SIMOTION D4x5-2 or SIMOTION IT web server
  See also Section Backing up diagnostic data and non-volatile SIMOTION data (Page 7266).

Procedure

The backup of the user data must be performed **before** the upgrade of the SCOUT project. This is possible for a version upgrade with the "old" SCOUT version or the "new" SCOUT version.

The following is a description of the procedure with a "new" SCOUT.

1. Open the project.
   When you open the project, a window appears with a message that the project you are opening was created with a different SCOUT version, as well as a query asking whether you want to perform an upgrade.

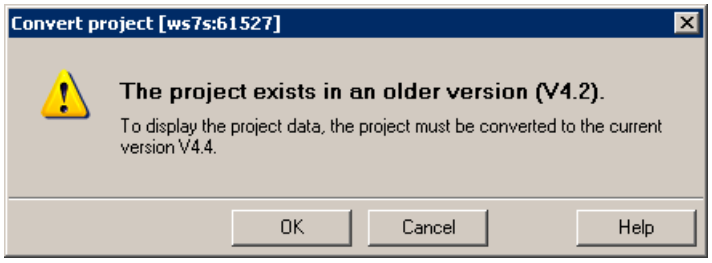


Figure 10-125 "Convert project" message

2. Confirm the prompt with "OK". The project will be converted to the current version.
3. A dialog then appears with a prompt asking whether the project should be opened write-protected.

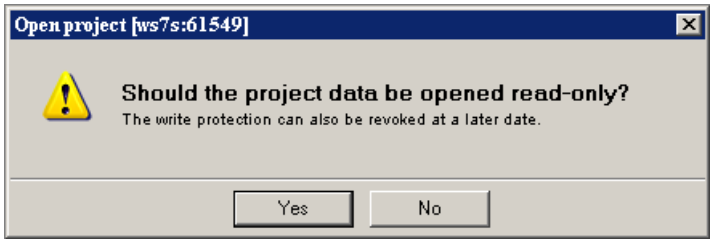


Figure 10-126 Prompt for write-protected opening

4. Confirm this prompt with "Yes" (open write-protected).
5. Set the SIMOTION D4x5-2 to the STOP operating state.
6. Set the SIMOTION D4x5-2 to online mode and perform the SCOUT **Save variables** function. The retain variables (interface and implementation) and the user files (with _saveUnitDataSet or _exportUnitDataSet) are saved to the PG/PC.
7. Then close the project.

Note

An online connection is possible only when the PG/PC is configured for the controller.

Update the PG/PC assignment using .

Online connection is now possible.

Additional references

For further information, see the *SIMOTION SCOUT* Configuration Manual.

Upgrading a user project to the new SCOUT version

Requirement

It is essential that a backup copy be made of the original project before the upgrade, because the data storage of the project is also upgraded during the upgrade. This ensures that you can always return to the original project if the upgrade fails (power interruptions, unexpected faults, incorrect operation, etc.).

Procedure

1. When opening the project a window appears with a message that the project to be opened was created with another SCOUT version, as well as a prompt as to whether the upgrade should be performed.
   Confirm the prompt with "OK".

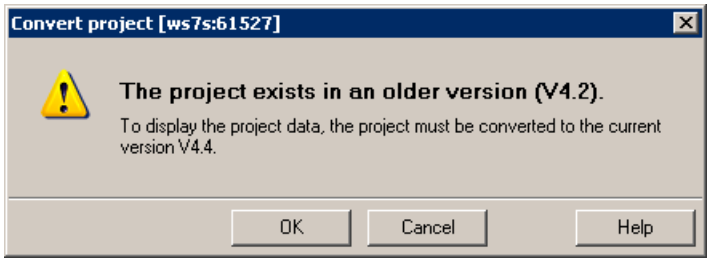


Figure 10-127 Prompt as to whether the project should be upgraded

2. After the conversion another prompt appears as to whether the project should be opened write-protected. Confirm this prompt with "No" for a version upgrade (do not open write-protected).

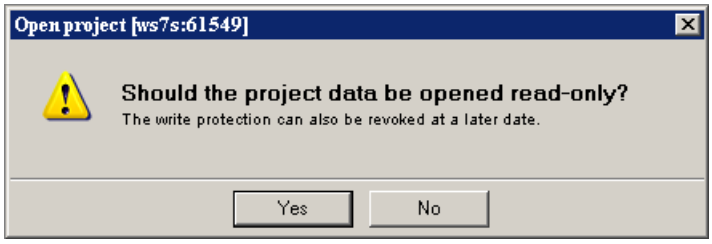


Figure 10-128 Prompt as to whether the project should be opened as write-protected

Note

A project that was last edited with a higher SCOUT version cannot be opened by a SCOUT with a lower version. However, the project with the more current SCOUT version can first be converted to the required software version (command: "Project" > "Old project format" > "Save in old project format"). The project can then be opened with the lower SCOUT version.

Platform replacement via XML export/import

Overview

A platform replacement is always required when an existing project is to be used for another SIMOTION platform. The platform replacement is always performed via an XML export/import.

The following devices can be interchanged via a platform replacement:

- Replacement between SIMOTION C, P, and D (e.g. C240 → D445-2 DP/PN)
- Replacement between D410 and D4x5/D4x5-2 (e.g. D410-2 → D445-2 DP/PN)
- Replacement between SIMOTION D (SINAMICS S120 Integrated) → SIMOTION D (SINAMICS SM150 Integrated)

Platform replacement during project downgrades:

It is not possible to downgrade to a lower SINAMICS version. However, it is possible to transfer project data by means of an XML export/import.

Preparations

Before the platform replacement can be performed, preliminary work may be necessary in the existing project.

If a D4x5-2 is to be imported into a D410-2, then only the permitted quantity structure of the D410-2 may be configured in the D4x5-2. This applies for all components, e.g. not only an infeed, but also the permissible power units.

A D4x5-2 with CU adapter and blocksize Power Module can be imported into a D410-2 when the CU adapter is connected to port 0. Otherwise, the topology will be destroyed.

Generally, the success of an import always also depends on the specific configurations of the drive units, and whether the configuration is possible for the device to which the import is to be performed. Also note any error messages that may occur.

Procedure

Proceed as follows:

1. In the project navigator of SIMOTION SCOUT, right-click the SIMOTION controller that is to be replaced.
   Select "Expert" > "Save project and export object" in the context menu.
   "Save project and export object" exports selected data of the selected object in XML format. This data export can then be reimported into other projects. The entire project is not exported, only the data of the selected object (e.g. only the D4x5-2 or only the SINAMICS Integrated).
2. Specify the desired path and start the XML export.
3. When the export has been performed error-free, delete the device from the project and confirm the prompt.
4. Insert the desired platform as new device in the project navigator of SIMOTION SCOUT. With the selection of the device, you also define the SIMOTION version, and with a SIMOTION D, also the SINAMICS version.

5. Import the data of the original platform into the new device. To do this, right-click the new device and select "Expert" > "Import object" in the context menu.
6. Select the location where the XML export data is to be stored and start the import. Confirm the prompt to continue with the import.

Confirm the message with regard to the import of a "non-compatible type" with "OK".

Preparing the device replacement

Overview

The device replacement process differs from the platform replacement process in that it is really easy to accept project data during device replacement.
The device replacement is performed via **HW Config**, whereas an **XML export/import** is required for a platform replacement.

A device replacement is only possible within SIMOTION D.

The following devices can be interchanged:

- Replacement between different performance classes (e.g. D445-2 DP/PN ⇔ D455-2 DP/PN)
- Replacement between generations (D4x5 ⇒ D4x5-2)
- Replacement between variants (D425-2 DP ⇔ D425-2 DP/PN)
- Replacement between SIMOTION, SINAMICS and/or PROFINET version (e.g. D425 V4.1 - PN V2.1 SINAMICS S120 V2.5 ⇒ D425 V4.2 - PN V2.2 SINAMICS S120 V2.6.2).

One SIMOTION D can only be replaced with another SIMOTION D if the SINAMICS version involved is the same or higher. It is not possible to downgrade to a lower SINAMICS version.

Preparation

Before the device replacement can be performed, it is recommended that preliminary work be performed in the existing project. The existing project must be adapted so that it can be mapped to the new device during the device replacement.

Examples:

1. Number of DRIVE-CLiQ interfaces
   If the replacement is to a device with fewer DRIVE-CLiQ interfaces, then the ports that are no longer available must be rewired before the device replacement (double-click "Topology" in the project navigator). Components on DRIVE-CLiQ ports that are no longer available are moved to the component archive during the device replacement.
   Example:
   Replacement of D445-2 DP/PN (six DRIVE-CLiQ ports) with D425-2 DP/PN (four DRIVE-CLiQ ports)
   → ports X104 and X105 are no longer available.
2. Number of CX32-2s
   If the replacement is with a D4x5-2 that only supports a small number of CX32-2 modules, then a device replacement in **HW Config** is not possible.
   In this case, the CX32-2 must be replaced by a SINAMICS S120 CU320-2.
3. PROFINET interface
   If a D4x5 with CBE30 is replaced by a D4x5-2 DP/PN with onboard PROFINET interface, then the number of PROFINET ports is reduced from four to three.
   Components on port 4 have to be rewired

TIP for synchronous grouping

If you use device-dependent ST libraries, then the upgrade can be accelerated through the following procedure:

1. Upgrade the device version for all devices that participate in the synchronous grouping in the ST libraries
   (example: Change D445 V4.1 to D445-2 V4.2).
   See also Section Upgrading a library (Page 7232).
2. Save the project and close it in SIMOTION SCOUT.
3. Perform the module replacement for all devices of the synchronous grouping in **HW Config** for this project and then save.
4. Open the project in SIMOTION SCOUT.
   The entire synchronous grouping is now completely upgraded.

Device replacement in HW Config

Procedure

1. Double-click the SIMOTION device to be replaced in the project navigator in SIMOTION SCOUT. **HW Config** opens.
2. Open the "SIMOTION Drive-based" folder in the hardware catalog.

   Note

   SIMOTION D is modeled as a compact device in **HW Config**. When modules are being replaced, this means the new module must be moved to the header of the module rack shown and **not to slot 2**. Please make sure that you do not delete the D4x5-2 rack!

   When you move the new module to the rack header using drag-and-drop, the old module will be replaced. Alternatively, you can:

   - Select the rack header and double-click the new module in the module catalog to replace the previous module, or:
   - Right-click the rack header and select "Replace Object"

3. Move the new module to the top field of the module rack using drag-and-drop.
4. Confirm the dialog box that appears with "Yes" if you want to replace the SIMOTION device. The module is replaced.
5. Accept the changes made to the hardware configuration with "Station" > "Save and compile".
6. Close **HW Config**.

   Note

   During the module replacement, the following actions are performed automatically by the engineering system (if required):

   - Update of the technology packages (TPs)
   - Automatic upgrade of the SINAMICS Integrated
   - Automatic upgrade of all connected CX32-2

   The updated data is transferred to the project and the entire project saved.

If the module hardware changes (e.g. a D445-2 DP/PN is replaced with a D455-2 DP/PN), proceed as described in Section Removal and replacement of the SIMOTION D4x5-2 (Page 7214).

Upgrading technology packages

Overview

The SIMOTION technology packages (e.g. TP CAM, TP PATH, DCBlib) are available in various versions.

You can only use the functions of the technology objects selected if the technology objects are available in the target system. You can select the technology packages and their version for each SIMOTION device. Each version of SIMOTION SCOUT has a kernel (FW version) for the SIMOTION CPU and an appropriate technology package.

TPs during upgrades

Device replacement (in **HW Config**), platform replacement (XML export/import), or even upgrades may cause versions of SIMOTION technology packages (TPs), which are assigned to individual technology objects (TOs), to change.

- The TP version may change when the main version is changed.
  The TP version depends on the relevant main version in all cases; it may, however, remain unchanged through a number of main versions.
- If service packs and hotfixes are installed, there may even be a selection of TP product versions available for the same TP version.

During device replacement (in HW Config) the TP version is automatically updated. If the TP version is changed and more than one version of the new TP version is available, the latest version is automatically installed. If another product version is preferred, this must be set manually (e.g. selection of V4.1.5.3).

With a platform replacement (XML export/import), the required technology package along with the TP version and, if necessary, the product version have to be selected manually after the import.

When a SIMOTION CPU is inserted, the TP CAM (latest TP version and product version) is preset per default.

Special displays in the "Product version" field:

- "Select" means that no TP product version has been selected; this state occurs when older projects, in which the selection of a specific product version was not supported, are upgraded. If the project is loaded to the CPU without making a selection, the latest available technology package is loaded automatically.
- "---" means that no version can be determined (e.g. for the TP DCBlib or for older CPU versions < V4.1). If no version can be determined, you must select "---".

Selecting the TP product version

The desired technology package is selected with fine granularity in SIMOTION SCOUT at "Target device" > "Select technology packages ...".

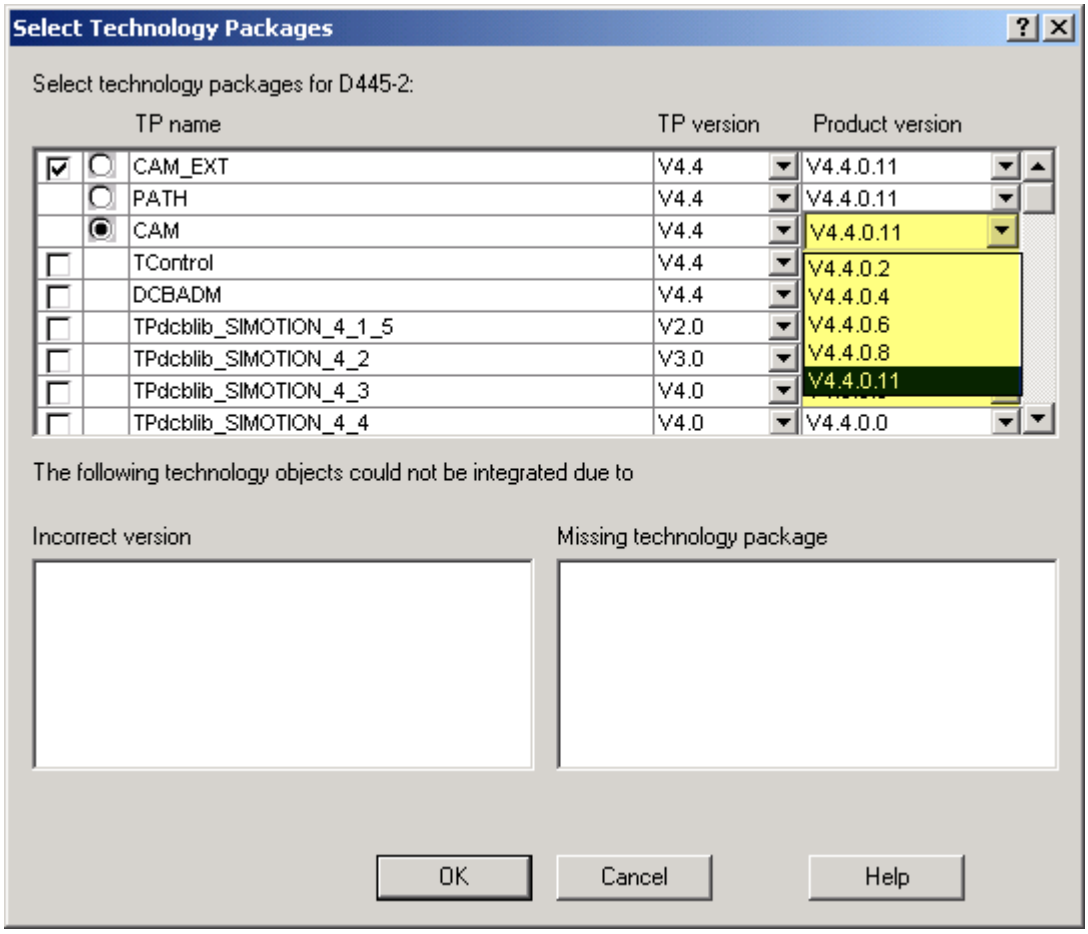


Figure 10-129 Selecting technology packages (example: D445-2)

Note

Device diagnostics can provide information on which technology package product version has been loaded to a CPU.

Loading technology packages to the target device

Technology packages are only loaded to the target device if no technology package has been loaded so far or if "Load to file system" is executed.

If a technology package version changes, the technology package must be explicitly reloaded to the target device. To do this, proceed as follows:

1. Select "Download project to target system" in SIMOTION SCOUT.
2. Select the "Replace product versions of the technology packages" option at "Additional CPU options" and confirm with OK.

For further information, please refer to the online help for SIMOTION SCOUT.

Upgrading the device version of SINAMICS S120 control units

Overview

You can upgrade the device versions of SINAMICS S120 control units that are connected to the SIMOTION D via PROFIBUS or PROFINET in the SIMOTION SCOUT. The SINAMICS version can only ever be upgraded in a project; it cannot be downgraded.

Note

During the device replacement in **HW Config**, the SINAMICS version

- Of the SINAMICS Integrated of the SIMOTION D4x5-2 as well as
- The connected CX32-2 controller extensions

are also upgraded automatically.

A CX32-2 always has the same SINAMICS version as the SINAMICS Integrated. During the module replacement in **HW Config**, the SIMOTION **and the SINAMICS version** is always defined with the selection of a D4x5-2 module.

If a SINAMICS S120 control unit is connected via PROFIBUS or PROFINET, the SINAMICS version can be selected independently of the SINAMICS Integrated version.

Procedure

To upgrade a SINAMICS drive unit:

1. Right-click the relevant device, e.g. the SINAMICS S120 CU320-2 DP.
2. Select "Target device" > "Upgrade device version/characteristic" in the context menu.
   The "Upgrade Device Version/Characteristic" dialog box is displayed. It lists all available firmware versions.

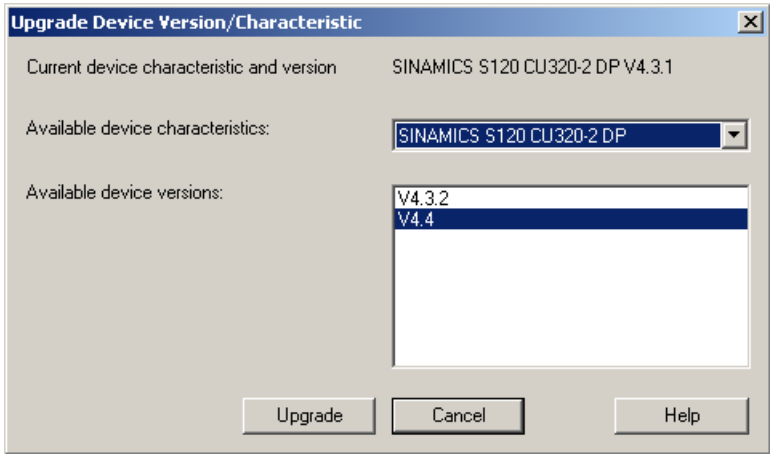


Figure 10-130 Upgrading the device version

3. Select the desired device version/characteristic and click "Upgrade".
   This upgrades the SINAMICS S120 control unit.

Upgrading a library

Depending on the configured properties of the libraries used in the project (device-dependent or device-independent), an upgrade of the libraries may be required if the SIMOTION device or the device version changes.

1. Open the LIBRARIES directory in the project navigator.
2. Select a library, right-click to open the context menu, and select "Properties...."
3. Select the "Technology packages" tab in the "Properties" window.
4. Select the SIMOTION device and the technology packages for which the library is to be valid.
5. Close the dialog box with "OK".

Note

Please also observe the information on device-dependencies in the SIMOTION SCOUT online help.

Save project, compile and check consistency

Procedure

1. Execute "Save and recompile all" ("Project" > "Save and recompile all" menu).
2. Then perform a consistency check ("Project" > "Check consistency" menu).
3. If error messages occur, correct these and repeat the operation.

Note

Note the difference between

- "Save and recompile all" and
- "Save and compile changes"

Save and recompile all

All sources of the entire project are recompiled with this command.

The command is suitable if you are quite sure that all the old data from older SCOUT versions should be removed and replaced with new compilation results.

Use this command if you specifically want to convert a project from an earlier SCOUT version to a newer version. In this way, you take over all error corrections and optimizations.

Save and compile changes

On this command, the whole project is searched for changes. Therefore only the changes are compiled. Use this command for day-to-day operations within a SCOUT version.

Performing a firmware and project update

Upgrading the boot loader on the CompactFlash card

The boot loader may have to be upgraded in the following cases:

- Upgrade of the SIMOTION D
- If a D4x5 CF card is to be used for a D4x5-2 (or vice versa).
- Error corrections and optimizations

Generally, we recommend that the latest boot loader available for the associated CF card be used.

Detailed information on the compatibility relationships for the CF card, boot loader version, SIMOTION D hardware and SIMOTION firmware version can be found in the software compatibility list. This list can be found on the Internet at (https://support.industry.siemens.com/cs/ww/en/view/18857317).

See also

Bootloader on the CompactFlash card (Page 7247)

Update - preparatory measures

Upgrading the SIMOTION D

The actions described in this section also apply to downgrading to an older version.

Various options are available for performing a firmware and/or project update on the SIMOTION D.

- Update via SIMOTION IT web server (Page 7235)
- Upgrade via device update tool (upgrading SIMOTION devices) (Page 7235)
- Update via CF card (Page 7239)

Note

Upgrading using the device update tool offers a number of advantages (keeping retain data, option of downgrading, no license key handling, etc.).

We therefore recommend using this method for firmware and/or project updates.

Requirement (firmware update)

The current firmware for SIMOTION D can be found at Internet address (https://support.industry.siemens.com/cs/ww/en/view/31045047).

Upgrading the SIMOTION D automatically upgrades the firmware of all connected SINAMICS DRIVE-CLiQ components.

Note

Observe the information in the the Read Me files and the upgrade instructions included in the scope of delivery of new SIMOTION versions.

Use only CF cards that have been released for SIMOTION D and have an appropriate and correct boot loader version.

The compatibility relationships can be found at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

NOTICE
The upgrade operation deletes all project data and parameters from the CF card! Back up the data before starting an upgrade.

Requirement (project update)

You have upgraded your project and, if necessary, adapted the device type and device version, see section Adapting a project (Upgrading the project / Replacing the SIMOTION controller) (Page 7221).

Update via SIMOTION IT web server

The SIMOTION D features an integrated web server.

In addition to customized web pages and comprehensive device/diagnostic information, SIMOTION IT web server also allows you to update the firmware and project using a standard PC with a web browser.

As of firmware version V4.2, it is no longer necessary to purchase licenses for the web server (IT DIAG license) and for SIMOTION IT OPC XML-DA.

You will find detailed information in the
SIMOTION IT Diagnostics and Configuration Diagnostics Manual.

See also

Upgrade via device update tool (upgrading SIMOTION devices) (Page 7235)

Upgrade via device update tool (upgrading SIMOTION devices)

Overview

SIMOTION D Control Units and projects can be upgraded using previously created upgrade data.

Performing an upgrade using upgrade data has the following advantages:

- User-friendly creation of upgrade data via SIMOTION SCOUT with the aid of a wizard (at the machine manufacturer's site)
- SIMOTION devices can be upgraded by the machine operator without the SIMOTION SCOUT Engineering System.
- The machine manufacturer can conveniently send upgrade data via e-mail or post to the machine operator
- There is no need to use license keys, as licenses are retained
- Retain data and unit data is retained when upgrades are performed, even across versions
- An upgrade which has been imported can be discarded again, and the previous configuration restored
- You can upgrade either a single SIMOTION device or multiple devices from one or more SIMOTION projects.
- It is possible to upgrade parts of a configuration only (e.g. Technology Packages only, firmware only, project only, etc.).

Handling

Upgrade data is created by the application engineer at the machine manufacturer's premises using SIMOTION SCOUT. The upgrade data can then be handled flexibly depending on both the SIMOTION device in question (SIMOTION C, D, or P) and the customer requirements:

- Creating upgrade data and then copying it to a storage or upgrade medium:
  - CF card
  - USB stick or
  - Upgrade file for the SIMOTION IT web server
- Alternatively, the upgrade data can be created and stored in an archive on the PC, with a view to importing it to an upgrade medium suitable for SIMOTION devices at a later time.
- The process of importing the data to an upgrade medium can be performed at the machine manufacturer's premises; alternatively, if the upgrade archive has been transferred to the machine operator, the service engineer can do this on site.
- The service engineer imports the upgrade data on an operator-guided basis (without any involvement by the application engineer) to the SIMOTION device(s), and upgrades the SIMOTION devices in the process (SIMOTION SCOUT is not required on-site).

The following describes how to upgrade a SIMOTION D with a USB stick. This assumes you have a USB stick containing the appropriate upgrade data.

Requirement

You have a USB stick containing the upgrade data.

Note

When upgrading using a USB stick, the points listed below must be observed. The version information refers to the firmware version on the CF card (that is, the original version from which upgrading is to take place).

- Version V4.2: Upgrading via a USB stick is not supported.
- Version < V4.3 SP1 HF12: The USB stick may only be inserted when the Control Unit is switched off.
- Version ≥ V4.3 SP1 HF12: The USB stick can be inserted when the Control Unit is switched on or off.

Procedure

When using a USB stick to perform an upgrade, proceed as follows:

1. Check the position of the service selector switch (upper rotary switch SVC/NCK). The switch must be in the "0" position.
2. Insert the USB stick in one of the two USB interfaces of the activated D4x5-2 (only one USB stick may be inserted).
3. Switch the device OFF/ON or reset it using the RESET button.

4. SIMOTION D4x5-2 will now begin copying the data from the USB stick to the CF card. Copying begins with some LED state changes until finally a yellow/green flashing RDY LED is displayed (0.5 Hz). In the end phase of the upgrade, the RDY LED has longer steady green or yellow phases (< 10 s).
   Once copying is complete, the RDY LED will change to:
   - "Steady green" if the procedure has been completed successfully (the copying procedure has been completed successfully when the RDY LED is constantly green > 10 s).
   - "Steady red" if copying was not successful.

Meaning of the 7-segment display	
0	No memory card available.
1	Internal error
2	Internal error
3	No or incomplete update data available on the USB memory stick.
4	No update data can be copied because there is no project available in the SIMOTION device, but update data for several SIMOTION devices is available on the USB memory stick. Because of the missing project on the SIMOTION device, assignment to the data on the USB memory stick is not possible.
5	The update data cannot be copied, for example, because there is not sufficient memory space.
6	Internal error
7	Unknown update data available on the USB memory stick.

5. Switch the D4x5-2 off and remove the USB stick.
6. Switch the D4x5-2 on again. The D4x5-2 now begins with the actual upgrade. The SF LED flashes green (0.5 Hz) during the upgrade. The procedure can take several minutes.
7. Observe the green flashing of the SF LED.
   - As soon as the update procedure has been completed successfully, the SF LED is extinguished. An automatic power-up with the updated configuration is then performed (SF/BF LED display then depends on the respective operating state of the device).
   - If the upgrade was not successful, the SF LED flickers red.

If the upgrade was not successful "from the point of view of the application" (for example, the machine is not behaving as desired), it can be undone as follows:

1. Switch the D4x5-2 off.
2. Turn the upper rotary switch (SVC/NCK) to position "B".
3. Switch the D4x5-2 on again.
   The flickering green SF LED indicates that restoration is requested.
4. Set the service selector switch back to position "0."
   The system restores the data saved during the upgrade. The data from the upgrade will be deleted.

The restoration is indicated by a green flashing SF LED (0.5 Hz) and can take several minutes.

After successful restoration, the module starts up automatically. (SF LED display then depends on the respective operating state of the device.)

If the restoration procedure was not successful, the SF LED flickers red.

Note

For SIMOTION < V4.4, restoration will start when the module is switched on (step 3). For this, the service selection switch must be rotated to position "0" immediately when the flashing code is displayed (SF LED flashes green with 0.5 Hz).
If the service selector switch is not reset or not reset in good time to "0," this results in fault state "Service selector switch is still set to restore" (SF LED flickers red).
In this case, switch the D4x5-2 off, reset the service selector switch and switch the D4x5-2 on again. If the restoration was otherwise successful, the D4x5-2 boots up with the restored configuration.

Note

In the case of a CPU update via USB stick the SIMOTION D4x5-2 is booted from the USB stick. A bootable USB stick must therefore be used. Due to the rapid developments within the market for USB sticks, it is not possible to recommend any specific devices. SIMATIC USB sticks are an exception. Information on this is available on the Internet (https://support.industry.siemens.com/cs/ww/en/view/32580863).

Note

If the firmware for SIMOTION D is being upgraded or downgraded, the component firmware will also be upgraded or downgraded automatically depending on the firmware version on the CF card and the firmware version on the SINAMICS components (DRIVE-CLiQ components, CBE30-2, TB30, Power Modules, etc.).

The upgrade can take several minutes and is indicated by LED displays.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- Firmware update running: RDY LED flashes slowly (0.5 Hz)
- Firmware update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2 and indicate that a firmware update is in progress on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components that require a POWER ON after the firmware update will indicate this through a rapidly flashing RDY-LED. Go offline with SCOUT and switch the 24 V supply off/on (POWER ON) at the respective components for initialization.

CBE30-2 option board:

During the firmware update, the OPT LED of the SIMOTION D module and the SYNC LED of the CBE30-2 flash green.

Additional references

You will find detailed information on "upgrading devices" in the *Upgrading SIMOTION Devices* Operating Instructions.

See also

Upgrading the boot loader on the CompactFlash card (Page 7233)

Backup of diagnostic data and non-volatile SIMOTION data (Page 7266)

Update via CompactFlash card

Backup of the CompactFlash card data

Backing up licenses, retain data, and user data

Prior to upgrading/downgrading, as a precaution we recommend backing up the entire contents of the CF card to the PG/PC using the card adapter and Windows Explorer.

How you should proceed when backing up and subsequently restoring data on the CF card depends on whether licenses and/or other retain data and user data that will be required again in the future are located on the CF card.

Case 1: The CF card contains no licenses and no retain data or user data that will be required in the future

You do not need to take any steps in this case. Delete the contents of the CF card and install the firmware as described.

Case 2: The CF card contains licenses (e.g. axis licenses)

Before loading the new firmware, back up the "KEYS" directory to your PC. This can then be copied back to the CF card once you have installed the new firmware.

Note

The license key is stored in the "KEYS" directory on the CF card. When the SIMOTION device starts up for the first time, the license key is saved in the boot sector of the CF card.

A license key saved in the boot sector cannot be deleted by means of a user operation; nor can it be deleted by formatting the CF card or rewriting the boot loader.

If the Keys.txt file is no longer present on the CF card (because the "KEYS" directory has been deleted, for example), it will be written again from the boot sector to the "KEYS" directory while the SIMOTION device is starting up. The license key can be changed at any time (by relicensing, for example). When the device is next started up, the license key will be saved in the boot sector again.

In the event of the loss of a license key, a copy can be obtained via the Web License Manager at the following Internet address (http://www.siemens.com/automation/license). You will need the hardware serial number printed on the CF card. In the Web License Manager, you have the option of displaying the associated license key.

Case 3: The CF card contains retain data / user data that is still required in the future

If you are using your application to back up data to the CF card, you must back this up before upgrading the new firmware.

Example:

- Backup of the retain data
  - user\simotion\pmemory.xml
    For non-volatile SIMOTION data backed up with **_savePersistentMemoryData**
  - user\sinamics\nvram\pmemory.acx or \nvram\xx\pmemory.acx
    For non-volatile SINAMICS data (Integrated or CX32-2) backed up with p7775
- Backing up SIMOTION IT user files, settings (e.g. trace.xml), task trace data, log files, and Java files (classes, archives, user file system, etc.), stored in the following directories:
  - user\simotion\hmicfg
  - user\simotion\hmi
- Backing up configuration data for modular machines in conjunction with the **_activateConfiguration** system function, stored in the following directory:
  - install\simotion
- Backing up unit data (data saved on the CF card using **_saveUnitDataSet/ _exportUnitDataSet**), stored in the following directory:
  - user\simotion\user dir\<unitname>

Note

When a version is changed, you must use the "Save variables" function to back up data saved using **_saveUnitDataSet** or **_exportUnitDataSet** in a way that is not dependent on the version. You then have the option of restoring them using "Restore variables."

When performing an upgrade with a firmware version that is ≥ V4.1, these two functions are only required for backing up and restoring unit data sets that have been created using **_saveUnitDataSet**.

Retain and unit data (saved with **_exportUnitDataSet**) remain valid even after a version upgrade.

Firmware update via CompactFlash card

Procedure

Proceed as follows to perform the upgrade:

1. Switch off the power supply to the D4x5-2.
2. Remove the CompactFlash card from the SIMOTION D4x5-2 and insert it into the CompactFlash card adapter on your PC.
3. Open Windows Explorer. The CompactFlash card must be visible as a removable data carrier in the Windows Explorer under an arbitrary drive letter.

4. If necessary, back up the licenses, retain data, and user data on the CompactFlash card to your PC (see Section Backup of the CompactFlash card data (Page 7239)).
5. Delete all the data from the CompactFlash card.
6. Unzip the firmware file to the CompactFlash card using a ZIP file utility such as WinZip. Always maintain the file structure when setting up the unpacking tool.
7. Copy the data saved in step 4 back to the appropriate folder structure on the CompactFlash card.
8. Remove the CompactFlash card from the CompactFlash card adapter on your PG/PC.
9. Insert the CompactFlash card into the D4x5-2.
10. Switch on the power supply for the D4x5-2. The new firmware is loaded from the CompactFlash card to the SIMOTION D4x5-2.

Upgrading SINAMICS

Depending on the settings, the SINAMICS components are also automatically upgraded to the component version of the CF card with a firmware update of the SIMOTION D.

In order for a FW update to be performed for all components, the components must be correctly connected in accordance with the configured topology.

The component version can be obtained from the CONTENT.TXT file in the main directory of the CF card.

Upgrading the firmware of SINAMICS components automatically

When starting up, the system automatically upgrades or downgrades all DRIVE-CLiQ components to the version of the component firmware on the CF card. Components that cannot be downgraded to the component firmware version on the CF card (e.g. old firmware on the CF card and new components to which the old firmware cannot be loaded) retain their firmware version. The resulting firmware version combinations are always functional.

Note

The components' firmware is upgraded or downgraded automatically based on the FW version on the CF card and the FW version on the SINAMICS components (DRIVE-CLiQ components, CBE30-2, TB30, Power Modules, etc.).

The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

A firmware update on DRIVE-CLiQ components is indicated by the RDY LED flashing red and green:

- Firmware update running: RDY LED flashes slowly (0.5 Hz)
- Firmware update complete: RDY LED flashes quickly (2 Hz), POWER ON required

These flashing patterns are also displayed by the yellow RDY LED on the SIMOTION D/CX32-2 and indicate that a firmware update is being carried out on components connected to the SIMOTION D/CX32-2 or that all components have completed the firmware update.

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

CBE30-2 option board:

During the firmware update, the OPT LED of the SIMOTION D module and the SYNC LED of the CBE30-2 flash green.

Updating the firmware of the SINAMICS components

The SINAMICS components' firmware is updated automatically, depending on the setting of parameter p7826.

- p7826 = 0: Upgrade/downgrade deactivated
- p7826 = 1: Upgrade and downgrade (factory setting)
- p7826 = 2: Upgrade only

Note

The automatic firmware update via p7826 = 1 (upgrade and downgrade) must not be deactivated when using Safety Integrated.

If you are updating the firmware manually, proceed as follows:

1. Select the SINAMICS component in the Project Navigator, e.g. SINAMICS Integrated.
2. Double-click "Overview" in the Project Navigator.
   The "SINAMICS_Integrated - Overview" dialog box opens with a list of available drive objects.

3. Click "Version overview" to open the list of connected SINAMICS components.
4. Go online and select the devices whose firmware you wish to update.
   The list displays the current firmware version of the devices.
5. Click "Firmware update" to download the new firmware to the devices. To do so, you must select all components whose firmware is to be updated.
6. When the firmware update is complete, switch the 24 V power supply off and back on again. The device is now ready for operation.

Note

The SINAMICS components must be configured for a firmware update to take place. The firmware cannot be updated if the components have not been configured.

You can also update the firmware using the expert list. See the *SINAMICS S120* Commissioning Manual for a description of how to do this.

Download project to target system

Once all the changes required for upgrading your project have been made, you must download the project to the SIMOTION D4x5-2.

Requirement

The firmware required is located on the CompactFlash card; for information, refer to the section titled Firmware update via CompactFlash card (Page 7240).

You have recompiled the project and checked it for consistency. See Section Save project, compile and check consistency (Page 7233).

Procedure

1. Save the project.
2. Click "Connect to selected target systems" to establish a connection to the target system.
3. Execute "Download project to target system" and then "Copy RAM to ROM" to download the upgraded project to the CompactFlash card as well.
4. Because of the automatic follow-up configuration in the SINAMICS Integrated drive, you must now execute "Load CPU / drive unit to PG".
5. Save the project.

Note

When upgrading SINAMICS drive units (e.g. SINAMICS Integrated) only the p parameters (setting parameters) are loaded into the upgraded project. The r parameters (monitoring parameters) are not loaded. The r parameters in the drive unit are derived or calculated from an automatic subsequent parameterization and must therefore be uploaded to the project. To do this, execute "Load CPU/drive unit to PG". If the upload is not performed, this can lead to inconsistencies in the drive parameterization dialog boxes.

General information on SINAMICS firmware update

Requirements for firmware update

For a firmware update of the SINAMICS Integrated incl. connected CX32-2, it is enough to plug a CF card with new drive firmware into SIMOTION D4x5-2.

To ensure that a firmware update of all connected SINAMICS components is performed (Line and Motor Modules, TM, SMC, ...), the CF card must additionally contain a project, incl. configured drive component, and the drive components must be topologically correctly connected.

Impermissible operating actions during firmware update

The following actions must be avoided while a firmware update is in progress because they could cause the firmware update to be interrupted:

- Disconnection of the DRIVE-CLiQ cable to the CX32-2
- Connection of an CX32-2 to DRIVE-CLiQ
- POWER ON on D4x5-2 or CX32-2 Controller Extensions
- Operation of the RESET button on D4x5-2 or CX32-2 Controller Extensions
- Memory reset with the hardware mode switch
- Updates via the SIMOTION IT web server
- CPU download and memory reset via HW Config

If a firmware update is interrupted, it may be necessary switch the power off and then on the power again, and repeat the firmware update.

The following operating actions are not possible during a firmware update:

- Going online to the SINAMICS
- Delete user data on card
- Project download
- Download CPU / drive unit to target device
- Memory reset via software mode switch
- Restoring the default settings
- Automatic drive configuration

If an attempt is made to go online with SIMOTION SCOUT to the SINAMICS Integrated/CX32-2 during the update, the connection attempt with be abandoned after > 1 minute, depending on the constellation.

Note

With a CF card adapter, in addition to new firmware, "Load to file system" can be used to store a project with configured drive components on the CF card.

If the CF card is plugged into the D4x5-2 Control Unit and the latter is switched on, the firmware of all connected components is updated immediately. Depending on the quantities involved (number of CX32-2, Line and Motor Modules, TM, SMC, ...), the update can take several minutes (RDY-LED flashes slowly).

Online connection to the SINAMICS Integrated/CX32-2 is not possible during this time.

Components requiring POWER OFF-ON following a firmware update signal this by means of the fast flashing RDY LED. Switch the 24 V supply to the relevant components off and on again to initialize.

SIMOTION CompactFlash card

Changing the CompactFlash card

Requirement

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Comply with the ESD rules.

Procedure

To change the CF card, proceed as follows:

1. Switch off the power supply.
2. Remove the CF card from the plug-in slot of the Control Unit.
3. Gently insert the new CF card into the empty plug-in slot until it clicks into place. The direction of insertion of the CF card is indicated by an arrow located on both the plug-in slot and the CF card.
   When properly installed, the card does not extend beyond the housing of the SIMOTION D4x5-2.
4. Switch the power supply on again.

Writing to the the CompactFlash card

Overview

You can write to the CF card by:

- Writing to the CF card inserted into a SIMOTION D
  This function requires the connection to be established between the PG/PC and the module.
- Writing to the CF card without a SIMOTION D module
  For this function, you need a CF card adapter.

Note

The CF card always comes formatted. It contains the SIMOTION Kernel (SIMOTION D firmware).

To ensure that the CF card functions properly, the card must not be repartitioned.

Writing to the CF card inserted into a SIMOTION D

The CF card can be used to store technology packages and user data (programs, configuration data and parameter assignments) from the "volatile data" area on the CF card. Proceed as follows:

1. Establish the connection between the SIMOTION D and the PG/PC.
2. In SIMOTION SCOUT, the CF card is written to by means of the "Copy RAM to ROM" menu command.

Writing to the CF card without a SIMOTION D module

With a suitable memory card adapter, you can write to the CF card directly via a PG/PC. Writing to the CF card using the PG/PC is required, for example, when you want to upgrade the SIMOTION firmware.

Note

Files that have been written to the CF card with "Copy RAM to ROM" in SIMOTION SCOUT must not be modified or deleted with Windows. This can corrupt the project.

Formatting the CompactFlash card

You can format a faulty CF card, for example.

Before formatting the CF card, please observe the notes in Section Backup of the CompactFlash card data (Page 7239).

The procedure for formatting the CF card is as follows:

1. Insert the CF card into a CF card adapter connected to your PG/PC.
2. Format the CF card in Windows (FAT, FAT16 or FAT32 file system).
3. If the boot sector of the CF card is also defective, you will have to rewrite the boot loader.

Note

The CF card may not formatted with NTFS.

The following formats are permitted:

- For D410-2/D4x5-2: FAT, FAT16 and FAT32
- For D410/D4x5: FAT and FAT16

Because of the improved memory utilization on the CF card, FAT32 formatting is preferable for the D410-2 and D4x5-2. As of Kernel/firmware V4.3, D410-2 and D4x5-2 CF cards supplied as standard with FAT32 formatting.

FAT32 requires at least the boot loader version V3.02.

Bootloader on the CompactFlash card

Writing a boot loader

A boot loader may need to be written in the following situations:

- When a new boot loader is required for the SIMOTION D firmware version used
- When a new boot loader is required for the SIMOTION D hardware version used
- The boot loader is defective.
- A D410-2 CF card is to be used for SIMOTION D4x5-2.

The boot loader version can be read out using the SIMOTION SCOUT device diagnostics. If this is not possible, the boot loader version may be incorrect.
Possible error profile: All 10 LEDs light up yellow.

In this case, replace the boot loader version with the current version.

Use the "Options > Write boot sector..." function to write the boot loader version in the SIMOTION SCOUT to the CF card.

Note

You require PG/PC administrator rights to write to the boot sector. If you do not have administrator rights on your PG/PC, an administrator can enter an administrator login for you to use this function at "Options" > "Settings" > "Rights."

Detailed information on the compatibility relationships for the CF card, boot loader version, SIMOTION D hardware and SIMOTION firmware version can be found in the software compatibility list.

This list can be found on the Internet at (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Note

Note that the CF cards for SIMOTION D410-2 and D4xx have a different boot loader!

Recommended method of handling CompactFlash cards

Handling CF cards correctly

Please note the following when handling the CF card:

- The CF card may only be inserted or removed when the system is de-energized.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Comply with the ESD rules.

- CF cards are not designed to be rewritten as many times as the user wishes.
  With this in mind, you should avoid writing user data from the application to the CF card cyclically. Depending on the system, a write operation from the application may trigger one or more write operations on the CF card.
  Therefore, we recommend you adopt a conservative approach in terms of the number of writing processes. In other words, do not perform more than 100,000 write access instances from the user program over the estimated service life of the application.
- Never switch off the SIMOTION D Control Unit during write accesses to the CF card. If the SIMOTION D Control Unit is switched off during write accesses, this can result in destruction of the data and, in the worst case, to damage to the file system (FAT Table = directory) on the CF card.
  If the FAT table is destroyed, the CF card will have to be reformatted and the firmware/user data reloaded. During this process the licenses remain on the CF card.
  The FAT table can be destroyed if updating the FAT table is interrupted by switching off the SIMOTION D Control Unit. The FAT table is updated, for example, by functions such as **_exportUnitDataSet**, copy RAM to ROM, or save SINAMICS NVRAM data via p7775.
  The FAT table can also be destroyed if you pull a CF card out of a CF card adapter while Windows is accessing the CF card.
- The following functions do not update the FAT table:
  - **_saveUnitDataSet** (writing the data to an existing file)
  - **savePersistentMemoryData** (as of V4.4: writing the data to an existing file)

  Once the backup files have been created with **_saveUnitDataSet** / **savePersistentMemoryData**, this also results in updating the FAT table.

Card reader for CF cards

Because of the quickly changing market and the large differences in the quality of card readers, no specific recommendation can be made.

If problems occur identifying the CF card, this may be due to an incorrect power up of the card reader.

License key on the CF card

Depending on the type and number of runtime functions used in the project, licenses must be purchased as part of the licensing procedure for SIMOTION.

For further information on the licensing of runtime functions, see (https://support.industry.siemens.com/cs/en/en/view/42014324)

The licenses required for SIMOTION D are assigned to an individual license key for the CF card (license key cannot be transferred to another CF card).

The license key must be stored in the \KEYS\SIMOTION directory in the keys.txt file on the CF card.

Backup in the boot sector

The license key is stored in the "KEYS" directory on the CF card. When the SIMOTION D Control Unit is ramped up for the first time, the license key is backed up in the boot sector of the CF card.

A license key saved in the boot sector cannot be deleted by means of a user operation; nor can it be deleted by formatting the CF card or rewriting the boot loader.

If the keys.txt file is no longer present on the CF card (e.g. because the "KEYS" directory has been deleted), it will be written again from the boot sector to the "KEYS" directory while the SIMOTION D Control Unit is ramping up. The license key can be changed at any time (e.g. through relicensing). At the next power-up, the license key is backed up in the boot sector again.

Loss of the license key

In the event of the loss of a license key, a copy can be obtained via the Web License Manager at the following Internet address (http://www.siemens.com/automation/license). You require the hardware serial number printed on the CF card to do this. You can display the associated license key in the Web License Manager. The license file (Keys.txt) is also offered as download.

Notes for Keys.txt

- If a firmware card image is unpacked on the CF card, a KEYS directory is not available per default.
- If the CF card ramps-up **without a KEYS directory** in a controller, an empty file structure (\KEYS\SIMOTION) is created.
- The keys.txt must be stored in this directory (\KEYS\SIMOTION\keys.txt).
- If the directory contains a keys.txt with an invalid license key, this is deleted from the keys.txt when the controller ramps up, i.e. the keys.txt file exists, but is empty.
- If the directory contains a keys.txt with a valid license key, this is also stored in the boot sector of the CF card as backup when the controller ramps up.

- The keys.txt can be written online via SCOUT, with a CF card reader, or via a set up remote access (FTP).
- When licensing via the Web License Manager, you receive a keys.txt that you must store on the CF card under \KEYS\SIMOTION
- It is also possible to create this text file yourself (without formatting) with a text editor (e.g. Notepad). In this case, make sure that only the license key and a line break (CR/LF) is contained in the file.
- If the SF LED flashes slowly in red (0.5 Hz) after the control has ramped up, then either the keys.txt file is faulty (error in the license key, formatting or file/directory name) or the licensing is not adequate for the used objects that require a license.

10.1.1.9 Diagnostics

Diagnostics via LED displays

Arrangement of LED displays

The front side of the SIMOTION D4x5-2 has 10 LED displays arranged vertically in a row. There is also a 7-segment display below the blanking cover.

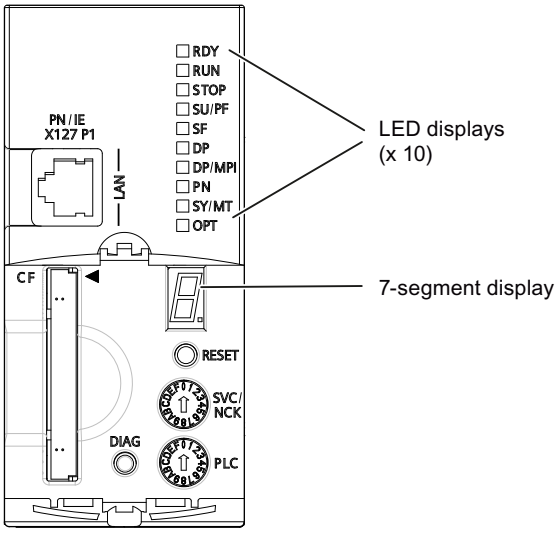


Figure 10-131 7-segment and LED displays on the D4x5-2

LED status key

The LED displays indicate the different operating modes and any errors occurring on the SIMOTION D4x5-2. They do so by illuminating, flashing, or flickering in different colors.

The following tables provide an overview of all occurring LED display combinations.

Symbols in the tables for states of the LEDs:

- 1 = LED an
- 0 = LED off
- 0.5/1 = flashing LED (0.5 Hz)
- 2/1 = flashing LED (2 Hz)
- Λ = flickering LED
- X = LED state any

SIMOTION D4x5-2 and SINAMICS Integrated displays

LED displays

Every LED can illuminate in yellow, red, or green. In the following table, the color of the LED is indicated along with the illumination state.

Table 10-65 LED displays

Meaning	LED display									
	RDY	RUN	STOP	SU/PF	SF	DP	DP/MPI	PN	SY/MT	OPT
Power-up	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)
Power-up of the D4x5-2 without CF card or with CF card without valid operating system (possibly incorrect / missing defective boot loader).	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)
Power-up stopped, reason e.g.: • incorrect firmware (e.g. D4x5) • CBE30-2 plugged into D4x5-2 DP	x	2/1 (yellow)	2/1 (yellow)	2/1 (yellow)	2/1 (yellow)	2/1 (yellow)	2/1 (yellow)	x	x	x
CF card has begun to boot, however, an error has occurred (faulty firmware).	0.5/1 (red)	0	0	0	0	0	0	0	0	0
D4x5-2 ready for operation: SIMOTION task system is running, and SINAMICS Integrated is ready to operate	1 (green)	x	x	x	x	x	x	x	x	x
Read or write access to CF card	Λ (yellow)	x	x	x	x	x	x	x	x	x
RUN	x	1 (green)	0	0	x	x	x	x	x	x

Meaning	LED display									
	RDY	RUN	STOP	SU/PF	SF	DP	DP/MPI	PN	SY/MT	OPT
Transition from RUN - STOPU	x	1 (green)	0	2/1 (yellow)	x	x	x	x	x	x
Transition from STOPU - RUN	x	2/1 (green)	0	1 (yellow)	x	x	x	x	x	x
STOPU	x	0	0	1 (yellow)	x	x	x	x	x	x
Service operating state (axis control panel in the STOPU/ measuring function)	x	2/1 (green)	0	2/1 (yellow)	x	x	x	x	x	x
Transition from STOPU - STOP	x	0	2/1 (yellow)	1 (yellow)	x	x	x	x	x	x
STOP	x	0	1 (yellow)	0	x	x	x	x	x	x
Transition from STOP - STOPU	x	0	1 (yellow)	2/1 (yellow)	x	x	x	x	x	x
Request for general reset by the D4x5-2 itself or via the mode selector	x	0	0.5/1 (yellow)	0	x	x	x	x	x	x
General reset in progress [1)]	x	0	0	0	0	0	0	0	0	0
General reset completed	x	0	1 (yellow)	0	0	0	0	0	0	0
An interrupt that can be acknowledged (alarm, message, note) is pending	x	x	x	x	1 (red)	x	x	x	x	x
"FAULT" state (7-segment display shows "F") For details, see "7-segment display" and "Special operating states"	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)	Λ (red)
HOLD state SIMOTION switches to HOLD state as soon as the program comes to a breakpoint. When the program leaves the breakpoint, SIMOTION switches out of HOLD state	x	0.5/1 (green)	1 (yellow)	1 (yellow)	x	x	x	x	x	x
"RESET" state (7-segment display shows "8") For details, see "7-segment display" and "Special operating states"	2/1 (red/ yellow)	x	x	x	x	x	x	x	x	x

Meaning	LED display									
	RDY	RUN	STOP	SU/PF	SF	DP	DP/MPI	PN	SY/MT	OPT
DCP flashing (for interface X120, X127, X130, X150, X1400) This function is used to check the correct assignment to a module and its interfaces. DCP flashing of the module is activated in HW Config under "Target system" > "Ethernet" > "Edit Ethernet node" > "Browse" button > "Flashing" button	x	x	x	x	2/1 (red)	x	x	x	x	x
Incompatible hardware; SIMOTION Kernel is no longer operating with complete functionality on the hardware being used	1 (yellow)	1 (red)	x	x	x	x	x	x	x	x
Underlicensing of technology/ option objects	x	x	x	x	0.5/1 (red)	x	x	x	x	x
SIMOTION IT service mode / switch position 8 for 120 minutes.	x	x	x	x	0.5/1 (red)	x	x	x	x	x
SINAMICS Integrated										
Initialization of SINAMICS firmware	1 (yellow)	x	x	x	x	x	x	x	x	0
Component detection via LED is activated (p0124[0]).[3)]	2/1 (green-yellow or red-yellow)	x	x	x	x	x	x	x	x	x
Commissioning/reset	0.5/1 (green)	x	x	x	x	x	x	x	x	0
SINAMICS Integrated is ready for operation and cyclic DRIVE-CLiQ communication is taking place	1 (green)	x	x	x	x	x	x	x	x	x
General SINAMICS Integrated error (check parameterization/ configuration)	2/1 (red)	x	x	x	x	x	x	x	x	x
SINAMICS Integrated has not powered up (SINAMICS firmware not available / faulty / incorrect firmware on CF) or a fault has occurred in SINAMICS Integrated	1 (red)	x	x	x	x	x	x	x	x	x
Read or write access to CF card	Λ (yellow)	x	x	x	x	x	x	x	x	x
Upgrade/downgrade of DRIVE-CLiQ components running	0.5/1 (yellow)	x	x	x	x	x	x	x	x	x

Meaning	LED display									
	RDY	**RUN**	**STOP**	**SU/PF**	**SF**	**DP**	**DP/MPI**	**PN**	**SY/MT**	**OPT**
The upgrade/downgrade of DRIVE-CLiQ components is completed (power OFF/ON of the upgraded/downgraded devices is necessary)	2/1 (yellow)	x	x	x	x	x	x	x	x	x
Underlicensing SINAMICS functions [5]	x	x	x	x	0.5/1 (red)	x	x	x	x	x
PROFIBUS DP interfaces as master										
No parameter assignment available	x	x	x	x	x	0	0	x	x	x
At least one slave is missing	x	x	x	x	x	1 (red)	1 (red)	x	x	x
Bus state "Clear"	x	x	x	x	x	0.5/1 (green)	0.5/1 (green)	x	x	x
Bus state "Operate"	x	x	x	x	x	1 (green)	1 (green)	x	x	x
PROFIBUS DP interfaces as i-slave										
No parameter assignment available	x	x	x	x	x	0	0	x	x	x
No parameter assignment master available	x	x	x	x	x	1 (red)	1 (red)	x	x	x
Bus state "Clear"	x	x	x	x	x	0.5/1 (green)	0.5/1 (green)	x	x	x
Bus state "Operate"	x	x	x	x	x	1 (green)	1 (green)	x	x	x
Upgrading SIMOTION devices (device update tool)										
Downgrading requested	x	x	x	x	Λ (green)	x	x	x	x	x
Upgrade/downgrade running	x	x	x	x	0.5/1 (green)	x	x	x	x	x
Upgrade/downgrade completed with error	x	x	x	x	Λ (red)	x	x	x	x	x
Upgrade/downgrade completed without error	x	x	x	x	0 [4]	x	x	x	x	x
Update data is being copied from USB stick to CF card	0.5/1 (yellow/ green)	x	x	x	x	x	x	x	x	x
Booting is not possible from the USB stick	0.5/1 (yellow) min. 10 s	x	x	x	x	x	x	x	x	x

Meaning	LED display									
	RDY	RUN	STOP	SU/PF	SF	DP	DP/MPI	PN	SY/MT	OPT
Upgrade data copying from USB stick to CF card completed with error	1 (red)	x	x	x	x	x	x	x	x	x
Upgrade data copying from USB stick to CF card completed without error	1 (green) min. 10 s [2)]	x	x	x	x	x	x	x	x	x
Backing up and restoring diagnostic data and non-volatile data										
Backing up diagnostic data and non-volatile data (backup running)	x	x	Λ (yellow)	Λ (yellow)	x	x	x	x	x	x
Backing up diagnostic data and non-volatile data (backup completed)	x	Λ (green)	x	x	x	x	x	x	x	x
Request: "Restore non-volatile data" (with switch position "A")	x	x	x	x	Λ (green)	x	x	x	x	x
PROFINET IO interface for D4x5-2 DP/PN (onboard interface, X150)										
Onboard PROFINET interface is operating without error; the data exchange to all configured IO devices is running.	x	x	x	x	x	x	x	0	x	x
Bus error	x	x	x	x	x	x	x	1 (red)	x	x
Failure of a connected IO device	x	x	x	x	x	x	x	2/1 (red)	x	x
The PROFINET interface has not yet synchronized with the send cycle clock of PROFINET IO with IRT, or PROFINET IO with IRT has not been configured	x	x	x	x	x	x	x	x	0	x
The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT	x	x	x	x	x	x	x	x	1 (green)	x
The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT (SINAMICS Integrated and ext. DP interfaces have not synchronized yet)	x	x	x	x	x	x	x	x	2/1 (green)	x
PROFINET IO interface (CBE30-2, X1400)										
No CBE30-2 inserted	x	x	x	x	x	x	x	x	x	0
CBE30-2 runs error-free	x	x	x	x	x	x	x	x	x	1 (green)

Meaning	LED display									
	RDY	RUN	STOP	SU/PF	SF	DP	DP/MPI	PN	SY/MT	OPT
Bus error (CBE30-2): • Failure of a connected IO device • At least one of the assigned IO devices cannot be addressed • Incorrect or no configuration	x	x	x	x	x	x	x	x	x	2/1 (red)
Firmware download running	x	x	x	x	x	x	x	x	x	2/1 (green)
Firmware download faulty	x	x	x	x	x	x	x	x	x	0.5/1 (red)

1) The LED state appears only very briefly due to the speed at which the "general reset" is completed. The module then powers up (the STOP LED flashes yellow at 2 Hz).

2) Copying is completed when the SF LED is steady green for at least 10 s.

3) Both options depend on the LED state when activated via p0124[0] = 1.

4) The upgrade or downgrade is complete when the SF LED goes out. The device then powers up automatically in the upgraded or downgraded configuration (SF LED display then depends on the operating state of the device).

5) In the case of SINAMICS licenses (e.g. SINAMICS DCB Extension), underlicensing of SINAMICS Integrated/CX32-2 via the flashing SF-LED is indicated on the SIMOTION D Control Unit. An entry is also made in the diagnostic buffer and the underlicensing is displayed in the license dialog box of SIMOTION SCOUT. The licensing is effected (like for SIMOTION licenses) via SIMOTION SCOUT or via the SIMOTION license key on the CF card.

See also

For detailed information on the LED states, see Section LED displays of the PROFINET interface (Page 7260).

Additional references

Further references can be found in the *Upgrading SIMOTION Devices* Operating Instructions.

7-segment display

The 7-segment display provides further status information in addition to the LED displays.

State (code)	Meaning
5	**CBE30-2 is plugged into wrong Control Unit.** (CBE30-2 is only supported by SIMOTION D4x5-2 DP/PN, not by SIMOTION D4x5-2 DP) In addition: • All LEDs light up yellow (RUN, STOP, SU/PF, SF, DP, DP/MPI flashing at 2 Hz) • CBE30-2: SYNC lights up green; FAULT flashes red at 0.5 Hz.
6	**SIMOTION D has ramped up**
8	**Module in RESET state** (RESET button pressed; overtemperature; faulty/ disconnected fan/battery module) Fan faults are detected by • A cyclic fan test • Or when the fan is switched on a malfunction is detected (fan does not turn or fan turns at too low a speed). State 8 is also displayed when a fan module is not connected at switch on. For further information, see "Special operating states".
8 Middle segment of the "8" flickers	**Power-up of the SIMOTION D4x5-2 has failed** Possible cause: Incorrect boot loader on the CF card Update this with SIMOTION SCOUT (see *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual* Update boot loader of the CF card (Page 7247)).
A or E	**Software system error** Backup the diagnostics data for further diagnostics (see *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual* Backup of diagnostic data and non-volatile SIMOTION data (Page 7266)). If you cannot determine the cause with the diagnostics data, contact the Hotline and provide this code and the diagnostics data.
F	**F state (FAULT)** In the F state, the CPU of the D4x5-2 is in HOLD state, i.e. no software is running (D4x5-2 operating system is stopped) For further information, see "Special operating states".

State (code)	Meaning
6.. Flashing dot	**Servo running** (flashing frequency depends on the servo cycle clock)
1.2.3.4. 4-digit code (followed by a dot)	**BIOS error** Possible cause: Module defect Contact the Hotline and provide the code.
F.2 2-digit code (starting with "F")	**Boot loader error** F2: Sector "0" cannot be read F3: File system cannot be initialized F5: Bin file too large for the internal buffer Try a different CF card with F2/F3. If it functions correctly, the CF card is defective. The cause for F5 can be an older boot loader; update this with SIMOTION SCOUT (see *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual* Update boot loader of the CF card (Page 7247)). For code F1, F4 and F6-F9, contact the Hotline and provide the code.
Further states for a firmware and project update	See Section Update via device update tool (Page 7235)

Special operating states

The following "special operating states" are displayed via the status LEDs and the 7-segment display.

F state (FAULT)

In the F state, the CPU of the D4x5-2 is in HOLD state, i.e. no software is running (D4x5-2 operating system is stopped)

The state occurs with faults to which the D4x5-2 operating system cannot respond.

The state is displayed as follows:

- All LEDs flicker red and
- The 7-segment display shows "F"

Possible measures:

- Check the CF card.
- Perform a commissioning again.
- Replace the D4x5-2.
- Back up the diagnostics data for further diagnostics (see Section Backup of diagnostic data and non-volatile SIMOTION data (Page 7266)).

The state can only be exited by switching the D4x5-2 off and on.

If you cannot determine the cause, contact the Hotline and provide this code and the diagnostics data.

RESET state (permanent RESET)

In the RESET state, the CPU of the D4x5-2 is in a permanent RESET, i.e.

- No software is running (D4x5-2 operating system is stopped) and
- The hardware is in a state that produces less power loss

This state occurs, for example, if the max. module temperature has been greatly exceeded.

The state is displayed as follows:

- The RDY LED flashes red/yellow at 2 Hz
- The 7-segment display shows "8"

Possible measures:

- Correct any overtemperature.
- Ensure that the 24 V supply is stable without any power dips.
- Insert the fan/battery module.

The state can only be exited by switching the D4x5-2 off and on.

LED displays of the PROFINET interface

Position of the onboard PN interface (X150)

The following figure contains information on the PROFINET interface of the SIMOTION D4x5-2 DP/PN Control Unit. Position of the interface, labeling of the ports and the associated displays are described.

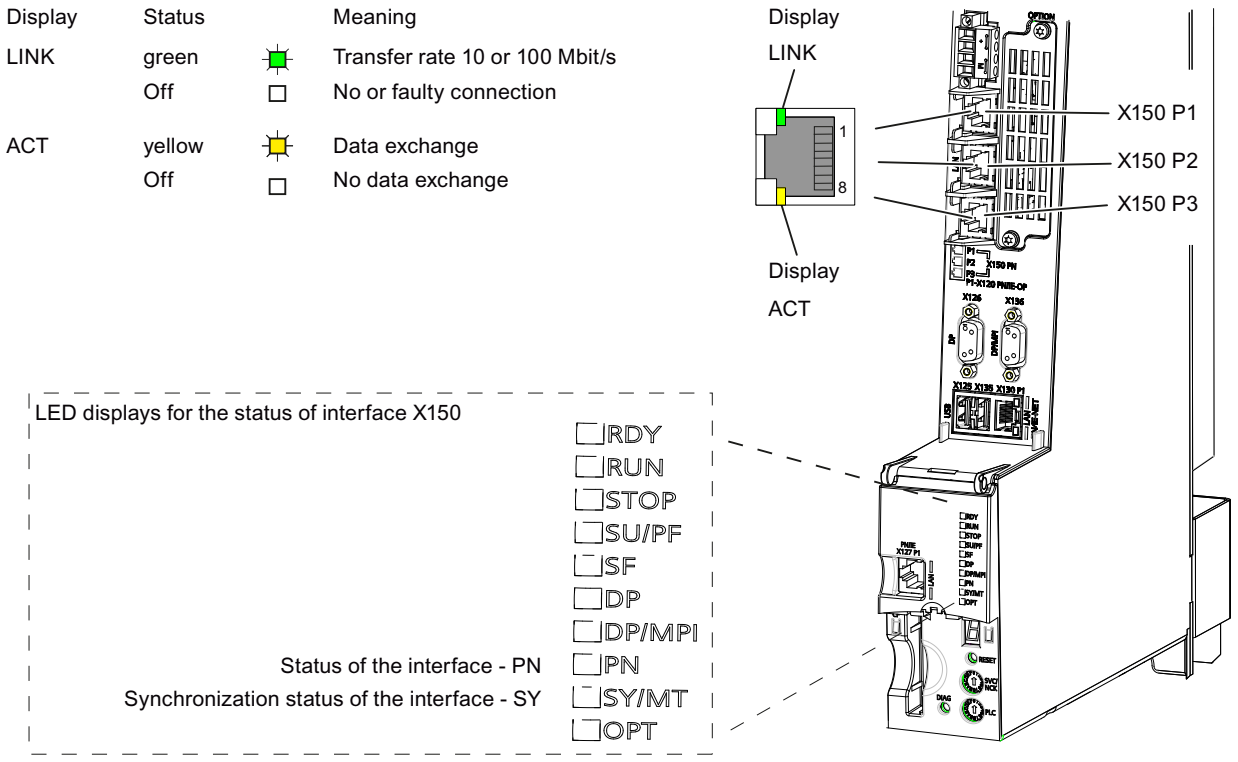


Figure 10-132 Position of the PROFINET interface X150

Note

The 3rd port of the PROFINET IO interface X150 P3 is also designated as X120 PN/IE OP. This designation is not relevant for SIMOTION D.

Position of the PN interface of the CBE30-2 (X1400)

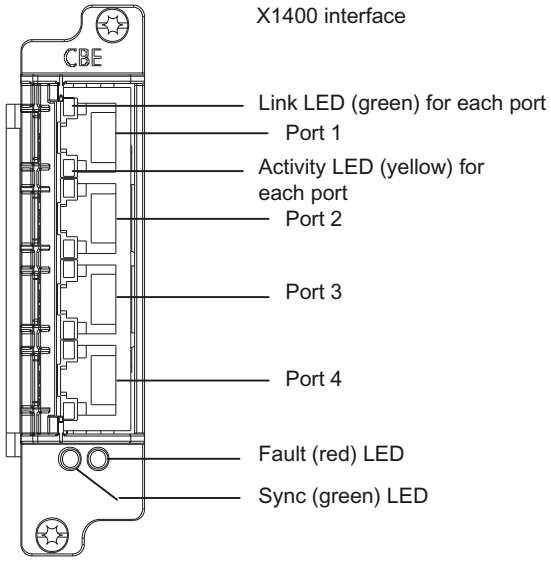


Figure 10-133 CBE30-2 front view

LED displays for PROFINET

Table 10-66 Meaning of the LED displays for the D4x5-2 DP/PN and CBE30-2

Name	Color	State	Meaning
LINK	Green	On	A different device is connected to port x and a physical connection exists.
ACT	Yellow	Flashing	Data is being received or sent at port x.

Name	Color	State	Meaning
D4x5-2: PN CBE30-2: FAULT	red	Off	as an IO controller: • PROFINET interface is operating without error; the data exchange to all configured IO devices is running as an I-Device: • PROFINET interface is running without error, there is at least one communication link with the I-Device
		On	PROFINET interface bus fault • No physical connection to a subnet/switch • Incorrect transmission rate • Full duplex transmission is not activated • More than one IP address/NameOfStation in the network
		Flashing (2 Hz)	as an IO controller: • Failure of a connected IO device • At least one of the assigned IO devices cannot be addressed. • Incorrect or no configuration as an I-device: The LED flashes until at least one controller has correctly established communication with this I-Device. Possible causes: • Incorrect IP address • Incorrect configuration / parameterization • IO controller not connected / switched off, although an Ethernet connection has been established. • IN Shared I-Device operation: all configures IO controllers are not connected/switched off, but the Ethernet connection is up (link established to the neighboring device) • Incorrect or missing device name. • The response monitoring interval has elapsed • The CPU is an I-Device and communication with the higher-level controller fails
		flashing (0.5 Hz)	CBE30-2 power-up stopped, possible reason: CBE30-2 is plugged into wrong Control Unit. (CBE30-2 is only supported by SIMOTION D4x5-2 DP/PN, not by SIMOTION D4x5-2 DP)

Name	Color	State	Meaning
D4x5-2: SY/MT [1] CBE30-2: SYNC [1]	Green	Off	The PROFINET interface has not synchronized yet to the send cycle clock of PROFINET IO with IRT, or PROFINET IO with IRT has not been configured (e.g. only PROFINET IO with RT or TCP/IP communication). If IRT data has been configured for SIMOTION, the PROFINET interface generates a local substitute cycle clock as long as there is no synchronization to the send cycle clock of PROFINET IO with IRT: SIMOTION task system has synchronized to the local substitute cycle clock of the PROFINET interface. SINAMICS Integrated and external isochronous DP interfaces are synchronized to the local substitute cycle clock of the PROFINET interface.
		On	The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT. If isochronous IRT data has been configured for SIMOTION, then the following applies: The task system of SIMOTION has synchronized to the send cycle clock of PROFINET IO with IRT. SINAMICS Integrated and external isochronous DP interfaces are synchronized to the send cycle clock of PROFINET IO with IRT. If no isochronous IRT data has been configured for SIMOTION, this state indicates that only the PROFINET interface is synchronized to the send cycle clock of PROFINET IO with IRT. The PROFINET interface is then used to forward the IRT data.
		Flashing (2 Hz)	The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT If IRT data has been configured for SIMOTION, then the following applies: The task system of SIMOTION has synchronized to the send cycle clock of PROFINET IO with IRT. SINAMICS Integrated and external isochronous DP interfaces are not yet synchronized to the send cycle clock of PROFINET IO with IRT.

[1] If no IRT has been configured, then generally no synchronization will be made to the send cycle clock. LED is off.

Note

LED displays of the PROFINET interface for D4x5-2 DP

Up to and including firmware V4.4, the PN and SY/MT LEDs light up yellow in normal operation.

As of firmware V4.5, the LEDs are off.

Note

If the I-Device is deactivated via the system function _deactivateDpSlave, the PN or FAULT LED will respond as if the I-Device had not been configured.

The PN/FAULT LED is lighted if an IO controller is configured and no IP address exists.

If no IO controller is configured, the LED will not light up.

LED displays of the Ethernet interface

The Ethernet ports are equipped with LEDs to display Link and Activity.

Table 10-67 State of the Link and Activity LEDs

LED	State	Meaning
LINK (upper LED)	OFF	No or faulty connection
	Lights up green	Transfer rate 10 or 100 Mbit/s: A different device is connected to port x and a physical connection exists
	Lights up yellow	Transfer rate 1,000 Mbit/s: A different device is connected to port x and a physical connection exists
ACT (lower LED)	OFF	No data exchange
	Flickers yellow	Data exchange: Data is being received or sent at port x

LED displays of the CX32-2 controller extension

The different states that occur during power-up are indicated by the LEDs on the CX32-2 controller extension.

- The duration of the individual states varies.
- If an error occurs, the power-up is terminated and the cause is indicated accordingly via the LEDs.
- At the end of an error-free power-up, all LEDs are switched off briefly.
- After power-up, the LEDs are controlled via the loaded software.

Table 10-68 Load software

LED		State	Comment
RDY	DP		
Yellow	Yellow	Reset	Hardware reset All other LEDs light up yellow
Red	Red	BIOS loaded	-
Red 2 Hz	Red	BIOS error	Error occurred while loading the BIOS
Red 2 Hz	Red 2 Hz	File error	• D4x5-2 CompactFlash card not available or faulty • Software on D4x5-2 CompactFlash card not available or faulty
Red	Yellow (flashing)	FW loading	RDY LED lights up red, DP LED flashes yellow without fixed frequency
Red	Off	FW loaded	-
Off	Red	FW checked (no CRC error)	-
Red 0.5 Hz	Red 0.5 Hz	FW checked (CRC error)	CRC invalid

Table 10-69 Firmware

LED		State	Comment
RDY	DP		
Yellow	Not relevant	Initializing	–
Alternating		Running	See the table below

Table 10-70 CX32-2 – Description of the LEDs after ramp-up

LED	Color	State	Description, cause	Remedy
RDY (READY)	-	Off	Electronic power supply is missing or outside the permissible tolerance range.	Check power supply
	Green	Continuous light	The component is ready for operation and cyclic DRIVE-CLiQ communication is taking place.	–
		Flashing 0.5 Hz	Commissioning/reset	–
		Flashing 2 Hz	Writing to the memory card	–
	Red	Flashing 2 Hz	General errors	Check parameterization/ configuration data
	Red/ green	Flashing 0.5 Hz	CX32-2 is ready for operation. However, software licenses are missing.	Obtain licenses
	Yellow	Flashing 0.5 Hz	Firmware update of the connected DRIVE-CLiQ components running	–
		Flashing 2 Hz	DRIVE-CLiQ component firmware update complete. Wait for POWER ON for the components in question.	Perform POWER ON for the respective component
	Green/ yellow or red/ yellow	Flashing 2 Hz	Component detection via LED is activated (p0124[0]). **Note**: Both options depend on the LED state when component detection is activated via p0124[0] = 1.	–

LED	Color	State	Description, cause	Remedy
DP (PROFIdrive cyclic operation)	–	Off	Cyclic communication has not (yet) taken place. **Note:** The PROFIdrive is ready to communicate when the CX32-2 is ready for operation (see RDY LED).	–
	Green	Continuous light	Cyclic communication is taking place.	–
		Flashing 0.5 Hz	Full cyclic communication has not yet taken place. Possible causes: • D4x5-2 does not transfer any setpoints. • During isochronous operation, no global control (GC) or a faulty global control (GC) is transferred by the controller.	–
	Red	Flashing 0.5 Hz	D4x5-2 sends faulty parameterization/configuration.	Adapt configuration between D4x5-2 and CX32-2
		Flashing 2 Hz	Cyclic bus communication has been interrupted or could not be established.	Correct fault
RDY and DP	Red	Flashing 2 Hz	Bus error - communication has been interrupted.	Correct fault

Diagnostic data and non-volatile SIMOTION data

Overview

With simple operation actions (e.g. by setting the switch position) and without the need for the SCOUT engineering system, you can:

- Back up diagnostic data, including non-volatile SIMOTION data (retain data) to the CompactFlash card; for information, see Section Backing up diagnostics data and non-volatile SIMOTION data (Page 7266).
- Back up HTML pages (including the most up-to-date content) to the CompactFlash card for diagnostic purposes; for information, see Section Diagnostics via HTML pages (Page 7273).
- Restore backed-up non-volatile SIMOTION data (retain data); for information, see Section Deleting/restoring non-volatile SIMOTION data (Page 7274).

Backup of diagnostic data and non-volatile SIMOTION data

Diagnostic data

Following a fault on a SIMOTION device, diagnostic data (e.g. diagnostic buffer content, up-to-date content of HTML pages, etc.) can provide important information on the cause of the fault. For this purpose, data can be backed up to the CF card via a "simple operator action" (e.g. via service selector switch or DIAG button on the D4x5-2).

You then have the following options for the diagnostic data:

- Fetching them from the CF card using a card reader
- You can load it with the SIMOTION IT web server or by FTP

You can also use them for diagnostic purposes or provide Technical Support with them for evaluation purposes.

Various options are available to you for backing up diagnostic data:

- Backing up during operation (in STOP/STOPU/RUN mode); see Section Procedure for backing up during operation (Page 7267)
  - Using the SIMOTION IT web server;
    The web server also offers the option of fetching diagnostic data online.
  - Via the DIAG button
  - Via the service selector switch
- Backing up while the module is starting up; for details, see Section Procedure for backing up during startup (Page 7269)
  - Via the DIAG button
  - Via the service selector switch
  - Control of the diagnostic data creation using an INI file stored on the CF card

Non-volatile SIMOTION data (retain data)

In addition to the diagnostic data, non-volatile SIMOTION data (retain data) is also saved on the CompactFlash card. You can use these in situations where the non-volatile SIMOTION data has not been saved on the CompactFlash card using the **_savePersistentMemoryData** system function, and you wish to restore the non-volatile SIMOTION data after a CPU has been replaced.

Note

While the non-volatile SIMOTION data is stored as a "PMEMORY.XML" backup file in the "...USER\SIMOTION" directory using the **_savePersistentMemoryData** system function, backing up diagnostic data and non-volatile SIMOTION data stores the data in the "...\USER\SIMOTION\HMI\SYSLOG\DIAG directory.

Procedure for backing up during operation

Procedure

The advantage of backing up diagnostic data and non-volatile SIMOTION data during operation is that enhanced diagnostic information via HTML pages and TO alarm information is available.

Data are backed up:

- With the SIMOTION IT web server by selecting "Diagnostics > Diagnostics Files"; see Section "Backing up diagnostic data and non-volatile SIMOTION data using the web server.
- Via the DIAG button (see the description below).
- Via the service selector switch (see the description below).

Note

In addition to the service selector switch, SIMOTION D4x5-2 modules also have a DIAG button.

If the diagnostic data are to be backed up "during operation", simply press the DIAG button. The DIAG button is therefore preferable to switch position "D" of the service selector switch.

DIAG button (preferred solution)

To back up diagnostic data and non-volatile SIMOTION data via the DIAG button, proceed as follows:

1. Press the DIAG button.
   The diagnostic data and non-volatile SIMOTION data can be created in STOP, STOPU and RUN states.
2. The diagnostic data and non-volatile SIMOTION data are backed up to the CF card.
   Backup is displayed via the status LEDs as in the table below:
3. Once the backup is complete, switch the D4x5-2 off.
4. Remove the CF card.

Table 10-71 LED displays during backup

Status	LED displays on the D4x5-2
Backup in progress	STOP LED and SU/PF LED flicker yellow
Backup complete	RUN LED flickers green

Service selector switch (alternative)

To back up diagnostic data and non-volatile SIMOTION data using a service selector switch, proceed as follows:

1. Set the service selector switch to "Diagnostics" (position "D").
   The positions of the mode selector are not relevant (i.e. the set operating mode remains unchanged).
   The diagnostic data and non-volatile SIMOTION data can be created in STOP, STOPU, and RUN states.

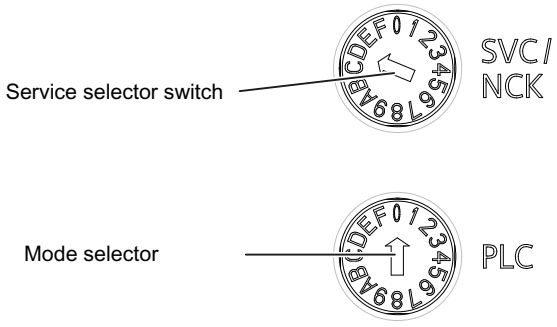


Figure 10-134 Service selector switch and mode selector

2. The diagnostic data and non-volatile SIMOTION data are backed up to the CF card. Backup is displayed via the status LEDs as in the table below.
3. Once the backup is complete, switch the D4x5-2 off.
4. Remove the CF card and reset the service selector switch to its original setting.

Table 10-72 LED displays during backup

Status	LED displays on the D4x5-2
Backup in progress	STOP LED and SU/PF LED flicker yellow
Backup complete	RUN LED flickers green

Procedure for backing up during startup

Procedure

Backing up diagnostic data and non-volatile SIMOTION data "during power-up" provides you with diagnostic information **without HTML pages / TO alarm information**.

Backing up during startup is particularly advisable for SIMOTION devices that cannot run or have crashed.

Diagnostic data and non-volatile SIMOTION data are backed up

- Via the service selector switch
- Via the DIAG button or
- Using an INI file saved on the CF card.

The procedure for each is described below.

Note

In addition to the service selector switch, SIMOTION D4x5-2 modules also have a DIAG button.

As an alternative to setting the service selector switch to the "D" position, you have the option of backing up diagnostic data and non-volatile SIMOTION data by pressing the DIAG button. When backing up data during power-up, the DIAG button must be held down until the backup is completed. As this can take 20-30 seconds, switch position "D" is preferable here.

Service selector switch (preferred solution)

Data are backed up using a service selector switch as follows:

1. Set the service selector switch to "Diagnostics" (position D).
   The positions of the mode switch are not relevant (i.e. the set operating mode remains unchanged).

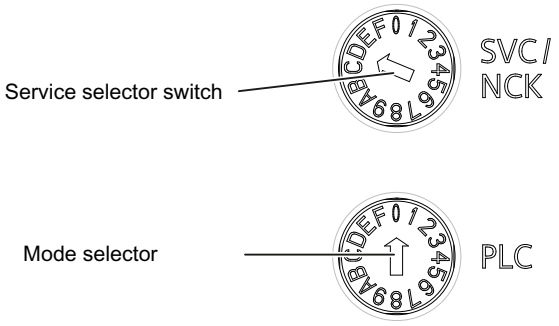


Figure 10-135 Service selector switch and mode switch, Service switch in position "D"

2. Switch the D4x5-2 off and back on again.
3. Wait for the device to start up.
   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card during power-up, provided that this is still possible and is not prevented by HW defects, for example.
4. Once the backup is complete, switch the D4x5-2 off.
5. Remove the CF card and reset the service selector switch to its original setting.

Table 10-73 LED displays during backup

Status	LED displays on the D4x5-2
Backup in progress	STOP LED and SU/PF LED flicker yellow
Backup complete	RUN LED flickers green

DIAG button (alternative)

To back up data via the DIAG button, proceed as follows:

1. Switch the D4x5-2 off.
2. Press the DIAG button and hold it down. Switch the D4x5-2 on.
3. Wait for the device to ramp up.
   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card during power-up, provided that this is still possible and is not prevented by hardware defects, for example.
4. Once the backup is complete, you can release the button and switch off the D4x5-2.
5. Remove the CF card.

Table 10-74 LED displays during backup

Status	LED displays on the D4x5-2
Backup in progress	STOP LED and SU/PF LED flicker yellow
Backup complete	RUN LED flickers green

INI file in the main directory of the CF card

1. Use a text editor (such as Notepad) to create a file called *simotion.ini*
2. Add the following text: **DIAG_FILES=1**
   You must use a text editor and may not use any formatting in the text.
3. Copy the simotion.ini file to the main directory of the CF card.
4. Insert the CF card into the module, which is switched off.
5. Switch the D4x5-2 on and allow the SIMOTION device to power up.
   The diagnostic data and the non-volatile SIMOTION data will be backed up to the data carrier during power-up, provided that this is still possible and is not prevented by HW defects, for example.
6. Once the backup is complete, switch off the SIMOTION device.
7. Remove the CF card.

Note

To suppress startup in diagnostics mode again, you must delete the *simotion.ini* file from the CF card.

Table 10-75 LED displays during backup

Status	LED displays on the D4x5-2
Backup in progress	STOP LED and SU/PF LED flicker yellow
Backup complete	RUN LED flickers green

Storing data

Storing diagnostic data and non-volatile SIMOTION data

You can find diagnostic data and non-volatile SIMOTION data on the CF card in the \USER \SIMOTION\HMI\SYSLOG\DIAG directory.

Copy these data and transfer them to Technical Support when requested to do so. A standard card reader can be used to transfer the diagnostic data from the CF card via standard SIMOTION IT web server pages or via FTP.

The following data are stored:

Table 10-76 Diagnostic data on the CF card

File	Application
DIAGBUF.TXT	Diagnostic buffer in a simple text format: Numerical values; no specific plain text. A text editor is used for evaluation purposes.
PMEMORY.XML	Non-volatile data (retain data) You can restore the backed up non-volatile SIMOTION data "via an operator action" after a CPU has been replaced. (See Section Overview (Page 7274).)
TOALARMS.TXT	Text file containing the pending TO alarms. Only TO IDs, alarm numbers, and associated HEX values. Note: The TO alarms are only created if diagnostic data have been created during operation (STOP / STOPU / RUN).
HTML page	If the diagnostic data are backed up, the URLs are requested from the text file (DIAGURLS.TXT) and stored as HTML pages together with their contents. (See Section Diagnostics via websites (Page 7273).) Note: The HTML pages are only stored if diagnostic data are created during operation (STOP/STOPU/RUN).
Other files	All other files stored in the directory are only of relevance to Technical Support.

Note

Use HTML pages if you wish to back up diagnostic data in plain-text format. HTML pages enable user-friendly diagnostics. In addition to the standard SIMOTION IT web server pages, you have the option of creating your own HTML pages (e.g. for the axis status or for machine diagnostics). Customized diagnostics pages are particularly suitable for application problems, as you can define the contents yourself.

Diagnostics via websites

In the "DIAGURLS.TXT" text file found in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory, you can specify HTML files whose current content is to be stored on the CF card when diagnostic data is created during operation (e.g. "devinfo.mwsl" must be entered for the "devinfo.mwsl" website).

Since the pages in question are stored together with their most up-to-date contents, this enables the latest status information regarding the SIMOTION device, as well as the machine/system, from the point at which diagnostic data was created (e.g. when the service selector switch was activated) to be archived.

In addition to the standard SIMOTION IT web server pages, it is possible to store customized pages. Information on creating pages of this type can be found in, for example, an FAQ of the Utilities & Applications.

A tutorial on the creation of user-defined websites is also contained on the Utilities & Applications-DVD.

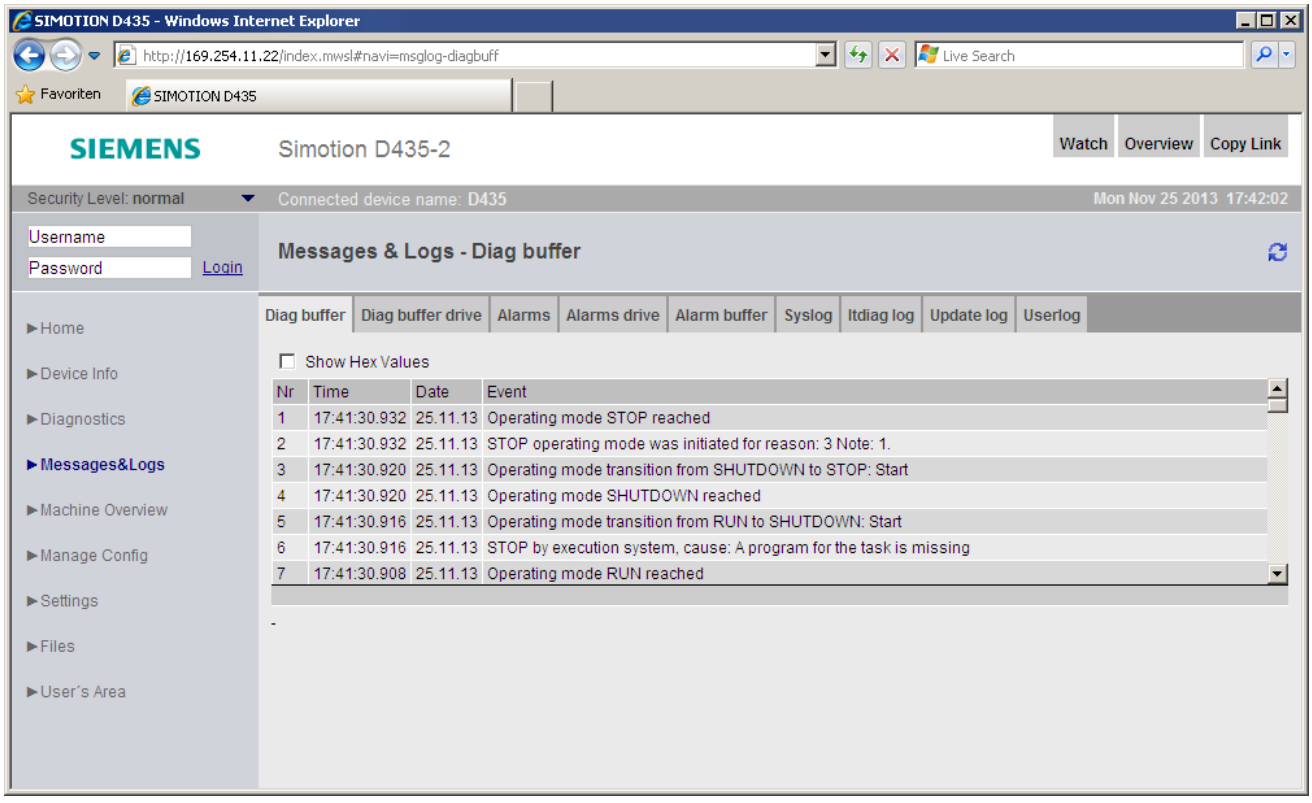


Figure 10-136 Diagnostic buffer with generated diagnostic data

The following points must be noted for the DIAGURLS.TXT file:

- A DIAGURLS.TXT file containing the standard websites is automatically created if you have not stored your own DIAGURLS.TXT file.
- Standard websites are entered "without" specifying a path (e.g. "devinfo.mwsl" for the standard website "devinfo.mwsl").
- User websites (such as "user.mwsl") in the \USER\SIMOTION\HMI\FILES directory on the CF card must contain the FILES/ path specification.

- If you have created subfolders (e.g. "myfolder" in the FILES directory), these must also appear in the path.
- Only one file name may be used per line.
- Empty lines are not permitted (an empty line will be interpreted as the end of the list).
- No distinction is made between upper-case and lower-case letters.
- It does not matter whether you use "\" or "/" in the path name.

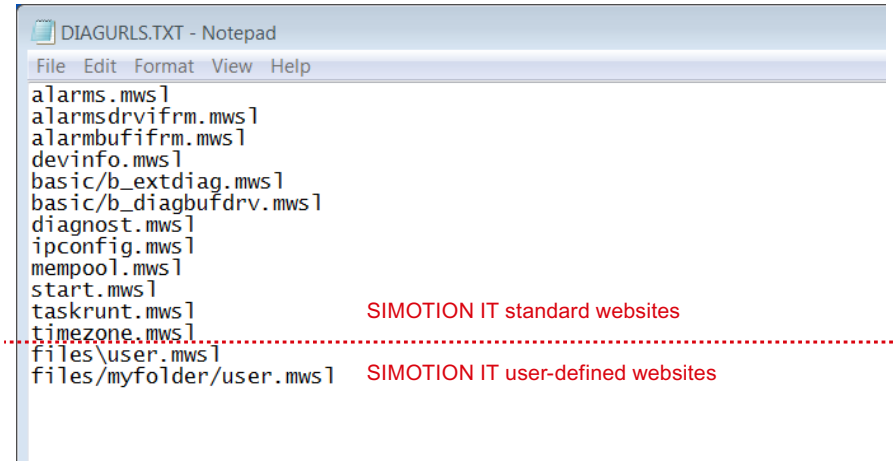


Figure 10-137 Depiction of DIAGURLS.TXT editor

Additional references

You will find detailed information on device and diagnostic information of the SIMOTION IT web server in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Delete/restore non-volatile SIMOTION data

Overview

Requirement

The non-volatile SIMOTION data have been backed up on the CF card by one of the following methods:

- by system function (_savePersistentMemoryData), see also Section Operations and their effect on the user memory (Page 6990).
- manually by service selector switch / web server / DIAG button, see Section Backup of diagnostic data and non-volatile SIMOTION data (Page 7266).

Procedure

The non-volatile SIMOTION data are restored automatically during a module replacement, see Section Replacing modules in the spare part scenario (Page 6998). The non-volatile data can also be restored manually (by manual operation).

The CF card may contain backups of non-volatile SIMOTION data in various storage locations:

- data backed up with the system function _savePersistentMemoryData
  Storage location on CF card:
  - /USER/SIMOTION/PMEMORY.XML
  - /USER/SIMOTION/PMEMORY.BAK (backup file)
- manually by service selector switch / web server / DIAG button, backed-up data
  Storage location on CF card:
  - /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML

On manual restoration, the position of the service selector switch defines which of these data will be preferably restored.

During restoration, the non-volatile SIMOTION data are first deleted and then the non-volatile SIMOTION data are restored via the PMEMORY backup file.

If restoration is not possible (e.g. file does not exist or corrupt), the next file in the priority list is accessed.

Table 10-77 Restoration of the non-volatile SIMOTION data

Position of the service selector switch	Use case	Priority sequence for use of the data backups
1	The data backed up with the system function _savePersistentMemoryData are preferably restored	1. /USER/SIMOTION/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.BAK 3. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML
A (as of V4.4)	The data backed up by service selector switch position "D" / web server / DIAG pushbutton are preferably restored	1. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.XML 3. /USER/SIMOTION/PMEMORY.BAK

For how to proceed, see Section Restoring data with switch position "1" or "A" (Page 7276).

Note

Firmware / kernel < V4.4

Because switch position "A" is only supported as of V4.4, for < V4.4, restoration must be performed with switch position "1." To force restoration of the data backed up by service selector switch position "D" / web server / DIAG button, it may be necessary to delete existing files PMEMORY.XML and PMEMORY.BAK in the directory /USER/SIMOTION/ on the CF card.

Restoring data with switch position "1" or "A"

Procedure

To restore the non-volatile SIMOTION data, proceed as follows:

Step	Switch position "1"	Switch position "A" (as of V4.4)
1.	Insert the CF card into the new SIMOTION D4x5-2. The SIMOTION D4x5-2 must be switched off!	
2.	Set the service selector switch to "**1**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service selector switch (position 1)	Set the service selector switch to "**A**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service selector switch (position A)
3.	Switch the SIMOTION D4x5-2 on.	Switch the SIMOTION D4x5-2 on. The flickering green SF LED indicates that restoration is requested via position "A."
4.	Restoration is started automatically.	Turn the service selector switch to "1" to start restoration.
5.	Once restoration has been completed, the module will start up automatically.	
6.	Switch the module off and turn the service selector switch back to "0."	
7.	Switch the SIMOTION D4x5-2 on again.	

Back up diagnostic data and non-volatile SIMOTION data via the web server

SIMOTION devices provide a web server with already prepared standard web pages. These pages can be displayed via Ethernet using a commercially available browser. Additionally, you have the option of creating your own HTML pages and incorporating service and diagnostic information.

Diagnostic data and non-volatile SIMOTION data can be backed up via the web server. The home page of the web server is opened by entering the IP address of the SIMOTION device in the address line of the browser; e.g. http://169.254.11.22

This opens the home page of the web server. To back up diagnostic data and non-volatile SIMOTION data, call the "Diagnostic files" page from the "Diagnostics" menu.

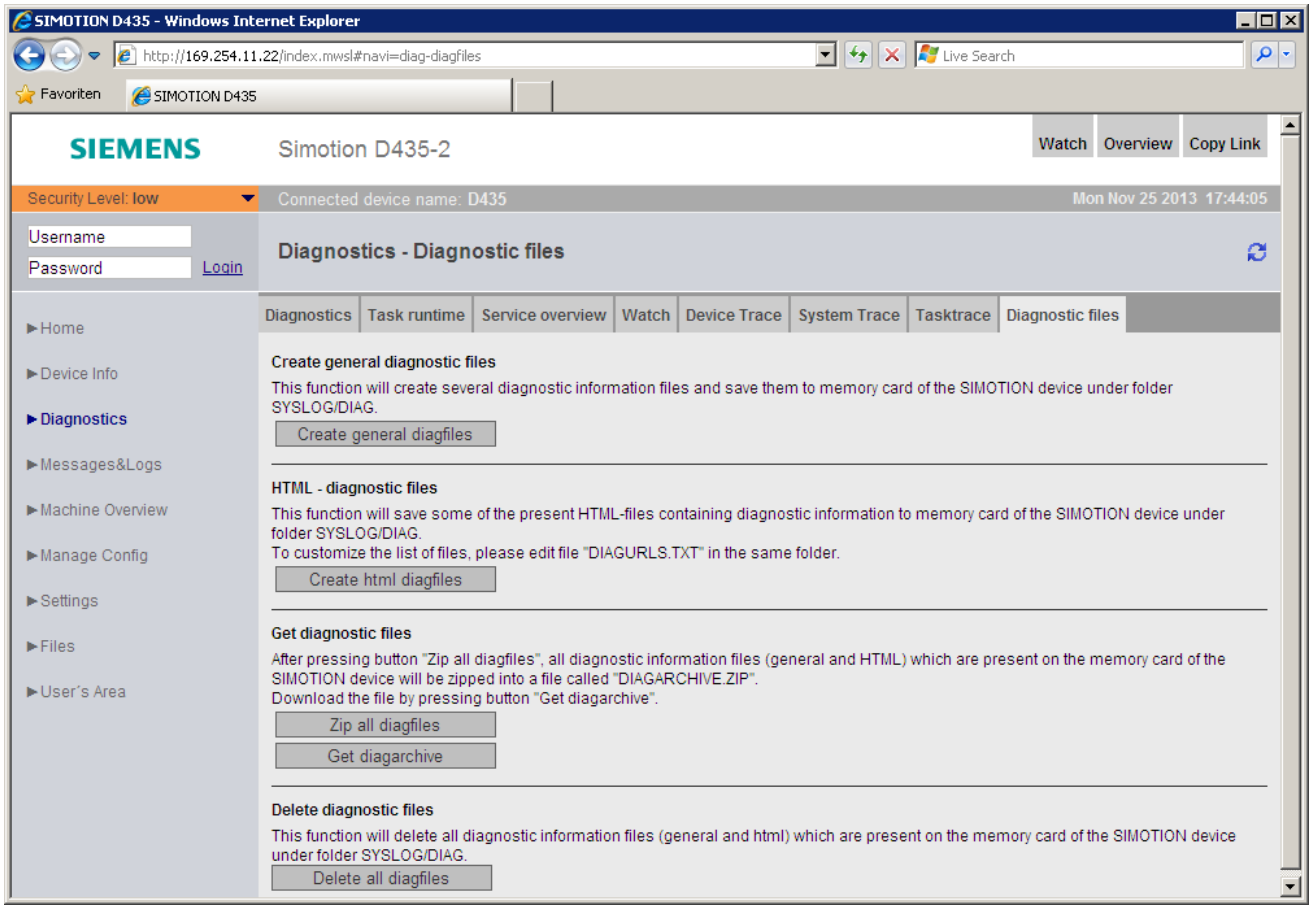


Figure 10-138 SIMOTION IT web server

Table 10-78 Functions on the "Diagnostic files" HTML page

Button	Function
Create general diagfiles	This button saves the diagnostic data and non-volatile SIMOTION data to the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory. HTML files used for diagnostics purposes are not saved.
Create html diagfiles	This button is used to save diagnostics HTML pages on the data carrier. It should be noted that only those pages that are listed in the DIAGURLS.TXT file in directory ...\USER\SIMOTION\HMI\SYSLOG\DIAG are backed up, see Section Diagnostics via HTML pages (Page 7273).
Zip all diagfiles	The "zip all Diagfiles" button enables you to compress diagnostics files. This stores all files and folders in a ZIP file in directory ...\USER\SIMOTION\HMI\SYSLOG\DIAG, while retaining the folder structure.
Get diagarchive	This button is used to save the ZIP archive to connected programming devices/PCs.
Delete all diagfiles	This button is used to delete all data stored in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory; the directory itself is not removed, however.

You can find diagnostic data and non-volatile SIMOTION data on the CF card in the following directory:
\USER\SIMOTION\HMI\SYSLOG\DIAG

Additional references

You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Storing diagnostics data and trace on the CF card

As of V4.5, the following data can be saved to the CF card with the **_saveTraceAndDiagnosticFiles** system function:

- Diagnostics data (same data as when the DIAG button is actuated)
- TO trace
- Device trace

The function can be used, for example, to collect diagnostic information (e.g. in the event of a fault) locally on the machine and without backing up SCOUT. The function saves the available data in a ZIP file, whereby the file name and the archive directory can be specified.

The packed data can be accessed, for example, via the web server.

Additional service and diagnostics options

SIMOTION Task Profiler application

The SIMOTION Task Profiler is a dedicated application installed parallel to SIMOTION SCOUT during SIMOTION SCOUT setup. The Task Profiler can be called in online mode using device diagnostics in SIMOTION SCOUT or the Windows application. In the event of a malfunction or error, the Task Trace data can be written to a directory or the CF card. They can then be processed using the Task Trace Viewer.

Additional references

For detailed information, see the *SIMOTION Task Trace* Function Manual.

Diagnostics via the SIMOTION IT web server

The SIMOTION D features an integrated web server.

In addition to customized websites and the option of performing firmware and project updates, SIMOTION IT web server provides comprehensive device and diagnostic information that can be called using a standard PC with a web browser.

Security concept of HTTP/S, FTP, and Telnet access on the web server

As of version V4.4, access to the SIMOTION IT web server is protected by a multi-level security concept.

The security state of the web server is indicated by the security level on the website. This security level can have three different stages: Low, normal, high.

Security Level Low

The device is supplied with an empty user database. No projects exist yet. The security level is low to allow configuration of the device.

- In this state, access to the web server as an anonymous user is possible to enable use of functions such as the project and firmware update or OPC XML.
- Access to the FTP and Telnet is also possible.
- New users can be entered in the empty user database.

In this state, series commissioning is possible via the web server.

NOTICE
Protecting the device Security level low should only be used for commissioning and service as otherwise the device is not adequately access protected.

Security Level Normal

The controller has a user database. There is a project on the controller; HTTP, HTTPS, FTP, and Telnet have been activated in the HW Config.

User password authentication is mandatory for access to websites with sensitive content (e.g. firmware update watch table, ...), FTP, and Telnet.

Security Level High

High security with maximum access protection:

HTTP, HTTPS, FTP, and Telnet have been deactivated via HW Config. Access to the Ethernet via the various ports of the services is then no longer possible.

State transition from security level low to normal

After taking delivery of the device, the user creates a project and loads it onto the device. This can be done by using the download functions of the SCOUT, by loading it directly onto the memory card (e.g. also via FTP), or via the Manage Config website, Device update tab.

Whichever method is used, the act of loading a project onto the device corresponds to a transition from security level low to security level normal from the point of view of the web server.

Resetting the security level from normal to low

If the user forgets to edit the UserDataBase.xml during initial commissioning, it will no longer be possible to access FTP, web services, or access-protected pages during use.

If there is no mechanical access to the memory card or the device, this can be achieved with the SCOUT function "Delete user data on card". After setting up the user administration, the project must be downloaded again.

Alternatives without SCOUT:

Setting the service selector switch to position "8" restores security level low (SIMOTION IT service mode). Using this method, the device can always be reset to security level low by hardware means. The following behavior must be taken into account:

- The "Service mode" is activated by turning the SVC switch to position "8".
- If the switch is already set to "8" at ramp-up, it is ignored (protection against forgetfulness).
- The service mode stops immediately when position "8" is exited.
- The service mode stops automatically after 120 minutes.
- It is possible to retrigger 120 minute timeout at any time by turning the switch briefly from "8" to "7" and back, for example.
- The service mode is indicated (as for underlicensing) through slow red flashing of the SF LED.

Note

As an alternative to switch position "8," security level low can also be activated via a simotion.ini file in the main directory of the CF card.

To achieve this, use a text editor (such as Notepad) to create a file called simotion.ini Add the following text: SERVICE_SELECTOR_MODE=8

You must use a text editor and may not use any formatting in the text.

To exit security level low again, undo the changes you have made.

Additional references

You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

10.1.1.10 Configuration of drive-related I/Os (without symbolic assignment)

Overview

The configuration of **local measuring inputs** is fundamentally different to the configuration of **global measuring inputs**.
The assignment of the local measuring inputs is permanently related to the hardware of the control unit and is performed

- On the drive side via the drive expert list and
- During the configuration of the TO measuringInput via the measuring input number.

Local and global measuring inputs have different properties. Detailed information on the differences can be found in the appendix,
- in Section Local and global measuring inputs (Page 7281).

Information about the configuration can be found

- For global measuring inputs (with symbolic assignment) in Section Configuration of global measuring inputs (Page 7163).
- For local measuring inputs in the appendix, in Section Configuration of local measuring inputs (Page 7283)

Additional references

Further information and programming examples for configuring drive-related I/Os without symbolic assignment can be found

- At the following Internet address (https://support.industry.siemens.com/cs/ww/en/view/29063656)
- In SIMOTION Utilities & Applications
  SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Local and global measuring inputs

Local and global measuring inputs

Depending on the used hardware platform, the following local and global measuring inputs are available for the measuring tasks:

- **Local measuring inputs** are axis-related and implemented in the SINAMICS drive. The actual position value is measured.
- **Global measuring inputs** can be freely assigned to the axes and add an internal time stamp to the measurement result for more precise determination of the axis positions.
  The term "central measuring input" is also used within the context of drives.

Table 10-79 Comparison of local and global measuring inputs

	Local measuring input	Global measuring input
Hardware supported	D410, D410-2, D4x5, D4x5-2 (terminal X122/X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2	TM15, TM17 High Feature, D410, D410-2, D4x5, D4x5-2 (terminal X122, X132, X142), CX32, CX32-2, CU310, CU310-2,CU320, CU320-2, ET 200SP/MP timer DIDQ
Measurement procedure	With a signal edge at the relevant input, the current actual values of an encoder connected to a Control Unit are measured with positioning accuracy to determine lengths and distances.	With a signal edge at the relevant input, the current actual values of one or more encoders are measured using time stamp functionality with positioning accuracy in order to provide information for determining lengths and distances (possible with any encoders included in the project).
Configuration of the measuring input TO in SIMOTION SCOUT	The assignment of inputs is always permanent depending on the hardware of the Control Unit and is performed during the configuration of the TO measuringInput using the measuring input number.	The assignment of inputs is not fixed depending on the hardware and is performed during the configuration of the TO measuringInput by means of symbolic assignment or the hardware address.

	Local measuring input	Global measuring input
TO measuringInput setting: Single measurement (Measurement jobs must be issued individually for each measurement. Several interpolation cycle clocks lie between two measurements.)	Yes	Yes
TO measuringInput setting: Cyclical measurement (The measurement is activated just once and runs cyclically until deactivated.)	No	Yes **D410, D410-2, D4x5, D4x5-2 (terminal X122, X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2:** The minimum interval between two measurements is three servo cycle clocks (max. two edges per measurement). **D4x5-2 (terminalX142), TM17 High Feature, ET 200SP/MP timer DIDQ:** The minimum interval between two measurements is one servo cycle clock (max. two edges per measurement). **TM15:** No cyclic measurement available
Use of multiple TO measuringInputs on one axis/encoder. They can be active concurrently	No	Yes
Listening TO measuringInput	No	Yes
Measuring on virtual axes	No	Yes
Measuring on axes attached to a different drive unit	No	Yes

SIMOTION Utilities & Applications includes, for example, a tool to estimate:

- The time between a measurement job being initiated and it being effective in the drive.
- The minimum time between two measurement jobs.

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

	Max. number of measuring input inputs	As local measuring input configurable	As global measuring input configurable
D410-2, CU310-2	8	x	x
D4x5-2 • X122/X132 • X142	 • 8 • 8	 • 8 • 0	 • 8 • 8
CX32-2	4	x	x
D410, CU310, CX32	3	x	x
D4x5, CU320	6	x	x
ET 200MP TM timer DIDQ 16x24V	8	-	x
TM15	24	-	x
TM17 High Feature	16	-	x

Configuring local measuring inputs

Properties

Local measuring inputs are always permanently assigned to an axis (drive). They are configured separately for each drive. The drive and the measuring input input must always be located on the same control unit. The measurement results are transferred using the axis message frame in accordance with the PROFIdrive profile. Message frame 39x does not need to be configured for local measuring inputs.

The settings for the use of the local measuring inputs must be made in the expert list.

Procedure

In order to use an I/O terminal on a D4x5-2, CX32-2 or SINAMICS control unit as measuring input input, proceed as follows:

1. Double-click the "Inputs/outputs" entry below the control unit in the project navigator.
2. Click the "Bidirectional digital I/Os" tab.
3. In this tab, configure the required I/O terminal as input. The configuration can also be set channel-granular on the p0728 parameter using the expert list of the control unit.

Specification of the measuring input input terminal must be made in the expert list of the respective drive for local measuring inputs.

Table 10-80 Local measuring inputs, required settings in the expert list (2)

	Parameters in the expert list of the drive	**Parameterization as**	
		D4x5-2, CU320-2	**CX32-2**
Specification of the input terminal of the measuring input in the expert list of the drive	p0488[0] (measuring input 1 input terminal, encoder 1)	DI/DO 8 or DI/DO 9 or DI/DO 10 or DI/DO 11 or DI/DO 12 or DI/DO 13 or DI/DO 14 or DI/DO 15	DI/DO 8 DI/DO 9 or DI/DO 10 or DI/DO 11
	p0488[1] (measuring input 1 input terminal, encoder 2)		
	p0488[2] (measuring input 1 input terminal, encoder 3)		
	p0489[0] (measuring input 2 input terminal, encoder 1)		
	p0489[1] (measuring input 2 input terminal, encoder 2)		
	p0489[2] (measuring input 2 input terminal, encoder 3)		

As a maximum of three encoders can be assigned to a drive, the index [0..2] specifies whether the measurement applies to encoder 1, 2, or 3.

The following must be taken into account:

- Only two TO measuringInputs can be configured per TO axis or TO externalEncoder
- Only one TO measuringInput can be active on a TO axis or TO externalEncoder.

Table 10-81 Local measuring inputs, configuration of the TO measuringInput

Axis measuring system no.	Under axis measuring system number, enter the number of the used encoder system (namely, encoder 1, 2 or 3). Encoder system 1 is the default setting.
Drive-related (local measuring input)	Activate the checkbox when a local measuring input is used.
Measuring input number	Enter here which measuring input is used (namely, 1 or 2). Input 1 is the default setting.

Detailed information can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs* Function Manual.

10.1.1.11 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

EMC limits in South Korea

	KC registration number: KCC-REM-S49-SIMOTION For sellers or other users, please keep in mind that this device in an A-grade electromagnetic wave device. This device is intended to be used in areas other than home. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

The EMC limits to be observed for Korea correspond to the limits of the EMC product standard for variable-speed electric drives EN 61800-3 of category C2 or the limit class A, Group 1 according to EN 55011. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. Additional measures, such as the use of an additional RFI suppression filter (EMC filter), may be necessary.

The measures for EMC-compliant design of the system are described in detail in this manual respectively in the Installation Guideline EMC.

Note that the final statement on compliance with the standard is provided by the respective label attached to the individual unit.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert).

Marking for Australia and New Zealand

	SIMOTION D4x5-2 incl. CX32-2 and CBE30-2 satisfy the requirement of the standard AS/NZS CISPR 16.
or	
	Marking with RCM (Regulatory Compliance Mark) or C-Tick with older components.

Marking for the Eurasian customs union

	EAC (Eurasian Conformity) Customs union of Russia, Belarus and Kazakhstan Declaration of conformity in accordance with the technical regulations of the customs union (TR CU).

Standards that are not relevant

	China Compulsory Certification SIMOTON D does not belong to the validity area of the China Compulsory Certification (CCC).

China RoHS

SIMOTION D complies with the China RoHS directive. You can find more information on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109738656).

SIMOTION D4x5-2 device-specific notes

Note regarding SIMOTION D

Note

The product standard EN 61800-3 describes the EMC requirements placed on "Variable-speed drive systems". As such, it defines different limits depending on the location of the drive system.

SINAMICS S120 power units are designed for use in the second environment. The term second environment refers to all locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

The same installation instructions apply for the SIMOTION D4x5-2/CX32-2 Control Units as for the SINAMICS S120 CU320-2 Control Units with regard to EMC.

It is essential to follow the installation instructions in the SINAMICS S120 Manuals in order to ensure compliance with emitted interference and immunity values.

For further information on this topic also refer to the *SIMOTION PM 21* Catalog as well as the SINAMICS Function Manuals.

10.1.1.12 ESD guidelines

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

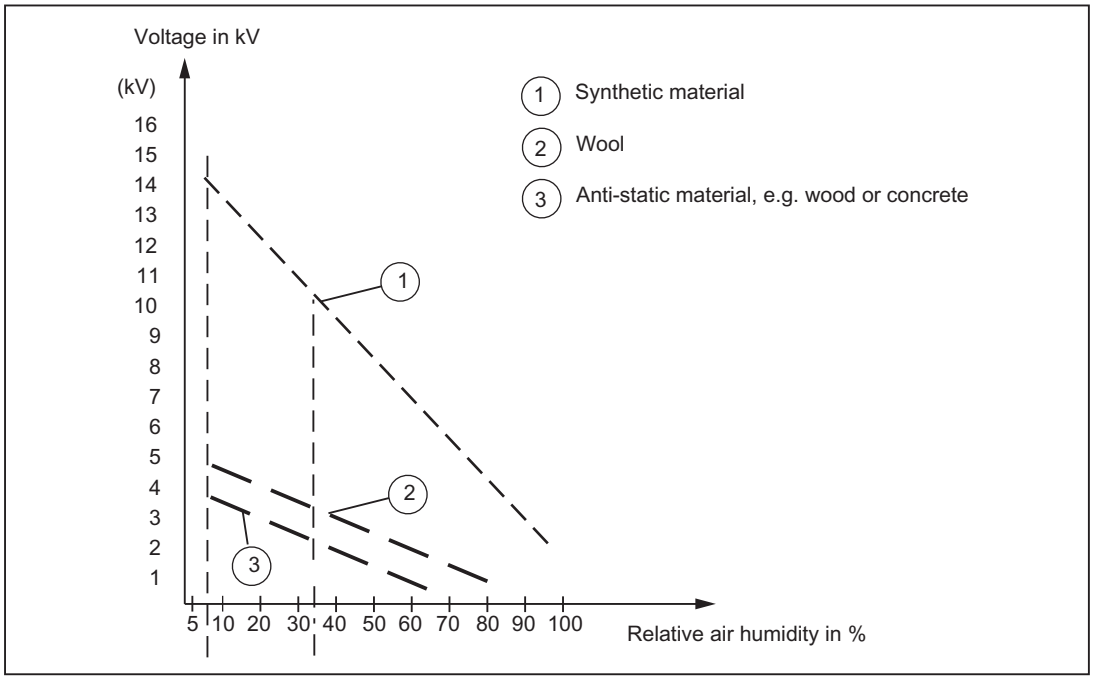


Figure 10-139 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

10.1.2 SIMOTION D410-2

Preface

Contents of the Commissioning and Hardware Installation Manual

This document is part of the **SIMOTION D** documentation package.

Scope

The SIMOTION D410-2 Commissioning and Hardware Installation Manual describes the commissioning and installation of the SIMOTION D410-2 DP and SIMOTION D410-2 DP/PN control units. A separate *SIMOTION D410* Commissioning Manual is available for the SIMOTION D410 DP and SIMOTION D410 PN Control Units.

The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 Version V5.x.
Information of configuration of the SIMOTION D Control Units in the Engineering Framework Totally Integrated Automation Portal (SCOUT in the TIA Portal), you will find in the configuration manual *SIMOTION SCOUT TIA*. The TIA Portal requires at least SIMOTION SCOUT V4.4 and SIMOTION D4xx-2 Control Units as of firmware V4.3.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Sections in this manual

The following is a description of the purpose and use of this Commissioning and Hardware Installation Manual:

- Description
  This section describes the SIMOTION system and its integration into the information landscape.
- Installing
  This section provides information on the the various installation options for the device.
- Connecting
  This section provides information about connecting and cabling the various devices, and about the communication interfaces.
- Commissioning (hardware)
  This section describes how to start up the device and what you must take into account.
- Parameter assignment / addressing
  This section describes how to integrate the SIMOTION D410-2 cm a project and how to configure the interfaces.
- Commissioning (software)
  This section describes how to configure a system and how to test the configured drives and axes.
- Service and maintenance
  This section describes how to replace a module, how to run updates, and how to modify settings.
- Diagnostics
  This section provides information on the service and diagnostics options, as well as LED states.
- Appendices with factual information for reference (for example, Standards and Approvals, ESD guidelines, etc.)
- Index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References

- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Disposal and recycling

SIMOTION D410-2 is an environmentally friendly product! It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner

Further information / FAQs

You can find further information on this manual at the following FAQ:

https://support.industry.siemens.com/cs/ww/de/view/27585482

The following information sources are also available:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling)
- The latest SIMOTION FAQs at https://support.industry.siemens.com/cs/ww/de/ps/14505/faq
- SIMOTION SCOUT online help
- For additional documentation, see the *Overview of SIMOTION documentation* (separate document).

Open source software

Third-party software - License conditions and copyright notes

Copyright notes for the third-party software contained in this product, in particular the open source software, as well as the applicable license conditions, can be found in the READ_OSS.ZIP file on the SIMOTION D CF card or in the corresponding firmware files.

Special note for resellers

The notes and the license conditions contained in the READ_OSS.ZIP file must be passed on to the purchaser in order to avoid the reseller and purchaser from violating the license conditions.

Source code availability

Some license terms of third-party software components used in this product may require us to provide you with the source code and other information for those components. You can find this information directly on or with the product (e.g. on mass storage devices, DVD). If this is not possible for technical reasons, Siemens will be happy to send you this OSS source code in exchange for reimbursement of the processing costs. Please contact the address provided at the end of this section.

Siemens AG

Digital Factory Customer Services

DI CS SD CCC TS

Gleiwitzer Str. 555

D-90475 Nuremberg, Germany

Internet (https://support.industry.siemens.com/cs/ww/en/ps)

Tel.: +49 911 895 7222

10.1.2.1 Safety instructions

Fundamental safety instructions

General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety information for SIMOTION D410-2

Observe the following safety information when working with SIMOTION D410-2 and its components!

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra low voltage in accordance with EN/IEC 609501 may be connected at all connectors and terminals.

⚠ WARNING

Danger to life from unexpected movement of machines on automatic restart

An automatic restart can be programmed for SIMOTION controllers. When the power returns, the axes start automatically.

Make sure this presents no hazard to personnel or property.

NOTICE

Damage to the CompactFlash card from electrical fields or electrostatic discharge

The CompactFlash card is an ESD-sensitive component.

De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF.

Comply with the ESD rules.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of systems / devices. Make sure the ventilation clearances of 50 mm are provided above and below the components. The ventilation openings may not be covered by connecting cables.

10.1.2.2 Description

System overview

SIMOTION D

SIMOTION D is a drive-based version of SIMOTION based on the SINAMICS S120 drive family.

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S120 drive software run on shared control hardware.

SIMOTION D is available in two versions:

- SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications.
- SIMOTION D4x5-2 is a Control Unit for multi-axis applications in the SINAMICS S120 booksize format.
  The following performance variants of the SIMOTION D4x5-2 Control Units are offered:

Control Unit	Performance variant	Range of applications
SIMOTION D425-2	BASIC performance	For up to 16 axes
SIMOTION D435-2	STANDARD performance	For up to 32 axes
SIMOTION D445-2	HIGH performance	For up to 64 axes
SIMOTION D455-2	ULTRA-HIGH performance	For up to 128 axes or applications with very short control cycles

Note

The SIMOTION D410-2 is described in this manual.

Separate manuals are available for the SIMOTION D4x5-2 and the SIMOTION D4x5 and SIMOTION D410 predecessor modules.

SIMOTION D is an integral part of the Totally Integrated Automation (TIA) concept. TIA is characterized by integrated data management, configuration, and communication for all

products and systems. Thus, an extensive toolbox of automation modules is also available for the SIMOTION D410-2.

Note

In order to cover all variants of SIMOTION D in blocksize format, the product will be referred to as "D410-2". Specific product designations will be used for information that applies only to one product version, e.g. D410-2 DP/PN.

SIMOTION D410-2

Figure 10-140 SIMOTION D410-2 DP (pictured on left), SIMOTION D410-2 DP/PN (pictured on right)

SIMOTION D410-2 is a compact Control Unit for single-axis applications.

The Control Unit is snapped directly on to the SINAMICS Power Module in blocksize format and has an integrated drive control for either one servo, one vector or one V/f axis.

SIMOTION D410-2 can be extended with additional SINAMICS S110/S120 control units (e.g. CU310-2) and so can also be used for smaller multi-axis applications (e.g. with 2 - 3 axes).

Example of a single-axis application

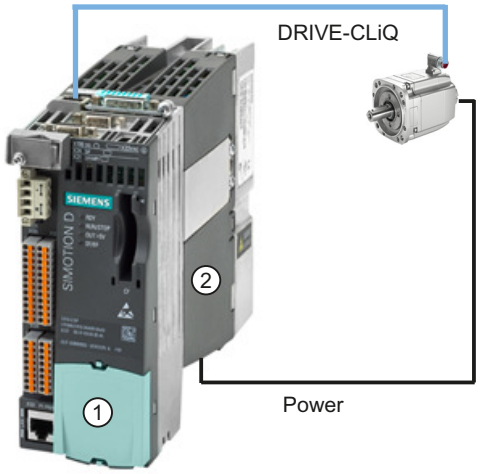


Figure 10-141 Application example with one axis

The example shows a single-axis application, consisting of a SIMOTION D410-2 (Control Unit) ① that is snapped directly on to the SINAMICS Power Module in blocksize format ②. The motors are supplied with power via the Power Module. The encoder is connected by means of DRIVE-CLiQ.

Example of a multi-axis application

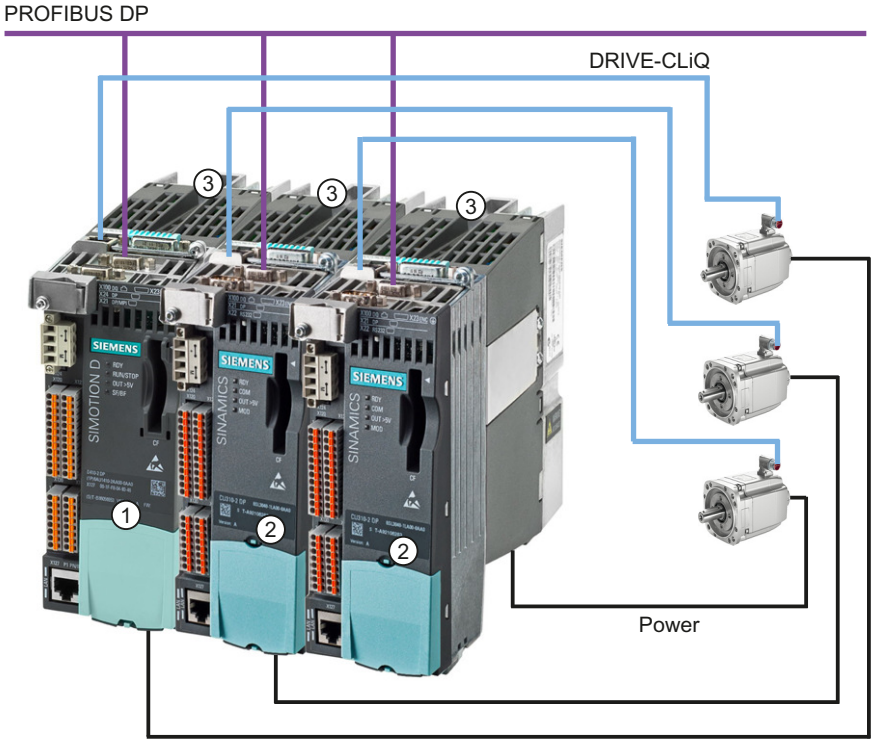


Figure 10-142 Application example with 3 axes

The example shows an application with 3 axes, consisting of:

- One SIMOTION D410-2 DP (Control Unit) ①, snapped on to the Power Module in blocksize format ③
  The SIMOTION D410-2 DP is snapped directly on to the SINAMICS Power Module. The motors are supplied with power via the Power Module. The encoder is connected by means of DRIVE-CLiQ.
- Two SINAMICS S120 CU310-2 DP ②, snapped onto a Power Module in blocksize format ③
  The Control Units are connected to the SIMOTION D410-2 DP via PROFIBUS DP. The two SINAMICS S120 CU310-2 DP are snapped directly on to the SINAMICS Power Module. The motors are supplied with power via the Power Modules. The encoders are connected by means of DRIVE-CLiQ.

Note

Path interpolation is supported as of V4.4.

Application

Combining a Power Module with SIMOTION D410-2 forms a compact single drive for machine and plant engineering.

Applications include:

- Machine concepts with central drive (e.g. presses, printing and packaging machines, etc.)
- Modular machine concepts where the machine modules were broken down into single axes
- Single drives with high accuracy, stability and concentricity requirements (compared with standard drives) in machine and industrial plant engineering
- Single drives for transport tasks (conveying, raising, lowering)
- Single drives with integrated PLC functionality and expanded motion control functionality such as output cams or cams
- Drives without power recovery (wire drawing, extruding)
- Drive connections with high availability requirements (incoming supply failure may not cause all axes to fail)
- Small multi-axis groupings (typically 2 to 3 axes) based on SINAMICS S110/120 blocksize.

Hardware components

As central hardware the SIMOTION D410-2 Control Unit is made up of the SIMOTION runtime system and the SINAMICS drive control.

A range of additional SINAMICS S120 components, such as SMx encoder systems or Terminal Modules can be connected via DRIVE-CLiQ.

With a few exceptions (e.g. no BOP20 Basic Operator Panel, etc.), the drive control integrated in SIMOTION D410-2 has the same control properties and performance features as the SINAMICS S120 CU310-2 Control Unit.

Extension of the drive computing performance

To fully utilize the motion control performance of a SIMOTION D410-2 when required, the drive-side computing performance can be extended by connecting additional SINAMICS S/G Control Units (e.g. CU305, CU310-2, CU320-2, CU250S-2, etc.) via PROFIBUS or PROFINET to the SIMOTION D410-2.

Software components

The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following:

- The SIMOTION runtime system with the following functions:
  - Freely programmable runtime system (IEC 61131)
  - Various runtime levels (tasks)
  - PLC and arithmetic functionality
  - Motion control functions
  - Communication functions
- The SINAMICS S120 drive control with the following functions:
  - Closed-loop current and torque control
  - Closed-loop speed control

System components

Overview

SIMOTION D410-2 communicates with the components of the automation landscape via the following interfaces:

- PROFIBUS DP (D410-2 DP and D410-2 DP/PN)
- PROFINET IO (D410-2 DP/PN only)
- Ethernet
- DRIVE-CLiQ (DRIVE Component Link with IQ)
- Power Module interface (PM-IF)

SIMOTION D features a SINAMICS Integrated drive element. Communication with the SINAMICS Integrated is via PROFIBUS mechanisms (DP Integrated), via PROFIdrive telegrams.

Shorter cycle times and greater numbers of addresses for each node are achieved with the "DP Integrated" compared to the "external PROFIBUS DP."

The most important components of the system and their functions are shown below.

Table 10-82 System components

Component	Function
SIMOTION D410-2	... is the central motion control module. The module contains the programmable SIMOTION runtime of SIMOTION D410-2 and the SINAMICS S120 drive runtime software. You can use the integrated high-speed I/Os (onboard I/Os) as: • User-addressable process I/Os • Homing inputs • Fail-safe digital inputs • Fail-safe digital output • Inputs for measuring inputs • Outputs for fast output cams • Analog input The measuring sockets can output any analog signals. The DRIVE-CLiQ interface permits a fast connection to the SINAMICS drive components.
System software	The basic functionality of SIMOTION D410-2 is supplied separately on a CompactFlash Card containing the following: • SIMOTION runtime (kernel) • Drive software of SINAMICS S120 The CompactFlash card is not included in the scope of delivery.
Power supply (PS)	... provides the electronic power supply for SIMOTION D410-2 (e.g. SITOP power supply).

PROFIBUS DP

SIMOTION D410-2 can communicate with the following components via the PROFIBUS DP interface.

Table 10-83 Components on PROFIBUS DP

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of the SIMOTION D410-2.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.

Component	Function
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200S	Finely scalable I/O system for cabinet installation and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/IP67 degree of protection for machine-related applications with no cabinet; with features such as more compact designs, integrated PROFIsafe safety technology, PROFINET connection, and live module replacement.
SIMATIC ET 200eco	I/O system with IP65/IP67 degree of protection for machine-related applications with no cabinet, with a flexible and fast connection system in ECOFAST or M12.
Other PROFIBUS I/O	
Gateways	• DP/AS-Interface Link 20E and DP/AS-Interface Link Advanced for the PROFIBUS DP gateway to AS-Interface • DP/DP coupler for connecting two PROFIBUS DP networks
Drive interfaces	• ADI4 (Analog Drive Interface for 4 axes) for the connection of drives with analog ±10 V setpoint interface or for external encoders • IM 174 (Interface Module for 4 axes) for the connection of drives with analog ±10 V setpoint interface, external encoders or the connection of stepper drives with pulse/direction interface
Drive units with PROFIBUS DP interface (e.g. CU310-2 DP)	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors. Can also be operated as an isochronous slave on the PROFIBUS DP.
Teleservice adapter	Remote diagnostics

PROFINET IO

The SIMOTION D410-2 DP/PN can communicate with the following components via the onboard PROFINET IO interface.

Table 10-84 Components on the PROFINET IO

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Master computer	... communicates with other devices via UDP, TCP/IP.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data. With the time-based I/O, signals can be recorded or output to the precise µs.

Component	Function
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety. With the time-based I/O, signals can be recorded or output to the precise µs.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET IO connection and live module replacement.
SIMATIC ET 200eco PN	Compact block I/O with IP65/66/67 degree of protection for cabinet-free usage in machines with M12 connection method. Very rugged and resistant encapsulated metal enclosure.
Other PROFINET IO I/O devices	
Drive units with PROFINET IO interface	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors.
Gateways	• IE/AS-Interface link PN IO for the PROFINET IO gateway to AS-Interface • PN/PN coupler for connecting two PROFINET IO networks

Ethernet

The Control Unit can communicate with the following components via the Ethernet interfaces or be embedded in an automation environment:

Table 10-85 Components on the Ethernet

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
Master computer	... communicates with other devices via UDP, TCP/IP.
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of the SIMOTION D410-2.

DRIVE-CLiQ

SIMOTION D410-2 can communicate via the DRIVE-CLiQ interface with the following components:

Table 10-86 Components on DRIVE-CLiQ

Component	Function
SINAMICS S120 AC DRIVE drive units (with CUA31/CUA32)	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors. The Power Module is connected via CUA31/CUA32. No more than one Power Module can be connected. The chassis Power Module is connected via DRIVE-CLiQ. **Note**: Components in booksize format are not supported!
TM15, TM17 High Feature Terminal Modules	The Terminal Modules TM15 and TM17 High Feature are used to implement measuring inputs inputs and output cam outputs. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times.
TM31 Terminal Module	... enables terminal expansion via DRIVE-CLiQ (additional analog and digital I/Os).
TM41 Terminal Module	... enables terminal expansion (analog and digital I/Os) and encoder simulation via DRIVE-CLiQ. The TM41 can be connected to a real axis.
TM54F Terminal Module	... enables terminal expansion (fail-safe digital inputs/outputs) for controlling the safe motion monitoring functions of the integrated drive. A TM54F is not usually necessary because the SIMOTION D410-2 has 3 F-DI and 1 F-DO.
TM120 Terminal Module	Four temperature sensors (KTY84-130 or PTC) can be evaluated via the TM120 Terminal Module. The temperature sensor inputs are safely electrically separated from the evaluation electronics in the TM120 Temperature Module and are suitable for evaluating the temperature of special motors, e.g. 1FN linear motors and 1FW6 built-in torque motors.
TM150 Terminal Module	The TM150 Terminal Module can be used to evaluate temperature sensors (KTY, PT100, PT1000, PTC, and bimetal normally closed contact). This means, for example, that other temperatures from the process can be measured in addition to the motor temperature. Temperature sensors can be evaluated using a 2, 3 or 4-wire system. Twelve temperature sensors can be evaluated with 2-wire evaluation and six temperature sensors with 3 and 4-wire evaluation.
SMx Sensor Modules	... enable acquisition of encoder data from connected motors via DRIVE-CLiQ.
Motors with DRIVE-CLiQ interface	... allow simplified commissioning and diagnostics, as the motor and encoder type are identified automatically.
DMC20/DME20 DRIVE-CLiQ hub	... enables the number of DRIVE-CLiQ interfaces to be increased and the creation of a point-to-point topology.

Note

Please note that SIMOTION D410-2 components in booksize format (Controller Extension, Motor Modules, Line Modules, etc.) are not supported.

SIMOTION D410-2 can only be used with the following Power Modules:

- PM340
- PM240-2 as of SIMOTION V4.4/SINAMICS V4.7

Other Power Modules are not supported by SINAMICS G120 (e.g. PM230).

Note

You will find detailed information on components in the SINAMICS S110/S120 family of products in the SINAMICS S110/S120 manuals.

It is possible that older DRIVE-CLiQ components can no longer be used with SIMOTION D410-2. You will find detailed information on this in the SIMOTION D410-2 Commissioning and Hardware Installation Manual in Section "Migration of SIMOTION D410 to SIMOTION D410-2" under "Permissible combinations".

See also

Permissible combinations (Page 7511)

I/O integration

Note

Note that not all modules in the ET 200 I/O family are approved for SIMOTION. Moreover, system-related functional differences can come into play when these I/Os or I/O systems are used on SIMOTION vs. on SIMATIC. For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

A detailed, regularly updated list of the I/O modules approved for use with SIMOTION, as well as notes on their use, can be found at Internet address (https://support.industry.siemens.com/cs/ww/en/view/11886029)

In addition to the I/O modules enabled for SIMOTION, in principle all certified standard PROFIBUS slaves (DP-V0/DP-V1/DP-V2) and PROFINET IO devices with RT and IRT real-time classes may be connected to SIMOTION D410-2. These modules are integrated using the GSD file (PROFIBUS) or GSDML file (PROFINET) provided by the relevant device manufacturer.

Note

Please note that in isolated cases, additional boundary conditions must be fulfilled in order to integrate a module into SIMOTION. Thus, a few modules require "driver blocks" , e.g. in the form of function blocks, that permit (or simplify) integration.

For modules enabled for SIMOTION (e.g. SIMATIC S7-300 module FM 350-1, etc.), these driver blocks are part of the SIMOTION SCOUT engineering system command library.

Commissioning software

Requirement

To create and edit projects on your PG/PC, you need the SIMOTION SCOUT commissioning and configuration tool.

For information on how to install SIMOTION SCOUT, see the *SIMOTION SCOUT* Configuration Manual.

Note

SIMOTION SCOUT contains the functionality of STARTER and SIMATIC S7-Technology.

Simultaneous operation of SIMOTION SCOUT, STARTER and SIMATIC S7-Technology as a single installation on one PG/PC is **not** possible.

Integrated STARTER

You can insert a standalone drive (e.g. SINAMICS S120) with the "Insert single drive unit" element in the project navigator. It is commissioned using wizards in the working area of the workbench that contains the STARTER functionality.

SINAMICS Support Package (SSP)

The following SSPs are relevant for SIMOTION SCOUT:

- SSP "SINAMICS" for single drive units (e.g. CU3xx)
- "SIMOTION SINAMICS Integrated" SSP for the SINAMICS drive integrated into SIMOTION D410-2.

You will find detailed information on the SSPs in the readme files and in the software compatibility list at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Upgrading SIMOTION D410-2 projects and hardware

Projects that you have created for one SIMOTION D410-2 firmware version can also be converted for other firmware versions. See, for example, Section Service and maintenance (Page 7515).

SIMOTION IT web server

The SIMOTION D410-2 features an integrated web server.

The web server supports the display of diagnostics and system data in standard Internet browsers, even in the absence of an engineering system, and the carrying out of project/firmware updates.

Additional references

You will find detailed information on working with projects in the *SIMOTION SCOUT* Configuration Manual.

You will find detailed information on SIMOTION IT web server in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

10.1.2.3 Installing

General requirements

Mounting options

SIMOTION D410-2 comes in two designs:

- Mounting on the Power Module in blocksize format (mounted use).
- Mounting on a mounting plate (detached use).

Open components

SIMOTION D410-2 is an open device! This means, you can only install the modules in housings, cabinets or electrical equipment rooms. These may only be accessible via key or tool. Housings, cabinets, or electrical equipment rooms may only be accessed by trained or authorized personnel. An external fire protection casing is required.

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

Note

The components must be protected against conductive contamination, e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12.

If conductive contamination can be excluded at the installation site, a lower degree of cabinet protection may be permitted.

Mounting the SIMOTION D410-2 on the power module

Overview

A SIMOTION D410-2 can be snapped directly on to a SINAMICS S120 Power Module in blocksize format via the PM-IF interface. Power Modules PM340 and PM240-2 (PM240-2 as of SIMOTION V4.4/SINAMICS V4.7) can be used. Operation with SINAMICS G120 PM2x0 Power Modules or booksize Motor Modules is not possible.

Note

You can connect a Power Module in blocksize format to the DRIVE-CLiQ interface of SIMOTION D410-2 using the CUA31/CUA32 adapter module. Power modules in chassis format AC/AC are connected to SIMOTION D410-2 via the DRIVE-CLiQ interface of the Power Module.

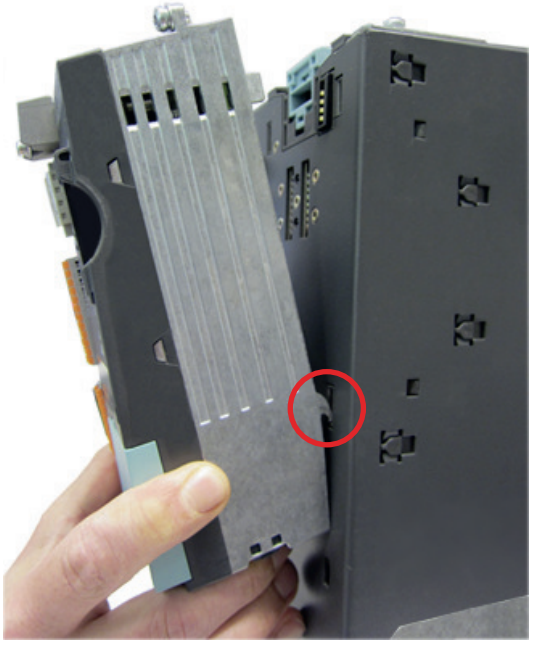
Requirement

As soon as the Power Module is properly installed, you can mount the SIMOTION D410-2 on the Power Module.

Note

Note the information in the *SINAMICS S120 AC Drive*Manual when commissioning the Power Module.

Mounting on the Power Module



Snapping the SIMOTION D410-2 on to the Power Module (example PM340)

PM340 Power Module with SIMOTION D410-2 (example PM340)

Disassembling the SIMOTION D410-2

To remove the SIMOTION D410-2 from the Power Module, the blue release, as shown in the figure, must be pressed down and SIMOTION D410-2 tilted forward.

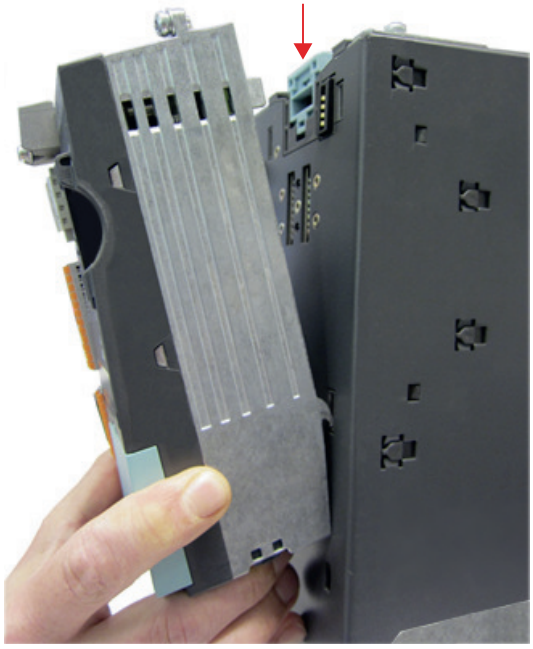


Figure 10-143 Removing the SIMOTION D410-2 from the Power Module

Mounting the SIMOTION D410-2 on the mounting plate

Overview

Using a mounting plate, SIMOTION D410-2 can be operated separately, which means it is not mounted directly on the Power Module.

Application examples:

- Offset installation using CUA31/CUA32 Control Unit Adapter (e.g. for reducing the mounting depth or in order to use additional interfaces on the CUA).
  The SIMOTION D410-2 is snapped on to the mounting plate and connected to the Power Module in blocksize format via DRIVE-CLiQ using CUA31/CUA32. No more than one Control Unit Adapter can be connected to the SIMOTION D410-2.
- Using SIMOTION D410-2 without a Power Module.
  A SIMOTION D410-2 mounted on the mounting plate is operated without the Power Module (e.g. for hydraulic applications).

Requirements

You must order the mounting plate for the SIMOTION D410-2 separately. For the article number, see Section "Spare parts and accessories" in the *SIMOTION D410-2* Manual.

Mounting on mounting plate

1. The mounting plate is attached to the control cabinet.
2. The SIMOTION D410-2 is snapped on to the mounting plate.

Figure 10-144 Mounting the SIMOTION D410-2 on the mounting plate

Note

When operating away from the Power Module, generally the power supply must be realized via the power supply connection (X124).

It is also not possible to use the Safety Integrated Extended and Advanced Functions via the onboard terminals (F-DI, F-DO) in distributed operation.

Observe the information on control cabinet installation in the *SINAMICS S120 AC Drive* Manual.

Mounting the SIMOTION D410-2 in the power module chassis

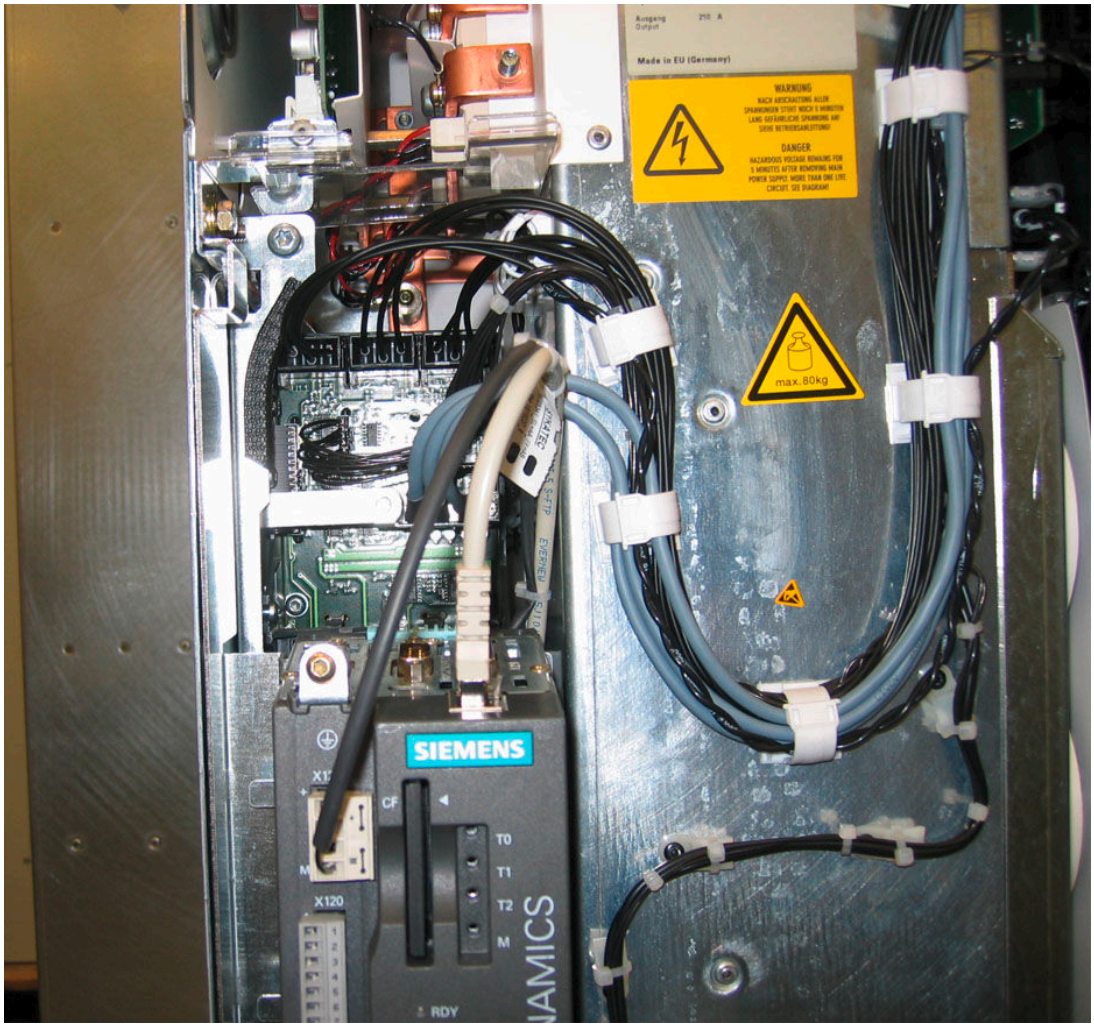


Figure 10-145 Mounting the CU310, for example, in the power module chassis, FX frame size

The DRIVE-CLiQ cable and the cable for the 24 V supply must be correctly routed so that the front flap can close.

Note

For the power module, a connecting cable is also supplied for the power supply of the SIMOTION D410-2. This cable must be connected to the SIMOTION D410-2.

10.1.2.4 Connecting

General Overview

Overview

The SIMOTION D410-2 has a number of interfaces that can be used for connecting the power supply and for communication with the other components of the system.

- The different SINAMICS components are interconnected via DRIVE-CLiQ.
- Actuators and sensors can be connected to the inputs/outputs.
- The SIMOTION D410-2 can be connected to PROFIBUS DP, MPI, Ethernet, and PROFINET (D410-2 DP/PN only) for communication purposes.

Arrangement of the interfaces on the device

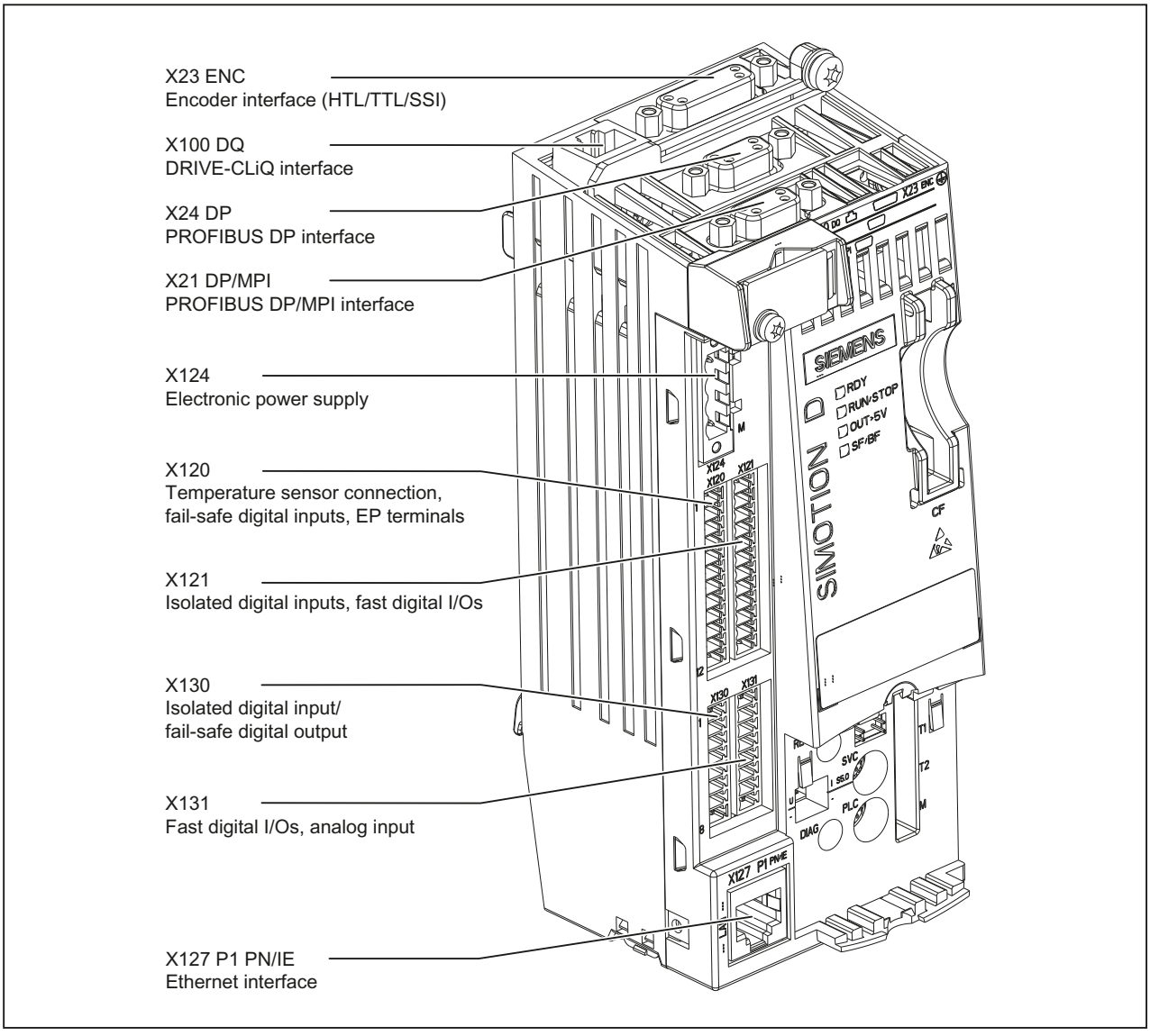


Figure 10-146 SIMOTION D410-2 DP, arrangement of the interfaces on the device

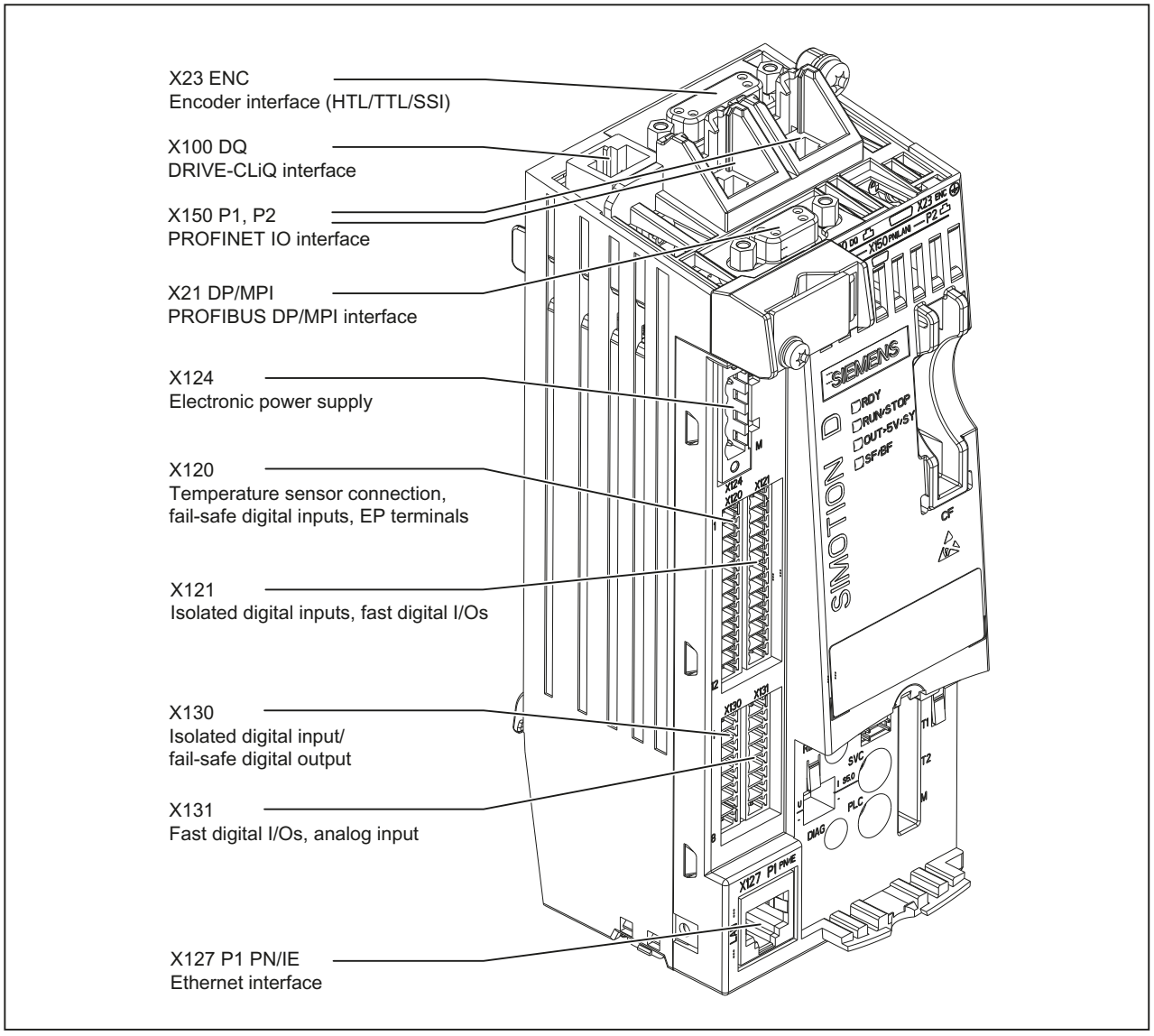
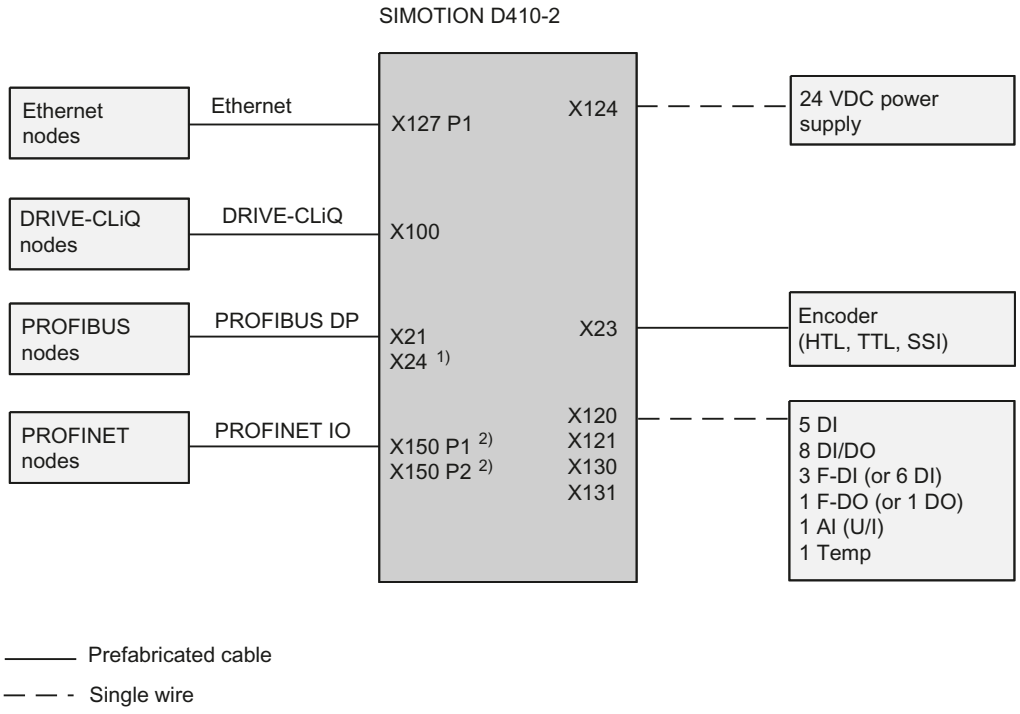


Figure 10-147 SIMOTION D410-2 DP/PN, arrangement of the interfaces on the device

Overview of connections

The following overview shows examples of the various interfaces and their connection possibilities.



1) X24 only with SIMOTION D410-2 DP
2) X150 P1, P2 only with SIMOTION D410-2 DP/PN

Figure 10-148 SIMOTION D410-2 connection possibilities

General rules for operating the SIMOTION D410-2

When integrating the SIMOTION D410-2 into a system, you must observe the following general rules.

System startup after certain events

Observe the following rules when starting up a system following certain events:

- During startup after a voltage dip or power failure, no dangerous operating states may result. If necessary, force an EMERGENCY STOP.
- At startup, after unlocking the EMERGENCY STOP apparatus, there must not be an uncontrolled or undefined startup.

Supply voltage

Observe the following rules for the supply voltage:

- For stationary installations or systems that do not have all-pole line disconnect switches, the building installation must include a line side switch or a fuse.
- For load power supplies and power supply modules, the rated voltage range set must correspond to the local line voltage.
- For all electrical circuits, the fluctuation/deviation of the line voltage from the rated value must be within the permitted tolerance (refer to the technical data of the used components)

24 V DC supply voltage

Observe the following rules for the 24 V supply:

- Provide lightning protection (e.g. lightning protection devices):
  - External lightning protection is required for buildings.
  - Internal lightning protection must be provided for 24 V DC supply cables and signal cables.
- Ensure safe (electrical) isolation of the extra-low voltage for the 24 V supply.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Implement the 24 V DC supply with protective separation as protective extra-low voltage.

Disconnection of 24 V plug-in connections during operation

Observe the following safety notice when using SIMOTION D410-2:

⚠ WARNING

Personal injury and damage to property can occur

Personal injury and property damage can occur if 24 V plug-in connections are disconnected during operation. Disconnection of 24 V plug-in connections is only permitted when the power is off.

Protection against external electrical interference

Observe the following rules for protection against electric effects or faults:

- For all plants or systems, in which SIMOTION is installed, the plant or system must be connected to a protective conductor for the discharge of electromagnetic interference.
- The wiring arrangement and installation must comply with EMC regulations for the supply, signal and bus cables.
- A cable or wire break cannot lead to undefined states in the plant or system for signal or bus cables.

Rules for current consumption and power loss in an installation

The power loss of all components used in a cabinet must not exceed the maximum amount that can be dissipated from the cabinet.

Note

When designing the control cabinet, ensure that the temperature inside the cabinet does not exceed the permitted ambient temperature for the components even at high external temperatures.

Additional references

The same installation instructions apply for the SIMOTION D410-2 control unit as for the SINAMICS S120 CU310-2 control unit with regard to EMC.

Refer to *SINAMICS S120 for AC Drives* Manual.

Electromagnetic compatibility

General

Electromagnetic compatibility (EMC) means that the devices function satisfactorily, without interfering with other devices and without being disrupted by other devices. The EMC requirements for variable-speed drive systems (PDS, Power Drive System) are described in the product standard IEC/EN 61800-3.
A variable-speed drive system consists of the Control Unit and Power Module as well as the relevant electric motors and encoders, including connecting cables. The driven machine is not part of the drive system.

Note

PDS as component of machines or systems

For the integration of PDS into machines or systems, additional measures may be required so that the product standards of these machines or systems are complied with. The machine or system builder is responsible for taking these measures.

Environments and categories

The EMC environments and categories are defined in the EMC Product Standard EN 61800-3, as follows:

Environments

IEC/EN 61800-3 differentiates between the first and second environments - and defines different requirements for these environments.

First environment

Residential buildings or locations at which the drive system is directly connected to a public low-voltage line supply without intermediate transformer.

Second environment

All locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

Categories

IEC/EN 61800-3 differentiates between four drive system categories:

Category C1

Drive systems for rated voltages < 1000 V for unrestricted use in the first environment.

Category C2

Stationary drive systems for rated voltages < 1000 V for operation in the second environment. The drive system must be installed by appropriately qualified and trained personnel. Additional measures are required for operation in the first environment.

Category C3

Drive systems for rated voltages < 1000 V, only for operation in the second environment.

Category C4

Drive systems for IT line supplies for operation in complex systems in the second environment. An EMC plan must be drawn up.

SIMOTION

SIMOTION D is designed for industrial use in accordance with Category C2.

Interference immunity

SIMOTION D and SINAMICS S120 are suitable for operation in the second environment. With regard to interference immunity, SIMOTION D and SINAMICS S120 can be used in the first and second environments.

Emitted interference

With regard to interference emission, to comply with the limit values in accordance with IEC/EN 61800-3 second environment, Category C2; in particular for the drive system, certain measures must be observed (EMC-conform installation by appropriately qualified and trained personnel, possibly required radio interference suppression filters, etc.) Observe the information on electromagnetic compatibility (EMC).

EMC notes

Note

Further information can be found in chapter "Control cabinet installation and EMC" in the SINAMICS S120 AC-Drive Equipment Manual.

Note

Requirements for implementing EMC are listed in EN 61000-6-2, EN 61000-6-4, EN 61800-3, EN 60204-1 and in the "EMC Installation Guidelines" Configuration Manual. (https://support.industry.siemens.com/cs/ww/de/view/60612658)

Note

Conformance with the EMC Directive of the EC is ensured by following the measures described in the "EMC Installation Guideline" Configuration Manual.

Protective conductor connection and potential equalization

Requirement

You have mounted the Control Unit in the control cabinet.

The SIMOTION D410-2 control unit has a protective conductor connection (M4 screw, Torx T20). This connection is also for the connection of an equipotential bonding conductor.

Note

Requirements for functional safety for machines and systems; reliability and EMC are only guaranteed with original SIEMENS cables.

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

Protective conductor connection

SIMOTION D and the SINAMICS S120 drive system are designed for use in cabinets with a protective conductor connection.

All system components and machine parts must be incorporated in the protection concept.

The drive line-up must be arranged on a common bright mounting plate ① in order to comply with the EMC limit values. The connection ② establishes a low-impedance connection to the mounting plate.

The mounting plate must be connected to the protective conductor connection of the control cabinet. For this purpose, a connection ③ must be established to the protective conductor bar ④. The protective conductor bar ④ must be connected to the protective conductor ⑤.

The protective connection (PE connection) that is used for the motors ⑥ must be established through the motor cable.

For EMC reasons, the motor cable shield must be laid flat at both the Motor Module (MM) / Power Module (PM) and motor.

A protective connection ⑧ must also be established for components that are **not** connected with low impedance (e.g. control cabinet door via hinges ⑦).

Example: Booksize axis grouping comprising Control Unit (CU), Line Module (LM), and Motor Modules (MM) as well as drive in blocksize format comprising a Power Module (PM) with snapped on Control Unit (CU)

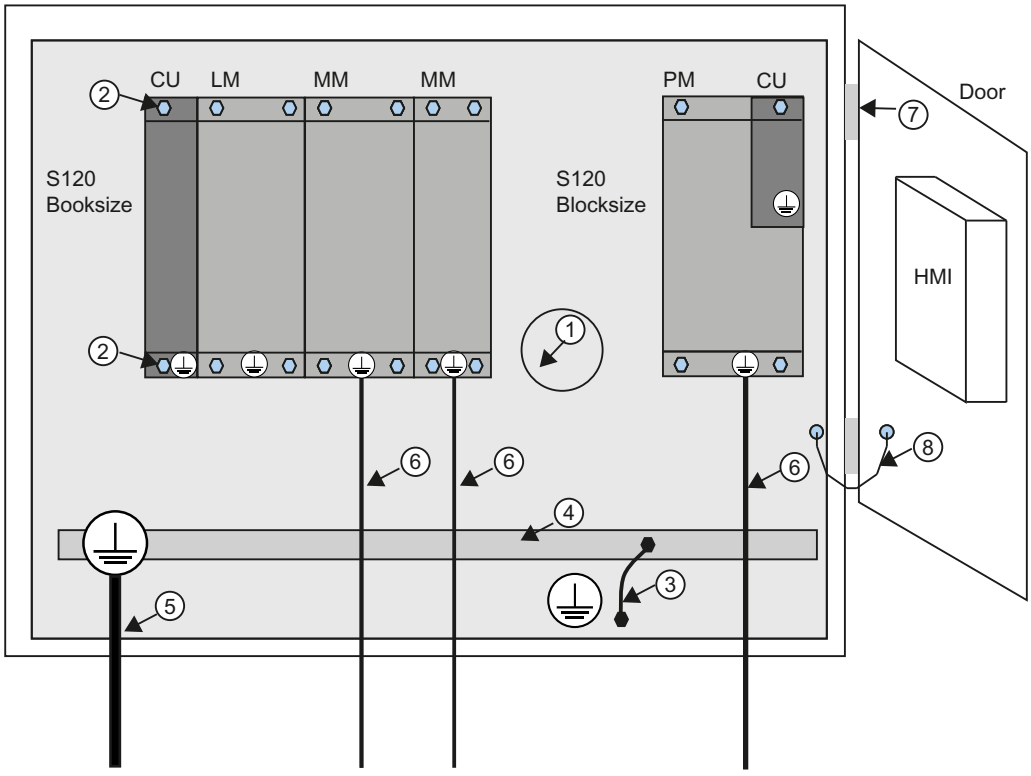


Figure 10-149 Protective conductor connection, cabinet with mounting plate / equipotential bonding surface

The protective conductor connections must be dimensioned as follows:

Table 10-87 Conductor cross-section for copper protective connections

Line supply cable in mm²	Copper protective connections in mm²
Up to 16 mm²	The same as the line supply cable
From 16 mm² to 35 mm²	16 mm²
From 35 mm²	0.5 x line supply cable

For materials other than copper, the cross-section should be increased so that as a minimum, the same conductivity is attained.

Equipotential bonding

A mounting plate serves simultaneously as an equipotential bonding surface. This means that no additional equipotential bonding is required within the drive line-up. If a common bright mounting plate is not available, then equally good equipotential bonding ⑨ must be established using cable cross-sections as listed in the table above or, as a minimum, with the same conductivity.

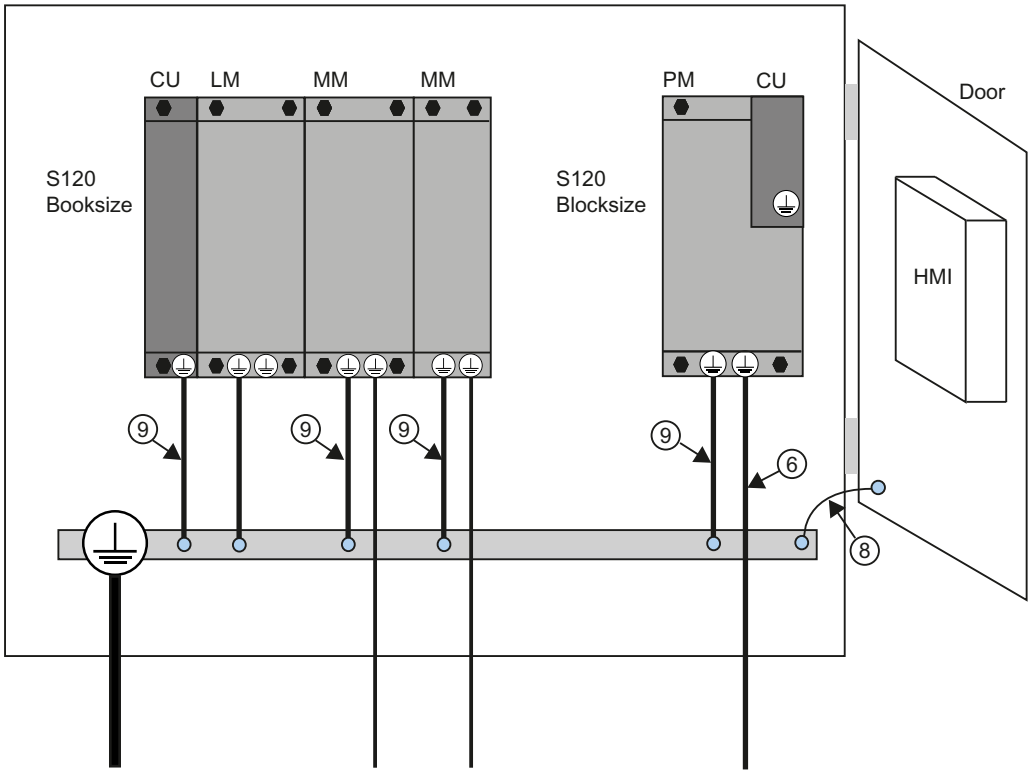


Figure 10-150 Protective conductor connection, cabinet without equipotential bonding surface

Note

Ground potential and housing (PE) for the SIMOTION D410-2 are connected internally with low impedance.

Communication links

Within the same control cabinet, no equipotential bonding conductor is required for fieldbus components if they installed as described above.

You have to ensure equipotential bonding for communication connections between remote components in a system (e.g. devices in different control cabinets) as well as between buildings or building sections.

If, for example, data cables (PROFIBUS, PROFINET, Ethernet or DRIVE-CLiQ) are routed through several control cabinets, equipotential bonding must be established with an equipotential bonding conductor. Install the equipotential bonding conductor together with the data cable.

The following minimum cross-sections are required according to IEC 60364-5-54:

- For copper, at least 6 mm^2
- For aluminum, at least 16 mm^2
- For steel, at least 50 mm^2

NOTICE
Fault of the data link or device defect with missing equipotential bonding
Considerable leakage currents can flow via the data cable if equipotential bonding is not provided. Disturbance of the data link or device faults may result.
Install an equipotential bonding conductor together with the data cable.

Due to the maximum length of 100 m for PROFIBUS copper cable at 12 Mbit/s or for PROFINET copper cable and due to the electrical isolation, EMC protection and equipotential bonding aspects, it is recommended that fiber-optic cables be used for connections between buildings.

Additional information

Further information on the protective connection and equipotential bonding can be found in the following references:

- SINAMICS drive system: See the SINAMICS manuals
- PROFIBUS and PROFINET: See the following Internet address (http://www.profibus.com) (at Downloads)

Connecting the power supply

Safety rules

Because of the wide range of possible applications, only the basic rules for electrical installation are described in this section. At a minimum, you must comply with these basic rules to ensure problem-free operation.

Rules for safe operation

In order to ensure safe operation of your equipment, implement the following measures and adapt them to suit your particular conditions:

- An EMERGENCY-OFF concept based on applicable technical specifications (e.g. European Standards EN 60204, EN 418 and associated standards).
- Additional measures to limit the travel of axes (e.g. hardware limit switch).
- Equipment and measures to protect motors and power electronics according to the SINAMICS Installation Guidelines.

In order to identify hazards, we also recommend that a risk analysis be conducted on the entire system in accordance with the basic safety requirements set out in Appendix 1 of the EU machinery directive.

Additional references

- Guidelines on Handling Electrostatic Sensitive Devices (ESD), see Appendix ESD guidelines (Page 7576) in this manual.
- For installing a system with SIMATIC ET 200 I/O (e.g. ET 200SP, ET 200 MP, etc.), see the manuals for the relevant ET 200 I/O systems.
- For further information on EMC, we recommend the *EMC Installation Guidelines* / *Basic System Requirements* Configuration Manual
  Download: https://support.industry.siemens.com/cs/ww/en/view/60612658

Standards and regulations

You must comply with the applicable VDE guidelines when wiring the SIMOTION D410-2, in particular VDE 0100 and VDE 0113 for disconnecting devices, short-circuit and overload protection.

Wiring the power supply

If no digital outputs are used or if the SIMOTION D410-2 is being operated on a mounting plate, the SIMOTION D410-2 can be supplied by the Power Module via the PM-IF interface.

If digital outputs are used, a 24 V load power supply must be connected to the X124 screw-type terminal block.

Note

If a digital output is parameterized and the 24 V load power supply is not connected (or the level is too low), alarm A03506 is issued on the SINAMICS side (can also be parameterized as a fault).

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Implement the 24 V DC supply with protective separation as protective extra-low voltage.

Wiring the screw-type terminal block

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

Use flexible cables with a cross-section of at least 1 mm^2 when you wire the power supply.

Wire-end sleeves are required if you wire several cables per connection.

1. Strip the cable ends.
2. Put on the wire-end sleeve if applicable.
3. Plug the cable end (with wire-end sleeve) into the screw-type terminal connection.
4. Tighten the mounting screw.
5. Plug the screw-type terminal with cables into the X124 connection.
6. Tighten the screw-type terminal block with a flat-bladed screwdriver.

Note

The 24 V DC cable must be approved for temperatures of up to 75 °C.

The maximum permissible cable length is 10 m.

Reverse polarity protection

The LED "RDY" lights up green when the power supply is connected correctly and switched on.

Note

The control does not run if the polarity is reversed. However, integrated reverse polarity protection protects the electronics from damage.

Fuse

If the control is defective, an internal fuse protects the electronics from consequential damage (e.g. fire). In this case, the module must be replaced.

Connecting DRIVE-CLiQ components

Overview

DRIVE-CLiQ connects the components in the SINAMICS S120 drive family as well as SIMOTION D410-2. DRIVE-CLiQ is a communications system, which enables SIMOTION D410-2 to automatically detect connected components. DRIVE-CLiQ provides a wiring tree, the topology of which can be visualized in SIMOTION SCOUT.

Information on the components that can be connected to DRIVE-CLiQ can be found in Section"DRIVE-CLiQ interface" of the *SIMOTION D410-2* Manual.

Rules for wiring DRIVE-CLiQ

The following rules must be followed for wiring DRIVE-CLiQ:

- Ring wiring is not permitted.
- Components must not be double-wired.

You will find detailed information about DRIVE-CLiQ wiring in the *SINAMICS S120 Control Units and Additional System Components* Manual.

Procedure

Connect the SIMOTION D410-2 X100 socket to the corresponding sockets on the drive components using the DRIVE-CLiQ signal cable (motor with DRIVE-CLiQ interface, TM and SMx modules).

Note

Please note that SIMOTION D410-2 components in booksize format (controller extension, motor modules, line modules, etc.) are not supported.

Connecting I/Os

Connecting cables

For the input/output wiring (X120, X121, X130, X131), use rigid or flexible cables with a conductor cross-section as stated in the manual.

See *SIMOTION D410-2* Manual, Section "Digital inputs/outputs / temperature sensor / analog input."

Note

To achieve optimum interference immunity, shielded cables must be used for connecting analog signals, measuring inputs or external zero marks.

Tools required

Screwdriver 0.4 x 2.0 mm

Wiring inputs/outputs

1. Strip 10 mm off the cable ends and press on a ferrule if required.
2. Wire:
   - The digital inputs for connection of sensors.
   - The digital outputs for the connection of actuators.
   - The analog input
3. Insert the cable into the corresponding spring-loaded terminals of the interfaces. Pressure can be applied to the spring with the tool to facilitate insertion.

Switching the analog input

For the use of the analog input (X131 connector) as analog voltage or current input, DIP switch S5.0 must be switched accordingly:

Table 10-88 S5.0 switch positions

Position	Function
U (left)	The analog input is used as voltage input.
I (right)	The analog input is used as current input.

Additional references

The connection examples for the wiring of the inputs/outputs can be found in the *SIMOTION D410-2* Manual, Section "Digital inputs/outputs / temperature sensor / analog input".

Creating a shield connection

Using shielded cables

The following options are available for the shield connection when using shielded cables:

- A shield connection using a shielding bus supplied separately
- Shield connection via the M3 screw-on shield connecting element on the housing of the SIMOTION D410-2

Using a shielding bus

If a shielding bus is used, proceed as follows:

1. Attach the cable shield to a grounded shielding bus after the cable entry point in the cabinet. Strip the insulation off the cable first.
2. Continue routing the shielded cable as far as the module, but do not make a connection to the shield there.

Using a shield connection on the SIMOTION D410-2

1. Unscrew the holding clamp of the M3 shield connection (Torx screwdriver T10) at the top of the SIMOTION D410-2 until there is a space below the clamp.
2. Insert the cable. The cable shield must first be exposed.
3. Tighten the fixing bracket so that the cable shield and cable are pressed against the shield connection through the holding clamp (tightening torque 0.8 Nm or 7.1 lbf in).

The following figure shows how to connect the cable shield.

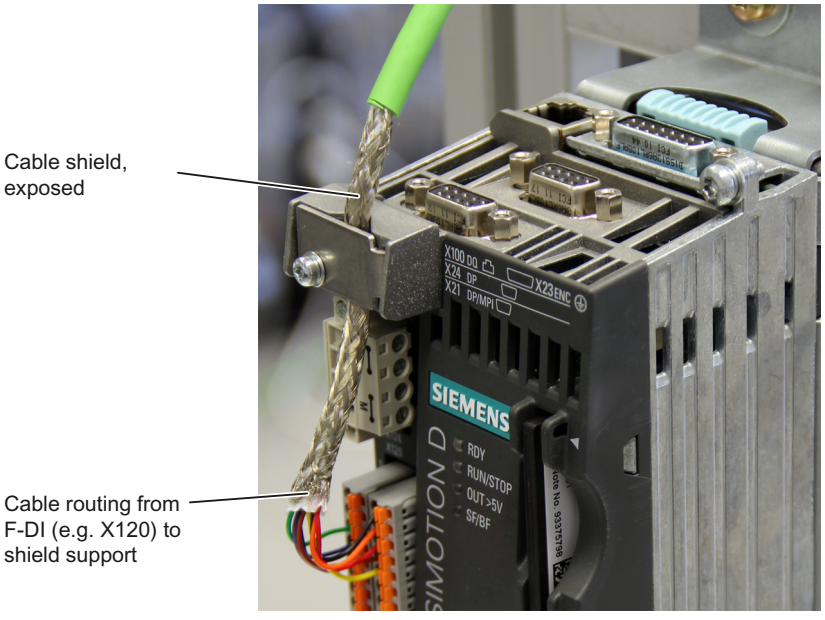


Figure 10-151 Shield connection using the SIMOTION D410-2 DP as an example

Connecting PROFIBUS/MPI

PROFIBUS connection components

Connection components

Individual nodes are connected by means of bus connectors and PROFIBUS cable. Remember to provide a bus connector with a programming port at the ends of the subnet. This will give you the option of expanding the subnet if required (for example, for a PG or SIMATIC HMI device).

Use RS 485 repeaters for the connection between segments and to extend the cable.

Segments

A segment is a bus cable between two terminating resistors. A segment can contain up to 32 nodes. In addition, a segment is limited by the permissible cable length, which varies according to the transmission rate.

Terminating resistor

A cable must be terminated with its own surge impedance to prevent line disturbances caused by reflections. Activate the terminating resistor at the first and last node of a subnet or segment.

Make sure that the nodes to which the terminating resistor is connected are always supplied with voltage during power-up and operation.

PROFIBUS cables and connectors

Properties of PROFIBUS cables

The PROFIBUS cable is a two-core, twisted and shielded cable with defined properties:

Table 10-89 Properties of PROFIBUS cables

Characteristics	Values
Characteristic impedance	Approximately 135 to 160 Ω (f = 3 to 20 MHz)
Loop resistance	≤ 115 Ω/km
Effective capacitance	30 nF/km
Damping	0.9 dB/100 m (f = 200 kHz)
Permissible conductor cross-section	0.3 mm^2 to 0.5 mm^2
Permissible cable diameter	8 mm + 0.5 mm

Connector features

The bus connector is used to connect the PROFIBUS cable to the PROFIBUS DP interfaces. thus establishing a connection to additional nodes.

Table 10-90 SIMOTION D410-2 PROFIBUS interfaces

	D410-2 DP	D410-2 DP/PN
PROFIBUS DP/MPI interface	X21	X21
PROFIBUS DP interface	X24	–

You will find an overview of the bus connectors available for order in the *SIMOTION D410-2* Manual, Section "Spare parts and accessories".

Length of PROFIBUS cables

Cable lengths and baud rate

The baud rate determines the cable length of a subnet segment.

Table 10-91 Permitted cable length of a subnet segment for specific baud rates

Baud rate	Max. cable length of a segment (in m)
19.6 to 187.5 kbit/s	1000 [1)]
500 kbit/s	400
1.5 Mbit/s	200
3 to 12 Mbit/s	100

[1)] With isolated interface

Longer cable lengths

If you must implement longer cable lengths than permitted in one segment, you must use RS 485 repeaters. The maximum possible cable lengths between two RS 485 repeaters correspond to the cable length of a segment. You can connect up to nine RS 485 repeaters in series.

Note that an RS 485 repeater must be counted as a subnet node when determining the total number of nodes to be connected, even if it does not have its own PROFIBUS address.

Rules for the laying of PROFIBUS cables

Laying of bus cable

During laying of the PROFIBUS cable, you must:

- Not twist the cable
- Not stretch the cable
- Not squeeze the cable

Supplementary conditions

In addition, when laying a bus cable for indoor use, you must take into account the following supplementary conditions (dA = external cable diameter):

Table 10-92 Supplementary conditions for the laying of PROFIBUS cables

Characteristics	Supplementary conditions
Bending radius for a single bend	80 mm (10xdA)
Bending radius for multiple bends	160 mm (20xdA)
Permissible temperature range for cable routing	-5° C to +50° C
Temperature range for storage and stationary operation	-30° C to +65° C

Additional references

Length codes for prefabricated cables can be found in:

- *SIMOTION PM 21, SIMOTION Motion Control* Catalog
- *IK PI Industrial Communication* Catalog

Connecting PROFIBUS DP (interface X21 and X24)

PROFIBUS cables are connected to the corresponding interface via a bus connector.

Table 10-93 SIMOTION D410-2 PROFIBUS interfaces

	D410-2 DP	D410-2 DP/PN
PROFIBUS DP/MPI interface	X21	X21
PROFIBUS DP interface	X24	–

Connecting the bus connector

Proceed as follows to connect the bus connector:

1. Plug the bus connector into the corresponding interface of the control unit.
2. Screw the bus connector into place.
   If the control unit is located at the start or end of a segment, you must switch on the terminating resistor ("ON" switch setting).

Figure 10-152 Terminating resistor "switched on" or "switched off"

Note

Make sure that the nodes at which the terminating resistor is located are always supplied with voltage during startup and operation.

Removing the bus connector

You can remove the bus connector with a looped-through bus cable from the PROFIBUS DP interface at any time without interrupting data traffic on the bus.

NOTICE
Data communication disturbed because bus terminator missing
A bus segment must be terminated at both ends with a terminating resistor. This is not the case if the last bus connector node is de-energized, for example. Because the bus connector takes its voltage from the station, this terminating resistor is ineffective.
Make sure that the stations at which the terminating resistor is connected are always energized.

Connection rules in the PROFIBUS subnet

Introduction

There are a number of rules for configuring and installing cables for PROFIBUS networks to ensure seamless communication over PROFIBUS. These rules apply to both configuring and cabling as well as address assignment for the different network nodes.

Connection rules

- **Before** you interconnect individual nodes in a subnet, you must assign a unique PROFIBUS address to each node.
- Narrow down the number of nodes by limiting the PROFIBUS addresses to the highest address in the network.
  Tip: Mark the address on the housing of all nodes in a subnet. Then you can always see which address is assigned to which node in your system.
- Connect all nodes in a subnet "in series". No spur lines may be routed to the PROFIBUS DP. In addition, integrate the PGs and SIMATIC HMI devices for commissioning or servicing in the subnet in series.
- If you operate more than 32 nodes on a subnet, you must use RS 485 repeaters to connect the bus segments. More detailed information can be found in the description of the RS 485 repeater, see the *S7-300 Automation Systems, Module Data* Manual.
  In a PROFIBUS subnet, all segments combined must have at least one DP master and one DP slave.
- Use RS 485 repeaters to connect ungrounded bus segments and grounded bus segments.
- The maximum number of nodes per bus segment decreases with each RS 485 repeater. This means that when a bus segment contains an RS 485 repeater, the bus segment can contain no more than 31 additional nodes. However, the number of RS 485 repeaters does not affect the maximum number of nodes on the bus.
- Up to 10 segments can be connected in a row (max. 9 repeaters).
- **At least** one terminator must be supplied with **5 V**.
  To accomplish this, the PROFIBUS DP connector with an activated terminating resistor must be connected to a device that is switched on.
- Before inserting a new node on the subnet, you must switch off its supply voltage.
  The station must be inserted **first** and then switched on.
  When a station is disconnected, the connection must **first** be deactivated and then the connector withdrawn.
- The bus line of a segment must be terminated at **both ends**. This is achieved by switching on the terminating resistor in the PROFIBUS DP connector at the first and last node and switching off the other terminating resistors.

Example

This illustration below shows an example configuration of a subnet with SIMOTION D410-2 DP.

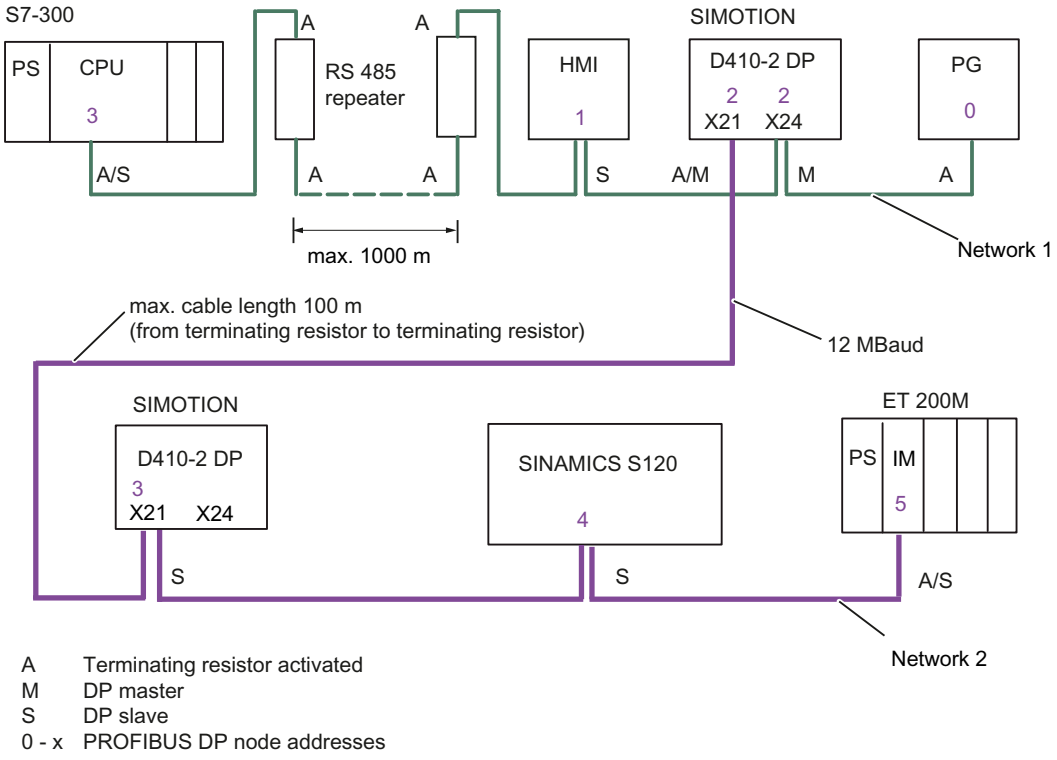


Figure 10-153 Networking example of SIMOTION D410-2 DP

Operating interface X21 as an MPI

Applications

The X21 interface can also be operated as an MPI interface instead of a PROFIBUS DP interface. The typical (default) baud rate is 187.5 Kbaud. A baud rate of up to 12 MBaud can be set for communication with other CPUs. It should be noted, however, that a rate of 12 MBaud is not supported by all CPUs (e.g. smaller SIMATIC S7 CPUs).

The following list provides examples of when using MPI (multi-point interface) may prove effective:

- If a PC/PG is being used with an MPI interface
- If an OP/TP only has an MPI interface
  (newer devices have PROFIBUS or PROFINET interfaces)
- If SIMOTION and SIMATIC CPUs are coupled via XSEND/XRECEIVE

When communicating with XSEND/XRECEIVE, there is no need to configure the connection in **NetPro**. XSEND/XRECEIVE can be used via PROFIBUS or MPI.

- Via PROFIBUS: For communication between SIMOTION devices
- Via MPI: For communication between SIMOTION and SIMATIC S7 devices
  The SIMOTION interface must be connected to the MPI interface of the SIMATIC S7 devices. Connection via PROFIBUS is not possible.
  The baud rate of the SIMATIC S7 device must be set at the SIMOTION interface (see documentation for the relevant SIMATIC S7 devices).

Operate MPI like PROFIBUS

The information on wiring the connector (terminating resistors) and the rules for routing of cables for PROFIBUS apply to this interface as well. When carrying out this procedure, consult the relevant references.

Connector features

The bus connector is used to connect the MPI bus cable to the MPI interface (X21). This enables you to establish connections to additional nodes (e.g. PG or SIMATIC S7-CPU).

You will find an overview of the bus connectors available for order in the *SIMOTION D410-2* Manual, Chapter "Spare parts and accessories".

MPI bus cable

The PROFIBUS cable specifications apply here as well;

Please note the relevant information on setting up an MPI network.

Setting up an MPI network

Keep in mind the following basic rules when setting up an MPI network:

- When using interface X21 as an MPI interface, it is not possible to arrange additional control for a drive in isochronous mode or to connect distributed I/Os to this interface.
- An MPI bus line must be terminated at both ends. This is achieved by activating the terminating resistor in the MPI connector in the first and last station and deactivating the other terminating resistors.
- At least one terminator must be supplied with 5 V.
  This means that an MPI connector with an activated terminating resistor must be connected to a device that is switched on.
- Spur lines (cables leading from the bus segment to the station) should be as short as possible, that is, < 5 m in length. Unused spur lines should be removed wherever possible.

- Every MPI station must be connected to the bus first and then activated.
  To disconnect the station, it must first be deactivated. Then, the station can be removed from the bus.
- Maximum cable lengths:
  - 200 m per bus segment
  - 2,000 m total length with RS 485 repeaters

Note

You can also use intelligent DP slave functionality for PROFIBUS communication between CPUs.

Connecting PROFINET IO components (D410-2 DP/PN only)

Wiring PROFINET

Procedure

As standard, a SIMOTION D410-2 DP/PN has a PROFINET IO interface with two ports.

Suitable PROFINET cables and connectors must be used for the PROFINET connection. The autocrossing functionality of the PROFINET interface means crossed as well as uncrossed cables can be used.

Mixed operation of IRT and RT

For mixed operation of IRT and RT, note that the IRT-capable devices must form a so-called IRT domain, i.e. there must not be any non-IRT devices on the data transmission link between the IRT devices.

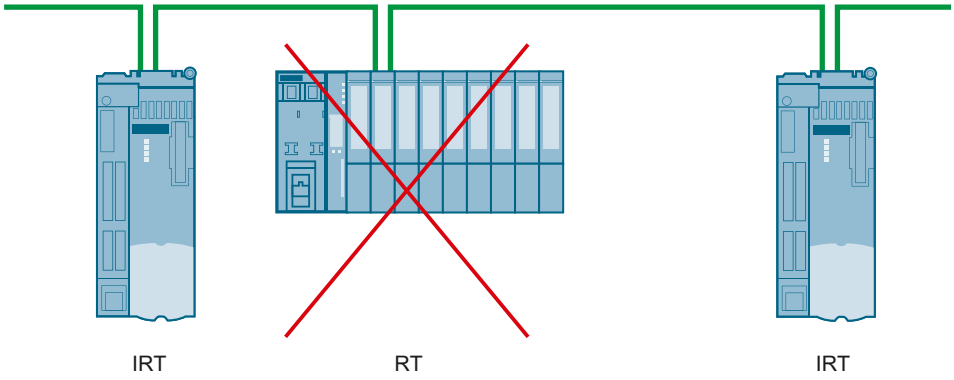


Figure 10-154 Mixed operation of IRT and RT

PROFINET cables and connectors

Cable and connector types

Note

For connecting PROFINET IO to D410-2, a connector with a 180° cable outlet is recommended (IE FC RJ45 plug 180).

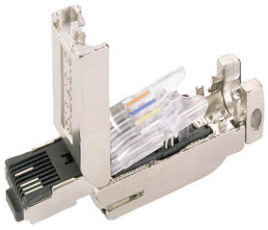


Figure 10-155 RJ45 PN connector with a 180° cable outlet

Table 10-94 Connector types for PROFINET

Connectors	Designation	Article number
IE FC RJ45 plug 180	RJ45 FastConnect connector for Industrial Ethernet/PROFINET with 180° cable outlet	
	1 pack = 1 unit	6GK1901-1BB10-2AA0
	1 pack = 10 units	6GK1901-1BB10-2AB0

Table 10-95 Cable types for PROFINET

Cable	Designation	Article number
IE FC Cable GP 2x2 (Type A)	4-wire, shielded TP installation cable for IE FC RJ45	6XV1840-2AH10
IE FC Flexible Cable GP 2x2 (Type B)	4-wire, shielded flexible TP installation cable for IE FC RJ45	6XV1870-2B
IE FC Trailing Cable GP 2x2 (Type C)	4-wire TP installation cable for trailing cable use	6XV1870-2D
IE FC Trailing Cable 2x2 (Type C)	4-wire, shielded TP installation cable for connection to FC OUTLET RJ45, for trailing cable use	6XV1840-3AH10
IE FC Marine Cable 2x2	4-wire, shielded marine-certified TP installation cable for connection to FC OUTLET RJ45	6XV1840-4AH10

Table 10-96 Stripping tool for Industrial Ethernet / PROFINET

Tool	Designation	Article number
IE FC Stripping Tool	Stripping tool for Industrial Ethernet / PROFINET	6GK1901-1GA00

Additional references

For further information about the cables, connectors, and stripping tool, see

- Section "Spare parts and accessories" in the SIMOTION D410-2 Manual
- *IK PI, Industrial Communication* Catalog

Connecting Ethernet

Wiring Ethernet

An Industrial Ethernet can be connected to the 8-pin RJ45 socket X127 P1.

The interface supports a transmission rate of 10/100 Mbit/s. Suitable Ethernet cables and connectors must be used for the Ethernet connection.

A shielded twisted pair cable is used for the networking.

Note

The Ethernet interface supports PROFINET basic services and therefore has the designation PN/IE. These PROFINET basic services (e.g. DCP, LLDP, SNMP) provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives, I/O modules, etc.

Recommended connecting cables

The following cables are available:

- SIMATIC NET, Industrial Ethernet TP XP CORD RJ45/RJ45
  - TP cable prefabricated with 2xRJ45 connectors
  - Crossed send and receive cable
  - Article number: 6XV1870-3R□□□ (□□□ - length code)
- SIMATIC NET, Industrial Ethernet TP CORD RJ45/RJ45
  - TP cable prefabricated with 2xRJ45 connectors
  - Uncrossed send and receive cable
  - Article No.: 6XV1870-3Q□□□ (□□□ - length code)

The autocrossing functionality of the Ethernet interface means crossed as well as uncrossed cables can be used.

Autocrossing

Per default, the Ethernet interface is set to "Automatic setting" in **HW Config**, so that autocrossing is available. The settings can be found in the port properties (X127 P1) in the module rack of **HW Config**.

Autocrossing is deactivated with "Manual setting". As the Ethernet interface of the D410-2 is wired as Ethernet terminal, in this case you must use crossed cables for the networking with a PG/PC or another D410-2.

If the communication peer has autocrossing (e.g. PC, switch or hub), crossed and uncrossed cables can be used.

Additional references

For further information about the cables and connectors, see the *Industrial Communication IK PI* Catalog.

Routing

Routing describes the cross-network transfer of information from Network x to Network y.

Routing on SIMOTION D

Routing between the different interfaces

SIMOTION D410-2 supports S7 routing between the following interfaces:

- Between Ethernet interface X127 and PROFIBUS interfaces X21 and X24 (X24 only available on D410-2 DP)
- Between Ethernet interface X127 and PROFINET interface X150 (only available on D410-2 DP/PN)
- Between PROFINET interface X150 and PROFIBUS interface X21 (only available on D410-2 DP/PN)

Each combination of Ethernet interface X127 and PROFINET interface X150 represents a separate IP subnet.

All ports of the X150 PROFINET interface belong to the same IP subnet.

IP routing from IP subnet to IP subnet is not supported. You can use an external IP router for this.

Below is a list of the options available for connecting a PG/PC or HMI device to a SIMOTION D using S7 routing.

Engineering System/HMI on PROFIBUS (example: D410-2 DP)

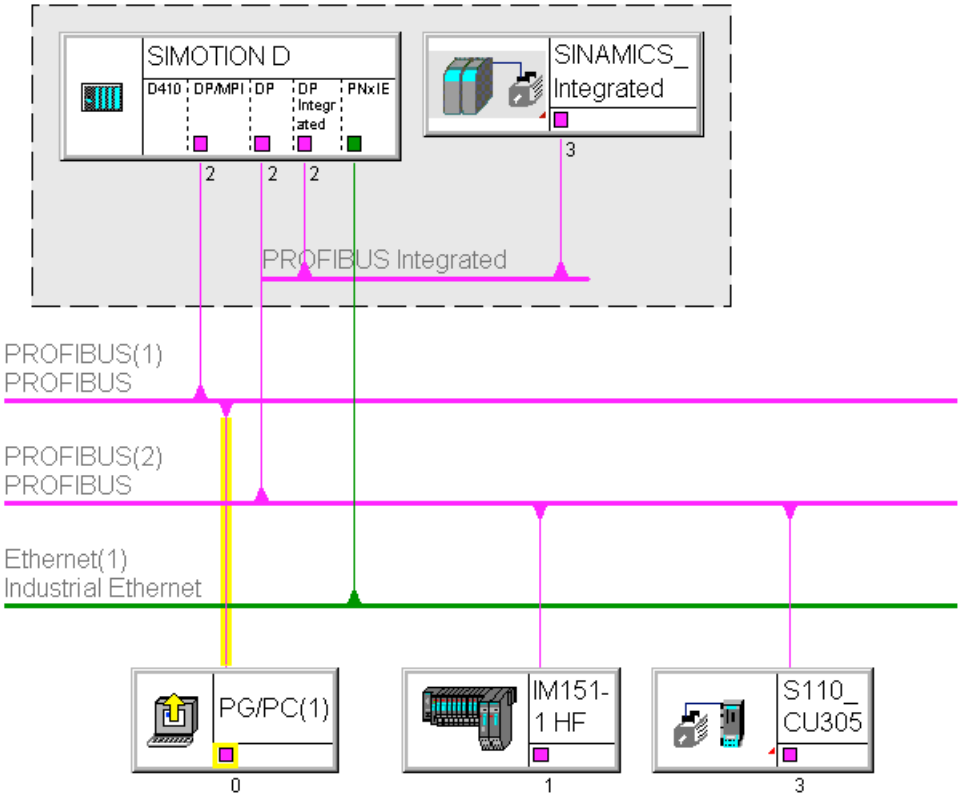


Figure 10-156 Example of PG/PC to PROFIBUS interface (X21)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to the Ethernet interface (X127 P1)

Engineering System/HMI on PROFINET (example: D410-2 DP/PN)

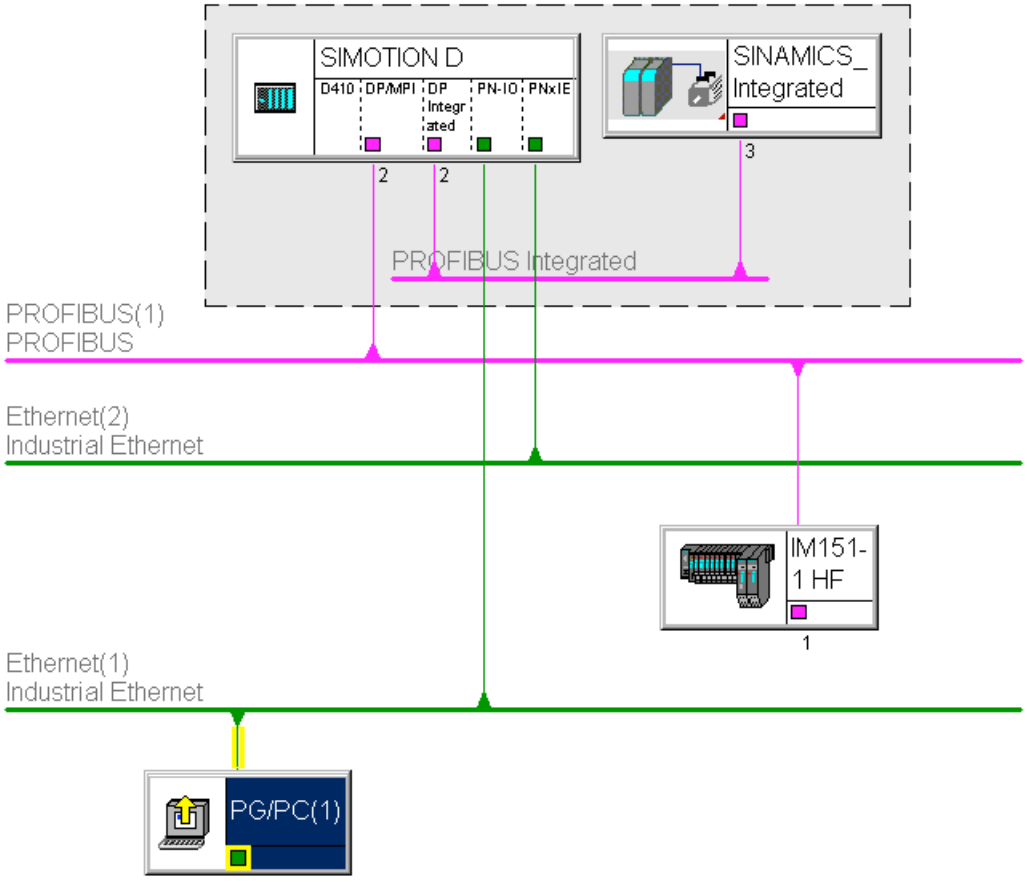


Figure 10-157 Example of PG/PC to PROFINET interface (PNxIO, X150)

- S7 routing to the (master) PROFIBUS interface (only if configured)
- S7 routing to the PROFIBUS Integrated
- S7 routing to Ethernet interface PN/IE (X127 P1)
- Access to the components on the same subnet via the switch functionality of the PROFINET IO interface

Engineering System/HMI on Ethernet (example: D410-2 DP)

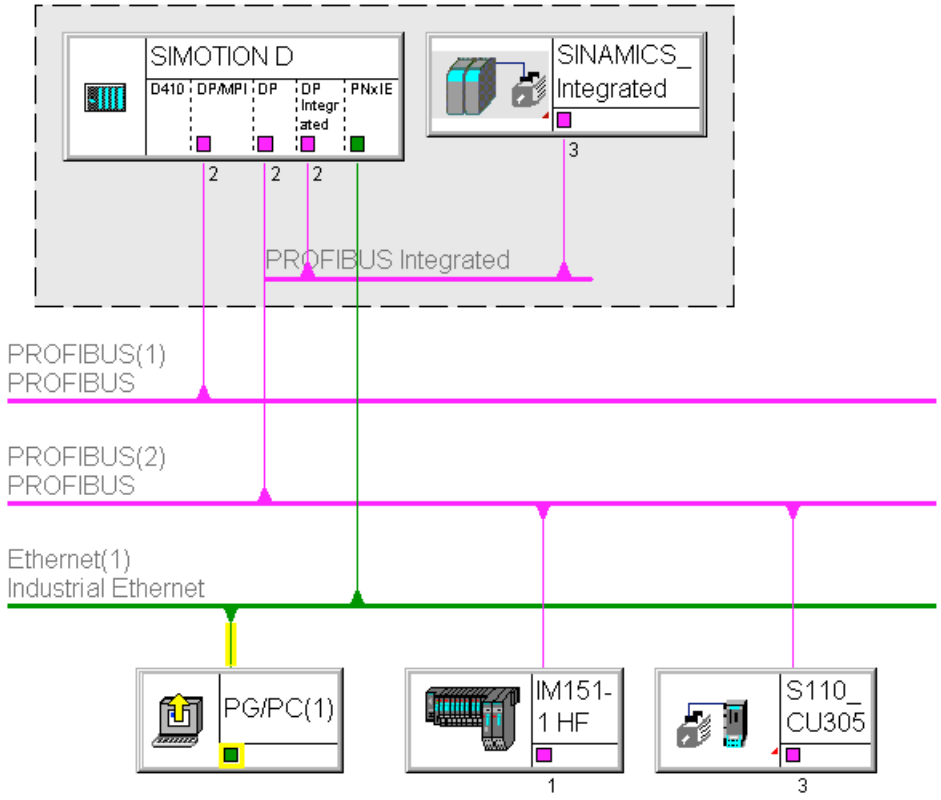


Figure 10-158 Example of PG/PC to Ethernet interface (X127 P1)

- S7 routing to the other (master) PROFIBUS interfaces (only if configured)
- S7 routing to the PROFIBUS Integrated

Routing on SIMOTION D (SINAMICS integrated)

S7 routing to the internal PROFIBUS on SINAMICS Integrated

All SIMOTION D have an integrated SINAMICS drive control. In order to be able to access drive parameters, the message frames must be routed from the external SIMOTION D interfaces to the internal PROFIBUS DP. S7 routing can be used to access the integrated PROFIBUS. Here, the internal PROFIBUS DP forms a separate subnet. This must be especially taken into account for the communication to several routing nodes.

Additional references

Further information about routing and the differences between IP and S7 routing can be found in the *SIMOTION Communication* System Manual.

Connecting an external encoder

Connecting the connecting cable

Use only shielded cables to connect an external encoder to the encoder interface (X23). The shielding must be connected with the metallic or metallized connector housing.

We recommend that bipolar encoders are used. When using unipolar encoders, the unused negative track signals can either be connected or connected to ground. This results in different switching thresholds.
See *SIMOTION D410-2* Manual, Section "Encoder interface (HTL/TTL/SSI)".

The pre-assembled connecting cables offer optimal noise immunity as well as sufficiently measured cross-sections for the supply voltage of the external encoder.

The connecting cables are available in various lengths, see *NC 60* Catalog.

Procedure for connecting encoders

To connect an external encoder (HTL, TTL or SSI encoder), proceed as follows:

1. Connect the connecting cable to the encoder.
2. Plug the D-Sub connector (15-pin) to the socket X23.
3. Lock the connector using the finger screws.

NOTICE
Destruction of the encoder electronics Operation of a 5 V encoder on the 24 V encoder supply may result in the destruction of its electronic components! Make sure that you can operate the connected encoder on a 24 V power supply (e.g. HTL encoder). This setting can be set in the expert list of the drive in parameter p0400 and in the following parameters.

10.1.2.5 Commissioning (hardware)

Overview

Requirements

The following requirements must be satisfied for the first commissioning of the SIMOTION D410-2:

- Your system with SIMOTION D410-2 has been installed and wired.
- Your PG/PC is connected to the SIMOTION D410-2 via the PROFIBUS DP, Ethernet, or PROFINET (D410-2 DP/PN only) interface.

Commissioning steps

Commissioning the hardware involves the following steps:

1. Inserting the CompactFlash card (Page 7347)
2. Checking the system (Page 7348)
3. Switching on the power supply (Page 7349).

Additional references

For information on installing/mounting and commissioning the SINAMICS S120 components, refer to the *SINAMICS S120* Commissioning Manual.

Inserting the CompactFlash card

Characteristics of the CF card

The CF card is mandatory for operation of the SIMOTION D410-2. The SIMOTION Kernel (SIMOTION D firmware) and the software used to control the drives (SINAMICS firmware) are on the CF card.

To load the SIMOTION Kernel, the CF card must be inserted when the SIMOTION D410-2 is powered up.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge The CompactFlash card is an ESD-sensitive component. De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF. Comply with the ESD rules.

Note

The ramp-up time of the SIMOTION D410-2 as well as the time behavior during CF card accesses depends on the type of memory card used. Take this into account in your application, as different memory card types are used for availability reasons.

Procedure

To insert the CF card, perform the following steps:

1. The direction of insertion of the CF card is indicated by an arrow located on both the plug-in slot and the CF card. Align the CF card with the arrows.
2. Gently insert the CF card into the empty plug-in slot of the SIMOTION D410-2 until it clicks into place.
   If correctly inserted, the CF card is flush with the housing.

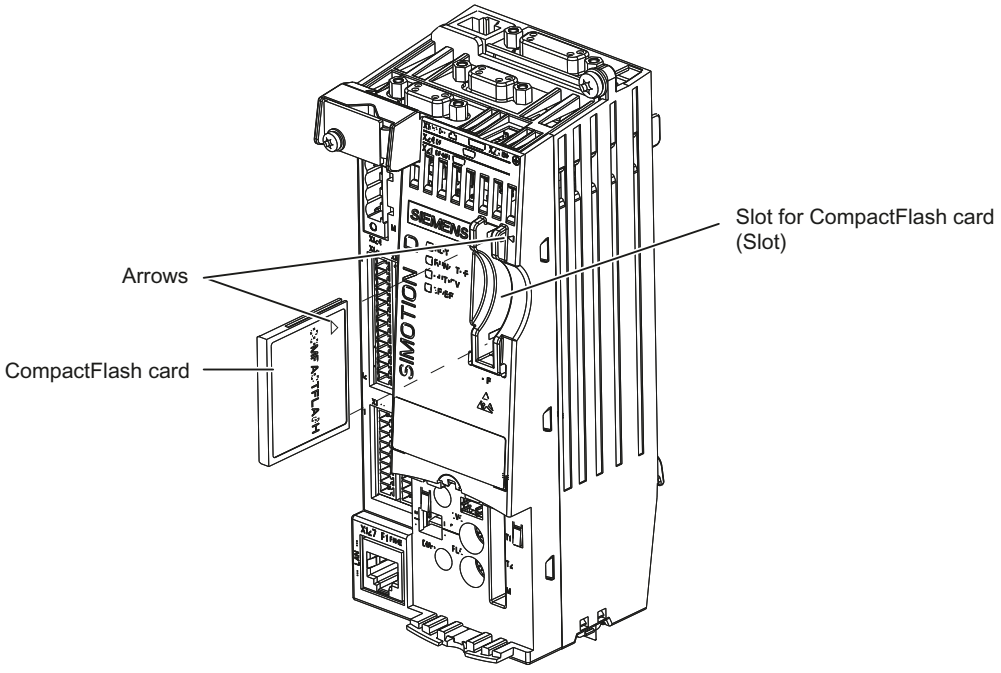


Figure 10-159 Inserting the CF card

Checking the system

Procedure

Check the final installed and wired system one more time before it is switched on, Observe the safety-relevant items of the following checklist.

Checklist	✓
Have you observed all ESD measures when handling the components?	
Have all screws been tightened to their specified torque?	
Have all connectors been properly inserted and locked/screwed?	
Are all components grounded and have all shields been attached?	
Have you taken the load capacity of the central power supply into consideration?	

Switching on the power supply

Switching on the external power supply

Power is supplied to the SIMOTION D410-2 via an external power supply unit, e.g. via a SITOP power supply. In exceptional cases, the SIMOTION D410-2 can also be supplied via the Power Module in blocksize format, see Section "Power supply" in the SIMOTION D410-2 Manual.

Switch on this power supply.

NOTICE
SIMOTION D410-2 shuts down if the power supply is interrupted It is essential to ensure that the external 24 VDC power supply to the SIMOTION D410-2 is not interrupted for longer than 3 ms. After a longer interruption, the SIMOTION D410-2 shuts down and can only be restarted with OFF/ON. You will find further information in Section Properties of the user memory (Page 7352).

Power-up of Control Unit

Once the power supply has been switched on, the SIMOTION D410-2 begins to power up:

1. At the start of the power-up, all LEDs are briefly illuminated in yellow for a short LED test. The LEDs on the SIMOTION D410-2 enable you to track the progress of the power-up. Any errors are displayed.
2. Startup of the SIMOTION Kernel.

3. All DRIVE-CLiQ connections (with SINAMICS Power Module, for example) are identified automatically.

   Note

   As long as the RDY LED continues to flicker, power-up is not complete and it is not possible to go online.

   During commissioning the firmware of the components is upgraded or downgraded automatically based on the FW version on the CF card and the FW version on the SINAMICS components (DRIVE-CLiQ components, Power Modules, etc.).

   The update can take several minutes and its progress is tracked by corresponding messages appearing in the alarm window of SIMOTION SCOUT.

   SIMOTION D410-2 / DRIVE-CLiQ components:

   A FW update on the SIMOTION D410-2 is indicated by yellow flashing and for DRIVE-CLiQ components by red-green flashing of the RDY LED:

   - FW update running: RDY LED flashes slowly (0.5 Hz)
   - FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

   Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

4. The first time it is switched on, the SIMOTION D410-2 goes into STOP state following power-up.

Following power-up, the SIMOTION D410-2 is in a state in which it can be configured.

Fan

SIMOTION D410-2 has an integrated fan. This fan is always required for operation.

Fan faults as well as overtemperature of the module are signaled by a system variable, PeripheralFaultTask and a diagnostic buffer entry .

Performing a reset

The RESET button is located behind the blanking cover on the SIMOTION D410-2.

The entire system is reset when the RESET button is pressed and a new power-up of the system forced. This operation is comparable with a "Power on Reset" but does not require the disconnection of the 24 V power supply.

User memory concept

SIMOTION D410-2 memory model

The following figure provides an overview of the memory model of SIMOTION D410-2.

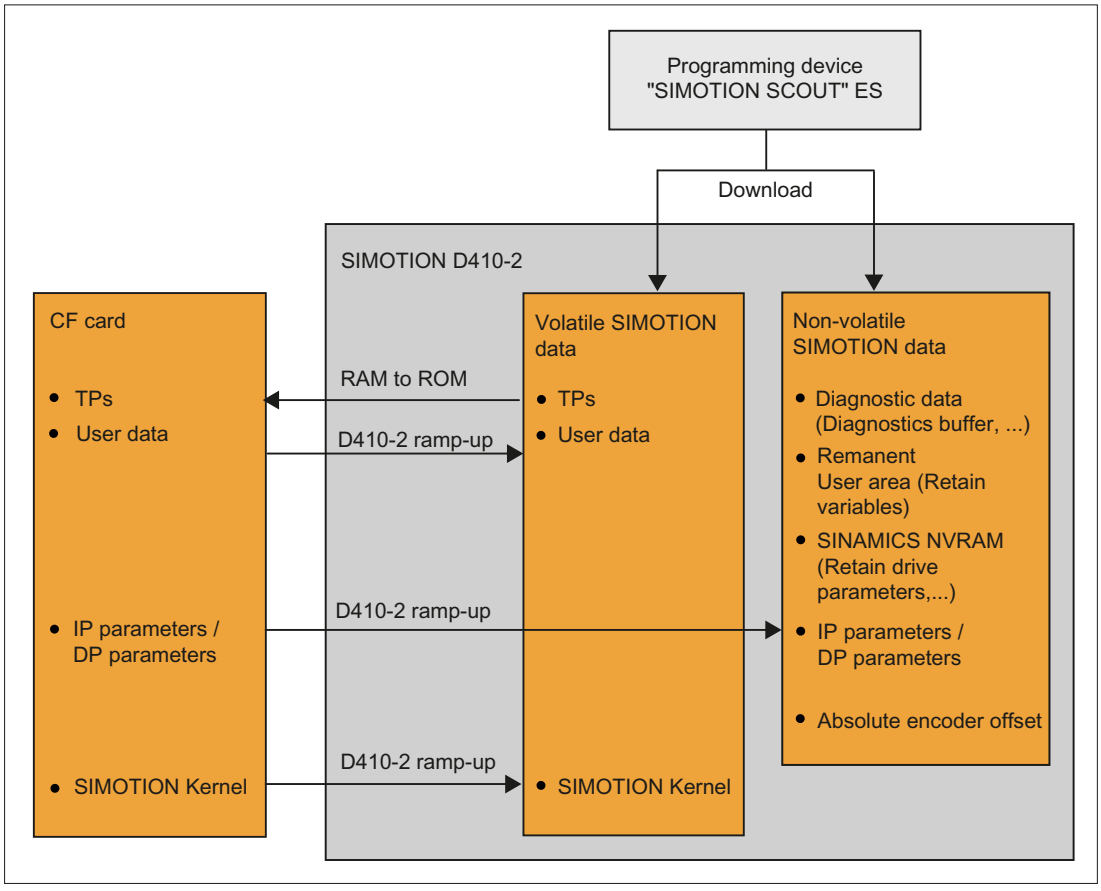


Figure 10-160 SIMOTION D410-2 memory model

The SIMOTION Kernel (SIMOTION D firmware, including SINAMICS Integrated Firmware) contains the functions required for virtually all applications, and essentially acts as a PLC with the command set in accordance with IEC 61131-3 as well as system functions for controlling various components, such as inputs and outputs.

The SIMOTION Kernel can be expanded by loading technology packages (TPs), e.g. for motion control or temperature controllers.

In the following sections, you will find information about the user memories and the steps involved in certain operations.

Properties of the user memory

Non-volatile data

Non-volatile data makes it possible to retain relevant data for the user and the system even when the SIMOTION D410-2 has been switched off. Information about the area which can be used for non-volatile data is available in the *SIMOTION D410-2* Manual, section "Technical Data".

A SIMOTION device has the following non-volatile data:

Table 10-97 Non-volatile data contents

Non-volatile data	Content
Kernel data	• Last operating state • IP parameters (IP address, subnet mask, router address) • DP parameters (PROFIBUS DP address, baud rate) • Diagnostic data (diagnostics buffer,...)
Retain variables	• Variables in the interface or implementation section of a unit declared with VAR_GLOBAL RETAIN • Global device variables with the "RETAIN" attribute
Retain TO	Absolute encoder offset
DCC blocks	SAV blocks and user-defined blocks with retain behavior ("SAV = SAVE", blocks for the non-volatile data backup).
NVRAM (SINAMICS)	For the SINAMICS Integrated and for the SINAMICS S120 CU310-2/CU320-2, data protected against loss at power failure is called NVRAM data or non-volatile data.

Note

DCC SIMOTION blocks with retain behavior act like retain variables in terms of copying RAM to ROM, resetting memory, downloading, backing up non-volatile SIMOTION data (_savePersistentMemoryData) and backing up data.

With DCC SINAMICS blocks, data buffering is only via NVRAM. SINAMICS data is not saved with _savePersistentMemoryData. The non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

For further information on DCC, see the *DCC Programming* Programming Manual.

The non-volatile data of the SIMOTION D410-2 has the following properties:

Table 10-98 Non-volatile data and real-time clock properties

Properties	Non-volatile data	Real-time clock (RTC)
Location	Non-volatile data is located in the NVRAM of the SIMOTION D410-2.	The real-time clock is backed up maintenance-free via a SuperCap.
Back-up battery	No	No
Backup time	There is no maximum backup time	At least 5 days

If the buffer time is exceeded on the real-time clock, the time is reset.

CF card

With the _savePersistentMemoryData system function, the user program can back up the contents of the non-volatile SIMOTION data to the CF card. This ensures that the retain variables and the absolute encoder position are backed up in the event that a spare part is used.

For the SINAMICS Integrated and SINAMICS S120 CU310-2/CU320-2, the non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

Note

IP and DP parameters in the non-volatile data

If the CF card contains a configuration, the IP and DP parameters are loaded from the CF card during ramp-up and used by the SIMOTION device. The SIMOTION D410-2 uses the addresses defined in these parameters to go online. During ramp-up, the IP and DP parameters on the CF card are also written to the non-volatile data. If the SIMOTION device is then powered up with a CF card with no configuration, the IP and DP parameters are retained in the non-volatile data and are used by the device. Thus, the SIMOTION device can continue to go online if a configuration was loaded with SIMOTION SCOUT at least once or if the SIMOTION device is powered up with a CF card containing a configuration.

The CF card contains the following data:

- SIMOTION Kernel (SIMOTION D firmware)
- Technology packages (TP)
- User data (units, configuration data, parameter settings, task configuration)
- IP parameters (IP address, subnet mask, router address)
- DP parameters (PROFIBUS DP address, baud rate)

It may also contain:

- User data saved with _savePersistentMemoryData and _export/_saveUnitDataSet
- Non-volatile SINAMICS data (NVRAM data) of the SINAMICS Integrated backed up with CU parameter p7775 = 1
- Data from SIMOTION IT
- Archived SCOUT project

Volatile SIMOTION data (RAM / current data RAM)

The volatile SIMOTION data is defined by the following properties:

- The volatile SIMOTION data is located in the RAM memory of the SIMOTION device.
- The download data of SIMOTION SCOUT is written to this memory.

- This data is lost when the SIMOTION D410-2 is switched off.
- The "volatile SIMOTION data" area contains the following data:
  - SIMOTION Kernel (D410-2 firmware)
  - Technology packages (TP)
  - User data (programs, configuration data, parameter settings)

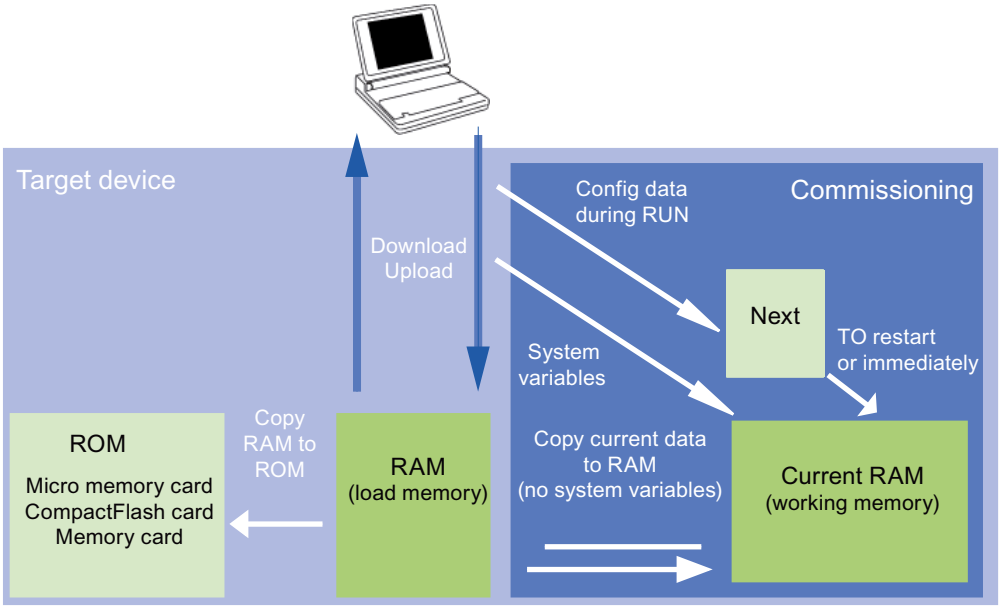


Figure 10-161 Configuration data and system variables in the volatile memory (memory concept, principle)

You can find additional information about memory management in SIMOTION in the *SIMOTION Runtime Basic Functions* Function Manual.

Operations and their effect on the user memory

The operator actions and their effects on the user memory are described below. These operator actions are marked with arrows in the following displays:

- Figure 10-160 SIMOTION D410-2 memory model (Page 7351)
- Figure 10-161 Configuration data and system variables in the volatile memory (memory concept, principle) (Page 7354)

SIMOTION SCOUT download

The following data is transferred with "Load project to target system" or "Load CPU / drive unit to target device" from the engineering system to the "volatile SIMOTION data" area:

- User data (units, configuration data, parameter settings, task configuration)
- Technology packages (TPs)

In addition, the IP and DP parameters are saved to the "non-volatile SIMOTION data" area. The retain variables are set to their initial values, but this depends on the settings in SIMOTION SCOUT. If the SIMOTION D410-2 is switched off following the download, the volatile SIMOTION data is lost.

Copy RAM to ROM

The Copy RAM to ROM menu command saves the following data via the engineering system to the CF card:

- Technology packages and user data (units, configuration data, parameter assignments, task configuration) from the "volatile SIMOTION data" area.
- Current values are copied to the "volatile SIMOTION data" area, depending on the settings in SIMOTION SCOUT.

Note

The "Copy RAM to ROM" menu command does not save the current values of the retain variables to the CF card. Use the system function "_savePersistentMemoryData" for this.

Note

The "Copy RAM to ROM" function is also available for drive units and saves the volatile SINAMICS data to the non-volatile memory (CF card).

Current RAM

If you change the system variable values, these take immediate effect in the current RAM. New configuration data values are initially stored in the Next memory. Configuration data that take immediate effect are automatically transferred to the current RAM. Configuration data that will only become active following a RESTART on the technology object (set the restartactivation system variable to the ACTIVATE_RESTART value) is not written to the current RAM until after the RESTART.

To save the configuration data changed online to the offline project, you must first transfer the content of the current data RAM to the RAM using the menu command "Target system" > "Copy current data to RAM".

Once you have done this, the configuration in SCOUT will no longer be consistent with the configuration in the target device, as a consistency check is performed on the RAM data. Read the data from the RAM using the menu command "Target system" > "Load" > "Load CPU / drive unit to PG" (for the configuration data only) to re-establish a consistent system state.

Use the "Target system" > "Copy RAM to ROM" menu command to save the configuration to the non-volatile memory on the CF card.

Note

The "Copy current data to RAM" command does not transfer the values of the system variables to the RAM memory. This means that "Save to memory card (Copy RAM to ROM)" or "Save in the engineering project (Load CPU / drive unit to PG)" is not possible.

In order to ensure that system variable values can also be saved to the engineering project and memory card, the system variable values must be changed OFFLINE and then downloaded to the target device and saved.

SIMOTION D410-2 power-up

During power-up of the SIMOTION D410-2, the SIMOTION Kernel is loaded from the CF card to the "volatile SIMOTION data" area.

When the SIMOTION D410-2 is switched off, the contents of the "volatile SIMOTION data" area are lost. When the unit is powered up again, the following data is loaded from the CF card:

- Technology packages and user data to the "volatile SIMOTION data" area.
- IP and DP parameters to the "non-volatile SIMOTION data" area.

Backing up non-volatile SIMOTION data

You have the following options for backing up non-volatile SIMOTION data on the CF card:

- In the user program:
  With the "_savePersistentMemoryData" system function, the user program can back up the non-volatile SIMOTION data content to the CF card. This ensures that the retain variables and the absolute encoder position are backed up in the event that a spare part is used. The contents are saved to the "PMEMORY.XML" backup file in the "USER\SIMOTION" directory.
- Via switch/button (service selector switch or DIAG button of the SIMOTION D410-2) or SIMOTION IT web server. See Section Diagnostic data and non-volatile SIMOTION data (Page 7557). The contents are saved to the "PMEMORY.XML" backup file in the "USER \SIMOTION\HMI\SYSLOG\DIAG" directory.

On the system side, this system function ensures that a consistent overall image of the non-volatile SIMOTION data is always available the next time the unit is switched on, even if there is a power failure during backup. An already existing backup file is renamed to "PMEMORY.BAK" before a new backup file is generated. If the save operation to the new backup file fails (e.g.

because the capacity of the CF card is insufficient), this backup copy of the backup file is used the next time an attempt is made to restore the non-volatile SIMOTION data content.

NOTICE
Loss of data due to failure to make backup copies Non-backed-up non-volatile SIMOTION data can be lost on hardware replacement (module defect), for example, if the current value of the retain variables have not been backed up and replaced by their initial values again. Back up the non-volatile SIMOTION data on the CF card.

NOTICE
Repeat referencing necessary after absolute encoder overflow If an absolute encoder overflow occurs after **_savePersistentMemoryData**, the actual position value is no longer correct after the non-volatile SIMOTION data is restored. In this case, homing (absolute encoder adjustment) must be repeated.

With the SCOUT functions "Save variables" and "Restore variables," you also have the option of backing up to your PC and restoring data that was changed during operation and only stored in the runtime system.

Reloading non-volatile SIMOTION data

SIMOTION data backed up on the CF card with **_savePersistentMemoryData** is reloaded in the following scenarios:

1. After a module replacement, see Section Replacing modules in the spare part scenario (Page 7361)
2. After a memory reset, see Section SIMOTION D410-2 memory reset (Page 7493)
3. Via switch position, see Section Delete/restore non-volatile SIMOTION data (Page 7564)

Reloading non-volatile SINAMICS data

For the SINAMICS Integrated and SINAMICS S120 CU310-2/CU320-2, as of SINAMICS FW version V4.5, the non-volatile SINAMICS data (NVRAM data) is backed up by setting the CU parameter p7775 to 1.

Restoring

- Is performed automatically in the event of a module replacement
  A module replacement is detected on the basis of the serial number.
- Can be performed manually
  Restoring can be initiated manually by setting the CU parameter p7775 to 2.

For further information, see Section Backing up / restoring / deleting SINAMICS NVRAM data (Page 7437).

Power failure

If a power failure occurs, the real-time clock is buffered by means of an internal SuperCap.

The non-volatile data is backed up on the D410-2 permanently and maintenance-free in an NVRAM. In this way, the Control Unit is immediately ready for operation without data loss after a power failure.

Power-up and non-volatile SIMOTION data

The table below lists the cases that can arise during power-up in conjunction with the non-volatile data and explains how they are handled.

Table 10-99 Power-up scenarios for non-volatile SIMOTION data

Case	Initial condition	Result
1	The non-volatile SIMOTION data is valid.	SIMOTION D410-2 powers up with the non-volatile SIMOTION data, i.e. with the PROFIBUS address in the non-volatile data.
2	The non-volatile SIMOTION data is invalid and there is no backup file (PMEMORY.XML) and no backup copy of the backup file (PMEMORY.BAK).	SIMOTION D410-2 copies the default settings to the non-volatile SIMOTION data and powers up with this data. In this case, for example, the default PROFIBUS address is used.
3	The non-volatile SIMOTION data is invalid and backup file (PMEMORY.XML) exists and is valid.	SIMOTION D410-2 copies the backup file contents to the non-volatile SIMOTION data and powers up with this data.
4	The non-volatile SIMOTION data is invalid, the backup file is invalid and there is no backup copy of the backup file (PMEMORY.BAK).	SIMOTION D410-2 copies the default settings to the non-volatile SIMOTION data and powers up with this data, in which case, for example, the default PROFIBUS address is used.
5	The non-volatile SIMOTION data is invalid; a backup file exists, but it is invalid; a backup copy of the backup file exists and is valid.	SIMOTION D410-2 copies the backup file contents to the non-volatile SIMOTION data and powers up with this data.

Non-volatile SIMOTION data diagnostics

The user can determine the state of the non-volatile SIMOTION data using the diagnostic buffer, system variables, and PeripheralFaultTask.

Evaluating via the diagnostic buffer

When they are issued, the following messages are entered once in the diagnostic buffer:

Table 10-100 Messages of the diagnostic buffer

Entry	Meaning	Remedy
Non-volatile data loaded from a file (Persistent Data File Loading done)	Non-volatile SIMOTION data has been successfully restored from the backup file on the CF card.	-
Non-volatile data loaded from the backup file (Persistent Data Backup File Loading done)	Non-volatile SIMOTION data has been successfully restored from the backup copy of the backup file on the CF card.	-
Error while loading non-volatile data from a file (Persistent Data File Loading Failure)	Backup file or backup copy could not be loaded. Possible causes: • Backup file or backup copy not available • Invalid data in backup file	Use the "_savePersistentMemoryData" system function to generate a valid backup file.
Module replacement detected - NVRAM has been initialized	A module replacement was detected on the basis of the serial number. The non-volatile SIMOTION data on the controller is deleted and the data from the CF card transferred to the controller.	
Module replacement not detected - NVRAM has not been initialized	An error has occurred. The non-volatile SIMOTION data on the controller has not been deleted.	Possible causes: • Incorrect controller type • File system of the CF card defective

Refer to the *SIMOTION SCOUT* Configuration Manual for how to read out the contents of the diagnostic buffer.

Evaluating via system variables

The system variables in the **device.persistentDataPowerMonitoring** structure indicate the state of the non-volatile SIMOTION data.

Table 10-101 State of the non-volatile SIMOTION data

System variable	Designation	State	Updating
rtcFailure	Indicates that the clock contents (RTC) are invalid (clock must be set again)	NO (91) YES (173)	The status is updated once during power-up; the status must be reset to "NO" via the application; the status is retained even after power off/on.
retainDataFailure	Indicates a checksum error of the non-volatile SIMOTION data; can be an indication of defective hardware	NO (91) YES (173)	The status is updated once during power-up; the status must be reset to "NO" via the application; the status is retained even after power off/on.
persistentDataState	Reading the persistent data	See the table below	The status is updated during power-up.

A data loss of the real-time clock is signaled via the system variable **device.persistentDataPowerMonitoring.rtcFailure = YES**.

The system variable **device.persistentDataPowerMonitoring.persistentDataState** shows the state of the non-volatile SIMOTION data after power-up.

Table 10-102 State of the non-volatile SIMOTION data after power-up (persistentDataState system variable)

State	Meaning
FROM_RAM (1)	Non-volatile SIMOTION data in the SIMOTION device is used
FROM_FILE (2)	Non-volatile SIMOTION data is restored from the backup file
FROM_BACKUP (3)	Non-volatile SIMOTION data is restored from the backup copy of the backup file
INVALID (4)	Data in the non-volatile SIMOTION data and in the backup file / backup copy of backup file is invalid or non-existent/deleted. The SIMOTION device has copied the default settings to the non-volatile SIMOTION data and used this data to power up.

Requirement/availability of the battery

System variables can be used to evaluate:

- Whether a battery is required for the operation of the device (or not)
- Whether a battery is available (or not)

SIMOTION D410-2 has no battery.

Table 10-103 System variable batterynecessary/batteryexisting

System variable on the device	States	Description
fanbattery of data type StructDeviceFanBattery (the system variables are of the data type EnumFanBattery)		
.batterynecessary	MANDATORY	Battery is required for the backup of the non-volatile data and the real-time clock (RTC) of the device. **.batteryexisting** can be used to query whether a battery is installed.
	OPTIONAL	The non-volatile data and the real-time clock (RTC) are backed up via SuperCap. A battery can be used as an option to extend the backup time. **.batteryexisting** can be used to query whether a battery is installed. **Example: D4x5**
	OPTIONAL_RTC [1)]	A battery is not required for backing up the non-volatile data. Only the real-time clock (RTC) is backed up via SuperCap. A battery can be used as an option to extend the backup time of the real-time clock. **.batteryexisting** can be used to query whether a battery is installed. **Example: D4x5-2**
	NOT_MANDATORY [1)]	A battery is not required for backing up the non-volatile data. The real-time clock (RTC) is backed up via SuperCap. **Example: D410-2**

System variable on the device	States	Description
.batteryexisting	EXISTING	EXISTING is only displayed when **.batterynecessary** is set to: • MANDATORY or • OPTIONAL or • OPTIONAL_RTC and a battery is installed.
	NOT_EXISTING	Battery is not available. **State is statically set for D410-2.**

[1] If the SuperCap is discharged, the contents of the real-time clock (RTC) are lost.

Replacing modules in the spare part scenario

SIMOTION module replacement

During a module replacement, a CF card that contains the non-volatile SIMOTION data backed-up with _savePersistentMemoryData, is inserted in a new device of the same type.

A module replacement is detected by the SIMOTION D410-2 on the basis of the serial number. The data backed up on the CF card with _savePersistentMemoryData are then automatically transferred to the new device.

Note

As an additional option, you can back up the non-volatile data by switching the service selector switch, with the DIAG button, or via SIMOTION IT web server. For details, see Section Diagnostic data and non-volatile SIMOTION data (Page 7557).

Initial power-up with the CF card

This requires a CF card on which no device serial number is stored.

If the SIMOTION D410-2 powers up successfully with a new CF card, the serial number of the device is stored on the CF card. In this case, a module replacement cannot be detected.

Note

The serial number stored on the CF card remains unaffected by the following actions:

- Copy RAM to ROM
- Project download
- Writing the CF card via the SCOUT function "Load to file system"
- FW/project update via device update tool

If the CF card contents are copied to another CF card, the serial number is also copied. A serial number stored on the CF card can only by removed by deleting the CF card contents.

SIMOTION D410-2 power-up with the CF card (no module replacement)

Prerequisite is that the same CF card and the same device are used.

If the SIMOTION device powers up successfully, the serial number stored on the CF card is compared with the serial number of the device.

If the serial numbers are identical, there has been no module replacement.

The device powers up. In the non-volatile data present in the device is valid, this will be used (for details, see Table 10-99 Power-up scenarios for non-volatile SIMOTION data (Page 7358)).

SIMOTION D410-2 power-up with the CF card (module replacement)

Prerequisite is that the same CF card and a different device (e.g. replacement because of a defect) are used.

If the device powers up successfully, the serial number stored on the CF card is compared with the serial number of the device.

If the serial numbers are not identical, there has been a module replacement.

This means:

- The serial number of the new module is stored on the CF card.
- The non-volatile SIMOTION data are deleted in the device.
- A diagnostic buffer entry is issued which signals that a module has been replaced.
- The non-volatile SIMOTION data stored on the CF card is transferred to the device (for details, see Table 10-99 Power-up scenarios for non-volatile SIMOTION data (Page 7358)).

NOTICE
Irrevocable data deletion due to incorrect CF card with stored device serial number A module replacement is detected only on the basis of the changed serial number. Inserting the wrong CF card with saved device serial number has the following consequences: • The non-volatile data on the device is permanently deleted. • The IP/DP address set in the device is deleted. You can no longer go online via the IP/DP address set originally. Make sure that the correct CF card is inserted into the SIMOTION D410-2.

Error scenarios

In the event of an error, a diagnostic buffer entry signals that it was not possible to determine whether a module has been replaced. This may have the following reasons:

- The serial number of the device cannot be determined.
- The serial number saved on the CF card cannot be determined (e.g. due to a corrupt file system).
- The controller has not powered up.
- The new serial number could not be transferred to the CF card (e.g. due to a corrupt file system).

Restart after reloading

The system variable **device.startupData.operationMode** is used to define whether the SIMOTION D410-2 Control Unit goes into the RUN state or the last operating state after a power-on/restart.

Possible values of **device.startupData.operationMode**:

- LAST_OPERATION_MODE [0] (default setting)
  The module remains in the STOP operating state after reloading the non-volatile SIMOTION data and must be switched manually to the RUN state with SCOUT, the web server, or the mode switch .
- RUN [1]
  The module goes automatically into the RUN state after reloading.

SINAMICS module replacement

A module replacement is detected as of SINAMICS V4.5.

For the SINAMICS Integrated and SINAMICS S120 CU310-2/CU320-2, a module replacement is also identified via the serial number.

Non-volatile SINAMICS data (NVRAM data) backed up previously via the CU parameter p7775 = 1 on the CF card, is then automatically transferred to the Control Unit.

Fan

Cooling the SIMOTION D410-2

Overview

A fan is always required for operation of the SIMOTION D410-2. This is included in the scope of delivery of the D410-2 control unit. The fan operates temperature-controlled and is switched on depending on the air intake temperature and the CPU load.

Fan faults

Fan faults are indicated as follows:

- Entry in diagnostic buffer
- Indicated via system variable
- Calling the PeripheralFaultTask

If a fan fault occurs or the fan is unplugged, the module continues to run. The controller goes into FAULT state when overtemperature occurs (temperature threshold 2 exceeded), whereby all LEDs flicker red. The state can only exited through power off/on

For further information on the evaluation of fan faults, see Section Overview of fan states (Page 7364).

Overview of fan states

Evaluation of fan faults

SIMOTION D410-2 has a single fan.

Fan faults are detected through a cyclic fan test or when a malfunction is determined when the fan is activated (fan does not rotate or fan rotates at too low a speed).

Fan faults are signaled by the following diagnostic buffer entry:
Fan on the module is defective.

The states that can occur during operation are described in the following.

Table 10-104 Overview of fan states

State	PeripheralFaultTask	System variable [1]
		_cpuDataRW.fanWarning
The fan fails in the STOP operating state, then RUN	PeripheralFaultTask: Is not called	=YES
The fan fails in the RUN operating state	PeripheralFaultTask: TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) TSI#details = 16#00000080	=YES

[1] The "YES" value must be reset to "NO" by the application.

Requirement for/presence of a fan

System variables can be used to evaluate:

- Whether a fan is required for the operation of the device (or not)
- Whether a fan is installed (or not)

Table 10-105 System variable fannecessary/fanexisting

System variable on the device	States	Description
fanbattery of data type StructDeviceFanBattery (the system variables are of the data type EnumFanBattery)		
.fannecessary [1]	MANDATORY	Fan is required for operation of the device. **.fanexisting** can be used to query whether a fan is installed. **Examples: D410-2, D445-2, D455-2**
	OPTIONAL	Fan can be used optionally. **.fanexisting** can be used to query whether a fan is installed. **Examples: D425, D435**
	NOT_MANDATORY	Fan is not required for operation of the device.

System variable on the device	States	Description
.fanexisting [1)]	SINGLE	A single fan is installed **Examples: D410-2**
	REDUNDANT	Double fan is installed. **Examples: D445-2, D455-2**
	NOT_EXISTING	No fan is installed. **Example: D425 and D435 without optional fan**.

1) Value is statically set to MANDATORY/SINGLE in the SIMOTION D410-2.

SIMOTION D410-2 only detects an unplugged fan indirectly.
The system variable fanexisting has statically the state SINGLE for the SIMOTION D410-2 . If the module temperature increases to impermissible values due to a fan having been unplugged, the module signals overtemperature.

The fan speed is available in system variable **_cpuData.fanRpm** as of V4.4.

Additional references

You will find detailed information on setting up Taskstartinfo (#TSI) in the *SIMOTION Runtime Basic Functions* Function Manual.

Response to overtemperature

Operation at overtemperature reduces the module service life and can result in damage to the module.

Causes

Cause of problems in the heat dissipation of the module can be, for example:

- Violation of the maximum permissible air intake temperature
- Free convection is not ensured (clearances are not maintained, pollution, convection is prevented by cables)
- Impermissible mounting position of the module

Temperature thresholds

The internal module temperature is monitored via two module-specific temperature thresholds:

- Overtemperature is signaled when the first (lower) temperature threshold is exceeded.
- When the temperature falls below the first temperature threshold again (minus a hysteresis of approx. 5° C), "Normal temperature" is signaled.
- When the second (higher) temperature threshold is exceeded, the module goes into the FAULT state to protect itself.

The internal module temperature is available in system variable **_cpuData.moduletemperature** as of V4.4.

Response to overtemperature

Table 10-106 Response of the temperature monitoring

Temperature...	Response
... exceeds the 1st temperature threshold (overtemperature)	Call of the PeripheralFaultTask: • TSI#InterruptI = _SC_PC_INTERNAL_FAILURE (= 205) • TSI#details = 16#00000002 Diagnostic buffer entry: "Temperature exceeded in the housing"
... falls below the 1st temperature threshold minus a hysteresis of approx. 5° C	Call of the PeripheralFaultTask: • TSI#InterruptId = _SC_PC_INTERNAL_FAILURE (= 205) • TSI#details = 16#00000004 Diagnostic buffer entry: "Temperature in the housing has returned to normal"
... exceeds the 2nd temperature threshold	Module goes into the FAULT state to protect itself (all LEDs flicker red) Diagnostic buffer entry: "Temperature in the housing too high, self-protection function of the module activated"

Behavior of SINAMICS Integrated

If the internal temperature of the SIMOTION D410-2 exceeds the permissible limit value, the following warning message is displayed:

A1009: CU warning: Control unit overtemperature

The warning message will be cleared as soon as the fault is rectified and the temperature falls below the maximum permissible limit.

The behavior for overtemperatures in the power unit as well as possible STOP responses for the SIMOTION D410-2 corresponds to the behavior for a SINAMICS S120 CU310-2.

10.1.2.6 Parameter assignment / addressing

Software requirements

Engineering

The following requirements must be satisfied for the commissioning of the SIMOTION D410-2 Control Units:

- SIMOTION D410-2 DP: at least SCOUT V4.3 SP1 HF1 and firmware V4.3 SP1 HF2
- SIMOTION D410-2 DP/PN: at least SCOUT V4.3 SP1 HF3 and firmware V4.3 SP1 HF3

Please note the information on the latest SIMOTION SCOUT DVD.

For information on how to install SIMOTION SCOUT on your PG/PC, see the *SIMOTION SCOUT* Configuration Manual.

Note

The software configuration is described in this manual based on SIMOTION SCOUT and SIMATIC STEP 7 Version V5.x.

Information of configuration of the SIMOTION D Control Units in the Engineering Framework Totally Integrated Automation Portal (SCOUT in the TIA Portal), you will find in the configuration manual *SIMOTION SCOUT TIA*.

The TIA Portal requires at least SIMOTION SCOUT V4.4 and SIMOTION D4xx-2 Control Units as of firmware V4.3.

Creating a project and configuring the communication

Creating a SIMOTION project and inserting a SIMOTION D410-2

Procedure

Proceed as follows to create a new project in SIMOTION SCOUT and insert a SIMOTION D410-2:

1. Select the Project > New... menu command.
2. Enter a name for your project in the "New Project" dialog box and confirm your entry with "OK".
   A new folder with the name of the project will be created in the project navigator.

3. In the project navigator, double-click "Insert SIMOTION device". The "Insert SIMOTION Device" dialog box is opened:

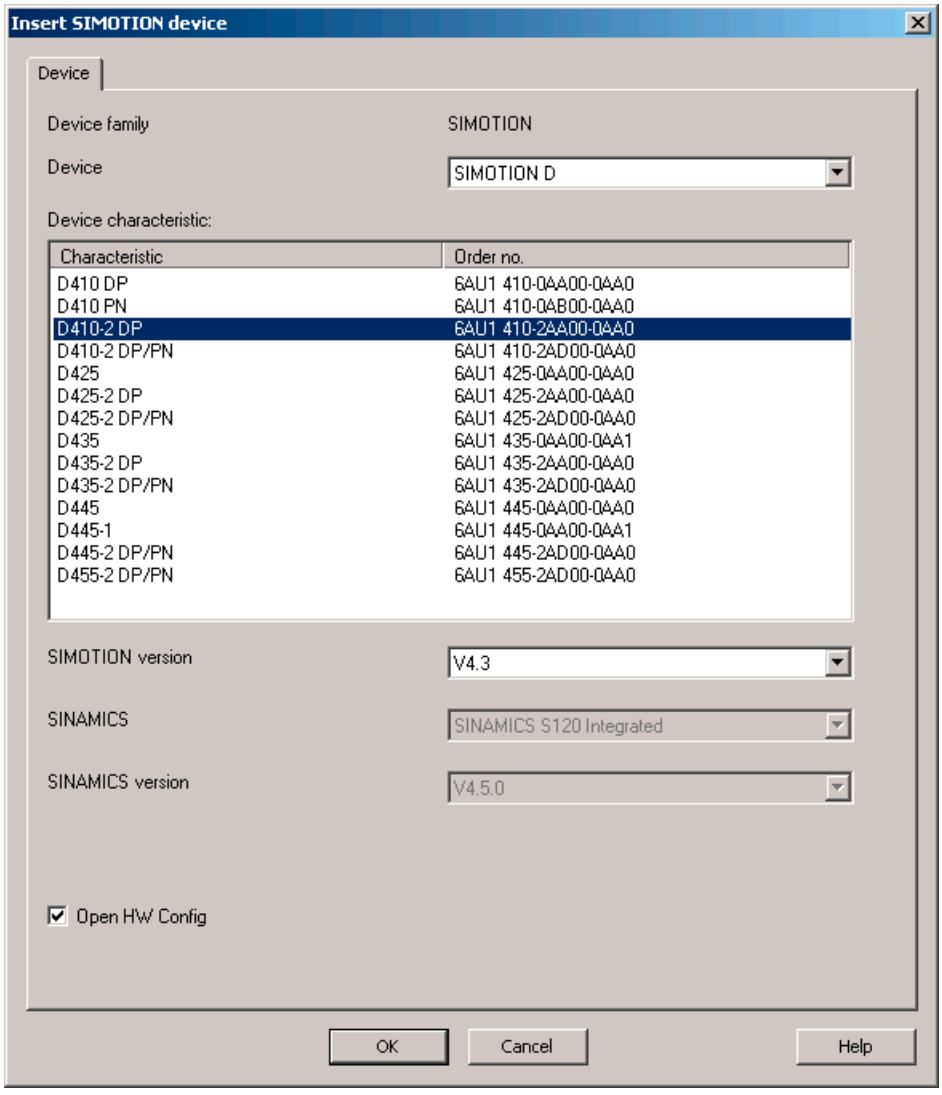


Figure 10-162 Inserting a SIMOTION device

4. In the "Insert SIMOTION Device" dialog box, select the device, its version and the SIMOTION version.
5. If required, make further settings:
   - SINAMICS: For SIMOTION D410-2 always set "SINAMICS S120 Integrated".
   - SINAMICS version: Select the SINAMICS Integrated version if several drive versions are available for a SIMOTION version.
6. The "Open HW Config" option allows you to select whether **HW Config** is opened in the next step (e.g. to configure the bus interfaces).
7. Confirm the "Insert SIMOTION Device" dialog with "OK".

SINAMICS Integrated type

For SIMOTION D410-2, the type is always SINAMICS S120 Integrated.

Version of the SINAMICS Integrated

Depending on the selected SIMOTION version, several versions are available for the SINAMICS Integrated. Please note that separate SIMOTION D firmware is available for each version of the SINAMICS Integrated.

Configuring the PROFINET interface

After you have acknowledged the "Insert SIMOTION Device" dialog with "OK", the "Properties - Ethernet Interface" dialog box opens in the case of a D410-2 DP/PN.

If you are using the PROFINET interface, set the interface properties in the "Properties - Ethernet Interface" dialog box.

To this end, proceed as follows:

1. Click the "New" button.
   The "New Subnet Industrial Ethernet" dialog box opens. Rename the new subnet, or accept the default name by clicking "OK".
2. Select the new Ethernet subnet, which is now displayed in the "Properties - Ethernet Interface" dialog box.
3. Enter the required addresses in the "IP address" and "Subnet mask" fields of the "Properties - Ethernet Interface" dialog box. Under "Gateway", define whether you are going to use a router and, if yes, enter the router address. Confirm with "OK".

Result

If you have not yet configured a PG/PC in your project, you can select the interface for the PG/PC connection now.

Configuring the PROFIBUS PG/PC interface

Requirements

The following requirements must be satisfied in order to configure the PG/PC interface:

- You have completed the "Insert SIMOTION Device" dialog box with "OK"
- A PG/PC has not yet been configured in the project.

If these requirements have been satisfied, you can configure the interface for the PG/PC connection in the "Interface Selection - D410" dialog box.

Procedure

Proceed as follows to configure the PROFIBUS DP interface:

1. In the "Interface Selection - D410" dialog box, select the entry "PROFIBUS DP/MPI (X21)" (in the case of the D410-2 DP, the following alternative is also available: PROFIBUS DP (X24)).

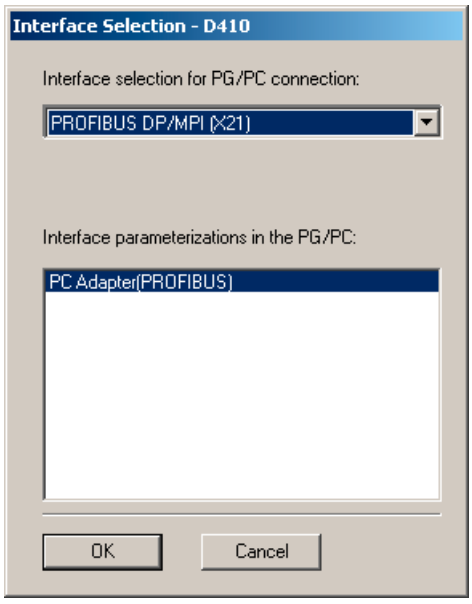


Figure 10-163 Selecting a PROFIBUS interface

2. Select the interface parameter assignment that you would like to use to go online, and confirm with "OK".

The dialog box is closed, the SIMOTION D410-2 is created in the project navigator and **HW Config** is started automatically (if parameterized).

A PROFIBUS subnet with factory settings (1.5 Mbit/s transmission rate) is created automatically.

Result

The PG/PC is now connected to the SIMOTION D410-2 via PROFIBUS. You can configure and parameterize your system.

Note

If you do not use the factory settings, you must configure the PROFIBUS interfaces in **HW Config**.

Please make sure that the S7 online access has been **activated** (PG/PC connection must be yellow and bold in **NetPro**).

Inserting a further SIMOTION device

If you insert a further SIMOTION device using "Insert SIMOTION device", the PG/PC interface selection dialog box is not displayed. Any further SIMOTION device is automatically connected to the PG/PC via PROFIBUS and a new unique DP address (address 4, 5 ... up to 125) is calculated.

Additional references

Further information on the topic of "Going online" can be found

- In the online help via the "Contents" tab at
  - "Diagnostics" > "Overview of service and diagnostics options" > "Part III" > "Go online"
  - "Insert device and connect to target system" > "Go online/offline"
- On the Internet at Internet address (https://support.industry.siemens.com/cs/ww/en/view/22016709)
- In SIMOTION Utilities & Applications, FAQ "Online connections to SIMOTION devices".

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Configuring the Ethernet PG/PC interface

Requirements

The following requirements must be satisfied in order to configure the PG/PC interface:

- You have completed the "Insert SIMOTION Device" dialog box with "OK".
- A PG/PC has not yet been configured in the project.

If these prerequisites have been satisfied, you can configure the interface for the PG/PC connection in the "Interface selection - D410" dialog box.

Proceed as follows to configure the Ethernet interface:

Procedure

1. In the "Interface Selection - D410" dialog box, select the entry "Ethernet PNxIE (X127)".

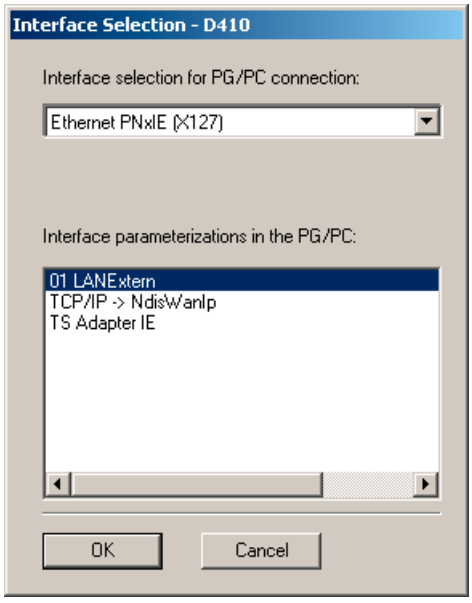


Figure 10-164 Selecting the Ethernet interface

2. Select the interface parameter assignment that you would like to use to go online, and confirm with "OK".

The dialog box is closed, the SIMOTION D410-2 is created in the project navigator and **HW Config** is started automatically (if parameterized).

An Ethernet subnet with factory settings is created automatically. For factory settings, see Section General information about communication via Ethernet (Page 7396).

Result

The PG/PC is now connected to the SIMOTION D410-2 via Ethernet.

You can configure and parameterize your system.

Note

If you want to change the default settings for IP addresses and the transmission rate, you must configure the Ethernet interfaces in **HW Config** and **NetPro**.

Make sure that the PG/PC and SIMOTION D410-2 are located in the same subnet and that S7Online access has been **activated** (the PG/PC connection must appear yellow and bold in **NetPro**).

Inserting a further SIMOTION device

If you insert a further SIMOTION device using "Insert SIMOTION device", the PG/PC interface selection dialog box is not displayed. The second SIMOTION device is automatically connected to the PG/PC via Ethernet and a new unique IP address (last digit + 1 up to 255) is calculated.

Additional references

Further information on the topic of "Going online" can be found

- In the online help via the "Contents" tab at
  - "Diagnostics" > "Overview of service and diagnostics options" > "Part III" > "Go online"
  - "Insert device and connect to target system" > "Go online/offline"
- On the Internet at Internet address (https://support.industry.siemens.com/cs/ww/en/view/22016709)
- In SIMOTION Utilities & Applications, FAQ "Online connections to SIMOTION devices".

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Display of the SIMOTION D410-2 in HW Config

Once you have created a project and inserted a SIMOTION D410-2 as module, **HW Config** opens automatically (if parameterized).

In **HW Config**, the SIMOTION D410-2 is displayed with the SINAMICS Integrated and the interfaces.

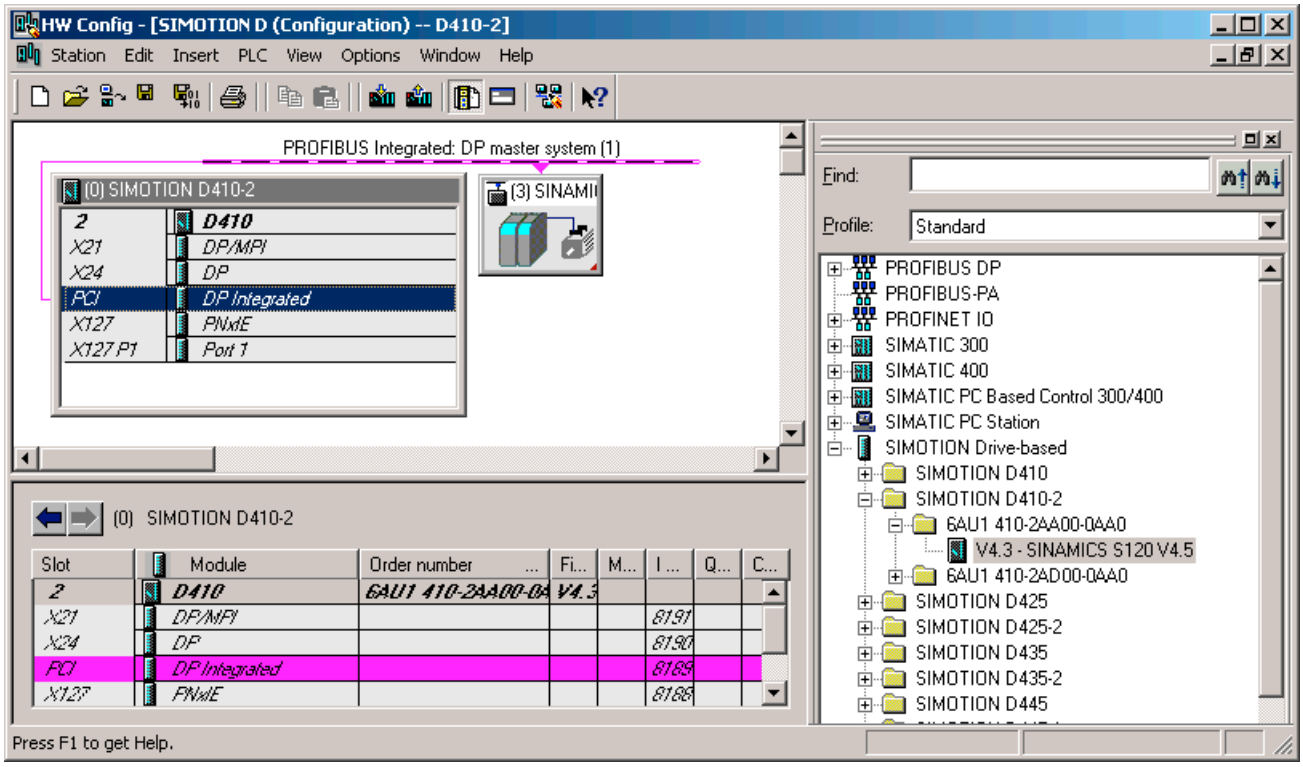


Figure 10-165 Display of a SIMOTION D410-2 in HW Config

Note

According to the DNS conventions, "/" is a permissible character. For this reason, the Ethernet and PROFINET interfaces have a different name in the engineering software than on the module lettering ("/" is replaced by "x").

Example: PN/IE (lettering on module) → PNxIE (as shown in SCOUT, HW Config, NetPro)

Configuring PROFIBUS DP

General information about PROFIBUS DP communication

Definition of PROFIBUS DP

PROFIBUS DP is an international, open field bus standard specified in the European field bus Standard EN 50170 Part 2. PROFIBUS DP is optimized for high-speed, time-sensitive data transfer at field level.

Components communicating on PROFIBUS DP are classified as master and slave components.

- Master (active bus device)
  Components that represent a master on the bus define data transfer along the bus, and are therefore known as active bus nodes.
  Masters components are divided into two classes:
  - DP master class 1 (DPMC1):
    Central master devices are thus designated, which exchange information with the slaves in specified message cycles.
    Examples: SIMOTION D410-2 DP, C240, P350, SIMATIC S7, etc.
  - DP master class 2 (DPMC2):
    These are devices for configuration, commissioning, and operator control and monitoring while the bus is in operation.
    Examples: Programming devices, operator control/monitoring devices
- Slaves (passive bus nodes):
  These devices may only receive, acknowledge and transfer messages to a master when so requested.
  Examples: SINAMICS drives, I/O modules

Functions on PROFIBUS DP

The functional scope can differ between DP masters and DP slaves. The functional scope is different for DP-V0, DP-V1 and DP-V2.

These functions on the PROFIBUS DP are characterized by:

- Configurable, equidistant, isochronous PROFIBUS DP cycle
- Synchronization of slaves by the master by means of a global control message frame in each cycle clock
- Independent maintenance of the isochronous cycle clock by the slaves in the event of a short-term communication failure.

Additional references

You will find additional information about PROFIBUS DP in the *SIMOTION Communication* System Manual.

Operating SIMOTION D410-2 on PROFIBUS DP

PROFIBUS DP interfaces (X21, X24)

For connection to PROFIBUS DP, the SIMOTION D410-2 DP has two interfaces (X21 and X24) and the SIMOTION D410-2 DP/PN has one interface (X21).

Table 10-107 SIMOTION D410-2 PROFIBUS interfaces

	D410-2 DP	**D410-2 DP/PN**
PROFIBUS DP/MPI interface	X21	X21
PROFIBUS DP interface	X24	–

Transmission rates up to 12 Mbits/s are possible. Interfaces X21 and X24 can both be operated isochronously.

Alternatively, the X21 interface can be used as an MPI interface with a transmission rate of 19.2 kbit/s up to 12 Mbit/s, see Section Configuring the MPI bus (Page 7385).

In the delivery condition, PROFIBUS DP interfaces X21 and X24 are both preset as a master with address 2 and a transmission rate of 1.5 Mbit/s. The PROFIBUS DP network is automatically detected and generated for this setting.

However, other settings can also be configured. This requires that you configure the network manually using **HW Config** and **NetPro**.

Note

Communication with the SINAMICS Integrated of a SIMOTION D410-2 is always equidistant. Here, the SIMOTION D410-2 is the master and the SINAMICS Integrated drive is the slave.

SIMOTION D410-2 DP master-slave configuration

Master-slave configurations can be used, for example, to establish hierarchical PROFIBUS networks that can be used to implement a modular machine concept.

Table 10-108 SIMOTION D410-2 DP master-slave configuration

X21 DP/MPI	X24 DP	Remark	Actions in the application
DP master, isochronous	DP slave, isochronous	Application synchronized to DP master (X21), application controls synchronization to DP slave (X24) Internal drive is synchronous with external cycle clock Cycle clock X21 = cycle clock DP Integrated	DP master / DP slave synchronization mechanisms
DP slave, isochronous	DP master, isochronous	Application synchronized to DP master (X24), application controls synchronization to DP slave (X21) Internal drive is synchronous with external cycle clock Cycle clock X24 = cycle clock DP Integrated	DP master / DP slave synchronization mechanisms
DP master, not isochronous	DP slave, isochronous	Application synchronized to DP slave (X24) (can be monitored by the application) Internal drive is synchronous with X24	DP slave synchronization mechanisms
DP slave, isochronous	DP master, not isochronous	Application synchronized to DP slave (X21) (can be monitored by the application) Internal drive is synchronous with X21	DP slave synchronization mechanisms
DP master, isochronous	DP master, isochronous	Application synchronized to DP master (X24, X21) Internal drive is synchronous with external cycle clock Cycle clock X24 = cycle clock X21 = cycle clock DP Integrated	None
DP master, not isochronous	DP master, isochronous	Application synchronized to DP master (X24) Internal drive is synchronous with X24 Cycle clock X24 = cycle clock DP Integrated	None
DP slave, not isochronous	DP master, isochronous	Application synchronized to DP master (X24) Internal drive is synchronous with X24 Cycle clock X24 = cycle clock DP Integrated	None
DP master, isochronous	DP master, not isochronous	Application synchronized to DP master (X21) Internal drive is synchronous with X21 Cycle clock X21 = cycle clock DP Integrated	None
DP master, isochronous	DP slave, not isochronous	Application synchronized to DP master (X21) Internal drive is synchronous with X21 Cycle clock X21 = cycle clock DP Integrated	None
DP master, not isochronous	DP master, not isochronous	Application synchronized to internal drive cycle clock	None
DP master, not isochronous	DP slave, not isochronous	Application synchronized to internal drive cycle clock	None
DP slave, not isochronous	DP master, not isochronous	Application synchronized to internal drive cycle clock	None
DP slave, not isochronous	DP slave, not isochronous	Application synchronized to internal drive cycle clock	None

X21 DP/MPI	X24 DP	Remark	Actions in the application
DP slave, not isochronous	DP slave, isochronous	Application synchronized to DP slave (X24) (can be monitored by the application) Internal drive is synchronous with X24	DP slave synchronization mechanisms
DP slave, isochronous	DP slave, not isochronous	Application synchronized to DP slave (X21) (can be monitored by the application) Internal drive is synchronous with X21	DP slave synchronization mechanisms

For detailed information about controlling synchronization across the application, see the *Basic Functions for Modular Machines* Description of Functions.

Assigning PROFIBUS addresses in HW Config

Assigning PROFIBUS addresses

Assign a PROFIBUS address to all devices **before** you start networking these in order to enable intercommunication.

Note

Before you assign any PROFIBUS addresses, please remember that all addresses must be unique on the PROFIBUS subnet.

Define the PROFIBUS address separately for each device with the PG/PC in **HW Config**. Certain PROFIBUS DP slaves are equipped with an address switch.

Note

The PROFIBUS addresses set at the devices using these switches must correspond with the address settings in **HW Config**.

Recommendation for PROFIBUS addresses

Reserve PROFIBUS address "0" for a service PG and "1" for a service HMI device which are connected to the subnet as required.

Recommended PROFIBUS address setting for SIMOTION D410-2 in case of replacement or service:

Reserve address "2" for a SIMOTION D410-2. This prevents duplicate addresses when installing a SIMOTION D410-2 with a default setting on the subnet (e.g. when a SIMOTION D410-2 is replaced). Assign addresses higher than "2" to any additional devices on the subnet.

Setting the DP cycle and system cycle clocks

Adapting the DP cycle of SINAMICS Integrated

SINAMICS Integrated serves as the basis for all cycle clocks of a SIMOTION D410-2 under the following circumstances:

- SIMOTION D410-2 DP: Always
- SIMOTION D410-2 DP/PN: Only if no synchronized data exchange takes place via the PROFINET interface (see also Section Setting a send cycle clock and system cycle clocks (Page 7391)).

To set the DP cycle of the SINAMICS Integrated, double-click the SINAMICS drive on the integrated PROFIBUS. The "DP Slave Properties" dialog box opens. You can synchronize the DP cycle of SINAMICS Integrated in the "Isochronous mode" tab.

Table 10-109 SIMOTION D410-2 value range

DP cycle	≥ 0.5 ms (DP internal)≥ 1 ms (DP external)
Grid	0.125 ms (the use of a clock grid ≥ 0.250 ms is recommended)

External DP interfaces can only be operated with a DP cycle of ≥ 1 ms.

SINAMICS Integrated always runs in isochronous mode. The cyclic tasks of SIMOTION are therefore always in synchronism with SINAMICS Integrated.

The set DP cycle of SINAMICS Integrated is displayed as "Bus data cycle" in the "System Cycle Clocks - D410" dialog box in SIMOTION SCOUT. Select the SIMOTION D410-2 in the project tree, and then select the "Set system cycle clocks" option in the "Target system" > "Expert" menu.

The table below shows the ratios you can set for the system cycle clocks of SIMOTION D410-2 based on the bus cycle clock.

Table 10-110 Ratios of system cycle clocks

Servo cycle clock[1] : Bus cycle clock	**IPO cycle clock: Servo cycle clock**	**IPO2 cycle clock: IPO cycle clock**
1 ... 4, 8	1 ... 6	2 ... 64

[1] When using the TO axis and the integrated drive control, the minimum servo cycle clock is 1 ms.

Note

The following statements relate to a SIMOTION D410-2 DP with 2 DP interfaces (X21/X24). The statements are equally applicable to a SIMOTION D410-2 DP/PN with only one DP interface (X21).

In addition, if the DP interfaces (X21/X24) are configured as equidistant master interfaces, you must set both DP cycles equal to the bus cycle clock of the SINAMICS Integrated in **HW Config**.

If the DP interfaces (X21/X24) are operated as the master, the system cycle clocks are obtained from an internal cycle clock of the module.

Of the two DP interfaces (X21/X24), no more than one can also be operated as an isochronous slave interface. In this case, the system cycle clocks are obtained from the cycle clock of the slave interface. As a result, the task system of SIMOTION and SINAMICS Integrated runs synchronously to the slave cycle clock. This assumes that a slave cycle clock exists and synchronization with the slave cycle clock has been achieved. If this is not the case, the system cycle clocks are acquired from an internal replacement cycle clock.

When the project is downloaded, the cycle clock configuration is downloaded to the SIMOTION D410-2 and automatically set according to the specifications.

Also observe the cycle clock setting rules in Section Rules for cycle clock settings with SIMOTION D410-2 DP (Page 7379).

Rules for cycle clock settings with SIMOTION D410-2 DP

The rules for making SIMOTION D410-2 DP cycle clock settings are described below.

For SIMOTION D410-2 DP/PN, see Section Rules for cycle clock settings with SIMOTION D410-2 DP/PN (Page 7394).

Rules for adjusting system cycle clocks in the case of SIMOTION D410-2 DP

You must conform to the following rules when setting the DP cycle and SINAMICS Integrated cycle clocks:

1. The DP cycle must be an integer multiple of the current controller cycle.
2. The master application cycle (T_{mapc}), which corresponds to the servo cycle, must be an integer multiple of the speed controller cycle. The smallest-possible T_{mapc} thus results from the smallest common multiple of the DP cycle and the speed controller cycle.
   If the master application cycle = 1, it follows that the DP cycle is also an integer multiple of the speed controller cycle.
3. The DP cycle must be an integer multiple of the basic cycles r0110[x] (DRIVE-CLiQ basic sampling rates).
   You can determine parameter r0110[x] using the expert list in SIMOTION SCOUT (select the "Control_Unit" from "SINAMICS_Integrated" in the project navigator, and then open the "Expert list" by selecting the "Expert" command from the shortcut menu).

Note

An overview of errors reported by the SINAMICS Integrated is provided in the *SINAMICS S* List Manual.

Dependencies of SINAMICS cycle clocks

The general rule for SIMOTION D410-2 DP is that the DP cycle serves as the basic cycle clock for the task system. All SIMOTION clock cycles (servo, IPO, IPO_2, etc.) longer than this basic cycle clock must be an integer multiple of the basic cycle clock.

This rule also applies to SINAMICS Integrated cycle clocks, if one of the successive cycle clocks is longer than the basic cycle clock:

- Speed controller p0115[1] (drive)
- Flow controller p0115[2] (drive)
- Setpoint channel p0115[3] (drive)
- Position controller p0115[4] (drive)
- Positioning p0115[5] (drive)
- Technology controller p0115[6] (drive)
- Onboard I/O p0799[0...2] (Control Unit)
- Terminal module I/O p4099

The corresponding cycle must be an integer multiple of the basic clock (DP cycle).

If any change to the DP cycle violates this rule you must also change the SINAMICS cycle clocks. In order to change the cycles in the Expert list of **SIMOTION SCOUT**, select the "Control_Unit" or the "Drive" from "SINAMICS_Integrated" in the Project Navigator, and then open the "Expert List" by selecting the "Expert" command from the shortcut menu.

If p0115 sampling times are required which cannot be set using p0112 > 1, then you can directly set the sampling times using p0115. This requires that p0112 must be set to "0" (expert).
If p0115 is changed online, the values of the higher indices are adapted automatically.

Note

At the SINAMICS end, there are further rules for setting the sampling times. You will find these described in the *SINAMICS S120* Function Manual, Section Rules for setting the sampling time.

Example

A default value of 4 ms is set for the SINAMICS Integrated setpoint channel p0115[3]. If the DP cycle is to be set to 3 ms, the fact that an integer multiple value is required means that you will need to set 3 ms or 6 ms (for example) for the setpoint channel.

Incorrect cycle clock setting

If the SINAMICS cycle clocks are not set correctly, this can be seen by the following messages:

- A01223 CU: Sampling time inconsistent and/or
- A01902 PB/PN isochronous operation parameterization impermissible and/or
- F01043 severe error downloading project
- F01951 CU SYNC: Synchronization application cycle clock missing

In this case, check the cycle clock settings on all drive objects signaling this error. For F01951, check the DP cycle. In this case, use a DP cycle that is a multiple of 0.25 ms or 0.5 ms.

Note

An overview of errors reported by the SINAMICS Integrated is provided in the *SINAMICS S* List Manual.

Current controller cycle clock

With the SIMOTION D410-2, the following current controller cycle clocks p0115[0] can be configured for the SINAMICS Integrated:

- for servo: 125 µs (default) or 250 µs
- For vector and vector V/f:
  - For blocksize Power Modules: 250 µs or 500 µs (default)
  - For chassis Power Modules: 375 µs.

See also

Using vector drives (Page 7431)

Cycle clock scaling of external PROFIBUS interface to internal PROFIBUS interface

Definition

Cycle clock scaling means that an external PROFIBUS interface of the SIMOTION D410-2 DP (X21/X24) or SIMOTION D410-2 DP/PN (X21) can be operated in an integer multiple of the internal PROFIBUS interface. This reduces the CPU load, thereby allowing you to operate more axes, for example. The settings of the scaled cycle clocks for the external DP interfaces are made in **HW Config**.

Supplementary conditions

The following supplementary conditions are applicable to cycle clock scaling:

- An external DP interface of the D410-2 is used as an isochronous slave interface. Only in this case can an **integer** cycle clock scaling of isochronous external DP slave interface to internal interface be specified. This is checked during compilation and an error message is output in the event of noncompliance. If the external DP interfaces are configured as equidistant interfaces but none are configured as slaves and cycle clock scaling is specified for these interfaces, an error is output during compilation.
- For SERVO, IPO, and IPO2, settings can also be made for all permissible cycle clocks. Master and slave axes can run in different IPO levels. Different cycle clocks and phase offsets are tolerated by the system.

  Note

  The IPO cycle clock of the IPO in which the synchronous operation technology object runs must be set equal to the cycle clock of the isochronous external DP slave interface.

- The second external DP interface of a SIMOTION D410-2 DP can be operated as an isochronous master (while the other is an isochronous slave) in order to operate external drives, for example. In this case, the cycle clock must be the same as the cycle clock of the internal PROFIBUS DP. If this condition is not satisfied, an error message is output during compilation.
- One or both external DP interfaces can also be operated as non-isochronous, free-running interfaces. In this case, there is no effect on the cycle clock settings.

Application example

The system consists of a synchronous master (DP master) and at least one SIMOTION D410-2 DP synchronous slave (DP slave). The synchronous master contains the master axis; the synchronous slave contains the following axes:

- The axes in the SINAMICS Integrated of the D410-2 DP synchronous slave must exhibit high performance with a servo cycle clock of 1.5 ms and an internal DP cycle of 1.5 ms. This requires that the internal fast PROFIBUS DP be decoupled from the slower external PROFIBUS DP.
- The PROFIBUS DP has, for example, a cycle time of 6 ms due to the quantity framework on the bus; in all cases, its cycle time exceeds that of the cycle clock of the internal DP interface.
- The master values are transmitted via the DP bus. Further nodes can also be connected to the DP bus, e.g. DP drives, distributed I/O, etc.

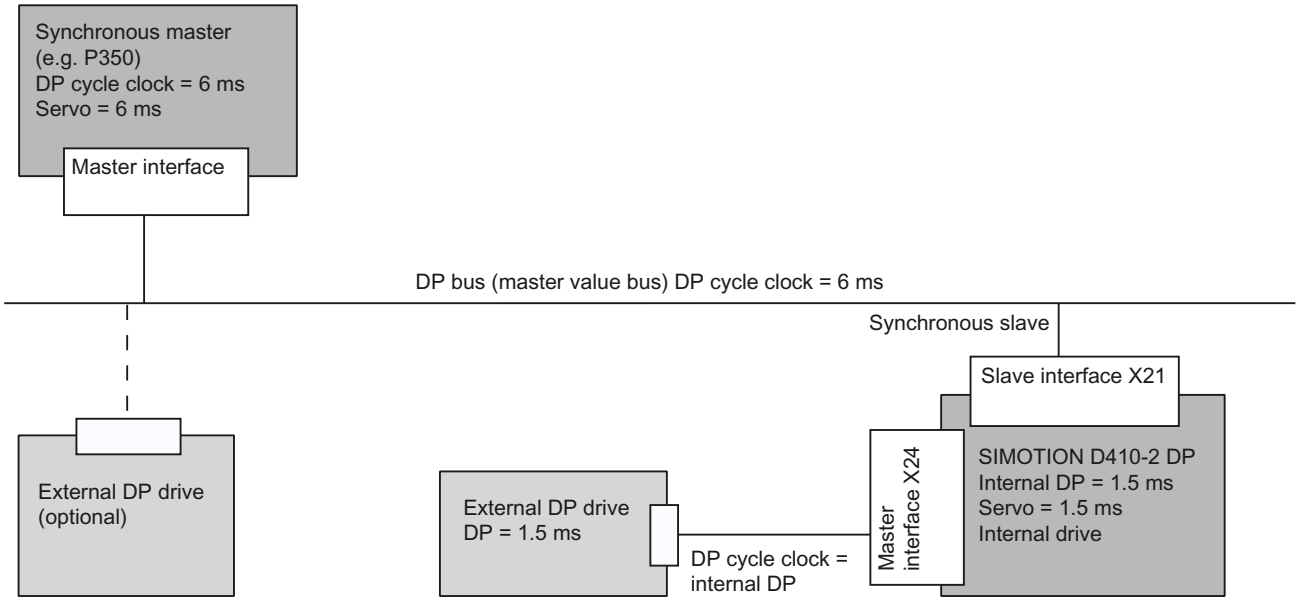


Figure 10-166 Example of a cycle clock scaling for PROFIBUS DP

Creating a new PROFIBUS DP subnet

SIMOTION SCOUT is used to network the SIMOTION D410-2. Set up your user-specific bus parameters for the PROFIBUS DP interfaces when you configure the network.

Note

If a hardware configuration is loaded without a PROFIBUS network being configured on the CPU, a new PROFIBUS address that was previously set in **HW Config** or **NetPro** will not be accepted by the CPU.

Requirement

You have created a project and have inserted a SIMOTION D410-2.

Note

The actions outlined below only need to be taken if you have not selected an interface when integrating the SIMOTION D410-2 into the project (see Section Creating a project and configuring the communication (Page 7367)).

Next the connection to the PG/PC must be established via **NetPro**, refer to:

- Section Configuring the PROFIBUS PG/PC interface (Page 7369)
- Section Configuring the Ethernet PG/PC interface (Page 7371).

Procedure

To create a new PROFIBUS subnet, proceed as follows:

1. Double-click the "D410-2" entry in the project navigator in order to open **HW Config**.
2. In the SIMOTION D410-2 representation, double-click the DP interface for which you want to create a PROFIBUS subnet.
   The "Properties - DP/MPI" dialog box opens.
3. Click "Properties" in the "General" tab to open the "PROFIBUS Interface DP/MPI" dialog box.
4. Click "New" to open the "Properties - New PROFIBUS Subnet" dialog box.
5. Name the new subnet and enter the properties of the new subnet, e.g. transmission rate, on the "Network settings" tab.
6. If the PROFIBUS interface is to be operated isochronously, click "Options". Activate the "Activate isochronous bus cycle" option in the opened dialog box and set the DP cycle. Confirm the settings with "OK" to exit the "Options" dialog box.
7. Accept the settings in the "Properties - New PROFIBUS Subnet" dialog box with "OK".
   The new subnet is now displayed in the "Properties - PROFIBUS Interface DP/MPI" dialog box.
   You can now connect the new subnet to the corresponding PROFIBUS interface.
   Follow the same steps to configure the second PROFIBUS interface for a SIMOTION D410-2 DP.
8. Save and compile the changes.

The PROFIBUS subnet you created is displayed as a graphic object in **HW Config**.

Note

PROFIBUS DP functionality is both equidistant and isochronous in nature. As such, it can guarantee that bus cycles will have exactly the same length and ensures deterministic behavior.

Applications: Connecting drives or synchronized I/O devices.

Additional references

For further information, see the *SIMOTION Runtime Basic Functions* Function Manual, Section "Isochronous I/O processing on fieldbus systems".

Establishing a PG/PC assignment

Introduction

A PG/PC is required to create projects for a SIMOTION D410-2 and download them to the target device. The interface via which the PG/PC can be connected is polled during the automatic communication configuration. If you change these settings, you must reestablish the active designation of the PG/PC in **NetPro** (the PG/PC connection must appear yellow and bold in **NetPro**).

Procedure

1. Open the project in SIMOTION SCOUT.
2. Click the "Open NetPro" button.
   NetPro is accessed, and the configured network is graphically displayed. The PG/PC connection to the configured network is shown in bold in a color other than yellow.
3. Double-click the PG/PC you would like to configure.
   The "Properties - PG/PC" dialog will be displayed with the "Assignment" tab in foreground.
4. Select the interface in the "Assigned" field and activate S7ONLINE access by clicking the appropriate checkbox.
5. Click "OK" to accept the settings.
   The PG/PC connection to the configured network is displayed again in bold and yellow.
6. Save and compile the changes and download them to the SIMOTION D410-2.

Now you can go online again via the PG/PC.

Alternatively, you can make the assignment in SIMOTION SCOUT by clicking the "Assign PG/PC" button. This calls the properties window for PG/PC assignment, where you can modify the assignment and "activate" it (S7ONLINE access).

Configuring the MPI bus

Operating interface X21 as an MPI

The X21 interface can also be operated as an MPI interface.

The typical (default) baud rate is 187.5 Kbaud. A baud rate of up to 12 MBaud can be set for communication with other CPUs. It should be noted, however, that a rate of 12 MBaud is not supported by all CPUs (e.g. smaller SIMATIC S7 CPUs).

The following list provides examples of when using MPI (Multi Point Interface) may prove effective:

- If a PC/PG is being used with an MPI interface
- If an OP/TP only has an MPI interface (newer devices have PROFIBUS or PROFINET interfaces)
- If SIMOTION and SIMATIC CPUs are coupled via XSEND/XRECEIVE

Note

When the X21 interface is used as an MPI bus, additional activation of a drive on this interface is not possible.

Additional references

For general information on MPI, see the *SIMOTION Communication* System Manual.

MPI parameters

MPI bus addresses and data transmission rate

Every node on the MPI bus must have a bus address in the range 0 to 31.

The data transmission rate on the MPI bus can be set to any value for the SIMOTION D410-2.

Communication attempt unsuccessful

If communication cannot be established at all, or if it cannot be established with individual nodes on the MPI bus, check the following:

- Is the transmission rate setting for the SIMOTION D410-2 used for all nodes?
- Are there any loose plug connections?
- Are all bus segments terminated properly?
  Bus segments that are not terminated properly will disrupt communication on the MPI bus.

Configuring PROFINET IO

General information about communication via PROFINET IO

Communication cycle

In PROFINET, the communication cycle is subdivided into different, time-specific intervals. The first interval is used for isochronous real-time communication (IRT), followed by real-time communication (RT) and standard TCP/IP communication. The bandwidth reservation for IRT ensures that RT communication and standard communication have no effect on the transmission of IRT telegrams, which are important for motion control applications.

The following figure shows how the PROFINET communication cycle is divided into isochronous real-time communication (IRT), real-time communication (RT), and standard TCP/IP communication.

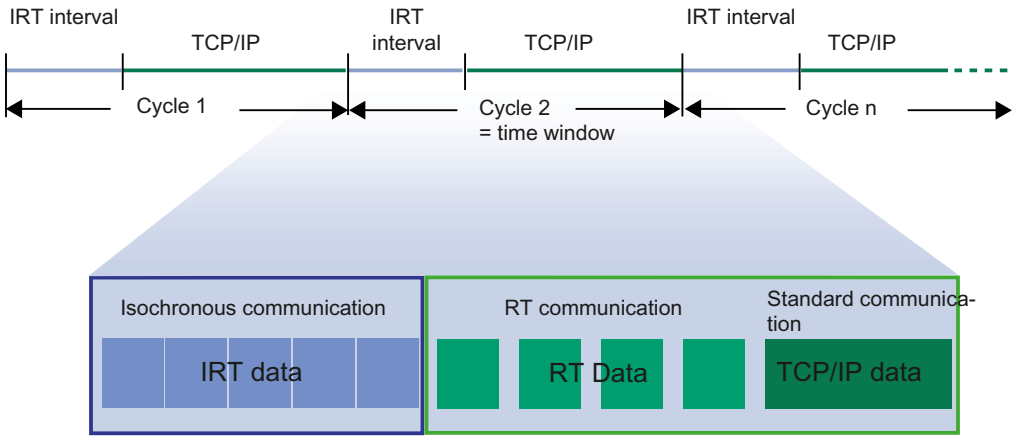


Figure 10-167 PROFINET communication cycle

Isochronous real-time Ethernet

STEP 7 can be used to configure PROFINET devices supporting data exchange via isochronous real-time Ethernet (IRT). IRT telegrams are transferred deterministically via planned communication paths in a defined sequence to achieve the best possible synchronism and performance.

IRT requires special network components supporting a planned data transfer.

Isochronous operation and mode

Equidistant mode and isochronous mode function similarly in PROFINET IO to the way they function in PROFIBUS DP.

For PROFIBUS DP, in isochronous operation all nodes are synchronized using a Global Control Signal created by the DP master.

In PROFINET IO with IRT, a sync master generates a signal to which sync slaves synchronize themselves. Sync master and sync slaves belong to a sync domain which is assigned a name via configuration. The role of the sync master can in principle be played by an I/O controller as well as an I/O device. A sync domain has exactly one sync master.

Context: Sync domain and I/O systems

An important fact is that sync domains do not need to be limited to one PROFINET IO system: The devices of several I/O systems can be synchronized by a single sync master, provided they are connected to the same Ethernet subnet.

The following applies the other way around: An I/O system must only belong to a single sync domain.

Signal propagation delays not negligible

For the extremely exact synchronization interval, line lengths, namely the associated delay times, must be taken into consideration. You can use a topology editor to enter the properties of the lines among the ports of the switches. STEP 7 uses this data and the other configuration data to calculate the optimized process of the IRT communication and the resulting updating time.

IRT runs in parallel to real-time and TCP/IP communication

Apart from IRT communication, for which a defined bandwidth is reserved within the update time, RT communication and TCP/IP communication are also permitted within the update time.

With RT communication (real-time communication), the cyclic data is transferred between I/O controller and I/O device, but without "best possible synchronism".

With non-synchronized I/O devices, data is exchanged automatically via RT communication.

Due to the fact that TCP/IP communication is also possible, other data, e.g. non-real-time data, configuration data or diagnostic data, can be transported.

PROFINET IO controller

Typically, the function of a PROFINET IO controller is taken on by controllers (e.g. SIMOTION C/P/D, SIMATIC S7 CPUs, ...).

The PROFINET IO controller takes on the master function for I/O data communication of the distributed field devices. The function is comparable to a PROFIBUS DP master class 1.

PROFINET IO device

Distributed field devices such as I/Os, drives (e.g. SINAMICS S120) or operator terminals are designated as IO devices. The function is comparable to a PROFIBUS DP slave.

Addressing

In the delivery condition, the onboard PROFINET IO interface does not have an IP address or a subnet mask.

Note

The IP addresses 192.168.215.240 to 192.168.215.255 are reserved for internal communication in the SIMOTION D410-2 (subnet mask 255.255.255.240). When configuring the PROFINET interface (X150), make sure that the internal addresses are not located within the network of this interface. In IP, the network is defined as an AND link of IP address and subnet mask.

Media redundancy (MRP)

Requirement: SIMOTION V4.4.

It is possible to establish redundant networks via the Media Redundancy Protocol (MRP). Redundant transmission links (ring topology) ensure that an alternative communication path is made available when a transmission link fails. The PROFINET devices that are part of this redundant network form an MRP domain.

MRP guarantees media redundancy in the event of a problem in the ring. The switchover of the ring is performed by the Redundancy Manager.

The switchover times depend on:

- The actual topology
- The devices used and
- The network load in the relevant network

The typical reconfiguration time of the communication paths for TCP/IP and RT frames in the event of a fault is < 200 ms.

In most systems, the switchover time of MRP is far above the PROFINET update time for cyclic data, so that a failure for cyclic data is detected. The PROFINET connection therefore fails and is reestablished after the switchover by the Redundancy Manager. In this way, an error can be corrected in the network, while the system continues to run **with bumps**.

Note

During the interruption of the ring, as well as when correcting the interruption (e.g. repair of the defective cable), there is a brief failure of the communication.

Ring ports

A SIMOTION/SINAMICS device may only be inserted in an MRP ring as a node with MRP-capable ports. For SIMOTION D, the first two ports of the PROFINET IO interfaces are designed as ring ports.

These two ports are marked with an "R" in the module rack in HW Config.

Note

Only devices with MRP-capable ports may be inserted in an MRP ring. If MRP-capable ports are not used, the reconfiguration times can be in the seconds range.

Bumpless media redundancy (MRPD)

Requirement for SIMOTION D410-2:

- SIMOTION SCOUT as of V4.4
- SIMOTION SCOUT TIA as of V4.5

MRPD is a procedure for bumpless media redundancy for PROFINET IO with IRT. MRPD also requires MRP.

The combination of MRP with MRPD provides bumpless PROFINET operation for short cycle times in the event of a fault in the ring. MRPD is based on IRT and ensures bumpless operation by the provider sending the cyclic data in both directions which the consumers then receive twice. If

the ring is interrupted at one position (e.g. through the failure of a ring node), receipt of the cyclic data via the problem-free side of the ring is still guaranteed.

Bumpless media redundancy MRPD always requires the activation of MRP in the individual rings.

A maximum of two Ethernet nodes may be between the sync master and the redundant sync master. If the redundant sync master is used together with MRPD, it is recommended that the redundant sync master be connected directly to the sync master and the two nodes located in a shared cabinet so that the cable connection between both nodes is protected.

If there is an interruption in the link between the sync master and the redundant sync master, the system continues to run without bumps, but after switching off and on again faults can occur.

Too high a network load or too-rapid coming and going of faults can also result in unfavorable cases in the failure of the PROFINET connection with activated MRPD because of delayed or incomplete switchover actions of MRP/MRPD.

For example, with two consecutive faults at different points in the ring, bumpless operation is only ensured when there is approx. three seconds between the two faults.

Additional information

You will find additional information about media redundancy in the *SIMOTION Communication* System Manual.

Setting a send cycle clock and system cycle clocks

Requirement

The SIMOTION system cycle clocks (servo/IPO/IPO_2) are based on the SINAMICS Integrated DP cycle or the PROFINET send cycle clock, depending on the PROFINET real-time class and the type of data transfer. The cycle clock source can be generated "internally" by the SIMOTION D410-2; otherwise, it is obtained "externally" from the clock signals received at the PROFINET interface.

Table 10-111 Basis for the SIMOTION system cycle clocks/cycle clock source

Real-time class in which the PROFINET interface is operated	Data transfer	Basis for the SIMOTION system cycle clocks		Cycle clock source
		DP cycle (Integrated)	PROFINET send cycle clock [1]	
RT communication		X		Internal
IRT communication	The D410-2 DP/PN is the SYNC master in the I/O system; synchronized data traffic takes place [2]		X	Internal
	The D410-2 DP/PN is the SYNC slave in the I/O system; synchronized data traffic takes place [2]		X	External Internal (as substitute value)
	No synchronized data traffic [2]	X		Internal

[1] If the PROFINET send cycle clock forms the basis for the SIMOTION system cycle clocks, the DP cycle (SINAMICS Integrated and external PROFIBUS interface) and the servo cycle clock must be the same.

[2] Synchronized data traffic, e.g. by means of:
- controller-controller slave-to-slave communication
- IO device in its own IO system

Setting the DP cycle in HW Config

To set the DP cycle of the SINAMICS Integrated, double-click the SINAMICS block on the integrated PROFIBUS in HW Config.

The "DP Slave Properties" dialog box opens. You can adjust the DP cycle of the SINAMICS Integrated on the "Isochronous mode" tab. See also Setting DP Slave properties (Page 7429).

Table 10-112 SIMOTION D410-2 DP/PN value range

	D410-2 DP/PN
DP cycle	≥ 0.5 ms (DP internal) ≥ 1.0 ms (DP external)
Grid	0.125 ms (the use of a clock grid ≥ 0.250 ms is recommended)

If, in addition to the drive on the SINAMICS Integrated, external drives are also to be connected via isochronous PROFIBUS, the DP cycle must be ≥ 1 ms.

Setting the send cycle clock in HW Config

The send cycle clock for PROFINET IO must be set in the "Domain Management" dialog in **HW Config**. To do this, select the "Edit" > "PROFINET IO" > "Domain management ..." menu command in **HW Config** and set the desired cycle clock.

The PROFINET interface can be operated with a send cycle clock in the range of: 0.25 ms ≤ send cycle clock ≤ 4 ms. The smallest configurable time base is 0.125 ms.

Note

Information for version < V4.4

If there are IO devices with RT class "RT" in a sync domain, it is only possible to set the send cycle clocks 0.5 ms, 1 ms, 2 ms, and 4 ms.

Cycle clock scaling

The PROFIBUS cycle clock can be scaled down in relation to the servo cycle clock. The reduction ratio is only permitted when PROFINET IO with IRT has not been configured. A reduction ratio for the PROFINET send cycle clock to the PROFIBUS cycle clock is also possible.

Example:
PROFINET send cycle clock = 0.5 ms
PROFIBUS cycle clock = servo cycle clock = 1 ms

The PROFIBUS cycle clock can be operated relative to the PROFINET send cycle clock at a ratio of 1:1 to 16:1.

The table below shows which ratios you can set for the system cycle clocks of the SIMOTION D410-2 DP/PN based on the DP cycle of SINAMICS Integrated or on the PROFINET send cycle clock.

Table 10-113 Ratios of system cycle clocks

Cycle clock name	Settable factors	Reference cycle clock
PROFIBUS DP bus cycle clock	1, 2, 3, 4, 6, 8, 10, 12, 14, 16	PROFINET IO send cycle clock
Servo[1)]	1, 2, 3, 4, 8[2)]	PROFIBUS DP bus cycle clock
IPO	1, 2, 3, 4, 5, 6	Servo
IPO_2	2, 3, 4, 5, ..., 64	IPO

1) When using the TO axis and the integrated drive control, the minimum servo cycle clock is 1 ms.

2) Always "1", if PROFINET with IRT is configured.

Setting the cycle clock ratios

The set bus cycle clock is displayed in SIMOTION SCOUT as the "Bus data cycle" in the "System Cycle Clocks - D410" dialog box. Select the SIMOTION D410-2 and then select the "Set system cycle clocks" option in the "Target system" > "Expert" menu.

Set the required cycle clock ratios for the servo, IPO, and IPO_2 in the "System Cycle Clocks - D410" dialog box.

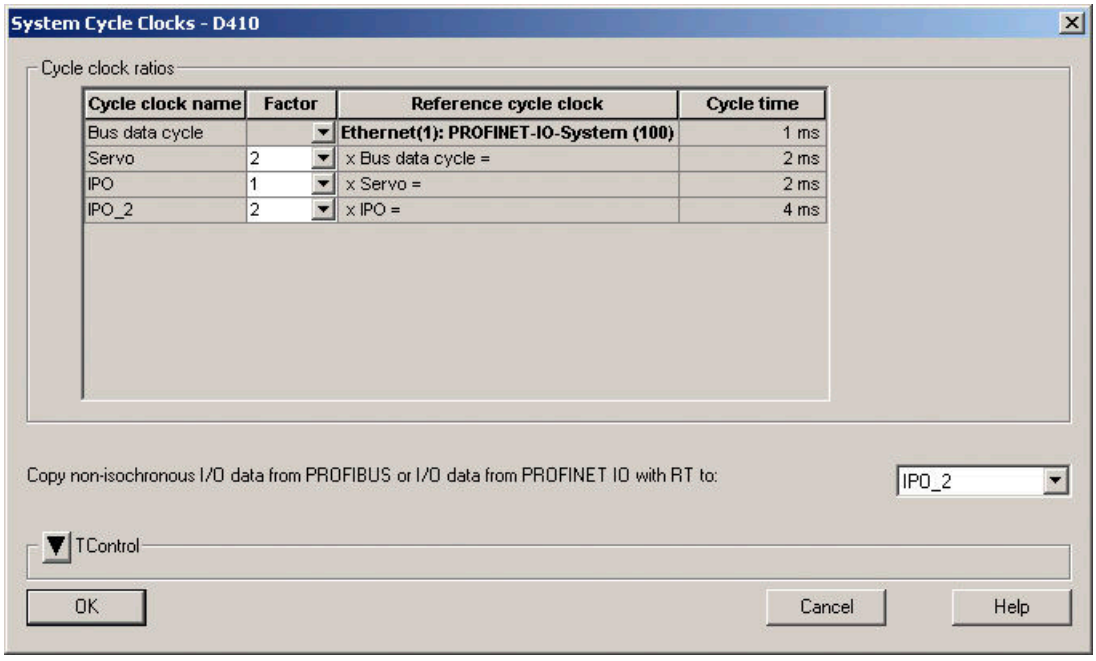


Figure 10-168 System cycle clocks

Additional information

For further information, see the *SIMOTION Communication* System Manual.

Properties of PROFINET

Properties

The onboard PROFINET IO interface supports the parallel operation of:

- IRT - isochronous real-time Ethernet
  - Operation of IRT I/O (e.g. ET 200SP)
  - Operation of a SINAMICS S120 as an IRT device
- RT - real-time Ethernet
  - Operation of RT I/O (e.g. ET 200SP, ET 200pro, etc.)
  - Operation of a SINAMICS S120 as an RT device
- TCP/IP, UDP, HTTP, ... standard Ethernet services

Note

For mixed operation of IRT and RT, make sure that the IRT-compatible devices form what is referred to as an IRT domain; i.e. there must not be any non-IRT devices on the data transmission link between the IRT devices.

Additional references

You will find an overview of the specific properties of PROFINET IO on SIMOTION D in the *SIMOTION Communication* System Manual.

Configuration tasks

Configuration of PROFINET involves the following steps:

1. Insert the SIMOTION D410-2 DP/PN.
2. Configure the onboard PROFINET IO interface in **HW Config**.
3. Create a topology: Here, you specify how the individual ports of the PROFINET IO devices are interconnected.
4. Configure the sync domain: Here, you specify which PROFINET nodes are sync masters (clock generators) and sync slaves.
5. Specify the send clock: Describes the time during which a PROFINET IO device exchanges user data with the PROFINET IO controller.
6. Configure the direct data exchange: The direct data exchange specifies which address areas are to be used for sending and receiving respectively.

Additional references

You will find a detailed description of each configuration step in Section "Configuring PROFINET IO with SIMOTION" of the *SIMOTION Communication* System Manual.

Rules for cycle clock settings with SIMOTION D410-2 DP/PN

The rules for making SIMOTION D410-2 DP/PN cycle clock settings are described below.

For SIMOTION D410-2 DP, see Section Rules for cycle clock settings with SIMOTION D410-2 DP (Page 7379).

Rules for using the PROFINET send cycle

If the PROFINET send cycle clock is the basis for the cycle clocks, ensure that the DP cycle and the servo cycle clock are the same. This applies to PROFIBUS interface X21 and the PROFIBUS of the SINAMICS Integrated.

The basis for the system clocks is generated internally if the PROFINET interface is not operated with RT class IRT or if IRT is set and no data is being transferred. This also applies if the PROFINET interface is operated as synchronization master with RT class IRT and data is being transferred. The SIMOTION device does not have to synchronize itself with an external cycle.

The basis for system clocks is derived from the clock signals received at the PROFINET interface if this interface is operated as synchronization slave with RT class IRT. The SIMOTION device does not have to synchronize itself with this external clock.

A substitute clock of a duration equivalent to the configured clock is generated internally if the PROFINET interface has not received a clock signal.

The clock settings are included in the project download to the SIMOTION device and are adjusted according to specification.

Dependencies of SINAMICS cycle clocks

The rule for SIMOTION D410-2 DP/PN in the case of synchronized data traffic is that the send cycle clock serves as the basic cycle clock for the task system. All SIMOTION clock cycles (servo, IPO, IPO_2, etc.) longer than this basic cycle clock must be an integer multiple of the basic cycle clock.

This rule also applies to SINAMICS cycles, if one of the successive cycles is longer than the basic clock:

- Speed controller p0115[1] (drive)
- Flow controller p0115[2] (drive)
- Setpoint channel p0115[3] (drive)
- Position controller p0115[4] (drive)
- Positioning p0115[5] (drive)
- Technology controller p0115[6] (drive)
- Onboard I/O p0799[0...2] (Control Unit)
- Terminal module I/O p4099

The corresponding cycle must be an integer multiple of the basic clock.

If any change to the send cycle violates this rule you must also change the SINAMICS cycles. In order to change the cycles in the Expert list of **SIMOTION SCOUT**, select the "Control_Unit" or the "Drive" from "SINAMICS_Integrated" in the Project Navigator, and then open the "Expert List" by selecting the "Expert" command from the shortcut menu.

If p0115 sampling times are required which cannot be set using p0112 > 1, then you can directly set the sampling times using p0115. This requires that p0112 must be set to "0" (expert).
If p0115 is changed online, the values of the higher indices are adapted automatically.

Note

At the SINAMICS end, there are further rules for setting the sampling times. You will find these described in the *SINAMICS S120* Function Manual, Section Rules for setting the sampling time.

Example

A default value of 4 ms is set for the SINAMICS setpoint channel p0115[3]. If the send cycle is to be set to 3 ms, the fact that an integer multiple value is required means that you will need to set 3 ms or 6 ms (for example) for the setpoint channel.

Incorrect cycle clock setting

If the SINAMICS cycle clocks are not set correctly, this can be seen by the following messages:

- A01223 CU: Sampling time inconsistent and/or
- A01902 PB/PN isochronous operation parameterization impermissible and/or
- F01043 severe error downloading project
- F01951 CU SYNC: Synchronization application cycle clock missing

In this case, check the cycle clock settings on all drive objects signaling this error. For F01951, check the DP Integrated cycle. In this case, use a DP Integrated cycle that is a multiple of 0.25 ms or 0.5 ms.

Note

An overview of errors reported by the SINAMICS Integrated is provided in the *SINAMICS S* List Manual.

Current controller cycle clock

With the SIMOTION D410-2, the following current controller cycle clocks p115[0] can be configured for the SINAMICS Integrated:

- for servo: 125 µs (default) or 250 µs
- For vector and vector V/f:
  - For blocksize Power Modules: 250 µs or 500 µs (default)
  - For blocksize Power Modules: 375 µs.

See also

Using vector drives (Page 7431)

Configuring an Ethernet subnet

General information about communication via Ethernet

Properties of Ethernet

SIMOTION D410-2 has an onboard Ethernet interface X127 P1 PN/IE.

You can connect an Industrial Ethernet with a transmission rate of 10/100 Mbit/s to the 8-pin RJ45 socket **X127 P1**.

Note

The Ethernet interface supports PROFINET basic services. It therefore has the designation PN/IE.

The Ethernet interface has autocrossing functionality.

Ethernet communication

SIMOTION D410-2 offers the following functions via Industrial Ethernet:

- Communication with STEP 7, SIMOTION SCOUT and SIMATIC NET OPC via a PG/PC
- Communication via UDP (user datagram protocol) with other components, e.g. other D410-2 devices
- Communication with other devices via TCP/IP
- Connection of SIMATIC HMI devices or PC-based HMIs
- IT communication (e.g. via SIMOTION IT OPC XML-DA)
- Communication based on OPC UA (Unified Architecture)
- PROFINET basic services (e.g. DCP, LLDP, SNMP).
  These PROFINET basic services provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives or I/O modules, for example.

Routing

S7 routing is possible from the Ethernet interface to the PROFIBUS interfaces and the PROFIBUS Integrated.

You can find the MAC address on the rating plate located on the front of the SIMOTION D410-2.

For further information on routing, see the *SIMOTION Communication* System Manual.

Default Ethernet addresses

The following IP addresses are assigned by default for the Ethernet interface:

Table 10-114 IP address assignment for SIMOTION D410-2

Interface	Use case	Default address	
X127 P1 PN/IE	Insert SIMOTION device or HW Config	IP address: Subnet mask: Router address: Automatic private IP address	169.254.11.22 255.255.0.0 0.0.0.0
	SIMOTION D410-2 in the delivery condition	IP address: Subnet mask: Router address: Automatic private IP address	169.254.11.22 255.255.0.0 0.0.0.0

Note

The IP addresses 192.168.215.240 to 192.168.215.255 are reserved for internal communication in the SIMOTION D410-2 (subnet mask 255.255.255.240). When configuring the external Ethernet interface (X127 P1), make sure that the internal addresses are not inside their network. In IP, the network is defined as an AND link of IP address and subnet mask.

Note

If you want to go online via Ethernet, you have to make sure that the connection from PG/PC to SIMOTION D410-2 is active. You can check this in **NetPro**. You will find a description of how to reactivate the connection in Section Establishing a PG/PC assignment (Page 7384).

Automatic Private IP Addressing

The Ethernet interface supports the automatic IP assignment according to the Automatic Private IP Addressing (APIPA) process.

Interface X127 P1 already has IP address 169.254.11.22 in the delivery condition. Further information on Automatic Private IP Addressing can be found on the Internet, e.g. in WIKIPEDIA with the keywords *APIPA* or *Zeroconf*.

Configuring the Ethernet connection in HW Config

The Ethernet connection of the SIMOTION D410-2 can be configured in **HW Config**.

Procedure

1. Open your project.
2. Open **HW Config**. Double-click the Ethernet port (X127 P1) to open the "Properties - PNxIE" dialog box.
3. You can configure the Ethernet connection in the "Options" tab.
   Recommendation: Use the default setting "Automatic setting". With Automatic setting, the baud rate and duplex operating mode are automatically aligned with the connection partner. The autocrossing functionality is also available so that you can use crossed and uncrossed cables.
   If transmission is to be set manually, not only must the connection (e.g. 10 Mbit/s half duplex) be set manually but autonegotiation must also be deactivated.
4. Close the "Properties - PNxIE" dialog box with "OK".
5. Save and compile the modified hardware configuration.
6. Load the new hardware configuration to the SIMOTION D410-2 via PROFIBUS DP/Ethernet/ PROFINET IO.

Shielded twisted pair cables are used for the networking. 4- and 8-wire cables can be used for 10/100 Mbit/s.

Note

The TCP/IP timeout parameters are configured in **HW Config** by double-clicking the D410 module in the "Ethernet extended" tab.

Additional references

For additional information, see the *SIMATIC NET, Industrial Twisted Pair, and Fiber Optic Networks* Manual.

For further information about the cabling spectrum for Ethernet, see the *Industrial Communication IK PI* Catalog.

Configuring Ethernet addresses in HW Config

Requirement

For configuration using Industrial Ethernet, SIMOTION D410-2 must be provided with an IP address, the subnet mask and the router address.

Note

Only one router may be configured.

Procedure

To configure and transfer Ethernet addresses to the D410-2, proceed as follows:

1. Open your project.
2. Open **HW Config**. Double-click the interface to be configured (X127) to open the "Properties" dialog box.
3. On the "General" tab, click the "Properties" button of the Ethernet interface. The "Properties - Ethernet Interface" dialog is displayed.
4. Click the "New" button. The "New Industrial Ethernet" subnet dialog is displayed. In this dialog box, you can change the name of the new subnet or confirm the default setting with "OK".
5. The newly created Ethernet subnet is now displayed under "Subnet" in the "Properties - Ethernet Interface" dialog box and must be selected.
6. In this dialog box, enter the required addresses for "IP Address" and "Subnet". Under "Router", choose whether a router is to be used. If using a router, enter the router address.
7. Confirm this dialog box with "OK".
8. Close the "Properties" dialog by clicking "OK".

9. Save and compile the modified hardware configuration.
10. Load the new hardware configuration to the SIMOTION D410-2.

Assigning the Ethernet address later

It is possible to assign the IP address later on (e.g. on modular machines). In this case, an IP address is not assigned in HW Config, but activated differently with the "Set IP address using different method" option. The address will then be assigned later on the machine, for example, by a the user program or by commissioning tools, such as PRONETA.

Further information on PRONETA can be found on the Internet:

- PRONETA: See Internet address (https://support.industry.siemens.com/cs/ww/en/view/67460624)

Reading out IP and MAC address

Requirement

To read out the IP and MAC addresses, the following requirements must be met:

- SIMOTION D410-2 is wired.
- You have assigned the communication parameters.
- You are online.

Procedure

The IP address and MAC address of SIMOTION D410-2 can be displayed as follows via SIMOTION SCOUT.

1. Right-click the module.
2. Select "Target device" > "Device diagnostics" in the context menu.

Example

The addresses are displayed as follows for SIMOTION D410-2:

X127 (IE)

- Active MAC Address: 08-00-06-73-25-3E
- IP address: 169.254.11.22
- Subnet mask: 255.255.0.0
- Standard gateway: 0.0.0.0.

As an alternative, you can determine the IP address as follows:

- By selecting "Project" > "Accessible nodes" in SIMOTION SCOUT
- By calling "Target system" > "Ethernet" > "Edit Ethernet node..." in HW Config and browsing to "Online accessible nodes"
- Using the system function _getIpConfig

Note

The MAC address is listed on the nameplate on the front of the module.

10.1.2.7 Commissioning (software)

Overview of commissioning

Requirements for commissioning

The requirements for commissioning the SIMOTION D410-2 are as follows:

- The system has been connected and wired.
- The SIMOTION D410-2 is switched on and powered up (STOP operating state).
- SIMOTION SCOUT (with integrated STARTER) has been installed and started on the PG/PC.
- Communication between the SIMOTION D410-2 and the PG/PC is configured.
- You have created a project and installed a SIMOTION D410-2 in the project.

Symbolic assignment / adaptation

Symbolic assignment

SIMOTION supports the symbolic assignment on SINAMICS drive objects (DOs) during the configuration of technology objects (TOs) and I/Os.

This simplifies the configuration of the technological relationships including the communication between controller and drive.

With the symbolic assignment:

- Only suitable assignment partners are offered in an assignment dialog box.
- Communication between axis and drive is set up automatically by the engineering system and the required PROFIdrive axis telegrams as well as the used addresses set up.
- Telegrams are extended and interconnections created automatically in the drive depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

- Axis and drive configuration are first performed independently.
- Communication connections are established automatically during the configuration of I/O variables on SINAMICS I/Os (telegrams are set up automatically, the I/Os interconnected to the telegram and the addresses set up).

Apart from the symbolic assignment, no further configuration is required for the communication. As addresses no longer have to be configured, the connection is retained even with address offsets.

Note

During the configuration of drive objects (DO drive, DO encoder, ...) as well as in the Telegram Configuration dialog box (see Section Telegram configuration (Page 7458)), you can deactivate the **automatic telegram configuration** and the **automatic telegram adaptation**.

A deactivation should only be performed in justified exceptional cases because many of the above mentioned benefits are lost.

The symbolic assignment enables an independent configuration of the axes on the SIMOTION side and the drives on the SINAMICS side. This allows the following:

- The PLC and motion control functions can be completely configured by a programmer even without drive know-how using technology objects (e.g. TO axis) and loaded to the device.
- The drives can be separately configured and optimized by a drive expert.
- The technology objects can be symbolically assigned later to the drive objects via an interconnection dialog box.

Note

The previous methods of drive, axis and I/O configuration are still available. Symbolic assignment must be deactivated for these methods.

For newly created projects, the symbolic assignment is used by default.

If projects < V4.2 are upgraded, the symbolic assignment is deactivated by default and must be activated when required.

Symbolic assignment can be activated or deactivated in SIMOTION SCOUT via the menu "Project" > "Use symbolic assignment."

Activating symbolic assignment later

Symbolic assignment is recommended and is automatically activated.

Conversion from upgraded projects to symbolic assignment is possible, but this requires post-editing of the project, especially for free telegram configurations (e.g. for TM15 DI/DO, TM31).

Note

If the symbolic assignment is activated later, previously existing telegram settings and BICO interconnections for all SINAMICS telegrams that are set to Standard/Automatic (Communication -> Telegram configuration) are replaced at the next compilation. This may change telegrams and BICO interconnections and delete telegram extensions previously set up manually.

For this reason, make a backup copy of your project before activating the symbolic assignment.

To retain the settings that have been made, after the selection of "Use symbolic assignment," the setting "User-defined" must be selected and the check box cleared for automatic telegram setting/address adaptation for the respective message before compilation.

For further details, see the *SIMOTION Runtime Basic Functions* Function Manual.

Assigning a drive later

You can create an axis in SIMOTION SCOUT and assign it to a drive later. You can thus load your user program to the controller and (with the exception of the non-existent drives) test it.

Compared to a procedure with temporarily created "virtual axes", "axes without assigned drive" have the advantage that the configuration data is completely available and do not require a "virtual axis -> real axis" reconfiguration.

Simulation of axes

You can also use the axis simulation to test the user program. You can find a script for switching the axis simulation on and off in SIMOTION Utilities & Applications, which is part of the scope of delivery of SIMOTION SCOUT.

Further details can be found in the *TO Axis Electrical/Hydraulic, External Encoder* Function Manual.

Adaptation

In addition to the symbolic assignment, the **automatic adaptation** also facilitates the configuration of SINAMICS S120 data. When SIMOTION devices ramp up, reference variables, along with drive and encoder data for SINAMICS S120, are transferred automatically for the configuration data of the SIMOTION technology objects "Axis" and "External Encoder". This data no longer has to be entered in SIMOTION.

For additional information, see the following sources:

- *SIMOTION Runtime Basic Functions* Function Manual
- *TO Axis Electrical/Hydraulic, External Encoder* Function Manual

Requirement

Symbolic assignment is supported by the TO axis, TO externalEncoder and the TO outputCam, TO camTrack and TO measuringInput. The onboard I/Os of a SIMOTION D, of a SINAMICS S110/S120 Control Unit, and selected Terminal Modules can be interconnected symbolically.

Module	Supports symbolic assignment
SIMOTION D	
• SIMOTION D410-2	• As of SIMOTION V4.3
• SIMOTION D410	• As of SIMOTION V4.2
• SIMOTION D4x5-2	• As of SIMOTION V4.2
• SIMOTION D4x5	• As of SIMOTION V4.2
Controller extension • CX32-2 • CX32	As of SIMOTION V4.2
SINAMICS S110 CU305	As of SINAMICS V4.3
SINAMICS S120	
• CU310-2	• As of SINAMICS V4.4
• CU310	• As of SINAMICS V2.6.2
• CU320-2	• As of SINAMICS V4.3
• CU320	• As of SINAMICS V2.6.2

See also

Only the drive configuration by means of symbolic assignment is described in this documentation.

You will find the documentation of older SIMOTION versions on the Internet at the following address (https://support.industry.siemens.com/cs/ww/en/view/40211807).

For further information on the configuration of the TO axis and TO externalEncoder, see the *TO Axis Electrical/Hydraulic, External Encoder* Function Manual.

Procedure when commissioning

Commissioning steps

This section shows you how to configure a system and test the configured drives and axes. The steps in commissioning are listed below in the order as recommended:

1. Configuring SINAMICS Integrated
   The integrated drive SINAMICS Integrated can be configured in offline or online mode:
   - Performing an offline configuration (Page 7406)
     For offline configuration, all of the components and their article numbers must be known.
   - Performing an online configuration (Page 7423)
     During online configuration, you can load all of the information from the connected DRIVE-CLiQ components to your user project.
2. Test the configured drive using the drive control panel (Page 7446)
3. Create an axis with the axis wizard (Page 7448)
4. Test a configured axis with the axis control panel (Page 7456)
5. Set up addresses and telegrams (Page 7458)
6. Link an additional encoder (optional) (Page 7461)
7. Configure drive-related I/Os (with symbolic assignment) (Page 7469)
8. Configure technology objects and I/O variables (Page 7474)
9. Optimize the drive and controller (Page 7483)

When carrying out this procedure, consult the relevant references.

This section also contains additional configuration information, e.g. for vector drives, Safety Integrated, etc.

Important functions for the project handling and for the commissioning

Below you will find an overview of the most important functions for project handling and for commissioning with the associated icons.

Save project and compile changes

The entire project is saved and the project data (e.g. programs) compiled into an executable code.

Connect to selected target devices

The online connection is established to the selected target devices. Under "Target system" > "Select target devices", you can set which target devices are to go online.

Download project to target system

The programs are loaded to the SIMOTION device as well as the configuration for the SINAMICS Integrated.

Download CPU / drive unit to target device

The configuration is only loaded to the device selected in the project tree which means that the function needs to be performed separately for each D410-2 and SINAMICS Integrated.

Load CPU / drive unit to PG

The unit's configuration is only loaded to the PG selected in the project tree which means that the function needs to be performed separately for each D410-2 and SINAMICS Integrated.

Copy RAM to ROM

Copying from RAM to ROM is only performed for the device selected in the project tree which means that the function needs to be performed separately for each D410-2 and SINAMICS Integrated.

Note

Tips for going online:

In online operation, SIMOTION SCOUT attempts to conduct online operation with all hardware components contained in the project. This means that the time needed for going online increases.

We recommend that you make settings for SIMOTION SCOUT so that online operation is made only with those components currently needed. The setting can be found at "Target system" > "Select target devices ..." in the menu. The selection and deselection of the devices in the online state can be made via the "Connect target device" context menu on the device.

This procedure is also advantageous when the configuration of the drive unit is completed. Without going completely offline, the connection can be simply deselected via the context menu on the drive unit.

Performing an offline configuration

Overview

Introduction

In the case of offline configuration, the project is created while not all of the hardware components (in particular drives) are available. In this way, a SIMOTION project in the office environment can be created up to a point where a basic project specification including a program exists. At a later time, the finished project can be loaded to the SIMOTION D410-2 and tested together with the drive.

Requirements

- For offline configuration, all of the components and their article numbers must be known.
- You have created a project in SIMOTION SCOUT and inserted a SIMOTION D410-2 in the project.
- You have configured the communication between the SIMOTION D410-2 and the PG/PC, see Section Creating a project and configuring the communication (Page 7367).

Procedure

The offline configuration involves the following steps:

- Accessing the drive wizard (Page 7407)
- Configuring components (Page 7408)
- Downloading the project to SIMOTION D410-2, by means of the following options:
  - Downloading to the target system (Page 7419)
  - Downloading to the CF card (Page 7420)
  - Downloading incl. sources and additional data (Page 7421)
  - Archiving to the CF card (Page 7422)

Note

During offline configuration, you can configure Terminal Modules such as the TM15, for example.

Accessing the drive wizard

Integrated drive

SIMOTION D410-2 features an integrated SINAMICS S120 drive (control unit) which is automatically included when you insert the SIMOTION D410-2 in the project navigator. The integrated drive must be operated in isochronous mode using PROFIdrive-compliant telegram types.

The drive wizard for the integrated STARTER in SIMOTION SCOUT is available for the purpose of configuring the integrated drive and its associated modules (SINAMICS S120 Power Module, for example).

Note

Take note of all the necessary safety instructions and rules governing connections, which can be found in the latest SINAMICS S120 documentation on the SIMOTION SCOUT DVD.

Procedure

Select the "SINAMICS_Integrated" > "Configure drive unit" drive element in the project navigator to open the drive wizard.

You can configure the following components:

- Power unit (e.g. SINAMICS S120 Power Module)
- Motor
- Encoder

Configuring components

Procedure

Note

An overview of permissible configurations, quantity structures and DRIVE-CLiQ topologies can be found in Section Quantity structures (Page 7507) and in the *SINAMICS S120* Commissioning Manual.

Failure to comply with the rules listed in this manual will result in errors that are not output until the download is performed, rather than at the configuration stage.

1. Enter a drive name in the "Drive Properties" dialog box and select the drive type (servo or vector).

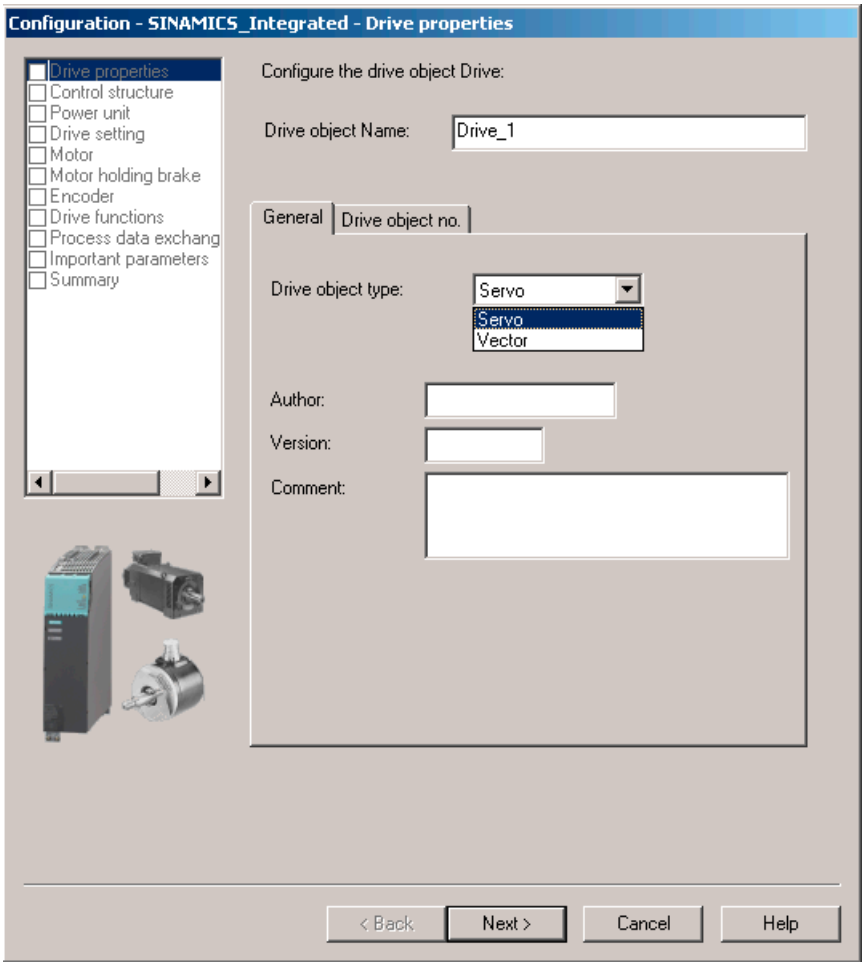


Figure 10-169 Drive properties

2. In the "Control Structure" dialog, you can select the function modules and the control type.

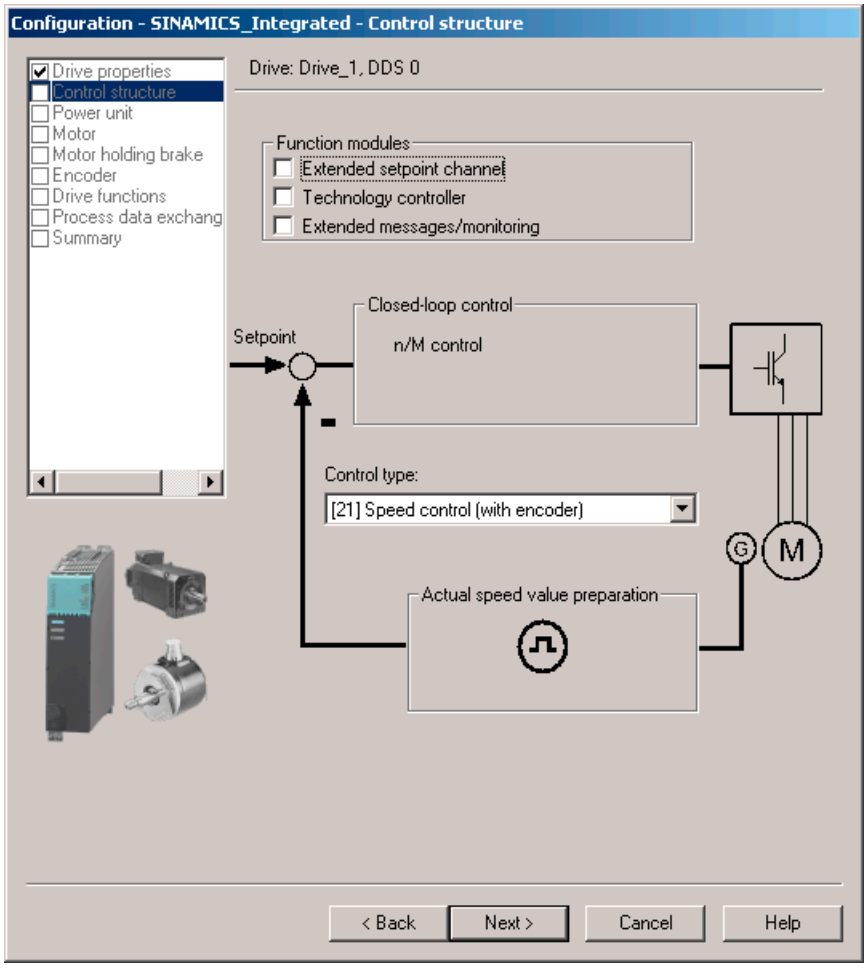


Figure 10-170 Control structure

3. In the "Power Unit" dialog, use the article number to select your power unit from the list.

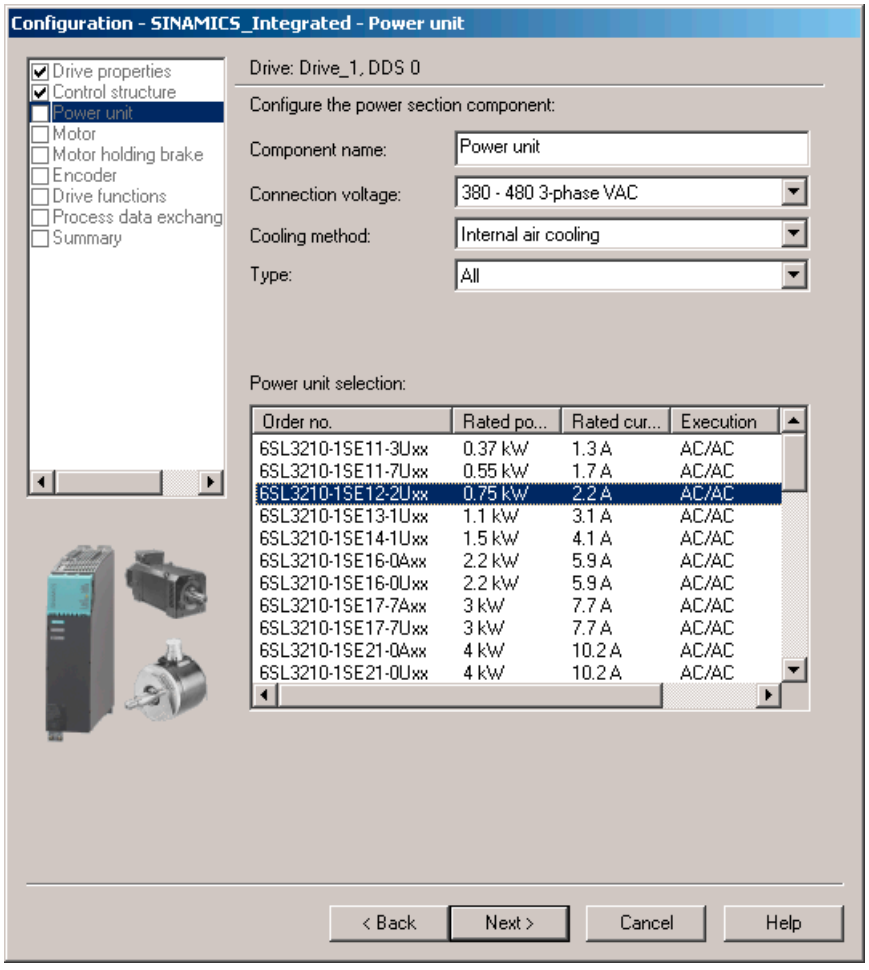


Figure 10-171 Selecting a power unit

4. Select in the "Power Unit Supplementary Data" dialog the component added to the power unit.
   The selection of the component depends on the construction type.
   - SIMOTION D410-2 DP or D410-2 DP/PN: The SIMOTION D410-2 is snapped directly onto the blocksize format Power Module.
   - CUA31 or CUA32: The CUA3x is snapped directly onto the blocksize format Power Module. The SIMOTION D410-2 is mounted separately on a mounting plate. The SIMOTION D410-2 is connected to the CUA3x using a DRIVE-CLiQ cable.

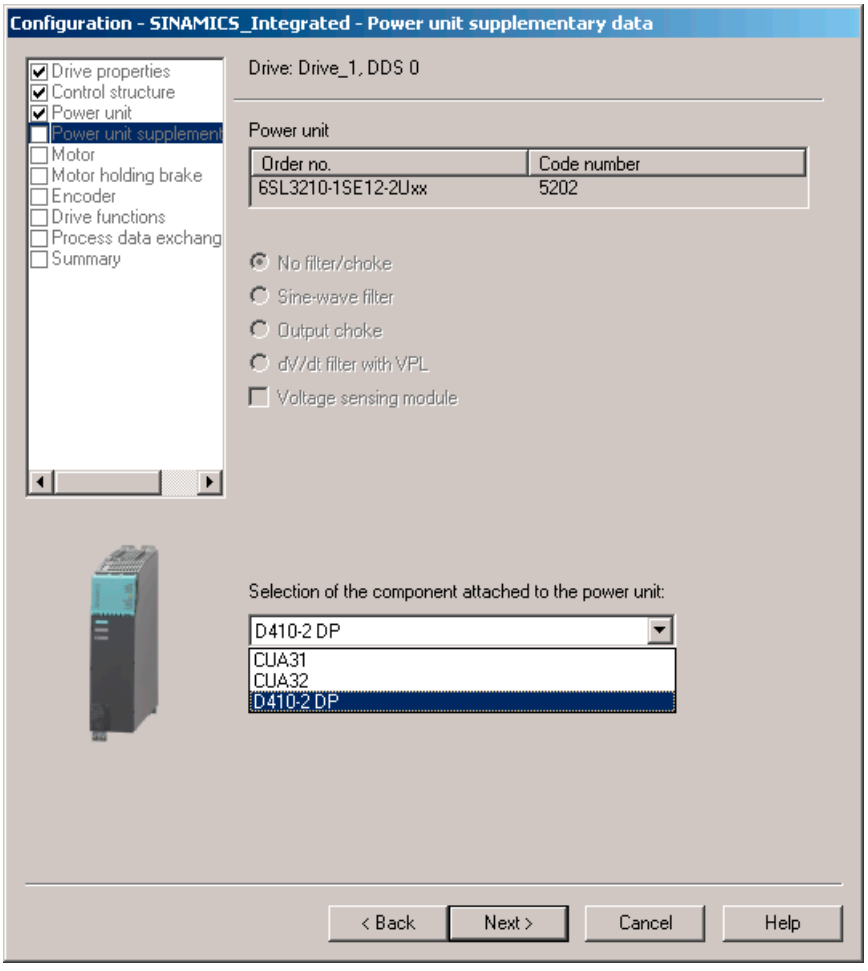


Figure 10-172 Selecting the mounting type

Note

If SIMOTION D410-2 is mounted separately (Power Module in blocksize format connected to SIMOTION D410-2 via CUA31/32), use of the Safety Integrated Extended and Advanced Functions via the onboard terminals (F-DI, F-DO) is not possible.

5. In the next dialog, select the motor and, where applicable, the motor type:
   - Either by selecting a standard motor from the list
   - Or by entering the motor data
   - Or by automatically identifying the motor (motor with DRIVE-CLiQ interface)

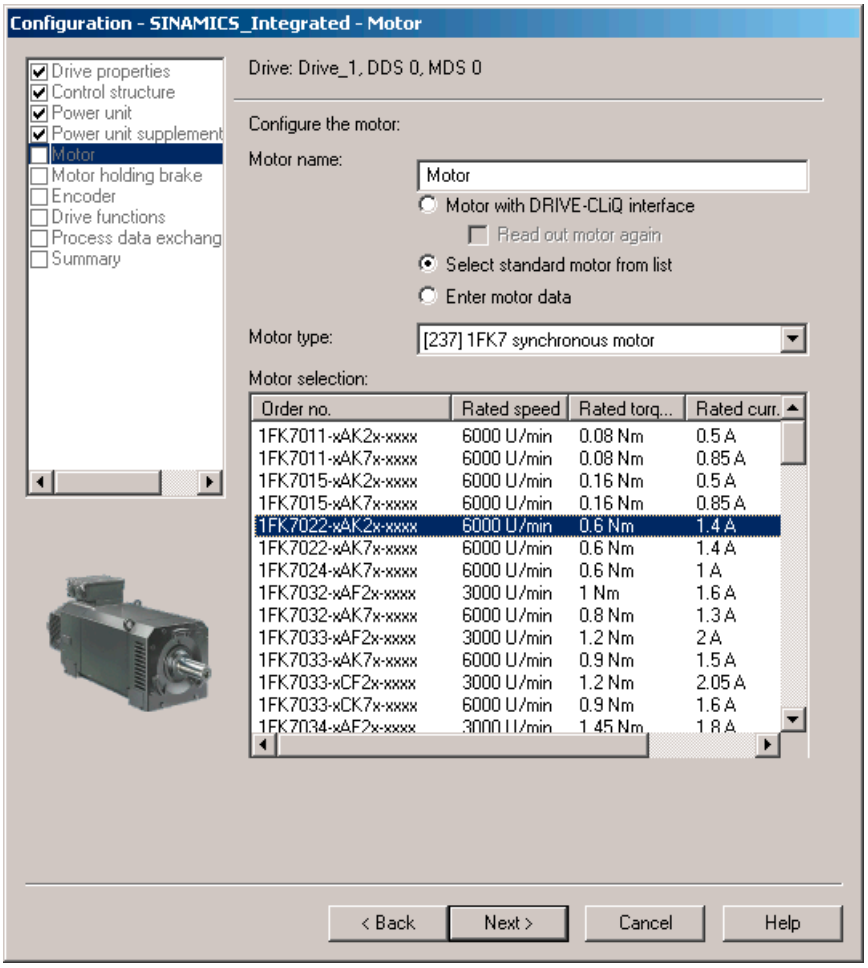


Figure 10-173 Selecting a motor

Note

Motors with DRIVE-CLiQ interface have an integrated encoder evaluation that is connected to the Power Module via a fully digital communication interface (DRIVE-CLiQ).

In this way, motor encoder and temperature signals as well as electronic rating plate data, such as unique article numbers, rated data (voltage, current, torque) can be transferred directly to the Control Unit.

6. Select a motor holding brake (if installed).

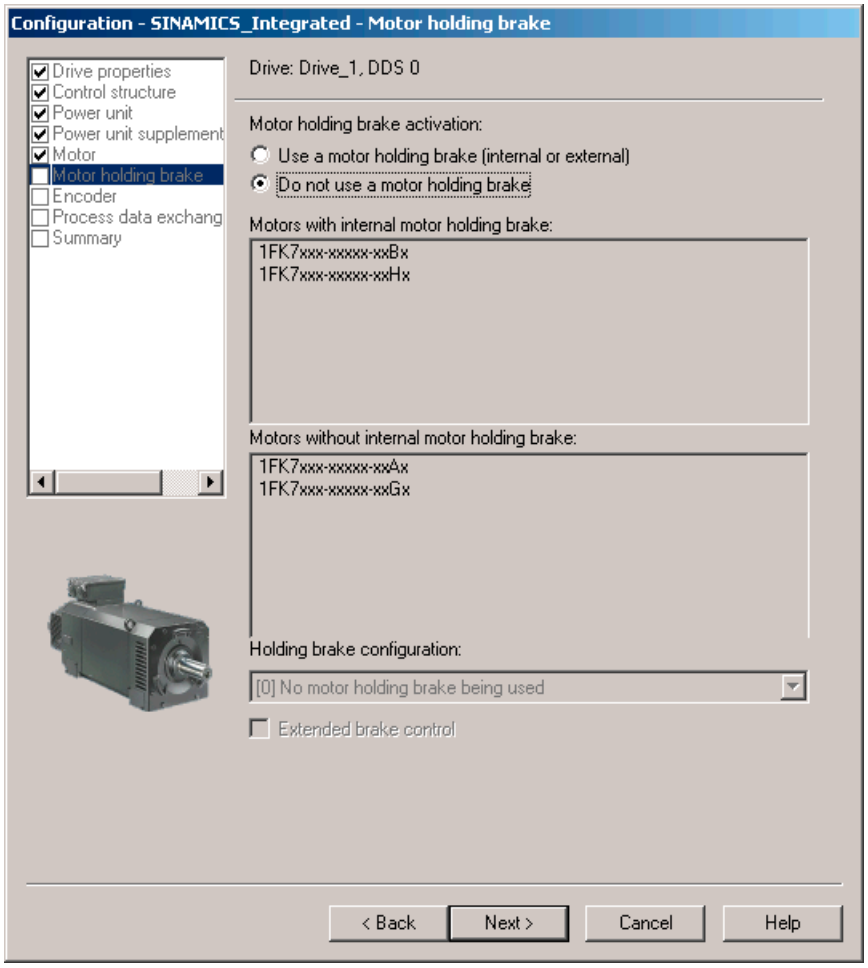


Figure 10-174 Selecting a motor holding brake

7. If you are using a motor that is not equipped with a DRIVE-CLiQ interface, select the encoder order number in the "Encoder Selection via Motor Order Number" dialog box.

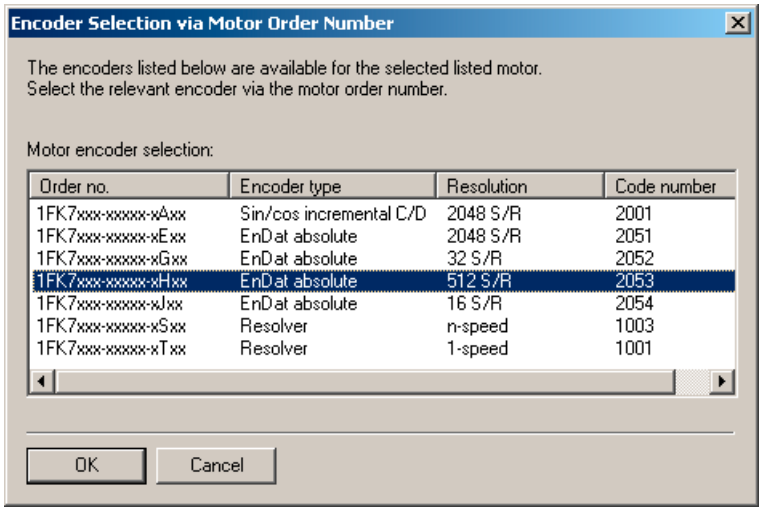


Figure 10-175 Selecting a motor encoder (1)

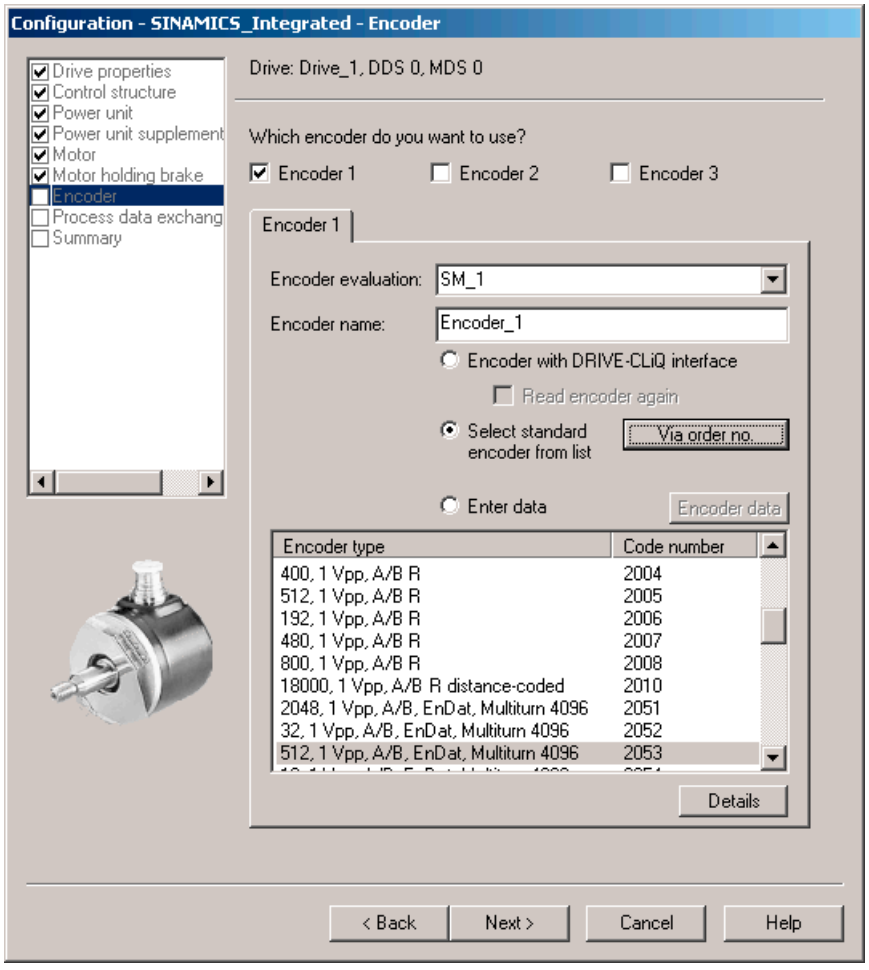


Figure 10-176 Selecting a motor encoder (2)

Note

If required, you can configure a second or third encoder in the "Encoder" dialog box. You can transmit a maximum of two encoder values to SIMOTION via the axis telegram. In the case of motors with a DRIVE-CLiQ interface, the motor encoder is identified automatically. It is not necessary to enter encoder data in such cases (the box for selecting Encoder 1 is grayed out and therefore inactive).

8. The communication for the control of the SINAMICS drive is configured in the following dialog box. It is recommended that these settings be made automatically by the engineering system.

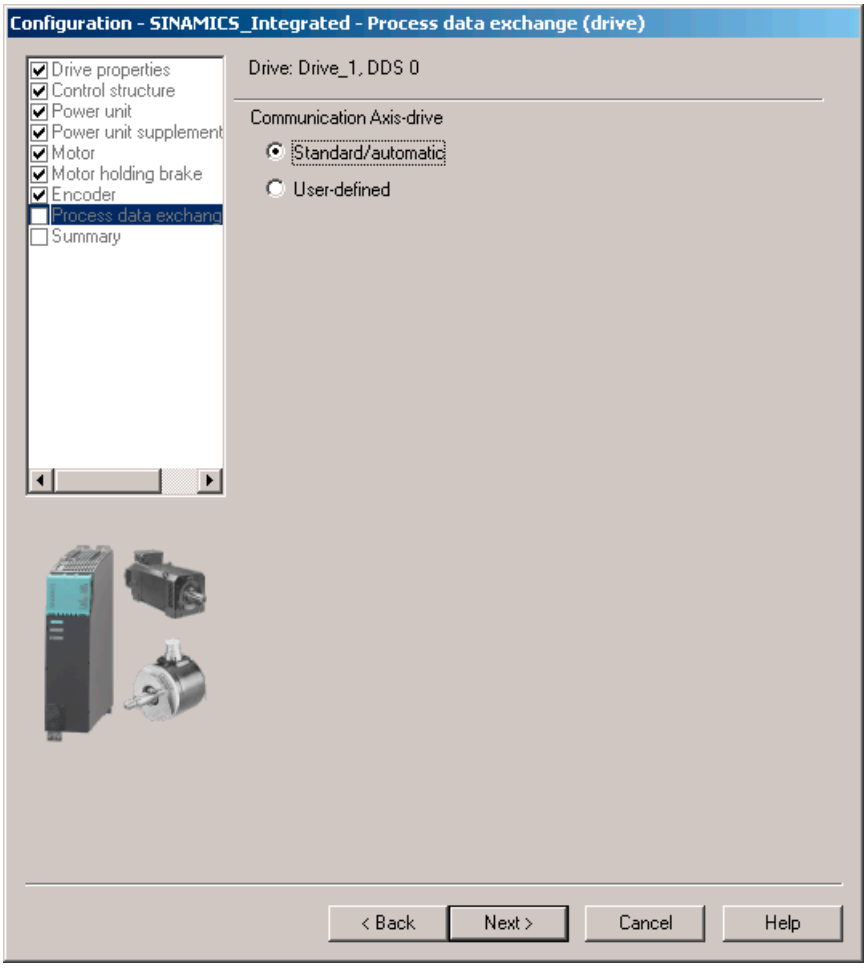


Figure 10-177 Configuring the process data exchange (standard)

You can also make the settings manually for the process data exchange by selecting "User-defined".

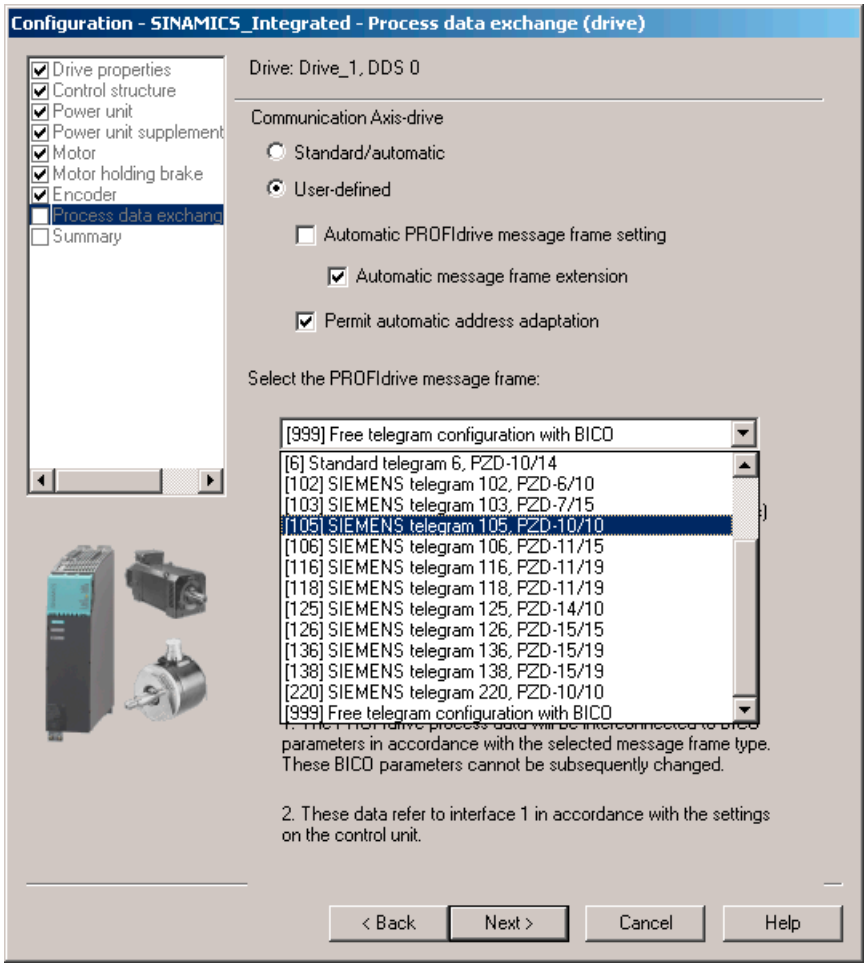


Figure 10-178 Configuring the process data exchange (user-defined)

Information about the manual setting options can be found in the online help and in the manuals for the SINAMICS S120 drive system.

9. After you have configured all of the settings in the drive wizard, the "Summary" dialog box displays a list of all settings. You can now choose to activate your settings by clicking "Finish", or to return to the component configuration for further editing.

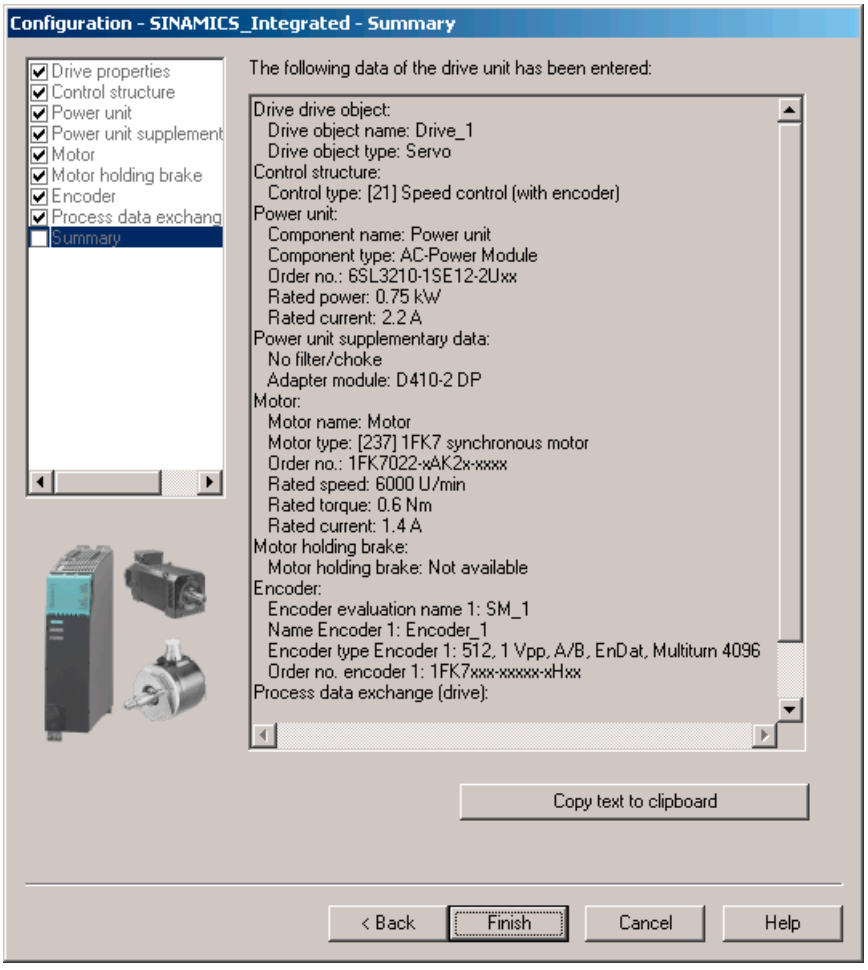


Figure 10-179 Finishing the drive

The configured drive is displayed in the project navigator. An overview of your configured SINAMICS components is available in the "SINAMICS_Integrated" > "Topology" dialog.

Additional references

If you configure the drive telegrams manually, you can find detailed information on the respective telegram types in the following:

- *Motion Control, TO Axis Electric/Hydraulic, External Encoder* Function Manual
- *SINAMICS S120* Function Manual

Downloading the project to the target system

Procedure

1. Save and compile the project.
2. Go online with the SIMOTION D410-2.

3. To load the project, perform "Download project to target system".
   The data must also be saved on the CompactFlash card to ensure that the project is retained in the event of a power failure. The following options are available:
   - Perform the "Copy RAM to ROM..." function manually on the SIMOTION D410-2 and the drive (SINAMICS Integrated).
   - In the "Download to Target System" dialog box, select the option "After loading, copy RAM to ROM". You can change the default setting for this dialog box in "Options" > "Settings" > "Download".
4. To save the parameter calculations of the drive in the project, perform "Target device" > "Load CPU / drive unit to PG" for the drive.

Result

The drive has been assigned parameters and commissioned. You can now test the drive using the drive control panel.

Note

Online access to SINAMICS Integrated is not possible if **HW Config** is not loaded at the time you initially connect to the target system.

Download the data to **HW Config** in order to enable online access to SINAMICS Integrated.

Note

If you have deselected the "Drives" option under "Tools" > "Settings" > "Download" in SIMOTION SCOUT, you must download the configuration separately to each drive (SINAMICS Integrated).

To do this, select the drive (e.g. SINAMICS Integrated) in the project navigator and perform "Download CPU / drive unit to target device".
To work quickly, we recommend that drives be generally deselected and a download only be performed when required.

Loading a project created offline to the CompactFlash card

Procedure

You can use a card reader to write the entire project to the CF card, even in offline mode. In SIMOTION SCOUT, you can call the "Load to file system" function in the context menu of the SIMOTION device.

1. Save and compile the project.
2. Switch off the SIMOTION D410-2.
3. Remove the CF card and insert it in a card adapter. The card adapter must be connected to a PG/PC.
4. In the SCOUT project, select the SIMOTION D410-2 device that you want to download to the CF card.
5. Click "Load to file system" in the context menu. A dialog box opens.

6. In the "Load to File System" dialog box, select the "Save normally" option and click the "Select target" button.
7. Select the target drive.
8. Confirm your entries with "OK". The data is written to the CF card.
9. Remove the CF card and insert it into the slot on the SIMOTION D410-2.
10. Switch on the SIMOTION D410-2.

Note

With "Load to file system", the project data of the controller (including SINAMICS Integrated) is written directly to the CF card. Separate execution of "Load to file system" for the SINAMICS Integrated is not possible.

Result

The SIMOTION D410-2 ramps up with the loaded project.

Note

The components' firmware is automatically updated, depending on the firmware version on the SINAMICS components and on the CF card. During a firmware update, please take note of the messages and alarms in the SIMOTION SCOUT detail window. A FW update on the SIMOTION D410-2 is indicated by the RDY LED flashing yellow, while on the DRIVE-CLiQ components (TM, SMC, etc.) it is indicated by the RDY LED flashing red/green.

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

Components that require a POWER ON after the firmware update will indicate this through a rapidly flashing RDY-LED. Go offline with SCOUT and switch the 24 V supply off/on (POWER ON) at the respective components for initialization.

Downloading the project incl. sources and additional data

Overview

It is possible to download additional data (e.g. sources) to the target device when saving a project to the CF card or downloading to the SIMOTION D410-2.

These data are required for:

- Online object comparison (e.g. additional properties)
- Various detailed comparisons (e.g. ST source file comparison)
- Synchronization with online objects.

In order to be able to load a project's sources and additional data to the PG, this must be specified in the project under "Options" > "Settings" > "Download" > "Store additional data on the target device". Alternatively, this setting can also be made when the data is loaded to the target device/target system.

You can, for example, perform a project comparison with the sources and additional data stored on the CF card (see example below).

Project comparison (example)

You intend to perform servicing work on a commissioned system and have placed a project on your PG/PC. This project is not consistent with the project on the SIMOTION D410-2 cm the system. In order to analyze the differences, perform an object comparison via "Start object comparison".

You have the following options in terms of re-establishing consistency:

- In the object comparison, it is possible to establish consistency in sources and technology objects on an object-granular basis.
- Consistency can be established for the entire Control Unit by loading from the CF card with "Target system" > "Load" > "Load CPU / drive unit to PG...."

Additional references

Detailed information on loading data to the target device can be found in the *SIMOTION Runtime Basic Functions* Function Manual.

Archiving a project on the CompactFlash card (zip file)

Procedure

In SIMOTION SCOUT, you can save the project as a *.zip file to the CompactFlash card.

Proceed as follows to archive the SIMOTION project on the CompactFlash card:

1. Open SIMOTION SCOUT and select the "Project" > "Archive" menu command.
2. In the "Archive" dialog box, select the SIMOTION project and save it to your drive (PG/PC).
3. Open the project.
4. Go online with the SIMOTION D410-2.
5. In the project navigator, select the SIMOTION D410-2 and select the "Target system" > "Load" > "Save archived project to card..." menu command.
6. In the dialog that is displayed, select the project and click "Open". This saves the project to the CompactFlash card as Project.zip in the directory: USER\SIMOTION\HMI\PRJLOG.

Note

If you want to load the current project from the card, select the "Target system" > "Copy archived project from card to PG/PC..." menu command.

Prerequisite is that you have backed up the project with "Save archive project on card..." each time a change was made.

Additional references

Detailed information on loading data to the target device can be found in the *SIMOTION Runtime Basic Functions* Function Manual.

Performing an online configuration

Overview

Introduction

You can configure the plant in online mode after having completed its wiring. You can use the "Automatic configuration" function to load the SINAMICS components connected via DRIVE-CLiQ to your PG/PC. However, this is only possible for initial commissioning.

Note

Components without DRIVE-CLiQ connection must be edited in offline mode. You may need to edit DRIVE-CLiQ components which were detected in the course of automatic configuration (for example, adding encoder data if using SMC modules).

Prerequisites for online configuration

- You have created a project in SIMOTION SCOUT and inserted a SIMOTION D410-2 in the project.
- You have configured the communication between the SIMOTION D410-2 and the PG/PC.
- Your system has been installed and wired.

Procedure

The online configuration involves the following steps:

- Establishing the online connection (Page 7424)
- Starting automatic configuration (Page 7424)
- Reconfiguring SINAMICS components (Page 7427)
- Downloading a project to the SIMOTION D410-2 (Page 7428)

Establishing the online connection

Procedure when commissioning the drive for the first time

To perform an online configuration, you must establish an online connection to the SIMOTION D410-2. In this case, no connection can yet be established to SINAMICS Integrated. An appropriate message is output. Once the hardware configuration has been loaded to the target device, an online connection to the SINAMICS Integrated is established automatically. Proceed as follows:

1. Save and compile the project.
2. Establish an online connection.
3. Select the SIMOTION D410-2 device in the project navigator.
4. Use the "Download CPU / drive unit to target device" function to download the SIMOTION D410-2 device to the target device. The connection to SINAMICS Integrated is activated automatically.

You can now run automatic configuration on the SINAMICS Integrated. For details, see Section Starting automatic configuration (Page 7424).

Additional references

Further information about establishing an online connection to the programming device/PC can be found in the following documentation:

- *SIMOTION SCOUT* Configuration Manual
- *SIMOTION SCOUT* Online Help
- In the FAQ for *SIMOTION Utilities & Applications*
  SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.
- FAQOnline connection to SIMOTION (https://support.industry.siemens.com/cs/ww/en/view/22016709)

Starting automatic configuration

Requirement

You have established the online connection to SINAMICS Integrated.

Procedure

1. In the project navigator, open the "Automatic Configuration" dialog with the "SINAMICS_Integrated" > "Automatic configuration" menu command.

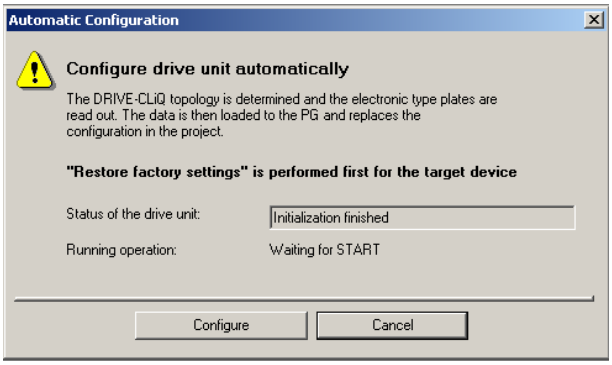


Figure 10-180 Starting automatic configuration

2. Click the "Configure" button.
3. If the drive unit is not in the "First commissioning" state, the factory settings are restored after acknowledging a prompt.
4. The drive object types can now be selected via a further dialog box.

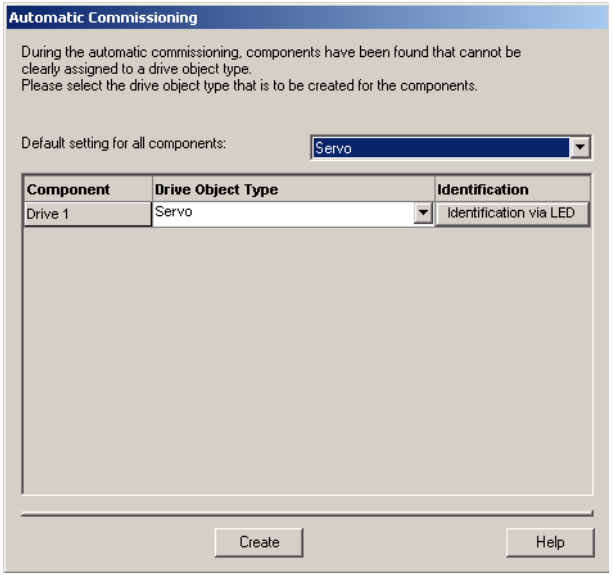


Figure 10-181 Selecting the drive object type

5. Select whether a servo- or vector-type drive object is to be used.

6. Click "Create" to start the automatic configuration.
   The configuration data is uploaded (Load to PG) automatically as soon as automatic configuration is completed.

   Note

   The components' firmware is automatically updated, depending on the firmware version on the SINAMICS components and on the CF card.

   The update procedure can take several minutes and is indicated in the "Automatic Configuration" dialog box by the following message:

   "State of the drive unit: Automatic FW update of DRIVE-CLiQ components".

   A FW update on the SIMOTION D410-2 is indicated by the RDY LED flashing yellow, while on the DRIVE-CLiQ components (TM, SMC, etc.) it is indicated by the RDY LED flashing red/green.

   - FW update running: RDY LED flashes slowly (0.5 Hz)
   - FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

   Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

   Following the automatic configuration, a prompt appears as to whether you want to "Go offline" or "Remain online" with the drive unit.

7. Execute the "Copy RAM to ROM ..." function on the SIMOTION D410-2 and SINAMICS Integrated. This saves the project on the CF card so that it does not need to be reloaded after switching off and on.

Result

The DRIVE-CLiQ components loaded to the user project by means of automatic configuration are displayed in the project navigator.

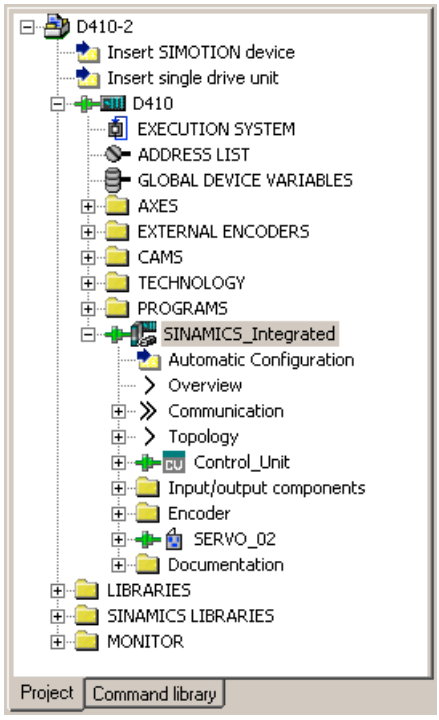


Figure 10-182 Project navigator with the downloaded DRIVE-CLiQ components

You must then

- If required, reconfigure SINAMICS components (e.g. components without a DRIVE-CLiQ interface, such as an encoder connected to the onboard encoder interface).
- Assign the "TO axis" to "Drive".

Reconfiguring SINAMICS components

Requirements

- You have uploaded all connected DRIVE-CLiQ components to the user project.
- You have shut down the connection to the target system (offline mode).

Procedure

Now go ahead and adapt your components to suit the application.

Run through the wizard for all the DRIVE-CLiQ components to be adapted and perform the required reconfigurations.

This procedure corresponds to the description in Section Performing an offline configuration (Page 7406).

The amount of editing work involved depends on the components used. For example, in the case of a motor with a DRIVE-CLiQ interface, the motor and encoder type are identified automatically.

Downloading the project to SIMOTION D410-2

Procedure

After you have performed the reconfigurations, you must download the configuration to the SINAMICS Integrated.

1. Save and compile the project.
2. Go online with the SIMOTION D410-2.
3. To load the project, perform "Download project to target system".
   The data must also be saved on the CompactFlash card to ensure that the project is retained in the event of a power failure. The following options are available:
   - Perform the "Copy RAM to ROM..." function manually on the SIMOTION D410-2 and the drive (SINAMICS Integrated).
   - In the "Download to Target System" dialog box, select the option "After loading, copy RAM to ROM". You can change the default setting for this dialog box in "Options" > "Settings" > "Download".
4. To save the parameter calculations of the drive in the project, perform "Target device" > "Load CPU / drive unit to PG" for the drive.

Result

The drive has been assigned parameters and commissioned. You can now test the drive using the drive control panel.

Note

Online access to SINAMICS Integrated is not possible if **HW Config** is not loaded at the time you initially connect to the target system.

Download the data to **HW Config** in order to enable online access to SINAMICS Integrated. During the "Download to target system" process, SIMOTION SCOUT automatically attempts to establish an online connection to SINAMICS Integrated.

Note

If you have deselected the "Drives" option under "Tools" > "Settings" > "Download" in SIMOTION SCOUT, you must download the configuration separately to each drive (SINAMICS Integrated).

To do this, select the drive (e.g. SINAMICS Integrated) in the project navigator and perform "Download CPU / drive unit to target device".
To work quickly, we recommend that drives be generally deselected and a download only be performed when required.

Additional information on configuring the SINAMICS Integrated

Setting DP Slave properties

Settings in HW Config

Depending on the cycle clock ratios (bus cycle, servo cycle) and the drive used, it may be necessary to adapt the properties of the DP slave (SINAMICS Integrated) on the PROFIBUS Integrated.

Open **HW Config**. Double-clicking the SINAMICS Integrated enables you to display and, if required, change the properties of the DP slave on the "Clock synchronization" tab. Possible changes include:

- Synchronize drive to the equidistant DP cycle
  The SINAMICS Integrated of a SIMOTION D410-2 can only be operated isochronously. For this reason, this option cannot be deactivated.
- Changing the master application cycle (T_{MAPC})
  The master application cycle must always be the same as the servo cycle clock set (setting: in context menu of the D410-2 > "Set system cycle clocks" in the project tree).
  Provided that the DP cycle is not scaled down to the servo cycle, the master application cycle will always be the same as the DP cycle.

Note

The PROFIBUS cycle clock can be scaled down in relation to the servo cycle clock. Down-scaling is only permitted if PROFINET IO with IRT has not been configured.

- Changing the DP cycle (T_{DP})
  Depending on the requirements in terms of the quantity structures and response times, the DP cycle may need to be changed. (See also *SIMOTION Runtime Basic Functions* Function Manual).
  In addition, the minimum DP cycle for vector drives also depends on the speed controller cycle clock, which in turn depends on the used device type. This means that, particularly in the case of vector drives, the DP cycle must be checked and changed if necessary, see Section Using vector drives (Page 7431).

  Note

  After T_{DP} has been changed on the PROFIBUS master, the drive system must be switched on (POWER ON).

- Changing the T_I and T_O times
  A change to T_I/T_O is required in the case of vector drives, for example, where the T_I/T_O time for devices in chassis format depends on the type of device used.

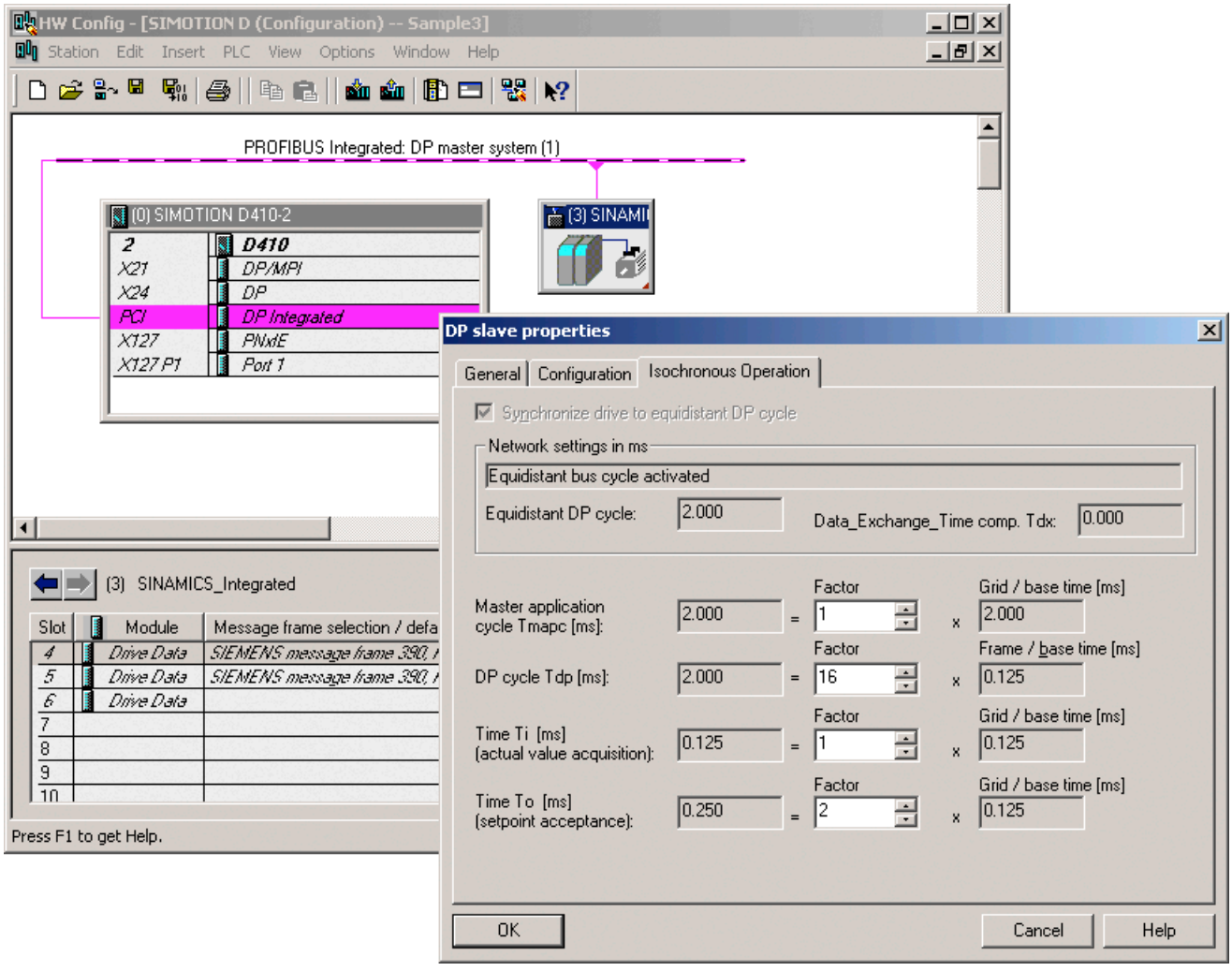


Figure 10-183 HW Config setting

The times are modified by changing the value in the "Factor" field.

SCOUT TIA

In order to parameterize the isochronous cycle clock for the PROFIBUS Integrated in the TIA Portal, proceed as follows:

1. Select the PROFIBUS Integrated in the network view.
2. Select the "General" tab in the Inspector window and click "Equidistance" to display the parameters for the isochronous PROFIBUS. The "Activate isochronous bus cycle" setting is permanently selected and cannot be edited.

The other parameters are set to default settings and can be edited.

Additional references

Additional information can be found

- In the *SINAMICS S120* Function Manual
- *SIMOTION Runtime Basic Functions* Function Manual
- *SIMOTION SCOUT TIA* Configuration Manual

Using vector drives

Changes need to be made in **HW Config** when SINAMICS vector drives are used. This means, for example, that the T_I/T_O time and the minimum DP cycle for drives in chassis format depend on the type of device used.

Therefore, we recommend adopting the following procedure when using a vector drive with the SIMOTION D410-2.

Procedure

1. Open **HW Config**. By "double clicking" the SINAMICS Integrated, you can change the properties of the DP slave in the "Clock synchronization" tab.
2. For $T_I = T_O =$, enter an integer multiple of the current controller cycle. For the current controller cycle clock, use 375 µs for devices in chassis format and 250 µs or 500 µs for the blocksize devices (PM340, PM240-2).
3. For T_{DP}, enter an integer multiple of the speed controller cycle. For a drive on the SINAMICS Integrated, T_{DP} must be >= T_O in all cases.
4. Enter $T_{MAPC} = T_{DP}$ (exception: You are working with cycle reduction; i.e. servo cycle > DP cycle).
5. Use the "Target system" > "Load" > "Load project to target system" menu command to download the parameterization to the SIMOTION D410-2.

6. After the download has successfully completed, you should determine the current and speed controller cycles of the drive from the drive's expert list, because the cycle clocks are set in the SINAMICS drive unit after a project has been downloaded.
   - p0115[0] current controller cycle clock
   - p0115[1] speed controller cycle clock
7. If the current and speed controller cycles from the expert list are different to the cycles used in steps 2 and 3, you will have to repeat the steps using the most up-to-date values for the current and speed controller cycles.

Note

The described procedure is also required for an online configuration with an automatic configuration.

Table 10-115 Vector drive

Example	Settings
Vector drive (chassis format) Current controller cycle = 375 µs Speed controller cycle = 1.5 ms	$T_I = T_O$ = at least 375 µs T_{DP} = 1.5 ms (... or 3 ms, or 6 ms ....) $T_{MAPC} = T_{DP}$ For the SIMOTION D410-2, a minimum of 1 ms is recommended for the servo cycle clock. For this reason, set $T_{MAPC} = T_{DP}$ = 3 ms or higher.
Vector drive (PM340/PM240-2) Current controller cycle clock = 500 µs (default) Speed controller cycle = 2 ms	$T_I = T_O$ = at least 500 µs T_{DP} = 2 ms (... or 4 ms, 8 ms, ...) $T_{MAPC} = T_{DP}$

Blocksize Power Modules

Current controller cycle clock 500 µs (default)

The default current controller sampling time for SIMOTION D410-2 with PM340/PM240-2 is 500 µs. This current controller cycle is set automatically following project download if p0112 = 3 (default). The sampling times in p0115 are adapted automatically by the system following the download; therefore, they may deviate from the offline values. The system also sets p0112 = 0 (expert) after the download.

Current controller cycle clock 250 µs (manual setting required)

If the current controller cycle clock for SIMOTION D410-2 is to be changed from 500 µs (default) to 250 µs, p0112 = 0 (expert) must be set.

p0112 = 0 enables the individual sampling times to be adjusted in p0115. Moreover, with this setting, the system does not change the current controller sampling time automatically.

Note

SIMOTION D410-2 is a single-processor system. CPU utilization is, therefore, dependent on the SIMOTION configuration (PLC and motion control) and the SINAMICS configuration (drive control).
For more performance on the SIMOTION side, we recommend not setting any unnecessary cycle clocks on the SINAMICS side.

400 µs current controller cycle clock

Vector drives in chassis format can also be operated with a current controller sampling time of 400 µs (amongst other settings).

In the SIMOTION context, the following should be considered:

- A current controller sampling time of 400 µs is only possible if control is via a SINAMICS S120 Control Unit, which is not operated isochronously via PROFIBUS/PROFINET on the SIMOTION D410-2.
- If the bus is operated isochronously, only cycle clocks with an integer multiple of 125 µs are possible (i.e. 375 µs or 500 µs instead of 400 µs, for example).
- The PROFIBUS Integrated of a SIMOTION D410-2 is always isochronous! This means that a current controller sampling time of 400 µs is not possible.
- With CU parameter p0092 = 1, the sampling times are preassigned in a way that enables isochronous operation with a control system.

Output cams / measuring inputs with vector drives

In the case of devices in chassis format, the cycle clock ratios (current controller cycle clock, speed controller cycle clock, input/output sampling time, etc.) also depend on the type of device used.

Note the information in Section Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs (Page 7443).

Additional references

Additional information on quantity structures and cycle clock settings can be found in the *SINAMICS S120* Function Manual.

Setting the SIMOTION time of day

Time on SIMOTION (real-time clock)

SIMOTION D410-2 features an integrated real-time clock. All events on a module (alarms, messages, etc.) are "time-stamped" based on the time shown by this real-time clock.

Procedure

Various options are available to set the clock of the SIMOTION D410-2:

- Setting the clock via the engineering system
  - SIMOTION SCOUT: Select the SIMOTION D410-2 in the project tree and then "Target system" > "Set time" in the menu.
  - SIMOTION SCOUT TIA: Call the "Set time" function at "Online & diagnostics" in the TIA Portal in the project tree.
- Setting the clock using the "rtc" system function block
- Synchronization of the local time with an NTP time server (as of V4.5)

The status of the NTP time synchronization can be determined with the **_getStateOfNTPClockSynchronisation** system function.

For further information on the NTP process, see the *SIMOTION Communication* System Manual and the *SIMOTION SCOUT TIA* Configuration Manual.

Synchronizing the SINAMICS clock

SINAMICS system runtime (operating hours counter)

For SINAMICS S120 Control Units and the SINAMICS Integrated on a SIMOTION D410-2, faults and alarms are "time-stamped" on the basis of the system runtime. This means that events are recorded by default on the basis of operating hours rather than a particular time of day or date.

System runtime

The entire system runtime is displayed in CU parameter p2114.

- r2114[0] indicates the system runtime in milliseconds. After reaching 86,400,000 ms (24 hours), the value is reset.
- r2114[1] indicates the system runtime in days.

The counter value is saved when the power is switched off. After the drive unit has been switched on, the counter continues to run with the value stored when the power was last switched off.

As a result, the drive displays the system runtime from 00:00:00 on 01/01/1992 in both the alarm window in SIMOTION SCOUT and the diagnostic buffer for entries.

If faults and alarms need to be "time-stamped" on the basis of a time of day, "Time-stamp operating hours" needs to be changed to "Time-stamp UTC format" as described below.

Requirements

A telegram 39x is required for the time synchronization. If the automatic PROFIdrive telegram setting is selected for the Control Unit, this telegram is used automatically (see Section Accessing the drive wizard (Page 7407), Standard/Automatic setting).

If the telegrams are specified manually, telegram 39x must be set up. See Section Message frame configuration (Page 7458)

So that drive units can be synchronized with the SIMOTION time, they must support telegram 39x and the UTC (coordinated universal time) time format.

For precise time synchronization, the drive unit must also be operated via an isochronous bus on SIMOTION.
The SINAMICS Integrated with SIMOTION D is always connected isochronously.

The following Control Units support time synchronization:

- SINAMICS Integrated of the SIMOTION D410-2
- SINAMICS S120 CU310, CU310-2, CU320, CU320-2 Control Units connected via PROFIBUS or PROFINET
- SINAMICS S110 Control Units CU305, connected via PROFIBUS or PROFINET (precondition: as from SCOUT V4.4)

Procedure

Proceed as follows to convert the SINAMICS clock to UTC format and to synchronize this with the SIMOTION clock:

1. In the project navigator, call the shortcut menu of the SIMOTION D410-2.
2. Select the "Properties" entry in the context menu.
3. Select the "Perform time synchronization with SINAMICS drive units" option in the "Settings" tab of the "D410-2 Properties" dialog box.

   Note

   This setting is activated automatically and applies to all drive units connected to the SIMOTION D410-2. The SINAMICS clock is automatically synchronized with the SIMOTION clock for all drive units with configured telegram 39x.

The first time synchronization is performed after the SIMOTION D Control Unit has reached the RUN operating mode.

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically resynchronized at regular intervals.

The user program can use the _driveStates.allClocksSynchronized on the device to query whether automatic time synchronization is enabled (=YES) or disabled (=NO).

Before the first synchronization, alarms and messages are stored with the time stamp valid in the SINAMICS at this time, all subsequent alarms and messages with the synchronized time.

The first time synchronization after switching on is entered with the status of the operating hours counter and the time (UTC time, synchronized with SIMOTION) in the diagnostic buffer of the drive (e.g. SINAMICS Integrated).

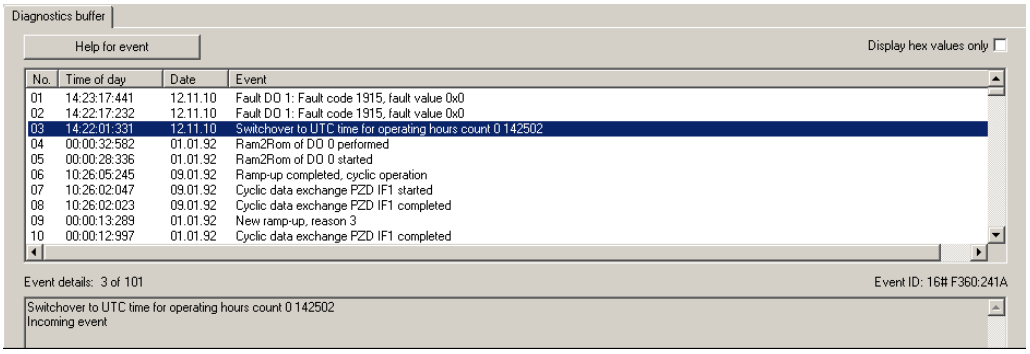


Figure 10-184 Diagnostic buffer entry, time synchronization

Error correction

If problems occur with time synchronization, this may be due to deactivated symbolic assignment to incorrect configuration information (Fast IO configuration).

For more information, see Section Message frame configuration (Page 7458).

Compensation of runtime deviations

To compensate for deviations between the SIMOTION and SINAMICS clocks, the time of day is automatically synchronized at regular intervals.

The following behavior must be taken into consideration when setting the SIMOTION time:

- "Time/date to be set" is after "Time/date on SINAMICS": Time and date are corrected on the SINAMICS.
- "Time/date to be set" is before "Time/date on SINAMICS": The SINAMICS clock must be stopped until the SINAMICS "Time/date" has caught up with "Time/date to be set".

Adopting this procedure ensures that the sequence of SINAMICS diagnostic buffer entries remains the same, even when the runtime differences are aligned.

The SINAMICS clock operates with a resolution of 1 ms. A synchronization accuracy of 1 ms can be achieved for all bus cycle clocks that can be divided exactly by 1 ms (e.g. 1 ms, 2 ms, 3 ms, etc.).

Due to system considerations, a slightly lower synchronization accuracy is achieved for all bus cycle clocks that cannot be divided exactly by 1 ms (e.g. 1.25 ms).

If the drive unit does not have an isochronous connection on SIMOTION, this can result in runtime differences of milliseconds (depending on the set cycle clock ratios).

Resetting the time

Requirement:

- SIMOTION D410-2: As of SIMOTION V4.3
- SINAMICS S120: As of SINAMICS V4.5
- SINAMICS S110: Not available

A threshold value is defined via parameter cu.p3109 and is effective as follows:

- In the case of negative time jumps less than the threshold value (p3109), the time is halted (for details, see "Compensation of runtime deviations")
- In the case of negative time jumps greater than the threshold value (p3109), the time is reset.
  Default setting: cu.p3109 = 100 ms
  This means that in the case of negative time jumps greater than 100 ms, the time is reset. The default value is configured so that normal runtime deviations (drift of the quartzes) are below the threshold value.
  If the SIMOTION clock is reset by more than 100 ms, this is interpreted as a "specific reset of the time" and the time of the drives is also immediately reset.

If the real-time clock is reset by more than 60 seconds, a diagnostic buffer entry is also made in the drive:

Time correction (reset) by **<correction value> seconds**

After a resynchronization (negative time jump greater than the threshold value)

- cu.r3107[0..1] the UTC time after synchronization
- cu.r3107[2..3] the UTC time before synchronization

is displayed in the parameter, whereby [0] and [2] are milliseconds and [1] and [3] are days.

Note

The diagnostic buffer entries are not converted to the new time when the time is changed.

Backing up / restoring / deleting SINAMICS NVRAM data

Requirement

SIMOTION D410-2 as of SIMOTION V4.3

SINAMICS S120 CU310-2/CU320-2 as of SINAMICS V4.5

For other supported SINAMICS Control Units, refer to the SINAMICS manuals.

Saving the NVRAM data

The backup of the SINAMICS NVRAM data is performed by setting parameter p7775 to 1. The following applies:

- Parameter p7775 can also be set to 1 during pulse enable.
- A warm restart is not required.

Note

The consistent backup of the NVRAM data must be ensured by the application.

Make sure that there is no change of the NVRAM contents during the data backup. The greatest possible consistency is achieved when

- Backup is performed during pulse disable or
- The drives are run down to 0 Hz.

The data for the S120 Control Units and the SINAMICS Integrated is stored in the backup file "PMEMORY.ACX" in the folder " ... \USER\SINAMICS\NVRAM" on the CF card.

For the S120 Control Units, the data is stored on the CF card of the S120 Control Unit.

Similar to the non-volatile SIMOTION data, an already existing PMEMORY.ACX file is first renamed as PMEMORY.BAK and then the PMEMORY.ACX file is generated when you save the SINAMICS data.

If an error occurs during backup, the BAK file can be used to restore the data.

Parameter p7775 is set to 0 at the end of the backup.

Restoring the NVRAM data

The SINAMICS NVRAM data can only be restored when none of the connected drives has a pulse enable.

When restoring, the PMEMORY.ACX file is accessed first. If this is available and is error-free, it is loaded. If no PMEMORY.ACX file is available or is corrupt, the PMEMORY.BAK file is loaded (if available and error-free).

Restoration of the NVRAM data can be performed manually and automatically.

Automatic restoration during module replacement

SINAMICS detects whether the Control Unit has been replaced on the basis of the serial number. In this case, the NVRAM of the used Control Unit is deleted first after POWER ON.

The following are not deleted:

- CU operating hours counter
- CU temperature and
- Safety logbook
- Crash diagnostic data

If there is an error-free PMEMORY.ACX or PMEMORY.BAK file on the CF card, the data is then loaded to the NVRAM.

If no error-free backup file is available, ramp-up is as for SINAMICS < V4.5. A fault is not issued and any existing "corrupt" backup files are not deleted.

Manual restoration

The manual restoration of the NVRAM data is performed by setting parameter p7775 to 2.

If the backup file (ACX or BAK) is error-free is, a warm restart is performed. The contents of the NVRAM are deleted first and then the data is loaded from the backup file to the NVRAM.

The following are not loaded:

- CU operating hours counter
- CU temperature
- Safety logbook
- Crash diagnostic data

Parameter p7775 is set to 0 at the end of the backup.

If the backup files are corrupt or no backup exists, the job is acknowledged with an error value in parameter p7775.

Deleting the NVRAM data

The SINAMICS NVRAM data can only be deleted when none of the connected drives has a pulse enable.

The deletion of the NVRAM data is performed by setting parameter p7775 to 3.

A warm restart is then performed automatically.

During the subsequent ramp-up, all NVRAM data on the CU is first deleted.

The following are not deleted:

- CU operating hours counter
- CU temperature and
- Safety logbook
- Crash diagnostic data

At the end of the deletion, the initialization data of the applications is in the NVRAM, similar to after a device automatic commissioning (with the exception of the above-mentioned data).

Parameter p7775 is set to 0 at the end of the deletion.

Explanations for the parameter p7775

The following jobs can be set with the parameter p7775:

Value of the parameter p7775	Job
1	Backup NVRAM data (on CF card)
2	Restore NVRAM data (from CF card)
3	Delete NVRAM data on the Control Unit

The acknowledgement of a parameter job is delayed.

- If the job cannot be executed, it is acknowledged negatively.
- If the job can be executed, it is acknowledged with the value 255.

Table 10-116 Acknowledgments of the parameter jobs

Acknowl-edgement	Description
17	Job cannot be executed because of the operating state
20	Illegal value
107	Write access is not permitted (cause: At least one DO has a pulse enable) Because the SINAMICS must perform a warm restart when restoring and deleting the data, none of the connected drives may have a pulse enable.
132	Parameter change blocked (see p0300, p0400, p0922, p7760, macro being executed)
204	Write access not permitted
255	"OK": Job being executed

If an error is detected during the execution of a parameter job, the error cause is signaled via the parameter itself (no message via the fault buffer).

Table 10-117 Causes of error for parameter jobs

Error	Cause
10	Error occurred while deleting
11	No backup possible: Memory card is not inserted
12	No backup possible: Memory card is full
13	Backup could not be completed (e.g. memory card removed during the backup)
14	Restoration not possible: Memory card is not inserted
15	Restoration not possible: Checksum of the NVRAM data backup file is faulty
16	Restoration not possible: No backup available

Note

The memory card must always remain inserted for the SIMOTION D410-2. For information which SINAMICS Control Units permit the removal of the memory card, refer to the SINAMICS manuals.

Parameter p7775 is only reset automatically to 0 after the successful completion of a job.

Know-how and write protection

Parameter p7775 is not part of the know-how protection. This allows the parameter to be read and written independent of the know-how protection.

Parameter p7775 is part of the write protection. This means the parameter can only be written by the controller that configured a PZD cyclic communication with the SINAMICS when write protection is activated (p7761 = 1).

All other write jobs are acknowledged negatively, e.g.

- SIMOTION SCOUT / STARTER
- For SINAMICS BOP, IOP, AOP Control Units
- Other masters that only communicate acyclically with the SINAMICS

SINAMICS diagnostic buffer

Requirements

The SINAMICS Integrated diagnostic buffer can be displayed in SIMOTION SCOUT.

Procedure

Select the SINAMICS Integrated in the project tree, and then "Target system" > "Device diagnostics" in the menu.

In addition, the SINAMICS diagnostic buffer entries are also displayed in the SIMOTION D410-2 device diagnostics. All SIMOTION D410-2 diagnostic buffer entries are displayed first, and then all diagnostic buffer entries of the the SINAMICS Integrated. The start of the SINAMICS Integrated diagnostic buffer entries is identified by the following entry:

>>>>>> Start of SINAMICS Integrated diagnostic buffer, station address = x <<<<<<

You also have the option of viewing the SIMOTION D410-2 and SINAMICS Integrated diagnostic buffers via SIMOTION IT web server.

Acyclic communication with the drive

Overview

PROFIdrive drive units are supplied with control signals and setpoints by the controller and return status signals and actual values. These signals are normally transferred cyclically (i.e. continuously) between the controller and the drive.

For SINAMICS S110/S120, configure the axis message frames for data exchange (refer to Performing an offline configuration (Page 7406)).

In addition to cyclic data exchange, PROFIdrive drive units feature an acyclic communication channel. In particular, this is used for reading and writing drive parameters (e.g. error codes, warnings, controller parameters, motor data, etc.).

As a result, data can be transferred on an acyclic as opposed to a cyclic basis when required. Acyclic reading and writing of parameters for PROFIdrive drives is based on the DP V1 services "read data set" and "write data set".

The acyclic DP V1 services are transferred while cyclic communication is taking place via PROFIBUS or PROFINET. The PROFIdrive profile specifies precisely how these basic mechanisms are used for read/write access to parameters of a PROFIdrive-compliant drive.

The PROFIdrive standard states that "pipelining" of jobs on PROFIdrive drives is not supported.

- Only one "write/read data record" can be performed at any one time on a drive unit (e.g. SINAMICS S120 control unit or the SINAMICS Integrated of a SIMOTION D).
- However, if several PROFIdrive drive units are connected to a controller, a job can be processed for each of these drive units at the same time. In this case, the maximum total number of jobs will depend on the control system. (for SIMOTION, this is a maximum of 8 jobs at a time).

In the case of acyclic data exchange with SINAMICS drives, this means you will have to coordinate the write/read jobs with one another (buffer management). An interlock must be set in order to prevent the application or different parts of the application from sending overlapping jobs to the same PROFIdrive drive unit.

Additional references

Additional information on how to use DP V1 services can be found in the *SIMOTION Communication* System Manual.

SIMOTION Utilities & Applications also has a DP V1 library with functions that are capable of undertaking coordination tasks commonly associated with acyclic communication. The library not only coordinates access to the system functions (_ReadRecord/_WriteRecord/ _readDriveParameter/_writeDriveParameter/ etc.), but also expands the range of functions for frequently required tasks, e.g. the reading of faults and alarms from the drive unit.

SIMOTION Utilities & Applications is provided as part of the SIMOTION SCOUT scope of delivery.

The functions available in the DP V1 library include:

- Buffer management (coordination of a number of parallel DP V1 services)
- StartUp (function for coordinating startup of the SINAMICS drive with SIMOTION)
- TimeSync (applicative time synchronization: Transfer of SIMOTION time of day to the SINAMICS drives)
- SetActIn (activating/deactivating objects in SIMOTION and SINAMICS)
- RwnPar (reading and writing of drive parameters)
- GetFault (reading errors and warnings from the drive)

Control properties and performance features

With a few exceptions, the integrated drive control of the SIMOTION D410-2 has the same control properties and performance features as the SINAMICS S120 CU310-2 control unit.

However, the following points must be particularly observed:

- The SINAMICS Integrated does not feature a basic positioner (EPOS). EPOS functionality is provided by SIMOTION technology functions.
- It is not possible to connect a BOP20 Basic Operator Panel to the SIMOTION D410-2. The following alternative options are available:
  - Use of SIMATIC HMI devices (e.g. TP177B, configurable with WinCC flexible)
  - Use of the SIMOTION IT web server.
    You can use a Web browser to access the Standard diagnostics pages of the SIMOTION D410-2 (diagnostic and alarm buffer, watch table, SIMOTION variables and read/write drive parameters, access protection, trace function, ...).
    You also have the option of creating your own Web pages, for example, to visualize machine states and to permit service functions. The SIMOTION D410-2 web pages can be accessed, for example, using a PC or PDA via Ethernet. Wireless access is also possible in conjunction with WLAN.
- Not all SINAMICS function modules are supported (e.g. the function modules free function blocks, spindle diagnostics, and CAN are not supported)
- The SINAMICS Integrated of the SIMOTION D410-2 does not support a reset via parameter p0972. A reset via p0972, is only supported by SINAMICS Control Units.

Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs

If current controller cycle clocks <> 125 µs are used, the parameter calculations of the drive must be transferred to the PG, in particular when using cam outputs on the TM15 / TM17 High Feature or for global measuring inputs, so as to regenerate the Fast IO configuration.

A change of the current controller cycle clock will have effects on the sampling times of the I/Os on the drive side (e.g. TM15/TM17 High Feature, p4099 sampling time of I/Os). For the cam outputs and the measuring input inputs (only for global measuring inputs) to function correctly, the sampling times must be known to the engineering system.

Sampling times <> 125 µs occur in the following cases:

- For servo drives with manual change of the current controller sampling time (drive parameters p0112 and p0115[0])
- For vector drives.

Table 10-118 Influence of the current controller cycle clock on the dead time compensation

	Current controller cycle clock has no effect on the function	Current controller cycle clock has an effect on the function
Cam outputs	• SIMOTION D • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	TM15 / TM17 High Feature
Measuring input inputs (global measuring inputs)	• D4x5-2 (terminal X142) • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	• TM15 / TM17 High Feature • SIMOTION D (except D4x5-2, terminal X142) • SINAMICS S110/S120 Control Units
Measuring input inputs (local measuring inputs)	-	-

In order that the changed cycle clock ratios are taken into account by the engineering system, proceed as follows:

1. Go online and perform a project download. The SINAMICS performs parameter calculations once.
2. Perform an upload to the PG ("Target system" > "Load" > "Load CPU / drive unit to PG").
3. This transfers the parameter calculations of the drive to the PG. The cycle clock ratios are then known in the engineering system.
4. Go offline.
5. Generate the configuration information (Fast IO configuration) again. To do this, select the SIMOTION CPU in the project tree and right-click to open the context menu "Fast IO" > "Create new configuration".
6. Execute "Project" > "Save and compile all".
7. Go online and download the project to the target system.
8. Save the data on the CF card.

SIMOTION SCOUT uses the described procedure to calculate internal system data that is required for outputting/detecting signals with a high level of position accuracy.

Note

If the cycle clock ratios are not set correctly, an appropriate message is output in the diagnostic buffer.

Licenses for the SINAMICS Integrated

The following SINAMICS functions are licensed via SIMOTION:

- SINAMICS Safety Integrated Extended Functions for SIMOTION D
- SINAMICS Safety Integrated Advanced Functions for SIMOTION D (as of V5.2)
- SINAMICS high output frequency for SIMOTION D

SINAMICS Integrated

SINAMICS Safety Integrated Extended Functions for SIMOTION D

Highly effective protection of personnel and machinery can be implemented with SIMOTION D thanks to the integrated safety functions of SINAMICS S120.

Whereas the Safety Integrated Basic Functions are license-free, the Safety Integrated Extended Functions and Advanced Functions require a license for the SINAMICS Integrated.

- SINAMICS Safety Integrated Extended Functions for SIMOTION D or
- SINAMICS Safety Integrated Advanced Functions for SIMOTION D

The Safety Integrated Extended and Advanced Functions are licensed for the SINAMICS Integrated of SIMOTION D410-2 Control Unit with the licenses. The Safety Integrated Advanced Functions include the Safety Integrated Extended Functions.

SINAMICS high output frequency for SIMOTION D

Because of legal specifications, the output frequency for converter systems is limited to 550 Hz per default.

- SINAMICS S120 Control Units as of firmware V4.7 HF7
- SIMOTION D Control Units as of firmware V4.4 HF6

The **SINAMICS high output frequency for SIMOTION D** license allows users to cancel the output frequency limit if, as a result of the application, output frequencies of more than 550 Hz are required for drives. The license cancels the limitation **for the SINAMICS Integrated** of SIMOTION D410-2 Control Unit.

Note

The display of the Runtime licenses in the SCOUT Licensing dialog is updated after the power-up of the SINAMICS Integrated.

SINAMICS Control Units

SINAMICS Control Units connected via PROFINET or PROFIBUS (e.g. CU310-2, CU320-2) are licensed via the SINAMICS memory card. Licensing via SIMOTION is not possible.

Further information

- Changes to the EC Dual Use regulation and how it impacts SIMOTION
  https://support.industry.siemens.com/cs/ww/en/view/104020777
- SINAMICS S120/S150: Release for delivery, license for high output frequency
  https://support.industry.siemens.com/cs/en/en/view/104020669

Support of motors with PT1000 temperature sensor

Because of the discontinuation of the KTY84-130 temperature sensor, SIMOTICS motors will be equipped with PT1000 temperature sensor in the future.

- Motors with DRIVE-CLiQ interface remain compatible and can be operated on any SW version.
- Motors without DRIVE-CLiQ interface require SIMOTION SCOUT as of V4.5 for the configuration of the PT1000 temperature sensor.

Testing the configured drive using the drive control panel

You can test a configured drive using the drive control panel where you can define a speed value by setting a scaling factor. The drive control panel should only be used for commissioning.

Requirements

- The project has been downloaded to the target system.
- SIMOTION SCOUT is in online mode.
- The drive is not being used by a current project in RUN mode.

⚠ WARNING
Danger to life from unexpected movement of driven machine parts
Make sure this presents no hazard to personnel or property.

Procedure

1. Change to the configured drive in the Project Navigator and open the drive control panel by selecting "Commissioning" > "Control panel". The drive control panel opens in the detail view.

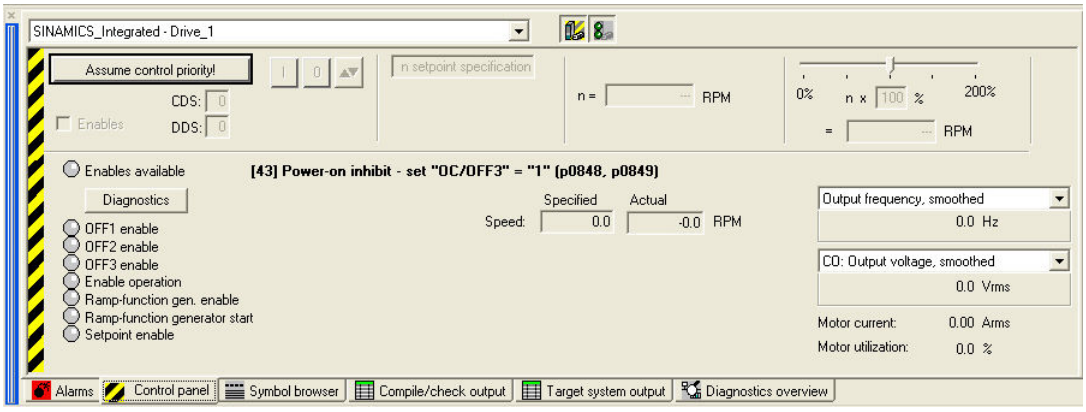


Figure 10-185 Drive control panel

2. To display the control area and axis diagnostics, click the "Show/hide control area" and "Show/hide diagnostics area" buttons.
3. Click "Assume control priority". The "Assume Control Priority" dialog box is opened.

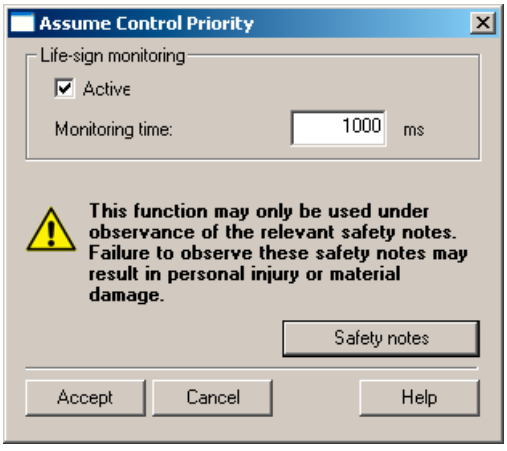


Figure 10-186 Assuming control priority

4. Read the notes and confirm these with "Accept."
5. Activate the "Enables" checkbox to enable the drive.
   All enables are now set with the exception of ON/OFF1.

6. Enter the desired setpoint in the entry field, and, as a safety setting, slide the scaling to 0%.

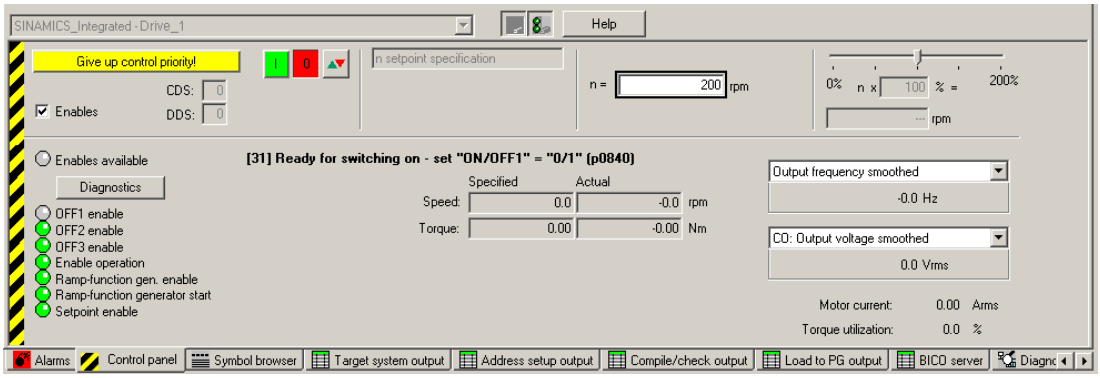


Figure 10-187 Entering a setpoint

7. Click the "Drive On" button. The green "Enable available" LED lights up. If you move the slider to the right, the drive rotates. The current motor speed is displayed in "Actual."
8. Click "Drive Off" to stop the drive after you completed the test.
9. Deactivate the enable and click the "Give up control priority" button to deactivate control from the PG/PC.

Creating and testing axes

Overview of SIMOTION Engineering

Performing engineering with SIMOTION SCOUT

The SIMOTION SCOUT engineering system can be used to insert axes in your project.

1. First, run through the axis wizard to configure the axes and interconnect to the real drive (e.g. SINAMICS Integrated).
2. Provided you have completed the configuration at the drive end, we strongly recommend for faster working that the SINAMICS Integrated is deactivated via "Target system" > "Select target device".
3. Complete your SIMOTION application, for example, by creating axis functions and SIMOTION execution programs.
4. Compile the project and download it to the SIMOTION D410-2.

Creating an axis with the axis wizard

Overview

The TO axis provides the user with the technological functionality and the interface to the drive/actuator. The TO axis processes the motion control commands from the user program (e.g. MCC) and coordinates the interface to the drives. It executes control and motion commands and indicates statuses and actual values.

The TO axis communicates with an actuator (drive or hydraulic valve) via a fieldbus system (PROFIBUS or PROFINET via PROFIdrive protocol) or via a direct setpoint interface (analog ±10 V or pulse/direction).

When running through the axis wizard, the basic settings are made for the axis and the TO axis interconnected to a drive (e.g. SINAMICS Integrated). The following extended options are available when "Use symbolic assignment" has been activated:

- A real axis is interconnected to an already configured drive
- A real axis including drive is created via the axis wizard and the drive interconnected to the axis
- A real axis is created without assigning this to a drive (assignment is made later)

Inserting an axis

1. In the Project Navigator, double-click the entry "Axis" > "Insert Axis."
   This will access the axis wizard. Set the required technology and then click "OK."

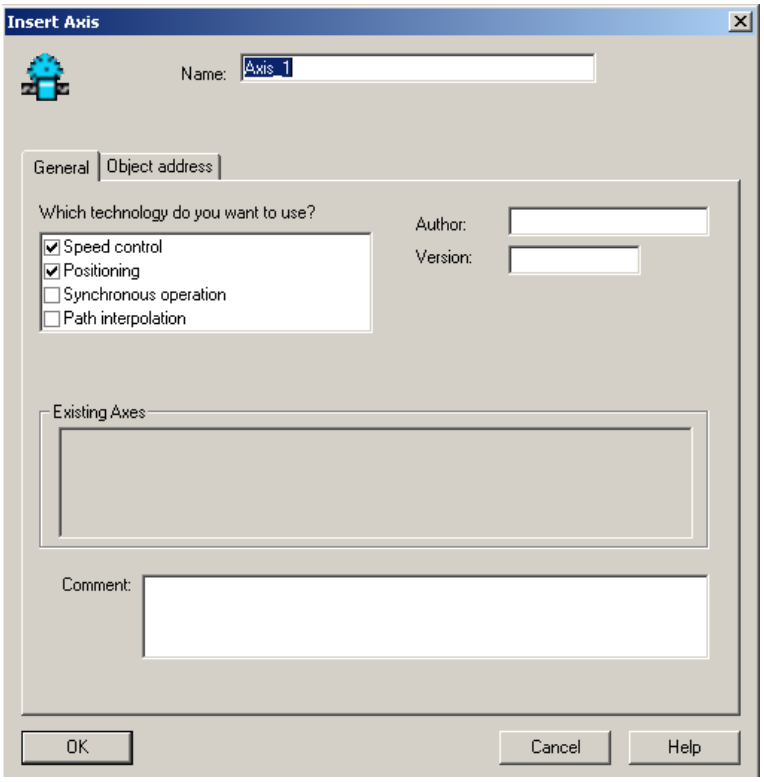


Figure 10-188 Inserting an axis

2. Set an axis type and, if required, configure the units.

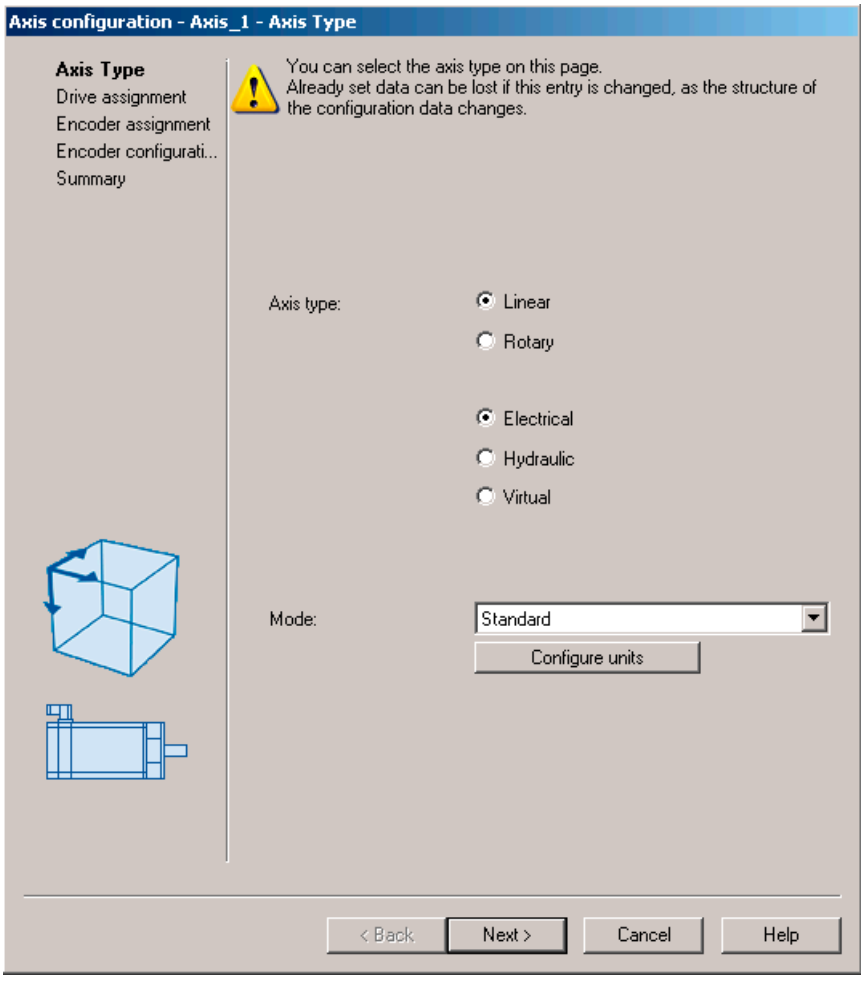


Figure 10-189 Defining the axis type

3. Create a new drive or make the assignment to an existing drive.

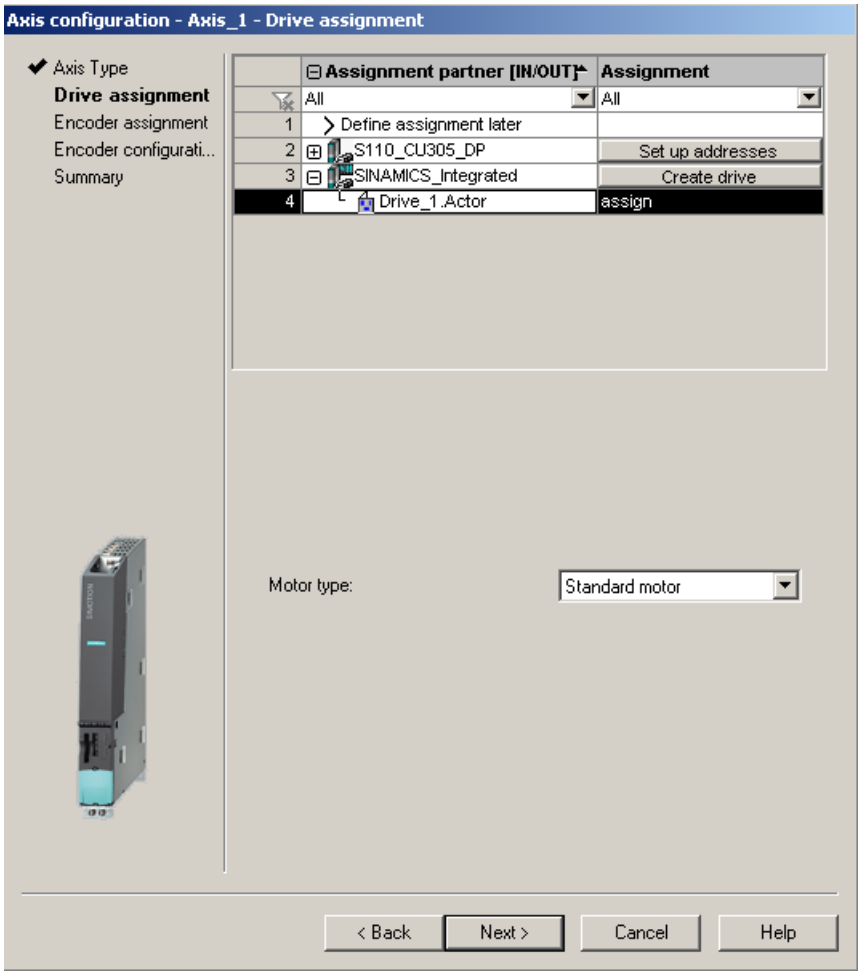


Figure 10-190 Assigning a drive

The following setting options are available for the drive assignment:

- Assigning a drive
  Assigning a previously configured drive.
- Define a subsequent assignment
  The axis should not be assigned to a drive until a later point in time. Accordingly:
  - The PLC and motion control functions to be completely configured by a programmer even without drive know-how using technology objects (e.g. TO axis) and loaded to the device.
  - The drives to be separately configured and optimized by a drive expert.
  - The technology objects to be symbolically assigned later to the drive objects via an interconnection dialog box.

- Creating a drive
  From the Assignment dialog box, a new drive can be created on an existing drive unit (e.g. SINAMICS Integrated) and assigned to the axis. This allows the axis, including the drive, to be created in one operation. It is not necessary to configure a drive before creating an axis.
- Setting up addresses
  If "Use symbolic assignment" has been deactivated, the addresses must be set up manually. This is required, for example, for drive units that do not support symbolic assignment (e.g. SINAMICS S120 with FW version < 2.6.2, MASTERDRIVES, SIMODRIVE, etc.).

The address list in the "All Addresses" view provides an overview of the assignments to all interfaces of the TO axis. From this view, the assignments can also be changed via the Interconnection dialog box (... button).

Note

The symbolic assignment must be deactivated for the drive and axis configuration without symbolic assignment and adaptation.

Run through the wizard and enter the settings of your system. The required axis telegram as well as the addresses used are automatically determined by the engineering system. The telegram is also extended and interconnections in the drive created automatically depending on the selected TO technology (e.g. SINAMICS Safety Integrated).

Click **Finish** to confirm the "Summary" window.

The configured real axis is displayed in the project navigator.

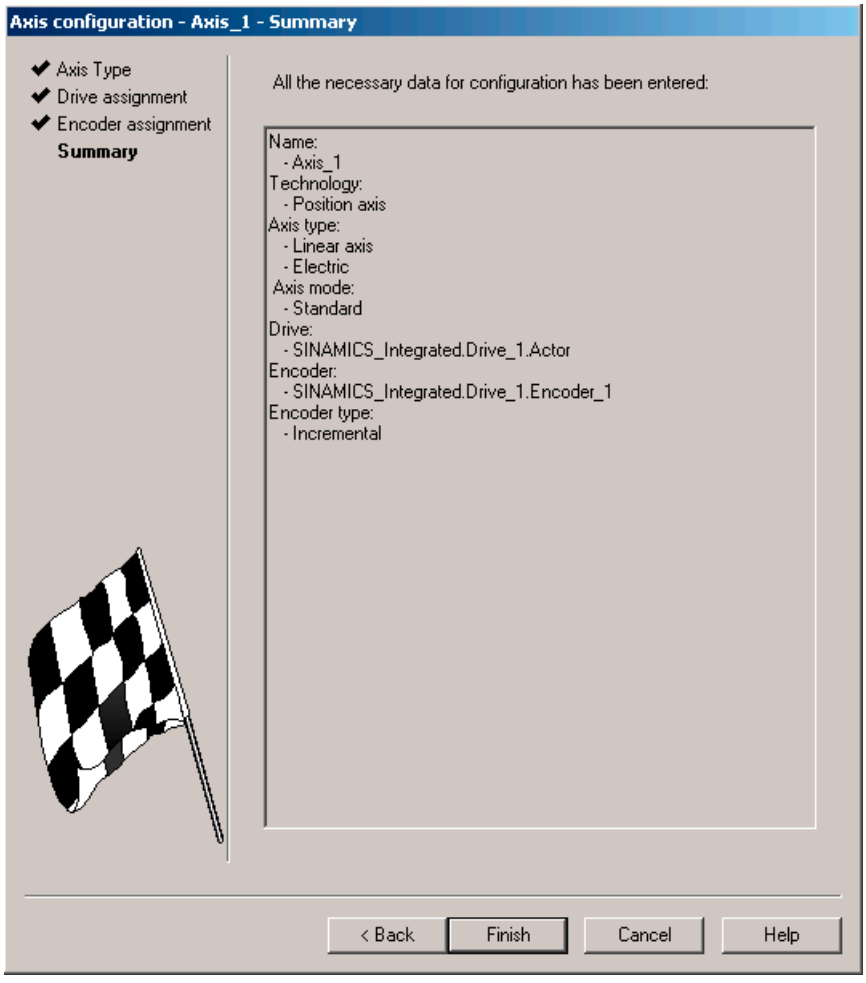


Figure 10-191 Axis wizard summary

Note

During system power-up, reference variables as well as drive and encoder data of the SINAMICS are automatically taken over for the SIMOTION configuration data of the SIMOTION technology objects "TO axis" and "TO externalEncoder".

Encoder assignment

With a position axis, encoder 1 is also created on the TO axis (motor encoder) and automatically assigned to the first encoder on the drive.

If encoder 2 (direct encoder) is created at the TO axis, it is assigned to the second encoder of the drive control.

Result

The configured axis will appear in the project navigator.

Save and compile the project and download it to the target system.

After running through the axis wizard, the symbolic drive assignment is displayed via "Configuration" of the axis as well as via the address list (All addresses view).

The Assignment dialog box can be called again from these dialog boxes using the ... button.

Instead of calling the Assignment dialog box, it is also possible to edit the input field SINAMICS_Integrated.SERVO_02.Actor ... containing the symbolic name directly.

TBD, DSDB, and SIDB

In the "Configuration" dialog box of the TO axis, you can activate the following functions from "Functions" > "Change":

- Technology data block (TDB): for the cyclic exchange of technology data, e.g. actual torque value
- Drive Safety data block (DSDB), V4.4 and higher: to support the SINAMICS Safety Integrated Functions by means of the TO
- Safety Info Data Block (SIDB): Predecessor of the DSDB (for compatibility only)

The assignment is always made to the drive DO of the actuator of the axis. The system automatically generates a telegram extension and the BICO interconnection of the relevant SINAMICS parameters.

Note

The safety data blocks (DSDB or SIDB) are automatically configured by the engineering system and interconnected in the drive.

The PROFIsafe telegram must be configured by the user.

If the activation of the safety functions is to be made using PROFIsafe, configure the PROFIsafe communication to the higher-level SIMATIC F-CPU (see *SINAMICS S120 Safety Integrated* Function Manual).

I/O signals at the TO axis

For the assignment of I/O signals on the TO axis (e.g. the inputs for the homing output cam or hardware limit switches), call the assignment dialog box from the parameterization dialog boxes of the TOs created or from the address list (view of all addresses) by clicking the ... button.

Additional references

See Section Downloading the project to the target system (Page 7419).

For further information about the symbolic assignment, see the *SIMOTION Runtime Basic Functions* Function Manual.

Testing an axis with the axis control panel

Axis control panel

The axis control panel is used exclusively for testing axes.

You can use the axis control panel for the following tasks, for example:

- Testing all system components before the axis movement is controlled by a program.
- Testing as to whether you can move the axis using the axis control panel if a fault is detected.
- Moving the axes for tuning purposes (controller tuning).
- Executing active homing.
- Setting and resetting the axis enable signal.
- Testing the axis that was created.

Requirements

The following requirements must be fulfilled for testing:

- The project has been downloaded to the target system.
- SIMOTION SCOUT is in online mode.

Axis test

1. Open the "AXES" folder in the Project Navigator and click the "Control panel" entry below the axis (for example, Axis_1).
   The axis control panel is displayed.

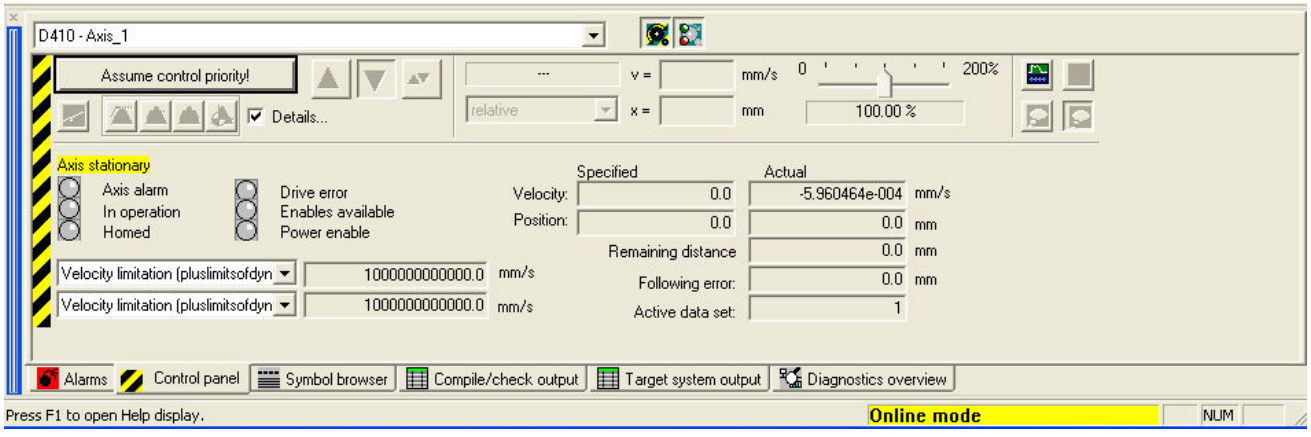


Figure 10-192 Axis control panel

2. To display the control area and axis diagnostics, click the "Show/hide control area" and "Show/hide diagnostics area" buttons.

3. Click "Assume control priority".

Note

In order to move the axis from the PG/PC, you must assume control priority. However, by pressing the SPACER bar, you can stop the axis at any time.

4. The further procedure depends on the CPU status:
   - Case 1: CPU in the STOP/STOPU state
     If the CPU is in the STOP state, a message will appear, indicating that the CPU has been put in the STOPU state.
     A safety message appears in a further dialog box and must be accepted.
     The activated service function is then displayed via the LEDs (RUN/STOP flashing yellow/green with 2 Hz).
   - Case 2: CPU in the RUN state (as of SCOUT/Kernel V4.4)
     Control priority can only be assumed if the axis is not in motion.
     After accepting the safety message, a message is shown on the control panel that the CPU is in the RUN state, that the user program is running, and further axes may be moving. (LED display: RUN).
     If the control priority for a TO is active, commands for the TO from the user program are rejected with an error code. Alarm 30009: reason 0x04 is output.
5. To enable the axis, click "Set/reset enables".
   Confirm the "Switch Axis Enable" dialog box with "OK".

Note

If the control panel is operated in the RUN state, setting/canceling the axis enable can alternatively be controlled via the user program.

6. To traverse the axis, click the "Position-controlled traversing of the axis" button.
7. Enter a velocity and close the dialog box by clicking "OK".
8. Click the "Start movement" button. You can monitor the traversing motion under velocity and position. Use "Stop motion" to stop axis movement again.
9. Click "Set/reset enables" to remove the enable. Confirm the "Remove Axis Enable" dialog box with "OK".
10. Click the "Give up control priority" button to deactivate axis control from the PG/PC. In this operating state, the axes can no longer be controlled from the PG/PC.

Note

For commissioning kinematics transformations, a path control panel is available as of SIMOTION V4.4

See also

Please refer to the additional information in the SCOUT Online Help (Axis control panel index).

Setting up addresses and message frames

Overview

After all SINAMICS components have been configured, addresses must be determined for the process data exchange between the drive and the controller.

This procedure depends on whether **symbolic assignments** are used.

- With symbolic assignment, the addresses are determined automatically by the engineering system; refer to Section Setting up communication for symbolic assignment (Page 7458).
- In the absence of symbolic assignment, addresses must be determined manually; refer to Section Message frame configuration (Page 7458).

Setting up communication for symbolic assignment

The communication for the symbolic assignment can be set up with the following actions:

- From the SCOUT menu
  Call the "Project" > "Setup communication for symbolic assignment" menu.
- At "Download project to target system"
- At "Save project and compile changes"

When setting up the communication, the message frames, BICO interconnections and addresses are set up for the entire project.

Message frame configuration

Requirement

You have configured the drive unit.

On the basis of this configuration, one or more of the following actions should be performed:

- The automatic PROFIdrive telegram setting for a drive object should be activated/deactivated.
- The automatic telegram extension for a drive object should be activated/deactivated.
- The automatic address adaptation for a drive object should be activated/deactivated.
- PROFIdrive telegrams should be configured for drive objects.
- The addresses should be set up.
- Telegrams should be extended manually.

Procedure

In the project navigator, open the "Communication" > "Telegram configuration" entry under "SINAMICS_Integrated."

The "SINAMICS Integrated - Telegram Configuration" dialog box is displayed with the PROFIdrive PZD telegrams tab.

The dialog box lists all the available drive objects. The possible setting options are described in the following.

Note

When using the symbolic assignment, the default setting does not have to be changed or configured.

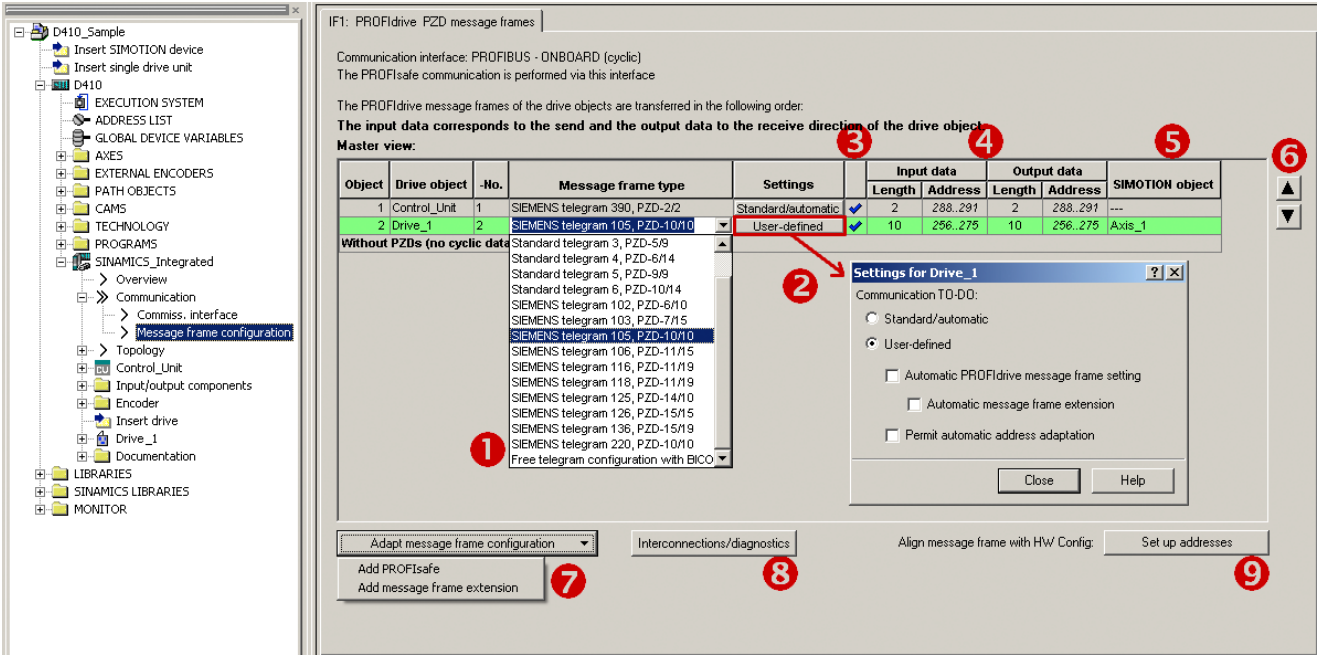


Figure 10-193 Telegram configuration

Table 10-119 Explanation of the figure

No.	Meaning
①	Selection of a telegram • The drive telegrams (telegrams 1 ... 6 and telegram 1xx) are defined in accordance with the PROFIdrive specification and can be selected based on the required functional scope. • You can transfer the signals of the I/Os or the global measuring inputs, for example, via the telegrams 39x. Telegram 39x is also required for the time synchronization between SIMOTION and SINAMICS. • Free telegram configuration with BICO allows you to define your own telegram. • Free telegram configuration with p0915/p0916 (for TM15/17).
②	The "Standard/automatic" and "User-defined" settings are only visible if "Use symbolic assignment" is activated. Using the setting "Standard/automatic" is generally recommended. The "User-defined" setting allows the automatic telegram setting, telegram extension, and address adaptation to be deactivated or activated. • "Automatic PROFIdrive telegram setting" allows the telegram to be set by the system depending on the configured technology (telegram selection, e.g. for drive and control unit incl. onboard I/O). • "Automatic telegram extension" allows the telegram to be extended by the system depending on the configured technology (e.g. if the technology data block is activated in the axis configuration). • "Permit automatic address adaptation" allows addresses to be adapted by the system in the case of address offsets, for example. Address offsets can occur, for example, if a telegram is extended and the adjacent addresses are already occupied by other telegrams. With TM15/TM17 High Feature, "Automatic PROFIdrive telegram setting", "Automatic telegram extension", and "Automatic address adaptation" cannot be deactivated by design, since for these drive objects the telegram is always set up in accordance with the parameterized terminal functionality (DI, DO, output cam, measuring input) and cannot be extended. "Automatic PROFIdrive telegram setting" and "Automatic telegram extension" must be deactivated if the telegrams are to be configured manually for TM15 DI/DO and TM31 and interconnected with BICO. See Section Setting up communication for symbolic assignment (Page 7458).
③	Telegram status (Meaning of the symbols, see table below)
④	Length: Displays the size of the telegram component. Address: Addresses in **HW Config** The addresses will be displayed only if the addresses have been defined.
⑤	Displays the SIMOTION object that is interconnected to the SINAMICS object (e.g. axis or encoder).
⑥	Changing the telegram order Note: Before the alignment, all drive objects without I/O addresses ("---..---") must be moved behind the objects with valid I/O addresses or those still to be aligned ("???..???").
⑦	"Manual" adaptation of the telegram configuration (e.g. when additional data, such as a motor temperature, is to be transferred via the telegram).
⑧	Display of the individual control and status words of the associated telegram.
⑨	Setting up the addresses (alignment of the addresses with **HW Config**) Only the addresses for the respective drive unit are determined (no automatic determination of telegrams / BICO interconnections).

Note

If symbolic assignment is deactivated, the following applies:

If the telegrams for drive objects (drives, Terminal Modules, etc.) change, you must set up the addresses again. The addresses are not updated automatically.

Telegram status

The icons in the status column show the following information:


The telegram differs from the configuration in **HW Config**. You must align the configuration with **HW Config**.


You are using a predefined standard telegram or a free BICO interconnection.


You are using an altered standard telegram that you have extended to include supplementary data.


You are using a telegram for which one of the two telegram lengths is too long. The drive project cannot process this entry.

Error correction (symbolic assignment deactivated)

Based on the 39x telegram, SIMOTION generates further configuration information (FastIO configuration) for the following functions:

- Time-of-day synchronization SIMOTION ⇔ SINAMICS
- Use of onboard I/Os of SIMOTION D, CU, or CX
- Use of cams and global measuring inputs
- System function _setDriveObjectSTW

If the telegrams are defined manually (symbolic assignment is deactivated), a telegram 39x must be set up in the telegram configuration. The telegram then has to be aligned with HW Config via "Set up addresses."

If use of the functions stated above is not possible, generate the FastIO configuration anew. For this purpose select in the project tree the affected SIMOTION D Control Unit, the SINAMICS CU or Controller Extension CX and open the "FastIO" > "Create new configuration" shortcut menu with the right mouse button. Then compile the project and load it into the CPU. Perform a restart.

The FastIO configuration is also used for the telegram of the Terminal Modules TM15 and TM17 High Feature. Proceed in the same way if problems occur.

Linking an additional encoder (optional)

General information

SIMOTION D410-2 features a DRIVE-CLiQ X100 interface for connecting an encoder. SIMOTION D410-2 provides the option of integrating and configuring further encoders in addition to the motor encoder.

The following encoders are supported for operation with SIMOTION D410-2:

- Encoders with a DRIVE-CLiQ interface
- Encoders connected to SIMOTION D410-2 or CUA32 via the onboard encoder interface (X23)
- Encoders connected to SIMOTION D410-2 using an SMx module
- Encoders connected using PROFIBUS or PROFINET

Configuring additional encoders

The second encoder can be used at SIMOTION D410-2, for example, as:

- A machine encoder (second encoder = direct measuring system).
  A direct measuring system measures the technological parameter directly, i.e. without the interference of influences such as torsion, backlash, slip, etc. This may enable improved smoothing of mechanical influences. If you use a second encoder as a machine encoder, you can work with the encoder changeover function.
- An external encoder.
  You can use the external encoder for recording an external master value, for example.
- Encoder for hydraulic axes
- Encoders for the implementation of cam controllers

Configuring tasks

Encoders connected via PROFIBUS/PROFINET are only configured in SIMOTION.

Encoders connected using SMx, DRIVE-CLiQ or the onboard encoder interface must be configured on the drive side (SINAMICS Integrated) and in SIMOTION.

Configure this additional encoder on drive side (SINAMICS Integrated) and in SIMOTION:

1. Configuring additional encoders on the drive (Page 7462)
2. Connecting additional encoders via PROFIBUS/PROFINET (Page 7464)

These steps in configuring are described in the next section.

Configuring additional encoders on the drive

The following options are available for configuring additional encoders on the drive:

- Configuration of a second encoder on the drive
- Configuration of an encoder as drive object (as of SINAMICS firmware V4.3)

Configuring a second encoder on the drive

The configuration of a second encoder on the drive is useful when the second encoder value is also to be used for this drive (e.g. motor or machine encoder). Note that a maximum of only two encoder values can be transferred via PROFIdrive message frames.

In principle, the second encoder value can be freely used (e.g. for acquisition of an external master value), however the use of an encoder as a separate drive object (drive object DO encoder) is preferable because of the clear functional separation.

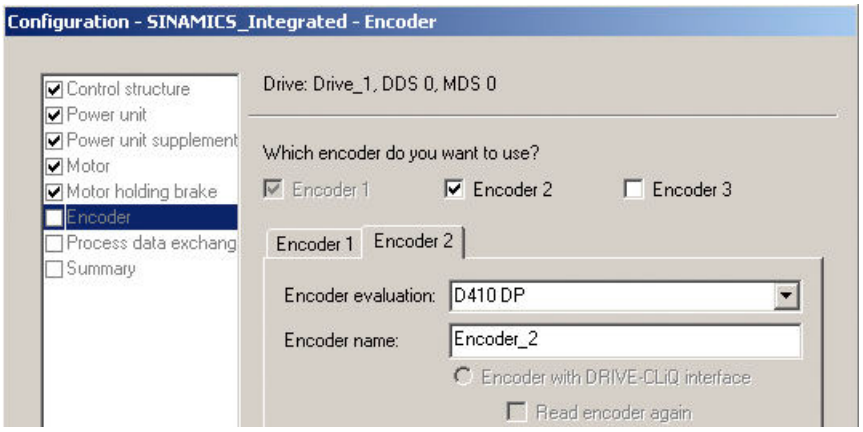


Figure 10-194 Configuration of a second encoder on the drive

Configuring an encoder as drive object

The configuration of an encoder as drive object (drive object DO encoder) has the advantage that this encoder can be used independently of a configured drive (e.g. for acquisition of a master value).

The configuration is performed by inserting an encoder via the project navigator.

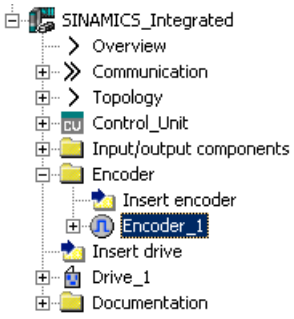


Figure 10-195 Configuration of an encoder as drive object

Note

In the same way as for axes, a "DO encoder" can also be symbolically interconnected to a "TO externalEncoder".

Connecting additional encoders via PROFIBUS/PROFINET

Options

With regard to encoder integration, further encoders can also be connected via PROFIBUS or PROFINET. The following options are available:

- Encoder interconnection using a PROFIdrive message frame (encoder with message frame type 81 and 83)
- Encoder interface as a direct value in the I/O area.

Additional references

Detailed information is contained in the *SIMOTION TO Axis, Electric/Hydraulic, External Encoder* Function Manual

Symbolic assignment of I/O variables

Assignment to the PROFIdrive message frame of the TO axis

I/O variables which you require for purposes such as display and diagnostics can be assigned from the address list to individual components (status word, for example) via the assignment dialog of the PROFIdrive message frame. Only components suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, this is determined via the assignment partner after the selection.

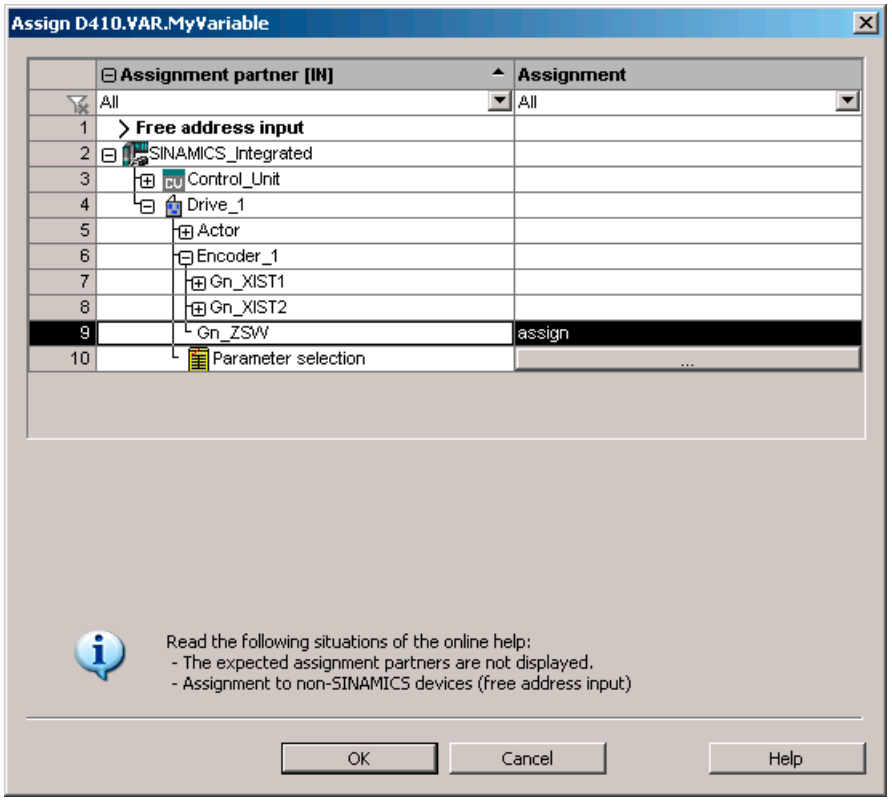


Figure 10-196 Assignment of I/O variables to the PROFIdrive message frame

Assignment to drive parameters

I/O variables from the address list can be assigned to drive parameters using the assignment dialog. Only parameters suitable for the data type of the I/O variable are displayed. If no data type is specified at the I/O variable, this is determined by the parameter selection.

An extension of the standard message frame is created automatically for the transfer of the parameters to/from the drive.

Procedure

1. Open the Assignment dialog box from the address list (view of all addresses).
   The Assignment dialog opens with the corresponding assignment partners.
2. Click the "..." button in the parameter selection line to open the parameter list.

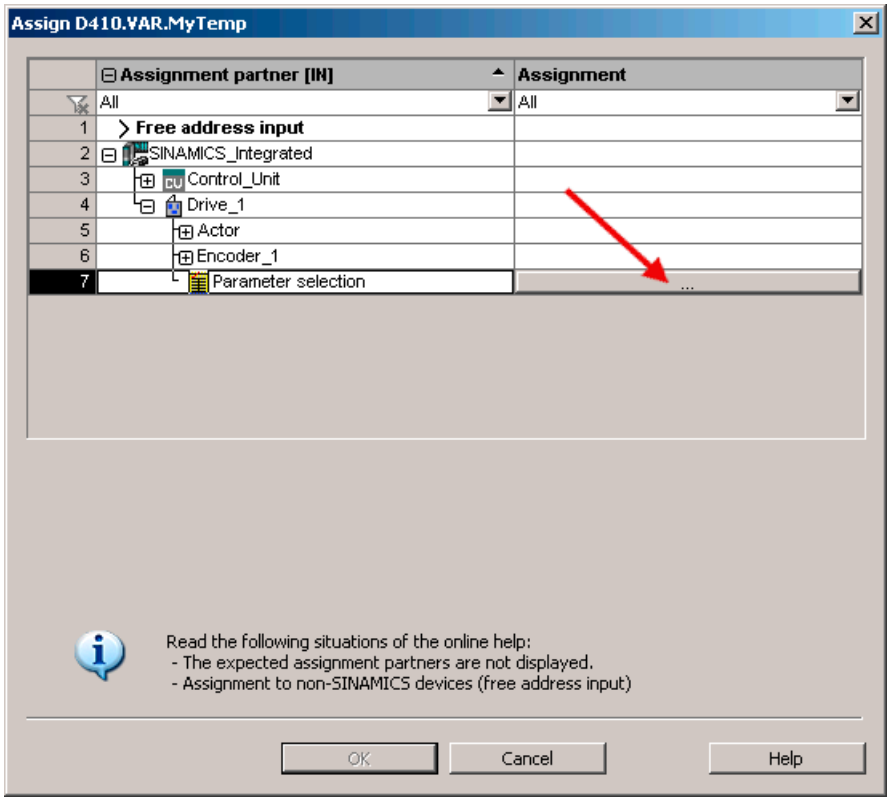


Figure 10-197 Assignment dialog for drive parameters

3. Select the desired signal source (e.g. DO drive). Then select the required parameter.

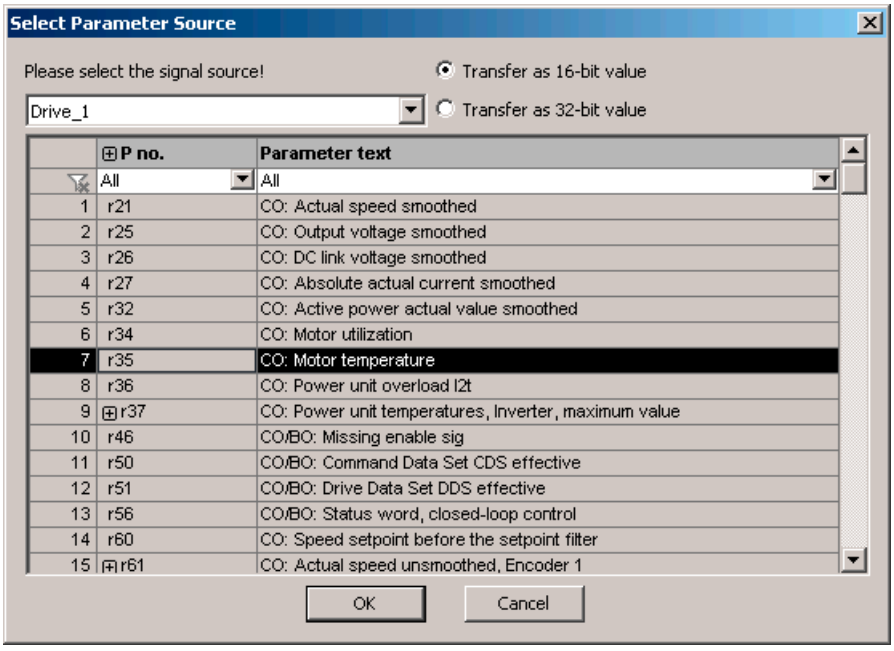


Figure 10-198 Dialog box for the DO and parameter selection

4. Click "OK" to accept the selection.

5. The desired SINAMICS parameter is assigned to the I/O variable in the interconnection dialog box.

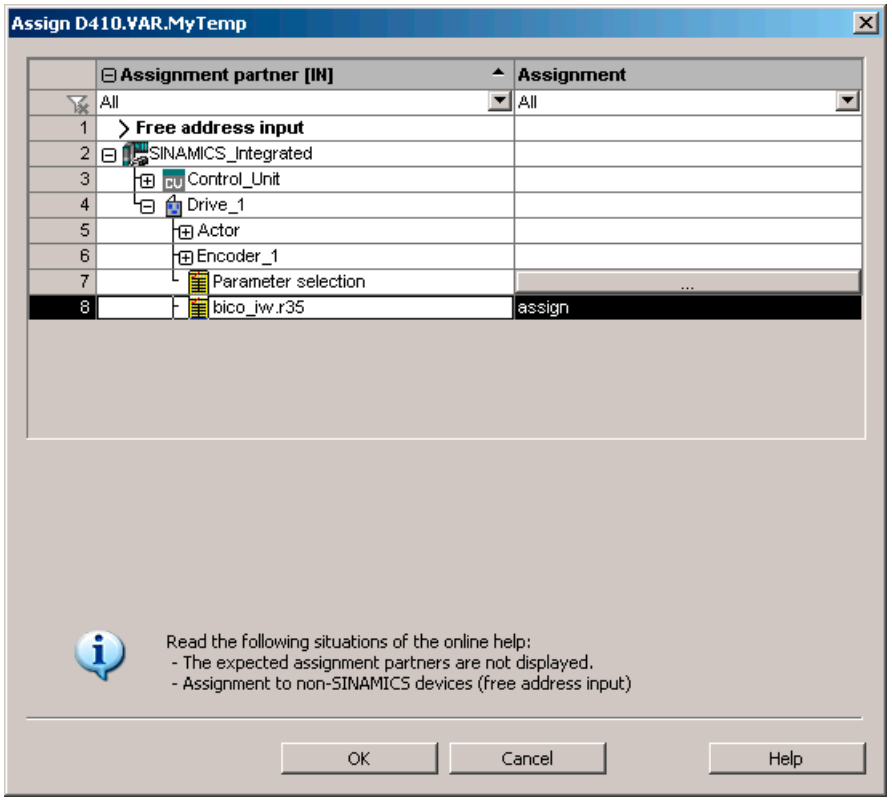


Figure 10-199 Assigned drive parameters

6. Click "OK" to accept the assignment.

The following table shows the possible types of assignment:

Name of the assignment	Data type	Direction	Transferrable BICO parameters
BICO_IW.<parameter number>	WORD	Input	All CO parameters (BICO source)
BICO_QW.<parameter number>	WORD	Output	All CI parameters (BICO sink)
BICO_ID.<parameter number>	DWORD	Input	All CO parameters (BICO source)
BICO_QD.<parameter number>	DWORD	Output	All CI parameters (BICO sink)

Syntax of the assignment names

- A number of parameters (separated by periods) are specified for outputs (SINAMICS side = received data) which can be interconnected with a number of BICO sinks
- If the transferred parameter is on another drive object (DO), the DO name precedes the parameter. "#" is used as a separator between the DO name and the parameter.
- Individually transferred bits of a parameter appear in brackets [x]

Configuring drive-related I/Os

Overview of the symbolic configuration of I/Os

Overview

SIMOTION D410-2 as well as the SINAMICS S110/S120 control units and additional components (TMs) have I/Os that can be used by the drive unit and by SIMOTION.

I/Os originally assigned to SINAMICS can be used by SIMOTION only when they have been interconnected to a message frame.

Symbolic assignment

SIMOTION SCOUT supports by default the symbolic configuration of I/Os, see Section Symbolic assignment / adaptation (Page 7401).

The symbolic assignment simplifies the configuration significantly:

Table 10-120 Comparison of configuration with/without symbolic assignment

Configuration	With symbolic assignment	Without symbolic assignment
Configuring message frames	So that SIMOTION can use SINAMICS I/Os, the required message frames are created automatically	Message frames must be set manually (either predefined message frame (e.g. 39x) or free message frame configuration)
BICO interconnections	The required BICO interconnections are performed automatically (I/Os are interconnected to a message frame)	With predefined message frames (e.g. 39x), the BICO interconnections are made automatically With free message frame configuration with BICO, the interconnection must be made by the user
Parameterization of the I/O functionality (e.g. measuring input)	Parameterization via screen forms	Parameterization via screen forms and partly via parameters in the expert list
Handling of I/O addresses	Handling of addresses is not required because of symbolic assignment	I/O addresses must be determined
Setting up addresses	Addresses are set up automatically, refer also to Section Setting up communication for symbolic assignment (Page 7458)	Addresses must be set up manually, refer also to Section Message frame configuration (Page 7458)

Only the configuration with symbolic assignment is described in the following. For further information on the configuration of drive-related I/Os without symbolic assignment, see Appendix Configuring drive-related I/Os (without symbolic assignment) (Page 7570).

Procedure

The configuration of the I/Os is divided into two basic steps:

1. Configuring I/O terminals (Page 7470)
   The functionality of an I/O channel is configured (e.g. configuration of a DI/DO as digital output).
2. Configuring technology objects and I/O variables (Page 7474)
   The access of technology objects and I/O variables to I/Os is configured. The configuration is symbolical, whereby only "function-compatible" I/O channels are offered for selection.
   Example:
   For the TO measuringInput, only symbolic assignments of the type MI (measuring input) are offered for selection.

The engineering system automatically makes the required message frames and the interconnections to the configured I/Os.

Configuration options

The following table provides an overview of the configuration options for the I/O terminals of various modules.

Table 10-121 Configuration of the I/O terminals overview

Module	Use of the I/Os by		Configuration of the I/O terminals	Supports symbolic assignment
	SIMOTION	SINAMICS		
SIMOTION D410-2 • Terminal X120/X121 • Terminal X130/X131	X [1]	X	On the drive unit (CU)	As of SIMOTION V4.3
SIMOTION D4x5-2 • Terminal X122/X132 • Terminal X142	 X [1] X	 X -	 On the drive unit (CU) On the D4x5-2 (HW Config)	As of SIMOTION V4.2
SIMOTION D4x5	X [1]	X	On the drive unit (CU)	As of SIMOTION V4.2
CX32-2, CX32	X [1]	X	On the drive unit (CU)	As of SIMOTION V4.2
SINAMICS S110 CU305	X [1]	X	On the drive unit (CU)	As of SINAMICS V4.3
SINAMICS S120 • CU310 • CU310-2 • CU320 • CU320-2	X [1]	X	On the drive unit (CU)	 • As of SINAMICS V2.6.2 • As of SINAMICS V4.4 • As of SINAMICS V2.6.2 • As of SINAMICS V4.3
TB30, TM15 DI/DO, TM31	X [1]	X	On the drive unit (TB30 or TM)	Yes
TM41 [2]	X [1]	X	On the drive unit (TM41)	Yes [2]

Module	Use of the I/Os by		Configuration of the I/O terminals	Supports symbolic assignment
	SIMOTION	SINAMICS		
TM15, TM17 High Feature	X	-	On the drive unit (TM15 or TM17)	Yes
Time-based I/O • ET 200MP TM timer DIDQ 16x24V • ET 200SP TM timer DIDQ 10x24V	X	-	On the ET 200 I/O system	As of SIMOTION V4.4 HF6

1) I/Os are originally assigned to a SINAMICS drive unit and can be assigned to SIMOTION via configuration

2) TM41 supports symbolic assignment only for encoder interfaces (not symbolic assignment for DI, DO and AI)

Note

The module hardware for TM15 and TM15 DI/DO is identical. A distinction is only made by the addition of the component in the SIMOTION SCOUT project navigator using "Insert input/output component".

I/Os that are originally assigned to the SINAMICS drive unit can also be used by SIMOTION via configuration.

- An output is always only exclusively available for the SINAMICS drive unit or SIMOTION.
- An input used by SIMOTION can also be interconnected on the drive side.

In the following sections describe in detail how to configure the I/O terminals.

SIMOTION D410-2 Configuring I/Os

Procedure

The onboard inputs/outputs of the SIMOTION D410-2 are assigned to SINAMICS Integrated. The configuration is therefore performed via the drive unit ("SINAMICS_Integrated" > "Control_Unit" > "Inputs/outputs").

The properties of the I/O channel can be configured in the parameterization dialog box. With the bidirectional digital I/Os, for example, an I/O channel can be:

- Parameterized as input or output
- Inverted
- BICO-interconnected (use as drive I/O)
- Used as digital input for SIMOTION with "DI (SIMOTION)"
- Used as digital output for SIMOTION with "DO (SIMOTION)"

- Used as a global measuring input for SIMOTION with "Measuring input (SIMOTION)"
- Used as an output cam output for SIMOTION with "output cam (SIMOTION)"

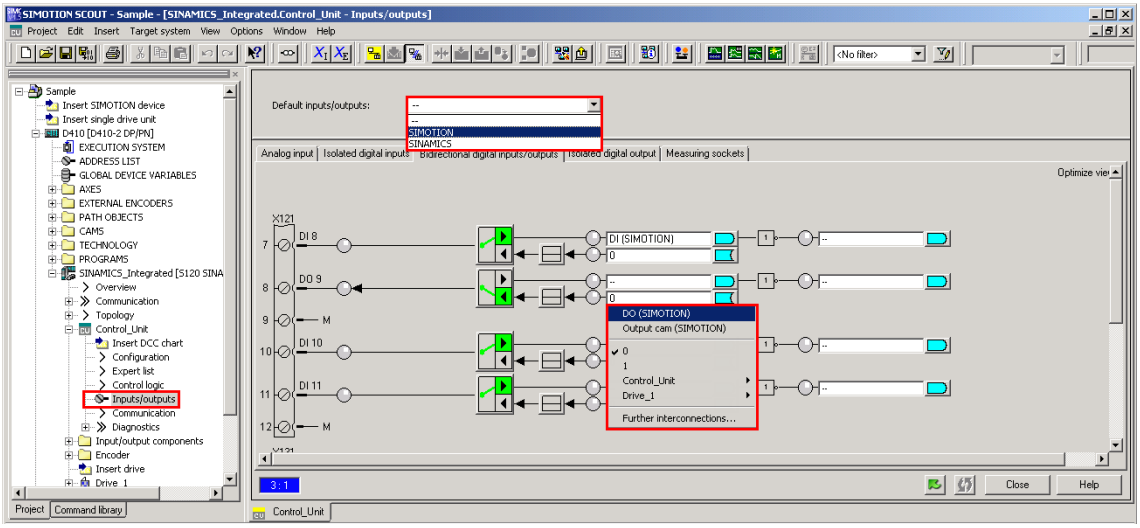


Figure 10-200 Configuration of D410-2 I/Os (terminal X121)

Note

The properties of the I/O channel must be configured offline; Online changes only take effect after a restart (power off/on), provided that the setting has been previously saved retentively with "Copy RAM to ROM".

24 V supply for DO

If no digital outputs are used, the SIMOTION D410-2 can be supplied via the Power Module. A 24 V supply must be connected to terminal X124 for the digital outputs to be used. If a digital output is parameterized and the 24 V supply is not connected (or the level is too low), alarm A03506 is issued on the SINAMICS side (can also be parameterized as a fault).

Data transfer

If the D410-2 onboard I/Os are interconnected using symbols (or if telegram 39x is used for the onboard I/Os), the status information of the DI and DO will be transmitted to cu.p2048 at the PROFIdrive PZD sampling rate. Sampling of the inputs and outputs is also performed in the sampling time parameterized according to cu.p0799.

The same applies if the I/Os are manually interconnected to a drive telegram via a BICO converter.

Transfer of the output values and feedback of the input values is therefore subject to dead times and jitter.

For time-critical applications, use of measuring probes or cams is recommended. Alternately, the isochronous I/Os, I/Os of TM15, TM17, or isochronous ET 200 I/Os can be used.

Configuration of CU3xx/TMxx I/Os

Overview

The configuration is performed in a similar way as for the onboard I/Os for the SIMOTION D410-2, i.e. the I/Os can be

- BICO-interconnected (use as drive I/O)
- Used by SIMOTION

See also Section SIMOTION D410-2 Configuring I/Os (Page 7471)

Note

If symbolic assignment is subsequently activated for a project in which telegrams have already been configured and interconnected, this project can be changed together with the BICO interconnections!

For this reason, make a backup copy of your project before activating the symbolic assignment.

The TM15 DI/DO and the TM31 are particularly affected here (see Section Symbolic assignment / adaptation (Page 7401)).

Configuration of the ET 200SP/MP timer DIDQ

The I/O terminals of the ET 200SP and ET 200MP timer DIDQ can be used as drive-related I/Os for measuring input and output cam applications.

The DI/DQ timers are configured via the hardware configuration of STEP 7 or TIA Portal. The properties of the I/O channel can be configured in the parameterization dialog box, for example

- Timer DI for use of the I/O as measuring input input or
- Timer DQ for use of the I/O as cam output

Note

For the technological use of the timer DIDQ, you must operate the I/O station isochronously on PROFINET:

- ET 200SP timer DIDQ: With SCOUT or SCOUT TIA, IM 155-6 PN HF required
- ET 200MP timer DIDQ: Only with SCOUT TIA, IM 155-5 PN ST or HF required

Operation via PROFIBUS is not possible.

For detailed information on the configuration, see *Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA* Commissioning Manual

Configuring technology objects and I/O variables

Configuring global measuring inputs

Overview

The type of measuring input must be selected during the configuration of the TO measuring input, see following table.

Table 10-122 Measuring input types

Measuring input types	Explanation
Standard (global measuring input)	Compared with the drive-related local measuring inputs, global measuring inputs have extended functionality and also support a symbolic configuration. They are therefore set as standard.
Drive-related (local measuring input)	The drive-related local measuring inputs are configured via drive parameters, see Section Configuring drive-related I/Os (without symbolic assignment) (Page 7570) in the Appendix.
Listening measuring input	Through the configuration of a listening measuring input, measuring can be performed simultaneously on several axes or external encoders with one measuring input. Detailed information can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs* Function Manual.

A detailed comparison of "local" and "global" measuring inputs as well as an overview of which modules support local or global measuring inputs can be found in Appendix Configuring drive-related I/Os (without symbolic assignment) (Page 7570).

Procedure

If a global measuring input is selected, it must be assigned a hardware input.

Click the "Assign" button ... to open the Assignment dialog box and select a free (i.e. not yet used) I/O.

Note

Only those I/Os are displayed that have the appropriate measuring input functionality (MI_xx [channel name, terminal number]). If no suitable I/Os are displayed, you must first configure the I/Os (I/O must be configured as "measuring input").

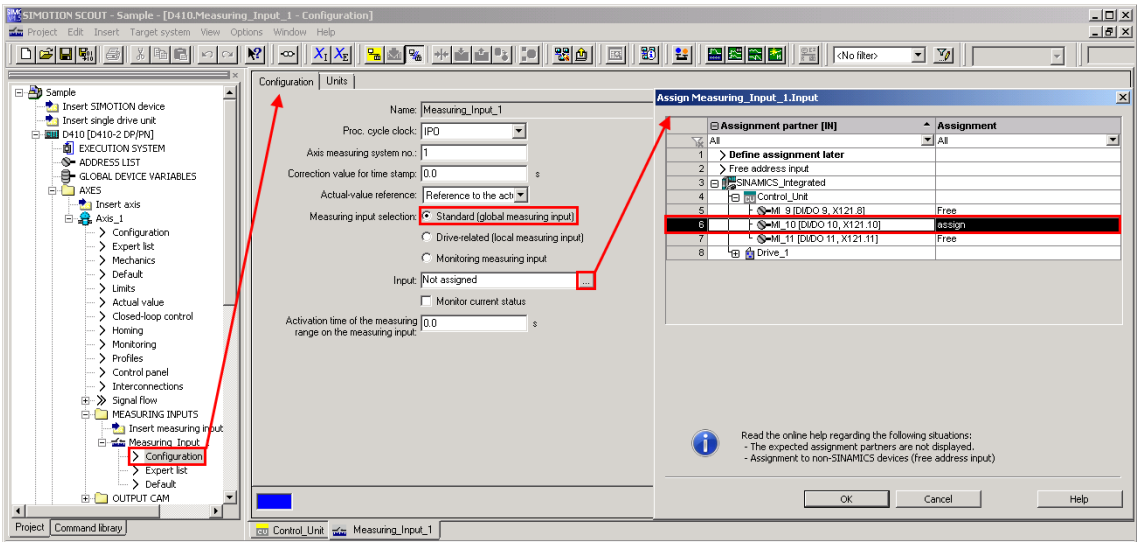


Figure 10-201 Configuration of a global measuring input for SIMOTION D410-2

Additional references

Detailed information on the configuration of the TO measuring input can be found in the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Configuring local measuring inputs

Local measuring inputs are drive-related measuring inputs. The configuration is performed via drive parameters.

For further details, see:

- Configuring drive-related I/Os (without symbolic assignment) (Page 7570) In the Appendix
- *SIMOTION Output Cams and Measuring Inputs* Function Manual

Configuring output cams / cam tracks

Overview

The type of cam output must be selected during the configuration of the TO outputCam and TO camTrack.

The following output types are available:

Table 10-123 TO outputCam / TO camTrack output types

Cam output on...	Explanation
Output cam output (CAM)	The cam output is performed on the basis of an internal time stamp. The temporal resolution of the cam output depends on the hardware used. Supported hardware: • SIMOTION D4x5-2 (terminal X142): Resolution 1 µs • TM17 High Feature: Resolution 1 µs • TM15: Typical resolution 125 µs (DRIVE-CLiQ cycle clock) • SIMOTION D410-2 (DI/DO 8 to 15): Typical resolution 125 µs • ET 200MP TM timer DIDQ 16x24V: Resolution 1 µs • ET 200SP TM timer DIDQ 10x24V: Resolution 1 µs
High-speed digital output (DO)	The cam output is performed via onboard outputs of the SIMOTION CPU. The output is via a hardware timer and the cam output is achieved with a resolution with respect to time < servo cycle. Supported hardware: • SIMOTION D410 (terminal X121) • SIMOTION D4x5 (terminal X122, X132) • SIMOTION C240, C240 PN (terminal X1)
Standard digital output (DO)	The output cam calculations are performed in the processing cycle (IPO or IPO2 cycle or in the servo cycle). Actual cam output is performed in servo cycles. The resolution with respect to time of the cam output is generally reduced by the output cycle of the I/O used. The resolution is therefore dependent as follows: • For the standard I/O (e.g. ET 200), on the cycle time of the bus system (PROFIBUS DP / PROFINET I/O) • For the TM15 / TM17, on the cycle time of the bus system (PROFIBUS Integrated / PROFIBUS DP / PROFINET IO) • For the TM15 DI/DO, TM31, TM41, TB30, on the configured sampling time: – cu.p0799 (CU inputs/outputs sampling time) for the onboard outputs – p4099 (TMxx inputs/outputs sampling time) for TB30, TM15 DI/DO, TM31, and TM41 Supported hardware: • Onboard outputs (SIMOTION D, Controller Extension CX, SINAMICS Control Unit CU3xx) • Centralized I/O (SIMOTION C) • Distributed I/O via PROFIBUS DP / PROFINET IO (e.g. ET 200, etc.) • Drive-related I/O (TM15, TM15 DI/DO, TM17 High Feature, TM31, TM41, TB30)

Procedure

To achieve the best possible output cam resolution on the onboard I/Os of a SIMOTION D410-2, activate the output and select "Cam output on output cam output (CAM)."

Then assign a hardware output. For this purpose, click ... ("assign") to open the Assignment dialog box and select a free (i.e. not yet used) I/O.

Note

Only those I/Os that have the appropriate functionality (DO_xx [channel name, terminal number]) are displayed. If no suitable I/Os are displayed, you must first configure the I/Os (I/O must be configured as "output cam (CAM)")

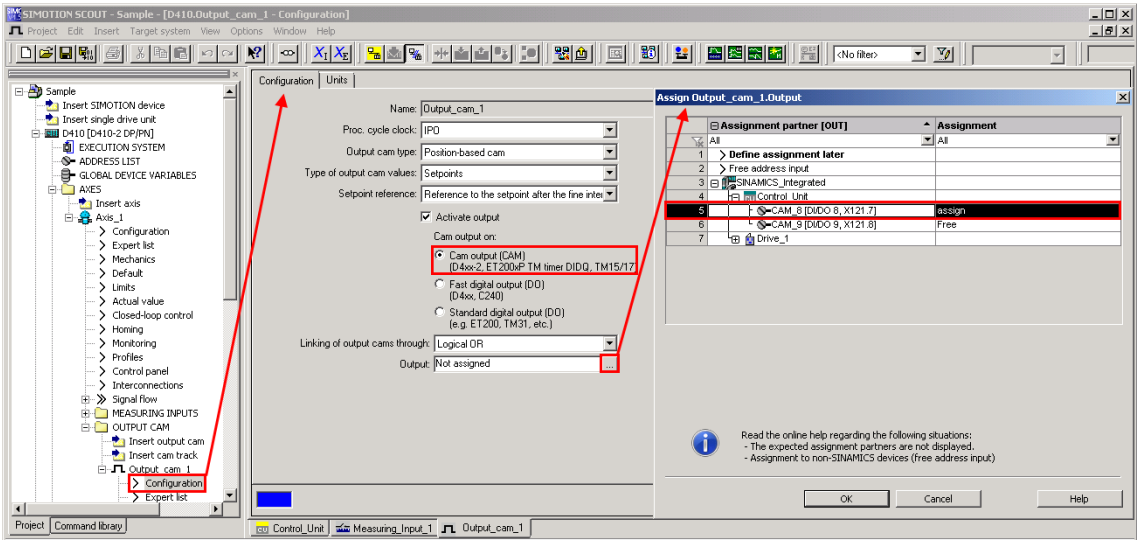


Figure 10-202 Configuration of a cam for SIMOTION D410-2

A maximum of two edges can be output per processing cycle clock of TO outputCam or TO camTrack.

Additional references

Detailed information on configuring the output cam and cam track technology objects can be found in the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Configuring an I/O variable

Overview

You have two ways to assign an I/O variable to I/O terminals:

- Assignment via preferred interconnection (e.g. DI_8 [DI/DO 8, X121.7])
  To do this, you must use the SIMOTION preferred interconnection for the corresponding input/outputs of the SINAMICS DOs. The BICO interconnection is performed automatically.
- Assignment via PZD (e.g. via DI_0_15 or DO_0_15).
  Note that for these signals a message frame of the appropriate length is generated, but the BICO interconnection is not performed.

Interconnection via preferred interconnection

The I/O variables are configured via the address list. Components that support a symbolic assignment can be configured without I/O addresses.
Preferred interconnections are displayed as assignment targets in the Assignment dialog box (e.g. DI_8 [DI/DO 8, X127.7]). The assignment is made by direct selection of the corresponding terminal signal.

Components that do not support symbolic assignment (e.g. standard PROFIBUS I/O) are configured via I/O addresses.

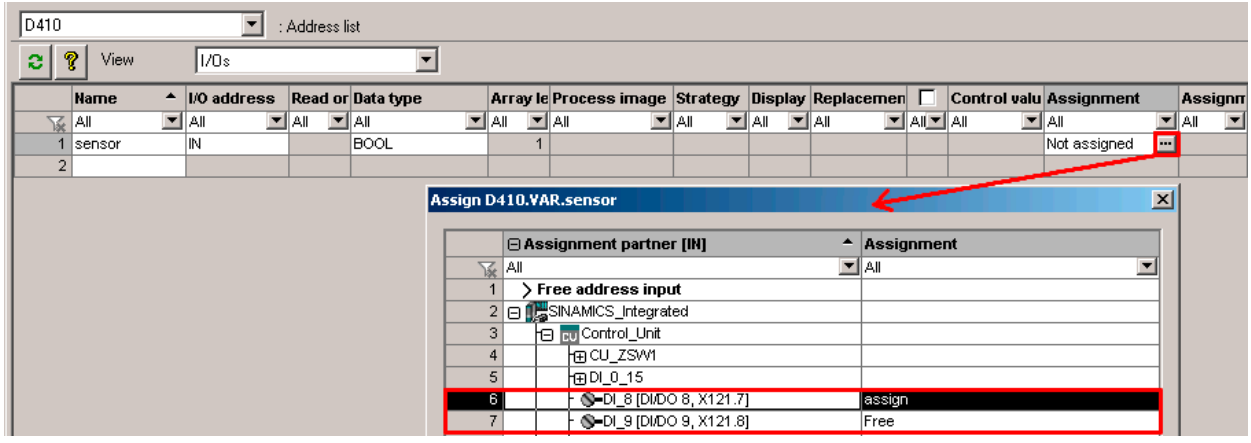


Figure 10-203 Address list

Interconnection via PZD

In principle, an assignment via the PZD is also possible (e.g. via DI_0_15 or DO_0_15). Note that for these signals a message frame of the appropriate length is generated, but the BICO interconnection is not performed.

For this purpose, switch to the "Communication" dialog box of the corresponding SINAMICS DO. You can now see the individual bits of the PZD (e.g. I_Digital or O_Digital) listed there. Interconnect the appropriate bit of the PZD with a signal.

Alternatively, you can assign SIMOTION to an I/O channel during the terminal configuration (e.g. by selecting "DI (SIMOTION)", see Section SIMOTION D410-2 Configuring I/Os (Page 7471)).

TO axis

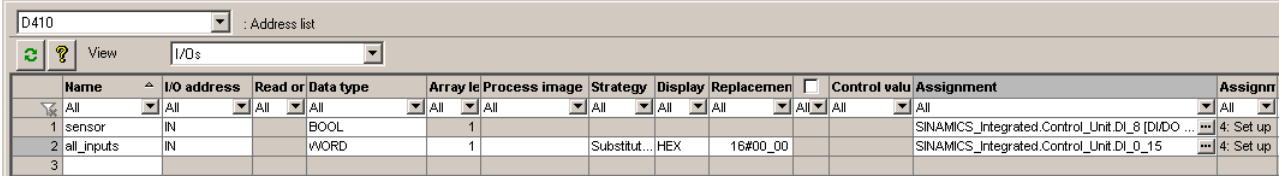
The symbolic assignment of I/Os is also supported by the TO axis (e.g. for a HW limit switch).

Substitute values for I/O variables

Substitute values cannot be specified for input variables of the BOOL data type. If you do require substitute values however, proceed as follows:

1. Assign a digital input (e.g. SINAMICS_Integrated.Control_Unit.DI_8 [DI/DO 8, X121.7]) to an input variable of the BOOL type (e.g. sensor).
2. Create a global variable (e.g. all_inputs) (at least data type WORD, e.g. SINAMICS_Integrated.Control_Unit.DI_0_15).
3. Configure the substitute value
   The corresponding bit of the substitute value must then contain the substitute value for the BOOL variables.
   In the same way, you can assign a substitute value to a BICO parameter.

For various SINAMICS drive objects, higher-level types are available for the assignment of substitute values.



	Name	I/O address	Read or	Data type	Array le	Process image	Strategy	Display	Replacemen		Control valu	Assignment	Assignm
	All	All	All	All	All	All	All	All	All	All	All	All	All
1	sensor	IN		BOOL	1							SINAMICS_Integrated.Control_Unit.DI_8 [DI/DO ...	4: Set up
2	all_inputs	IN		WORD	1		Substitut...	HEX	16#00_00			SINAMICS_Integrated.Control_Unit.DI_0_15	4: Set up
3													

Figure 10-204 Configuration of substitute values

Creating a DMC20/DME20 DRIVE-CLiQ hub

Hub properties

DRIVE-CLiQ hub characteristics

The DMC20 and DME20 DRIVE-CLiQ hub modules are used to implement point-to-point distribution of a DRIVE-CLiQ line. With the DMC20/DME20, an axis grouping can be expanded with 4 DRIVE-CLiQ sockets for additional subgroups.

- DMC20 is the hub for the control cabinet configuration
- DME20 is the hub for use without a control cabinet (IP67 degree of protection).

The modules are especially suitable for applications which require DRIVE-CLiQ nodes to be removed in groups, without interrupting the DRIVE-CLiQ line and therefore the data exchange.

Application examples

Encoder expansion and hot-plugging are typical applications implemented by means of a DRIVE-CLiQ hub.

- In an encoder expansion, direct measuring systems are connected. For example, these are attached directly to the machine in the control cabinet. Several encoders can be connected to one hub in the cabinet.

  Note

  The SIMOTION D410-2 has just one DRIVE-CLiQ interface. You can use the DMC20/DME20 to evaluate the motor encoder and an additional encoder by means of SMx. The DRIVE-CLiQ hub must be directly connected to the Control Unit.

- Hot plugging is the option for changing motor components while in operation. These components are connected to a star topology using a DRIVE-CLiQ hub. This setup allows their deactivation without impairing downstream components.

Additional references

Additional information on the DMC20/DME20 DRIVE-CLiQ hub is contained in the

- *SIMOTION D410-2* Manual
- *SINAMICS S120 Control Units and Supplementary System Components* Manual

Creating a DRIVE-CLiQ hub

Introduction

You can directly insert a DMC20/DME20 in the project navigator. The hub is not wired when you insert the DMC20/DME20 and is displayed in the topology tree in the component storage. The hub has to be wired manually. Proceed as follows:

Procedure

1. Right-click the "Topology" entry in the Project Navigator.
2. Select the "Insert new object" > "DRIVE-CLiQ Hub" command from the shortcut menu and confirm with "OK".
3. Double-click "Topology" to open the topology tree.
   The hub is saved to the component storage of the topology tree.
4. Drag-and-drop the hub to the required DRIVE-CLiQ interface.
   The components connected to the hub are displayed in the topology tree.

Result

The hub you inserted is displayed as an icon at the "Topology" entry in the Project Navigator. All components connected to a hub are also displayed in the course of automatic configuration.

Creating and programming TM41

TM41 properties

The TM41 terminal module can be used to expand the number of digital I/Os and analog inputs within a drive system. TM41 also returns TTL signals which emulate an incremental encoder, for example, for a master control system.

The emulated encoder signal has the signal characteristic of an incremental TTL encoder (A track, B track, R track). The resolution of the encoder signal can be specified in the configuration.

Note

The digital I/Os and the analog input can be interconnected using the BICO configuration.

The TM41 encoder interface (incremental encoder representation) can

- Be interconnected with a Control Unit encoder signal (from sin/cos incremental encoders, for example), using parameterization. For detailed information, consult the SINAMICS manuals.
- From the SIMOTION viewpoint, be accessed as an axis. This allows you to make the axis position (a master value) available to a second controller as an encoder signal, for example.

Configuring the TM41 involves the following steps:

- Configuring TM41 at SINAMICS Integrated (Page 7481)
- Configuring the TM41 using the axis wizard (Page 7482)

Configuring TM41 at SINAMICS Integrated

Procedure

The TM41 can be configured after you completed the configuration of SINAMICS Integrated. Proceed as follows:

1. Double-click "Insert input output component" at "Input/output component" in the Project Navigator.
2. Select the TM41 from the "Drive object type" field of the "Insert Input/Output Component" dialog box and assign a unique name to the module.
3. Confirm your entry with "OK".

Result

The TM41 is inserted in the Project Navigator by the name you entered.

Configuring the TM41 using the axis wizard

Requirement

After configuring the TM41 for a SINAMICS Integrated device in the project navigator, you can interconnect it with an axis using the Axis Wizard. The Wizard implements the TM41 as drive device.

Procedure

1. Open the Axis Wizard and create a positioning or synchronization axis (electrical).
2. Step the Axis Wizard forward until the "Drive Assignment" dialog box opens.
3. Select "SINAMICS_Integrated" as the drive device and "TM41" as the drive.
   TM41 operates as setpoint sink of the axis with this setup.

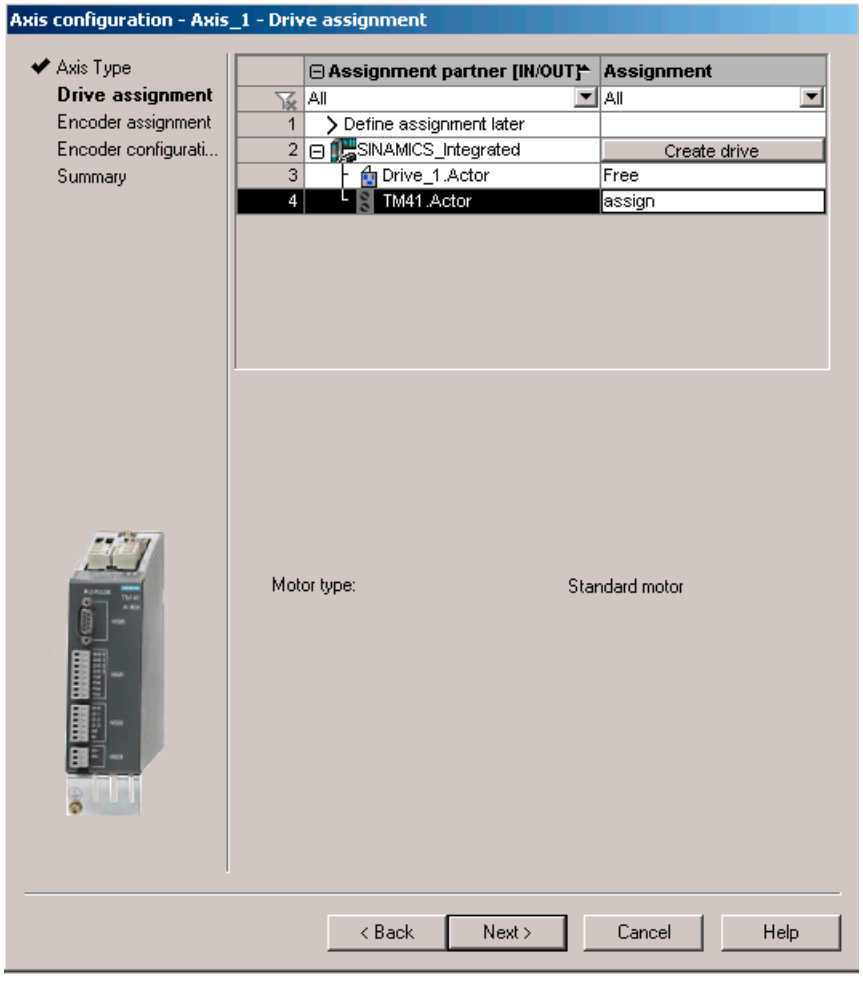


Figure 10-205 Drive assignment

4. Work through the remainder of the Axis Wizard.

Reference

Detailed information on configuring incremental encoder emulation with the TM41 can be found in:

- FAQs at the following
  Internet address (https://support.industry.siemens.com/cs/ww/en/view/27554028)
- SIMOTION Utilities & Applications
  SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Optimizing the drive and controller

Overview of automatic controller setting

Overview

The SIMOTION SCOUT engineering system offers a wizard for the automatic controller setting for the controller optimization of the drive.

In the "Automatic Controller Setting" screen form, you can configure an automatic setting for the speed controller and the DSC (dynamic servo control) position controller for SINAMICS drive units. The necessary steps for this calculation can be controlled from this screen form. The calculated parameter values for the speed controller or position controller are displayed and can then be transferred online to the drive or axis on the controller.

The automatic control setting is made from the "Target system" > "Automatic control setting" menu.

A detailed description of the settable parameters can be found in the *SIMOTION SCOUT* Online Help.

Requirements

- You have configured a SINAMICS drive.
- The configured drive is operated in the "Servo" drive object type.
- Closed-loop control takes place with the motor encoder.
- There is an online connection to the relevant drive unit.

Procedure

Automatic controller setting involves the following steps:

1. Setting the speed controller (Page 7484)
2. Setting the position controller (Page 7485)

Note

You can cancel the automatic controller setting by pressing the SPACEBAR.

- The step currently being executed is aborted.
- The drive enable is canceled.

Additional references

Information on the controller structure can be found in the *SIMOTION TO Axis, Electric/ Hydraulic, External Encoder* Function Manual.

In addition to the automatic controller setting, SIMOTION SCOUT also offers the option to optimize the drive and controller manually using the measuring functions, trace, and function generator.

See also

Measuring functions, trace, and function generator (Page 7486)

Manual speed controller optimization (Page 7488)

Automatic speed controller setting

Characteristics

The automatic speed controller setting has the following features:

- Damping of resonances in the speed-controlled system
- Automatic setting of the Kp gain factor and the Tn reset time of the speed controller
- The speed setpoint filter and the reference model are not changed.

Procedure

To perform an automatic setting of the speed controller, proceed as follows:

1. Select the "Target system" > "Automatic controller setting" menu command.
2. Select the drive unit and the drive.
3. Select the "Speed controller" from the "Controller selection".
4. Click "Assume control priority" to assume control priority.
5. Click the "Drive on" button to enable the drive.
   Perform these steps (1 to 4) in automatic mode or as individual steps.
6. Click "Transfer" to transfer the calculated parameter values for the speed controller to the drive.
7. Disable the drive by clicking the "Drive off" button.

8. Click "Give up control priority" to give up control priority of the PG/PC.
9. Save the online parameters.

You can now transfer the automatically set parameters to the project.

Backing up parameters

Proceed as follows to back up the parameters:

1. In the project navigator, select the SINAMICS unit with the drive for which you want to perform the automatic setting.
2. Select "Target device" > "Copy RAM to ROM" in the context menu.
3. Select "Target device" > "Load CPU / drive unit to PG" in the context menu.

If required, the automatic controller settings can be checked using the measuring functions.

Automatic position controller setting

Introduction

In the "Automatic Controller Setting" screen form, you can select the SINAMICS drive unit and the drive for which you want to carry out an automatic DSC position controller setting. The necessary steps for this calculation can be performed from this screen form. The calculated Kv value is displayed and can then be accepted online in the configuration data of the axis that is assigned to the drive.

Requirements

In addition to the General requirements for the automatic controller setting, the following boundary conditions apply for setting the position controller:

- DSC is required for the position controller setting.
  Tip: Activate the project setting "Use symbolic assignment" and select the Standard/ Automatic option for the axis-drive communication when configuring the drives. You automatically use DSC for the servo drives with these settings.
- The speed controller has already been configured (e.g. with the automatic speed controller setting).
- At least one axis is connected to the SINAMICS drive (servo).
- An online connection to the SIMOTION device must be established to transfer the results of the automatic position controller setting.
- The balancing filter is not changed.
- For operation without precontrol, the equivalent time constant of the position controller must be adjusted manually by the user (PositionTimeConstant = 1/Kv).
- Vibration on the load side is not taken into account for the position controller setting.

Procedure

To perform an automatic setting of the position controller, proceed as follows:

1. Select the "Target system" > "Automatic controller setting" menu command.
2. Select the drive unit and the drive (axis).
3. Select the "Position controller (DSC)" from "Controller selection".
4. Click "Assume control priority" to assume control priority.
5. Click the "Drive on" button to enable the drive.
   Perform the steps either in automatic mode or as individual steps.
6. Select the axis data sets to which the Kv factor is to be transferred.
7. Click "Accept values" to transfer the calculated Kv factor to the axis data sets.
8. Disable the drive by clicking the "Drive off" button.
9. Give up the control priority of the PG/PC.
10. Save the online parameters.

You can now transfer the automatically set parameters to the project.

Backing up parameters

Proceed as follows to back up the parameters:

1. In the project navigator, select the SIMOTION unit with the axis for which you want to perform the automatic setting.
2. Select "Target device" > "Copy current data to RAM" in the context menu.
3. Select "Target device" > "Copy RAM to ROM" in the context menu.
4. Select "Target device" > "Load CPU / drive unit to PG" in the context menu.

If necessary, you can check the automatic controller settings using the measuring functions.

Measuring functions, trace, and function generator

Drive optimization

Drive optimization is part of commissioning and can be performed with SIMOTION SCOUT.

Note

Controller optimization may only be performed by skilled personnel with control engineering knowledge.

Controller optimization

Various measuring functions are available for controller optimization of the drive. These measuring functions enable the control of the higher-level control loop to be selectively switched off and the dynamic response of individual drives to be analyzed through simple parameter assignment. The function generator and the trace recorder are used.

The control loop is supplied with the ramp-function generator signal at a specific point (e.g. speed setpoint), and the signal from the trace recorder is recorded at another point (e.g. speed actual value).

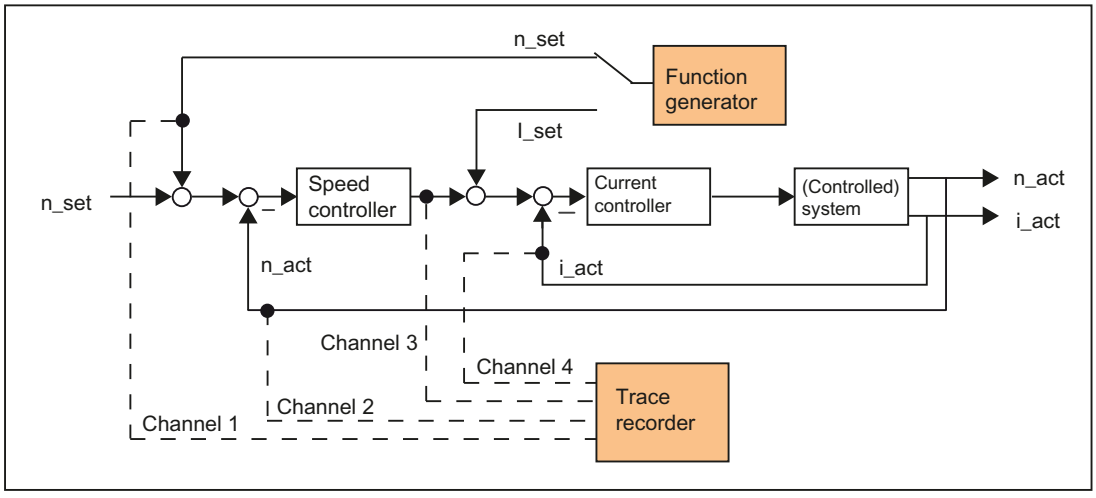


Figure 10-206 Optimizing a controller

Depending on the form of controller optimization to be performed, it is possible to define the quality (e.g. signal form, amplitude, transient recovery time) of the disabled signal, the measuring duration for step functions in the time range, or the bandwidth and number of averaging operations in the frequency range for the trace. The analytical and graphical evaluation can then be performed accordingly (FFT diagram, Bode diagram).

The following measuring functions are available:

- Setpoint jump at current controller
- Reference frequency response at current controller
- Setpoint jump at speed controller
- Disturbance variable jump at speed controller
- Reference frequency response at speed controller
- Disturbance frequency response at speed controller
- Speed-controlled system (input at current setpoint filter)

Additional references

For additional information about drive optimization, consult the *SINAMICS S120* Commissioning Manual.

Additional information on trace and measuring functions, as well as on the function generator, can be found in the *SIMOTION SCOUT* online help.

Manual speed controller optimization

Requirement

You have already created a project and configured an axis and a drive. You can now optimize the speed controller.

Procedure

1. Open the project and go to online mode.
2. Click to call the "Measuring Functions" dialog.
3. Select the drive unit and the drive.
4. Select "Speed controller setpoint jump".
   You can change the values in the following fields: "Settling time", "Amplitude", "Offset", "Ramp-up time" and "Measuring time".

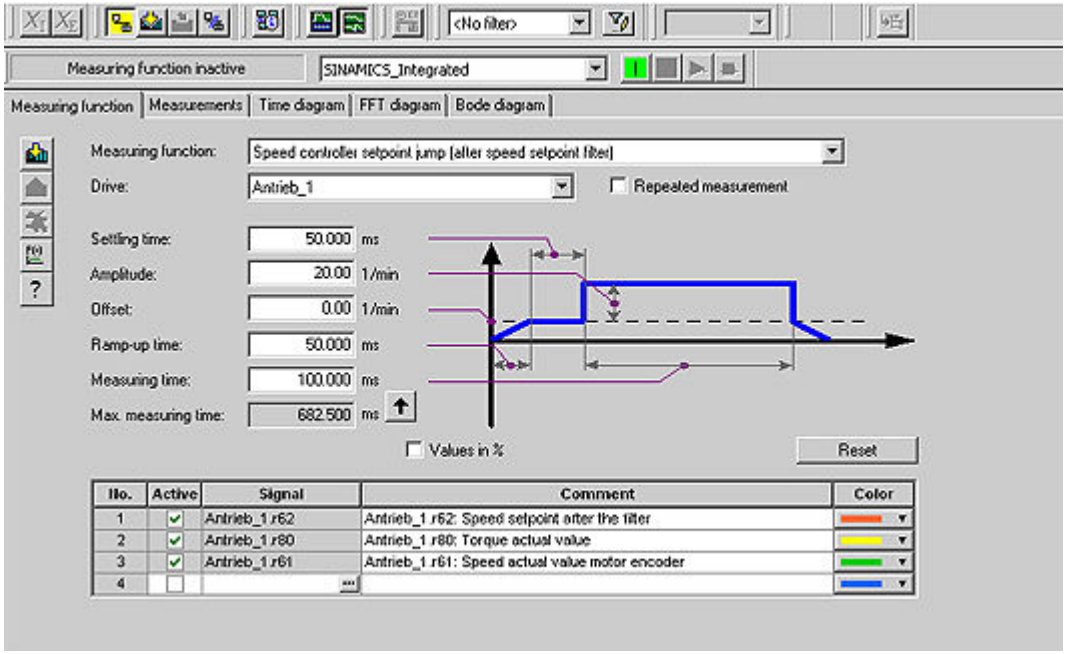


Figure 10-207 Speed controller measuring function

Four channels can be traced. Certain channels are preassigned, depending on the measuring function.

5. Download the changes to the drive by clicking (Download parameter assignment).

Starting the measuring function

1. Click "Assume control priority" to assume control priority.
   Read the notice that appears and click "Accept" to confirm. The activated service function is displayed via the LEDs (RUN/STOP flashing yellow/green with 2 Hz).
2. Click the "Drive on" button to enable the drive.

3. Click (Start measuring function) to start the measuring function.
   The axis is moved during the measurement. For this reason, a safety message that allows the process to be aborted is displayed.
4. The traced signals are represented on the "Time diagram" tab.

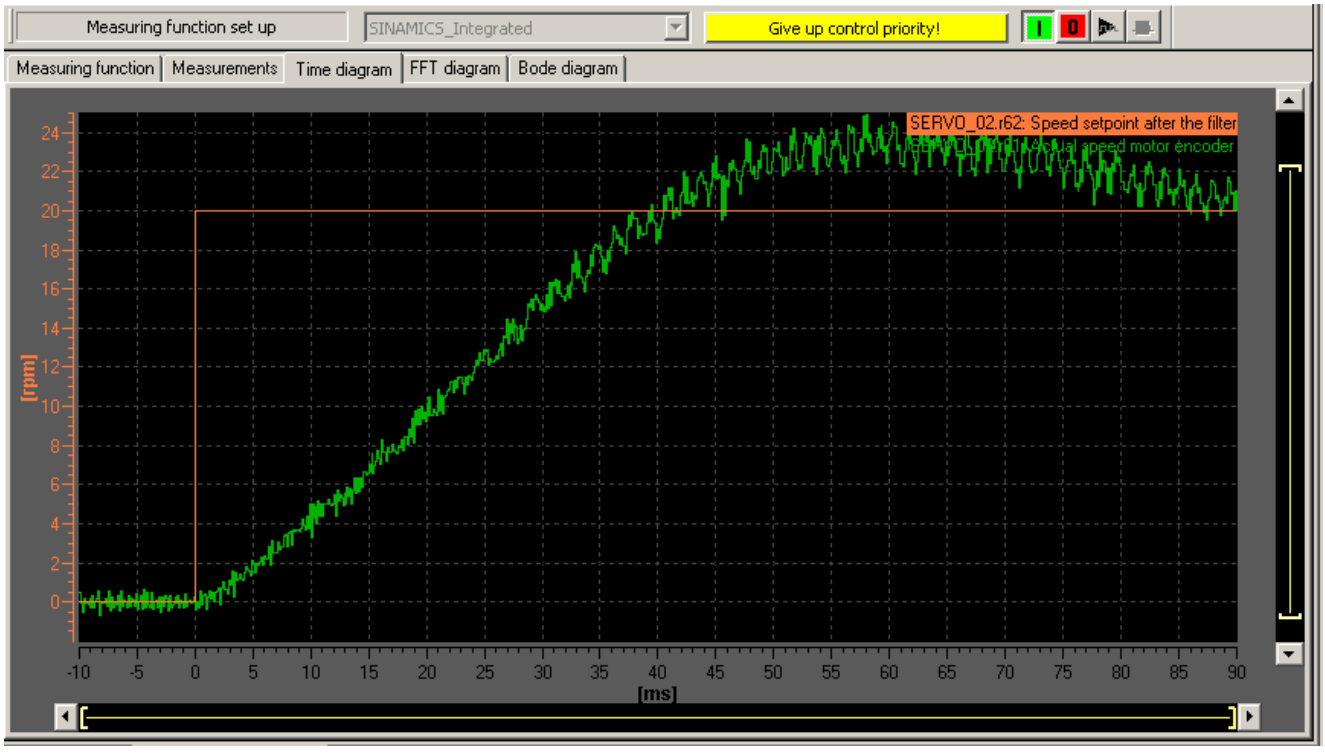


Figure 10-208 Time diagram before parameter change

Adjusting the P-gain

You can adjust the P-gain of the controller to optimize the transient response.

1. In the project navigator under the corresponding drive, for example, Servo_1, use the menu command "Open-loop/closed-loop control" > "Speed controller" to display the "Speed Controller with Encoder" dialog box.
2. Enter an appropriate value in the "P-gain" field and the "Integral time" field.

Note

The values entered take immediate effect.

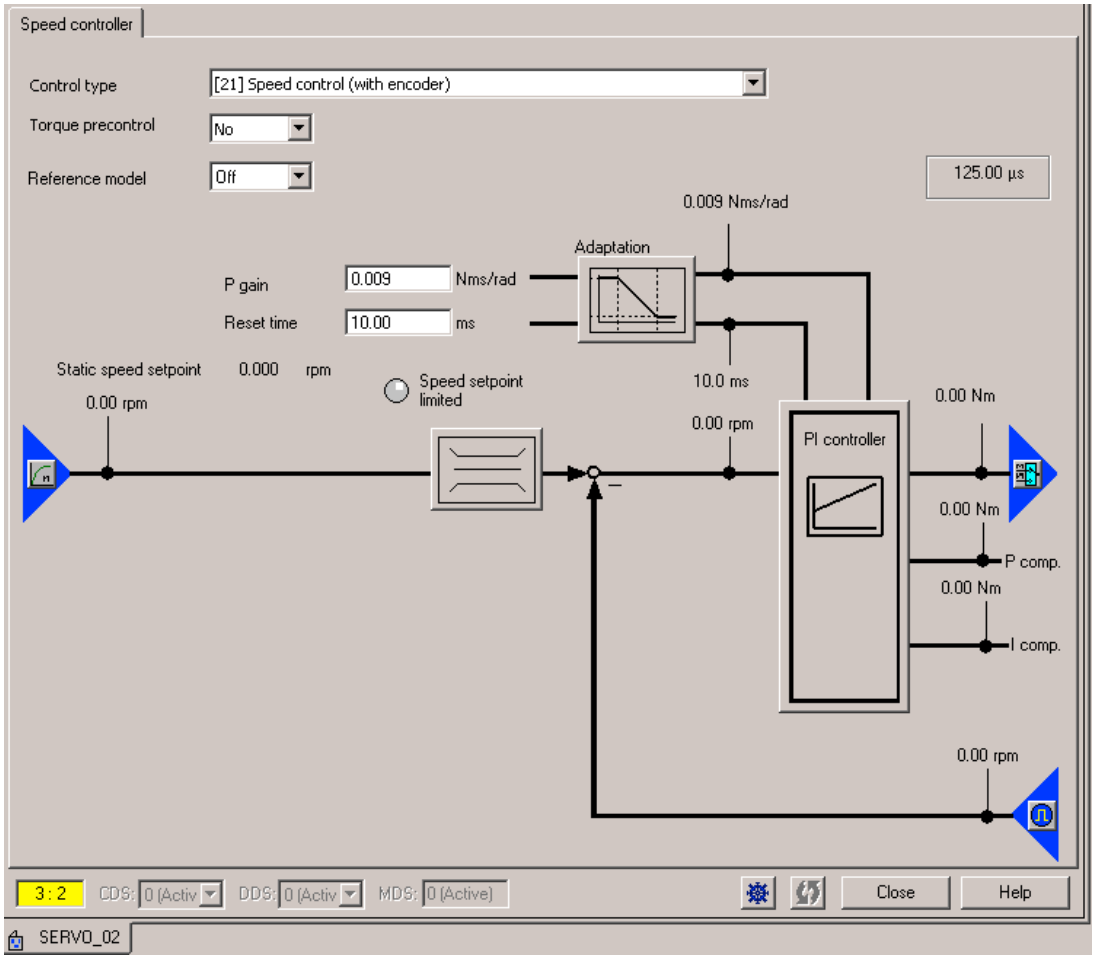


Figure 10-209 Entering P-gain

3. For verification purposes, perform the measurement again.
4. With the modified parameters, the controller displays a much better transient response. If necessary, you can continue changing the value until the transient response is optimal.

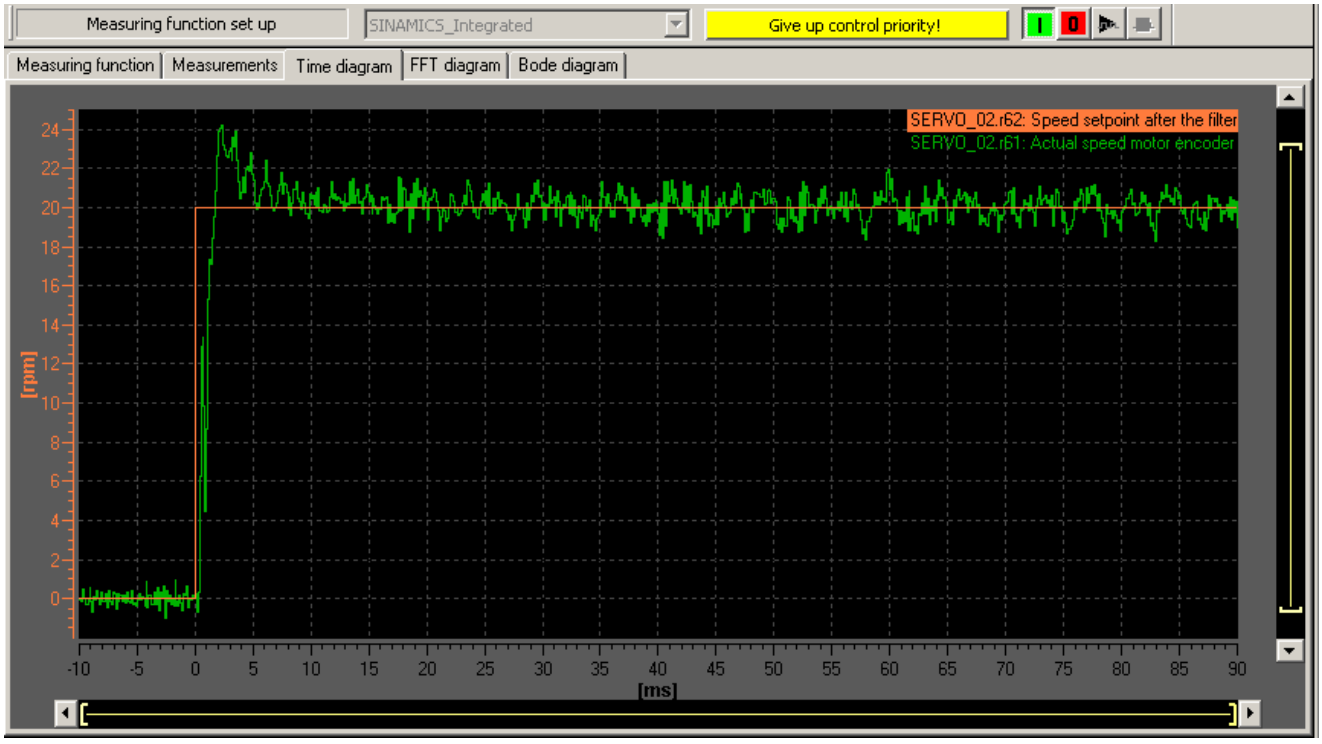


Figure 10-210 Measurement with modified P-gain

Loading and saving SIMOTION user data

Overview

After commissioning the SIMOTION D410-2, we recommend that you back up the SIMOTION user data (programs, configuration data and parameter assignments) on the CF card.

Loading user data

Use the "Target system" > "Load" > "Load to target system" command to transfer the following data from the SIMOTION SCOUT engineering system (ES) to the "non-volatile SIMOTION data" area of SIMOTION D410-2:

- Configuration data
- Programs
- Parameterization
- Technology packages

The hardware configuration of the SIMOTION D410-2 and the retain variables are also stored in the "non-volatile SIMOTION data" area.

Note

Using the menu:

- "Target system" > "Load" > "Load project to target system" loads all of the project data to the target system.
- "Target system" > "Load" > "Load CPU / drive unit to target device" only loads the data of the selected device / drive element to the target device.

After the SIMOTION D410-2 is switched off, the contents of the "volatile SIMOTION data" area are lost.

Additional references

Additional information about the SIMOTION SCOUT engineering system can be found in the *SIMOTION SCOUT* Configuration Manual.

Saving user data

The "Copy RAM to ROM" function is used in SIMOTION SCOUT to save the following data from RAM to the CF card:

- Technology packages and user data (units, configuration data, parameter assignments, task configuration) from the "volatile SIMOTION data" area
- Current data values are copied to the "volatile SIMOTION data" area, depending on the settings in SIMOTION SCOUT.

  Note

  The "Copy RAM to ROM" command **cannot** be used to save the current values of retain variables to the CF card.

You have the following options for backing up the current values of retain variables to the CF card:

- User program
  Use the "_savePersistentMemoryData" system function in the user program.
- Save with the service selector switch or the DIAG button on the SIMOTION D410-2 or using SIMOTION IT web server, see Section Diagnostic data and non-volatile SIMOTION data (Page 7557)

The SCOUT functions "Save variables" and "Restore variables" also give you the option of backing up to your PC and restoring data that was changed during operation and only stored in the runtime system.

Execute the "Copy RAM to ROM" function separately for the SINAMICS Integrated. This requires that the drive element has been selected in the Project Navigator.

See also

Properties of the user memory (Page 7352)

Deleting data

Overview of data deletion

The memory of the SIMOTION D410-2 described in the User memory concept (Page 7351) can be deleted in various granualarities. That is, you can choose to delete all data or only specific parts.

You have the following means of deleting the SIMOTION D410-2 data:

- SIMOTION D410-2 memory reset (Page 7493)
- Deleting user data on CompactFlash card (Page 7496)
- Setting SINAMICS Integrated to the factory settings (Page 7497)
- Restoring the default settings for the SIMOTION D410-2 (Page 7497)
- Deleting/restoring non-volatile SIMOTION data (Page 7564)
- Backing up / restoring / deleting SINAMICS NVRAM data (Page 7437)

SIMOTION D410-2 memory reset

Introduction

During the memory reset, the memory of the SIMOTION D410-2 and the non-volatile SIMOTION data in the NVRAM with the exception of the communication configuration (baud rates, network addresses, etc.), are deleted. The data on the CF card is retained during the memory reset.

Reset the memory on the SIMOTION D410-2 if

- If you want to undo changes made to user data (programs, configuration data, parameter assignments) which you have not backed up by means of the "Copy RAM to ROM" command.
- The RUN/STOP LED is flashing (slowly flashing yellow) to indicate that the SIMOTION D410-2 is requesting a memory reset.
- If the non-volatile SIMOTION data with the project on the CF card do not match and an error occurs (diagnostic buffer entry).

You can perform a memory reset either in offline mode using the mode switch of the SIMOTION D410-2 or in online mode using the SIMOTION SCOUT.

Data deleted on memory reset

The following data are deleted during a memory reset:

- User data (units, configuration data, parameter settings, task configuration)
- Technology packages

- Retain TO (absolute encoder adjustment)
- Retain variables
  Retain variables are variables in the interface or implementation section of a UNIT that are declared with VAR_GLOBAL RETAIN, or global device variables with the RETAIN attribute.

Note

Absolute encoder data are deleted during a memory reset and must therefore be readjusted after the memory reset.

Data retained during memory reset

The following data are retained during a memory reset:

- TCP/IP and DP parameters
- Diagnostic buffer
- Data that was saved with the _savePersistentMemoryData, _saveUnitDataSet or _exportUnitDataSet commands and with the "Copy RAM to ROM" function.
  If backup files (PMEMORY.XML/PMEMORY.BAK) have been created with _savePersistentMemoryData, the data in these files will be backed up again to the non-volatile SIMOTION data after the memory reset. Users can therefore force the restoration of non-volatile SIMOTION data by means of memory reset. This also includes the absolute encoder position.
- Licenses
- SINAMICS NVRAM data

The technology packages and user data (configuration data, programs, parameter assignments) that were previously backed up on the CF card using the "Copy RAM to ROM" menu command will be transferred to the "non-volatile SIMOTION data" area of the SIMOTION D410-2 during the next power-up. Thus, an existing configuration on the CF card is loaded to the SIMOTION device following the memory reset.

Memory reset by means of SIMOTION SCOUT

The SIMOTION device for which overall reset is to be performed must be online.

1. Open the "Control Operating State" dialog box in SIMOTION SCOUT. Select the "Target system" > "Control operating state" menu command. Or click on "Control Operating State" on the toolbar.
2. Switch the SIMOTION device for which memory reset is to be performed to STOP in the "Control Operating State" dialog box.
3. Select the SIMOTION device under "Memory reset (MRES)." Click the "Run" button. Confirm the command by clicking "Yes."
   The memory reset will now be performed.

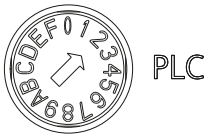
Memory reset with the mode switch

You can perform a memory reset with the mode switch when you are offline with the SIMOTION D410-2.

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Comply with the ESD rules.

Proceed as follows to reset the memory:

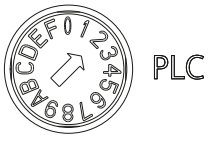
1. Place the mode switch in the STOP position (selector setting 2).



2. When the RUN/STOP LED is steadily yellow, turn the switch to the MRES position (switch position 3).

   The RUN/STOP LED starts to flash slowly (slow yellow flashing).
   Wait until the RUN/STOP LED stops flashing.

3. Turn the selector back to the STOP setting.



4. You must turn the selector back to the MRES position again within 3 seconds.

The memory reset will now be performed.
The SIMOTION D410-2 has completed the memory reset when the RUN/STOP LED is steadily yellow.

Note

Memory reset is canceled if you do not return the mode switch to MRES as specified within 3 seconds. Repeat the procedure in this case.

5. Now move the mode switch back to the required operating mode.

NOTICE
Unintentional restoration of the default settings instead of memory reset
Note that the MRES position (switch setting 3) during power-up causes the default settings to be restored. See Section Restoring the default settings for the SIMOTION D410-2 (Page 7497).
Make sure you do not accidentally switch the power supply OFF/ON in the MRES selector setting, as this restores the default settings instead of performing the desired memory reset.

Deleting user data on CompactFlash card

Overview

Deletion of the user data from the CompactFlash card is necessary, for instance, if you wish to load another (new) project to the CompactFlash card and therefore find it necessary to delete the existing user data for an "old project" (e.g. unit data sets) from the CompactFlash card.

You can delete the user data with SIMOTION SCOUT. The SIMOTION D410-2 must be operating in online mode. The following data is deleted:

- User data from the "volatile data" area
- Non-volatile data, with the exception of IP and DP parameters
- User data on the CompactFlash card (user directories), including the SINAMICS configuration

You can still go online to SIMOTION D410-2 at your PG/PC. The licenses on the CompactFlash card are retained.

Procedure

1. Open the project you want to edit in SIMOTION SCOUT.
2. Go online with the SIMOTION D410-2.

3. Select SIMOTION D410-2 in the Project Navigator and then select the "Delete user data on card" option from the "Target system" menu.
4. Confirm the "Delete user data on card" prompt with "OK".
   The user data is deleted. SINAMICS Integrated goes into offline mode.

Setting SINAMICS Integrated to the factory settings

Requirement

You must be online to SINAMICS Integrated in order to restore its default settings.

Procedure

1. Right-click "SINAMICS_Integrated" in the Project Navigator.
2. Select the "Target device > Restore default settings" command from the shortcut menu.

This restores the delivery state of SINAMICS Integrated.

Restoring the default settings for the SIMOTION D410-2

Overview

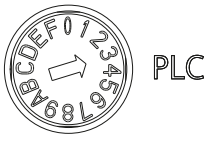
SIMOTION D410-2 is supplied with default parameters such as the transmission rate or PROFIBUS addresses. You can restore the factory settings with the mode selector switch. The following data is deleted:

- Non-volatile SIMOTION data in the SIMOTION device
- The backup copy of non-volatile SIMOTION data on the CompactFlash card (PMEMORY.XML/ PMEMORY.BAK)
- User data in the "volatile SIMOTION data" area and on the CompactFlash card
- The communication configuration (IP and DP parameters) on the CompactFlash card is set to the factory settings.

All licenses on the CompactFlash card are retained.

Restoring factory settings with the mode selector switch

1. The power supply is switched off.
2. Set the mode selector switch on the SIMOTION D410-2 to MRES (switch position 3).



3. Switch on the power supply.
   The NVRAM and the user data is deleted. The factory settings are loaded. SIMOTION D410-2 remains in the STOP operating state.
4. Now use the mode selector switch to change to the desired operating state.

Note

The default communications parameters are now restored. You must re-configure the communication parameters of SIMOTION D410-2.

System shutdown

All axes and system components must be in safe state before you shut down the system. You can set up this safe state by providing a separate motion task, for example.

You can shut down the power supply after the system has reached a standstill state.

Note

You must observe the safety notices for SINAMICS components, which you can find in the corresponding SINAMICS manuals.

Configuring Safety Integrated functions

Overview

Integrated safety functions

When used in conjunction with SIMOTION D, the integrated safety functions of SINAMICS S120 provide highly effective practical protection for personnel and machinery.

- Safety Integrated Basic Functions
- Safety Integrated Extended Functions
- Safety Integrated Advanced Functions

Note

As of SIMOTION V5.2 / SINAMICS Integrated V5.1, a distinction is made between Safety Integrated Extended and Advanced Functions. The Safety Integrated Advanced Functions also include the Safety Integrated Extended Functions.

The safety functions listed here conform to:

- Safety Integrity Level (SIL) 2 according to IEC 61508
- Category 3 according to DIN EN ISO 13849-1
- Performance Level (PL) d according to DIN EN ISO 13849-1

The safety functions correspond to the functions according to DIN EN 61800-5-2 (if they are defined there).

Übersicht der Safety Integrated Functions

If motors without a (safety-capable) encoder are being used, not all Safety Integrated Functions can be used.

Note

Definition: "Without encoder"

When "without encoder" is used in this manual, then this always means that either no encoder or no safety-capable encoder is being used.

In operation without encoder, the actual velocity values are calculated from the measured electrical actual values. Therefore, velocity monitoring is also possible during operation without encoder.

Table 10-124 Overview of the Safety Integrated Functions

	Functions	Abbreviation	With encoder	Without encoder	Brief description
Basic Functions	Safe Torque Off	STO	Yes	Yes	Safe torque off
	Safe Stop 1	SS1 [1)]	Yes	Yes	Safe stop according to stop category 1
	Safe Brake Control	SBC [2)]	Yes	Yes	Safe brake control

	Functions	Abbreviation	With encoder	Without encoder	Brief description
Extended Functions	Safe Torque Off	STO	Yes	Yes[4]	Safe torque off
	Safe Stop 1	SS1[1]	Yes	Yes[4]	Safe stop according to stop category 1
	Safe Brake Control	SBC	Yes	Yes[4]	Safe brake control
	Safe Operating Stop	SOS	Yes	No	Safe monitoring of the standstill position
	Safe Stop 2	SS2	Yes	No	Safe stop according to stop category 2
	Safely-Limited Speed	SLS	Yes	Yes[5][6]	Safe monitoring of the maximum velocity
	Safe Speed Monitor	SSM	Yes	Yes[5][4]	Safe monitoring of the minimum velocity
	Safe Direction	SDI	Yes	Yes[5][4]	Safe monitoring of the direction of motion
	Sicheres Referenzieren	SR	Yes	No	Safe homing
	Safe Acceleration Monitor	SAM	Yes	Yes[4]	Safe monitoring of the drive acceleration
	Safe Brake Ramp	SBR	Yes	Yes[4]	Safe braking ramp
	Safe gear change	–	Yes	No	–
	Safely-Limited Acceleration	SLA[3]	Yes	No	Safely-limited acceleration
	Safe Brake Test	SBT	Yes[5]	No	Safe test of the required holding torque of a brake
Advanced Functions	Safely-Limited Position	SLP	Yes[5]	No	Safely-limited position
	Transfer of safe position values	SP	Yes[5]	Yes[4]	Transfer of safe position values
	Safe Cam	SCA[3]	Yes	No	Safe cams

1) Incl. SS1E (SS1 with external brake). In this variant of SS1, SIMOTION is responsible for the stop response.

2) For SBC, you also need a Safe Brake Relay for SIMOTION D410-2 in conjunction with a Power Module in blocksize format (PM340, PM240-2). A Safe Brake Adapter is required for Power Modules in chassis format.

3) As of SIMOTION V5.2 / SINAMICS Integrated V5.1

4) The use of this safety function without encoder is permitted only for induction motors or synchronous motors of the SIMOTICS A-1FU (previously: SIEMOSYN).

5) As of SIMOTION V4.4 / SINAMICS Integrated V4.7. (SBT: As of V4.4 HF4)

Note

Details on the Safety Integrated Functions, such as information on the configuration of safety functions as well as operating conditions for the encoderless operation can be found in the *SINAMICS S120 Safety Integrated* Function Manual.

Notes

Parameterize the desired Safety Integrated functions and the monitoring with or without encoder and activate them in the safety screen forms of the SIMOTION SCOUT engineering system.

If motors are used without an encoder or with an encoder that is not suitable for Safety Integrated Extended/Advanced Functions, not all Safety Integrated Functions can be used (see previous table, "Without encoder" column).

The safe speed monitoring without encoder also functions at standstill as long as the drive is switched on.

PROFIsafe telegrams

SIMOTION D410-2 supports PROFIsafe telegrams 30, 31, 901, 902 and 903. (902 as of SIMOTION V4.4; only in conjunction with the TIA Portal; 903 as of SIMOTION V5.1 for Safe Cam, SCA).

Telegrams 31, 901, 902 and 903 enable the states of the F-DIs to be routed through to an F-CPU via PROFIsafe. Telegrams 901 and 902 enable you to apply a factor to SLS limit value 1 that has been set in the drive, allowing the SLS monitoring limit to be changed during operation. Moreover, in telegrams 901 and 902, the safe position actual values communicated in the SINAMICS S120 are transferred via PROFIsafe to the F-CPU (function SP).

With telegram 903, the state of up to 30 cams can be transferred.

Direct fail-safe data exchange via telegram 901 is supported as of SIMOTION V4.3 SP1 HF9.

The "Transfer safe position values via PROFIsafe" function is supported as of SIMOTION V4.4.

Activate Safety Integrated functions

Control

The Safety Integrated functions are fully integrated in the drive system. They can be activated as follows:

- Via the onboard terminals (F-DI, F-DO) on the Control Unit
- Via a PROFIsafe telegram using PROFIBUS or PROFINET
- Via the terminals of a connected TM54F
- As of SIMOTION V4.5, the Safety Integrated Basic Functions STO, SS1 and SBC can also be controlled license-free via the TM54F. (Requirement: STARTER V4.4 SP1; STARTER V4.4 SP1 is not part of SCOUT V4.4 - the functionality is therefore only available as of SCOUT V4.5)
- The SLS and SDI functions can also be activated permanently via parameter assignment.
- The SLA Extended Function can only be activated via PROFIsafe.

The Safety Integrated functions are implemented electronically and therefore offer short response times compared to solutions with externally implemented monitoring functions.

The output signals of the SCA Advanced Function can only be output via PROFIsafe.

Note

Although SIMOTION does not contain any safety-related functionality, it provides support for SINAMICS drives that can perform safety-related functions.

The purpose of this support that SIMOTION offers for the safety-related monitoring functions is to prevent fault reactions at the drive end by ensuring that the drive does not exit the operating state being monitored.

Further information about support of the SINAMICS Integrated functions on the TO axis can be found in the *TO Axis Electric/Hydraulic, External Encoder* Function Manual.

The Safety Integrated Basic Functions via the onboard terminals (F-DI 0) and the Safety Integrated Extended/Advanced Functions (via TM54F or PROFIsafe) can be used simultaneously.

Note

Safety Integrated Extended and Advanced Functions can be controlled via:

- The onboard terminals of the Control Unit or
- PROFIsafe or
- TM54F

Mixed operation is not permitted.

Note

If SIMOTION D410-2 is mounted separately (Power Module in blocksize format connected to SIMOTION D410-2 via CUA31/32), use of the Safety Integrated Extended and Advanced Functions via the onboard terminals (F-DI, F-DO) is not possible.

F-DI and F-DO as standard I/Os

Unused F-DI and F-DO can be used as standard I/Os (one F-DI as two standard inputs, the F-DO as a standard output).

Using F-DI through F-CPU

Prerequisite for use of the F-DI by F-CPU:

- With encoder as of SIMOTION V4.3 SP1 HF3
- Without encoder as of SIMOTION V4.4

The onboard F-DIs can be used as F-DI for the F-CPU if these are not used by SIMOTION D410-2.

In "Extended Functions via PROFIsafe and Basic Functions via onboard terminal" mode, the Basic Functions STO or SS1 (with or without SBC) can be controlled locally. At the same time the routing of F-DI 1 and F-DI 2 for the F-CPU is possible.

Filter setting F-DI

The F-DIs of SIMOTION D410-2 have a parameterizable input filter (drive parameter p10017, debouncing time for digital inputs).

With this input filter, unwanted signals (e.g. fault signals or test pulses of F-DO) can be filtered out so that these do not cause any faults.
Failsafe output modules (e.g. ET 200S F-DO) test the outputs at regular intervals. To achieve this, the F-DO modules switch test signals to their outputs in the form of bit patterns to detect a short-circuit, short-circuit to frame, or a short-circuit to ground.
If the filter time of the F-DI is set too small, these test pulses will result in unwanted triggering of the safety function in the case of D410-2.

For setting the input filter, the following rules must therefore be complied with:

- Debouncing time p10017 ≥ 3 x sampling time inputs/outputs p0799
- If 3 x p0799 < 1 ms, then: Debouncing time p10017 = N x 1 ms (N = 1, 2, 3, ...)
- Debouncing time p10017 >> fault pulse time

Unwanted triggering of the safety function can indicate incorrect settings.
Note that the settings also impact the response time of the system.

Example

ET 200S I/O system with F-DO modules
Because of the test pulses, the debouncing time p10017 must be at least 4 ms.

This results in the following setting options, for example:

- p10017 = 12 ms; p0799 = 4 ms **or**
- p10017 = 4 ms; p0799 = 1 ms

You will find additional information on ET 200S test pulses in Internet (https://support.industry.siemens.com/cs/ww/en/view/44452714).

SI monitoring cycle clock

The response times of the various safety functions are derived from the Safety Integrated monitoring cycle clock (r9780).

The value of the Safety Integrated monitoring cycle clock (r9780) is determined automatically - you therefore only see it when you are connected online to the drive. With the SINAMICS Integrated of a SIMOTION D410-2, different values can be determined as for a SINAMICS S120 CU310-2 connected to SIMOTION D410-2 because of the architecture.

The Safety Integrated monitoring cycle can be calculated as follows for the SINAMICS Integrated:

x = 32 x p0115[0] (p0115[0] = current controller cycle clock; for D410-2, typically 125 µs for servo drives)

Case A: For PROFIBUS Integrated cycle clock > x: r9780 = x

Case B: For PROFIBUS Integrated cycle clock ≤ x: r9780 = (rounded up (x/Tdp))* Tdp

For drives connected via CU310-2, case A applies generally.

Example:

Servo drive with current controller cycle clock (p115[0] = 125 µs)

PROFIBUS Integrated cycle clock: Tdp = 3 ms

x = 32 x 125 µs = 4 ms ⇒ case B

r9780 = (rounded up (4 ms / 3ms))*3 ms = 6 ms

For detailed information about Safety Integrated response times, see the *SINAMICS S120 Safety Integrated* Function Manual.

SI monitoring cycle clock

The response times of the various safety functions are derived from the Safety Integrated monitoring cycle clock (r9780).

The value of the Safety Integrated monitoring cycle clock (r9780) is determined automatically - you therefore only see it when you are connected online to the drive.

With the SINAMICS Integrated of a SIMOTION D410-2, different values can be determined as for a SINAMICS S120 CU310-2 connected to SIMOTION D410-2 because of the architecture.

The Safety Integrated monitoring cycle can be calculated as follows for the SINAMICS Integrated:

x = 32 x p0115[0] (p0115[0] = current controller cycle clock; for D410-2, typically 125 µs for servo drives)

Case A: For PROFIBUS Integrated cycle clock > x: r9780 = x

Case B: For PROFIBUS Integrated cycle clock ≤ x: r9780 = (rounded up (x/Tdp))* Tdp

For drives connected via CU310-2, case A applies generally.

Example:

Servo drive with current controller cycle clock (p115[0] = 125 µs)

PROFIBUS Integrated cycle clock: Tdp = 3 ms

x = 32 x 125 µs = 4 ms ⇒ case B

r9780 = (rounded up (4 ms / 3ms))*3 ms = 6 ms

For detailed information about Safety Integrated response times, see the SINAMICS S120 Safety Integrated Function Manual.

Required hardware

Controlling the safety functions requires at least the following hardware versions:

Table 10-125 Required hardware versions

Module	Article number	Required product version
SIMOTION D410-2 DP	6AU1410-2AA00-0AA0	C
SIMOTION D410-2 DP/PN	6AU1410-2AD00-0AA0	B

The hardware requirements of the drive components can be found in the *SINAMICS S120 Safety Integrated* Function Manual.

Safety Integrated functions with PROFIsafe (PROFINET example)

Safety Integrated functions are activated via "PROFIsafe on PROFINET" safe communication. Control (F logic) is via a SIMATIC F-CPU which is connected to PROFIsafe via PROFINET, e.g. a SIMATIC S7-1500 F-CPU.

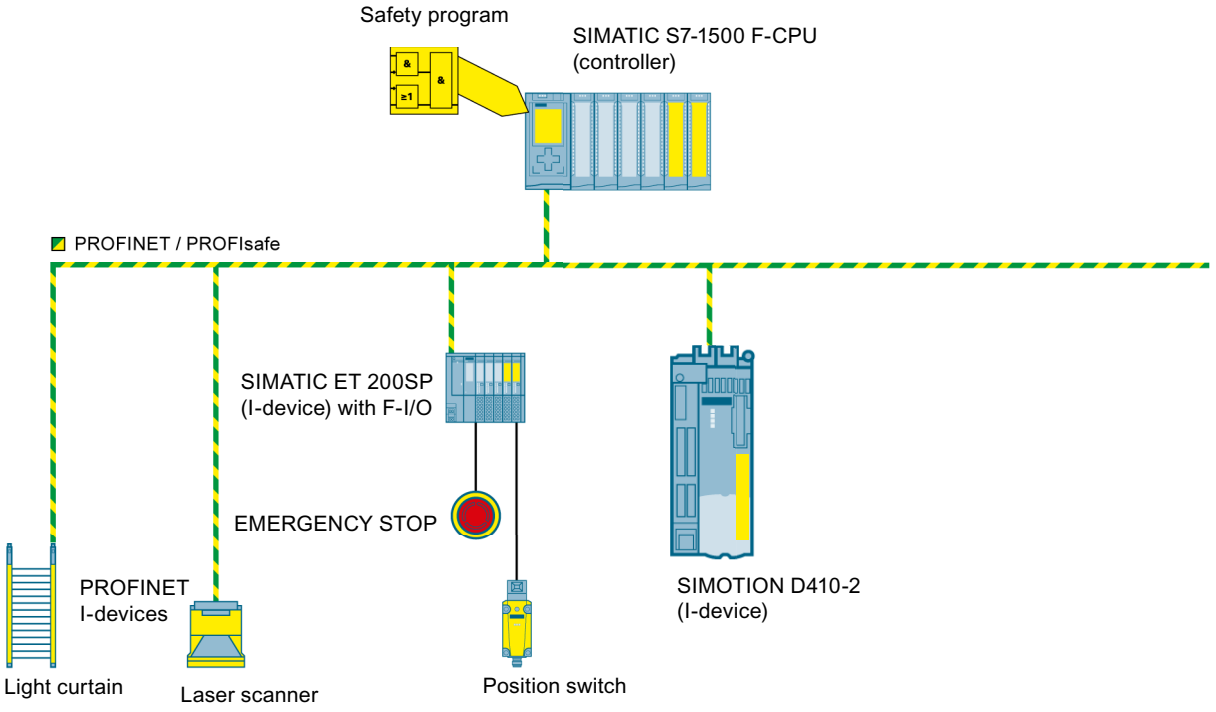


Figure 10-211 SIMOTION D, control of F functions via PROFINET with PROFIsafe

Topologies (F proxy)

The F proxy functionality enables transparent routing of safety telegrams from the SIMOTION I-device (or slave) to the SIMOTION controller (or master).

The topologies that may be used with the SIMOTION D410-2 are listed below. In each case, the description specifies whether the control of Safety Integrated functions is routed through to the drives.

- SIMATIC F CPU (master), connected via PROFIBUS with PROFIsafe to SIMOTION D410-2 (I-slave).
  Note: PROFIBUS not possible configuration via SCOUT TIA
  - Routing to the SINAMICS Integrated drive of SIMOTION D410-2.
  - Routing to drives of CUs that are connected to SIMOTION D410-2.
    The CUs are connected as slaves via PROFIBUS to the DP master interface of the SIMOTION D410-2.
- SIMATIC F-CPU (controller), connected via PROFINET with PROFIsafe to SIMOTION D410-2 DP/PN (I-device)
  - Routing to the SINAMICS Integrated drive of the D410-2 DP/PN.
  - Routing to the drives of a SINAMICS S110/S120 CU connected to the D410-2 DP/PN.
    The CU is connected as a device via PROFINET to the PROFINET interface of the D410-2 DP/PN (= controller) or the CU is connected as a slave via PROFIBUS to the DP master interface of the D410-2 DP/PN.
- The SIMOTION D410-2 is the PROFIBUS master for a direct fail-safe slave-to-slave communication, e.g. between a SIMATIC F-CPU and a SINAMICS S110/S120 CU.
  Direct fail-safe slave-to-slave communication with telegram 901: As of V4.3 SP1 HF9.
  Not for SCOUT TIA (SIMOTION in the TIA Portal)
  See also application example at the following Internet address (https://support.industry.siemens.com/cs/ww/en/view/38701812).

Note

Control for the Safety Integrated functions cannot be routed through to the SINAMICS Integrated of the SIMOTION D410-2 in this configuration.

Topologies (Shared Device)

With the Shared Device functionality, you can configure access to an IO device with several IO controllers using PROFINET. This enables channels/modules to be flexibly assigned to different IO controllers. This option is available for inputs and outputs. You can use this mechanism to access the fail-safe data of a drive configured below a SIMOTION CPU via the F-CPU (e.g. access of an S7-1500 F-CPU to a SINAMICS S120 CU310-2 which is subordinate to a SIMOTION D410-2).

Requirement:

SIMOTION SCOUT (not possible with SCOUT TIA)

Note

When configuring PROFIsafe, it is recommended that you use the I-device F proxy functionality instead of the Shared Device.

Additional references

Additional information on configuring Safety Integrated functions can be found in the

- *SINAMICS S120 Safety Integrated* Function Manual
- *TO Axis Electric/Hydraulic, External Encoder* Function Manual
- *SIMOTION Communication* System Manual
- Internet at the following Internet address (https://www.automation.siemens.com/safety)

Hot plugging

SIMOTION D410-2 is capable of hot plugging, i.e. SIMOTION D410-2 can be unplugged or plugged in again to the PM340/PM240-2 energized.

The following scenarios are therefore possible:

- SIMOTION D410-2 is supplied with power via X124, Power Module is de-energized
- SIMOTION D410-2 is de-energized, Power Module is energized
- SIMOTION D410-2 is supplied with power via X124, Power Module is energized

Unplugging and plugging of a SIMOTION D410-2 is only permitted on one and the same Power Module. If the SIMOTION D410-2 is plugged into a different Power Module (power, type, article number, serial number), then alarm 30074 is issued on the drive side.

SIMOTION D410-2 must always be supplied via X124 **for operation**. There is no warm restart when plugged in again.

If the SIMOTION D410-2 is unplugged from the Power Module, a drive connected to the Power Module stops.

Quantity structures

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S 120 drive software run on a shared control hardware. The IEC 61131-3-compliant PLC integrated in SIMOTION D means that the system is not just capable of controlling sequences of motions, but the entire machine as well.

The **scaling of the PLC and motion control functionality** is performed via Control Units of various performance classes for SIMOTION D

- SIMOTION D410-2 for single-axis solutions and small multi-axis solution with typically 2 to 3 axes
- SIMOTION D425-2 (BASIC performance) for up to 16 axes
- SIMOTION D435-2 (STANDARD performance) for up to 32 axes
- SIMOTION D445-2 (HIGH performance) for up to 64 axes
- SIMOTION D455-2 (ULTRA-HIGH performance) for up to 128 axes or applications with very short control cycle clocks

The possible axis quantity structures depend on the required servo and interpolator cycle clocks and apply for electric, hydraulic and virtual axes.

The **scaling of the drive computing performance** is performed via SINAMICS drive controls for the SIMOTION D410-2. The drive control of a CU310-2 is already integrated (SINAMICS Integrated) for the SIMOTION D410-2. This can be extended with SINAMICS S110/S120 Control Units connected via PROFIBUS or PROFINET.

SIZER

With the SIZER configuration tool, you can easily configure the SINAMICS S110/S120 drive family including SIMOTION. It provides you with support for selecting and dimensioning the components required for a Motion Control task.

- Dimensioning of the PLC and motion control functionality --> selection of the SIMOTION D Control Unit
- Dimensioning of the drive computing performance and the required drive components.

Depending on your performance requirements, SIZER determines the possible number of axes and the resulting utilization on the SIMOTION and SINAMICS side.

Quantity structure of DRIVE-CLiQ components

The following maximum number of DRIVE-CLiQ components can be connected to the SIMOTION D410-2:

- Max. 8 Terminal Modules, of these max.
  - 3 TM15, TM17, TM41
  - 8 TM15 DI/DO, TM31, TM120, TM150
  - 1 TM54F
- Max. 5 encoder systems
  (SMx Sensor Modules or encoders/motors with DRIVE-CLiQ interface)
- Max. 1 DMC20/DME20

You require a DRIVE-CLiQ hub module (DMC20/DME20) or a CUA32 to connect more than one encoder system via DRIVE-CLiQ.

Migration from SIMOTION D410 to SIMOTION D410-2

Upgrade from SIMOTION D410 to SIMOTION D410-2

SIMOTION D410-2 differs both in the structure and in the functionality of a SIMOTION D410.

This has effects that must be taken into account during a changeover.

Change from D410 to D410-2 (upgrade)

The change from SIMOTION D410 to SIMOTION D410-2 is performed with a module replacement in **HW Config**. The procedure when changing is similar to that when changing, for example, from SIMOTION D445-2 DP/PN to SIMOTION D455-2 DP/PN.

The module replacement is started by dragging the new module to the frame of the module rack with the existing module in **HW Config**, see Section Device replacement in HW Config (Page 7530).

The module replacement is performed automatically. During this operation, the technology packages and the device versions are updated. Existing configurations of the SIMOTION D410 are transferred to the SIMOTION D410-2 during the module replacement.

Information on where project adaptations are required or not required when changing from a SIMOTION D410 to a SIMOTION D410-2 can be found in the following sections (see Table "Transfer of existing SIMOTION D410 projects to SIMOTION D410-2").

For further information on generally required measures for project adaptations, see Section Service and maintenance (Page 7515).

Change from D410-2 to D410 (downgrade)

As a SINAMICS drive cannot be downgraded, a module replacement of a SIMOTION D410-2 (SINAMICS Integrated firmware V4.x/V5.x) with a SIMOTION D410 (SINAMICS Integrated firmware V2.x) is not possible.

However, project data can be transferred by means of an XML export/import.

Transfer of existing projects

Existing configurations of the SIMOTION D410 are transferred to the SIMOTION D410-2 during the module replacement.

The following table provides information on where project adaptations are required or not required:

Table 10-126 Transfer of existing SIMOTION D410 projects to SIMOTION D410-2

Keyword	Explanation
PROFIBUS interfaces	The configuration on the PROFIBUS interfaces including any existing PG/PC assignment is retained. Figure: SIMOTION D410 to SIMOTION- D410-2: X21 DP/MPI → X21 DP/MPI One of the differences of SIMOTION D410-2 DP compared to D410 DP is that it has an additional (second) PROFIBUS interface (X24). A key difference of SIMOTION D410-2 DP/PN compared to D410 PN is that it also has a PROFIBUS interface (X21).
PROFINET interface	The configuration on the PROFINET interface, including any existing PG/PC assignment, is retained. Figure: SIMOTION D410 PN to SIMOTION D410-2 DP/PN: X200 → X150 P1 X201 → X150 P2
Message frame configurations	The message frame configurations on the PROFIBUS or PROFINET interface are retained. The same applies for the addresses of the slots/subslots.
SINAMICS Integrated	The SINAMICS Integrated is to be replaced with the new type and its corresponding version. The drive configuration and the assignment of the DRIVE-CLiQ interface are retained.
PG/PC connection / Ethernet interface	In contrast to SIMOTION D410, SIMOTION D410-2 has an Ethernet interface. Due to the significantly improved engineering performance, we recommend that you connect a PG/PC via the Ethernet interface of the SIMOTION D410-2 instead of via PROFIBUS.

Keyword	Explanation
Configuration of the onboard I/Os	The configuration of the onboard I/Os (4 DI, 4 DI/DO) is retained. The new features mean the functionality, quantity structures and terminal designations for the onboard I/Os differ for the SIMOTION D410-2 compared with the SIMOTION D410. The number of onboard I/Os has more than doubled so that previously required terminal modules / I/O modules may no longer be necessary.
Safety	SIMOTION D410-2 has 3 F-DI and 1 F-DO for Safety Integrated Extended/Advanced Functions so that a TM54F used for SIMOTION D410 may be omitted.
CompactFlash card / firmware	SIMOTION D410 and SIMOTION D410-2 have different CompactFlash cards and different card mappers (firmware). A CompactFlash card of the SIMOTION D410/D4x5-x must not be inserted into a SIMOTION D410-2 and vice-versa! 1 GB CompactFlash cards of the SIMOTION D410/D4x5-x can also be used for the SIMOTION D410-2 when a new boot loader is loaded. For details, see Section Boot loader on the CompactFlash card (Page 7547).
Connectable drive components	Like the SINAMICS S120 CU310-2, the SIMOTION D410-2 also no longer supports some older drive components. For details, see Section Permissible combinations (Page 7511)
User program	Although, in principle, a SIMOTION D410 user program can run on a SIMOTION D410-2, modifications may be required because of innovations in the hardware. Examples: • The runtime behavior of the SIMOTION D410-2 changes because of the increased performance. As long as the user program does not contain any "runtime-dependent code", adaptations should not be necessary. • Fan faults for the SIMOTION D410-2 are signaled via a diagnostic buffer entry PeripheralFaultTask and a system variable (fanWarning). For the SIMOTION D410, fan faults were signaled by a warning on the drive side (A1009). • The following additional functions are available for the SIMOTION D410-2: – Backup of the non-volatile SINAMICS data (NVRAM data) via CU parameter p7775 – Automatic detection of a module replacement on the basis of the serial number of the Control Unit
Hardware	Due to the innovative hardware, the following changes have been made for the SIMOTION D410-2 compared with the SIMOTION D410: • The control elements of the SIMOTION D410-2 have been standardized with the SIMOTION D4x5-2 (rotary switch instead of DIP switch for SIMOTION D410, DIAG button, etc.). • The interfaces on the top of the module are positioned differently. • Changed I/O terminals: – Spring-loaded terminals instead of screw terminals – Fitting for the 24 V terminal block – The SIMOTION D410 connector X120 must be re-wired – The wiring of the SIMOTION D410 connector X121 can be taken over exactly for the new connector of the SIMOTION D410-2 (the terminal designation for the reference ground DI 0 - DI 3 changes from M1 to M2) SIMOTION D410-2 has the following additional interfaces: • Additional Ethernet interface • Additional PROFIBUS interface The mounting depth of the SIMOTION D410-2 has been reduced by approx. 15% compared to the SIMOTION D410.

See also

Operations and their effect on the user memory (Page 7354)

Permissible combinations

Some "older" DRIVE-CLiQ components can no longer be used with the SIMOTION D410-2.

Table 10-127 DRIVE-CLiQ components not supported with SIMOTION D410-2

DRIVE-CLiQ components	Article number suffix
SMC30 Sensor Module Cabinet	< 2
TM31/TM41 Terminal Modules	< 1
SME20/25 Sensor Modules External	< 3
CUA31 Control Unit Adapter	< 1
Power Modules (chassis)	< 3

A detailed, regularly updated list of the DRIVE-CLiQ components approved for use with SIMOTION, as well as notes on their use, can be found on the Internet (https://support.industry.siemens.com/cs/ww/en/view/11886029).

If incorrect components are used, a topology error is signaled
F01360 Topology: Actual topology not permitted.

CompactFlash card and license combinations

CF card

A CF card with D410 firmware / drive software cannot run on a D410-2. The same applies for the reverse situation.

In the event of a fault all 4 LED displays light up yellow.

SIMOTION D410-2 licensing

SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications. SIMOTION D410-2 has an integrated drive control for either a servo, a vector or a V/f axis. One real axis can be used without requiring a license for a SIMOTION D410-2. Speed-controlled axes and virtual axes never require a license.

SIMOTION D410-2 can be extended with additional SINAMICS S110/S120 Control Units (e.g. CU305) and so can also be used for smaller multi-axis applications (e.g. with 2 - 3 axes). These additional axes must be licensed with the single-axis licenses or the "D410-2 MultiAxes Package." See Section "CompactFlash card" in the *SIMOTION D410-2* Manual.

If you need to license one POS axis, use the POS single-axis license; in the case of GEAR/CAM or if you require more than one POS licenses, it is better to use the D410-2 MultiAxes Package,

Note

If you use more than one real axis with SIMOTION D410-2, you must license these additional axes. The axis license with the highest functionality is covered by the inclusive license (a real axis). The functionality has the following granularity: CAM > GEAR > POS.

Example:

You use 2 real axes: 1 POS, 1 CAM.
Because the CAM license has a higher value, and so inclusive, you only need to purchase a POS license.

Licenses are required for runtime functions such as SIMOTION IT Virtual Machine. These can be pre-installed on a CompactFlash card (CF card) or ordered separately.

Note

Path interpolation is supported as of V4.4.

Special functions SIMOTION D410-2

Automatic restart after FAULT state

Overview

A SIMOTION D410-2 can perform a restart automatically after the FAULT operating state has occurred via the system variable **_automaticRestart** available as of V4.4.

The system variable **_automaticRestart** is intended for applications in which an automatic restart is mandatory after a crash caused by an extreme situation, e.g. wind generators after a lightning strike.

The automatic restart causes the device with the project to boot automatically from the CF card.

Note

System variable **_automaticRestart** is visible on all SIMOTION CPUs, but the functionality is only supported by SIMOTION D410-2.

Note

Restart cannot be ensured in all cases, for example, if a hardware fault occurs.

Programming

Table 10-128 System variable _automaticRestart

Value	Response
16#00000000	Automatic restart after FAULT state is not active (default).
16#00000000 < n < 16#FFFFFFFF	Automatic restart after FAULT state is active until the value 0 is reached. On a restart after FAULT state, the value is decremented by 1.
16#FFFFFFFF	Automatic restart after FAULT state. The value remains unchanged.

The system variable can be set to the desired value and decrementing monitored in the user program.

Alternately, the start value can be set manually in the SIMOTION SCOUT symbol browser under "force variable." To ensure the manually set start value remains effective, note the following:

- The start value must be set in the offline project and loaded onto the device by project download.
- The value that is loaded onto the device must differ from the start value that was last loaded onto the device by project download.
  This ensures that the current value of the system variable **_automaticRestart** in the runtime system is not reset on every project download (crash history is not deleted).

The system variable is retentive, its content and therefore the "automatic restart after FAULT state" function is retained after an interruption in the power supply.

An automatic restart after FAULT state that has occurred can be diagnosed by the diagnostic buffer entry **Automatic restart of the CPU performed after FAULT operating state**.

System variable **_automaticRestart** is not backed up with **_savePersistentMemoryData**:

- Because an automatic restart can be caused by hardware problems and this cause is not to be transferred to the new module if a module is replaced.
- Because the user program will decide for itself under what condition system variable **_automaticRestart** will be set to a certain value.

If an error occurs after the automatic restart, there is not special error handling for this case. If this error prevents startup, the SIMOTION device cannot enter the RUN state. An error during startup does not trigger an automatic restart.

Triggering a restart with a user program

A user program of the SIMOTION device can trigger restart of the device via the system function **_restart** available as of V4.4.

This system function, like the system variable **_automaticRestart**, described in Section Automatic restart after FAULT state (Page 7512) is intended for applications in which an automatic restart is mandatory after a crash caused by an extreme situation, e.g. wind generators after a lightning strike.
The system function **_restart** enables a response to non-acknowledgeable drive errors with a

OFF2 reaction, which result in the inability of the drive to move. These drive errors can only be exited via a restart or by power off/on.

Note

The system function **_restart** is provided for all SIMOTION devices, but restart itself is only performed for the D410-2 device.

The restart causes the device with the project to boot from the CF card.

So that, after a restart has been performed, the SIMOTION device enters the RUN state again, the system variable **_startUpData.operationMode** must be on RUN.

The system function **_restart** only supports synchronous command processing.

Return values:

- 16#00000000: Function performed without errors.
- 16#FFFF8091: Functionality is not supported for the device.

If the restart was triggered by the system function **_restart**, the following entry is made in the diagnostic buffer: The SIMOTION device has been switched to STOP and restarted by the user program using the _restart system function.

The user program must ensure that, when the D410-2 restarts, no axis movements or other unwanted influences on the machine by the user program occur.

If an error occurs after the automatic restart, there is not special error handling for this case. If this error prevents startup, the SIMOTION device cannot enter the RUN state. An error during startup does not trigger an automatic restart.

10.1.2.8 Service and maintenance

Overview

Introduction

It is possible to distinguish between the following scenarios when replacing and updating components:

- Replacing modules (spare part scenario)
  - Spare parts replacement for SIMOTION D410-2 (Page 7519)
  - Removing and replacing the SIMOTION D410-2 (Page 7519)
  - Replacing DRIVE-CLiQ components (Page 7521)
  - Replacing the fan (Page 7523)
  - Replacing the CompactFlash card (Page 7525)
- Adapting a project (new device type / new device version)
  The project needs to be adapted if you want to change the type (e.g. D410 DP -> D410-2 DP) or version of the SIMOTION device in your existing project.
  - Creating backup copies (project/CF) (Page 7525)
  - Backing up user data (back up variables) (Page 7526)
  - Upgrading a user project to the new SCOUT version (Page 7527)
  - Platform replacement via XML export/import (Page 7528)
  - Preparing the device replacement (Page 7530)
  - Device replacement in HW Config (Page 7530)
  - Upgrading technology packages (Page 7531)
  - Upgrading the device version of SINAMICS S120 Control Units (Page 7534)
  - Upgrade the libraries (Page 7535)
  - Save project, compile and check consistency (Page 7535)
- Performing a firmware and project update
  - Upgrading the boot loader on the CompactFlash card (Page 7536)
  - Update - preparatory measures (Page 7536)
  - Update via SIMOTION IT web server (Page 7537)
  - Upgrade via device update tool (upgrading SIMOTION devices) (Page 7537)

– Update via CF card
  Backup of CompactFlash card data (Page 7540)
  Firmware update using a CompactFlash Card (Page 7542)
  Upgrading SINAMICS (Page 7542)
  Download project to target system (Page 7544)

Note

Upgrading using the device update tool offers a number of advantages (keeping retain data, option of downgrading, no license key handling, etc.). We would, therefore, recommend using this method for firmware and project updates.

Please also observe the information on handling the CompactFlash card.

- Changing the CompactFlash card (Page 7545)
- Writing to a CompactFlash card (Page 7546)
- Formatting the CompactFlash card (Page 7546)
- Boot loader on the CompactFlash card (Page 7547)
- Recommended method of handling CompactFlash cards (Page 7548)
- Card reader for CompactFlash cards (Page 7549)

Note

This document uses the following terms:

- Upgrade: Denotes upgrading to a higher version of a component/software
- Downgrade: Denotes reverting to a previous version of a component/software
- Update: In general terms, denotes the act of bringing a component/software up-to-date (in isolated cases, this may see an upgrade or downgrade)

Upgrade options

The exact procedure when replacing or updating components depends on various factors.

If a project is upgraded, the procedure depends on the scope of change of the versions.

- Change of the SIMOTION main version
- Change of the SIMOTION service pack or hotfix version
- Change of the PROFINET version
- Change of the SINAMICS version
  (There are SIMOTION versions that contain several SINAMICS versions for a device.)

If a different SIMOTION controller is to be used, the procedure depends on whether a device or a platform replacement is required.

Examples of upgrade scenarios are listed in the following overview table. They are shown in the columns. The lines list the principle measures that have to be performed. Whether the measure

has to be performed in a specific case, must be decided project-specifically.
Grayed-out cells mean that a measure is not required.

Note

If the version is changed and the SIMOTION controller replaced at the same time, then all measures apply; the measures must be performed in the TOP-DOWN sequence according to the table.

Action/Measure	Upgrade project				Replacement of SIMOTION controller		Activity relates to
	Main version	Service pack or hotfix version	PROFINET version	SINAMICS version	Device replacement via HW Config	Platform replacement via XML export/import	
Customize project							
Examples	V4.1 ⇨ V4.2	V4.1 SP2 ⇨ V4.1 SP4	PN2.1 ⇨ PN2.2	V2.5 ⇨ V2.6.2	D445-2 ⇨ D455-2 D4x5 ⇨ D4x5-2 D410-2 DP ⇨ D410-2 DP/PN	C240 ⇨ D445-2 D410-2 ⇨ D445-2 D445 ⇨ D445 (S120) (SM150)	Project/SIMOTION SCOUT
Create backup copy (project/CF)							
Back up user data (back up variables)	Only if required				Only if required	Only if required	
Upgrade user project to new SCOUT version							
Platform replacement via XML export/import	During platform replacement, the target version is specified (= version of the device that is the recipient of the import)						
Prepare device replacement							
Device replacement in HW Config							
Upgrade technology packages (TP)	[4]	[4]			[4]		
Upgrade device version of SINAMICS S120 CUs [3]				For external CUs, only if necessary			
Upgrade library	Libraries are version-dependent				Libraries may be device-dependent	Libraries may be device-dependent	
Save/compile project; check consistency							
Perform a firmware and/or project update							
Upgrade the boot loader on the CompactFlash card	Check whether new version requires a new boot loader [2]				Check whether the new [2] device requires a new boot loader		Memory card/Target hardware
Update - preparatory measures							
Upgrade via IT DIAG	Selection of one of the 3 update methods				Only possible if the CF card contains valid firmware		
Update via device update tool							
Update via CompactFlash Card							
Backup of the CompactFlash card data							
Firmware update using a CompactFlash card							
Upgrade SINAMICS [5]							
Download the project to the target system [1]							

Not relevant Relevant

1) Alternative: Load the project to the CF card using a card reader

2) Refer to the compatibility list (https://support.industry.siemens.com/cs/ww/en/view/18857317)

3) The versions of SINAMICS Integrated and Controller Extensions are upgraded automatically in the **HW Config** during device replacement

4) The technology packages are automatically upgraded. The user can directly specify a TP

5) SINAMICS components are upgraded/downgraded to the component version of the CF card. Observe the LED codes! After the upgrade it is necessary to switch the device off and on.

Figure 10-212 Overview of upgrade options

Replacing modules

Spare parts replacement for SIMOTION D410-2

A replaced module is detected by the controller on the basis of the serial number. This automatically deletes the non-volatile data and transfers the data saved on the CF card to the controller. See Section Replacing modules in the spare part scenario (Page 7361).

Removing and replacing the SIMOTION D410-2

Overview

The following describes the procedure for replacing a module (spare part). In principle, it is also possible to replace the SIMOTION D410-2 when the PM340/PM240-2 is energized.

You will find more information on this in Section Hot plugging (Page 7507).

⚠ WARNING

Danger to life through electric shock

Note that Power Modules with FSB performance rating or higher still carry a residual intermediate voltage after shutdown.

Wait 10 minutes before you remove any screws.

For further information, see the *SINAMICS S120 AC Drive* Manual.

Removing a defective module

How to remove the SIMOTION D410-2 module:

1. Switch off the power supply to terminal X124.
2. Remove the CF card from the card slot.
3. Remove the connections for the power supply (X124).
4. Disconnect the communication interface connectors from the device:
   - DRIVE-CLiQ interface (X100)
   - PROFIBUS DP interface (X21)
   - PROFIBUS DP interface (X24; D410-2 DP only)
   - PROFINET interface (X150 P1 and P2; D410-2 DP/PN only)
   - Ethernet interface (X127)
5. If necessary, disconnect the plug-in connectors to the digital inputs/outputs of the interfaces (X120, X121, X130, X131).

6. Disconnect the plug of any encoder connected to the X23 encoder interface.
7. Depending on the mounting method, disassemble the SIMOTION D410-2 from the Power Module or remove the SIMOTION D410-2 from the mounting plate (see Section Mounting (Page 7311)).

Installing a new module

How to install the new SIMOTION D410-2 module:

Note

Observe the information for installation (Page 7311), wiring and connecting (Page 7316) the SIMOTION D410-2.

1. Install the new SIMOTION D410-2 on the Power Module or on the mounting plate.
2. Connect all previously removed connectors.
3. Terminate the load voltage supply cables at the terminal block.
4. Rewire the shielding of all cables.
5. Insert the original CF card into the card slot of the new SIMOTION D410-2.
6. Switch on the power supply. The new SIMOTION D410-2 is immediately ready for operation.

Replacing SIMOTION D410-2 modules without PG/PC

To enable a module replacement without a PG/PC, you must back up the current non-volatile SIMOTION and SINAMICS data on the CF card during operation.

NOTICE

Loss of data due to failure to make backup copies

Non-backed-up non-volatile SIMOTION data can be lost on hardware replacement (module defect), for example, if the current value of the retain variables have not been backed up and replaced by their initial values again.

Back up the non-volatile SIMOTION data on the CF card.

NOTICE

Repeat referencing necessary after absolute encoder overflow

If an absolute encoder overflow occurs after **_savePersistentMemoryData**, the actual position value is no longer correct after the non-volatile SIMOTION data are restored.

In this case, homing (absolute encoder adjustment) must be repeated.

See also

Hot plugging (Page 7507)

Spare parts replacement for SIMOTION D410-2 (Page 7519)

Operations and their effect on the user memory (Page 7354)

Replacing modules in the spare part scenario (Page 7361)

Replacing DRIVE-CLiQ components

Requirement

DRIVE-CLiQ components not only support replacement in the switched-off state of the machine/system (Power Off), but also replacement during operation. For this, the component to be replaced must be at the end of the DRIVE-CLiQ line.

Removing DRIVE-CLiQ components

1. Deactivate the affected component or drive object.
2. Remove the DRIVE-CLiQ connector.
3. Remove the supply voltage of the component and uninstall the component.

Installing DRIVE-CLiQ components

1. Mount the component and reconnect the supply voltage.
2. Reconnect the DRIVE-CLiQ cable at the same location (port). The cable must have the same length as the old one.
3. Activate the affected component or drive object.

Parameters for topology comparator and component replacement

In the expert list, you can use CU parameter p9906 to specify how the electronic rating plates are compared for all the components of a Control Unit. The type of comparison can be changed subsequently for each individual component by using p9907/p9908 or right-clicking in the topology. All data on the electronic rating plate is compared by default.

- When p9909 = 1, the serial number and the hardware version of the new replacement component are automatically transferred from the actual topology to the target topology, and then saved in the non-volatile memory.
- When p9909 = 0, serial numbers and hardware versions are not automatically transferred.

The setting p9901 = 1 enables the **spare parts / components replacement without tool support** to be carried out. The new serial number of the spare part is automatically transferred from the actual topology to the target topology and saved in non-volatile memory. In order for this to occur, the components that have been replaced must be of the same type and have the same article number, such as "6SL3055-0AA0-5BA0". The last or last two digits of the article number (depending on the component type) are not checked, as the HW version, for example, is coded in these. This mechanism is also applied when several components are replaced.

Modified wiring following module replacement

In the default setting for the topology comparator, modified wiring configurations of the DRIVE-CLiQ components (e.g. a cross-exchange) are not accepted for safety reasons and are generated by a fault.

If a cross-exchange of components is required (i.e. existing components are replaced with other existing components, and no spare parts are used), e.g. for troubleshooting purposes, the topology comparator must be reduced via parameter p9906 or preferably via p9907/p9908; or as an alternative, by right-clicking in the topology.

Note

In this case, incorrect insertion of components is no longer monitored.

Automatic upgrading/downgrading (firmware update)

During the startup, the system automatically upgrades or downgrades all DRIVE-CLiQ components to the version of the component firmware on the CF card. Components that cannot be downgraded to the component firmware version of the CF card (e.g. old firmware on the CF card and new components to which the old firmware cannot be loaded) retain their firmware version. The resulting firmware version combinations are always functional.

The CONTENT.TXT file in the main directory of the CF card contains the component version.

Note

During an automatic FW update, please take note of the messages and alarms in the SIMOTION SCOUT detail window.

A firmware update on the SIMOTION D410-2 is signaled by the RDY LED flashing yellow, while on the DRIVE-CLiQ components (TM, SMC, etc.) it is signaled by the RDY LED flashing red/green.

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

The update function can be deactivated using CU parameter p7826 in the expert list.

Additional references

For more information on this topic, see the following references:

- *SINAMICS S120* Commissioning Manual
- *SINAMICS S120* Function Manual

Replacing the fan

Overview

The SIMOTION D410-2 fan switches on depending on the internal module temperature.

Fan faults are signaled by a diagnostic buffer entry, system variable and PeripheralFaultTask, see section Fan (Page 7363).

The SIMOTION D410-2 does not have to be removed from the Power Module in blocksize format to replace the fan. A defective fan can also be replaced when the SIMOTION D410-2 is switched on.

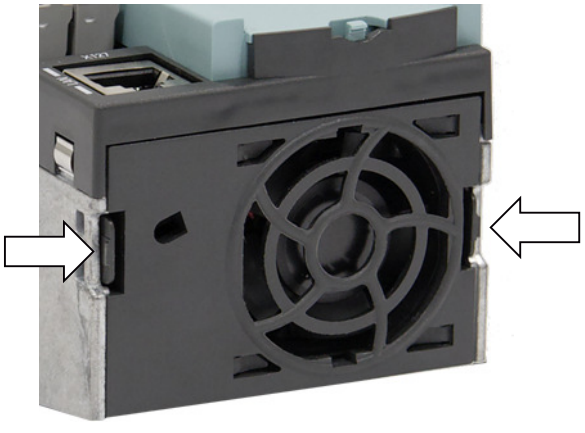
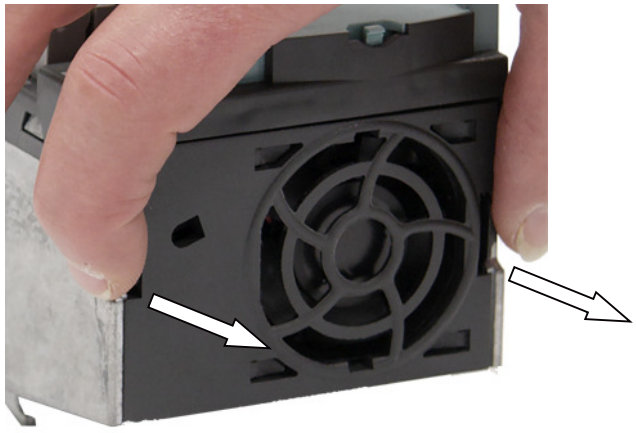
Note

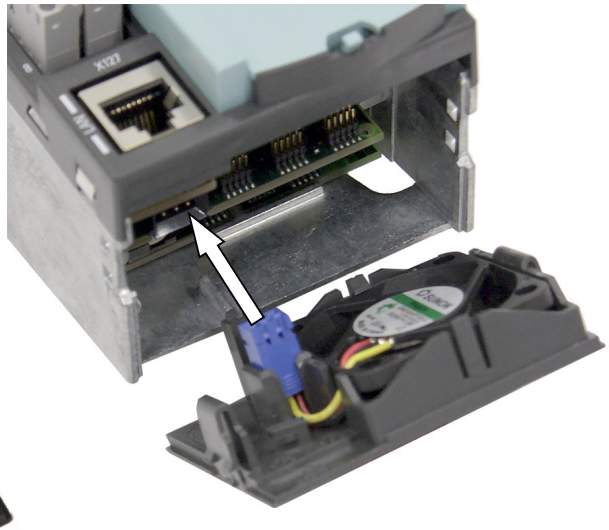
The fan/battery module should preferably be replaced when the power is switched off and only when the CPU is in the STOP state. Otherwise you risk an unintentional failure of the machine/system, e.g. through accidental disconnection of cables.

Replacing the fan

The fan is available as a spare part. For the article number, see section "Spare parts and accessories" in the SIMOTION D410-2 Manual.

NOTICE
Damage to device electronics caused by electric fields or electrostatic discharge The fan on the SIMOTION D410-2 may only be replaced by qualified skilled personnel and strictly in accordance with ESD guidelines (Page 7576).

	
1) Release the snap hooks on the left and right side of the fan by pressing them inwards.	2) Pull the fan forwards and out of the SIMOTION D410-2 housing.

	
3) Locate the centering frame of the replacement fan on the mounting lug of the SIMOTION D410-2.	4) Slide the replacement fan on the mounting lug into the Control Unit. Make sure that the fan connector fits into the centering frame of the the printed circuit board. Engage both snap hooks on the fan in the housing of the Control Unit.

Fan service life

Fans are parts subject to wear. The fan is monitored for SIMOTION D410-2. A failure is signaled in the diagnostic buffer and can be evaluated by the user program (e.g. via the PeripheralFaultTask).

Contamination is the main cause of fan failure. A visual inspection should be made as a first criterion for replacement. If the fan is highly contaminated, it must be replaced.
If no contamination is present, we recommend that the service period is used as criterion. Because the service life of the fans depends greatly on the operating conditions (temperature, humidity, number of operating hours per day, contamination caused by dust, etc.), no fixed limits for all applications are can be specified.
In the field, under average industrial conditions, a replacement interval of 5 to 7 years has proven itself.
We recommend that the maintenance intervals are adapted for the specific application based on empirical values.

Replacing the CompactFlash card

If spare parts are used, you must contact the Technical Support Center to transfer the license key of the defective CompactFlash card to the new CompactFlash card.

Proceed as follows to write your project to the new CompactFlash card:

- Changing the CompactFlash card (Page 7545)
- Writing to a CompactFlash card (Page 7546)

Detailed information about licensing can be found:

- In the *SIMOTION SCOUT* Configuration Manual
- In the FAQs on the Internet (https://support.industry.siemens.com/cs/ww/en/view/73501913)

Customizing the project

Overview

Overview

The project needs to be adapted if you want to replace the type (e.g. D410-2 DP ⇒ D410-2 DP/PN) or version of the SIMOTION device in your existing project.

Procedure

The exact procedure for project adaptations depends on the scope of the target hardware and version changes.

An overview can be found in the Upgrade options overview (Page 7518) figure.

Creating backup copies (project/CF)

Requirement

Before adapting the project, it is essential that you create the following backup copies:

- A backup copy of the project and
- A backup copy of the CF card contents, see Saving CF card data (Page 7540).

See also

Update - preparatory measures (Page 7536)

Backing up user data (back up variables)

Overview

With the SCOUT functions "Save variables" and "Restore variables", you have the option of backing up and restoring data that were changed during operation and only stored in the runtime system. This is necessary if a SIMOTION platform is changed or a version upgraded, for example.

The "Save variables" function creates XML files that can be saved in any folder.

The following types of data can be backed up:

- Remanent global device variables and unit variables, as well as TO retain data located in the NVRAM of the controller
- Data saved with _saveUnitDataSet or _exportUnitDataSet and located on the CF card

Note

During an upgrade, this function is only required for the purpose of backing up and restoring unit data records created with _saveUnitDataSet.

Retain and unit data (saved with _exportUnitDataSet) remain valid even after a version upgrade.

SIMOTION retain data can also be backed up to a memory card without the use of SIMOTION SCOUT. For this purpose, use:

- The _savePersistentMemoryData function or
- The service selector switch or the DIAG pushbutton or SIMOTION IT web server, see Section Non-volatile SIMOTION data (retain data) (Page 7558).

Procedure

The user data must be saved **before** the SCOUT project is upgraded. This can be done regardless of whether the version upgrade is performed with the "old" or the "new" SCOUT version.

The procedure is described below based on the example of a new SCOUT version.

1. Open the project.
   When you open the project, a window appears with a message that the project you are opening was created with a different SCOUT version, as well as a query asking whether you want to perform an upgrade.

Figure 10-213 "Convert project" message

2. Confirm the prompt with "OK". The project will be converted to the current version.

3. A dialog then appears with a prompt asking whether the project should be opened write-protected.

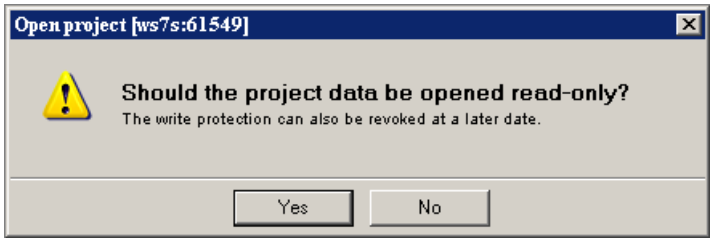


Figure 10-214 Prompt for write-protected opening

4. Confirm this prompt with "Yes" (open write-protected).
5. Set the SIMOTION D410-2 to the STOP operating state.
6. Set the SIMOTION D410-2 to online mode and perform the SCOUT **Save variables** function. The retain variables (interface and implementation) and the user files (with _saveUnitDataSet or _exportUnitDataSet) are saved to the PG/PC.
7. Then close the project.

Note

An online connection is possible only when the PG/PC is configured for the controller.

Update the PG/PC assignment using .

Online connection is now possible.

Additional references

For further information, see the *SIMOTION SCOUT* Configuration Manual.

See also

Diagnostic data and non-volatile SIMOTION data (Page 7557)

Upgrading a user project to the new SCOUT version

Requirement

It is essential that a backup copy be made of the original project before the upgrade, because the data storage of the project is also upgraded during the upgrade. This ensures that you can always return to the original project if the upgrade fails (power interruptions, unexpected faults, incorrect operation, etc.).

Procedure

1. Open the project. A window appears with a message that the project to be opened was created with another SCOUT version as well as a prompt as to whether the upgrade should be performed.

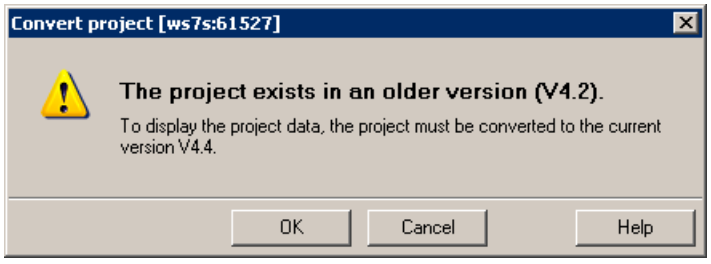


Figure 10-215 "Convert project" message

2. Confirm the prompt with "OK". The project will be converted to the current version.
3. A dialog then appears with a prompt asking whether the project should be opened write-protected.

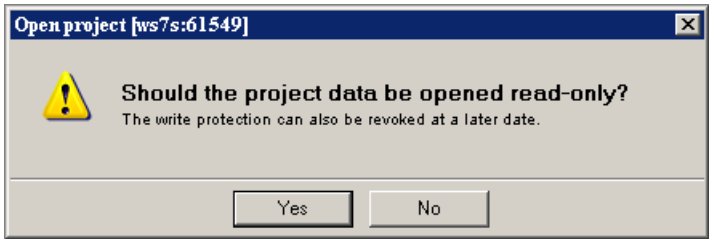


Figure 10-216 Prompt for write-protected opening

4. When upgrading the version, click "No" (open not write-protected).

Note

A project that was last edited with a higher SCOUT version cannot be opened by a SCOUT with a lower version.

Remedy:
Convert the project beforehand with the more recent SCOUT version to the required software version ("Project" > "Old project format" > "Save in old project format"). The project can then be opened with the lower SCOUT version.

Platform replacement via XML export/import

Overview

A platform replacement is always required when an existing project is to be used for another SIMOTION platform. The platform replacement is always performed via an XML export/import.

The following devices can be interchanged via a platform replacement:

- Replacement between SIMOTION C, P and D (e.g. C240 ⇒ D445-2 DP/PN)
- Replacement between D410/D410-2 and D4x5/D4x5-2 (e.g. D410-2 DP/PN ⇒ D445-2 DP/PN)
- Replacement between SIMOTION D (SINAMICS S120 Integrated) ⇒ SIMOTION D (SINAMICS SM150 Integrated)

Platform replacement during project downgrades:

It is not possible to downgrade to a lower SINAMICS version. However, it is possible to transfer project data by means of an XML export/import.

Preparations

Before the platform replacement can be performed, preliminary work may be necessary in the existing project.

If a SIMOTION D4x5-2 is to be imported into a SIMOTION D410-2, then only the permitted quantity structure of the D410-2 may be configured in the D4x5-2. This applies for all components, e.g. not only an infeed, but also the permissible power units.

A SIMOTION D4x5-2 with CU adapter and Power Module in blocksize format can be imported into a SIMOTION D410-2 when the CU adapter is connected to port 0. Otherwise, the topology will be destroyed.

Generally, the success of an import always also depends on the specific configurations of the drive units, and whether the configuration is possible for the device to which the import is to be performed. Also note any error messages that may occur.

Procedure

Proceed as follows:

1. In the project navigator of SIMOTION SCOUT, right-click the SIMOTION controller that is to be replaced.
   Select "Expert" > "Save project and export object" in the context menu.
   "Save project and export object" exports selected data of the selected object in XML format. This data export can then be reimported into other projects. The entire project is not exported, only the data of the selected object (e.g. only the D410-2 or only the SINAMICS Integrated).
2. Specify the desired path and start the XML export.
3. When the export has been performed error-free, delete the device from the project and confirm the prompt.
4. Insert the desired platform as new device in the project navigator of SIMOTION SCOUT. With the selection of the device, you also define the SIMOTION version, and with a SIMOTION D, also the SINAMICS version.
5. Import the data of the original platform into the new device. To do this, right-click the new device and select "Expert" > "Import object" in the context menu.
6. Select the location where the XML export data is to be stored and start the import. Confirm the prompt to continue with the import.

Confirm the message with regard to the import of a "non-compatible type" with "OK".

Preparing the device replacement

Overview

In contrast to the platform replacement, project data can be particularly easily transferred when a device is replaced. The device replacement is performed via **HW Config**, whereas an **XML export/import** is required for a platform replacement.

A device replacement is only possible within SIMOTION D.

The following devices can be interchanged:

- Replacement between generations (D410 ⇒ D410-2)
- Replacement between variants of the generations (D410-2 DP ⇒ D410-2 DP/PN)
- Replacement of SIMOTION, SINAMICS and/or PROFINET version (e.g. D410 V4.1 - PN V2.1 SINAMICS S120 V2.5 ⇒ D410 V4.2 - PN V2.2 SINAMICS S120 V2.6.2).

Note

When replacing a device, note that the SINAMICS version of the new SIMOTION device must be the same or higher.

A downgrade to a lower SINAMICS version (e.g. SIMOTION D410-2 with SINAMICS V4.x to SIMOTION D410 with SINAMICS V2.x) is not possible.

When carrying out a "D410-2 DP/PN ⇒ D410-2 DP" device replacement, the PROFINET IO system is omitted and the bus interface changes. You must reconfigure this after the device replacement. Any existing PROFINET field devices on the PROFINET interface must be replaced with PROFIBUS field devices on the PROFIBUS interface.

Migration D410 ⇒ D410-2

If a SIMOTION D410 is replaced with a SIMOTION D410-2, the SINAMICS Integrated is also automatically changed to a new type and the corresponding version.

Device replacement in HW Config

Procedure

1. Double-click the SIMOTION device to be replaced in the project navigator in SIMOTION SCOUT. **HW Config** opens.
2. Open the "SIMOTION Drive Based" folder in the hardware catalog.

Note

SIMOTION D is modeled as a compact device in **HW Config**. When replacing a module, you must drag the new module to the header of the displayed module rack and **not to slot 2**. Ensure that you do not delete the D410-2 rack!

3. Drag-and-drop the new module to the header of the module rack.
   The old module is replaced.
   Alternatively, you can:
   - Select the rack header and double-click the new module in the module catalog to replace the previous module, or:
   - Right-click the rack header and select the "Replace object" option.

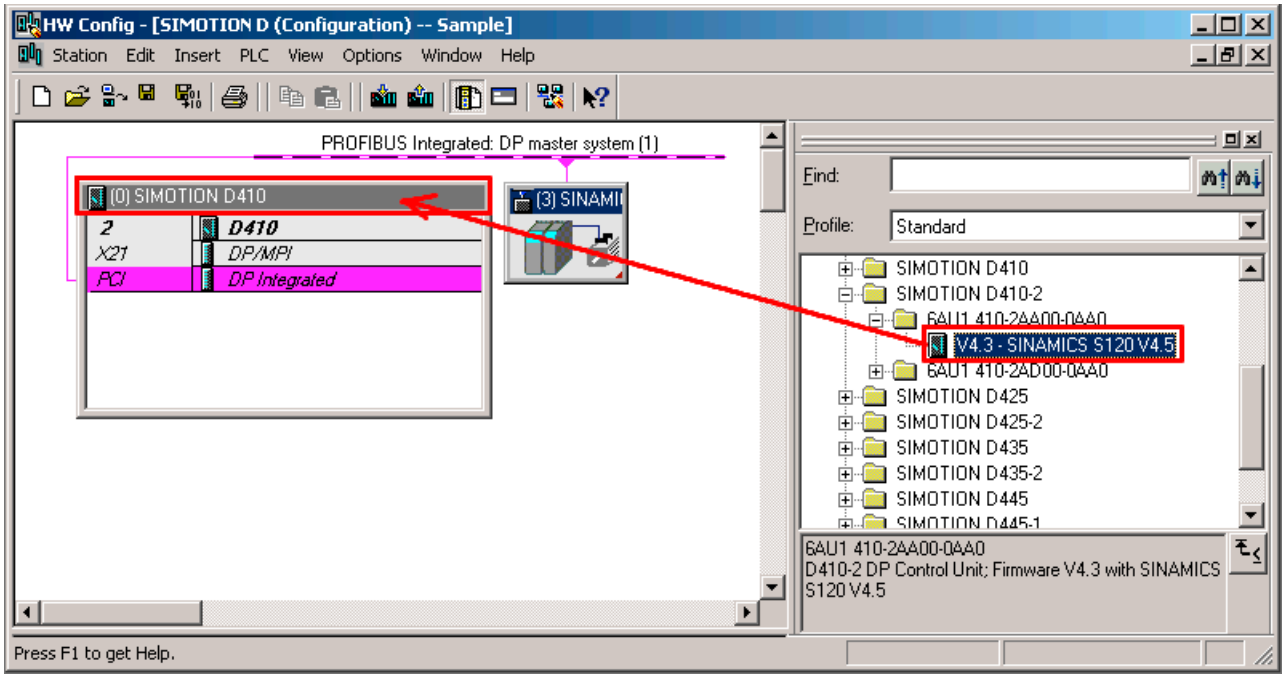


Figure 10-217 HW Config module replacement

4. Confirm the dialog box that appears with "Yes" to replace the SIMOTION device. The module is replaced.
5. Accept the changes made to the hardware configuration with "Station" > "Save and compile".
6. Close **HW Config**.

Note

The engineering system automatically performs the following actions when replacing a module (if necessary):

- Update of the technology packages (TPs)
- Automatic upgrade of the SINAMICS Integrated

The updated data is transferred to the project and the entire project saved.

Upgrading technology packages

Overview

The SIMOTION technology packages (e.g. TP CAM, DCBlib) are available in various versions.

You can only use the functions of the technology objects selected if the technology objects are available in the target system. You can select the technology packages and their version for each SIMOTION device. Each version of SIMOTION SCOUT has a kernel (FW version) for the SIMOTION CPU and an appropriate technology package.

TPs during upgrades

Device replacement (in **HW Config**), platform replacement (XML export/import), or even upgrades may cause versions of SIMOTION technology packages (TPs), which are assigned to individual technology objects (TOs), to change.

- The TP version may change if the main version changes.
  The TP version depends on the relevant main version in all cases; it may, however, remain unchanged through a number of main versions.
- If service packs and hotfixes are installed, there may even be a selection of TP product versions available for the same TP version.

During device replacement (in HW Config) the TP version is automatically updated. If the TP version is changed and more than one version of the new TP version is available, the latest version is automatically installed. If another product version is preferred, this must be set manually (e.g. selection of V4.1.5.3).

During platform replacement (XML Export/Import) , however, the required technology package including the TP version and, if necessary, the product version must be selected manually after the import.

When a SIMOTION CPU is inserted, the TP CAM (latest TP version and product version) is preset by default.

Special displays in the "version" field:

- "Select" means that no TP product version has been selected; this state occurs when older projects, in which the selection of a specific product version was not supported, are upgraded. If the project is loaded to the CPU without having previously made a selection, then the latest available technology package is loaded automatically.
- "---" means that no version can be determined (e.g. for the TP DCBlib or for older CPU versions < V4.1). If no version can be determined, you must select "---".

Selecting the TP product version

The desired technology package is selected with fine granularity in SIMOTION SCOUT at "Target device" > "Select technology packages ...".

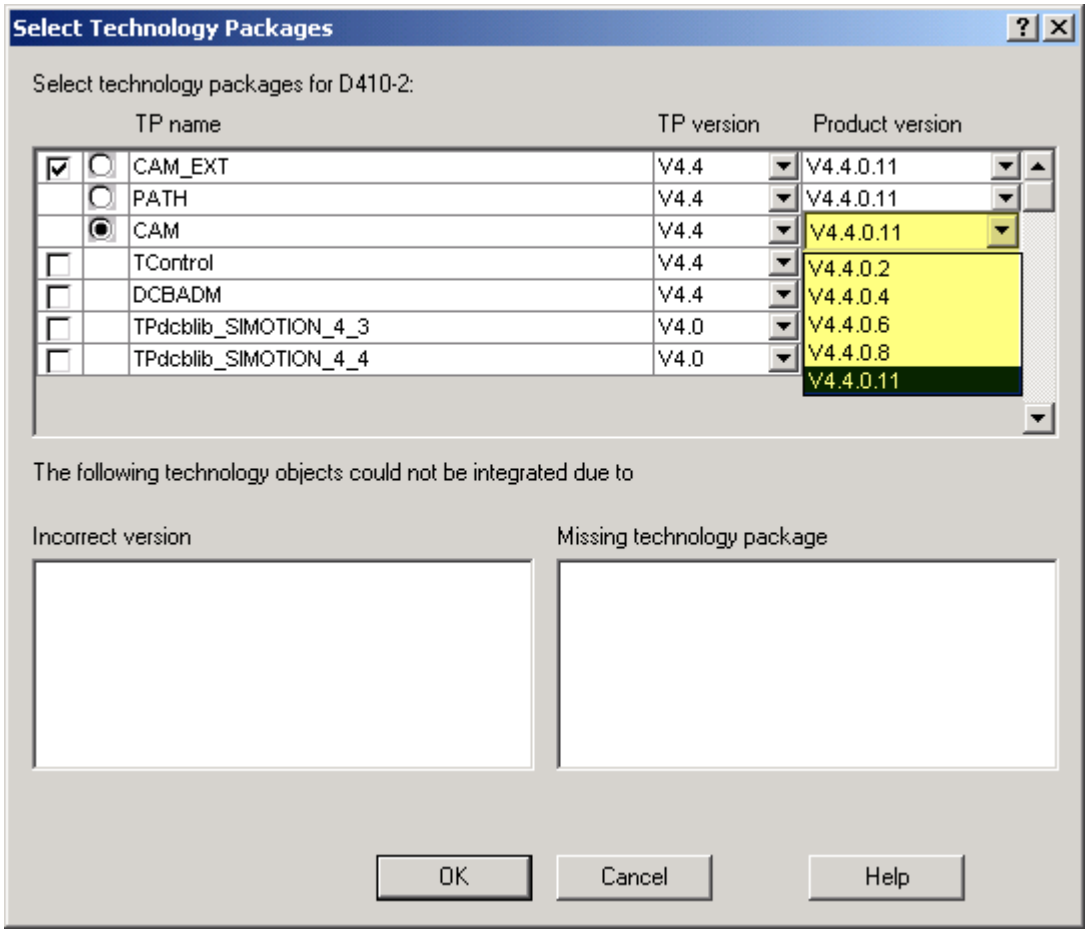


Figure 10-218 Selecting technology packages (based on the example of the D410-2)

Note

Device diagnostics can provide information on which technology package product version has been loaded to a CPU.

Loading technology packages to the target device

Technology packages are only loaded to the target device if no technology package has been loaded so far or if "Load to file system" is executed.

If a technology package version changes, the technology package must be explicitly reloaded to the target device. To do this, proceed as follows:

1. Select "Download project to target system" in SIMOTION SCOUT.
2. Select the "Replace product versions of the technology packages" option at "Additional CPU options" and confirm with "OK".

For further information, please see the online help for SIMOTION SCOUT.

Upgrading the device version of SINAMICS S120 Control Units

Overview

You can upgrade the device versions of SINAMICS S120 Control Units that are connected to the SIMOTION D via PROFIBUS or PROFINET in the SIMOTION SCOUT. The SINAMICS version can only ever be upgraded in a project; it cannot be downgraded.

Note

During device replacement in **HW Config**, the SINAMICS version of the SINAMICS Integrated of the SIMOTION D410-2 is automatically upgraded.

During module replacement in **HW Config**, the selection of a SIMOTION D410-2 module always defines the SIMOTION and the SINAMICS version.

If a SINAMICS S120 Control Unit is connected via PROFIBUS or PROFINET, the SINAMICS version can be selected independently of the SINAMICS Integrated version.

Procedure

To upgrade a SINAMICS drive unit, proceed as follows:

1. Right-click the relevant device, e.g. the SINAMICS S120 CU310-2 DP.
2. Select "Target device" > "Upgrade Device Version/Characteristic" in the context menu. The "Upgrade Device Version/Characteristic" dialog box is displayed. It lists all available firmware versions.

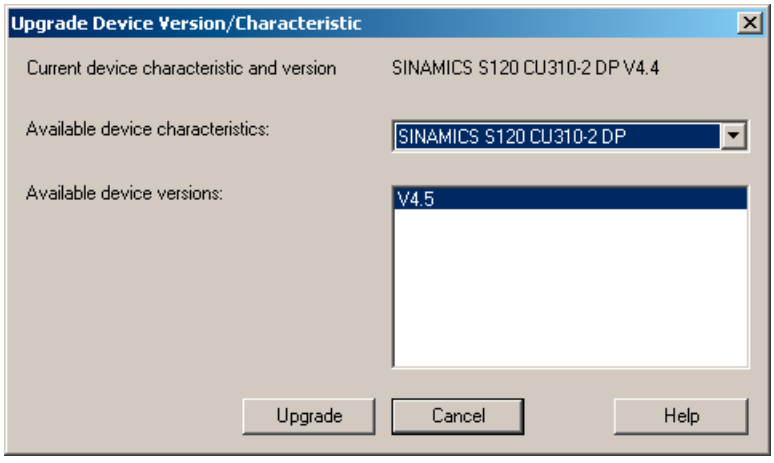


Figure 10-219 Upgrading the device version

3. Select the desired device version/characteristic and click "Upgrade". This upgrades the SINAMICS S120 Control Unit.

Upgrade the libraries

Depending on the configured properties of the libraries used in the project (device-dependent or device-independent), an upgrade of the libraries may be required if the SIMOTION device or the device version changes.

1. Open the LIBRARIES directory in the project navigator.
2. Select a library, right-click to open the context menu, and select "Properties...."
3. Select the "TPs/TOs" tab in the "Properties" window.
4. Select the SIMOTION device and the technology packages for which the library is to be valid.
5. Close the dialog box with "OK".

Note

Please also observe the notes on device-dependencies in the SIMOTION SCOUT online help.

Save project, compile and check consistency

Procedure

1. Save and compile the project from the "Project" > "Save and recompile all" menu.
2. Then perform a consistency check with the "Project" > "Check consistency" menu.

If error messages occur, correct these and repeat the operation.

Note

Note the difference between

- "Save and recompile all" and
- "Save and compile changes"

Save and recompile all

All sources of the entire project are recompiled with this command.

The command is suitable if you are quite sure that all the old data from older SCOUT versions should be removed and replaced with new compilation results.

Use this command if you specifically want to convert a project from an earlier SCOUT version to a newer version. In this way, you take over all error corrections and optimizations.

Save and compile changes

On this command, the whole project is searched for changes. Therefore only the changes are compiled. Use this command for day-to-day operations within a SCOUT version.

Performing a firmware and project update

Upgrading the boot loader on the CompactFlash card

The boot loader may have to be upgraded in the following cases:

- Upgrade of the SIMOTION D
- If a D4x5-2 CF card is to be used for a D410-2 (or vice versa).
- Error corrections and optimizations

Generally, we recommend using the latest boot loader available for the respective CF card.

Upgrading the SIMOTION D410-2 may also render it necessary to upgrade the boot loader of the CF card.

Detailed information on the CF card, boot loader version, SIMOTION D410-2 hardware, and SIMOTION firmware version compatibility relationships can be found in the software compatibility list at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Update - preparatory measures

Upgrading the SIMOTION D410-2

The actions described in this section also apply to downgrading to an older version.

Various options are available for performing a firmware and/or project update on the SIMOTION D410-2:

- Update via CompactFlash card (Page 7540)
- Update via SIMOTION IT web server (Page 7537)
- Firmware and project update using the device update tool (Page 7537)

Note

Upgrading using the device upgrade tool offers a number of advantages (keeping retain data, option of downgrading, no license key handling, etc.).

We therefore recommend using this method for firmware and/or project updates.

Requirement (firmware update)

The current firmware for SIMOTION D can be found on the Internet at Internet address (https://support.industry.siemens.com/cs/ww/en/view/31045047).

Upgrading the SIMOTION D410-2 automatically upgrades the firmware of all connected SINAMICS DRIVE-CLiQ components.

Note

Observe the information in the the Read Me files and the upgrade instructions included in the scope of delivery of new SIMOTION versions.

Use only CF cards that have been approved for use with the SIMOTION D410-2 and whose boot loader version is correct.

You can find the compatibility relationships in the software compatibility list at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

NOTICE
The upgrade operation deletes all project data and parameters from the CF card! Back up the data before starting an upgrade.

Requirement (project update)

You have updated your project and, if necessary, adapted the device type and device version. See section Customizing the project (Page 7525).

Update via SIMOTION IT web server

SIMOTION D410-2 has an integrated web server.

In addition to customized web pages and comprehensive device/diagnostic information, SIMOTION IT web server also allows you to update the firmware and project using a standard PC with a web browser.

You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Upgrade via device update tool (upgrading SIMOTION devices)

Overview

SIMOTION D Control Units and projects can be upgraded using previously created upgrade data.

Performing an upgrade using upgrade data has the following advantages:

- User-friendly creation of upgrade data via SIMOTION SCOUT with the aid of a wizard (at the machine manufacturer's site)
- SIMOTION devices can be upgraded by the machine operator without the SIMOTION SCOUT engineering system
- The machine manufacturer can conveniently send upgrade data via e-mail or post to the machine operator

- There is no need to use license keys, as licenses are retained
- Retain data and unit data is retained when upgrades are performed, even across versions
- An upgrade which has been imported can be discarded again, and the previous configuration restored
- You can upgrade either a single SIMOTION device or multiple devices from one or more SIMOTION projects.
- It is possible to upgrade parts of a configuration only, e.g. Technology Packages only, firmware only, project only, etc.

Handling

Upgrade data is created by the application engineer at the machine manufacturer's premises using SIMOTION SCOUT. The upgrade data can then be handled flexibly depending on both the SIMOTION device in question (SIMOTION C, D, or P) and the customer requirements:

- Creating upgrade data and then copying it to a storage or upgrade medium:
  - CF card
  - USB stick (D4x5/D4x5-2 only; not for D410-2)
  - Upgrade file for the SIMOTION IT web server
- Alternatively, the upgrade data can be created and stored in an archive on the PC, with a view to importing it to an upgrade medium suitable for SIMOTION devices at a later point.
- The process of importing the data to an upgrade medium can be performed at the machine manufacturer's premises; alternatively, if the upgrade archive has been transferred to the machine operator, the service engineer can do this on site.
- The service engineer imports the upgrade data on an operator-guided basis (without any involvement by the application engineer) to the SIMOTION device(s), and upgrades the SIMOTION devices in the process (SIMOTION SCOUT is not required on site).

Note

If a SIMOTION D410-2 is upgraded with an upgrade archive of a SIMOTION D4x5-2, the upgrade and the later downgrade fail. In this case, a correct card image (firmware incl. project) must be loaded to the CF card.

Upgrading with a CF card

1. Switch off the SIMOTION D410-2 to be updated.
2. Insert the CF card into the SIMOTION D410-2.

3. Switch on the SIMOTION D410-2 again. The SIMOTION D410-2 starts to process the upgrade data.
   The SF/BF LED flashes green (0.5 Hz) during the upgrade.
4. Monitor the green flashing of the SF/BF LED.
   - As soon as the upgrade has been completed successfully, the SF LED goes out.
     An automatic power-up with the upgraded configuration is then performed
     (SF/BF LED display then depends on the associated operating state of the device).
   - If the upgrade was not successful, the SF/BF LED flickers red.

Downgrading

If the upgrade has not been successful (for example, the machine is not behaving as desired), the upgrade can be undone as follows:

1. Switch off the SIMOTION D410-2.
2. Set the service selector switch to the switch position B (service mode: downgrade).

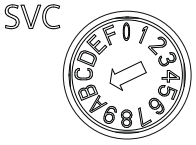


Figure 10-220 Service selector switch, switch position for downgrading

3. Switch the D410-2 on again.
   The flickering green SF/BF LED indicates that restoration is requested.
4. Set the service selector switch back to position "0."
   The system restores the data saved during the upgrade. The data from the upgrade will be deleted.

The downgrade is indicated by a green flashing SF LED (0.5 Hz) and can take several minutes. After successful restoration, the module starts up automatically. (SF/BF LED display will then depend on the respective operating state of the device).
If the restore was not successful, the SF/BF LED will flicker red.

Note

For SIMOTION < V4.4, restoration will start when the module is switched on (step 3). For this, the service selection switch must be rotated to position "0" immediately when the flashing code is displayed (SF/BF LED flashes green with 0.5 Hz).
If the service selector switch is not reset or not reset in good time to "0," this results in fault state "Service selector switch is still set to restore" (SF/BF LED flickers red).
In this case, switch the D410-2 off, reset the service selector switch and switch the D410-2 on again. If the restoration was otherwise successful, the D410-2 boots up with the restored configuration.

Note

The components' firmware is automatically updated, depending on the firmware version on the SINAMICS components and on the CF card. During a firmware update, please heed the messages and alarms in the SIMOTION SCOUT detail window. A firmware update on the SIMOTION D410-2 is signaled by the RDY LED flashing yellow, while on the DRIVE-CLiQ components (TM, SMC, etc.) it is signaled by the RDY LED flashing red/green.

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

Components that require a POWER ON after the firmware update will indicate this through a rapidly flashing RDY-LED. Go offline with SCOUT and switch the 24 V supply off/on (POWER ON) at the respective components for initialization.

Additional references

You will find detailed information for the device upgrade in the *Upgrading SIMOTION Devices* Operating Instructions.

Update via CompactFlash card

Backup of CompactFlash card data

Backing up licenses, retain data, and user data

Prior to upgrading/downgrading firmware, as a precaution we would generally recommend backing up the entire content of the CF card to the PG/PC using the card adapter and Windows Explorer.

How you should proceed to back up and subsequently restore data on the CF card depends on whether the CF data contains any licenses and/or other retain data and user data which will be required again in the future.

Case 1: The CF card contains no licenses and no retain or user data that is still required

You do not need to take any steps in this case. Delete the contents of the CF card and install the firmware as described.

Case 2: The CF card contains licenses (e.g. axis licenses)

Before loading the new firmware, back up the "KEYS" directory to your PC. This can then be copied back to the CF card once you have installed the new firmware.

Note

The license key is stored in the "KEYS" directory on the CF card. When the SIMOTION device starts up for the first time, the license key is saved in the boot sector of the CF card.

A license key saved in the boot sector cannot be deleted by means of a user operation; nor can it be deleted by formatting the CF card or rewriting the boot loader.

If the Keys.txt file is no longer present on the CF card (because the "KEYS" directory has been deleted, for example), it will be written again from the boot sector to the "KEYS" directory while the SIMOTION device is starting up. The license key can be changed at any time (by relicensing, for example). When the device is next started up, the license key will be saved in the boot sector again.

If you lose the license key, you can get it back via the Web License Manager on the Internet, at Internet address (http://www.siemens.com/automation/license). You require the hardware serial number printed on the CF card to do this. In the Web License Manager, you have the option of displaying the associated license key.

Case 3: The CF card contains retain data / user data that is still required in the future

If you are using your application to back up data to the CF card, you must back this up before upgrading the new firmware.

Example:

- Backing up retain data (non-volatile data saved using _savePersistentMemoryData):
  - user\simotion\pmemory.xml
- Backing up SIMOTION IT user files, settings (e.g. trace.xml), task trace data, log files, and Java files (classes, archives, user data system, etc.), stored in the following directories:
  - user\simotion\hmicfg
  - user\simotion\hmi

- Backing up configuration data for modular machines in conjunction with the _activateConfiguration system function, stored in the following directory:
  - install\simotion
- Backing up unit data (data saved on the CF card using _saveUnitDataSet/ _exportUnitDataSet), stored in the following directory:
  - user\simotion\user dir\<unitname>

Note

When a version is changed, you must use the "Save variables" function to back up data saved using _saveUnitDataSet or _exportUnitDataSet in a way that is not dependent on the version. You then have the option of restoring them using "Restore variables."

During an upgrade, the two functions are required only for the backup and restore of unit data records created with _saveUnitDataSet.

Retain and unit data (saved with _exportUnitDataSet) remain valid even after a version upgrade.

Firmware update using a CompactFlash Card

Procedure

Proceed as follows to perform the upgrade:

1. Switch off the power to the SIMOTION D410-2.
2. Remove the CF card from the SIMOTION D410-2 and insert it into the CF card adaptor of your PC.
3. Open Windows Explorer. The CF card must be visible as a removable data carrier in the Windows Explorer under an arbitrary drive letter.
4. If necessary, back up the licenses, retain data and user data on the CF card to your PC (see Backup of CompactFlash card data (Page 7540)).
5. Delete all the data from the CF card.
6. Unzip the firmware file to the CF card using a ZIP file utility such as WinZip. Always maintain the file structure when setting up the unpacking tool.
7. Copy the data saved in step 4 back to the appropriate folder structure on the CF card.
8. Remove the CF card from the CF card adapter on your PG/PC.
9. Insert the CF card into the SIMOTION D410-2.
10. Switch on the power supply for the SIMOTION D410-2. The new firmware is loaded from the CF card to the SIMOTION D410-2 module.

Upgrading SINAMICS

Depending on the settings, the SINAMICS components are also automatically upgraded to the component version of the CF card with a firmware update of the SIMOTION D.

In order for a FW update to be performed for all components, the components must be correctly connected in accordance with the configured topology.

The CONTENT.TXT file in the main directory of the CF card contains the component version.

Upgrading the firmware of SINAMICS components automatically

When starting up, the system automatically upgrades or downgrades all DRIVE-CLiQ components to the version of the component firmware on the CF card. Components that cannot be downgraded to the component firmware version on the CF card (for example, old firmware on the CF card and new components to which the old firmware cannot be loaded) retain their firmware version. The resulting firmware version combinations are always functional.

Note

The components' firmware is automatically updated, depending on the firmware version on the SINAMICS components and on the CF card. During an FW update, please take note of the messages and alarms in the SIMOTION SCOUT detail window. A firmware update on the SIMOTION D410-2 is signaled by the RDY LED flashing yellow, while on the DRIVE-CLiQ components (TM, SMC, etc.) it is signaled by the RDY LED flashing red/green.

- FW update running: RDY LED flashes slowly (0.5 Hz)
- FW update complete: RDY LED flashes quickly (2 Hz), POWER ON required

Components requiring POWER ON following a firmware update signal this by means of the fast flashing RDY LED. Go offline with SCOUT and switch the 24 V supply to the relevant components off/on (POWER ON) to initialize.

Updating the firmware of the SINAMICS components

The SINAMICS components' firmware is updated automatically, depending on the setting of parameter p7826.

- p7826 = 0: Upgrade/downgrade deactivated
- p7826 = 1: Upgrade and downgrade (factory setting)
- p7826 = 2: Upgrade only

Note

The automatic FW update via p7826 = 1 (upgrade and downgrade) must not be deactivated when Safety Integrated is used.

If you are updating the firmware manually, proceed as follows:

1. Select the SINAMICS component in the Project Navigator, e.g. SINAMICS Integrated.
2. Double-click "Overview" in the Project Navigator.
   The "SINAMICS_Integrated - Overview" dialog box opens with a list of available drive objects.
3. Click "Version overview" to open the list of connected SINAMICS components.
4. Go online and select the devices whose firmware you wish to update.
   The list displays the current firmware version of the devices.

5. Click "Firmware update" to download the new firmware to the devices. To do so, you must select all components whose firmware is to be updated.
6. When the firmware update is complete, switch the 24 V power supply off and back on again. The device is now ready for operation.

Note

The SINAMICS components must be configured for a firmware update to take place. The firmware cannot be updated if the components have not been configured.

You can also update the firmware using the expert list. See the *SINAMICS S120* Commissioning Manual for a description of how to do this.

Download project to target system

Once all the changes required for upgrading your project have been made, you must download the project to the SIMOTION D410-2.

Requirement

The firmware required is located on the CompactFlash card; for information, refer to the section titled Firmware update using a CompactFlash Card (Page 7542).

You have recompiled the project and checked it for consistency. See Section Save project, compile and check consistency (Page 7535).

Procedure

1. Save and compile the project.
2. Click "Connect to selected target systems" to establish a connection to the target system.
3. Execute "Download project to target system" and then "Copy RAM to ROM" to download the upgraded project to the CompactFlash card as well.
4. Because of the automatic follow-up configuration in the SINAMICS Integrated drive, you must now execute "Load CPU / drive unit to PG".
5. Save the project.

Note

When upgrading SINAMICS drive units (e.g. SINAMICS Integrated) only the p parameters (setting parameters) are loaded into the upgraded project. The r parameters (monitoring parameters) are not loaded.

The r parameters in the drive unit are derived or calculated from an automatic subsequent parameterization and must therefore be uploaded to the project.

To do this, execute "Load CPU/drive unit to PG".

If the upload cannot be performed, this can lead to inconsistencies in the drive parameterization dialog boxes.

SIMOTION CompactFlash card

Changing the CompactFlash card

Requirement

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF.
Comply with the ESD rules.

Procedure

To change the CF card, proceed as follows:

1. Switch off the power supply.
2. Remove the CF card from the plug-in slot on the SIMOTION D410-2.
   To do this, grasp the gripping cavity between your thumb and forefinger and pull the card out.
3. Gently insert the new CF card into the empty plug-in slot until it clicks into place. The direction of insertion of the CF card is indicated by an arrow located on both the plug-in slot and the CF card.
   The properly inserted card does not protrude from the SIMOTION D410-2 housing.
4. Switch the power supply on again.

Writing to a CompactFlash card

Overview

You have the following options of writing data to a CompactFlash card:

- Writing to a CompactFlash card that is inserted in a SIMOTION D
  The PG/PC must be online to SIMOTION D410-2 in order to execute this function.
- Writing to a CompactFlash card without a SIMOTION D module
  This function requires a CompactFlash card adapter.

Note

CompactFlash cards are always shipped in formatted state. It contains the SIMOTION Kernel (SIMOTION D firmware).

Trouble-free operation of the CompactFlash card can only be guaranteed if you do not modify its partitioning.

Writing to a CompactFlash card that is inserted in the SIMOTION D

The CompactFlash card can be used as storage volume for technology and user data (programs, configuration data and parameters) from the "volatile data" area. Proceed as follows:

1. Establish the connection between the SIMOTION D410-2 and the PG/PC (see Section Creating a project and configuring the communication (Page 7367)).
2. Select the "Copy RAM to ROM" command in SIMOTION SCOUT to write the data to your CompactFlash card.

Writing to a CompactFlash card without a SIMOTION D module

A CompactFlash card adapter must be installed on the PG/PC and can be used to write data to the CompactFlash card. Always save your project data to CompactFlash card using the PG/PC before you update the SIMOTION firmware, for example.

Note

Do not use any of the onboard tools in Windows to modify or delete files which were written to the CompactFlash cardCompactFlash card using the "Copy RAM to ROM" function in SIMOTION SCOUT. Such actions may irrevocably destroy your project.

Formatting the CompactFlash card

You can format a faulty CF card, for example.

Before formatting the CF card, please observe the notes in Section Backup of CompactFlash card data (Page 7540).

Proceed as follows to format the CF card:

1. Insert the CF card into a CF card adapter connected to your PG/PC.
2. Format the CF card in Windows (FAT, FAT16 or FAT32 file system).
   The CF card is formatted.
3. If the CF card boot loader is also defective, you will have to rewrite it.

Note

The CF card may not formatted with NTFS.

The following formats are permitted: FAT, FAT16 and FAT32.

Because of the improved memory utilization on the CF card, FAT32 formatting is preferable. SIMOTION D410-2 CF cards are supplied as standard with FAT32 formatting.

Boot loader on the CompactFlash card

Writing a boot loader

A boot loader may need to be written in the following situations:

- A new boot loader is required for the used SIMOTION D410-2 firmware version.
- A new boot loader is required for the used SIMOTION D410-2 hardware.
- The boot loader is defective.
- A D4x5-2 CF card is to be used for SIMOTION D410-2.

The boot loader version can be fetched using SIMOTION SCOUT device diagnostics. If this is not possible, the boot loader version may be incorrect.

A potential fault description could be as follows:
The SIMOTION D410-2 does not power up, the RDY LED flashes red at 0.5 Hz and the RUN/STOP LED lights up red, or all the LEDs remain off.
In this case, replace the boot loader version with the current version.

Use the "Options > Write boot sector..." function to write the boot loader version in SIMOTION SCOUT to the CF card.

Note

You require PG/PC administrator rights to write to the boot sector. If you do not have administrator rights on your PG/PC, an administrator can enter an administrator login for you to use this function at "Options" > "Settings" > "Rights."

Detailed information on the CF card, boot loader version, SIMOTION D410-2 hardware, and SIMOTION firmware version compatibility relationships can be found in the software compatibility list at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Note

Note that the CF cards for SIMOTION D4x5-2 and D410 can have a different boot loader!

Recommended method of handling CompactFlash cards

Handling CF cards correctly

Note the following points when working with the CF card:

- The CF card may only be inserted or removed when the system is de-energized.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF.
Comply with the ESD rules.

- CF cards are not designed to be rewritten as many times as the user wishes.
  With this in mind, you should avoid writing user data from the application to the CF card cyclically. Depending on the system, a write operation from the application may trigger one or more write operations on the CF card.
  We therefore recommend that the user program does not make more than 100,000 write accesses over the estimated service life of the application.
- Never switch off the SIMOTION D Control Unit during write accesses to the CF card. If the SIMOTION D Control Unit is switched off during write accesses, this can result in destruction of the data and, in the worst case, to damage to the file system (FAT Table = directory) on the CF card.
  If the FAT table is destroyed, the CF card will have to be reformatted and the firmware/user data reloaded. During this process the licenses remain on the CF card.
  The FAT table can be destroyed if updating the FAT table is interrupted by switching off the SIMOTION D Control Unit. The FAT table is updated, for example, by functions such as **_exportUnitDataSet**, copy RAM to ROM, or save SINAMICS NVRAM data via p7775.
  The FAT table can also be destroyed if you pull a CF card out of a CF card adapter while Windows is accessing the CF card.
- The following functions do not update the FAT table:
  - **_saveUnitDataSet** (writing the data to an existing file)
  - **savePersistentMemoryData** (as of V4.4: writing the data to an existing file)

  Once the backup files have been created with **_saveUnitDataSet** / **savePersistentMemoryData**, this also results in updating the FAT table.

Card reader for CompactFlash cards

Because of the quickly changing market and the large differences in the quality of card readers, no specific recommendation can be made.

If problems occur identifying the CF card, this may be due to an incorrect power up of the card reader.

License key on the CF card

Depending on the type and number of runtime functions used in the project, licenses must be purchased as part of the licensing procedure for SIMOTION.

For further information on the licensing of runtime functions, see (https://support.industry.siemens.com/cs/en/en/view/42014324)

The licenses required for SIMOTION D are assigned to an individual license key for the CF card (license key cannot be transferred to another CF card).

The license key must be stored in the \KEYS\SIMOTION directory in the keys.txt file on the CF card.

Backup in the boot sector

The license key is stored in the "KEYS" directory on the CF card. When the SIMOTION D Control Unit is ramped up for the first time, the license key is backed up in the boot sector of the CF card.

A license key saved in the boot sector cannot be deleted by means of a user operation; nor can it be deleted by formatting the CF card or rewriting the boot loader.

If the keys.txt file is no longer present on the CF card (e.g. because the "KEYS" directory has been deleted), it will be written again from the boot sector to the "KEYS" directory while the SIMOTION D Control Unit is ramping up. The license key can be changed at any time (e.g. through relicensing). At the next power-up, the license key is backed up in the boot sector again.

Loss of the license key

In the event of the loss of a license key, a copy can be obtained via the Web License Manager at the following Internet address (http://www.siemens.com/automation/license). You require the hardware serial number printed on the CF card to do this. You can display the associated license key in the Web License Manager. The license file (Keys.txt) is also offered as download.

Notes for Keys.txt

- If a firmware card image is unpacked on the CF card, a KEYS directory is not available per default.
- If the CF card ramps-up **without a KEYS directory** in a controller, an empty file structure (\KEYS\SIMOTION) is created.
- The keys.txt must be stored in this directory (\KEYS\SIMOTION\keys.txt).
- If the directory contains a keys.txt with an invalid license key, this is deleted from the keys.txt when the controller ramps up, i.e. the keys.txt file exists, but is empty.
- If the directory contains a keys.txt with a valid license key, this is also stored in the boot sector of the CF card as backup when the controller ramps up.

- The keys.txt can be written online via SCOUT, with a CF card reader, or via a set up remote access (FTP).
- When licensing via the Web License Manager, you receive a keys.txt that you must store on the CF card under \KEYS\SIMOTION
- It is also possible to create this text file yourself (without formatting) with a text editor (e.g. Notepad). In this case, make sure that only the license key and a line break (CR/LF) is contained in the file.
- If the SF LED flashes slowly in red (0.5 Hz) after the control has ramped up, then either the keys.txt file is faulty (error in the license key, formatting or file/directory name) or the licensing is not adequate for the used objects that require a license.

10.1.2.9 Diagnostics

Diagnostics via LED displays

Overview

The status LEDs display the operating states or fault states of SIMOTION D410-2. They do so by illuminating, flashing, or flickering in different colors.

Arrangement of LED displays

The front side of the SIMOTION D410-2 has four LED displays arranged vertically one above the other.

Figure 10-221 LED displays: D410-2 DP (pictured on the left), D410-2 DP/PN (pictured on the right)

Legend of the LED states

The LED displays can assume the following illumination states.

Table 10-129 Table with explanation of the LED states

Symbol	Meaning
1	LED static ON
0	LED off
0.5/1	LED flashes at 0.5 Hz:
2/1	LED flashes at 2 Hz:
Λ	LED flickers
x	Any LED state

LED displays

The two tables below provide an overview of all relevant LED display combinations. Every LED can illuminate in yellow, red, or green. The color which corresponds with the LED signal state is also defined.

Table 10-130 SIMOTION D410-2 DP and D410-2 DP/PN: Diagnostics by means of LED display

Meaning	Display priority [1]	RDY	RUN/STOP	OUT>5V OUT>5V/SY [2]	SF/BF:
States during power-up					
Hardware reset Power-up of the D410-2 without a CF card or power-up with a CF card (CF card with incorrect / missing / faulty boot loader or without a valid operating system)	1	1 (yellow)	1 (yellow)	1 (yellow)	1 (yellow)
Firmware error • No or incorrect firmware on the CF card • File system of the CF card is destroyed (e.g through power off during writing operation)	x [3]	2/1 (red)	0	0	2/1 (red)
Firmware checked (checksum incorrect)	x [3]	0.5/1 (red)	0	0	0.5/1 (red)
Firmware being loaded	x [3]	Λ (yellow)	0	0	1 (red)
SIMOTION states					
Write/read SIMOTION access to CF card	1	Λ (yellow)	x	x	x

Meaning	Display priority [1]	RDY	RUN/STOP	OUT>5V OUT>5V/SY [2]	SF/BF:
"FAULT" state (F state) Fault to which the user program (SIMOTION) cannot respond (e.g. overtemperature). The following actions may be required to rectify the fault: • Power OFF/ON • Check of the CF card • Re-commissioning • Replace the SIMOTION D410-2	2	Λ (red)	Λ (red)	Λ (red)	Λ (red)
DCP flashing (for interface X150, X127) This function is used to check the correct assignment to a module and its interfaces. DCP flashing of the module is activated in HW Config under "Target system" > "Ethernet" > "Edit Ethernet node" > "Browse" button > "Flashing" button	3	x	x	x	2/1 (green)
PROFIBUS DP interface bus error • PROFIBUS master: At least one slave is missing • PROFIBUS slave: No parameter assignment master available	4	x	x	x	2/1 (red)
Communication error PROFINET [7]	4	x	x	x	2/1 (red)
An interrupt that can be acknowledged (alarm, message, note) is pending	5	x	x	x	1 (red)
Underlicensing of technology/option objects	6	x	x	x	0.5/1 (red)
SIMOTION ready for operation	6	x	Static/flashing (green or yellow)	x	x
RUN	6	x	1 (green)	x	x
Transition from STOP/STOPU to RUN	6	x	2/1 (green)	x	x
STOP/STOPU	6	x	1 (yellow)	x	x
Transition • from RUN to STOP/STOPU • STOP to STOPU • STOPU to STOP	6	x	2/1 (yellow)	x	x

Meaning	Display priority [1]	RDY	RUN/STOP	OUT>5V OUT>5V/SY [2]	SF/BF:
Service operating state (axis control panel in the STOPU/ measuring function)	5	x	2/1 (yellow/green)	x	x
General reset requested by SIMOTION D410-2 or with the mode switch	5	x	0.5/1 (yellow)	x	x
General reset in progress	5	x	0	x	0
General reset stopped (STOP)	6	x	1 (yellow)	x	x
States of the SINAMICS Integrated (LED RDY)					
Updating the firmware of the DRIVE-CLiQ components	6	0.5/1 (yellow)	x	x	x
Firmware update of the connected DRIVE-CLiQ components has been completed (Power OFF/ON of the upgraded/downgraded devices is required)	6	2/1 (yellow)	x	x	x
Detection of the components via LED **Note**: The displayed color combination depends on the p0124[0] parameter =1 yellow/green flashing =0 yellow/red flashing	6	2/1 (yellow/green) (yellow/red)	x	x	x
Commissioning/reset	6	0.5/1 (green)	x	x	x
Write/read SINAMICS access to CF card	6	Λ (yellow)	x	x	x
SINAMICS Integrated ready for operation	6	1 (green)	x	x	x
SINAMICS Integrated in fault state (Check parameterization/configuration)	6	2/1 (red)	x	x	x
Underlicensing SINAMICS functions [6]	6	x	x	x	0.5/1 (red)
SIMOTION IT service mode / switch position 8 for 120 minutes.	6	x	x	x	0.5/1 (red)
Backing up and restoring diagnostic data and non-volatile SIMOTION data					
Backing up diagnostic data and non-volatile SIMOTION data (backup running)	4	x	Λ (yellow)	x	x
Backing up diagnostic data and non-volatile SIMOTION data (backup completed)	4	x	Λ (green)	x	x
Request: "Restore non-volatile data" (with switch position "A")	4	x	x	x	Λ (green)

Meaning	Display priority [1]	RDY	RUN/STOP	OUT>5V OUT>5V/SY [2]	SF/BF:
Upgrading SIMOTION devices (device update tool)					
Restoration requested		x	x	x	Λ (green)
Upgrade/downgrade running		x	x	x	0.5/1 (green)
Upgrade/downgrade completed with error		x	x	x	Λ (red)
Upgrade/downgrade completed without error		x	x	x	0 [4]
Encoder power supply [5]					
The electronics power supply is missing or outside the permissible tolerance range. Power supply ≤ 5 V. Use of an encoder with 5 V power supply.		x	x	D410-2 DP: 0 D410-2 DP/PN: 0 1 2/1 (green)	x
Electronics power supply for measuring system available. Power supply >5 V. **Notice:** Always ensure that you can operate the connected encoder on a 24 V power supply (e.g. HTL encoder). Operation of a 5 V encoder on the 24 V encoder supply may result in the destruction of its electronic components! This setting can be selected in the expert list of the drive in parameter p0405.1.		x	x	D410-2 DP: 1 (yellow) D410-2 DP/PN: 0.5/1 1 2/1 (yellow)	x

1) Priority of the displays: The displays always visualize the state assigned the highest priority. "1" has the highest priority. The state of the next lower priority class is displayed after the cause of the previous signal state was eliminated. If a state does not have any specified priority, no other state other than the associated state can occur.

2) Labeling of LEDs:
D410-2 DP: OUT>5V
D410-2 DP/PN: OUT>5V/SY

3) States occur in succession during power-up.

4) The upgrade or downgrade has been completed when the SF/BF LED goes out. The device then powers up automatically in the upgraded or downgraded configuration (the SF/BF LED display then depends on the operating state of the device).

5) The "OUT>5V" or "OUT>5V/SY" LED display indicates whether the encoder supply level is > 5 V. In the case of SIMOTION D410-2 DP/PN, not only is the configured encoder supply indicated, but also the state of PROFINET interface synchronization to the send cycle clock. See the following table.

6) In the case of SINAMICS licenses (e.g. SINAMICS DCB Extension), underlicensing of SINAMICS Integrated via the flashing SF-LED is indicated on the SIMOTION D Control Unit. An entry is also made in the diagnostic buffer and the underlicensing is displayed in the license dialog box of SIMOTION SCOUT. The licensing is effected (like for SIMOTION licenses) via SIMOTION SCOUT or via the SIMOTION license key on the CF card.

7) PROFINET communication error
As IO controller:

- Failure of a connected I/O device
- At least one of the assigned I/O devices cannot be addressed
- Incorrect or no configuration

As I-Device:
The LED flashes until at least one controller has correctly established communication with this I-Device.
Possible causes:
- IP address is incorrect
- Incorrect configuration / parameterization
- IO controller not available / switched off, but Ethernet connection is established.
- In Shared IDevice operation: All configured IO controllers are not available / switched off, but Ethernet connection is established (link established to a neighboring device)
- Incorrect or missing device name
- The response monitoring interval has elapsed
- The CPU is I-Device and the communication to the higher-level controller fails

Table 10-131 SIMOTION D410-2 DP/PN: SYNC state of the PROFINET IO interface diagnosed via the OUT>5V/SY LED display

Meaning	SYNC state LED display[1)]	
	Encoder supply ≤ 5V	Encoder supply > 5V
The PROFINET interface has not synchronized yet to the send cycle clock of PROFINET IO with IRT, or PROFINET IO with IRT has not been configured (e.g. only PROFINET IO with RT or TCP/IP communication). If IRT data has been configured for SIMOTION, the PROFINET interface generates a local substitute cycle clock as long as there is no synchronization to the send cycle clock of PROFINET IO with IRT: SIMOTION task system has synchronized to the local substitute cycle clock of the PROFINET interface. SINAMICS Integrated and the external isochronous DP interface are synchronized to the local substitute cycle clock of the PROFINET interface.	0	0.5/1 (yellow)
The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT. If isochronous IRT data has been configured for SIMOTION, then the following applies: The task system of SIMOTION has synchronized to the send cycle clock of PROFINET IO with IRT. SINAMICS Integrated and the external isochronous DP interface are synchronized to the send cycle clock of PROFINET IO with IRT. If no isochronous IRT data has been configured for SIMOTION, this state indicates that only the PROFINET interface is synchronized to the send cycle clock of PROFINET IO with IRT. The PROFINET interface is then used to forward the IRT data.	1 (green)	1 (yellow)
The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT If IRT data has been configured for SIMOTION, then the following applies: The task system of SIMOTION has synchronized to the send cycle clock of PROFINET IO with IRT. SINAMICS Integrated and the external isochronous DP interface are not yet synchronized to the send cycle clock of PROFINET IO with IRT.	2/1 (green)	2/1 (yellow)

1) How the colors are assigned differs from the SYNC color code used for other SIMOTION devices.

Additional references

You will find detailed information in the *Upgrading SIMOTION Devices* Operating Instructions.

LED displays of the PROFINET interface

The PROFINET ports X150 P1 and P2 have 2 integrated LEDs each for displaying link and activity.

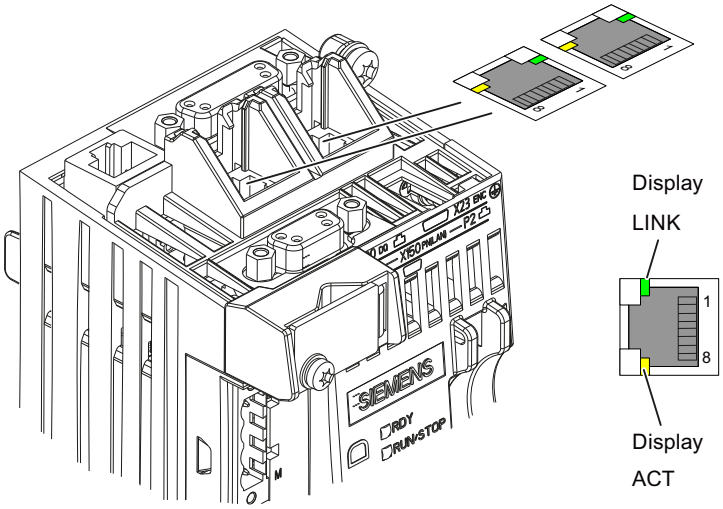


Figure 10-222 PROFINET ports of the D410-2 DP/PN

Table 10-132 State of the Link and Activity LEDs

LED	State	Meaning
LINK	OFF	No or faulty connection
	Lights up green	Transfer rate 10 or 100 Mbit/s: A different device is connected to port x and a physical connection exists
ACT	OFF	No data exchange
	Flickers yellow	Data exchange: Data is being received or sent at port x.

Diagnostic data and non-volatile SIMOTION data

Overview

On the basis of simple operations (e.g. by setting the switch position) and without the need for the SIMOTION SCOUT engineering system, you can

- Back up diagnostic data, including non-volatile SIMOTION data (retain data) on a CF card:
  - Save diagnostic data during running operation (Page 7559)
  - Save diagnostic data during the startup (Page 7560)
- Back up HTML pages (including the most up-to-date content) to the CF card for diagnostic purposes; for information, see Section Diagnostics via HTML pages (Page 7562).
- Restore backed-up non-volatile SIMOTION data (retain data); for information, see Section Delete/restore non-volatile SIMOTION data (Page 7564).

Backup of diagnostic data and non-volatile SIMOTION data

Diagnostics data

Following a fault on a SIMOTION device, diagnostic data (e.g. diagnostic buffer content, up-to-date content of HTML pages, etc.) can provide important information on the cause of the fault. For this purpose, data can be backed up to the CF card with a "simple operator action" (e.g. via service selector switch or DIAG button on the SIMOTION D410-2).

You then have the following options for the diagnostic data:

- Fetching them from the CF card using a card reader
- You can load it with the SIMOTION IT web server or by FTP

You can also use them for diagnostic purposes or provide Technical Support with them for evaluation purposes.

Various options are available to you for backing up diagnostic data:

- Backup during operation (in STOP/STOPU/RUN operating state)
  - with SIMOTION IT web server;
    The web server also offers the possibility of fetching the diagnostic data online.
  - Via the DIAG button
  - Via the service selector switch
- Backup during the module startup
  - Via the DIAG button
  - Via the service selector switch
  - Controlling diagnostic data creation using an INI file stored on the CF card

Non-volatile SIMOTION data (retain data)

In addition to the diagnostic data, non-volatile SIMOTION data (retain data) is also saved on the CompactFlash card. Use the function when the non-volatile SIMOTION data has not been saved on the CompactFlash card using the _savePersistentMemoryData system function but you wish to restore the non-volatile SIMOTION data after a CPU has been replaced.

Note

While the non-volatile SIMOTION data is stored as a "PMEMORY.XML" backup file in the "...USER\SIMOTION" directory using the _savePersistentMemoryData system function, backing up diagnostic data and non-volatile data stores the data in the "...\USER\SIMOTION\HMI\SYSLOG\DIAG" directory.

Save diagnostic data during running operation

Options

The advantage of backing up diagnostic data and non-volatile SIMOTION data during operation is that enhanced diagnostic information via HTML pages and TO alarm information is available.

Data are backed up:

- Using "Diagnostics > Diagnostics Files" in SIMOTION IT web server; see Section Back up diagnostic data and non-volatile SIMOTION data via the web server (Page 7566)
- Via the DIAG button
- Via the service selector switch

Backup of the data via the DIAG pushbutton (preferred solution)

The diagnostic data and non-volatile SIMOTION data can be created in the STOP, STOPU, and RUN operating states.

1. Press the DIAG pushbutton.
   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card.
   The backup task progress is displayed by the RUN/STOP LED flickering yellow.
2. The backup is finished when the RUN/STOP LED flickers green.
   Switch off the SIMOTION D410-2.
3. Remove the CF card.

Backup of the data using the service selector switch (alternative)

The positions of the mode switch are not relevant, i.e. the set operating mode remains unchanged.

The diagnostic data and non-volatile SIMOTION data can be created in the STOP, STOPU, and RUN operating states.

1. Set the service selector switch to "Diagnostics" ("D" position).

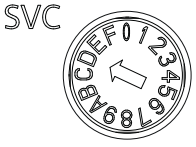


Figure 10-223 Service selector switch (position D)

   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card.
   The backup task progress is displayed by the RUN/STOP LED flickering yellow.
2. The backup is finished when the RUN/STOP LED flickers green.
   Switch off the SIMOTION D410-2.
3. Remove the CF card and reset the service selector switch to its original setting.

Save diagnostic data during the startup

Options

Backing up diagnostic data and non-volatile SIMOTION data during power-up provides you with diagnostic information **without HTML pages / TO alarm information**.

"Backing up during startup" is particularly advisable for SIMOTION devices that cannot run or have crashed.

Diagnostic data and non-volatile SIMOTION data are backed up

- Via the service selector switch
- Via the DIAG button
- Using an INI file saved on the CF card.

Backup of the data using the service selector switch (alternative)

The positions of the mode switch are not relevant, i.e. the set operating mode remains unchanged.

1. Set the service selector switch to "Diagnostics" ("D" position).

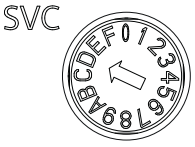


Figure 10-224 Service selector switch (position D)

2. Switch the SIMOTION D410-2 off and on again.
3. Wait for the device to ramp up.
   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card during power-up provided this is still possible and is not prevented by hardware defects, etc.
   The backup task progress is displayed by the RUN/STOP LED flickering yellow.
4. The backup is finished when the RUN/STOP LED flickers green.
   Switch off the SIMOTION D410-2.
5. Remove the CF card and reset the service selector switch to its original setting.

Backup of the data via the DIAG pushbutton (alternative)

1. Switch off the SIMOTION D410-2.
2. Press the DIAG button and hold it down. Switch on the SIMOTION D410-2 again.
3. Wait for the device to ramp up.
   The diagnostic data and non-volatile SIMOTION data are backed up to the CF card during power-up provided this is still possible and is not prevented by hardware defects, etc.
   The backup task progress is displayed by the RUN/STOP LED flickering yellow.
4. The backup is finished when the RUN/STOP LED flickers green.
   Release the DIAG pushbutton and switch the SIMOTION D410-2 off.
5. Remove the CF card.

Note

When backing up data during power-up, the DIAG button must be held down until the backup is completed. As this can easily take 20-30 seconds, it is better to use the service selector switch with switch position "D" for a backup during power-up.

INI file in the main directory of the CF card

1. Use a text editor (such as Notepad) to create a file called **simotion.ini**
2. Add the following text: **DIAG_FILES=1**
   You must use a text editor and may not use any formatting in the text.
3. Copy the simotion.ini file to the main directory of the CF card.
4. Insert the CF card into the module, which is switched off.
5. Switch the SIMOTION D410-2 on and allow the SIMOTION device to start up.
   The diagnostic data and non-volatile SIMOTION data are backed up to the data medium during startup provided this is still possible and is not prevented by hardware faults, etc.
   The backup task progress is displayed by the RUN/STOP LED flickering yellow.
6. The backup is finished when the RUN/STOP LED flickers green.
   Switch off the SIMOTION D410-2.
7. Remove the CF card.

Note

To suppress startup in diagnostics mode again, you must delete the simotion.ini file from the CF card.

Storing the diagnostic data and non-volatile SIMOTION data

You can find diagnostic data and non-volatile SIMOTION data on the CF card in the \USER \SIMOTION\HMI\SYSLOG\DIAG directory.

Copy these data and transfer them to Technical Support when requested to do so. A standard card reader can be used to transfer the diagnostic data from the CF card via standard SIMOTION IT web server pages or via FTP.

The following data are stored:

Table 10-133 Diagnostic data on the CF card

File	Application
DIAGBUF.TXT	Diagnostic buffer in a simple text format: Numerical values; no specific plain text. A text editor is used for evaluation purposes.
PMEMORY.XML	Non-volatile SIMOTION data (retain data) An operator action can be used to restore the backed up non-volatile SIMOTION data after a CPU has been replaced See Delete/restore non-volatile SIMOTION data (Page 7564)
TOALARMS.TXT	Text file containing the pending TO alarms. Only TO IDs, alarm numbers, and associated HEX values. **Note:** The TO alarms are only created if diagnostic data has been created during operation (STOP/STOPU/RUN).
HTML page	If the diagnostic data are backed up, the URLs are requested from the text file (DIAGURLS.TXT) and stored as HTML pages together with their content. See Diagnostics via HTML pages (Page 7562) **Note:** The HTML pages are only stored if diagnostic data are created during operation (STOP/STOPU/RUN).
Other files	All other files stored in the directory are only of relevance to Technical Support.

Note

Use HTML pages if you wish to back up diagnostic data in plain-text format. HTML pages enable user-friendly diagnostics. In addition to the standard SIMOTION IT web server pages, you have the option of creating your own HTML pages (e.g. for the axis status or for machine diagnostics). Customized diagnostics pages are particularly suitable for application problems, as you can define the contents yourself.

Diagnostics via websites

In the "DIAGURLS.TXT" text file, located in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory, you can specify HTML files whose state will be stored on the CF card when diagnostic data is created during operation. For example, "devinfo.mwsl" must be entered for the "devinfo.mwsl" website.

Since the pages in question are stored together with their most up-to-date contents, this enables the latest status information regarding the SIMOTION device, as well as the machine/system, from the point at which diagnostic data was created (e.g. when the service selector switch was activated) to be archived.

In addition to the standard SIMOTION IT web server pages, it is possible to store customized pages. Information on creating pages of this type can be found in, for example, an FAQ of the Utilities & Applications.

A tutorial on the creation of user-defined websites is also contained on the Utilities & Applications-DVD.

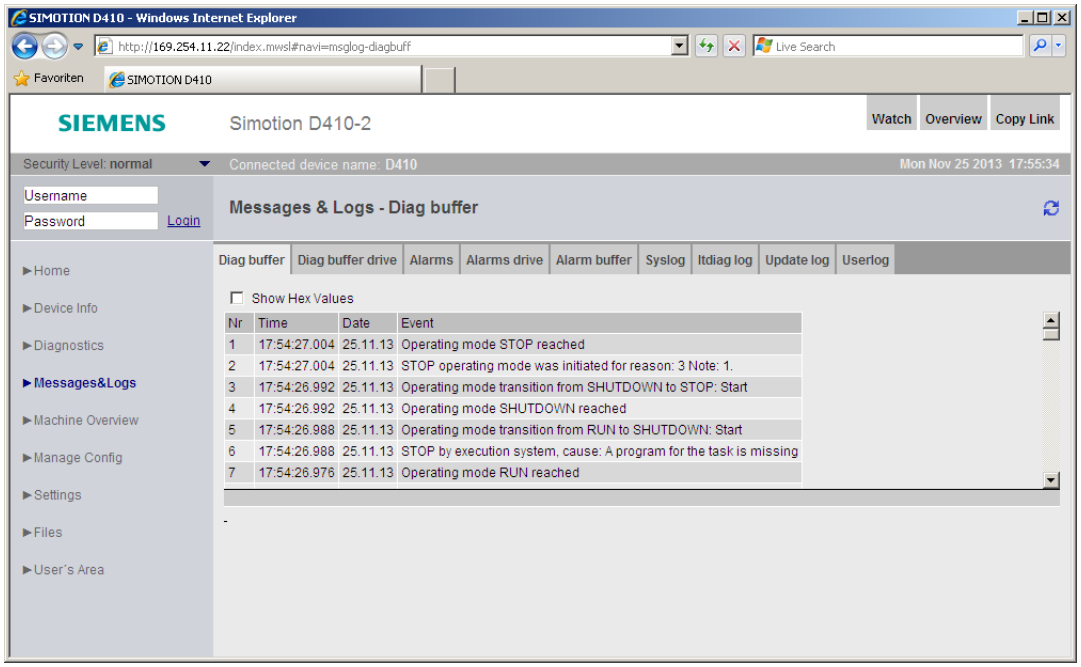


Figure 10-225 Diagnostic buffer at the point when diagnostic data was created

The following points must be noted for the DIAGURLS.TXT file:

- A DIAGURLS.TXT file containing the standard websites is automatically created if you have not stored your own DIAGURLS.TXT file.
- Standard websites are entered "without" specifying a path (e.g. "devinfo.mwsl" for the standard website "devinfo.mwsl").
- User websites (such as "user.mwsl") in the \USER\SIMOTION\HMI\FILES directory on the CF card must contain the FILES/ path specification.
- If you have created subfolders (e.g. "myfolder" in the FILES directory), these must also appear in the path.
- Only one file name may be used per line.
- Empty lines are not permitted (an empty line will be interpreted as the end of the list).

- No distinction is made between upper-case and lower-case letters.
- It does not matter whether you use "\" or "/" in the path name.

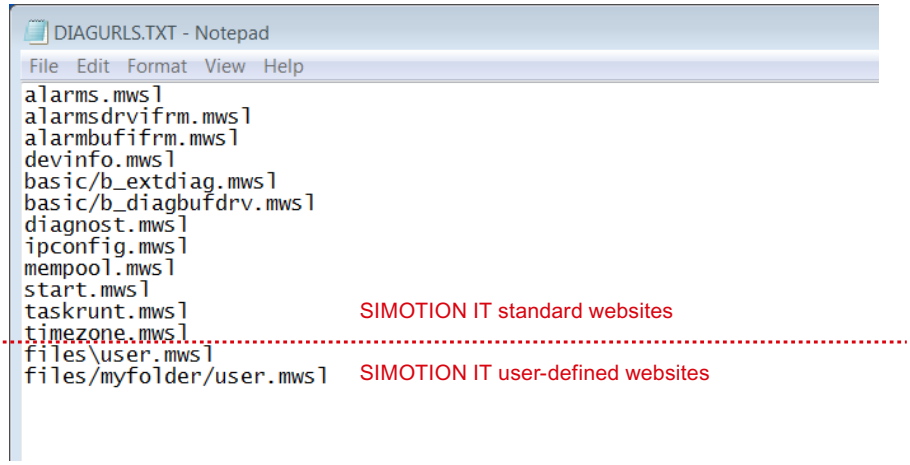


Figure 10-226 Depiction of DIAGURLS.TXT editor

Additional references

You will find detailed information on SIMOTION IT in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Delete/restore non-volatile SIMOTION data

Overview

Requirement

The non-volatile SIMOTION data have been backed up on the CF card by one of the following methods:

- by system function (_savePersistentMemoryData), see also Section Operations and their effect on the user memory (Page 7354)
- manually by service selector switch / Web server / DIAG button, see Section Backup of diagnostic data and non-volatile SIMOTION data (Page 7558).

Procedure

The non-volatile SIMOTION data is restored automatically during a module replacement, see Section Replacing modules in the spare part scenario (Page 7361). The non-volatile data can also be restored manually (by manual operation).

The CF card may contain backups of non-volatile SIMOTION data in various storage locations:

- data backed up with the system function _savePersistentMemoryData
  Storage location on CF card:
  - /USER/SIMOTION/PMEMORY.XML
  - /USER/SIMOTION/PMEMORY.BAK (backup file)
- manually by service selector switch / Web server / DIAG button, backed-up data
  Storage location on CF card:
  - /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML

On manual restoration, the position of the service selector switch defines which of these data will be preferably restored.

During restoration, the non-volatile SIMOTION data is first deleted and then the non-volatile SIMOTION data is restored via the PMEMORY backup file.

If restoration is not possible (e.g. file does not exist or corrupt), the next file in the priority list is accessed.

Table 10-134 Restoration of the non-volatile SIMOTION data

Position of the service selector switch	Use case	Priority sequence for use of the data backups
1	The data backed up with the system function _savePersistentMemoryData is preferably restored	1. /USER/SIMOTION/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.BAK 3. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML
A (as of V4.4)	The data backed up by service selector switch position "D" / Web server / DIAG pushbutton are preferably restored	1. /USER/SIMOTION/HMI/SYSLOG/DIAG/PMEMORY.XML 2. /USER/SIMOTION/PMEMORY.XML 3. /USER/SIMOTION/PMEMORY.BAK

For how to proceed, see Section Restoring data with switch position "1" or "A" (Page 7566).

Note

Firmware / kernel < V4.4

Because switch position "A" is only supported as of V4.4, for < V4.4, restoration must be performed with switch position "1." To force restoration of the data backed up by service selector switch position "D" / Web server / DIAG button, it may be necessary to delete existing files PMEMORY.XML and PMEMORY.BAK in the directory /USER/SIMOTION/ on the CF card.

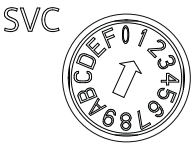
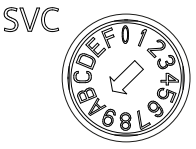
Note

If the data has been restored once via switch setting "A", this advantage is retained, even when the data should be restored later with "1".

Restoring data with switch position "1" or "A"

Procedure

To restore the non-volatile SIMOTION data, proceed as follows:

Step	Switch position "1"	Switch position "A" (as of V4.4)
1.	Insert the CF card into the new SIMOTION D410-2. The SIMOTION D410-2 must be switched off!	
2.	Set the service selector switch to "**1**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service selector switch (position 1)	Set the service selector switch to "**A**." The position of the mode switch is not relevant, i.e. the set operating mode remains unchanged.  Service selector switch (position A)
3.	Switch on the SIMOTION D410-2.	Switch on the SIMOTION D410-2. The flickering green SF LED indicates that restoration is requested via position "A."
4.	Restoration is started automatically.	Turn the service selector switch to "1" to start restoration.
5.	Once restoration has been completed, the module will start up automatically.	
6.	Switch the module off and turn the service selector switch back to "0."	
7.	Switch on the SIMOTION D410-2 again	

Back up diagnostic data and non-volatile SIMOTION data via the web server

SIMOTION devices provide a web server with already prepared standard web pages. These pages can be displayed via Ethernet using a commercially available browser. Additionally, you have the option of creating your own HTML pages and incorporating service and diagnostic information.

Diagnostic data and non-volatile SIMOTION data can be backed up via the web server. The home page of the web server is opened by entering the IP address of the SIMOTION device in the address line of the browser; e.g. http://169.254.11.22

This opens the home page of the web server. To back up diagnostic data and non-volatile data, call the "Diagnostic files" page from the "Diagnostics" menu.

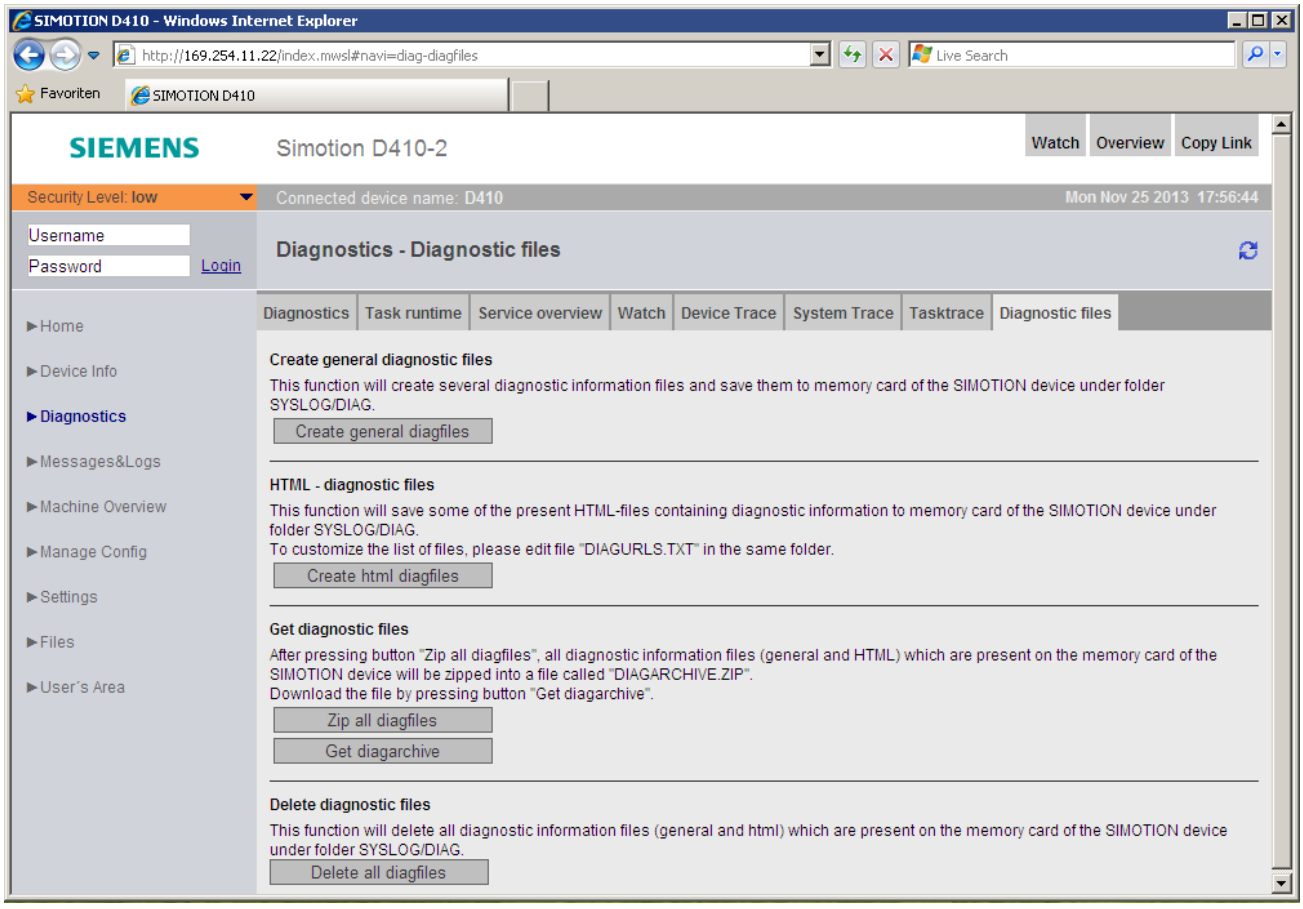


Figure 10-227 SIMOTION IT web server

Table 10-135 Functions on the "Diagnostic files" HTML page

Button	Function
Create general diagfiles	The diagnostic data and non-volatile SIMOTION data are backed up in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory. HTML files used for diagnostics purposes are not saved.
Create html diagfiles	The diagnostic websites are saved on the data medium. Note that only those sites listed in the DIAGURLS.TXT file in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG are saved, see Section Diagnostics via websites (Page 7562).
Zip all diagfiles	The diagnostic files are compressed and stored with their original folder structure in the DIAGARCHIVE.ZIP ZIP file in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory.
Get diagarchive	The ZIP file is saved on a connected PG/PC.
Delete all diagfiles	This button is used to delete all data stored in the ...\USER\SIMOTION\HMI\SYSLOG\DIAG directory; the directory itself is not removed, however.

The diagnostic data and non-volatile SIMOTION data can be found on the CF card in directory: \USER\SIMOTION\HMI\SYSLOG\DIAG

Additional references

You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

Storing diagnostics data and trace on the CF card

As of V4.5, the following data can be saved to the CF card with the **_saveTraceAndDiagnosticFiles** system function:

- Diagnostics data (same data as when the DIAG button is actuated)
- TO trace
- Device trace

The function can be used, for example, to collect diagnostic information (e.g. in the event of a fault) locally on the machine and without backing up SCOUT. The function saves the available data in a ZIP file, whereby the file name and the archive directory can be specified.

The packed data can be accessed, for example, via the web server.

Additional service and diagnostics options

SIMOTION Task Profiler application

The SIMOTION Task Profiler is a dedicated application installed parallel to SIMOTION SCOUT during SIMOTION SCOUT setup. The Task Profiler can be called in online mode using device diagnostics in SIMOTION SCOUT or the Windows application. In the event of a malfunction or error, the Task Trace data can be written to a directory or the CompactFlash card. It can then be processed using the Task Trace Viewer.

Additional references

For detailed information, refer to the *SIMOTION Task Trace* Function Manual.

Diagnostics via the SIMOTION IT web server

SIMOTION D410-2 has an integrated web server. Use is not subject to a license.

In addition to customized websites and the option of performing firmware and project updates, SIMOTION IT web server provides comprehensive device and diagnostic information that can be called using a standard PC with a web browser.

Security concept of HTTP/S, FTP, and Telnet access on the web server

As of version V4.4, access to the SIMOTION IT web server is protected by a multi-level security concept.

The security state of the web server is indicated by the security level on the website. This security level can have three different stages: Low, normal, high.

Security Level Low

The device is supplied with an empty user database. No projects exist yet. The security level is low to allow configuration of the device.

- In this state, access to the web server as an anonymous user is possible to enable use of functions such as the project and firmware update or OPC XML.
- Access to the FTP and Telnet is also possible.
- New users can be entered in the empty user database.

In this state, series commissioning is possible via the web server.

NOTICE
Protecting the device Security level low should only be used for commissioning and service as otherwise the device is not adequately access protected.

Security Level Normal

The controller has a user database. There is a project on the controller; HTTP, HTTPS, FTP, and Telnet have been activated in the HW Config.

User password authentication is mandatory for access to websites with sensitive content (e.g. firmware update watch table, ...), FTP, and Telnet.

Security Level High

High security with maximum access protection:

HTTP, HTTPS, FTP, and Telnet have been deactivated via HW Config. Access to the Ethernet via the various ports of the services is then no longer possible.

State transition from security level low to normal

After taking delivery of the device, the user creates a project and loads it onto the device. This can be done by using the download functions of the SCOUT, by loading it directly onto the memory card (e.g. also via FTP), or via the Manage Config website, Device update tab.

Whichever method is used, the act of loading a project onto the device corresponds to a transition from security level low to security level normal from the point of view of the web server.

Resetting the security level from normal to low

If the user forgets to edit the UserDataBase.xml during initial commissioning, it will no longer be possible to access FTP, web services, or access-protected pages during use.

If there is no mechanical access to the memory card or the device, this can be achieved with the SCOUT function "Delete user data on card". After setting up the user administration, the project must be downloaded again.

Alternatives without SCOUT:

Setting the service selector switch to position "8" restores security level low (SIMOTION IT service mode). Using this method, the device can always be reset to security level low by hardware means.

The following behavior must be taken into account:

- The "Service mode" is activated by turning the SVC switch to position "8".
- If the switch is already set to "8" at ramp-up, it is ignored (protection against forgetfulness).
- The service mode stops immediately when position "8" is exited.
- The service mode stops automatically after 120 minutes.
- It is possible to retrigger 120 minute timeout at any time by turning the switch briefly from "8" to "7" and back, for example.
- The service mode is indicated (as for underlicensing) through slow red flashing of the SF LED.

Note

As an alternative to switch position "8," security level low can also be activated via a simotion.ini file in the main directory of the CF card.

To achieve this, use a text editor (such as Notepad) to create a file called simotion.ini Add the following text: SERVICE_SELECTOR_MODE=8

You must use a text editor and may not use any formatting in the text.

To exit security level low again, undo the changes you have made.

Additional references

You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.

10.1.2.10 Configuring drive-related I/Os (without symbolic assignment)

Overview

The configuration of **local measuring inputs** is fundamentally different to the configuration of **global measuring inputs**.

The assignment of the local measuring inputs is permanently related to the hardware of the control unit and is performed

- On the drive side via the drive expert list and
- During the configuration of the TO measuringInput via the measuring input number.

Local and global measuring inputs have different properties. Detailed information concerning the differences can be found in Section Local and global measuring inputs (Page 7571).

Information about the configuration can be found

- for global measuring inputs (with symbolic assignment) in Section Configuring global measuring inputs (Page 7474)
- for local measuring inputs in the appendix, in Section Configuring local measuring inputs (Page 7573).

Additional references

Further information and programming examples for configuring drive-related I/Os without symbolic assignment can be found

- at Internet address (https://support.industry.siemens.com/cs/ww/en/view/29063656)
- in the SIMOTION Utilities & Applications

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Local and global measuring inputs

Local and global measuring inputs

Depending on the used hardware platform, the following local and global measuring inputs are available for the measuring tasks:

- **Local measuring inputs** are axis-related and implemented in the SINAMICS drive. The measurement records the actual position value.
- **Global measuring inputs** can be freely assigned to the axes and add an internal time stamp to the measurement result for more precise determination of the axis positions.
  The term "central measuring input" is also used within the context of drives.

Table 10-136 Comparison of local and global measuring inputs

	Local measuring input	Global measuring input
Hardware supported	D410, D410-2, D4x5, D4x5-2 (terminal X122/X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2	TM15, TM17 High Feature, D410, D410-2, D4x5, D4x5-2 (terminal X122, X132, X142), CX32, CX32-2, CU310, CU310-2,CU320, CU320-2, ET 200SP/MP timer DIDQ
Measurement procedure	With a signal edge at the relevant input, the current actual values of an encoder connected to a Control Unit are measured with positioning accuracy to determine lengths and distances.	With a signal edge at the relevant input, the current actual values of one or more encoders are measured using time stamp functionality with positioning accuracy in order to provide information for determining lengths and distances (possible with any encoders included in the project).
Configuration of the TO measuringInput in SIMOTION SCOUT	The assignment of inputs is always permanent depending on the hardware of the Control Unit and is performed during the configuration of the TO measuringInput using the measuring input number.	The assignment of inputs is not fixed depending on the hardware and is performed during the configuration of the TO measuringInput by means of symbolic assignment or the hardware address.
TO measuringInput setting: Single measurement [1)]	Yes	Yes

	Local measuring input	Global measuring input
TO measuringInput setting: Cyclic measurement [2]	No	Yes **D410, D410-2, D4x5, D4x5-2 (terminal X122, X132), CX32, CX32-2, CU310, CU310-2, CU320, CU320-2:** The minimum interval between 2 measurements is 3 servo cycle clocks (max. 2 edges per measurement). **D4x5-2 (terminalX142), TM17 High Feature, ET 200SP/MP timer DIDQ:** The minimum interval between 2 measurements is 1 servo cycle (max. 2 edges per measurement). **TM15:** No cyclic measurement available
Use of multiple TO measuringInputs on one axis/encoder. They can be active concurrently	No	Yes
Listening TO measuringInput	No	Yes
Measuring on virtual axes	No	Yes
Measuring on axes attached to a different drive unit	No	Yes

[1] Measurement jobs must be issued individually for each measurement. Several interpolation cycle clocks lie between two measurements.

[2] The measuring is activated just once and runs cyclically until deactivated.

SIMOTION Utilities & Applications includes, for example, a tool with the following functions:

- Estimate of the time between initiating a measurement job to the time at which the measuring input job in the drive acts
- Estimate of the minimum time between two measurement jobs.

SIMOTION Utilities & Applications is part of the scope of delivery of SIMOTION SCOUT.

Table 10-137 Measuring inputs - Overview of quantity structures and functionality

Functionality/device	Max. number of measuring input inputs	As local measuring input configurable	As global measuring input configurable
D410-2, CU310-2	8	x	x
D4x5-2 • X122/X132 • X142	 • 8 • 8	 • 8 • 0	 • 8 • 8
CX32-2	4	x	x
D410, CU310, CX32	3	x	x
D4x5, CU320	6	x	x
TM15	24	-	x
TM17 High Feature	16	-	x

Functionality/device	Max. number of measuring input inputs	As local measuring input configurable	As global measuring input configurable
ET 200SP TM timer DIDQ 10x24V	4	-	x
ET 200MP TM timer DIDQ 16x24V	8	-	x

Configuring local measuring inputs

Properties

Local measuring inputs are always permanently assigned to an axis (drive). They are configured separately for each drive. The drive and the measuring input must always be located on the same control unit. The measurement results are transferred using the axis message frame in accordance with the PROFIdrive profile. Message frame 39x does not need to be configured for local measuring inputs.

The settings for the use of the local measuring inputs must be made in the expert list.

Procedure

To use an I/O terminal on the SIMOTION D410-2 or SINAMICS Control Unit as a measuring input, proceed as follows:

1. Double-click the "Inputs/outputs" entry below the control unit in the project navigator.
2. Configure the desired I/O terminal as input in the "Bidirectional digital I/Os" tab. The configuration can also be set channel-granular on the p0728 parameter using the expert list of the control unit.

The specification of the measuring input terminal must be performed at the local measuring inputs in the expert list of the respective drive (refer to the following table).

Table 10-138 Local measuring inputs, required settings in the expert list of the drive

Parameters in the expert list of the drive	Parameterization as	
	D410-2, CU310-2	D4x5-2, CU320-2
p0488[0] (measuring input 1 input terminal, encoder 1)	DI/DO 8 or DI/DO 9 or DI/DO 10 or DI/DO 11 or DI/DO 13 or DI/DO 14 or DI/DO 15	DI/DO 8 or DI/DO 9 or DI/DO 10 or DI/DO 11 or DI/DO 13 or DI/DO 14 or DI/DO 15
p0488[1] (measuring input 1 input terminal, encoder 2)		
p0488[2] (measuring input 1 input terminal, encoder 3)		
p0489[0] (measuring input 2 input terminal, encoder 1)		
p0489[1] (measuring input 2 input terminal, encoder 2)		
p0489[2] (measuring input 2 input terminal, encoder 3)		

Since a maximum of three encoders can be assigned to a drive, the index [0...2] specifies whether the measurement applies to encoder 1, 2, or 3.

The following must be taken into account:

- Two TO measuring inputs can be configured per TO axis or TO external encoder.
- Only one TO measuringInput can be active on a TO axis or TO externalEncoder.

Table 10-139 Local measuring inputs, configuration of the Measuring Input TO

Axis measuring system no.	Under axis measuring system number, enter the number of the used encoder system (namely, encoder 1, 2 or 3). Encoder system 1 is the default setting.
Drive-related (local measuring input)	Activate the checkbox when a local measuring input is used.
Measuring input number	Enter here which measuring input is used (namely, 1 or 2). Input 1 is the default setting.

Detailed information can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs* Function Manual.

10.1.2.11 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

EMC limits in South Korea

	KC registration number: KCC-REM-S49-SIMOTION For sellers or other users, please keep in mind that this device in an A-grade electromagnetic wave device. This device is intended to be used in areas other than home. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

The EMC limits to be observed for Korea correspond to the limits of the EMC product standard for variable-speed electric drives EN 61800-3 of category C2 or the limit class A, Group 1 according to EN 55011. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. Additional measures, such as the use of an additional RFI suppression filter (EMC filter), may be necessary.

The measures for EMC-compliant design of the system are described in detail in this manual respectively in the Installation Guideline EMC.

Note that the final statement on compliance with the standard is provided by the respective label attached to the individual unit.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert).

Marking for Australia and New Zealand

	SIMOTION D410-2 satisfies the requirement of the standard AS/NZS CISPR 16.
or	
	Marking with RCM (Regulatory Compliance Mark) or C-Tick with older components.

Marking for the Eurasian customs union

	EAC (Eurasian Conformity) Customs union of Russia, Belarus and Kazakhstan Declaration of conformity in accordance with the technical regulations of the customs union (TR CU).

Standards that are not relevant

	China Compulsory Certification SIMOTON D does not belong to the validity area of the China Compulsory Certification (CCC).

China RoHS

SIMOTION D complies with the China RoHS directive. You can find more information on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109738656).

Device-specific information

Note regarding SIMOTION D

Note

The product standard EN 61800-3 describes the EMC requirements placed on "Variable-speed drive systems". As such, it defines different limits depending on the location of the drive system.

SINAMICS S120 power units are designed for use in the second environment. The term second environment refers to all locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

It is essential to follow the installation instructions in the SINAMICS S120 Manuals in order to ensure compliance with emitted interference and immunity values.

The same installation instructions apply for the SIMOTION D410-2 Control Unit as for the SINAMICS S120 CU310-2 Control Unit with regard to EMC.

For further information on this topic also refer to the *SIMOTION PM 21* Catalog as well as the SINAMICS Function Manuals.

10.1.2.12 ESD guidelines

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

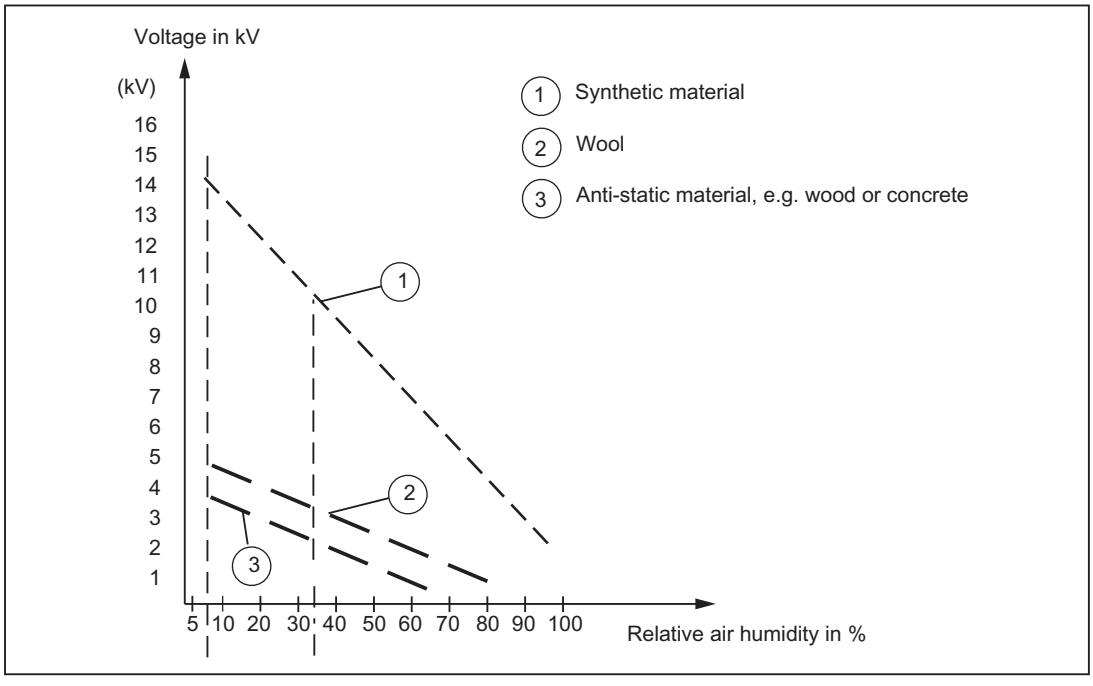


Figure 10-228 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

10.2 Equipment Manual

10.2.1 SIMOTION D4x5-2

Preface

Content of the Manual

This **document** is part of the **SIMOTION D documentation package**.

Scope

The *SIMOTION D4x5-2* manual is valid for the SIMOTION D4x5-2 Control Units, as well as the supplementary CX32-2, CBE30-2 and TB30 system components.

A separate *SIMOTION D4x5* Manual is available for the SIMOTION D425, SIMOTION D435 and SIMOTION D445/D445-1 devices including the CX32, CBE30 and TB30 system components.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Content of the manual

The following is a description of the purpose and use of the product manual:

- Description
  Provides information about the SIMOTION system and its integration in the automation environment.
- Operator control (hardware)
  Provides information about the structure and architecture of the devices.
- Interfaces
  Provides information about the different interfaces of the devices, their pin assignment, and possible applications.
- Technical data
  Provides information about the properties and features of the devices.
- Dimension drawings
- Spare parts / accessories
  Provides information about spare parts and accessories of the SIMOTION D4x5-2, CX32-2 and CBE30-2.
- Appendix
  Provides information about the various standards and specifications fulfilled by the device.
- Index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Disposal and recycling of the device

SIMOTION D is an environmentally friendly product. It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner/index.asp

Further information / FAQs

You can find further information on this manual at the following FAQ:

https://support.industry.siemens.com/cs/ww/de/view/27585482

The following information sources are also available:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling)
- The latest SIMOTION FAQs at https://support.industry.siemens.com/cs/ww/de/ps/14505/faq
- SIMOTION SCOUT online help
- For additional documentation, see the *Overview of SIMOTION documentation* (separate document)

Open source software

Third-party software - License conditions and copyright notes

Copyright notes for the third-party software contained in this product, in particular the open source software, as well as the applicable license conditions, can be found in the READ_OSS.ZIP file on the SIMOTION D CF card or in the corresponding firmware files.

Special note for resellers

The notes and the license conditions contained in the READ_OSS.ZIP file must be passed on to the purchaser in order to avoid the reseller and purchaser from violating the license conditions.

Source code availability

Some license terms of third-party software components used in this product may require us to provide you with the source code and other information for those components. You can find this information directly on or with the product (e.g. on mass storage devices, DVD). If this is not possible for technical reasons, Siemens will be happy to send you this OSS source code in exchange for reimbursement of the processing costs. Please contact the address provided at the end of this section.

Siemens AG

Digital Factory Customer Services

DI CS SD CCC TS

Gleiwitzer Str. 555

D-90475 Nuremberg, Germany

Internet (https://support.industry.siemens.com/cs/ww/en/ps)

Tel.: +49 911 895 7222

10.2.1.1 Safety instructions

Fundamental safety instructions

General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety information for SIMOTION D4x5-2

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra low voltage in accordance with EN/IEC 60950-1 may be connected at all connectors and terminals.

⚠ WARNING

Danger to life from electric shock due to insufficient safety isolation

Employing protection against direct contact using DVC A (PELV) is only permissible in areas with equipotential bonding and in dry rooms indoors.

Use other protective measures against electric shock, such as touch protection, if the specified conditions are not met.

⚠ WARNING

Danger to life from unexpected movement of machines on automatic restart

An automatic restart can be programmed for SIMOTION controllers. When the power returns, the axes start automatically.

Make sure this presents no hazard to personnel or property.

NOTICE
Damage to option boards caused by electric fields or electrostatic discharge
Option boards are ESD-sensitive components.
De-energize the SIMOTION D4x5-2 device before inserting or removing the option board. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Comply with the ESD rules.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Comply with the ESD rules.

NOTICE
Higher operating temperature if ventilation clearances are too small
The 80 mm clearances above and below the components must be observed.
The unit protects itself from overheating by shutting down.
The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

10.2.1.2 Description

System overview

Overview

SIMOTION D is a drive-based version of SIMOTION based on the SINAMICS S120 drive family.

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S120 drive software run on shared control hardware.

SIMOTION D is available in two versions:

- SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications. The Control Unit is snapped directly on to the SINAMICS Power Module in blocksize format and has an integrated drive control for either one servo, one vector or one *V/f* axis.
- SIMOTION D4x5-2 is a Control Unit for multi-axis applications in SINAMICS S120 booksize format. The following performance versions are offered:
  - SIMOTION D425-2 (BASIC performance) Control Unit for up to 16 axes
  - SIMOTION D435-2 (STANDARD performance) Control Unit for up to 32 axes
  - SIMOTION D445-2 (HIGH performance) Control Unit for up to 64 axes
  - SIMOTION D455-2 (ULTRA-HIGH performance) Control Unit for up to 128 axes or applications with very short control cycles

The SIMOTION D4x5 2 is described in this manual. Separate manuals are available for the SIMOTION D410-2 and the D4x5/D410 predecessor modules.

Like SINAMICS S120, SIMOTION D also follows the Totally Integrated Automation (TIA) concept. TIA is characterized by integrated data management, configuration, and communication for all products and systems. Thus, an extensive toolbox of automation modules is also available for SIMOTION D.

Note

In order to cover all versions of SIMOTION D for multi-axis applications, the product will be referred to as "D4x5-2". Specific product designations will be used for information that applies only to one product version, e.g. D445-2 DP/PN.

SIMOTION D4x5-2 DP describes all PROFIBUS versions, and SIMOTION D4x5-2 DP/PN all PROFIBUS/PROFINET versions of the SIMOTION D4x5-2 Control Units.

Application

The SIMOTION D4x5-2 is ideally suited to applications with many coordinated axes with high clock-pulse rates.

Typical applications include:

- Compact multiple-axis machines
- High-performance applications with short machine cycles
- Compact machines
  - Including the complete machine control in the drive
  - With extensive connection possibilities for communication, HMI and I/O
- Distributed drive concepts
  - Applications with many axes
  - Synchronization of several SIMOTION D Control Units using distributed synchronous operation

Versions

The Control Units are available in the versions SIMOTION D425-2 (BASIC performance), SIMOTION D435-2 (STANDARD performance), SIMOTION D445-2 DP/PN (HIGH performance) and SIMOTION D455-2 DP/PN (ULTRA-HIGH performance). The versions differ in their PLC performance and in their motion control performance. The main distinguishing features are:

Table 10-140 Device versions and features

	SIMOTION D425-2	**SIMOTION D435-2**	**SIMOTION D445-2**	**SIMOTION D455-2**
Maximum number of axes	16	32	64	128
Minimum servo/interpolator cycle clock	0.5 ms	D435-2 DP: 0.5 ms D435-2 DP/PN: 0.25 ms	0.25 ms	0.25 ms 0.125 ms [1]
DRIVE-CLiQ interfaces	4	6	6	6

[1] Only with ET 200SP, SCOUT TIA and Servo_fast / IPO_fast

The Control Units feature PLC and motion control performance (open-loop control and motion control) for up to 16, 32, 64 or 128 axes, as required.

The integrated drive computing performance of the Control Units allows up to 6 servo, 6 vector or 12 *V/f* axes on each D4x5-2 Control Unit (drive control based on CU320-2, firmware version ≥ V4.x).

The drive control supports servo control (for a highly dynamic response), vector control (for maximum torque accuracy) and *V/f* control.

SIMOTION D435-2 DP/PN and D455-2 DP/PN are also available as SIPLUS version for use under extremely harsh environmental conditions, e.g. in toxic atmospheres (for details refer to technical specifications). As BasedOn products, the SIPLUS versions have the same functionality as the standard modules and are configured in the same way.

The SIMOTION D4x5-2 Control Units and their CX32-2, CBE30-2 and TB30 supplementary system components are described in the following.

Note

With the SIZER configuration tool, you can easily configure the SINAMICS S110/120 drive family including SIMOTION.

It provides you with support for selecting and dimensioning the components required for a Motion Control task.

You can also determine the possible number of axes and the resulting load with SIZER in accordance with your performance requirements.

Hardware components

As the central hardware, SIMOTION D uses the SIMOTION D4x5-2 as Control Unit consisting of the SIMOTION runtime system and the SINAMICS drive control. The Control Unit uses the SINAMICS Integrated drive with various SINAMICS S120 drive modules (Line and Motor Modules) to perform open-loop and closed-loop control of the axis grouping. A range of additional SINAMICS S120 components, such as SMx encoder systems or Terminal Modules can also be connected via DRIVE-CLiQ. With a few exceptions (e.g. no basic positioner EPOS, no Basic Operator Panel BOP20, etc.), the drive control integrated in SIMOTION D has the same control characteristics and performance features as the SINAMICS S120 CU320-2 Control Unit. The EPOS functionality is provided by the SIMOTION technology functions. The functionality of SIMOTION D can be expanded with distributed I/O via PROFIBUS or PROFINET IO.
The following figure shows a typical SIMOTION D axis grouping.

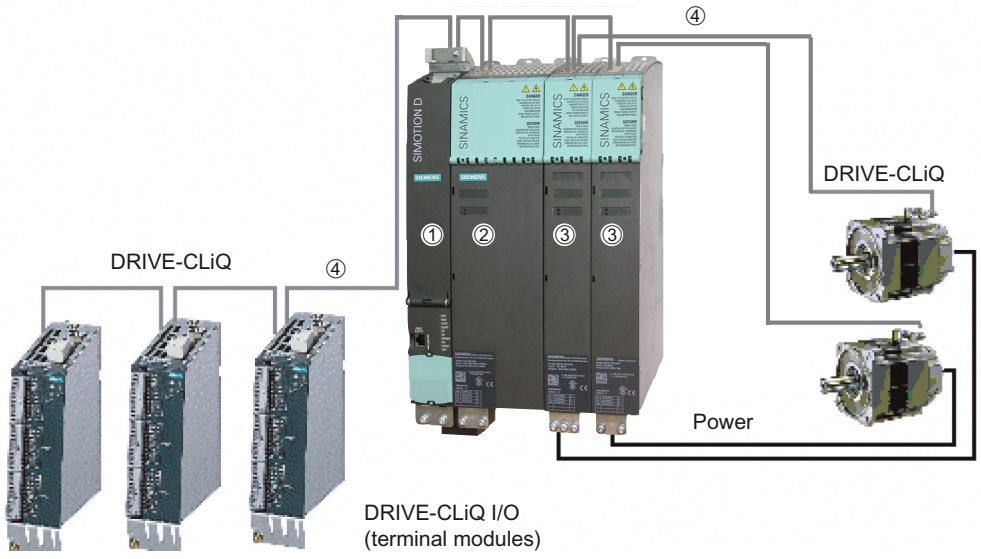


Figure 10-229 Example of an axis grouping with SIMOTION D4x5-2

A SIMOTION D axis grouping generally consists of the following elements:

- **SIMOTION D** (Control Unit) (1)
  This unit contains the programmable runtime system of SIMOTION and the drive software of SINAMICS S120. In principle, SIMOTION D is capable of controlling multiple axes/drives.
- One **SINAMICS infeed** (Line Module) (2)
  This module generates a DC link from the supply system.
- **SINAMICS power units** (Motor Modules) (3)
  These modules are used to control motors.
  It is also possible to operate SINAMICS Power Modules in blocksize format with the SINAMICS Control Unit Adapter (CUA). A separate infeed is then unnecessary.
- **DRIVE-CLiQ components** (4)
  In SINAMICS S120 / SIMOTION D, the individual components of the drive system communicate with each other via DRIVE-CLiQ. In addition to power components, it is also possible to link encoder systems and special DRIVE-CLiQ I/O devices via DRIVE-CLiQ.

Extension of the drive computing performance

The motion control performance of a SIMOTION D4x5-2 can be utilized in full by expanding the computing performance at the drive in two different ways:

- SINAMICS S/G Control Units (e.g. CU320-2, CU310-2, CU305, CU250S-2, etc.) together with further drive components can be connected via PROFIBUS or PROFINET.
- With SIMOTION D4x5-2, the CX32-2 Controller Extension can be connected via DRIVE-CLiQ. This module is extremely compact, does not require a separate CompactFlash card and can control up to 6 servo, 6 vector or 12 *V/f* axes.

Software components

The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following:

The SIMOTION runtime system with the following functions:

- Freely programmable runtime system (IEC 61131)
- Various runtime levels (tasks)
- PLC and arithmetic functionality
- Motion control functions
- Communication functions

The SINAMICS S120 drive control with the following functions:

- Closed-loop current and torque control
- Closed-loop speed control
- Closed-loop infeed

System components

Central components

SIMOTION D4x5-2 communicates with automation components via the following interfaces:

- PROFIBUS DP
- Ethernet
- PROFINET IO
- DRIVE-CLiQ (DRIVE Component Link with IQ)

SIMOTION D features a SINAMICS Integrated drive element. The communication with the SINAMICS Integrated is performed via PROFIBUS mechanisms (DP Integrated), i.e. the communication is handled, for example, via PROFIdrive telegrams.

Shorter cycle times and greater numbers of addresses for each node are achieved with the "DP Integrated" compared to the "external PROFIBUS DP".

The most important components of the system and their functions are shown below.

Table 10-141 Central components

Component	Function
SIMOTION D4x5-2 controller	... is the central motion control module. This module contains the programmable SIMOTION runtime for the SIMOTION D4x5-2 and the SINAMICS S120 drive software. You can use the integrated high-speed digital I/Os as: • User-addressable process I/Os • Homing inputs • Inputs for measuring inputs • Outputs for fast output cams The measuring sockets can output any analog signals.
System software	The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following: • SIMOTION runtime (kernel) • Drive software of SINAMICS S120 - implements all drive functions
Power supply	... provides the electronics power supply for SIMOTION D, e.g. via the SITOP power supply.

PROFIBUS DP

The Control Unit can communicate with the following components via the PROFIBUS DP interfaces:

Table 10-142 Components on PROFIBUS DP

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs, and tests with the "SIMOTION SCOUT" engineering system (ES)
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.

Component	Function
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET connection and live module replacement.
SIMATIC ET 200eco	I/O system with IP65/67 degree of protection for machine-related applications with no cabinet, and with flexible and fast ECOFAST or M12 connection methods.
Other PROFIBUS I/O	
Gateways	• DP/AS-Interface link 20E and DP/AS-Interface link Advanced for the PROFIBUS DP gateway to AS-Interface • DP/DP coupler for connecting two PROFIBUS DP networks
Drive interfaces	• ADI4 (Analog Drive Interface for 4 axes) for connection of drives with analog ± 10 V setpoint interface or for external encoders • IM174 (Interface Module for 4 axes) for connection of drives with analog ± 10 V setpoint interface, for external sensors, or for connection of stepper drives with pulse-direction interface
Drive units with PROFIBUS DP interface (e.g. SINAMICS S120)	... convert speed setpoints into signals for controlling the motors and supply the power required to operate the motors. Also can be operated as an isochronous slave on PROFIBUS DP.
Teleservice adapter	Remote diagnostics

Ethernet

The Control Unit can communicate with the following components via the Ethernet interfaces or be embedded in an automation environment:

Table 10-143 Components on the Ethernet

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs, and tests with the "SIMOTION SCOUT" engineering system (ES)
Master computer	... communicates with other devices via UDP, TCP/IP
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.

PROFINET IO

The D4x5-2 DP/PN can communicate with the following components via the onboard PROFINET IO interface or via the Ethernet communication board (CBE30-2).

Table 10-144 Components on the PROFINET IO

Component	Function
Programming device (PG/PC)	... configures, parameterizes, programs and tests with the "SIMOTION SCOUT" engineering system (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Master computer	... communicates with other devices via UDP, TCP/IP
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data. With the time-based I/Os, signals can be recorded or output to the precise µs.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety. With the time-based I/Os, signals can be recorded or output to the precise µs.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET IO connection and live module replacement.
SIMATIC ET 200eco PN	Compact block I/O with IP65/66/67 degree of protection for machine-related applications with no cabinet, and with M12 connection method. Very rugged and resistant encapsulated metal enclosure.
Other PROFINET IO I/O devices	
Drive units with PROFINET IO interface	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors.
Gateways	• IE/AS-Interface link PN IO for the PROFINET IO gateway to AS-Interface • PN/PN coupler for connecting two PROFINET IO networks

DRIVE-CLiQ

The DRIVE-CLiQ interfaces permit a fast connection to the SINAMICS drive components.

DRIVE-CLiQ offers the following advantages within the DRIVE-CLiQ topology rules:

- Expandability of components
- Automatic detection of components by the Control Unit
- Standardized interfaces to all components
- Uniform diagnostics down to the components
- Uniform service through to the components
- Simple mechanical handling

The controller can communicate with the following components via DRIVE-CLiQ:

Table 10-145 Components connected to DRIVE-CLiQ

Component	Function
Control Unit (SINAMICS S110/ S120)	Central control module in which the open- and closed-loop control functions for the drive are implemented.
Line Module (SINAMICS S120)	... generates a DC link from the supply system.
Motor Module (SINAMICS S120)	... controls motors (DC/AC inverters, booksize).
Power Module (SINAMICS S110/ S120)	... controls motors (AC/AC converters, blocksize).
CX32-2 Controller Extension	... enables additional axes to be connected for SIMOTION D4x5-2.
CUA31/CUA32 Control Unit Adapter	... enables a Power Module in blocksize format to be connected to a booksize D4x5-2, CX32-2 or CU320-2 Control Unit.
TM15, TM17 High Feature Terminal Modules	The TM15 and TM17 High Feature Terminal Modules are used to implement inputs of measuring inputs and outputs of output cams. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times.
TM31 Terminal Module	... enables a terminal expansion via DRIVE-CLiQ (additional analog and digital I/Os).
TM41 Terminal Module	... enables a terminal expansion (analog and digital I/Os) and encoder emulation.
TM54F Terminal Module	... enables terminal expansion (fail-safe digital I/Os) for controlling the safe motion monitoring functions of the integrated drives.
TM120 Terminal Module	Four temperature sensors (KTY84-130 or PTC) can be evaluated via the TM120 Terminal Module. The temperature sensor inputs are safely electrically separated from the evaluation electronics in the TM120 Temperature Module and are suitable for evaluating the temperature of special motors, e.g. 1FN linear motors and 1FW6 built-in torque motors.
TM150 Terminal Module	The TM150 Terminal Module can be used to evaluate temperature sensors (KTY, PT100, PT1000, PTC, and bimetal normally closed contact). This means, for example, that other temperatures from the process can be measured in addition to the motor temperature. Temperature sensors can be evaluated using a 2, 3 or 4-wire system. Twelve temperature sensors can be evaluated with 2-wire evaluation and six temperature sensors with 3 and 4-wire evaluation.

Component	Function
SMx Sensor Modules	... enable acquisition of encoder data from connected motors via DRIVE-CLiQ.
Motors with DRIVE-CLiQ interface	... enable simplified commissioning and diagnostics, as the motor and encoder type are identified automatically.
DMC20/DME20 DRIVE-CLiQ hub	... enables the number of DRIVE-CLiQ interfaces to be increased and the creation of a point-to-point topology.

Power Modules via CUA31/32

The following Power Modules are supported:

- PM340
- PM240-2 (as of SIMOTION V4.4 / SINAMICS V4.7)

The mixed operation of a PM240-2 with booksize modules and/or PM340 blocksize modules on a CU320-2/D4x5-2/CX32-2 is possible as of SINAMICS V4.7 HF12 or SIMOTION V4.4 HF6 (SINAMICS Integrated V4.7 HF12).

Note

You will find detailed information on components of the SINAMICS S110/S120 product family in the SINAMICS S110/S120 manuals.

It is possible that older DRIVE-CLiQ components can no longer be used with SIMOTION D4x5-2/CX32-2. You will find detailed information on this topic in the SIMOTION D4x5-2 Commissioning and Hardware Installation Manual at "Migration of D4x5 to D4x5-2" in Section "Permissible combinations".

Optional components

The functionality of the D4x5-2 Control Unit can be expanded with the following components:

Table 10-146 Optional components

Component	Function	D4x5-2 DP	D4x5-2 DP/PN
CBE30-2 Ethernet communication board	Communication via PROFINET IO with IRT and PROFINET IO with RT	No	Yes
TB30 Terminal Board	Terminal expansion, i.e. additional analog and digital I/Os	Yes	Yes

The components are plugged into the option slot of the Control Unit.

I/O integration

Note

Note that not all modules in the ET 200 I/O family are approved for SIMOTION. Moreover, system-related functional differences can come into play when these I/Os or I/O systems are used on SIMOTION vs. on SIMATIC. For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

A detailed, regularly updated list of the I/O modules approved for use with SIMOTION, as well as notes on their use, can be found on the Internet at: (https://support.industry.siemens.com/cs/ww/en/view/11886029)

In addition to the I/O modules enabled for SIMOTION, in principle all certified standard PROFIBUS slaves (DP-V0/DP-V1/DP-V2) and PROFINET IO devices with RT and IRT real-time classes may be connected to SIMOTION D4x5-2. These modules are integrated using the GSD file (PROFIBUS) or GSDML file (PROFINET) provided by the relevant device manufacturer.

Note

Please note that in individual cases further boundary conditions must be fulfilled in order to integrate a standard slave/standard device into SIMOTION. Thus, a few modules require "driver blocks" , e.g. in the form of function blocks, that permit (or simplify) integration.

For modules released with SIMOTION (e.g. SIMATIC S7-300 module FM 350-1, etc.), these driver blocks are part of the SIMOTION SCOUT engineering system command library.

Representation of SIMOTION D425-2 DP and D435-2 DP

The following figure shows the SIMOTION D425-2 DP and D435-2 DP with their interfaces and front panel elements (fault and status displays).

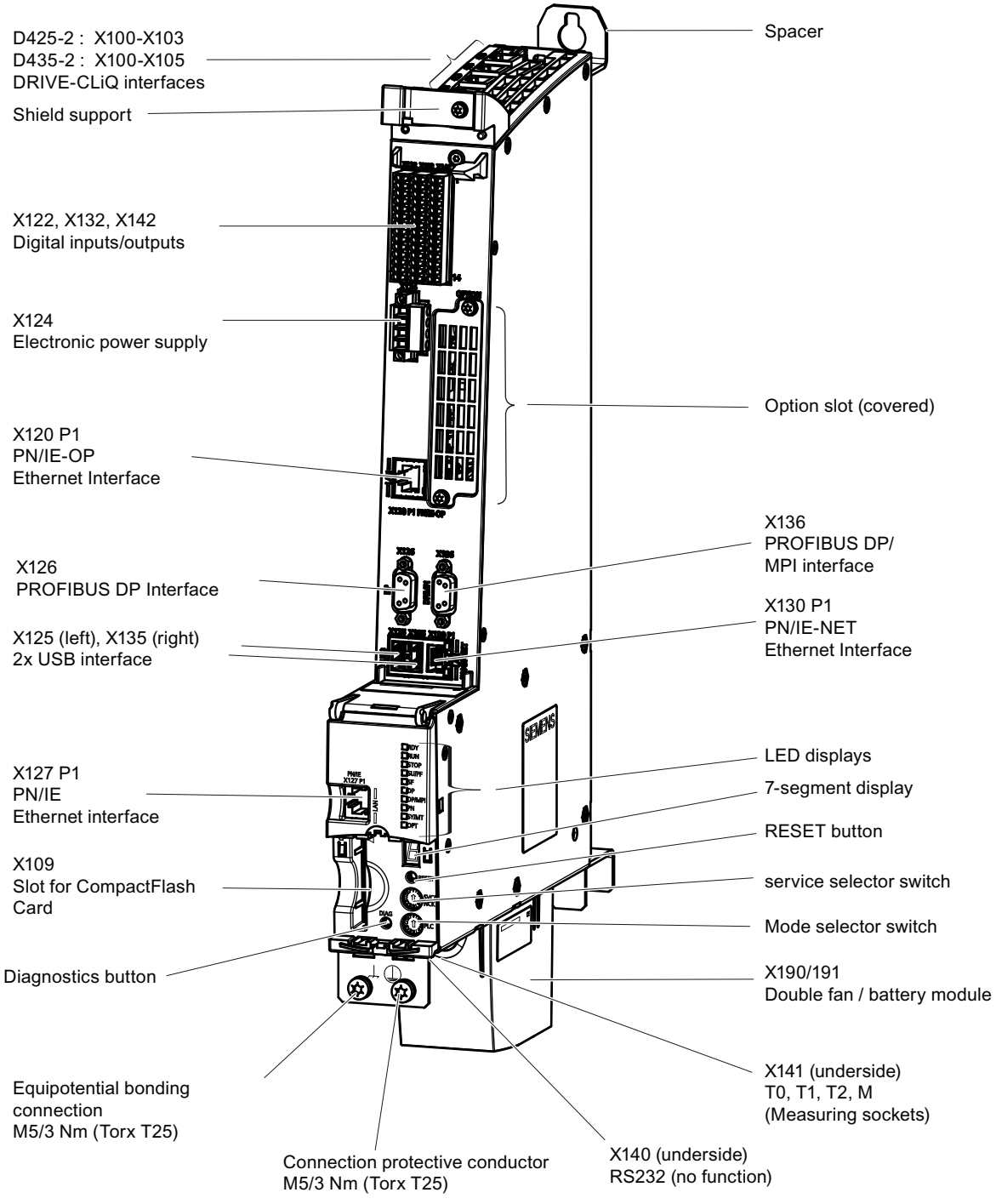


Figure 10-230 D425-2 DP and D435-2 DP representation

Note

SIMOTION D425-2 DP and D435-2 DP must be operated with a double fan/battery module for heat dissipation. Without this module, the Control Units will not start up and cannot be commissioned.

Information on how to install the double fan/battery module can be found in "Supplementary system components", in Section Installing the fan/battery module (Page 7667).

Note

SIMOTION D425-2 DP and D435-2 DP are supplied with pre-assembled spacers. These can be removed if necessary.

For further details, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Representation of SIMOTION D425-2 DP/PN and D435-2 DP/PN

The following figure shows the SIMOTION D425-2 DP/PN and D435-2 DP/PN with their interfaces and front panel elements (fault and status displays).

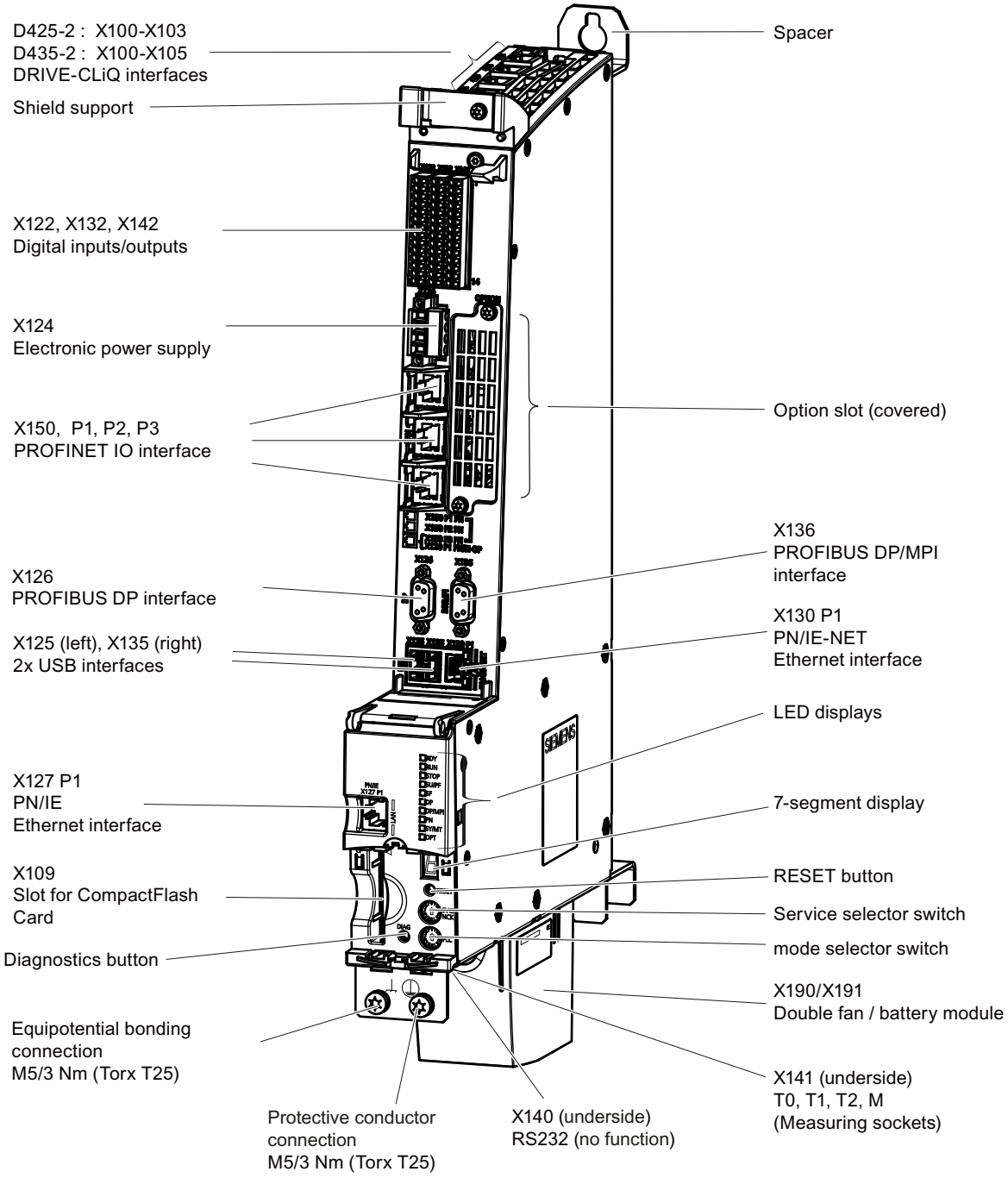


Figure 10-231 D425-2 DP/PN and D435-2 DP/PN representation

Note

SIMOTION D425-2 DP/PN and D435-2 DP/PN must be operated with a double fan/battery module for heat dissipation. Without this module, the Control Units will not start up and cannot be commissioned.

Information on how to install the double fan/battery module can be found in "Supplementary system components", in Section Installing the fan/battery module (Page 7667).

Note

The SIMOTION D425-2 DP/PN and D435-2 DP/PN are supplied with pre-assembled spacers. These can be removed if necessary.

For further details, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Representation of SIMOTION D445-2 DP/PN and D455-2 DP/PN

The following figure shows the SIMOTION D445-2 and D455-2 with its interfaces and front panel elements (fault and status displays).

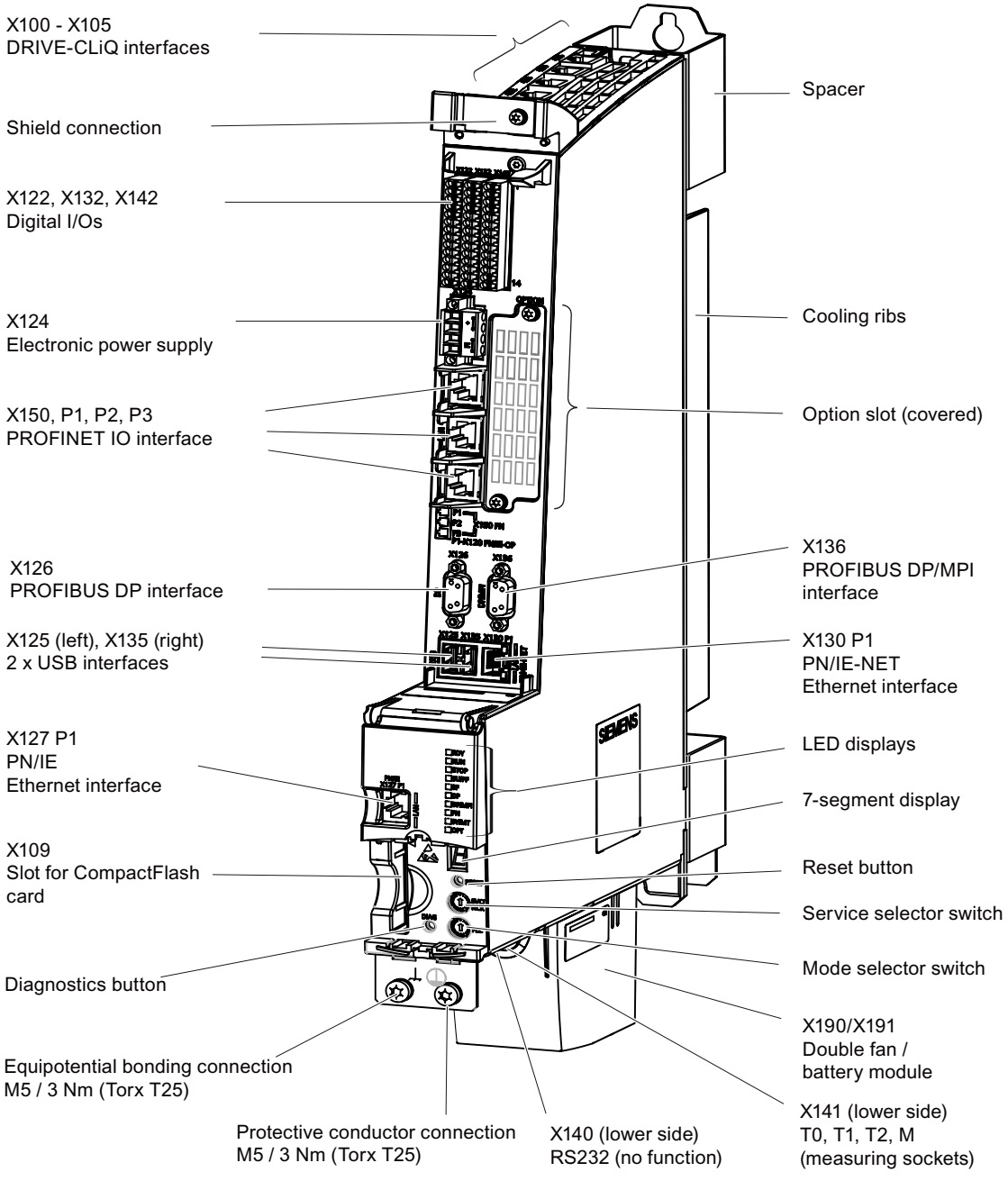


Figure 10-232 View of D445-2 DP/PN and D455-2 DP/PN

Note

SIMOTION D445-2 DP/PN and D455-2 DP/PN must be operated with a double fan / battery module for heat dissipation. Without this module, the Control Units will not start up and cannot be commissioned.

Information on how to install the double fan/battery module can be found in "Supplementary system components", in Section Installing the fan/battery module (Page 7667).

Note

With the D445-2 DP/PN and D455-2 DP/PN, the spacers can only be removed with the "external air cooling" installation method. In this installation method, the cooling fins are inserted through a cutout in the rear cabinet panel.

For further details, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Type plates

Side-mounted rating plate

The following figure shows the information contained on the rating plate mounted on the side of the housing.

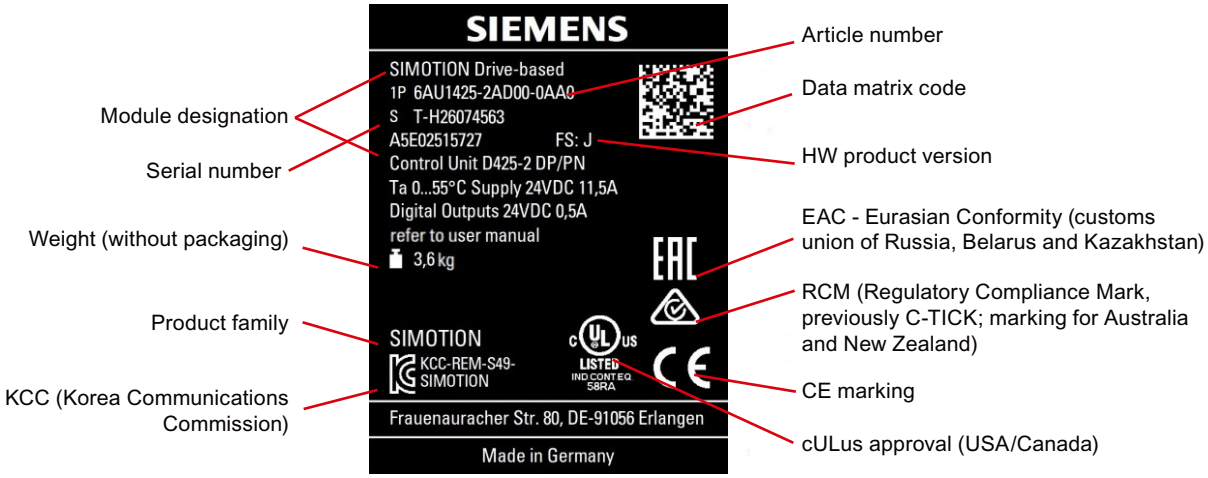


Figure 10-233 Rating plate using the D425-2 DP/PN as an example

Note

The information contained in each field of the rating plate on your actual Control Unit may differ from that presented in this manual (for example, a later product version, approvals and marks that have not yet been earned, etc. may be shown).

Depending on the rating plate, the HW version may be designated as "Version" or "FS" (Function State). Older components are marked with C-Tick instead of RCM. Depending on the module, the serial number is only on the front rating plate, which is also visible when the module is installed.

Front rating plate

A second rating plate for the MAC addresses of the Ethernet interfaces and the PROFINET IO interface is attached to the front of the device. You see this rating plate when you open the front cover of the Control Unit.

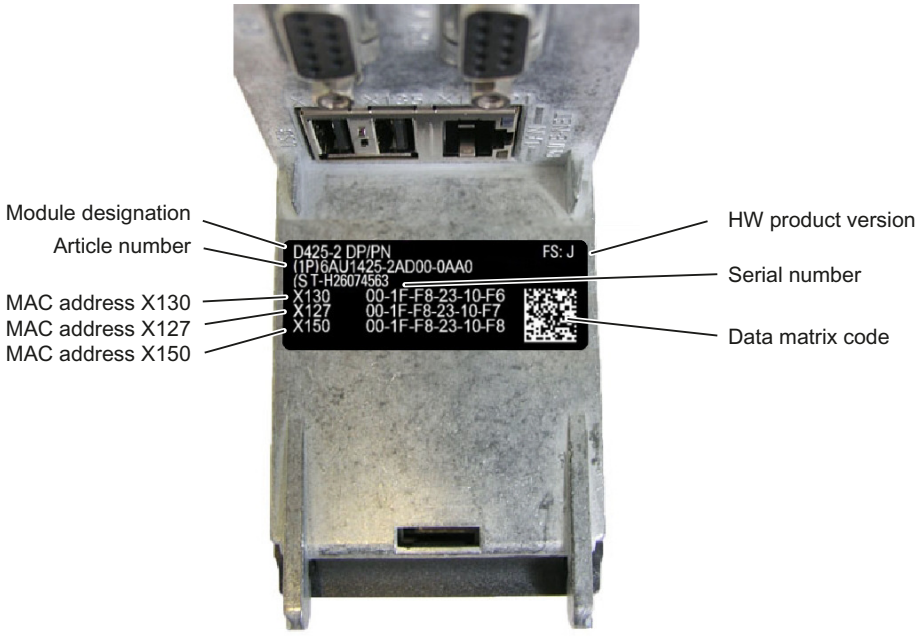


Figure 10-234 MAC addresses of SIMOTION D425-2 DP/PN

Note

For SIMOTION D4x5-2 DP, the MAC address for the X120 interface is printed instead of the MAC address for the X150 interface.

Depending on the rating plate, the HW version may be designated as "Version" or "FS" (Function State).

Industry Online Support app

With our app, you have access to more than 300,000 documents.

Scan the data matrix code and display all the technical information on this product incl. graphic data (CAx data). Link to the app: https://support.industry.siemens.com/sc/en/en/sc/2067

CompactFlash card

Usage and function of the CompactFlash Card

The CompactFlash card (CF card) is inserted in the slot with the designation CF (X109 interface).

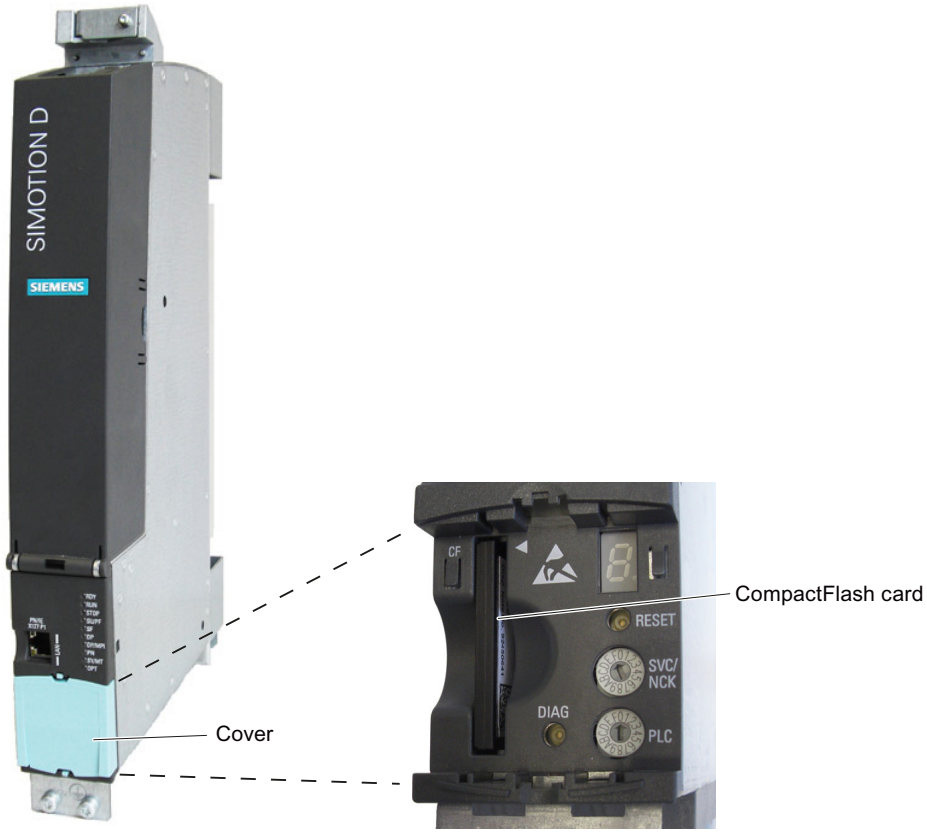


Figure 10-235 Slot for CompactFlash card

The CF card does not extend beyond the housing. An ergonomic recessed grip enables the CF card to be removed.

Characteristics of the CF card

The CF card is essential for operation of the SIMOTION D4x5-2. The CF card is not supplied with the SIMOTION D4x5-2 and must be ordered separately.

The SIMOTION Kernel (SIMOTION D4x5-2 firmware) and the software used to control the drives (SINAMICS firmware) are contained on the CF card.

The CF card is used for

- Backing up the technology packages and user data (programs, configuration data, parameter assignments).
- Update (e.g. SIMOTION firmware update).

The licenses for the technology functions are linked to the serial number of the CF card. This means the CF card can be inserted in a different SIMOTION D without having to change the licenses.

The CF card is supplied in a bootable format with the latest SIMOTION Kernel and drive software. Please note that a CF card with a D4x5 kernel/drive software cannot run on a D4x5-2. The same applies for the reverse situation.

If an error occurs, all LEDs flash yellow with 2 Hz. An entry is also made in the diagnostic buffer and the D4x5-2 does not start.

Licenses purchased for SIMOTION D can be used for both the D4x5 and D4x5-2.

NOTICE
Damage to the CompactFlash card from electrical fields or electrostatic discharge The CompactFlash card is an ESD-sensitive component. De-energize the SIMOTION D4x5-2 device before inserting or removing the CompactFlash card. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off. Comply with the ESD rules.

Additional information

For additional information about inserting, changing, writing and formatting the CF card, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

CompactFlash card

CF card

CF cards with different storage capacities are available for SIMOTION D4x5-2.

- 2 GB CF, article number 6AU1400-2QA20-0AA0
- 1 GB CF, article number 6AU1400-2PA23-0AA0
- 1 GB CF, article number 6AU1400-2PA22-0AA0
- 1 GB CF, article number 6AU1400-2PA21-0AA0

You will find detailed information on the compatibility relationships for the CF card, boot loader version, SIMOTION D hardware and SIMOTION firmware version in the software compatibility list.

This list can be found on the Internet at (https://support.industry.siemens.com/cs/ww/en/view/18857317).

Rating plate information

The following figure shows you all the information contained on the rating plate of the CF card.

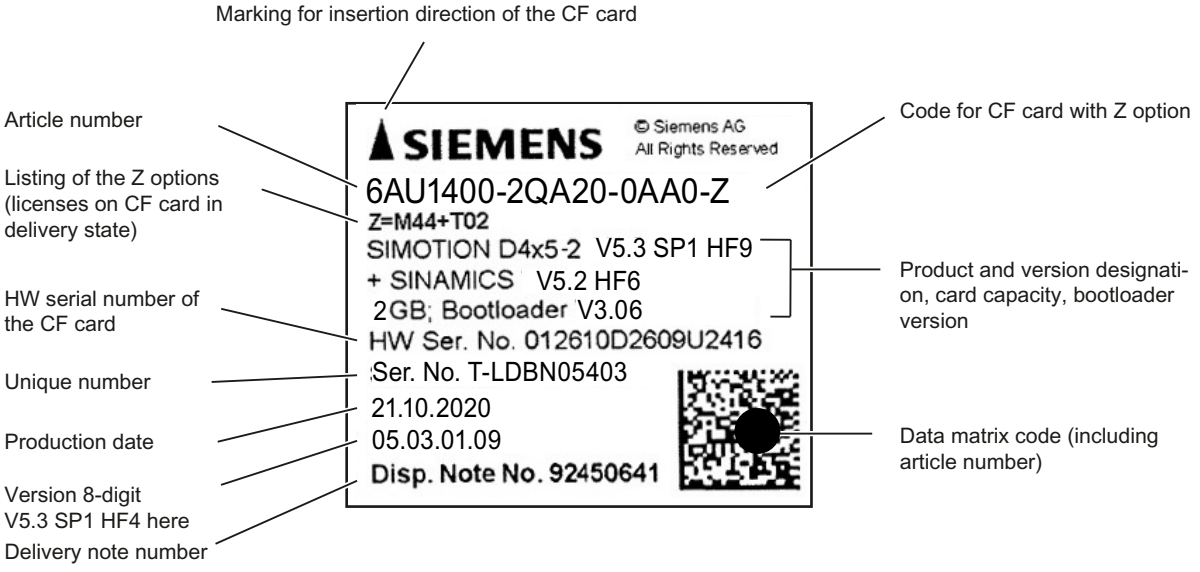


Figure 10-236 Rating plate of the CF card (example of 2 GB CF card)

Pre-installed runtime licenses

The preinstalled licenses are printed on the label as Z option below the article number.

Example

Example with MultiAxes package Z option for D445/D445-1/D445-2/D455-2 + two TControl licenses:

- 6AU1400-2QA20-0AA0-Z

- Z=M44+T02

A maximum of seven different Z options are printed on the label of the CF card. When there are more than seven different Z options, the text "Z = see delivery order" is printed on the CF card in place of the Z options.

Available Z options / licenses for CF cards

- Axis licenses
  - Pxx POS license and number (e.g. P02 = 2x POS licenses)
  - Gxx GEAR license and number (e.g. G03 = 3x GEAR licenses)
  - Cxx CAM license and number (e.g. C01 = 1x CAM license)
- MultiAxes package
  - M00 MultiAxes package license (platform independent)
  - M42 MultiAxes package license for D425/D425-2
  - M43 MultiAxes package license for D435/D435-2 (incl. D425/D425-2)
  - M44 MultiAxes package license for D445/D445-1/D445-2/D455-2 (incl. D425/D425-2 and D435/D435-2).
- MultiAxes and Safety Extended package
  - S42/S43/S44 license as M42/M43/M44 license, but also including licensing of the Safety Integrated Extended Functions for all drives on SINAMICS Integrated/CX32-2 Licenses for Safety Integrated Advanced Functions are not included; they must always be ordered as single licenses.
- TControl temperature control
  - Txx TControl license and number (e.g. T03 = 3x TControl licenses)
- SIMOTION IT
  - SIMOTION IT Virtual Machine J00 license for Java applications
- Safety functions
  - Fxx license for SINAMICS Safety Integrated Extended Functions (for integrated SINAMICS drives and CX32-2 for SIMOTION D) (e.g. F02 = two Safety Integrated Extended Functions)
  - Lxx license for SINAMICS Safety Integrated Advanced Functions (for integrated SINAMICS drives and CX32-2 for SIMOTION D) (e.g. L03 = three Safety Integrated Advanced Functions) The Safety Integrated Advanced Functions also include the Safety Integrated Extended Functions.
- High output frequency
  - H00 license SINAMICS high output frequency for SIMOTION D
- Other SIMOTION licenses
  - B02 Multipurpose Information Interface (MIIF) communication function license
  - B03 license Vibration damping of axes (VIBX Vibration Extinction)
  - B04 license Motion profiles for servo presses (OACAMGEN cam generation)
  - B07 license Compensation of cyclic errors from the production process (LECo - Learning Error Compensation)

SINAMICS licenses

Selected SINAMICS licenses can be used with a SIMOTION D CF card. Only one relicensing is possible. A prelicensing of SIMOTION D CF cards via Z options is not possible with SINAMICS licenses.

Examples:

- SINAMICS S120 Advanced Position Control (APC)
  Article no. 6SL3074-0AA05-0AA0
  License for each drive (on CU, SINAMICS Integrated, CX32-2)
- SINAMICS S120 cogging torque compensation
  Article no. 6SL3074-0AA15-0AA0
  License for each drive (on CU, SINAMICS Integrated, CX32-2)
- SINAMICS Technology Extension "Vibration Extinction" (VIBX)
  Article no. 6SL3077-0AA00-5AB0
  License for a target device (CU, SINAMICS Integrated, CX32-2)
- SINAMICS DCB Extension
  Article no. 6SL3077-0AA00-0AB0
  License for a target device (CU, SINAMICS Integrated, CX32-2)

With SINAMICS licenses, underlicensing of SINAMICS Integrated/CX32-2 is indicated by the flashing SF LED on the SIMOTION D Control Unit. An entry is also made in the diagnostic buffer and the underlicensing is displayed in the License dialog box of SIMOTION SCOUT. The licensing is performed (as for SIMOTION licenses) via SIMOTION SCOUT or via the SIMOTION license key on the CF card.

Data matrix code

SIMOTION D components (e.g. CF cards, Control Units, etc.) have a machine-readable identification in the form of a data matrix code (2D code).

Reader units that support the data matrix code in accordance with ECC 200 are suitable for reading the code used here.

The volume of the information contained in the data matrix code depends on the product and, for example, on the available space.

Analysis

Example of a data string from the reader unit:

1P6AU1400-2QA20-0AA0-Z+ST-LDBN05403+30S012610D2609U2416.

Table 10-147 Machine-readable identification via 2D code

Characteristic	Property (example)
Article number ("1P" identifier to identify the products)	6AU1400-2QA20-0AA0-Z
Serial number ("S" identifier, item number)	T-LDBN05403

Characteristic	Property (example)
Hardware serial number (CF cards only) ("30S" identifier)	012610D2609U2416
Hardware version (identifier 2PE)	Not used in the example
Material number (identifier P)	Not used in the example

In addition to the "serial number", CF cards also have a "hardware serial number".

If licenses are purchased for functions under license, a "license key" is generated from the hardware serial number of the CF card and the serial number of the purchased licenses, which is only valid for the respective CF card.

The data required for the licensing can be read by a reader unit via the bar codes on the license certificates (Certificate of License "CoL") and the 2D code on the CF card in order, for example, to automate the licensing process.

Industry Online Support app

With our app, you have access to more than 300,000 documents.

Scan the data matrix code and display all the technical information on this product incl. graphic data (CAx data). Link to the app: https://support.industry.siemens.com/sc/en/en/sc/2067

10.2.1.3 Operator control (hardware)

Overview of operator control and display elements

The following figure shows the arrangement of the operator control and display elements of a SIMOTION D445-2 DP/PN.

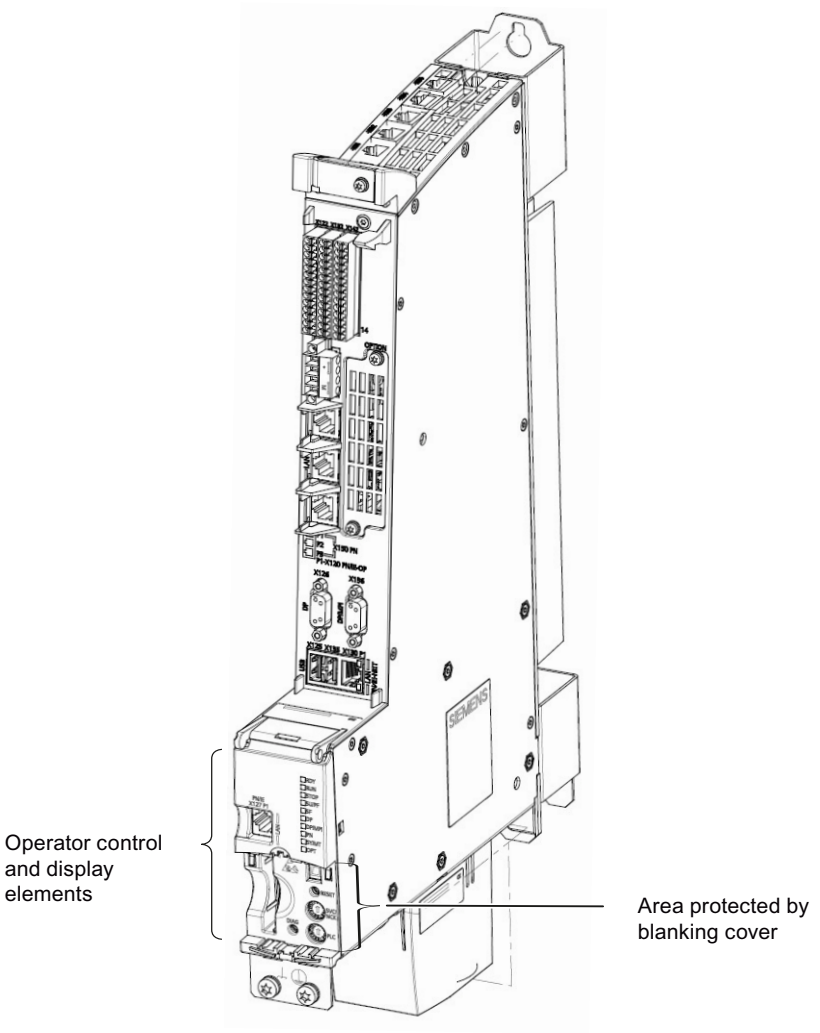


Figure 10-237 Position of operator control and display elements (example of SIMOTION D445-2 DP/PN)

The lower part of the operator control and display elements has a blanking cover during operation. This cover is removed for service work.

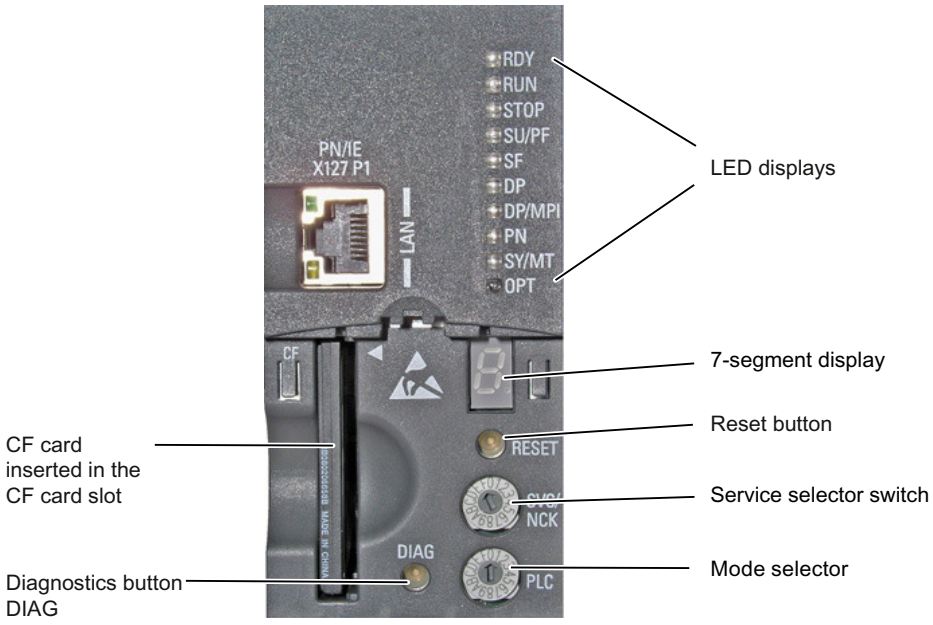


Figure 10-238 Control and display elements of the SIMOTION D4x5-2

Operator controls

Service and operating mode switch

Characteristics of the Service switch and mode switch

SIMOTION D4x5-2 has two selector switches on the lower front side for selection of the service functions and operating modes.

The upper selector switch (labelled SVC/NCK) is for the selection of service and diagnostic functions. In "normal" operation this switch must remain in the 0 position (see figure below).

The lower switch, labelled PLC, is used to set one or more operating modes of the SIMOTION D4x5-2.

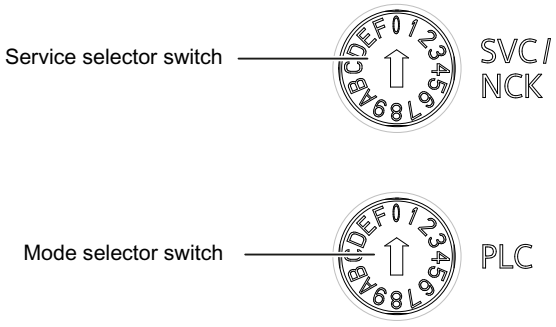


Figure 10-239 Selector switches for service and operating modes of the SIMOTION D4x5-2

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Comply with the ESD rules.

Mode switch

The following table contains the possible mode switch positions and the associated LED displays. The mode switch positions are explained in the order in which they are arranged on the SIMOTION D4x5-2.

Table 10-148 Mode switch position

Switch position	Meaning	LED
0	RUN	RUN
1	STOPU	SU/PF
2	STOP	STOP
3	MRES	The MRES operating states are indicated on the STOP LED (on/off/ flashing, see *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual)
Other selector positions are not assigned		

The following table contains the states of the SIMOTION D4x5-2 that can be set via the mode switch.

Table 10-149 Mode switch settings

Meaning	Explanations
RUN	SIMOTION D4x5-2 processes the user program and the associated system services: • Reading process image of inputs • Execution of the user programs assigned to the execution system. • Writing process image of outputs The technology packages are active in this state. They can execute commands from the user program.
STOPU	SIMOTION D4x5-2 is not processing a user program. • The technology packages are active. Test and commissioning functions can be executed. The user program is not active. • The I/O modules are in the safe state (this means, for example, digital outputs are "LOW" and analog outputs are de-energized or at zero current).
STOP	SIMOTION D4x5-2 is not processing a user program. • It is possible to load a complete user program. • All system services (communications, etc.) are active. • The I/O modules are in the safe state (this means, for example, digital outputs are "LOW" and analog outputs are de-energized or at zero current). • The technology packages are inactive, i.e. all enables are deleted. No axis motions can be executed.
MRES	Performing a memory reset on the SIMOTION D4x5-2 / restoring the factory setting Using the MRES switch position, you can perform depending on the operating sequence • Memory reset of the SIMOTION D4x5-2 or • Restore the SIMOTION D4x5-2 to its factory setting, depending on the operating sequence. For further details, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Note

It is recommended that SIMOTION SCOUT be used exclusively to switch the operating modes of the module. Therefore, leave the mode switch at position 0 (RUN). The LED display indicates the current mode selection.

For information on how to set the operating state using SIMOTION SCOUT, see the *SIMOTION SCOUT* Configuration Manual.

Service selector switch

The following table shows the possible positions of the service selector switch. The service selector switch positions are explained in the order in which they are arranged on the SIMOTION D4x5-2.

Table 10-150 Switch positions of the service selector switch

Service mode	Switch position	Meaning	
-	0	No service/diagnostic functions activated (default setting)	
Delete/restore non-volatile SIMOTION data	1 or A → 1	When the "Delete/restore non-volatile SIMOTION data" switch setting is selected, the non-volatile data of the D4x5-2 is first deleted and then restored along with the contents of the PMEMORY backup file.	
		Position "1":	The data backed up with the system function _savePersistentMemoryData is preferably restored
		Position "A" → "1": (as of V4.4)	The data backed up by service selector switch position "D" / Web server / DIAG pushbutton are preferably restored
Web server in Security Level Low	8	Switches the SIMOTION IT Web server to Security Level Low for 120 minutes. You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.	
Downgrade (Device Update Tool)	B	SIMOTION D4x5-2 control units and projects can be upgraded using upgrade data created at an earlier point in time. This upgrade data is generated with the Device Update Tool (Menu: "Project>Start Device Update Tool" in SIMOTION SCOUT). If the upgrade process fails to bring about the desired result, the upgrade can be rejected by means of the switch position. This will roll the system back to the previous configuration.	
Backup of diagnostic data and non-volatile SIMOTION data	D	The diagnostic data and non-volatile SIMOTION data can be backed up in STOP, STOPU, and RUN state. The advantage of backing up in RUN state is the availability of enhanced diagnostic information (via HTML pages) and TO alarm information.	

Note

Alternatively, diagnostic data and non-volatile SIMOTION data can also be backed up via the DIAG button, see Section DIAG button (Page 7619).

Additional references

You will find detailed information on the individual topics in the following table:

Table 10-151 References

Subject	Reference
Setting of the operating modes	• *SIMOTION SCOUT* Configuration Manual
Upgrading devices (Device Update Tool)	• *Upgrading SIMOTION Devices* Operating Instructions and • *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
• Creating diagnostic data and • Saving/restoring non-volatile SIMOTION data	• *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual

DIAG button

Layout

The DIAG button is located behind the blanking cover on the SIMOTION D4x5-2.

Function

The diagnostic data and non-volatile SIMOTION data is backed up on the CompactFlash card via the DIAG button.

The DIAG button function therefore corresponds to the function of switch position "D" of the Service selection switch.

Various options are available for backing up the data:

- Option 1: Backup during operation (in STOP/STOPU/RUN operating state)
- Option 2: Backup during the module startup

With option 1, the DIAG button only has to be pressed briefly to trigger the data backup. The DIAG button is therefore preferable to switch position "D" of the Service selection switch.

With option 2, the DIAG button has to be pressed until the boot procedure is completed. As this can take 20-30 seconds, switch position "D" is preferable here.

Additional references

For detailed information on creating diagnostic data and backing up / restoring non-volatile SIMOTION data, refer to the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

RESET button

Arrangement

The RESET button is located behind the blanking cover on the SIMOTION D4x5-2.

Performing a reset operation

A reset causes the entire system to be reset and requires the system to be ramped-up again. It is similar to a "Power On Reset" except that the 24 V power supply does not have to be switched off.

7-segment and LED displays

Arrangement of the displays

The front side of the SIMOTION D4x5-2 has ten LED displays arranged arranged vertically in one row. There is also a 7-segment display below the blanking cover.

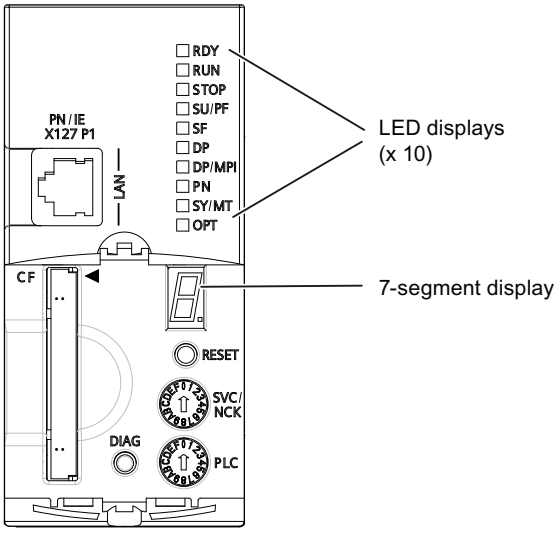


Figure 10-240 7-segment and LED display on the SIMOTION D4x5-2

Meaning of the LED displays

This table describes the LEDs and their meaning. The PN and *SY* LEDs have no function for SIMOTION D4x5-2 DP.

Table 10-152 Error and status displays

LED	Meaning
RDY	Operating states of SIMOTION D incl. SINAMICS Integrated.
RUN	User program is running
STOP	No user program is running. The technology packages are not active
SU/PF	The technology packages are active. The user program is not active
SF	An error state of the SIMOTION D4x5-2
DP	State of the PROFIBUS DP interface
DP/MPI	State of the PROFIBUS DP/MPI interface
PN	State of the onboard PROFINET IO interface (X150)
SY/MT	- Synchronization status (SY) of the onboard PROFINET IO interface (X150) - Maintenance status (MT) of the D4x5-2 (currently without function)
OPT	State of the option module (if available).

Note

While the SIMOTION D4x5-2 is ramping up, all LEDs are briefly illuminated in yellow.

7-segment display

The 7-segment display provides further status information in addition to the LED displays.

The status "6" and a flashing "." indicate that the D4x5-2 has ramped up and communication has been established to the SINAMICS Integrated.

Additional information

You can perform a detailed diagnosis with a PG/PC and the engineering system. Information on the *Diagnostics* via LED displays can also be found in the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual and in the online help of this section via the link under the *Instructions* menu.

10.2.1.4 Interfaces

Interface overview

This section describes the interfaces of the SIMOTION D4x5-2.

The position of the interfaces on the module can be found in the following sections:

- Representation of SIMOTION D425-2 DP and D435-2 DP (Page 7601)
- Representation of SIMOTION D425-2 DP/PN and D435-2 DP/PN (Page 7603)
- Representation of SIMOTION D445-2 DP/PN and D455-2 DP/PN (Page 7605)

Available interfaces

Table 10-153 Overview of available interfaces

Interface	Designation	Connector type
DRIVE-CLiQ interface	X100	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X101	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X102	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X103	DRIVE-CLiQ socket
DRIVE-CLiQ interface (not for SIMOTION D425-2)	X104	DRIVE-CLiQ socket
DRIVE-CLiQ interface (not for SIMOTION D425-2)	X105	DRIVE-CLiQ socket
Ethernet interface PN/IE	X127 P1	RJ45 socket connector
PN/IE OP Ethernet interface (only for SIMOTION D4x5-2 DP)	X120 P1	RJ45 socket connector
Ethernet interface PN/IE-NET	X130 P1	RJ45 socket connector
PROFINET PN IO interface (only for SIMOTION D4x5-2 DP/PN)	X150 (P1, P2, P3)	RJ45 socket connector
Digital I/Os	X122, X132, X142	Mini Combicon, 3.5 mm 3x14-pin
Power supply connector	X124	Combicon, 4-pin
PROFIBUS DP interface	X126	9-pin Sub-D socket
PROFIBUS DP/MPI interface	X136	9-pin Sub-D socket
Measuring sockets (T0, T1, T2, and M)	X141	4-pin, socket
SIMOTION CF plug-in	X109	CompactFlash card connector
Fan/battery module interface	X190/X191	Fan/battery module
1. USB interface	X125	USB socket
2. USB interface	X135	USB socket
Option slot		Sockets

Note

The third port of the PROFINET X150 P3 IO interface has an additional caption for a SIMOTION D4x5-2 DP/PN. It is called X120 PN/IE-OP.

This designation is not relevant for SIMOTION D.

Non-usable interfaces

Table 10-154 Overview of interfaces that cannot be used for SIMOTION D

Interface name	Interface	Connector type
RS232 interface	X140	9-pin Sub-D connector

DRIVE-CLiQ interfaces

DRIVE-CLiQ interfaces

All SINAMICS S120 drive system components, including the motors and encoders, are interconnected by a shared serial interface called DRIVE-CLiQ. The standardized cables and connectors reduce the variety of different parts and cut storage costs.

DRIVE-CLiQ has the following properties:

- Automatic detection of components by the Control Unit
- Standardized interfaces to all components
- Uniform diagnostics down to the components
- Complete service down to the components
- 24 V / 450 mA per DRIVE-CLiQ interface are provided for the connection of encoders and measuring systems.
  Note:
  The DRIVE-CLiQ cable with 24 V supply is used only for components that require this (e.g. motors with a DRIVE-CLiQ interface).

Position of connectors

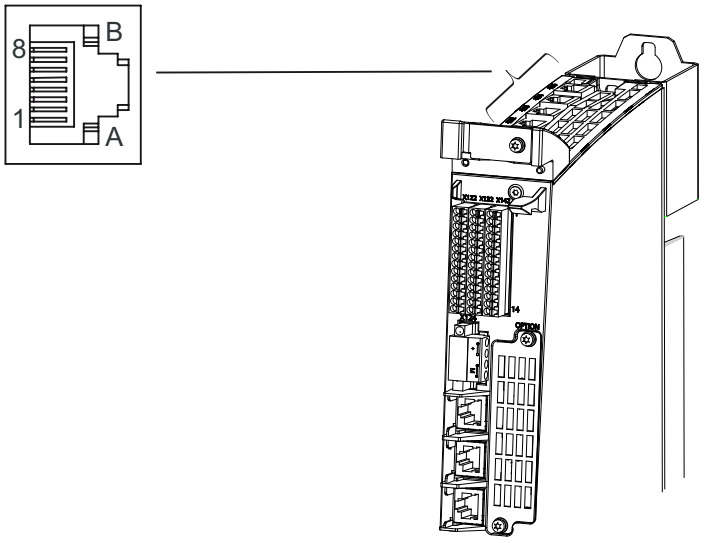


Figure 10-241 The position of the DRIVE-CLiQ interfaces on the D4x5-2 (example of D445-2 DP/PN)

Characteristics

Table 10-155 DRIVE-CLiQ interface X100 – X105 (D425-2: Only X100 - X103)

Characteristic	Type
Connector type	DRIVE-CLiQ connector (RJ45 socket)
Cable type	DRIVE-CLiQ standard (inside the cabinet)
Cable type	MOTION CONNECT (outside the cabinet)
Max. cable length	100 m
Dust protection filler plugs for sealing unused DRIVE-CLiQ ports	Five filler plugs contained in the D4x5-2 scope of delivery Filler plugs (50 pcs) article number: 6SL3066-4CA00-0AA0

DRIVE-CLiQ pin assignment

Table 10-156 DRIVE-CLiQ interface X100 – X105 (D425-2: Only X100 - X103)

PIN	Signal name	Signal type	Meaning
1	TXP	O	Send data +
2	TXN	O	Send data -
3	RXP	I	Receive data +
4	----	----	Reserved, do not use

PIN	Signal name	Signal type	Meaning
5	----	----	Reserved, do not use
6	RXN	I	Receive data -
7	----	----	Reserved, do not use
8	----	----	Reserved, do not use
A	+ (24 V)	VO	Power supply for DRIVE-CLiQ, 450 mA maximum
B	M (0 V)	VO	Ground to 24 V
Signal type: I = Input; O = Output; VO = Voltage Output			

Additional references

- *SINAMICS S120 Control Units and Additional System Components* Manual
- *SINAMICS S120 Booksize Power Units* Manual
- *SINAMICS S120 for AC Drives* Manual
- *SINAMICS S120* Commissioning Manual
- *Terminal Modules TM15 and TM17 High Feature* Commissioning Manual
- *TM15/TM17 High Feature* Manual

PROFINET IO interface (only for SIMOTION D4x5-2 DP/PN)

PROFINET is an open component-based industrial communication system using Ethernet for distributed automation systems.

SIMOTION D4x5-2 DP/PN has a PROFINET interface with three ports (X150 P1-P3) onboard. The PROFINET interface supports operation of a SIMOTION D4x5-2 DP/PN as an IO controller and/or as an I device.

Interface position

The following figure contains information on the PROFINET interface of the control unit. Position of the interface, labeling of the ports and the associated displays are described.

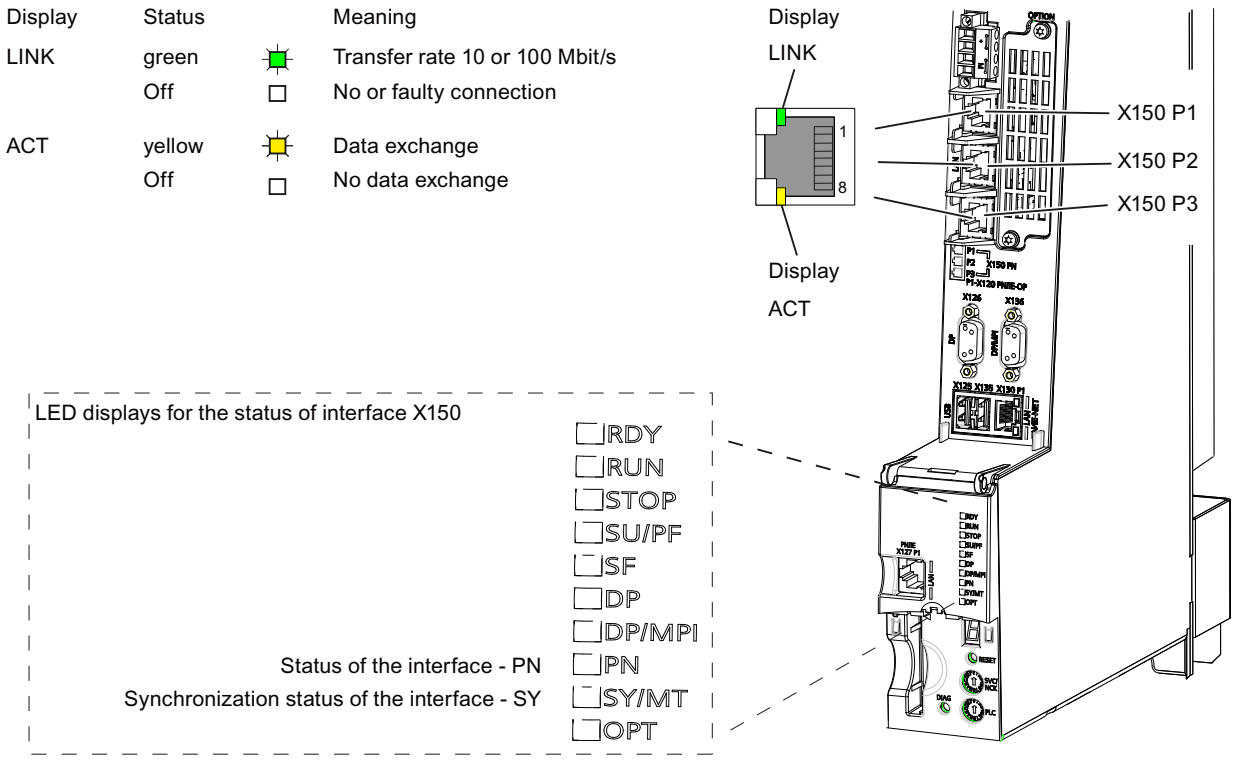


Figure 10-242 The position of the PROFINET X150 P1 to P3 interfaces and their displays (SIMOTION D445-2 DP/PN)

Note

The 3rd port of the PROFINET IO interface X150 P3 is also designated as X120 PN/IE OP. This designation is not relevant for SIMOTION D.

Additional references

Detailed information on the states of the status LEDs can be found in the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Section *Diagnostics*.

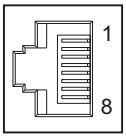
Interface characteristics

Table 10-157 Ports X150 P1 to P3

Characteristic	Type
Connector type	RJ45 socket connector
Cable type	PROFINET
Maximum cable length	100 m
Minimum send cycle clock	0.25 ms (D455-2 DP/PN: 0.125 ms)
Autocrossing	Yes i.e. both crossed and uncrossed cables can be used
Dust protection filler plugs for sealing unused PROFINET ports	Five filler plugs contained in the D4x5-2 scope of delivery Filler plugs (50 pcs) article number: 6SL3066-4CA00-0AA0

Interface assignment

Table 10-158 Assignment of the ports X150 P1 to P3

Representation	Pin	Name	Description
	1	TXP	Send data +
	2	TXN	Send data -
	3	RXP	Receive data +
	4	-	Reserved, do not use
	5	-	Reserved, do not use
	6	RXN	Receive data -
	7	-	Reserved, do not use
	8	-	Reserved, do not use

Connectable devices

The following devices can be connected to the PROFINET IO interface:

- PG/PC programming devices (communication with SIMOTION SCOUT / STEP 7)
- SIMATIC HMI devices
- SIMATIC controllers with PROFINET interface
- Distributed I/O
- Drive units with PROFINET IO interface (standard devices)

The SIMOTION D4x5-2 DP/PN then assumes the role of a PROFINET IO controller and can offer the following functions:

- PROFINET IO controller, I-device (also controller and device simultaneously)
- Supports real-time classes of PROFINET IO:
  - RT (real-time)
  - IRT (isochronous real-time)

The following functions are also supported by Industrial Ethernet:

- Communication between SIMOTION and SIMATIC NET OPC.
  The "SIMATIC NET SOFTNET S7 (S7 OPC server)" software must be installed on the PG/PC for this function.
- Communication with other devices over TCP/IP or UDP communication
- IT communication (e.g. via SIMOTION IT OPC XML-DA)
- Communication based on OPC UA (Unified Architecture)

For further information on the software packages, see Catalog *SIMOTION PM 21*

Note

A list of the modules released with SIMOTION is available at (https://support.industry.siemens.com/cs/ww/en/view/11886029).

The list is updated regularly and contains information on the use of these modules.

Take note of the documentation on the individual modules or devices!

Second PROFINET interface

The Ethernet Communication Board (CBE30-2) provides as option a second PROFINET interface for the D4x5-2 DP/PN Control Units.

The CBE30-2 cannot be used in SIMOTION D4x5 2 DP Control Units.

For details, see Section CBE30-2 Ethernet communication board (Page 7676).

Digital I/Os

Properties

Interface characteristics

The digital I/Os on the X122, X132 and X142 connectors are for the connection of sensors and actuators.

Table 10-159 Wiring of X122, X132 and X142

Features		Type
Connector type		14-way spring-loaded terminal
Number of cables that can be connected		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 1.5 mm²
	Flexible	0.2 mm² ... 1.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.25 mm² ... 1.5 mm²

Features		Type
	Flexible, with wire-end ferrule with plastic sleeve	0.25 mm² ... 0.75 mm²
	AWG / kcmil	24 ... 16
Stripped length		10 mm
Tool		Screwdriver 0.4 x 2.0 mm
Max. cable length		30 m
Max. current carrying capacity (ground)		6 A

Position of connectors

The following figure shows the position of the interface connectors on the D4x5-2 and the distribution of the various digital I/Os.

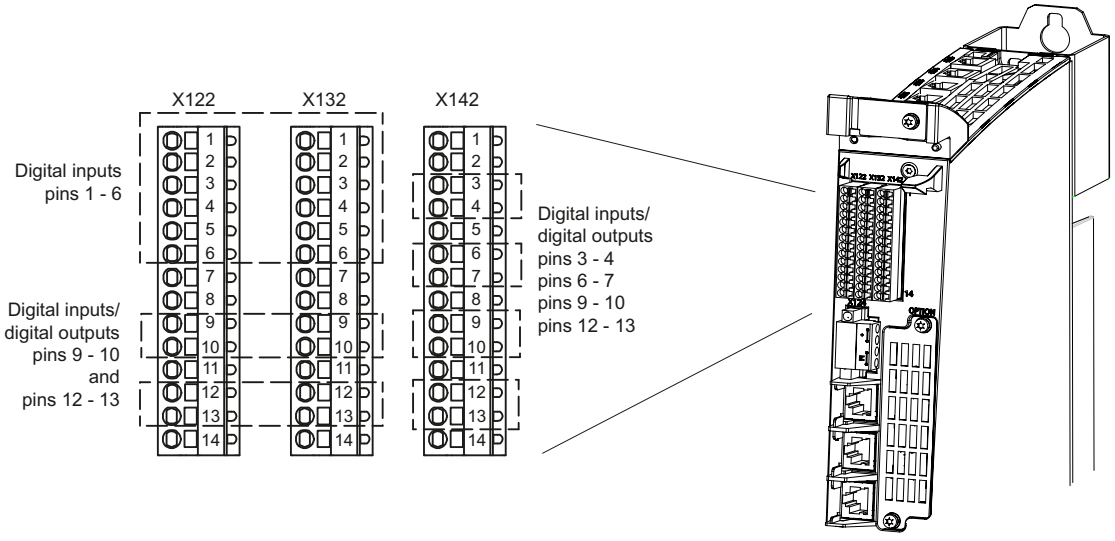


Figure 10-243 Position of the X122, X132 and X142 digital interfaces (example of SIMOTION D445-2 DP/PN)

Connection and circuit diagram for SIMOTION D4x5-2

The following figure shows the wiring and block diagram of the digital inputs as well as the digital input/outputs using the example of a SIMOTION D4x5-2 DP/PN.

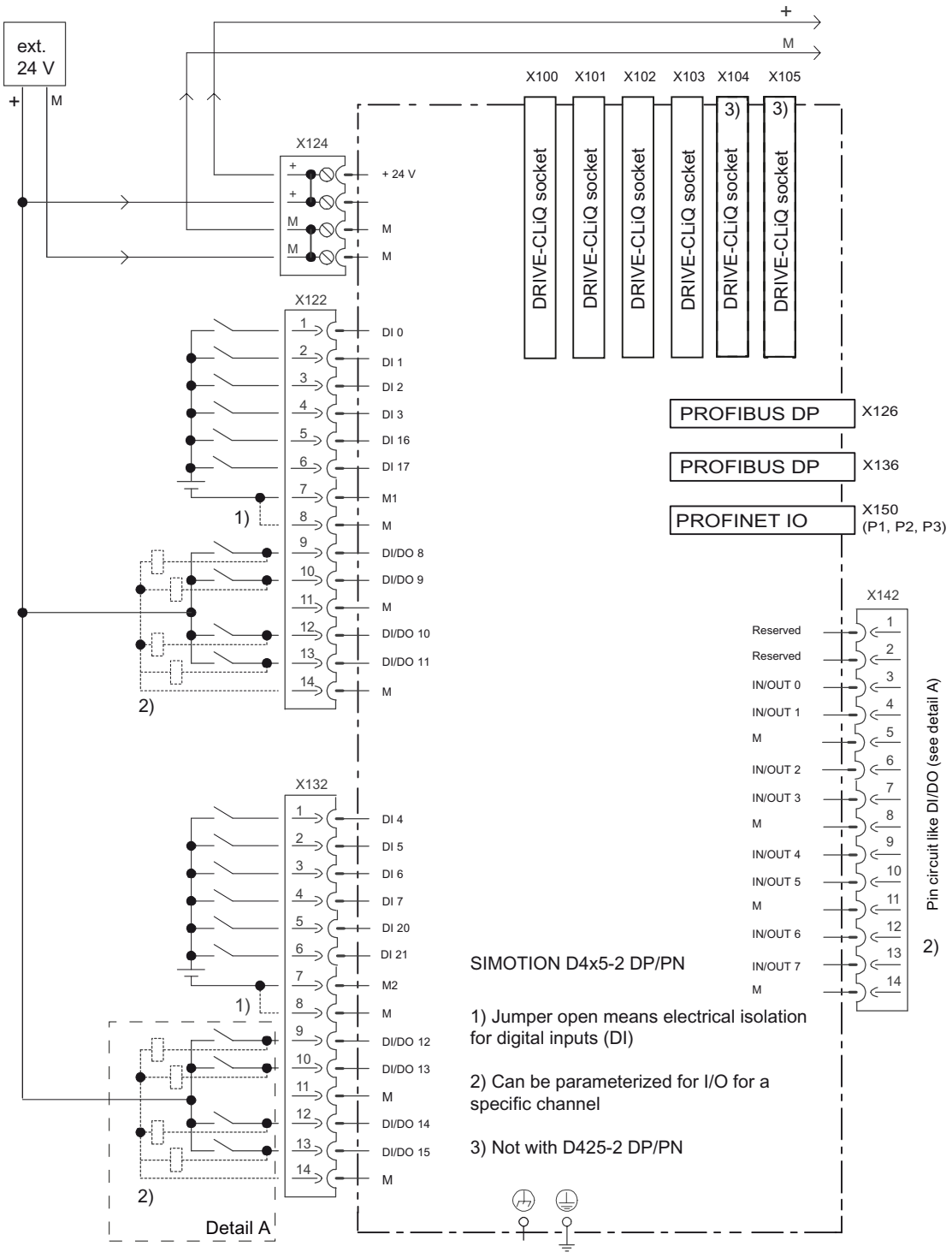


Figure 10-244 Connection and circuit diagram of the digital I/Os

Interface assignment of X122, X132 and X142

Table 10-160 Digital inputs/outputs X122

Pin	Designation [1]	Signal type [2]	Notes
1	DI 0	I	Digital input 0
2	DI 1	I	Digital input 1
3	DI 2	I	Digital input 2
4	DI 3	I	Digital input 3
5	DI 16	I	Digital input 16
6	DI 17	I	Digital input 17
7	M1	GND	Ground for DI 0 – DI 3, DI 16, DI 17 (electrically isolated relative to M)
8	M	GND	Ground
9	DI/DO 8	B	Digital input/output 8 (can also be used as input for measuring input or as input for the external zero mark)
10	DI/DO 9	B	Digital input/output 9 (can also be used as input for measuring input or as input for the external zero mark)
11	M	GND	Ground
12	DI/DO10	B	Digital input/output 10 (can also be used as input for measuring input or as input for the external zero mark)
13	DI/DO 11	B	Digital input/output 11 (can also be used as input for measuring input or as input for the external zero mark)
14	M	GND	Ground

[1] DI: Digital input; DI/DO: Bidirectional digital input/output; M: Electronics ground; M1: Ground reference

[2] B = Bidirectional; I = Input; GND = Reference potential (ground)

Table 10-161 Digital inputs/outputs X132

Pin	Designation [1]	Signal type [2]	Notes
1	DI 4	I	Digital input 4
2	DI5	I	Digital input 5
3	DI 6	I	Digital input 6
4	DI 7	I	Digital input 7
5	DI 20	I	Digital input 20
6	DI 21	I	Digital input 21
7	M2	GND	Ground for DI 4 – DI 7, DI 20, DI 21 (electrically isolated relative to M)
8	M	GND	Ground
9	DI/DO 12	B	Digital input/output 12 (can also be used as input for measuring input or as input for the external zero mark)
10	DI/DO 13	B	Digital input/output 13 (can also be used as input for measuring input or as input for the external zero mark)
11	M	GND	Ground

Pin	Designation [1]	Signal type [2]	Notes
12	DI/DO 14	B	Digital input/output 14 (can also be used as input for measuring input or as input for the external zero mark)
13	DI/DO 15	B	Digital input/output 15 (can also be used as input for measuring input or as input for the external zero mark)
14	M	GND	Ground

[1] DI: Digital input; DI/DO: Bidirectional digital input/output; M: Electronics ground; M2: Ground reference

[2] B = Bidirectional; I = Input; GND = Reference potential (ground)

Note

An open input is interpreted as "low".
To enable the digital inputs to work, terminal M1 or M2 must be connected. The following alternatives are available:

- Connect the carried digital input reference ground to M1 or M2.
- Insert a bridge between terminals M and M1 (or between M and M2).
  This removes the electrical isolation for these digital inputs.

Table 10-162 Digital inputs/outputs X142

Pin	Designation [1]	Signal type [2]	Notes
1	---	---	Reserved, do not use
2	---	---	Reserved, do not use
3	IN/OUT 0	B	Digital input/output 0 (can be used as input of a measuring input or output of an output cam)
4	IN/OUT 1	B	Digital input/output 1 (can be used as input of a measuring input or output of an output cam)
5	M	GND	Ground
6	IN/OUT 2	B	Digital input/output 2 (can be used as input of a measuring input or output of an output cam)
7	IN/OUT 3	B	Digital input/output 3 (can be used as input of a measuring input or output of an output cam)
8	M	GND	Ground
9	IN/OUT 4	B	Digital input/output 4 (can be used as input of a measuring input or output of an output cam)
10	IN/OUT 5	B	Digital input/output 5 (can be used as input of a measuring input or output of an output cam)
11	M	GND	Ground
12	IN/OUT 6	B	Digital input/output 6 (can be used as input of a measuring input or output of an output cam)
13	IN/OUT 7	B	Digital input/output 7 (can be used as input of a measuring input or output of an output cam)
14	M	GND	Ground

[1] IN/OUT: Bidirectional digital input/output; M: Electronic ground

[2] B = Bidirectional; GND = Reference potential (ground)

Using the digital inputs/outputs

Connecting sensors and actuators

Digital I/Os can be used to connect various sensors and actuators to the three 14-pin X122, X132 and X142 front connectors.

The following types of digital I/Os are available:

- Digital inputs (DI)
- Bidirectional digital I/Os (DI/DO, IN/OUT)

Bidirectional digital I/Os can be configured individually as digital inputs or outputs.

Assignment of the I/Os to functions can be parameterized as required. Special functions (e.g. input of measuring input and output for output cam) can also be assigned to the I/Os.

The digital I/Os on the X122 and X132 front connectors can be used by either SIMOTION or SINAMICS (e.g. as enable signal for a drive).
The digital I/Os on the X142 front connector are permanently allocated to SIMOTION.

Table 10-163 Use of the digital I/Os

	DI 0-7, DI 17, DI 18, DI 20, DI 21 (X122, X132)	**DI/DO 8-15 (X122, X132)**	**IN/OUT 0-7 (X142)**
Galvanic isolation	Electrically isolated (ground reference M1 or M2)	Non-isolated (ground reference M)	Non-isolated (ground reference M)
Use as:			
• Freely addressable I/Os for SIMOTION	Yes	Yes	Yes
• I/Os that are assigned to the drive	Yes	Yes	No
• Measuring inputs	No	Yes (global and local measuring inputs)	Yes (global measuring inputs)
• Inputs for the external zero mark	No	Yes	No
• Cam outputs	No	No	Yes
Configuration:			
Assignment	Can be configured channel-by-channel on the drive	Can be configured channel-by-channel on the drive	Can be configured channel-by-channel in HW Config

Note

For optimal noise immunity of the digital inputs, the use of shielded cables is necessary if they are to be used as

- Inputs of measuring inputs or
- Inputs for the external zero mark

Inductive loads

If the digital inputs are used together with inductive loads, it must be ensured that the permissible voltage range of -30 V to +30 V for digital inputs is not exceeded by the switching overvoltage of inductive loads. This is preferably achieved by using auxiliary contacts on the contactor/relay for the digital inputs.
If no auxiliary contacts are available, overvoltage protection must be used. Protective circuits with a freewheeling diode or a diode combination with a Z-diode are possible. RC elements and varistors are not suitable due to the remaining residual voltages.

Additional references

For information on configuring the digital I/Os as freely addressable I/Os, inputs of measuring inputs or outputs of output cams, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

For information on the configuration and function of the measuring input, output cam, and cam track technology objects, refer to the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Power supply

This interface is provided for connection of the external power supply.

Note

When using external power supplies (e.g. SITOP), the ground potential must be connected with the protective ground terminal (PELV).

Note

Ground potential and housing (PE) are connected internally with low impedance.

Note

The 24 V DC cable must be approved for temperatures up to at least 75 °C.
The maximum permissible cable length is 10 m.

Features of the interface

Table 10-164 Interface X124

Features		Type
Connector type		4-way screw-type terminal
Number of cables that can be connected		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 2.5 mm²
	Flexible	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.2 mm² ... 1.5 mm²
	AWG / kcmil	22 ... 12
Stripped length		6 ... 7 mm
Tool		Screwdriver 0.5 x 3 mm (M2.5)
Tightening torque		0.4 to 0.5 Nm (3.5 ... 4.4 lbf in)
Max. current carrying capacity, incl. loop-through		20 A (15 A per UL/CSA)
Max. cable length		10 m

Interface assignments

Table 10-165 Power supply X124

Pin	Signal name	Meaning
1	+	Power supply 24 V
2	+	Power supply 24 V
3	G	Ground
4	G	Ground

Note

The 24 V is looped through via the 24 V connector. In this case, pin 1 is jumpered with pin 2, and pin 3 is jumpered with pin 4 in the connector. The maximum current can be limited through the current carrying capacity of the cable. The current carrying capacity of the cable depends, for example, on the type of cable installation (cable duct, laying on a cable rack, etc.).

Position of power supply interface

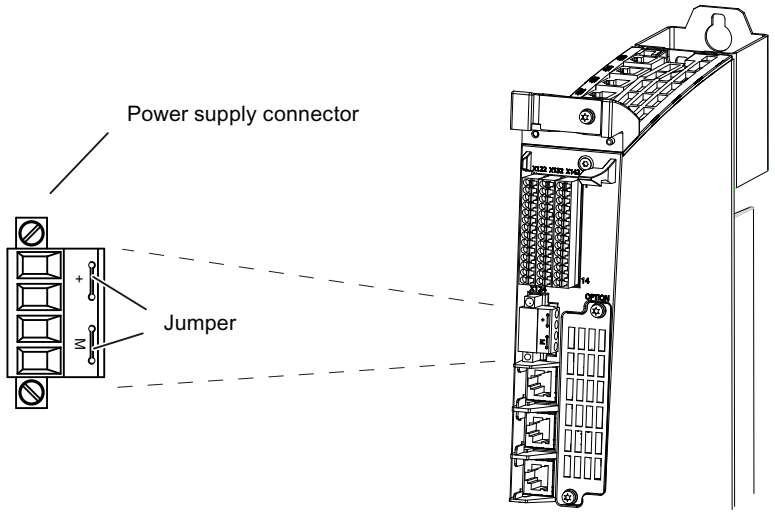


Figure 10-245 Position of the power supply interface (example of SIMOTION D445-2 DP/PN)

Note

The power supply terminal strip must be screwed on tightly using a flat-bladed screwdriver.

Disconnecting 24 V plug-in connections during operation

Observe the following safety instructions when using SIMOTION D4x5-2/CX32-2:

⚠ WARNING

Personal injury and damage to property may occur

Personal injury and damage to property may occur if you disconnect 24 V plug-in connections during operation.
Disconnecting 24 V plug-in connections is only permitted in a de-energized state.

Ethernet interfaces

Interfaces for connection to Industrial Ethernet

Industrial Ethernet is a communications network with a transmission rate of 10/100/1000 Mbit/s.
SIMOTION D4x5-2 offers the following functions via Ethernet interfaces:

- Communication with STEP 7 and SIMOTION SCOUT
- Communication between SIMOTION and SIMATIC NET OPC
  The following software must be installed on the PG/PC for this function:
  "SIMATIC NET SOFTNET-S7 (S7-OPC server)"
- Connection of HMI systems
- Communication with other devices over TCP/IP or UDP communication
- IT communication (e.g. via SIMOTION IT OPC XML-DA).

For further information on the software packages, see Catalog *SIMOTION PM 21*

Position of connectors for SIMOTION D4x5-2 DP

The following figure shows the position of the Ethernet interfaces on the D4x5-2 DP and their displays.

Display	Status		Meaning
LINK	orange		Transfer rate 1000 Mbit/s
	green		Transfer rate 10 or 100 Mbit/s
	Off		No or faulty connection
ACT	yellow		Data exchange
	Off		No data exchange

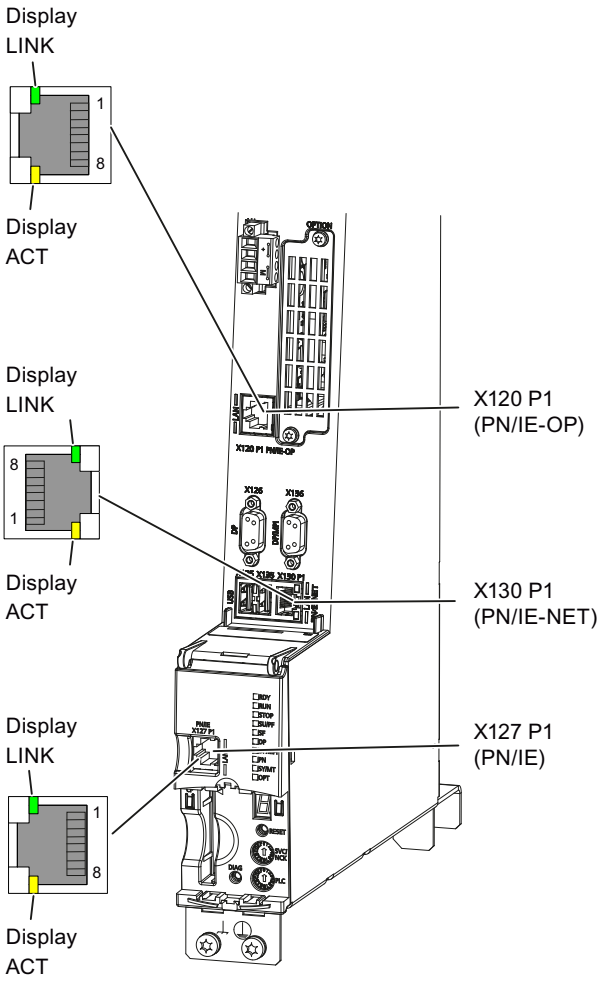


Figure 10-246 The position of the Ethernet interfaces (example of SIMOTION D435-2 DP)

Note

As of V4.3, the three Ethernet interfaces support the PROFINET basic services - they therefore have the designation PN/IE-NET, PN/IE-OP or PN/IE.

These PROFINET basic services (e.g. DCP, LLDP, SNMP) provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives, I/O modules, etc.

Position of connectors for SIMOTION D4x5-2 DP/PN

The following figure shows the position of the Ethernet interfaces on the D4x5-2 DP/PN and their displays.

Display	Status		Meaning
LINK	orange	☀	Transfer rate 1000 Mbit/s
	green	☀	Transfer rate 10 or 100 Mbit/s
	Off	□	No or faulty connection
ACT	yellow	☀	Data exchange
	Off	□	No data exchange

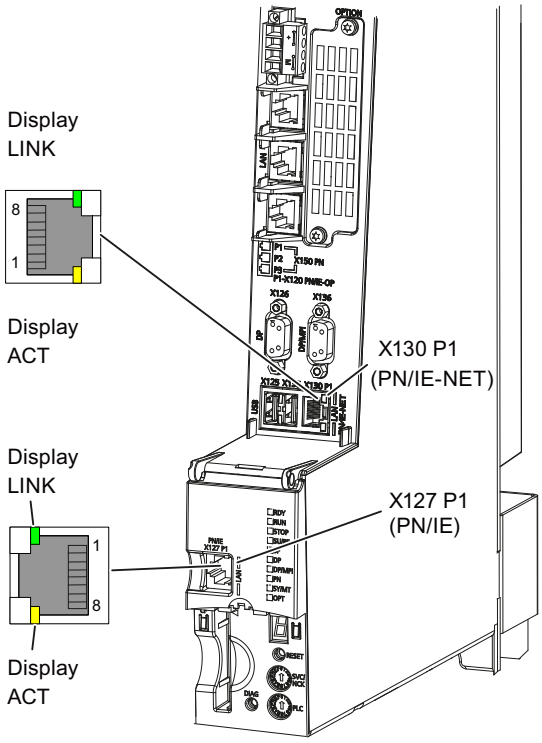


Figure 10-247 The position of the Ethernet interfaces and their displays (example of SIMOTION D445-2 DP/PN)

Note

As of V4.3, the two Ethernet interfaces support the PROFINET basic services - they therefore have the designation PN/IE-NET or PN/IE.

These PROFINET basic services (e.g. DCP, LLDP, SNMP) provide uniform functions for the address assignment and diagnostics, but do not provide PROFINET IO communication for the connection of drives, I/O modules, etc.

Additional references

You will find detailed information on the states of the status LEDs in the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Chapter *Diagnostics*.

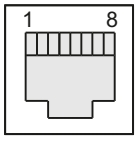
Interface characteristics

Table 10-166 X127 P1 and X130 P1 (X120 P1 only for SIMOTION D4x5-2 DP)

Characteristic	Type
Connector type	RJ45 socket connector
Cable type	Industrial Ethernet cable • 4- and 8-wire cables can be used for 10/100 Mbit/s • 8-wire cables must be used for 1000 Mbit/s
Max. cable length	100 m
Autocrossing	Yes
Dust protection filler plugs for sealing unused Ethernet ports	Five filler plugs contained in the D4x5-2 scope of delivery Filler plugs (50 pcs) article number: 6SL3 066-4CA00-0AA0
Miscellaneous	X127 P1, X120 P1 and X130 P1 are full-duplex 10/100/1000 Mbit/s Ethernet ports

Pin assignment

Table 10-167 X127 P1 and X130 P1 Ethernet interfaces (X120 P1 only for SIMOTION D4x5-2 DP)

Representation	Pin	Assignment in 10/100 Mbit mode			Assignment in 1 Gbit mode		
Signal name [1]	Signal type	Meaning	Signal name [1]	Signal type	Meaning		
	1	TXP	O	Ethernet send differential signal	DA+	B	Bidirectional pair A+
	2	TXN	O	Ethernet send differential signal	DA-	B	Bidirectional pair A-
	3	RXP	I	Ethernet receive differential signal	DB+	B	Bidirectional pair B+
	4	- - -	- - -	Reserved, do not use	DC+	B	Bidirectional pair C+
	5	- - -	- - -	Reserved, do not use	DC	B	Bidirectional pair C-
	6	RXN	I	Ethernet receive differential signal	DB-	B	Bidirectional pair B-
	7	- - -	- - -	Reserved, do not use	DD+	B	Bidirectional pair D+
	8	- - -	- - -	Reserved, do not use	DD-	B	Bidirectional pair D-
I = Input; O = Output; B = Bidirectional							

[1] Autocrossing functionality (if required, send and receive lines switch over)

Note

The MAC addresses are imprinted on an adhesive label that is located behind the protective cover and can be seen from the front.

PROFIBUS DP interfaces

Characteristics of the interface

Table 10-168 Interfaces X126 and X136

Characteristics	Type
Connector type	9-pin Sub-D socket
Cable type	PROFIBUS cable
Max. cable length	100 m at 12 Mbit/s

Position of connectors

The following figure shows the mounting position and designation of the connectors on the control unit.

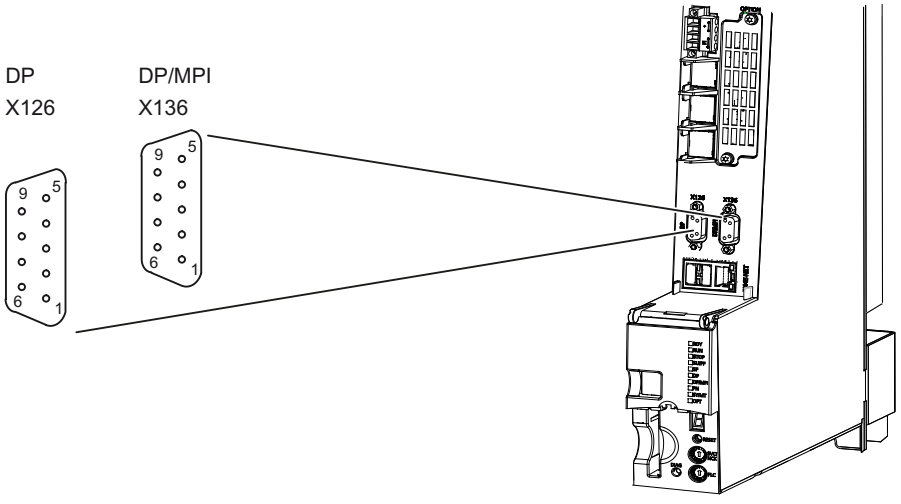


Figure 10-248 The position of the PROFIBUS X126 and X136 interfaces (example of SIMOTION D4x5-2 DP/PN)

Note

For the X126 interface, an adapter connector is available for raising the PROFIBUS connector to make more cabling space. This connector is required in certain wiring scenarios.

For further details, see

- Section Available spare parts and accessories (Page 7702)
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Section Connecting PROFIBUS DP

Interface assignment for X126

Table 10-169 PROFIBUS DP interface X126

Pin	Signal name	Signal type	Meaning
1	--	--	Reserved, do not use
2	M	VO	Ground to P24_SERV
3	1RS_DP	B	RS-485 differential signal
4	1RTS_DP	O	Request to send
5	1M	VO	Ground to 1P5
6	1P5	VO	5 V power supply for bus terminal, external, short-circuit proof
7	P24_SERV	VO	24 V for teleservice, short-circuit proof, 150 mA maximum
8	1XRS_DP	B	RS-485 differential signal
9	--	--	Reserved, do not use
The 1P5 voltage is provided exclusively for the bus terminal. No OLPs are permitted.			
Signal type: VO = Voltage output (power supply); O = Output; B = Bidirectional			

Interface assignment for X136

Table 10-170 PROFIBUS DP interface X136

Pin	Signal name	Signal type	Meaning
1	--	--	Reserved, do not use
2	M	VO	Ground to P24_SERV
3	2RS_DP	B	RS-485 differential signal
4	2RTS_DP	O	Request to send
5	1M	VO	Ground to 1P5
6	1P5	VO	5 V power supply for bus terminal, external, short-circuit proof
7	P24_SERV	VO	24 V for teleservice, short-circuit proof, 150 mA maximum
8	2XRS_DP	B	RS-485 differential signal
9	--	--	Reserved, do not use

Pin	Signal name	Signal type	Meaning
The 1P5 voltage is provided exclusively for the bus terminal. No OLPs are permitted.			
Signal type: VO = Voltage output (power supply); O = Output; B = Bidirectional			

Connectable devices

The following devices can be connected to the PROFIBUS DP interfaces:

- PG/PC
- SIMATIC HMI devices
- SIMATIC controllers with PROFIBUS DP interface
- Distributed I/O
- Teleservice adapter
- Drive units with PROFIBUS DP interface (standard slaves)

Note

For remote diagnosis, a teleservice adapter can be connected to the PROFIBUS X126 or X136 interface. A teleservice adapter can only be connected to one of the two interfaces.

The power supply for the teleservice adapter (terminals 2 and 7) can accept current loads as high as 150 mA and is sustained short-circuit proof.

Slot for CompactFlash card

Characteristics

Connector type: 50-pin connector

This interface should only be used to insert a special SIMOTION CompactFlash card (CF card).

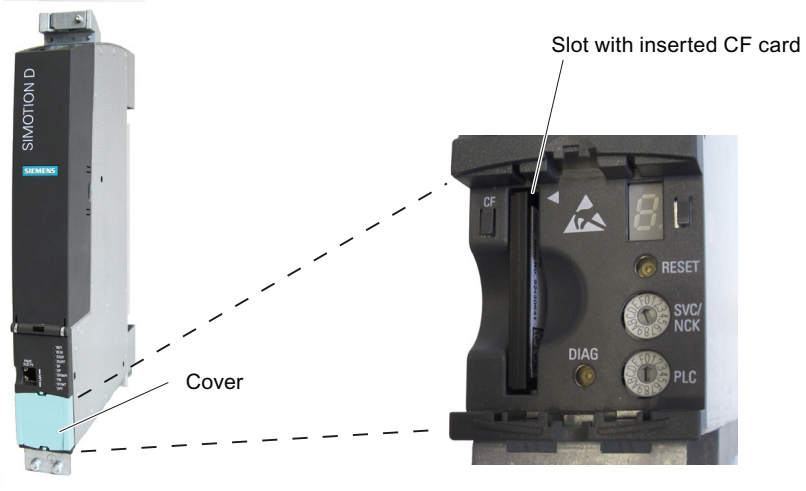


Figure 10-249 Slot for the CompactFlash card

Consult the relevant references for detailed information about the SIMOTION CompactFlash card in Section CompactFlash card (Page 7609).

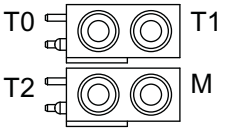
Measuring sockets

Application

The measuring sockets X141 (T0, T1 and T2) are on the lower side of the module and are used to output analog signals. Any interconnectable signal can be output via SINAMICS on every measuring socket on the control unit.

Interface assignment

Table 10-171 Measuring sockets T0, T1, T2

	Socket	Function	Technical specifications
	T0	Measuring socket 0	Voltage: 0 V to 5 V Resolution: 8 bits Load current: Max. 3 mA Continuous short-circuit-proof Reference potential is G terminal
	T1	Measuring socket 1	
	T2	Measuring socket 2	
	M	Ground	
The measuring sockets are suited for multiple-spring wire connectors with a diameter of 2 mm.			

Note

The measuring sockets support commissioning and diagnostic functions. Connection for normal operation is not permitted.

Measuring socket position

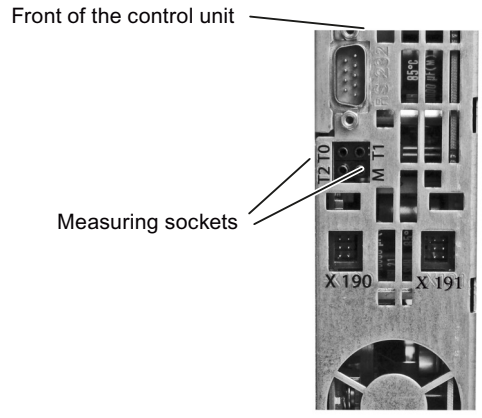


Figure 10-250 Measuring socket position

USB interfaces

The USB interfaces are used for upgrading the SIMOTION D4x5-2 via a USB stick.

Table 10-172 Interfaces X125 and X135

Characteristics	Versions
Connector type	Double USB socket – type A
Version	USB 2.0
Power supply	5 V (short-circuit proof)
Current carrying capacity	0.5 A per channel

The USB interfaces are located on the front of the SIMOTION D4x5-2.

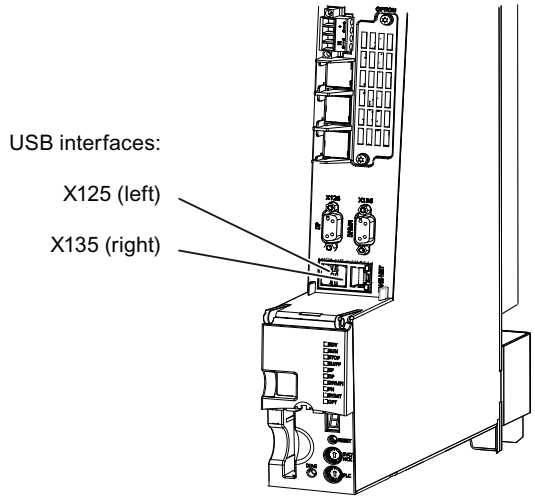


Figure 10-251 The position of the USB X125 and X135 interfaces (example of SIMOTION D4x5-2 DP/PN)

10.2.1.5 Technical data of the D4x5-2

Shipping and storage conditions

Transportation and storage conditions

The following conditions apply to modules that are shipped and stored in the original packaging.

Table 10-173 Transportation conditions

Type of condition	Permissible range/class	
	Transport	**Long-term storage**
Climate class	2K4 according to EN 60721-3-2:1997	1K4 according to EN 60721-3-1:1997
Temperature	From -40° C to +70° C	From -25° C to +55° C
Relative humidity	From 5 to 95%	From 10% to 100%
Height	Max. 4000 m above sea level, For SINAMICS S120 drive components, see SINAMICS Manuals.	
Atmospheric pressure	• > 620 hPa • < 1060 hPa The specified values apply to a transportation altitude of up to 4,000 m	• > 620 hPa • < 1060 hPa The specified values apply to a storage altitude of up to 4,000 m
Biological environmental conditions	Class 2B1 according to EN 60721-3-2:1997	Class 1B1 acc. to EN 60721-3-1:1997
Chemically active environmental conditions	Class 2C2 according to EN 60721-3-2:1997	Class 1C2 according to EN 60721-3-1:1997

Shipping backup batteries

Backup batteries may only be shipped in the original packaging. No special authorization is required to ship backup batteries. The lithium content of the backup battery is approximately 300 mg.

Note

The backup battery is classified as a hazardous substance, Class 9 in accordance with the relevant air-freight transportation regulations.

Notes on handling backup batteries, see Replacing the battery in the fan/battery module.

Storage of backup batteries

Always store backup batteries in a cool and dry place. The batteries have a maximum shelf life of 10 years.

Note

If you have a spare parts inventory, you must not store a SIMOTION D4x5-2 with the fan/battery module mounted. Only connect the fan/battery module if the fan or battery backup voltage is required.

See also

Fan/battery module (Page 7665)

Ambient conditions

Protect the device from environmental effects

SIMOTION D4x5-2 is designed for use in stationary, weather-protected locations.
Protect the device against the following environmental factors:

- Direct sunshine and heat sources
- Mechanical vibration and shock
- Dust
- Humidity
- Strong magnetic fields

Use prohibition

SIMOTION D4x5-2 must not be used in the following applications without additional measures:

- Locations with a high percentage of ionizing radiation
- Locations with extreme operating conditions, e.g.
  - Dust accumulation
  - Corrosive vapors or gases
- Installations requiring special monitoring such as:
  - Elevator installations
  - Electrical installations in particularly hazardous rooms

An additional measure for using SIMOTION D4x5-2 can, for example, be installation in cabinets.

Note

The components must be protected against conductive contamination, e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12.

If conductive contamination can be excluded at the installation site, a lower degree of cabinet protection may be permitted.

See Industrial security (Page 957).

Ambient conditions for operation

SIMOTION D4x5-2 can be used under the following ambient conditions:

Ambient conditions	Fields of application	Remarks
Climatic ambient conditions		
Climate class	3K3	According to EN 60721-3:1995 Ambient conditions better than 3K3
Permissible ambient temperature: During operation when installed vertically	From 0° to 55° C, to 2000 m above sea level	The max. ambient temperature decreases as of an elevation of 2000 m by 7° C per 1000 m increase in elevation. The maximum supply air temperature for all modules is 55° C. The fan/battery module is always required for SIMOTION D4x5-2 Control Units.
Relative humidity	From 5 to 95 %	For SIMOTION D4x5-2
	From 0 to 100 %	For SIPLUS D4x5-2
Condensation, formation of ice, drip, spray, and splash water	Not permissible	For SIMOTION D4x5-2
	Condensation/frost permitted (no commissioning under condensation)	For SIPLUS D4x5-2
Conformal coating	No	For SIMOTION D4x5-2
	Yes	For SIPLUS D4x5-2

Installation altitude	Max. 4000 m above sea level	For SINAMICS S120 drive components, see SINAMICS Manuals.
Atmospheric pressure	620 hPa ... 1060 hPa	Corresponding elevation of 4000 m - 0 m above sea level
Biological, chemical and mechanical influences, pollutants		
Biological ambient conditions:	For SIMOTION D4x5-2 Class 3B1 according to EN 60721-3-3:1995 Mold, mold growth, slime, rodents, termites and other animal vermin are not permissible	
	For SIPLUS D4x5-2 Resistant to biologically active substances, conformity according to EN 60721-3-3. Class 3B2, mold and fungal spores (except fauna) For operation in a harmful gas atmosphere, the plug covers included in delivery must be left on the unused interfaces!	
Chemically active environmental conditions	For SIMOTION D4x5-2 Class 3C1 according to EN 60721-3-3:1995	
	For SIPLUS D4x5-2 Resistant to chemically active substances, conformity according to EN 60721-3-3. Class 3C4 incl. salt spray according to EN 60068-2-52 (severity 3); the plug covers included in delivery must be left on the unused interfaces!	
Mechanically active environmental conditions	Class 3S1 according to EN 60721-3-3:1995, conductive dust not permitted	
Vibratory load - D4x5-2 - CX32-2 - CBE30-2 - S120 (without SME/DME)	Vibratory load during operation according to EN 60721-3-3: Class 3M1 Test values according to EN 60068-2-6 (sinusoidal) • 10 ... 57 Hz: 0.075 mm deflection amplitude • 57 ... 150 Hz: 1 g acceleration amplitude • 10 frequency cycles per axis	
Shock load - D4x5-2 - CX32-2 - CBE30-2 - S120 (without SME/DME)	Shock load during operation according to EN 60721-3-3: Class 3M1 Test values according to EN 60068-2-27 (half-sinusoidal) • 5 g peak acceleration, 30 ms duration • 3 shocks in all three axes in both directions	

SIMOTION D435-2 DP/PN and D455-2 DP/PN are also available as SIPLUS version for use under harsh environmental conditions, e.g. in toxic atmospheres. As BasedOn products, the SIPLUS versions have the same functionality as the standard modules and are configured in the same way.

Permitted mounting positions

The following mounting positions are permitted:

- vertical installation (preferred standard mounting position)
- lying on back (e.g. for applications in which the installation situation makes a low installation height necessary)

For details, see *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Section Alternative mounting position.

Other data

Table 10-174 Other data

Data	
Degree of protection according to EN 60529 (IEC 60529)	IP 20
Pollution degree	2 according to EN 60664-1:2008

Dimensions and weights

Dimensions and weights

Table 10-175 Dimensions and weight of a SIMOTION D4x5-2

Parameter	**D425-2 DP** **D425-2 DP/PN** **D435-2 DP** **D435-2 DP/PN**	**D445-2 DP/PN** **D455-2 DP/PN**
Dimensions W x H x D [mm]		
• Without fastening using spacers, without fan/battery module	50 x 380 x 230	50 x 380 x 230
• With fastening using spacers, without fan/battery module	50 x 380 x 270	50 x 380 x 270
Weight [g] - Without packaging - With packaging	 Approx. 3.7 kg Approx. 4.5 kg	 Approx. 4.3 kg Approx. 5.0 kg

Note

The spacers can be removed for D425-2 and D435-2.

D425-2 and D435-2 do not have any cooling fins.

The presence of cooling fins for D445-2 DP/PN and D455-2 DP/PN means the spacers can be removed only for the "external cooling" installation type. In this installation method, the cooling fins are inserted through a cutout in the rear cabinet panel.

For further details, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Power supply

External 24 V power supply

The control unit is supplied by an external 24 V power supply (e.g. SITOP).

The tolerance range for the input voltage of the SIMOTION D4x5-2 is between 20.4 and 28.8 VDC.

Table 10-176 Power supply data

	D425-2 DP **D435-2 DP**	**D425-2 DP/PN** **D435-2 DP/PN**	**D445-2 DP/PN** **D455-2 DP/PN**
Power supply • Rated value • Permissible range	Safety extra-low voltage DVC A (PELV) 24 VDC (20.4 ... 28.8 V)		
Current consumption, typically [1]	0.7 A	1.0 A	1.9 A
Starting current, typically [1]	5 A	5 A	5 A
Power loss, typically [1]	17 W	24 W	46 W

[1] With no load on inputs/outputs, no 24 V supply via DRIVE-CLiQ and PROFIBUS interfaces

Table 10-177 Input voltage specification

Input voltage	**D4x5-2**
Minimum input voltage	20.4 V
Nominal input voltage	24 V
Maximum input voltage	28.8 V

Table 10-178 Specification of the input current - typical current consumption

	Typical current consumption [1]		
Device type	**D425-2 DP** **D435-2 DP**	**D425-2 DP/PN** **D435-2 DP/PN**	**D445-2 DP/PN** **D455-2 DP/PN**
Minimum input voltage	0.8 A	1.2 A	2.24 A
Nominal input voltage	0.7 A	1.0 A	1.9 A
Maximum input voltage	0.6 A	0.8 A	1.58 A

[1] With no load on inputs/outputs and no 24 V supply via DRIVE-CLiQ or PROFIBUS interface

Table 10-179 Specification of the input current - maximum current consumption

	Maximum current consumption				
Device type	**D425-2 DP**	**D425-2 DP/PN**	**D435-2 DP**	**D435-2 DP/PN**	**D445-2 DP/PN** **D455-2 DP/PN**
Minimum input voltage	12.8 A	13.6 A	13.9 A	14.6 A	18.35 A
Nominal input voltage	10.9 A	11.5 A	11.8 A	12.4 A	15.6 A
Maximum input voltage	9.1 A	9.6 A	9.8 A	10.3 A	13 A

Note

If the D4x5-2 detects undervoltage, the module performs a RESET. If all LEDs are off, either there is no power supply or the voltage level is too low.

When the voltage level is in the permissible range again, the D4x5-2 restarts.

Undervoltages are detected when:

- The voltage level of the 24-V supply falls below the minimum permissible input voltage of the D4x5-2
- A temporary voltage dip (> 3 ms) results in the supply falling below the minimum permissible input voltage on the D4x5-2.

Additional references

Recommended power supply units and tables for calculating the current consumption for the assembly with SINAMICS S120 modules can be found in the "Control cabinet installation and EMC booksize" chapter in the *SINAMICS S120 Booksize Power Units* Manual.

Interfaces and performance features

Memory for system data

Table 10-180 Memory for system data and its memory capacity

Data	Memory capacity D425-2	Memory capacity D435-2	Memory capacity D445-2 DP/PN	Memory capacity D455-2 DP/PN
Diagnostic buffer (non-volatile)	200 messages (SIMOTION) 200 messages (SINAMICS Integrated)	200 messages (SIMOTION) 200 messages (SINAMICS Integrated)	200 messages (SIMOTION) 200 messages (SINAMICS Integrated)	200 messages (SIMOTION) 200 messages (SINAMICS Integrated)
RAM (working memory) [1]	76 MB (as of V5.3 SP1) 64 MB (V4.4 - V5.2 SP1) 48 MB (up to V4.3 SP1)	105 MB (as of V5.3 SP1) 86 MB (V4.4 - V5.2 SP1) 64 MB (up to V4.3 SP1)	190 MB (as of V5.3 SP1) 160 MB (V4.4 - V5.2 SP1) 128 MB (up to V4.3 SP1)	380 MB (as of V5.3 SP1) 320 MB (V4.4 - V5.2 SP1) 256 MB (up to V4.3 SP1)
RAM disk (load memory)	38 MB (as of V5.3 SP1) 31 MB (up to V5.2 SP1)	50 MB (as of V5.3 SP1) 41 MB (up to V5.2 SP1)	68 MB (as of V5.3 SP1) 56 MB (V4.3 SP1 - V5.2 SP1) 50 MB (V4.2 SP1)	90 MB (as of V5.3 SP1) 76 MB (V4.3 SP1 - V5.2 SP1) 70 MB (V4.2 SP1)
Retentive memory	364 KB	364 KB	512 KB	512 KB
Persistent memory (user data on CF)	2 GB CF: 1.5 GB 1 GB CF: 300 MB	2 GB CF: 1.5 GB 1 GB CF: 300 MB	2 GB CF: 1.5 GB 1 GB CF: 300 MB	2 GB CF: 1.5 GB 1 GB CF: 300 MB

[1] A separate 20 MB of working memory is available for Java applications.

Note

The memory capacities may be increased for the current version after the time for going to press for the documentation. The latest values can be found at Internet (https://support.industry.siemens.com/cs/ww/en/view/18857317)

PLC and motion control performance

Data	D425-2 DP D425-2 DP/PN D435-2 DP	D435-2 DP/PN	D445-2 DP/PN	D455-2 DP/PN
Minimum PROFINET send cycle clock	D425-2 DP/PN: 0.25 ms	0.25 ms	0.25 ms	0.125 ms (as of V4.5) [1] 0.25 ms (to V4.4)
Minimum servo/ interpolator cycle clock (with SINAMICS Integrated / CX32-2)	0.5 ms	0.25 ms (as of V4.5) 0.5 ms (to V4.4)	0.25 ms (as of V4.5) 0.5 ms (to V4.4)	0.25 ms (as of V4.5) 0.5 ms (to V4.4)
Minimum servo/ interpolator cycle clock (with Servo_fast and IPO_fast)	---	0.25 ms	0.25 ms	0.125 ms (as of V4.5) [1] 0.25 ms (to V4.4)

[1] 0.125 ms (only with ET 200SP, SCOUT TIA and Servo_fast/IPO_fast)

You can find detailed information on the cycle clock settings for cycle clocks ≤ 0.25 ms in the *SIMOTION D4x5-2 Commissioning and Hardware Installation Manual*.

Integrated drive control

Table 10-181 Controls for integrated drives

Data	SIMOTION D4x5-2 DP/PN
Max. number of axes for integrated drive control (servo/vector/*V/f*)	6 / 6 / 12 (alternative) Drive control based on SINAMICS S120 CU320-2, firmware version V4.x/V5.x

Communication

Table 10-182 Interfaces

Data	SIMOTION D4x5-2 DP	SIMOTION D4x5-2 DP/PN
DRIVE-CLiQ interfaces	6 (D425-2 DP: 4)	6 (D425-2 DP/PN: 4)
Ethernet interfaces	3	2
PROFIBUS interfaces [1]	2	2
PROFINET interfaces [2]	---	Onboard: 1 interface with 3 ports Optionally: Second interface with 4 ports via CBE30-2

[1] Supports PROFIBUS DP, isochronous, can be configured either as master or slave

[2] Supports PROFINET IO with IRT and RT, can be configured as PROFINET IO controller and/or device

Address space

Table 10-183 Address space

		SIMOTION D4x5-2 DP	SIMOTION D4x5-2 DP/PN
Logical I/O address space		16 KB	16 KB
Physical I/O address space for each interface, one each for inputs and outputs			
	PROFIBUS	1 KB [1]	1 KB [1]
	PROFIBUS Integrated	4 KB [1]	4 KB [1]
	PROFINET		6 KB (as of V4.5) [1][2] 4 KB (to V4.4) [1][2]
Permanent process image for BackgroundTask (I/O variables)		64 bytes	64 bytes
Additional configurable process image for each cyclic task (I/O variables)		Yes	Yes
Address space for each PROFIBUS DP station		244 bytes	244 bytes
Address space for each SINAMICS Integrated/ CX32-2 (PROFIBUS Integrated)		512 bytes	512 bytes
Address space for each PROFINET device		---	1400 bytes
Maximum consistency range			
	Onboard PROFINET interface X150 • For controller-controller direct data exchange • For I-Device	---	 254 bytes 1024 bytes (as of V4.4) 254 bytes (< V4.4)
	CBE30-2 • For controller-controller direct data exchange • For I-Device	---	 254 bytes 254 bytes

[1] When PROFIBUS and PROFINET are used, the total address space applies: All I/O are assigned in the logical I/O address space. The maximum number of I/Os is limited by the number that can be addressed using the logical I/O address space.

[2] A second PROFINET interface for the D4x5-2 DP/PN Control Units is optionally available for the CBE30-2.

Digital inputs

Table 10-184 Digital inputs on SIMOTION D4x5-2

Data	SIMOTION D4x5-2
Digital inputs	12
• Rated value	24 VDC
• For signal "1"	15 ... 30 V
• For signal "0" [2]	-3 ... +5 V
Galvanic isolation	Yes, in groups of 6 [1]

Data	SIMOTION D4x5-2
Current consumption typ. at signal level "1"	9 mA at 24 V
Input delay, typ. (hardware)	Signal "0" → "1": 50 µs Signal "1" → "0": 150 µs

1) Reference potential is terminal G1 or G2

2) The digital inputs are protected against polarity reversal up to -30 V

Digital inputs/outputs (parameterizable)

Table 10-185 Digital inputs/outputs on SIMOTION D4x5-2

Data	SIMOTION D4x5-2
Digital inputs/outputs	16 • Max. 16 as measuring input inputs • Max. 8 as output cam outputs
If used as an input	
• Input voltage, rated value	24 VDC
• Input voltage, for signal "1"	15 ... 30 V
• Input voltage, for signal "0" 2)	-3 ... +5 V
Galvanic isolation	No
Current consumption typ. at signal level "1"	9 mA at 24 V
Input delay, typ. (hardware) 3)	Signal "0" → "1": 5 µs Signal "1" → "0": 50 µs
Measuring input input, resolution	1 µs
Measuring input input, reproducibility	5 µs
If used as an output	
• Rated load voltage, permissible range	24 VDC, 20.4 ... 28.8 V
• Galvanic isolation	No
• Current load, max.	500 mA per output
• Residual current, max.	2 mA
• Output delay time, typ./max. (hardware) 1)	Signal "0" → "1": 150/400 µs Signal "1" → "0": 75/150 µs
• Output cam output, resolution terminal X142	1 µs
• Output cam output, reproducibility terminal X142	10 µs
Switching frequency of the outputs, max.	
• With resistive load	4 kHz
• With inductive load	2 Hz

Data	SIMOTION D4x5-2
• With lamp load	11 Hz
Short-circuit protection	Yes

1) Specification for Vcc = 24 V, load 48 Ohm, H = 90% V_{Out}; L = 10% V_{Out}

2) The digital inputs are protected against polarity reversal up to -30 V

3) A filter time of 1 µs or 125 µs can also be set for the X142 inputs.

Max. switching frequency of the DO

The max. switching frequency of the hardware depends on the load. For an ohmic load of 24 V / 0.5 A, it is up to 4 kHz (typical value; low-high ratio = 50:50; short cable lengths).

Logic control of the digital output is also a limiting factor.

- If an X142 DO is set or reset via the TO cam / TO camTrack, up to two edges are possible per servo or ServoFast cycle.
  → with servo cycles of at least 500 µs, a max. switching frequency of 2 kHz is achieved
  → with a ServoFast cycle clock (D435-2 DP/PN to D455-2 DP/PN only) of 250 µs, a max. switching frequency of 4 kHz is achieved
- If an X142 DO is set or reset from the user program, no more than one edge is possible per servo or ServoFast cycle.
  → with servo cycles of at least 500 µs, a max. switching frequency of 1 kHz is achieved
  → with a ServoFast cycle clock (D435-2 DP/PN to D455-2 DP/PN only) of 250 µs, a max. switching frequency of 2 kHz is achieved
- If an X122/X132 DO is set or reset from the user program, no more than one edge is possible per servo cycle.
  → with servo cycles of at least 500 µs, a max. switching frequency of 1 kHz is achieved
  The max. achievable switching frequency can also be limited by the CU parameter p0799[0] (sampling time of the inputs/outputs of the CU) or p2048 (PROFIdrive PZD sampling time).

Reproducibility

The reproducibility at the measuring input input depends on the edge steepness of the measurement signal. Generally, the following is valid: The steeper the edges of the input signal are, the easier it is to reproduce the measurement results. Sloping signals are achieved by switching the signal level to "active". This is typically with rising edges as the signal here is "actively" switched to HIGH by the digital output (example: Output of a TM17 module: Rising edge).

Falling edges typically have less edge steepness (signal level that falls slowly), as the signal level is "not actively" forced to LOW (example: Output of a TM17 module: Falling edge).

Recommendation:
Where the connected components do not have special output drivers, the recommendation is to use the rising edges for measurements.

Further technical data

Table 10-186 Fan, non-volatile data backup, and approvals

Data	SIMOTION D425-2 DP SIMOTION D425-2 DP/PN SIMOTION D435-2 DP SIMOTION D435-2 DP/PN	SIMOTION D445-2 DP/PN SIMOTION D455-2 DP/PN
Fan	Double fan/battery module is included in the scope of delivery	Double fan/battery module is included in the scope of delivery
• Backup time, min. • Charging time, typ.	• 4 days (real-time clock backup) • A few minutes	• 4 days (real-time clock backup) • A few minutes

If a double fan/battery module is used with included battery, the backup time of the real-time clock is at least 3 years.

For further technical data such as the max. number of online connections, HMI devices that can be used as well as a list of tasks available in the execution system, see the function overview in the *SIMOTION PM 21* Catalog.

CompactFlash card

CompactFlash card

Table 10-187 CF card

Memory capacity	2 GB (Article no. 6AU1400-2QA20-0AA0) 1 GB (Article no. 6AU1400-2PA23-0AA0) 1 GB (Article no. 6AU1400-2PA22-0AA0) 1 GB (Article no. 6AU1400-2PA21-0AA0)
Weight	10 g

Clock

Properties of the real-time clock

The following table lists the properties and functions of the SIMOTION D clock.

Table 10-188 Clock properties

Properties	Meaning
Type	Hardware clock (integrated "real-time clock")
Default setting when delivered	12:00 a.m. (date 01.01.2001)
Accuracy • +25° C • -40° C ... +85° C	Max. deviation per day: • ±2 s • ±5 s

Properties	Meaning
Backup time at least	• 4 days (at 0 ... 25° C) • With battery in the double fan/battery module 3 years
Charging time	A few minutes
Backup	Maintenance-free SuperCap or battery in the double fan/battery module

With power OFF

In the POWER OFF state, the SIMOTION D clock continues to run during the backup time (with the exception of the software clock). The backup battery is recharged in the POWER ON state.

An error message is output if the backup function is defective. When the power is switched ON, the clock then resumes at the time set at the factory.
If the SIMOTION D4x5-2 is reset to its factory setting, the clock is also reset to the "default setting when delivered".

Input and output circuit

Protective circuit

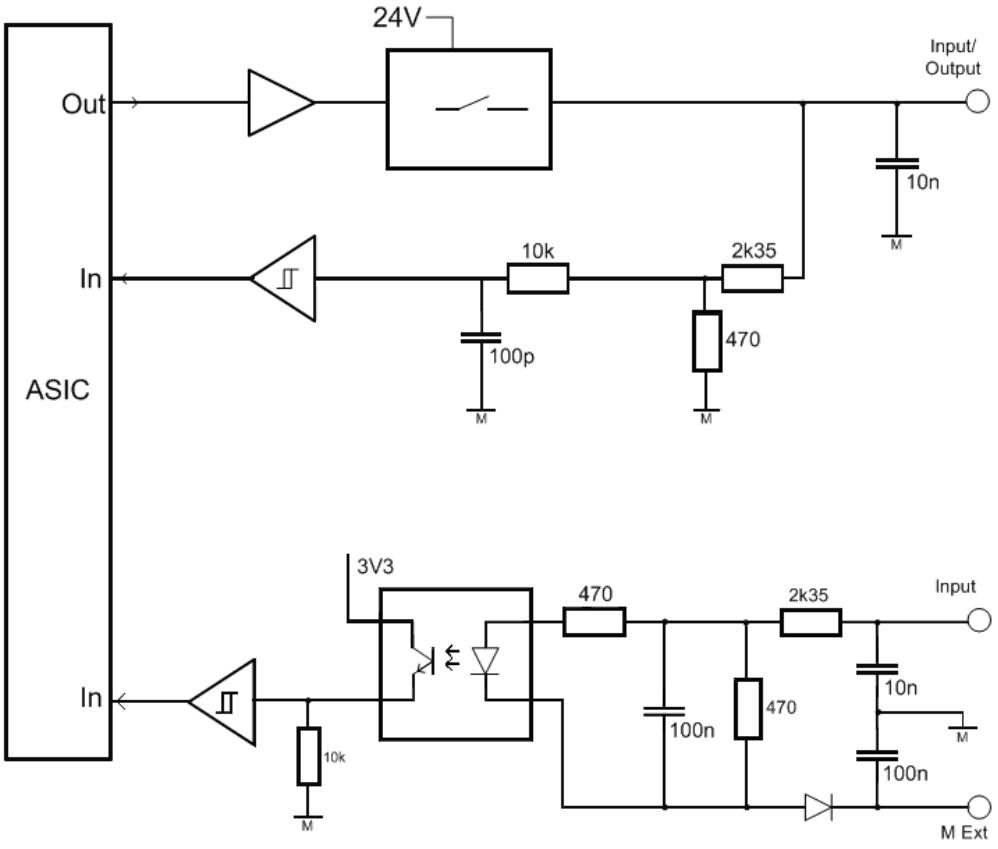


Figure 10-252 SIMOTION D4x5-2/CX32-2 input and output circuits

Certificates, approvals, declarations of conformity

You can find an overview of the certifications available for the SIMOTION D4x5-2 in Appendix A (Page 7705).

You can also find further information on the Internet at:

https://support.industry.siemens.com/cs/ww/en/ps/14513/cert

Note

SIMOTION D435-2 DP/PN and D455-2 DP/PN are also available as SIPLUS version for use under harsh environmental conditions, e.g. in toxic atmospheres.

The SIPLUS versions can have different certifications/approvals. For details, refer to the technical data for the module in the Industry Mall.

https://mall.industry.siemens.com

10.2.1.6 Dimension drawings

D425-2 and D435-2 dimension drawing

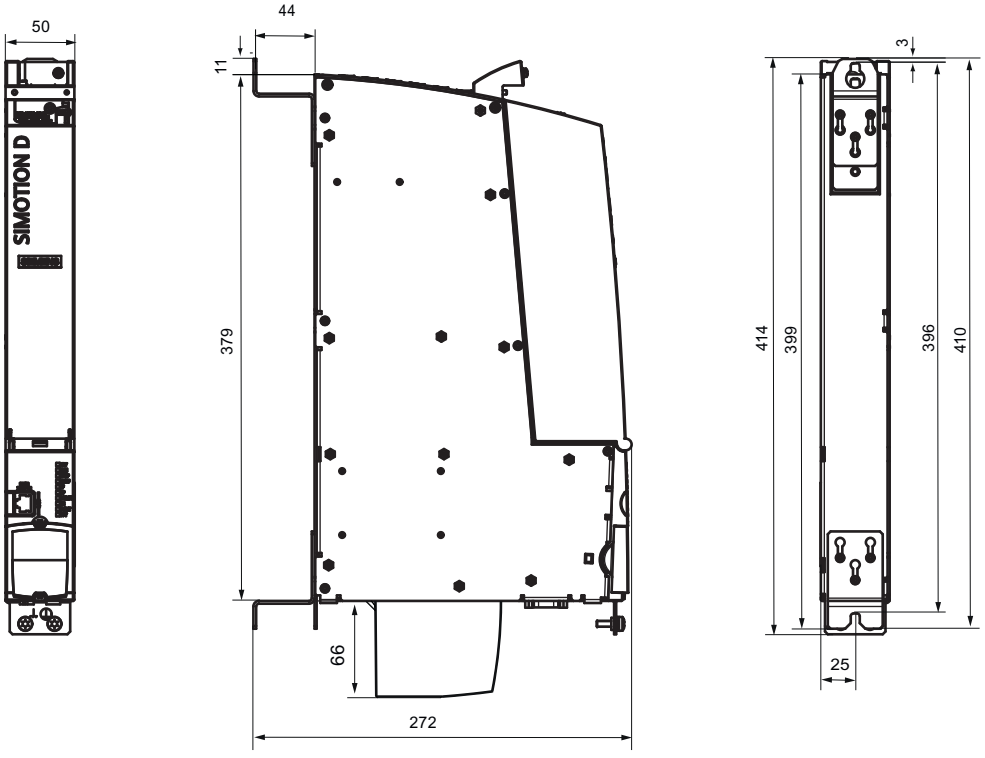


Figure 10-253 D425-2 and D435-2 dimension drawing (dimensions in mm)

SIMOTION D425-2 DP, D425-2 DP/PN, D435-2 DP and D435-2 DP/PN must always be operated with a double fan/battery module.

NOTICE
Higher operating temperature if ventilation clearances are too small
The 80 mm clearances above and below the components must be observed.
The unit protects itself from overheating by shutting down.
The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

D445-2 DP/PN and D455-2 DP/PN dimension drawing

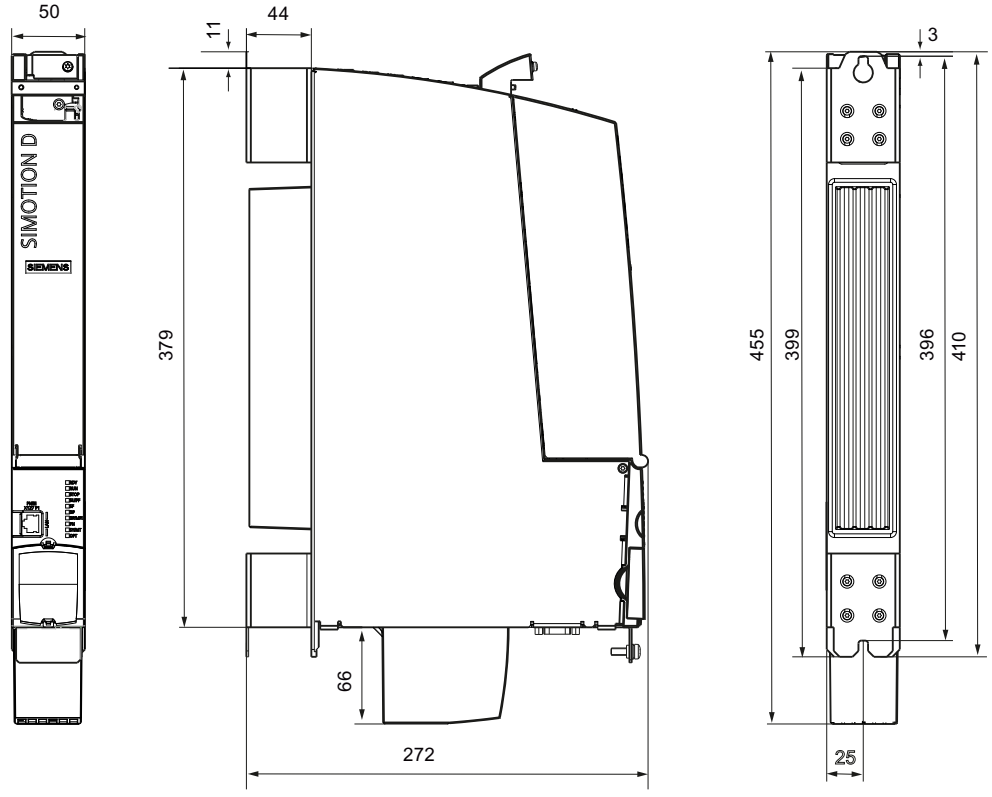


Figure 10-254 D445-2 DP/PN and D455-2 DP/PN dimension drawing (dimensions in mm)

SIMOTION D445-2 DP/PN and D455-2 DP/PN must always be operated with a double fan / battery module.

NOTICE
Higher operating temperature if ventilation clearances are too small
The 80 mm clearances above and below the components must be observed.
The unit protects itself from overheating by shutting down.
The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

CAD data, dimension drawings, and circuit-diagram macros

Dimension drawings and CAD data

Dimension drawings, as well as 2D and 3D CAD data can be found:

- In the CAD Creator (https://support.industry.siemens.com/cs/ww/en/view/30559271)
- In the DT Configurator (http://www.siemens.com/dt-configurator) of the Industry Mall
- Via CAx Download Manager (https://support.industry.siemens.com/my/ww/en/CAxOnline#CAxOnline))

Circuit-diagram macros

EPLAN circuit-diagram macros are available for SIMOTION D. The macros assist you when creating circuit diagrams.

The EPLAN circuit diagram macros can be ordered at the following Internet addresses:

- Drive Technology Configurator (http://www.siemens.com/dt-configurator)
- CAx Download Manager (https://support.industry.siemens.com/my/ww/en/CAxOnline#CAxOnline))
- Product support (https://support.industry.siemens.com/cs/ww/en/view/31622426)

See also

Macros (https://support.industry.siemens.com/cs/ww/en/view/31622426)

10.2.1.7 Supplementary system components

Connection options overview

Supplementary system components

The following figure shows the connection of the supplementary system components. The connection is:

- Directly on the SIMOTION D module (fan/battery module)
- Via the option slot (TB30, CBE30-2)
- Via the DRIVE-CLiQ interfaces (Terminal Modules, Control Unit Adapter, ...).

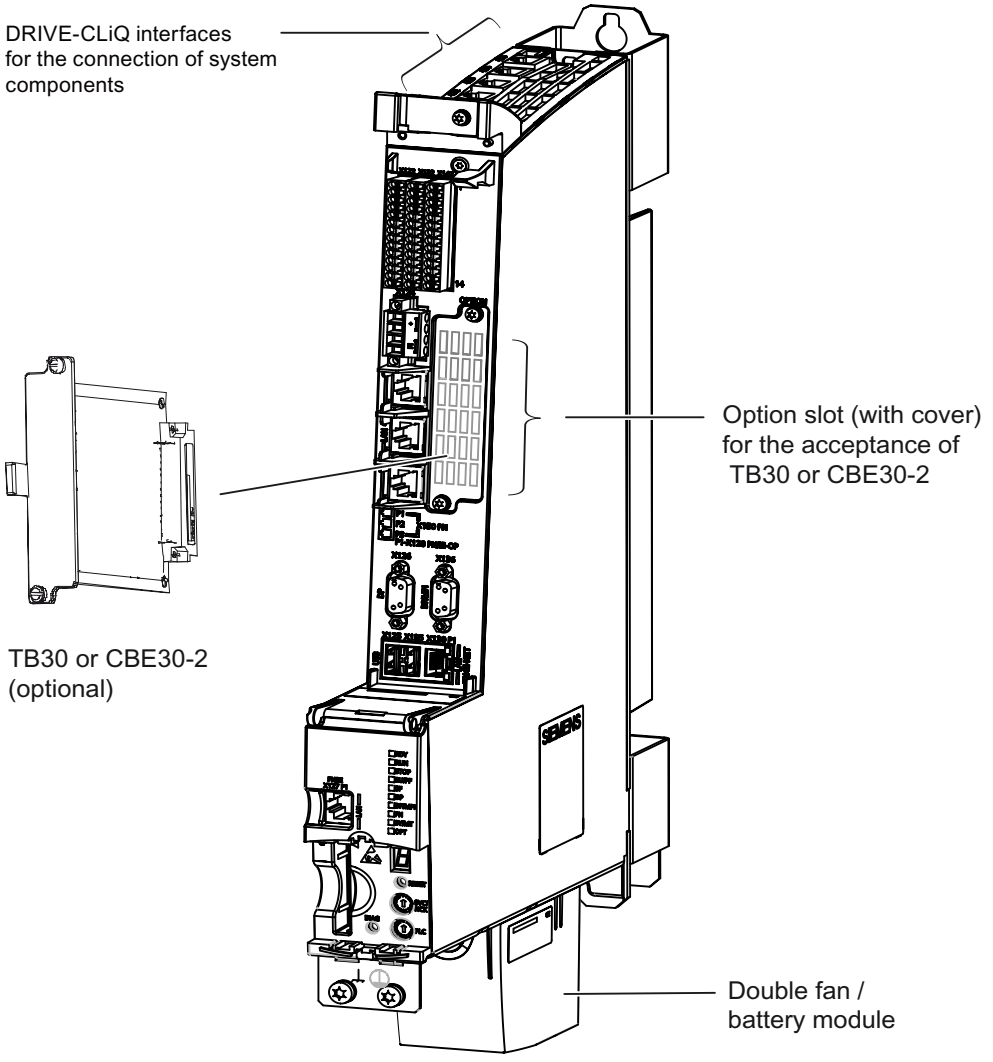


Figure 10-255 Connection of supplementary system components on the D4x5-2

Table 10-189 Applicable option modules

Module	Article number	D4x5-2 DP	D4x5-2 DP/PN
TB30	6SL3055-0AA00-2TA0	Yes	Yes
CBE30-2	6FC5312-0FA00-2AA0	No	Yes (as of V4.3)
CBE30	6FC5312-0FA00-0AA0	No	No

Fan/battery module

Cooling the SIMOTION D4x5-2 and backing up the real-time clock

Functions of a fan/battery module

The fan/battery module has the following tasks:

- CPU cooling
- Backing up the real-time clock if the SuperCap is insufficient.

The Control Unit monitors the temperature and the functioning of the fan.

Cooling the SIMOTION D4x5-2

A fan/battery module is always required for cooling the SIMOTION D4x5-2 Control Unit.

Table 10-190 Fan/battery module for SIMOTION D4x5-2

Property	SIMOTION D425-2 DP SIMOTION D435-2 DP/PN	SIMOTION D435-2 DP SIMOTION D435-2 DP/PN	SIMOTION D445-2 DP/PN SIMOTION D455-2 DP/PN
Fan/battery module	Always required (double fan/battery module included in the scope of supply)		
Usable fan/battery modules	Double fan / battery module, 6FC5348-0AA02-0AA0		
Max. permissible supply air temperature	55° C		
Fan control	Temperature-controlled fan unit will be switched on depending on supply air temperature and CPU load		

The double fan/battery module guarantees sufficient cooling even with just one functional fan.

Note

Use of the fan/battery module **with one fan** (Article No. 6FC5348-0AA01-0AA0) on the D4x5-2 is **not** possible.

An entry is made in the diagnostic buffer when a fan fault occurs (failure of one or both fans in the double fan/battery module). The fan failure is also signaled by the generation of an event in the PeripheralFaultTask and via a system variable.

If only one of the fans fails, the remaining fan continues under full load.

If both fans fail or with overtemperature, the controller switches to the RESET state, whereby the SF LED flashes red/yellow (2 Hz) and the 7-segment display shows the state "8".

Fan faults are detected if through

- A cyclic fan test
- Or when the fan is switched on

a malfunction is detected (fan does not turn or fan turns at too low a speed).

Cooling clearance

NOTICE
Higher operating temperature if ventilation clearances are too small
The ventilation clearances of 80 mm above and below the components must be observed.
The unit protects itself from overheating by shutting down.
The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

Buffering data

For the retentive storage of process variables, the SIMOTION D4x5-2 has an NVRAM memory that permanently backs up the data against a power failure.
The real-time clock is backed up by a SuperCap and continues to run when there is a power failure. This backup is for at least 4 days.
If this backup time is not sufficient, the real-time clock can be backed up by a battery that is inserted in the fan/battery module. The battery is already included in the scope of delivery of fan/battery modules.

Note

The backup time when a battery is used is at least 3 years. For the replacement part case, you should back up the NVRAM data additionally on the CF card via the application ("_savePersistentMemoryData").

Battery

A 3 V lithium battery can be inserted in the fan/battery module. The battery is pre-assembled with an approximately 4 cm long cable with plug connector. The matching mating connector is attached to a small printed circuit board for connection in the fan/battery module.

See also

Available spare parts and accessories (Page 7702)

Replace battery in the fan/battery module

Installing the fan/battery module

Overview

The procedure for installing the double fan/battery module is described below.

Procedure

There are cutouts on the lower side of the control unit to attach the module. Proceed as follows to install the double fan/battery module:

1. If required, insert a battery in the double fan/battery module.
2. Hold the double fan/battery module at an angle to the front with the open side facing up. The battery must be visible.
3. Push the plastic lug into the slot-like cutout on the lower side of the control unit.
4. Tilt the double fan/battery module up until the two latches snap into place at the front. Note the two contact strips that are lead through cutouts of the control unit.
   This establishes the electrical connection between the double fan/battery module and the control unit.

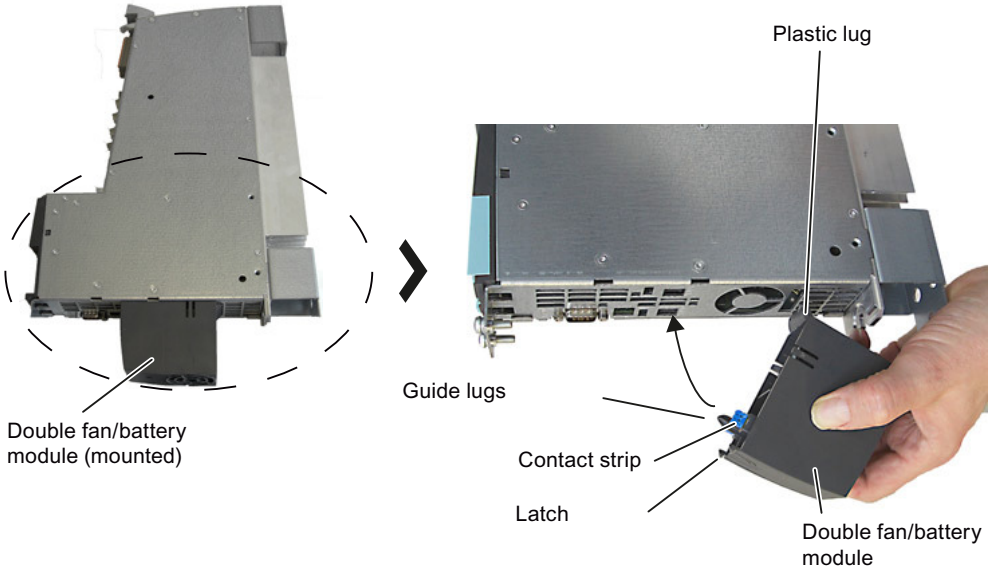


Figure 10-256 Installing the double fan/battery module

TB30 terminal board

Description

The TB30 terminal board is a terminal expansion module for SIMOTION D4x5-2. The module is inserted in the option slot of the D4x5-2 control unit.

Table 10-191 Interface overview of the TB30

Type	Quantity
Digital inputs	4
Digital outputs	4
Analog inputs	2
Analog outputs	2

Safety information for the TB30

NOTICE
Damage to the TB30 caused by electric fields or electrostatic discharge
Option boards are ESD-sensitive components.
De-energize the SIMOTION D4x5-2 device before inserting or removing the option board. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Observe ESD guidelines; for details, see Section ESD guidelines (Page 7707).

Interfaces

Overview

The following figure shows the arrangement of the interfaces on the front of the TB30.

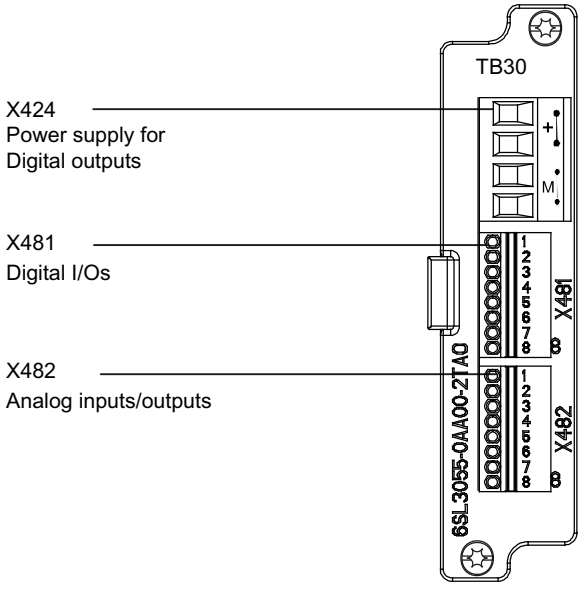


Figure 10-257 Interface arrangement on the TB30

Connection diagram

The following figure shows the schematic diagram of the TB30 as well as its connections for inputs (DI, AI), outputs (DO, AO) and power supply.

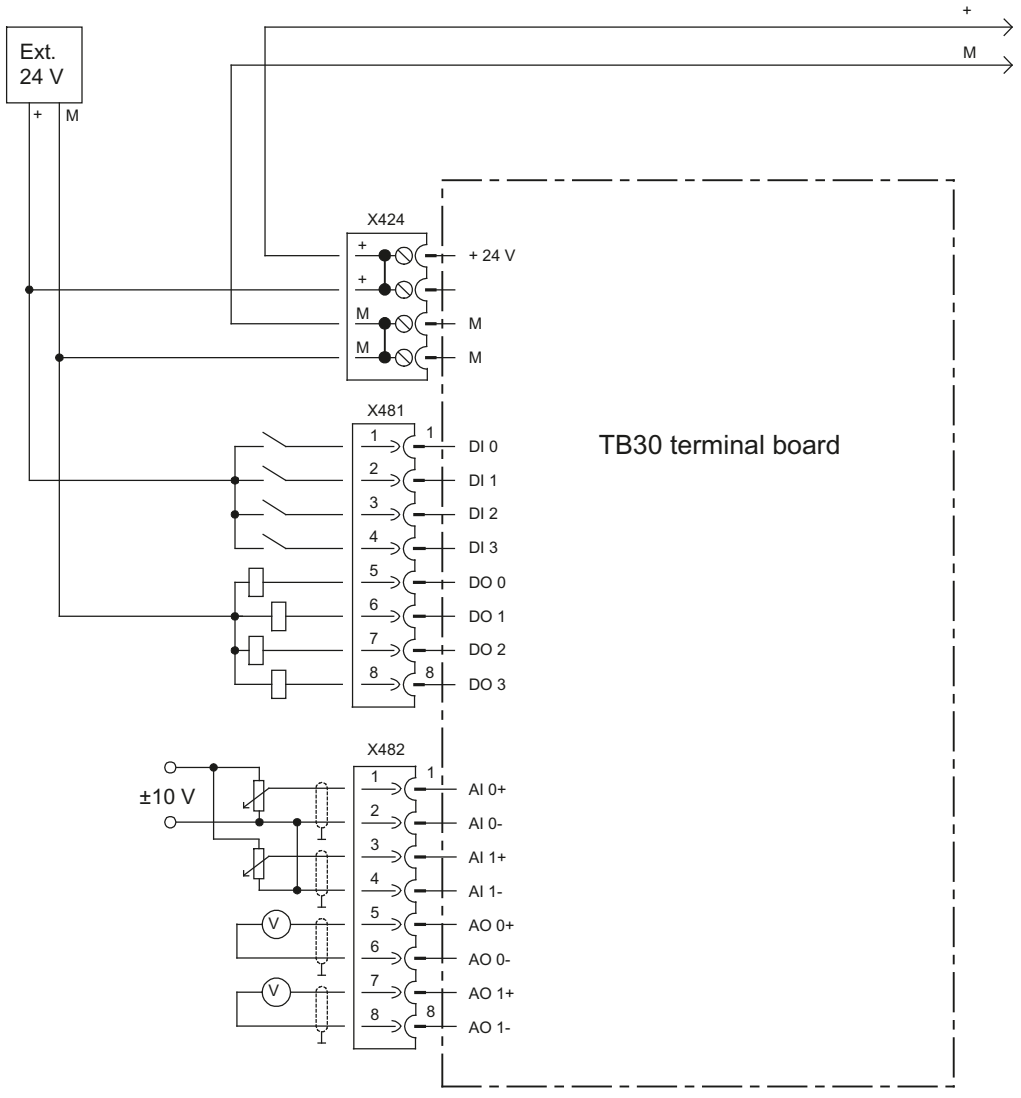


Figure 10-258 TB30 connection diagram

Power supply of digital outputs

Table 10-192 Terminal block X424

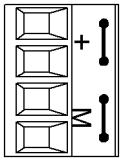
	Terminal	Function	Technical specifications
	+	Power supply	Max. current via bridges in the connector: 20 A
	+	Power supply	
	G	Ground	
	G	Ground	

Table 10-193 X424 wiring

Features		Type
Connector type		4-way screw-type terminal
Number of cables that can be connected		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 2.5 mm²
	Flexible	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.2 mm² ... 1.5 mm²
	AWG / kcmil	22 ... 12
Stripped length		6 ... 7 mm
Tool		Screwdriver 0.5 x 3 mm (M2.5)
Tightening torque		0.4 to 0.5 Nm (3.5 ... 4.4 lbf in)
Max. current carrying capacity, incl. loop-through		20 A (15 A per UL/CSA)
Max. cable length		10 m

Features

Two "+" terminals and two "M" terminals are available. These are jumpered in the connector. This loops through the power supply.

This power supply is only required for the digital outputs.

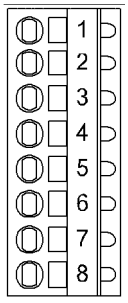
The option slot of the D4x5-2 control unit also provides the following power supplies:

- Electronic power supply of the TB30
- Supply of the analog inputs and outputs.

The power supply of the digital outputs and the electronic power supply of the Control Unit are isolated.

Digital I/Os

Table 10-194 Terminal block X481

	Terminal	Designation[1)]
	1	DI 0
	2	DI 1
	3	DI 2
	4	DI 3
	5	DO 0
	6	DO 1
	7	DO 2
	8	DO 3

1) DI: digital input, DO: Digital output

Table 10-195 X481 wiring

Characteristics	Type
Connectable cable types: - Rigid - Flexible - Flexible, with end sleeve without plastic sleeve - AWG/kcmil	Conductor cross-sections: 0.14 mm^2 to 0.5 mm^2 0.14 mm^2 to 0.5 mm^2 0.25 mm^2 to 0.5 mm^2 26 to 20
Number of cables that can be connected	1
Stripped length	8 to 9 mm
Tool	Screwdriver 0.4 x 2.0 mm
Max. cable length	30 m

Note

An open input is interpreted as "low".

The power supply and the digital I/Os are isolated from the control unit.

Note

With momentary interruptions in the 24 V supply, the digital outputs are deactivated during this time.

Analog inputs and outputs

Table 10-196 Terminal block X482

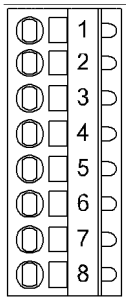
	Terminal	Designation[1)]
	1	AI 0+
	2	AI 0-
	3	AI 1+
	4	AI 1-
	5	AO 0+
	6	AO 0-
	7	AO 1+
	8	AO 1-

Table 10-197 X482 wiring

Features	Type
Connectable conductor types - Rigid - Flexible - Flexible with end sleeve, without plastic sleeve - AWG/kcmil	Conductor cross-section 0.14 mm^2 to 0.5 mm^2 0.14 mm^2 to 0.5 mm^2 0.25 mm^2 to 0.5 mm^2 26 to 20
Number of cables that can be connected	1
Stripped length	8 to 9 mm
Tool	Screwdriver 0.4 x 2.0 mm
Max. cable length	30 m

Note

An open input is interpreted as approximately "0 V."

The power supply of the analog I/Os of the TB30 is via the option slot of the D4x5-2 control unit and not via X424.

The shield is connected to the Control Unit. For more information on "Establishing a shield connection", see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Chapter "Connecting I/Os."

NOTICE
Incorrect results of analog-to-digital conversion due to impermissible input voltage
The common mode range must not be violated.
Makes sure that the analog input voltage signals can have a maximum voltage of ±30 V with respect to the reference potential. If the range is infringed, incorrect results may occur during analog/digital conversion.

Working with analog inputs

The following reference contains more information about analog inputs:

- /IH1/ *SINAMICS S120* Commissioning Manual.
- /GH1/ *SINAMICS S120* Control Units and Additional System Components Manual

Commissioning

Information about commissioning can be found in the *SIMOTION D4x5-2*Commissioning and Hardware Installation Manual.

Technical Specifications

Power supply of digital outputs

Table 10-198 Terminal block X424

Characteristic	Value/Range
Power supply	Voltage: 24 VDC (20.4 V – 28.8 V) Current via the option slot of the D4x5-2 (without digital outputs): 0.05 A Max. power consumption: 4 A Max. power consumption per digital output: 0.5 A Power loss: < 3 W
Response time	The response time of digital inputs/outputs and analog inputs/outputs depends on the evaluation on the Control Unit (see function diagram). **References:** SINAMICS S120/S150 List Manual, Section "Function diagrams".

Digital I/Os

Table 10-199 Terminal block X481

Characteristic	Value/Range
Digital inputs	Voltage: - 3 V to 30 V Current input (typical): 10 mA at 24 VDC Ground reference: X424 (G terminal) Input delay: - L ⇒ H: Approx. 20 µs - H ⇒ L: Approx. 100 µs Level (including ripple): - High level: 15 V to 30 V - Low level: -3 V to 5 V
Digital outputs	Voltage: 24 VDC Max. load current per output: 500 mA Ground reference: X424 (M terminal) Continuous short-circuit proof Output delay: - L ⇒ H: Typically 150 µs at 0.5 A ohmic load (500 µs max.) - H ⇒ L: Typically 50 µs at 0.5 A ohmic load Switching frequency: - for resistive load: Max. 100 Hz - for inductive load: Max. 0.5 Hz - for lamp load: Max. 10 Hz Maximum lamp load: 5 W

Analog I/Os

Table 10-200 Terminal block X482

Characteristic	Value/Range
Analog inputs (AI):	Voltage: - -10 V to +10 V Internal resistance: 65 kΩ Resolution: 13 bits + sign
Analog outputs (AO):	Voltage range: -10 V to +10 V Load current: max. -3 mA to +3 mA Resolution: 11 bits + sign Sustained short-circuit strength

Dimensions and weight

Characteristic	Value
Dimensions (H x W x D), approx.	25 x 95 x 143 mm
Weight [g]	• Without packaging approx. 100 g • With packaging approx. 240 g

CBE30-2 Ethernet communication board

Overview

Properties of the CBE30-2

A second PROFINET interface can be implemented for the SIMOTION D4x5-2 DP/PN with the CBE30-2 Ethernet communication board.

The CBE30-2 cannot be used with the SIMOTION D4x5-2 DP.

The CBE30-2 offers the following functions:

- PROFINET IO controller, I device (also controller and device simultaneously)
- 100 Mbps full duplex / autocrossing
- Supports real-time classes of PROFINET IO:
  - RT (real-time)
  - IRT (isochronous real-time)

The CBE30-2 has an X1400 interface with an integral 4-port switch based on PROFINET ASICs ERTEC400.

View

The connections and LED displays are located on the front of the CBE30-2.

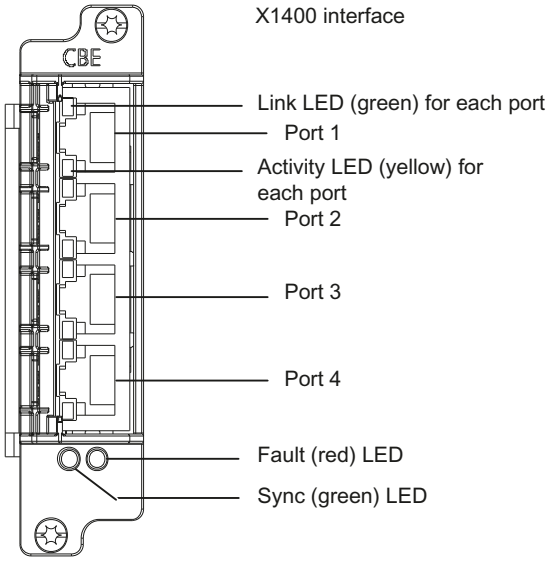


Figure 10-259 CBE30-2 front view

Type plate

Type plate

The figure below contains all the information included on the type plate.

Figure 10-260 CBE30-2 type plate

You might need to access the information provided on the type plate after mounting. Because the type plate is located on the under side of the CBE30-2, we recommend that you note the serial number before installing it.

Note

The contents of the individual type plate fields of the CBE30-2 may differ from those described in this manual (e.g. updated product status, approvals and markings not yet issued, etc.). KCC (Korea Communications Commission) for CBE30-2 is marked with a separately attached plate.

MAC address

A second plate for the MAC address of the PROFINET interfaces is attached to the top side of the board:

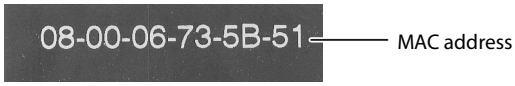


Figure 10-261 CBE30-2 MAC address

This plate is only visible when the CBE30-2 has been removed. You need the MAC address to assign an IP address.

Note

The MAC address is affixed to the top side of the CBE30-2, behind the RJ45 sockets.

Safety information

NOTICE
CBE30-2 damage caused by electric fields or electrostatic discharge
Option boards are ESD-sensitive components.
De-energize the SIMOTION D4x5-2 device before inserting or removing the option board. The SIMOTION D4x5-2 is in a de-energized state when all the LEDs are off.
Observe the ESD guidelines (see Section ESD guidelines (Page 7707)).

Interface description

Properties

The X1400 interface has full-duplex 10/100 Mbit Ethernet ports. The module has an integrated 4-port switch.

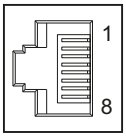
Interface characteristics

Table 10-201 X1400 features

Characteristic	Type
Connector type	RJ45 socket
Cable type	Industrial Ethernet cable
Maximum cable length	100 m
Dust protection filler plugs for sealing unused PROFINET ports	Five filler plugs contained in the D4x5-2 scope of delivery Filler plugs (50 pcs) article number: 6SL3066-4CA00-0AA0

Interface assignment

Table 10-202 X1400 interface

Representation	Pin	Name	Signal type	Description
	1	TXP	Output	Ethernet transmit differential signal
	2	TXN	Output	Ethernet transmit differential signal
	3	RXP	Input	Ethernet receive differential signal
	4	--		4 together with 5 via 75 ohm at the 1 nF capacitor to the shield ground
	5	--		4 together with 5 via 75 ohm at the 1 nF capacitor to the shield ground
	6	RXN	Input	Ethernet receive differential signal
	7	--		7 together with 8 via 75 ohm at the 1 nF capacitor to the shield ground
	8	--		7 together with 8 via 75 ohm at the 1 nF capacitor to the shield ground
	Screened backshell	M_EXT		Screen, permanently connected

Position of the ports

The interfaces are located on the front side of the CBE30-2.

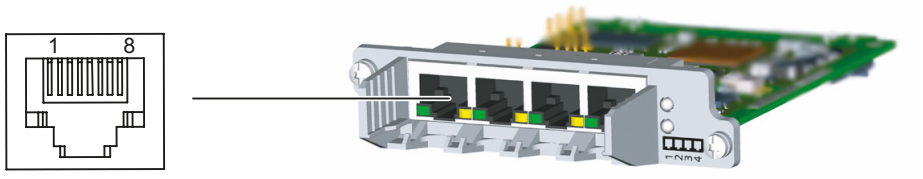


Figure 10-262 CBE30-2 interface

LED displays

Position of the LEDs

The X1400 interface with the four ports has integrated LEDs for displaying the link and the activity for each port. The front panel of the board is also fitted with two LEDs (Fault and Sync), which indicate the bus status.

Table 10-203 Meaning of the LED displays

LED	Meaning
link	... indicates whether a different device is connected to port x and a physical connection exists
Activity	... indicates whether data is being received or sent at port x
Sync	... indicates the synchronization status of the PROFINET IO interface
Fault	... indicates a fault state of the PROFINET IO interface

Additional references

Detailed information on the states of the status LEDs can be found in the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Section *Diagnostics*.

Dimension drawing

CBE30-2 representation

The following illustrations show the components in four views.

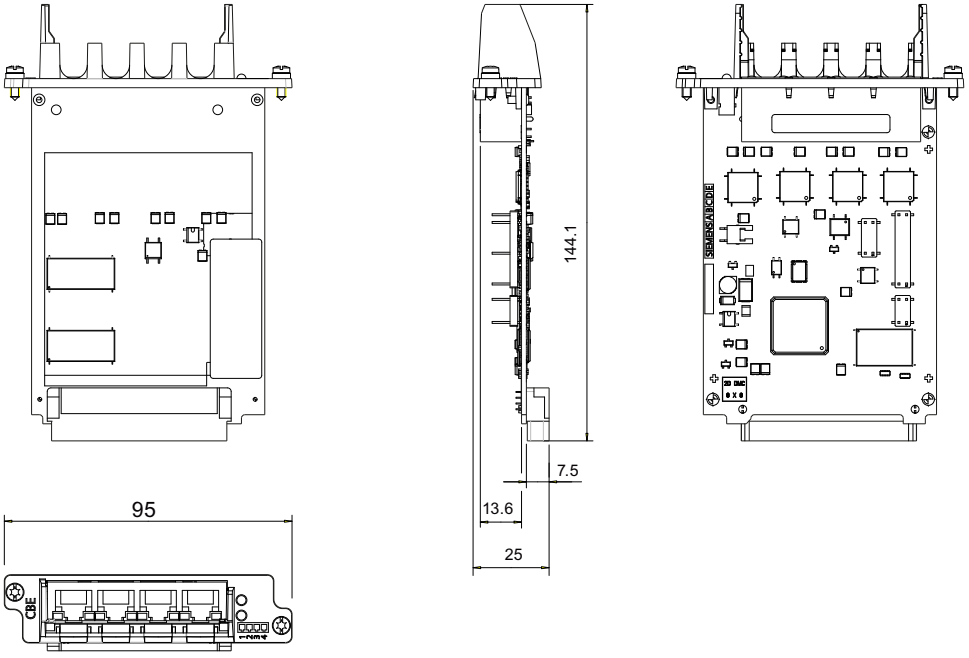


Figure 10-263 CBE30-2 dimension drawing (dimensions in mm)

Commissioning

Additional references

- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual
- *SIMOTION Communication* System Manual

Technical data

CBE30-2 technical data

Table 10-204 CBE30-2 technical data

Characteristic	Value
Current requirement at 24 VDC	0.25 A

Storage and operation

Table 10-205 Ambient conditions for CBE30-2

Characteristic	Value/Range	Standard
Permissible ambient temperature • Storage • Operation	 -40° C... +70° C 0° C ... +55° C	
Permissible relative humidity (without condensation)	> 5% to 95%	DIN EN 60721-3-3, Class 3K5
Degree of protection	IP00	DIN EN 60529
Vibratory load		DIN EN 60721-3-3, Class 3M6
Shock load		DIN EN 60721-3-3, Class 3M4
Free fall		DIN EN 60721-3-2, Class 2M1 and 2M2
Toppling		DIN EN 60721-3-2, Class 2M1

Dimensions and weight

Characteristic	Value
Dimensions (H x W x D)	25 x 95 x 143 mm
Weight [g]	• Without packaging approx. 100 g • With packaging approx. 240 g

Certificates, approvals, declarations of conformity

You can find an overview of the certifications available for the SIMOTION CBE30-2 in Appendix A (Page 7705).

You can also find further information on the Internet at:

https://support.industry.siemens.com/cs/ww/en/ps/14513/cert

CX32-2 controller extension

Overview of CX32-2

Properties

The CX32-2 (Article No. 6AU1432-0AA00-0AA0) is a module in the SINAMICS S120 booksize format. The CX32-2 allows scaling for the drive-end computing performance of the SIMOTION D4x5-2 Control Units.

Each CX32-2 can control up to 6 additional servo, 6 vector or 12 *V/f* axes.
The Controller Extension has 6 DI, 4 DI/DO, and 4 DRIVE-CLiQ interfaces.

Note

The CX32 (Article No. 6SL3040-0NA00-0AA0) cannot be used with the D4x5-2. If an incorrect Controller Extension is used, a topology error will be signaled (F01360 Topology: Actual topology not permitted).

Drive quantity structure

Table 10-206 Drive quantity structure

Characteristic	Quantity structure
Number of CX32-2	D425-2: Max. 3 CX32-2 D435-2/D445-2/D455-2: Max. 5 CX32-2
Maximum number of drives on the SINAMICS Integrated with connected CX32-2	• 6 servo or • 6 vector or • 12 *V/f* incl. an infeed (ALM, BLM, SLM)
Maximum number of drives per CX32-2	• 6 servo or • 6 vector or • 12 *V/f* incl. an infeed (ALM, BLM, SLM)

The maximum quantity structures can be reduced depending on the configuration (for example, for connected Terminal Modules).

Note

In principle, a fourth CX32-2 can be connected to the SIMOTION D425-2 and a sixth CX32-2 connected to the SIMOTION D435-2/D445-2/D455-2.

Note however that no further drives can then be connected on the SINAMICS Integrated of the D4x5-2. Possible fields of application, for example, are modular machine concepts with a central controller.

Note

Mixed operation of servo and vector-controlled drives

Mixed operation of servo and vector-controlled drives is not possible on a CX32-2. Therefore, drives on a CX32-2 must be operated either in servo or in vector mode only. The following mixed operation is possible on a CX32-2:

- Servo and *V/f*-controlled drives
- Vector and *V/f*-controlled drives

This corresponds to the possible mixed operation on the SIMOTION D4x5-2.

V/f-controlled drives

A maximum of 12 *V/f*-controlled drives are supported by each CX32-2.

SIZER

For a detailed estimation of the drive quantity structures, we recommend that you use the SIZER configuration tool.

With SIZER, you can easily configure the SINAMICS S120 drive family including SIMOTION. It provides you with support for selecting and dimensioning the components required for a Motion Control task.

You can also determine the possible number of axes and the resulting load with SIZER in accordance with your performance requirements.

Interfaces

Overview of interfaces

Position of the interfaces

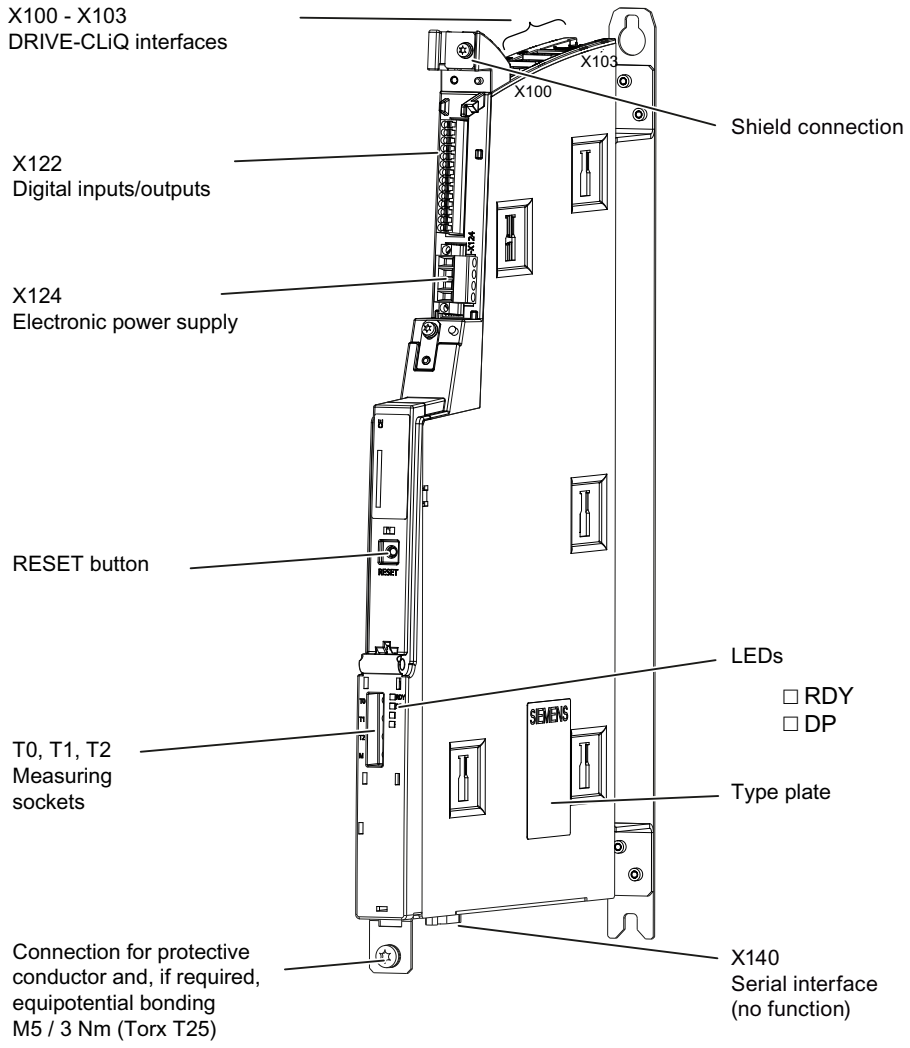


Figure 10-264 CX32-2 (without cover) with interfaces and operator control

NOTICE
Higher operating temperature if ventilation clearances are too small The 80 mm clearances above and below the components must be observed. The unit protects itself from overheating by shutting down. The ventilation clearance is measured from the lower edge of the module, i.e. the fan/battery module is not included in the dimension.

List of interfaces

The CX32-2 has the following interfaces:

- 4 DRIVE-CLiQ interfaces
- 4 digital inputs/outputs
- 6 digital inputs
- Power supply connector

Available interfaces

Table 10-207 Overview of available interfaces

Interface	Designation	Connector type
DRIVE-CLiQ interface	X100	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X101	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X102	DRIVE-CLiQ socket
DRIVE-CLiQ interface	X103	DRIVE-CLiQ socket
Digital inputs/outputs	X122	Mini Combicon, 3.5 mm, 1x14-pin
Power supply connector	X124	Combicon, 4-pin
Measuring sockets (T0, T1, T2, and M)	X131 - X134	Sockets

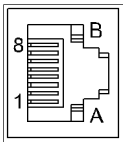
Non-usable interfaces

Table 10-208 Overview of interfaces that cannot be used for the CX32-2

Interface name	Interface	Connector type
RS232 interface	X140	9-pin Sub-D connector

DRIVE-CLiQ interface

Table 10-209 DRIVE-CLiQ interface X100 – X103

	Pin	Signal name	Technical specifications
	1	TXP	Transmit data +
	2	TXN	Transmit data -
	3	RXP	Receive data +
	4	Reserved, do not use	
	5	Reserved, do not use	
	6	RXN	Receive data -
	7	Reserved, do not use	
	8	Reserved, do not use	
	A	+ (24 V)	Power supply
	B	M (0 V)	Electronic ground
Dust protection filler plugs for sealing unused DRIVE-CLiQ ports: • Three filler plugs contained in the CX32-2 scope of delivery • Filler plugs (50 pcs) article number: 6SL3066-4CA00-0AA0			

Digital I/Os (X122)

Interface characteristics

Sensors and actuators can be connected to the X122 connector via digital inputs and outputs.

Table 10-210 X122 wiring

Features		Type
Connector type		14-way spring-loaded terminal
Number of cables that can be connected		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 1.5 mm²
	Flexible	0.2 mm² ... 1.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.25 mm² ... 1.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.25 mm² ... 0.75 mm²
	AWG / kcmil	24 ... 16
Stripped length		10 mm
Tool		Screwdriver 0.4 x 2.0 mm
Max. cable length		30 m
Max. current carrying capacity (ground)		6 A

Position of the connector

The X122 connection is on the front side of the CX32-2 at the top, see appropriate figure in Section Overview of interfaces (Page 7685).

Connection and circuit diagram

The following figure shows the schematic diagram and the connection of the digital I/Os on the CX32-2 and the associated external power supply.

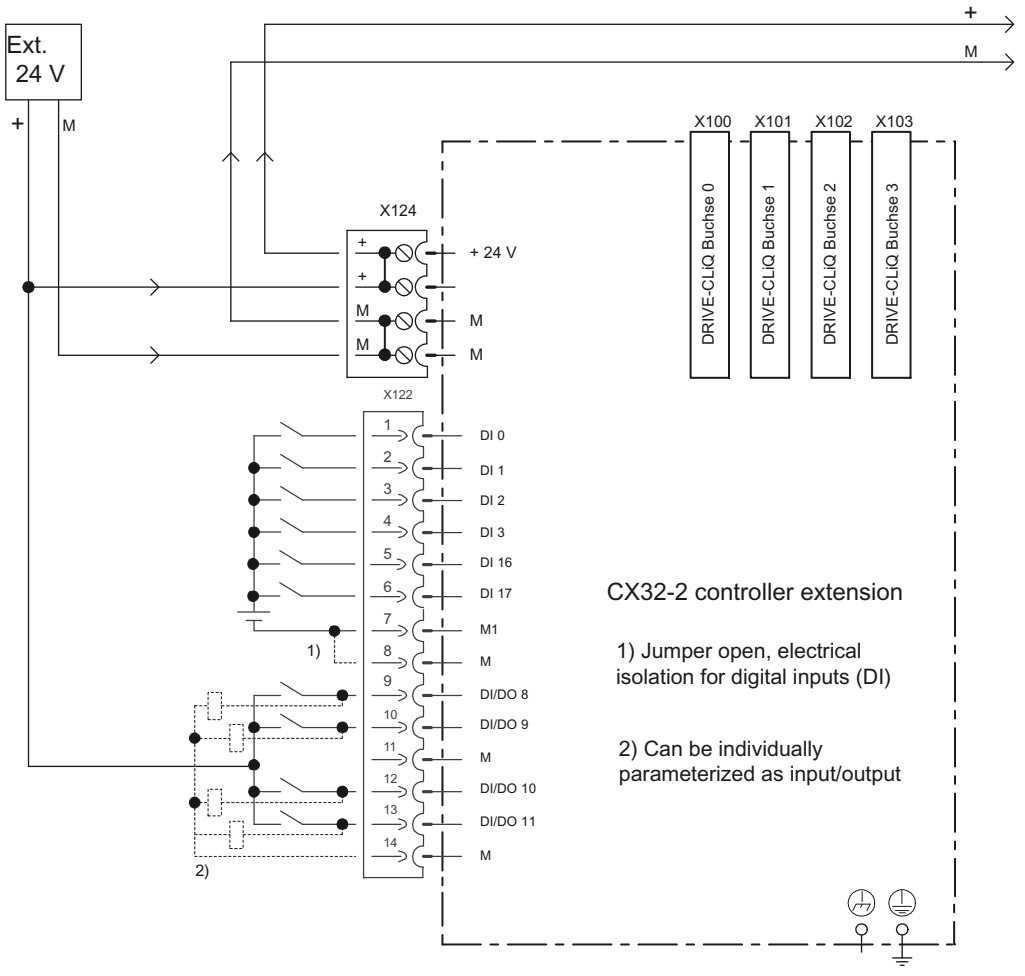


Figure 10-265 Digital I/Os connection diagram

Interface assignment of X122

Table 10-211 Digital inputs/outputs X122

Pin	Designation [1]	Signal type [2]	Notes
1	DI 0	I	Digital input 0
2	DI 1	I	Digital input 1
3	DI 2	I	Digital input 2
4	DI 3	I	Digital input 3
5	DI 16	I	Digital input 16
6	DI 17	I	Digital input 17
7	M1	GND	Ground for DI 0 – DI 3, DI 16, DI 17 (electrically isolated relative to G)
8	G	GND	Ground
9	DI/DO 8	B	Digital input/output 8 (can also be used as a measuring input input or as an input for the external zero mark)
10	DI/DO 9	B	Digital input/output 9 (can also be used as a measuring input input or as an input for the external zero mark)
11	G	GND	Ground
12	DI/DO 10	B	Digital input/output 10 (can also be used as a measuring input input or as an input for the external zero mark)
13	DI/DO 11	B	Digital input/output 11 (can also be used as a measuring input input or as an input for the external zero mark)
14	G	GND	Ground

[1] DI: Digital input; DI/DO: Bidirectional digital input/output; M: Electronics ground; M1: Ground reference

[2] B = Bidirectional; I = Input; GND = Reference potential (ground)

Note

An open input is interpreted as "Low".
Terminal G1 must be connected for the digital inputs to function. The following alternatives are available:

- Connect the incorporated ground reference of the digital input to M1
- Create the bridge between terminal M and terminal M1.
  This removes the electrical isolation for these digital inputs.

Using the digital inputs/outputs

Connecting sensors and actuators

Digital inputs and digital outputs can be used to connect various sensors and actuators to the 14-pin X122 front connector.

The following types of digital inputs/outputs are used:

- Digital inputs (DI)
- Bidirectional digital inputs/outputs (DI/DO)

Bidirectional digital I/Os can be configured individually as digital inputs or outputs.

Assignment of the I/Os to functions can be parameterized as required. Special functions (e.g. input of the measuring input) can be assigned to the I/Os.

The digital inputs/outputs on the X122 front connector can be used by either SIMOTION or SINAMICS (e.g. as enable signal for a drive).

Table 10-212 Use of the digital inputs/outputs

	DI 0-3, DI 16, DI 17 (X122)	**DI/DO 8-11 (X122)**
Galvanic isolation	Electrically isolated (ground reference M1)	Non-isolated (ground reference M)
Use as:		
• Freely addressable I/Os for SIMOTION	yes	yes
• I/Os that are assigned to the drive	yes	yes
• Measuring inputs	no	Yes (global and local measuring inputs)
• Inputs for the external zero mark	no	yes
• Cam outputs	no	no
Configuration:		
Assignment	Can be configured channel-by-channel on the drive	Can be configured channel-by-channel on the drive

Note

For optimal noise immunity of the digital inputs, the use of shielded cables is necessary in certain cases. This is necessary when the digital inputs are to be used as

- Inputs of measuring inputs or
- Inputs for the external zero mark

Additional references

For information on configuring the I/Os as freely addressable I/Os or as measuring inputs, see the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

For information on the configuration and function of the measuring input, output cam, and cam track technology objects, see the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

Power supply

Application

This interface is provided for the connection of an external power supply.

Note

When using external power supplies (e.g. SITOP), the ground potential must be connected with the protective ground terminal (PELV).

Note

Ground potential and housing (PE) are connected internally with low impedance.

Features of the interface

Table 10-213 Interface X124

Features		Type
Connector type		4-way screw-type terminal
Number of cables that can be connected		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 2.5 mm²
	Flexible	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.2 mm² ... 1.5 mm²
	AWG / kcmil	22 ... 12
Stripped length		6 ... 7 mm
Tool		Screwdriver 0.5 x 3 mm (M2.5)
Tightening torque		0.4 to 0.5 Nm (3.5 ... 4.4 lbf in)
Max. current carrying capacity, incl. loop-through		20 A (15 A per UL/CSA)
Max. cable length		10 m

Interface assignments

Table 10-214 Power supply X124

Pin	Signal name	Meaning
1	P24	Power supply 24 V
2	P24	Power supply 24 V
3	G	Ground
4	G	Ground

Note

The 24 V supply voltage is looped through via the 24 V connector. In this case, pin 1 is jumpered with pin 2, and pin 3 is jumpered with pin 4 in the connector. The maximum current can be limited through the current carrying capacity of the cable. The current carrying capacity of the cable depends, for example, on the type of cable installation (cable duct, laying on a cable rack, etc.)

Note

The power supply terminal strip must be screwed on tightly using a flat-bladed screwdriver.

Measuring sockets

Application

The T0, T1 and T2 measuring sockets are used to output analog signals. Any signal interconnectable via SINAMICS can be output on any measuring socket of the CX32-2.

Note

The measuring sockets should be used exclusively for servicing purposes.

The measurements may only be performed by appropriately trained specialists.

The measuring sockets are suited for multiple-spring wire connectors with a diameter of 2 mm.

Displays of the LEDs

Description

Table 10-215 CX32-2 LEDs

LED	Description
RDY	Operating modes of the CX32-2
DP	Status of the communication connection between the D4x5-2 and the CX32-2

Additional references

Detailed information on the states of the status LEDs can be found in the *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual, Section *Diagnostics*.

Cause and rectification of faults

The following reference contains information about the cause of faults and how they can be rectified:

- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

RESET button

The RESET button is on the front of the device under the cover.

Function of the RESET button

The following reference contains information about the RESET button function:

- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Commissioning

Information on the commissioning in the following reference:

SIMOTION D4x5-2 Commissioning and Hardware Installation Manual.

Technical data of the CX32-2

Memory for system data

Table 10-216 Diagnostic buffer

Data	SIMOTION CX32-2
Diagnostic buffer (non-volatile)	200 messages

The data for the CX32-2 is stored on the SIMOTION D4x5-2, which means no action has to be taken on the CX32-2 when the module is replaced.

Dimensions and weights

Table 10-217 Dimensions and weight of a SIMOTION CX32-2

Parameter	SIMOTION CX32-2
Dimensions W x H x D [mm] (max. expansion)	
• Without fastening using spacers	• 25 x 380 x 230
• With fastening using spacers	• 25 x 380 x 270
Weight CX32-2 [g]	
• Without packaging	• approx. 2600
• With packaging	• approx. 3150

Ambient conditions

The following conditions apply to modules that are shipped and stored in the original packaging.

Table 10-218 CX32-2 environmental requirements

Parameter	Values
Permissible ambient temperature	
• Transport • Long-term storage • Operation	• -40° C ... +70° C • -25° C ... +55° C • 0° C ... +55° C up to 2000 m above sea level. As of an altitude of 2000 m, the maximum ambient temperature decreases by 7° C every 1000 m increase in altitude
Atmospheric pressure	620 ... 1060 hPa
Permissible relative humidity	
• During transport and storage • During operation (condensation, icing, drip, spray and splash water not permitted)	• 10% ... 100% • 5% ... 90%
Installation altitude	Max. 4000 m above sea level. For SINAMICS S120 drive components, see SINAMICS Manuals.
Biological environmental conditions • Storage • Transport • Operation	 • Class 1B1 according to EN 60 721-3-1 • Class 2B1 according to EN 60 721-3-2 • Class 3B1 according to EN 60 721-3-3
Degree of protection according to EN 60529 (IEC 60529)	IP20
Pollution degree	2 according to EN 60 664-1

Integrated drive control

Table 10-219 Controls for integrated drives

Data	SIMOTION CX32-2
Max. number of axes for integrated drive control (servo/vector/*V/f*)	6 / 6 / 12 (alternative) Drive control based on SINAMICS S120 CU320-2, firmware version V4.x/V5.x

Communication

Table 10-220 Interface communication

Data	SIMOTION CX32-2
DRIVE-CLiQ interfaces	4

General technical data

Table 10-221 Technical data (general)

Data	SIMOTION CX32-2
Power supply • Rated value • Permissible range	 24 VDC (20.4 ... 28.8 V)
Current consumption, typically [1]	300 mA
Starting current, typical	1.6 A
Power loss, typical	7 W
Power loss, max.	14 W

[1] With no load on inputs/outputs, no 24-V supply via DRIVE-CLiQ interface

Digital inputs

Table 10-222 Digital inputs on SIMOTION CX32-2

Data	SIMOTION CX32-2
Digital inputs	6
• Rated value	24 VDC
• For signal "1"	15 ... 30 V
• For signal "0" [2]	-3 ... +5 V
Galvanic isolation	Yes, in groups of 6 [1]
Current consumption typ. at High level	3.5 mA at 24 V 9 mA at 24 V [3]
Input delay, typical (hardware)	Signal "0" → "1": 50 µs Signal "1" → "0": 150 µs

[1] The reference potential is terminal M1

[2] The digital inputs are protected against polarity reversal up to -30 V

[3] Up to and including HW version "D"

Digital inputs/outputs (parameterizable)

Table 10-223 Digital inputs/outputs on SIMOTION CX32-2

Data	SIMOTION CX32-2
Number of digital inputs/outputs	4 • Max. 4 as measuring input inputs • Max. 0 as output cam outputs
If used as an input:	
• Input voltage, rated value	24 VDC
• Input voltage, for signal "1"	15 ... 30 V
• Input voltage, for signal "0" [2]	-3 ... +5 V

Data	SIMOTION CX32-2
Galvanic isolation	No
Current consumption typ. at signal level "1"	3.5 mA at 24 V 9 mA at 24 V [3]
Input delay, typical (hardware)	Signal "0" → "1": 5 µs Signal "1" → "0": 50 µs
Measuring input input, resolution	1 µs
Measuring input input, reproducibility	5 µs
If used as an output	
• Rated load voltage, permissible range	24 VDC, 20.4 ... 28.8 V
• Galvanic isolation	No
• Current load, max.	500 mA per output
• Residual current, max.	2 mA
• Output delay time, typical/max. (hardware) [1]	Signal "0" → "1": 150 µs / 400 µs Signal "1" → "0": 75 µs / 100 µs
Switching frequency of the outputs, max.	
• With resistive load	4 kHz
• With inductive load	2 Hz
• With lamp load	11 Hz
Maximum lamp load	5 W
Short-circuit protection	Yes

1) Data for: Vcc = 24 V; load 48 Ohm; H = 90% V_{out}, L = 10% V_{out}

2) The digital inputs are protected against polarity reversal up to -30 V

3) Up to and including HW version "D"

Max. switching frequency of the DO

The max. switching frequency of the hardware depends on the load. For an ohmic load of 24 V / 0.5 A, it is up to 4 kHz (typical value; low-high ratio = 50:50; short cable lengths).
Logic control of the digital output is also a limiting factor.
If an X122 DO is controlled from the user program, no more than one edge is possible for servo cycle clock or CU sampling time of the inputs/outputs (cu.p0799[0]).
With a servo cycle clock of at least 500 µs, a max. switching frequency of 1 kHz is achieved.
The max. achievable switching frequency can also be limited by CU parameter p0799[0] (sampling time of the CU inputs/outputs) or p2048 (PROFIdrive PZD sampling time).

Certificates, approvals, declarations of conformity

You can find an overview of the certifications available for the SIMOTION CX32-2 in Appendix A (Page 7705).

You can also find further information on the Internet at:

https://support.industry.siemens.com/cs/ww/en/ps/14513/cert

CX32-2 dimension drawings

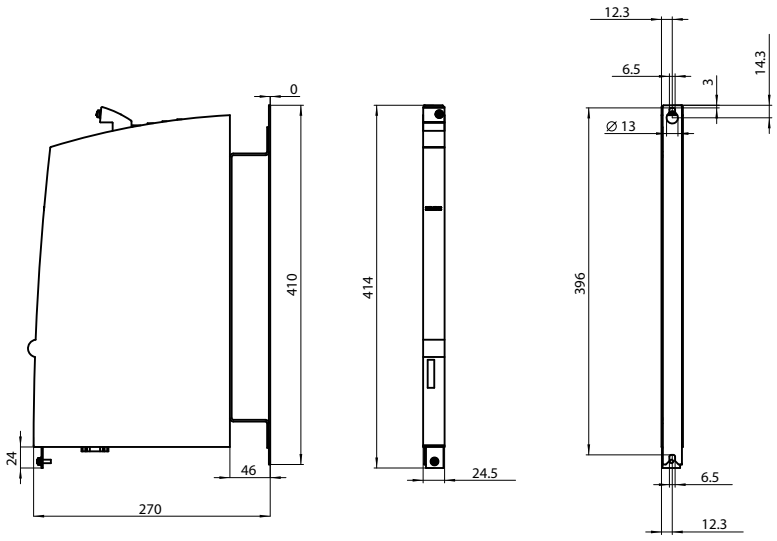


Figure 10-266 CX32-2 dimension drawings (dimensions in mm)

NOTICE
Higher operating temperature if ventilation clearances are too small
The cooling clearances of 80 mm above and below the component must be observed.

Terminal module TM31

Characteristics of the TM31

With the TM31 Terminal Module, the number of available digital inputs/digital outputs and the number of analog input/analog outputs within a drive system can be expanded. The TM31 is connected via DRIVE-CLiQ. It has 2 DRIVE-CLiQ interfaces for this.

The TM31 contains the following terminals:

Table 10-224 Interface overview

Interface	Quantity
Digital inputs	8
Bidirectional inputs/outputs	4
Relay outputs with changeover contact	2
Analog inputs	2
Analog outputs	2
Temperature sensor input (KTY84-130 or PTC)	1

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

You will find detailed information about the TM31 in the

- *SINAMICS S120 Control Units and Additional System Components* Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Terminal module TM41

Characteristics of the TM41

With the TM41 Terminal Module, the number of available digital inputs/digital outputs and the number of analog inputs within a drive system can be expanded. In addition, the TTL output can be used for encoder emulation. The TM41 is connected via DRIVE-CLiQ.

The TM41 contains the following terminals:

Table 10-225 Interface overview

Type	Quantity
Digital inputs	4
Digital inputs/outputs	4
Analog inputs	1
TTL encoder output	1

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

You will find detailed information about the TM41 Terminal Module in the

- *SINAMICS S120 Control Units and Additional System Components* Manual
- *SIMOTION D4x5-2* Commissioning and Hardware Installation Manual.

Terminal Module TM54F

Characteristics of the TM54F

The TM54F Terminal Module is a terminal expansion module for snapping on to a DIN EN 60715 mounting rail. The TM54F offers safe digital inputs and outputs for control of Safety Integrated functions of SINAMICS.

No more than one TM54F can be assigned to each drive control (SINAMICS Integrated of a D4x5-2, CX32-2, CU320-2, etc.). Connection is via DRIVE-CLiQ. Each drive control must have its own dedicated TM54F.

TM54 is equipped with the following terminals:

Table 10-226 Interface overview

Type	Quantity
Fail-safe digital outputs (F-DO)	4
Fail-safe digital inputs (F-DI)	10
Sensor [1] power supplies, dynamic response supported [2]	2
Sensor [1] power supplies, no dynamic response	1
Digital inputs for testing the F-DO during test stop	4

[1] Sensors: Fail-safe devices for commanding and detecting, such as emergency stop pushbuttons and safety locks as well as position switches and light arrays / light curtains.

[2] Dynamic response: The sensor power supply is switched on and off during test stop for testing the sensors, the cable routing, and the evaluation electronics of TM54F.

The TM54F has 4 fail-safe digital outputs and 10 fail-safe digital inputs. A fail-safe digital output consists of a P/M-switching output as well as a digital input for reading back the switching state. A fail-safe digital input is made up of two digital inputs.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

You will find detailed information on the TM54F Terminal Module in the following sources:

- *SINAMICS S120 Control Units and Additional System Components* Manual
- *SINAMICS S120 Safety Integrated* Function Manual

TM15 and TM17 High Feature terminal modules

Features of TM15 and TM17 High Feature

The TM15 and TM17 High Feature Terminal Modules are used to implement inputs of measuring inputs and outputs of output cams for SIMOTION D. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times. TM15 and TM17 High Feature are connected via DRIVE-CLiQ.

TM15

Each of the 24 electrically isolated digital I/Os can be parameterized channel-by-channel as a digital input (DI), digital output (DO), a measuring input input, or an output cam output.

TM15 DI/DO

Each of the 24 isolated digital I/Os can be configured on a channel-specific basis as a digital input (DI) or digital output (DO). The digital I/Os can be interconnected using BICO technology and thus used from the drive side as well. Unlike the TM15, measuring input inputs and cam outputs are not available with the TM15 DI/DO.

Note: The module hardware for TM15 and TM15 DI/DO is identical. A distinction is only made by the addition of the component in the SIMOTION SCOUT Project Navigator using "Inserting input/ output component."

TM17 High Feature (only available as spare part)

Each of the 16 digital I/Os can be parameterized channel-by-channel as a digital input (DI), digital output (DO), measuring input input, or an output cam output.

TM17 High Feature has fewer I/O channels than TM15, but more functionality. TM17 High Feature is distinguished by especially high resolution and accuracy as well as a parameterizable input filter and enabling inputs (max. 6 units). Parameterized enable inputs can enable measuring inputs or outputs of output cams (gate function). Due to their high accuracy, the digital I/O channels of the TM17 High Feature are non-isolated.

Note

NOTICE
Overheating if ventilation clearances are too small
Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component.
Maintain 50 mm ventilation clearances above and below the component.

Additional references

You will find further information on TM15 and TM17 High Feature in the

- *TM15 / TM17 High Feature Terminal Modules* Manual
- *Terminal Modules TM15 and TM17 High Feature* Commissioning Manual

CUA31/CUA32 control unit adapter

Properties of the CUA31/CUA32

You can connect Power Modules in blocksize format via DRIVE-CLiQ to the D4x5-2 Control Units using the CUA31/CUA32 adapter modules.
The CUA32 adapter module also has an additional encoder interface for an HTL, TTL, or SSI encoder.

The following Power Modules are supported:

- PM340
- PM240-2 (as of SIMOTION V4.4 / SINAMICS V4.7)

The mixed operation of a PM240-2 with booksize modules and/or PM340 blocksize modules on a CU320-2/D4x5-2/CX32-2 is possible as of SINAMICS V4.7 HF12 or SIMOTION V4.4 HF6 (SINAMICS Integrated V4.7 HF12).

Table 10-227 Number of interfaces on the adapter modules

Interface	CUA31[1)]	CUA32
DRIVE-CLiQ interface	3	3
EP terminal / temperature sensor connection	1	1
Power Module Interface (PM-IF)	1	1
24 V electronic power supply	1	1
Encoder interface (HTL, TTL, SSI) Only SSI encoders without incremental tracks may be operated.	0	1
DRIVE-CLiQ cable length, max.	100 m	100 m

1) CUA31 with article number 6SL3040-0PA00-0AAx (x ≥ 1 required)

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component. The ventilation openings must not be covered by cables.

Additional references

You will find more information on the CUA31/CUA32 Control Unit Adapter in the *SINAMICS S120 AC Drive* Manual.

DMC20 DRIVE-CLiQ hub

Properties

The DMC20 and DME20 DRIVE-CLiQ hub modules are used to implement point-to-point distribution of a DRIVE-CLiQ line. With the DMC20/DME20, an axis grouping can be expanded with four DRIVE-CLiQ sockets for additional subgroups.

- DMC20 is the hub for the control cabinet configuration
- DME20 is the hub for use without a control cabinet (IP67 degree of protection).

The modules are especially suitable for applications which require DRIVE-CLiQ nodes to be removed in groups, without interrupting the DRIVE-CLiQ line and therefore the data exchange.

Additional references

You will find detailed information about the DMC20/DME20 in the following source:

SINAMICS S120 Control Units and Supplementary System Components Manual

10.2.1.8 Spare parts/accessories

Available spare parts and accessories

Table 10-228 Spare parts and accessories

Parts for the SIMOTION D4x5-2	Article number	Accesso-ries	Spare part
CompactFlash card (CF card) 2 GB CompactFlash card (CF card) with drive software and SIMOTION Kernel (latest CF card at the time of writing)	6AU1400-2QA20-0AA0	x	
Seal for external heat dissipation (only for D445-2 DP/PN and D455-2 DP/PN)	6FC5348-0AA07-0AA0	x	
Double fan/battery module incl. battery The double fan/battery module is already included in the scope of delivery for the SIMOTION D4x5-2.	6FC5348-0AA02-0AA0		x
3 V lithium battery for fan/battery module	6FC5247-0AA18-0AA0		x
Terminal kit, contains • 3 x I/O connectors for X122/X132/X142 • 1 x 24 V connector for X124 • 5 x DRIVE-CLiQ blanking covers for X100-X105	6SL3064-2CB00-0AA0		x
Option slot protective cover	6SL3064-3CB00-0AA0		x

Parts for the SIMOTION D4x5-2	Article number	Accessories	Spare part
Spacer			
• For SIMOTION D425-2/D435-2	6SL3064-1BB00-0AA0		x
• For SIMOTION D445-2/D455-2	6FC5348-0AA06-0AA0		x
Dust protection blanking plugs for sealing unused DRIVE-CLiQ, Ethernet, or PROFINET ports			
• Filler plugs (50 pcs)	6SL3066-4CA00-0AA0	x	x
Blanking cover for the protection of the operator controls	6SL3064-3BB00-0AA0		x

Accessories for PROFIBUS	Article number	Accessories	Spare part
PROFIBUS RS485 bus connector with angular cable outlet (35°) with screw-type terminals, max. transmission rate 12 Mbit/s			
• Without PG/PC interface	6ES7972-0BA42-0XA0	x	
• With PG/PC interface	6ES7972-0BB42-0XA0	x	
PROFIBUS FastConnect bus RS485 connector with angular cable outlet (35°) with insulation displacement terminals, max. transmission rate 12 Mbit/s			
• Without PG/PC interface	6ES7972-0BA61-0XA0	x	
• With PG/PC interface	6ES7972-0BB61-0XA0	x	
PROFIBUS adapter connector for raising the PROFIBUS connector to create more wiring space	6FX2003-0BB00	x	

Accessories for PROFINET (interface X150)	Article number	Accessories	Spare part
RJ45 FastConnect connector for Industrial Ethernet / PROFINET			
• 145° cable outlet (10/100 Mbps)			
- 1 pack = 1 unit	6GK1901-1BB30-0AA0	x	
- 1 pack = 10 units	6GK1901-1BB30-0AB0	x	
- 1 pack = 50 units	6GK1901-1BB30-0AE0	x	
FastConnect cables for Industrial Ethernet / PROFINET [1]			
• IE FC standard cable GP 2x2	6XV1840-2AH10	x	
• IE FC flexible cable GP 2x2	6XV1870-2B	x	
• IE FC trailing cable GP 2x2	6XV1870-2D	x	
• IE FC trailing cable 2x2	6XV1840-3AH10	x	
• IE FC marine cable 2x2	6XV1840-4AH10	x	
Stripping tool for Industrial Ethernet / PROFINET FastConnect cables			
• IE FC Stripping Tool	6GK1901-1GA00	x	

[1] Sold by the meter; max. length 1000 m; minimum order 20 m.

Accessory for Industrial Ethernet (interface X120, X127, X130)	Article number	Accesso-ries	Spare part
RJ45 FastConnect connector for Industrial Ethernet / PROFINET			
• 180° cable outlet (10/100/1000 Mbit/s)			
- 1 pack = 1 unit	6GK1901-1BB11-2AA0	x	
- 1 pack = 10 units	6GK1901-1BB11-2AB0	x	
- 1 pack = 50 units	6GK1901-1BB11-2AE0	x	
FastConnect cables for Industrial Ethernet / PROFINET [1]			
• IE FC Standard Cable GP 4x2	6XV1878-2A	x	
• IE FC flexible cable GP 4x2	6XV1878-2B	x	
Stripping tool for Industrial Ethernet / PROFINET FastConnect cables			
• IE FC Stripping Tool	6GK1901-1GA00	x	
Dust-proof filler plugs for sealing unused DRIVE-CLiQ, Ethernet, or PROFINET ports			
• Filler plugs (50 pcs)	6SL3066-4CA00-0AA0	x	

[1] Sold by the meter; max. length 1000 m; minimum order 20 m.

You can find order data information for other SINAMICS drive components, such as Line Modules, Motor Modules, DRIVE-CLiQ cables, etc. in Catalog *SIMOTION PM 21*.

Connectors and cables

The adapter plug (Article No. 6FX2003-0BB00) is required for the D4x5-2 when the bus cable has to be looped through the left-hand PROFIBUS interface (X126; two PROFIBUS cables wired to the plug) and also

- Ethernet interface X120, in the case of D4x5-2 DP or
- Port 3 of the PROFINET interface X150 in the case of D4x5-2 DP/PN

has to be wired to a FastConnect plug. When using the adapter plug, the PROFIBUS connector is higher, which creates extra wiring space.

Ethernet interfaces X120, X127 and X130 support 10, 100 and 1000 Mbps. For 1000 Mbps, 8-core cables (4x2) must be used as well as the 1000 Mbit version of the 180° FastConnect plug.

The 145° FastConnect plugs cannot be used for Ethernet interface X130 (cable outlet downward). They also only support a maximum of 100 Mbit/s.

Spares On Web

Spares On Web is an information system that displays which spare parts are available for your device.

Spares On Web (https://www.sow.siemens.com)

In order to view the spare parts, you require the article number and the serial number of the module. Both numbers can be found on the rating plate on the module or the packaging label.

Repair parts

Repair parts (e.g. front flaps, shield connection terminals) have only one material number (A5E... or GWE-...).

In Spares On Web, repair parts are only visible to repair centers.

In the case of SIMOTION D, selected repair parts can also be ordered by customers.

For additional information, see FAQ (https://support.industry.siemens.com/cs/ww/en/view/29727640).

10.2.1.9 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

EMC limits in South Korea

KC	KC registration number: KCC-REM-S49-SIMOTION For sellers or other users, please keep in mind that this device in an A-grade electromagnetic wave device. This device is intended to be used in areas other than home. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

The EMC limits to be observed for Korea correspond to the limits of the EMC product standard for variable-speed electric drives EN 61800-3 of category C2 or the limit class A, Group 1 according to EN 55011. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. Additional measures, such as the use of an additional RFI suppression filter (EMC filter), may be necessary.

The measures for EMC-compliant design of the system are described in detail in this manual respectively in the Installation Guideline EMC.

Note that the final statement on compliance with the standard is provided by the respective label attached to the individual unit.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert)).

Marking for Australia and New Zealand

	SIMOTION D4x5-2 incl. CX32-2 and CBE30-2 satisfy the requirement of the standard AS/NZS CISPR 16.
or	
	Marking with RCM (Regulatory Compliance Mark) or C-Tick with older components.

Marking for the Eurasian customs union

EAC	EAC (Eurasian Conformity) Customs union of Russia, Belarus and Kazakhstan Declaration of conformity in accordance with the technical regulations of the customs union (TR CU).

Standards that are not relevant

	China Compulsory Certification SIMOTON D does not belong to the validity area of the China Compulsory Certification (CCC).

China RoHS

SIMOTION D complies with the China RoHS directive. You can find more information on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109738656).

SIMOTION D4x5-2 device-specific notes

Note regarding SIMOTION D

Note

The product standard EN 61800-3 describes the EMC requirements placed on "Variable-speed drive systems". As such, it defines different limits depending on the location of the drive system.

SINAMICS S120 power units are designed for use in the second environment. The term second environment refers to all locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

The same installation instructions apply for the SIMOTION D4x5-2/CX32-2 Control Units as for the SINAMICS S120 CU320-2 Control Units with regard to EMC.

It is essential to follow the installation instructions in the SINAMICS S120 Manuals in order to ensure compliance with emitted interference and immunity values.

For further information on this topic also refer to the *SIMOTION PM 21* Catalog as well as the SINAMICS Function Manuals.

10.2.1.10 ESD guidelines

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

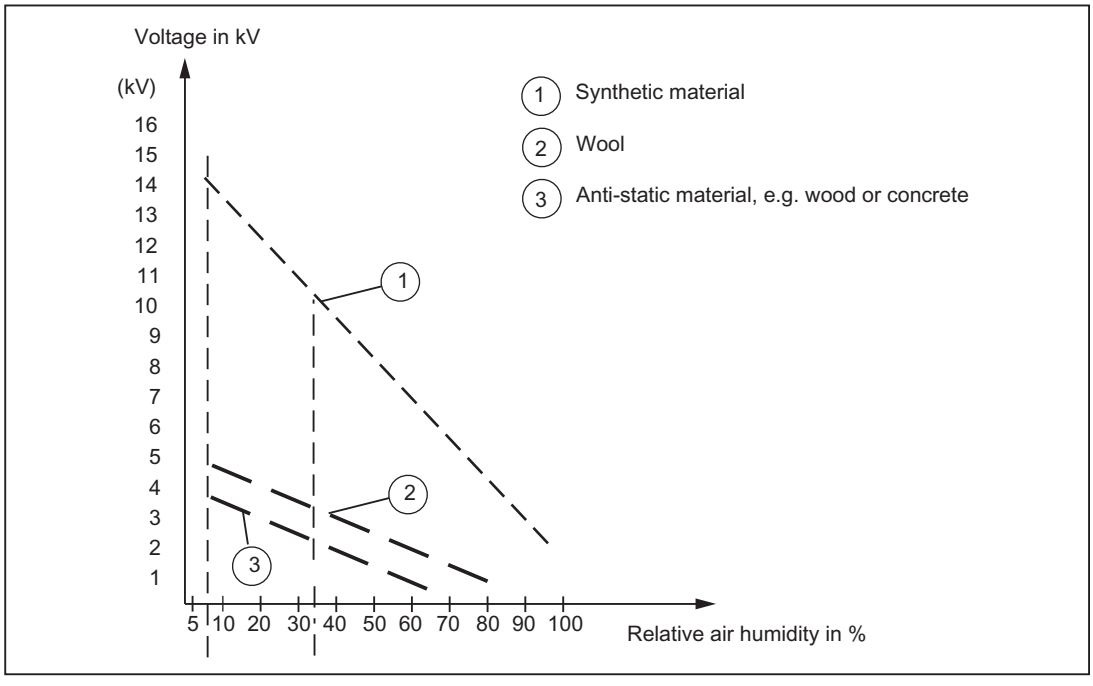


Figure 10-267 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

10.2.2 SIMOTION D410-2

Preface

Contents of the Product Manual

This document is part of the **SIMOTION D** documentation package.

Scope

The SIMOTION D410-2 Manual describes the SIMOTION D410-2 DP and SIMOTION D410-2 DP/PN control units.

Note

A separate SIMOTION D410 Manual is available for the SIMOTION D410 DP and SIMOTION D410 PN control units.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Sections in this manual

The following sections describe the purpose and the use of the manual:

- Description
  This section provides information pertaining to the SIMOTION system and its integration in the information landscape.
- Operator control (hardware)
  This section describes the operator control and display elements of the SIMOTION D410-2.
- Interfaces
  This section provides information about the interfaces, their pin assignments and application options.
- Technical data
  This section describes the properties and features of the SIMOTION D410-2.
- Dimension drawings
- Spare parts / accessories
  This section provides information about accessories and spare parts for the SIMOTION D410-2.
- Appendix
  This section provides information about the various standards, approvals and EMC directives that the device complies with.
- Index to locate information

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Disposal and recycling

SIMOTION D410-2 is an environmentally friendly product! It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner

Further information / FAQs

You can find further information on this manual at the following FAQ:

https://support.industry.siemens.com/cs/ww/de/view/27585482

The following information sources are also available:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling)
- The latest SIMOTION FAQs at https://support.industry.siemens.com/cs/ww/de/ps/14505/faq
- SIMOTION SCOUT online help
- For additional documentation, see the *Overview of SIMOTION documentation* (separate document).

Open source software

Third-party software - License conditions and copyright notes

Copyright notes for the third-party software contained in this product, in particular the open source software, as well as the applicable license conditions, can be found in the READ_OSS.ZIP file on the SIMOTION D CF card or in the corresponding firmware files.

Special note for resellers

The notes and the license conditions contained in the READ_OSS.ZIP file must be passed on to the purchaser in order to avoid the reseller and purchaser from violating the license conditions.

Source code availability

Some license terms of third-party software components used in this product may require us to provide you with the source code and other information for those components. You can find this information directly on or with the product (e.g. on mass storage devices, DVD). If this is not possible for technical reasons, Siemens will be happy to send you this OSS source code in exchange for reimbursement of the processing costs. Please contact the address provided at the end of this section.

Siemens AG

Digital Factory Customer Services

DI CS SD CCC TS

Gleiwitzer Str. 555

D-90475 Nuremberg, Germany

Internet (https://support.industry.siemens.com/cs/ww/en/ps)

Tel.: +49 911 895 7222

10.2.2.1 Safety instructions

Fundamental safety instructions

General safety instructions

⚠ WARNING

Electric shock and danger to life due to other energy sources

Touching live components can result in death or severe injury.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, the following six steps apply when establishing safety:

1. Prepare for disconnection. Notify all those who will be affected by the procedure.
2. Isolate the drive system from the power supply and take measures to prevent it being switched back on again.
3. Wait until the discharge time specified on the warning labels has elapsed.
4. Check that there is no voltage between any of the power connections, and between any of the power connections and the protective conductor connection.
5. Check whether the existing auxiliary supply circuits are de-energized.
6. Ensure that the motors cannot move.
7. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water. Switch the energy sources to a safe state.
8. Check that the correct drive system is completely locked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Electric shock if there is no ground connection

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Electric shock due to connection to an unsuitable power supply

When equipment is connected to an unsuitable power supply, exposed components may carry a hazardous voltage. Contact with hazardous voltage can result in severe injury or death.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV- (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Electric shock due to equipment damage

Improper handling may cause damage to equipment. For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Ensure compliance with the limit values specified in the technical data during transport, storage and operation.
- Do not use any damaged devices.

⚠ WARNING

Electric shock due to unconnected cable shield

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the conductors of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Spread of fire from built-in devices

In the event of fire outbreak, the enclosures of built-in devices cannot prevent the escape of fire and smoke. This can result in serious personal injury or property damage.

- Install built-in units in a suitable metal cabinet in such a way that personnel are protected against fire and smoke, or take other appropriate measures to protect personnel.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Unexpected movement of machines caused by radio devices or mobile phones

The use of radio devices or mobile telephones in the immediate vicinity of the components can result in equipment malfunction. Malfunctions may impair the functional safety of machines and can therefore put people in danger or lead to property damage.

- If you come closer than around 2 m to such components, switch off any radios or mobile phones.
- Use the "SIEMENS Industry Online Support app" only on equipment that has already been switched off.

⚠ WARNING

Fire due to inadequate ventilation clearances

Inadequate ventilation clearances can cause overheating of components with subsequent fire and smoke. This can cause severe injury or even death. This can also result in increased downtime and reduced service lives for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

NOTICE

Overheating due to inadmissible mounting position

The device may overheat and therefore be damaged if mounted in an inadmissible position.

- Only operate the device in admissible mounting positions.

⚠ WARNING

Unrecognized dangers due to missing or illegible warning labels

Dangers might not be recognized if warning labels are missing or illegible. Unrecognized dangers may cause accidents resulting in serious injury or death.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, where necessary in the national language.
- Replace illegible warning labels.

⚠ WARNING

Unexpected movement of machines caused by inactive safety functions

Inactive or non-adapted safety functions can trigger unexpected machine movements that may result in serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for Safety Integrated functions

If you want to use Safety Integrated functions, you must observe the safety notices in the Safety Integrated manuals.

Malfunctions of the machine as a result of incorrect or changed parameter settings

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

Note regarding the general data protection regulation

Siemens observes the principles of data protection, in particular the principle of data minimization (privacy by design). For this product this means:

The product does not process / store any personal data, only technical functional data (e.g. time stamp). If the user links this data with other data (e.g. shift plans) or stores personal data on the same medium (e.g. hard disk) and thus establishes a personal reference, the user must ensure compliance with data protection regulations.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage

3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety information for SIMOTION D410-2

Observe the following safety information when working with SIMOTION D410-2 and its components!

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra low voltage in accordance with EN/IEC 609501 may be connected at all connectors and terminals.

⚠ WARNING

Danger to life from unexpected movement of machines on automatic restart

An automatic restart can be programmed for SIMOTION controllers. When the power returns, the axes start automatically.

Make sure this presents no hazard to personnel or property.

NOTICE

Damage to the CompactFlash card from electrical fields or electrostatic discharge

The CompactFlash card is an ESD-sensitive component.

De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF.

Comply with the ESD rules.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of systems / devices. Make sure the ventilation clearances of 50 mm are provided above and below the components. The ventilation openings may not be covered by connecting cables.

10.2.2.2 Description

System overview

SIMOTION D

SIMOTION D is a drive-based version of SIMOTION based on the SINAMICS S120 drive family.

With SIMOTION D, the SIMOTION PLC and motion control functionalities as well as the SINAMICS S120 drive software run on shared control hardware.

SIMOTION D is available in two versions:

- SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications.
- SIMOTION D4x5-2 is a Control Unit for multi-axis applications in the SINAMICS S120 booksize format.
  The following performance variants of the SIMOTION D4x5-2 Control Units are offered:

Control Unit	Performance variant	Range of applications
SIMOTION D425-2	BASIC performance	For up to 16 axes
SIMOTION D435-2	STANDARD performance	For up to 32 axes
SIMOTION D445-2	HIGH performance	For up to 64 axes
SIMOTION D455-2	ULTRA-HIGH performance	For up to 128 axes or applications with very short control cycles

Note

The SIMOTION D410-2 is described in this manual.

Separate manuals are available for the SIMOTION D4x5-2 and the SIMOTION D4x5 and SIMOTION D410 predecessor modules.

SIMOTION D is an integral part of the Totally Integrated Automation (TIA) concept. TIA is characterized by integrated data management, configuration, and communication for all

products and systems. Thus, an extensive toolbox of automation modules is also available for the SIMOTION D410-2.

Note

In order to cover all variants of SIMOTION D in blocksize format, the product will be referred to as "D410-2". Specific product designations will be used for information that applies only to one product version, e.g. D410-2 DP/PN.

SIMOTION D410-2

Figure 10-268 SIMOTION D410-2 DP (pictured on left), SIMOTION D410-2 DP/PN (pictured on right)

SIMOTION D410-2 is a compact Control Unit for single-axis applications.

The Control Unit is snapped directly on to the SINAMICS Power Module in blocksize format and has an integrated drive control for either one servo, one vector or one V/f axis.

SIMOTION D410-2 can be extended with additional SINAMICS S110/S120 control units (e.g. CU310-2) and so can also be used for smaller multi-axis applications (e.g. with 2 - 3 axes).

Example of a single-axis application

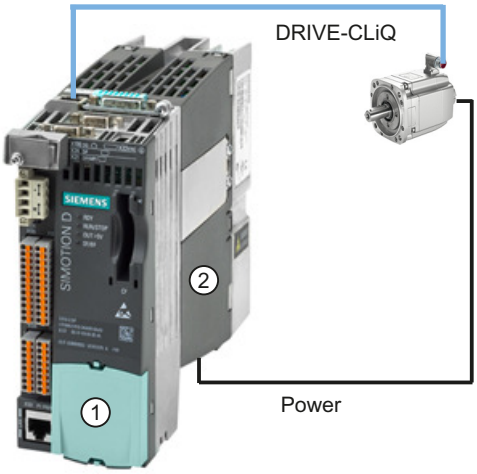


Figure 10-269 Application example with one axis

The example shows a single-axis application, consisting of a SIMOTION D410-2 (Control Unit) ① that is snapped directly on to the SINAMICS Power Module in blocksize format ②. The motors are supplied with power via the Power Module. The encoder is connected by means of DRIVE-CLiQ.

Example of a multi-axis application

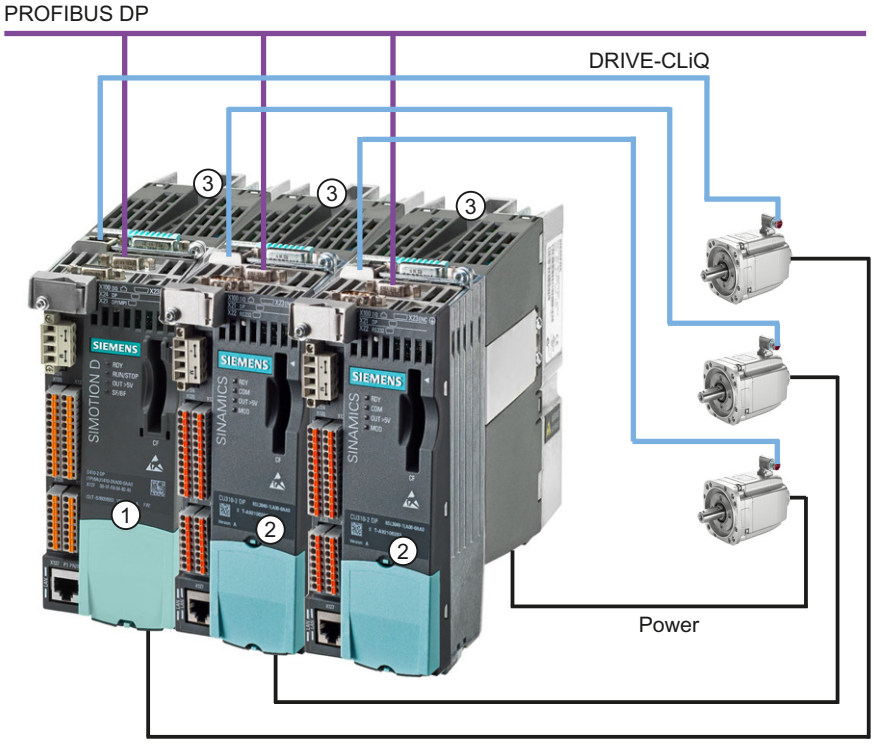


Figure 10-270 Application example with 3 axes

The example shows an application with 3 axes, consisting of:

- One SIMOTION D410-2 DP (Control Unit) ①, snapped on to the Power Module in blocksize format ③
  The SIMOTION D410-2 DP is snapped directly on to the SINAMICS Power Module. The motors are supplied with power via the Power Module. The encoder is connected by means of DRIVE-CLiQ.
- Two SINAMICS S120 CU310-2 DP ②, snapped onto a Power Module in blocksize format ③
  The Control Units are connected to the SIMOTION D410-2 DP via PROFIBUS DP. The two SINAMICS S120 CU310-2 DP are snapped directly on to the SINAMICS Power Module. The motors are supplied with power via the Power Modules. The encoders are connected by means of DRIVE-CLiQ.

Note

Path interpolation is supported as of V4.4.

Application

Combining a Power Module with SIMOTION D410-2 forms a compact single drive for machine and plant engineering.

Applications include:

- Machine concepts with central drive (e.g. presses, printing and packaging machines, etc.)
- Modular machine concepts where the machine modules were broken down into single axes
- Single drives with high accuracy, stability and concentricity requirements (compared with standard drives) in machine and industrial plant engineering
- Single drives for transport tasks (conveying, raising, lowering)
- Single drives with integrated PLC functionality and expanded motion control functionality such as output cams or cams
- Drives without power recovery (wire drawing, extruding)
- Drive connections with high availability requirements (incoming supply failure may not cause all axes to fail)
- Small multi-axis groupings (typically 2 to 3 axes) based on SINAMICS S110/120 blocksize.

Hardware components

As central hardware the SIMOTION D410-2 Control Unit is made up of the SIMOTION runtime system and the SINAMICS drive control.

A range of additional SINAMICS S120 components, such as SMx encoder systems or Terminal Modules can be connected via DRIVE-CLiQ.

With a few exceptions (e.g. no BOP20 Basic Operator Panel, etc.), the drive control integrated in SIMOTION D410-2 has the same control properties and performance features as the SINAMICS S120 CU310-2 Control Unit.

Extension of the drive computing performance

To fully utilize the motion control performance of a SIMOTION D410-2 when required, the drive-side computing performance can be extended by connecting additional SINAMICS S/G Control Units (e.g. CU305, CU310-2, CU320-2, CU250S-2, etc.) via PROFIBUS or PROFINET to the SIMOTION D410-2.

Software components

The basic functionality of SIMOTION D is supplied on a CompactFlash card containing the following:

- The SIMOTION runtime system with the following functions:
  - Freely programmable runtime system (IEC 61131)
  - Various runtime levels (tasks)
  - PLC and arithmetic functionality
  - Motion control functions
  - Communication functions
- The SINAMICS S120 drive control with the following functions:
  - Closed-loop current and torque control
  - Closed-loop speed control

System components

Overview

SIMOTION D410-2 communicates with the components of the automation landscape via the following interfaces:

- PROFIBUS DP (D410-2 DP and D410-2 DP/PN)
- PROFINET IO (D410-2 DP/PN only)
- Ethernet
- DRIVE-CLiQ (DRIVE Component Link with IQ)
- Power Module interface (PM-IF)

SIMOTION D features a SINAMICS Integrated drive element. Communication with the SINAMICS Integrated is via PROFIBUS mechanisms (DP Integrated), via PROFIdrive telegrams.

Shorter cycle times and greater numbers of addresses for each node are achieved with the "DP Integrated" compared to the "external PROFIBUS DP."

The most important components of the system and their functions are shown below.

Table 10-229 System components

Component	Function
SIMOTION D410-2	... is the central motion control module. The module contains the programmable SIMOTION runtime of SIMOTION D410-2 and the SINAMICS S120 drive runtime software. You can use the integrated high-speed I/Os (onboard I/Os) as: • User-addressable process I/Os • Homing inputs • Fail-safe digital inputs • Fail-safe digital output • Inputs for measuring inputs • Outputs for fast output cams • Analog input The measuring sockets can output any analog signals. The DRIVE-CLiQ interface permits a fast connection to the SINAMICS drive components.
System software	The basic functionality of SIMOTION D410-2 is supplied separately on a CompactFlash Card containing the following: • SIMOTION runtime (kernel) • Drive software of SINAMICS S120 The CompactFlash card is not included in the scope of delivery.
Power supply (PS)	... provides the electronic power supply for SIMOTION D410-2 (e.g. SITOP power supply).

PROFIBUS DP

SIMOTION D410-2 can communicate with the following components via the PROFIBUS DP interface.

Table 10-230 Components on PROFIBUS DP

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of the SIMOTION D410-2.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.

Component	Function
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200S	Finely scalable I/O system for cabinet installation and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/IP67 degree of protection for machine-related applications with no cabinet; with features such as more compact designs, integrated PROFIsafe safety technology, PROFINET connection, and live module replacement.
SIMATIC ET 200eco	I/O system with IP65/IP67 degree of protection for machine-related applications with no cabinet, with a flexible and fast connection system in ECOFAST or M12.
Other PROFIBUS I/O	
Gateways	• DP/AS-Interface Link 20E and DP/AS-Interface Link Advanced for the PROFIBUS DP gateway to AS-Interface • DP/DP coupler for connecting two PROFIBUS DP networks
Drive interfaces	• ADI4 (Analog Drive Interface for 4 axes) for the connection of drives with analog ±10 V setpoint interface or for external encoders • IM 174 (Interface Module for 4 axes) for the connection of drives with analog ±10 V setpoint interface, external encoders or the connection of stepper drives with pulse/direction interface
Drive units with PROFIBUS DP interface (e.g. CU310-2 DP)	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors. Can also be operated as an isochronous slave on the PROFIBUS DP.
Teleservice adapter	Remote diagnostics

PROFINET IO

The SIMOTION D410-2 DP/PN can communicate with the following components via the onboard PROFINET IO interface.

Table 10-231 Components on the PROFINET IO

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of a Control Unit.
Other controllers (e.g. SIMOTION or SIMATIC)	... e.g. higher-level controller (plant controller); modular machine concepts with multiple controllers, distributed across individual machine modules.
Master computer	... communicates with other devices via UDP, TCP/IP.
Distributed I/O systems	
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data. With the time-based I/O, signals can be recorded or output to the precise µs.

Component	Function
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-300 packaging system.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety. With the time-based I/O, signals can be recorded or output to the precise µs.
SIMATIC ET 200S	Finely scalable I/O system for cabinet configuration and particularly time-critical applications; including motor starters, safety technology and individual grouping of load groups.
SIMATIC ET 200AL	Modular, distributed I/O system with compact I/O modules in IP65/67; simple installation in all mounting positions even in small spaces; front and transverse screw fastenings on flat surfaces or on aluminum supporting channels; flexible connection to PROFINET or PROFIBUS or simple integration in SIMATIC ET 200SP.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related applications with no cabinet; with features such as compact designs, integrated PROFIsafe safety technology, PROFINET IO connection and live module replacement.
SIMATIC ET 200eco PN	Compact block I/O with IP65/66/67 degree of protection for cabinet-free usage in machines with M12 connection method. Very rugged and resistant encapsulated metal enclosure.
Other PROFINET IO I/O devices	
Drive units with PROFINET IO interface	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors.
Gateways	• IE/AS-Interface link PN IO for the PROFINET IO gateway to AS-Interface • PN/PN coupler for connecting two PROFINET IO networks

Ethernet

The Control Unit can communicate with the following components via the Ethernet interfaces or be embedded in an automation environment:

Table 10-232 Components on the Ethernet

Component	Function
Programming device (PG/PC)	... configures, assigns parameters, programs, and tests using the SIMOTION SCOUT Engineering System (ES).
Master computer	... communicates with other devices via UDP, TCP/IP.
SIMATIC HMI device	... is used for operating and monitoring functions. This is not an essential requirement for the operation of the SIMOTION D410-2.

DRIVE-CLiQ

SIMOTION D410-2 can communicate via the DRIVE-CLiQ interface with the following components:

Table 10-233 Components on DRIVE-CLiQ

Component	Function
SINAMICS S120 AC DRIVE drive units (with CUA31/CUA32)	... convert speed setpoints into signals for controlling the motor and supply the power required to operate the motors. The Power Module is connected via CUA31/CUA32. No more than one Power Module can be connected. The chassis Power Module is connected via DRIVE-CLiQ. **Note**: Components in booksize format are not supported!
TM15, TM17 High Feature Terminal Modules	The Terminal Modules TM15 and TM17 High Feature are used to implement measuring inputs inputs and output cam outputs. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times.
TM31 Terminal Module	... enables terminal expansion via DRIVE-CLiQ (additional analog and digital I/Os).
TM41 Terminal Module	... enables terminal expansion (analog and digital I/Os) and encoder simulation via DRIVE-CLiQ. The TM41 can be connected to a real axis.
TM54F Terminal Module	... enables terminal expansion (fail-safe digital inputs/outputs) for controlling the safe motion monitoring functions of the integrated drive. A TM54F is not usually necessary because the SIMOTION D410-2 has 3 F-DI and 1 F-DO.
TM120 Terminal Module	Four temperature sensors (KTY84-130 or PTC) can be evaluated via the TM120 Terminal Module. The temperature sensor inputs are safely electrically separated from the evaluation electronics in the TM120 Temperature Module and are suitable for evaluating the temperature of special motors, e.g. 1FN linear motors and 1FW6 built-in torque motors.
TM150 Terminal Module	The TM150 Terminal Module can be used to evaluate temperature sensors (KTY, PT100, PT1000, PTC, and bimetal normally closed contact). This means, for example, that other temperatures from the process can be measured in addition to the motor temperature. Temperature sensors can be evaluated using a 2, 3 or 4-wire system. Twelve temperature sensors can be evaluated with 2-wire evaluation and six temperature sensors with 3 and 4-wire evaluation.
SMx Sensor Modules	... enable acquisition of encoder data from connected motors via DRIVE-CLiQ.
Motors with DRIVE-CLiQ interface	... allow simplified commissioning and diagnostics, as the motor and encoder type are identified automatically.
DMC20/DME20 DRIVE-CLiQ hub	... enables the number of DRIVE-CLiQ interfaces to be increased and the creation of a point-to-point topology.

Note

Please note that SIMOTION D410-2 components in booksize format (Controller Extension, Motor Modules, Line Modules, etc.) are not supported.

SIMOTION D410-2 can only be used with the following Power Modules:

- PM340
- PM240-2 as of SIMOTION V4.4/SINAMICS V4.7

Other Power Modules are not supported by SINAMICS G120 (e.g. PM230).

Note

You will find detailed information on components in the SINAMICS S110/S120 family of products in the SINAMICS S110/S120 manuals.

It is possible that older DRIVE-CLiQ components can no longer be used with SIMOTION D410-2. You will find detailed information on this in the SIMOTION D410-2 Commissioning and Hardware Installation Manual in Section "Migration of SIMOTION D410 to SIMOTION D410-2" under "Permissible combinations".

I/O integration

Note

Note that not all modules in the ET 200 I/O family are approved for SIMOTION. Moreover, system-related functional differences can come into play when these I/Os or I/O systems are used on SIMOTION vs. on SIMATIC. For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

A detailed, regularly updated list of the I/O modules approved for use with SIMOTION, as well as notes on their use, can be found at Internet address (https://support.industry.siemens.com/cs/ww/en/view/11886029)

In addition to the I/O modules enabled for SIMOTION, in principle all certified standard PROFIBUS slaves (DP-V0/DP-V1/DP-V2) and PROFINET IO devices with RT and IRT real-time classes may be connected to SIMOTION D410-2. These modules are integrated using the GSD file (PROFIBUS) or GSDML file (PROFINET) provided by the relevant device manufacturer.

Note

Please note that in isolated cases, additional boundary conditions must be fulfilled in order to integrate a module into SIMOTION. Thus, a few modules require "driver blocks" , e.g. in the form of function blocks, that permit (or simplify) integration.

For modules enabled for SIMOTION (e.g. SIMATIC S7-300 module FM 350-1, etc.), these driver blocks are part of the SIMOTION SCOUT engineering system command library.

SIMOTION D410-2 DP representation

View

The following figure shows a SIMOTION D410-2 DP with the interfaces and front elements.

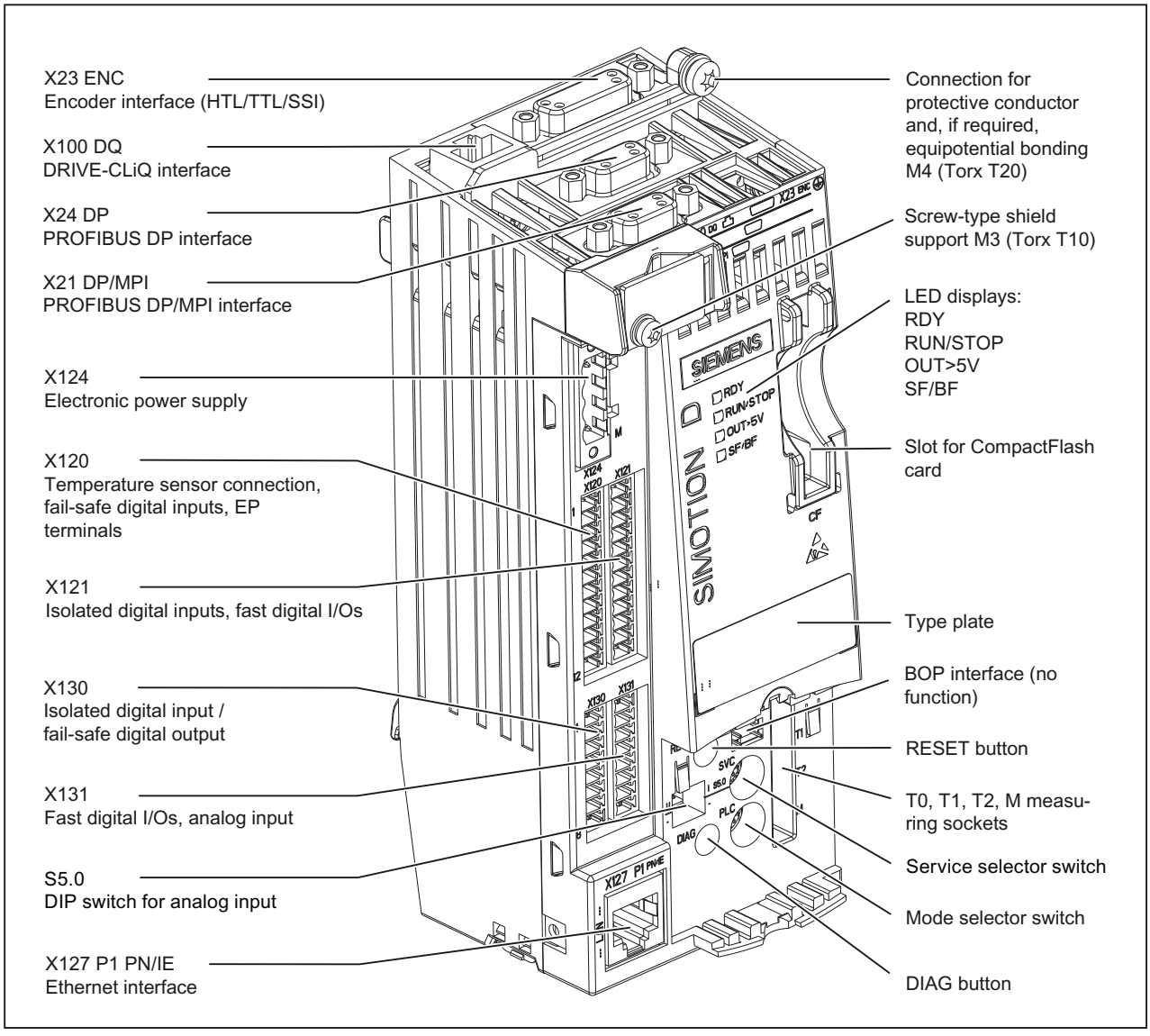


Figure 10-271 Location of interfaces and front elements for SIMOTION D410-2 DP

The interface to the power module (PM) is located at the rear of the SIMOTION D410-2.

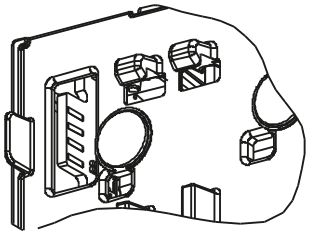


Figure 10-272 Power Module Interface (PM-IF)

See also

Interfaces (Page 7748)

SIMOTION D410-2 DP/PN drawing

View

The following figure shows a SIMOTION D410-2 DP/PN with the interfaces and front elements.

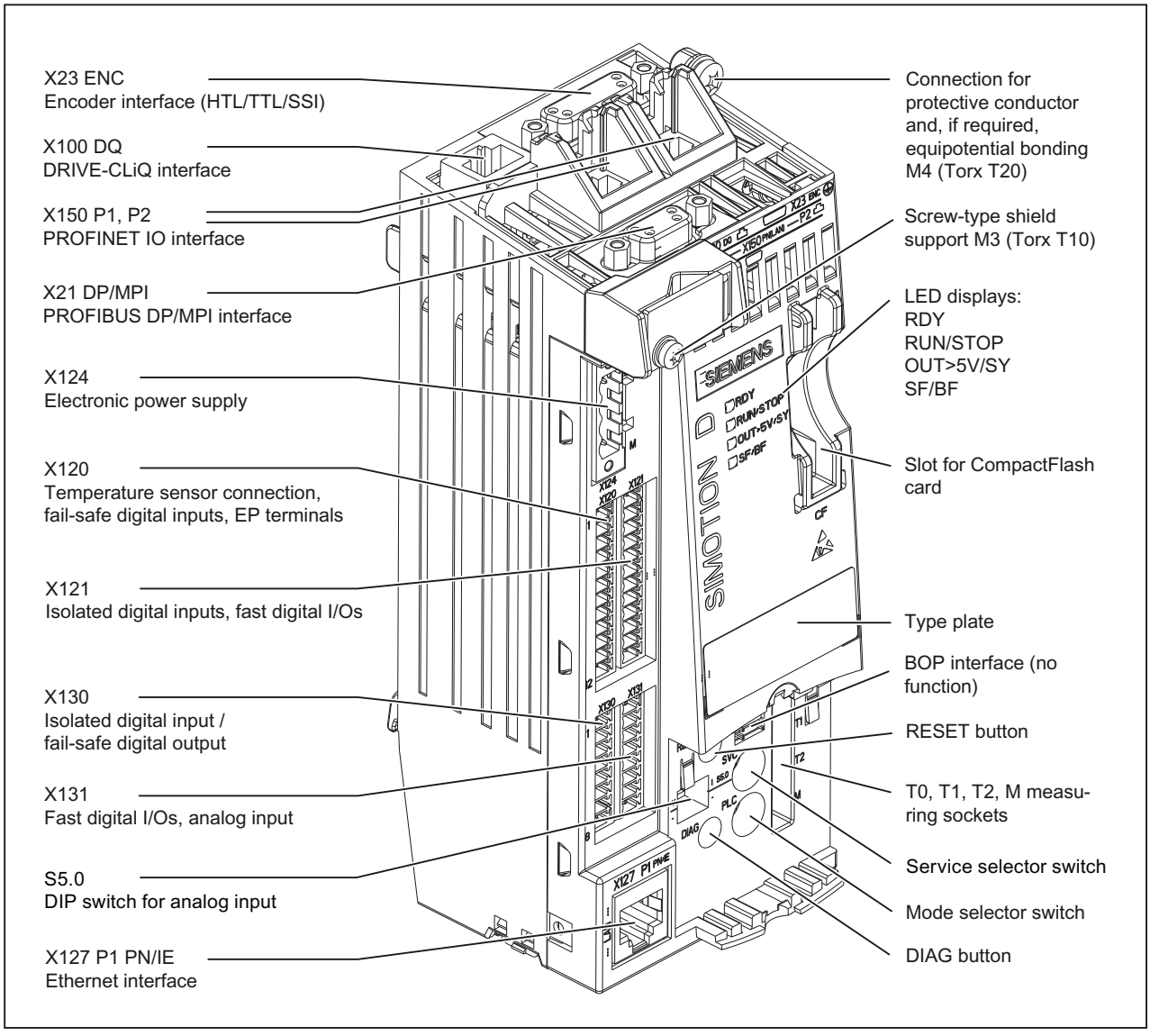


Figure 10-273 Location of interfaces and front elements for SIMOTION D410-2 DP/PN

The interface to the power module (PM) is located at the rear of the SIMOTION D410-2.

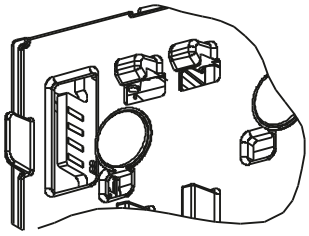


Figure 10-274 Power module interface (PM-IF)

See also

Interfaces (Page 7748)

Type plates

The following figure shows all the information that the rating plate on the module rear contains.

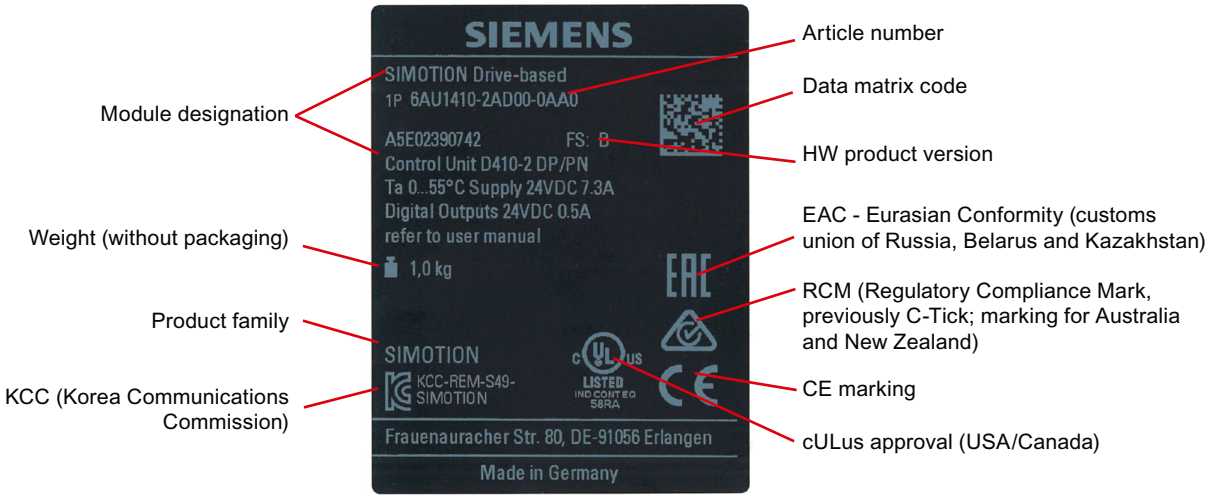


Figure 10-275 Rating plate using the SIMOTION D410-2 DP/PN as an example

Note

The information contained in each field of the rating plate on your actual Control Unit may differ from that presented in this manual (for example, a later product version, approvals and marks that have not yet been earned, etc. may be shown).

Depending on the rating plate, the HW version may be designated as "Version" or "FS" (Function State). Older components are marked with C-Tick instead of RCM.

MAC addresses

A second rating plate for the MAC address of the Ethernet interface and the PROFINET interface (D410-2 DP/PN only) is attached to the front of the device.

Figure 10-276 SIMOTION D410-2 DP/PN MAC addresses

Depending on the rating plate, the HW version may be designated as "Version" or "FS" (Function State).

Industry Online Support app

With our app, you have access to more than 300,000 documents.

Scan the data matrix code and display all the technical information on this product incl. graphic data (CAx data).

Link to the app: https://support.industry.siemens.com/sc/en/en/sc/2067

CompactFlash card

Properties

The CF card is mandatory for operation of the SIMOTION D410-2. The CF card must be ordered as a separate component; it is not included in the SIMOTION D410-2 scope of delivery.

The SIMOTION Kernel (SIMOTION D410-2 firmware) and the software used to control the drives (SINAMICS firmware) are contained on the CF card.

The CF card is used for:

- Backing up the technology packages and user data (programs, configuration data, parameter assignments)
- Update (e.g. SIMOTION firmware update)

The licenses for the technology functions are linked to the serial number of the CF card. This means the CF card can be inserted in different SIMOTION D410-2s without having to change the licenses.

The CF card is supplied in a bootable format with the latest SIMOTION Kernel and drive software.

CF card

Different CF cards are available for SIMOTION D410-2.

- 2 GB CF, article number 6AU1400-1QA20-0AA0
- 1 GB CF, article number 6AU1400-1PA23-0AA0
- 1 GB CF, article number 6AU1400-1PA22-0AA0

You will find detailed information on the compatibility relationships for the CF card, boot loader version, SIMOTION D hardware and SIMOTION firmware version in the software compatibility list.

This list can be found on the Internet at (https://support.industry.siemens.com/cs/ww/en/view/18857317).

CF card rating plate

The following figure shows you all the information included on the rating plate of the CompactFlash card (CF card).

Figure 10-277 CompactFlash card (example for 2 GB CF card)

Pre-installed runtime licenses

The pre-installed licenses are provided as Z option on the label below the article number.

Example with Z option for D410-2 MultiAxes package + two TControl licenses:
6AU1400-1QA20-0AA0-Z
Z=M41+T02

A maximum of seven different Z options are printed on the rating plate of the CF card. When there are more than seven different Z options, the text "Z = see delivery order" is printed on the CF card in place of the Z option.

Available Z options / licenses

The following Z options are available for SIMOTION D410-2:

Table 10-234 Z options / licenses available for the SIMOTION D410-2

Designation	Z option / license	Example
TControl temperature control	Txx - TControl license and number	T03 = three TControl licenses
SIMOTION IT	J00 - SIMOTION IT Virtual Machine license for Java applications	J00
Safety functions	Fxx - SINAMICS Safety Integrated Extended Functions license, specification of the number Lxx - SINAMICS Safety Integrated Advanced Functions license, specification of the number The Safety Integrated Advanced Functions also include the Safety Integrated Extended Functions.	F01 = one license for one drive with Safety Integrated Extended Functions L01 = one license for one drive with Safety Integrated Advanced Functions
High output frequency	H00 - SINAMICS high output frequency for SIMOTION D license	H00
Axis licenses	• Pxx - POS license and number • Gxx - GEAR license and number • Cxx - CAM license and number	• P02 = two POS licenses • G03 = three GEAR licenses • C01 = one CAM license
MultiAxes package	M41 - MultiAxes package license for SIMOTION D410-2	
MIIF	B02 - Multipurpose Information Interface	B02
VIBX	B03 - Vibration Extinction (vibration damping of axes)	B03
OACAMGEN	B04 - Cam generation Motion profiles for servo presses	B04
LECo	B07 - Learning Error Compensation Compensation of cyclic errors from the production process	B07

Note

Path interpolation is supported as of V4.4.

SINAMICS licenses

Selected SINAMICS licenses can be used with a SIMOTION D CF card. Only one relicensing is possible. A prelicensing of SIMOTION D CF cards via Z options is not possible with SINAMICS licenses.

Examples:

- SINAMICS S120 Advanced Position Control (APC)
  Article no. 6SL3074-0AA05-0AA0
  License for each drive (on CU, SINAMICS Integrated)
- SINAMICS S120 cogging torque compensation
  Article no. 6SL3074-0AA15-0AA0
  License for each drive (on CU, SINAMICS Integrated)
- SINAMICS Technology Extension "Vibration Extinction" (VIBX)
  Article no. 6SL3077-0AA00-5AB0
  License for a target device (CU, SINAMICS Integrated)
- SINAMICS DCB Extension
  Article no. 6SL3077-0AA00-0AB0
  License for a target device (CU, SINAMICS Integrated)

With SINAMICS licenses, underlicensing of SINAMICS Integrated is indicated by the flashing SF LED on the SIMOTION D Control Unit. An entry is also made in the diagnostic buffer and the underlicensing is displayed in the License dialog box of SIMOTION SCOUT. The licensing is performed (as for SIMOTION licenses) via SIMOTION SCOUT or via the SIMOTION license key on the CF card.

Data matrix code

SIMOTION D components (e.g. CF cards, Control Units, etc.) have a machine-readable identification in the form of a data matrix code (2D code).

Reader units that support the data matrix code according to ECC 200 are suitable for reading the code used here.

Example of a data string from the reader:
1P6AU1400-1QA20-0AA0-Z+ST-LDBN05403+30S012610D2609U2416

The volume of the information contained in the data matrix code depends on the product and, for example, on the available space.

Table 10-235 Machine-readable identification via 2D code

Characteristic	Property (example)
Article number ("1P" identifier to identify the products)	6AU1400-1QA20-0AA0-Z
Serial number ("S" identifier, item number)	T-LDBN05403
Hardware serial number (CF cards only) ("30S" identifier)	012610D2609U2416
Hardware version (identifier 2PE)	Not used in the example
Material number (identifier P)	Not used in the example

In addition to the "serial number", CF cards also have a "hardware serial number".

If licenses are purchased for licensed functions, a "license key" is generated from the hardware serial number of the CF card and the serial number of the purchased licenses; such licenses are valid only for the associated CF card.

The data required for the licensing can be read by a reader unit via the bar codes on the license certificates (Certificate of License "CoL") and the 2D code on the CF card in order, for example, to automate the licensing process.

Industry Online Support app

With our app, you have access to more than 300,000 documents.

Scan the data matrix code and display all the technical information on this product incl. graphic data (CAx data).

Link to the app: https://support.industry.siemens.com/sc/en/en/sc/2067

Licensing

SIMOTION D410-2 licensing

SIMOTION D410-2 is a compact Control Unit predestined for single-axis applications. SIMOTION D410-2 has an integrated drive control for either a servo, a vector or a V/f axis. One real axis can be used without requiring a license for a SIMOTION D410-2. Speed-controlled axes and virtual axes never require a license.

SIMOTION D410-2 can be extended with additional SINAMICS S110/S120 Control Units (e.g. CU305) and so can also be used for smaller multi-axis applications (e.g. with 2 - 3 axes). These additional axes must be licensed with the single-axis licenses or the "D410-2 MultiAxes Package". See Section CompactFlash card (Page 7735).

The POS single-axis license is available if a POS axis has to be licensed; with GEAR/CAM or more than one POS license, it is better to use the D410-2 MultiAxes Package.

Note

If you use more than one real axis with SIMOTION D410-2, you must license these additional axes. The axis license with the highest functionality is covered by the inclusive license (a real axis). The functionality has the following granularity: CAM > GEAR > POS.

Example:

You use two real axes: 1 POS, 1 CAM.
Because the CAM license has a higher value, and so inclusive, you only need to purchase one POS license.

Licenses are required for runtime functions such as SIMOTION IT Virtual Machine. These licenses can be pre-installed on a CompactFlash card (CF card) or ordered separately.

Additional references

For more information about license management, see the *SIMOTION SCOUT* Configuration Manual. General information on the subject of licensing can be found in the *SIMOTION PM 21* Catalog.

10.2.2.3 Operator control (hardware)

Overview of operator control and display elements

The following figure shows the arrangement of the operator control and display elements on the SIMOTION D410-2.

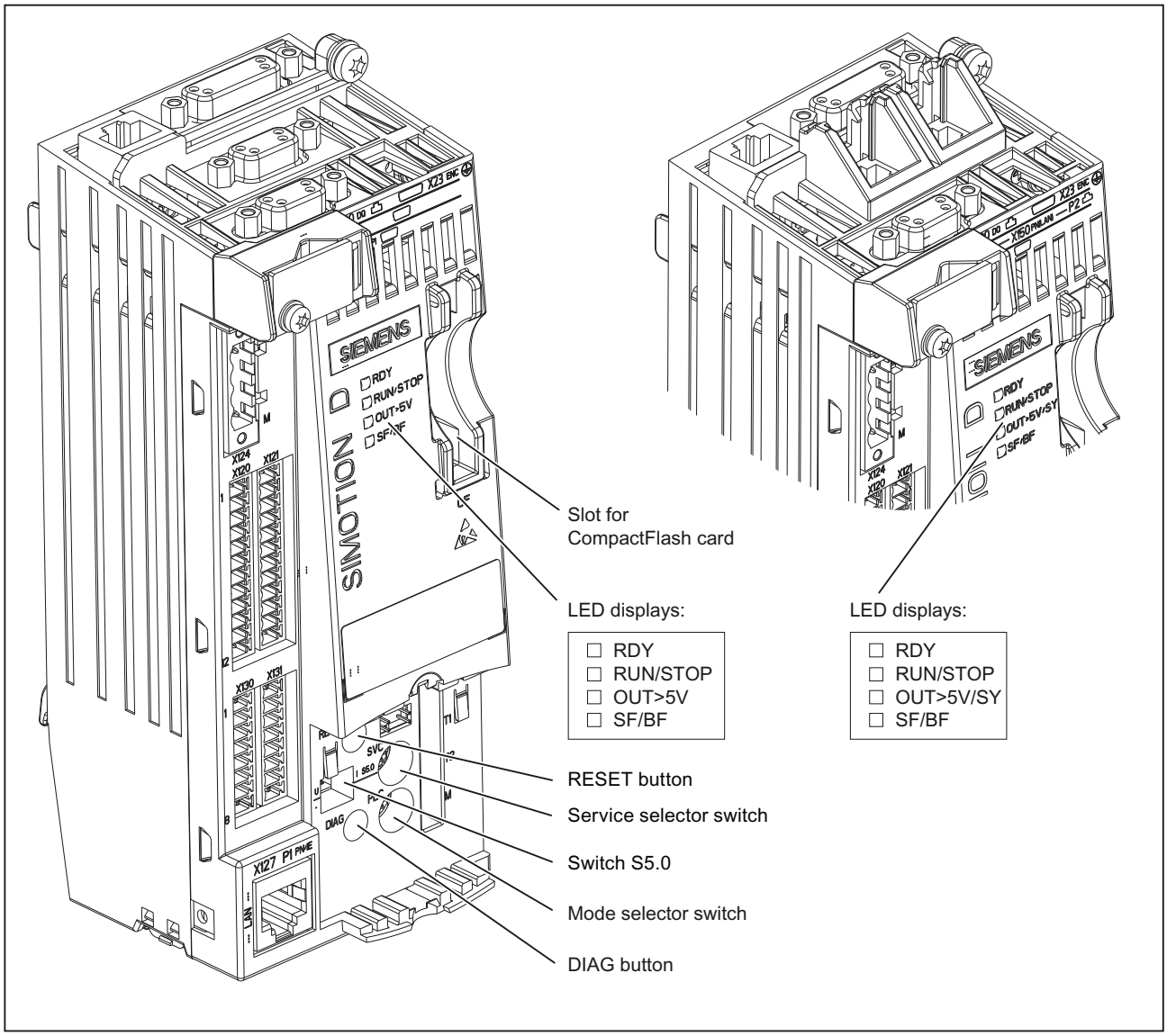


Figure 10-278 Operator control and display elements: SIMOTION D410-2 DP (on the left), SIMOTION D410-2 DP/PN (on the right)

Operator controls

Service selector switch

Layout

SIMOTION D410-2 provides a service selector switch (SVC) behind the blanking cover in the lower area of the front panel.

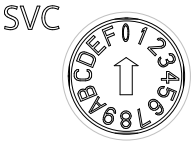


Figure 10-279 Service selector switch, switch position "0"

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Comply with the ESD rules.

Function

The service selector switch is used to select service functions.

In "normal" operation, this switch must remain in the "0" position.

The following table shows the possible positions of the service selector switch. The service selector switch positions are explained in the order in which they are arranged on the SIMOTION D410-2. The service functions can generally be used in any set operating mode.

Table 10-236 Switch positions of the service selector switch

Position	Service mode	Meaning	
0		No service/diagnostic function activated	
1 or A → 1	Delete/restore non-volatile SIMOTION data	The non-volatile SIMOTION data of the SIMOTION D410-2 is first deleted and then restored with the contents of the PMEMORY backup file.	
		Position "1"	The data backed up with the system function _savePersistentMemoryData is preferably restored
		Position "A" → "1" (as of V4.4)	The data backed up by service selector switch position "D" / Web server / DIAG pushbutton are preferably restored
8	Web server in security level low	Switches the SIMOTION IT Web server to Security Level Low for 120 minutes. You will find detailed information in the *SIMOTION IT Diagnostics and Configuration* Diagnostics Manual.	

Position	Service mode	Meaning
B	Downgrade (Device Update Tool)	SIMOTION D410-2 Control Units and projects can be upgraded using upgrade data created previously. These upgrade data is generated with the Device Update Tool ("Project > Start Device Update Tool" menu in SIMOTION SCOUT). If the upgrade process fails to bring about the desired result, the upgrade can be rejected by means of the switch position. This will roll the system back to the previous configuration.
D	Backup of diagnostic data and non-volatile SIMOTION data	The diagnostic data and non-volatile SIMOTION data can be backed up in STOP, STOPU, and RUN state. The advantage of backing up in RUN state is the availability of enhanced diagnostic information (via HTML pages) and TO alarm information.

Note

Alternatively, diagnostic data and non-volatile SIMOTION data can also be backed up via the DIAG button. See Section DIAG button (Page 7744).

Mode selector switch

Layout

SIMOTION D410-2 provides a mode switch (PLC) behind the blanking cover in the lower area of the front panel.

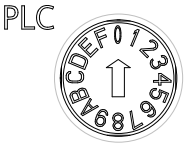


Figure 10-280 Mode switch, switch position "0"

NOTICE
Damage from electrostatic discharge
The rotary switch can be destroyed by static electricity.
Operate the rotary switch only with an insulated screwdriver.
Comply with the ESD rules.

Function

The following table contains the possible mode switch positions and the associated operating mode.

Table 10-237 Mode switch positions

Position	Operating mode	Meaning
0	RUN	SIMOTION D410-2 executes the user program and the associated system services: • Read process image of inputs. • Execution of the user programs assigned to the execution system. • Write process image of outputs. The technology packages are active in this state. They can execute commands from the user program.
1	STOPU	SIMOTION D410-2 does not execute any user program. • The technology packages are active. Test and commissioning functions can be executed. The user program is not active. • The I/O modules are in a secure state, i.e. the digital outputs have the status "LOW" and the analog outputs are at zero current/voltage.
2	STOP	SIMOTION D410-2 does not execute any user program. • It is possible to load a complete user program. • All system services (communications, etc.) are active. • The I/O modules are in a secure state, i.e. the digital outputs have the status "LOW" and the analog outputs are at zero current/voltage. • The technology packages are inactive, i.e. all enables are deleted. No axis motions can be executed.
3	MRES	Module memory reset / reset the SIMOTION D410-2 to the default settings. Using the MRES switch position, you can perform depending on the operating sequence • Memory reset of the SIMOTION D410-2 or • Restore the SIMOTION D410-2 to the default settings. For additional details on the operating sequence, see the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

Note

In the "RUN" setting, you can also control the SIMOTION D410-2 operating mode from the SIMOTION SCOUT engineering system. This means that it is not necessary to adjust the mode switch to change the operating mode.

Additional references

Detailed information

- For information on setting the operating modes, see the *SIMOTION SCOUT* Configuration Manual.
- For device upgrade (device update tool), see *Upgrading SIMOTION Devices* Operating Instructions.

DIAG button

Layout

The DIAG button is located on the SIMOTION D410-2 behind the blanking cover on the front.

Figure 10-281 DIAG button

Function

The diagnostic data and non-volatile SIMOTION data is backed up on the CompactFlash card via the DIAG button. The DIAG button function therefore corresponds to the function of switch position "D" of the Service selection switch.

The following options are available to backup the diagnostic data and the non-volatile SIMOTION data:

- Backup during operation (in STOP/STOPU/RUN operating state)
  A short pressing of the DIAG button suffices to initiate the backup of the data. The DIAG button is therefore preferable to switch position "D" of the Service selection switch.
- Backup during the module startup
  The DIAG button must be kept pressed until the boot process has completed. Since this can take between 20 and 30 seconds, the switch position "D" of the Service selector switch is preferable in this case.

Additional references

For detailed information on creating diagnostic data and backing up / restoring non-volatile SIMOTION data, refer to the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

RESET button

Layout

The RESET button is located behind the blanking cover on the SIMOTION D410-2.

Figure 10-282 RESET button

Function

The entire system is reset when the RESET button is pressed and a new power-up of the system forced.

Switch S5.0

Layout

SIMOTION D410-2 provides the S5.0 switch behind the blanking cover in the lower area of the front panel.

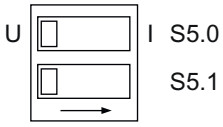


Figure 10-283 Switch S5.0

Function

The DIP switch is used for switching the analog input (X131 connector) as voltage or current input.

Table 10-238 S5.0 switch positions

Position	Function
U	The analog input is used as voltage input.
I	The analog input is used as current input.

See also

Interface assignment (Page 7759)

SIMOTION CompactFlash card

Function

The SIMOTION Kernel (SIMOTION D410-2 firmware) and the software used to control the drives (SINAMICS firmware) are contained on the CF card.

The CompactFlash card (CF card) is used to

- Backup technology packages and user data
- Update (e.g. SIMOTION firmware update)

Slot for CompactFlash card

The CF card is inserted into the plug-in module over the blanking cover (see Overview of operator control and display elements (Page 7740)).

NOTICE
Impermissible use of the CompactFlash card
The CF card of the SIMOTION D410-2 must not be used in a SIMOTION D410, D4x5 or D4x5-2!

NOTICE
Elektrostatisch gefährdete Bauelemente
The CompactFlash card is an ESD-sensitive component.
De-energize the SIMOTION D410-2 device before inserting or removing the CompactFlash card. The SIMOTION D410-2 is in a de-energized state when all the LEDs are OFF.
Comply with the ESD rules.

Additional information

For more information on writing and formatting the CF card, see the *SIMOTION D410-2* Commissioning Manual.

Error and status displays

Arrangement of LED displays

The LED displays are located next to the CompactFlash card plug-in slot on the SIMOTION D410-2.

Figure 10-284 LED displays: D410-2 DP (on the left), D410-2 DP/PN (on the right)

Meaning of the LED displays

This table describes the LEDs and their meaning.

Table 10-239 Error and status displays

LED		Meaning
D410-2 DP	**D410-2 DP/PN**	
RDY	RDY	Status indicator of the SINAMICS Integrated
RUN/STOP	RUN/STOP	SIMOTION D410-2 operating states
OUT>5V	–	Encoder current supply > 5 V (TTL/HTL)
–	OUT>5V/SY	Encoder current supply > 5 V (TTL/HTL) Synchronization status (SY) of the onboard PROFINET IO interface (X150)
SF/BF:	SF/BF:	Group error / bus fault

Additional information

You can perform a detailed diagnosis with a PG/PC and the engineering system. For information about diagnostics using LED displays, refer to the *SIMOTION D410-2* Commissioning and Hardware Installation Manual, Section "Diagnostics using LED displays".

10.2.2.4 Interfaces

Overview of interfaces

This section describes the interfaces of the SIMOTION D410-2. Information on the arrangement of the interfaces on the module can be found in Sections SIMOTION D410-2 DP representation (Page 7731) and SIMOTION D410-2 DP/PN drawing (Page 7733).

Available interfaces

Table 10-240 Overview of available SIMOTION D410-2 interfaces

Interface	Type	Connector type
X100	DRIVE-CLiQ interface (DQ)	8-pin RJ45plus socket to connect DRIVE-CLiQ nodes
X21	PROFIBUS DP/MPI interface	9-pin SUB-D socket to connect to PROFIBUS DP or MPI
X24	PROFIBUS DP interface (SIMOTION D410-2 DP only)	9-pin SUB-D socket to connect to PROFIBUS DP
X150 P1, P2	PROFINET IO interface (SIMOTION D410-2 DP/PN only)	8-pin RJ45 socket to connect to PROFINET IO
X23	Encoder interface (ENC)	15-pin SUB-D socket for connecting HTL, TTL and SSI encoders
X120	Temperature sensor connection, fail-safe digital inputs	12-pin spring-loaded terminal
X121	Isolated digital inputs, fast digital I/Os	12-pin spring-loaded terminal
X130	Isolated digital input, fail-safe digital output	8-pin spring-loaded terminal
X131	Fast digital I/Os, analog input	8-pin spring-loaded terminal
X124	Power supply connection	4-pin screw-type terminal connection
X127 P1	Ethernet interface (PN/IE)	8-pin RJ45 socket for Ethernet connection (LAN)
T0, T1, T2 and G	Measuring sockets	Sockets to output analog signals
PM-IF	Power module interface	8-pin direct connector to connect to a blocksize power module

Non-usable interfaces

Table 10-241 Overview of non-usable SIMOTION D410-2 interfaces

Interface	Designation	Connector type
Interface for BOP	BOP	8-pin multipoint connector

DRIVE-CLiQ interface

Properties

DRIVE-CLiQ has the following properties:

- Automatic detection of components by the Control Unit
- Independent expansion of components possible
- Standardized interfaces to all components
- Uniform diagnostics down to the components
- Complete service down to the components
- Mechanical parts are easy to use

For the DRIVE-CLiQ interface, 24 V / 450 mA are available to connect encoders and measuring systems.

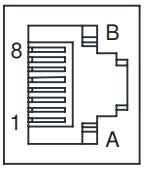
Interface characteristics

Table 10-242 Interface X100

Characteristic	Type
Connector type	DRIVE-CLiQ connector
Maximum cable length	100 m
Data rate	100 Mbits
Connector type: RJ45 socket; blanking cover for DRIVE-CLiQ interface included in the scope of delivery; blanking cover (50 pieces) Article number: 6SL3066-4CA00-0AA0)	

Interface assignment

Table 10-243 Interface assignment X100

Representation	Pin	Name	Signal type	Description
	1	TXP	O	Transmit data +
	2	TXN	O	Transmit data -
	3	RXP	I	Receive data +
	4	Reserved, do not use	-	-
	5	Reserved, do not use	-	-
	6	RXN	I	Receive data -
	7	Reserved, do not use	-	-
	8	Reserved, do not use	-	-
	A	+ (24 V)	VO	Power supply
	B	M (0 V)	VO	Electronic ground
Signal type: I = Input; O = Output; VO = Voltage Output				

Connectable devices

The following table contains the components that can communicate with SIMOTION D410-2 via the DRIVE-CLiQ interface. Note the max. number of nodes that can be connected to the DRIVE-CLiQ!

Note

Note also the topology rules of the SINAMICS S120, see *SINAMICS S120* Function Manual, Chapter "Rules for wiring with DRIVE-CLiQ".

Table 10-244 DRIVE-CLiQ connection topology

Component	Max. number of connectable nodes
Drive	Max. 1 drive from the following: • Blocksize Power Module (D410-2 directly snapped on) • Blocksize Power Module (D410-2 issued via CUA31/CUA32) • Chassis Power Module AC/AC
Motors with DRIVE-CLiQ interface, DRIVE-CLiQ encoder and SMx Sensor Modules	Max. 5 encoder systems via DRIVE-CLiQ: • Sensor Modules (SMx) for transferring an encoder signal to DRIVE-CLiQ • Encoders with DRIVE-CLiQ interface • Motors with DRIVE-CLiQ interface You require a DRIVE-CLiQ hub module (DMC20/DME20) or a CUA32 to connect more than one encoder system via DRIVE-CLiQ.
Terminal expansion modules	Max. 8 Terminal Modules (TM), of which • Up to 3 TM15, TM17 High Feature, TM41 • Up to 8 TM15 DI/DO, TM31, TM120, TM150 • Maximum one TM54F
DRIVE-CLiQ hub module 20 (DMC20/DME20)	Max. 1 DMC20 or DME20 **Note:** Because an SMx Sensor Module and a motor with a DRIVE-CLiQ interface have only one DRIVE-CLiQ interface, a DMC20/DME20 must be used with a second encoder on the DRIVE-CLiQ. If a CUA31/CUA32 is used, the DMC20/DME20 is not required. Alternatively, a second encoder can also be connected via the X23 encoder interface.

Additional information

For information on the components that can be connected via DRIVE-CLiQ (setup, connection, configuration, etc.) see

- *SINAMICS S120 Control Units and Additional System Components* Manual
- *SINAMICS S120 for AC Drives* Manual
- *SINAMICS S120* Commissioning Manual
- *SINAMICS S120 Safety Integrated* Function Manual

- *TM15/TM17 High Feature SIMOTION Terminal Modules* Commissioning Manual
- *SIMOTION Terminal Modules TM15 / TM17* Manual

PROFIBUS DP interfaces

Properties

SIMOTION D410-2 provides the following interfaces for connection on the PROFIBUS DP:

Table 10-245 SIMOTION D410-2 PROFIBUS interfaces

	D410-2 DP	D410-2 DP/PN
PROFIBUS DP/MPI interface	X21	X21
PROFIBUS DP interface	X24	–

The interfaces can be run asynchronously or isochronously, equidistant.

SIMOTION D410-2 includes master and I-slave functionality.

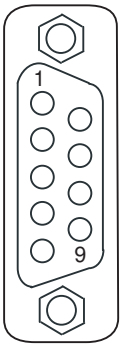
Interface characteristics

Table 10-246 X21 and X24 interfaces

Characteristics	Type
Connector type	9-pin sub D socket
Cable type	PROFIBUS cable
Max. cable length	100 m at 12 Mbit/s
Max. data rate	12 Mbit/s

Interface assignment X21

Table 10-247 PROFIBUS DP/MPI interface (X21)

Representation	Pin	Signal name	Signal type	Meaning
	1	-	-	Reserved, do not use
	2	M	VO	Ground to P24_SERV
	3	1RS_DP	B	RS-485 differential signal
	4	1RTS_DP	O	Request to send
	5	1M	VO	Ground to 1P5
	6	1P5	VO	5 V power supply for bus terminal, external, short-circuit proof
	7	P24_SERV	VO	24 V for teleservice, short-circuit proof, 150 mA maximum
	8	1XRS_DP	B	RS-485 differential signal
	9	-	-	Reserved, do not use

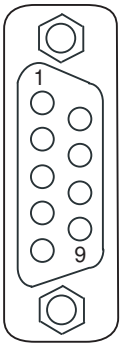
The 1P5 voltage is provided exclusively for the bus terminal.

No OLPs are permitted.

Signal type: VO = Voltage output (power supply); O = Output; B = Bidirectional

X24 interface assignment

Table 10-248 PROFIBUS DP interface X24 (SIMOTION D410-2 DP only)

Representation	Pin	Signal name	Signal type	Meaning
	1	--	--	Reserved, do not use
	2	M	VO	Ground to P24_SERV
	3	2RS_DP	B	RS-485 differential signal
	4	2RTS_DP	O	Request to send
	5	1M	VO	Ground to 1P5
	6	1P5	VO	5 V power supply for bus terminal, external, short-circuit proof
	7	P24_SERV	VO	24 V for teleservice, short-circuit proof, 150 mA maximum
	8	2XRS_DP	B	RS-485 differential signal
	9	--	--	Reserved, do not use

The 1P5 voltage is provided exclusively for the bus terminal.

No OLPs are permitted.

Signal type: VO = Voltage output (power supply); O = Output; B = Bidirectional

Connectable devices

The following devices can be connected to the PROFIBUS DP interfaces:

- PG/PC
- SIMATIC HMI devices

- SIMATIC controllers with PROFIBUS DP interface
- Distributed I/O
- Teleservice adapter
- Drive units with PROFIBUS DP interface (standard slaves)

Note

For remote diagnosis, a teleservice adapter can be connected to the PROFIBUS X21 or X24 interface. A teleservice adapter can only be connected to one of the two interfaces.

The power supply for the teleservice adapter (terminals 2 and 7) can accept current loads as high as 150 mA and is sustained short-circuit proof.

PROFINET IO interface (SIMOTION D410-2 DP/PN only)

Properties

PROFINET is an open component-based industrial communication system using Ethernet for distributed automation systems.

SIMOTION D410-2 DP/PN has a PROFINET interface with two ports (X150 P1-P2) onboard. The PROFINET interface supports operation of a SIMOTION D410-2 DP/PN as an IO controller and/or as an I device.

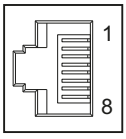
Interface characteristics

Table 10-249 Ports X150 P1 to P2

Characteristic	Type
Connector type	RJ45 socket connector
Cable type	PROFINET
Maximum cable length	100 m
Minimum send cycle clock	0.25 ms
Autocrossing	Yes i.e. both crossed and uncrossed cables can be used
Dust protection filler plugs for sealing unused PROFINET ports	One filler plug contained in the D410-2 scope of delivery Filler plugs (50 pcs) article number: 6SL3066-4CA00-0AA0

Interface assignment

Table 10-250 Assignment of the ports X150 P1 to P2

Representation	Pin	Name	Description
	1	TXP	Send data +
	2	TXN	Send data -
	3	RXP	Receive data +
	4	-	Reserved, do not use
	5	-	Reserved, do not use
	6	RXN	Receive data -
	7	-	Reserved, do not use
	8	-	Reserved, do not use

LED displays of the PROFINET interface

The PROFINET ports X150 P1 and P2 have two integrated LEDs each for displaying link and activity.

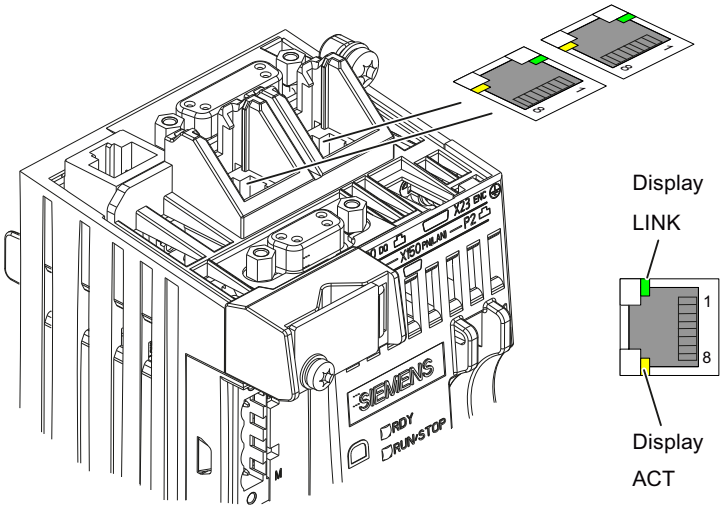


Figure 10-285 PROFINET ports of the D410-2 DP/PN

Table 10-251 State of the Link and Activity LEDs

LED	State	Meaning
LINK	OFF	No or faulty connection
	Lights up green	Transmission rate 10 or 100 Mbit/s: A different device is connected to port x and a physical connection exists
ACT	OFF	No data exchange
	Flickers yellow	Data exchange: Data is being received or sent at port x.

Connectable devices

The following devices can be connected to the PROFINET IO interface:

- PG/PC programming devices (communication with SIMOTION SCOUT / STEP 7)
- SIMATIC HMI devices
- SIMATIC controllers with PROFINET interface
- Distributed I/O
- Drive units with PROFINET IO interface (standard devices)

The SIMOTION D410-2 DP/PN then assumes the role of a PROFINET IO controller and can offer the following functions:

- PROFINET IO controller, I-device (also controller and device simultaneously)
- Supports real-time classes of PROFINET IO:
  - RT (real-time)
  - IRT (isochronous real-time)

The following functions are also supported by Industrial Ethernet:

- Communication between SIMOTION and SIMATIC NET OPC.
  The "SIMATIC NET SOFTNET S7 (S7 OPC server)" software must be installed on the PG/PC for this function.
- Communication with other devices over TCP/IP or UDP communication
- IT communication (e.g. via SIMOTION IT OPC XML-DA)
- Communication based on OPC UA (Unified Architecture)

For further information on the software packages, see the *SIMOTION PM 21* Catalog.

Encoder interface (HTL/TTL/SSI)

The HTL/TTL/SSI encoder interface is used to connect external encoders.

Interface characteristics

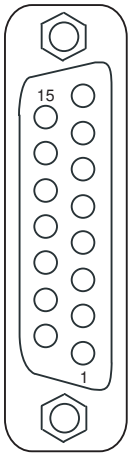
Table 10-252 Interface X23

Characteristic	Type
Encoder interface	• TTL or HTL incremental encoders (with adjustable parameters) • Absolute encoder SSI
Connector type	15-pin sub D connector
Measuring current via temperature sensor connection	2 mA

NOTICE
Destruction of the encoder electronics Operation of a 5 V encoder on the 24 V encoder supply may result in the destruction of its electronic components! Make sure that you can operate the connected encoder on a 24 V power supply (e.g. HTL encoder). This setting can be set in the expert list of the drive in parameter p0400 and in the following parameters.

Interface assignment

Table 10-253 Interface assignment X23

Representation	Pin	Name	Description
	1	+Temp	KTY, PT1000 or PTC input
	2	SSI_CLK	SSI clock positive
	3	SSI_XCLK	SSI clock negative
	4	P_Encoder 5 V / 24 V	Encoder power supply
	5	P_Encoder 5 V / 24 V	Encoder power supply
	6	P_Sense	Sense input encoder power supply
	7	G_Encoder (G)	Ground for sensor power supply
	8	-Temp (G)	Ground for KTY, PT1000 or PTC
	9	G_Sense (G)	Ground sense input
	10	RP	R track positive
	11	RN	R track negative
	12	BN	B track negative
	13	BP	B track positive
	14	AN_SSI_XDAT	A track negative / SSI data negative
	15	AP_SSI_DAT	A track positive / SSI data positive

For Pin 1 / Pin 8: The associated temperature channel (T1) can be assigned parameters as an individual channel or together in combination with the second temperature channel (T2) at interface X120. (For parameterization, see the *SINAMICS S120* Commissioning Manual).

For Pin 6 / Pin 9: At an encoder supply of 5 V, the voltage drops on the encoder supply cables are recorded and compensated by means of the sense cables. For this purpose, the sensor supply is corrected on the SIMOTION D410-2.

NOTICE

Motor overheating due to an incorrectly connected KTY temperature sensor

A sensor connected with the incorrect polarity cannot detect the motor overheating.

Connect the KTY temperature sensor with the correct polarity.

For more information on the temperature sensors and how to use them, see the *SINAMICS S120* Commissioning Manual, in Chapter Temperature sensors for SINAMICS components.

Note

There are two ways of connecting the temperature sensor:
1. Via X120, terminal 1 and 2
2. Via X23, pins 1 and 8

Table 10-254 Specification of measuring systems that can be connected

Parameter	Designation	Threshold	Min.	Type	Max.	Unit
Permissible signal level in the bipolar mode (parameter p0405.1=1); (TTL, SSI, HTL bipolar at X23) [1), 2)]	U_{diff}		2,0		V_{CC}	V
Permissible signal frequency	f_S		-		500	kHz
Required edge clearance	t_{min}		100		-	ns
Permissible zero pulse (with $T_s = 1/f_s$)	Length		¼ · T_s		¾ · T_s	
	Center of the pulse position		50	135	220	Degrees
Switching threshold in the unipolar mode (parameter p0405.0=0) and signals AN_SSI_XDAT, BN, RN at X23 connected to M_Encoder [3)]	$U_{(Schalt)}$	High [4)] (p0405.4=1)	8,4	10,6	13,1	V
		Low (p0405.4=0)	3,5	4,8	6,3	V
Switching threshold in the unipolar mode (parameter p0405.0=0) and signals AN_SSI_XDAT, BN, RN not connected to X23	$U_{(Schalt)}$	High [4)] (p0405.4=1)	9	11,3	13,8	V
		Low (p0405.4=0)	5,9	7,9	10,2	V

1) Other signal levels according to the RS422 specification.

2) The absolute level of the individual signals varies between 0 V and VCC of the measuring system.

3) See SINAMICS S120/S150 List Manual for setting the mode.

4) See SINAMICS S120/S150 List Manual for setting the threshold.

Note

We recommend that bipolar encoders are used

When using unipolar encoders, the unused negative track signals can either be connected or connected to ground. This results in different switching thresholds.

Note

Prefabricated cable for 5 V - TTL encoder

If a 5 V - TTL encoder (6FX encoder) is used, the connecting cable 6FX8002-2CR00-... must be used.

Digital I/Os / temperature sensor / analog input

Properties

The onboard I/Os (Onboard I/Os) of the SIMOTION D410-2 are assigned to the SINAMICS Integrated. An appropriate configuration allows the I/Os also to be used by SIMOTION.

Digital I/Os

The digital I/Os at the X120, X121 and X130, X131 connectors are provided for the connection of sensors and actuators.

Following types of digital I/Os are available on the SIMOTION D410-2:

- Three fail-safe electrically-isolated digital inputs (F-DI)
  (can be used alternatively as six standard digital inputs, DI 17 can also be used as EP terminal)
- One fail-safe electrically-isolated digital output (F-DO)
  (can be used alternatively as one standard digital output)
- Five electrically-isolated digital inputs (DI)
- Eight high-speed non-isolated digital I/Os (DI/DO)

Assignment of the I/Os to functions can be parameterized as required. Special functions (e.g. input of measuring input and output for output cam) can also be assigned to the I/Os.

Note

For optimal noise immunity of the digital inputs, the use of shielded cables is necessary if they are to be used as

- Inputs of measuring inputs or
- Inputs for equivalent zero mark

Analog input

The analog input at connector X131 can be parameterized as voltage or current input.

Switching between the voltage or current input using a DIP switch, refer to Switch S5.0 (Page 7745).

Interface characteristics

Table 10-255 Interface characteristics

Features		Type
Connector type (X120, X121)		12-pin spring-loaded terminal
Connector type (X130, X131)		8-pin spring-loaded terminal
Number of connectable conductors		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 1.5 mm²
	Flexible	0.2 mm² ... 1.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.25 mm² ... 1.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.25 mm² ... 0.75 mm²
	AWG / kcmil	24 ... 16
Stripped length		10 mm
Tool		Screwdriver 0.4 x 2.0 mm
Max. cable length		30 m

Note

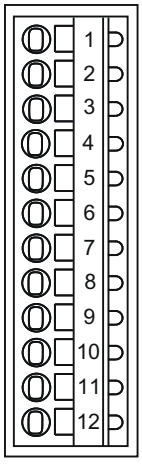
To prevent an incorrect connection, the X120, X121, X130 and X131 connectors are supplied coded. The terminal and pin numbers are also inscribed on the connectors.

Interface assignment

The following tables contain the pin assignments of the onboard I/Os.

X120

Table 10-256 Interface assignment X120

Representation	Pin	Designation		Notes
	1	+Temp		Motor temperature sensor input. Temperature sensors: KTY84–1C130 / PT1000 / PTC Measuring current via temperature sensor connection: 2 mA
	2	-Temp		
	3	F-DI 0 [2]	DI 16	Fail-safe digital input 0 or digital inputs 16 and 17 EP function (Enable Pulses) when using Safety Integrated Basic Functions via terminal
	4		DI 17+ / EP +24 V3 (Enable Pulses)	
	5		DI 17- / EP M3 (Enable Pulses) [1]	
	6	F-DI 1 [2]	DI 18	Fail-safe digital input 1 or digital inputs 18 and 19
	7		DI 19+	
	8		DI 19- [1]	
	9	F-DI 2 [2]	DI 20	Fail-safe digital input 2 or digital inputs 20 and 21
	10		DI 21+	
	11		DI 21- [1]	
	12	M1		Reference potential for: • DI 16, DI 18 and DI 20 (or F-DI 0 to F-DI 2; first shutdown path) • DO 16+ (or F-DO 0)

[1] Reference potential for DI 17+ / DI 19+ / DI 21+ (or F-DI 0 to F-DI 2; second shutdown path)

[2] Functionality depends on the parameterized Safety Integrated functions.

The functionality of the digital inputs DI 16 to DI 21 depends on the parameterized Safety Integrated functions.

Table 10-257 Safety Integrated functions using onboard terminals

Safety Integrated functions	Relevant digital inputs	
Basic Functions	Control takes place via two switch-off signal paths:	
	1st switch-off signal path	via D410-2 terminals DI 0 ... DI 3, DI 16 (X121.1...4 and X120.3) The desired input terminal is selected via BICO interconnection (BI: p9620[0]).
	2nd switch-off signal path	via EP terminal • D410-2: X120.4 and X120.5 • Power Module Blocksize with CUA3x: X210.3 and X210.4 • Power Module Chassis: X41.1 and X41.2 on the Control Interface Module (CIM)
Extended Functions	Control takes place via the 2-channel F-DI of the SIMOTION D410-2: F-DI 0 ... F-DI 2 (X120.3...11) A fail-safe digital input is made up of two digital inputs. If SIMOTION D410-2 is mounted separately (Power Module connected to SIMOTION D410-2 via CUA31/32), use of the Safety Integrated Extended and Advanced Functions via the onboard terminals (F-DI, F-DO) is not possible.	

Note

Using the "STO" safety function via the Power Module terminals for the PM240-2 (FSD, FSE and FSF). With enabled Safety Integrated functions of the D410-2, a simultaneously active STO function via Power Module terminals results in error messages being output (e.g. fault F30600, reason 1005).

- Deactivate the function "STO via Power Module terminals" by setting both the DIP switches for the interface STO_A/STO_B to the "0" position. Further information, see SINAMICS S120 Manual for AC Drives, section Power Modules Blocksize (PM240-2)

The X120 interface has a connection for the motor temperature sensor.

You can use the temperature sensor connection to connect the temperature sensing via KTY84-1C130 (special temperature sensor) or PTC (positive temperature coefficient). The temperature sensing provides thermal motor protection by detecting critical motor conditions.

For more information on "thermal motor protection", see the *SINAMICS S120* Commissioning Manual.

NOTICE

Motor overheating due to an incorrectly connected KTY temperature sensor

A sensor connected with the incorrect polarity cannot detect the motor overheating.

Connect the KTY temperature sensor with the correct polarity.

For more information on the temperature sensors and how to use them, see the *SINAMICS S120* Commissioning Manual, Section Temperature sensors for SINAMICS components.

The maximum length of the temperature sensor cable is 300 m. The cables must be shielded. For cable lengths >100 m, cables with a cross-section of ≥1 mm² must be used.

⚠ DANGER

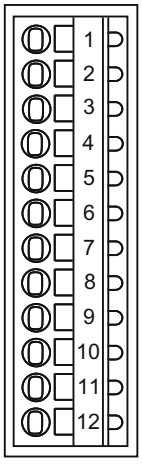
Danger to life from electric shock due to insufficient safety isolation

Only temperature sensors that meet the safety isolation specifications stipulated in EN 61800-5-1 may be connected to terminals "+Temp" and "-Temp".

Use a Sensor Module External (SME120 or SME125) or Terminal Module TM120 if safe electrical separation cannot be guaranteed (for linear motors or third-party motors, for example).

X121

Table 10-258 Interface assignment X121

Representation	Pin	Designation	Notes
	1	DI 0	Isolated digital input 0
	2	DI 1	Isolated digital input 1
	3	DI 2	Isolated digital input 2
	4	DI 3	Isolated digital input 3
	5	M2	Ground reference for DI 0 ... DI 3
	6	M	Ground reference of the electronics [1]
	7	DI/DO 8	High-speed digital I/O 8, not isolated
	8	DI/DO 9	High-speed digital I/O 9, not isolated
	9	M	Ground reference of the electronics [1]
	10	DI/DO 10	High-speed digital I/O 10, not isolated
	11	DI/DO 11	High-speed digital I/O 11, not isolated
	12	M	Ground reference of the electronics [1]

[1] Reference potential for the digital inputs/outputs and analog input

Note

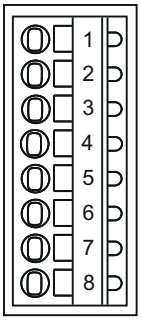
An open input is interpreted as "low".

The use of the digital inputs (DI 0 ... DI 3) requires terminal M2 be connected. This is achieved by:

- Providing the reference ground of the digital inputs, or
- a jumper to terminal M. This removes the electrical isolation for these digital inputs.

X130

Table 10-259 X130 interface assignment

Representation	Pin	Designation		Notes
	1	DI 22+		Isolated digital input 22
	2	DI 22- [2]		Reference potential for DI 22+
	3	M2		Ground reference for DI 0 ... DI 3
	4	M		Ground reference of the electronics [1]
	5	M1		Reference potential for: • DI 16, DI 18 and DI 20 (or F-DI 0 to F-DI 2; first shutdown path) • DO 16+ (or F-DO 0)
	6	24 V1		Power supply for DO 16+ (or F-DO 0)
	7	F-DO 0 [4]	DO 16+ [3]	Fail-safe digital output 0 or digital output 16
	8		DO 16- [5]	

[1] Reference potential for the digital inputs/outputs and analog input

[2] Reference potential for DI 20+

[3] The proper functioning of the DO 16 requires that the terminals 5/6 be connected.

[4] F-DO 0 for Safety Integrated Extended and Advanced Functions

[5] For applications "without safety function", DO 16- does not function

Note

An open input is interpreted as "low".

Note

If M1 or M2 is connected with M, the electrical isolation no longer exists.

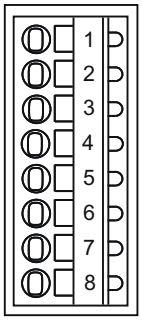
If the digital output (DO 16) cannot be set by the signal source despite being set, an alarm A03507 (digital output not set) is issued. This alarm can also parameterized as a fault.
The possible cause may be missing power supply (X130.6) or a short-circuit. For further details, see the *SINAMICS S120/S150 List Manual*, Section List of faults and alarms.

Note

If the 24 V supply is briefly interrupted, the digital output is deactivated until the interruption has been rectified.

X131

Table 10-260 X131 interface assignment

Representation	Pin	Designation	Notes
	1	DI/DO 12	High-speed digital I/O 12, not isolated
	2	DI/DO 13	High-speed digital I/O 13, not isolated
	3	M	Ground reference of the electronics [1]
	4	DI/DO 14	High-speed digital I/O 14, not isolated
	5	DI/DO 15	High-speed digital I/O 15, not isolated
	6	M	Ground reference of the electronics [1]
	7	AI 0+	Analog voltage or current input
	8	AI 0-	

[1] Reference potential for the digital inputs/outputs and analog input

NOTICE

Incorrect results of analog-to-digital conversion due to impermissible input voltage

The common mode range must not be violated.

Makes sure that the analog input voltage signals can have a maximum voltage of ±12 V with respect to the reference potential. If the range is infringed, incorrect results may occur during analog/digital conversion.

Note

A 24 V supply voltage must be connected to terminal X124 for the digital outputs to be used.

If momentary interruptions in the voltage occur in the 24 V supply, the digital outputs are deactivated until the interruption has been rectified.

If a digital output (DO 8 ... DO 15) is parameterized and the external 24 V power supply is not connected (or the level is too low), the alarm A03506 will be issued. This alarm can also be parameterized as a fault.
For further details, see the *SINAMICS S120/S150 List Manual*, Section List of faults and alarms.

See also

SIMOTION D410-2 DP connection examples (Page 7769)

SIMOTION D410-2 DP/PN connection examples (Page 7773)

Use of the interfaces

Fail-safe digital I/Os (F-DI/F-DO)

The SIMOTION D410-2 provides three fail-safe isolated digital inputs (F-DI) and one fail-safe isolated digital output (F-DO):

- An F-DI consists of a digital input and a second digital input to which the cathode of the optocoupler is connected.
  Each of the F-DIs can also be used as two standard digital inputs, e.g. the use of the F-DI 0 as DI 16 and DI 17.
- The F-DO 0 can also used as a standard digital output.
  The F-DO 0 consists of a high-side switch and a low-side switch. For applications without the safety function, the high-side switch may be used as an additional digital output. The low-side switch is not available.

Note

The following safety functions are available for SIMOTION D410-2:

- Safety Integrated Basic Functions via the EP terminals
- Safety Integrated Extended and Advanced Functions via onboard I/Os (3 F-DI and 1 F-DO)
- Safety Integrated Extended and Advanced Functions with TM54F
- Safety Integrated Basic and Extended/Advanced Functions via secure communication from "PROFIsafe on PROFIBUS"
  Control (F logic) is via a SIMATIC F-CPU which is connected to PROFIsafe via PROFIBUS (not SCOUT TIA).
- Safety Integrated Basic and Extended/Advanced Functions via secure communication from "PROFIsafe on PROFINET" (only for SIMOTION D410-2 DP/PN)
  Control (F logic) is via a SIMATIC F-CPU which is connected to PROFIsafe via PROFINET (e.g. an S7-1500 F-CPU).

For further information on Safety Integrated, see the *SINAMICS S120 Safety Integrated* Function Manual.

EP terminal

The pulse inhibit function (EP) is only available when Integrated Basic Functions are enabled.

Digital inputs (DI)

The SIMOTION D410-2 provides five digital inputs (DI).

The electrically isolated inputs can be used as freely addressable inputs.

Note

An open input is interpreted as "low".

For the DI 22 digital input to function correctly, the coupled reference potential (DI 22-) must be connected. The following options are available:

- Connect the coupled reference potential of the digital input to M1, M2 or M.
  This assigns the input to the potential of the associated pin.
- Create a bridge between terminal M and terminal M1 or M2.
  Caution: This will cancel the electrical isolation for this digital input!

Bidirectional digital I/Os (DI/DO)

The SIMOTION D410-2 provides eight bidirectional digital I/Os (DI/DO) that can be parameterized channel-specific as digital input or output.

This produces the following usage options for the parameterization of the DI/DO:

Table 10-261 DI/DO usage options

DI/DO	Interface	Use
Parameterization of the DI/DO as digital inputs:		
DI/DO 8 to DI/DO 15	X121, X131	"Fast inputs" for measuring inputs [1] or homing inputs
DI/DO 8 to DI/DO 15	X121, X131	Freely addressable inputs
Parameterization of the DI/DO as digital outputs:		
DI/DO 8 to DI/DO 15	X121, X131	"High-speed" outputs of output cams
DI/DO 8 to DI/DO 15	X121, X131	Freely addressable outputs

[1] With a signal edge at the relevant input, the current actual values of one or more encoders are measured with positioning accuracy to determine lengths and distances. The assignment of the inputs is not fixed, and the special use is activated in the SIMOTION SCOUT engineering system.

Note

An additional external electronics power supply via terminal X124 is required in two cases:

- If the digital outputs DO 8 to DO 15 are in use, the power supply needs to be connected to X124.
- The Power Module provides the electronics power supply of the SIMOTION D410-2. If the SIMOTION D410-2 needs to remain functional when the Power Module is switched off, X124 must be used for the electronics power supply.

Inductive loads

If the digital inputs are used together with inductive loads, it must be ensured that the permissible voltage range of -30 V to +30 V for digital inputs is not exceeded by the switching overvoltage of inductive loads. This is preferably achieved by using auxiliary contacts on the contactor/relay for the digital inputs.

If no auxiliary contacts are available, overvoltage protection must be used. Protective circuits with a freewheeling diode or a diode combination with a Z-diode are possible. RC elements and varistors are not suitable due to the remaining residual voltages.

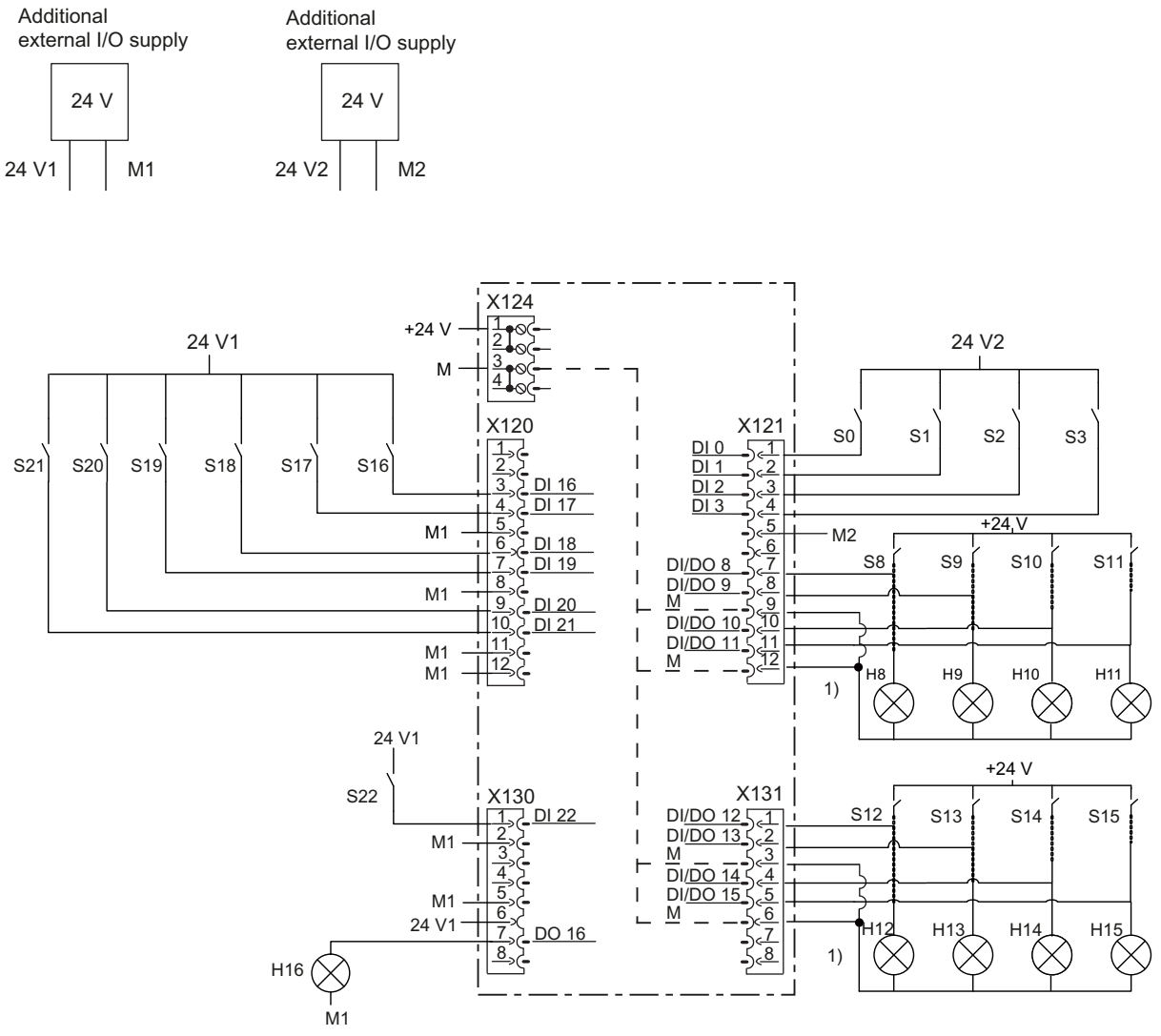
Additional references

For information on configuring the digital I/Os as freely addressable I/Os, inputs of measuring inputs or outputs of output cams, see the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

For information on the configuration and function of the measuring input and output cam technology objects, refer to the *SIMOTION Output Cams and Measuring Inputs* Function Manual.

SIMOTION D410-2 DP connection examples

Connection examples without Safety Integrated functions



1) X121: Pin 7/8, 10/11 and X131: Pin 1/2, 4/5 can be parameterized as input or output.

Figure 10-286 Example of circuits for the DI/DO without Safety Integrated functions

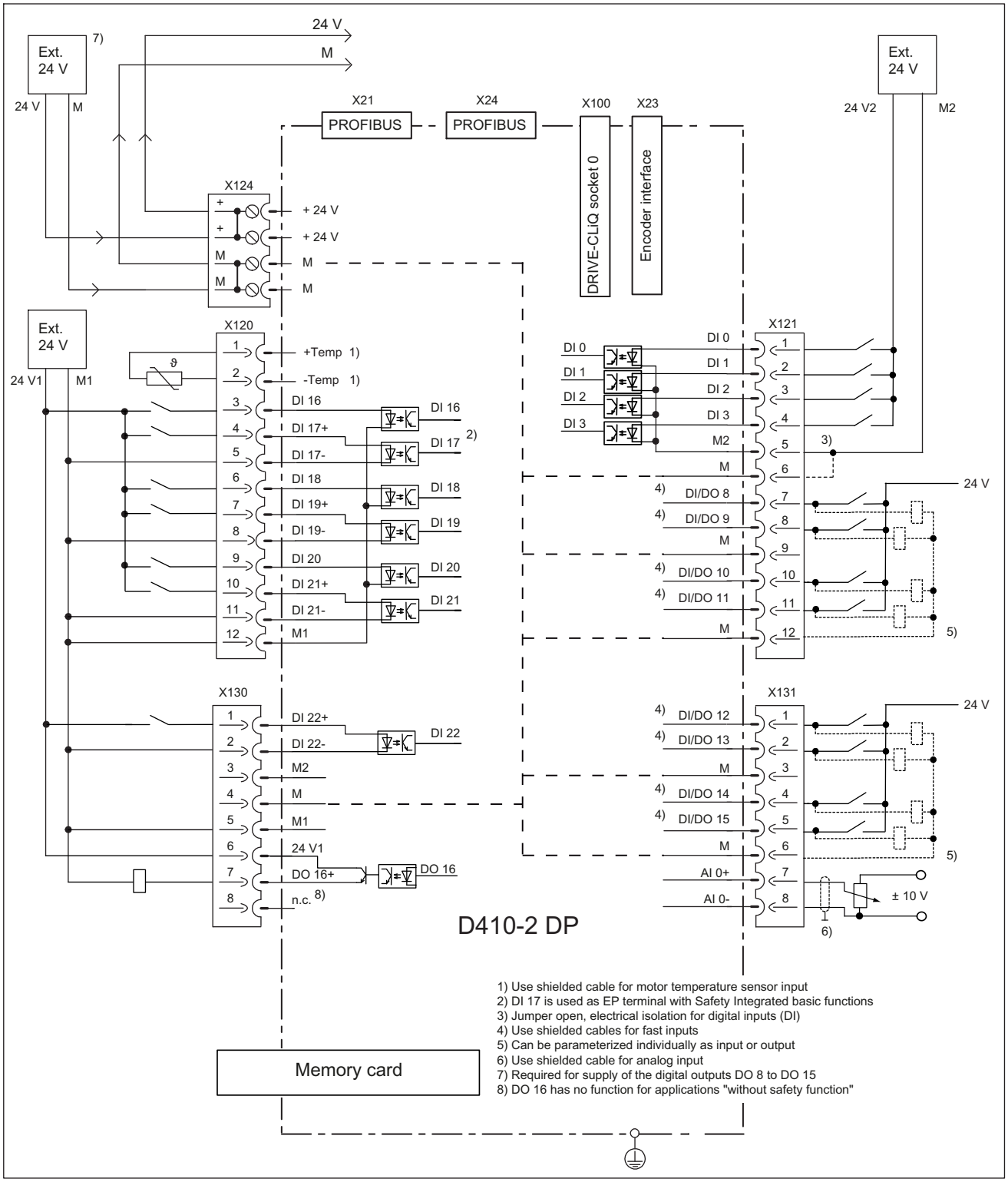


Figure 10-287 Connection example of SIMOTION D410-2 DP without Safety Integrated functions

Connection examples with Safety Integrated Extended and Advanced Functions

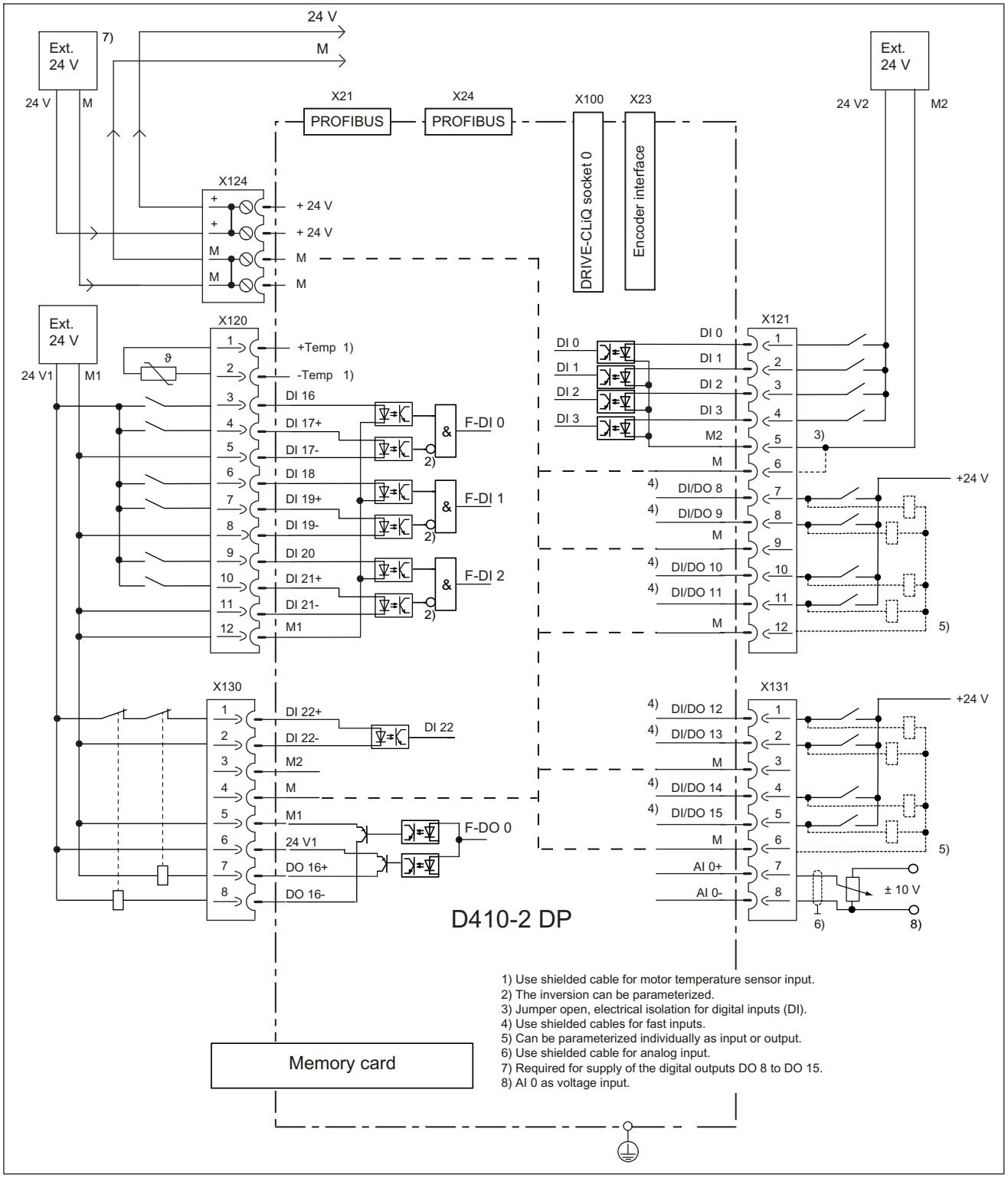
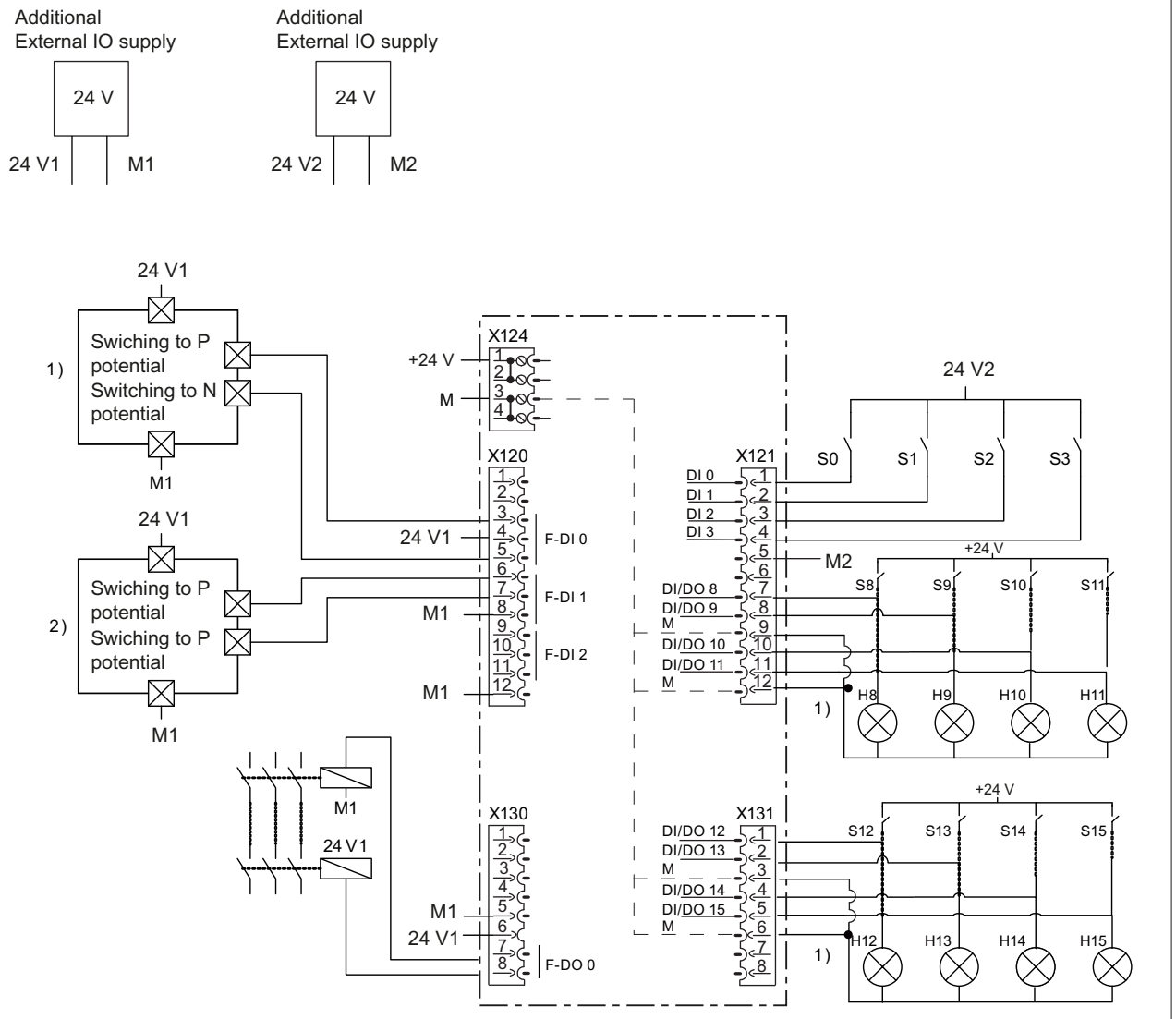


Figure 10-288 Connection example of SIMOTION D410-2 DP with Safety Integrated Extended and Advanced Functions

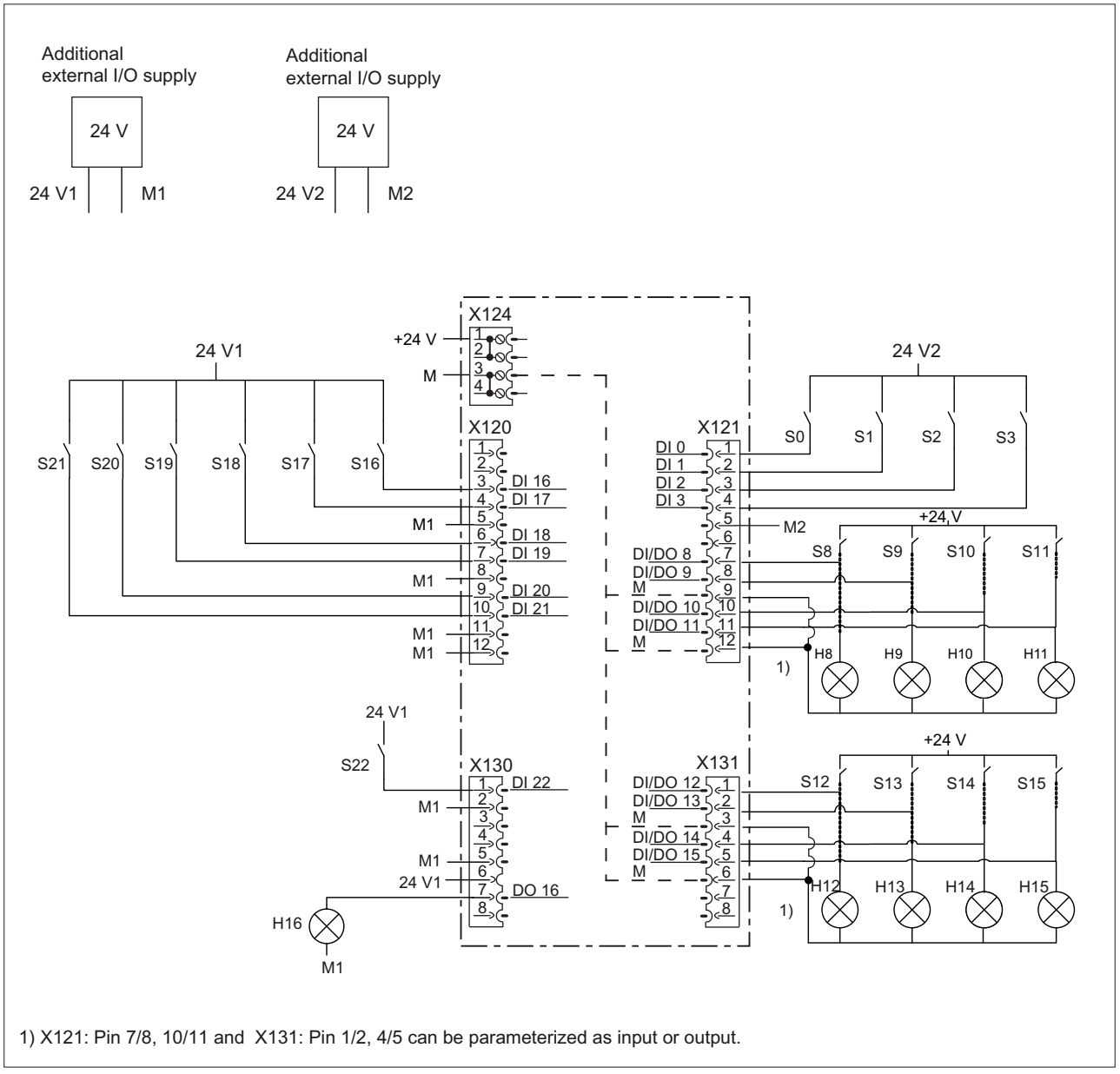


1) Example of connecting a safety controller with outputs switching to P and N potential
2) Example of connecting a safety controller with two outputs switching to P potential

Figure 10-289 Example of circuits for the F-DI and F-DO with Safety Integrated Extended and Advanced Functions

SIMOTION D410-2 DP/PN connection examples

Connection examples without Safety Integrated Extended and Advanced Functions



1) X121: Pin 7/8, 10/11 and X131: Pin 1/2, 4/5 can be parameterized as input or output.

Figure 10-290 Example of circuits for the DI/DO without Safety Integrated Extended and Advanced Functions

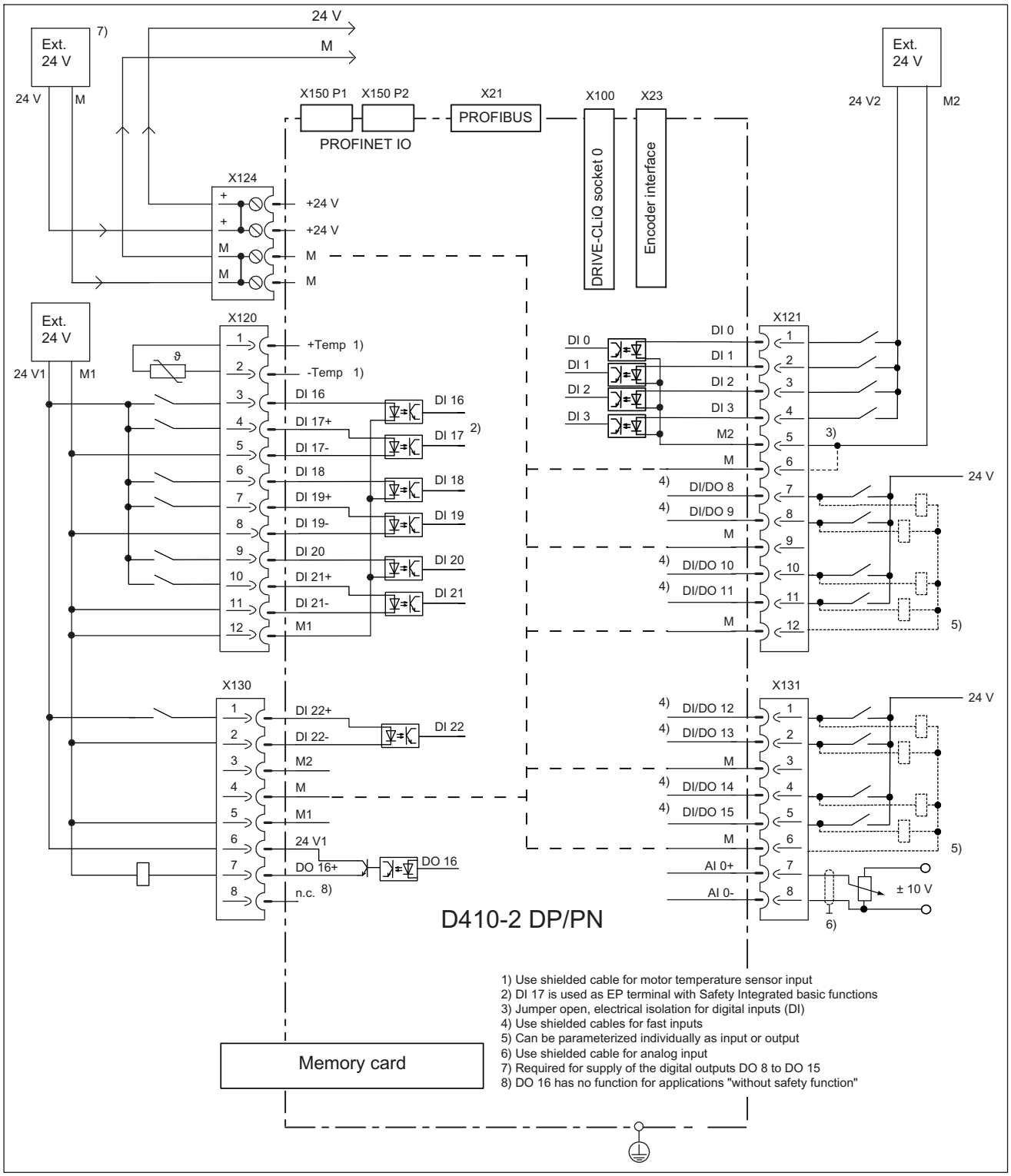


Figure 10-291 Connection example of SIMOTION D410-2 DP/PN without Safety Integrated Extended and Advanced Functions

Connection examples with Safety Integrated Extended and Advanced Functions

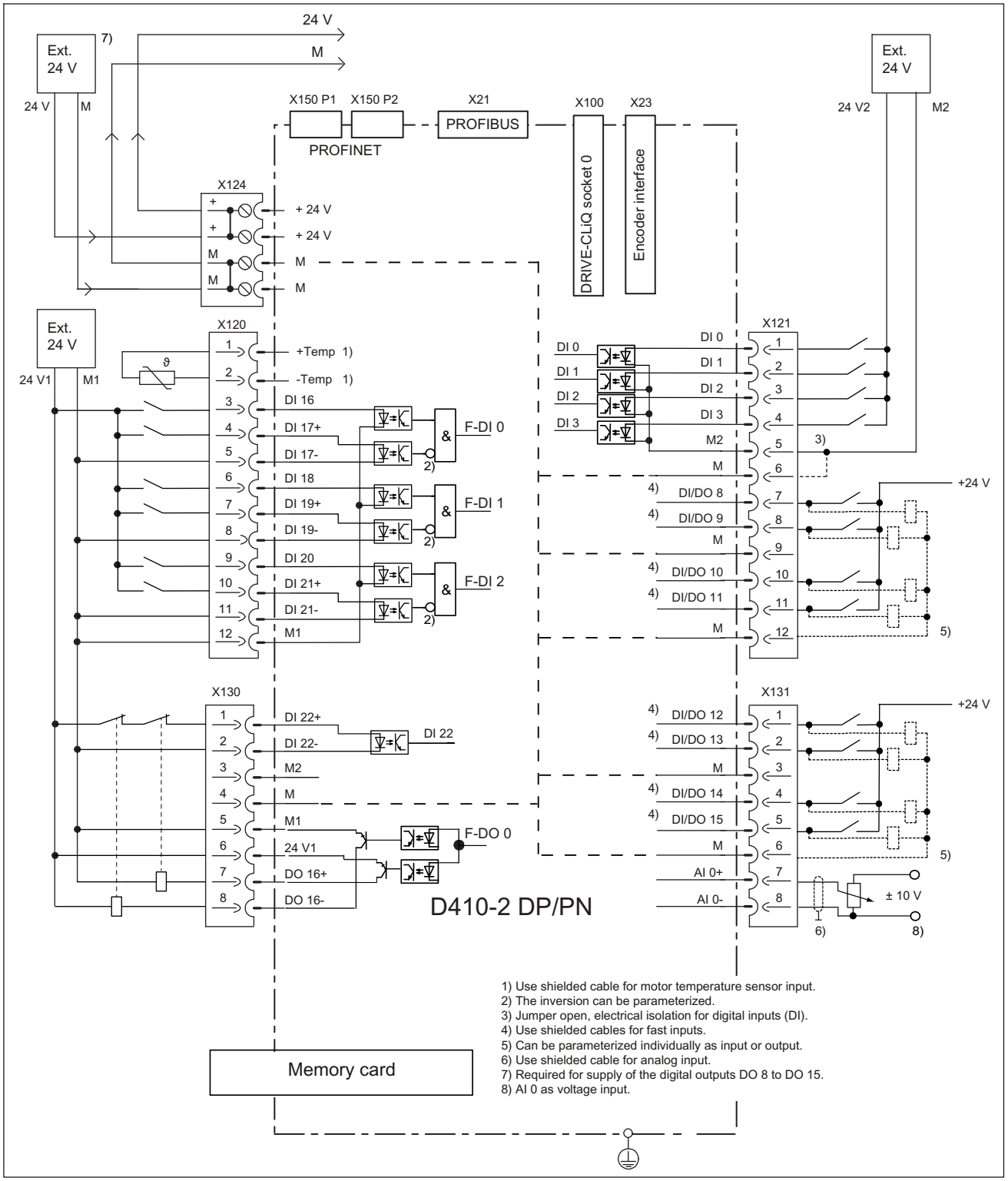
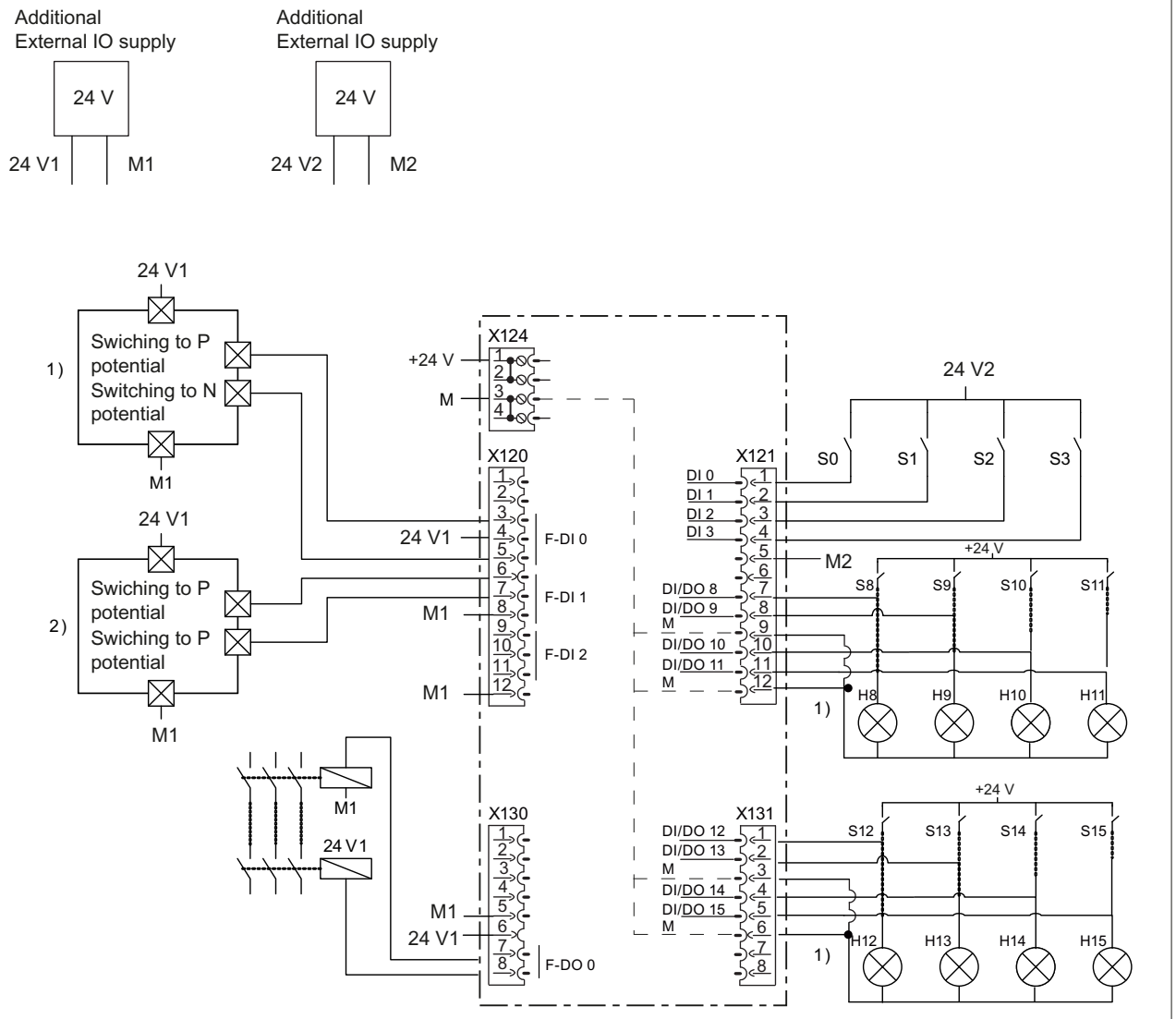


Figure 10-292 Connection example of SIMOTION D410-2 DP/PN with Safety Integrated Extended and Advanced Functions



1) Example of connecting a safety controller with outputs switching to P and N potential
2) Example of connecting a safety controller with two outputs switching to P potential

Figure 10-293 Example of circuits for the F-DI and F-DO with Safety Integrated Extended and Advanced Functions

Power supply

The X124 interface is provided for connection of the external power supply.

Note

When using external power supplies (e.g. SITOP), the ground potential must be connected with the protective ground terminal (PELV).

Note

Ground potential and housing (PE) are connected internally with low impedance.

Note

The 24 V DC cable must be approved for temperatures of up to 75 °C.

The maximum permissible cable length is 10 m.

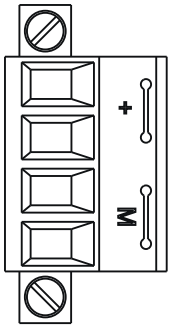
Interface characteristics

Table 10-262 Interface X124

Features		Type
Connector type		4-way screw-type terminal
Number of connectable conductors		1
Connectable cable types and conductor cross-sections		
	Rigid	0.2 mm² ... 2.5 mm²
	Flexible	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule without plastic sleeve	0.2 mm² ... 2.5 mm²
	Flexible, with wire-end ferrule with plastic sleeve	0.2 mm² ... 1.5 mm²
	AWG / kcmil	22 ... 12
Stripped length		6 ... 7 mm
Tool		Screwdriver 0.5 x 3 mm (M2.5)
Tightening torque		0.4 to 0.5 Nm (3.5 ... 4.4 lbf in)
Max. current carrying capacity, incl. loop-through		20 A (15 A per UL/CSA)
Max. cable length		10 m

Interface assignments

Table 10-263 Power supply X124

Representation	Terminal	Designation
	+	Electronic power supply
	+	Electronic power supply
	G	Electronic ground
	G	Electronic ground

Note

The power supply terminal strip must be screwed on tightly using a flat-bladed screwdriver.

Note

The 24 V is looped through via the 24 V connector. In this case, pin 1 is jumpered with pin 2, and pin 3 is jumpered with pin 4 in the connector. The maximum current can be limited through the current carrying capacity of the cable. The current carrying capacity of the cable depends, for example, on the type of cable installation (cable duct, laying on a cable rack, etc.).

Disconnection of 24 V plug-in connections during operation

Observe the following safety notice when using SIMOTION D410-2:

⚠ WARNING

Personal injury and damage to property can occur

Personal injury and property damage can occur if 24 V plug-in connections are disconnected during operation. Disconnection of 24 V plug-in connections is only permitted when the power is off.

Ethernet interface

Properties

SIMOTION D410-2 has an X127 interface for connection to Industrial Ethernet.
Industrial Ethernet is a communication network with a transmission rate of 10/100 Mbit/s.

SIMOTION D410-2 offers the following functions via the Ethernet interface:

- PROFINET basic services (e.g. DCP, LLDP, SNMP)
  Although these PROFINET basic services provide uniform functions for the address assignment and diagnostics, they do not provide PROFINET IO communication for the connection of drives or I/O modules, for example.
- Communication with STEP 7 and SIMOTION SCOUT
- Communication between SIMOTION and SIMATIC NET OPC
  The "SIMATIC NET SOFTNET-S7 (S7-OPC server)" software must be installed on the PG/PC for this function.
- Connection of HMI systems
- Communication with other devices over TCP/IP or UDP communication
- IT communication (e.g. via SIMOTION IT OPC XML-DA)

For further information on the software packages, see the *SIMOTION PM 21* Catalog.

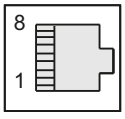
Interface characteristics

Table 10-264 X127 interface

Characteristic	Type
Connector type	RJ45 socket connector
Cable type	Industrial Ethernet cable
Maximum cable length	100 m
Autocrossing	Yes, i.e. both crossed and uncrossed cables can be used
Blanking cover for Ethernet interface included in the scope of delivery; blanking cover (50 units), article number: 6SL3066-4CA00-0AA0	

Interface assignment

Table 10-265 X127 interface assignment

Representation	Pin	Signal name	Signal type	Meaning
	1	TXP	Output	Ethernet send differential signal
	2	TXN	Output	Ethernet send differential signal
	3	RXP	Input	Ethernet receive differential signal
	4	-	-	Reserved, do not use
	5	-	-	Reserved, do not use
	6	RXN	Input	Ethernet receive differential signal
	7	-	-	Reserved, do not use
	8	-	-	Reserved, do not use

Note

The MAC address is on a printed label visible from the front.

Measuring sockets

Properties

The T0 - T2 measuring sockets are used to output analog signals. Any interconnectable signal can be output via SINAMICS on every measuring socket on the control unit.

Note

The measuring sockets are suited for multiple-spring wire connectors with a diameter of 2 mm.

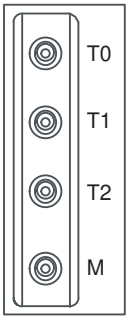
Interface characteristics

Table 10-266 T0, T1 and T2 interfaces

Characteristic	Type
Connector type	2 mm sockets
Voltage	0 V to 5 V
Resolution	8-bit
Load current	max. 3 mA
Sustained short-circuit proof Reference potential is M terminal	

Interface assignment

Table 10-267 Interface assignments T0, T1 and T2

Representation	Pin	Designation
	T0	Measuring socket 0
	T1	Measuring socket 1
	T2	Measuring socket 2
	M	Ground

Note

The test sockets are provided as a support to commissioning and diagnostics; they must not be connected for normal operation.

Power Module Interface

SIMOTION D410-2 can be connected to a SINAMICS Power Module in blocksize format via the Power Module interface.

Power Modules PM340 and PM240-2 (PM240-2 as of SIMOTION V4.4/SINAMICS V4.7) can be used.

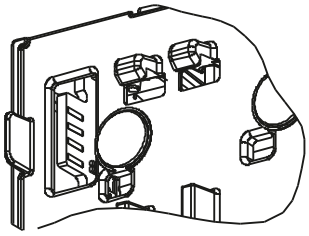


Figure 10-294 SIMOTION D410-2 interface to the Power Module interface (PM-IF)

Note

SIMOTION D410-2 can only be connected to a SINAMICS Power Module in blocksize format via the Power Module interface. A Power Module in chassis format must be connected via the DRIVE-CLiQ interface (see *SIMOTION D410-2* Commissioning and Hardware Installation Manual).

A SIMOTION D410-2 cannot be operated with SINAMICS G120 (PM2x0) Power Modules. Booksize Motor Modules can likewise not be connected.

10.2.2.5 Technical data

Shipping and storage conditions

The following conditions apply to modules that are shipped and stored in the original packaging.

Table 10-268 Shipping and storage conditions

Type of condition	Permissible range/class	
	Transport	Long-term storage
Climate class	2K4 according to EN 60721-3-2:1997	1K4 according to EN 60721-3-1:1997

Type of condition	Permissible range/class	
Vibration and shock stressing (in transport packaging)	EN 60721-3-2:1997, class 2M3	EN 60721-3-1:1997, class 1M2
Permissible ambient temperature	-40° C ...+70° C	-25° C ...+55° C
Relative humidity	5 ... 95%	10 ... 100%
Height	Max. 4000 m above sea level	
Atmospheric pressure	620 hPa ... 1060 hPa The specified values apply to a transportation altitude of up to 4000 m.	620 hPa ... 1060 hPa The specified values apply to a storage altitude of up to 4000 m.
Biological environmental conditions	Class 2B1 according to EN 60721-3-2:1997	Class 1B1 according to EN 60721-3-1:1997
Chemically active environmental conditions	Class 2C2 according to EN 60721-3-2:1997	Class 1C2 according to EN 60721-3-1:1997

Ambient conditions

Conditions of use

SIMOTION D410-2 meets the conditions of use for Class 3C3 according to DIN EN 60721-3-3:1995 (operating locations with high traffic densities and in the immediate vicinity of industrial equipment with chemical emissions).

Protect the device from environmental effects

SIMOTION D410-2 is designed for use in stationary, weather-protected locations.

Protect the device against the following environmental effects:

- Direct sunshine and heat sources
- Mechanical vibration and shock
- Dust
- Humidity
- Strong magnetic fields

Use prohibition

Without additional measures, SIMOTION D410-2 must **not** be used in

- Locations with a high percentage of ionizing radiation
- Locations with extreme operating conditions, e.g.
  - Dust accumulation
  - Corrosive vapors or gases
- Installations requiring special monitoring such as:
  - Elevator installations
  - Electrical systems in particularly hazardous rooms

An additional measure for using SIMOTION D410-2 can, for example, be installation in cabinets.

Note

The components must be protected against conductive contamination, e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12.

If conductive contamination can be excluded at the installation site, a lower degree of cabinet protection may be permitted.

See Industrial security (Page 957).

Ambient operating conditions for the operation

SIMOTION D410-2 can be used under the following ambient conditions:

Table 10-269 Environmental requirements

Ambient conditions	Fields of application	Remarks
Climatic ambient conditions		
Climate class	3K3	According to EN 60721-3:1995 Ambient conditions better than 3K3
Permissible ambient temperature	0 to +55° C, up to 2000 m above sea level	As of an altitude of 2000 m, the maximum ambient temperature decreases by 7 °C every 1000 m increase in altitude.
Maximum installation altitude	2000 m to max. 4000 m above sea level	
Relative humidity	5 to 95%	
Condensation, icing, drip, spray and splash water	Not permissible	
Atmospheric pressure	620 to 1060 hPa	Corresponding height 4000 m - 0 m above sea level.
Mechanical ambient conditions		
Pollution degree	2 according to EN 60664-1:2008	
Biological ambient conditions:	Class 3B1 according to EN 60721-3-3:1995	Mold, mold growth, slime, rodents, termites and other animal vermin are not permissible

Ambient conditions	Fields of application	Remarks
Chemically active environmental conditions	Class 3C1 according to EN 60721-3-3:1995	
Mechanically active environmental conditions	Class 3S1 according to EN 60721-3-3:1995	Conductive dusts are not permitted.
Vibratory load - D410-2 - S120 (without SME/DME)	Vibratory load during operation according to EN 60721-3-3: Class 3M1	Test values according to EN 60068-2-6 (sinusoidal) • 10 ... 57 Hz: 0.075 mm deflection amplitude • 57 ... 150 Hz: 1 g acceleration amplitude • 10 frequency cycles per axis
Shock load - D410-2 - S120 (without SME/DME)	Shock load during operation according to EN 60721-3-3: Class 3M1	Test values according to EN 60068-2-27 (half-sinusoidal) • 5 g peak acceleration, 30 ms duration • 3 shocks in all three axes in both directions

Note

Observe the fields of application of the Power Modules along with their derating. Refer to *SINAMICS S120 for AC Drives* Manual.

Vibration reduction

If SIMOTION D410-2 is subjected to larger shocks or vibrations, you must use suitable measures to reduce the acceleration or the amplitude.

We recommend installation on shock-absorbing material (e.g. rubber-metal vibration dampers).

Other data

Condition	Field of application
Protection against the ingress of foreign matter and water	IP20 degree of protection according to EN 60529
Class of protection	Class 1 (with protective conductor system) and class 3 (PELV) according to EN 61800-5-1:2007

System data, connection values, dimensions and weight

General technical data

The SIMOTION D410-2 has an integrated fan.

Table 10-270 General technical data

Dimensions and weight	
Dimensions W x H x D D410-2 DP D410-2 DP/PN	 73 x 186.8 x 74.4 mm 73 x 190.7 x 74.4 mm
Weight • Without packaging • With packaging	 • 830 g • 1,000 g
Electrical connection values	
Power supply	24 VDC (permissible range: 20.4 ... 28.8 V)
Ripple	Max. 5% at 24 VDC
Non-periodic overvoltage	Max. 35 VDC (Condition: Max. 500 ms, 50 s recovery time, max. 10 results/h)
Current consumption, typically [1)]	< 0.8 A
Starting current, typically [1)]	3.0 A
Power loss, typically [1)]	< 20 W

[1)] With no load on I/Os and no 24 V supply via DRIVE-CLiQ or PROFIBUS interface

Memory

Table 10-271 Technical data for memory

RAM (work memory)	120 MB (as of V5.3 SP1) 96 MB (V4.4 - V5.2 SP1) 48 MB (V4.3 SP1)
RAM disk (load memory)	60 MB (as of V5.3 SP1) 47 MB (V4.4 - V5.2 SP1) 31 MB (V4.3 SP1)
Retentive memory (retain variables)	At least 108 KB
Persistent memory (user data on CF card)	2 GB CF card: At least 1.5 GB 1 GB CF card: At least 300 MB
Work memory for Java applications	20 MB
Non-volatile data buffering buffer time, min.	 Unlimited (maintenance-free backup)
Memory for system data	
Diagnostic buffer (non-volatile)	100 messages (SIMOTION) 100 messages (SINAMICS Integrated)

Note

The memory sizes may be increased for the current version after the time for going to press for the documentation. The latest values can be found at Internet address (https://support.industry.siemens.com/cs/ww/en/view/18857317)

CompactFlash card

Table 10-272 CF card

Memory capacity	2 GB (article number 6AU1400-1QA20-0AA0) 1 GB (article number 6AU1400-1PA2*-0AA0)
Weight	10 g

Mounting plate

Table 10-273 Mounting plate data

Dimensions W x H x D	74.5 x 236 x 36.5 mm
Weight - Without packaging - With packaging	 380 g 450 g

Further technical data

For further technical data such as the max. number of online connections, HMI devices that can be used as well as a list of tasks available in the execution system, see the function overview in the *SIMOTION PM 21* Catalog.

Interfaces and performance features

PLC and motion control performance

Number of axes and clock cycles

Table 10-274 Maximum number of axes and minimum cycles for SIMOTION D410-2

PLC and motion control performance	
Maximum number of axes	8 axes [1)]
Minimum PROFIBUS cycle clock	0.5 ms PROFIBUS Integrated 1 ms PROFIBUS external

Minimum PROFINET send cycle clock (D410-2 DP/PN only)	0.25 ms
Minimum position control/interpolator cycle clock	0.5 ms (1 ms when using the TO axis and the integrated drive control)

1) Path interpolation is supported as of V4.4.

Integrated drive control

Table 10-275 Controls for integrated drives

Max. number of axes for integrated drive control (servo/vector/V/f)	1 / 1 / 1 (alternative) Drive control based on SINAMICS S120 CU310-2, firmware version V4.x/V5.x

Communication

Interfaces

Table 10-276 Interfaces

DRIVE-CLiQ interface	1
Ethernet interface	1
PROFIBUS interface	1 (D410-2 DP/PN) 2 (D410-2 DP) • Isochronous • Can be configured as master or slave
PROFINET interface	One interface with two ports (D410-2 DP/PN only) • Supports PROFINET IO with IRT and RT • Can be configured as PROFINET IO controller and/or device

Address space

Table 10-277 Address space

Logical I/O address space of the Control Unit		16 KB (as of V4.5) 8 KB (up to V4.4)
Physical I/O address space for each interface, one each for inputs and outputs		
	PROFIBUS Integrated	4 KB
	PROFIBUS	1 KB
	PROFINET (D410-2 DP/PN only)	6 KB (as of V4.5) 4 KB (up to V4.4)
Permanent process image for BackgroundTask (I/O variables)		64 bytes

Additional configurable process image for each cyclic task (I/O variables)		yes
Address space for each PROFIBUS DP station		244 Byte
SINAMICS Integrated (PROFIBUS Integrated) address space		512 Byte
Address space for each PROFINET device		1400 Byte
Maximum consistency range Onboard PROFINET interface X150		
	For controller-controller direct data exchange	254 bytes
	For I-device	1024 bytes (as of V4.4) 254 bytes (< V4.4)

Logical address space: Logical addresses can be assigned in STEP 7 within this space.

Physical address space: Address space with assigned data.

The address space contains all the logical addresses, i.e. also diagnostic addresses, I-device, etc.

Number of slaves/devices

Table 10-278 Number of slaves/devices

Maximum number of slaves per PROFIBUS interface	64
Maximum number of devices for PROFINET	64

Onboard I/Os

Digital inputs

Table 10-279 Technical data for digital inputs

Number of inputs	5
Input voltage • Rated value • For signal "1" • For signal "0" [2]	 • 24 VDC • 15 ... 30 V • -3 ... +5 V
Electrical isolation [1]	Yes (via optocoupler)
Current consumption typical at 1 signal level	3.5 mA at 24 V 5 mA at 24 V [3]
Input delay, typical (hardware) • Signal "0" → "1" • Signal "1" → "0"	 • 50 µs • 150 µs

Permissible quiescent current	2 mA
Protection	Protected against polarity reversal

1) Reference potential for DI 0 ... DI 3 is terminal M2 (X121, X130), reference potential for DI 22+ is terminal DI 22- (X130)

2) The digital inputs are protected against polarity reversal up to -30 V

3) Up to and including hardware version "C" for D410-2 DP or "B" for D410-2 DP/PN

Digital inputs/outputs

Table 10-280 Technical data regarding the digital inputs/outputs with adjustable parameters

Number of digital inputs/outputs	8 • Max. 8 as measuring input inputs • Max. 8 as cam outputs
If used as an input	
Input voltage • Rated value • For signal "1" • For signal "0" 2)	 • 24 VDC • 15 ... 30 V • -3 V ... +5 V
Galvanic isolation	No
Current consumption typical at 1 signal level	3.5 mA at 24 V 5 mA at 24 V 3)
Input delay, typ. (hardware): • Signal "0" → "1" • Signal "1" → "0"	 • 5 µs • 50 µs
Measuring input input, resolution	1 µs
Measuring input input, reproducibility	Typ. 5 µs
If used as an output	
Rated load voltage, permissible range	24 VDC, 20.4 ... 28.8 V
Galvanic isolation	No
Current load, max.	500 mA per output
Residual current, max.	2 mA
Output delay time, typ./max. (hardware) 1) • Signal "0" → "1" • Signal "1" → "0"	 • 150 µs / 400 µs • 75 µs / 100 µs
Output cam output, resolution	Typ. 125 µs
Output cam output, reproducibility	Typ. 125 µs

Switching frequency of the outputs, max. • With resistive load • With inductive load • With lamp load (max. 5 W)	 • 4 kHz • 0.5 Hz • 10 Hz
Protection	Short circuit, ground fault and overload proof Automatic restart after overload tripping

1) Data for: V_{cc} = 24 V; load 48 Ω; high ("1") = 90% V_{out}; low ("0") = 10% V_{out}

2) The digital inputs are protected against polarity reversal up to -30 V

3) Up to and including hardware version "C" for D410-2 DP or "B" for D410-2 DP/PN

Max. switching frequency D410-2

The max. switching frequency of the hardware depends on the load and is up to 4 kHz for an ohmic load of 0.5 A (typical value; low-high ratio = 50:50; short cable lengths).

Logic control of the digital output is also a limiting factor.

- If a DO 8...15 is set or reset via the TO cam/TO cam track, up to 2 edges are possible per servo cycle
  → with servo cycles of at least 500 µs, a max. switching frequency of 2 kHz is achieved
- if a DO 8...15 is set or reset from the user program, no more than 1 edge per servo cycle is possible.
  → with servo cycles of at least 500 µs, a max. switching frequency of 1 kHz is achieved

The max. achievable switching frequency can also be limited by the CU parameter p0799[0] (sampling time of the inputs/outputs of the CU) or p2048 (PROFIdrive PZD sampling time).

Reproducibility

The reproducibility at the measuring input input depends on the edge steepness of the measurement signal.

Generally, the following is valid: The steeper the edges of the input signal are, the easier it is to reproduce the measurement results.

Sloping signals are achieved by switching the signal level to "active". This is typically with rising edges as the signal here is "actively" switched to HIGH by the digital output (example: Output of a TM17 module: Rising edge).

Falling edges typically have less edge steepness (signal level that falls slowly), as the signal level is "not actively" forced to LOW (example: Output of a TM17 module: Falling edge).

Recommendation: Where the connected components do not have special output drivers, the recommendation is to use the rising edges for measurements.

Fail-safe digital inputs/outputs

Table 10-281 Technical data for fail-safe digital inputs (F-DI)

Number of inputs	3 F-DI (or as 6 DI)
Input voltage • Rated value • For signal "1" • For signal "0" [1]	 • 24 VDC • 15 ... 30 V • -3 ... +5 V
Electrical isolation [2]	Yes (via optocoupler)
Current consumption typical at 1 signal level	3.5 mA at 24 V 5 mA at 24 V [3]
Input delay, typical (hardware) • Signal "0" → "1" • Signal "1" → "0"	 • 50 µs • 150 µs
Protection	Short circuit, ground fault and overload proof

1) The digital inputs are protected against polarity reversal up to -30 V

2) Reference potential for DI 16, DI 18, DI 20 and DO 16 is terminal M1 (X120, X130)

3) Up to and including hardware version "C" for D410-2 DP or "B" for D410-2 DP/PN

Table 10-282 Technical data for fail-safe digital output (F-DO)

Number of outputs	1 F-DO (or as 1 DO)
Rated load voltage, permissible range	24 VDC, 20.4 ... 28.8 V
Electrical isolation	Yes (via optocoupler)
Current load, max.	500 mA
Residual current, max.	2 mA
Output delay time, typ./max. (hardware) [1] • Signal "0" → "1" • Signal "1" → "0"	 • 150 µs / 400 µs • 75 µs / 100 µs
Permissible quiescent current	2 mA
Protection	Short circuit, ground fault and overload proof Automatic restart after overload tripping

1) Data for: V_{cc} = 24 V; load 48 Ω; high ("1") = 90% V_{out}; low ("0") = 10% V_{out}

Analog input

Table 10-283 Technical data for the analog input

Number of inputs	1
Galvanic isolation	No
Common-mode range	-12 ... +12 V

Integration time / conversion time (per channel) • Parameters can be assigned • Integration time	 Yes ≥ 62.5 µs
Operational limit (across temperature range, in relation to the end value of the measuring range in the input range) • Voltage input • Current input	 ±0.65% ±1.1%
Basic error limit (operational limit at 25° C, in relation to the end value of the measuring range in the input range) • Voltage input • Current input	 ±0.5% ±0.5%
Linearity error (with reference to the input range)	±0.05%
Repeatability (in steady state at 25° C, with reference to input range)	±0.05%
When used as analog voltage input	
Input voltage range	-10 ... +10 V
Resolution	12-bit + sign (based on ± 11 V)[1)]
Input resistance (R_i)	> 100 kΩ
Temperature error (with reference to the input range)	±0.005%
When used as analog current input	
Input current range	-20 ... +20 mA
Resolution	11-bit + sign (based on ± 22 mA)[1)]
Input resistance (R_i)	250 Ω
Temperature error (with reference to the input range)	±0.02%
[1)] The maximum controllable area is approx. ± 11 V or ± 44 mA The resolution refers to the specified area (irrespective of the engineering settings).	

Onboard encoder interface

Table 10-284 Technical data of the encoder interface

Encoder interface	• TTL or HTL incremental encoders (with adjustable parameters) • Absolute encoder
Power supply	24 VDC / 0.35 A or 5 VDC / 0.35 A Short-circuit and overload proof
Limit frequency	500 kHz
SSI baud rate	100 ... 1000 kBaud

Resolution absolute position SSI	30 bit
Max. cable lengths	
• For TTL incremental encoder	• 100 m (bipolar signals only) [1), 3)]
• For HTL incremental encoder	• 100 m (for unipolar signals) • 300 m (for bipolar signals) [1), 2)]
• For SSI absolute encoder	• 100 m (depending on the baud rate) [4)]

1) Signal lines twisted in pairs and shielded.
Because the transmission technology is more robust, the bipolar connection should always be used.
The unipolar connection should only be used if the encoder type does not output push-pull signals.

2) As of a cable length of 200 m, use a power supply cable with a cable cross-section ≥ 0.75 mm²!

3) 100 m with remote sense

4) For the cable length, see the diagram "Maximum cable lengths depending on the SSI baud rate for SSI encoders"

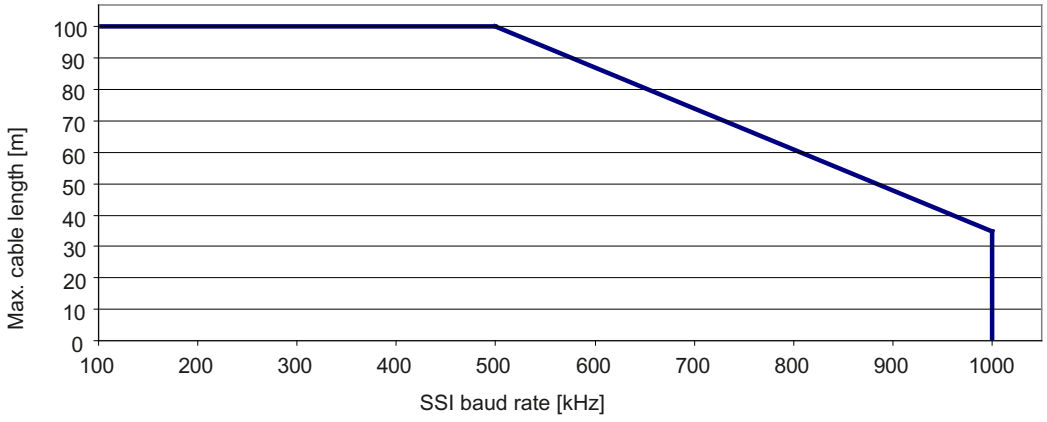


Figure 10-295 Maximum cable length depending on the SSI baud rate for SSI encoders

Note

The CUA32 control unit adapter also provides an encoder interface for an HTL, TTL or SSI encoder.

The technical data of the CUA32 adapter module can be found in the *SINAMICS S120 AC Drive* Manual.

Clock

Properties of the real-time clock

The table below contains the features and functions of the Control Unit clock.

Table 10-285 Clock properties

Properties	Meaning
Type	Hardware clock (integrated "real-time clock")
Default setting when delivered	DT#1992-01-01-00:00:00

Properties	Meaning
Maximum deviation per day for supply voltage switched on and switched off at 0° to 55° C	±5 s
Backup time	At least 5 days (at 0 to 55° C)
Charging time	1 h

With power OFF

The Control Unit clock continues to operate with the POWER OFF for the duration of the battery backup time (excluding software clock). The buffer is recharged in the POWER ON state. If the real-time clock backup time is exceeded, the time is reset. If the SIMOTION D410-2 is reset to its factory settings, the clock is also reset to the "default setting when delivered".

Certificates, approvals, declarations of conformity

You can find an overview of the certifications available for the SIMOTION D410-2 in Appendix A (Page 7805).

You can also find further information on the Internet at:

https://support.industry.siemens.com/cs/ww/en/ps/14513/cert

10.2.2.6 Dimension drawings

SIMOTION D410-2 DP dimension drawing

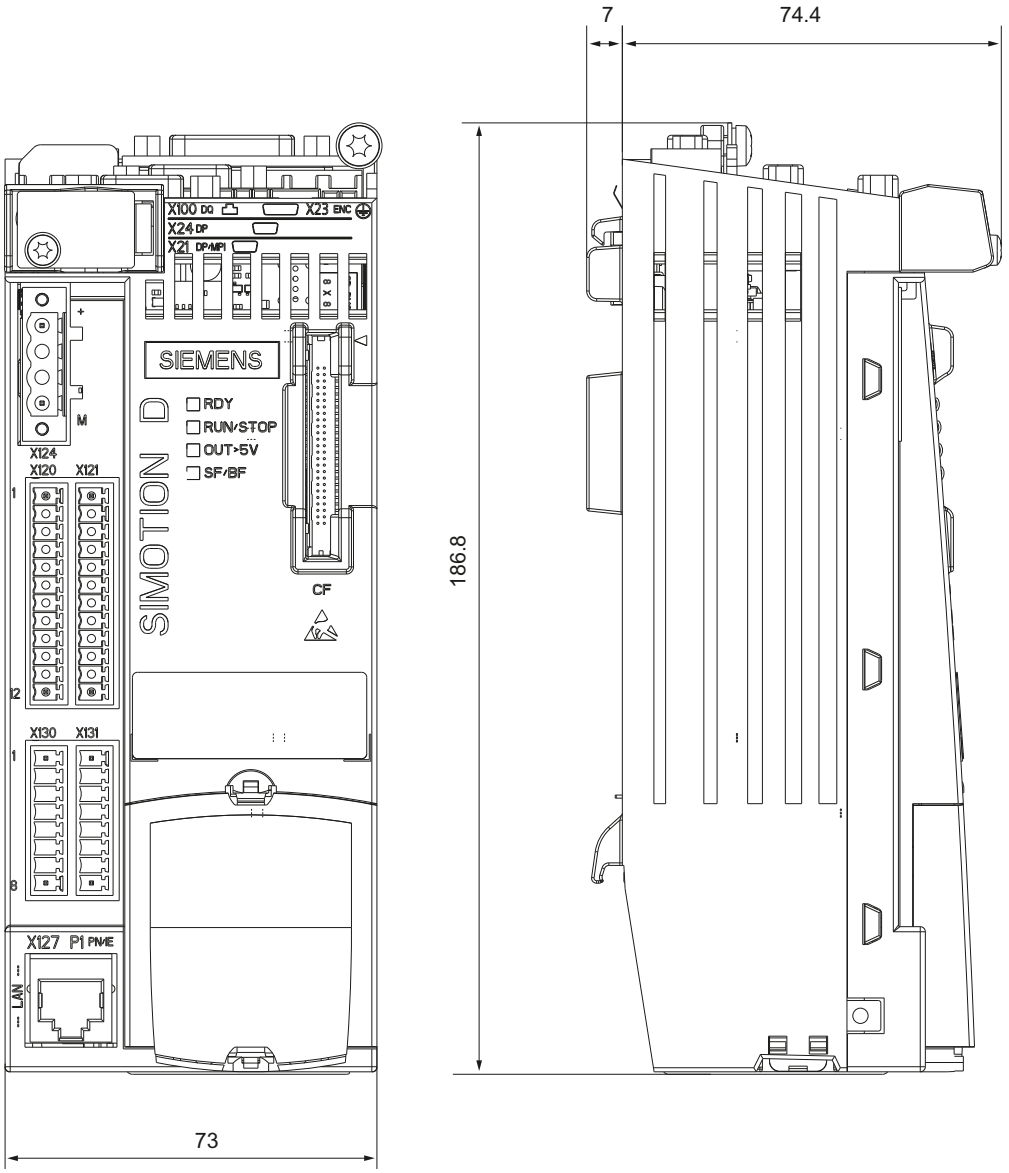


Figure 10-296 SIMOTION D410-2 DP dimension drawing (dimensions in mm)

Note

Overheating if ventilation clearances are too small

Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened service life of the component.

Maintain 50 mm ventilation clearances above and below the component.

SIMOTION D410-2 DP/PN dimension drawing

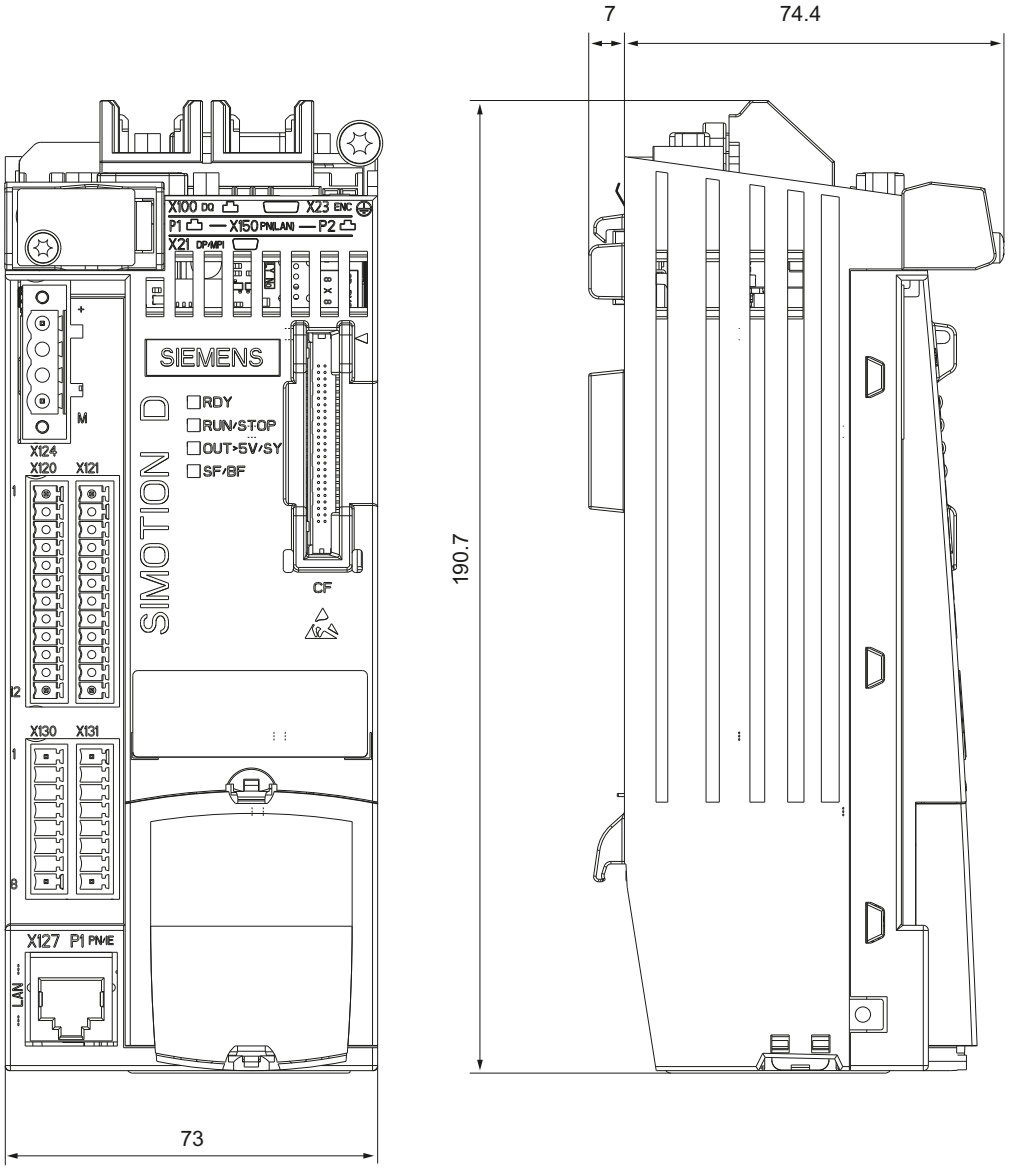


Figure 10-297 SIMOTION D410-2 DP/PN dimension drawing (dimensions in mm)

Note

Overheating if ventilation clearances are too small

Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened service life of the component.

Maintain 50 mm ventilation clearances above and below the component.

Mounting plate dimension drawing

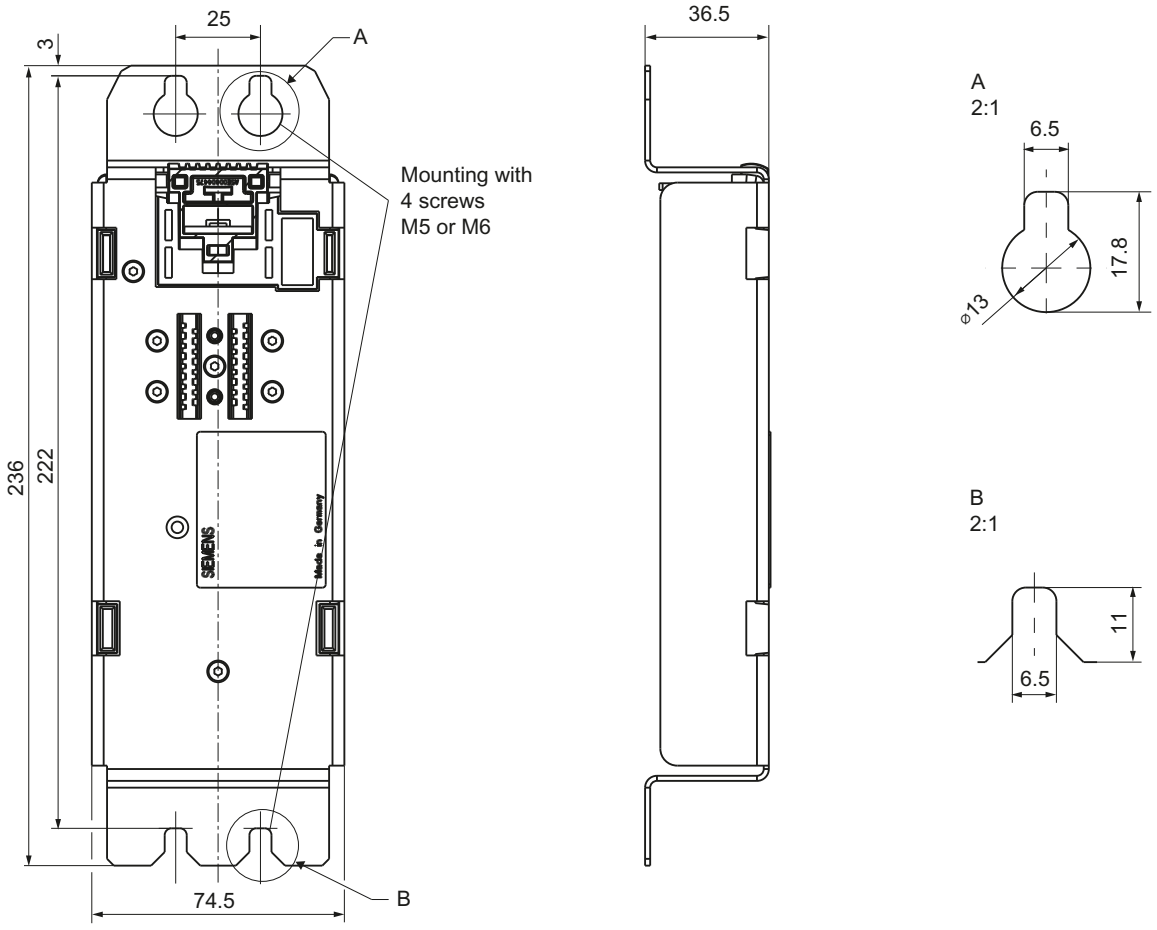


Figure 10-298 Mounting plate dimension drawing (dimensions in mm)

CAD data, dimension drawings, and circuit-diagram macros

Dimension drawings and CAD data

Dimension drawings, as well as 2D and 3D CAD data can be found:

- In the CAD Creator at the following Internet address (https://support.industry.siemens.com/cs/ww/en/view/30559271).
- In the DT Configurator of the Industry Mall (http://www.siemens.com/dt-configurator).
- Via CAx Download Manager (https://support.industry.siemens.com/my/ww/en/CAxOnline#CAxOnline).

Circuit-diagram macros

EPLAN circuit-diagram macros are available for the SIMOTION D410-2. The macros assist you when creating circuit diagrams.

The EPLAN circuit diagram macros can be ordered at the following Internet addresses:

- Drive Technology Configurator Industry Mall (http://www.siemens.com/dt-configurator)
- CAx Download Manager (https://support.industry.siemens.com/my/ww/en/CAxOnline#CAxOnline)
- Product support Support (https://support.industry.siemens.com/cs/ww/en/view/31622426)

10.2.2.7 Spare parts / accessories

Available spare parts and accessories

Table 10-286 Spare parts and accessories

Parts for SIMOTION D410-2	Article number	Accessories	Spare part
2 GB CompactFlash card (CF card) (latest CF card at the time of publication of this documentation)	6AU1400-1QA20-0AA0	X	-
Blanking cover for the protection of the operator controls	6SL3064-3BB00-0AA0	-	X
Backplane mounting plate for remote operation	6AU1400-7AA05-0AA0	X	-
CU310-2/D410-2 terminal kit contains: • 4 x I/O connectors for X120/X121, X130/X131 • 1 x 24 V connector for X124 • 3 x blanking covers for DRIVE-CLiQ/PROFINET • 1 x shield connection terminal incl. screw	6SL3064-8LA01-0AA0	-	X
Dust protection filler plugs for sealing unused DRIVE-CLiQ, PROFINET and Ethernet ports Filler plugs (50 pcs)	6SL3066-4CA00-0AA0	X	X
Fan	6SL3064-1AC00-0AA0	-	X
Plug connector for PROFIBUS interface up to 12 Mbps, 90° cable outlet:			
Without PG socket without FastConnect insulation displacement	6ES7972-0BA12-0XA0	X	-

Parts for SIMOTION D410-2	Article number	Accessories	Spare part
Without PG socket with FastConnect insulation displacement	6ES7972-0BA52-0XA0	X	-
With PG socket without FastConnect insulation displacement	6ES7972-0BB12-0XA0	X	-
With PG socket with FastConnect insulation displacement	6ES7972-0BB52-0XA0	X	-
Plug connector for PROFIBUS interface up to 12 Mbps, 35° angular cable outlet:			
Without PG socket without FastConnect insulation displacement	6ES7972-0BA42-0XA0	X	-
Without PG socket with FastConnect insulation displacement	6ES7972-0BA61-0XA0	X	-
With PG socket without FastConnect insulation displacement	6ES7972-0BB42-0XA0	X	-
With PG socket with FastConnect insulation displacement	6ES7972-0BB61-0XA0	X	-
Plug connector for PROFIBUS interface up to 12 Mbps, 180° axial cable outlet:			
Without PG socket without FastConnect insulation displacement	6GK1500-0EA02	X	-
Without PG socket with FastConnect insulation displacement	6GK1500-0FC10	X	-
Plug connector for Industrial Ethernet / PROFINET, 180° cable outlet:			
RJ45 plug connector, IE FC RJ45 Plug 180 • 1 unit package • 10 unit package • 50 unit package	 6GK1901-1BB10-2AA0 6GK1901-1BB10-2AB0 6GK1901-1BB10-2AE0	X	-
Plug connector for Industrial Ethernet / PROFINET, 145° cable outlet:			
RJ45 plug connector, IE FC RJ45 Plug 145 • 1 unit package • 10 unit package • 50 unit package	 6GK1901-1BB30-0AA0 6GK1901-1BB30-0AB0 6GK1901-1BB30-0AE0	X	-
FastConnect cables for Industrial Ethernet / PROFINET			
IE FC standard cable GP 2x2	6XV1840-2AH10	X	-
IE FC flexible cable GP 2x2	6XV1870-2B	X	-
IE FC trailing cable GP 2x2	6XV1870-2D	X	-
IE FC trailing cable 2x2	6XV1840-3AH10	X	-
IE FC marine cable 2x2	6XV1840-4AH10	X	-
Stripping tool for Industrial Ethernet / PROFINET Fast Connect cables			
IE FC stripping tool	6GK1901-1GA00	X	-

Reference

You can find order data information for other SINAMICS drive components, such as Line Modules, Motor Modules, DRIVE-CLiQ cables, etc. in Catalog *SIMOTION PM 21*.

Note

The procedure for replacing the SIMOTION D410-2 fan is described in the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

Spares On Web

Spares On Web is an information system that enables you to find out which spare parts are available for your device. For information, visit the following Internet address (https://www.sow.siemens.com).

In order to view the spare parts, you require the article number and the serial number of the module.

Both numbers can be found on the rating plate on the module or the packaging label.

TM31 terminal module

Characteristics

The TM31 Terminal Module allows you to expand the number of available digital I/Os as well as the number of analog I/Os within a drive system. The TM31 is connected via DRIVE-CLiQ. It has 2 DRIVE-CLiQ interfaces for this.

Interfaces

The TM31 contains the following terminals:

Table 10-287 Interface overview of the TM31

Interface	Quantity
Digital inputs	8
Bidirectional inputs/outputs	4
Relay outputs with changeover contact	2
Analog inputs	2
Analog outputs	2
Temperature sensor input (KTY84-130 or PTC)	1

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

For further information on the TM31 Terminal Module, see the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

TM41 terminal module

Characteristics

The TM41 Terminal Module can be used to expand the number of digital I/Os and analog inputs within a drive system. In addition, it enables the use of the TTL output for encoder simulation. The TM41 is connected via DRIVE-CLiQ.

Interfaces

The TM41 contains the following terminals:

Table 10-288 TM41 interface overview

Type	Quantity
Digital inputs	4
Digital I/Os	4
Analog inputs	1
TTL encoder output	1

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

For further information on the TM41 Terminal Module, see the *SIMOTION D410-2* Commissioning and Hardware Installation Manual.

TM54F terminal module

Characteristics

The TM54F Terminal Module is a terminal expansion module for snapping on to a DIN EN 60715 mounting rail. The TM54F offers safe digital inputs and outputs for control of Safety Integrated functions of SINAMICS.

A SIMOTION D410-2 can be assigned exactly one TM54F which is connected via DRIVE-CLiQ.

The TM54F Terminal Module is an alternative to using Safety Integrated functions via the onboard terminals (F-DI, F-DO) or via PROFIsafe.

Interfaces

Table 10-289 The following terminals are located on the TM54F:

Type	Quantity
Fail-safe digital outputs (F-DO) [1]	4
Fail-safe digital inputs (F-DI) [2]	10
Sensor power supplies, can be made dynamic [3], [4]	2
Sensor power supply, cannot be made dynamic [3]	1
Digital inputs for testing the F-DO during test stop	4

1) A fail-safe digital output consists of a P/M-switching output as well as a digital input for reading back the switching state.

2) A fail-safe digital input consists of two digital inputs.

3) Sensors: Fail-safe devices for commanding and detecting, such as emergency stop pushbuttons and safety locks as well as position switches and light arrays / light curtains.

4) Dynamic response: The sensor power supply is switched on and off during test stop for testing the sensors, the cable routing, and the evaluation electronics of TM54F.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

You find detailed information on the TM54F Terminal Module in the *SINAMICS S120 Safety Integrated* Function Manual.

See also

DRIVE-CLiQ interface (Page 7749)

TM15 and TM17 High Feature terminal modules

Characteristics

The TM15 and TM17 High Feature Terminal Modules are used to implement inputs of measuring inputs and outputs of output cams for SIMOTION D. In addition, these Terminal Modules provide drive-related digital I/Os with short signal delay times. TM15 and TM17 High Feature are connected by means of DRIVE-CLiQ.

TM15

Each of the 24 electrically isolated digital I/Os can be parameterized channel-by-channel as a digital input (DI), digital output (DO), a measuring input input, or an output cam output.

TM15 DI/DO

Each of the 24 isolated digital I/Os can be configured on a channel-specific basis as a digital input (DI) or digital output (DO). The digital I/Os can be interconnected using BICO technology and thus used from the drive side as well. Unlike the TM15, measuring input inputs and cam outputs are not available with the TM15 DI/DO.

Note

The module hardware for TM15 and TM15 DI/DO is identical. A distinction is only made by the addition of the component in the SIMOTION SCOUT project navigator using "Inserting I/O component".

TM17 High Feature (only available as spare part)

Each of the 16 digital I/Os can be parameterized channel-by-channel as a digital input (DI), digital output (DO), measuring input input, or an output cam output.

TM17 High Feature has fewer I/O channels than TM15, but more functionality. TM17 High Feature is distinguished by especially high resolution and accuracy as well as a configurable input filter and enabling inputs (max. 6 units). Parameterized enable inputs can enable measuring inputs or outputs of output cams (gate function). Due to their high accuracy, the digital I/O channels of the TM17 High Feature are non-isolated.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

Detailed information about the TM15 and TM17 High Feature can be found in the *SIMOTION Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

CUA31/CUA32 control unit adapter

Properties

You can connect a Power Module in blocksize format to the DRIVE-CLiQ interface using the CUA31 adapter module. The CUA32 adapter module also provides an encoder interface for an HTL, TTL or SSI encoder.

Interfaces

Table 10-290 Overview of the interfaces for the adapter modules

Interface	CUA31 [1]	CUA32
DRIVE-CLiQ interface	3	3
EP terminals/temperature sensor connection	1	1
Power Module Interface (PM-IF)	1	1
24 V electronic power supply	1	1
Encoder interface (HTL, TTL, SSI) Only SSI encoders without incremental tracks may be operated.	0	1
Maximum DRIVE-CLiQ cable length	100 m	100 m

[1] CUA31 with article number 6SL3040-0PA00-0AAx (x ≥ 1 required)

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component.

Additional references

You will find more information on the CUA31/CUA32 Control Unit Adapter in the *SINAMICS S120 AC Drive* Manual.

DMC20/DME20 DRIVE-CLiQ hub

Characteristics

The DMC20 and DME20 DRIVE-CLiQ hub modules are used to implement point-to-point distribution of a DRIVE-CLiQ line.

- DMC20 is the hub for the control cabinet configuration
- DME20 is the hub for use without a control cabinet (IP67 degree of protection).

The modules are particularly well suited to applications that require DRIVE-CLiQ link nodes to be removed in groups without interrupting the DRIVE-CLiQ link and therefore the data exchange.

The DMC20/DME20 is also used with a SIMOTION D410-2 when a second encoder is required. As an SMx Sensor Module and a motor with DRIVE-CLiQ interface only have one DRIVE-CLiQ interface, a DMC20/DME20 must be used for a second encoder via DRIVE-CLiQ. If a CUA31/ CUA32 is used, the DMC20/DME20 is not required. Alternatively, a second encoder can also be connected via the X23 interface on the SIMOTION D410-2.

NOTICE
Overheating if ventilation clearances are too small Insufficient ventilation clearances result in overheating and therefore in more failures and a shortened life of the component. Maintain 50 mm ventilation clearances above and below the component. The ventilation openings may not be covered by connecting cables.

Additional references

You will find detailed information about the DMC20/DME20 in the *SINAMICS S120 Control Units and Additional System Components* Manual.

10.2.2.8 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

EMC limits in South Korea

	KC registration number: KCC-REM-S49-SIMOTION For sellers or other users, please keep in mind that this device in an A-grade electromagnetic wave device. This device is intended to be used in areas other than home. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

The EMC limits to be observed for Korea correspond to the limits of the EMC product standard for variable-speed electric drives EN 61800-3 of category C2 or the limit class A, Group 1 according to EN 55011. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. Additional measures, such as the use of an additional RFI suppression filter (EMC filter), may be necessary.

The measures for EMC-compliant design of the system are described in detail in this manual respectively in the Installation Guideline EMC.

Note that the final statement on compliance with the standard is provided by the respective label attached to the individual unit.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (https://support.industry.siemens.com/cs/ww/en/ps/14506/cert)).

Marking for Australia and New Zealand

	SIMOTION D410-2 satisfies the requirement of the standard AS/NZS CISPR 16.
or	
	Marking with RCM (Regulatory Compliance Mark) or C-Tick with older components.

Marking for the Eurasian customs union

	EAC (Eurasian Conformity) Customs union of Russia, Belarus and Kazakhstan Declaration of conformity in accordance with the technical regulations of the customs union (TR CU).

Standards that are not relevant

	China Compulsory Certification SIMOTON D does not belong to the validity area of the China Compulsory Certification (CCC).

China RoHS

SIMOTION D complies with the China RoHS directive. You can find more information on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109738656).

Device-specific information

Note regarding SIMOTION D

Note

The product standard EN 61800-3 describes the EMC requirements placed on "Variable-speed drive systems". As such, it defines different limits depending on the location of the drive system.

SINAMICS S120 power units are designed for use in the second environment. The term second environment refers to all locations outside residential areas. These are basically industrial areas which are supplied from the medium-voltage line supply via their own transformers.

It is essential to follow the installation instructions in the SINAMICS S120 Manuals in order to ensure compliance with emitted interference and immunity values.

The same installation instructions apply for the SIMOTION D410-2 Control Unit as for the SINAMICS S120 CU310-2 Control Unit with regard to EMC.

For further information on this topic also refer to the *SIMOTION PM 21* Catalog as well as the SINAMICS Function Manuals.

10.2.2.9 ESD guidelines

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

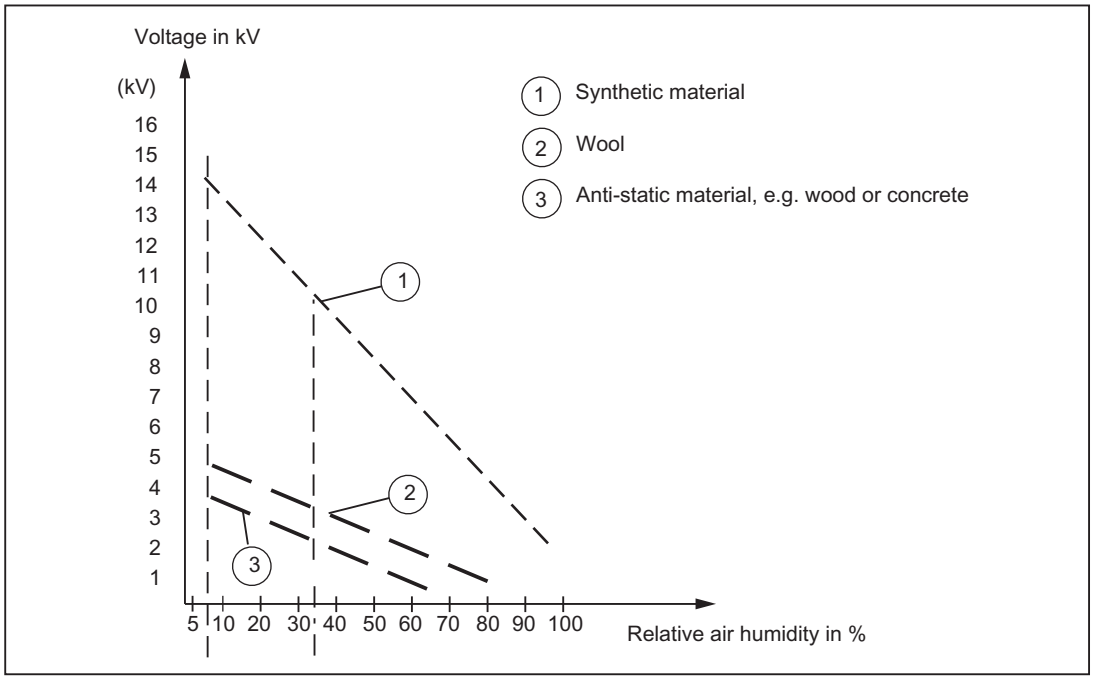


Figure 10-299 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

SIMOTION P 11

11.1 Commissioning Manual

11.1.1 SIMOTION P320-4 E / P320-4 S

Preface

Preface

This document is part of the **SIMOTION P documentation package**.

This documentation describes the SIMOTION P320-4 hardware platform which can be delivered in the SIMOTION P320-4 E and SIMOTION P320-4 S hardware versions:

- SIMOTION P320-4 E with the Windows Embedded Standard 7 32-bit operating system and real-time expansion for SIMOTION.
  Successor to SIMOTION P320-3.
- SIMOTION P320-4 S with the Windows 7 Ultimate 32-bit operating system and real-time expansion for SIMOTION.
  Successor to SIMOTION P350-3.

References

The following documents contain the descriptions for the SIMOTION P hardware platform:

- SIMOTION P320-4 E / P320-4 S, Manual, Edition 10/2016
- SIMOTION P320-4 E / P320-4 S, Commissioning and Hardware Installation Manual, Edition 10/2016

Validity range

This Commissioning and Hardware Installation Manual is valid for the SIMOTION P320-4 E and SIMOTION P320-4 S devices as of product level SIMOTION V4.5.

Standards

The SIMOTION system has been developed in accordance with ISO 9001 quality guidelines.

Sections in this documentation

The following sections describe the purpose and the use of this documentation:

- Safety instructions
  This section contains fundamental safety instructions for SIMOTION and specific safety instructions for the SIMOTION P320-4.
- Industrial Security
  You can find important information on industrial security here. What is industrial security? Which specific measures can be taken to protect your system from threats.
- Description
  System overview and product description for the SIMOTION P320-4.
  The communication versions are displayed.
- Application planning
  Points to note in advance:
  Upon delivery, the permitted installation positions, environmental and ambient conditions and electromagnetic compatibility.
  You will find this information in the manual SIMOTION P320-4 E / P320-4 S.
- Installation
  Description of the installation, mounting and assembly of the SIMOTION P320-4.
- Connection
  This section describes the requirements for connecting and the connection overview for the SIMOTION P320-4, and information on the connection of PROFINET, PROFIBUS and Ethernet.
- Power on and software installation
  Information on the first power-up of the SIMOTION P320-4 and additional software for HMI.
- Operation (hardware)
  Detailed description of the SIMOTION P State application.
- Parameter assignment / addressing
  This section contains the requirements for parameter assignment / addressing and an overview of the factory settings.
  One section describes the SIMOTION P Control Manager.
  Communication with Ethernet and PC internal is described.
  References are supplied for the PROFINET and PROFIBUS communication.
- Commissioning (software)
  This section contains information, notes and requirements for the commissioning and a recommended sequence for the first commissioning of the SIMOTION P320-4.
  The topics data backup, data storage concept and SIMOTION P general reset are described.
- Service and maintenance
  Information on the recording of diagnostics data, restoring factory settings, installing and removing the backup battery.
  Description of special power-up situations.
- Alarm, fault, and system messages
  Information on diagnostics via LED displays.
  Description of possible alarm and fault messages.
- Troubleshooting/FAQs
  List of possible errors and their remedies.

- Appendix
  The appendices contain information on standards and approvals, on the ESD guideline as well as a list of abbreviations.
- Index
  Alphabetical directory for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

11.1.1.1 Safety notes

Fundamental safety instructions

General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety instructions for SIMOTION P320-4

General safety instructions for the SIMOTION P320-4

⚠ WARNING

Life-threatening voltages are present with an open control cabinet

When you install the device in a control cabinet, some areas or components in the open control cabinet may be carrying life-threatening voltages.

If you touch these areas or components, you may be killed by electric shock.

Switch off the power supply to the cabinet before opening it.

System expansions

NOTICE

Damage through system expansions

Device and system expansions may be faulty and can affect the entire machine or plant.

The installation of expansions can damage the device, machine or plant.

Device and system expansions may violate safety rules and regulations regarding radio interference suppression.

If you install or exchange system expansions and damage your device, the warranty becomes void.

Note the following for system expansions:

- Only install system expansion devices designed for this device. Contact your technical support team or where you purchased your PC to find out which system expansion devices may safely be installed.
- Observe the information on electromagnetic compatibility (Page 7989).

NOTICE

"Open Type" UL508

Note that the device is classified as "Open Type" for use in the area of Industrial Control Equipment (UL508). Installation of the device in an enclosure according to UL508 is conditional for approval or operation according to UL508.

Battery and rechargeable battery

⚠ WARNING

Risk of explosion and release of harmful substances

Improper handling of lithium batteries can result in an explosion of the batteries.

Explosion of the batteries and the released pollutants can cause severe physical injury. Worn batteries jeopardize the function of the device.

Note the following when handling lithium batteries:

- Replace used batteries in good time, see the section "Replacing the backup battery" in the Commissioning and Hardware Installation Manual.
- Replace the lithium battery only with an identical battery or types recommended by the manufacturer (Article No.: A5E30314053).
- Do not throw lithium batteries into fire, do not solder on the cell body, do not recharge, do not open, do not short-circuit, do not reverse polarity, do not heat above 100°C and protect from direct sunlight, moisture and condensation.

High frequency radiation

NOTICE

Unintentional operating situations

High frequency radiation, e g. from a cellular phone, interferes with device functions and can result in malfunctioning of the device.

Persons are injured and the plant is damaged.

Avoid high-frequency radiation:

- Remove radiation sources from the environment of the device.
- Switch off radiating devices.
- Reduce the radio output of radiating devices.
- Observe the information on electromagnetic compatibility (Page 7989).

ESD Guideline

Electrostatic sensitive devices can be labeled with an appropriate symbol.

NOTICE
Electrostatic sensitive devices (ESD) When you touch electrostatic sensitive components, you can destroy them through voltages that are far below the human perception threshold. If you work with components that can be destroyed by electrostatic discharge, observe the ESD Guideline.

Further information

You can find more detailed information about the **EGB Guideline** in Annex B in the section with the same name.

Notes on use

⚠ WARNING
Hazards on an unprotected machine or plant According to the results of a risk analysis, hazards can occur on an unprotected machine. The hazards can result in personal injury. According to the risk analysis, the risk of personal injury can be avoided with the following measures: • Additional protective devices on the machine or plant. With this, especially the programming, configuration and wiring of the inserted I/O modules have to be executed, in accordance with the necessary risk analysis identified safety performance (SIL, PL or Cat.). • The correct use of the device has to be verified with a function test on the system. This test can detect programming, configuration and wiring errors. • Documentation of the test results that you can enter in the relevant safety records when required.

NOTICE

Ambient conditions

Ambient conditions for which the device is not suitable can cause faults or damage the device.

Note the following:

- Operate the device only in closed rooms. Failure to comply nullifies the warranty.
- Operate the device only in accordance with the ambient conditions specified in the technical specifications.
- Protect the device against dust, moisture and heat.
- Do not expose the device to direct sunlight or other strong sources of light.
- Without additional measures, such as a supply of clean air, the device may not be used in locations with harsh operating conditions caused by acidic vapors or gases.
- Observe the permissible mounting positions of the device.
- Do not obstruct the venting slots of the device.

Note

Use in an industrial environment without additional protective measures

This device was designed for use in a normal industrial environment according to IEC 60721-3-3.

11.1.1.2 Industrial security

Security concept for SIMOTION P320-4

Security

Note

Observe the general security information in this documentation for Industrial security (Page 957).

Regular change of the Windows password

Note

Changing the Windows password

For security reasons, the Windows password should be changed regularly.

It is essential that the AutoLogin is also adapted for this purpose.

You can find instructions in the following sections:

- Changing the Windows user password (Page 7837)
- AutoLogin for SIMOTION P (Page 7841).

Unlocking Windows

Note

Windows locked

Windows may be accidentally locked, e.g. through shortcut key Windows + L.

If you do not know the password for the SIMOTION P320-4, please contact the Siemens Industry Online Support (https://support.industry.siemens.com).

Windows firewall

Note

Windows firewall

To allow the Windows IP to be accessed externally, this must be set as an exception in the Windows firewall **File and Printer Sharing**. Otherwise, there is only limited access to Windows from outside.

Security and networks

Note

You will find information on the security of networks in Section General security measures (Page 7828).

Remote desktop connection

Note that for the SIMOTION P320-4 with Headless operation, you require a user name and password for a remote desktop connection. Per default, the remote desktop connection is already set up.

Note

Deactivating the remote desktop connection

If you do **not** use the remote desktop connection, it must be deactivated for security reasons.

See Section Deactivating the remote desktop connection (Page 7848)

SIMOTION IT

Note

Security concept

Note the security concept of HTTP/S, FTP and Telnet access on the Web server when working with SIMOTION IT.

You will find information in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual or the SIMOTION online help in Section Security concept.

Note

User administration

Note the information on the user administration when working with SIMOTION IT.

You will find information in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual or the SIMOTION online help in Section User administration.

Information on industrial security

The following sections are taken from the Motion Control Industrial Security Configuration Manual:

- Why is industrial security so important? (Page 7826)
- General security measures (Page 7828)
- Product-specific measures (virus scanners) (Page 7836)

You can view the entire document in the Industry Online Support (https://support.industry.siemens.com/cs/ww/en/view/108862708).

Why is industrial security so important?

The topic of data security and access protection (security) is becoming more and more important in industrial environments. The progressive networking of entire industrial plants, the vertical integration and networking of the individual levels of a company, and new technologies, such as remote maintenance and remote access, are leading to increased requirements for protecting industrial plants.

The threats are diverse and the consequences far-reaching.

Possible threats:

- Espionage of data, recipes, etc.
- Sabotage of production plants
- System stoppage, e.g. due to virus infection and malware

- Manipulation of data or application software
- Unauthorized use of system functions

Possible effects of a security incident

- Loss of intellectual property
- Loss of production or reduced product quality
- Company image and economic damage
- Catastrophic environmental influences
- Danger to persons and machines

Trends in the IT sector

Overview

There are many new trends which affect industrial security:

- Cloud computing in general
  The number of network connections across the world is constantly increasing. This enables innovations such as cloud computing and the applications that go hand in hand with it. In conjunction with cloud computing, there has been a massive increase in the number of mobile devices, such as mobile phones and tablet PCs.
- Wireless technology
  On the other hand, the increasing use of mobile devices has only become possible thanks to the ubiquitous availability of mobile networks. Wireless LAN is also becoming increasingly available.
- Smart Grid
  Networking is not only limited to data networks, it also influences our energy infrastructure.
- Worldwide remote access to plants, machines and mobile applications
- The "Internet of things"
  Millions of electronic devices are becoming network-capable and are communicating via the Internet, such as onboard computers in cars, which send warranty information to dealers, or water meter sensors that transmit water consumption data to municipal water suppliers via radio.

However, in order for everything from cloud computing to sensors to work without service disruptions, you need reliable network infrastructures that are well protected against attacks from malware and hackers.

Possible corporate security holes

Possible security holes or weak points

The security chain of a company is only as strong as its weakest link. Security holes can exist at numerous points. The following list gives only a few examples:

- Employees
- Production plants
- Network infrastructure
- Data centers
- PC workstations
- Laptops
- Tablet PCs
- Printers
- Smartphones
- Portable storage media
- Guidelines and regulations

For this reason, a holistic approach is required to deal with the issue of security. Coordinated guidelines and regulations are required that cover all areas: Devices, systems, processes and employees.

General security measures

Overview

In the following section you will learn about the general security measures you can take in order to protect your system from threats. All of the measures are recommended.

Additional specific security measures for SINUMERIK, SIMOTION and SINAMICS products can be found in Section Product-specific security measures (Page 7836).

Basically, the measures should be coordinated with one another and correspond to the ring-shaped principle of the "Defense in Depth" strategy. The measures are structured according to the "onion" principle and each measure forms an additional protective layer around the core: the production plant.

Figure 11-1 Defense in depth strategy

- **Plant security**
  Plant security represents the outermost protective ring. Plant security includes comprehensive physical security measures, e.g. entry checks, which should be closely coordinated with protective measures for IT security.
- **Network security**
  The measures, grouped under the keyword "Network security", form the core of the protective measures. This refers to the segmentation of the plant network with limited and secure communication between subnetworks ("secure islands") and the interface check with the use of firewalls.
- **System integrity**
  "System integrity" represents the combination two major measures. PC-based systems and the control level must be protected against attacks. Steps include the following measures:
  - User authentication for machine or plant operators with individual authorization levels
  - Integrated access protection mechanisms in the automation components to prevent unauthorized changes via the engineering system or during maintenance
  - The use of antivirus and whitelisting software to protect PC systems against malware
  - Maintenance and update processes to keep the automation systems up-to-date (e.g. patch management, firmware updates, etc.)

Plant security

Physical protection of critical production areas

Unauthorized persons may be able to enter the production site/building and damage or alter production equipment as a result of gaps in a company's physical security. Confidential information can also be lost. This can be prevented if both the company's site and the production areas are protected accordingly.

Company security

The company's physical security can be ensured via the following measures:

- Closed off and monitored company premises
- Entry control, keys / card readers and/or security personnel
- Escorting of external personnel by company employees

Physical production security

The physical security of a production location can also be ensured via the following measures:

- Separate access control for production areas.
- Installation of critical components in securely lockable cabinets / switching rooms including monitoring and alarm signaling options
- Prohibited production areas with restricted access rights
- Configuration of the radio field to restrict the WLAN range so that it is not available outside the defined areas (e.g. factory building).
- Guidelines that prevent the use of third-party data storage media (e.g. USB sticks) and IT devices (e.g. notebooks) classified as insecure on the control.

Further information

Further information on integrated security solutions can be found on the Surveillance page (http://www.buildingtechnologies.siemens.com/bt/global/en/security-solution/Pages/security-solution.aspx).

Network security

Network segmentation

Separation between production and office networks

One important protective measure for your control is the strict separation of the production networks and the other company networks. This separation creates protection zones for your production networks.

Note

The products – drives, controllers, commissioning tools (e.g. STARTER or Startdrive) – described in this manual must only be operated in protection zones.

Separation by means of a firewall system

In the simplest scenario, separation is achieved by means of an individual firewall system which controls and regulates communication between networks.

Separation via a DMZ network

In the more secure version, the coupling is established via a separate DMZ network. In this case, direct communication between the production network and the company network is completely prevented by firewalls and only takes place indirectly via servers in the DMZ network.

Note

The production networks should also be divided into separate automation cells in order to protect critical communication mechanisms.

General security measures

Observe the general security measures even within protection zones, for example:

- Virus scanners (Page 7834)
- Reduction of attack points (Page 7834)

Network segmentation with SCALANCE S

Siemens provides SCALANCE S security modules to meet network protection and network segmentation requirements.

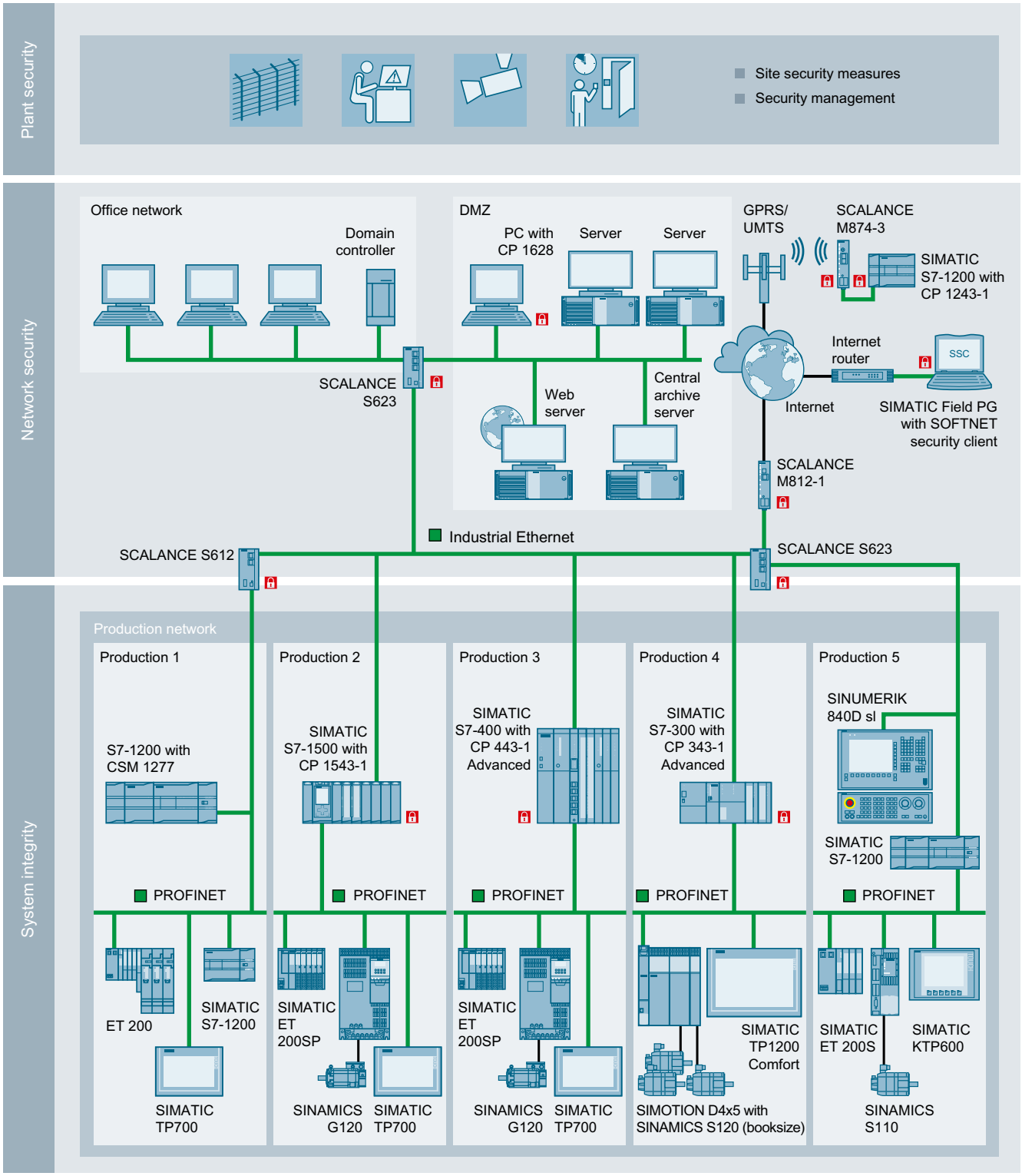
SCALANCE S security modules

SCALANCE S security modules with Security Integrated provide:

- Stateful inspection firewall
  In order to implement user-specific control and logging, firewall rules can also be specified that only apply to certain users.
- VPN via IPsec (data encryption and authentication)
  This establishes a secure tunnel between authenticated users whose data cannot be intercepted or manipulated. The most important aspect is the protection against external access via the Internet.
- NAT/NATP (address translation)
- Router functionality (PPPoE, DDNS) for broadband Internet access (DSL, cable)
- S623 with additional VPN port (DMZ) enables the secure connection of an additional network for service and remote maintenance purposes. S623 also permits the secure, redundant connection of subordinate networks by means of routers and firewall redundancy.

Principle

This application example shows cell segmentation by several SCALANCE S modules, each of which is upstream of the automation cell. The data traffic to and from the devices within automation cells can be filtered and controlled with the SCALANCE S firewall. If required, the traffic between the cells can be encrypted and authenticated. Secure channels and client access from the PCs to the cells can be established via SOFTNET Security Client, VPN client software for PCs.


Plant security
Site security measures
Security management
Network security
Office network
Domain controller
SCALANCE S623
DMZ
PC with CP 1628
Server
Server
Web server
Central archive server
GPRS/ UMTS
SCALANCE M874-3
SIMATIC S7-1200 with CP 1243-1
Internet router
SSC
Internet
SIMATIC Field PG with SOFTNET security client
SCALANCE M812-1
Industrial Ethernet
SCALANCE S612
SCALANCE S623
System integrity
Production network
Production 1
S7-1200 with CSM 1277
PROFINET
ET 200
SIMATIC S7-1200
SIMATIC TP700
Production 2
SIMATIC S7-1500 with CP 1543-1
PROFINET
SIMATIC ET 200SP
SINAMICS G120
SIMATIC TP700
Production 3
SIMATIC S7-400 with CP 443-1 Advanced
PROFINET
SIMATIC ET 200SP
SINAMICS G120
SIMATIC TP700
Production 4
SIMATIC S7-300 with CP 343-1 Advanced
PROFINET
SIMATIC TP1200 Comfort
SIMOTION D4x5 with SINAMICS S120 (booksize)
Production 5
SINUMERIK 840D sl
SIMATIC S7-1200
PROFINET
SIMATIC ET 200S
SIMATIC KTP600
SINAMICS S110

System integrity

System hardening

Reduction of attack points

Network services and ports

Activated services represent a risk. To minimize the risk, only the necessary services for all of the automation components should be activated. Ensure that all activated services are taken into account (especially web servers, FTP, remote maintenance, etc.) in the security concept.

A description of the ports used can be found in the Manuals and Function Manuals of the respective products.

User accounts

Any active user account allows access to the system is thus a potential risk. Therefore, take the following security measures:

- Reduction of configured/activated user accounts to the actually needed minimum
- Use of secure access data for existing accounts
- Regular checks, especially of the locally configured user accounts
- Regular change of passwords

Passwords

NOTICE
Changing default passwords The misuse of passwords can also represent a considerable security risk. We recommend that default passwords be changed during the commissioning and changed at regularly defined intervals as required.

Virus scanner

The use of a virus scanner must not impact the production operations of a plant. As the last consequence, this will lead to even a virus-infected computer not being permitted to immediately shut down if this would cause the control of the production process to be lost.

In order to be used on industrial control components, a virus scanner should therefore meet the following requirements:

Virus scanner requirements

- If a local firewall that has been adapted to the production operations is used, it must be possible to install the virus scanner without its own firewall.
- The virus scan clients can be divided into (product- and task-specific) groups and configured separately.
- It must be possible to deactivate the automatic distribution of the virus signatures and other updates.
- It must be possible to carry out the distribution of the virus signatures and updates manually and in groups.
- It must be possible to conduct a file scan and system scan manually and in groups.
- For the virus detection scenario, a message can be configured without a file action such as "Delete", "Clean", etc. being automatically carried out.
- It must be possible to log all of the messages on the virus scan server.
- On a virus scan client, it must be possible to suppress the local message window because it could obscure important messages from the production process.

Note

Installation of software

The installation of software is often a process which represents a serious and complicated change to the respective system. The storage location of the files to be installed must always be free of viruses (e.g. a file server with its own virus scanner or DVD checked for viruses).

Patch management

Microsoft security updates

The **WSUS** (Windows Server Update Service) system functionality provided by Microsoft is available for current Windows systems. WSUS supports administrators by providing Microsoft updates in large local networks. WSUS automatically downloads update packages from the Internet (Microsoft Update) and offers them to the Windows clients for installation.

The fully automatic update process ensures that Microsoft security updates are always available on Siemens clients.

Product-specific measures

Virus scanners, Windows security patches, SIMOTION P

General information on virus scanners

Once an industrial PC system is connected to the Internet, either directly or via an internal company network, there is a danger that it can become infected with a virus. However, malicious software is not only able to reach the system via the Intranet/Internet, but also, for example, via a removable storage device (such as a USB memory stick) attached to the system for backing up data.

SIMOTION P320-4 virus scanners

A virus scanner that runs on Microsoft Windows, as used in office or home computers, has a deep impact on a system's processes. There are, for example, processes such as real-time scans or regular system scans. Such interventions can cause performance issues for the system, and as a result, for the SIMOTION Runtime software. Although the SIMOTION Runtime software runs in a real-time environment, it still depends on the available system resources.

Note

Because of the resulting performance impairments, the installation and use of a standard virus scanner on a SIMOTION P320-4 during system runtime does not make sense and is not permitted.

Using a virus scanner

As a standard virus scanner cannot be used for SIMOTION P320-4, an alternative procedure is followed. The virus scanner is installed to a separately bootable Windows PE operating system. It is started, for example, from a CD or a USB storage device and then performs a virus scan.

Note

FAQ in Industry Online Support

More information on using a virus scanner on a SIMOTION P320-4 can be found in the FAQ "How can a virus scanner be used on a SIMOTION P320-4?" (https://support.industry.siemens.com/cs/ww/en/view/59381507), which is available as a download from the Industry Online Support portal.

Windows security patches

A brief test is performed when a new SIMOTION version is released. During this brief test, a check is performed to establish whether the installation of the security update has affected any basic functions.

Changing the Windows user password

Permitting the Windows password to be changed

Before the Windows password can be changed, it must be permitted for the user.

If the following procedure is not observed, an error message is issued and the Windows password cannot be changed.

Note the following procedure:

1. Select the entry point **Control Panel** > **All Control Panel Items** > **Administrative Tools** > **Computer Management**.
   Double-click **Computer Management**.

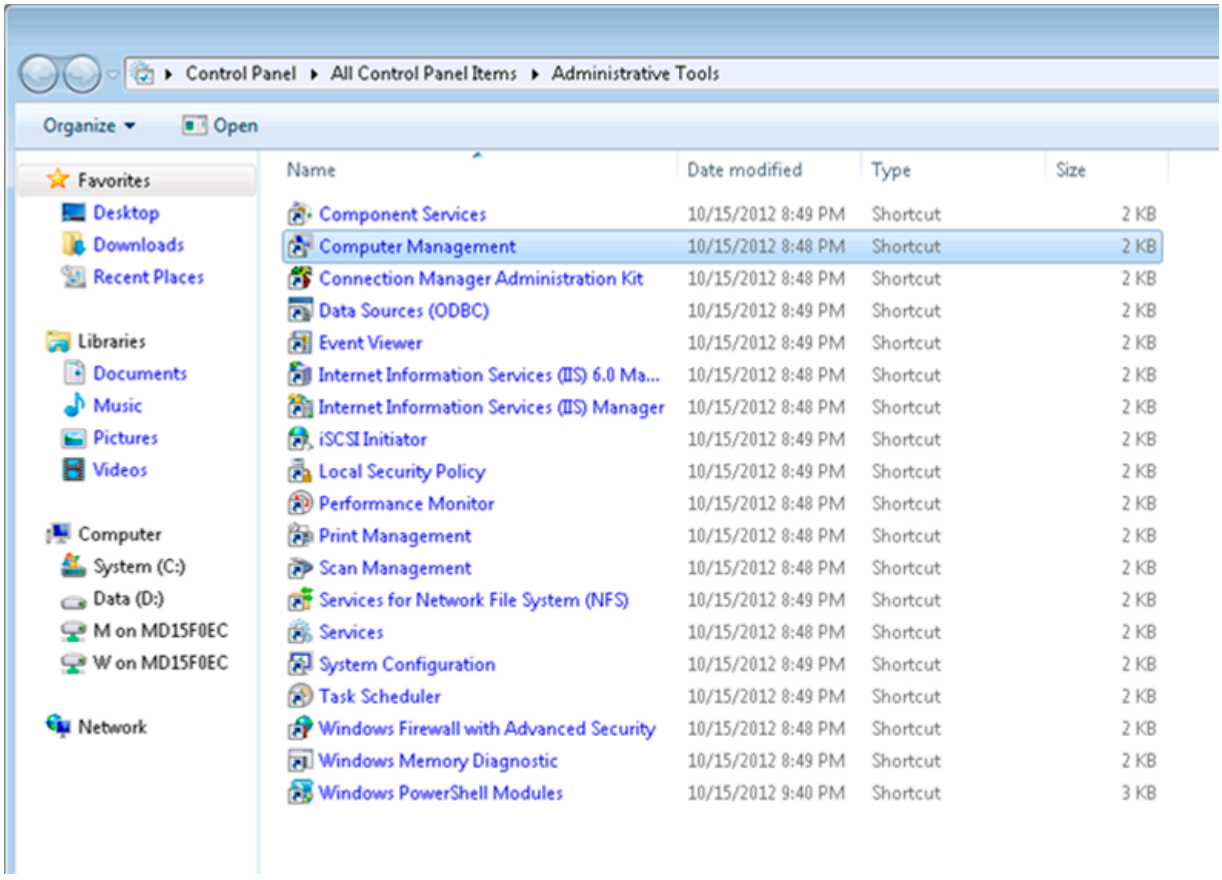


Figure 11-2 Homepage:

2. The **Users** folder is displayed at **Local Users and Groups** in the **Computer Management** window.
   Click the **Users** folder. The available user groups are displayed.

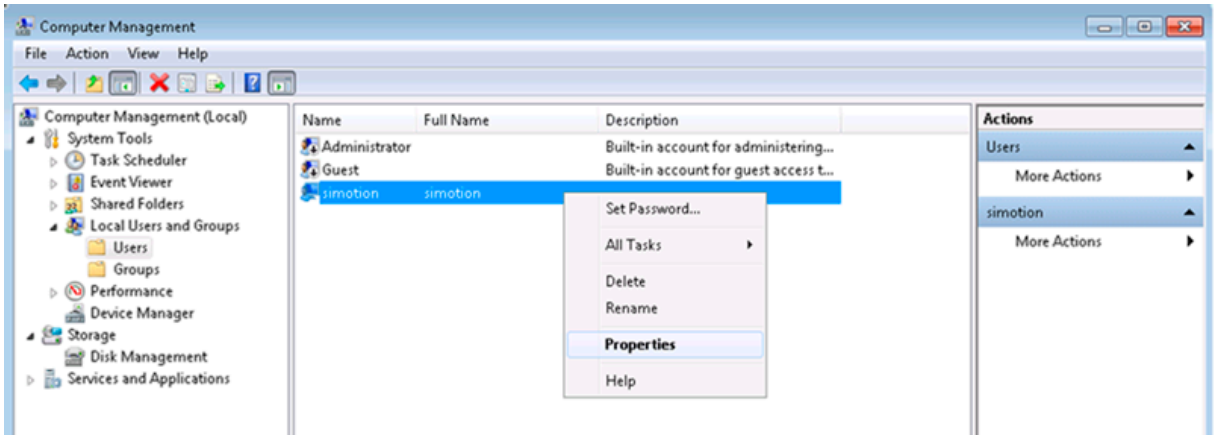


Figure 11-3 Computer management

3. With the selection of **Users simotion**, you can display the properties via **Properties**.
4. Deactivate the "**User cannot change password**" checkbox in the **General** tab on the **simotion Properties** dialog box.
   This checkbox must be deactivated for the user to change the Windows password.

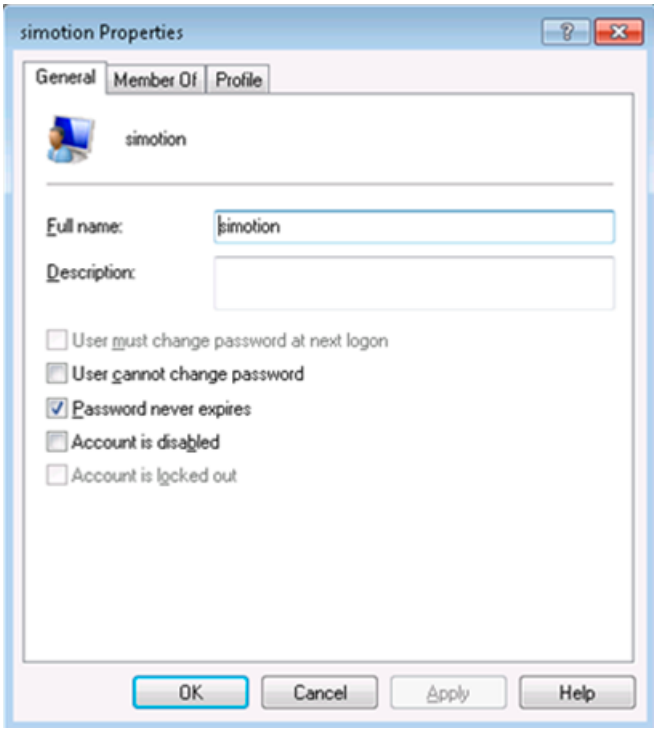


Figure 11-4 SIMOTION properties

Changing the Windows password

You can find general information on the Windows password on the website at Microsoft - Changing des Windows-password (http://windows.microsoft.com/en-us/windows/change-windows-password#change-windows-password=windows-7).

To change the Windows password, proceed as follows:

1. Press Ctrl+Alt+Del. The following Windows window opens.

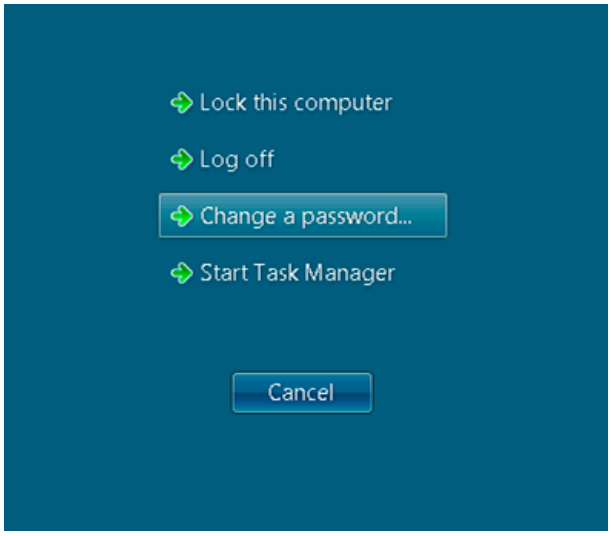


Figure 11-5 Changing the Windows password

2. After you have confirmed **Change a password...** , the Windows window to change the Windows password opens.
First enter your current password and then the new Windows password. You must enter the new Windows password a second time for confirmation.

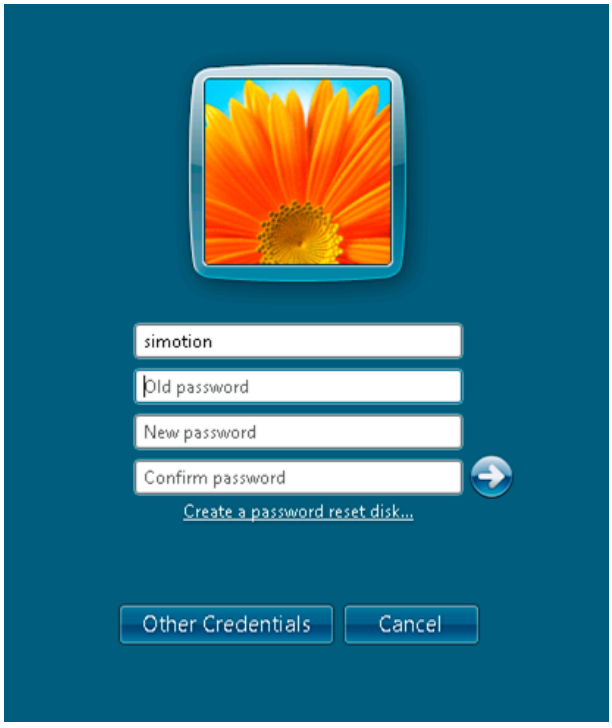


Figure 11-6 Entering the new Windows password

3. The password was changed successfully.

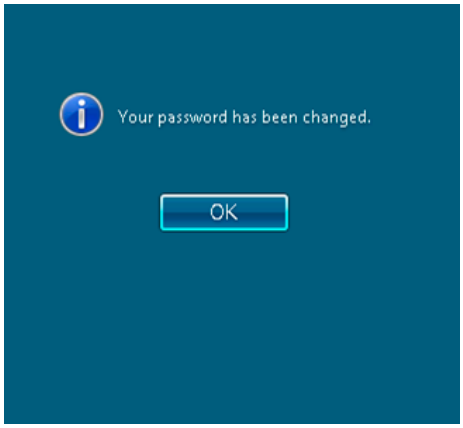


Figure 11-7 Confirmation of change

4. In the event of an error when the "User cannot change password" checkbox is activated, the following error message is displayed.
The Windows password has not been changed.

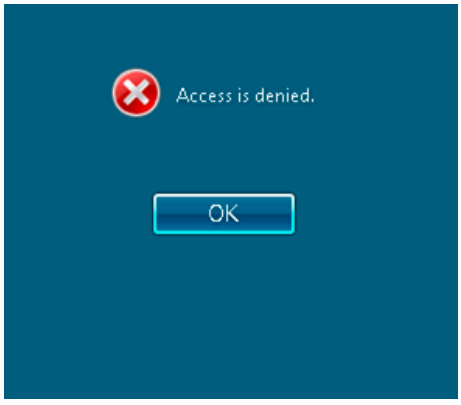


Figure 11-8 Error message when changing the Windows password has not been permitted.

Changing the AutoLogin

SIMOTION P AutoLogin

The password for the AutoLogin must be changed when the user has changed the Windows password.

The Windows user password is changed in the normal way, e.g. "Change Password" screen. See also Section Changing the Windows user password (Page 7837).

Automatically Log On

SIMOTION P320-4 is preset so that the user does not have to enter the user name and the password to log on.

Preparation for changing the AutoLogin

In order that an AutoLogin still functions after changing the Windows password, the password for the AutoLogin must be changed.

The required procedure is shown in the following so that the change of the AutoLogin for SIMOTION P320-4 can be prepared:

1. Open the **Computer Management** window via **Control Panel** > **All Control Panel Items** > **Administrative Tools** .

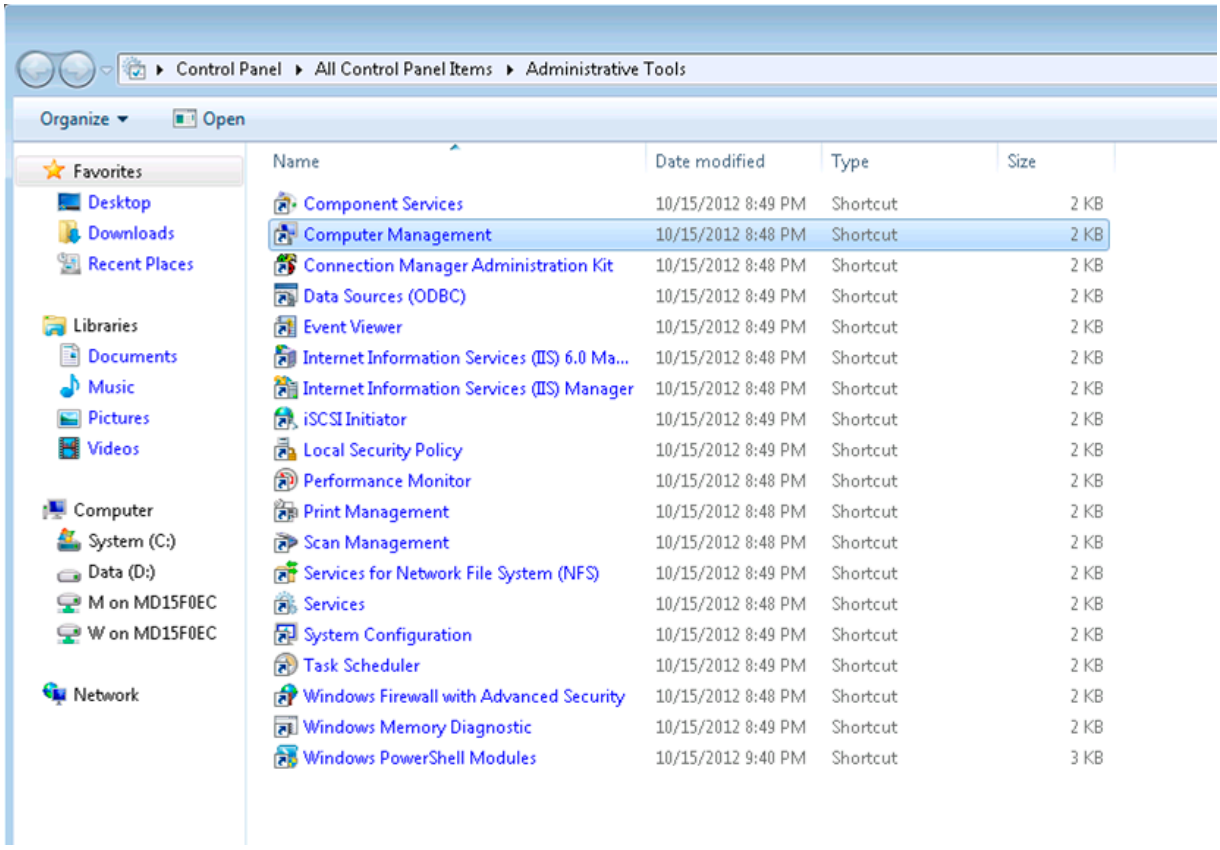


Figure 11-9 Computer Management entry point

2. The **Users** folder is displayed at **Local Users and Groups** in the **Computer Management** window.
   Click the **Users** folder. The available user groups are displayed.

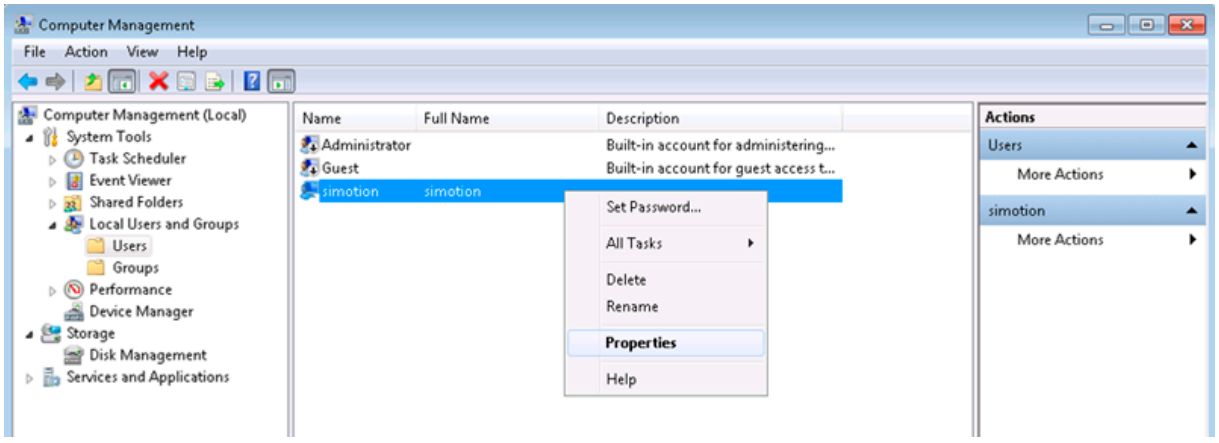


Figure 11-10 Computer Management - Users

3. With the selection of **Users**> **simotion**, you can display the properties via **Properties**.

4. Deactivate the **User cannot change password** checkbox in the **simotion Properties** dialog box.

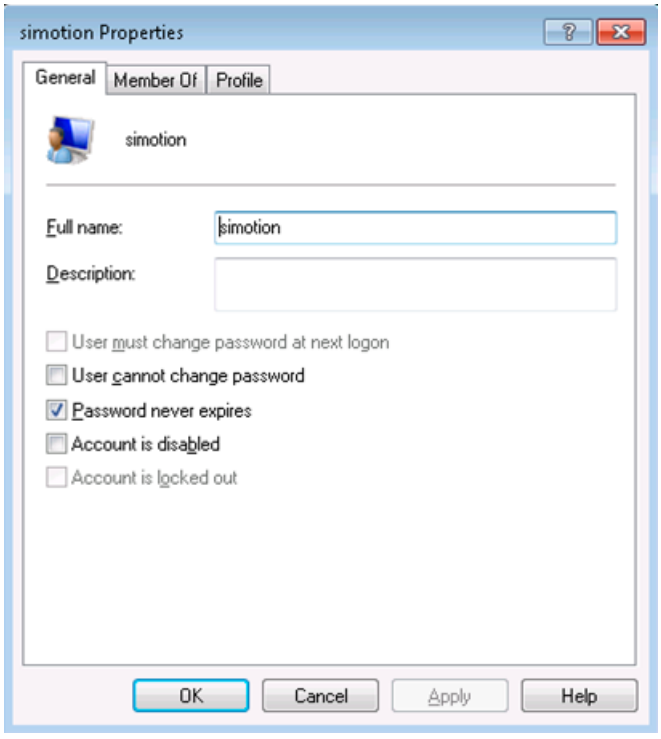


Figure 11-11 simotion Properties - General dialog box

5. Confirm with **OK.**

Changing the password for AutoLogin

To change the password for the AutoLogin, proceed as follows:

1. Open the search via **Windows-Start** and enter: **netplwiz**

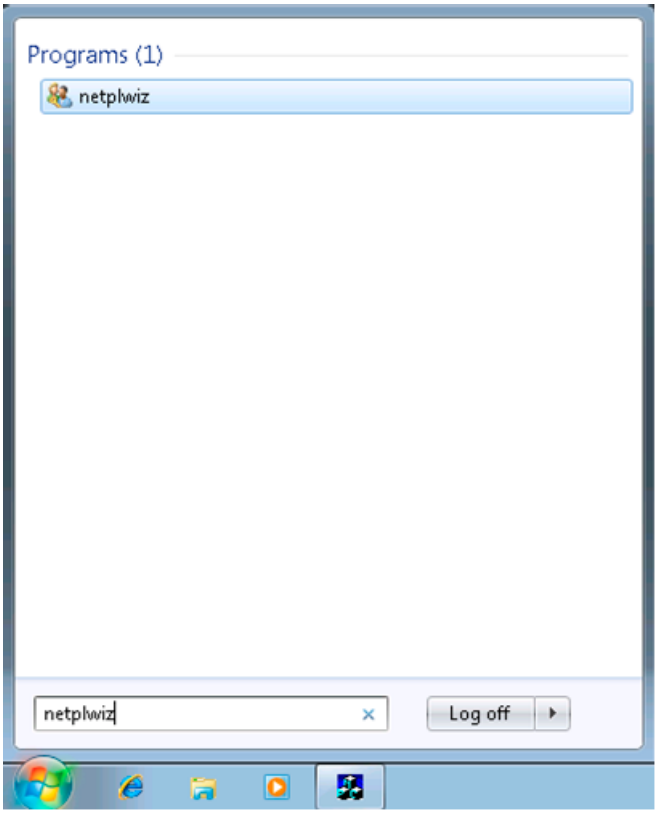


Figure 11-12 Programs netplwiz

2. The following **User Accounts** dialog box opens.

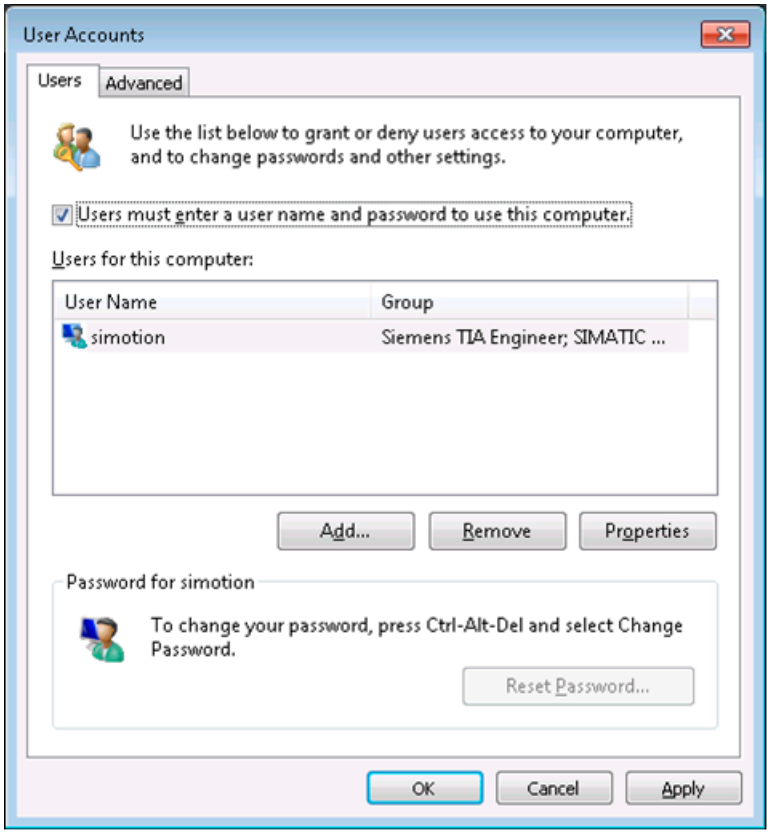


Figure 11-13 User Accounts

3. In the **Users** tab, activate the **Users must enter a user name and password to use this computer** checkbox.
   The **Apply** button is now active. Deactivate the checkbox and accept the changes now with the **OK** or **Apply** button.
   This is necessary so that the **Automatically Log On** dialog box is subsequently displayed.

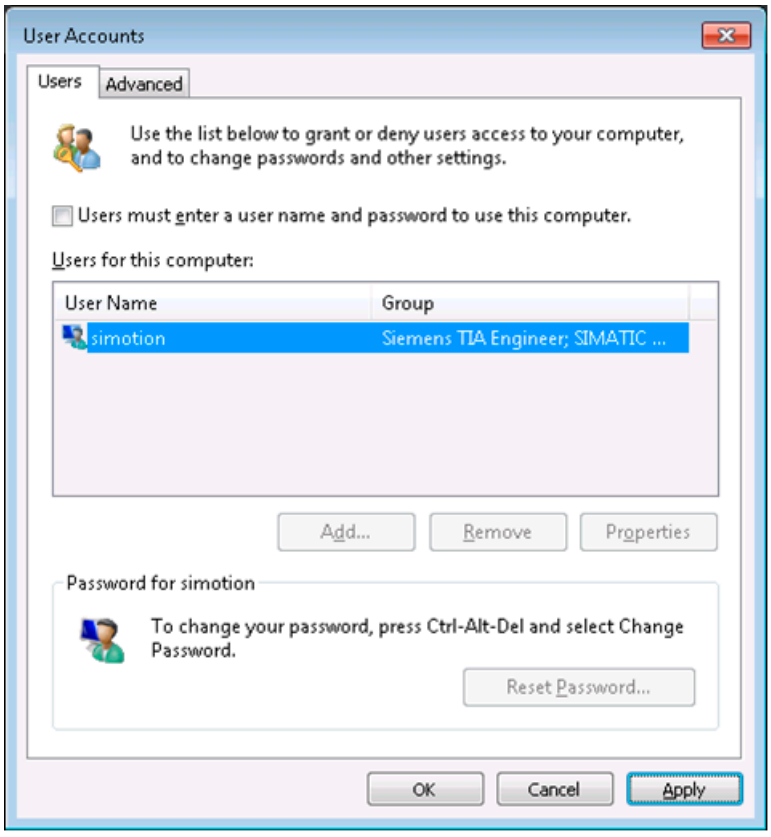


Figure 11-14 User accounts

4. Enter your changed Windows user password in the **Automatically Log On** dialog box and confirm it by entering it a second time.

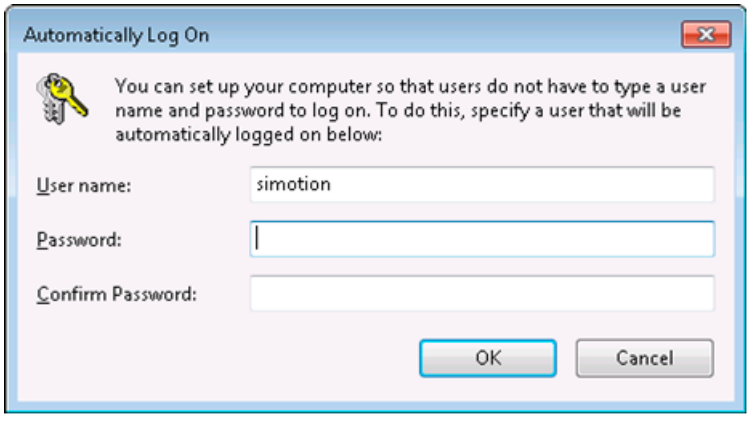


Figure 11-15 Automatically Log On dialog box

5. Confirm the password with **OK**.

Deactivating the remote desktop connection

Deactivating the remote desktop connection

If you do not use the headless mode and do not require a remote desktop connection, we recommend that you explicitly disable the remote desktop connection.

NOTICE
Deactivating the remote desktop connection
To ensure that the remote desktop connection is not used, it must be explicitly deactivated.

Procedure for deactivating the remote desktop connection

1. Using the Control Panel, select **System** > **Remote settings**.

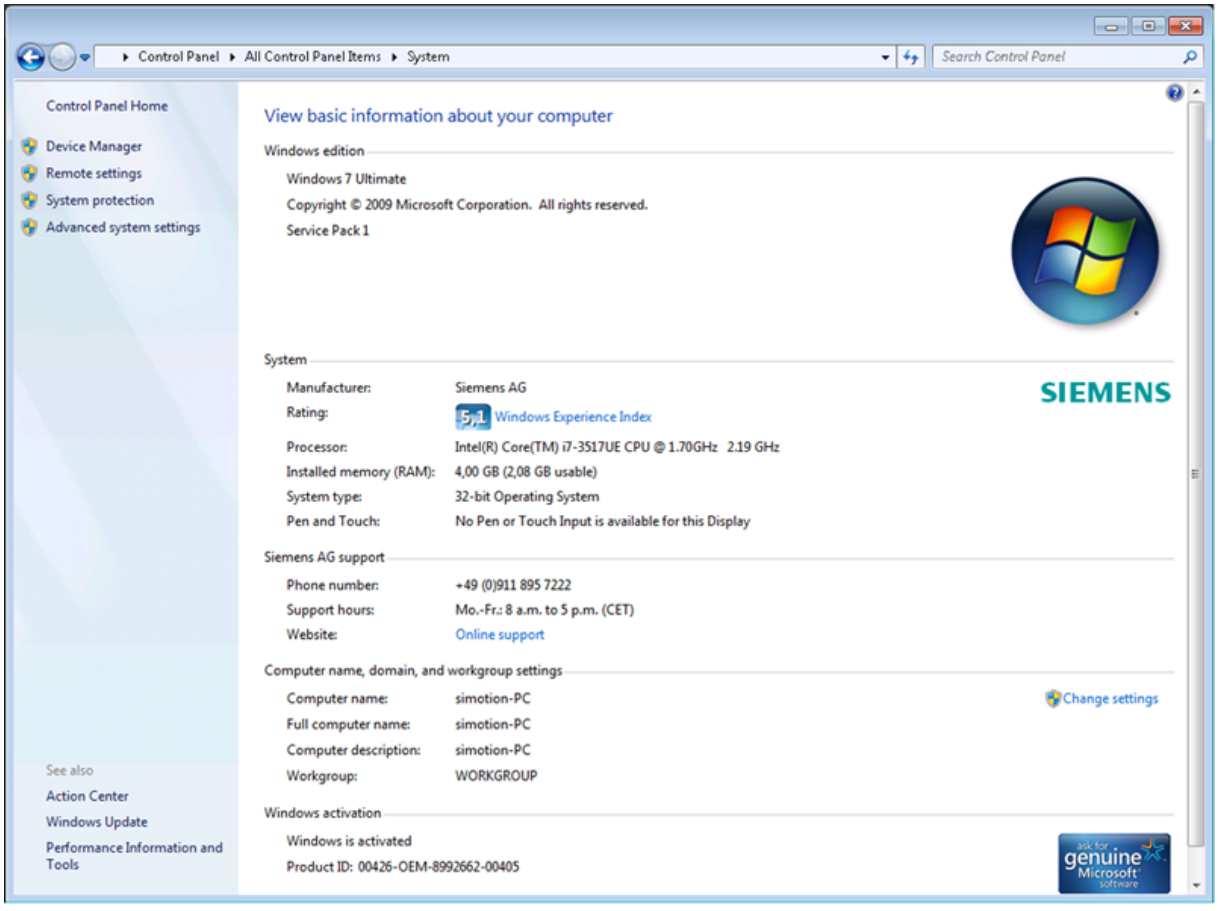


Figure 11-16 Remote settings

2. Select the **Remote** tab in the **System Properties** dialog box.

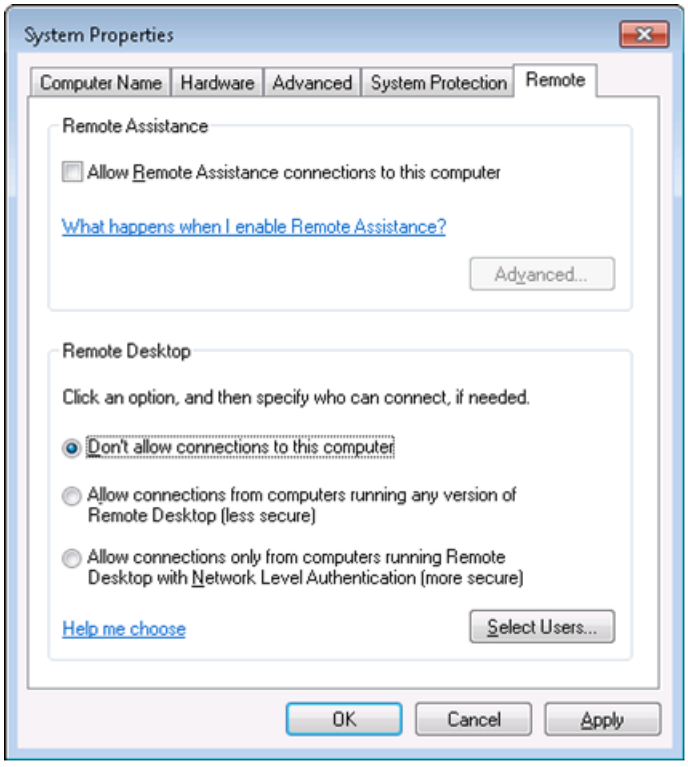


Figure 11-17 System Properties dialog box

3. Select the **Don't allow connections to this computer** option at **Remote Desktop**.
4. Confirm with **OK**.

Any remote desktop access is deactivated with this operation.

11.1.1.3 Description

System overview

Overview

SIMOTION P is a PC-based, open Motion Control System from SIMOTION.
Control, motion control, and HMI functions are executed together with standard PC applications on the SIMOTION P hardware platform. SIMOTION P combines the compatibility of the Windows operating system with real-time capability of SIMOTION P Runtime.

The fully independent SIMOTION P Runtime runs in parallel to Windows on SIMOTION P. This real-time expansion makes it possible to implement demanding motion control applications with high performance requirements on platforms of the SIMOTION P range. The hardware consists of a computing unit with innovative Intel technology, which is ready for operation at the time of delivery.

The drives and I/O devices are connected either via PROFINET onboard or IsoPROFIBUS board (optional).

The SIMATIC Flat Panel IFP1500, IFP1900 and IFP2200 can be used for the operation of the SIMOTION P320-4 hardware platform in a distributed configuration.

SIMOTION P320-4 versions

The SIMOTION P320-4 can control various I/O systems and HMI components via PROFINET onboard or the optional IsoPROFIBUS board.
If required, a USB DVD drive can be connected, for example.

The following versions of the SIMOTION P320-4 are available:

- SIMOTION P320-4 E
  with the Windows Embedded Standard 7 32-bit operating system and real-time expansion for SIMOTION.
- SIMOTION P320-4 S
  with the Windows 7 Ultimate 32-bit operating system and real-time expansion for SIMOTION.

Application

The SIMOTION P320-4 applications are directed at machines that require a high level of integration of PLC, motion control and technology functions on account of the increasing use of servo drives:

- Packaging machines
- Plastic and rubber processing machines
- Presses, wire-drawing machines
- Textile machines
- Printing machines
- Machines for processing wood, glass, ceramics, and stone
- Production lines in the renewable energy sector, e.g. solar technology, wind power installations

SIMOTION P320-4 product description

SIMOTION P320-4 overview

The SIMOTION P320-4 provides high-level industrial performance. It features:

- Compact design
- Maintenance-free operation
- High degree of ruggedness
- Long-term availability

Figure 11-18 SIMOTION P320-4 view

Figure 11-19 SIMOTION P320-4 (open perspective) with plugged-in optional IsoPROFIBUS board

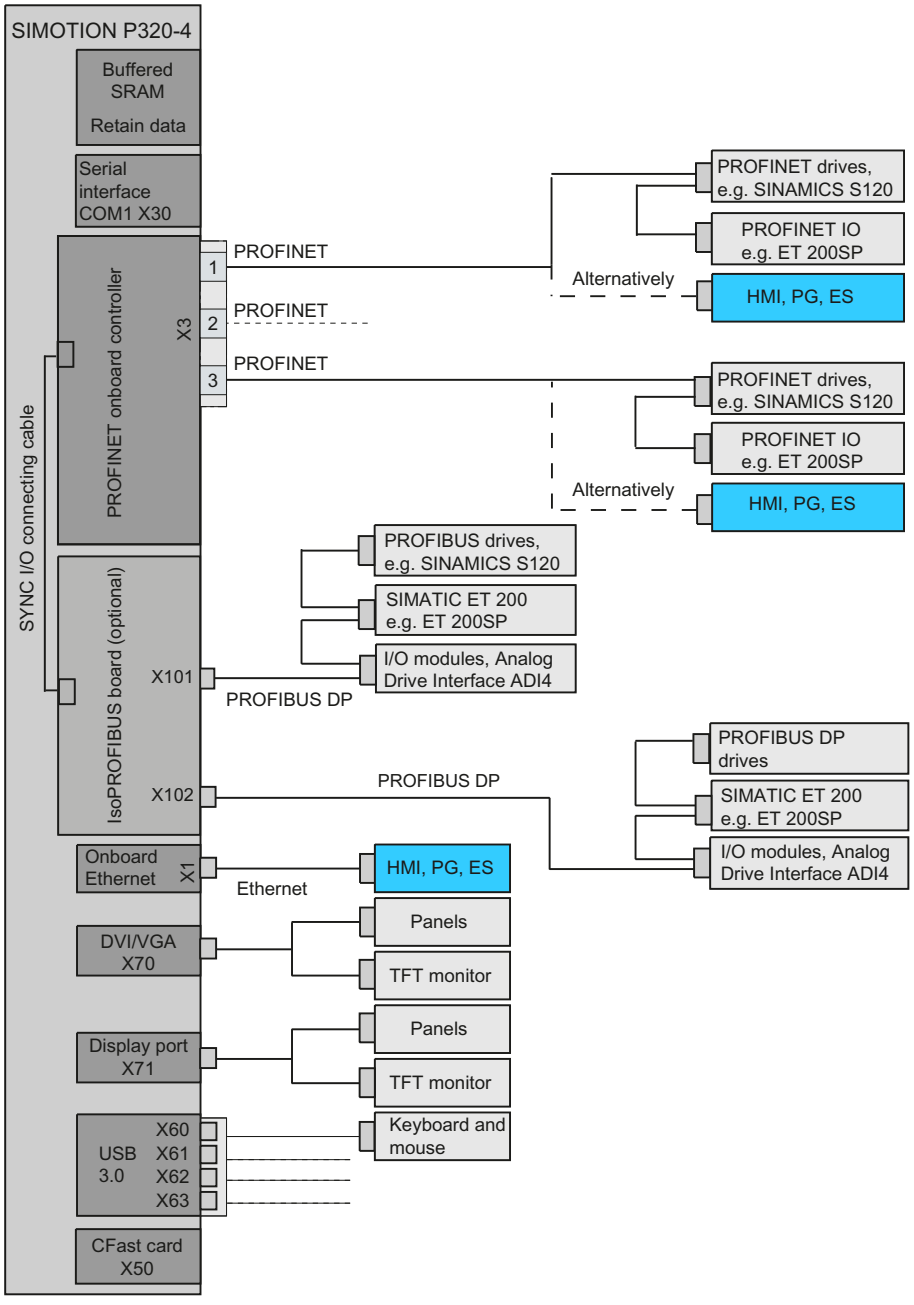
Features

Basic data	
Installation	• Standard rail mounting • Vertical mounting
Processor	
P320-4 E **P320-4 S**	Intel Core i3-3217UE, 2 x 1.6 GHz, 3 MB cache Intel Core i7-3517UE, 2 x 1.7 GHz, 4 MB cache
Main memory	4 GB DDR3 RAM
Optional IsoPROFIBUS board (PROFIBUS DP)	2 x SUB-D socket with configurable baud rates (9.6 Kbit/s - 12 Mbps)
Graphics	• Integrated Intel HD2000 or HD4000 • DVI resolution of 640 × 480 pixels up to 1920 × 1200 pixels • Display port resolution max. 1920 × 1200 pixels • Graphics memory is occupied in the main memory (dynamic UMA)
Power supply	24 VDC (-20%/+20%) max. 4 A
Operating conditions	Operation without fan
Drives and storage media	
CFast card or SSD (Solid State Disk)	Depending on the hardware version of the SIMOTION P320-4
P320-4 E	**2 x CFast card** Internal interface: CFast External interface: CFast
P320-4 S	**SSD (Solid State Disk)** Internal interface: SSD **CFast card** External interface: CFast
USB stick	External, can be connected via USB interface
Interfaces	
Serial	COM (RS 232)
Graphics	DVI-I: Suitable for use as DVI or VGA DPP++: Display port, DVI via DPP-to-DVI adapter
USB	4 × USB 3.0, simultaneous operation of high current, backward compatible with USB 2.0/1.1
Ethernet	1 × RJ45 (10/100/1000 Mbps)
PROFINET I/O	3 × RJ45 (100 Mbps)
Keyboard, mouse	Can be connected via USB interface

Software	
Operating systems	
P320-4 E	Windows Embedded Standard 7 32-bit
P320-4 S	Windows 7 Ultimate 32-bit

SIMOTION P320-4 (hardware) structure

The following figure shows the integration of the SIMOTION P320-4 with integrated PROFINET onboard and optional IsoPROFIBUS board in a target system.



HMI Human Machine Interface

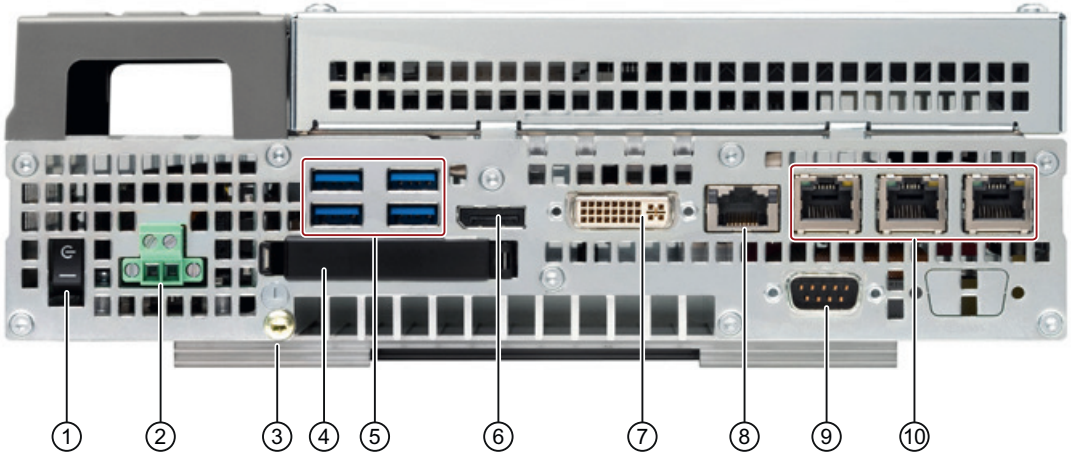
PG Programming device

ES Engineering system

Figure 11-20 SIMOTION P320-4 system overview

Interfaces and operating elements

SIMOTION P320-4



①	On/off switch	Position **OFF** is the delivery state. The on/off switch does not isolate the device from the power supply.
②	24 V DC	Power supply connection
③	Protective conductor	Protective conductor terminal
④	Memory card slot	Cover for the CFast card
⑤	4 × USB 3.0	USB 3.0 high current, backward compatible with USB 2.0/1.1
⑥	Display port	Display port connection for digital monitor
⑦	DVI-I	DVI connector for CRT or LCD monitor with DVI interface
⑧	Ethernet	RJ45 Ethernet connection for 10/100/1000 Mbps
⑨	COM1	Serial interface
⑩	3 × PROFINET	PROFINET ports with three RJ45 sockets

SIMOTION P Runtime (software) structure

SIMOTION P Runtime contains the SIMOTION P Kernel which performs the motion control and interface control.

A SIMOTION project is planned, configured, assigned parameters, commissioned and programmed via the SIMOTION SCOUT engineering system (ES).

The user data can be stored on the data media of the SIMOTION P320-4 .

Siemens WinCC flexible software can be used to visualize operational sequences or to operate the machine. Third-party systems can be linked via the OPC interface.

SIMOTION P Runtime

With the SIMOTION P320-4, the PLC and motion control functionality (starting from position controller upwards) is located centrally in a strictly deterministic task outside the Windows operating system.

Its main field of application is centralized motion control and control tasks requiring close coordination between multiple axes and/or input/output modules.

Functionality ranges from simple positioning to high-performance synchronous operation.

SIMOTION SCOUT

The SIMOTION SCOUT engineering system can be installed on the SIMOTION P320-4 S or connected via the interfaces integrated in SIMOTION P320-4 .

SIMOTION SCOUT TIA

As of version V4.5, the SIMOTION SCOUT TIA engineering system no longer supports the Windows 7 32-bit operating system and therefore cannot be installed.

HMI software

HMI/Runtime software can be operated on the same PC, e.g. the Siemens WinCC flexible software. Other software packages can be linked by means of the OPC interface.

The **HMI software** is used as the general term in the following.

Internal communication

A locally installed HMI can use the local communication to access the following:

- Variables in SIMOTION RT
- Drives on PROFINET IO or PROFIBUS DP
- Other SIMOTION devices on PROFINET IO or PROFIBUS DP

Scope of delivery

The system software supplied with the SIMOTION P320-4 is either already installed on the SIMOTION P320-4 storage medium or is ready to be installed:

- SIMOTION P Kernel
- SIMOTION IT
- SIMOTION IT VM Virtuell Machine (subject to license)

Components

The most important components of the SIMOTION P320-4 and their functions are listed below.

Distributed I/O systems (PROFINET)

Table 11-1 Components for distributed I/O

Component	Function
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities.
SIMATIC ET 200S	Finely modular I/O system for cabinet installation including motor starters, safety technology, and individual grouping of the load groups.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; SIMATIC ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200eco	I/O system with IP67 degree of protection for machine-related, cabinet-free applications, featuring a flexible and fast connection system in ECOFAST or M12.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related, cabinet-free applications, including motor starters.

Note

Note that not all modules of the above-mentioned I/Os or I/O systems have been released for SIMOTION.

There may also be system-dependent functional differences with regard to the use on SIMOTION and on SIMATIC.
For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

For a detailed and routinely updated list of I/O modules enabled with SIMOTION as well as application information, visit us online at:

SIEMENS Industry Online Support - Product support - Supplementary system components (https://support.industry.siemens.com/cs/de/en/view/11886029)

Further modules are integrated via the GSD file of the device's manufacturer.

Note

Note that in individual cases, additional supplementary conditions must be fulfilled.

Drive systems (via PROFINET IO)

Table 11-2 List of typically connected drive systems

Component	Function
SINAMICS S120	Servo drives, innovative single-axis and multi-axis solution

Optional components

Table 11-3 Optional components

Centralized I/O	Function
IsoPROFIBUS board	PROFIBUS connection
UPS	Uninterruptible Power Supply The drivers for the UPS are not pre-installed on the PC. Install the current drivers that are supplied with the UPS.

Note

An IsoPROFIBUS board may be inserted in addition to the PROFINET onboard.

HMI and SIMOTION SCOUT

HMI and SIMOTION SCOUT Overview

The operational sequences on the SIMOTION P320-4 are either monitored via the HMI system (Human Machine Interface) or the SIMOTION SCOUT engineering system.

The HMI or ES software can be connected using the following versions:

- Local (Page 7858)
- via PROFINET (Page 7871)
- via Ethernet (Page 7872)
- via IsoPROFIBUS (optional) (Page 7873)

Local communication of the HMI/ES

If HMI or ES is installed locally on the SIMOTION P320-4, **PC internal** can establish a connection to the local **SIMOTION P Runtime** . Only a local connection is permitted.

See section Local HMI or ES on SIMOTION P320-4 (Page 7858)

Communication of the HMI / ES via PROFINET / Ethernet / PROFIBUS

By using one of these communication versions, you can control and monitor several SIMOTION devices.

Local HMI or ES on SIMOTION P320-4

The model described below is used for the communication of the SIMOTION P320-4 with a local HMI or ES. The SIMOTION SCOUT engineering system or the HMI system can be installed directly on the **SIMOTION P320-4 S**.

For this model, you must set the communication (access point) to **TCP/IP** .

The SIMOTION P320-4 is already configured for this communication version at the time of delivery.

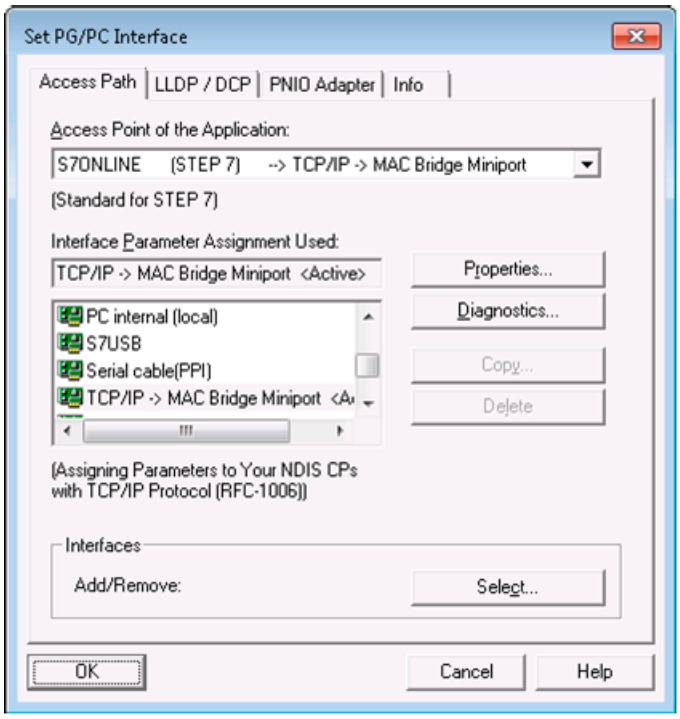


Figure 11-21 Set PG/PC Interface - TCP/IP -> MAC Bridge Miniport dialog box

Note

Local communication via Ethernet (TCP/IP)

For the communication via Ethernet (TCP/IP) to function locally, a link must be available to the Ethernet interface or a LAN cable must be connected **or** the MAC bridge set up.

Note

Setting up the MAC bridge miniport

See also Section Setting up a network bridge - MAC bridge miniport (Page 7860)

Note

Only HMI-Runtime can be installed on the **SIMOTION P320-4 E** version.

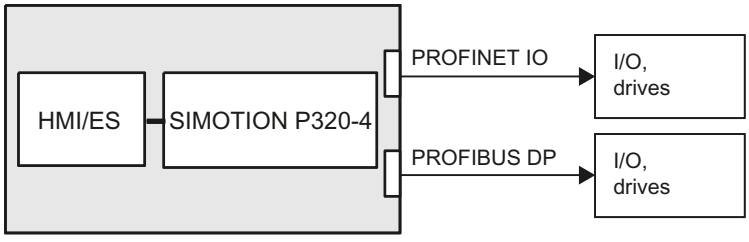


Figure 11-22 Model: Local HMI or ES

Alternatively to TCP/IP , SIMOTION SCOUT can also communicate locally via PC internal (local) .

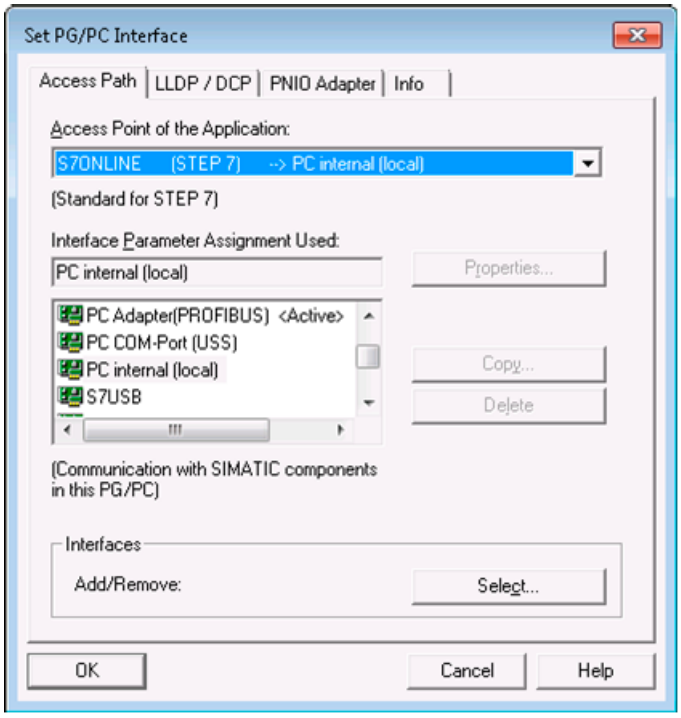


Figure 11-23 Set PG/PC Interface - PC internal (local) dialog box

Setting up a network bridge - MAC bridge miniport

Note

If you are working with SIMOTION P320-4 version V4.5, the Network Bridge is already preset.

The network bridge must be preset, configured via script and set up as described in the following. Note that the MAC address of the network bridge is different to the MAC address of the Ethernet adapter.

The following is a step-by-step description of how the network bridge **MAC Bridge Miniport** is set up.

Network connections initial situation

Open the **Network and Internet** via the Control Panel.

- Select **Control Panel** > **All Control Panel Items** > **Network and Sharing Center** > **Change adapter settings**.
- Open the **Network and Internet** > **Network Connections** dialog box.
  The network connections of your PC are shown here.

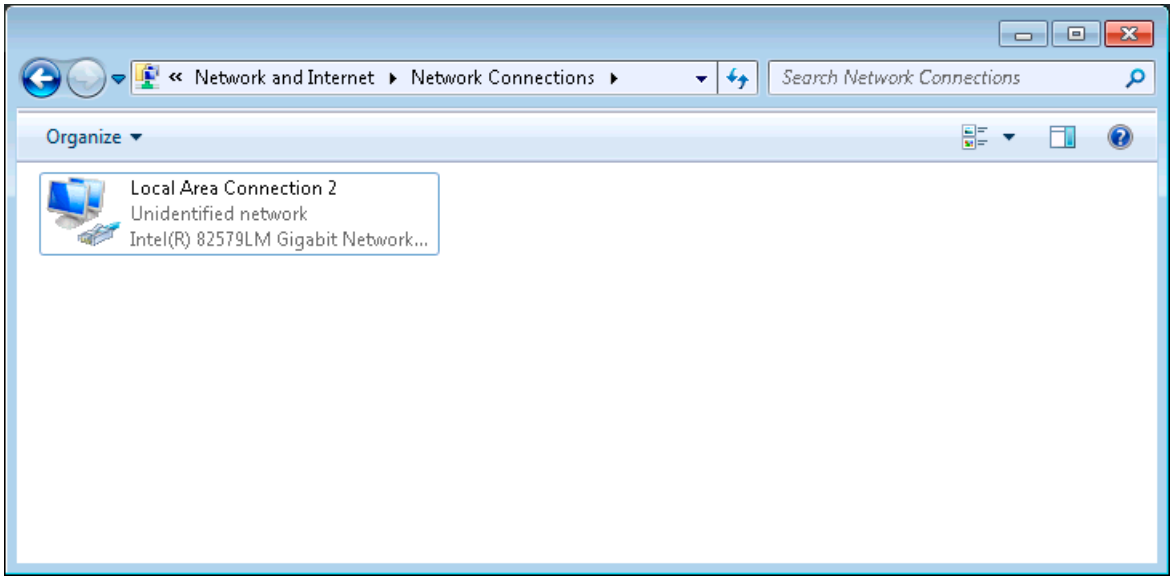


Figure 11-24 Initial situation - Network connections

Device Manager

Open the Device Manager via the Control Panel:

Control Panel > **All Control Panel Items** > **System** > **Device Manager**.

1. Select the **Network adapters** in the dialog box below the "SIMOTION PC". The available network connections are shown here.

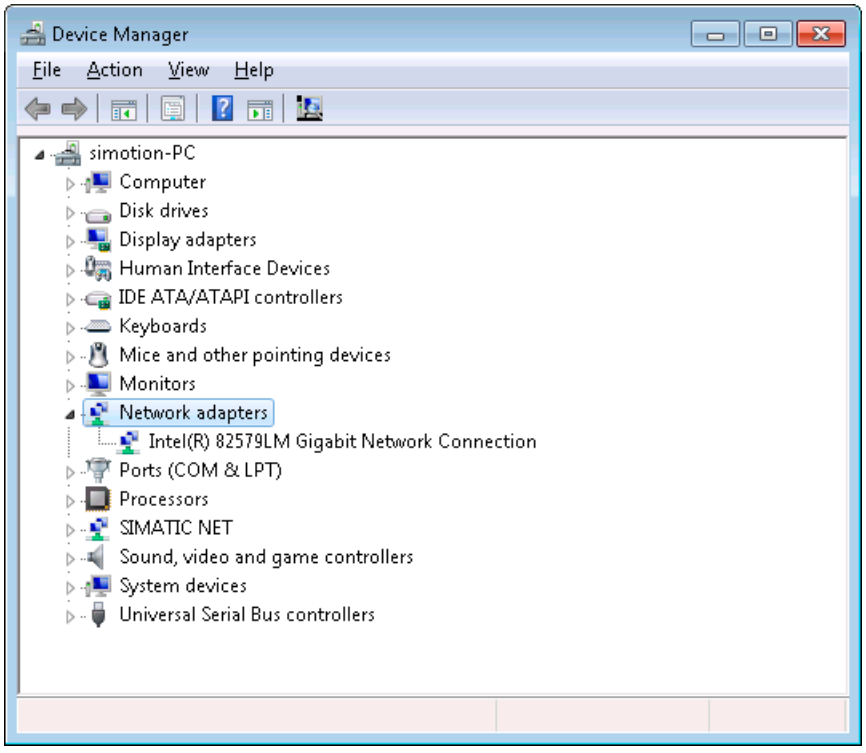


Figure 11-25 Opening the Device Manager

2. Install the **Microsoft Loopback Adapter** via the Device Manager.
3. To do this, select the menu **Action** > **Add legacy hardware**. The **Add Hardware Wizard** then opens.

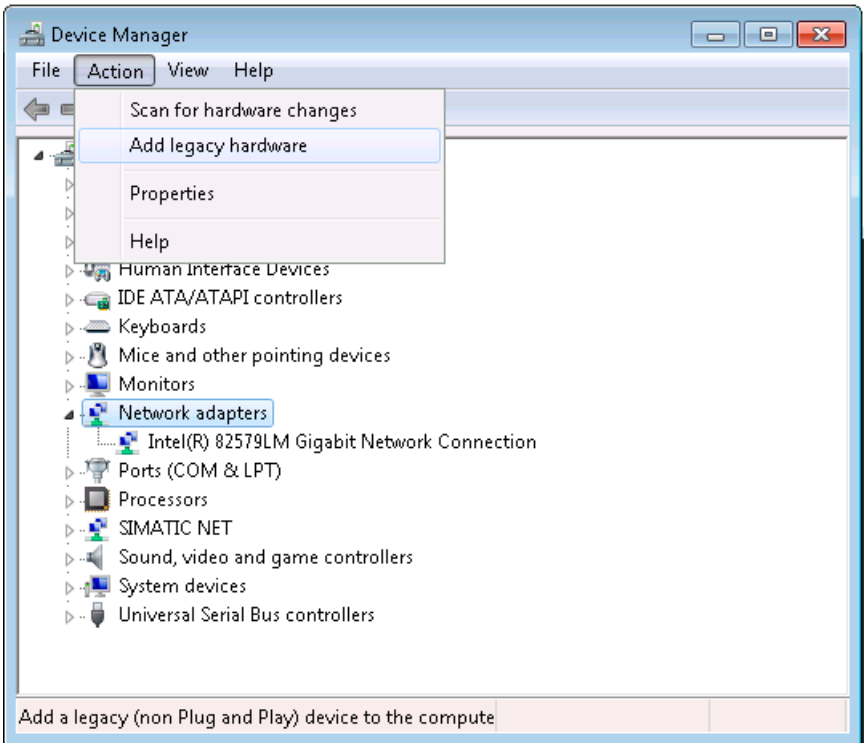


Figure 11-26 Device Manager - Action menu

Add Hardware Wizard

Use the **Add Hardware Wizard** to install the network bridge.

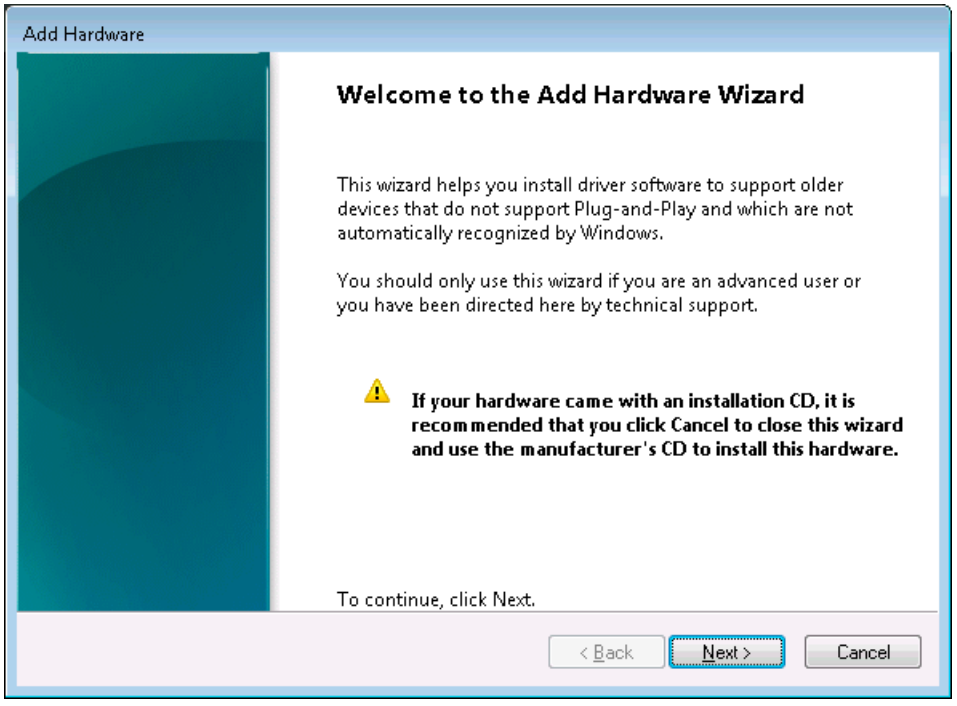


Figure 11-27 Add Hardware - Starting the wizard

1. Click **Next** to start the software wizard.
2. Select the option: **Install the hardware that I manually select from a list (Advanced)** and continue in the wizard with **Next**.

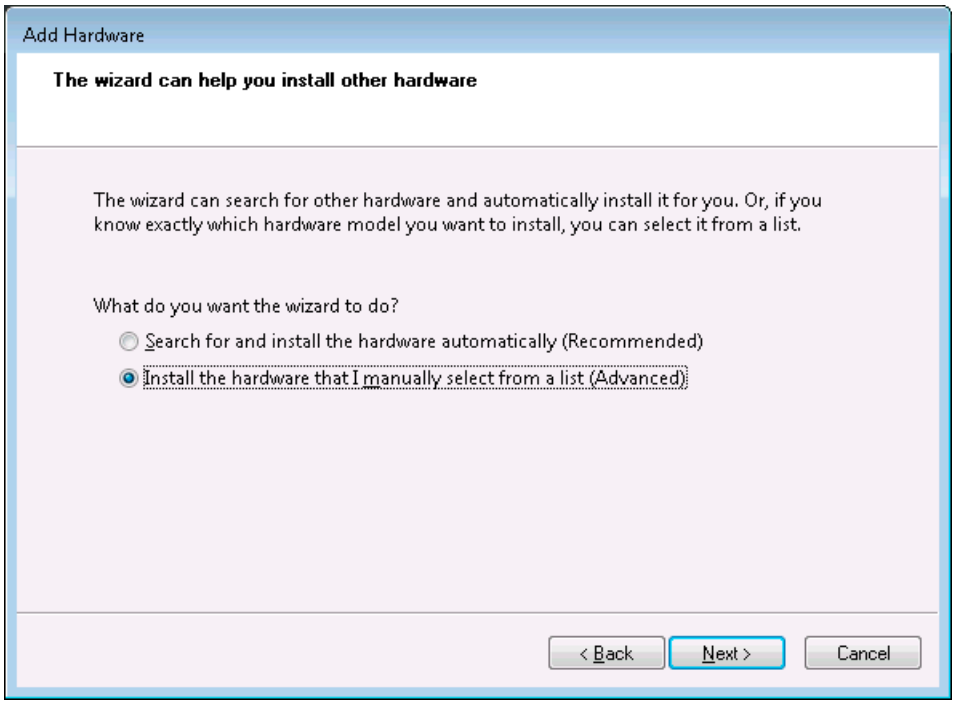


Figure 11-28 Add Hardware - Manual selection from a list

3. Select **Network adapters** in the dialog box now open and continue in the wizard with **Next**.

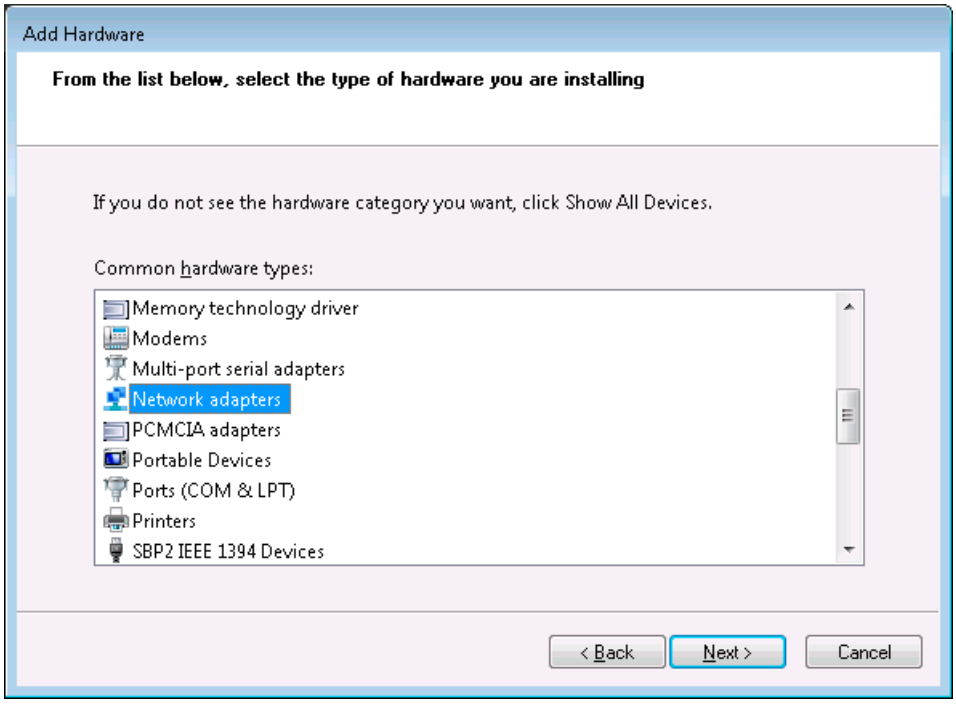


Figure 11-29 Select Add Hardware - Network adapters

4. In the following dialog box, select the **Microsoft Loopback Adapter**.
5. To do this, select Microsoft at **Manufacturer** and the Microsoft Loopback Adapter at **Network Adapter**.

6. Continue in the wizard with **Next**.

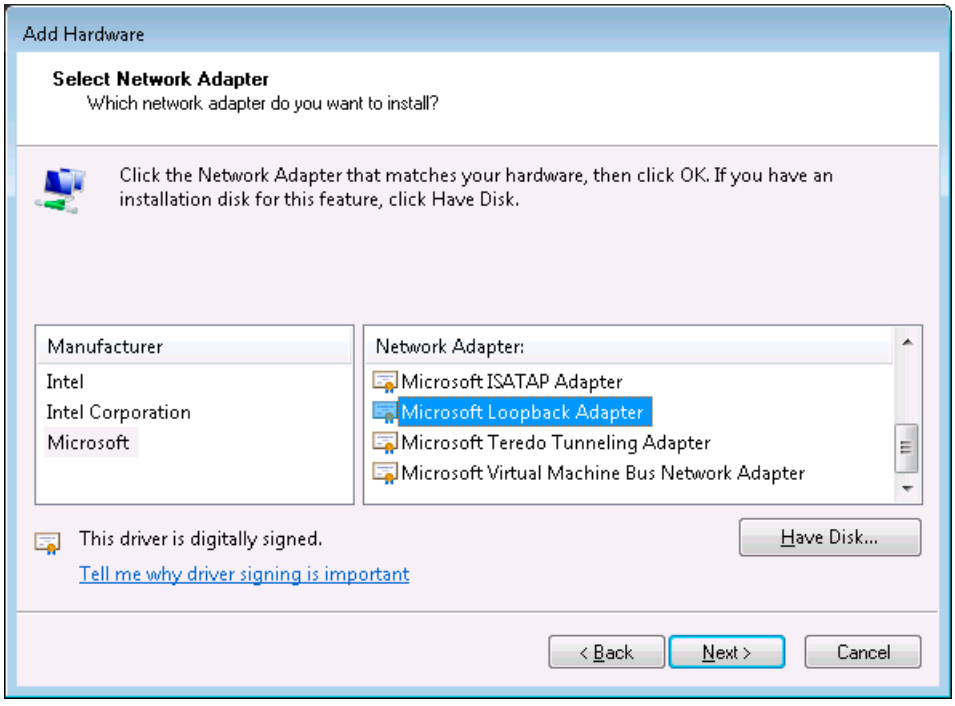


Figure 11-30 Add Hardware - Selecting the desired Network Adapter

7. The selected Network Adapter is shown in the following dialog box.
   Click **Next** to confirm and install.

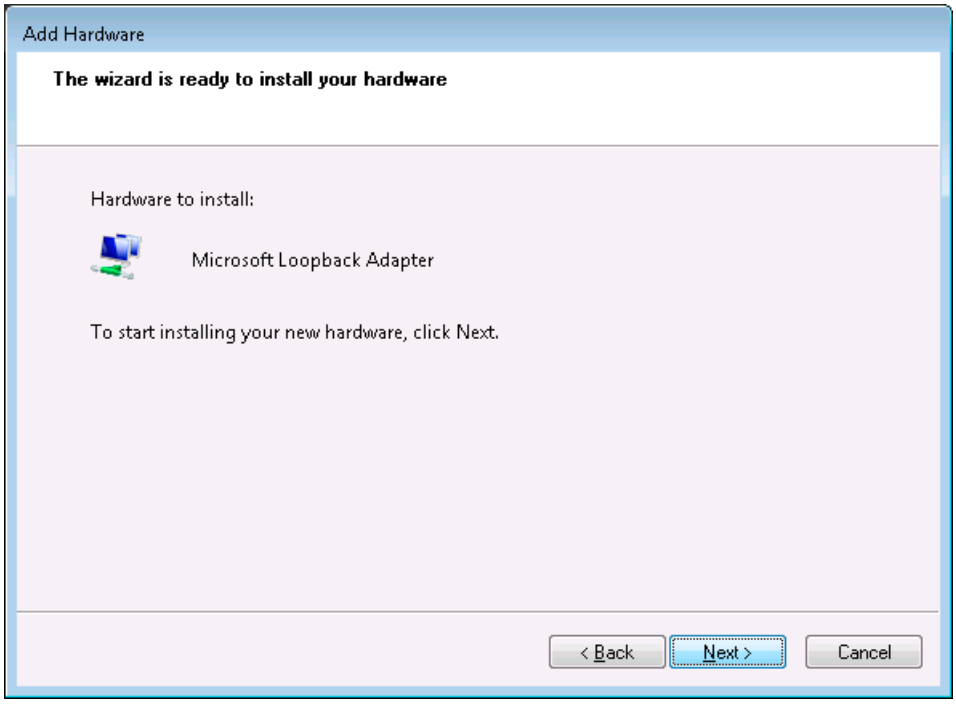


Figure 11-31 Add Hardware - Selected Network Adapter can be installed

8. After finishing the installation, the following dialog box is displayed.
   Exit the wizard with **Finish**.

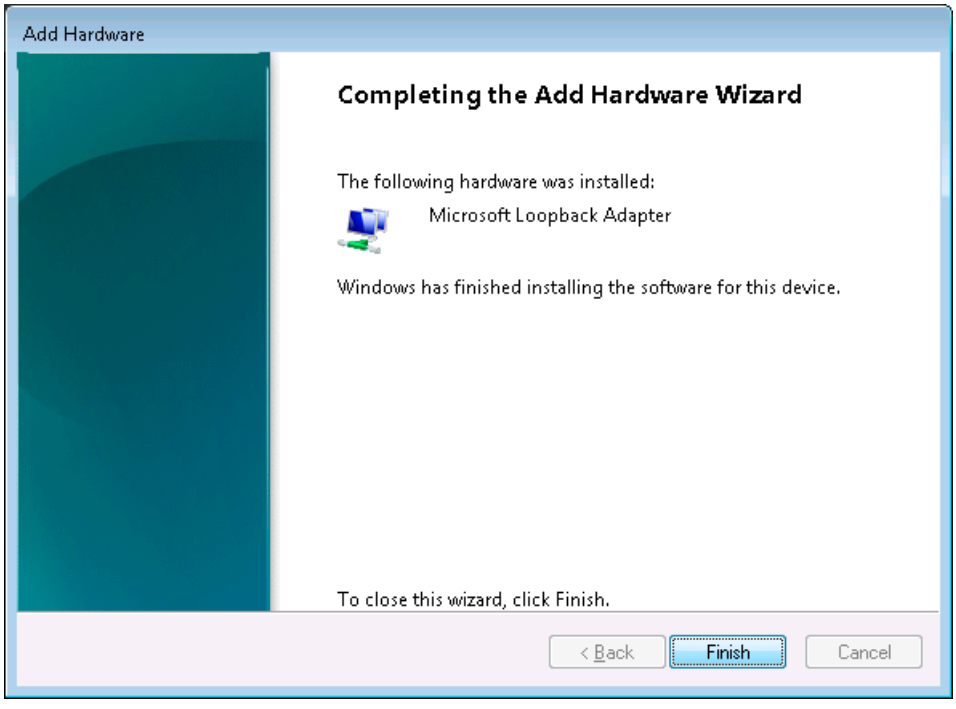


Figure 11-32 Add Hardware - Installation of the Microsoft Loopback Adapter is finished

Checking the installation of the network adapter

You can check the successful installation of the **Microsoft Loopback Adapters** in the

- Device Manager
- Internet connections

Device Manager

The Microsoft Loopback Adapter is displayed at **Network adapters** in the Device Manager.

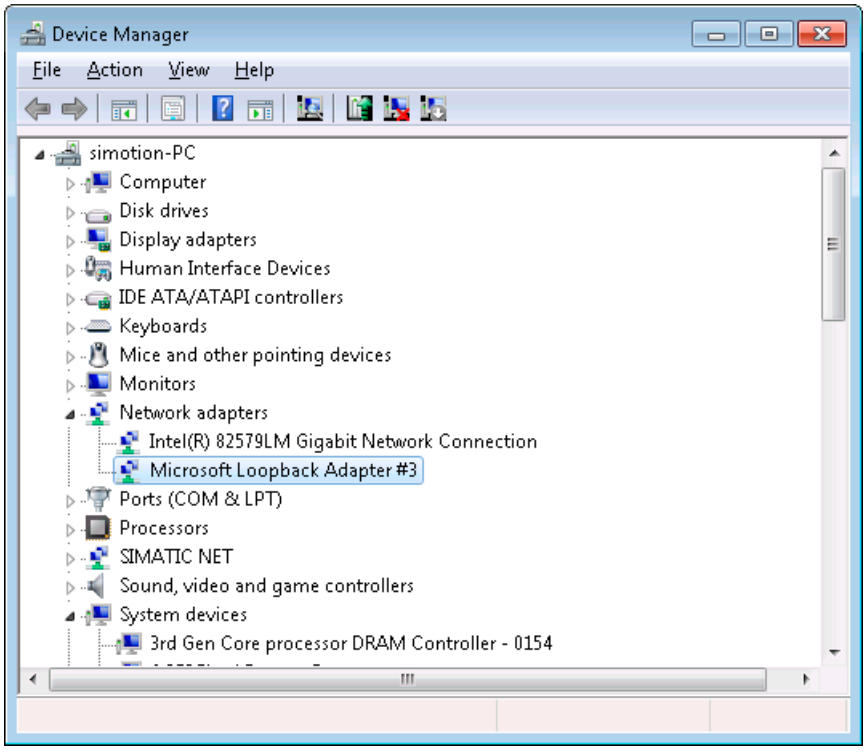


Figure 11-33 Device Manager - Microsoft loopback adapter

Network connections

The newly installed Microsoft Loopback Adapter is also displayed at **Network Connections**.

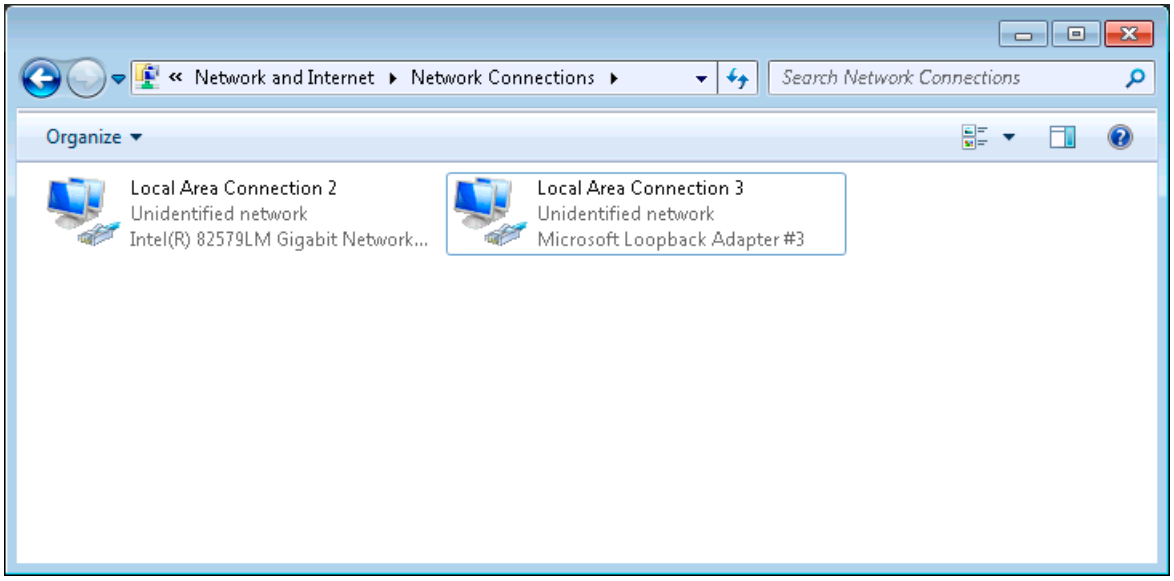


Figure 11-34 Network connections - Display of the newly installed Microsoft loopback adapter

Creating the network bridge

To activate the network bridge, proceed as follows:

1. Select both adapters at Network Connections
2. Select Bridge Connections via the context menu.

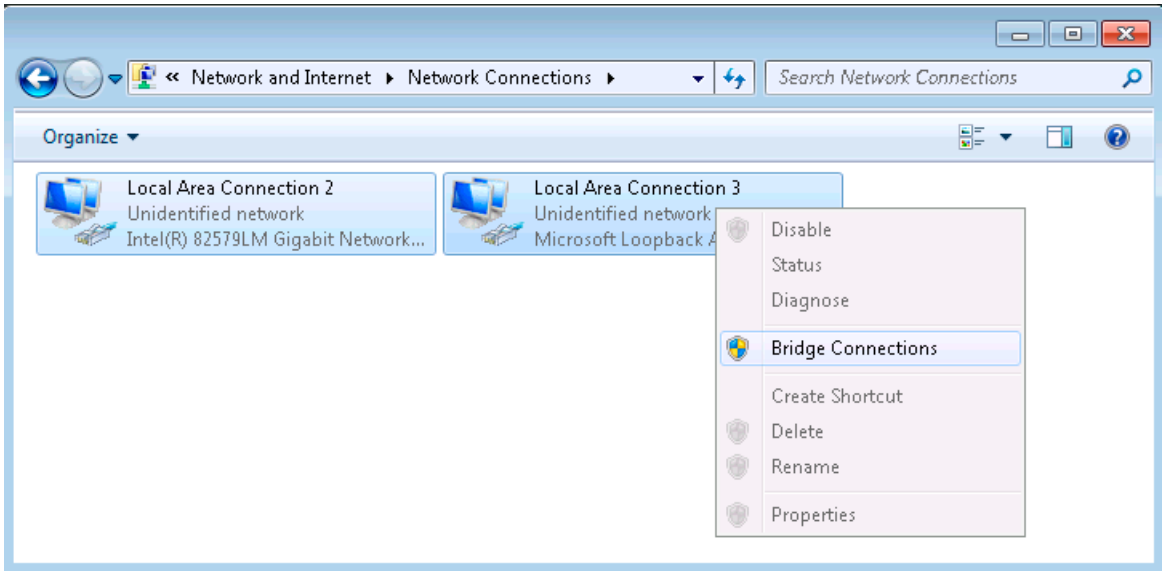


Figure 11-35 Network Connections - Setting the Bridge Connection

3. The Network Bridge MAC Bridge Miniport is displayed.

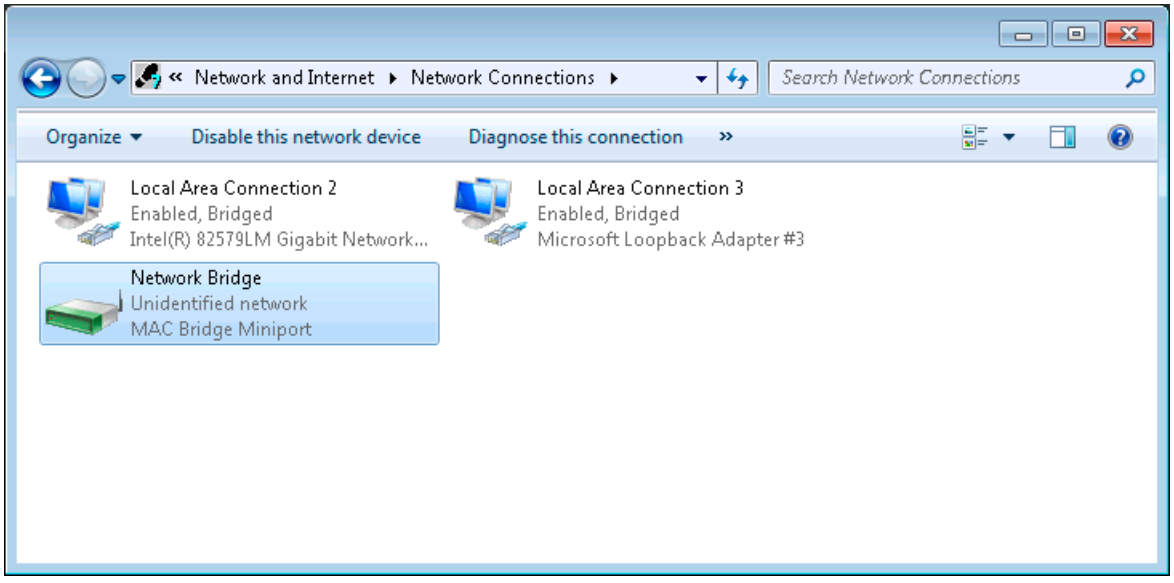


Figure 11-36 Network Connections - Display of the Network Bridge MAC Bridge Miniport

4. To initialize the network bridge, execute the **sp_bridge.cmd** Windows command script.
   The additional script can be found at C:\SiMotion\tools\sp_bridge.cmd
   After executing the batch file, the network bridge has been set up.

Setting "Set PG/PC interface"

Now open the PG/PC interface to complete the setting for the **Network Bridge** .

1. Open the **Set PG/PC Interface** dialog box via the **Control Panel** entry point.
2. Deactivate the **MAC Bridge Miniport DCP** checkbox in the **LLDP / DCP** tab.
   If you do not deactivate the MAC Bridge Miniport DCP, the SIMOTION P320-4 may not be found via Accessible nodes.

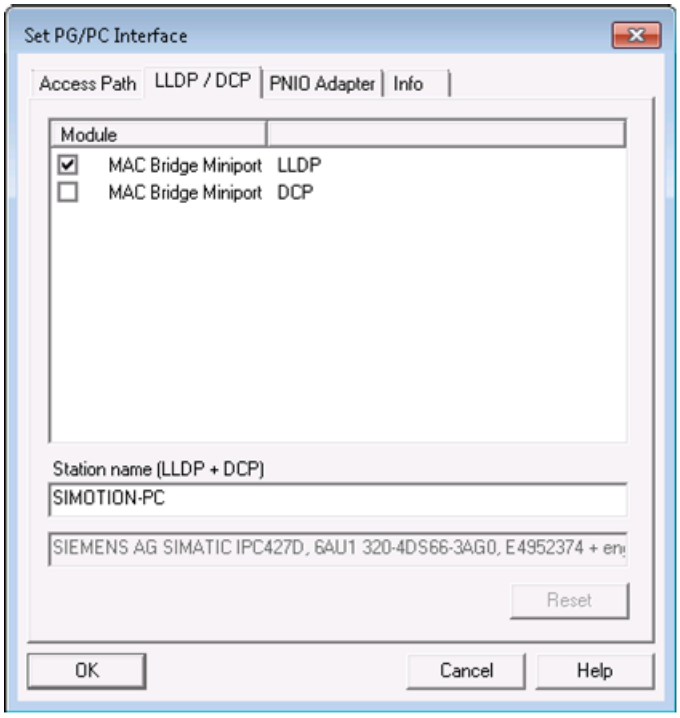


Figure 11-37 Set PG/PC interface

The settings for the network bridge have now been completed.

HMI or ES via PROFINET

Networking via PROFINET

PROFINET is the innovative and open Industrial Ethernet standard (EN 61158) for industrial automation. With PROFINET, devices can be linked up from the field level through to the management level.

With PROFINET, an external HMI or ES can also be integrated into the network and data can be exchanged directly with the SIMOTION P320-4 device.

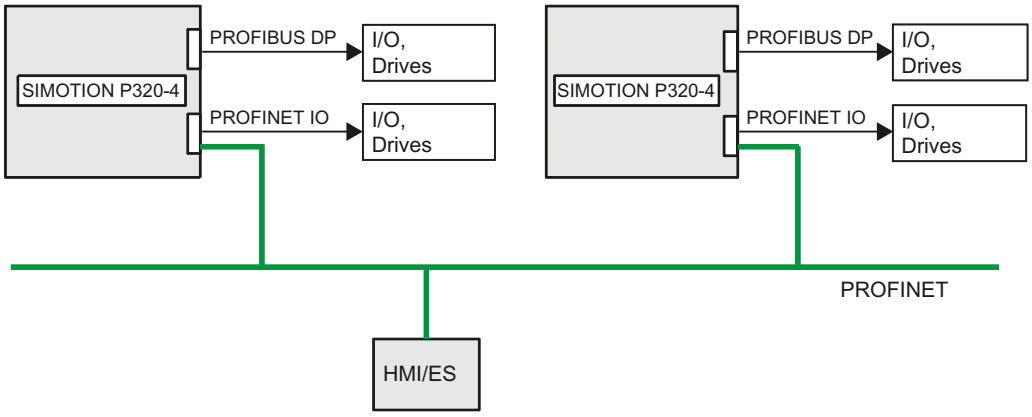


Figure 11-38 Model: Networking via PROFINET

References

You can find further information on the HMI in the documentation:

- SIMOTION Runtime Basic Functions, Section HMI (Human Machine Interface) coupling.
- SIMOTION SCOUT, Configuration Manual
- SIMOTION SCOUT TIA, Configuration Manual
- SIMOTION SCOUT TIA Device Proxy, Configuration Manual

HMI or ES via Ethernet

Networking via Ethernet

A complex interconnection with several HMIs or ESs is only possible using an Ethernet communication. This allows both an external HMI or ES to access several SIMOTION devices or one SIMOTION device to access another one, for example, to display the production data.

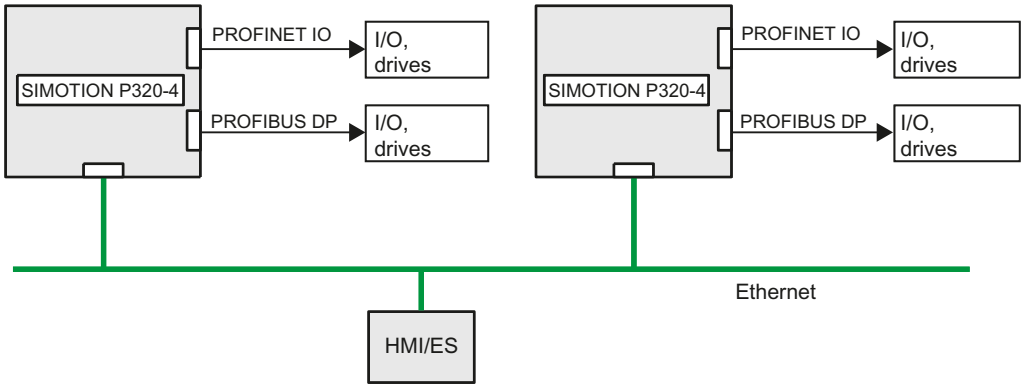


Figure 11-39 Model: Networking via Ethernet

The following services are possible on a SIMOTION device via Ethernet:

- HMI software
  WinCC flexible or the OPC server can use Ethernet to access one or more SIMOTION devices.
- SIMOTION SCOUT Engineering System
  SIMOTION SCOUT can also use Ethernet to access one or more SIMOTION devices.
- SIMOTION IT
  SIMOTION P320-4 offers communication with standard IT protocols (HTTP) over the integrated Ethernet interface.
  This makes it possible to access data or diagnostic information in the SIMOTION P320-4 from any location via intranet or the Internet.
- HTTP (browser, OPC XML)
  Setting in WebCFG.xml - default 80
- HTTPS (browser, OPC XML)
  Setting in WebCfg.xml - default 443
- FTP 21
- Telnet

References

You can find further information in the documentation for SIMOTION IT:

- SIMOTION IT Diagnostics and Configuration
- SIMOTION IT Programming and Web Services
- SIMOTION IT Virtual Machine and Servlets

HMI or ES via IsoPROFIBUS (optional)

Networking via PROFIBUS DP

If you connect a central HMI via PROFIBUS DP, the X101 and X102 interfaces of the IsoPROFIBUS board are available.

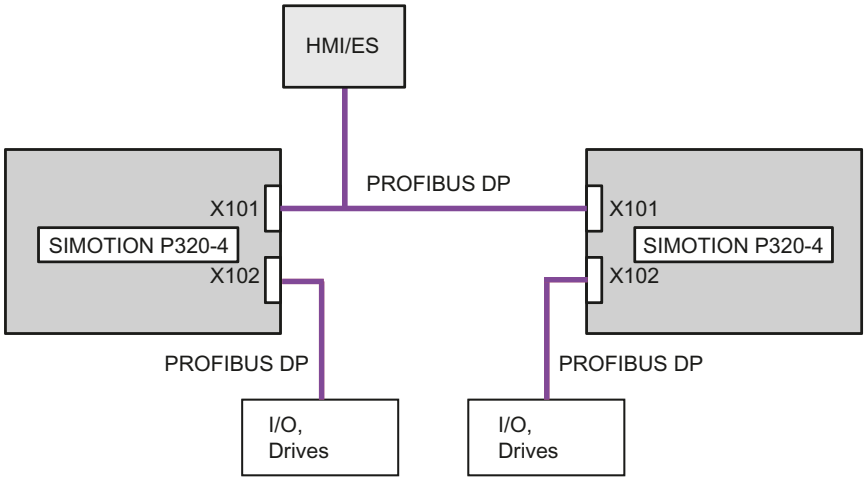


Figure 11-40 Model: Networking via PROFIBUS DP

References

You can find further information on the HMI in the documentation:

- SIMOTION Runtime Basic Functions, section HMI (Human Machine Interface) coupling.

11.1.1.4 Use planning

The use planning is described in the SIMOTION P320-4 E / P320-4 S manual.
There you receive the following information:

- Unpacking and checking the delivery
- Device identification data
- Information about the permitted mounting positions

- Environmental conditions
- Electromagnetic compatibility
  Link to further information about relevant standards for electromagnetic compatibility (EMC).

11.1.1.5 Mounting

Installation overview

The SIMOTION P320-4 is installed in a cabinet or in a controller housing.

Unlocking Windows

Note

Windows locked

Windows may be accidentally locked, e.g. through shortcut key Ctrl+L. If you do not know the password for the SIMOTION P320-4, please contact the Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/).

Operating mode

Two operating modes are available for the installation of a SIMOTION P320-4:

- SIMOTION P320-4 - Headless operation (Page 7874)
- Distributed configuration of the SIMOTION P320-4 with a remote panel (Page 7877)

Headless operation

SIMOTION P320-4 - Headless operation

The SIMOTION P320-4 is operated without a screen, e.g. SIMATIC panel, SIMOTION P320-4 is installed in a cabinet.

The maintenance and configuration is performed via a computer that accesses the SIMOTION P320-4 by means of remote desktop.

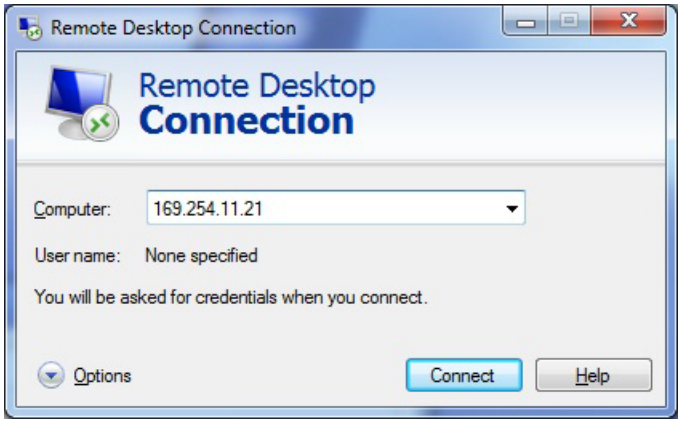


Figure 11-41 Headless to be accessed by means of remote desktop connection

Note

Headless operation or remote desktop connection

To log in via a remote desktop connection, you require a user name and a password.

If you do not have a password, please contact the hotline.

Siemens Industry Online Support (https://support.industry.siemens.com/cs/ww/en/)

Deactivating the remote desktop connection

Deactivating the remote desktop connection

If you do not use the headless mode and do not require a remote desktop connection, we recommend that you explicitly disable the remote desktop connection.

NOTICE
Deactivating the remote desktop connection To ensure that the remote desktop connection is not used, it must be explicitly deactivated.

Procedure for deactivating the remote desktop connection

1. Using the Control Panel, select **System** > **Remote settings**.

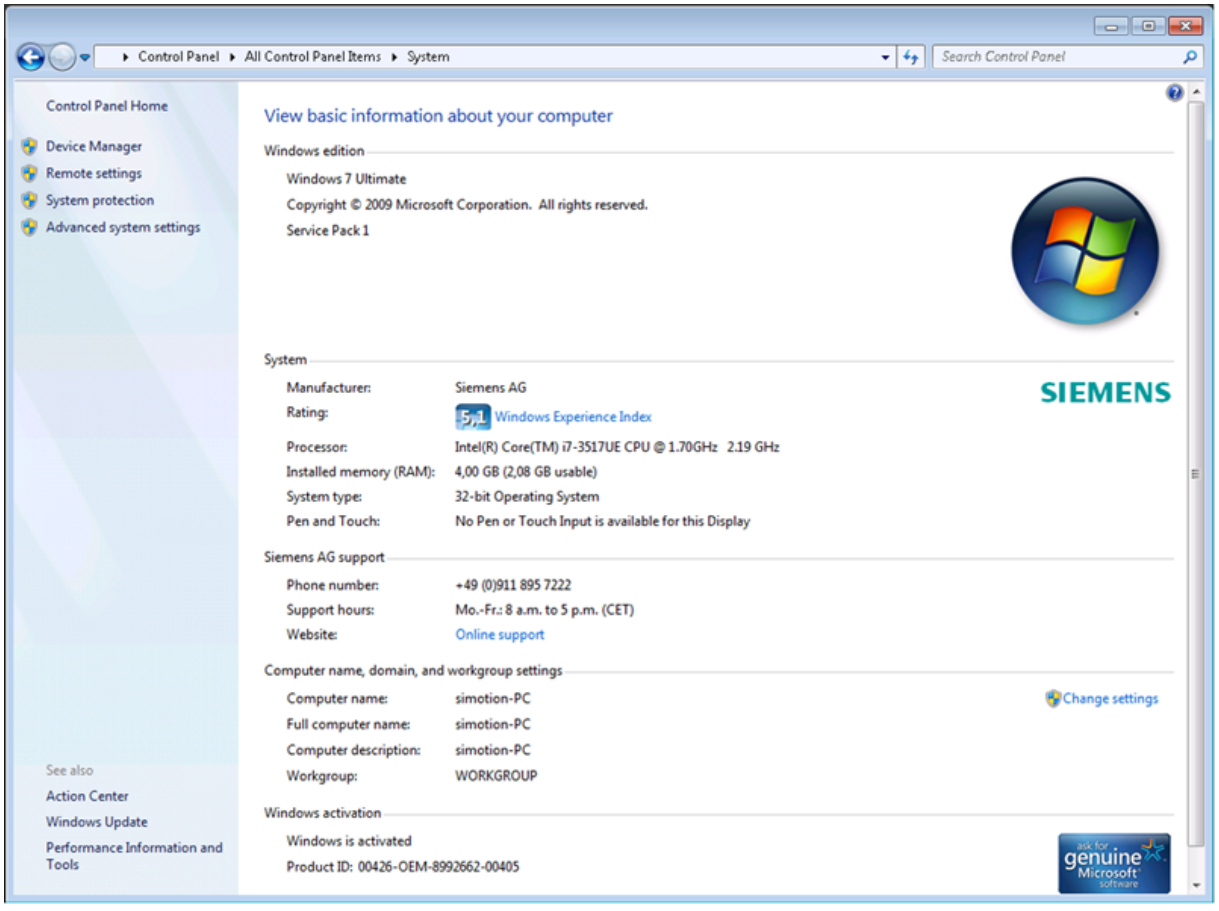


Figure 11-42 Remote settings

2. Select the **Remote** tab in the **System Properties** dialog box.

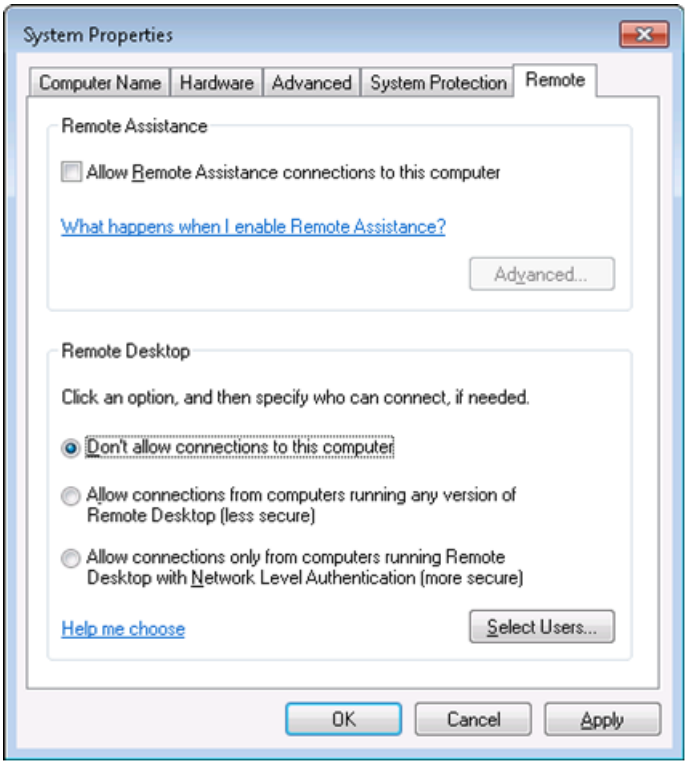


Figure 11-43 System Properties dialog box

3. Select the **Don't allow connections to this computer** option at **Remote Desktop**.
4. Confirm with **OK**.

Any remote desktop access is deactivated with this operation.

Decentralized structure

Distributed configuration of the SIMOTION P320-4 with a remote panel

The distributed configuration provides greater flexibility in terms of location and enables the SIMOTION P320-4 to be positioned in non-critical areas of the plant (e.g. cabinet).

The distributed configuration of a panel with the SIMOTION P320-4 can be performed via the DisplayPort or DVI connection. Please note the cable lengths that are supported for the panel.

Important information about installation

When installing the unit in an enclosed housing, make sure that sufficient space is available for air circulation.

Ensure that the clearance to other components or the sides of cabinets is at least 50 mm above and 100 mm below the device.

The maximum ambient temperature of the SIMOTION P320-4 depends on the mounting position:

- Horizontal (preferred position): Maximum +55° C (in RAL)
- Vertical (power supply at the top): Maximum +45° C
- Suspended: Maximum +40° C
- Vertical mounting: Maximum +45° C

RAL = Restricted Access Location (installation of device in operating facilities with restricted access, for example, a locked control cabinet)

Further information can be found in the SIMOTION P320-4 E / P320-4 S Manual.

Withdrawing/inserting the CFast card

SIMOTION P320-4 has an external slot for a CFast card on the interface side. Always use SIMATIC IPC CFast cards for industrial applications.

The CFast card is mandatory for operation of the SIMOTION P320-4. The CFast card is supplied with the SIMOTION P320-4.

NOTICE
Damage to the device The CFast and CompactFlash connections are not compatible. The device will be damaged. Use the slot described in this section only for a CFast card.

Note

Note the version of the CFast card

Note the following:

- **Always** insert a CFast card as of production version 02.
- Always replace a CFast card with a CFast card of the same or higher production version.
- The production version can be found on the CFast card (see marking).

CFast card - ordering data			
Designation	**Article number**	**Further information**	
CFast	6ES7648-2BF10-0XG0	4 GB	INDUSTRIAL GRADE

Note

Only use the CFast card specified with the article number above for the external interface.

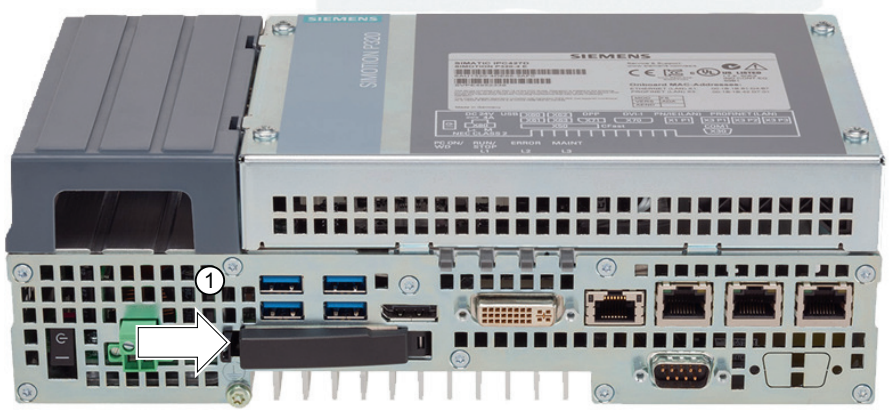
Requirement

- The device is switched off.
- SIMATIC IPC CFast card approved for industrial applications.

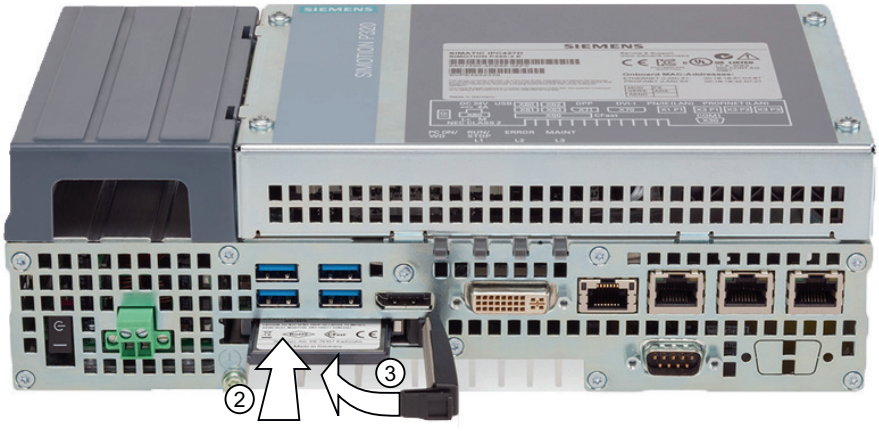
Withdrawing the CFast card

To withdraw the CFast card, proceed as follows:

1. Release the lock of the cover. Push against the cover in the direction indicated.
   Open the cover completely.



2. Remove the CFast card from the memory card slot by pushing it in until it is ejected by about 5 mm (ball-point pen mechanism).
   Then withdraw the CFast card from the memory card slot.



3. After removing the CFast card, close and lock the cover.

Inserting the CFast card

NOTICE
Inserting the CFast card If you insert a CFast card into a built-in SIMOTION P320-4 in a system, observe the safety regulations for working with electrical systems. Insert the CFast card carefully into the slot, and without applying excess force.

To insert the CFast card into the memory card slot again:

Push the CFast card into the memory card slot until it locks into place (ball-point pen mechanism).

Automatic restart for SIMOTION P Runtime

Note

Not a Windows restart

Note that this is not a Windows restart!

Starting the SIMOTION P320-4 with withdrawn CFast card

Power-up of the SIMOTION P Runtime is **not** possible without the CFast card. A prompt referring to the missing CFast card is displayed. The prompt is displayed until the CFast card is inserted. The power-up is then continued and the prompt closed automatically.

The automatic restart of the SIMOTION P Runtime is performed when the **Enabled** checkbox is activated in the **SIMOTION P Control Manager** in the **Administration** tab at **Automatic Restart By Plug-In External Memory Card**.

Figure 11-44 Prompt about the missing CFast card - automatic restart Enabled

Note

The prompt does not close automatically after inserting the CFast card.

If the prompt shown above is not automatically closed after inserting the CFast card and there is no automatic restart of the SIMOTION P Runtime, check the BIOS settings.

See Section Hotplug Enabled BIOS settings (Page 7985).

If the **Disabled** status has been selected in the SIMOTION P Control Manager, there is no automatic restart of the SIMOTION P Runtime after the CFast card has been inserted. In this case, the following prompt is displayed.

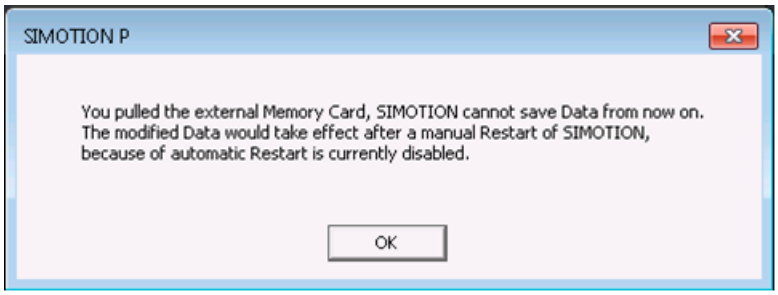


Figure 11-45 Prompt about the missing CFast card - automatic restart Disabled

CFast card withdrawn during operation

If the CFast card is withdrawn during normal operation, the SIMOTION P Runtime goes into the **General reset request** state.

This state can be exited through MRES (general reset) on the mode selector or a restart via SIMOTION P State.

In addition to the general reset request, a prompt is also displayed informing about the withdrawn CFast card and whether an automatic restart of the SIMOTION P Runtime will be performed.

Note

Automatic restart for SIMOTION P Runtime setting

The automatic restart settings are made in the SIMOTION P Control Manager.

The default setting in the Administration tab at Start SIMOTION P is Enabled.

Further information can be found in Section **SIMOTION P Control Manager** > **Administration tab** > Automatic Restart By Plug-In External Memory Card (Page 7921).

Installing the SIMOTION P320-4 in the control cabinet

Notes on installation

Requirement

Before you install the device, observe the following installation notes:

NOTICE
Observe the DIN/VDE requirements Adhere to the relevant DIN/VDE requirements or the country-specific regulations when installing in cabinets.

NOTICE
Observe UL508 For use in the area of **Industrial Control Equipment** (**UL508**), remember that SIMOTION P320-4 is classified as **Open type**. Installation of the device in an enclosure according to **UL508** is conditional for approval or operation according to **UL508**.

Types of installation

The following SIMOTION P320-4 installation types are permitted and described in the following sections:

- Standard rail mounting (Page 7882)
- Vertical mounting (Page 7885)

Standard rail mounting

Mounting the SIMOTION P320-4 on a standard rail

Note

Use of Siemens 35 mm standard mounting rail is recommended.

Mounting the standard mounting rail clip

Before you can install the device on a standard mounting rail, you will need to attach the standard mounting rail clip included in the scope of delivery.

The fasteners and screws required are supplied with the device for mounting depending on the order number.

Requirement

- 1 standard mounting rail clip
- 2 screws
- 1 T20 screwdriver

Use the two screws to mount the standard mounting rail clip on the SIMOTION P320-4 enclosure.

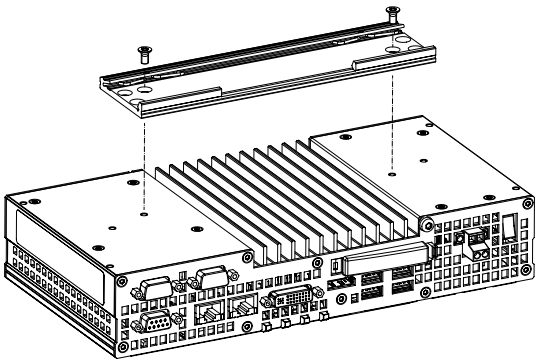


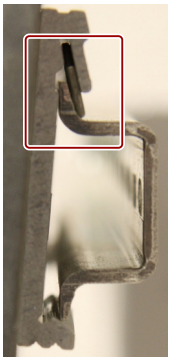
Figure 11-46 Mounting the standard mounting rail clip

Standard rail mounting

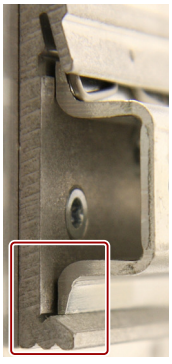
Steps for mounting on standard rail

Proceed as follows:

1. First place the SIMOTION P320-4, which is equipped with rail clips, at an angle on the upper rail guide.



2. Press the SIMOTION P320-4 slightly downward.
3. Swivel the SIMOTION P320-4 completely onto the rail until both clamps lock completely into place.
   Ensure that the locking into place of the SIMOTION P320-4 is audible.



Note

To ensure secure vertical mounting on standard rails, a standard rail ground terminal should be mounted beneath the device.

Mounting/installing the device

Examples of mounting types		
Material	**Bore diameter**	**Fixing**
Concrete	8 mm diameter 60 mm depth	Wall plug: 8 mm diameter 50 mm length Screws: 4 mm diameter 50 mm length
Plasterboard (min. 13 mm thick)	14 mm diameter	Cavity fixings: 4 mm diameter 50 mm length
Metal (min. 2 mm thick)	5 mm diameter	Metal screws M 4: 4 mm diameter 15 mm length

NOTICE

Observe the total weight of the SIMOTION P320-4

Ensure that the wall can hold four times the total weight of the device (including mounting rail and additional expansion modules).

Removing the device from the mounting rails

- Push the device down until the bottom clamp releases the device.
- Swing the device away from the rails.

Additional references

Further information on standard rail and vertical mounting can be found in the SIMOTION P320-4 E / P320-4 S Manual.

Vertical mounting

The optionally available vertical mounting front kit allows a space-saving installation.

Note

Vertical mounting front kit - SIMOTION P320-4

Only use the mounting kit that is specified for SIMOTION P320-4.

Mounting kits for previous versions cannot be used.

Fixing the vertical mounting bracket to the device

Note

Information on installation and operation is available in the supplement of the accessories.

Spare parts and accessories			
Parts for the SIMOTION P320-4	**Article number**	**Scope of delivery**	**Accessories**
Vertical mounting front kit	6ES7648-1AA20-0YP0	-	x

You can order the vertical mounting front kit via the Siemens Industry Mall (https://mall.industry.siemens.com).

Additional references

Further information on standard rail mounting and vertical mounting can be found in the SIMOTION P320-4 E / P320-4 S Manual.

SIMOTION P AutoLogin

The password for the AutoLogin must be changed when the user has changed the Windows password.

The Windows user password is changed in the normal way, e.g. "Change Password" screen. See also Section Changing the Windows user password (Page 7837).

Automatically Log On

SIMOTION P320-4 is preset so that the user does not have to enter the user name and the password to log on.

Preparation for changing the AutoLogin

In order that an AutoLogin still functions after changing the Windows password, the password for the AutoLogin must be changed.

The required procedure is shown in the following so that the change of the AutoLogin for SIMOTION P320-4 can be prepared:

1. Open the **Computer Management** window via **Control Panel** > **All Control Panel Items** > **Administrative Tools** .

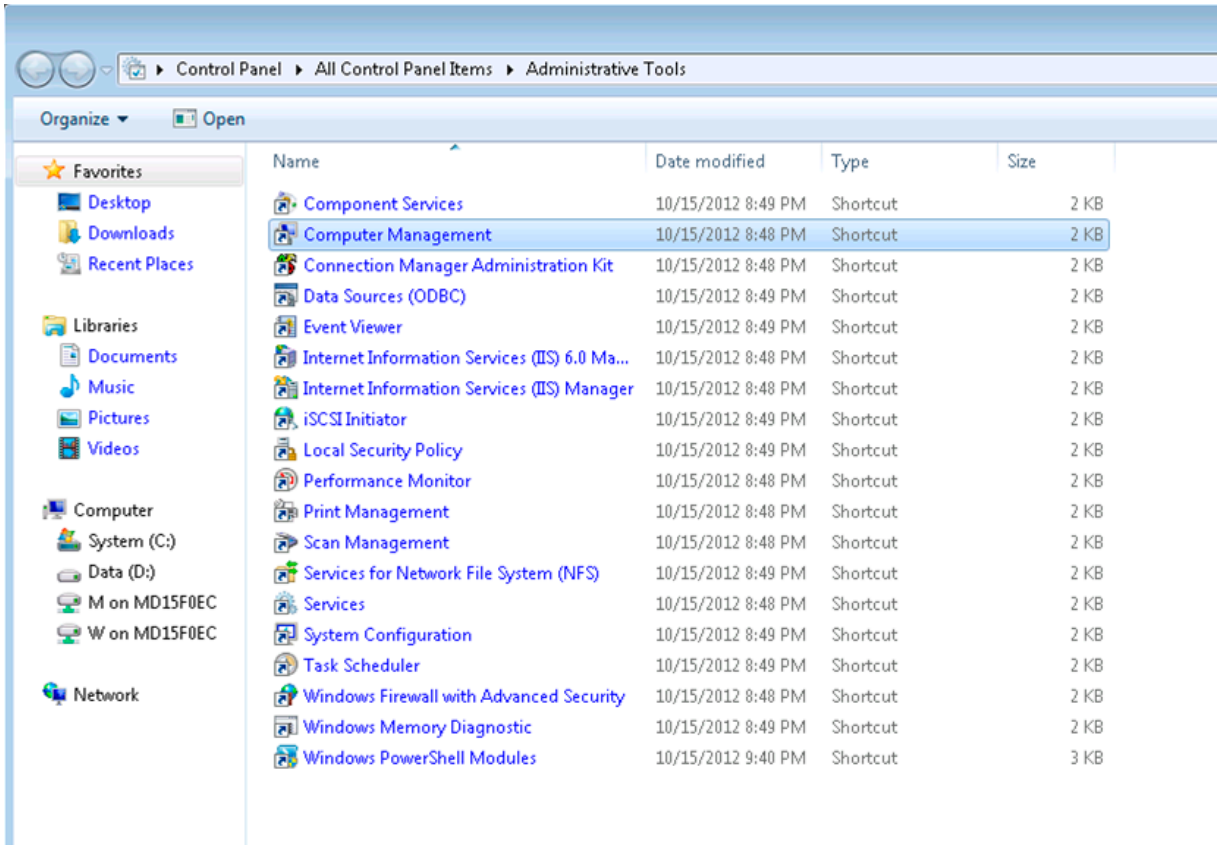


Figure 11-47 Computer Management entry point

2. The **Users** folder is displayed at **Local Users and Groups** in the **Computer Management** window.
   Click the **Users** folder. The available user groups are displayed.

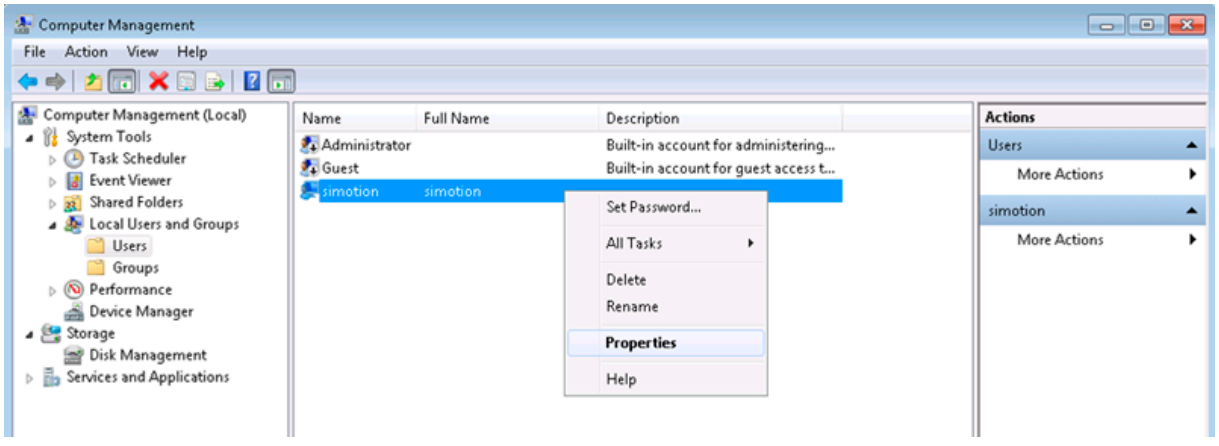


Figure 11-48 Computer Management - Users

3. With the selection of **Users**> **simotion**, you can display the properties via **Properties**.

4. Deactivate the **User cannot change password** checkbox in the **simotion Properties** dialog box.

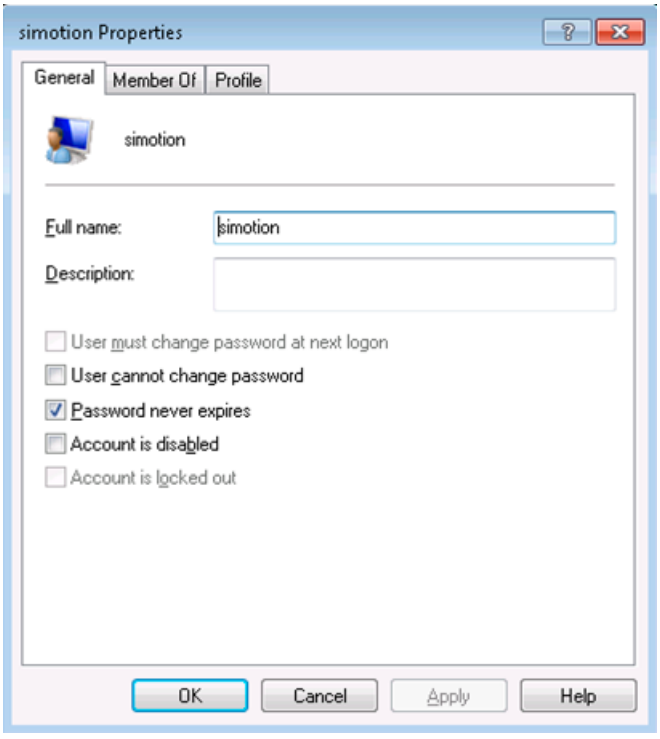


Figure 11-49 simotion Properties - General dialog box

5. Confirm with **OK.**

Changing the password for AutoLogin

To change the password for the AutoLogin, proceed as follows:

1. Open the search via **Windows-Start** and enter: **netplwiz**

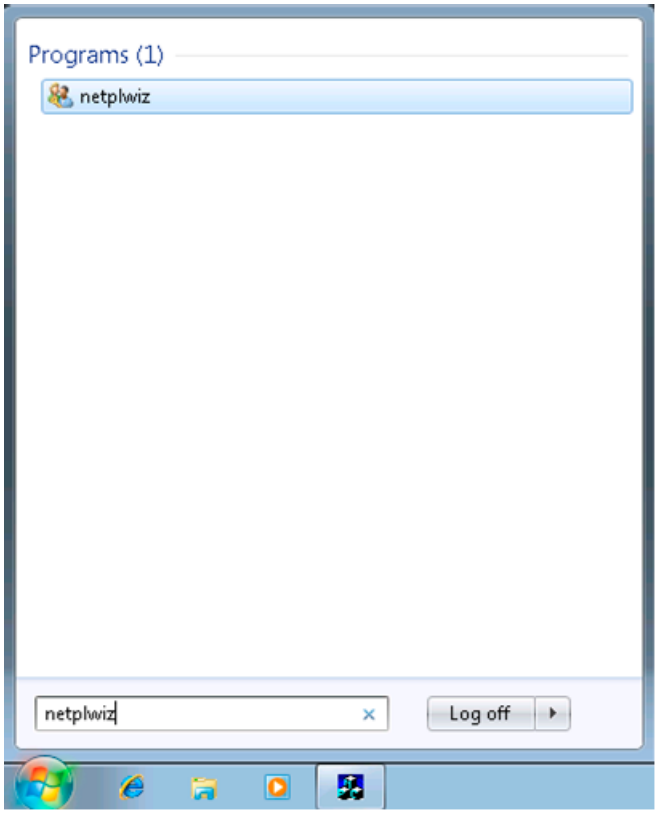


Figure 11-50 Programs netplwiz

2. The following **User Accounts** dialog box opens.

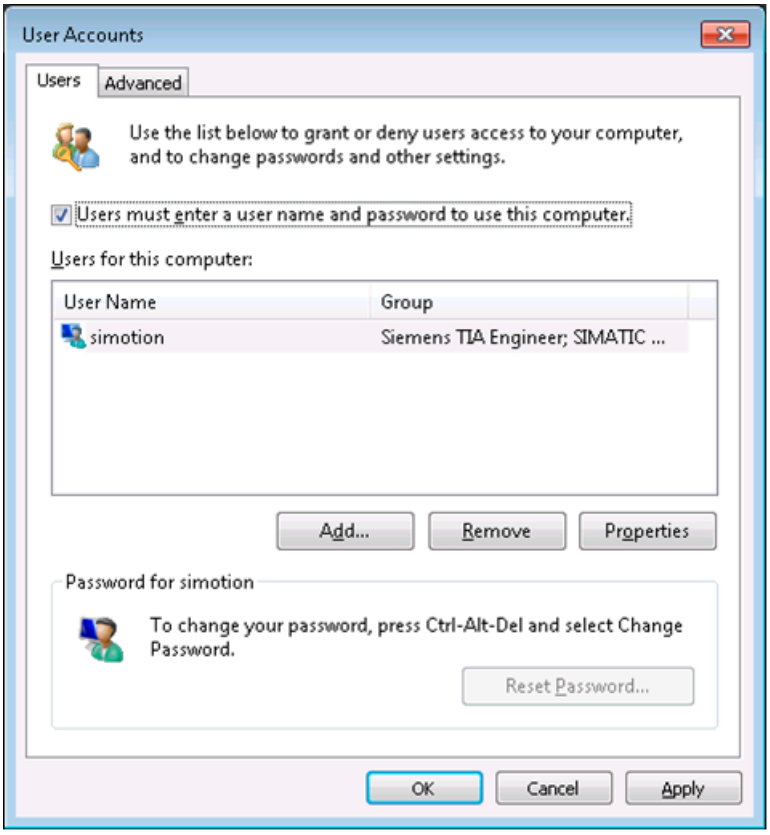


Figure 11-51 User Accounts

3. In the **Users** tab, activate the **Users must enter a user name and password to use this computer** checkbox.
   The **Apply** button is now active. Deactivate the checkbox and accept the changes now with the **OK** or **Apply** button.
   This is necessary so that the **Automatically Log On** dialog box is subsequently displayed.

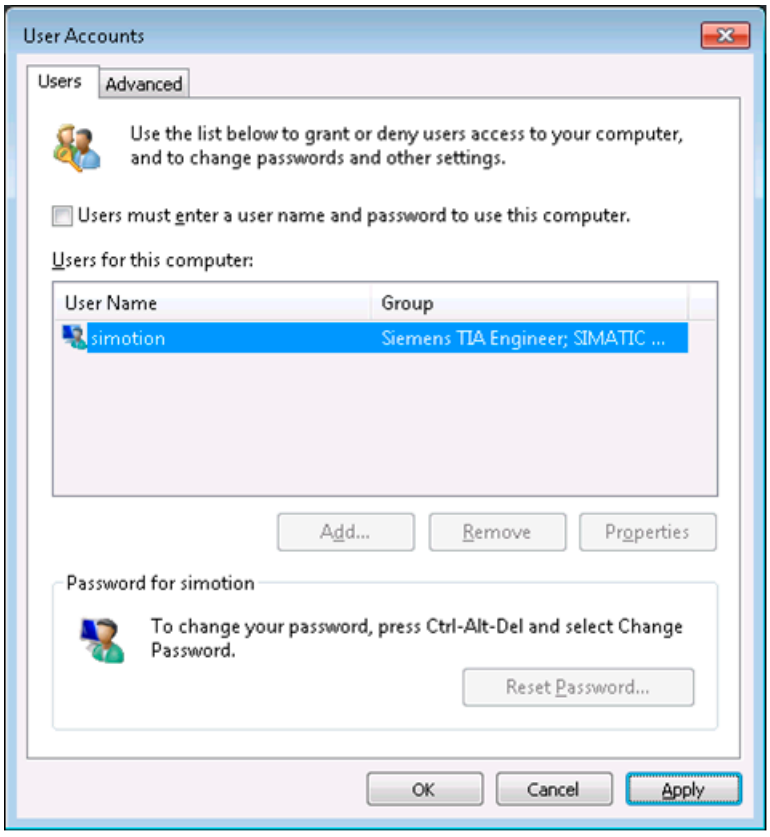


Figure 11-52 User accounts

4. Enter your changed Windows user password in the **Automatically Log On** dialog box and confirm it by entering it a second time.

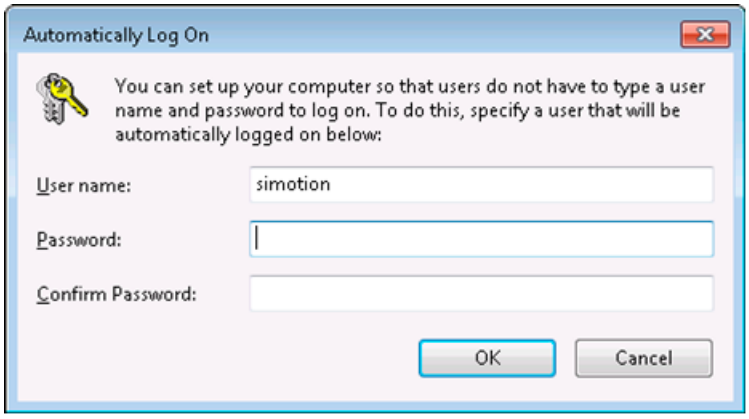


Figure 11-53 Automatically Log On dialog box

5. Confirm the password with **OK**.

11.1.1.6 Connection

Connection requirements

In this section it is described how the components of SIMOTION P320-4 are connected and the elements for communication prepared.

During the installation of SIMOTION modules, you must pay attention to the electrical configuration design.

You obtain information on wiring and networking a complete SIMOTION system.

Requirement

The SIMOTION P320-4 is already installed in a cabinet or in a controller housing.

Open equipment

These modules are open equipment. Therefore, only install modules in housings, cabinets, or in electrical equipment rooms that can only be entered or accessed with a key or tool. Only instructed or authorized personnel have access.

Observe the following points during installation:

- The device may only be connected to 24 VDC power supplies that meet the requirements in accordance with SELV. The cable cross-section must be chosen large enough to ensure that no damage can result from cable overheating in the event of a short-circuit in the SIMOTION P320-4.
- Protect the device against dust, moisture, heat and severe vibration.
- Do not subject the SIMOTION P320-4 to direct sunlight.
- Install the device in such a way that it cannot present a hazard (e.g. by toppling).
- Ensure that the clearance to other components or the sides of cabinets is at least 50 mm above and 100 mm below the device.

Overview of connections

Below the SIMOTION P320-4 interfaces to which the I/O devices can be connected are shown.

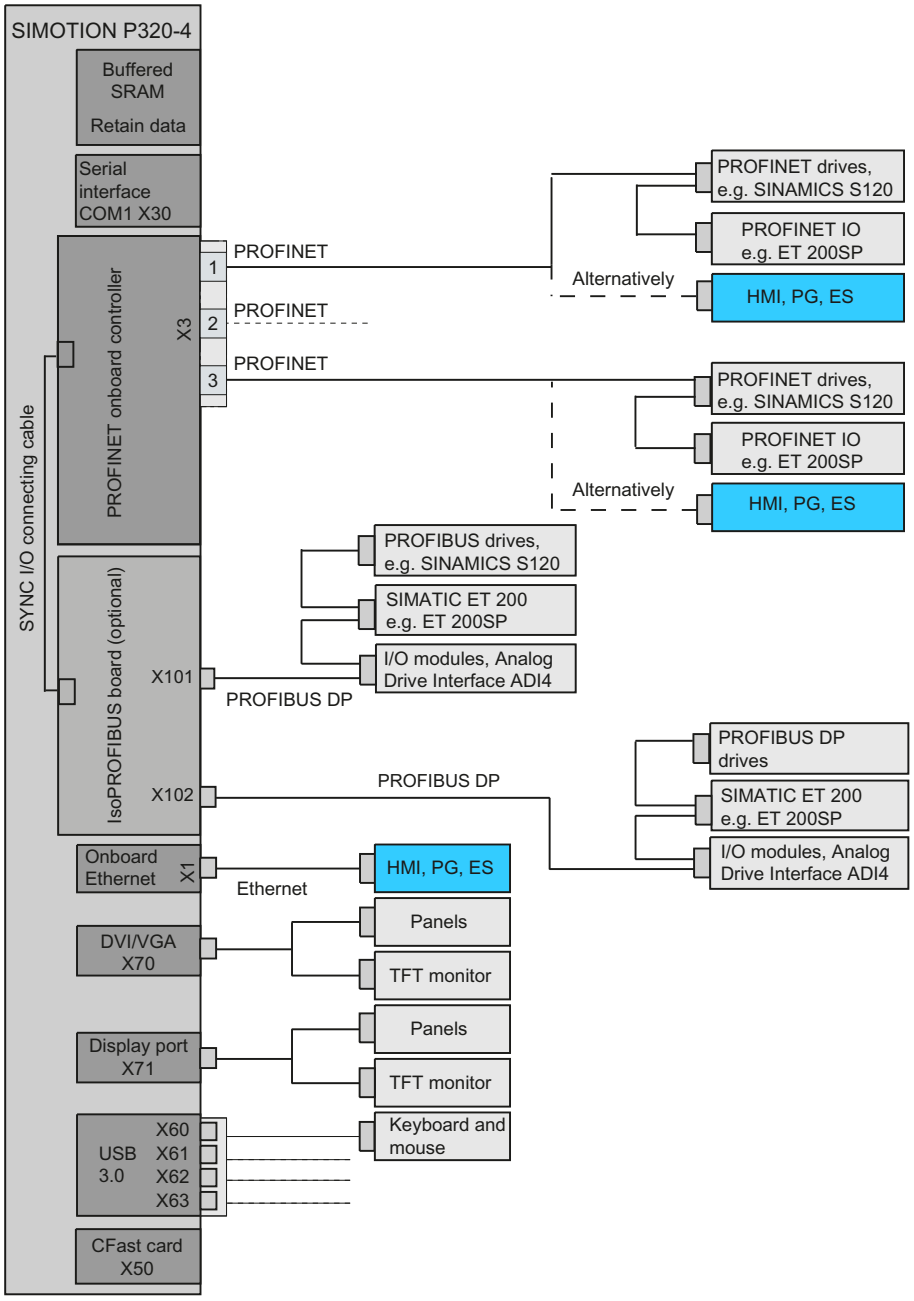


Figure 11-54 System overview SIMOTION P320-4 (hardware)

The cabling for all elements and components of the complete system must only be carried out when disconnected from the mains.
It does not matter in which order connection takes place.

The connection of the individual elements is described in the following sections:

- Connecting PROFINET (Page 7893)
- Inserting IsoPROFIBUS (Page 7896)
- Connecting Ethernet (Page 7904)
- Connecting keyboard/mouse (Page 7905)
- Connecting the power supply (Page 7905)

Connecting PROFINET

PROFINET onboard

The system is connected to the Real-time Ethernet (PROFINET IO with RT and IRT) using the permanently integrated PROFINET interface. The device has a PROFINET interface with 3 ports.

The PROFINET IO with IRT allows IT services to be performed in parallel to the real-time communication on an Ethernet cable, where the PROFINET IO with IRT has a reserved area that provides a decoupling from the standard communication.

Note

PROFINET onboard is not a plug-in board. Instead, it is integrated in the SIMOTION P320-4.

Connection

Connection example

The figure shows a connection example of a PROFINET application with SIMOTION P320-4, SINAMICS S120 (CU320-2) and ET 200 PN.

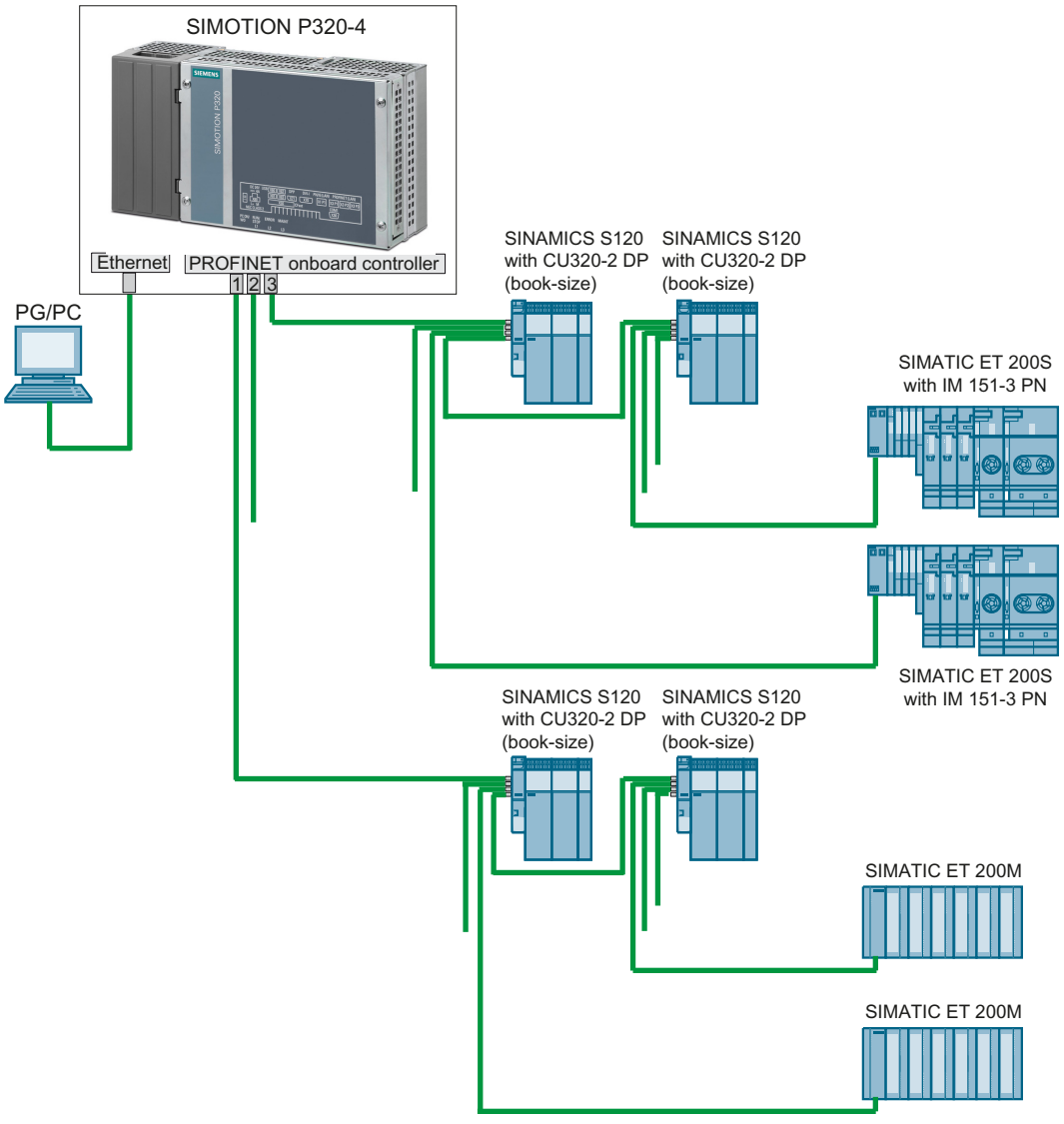


Figure 11-55 Example for a PROFINET application

Note

If the integrated ports of SIMOTION P320-4 and SINAMICS are not sufficient, an external SCALANCE switch can be used.
If PROFINET IO with IRT communication is to be enabled downstream of the switch, then a suitable switch (which supports IRT) must be used.

Cable

4-wire, shielded installation cables are used as PROFINET cables for IE FC RJ45.

Connector

RJ45 PN connectors with straight or angled exit can be used. Standard RJ45 connectors cannot be used.

Strain relief

When inserted, a PROFINET cable will cause unwanted forces to be exerted on the sockets of the PROFINET onboard (strain, lateral pressure). In order to prevent the sockets being subjected to these forces, you must install the strain-relief bracket and attach the cable tie supplied in the accessory kit.

For further information see Mounting a strain relief (Page 7895).

Additional references

You can find further information on the website www.siemens.com/simotion (www.siemens.com/simotion):

- Drives and I/O modules that are released for SIMOTION
- You will also find current brochures and catalogs

Mounting a strain relief

A strain relief for SIMOTION P320-4 is included in the scope of delivery.

The strain relief is designed to prevent the lines connected to the device from being accidentally pulled out.

Note

One cable tie is required for each interface. The cable ties are not included in the scope of delivery and must be ordered separately.

Requirement

- 1 strain relief
- 2 screws
- 1 T10 screwdriver

Procedure

Proceed as follows to secure the strain relief:

1. Remove the marked countersunk screws of the SIMOTION P320-4.

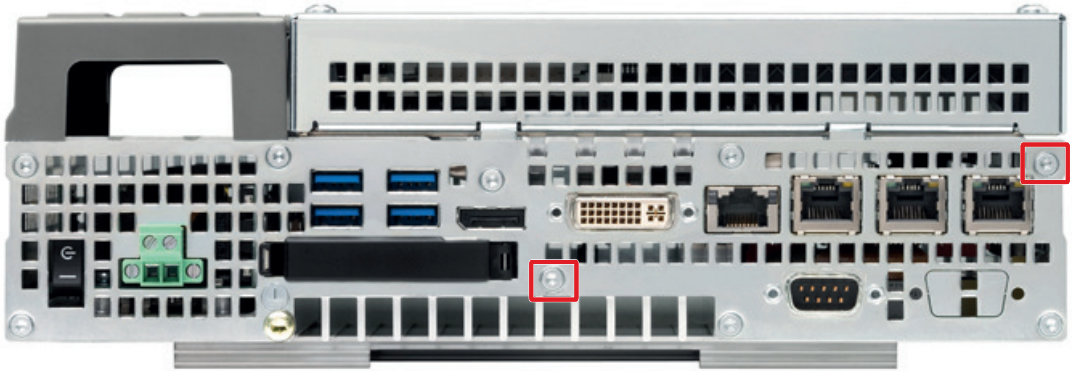


Figure 11-56 Strain relief - marking of the screws

2. Secure the strain relief at the marked points.
   Only use the enclosed lens head screws for securing the strain relief.

Connecting PROFIBUS

Requirement

- SIMOTION P320-4 is switched off and disconnected from the mains
- IsoPROFIBUS board (Article No. 6AU1390-0AA00-0AA1)

SYNC I/O connecting cable / Plugging in the IsoPROFIBUS board

To be able to plug in the IsoPROFIBUS board, the device must be opened and the SYNC I/O connecting cable must be rerouted:

1. Remove the screws of the upper cover plate and open the SIMOTION P320-4.
   At the factory, the SYNC I/O connecting cable is secured in the device using an adhesive tape.

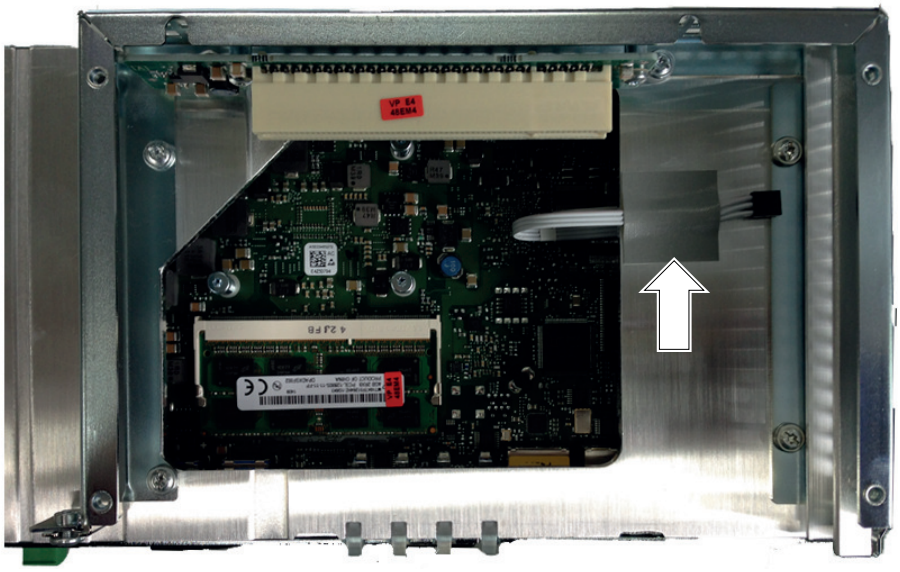


Figure 11-57 Opened SIMOTION P320-4 with SYNC I/O connecting cable as delivered ex works

2. Remove the adhesive tape and change the position of the SYNC I/O connecting cable as follows:
   the SYNC I/O connecting cable is routed to the interface side of the device and secured there again.

Figure 11-58 SIMOTION P320-4 with changed position of the SYNC I/O connecting cable

3. Remove the slot plate on the left side to expose the cutout for the interfaces of the IsoPROFIBUS board.
4. The IsoPROFIBUS board is inserted in the next step:
   Slide the IsoPROFIBUS board on the left into the guide rail of the previously removed slot plate and push it backward guided by the rail.

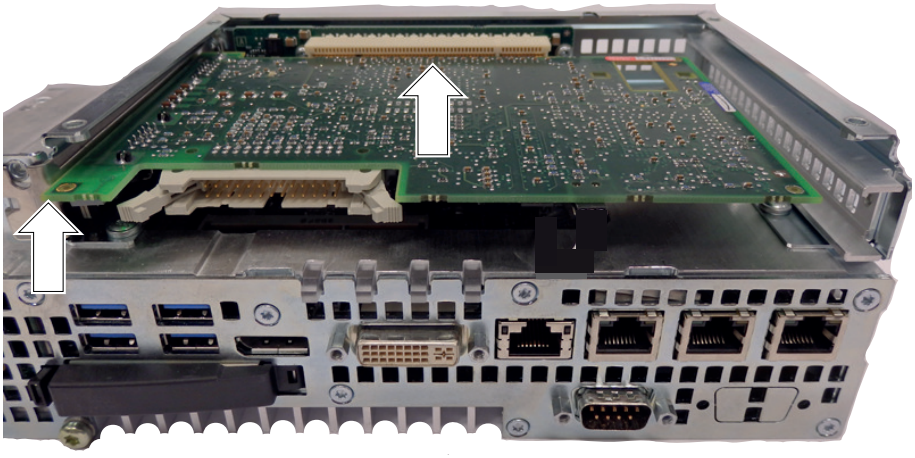


Figure 11-59 Inserting the IsoPROFIBUS board

5. The IsoPROFIBUS board is inserted into the rear PCI slot by applying slight pressure.
6. Then the SYNC I/O connecting cable is connected to the IsoPROFIBUS board.
   The cable is plugged directly into the 3-pole socket on the IsoPROFIBUS board for this purpose.

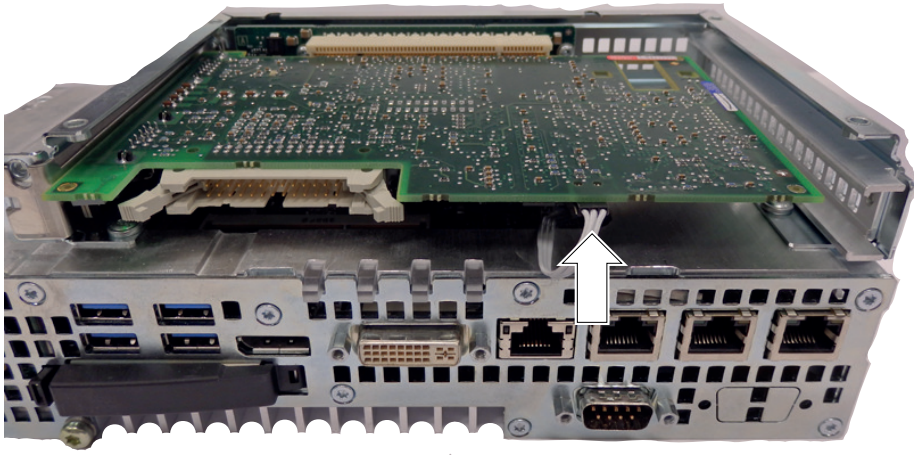


Figure 11-60 Plugging in the SYNC I/O connecting cable

7. Close the cover plate of the SIMOTION P320-4 and fasten the screws.

Connecting the IsoPROFIBUS board

The interface for the optional IsoPROFIBUS board is protected by a cover. There are two holes in the cover for wiring.

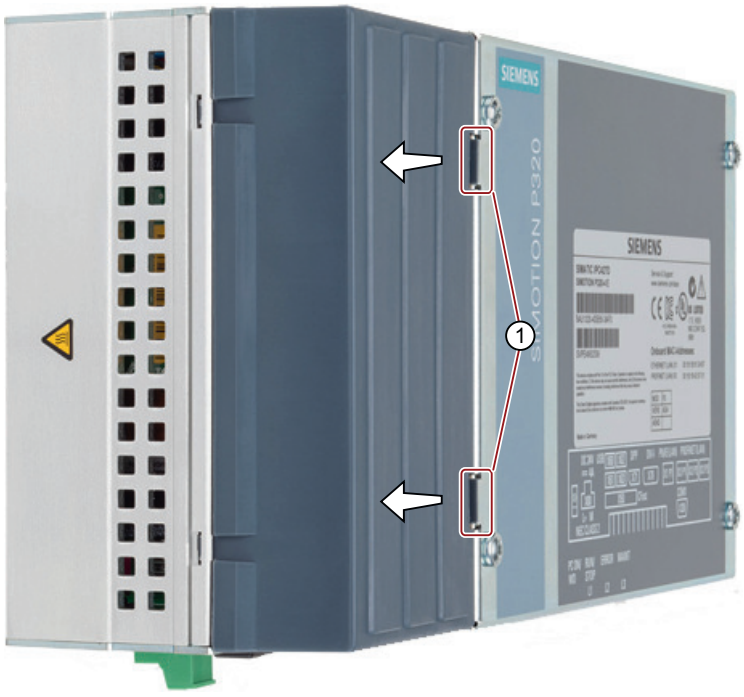
Requirement

The device is switched off.

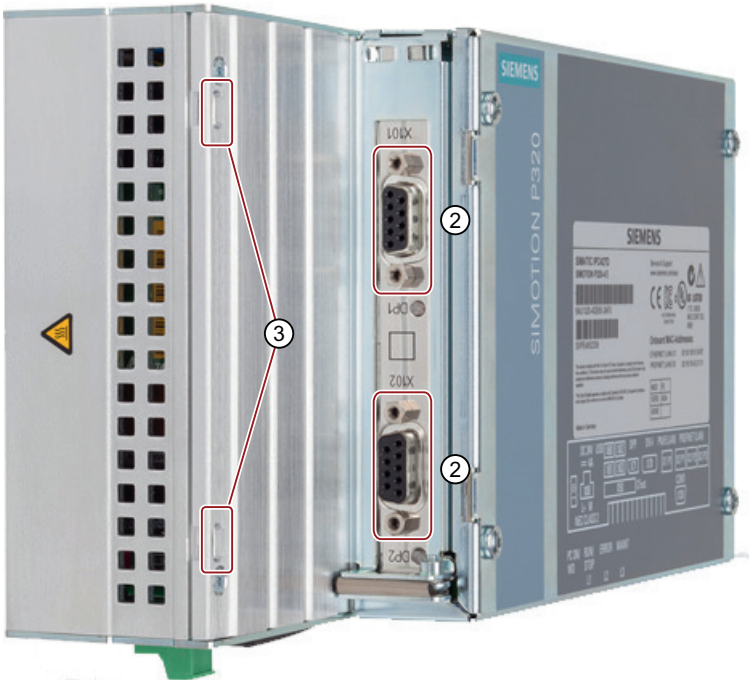
Procedure

To connect the IsoPROFIBUS board, proceed as follows:

1. Unlock the cover by pushing the latches in the direction of the arrow;
   swivel the released cover in the direction of the arrow and remove it. The interfaces for the optional IsoPROFIBUS board are then free.



2. Connect the IsoPROFIBUS board to the slots shown.



3. Place the cover on the positions shown and close it.
   Make sure that the latches engage to secure the cover in its position.

IsoPROFIBUS board and PROFIBUS DP

PROFIBUS DP connection example

Both interfaces of the IsoPROFIBUS board can be used as PROFIBUS DP interfaces. The following screen shows an example of a PROFIBUS application.

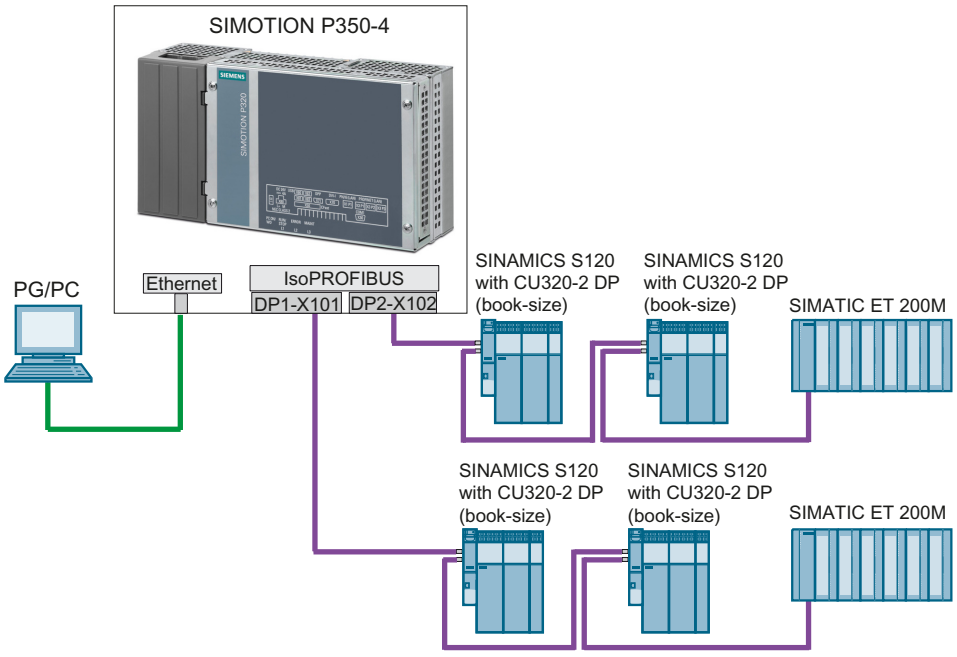


Figure 11-61 Example for a PROFIBUS application

Connecting the bus connectors for the X101 and X102 interfaces (IsoPROFIBUS)

The bus connector is used to connect the PROFIBUS cable to the IsoPROFIBUS interface (X101, X102). thus establishing a connection to additional stations.

Connecting the bus connector

Proceed as follows to connect the bus connector:

1. Plug the bus connector into the IsoPROFIBUS interface on the module.
2. Screw the bus connector onto the interface.
3. If the bus connector is located at the start or end of a segment, you must connect the terminating resistor (switch position "ON").

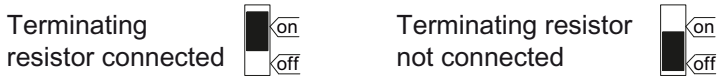


Figure 11-62 Bus connector: Terminating resistor switched on and off

⚠ WARNING

Data exchange on the bus can be interrupted!

A bus segment must always be terminated with the terminating resistor at both ends. This is not the case, for example, if the last node with a bus connector is de-energized. Because the bus connector takes its voltage from the station, this terminating resistor is ineffective.

Make sure that the stations at which the terminating resistor is connected are always energized.

PROFIBUS cable

Properties of the PROFIBUS cable

The PROFIBUS cable is a two-stranded, twisted, and shielded cable.

Table 11-4 Properties of the PROFIBUS cable

Features	Values
Surge impedance	Approximately 135 to 160 Ω (f = 3 to 20 MHz)
Loop resistance	≤ 115 Ω/km
Operating capacity	30 nF/km
Attenuation	0.9 dB/100 m (f = 200 kHz)
Permitted strand cross-section	0.3 mm^2 to 0.5 mm^2
Permissible cable diameter	8 mm ± 0.5 mm

Rules for laying

When you install a PROFIBUS cable, you must not:

- Twist
- Stretch
- Or compress it.

When installing the indoor bus cable, you should also remember the following boundary conditions (d_A = outer diameter of the cable):

Table 11-5 Boundary conditions for indoor routing of bus cable

Features	General conditions
Bending radius for a single bend	≥ 80 mm (10 × d_A)
Bend radius for multiple bends	≥160 mm (20 × d_A)
Permissible temperature range for installation	- 5° C to + 50° C
Temperature range for storage and stationary operation	- 30° C to + 65° C

Note

The length codes for the pre-assembled cables can be found on the website www.siemens.com/simotion (www.siemens.com/simotion).

Connecting Ethernet

Procedure

Plug the LAN cable into the Ethernet interface of the SIMOTION P320-4. You must hear the connection engage properly.

Cables

Standard shielded twisted pair cables are used for the networking in this case.

The following connection cables are recommended:

- SIMATIC NET, ind. Ethernet TP XP CORD RJ45/RJ45, TP CORD assembled with 2 RJ45 connectors, send and receive cables crossed (cross-over)
  Order no.: 6XV1850-2H*)
- SIMATIC NET, ind. Ethernet TP CORD RJ45/RJ45, TP CORD assembled with 2 RJ45 connectors
  Order no.: 6XV1850-2G*)

*) Length code		Dimensions
	E50	0.5 m
	H10	1 m
	H20	2 m
	H60	6 m
	N10	10 m

Note

Further information on the use of these cables can be found in the SIMATIC NET Twisted-Pair and Fiber-Optic Networks Manual (https://support.industry.siemens.com/cs/ww/en/view/8763736).

Connecting the keyboard and mouse

USB interfaces

You can connect the keyboard and the mouse to the USB interfaces of the SIMOTION P320-4.

Connecting the power supply

Safety rules

Basic rules

Only the basic rules for electrical installation can be described in this section on account of the diverse applications possible. At a minimum, you must comply with these basic rules to ensure problem-free operation.

Rules for safe operation

In order to ensure safe operation of your equipment, implement the following measures, adapting them to suit your conditions:

- An EMERGENCY OFF concept in accordance with the generally accepted rules of current engineering practice (e.g. European standards EN 60204, EN 418, and similar).
- Additional measures for limiting the end position of axes (e.g. hardware limit switches).
- Equipment and measures for protection of motors and power electronics in accordance with the SIMOTION installation guidelines.
  In order to identify hazards, we also recommend that a risk analysis be conducted on the entire system in accordance with the basic safety requirements set out in Appendix 1 of EU machinery directive.

Additional references

- Guidelines on Handling Electrostatic Sensitive Devices (ESD), see Appendix ESD Guidelines (Page 7990).
- For installing a system with SIMATIC ET 200 I/O (e.g. ET 200S, ET 200M, etc.), refer to the documentation for the relevant ET 200 I/O systems.

Standards and Regulations

VDE guideline compliance

During wiring, you must observe the appropriate VDE guidelines, in particular VDE 0100 and VDE 0113 for tripping devices and short-circuit and overload protection.

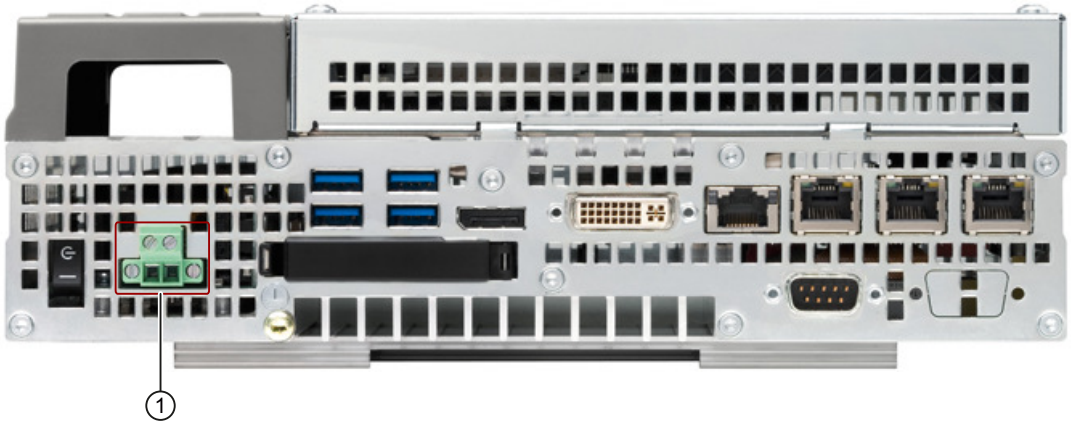
System start-up after specific events

The following list identifies considerations required for startup of a system following certain events.

- If the system starts up again following a voltage drop or power failure, all hazardous operating states must be prevented from occurring. If necessary, force an EMERGENCY OFF.
- If the system starts up again after the EMERGENCY OFF apparatus is released, the startup must not be unchecked or undefined.

Connecting the power supply

Connect your power supply (e.g. SITOP smart 24 V / 10 A) directly to the power-supply socket of the SIMOTION P320-4.



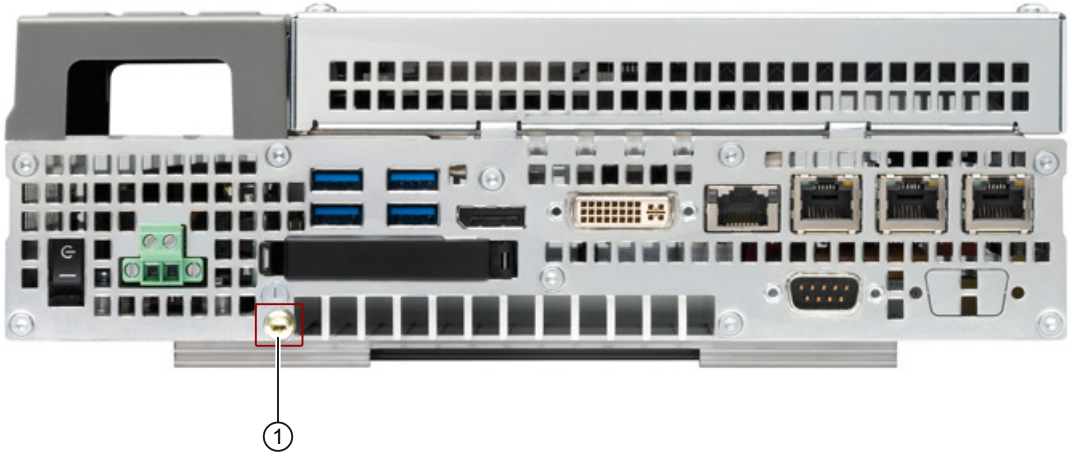
① 24 VDC power supply

Figure 11-63 Power supply connection

⚠ WARNING
On/Off switch
The On/Off switch does not disconnect the device from the power supply system.
To remove power from the device, the power supply plug must be removed.

Protective conductor connection and potential equalization

Connect the equipotential bonding strip on the SIMOTION P320-4 (large surface, large-area contact) with the central grounding point of the cabinet or system in which the devices are installed.
The protective conductor must not have less than a minimum core cross-section of 2.5 mm^2.



① Protective conductor connection / potential equalization

11.1.1.7 Power on and software installation

Power-up

SIMOTION P320-4 is supplied with pre-installed software modules. After activation, Windows with SIMOTION P is powered up automatically.

The SIMOTION P320-4 issues the following message after successful power-up:

Figure 11-64 SIMOTION P320-4 after power-up

The SIMOTION P320-4 responds in this way to every subsequent power-up.

Additional software for HMI

Note

Action before installing additional software

Before installing additional software, make sure that you terminate the runtime of the SIMOTION P.

You terminate SIMOTION P in the application SIMOTION P State using the button **Terminate SIMOTION P**.

If required, the following software can be installed directly on the SIMOTION P320-4:

- WinCC flexible Runtime
- WinCC flexible engineering system for P320-4 S
- OPC client from other manufacturers
- Utilities of the OEMs
- SIMATIC NET

This software must be purchased separately and is not contained in the SIMOTION P320-4 scope of delivery.

Please refer to the documentation supplied with the software for details regarding the procedure for installing the respective software.

Customer-specific software

To ensure high-quality and reliable working of the device, SIMOTION P320-4 is fully configured and ready for operation at the time of delivery.

For this purpose, the system components used are subject to a certification procedure with Siemens as the system manufacturer. The certification process establishes and documents the real-time features of the complete system.

NOTICE
Changes or expansions of PC components (hardware or software) by a third party If PC components (hardware or software) are modified or expanded by a third party, compliance with product features cannot be guaranteed. The OEM or user involved must assume sole responsibility for such components. Compliance with product features can only be guaranteed by SIEMENS.

11.1.1.8 Operator Control (hardware)

SIMOTION P state application

SIMOTION P State application overview

The LEDs, mode selector, and CF card handling indicators, which are implemented as hardware on other SIMOTION platforms, are visualized on the screen for SIMOTION P320-4.

The display is performed by the **SIMOTION P State** application. The following figures show SIMOTION P State in the started state with/without inserted IsoPROFIBUS board.

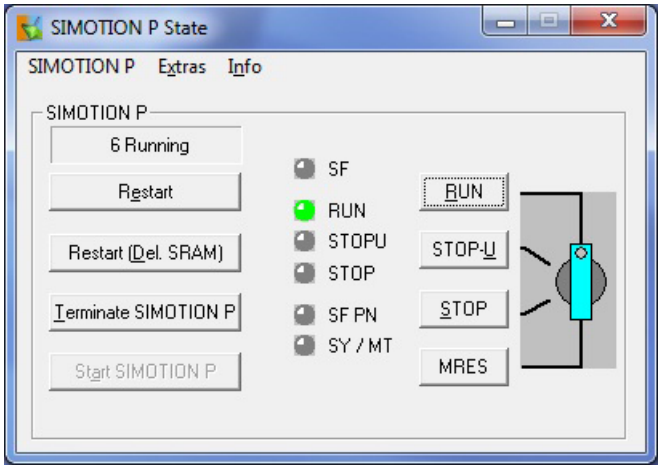


Figure 11-65 SIMOTION P State - PROFINET onboard

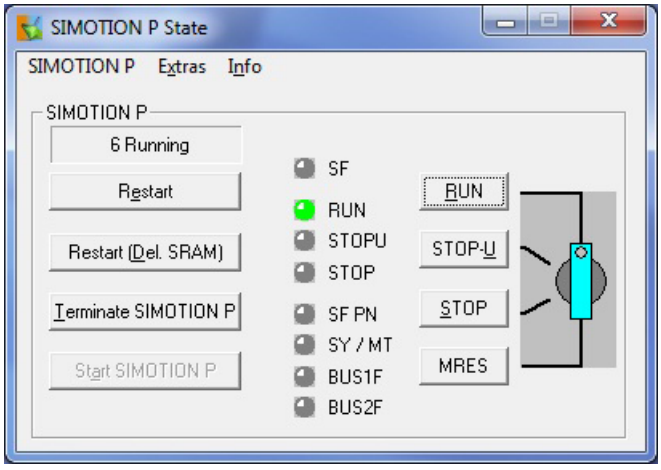


Figure 11-66 SIMOTION P State - PROFINET onboard and additionally with optional IsoPROFIBUS board

Note

The SIMOTION P State application is in English.

Generally, only English is supported on the SIMOTION P320-4, this also applies for the Win7 operating system.

LED display

Meaning of LEDs

The meaning of the LEDs and their colors is described in the table below.

LED display		Meaning	Description
SF	Red	Fault	The fault state of the SIMOTION P320-4 is displayed.
RUN	Green	SIMOTION P320-4 in RUN mode	The user program is running.
STOPU	Yellow	SIMOTION P320-4 in the STOP User Program	The technology packages (e.g. synchronous operation, cam) are active. Test and commissioning functions can be executed without a user program. The user program is not active.
STOP	Yellow	SIMOTION P320-4 in STOP mode	There is no user program running. The technology packages are not active.
SF PN	Red	PROFINET bus fault	A fault state is displayed on the PROFINET interface (X3).
SY/MT	Yellow	Sync of the PROFINET interface	It is displayed whether the PROFINET interface has been synchronized.
BUS1F	Red	Group fault	A fault state is displayed on the IsoPROFIBUS interface (X101).
BUS2F	Red	Group fault	A fault state is displayed on the IsoPROFIBUS interface (X102).

Note

LEDs for IsoPROFIBUS board

The LEDs BUS1F and BUS2F are **only** shown if an IsoPROFIBUS board is connected.

Additional references/information

You can find additional information at:

- Diagnostics via LED displays (Page 7975) (for a more accurate diagnosis of an LED state)
- Operating displays (SIMOTION P320-4 E / P320-4 S Manual: Information on display modes, of the SF PROFINET LEDs only displayed on the hardware side, in the operating displays section)
- PROFINET bus error occurs (Page 7981)

Mode selector

Mode selector in SIMOTION P State

Use the mode selector to set the SIMOTION P320-4 to different states. The controller can be switched in **RUN** or **STOP** / **STOPU** mode and perform an overall reset.

The change of the operating state with the mode selector has **priority** over a state selected via SIMOTION SCOUT.

Note

Setting the operating state via the mode selector

The **STOP/STOPU** switch position cannot be set to **RUN** via the SIMOTION SCOUT.

To change the operating state, use the mode selector.

Further information

Further information can be found in Section Overall reset with the mode selector (on the screen) (Page 7959).

Current operating status

The current state of the SIMOTION P320-4 is displayed in the upper left area of the **SIMOTION P State** application.

Table 11-6 The following state displays are possible:

Not started	SIMOTION P320-4 is not started.
0 ... 5	The progress of the SIMOTION P320-4 power-up is displayed.

6 Running	SIMOTION P320-4 has completed power up and the cyclic tasks are activated.
Disabled	SIMOTION P320-4 is deactivated.

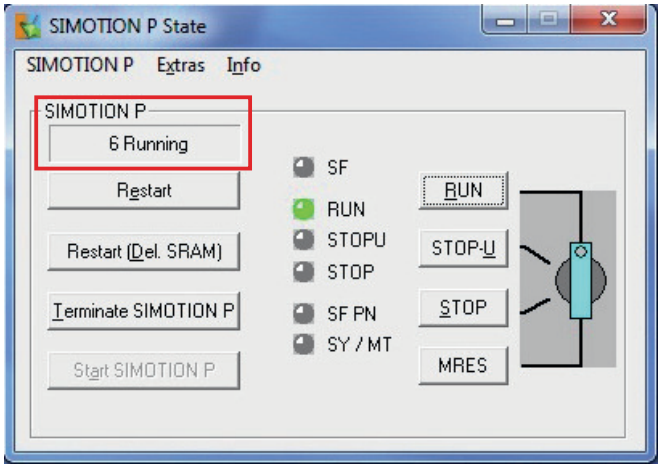


Figure 11-67 SIMOTION P State - state display

The status LEDs and the current operating state are also displayed in the Windows task bar:

RUN state: Green

STOP state: Yellow

NOT STARTED state: Gray

Note

FAQs can be found in Section Troubleshooting/FAQs (Page 7983).

Function elements

Functions of the buttons

Below the current SIMOTION P320-4 status display, you will find a selection of buttons which can only be operated with administrator rights.

Description of the button functions:

Table 11-7 Function of the buttons in SIMOTION P State

Button	Function
Restart	Restarting SIMOTION P320-4. The information in the SRAM is retained. If RamToRom is called beforehand, the project backup is restored. **Note** This is only possible if SIMOTION P320-4 has been powered up (status not "0" to "5", "not started").
Restart (Del. SRAM)	Restarts SIMOTION P320-4 and deletes the information in the SRAM. If RamToRom is called beforehand, the project backup is restored. **Note** This is only possible if SIMOTION P320-4 has been powered up (status not "0" to "5", "not started").
Terminate SIMOTION P	Stop SIMOTION P320-4. **Note** This is possible only if SIMOTION P320-4 has powered up.
Start SIMOTION P	Start SIMOTION P320-4. **Note** This is possible only if SIMOTION P320-4 has not been started

Resetting to the delivery condition

SIMOTION P320-4 can be reset to the default settings at the time of delivery (Page 7966) via SIMOTION P State.

SIMOTION P State - menus

The menus of the SIMOTION P State application are described below:

- SIMOTION P
- Options
- Info

SIMOTION P menu

SIMOTION P	Description
Temperature	Display of the temperatures in the device, monitoring of the threshold value. An alarm (Page 7979) is issued when a threshold value is reached or exceeded.
Set Diagnostic Switch	Diagnostic switch for recording diagnostic data and backing up the non-volatile SIMOTION data on the external CFast card. Diagnostics data can be recorded during power-up and during operation for the error diagnostics. The diagnostics switch is automatically reset after writing the data. **See also the following sections**: Recording diagnostic data (Page 7964). Backing up non-volatile SIMOTION data (retain data) (Page 7951)
Webserver	Access to the SIMOTION IT Web server is protected by a multi-level security concept. Upon delivery, the security level is low to allow configuration of the device. The security state of the Web server can be set here. The following selection options are available: • Security Level: normal • Security Level: low

Temperature specifications

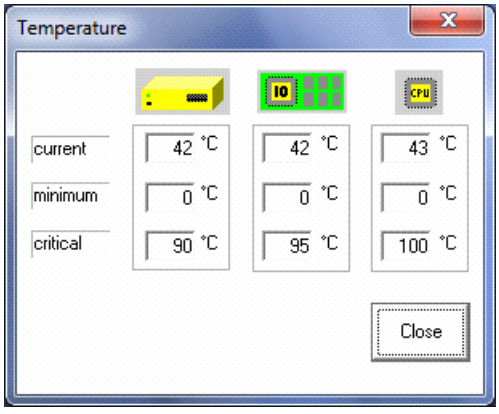


Figure 11-68 Dialog Temperature

The temperature specifications for the SIMOTION P320-4 can be found at **SIMOTION P > Temperature**.

The current, the minimum and the critical temperature are determined at the housing, mainboard and CPU measuring points and displayed in this dialog box.

Options menu

Options	Description
Reset to factory default	Restore the factory default. This will delete the following data: • The non-volatile SIMOTION data (retain data) in the SIMOTION device • The backup of the non-volatile SIMOTION data on the external CFast card (PMEMORY.XML/PMEMORY.BAK) • The user data from the "Volatile SIMOTION data" area (RAM) and on the external CFast card (ROM) • The communication configuration (IP and DP parameters) on the external CFast card is set to the factory settings. The licenses on the CFast card are retained.

Note

Station Configurator settings

Settings made with the Station Configurator are reset to the default values at the time of delivery as well.

Further information

can be found in Section Defined states > Restore the factory settings using SIMOTION P State (Page 7955)

Info menu

Info	Description
Hardware Version	Display of the hardware information.
Software Version	Display of the software versions.
Info about SIMOTION P State	Display of the SIMOTION P version.

Note

The versions in the **Info** menu only become visible after the Windows power-up after the SIMOTION P320-4 has been started once.

11.1.1.9 Parameter assignment/addressing

Parameter assignment / addressing requirements

The following actions should be completed before you start parameterizing or addressing:

- The SIMOTION P320-4 and the other hardware are installed.
- The devices are connected.

- The required software is installed.
- The SIMOTION P320-4 has been switched on.

The actual state should be saved after the parameterization.

Factory settings

Factory settings overview

Requirement

You have obtained information about the various communications models. Further information can be found under HMI and SIMOTION SCOUT (Page 7858).

You are familiar with the Windows Control Panel.

Resetting the factory settings

It is possible to reset SIMOTION P320-4 to the factory settings (see section Resetting the SIMOTION P to the initial state (Page 7966)).

Ethernet - TCP/IP local

SIMOTION P320-4 delivery state

Local communication in the SIMOTION P320-4 between Windows programs such as HMI and the SIMOTION P Runtime software is handled via the preset **TCP/IP** Ethernet communication.

As an alternative to TCP/IP, the **PC internal (local)** communication can also be used. Further information on how to check the settings in the Windows Control Panel can be found in Section Check PC internal settings (Page 7926).

SIMOTION P control manager

Requirement

NOTICE
SIMOTION P Control Manager
Because the settings made via the SIMOTION P Control Manager change the factory default parameters, they should only be made by authorized persons.

SIMOTION P Control Manager overview

The SIMOTION P Control Manager is used to configure the SIMOTION P software.

Call

The SIMOTION P Control Manager is only used by the **administrator**. Only the administrator is authorized to make changes.

The **SIMOTION P Control Manager application** is called by means of the operating system's Control Panel.

Properties

SIMOTION P Control Manager has the following features:

- The following settings take effect immediately after a change:
  - Start SIMOTION P
  - Automatic Restart By Plug-In External Memory Card
  - Ethernet-IP address
- The values to be set are generally required only by the administrator.
  These are:
  - License handling
  - IP addresses

Calling SIMOTION P Control Manager from the control panel

You can call the SIMOTION P Control Manager in the Windows Control Panel (Start > Control Panel).

The call can only be made as administrator. If you are not logged on as administrator, a prompt will appear notifying you of this fact.

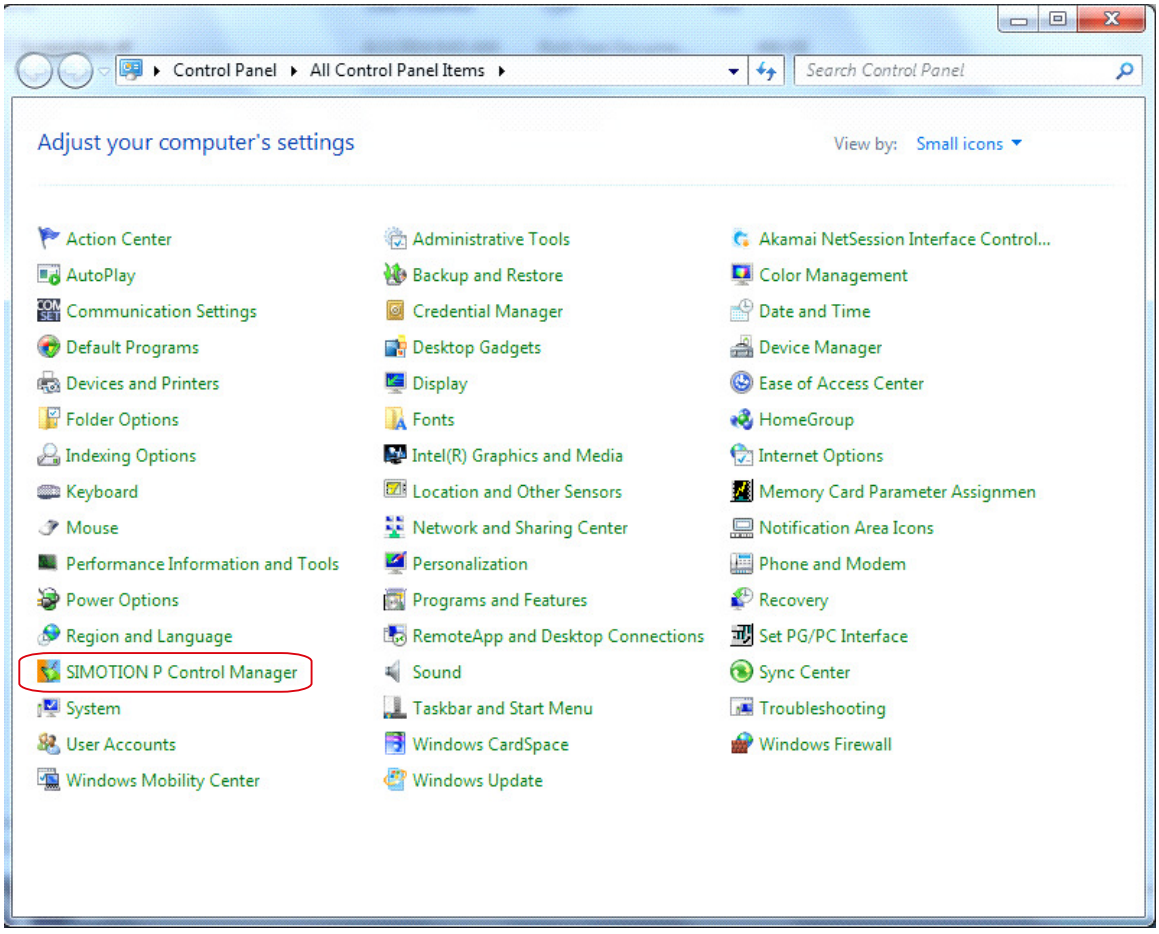


Figure 11-69 Calling the SIMOTION P Control Manager

If the call was successful, the **SIMOTION P Control Manager** dialog, **Administration** tab, is displayed.

The SIMOTION P Control Manager application is in English.

Administration tab

The **Administration** tab contains:

- Start SIMOTION P (Page 7920)
- Automatic Restart By Plug-In External Memory Card (Page 7921)
- Runtime Network Connections (Page 7922)

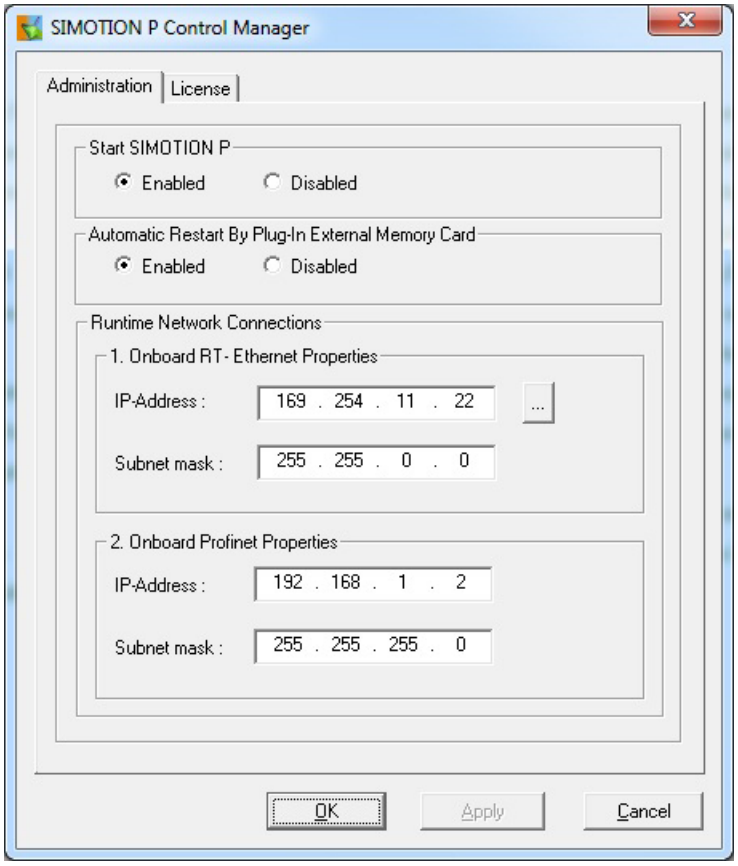


Figure 11-70 SIMOTION P Control Manager - Administration tab

Start SIMOTION P

The following settings are possible:

- **Enabled**
  Immediately cancels the lock set with **Disabled**. The SIMOTION P320-4 must be booted to automatically start SIMOTION P.
  SIMOTION P can also be started by means of the SIMOTION P State application using the **Start SIMOTION P** button.
- **Disabled**
  Prevents the automatic starting of SIMOTION P while the Windows system is booting up. SIMOTION P cannot be started until this lock is canceled.

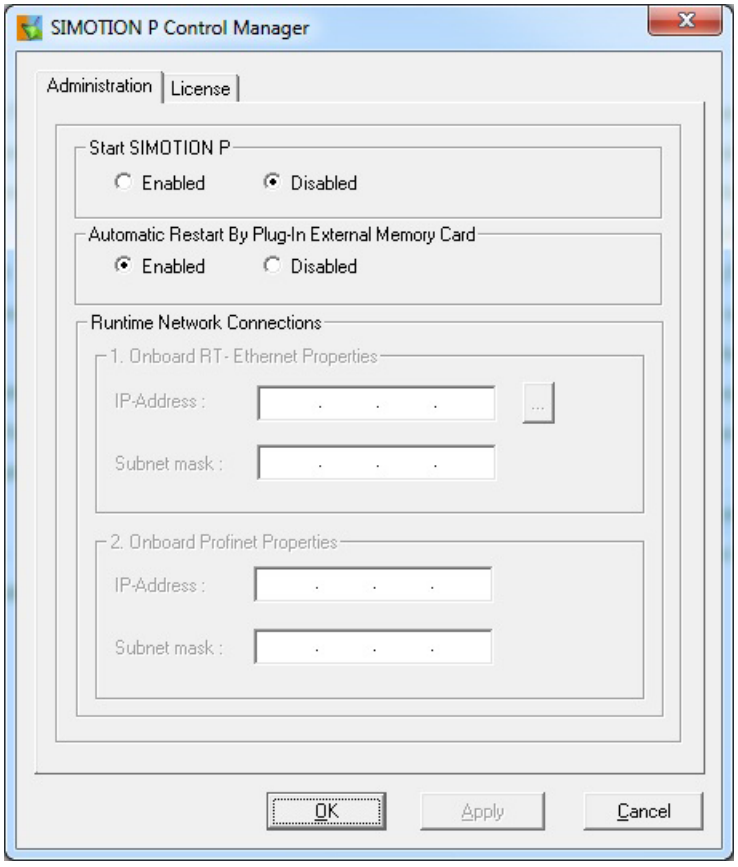


Figure 11-71 Start SIMOTION P - Disabled

SIMOTION P State display

If "Disabled" is selected in the **Administration** tab at **Start SIMOTION P**, the "Disabled" status is also displayed at **SIMOTION P State** .

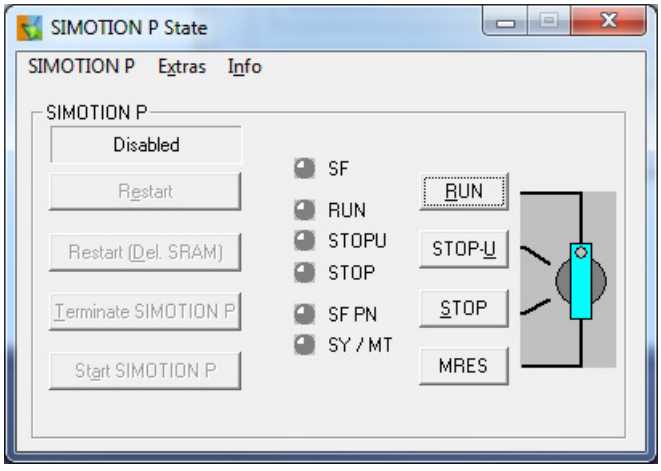


Figure 11-72 SIMOTION P State - Disabled status

Automatic Restart By Plug-In External Memory Card

Note

Not a Windows restart

Note that this is not a Windows restart!

The following settings are possible:

- **Enabled**
  An automatic restart of the SIMOTION P Runtime is performed at the start of the SIMOTION P320-4 with withdrawn CFast card or for a CFast card withdrawn during operation. The automatic restart of the SIMOTION P Runtime for the Enabled default setting is performed when the CFast card is inserted.

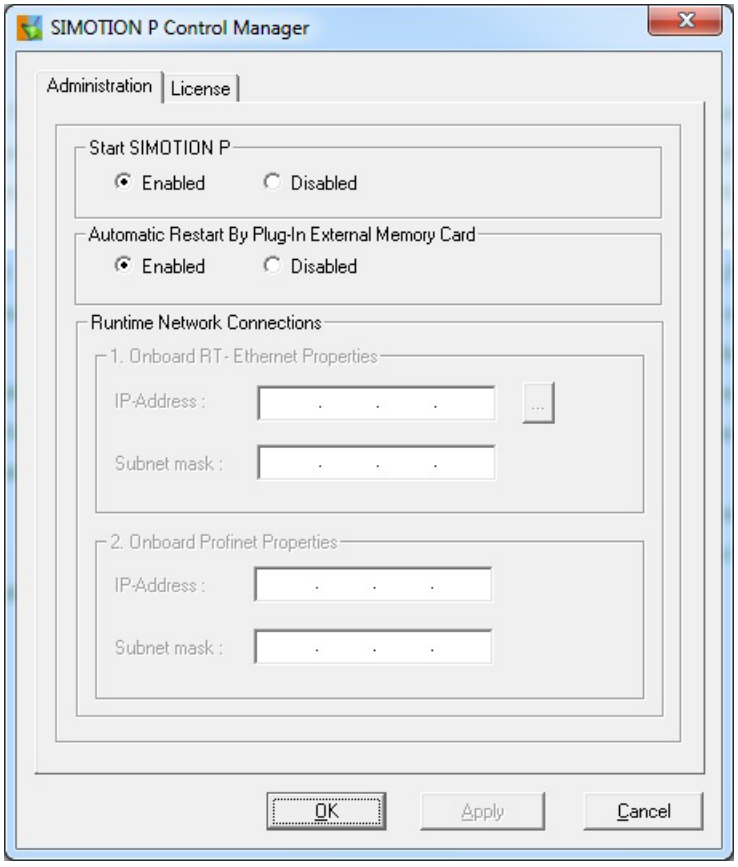


Figure 11-73 Administration tab - Automatic Restart By Plug-In External Memory Card

- **Disabled**
  The automatic restart of the SIMOTION P Runtime is prevented after inserting the CFast card.

Note

Behavior with withdrawn CFast card

Further information on the behavior of the SIMOTION P320-4 with withdrawn CFast card can be found in Section Automatic restart of the SIMOTION P Runtime (Page 7880).

Runtime Network Connections

The Ethernet interface of the SIMOTION P320-4 is displayed here. It can be addressed via the SIMOTION P Runtime.

The IP address can be set here, however, it is not permanent.

The IP address set here loses its validity when the following actions are carried out:

- Booting of the SIMOTION P320-4
- Downloading a project to the SIMOTION P320-4
- Exiting the SIMOTION P Runtime (Terminate SIMOTION P)
- Restarting SIMOTION P Runtime (Restart)

The IP address in SIMOTION SCOUT can be changed permanently using the HW Config or user program.

Changing the IP address

You can change the IP settings after selecting the button, the **Onboard RT - Ethernet Properties** dialog box opens.

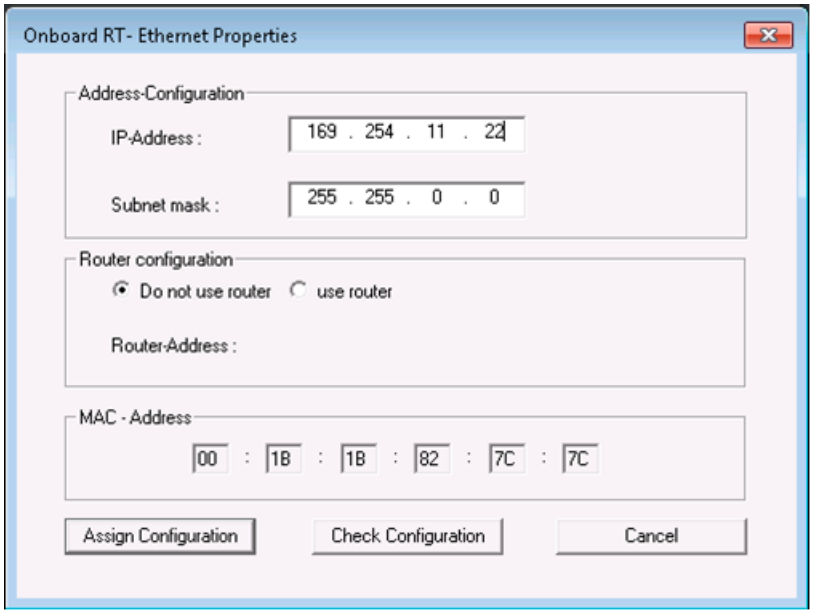


Figure 11-74 Onboard RT - Ethernet Properties

Click **Check Configuration** to validate the settings of the Runtime Ethernet configuration and check for agreement with the Windows Ethernet interface.
You can find further information about this in Section Ethernet communication overview (Page 7929).

Click **Assign Configuration** to also check the settings and accept them.

Note

The IP data is only displayed when the SIMOTION P320-4 is started.

License tab

The **License** tab can be used

- To view currently installed licenses
- To enter new License Keys.

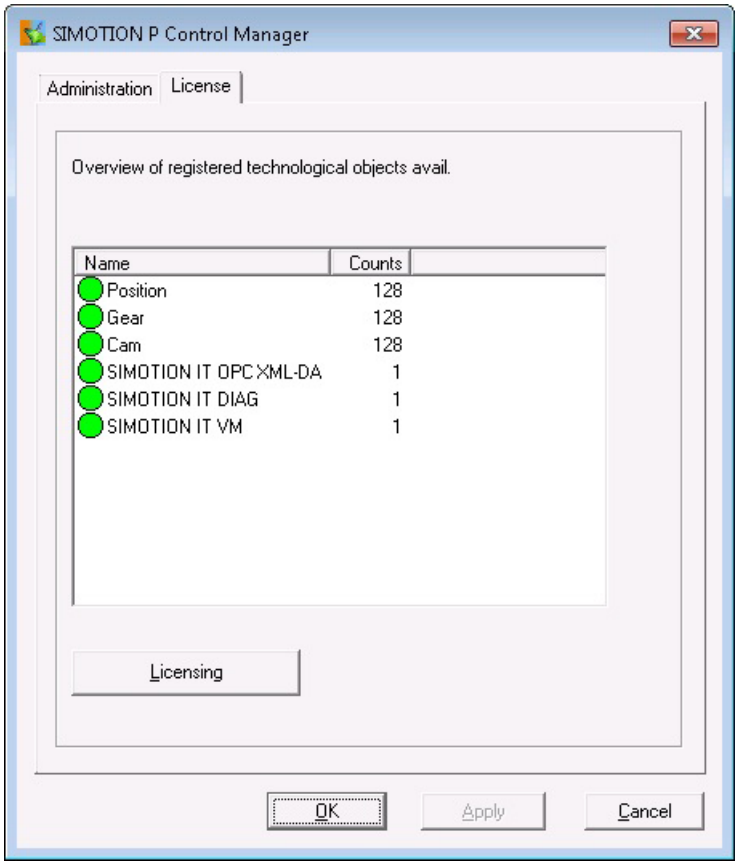


Figure 11-75 License tab

This tab displays the licenses currently installed on this hardware. If you wish to modify the licensing of this hardware, click the **Licensing** button.

The **License Manager** dialog box is displayed; here you enter a new **License Key**.

License Manager

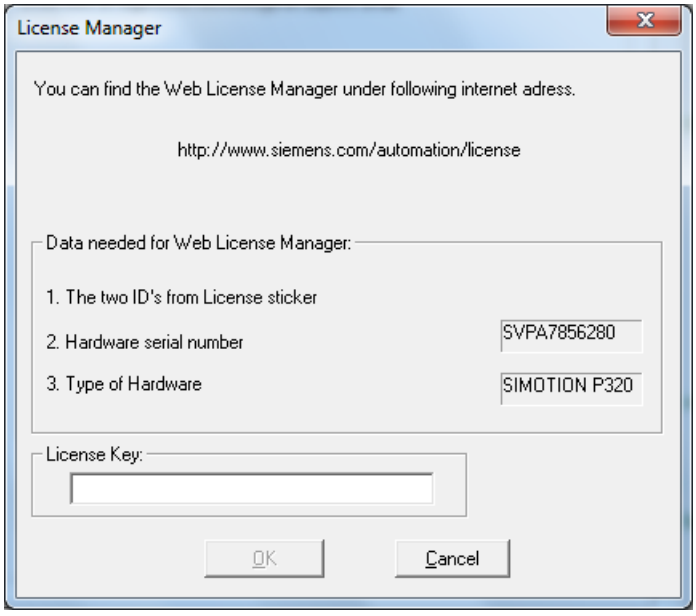


Figure 11-76 Dialog License Manager

Impact of hardware replacement on licensing:

The replacement of the SIMOTION P320-4 means that the License Keys which were generated for the previous SIMOTION P320-4 must be regenerated and transferred to the new SIMOTION device.

In this case, contact the hotline: http://support.automation.siemens.com

Note

You must restart the SIMOTION P320-4 after licensing. To do this, in SIMOTION P State press

- either the **Restart** button
- or the "**Terminate SIMOTION P**" button and then **Start SIMOTION P**.

Note

For more information on licensing, see the SIMOTION SCOUT Configuration Manual.

Local communication via PC internal

Local communication (PC-internal) overview

Local communication in the SIMOTION P320-4 between Windows programs such as HMI and the SIMOTION P runtime software is handled by the PC's internal communications mechanisms.

Checking the PC-internal settings

Configuration in the Windows Control Panel of the SIMOTION P320-4

To enable local communication the following settings must have been made:

You can access the settings via the Control Panel of Windows:

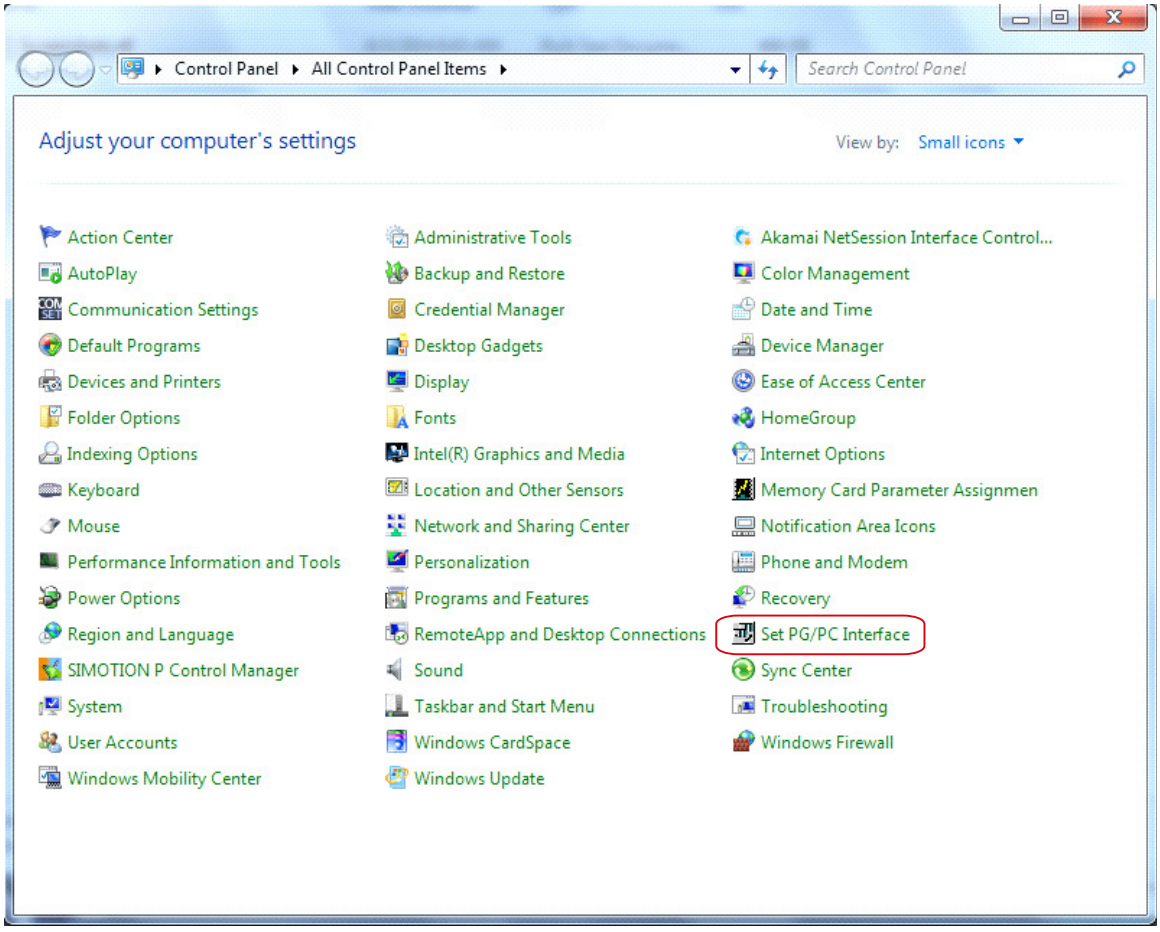
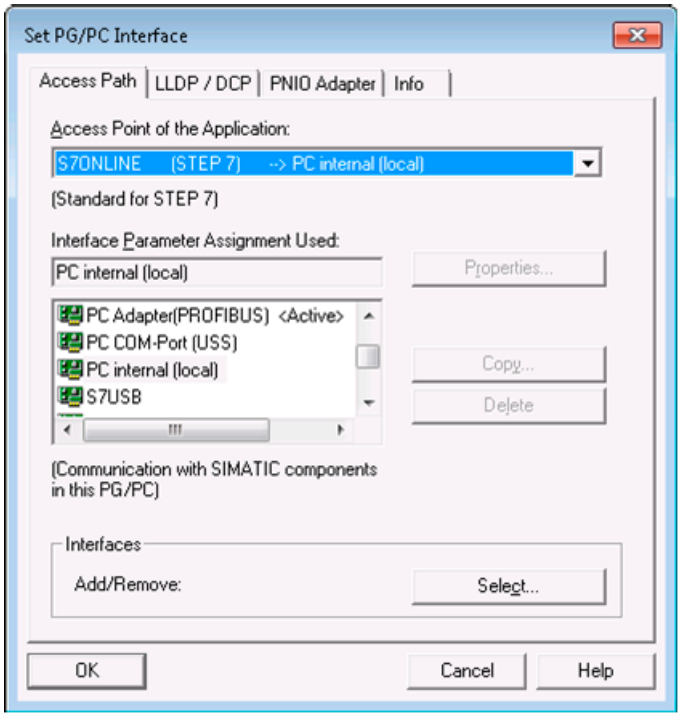


Figure 11-77 Control Panel - Set PG/PC Interface

Procedure

Proceed as follows to set the local communication to "PC internal (local)":

1. Call **Set PG/PC Interface**.
2. In the **Access Path** tab, set the field **Interface Parameter Assignment Used** to **PC internal (local)**.



3. Confirm with **OK**.

Note

SIMOTION P320-4 delivery condition

The preset communication version for the SIMOTION P320-4 is TCP/IP.

Further information can be found in Section Local HMI or ES on SIMOTION P320-4 (Page 7858).

Station Configurator

The **Station Configurator** is required to configure the communication paths for the HMI software.

The Station Configurator is provided directly as desktop icon of the SIMOTION P320-4.

Figure 11-78 SIMOTION desktop with Station Configurator

Note

Changes via Station Configurator

If you want to make changes via the **Station Configurator** , it is essential that you terminate SIMOTION P before calling the function (with **Terminate SIMOTION P** in the SIMOTION P State application).

Observe uniform configuration

Check if the offline configuration set via SIMOTION SCOUT (HW Config) corresponds exactly to the online configuration set in the Station Configurator :

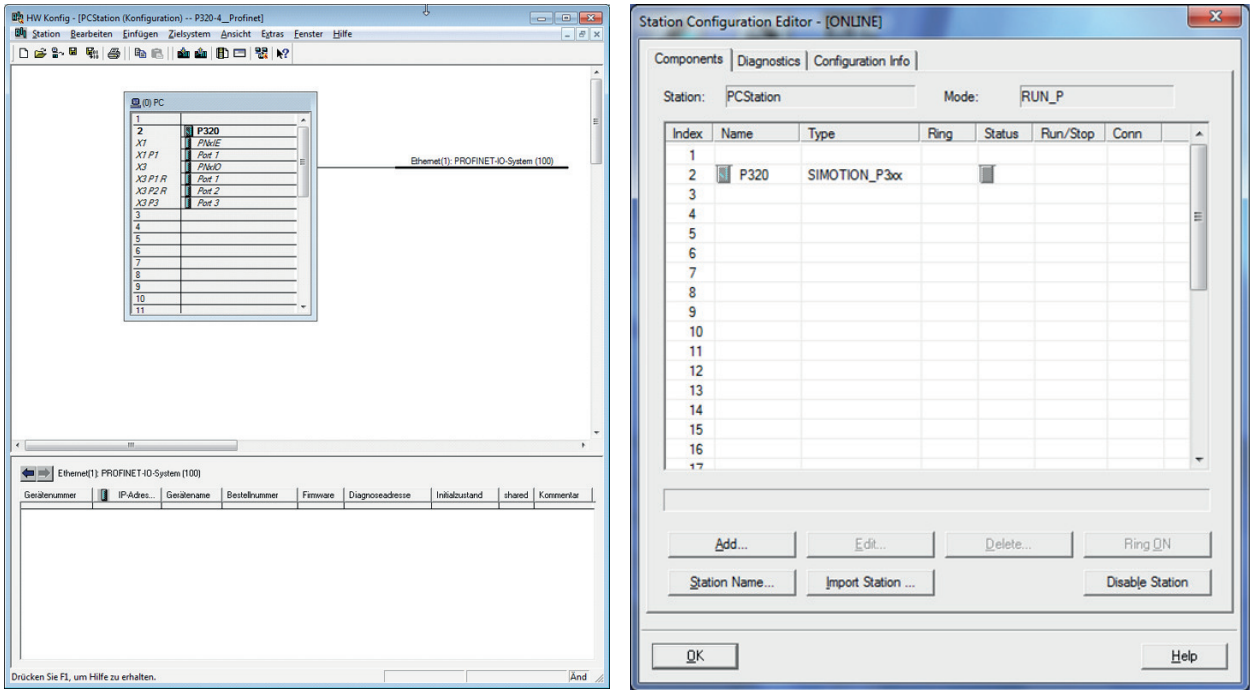


Figure 11-79 Illustration of the offline HW Config (left) - online Station Configurator (right)

Ethernet communication

Ethernet communication overview

SIMOTION P320-4 has one onboard Ethernet interface.

NOTICE
SIMOTION P Runtime During SIMOTION P Runtime, the onboard Ethernet interface must be **activated** under Windows. The onboard Ethernet interface must not be deactivated or activated during operation.

You can connect Industrial Ethernet to the Ethernet interface X1.

Industrial Ethernet is a communication network with a transmission rate of 10/100/1000 Mbps.

Note

Operating Ethernet locally

The SIMOTION P320-4 is configured for the **TCP/IP** communication version in the delivery condition.

See also Local HMI or ES on SIMOTION P320-4 (Page 7858).

Ethernet interface

SIMOTION P320-4 has an onboard Ethernet interface.

An Ethernet interface can always have two IP addresses, one for Windows and one for the access from SIMOTION.

Table 11-8 Ethernet ports overview

Ethernet ports	Access to/from the SIMOTION P Runtime	Access to/from Windows
Onboard Ethernet 1	X	X
PROFINET onboard	X	-

The IP addresses have been set to the default values.

- Onboard Ethernet interface X1 reserved for Windows/SIMOTION (see section Default IP addresses for Windows and SIMOTION P (Page 7931))
  - Windows setting: Default address
  - SIMOTION P Runtime: Default address

If the Windows IP address is taken from a DHCP server, the assignment of the runtime IP address and runtime subnet mask must also be selected in accordance with the default setting of the DHCP server.

The following applies (see Section Communication via an Ethernet interface (Page 7932)):

- Windows IP address ≠ runtime IP address
- Windows IP address and runtime IP address must be in the same subnet

Ethernet interface for SIMOTION P Runtime

The IP address of the **Ethernet interface for SIMOTION P Runtime** can be:

- Displayed and changed via SIMOTION SCOUT
- Displayed in the **Administration** tab via the SIMOTION P Control Manager application. Here, however, the IP address can only be changed temporarily.
  Its validity ends with:
  - Booting, downloading, as well as terminating or restarting of SIMOTION P Runtime.
    Further information can be found in Section Administration tab (Page 7918).

Ethernet interface for Windows

The IP address of the **Ethernet interface for Windows** can **only** be modified via the Windows Control Panel > Network and Sharing Center.

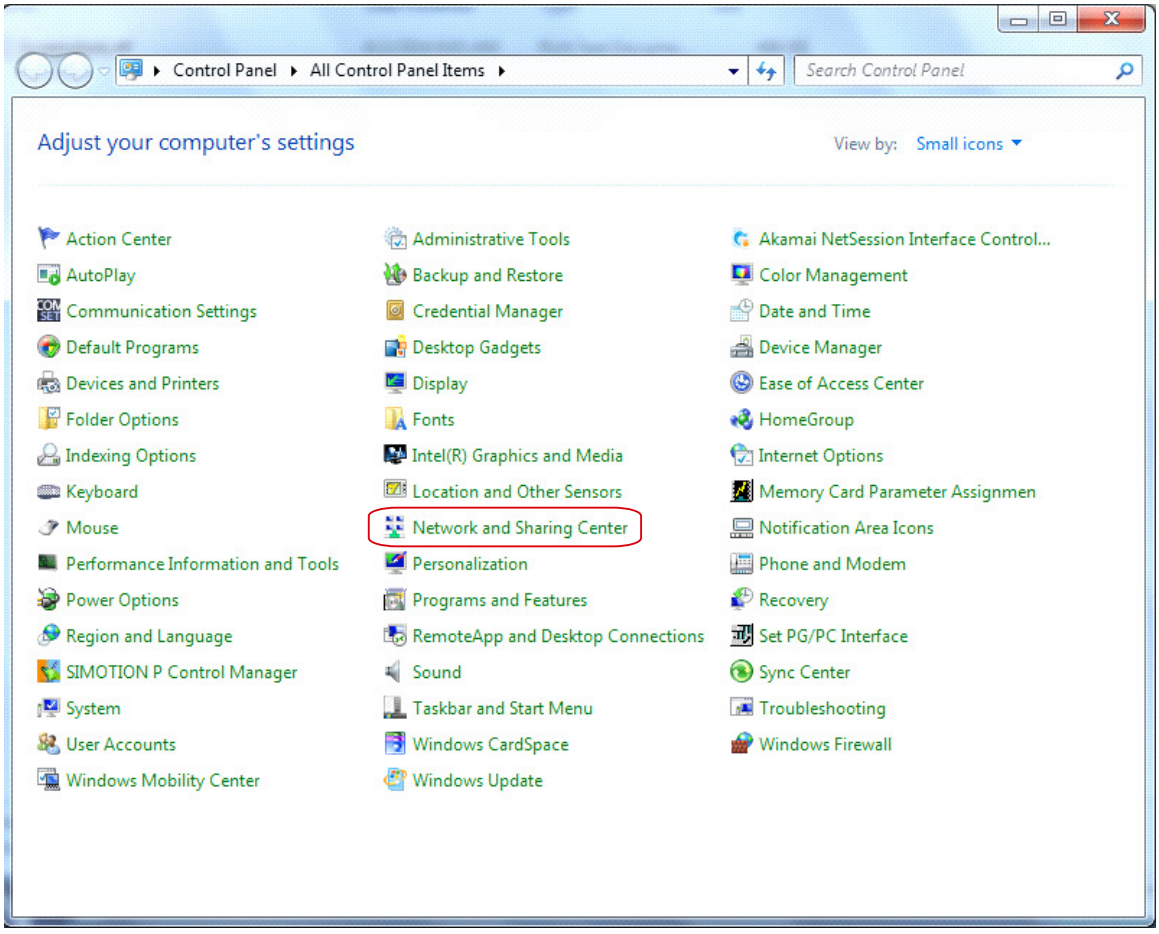


Figure 11-80 Windows Control Panel-Network and Sharing Center

Note

Windows Firewall

To allow the Windows IP to be accessed externally, this must be set as an exception in the Windows firewall **File and Printer Sharing**. Otherwise, there is only limited access to Windows from outside.

Default IP addresses for Windows and SIMOTION P Runtime

Default Ethernet addresses for use from Windows and SIMOTION P are preset.

Table 11-9 Default Ethernet address for Windows and SIMOTION P Runtime

Ethernet interface	Windows	SIMOTION P Runtime
Onboard Ethernet 1	169.254.11.21	169.254.11.22

The default IP address for Windows can result in IP addresses being assigned twice, e.g. during series commissioning (see Troubleshooting/FAQs (Page 7983)).

If DHCP is set under Windows, the IP addresses for Windows are determined automatically.

Communication via an Ethernet interface

If both Windows and SIMOTION use the onboard Ethernet interface X1, the following configuration is present:

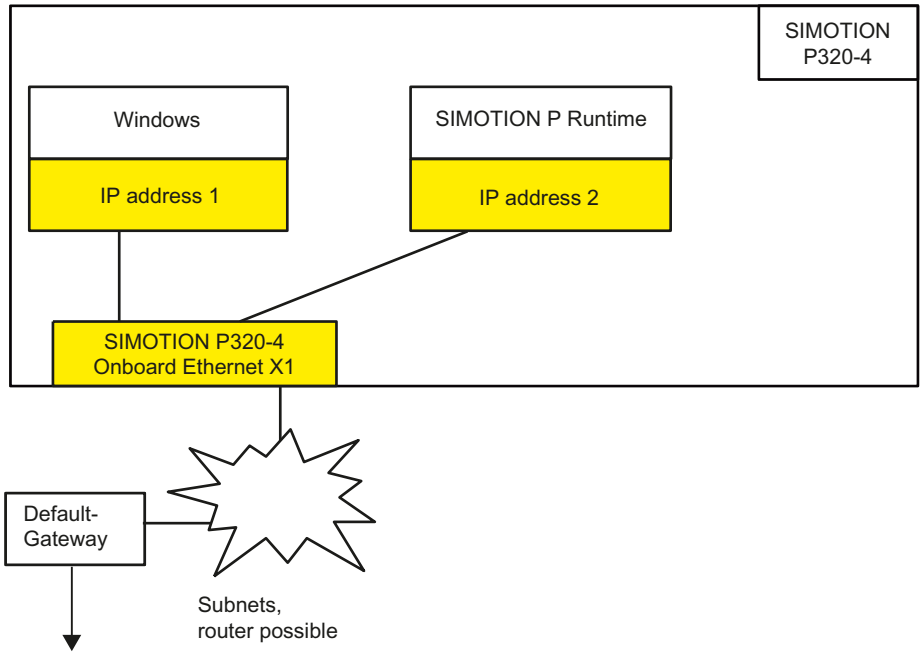


Figure 11-81 Distribution of the IP addresses for Windows and SIMOTION P Runtime

Note

If no change of the default IP addresses is desired, all conditions with the default IP addresses are fulfilled and no change is required.

The following conditions apply:

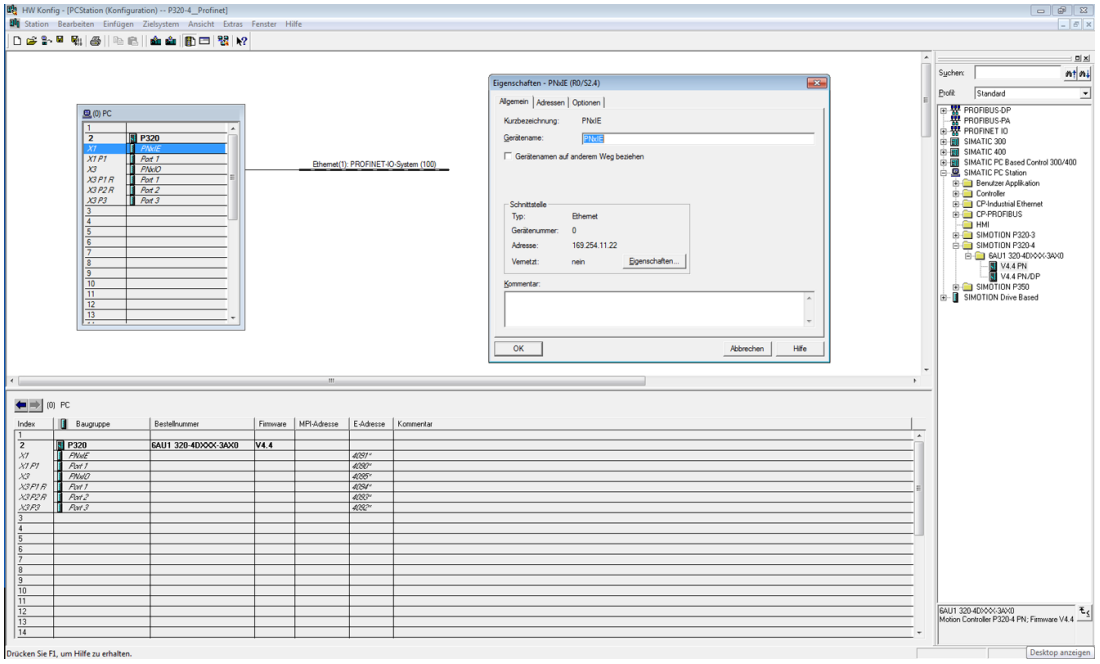
- IP address 1 ≠ IP address 2
- IP address 1 is in the same subnet as IP address 2
- IP address 1 is configured as of Windows 7 under > Control Panel > Network and Sharing Center.

Note

The setting for the default gateway must be identical for the IP addresses 1 and 2.

Procedure in HW Config to change the default IP addresses

1. Double-click **PNxIE** of the PC station. The following dialog box is opened.



2. Click the **Properties...** button of the Properties - PNxIE dialog box.
3. Enter the **IP address** for SIMOTION in the entry field (the default IP address is shown here).

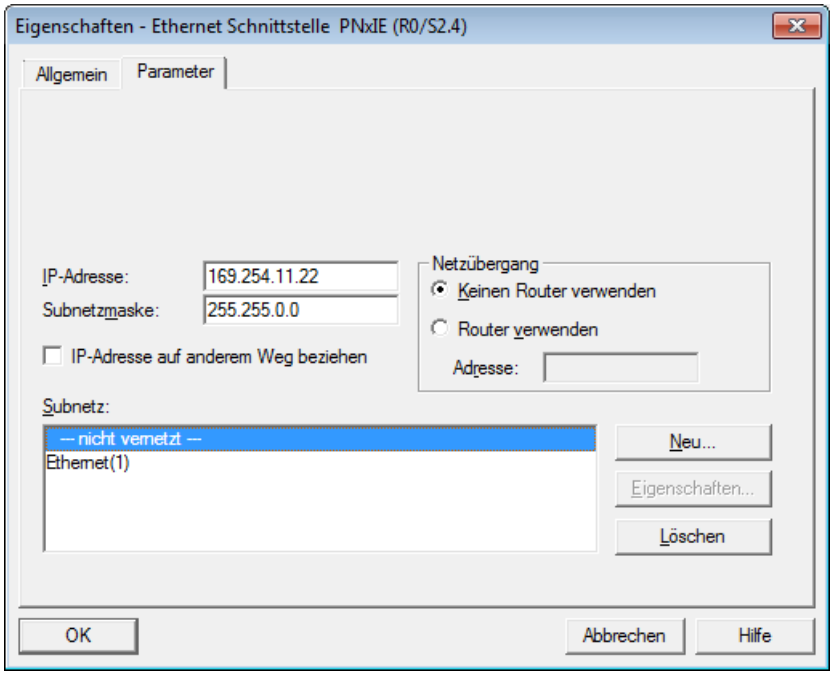


Figure 11-82 Changing the RT address for onboard Ethernet (1) - Properties dialog box

4. Enter the desired subnet mask in the **Subnet mask** entry field and, if required, a router that is to be used as default gateway.
5. Confirm with **OK**.

Activation state of the onboard Ethernet interfaces

The onboard Ethernet interface must be activated for the SIMOTION P Runtime (default setting = activated).

You can check the activation via the **Administration** tab of the **SIMOTION P Control Manager** (accessible via the Windows Control Panel with Start > Control Panel).

The **Activate Device** checkbox is displayed after selecting the button  for **Onboard RT - Ethernet Properties**.

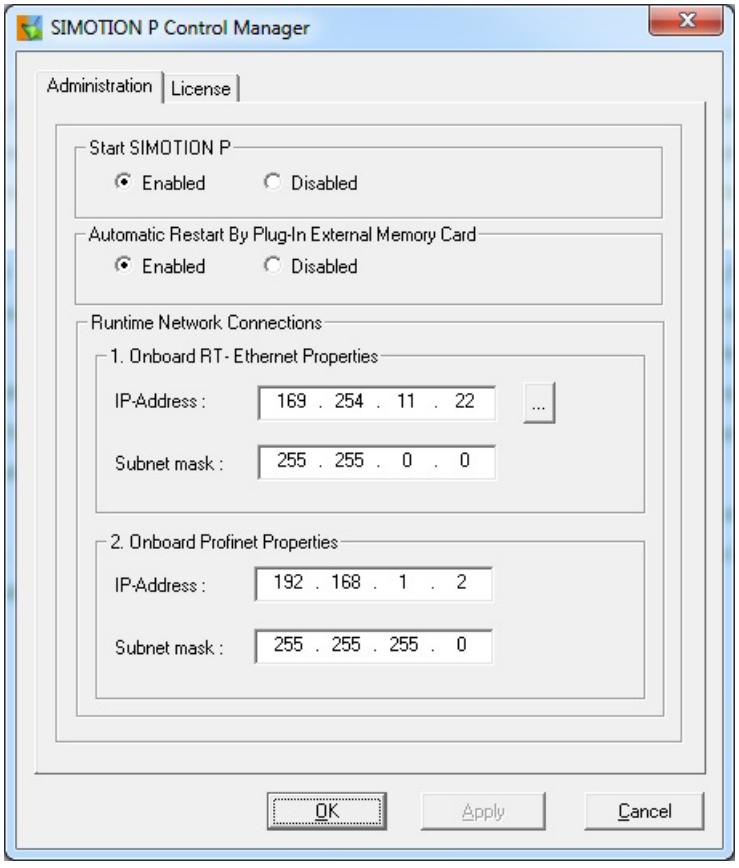
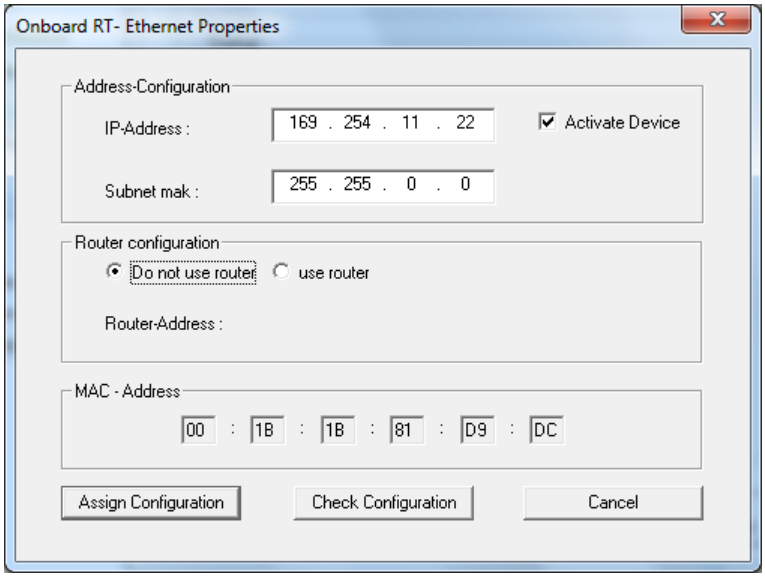


Figure 11-83 SIMOTION P Control Manager - Administration



NOTICE
SIMOTION P Runtime During the runtime of SIMOTION P, the Windows settings must be **activated** for the onboard Ethernet interface. The onboard Ethernet interface must not be deactivated or activated during operation.

Establishing the active flag for an external PG/PC

Requirement

A PG/PC is required to create projects and download them to the target device. SIMOTION SCOUT can only be installed on the SIMOTION P320-4 S.

Note

Observe the online access point for SIMOTION P320-4 S

For SIMOTION P320-4 S, use **S7ONLINE** as access point to avoid inconsistencies.

Procedure

Proceed as follows to open NetPro and assign an interface in the PG/PC:

1. Open the project in SIMOTION SCOUT.
2. Click the button **Open NetPro**.
   NETPRO is accessed, and the configured network is graphically displayed. The PG/PC connection to the configured network is shown not bold and in a color **other than** yellow.

3. Double-click on the PG/PC that you would like to configure.
   The **Properties - PG/PC** dialog box is displayed with the **Assignment** tab in the foreground.
4. In the field **Assigned:** select the interface and activate S7ONLINE access by clicking the corresponding checkbox.
5. Accept the settings by clicking **OK**.
   The PG/PC connection to the configured network is now displayed in **bold** and **yellow**.
6. Save the changes via **Network** > **Save and compile** and upload them to SIMOTION P320-4.

Result

Now you can go online via the PG/PC.

PROFINET communication

Documentation

Description of PROFINET communication

A detailed description of PROFINET communication can be found in:

- SIMOTION Communication System Manual or
- SIMOTION SCOUT Online Help under **Basics** > **Communication** > **PROFINET IO**.

PROFIBUS communication

Documentation

Description of PROFIBUS communication

A detailed description of PROFIBUS communication can be found in:

- SIMOTION Communication System Manual or
- SIMOTION SCOUT Online Help under **Basics** > **Communication** > **PROFIBUS DP**.

Project download

Requirement

The **P320-4 PN** and **P320-4 PN/DP** modules have the same article number. For this reason, there is no distinction between the device during a download.

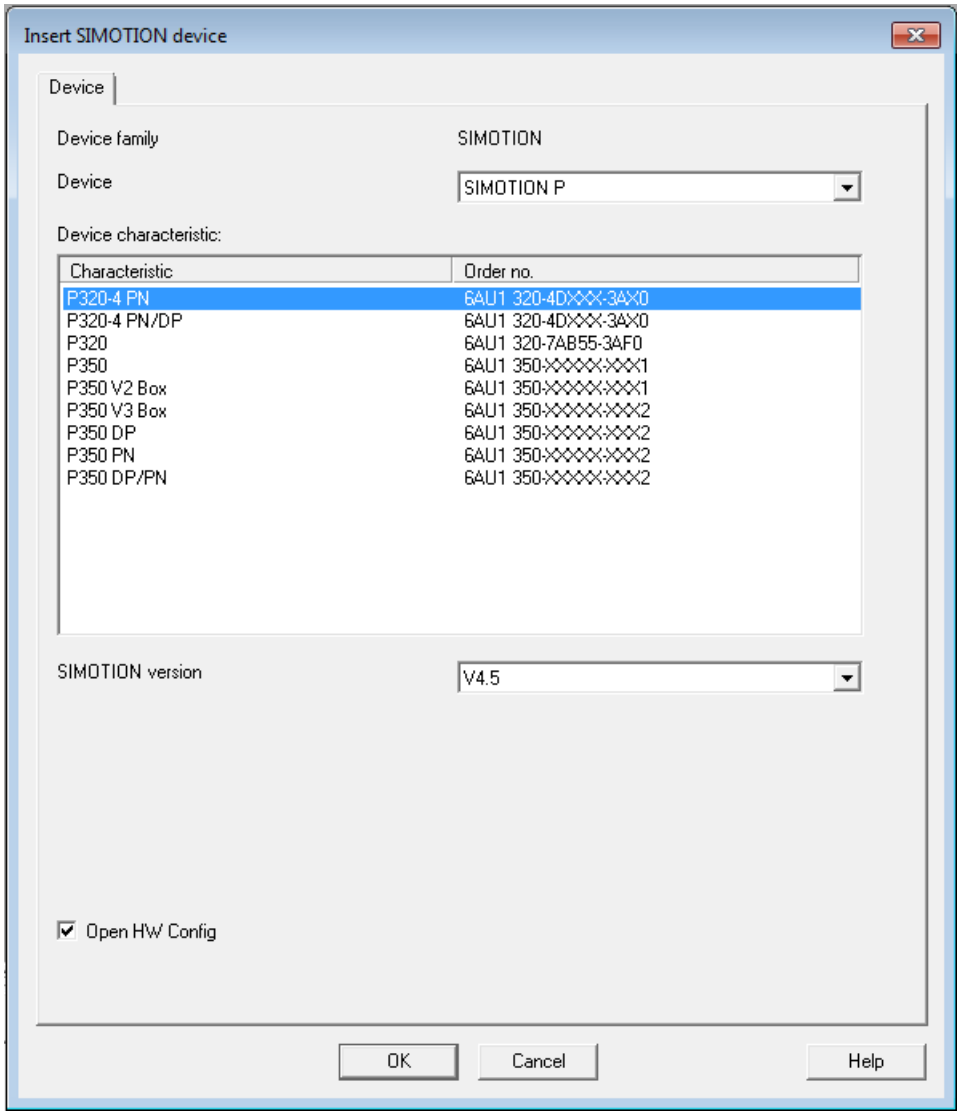


Figure 11-84 Inserting a SIMOTION device - selecting the device version

Behavior of the P320-4 PN device with the P320-4 PN/DP module

- If a **P320-4 PN** device with created PN line is loaded to a **P320-4 PN/DP module**, the project runs exactly as on a **P320-4 PN module**.

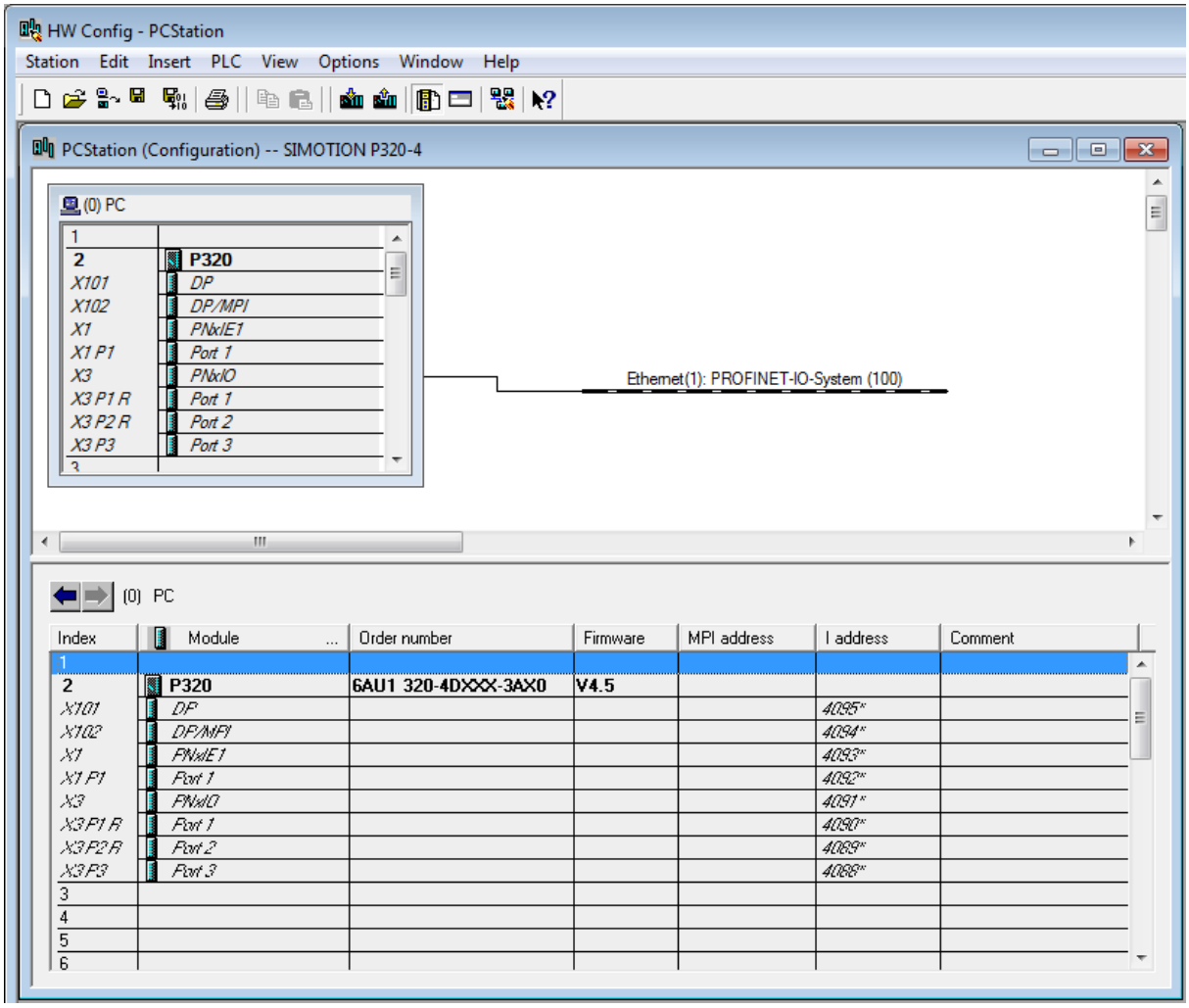


Figure 11-85 P320-4 PN/DP with created PN line (PROFINET-IO system)

- If a **P320-4 PN** device without created PN line is loaded to a **P320-4 PN/DP module**, there is no guarantee that the module runs with the configured base cycle clock.

Behavior of the P320-4 PN/DP device with the P320-4 PN module

Note

A **P320-4 PN/DP** device generally does not run on a **P320-4 PN module**.

The following message is displayed in the diagnostic buffer:

"Source 0 could not be loaded (inconsistency)".

The loaded configuration is not suitable.

Battery monitoring

The installed buffer battery has a service life of at least 5 years. The status can be checked with two-tier battery monitoring.

When the first warning level is reached, the remaining service life of the battery for buffering CMOS data and buffered SRAM is at least 1 month.

You will find further information on the battery warning at: Two-stage battery monitoring (Page 7969)

Lithium battery - ordering data		
Designation	**Article number**	**Further information**
Lithium battery	A5E30314053	Spare part

11.1.1.10 Commissioning (software)

Note before commissioning

Delivery condition

SIMOTION P320-4 is delivered with:

- Windows operating system
- SIMOTION P Runtime application

NOTICE
Risk of damage to the SIMOTION P320-4
If condensation has developed, wait at least 12 hours before commissioning the device.

Basic commissioning - initial startup

Requirement

The following requirements must be met before initial commissioning:

- The device is connected to the 24 VDC power supply.
- Equipotential bonding is connected.

Commissioning requirements

Prior activities

You have already performed the following actions:

- Your system has been installed and wired.
- The installation and parameterization has been completed.
- When networking via PROFINET, the PROFINET addresses of the bus nodes with which the SIMOTION P320-4 communicates are set.

See also

Connection requirements (Page 7891)

Parameter assignment / addressing requirements (Page 7915)

Notes on commissioning

Starting up communications

The following components must be commissioned:

- SIMOTION P320-4
- PROFINET component

Commissioning is performed either via the programming device or by installing the software required for commissioning on the SIMOTION P320-4.

If a panel is used, it must be connected.

Commissioning the components

Please refer to the relevant documentation for the commissioning of the components used with SIMOTION P320-4:

- ET 200 PN I/O
- PROFINET IO drives,

Recommended order for first commissioning

The individual steps for first commissioning are listed below in the recommended order.

1. The complete system is connected mechanically and electrically, and has been verified.
   - SIMOTION P320-4
   - Drives
   - Motors
   - SIMATIC S7 I/O components
   - HMI user interfaces
2. The article numbers of the drives and SIMATIC S7 I/O components should be available. You need these order numbers when setting up a SIMOTION project to verify that the components selected from the hardware catalog in the **HW Config** application correspond to the ones used in the system.
3. Prepare the drives for communication on the PROFINET IO.
4. Commission the drives via PROFINET IO.
   Literature: Function description of the associated drives
5. Carry out a test run for all axes.
6. Perform the optimization of the drives.
7. Switch on and power up the SIMOTION P320-4; all the basic settings are made automatically.

See also

License tab (Page 7924)

Data backup

Memory media

The storage media of the SIMOTION P320-4 are divided into internal and external memories on drives C: and D:.

The SIMOTION P Kernel and Windows system software are located on the internal storage medium on drive C:.

Media

The following figure shows the partitioning of the storage medium in the delivery state, depending on the version of the SIMOTION P320-4.

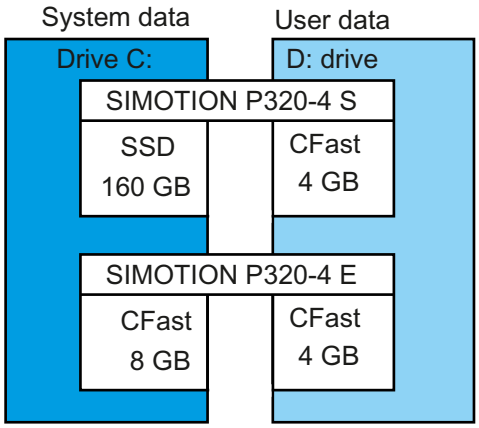


Figure 11-86 Storage media of the SIMOTION P320-4

Drive C:

Drive C: reserved for the Windows system software, SIMOTION P kernel, etc.

NOTICE
Compression for drive C: Not permissible Compression must not be activated for files and folders on drive C: for correct operation of the SIMOTION P320-4.

Drive D:

Drive D: contains the **user data**, e.g. programs, configuration data, parameter assignments of SIMOTION P Runtime.

Note

Storage of user data

It is recommended that only the user data be saved on drive D:.

SIMATIC data backup

Data backup under Windows Embedded Standard 7 32-bit / Windows 7 Ultimate 32-bit

For backing up data under Windows Embedded Standard 7 32-bit and Windows 7 Ultimate 32-bit, we recommend the software tool **SIMATIC IPC Image & Partition Creator** (as of V3.3.1). This tool allows the easy backing up and quick restoration of entire memory cards.

The SIMATIC IPC Image & Partition Creator only supports the burning of DVD media.

This software is available via the online ordering system Siemens Industry Mall (http://www.siemens.com/industrymall). You can find more accurate information about the tool in the associated product documentation.

Restore DVD

SIMOTION P320-4 is supplied with pre-installed hardware and software and is therefore ready for connection and operation.

The delivery kit includes a restore DVD. Using the restore DVD, you can restore the delivery state of the SIMOTION P320-4, e.g. in case of a problem when installing additional software packages such as HMI software.

It is recommended that you create a data backup after commissioning by the OEM. Save the data backup on an external hard disk or a USB flash drive.

The fundamentals of the data storage concept and the various types of data back-ups of the SIMOTION P320-4 are described in the section on the Data storage concept (Page 7944).

Note

SIMOTION licenses

SIMOTION Runtime licenses are retained (e.g. axis licenses) when the SIMOTION P320-4 is restored to its delivery state.

Only the licenses of the **Automation License Manager** are deleted (e.g. for the SIMOTION SCOUT engineering system).

Power-up of the SIMOTION P Kernel

The SIMOTION P Kernel is started by default when Windows is booting. It is therefore available after each power-up of the SIMOTION P320-4. The startup process is triggered by an installed service, the **SIMOTION P Startup Service**. This service ensures that the SIMOTION P Kernel starts automatically during the Windows power-up.

SIMOTION P320-4 is started with the project last saved from RAM to ROM and starts running immediately if the operating state was set to RUN when the system was switched off.

If you want to explicitly terminate the SIMOTION P320-4 without exiting Windows, use the **SIMOTION P State** application.

To do this, you must be logged on with administrator privileges. Only the administrator is offered the button for the **Terminate SIMOTION P** function.

After explicitly terminating the SIMOTION P Kernel via the SIMOTION P State application, the SIMOTION P Kernel can be restarted via this application (Start SIMOTION P button). A restart of Windows also results in a restart of the SIMOTION P Kernel.

You can find more detailed information about SIMOTION P State in Section SIMOTION P State application (Page 7909).

Data storage concept

Data storage concept overview

Overview

The backup possibilities described in the following section can be used to backup and restore the complete SIMOTION P320-4 installation with the current configuration and customization.

The SIMOTION version on the target device does not need to match the version of the device on which the backup was created.
Under certain preconditions, e.g. identical hardware for storage medium image, series commissioning can also be carried out using these backup and restoration mechanisms.

Storage medium image

With a storage medium image, the entire storage medium can be backed up and restored. This allows series commissioning to be performed (requirement: identical hardware!).

A storage medium image must be created both for the internal storage medium (Drive C) and for the external CFast card (Drive D). When creating the image with the **SIMATIC IPC Image Creator**, an image can be created jointly for one internal and one external storage medium.

When loading an image, be aware that the images for the internal and external storage media must be loaded separately and that the wizard must therefore be run twice. It is not possible to load the image onto both drives in a single process.

The restore DVD contains a storage medium image of the plant with which the delivery state of the SIMOTION P320-4 can be restored.

Various mechanisms are available for backing up and restoring data for the SIMOTION P320-4.

These mechanisms are described in the following sections, starting with the largest possible scope of a backup to the backing up of variables.

Although the backup/restore of data in many cases is initiated by the user, it is sometimes also performed automatically or started by the user program.

In addition to restoring defined states and updating/reloading a project in the target device, data backups are also needed to prevent data loss resulting from a hardware defect or failure of the backup battery.

Project update

The Project update (Page 7946) section describes options for updating/retransferring or backing up a SIMOTION project.

Application

- Transfer of a project to the newly installed SIMOTION P320-4
- Series commissioning

Advantage

Series commissioning is also possible without a programming device or using remote maintenance.

Disadvantages

The SIMOTION version on the target device must exactly match the version of the device on which the backup was created.

Single backups

The Single backups (Page 7950) section describes the options for backing up and restoring retain variables, retain TOs, unit variables, and global device variables.

Application

- Data backup when replacing the SIMOTION P320-4
- Restore the SRAM after a failed consistency check (battery failure,...)
- Backup the data for Unit variables and device-global variables from the RAM
- Backup values (variables) from the RAM
- Restore a defined state of the machine

Advantages

- The user program can use system functions to perform a backup
- Version-independent restoration of the retain data (non-volatile SIMOTION data)

Disadvantages

No complete backup

Defined states

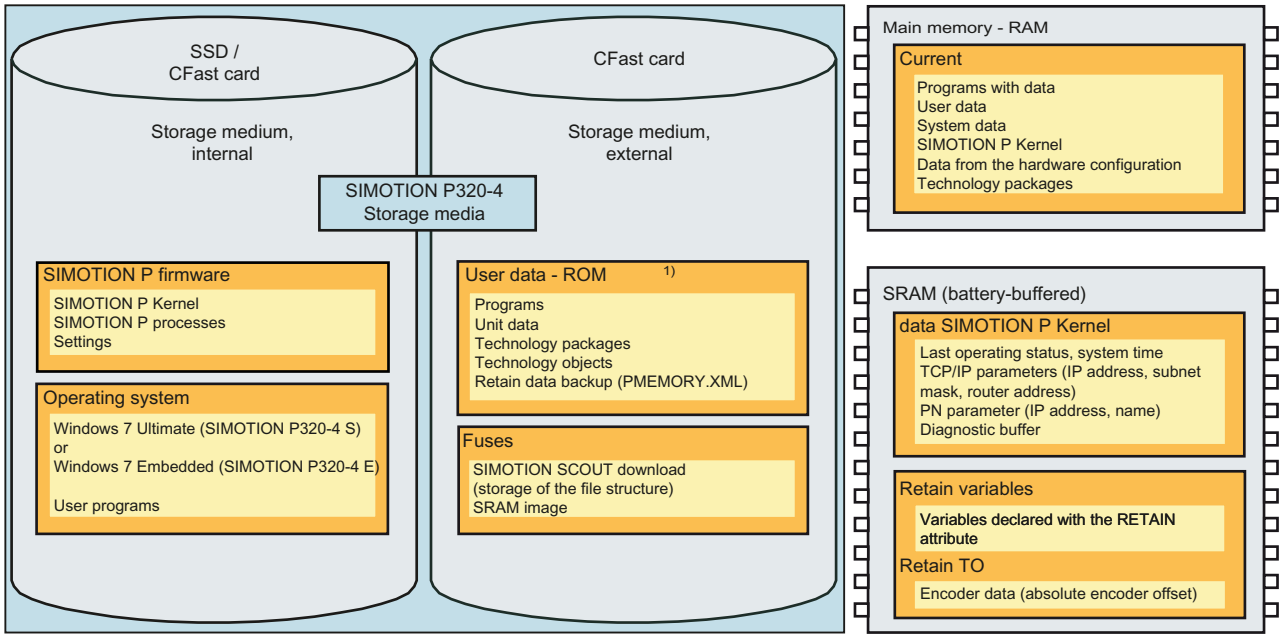
The Defined states (Page 7954) section describes the options for restoring a defined state / initial state.

Application

- System defect
- Windows defect
- Restore a defined initial state, for example, during commissioning

Memory model

The following figure shows which data memories are used by SIMOTION P320-4 and which data is saved in them.



1) The user data must not be written by Windows during runtime (SIMOTION P State = RUN).
User data can be written from the Explorer or other applications in the **Terminate SIMOTION P** state.

Figure 11-87 SIMOTION P320-4 memory model

Project update

Overview

The backup possibilities described in the following section can be used to update, retransmit or back up a SIMOTION project.

The **download of project data** (file structure or ZIP file) via **SIMOTION SCOUT** can be used to perform series commissioning of a SIMOTION project.

SIMOTION IT allows series commissioning using remote maintenance via a web browser. Supported web browsers are, for example, Internet Explorer or Mozilla Firefox.

Detailed information about the backup options for the project update can be found at:

- SIMOTION Project Download online (Page 7947)
- SIMOTION SCOUT - download of the project data (file structure) (Page 7947)
- SIMOTION SCOUT - download of the project data with SIMOTION IT (ZIP file) (Page 7948)
- Backing up and restoring project data (file structure) (Page 7949)

SIMOTION Project Download online

The project data is loaded from a programming device with engineering system to the work memory (RAM) and selected parameters (e.g. TCP/IP) to the SRAM or exchanged between SRAM and work memory (RAM).

The explicit import of user data into the external CFast card is initiated by SIMOTION SCOUT (**Copy RAM to ROM**).

Note

Identical names in SIMOTION SCOUT / Station Configurator are essential

It is essential for the download that the name of the station is identical in the engineering system (SIMOTION SCOUT) and in the Station Configurator of the SIMOTION P320-4.

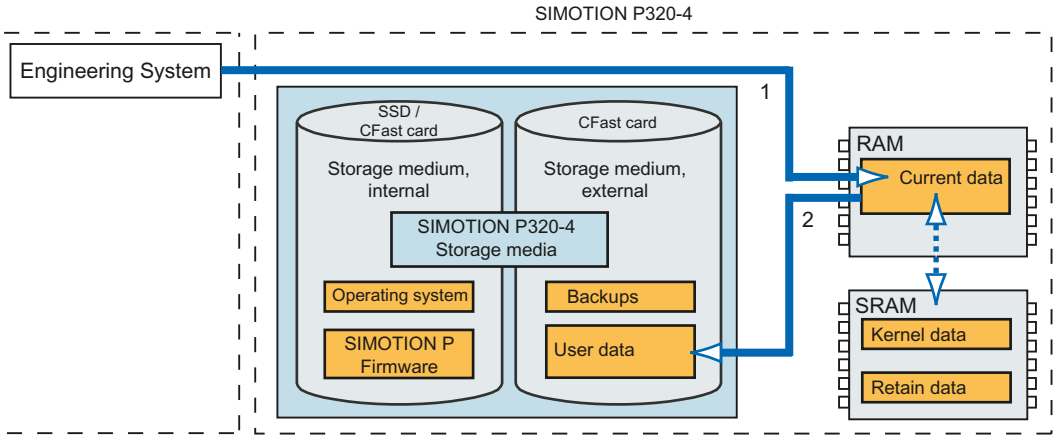


Figure 11-88 SIMOTION SCOUT download from the engineering system

SCOUT project download online process

1. Load the project data via **SIMOTION SCOUT**, menu **Target system > Load > Load project into target system...**, into the working memory of the SIMOTION P320-4.
2. To transfer the data to the external storage medium, use the menu **Target system > Copy RAM to ROM** in SIMOTION SCOUT.

The project data is transferred to the controller.

SIMOTION SCOUT - download of the project data (file structure)

The project data is saved by SIMOTION SCOUT in the form of the file structure in a file system (e.g. a programming device with an engineering system).
The user copies this file structure to the external CFast card of the SIMOTION P320-4.

This permits a series commissioning of SIMOTION programs.

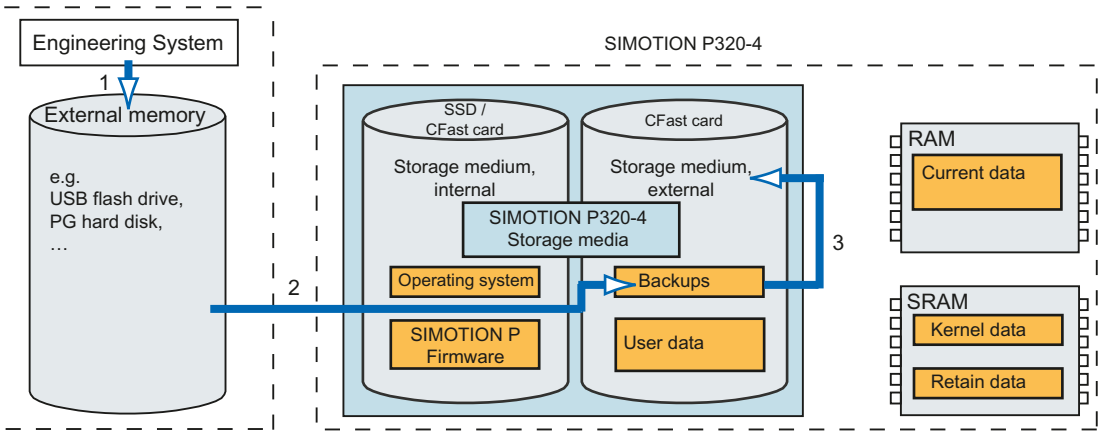


Figure 11-89 SCOUT download (file structure)

Download of the project data process (file structure)

1. Save the data via SIMOTION SCOUT, menu **Edit > Load to file system ...** (option **Save normally**), on the file system of your programming device.
2. Copy this data (file structure) to the external CFast card (drive D) of SIMOTION P320-4 (e.g. with USB memory stick and Windows Explorer).
   SIMOTION P must be in the **terminated** state for this.

Note

Note the SIMOTION version

The SIMOTION version on the target device must exactly match the version of the device on which the backup was created.

SIMOTION SCOUT - download of the project data with SIMOTION IT (ZIP file)

SIMOTION SCOUT stores the project data in the form of a ZIP file to a file system (e.g. a programming device with engineering system). This ZIP file can be transferred by the user to the SIMOTION IT web server with the web browser and it is then automatically imported into the user data.

This permits a project download using remote maintenance or series commissioning.

Note

Note the SIMOTION version

The SIMOTION version on the target device must exactly match the version of the device on which the backup was created.

After the upload of a SIMOTION project into the SIMOTION P320-4 (SIMOTION device update), the time stamps in the user data and the SIMOTION SCOUT project are different.
The project is shown as inconsistent in SIMOTION SCOUT.
The consistency can be established via a project download or a download of the HW Config in the SIMOTION SCOUT.

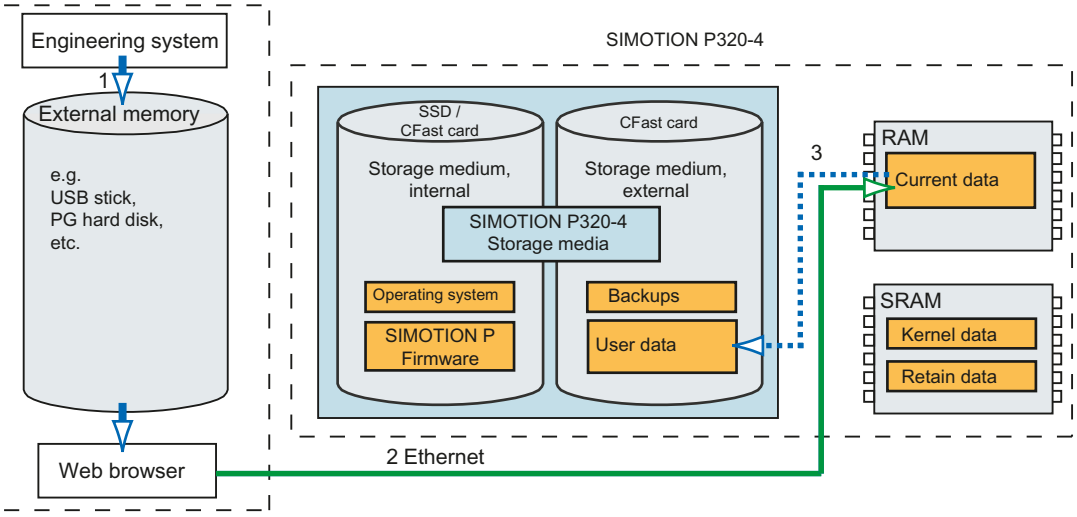


Figure 11-90 SIMOTION SCOUT download (ZIP file)

Download of the project data process (ZIP file)

1. Save the data via SIMOTION SCOUT, menu **Edit > Load to file system ...** (option Save compressed), on the file system of your programming device.
2. Transfer the ZIP file via the web browser and the project download of SIMOTION IT**:** **Manage Config** > [Login dialog] > **Device Update** > **Send new update data (*.ZIP)**
3. The unpacking and the import of the data into the user data on the external CFast card is performed automatically.

Note

SIMOTION IT

You can find further information about SIMOTION IT in the following documentation:

- SIMOTION IT Diagnostics and Configuration Diagnostics Manual
- SIMOTION IT Programming and Web Services Programming Manual
- SIMOTION IT Virtual Machine and Servlets Programming Manual

Backing up and restoring project data (file structure)

The project data (file structure) is located in the **USER** directory on the **external CFast card** (drive D) of the SIMOTION P320-4.

1. To save the project data, copy the USER directory from the external CFast card, e.g. to a storage medium of the SIMOTION P320-4 or to a USB flash drive.
2. To restore the project data, copy the USER directory, e.g. from a storage medium of the SIMOTION P320-4 or a USB flash drive, to the external CFast card and replace the existing directory.

This also enables series commissioning without a programming device or SIMOTION SCOUT.

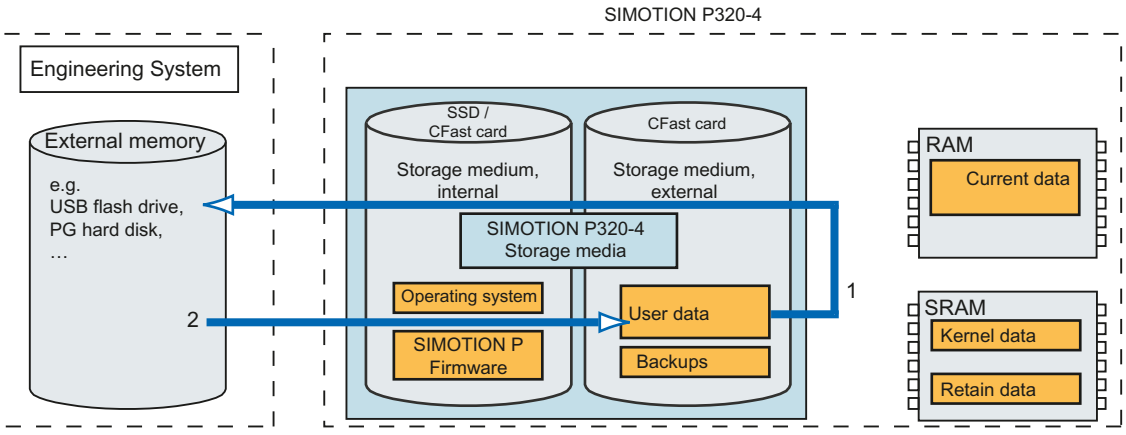


Figure 11-91 Backing up and restoring a file system

Sequence of a series commissioning

Copy the **USER** directory from a storage medium to the external CFast card (Drive D) of the SIMOTION P320-4 to be newly commissioned.

SIMOTION P320-4 must be in the **terminated** state for this.

Single backups

Overview

The backup options described in the following section can be used to back up and restore retain variables, retain TOs, unit variables and global device variables.

If the consistency check results in an SRAM error (because of a power failure if the battery has fallen below the threshold value) or the SIMOTION P320-4 is replaced, the backup can be used to restore the contents of the SRAM.

A user program can use system functions to initiate a backup.

Unit variables and global device variables can be backed up from the RAM to allow them to be recovered in the event of a power failure, for example.

This allows the initial state of the machine (reference points, sensor values, ...) to be restored.

Detailed information about the individual backup options can be found under:

- Backup/restore SRAM (Page 7951)
- Backing up non-volatile SIMOTION data (retain data) (Page 7951)
- Restoring non-volatile SIMOTION data (retain data) (Page 7953)
- Backing up unit variables and global device variables (Page 7954)

Back up/restore SRAM

The correct shutdown of Windows automatically initiates **Terminate SIMOTION P** (SIMOTION P State). This automatically saves the retain data from the SRAM to the external CFast card.

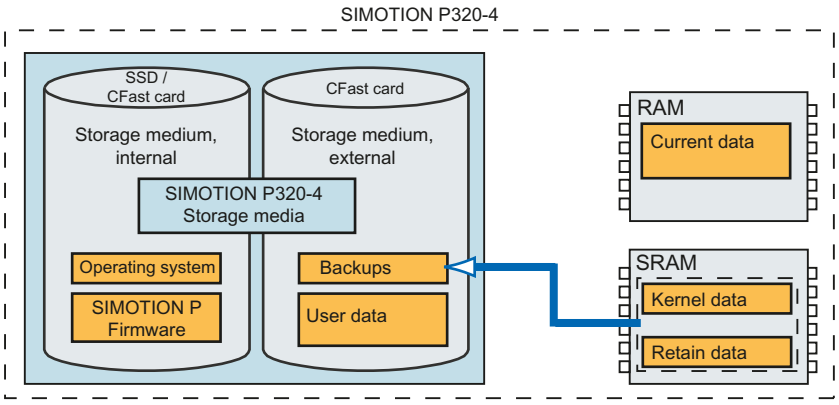


Figure 11-92 Saving SRAM with Terminate SIMOTION P

A consistency check of the SRAM is performed at each SIMOTION power-up. If an error is detected here, the SRAM data will be restored from the backup.

If, for example, the SIMOTION P320-4 needs to be replaced, this backup/restore mechanism allows the data to be transferred to the SIMOTION P320-4.

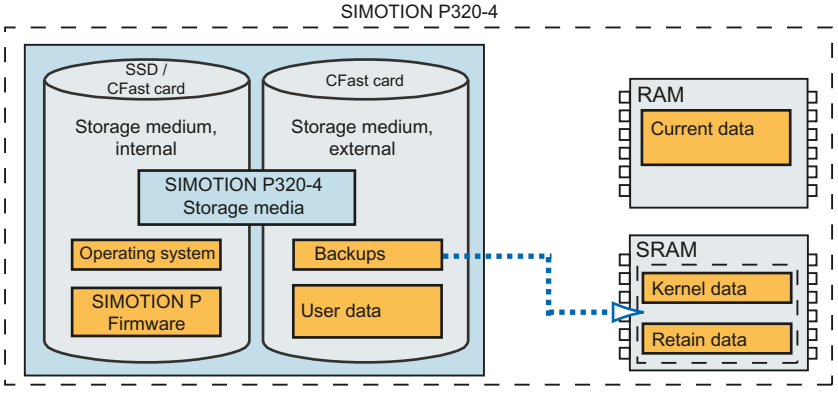


Figure 11-93 SRAM restore at startup

Further information

can be found in the section Non-volatile data and SRAM handling (Page 7972).

Backing up non-volatile SIMOTION data (retain data)

The non-volatile SIMOTION data (retain data) can be backed up from the user program and via SIMOTION SCOUT.

You can use the backup to restore the retain data independent of the version after an update or a hardware replacement (SIMOTION P320-4) or to create a current backup to use in the event of an inadvertent deletion.

You can save the retain data as follows:

- From the user program with the **_savePersistentMemoryData()** system function
  The contents are saved to the "PMEMORY.XML" backup file in the "USER\SIMOTION" directory.
  The data is saved in the PMEMORY.XML file on the external CFast card.
- To back up the retain data via SIMOTION SCOUT , go online and in the context menu of a created source, execute **Save variables...**.
  The data is saved in the **unitdata.xml** file on the computer with SIMOTION SCOUT .
- Alternatively, the retain data is saved via the **SIMOTION P State** application, via the **SIMOTION P > Set Diagnostic Switch** menu.
  The contents are saved to the "PMEMORY.XML" backup file in the "USER\SIMOTION\HMI \SYSLOG\DIAG" directory.
- To restore the retain data via SIMOTION SCOUT, go online and execute **Restore variables...** in the menu.
  Restores the saved variables, also over different versions.

On the system side, the **_savePersistentMemoryData()** system function ensures that a consistent overall image of the non-volatile SIMOTION data is always available the next time the device is switched on, even if there is a power failure during backup.

An already existing backup file is renamed to "PMEMORY.BAK" before a new backup file is generated. If the save operation to the new backup file fails (e.g. because the capacity of the external CFast card is insufficient), this backup copy of the backup file is used the next time an attempt is made to restore the non-volatile SIMOTION data content. If the new file is successfully created, the backup copy of the backup file is deleted.

NOTICE
Loss of data due to failure to make backup copies Back up the non-volatile SIMOTION data on the CFast card. Non-backed-up non-volatile SIMOTION data can be lost on hardware replacement (module defect). For example, failing to back up the current values of the retain variables will result in them being lost and reset to their initial values.

NOTICE
Homing required again after absolute encoder overflow If an absolute encoder overflow occurs after _savePersistentMemoryData, the actual position value is no longer correct after the non-volatile data is restored. In this case, homing (absolute encoder adjustment) must be repeated.

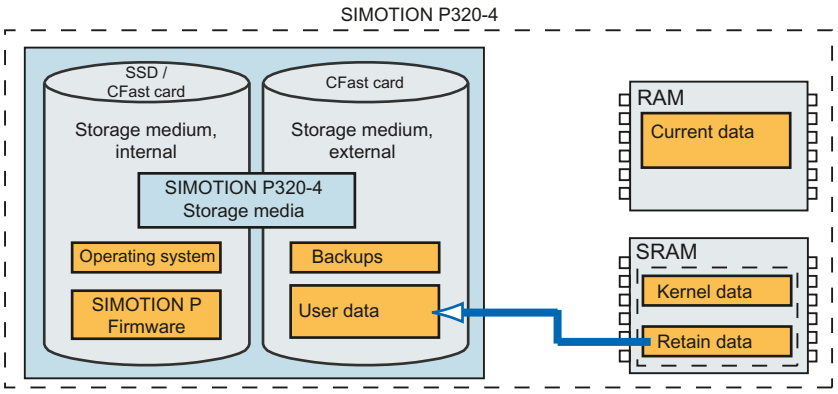


Figure 11-94 Save retain data

Restoring non-volatile SIMOTION data (retain data)

The backup of the non-volatile SIMOTION data (variables declared as retain) is restored in the memory if a data inconsistency is detected after a power failure (the backup battery has fallen below a threshold value) or a general reset has been performed on the CPU. This means that the user can perform a general reset to force the data to be reloaded.

Condition for restoring is that the **PMEMORY.XML** file is present on the external CFast card and the content of this backup file matches the project on the external CFast card.

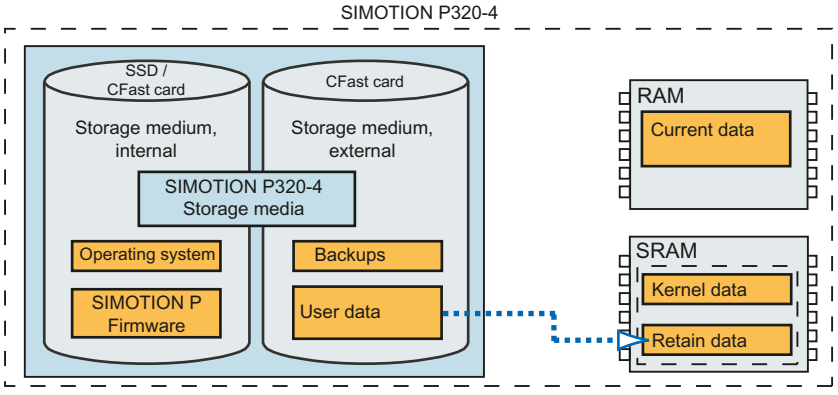


Figure 11-95 Restore retain data

Note

In the event of a power failure of the SIMOTION P320-4, the retain data in PMEMORY.XML will **not** be restored automatically.

Note

If a data inconsistency exists in the event of a power failure because the backup battery has fallen below a threshold value, the data backup from PMEMORY.XML will be written back into the SRAM.

Save unit variables and global device variables

Unit variables and global device variables are never retentive; the data will be lost in the event of a power failure.

If the unit variables and global device variables which have not been declared as retain variables are to be recovered after a power failure, the user program can use system functions to save this data to data sets (files on the external CFast card).

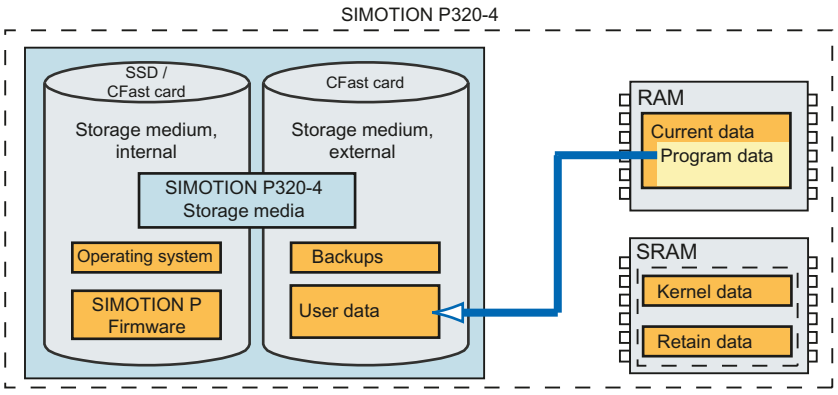


Figure 11-96 Saving unit variables and global device variables

System functions for backing up / restoring unit variables and global device variables:

- _loadUnitDataset()
- _saveUnitDataset()
- _importUnitDataset()
- _exportUnitDataset()

With unit variables, however, generally only unit variables of the interface section can be saved. In addition, all variables of the unit declared as retain variables will be saved.

Save variables... SIMOTION SCOUT function:

- The data is saved in the unitdata.xml file on the SIMOTION SCOUT computer.

Restore variables... SIMOTION SCOUT function:

- Restores the saved retain variables, also across different versions.

Defined states

Overview

An initial state can be restored by loading data backups, as described in the section Project update (Page 7946), or via the reset functions of the **SIMOTION P State** application.

Transfer backup from the file system

Download backups that were stored on the file system using the backup mechanisms described previously.

For further information see Single backups (Page 7950).

Detailed information about the backup options for detailed states can be found at:

- Restore the factory setting using SIMOTION P State (Page 7955)
- Overall reset using SIMOTION P State - operating mode switch setting MRES (Page 7956)
- Delete SRAM via SIMOTION P State - button Restart (Del. SRAM) (Page 7956)

Restore the factory setting using SIMOTION P State

The condition on delivery is restored via the menu item **Extras > Reset to factory default**.

Note

The SRAM and the PMEMORY.XML file will also be deleted.

The controller is restarted automatically.

Any licenses on the card will not be deleted.

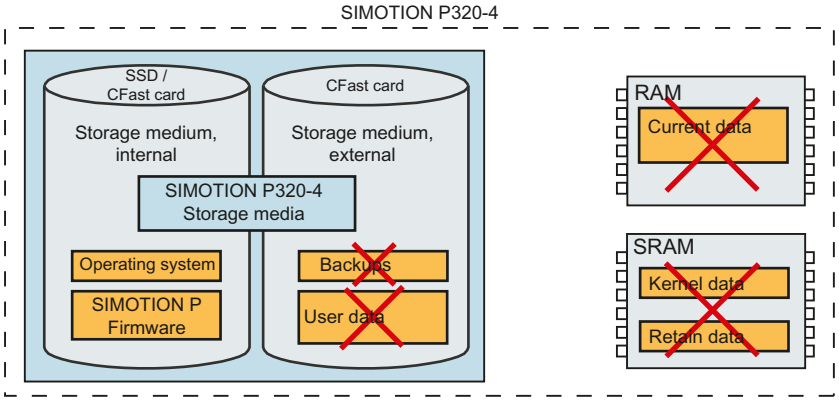


Figure 11-97 Restoring factory settings

Further information

on the SIMOTION P State - menus (Page 7913)

Overall reset using SIMOTION P State - operating mode switch setting MRES

A general reset deletes the non-volatile SIMOTION data and retain data.

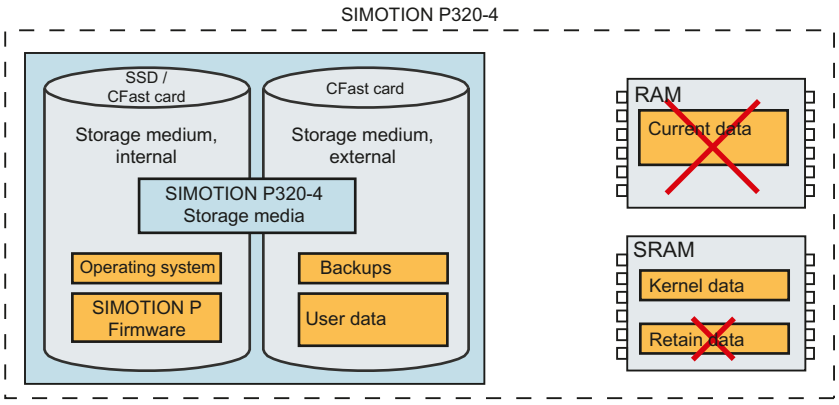


Figure 11-98 General reset

If a backup of the retain data (PMEMORY.XML) exists, this data will be accepted after the general reset. Thus, the user can perform a general reset to force the saved retain data to be restored.

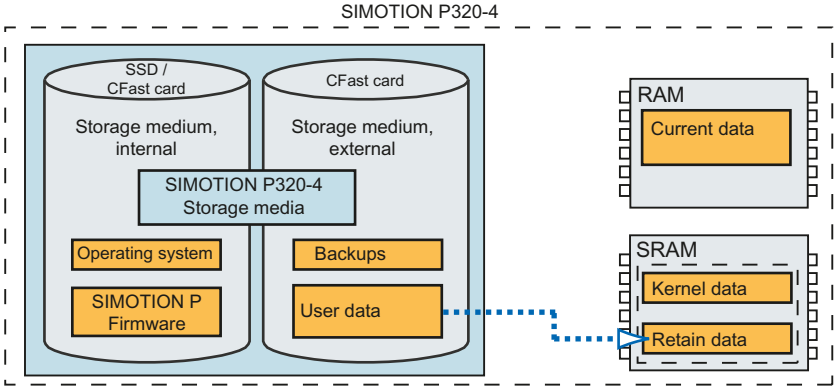


Figure 11-99 Restore retain data

Delete SRAM via SIMOTION P State - button Restart (Del. SRAM)

SIMOTION P320-4 will be restarted and the data in the SRAM will also be deleted.

The loading of a backup allows the SIMOTION P320-4 to be returned to a defined initial state. The backed up program variables and retain variables and thus the encoder values, homing data will be restored.

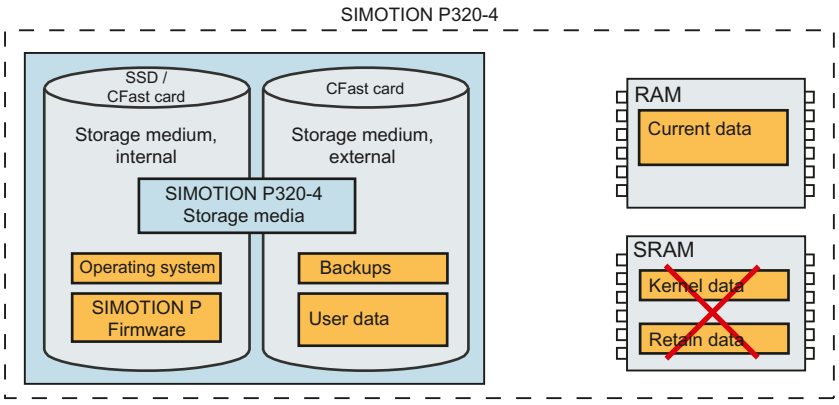


Figure 11-100 Delete SRAM - Restart (Del. SRAM)

If the SIMOTION SCOUT function **Target system > Copy RAM to ROM** was called previously, the saved project will be reloaded.

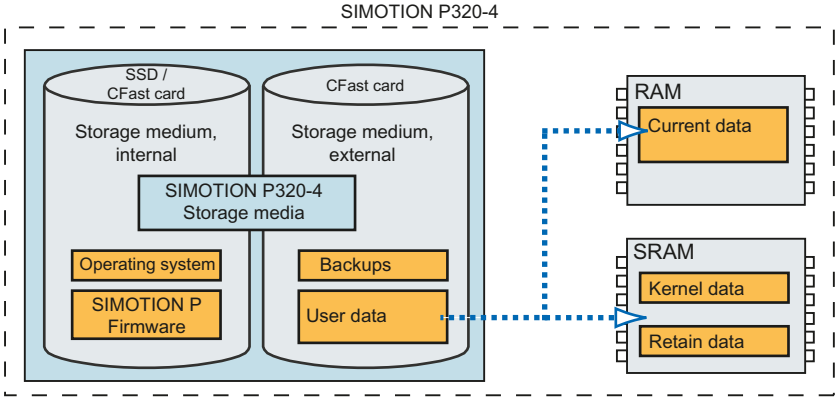


Figure 11-101 Load project after Restart (Del. SRAM)

Further information

on the SIMOTION P State buttons can be found in the Function elements (Page 7912) section.

SIMOTION P overall reset

SIMOTION P overall reset overview

Introduction

When a general reset is performed, the RAM on the SIMOTION P320-4 will be cleared along with any non-volatile data in the SRAM - with the exception of the communication configuration (e.g. baud rates and network addresses). For further information see Non-volatile data and SRAM handling (Page 7972).

Below is a list of the data that is either capable or not capable of withstanding a general reset. Data on the external CFast card is retained.

When must a general reset be performed on SIMOTION P320-4?

If you wish to undo changes in your user data (programs, configuration data, parameter assignments) in the **volatile data** area of the SIMOTION P320-4.

NOTICE
Differences when performing a general reset in connection with the RAM to ROM function
The following generally applies:
If **no** RAM to ROM was executed before the general reset, all programs and parameter assignments are considered as **data deleted on general reset**.
If data was backed up **with** RAM to ROM before the general reset, this data is considered as **data not deleted on general reset**.

Data deleted on general reset

The following data is deleted on the SIMOTION P320-4:

- Configuration data
- Programs
- Parameter assignments
- Technology packages
- Data from absolute encoders (absolute encoder adjustment, cyclic range overflows)

  Note

  Absolute encoder data is deleted during a general reset operation and must, therefore, be readjusted after the general reset.

- Retain variables
  Retain variables are variables that are defined as **non-volatile data** in the user program by the key word **Retain**.

Data not deleted on general reset

The following data is retained:

- Baud rate
- PROFINET IP address / name of the SIMOTION P320-4
- IP addresses
- Contents of the diagnostics buffer

- Data that was saved with the _savePersistentMemoryData(), _saveUnitDataSet(), _exportUnitDataSet() commands and with RAM to ROM.
  If the retain data is backed up (with _savePersistentMemoryData()), this data is backed up again to the non-volatile data after the general reset. Thus, the user can perform a general reset to force the saved retain data to be restored.
- Licenses

The technology packages and user data (configuration data, programs, parameter assignments) previously backed up to an external CFast card by means of the **Copy RAM to ROM** menu command are transferred to the **volatile data** area of the SIMOTION P320-4 after the general reset.
Thus, an existing configuration on the external CFast card is loaded to the SIMOTION device following the general reset.

Further information

Data storage concept overview (Page 7944)

Overall reset using the mode selector (on screen)

For SIMOTION P320-4, the overall reset using a simulated mode selector is performed using the SIMOTION P State application.

Note

MRES mode switch position

If you set **MRES** via the SIMOTION P State application, the SIMOTION P320-4 automatically goes into the **STOP** operating state.

Proceed as follows to initiate an overall reset:

1. Turn the mode selector to **MRES** and keep **MRES** pressed until the **STOP** LED stops flashing. When the LED permanently lights up yellow, the **STOP** operating state has been reached.

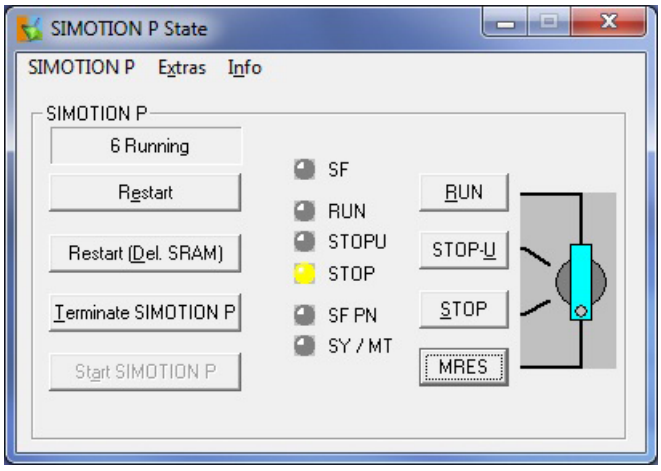


Figure 11-102 SIMOTION P State - STOP operating state

2. When you release **MRES**, the mode selector points to the **STOP** position again.

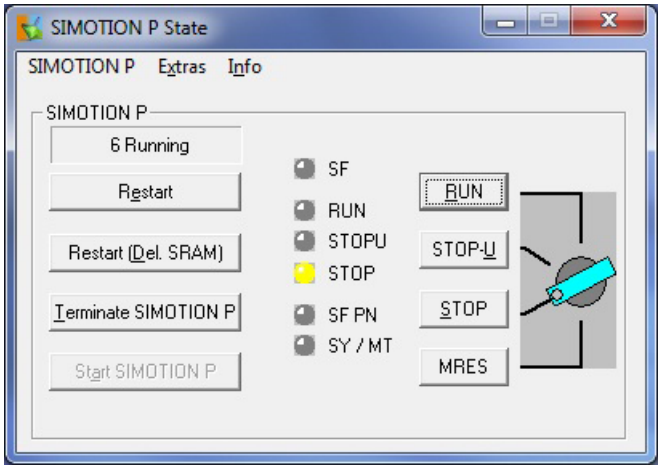


Figure 11-103 SIMOTION P State - Mode selector from position MRES to STOP

3. You must turn the switch back to the **MRES** position within three seconds.
   MRES is then displayed in the status.

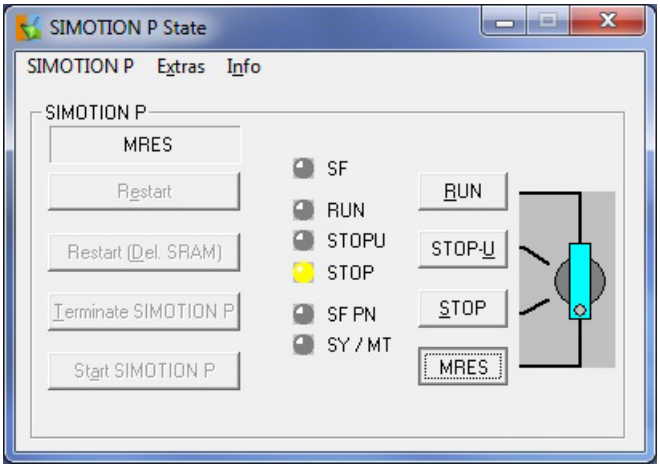


Figure 11-104 SIMOTION P State - status MRES

The mode selector then goes into the **STOP** position.

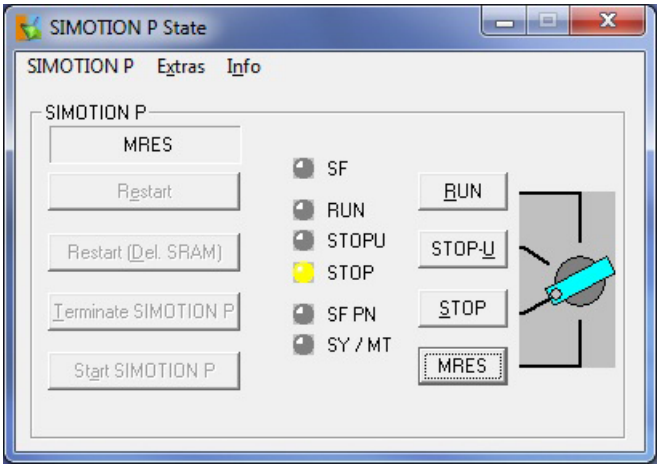


Figure 11-105 SIMOTION P State - status MRES, mode selector STOP

4. The SIMOTION P320-4 then restarts automatically.

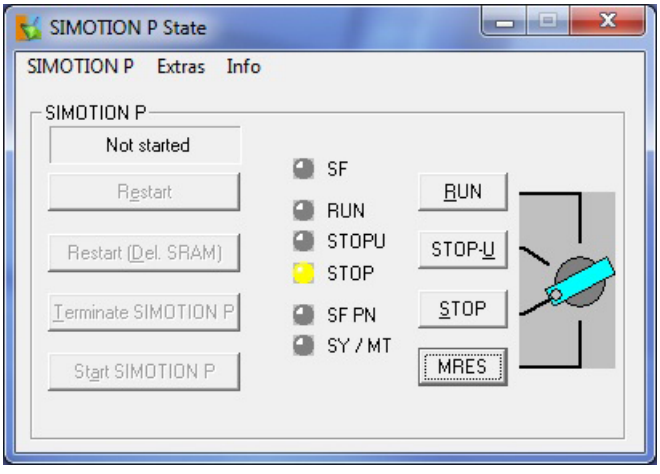


Figure 11-106 SIMOTION P State - status before the automatic restart

Deleting user data on the CFast card

Requirement

You can delete the user data with SIMOTION SCOUT. The SIMOTION P320-4 must be online. The user data in the RAM, the persistent data in the SRAM (except for the communication configuration), and the user data on the CFast card are deleted. You can thus continue to access the SIMOTION P320-4 with your PG/PC.

Procedure

To delete the user data:

1. In SIMOTION SCOUT, open the project you want to edit.
2. Go online with SIMOTION P320-4.
3. Select **Target system > Delete user data on card...**.

The data is deleted.

Switching off

Windows

To ensure the safe operation of the SIMOTION P320-4, Windows must be properly shut down before the SIMOTION P320-4 is switched off.

Note

Use the Windows start menu to correctly shut down the Windows system: **Start > Shut down**

If Windows is not shut down properly, the Windows installation may become damaged to the extent that it may no longer be possible to run the SIMOTION P320-4.

Note

Only applies to the SIMOTION P320-4 E

SIMOTION P320-4 E is delivered with deactivated EWF (Enhanced Write Filter).

After commissioning the SIMOTION P320-4 E, activate the EWF for drive C:, which ensures:

- An extended service life for the **internal** CFast card
- No data loss on the **internal** CFast card when switching the device off via the line side switch or through disconnection from the power supply

NOTICE
Do not protect external CFast cards with EWF Drive D: - the **external** CFast card - must not be protected with EWF as otherwise the functionality of SIMOTION cannot be assured.

NOTICE
Threat of data loss **Always** shut down SIMOTION correctly as otherwise there is a threat of data loss.

SIMOTION P320-4

If Windows is shut down properly, the SIMOTION P320-4 will also be shut down properly.

If the SIMOTION P320-4 is switched off before Windows is shut down correctly, the POWER FAIL functionality causes the following events to occur.

- SIMOTION P320-4 is stopped correctly
- The SIMOTION P retain variables are saved in the non-volatile memory of the SIMOTION P320-4.

When SIMOTION P320-4 powers up again, the backed-up user data will be available once more.

This means that the SIMOTION P320-4 is ready to run immediately without loss of data,

provided that the Windows installation was not damaged by the switch-off.

Behavior with Shut down, Logoff or Restart

If a Windows Shutdown, Logoff or Restart is initiated although the SIMOTION P320-4 is still in the **Running** state, the SIMOTION P320-4 is terminated automatically.

With large projects, a black screen may be displayed in Windows with the message **Waiting for SIMOTION P – Please wait - SIMOTION P is terminating**.

The termination of the SIMOTION P Runtime can vary depending on the size of the project.

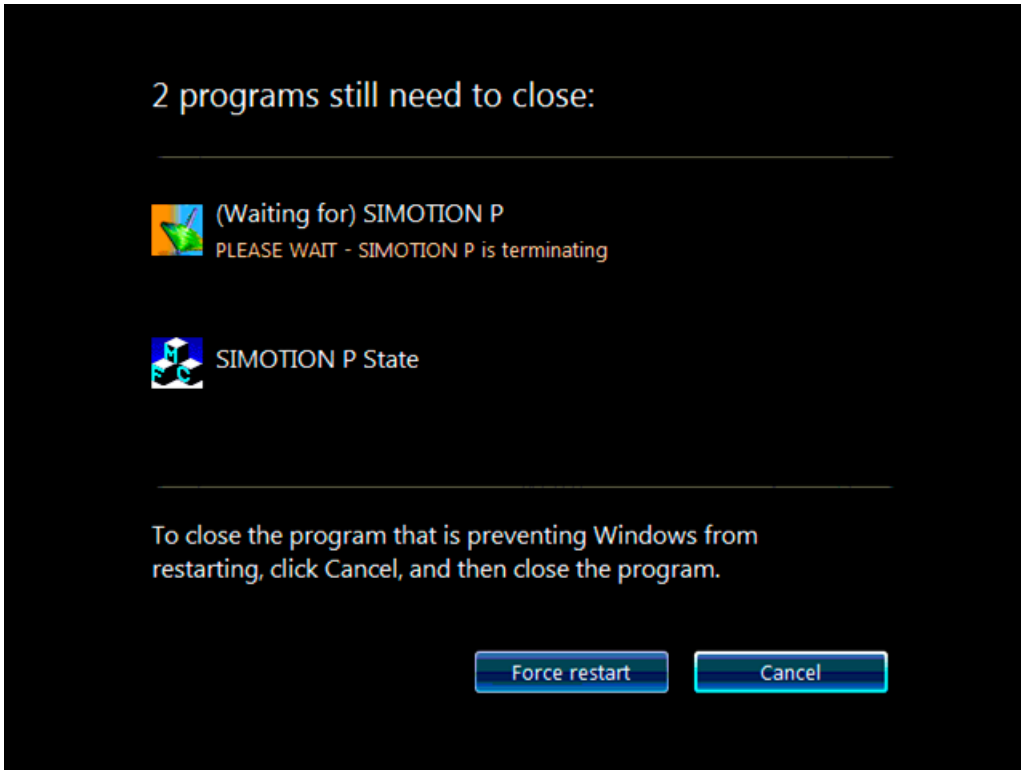


Figure 11-107 Black screen

11.1.1.11 Service and maintenance

Recording diagnostic data

Diagnostics data can be recorded during power-up and operation for the diagnostics and error analysis.

Saving diagnostic data during power-up

1. Terminate the SIMOTION P320-4 via the **Terminate SIMOTION P** button of the **SIMOTION P State** application.
2. Activate the diagnostics switch via the following option:
   SIMOTION P State, menu item **SIMOTION P > Set Diagnostic Switch**.
   The diagnostic data is recorded during power-up and the **SIMOTION P > Set Diagnostic Switch** switch automatically reset after writing the data.
3. After writing the diagnostic data, **SIMOTION P** must be terminated (RUN LED flashes continuously green).

Saving diagnostic data during operation

1. Make sure that the SIMOTION P320-4 has been started and powered up, e.g. with SIMOTION P State.
2. Activate the diagnostics switch via the following option:
   SIMOTION P State, menu item **SIMOTION P > Set Diagnostic Switch**

The diagnostic data is recorded during operation and the **SIMOTION P > Set Diagnostic Switch** switch automatically reset after writing the data.

Backing up diagnostic data

Following a fault on a SIMOTION device, diagnostic data (e.g. diagnostic buffer content, current content of website, etc.) can provide important information on the cause of the fault. The diagnostic data can be recorded during power-up or operation and backed up on the external CFast card by means of a simple operation as described below.

External CFast card

You can copy the backed-up data from the external CFast card.

Path: D:\USER\SIMOTION\HMI\SYSLOG\DIAG

SIMOTION IT / FTP

You can load the backed-up diagnostic data by means of SIMOTION IT or FTP:

You can find more detailed information at:

- SIMOTION IT Diagnostics and Configuration Diagnostics Manual
- SIMOTION online help

Evaluating diagnostic data

You can also use the diagnostic data for diagnostic purposes or forward it to Technical Support for evaluation purposes.

Further information

SIMOTION P State - menus (Page 7913)

Diagnostics via LED displays (Page 7975)

Single backups (Page 7950)

Restoring factory settings

Overview of restoring the factory settings

Note

When the factory settings are reset/restored, licenses are retained.

Resetting the SIMOTION P320-4 to the delivery status

The **SIMOTION P State** application is used for this function.

Procedure

Proceed as follows to restore the delivery status of SIMOTION P320-4:

1. First, terminate SIMOTION P320-4 via the **Terminate SIMOTION P** button of the SIMOTION P State application.
2. Then select **Extras** > **Reset to factory default** in the menu.

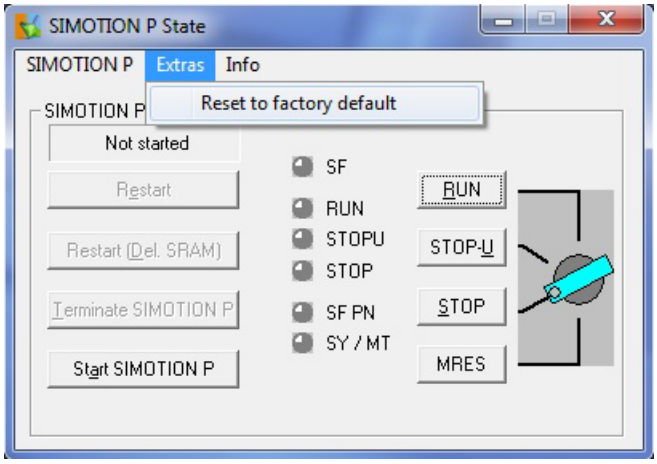


Figure 11-108 Restoring the delivery status

NOTICE
Note the consequences of user changes!
All data and settings for SIMOTION P320-4 changed by the user will be deleted!
The settings made using the **Station Configurator** will also be reset to the delivery status.

Reset SIMOTION P320-4 to the initial state

If an error occurred during parameter assignment, download, or when other settings were configured or if you are not sure whether you have specified all the information correctly, it may be useful or necessary to reset the SIMOTION P320-4 to its state prior to parameter assignment.

Requirements

You have saved an image of the memory medium on a remote computer/remote medium.

You can now use this image to reset the SIMOTION P320-4 to the factory settings. Note that the changes made on the SIMOTION P320-4 will be lost in the process.

NOTICE
Retain data
Once the image has been loaded, the user may have to delete the retain data.

Procedure

Use the **Restart (Del. SRAM)** function from the SIMOTION P State application to reset.

Further information

Function elements (SIMOTION P State) (Page 7912)

Upgrading runtime and firmware

As of the SIMOTION P320-4 Box, the current software version is always supplied. The previous version is available on the included restore DVD.

The upgrade to the latest hotfix (HF) can be performed via a setup.

Procedure

Possible types of upgrade:

- Upgrade with external DVD drive.
  First, the upgrade software must be copied to an external DVD or USB memory stick.
- Upgrade using a remote computer in the network (connect the SIMOTION P320-4 to the drive of the remote computer via the network)

Start the upgrade by calling the **Setup.exe** file.

See also

Further information on upgrading can be found in the following documentation:

- Updating SIMOTION devices
- SIMOTION online help

Installing/removing the back-up battery of the SIMOTION P320-4

Preventive maintenance

To maintain high system availability, we recommend the preventative exchange of those PC components that are subject to wear. The table below indicates the intervals for this exchange.

Component	Exchange interval:
CMOS backup battery	4 years

Replacing the backup battery

Note

The service life of the backup battery is approximately 5 - 8 years, depending on the environmental conditions.

To be noted before you replace the battery

NOTICE

Risk of damage!

The lithium battery may only be replaced with an identical battery or with a type recommended by the manufacturer (article number: A5E30314053)

⚠ WARNING

Risk of explosion and release of harmful substances!

Therefore, do not throw lithium batteries into an open fire, do not solder or open the cell body, do not short-circuit or reverse the polarity, do not heat to above 100° C, dispose of them in accordance with regulations, and protect them from direct exposure to sunlight, humidity, and condensation.

Note

A backup battery maintains the contents of the SRAM data on the SIMOTION P320-4 after the supply power is switched off.

Battery monitoring

The 3 V lithium battery to back up the SRAM and the clock module is monitored in two stages:

Table 11-10 Two-stage battery monitoring

Battery voltage	Message
2.7 V ± 0.1 V	Diagnostic buffer entry: Battery warning Battery warning threshold reached
2.25 V ± 0.25 V	Diagnostic buffer entry: Battery error Voltage below limit

In order to prevent data being lost, the battery should be replaced once the **Battery warning** entry appears in the diagnostics buffer or, at the very latest, the first time the **Battery error** entry appears in the diagnostics buffer.

Disposal

NOTICE
Dispose of batteries properly Batteries must be disposed of in accordance with local regulations.

Preparation

Note

The configuration data and contents of the SRAM in the device are buffered for at least 30 seconds when the battery is replaced.

Preparation – make a note of the BIOS settings

Note

Note specific BIOS setup settings if changes have been made.

Preparation – save SRAM data

As a result of replacing the battery (SRAM not backed up during this time), data loss may occur in the SRAM of the SIMOTION P320-4 if the battery replacement takes longer than 30 seconds.

1. To avoid a lengthy repeat of the commissioning procedure, ensure that a backup of the non-volatile data is available before changing the battery. This is performed, for example, automatically when Windows is shutdown correctly.
2. Shut down the SIMOTION PP320-4 or Windows correctly via:
   Windows taskbar: **Start > Shut down**
3. Isolate the device from the power supply and disconnect all connecting cables.

Note

If Windows was not shut down properly before backup battery replacement, loss of bit values in SRAM memory cells during battery replacement cannot be detected reliably.

SIMOTION P320-4 must then undergo an overall reset.

Tools

You will need a T10 screwdriver to open the battery compartment.

Replacing the battery

NOTICE
Switch off the device before replacing the battery The battery can also be exchanged when the device is operating. However, we recommend switching off the device beforehand.

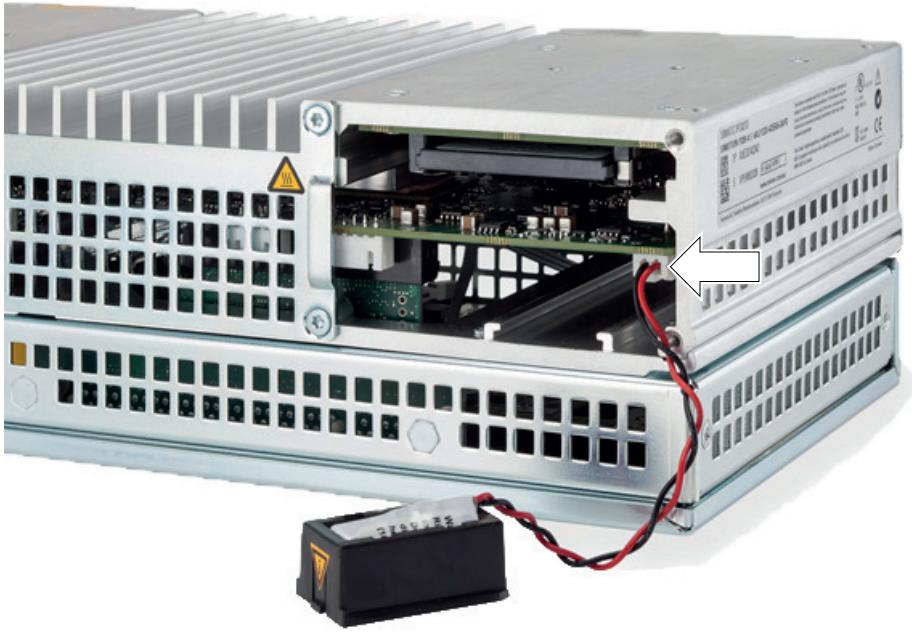
NOTICE
Time setting is lost If the battery change takes longer than 30 seconds, the time setting is deleted. The time on the device is no longer synchronized. Time-controlled programs no longer run or they run at the wrong time. This can damage the system. Set the device time again.

Procedure

Steps for replacing the battery:

1. Open the battery compartment.

2. Remove the battery holder.
3. Unplug the battery connector.



4. Connect the battery plug to the new battery.
5. Remove the old battery from the holder.
6. Secure the new battery in the holder.
7. Close the battery compartment.

Reconfiguring the BIOS Setup

The battery change does not take longer than 30 seconds with the device switched off, the BIOS settings are retained.

The SRAM and CMOS data of the SIMOTION P320-4 is certainly deleted only when the battery is disconnected for a longer period of time (more than 15 minutes) and in the switched-off state.

The BIOS Setup must be performed again if the data has been cleared. The CMOS data must be restored to the default values; the date, time, and any special settings must be set again.

Note

Exception:

If the CMOS data has been saved previously with a USER profile, only the date and the time may be set again. All other settings will be made automatically. This means that the default values may **not** be loaded in this special case.

Special power-up situations

Special startup situations overview

The special power-up situations are described in the following:

- Non-volatile data and SRAM handling (Page 7972)
- After new installation / update of the SIMOTION P Kernel (Page 7974)
- After importing a backup copy (Page 7974)
- After a power failure/POWER FAIL (Page 7974)

Requirements

You are familiar with the user storage concept of the non-volatile data on the SIMOTION P320-4.

Non-volatile data and SRAM handling

Non-volatile data, also called retain data, is retained when SIMOTION P320-4 suffers a power failure. The data is buffered with the battery in the static memory area (SRAM) on the SIMOTION P320-4.

SRAM handling

If the serial number of the SIMOTION P320-4 is known:

The serial number of the SIMOTION P320-4 corresponds to the one that was last saved on the SIMOTION P320-4.

SRAM image (storage medium) or SRAM backup (storage medium) is **OK**, if:

- The SIMOTION software release of the SRAM image/backup matches the installed software release.
- Windows has been shut down properly (the POWER FAIL mechanism of SIMOTION P does not suffice here).
- The checksum test on the SRAM image/backup was successful.
- The battery status was good when the SRAM image was backed up.

Table 11-11 SRAM handling if the serial number is known

SRAM "OK"?	SRAM image (storage medium) "OK"?	SRAM backup (storage medium) "OK"?	User data used	Note
Yes	Not applicable	Not applicable	Board [1]	Normal statup
No	Yes	Not applicable	IMAGE [2]	No message box or alarms
No	No	Yes	BACKUP [3]	No message box or alarms
No	No	No	Overall reset	Required

[1] The user data backed up in the SRAM of the SIMOTION P320-4 is used.

[2] The user data backed up in the SRAM image on the storage medium of the SIMOTION P320-4 is used.

[3] The user data backed up in the SRAM backup on the storage medium of the SIMOTION P320-4 is used.

If the serial number of the SIMOTION P320-4 is not known:

The serial number of the SIMOTION P320-4 does not correspond to the one that was last saved on the SIMOTION P320-4, for example, after the replacement of the SIMOTION P320-4.

SRAM image (storage medium) or SRAM backup (storage medium) is **OK**, if:

- The SIMOTION software release of the SRAM image/backup matches the installed software release.
- Windows has been shut down properly (the POWER FAIL mechanism of SIMOTION P does not suffice here).
- The checksum test on the SRAM image/backup was successful.
- The battery status was good when the SRAM image was backed up.

Table 11-12 SRAM handling if the serial number is unknown

SRAM "OK"?	SRAM image (storage medium) "OK"?	SRAM backup (storage medium) "OK"?	User data used	Note
Yes	Yes	Not applicable	Board [1] or IMAGE [2]	Query made
Yes	No	Yes	Board [1] or BACKUP [3]	Query made
Yes	No	No	Board [1]	No message box
No	Yes	Not applicable	IMAGE [2]	No message box

SRAM "OK"?	SRAM image (storage medium) "OK"?	SRAM backup (storage medium) "OK"?	User data used	Note
No	No	Yes	BACKUP [3]	No message box or alarms
No	No	No	Overall reset	Required

[1] The user data backed up in the SRAM of the SIMOTION P320-4 is used.

[2] The user data backed up in the SRAM image on the storage medium of the SIMOTION P320-4 is used.

[3] The user data backed up in the SRAM backup on the storage medium of the SIMOTION P320-4 is used.

After new installation / update of the SIMOTION P Kernel

If a compatible SIMOTION P Kernel is installed on an already commissioned SIMOTION P320-4, the user data backed up in the SRAM is retained.

However, the current software version of the SIMOTION PP320-4 must be identical to the software version with which the backed-up user data of the SRAM was created.

Now SIMOTION P320-4 is immediately ready to run again.

Note

If you do not want the backed up user data to be used again, you will have to recommission the SIMOTION P320-4.

Note

Restart of SIMOTION P320-4 required

After a new installation or update of the SIMOTION P Kernel, a restart of the SIMOTION P320-4 is generally required.

If the device is not restarted, the new SIMOTION P Kernel is not activated and you continue to work with the old firmware.

After restoring a backup copy

If a backup copy (storage medium image) of a commissioned SIMOTION P320-4 is reloaded, the backed-up user data stored in the SRAM of the SIMOTION P320-4 is used.

Now SIMOTION P320-4 is immediately ready to run again.

After a power failure/POWER FAIL

Case 1: SRAM saved

In the event of a power failure, the POWER FAIL detection feature integrated in the SIMOTION P320-4 will automatically cause the SIMOTION P firmware to save the user data in the SRAM of the SIMOTION P320-4. An SRAM image, however, cannot be created any more in this case.

When the power supply returns or when the system is booted again, the data is available again.

Now SIMOTION P320-4 is immediately ready to run again.

Note

Backup of the user data in the SRAM of the SIMOTION P320-4 in case of power failure is only guaranteed if the SIMOTION P320-4 is operated within its defined specifications.

Case 2: SRAM not saved

If the hardware of the SIMOTION P320-4 was operated outside of the defined specifications, the user data might not have been backed up in the SRAM of the SIMOTION P320-4 under certain circumstances.

11.1.1.12 Alarm, error and system messages

Diagnosis using the LEDs

Each LED can assume various states, such as ON, OFF, flashing. The status of the respective LED indicates the current status of the device.

Table 11-13 Diagnostics LEDs of the SIMOTION P320-4

LED display		Meaning	Meaning
SF	Red	**System fault**	This LED indicates that a fault has occurred on SIMOTION P320-4.
LED	On		An interrupt which can be acknowledged is present (alarm, message, note). (See Diagnostics Guide)
LED flashing	0.5 Hz		No license exists for technology/optional objects under license. Licensing must be performed to correct the fault, see License tab.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).
LED	Off		SIMOTION P320-4 works error-free.
RUN	Green	**SIMOTION P in RUN mode**	This LED indicates the state of the user program.
LED	On		The user program is running.
LED flashing	2 Hz		When the **RUN** operating state is selected, the LED will flash until the operating state is actually attained.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).

LED display		Meaning	Meaning
STOPU	Yellow	**SIMOTION P in the STOP user program**	This LED indicates that the technology packages are active. No user program is being executed.
LED flashing	2 Hz		When the **STOPU** operating state is selected, the LED will flash until the operating state is actually attained.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).
STOP	Yellow	**SIMOTION P in STOP mode**	This LED indicates that no user program is running.
LED	On		The technology packages are not active.
LED flashing	0.5 Hz		An overall reset request is indicated by slow flashing, see Overall reset with the mode selector (on the screen) (Page 7959).
	2 Hz		When the **STOP** operating state is selected, the LED will flash until the operating state is actually attained.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).
SF PN		**PROFINET bus error**	
LED	Red		This LED indicates a fault at the PROFINET interface.
LED	Off		PROFINET onboard interface is operating without error; the data exchange to all configured I/O devices is running.
LED	On		Bus fault
LED flashes	2 Hz		Failure of a connected I/O device. At least one of the assigned I/O devices cannot be addressed. Configuring error.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).
SY/MT		**Sync of PROFINET interface**	
LED	Yellow		This LED indicates whether the PROFINET interface has been synchronized.
LED	Off		The PROFINET interface has not yet synchronized with the send cycle clock of PROFINET IO with IRT, or PROFINET IO with IRT has not been configured.
LED	On		The PROFINET interface has synchronized to the send cycle clock of PROFINET IO with IRT.
LED flashes	2 Hz		A PROFINET firmware download is in progress.
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).

LED display		Meaning	Meaning
BUS1F **BUS2F**	X101 X102	**Status of IsoPROFIBUS interface**	
LED	Red		If this LED is red, it indicates a fault at the IsoPROFIBUS interface.
LED	On		Bus fault Parameter assignment error
LED	Green		If this LED is green, it indicates the bus status of the IsoPROFIBUS interface.
LED	On		Cyclic data exchange
LED flashes	2 Hz		Bus status "Clear"
	5 Hz		FAULT state SIMOTION software does not run in the FAULT state (all LEDs flash simultaneously).
LED	Off		No fault or no interface configured.

LED combinations

STOP

When the user program reaches a breakpoint, SIMOTION P320-4 then assumes the **HALT** state. A LED display indicates this state. The **HALT** state is also left when the breakpoint is left.

Table 11-14 LED combinations

LED	SF	RUN	STOPU	STOP	SF PN	SY/MT	BUSF1	BUSF2
STOP	x	0.5	1	1	x	x	x	x

x Not applicable

0.5 Flashes with a frequency of 0.5 Hz.

1 ON

DCP flashing

The DCP flashing function checks the correct assignment of a module and its interfaces.

The DCP flashing function is accessed via:

- Accessible nodes
- HW Config
- NetPro

Start the **Accessible nodes** function in SIMOTION SCOUT . Select a device and "Flashing" or "Stop flashing" is selected via the context menu.

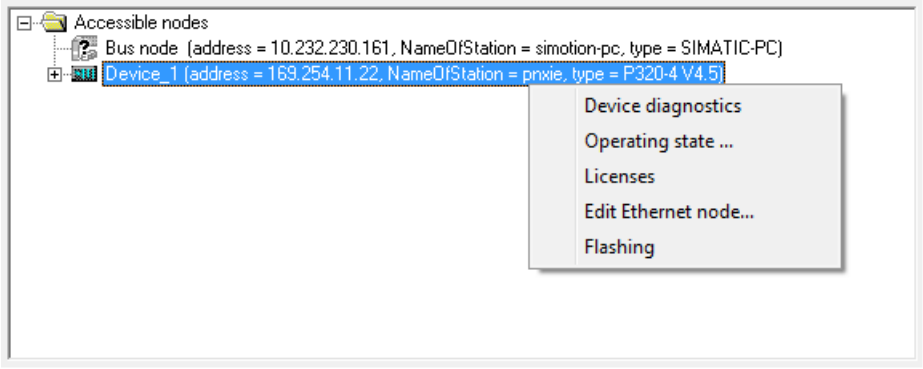


Figure 11-109 Accessible nodes

Alternatively, you can also activate the DCP flashing function in **HW Config** or in **NetPro** under the following path:

Target system > Ethernet > Edit Ethernet node... via the **Browse...** button, in the **Search Network** dialog box via the **Flashing** button of STEP 7.

The LED combination flashes as long as the dialog box is open and the **Flashing** function has not been deactivated.

If the dialog box is closed without deactivating the **Flashing** function, the previous state of the LED becomes active again.

Table 11-15 LED combinations

LED	SF	RUN	STOPU	STOP	SF PN	SY/MT	BUSF1	BUSF2
DCP flashing	x	2 (green/ yellow)	x	x	x	x	x	x

x Not applicable

1 ON

2 Flashes with a frequency of 2 Hz.

Note

You can carry out a detailed diagnosis using a PG/PC and the Engineering System.

FAULT

A fault has occurred to which the user program cannot respond.

The following actions may be required to rectify the fault:

- Power OFF/ON
- Recommissioning

Table 11-16 LED combinations

LED	SF	RUN	STOPU	STOP	SF PN	SY/MT	BUSF1	BUSF2
FAULT	5 (red)	5 (red)	5 (red)	5 (red)	5 (red)	5 (red)	5 (red)	5 (red)

5 Flashes with a frequency of 5 Hz.

Temperature alarm

For SIMOTION P320-4, two different temperatures are monitored in the device as regard to whether they reach their respective threshold value.

System variables

_cpudata.temperature

_cpudata.moduletemperature

Error response

An I/O alarm is activated.

Cause of error / troubleshooting

One of the two monitored temperatures has reached or exceeded its threshold.
The temperature must fall below the threshold value before the alarm is reset.

If the temperature alarm is issued, the user and/or the machine manufacturer (user program) must decide whether or not processing should be interrupted and SIMOTION P320-4 exited and switched off.

The user can program a customized response to this and allow the program to continue, if appropriate.

The default response is for the operating mode to switch to **STOP**.

Note

Further information on programming alarm responses can be found in the SIMOTION Runtime Basic Functions Function Manual.

Battery alarm

When the critical battery status is reached, the battery must be replaced as quickly as possible. Because the SIMOTION P320-4 often cannot be shut down at short notice during operation, it is important to perform a programmable response to this battery state.

For this case, the system variable

- persistentdatapowermonitoring.warningbatteryvoltagelevel1

can be used to catch the complete loss of the retain data if the battery cannot be replaced fast enough.

Users should periodically save their retain data (e.g. with the _savePersistentMemoryData function). This ensures that the operating state established after the next start up represents the most current status of the retain data possible.

⚠ WARNING

System variable persistentDataPowerMonitoring

If no response to this system variable has been programmed, there is the danger of the complete loss of retain data if a power failure occurs before the battery can be replaced.

In this case, only the retain data for the last back up, normally the last shut down of the SIMOTION P320-4, is available after the next power-up.

Power failure / POWER FAIL

In the event of a power failure, the consequences described below must be considered separately.

SIMOTION P Runtime

If a power failure occurs, the POWER FAIL functionality of the SIMOTION P Runtime will perform the following steps:

- SIMOTION P Runtime is shut down correctly
- The user data marked as retain is saved in the non-volatile memory of the SIMOTION P320-4

The backed-up user data is available again at the next power-up of the SIMOTION P Runtime. SIMOTION P Runtime is therefore ready for operation again without any data loss.

Note

Backup of the user data in the SRAM of the SIMOTION P320-4 in case of power failure is only guaranteed if the SIMOTION P320-4 is operated within its defined specifications.

The message **POWER FAIL** is displayed if the supply voltage for SIMOTION P320-4 falls below 20.4 V. In this case, check the voltages in the external power supply:

1. Check if the input voltage (mains voltage) for the external power supply has the rated value.
2. If so, check the output voltage of the external power supply. If the voltage is not within the limits for the supply voltage of the SIMOTION P320-4 (19.2 ... 28.8 VDC), then change the external power supply.

Windows

If the SIMOTION P320-4 is shut down during operation, the Windows installation may be damaged.

The consistency of Windows applications cannot be guaranteed during a power failure.

In these cases, we strongly recommend use of an EWF or a UPS system.

NOTICE
Do not protect external CFast cards with EWF Drive D: - the **external** CFast card - must not be protected with EWF as otherwise the functionality of SIMOTION cannot be assured.

Note

Only applies to the SIMOTION P320-4 E

SIMOTION P320-E is delivered with deactivated EWF (Enhanced Write Filter).

After the device is commissioned, activate the EWF for drive C: because this ensures:

- An extended service life for the **internal** CFast card
- No data loss on the **internal** CFast card when switching the device off via the line side switch or through disconnection from the power supply

NOTICE
Data loss on the external CFast card **Always** shut down SIMOTION correctly as otherwise there is a threat of data loss on the **external** CFast card. See also Section: Switching off (Page 7962).

PROFINET bus error occurs

The flashing red **SF PN** LED on the SIMOTION P320-4 indicates a problem of an I/O device on the bus or a configuring error.

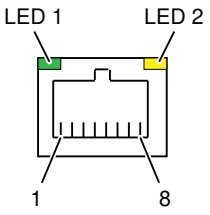
Information about further display modes of the LED can be found under SIMOTION P State application > LED display (Page 7910).

Error causes with flashing

- Failure of a connected I/O device
- At least one of the assigned I/O devices cannot be addressed
- Incorrect or no configuration.

PROFINET (LAN) X3 Port P1, P2, P3

The two LEDs on the respective PROFINET ports also provide information on the current connection status:

PROFINET interface		
		
LED 1	On	Lights up green: Link
	Off	No physical connection available. Troubleshooting: Check for loose connector, defective cable, etc.
LED 2	On	Lights up yellow: Activity
	Off	No logical connection, i.e. no Ethernet telegrams are exchanged. Troubleshooting: Check IP configurations, parameterize required router, etc.

Under-licensing

Error

The red system error LED (**SF**) flashes at 0.5 Hz.

Cause

No license exists for technology/optional objects under license.

Licensing must be performed to correct the error.

Note

Further information for the licensing of technology/option objects can be found in the SIMOTION SCOUT Configuration Manual.

Boot error messages

During power-up (the boot process), the BIOS first performs a **P**ower **O**n **S**elf **T**est (POST) and checks as to whether certain functional units of the PC are operating correctly. The boot sequence is immediately interrupted if critical errors occur.

If the POST does not return an error, the BIOS initializes and tests further functional units. In this startup phase, the graphics controller is initialized and any error messages are output to the screen.

The following lists the error messages from the system BIOS. For information on error messages output by the operating system or programs, refer to the corresponding manuals.

Error messages on the screen

Error message on the screen	Meaning / suggestions
Operating system not found	Possible causes: • No operating system present • Incorrect active boot partition • Wrong drive settings in SETUP

11.1.1.13 Troubleshooting/FAQs

Error correction

This section provides you with tips on how to locate and troubleshoot common problems.

FAQs / error correction	Possible cause	Possible remedy
The device is not operational.	There is no power supply to the device.	• Check whether the 24 VDC power supply is connected. See SIMOTION P320-4 E / P320-4 S Manual, Section Connecting the power supply (24 VDC). • Check whether the line side switch is activated. See SIMOTION P320-4 E / P320-4 S Manual, Section Operator controls.
SIMOTION P does not start after connecting the CFast card.	BIOS settings are not set to Hotplug Enabled.	You can find out how to check and correct the BIOS settings in Section Hotplug Enabled BIOS settings (Page 7985).
SIMOTION P does not start after a reboot.	The user is not logged in automatically.	Set the password for the AutoLogin (Automatically Log On). See Section SIMOTION P AutoLogin (Page 7885).
The monitor remains dark.	The monitor is switched off.	Switch on the monitor.
	The monitor is in "power-save" mode.	Press any key on the connected keyboard.
	The brightness button has been set to dark.	Set the monitor brightness button to obtain more light. For detailed information, refer to the monitor operating instructions.
	The power cord or the monitor cable is not connected.	• Check whether the power cord has been properly connected to the monitor and to the system unit or to the grounded shockproof outlet. • Check whether the monitor cable has been properly connected to the system unit and to the monitor.
		If the monitor screen still remains dark after you have performed these checks, please contact your technical support team.

FAQs / error correction	Possible cause	Possible remedy
The mouse pointer does not appear on the screen.	The mouse driver is not loaded.	Check whether the mouse driver is properly installed and present when you start the application program. For more detailed information on the mouse driver, refer to the documentation for the mouse or application programs.
	The mouse is not connected.	Check whether the mouse is properly connected to the system unit. If you use an adapter or extension on the mouse cable, also check the connectors.
		If the cursor still does not appear on the screen after you have performed these checks and measures, please contact your technical support team.
Error message on the screen: Operating system not found	• No operating system present. • Incorrect active boot partition. • Incorrect drive entries in the SETUP.	Check whether one of the specified causes is present:
Wrong time and/or date on the PC.	-	• Press <ESC> during the boot sequence to open BIOS Setup. Detailed information on the BIOS setup can be found in the section Starting Bios settings Hotplug Enabled - BIOS Setup (Page 7985). • Set the time and date in the setup menu.
Although the BIOS setting is OK, the time and date are still wrong.	The backup battery is empty.	Replace the backup battery. See Section Service and maintenances > Replacing the back-up battery (Page 7968). If the problem cannot be resolved by replacing the battery, then contact your technical support team.
UPS does not function.	The operating system does not have a suitable driver for the UPS.	Install a suitable driver; the correct driver can often be downloaded from the homepage of the device's manufacturer. See also Section Customer-specific software (Page 7909).
IP address assigned twice (e.g. through series commissioning).	Default IP address for Windows See Default IP addresses for Windows and SIMOTION P (Page 7931).	Assign a new IP address for Windows. The IP address must be different to the previous specified IP address and must not have been assigned already in the network.
The display in the BIOS for the PROFINET MAC address is incorrect (00:00:00:00:00:00).	-	The correct MAC addresses can be taken from the rating plate on the housing. See also Manual SIMOTION P320-4 E / P320-4 S Manual, Section Device identification data, MAC addresses.

Hotplug Enabled BIOS settings

Starting the BIOS setup

Proceed as follows:

1. Reset the SIMOTION P320-4 (warm or cold restart).
   Depending on the device version, the default settings may differ from the following figures.
2. After completing the startup test, you can start the program setup with BIOS.
   The following message is shown on the display:

Figure 11-110 SIEMENS Press Esc boot options

Press **Esc**.

3. After booting, the BIOS selection menu is displayed:

Figure 11-111 BIOS selection menu

4. The following buttons are available in the BIOS selection menu:

Button	Function
Continue	Exit BIOS menu, continue with the startup process
Boot Manager	Selection of the boot drive
Boot From File	Starting from a .EFI file
SCU	Device configurator (Setup Configuration Utility)
BIOS Update	BIOS update from USB stick
MEBx	Start the Intel Management Engine BIOS Extension

5. Select the **SCU** device configurator.

6. In the BIOS window, select the **Advanced** tab using the arrow keys and then select **IDE Configuration**.
   Press Enter to confirm the selection.

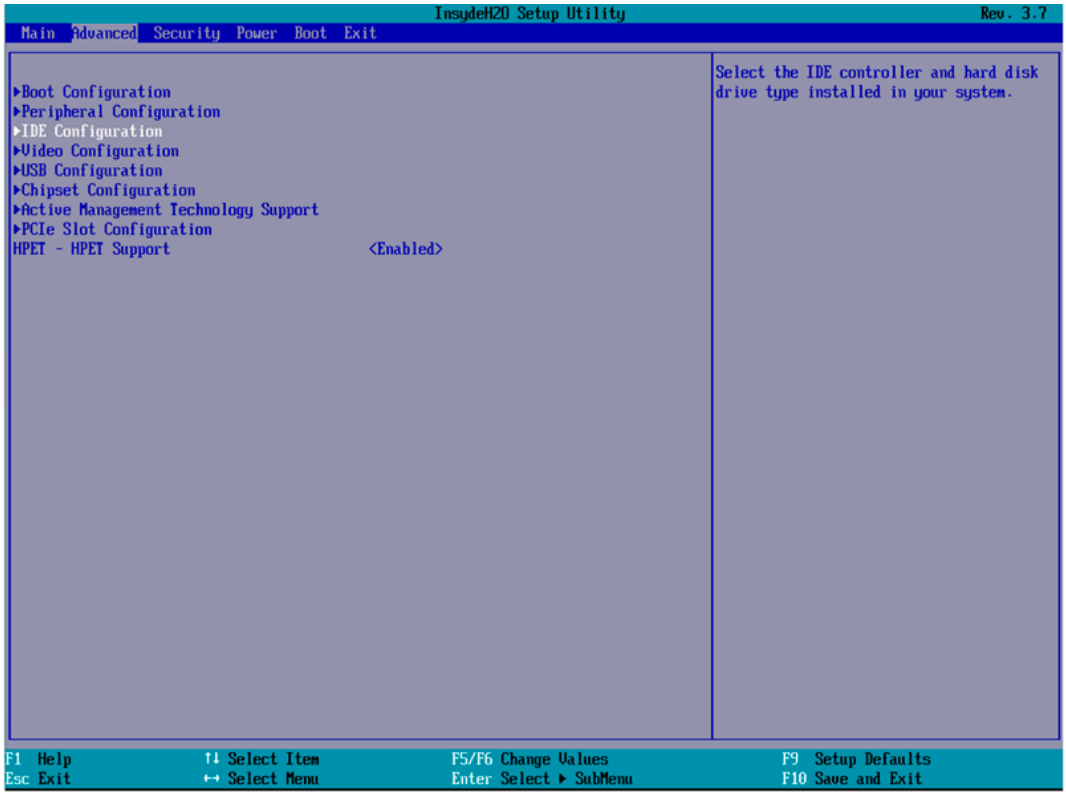


Figure 11-112 Advanced > IDE configuration

7. In the **IDE Configuration** window, select **SATA Port 2 HotPlug**.

8. Change the **Disabled** selection shown here to **Enabled**.

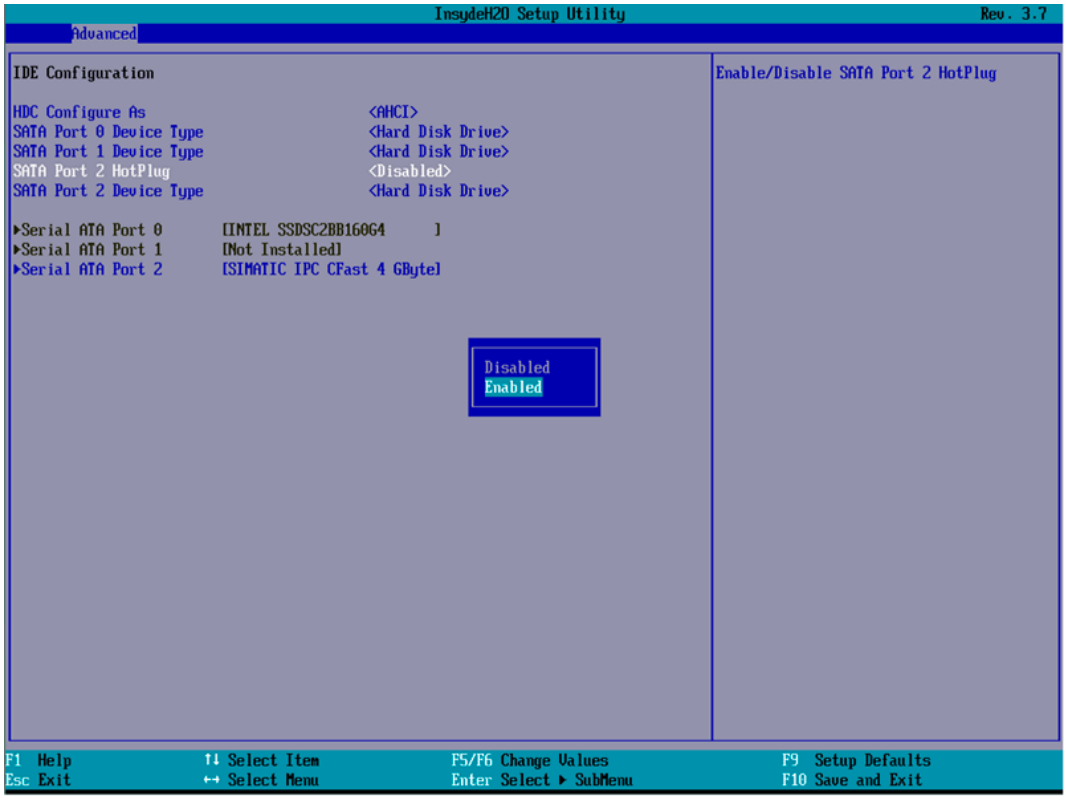


Figure 11-113 IDE Configuration > SATA Port 2 HotPlug

9. Press **F10** Save and Exit. The new entries are accepted.
10. Confirm the subsequent prompt to save the settings and exit the BIOS setup with **Yes**.

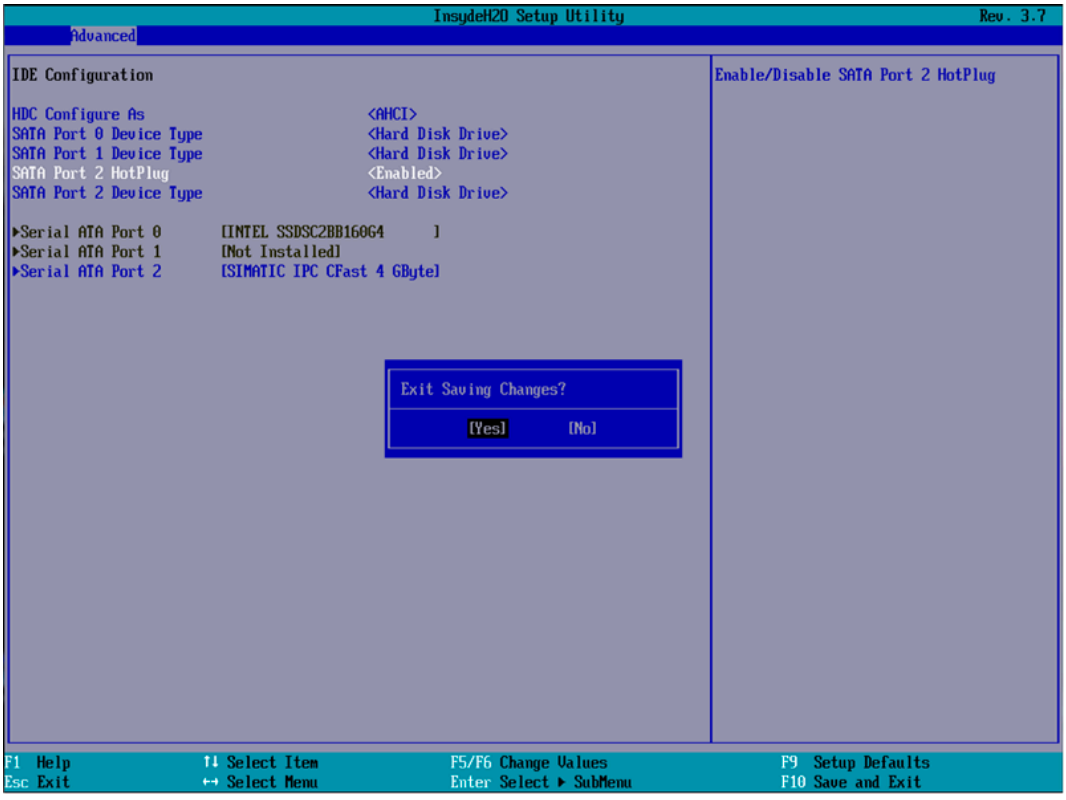


Figure 11-114 Confirming SATA Port 2 HotPlug > Enabled

11.1.1.14 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

Korea certification

	KC registration number: KCC-REM-S49-SIMOTION Note that this device complies with limit class A with regard to the emission of radio frequency interference. This device can be used in all areas except residential areas. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (http://support.automation.siemens.com/WW/view/en/10805446/134200).

11.1.1.15 ESD guidelines

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

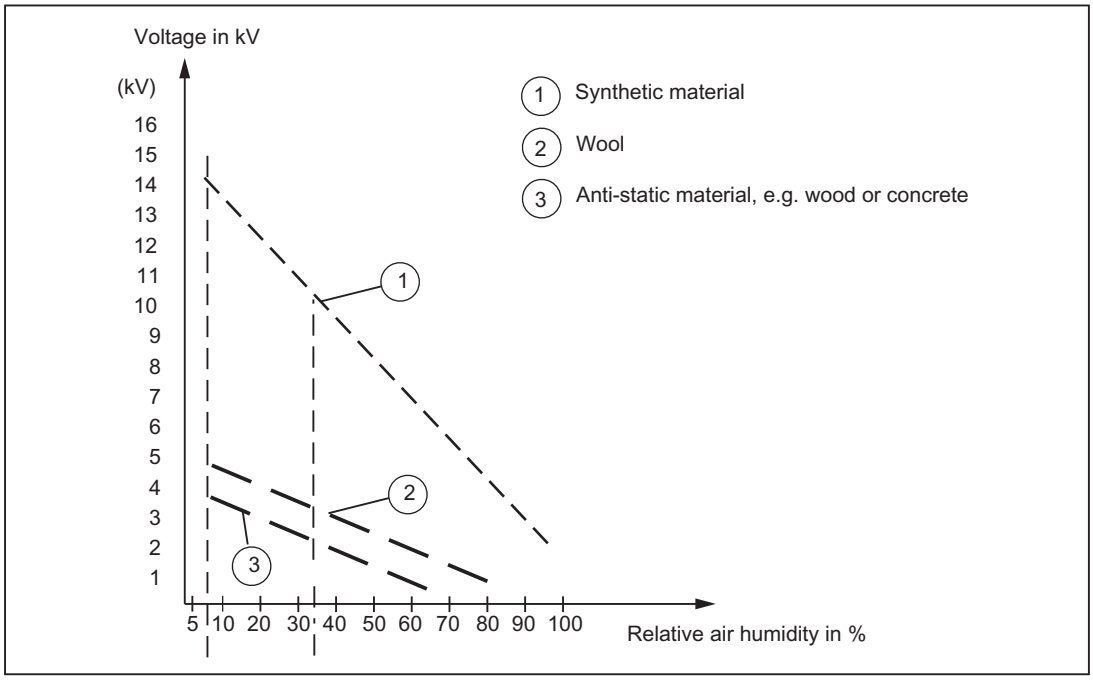


Figure 11-115 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

11.1.1.16 List of abbreviations

Commissioning Manual, Abbreviations

Abbreviation	Term	Meaning
CE	Communauté Européenne (CE-Symbol)	The product is in conformance with all applicable EC directives.
CMOS	Complementary Metal Oxide Semiconductor	Complementary metal oxide semiconductors
COM	Communications Port	Term for the serial interface
CPU	Central Processing Unit	CPU
CRT	Cathode Ray Tube	-
CSA	Canadian Standards Association	Canadian organization for tests and certifications according to own or binational standards (with UL / USA) standards
CTS	Clear To Send	Clear to send
DRAM	Dynamic Random Access Memory	Dynamic RAM
DC	Direct Current	DC current
DMA	Direct Memory Access	Direct memory access
DP	Distributed I/O	-
DQS	Deutsche Gesellschaft zur Zertifizierung von Management-Systemen	-
DSR	Data Set Ready	Ready for operation
DTR	Data Terminal Ready	Data terminal is ready
DVI-I	Digital Visual Interface	Digital display interface with digital and VGA signals
ESD	Components sensitive to electrostatic charge	-
EN	European standard	-
EWF	Enhanced Write Filter	Configurable write filter (SIMOTION P320-4 E)
FAQ	Frequently Asked Questions	FAQs

Abbreviation	Term	Meaning
GND	Ground	Chassis ground
HMI	Human Machine Interface	User interface
HW	Hardware	-
I/O	Input/Output	Data input/output on computers
IEC	International Electronical Commission	-
IP	Ingress Protection	Degree of protection
IRQ	Interrupt Request	Interrupt request
LAN	Local Area Network	Computer network that is limited to a local area.
LCD	Liquid Crystal Display	Liquid crystal display
LED	Light Emmitting Diode	Light emitting diode
MUI	Multilanguage User Interface	Language localization in Windows
NC	Not Connected	Not connected
OPC	OLE for Process Control	Standardized interface for industrial processes
PC	Personal Computer	-
PCI	Peripheral Component Interconnect	High-speed expansion bus
PE	Protective Earth	Protective conductor
PG	Programming device	-
POST	Power On Self Test	-
RAL	Restricted Access Location	Installation of device in operating facilities with restricted access, for example, a locked switchgear cabinet
RAM	Random Access Memory	RAM
ROM	Read-Only Memory	-
RTS	Reliable Transfer Service	Request to send
RxD	Receive Data	Data transfer signal
SDRAM	Synchronous DRAM	-
SELV	Safety Extra Low Voltage	Safety extra low voltage
SRAM	Static Random Access Memory	Static RAM
SVP	Serial No. of the device	-
SW	Software	-
TFT	Thin-Film-Transistor	Type of LCD flat-screen
TxD	Transmit Data	Data transfer signal
UL	Underwriters Laboratories Inc.	US organization for tests and certifications according to own or binational standards (with CSA / Canada) standards
UMA	Unified Memory Architecture	Video memory
USB	Universal Serial Bus	-
UPS	Uninterruptible power supply	-
VCC	-	Positive supply voltage of integrated circuits
VDE	Verein deutscher Elektrotechniker (Union of German Electrical Engineers)	-
VGA	Video Graphics Array	Video adapter which meets industrial standard
WD	Watchdog	Program monitoring with error detection and alarming.

11.2 Equipment Manual

11.2.1 SIMOTION P320-4 E / P320-4 S

Preface

Preface

This document is part of the **SIMOTION P documentation package**.

This documentation describes the SIMOTION P320-4 hardware platform which can be delivered in the SIMOTION P320-4 E and SIMOTION P320-4 S hardware versions:

- SIMOTION P320-4 E with the Windows Embedded Standard 7 32-bit operating system and real-time expansion for SIMOTION.
  Successor to SIMOTION P320-3.
- SIMOTION P320-4 E with the Windows 7 Ultimate 32-bit operating system and real-time expansion for SIMOTION.
  Successor to SIMOTION P350-3.

References

The following documents contain the descriptions for the SIMOTION P hardware platform:

- SIMOTION P320-4 E / P320-4 S, Manual, Edition 09/2016
- SIMOTION P320-4 E / P320-4 S, Commissioning and Hardware Installation Manual, Edition 09/2016

Validity range

This manual applies to the SIMOTION P320-4 E and SIMOTION P320-4 S devices as of product level SIMOTION V4.5.

Standards

The SIMOTION system was developed in accordance with ISO 9001 quality guidelines.

Chapters in this documentation

The following sections describe the purpose and the use of this documentation:

- Safety instructions
  Contains fundamental safety instructions for SIMOTION and specific safety instructions for SIMOTION P320-4.
- Description
  System overview and product description for SIMOTION P320-4. The communication versions are also described.
- Use planning
  Points to note in advance: upon delivery, the permitted installation positions, environmental and ambient conditions and electromagnetic compatibility.
- Interfaces
  Description of the interfaces and operator control and display elements for SIMOTION P320-4.
- Installation/mounting
  An overview of the installation of the SIMOTION P320-4 taking into account the mounting positions.
- Connecting
  This section provides general information and important notes that you must observe when connecting the SIMOTION P320-4.
- Troubleshooting/FAQs
  List of possible errors and their remedies.
- Technical specifications
  This section contains an overview of the technical data, which is listed in detail for the individual components.
- Dimension drawings
  In this section you can find dimension drawings and dimensions of the SIMOTION P320-4.
- Spare parts
  You can find information about the spare parts for the SIMOTION P320-4 here.
- Appendix
  In addition to the safety information, the annexes also contain information about the standards, approvals and EGB guideline.
- Index
  Alphabetical directory for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.2:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

Product disposal

SIMOTION P is an environmentally friendly product! It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Marking of the plastic materials as per ISO 11469.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The product is to be disposed of in accordance with national regulations.

The product described in this manual can be recycled owing to its low pollutant content. For environmentally friendly recycling and disposal of your old device, please contact a company certified for the disposal of electronic waste and dispose of the device in accordance with the regulations in your country.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner/index.asp

11.2.1.1 Safety instructions

Fundamental safety instructions

General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under (https://www.siemens.com/industrialsecurity).

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

Specific safety instructions for the SIMOTION P320-4

General safety instructions for the SIMOTION P320-4

⚠ WARNING

Life-threatening voltages are present with an open control cabinet

When you install the device in a control cabinet, some areas or components in the open control cabinet may be carrying life-threatening voltages.

If you touch these areas or components, you may be killed by electric shock.

Switch off the power supply to the cabinet before opening it.

System expansions

NOTICE

Damage through system expansions

Device and system expansions may be faulty and can affect the entire machine or plant.

The installation of expansions can damage the device, machine or plant.

Device and system expansions may violate safety rules and regulations regarding radio interference suppression.

If you install or exchange system expansions and damage your device, the warranty becomes void.

Note the following for system expansions:

- Only install system expansion devices designed for this device. Contact your technical support team or where you purchased your PC to find out which system expansion devices may safely be installed.
- Observe the information on electromagnetic compatibility (Page 8108).

NOTICE

"Open Type" UL508

Note that the device is classified as "Open Type" for use in the area of Industrial Control Equipment (UL508). Installation of the device in an enclosure according to UL508 is conditional for approval or operation according to UL508.

Battery and rechargeable battery

⚠ WARNING

Risk of explosion and release of harmful substances

Improper handling of lithium batteries can result in an explosion of the batteries.

Explosion of the batteries and the released pollutants can cause severe physical injury. Worn batteries jeopardize the function of the device.

Note the following when handling lithium batteries:

- Replace used batteries in good time, see the section "Replacing the backup battery" in the Commissioning and Hardware Installation Manual.
- Replace the lithium battery only with an identical battery or types recommended by the manufacturer (Article No.: A5E30314053).
- Do not throw lithium batteries into fire, do not solder on the cell body, do not recharge, do not open, do not short-circuit, do not reverse polarity, do not heat above 100°C and protect from direct sunlight, moisture and condensation.

High frequency radiation

NOTICE

Unintentional operating situations

High frequency radiation, e g. from a cellular phone, interferes with device functions and can result in malfunctioning of the device.

Persons are injured and the plant is damaged.

Avoid high-frequency radiation:

- Remove radiation sources from the environment of the device.
- Switch off radiating devices.
- Reduce the radio output of radiating devices.
- Observe the information on electromagnetic compatibility (Page 8108).

ESD Guideline

Electrostatic sensitive devices can be labeled with an appropriate symbol.

NOTICE

Electrostatic sensitive devices (ESD)

When you touch electrostatic sensitive components, you can destroy them through voltages that are far below the human perception threshold.

If you work with components that can be destroyed by electrostatic discharge, observe the ESD Guideline.

Further information

You can find more detailed information about the **EGB Guideline** in Annex B in the section with the same name.

Notes on use

⚠ WARNING

Hazards on an unprotected machine or plant

According to the results of a risk analysis, hazards can occur on an unprotected machine. The hazards can result in personal injury.

According to the risk analysis, the risk of personal injury can be avoided with the following measures:

- Additional protective devices on the machine or plant. With this, especially the programming, configuration and wiring of the inserted I/O modules have to be executed, in accordance with the necessary risk analysis identified safety performance (SIL, PL or Cat.).
- The correct use of the device has to be verified with a function test on the system. This test can detect programming, configuration and wiring errors.
- Documentation of the test results that you can enter in the relevant safety records when required.

NOTICE

Ambient conditions

Ambient conditions for which the device is not suitable can cause faults or damage the device.

Note the following:

- Operate the device only in closed rooms. Failure to comply nullifies the warranty.
- Operate the device only in accordance with the ambient conditions specified in the technical specifications.
- Protect the device against dust, moisture and heat.
- Do not expose the device to direct sunlight or other strong sources of light.
- Without additional measures, such as a supply of clean air, the device may not be used in locations with harsh operating conditions caused by acidic vapors or gases.
- Observe the permissible mounting positions of the device.
- Do not obstruct the venting slots of the device.

Note

Use in an industrial environment without additional protective measures

This device was designed for use in a normal industrial environment according to IEC 60721-3-3.

11.2.1.2 Industrial security

Security concept for SIMOTION P320-4

Security

Note

Observe the general security information in this documentation for Industrial security (Page 957).

Regular change of the Windows password

Note

For security reasons, the Windows password should be changed regularly.

It is essential that the AutoLogin is also adapted for this purpose.

You can find instructions in the following sections:

- Changing the Windows user password (Page 8020)
- AutoLogin for SIMOTION P (Page 8024).

Unlocking Windows

Note

Windows locked

Windows may be accidentally locked, e.g. through shortcut key Windows + L.

If you do not know the password for the SIMOTION P320-4, please contact the Siemens Industry Online Support (https://support.industry.siemens.com/cs/?lc=en-DE).

Windows firewall

Note

Windows firewall

To allows the Windows IP to be accessed externally, this must be set as an exception in the Windows firewall **File and Printer Sharing**. Otherwise, there is only limited access to Windows from outside.

Security and networks

Note

You will find information on the security of networks in Section General security measures (Page 8011).

Remote desktop connection

Note that for the SIMOTION P320-4 with Headless operation, you require a user name and password for a remote desktop connection. Per default, the remote desktop connection is already set up.

Note

Deactivating the remote desktop connection

If you do **not** use the remote desktop connection, it must be deactivated for security reasons.

See Section Deactivating the remote desktop connection (Page 8030)

SIMOTION IT

Note

Security concept

Note the security concept of HTTP/S, FTP and Telnet access on the Web server when working with SIMOTION IT.

You will find information in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual or the SIMOTION online help in Section Security concept.

Note

User administration

Note the information on the user administration when working with SIMOTION IT.

You will find information in the SIMOTION IT Diagnostics and Configuration Diagnostics Manual or the SIMOTION online help in Section User administration.

Information on industrial security

The following sections are taken from the Motion Control Industrial Security Configuration Manual:

- Why is industrial security so important? (Page 8009)
- General security measures (Page 8011)
- Product-specific measures (virus scanners) (Page 8019)

You can view the entire document in the Industry Online Support (https://support.industry.siemens.com/cs/ww/en/view/108862708).

Why is industrial security so important?

The topic of data security and access protection (security) is becoming more and more important in industrial environments. The progressive networking of entire industrial plants, the vertical integration and networking of the individual levels of a company, and new technologies, such as remote maintenance and remote access, are leading to increased requirements for protecting industrial plants.

The threats are diverse and the consequences far-reaching.

Possible threats:

- Espionage of data, recipes, etc.
- Sabotage of production plants
- System stoppage, e.g. due to virus infection and malware

- Manipulation of data or application software
- Unauthorized use of system functions

Possible effects of a security incident

- Loss of intellectual property
- Loss of production or reduced product quality
- Company image and economic damage
- Catastrophic environmental influences
- Danger to persons and machines

Trends in the IT sector

Overview

There are many new trends which affect industrial security:

- Cloud computing in general
  The number of network connections across the world is constantly increasing. This enables innovations such as cloud computing and the applications that go hand in hand with it. In conjunction with cloud computing, there has been a massive increase in the number of mobile devices, such as mobile phones and tablet PCs.
- Wireless technology
  On the other hand, the increasing use of mobile devices has only become possible thanks to the ubiquitous availability of mobile networks. Wireless LAN is also becoming increasingly available.
- Smart Grid
  Networking is not only limited to data networks, it also influences our energy infrastructure.
- Worldwide remote access to plants, machines and mobile applications
- The "Internet of things"
  Millions of electronic devices are becoming network-capable and are communicating via the Internet, such as onboard computers in cars, which send warranty information to dealers, or water meter sensors that transmit water consumption data to municipal water suppliers via radio.

However, in order for everything from cloud computing to sensors to work without service disruptions, you need reliable network infrastructures that are well protected against attacks from malware and hackers.

Possible corporate security holes

Possible security holes or weak points

The security chain of a company is only as strong as its weakest link. Security holes can exist at numerous points. The following list gives only a few examples:

- Employees
- Production plants
- Network infrastructure
- Data centers
- PC workstations
- Laptops
- Tablet PCs
- Printers
- Smartphones
- Portable storage media
- Guidelines and regulations

For this reason, a holistic approach is required to deal with the issue of security. Coordinated guidelines and regulations are required that cover all areas: Devices, systems, processes and employees.

General security measures

Overview

In the following section you will learn about the general security measures you can take in order to protect your system from threats. All of the measures are recommended.

Additional specific security measures for SINUMERIK, SIMOTION and SINAMICS products can be found in Section Product-specific security measures (Page 8019).

Basically, the measures should be coordinated with one another and correspond to the ring-shaped principle of the "Defense in Depth" strategy. The measures are structured according to the "onion" principle and each measure forms an additional protective layer around the core: the production plant.

Figure 11-116 Defense in depth strategy

- **Plant security**
  Plant security represents the outermost protective ring. Plant security includes comprehensive physical security measures, e.g. entry checks, which should be closely coordinated with protective measures for IT security.
- **Network security**
  The measures, grouped under the keyword "Network security", form the core of the protective measures. This refers to the segmentation of the plant network with limited and secure communication between subnetworks ("secure islands") and the interface check with the use of firewalls.
- **System integrity**
  "System integrity" represents the combination two major measures. PC-based systems and the control level must be protected against attacks. Steps include the following measures:
  - User authentication for machine or plant operators with individual authorization levels
  - Integrated access protection mechanisms in the automation components to prevent unauthorized changes via the engineering system or during maintenance
  - The use of antivirus and whitelisting software to protect PC systems against malware
  - Maintenance and update processes to keep the automation systems up-to-date (e.g. patch management, firmware updates, etc.)

Plant security

Physical protection of critical production areas

Unauthorized persons may be able to enter the production site/building and damage or alter production equipment as a result of gaps in a company's physical security. Confidential information can also be lost. This can be prevented if both the company's site and the production areas are protected accordingly.

Company security

The company's physical security can be ensured via the following measures:

- Closed off and monitored company premises
- Entry control, keys / card readers and/or security personnel
- Escorting of external personnel by company employees

Physical production security

The physical security of a production location can also be ensured via the following measures:

- Separate access control for production areas.
- Installation of critical components in securely lockable cabinets / switching rooms including monitoring and alarm signaling options
- Prohibited production areas with restricted access rights
- Configuration of the radio field to restrict the WLAN range so that it is not available outside the defined areas (e.g. factory building).
- Guidelines that prevent the use of third-party data storage media (e.g. USB sticks) and IT devices (e.g. notebooks) classified as insecure on the control.

Further information

Further information on integrated security solutions can be found on the Surveillance page (http://www.buildingtechnologies.siemens.com/bt/global/en/security-solution/Pages/security-solution.aspx).

Network security

Network segmentation

Separation between production and office networks

One important protective measure for your control is the strict separation of the production networks and the other company networks. This separation creates protection zones for your production networks.

Note

The products – drives, controllers, commissioning tools (e.g. STARTER or Startdrive) – described in this manual must only be operated in protection zones.

Separation by means of a firewall system

In the simplest scenario, separation is achieved by means of an individual firewall system which controls and regulates communication between networks.

Separation via a DMZ network

In the more secure version, the coupling is established via a separate DMZ network. In this case, direct communication between the production network and the company network is completely prevented by firewalls and only takes place indirectly via servers in the DMZ network.

Note

The production networks should also be divided into separate automation cells in order to protect critical communication mechanisms.

General security measures

Observe the general security measures even within protection zones, for example:

- Virus scanners (Page 8017)
- Reduction of attack points (Page 8017)

Network segmentation with SCALANCE S

Siemens provides SCALANCE S security modules to meet network protection and network segmentation requirements.

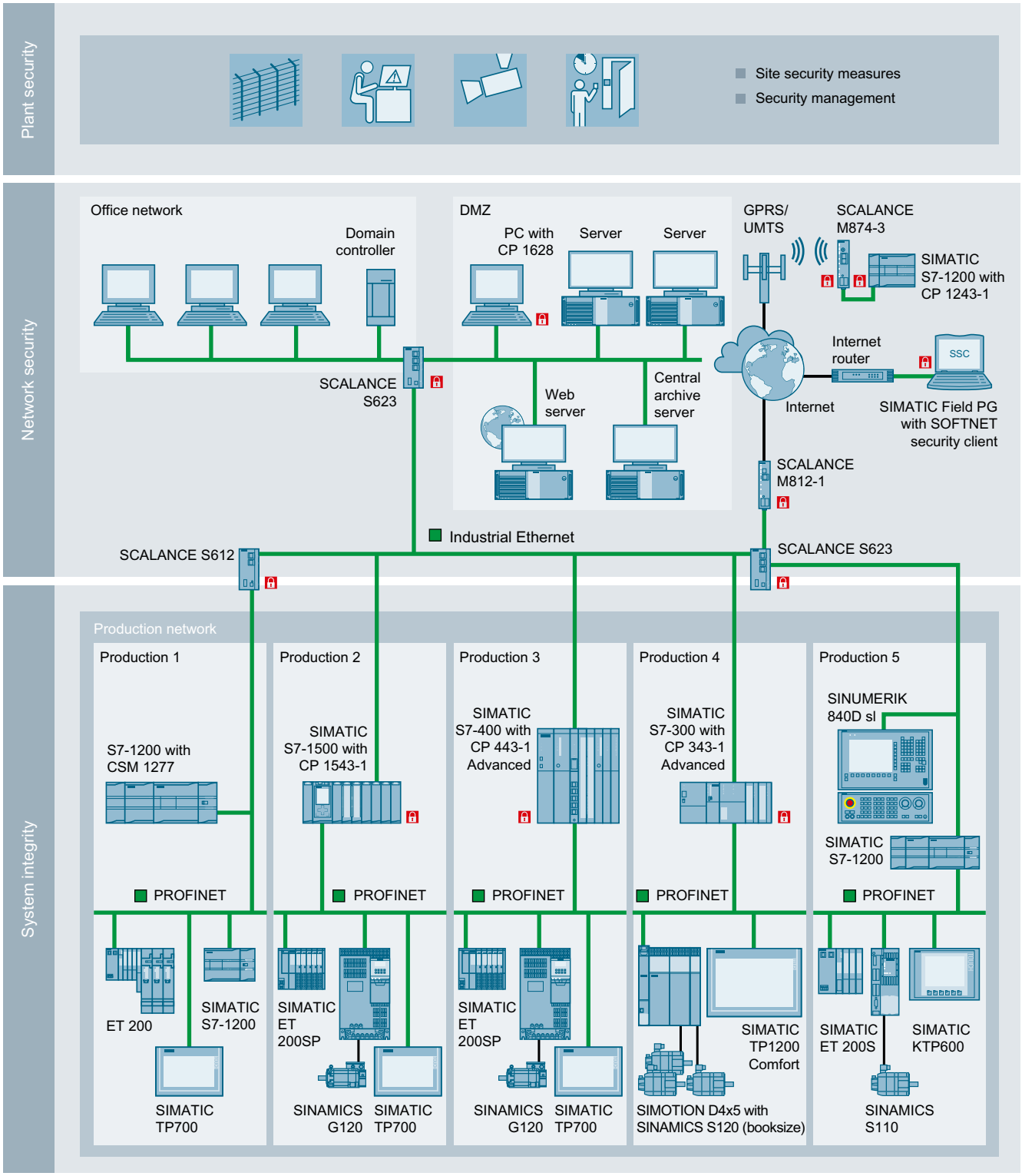
SCALANCE S security modules

SCALANCE S security modules with Security Integrated provide:

- Stateful inspection firewall
  In order to implement user-specific control and logging, firewall rules can also be specified that only apply to certain users.
- VPN via IPsec (data encryption and authentication)
  This establishes a secure tunnel between authenticated users whose data cannot be intercepted or manipulated. The most important aspect is the protection against external access via the Internet.
- NAT/NATP (address translation)
- Router functionality (PPPoE, DDNS) for broadband Internet access (DSL, cable)
- S623 with additional VPN port (DMZ) enables the secure connection of an additional network for service and remote maintenance purposes. S623 also permits the secure, redundant connection of subordinate networks by means of routers and firewall redundancy.

Principle

This application example shows cell segmentation by several SCALANCE S modules, each of which is upstream of the automation cell. The data traffic to and from the devices within automation cells can be filtered and controlled with the SCALANCE S firewall. If required, the traffic between the cells can be encrypted and authenticated. Secure channels and client access from the PCs to the cells can be established via SOFTNET Security Client, VPN client software for PCs.


Plant security
Site security measures
Security management
Network security
Office network
Domain controller
DMZ
PC with CP 1628
Server
Server
GPRS/ UMTS
SCALANCE M874-3
SIMATIC S7-1200 with CP 1243-1
Internet router
SSC
Internet
SIMATIC Field PG with SOFTNET security client
SCALANCE S623
Web server
Central archive server
SCALANCE M812-1
Industrial Ethernet
SCALANCE S612
SCALANCE S623
System integrity
Production network
Production 1
S7-1200 with CSM 1277
PROFINET
ET 200
SIMATIC S7-1200
SIMATIC TP700
Production 2
SIMATIC S7-1500 with CP 1543-1
PROFINET
SIMATIC ET 200SP
SINAMICS G120
SIMATIC TP700
Production 3
SIMATIC S7-400 with CP 443-1 Advanced
PROFINET
SIMATIC ET 200SP
SINAMICS G120
SIMATIC TP700
Production 4
SIMATIC S7-300 with CP 343-1 Advanced
PROFINET
SIMATIC TP1200 Comfort
SIMOTION D4x5 with SINAMICS S120 (booksize)
Production 5
SINUMERIK 840D sl
SIMATIC S7-1200
PROFINET
SIMATIC ET 200S
SIMATIC KTP600
SINAMICS S110

System integrity

System hardening

Reduction of attack points

Network services and ports

Activated services represent a risk. To minimize the risk, only the necessary services for all of the automation components should be activated. Ensure that all activated services are taken into account (especially web servers, FTP, remote maintenance, etc.) in the security concept.

A description of the ports used can be found in the Manuals and Function Manuals of the respective products.

User accounts

Any active user account allows access to the system is thus a potential risk. Therefore, take the following security measures:

- Reduction of configured/activated user accounts to the actually needed minimum
- Use of secure access data for existing accounts
- Regular checks, especially of the locally configured user accounts
- Regular change of passwords

Passwords

NOTICE
Changing default passwords The misuse of passwords can also represent a considerable security risk. We recommend that default passwords be changed during the commissioning and changed at regularly defined intervals as required.

Virus scanner

The use of a virus scanner must not impact the production operations of a plant. As the last consequence, this will lead to even a virus-infected computer not being permitted to immediately shut down if this would cause the control of the production process to be lost.

In order to be used on industrial control components, a virus scanner should therefore meet the following requirements:

Virus scanner requirements

- If a local firewall that has been adapted to the production operations is used, it must be possible to install the virus scanner without its own firewall.
- The virus scan clients can be divided into (product- and task-specific) groups and configured separately.
- It must be possible to deactivate the automatic distribution of the virus signatures and other updates.
- It must be possible to carry out the distribution of the virus signatures and updates manually and in groups.
- It must be possible to conduct a file scan and system scan manually and in groups.
- For the virus detection scenario, a message can be configured without a file action such as "Delete", "Clean", etc. being automatically carried out.
- It must be possible to log all of the messages on the virus scan server.
- On a virus scan client, it must be possible to suppress the local message window because it could obscure important messages from the production process.

Note

Installation of software

The installation of software is often a process which represents a serious and complicated change to the respective system. The storage location of the files to be installed must always be free of viruses (e.g. a file server with its own virus scanner or DVD checked for viruses).

Patch management

Microsoft security updates

The **WSUS** (Windows Server Update Service) system functionality provided by Microsoft is available for current Windows systems. WSUS supports administrators by providing Microsoft updates in large local networks. WSUS automatically downloads update packages from the Internet (Microsoft Update) and offers them to the Windows clients for installation.

The fully automatic update process ensures that Microsoft security updates are always available on Siemens clients.

Product-specific measures

Virus scanners, Windows security patches, SIMOTION P

General information on virus scanners

Once an industrial PC system is connected to the Internet, either directly or via an internal company network, there is a danger that it can become infected with a virus. However, malicious software is not only able to reach the system via the Intranet/Internet, but also, for example, via a removable storage device (such as a USB memory stick) attached to the system for backing up data.

SIMOTION P320-4 virus scanners

A virus scanner that runs on Microsoft Windows, as used in office or home computers, has a deep impact on a system's processes. There are, for example, processes such as real-time scans or regular system scans. Such interventions can cause performance issues for the system, and as a result, for the SIMOTION Runtime software. Although the SIMOTION Runtime software runs in a real-time environment, it still depends on the available system resources.

Note

Because of the resulting performance impairments, the installation and use of a standard virus scanner on a SIMOTION P320-4 during system runtime does not make sense and is not permitted.

Using a virus scanner

As a standard virus scanner cannot be used for SIMOTION P320-4, an alternative procedure is followed. The virus scanner is installed to a separately bootable Windows PE operating system. It is started, for example, from a CD or a USB storage device and then performs a virus scan.

Note

FAQ in Industry Online Support

More information on using a virus scanner on a SIMOTION P320-4 can be found in the FAQ "How can a virus scanner be used on a SIMOTION P320-4?" (https://support.industry.siemens.com/cs/ww/en/view/59381507), which is available as a download from the Industry Online Support portal.

Windows security patches

A brief test is performed when a new SIMOTION version is released. During this brief test, a check is performed to establish whether the installation of the security update has affected any basic functions.

Changing the Windows user password

Permitting the Windows password to be changed

Before the Windows password can be changed, it must be permitted for the user.

If the following procedure is not observed, an error message is issued and the Windows password cannot be changed.

Note the following procedure:

1. Select the entry point **Control Panel** > **All Control Panel Items** > **Administrative Tools** > **Computer Management**.
   Double-click **Computer Management**.

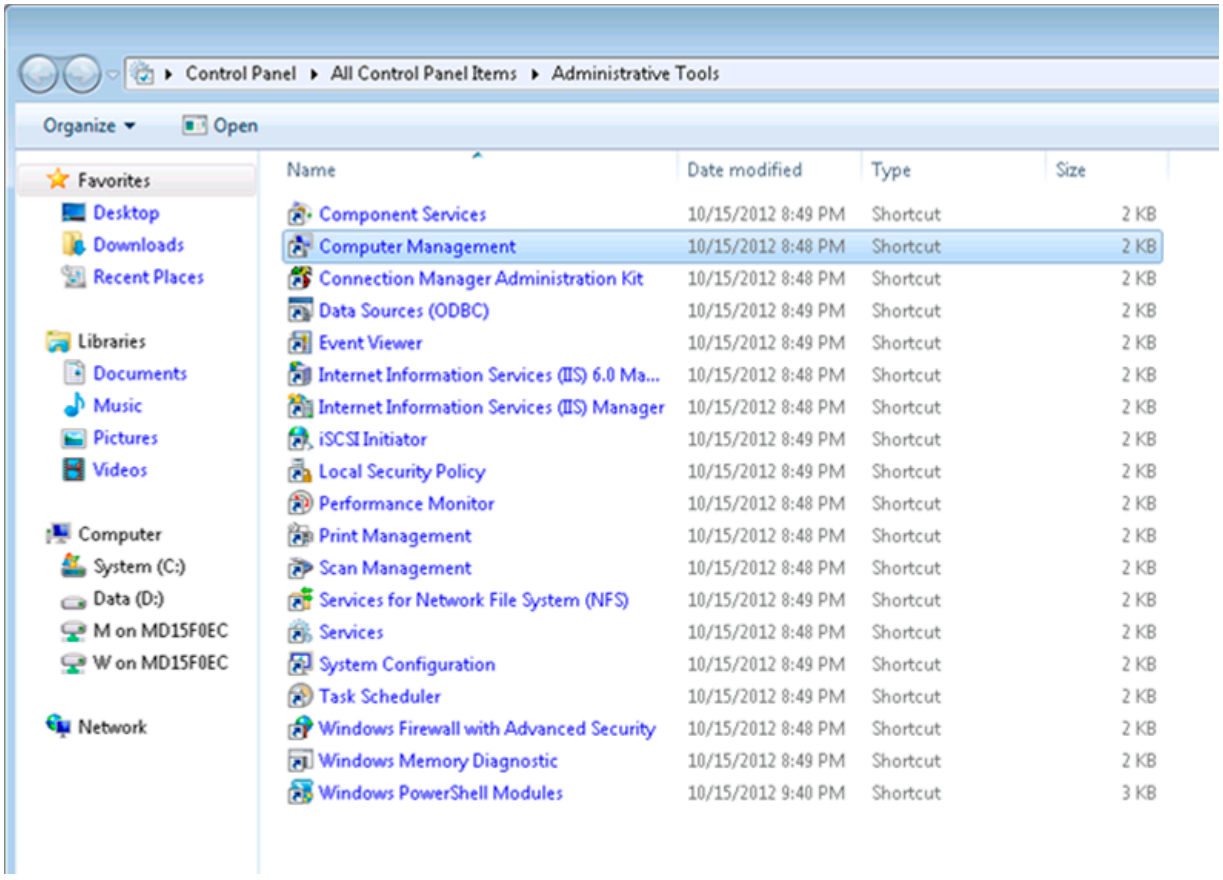


Figure 11-117 Homepage:

2. The **Users** folder is displayed at **Local Users and Groups** in the **Computer Management** window.
   Click the **Users** folder. The available user groups are displayed.

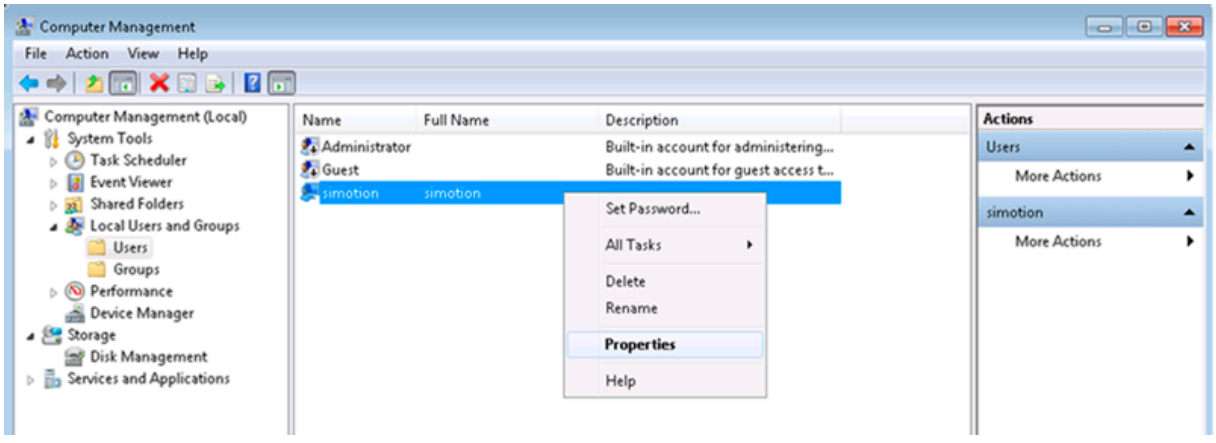


Figure 11-118 Computer management

3. With the selection of **Users simotion**, you can display the properties via **Properties**.
4. Deactivate the "**User cannot change password**" checkbox in the **General** tab on the **simotion Properties** dialog box.
   This checkbox must be deactivated for the user to change the Windows password.

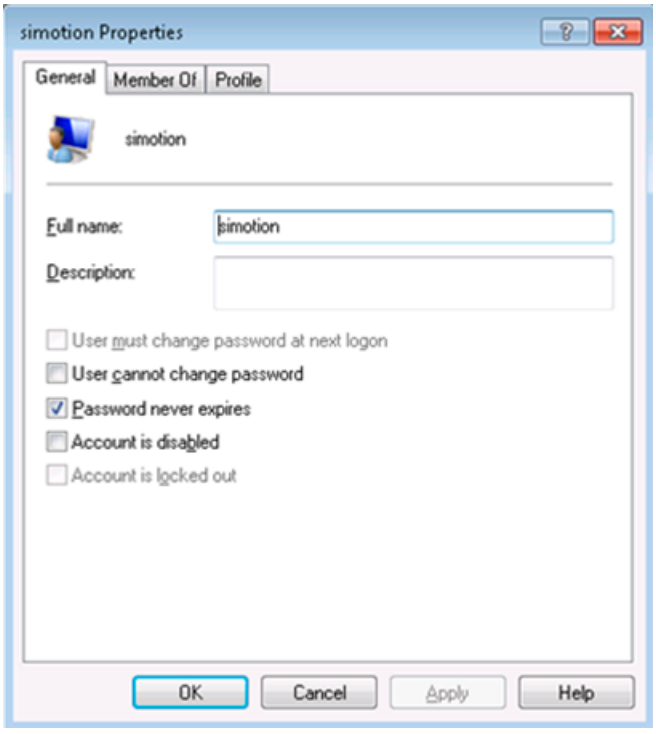


Figure 11-119 SIMOTION properties

Changing the Windows password

You can find general information on the Windows password on the website at Microsoft - Changing des Windows-password (http://windows.microsoft.com/en-us/windows/change-windows-password#change-windows-password=windows-7).

To change the Windows password, proceed as follows:

1. Press Ctrl+Alt+Del. The following Windows window opens.

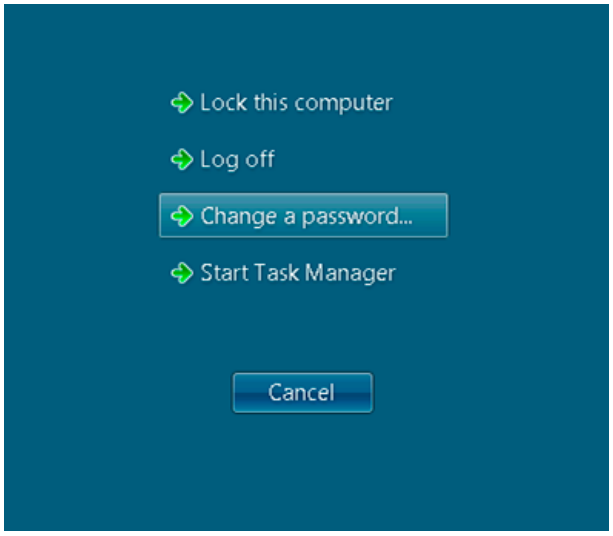


Figure 11-120 Changing the Windows password

2. After you have confirmed **Change a password...** , the Windows window to change the Windows password opens.
   First enter your current password and then the new Windows password. You must enter the new Windows password a second time for confirmation.

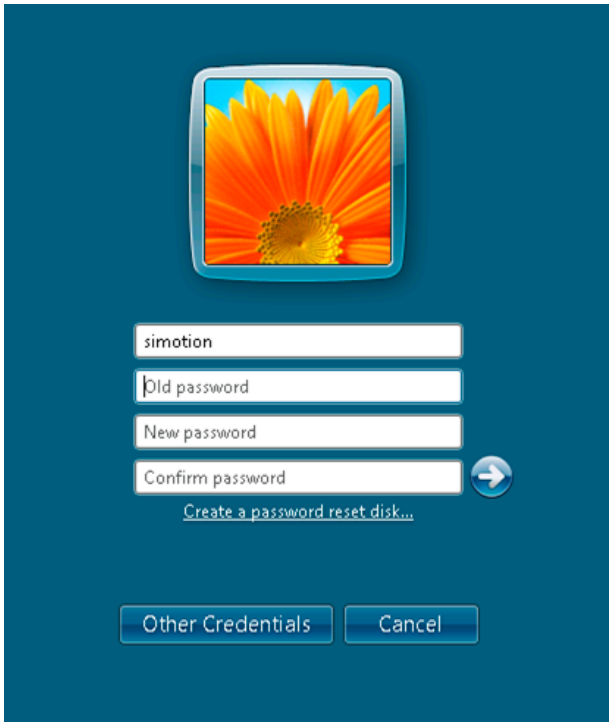


Figure 11-121 Entering the new Windows password

3. The password was changed successfully.

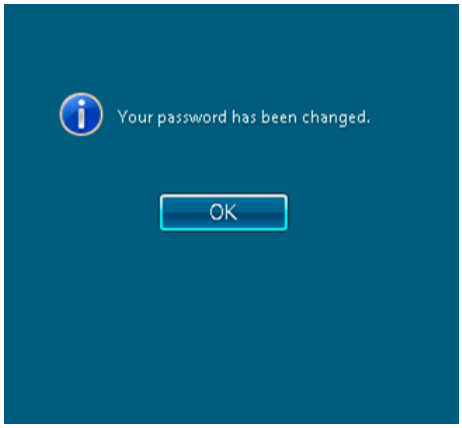


Figure 11-122 Confirmation of change

4. In the event of an error when the "User cannot change password" checkbox is activated, the following error message is displayed.
The Windows password has not been changed.

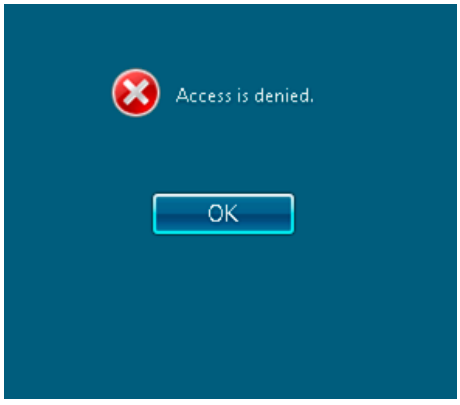


Figure 11-123 Error message when changing the Windows password has not been permitted.

Changing the AutoLogin

The password for the AutoLogin must be changed when the user has changed the Windows password.

The Windows user password is changed in the normal way, e.g. "Change Password" screen. See also Section Changing the Windows user password (Page 8020).

Automatically Log On

SIMOTION P320-4 is preset so that the user does not have to enter the user name and the password to log on.

Preparation for changing the AutoLogin

In order that an AutoLogin still functions after changing the Windows password, the password for the AutoLogin must be changed.

The required procedure is shown in the following so that the change of the AutoLogin for SIMOTION P320-4 can be prepared:

1. Open the **Computer Management** window via **Control Panel** > **All Control Panel Items** > **Administrative Tools** .

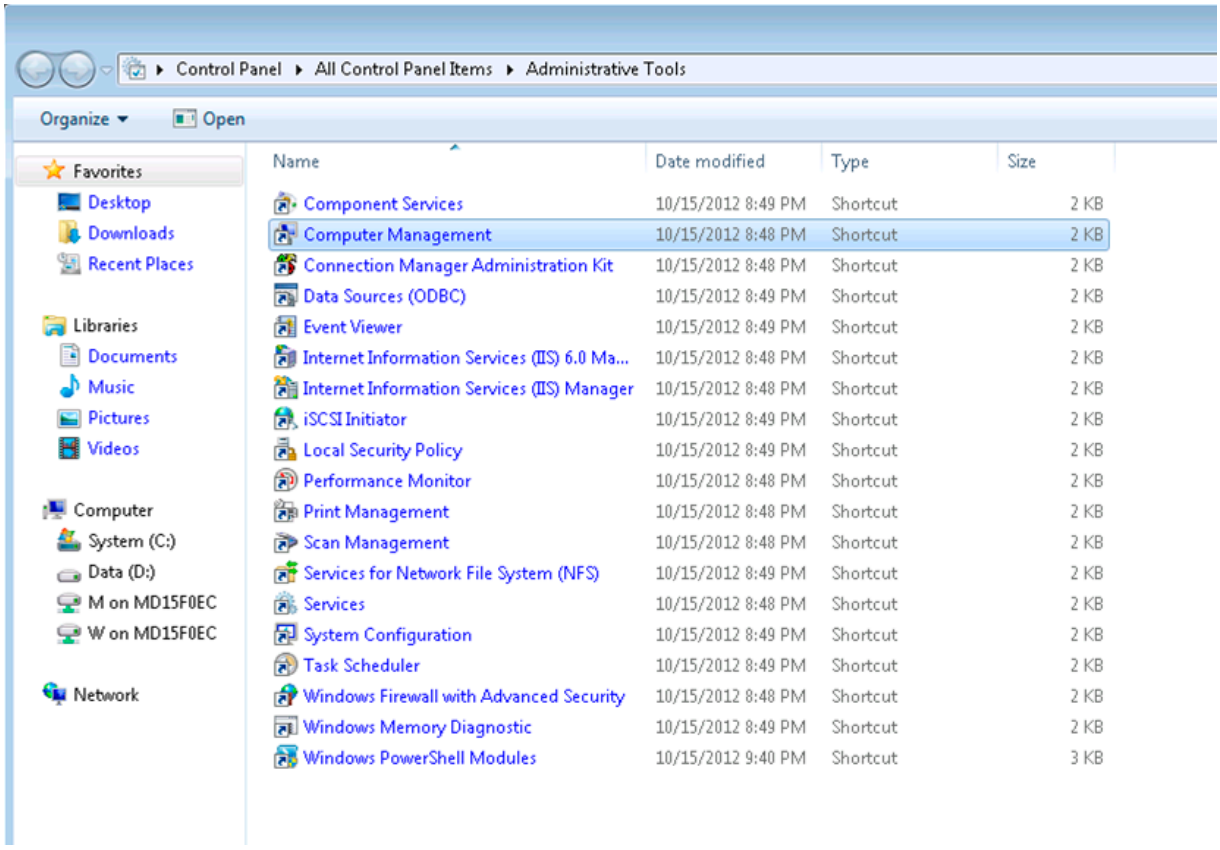


Figure 11-124 Computer Management entry point

2. The **Users** folder is displayed at **Local Users and Groups** in the **Computer Management** window.
   Click the **Users** folder. The available user groups are displayed.

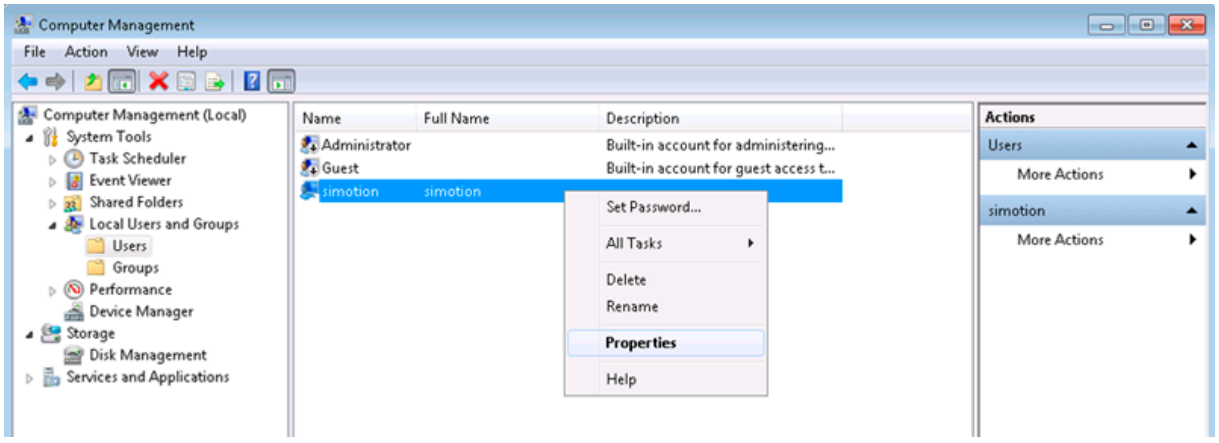


Figure 11-125 Computer Management - Users

3. With the selection of **Users**> **simotion**, you can display the properties via **Properties**.

4. Deactivate the **User cannot change password** checkbox in the **simotion Properties** dialog box.

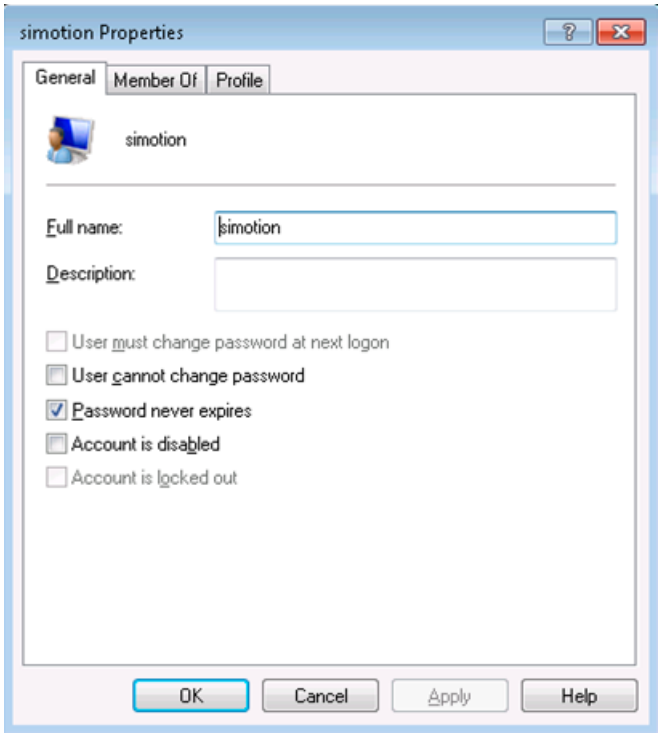


Figure 11-126 simotion Properties - General dialog box

5. Confirm with **OK.**

Changing the password for AutoLogin

To change the password for the AutoLogin, proceed as follows:

1. Open the search via **Windows-Start** and enter: **netplwiz**

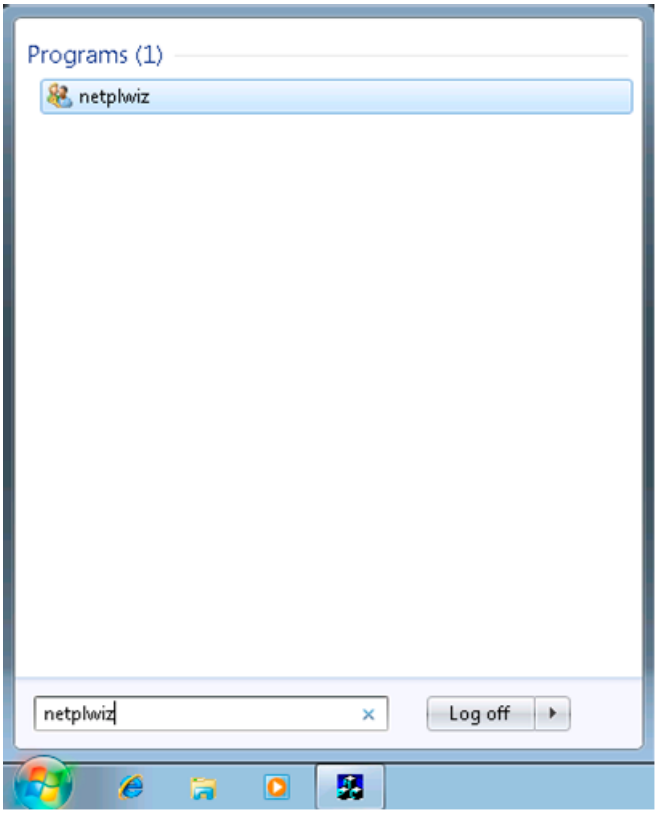


Figure 11-127 Programs netplwiz

2. The following **User Accounts** dialog box opens.

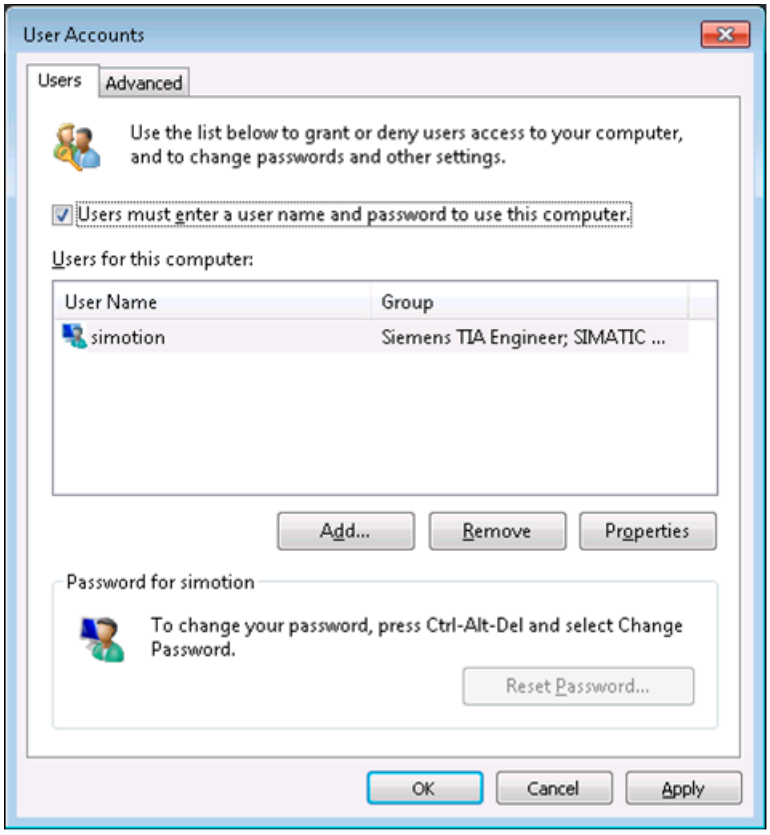


Figure 11-128 User Accounts

3. In the **Users** tab, activate the **Users must enter a user name and password to use this computer** checkbox.
   The **Apply** button is now active. Deactivate the checkbox and accept the changes now with the **OK** or **Apply** button.
   This is necessary so that the **Automatically Log On** dialog box is subsequently displayed.

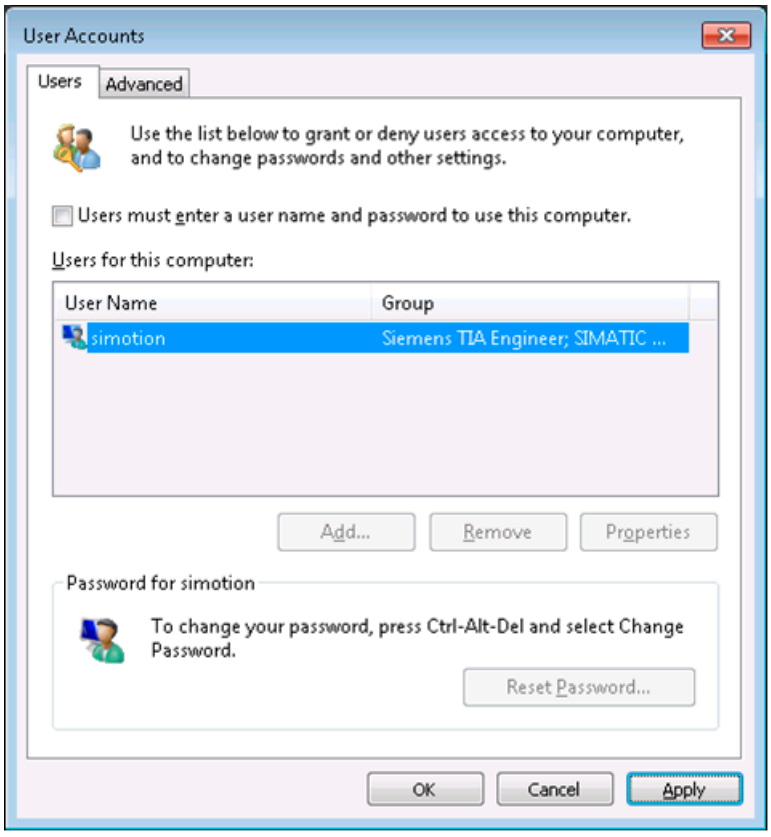


Figure 11-129 User accounts

4. Enter your changed Windows user password in the **Automatically Log On** dialog box and confirm it by entering it a second time.

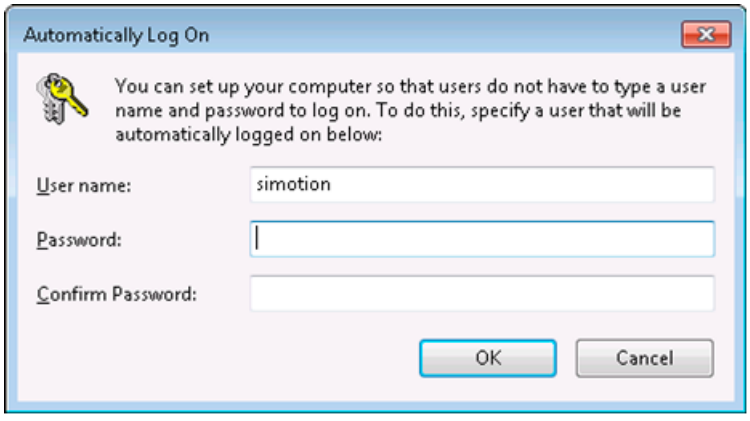


Figure 11-130 Automatically Log On dialog box

5. Confirm the password with **OK**.

Deactivating the remote desktop connection

Deactivating the remote desktop connection

If you do not use the headless mode and do not require a remote desktop connection, we recommend that you explicitly disable the remote desktop connection.

NOTICE
Deactivating the remote desktop connection
To ensure that the remote desktop connection is not used, it must be explicitly deactivated.

Procedure for deactivating the remote desktop connection

1. Using the Control Panel, select **System** > **Remote settings**.

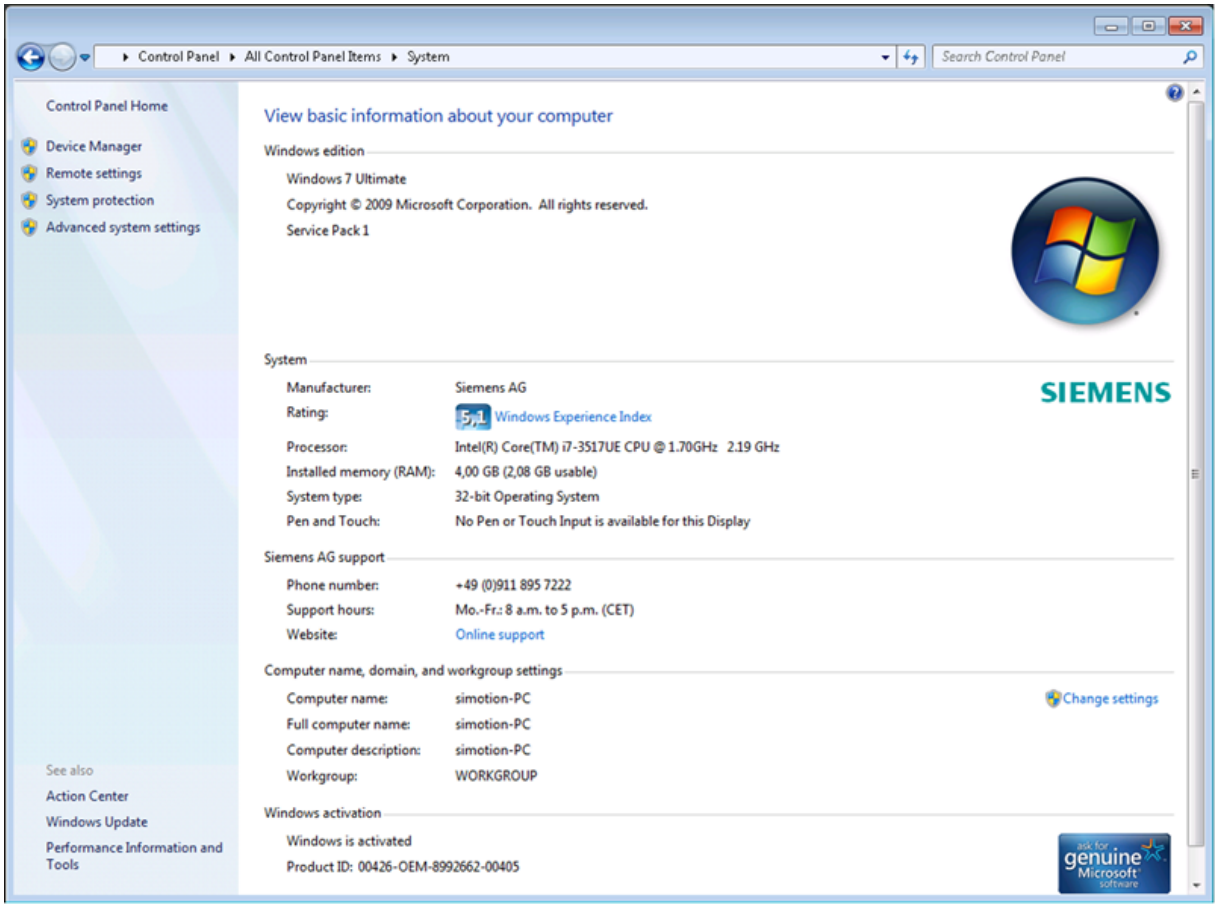


Figure 11-131 Remote settings

2. Select the **Remote** tab in the **System Properties** dialog box.

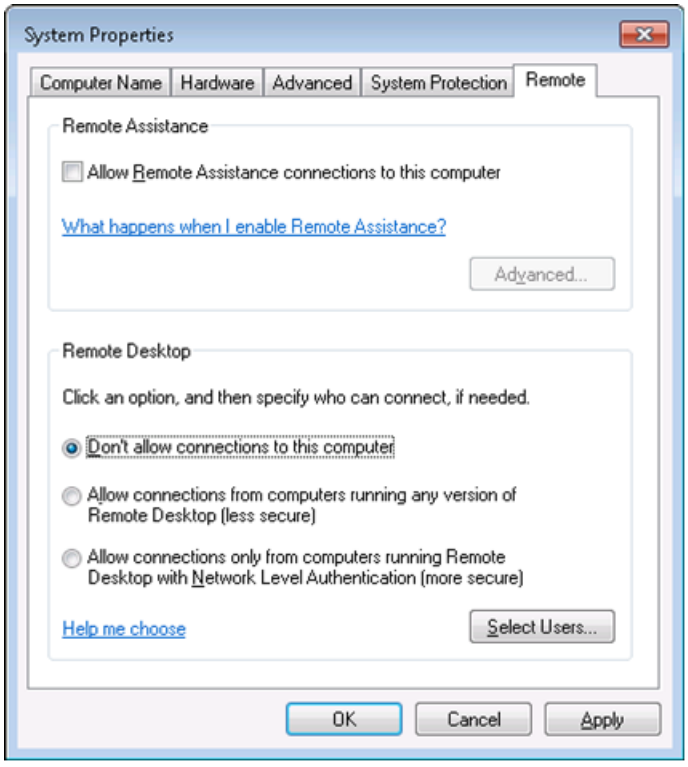


Figure 11-132 System Properties dialog box

3. Select the **Don't allow connections to this computer** option at **Remote Desktop**.
4. Confirm with **OK**.

Any remote desktop access is deactivated with this operation.

11.2.1.3 Description

System overview

Overview

SIMOTION P is a PC-based, open Motion Control System from SIMOTION.
Control, motion control, and HMI functions are executed together with standard PC applications on the SIMOTION P hardware platform. SIMOTION P combines the compatibility of the Windows operating system with real-time capability of SIMOTION P Runtime.

The fully independent SIMOTION P Runtime runs in parallel to Windows on SIMOTION P. This real-time expansion makes it possible to implement demanding motion control applications with high performance requirements on platforms of the SIMOTION P range. The hardware consists of a computing unit with innovative Intel technology, which is ready for operation at the time of delivery.

The drives and I/O devices are connected either via PROFINET onboard or IsoPROFIBUS board (optional).

The SIMATIC Flat Panel IFP1500, IFP1900 and IFP2200 can be used for the operation of the SIMOTION P320-4 hardware platform in a distributed configuration.

SIMOTION P320-4 versions

The SIMOTION P320-4 can control various I/O systems and HMI components via PROFINET onboard or the optional IsoPROFIBUS board.
If required, a USB DVD drive can be connected, for example.

The following versions of the SIMOTION P320-4 are available:

- SIMOTION P320-4 E
  with the Windows Embedded Standard 7 32-bit operating system and real-time expansion for SIMOTION.
- SIMOTION P320-4 S
  with the Windows 7 Ultimate 32-bit operating system and real-time expansion for SIMOTION.

Application

The SIMOTION P320-4 applications are directed at machines that require a high level of integration of PLC, motion control and technology functions on account of the increasing use of servo drives:

- Packaging machines
- Plastic and rubber processing machines
- Presses, wire-drawing machines
- Textile machines
- Printing machines
- Machines for processing wood, glass, ceramics, and stone
- Production lines in the renewable energy sector, e.g. solar technology, wind power installations

SIMOTION P320-4 product description

SIMOTION P320-4 overview

The SIMOTION P320-4 provides high-level industrial performance. It features:

- Compact design
- Maintenance-free operation
- High degree of ruggedness
- Long-term availability

Figure 11-133 SIMOTION P320-4 view

Figure 11-134 SIMOTION P320-4 (open perspective) with plugged-in optional IsoPROFIBUS board

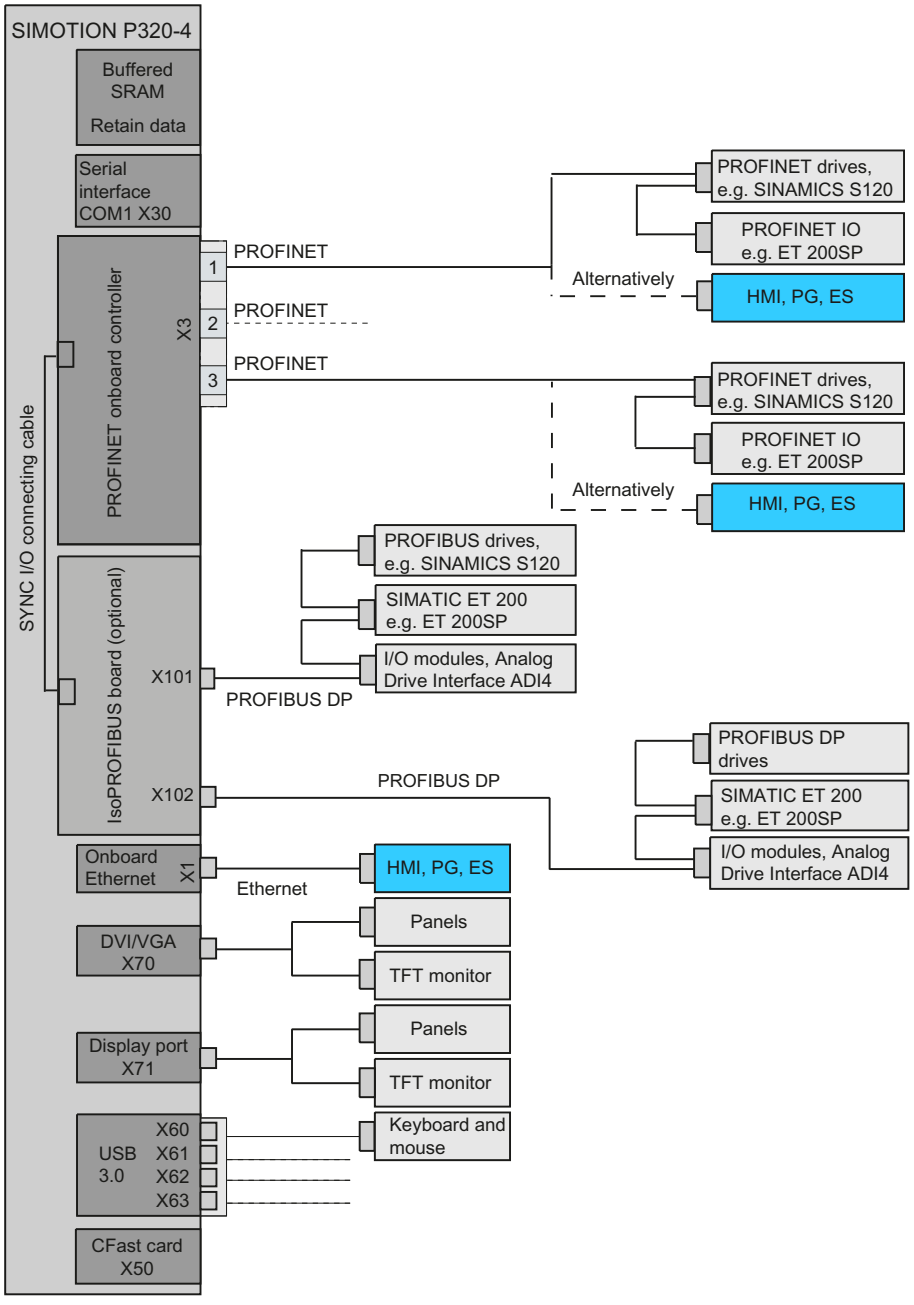
Features

Basic data	
Installation	• Standard rail mounting • Vertical mounting
Processor	
P320-4 E	Intel Core i3-3217UE, 2 x 1.6 GHz, 3 MB cache
P320-4 S	Intel Core i7-3517UE, 2 x 1.7 GHz, 4 MB cache
Main memory	4 GB DDR3 RAM
Optional IsoPROFIBUS board (PROFIBUS DP)	2 x SUB-D socket with configurable baud rates (9.6 Kbit/s - 12 Mbps)
Graphics	• Integrated Intel HD2000 or HD4000 • DVI resolution of 640 × 480 pixels up to 1920 × 1200 pixels • Display port resolution max. 1920 × 1200 pixels • Graphics memory is occupied in the main memory (dynamic UMA)
Power supply	24 VDC (-20%/+20%) max. 4 A
Operating conditions	Operation without fan
Drives and storage media	
CFast card or SSD (Solid State Disk)	Depending on the hardware version of the SIMOTION P320-4
P320-4 E	**2 x CFast card** Internal interface: CFast External interface: CFast
P320-4 S	**SSD (Solid State Disk)** Internal interface: SSD **CFast card** External interface: CFast
USB stick	External, can be connected via USB interface
Interfaces	
Serial	COM (RS 232)
Graphics	DVI-I: Suitable for use as DVI or VGA DPP++: Display port, DVI via DPP-to-DVI adapter
USB	4 × USB 3.0, simultaneous operation of high current, backward compatible with USB 2.0/1.1
Ethernet	1 × RJ45 (10/100/1000 Mbps)
PROFINET I/O	3 × RJ45 (100 Mbps)
Keyboard, mouse	Can be connected via USB interface

Software	
Operating systems	
P320-4 E	Windows Embedded Standard 7 32-bit
P320-4 S	Windows 7 Ultimate 32-bit

SIMOTION P320-4 (hardware) structure

The following figure shows the integration of the SIMOTION P320-4 with integrated PROFINET onboard and optional IsoPROFIBUS board in a target system.



HMI Human Machine Interface

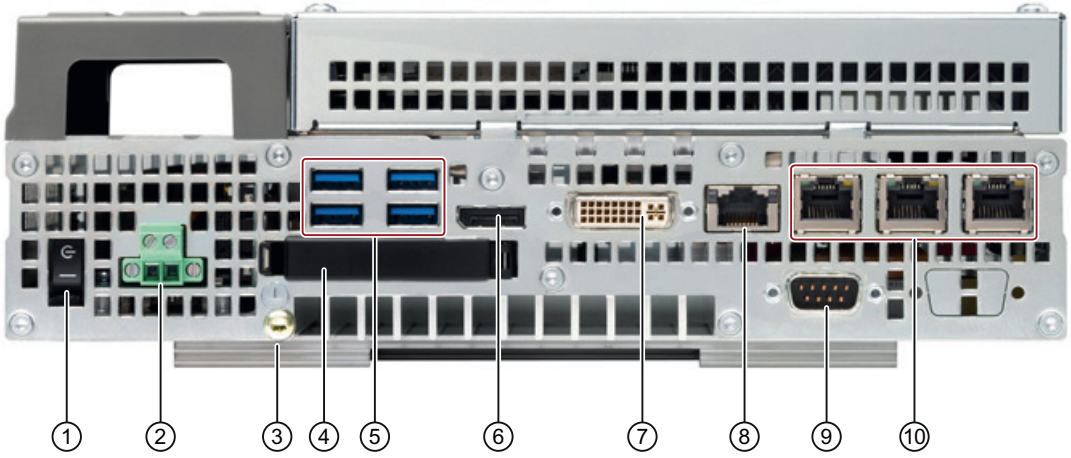
PG Programming device

ES Engineering system

Figure 11-135 SIMOTION P320-4 system overview

Interfaces and operating elements

SIMOTION P320-4



①	On/off switch	Position **OFF** is the delivery state. The on/off switch does not isolate the device from the power supply.
②	24 V DC	Power supply connection
③	Protective conductor	Protective conductor terminal
④	Memory card slot	Cover for the CFast card
⑤	4 × USB 3.0	USB 3.0 high current, backward compatible with USB 2.0/1.1
⑥	Display port	Display port connection for digital monitor
⑦	DVI-I	DVI connector for CRT or LCD monitor with DVI interface
⑧	Ethernet	RJ45 Ethernet connection for 10/100/1000 Mbps
⑨	COM1	Serial interface
⑩	3 × PROFINET	PROFINET ports with three RJ45 sockets

SIMOTION P Runtime (software) structure

SIMOTION P Runtime contains the SIMOTION P Kernel which performs the motion control and interface control.

A SIMOTION project is planned, configured, assigned parameters, commissioned and programmed via the SIMOTION SCOUT engineering system (ES).

The user data can be stored on the data media of the SIMOTION P320-4 .

Siemens WinCC flexible software can be used to visualize operational sequences or to operate the machine. Third-party systems can be linked via the OPC interface.

SIMOTION P Runtime

With the SIMOTION P320-4, the PLC and motion control functionality (starting from position controller upwards) is located centrally in a strictly deterministic task outside the Windows operating system.

Its main field of application is centralized motion control and control tasks requiring close coordination between multiple axes and/or input/output modules.

Functionality ranges from simple positioning to high-performance synchronous operation.

SIMOTION SCOUT

The SIMOTION SCOUT engineering system can be installed on the SIMOTION P320-4 S or connected via the interfaces integrated in SIMOTION P320-4 .

SIMOTION SCOUT TIA

As of version V4.5, the SIMOTION SCOUT TIA engineering system no longer supports the Windows 7 32-bit operating system and therefore cannot be installed.

HMI software

HMI/Runtime software can be operated on the same PC, e.g. the Siemens WinCC flexible software. Other software packages can be linked by means of the OPC interface.

The **HMI software** is used as the general term in the following.

Internal communication

A locally installed HMI can use the local communication to access the following:

- Variables in SIMOTION RT
- Drives on PROFINET IO or PROFIBUS DP
- Other SIMOTION devices on PROFINET IO or PROFIBUS DP

Scope of delivery

The system software supplied with the SIMOTION P320-4 is either already installed on the SIMOTION P320-4 storage medium or is ready to be installed:

- SIMOTION P Kernel
- SIMOTION IT
- SIMOTION IT VM Virtuell Machine (subject to license)

Components

The most important components of the SIMOTION P320-4 and their functions are listed below.

Distributed I/O systems (PROFINET)

Table 11-17 Components for distributed I/O

Component	Function
SIMATIC ET 200M	Modular I/O system for cabinet installation and high channel densities.
SIMATIC ET 200S	Finely modular I/O system for cabinet installation including motor starters, safety technology, and individual grouping of the load groups.
SIMATIC ET 200SP	Finely scalable I/O system for cabinet installation; SIMATIC ET 200SP features a single-cable and multi-cable connection with push-in terminals, compact dimensions, high performance, and low part variety.
SIMATIC ET 200MP	Modular I/O system for cabinet installation and high channel densities in the SIMATIC S7-1500 packaging system. SIMATIC ET 200MP permits the shortest bus cycle times and fastest response time even with large volumes of data.
SIMATIC ET 200eco	I/O system with IP67 degree of protection for machine-related, cabinet-free applications, featuring a flexible and fast connection system in ECOFAST or M12.
SIMATIC ET 200pro	Modular I/O system with IP65/67 degree of protection for machine-related, cabinet-free applications, including motor starters.

Note

Note that not all modules of the above-mentioned I/Os or I/O systems have been released for SIMOTION.

There may also be system-dependent functional differences with regard to the use on SIMOTION and on SIMATIC.
For example, special process-control functions (e.g. HART modules, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

For a detailed and routinely updated list of I/O modules enabled with SIMOTION as well as application information, visit us online at:

SIEMENS Industry Online Support - Product support - Supplementary system components (https://support.industry.siemens.com/cs/de/en)

Further modules are integrated via the GSD file of the device's manufacturer.

Note

Note that in individual cases, additional supplementary conditions must be fulfilled.

Drive systems (via PROFINET IO)

Table 11-18 List of typically connected drive systems

Component	Function
SINAMICS S120	Servo drives, innovative single-axis and multi-axis solution

Optional components

Table 11-19 Optional components

Centralized I/O	Function
IsoPROFIBUS board	PROFIBUS connection
UPS	Uninterruptible Power Supply The drivers for the UPS are not pre-installed on the PC. Install the current drivers that are supplied with the UPS.

Note

An IsoPROFIBUS board may be inserted in addition to the PROFINET onboard.

HMI and SIMOTION SCOUT

HMI and SIMOTION SCOUT Overview

The operational sequences on the SIMOTION P320-4 are either monitored via the HMI system (Human Machine Interface) or the SIMOTION SCOUT engineering system.

The HMI or ES software can be connected using the following variants:

- Local (Page 8040)
- via PROFINET (Page 8053)
- via Ethernet (Page 8054)
- via IsoPROFIBUS (optional) (Page 8055)

Local communication of the HMI/ES

Only one local connection to the local SIMOTION Runtime is possible. If HMI or ES is also installed locally, **PC internal** can access other controllers/devices via PROFINET / PROFIBUS.

See Section Local HMI or ES on SIMOTION P320-4 (Page 8040).

Communication of the HMI / ES via PROFINET / Ethernet / PROFIBUS

By using one of these communication variants, you can control and monitor several SIMOTION devices.

Local HMI or ES on SIMOTION P320-4

The model described below is used for the communication of the SIMOTION P320-4 with a local HMI or ES. The SIMOTION SCOUT engineering system or the HMI system can be installed directly on the **SIMOTION P320-4 S**.

For this model, you must set the communication (access point) to **TCP/IP** .

The SIMOTION P320-4 is already configured for this communication version at the time of delivery.

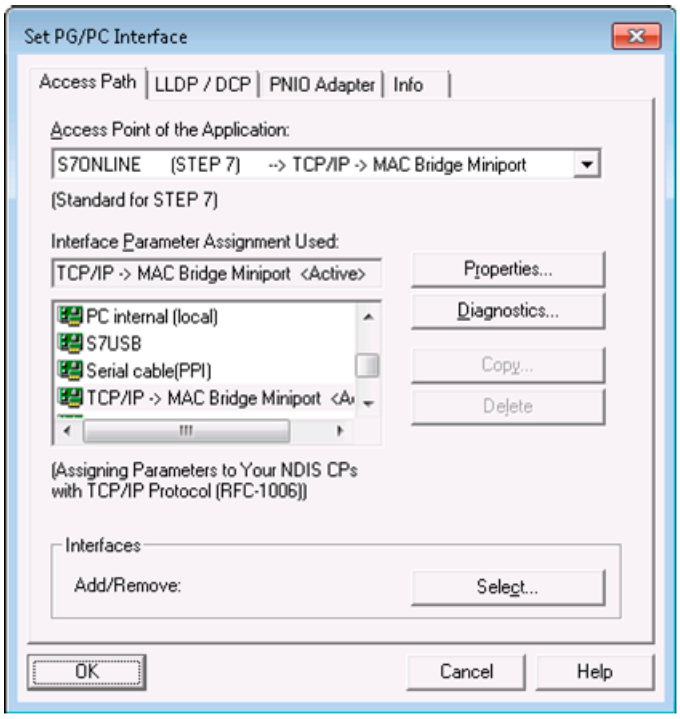


Figure 11-136 Set PG/PC Interface - TCP/IP -> MAC Bridge Miniport dialog box

Note

Local communication via Ethernet (TCP/IP)

For the communication via Ethernet (TCP/IP) to function locally, a link must be available to the Ethernet interface or a LAN cable must be connected **or** the MAC bridge set up.

Note

Setting up the MAC bridge miniport

See also Section Setting up a network bridge - MAC bridge miniport (Page 8042)

Note

Only HMI-Runtime can be installed on the **SIMOTION P320-4 E** version.

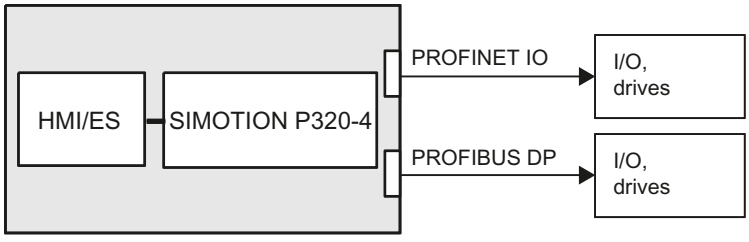


Figure 11-137 Model: Local HMI or ES

Alternatively to TCP/IP , SIMOTION SCOUT can also communicate locally via PC internal (local) .

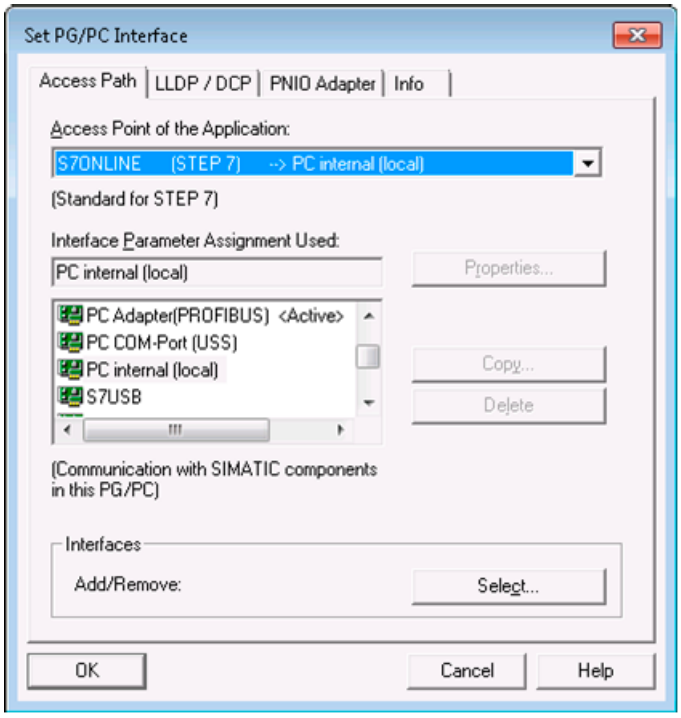


Figure 11-138 Set PG/PC Interface - PC internal (local) dialog box

See also

Local HMI or ES on SIMOTION P320-4 (Page 7858)

Setting up a network bridge - MAC bridge miniport

Note

If you are working with SIMOTION P320-4 version V4.5, the Network Bridge is already preset.

The network bridge must be preset, configured via script and set up as described in the following. Note that the MAC address of the network bridge is different to the MAC address of the Ethernet adapter.

The following is a step-by-step description of how the network bridge **MAC Bridge Miniport** is set up.

Network connections initial situation

Open the **Network and Internet** via the Control Panel.

- Select **Control Panel** > **All Control Panel Items** > **Network and Sharing Center** > **Change adapter settings**.
- Open the **Network and Internet** > **Network Connections** dialog box.
  The network connections of your PC are shown here.

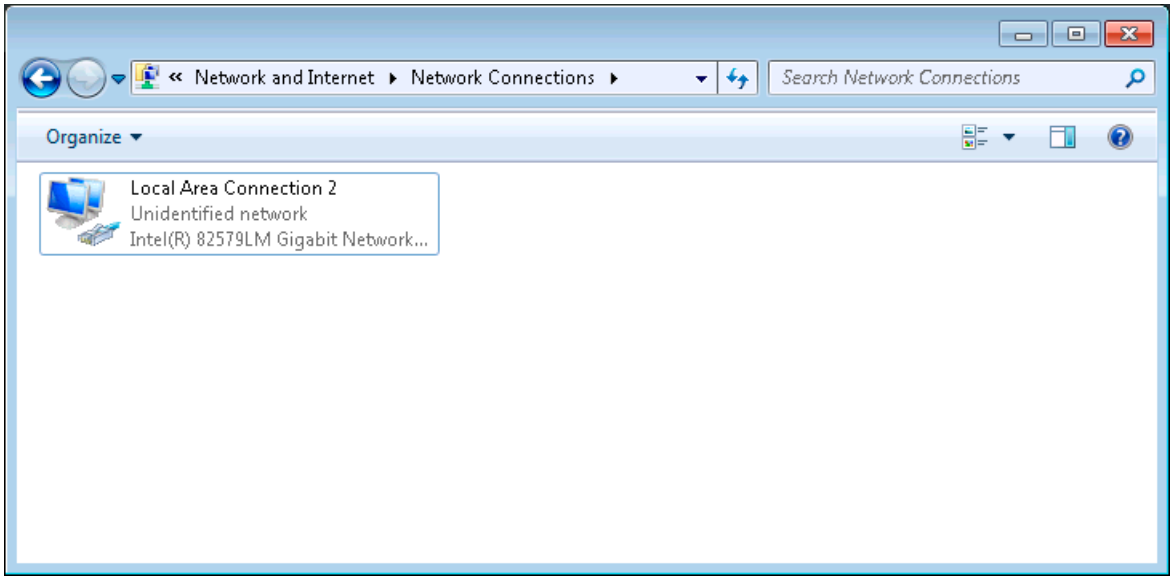


Figure 11-139 Initial situation - Network connections

Device Manager

Open the Device Manager via the Control Panel:

Control Panel > **All Control Panel Items** > **System** > **Device Manager**.

1. Select the **Network adapters** in the dialog box below the "SIMOTION PC". The available network connections are shown here.

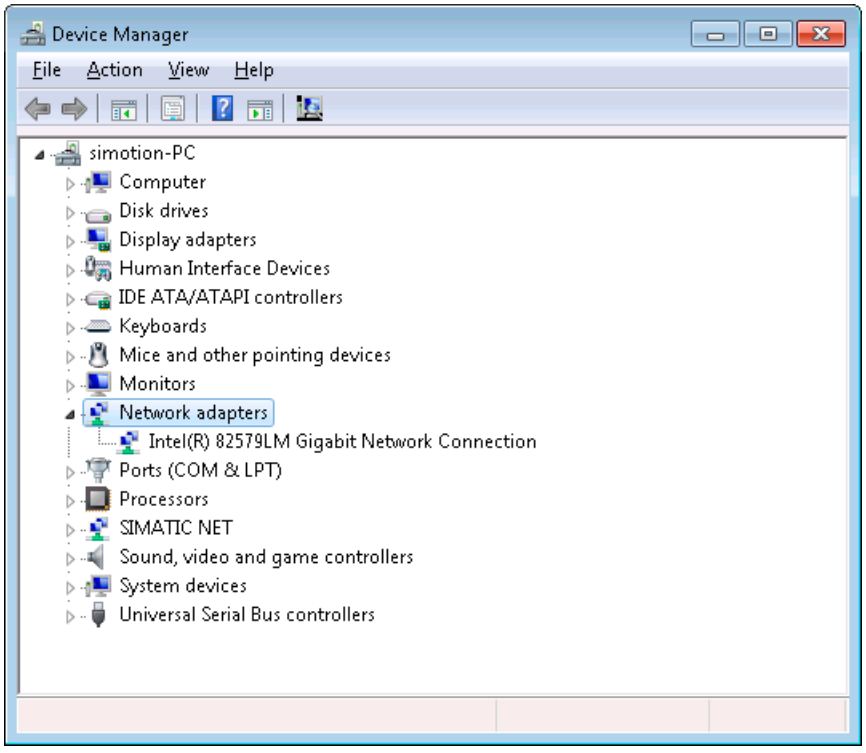


Figure 11-140 Opening the Device Manager

2. Install the **Microsoft Loopback Adapter** via the Device Manager.
3. To do this, select the menu **Action** > **Add legacy hardware**. The **Add Hardware Wizard** then opens.

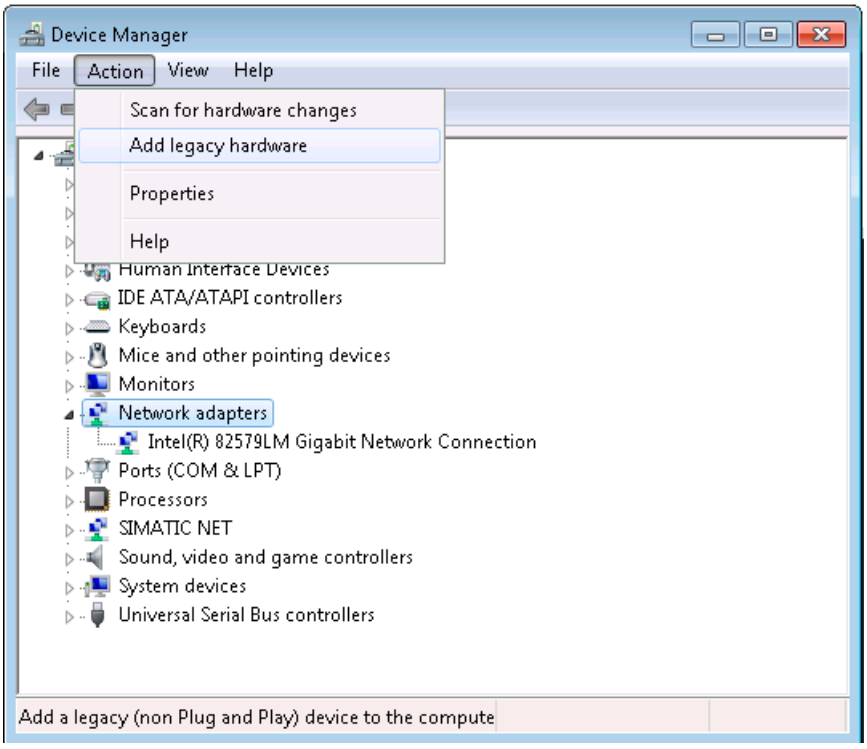


Figure 11-141 Device Manager - Action menu

Add Hardware Wizard

Use the **Add Hardware Wizard** to install the network bridge.

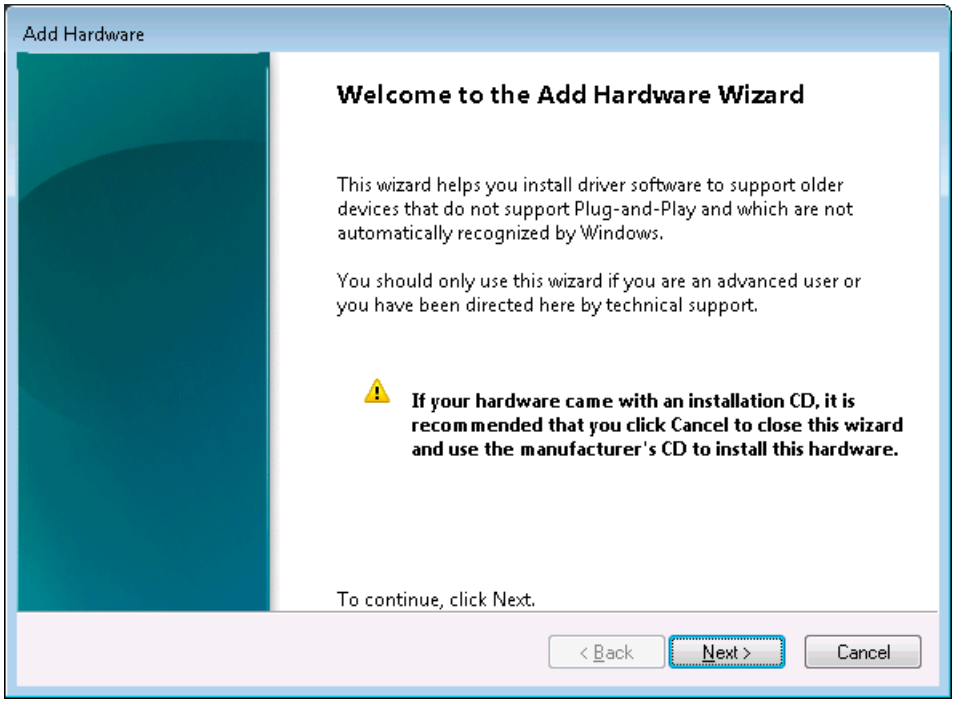


Figure 11-142 Add Hardware - Starting the wizard

1. Click **Next** to start the software wizard.
2. Select the option: **Install the hardware that I manually select from a list (Advanced)** and continue in the wizard with **Next**.

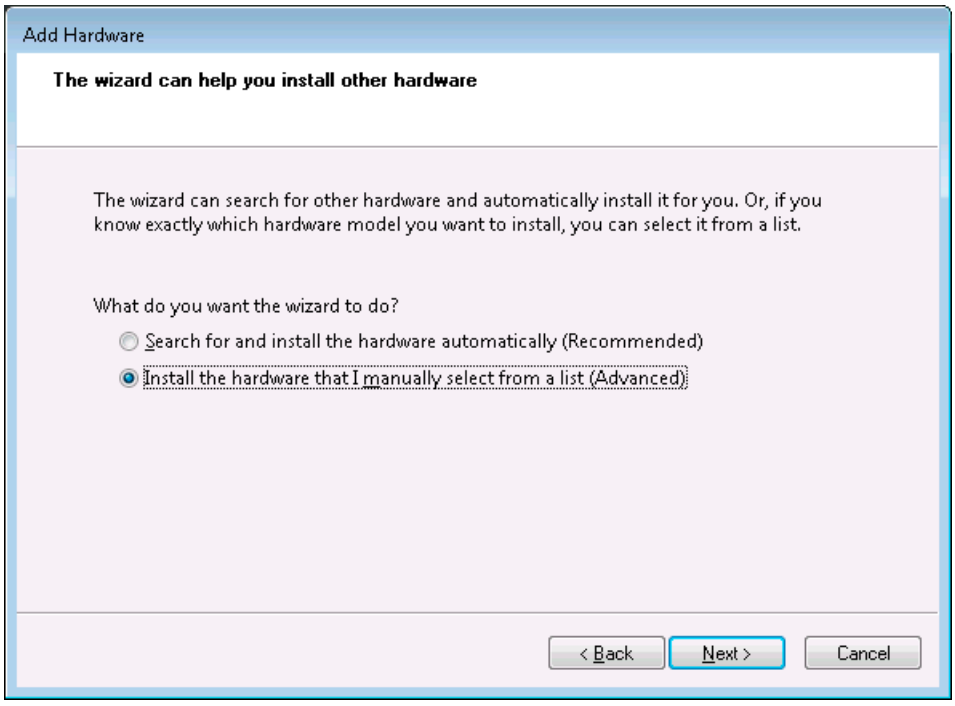


Figure 11-143 Add Hardware - Manual selection from a list

3. Select **Network adapters** in the dialog box now open and continue in the wizard with **Next**.

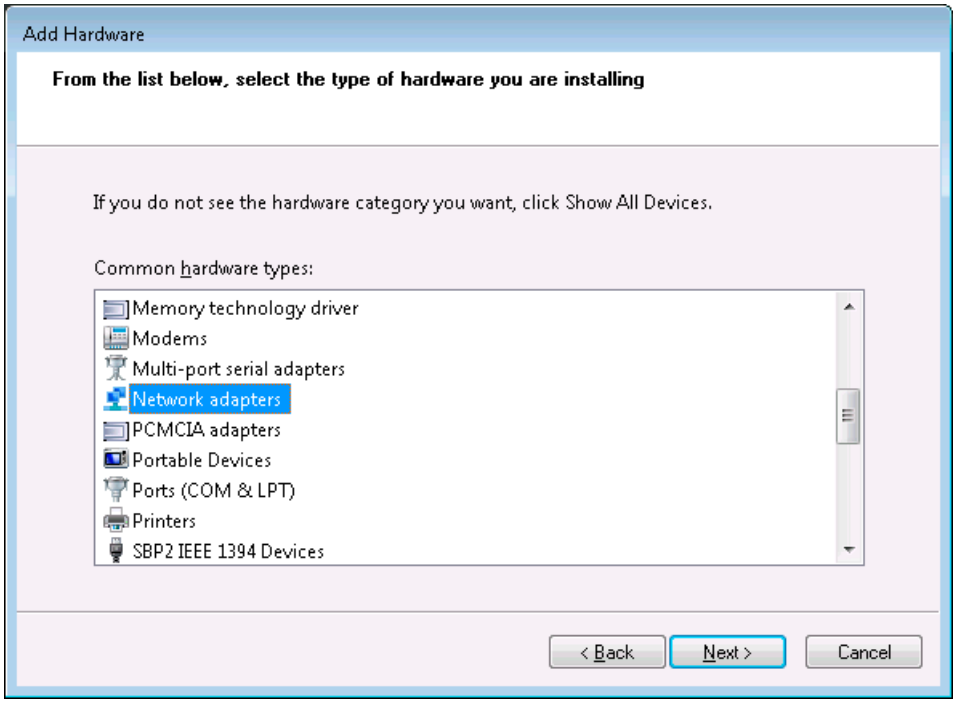


Figure 11-144 Select Add Hardware - Network adapters

4. In the following dialog box, select the **Microsoft Loopback Adapter**.
5. To do this, select Microsoft at **Manufacturer** and the Microsoft Loopback Adapter at **Network Adapter**.

6. Continue in the wizard with **Next**.

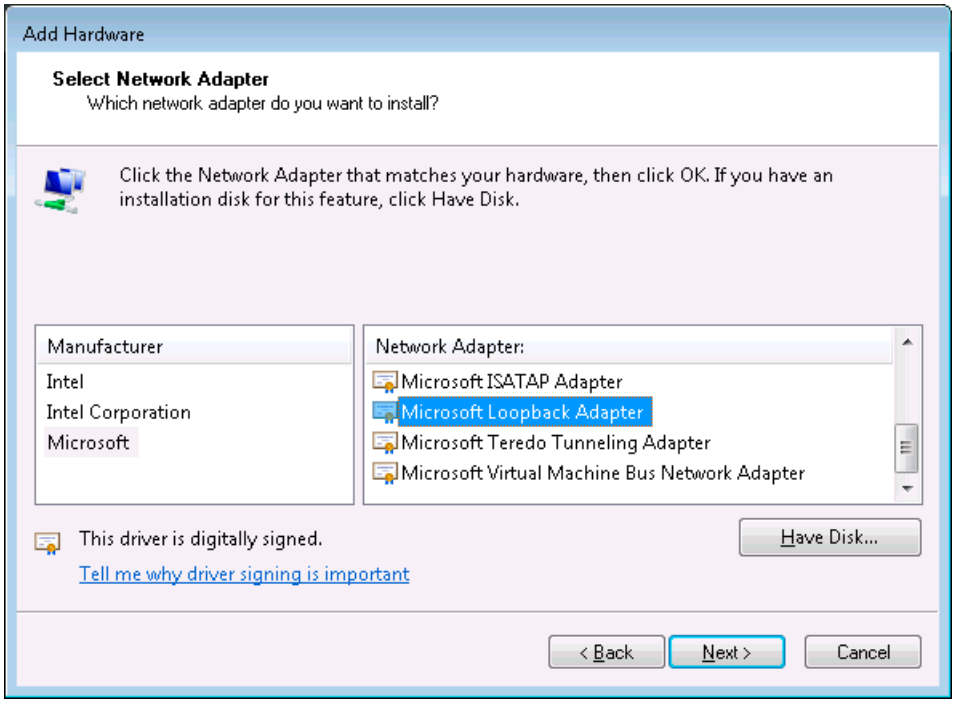


Figure 11-145 Add Hardware - Selecting the desired Network Adapter

7. The selected Network Adapter is shown in the following dialog box.
Click **Next** to confirm and install.

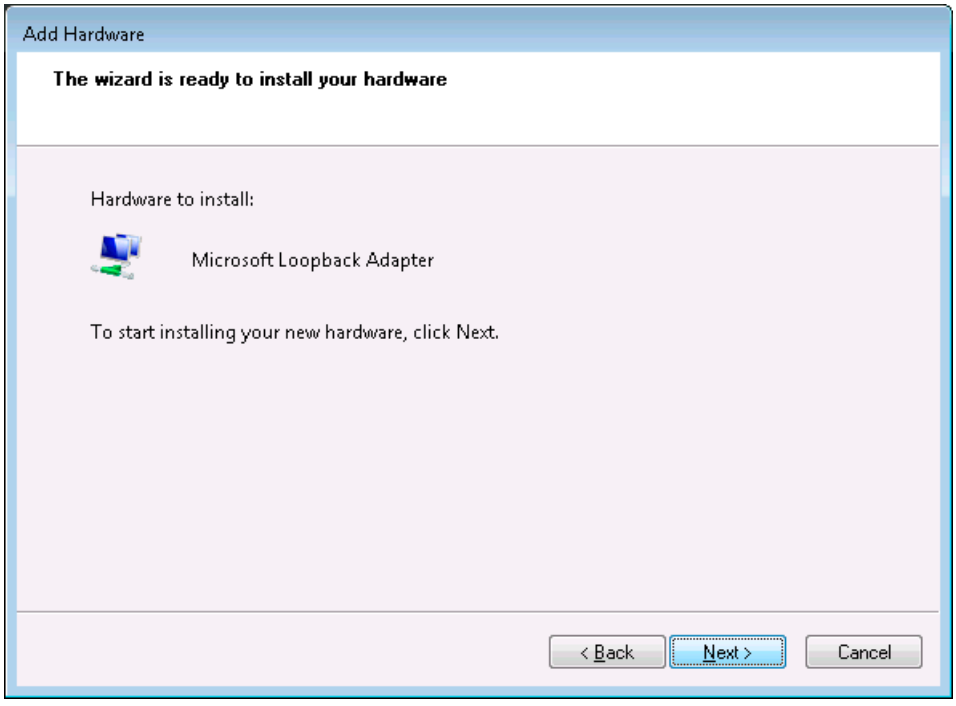


Figure 11-146 Add Hardware - Selected Network Adapter can be installed

8. After finishing the installation, the following dialog box is displayed.
Exit the wizard with **Finish**.

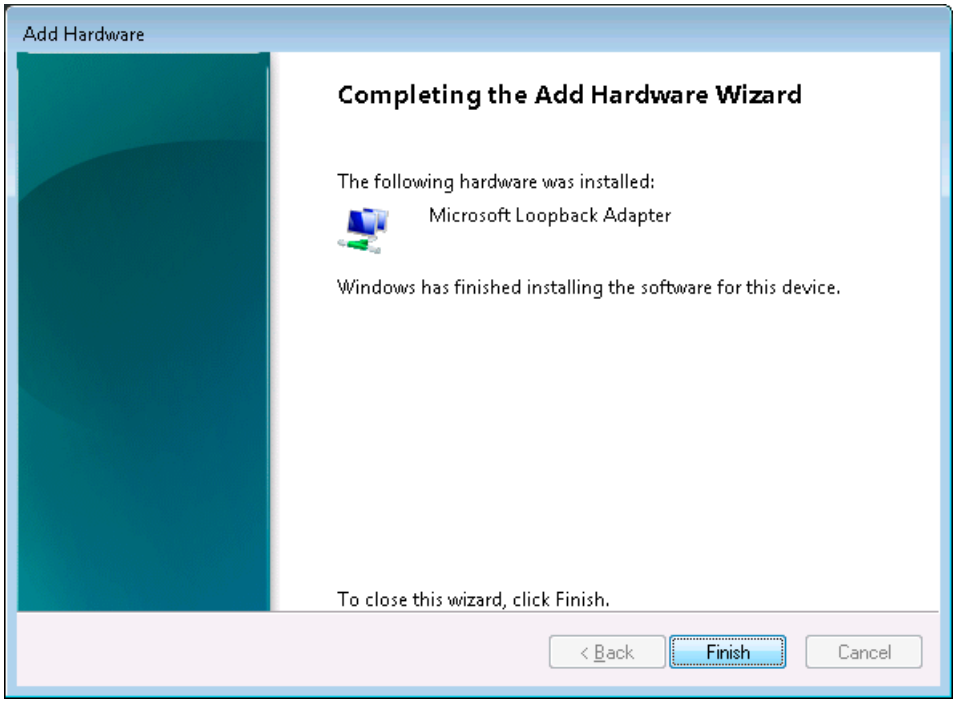


Figure 11-147 Add Hardware - Installation of the Microsoft Loopback Adapter is finished

Checking the installation of the network adapter

You can check the successful installation of the **Microsoft Loopback Adapters** in the

- Device Manager
- Internet connections

Device Manager

The Microsoft Loopback Adapter is displayed at **Network adapters** in the Device Manager.

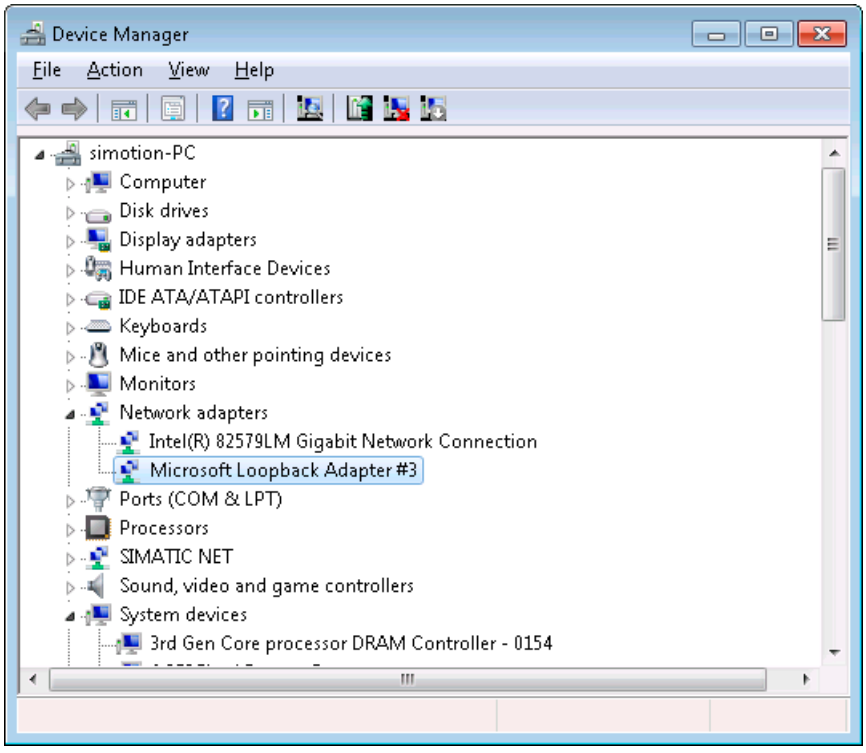


Figure 11-148 Device Manager - Microsoft loopback adapter

Network connections

The newly installed Microsoft Loopback Adapter is also displayed at **Network Connections**.

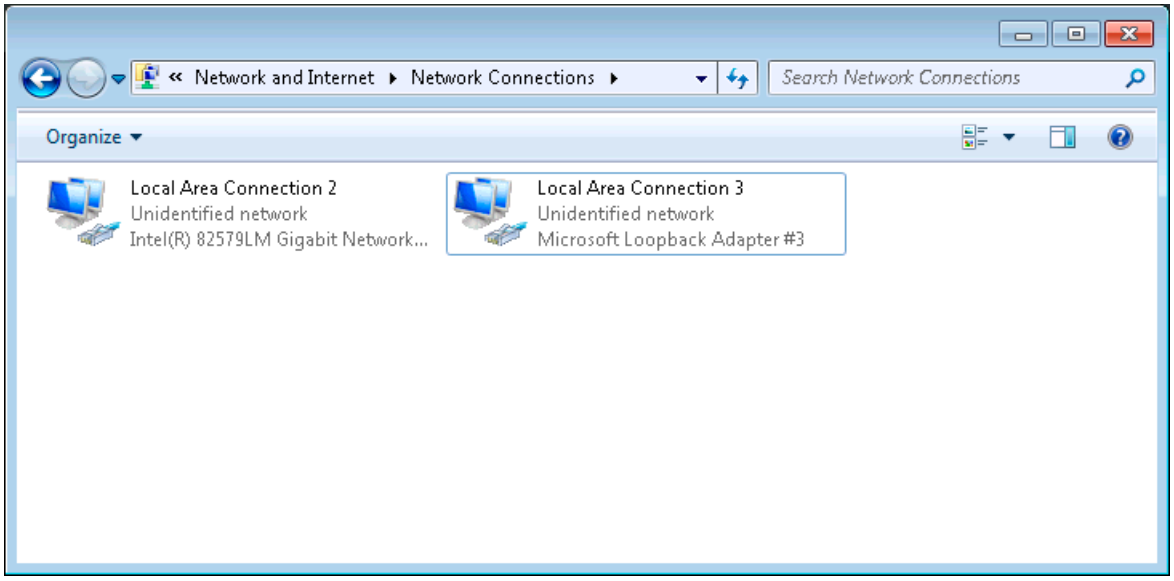


Figure 11-149 Network connections - Display of the newly installed Microsoft loopback adapter

Creating the network bridge

To activate the network bridge, proceed as follows:

1. Select both adapters at Network Connections
2. Select Bridge Connections via the context menu.

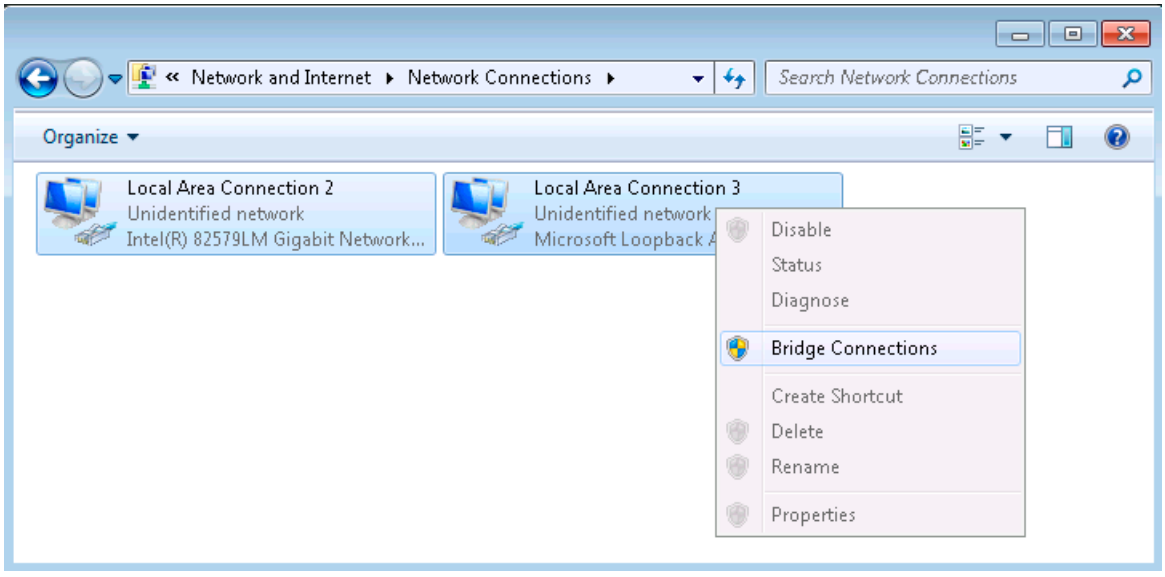


Figure 11-150 Network Connections - Setting the Bridge Connection

3. The Network Bridge MAC Bridge Miniport is displayed.

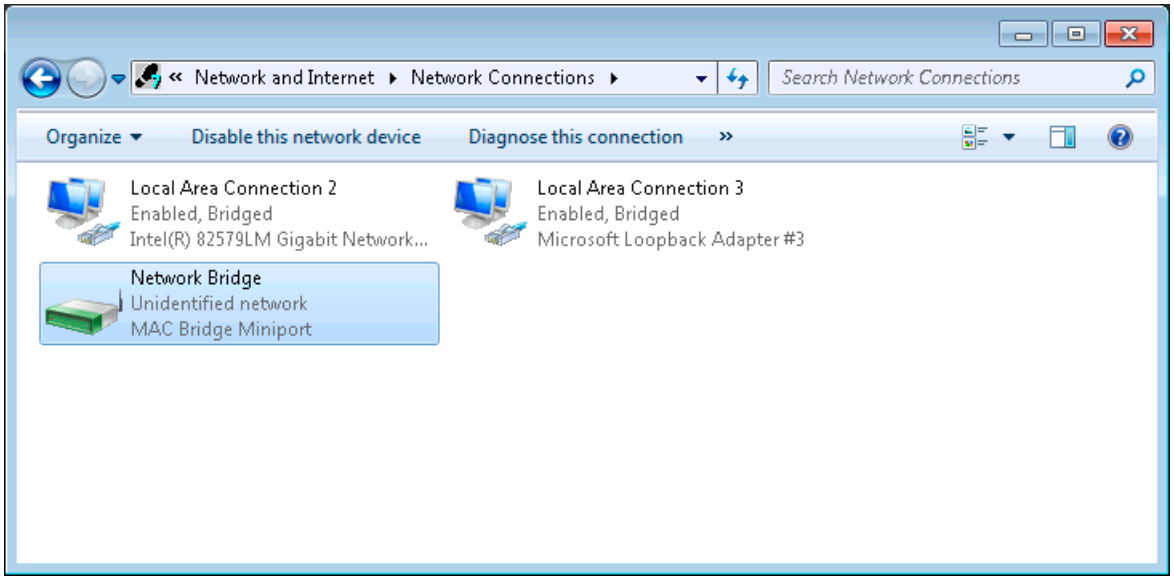


Figure 11-151 Network Connections - Display of the Network Bridge MAC Bridge Miniport

4. To initialize the network bridge, execute the **sp_bridge.cmd** Windows command script.
   The additional script can be found at C:\SiMotion\tools\sp_bridge.cmd
   After executing the batch file, the network bridge has been set up.

Setting "Set PG/PC interface"

Now open the PG/PC interface to complete the setting for the **Network Bridge** .

1. Open the **Set PG/PC Interface** dialog box via the **Control Panel** entry point.
2. Deactivate the **MAC Bridge Miniport DCP** checkbox in the **LLDP / DCP** tab.
   If you do not deactivate the MAC Bridge Miniport DCP, the SIMOTION P320-4 may not be found via Accessible nodes.

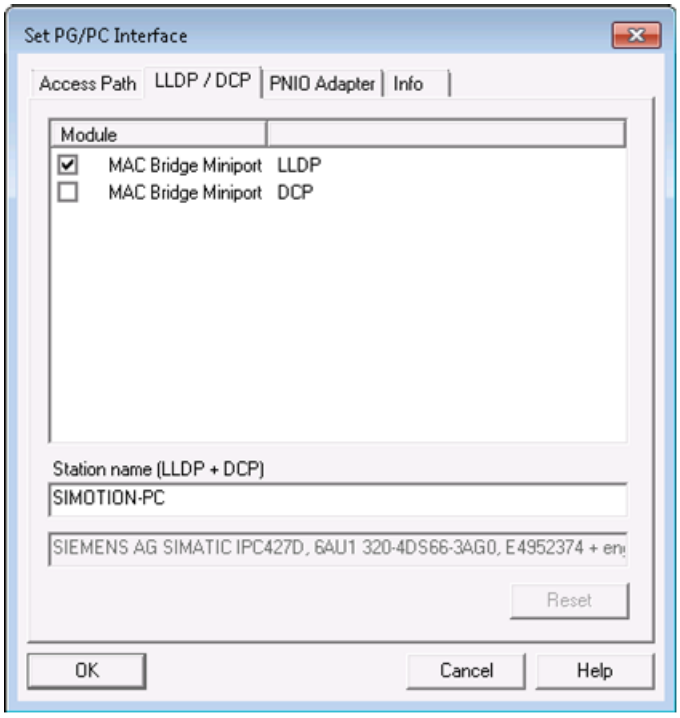


Figure 11-152 Set PG/PC interface

The settings for the network bridge have now been completed.

HMI or ES via PROFINET

Networking via PROFINET

PROFINET is the innovative and open Industrial Ethernet standard (EN 61158) for industrial automation. With PROFINET, devices can be linked up from the field level through to the management level.

With PROFINET, an external HMI or ES can also be integrated into the network and data can be exchanged directly with the SIMOTION P320-4 device.

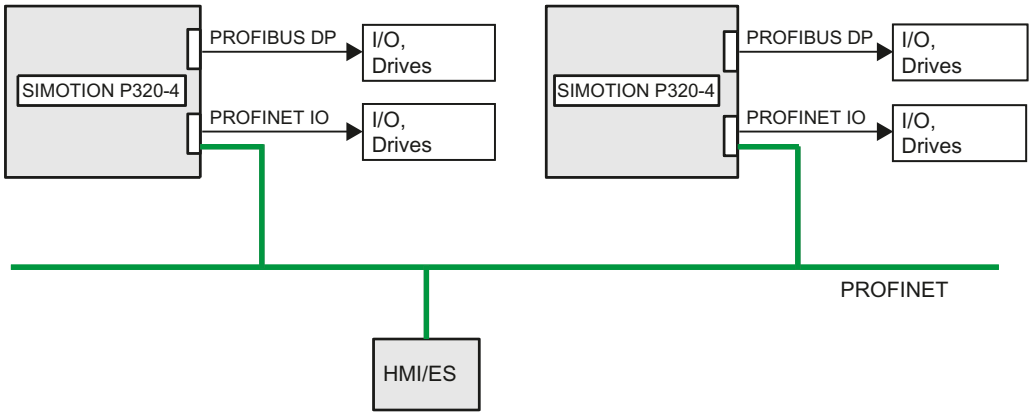


Figure 11-153 Model: Networking via PROFINET

References

You can find further information on the HMI in the documentation:

- SIMOTION Runtime Basic Functions, Section HMI (Human Machine Interface) coupling.
- SIMOTION SCOUT, Configuration Manual
- SIMOTION SCOUT TIA, Configuration Manual
- SIMOTION SCOUT TIA Device Proxy, Configuration Manual

HMI or ES via Ethernet

Networking via Ethernet

A complex interconnection with several HMIs or ESs is only possible using an Ethernet communication. This allows both an external HMI or ES to access several SIMOTION devices or one SIMOTION device to access another one, for example, to display the production data.

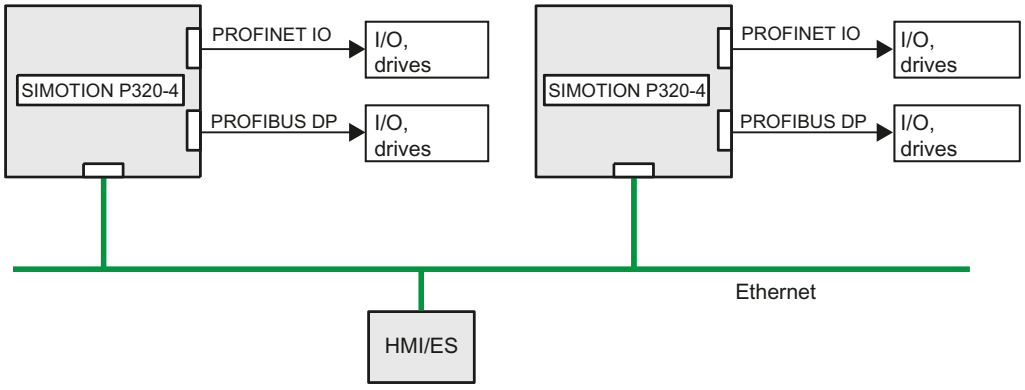


Figure 11-154 Model: Networking via Ethernet

The following services are possible on a SIMOTION device via Ethernet:

- HMI software
  WinCC flexible or the OPC server can use Ethernet to access one or more SIMOTION devices.
- SIMOTION SCOUT Engineering System
  SIMOTION SCOUT can also use Ethernet to access one or more SIMOTION devices.
- SIMOTION IT
  SIMOTION P320-4 offers communication with standard IT protocols (HTTP) over the integrated Ethernet interface.
  This makes it possible to access data or diagnostic information in the SIMOTION P320-4 from any location via intranet or the Internet.
- HTTP (browser, OPC XML)
  Setting in WebCFG.xml - default 80
- HTTPS (browser, OPC XML)
  Setting in WebCfg.xml - default 443
- FTP 21
- Telnet

Note

Information on the SIMOTION IT security concept

More detailed information on the SIMOTION IT security concept of HTTP/S, FTP and Telnet access on the Web server can be found in the following documentation:

- SIMOTION IT Diagnostics and Configuration, Diagnostics Manual, Section security concept

References

You can find further information in the documentation for SIMOTION IT:

- SIMOTION IT Diagnostics and Configuration
- SIMOTION IT Programming and Web Services
- SIMOTION IT Virtual Machine and Servlets

HMI or ES via IsoPROFIBUS (optional)

Networking via PROFIBUS DP

If you connect a central HMI via PROFIBUS DP, the X101 and X102 interfaces of the IsoPROFIBUS board are available.

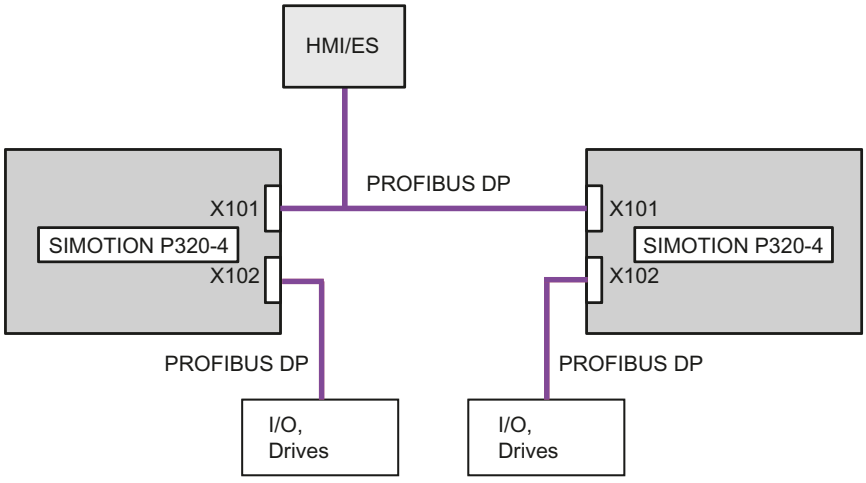


Figure 11-155 Model: Networking via PROFIBUS DP

References

You can find further information on the HMI in the documentation:

- SIMOTION Runtime Basic Functions, section HMI (Human Machine Interface) coupling.

11.2.1.4 Application planning

Unpacking and checking the delivery

Procedure

1. When accepting a delivery, please check the packaging for visible transport damage.
2. If any transport damage is present at the time of delivery, lodge a complaint at the shipping company in charge. Have the shipper confirm the transport damage immediately.
3. Unpack the device at its installation location.
4. Keep the **original packaging** in case you have to transport the device again.

   Note

   Damage to the device during transport and storage

   If a device is transported or stored without packaging, then shocks, vibrations, pressure and moisture may impact the unprotected device.
   A damaged packaging indicates that ambient conditions have already had a massive impact on the device.

   The device may be damaged.

   Do not dispose of the original packaging. Pack the device for transport and storage in the **original packaging**.

5. Check the contents of the packaging and any accessories you may have ordered for completeness and damage.

6. If the contents of the packaging are incomplete, damaged or do not match your order, inform the responsible delivery service **immediately**.

⚠ WARNING

Electric shock and fire hazard due to damaged device

A damaged device can be under hazardous voltage and trigger a fire in the machine or plant. A damaged device has unpredictable properties and states.

Death or serious injury could occur.

Make sure that the damaged device is not inadvertently installed and put into operation. Label the damaged device and keep it locked away. Send off the device for immediate repair.

NOTICE

Damage from condensation

If the device is subjected to low temperatures or extreme fluctuations in temperature during transportation, for example in cold weather, moisture could build up on or inside the HMI device.

Moisture can result in short-circuits in electrical circuits and damage the device.

In order to prevent damage to the device, proceed as follows:

- Store the device in a dry place.
- Bring the device to room temperature before starting it up.
- Do not expose the device to direct heat radiation from a heating device.
- If condensation develops, wait approximately 12 hours or until the device is completely dry before switching it on.

7. Please keep the enclosed documentation, it belongs to the device. You need the documentation when you commission the device for the first time.
8. Write down the identification data of the device.

Identification data of the device

Based on the identification data, the SIMOTION P320-4 device can be identified when service is required or in case of theft.

You also require the identification data in order to view further information on your SIMOTION P320-4 in the **Product Equipment Data** (PED) database.
All the important components of your SIMOTION P320-4 are stored in the PED. The entire history of the device is stored there, including the service reports.
To identify yourself in the PED database, enter the article number and the serial number of your device.

Link to the PED database

The Product Equipment Data database is available on the Internet via the following link (http://www.siemens.com/ped).

Identification data

Enter the identification data in the following table:

Identification data	Source	Value
Serial number	Rating plate	S VP ...
Article number of the device	Rating plate	6AU1320-...
Microsoft Windows Product Key Certificate of Authenticity (COA)	Rear of the device	Only devices with preinstalled Windows operating systems have COA labels
Ethernet address 1	BIOS setup, menu "Advanced", "Peripheral Configuration"	Input by the user.
PROFINET onboard MAC Address Layer 2		
PROFINET onboard MAC Address PROFINET		

SIMOTION P320-4 rating plate

Example of SIMOTION P320-4 S (standard)

You will find the following data on the rating plate of the SIMOTION P320-4:

- Device designation (here): SIMOTION P320-4 S
- Article number of the device (here): 6AU1320-4DS66-3AG0
- Serial number S VP<...>
  The computing unit is uniquely identifiable with the serial number at S VP <...>.
  <...> is an 8-digit character string.
- cULus Approval
- CE marking

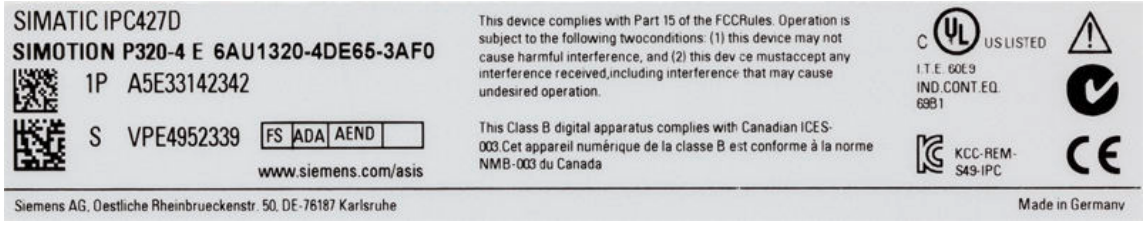


Figure 11-156 SIMOTION P320-4 rating plate

MAC addresses

Example of SIMOTION P320-4 S (standard)

In addition to the data that is found on the SIMOTION P320-4 rating plate, the MAC address plate also contains the **Onboard MAC addresses** for:

- ETHERNET (LAN) X1: VIPCTYSMAC1
- PROFINET (LAN) X3: VIPCTYSPROFINET

The data cited here is not real data.

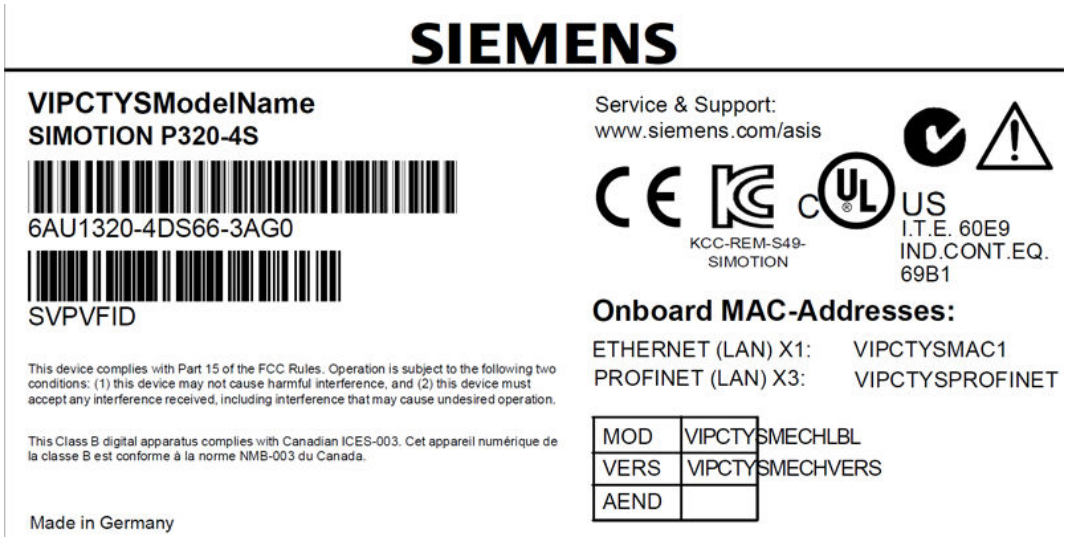


Figure 11-157 SIMOTION P320-4 S rating plate - MAC addresses

Example of a COA label

Microsoft Windows "Product Key" of the "Certificate of Authenticity" (COA):
The COA label is attached to the rear of all devices containing a Windows Embedded Standard 7 or Windows 7 operating system.

- COA label of a device with Windows Embedded Standard 7 32-bit operating system

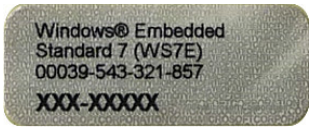
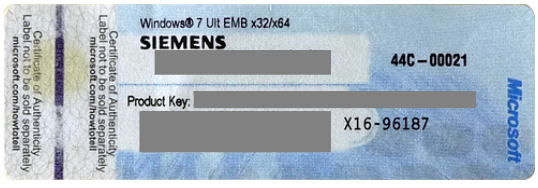


Figure 11-158 ipc477d label coa 01

- COA label of a device with Windows 7 operating system



Permissible mounting positions

Information on the permissible mounting positions for the SIMOTION P320-4 can be found in the section Installation/mounting > Permissible mounting positions (Page 8073).

Environmental conditions

Operating conditions

The SIMOTION P320-4 is designed for use in stationary, weather-protected locations.
The operating conditions exceed requirements according to EN 61131-2.

Observe the following points during installation:

- Note the climatic and mechanical environmental conditions specified in Section Technical specifications (Page 8094).
- The device is designed for use in a normal industrial environment. Without additional protective measures (such as the provision of clean air), the SIMOTION P320-4 computing unit cannot be used in places with severe operating environments, for example, locations with corrosive vapors or gases.
- The clearance in the area of the ventilation slots must be at least 100 mm, so that the SIMOTION P320-4 receives sufficient ventilation.
- Do not cover the ventilation slots of the device.
- The DC power supply does not meet requirements according to EN 60950-1 in the area of the power unit connection. As such, SIMOTION P320-4 must be installed so that it is part of an operating area with restricted access (e.g. a locked control cabinet, console or server room).
- Always observe the mounting positions permitted for this device.
- The connected or installed I/O should not generate a reverse voltage greater than 0.5 V in the device.

⚠ WARNING

Invalidation of the approvals

Failure to adhere to these conditions when installing the system will invalidate approvals according to UL 60950-1, UL 508, and EN 60950-1!

Use prohibition

Without additional measures, the SIMOTION P320-4 should not be used in

- Locations with a high percentage of ionizing radiation
- Aggressive environments characterized, for example, by:
  - Dust accumulation
  - Corrosive vapors or gases
- at locations outside of the prescribed ambient conditions
- Installations requiring special monitoring such as:
  - Elevator installations
  - Electrical plants in particularly hazardous areas

An additional measure for using the SIMOTION P320-4 could be its installation in a cabinet.

Ambient conditions

For the mechanical and climatic environmental conditions, refer to the technical specifications in the section Ambient conditions (Page 8097).

Electromagnetic compatibility

Definition

Standards for electromagnetic compatibility (EMC) are fulfilled if the EMC Installation Guideline is complied with.

See also:

Further information is provided in the Appendix, Section General regulations (Page 8108).

11.2.1.5 Interfaces

Hardware components of the SIMOTION P320-4

Note

The article numbers of the components listed below can be obtained from the online catalog in the Siemens Industry Mall (http://www.siemens.com/industrymall).

SIMOTION P320-4 complete system

The SIMOTION P320-4 device is supplied with the integrated PROFINET onboard communication module.

An additional IsoPROFIBUS board can be inserted as required.

SIMOTION P320-4 spare parts

The SIMOTION P320-4 consists of the following hardware components:

- **SIMOTION P320-4** computing unit
- **Back-up battery** for the motherboard of the SIMOTION P320-4
- IsoPROFIBUS board (optional communication module for the SIMOTION P320-4)

Operation and display

The following hardware components for operation and display can be used:

- SIMATIC Industrial Flat Panel, in the variants IFP1500, IFP1900 or IFP2200.

Power supply of the SIMOTION P320-4

The power supply must satisfy the requirements as described in section Connecting the power supply (24 VDC) (Page 8085). This can be ensured by using the following devices, for example:

- SITOP smart 24 V / 10 A
- 24 VDC UPS (optional)

Drives and I/O modules

The drives and I/O modules released for use with SIMOTION are specified on the website: www.siemens.com/simotion (www.siemens.com/simotion)

Note

Please note that not all modules of the I/O or I/O systems mentioned are approved for SIMOTION. Moreover, system-related functional differences can occur when these I/O or I/O systems are used on SIMOTION rather than SIMATIC. For example, special process-control functions (hot swapping, etc.) are not supported by SIMOTION for the ET 200M distributed I/O system.

A detailed and routinely updated list of I/O modules released for SIMOTION as well as instructions on their use is provided on the Internet at Siemens Product Support (https://support.industry.siemens.com/cs/ww/en/view/11886029).

In addition to the I/O modules approved for SIMOTION, all certified standard slaves can, in principle, be connected to SIMOTION provided that they support the following data traffic:

- Cyclic data traffic (DP V0)
- Acyclic data traffic (DP V1)
- Isochronous data traffic (DP V2)

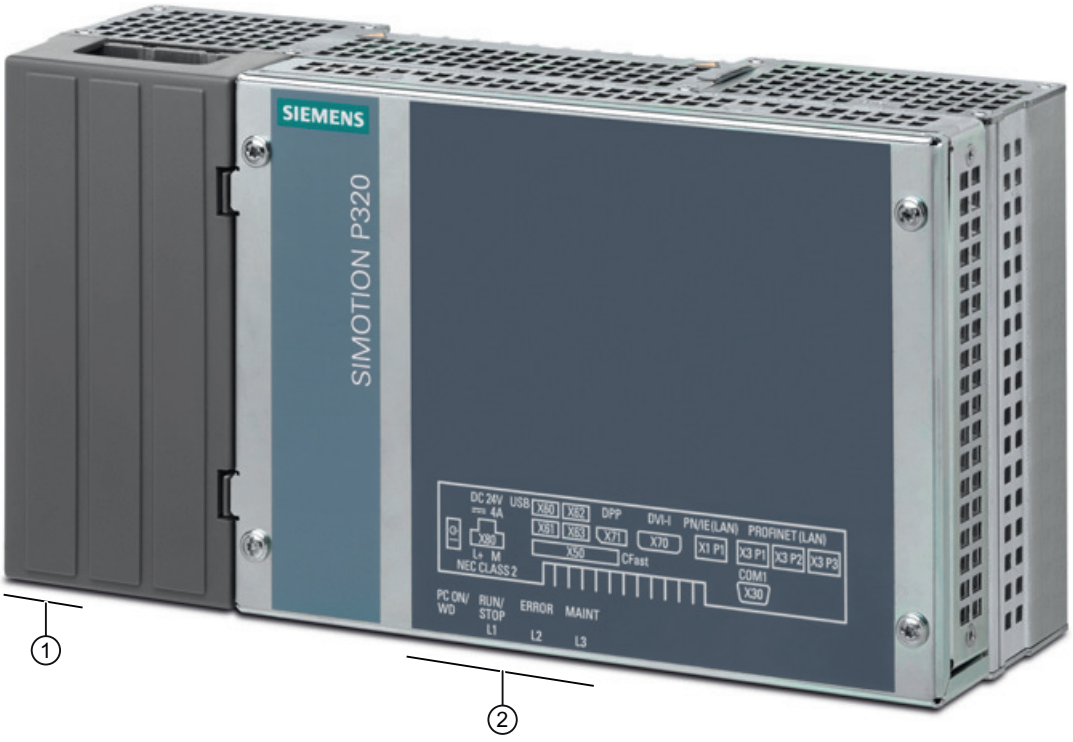
These modules are integrated via the GSD file of the device's manufacturer.

Note

Please note that in individual cases, additional boundary conditions must be fulfilled in order to integrate a standard slave into SIMOTION.

Overview of operator control and display elements

The following figure shows the arrangement of the indication and control elements of the SIMOTION P320-4.



① On/Off switch
② Operating displays (LEDs)

Figure 11-159 SIMOTION P320-4 control and display elements

Control elements

On/Off switch

① **Requirement**:
The BIOS setup entry **After Power Failure** is set to **Power on**.
The On/Off switch is used to switch the SIMOTION P320-4 on.

Figure 11-160 SIMOTION P320-4 - operator controls

⚠ WARNING
De-energize the SIMOTION P320-4! The On/Off switch does not disconnect the device from the power supply system. When the switch is in position **0** (Off), the device is still supplied with mains voltage. To remove power from the device, the power supply plug must be removed.

NOTICE
Close down the operating system first First close down the operating system before switching off the device with the on/off switch, otherwise data may get lost.

Status displays

The status indicators below the interfaces are described in the following table.

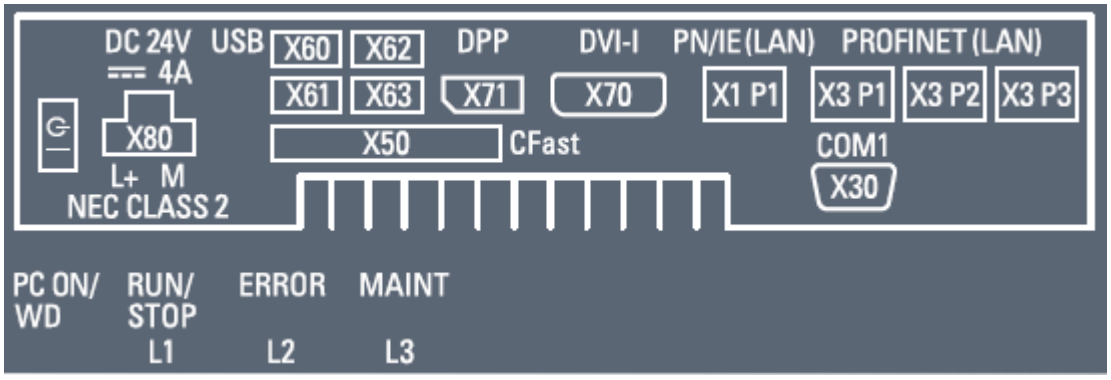


Figure 11-161 SIMOTION P320-4 status indicator

LED	Meaning	State	Description
PC ON/ WD	Power supply	Green	BIOS ready to boot. Indicates the correct supply voltage of 3.3 V, 5 V and 12 V from the integrated power unit.
		Off	Not relevant.
		Green/yellow flashing (1 Hz)	BIOS in POST, power switch on.
		Yellow	Idle state
RUN/STOP or L1	RUN STOP	Off	SIMOTION P Runtime is not running.
		Green	SIMOTION P320-4 in RUN.
		Yellow	SIMOTION P320-4 in STOP.
		Green/yellow: Flashes quickly (2 Hz)	DCP flashing
		Yellow: Flashes quickly (5 Hz)	FAULT state (only in combination with L2 and L3)

LED	Meaning	State	Description
ERROR or L2 (SF PROFINET LED)	-	Off	SIMOTION P320-4 is working error-free.
	Error	Red: Continuous light	Bus error at the PROFINET interface.
		Red: Flashes quickly (2 Hz)	Erroneous configuration.
		Red: Flashes quickly (5 Hz)	FAULT state (only in combination with L1 und L3)
	INIT	Red: Flashes slowly (0.5 Hz)	During the INIT process, the SF PROFINET LED is controlled by the boot loader. The precise flashing behavior therefore cannot be determined and displayed. In the INIT state, 0.5 Hz flashing is displayed in the SIMOTION P State. If no project has yet been loaded, 0.5 Hz flashing is also displayed for the INIT, which, in this case, is also retained in the Running state.
MAINT or L3 (SF LED)	-	Off	SIMOTION P320-4 works error-free.
	Error	Red	An event has occurred which needs to be acknowledged (alarm, message, notification). (See Diagnostics Guide)
		Red: Flashes slowly (0.5 Hz)	License missing for licensed technology objects. Carry out the licensing to correct the error. Further information on the licensing can be found in the SIMOTION P320-4 E / SIMOTION P320-4 S Commissioning and Hardware Installation Manual.
		Red: Flashes quickly (5 Hz)	FAULT state (only in combination with L1 and L2) A fault has occurred to which the user program cannot respond. The following actions may be required to rectify the fault: • Power OFF/ON • Recommissioning

Overview of the SIMOTION P320-4 interfaces

The following table provides an overview of the SIMOTION P320-4 interfaces.

Table 11-20 Overview of the interfaces

Interface	Description	Further information
PROFINET	Three PROFINET ports with RJ45 socket	PROFINET onboard interface (Page 8067)
Ethernet	1 x 8-pole RJ45 interface	Ethernet RJ45 interface (Page 8067)
DVI-I	26-pin socket for connection of a CRT or LCD monitor with DVI interface or VGA with DVI/VGA adapter	DVI-I interface (Page 8068)
Display port	20-pole DPP interface	Display port interface (Page 8069)
USB 3.0	Lower USB channel 0, upper USB channel 1 Lower USB channel 2, upper USB channel 3	USB 3.0 interface (Page 8070)

Interface	Description	Further information
CFast card	Slot for CFast card 50-pin CF socket, types I/II	CFast card (Page 8071)
COM1	Serial interface *V.24*	Serial interface COM1 (Page 8072)
IsoPROFIBUS board (optional)	2 x 9-pin socket	IsoPROFIBUS board (optional) (Page 8072)

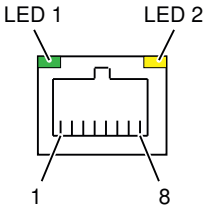
PROFINET onboard

Interface assignment

Designation: PROFINET LAN, X3, 3 ports: P1, P2, P3

Type: 8-pin RJ45 socket

PROFINET onboard LAN (X3) interface



Pin no.	Short description	Meaning	Input/output
1	RD+	Receive data [2]	Input
2	RD-	Receive data [2]	Input
3	TD+	Send data [2]	Output
4, 5 [1]	SYMR	Internal 75 Ohm terminating resistor	-
6	TD-	Receive data [2]	Output
7, 8 [1]	SYMT-	Internal 75 Ohm terminating resistor	-
S	-	Shield	-
-	LED 1	Lights up green: Link	-
-	LED 2	Lights up yellow: Activity	-

[1] Optional product feature

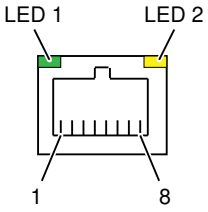
[2] Autonegotiation and autocrossover are supported.

Ethernet

Interface assignment

Designation: PN/IND. Ethernet (LAN), X1

Type: 8-pin RJ45 socket

Ethernet RJ45 interface			
			
Pin no.	**Short description**	**Meaning**	**Input/output**
1	BI_DA+	Bi-directional data A+	Input/output
2	BI_DA-	Bi-directional data A-	Input/output
3	BI_DB+	Bi-directional data B+	Input/output
4	BI_DC+	Bi-directional data C+	Input/output
5	BI_DC-	Bi-directional data C-	Input/output
6	BI_DB-	Bi-directional data B-	Input/output
7	BI_DD+	Bi-directional data D+	Input/output
8	BI_DD-	Bi-directional data D-	Input/output
S	–	Shield	–
	LED 1	Off: 10 Mbps Green light: 100 Mbps Orange light: 1000 Mbps	–
	LED 2	Lit: Active connection, e.g., to a hub Flashing: Activity	–

Note

The interfaces available on the device have been numbered for unique identification.

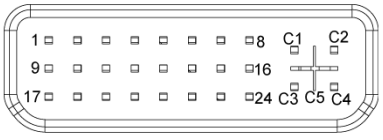
This numbering may deviate, however, from the numbering assigned by the operating system.

DVI-I

Interface assignment

Designation: DVI/VGA, X70

Type: 26-pin socket

DVI-I interface (standard socket)			
			
Pin no.	**Short description**	**Meaning**	**Input/output**
1	TMDS Data2-	DVI data channel	Output

DVI-I interface (standard socket)			
2	TMDS Data2+	DVI data channel	Output
3	TMDS Data2/4 shield	Cable shield	-
4	NC	-	-
5	NC	-	-
6	DDC clock (SCL)	Display Data Channel – Clock	Input/output
7	DDC data (SDA)	Display Data Channel – Data	Input/output
8	Analog vertical sync (VSYNC)	Analog Vertical Sync Signal	Output
9	TMDS Data1-	DVI data channel	Output
10	TMDS Data1+	DVI data channel	Output
11	TMDS Data1/3 shield	Cable shield	-
12	NC	-	-
13	NC	-	-
14	+5 V power (VCC)	+5 V power for DCC	Output
15	Ground (return for +5 V, Hsync and Vsync) (GND)	Analog ground	-
16	Hot Plug Detect	-	-
17	TMDS data 0-	DVI data channel	Output
18	TMDS data 0+	DVI data channel	Output
19	TMDS Data0/5 shield	Cable shield	-
20	NC	-	-
21	NC	-	-
22	TMDS clock shield	Cable shield	-
23	TMDS clock+	DVI clock channel	Output
24	TMDS clock-	DVI clock channel	Output
C1	Analog red (R)	Analog red signal	Output
C2	Analog green (G)	Analog green signal	Output
C3	Analog blue (B)	Analog blue signal	Output
C4	Analog horizontal sync (HSYNC)	Analog horizontal sSync signal	Output
C5	Analog ground (analog R, G, & return) (GND)	Analog ground	-

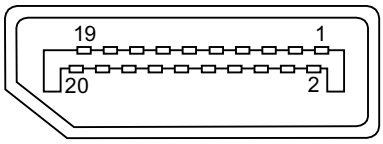
DisplayPort

Interface assignment

Designation: Display port interface, X71

Type: 20-pole DPP interface

Display port interface



Pin no.	Short designation	Meaning	Input/output
1	ML_Lane0+	DP data 0+	Output
2	GND	Ground	-
3	ML_Lane0-	DP data 0-	Output
4	ML_Lane1+	DP data 1+	Output
5	GND	Ground	-
6	ML_Lane1-	DP data 1-	Output
7	ML_Lane2+	DP data 2+	Output
8	GND	Ground	-
9	ML_Lane2-	DP data 2-	Output
10	ML_Lane3+	DP data 3+	Output
11	GND	Ground	-
12	ML_Lane3-	DP data 3-	Output
13	CONFIG1 CAD	Cable Adapter Detect	Input
14	CONFIG2	Ground (PullDown)	-
15	AUX_CH+	Auxiliary channel+	Bidirectional
16	GND	Ground	-
17	AUX_CH-	Auxiliary channel-	Bidirectional
18	HPD	Hot Plug Detect	Input
19	GND	Ground	-
20	DP_PWR	+3.3V (fused)	Output

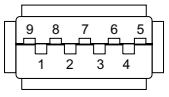
USB 3.0

Interface assignment

Designation: USB X62, X61, X62, X63

Type: 9-channel *high current*

USB 3.0 interface



Pin no.	Short designation	Meaning	Input/output
1	VBUS	+ 5 V (fused)	Output
2	D-	Data line	Input/output
3	D+	Data line	Input/output

USB 3.0 interface			
4	GND	Ground	–
5	RX-	Data line	Input
6	RX+	Data line	Input
7	GND	Ground	–
8	TX-	Data line	Output
9	TX+	Data line	Output

CFast card

Pin assignment of the interface

Designation: CFast X50

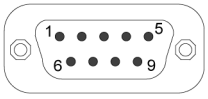
Pin no.	Short designation	Meaning
S1	SGND	Signal GND (ground for signal integrity)
S2	A+	SATA differential
S3	A-	SATA differential
S4	SGND	Signal GND (ground for signal integrity)
S5	B-	SATA differential
S6	B+	SATA differential
S7	SGND	Signal GND (ground for signal integrity)
PC1	CDI	Card Detect Out
PC2	GND	Device GND
PC3	TBD	TBD (not connected)
PC4	TBD	TBD (not connected)
PC5	TBD	TBD (not connected)
PC6	TBD	TBD (not connected)
PC7	GND	Device GND
PC8	LED1	LED Output (not connected)
PC9	LED2	LED Output (not connected)
PC10	IO1	Reserved Input/Output (not connected)
PC11	IO2	Reserved Input/Output (not connected)
PC12	IO3	Reserved Input/Output (not connected)
PC13	PWR	Device Power (3.3V)
PC14	PWR	Device Power (3.3V)
PC15	GND	Device GND
PC16	GND	Device GND
PC17	CDO	Card Detect In

COM1

Interface assignment

Designation: COM1, X30

Type: 9-pin (connector)

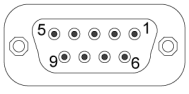
Serial interface COM1			
			
Pin no.	**Short designation**	**Meaning**	**Input/output**
1	DCD	Data carrier detect	Input
2	RxD	Receive data	Input
3	TxD	Transmit data	Output
4	DTR	Data terminal ready	Output
5	GND	Ground	-
6	DSR	Data set ready	Input
7	RTS	Request to send	Output
8	CTS	Clear to send	Input
9	RI	Incoming call	Input

IsoPROFIBUS board (optional)

Interface assignment

Designation: PROFIBUS, X101, X102

Type: 9-pin (socket)

PROFIBUS interface		
		
Pin no.	**Short designation**	**Meaning**
1-2	NC	Not connected
3	LTG_B	Data line (I/O)
4	RTS_AS	AS request to send (O)
5	GND	Ground electrically isolated
6	P5V_dp_fused	+5 V / max. 90 mA (fused) electrically isolated
7	NC	Not connected
8	LTG_A	Data line (I/O)
9	RTS_PG	PG request to send (O)

11.2.1.6 Installation/mounting

Installing the device

SIMOTION P320-4 is particularly suitable for installation in consoles, control cabinets and switchboards.

⚠ WARNING

Perform function test while installing the device in machines or plants

Following the results of a risk analysis, additional protection equipment on the machine or the system is necessary to avoid endangering persons. With this, especially the programming, configuration and wiring of the inserted I/O modules have to be executed, in accordance with the necessary risk analysis identified safety performance (SIL, PL or Cat.). The intended use of the device has to be secured.

The correct use of the device has to be verified with a function test on the system. This test can detect programming, configuration and wiring errors. The test results have to be documented and if necessary inserted into the relevant inputs.

Permissible mounting positions

Note

The SIMOTION P320-4 is approved for operation in closed rooms only.

Ensure the required minimum clearance to other components or enclosure panels:

- Below at least 100 mm
- Above at least 50 mm

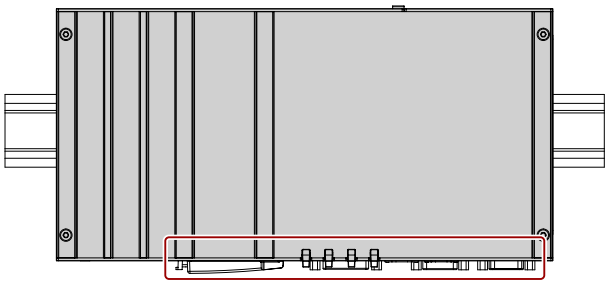
The permissible mounting positions for the SIMOTION P320-4 are standard rail mounting and vertical mounting.

Standard rail mounting

Horizontal mounting is the preferred position.

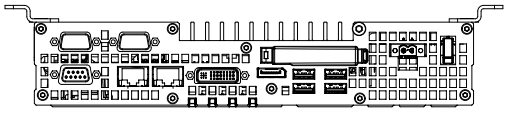
The following horizontal mounting option is permitted:

The interfaces are at the bottom.



The figure shows the interfaces on the front for representation purposes only.

For rail mounting, the interfaces always point downward.

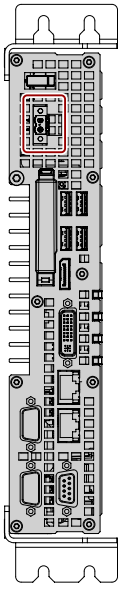


You can find mounting information under Standard rail mounting (Page 8075).

Vertical mounting

The following vertical mounting option is permitted:

The interfaces are at the front. The power supply connection is at the top.



You can find information on mounting under Vertical mounting (Page 8076).

Ambient conditions

Observe the permissible temperature range for operation in the respective mounting position as described in Section "Technical data > Ambient conditions (Page 8097)".

Standard rail mounting and vertical mounting

Standard rail mounting

Requirement

- Standard mounting rail, 35 mm standard profile
  The standard mounting rail is installed at the installation site.
- The standard mounting rail clip is attached to the SIMOTION P320-4.

Note

Ensure that the wall or ceiling can hold four times the total weight of the SIMOTION P320-4 including the standard mounting rail and additional expansion cards. See section Notes on installation (Page 8077).

Mounting the standard mounting rail clip

Before you can install the device on a standard mounting rail, you will need to attach the standard mounting rail clip included in the scope of delivery.

The fasteners and screws required are supplied with the device for mounting depending on the order number.

Requirement

- 1 standard mounting rail clip
- 2 screws
- 1 T20 screwdriver

Use the two screws to mount the standard mounting rail clip on the SIMOTION P320-4 enclosure.

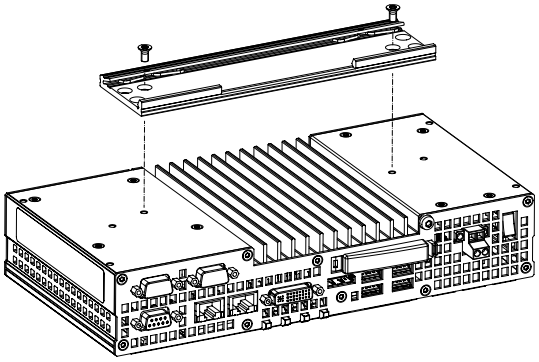
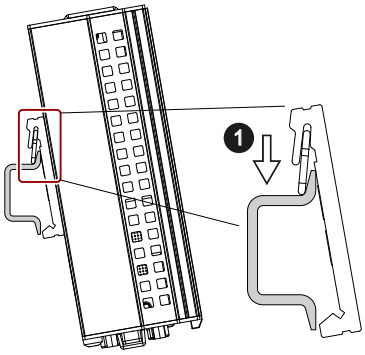


Figure 11-162 Mounting the standard mounting rail clip

Mounting

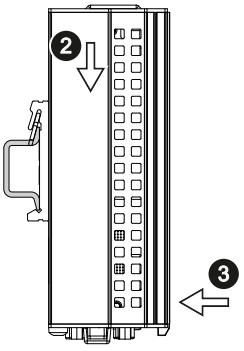
1. Place the SIMOTION P320-4 and rail clip on the upper edge of the standard profile rail at the position shown.



2. Press the SIMOTION P320-4 slightly downward.

3. Once the rail clip slides over the bottom edge of the standard mounting rail, push the device onto the rail.

   Ensure that you hear the SIMOTION P320-4 snapping into place.



Note

To ensure secure vertical mounting on standard mounting rails, install a standard rail ground terminal beneath the device.

Removing

1. Press the SIMOTION P320-4 down until the rail clip releases the device.
2. Swing the SIMOTION P320-4 away from the standard mounting rail.
3. Lift the device up and off.

Vertical mounting

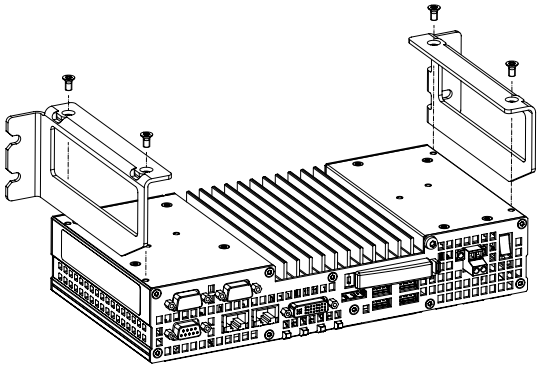
The fasteners and screws required are supplied with the device for the selected mounting option.

Requirement

- 2 mounting brackets
- 4 screws
- 1 T20 screwdriver

Procedure

1. Secure each mounting bracket with two screws.



Installation notes

Note the following:

- The SIMOTION P320-4 is approved for operation in closed rooms only.
- For installation in a control cabinet, observe the SIMATIC installation guidelines and applicable DIN/VDE requirements or other applicable country-specific regulations.
- When the device is used in the area of Industrial Control Equipment according to UL508, note that the device is classified as "Open Type". The installation of the device in a housing according to UL508 is therefore a mandatory requirement for approval or operation in according to UL508.

Securely fastening the SIMOTION P320-4

NOTICE
Insufficient load carrying capacity If the wall it is mounted on does not have a sufficient load-bearing capacity, the device may fall and be damaged. Ensure that the mounting surface on the wall can bear four times the total weight of the device, including fixing elements.

NOTICE
Incorrect fixing elements The device may not be securely fitted if you use dowels and screws other than those specified below for mounting. The device can fall and may be damaged. Use only the dowels and screws specified in the following table.

Connecting elements

The connecting elements for each mounting position are listed in the following:

- Standard rail mounting

Material	Bore diameter	Fastener
Metal, min. 2 mm thickness	5 mm	• 2 x M4 screws • 2 x M4 nuts
Concrete	6 mm, 40 mm depth	• 4 x dowels, ∅ 6 mm, 40 mm length • 4 x screws, ∅ 4 mm, 40 mm length
Plasterboard, min. 13 mm thickness	14 mm	4 x toggle bolts, ∅ 4 mm, 50 mm length

- Vertical mounting

Material	Bore diameter	Fastener
Concrete	6 mm, 40 mm depth 8 mm, 40 mm depth	• 4 x dowels, ∅ 6 mm, 40 mm length • 4 x screws, ∅ 4 mm, 40 mm length • Dowel, ∅ 8 mm, 40 mm length • Screw, ∅ 5 mm, 40 mm length
Plasterboard, min. 13 mm thickness	14 mm	Toggle bolt, ∅ 4 mm, 50 mm length

Mounting positions

The following mounting positions are permitted for the SIMOTION P320-4 and are described in detail in the following sections:

- Standard rail mounting (Page 8075)
- Vertical mounting (Page 8076)

Overview of operating modes for the SIMOTION P320-4

Overview

The following operating modes are available for mounting the SIMOTION P320-4.

The operating modes are:

- Decentralized structure
  The SIMOTION P320-4 is mounted separately from a panel in a decentralized structure.
- Headless operation
  The SIMOTION P320-4 is operated without panel.

Before mounting, the following must be noted:

Before you mount the SIMOTION P320-4 in a control cabinet or a rack, optional components are installed to expand the SIMOTION P320-4.

Example: installation of the optional IsoPROFIBUS board.

To do this, open the device and note the following procedure:

1. Open the cover of the SIMOTION P320-4.
2. Insert the optional IsoPROFIBUS board.
3. Plug in the SYNC I/O cable for synchronizing with PROFINET.
4. Close the cover.
5. Mount the SIMOTION P320-4.

Note

The installation and removal of components as well as the mounting of the SIMOTION P320-4 are described in detail in the SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual.

NOTICE
Avoid loss of warranty If you install or exchange system expansions and damage your device, the warranty becomes void.

Decentralized structure

Overview

The decentralized structure provides greater flexibility in terms of location and enables the SIMOTION P320-4 to be positioned in non-critical areas of the plant (e.g. control cabinet).

The decentralized structure of a panel with the SIMOTION P320-4 can be performed via the display port, USB or DVI connection. Please note the cable lengths that are supported for the panel.

For the use of an HMI system, the SIMATIC IFP, for example, can be connected by means of the described interfaces.

11.2.1.7 Connection

Requirements

General information

During the configuration of SIMOTION modules, you must pay attention to the "Electrical configuration design".

The following section provides information on wiring and networking the complete SIMOTION system.

Further information about the connection of the complete system and the individual components can be found in the SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual.

Open equipment

These modules are open equipment. This means they may only be installed in housings, cabinets, or in electrical equipment rooms that can only be entered or accessed with a key or tool. Housings, cabinets, or electrical equipment rooms may only be accessed by trained or authorized personnel. An external fire-protection housing is required.

Observe the following points during installation:

- The device should be connected only to **24 VDC** power supply networks that meet the requirements of **safety extra-low voltage** (**SELV**). The cable cross-section must be chosen large enough to ensure that no damage can result from cable overheating in the event of a short-circuit in the SIMOTION P320-4.
- Avoid extreme ambient conditions.
- Protect the device against dust, moisture, heat and severe vibration.
- Do not subject the SIMOTION P320-4 to direct sunlight.
- Install the device in such a way that it poses no danger, for example, by falling over.
- The clearance around the SIMOTION P320-4 must be at least 100 mm to ensure adequate ventilation.
- Do not cover the ventilation slots.

Overview of connections

The connection overview shows the SIMOTION P320-4 interfaces to which the appropriate I/O devices can be connected.

Note

The cabling for all elements and components of the complete system can only be carried out when disconnected from the mains.

It does not matter in which order connection takes place.

The connection of the individual components in the SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual.

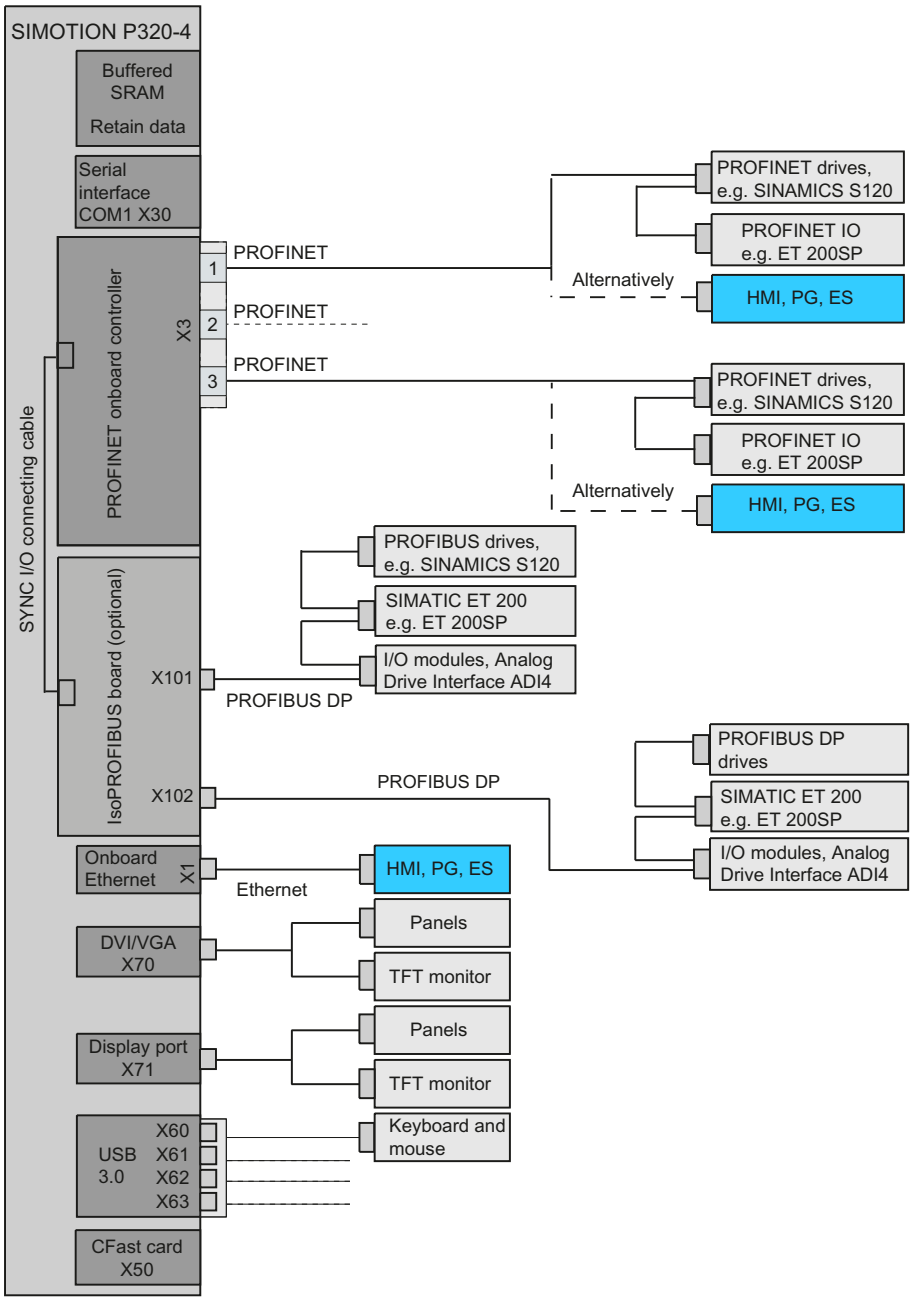


Figure 11-163 Hardware system overview - example with PROFINET onboard and optional IsoPROFIBUS board

Notes on connecting

⚠ WARNING

Risk of fire and electric shock

The On/Off switch does not isolate the device from the power supply. Risk of electric shock if the device is opened incorrectly or defective. There is also a risk of fire if the device or connecting lines are damaged.

You should therefore protect the device as follows:

- Always pull out the power plug when you are not using the device or if the device is defective. The power plug must be freely accessible.
- Connect the device to a protective conductor as instructed, see Section "Connecting the protective conductor".
- Use a central isolating switch in the case of cabinet installation.

⚠ WARNING

Risk of lightning strikes

A lightning strike may enter the mains cables and data transmission cables and reach to a person.

Death, serious injury and burns can be caused by lightning.

Take the following precautions:

- Disconnect the device from the power supply in good time when a thunderstorm is approaching.
- Do not touch mains cables and data transmission cables during a thunderstorm.
- Keep a sufficient distance from electric cables, distributors, systems, etc.

NOTICE

Fault caused by I/O devices

The connection of I/O devices can cause faults in the device.

The result may be personal injury and damage to the machine or plant.

Note the following when connecting I/O devices:

- Read the documentation of the I/O devices. Follow all instructions in the documentation.
- Only connect I/O devices which are approved for industrial applications in accordance with EN 61000-6-2 and IEC 61000-6-2.
- I/O devices that are not hotplug-capable may only be connected after the device has been disconnected from the power supply.

NOTICE

Damage through regenerative feedback

Regenerative feedback of voltage to ground by a connected or installed component can damage the device.

Connected or built-in I/Os, for example, a USB drive, are not permitted to supply any voltage to the device. Regenerative feedback is generally not permitted.

Protective conductor connection and potential equalization

The protective conductor connection is needed to protect the device and helps ensure that interference signals generated by power lines, signal lines or lines to I/O devices are safely discharged to earth.

The protective conductor connection on the device must be connected to the protective conductor of the control cabinet or system in which the device is installed.

⚠ WARNING

Electrical shock hazard and risk of fire

Internal components of a faulty device may carry dangerous voltages that pose the risk of fire or electrical shock. Risk of death and serious injury.

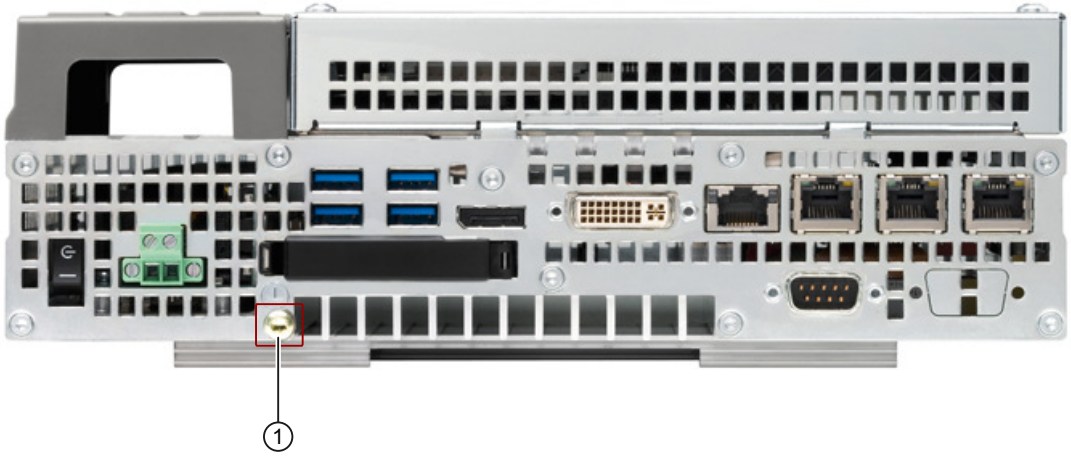
- Always connect the ground conductor before you commission the device.
- Never operate the device without protective conductor.
- Take a faulty device out of service immediately and mark it accordingly.

Requirement

- The device has been installed.
- 1 protective conductor, minimum cross section 2.5 mm^2
- 1 T20 screwdriver
- 1 x M4 cable lug

Procedure

1. Crimp the cable lug onto the protective conductor.
2. Screw the cable lug onto the protective conductor connection as shown.



① Protective conductor connection / potential equalization

3. Wire the protective conductor to the protective conductor connection of the control cabinet in which the device is installed.

Connecting peripheral equipment

Note

Observe suitability for industrial applications

Only connect I/O devices that are suitable for industrial applications in accordance with EN IEC 61000-6-2.

Note

I/O devices capable of hot-plugging (USB)

Hot-plug I/O devices (USB) may be connected while the PC is in operation.

NOTICE
Non-hot-plug I/O devices I/O devices that do not support hot-plugging may not be connected until the device is powered down. Strictly adhere to the specifications for peripheral equipment.

Note

Wait at least ten seconds before you reinsert USB devices.

Note that the EMC immunity of standard USB devices is designed only for office environments. These USB devices are appropriate for handling commissioning and service tasks. You may only use USB devices that are suitable for industrial applications. The USB devices are developed and marketed by the respective supplier. The respective product supplier provides support for the USB devices. The manufacturer's terms of liability shall apply.

Note

The connected or built-in I/Os, such as USB drives, should not introduce a counter EMF into the device.

Reverse voltages exceeding 0.5 V to ground that are generated by connected or installed components may prevent proper operation of the device or lead to its destruction.

Connecting the power supply (24 VDC)

Please note the following to ensure you operate the device safely and in accordance with regulations:

⚠ WARNING

Electrical shock hazard and risk of fire

Voltage exceeding SELV levels may cause fire or electric shock. Death or serious physical injury can result.

- Always wire the device to a 24 VDC power supply that is compliant with SELV requirements.
- You need a corresponding NEC Class 2 power source to comply with requirements to UL 50950-1 and UL 508 when operating the device on a wall, in an open frame, or at any other location.
- In all other cases (IEC / EN / DIN EN 60950-1), either a current source of limited output (LPS = Low Power Source), or a line-side fuse or a line-side circuit breaker is necessary. Current must be limited to 4.16 A. This requires a 4 A fuse max.

The device has reverse polarity protection.

Requirement

- The device has been installed.
- The protective conductor is connected.
- One wired terminal.
  The corresponding 24 VDC power supply is off.
- One 0.5 × 3 Philips screwdriver.

NOTICE
Damage to the device Do not turn the screws of the terminal when the terminal is inserted in the device. The device can be damaged through the pressure of the screwdriver on the terminal and consequently the connection socket. Only connect the wires to the terminal when it is unplugged.

Note

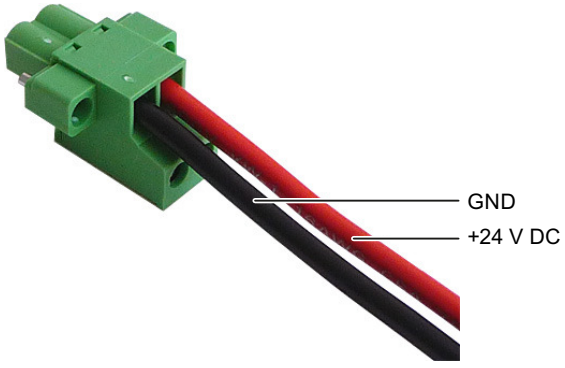
Make sure that the wires are not accidentally swapped over. Note the labeling for the contacts on the front cover of the device.

The terminal to connect the power supply is attached to the device. The terminal is designed for wires with a cross-section of 0.25 mm^2 to 2.5 mm^2. Only connect wires with a cross-section ≥ 0.5 mm^2.

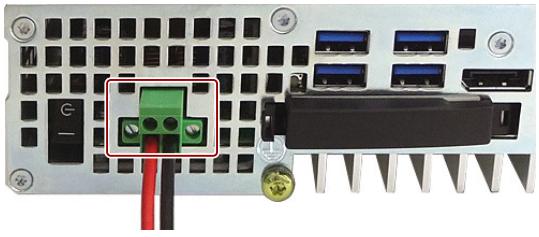
Solid or flexible cables can be used for the connection. Ferrules are not required.

Procedure

1. Connect the cables as shown.
   Note the position of the terminal.



2. Insert the terminal at the marked position.
3. Secure the terminal using the integrated screws.



Connecting the device to networks

Ethernet can be used for integrating the device into existing or planned networks.

Ethernet

The integrated Ethernet interface can be used for communication and for data exchange with automation devices such as SIMOTION SCOUT, SIMATIC S7.

Further information

Further information is available on the Internet at the Siemens Industry Mall (www.siemens.com/industrymall), the catalog and ordering system for automation and drives.

Installing the strain relief

Strain relief for the device is provided in the scope of delivery. The strain relief is designed to prevent the lines connected to the device from being accidentally pulled out.

Attaching PROFINET strain relief

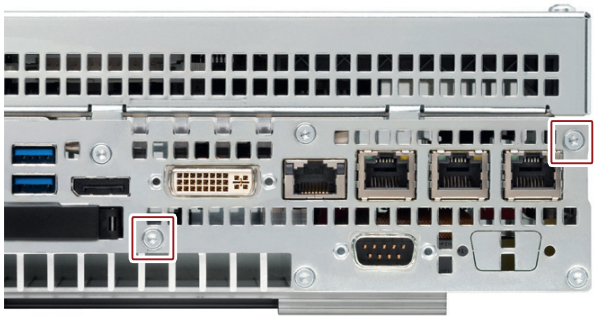
Requirement

- 1 strain relief
- 2 screws
- 1 T10 screwdriver

Mounting

1. Remove the marked countersunk screws.

2. Fasten the strain relief at the PROFINET interface(s).
   Use the oval head screws included with the strain relief.



Removing

Follow the mounting procedure in reverse to remove the strain relief.

Securing cables generally

Secure all cables connected to the device at the cable grip, each with a cable tie. The required cable ties are not part of the scope of delivery.

Requirement

- Maximum of 6 cable ties for USB, Ethernet, PROFIBUS
  Maximum cable tie width 3 mm.
- 1 cutting tool

Procedure

1. Place the cable tie around the attachment plate of the strain-relief assembly and secure the plugged-in cables.
2. Cut off the cable tie overhang.

11.2.1.8 Troubleshooting/FAQs

Error correction

This section provides you with tips on how to locate and troubleshoot common problems.

FAQs / error correction	Possible cause	Possible remedy
The device is not operational.	There is no power supply to the device.	• Check whether the 24 VDC power supply is connected. See Connecting the power supply (24 VDC) (Page 8085). • Check whether the line side switch is activated. See Operator controls (Page 8064).
SIMOTION P does not start after connecting the CFast card.	BIOS settings are not set to Hotplug Enabled.	You can find out how to check and correct the BIOS settings in Section Hotplug Enabled BIOS settings (Page 8090).
SIMOTION P does not start after a reboot.	The user is not logged in automatically.	Set the password for the AutoLogin (Automatically Log On). See Section Changing the AutoLogin > SIMOTION P AutoLogin (Page 8024).
The monitor remains dark.	The monitor is switched off.	Switch on the monitor.
	The monitor is in "power-save" mode.	Press any key on the connected keyboard.
	The brightness button has been set to dark.	Set the monitor brightness button to obtain more light. For detailed information, refer to the monitor operating instructions.
	The power cord or the monitor cable is not connected.	• Check whether the power cord has been properly connected to the monitor and to the system unit or to the grounded shockproof outlet. • Check whether the monitor cable has been properly connected to the system unit and to the monitor.
		If the monitor screen still remains dark after you have performed these checks, please contact your technical support team.
The mouse pointer does not appear on the screen.	The mouse driver is not loaded.	Check whether the mouse driver is properly installed and present when you start the application program. For more detailed information on the mouse driver, refer to the documentation for the mouse or application programs.
	The mouse is not connected.	Check whether the mouse is properly connected to the system unit. If you use an adapter or extension on the mouse cable, also check the connectors.
		If the cursor still does not appear on the screen after you have performed these checks and measures, please contact your technical support team.

FAQs / error correction	Possible cause	Possible remedy
Error message on the screen: Operating system not found	• No operating system present. • Incorrect active boot partition. • Incorrect drive entries in the SETUP.	Check whether one of the specified causes is present:
Wrong time and/or date on the PC.	-	• Press <ESC> during the boot sequence to open BIOS Setup. Detailed information on the BIOS setup can be found in the section Starting Bios settings Hotplug Enabled - BIOS Setup (Page 8090). • Set the time and date in the setup menu.
Although the BIOS setting is OK, the time and date are still wrong.	The backup battery is empty.	Replace the backup battery. See SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual > Service and maintenances > Replacing the backup battery. If the problem cannot be resolved by replacing the battery, then contact your technical support team.
UPS does not function.	The operating system does not have a suitable driver for the UPS.	Install a suitable driver; the correct driver can often be downloaded from the homepage of the device's manufacturer. Note the SIMOTION P320-4 E / P320-4 S Commissioning and Hardware Installation Manual > Power on and software installation > Customer-specific software.
The display in the BIOS for the PROFINET MAC address is incorrect (00:00:00:00:00:00).	-	The correct MAC addresses can be taken from the rating plate on the housing. See also Device identification data (Page 8057), Section MAC addresses.

Hotplug Enabled BIOS settings

Starting the BIOS setup

Proceed as follows:

1. Reset the SIMOTION P320-4 (warm or cold restart).
   Depending on the device version, the default settings may differ from the following figures.
2. After completing the startup test, you can start the program setup with BIOS.
   The following message is shown on the display:

Figure 11-164 SIEMENS Press Esc boot options

Press **Esc**.

3. After booting, the BIOS selection menu is displayed:

Figure 11-165 BIOS selection menu

4. The following buttons are available in the BIOS selection menu:

Button	Function
Continue	Exit BIOS menu, continue with the startup process
Boot Manager	Selection of the boot drive
Boot From File	Starting from a .EFI file
SCU	Device configurator (Setup Configuration Utility)
BIOS Update	BIOS update from USB stick
MEBx	Start the Intel Management Engine BIOS Extension

5. Select the **SCU** device configurator.

6. In the BIOS window, select the **Advanced** tab using the arrow keys and then select **IDE Configuration**.
   Press Enter to confirm the selection.

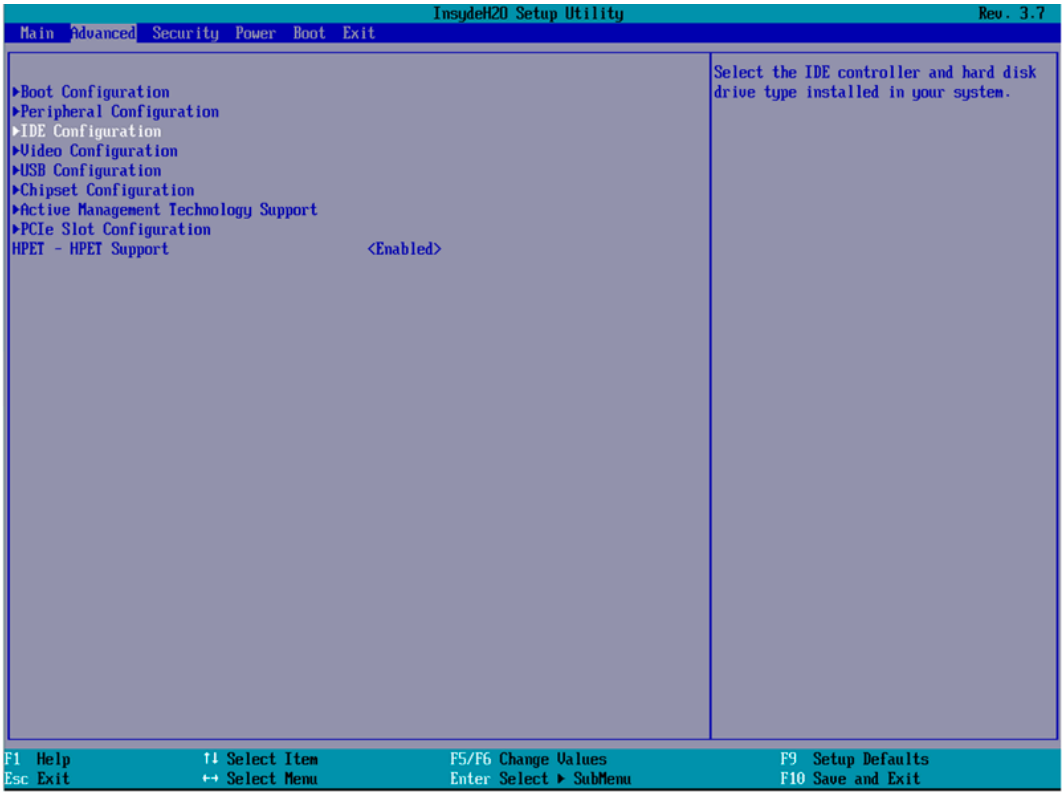


Figure 11-166 Advanced > IDE configuration

7. In the **IDE Configuration** window, select **SATA Port 2 HotPlug**.

8. Change the **Disabled** selection shown here to **Enabled**.

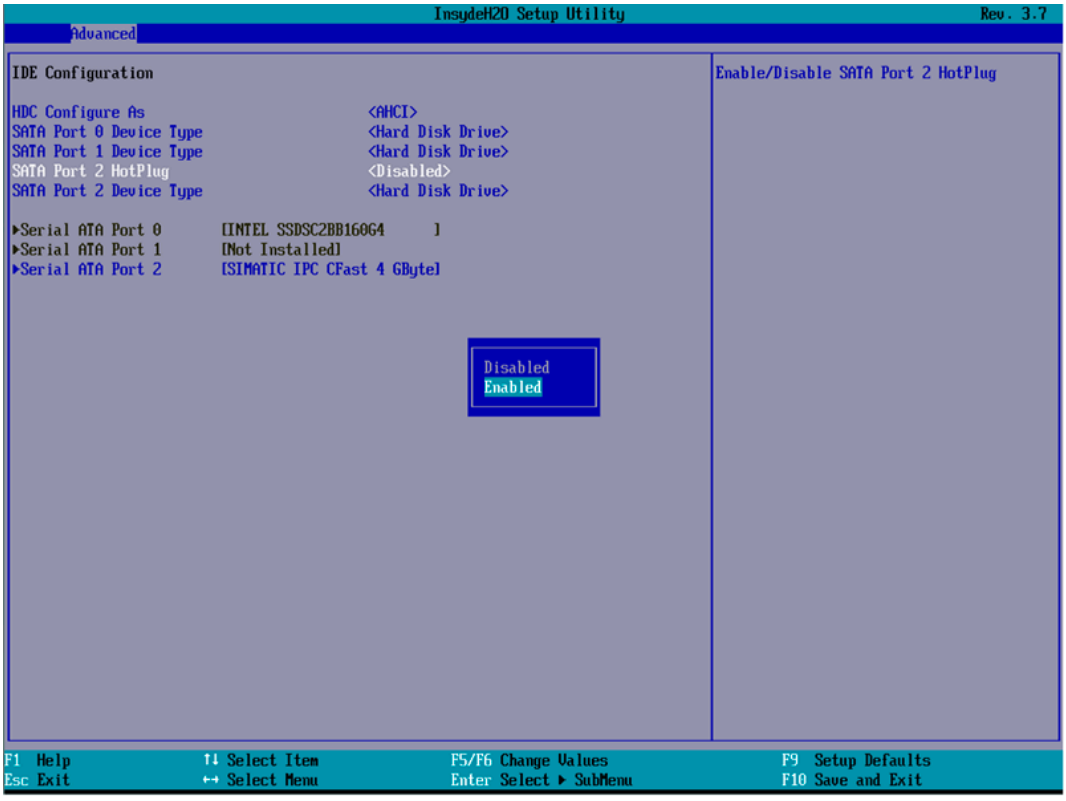


Figure 11-167 IDE Configuration > SATA Port 2 HotPlug

9. Press **F10** Save and Exit. The new entries are accepted.
10. Confirm the subsequent prompt to save the settings and exit the BIOS setup with **Yes**.

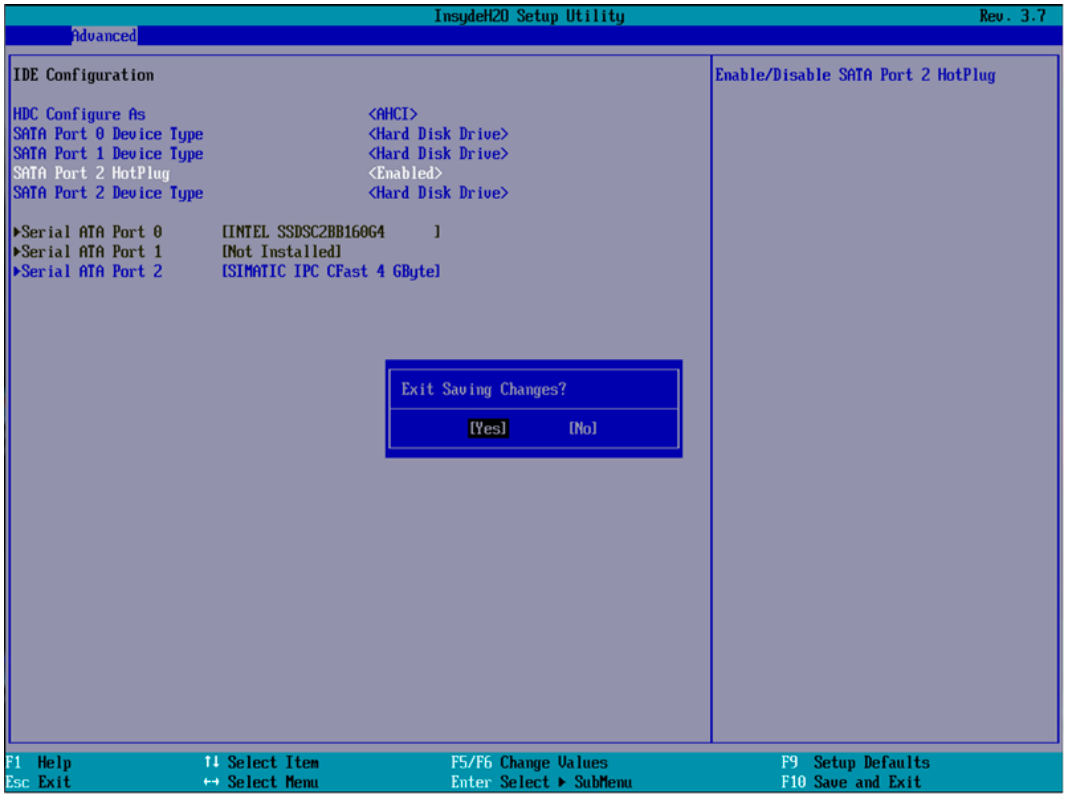


Figure 11-168 Confirming SATA Port 2 HotPlug > Enabled

11.2.1.9 Technical data

Overview of the technical specifications

Overview

The following technical specifications are available for SIMOTION P320-4 and the associated hardware components:

- General SIMOTION P320-4 technical specifications (Page 8095)
- Ambient conditions (Page 8097)
  - Climatic conditions
  - Mechanical conditions
- Technical specifications of the IsoPROFIBUS board (Page 8098)
- Power requirement of the components (Page 8099)
- Integrated DC voltage supply (Page 8100)
- Technical specifications of the SITOP smart 24 V / 10 A (Page 8100)

- Typical power consumption (Page 8102)

General technical specifications

General technical specifications

General technical specifications for SIMOTION P320-4	
Article numbers	**SIMOTION P320-4 E** • 6AU1320-4DE65-3AF0 **SIMOTION P320-4 S** • 6AU1320-4DS66-3AG0
Dimensions	262 × 133 × 50.5 (W × H × D in mm)
Weight	Approximately 2 kg
Supply voltage (DC) [1]	24 VDC [1] (-20%/+20%)
Short-term voltage interruption in accordance with Namur	Min. 15 ms (at 20.4 V) Max. 10 events/h; recovery time min. 1 s
Max. power consumption	64.8 W (at 24 V)
Degree of protection	IP 20 according to IEC 60529
Protection class	Protection class I according to IEC 61140
Safety regulations	EN 60950-1; UL 60950-1; UL 508;
Noise emission	<40dB (A) to DIN 45635-1
Quality assurance	In accordance with ISO 9001

[1] The upstream power supply must generate a safety extra-low voltage with safe electrical isolation according to IEC 60364-4-41 or an NEC Class 2 SELV and LPS according to IEC/UL/ EN/ DIN-EN 60950-1 (see the section "Connecting the power supply (24 V DC) (Page 8085)").

Electromagnetic compatibility

Electromagnetic compatibility	
Interference emission	EN 61000-6-3, EN 61000-6-4, CISPR220 class B; FCC class A
Immunity with regard to conducted interference on the supply lines	± 2 kV according to IEC 61000-4-4; burst ± 1 kV according to IEC 61000-4-5; symmetrical surge ± 2 kV according to IEC 61000-4-5; asymmetrical surge
Noise immunity on signal lines	± 1 kV according to IEC 61000-4-4; burst; length < 3 m ± 2 kV according to IEC 61000-4-4; burst; length > 3 m ± 2 kV according to IEC 61000-4-5; surge; length > 30 m
Immunity to electrostatic discharge	± 6 kV contact discharge according to IEC 61000-4-2 ± 8 kV air discharge according to IEC 61000-4-2

Electromagnetic compatibility	
Immunity to RF interference	10 V/m 80–1000 MHz and 1.4–2 GHz, 80% AM according to IEC 61000-4-3 1 V/m 2–2.7 GHz, 80% AM according to IEC 61000-4-3 10 V 10 kHz to 80 MHz, 80% AM according to IEC 61000-4-6
Immunity to magnetic fields	100 A/m, 50/60 Hz according to IEC 61000-4-8

Motherboard

Motherboard	
Processor	• Intel Core i3-3217UE 1.6 GHz, 3 MB SLC (SIMOTION P320-4 E) • Intel Core i7-3517UE 1.7 GHz, 4 MB SLC (SIMOTION P320-4 S)
Main memory	4 GB
Buffer memory	512 KB MRAM

Drive and memory media

Drive and memory media	
Solid State Disk (SSD)	≥ 80 GB Standard (SIMOTION P320-4 S)
CFast card	≥ 4 GB

Graphics

Graphics	
Graphics controller	Integrated Intel HD2000 or HD4000
Graphics memory	32 - 512 MB Shared Memory
Resolutions, frequencies, color depth	DVI-I: 640 × 480 to 1920 × 1200 / 60 Hz Display port: max. 1920 ×1200 / 60 Hz

Interfaces

Interfaces	
COM1	RS232, 115 kbps max., 9-pin SUB-D, male
DVI-I	Connection of display devices with DVI connector
Display port (DPP)	Connection of display devices with DPP connector
Keyboard	USB support
Mouse	USB support
USB	4 × USB 3.0
PROFINET	3 × RJ45 connector, onboard interface on ERTEC 400 basis, 100 Mbps electrically isolated
Ethernet	1 × Ethernet interface (RJ45) 10/100/1000 Mbps, electrically isolated, with teaming function

Ambient conditions

Climatic conditions

Temperature	Tested according to IEC 60068-2-1, IEC 60068-2-2, IEC 60068-2-14
- During operation	**Standard rail mounting (preferred mounting position)**:
	Operation with CFast card and/or SSD: • With max. 1 expansion card (max. load 10 W): 0 to +40° C • With max. 1 expansion card (max. load 10 W) in RAL [2]: 0 to +50° C [1] Operation with CFast card: • Without expansion cards in RAL [2]: 0 to +55° C [1]
	Vertical mounting:
	Operation with CFast card and/or SSD: • Without expansion cards: 0 to +40° C Operation with SSD: • With max. 1 expansion card (max. load 10 W) in RAL [2]: 0 to +45° C [1] Operation with CFast card: • Without expansion cards in RAL [2]: 0 to +50° C [1] • With max. 1 expansion card (max. load 10 W): 0 to +40° C • With max. 1 expansion card (max. load 10 W) in RAL [2]: 0 to +50° C [1]
- During storage/shipping	-40 °C to +70 °C (for devices with CFast or SSD)
- Gradient	Operating mode: Max. 10° C/h; Storage: 20° C/h; no condensation
Relative humidity	Tested according to IEC 60068-2-78, IEC 60068-2-30
- During operation	5% to 80% at 25° C (no condensation)
- During storage/shipping	5% to 95% at 25° C (no condensation)
Air pressure	
- During operation	1080 to 795 hPa (corresponds to an altitude of -1000 m to 2000 m)
- During storage/shipping	1080 to 660 hPa (corresponds to an altitude of -1000 m to 3500 m)

[1] For P320-4 S you must set "Turbo Mode Level" to "Temperature optimized" in the "Power" menu, selection "Advanced CPU Control", of the BIOS Setup; otherwise you have to reduce the maximum ambient temperature by 5 °C.

[2] RAL = Restricted Access Location
Installation of device in operating facilities with restricted access, for example, a locked control cabinet.

Mechanical conditions

Vibration	Tested according to DIN IEC 60068-2-6
- During operation	With CFast card or SSD: 5 to 9 Hz: 3.5 mm 9 to 500 Hz: 9.8 m/s2
- During storage/shipping	5 to 9 Hz: 3.5 mm 9 to 500 Hz: 9.8 m/s2
Resistance to shock	Tested according to DIN IEC 60068-2-27
- During storage/shipping	250 m/s^2, 6 ms

Technical data of the IsoPROFIBUS board

Technical specifications of the IsoPROFIBUS board (optional)

Table 11-21

IsoPROFIBUS board		
Power consumption for + 5 V PCI voltage		
Standard	2.5 W	
Maximum	3.5 W	
Permissible ambient conditions		
Heat dissipation	Open circuit ventilation	
Temperature	**Operation**	**Storage/transport**
• IsoPROFIBUS board in the SIMOTION P320-4	5° C to 45° C	-20° C to 60° C
Tested according to	EN 60068-2-1, EN 60068-2-2, EN 60068-2-14	
Relative atmospheric humidity	**Operation**	**Storage/transport**
• IsoPROFIBUS board in the SIMOTION P320-4	5 % ... 80 %	5 % ... 95 %
Tested according to	EN 60068-2-30	
Temperature change	Max. 10 K per hour in operation	
Moisture condensation and ice formation	Not permissible	
Design		
Module	Card (3.3 V/5 V, 32 bits)	
Dimensions (H x D) in mm	107 x 170	
Weight	150 g	
Space requirements	Short PCI slot	

Safety

Safety	
Degree of protection	IP20 in mounted state
Protection class	I complies with VDE 0106
Safety regulations	EN 60950
Approvals	CE, UL508, cULus

Quality assurance

Quality assurance complies with ISO 9001.

Note

The safety regulations, approvals, protection type and protection class specified are valid only if the module is inserted in a SIMOTION P320-4.

Power requirements of the components

Maximum power consumption of the auxiliary components

Auxiliary components		Maximum permitted power consumption			Max. total power
		+5 V	+3.3 V	+12 V	
USB device	High current	900 mA	-	-	10 W (for all USB devices)
Display port		-	500 mA	-	
DVI-I		500 mA	-	-	
PROFIBUS		500 mA	-	-	
PCI modules	Per slot	-	1.5 A	0.5 A	10 W (for entire device)
	Total	-	2 A	1 A	

[1] The total power of the PCI and USB cards may not exceed 15 W

Note

Device can overheat!

To avoid overheating, the power loss per PCI slot should not exceed 5 watts.

Integrated DC power supply

Technical specifications

DC power supply	
Input voltage	24 V DC (-20%/+20%)
Power consumption [1]	Max. 90 W
Power failure buffering	hold-up time = 20 ms at 20.4 V (DC_FAIL is active after > 5 ms)
Maximum continuous output power [1]	80 W
Degree of protection	IP 20
Protection class	Protection class I (a protective conductor must be connected to the device)

[1] The power specifications apply to the power supply component not to the device.

Note

Inrush current

The device requires an inrush current of at least 6.5 A for 50 ms.

The peak value of the startup current depends on the input voltage and the impedance of the 24 V power source. Peak currents greater than 6.5 A are possible. This will not have a negative impact on device functionality.

Technical data for the SITOP smart 24 V/10 A

Table 11-22 Technical specifications for the SITOP smart 24 V / 10 A

SITOP smart 24 V/10 A	
Input data	
Input voltage rated value	120/230 VAC
Input voltage range	85 to 132 VAC / 170 to 264 VAC
Power failure bridging	> 20 ms at V_{in} = 93/187 V
Mains frequency rated value	50/60 Hz
Mains frequency range	47 ... 63 Hz
Input current rated value	4.1/2.4 A
Switch-on current (at +25° C)	< 65 A
Recommended miniature circuit breaker	10 A, Characteristic C
Output data	
Output voltage rated value	24 VDC
Output voltage tolerance	± 3 %
Residual ripple/spikes	< 150/240 mV_{pp}
Output voltage adjustment range	22.8 to 28 VDC

SITOP smart 24 V/10 A	
Output current rated value	10 A (12 A up to +45° C)
Efficiency	Typ. 90%
Electronic short-circuit protection	Yes; constant current approx. 1.3 × output current rated value; for 5 seconds extra power at 1.5 × output current rated value
Ambient conditions	
Ambient temperature during storage and transport	-40° ... +85° C
Ambient temperature during operation	0° ... +60° C
Humidity class	Climate class 3K3 according to EN 60721; relative atmospheric humidity 5 to 95%; no condensation
Degree of protection (EN 60529)	IP20
Radio suppression level (EN 55022)	Class B
Line harmonic limitation according to EN 61000-3-2	No
EMC immunity	EN 61000-6-2; EN 61000-4-2/-3/-4/-5/-6/-11
EMC interference emission	EN 61000-6-4
Safety	
Safety class (IEC 536; VDE 1006 T1)	Class I
Primary/secondary galvanic isolation	SELV output voltage according to EN 60950 and EN 50178; transformer according to EN 61558-2-17; overvoltage protection in the event of an internal error $U_a < 60$ V
Dimensions and weight	
Dimensions (W x H x D) in mm	70 x 125 x 125
Weight	Approximately. 0.75 kg
Certifications and approvals	
CE	CE conformity according to 98/336 EEC and 73/23 EEC
UL/CSA	UL 508 (Listed, File E197259); CSA C22.2 No 14, No 60950-1-03
Shipbuilding	Germanischer Lloyd
Directive 94/9/EC	Declaration of conformity according to EN 60079-15: ATEX94/9/EC Cat.3;Eex, nA, II, T4 U
C-Tick	AS/NZS 2064:1997
Article numbers	
	6EP1334-2AA01, 6EP1334-2BA01

Typical power consumption

Technical specifications

Typical current and power consumption of the device at a rated voltage of 24 V

Description	Current consumption	Power consumption
Device with core i3 or core i7 processor	950 mA	23 W
Fieldbus (PROFINET)	120 mA	3 W
USB expansion [1]	Max. 500 mA	Max. 12 W
PCI expansion [1]	Max. 500 mA	Max. 12 W

[1] The total power of the PCI and USB expansion may not exceed 15 W

11.2.1.10 Dimension drawings

Overview of the dimensional drawings

This section contains the dimension drawings for the SIMOTION P320-4:

- Standard rail mounting (Page 8102)
- Vertical mounting (Page 8104)
- SIMOTION P320-4 with an expansion card (Page 8106)

Device dimension drawing for standard rail mounting

Various views of the standard rail mounting with dimensions are shown in the following:

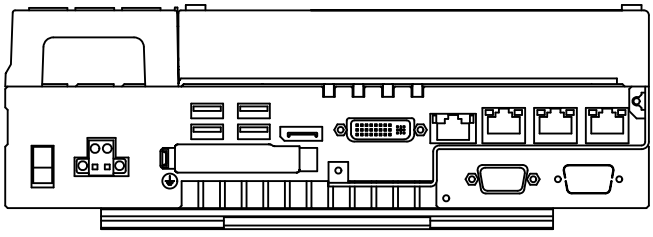


Figure 11-169 Standard rail mounting, bottom view

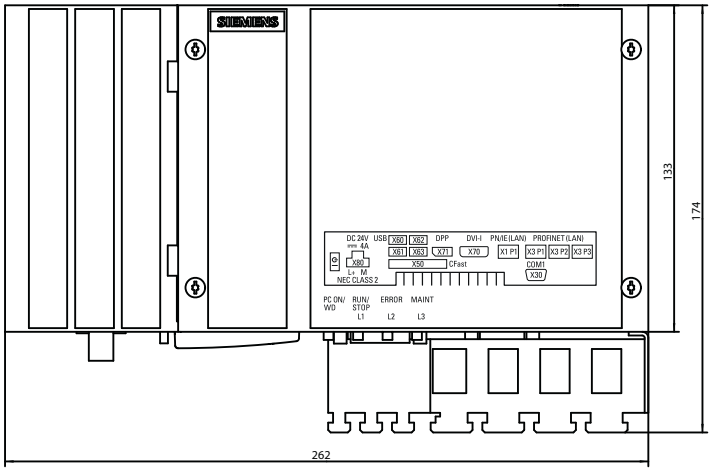


Figure 11-170 Standard rail mounting, front view

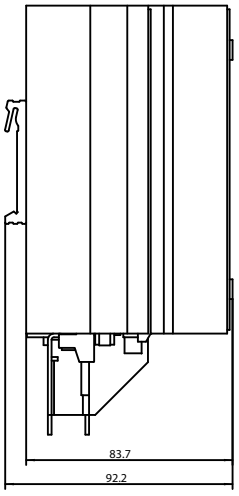


Figure 11-171 Standard rail mounting, side left view

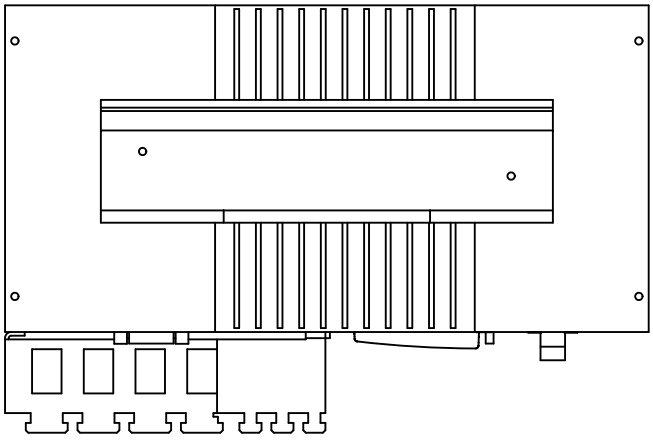


Figure 11-172 Standard rail mounting, rear view

All dimensions in mm.

Device dimension drawing for vertical mounting

Various views of the vertical mounting with dimensions are shown in the following:

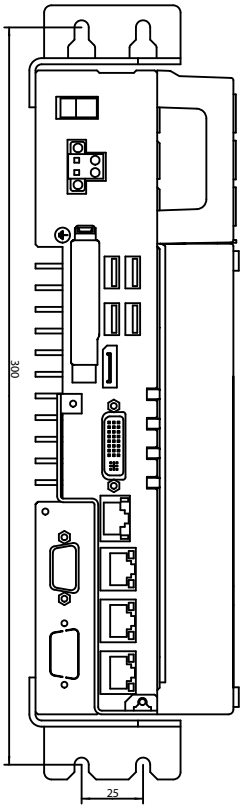


Figure 11-173 Vertical mounting, front view

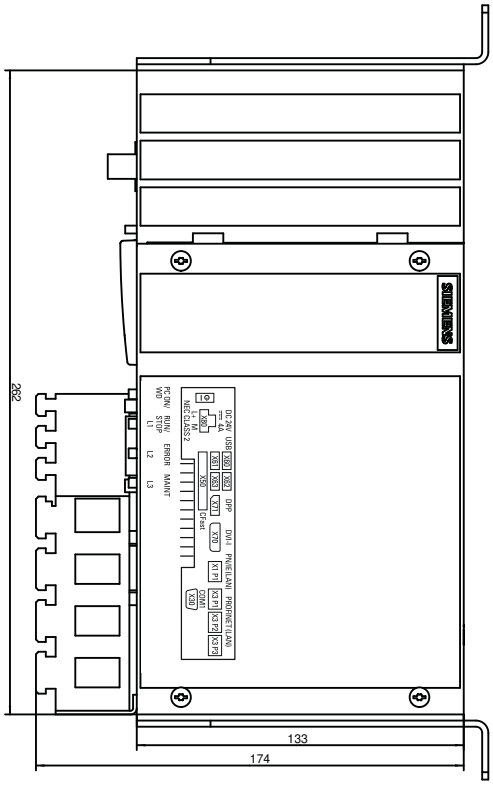


Figure 11-174 Vertical mounting, front side view

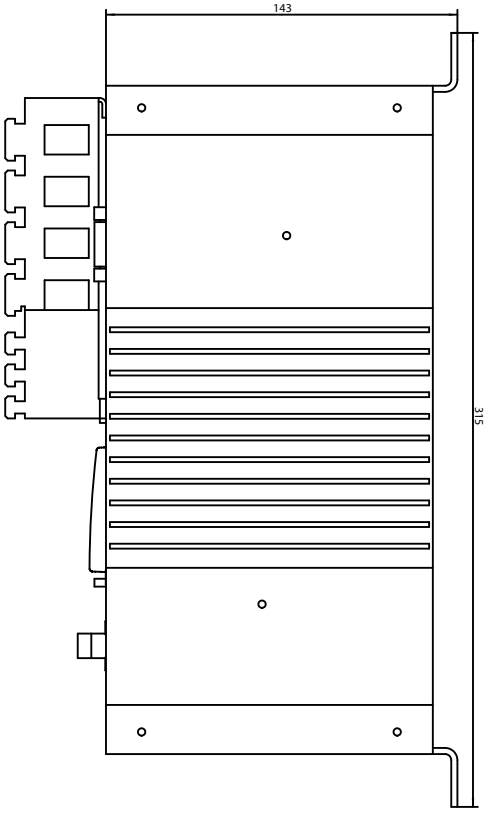


Figure 11-175 Vertical mounting, rear side view

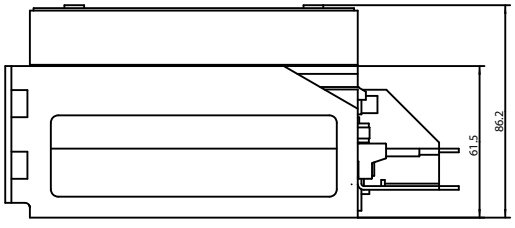


Figure 11-176 Vertical mounting, view from above

All dimensions in mm.

Dimension drawing of device with one expansion card

Device with one expansion card

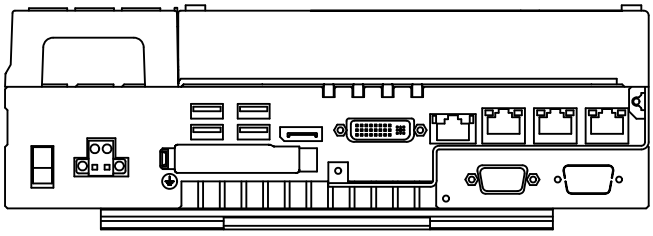
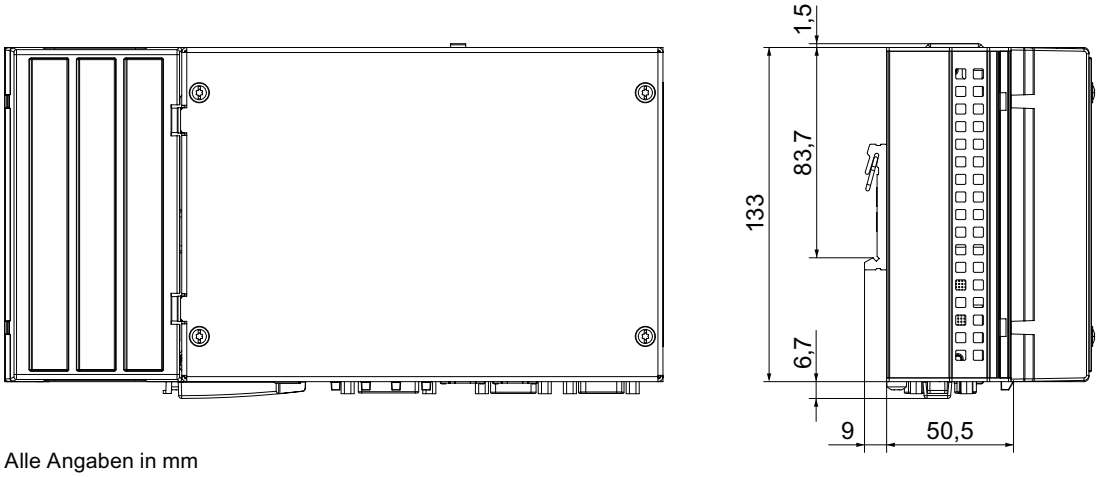


Figure 11-177 Standard rail mounting, bottom

SIMOTION P320-4 - Side views



Alle Angaben in mm

11.2.1.11 Spare parts

Available spare parts for SIMOTION P320-4

You can order the accessories via the Siemens Industry Mall (www.siemens.com/industrymall).

The following spare parts are available for SIMOTION P320-4:

Spare parts for SIMOTION P320-4			
Spare parts	**Article number**	**Scope of delivery**	**Accessories**
Lithium battery	A5E30314053	x	x
External CFast card The external CFast card contains user data. If a CFast card is used as a replacement for the external memory card, **no** Restore DVD or image is necessary.	6ES7648-2BF10-0XG0	x	x
IsoPROFIBUS board	6AU1390-0AA00-0AA1	-	x

Spares On Web

Spares On Web (http://support.automation.siemens.com/WW/view/en/16612315) is an information system that enables you to find out which spare parts are available for your device.

11.2.1.12 Standards and approvals

General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

Korea certification

KC	KC registration number: KCC-REM-S49-SIMOTION Note that this device complies with limit class A with regard to the emission of radio frequency interference. This device can be used in all areas except residential areas. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity (http://support.automation.siemens.com/WW/view/en/10805446/134200).

11.2.1.13 ESD Guideline

ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

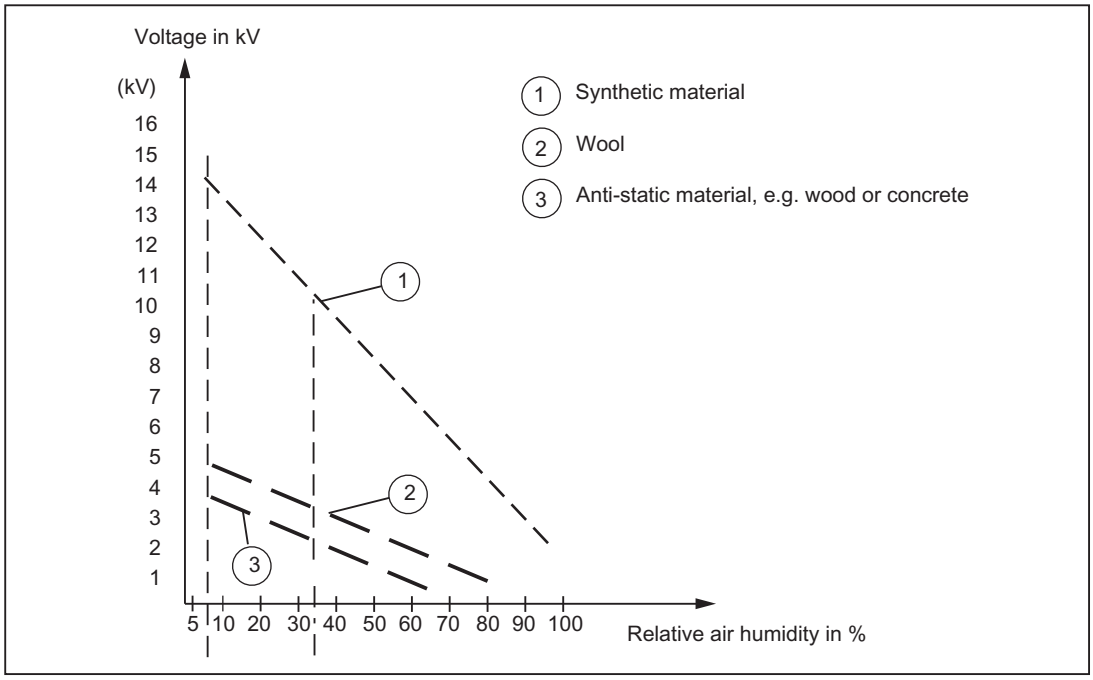


Figure 11-178 Electrostatic voltage that can accumulate on operating personnel

Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

Supplementary Documentation 12

12.1 SIMOTION documentation overview

Preface

Scope of validity

This SIMOTION documentation overview is valid for SIMOTION SCOUT product version V5.4.

SIMOTION overview

An introduction to SIMOTION and navigation to the required detailed information is available at:

www.siemens.com/simotion (www.siemens.com/simotion)

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION product version V5.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

12.1.1 SIMOTION Documentation

Overview of the SIMOTION documentation

The complete SIMOTION documentation is divided into various documentation packages and is contained on the SCOUT DVD Documentation, Utilities & Applications.

An initial overview is available as a short system overview for SIMOTION in the Basic Functions Function Manual (see the SIMOTION Description of System and Function Descriptions (Page 8114) documentation package).

Terms and abbreviations from the SIMOTION environment are described in the **SIMOTION Terms and Abbreviations, Glossary** (see SIMOTION Engineering System Handling (Page 8113) documentation package).

The search across all PDF documents of a selected language is possible using **SIMOTION_Index.pdx**. This file is included in the documentation directory of the associated language. To do this, open the file with the Adobe Acrobat Reader, and enter your search term and options in the extended search.

Extensive information especially for the programming and commissioning of a **SIMOTION D** is contained in the commissioning manuals (SIMOTION D (Page 8121) documentation package).

Example for beginners

"Getting Started" in the online help or the Engineering System documentation package is recommended as an introduction to configuring with SIMOTION.

In addition, the **Tutorial SIMOTION SCOUT TIA Getting Started** will give you detailed instructions as to how to, for example, create a project, compile and save it, insert and parameterize a technology object, and create a program. When you have worked through all these steps, you will be able to create more complex projects.

You can find the **Tutorial SIMOTION SCOUT TIA Getting Started** at https://support.industry.siemens.com/cs/ww/de/view/109474299 (https://support.industry.siemens.com/cs/ww/en/view/109474299).

A more detailed overview with appropriate instructions can be found in an example for beginners in the Utilities & Applications. .
The **SIMOTION Utilities & Applications** is contained on the **SCOUT DVD Documentation, Utilities & Applications**.

Start the navigation via "index.html" in the **Utilities_Applications**directory.

The example with a comprehensive documentation can be found at **Examples > Example for beginners**.

12.1.1.1 SIMOTION Engineering System Handling

The documentation package contains documents that describe the handling of the SIMOTION Engineering System and the SIMOTION CamTool option package.

There is also a glossary that contains the SIMOTION terms and abbreviations.

The documents of this documentation package are contained in the 1_Engineering_system_handling directory.

SIMOTION SCOUT, Configuration Manual **Edition 07/2021**
Describes SCOUT, the SIMOTION engineering system.

SIMOTION SCOUT TIA, Configuration Manual **Edition 07/2021**
Describes SCOUT TIA, the SIMOTION engineering system.

SIMOTION SCOUT TIA Device Proxy, Configuration Manual **Edition 11/2016**
Describes SIMOTION SCOUT TIA device proxy.

SIMOTION SCOUT **Edition 11/2016**
Getting Started with SIMOTION SCOUT
SIMOTION D435-2 sample project, Getting Started

SIMOTION SCOUT TIA **Edition 07/2017**
Getting Started with SIMOTION SCOUT TIA,
Getting Started

SIMOTION Terms and Abbreviations, Glossary **Edition 11/2010**
Special SIMOTION terminology, an alphabetically arranged overview.

SIMOTION CamTool, Configuration Manual **Edition 03/2018**
Describes the easy-to-use cam tool.

12.1.1.2 SIMOTION System and Function Descriptions

This documentation package contains descriptions for the basic functions of the SIMOTION system as well as the explanations for the technology objects (TO) and the communications topic.

The documents of this documentation package are contained in the 2_Description_of_system_and_functions directory.

Note

Additional information on system functions, system variables and configuration data is contained in the SIMOTION Programming - References (Page 8118) documentation package.

SIMOTION Runtime Basic Functions, Function Manual **Edition 07/2021**

This documentation provides a brief system overview for SIMOTION and describes the basic structure and the programming of the technology objects.
The runtime system and the memory concept of the SIMOTION controllers are also described.

Motion Control, TO Axis Electric/Hydraulic, External Encoder, Function Manual **Edition 07/2021**

Describes the operating principles of the speed, positioning, and hydraulic axes technology objects and the external encoder.

Motion Control, Synchronous Operation and Cam Technology Objects, Function Manual **Edition 11/2016**

Describes the operating principles of the gearing and camming technology objects.

Motion Control TO Path Object, Function Manual **Edition 04/2019**

Describes the functionality of the technology objects for path interpolation.

Motion Control Output Cam and Measuring Input, Function Manual **Edition 11/2016**

Describes the operating principles of the output cam, cam track, and measuring input technology objects.

Motion Control, Supplementary Technology Objects, Function Manual **Edition 07/2021**

Describes the operating principles of the fixed gear, addition object, formula object, sensor, controller object and temperature controller technology objects.

Motion Control, Basic Functions for Modular Machines, Function Manual **Edition 03/2018**

Describes the modular machines functionality in the SIMOTION and SINAMICS systems.

SIMOTION Communication, System Manual **Edition 07/2021**

Describes the communications capabilities for SIMOTION systems and also to devices outside the SIMOTION family, in particular, SIMATICSIMATIC.

Industrial Security, Configuration Manual **Edition 02/2020**

Describes the necessary measures and information for planning and configuring systems or plants.

12.1.1.3 SIMOTION Service and Diagnostics

This documentation package contains all information about the service and diagnostic functions of the system.

The documents of this documentation package are contained in the 3_Service_and_Diagnosis directory.

Overview of Service and Diagnostics Options, Product Information **Edition 03/2018**

Overview of the options for system diagnostics for SIMOTION devices and references to other manuals and online helps.

Technology Packages Alarms, Diagnostics Manual **Edition 07/2021**

Contains the alarms for the **Cam, Path, Cam_ext** and **TControl** technology packages.

The alarms are organized numerically below the technology packages (TP) according to technology object (TO).

Upgrading SIMOTION Devices, Operating Instructions **Edition 07/2021**

Describes a simple way of exchanging the configuration or firmware of one or more
SIMOTION devices using the Device Update Tool.

Task Trace, Function Manual **Edition 04/2014**

Describes the structure and handling of the SIMOTION Task Trace.

SIMOTION Project Comparison, Function Manual **Edition 07/2021**

Describes the SCOUT Project Comparison function with which you can compare objects within a project or with objects from other projects (offline) or objects of the project with the connected target system (online).

12.1.1.4 SIMOTION IT

This documentation package contains all information about the Web functions for SIMOTION IT with which the machine manufacturer and user can perform commissionings as well as service and diagnostic tasks without engineering tools.

The documents of this documentation package are contained in the 3_SIMOTION_IT directory.

SIMOTION IT Diagnosis and Configuration, Diagnostics Manual **Edition 07/2021**

Describes the diagnosis of the SIMOTION devices via the integrated Web server.
Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.

SIMOTION IT Programming and Web Services, Programming Manual **Edition 07/2021**

Describes the access to the diagnostic functions with Web services.
This function package comprises a Web service that permits the connection of applications to a controller via the Internet and, for example via OPC XML-DA, access to data and operating states in the SIMOTION device.
Commands are transferred via the SOAP (Simple Object Access Protocol) communication protocol.
Additional description of the Trace via SOAP (TVS) function package that permits variables from the environment of the SIMOTION variable provider to be recorded.

SIMOTION IT Virtual Machine and Servlets, Programming Manual **Edition 07/2021**

The Jamaica Virtual Machine (JamaicaVM) provides a runtime system with which Java applications can be executed on the SIMOTION device. It is an implementation of the **Java Virtual Machine Specification**.
The Servlets section of the documentation describes the use of servlets in a Web container of a SIMOTION device.

SIMOTION IT OPC UA **Edition 07/2021**

The SIMOTION IT OPC UA manual describes access to SIMOTION devices via OPC UA.

12.1.1.5 SIMOTION Programming

This documentation package contains documents with the descriptions for the various programming languages and editors.

The documents of this documentation package are contained in the 3_Programming directory.

SIMOTION ST Structured Text, Programming and Operating Manual **Edition 07/2021**

Describes the text-based Structured Text SIMOTION programming language Structured Text.

SIMOTION MCC Motion Control Chart, Programming and Operating Manual **Edition 07/2021**

Describes the graphics-based SIMOTION Motion Control Chart programming language.

SIMOTION LAD/FBD Programming and Operating Manual **Edition 07/2021**

Describes the graphics-based Ladder Diagram (LAD) and Function Block Diagram (FBD) SIMOTION programming languages.

SINAMICS/SIMOTION DCC Editor Description, Programming and Operating Manual **Edition 11/2018**

Describes the graphics-based Drive Control Chart Editor (DCC editor) based on CFC.
Graphics-based configuration of SIMOTION controllers and SINAMICS drives is possible.

12.1.1.6 SIMOTION Programming - References

SIMOTION Lists Manuals

The following documents are reference lists required for the programming of the **Cam**, **Path**, **Cam_ext** and **TControl** technology packages as well as the **SIMOTION devices**.

The documents of this documentation package are contained in the 3_Programming_reference_lists directory.

System Functions/Variables Devices, List Manual **Edition 04/2019**

Describes the system functions/variables for the **SIMOTION C**, **P** and **D** hardware platforms.

Technology Packages System Functions, List Manual **Edition 04/2019**

Describes the system functions for the **Cam_ext** and **TControl technology packages** (TP).
TP **Cam** and TP **Path** are part of TP **Cam_ext**.

The List Manual is organized based on the structure of the SIMOTION SCOUT command library.

The command library is located in the tab of the same name in the project navigator of SIMOTION SCOUT. The system functions are listed in the **PLCopen** and **Technology** folders there.

The list of reserved identifiers can be found in the SIMOTION Basic Functions manual (see SIMOTION System and Function Description (Page 8114) document package).

Technology Packages Configuration Data, List Manual **Edition 07/2021**

Describes the configuration data for the **Cam**, **Path**, **Cam_ext** and **TControl** technology packages.

The configuration data is listed as follows:
Below the technology packages (TP), in alphabetical order according to technology object (TO).

Technology Packages System Variables, List Manual **Edition 07/2020**

Describes the system variables for the **Cam**, **Path**, **Cam_ext** and **TControl** technology packages.

The system variables are listed as follows:

Below the technology packages (TP), in alphabetical order according to technology object (TO).

Other function blocks

The following documents contain descriptions of other **function blocks** from the **SIMOTION SCOUT command library**.

PLCopen Blocks, Function Manual **Edition 01/2015**

Describes the PLCopen blocks for motion control programming from a cyclic PLC viewpoint.

SINAMICS/SIMOTION Description of the standard DCC blocks, Function Manual **Edition 11/2018**

Description of the standard DCC blocks for SIMOTION and SINAMICS.

Drive connection

Standard Function for SINAMICS S120 Line Modules, Function Manual **Edition 01/2015**

Describes the function blocks for the activation and deactivation of the SINAMICS S120 Line Modules with DRIVE-CLiQ connection.

I/O

Supplement to the CP 340 and CP341 Modules, Function Manual **Edition 01/2015**

Describes the function blocks for the data exchange between a SIMOTION device and communication processors.

Supplement to the FM 350-1, FM 350-2 and FM 352 Modules, Function Manual **Edition 01/2015**

Describes the function blocks for the communication between the SIMOTION system and the FM 350-1, FM 350-2 and FM 352 modules.

Supplement to the ET 200S 1SI Serial Interface Module, Function Manual **Edition 04/2019**

Describes the function blocks for the communication between the SIMOTION system and the ET 200S 1SI serial interface.

Supplement to the ET 200S Frequency Converter, Function Manual **Edition 01/2015**

Describes the function block for controlling the ET 200S frequency converter.

Supplement to the Command Interface for AS-Interface Master Modules, Function Manual **Edition 01/2015**

Describes the function block for operation of the command interface of the AS-Interface master modules.

Standard Function for ASIsafe Safety Monitors, Function Manual **Edition 01/2015**
Describes the function blocks for reading out the diagnostic information of the ASIsafe safety monitor.

Standard Functions for RFID Systems, Function Manual **Edition 11/2016**
Describes the function blocks for the data exchange between the SIMOTION system and RFID systems according to the standard profile.

Supplement to the SIWAREX FTA Weighing Module, Function Manual **Edition 01/2015**
Describes the function block for controlling and assigning parameters for the SIWAREX FTA Weighing Module.

Controller

Basic Control, Function Manual **Edition 01/2015**
Describes the function blocks of the BasicControl software.

12.1.1.7 SIMOTION C

The documentation package contains the description for the **SIMOTION C** hardware platform.

The document of this documentation package is contained in the 5_SIMOTION_C directory.

The system functions and variables for the SIMOTION C hardware platform are described in the System Functions/Variables Devices, List Manual.
(See SIMOTION Programming - References (Page 8118) documentation package)

SIMOTION C, Operating Instructions **Edition 03/2018**
Describes the controller versions of the SIMOTION product family.

12.1.1.8 SIMOTION P

The following documents contain the descriptions for the **SIMOTION P** hardware platform.

The documents of this documentation package are contained in the 5_SIMOTION_P directory.

The system functions and variables for the SIMOTION P hardware platform are described in the System Functions/Variables Devices, List Manual.
(See SIMOTION Programming - References (Page 8118) documentation package).

SIMOTION P320-4 E / P320-4 S, Manual **Edition 03/2018**
Describes the PC-based hardware of the SIMOTION product family.

SIMOTION P320-4 E / P320-4 S, Commissioning and Hardware Installation Manual **Edition 03/2018**
Describes the PC-based hardware of the SIMOTION product family.

12.1.1.9 SIMOTION D

Documentation for the SIMOTION D hardware platform

The following documents contain the descriptions for the **SIMOTION D** hardware platform.

The documents of this documentation package are contained in the 5_SIMOTION_D directory.

The system functions and variables for the SIMOTION D hardware platform are described in the System Functions/Variables Devices, List Manual.
(See SIMOTION Programming - References (Page 8118) documentation package)

SIMOTION D4x5-2, Manual **Edition 07/2021**

Describes the drive-based hardware of the SIMOTION product family.

SIMOTION D4x5-2, Commissioning and Hardware Installation Manual **Edition 07/2021**

Describes the drive-based hardware of the SIMOTION product family.

SIMOTION D410-2, Manual **Edition 07/2021**

Describes the hardware for the SIMOTION modular drive system for single axes.

SIMOTION D410-2, Commissioning and Hardware Installation Manual **Edition 07/2021**

Describes the hardware for the SIMOTION modular drive system for single axes.

Documentation for SINAMICS Integrated

The documents for SINAMICS Integrated are contained in the 5_SIMOTION_D directory.

For **SIMOTION D**, the **SIMOTION** PLC and motion control functionality as well as the **SINAMICS S120** drive software run on a shared control hardware.

The integrated **SINAMICS Integrated** drive as well as further drive components are described in the documentation for **SINAMICS S120**.

For **SIMOTION D4xx-2**, the SINAMICS Integrated is based on **SINAMICS firmware version V5.x**

The documents for SINAMICS can also be supplied individually with the appropriate article number as hard copy.

References

Further manuals can be found at Product Support > SINAMICS S High-Performance Converter (https://support.industry.siemens.com/cs/ww/en/ps/13229/man).

12.1.1.10 SIMOTION Supplementary Documentation

This documentation package contains product information as well as the hardware descriptions for components that are operated together with SIMOTION (e.g. ADI4).

The documents of this documentation package are contained in the 4_Additional_documentation directory.

Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA, Commissioning Manual **Edition 11/2016**

This document describes the functionality and use of the Technology Modules TM Time DIDQ with SIMOTION SCOUT and SIMOTION SCOUT TIA.

TM15/TM17 High Feature SIMOTION Terminal Modules, Commissioning Manual **Edition 01/2015**

This document describes the functionality and use of the TM15 and TM17 High Feature Terminal Modules.

TM15/TM17 High Feature SIMOTION Terminal Modules, Manual **Edition 11/2016**

This document describes the functionality and use of the TM15 and TM17 High Feature Terminal Modules.

ADI4 - Analog Drive Interface for Four Axes, Manual **Edition 04/2014**

This document describes the functionality and use of the ADI4 - Analog Drive Interface with which as many as 4 drives with analog setpoint interface can be operated on the isochronous PROFIBUS DP.

SIMATIC Distributed I/Os IM 174 PROFIBUS Module, Manual **Edition 09/2011**

This document describes the standard functionality of the IM 174 module, an interface module with which as many as four drives can be operated with the analog setpoint interface with one TTL or SSI encoder per axis on the isochronous PROFIBUS DP.

SIMATIC NET (Win7/Win10) for SIMOTION, Product Information **Edition 08/2018**

This document describes the open OPC interface for access to various communication peers via SIMATIC NET.

12.1.2 Standard SIMOTION applications

Numerous standard applications are available for SIMOTION that already provide a solid basic framework or predefined sector-specific configurations. With the help of the provided documentation, the applications can be easily used, adapted and extended for the associated application.

The **SIMOTION Utilities & Applications** are supplied with SIMOTION SCOUT .

They contain standard applications such as, for example:

SIMOTION Flying Saw
Flying saw

SIMOTION Rotary Knife
Cross cutter

SIMOTION Winder
Winder

SIMOTION Traverser
Traverser

SIMOTION Line Tension Control
Tension control

SIMOTION Top Loading
Solution for flexible handling applications

Project generator SIMOTION easyProject
The **SIMOTION Utilities & Applications** also contain the SIMOTION easy-Project project generator.
Basic functions required in practically every application can be integrated quickly and easily in a new or existing project with the aid of **SIMOTION easyProject** .

Other standard applications

All of the complete SIMOTION standard applications can be found on the Internet at: SIMOTION industry-sector solutions (http://www.Siemens.com/simotion/solutions) and as a download in the Industry Online Support (https://support.industry.siemens.com/cs/ww/en/ps/14505/ae).

The documents are supplied with the associated application or on request.
Please contact your Siemens contact regarding this.

A selection of standard applications is listed below:

SIMOTION Easy Basics
Collection of standardized SIMOTION basic functionality
(also see Product Support > Motion Control System SIMOTION > SIMOTION Easy Basics (https://support.industry.siemens.com/cs/ww/en/view/43192803))

SIMOTION Modular Machine

Permits topology changes in a SINAMICS drive system during runtime.

SIMOTION Message Handling

The application for quick integration of the message handling in an existing SIMOTION project

SIMOTION Axis Function Block

Solution for controlling motion control basic function.

SIMOTION Startup Check

The application for the startup check of devices and I/O modules in the SIMOTION system

SIMOTION Cartoner

Solution for packaging machines

SIMOTION Intelligent Belt

Solution for the automatic operation of an intelligent belt (dual tension) as well as the provision of functions such as homing, positioning and jogging

SIMOTION/SIMATIC Ethernet Communication TCP/IP LCom

TCP/IP communication for SIMOTION and SIMATIC for data blocks up to 64 KB time synchronization

SIMOTION/SIMATIC OMAC V3

This software library provides a user-friendly basis for the configuration of an OMAC-compliant mode manager and a data interface for SIMOTION or SIMATIC

SIMOTION Hydraulik/Servo/Mechanical Press

The application for automating mechanical universal presses with SIMOTION

SIMOTION Electronic Tranfer

Solution for electronic transfer systems in the metal forming technology

SIMOTION Roll Feed

Solution for electronic roll feed in the metal forming technology

SIMOTION Feeder

Solution for press linking with feeder in the metal forming technology

SIMOTION Print Standard
Application example for various printing machine types

SIMOTION Application Traverser
Traverser

SIMOTION Application Weaving
Solution for weaving machines

SIMOTION Application Ring Spinning
Solution for ring spinning machines

SIMOTION Application Rowing Frame
Solution for flyer control in the spinning process

12.1.3 SIMOTION Ordering Information

Catalogs for SIMOTION and other components

SIMOTION, Catalog PM 21 **Edition 2017**
Equipment for production machines
Ordering information
Article number: E86060-K4921-A101-A4

SINUMERIK 840, Catalog NC 62, **Edition 2018**
Equipment for machine tools
Ordering information
Article number: E86060-K4462-A101-A3

SIMATIC Products for Totally Integrated Automation, Catalog ST 70, **Edition 2017**
Ordering information
Article number: E86060-K4670-A101-B6

SIMATIC Products for Totally Integrated Automation **Edition 2018**
Catalog News ST 70 N
Article number: E86060-K4670-A151-A9

SIMATIC NET, Industrial Communication, Catalog IK PI, Complete Catalog **Edition 2017**
Ordering information

Article number: E86060-K6710-A101-B8

SIMATIC NET, Industrial Communication, Short Catalog **Edition 11/2013**
Article number: E86060-K6710-B111-B3

Interactive catalogs

Products for Automation and Drives **Edition 10/2018**
Annual update in October, can be ordered at:
Product catalog CA 01 (http://w3.siemens.com/mcms/topics/en/ik/Pages/Default.aspx)
Article number: E86060-D4001-A500-D8

Industry Mall, Catalog and Online Ordering System for Automation and Drives
Industry Mall (http://www.siemens.com/industrymall)

Technical online documentation for SINUMERIK, SINAMICS, SIMOTION and SIMOTICS

Information and documentation for SINUMERIK, SINAMICS, SIMOTION and SIMOTICS are available on the Internet under:

Online documentation for SINUMERIK, SINAMICS, SIMOTION and SIMOTICS (https://support.industry.siemens.com/cs/ww/en/view/109476679)

In addition to many other useful documents, you will also find in the Information and Download Center catalogs relating to:

- SINUMERIK: NC 62, NC 81.1, NC 82
- SINAMICS: D 11, D 12, D 21.3, D 21.4, D 23.1, D 23.2, D 31, D 35
- SIMOTION: PM 21
- SIMOTICS: D 41, D 81.1, D 81.8, D 83.1

Information and Download Center (http://www.siemens.com/industry/infocenter)

12.1.4 Additional information for SIMOTION

The following documents contain advanced information about SIMOTION.

Michael Braun / Wolfgang Horn: Object-Oriented Programming with SIMOTION. **Edition 2016**
Basic Principles, Example Programs and Concepts according to IEC 61131-3.
1st Edition October 2016. Publicis Publishing, Erlangen
ISBN 978-3-89578-455-2

SIMATIC Manual Collection on DVD,

In 5 languages, all manuals for S7-1200/1500/200/300/400,LOGO!, SIMATIC DP, PC, PG, STEP 7, Engineering SW, Runtime SW, PCS7, SIMATIC HMI, SIMATIC NET, SIMATIC IDENT

Siemens Support Entry ID 4073541 (https://support.industry.siemens.com/cs/ww/en/ps/6ES7998-8XC01-8YE0)

Article number: **6ES7 998-8XC01-8YE0**

SIMATIC HMI WinCC flexible 2008 Runtime, **System Manual** **Edition 07/2008**

Engineering software for configuring SIMATIC Panels,

WinCC flex 2008 Runtime

Siemens Support Entry ID 18795593 (https://support.industry.siemens.com/cs/ww/en/view/18795593)

Article number: 6AV6691-1BA01-3AA0

SIMATIC WINCC ADVANCED V13 SP1, **System Manual** **Edition 12/2014**

Engineering software in the TIA Portal for configuring SIMATIC Panels, WinCC Runtime Advanced

Siemens Support Entry ID: 109091876 (https://support.industry.siemens.com/cs/ww/en/view/109091876)

Article number: 6AV2102-0AA03-0AA5

SIMATIC NET CP 343-2 / CP 343-2 P AS-Interface Master, Manual **Edition 08/2008**

Siemens support: Entry ID 5581657 (https://support.industry.siemens.com/cs/ww/en/view/5581657)

Document identification number: C79000-G8900-C149-04

SIMATIC NET DP/AS-Interface Link 20E, Manual **Edition 11/2002**

Siemens support: Entry ID 5281638 (https://support.industry.siemens.com/cs/ww/en/view/5281638)

Document identification number: C79000-G8900-C138-04

SIMATIC NET DP/AS-Interface Link Advanced, Manual **Edition 03/2008**

Siemens support: Entry ID 22710305 (https://support.industry.siemens.com/cs/ww/en/view/22710305)

Document identification number: C79000-G8900-C209-03

SIMATIC NET IE/AS-Interface Link PN IO, Manual **Edition 03/2008**

Siemens support: Entry ID 22712154 (https://support.industry.siemens.com/cs/ww/en/view/22712154)

Document identification number: C79000-G8900-C216-03

SIMATIC S7-300 CP 340 Point-to-Point Connection, Installation and Parameter Assignment, Manual **Edition 04/2011**
Siemens support: Entry ID 1137332 (hthttps://support.industry.siemens.com/cs/ww/en/view/1137332)
Document identification number: A5E00369891-03

SIMATIC S7-300 CP 341 Point-to-Point Connection, Installation and Parameter Assignment, Manual **Edition 04/2011**
Siemens support: Entry ID 1117397 (https://support.industry.siemens.com/cs/ww/en/view/1117397)
Document identification number: A5E02191070-03

SIMATIC S7-300 FM 350-1 Counter Module, Manual **Edition 05/2011**
Siemens support: Entry ID 1086726 (https://support.industry.siemens.com/cs/ww/en/view/1086726)
Document identification number: A5E03539812-01

SIMATIC S7-300 FM 350-2 Counter Module, Manual **Edition 05/2011**
Siemens support: Entry ID 1105178 (https://support.industry.siemens.com/cs/ww/en/view/1105178)
Document identification number: A5E00271803-03

SIMATIC S7-300 FM 352 Electronic Cam Controller **Edition 05/2011**
Installation and Parameter Assignment, Operating Instructions
Siemens support: Entry ID 2103044 (https://support.industry.siemens.com/cs/ww/en/view/2103044)
Document identification number: A5E01071719-03

SIMATIC ET 200S Serial Interface Modules, Operating Instructions **Edition 03/2009**
Siemens support: Entry ID 9260793 (https://support.industry.siemens.com/cs/ww/en/view/9260793)
Document identification number: A5E00124880-05

SIMATIC Logon, Configuration Manual **Edition 08/2008**
Siemens support: Entry ID 34519648 (https://support.industry.siemens.com/cs/ww/en/view/34519648)
Document identification number: A5E00496671-05

SIRIUS AS-Interface Safety Monitor, Operating Instructions **Edition 12/2013**
Edition RS-AB/004
Siemens support: Entry ID 12265037 (https://support.industry.siemens.com/cs/ww/en/view/12265037)
Article number: 3RK1701-2MB21-0AA0

AS-Interface Safety Monitor, Operating Instructions **Edition 09/2008**
Siemens support: Entry ID 24432172 (https://support.industry.siemens.com/cs/ww/en/view/24432172)
Document identification number: GWA 4NEB 333 1557 01 DS02
Article number: 3RK1701-2MB21-0AA0

ASIMON V3 **Edition 04/2016**
Configuration Software for AS-Interface Safety Monitor,
Programming and Operating Manual
Configuration Software for Microsoft® Windows®
Siemens support: Entry ID 24434774 (https://support.industry.siemens.com/cs/ww/en/view/24434774)
Document identification number: NEB333155801000/RS-AA/003

SIMATIC Ident RFID Systems ASM 456 Interface Module, Operating Instructions **Edition 07/2015**
Siemens support: Entry ID 32629442 (https://support.industry.siemens.com/cs/ww/en/view/32629442)
Document Identification Number: J31069-D0162-U001-A6-0018

SIWAREX FTA Weighing Electronics for Independent Weighing, Manual **Edition 11/2014**
Siemens support: Entry ID 17970155 (https://support.industry.siemens.com/cs/ww/en/view/17970155)
Article number: A5E00452858A

12.2 Technology Modules TM Timer DIDQ for SIMOTION SCOUT and SIMOTION SCOUT TIA

Preface

This manual describes TM Timer DIDQ 10x24V and 16x24V technology modules used with SIMOTION controllers with PROFINET connection.

This manual is aimed at machine manufacturers, plant engineers, commissioning personnel, and service personnel who use technology modules with SIMOTION SCOUT or with SIMOTION SCOUT TIA.

The following information blocks describe the purpose and use of the Commissioning Manual:

- Description
  This section describes the general use of the TM Timer DIDQ technology modules 10x24V and 16x24V.
- Configuring
  Provides information about configuring, parameterizing and commissioning the TM Timer DIDQ technology modules with the SIMOTION SCOUT and SIMOTION SCOUT TIA engineering systems.
- Functions
  This chapter provides an overview of the functions of the TM Timer DIDQ technology modules.
- Service and maintenance
  Provides information on maintenance work that must be implemented on the TM Timer DIDQ technology modules.
- Diagnostics
  Provides information about the available diagnostic information, how to interpret it, and its meaning.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

SIMOTION at a glance

We have compiled an overview page from our range of information about SIMOTION with the most important information on frequently asked topics - which can be opened with only one click.

Whether beginner or experienced SIMOTION user – the most important downloads, manuals, tutorials, FAQs, application examples, etc. can be found at

https://support.industry.siemens.com/cs/ww/en/view/109480700

Additional information

Click the following link to find information on the following topics:

- Documentation overview
- Additional links to download documents
- Using documentation online (find and search manuals/information)

https://support.industry.siemens.com/cs/ww/en/view/109479653

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

https://support.industry.siemens.com/cs/de/en/ps/14505/faq

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

https://support.industry.siemens.com/cs/ww/en/sc/2090

12.2.1 Safety notes

12.2.1.1 Fundamental safety instructions

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

Danger to life due to software manipulation when using removable storage media

⚠ WARNING

Danger to life due to software manipulation when using removable storage media

The storage of files on removable storage media involves a high risk of infection, e.g. via viruses or malware. Incorrect parameter assignment can cause machines to malfunction, which can lead to injuries or death.

- Protect the files on removable storage media against harmful software through appropriate protective measures, e.g. virus scanners.

Residual risks of power drive systems

When performing the risk assessment for a machine or plant in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer or plant constructor must take into account the following residual risks associated with the control and drive components of a drive system:

1. Unintentional movements of driven machine or system components during commissioning, operation, maintenance and repairs caused by, for example:
   - Hardware and/or software errors in the sensors, control system, actuators, and cables and connections
   - Response times of the control system and of the drive
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of wireless devices / mobile phones in the immediate vicinity of electronic components
   - External influences/damage
   - X-rays, ionizing radiation and cosmic radiation
2. Unusually high temperatures, including open flames, as well as emissions of light, noise, particles, gases, etc., can occur inside and outside the components under fault conditions caused by, for example:
   - Component failure
   - Software errors
   - Operation and/or environmental conditions outside the specification
   - External influences/damage
3. Hazardous shock voltages caused by, for example:
   - Component failure
   - Influence during electrostatic charging
   - Induction of voltages in moving motors
   - Operation and/or environmental conditions outside the specification
   - Condensation/conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc., if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

For more information about the residual risks of the drive system components, see the relevant sections in the technical user documentation.

12.2.2 Description

12.2.2.1 Introduction

The following provides an overview of the technical characteristics of the TM Timer DIDQ 10x24V and 16x24V technology modules. It also provides information on systems integration with SIMOTION and an overview of typical machinery applications.

Detailed information on the technology modules, including installation, can be found in the SIMATIC documentation. An overview of the available manuals can be found in Chapter SIMATIC documentation (Page 8135).

12.2.2.2 SIMATIC documentation

Overview of SIMATIC documentation for Timer DIDQ technology modules

The following table lists the SIMATIC documentation that you require when using the Timer DIDQ technology modules.

Note

This manual is intended solely as a supplement for the configuration of TM Timer DIDQ technology modules under SIMOTION SCOUT or SIMOTION SCOUT TIA. The SIMATIC documentation listed in the following table is therefore required when using the technology modules.

Subject	Documentation	Most important content
Description of the system	• ET 200SP Distributed I/O System System Manual • ET 200MP Distributed I/O System System Manual	• Use planning • Mounting • Connecting • Commissioning
	S7-1500 Automation System System Manual	
	Interface Module Manual	• Connecting • Alarms, diagnostic, error and system alarms • Technical specifications • Dimension drawing
	• Manual ET 200SP Technology Module TM Timer DIDQ 10x24V • Manual ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V	• Properties • Connecting • Configuring • Technical specifications
Configuring control systems so that they are interference-proof	Designing interference-free controllers Function Manual	• Fundamentals • Electromagnetic compatibility • Lightning protection

Subject	Documentation	Most important content
Time-based IO	Highly accurate input/output with time-based I/O Function Manual	• Fundamentals • Configuring • Programming • Diagnostics
Isochronous mode	PROFINET with STEP 7 Function Manual	• Benefit • Application • Parameter settings

SIMATIC Manuals

On the Internet (http://www.siemens.com/automation/service&support (http://www.siemens.com/automation/service&support)), all current manuals for SIMATIC products are available as a free download.

12.2.2.3 Properties

Properties of the TM Timer DIDQ 10x24V and 16x24V technology modules

The TM Timer DIDQ 10x24V and 16x24V technology modules have the following properties:

TM Timer DIDQ 10x24V technology module	TM Timer DIDQ 16x24V technology module
Technical properties	
4 digital inputs 6 digital outputs	16 digital inputs and outputs, electrically isolated in groups of 8 Various combinations of the digital inputs and outputs can be parameterized: • 0 digital inputs and 16 digital outputs (for cam applications with many outputs) • 3 digital inputs and 13 digital outputs (for mixed applications, involving cams, measuring inputs and incremental encoders) • 4 digital inputs and 12 digital outputs (for flexible mixed operation) • 8 digital inputs and 8 digital outputs (for measuring inputs and incremental encoders) **Note**: As of TIA Portal V14 and SIMOTION SCOUT TIA V4.5, depending on the channel configuration (see above), as many as eight additional digital inputs without technology are available. For further information, see sections Setting options for TM Timer DIDQ 16x24V technology module (Page 8166) and Linking symbolic I/O variables with TM Timer DIDQ (Page 8172).
24 VDC rated output voltage	
Rated output current 0.5 A or 0.1 A (high-speed operation) per digital output	
24 V encoder supply output, short-circuit proof	

TM Timer DIDQ 10x24V technology module	TM Timer DIDQ 16x24V technology module
Parameterizable substitute values (per digital output)	
Parameterizable diagnostics	
Two supply voltages L+ (only TM Timer DIDQ 16x24V technology module)	
Supported encoder/signal types for digital inputs	
• 24 V incremental encoder with signal A and B • 24 V pulse encoder with a signal	
Supported functions	
• Time stamp function for inputs and outputs (resolution 1 µs) • Counting (counting range 32 bits) • Oversampling for inputs and outputs • Pulse width modulation	
Supported system functions	
• Isochronous mode • Firmware update • I&M identification data	

12.2.2.4 System integration

System integration

To integrate and operate the TM Timer DIDQ technology modules, an interface module is always required; for interface module ET 200SP, the IM 155-6 PN HF and for interface module ET 200MP, the standard or HF version.

The following options exist to integrate the TM Timer DIDQ technology modules in a SIMOTION automation solution via PROFINET:

- SIMOTION SCOUT
  When configured using SIMOTION SCOUT, only TM Timer DIDQ 10x24V technology modules with interface module ET 200SP IM 155-6 PN HF can be used.

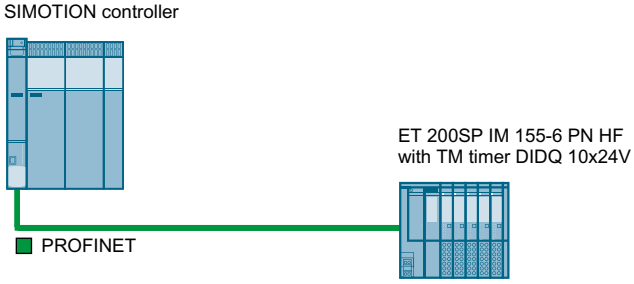


Figure 12-1 Integration in SIMOTION SCOUT

- SIMOTION SCOUT TIA
  During configuration with SIMOTION SCOUT TIA, the technology modules can be integrated as follows:
  - Integration of the TM Timer DIDQ 10x24V technology modules via ET 200SP IM 155-6 PN HF interface module.
  - Integration of the TM Timer DIDQ 16x24V technology modules via ET 200MP standard or HF version.

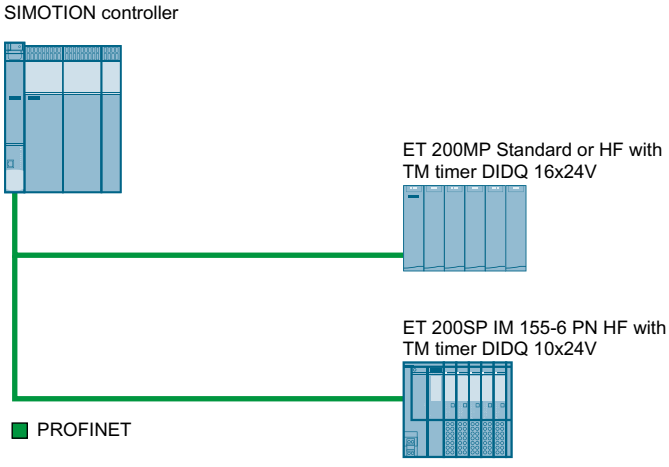


Figure 12-2 Integration SIMOTION SCOUT TIA

Setup

In the *SIMATIC ET 200SP TM Timer DIDQ 10x24V technology module* and *SIMATIC ET 200MP/ S7-1500 TM Timer DIDQ 16x24V technology modules* manuals you can find information about the design of the technology modules and a description of the interfaces.

12.2.2.5 Machine applications

Many production machines require fast, precise detection of signals or precise switching of digital outputs. The TM Timer DIDQ technology modules therefore represent an optimum solution for the requirements of a wide range of industrial applications.

Applications

Many production machines require fast, precise detection of signals or precise switching of digital outputs. Applications include the following:

- Edge detection
- Quality monitoring (e.g. product is good/bad)
- Product tracking (e.g. product is available / not available)
- Print-mark detection
- Tool monitoring (e.g. presses)
- Machine condition monitoring (e.g. plastic injection molding machines)
- Weft break monitoring (e.g. textile machines)

Applications requiring fast, high-precision output of signals include:

- Position-dependent switching of actuators
  - Camera trigger signal (quality assurance)
  - Control of an air nozzle for blowing away cut-offs
  - Control of nozzles for applying glue tracks
- Product extraction from production line
- Implementation of line motion control systems
- Output of pulse patterns

Application example: Application of glue tracks

In the following example, glue tracks are applied to a workpiece. The glue tracks are applied via glue guns. The control of the glue guns is via digital outputs with cam functionality on the TM Timer DIDQ technology module.

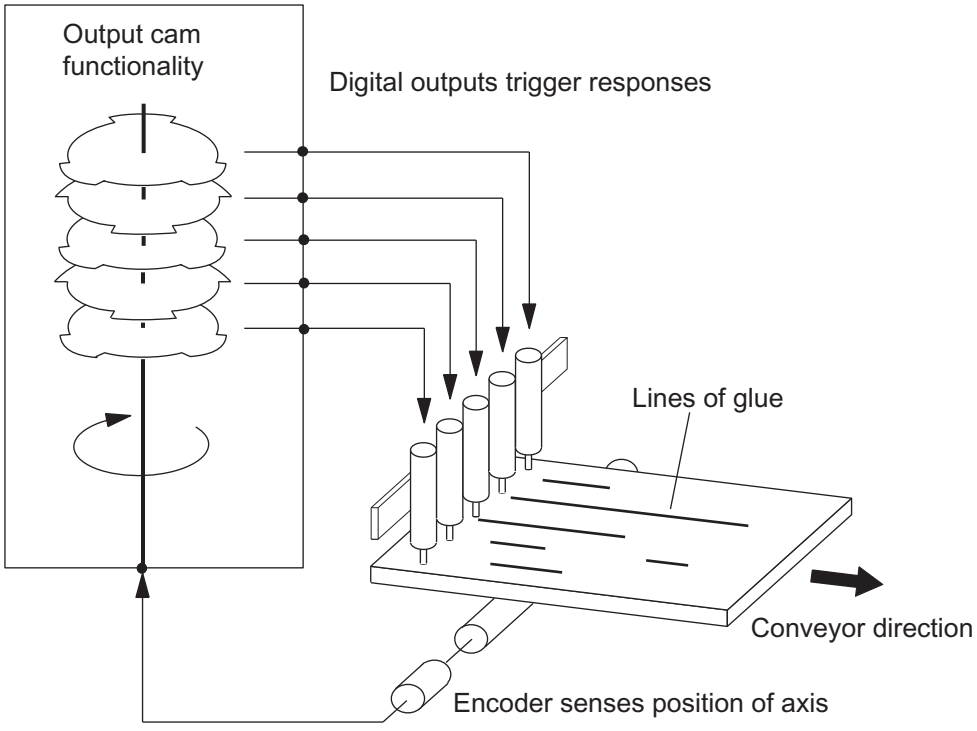


Figure 12-3 Example of an electronic cam control

Depending on the axis position, the SIMOTION "TO outputCam" or "TO camTrack" controls the digital outputs of the technology module.

The output delay times (output delay times of DO / actuators, etc.) are compensated in the technology object. This compensation ensures that the glue placement is always accurate regardless of machine speed.

12.2.3 Configuring

12.2.3.1 Introduction

The chapter describes the parameterization of the TM Timer DIDQ technology modules and the configuration of the communication under the engineering systems SIMOTION SCOUT and SIMOTION SCOUT TIA. The use of the I/O channels is shown using an interconnection with TO outputCam or TO measuring input and addressing using symbolic variables in the user program.

12.2.3.2 SIMOTION SCOUT

Hardware and software requirements

The following requirements must be satisfied in order to be able to work with the TM Timer DIDQ technology modules.

Hardware requirements

- SIMATIC ET 200SP interface module, HF or HS version
- TM Timer DIDQ 10x24V technology module
- SIMOTION controller with PROFINET interface (see section System integration (Page 8137))
- PROFINET cables
- External 24 VDC power supply (note the requirements in the *SIMATIC ET 200SP TM Timer DIDQ 10x24V technology module*)

Software requirements

Please observe the following requirements:

- SIMOTION software as of V4.4 HF6
- SIMATIC ET 200SP interface module in the HF version
- SIMATIC Step7 as of V5.5 SP4 and installation of the HSP file, minimum HSP240.V2.2

Note

Download source for the HSP:

http://support.automation.siemens.com/WW/view/de/23183356 (http://support.automation.siemens.com/WW/view/en/23183356)

Instructions to install the HSP for STEP7 V5.x are provided under the following FAQ:

http://support.automation.siemens.com/WW/view/de/22374877 (http://support.automation.siemens.com/WW/view/en/22374877)

The technology modules are integrated into the hardware configuration (HW Config) of the SIMATIC STEP 7 engineering system and are assigned parameters there.

HW Config is used by SIMOTION SCOUT and SIMATIC STEP 7 to configure the hardware of an automation project.

Irrespective of whether you work with SIMATIC STEP 7 or SIMOTION SCOUT, the parameterization is always performed in HW Config.

Requirements for operation

Preconditions

The following preconditions must be met before operating TM Timer DIDQ technology module.

1. Your system hardware including the technology module is installed and wired.
2. You have physically connected the technology module via an ET 200SP interface module, version HF or HS, with a SIMOTION controller via the PROFINET IO interface (see Chapter System integration (Page 8137)).
3. You have created a project with a SIMOTION controller with PROFINET interface in SIMATIC STEP 7 or in SIMOTION SCOUT and configured PROFINET IRT. For isochronous operation, the following settings must be made at the SIMOTION controller.

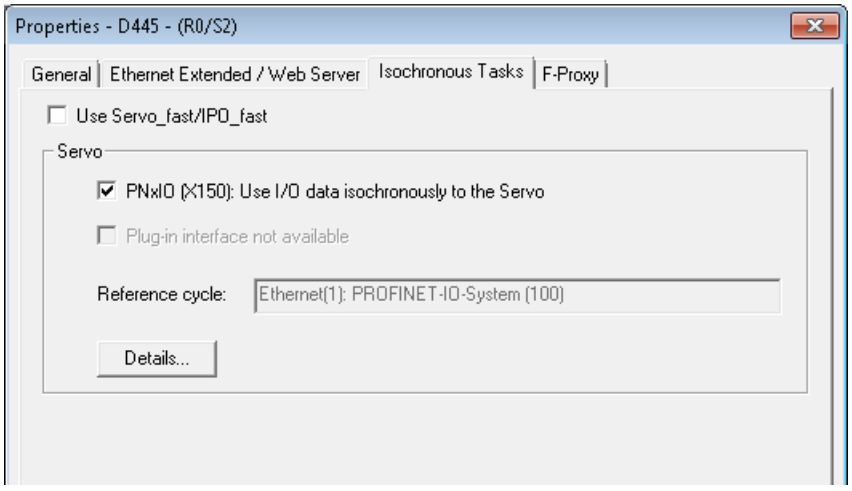


Figure 12-4 Isochronous setting at the SIMOTION controller

Note

For further information on configuration of the PROFINET communication please refer to the *SIMOTION SCOUT Communication* Manual.

Note

The procedures contained in this section assume that the user has a general understanding of SIMATIC STEP 7 and SIMOTION SCOUT.

Note

Please note that interface module ET 200SP – in conjunction with technology module TM Timer DIDQ – is operated with a minimum PROFINET clock cycle of 500 µs depending on the quantity structure.

Creating a technology module and configuring communication

Procedure

1. Open the HW configuration in SIMOTION SCOUT or SIMATIC STEP 7.
2. First, integrate an ET 200 SP interface module, version HF into the project. In the hardware catalog, navigate via PROFINET IO -> I/O -> ET 200SP to the corresponding module.
3. Select the module in the "Hardware Catalog" window.

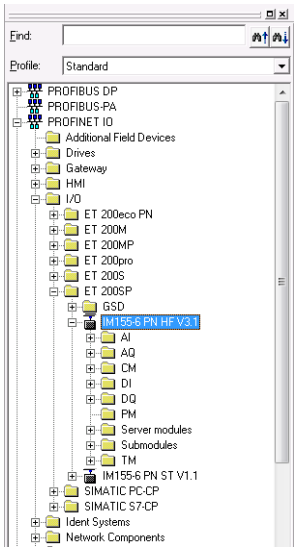


Figure 12-5 Hardware catalog - ET 200SP interface module, version HF

4. Drag and drop the module into the station window on the PROFINET line.

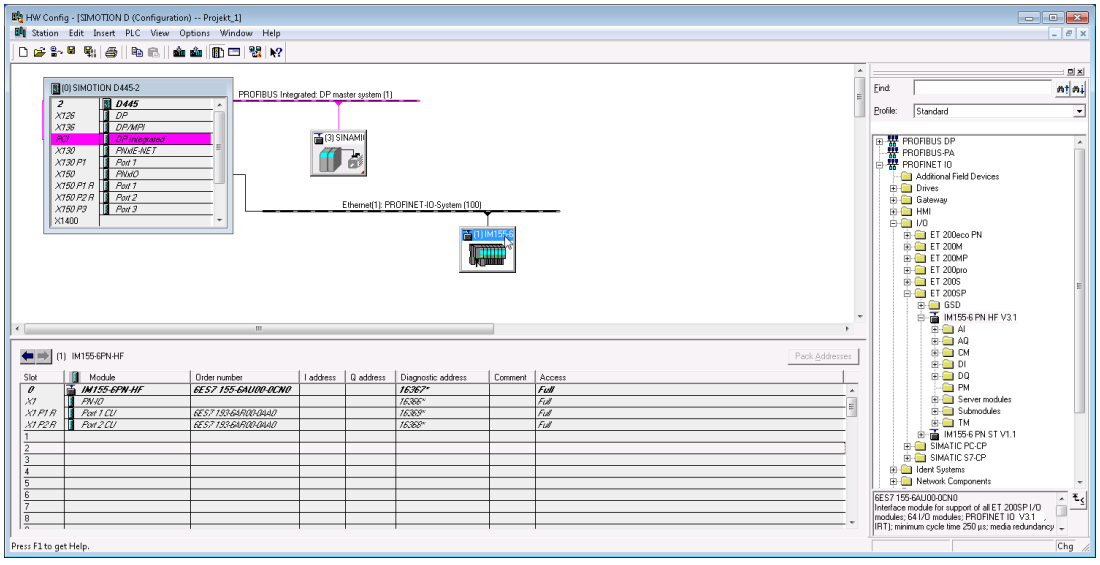


Figure 12-6 ET 200SP interface module inserted in the station window on PROFINET line

5. Now insert the TM Timer DIDQ 10x24V technology module into the project. In the hardware catalog, navigate via PROFINET IO -> I/O -> ET 200SP -> TM -> Time based IO to the module.
6. Select the module in the "Hardware Catalog" window.

7. Drag the module to the detail view under the ET 200SP.

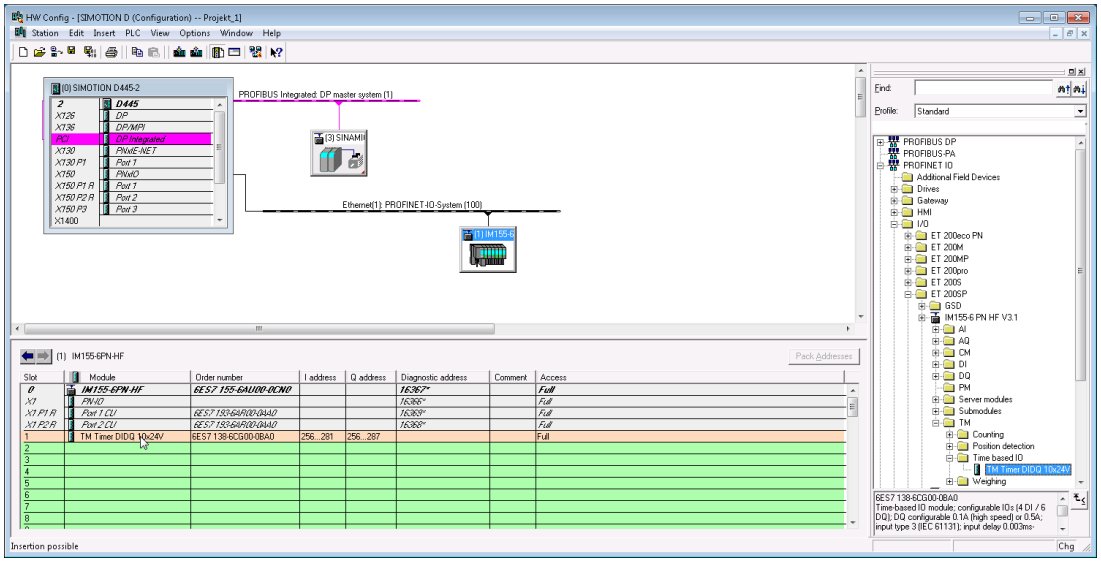


Figure 12-7 TM Timer DIDQ 10x24V technology module in the detail view

8. Then, add another server module to the detail view. With the server module, the ET 200SP distributed I/O system is completed.

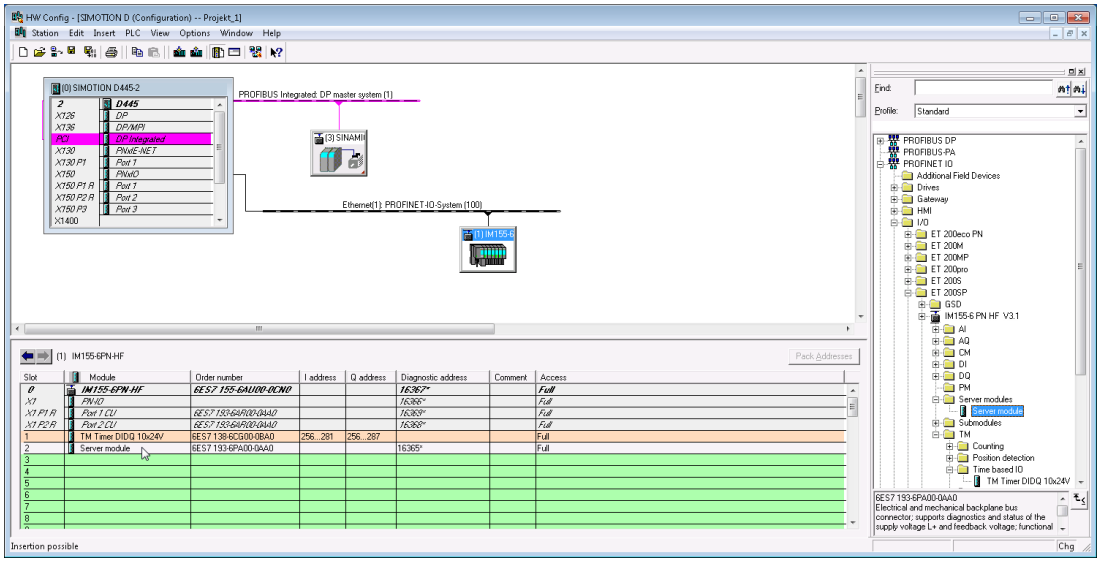


Figure 12-8 Server module in the detail view

Properties of the technology module

The object properties for the TM Timer DIDQ technology modules can be opened in the HW configuration via the context menu or by double-clicking the module in the detail view.

1. To open the context menu, select the TM Timer DIDQ technology module with the right mouse key.
2. Select Object properties. The following dialog box opens.

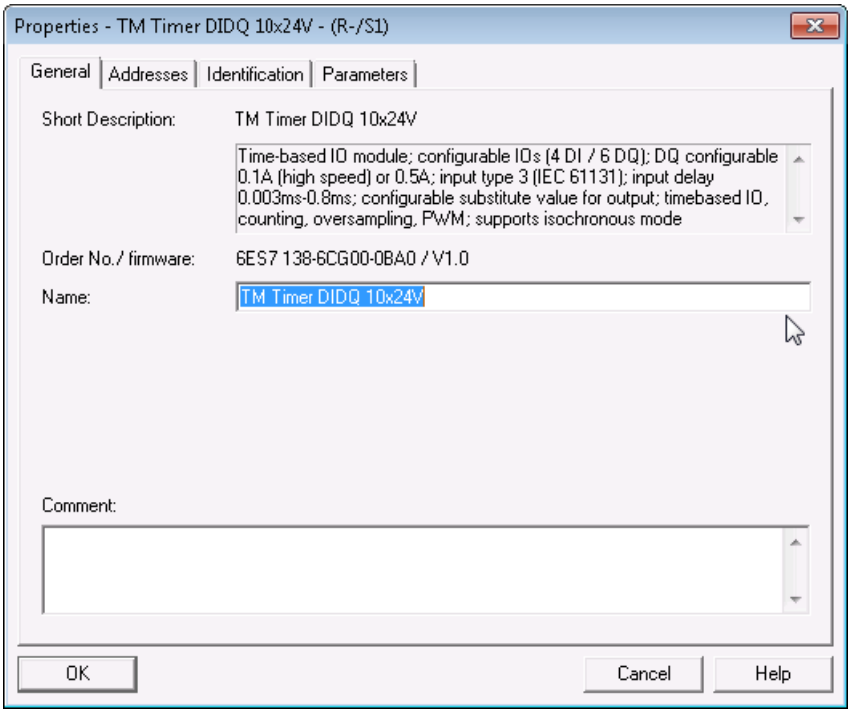


Figure 12-9 Properties of the TM Timer DIDQ 10x24V technology module

You can view or change the following properties in the object properties:

- General tab
  - Short designation, order number, name
    The short designation, the information below it, and the ordering number are identical to the entries in the Hardware Catalog window.
    In the Name field you can find the short description of the module, which can be changed according to your specifications. When you change the description, the new description is displayed in the configuration table.
  - Comments
    You can enter, for example, the application of the module in this field.
- Addresses tab
  Here you can assign the module one start address each for the inputs and outputs. The end addresses are calculated automatically.

- Identification tab
  - Plant ID of the module
  - Location ID of the module
  - Date on which the module was installed.
  - Further information: Free text, which is stored in the module.
- Parameters tab
  In this tab you parameterize the module, for example the configuration of the DQ/DI groups (see Setting options for TM Timer DIDQ 10x24V technology module (Page 8148)).

Configuring PROFINET communication

Preconditions

1. The respective ET 200SP interface module is initialized.
2. The ET 200SP interface module is integrated into the topology (prerequisite for operation in IRT mode).
3. The controller is configured as sync master (prerequisite for operation in IRT mode).

Note

For further information on configuration of the PROFINET communication please refer to the *SIMOTION SCOUT Communication* or *SIMATIC Communication* Manuals.

Operating the ET 200SP interface module and TM Timer DIDQ technology module in IRT mode

For use of the channels as cam output or measuring input input, the module must be synchronized with the controller (IRT mode).

To configure PROFINET communication on the PN interface module ET 200SP and on the TM Timer DIDQ technology module in **IRT mode**, please proceed as follows:

1. Configure the PROFINET IO topology.
2. Select the ET 200SP interface module in HW Config.
3. Select the **PN IO** line in the detail view of HW Config. Double-click to open the "Properties - PN IO" dialog box.

4. Switch to the "Synchronization" tab and select "Sync slave" for the **Synchronization role** and "IRT" for the **RT class**.

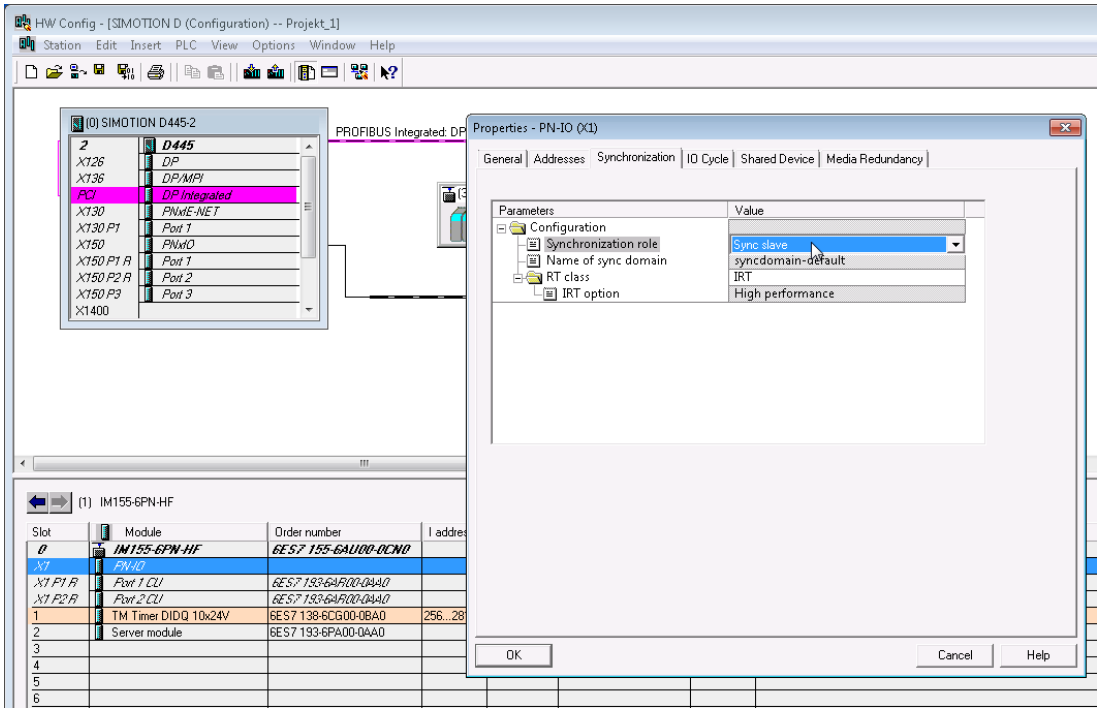


Figure 12-10 Configuring communication (IRT mode) - synchronization

5. Switch to the "IO cycle" tab and select the cycle "Servo" for **Assign IO device isochronously**.

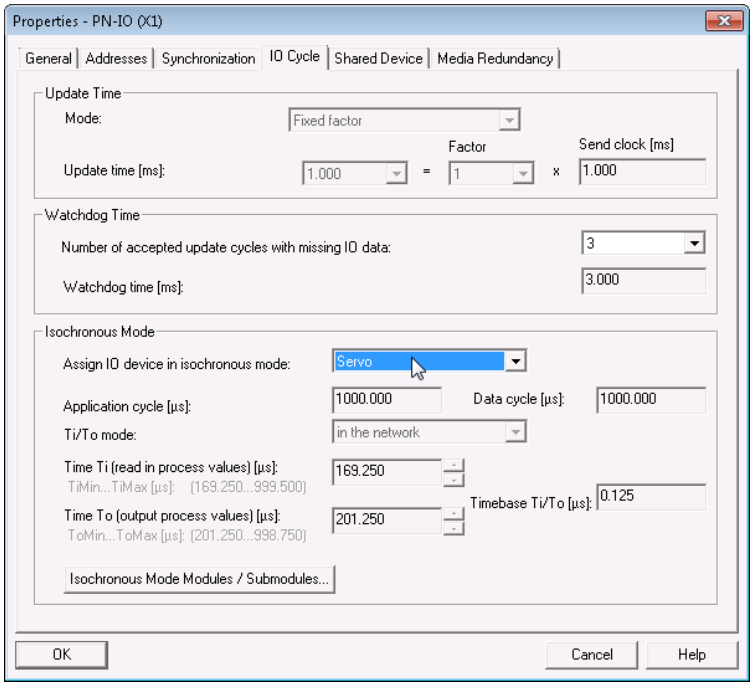


Figure 12-11 Configuring the communication - PROFINET IO cycle

Note

If you operating several ET 200SP interface modules and technology modules TM Timer DIDQ, you must set the isochronous mode at all of the modules.

Configuring I/O channels

Setting options for TM Timer DIDQ 10x24V technology module

Overview

The following pages provide an overview of the properties of the TM Timer DIDQ 10x24V technology module and the properties that you can define for the channels. Not all of the parameters are available depending on the settings.

Parameters	Value range	Default
Basic parameters		
PWM period for the digital outputs	• 10 ms • 5 ms • 2 ms • 1 ms • 0.5 ms • 0.2 ms	10 ms
Response to CPU STOP	• Output substitute value • Keep last value	Output substitute value
Enable diagnostics interrupts	• Deactivated • Activated	Deactivated
Channel parameters		
Configuration DQ/DI group (DQ0/DI0 or DQ1/DI1)	• Timer DQ with enable input • Use input/output individually	Timer DQ with enable input
Operating mode of the digital output	• Timer DQ • Oversampling • Pulse width modulation PWM	Timer DQ
Substitute value for the digital output	• 0 • 1	0
High-speed output (0.1 A)	• Deactivated • Activated	Activated
Inverting of the input or output signal	• Deactivated • Activated	Deactivated
HW enable via the digital input	• Level-controlled • Edge-controlled	Level-controlled

Parameters	Value range	Default
Level selection for HW enable	• Active at high level • Active at low level	Active at high level
Configuration DQ/DI group (DQ2/DI2/DI3)	• Incremental encoder (A, B phase shift) • Timer DI2 with enable input DI3 • Timer DQ2 with enable input DI2 • Use inputs individually	Incremental encoder (A, B phase shift)
Invert counting direction (incremental encoder)	• Deactivated • Activated	Deactivated
Operating mode of the digital output	• Numerator • Timer DI • Oversampling	Timer DI
Input delay for the digital input	• none • 0.05 ms • 0.1 ms • 0.4 ms • 0.8 ms	0.1 ms
Signal evaluation for the counter	• With rising edge • With falling edge	With rising edge

Further information

See Manual *ET 200SP TM Timer DIDQ 10x24V technology module*.

Configuring I/O channels

Procedure

To configure I/O channels as measuring input input (Timer DI) and output cam (Timer DQ) in the TM Timer DIDQ 10x24V technology module, proceed as follows.

1. Select the ET 200SP interface module in HW Config in the station window. Open the "Properties" dialog by double-clicking on the TM Timer DIDQ 10x24V technology module in the detail view.
2. You can configure the channels in the "Properties" dialog box of the technology module under the "Parameters" tab.
3. In the example, select the value "Use input/output individually" for the DQ0 / DI0 channel group from the "Configuration DQ/DI group" drop-down list. This allows the channels DQ0 and DI0 to be parameterized and used independently of one another.

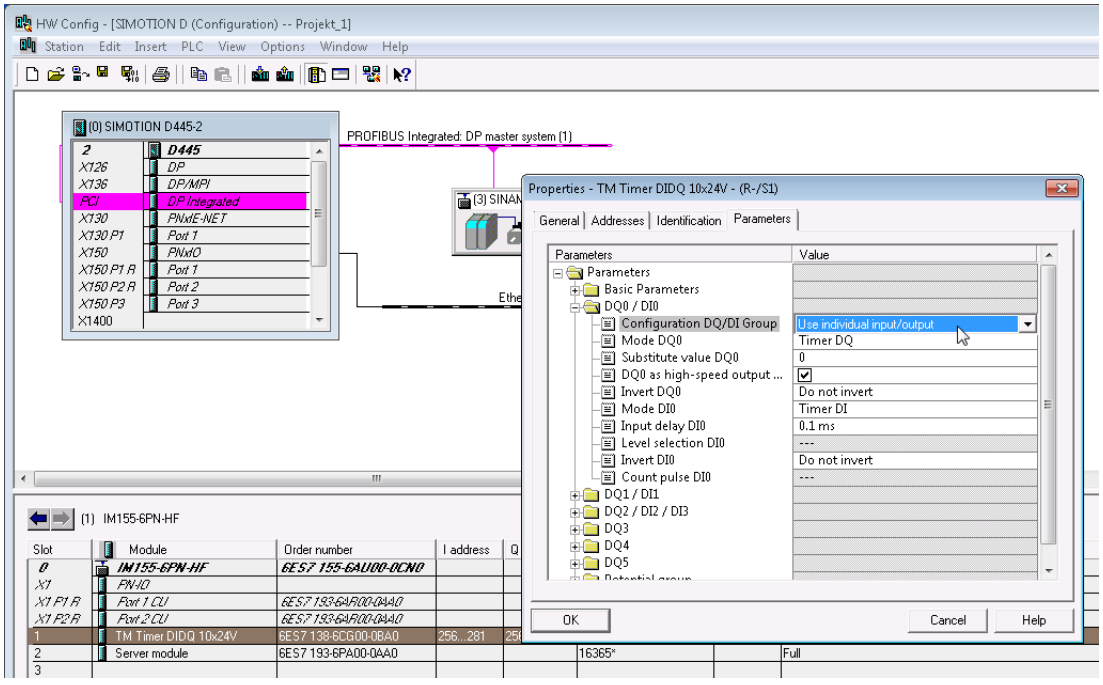


Figure 12-12 Configuring the TM Timer DIDQ 10x24V technology module

Note

This setting allows channel DQ0 to be interconnected as output cam and DI0 as measuring input under SIMOTION SCOUT.

Other information

Note

The technology modules and the individual channels may only be reparameterized in offline state. All changes become effective only after saving, compilation and eventual download of your changed user project to the controller (see Chapter Saving and compiling a user project with SCOUT/SCOUT TIA (Page 8178)).

Use of I/O channels

Introduction

After configuring the technology modules as described in section Configuring I/O channels (Page 8148), the channels can be used in your user project.

In Chapter Linking technology objects with TM Timer DIDQ (Page 8151) using an example, it is shown how channels **DQ0** (Timer DQ) and **DI0** (Timer DI) are interconnected with technology objects **TO cam** or **TO measuring input** .

In Chapter Linking symbolic I/O variables with TM Timer DIDQ (Page 8154), using an example, it is shown in the address list how a variable 'var_QI_MI' can be assigned via symbolic assignment of the quality information **QI** of the measuring input input.

You can find an overview of the I/O points of the channels in Chapter Overview of the I/O points of the channels (Page 8157).

Linking technology objects with TM Timer DIDQ

In the following example, the configured I/O channels (see section Configuring I/O channels (Page 8148)) are linked with one output cam technology object and with one measuring input technology object.

Cam outputs and measuring input inputs are assigned with the assignment partners **CAM_n** or **MI_n**.

Note

This assignment is only possible if Profinet communication was completely configured (see Creating a technology module and configuring communication (Page 8143)).

A general understanding of technology objects is therefore required to perform the required configurations.

Note

Technology objects may only access a TM Timer DIDQ technology module once the technology module has completely powered up (see Chapter Synchronization (Page 8177)). Otherwise, a technology alarm will be triggered.

TO outputCam

Note

Before you insert an output cam, a TO axis (position or synchronous axis) or a TO external encoder, to which the output cam is assigned, has to be created.

To insert a new output cam:

1. In the project navigator, highlight the OUTPUT CAMS folder under the relevant axis or external encoder.
2. Select Insert > Technology object > Output cam or double-click Insert output cam in the project navigator under the axis or external encoder in the OUTPUT CAMS folder. The Insert output cam window appears.
3. Enter a name for the output cam.
4. Confirm with OK. In the working area, the window for the configuration is displayed and the created output cam TO is shown in the project navigator.

In the Configuration window, define the configuration data values for the output cam.

1. Double-clicking in the project navigator below the output cam on the Configuration element displays the window in the working area.
2. Make the following settings:
   - Activate the "Activate output" checkbox.
   - Under the selection "Cam output on:" select entry "Output cam (CAM)".

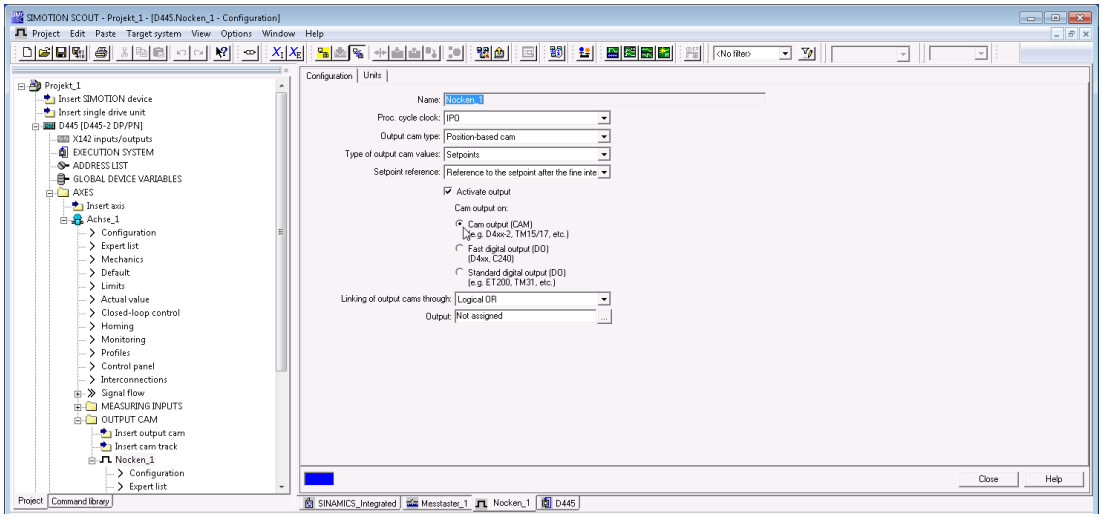


Figure 12-13 Configuring the TO output cam

3. Assign the assignment partner **CAM _0** of the technology module to the output.

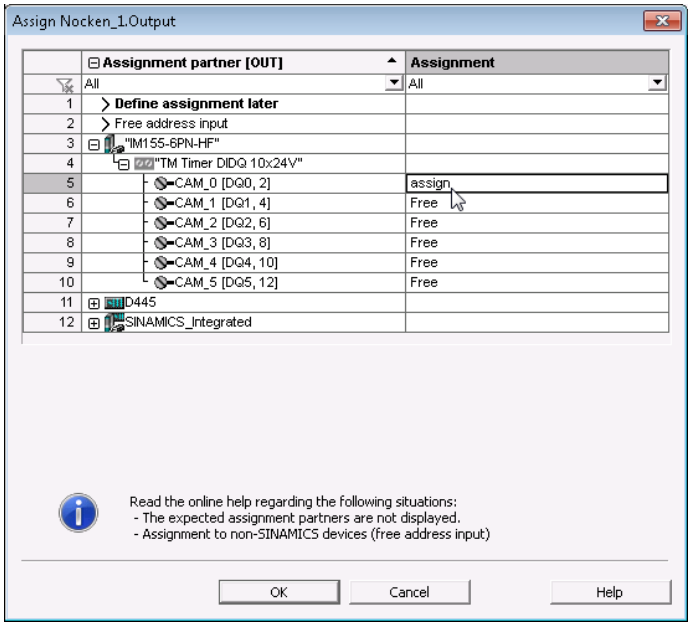


Figure 12-14 Assignment of cam output to TO output cam

4. Confirm with OK.

TO measuringInput

Note

Before you insert a measuring input, the hardware must have been configured – and a TO axis (position or synchronous axis) or a TO external encoder, to which the measuring input is assigned, has to be created.

This is how you enter a measuring input
1. In the project navigator, highlight the folder MEASURING INPUTS under the relevant axis or external encoder.

2. Select Insert > Technology Objects > Measuring Input, or double-click Insert Measuring Input in the project navigator at the axis or external encoder entry in the MEASURING INPUTS folder. The Insert Measuring Input window appears.

3. Enter a name for the measuring input.

4. Confirm with OK. In the working area, the window for the configuration is displayed and the measuring input created is shown in the project navigator.

In the Configuration window, define the configuration data values for the measuring input.

1. Double-clicking in the project navigator below the measuring input on the Configuration element displays the window in the working area.
2. Assign the assignment partner **MI _0** of the technology module to the input.

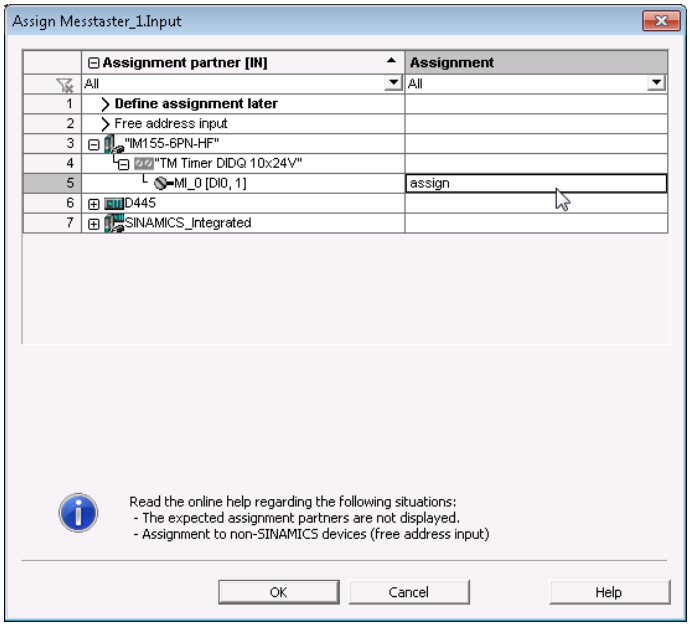


Figure 12-15 Assignment of measuring input input to TO measuringInput

3. Confirm with OK.

Further information

Further information on configuring the output cam and measuring input technology objects can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs Function Manual*.

See also

Service and maintenance (Page 8181)

Linking symbolic I/O variables with TM Timer DIDQ

You can address digital I/O channels using symbolic variables in the user program.

Precondition

The I/O channels are configured as described under Chapter Configuring I/O channels (Page 8148).

Procedure

In the following example, create variable 'var_QI_MI' and link this with quality information **QI** of the measuring input input **MI_0**.

1. Open the **address list** via the project navigator.
2. In the **View** field, select 'I/Os'.
3. Create the variable 'var_QI_MI', type "BOOL", and select 'IN' in the **I/O address** field.

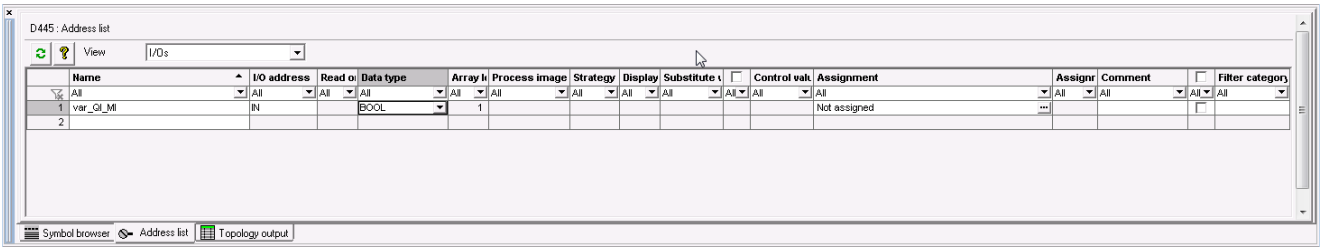


Figure 12-16 Creating variables in the address list

4. In the **Assignment** field, click on ...
5. First select the technology module in the following dialog.

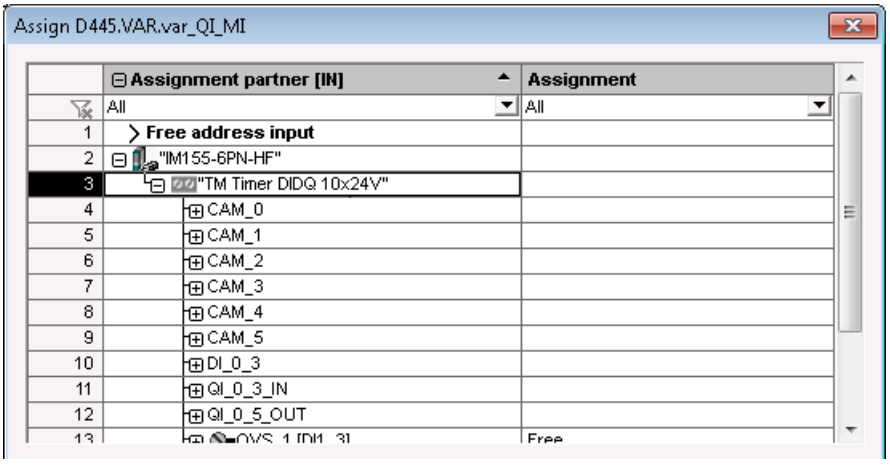


Figure 12-17 Dialog assignment - select module

6. Scroll downwards in the list of assignment partners and from the measuring input input, select the quality information **QI**.

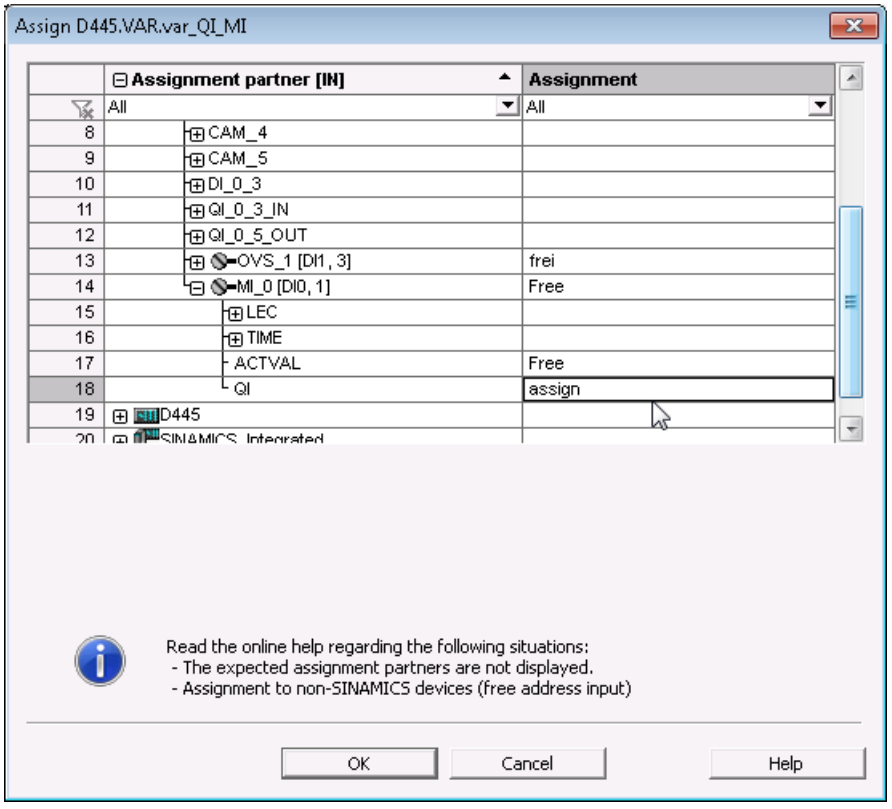


Figure 12-18 Dialog assignment - select QI of the measuring input input

7. Confirm the selection by clicking **OK**.

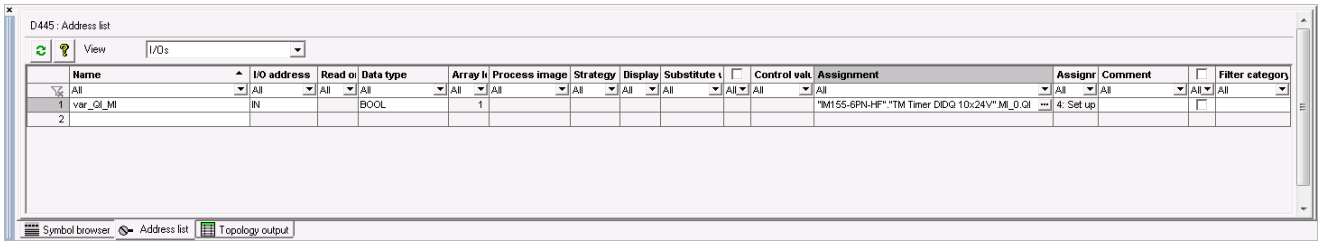


Figure 12-19 Address list following successful assignment

Note

The procedures for generating symbolic variables and for using the process image or direct I/O access are described in detail in the SIMOTION SCOUT Manual.

Overview of the I/O points of the channels

I/O points of the channels Timer DI and Timer DQ

Each I/O channel has different cyclic data in the form of I/O points.

I/O point	Input	Output	Function of the channel [1]	Description
ACTVAL	X		Timer DI, Timer DQ	Actual value of the input / reading back the channel status
QI	X		Timer DI, Timer DQ	Quality information
SETVAL		X	Timer DI, Timer DQ	Setpoint
TIME	X		Timer DI	Time stamp Timer DI
		X	Timer DQ	Time stamp Timer DQ
LEC	X		Timer DI	Counter for lost edges on Timer DI
SEL		X	Timer DI	Edge selection at Timer DI; is used by the TO measuring input
SETEN		X	Timer DI, Timer DQ	Enabling signal
EN	X		Timer DI, Timer DQ	Signal state of the enabling signal

1) Timer DI (measuring input), Timer DQ (output cam)

When using technology objects (interconnection of Timer DQ with TO outputCam/camTrack or Timer DI with TO measuringInput), the following I/O points are not relevant, because they are used or processed by technology objects:

- SEL
- TIME
- ACTVAL
- SETVAL

I/O points of the counter, oversampling, incremental encoder and pulse width modulation channels

Each I/O channel has different cyclic data in the form of I/O points.

I/O point	Input	Output	Channel function	Description
ACTVAL	X		Incremental encoder, oversampling, counter	Actual value of input
QI	X		Incremental encoder, oversampling, counter, pulse width modulation	Quality information
EVENTVAL	X		Incremental encoder, counter	Actual count value
REFVAL	X		Incremental encoder	
ACTSAMPLE	X		Oversampling	32 states of the digital input (oversampling pattern)

SETSAMPLE		X	Oversampling	Specification of the 32 states of the digital output (oversampling pattern)
		X	Pulse width modulation	Specification of pulse-pause ratio as percentage value

Further information

Further information on the I/O points of the channels can be found in the Manual *ET 200SP TM Timer DIDQ 10x24V Technology Module in* Chapter *Control and feedback signal interface*.

12.2.3.3 SIMOTION SCOUT TIA

Hardware and software requirements

The following requirements must be satisfied in order to be able to work with the TM Timer DIDQ technology modules.

Hardware requirements

- SIMATIC ET 200SP interface module, version HF/HS or SIMATIC ET 200MP interface module, version Standard/HF
- TM Timer DIDQ 10x24V or 16x24V technology module
- SIMOTION controller with PROFINET interface (see section System integration (Page 8137))
- PROFINET cables
- External 24 VDC power supply (note the requirements in the *SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V* or *SIMATIC ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V" Manual*)

Software requirements

Please observe the following requirements:

- SIMOTION software as of V4.4 HF6
- SIMATIC ET 200SP interface module, version HF or SIMATIC ET 200MP interface module in the Standard/HF version
- TIA Portal as of V13 SP1 as of Update 2

Requirements for operation

Preconditions

The following preconditions must be met before operating TM Timer DIDQ technology module.

1. Your system hardware including the technology module is installed and wired.
2. You have physically connected the technology module via an ET 200SP interface module or an ET 200MP interface module with a SIMOTION controller via the PROFINET IO interface (see Chapter System integration (Page 8137)).
3. You have created a project with a SIMOTION controller with PROFINET interface in the TIA Portal (sync master) and configured PROFINET IRT. The SIMOTION controller is configured for isochronous operation.
4. You have inserted an ET 200MP interface module, e.g. IM155-5 PN HF, assigned it to the SIMOTION controller, and configured it as sync slave with PROFINET IRT. The interface module is configured for isochronous operation.

   Note

   If you operating several ET 200SP/ET 200MP interface modules and technology modules TM Timer DIDQ, you must set the isochronous mode at all of the modules.

5. You have integrated the interface module into the PROFINET topology.

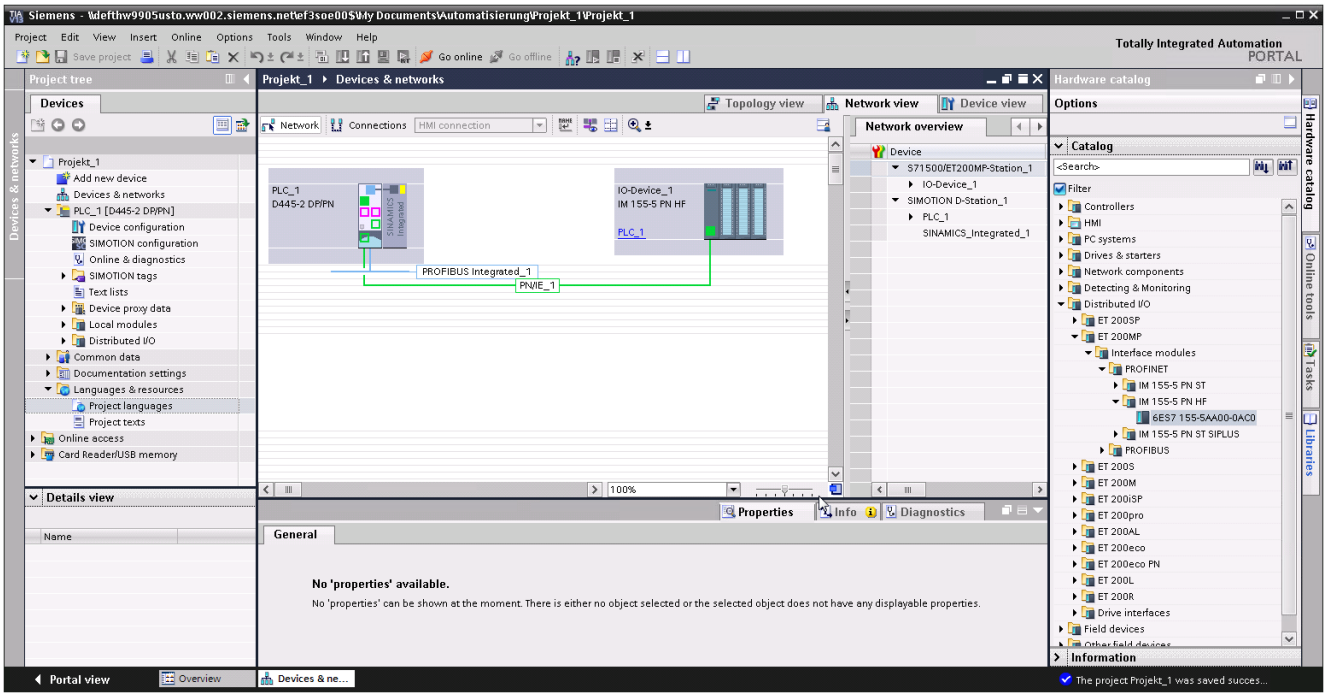


Figure 12-20 Project in the TIA Portal - network view

Note

For further information on configuration of the PROFINET communication please refer to the *SIMOTION SCOUT TIA* Configuration Manual.

Note

The procedures contained in this section assume that the user has a general understanding of TIA Portal or SCOUT TIA.

Note

Please note that interface module ET 200SP – in conjunction with technology module TM Timer DIDQ – can be operated with a minimum PROFINET clock cycle of 500 µs and the ET 200 MP interface module in conjunction with the TM Timer DIDQ technology module, with a minimum PROFINET clock cycle of 250 µs – depending on the quantity structure.

Creating and configuring a technology module

Procedure

1. Switch to the network view in the TIA Portal.
2. The device configuration opens by double-clicking on the ET 200MP interface module.

3. In the "Hardware Catalog" window, navigate to the TM Timer DIDQ 16x24V technology module via Technology modules -> Time-based IO.
4. Drag the technology module to the desired slot in the device view.

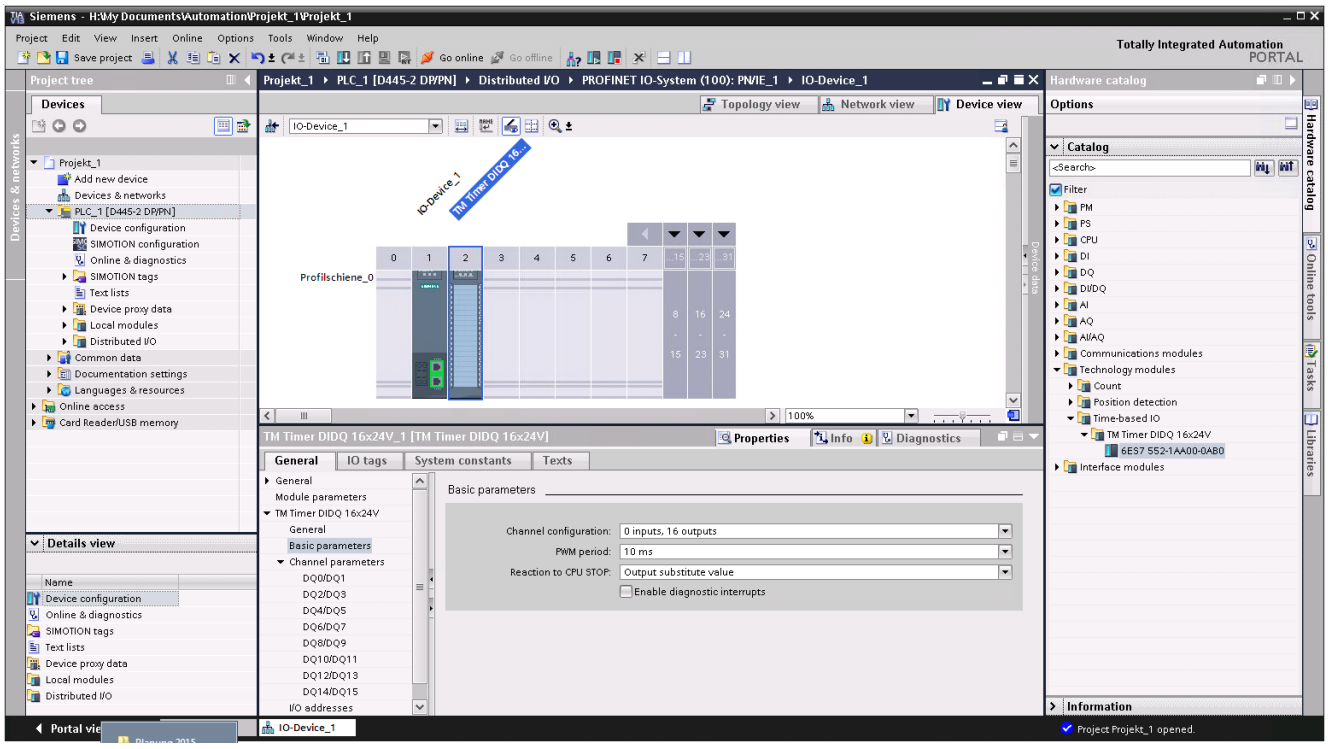


Figure 12-21 TM Timer DIDQ 16x24V technology module inserted in the project

Note

The interface module and the TM Timer DIDQ technology module must be configured for isochronous operation. Check the settings in the inspection window (device configuration of the ET 200MP interface module must be open) under tab PROFINET interface -> Extended options -> Isochronous mode.

Properties of the technology module

The object properties for the TM Timer DIDQ technology modules can be set in the inspection window (the technology module must be selected in the device view) in the "General" tab.

You can view or change the following properties in the object properties:

- General
  - Project information such as name, rack, slot, comment.
    In the Name field you can find the short description of the module, which can be changed according to your specifications. When you change the description, the new description is displayed in the configuration table.
  - Catalog information on the module
  - Identification & Maintenance, e.g. plant ID, date of installation

  Basic parameters
  - Channel configuration
  - PWM
  - Response to CPU STOP
  - Enable diagnostics interrupts
- Channel parameters
  In this tab you parameterize the module, for example the configuration of the DQ/DI groups (see Setting options for TM Timer DIDQ 10x24V technology module (Page 8165) and Setting options for TM Timer DIDQ 16x24V technology module (Page 8166)).
- I/O Addresses
  Here you can assign the module one start address each for the inputs and outputs. The end addresses are calculated automatically. In addition, you must assign the process image, e.g. servo, to the module.

Configuring the technology module

In the following example, a channel configuration with 4 inputs and 12 outputs is used for the TM Timer DIDQ 16x24V technology module.
The input and output addresses are assigned to the process image "Servo".

1. In the "Channel configuration" drop-down list under "Basic parameters", select the setting "4 inputs, 12 outputs".

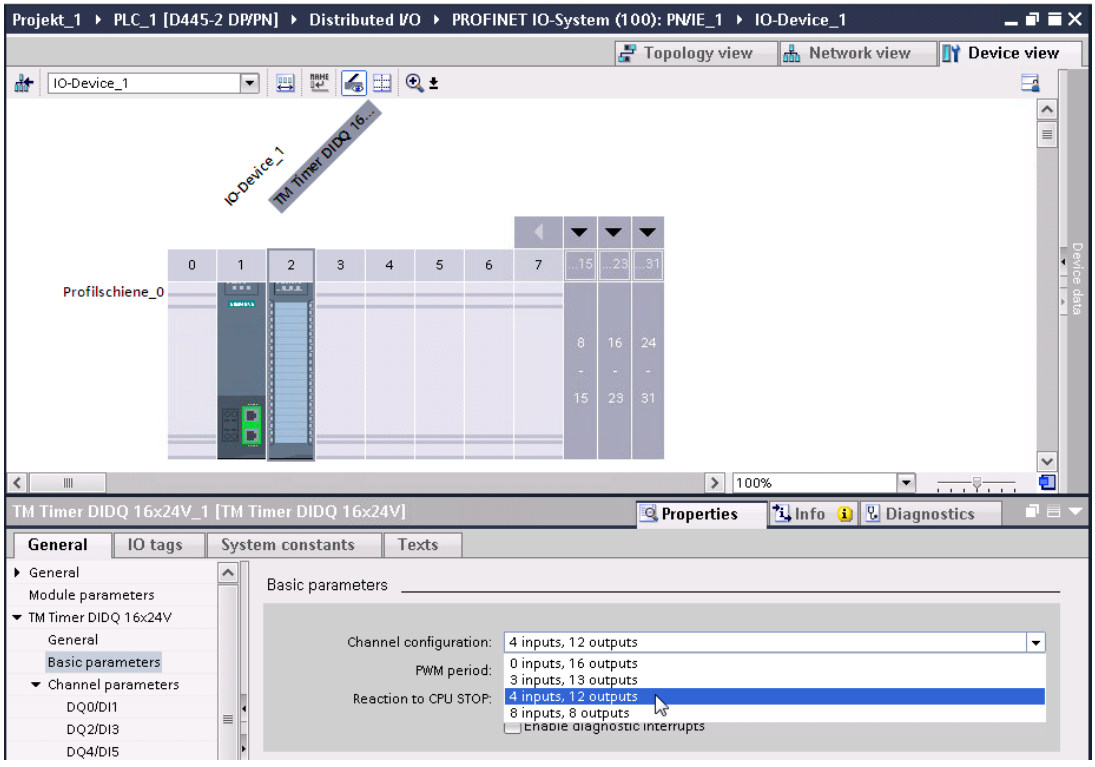


Figure 12-22 Basic parameters - channel configuration

2. In the "Process image" drop-down list under "I/O addresses", uncheck the "Servo" setting for each of the inputs and outputs.
   As start address of the input or output, only use addresses higher than 63; otherwise, when compiling the project, an error message will be output.

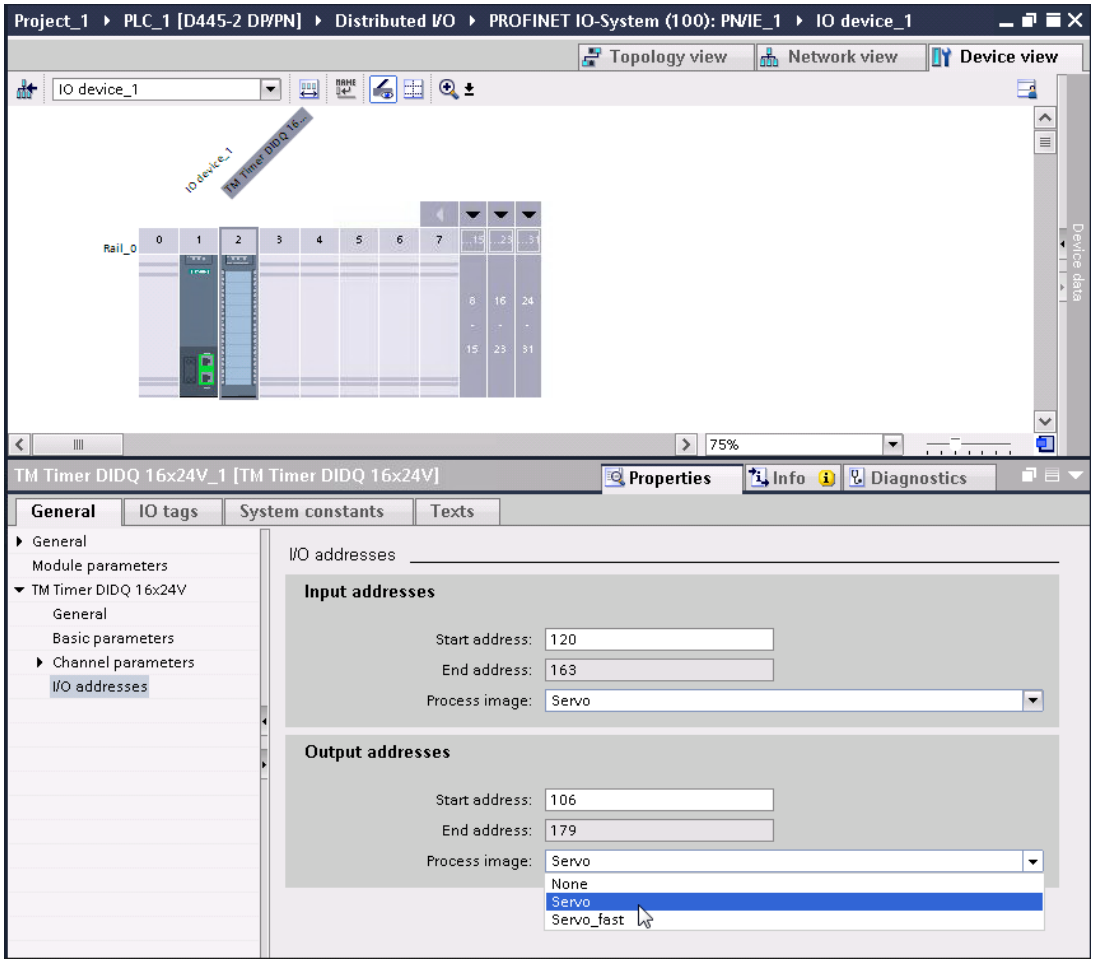


Figure 12-23 I/O addresses - process image

Configuring I/O channels

Setting options for TM Timer DIDQ 10x24V technology module

Overview

The following pages provide an overview of the properties of the TM Timer DIDQ 10x24V technology module and the properties that you can define for the channels. Not all of the parameters are available depending on the settings.

Parameters	Value range	Default
Basic parameters		
PWM period for the digital outputs	• 10 ms • 5 ms • 2 ms • 1 ms • 0.5 ms • 0.2 ms	10 ms
Response to CPU STOP	• Output substitute value • Keep last value	Output substitute value
Enable diagnostics interrupts	• Deactivated • Activated	Deactivated
Channel parameters		
Configuration DQ/DI group (DQ0/DI0 or DQ1/DI1)	• Timer DQ with enable input • Use input/output individually	Timer DQ with enable input
Operating mode of the digital output	• Timer DQ • Oversampling • Pulse width modulation PWM	Timer DQ
Substitute value for the digital output	• 0 • 1	0
High-speed output (0.1 A)	• Deactivated • Activated	Activated
Inverting of the input or output signal	• Deactivated • Activated	Deactivated
HW enable via the digital input	• Level-controlled • Edge-controlled	Level-controlled
Level selection for HW enable	• Active at high level • Active at low level	Active at high level
Configuration DQ/DI group (DQ2/DI2/DI3)	• Incremental encoder (A, B phase shift) • Timer DI2 with enable input DI3 • Timer DQ2 with enable input DI2 • Use inputs individually	Incremental encoder (A, B phase shift)

Parameters	Value range	Default
Invert counting direction (incremental encoder)	• Deactivated • Activated	Deactivated
Operating mode of the digital output	• Numerator • Timer DI • Oversampling	Timer DI
Input delay for the digital input	• none • 0.05 ms • 0.1 ms • 0.4 ms • 0.8 ms	0.1 ms
Signal evaluation for the counter	• With rising edge • With falling edge	With rising edge

Further information

See Manual *ET 200SP TM Timer DIDQ 10x24V technology module*.

Setting options for TM Timer DIDQ 16x24V technology module

Overview

The following pages provide an overview of the TM Timer DIDQ 16x24V technology module properties and the properties that you can define for the channels. Not all of the parameters are available depending on the settings.

Parameter	Value range	Default
Basic parameters		
Channel configuration of the module	• 0 inputs, 16 outputs • 3 inputs, 13 outputs • 4 inputs, 12 outputs • 8 inputs, 8 outputs See also section "Additional digital inputs without technology function".	0 inputs, 16 outputs
PWM period for the digital outputs	• 10 ms • 5 ms • 2 ms • 1 ms • 0.5 ms • 0.2 ms	10 ms
Response to CPU STOP	• Output substitute value • Keep last value	Output substitute value

Parameter	Value range	Default
Enable diagnostics interrupts	• Deactivated • Activated	Deactivated
Channel parameters		
Operating mode of the digital output	• Timer DQ • Oversampling • Pulse width modulation PWM	Timer DQ
Substitute value for the digital output	• 0 • 1	0
High-speed output (0.1 A)	• Deactivated • Activated	Activated
Inverting of the input or output signal	• Deactivated • Activated	Deactivated
HW enable via the digital input	• Level-controlled • Edge-controlled	Level-controlled
Level selection for HW enable	• Active at high level • Active at low level	Active at high level
Configuration DI group	• Incremental encoder (A, B phase shift) • Timer DI with enable input • Use inputs individually	Incremental encoder (A, B phase shift)
Invert counting direction (incremental encoder)	• Deactivated • Activated	Deactivated
Operating mode of the digital output	• Numerator • Timer DI • Oversampling	Timer DI
Input delay for the digital input	• none • 0.05 ms • 0.1 ms • 0.4 ms • 0.8 ms	0.1 ms
Signal evaluation for the counter	• With rising edge • With falling edge	With rising edge
Configuration DQ/DI group	• Timer DQ with enable input • Use input/output individually	Timer DQ with enable input

Additional digital inputs without technology function

Channel configuration	Number of additional digital inputs without technology function
0 inputs, 16 outputs	8
3 inputs, 13 outputs	5

Channel configuration	Number of additional digital inputs without technology function
4 inputs, 12 outputs	4
8 inputs, 8 outputs	0

Unused inputs are available as digital inputs without technology function. The input delay of an input parameter is parameterized on the associated output channel.

Further information

See *SIMATIC ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V* Manual.

Configuring I/O channels

Procedure

To configure I/O channels as measuring input input (Timer DI) and output cam (Timer DQ) in the TM Timer DIDQ 16x24V technology module,
proceed as follows.

1. The technology module is selected in the device view. To configure the I/O channels, click on channel parameters in the "General" tab in the inspection window.
2. In the example, select the value "Use input/output individually" for the DQ0 / DI1 channel group from the "Configuration DQ/DI group" drop-down list. This allows the channels DQ0 and DI1 to be parameterized and used independently of one another.

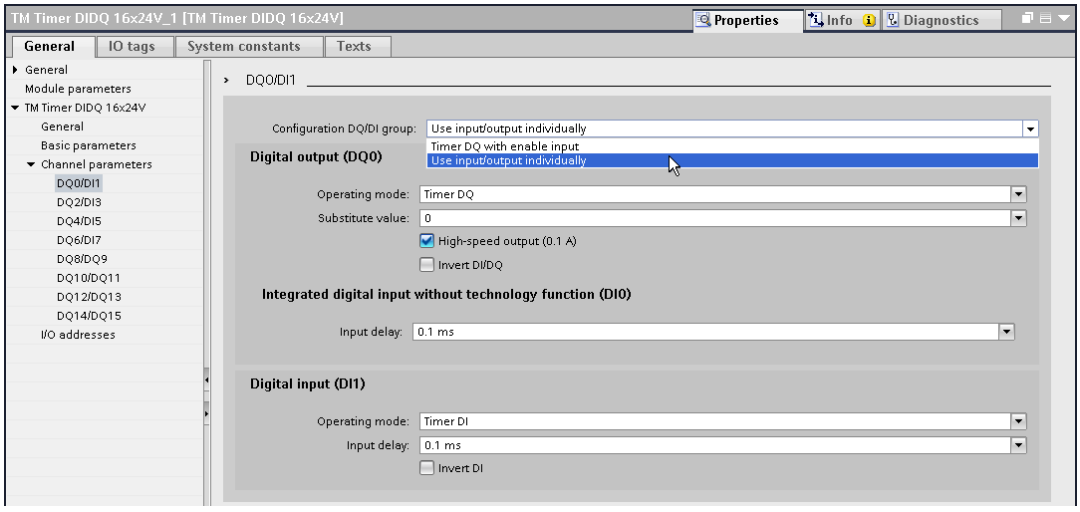


Figure 12-24 Channel parameter - configuration DQ/DI group

In addition to the (DQ0) digital output and the (DI1) digital input, the "Integrated digital input without technology function (DI0)" is also available.

Note

The additional "Integrated digital input without technology function (DI0)" can also be interconnected in SIMOTION SCOUT TIA.

3. Save and compile the project.
4. To interconnect the channels with the technology objects outputCam or measuringInput, open the SIMOTION configuration.

Note

This setting allows channel DQ0 to be interconnected as output cam and DI1 as measuring input input under SIMOTION SCOUT TIA.

Other information

Note

The technology modules and the individual channels may only be reparameterized in offline state. All changes become effective only after saving, compilation and eventual download of your changed user project to the controller (see Chapter Saving and compiling a user project with SCOUT/SCOUT TIA (Page 8178)).

Use of I/O channels

Introduction

After configuring the technology modules as described in section Configuring I/O channels (Page 8165), the channels can be used in your user project.

In section Linking technology objects with TM Timer DIDQ (Page 8169) using an example, it is shown how channels **DQ0** (Timer DQ) and **DI1** (Timer DI) are interconnected with technology objects **TO cam** or **TO measuring input** .

Chapter Linking symbolic I/O variables with TM Timer DIDQ (Page 8172) uses an example to show how in the address list of a 'var_DI' variable the symbolic assignment of the **DI_16** digital input (additional digital input without technology function (DI0)) and a 'var_QI_MI' variable are used to assign the **QI** quality information of the **MI_1** measurement sensing input.

You can find an overview of the I/O points of the channels in section Overview of the I/O points of the channels (Page 8176).

Linking technology objects with TM Timer DIDQ

In the following example, the configured I/O channels (see section Configuring I/O channels (Page 8165)) are linked with one output cam technology object and with one measuring input technology object.

Cam outputs and measuring input inputs are assigned with the assignment partners **CAM_n** or **MI_n**.

Note

This assignment is only possible if Profinet communication was completely configured (see Chapter Requirements for operation (Page 8159) and Creating and configuring a technology module (Page 8160)).

A general understanding of technology objects is therefore required to perform the required configurations.

Note

Technology objects may only access a TM Timer DIDQ technology module once the technology module has completely powered up (see Chapter Synchronization (Page 8177)). Otherwise, a technology alarm will be triggered.

TO outputCam

Note

Before you insert an output cam, a TO axis (position or synchronous axis) or a TO external encoder, to which the output cam is assigned, has to be created.

To insert a new output cam:

1. In the project navigator, highlight the OUTPUT CAMS folder under the relevant axis or external encoder.
2. Select Insert > Technology object > Output cam or double-click Insert output cam in the project navigator under the axis or external encoder in the OUTPUT CAMS folder. The Insert output cam window appears.
3. Enter a name for the output cam.
4. Confirm with OK. In the working area, the window for the configuration is displayed and the created output cam TO is shown in the project navigator.

In the Configuration window, define the configuration data values for the output cam.

1. Double-clicking in the project navigator below the output cam on the Configuration element displays the window in the working area.
2. Make the following settings:
   - Activate the "Activate output" checkbox.
   - Under the selection "Cam output on:" select entry "Output cam (CAM)".

3. Assign the assignment partner **CAM _0** of the technology module to the output.

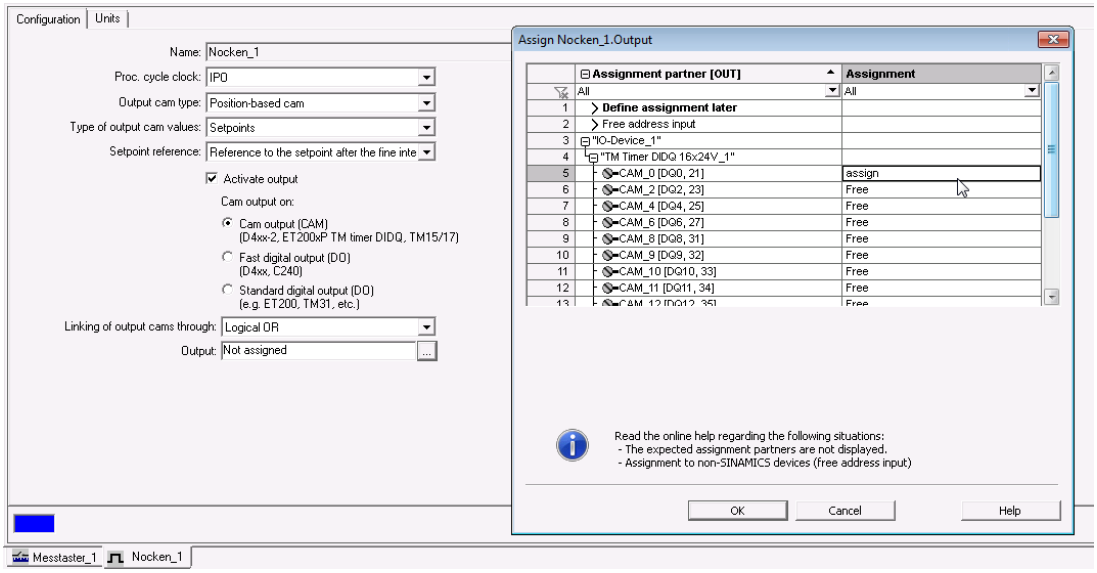


Figure 12-25 Assignment of cam output to TO output cam

4. Confirm with OK.

TO measuringInput

Note

Before you insert a measuring input, the hardware must have been configured – and a TO axis (position or synchronous axis) or a TO external encoder, to which the measuring input is assigned, has to be created.

This is how you enter a measuring input
1. In the project navigator, highlight the folder MEASURING INPUTS under the relevant axis or external encoder.

2. Select Insert > Technology Objects > Measuring Input, or double-click Insert Measuring Input in the project navigator at the axis or external encoder entry in the MEASURING INPUTS folder. The Insert Measuring Input window appears.

3. Enter a name for the measuring input.

4. Confirm with OK. In the working area, the window for the configuration is displayed and the measuring input created is shown in the project navigator.

In the Configuration window, define the configuration data values for the measuring input.

1. Double-clicking in the project navigator below the measuring input on the Configuration element displays the window in the working area.
2. Assign the assignment partner **MI _1** of the technology module to the input.

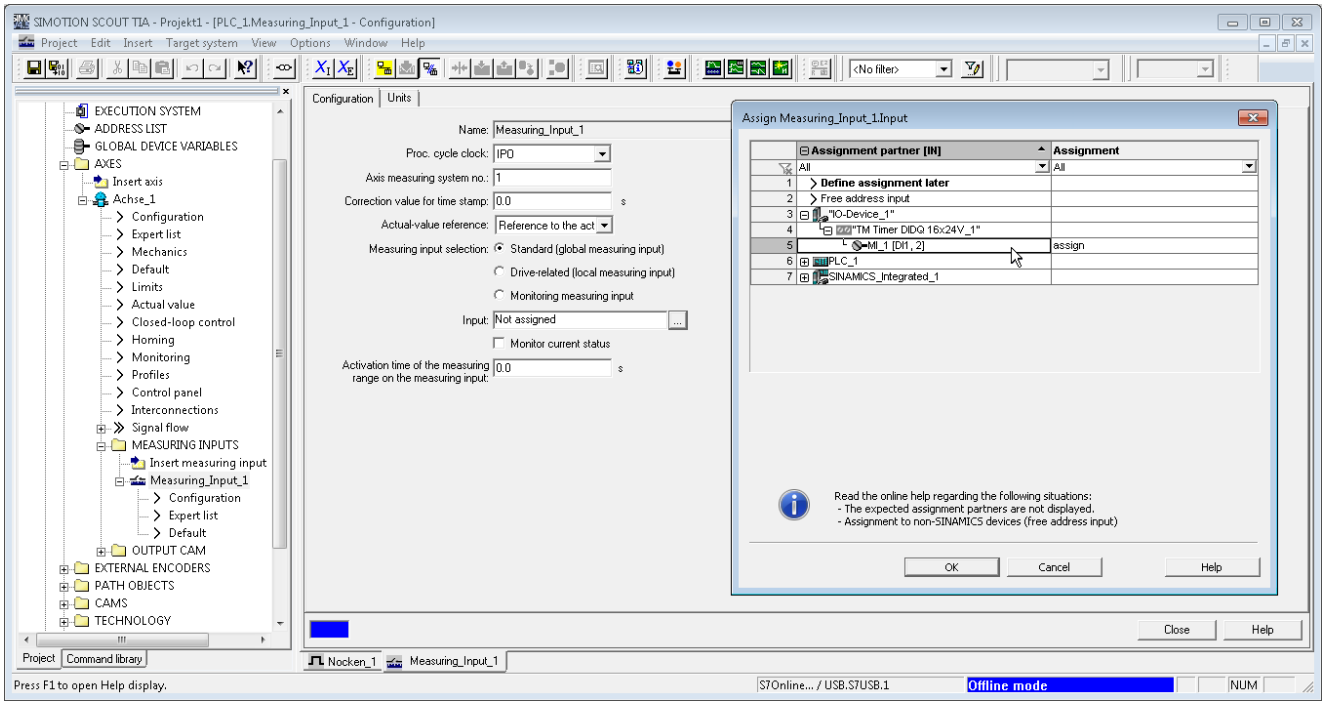


Figure 12-26 Assignment of measuring input input to TO measuringInput

3. Confirm with OK.

Further information

Further information on configuring the output cam and measuring input technology objects can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs Function Manual*.

See also

Service and maintenance (Page 8181)

Linking symbolic I/O variables with TM Timer DIDQ

You can address digital I/O channels using symbolic variables in the user program.

Requirement

The I/O channels are configured as described under section Configuring I/O channels (Page 8165).

Procedure

In the following example, create the 'var_DI' and 'var_QI_MI' variables. Link the 'var_DI' variable with the **DI_16** digital input (additional digital input without technology function (DI0)), and the 'var_QI_MI' variable with the **QI** quality information of the **MI_1** measuring input input.

1. Open the **address list** via the project navigator.
2. In the **View** field, select 'I/Os'.
3. Create the'var_DI' and 'var_QI_MI' variables of BOOL type, and select 'IN' in each **I/O address** field.

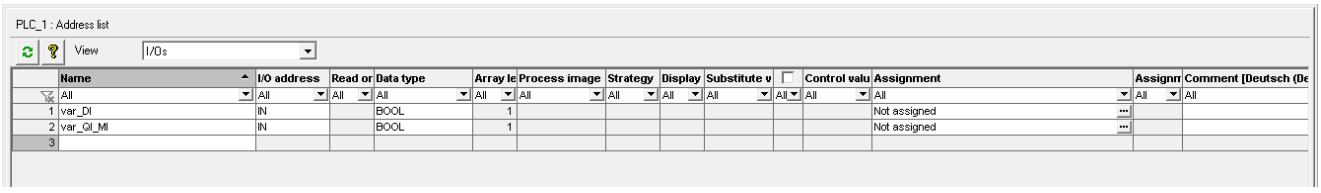


Figure 12-27 Creating variables in the address list

4. Click ... for the 'var_DI' variable in the **Assignment** field
5. First select the technology module in the following dialog.

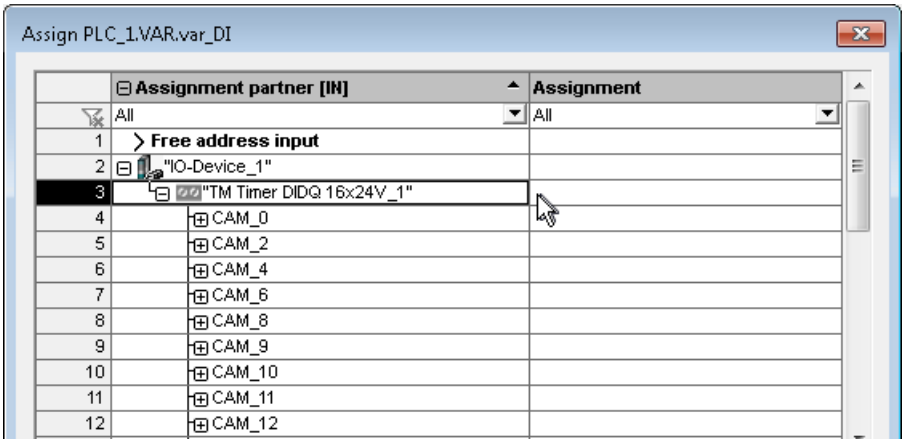


Figure 12-28 Dialog assignment - select module

6. Scroll downwards in the list of assignment partners and select the **DI_16** digital input.

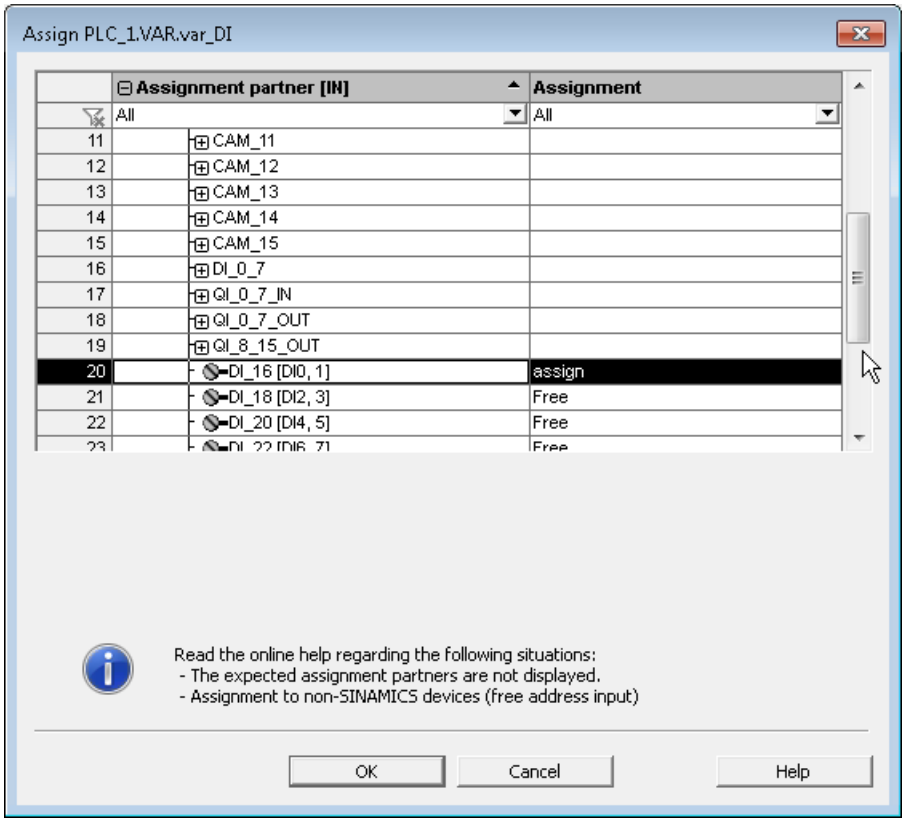


Figure 12-29 Dialog assignment - select DI (without technology function)

7. Confirm the selection by clicking **OK**.
8. Click ... for the 'var_QI_MI'variable in the **Assignment** field

9. Scroll downwards in the list of assignment partners and select the **QI** quality information from the **MI_1** measuring input input.

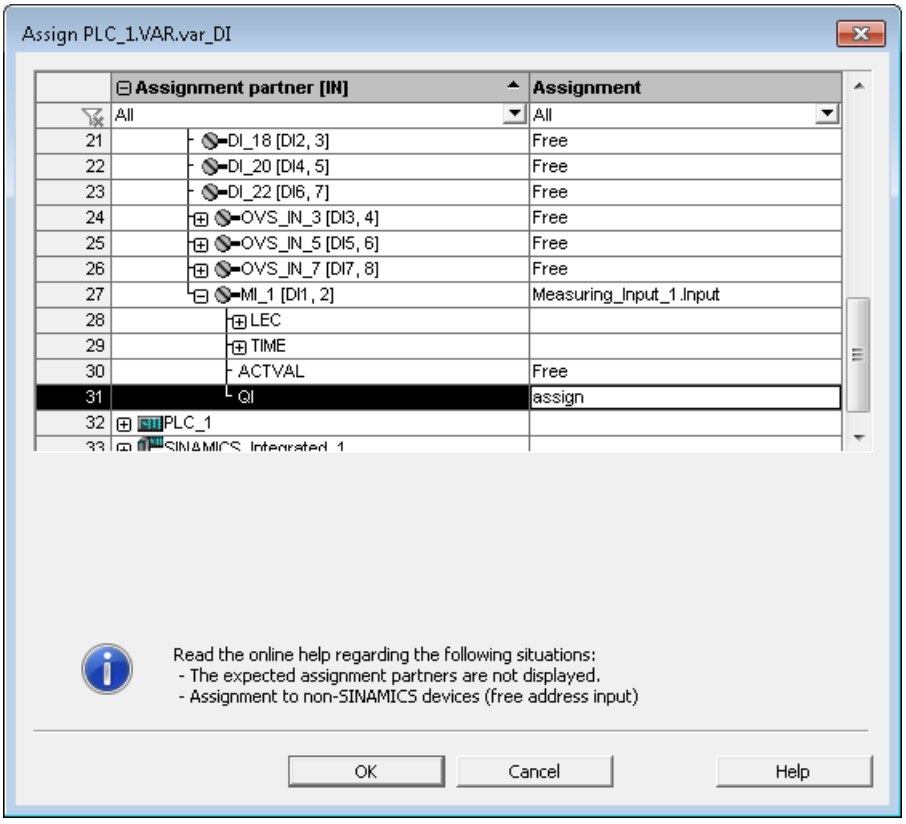


Figure 12-30 Dialog assignment - select QI of the measuring input input

10. Confirm the selection by clicking **OK**.

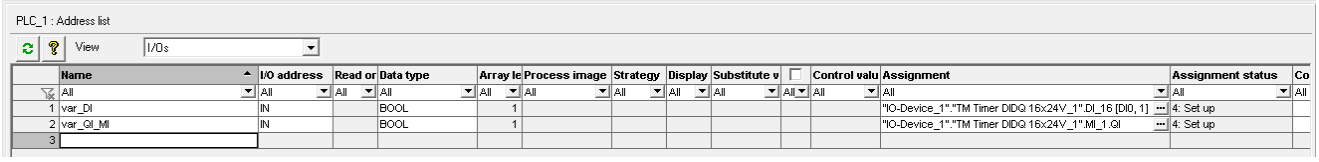


Figure 12-31 Address list following successful assignment

Note

The procedures for generating symbolic variables and for using the process image or direct I/O access are described in detail in the SIMOTION SCOUT TIA Manual.

Overview of the I/O points of the channels

I/O points of the channels Timer DI and Timer DQ

Each I/O channel has different cyclic data in the form of I/O points.

I/O point	Input	Output	Function of the channel [1)]	Description
ACTVAL	X		Timer DI, Timer DQ	Actual value of the input / reading back the channel status
QI	X		Timer DI, Timer DQ	Quality information
SETVAL		X	Timer DI, Timer DQ	Setpoint
TIME	X		Timer DI	Time stamp Timer DI
		X	Timer DQ	Time stamp Timer DQ
LEC	X		Timer DI	Counter for lost edges on Timer DI
SEL		X	Timer DI	Edge selection at Timer DI; is used by the TO measuring input
SETEN		X	Timer DI, Timer DQ	Enabling signal
EN	X		Timer DI, Timer DQ	Signal state of the enabling signal

1) Timer DI (measuring input), Timer DQ (output cam)

When using technology objects (interconnection of Timer DQ with TO outputCam/camTrack or Timer DI with TO measuringInput), the following I/O points are not relevant, because they are used or processed by technology objects:

- SEL
- ACTVAL
- SETVAL
- TIME

I/O points of the counter, oversampling, incremental encoder and pulse width modulation channels

Each I/O channel has different cyclic data in the form of I/O points.

I/O point	Input	Output	Channel function	Description
ACTVAL	X		Incremental encoder, oversampling, counter	Actual value of input
QI	X		Incremental encoder, oversampling, counter, pulse width modulation	Quality information
EVENTVAL	X		Incremental encoder, counter	Actual count value
REFVAL	X		Incremental encoder	
ACTSAMPLE	X		Oversampling	32 states of the digital input (oversampling pattern)

SETSAMPLE		X	Oversampling	Specification of the 32 states of the digital output (oversampling pattern)
		X	Pulse width modulation	Specification of pulse-pause ratio as percentage value

Further information

Further information on the I/O points of the channels can be found in the *SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V*and *SIMATIC ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V*Manuals, in Chapter *Control and feedback signal interface*.

12.2.3.4 Synchronization

After recording the connection between the controller and the TM Timer DIDQ technology module (the controller is switched on and powered up, the header module (IM) is switched on and powered up, the TM Timer DIDQ technology module is switched on and powered up, establishing a connection from the controller to the header module (IM) and to the TM Timer DIDQ technology module was successful), the TM Timer DIDQ is parameterized by the controller and the header module (IM) synchronizes to the PN cycle clock.

The cyclic connection is thus established, the cyclic I/O data of the TM Timer DIDQ technology module is now available in the controller (for access by the user program or for technology object access).

Note

The TM Timer DIDQ technology module must only accessed via I/O access if the header module (IM) and the technology module TM Timer DIDQ have been ramped-up. Otherwise, the CPU switches to STOP mode due to an I/O access error during I/O access.

The synchronization can be monitored via the PeripheralFaultTask.

Monitoring of synchronization between a technology module and a SIMOTION controller with PeripheralFaultTask

During the transition from START-UP to RUN, all technology modules with parameterized technology functions are in the "NOT_SYNCHRONIZED" state.

- As soon as the Terminal Modules have been successfully synchronized, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_SYNCHRONIZED" (=214) is called.
- If synchronization fails, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_NOT_SYNCHRONIZED" (=215) is called.

Example

To synchronize the user task, a user variable **TM_SYNC** is set to **FALSE** in the StartUpTask and set to **TRUE** in the PeripheralFaultTask with interrupt ID = SC_IO_MODULE_SYNCHRONIZED. The

status of **TM_SYNC** is queried in the user task before the (first) direct access to the technology module.

The following TaskStartInfo is supplied in the PeripheralFaultTask every time it is called:

```
UDINT   TSI#interruptID     // scan for triggering event
DINT    TSI#logBaseAdrIn    // valid only when not equal to _SC_INVALID_ADDRESS
DINT    TSI#logBaseAdrOut   // valid only when not equal to _SC_INVALID_ADDRESS
DINT    TSI#logDiagAdr      // valid only when not equal to _SC_INVALID_ADDRESS
DWORD   TSI#details         // set to 0
UINT    TSI#eventClass      // set to 0
UINT    TSI#faultId         // set to 0
```

The TaskStartInfo contains the logical address of the relevant technology module.

TSI#logDiagAdr, TSI#details, TSI#eventClass and TSI#faultId do not have any significance for technology modules.

Further information

Further information on synchronization can be found in the SIMATIC manuals, see SIMATIC documentation (Page 8135).

For further information on the TaskStartInfo, refer to the *SIMOTION ST Structured Text* Manual.

12.2.3.5 Saving and compiling a user project with SCOUT/SCOUT TIA

To save and compile with HW Config or SIMOTION SCOUT, proceed as described below:

Saving and compiling with HW Config

1. Save and compile the project via the menu Station -> Save and compile.
2. Before downloading the hardware configuration, you can select the "Check consistency" option in the Station menu. In this way, you can check whether the hardware configuration has been defined without errors, e.g. with regard to PROFINET settings, etc.
3. Transfer the project to the controller.

Saving and compiling with SIMOTION SCOUT/SCOUT TIA

1. Save and compile the project, e.g. via the Project -> Save and recompile changes menu.
2. You can select the "Check consistency" option in the Project menu of the main SIMOTION SCOUT/SCOUT TIA menu at any time. (For more information, refer to the SCOUT online help.) In this way, you can check whether the configuration, parameterization and programming in SIMOTION SCOUT/SCOUT TIA matches the defined hardware configuration.
3. Transfer the project to the controller.

12.2.3.6 Notes for those changing over

When changing over from the TM15/TM17 High Feature Terminal Modules to the TM Timer DIDQ technology modules, the following points must be worked through:

- Modifications must be made in the control cabinet and interconnections adapted.
- The project must be adapted analogously to the procedure described in Chapter Configuring.
- PROFINET communication with isochronous mode must be configured.

Note

The connection to the SIMOTION hardware cannot be realized using DRIVE-CLiQ as is the case for TM15/TM17 High Feature; PROFINET must always be used.

12.2.4 Functions

12.2.4.1 Overview

For the TM Timer DIDQ technology modules, you can evaluate the signals at the digital inputs and outputs for the following functions:

Function	Designation channel parameterization in HW Config/ HWCN	Access point in SCOUT/SCOUT TIA	Use in SCOUT/SCOUT TIA
Inputs			
Measuring input	Timer DI	MI_n	Interconnectable with SIMOTION TO measuringInput and IO variable
Oversampling	Oversampling	OVS_IN_n	Interconnectable with IO variable
Numerator	Numerator	EPC_n	Interconnectable with IO variable
Incremental encoder	Incremental encoder	EPC_n	Interconnectable with IO variable
Outputs			
Output cam	Timer DQ	CAM_n	Interconnectable with SIMOTION TO outputCam/camTrack and IO variable
Pulse width modulation	Pulse width modulation	PWM_n	Interconnectable with IO variable
Oversampling	Oversampling	OVS_OUT_n	Interconnectable with IO variable

Further information

- Additional information on the functions of the technology modules TM Timer DIDQ is provided in the *SIMATIC ET 200SP technology module TM Timer DIDQ 10x24V* and *SIMATIC ET 200MP/S7-1500 technology module TM Timer DIDQ 16x24V* Manuals.
- Further information on configuring output cams and measuring inputs can be found in the *SIMOTION Motion Control Output Cams and Measuring Inputs Function Manual*.

12.2.4.2 HW enable for the channels Timer DI and Timer DQ

Precondition

You have parameterized a channel at the TM Timer DIDQ as "Timer DQ with enabling input" or "Timer DI with enabling input".

Edge-controlled HW enable

With an edge-controlled enable, output cams are output at the output only if the enable status exists before the output cam start. The enable status is controlled by a configured Measuring Input Technology Object.

For the edge-controlled enable, the technology object measuring input is only measured if enabled. The enable status is controlled by a configured measuring input technology object linked with the enable channel.

For the measuring input TO, only the "one-time measuring" mode is permitted with the following edge detection:

- Rising edge
- Falling edge
- Both edges

Measuring of "Both edges, first rising" and "Both edges, first falling" and the "Cyclic measuring" mode is not supported.

Via the measuring input TO

- the positions of the enable edges can be evaluated (see SIMOTION Motion Control - Output Cams and Measuring Inputs" Function Manual). Please note that the measuring range is monitored in the IPO, IPO2 or position control cycle clock, depending on the configured measuring input cycle clock.
- Set or reset the enable status.
  The enable status is set when the configured edge arrives.
  The enable status is reset
  - At the beginning of the measuring range configured at the Measuring Input TO (with range setting: "Measure in specified range")
  - using the program command _enableMeasuringInput (with range setting: "Measure without a defined range")

The enable status can be monitored via the I/O area of the enable input.

If the enable status is retracted before the output cam ends, the output cam is still output until the end if it has already started. New output cams are no longer output.

Level-controlled HW enable

Note

Information on level-controlled hardware enable is provided in the *SIMATIC ET 200SP technology module TM Timer DIDQ 10x24V* and *SIMATIC ET 200MP/S7-1500 technology module TM Timer DIDQ 16x24V* in Chapter *Functions*.

12.2.5 Service and maintenance

12.2.5.1 Replacing a technology module

Information on replacing a TM Timer DIDQ technology module can be found in the system manuals *SIMATIC ET 200SP distributed I/O system* and *ET 200MP distributed I/O system*, Chapter Maintenance -> Replacing an I/O module.

12.2.5.2 Firmware update

Information on the firmware update can be found in the system manuals *SIMATIC ET 200SP distributed I/O system* and *ET 200MP distributed I/O system,* Chapter Maintenance -> Firmware update.

12.2.6 Diagnostics

12.2.6.1 Overview of the diagnostic possibilities

The following diagnostics options are available to you in the TM Timer DIDQ technology modules:

- LEDs (module)
- Quality information (telegram level)
- HW Config/HWCN (engineering system)

Further information

Further information on the diagnostics options can be found in the *SIMATIC ET 200SP Technology Module TM Timer DIDQ 10x24V*and *S7-1500/ET 200MP Technology Module TM Timer DIDQ 16x24V Manuals*.

12.3 TM15 / TM17 High Feature Operating Manual

Preface

Scope

This manual describes the supplementary TM15 / TM17 High Feature Terminal Modules in operation under SIMOTION D or under SINAMICS S120 in connection with SIMOTION C, P or D.

This manual is aimed at machine manufacturers, plant engineers, commissioning personnel, and service personnel who use SIMOTION in connection with SINAMICS.

Contents of the commissioning manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- Description
  This section describes the overall use of the TM15 and TM17 High Feature modules.
- Configuration/programming
  This section describes step-by-step instructions for configuration and integration into your user program.
- Commissioning
  This section details what is required at power up and what should be observed during the power up process.
- Error messages
  This section provides diagnostic information regarding possible faults.
- Application tips
  This section contains practical information for interconnection and use of the Terminal Modules.
- Appendix
  This chapter provides factual information (e.g. standards, approvals, and ESD guidelines).
- Index
  The index helps you to locate information in the Manual quickly and easily.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

Disposal and recycling of the device

TM15 / TM17 High Feature is an environmentally friendly product. It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with DIN 54840.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. To recycle and dispose of your old device in an environmentally friendly way, please contact a recycling company certified for electronic waste.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner/index.asp

Further information / FAQs

You can find further information on this manual under the following FAQs:

http://support.automation.siemens.com/WW/view/de/27585482

You can also find additional information under:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling).
- The latest SIMOTION FAQs at http://support.automation.siemens.com/WW/view/en/10805436/133000
- SIMOTION SCOUT online help
- Refer to the list of references (separate document) for additional documentation

12.3.1 Safety Instructions

12.3.1.1 Fundamental safety instructions

General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

⚠ WARNING

Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public line supplies requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are enabled.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the controller and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example:
   - Hardware defects and/or software errors in the sensors, controllers, actuators, and connection technology
   - Response times of the controller and drive
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of radio devices / cellular phones in the immediate vicinity of the controller
   - External influences / damage
2. In the event of a fault, exceptionally high temperatures, including an open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the inverter, for example:
   - Component malfunctions
   - Software errors
   - Operating and/or ambient conditions outside of the specification
   - External influences / damage

   Inverters of the Open Type / IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that the contact with fire inside and outside the inverter is not possible.
3. Hazardous touch voltages caused by, for example:
   - Component malfunctions
   - Influence of electrostatic charging
   - Induction of voltages in moving motors
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - External influences / damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc. if they are too close.
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly.

Note

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant sections in the technical user documentation.

12.3.1.2 Specific safety instructions SIMOTION TM15 / TM17 High Feature

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

⚠ WARNING

Danger to life from unexpected movement of machines on (automatic) restart

Dangerous mechanical movements may occur in the system during operation.

Make sure this presents no hazard to personnel or property.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra-low voltage in accordance with EN/IEC 60950-1 may be connected to all connections and terminals.

12.3.2 Description

12.3.2.1 TM15 and TM17 High Feature modules - Introduction

Introduction

The TM15 and TM17 High Feature terminal modules can be used to set up inputs of measuring inputs and outputs of output cams for the SIMOTION motion control system. In addition, these terminal modules provide drive-related digital inputs and outputs with short signal delay times.

In particular, wherever standard I/O only permit **one signal change per position control cycle clock**, TM15 and TM17 High Feature can:

- Process up to two edges per position control cycle clock for inputs of measuring inputs or outputs of output cams.
- Process input and output edges even **within** the position control cycle clock.

The use of an internal timer means that the resolution for outputs of output cams and inputs of measuring inputs is in the microsecond range rather than the millisecond range (switching edges are acquired/output under timer control rather than in the IPO or position control cycle clock).

System integration

As a result of the standardized modular design of the SIMOTION system, the terminal modules can be operated very easily by means of the TO outputCam, TO camTrack and TO measuringInput technology objects.

In principle, there are two possibilities for integration of the TM15 and TM17 High Feature modules into a SIMOTION automation solution:

- System configuration with integrated drives
  In this configuration, the TM1x modules are connected directly
  - To the SIMOTION D4x5/D4x5-2
  - To the SIMOTION D410 (not shown in the figure)
  - To the CX32/CX32-2 extension (not shown in the figure)

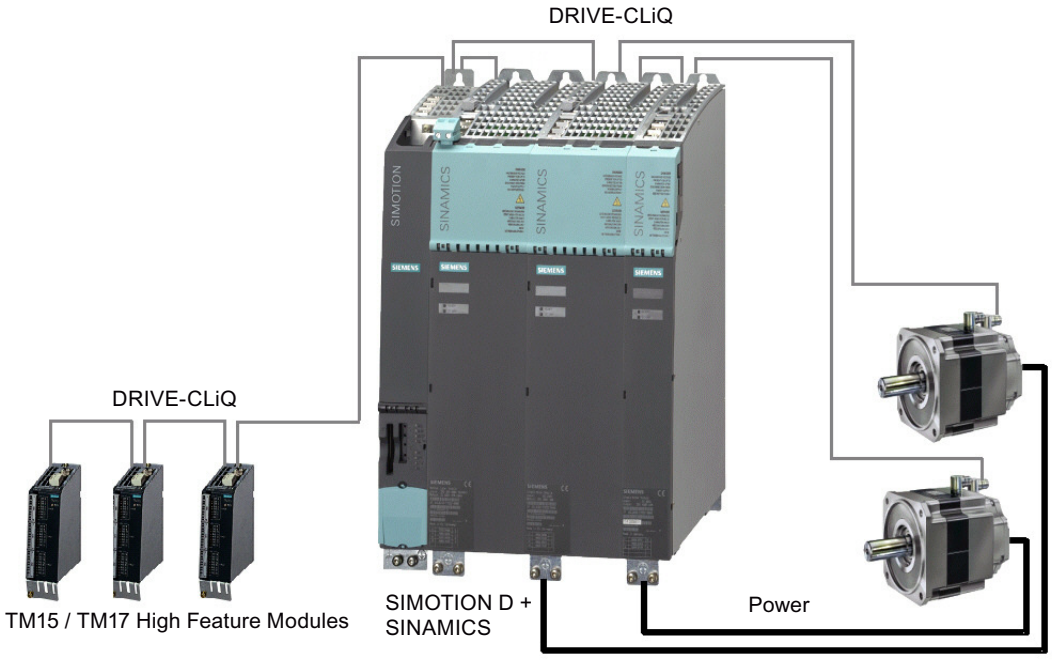


Figure 12-32 Integration of TM15 / TM17 High Feature with SIMOTION D4x5/D4x5-2

- System configuration with external drives
  In this configuration, the TM1x modules are connected to a SINAMICS S120 control unit CU320/CU320-2 or CU310, which is connected
  - To the SIMOTION C, P or D (see figure) via PROFIBUS DP, or

– To the SIMOTION C, P or D via PROFINET IO

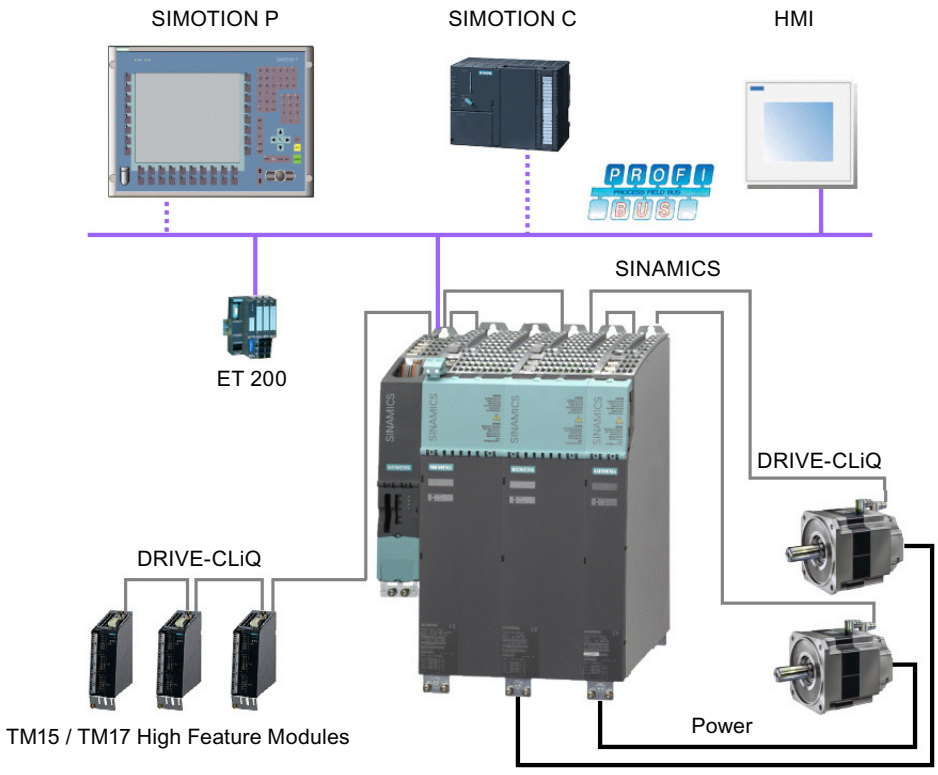


Figure 12-33 Integration of TM15 / TM17 High Feature with SIMOTION C or P

A characteristic of both options is that the DIs, DOs, cam outputs and measuring input inputs can only be accessed from the SIMOTION controller.

Note that the TM15 module can also be used with DI/DO functionality alone (selection: TM15 DI/DO input/output component). In this case, no output cam or measuring input functions are available, but DI and DO terminals can be used by both SIMOTION and SINAMICS (similar to TB30 or TM31).

Note

This Commissioning Manual describes how to integrate the TM15 and TM17 High Feature into SIMOTION. Integration of the TM15 via BICO technology is not covered in this Commissioning Manual. You will find more information on integrating the TM15 via BICO technology in the SINAMICS S120 Commissioning Manual as well as in the SIMOTION D Commissioning and Hardware Installation Manuals.

Project interconnection

As of SIMOTION SCOUT V4.2, in contrast to interconnection via the logical address, there is now an option to symbolically assign drive objects (DOs) to technology objects (TOs).

- Symbolic assignment (fixed assignment)
  The symbolic assignment is set by default and applies to the entire project.
  In the case of the symbolic assignment, the system automatically aligns logical addresses when establishing the DO/TO connection.
  Should any changes be made in the HW Config, you must not manually align the address to ensure that the project is consistent.
- Assignment via the logical address
  Only possible if you deactivate the symbolic assignment in the project or work with a SIMOTION SCOUT version < V4.2.
  As is normally the case, the DO/TOs must be assigned using the logical address. Changes to the logical address space, for example, reconfiguration of HW Config, call for manual re-alignment in order to ensure the consistency of the project.

Structure

The modules are mounted on a DIN rail in accordance with DIN EN 50 022 (35 x 15 / 7.5). They are connected to the SIMOTION/SINAMICS hardware via DRIVE CLiQ.

Several TM1x modules can be installed on one DRIVE CLiQ line.

The SINAMICS S120 Manuals provide more information on the system structure of SINAMICS S and associated hardware.

Note

In the "Supplemental SINAMICS System Components for SIMOTION" Manual, you will find information about the design of Terminal Modules TM15 and TM17 High Feature as well as a description of the interfaces and installation.

See also

Maximum number of terminal modules (Page 8241)

12.3.2.2 Properties: TM15 / TM17 High Feature

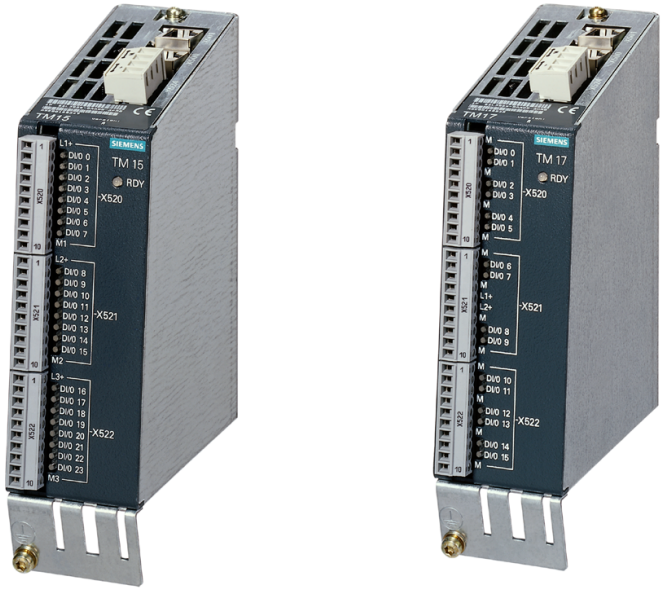


Figure 12-34 TM15 terminal module and TM17 High Feature terminal module

The properties of the terminal modules are presented below. This overview is intended to help you select the appropriate module for your application.

Properties

The TM15 and TM17 High Feature terminal modules are terminal extension modules designed to be mounted on a DIN rail in accordance with DIN EN 50 022.

The terminal modules contain the following:

- Two DRIVE-CLiQ sockets.
- Connection for 24 VDC electronic power supply.
- The logical status of the I/O channel is indicated by the associated green status LED.
- The status of the TM15 / TM17 High Feature is indicated by a multi-colored RDY LED.

Each of the 24 DI/DO (TM15) or 16 DI/DO (TM17 High Feature) can be parameterized on a channel-specific basis as a digital input (DI), digital output (DO), measuring input input or cam output, and can also be inverted. The SCOUT engineering software is used for parameterization.

- Each channel can be parameterized as a digital input (DI) or digital output (DO).
- Each channel can be parameterized as a measuring input input.
- Each measuring input input has a selectable edge detection (falling edge, rising edge, or both edges).
- Each channel can be parameterized as a cam output.

The outputs are equipped with short-circuit protection, shutdown in case of overtemperature, and reverse-polarity protection. The actual signal state of each output channel can be read back on the SIMOTION side.

The differences between the TM15 and TM17 High Feature terminal modules are defined by their range of application. The TM17 High Feature has less I/O channels than TM15, but offers increased functionality.

The TM17 High Feature is distinguished by especially high resolution and accuracy as well as a parameterizable input filter and enable inputs.

The enable inputs can activate measuring input inputs or cam outputs (gate function).

- Level-controlled enable for measuring input inputs
- Level- or edge-controlled enable for cam outputs

In addition, TM17 High Feature supports cyclic measurement of up to two edges per position control cycle clock.

Due to their high accuracy, the DI/DO channels of the TM17 High Feature are non-isolated.

Note

The module hardware of the TM15 and TM15 DI/DO is identical, but there is a difference between the two modules in terms of system integration. For this reason, they must be configured as different input/output components.

This Commissioning Manual describes how to integrate the TM15 and TM17 High Feature into SIMOTION. Integration of the TM15 DI/DO via BICO technology is not covered in this Commissioning Manual. You will find more information on integrating the TM15 DI/DO via BICO technology in the SINAMICS S120 Commissioning Manual as well as in the SIMOTION D Commissioning and Hardware Installation Manuals.

Table 12-1 Comparison table for module selection

Function	**TM15** [1]	**TM15 DI/DO** [1]	**TM17 High Feature** [1]
System integration	Use only in conjunction with SIMOTION (SIMOTION D, CX32/CX32-2 or CU310/ CU320/CU320-2 with SIMOTION as the higher-level controller)	Use also in conjunction with CU310/CU320/CU320-2 without SIMOTION	Use only in conjunction with SIMOTION (SIMOTION D, CX32/CX32-2 or CU310/ CU320/CU320-2 with SIMOTION as the higher-level controller)
Functionality	24 DI/DO can be parameterized on a channel-specific basis as DI, DO, cam output or measuring input input; DI/DO can only be addressed via SIMOTION controller	24 DI/DO-channels can be parameterized as DI or DO; DI/DO can only be interconnected via BICO technology	16 DI/DO can be parameterized on a channel-specific basis as DI, DO, cam output or measuring input input; DI/DO can only be addressed via SIMOTION controller
Short response time	X	X	X
Number of I/O channels	24	24	16
Electrical isolation of inputs and outputs	X	X	-
Grouping of channels	3 groups, each of 8 channels	3 groups, each of 8 channels	2 groups, each of 8 channels

Function	TM15 [1]	TM15 DI/DO [1]	TM17 High Feature [1]
Measuring input inputs (single measurement)	X	-	X
Measuring input inputs (cyclic measurement)	-	-	X
Several measuring inputs per axis	X	-	X
One measuring input for several axes	X	-	X
Measuring on virtual axes	X	-	X
Cam outputs	X	-	X
Resolution - measuring input inputs	Typ. 125 µs [2] (firmware support)	Not applicable	1 µs (hardware support)
Resolution - cam outputs	Typ. 125 µs [2] (firmware support)	Not applicable	1 µs (hardware support)
Accuracy (reproducibility) - measuring input inputs	Typ. ±125 µs [2]	Not applicable	≤ ±1 µs
Accuracy (reproducibility) - cam outputs	Typ. ±125 µs [2]	Not applicable	≤ ±10 µs
Input filtering	50 µs	50 µs	1 µs or 125 µs, user-assignable
Enable via external signal	-	Not applicable	6 channels, max.

[1] "X" = available; "-" = not available.

[2] Corresponds to the DRIVE-CLiQ cycle clock in use (125 µs, typical)

12.3.2.3 Machine applications

Overview

Many production machines require fast, precise detection of signals or exact switching of digital outputs. The measuring input and output cam feature of the TM15 and TM17 High Feature modules represents an optimal solution for a variety of industrial applications.

Applications requiring fast, high-precision detection of signals include:

- Edge detection
- Quality monitoring (e.g. product OK/not OK)
- Product tracking (e.g. product available/not available)
- Print-mark detection
- Tool monitoring (e.g. presses)
- Machine status monitoring (e.g. plastic injection molding machines)
- Weft break monitoring (e.g. textile machines)

Applications requiring fast, high-precision output of signals include:

- Position-dependent switching of actuators
  - Camera trigger signal (quality assurance)
  - Control of an air nozzle for blowing away cut-offs
  - Control of nozzles for applying glue tracks
- Product extraction from production line
- Implementation of line motion control systems
- Output of pulse patterns

Application examples

Edge processing

In this example, the movement of boards through a machine is controlled by fixed guides and clamping brackets. The sides of the boards are machined as they pass by a group of actuated cutters.

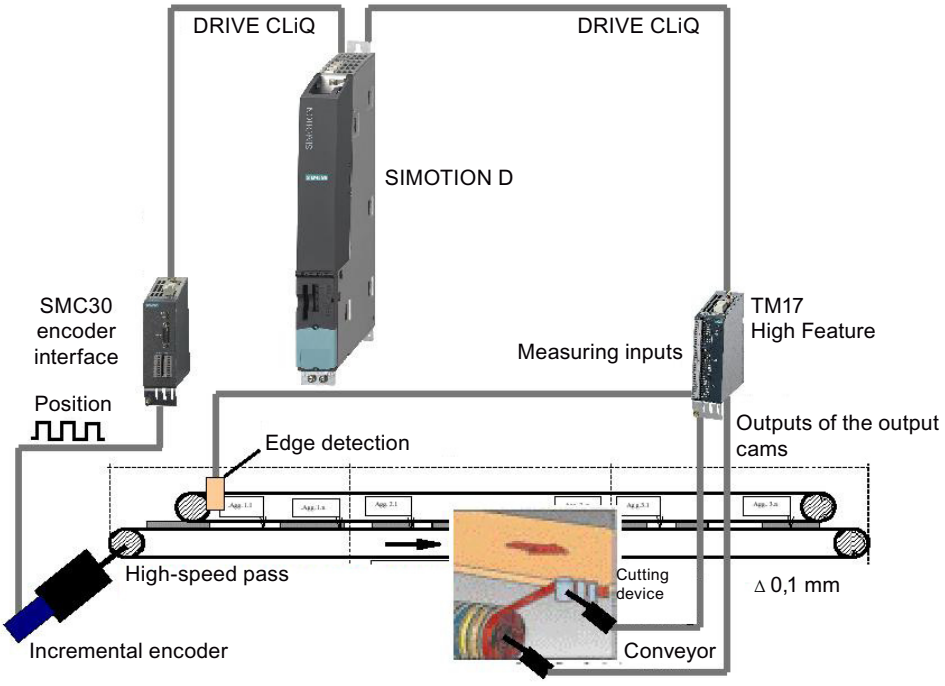


Figure 12-35 Example: Edge detection

The technology of the terminal modules includes:

- Exact encoder value measurements
- Exact edge acquisition
- Exact output of switching signals

As a result, applications with increased line speed are possible.

The switching conditions specific to each type of board are stored in a central controller. Depending on the exact position of the board (relative to its detected rising edge), the individual cutters are engaged and disengaged. The switching signal control relative to the detected edge is critical since cutting accuracy is directly dependent upon precise switching.

Application of glue tracks

In the following example, glue tracks are applied to a workpiece. Each track controls one glue gun via a digital output.

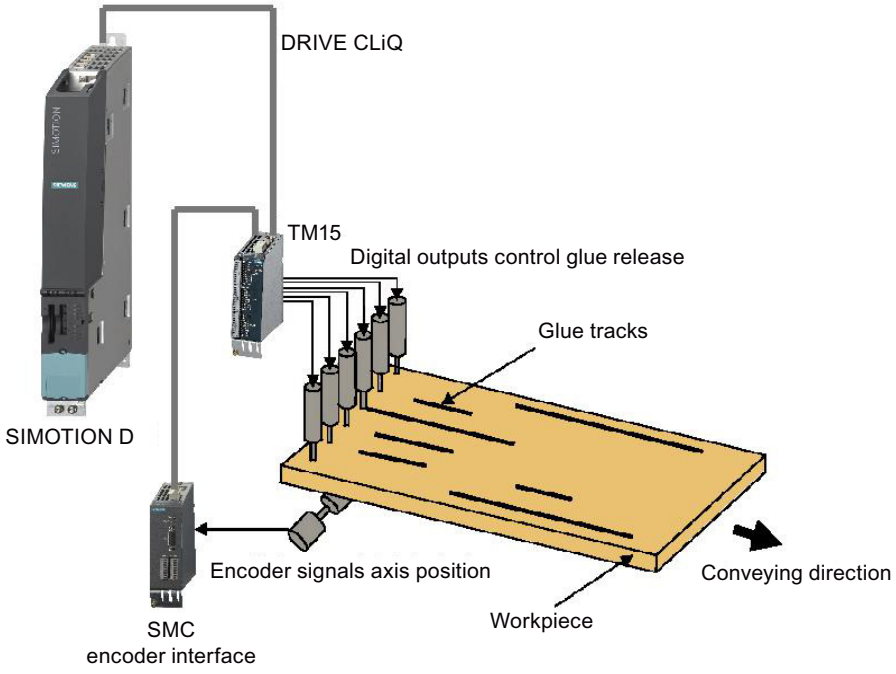


Figure 12-36 Example: Electronic output cam control

The glue guns are activated via outputs of an output cam based on the axis position.

The axis position is measured by an encoder that connects an SMC encoder interface to SIMOTION D via DRIVE-CLiQ.

The SIMOTION Output Cam or Output Cam Track technology object controls the digital outputs of the TM15 based on the axis position.

The output delay times (output delay times of DO / actuators, etc.) are compensated in the technology object. This compensation ensures that the glue placement is always accurate regardless of machine speed.

12.3.3 Configuration/programming

12.3.3.1 Hardware and software requirements

To begin using the terminal modules, you need the following:

Hardware requirements

- TM15 or TM17 High Feature
- One of the following components:
  - SIMOTION D (TM is connected directly to SIMOTION D or CX32/CX32-2)
  - SIMOTION C, P or D (TM is connected to a control unit, which is connected via PROFIBUS/ PROFINET to SIMOTION C, P or D)
  - DIN rail in accordance with DIN EN 50 022 (35 x 15 / 7.5)
  - DRIVE-CLiQ cable
  - PROFIBUS or PROFINET cable
  - External 24 VDC power supply (note the requirements for the TM15 and TM17 High Feature terminal modules)

Software requirements

This documentation describes

- SIMOTION V4.2 and
- SINAMICS V2.x/V4.x

Previous SIMOTION and SINAMICS versions do not include all the functions described in this manual.

See also

Version overview (Page 8275)

12.3.3.2 Requirement for the configuration and programming

Requirement

Once the terminal module hardware is mechanically and electrically installed, it must be integrated in the application project using the SCOUT engineering software.

The procedures contained in this chapter assume that the user has a general understanding of SCOUT.

In addition, a SCOUT project must be generated, for example, with SIMOTION D.

Note

The following description relates to the configuration of TM1x terminal modules, which are connected to the integrated drive of SIMOTION D.

The dialogs encountered during the configuration of terminal modules TM1x may differ slightly if the TM1x are connected to a SINAMICS control unit CU310 or CU320/CU320-2, which is connected to SIMOTION via PROFIBUS or PROFINET.

Project interconnection

Symbolic assignment

As of SIMOTION SCOUT V4.2, in addition to interconnection via the logical address, there is now an option to symbolically assign DO/TOs.

The symbolic assignment is preset for new projects. You can check this via the Project menu.

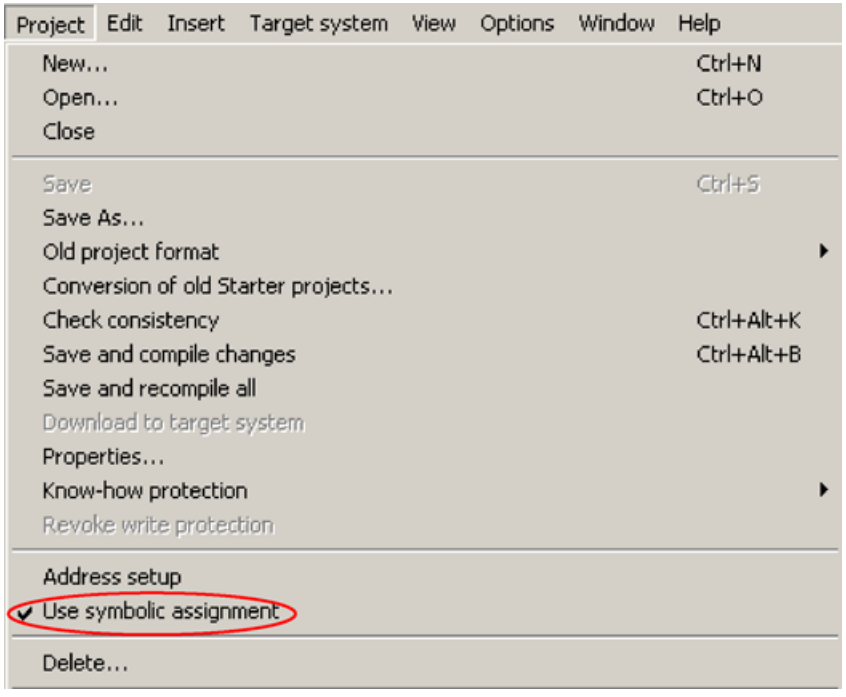


Figure 12-37 Project menu

After creating a terminal module, the automatic addressing in the message frame configuration is activated. The message frame configuration is opened via **SINAMICS_Integrated** -> **Communication** -> **Message frame configuration** in the project navigator.

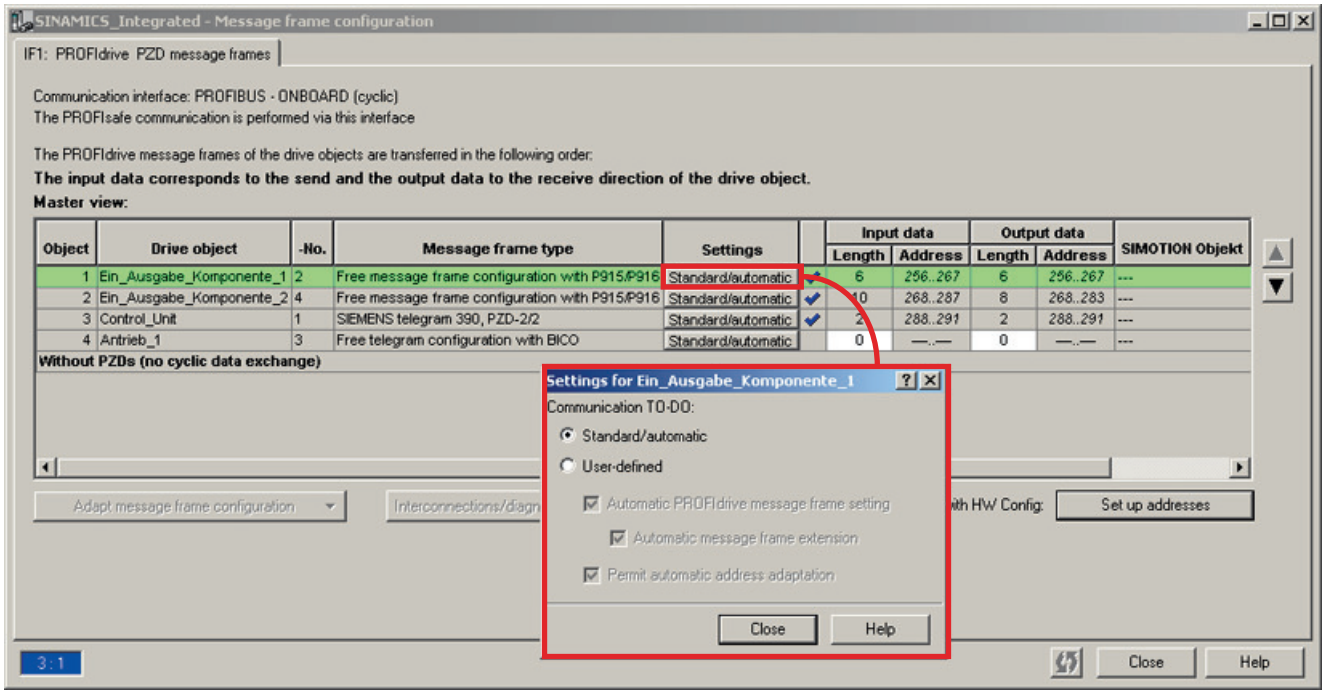


Figure 12-38 Message frame configuration

Note

We recommend that you do not change between symbolic assignment and assignment via the logical address. A change can result in data being lost.

Note

If symbolic assignment is subsequently activated for a project in which message frames have already been configured and interconnected, these can be changed together with the BICO interconnections.

For this reason, make a backup copy of your project before activating the symbolic assignment.

TB30, TM15 DI/DO and TM31 are especially affected.

Note

Further information on symbolic assignment can be found in the Basic Functions Manual, Section Symbolic assignment (as of V4.2).

Assignment via the logical address

If you want to continue using the interconnection of DO/TOs via the logical address, you must deselect the default setting in the Project menu.

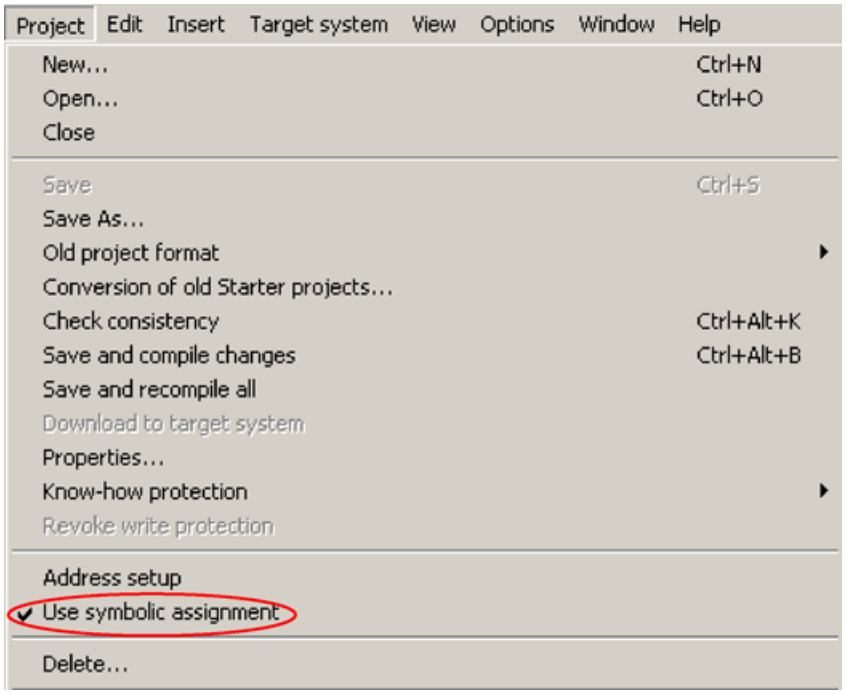


Figure 12-39 Project menu

The message frame configuration is opened via **SINAMICS_Integrated** -> **Communication** -> **Message frame configuration** in the project navigator. The following window with the I/O address assignments of the message frame is displayed.

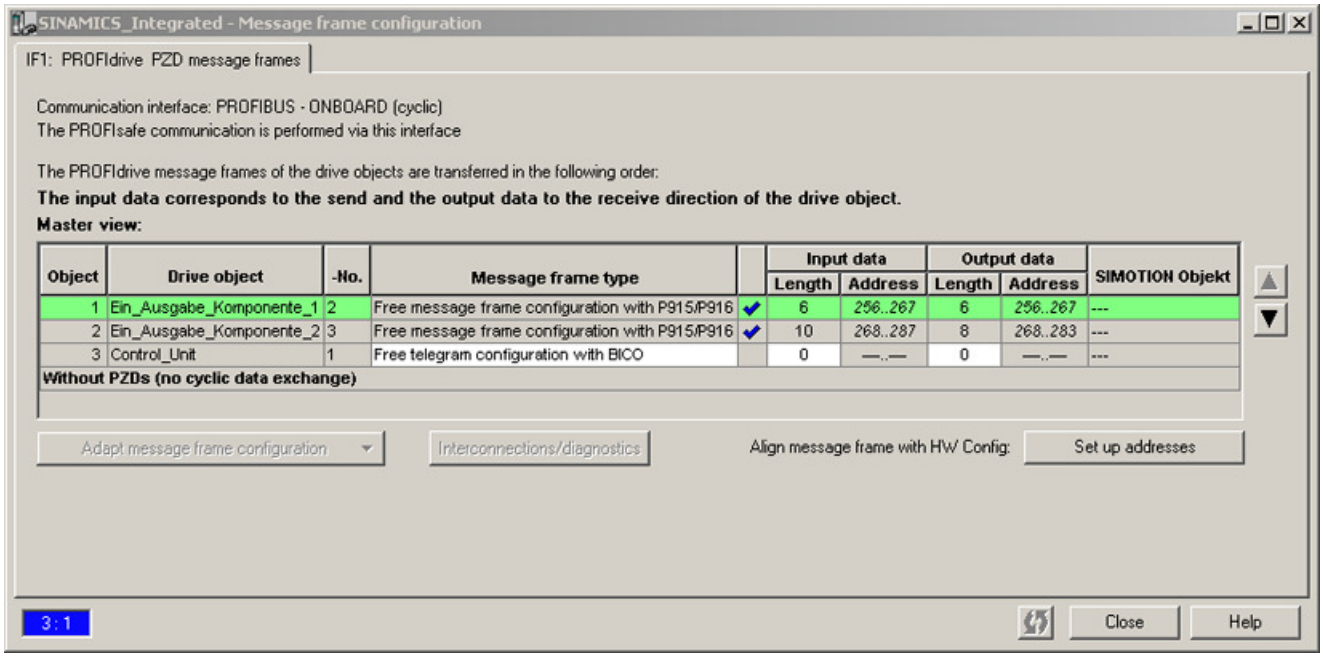


Figure 12-40 Message frame configuration

Note

We recommend that you do not change between symbolic assignment and assignment via the logical address. A change can result in data being lost.

Note

The terminal module must be powered up before it can be accessed via the I/O. Otherwise, an I/O access error will occur and the CPU will go to STOP mode.

12.3.3.3 Inserting new TM1x modules

Procedure

1. Click the **+** next to **Input/output component**. An expanded project tree similar to the one below will be displayed.

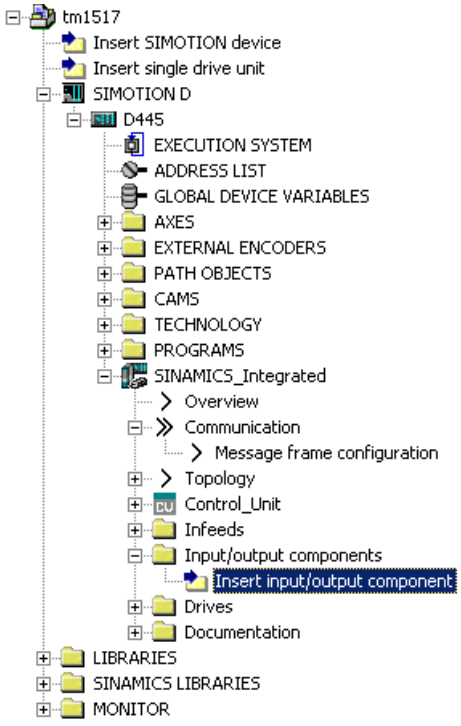


Figure 12-41 "Insert input/output component" window

2. Double-click **Insert input/output component**. A window similar to the one shown below will be displayed.

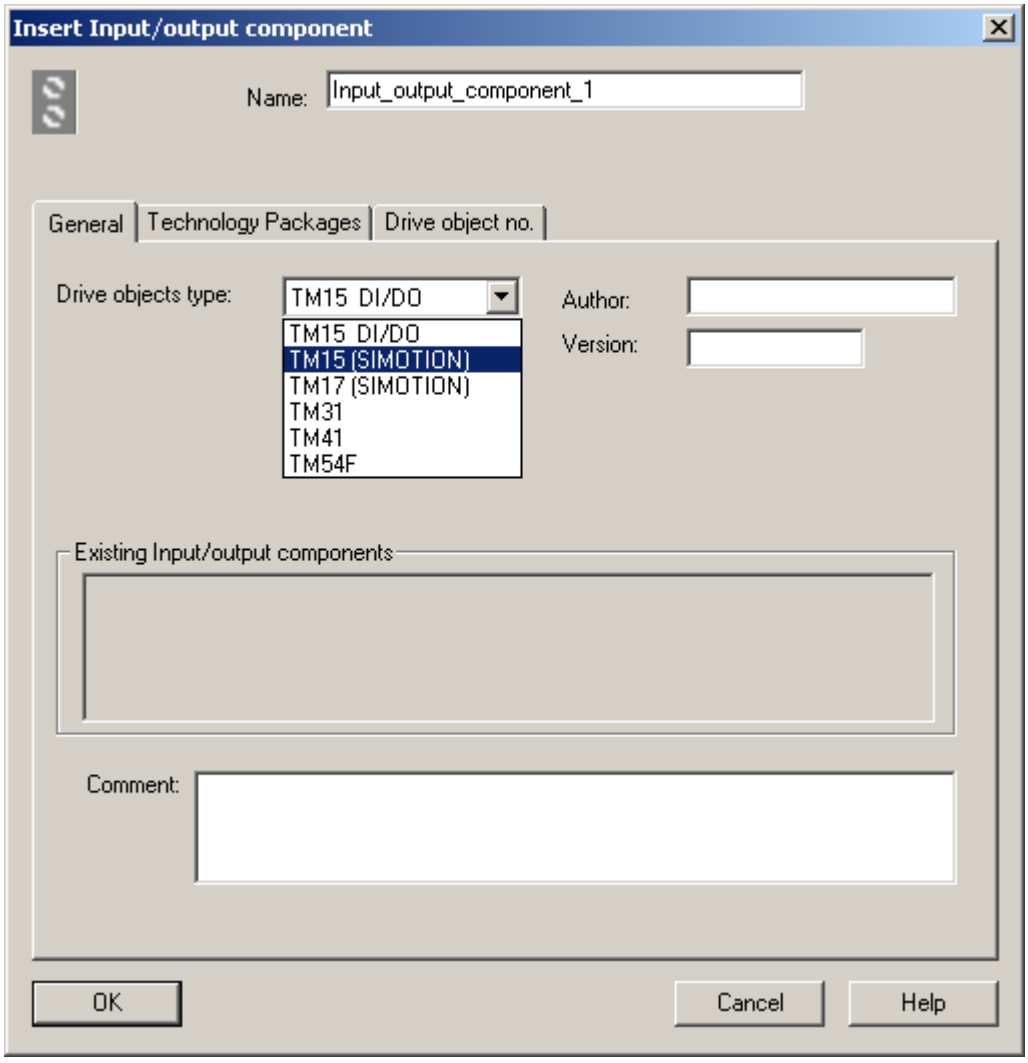


Figure 12-42 "Insert Input/Output Component" window

The **Name** field may be used to give the module a unique name (e.g. TM15_1). A drop-down list box containing the various module types will be displayed under Operating type.

- **TM15 (SIMOTION)** for digital inputs/outputs as well as measuring input inputs and cam outputs (can only be used by SIMOTION)
- **TM15 DI/DO** for digital inputs/outputs (no output cam / measuring input function, but DI/DO can be used by both SIMOTION and SINAMICS)
- **TM17(SIMOTION)** for digital inputs/outputs as well as measuring input inputs and cam outputs for the most stringent accuracy requirements (can only be used by SIMOTION)
- **TM31**, **TB30** can be used, in principle, the same as the TM15 DI/DO
  "TM15 (SIMOTION)" has been selected in the figure. The **Author**, **Version** and **Comments** fields can be used to further describe the module.

3. Click **OK** to insert the new module.

12.3.3.4 Terminal configuration

Configuring I/O channels – TM15

Procedure

1. "TM15_1" will now appear in the project tree. Open the directory tree underneath it and double-click **Configuration**. The window shown below will open.

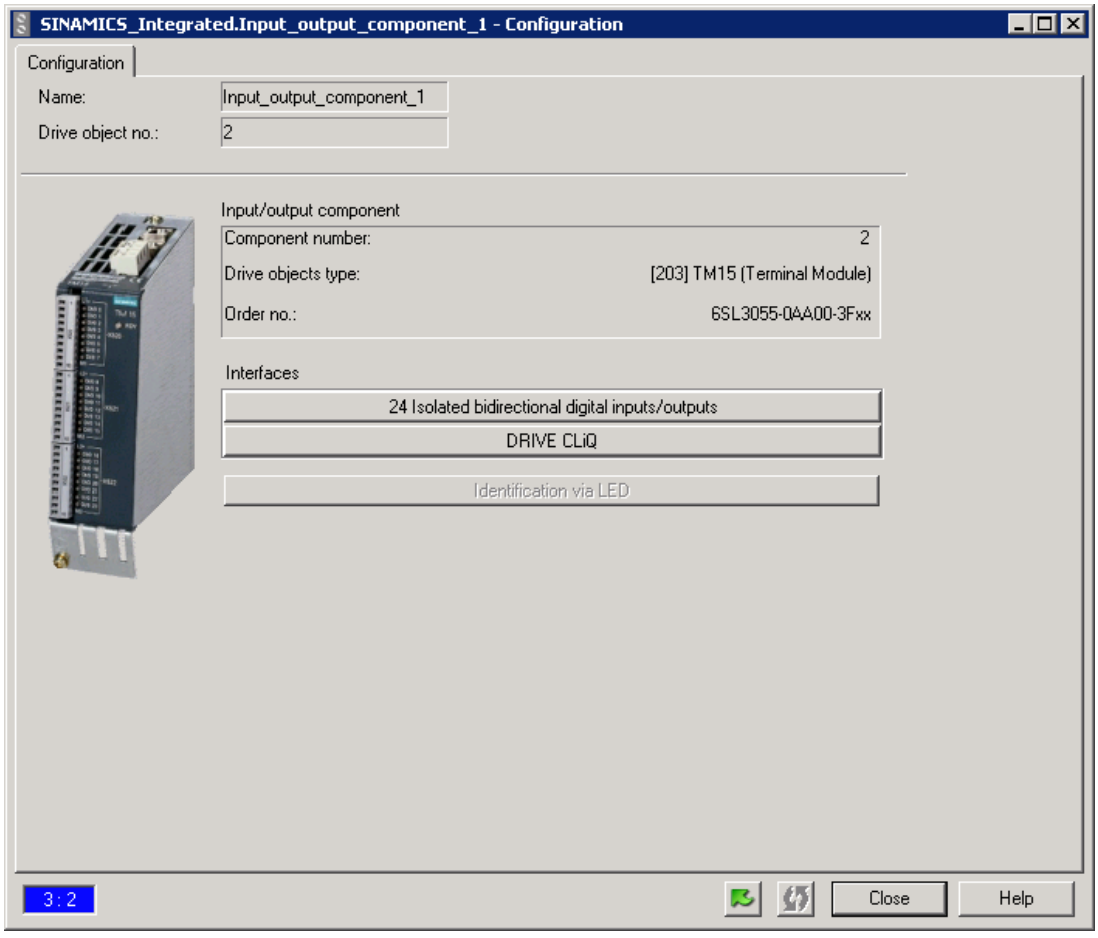


Figure 12-43 SCOUT – "TM15_1 - Configuration" window

Table 12-2 Function of the buttons

Button	Function
24 Isolated bidirectional digital inputs/outputs	Configuration of I/O channels
	Display of component overview

Button	Function
DRIVE-CLiQ	Display of the DRIVE-CLiQ topology
24 Isolated bidirectional digital inputs/outputs	When you click this button (available online only), the selected module can be identified by its RDY LED, which will flash (red-green).

2. Once you are satisfied that the displayed information is correct, you can configure the individual TM15 (SIMOTION) I/O channels. Click the **24 isolated bidirectional digital inputs/ outputs** button. (You can also double-click **Inputs/outputs** in the project tree instead.) The window shown below will open.

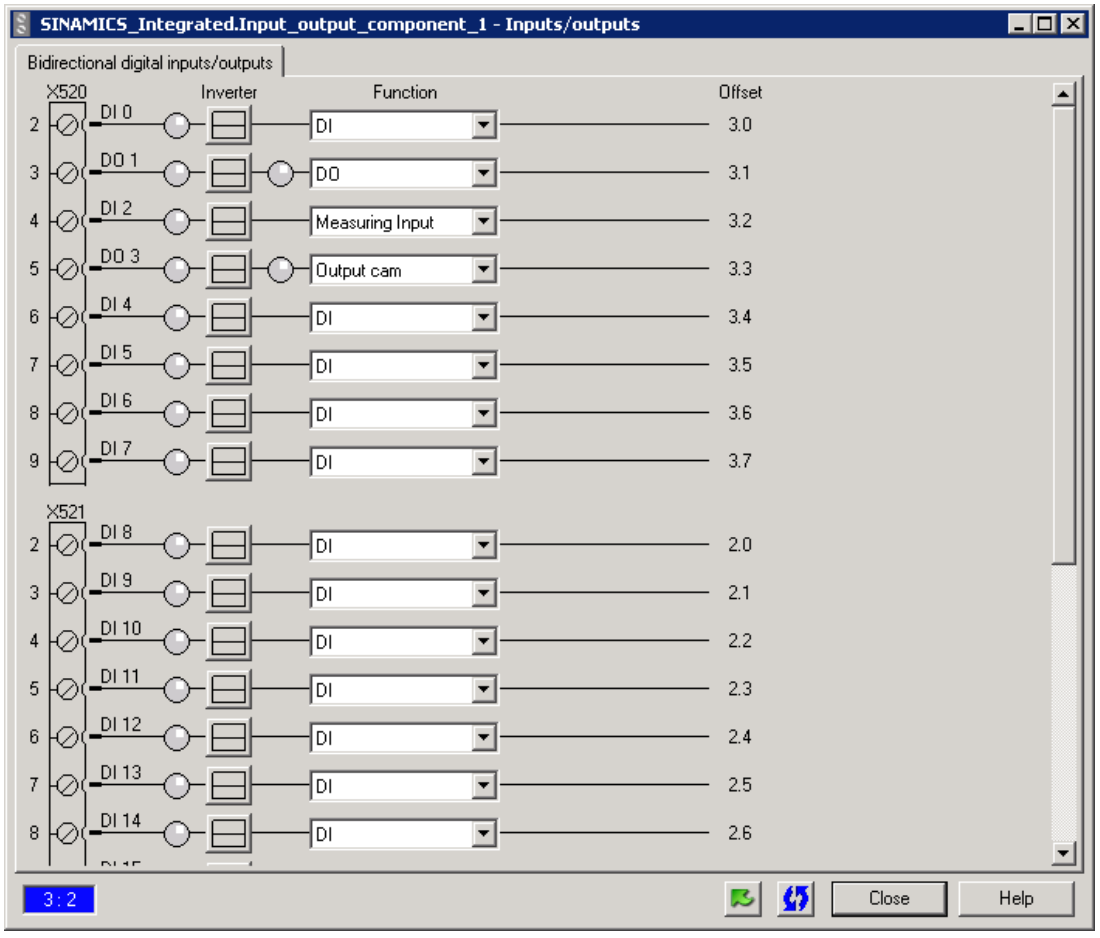


Figure 12-44 SCOUT – graphical configuration screen form for the TM15 (SIMOTION)

You can use the drop-down menus in the **Function** column to select the type of I/O channel to be used (i.e. DI, DO, input of measuring input or output of output cam). One of four possible functions can be assigned to each channel.

Table 12-3 "Function" column: Options in the drop-down menu

Menu option	Description
DI	Channel used as a digital input
DO (standard output[1])	Channel used as: - Digital output, or - Cam output (**without** high switching accuracy)
Measuring input	Channel used as a measuring input input
Output cam (fast output[2])	Channel used as a fast cam output (**with** high switching accuracy)

[1] The cam output is calculated by the SIMOTION technology object in the IPO cycle clock or position control cycle clock. (That is, the resolution is one IPO cycle clock or one position control cycle clock.)

[2] The switching instant of the cam output is calculated by the TM15. For this reason, the resolution is less than the position control cycle clock.

Depending on which function has been selected, you can switch between the input or output using the symbol displayed under "Inverter".

Table 12-4 "Inverter" column: Description of symbols

Symbol	Description
	Signal not inverted
	Inverted input
	Inverted output

Configuring I/O channels – TM17 High Feature

To begin configuring the TM17 High Feature terminal module, you can follow the procedure described in the section on "Configuring I/O Channels – TM15". The window shown below will open.

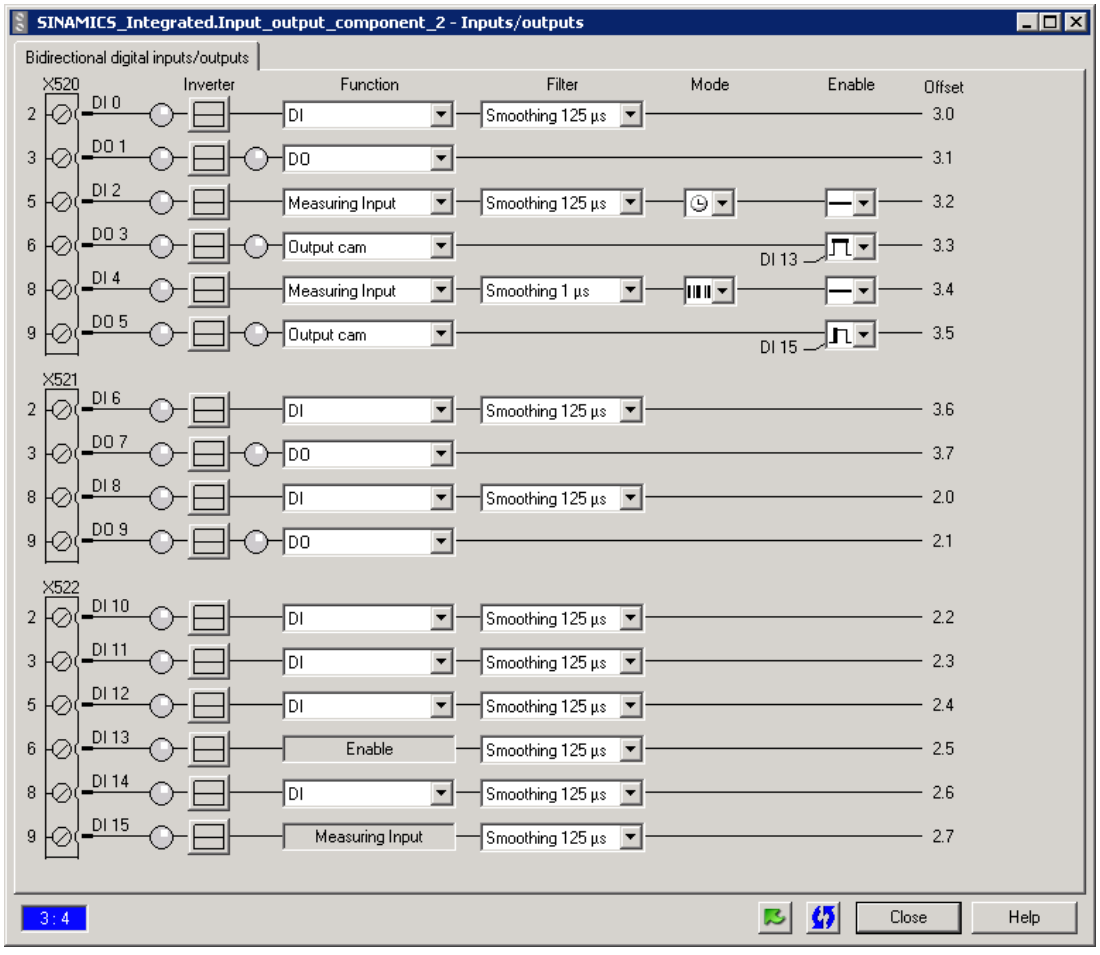


Figure 12-45 SCOUT – graphical configuration screen form for the TM17 (SIMOTION)

The **Inverter**, **Function** and **Offset** columns are identical to those in the TM15 display.

Additional configuration settings for the TM17 High Feature can also be made (see figure below).

Filter

You can select a 1 µs or 125 µs filter for the inputs via the drop-down menus in the **Filter** column. The shortest pulses can be detected using the 1 µs filter; however, the 125 µs filter will provide higher noise immunity.

Measurement modes

You can select the measurement mode from the drop-down menu in the **Mode** column.

Table 12-5 "Mode" column: Description of symbols

Symbol	Measurement mode
	Single measurement
	Cyclic measurement

The main difference between the **Single measurement** and **Cyclic measurement** modes is that with **Single measurement**, a measurement job must be issued by the TO measuringInput for each measurement.

The measurement job remains active until the measurement result has been obtained or until the job is terminated by a command. The measuring process must be reactivated for each new measurement. :

In contrast, the measuring function need only be activated once for **Cyclic measurement** mode. The measuring function remains active until it is deactivated again.

Up to two edges can be measured in each execution cycle of the TO measuringInput (IPO interpolation cycle clock, IPO2 interpolation cycle clock or position control cycle clock).

The measured values must be read out by the user program before new measured values are accepted.

Result:

- A maximum of two edges can be evaluated per IPO cycle clock if the scan routine of the user program is in the IPO synchronous task.
- A maximum of two edges can be evaluated per position control cycle clock if the scan routine of the user program is in the servo-synchronous task.

The servo-synchronous task is the lowest possible time level for a user program.

Table 12-6 Measuring functions in conjunction with TM15 and TM17 High Feature

	Single measurement	Cyclic measurement
Supported terminal modules	TM15 and TM17 High Feature	TM17 High Feature only
Measurement operation	Measurement job must be issued for each measurement	Measuring function need only be activated once. The measuring function remains active until it is deactivated again.
Time between two measurements	Several IPO cycle clocks or several position control cycle clocks[1)]	1 IPO cycle clock or 1 position control cycle clock [1]
Measuring range for measuring input can be defined (as TO function)	possible	possible

[1)] If scan routine is in the servo-synchronous task

See also

System timing for single measurement (Page 8270)

System behavior with cyclic measurement (Page 8271)

Enable function

With the **Enable** drop-down menu, the measuring input inputs or cam outputs of channels 0 to 5 can also be enabled by an enabling signal of channels 10 to 15.

If you select the enable function for channels 0 to 5, the following symbols in the **Function** column will be automatically selected for the assigned enable channels 10 to 15:

- Enable (for level-triggered enable)
- Measuring Input (for edge-triggered enable)

Note

If the enable input is to be used with an "inverted" cam output, the inversion must be parameterized for the TM17 High Feature terminal module and **not** for the TO outputCam or TO camTrack.

An inversion of the output on the "TO outputCam" or "TO camTrack" always causes a low level to be output when the enable is missing (with the desired inversion, however, this would correspond to a driven output cam).

By contrast, when inversion is parameterized for the TM17 High Feature terminal module, it only takes effect once the enabling signal has been sent, thus bringing about the desired result.

Table 12-7 **Enable** and **Function** columns: Description of symbols

Symbol for enabling signal selection (Enable column)	**Symbol for assigned input of enabling signal (Function column)**	**Explanation**
	No enable input	No enabling signal and therefore no enable input selected.
		Level-triggered enabling signal. The channel assigned for the enabling signal is displayed as an enable.
		Edge-triggered enabling signal (setting option only available for cam outputs). The channel assigned for the enabling signal is displayed as a measuring input. A TO measuringInput must be configured in SIMOTION for the measuring input input. The TO measuringInput can be used to evaluate the positions of the enable edges.

The enabling signal polarity depends upon the selection you have made in the "Inverter" column, as follows:

Table 12-8 **Inverter** column: Inverter options for enabling signals

Enabling signal selection (Enable column)	**Inverter options (Inverter column)**	
Not inverted 	Inverted 	
Level-triggered 	Enabling signal is high active	Enabling signal is low active
Edge-triggered 	Enabling signal takes effect on a rising edge	Enabling signal takes effect on a falling edge

Other information

Note

When parameters are reassigned for the I/O channels, the new parameters do not take effect until you restart (power on or hot restart) the SINAMICS or SIMOTION D device.

Note

Additional information and parameter setting options are available in the project tree next to **Configuration** and **Inputs/outputs** (e.g. **Control logic**, **Diagnostics**, etc.). However, this information is most relevant for modules that are configured via BICO technology, e.g. for TM31. These registers are irrelevant to the user in TM15 and TM17 High Feature.

Note

When you click "TM15_1" or "TM17_1" in the project tree, a number of parameters are displayed at the bottom of the symbol browser window. These parameters are irrelevant to the user.

Note

TM15 and TM17 High Feature with vector drives or servo drives

If DRIVE-CLiQ cycle clocks ≠ 125 µs are used with vector drives, inaccuracies will occur when detecting signals via the TO measuringInput or when outputting signals via the TO outputCam / TO camTrack.

The same phenomenon may arise with servo drives if you set sampling times via p112 that result in DRIVE-CLiQ cycle clocks ≠ 125 µs.

In the case of vector drives, the DRIVE-CLiQ cycle clock is calculated by the control unit during ramp-up.
The SCOUT Engineering System must be aware of this DRIVE-CLiQ cycle clock (p4099) that is set automatically by the control unit.

The same phenomenon may arise with servo drives if sampling times are set via p112 that result in DRIVE-CLiQ cycle clocks ≠ 125 µs.

In such cases, you should proceed as follows:

1. Carry out a project download.
2. Establish an online connection.
3. Upload to the SINAMICS drive. The updated p4099 parameter is read in as a result of the upload.
4. Go offline in SCOUT.
5. Only relevant if you have deactivated **Use symbolic assignment**:
   Create the configuration information (FastIO configuration) again. Select the SIMOTION CPU in the project tree and right-click to open the context menu. Recreate the configuration via the menu **FastIO** > **Create new configuration**.

12.3.3.5 Applicative access with symbolic assignment

Introduction

As of SIMOTION SCOUT V4.2, in addition to interconnection via the logical address, there is now an option to symbolically assign DO/TOs.

The procedure for symbolic assignment is described in the following sections. As of SIMOTION SCOUT V4.2, the symbolic assignment is set as default setting (for new projects). You can check this via the Project menu.

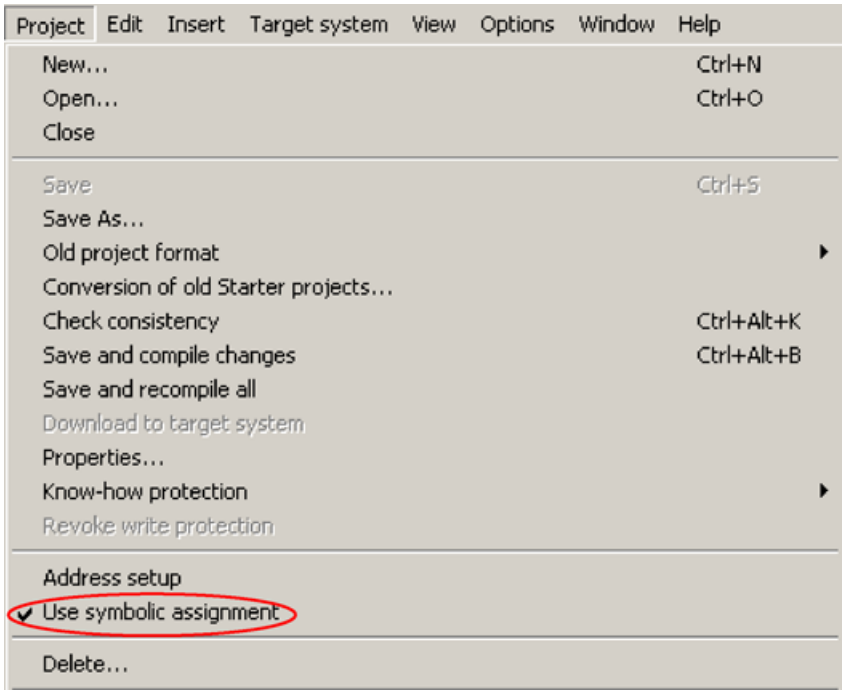


Figure 12-46 Project menu

After creating a terminal module, the automatic addressing in the message frame configuration is activated. The message frame configuration is opened via **SINAMICS_Integrated** -> **Communication** -> **Message frame configuration** in the project navigator.

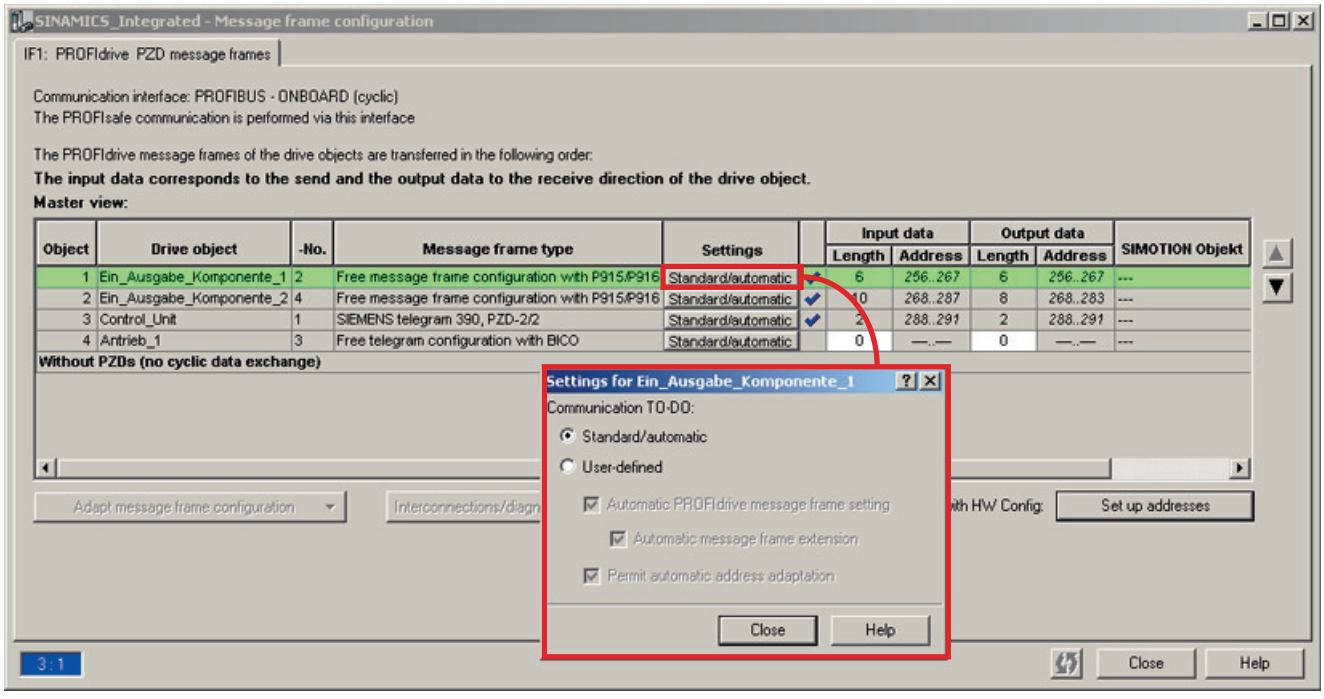


Figure 12-47 Message frame configuration

Note

We recommend that you do not change between symbolic assignment and assignment via the logical address. A change can result in data being lost.

Note

If symbolic assignment is subsequently activated for a project in which message frames have already been configured and interconnected, these can be changed together with the BICO interconnections.

For this reason, make a backup copy of your project before activating the symbolic assignment.

TB30, TM15 DI/DO and TM31 are especially affected.

Note

Further information on symbolic assignment can be found in the Basic Functions Manual, Section Symbolic assignment (as of V4.2).

Note

The terminal module must be powered up before it can be accessed via the I/O. Otherwise, an I/O access error will occur and the CPU will go to STOP mode.

See also

Power Up and Synchronization with the User Program (Page 8227)

Linking symbolic I/O variables with TM1x terminal modules

You can assign addresses to binary I/O channels using symbolic variables in the user program.

Procedure

In the following example, create the variable 'var_IO_DI0' and link this to input DI_0 and variable 'var_IO_DO1' with output DO_1.

The following configuration is the basis for the example.

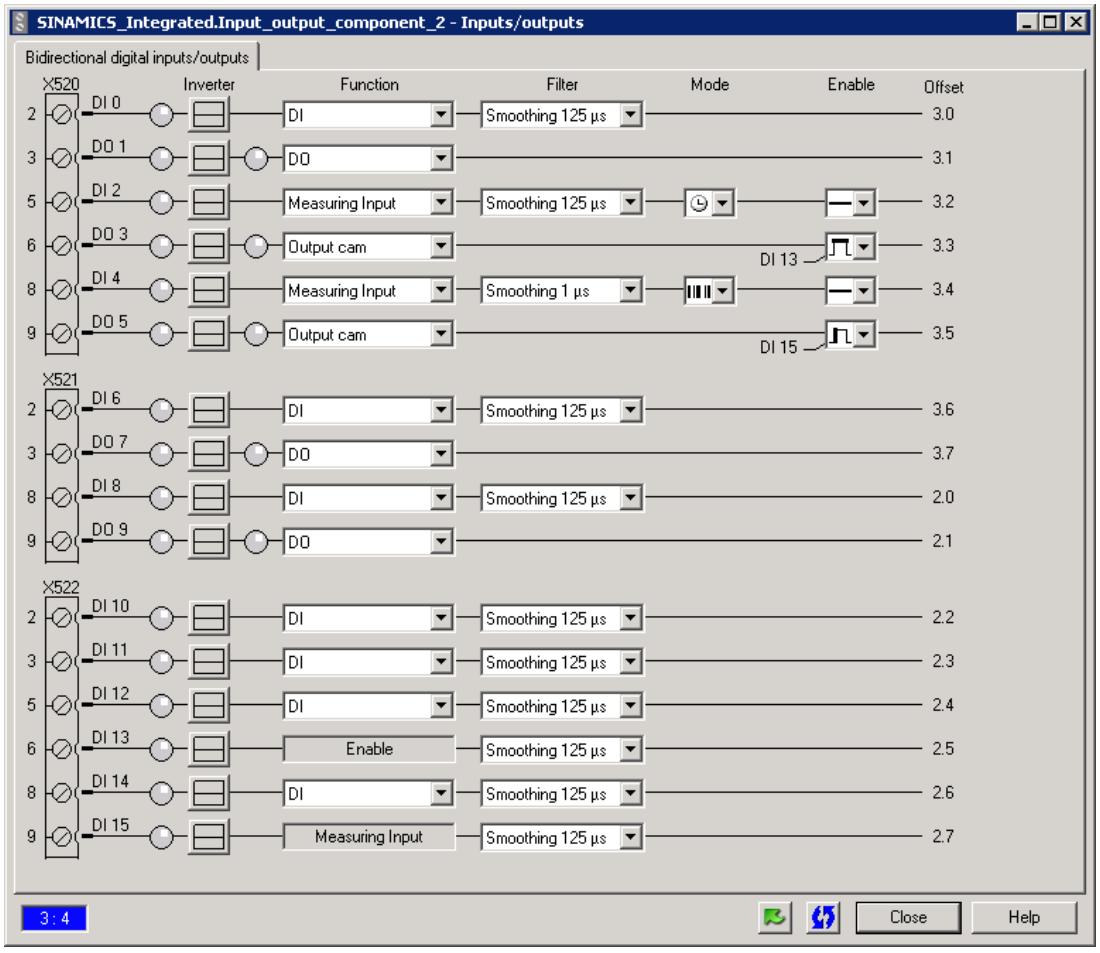


Figure 12-48 SCOUT – graphical configuration screen form for the TM17

Input DI 0

1. Open the **address list** via the project navigator.
2. In the **View** field, select 'I/Os'.
3. Create the variable 'var_IO_DI0' and in the **I/O address** field, select 'IN'.

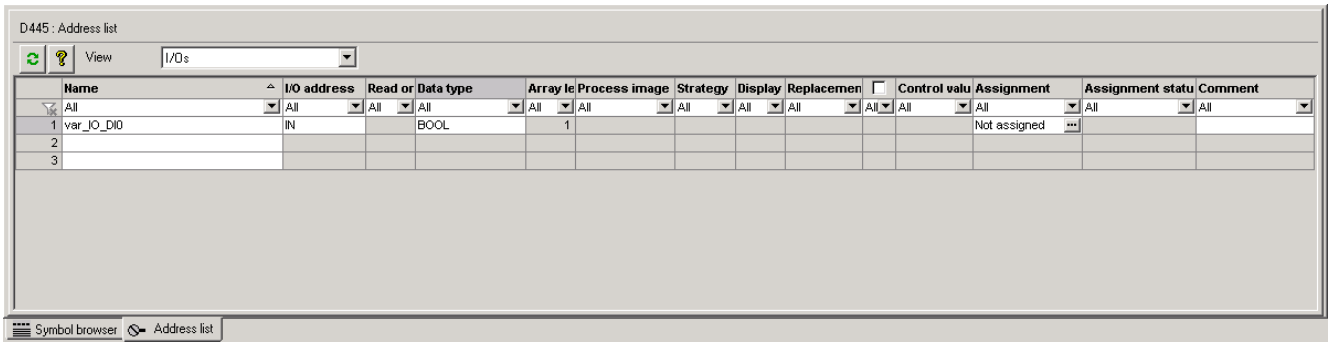


Figure 12-49 Creating variables in the address list

4. Click ... in the **Assignment** field.

5. Select the input DI_0 in the following dialog box (explanations on the identifiers can be found below).

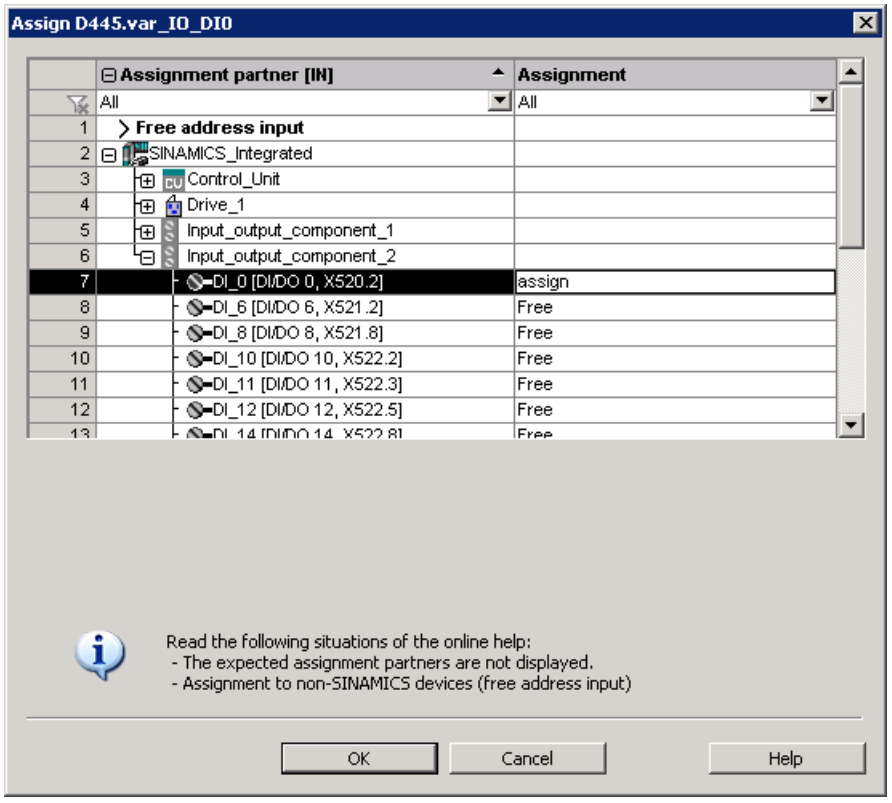


Figure 12-50 Selecting input DI_0

6. Confirm the selection by clicking **OK**.

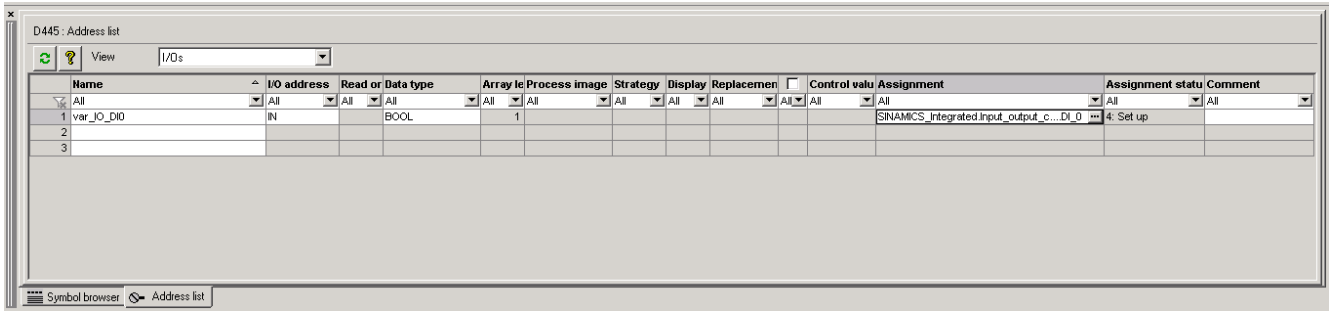


Figure 12-51 Address list following successful assignment

Output DO 1

1. Create the variable 'var_IO_DO1' and in the **I/O address** field, select 'OUT'.

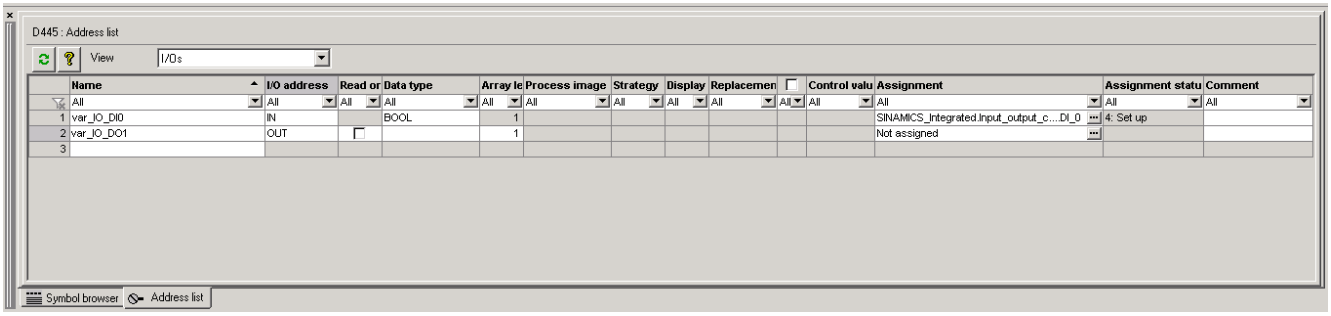


Figure 12-52 Creating variables in the address list

2. Click ... in the **Assignment** field.

3. Select the DO_1 input in the next dialog box.

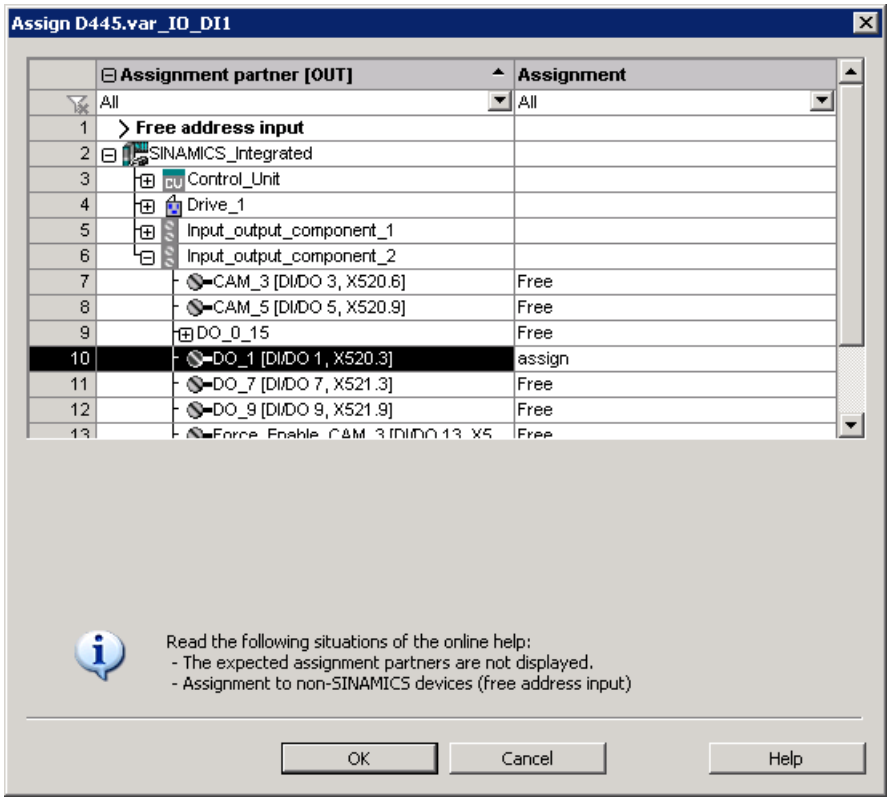


Figure 12-53 Selecting output DO 1

4. Confirm the selection by clicking **OK**.

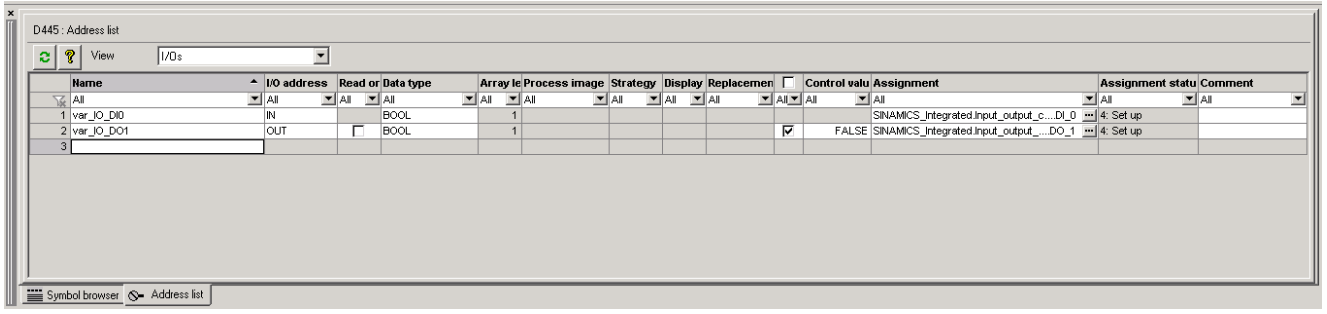


Figure 12-54 Address list following successful assignment

Identifiers

Assignment partner identifier	Description
DO_n	Output
DI_n	Input
MI_n	Measuring input
CAM_n	Output cam
Enable_MI_n	Enable input for measuring input input
Enable_CAM_n	Enable input for cam output

Assignment partner identifier	Description
Force_Enable_MI_n	Forcing enable input for measuring input input
Force_Enable_CAM_n	Forcing enable input for cam output
State_CAM_n	Reading back of the terminal state
State_DO_n	Reading back of the terminal state

Notes

Note

The procedures for generating symbolic variables and for using the process image or direct I/O access are described in detail in the SIMOTION SCOUT Manual.

Note

Please also note that the terminal module must have ramped up before it is accessed via I/Os. You can also configure an error strategy for I/O (CPU stop, substitute value or last value). Note that byte-specific setting is the minimum possible setting.

You cannot assign a substitute value for an output byte if individual bits have been allocated to an TO outputCam or TO camTrack.

See also

Power Up and Synchronization with the User Program (Page 8227)

Controlling the enabling signal

If the enabling signal has been selected for one of the channels on the TM17 High Feature, the user program can control this hardware function.

The control bit of the enabling signal has the following function (= force function):

- "0" → The I/O channel (e.g. channel 0) is controlled by the channel for the enabling signal (e.g. channel 10).
- "1" → The I/O channel (e.g. channel 0) is enabled irrespective of the channel for the enabling signal (e.g. channel 10).

If the I/O channel is activated by a HW enabling signal during forcing, this enable is retained even after the forcing function is terminated (provided that the enable conditions are still fulfilled).

Example

In this example, DO 13 on the TM17 High Feature terminal module is used as the channel for the enabling signal (see the following figure).

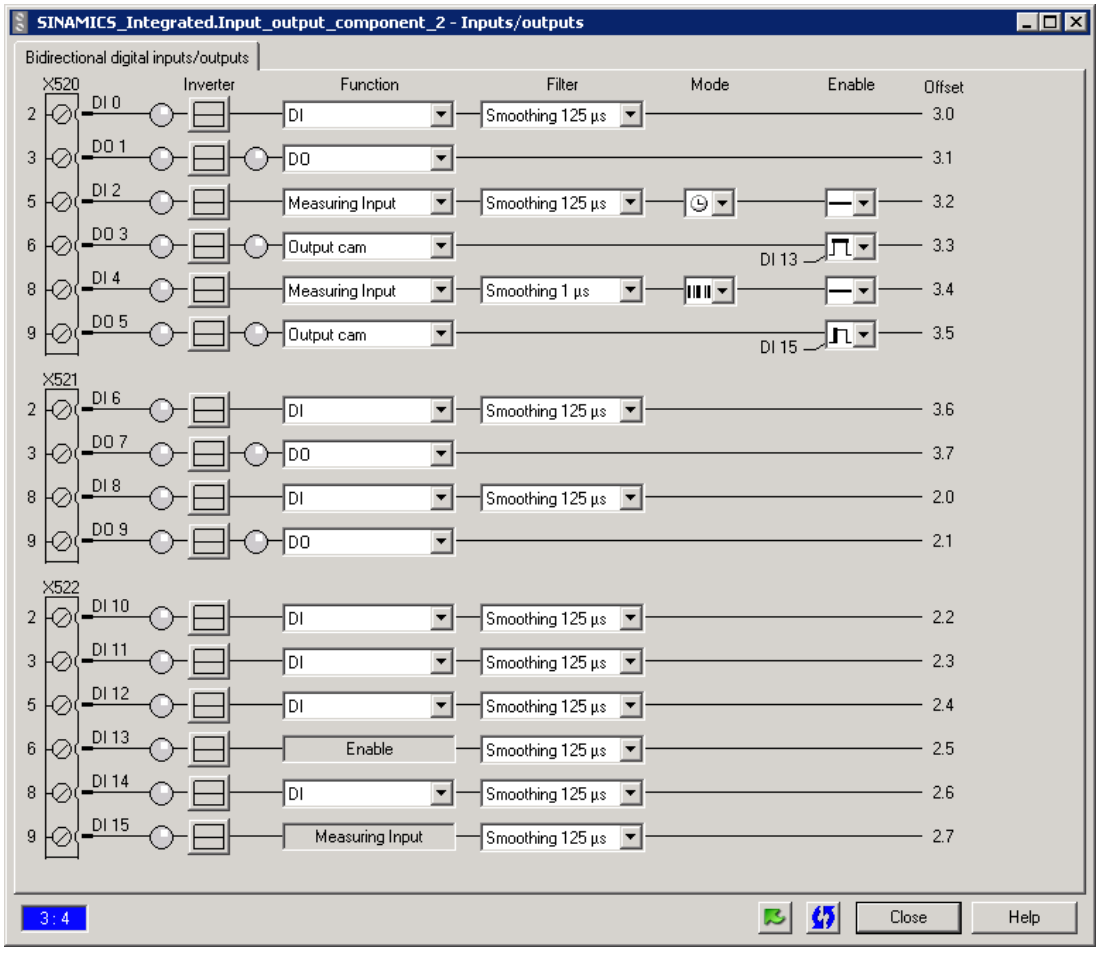


Figure 12-55 SCOUT – graphical configuration screen form for the TM17

Procedure

1. Create a variable in the **address list**, select 'OUT' in the **I/O address** field and 'BOOL' in the **Data type** field.

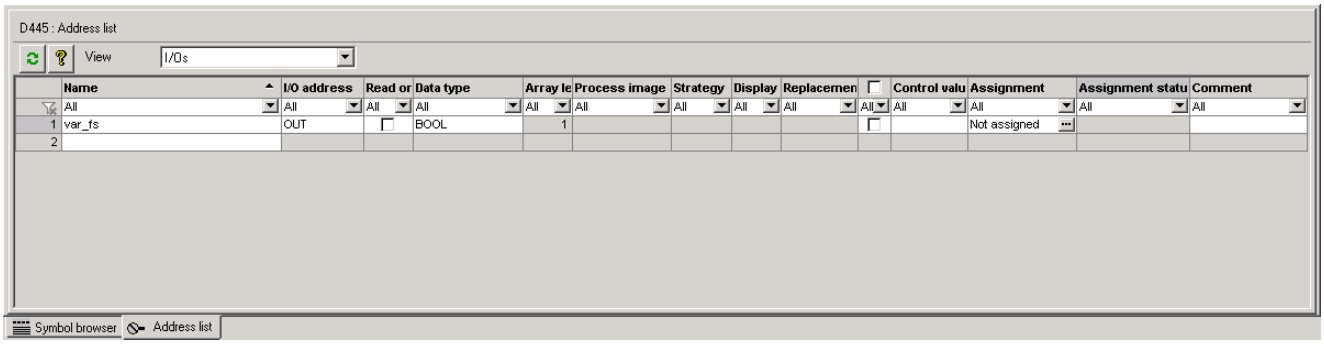


Figure 12-56 Creating a variable

2. Click ... in the **Assignment** field.

3. Select the **Force_Enable_CAM_3** assignment partner in the following dialog box.

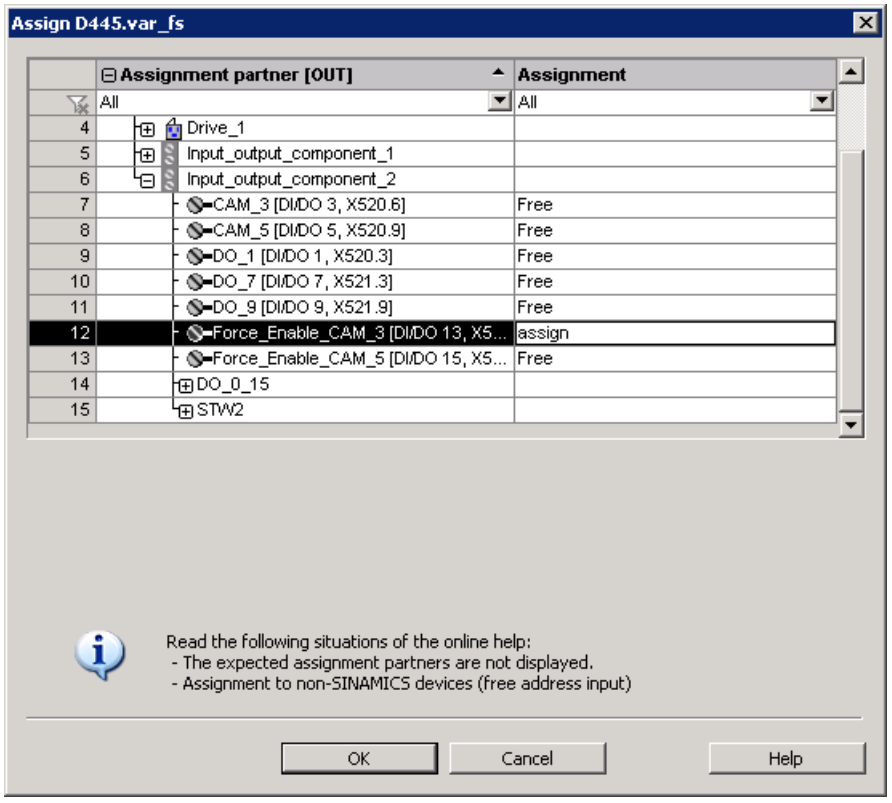


Figure 12-57 Selecting the Force_Enable_CAM_3 output

4. Confirm the selection by clicking **OK**.

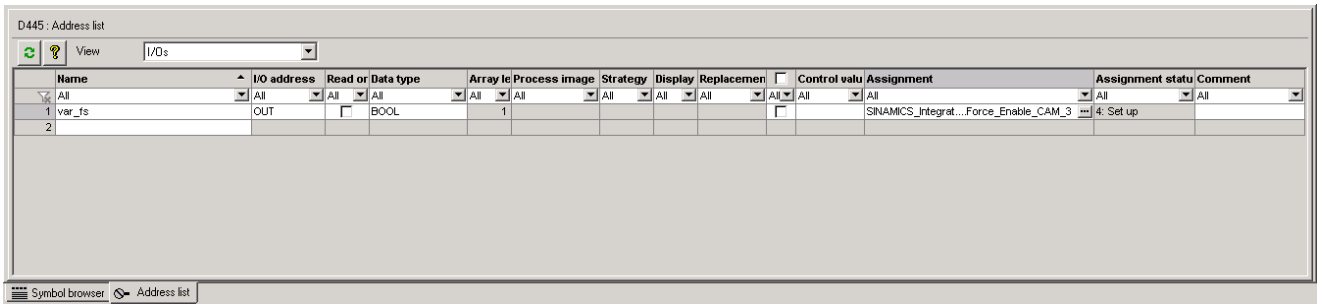


Figure 12-58 Address list following confirmation

Signal status of the enable signal

If required, the user program can read the signal state of the enabling signal via the **Enable_MI_n** or **Enable_CAM_n** assignment partner. The binary value reflects the signal state at time T_{in}.

See also

System behavior with binary inputs and outputs (Page 8268)

Read back function for outputs

The user program can read back the signal state of each output channel on the TM15 or TM17 High Feature via the **State_CAM_n** or **State_DO_n** assignment partner. The binary value reflects the signal state at time T_{in}.

Note

If an inverted output channel is parameterized, the inverted terminal status is read during the read-back.

See also

System behavior with binary inputs and outputs (Page 8268)

I/O assignment to technology objects

Linking technology objects with terminal modules

Measuring input inputs and cam outputs are operated with the associated technology objects (TOs). A general understanding of the use of technology objects is therefore needed to perform the desired configurations.

Note

The terminal module must be powered up before it can be accessed by technology objects. Otherwise, a technology alarm will be triggered.

The assignment of the measuring input inputs is performed as for the inputs (DI) and the assignment of the cam outputs as for the outputs (DO), but with the **CAM_n** or **MI_n** assignment partner.

You will find a detailed description of linking technology objects with terminal modules in the "SIMOTION Motion Control - Technology Objects for Output Cams and Measuring Inputs" Manual.

See also

Power Up and Synchronization with the User Program (Page 8227)

Power Up and Synchronization with the User Program

An isochronous bus is required for operation of the TM1x terminal modules.

The terminal modules must have powered up and reached the synchronized state before they can be read- or write-accessed. The synchronization of the terminal modules requires at least 18 servo cycles.

Until synchronization, the digital outputs are disabled (low level at the terminal).

The TM1x terminal modules are not ready to operate until they are successfully synchronized with the SIMOTION CPU. During this startup phase, the input and output variables must not be accessed directly or else an I/O access error will occur and the CPU will go to STOP mode.

In addition, all access attempts of the TO measuringInput, TO outputCam or TO camTrack in a non-synchronized state will trigger a technological alarm.

The start-up operation can be monitored by directly accessing the TM1x module status word or via the PeripheralFaultTask.

Monitoring synchronization by direct access to the module status word

The synchronization of the TM1x terminal modules is displayed cyclically to the SIMOTION motion control system via the module status word.

The synchronization bit **SYNC** is located in bit 8 of the module status word:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	ERR	**SYNC**	-	-	-	-	-	FPGA	PS	MF

"-" means reserved, cannot be used

Synchronization bit (bit 8):

SYNC = 0 → Module is not synchronized
SYNC = 1 → Module is synchronized

Example

You can evaluate the synchronization bit as shown in the following example. Create a variable of the BOOL type and assign it bit 8 from the module status word.

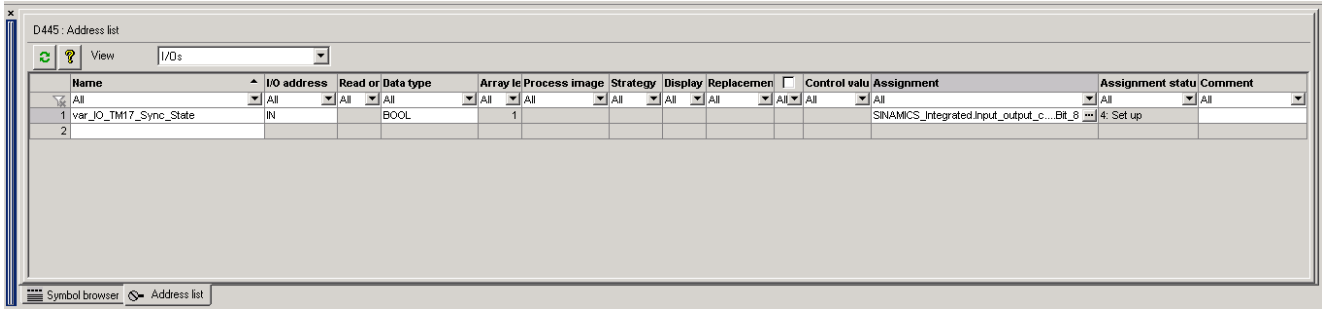


Figure 12-59 Creating variables in the address list

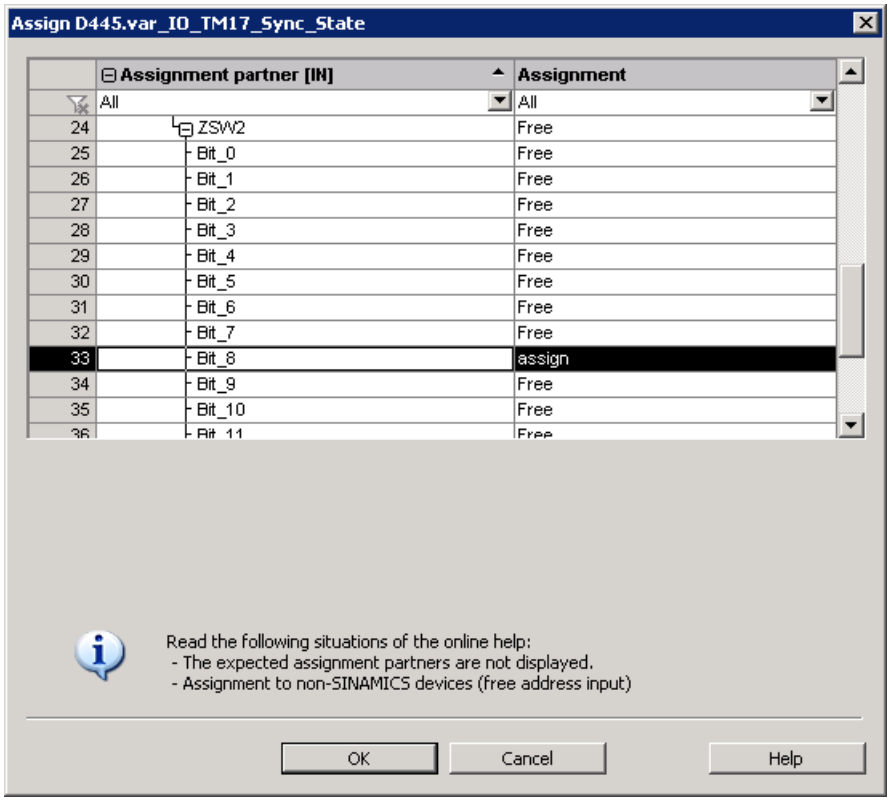


Figure 12-60 Assignment

As soon as SYNC bit = 1, the module is synchronized and may be accessed directly via the I/O or technology objects.

Monitoring the synchronization with PeripheralFaultTask

During the transition from STARTUP to RUN, all TM1x Terminal Modules are in the "NOT_SYNCHRONIZED" state.

- As soon as the Terminal Modules have been successfully synchronized, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_SYNCHRONIZED" (=214) is called.
- If synchronization fails, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_NOT_SYNCHRONIZED" (=215) is called.

Example

To synchronize the user task, a user variable **TM_SYNC** is set to **FALSE** in the StartUpTask and set to **TRUE** in the PeripheralFaultTask with interrupt ID = SC_IO_MODULE_SYNCHRONIZED. The status of **TM_SYNC** is queried in the user task before the (first) direct access operation.

The following TaskStartInfo is supplied in the PeripheralFaultTask every time it is called:

```
DINT    TSI#logBaseAdrIn    // valid only when not equal to _SC_INVALID_ADDRESS
DINT    TSI#logBaseAdrOut   // valid only when not equal to _SC_INVALID_ADDRESS
```

DINT	TSI#logDiagAdr	// valid only when not equal to _SC_INVALID_ADDRESS
DWORD	TSI#details	// set to 0
UINT	TSI#eventClass	// set to 0
UINT	TSI#faultId	// set to 0

The TaskStartInfo contains the logical address of the relevant module.

TSI#logDiagAdr, TSI#details, TSI#eventClass, and TSI#faultId are irrelevant for TM15/TM17 High Feature Terminal Modules.

For further information on the TaskStartInfo, refer to the "SIMOTION ST Structured Text" Manual.

12.3.3.6 Applicative access via the logical address

Introduction

The following sections describe the procedure for the assignment of DOs/TOs via the logical address.

As of SIMOTION SCOUT V4.2, the symbolic assignment is set as default setting for new projects. The symbolic assignment is deactivated for old projects. If you want to continue using the interconnection of DO/TOs via the logical address for new projects, you must deselect the default setting in the Project menu.

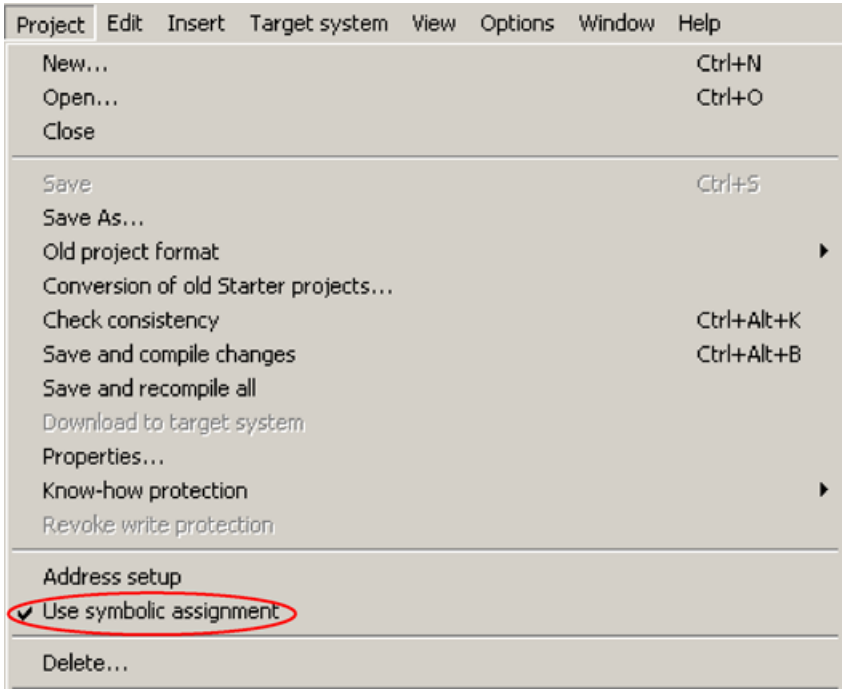


Figure 12-61 Project menu

The message frame configuration is opened via **SINAMICS_Integrated** -> **Communication** -> **Message frame configuration** in the project navigator. The following window with the I/O address assignments of the message frame is displayed.

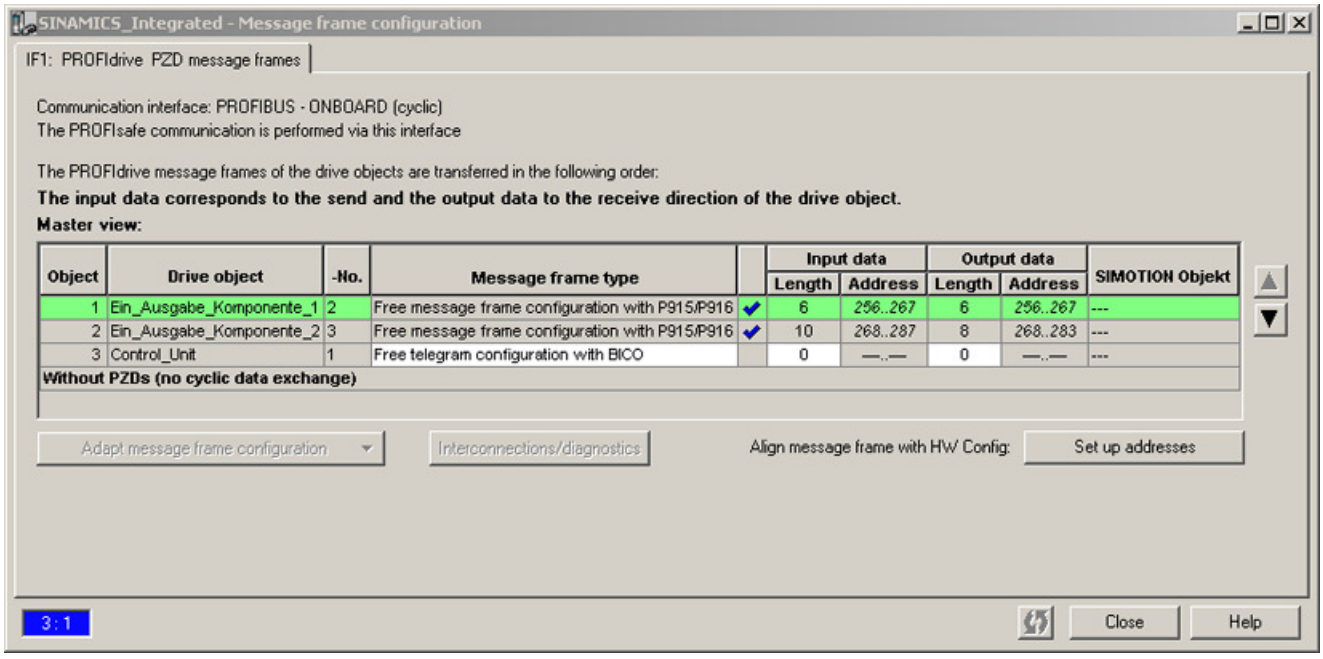


Figure 12-62 Message frame configuration

Note

We recommend that you do not change between symbolic assignment and assignment via the logical address. A change can result in data being lost.

Note

The terminal module must be powered up before it can be accessed via the I/O. Otherwise, an I/O access error will occur and the CPU will go to STOP mode.

See also

Power Up and Synchronization with the User Program (Page 8238)

Generating message frames and TM1x drive objects

The message frames are automatically generated for TM15 and TM17 High Feature (or updated if they previously existed) when you select the "Close" button in the graphical configuration screen form.

When using the TM15 DI/DO, TM31 or TB30 with SIMOTION SCOUT, you must set up the message frames as free message frames and interconnect the terminal functions to the message frame via BICO.

Note

Do not make any changes to the parameters in the Expert list, as these changes will not be automatically updated in the message frame and could therefore lead to errors.

See also

Message frames (Page 8273)

Maximum permissible message frame length (Page 8242)

Aligning the configuration with the hardware

I/O addresses must now be assigned to the message frame so that the runtime system can access it. These addresses are generated/updated by the hardware configuration tool included in SCOUT.

1. To begin assigning addresses in this example, double-click **Message frame configuration** under **SINAMICS_Integrated** -> **Communication** in the project tree. The following window with the I/O address assignments of the message frame appears.

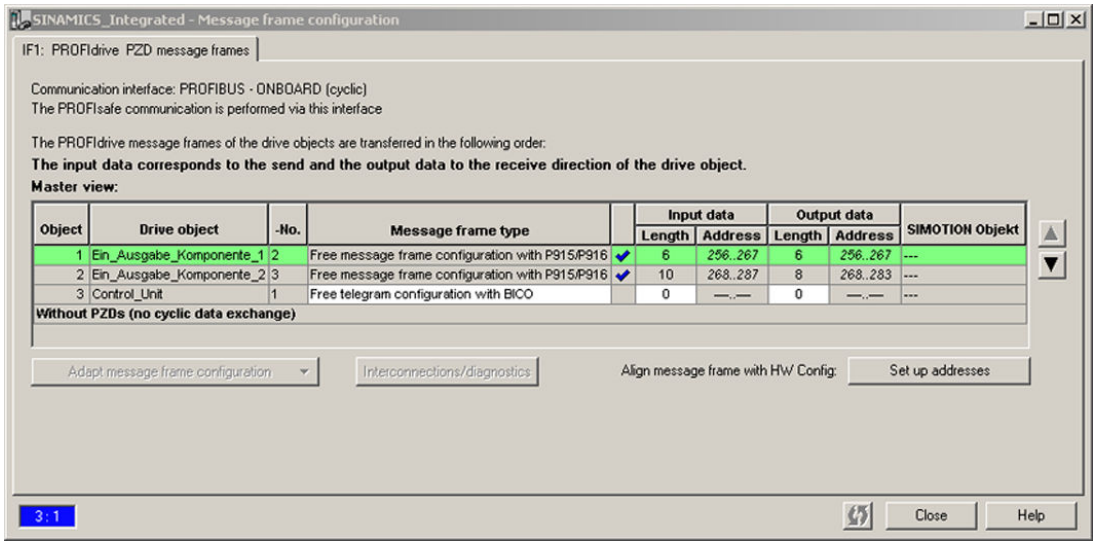


Figure 12-63 Aligning the configuration with the hardware

2. Click **Set up address** and then confirm with **Yes**. The addresses of the inputs and outputs are updated.

Note that the addresses have been assigned automatically. These addresses are the logical start addresses of the module in the message frame.

You must later add the offset values from the graphical configuration screen form to these addresses if you want to determine the addresses for the individual I/O channels.

As an alternative to the above procedure, you can also perform the alignment via the menu **Project** -> **Set up addresses**.

Notes

Note

When a TM1x is reconfigured, the structure of its message frame is changed, thus necessitating a realignment with HW Config. This process will create new (modified) addresses for the inputs and outputs. In addition, the I/O address assignment of the technology object must be updated.

Even if the operating mode of a measuring input is changed (once/cyclically), realignment with HW Config is required, although this does not generate any new (changed) addresses for the inputs and outputs.

Note

Before making changes in the configuration screen forms of the TM15 / TM17 High Feature terminal modules, you must close the screen form for aligning the hardware (see figure above).

Linking symbolic I/O variables with TM1x terminal modules

You can assign addresses to binary I/O channels using symbolic variables in the user program.

Procedure

In the following example, create the variable 'var_IO_DI0' and link this to input DI_0 and variable 'var_IO_DO1' with output DO_1.

For the addressing example, use the input and output addresses from the following figure.

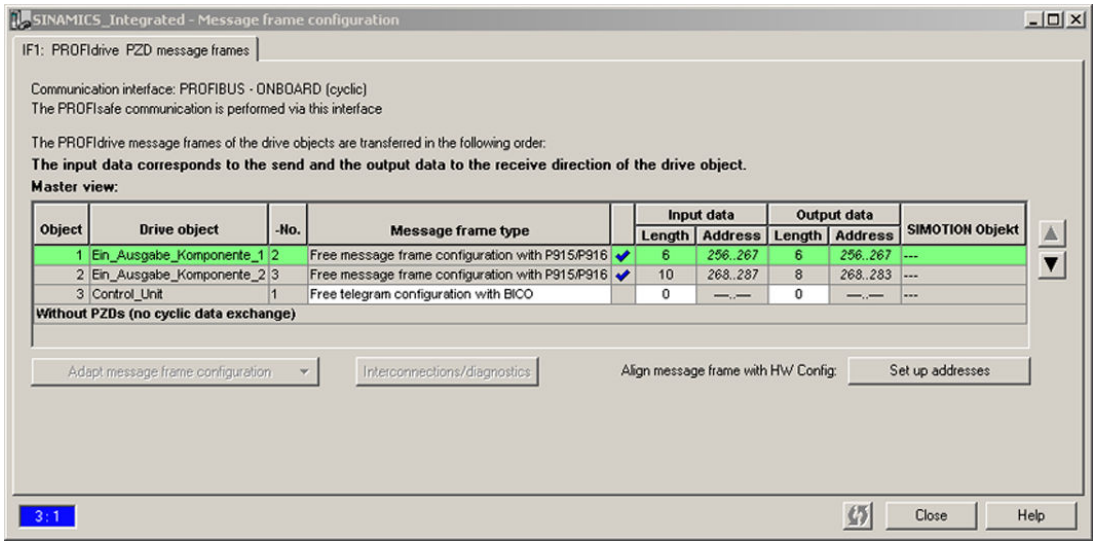


Figure 12-64 Aligning the configuration with the hardware

Refer to the following figure for the offset in the example.

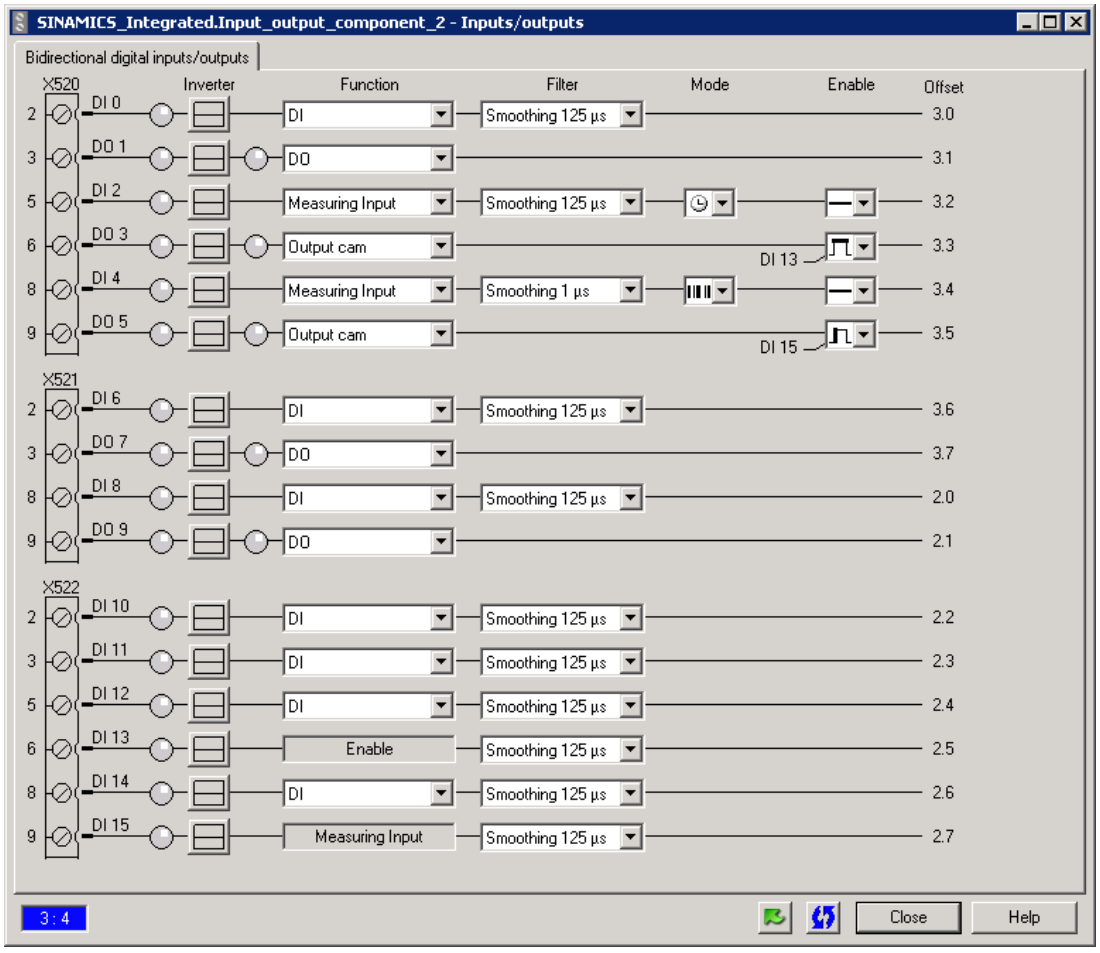


Figure 12-65 SCOUT – graphical configuration screen form for the TM17 (SIMOTION)

Input DI 0

The base "I-address" of the TM17 High Feature is "268". Now add the offset of 3.0. For DI 0, the resulting address is 271.0.

1. Open the **address list** via the project navigator.
2. Create the variable 'var_IO_DI0' and in the **I/O address** field, enter the value 'PI 271.0' (PI stands for input).

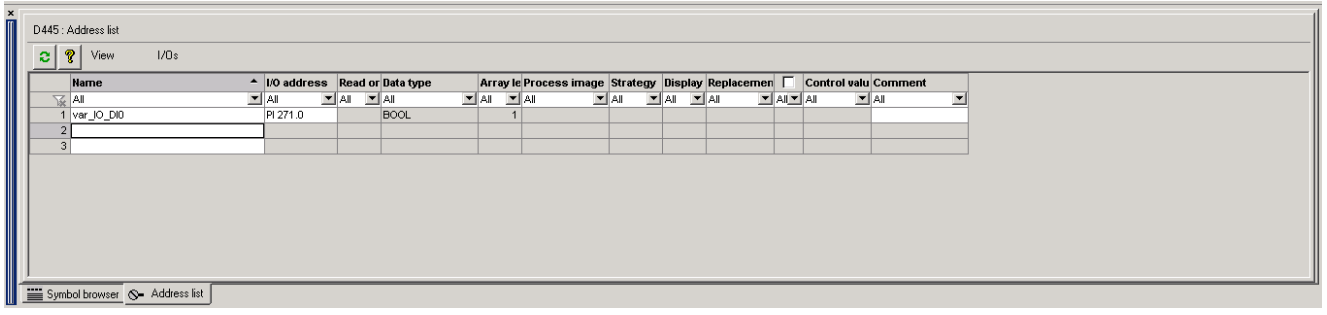


Figure 12-66 Creating variables in the address list

Output DO 1

The base "O-address" of the TM17 High Feature is "268". Now add the offset of 3.1. For DO 1, the resulting address is 271.1.

1. Open the **address list** via the project navigator.
2. Create the variable 'var_IO_DO1' and in the **I/O address** field, enter the value 'PQ 271.1' (PQ stands for output).

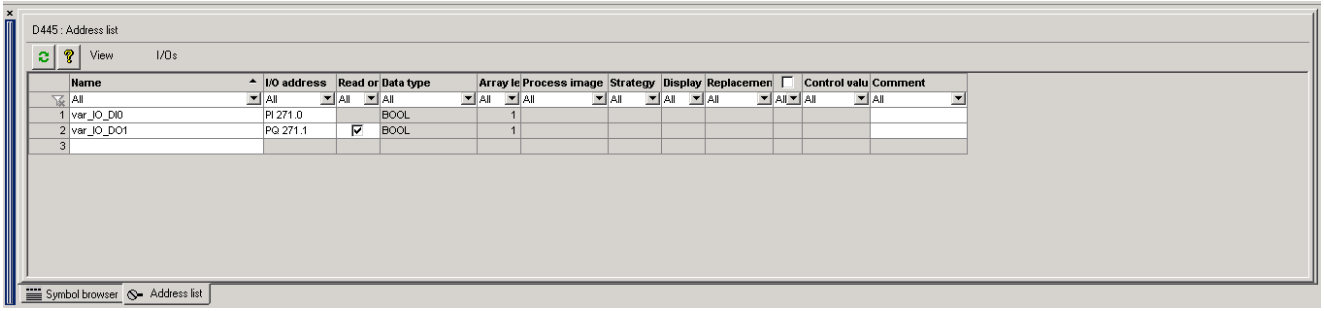


Figure 12-67 Creating variables in the address list

Notes

Note

The procedures for generating symbolic variables and for using the process image or direct I/O access are described in detail in the SIMOTION SCOUT Manual.

Note

You cannot use I/O variables to output to a digital output that is being used simultaneously by a TO outputCam or TO camTrack.

An error message will be output when the project is downloaded to the target system.

Please also note that the terminal module must have ramped up before it is accessed via I/Os. You can also configure an error strategy for I/O (CPU stop, substitute value or last value). Note that byte-specific setting is the minimum possible setting.

You cannot assign a substitute value for an output byte if individual bits have been allocated to an TO outputCam or TO camTrack.

See also

Power Up and Synchronization with the User Program (Page 8238)

Controlling the enable signal

If the enabling signal has been selected for one of the channels on the TM17 High Feature terminal module, the user program can control this hardware function by writing values to the output address assigned to the channel for the enabling signal.

The control bit of the enabling signal has the following function (= force function):

- "0" → The I/O channel (e.g. channel 0) is controlled by the channel for the enabling signal (e.g. channel 10).
- "1" → The I/O channel (e.g. channel 0) is enabled irrespective of the channel for the enabling signal (e.g. channel 10).

If the I/O channel is activated by a HW enabling signal during forcing, this enable is retained even after the forcing function is terminated (provided that the enable conditions are still fulfilled).

Example

In this example, DO 13 on the TM17 High Feature terminal module is used as the channel for the enabling signal (see the following figure).

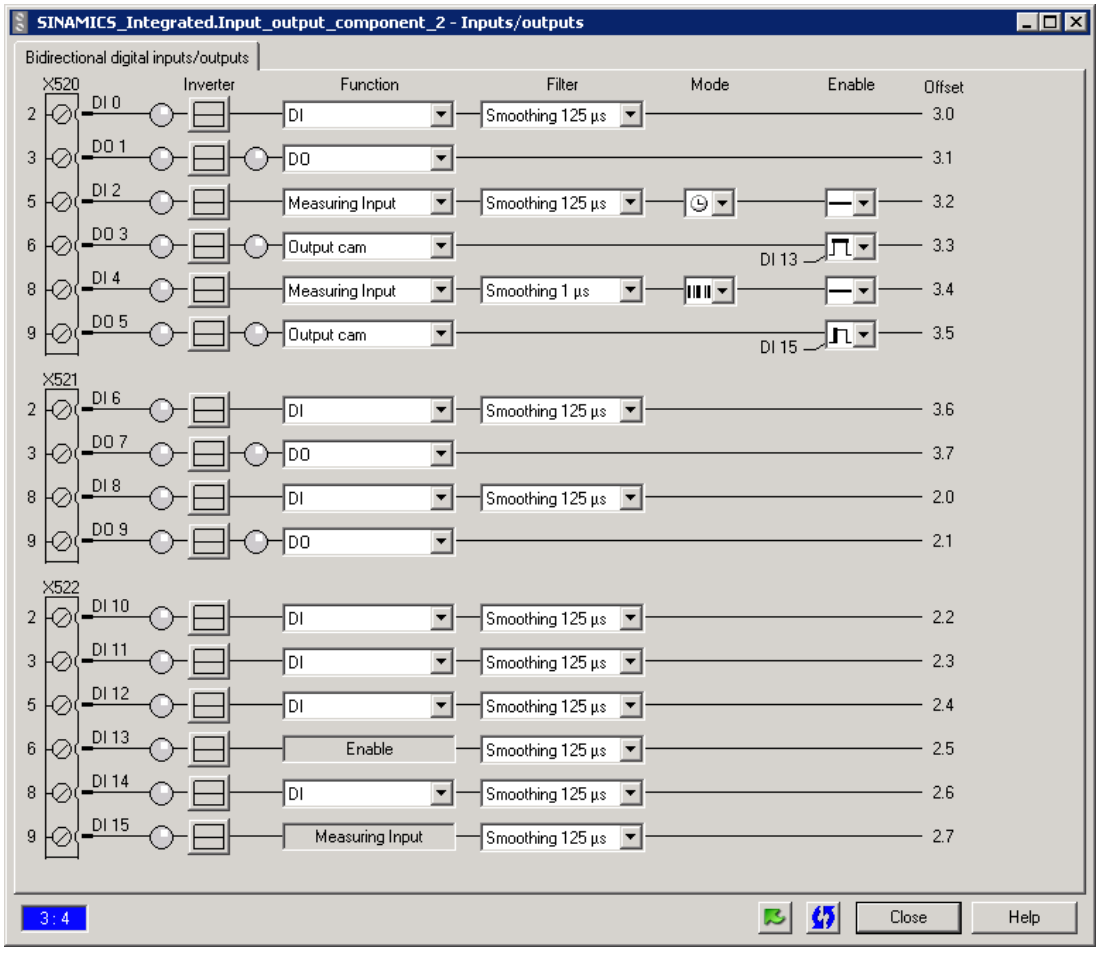


Figure 12-68 SCOUT – graphical configuration screen form for the TM17

Addressing the channel for the enabling signal

Table 12-9 Addressing the channel for the enabling signal

Channel for enabling signal	Assigned to I/O channel	I/O address of enable input (output address)
10	0	(Module address + 2).2
11	1	(Module address + 2).3
12	2	(Module address + 2).4
13	**3**	**(Module address + 2).5**
14	4	(Module address + 2).6
15	5	(Module address + 2).7

The output address for controlling the enabling signal DO 13 is "268" (output address of the TM17 HF in the example) plus an offset of "2.5" (see table). Consequently, the output address to control the enabling signal is 270.5.

Procedure

Create a variable in the **address list** and enter the value 'PQ 270.5' in the **I/O address** field.

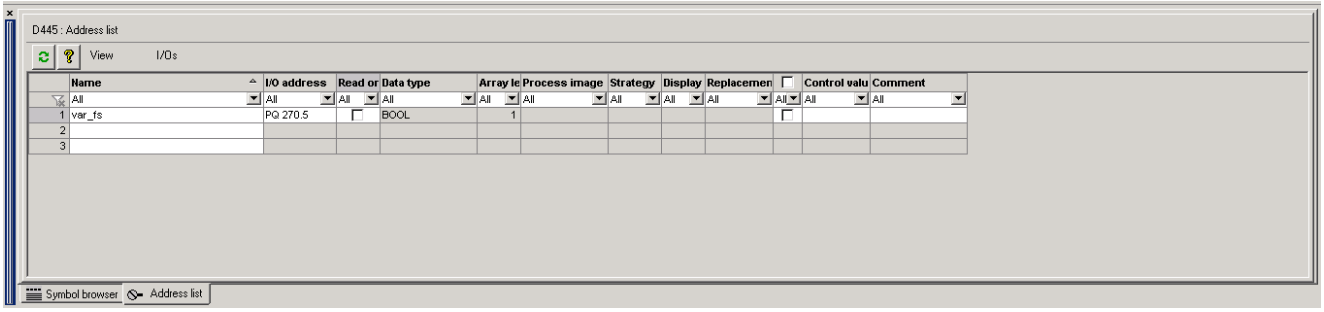


Figure 12-69 Creating a variable in the address list

Signal status of the enable signal

If required, the user program can read the signal state of the enabling signal using the assigned input address (e.g. for CAM 3: 268 + offset 2.5 = 270.5). The binary value reflects the signal state at time T_{in}.

Read back function for outputs

The user program can read back the signal state of each output channel on the TM15 or TM17 High Feature using the assigned input address (e.g. for DO1: 268 + offset 3.1 = 271.1). The binary value reflects the signal state at time T_{in}.

Note

If an inverted output channel is parameterized, the inverted terminal status is read during the read-back.

I/O assignment to technology objects

Linking technology objects with terminal modules

Measuring input inputs and cam outputs are operated with the associated technology objects (TOs). A general understanding of the use of technology objects is therefore needed to perform the desired configurations.

Note

The terminal module must be powered up before it can be accessed by technology objects. Otherwise, a technology alarm will be triggered.

The assignment of the measuring input inputs is performed as for the inputs (DI) and the assignment of the cam outputs as for the outputs (DO), but with the **CAM_n** or **MI_n** assignment partner.

You will find a detailed description of linking technology objects with terminal modules in the "SIMOTION Motion Control - Technology Objects for Output Cams and Measuring Inputs" Manual.

See also

Power Up and Synchronization with the User Program (Page 8238)

Power Up and Synchronization with the User Program

An isochronous bus is required for operation of the TM1x terminal modules.

The terminal modules must have powered up and reached the synchronized state before they can be read- or write-accessed. The synchronization of the terminal modules requires at least 18 servo cycles.

Until synchronization, the digital outputs are disabled (low level at the terminal).

The TM1x terminal modules are not ready to operate until they are successfully synchronized with the SIMOTION CPU. During this startup phase, the input and output variables must not be accessed directly or else an I/O access error will occur and the CPU will go to STOP mode.

In addition, all access attempts of the TO measuringInput, TO outputCam or TO camTrack in a non-synchronized state will trigger a technological alarm.

The start-up operation can be monitored by directly accessing the TM1x module status word or via the PeripheralFaultTask.

Monitoring synchronization by direct access to the module status word

The synchronization of the TM1x terminal modules is displayed cyclically to the SIMOTION motion control system via the module status word.

The synchronization bit **SYNC** is located in bit 8 of the module status word:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	-	-	ERR	**SYNC**	-	-	-	-	-	FPGA	PS	MF

"-" means reserved, cannot be used

Synchronization bit (bit 8):

SYNC = 0 → Module is not synchronized
SYNC = 1 → Module is synchronized

If you want to monitor the synchronization using I/O accesses, we recommend that you either configure an input WORD variable on the TM1x module status word with "substitute value=0" or perform the access by means of the secured **getSafeValue** system function.

If you are working with a SIMOTION SCOUT version < 4.1 SP2, you can only access the status word via the user program using a WORD variable (the individual bits must then be isolated by masking).

As soon as SYNC bit = 1, the module is synchronized and may be accessed directly via the I/O or technology objects.

Monitoring the synchronization with PeripheralFaultTask

During the transition from STARTUP to RUN, all TM1x Terminal Modules are in the "NOT_SYNCHRONIZED" state.

- As soon as the Terminal Modules have been successfully synchronized, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_SYNCHRONIZED" (=214) is called.
- If synchronization fails, the PeripheralFaultTask with interrupt ID "_SC_IO_MODULE_NOT_SYNCHRONIZED" (=215) is called.

Example

To synchronize the user task, a user variable **TM_SYNC** is set to **FALSE** in the StartUpTask and set to **TRUE** in the PeripheralFaultTask with interrupt ID = SC_IO_MODULE_SYNCHRONIZED. The status of **TM_SYNC** is queried in the user task before the (first) direct access operation.

The following TaskStartInfo is supplied in the PeripheralFaultTask every time it is called:

DINT	TSI#logBaseAdrIn	// valid only when not equal to _SC_INVALID_ADDRESS
DINT	TSI#logBaseAdrOut	// valid only when not equal to _SC_INVALID_ADDRESS
DINT	TSI#logDiagAdr	// valid only when not equal to _SC_INVALID_ADDRESS
DWORD	TSI#details	// set to 0
UINT	TSI#eventClass	// set to 0
UINT	TSI#faultId	// set to 0

The TaskStartInfo contains the logical address of the relevant module.

TSI#logDiagAdr, TSI#details, TSI#eventClass, and TSI#faultId are irrelevant for TM15/TM17 High Feature Terminal Modules.

For further information on the TaskStartInfo, refer to the "SIMOTION ST Structured Text" Manual.

12.3.3.7 Export/import project

A consistent project, which uses TM15 or TM17 High Feature modules, can be consistently exported and reimported via **Project Export/Import**.

If only the **SIMOTION CPU** is exported, the message frame configuration for TM1x is not exported with it. Thus, the current message frame configuration is not changed as a result of an import to a CPU.

If only the **SINAMICS drive unit** is exported, the message frame configuration for TM1x is not exported with it. The TM1x configuration must be recreated after an import.

Procedure

1. Select the SIMOTION CPU in the project tree and right-click to open the context menu.
2. Recreate the configuration via the menu **FastIO** > **Create new configuration**.
   The existing configuration is deleted.

12.3.3.8 Limitations of use

Overview

The maximum possible number of terminal modules for each SIMOTION D, SINAMICS control unit or CX32/CX32-2 depends on the following factors:

- Maximum number of terminal modules
- Maximum number of drive objects
- Maximum permissible message frame length
- Other limitations

Maximum number of terminal modules

The following maximum quantity structures apply when using terminal modules:

- The total number of TM15 (without the TM15 DI/DO variant) and TM17 High Feature terminal modules permitted per DRIVE-CLiQ line is three. This also applies when using a DMC20/DME20 DRIVE-CLiQ hub module. If there are more than three TM15 or TM17 High Feature terminal modules connected to a control unit in total, the maximum number of TM15 or TM17 High Feature terminal modules that can be installed on one DRIVE-CLiQ line is two. In addition, exactly one TM54F can be connected for each CU/D4xx/D4x5-2/CX32/CX32-2.
- The total number of all TM15 and TM17 High Feature terminal modules permitted per SIMOTION D4x5/D4x5-2, SINAMICS CU320/CU320-2 or CX32/CX32-2 is eight. TM54F is not included here, i.e. only a maximum of eight terminal modules can be connected, as well as an additional TM54F.
- The total number of all TM15 and TM17 High Feature terminal modules permitted per SIMOTION D410 or SINAMICS CU310 is three.

Note

For details of any other applicable restrictions, refer to the SIMOTION D and SINAMICS Manuals.

The TM15 and TM17 High Feature terminal modules require a DRIVE-CLiQ cycle time of at least 125 µs.

At present, the DRIVE-CLiQ cycle time always corresponds to the currently set current controller cycle clock. If a TM15, for example, is operated in conjunction with vector drives with a DRIVE-CLiQ cycle time > 125 µs, the resolution/accuracy of measuring input inputs or cam outputs deteriorates (resolution/accuracy identical to the DRIVE-CLiQ cycle time). Resolution and accuracy on the TM17 High Feature are not related to the cycle time in use.

The computing load of each TM15 / TM17 High Feature must be considered equivalent to one half of a SINAMICS axis, which can lead to a reduction in the number of axes on the part of drive control (does not apply for **TM15 DI/DO**).

If the TM15 or TM17 High Feature terminal module is operated on a DRIVE-CLiQ line with a cycle time less than 125 µs, communication with these modules is not possible.

Note

Exceeding the allowable number of terminal modules on a given DRIVE-CLiQ line can cause the DRIVE-CLiQ interface to cease communication.

Maximum number of drive objects

The maximum permissible number of drive objects per SIMOTION D, SINAMICS CU or SINAMICS CX32/CX32-2 depends on the controller / control unit used.

- SIMOTION D4x5-2, SINAMICS CU320-2 or SINAMICS CX32-2:
  Max. 24 drive objects
- SIMOTION D4x5, SINAMICS CU320 or SINAMICS CX32:
  Max. 16 drive objects
- SIMOTION D410 or SINAMICS CU310:
  Max. 5 drive objects

Examples of drive objects:

- CU320/CU320-2 control unit
- Regulated infeed
- CX32/CX32-2 extension module
- Motor modules (one double motor module constitutes two drive objects)
- TB30
- TM modules

Maximum permissible message frame length

Each terminal module uses two message frames to communicate with the SIMOTION motion control system:

- Setpoint message frame – contains the programmed setpoints and is sent by the SIMOTION Motion Control System to the TM.
- Actual value message frame – contains the current state of the I/O and is sent by the TM to the SIMOTION Motion Control System.

The maximum length of either the setpoint message frame or the actual value message frame is

- With integrated drives (SIMOTION D / CX32/CX32-2): 512 bytes
- With external drives (CU320/CU320-2/CU310/CU310-2):
  - On PROFIBUS DP: 244 bytes
  - On PROFINET IO: For CU320/CU320-2 with CBE20: 400 bytes
  - On PROFINET IO: For CU310-2 PN / CU320-2 PN (integrated PROFINET interface): 480 bytes

However, because the message frames are user-configured, the length of the message frames is determined by the current configuration, i.e. the actual length can be less.

Message frames contain all data for the drive unit, including:

- Message frame length for all modules of type TM15 / TM17 High Feature
- Message frame length for all axis message frames (e.g. standard message frame 5 has "PZD 9/9" → 9 words [18 bytes] in each direction)

- Message frame length for all line modules (e.g. SIEMENS message frame 370 for infeed with PZD 1/1 → 1 word [2 bytes] in each direction)
- .....

If the message frame contains too much data, the number of TMs must be reduced.

See also

Message frames (Page 8273)

Consistency checks

To ensure that applicable restrictions are not violated, SCOUT has the following built-in consistency checks:

- **At any time**, from the SCOUT main menu, "Check consistency" can be selected from the "Project" drop-down menu. (For more information, refer to the SCOUT online help.) This allows you to verify that the programmed configuration is consistent with the defined hardware configuration.
- **Before downloading the hardware configuration**, "Consistency Check" can be selected from the HW Config "Station" menu. (For more information, refer to the SCOUT online help.) This allows you to verify the integrity of the defined hardware configuration for such things as PROFIBUS settings, timing settings, SINAMICS parameters, etc.
- **Before saving a project in SCOUT**, you can check the consistency of the network by selecting the "Check consistency" option in the "Network" menu of NetPro. (For more information, refer to the SCOUT online help.) This allows you to verify the integrity of the network and ensure that such things as redundant network addresses, unconnected nodes, subnets that have only one node, inconsistent connections, etc., are not present.
- **When the SIMOTION Motion Control System is powering up**, a "topology test" is automatically carried out to ensure that the hardware is consistent with the programmed configuration.

12.3.4 Commissioning

12.3.4.1 Power-up

Once the terminal module is physically installed and connected electrically, power may be applied to the unit.

⚠ DANGER
Danger to life from improper commissioning Non-observance of the standards and safety regulations can result in serious injury or death. • For this reason, you must always observe the applicable standards and safety regulations when configuring a plant / system.

Status LED (RDY) of module

The status of the module and the DRIVE-CLiQ interface are indicated by means of a multicolored LED on the front panel of the TM. The individual colors are explained in the following table. The error display is the same as that used on other SINAMICS components.

Table 12-10 Module status

LEDs	Color	Status	Description
READY	-	Off	Electronics power supply outside the permissible tolerance range.
	Green	Continuous light	The component is ready for operation and cyclic DRIVE-CLiQ communication is taking place.
	Orange	Continuous light	DRIVE-CLiQ communication is being established.
	Red	Continuous light	This component has at least one fault.
	Green/Red	Flashing 2 Hz	Firmware is being downloaded.
	Green/Orange	Flashing 2 Hz	Component detection: no faults detected.
	Red/Orange	Flashing 2 Hz	Component detection: fault(s) detected.

Note

Interruption of module power supply: If the power supply to the module is interrupted, all outputs will switch to 0 V until synchronous communication resumes.

12.3.4.2 Updating the firmware

The module firmware can be updated using a CompactFlash card, which is inserted in the SIMOTION D or SINAMICS S120 control unit.

Procedure

1. Online connection for SCOUT
   In the project tree, click **SINAMICS_Integrated**.
   Go online by clicking the **Connect to target system** symbol in the main toolbar at the top.
   SCOUT is now connected to the SINAMICS device.

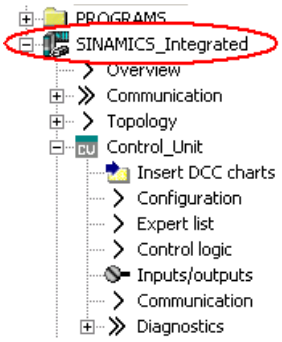


Figure 12-70 SCOUT project tree – Topology

2. Selection of a terminal module
   Double-click **Overview** under **SINAMICS_Integrated**. A window with two tabs **Overview** and **Version overview** appears (similar to figure below).
   The **Version overview** tab contains all the components that are connected to the SINAMICS control unit. The FW version currently installed on the component is displayed in the FW version column. The first two digits stand for the version, i.e. 24.... stands for version V2.4.

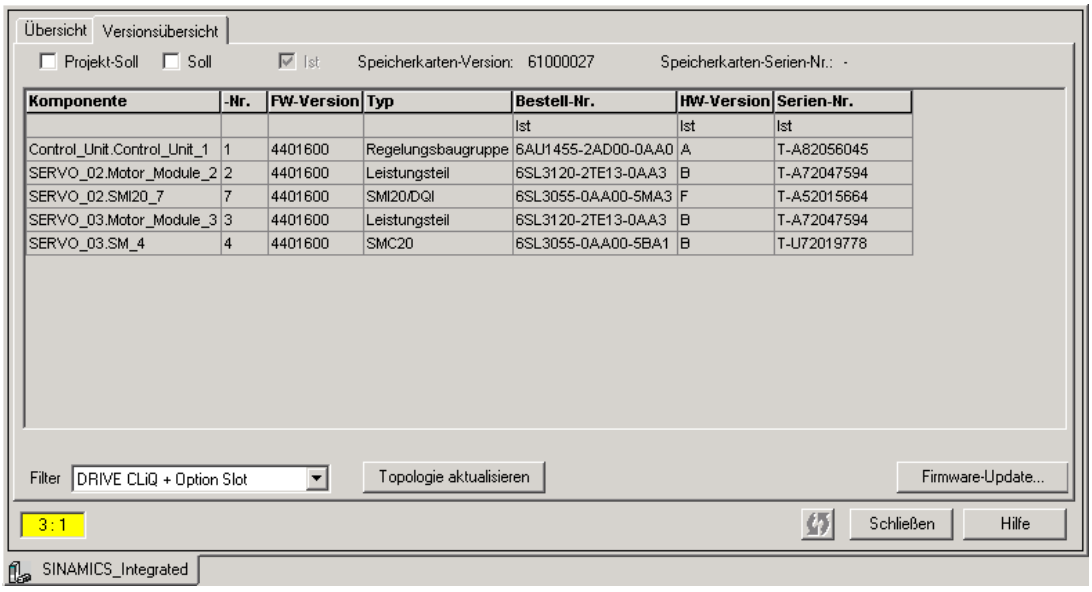


Figure 12-71 Screen displaying components connected to the control unit

3. Updating the firmware
   Right-click the **Firmware update** button. A window will open (similar to screen below).

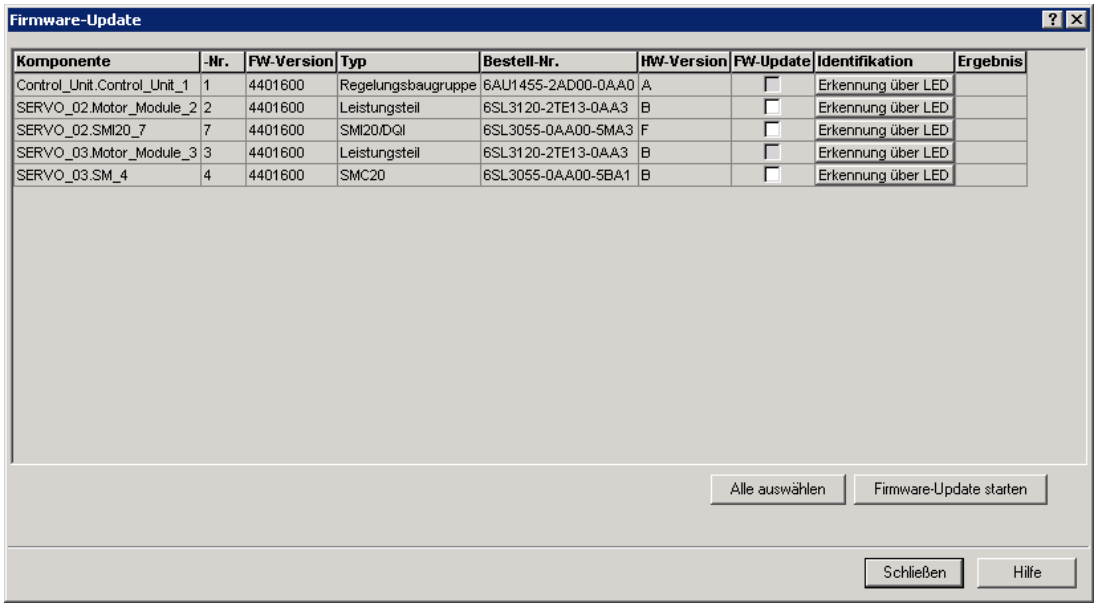


Figure 12-72 Firmware update

You can now select either **Select all** or an individual component in the **FW update** column; in the latter case, click the **Identification via LED** button first to check whether the correct module is selected. The RDY-LED on the selected component flashes green/orange (if no faults are detected) or red/orange (if faults are detected).

Select **Start firmware update** to perform the FW update for the selected components. The firmware update process may take a while - a message is displayed to indicate when it has been completed. During the update, the RDY-LED flashes green/red.

Now switch off the power supply for all drive components (SINAMICS control unit, CX32 extension module, SIMOTION D, terminal module, etc.) and switch it back on to activate the new firmware.

Note

The SIMOTION or SINAMICS CompactFlash card always contains the most recent firmware version for the module.

12.3.4.3 Synchronous mode

Note

This section applies only to systems with external drives since this type of drive must be synchronized during configuration of PROFIBUS/PROFINET. Conversely, systems with integrated drives (i.e. one using SIMOTION D or CX32) use an integrated PROFIBUS that is pre-configured and synchronized.

The TM15 and TM17 High Feature terminal modules must operate isochronously with the SIMOTION Motion Control System. This is achieved by exchanging sign-of-life signals.

Therefore, isochronous operation must be selected

- For PROFIBUS both on the master (SIMOTION) as well as on the slave (SINAMICS)
- For PROFINET both on the master (SIMOTION) as well as on the device (SINAMICS)

PROFIBUS procedure

To make this selection in SINAMICS:

1. Right-click the SINAMICS component in the project tree.
2. In the drop-down menu, select the **Open HW Config** option.
3. Right-click the SINAMICS icon.
4. In the drop-down menu, select the **Object properties** option.

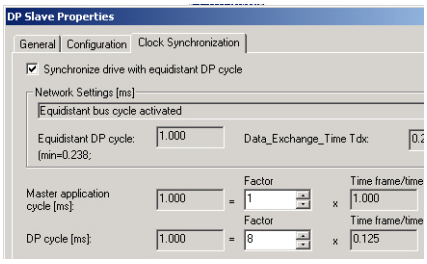


Figure 12-73 SCOUT - "DP Slave Properties" window

5. Select the **Isochronous operation**" tab.
6. Select the **Synchronize drive to equidistant DP cycle** checkbox.

If synchronization is lost (due to multiple consecutive transmission disturbances or interruption of the bus), a flag is set in the module's status word. Also, the module stops operation and all outputs are set to 0 V level. The module automatically attempts to restore the synchronization and to restart. This takes only a few position control cycle clocks if the restart is successful.

Note

To ensure error-free operation, we recommend that you program a synchronous status query in the user program before issuing control commands to technology objects or read/write accessing modules via I/O access.

See also

Error messages (Page 8248)

Power Up and Synchronization with the User Program (Page 8227)

PROFINET procedure

The basic procedure with PROFINET corresponds to PROFIBUS.

However, the synchronization must be activated in the properties of the device PROFINET connection in the **Application** tab.
In addition, a sync domain with the RT class **IRT top** must be selected in the **Synchronization** tab.

12.3.5 Error messages

Introduction

The error messages generated by the TM15 or TM17 High Feature modules are reported cyclically to the SIMOTION Motion Control System via the module status word as well as the diagnostic interrupt.

From SIMOTION V4.1 SP 1 onward, a diagnostic interrupt is also generated. Detailed diagnostic information can be found in the TSI#details of the TaskStartInfo.

The input address of the status word is identical to the start address (I-address) of the module.

Status word structure

The error bit corresponds to bit 9 of the module status word.

ERR = 0:	No error
ERR = 1:	Error. The cause of error is indicated via error bits MF, PS, and FPGA. For hotline purposes, drive parameter P2122 contains a detailed error code. **Error class if ERR = 1 is set:** MF: Internal module error PS: No I/O power supply FPGA: Error in FPGA code

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	-	-	-	-	CMP	-	**ERR**	SYNC	-	-	-	-	-	FPGA	PS	MF

-: Reserved, must not be used

ERR	ERR = 1 (error bit); indicates that the module is not ready; more details are specified with bits MF, PS and FPGA
SYNC	SYNC = 1 indicates that the TM1x module is synchronized
CMP	CMP = 1 indicates that the module is not ready and a firmware update is required.

Errors cannot be acknowledged. The error message is automatically deleted as soon as the device has been restarted or the cause of error has been corrected.

Note

The user program must always use a WORD variable to access the status word (individual bits must be isolated by masking).

Diagnostic Interrupt

In the case of an ERR or CMP error message in the status word of the module or errors in the TM1x configuration, diagnostic interrupt _SC_DIAGNOSTIC_INTERRUPT (=201) is triggered with the following TaskStartInfo:

```
DINT    TSI#logBaseAdrIn    // logical I-address of the module
DINT    TSI#logBaseAdrOut   // logical I-address of the module
DINT    TSI#logDiagAdr      // (valid only when not equal to(?)) _SC_INVALID_ADDRESS
DWORD   TSI#details         // Supplementary information
UINT    TSI#eventClass      // 0x39 = fault has occurred
                            // 0x38 = fault has been corrected
UINT    TSI#faultId         // 0x42
```

TSI#details supplies the following supplementary information:

Table 12-11 Supplementary information TSI#details of the diagnostic interrupt (DS0)

<table>
<tr><td rowspan="5">Bit 0 = 1
module fault 1)</td><td rowspan="3">Bit 1 = 1
internal error</td><td>Bit 7 = 1
parameter error 2)</td><td>TM parameters do not correspond to the TM1x configuration in SCOUT. For all SINAMICS devices, re-align them using HWConfig (see Aligning the configuration with the hardware (Page 8232)).</td></tr>
<tr><td>Bit 25 = 1
processor failure</td><td>MF:
Module status word bit 0 = 1
For notes see Error messages (Page 8248)</td></tr>
<tr><td>Bit 26 = 1
EPROM error</td><td>FPGA:
Module status word bit 2 = 1
For notes see Error messages (Page 8248)</td></tr>
<tr><td>Bit 2 = 1
external error</td><td>Bit 4 = 1
load current supply I/O is missing</td><td>PS:
Module status word bit 1 = 1
For notes see Error messages (Page 8248)</td></tr>
<tr><td colspan="2">Bit 16 = 1
incorrect module or incorrect firmware</td><td>CMP:
module status word bit 11 = 1.
Update the firmware for the module.</td></tr>
<tr><td colspan="3">Bit 8 = 1
Bit 9 = 1
Bit 10 = 1
Bit 11 = 1</td><td>SIMATIC module class
Digital I/O module</td></tr>
</table>

1) In addition to the diagnostic interrupt, an entry is made in the diagnostic buffer if at least one of the bits is set to "Module fault".

2) This alarm cause is not signaled in the status word of the module, but is generated by the parameter comparison. In addition to the diagnostic interrupt, the following entry is made in the diagnostic buffer for each individual inconsistency:
Parameters inconsistent for logical address input, reason xxx, note yyy

For further information on the TaskStartInfo, refer to the "SIMOTION ST Structured Text" Manual.

Error codes

Table 12-12 Error codes (for hotline)

ERROR CODE 35...	BITS	DESCRIPTION	REMEDY
...402	-	Inconsistent configuration in terms of sampling time (For detailed expert information, see P4099, note: delay time [µs])	Perform an upload, compile the project and download it to the target system.
...420	-	Inconsistent configuration of the measuring input channels. The TO measuringInput is connected to a TM1x channel that has not been configured as a measuring input input. (For detailed expert information, see P4029, P4049, note: channel number)	Compile the project and download it to the target system.
...430	-	Inconsistent configuration of the output cam channels. The TO outputCam / TO camTrack is connected to a TM1x channel that has not been configured as a cam output (For detailed expert information, see P4029, P4049, note: channel number)	Compile the project and download it to the target system.
...801 to ...905	MF	Internal module error or Communication problem on DRIVE-CLiQ interface	Replace the module and contact your local technical support.
...906	PS	At least one of the terminals for the I/O power supply is not connected or the I/O power supply is outside of the specified range. If the error is no longer present, the error code is retracted automatically.	Check all terminals for the I/O power supply.
...907	FPGA	FPGA programming error (TM17 High Feature only)	Try to update the firmware. If error is still present, replace the module and contact your local technical support.

The error code (see table above) is stored as parameter P2122 in the SINAMICS control unit. This code can be read by the user task with the "readDriveParameter" system function (see SIMOTION documentation "Reference List – System Functions / Variables").

Another option is to use the BICO technology editor to link parameter P2122 with an I/O word that has been added to one of the axis message frames of the SINAMICS control unit. This enables cyclic reading of the error code.

Error correction

If problems arise during commissioning of the TM15 or TM17 High Feature, please verify the following:

- Make sure that the planned application is permitted. Note the information in the SIMOTION general conditions list (see SCOUT CD).
- Check the firmware version of the TM15 / TM17 High Feature and update it if necessary.
- Create the FastIO configuration via SCOUT. Select, for example, the "D435" controller in the project tree. Right-click to open the context menu and select **FastIO** --> **Create new configuration**.
- If necessary, delete the user data on the CF card and completely reload the project. Then execute **Copy RAM to ROM**.
- Restart (power on or hot restart) the SINAMICS or SIMOTION D device. The new I/O channel parameters do not take effect until you execute the restart.

See also

Updating the firmware (Page 8244)

12.3.6 Application tips

12.3.6.1 Current controller cycle clocks <> 125 µs / use of output cams and measuring inputs

If current controller cycle clocks <> 125 µs are used, the parameter calculations of the drive must be transferred to the PG when using cam outputs on the TM15 / TM17 High Feature or for global measuring inputs, and Fast IO configuration generated again.

A change of the current controller cycle clock may have effects on the sampling times of the inputs/outputs on the drive side (e.g. TM15/TM17 High Feature, p4099 Sampling time of the inputs/outputs).

Sampling times <> 125 µs occur in the following cases:

- For servo drives with manual change of the current controller sampling time (drive parameters p0112 and p0115[0])
- For vector drives depending on the number of vector drives and with chassis units depending on the device type used
- If only an infeed and **no drives** are connected to the drive unit, then the sampling time is 250 µs.

For the cam outputs and the measuring input inputs (only for global measuring inputs) to function correctly, the sampling times must be known to the engineering system.

Table 12-13 Influence of the current controller cycle clock on the dead time compensation

	Current controller cycle clock has no effect on the function	**Current controller cycle clock has an effect on the function**
Cam outputs	SIMOTION D	TM15/TM17 High Feature
Measuring input inputs (global measuring inputs)	D4x5-2 (terminal X142)	• TM15/TM17 High Feature • SIMOTION D (except D4x5-2, terminal X142) • Controller extension (CX) • SINAMICS S110/S120 control units
Measuring input inputs (local measuring inputs)	---	---

In order for the changed cycle clock ratios to be taken into account by the engineering system, proceed as follows:

1. Go online and perform a project download. The SINAMICS performs parameter calculations once. These are automatically backed up on the CompactFlash Card
2. Perform an upload to the PG ("Target system" > Load" > "Load CPU / drive unit to PG").
3. This transfers the parameter calculations of the drive to the PG. The cycle clock ratios are then known in the engineering system.
4. Go offline.
5. Generate the configuration information (Fast IO configuration) again. To do this, select the SIMOTION CPU in the project tree and right-click to open the context menu "Fast IO" > "Create new configuration".
6. Click "Save project and compile all".
7. Go online and download the project to the target system.

SCOUT uses the described procedure to calculate internal system data that is required for outputting/detecting signals with a high level of position accuracy.

Note

If the cycle clock ratios are not set correctly, an appropriate message is output in the diagnostic buffer.

12.3.6.2 Tips on proximity switches

Proximity switches can be used in conjunction with the terminal modules to sense the presence or absence of a metallic surface. The advantages of using proximity switches over mechanical switches include the ability to perform non-contact sensing and a longer switch life.

Proximity switches contain electronic circuits in a compact, rugged enclosure. Note the following when using proximity switches:

- Use a **shielded conductor** from the switch to the DC input connection. This conductor should **not** be routed through a multi-signal connector.
- If an in-line connector is needed, use a **shielded connector** for each proximity switch cable. **Do not route any other signals through this connector**. This prevents crosstalk between pins of noisy cables and sensitive receive signals. If enough noise is injected on the 24 V supply (e.g., through testing or as a result of natural machine noise), the proximity switch circuitry will be re-initialized, causing faulty operation.
- The 24 V source (whether supplied from the TM or a separate power source) should originate at the module and should be included within the shield over the entire distance.

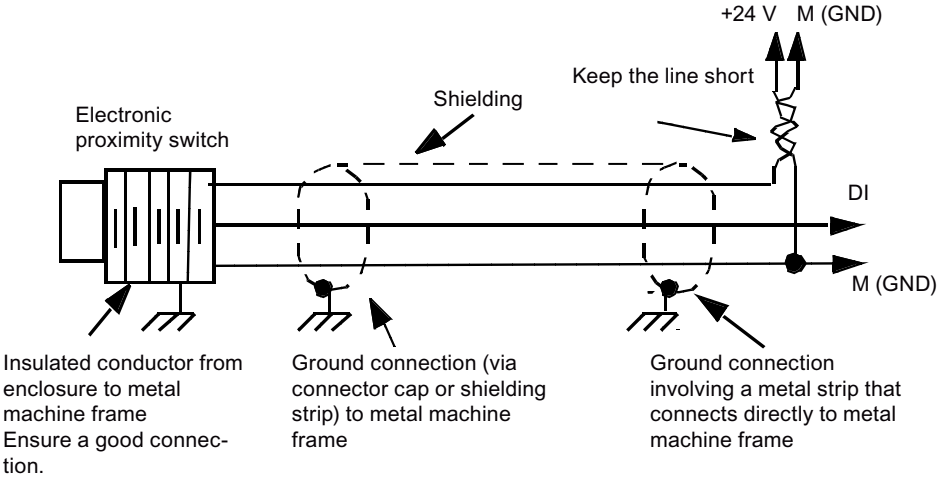


Figure 12-74 Proximity switch

Proximity switch cable shielding

The proximity switch shield should be grounded to the metal case on the switch side and to the shield connection on the terminal module at the other end. This grounding to the machine frame should be made in such a way that it will not corrode.

Machine safety and well-shielded grounding connections require good conductive connection between metal parts of machine components. Coatings and corrosion compromise the electrical continuity of the machine.

12.3.6.3 Information on leakage currents

Input configuration

Terminal module inputs occasionally receive their signal from an electronic device. Such a device can produce leakage current when the device is in de-energized state (OFF). If this current causes the input voltage at the terminal module to exceed 5 Volts, the circuit may not be able to detect whether this device is switched on or off. This may lead to faulty operation. If this occurs, a resistor must be added in parallel with the input circuit to divert a portion of this leakage current so that the input voltage does not exceed 5 VDC. The input resistance of the terminal module must be taken into account when selecting this resistor(2.8 kOhm for TM15 and TM17 High Feature). The wattage of the resistor must be great enough that the current consumption can be adapted to the output voltage of the switched-on source device.

Output configuration

Based on its design, the terminal module produces only 10 µA of leakage current for each output circuit. This is well below the level that would cause an electronic device to erroneously interpret an OFF state as an ON state. Therefore, unlike some output circuits that can have leakage currents as great as 2 mA, the TM requires no external resistor to divert leakage current.

12.3.6.4 Power switches ("SmartFETs")

The power switches used for outputs have built-in protection from excessive load currents and short circuits at the output. The hardware of the module monitors the overcurrent/over temperature signal.

Note

If the power switch is switched OFF due to an over temperature or overcurrent condition but the "ON" command persists, the output will be switched on again once the over temperature or overcurrent condition is eliminated.

The green channel LED illuminates when the power switch of the channel is switched on. A diode at each output protects the power switch from a reverse overvoltage failure when an inductive load is switched off.

You will find further information on handling inductive loads in our FAQs.

12.3.6.5 Input and output circuit

TM15

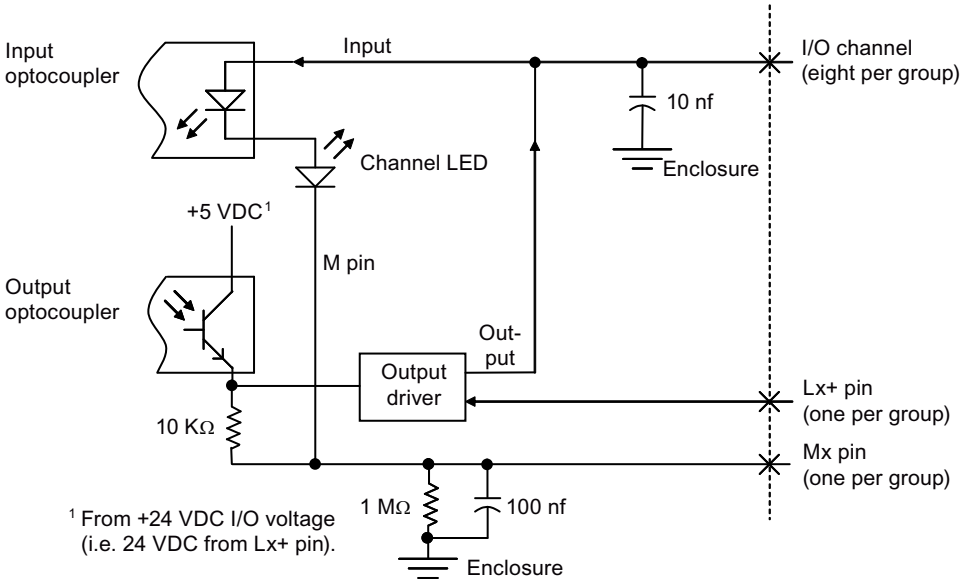


Figure 12-75 Input and output circuit - TM15

Grounding scheme

For TM15, the following are internally connected:

- Module power supply ground
- Logic ground
- TM chassis

The ground for the I/O power supply (terminals M1, M2, and M3) is capacitively coupled to the three grounds indicated above.

If terminal M1, M2 or M3 is connected to one of the three grounds indicated above, isolation for the associated group of I/O channels will be lost.

The I/O power supply ground (terminals M1, M2, and M3) must be connected to the reference ground of the current source in order for their associated I/O channels to function.

TM17 High Feature

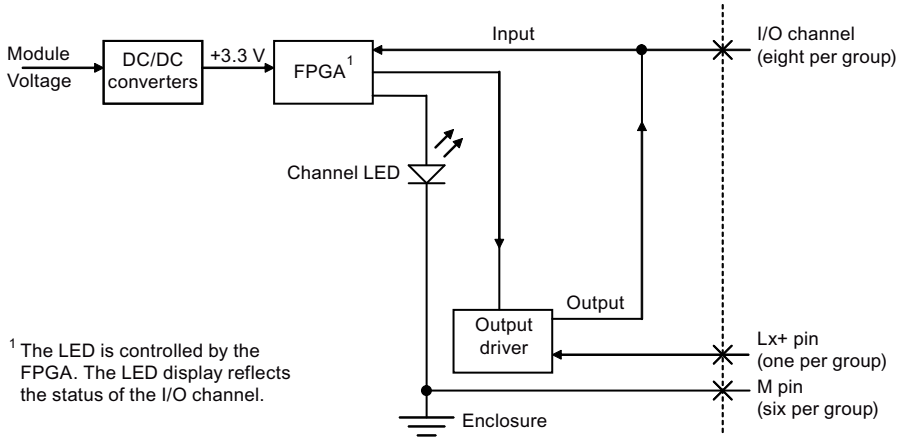


Figure 12-76 Input and output circuit - TM17

Grounding scheme

For TM17 High Feature, the following are internally connected:

- Module power supply ground
- Logic ground
- TM chassis

The ground of the I/O power supply is connected directly to the three grounds indicated above.

The ground connections for the I/O of the TM17 High Feature and not isolated. The ground of the I/O power supply (terminal M) must be connected to the reference ground of the current source in order for their associated I/O channels to function.

12.3.6.6 Other application examples

Use of inputs

The following figure shows an example of an input connection. Although TM15 is shown, the TM17 High Feature is wired in a similar manner. Note that no 24 VDC is required on the "L1+" terminal since it is only needed in an output application for the output driver. (See also "Supplementary SINAMICS System Components for SIMOTION" Manual.)

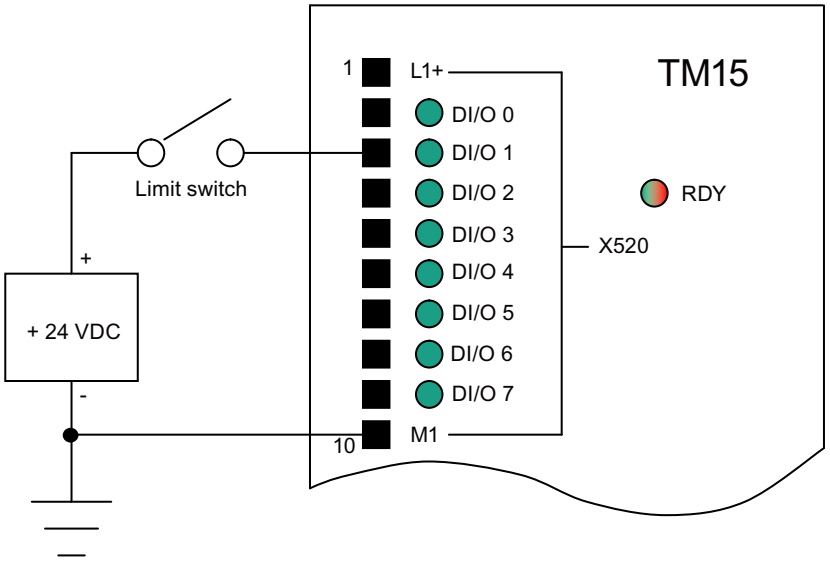


Figure 12-77 Example: Input circuitry on the TM15

Use of outputs

The following figure shows an example of an indicator lamp connection. Although the TM15 is shown, the TM17 High Feature can be wired in a similar manner. Please note that 24 VDC is required on the "L1+" terminal for the output driver to operate. (See also "Supplementary SINAMICS System Components for SIMOTION" Manual.)

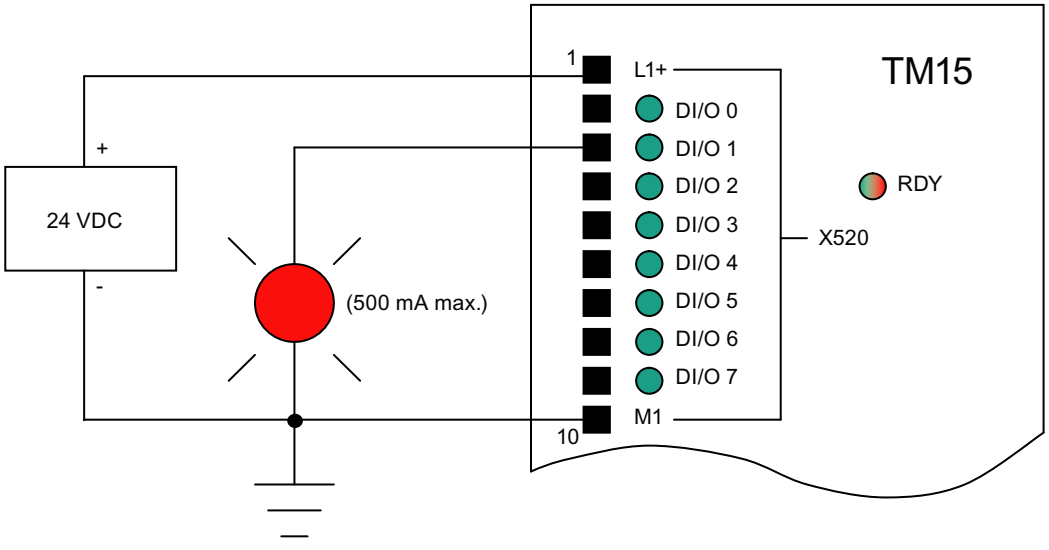


Figure 12-78 Example: Output circuitry on the TM15

Connection of a proximity switch

The following figure shows an example of a proximity switch connection. Several "M" (ground) terminals are available. In this example, the switch is supplied with 24 VDC via a DI/O (DI/O 2 in the example below). This DI/O may also be used to supply multiple switches, up to a current load of 500 mA per channel. (Typically, a 3-wire proximity switch requires about 150 mA.)

If DI/O 2 is configured as an inverted output, a 24 V (high level) supply is applied after system power up without being programmed. DI/O 3 must be configured as an input.

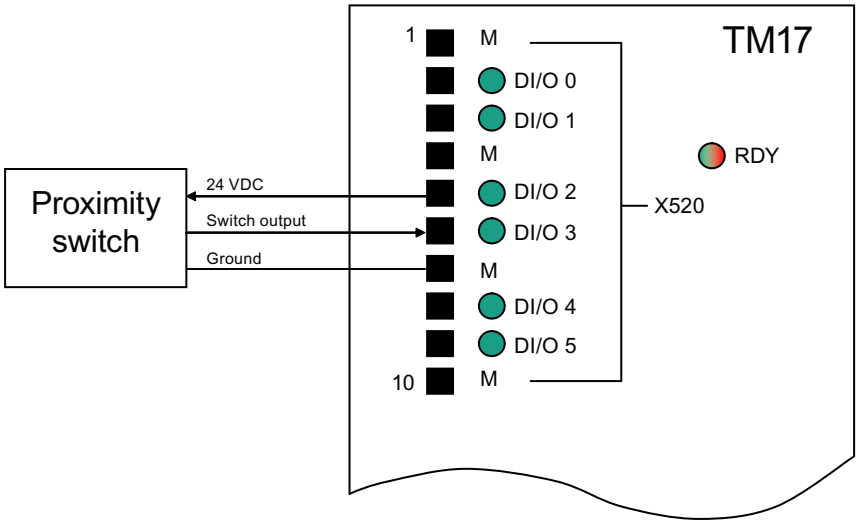


Figure 12-79 Example: Proximity switch connected to TM17 High Feature

Using the enable signal

This example illustrates the use of enable signals with outputs of output cams. When a workpiece is in the proper position, the output of the output cam is activated via the enable input. This ensures that the output cam (and thus the glue) is output only when a workpiece is present.

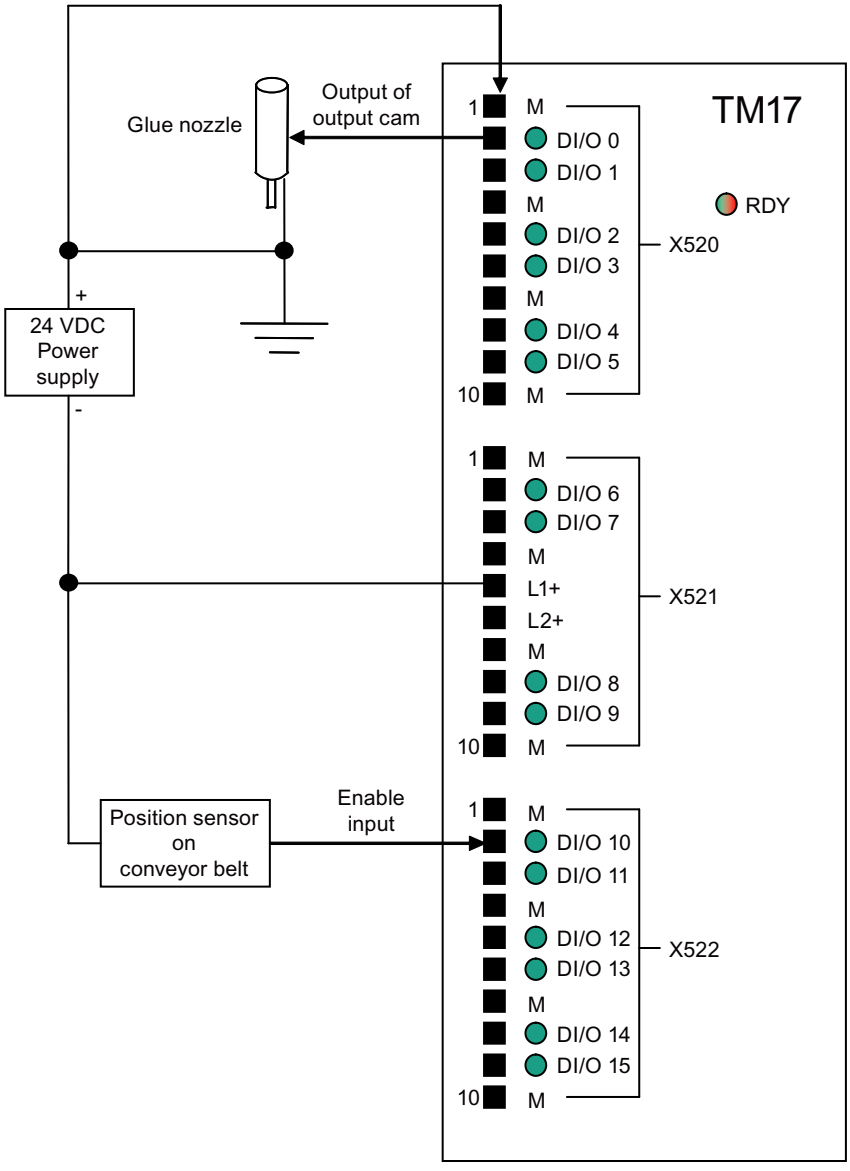


Figure 12-80 Controlling a glue nozzle using the output of an output cam

See also

Output modes (Page 8265)

Multiple µs-granular measuring ranges

A maximum of one measuring range can be specified for the Measuring Input technology object. Only measurements taken within this range are recorded.

The resolution of the measuring range of the Measuring Input technology object conforms to the measuring input processing cycle clock (IPO interpolator cycle clock, IPO2 interpolator cycle clock, or position control cycle clock).

If more than one measuring range is needed or if the measuring range is to be specified more precisely, this can be accomplished in the application as described below:

- Configure a Measuring Input technology object with level-triggered enable on the TM17 High Feature (measuring input e.g., DI/O 0; assigned enable input is therefore DI/O 10)
- Configure an output cam track, and output this to an output of an output cam of TM17 High Feature. (Output of output cam e.g., to DI/O 12)
- Connect the output of the output cam to the enable input of the measuring input

The output cam track is used to define the measuring range for the measuring input in the application (each output cam corresponds to one measuring range and can therefore be specified on a µs-specific basis).

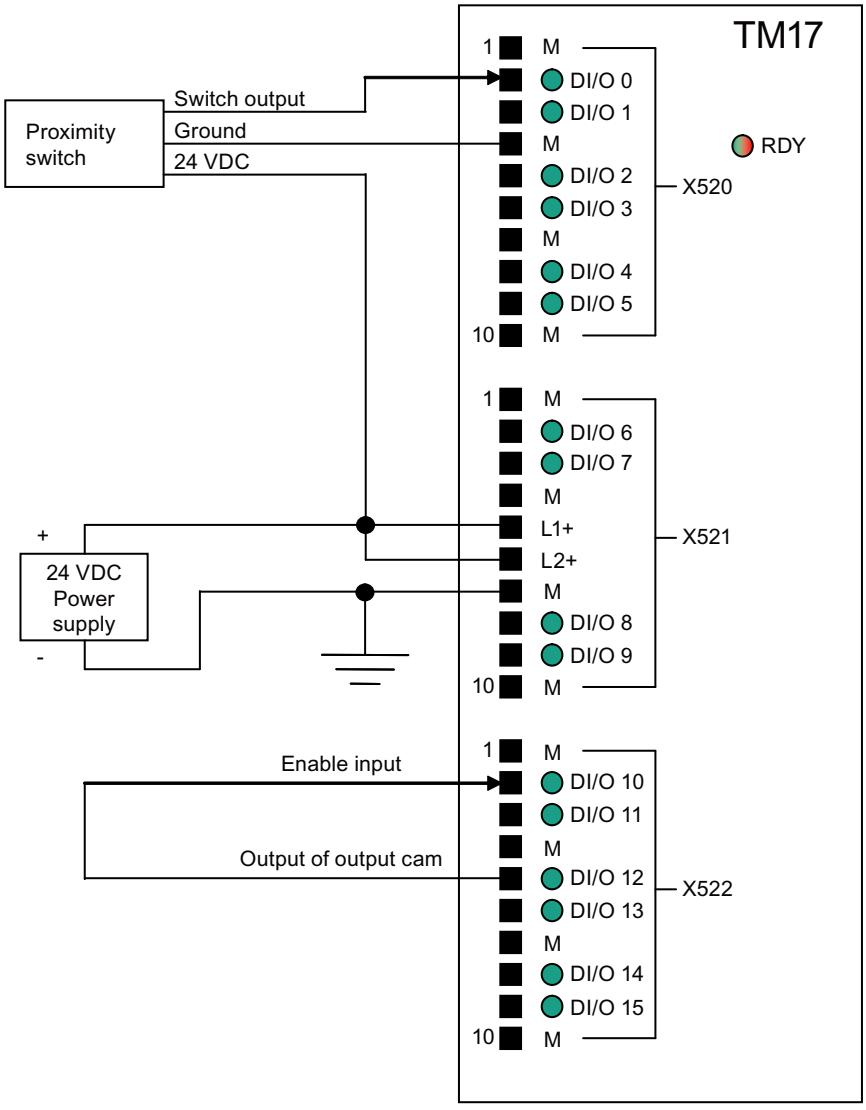


Figure 12-81 Multiple µs-granular measuring ranges

Acquisition of times / time-triggered output

If it is necessary to acquire times (e.g., measure the duration of a pulse) or to output a signal for a specific time period (e.g., output a pulse for a certain duration), a Measuring Input technology object or an Output Cam / Output Cam Track technology object connection to a virtual axis can be used for this purpose.

The virtual axis is moved at constant velocity, e.g., at 1,000 degrees per second.

Accordingly, an angular degree of 0.1 degree would correspond to a time period of 100 µs.

12.3.6.7 Frequently Asked Questions (FAQs)

Current FAQs on the Terminal Modules TM15 and TM17 High Feature are located at the following link:

http://support.automation.siemens.com/WW/view/de/24332926

12.3.7 Technical data

12.3.7.1 Operating modes

Overview of operating modes

Each channel can be individually configured for one of the following modes:

Table 12-14 Operating modes

Mode of operation	TM15	TM17 High Feature
Digital input (DI)	X	X
Input of measuring input (single measurement)	X	X
Input of measuring input (cyclic measurement)	-	X
Input of measuring input with level-triggered enable	-	X (channels 0 to 5)
Enabling signal	-	X (channels 10...15)
Digital output (DO)	X	X
Output of output cam	X	X
Output of output cam with level-triggered enable	-	X (channels 0 to 5)
Output of output cam with edge-triggered enable	-	X (channels 0 to 5)

Input modes

Digital input (TM15/TM17 High Feature)

If an I/O channel has been configured as a DI, it operates as a standard digital input.

Input of measuring input (TM15/TM17 High Feature)

If an I/O channel has been configured as an input of a measuring input, a time value is acquired with the input edge. This value represents the time of the event referenced to the local system time.

Note

The resolution of the measuring time varies according to the type of module:

- In the case of the TM15, the resolution corresponds to the DRIVE CLiQ clock cycle, but is at least 125 µs.
- In the case of the TM17 High Feature, the resolution is 1 µs.

Enabling signals (TM17 High Feature only)

Inputs of measuring inputs can be activated with enabling signals (= gate function). Specifically, channels 10 through 15 – if programmed as enabling signals – enable channels 0 through 5, respectively. Each enabling signal is level-triggered.

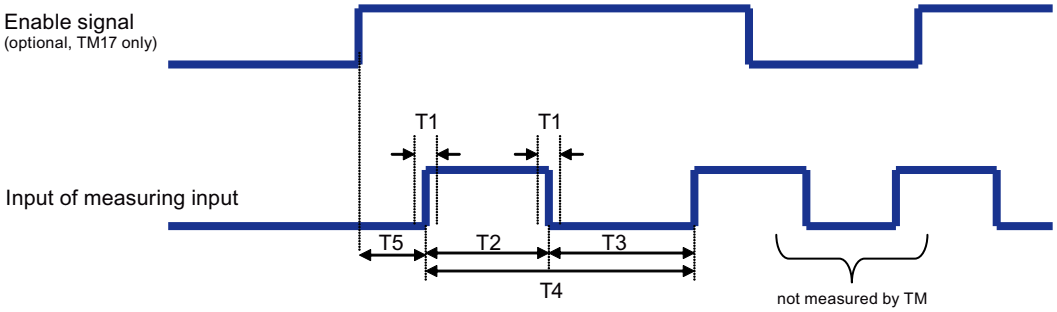


Figure 12-82 Diagram – Control of inputs

Table 12-15 Timing for input of measuring input

Timer	TM15	TM17 High Feature	Comments
T1	125 µs [1]	1 µs	Resolution
T2	≥ 125 µs [1]	≥ 1 µs/125 µs	TM17: Depends on selected filter times
T3	≥ 125 µs [1]	≥ 1 µs/125 µs	TM17: Depends on selected filter times
T4	Depends on ratio of IPO to servo		
T5	Not applicable (TM15 does not have an enable function)	≥ 1 µs/125 µs	TM17: Depends on selected filter times

Times specified for positive logic.

IPO = Interpolator cycle clock

[1] DRIVE CLiQ cycle clock, but at least 125 µs

See also

System behavior with binary inputs and outputs (Page 8268)

System timing for single measurement (Page 8270)

System behavior with cyclic measurement (Page 8271)

Configuring I/O channels – TM17 High Feature (Page 8212)

Output modes

Digital output (TM15/TM17 High Feature)

If a channel has been configured as a DO, it operates as a standard digital output. See figure under "System behavior for digital inputs and outputs"

Output of output cam (TM15/TM17 High Feature)

This operating mode treats output channels as outputs of an output cam (see section on "System Behavior"). The outputs are enabled and disabled depending on the time values calculated in the SIMOTION technology object.

Switching times are specified by two time values (ON and OFF values) referenced to the local system time.

Note

The resolution of the switching times varies according to the type of module:

- In the case of the TM15, the resolution corresponds to the DRIVE CLiQ clock cycle, but is at least 125 µs.
- In the case of the TM17 High Feature, the resolution is 1 µs.

See also

System behavior with binary inputs and outputs (Page 8268)

System behavior with outputs of output cam (Page 8272)

Output of output cam with enable signal (TM17 High Feature only)

The "Output of output cam with enable" mode treats output channels as outputs of an output cam, for which an enable signal is required (available on channels 0 through 5 only). This mode is selected on a channel-by-channel basis during I/O configuration.

Provided that an enable signal exists, output cams are output as a function of the time values calculated in the SIMOTION technology object.

Switching times are specified by two time values (ON and OFF values) referenced to the local system time.

Note

Provided that the enable function for outputs of output cams is used and the output cams are to be output "inverted", an inverted output must be assigned for TM17 High Feature (an inversion of the output on the Output Cam or Output Cam Track technology object causes a low level to be output whenever there is no enable, which would correspond in this case to an activated output cam).

Level-triggered enable

With a level-triggered enable, output cams are only output to the output if the enable signal is applied statically before the output cam start.

If the enable signal is retracted before the output cam end, the output cam is still output up to the end. New output cams are no longer output.

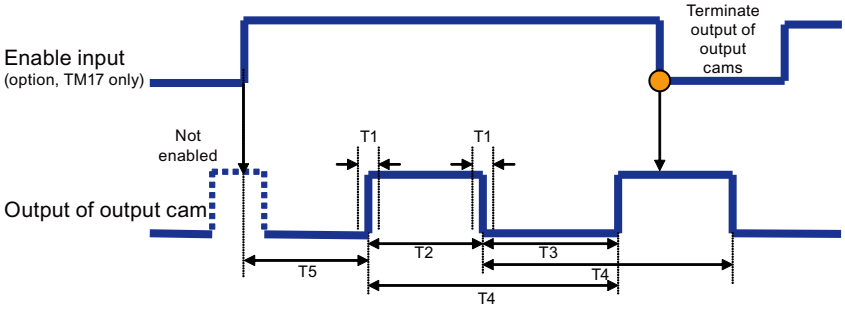


Figure 12-83 Diagram – Output of output cam with level-triggered enable signal

Table 12-16 Timing of output of output cam

Timer	TM15	TM17 High Feature	Comments
T1	125 µs [1]	1 µs	Resolution
T2	≥ 125 µs [1]	≥ 50 µs (typical) ≥ 100 µs (maximum)	Depends on the load conditions of the output. See "Supplementary SINAMICS System Components for SIMOTION" Manual.
T3	≥ 125 µs (typical) [1] ≥ 150 µs (maximum)	≥ 75 µs (typical) ≥ 250 µs (maximum)	Depends on the load conditions of the output. See "Supplementary SINAMICS System Components for SIMOTION" Manual.
T4	IPO or servo		Depends on the technology object cycle time.
T5	Not applicable (TM15 does not have an enable function)	≥ 1 µs/125 µs	Depends on selected filter times. See "Supplementary SINAMICS System Components for SIMOTION" Manual.

Times specified for positive logic.

IPO = Interpolator cycle clock

[1] DRIVE CLiQ cycle clock, but at least 125 µs

Edge-triggered enable

With an edge-triggered enable, output cams are output at the output only if the enable status exists before the output cam start. The enable status is controlled by a configured Measuring Input Technology Object.

For the Measuring Input TO, only the "one-time measuring" mode is permitted with the following edge detection:

- Rising edge
- Falling edge
- Both edges

Measuring of "Both edges, first rising" and "Both edges, first falling" and the "Cyclic measuring" mode is not supported.

Via the Measuring Input TO

- the positions of the enable edges can be evaluated (see "SIMOTION Motion Control - Technology Objects for Output Cams and Measuring Inputs" Manual). Please note that the measuring range is monitored in the IPO, IPO2 or position control cycle clock, depending on the configured measuring input cycle clock.
- Set or reset the enable status.
  The enable status is set when the configured edge arrives.
  The enable status is reset
  - At the beginning of the measuring range configured at the Measuring Input TO (with range setting: "Measure in specified range")
  - By program command _enableMeasuringInput (with range setting: "Measure without a defined range")

The enable status can be monitored via the I/O area of the enable input.

If the enable status is retracted before the output cam ends, the output cam is still output until the end if it has already started. New output cams are no longer output.

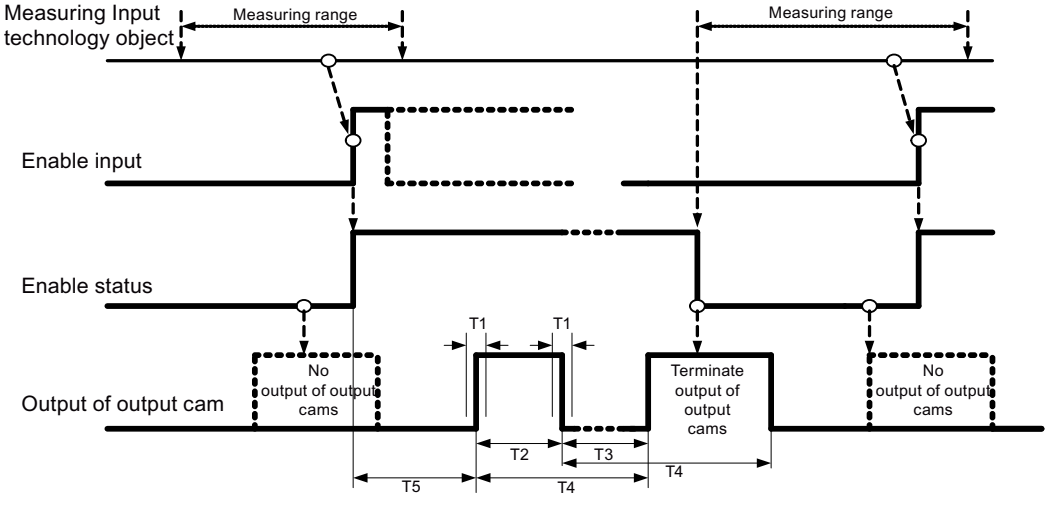


Figure 12-84 Diagram - Output of output cam with edge-triggered enable signal

Note

Times T1 to T5 correspond to the times in the table in section on "Level-Triggered Enabling".

12.3.7.2 System behavior

The following timing data relate to a clock cycle setting of

IPO: Servo: Bus cycle clock = 1 : 1 : 1 and a scan routine in the IPO synchronous task.

Note

The following describes the system behavior for PROFIBUS and can also, in principle, lay the basis for PROFINET configurations.

System behavior with binary inputs and outputs

The system behavior for **digital inputs and outputs** is shown in the figure below:

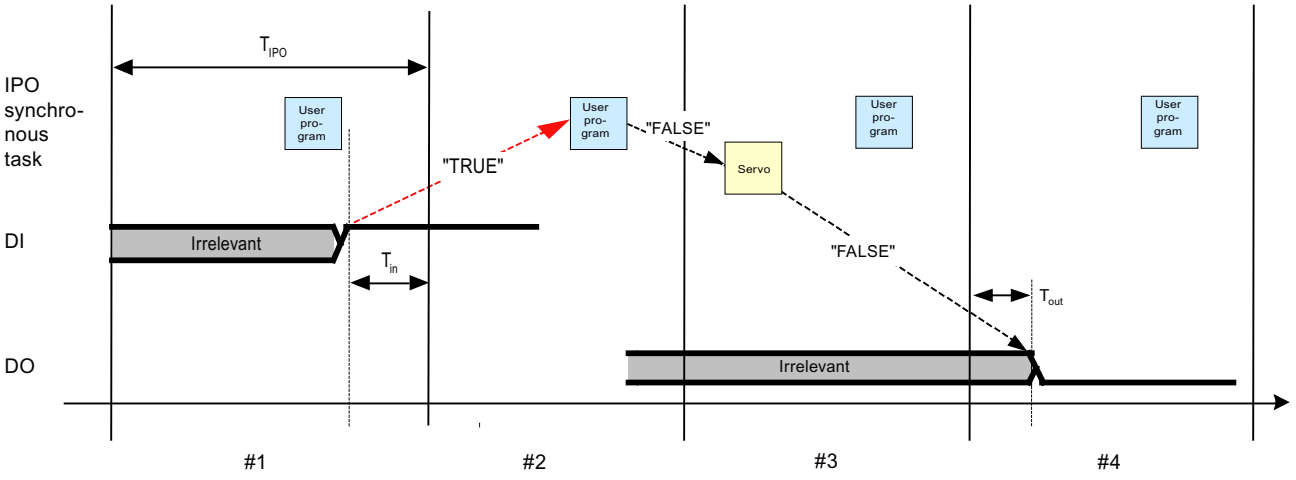


Figure 12-85 System behavior with digital inputs and outputs (TM15 and TM17 High Feature) with user program in the IPO synchronous task level

A digital input channel (DI) is always sampled at time T_{in} at the end of each IPO cycle clock. A digital output channel (DO) is always activated at time T_{out} at the beginning of the IPO cycle clock.

If the user program is executed in the IPO synchronous task, a DI state change (TRUE) in the first IPO cycle clock (#1) is detected in the next IPO cycle (#2). If an output signal (FALSE) is activated in response, it is first processed in position control cycle clock #3 and then transmitted to the module via PROFIBUS at the beginning of the next IPO cycle clock #4 (assuming an IPO: servo ratio of 1:1).

$T_{in} = T_i$ + 1 x DRIVE CLiQ cycle; SINAMICS: $T_i \geq 125$ µs

$T_{out} = T_o$ + 1 x DRIVE CLiQ cycle SINAMICS: $T_o \geq 125$ µs

where T_i and T_o are isochronous PROFIBUS time parameters. The value of T_i and T_o depends on the other PROFIBUS nodes; you can read off the exact value from the "DP Slave Properties" screen form in "HW Config". (In PROFINET screen "Properties - CBE20-PN" of the device.)

Note

If the user program is executed in the servo-synchronous user task rather than the IPO synchronous user task, the output signal is output one IPO cycle clock earlier.

Note

In order to operate the TM1X, the cycle time of the DRIVE CLiQ line must be at least 125 µs.

It is also possible to run a user program in the servo-synchronous task level. In this case, a user-programmed response in position control cycle clock #3 (terminal-terminal response time) is possible when there is a DI state change in position control cycle clock #1.

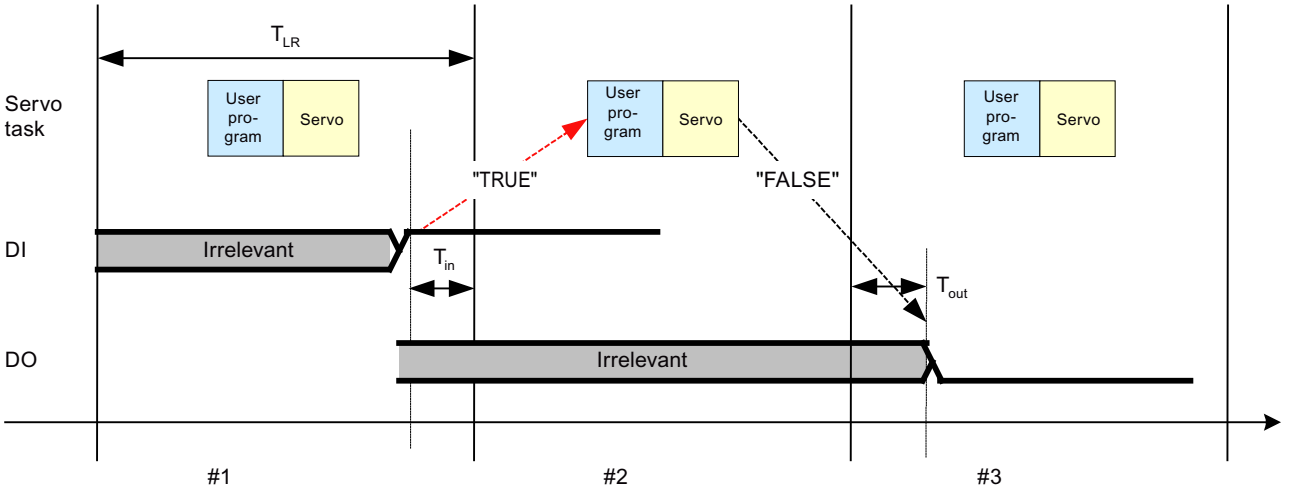


Figure 12-86 System behavior with digital inputs and outputs (TM15 and TM17 High Feature) with user program in the servo-synchronous task level

System timing for single measurement

The system behavior for a single measurement is shown in the figure below:

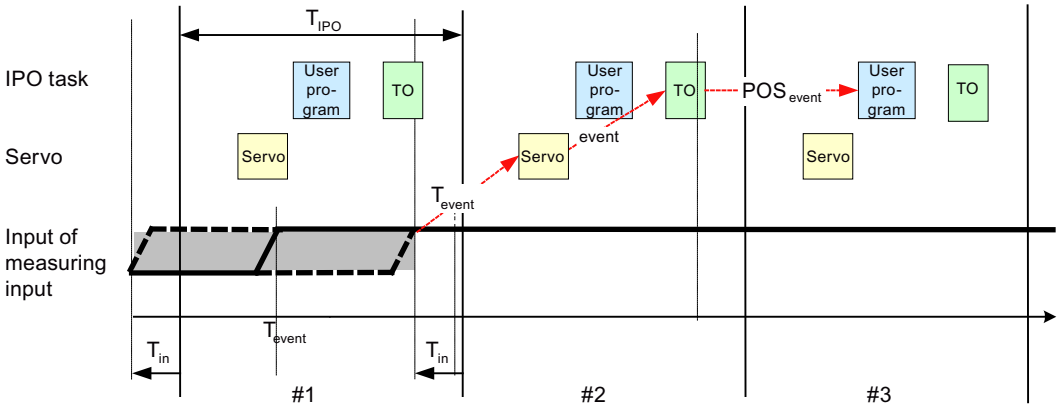


Figure 12-87 System behavior for single measurement (call Measuring Input technology object and user program in the IPO cycle clock)

If a measuring input event occurs in the shaded area in IPO cycle clock #1, then the actual event time T_{event} is transferred at the beginning of the following IPO cycle clock. In the Measuring Input technology object (TO), T_{event} is converted to a position value (POS) that is available to the user program in IPO cycle clock #3.

The time-related behavior for single measurement depends on two parameters:

- The ratio between the IPO cycle time and the cycle time of the position control cycle clock (i.e. ratio of IPO to servo).
- The time level to which the Measuring Input technology object is assigned.

The following table shows the minimum sampling cycle.

The minimum sampling cycle defines the time between two measurements (e.g. the time between detection of two print marks).

Table 12-17 Minimum sampling cycle – Single measurement

TO executed in IPO cycle clock	TO executed in position control cycle clock
5 position control cycle clocks + 2 IPO cycle clocks	6 position control cycle clocks + 1 IPO cycle clock

Each sampling cycle is started by a measurement command from the technology object. The TM15 and TM17 High Feature Terminal Modules are capable of sampling a maximum of two edges within one sampling cycle.

The minimum pulse width depends on the specified edge selection:

Edge selection	Minimum distance	Explanation
Rising edge	Minimum sampling cycle (see "Minimum Sampling Cycle" table).	Each new edge requires a new measurement command.
Falling edge		

Edge selection	Minimum distance	Explanation
Both edges	>1 µs/125 µs[1] (TM17) >125 µs (TM15)	Both edges are sampled within one sampling cycle.
Both edges, rising edge first		Minimum distance between first measurement edge and previous edge = 4 position control clock cycles
Both edges, falling edge first		

[1] Depending on filter selection.

Note

The parameter assignment screen for the Measuring Input technology object contains **two** fields for edge selection:

- Edge selection for (single) measurement
- Edge selection for cyclic measurement

System behavior with cyclic measurement

Up to two edges can be measured in each execution cycle of the Measuring Input TO (IPO interpolation cycle clock, IPO2 interpolation cycle clock or position control cycle clock).

The measured values must be read from the user program before they can be overwritten by a new measurement.

Result:

- A maximum of two edges can be evaluated per IPO cycle clock if the scan routine of the user program is in the IPO synchronous task.
- A maximum of two edges can be evaluated per position control cycle clock if the scan routine of the user program is in the servo-synchronous task

The servo-synchronous task is the lowest possible time level for a user program.

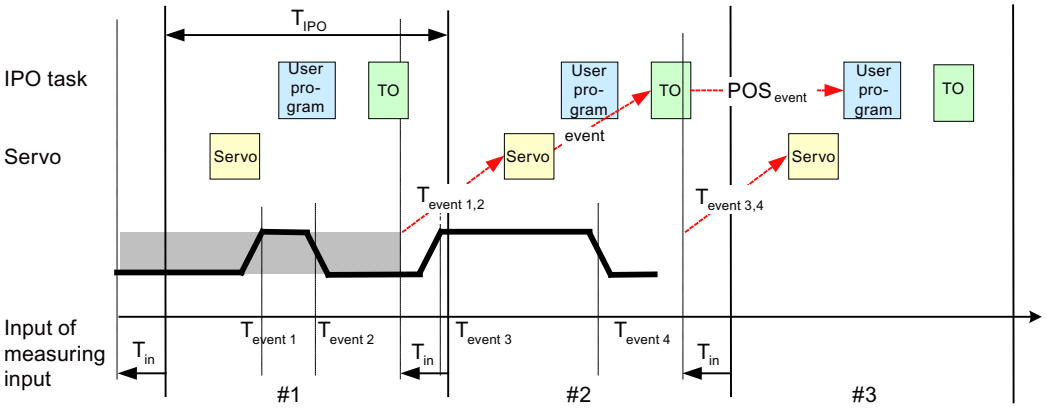


Figure 12-88 System behavior with cyclic measurement (call TO Measuring Input and user program in the IPO cycle clock)

Note

If no edges are measured in a clock cycle, the TM17 High Feature can measure up to 4 edges in the following clock cycle, whereby the 3rd and, if necessary, 4th edges are buffered (buffer of up to 2 edges).

In the following cycle clock, the buffered edges are transferred first even if new edges were recorded; these new edges are buffered again.

Table 12-18 Minimum time between two edges (cyclic measuring)

Edge selection	Minimum distance	Explanation
Rising edge	10 µs with filter setting of 1 µs	however, a maximum of 2 edges per task cycle of the scan routine
Falling edge	500 µs with filter setting of 125 µs	
Both edges	> 1 µs with filter setting of 1 µs > 125 µs with filter setting of 125 µs	however, a maximum of 2 edges per task cycle of the scan routine

Note

The parameter assignment screen for the Measuring Input technology object contains **two** fields for edge selection:

- Edge selection for (single) measurement

- Edge selection for cyclic measurement

System behavior with outputs of output cam

The system behavior for outputs of output cam is shown in the figure below:

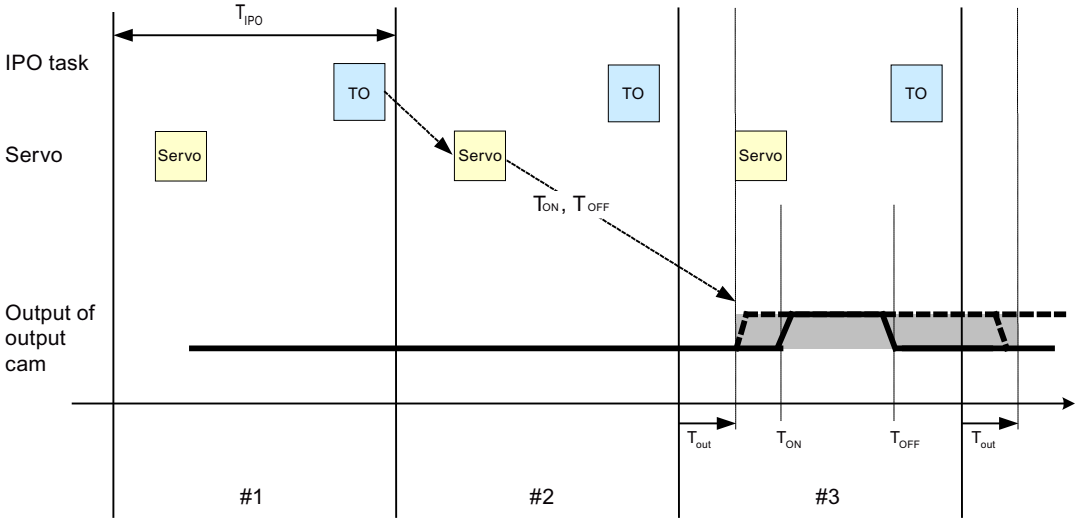


Figure 12-89 System behavior with outputs of output cams (call TO Output Cam and user program in the IPO cycle clock)

In order to switch an output of an output cam in the shaded area of IPO #3, the Output Cam technology object (TO) determines the switching positions in IPO #1. These switching positions are converted in the following position control clock cycle to time stamp T_{on} and T_{off} and then transferred at the beginning of IPO #3.

The same system timing applies to the Output Cam Track technology object.

Interpolation

A linear interpolation is used to calculate the output cam switching points and for the conversion of measuring input edges to axis positions. This can affect the accuracy in the event of strong axis acceleration.

The time that elapses between the "state change on the terminal" and the "calculation in the technology object" can be obtained from the timing diagrams.

See also

System timing for single measurement (Page 8270)

System behavior with cyclic measurement (Page 8271)

System behavior with outputs of output cam (Page 8272)

System Behavior of TM15 DI/DO

The exact timing of TM15 DI/DO is described in the Motion Control Function Manual; basic functions are described in chapter "Integration of drive I/O".

12.3.7.3 Message frames

The maximum length of either the setpoint message frame or the actual value message frame is

- With integrated drives (SIMOTION D / CX32/CX32-2): 512 bytes
- With external drives (CU320/CU320-2/CU310/CU310-2):
  - On PROFIBUS DP: 244 bytes
  - On PROFINET IO: For CU320/CU320-2 with CBE20: 400 bytes
  - On PROFINET IO: For CU310-2 PN / CU320-2 PN (integrated PROFINET interface): 480 bytes

However, because the message frames are user-configured, the length of the message frames is determined by the current configuration, i.e. the actual length can be less.

Note that this maximum length applies to the entire PROFIBUS slave or the entire PROFINET device and therefore contains all data for the SINAMICS drive unit, including the:

- Message frame length for all modules of type TM15 / TM17 High Feature
- Message frame length for all axes
- Message frame length for all line modules

If the message frame contains too much data, the number of permissible TMs may be reduced.

The message frame length of a TM1x module can be calculated using the following formulas:

N = number of time-driven output channels (i.e. cam outputs).

M = number of time-driven input channels (i.e. measuring input inputs or edge-triggered enable inputs).

- **TM15**: (N + M ≤ 24)
  Length of setpoint message frame (output address area)
  = 12 + 2 · N (max. = 60 bytes)
  Length of actual value message frame (input address area)
  = 12 + 2 · M (max. = 60 bytes)

- **TM17 High Feature**: (N + M ≤ 16)
  Length of setpoint message frame (output address area)
  = 8 + 4 · N (max. = 72 bytes)
  Length of actual value message frame (input address area)
  = 8 + 4 · M (max. = 72 bytes)

Example

Assume a TM17 High Feature is configured with:

- 4 measuring input inputs
- 6 cam outputs
- 3 digital inputs
- 3 digital outputs

The following applies:

- Length of setpoint message frame (output address area)
  = 8 + 4 · 6 = 32 bytes
- Length of actual value message frame (input address area)
  = 8 + 4 · 4 = 24 bytes

Note

Cam outputs **without high switching accuracy** (i.e. no time-driven output channels are used) count as a digital output.

See also

Maximum permissible message frame length (Page 8242)

12.3.8 Version overview

Function	Available as of
Use of TM15 / TM17 High Feature	
Use of TM15 and TM17 High Feature with SIMOTION D4x5 or on a SINAMICS S120 control unit CU320, which is connected via PROFIBUS DP to SIMOTION C, P or D	As of delivery release (SIMOTION V3.1.1, SINAMICS V2.1)
Use of TM15 and TM17 High Feature with SIMOTION D4x5-2 or on a SINAMICS S120 control unit CU320-2, which is connected via PROFIBUS DP to SIMOTION C, P or D	SIMOTION V4.2 SINAMICS V4.x
TM15 and TM17 High Feature with CX32	SIMOTION V3.2, SP1 SINAMICS V2.3
Use of TM15 and TM17 High Feature with CX32-2	SIMOTION V4.2 SINAMICS V4.x
Use of TM15 or TM17 High Feature on a SINAMICS S120 control unit CU320, which is connected to SIMOTION P or D via PROFINET IO.	SIMOTION V4.1 SINAMICS V2.5
Use of TM15 or TM17 High Feature on a SINAMICS S120 control unit CU320, which is connected to SIMOTION C via PROFINET IO.	SIMOTION V4.1 SP2 HF 3/4 SINAMICS V2.5
Use of TM15 or TM17 High Feature on a SINAMICS S120 control unit CU320-2, which is connected to SIMOTION C, P or D via PROFINET IO.	SIMOTION V4.2 SINAMICS V4.x
Use of TM15 or TM17 High Feature on a SINAMICS S120 control unit CU310, which is connected to SIMOTION via PROFIBUS DP or PROFINET IO.	SIMOTION V4.1, SP1 SINAMICS V2.5
Use of TM15 and TM17 High Feature with SIMOTION D410	SIMOTION V4.1, SP1 SINAMICS V2.5, SP1
Use of TM15, TM17 High Feature with DMC20/ DME20 DRIVE-CLiQ hub module	SIMOTION V4.1 SINAMICS V2.5
TM15 DI/DO	
The module hardware of the TM15 and TM15 DI/DO is identical, but there is a difference between the two modules in terms of system integration. For this reason, they must be configured as different input/ output components. The I/O of the TM15 DI/DO can be assigned drive functions or interconnected to a drive message frame via BICO interconnection technology, which transfers the I/O states between SINAMICS and SIMOTION via PROFIBUS DP or PROFINET IO. You will find more information about integrating the TM15 DI/DO using BICO interconnection in the SINAMICS S120 Commissioning Manual.	
Use of TM15 DI/DO on a SINAMICS S120 control unit CU320/CU320-2	SIMOTION V3.2, SP1 SINAMICS V2.3
Use of TM15 DI/DO on a SINAMICS S120 control unit CU310	SIMOTION V4.0 SINAMICS V2.4
Use of TM15 DI/DO on SIMOTION D4x5 or CX32	SIMOTION V4.0 SINAMICS V2.4
Use of TM15 DI/DO on SIMOTION D4x5-2 or CX32-2	SIMOTION V4.2 SINAMICS V4.x

Function	Available as of
Use of TM15 DI/DO on SIMOTION D410	SIMOTION V4.1, SP1 SINAMICS V2.5, SP1
Use of TM15 DI/DO with DMC20/DME20 DRIVE CLiQ hub module	SIMOTION V4.0 SINAMICS V2.4
Axis types (servo/vector)	
Use of TM15 and TM17 High Feature in conjunction with servo axes	As of delivery release (SIMOTION V3.1.1, SINAMICS V2.1)
Use of TM15 and TM17 High Feature in conjunction with vector axes	SIMOTION V4.1, SP1 SINAMICS V2.5
Use of TM15 DI/DO in conjunction with servo and vector axes	SIMOTION V3.2, SP1 SINAMICS V2.3
Measuring input/output cam functions	
Edge-triggered enable of cam outputs	SIMOTION V3.2 SINAMICS V2.2
Cyclic measurement in conjunction with TM17 High Feature (max. two edges per IPO)	SIMOTION V3.2 SINAMICS V2.2
Cyclic measurement in conjunction with TM17 High Feature (max. two edges per servo)	SIMOTION V4.0
Measuring on virtual axes with TM17 High Feature	SIMOTION V3.2
Measuring on virtual axes with TM15	SIMOTION V3.2, SP1
Several measuring inputs per axis	SIMOTION V3.2
One measuring input for several axes (monitoring measuring input)	SIMOTION V4.0
Measuring range for measuring input with cyclic measurement can be defined (as TO function)	SIMOTION V4.0
Quantity structures	
Total number of all TM15 and TM17 High Feature per SIMOTION D4x5/D4x5-2, CU320/CU320-2 or CX32/CX32-2	Up to SINAMICS V2.2: 2 max.; (0 CX32) up to SINAMICS V2.3: 3 max. SINAMICS V2.4, or later: 8 max. (of which max. 3 per DRIVE-CLiQ line)
Total number of all TM15 DI/DO per SIMOTION D4x5/D4x5-2, CU320/CU320-2 or CX32/CX32-2	8 max.
Total number of all TM15 DI/DO per D410/CU310	3 max.
Total number of all TM15, TM17 High Feature per D410/CU310	3 max.
Maximum permissible number of drive objects per SIMOTION D4x5, SINAMICS CU320, or CX32	Max. 10 drive objects SIMOTION V3.2 and SINAMICS V2.2, or later: Max. 16 drive objects
Maximum permissible number of drive objects per SIMOTION D4x5-2, SINAMICS CU320-2 or CX32-2	SIMOTION V4.2 and SINAMICS V4.x: Max. 24 drive objects
Maximum permissible number of drive objects per D410/CU310	5 max.

Function	Available as of
Maximum permissible message frame length per slave (PROFIBUS)	244 bytes SIMOTION V4.0 and SINAMICS V2.4, or later: 512 bytes with integrated drives of SIMOTION D4x5/D4x5-2, D410 or CX32/CX32-2
Maximum permissible message frame length per DEVICE (PROFINET)	1,380 bytes max.
Miscellaneous	
Servo-synchronous user task (SIMOTION Task System)	SIMOTION V4.0

Note

The SIMOTION version information always refers to the use of the respective SIMOTION SCOUT version, the SIMOTION runtime version as well as the corresponding version of the "CAM" or "CAM_EXT" technology package (including TO measuringInput, TO outputCam and TO camTrack).

The SINAMICS version information always refers to the SINAMICS runtime version including the respective firmware status of the terminal modules as well as the corresponding version of the STARTER.

Older TM15 and TM17 High Feature modules must be upgraded to the new firmware version before the new functions can be used.

See also

Updating the firmware (Page 8244)

12.3.9 Standards and approvals

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus Approval

	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find more information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm. Enter the first 7 characters of the article number under **Keyword**. Then click **Search**.

EMC

KOREA	
이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.	
For sellers or other users, please keep in mind that this device is an A-grade electromagnetic wave device. This device is intended to be used in areas other than home.	
The EMC limit values to be observed for Korea correspond to the limit values of the EMC product standard for variable-speed electric drives EN 61800-3 of Category C2 or the limit value class A, Group 1 to CISPR11. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. For this purpose, additional measures, such as e.g., the use of an additional RFI suppression filter (EMC filter) may be necessary. In addition, measures for EMC-compliant configuration of the plant are described in this Manual and/or the "EMC Installation Guideline" Configuration Manual. Please note that ultimately it is always the label on the device that provides the decisive information on the compliance with standards.	

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity.

See also

Declaration of conformity (http://support.automation.siemens.com/WW/view/en/10805446/134200)

12.3.10 ESD guidelines

12.3.10.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

12.3.10.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

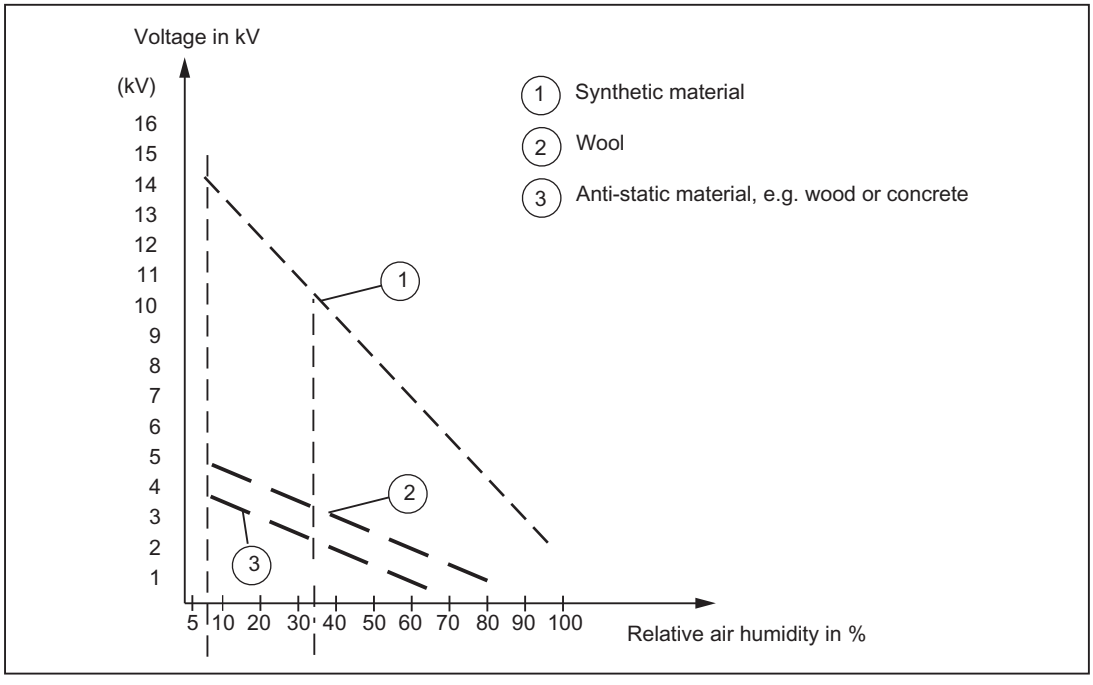


Figure 12-90 Electrostatic voltage that can accumulate on operating personnel

12.3.10.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

12.3.11 List of abbreviations

Refer to this list for an understanding of the abbreviations and acronyms that are used in this Manual:

BiCo	Binector/Connector
CU	Control unit
DI	Digital input
DI/O	Digital input/output
DO	Digital output
I/O	Input/output
FPGA	Field programmable gate array
IPO	Interpolator cycle clock
Lx+	Load power supply
M	Ground
Mx	Reference ground
TM	Terminal Module
TO	Technology object

12.4 TM15 / TM17 High Feature Terminal Modules

Preface

Scope

This manual describes the supplementary TM15 / TM17 High Feature Terminal Modules in operation under SIMOTION D or under SINAMICS S120 in connection with SIMOTION C, P or D.

This manual is aimed at machine manufacturers, plant engineers, commissioning personnel, and service personnel who use SIMOTION in connection with SINAMICS.

Contents of the manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

- System overview
  Provides information about the applications, versions, and integration of the hardware components of the SINAMICS S system in connection with operation under SIMOTION.
- Components
  - Description
    Provides a brief description of each system component and its interfaces.
  - Interfaces
    Provides information about the different interfaces of the devices, their pin assignment, and possible applications.
  - Installation/Mounting
    Provides information about installation and uninstallation of the devices.
  - Electrical connection
    Provides information about the electrical connection of system components.
  - Technical data
    Provides information about the relevant technical data for the device.
- Appendix
  This chapter provides factual information (e.g. standards, approvals, and ESD guidelines).

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.5:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

https://support.industry.siemens.com/My/ww/en/documentation

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

Disposal and recycling of the device

TM15 / TM17 High Feature is an environmentally friendly product. It includes the following features:

- In spite of its excellent resistance to fire, the flame-resistant agent in the plastic used for the housing does not contain halogens.
- Identification of plastic materials in accordance with DIN 54840.
- Less material used because the unit is smaller and with fewer components thanks to integration in ASICs.

The disposal of the products described in this manual should be performed in compliance with the valid national regulations.

The products can be largely recycled owing to their low pollutant content. To recycle and dispose of your old device in an environmentally friendly way, please contact a recycling company certified for electronic waste.

If you have any further questions about disposal and recycling, please contact your local Siemens representative. Contact details can be found in our contacts database on the Internet at:

http://www.automation.siemens.com/partner/index.asp

Further information / FAQs

You can find further information on this manual under the following FAQs:

http://support.automation.siemens.com/WW/view/de/27585482

You can also find additional information under:

- SIMOTION Utilities & Applications: SIMOTION Utilities & Applications will be included in the SIMOTION SCOUT scope of delivery and, along with FAQs, also contain free utilities (e.g. calculation tools, optimization tools, etc.) as well as application examples (ready-to-apply solutions such as winders, cross cutters or handling).
- The latest SIMOTION FAQs at http://support.automation.siemens.com/WW/view/en/10805436/133000
- SIMOTION SCOUT online help
- Refer to the list of references (separate document) for additional documentation

12.4.1 Safety instructions

12.4.1.1 Fundamental safety instructions

General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

⚠ WARNING

Danger to life caused by machine malfunctions caused by incorrect or changed parameterization

Incorrect or changed parameterization can cause malfunctions on machines that can result in injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).

Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity..

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public line supplies requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are enabled.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the controller and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example:
   - Hardware defects and/or software errors in the sensors, controllers, actuators, and connection technology
   - Response times of the controller and drive
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of radio devices / cellular phones in the immediate vicinity of the controller
   - External influences / damage
2. In the event of a fault, exceptionally high temperatures, including an open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the inverter, for example:
   - Component malfunctions
   - Software errors
   - Operating and/or ambient conditions outside of the specification
   - External influences / damage

   Inverters of the Open Type / IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that the contact with fire inside and outside the inverter is not possible.
3. Hazardous touch voltages caused by, for example:
   - Component malfunctions
   - Influence of electrostatic charging
   - Induction of voltages in moving motors
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - External influences / damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc. if they are too close.
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly.

Note

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant sections in the technical user documentation.

12.4.1.2 Specific safety instructions SIMOTION TM15 / TM17 High Feature

⚠ DANGER

Danger to life from energized parts

Death or serious injury will result if energized parts are touched.

Turn off and lock out all power supplying this device before working on this device.

⚠ WARNING

Danger to life from unexpected movement of machines on (automatic) restart

Dangerous mechanical movements may occur in the system during operation.

Make sure this presents no hazard to personnel or property.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Only safety extra-low voltage in accordance with EN/IEC 60950-1 may be connected to all connections and terminals.

12.4.2 Terminal Module TM15

12.4.2.1 Description

The Terminal Module TM15 is a terminal expansion module for snapping on to a DIN EN 60715 mounting rail. The TM15 can be used to increase the number of available digital inputs/outputs within a drive system.

Table 12-19 Interface overview of the TM15

Type	Quantity
Digital inputs/outputs	24 (isolation in 3 groups each with 8 DI/O)

12.4.2.2 Safety Information

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- It is essential that you maintain 50 mm ventilation clearances above and below the component.

12.4.2.3 Description of Ports

Overview

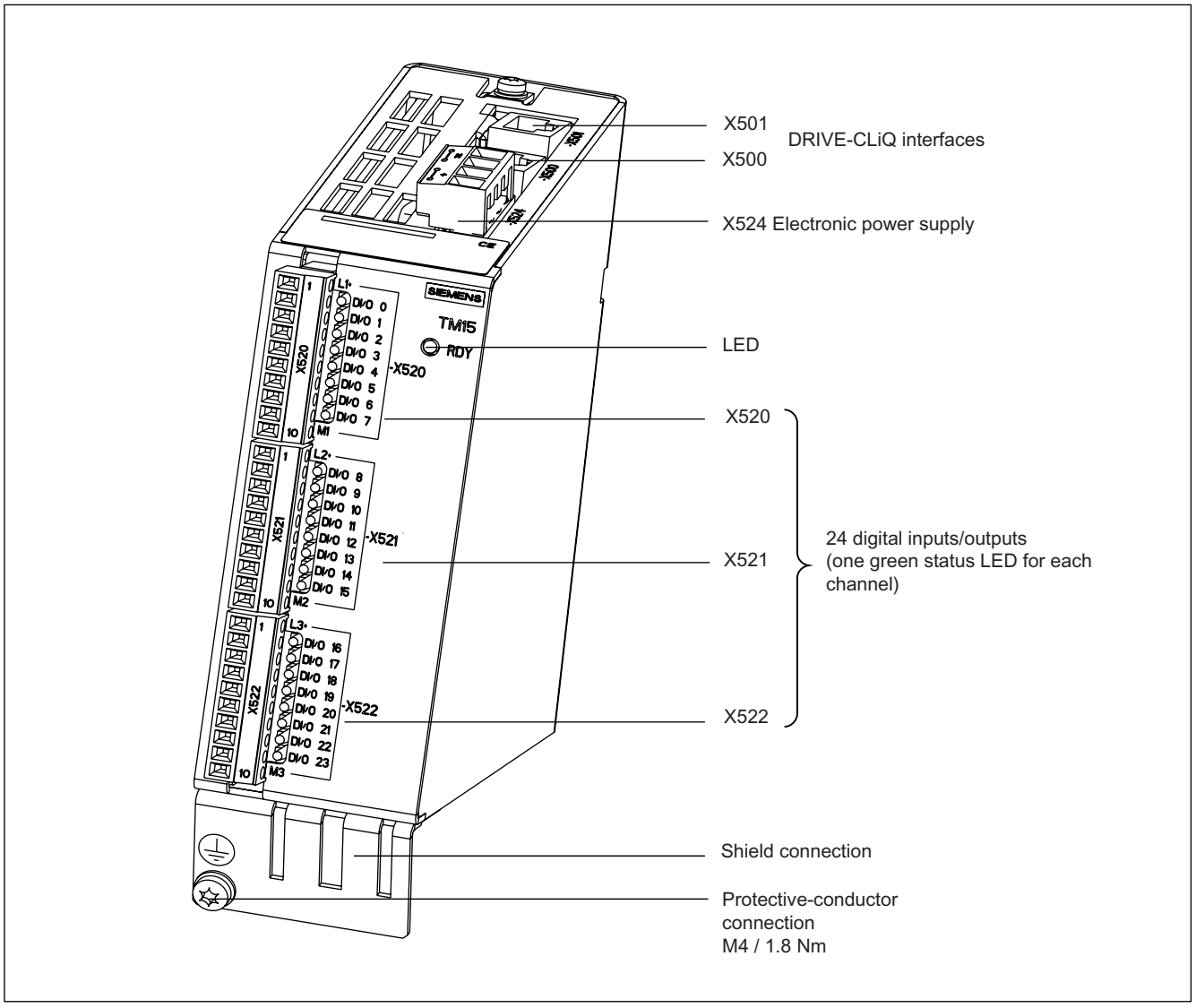


Figure 12-91 Interface description TM15

Connection example

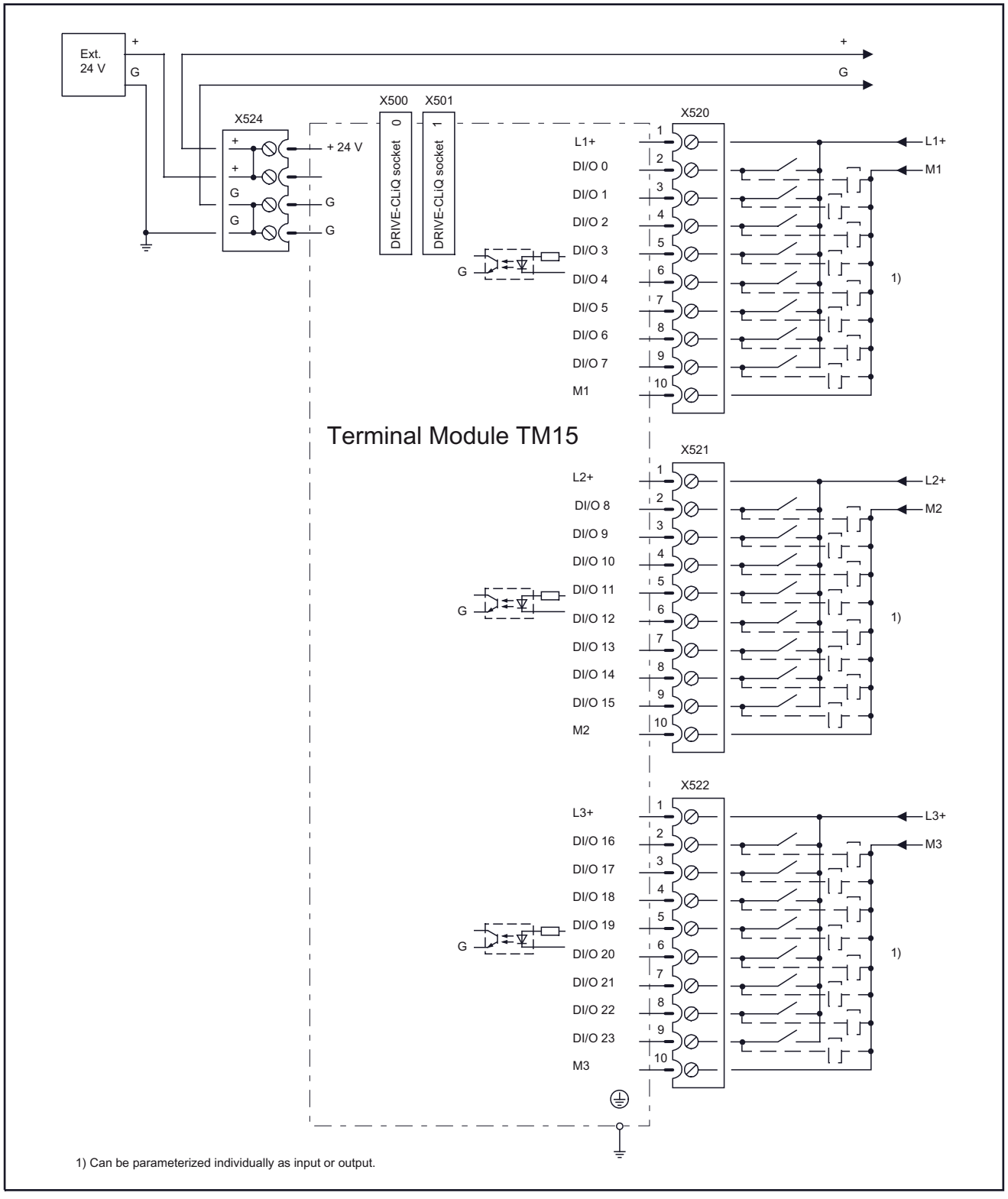
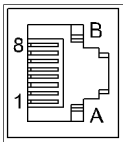


Figure 12-92 Example connection of TM15

X500 and X501 DRIVE-CLiQ interface

Table 12-20 DRIVE-CLiQ interface X500 and X501

	Pin	Signal name	Technical specifications
	1	TXP	Transmit data +
	2	TXN	Transmit data -
	3	RXP	Receive data +
	4	Reserved, do not use	
	5	Reserved, do not use	
	6	RXN	Receive data -
	7	Reserved, do not use	
	8	Reserved, do not use	
	A	+ (24 V)	Power supply
	B	GND (0 V)	Electronic ground
Blanking plate for DRIVE-CLiQ interface: Yamaichi, Article No.: Y-ConAS-13			

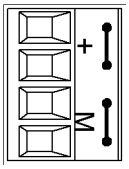
X524 Electronic power supply

NOTICE

The Terminal Modules are reset upon interruption of the supply voltage.

It is essential to ensure that the external 24 VDC power supply to the terminal module is not interrupted for longer than 3 ms. After an interruption of 3 ms, the command to reset the component is issued, causing all outputs to be reset.

Table 12-21 Terminals for the electronic power supply

	Terminal	Designation	Technical specifications
	+	Electronic power supply	Voltage: 24 V DC (20.4 V – 28.8 V) Current consumption: max. 0.15 A Max. current via jumper in connector: 20 A at 60 °C (15 A according to UL/CSA)
	+	Electronic power supply	
	M	Electronic ground	
	M	Electronic ground	
Max. connectable cross-section: 2.5 mm²			

Note

The two "+" and "M" terminals are jumpered in the connector. This ensures that the supply voltage is looped through.

The current consumption increases by the value for the DRIVE-CLiQ node. The digital outputs are supplied via terminals X520, X521, and X522.

Requirements for the power supply

Requirements for the power supply are as follows:

Table 12-22 Requirements for the electronic power supply

Parameter	Requirement
Current	150 mA[1] per module (TM15)

[1] Does not include the current provided for the DI/O or for the DRIVE-CLiQ Interface.

The maximum supply current for TM15 is calculated from the sum of the 3 currents below:

- 150 mA, maximum, via X524 connection
  (module logic must always be taken into account)
- 450 mA, maximum, via X524
  (24 V supply via DRIVE-CLiQ; relevant only if a module connected downcircuit of the TM15 is supplied via DRIVE-CLiQ, e.g. encoder without a separate 24 V connection)
- 24 x 0.5 A, maximum, via X520/X521/X522
  (all channels are parameterized as DO and loaded with 0.5 A)

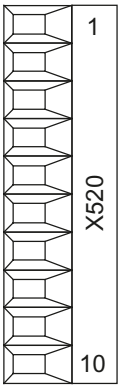
The terminal module monitors the electronic power supply for both overvoltage and undervoltage conditions.

Note

Avoid long cables. The 24 VDC power supply should be located as close as possible to the terminal modules. The total length of all power cables, when added together, must not exceed 10 meters.

X520 digital inputs/outputs

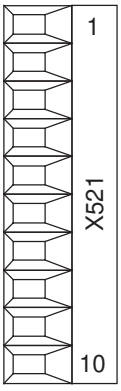
Table 12-23 Screw terminal X520

	Terminal	Designation[1]	Technical specifications
	1	L1+	See chapter "Technical specifications"
	2	DI/O 0	
	3	DI/O 1	
	4	DI/O 2	
	5	DI/O 3	
	6	DI/O 4	
	7	DI/O 5	
	8	DI/O 6	
	9	DI/O 7	
	10	M1 (GND)	
Max. connectable cross-section: 1.5 mm^2			

1) L1+: A 24 V DC power supply for DI/O 0 to 7 (first potential group) must always be connected if at least one DI/O of the potential group is used as output.
M1: A reference ground for DI/O 0 to 7 (first potential group) must always be connected if at least one DI/O of the potential group is used as either input or output.
DI/O: Digital input/output

X521 digital inputs/outputs

Table 12-24 Screw terminal X521

	Terminal	Designation[1]	Technical specifications
	1	L2+	See chapter "Technical specifications"
	2	DI/O 8	
	3	DI/O 9	
	4	DI/O 10	
	5	DI/O 11	
	6	DI/O 12	
	7	DI/O 13	
	8	DI/O 14	
	9	DI/O 15	
	10	M2 (GND)	
Max. connectable cross-section: 1.5 mm^2			

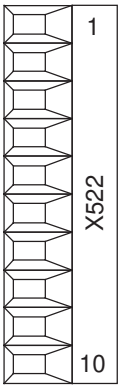
1 L2+: A 24 VDC infeed for DI/O 8 to 15 (second potential group) must always be connected when at least one DI/O of the potential group is used as an output.
M2: A reference ground for DI/O 8 to 15 (second potential group) must always be connected if at least one DI/O of the potential group is used as either input

or output.

DI/O: Digital input/output

X522 digital inputs/outputs

Table 12-25 Screw terminal X522

	Terminal	Designation[1]	Technical specifications
	1	L3+	See chapter "Technical specifications"
	2	DI/O 16	
	3	DI/O 17	
	4	DI/O 18	
	5	DI/O 19	
	6	DI/O 20	
	7	DI/O 21	
	8	DI/O 22	
	9	DI/O 23	
	10	M3 (GND)	
Max. connectable cross-section: 1.5 mm^2			

[1] L3+: A 24 V DC power supply for DI/O 16 to 23 (third potential group) must always be connected if at least one DI/O of the potential group is used as output.
M3: A reference ground for DI/O 16 to 23 (third potential group) must always be connected if at least one DI/O of the potential group is used as either input or output.
DI/O: Digital input/output

Description of the LEDs on the Terminal Module TM15

Table 12-26 Description of the LED

LED	Color	State	Description
READY	-	OFF	Electronics power supply outside permissible tolerance range.
	Green	Continuous	The component is ready for operation and cyclic DRIVE-CLiQ communication is taking place.
	Orange	Continuous	DRIVE-CLiQ communication is being established.
	Red	Continuous	At least one fault is present in this component.
	Green/red	Flashing 2 Hz	Firmware is being downloaded.
	Green/Orange	Flashing 2 Hz	Component detected: no fault present
	Red/Orange	Flashing 2 Hz	Component detected: Fault(s) present

Cause and rectification of faults

The following reference contains information about the cause of faults and how they can be rectified:

- SINAMICS S Commissioning Manual
- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- TM15 / TM17 High Feature Commissioning Manual

12.4.2.4 Dimension Drawing

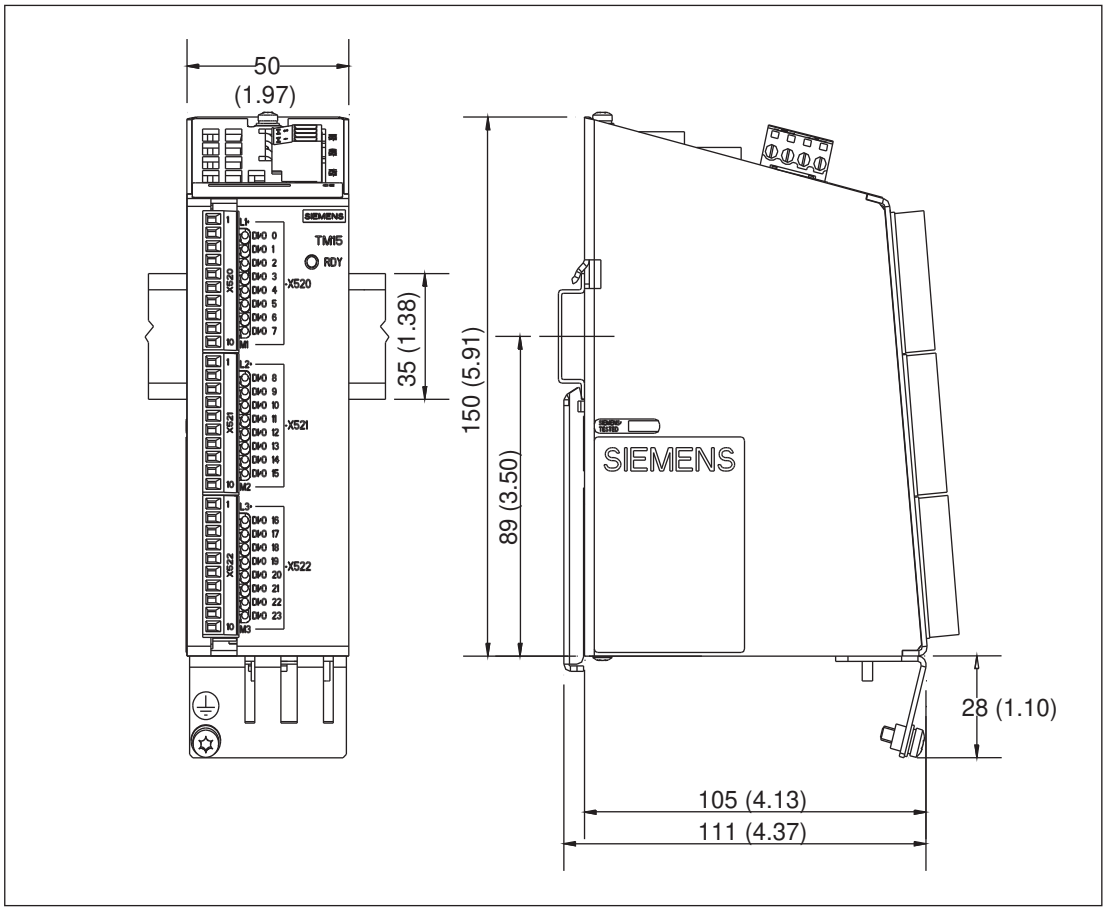


Figure 12-93 Dimension drawing of the TM15

12.4.2.5 Installation

Installation

1. Place the component on the DIN rail.
2. Snap the component on to the DIN rail. Make sure that the mounting slides at the rear latch into place.
3. You can now move the component on the DIN rail to the left or to the right to its final position.

Disassembly

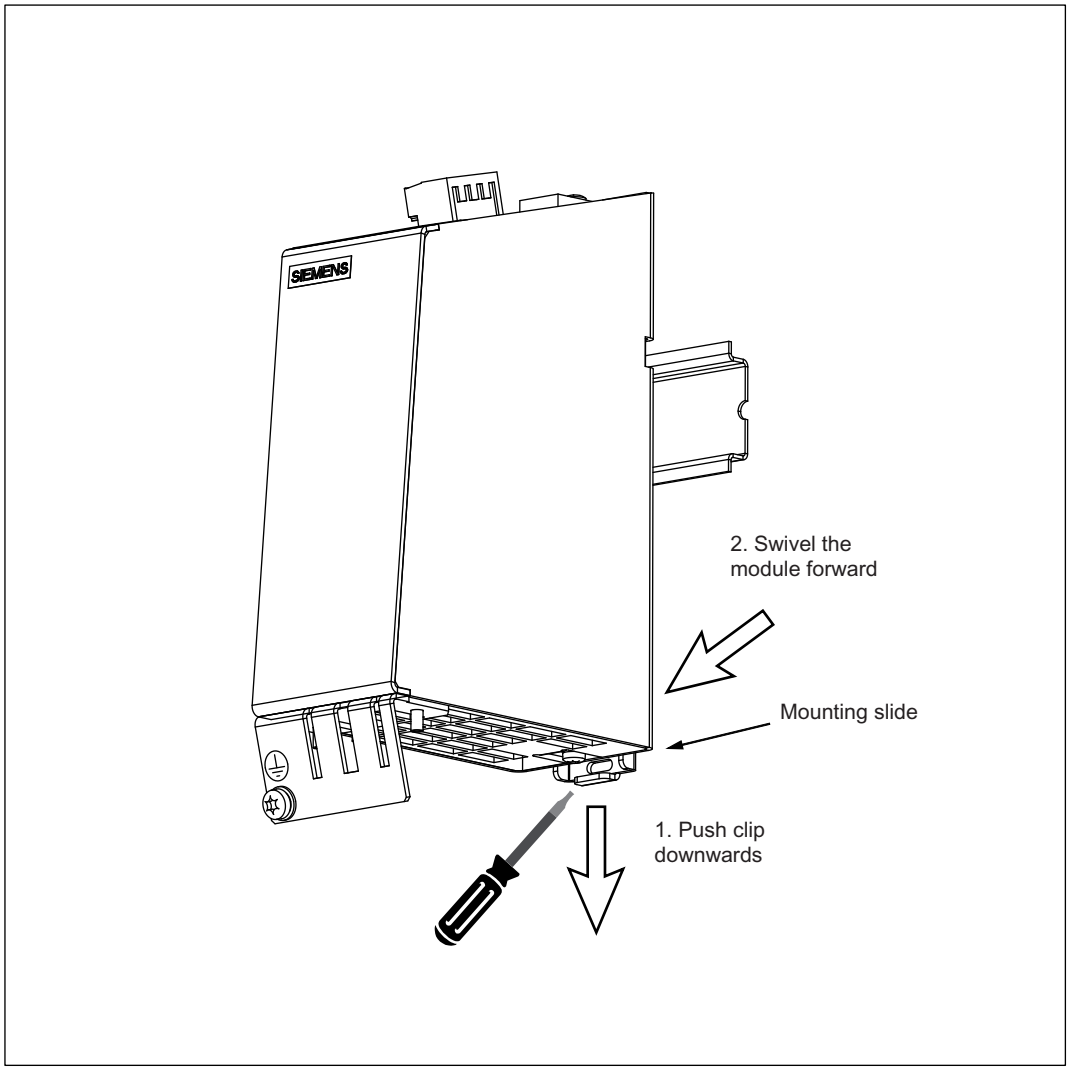


Figure 12-94 Releasing the component from a DIN rail

12.4.2.6 Electrical Connection

It is always advisable to shield the digital input/output wiring.

The following pictures show two typical shield connections from Weidmüller.

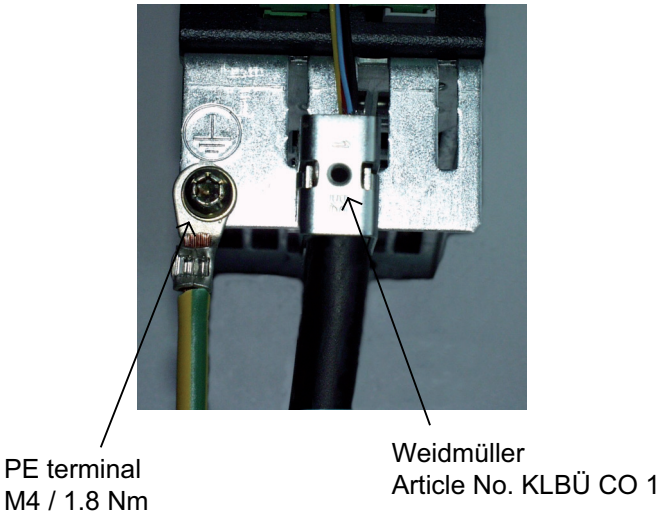


Figure 12-95 Shield connections

Internet address of the company:

Weidmüller: http://www.weidmueller.com

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

The TM15 housing is connected to the ground terminal of the module supply (terminal X524). As long as the chassis is grounded, the housing is also grounded. An additional ground connection using the M4 screw is especially necessary if high potential bonding currents can flow (e.g. through the cable shield).

Connector coding

Siemens supplies a series of coding keys (coding sliders) with each Terminal Module TM15. To code a connector, you must insert at least one coding slider and cut off at least one coding projection on the connector:

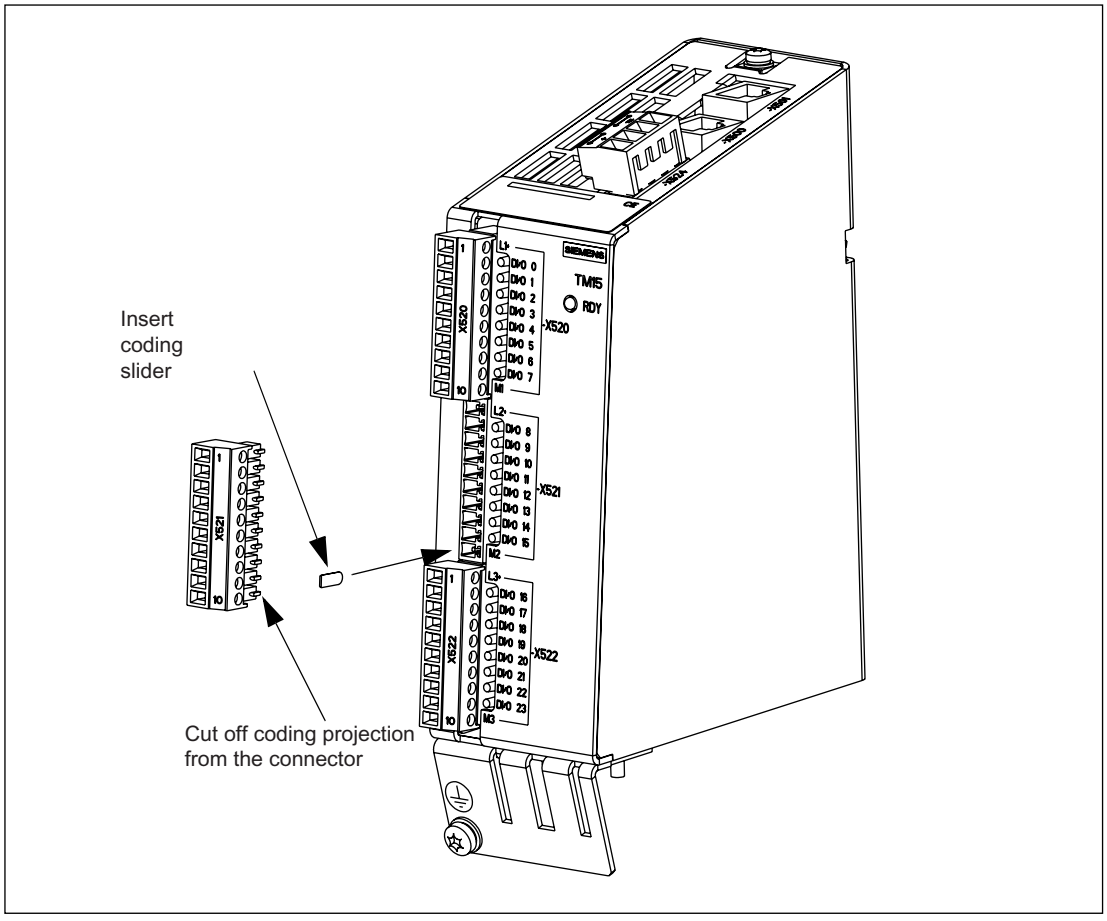


Figure 12-96 Procedure for encoding a connector

To avoid incorrect wiring, unique connector coding schemes for the I/O connectors X520, X521, and X522 may be defined. Examples of possible patterns:

- Different coding between the 3 plugs of a component (i.e., X520, X521, and X522).
- Different component types are encoded differently.
- Identical components on the same machine are encoded differently (e.g. several TM15-type components).

12.4.2.7 Commissioning

Note

For information about commissioning , see the *SIMOTION Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

12.4.2.8 Technical data

Table 12-27 Technical data

Terminal Module TM15 **6SL3055-0AA00-3FAx**	**Unit**	**Value**
Electronic power supply Voltage Current (without DRIVE-CLiQ or digital outputs) Power loss	 V_{DC} A_{DC} W	 24 DC (20.4 – 28.8) 0.15 <3
Ambient temperature up to an altitude of 2000 m	°C	0 - 60
Storage temperature	°C	-40 to +85
Relative humidity	5% to 95%, no condensation	
I/O		
• Digital inputs/outputs	Each can be parameterized separately as DI or DO	
• Number of digital inputs/outputs	24	
• Isolation	Yes, in groups of 8	
• Max. cable length	m	30
Digital inputs		
• Voltage	V_{DC}	-30 to +30
• Low-level (an open digital input is interpreted as "low")	V_{DC}	-30 to +5
• High level	V_{DC}	15 to 30
• Input impedance	kΩ	2.8
• Current consumption (at 24 V DC), typical	mA	9
• Max. voltage in OFF state	V_{DC}	5
• Current in OFF state	mA	0.0 to 1.0 (per channel)
• Input delay of digital inputs, typical [1)]	µs	For "0" to "1" 50 For "1" to "0" 100
Digital outputs (sustained short-circuit-proof)		
• Voltage	V_{DC}	24
• Max. load current per digital output	A_{DC}	0.5
• Output delay (ohmic load)		
• typical	µs	For "0" to "1" 50 For "1" to "0" 150
• maximum	µs	For "0" to "1" 100 For "1" to "0" 225

Terminal Module TM15 **6SL3055-0AA00-3FAx**	**Unit**	**Value**
• Min. output pulse (100% amplitude, 0.5 A with resistive load)	µs	125 (typ.) 350 (max.)
• Max. switching frequency (100% amplitude, 50%/50% duty cycle, with 0.5 A and a resistive load)	kHz	1 (typ.)
• Voltage drop in ON state	V_{DC}	0.75 (max.) with all circuits fully loaded
• Leakage current in OFF state	µA	max. 10 per channel
• Output voltage drop (I/O power supply to the output)	V_{DC}	0.5
• Max. total current of the outputs (per group) to 60 °C to 50 °C to 40 °C	 A_{DC} A_{DC} A_{DC}	 2 3 4
IEC enclosure specification	IP20 degree of protection	
Protective-conductor connection	On the housing with M4/1.8 Nm screw	
Response time	The response time for the digital inputs/outputs (TM15 DI/DO) consists of the following elements: • Response time on the component itself (approx. 1/2 DRIVE-CLiQ cycle). • Transmission time via the DRIVE-CLiQ connection (approx. 1 DRIVE-CLiQ cycle). • Evaluation on the control unit (see function diagram) **References:** SINAMICS S List Manual, chapter "Function diagrams".	
Weight	kg	0.86
Approval	UL and cULus http://www.ul.com File: E164110, Vol. 2, Sec. 9	

1) Pure hardware delay

12.4.3 Terminal Module TM17 High Feature

12.4.3.1 Description

The Terminal Module TM17 High Feature is a terminal expansion module for snapping on to a DIN EN 60715 mounting rail. The TM17 High Feature can be used to increase the number of available digital inputs/outputs within a drive system.

Table 12-28 Interface overview of the TM17 High Feature

Type	**Quantity**
Digital inputs/outputs	16 (non-isolated, 2 voltage groups, each with 8 DI/O)

12.4.3.2 Safety Information

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- It is essential that you maintain 50 mm ventilation clearances above and below the component.

12.4.3.3 Description of Ports

Overview

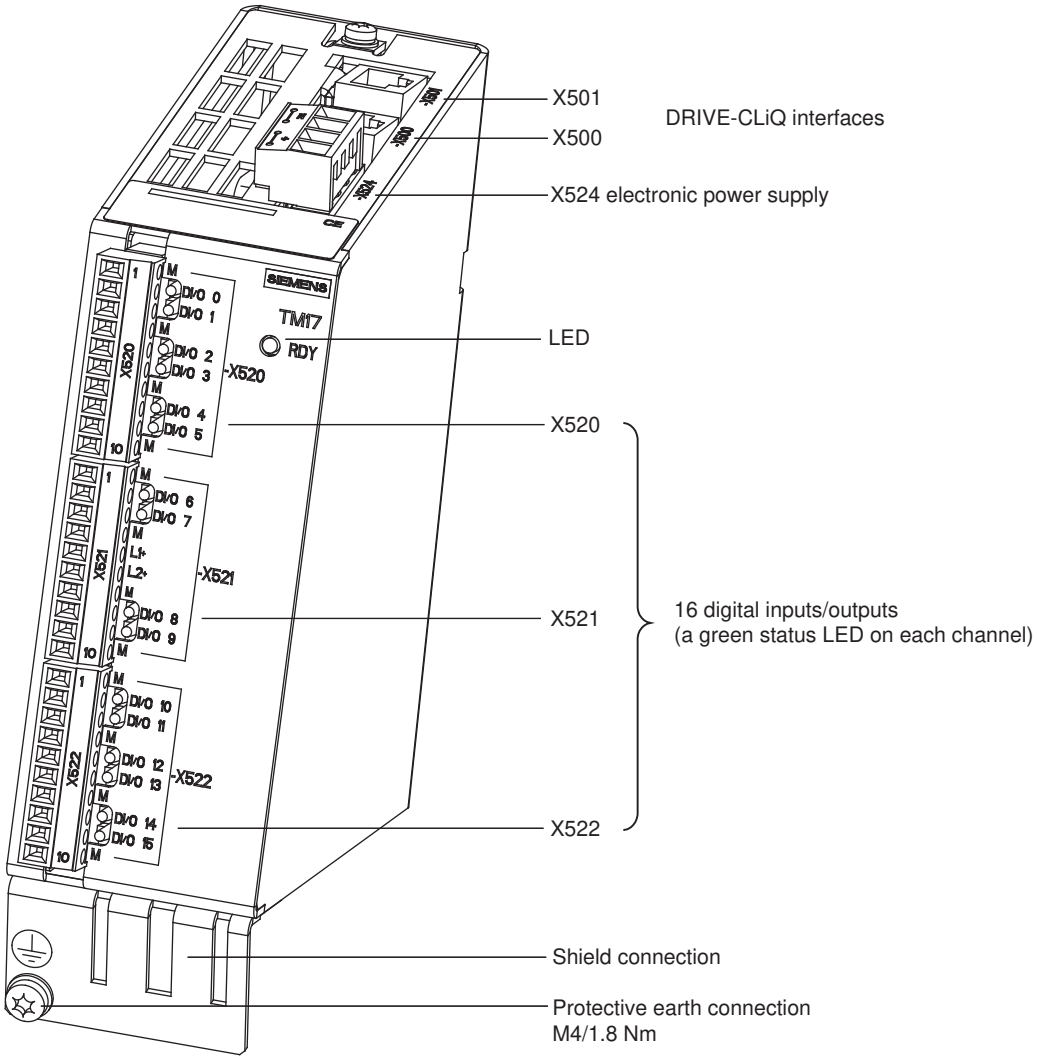
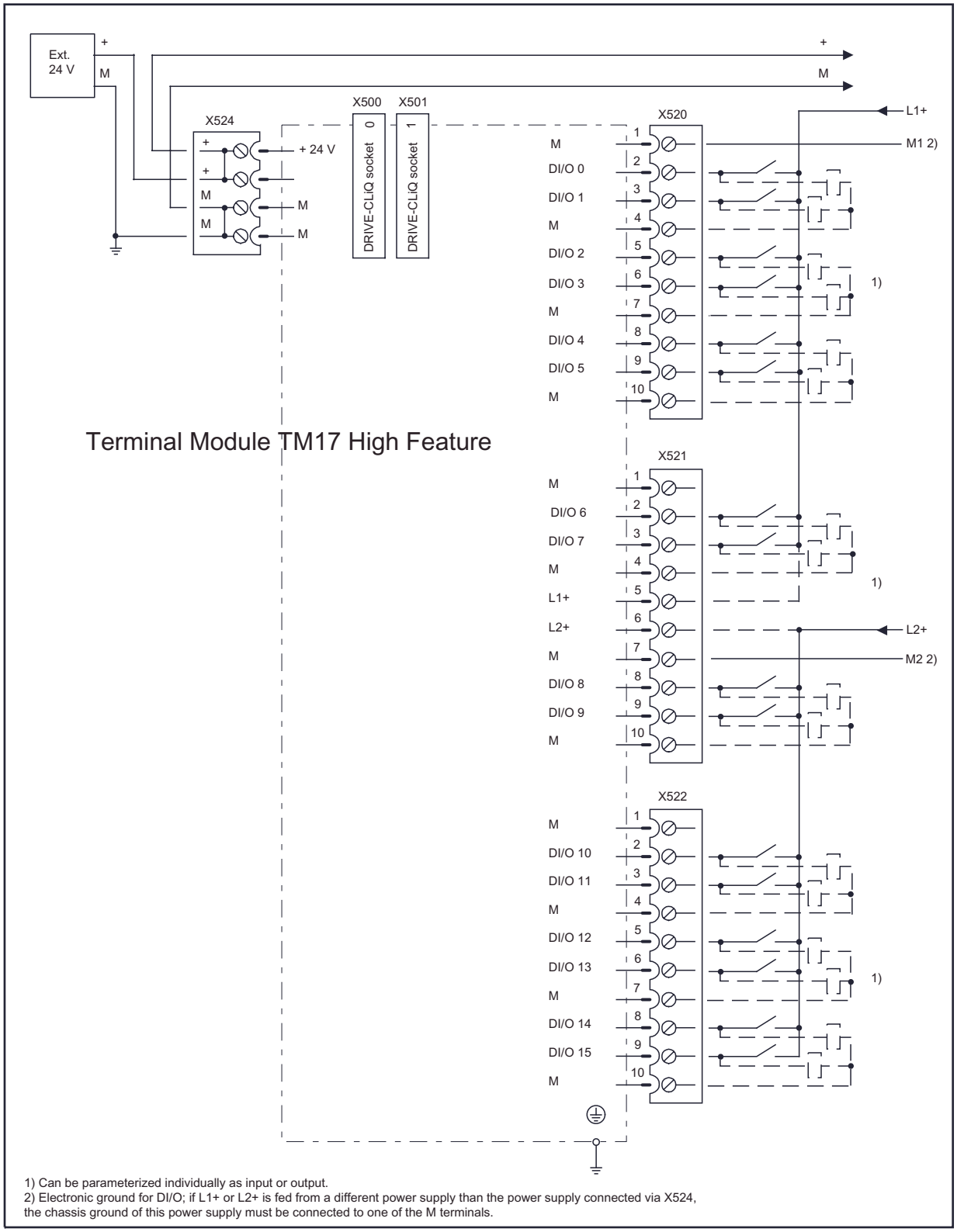


Figure 12-97 TM17 High Feature interface description

Sample connection

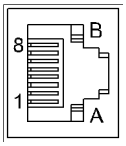


1) Can be parameterized individually as input or output.
2) Electronic ground for DI/O; if L1+ or L2+ is fed from a different power supply than the power supply connected via X524, the chassis ground of this power supply must be connected to one of the M terminals.

Figure 12-98 TM17 High Feature connection example

X500 and X501 DRIVE-CLiQ interface

Table 12-29 DRIVE-CLiQ interface X500 and X501

	Pin	Signal name	Technical specifications
	1	TXP	Transmit data +
	2	TXN	Transmit data -
	3	RXP	Receive data +
	4	Reserved, do not use	
	5	Reserved, do not use	
	6	RXN	Receive data -
	7	Reserved, do not use	
	8	Reserved, do not use	
	A	+ (24 V)	Power supply
	B	GND (0 V)	Electronic ground
Blanking plate for DRIVE-CLiQ interface: Yamaichi, Article No.: Y-ConAS-13			

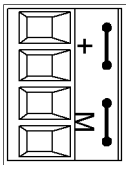
X524 Electronic power supply

NOTICE

The Terminal Modules are reset upon interruption of the supply voltage.

It is essential to ensure that the external 24 VDC power supply to the terminal module is not interrupted for longer than 3 ms. After an interruption of 3 ms, the command to reset the component is issued, causing all outputs to be reset.

Table 12-30 Terminals for the electronic power supply

	Terminal	Designation	Technical specifications
	+	Electronic power supply	Voltage: 24 VDC (20.4 V - 28.8 V) Current consumption: max. 0.2 A Max. current via jumper in connector: 20 A at 60 °C (15 A according to UL/CSA)
	+	Electronic power supply	
	M	Electronic ground	
	M	Electronic ground	
Max. connectable cross-section: 2.5 mm²			

Note

The two "+" and "M" terminals are jumpered in the connector and not in the device. This ensures that the supply voltage is looped through.

The current consumption increases by the value for the DRIVE-CLiQ node. The digital outputs are supplied via terminals X520, X521, and X522.

Requirements for the power supply

Requirements for the power supply are as follows:

Table 12-31 Requirements for the electronic power supply

Parameter	Requirement
Current	200 mA[1] per module (TM17 High Feature)

[1] Does not include the current provided for the DI/O or for the DRIVE-CLiQ Interface.

The maximum supply current for TM17 High Feature is calculated from the sum of the 3 currents below:

- 200 mA, maximum, via X524 connection
  (module logic must always be taken into account)
- 450 mA, maximum, via X524
  (24 V supply via DRIVE-CLiQ; relevant only if a module connected downcircuit of the TM17 High Feature is supplied via DRIVE-CLiQ, e.g. encoder without a separate 24 V connection)
- 16 x 0.5 A, maximum, via X520/X521/X522
  (all channels are parameterized as DO and charged with 0.5 A)

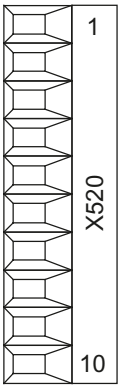
The terminal module monitors the electronic power supply for both overvoltage and undervoltage conditions.

Note

Avoid long cables. The 24 VDC power supply should be located as close as possible to the terminal modules. The total length of all power cables, when added together, must not exceed 10 meters.

X520 digital inputs/outputs

Table 12-32 Screw terminal X520

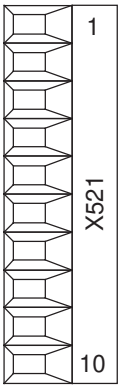
	Terminal	Designation[1]	Technical specifications
	1	M (GND)	See chapter "Technical specifications"
	2	DI/O 0	
	3	DI/O 1	
	4	M (GND)	
	5	DI/O 2	
	6	DI/O 3	
	7	M (GND)	
	8	DI/O 4	
	9	DI/O 5	
	10	M (GND)	
Max. connectable cross-section: 1.5 mm^2			

[1] M: Electronic ground for DI/O 0 to 15; if L1+ or L2+ is fed from a power supply other than the power supply connected via X524, the chassis ground of this power supply (L1+ or L2+) must be connected to one of the M-terminals.

DI/O: Digital input/output

X521 digital inputs/outputs

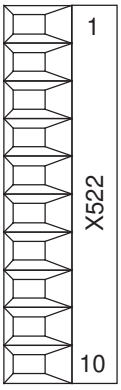
Table 12-33 Screw terminal X521

	Terminal	Designation[1]	Technical specifications
	1	M (GND)	See chapter "Technical specifications"
	2	DI/O 6	
	3	DI/O 7	
	4	M (GND)	
	5	L1+	
	6	L2+	
	7	M (GND)	
	8	DI/O 8	
	9	DI/O 9	
	10	M (GND)	
Max. connectable cross-section: 1.5 mm^2			

[1] L1+: A 24 VDC infeed for DI/O 0 to 7 (first voltage group) must always be connected when at least one DI/O of the voltage group is used as an output.
L2+: A 24 VDC infeed for DI/O 8 to 15 (second potential group) must always be connected when at least one DI/O of the potential group is used as an output.
M: Electronic ground for DI/O 0 to 15; if L1+ or L2+ is fed from a power supply other than the power supply connected via X524, the chassis ground of this power supply (L1+ or L2+) must be connected to one of the M-terminals.
DI/O: Digital input/output

X522 digital inputs/outputs

Table 12-34 Screw terminal X522

	Terminal	Designation[1]	Technical specifications
	1	M (GND)	See chapter "Technical specifications"
	2	DI/O 10	
	3	DI/O 11	
	4	M (GND)	
	5	DI/O 12	
	6	DI/O 13	
	7	M (GND)	
	8	DI/O 14	
	9	DI/O 15	
	10	M (GND)	
Max. connectable cross-section: 1.5 mm^2			

[1] M: Electronic ground for DI/O 0 to 15; if L1+ or L2+ is fed from a power supply other than the power supply connected via X524, the chassis ground of this power supply (L1+ or L2+) must be connected to one of the M-terminals.

DI/O: Digital input/output

Description of the LEDs on Terminal Module TM17 High Feature

Table 12-35 Description of the LED

LED	Color	State	Description
READY	-	OFF	Electronics power supply outside permissible tolerance range.
	Green	Continuous	The component is ready for operation and cyclic DRIVE-CLiQ communication is taking place.
	Orange	Continuous	DRIVE-CLiQ communication is being established.
	Red	Continuous	At least one fault is present in this component.
	Green/red	Flashing 2 Hz	Firmware is being downloaded.
	Green/Orange	Flashing 2 Hz	Component detected: no fault present
	Red/Orange	Flashing 2 Hz	Component detected: Fault(s) present

Cause and rectification of faults

The following reference contains information about the cause of faults and how they can be rectified:

- SIMOTION D4x5 Commissioning and Hardware Installation Manual
- TM15 / TM17 High Feature Commissioning Manual

12.4.3.4 Dimension drawing

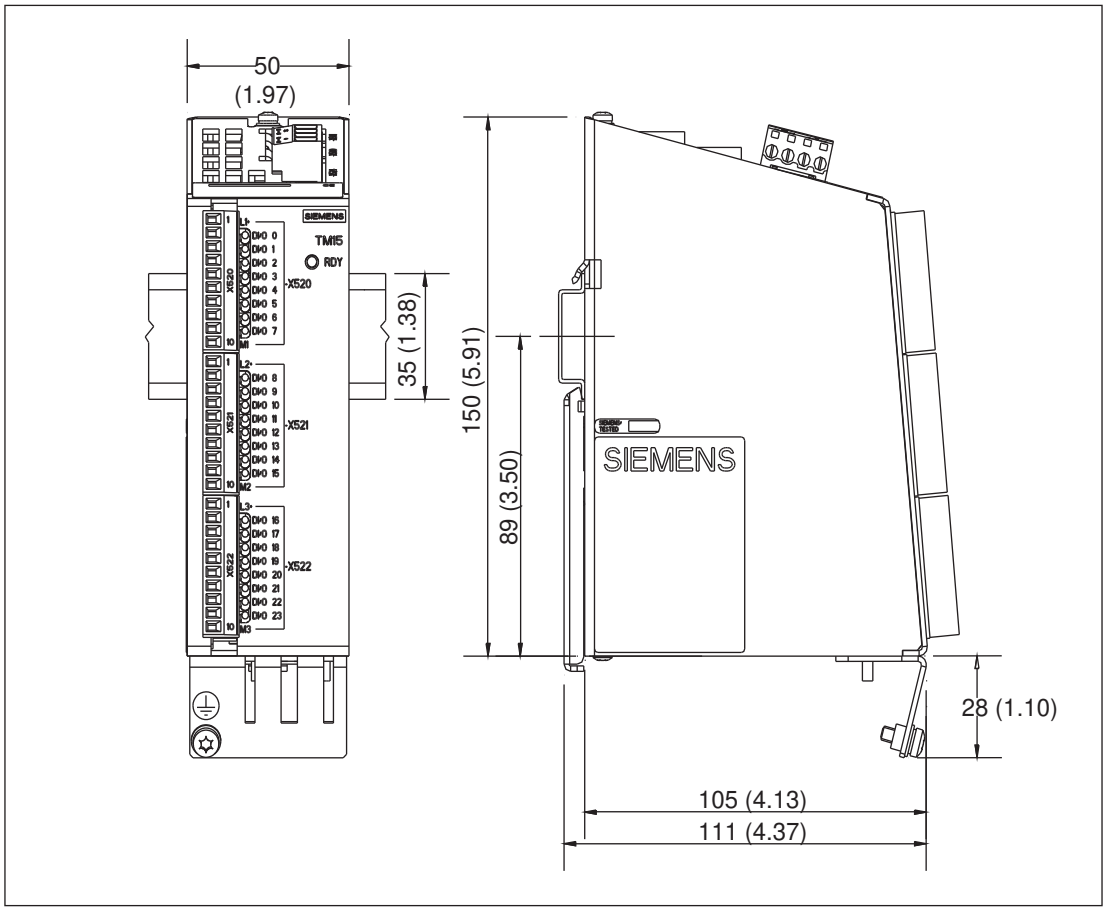


Figure 12-99 Dimension drawing of TM17 High Feature (like TM15)

12.4.3.5 Installation

Installation

1. Place the component on the DIN rail.
2. Snap the component on to the DIN rail. Make sure that the mounting slides at the rear latch into place.
3. You can now move the component on the DIN rail to the left or to the right to its final position.

Disassembly

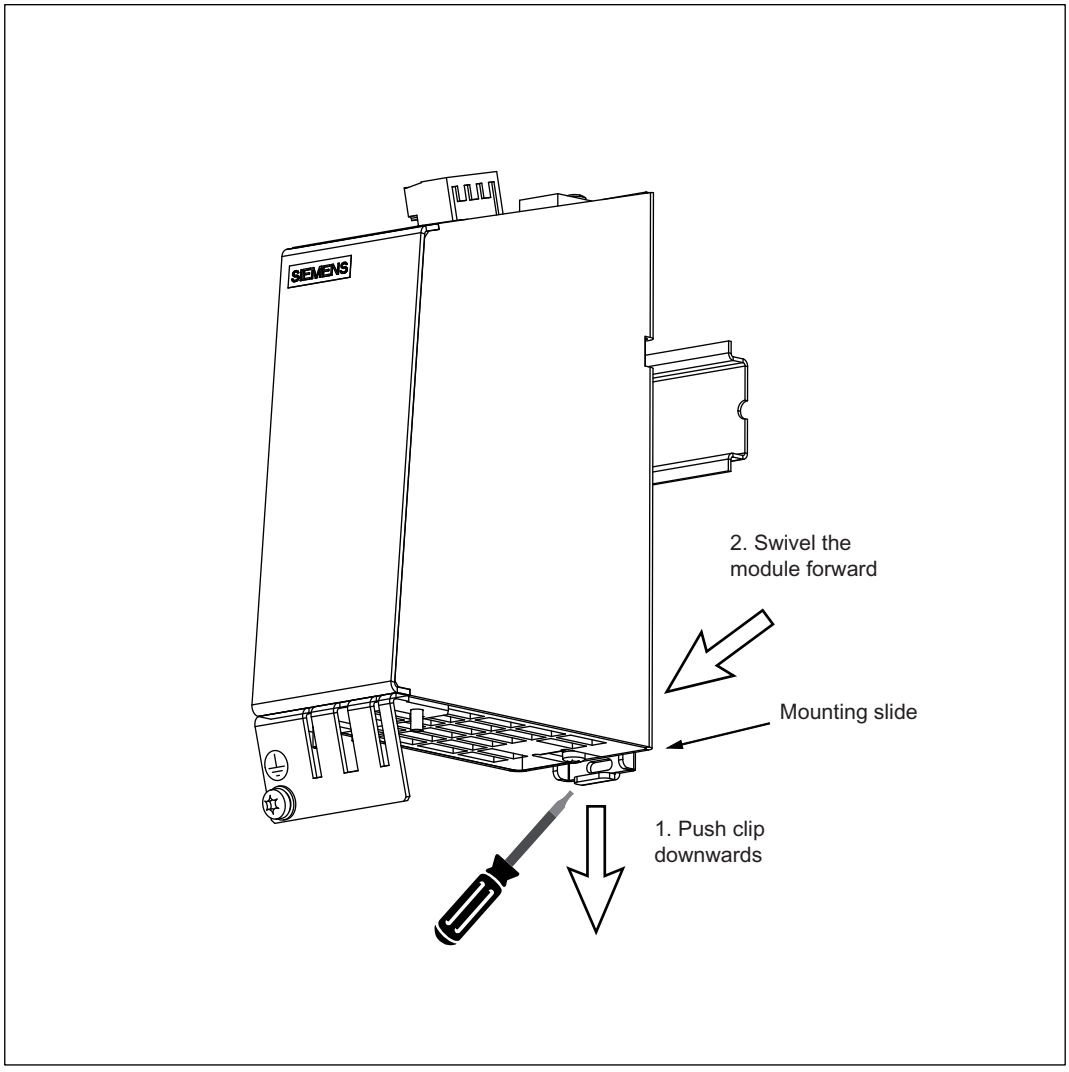


Figure 12-100 Releasing the component from a DIN rail

12.4.3.6 Electrical Connection

It is always advisable to shield the digital input/output wiring.

The following pictures show two typical shield connections from Weidmüller.

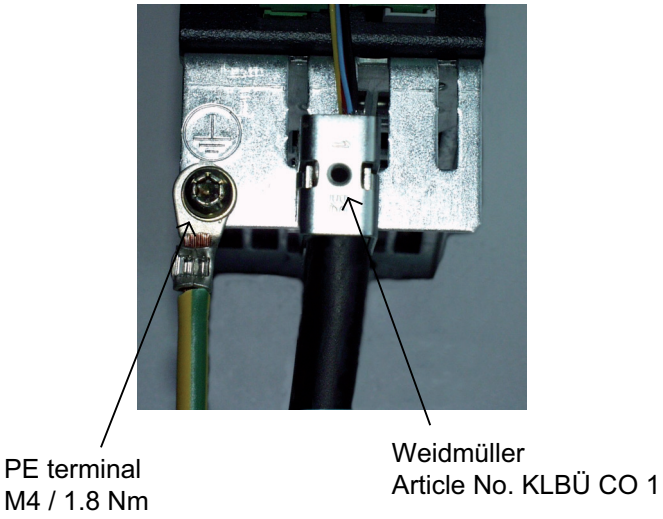


Figure 12-101 Shield connections

Company Internet addresses:

Weidmüller: http://www.weidmueller.com

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

The casing of the TM17 High Feature is connected to the chassis terminal of the module power supply (terminal X524). As long as the chassis is grounded, the housing is also grounded. Additional grounding via the M4 screw is required, in particular, when large equipotential bonding currents can flow (e.g. via the cable shield or the non-isolated I/O of the TM17 High Feature).

Connector coding

Siemens supplies a series of coding elements (coding sliders) with each Terminal Module TM17 High Feature. To code a connector, you must insert at least one coding slider and cut off at least one coding projection on the connector:

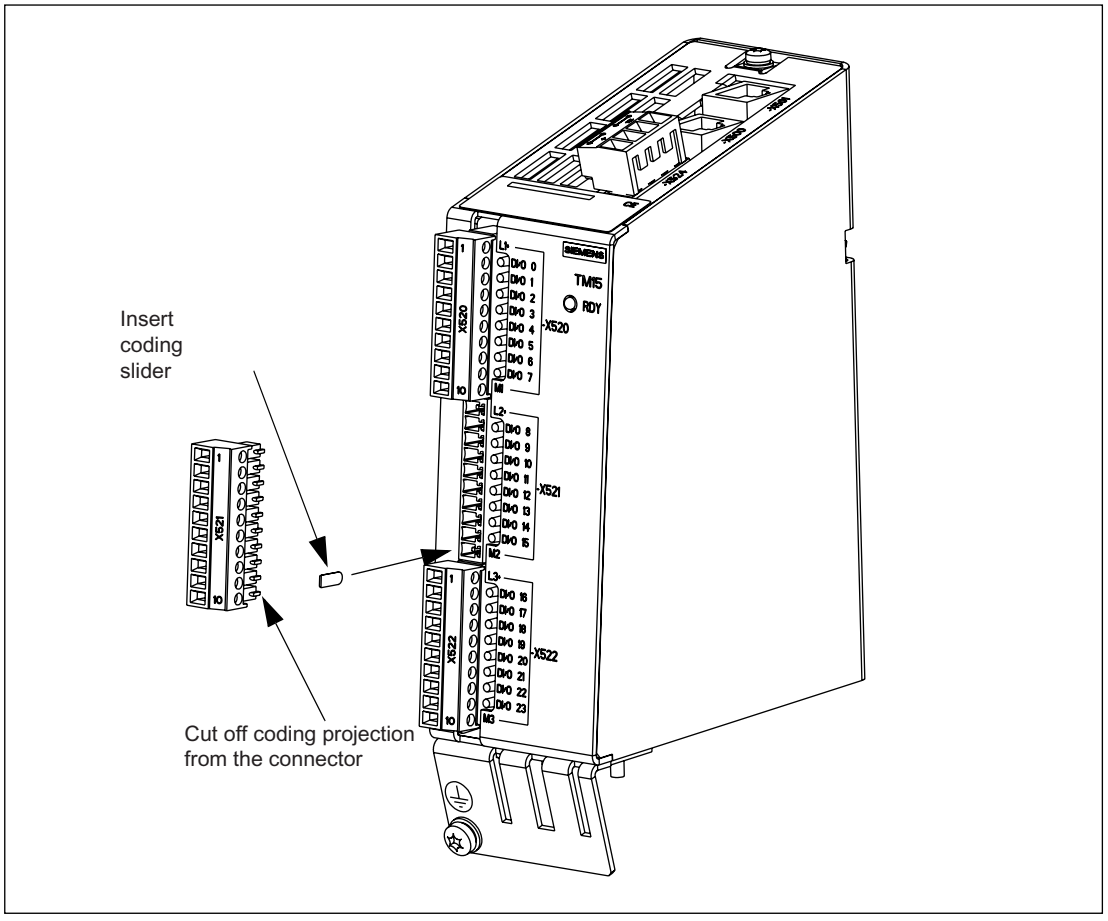


Figure 12-102 Connector coding - procedure (same as for TM 15)

To avoid incorrect wiring, unique connector coding schemes for the I/O connectors X520, X521, and X522 may be defined. Examples of possible patterns:

- Different coding between the 3 plugs of a component (i.e., X520, X521, and X522).
- Different components are coded differently.
- Identical components on the same machine are coded differently (e.g. several TM17 High Feature components).

12.4.3.7 Commissioning

Note

For information about commissioning , see the *SIMOTION Terminal Modules TM15 / TM17 High Feature* Commissioning Manual.

12.4.3.8 Technical data

Table 12-36 Technical data

Terminal Module TM17 High Feature	Unit	Value
Electronic power supply Voltage Current (without DRIVE-CLiQ or digital outputs) Power loss	 V_{DC} A_{DC} W	 24 VDC (20.4 – 28.8) 0.2 <4
Ambient temperature up to an altitude of 2000 m	°C	0 - 60
Storage temperature	°C	-40 to +85
Relative humidity	5% to 95%, no condensation	
I/O		
• Digital inputs/outputs	Each can be parameterized separately as DI or DO	
• Number of digital inputs/outputs	16	
• Isolation	No, 2 voltage groups	
• Max. cable length	m	30
Digital inputs		
• Voltage	V_{DC}	-30 to +30
• Low-level (an open digital input is interpreted as "low")	V_{DC}	-30 to +5
• High level	V_{DC}	15 to 30
• Input impedance	kΩ	2.8
• Current consumption (at 24 V DC), typical	mA	9
• Max. voltage in OFF state	V_{DC}	5
• Current in OFF state	mA	-11.0 to 2.0 (per channel)
• Input delay of digital inputs, typical	µs	(hardware filter that can be selected via the software) For "0" to "1" 1 or 125 ±15% [1] For "1" to "0" 1 or 125 ±15% [1]
• LEDs (per channel)	1 green at logic side	
Digital outputs (sustained short-circuit-proof)		
• Voltage	V_{DC}	24
• Max. load current per digital output	A_{DC}	0.5
• Output delay (ohmic load)		
typical	µs	For "0" to "1" 50 For "1" to "0" 75

Terminal Module TM17 High Feature	Unit	Value
maximum	µs	For "0" to "1" 100 For "1" to "0" 150
• Min. output pulse (100% amplitude, 0.5 A with resistive load)	µs	75 (typ.) 150 (max.)
• Max. switching frequency (100% amplitude, 50%/50% duty cycle, with 0.5 A and a resistive load)	kHz	1 (typ.)
• Voltage drop in ON state	V_{DC}	0.75 (max.) with all circuits fully loaded
• Leakage current in OFF state	µA	max. 10 per channel
• Output voltage drop (I/O power supply to the output)	V_{DC}	0.5
• Max. total current of the outputs (per group) to 60 °C to 50 °C to 40 °C	 A_{DC} A_{DC} A_{DC}	 2 3 4
IEC enclosure specification	IP20 degree of protection	
Protective-conductor connection	On the housing with M4/1.8 Nm screw	
Weight	kg	0.86
Approval	UL and cULus http://www.ul.com File: E164110, Vol. 2, Sec. 9	

[1] The shortest pulses can be detected using the 1 µs filter; however, the 125 µs filter will provide higher noise immunity.

12.4.4 Standards, Certificates and Approvals

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus approval

	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find further information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm Enter the first seven characters of the article number at **Keyword**. Then click **Search**.

Korea certification

	KC registration number: KCC-REM-S49-SIMOTION Note that this device complies with limit class A with regard to the emission of radio frequency interference. This device can be used in all areas except residential areas. 이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity.

See also

Declaration of conformity (http://support.automation.siemens.com/WW/view/en/10805446/134200)

12.4.5 ESD guidelines

12.4.5.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

12.4.5.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

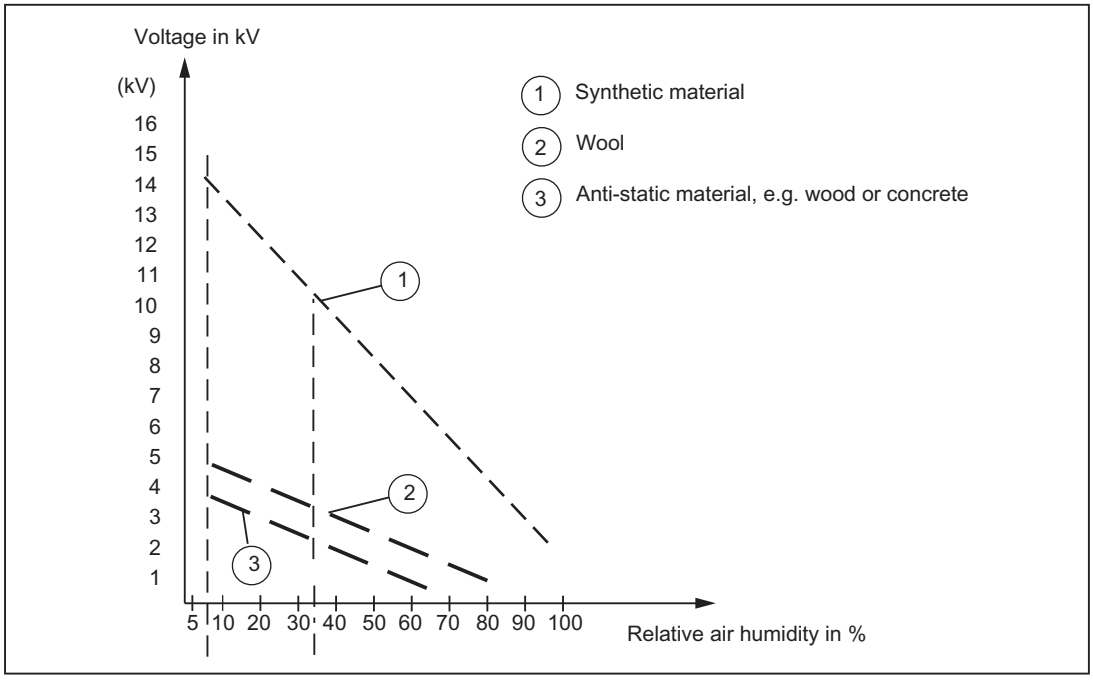


Figure 12-103 Electrostatic voltage that can accumulate on operating personnel

12.4.5.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

12.5 SIMOTION ADI4 - Analog Drive Interface for 4 Axes

Preface

Purpose of the manual

This manual describes the functionality and use of the ADI4 - Analog Drive Interface for 4 Axes.

Note

This manual provides information about the hardware and applications of the ADI4. In the SIMOTION manuals, you will find general information about working with the SIMOTION SCOUT engineering software, technology objects, etc.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.

This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics

- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in **Product Support**:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under **Contact**:

http://www.siemens.com/automation/service&support

Scope

ADI4 is supported by the following Siemens products:

- SCOUT Engineering System, Version 3.1.1 or later
- SIMOTION Runtime, Version 3.1.1 or later
- SINAMICS V2.1 or later

Organization of information

The information about ADI4 in this manual is organized as follows:

- General
  This chapter describes the properties and essential features as well as the higher-level boundary conditions of the ADI4.
- Hardware description
  This chapter describes the interfaces of the ADI4, the control cabinet installation, and the power supply of the module.
- Parameter assignment
  This chapter describes the assignment of ADI4 parameters for PROFIBUS DP and for functions.
- Commissioning
  This chapter presents a commissioning example using absolute and incremental encoders in conjunction with ADI4.
- Index
  The index helps you to locate information in the manual quickly and easily.

12.5.1 Fundamental safety instructions

12.5.1.1 General safety instructions

⚠ DANGER

Danger to life due to live parts and other energy sources

Death or serious injury can result when live parts are touched.

- Only work on electrical devices when you are qualified for this job.
- Always observe the country-specific safety rules.

Generally, six steps apply when establishing safety:

1. Prepare for shutdown and notify all those who will be affected by the procedure.
2. Disconnect the machine from the supply.
   - Switch off the machine.
   - Wait until the discharge time specified on the warning labels has elapsed.
   - Check that it really is in a no-voltage condition, from phase conductor to phase conductor and phase conductor to protective conductor.
   - Check whether the existing auxiliary supply circuits are de-energized.
   - Ensure that the motors cannot move.
3. Identify all other dangerous energy sources, e.g. compressed air, hydraulic systems, or water.
4. Isolate or neutralize all hazardous energy sources by closing switches, grounding or short-circuiting or closing valves, for example.
5. Secure the energy sources against switching on again.
6. Ensure that the correct machine is completely interlocked.

After you have completed the work, restore the operational readiness in the inverse sequence.

⚠ WARNING

Danger to life from hazardous voltage when connecting an unsuitable power supply

Touching live components can result in death or severe injury.

- Only use power supplies that provide SELV (Safety Extra Low Voltage) or PELV (Protective Extra Low Voltage) output voltages for all connections and terminals of the electronics modules.

⚠ WARNING

Danger to life from touching live parts on damaged devices

Improper handling of devices can result in damage.

For damaged devices, hazardous voltages can be present at the enclosure or at exposed components; if touched, this can result in death or severe injury.

- Observe the limit values specified in the technical specifications during transport, storage, and operation.
- Do not use damaged devices.

⚠ WARNING

Danger to life through electric shock due to unconnected cable shields

Hazardous touch voltages can occur through capacitive cross-coupling due to unconnected cable shields.

- As a minimum, connect cable shields and the cores of power cables that are not used (e.g. brake cores) at one end at the grounded housing potential.

⚠ WARNING

Danger to life due to electric shock when not grounded

For missing or incorrectly implemented protective conductor connection for devices with protection class I, high voltages can be present at open, exposed parts, which when touched, can result in death or severe injury.

- Ground the device in compliance with the applicable regulations.

⚠ WARNING

Danger to life due to fire spreading if housing is inadequate

Fire and smoke development can cause severe personal injury or material damage.

- Install devices without a protective housing in a metal control cabinet (or protect the device by another equivalent measure) in such a way that contact with fire inside and outside the device is prevented.
- Ensure that smoke can only escape via controlled and monitored paths.

⚠ WARNING

Danger to life from unexpected movement of machines when using mobile wireless devices or mobile phones

Using mobile radios or mobile phones with a transmit power > 1 W closer than approx. 2 m to the components may cause the devices to malfunction, influence the functional safety of machines therefore putting people at risk or causing material damage.

- Switch off wireless devices or mobile phones in the immediate vicinity of the components.

⚠ WARNING

Danger to life due to fire if overheating occurs because of insufficient ventilation clearances

Inadequate ventilation clearances can cause overheating of components followed by fire and smoke development. This can cause death or serious injury. This can also result in increased downtime and reduced service life for devices/systems.

- Ensure compliance with the specified minimum clearance as ventilation clearance for the respective component.

⚠ WARNING

Danger of an accident occurring due to missing or illegible warning labels

Missing or illegible warning labels can result in accidents involving death or serious injury.

- Check that the warning labels are complete based on the documentation.
- Attach any missing warning labels to the components, in the national language if necessary.
- Replace illegible warning labels.

⚠ WARNING

Danger to life when safety functions are inactive

Safety functions that are inactive or that have not been adjusted accordingly can cause operational faults on machines that could lead to serious injury or death.

- Observe the information in the appropriate product documentation before commissioning.
- Carry out a safety inspection for functions relevant to safety on the entire system, including all safety-related components.
- Ensure that the safety functions used in your drives and automation tasks are adjusted and activated through appropriate parameterizing.
- Perform a function test.
- Only put your plant into live operation once you have guaranteed that the functions relevant to safety are running correctly.

Note

Important safety notices for safety functions

If you want to use safety functions, you must observe the safety notices in the safety manuals.

12.5.1.2 Safety instructions for electromagnetic fields (EMF)

⚠ WARNING

Danger to life from electromagnetic fields

Electromagnetic fields (EMF) are generated by the operation of electrical power equipment such as transformers, converters or motors.

People with pacemakers or implants are at a special risk in the immediate vicinity of these devices/systems.

- Ensure that the persons involved are the necessary distance away (minimum 2 m).

12.5.1.3 Handling electrostatic sensitive devices (ESD)

Electrostatic sensitive devices (ESD) are individual components, integrated circuits, modules or devices that may be damaged by either electric fields or electrostatic discharge.

NOTICE

Damage through electric fields or electrostatic discharge

Electric fields or electrostatic discharge can cause malfunctions through damaged individual components, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber of aluminum foil.
- Only touch components, modules and devices when you are grounded by one of the following methods:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

12.5.1.4 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

⚠ WARNING

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.

12.5.1.5 Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public line supplies requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are enabled.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the controller and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example:
   - Hardware defects and/or software errors in the sensors, controllers, actuators, and connection technology
   - Response times of the controller and drive
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of radio devices / cellular phones in the immediate vicinity of the controller
   - External influences / damage
2. In the event of a fault, exceptionally high temperatures, including an open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the inverter, for example:
   - Component malfunctions
   - Software errors
   - Operating and/or ambient conditions outside of the specification
   - External influences / damage

   Inverters of the Open Type / IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that the contact with fire inside and outside the inverter is not possible.
3. Hazardous touch voltages caused by, for example:
   - Component malfunctions
   - Influence of electrostatic charging
   - Induction of voltages in moving motors
   - Operating and/or ambient conditions outside of the specification
   - Condensation / conductive contamination
   - External influences / damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc. if they are too close.
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly.

Note

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant sections in the technical user documentation.

12.5.2 General

12.5.2.1 Overview

Properties

Properties of ADI4 module

An ADI4 module (Analog Drive Interface for 4 axes) is an interface module suitable for operating up to four drives with an analog setpoint interface and TTL/SSI encoders on an equidistant PROFIBUS-DP.

Communication between the controller and the ADI4 is performed via an ADI4-specific message frame type which, in addition to digital input/output data, also contains a message frame type for each drive specified according to a PROFIDrive profile (standard message frame 3, see Chapter "AUTOHOTSPOT"). As part of cyclic DP communication, the actual drive values (encoder values) are transferred from the ADI4 module to the controller via PROFIBUS DP, and the speed setpoints calculated by the controller are transferred to the ADI4 module.

The transferred speed setpoints are then output from the ADI4 module to the drives as analog values.

Essential features

Features of the ADI4 module

The module has the following essential features:

- PROFIBUS DP connection (maximum of 12 Mbits/s)
- 4 servo interfaces
  - Inputs: TTL/SSI encoder for incremental and absolute measuring systems
  - Bidirectional analog outputs: ±10 V

- General and drive-specific digital input/output signals
- On-board status display via four diagnostic LEDs

To supply the module and digital outputs with power, an external voltage source (+24 VDC) is needed.

Article No. and firmware version

Article number

Article number: 6FC5 211-0BA01-0AA4

Firmware version

The firmware version is not displayed directly on the module. The article number and firmware version correlate as follows:

Article number	Firmware version
6FC5 211-0BA01-0AA2	1.3.3
6FC5 211-0BA01-0AA4	1.4.8

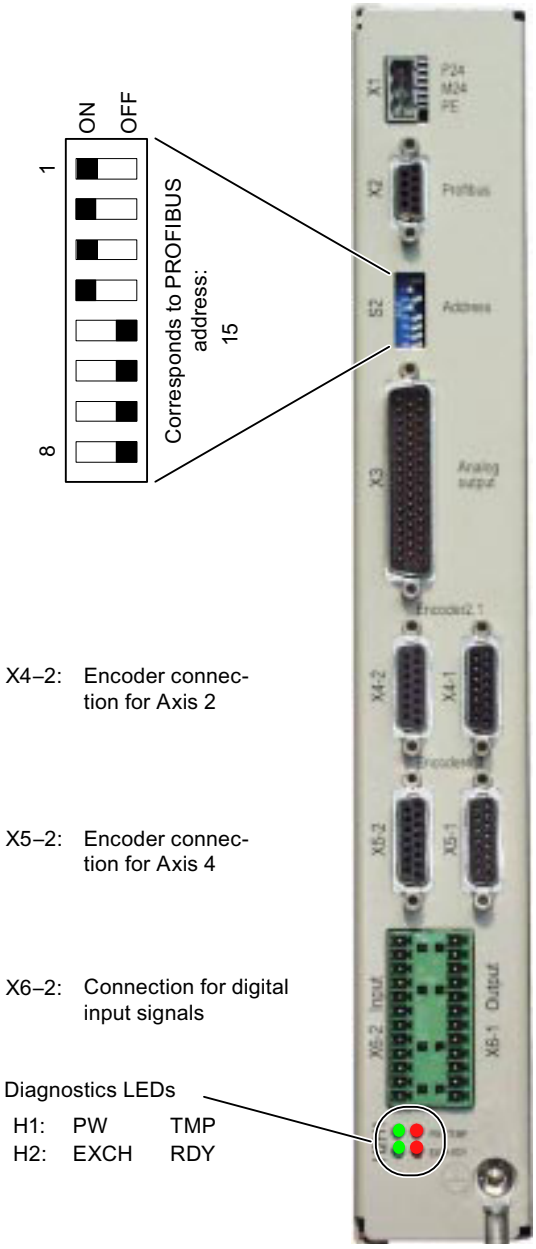
Boundary conditions

The following supplementary conditions must be taken into account for the operation of an ADI4 on the PROFIBUS DP:

- An ADI4 can only be operated on an **equidistant** PROFIBUS DP.
- An ADI4 is **not** a certified DP standard slave as defined by the PROFIDrive profile. For example, an ADI4 does not permit acyclic communication.

12.5.3 Hardware description

12.5.3.1 Overview of connections



X1: External power supply 24 VDC

X2: PROFIBUS connection

S2: DIP switch for setting the PROFIBUS address

X3: Analog setpoint outputs ±10 VDC, Axis 1 to 4

X4–2: Encoder connection for Axis 2

X4–1: Encoder connection for Axis 1

X5–2: Encoder connection for Axis 4

X5–1: Encoder connection for Axis 3

X6–2: Connection for digital input signals

X6–1: Connection for digital output signals

Diagnostics LEDs

H1: PW TMP
H2: EXCH RDY

Connection for protective ground Screw M6

Figure 12-104 Overview of connections

12.5.3.2 Interface description

Interface overview

The module has the following interfaces:

Table 12-37 Interface overview of ADI4

Interface	Designation	Type
External +24 V power supply	X1	Plug
PROFIBUS DP	X2	Socket
PROFIBUS DP address	S2	DIP switch
Analog setpoint interface	X3	Plug
Encoder connection for Axis 1	X4-1	Socket
Encoder connection for Axis 2	X4-2	Socket
Encoder connection for Axis 3	X5-1	Socket
Encoder connection for Axis 4	X5-2	Socket
Digital outputs	X6-1	Plug
Digital inputs	X6-2	Plug
Module status	H1/H2	LEDs

Interface (X1): External power supply

Connection

3-pin connector MSTB 2.5/3-ST-5.08 by Phoenix

Pin assignment

Table 12-38 Pin assignment: External power supply (X1)

Pin	Designation	Type[1)]	Function
1	P24EXT1	VI	External supply for module (+24 V)
2	M24EXT1	VI	Reference for external supply
3	PE	VI	Protective conductor of the external supply
[1)] VI: Voltage input			

Connection cables

The required connecting cables must be provided by the user:

Wire, conductor cross section: 1.0 - 1.5 mm^2 (AWG17 - AWG16)

Supply voltage

The specifications of the supply voltage can be found in the chapter "Power supply (Page 8348)".

Interface (X2): PROFIBUS DP

Connection

9-pin sub D socket

Pin assignment

Table 12-39 Pin assignment: PROFIBUS DP (X2)

Pin	Designation	Type[1)]	Function
1	-	-	-
2	-	-	-
3	RxD/TxD-P	B	Receive/transmit data P (B line)
4	RTS	O	Request to Send
5	DGND	VO	Data reference potential (M5V)
6	VP	VO	Supply voltage plus (P5V)
7	-	-	-
8	RxD/TxD-N	B	Receive/transmit data N (A line)
9	-	-	-

[1)] VO: Voltage output
O: Output
B: Bidirectional

Connectors

- 6ES7 972-0BA41-0XA0; cable outlet 35°, without PG connection socket
- 6ES7 972-0BB41-0XA0; cable outlet 35°, with PG connection socket

Cables

- 6XV1 830-0EH10; by the meter; without trailing capability
- 6XV1 830-3EH10; by the meter; with trailing capability

Other technical data

Maximum possible data rate: 12 Mbits/s

Interface (S2): PROFIBUS address

Setting

The PROFIBUS address of the ADI 4 DP slave can only be 15 or 16 for the 802D sl and is set using the S2 switch.

- PROFIBUS address 15: S2 switch, 1 to 4 set to ON
- PROFIBUS address 16: S2, only switch 5 set to ON

Table 12-40 Meaning of switch S2

Switches	Meaning
1	PROFIBUS address: $2^0 = 1$
2	PROFIBUS address: $2^1 = 2$
3	PROFIBUS address: $2^2 = 4$
4	PROFIBUS address: $2^3 = 8$
5	PROFIBUS address: $2^4 = 16$
6	PROFIBUS address: $2^5 = 32$
7	PROFIBUS address: $2^6 = 64$
8	Not used

Note

A newly set PROFIBUS address will only come into effect after power OFF/ON.

Interface (X3): Analog setpoint interface

Connection

50-pin sub D connector

Pin assignment

Table 12-41 Pin assignment: Analog setpoint interface (X3)

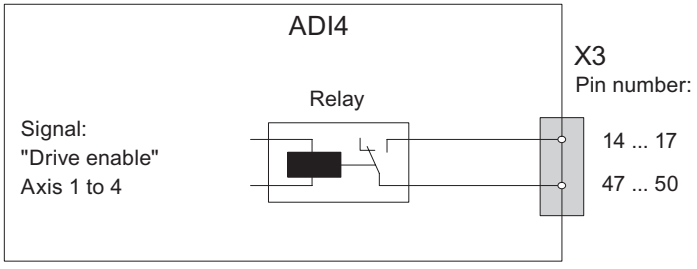
Pin	Designation	Type[1)]	Function
1	SW1	VO [3)]	Setpoint of Axis 1 (±10 V)
2	BS2	VO	Reference for setpoint of Axis 2
3	SW3	VO [3)]	Setpoint of Axis 3 (±10 V)
4	BS4	VO	Reference for setpoint of Axis 4
5-13	-	-	-
14	RF1_1	K [2)]	"Drive enable" of Axis 1, Relay Contact 1
15	RF2_1	K [2)]	"Drive enable" of Axis 2, Relay Contact 1

Pin	Designation	Type[1)]	Function
16	RF3_1	K [2)]	"Drive enable" of Axis 3, Relay Contact 1
17	RF4_1	K [2)]	"Drive enable" of Axis 4, Relay Contact 1
18-33	-	-	-
34	BS1	VO	Reference for setpoint of Axis 1
35	SW2	VO [3)]	Setpoint of Axis 2 (±10 V)
36	BS3	VO	Reference for setpoint of Axis 3
37	SW4	VO [3)]	Setpoint of Axis 4 (±10 V)
38-46	-	-	-
47	RF1_2	K [2)]	"Drive enable" of Axis 1, Relay Contact 2
48	RF2_2	K [2)]	"Drive enable" of Axis 2, Relay Contact 2
49	RF3_2	K [2)]	"Drive enable" of Axis 3, Relay Contact 2
50	RF4_2	K [2)]	"Drive enable" of Axis 4, Relay Contact 2

1) VO Voltage output
K Relay contact

2) Max. current carrying capacity: 2 A for 150 VDC or 125 VAC
Max. number of switching cycles:
- 24 VDC, 1 A: 10^7
- 24 VDC, 2 A: 10^5

3) Max. current carrying capacity: 10 mA (RL: 1 kW - 2 kW)



Prefabricated cables

Article number: 6FX2 002-3AD01-□□□□

Cable length: ≤ 35 m

Information regarding the length codes is provided in:
References: /Z/ Catalog NC Z

Resolution of setpoint outputs

The analog setpoint outputs on the interface (X3) have the following resolution: 16-bit, including sign.

Interfaces (X4-1/X4-2/X5-1/X5-2): Encoder interfaces

Connection

15-pin SUB-D socket

Pin assignment

Table 12-42 Pin assignment: Encoder interface of Axis 1 to 4 (X4-1/X4-2/X5-1/X5-2) for incremental encoder (TTL) and absolute encoder (SSI)

Pin	Designation[1]		Type[2]	Function
	Incremental	Absolute (SSI)		
1	Not assigned		-	-
2	-	CLSx	O	SSI shift clock
3	-	CLSx_N	O	SSI shift clock inverted
4	P5MS		VO	5 VDC supply voltage
5	P24SSI		VO	24 VDC supply voltage
6	P5MS		VO	5 VDC supply voltage
7	MEXT		VO	Reference for supply voltage
8	Not assigned		-	-
9	MEXT		VO	Reference for supply voltage
10	Rx_S	-	I	Zero mark signal (U_{a0})
11	XRx_S	-	I	Zero mark signal inverted ($/U_{a0}$)
12	XBx_S	-	I	Encoder signal track B inverted ($/U_{a2}$)
13	Bx_S	-	I	Encoder signal track B (U_{a2})
14	XAx_S	-	I	Encoder signal track A inverted (U_{a1})
	-	DATAx_N	I	SSI data inverse
15	Ax_S	-	I	Encoder signal track A (U_{a1})
	-	DATAx	I	SSI data

[1] x_ : Number of the encoder interface with (X4-1) = 1, (X4-2) = 2, (X5-1) = 3, (X5-2) = 4
[2] VO: Voltage output
I: Signal input
O: Signal output

Prefabricated cables

The following prefabricated cables can be used, depending on the encoder type:

- **Incremental encoder (TTL) with RS 422 and operating voltage 5 V or 24 V**
  Article number: 6FX8 002-2CD01-1□□0 (5 V)
  Article number: 6FX5 002-2CD24-1□□0 (24 V)
  Information on the cable lengths can be found in the "Maximum cable lengths" section.
- **Absolute encoder with SSI**
  Article number: 6FX8 002-2CC11-□□□0
  Information on the cable lengths can be found in the "Maximum cable lengths" section.
- **1FT5 motor with integrated ROD320 encoder**
  Article number: 6FX8 002-2CE02-1□□0
  Cable length: Can be found in the "Maximum cable lengths" section.
  Information regarding the length codes is provided in:
  References: /Z/ Catalog NC Z

Maximum cable lengths

The maximum cable length depends on the following two parameters:

- **Encoder supply voltage**

Table 12-43 Encoder supply voltage

Supply voltage: 5 V DC		
Tolerance	**Current consumption**	**Max. cable length**
4.75 V to 5.25 V	≤ 300 mA	25 m
4.75 V to 5.25 V	≤ 220 mA	35 m

Supply voltage: 24 VDC		
Tolerance	**Current consumption**	**Max. cable length**
20.4 V to 28.8 V	≤ 300 mA	100 m
11 V to 30 V	≤ 300 mA	300 m

- **Transmission frequency**

Table 12-44 Transmission frequency

Encoder type	**Power supply**	**Frequency**	**Max. cable length**
Incremental (TTL)	5 V	1 MHz	10 m
		500 kHz	35 m
	24 V	500 kHz	150 m
Absolute (SSI)	24 V	750 kbit/s	10 m
		187.5 Kbit/s	250 m

Note

If cable lengths longer than 25 m or 35 m are needed for incremental encoders, encoder types with a 24 VDC supply voltage can be used instead.

NOTICE
Maximum cable lengths To ensure error-free transmission of encoder data, do not exceed the maximum cable lengths shown in these tables.

Encoder supply voltages

The encoder supply voltages must comply with the following specification:

Table 12-45 Specification of encoder supply voltages

	Supply voltage[1)]	
	P5MS	P24SSI
Voltage		
• Minimum	4.75 V	20.4 V
• Nominal	5 V	24 V
• Maximum	5.25 V	28.8 V
Ripple		
• Maximum	50 mV_{pp}	3.6 V_{pp}
Current load		
• Per encoder connection	0.3 A	
• Maximum	1.35 A	1 A
[1)] P5MS: Supply voltage for encoder (+5 VDC) P24SSI: Supply voltage for encoder (+24 VDC)		

Connectable measuring systems

Incremental encoder (TTL)

- Differential transmission with RS 422 and operating voltage 5 V or 24 V:
  - Track A as true and inverted signal (U_{a1}, $/U_{a1}$)
  - Track B as true and inverted signal (U_{a2}, $/U_{a2}$)
  - Zero signal N as true and inverted signal (U_{a0}, $/U_{a0}$)
- Maximum output frequency: 1.5 MHz
- Phase shift of Track A to Track B: 90° ± 30°

- Current consumption: Max. 300 mA
- Encoders with distance-coded zero marks/reference marks are not generally enabled.

Absolute encoder (SSI)

- Transmission method: Synchronous serial interface (SSI) according to RS 485 with 5 V differential signal transmission (RS 422 standard)
  - Output signal: Data as true and inverted signal
  - Input signal: Shift clock as true and inverted signal
- Data transmission format: "Pine tree"
- Resolution: Max. 25 bits
- Maximum transmission rate: 1 Mbit/s
- Current consumption: Max. 300 mA

The following SSI encoders are supported:

- All SSI encoders in the pine tree format whose MsgLength is equal to the number of "significant data bits":
  - 25-bit with 12-bit multiturn and 13-bit singleturn information (setting of the MsgLength 25)
  - 21-bit with 8-bit multiturn and 13-bit singleturn information (setting of the MsgLength 21)
  - 21-bit with 9-bit multiturn and 12-bit singleturn information (setting of the MsgLength 21)
  - 13-bit with 13-bit singleturn information (setting of the MsgLength 13)
- SSI encoder in the pine tree format with the following data:
  - 25-bit with 12-bit multiturn and 12-bit singleturn (setting of the MsgLength 24)
  - 21-bit with 8-bit multiturn and 12-bit singleturn information (setting of the MsgLength 20)
  - 13-bit with 12-bit singleturn information (setting of the MsgLength 12)
  - This applies as long as the encoder permits that not all available data is read!
- All right-justified encoders that satisfy the first condition indicated above.

Interface (X6-1): Digital outputs

Connection

Two 12-pin connectors FK-MCP 1.5/15-ST-3.81 made by Phoenix

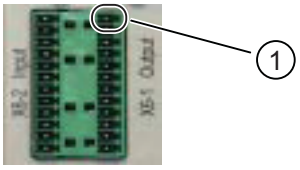
Pin assignment

Table 12-46 Pin assignment: Digital output interface (X6-1)

Pin	Designation	Type[1)]	Function
1	P24EXT2	VI	External 24 VDC supply voltage
2	Q0	DO	Digital output signal 1
3	Q1	DO	Digital output signal 2
4	Q2	DO	Digital output signal 3
5	Q3	DO	Digital output signal 4
6	DIR1	DO	Digital output signal 5 or directional signal of axis 1[3)]
7	DIR2	DO	Digital output signal 6 or directional signal of axis 2[3)]
8	DIR3	DO	Digital output signal 7 or directional signal of axis 3[3)]
9	DIR4	DO	Digital output signal 8 or directional signal of axis 4[3)]
10	RDY1	K [2)]	"Ready" signal of relay contact 1
11	RDY2	K [2)]	"Ready" signal of relay contact 2
12	MEXT2	VI	Reference of the external supply voltage

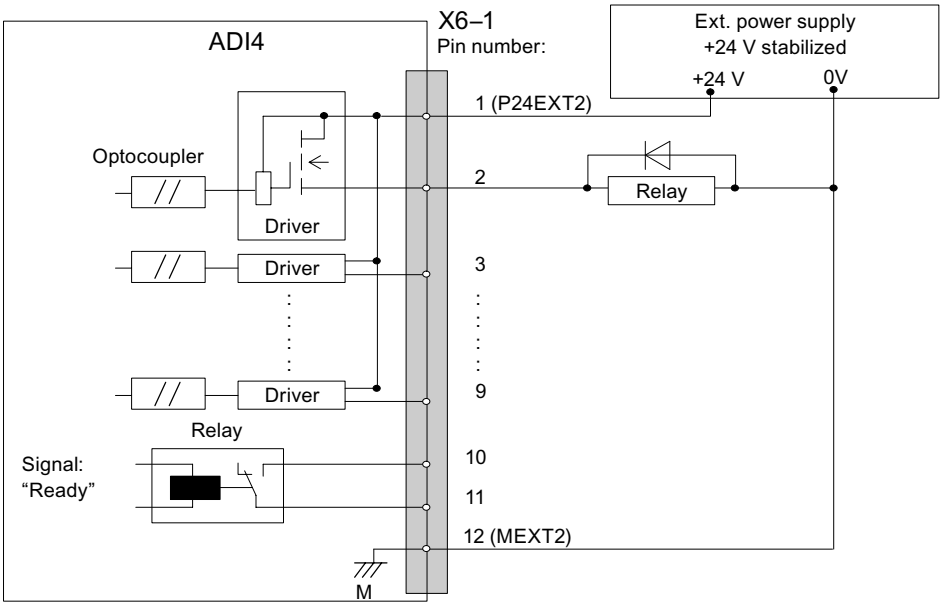
[1)] VI: Voltage input
DO: Digital output (24 V)
K: Relay contact
[2)] Max. current carrying capacity: 2 A for 150 VDC or 125 VAC;
Max. number of switching cycles:
- 24 VDC, 1 A: 10^7
- 24 VDC, 2 A: 10^5

[3)] For "unipolar spindle" function (or unipolar motor)



Pin 1a

① PIN 1



PIN assignment 2

Supply voltage

To supply the digital outputs with power, an external 24 VDC voltage source must be connected to X6-1, Pin 1 (P24EXT2).

The reference ground of the external voltage source must be connected to X6-1, Pin 15 (MEXT2).

You will find additional information in Section "Technical data (Page 8352)".

Electrical specification

Table 12-47 Electrical specification of the digital outputs

Digital outputs	Min.	Typical	Max.	Nominal
High-level voltage (U_H)	V_{CC} - 3 V	1)	V_{CC}	24 V
Output current I_{OUT}	-	-	500 mA	-
Voltage with low level (U_L)	-	-	-	0 V
Leakage current at low level	-	50 µA	400 µA	-
Signal delay T_{PHL}, T_{PLH} 2)	-	0.5 ms	-	-
Supply voltage of the digital outputs 1) Typical output voltage: V_{CC} - I_{OUT} * R_{ON} - 0.65 V V_{CC}: Actual operating voltage P24EXT2 Max. output current I_{OUT}: 500 mA Max. short-circuit current: 4 A (max. 100 µs, V_{CC} = 24 V) Internal resistance R_{ON}: 0.4 Ω 2) The PROFIBUS communication time as well as the application cycle time must also be taken into account. Incorrect connection (polarity reversal) causes neither high level nor destruction of the outputs.				

General electrical properties

- Galvanic isolation using optocouplers
- Current limitation to maximum 500 mA
- Protection against: short-circuit, overtemperature, and loss of ground
- Automatic disconnection in case of undervoltage

Relay contact: "Ready" signal

The relay contact remains open or is **opened** if the module is in one of the following states:

- Initialization of the module after Power ON
- Power failure or hardware interrupt (NMI)
- No cyclic communication to the DP master
- PLL error
- Synchronization error
- Overtemperature

The relay contact is **closed** if both conditions are present:

- Module status "Ready"
- Cyclic communication with the DP master

Connection cables

The required connection cables must be provided by the user:

- Supply voltage X6-1, Pin 1 and 12 (P24EXT2):
  Wire, conductor cross-section of 1.5 mm^2 (AWG16)
- Digital outputs X6-1, Pins 2 to 9:
  Wire, conductor cross-section 0.5 to 1.5 mm^2 (AWG20 - AWG16)
- Ready X6-1, Pin 10 and 11:
  Wire, conductor cross-section of 1.5 mm^2 (AWG16)

Note

The maximum permissible length of the digital signal cable is 30 m.

Interface (X6-2): Digital inputs

Connection

Two 12-pin connectors FK-MCP 1.5/15-ST-3.81 made by Phoenix

Note

Enable signals for analog axes

The Drive Ready signals are used as enable signals for the analog axes. If there is NO 24 V at these inputs, the POWER enable of the axis is NOT set. The axis therefore cannot be enabled/traversed.

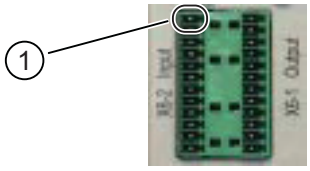
Pin assignment

Table 12-48 Pin assignment: Digital input interface (X6-2)

Pin	Designation	Type [1)]	Function
1	P24OUT	VI	24 VDC supply voltage
2	BERO1	DI	Input signal of BERO / external zero mark 1
3	BERO2	DI	Input signal of BERO / external zero mark 2
4	BERO3	DI	Input signal of BERO / external zero mark 3
5	BERO4	DI	Input signal of BERO / external zero mark 4
6	MEPU1	DI	Measuring signal, measuring input 1 (see "Measuring input" below)
7	MEPU2	DI	Measuring signal, measuring input 2 (see "Measuring input" below)

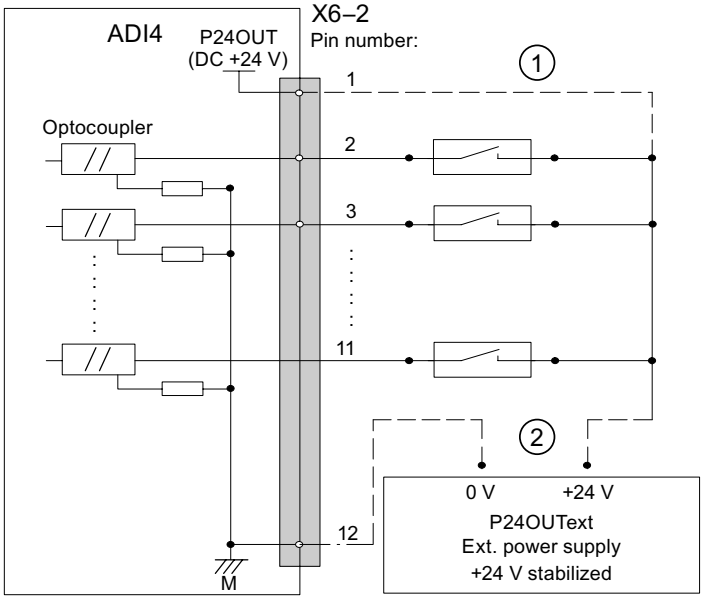
Pin	Designation	Type [1]	Function
8	DRV1_RDY	DI	"Drive Ready" signal of axis 1
9	DRV2_RDY	DI	"Drive Ready" signal of axis 2
10	DRV3_RDY	DI	"Drive Ready" signal of axis 3
11	DRV4_RDY	DI	"Drive Ready" signal of axis 4
12	MOUT	VI	Reference of the supply voltage

[1] VI: Voltage input
DI: Digital input (24 V)



Pin 1b

① PIN 1



PIN assignment 3

① Connection if the internal supply voltage P24OUT is used;
the connection in accordance with ② is no longer required.

② Connection if an external supply voltage P24OUText is used;
the connection in accordance with ① is no longer required.

Internal supply voltage P24OUT

Specification of the internal supply voltage P24OUT available at X6-2, Pin 1 for the digital inputs:

Table 12-49 Specification of the supply voltage P24OUT

Voltage	
• Minimum	20.4 V
• Nominal	24 V

Voltage	
• Maximum	28.8 V
Ripple	
• Maximum	3.6 V_{pp}
Current load	
• Typical	0.1 A
• Maximum	1 A
Power consumption	
• Typical	3.02 W
• Maximum	30.2 W
Insulation Class	A, in accordance with DIN 57110b
Typical output voltage: V_{CC} - I_{OUT} * RON - 0.65 V V_{CC}: Actual P24OUT operating voltage Max. output current I_{OUT}: 1 A Internal resistance R_{ON}: 0.4 Ω The supply voltage P24OUT is short-circuit proof.	

External supply voltage P24OUText

If an external supply voltage is used, its reference ground must be connected to X6-2, Pin 12 (M).

X6-2, Pin 1 (P24OUT) then remains open.

Electrical specification

Digital inputs	**Min.**	**Typical**	**Max.**	**Nominal**
High-level voltage (U_H)	15 V	1)	30 V	24 V
Input current I_{IN} at U_H	3.7 mA	-	7.5 mA	-
Voltage with low level (U_L)	-30 V	-	+5 V	0 V
Signal delay T_{PHL}, T_{PLH} 2)	-	3 µs	-	-

1) See table "Specification of the supply voltage P24OUT"

2) The PROFIBUS communication time as well as the application cycle time must also be taken into account.

Incorrect connection (polarity reversal) causes neither high level nor destruction of the inputs.

Connection cables

The required connection cables must be provided by the user.

- Supply voltage X6-2, Pin 1 (P24OUT), external supply voltage P24OUText:
  Wire, conductor cross-section of 1.5 mm^2 (AWG16)
- Digital inputs X6-2, Pins 2 to 11:
  Wire, conductor cross-section 0.5 to 1.5 mm^2 (AWG20 - AWG16)

General electrical properties

- Galvanic isolation using optocouplers
- Active current limiting of the inputs
- Protection from negative input voltage

Measuring input

ADI4 supports only measurement of a rising **or** falling edge of the measuring input. A simultaneous request for measurement on a rising edge and a falling edge of the measuring input cannot be parameterized.

Interface (H1/H2): Module status

The module status is displayed on the front of the module with four diagnostic LEDs.

Table 12-50 Diagnostic LEDs (H1/H2)

Designation		Color	Description
H1	POWER	Green	Supply voltage LED = Off: Supply voltage not applied LED = On: Supply voltage is applied
	OVTEMP	Red	Overtemperature display LED = Off: Device temperature < overtemperature limit LED = On: Device temperature ≥ Overtemperature limit
H2	EXCHANGE	Green	Status: Message frame exchange with DP master LED = Off: No message frame exchange with DP master LED = On: Cyclic message frame exchange with DP master
	READY	Red	Ready status: Message frame exchange with DP master LED = Off: Not yet ready LED = On: Ready LED = Off and EXCHANGE = On: Message frame exchange active LED = flashing: Error occurred during message frame exchange

12.5.3.3 Control cabinet installation

Installation

For high frequency interference currents, the housing of the ADI4 module must be connected with low-resistance to the back wall of the control cabinet, and this wall in turn must be connected with low-resistance to the motors/machines. The module should be installed on a bare mounting wall. The connection between the mounting wall and the motors/machines must be electrically conductive and have a large surface area. Coated cabinet walls and DIN rails, or similar mounting means with a small contact area, do not meet this requirement.

Cable routing

Power and signal cables must always be routed separately. All I/O interface (X6-1/X6-2) signal lines should exit jointly. Single strands that are related from the signal point of view must be twisted together. Signal cables and encoder cables should be installed separately.

All cables and lines within the control cabinet should always be placed as close as possible to the control cabinet walls. Extended installation through open space can cause interference injections (antenna effect). The proximity to sources of interference (contactors, transformers, etc.) must be avoided by placing a shield plate between the cable and the source of interference, if necessary. Cables and conductors should not be extended using terminals or similar devices. To protect against interference injections from external sources, signal cables must be shielded.

⚠ **WARNING**
The module has been designed for operation in an enclosed control cabinet. Operation outside an enclosed control cabinet is not permissible.

12.5.3.4 Power supply

ADI4 module

To supply the ADI4 module (+24 VDC), an external power source is needed. The power supply is connected through terminal X1 (P24EXT1) on the front panel of the ADI4 module. For more detailed information, refer to the chapter "Interface (X1): External power supply (Page 8334)".

Digital outputs

To supply (+24 VDC) the digital outputs, an external power source is needed. The power supply is connected through Terminal X6-1, Pin 1 (P24EXT2). For more detailed information, refer to the chapter "Interface (X6-1): Digital outputs (Page 8341)".

Digital inputs

If the digital inputs are not supplied with the internal supply voltage of X6-2, Pin 1 (P24OUT), this supply voltage can optionally be replaced with an external power source (+24 VDC, 1 A maximum).

The reference ground (GND) of the external power supply source must be connected with X6-2, Pin 12. X6-2, Pin 1 (P24OUT) remains open.

Specification of the supply voltages (+24 VDC)

The external supply voltages for the ADI4 module, the digital outputs, and optionally the digital inputs must comply with the specifications provided in the "Encoder supply voltages" table.

Table 12-51 Specification of the external supply voltages

	Supply voltage[1]		
	P24EXT1	**P24EXT2**	**P24OUText**
Voltage			
• Minimum		18.5 V	
• Nominal		24 V	
• Maximum		30.2 V	
Ripple			
• Maximum		3.6 V_{pp}	
Current load			
• Typical	0.5 A	-	0.1 A
• Maximum	1 A	8 A	1 A
Power consumption			
• Typical	12 W	-	3.02 W
• Maximum	30.2 W	241.6 W	30.2 W

[1] P24EXT1: Supply voltage of the ADI4 module
P24EXT2: Supply voltage for the digital outputs
P24OUText: Optional supply voltage for the digital inputs

NOTICE
The external supply voltages must each be generated as functional extra-low voltage with safe electrical isolation (DIN EN 60204-1, PELV).

Fuse

On the module side, supply voltages P24EXT1 and P24EXT2 must be protected against the following:

- Overvoltage
- Short-circuit (electrical current limiting of outputs)
- Polarity reversal
- Overload
  - P24EXT1: Fuse 2.5 A / 250 V
  - P24EXT2: Fuse 8 A / 125 V

12.5.3.5 Grounding

Grounding

The module must be installed according to EN 60204.

The user must ground each of the supply voltages. To do this, a connection must be established from Terminal X1, Pin 2 (MEXT1) or X6-1, Pin 15 (MEXT2) to a central grounding point of the system.

If a large-area, permanent metallic connection with the central grounding point is not possible using the rear panel, the module must be connected to the grounding rail by means of a wire (cross-section > 10 mm^2).

⚠ CAUTION
A protective conductor must be connected. An M6 screw is provided on the lower right of the front of the housing to connect the protective conductor. See Chapter "AUTOHOTSPOT".

See also

Overview of connections (Page 8333)

12.5.3.6 Dimension drawing

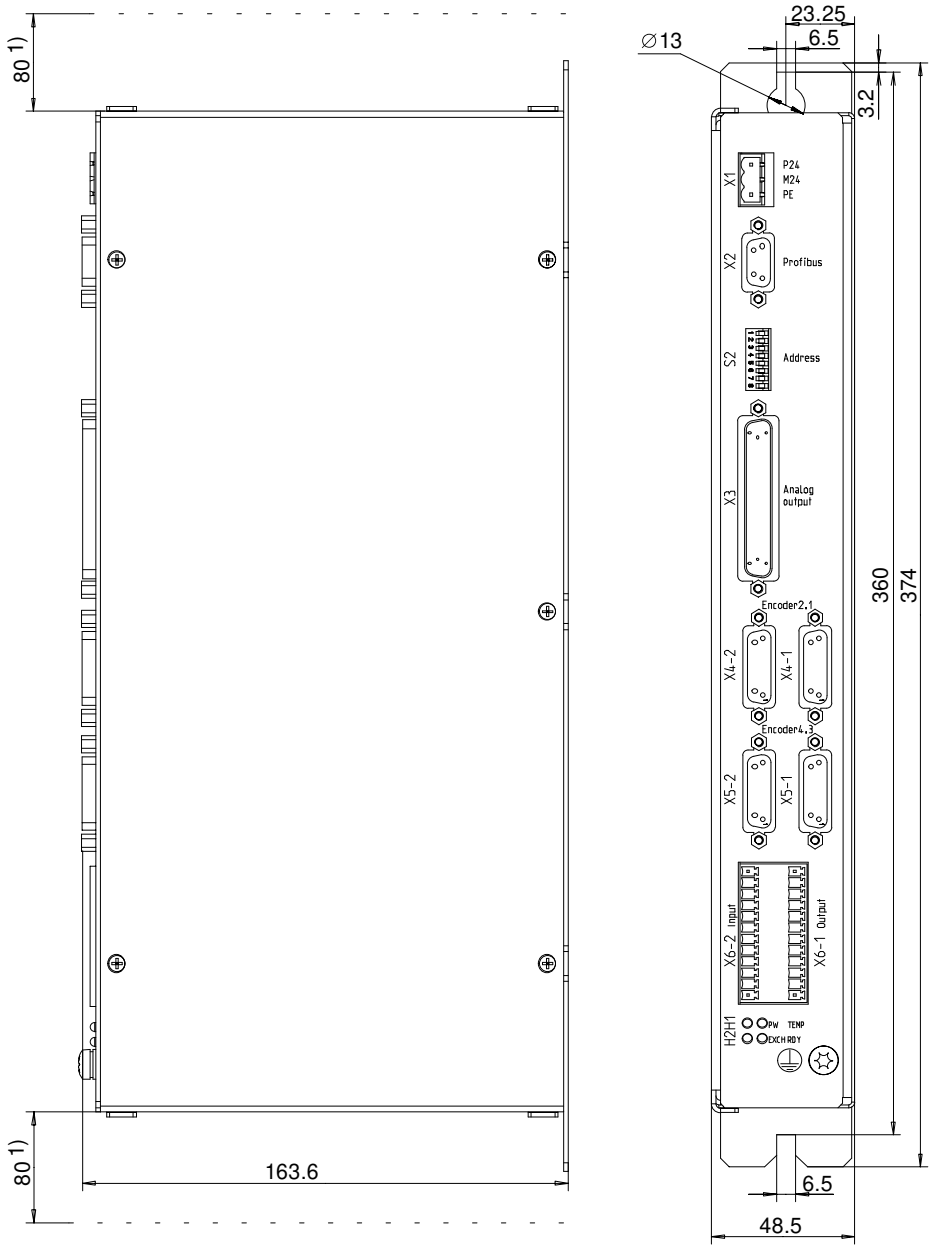


Figure 12-105 Dimension drawing: ADI4

1) Necessary clearance to ensure sufficient ventilation

Maximum tightening torque for all screws: 0.8 Nm

12.5.3.7 Technical data

Table 12-52 Technical data of the ADI4 module

Safety		
Degree of protection	IP20	
Protection class	Protection class I in accordance with VDE 0106 Part 1: 1982 (IEC 536); Protection against ingress of foreign bodies and water in accordance with IEC 529	
Approvals	UL/CSA, CE	
Power consumption		
Nominal load	12 W	
Maximum	30.2 W	
Mechanical specifications		
Dimensions WxHxD [mm]	154.4 x 325 x 48.5	
Weight	Approximately 1.5 kg	
Climatic ambient conditions		
Heat dissipation	Open-circuit-ventilated	
	Operation	**Storage/transport**
Temperature limits	0 ... +55° C	-20 to 55 °C/-40 to 70 °C
Relative humidity limits	5 to 95 % without condensation	5 to 95 % without condensation
Condensation	Not permitted	
Atmospheric pressure	700 to 1060 hPa	700 to 1060 hPa
Transportation altitude	-	-1000 to 3000 m
Shock stress during transportation		
Free fall in transport packaging	≤ 1000 mm	

12.5.4 Parameter assignment

12.5.4.1 Boundary conditions of ADI4 DP slave

Note

The following boundary conditions must be taken into account for the operation of an ADI4 DP slave on the PROFIBUS DP:

- An ADI4 DP slave is not a certified DP standard slave as defined by the PROFIDrive profile. For example, an ADI4 DP slave does not enable acyclic communication. Therefore, an ADI4 DP slave can only be operated on a DP master specially released for this purpose.
- An ADI4 DP slave can only be operated on an equidistant PROFIBUS DP. The minimum DP cycle is 1 ms.

12.5.4.2 Requirements

Components for parameter assignment

The following components are required for assigning parameters for an ADI4 DP slave:

- ADI4:
  as from article number: 6FC5 211-0BA01-0AA1
  Firmware Version 01.02.02 and higher
- SIMATIC STEP 7 Version 5.1 and higher
- SIMOTION
  - SIMOTION P or C: SIMOTION V2.1 and higher (SCOUT and Runtime)
  - SIMOTION D: SIMOTION V3.1 and higher (SCOUT and Runtime)

12.5.4.3 PROFIBUS DP parameter assignment

Parameter assignment sequence

Parameter assignment sequence

The PROFIBUS DP parameter assignment for the ADI4 DP slave can be generally divided into the following steps:

1. Step
   After inserting the ADI4 DP slave in the configuration, the following parameters are assigned on a slave-specific basis:
   - PROFIBUS parameters (see Section "PROFIBUS parameters (Page 8354)")
   - Function parameters (see Section "Function parameters (Page 8361)")

   Step 1 should be carried out first for **all** ADI4 DP slaves needed in the configuration.
2. Step
   Parameter assignment of the equidistant cyclic DP communication (see Section "Parameter assignment of the DP communication (Page 8370)")
   Step 2 can be performed **last** on **any** ADI4 DP slave. These settings can be transferred to all other ADI4 DP slaves by means of the adjustment function of the SlaveOM.

Inserting an ADI4 DP slave in the configuration

Procedure

1. To insert an ADI4 DP slave in the configuration, open the hardware catalog using the **View > Catalog** menu command.
   The ADI4 DP slave can be found at:
   – Profile: **Standard**
   PROFIBUS DP > SINUMERIK > ADI4

   SIMATIC Technology CPU
   If S7-Technology was installed for the Technology CPU, the ADI4 DP slave is located under:
   – Profile: **SIMATIC Technology CPU**
   PROFIBUS DP (DRIVE) > Other FIELD DEVICES > SINUMERIK > ADI4
2. Using a drag-and-drop operation, select the ADI4 DP slave and move it onto to the DP master system in the station window.
   The DP master system is displayed in the station window with the following symbol:

When you release the left mouse button, the DP slave ADI4 is inserted into the configuration.

Note

As you drag the DP slave, the cursor appears as a circle with a slash through it. When the cursor is positioned exactly over the DP master system, it changes to a plus sign, and the DP slave can be added to the configuration.

12.5.4.4 PROFIBUS parameters

Parameter components

Configuring the PROFIBUS parameters

The PROFIBUS parameters are a result of the following:

- PROFIBUS address
- Number of axes and encoders (message frame type)
- I/O addresses

PROFIBUS address

Procedure

Inserting an ADI4 DP slave into the configuration will open the "Properties - PROFIBUS Interface ADI4" dialog, "Parameters" tab:

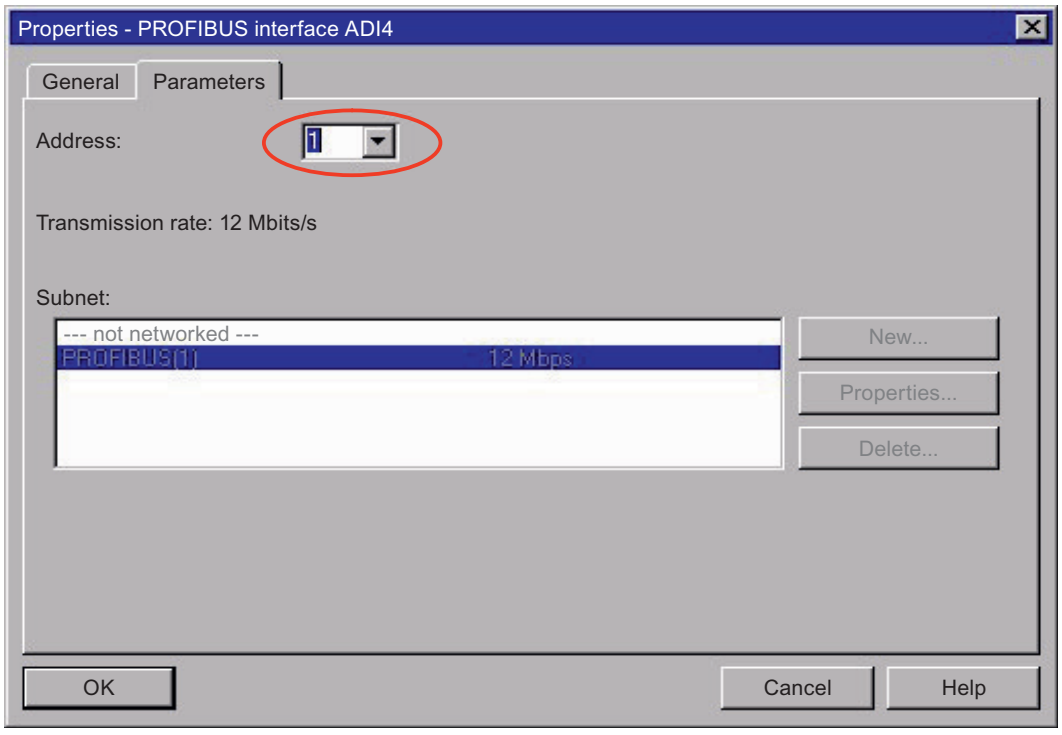


Figure 12-106 PROFIBUS address

The displayed address value was automatically set by HW Config to the next available PROFIBUS address within the configuration.

Note

The PROFIBUS address of the ADI4 DP slave can be set to any value, in principle. However, it must be ensured that the PROFIBUS address setting in HW Config matches the DIP switch setting on the ADI4 DP slave:

There is **no automatic adjustment**!

The following data must agree:

- SIMATIC S7 configuration ADI4 DP slave
  PROFIBUS address
- ADI4 module DIP switch S2
  PROFIBUS address

Note

PROFIBUS address 127

If PROFIBUS address is set to 127, when the module powers up, the firmware release is displayed on the internal module LEDs using a flashing code. It is not recommended to use PROFIBUS address 127 as this lengthens the time that the module requires to power up.

After the dialog is confirmed with "OK", the "DP Slave Properties" dialog box is opened. Continue with the parameter assignment for the message frame type.

Message frame type

Telegram type

The ADI4 DP slave is operated with a specific telegram type:

4 axes, each with one encoder (standard telegram 3) and I/O data

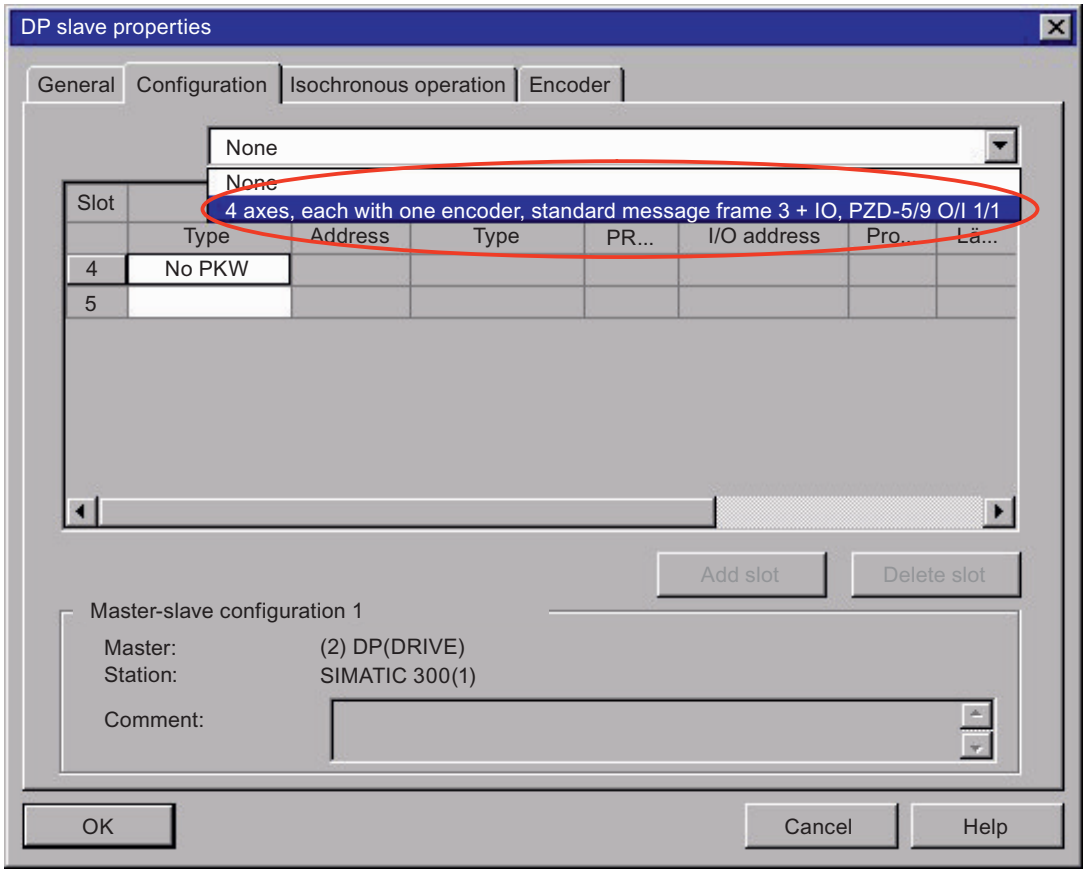


Figure 12-107 Telegram type

Setting the telegram type

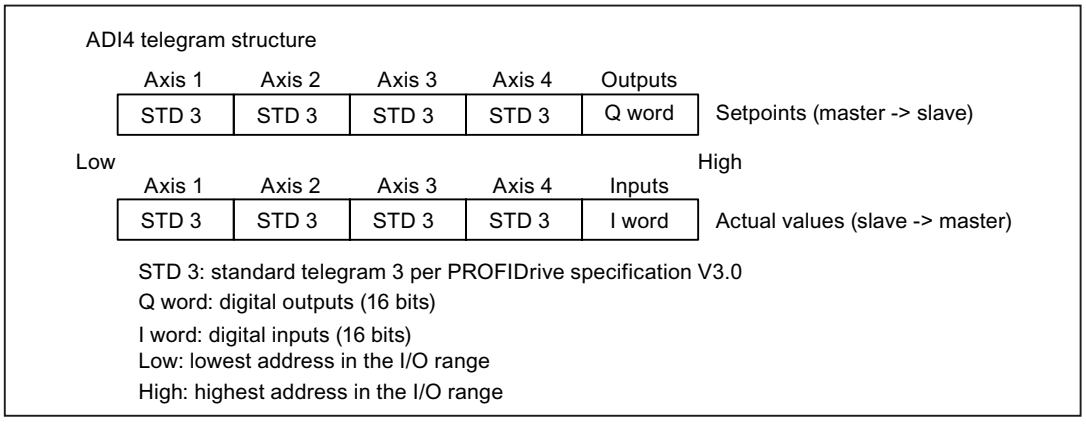
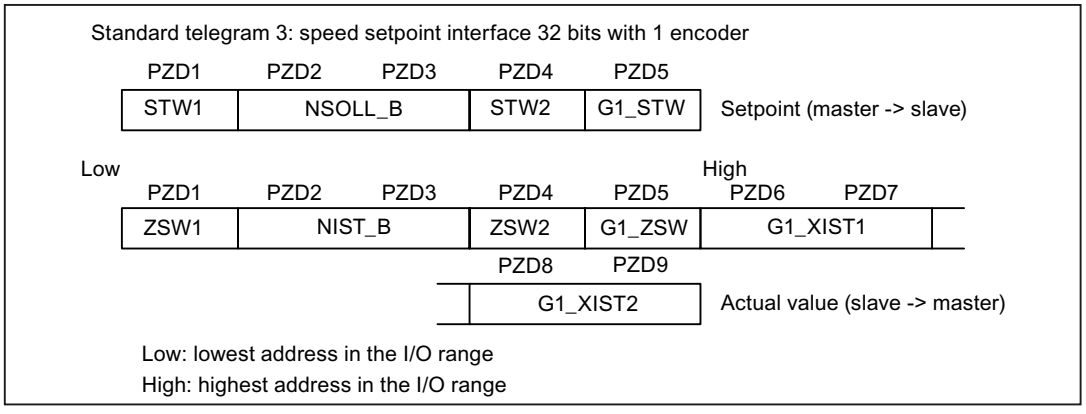
By default, no telegram type is selected. The ADI4-specific telegram type must be explicitly selected in the "Configuration" tab.

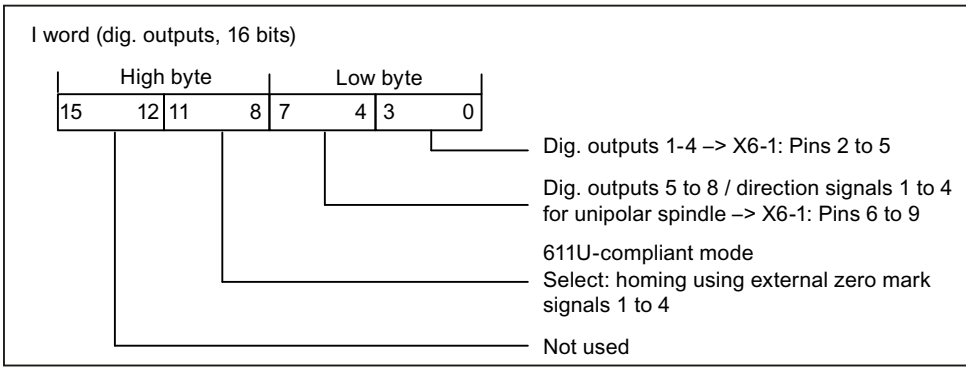
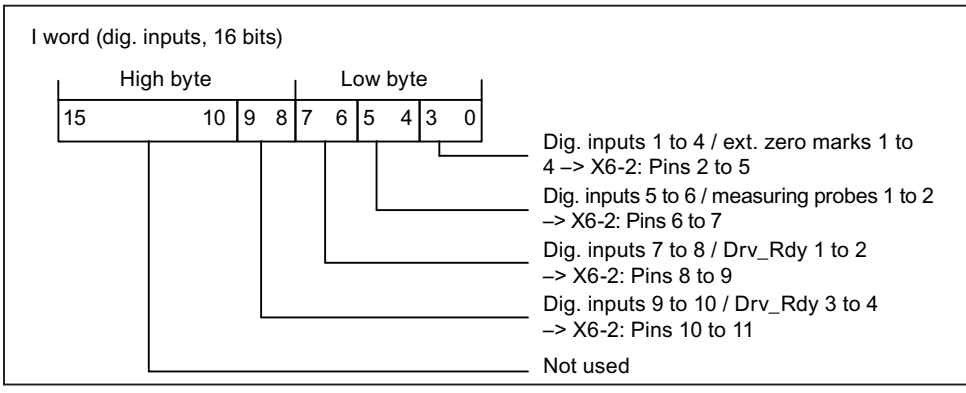
1. In the "DP Slave Properties" dialog box, select the "Configuration" tab.
2. In the "Default" list, select the entry "4 axes, each with one encoder, Standard telegram 3 + IO, PZD-5/9 O/I 1/1".
3. Click "OK".

Telegram structure

The telegram is structured as follows:

Table 12-53 Telegram structure

Telegram type	Description
4 axes, each with one encoder, Standard telegram 3 + IO, PZD-5/9 O/I 1/1	4 x Standard telegram 3 and 1 PZD word each for digital I/O data
PZD x/y Number of process data words, x: Setpoint, y: Actual value, e.g. PZD-5/9: 5 process data words for setpoints 9 process data words for actual values	
	
	

Telegram type	Description
	
	

Note

The telegram type setting for the ADI4 DP slave in HW Config must agree with the telegram type setting in the controller.

There is **no automatic adjustment**.

Encoder control word Gx_STW

Description of the encoder control word (extract) for:

- Find reference mark
- On-the-fly measurement
- Encoder error

Table 12-54 Encoder control word Gx_STW (extract)

Bit	Name		Signal status, description		
0	Find reference mark or On-the-fly measurement	**Functions**	**Find reference mark**: Bit 7 = 0		
			Bit	**Meaning**	
			0	Function 1: Homing using the encoder zero mark (except in "611U-compliant mode")	
			1	Function 2: Homing using rising edge of external zero mark	
			2	Function 3: Homing using falling edge of external zero mark	
			3	Function 4: Not used	
1			**On-the-fly measurement**: Bit 7 = 1		
2			**Bit**	**Meaning**	
3			0	Function 1: Measuring using measuring probe 1 rising edge	
			1	Function 2: Measuring using measuring probe 1 falling edge	
			2	Function 3: Measuring using measuring probe 2 rising edge	
			3	Function 4: Measuring using measuring probe 2 falling edge	
			Note • Bit 0 ... 3 Bit x = 1 Function requested Bit x = 0 Function not requested • If more than one function is enabled, the values for all functions cannot be read until all functions have ended and this has been signaled via the relevant status bit (G1_ZSW, Bit 0 - Bit 3 = 0). • Find reference mark A **falling** edge when homing using the external zero mark is **not** evaluated by SINUMERIK. • Meas. on-the-fly The rising and falling edges of the measuring input can be enabled simultaneously. The measuring input signal is detected according to the direction of the signal change. The measured values are read out consecutively. Notice ADI4 only supports measurement on a rising **or** falling edge. • Find reference mark and on-the-fly measurement Only one of the two functions can be active at a time.		
4		**Command**	**Bit 6, 5, 4**	**Meaning**	
			000	- -	
5			001	Activate function x	
6			010	Read value x	
			011	Cancel function x	
7		**Mode**	0	Find reference mark	
			1	On-the-fly measurement	
:			:		
15	Encoder error		0	No error	
			1	Encoder error pending; error code in Gx_XIST2	

Additional encoder actual value Gx_XIST2

Error codes in Gx_XIST2 where G1_ZSW, Bit 15 = 1

Table 12-55 Error codes in Gx_XIST2

G1_XIST2	Meaning	Possible causes/description
1_{Hex}	Encoder sum error	The encoder signal levels are too low, faulty (inadequate shielding) or cable breakage monitoring has been tripped.
2_{Hex}	Zero mark monitoring	A fluctuation in the measured rotor position has arisen between two encoder zero marks (encoder pulses may be lost).

I/O addresses

Requirements

For communication between the controller and the individual axes of an ADI4 DP slave, it is necessary that the setpoint and the actual value of an axis have the same I/O address.

HW Config takes this requirement into account automatically when an ADI4 DP slave is inserted in the configuration.

Inserting I/O addresses

1. In the "DP Slave Properties" dialog box, select the "Configuration" tab.
2. Under PROFIBUS partner, I/O addr., enter: <**I/O address**>.
3. Click "OK".

Note

The setpoint and actual value of an axis must have the same I/O address.

I address (actual value) = O address (setpoint)

If an ADI4 DP slave is inserted into an S7 project through a copy operation, e.g., from another S7 project, the I/O addresses are assigned directly by HW Config. This may have the consequence that an axis is assigned different I/O addresses for setpoint and actual values. The I/O addresses must be manually corrected in this case.

To avoid access conflicts between the PROFIBUS DP drives and the I/O modules, values **≥ 272** must be used for I/O addresses for the ADI4 DP slave.

Consistency

Consistency setting

The default setting for I/O data consistency is "Total length".

The "Total length" consistency setting means that direct access from the PLC user program (e.g. byte, word, or double word) to this address area of the PLC operating system is not permitted.

12.5.4.5 Function parameters

Parameters

The function-specific parameters of the ADI4 DP slave are entered under the "Encoder" tab:

- Encoder type
- Unipolar spindle (or unipolar motor)
- Shutdown ramp
- Shutdown delay
- Tolerable sign-of-life failures
- Reserved bits for fine resolution
- 611U conformant mode

The figure below shows the corresponding dialog box with sample values for the various encoder types and parameters.

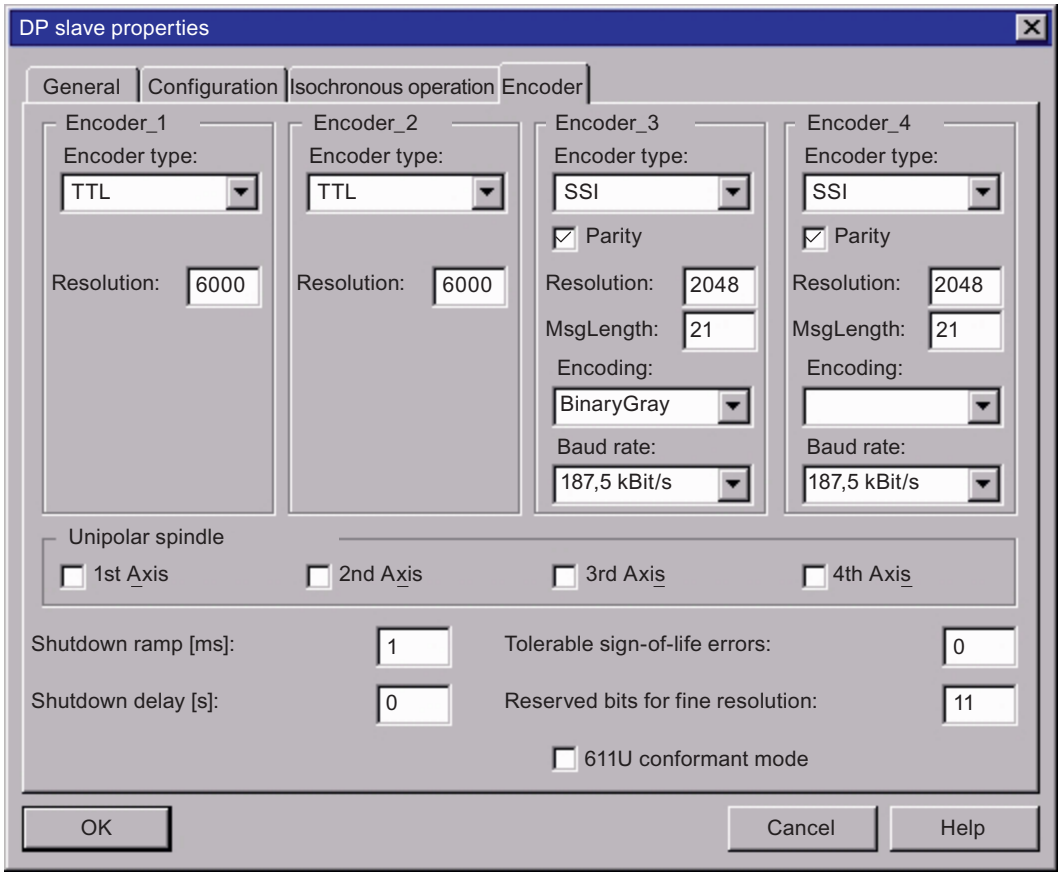


Figure 12-108 "DP Slave Properties" dialog box, "Encoder" tab

Encoder type

Encoder type "not available"

An encoder type setting of "not available" for Encoder x means that Axis x does not exist, or that it is not to be operated. User data transmitted for this axis in the PROFIBUS message frame are empty.

Encoder type TTL

Encoder parameters:

- Resolution
  Encoder resolution in encoder pulses per encoder revolution

---

Note

In the case of spindles with a low-resolution encoder, the actual-value characteristic (incrementing) may be displayed in a non-linear fashion. The incrementing that is visible on the actual-value display is the result of the speed resolution (RR), where:

- RR: 60000 / (T_{DP} * ER * PM)
- RR (speed resolution): [(revolutions/min) / encoder pulse]
- T_{DP} (position control cycle clock = PROFIBUS cycle clock): [ms]
- ER (encoder resolution): [Encoder pulse / revolution]
- PM (pulse multiplication)

Example:

T_{DP} (position control cycle clock = PROFIBUS cycle clock): 2 ms

ER (encoder resolution): 2500 pulses/revolution

PM (pulse multiplication): 4

RR = 60000 (2*2500*4) = 3 (revolutions/min) / encoder pulse

Encoder type SSI

Encoder parameters:

- Parity
  Select this check box if the encoder data are to be transmitted from the encoder to the ADI4 with a parity bit.
- Resolution
  Encoder resolution in encoder pulses per encoder revolution
- MsgLength
  Number of user data bits transmitted by the encoder

- Encoding
  The following encoder codes are supported:
  - Binary
  - Gray
- Baud rate
  The following baud rates are supported:
  - 187.5 Kbits/s
  - 375 Kbits/s
  - 750 Kbits/s

Note

The following boundary conditions must be taken into account for SSI encoders:

- The baud rate setting must be identical for all SSI encoders. If baud rate settings are different, the baud rate of the SSI encoder with the highest encoder number is used.
- In conjunction with ADI4, only absolute encoders (SSI) with "Pine tree" data output format (TSSI) can be operated.

Unipolar spindle or unipolar motor

Introduction

The drive can be moved in two directions. Selecting the "Unipolar spindle" option switches the voltage range of the analog output voltage.

Unipolar spindle not selected

If the "Unipolar spindle" option is **not** selected, an analog voltage in the range of **-10 V** to **+10 V** is output as the setpoint.

Unipolar spindle selected

If the "Unipolar spindle" check box **is selected**, an analog voltage in the range of **0 V** to **+10 V** is output as the setpoint. The direction of rotation is then output from the ADI4, depending on the current speed setpoint, via a digital output of the ADI4:

- Direction of rotation signal for Axis 1 → Digital output X6-1, Pin 6
- Direction of rotation signal for Axis 2 → Digital output X6-1, Pin 7
- Direction of rotation signal for Axis 3 → Digital output X6-1, Pin 8
- Direction of rotation signal for Axis 4 → Digital output X6-1, Pin 9

Note

"Unipolar spindle" (or "Unipolar motor") function is not available.

Shutdown ramp

The "Shutdown ramp" parameter specifies a function that is linear with respect to time. If an error is detected in the ADI4, all ADI4 drives are slowed down to Setpoint "0" in accordance with this function.

A parameter value of "0" brings the drives to an immediate stop (braking at the current limit).

- Unit: [ms]

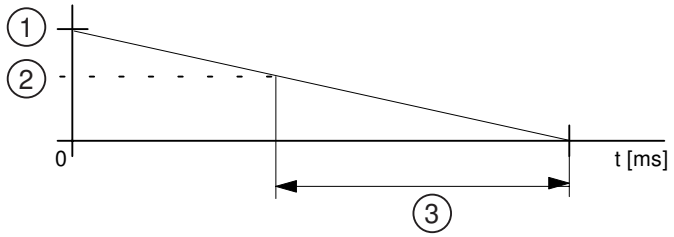


Figure 12-109 Parameter: Shutdown ramp

① Maximum setpoint
② Current setpoint
③ Parameter value: Shutdown ramp

Shutdown delay

The "Shutdown delay" parameter can be used to specify a time after which all ADI4 drives are slowed down to the setpoint "0" following a temperature alarm in the ADI4.

After the "Shutdown delay" has elapsed, the "Shutdown ramp" is taken into account.

- Unit: [s]

Tolerable sign-of-life failures

The "Tolerable sign-of-life failures" parameter specifies the number of sign-of-life failures tolerated for the DP master. If the assigned number is exceeded, the setpoint interfaces of the drives are ramped down to the value "0" using the "Shutdown ramp".

Note

Presently, the "Tolerable sign-of-life failures" parameter may only be used on values in the range of 0 to 13.

Reserved bits for fine resolution

"Additional substitute bits for fine resolution" parameter

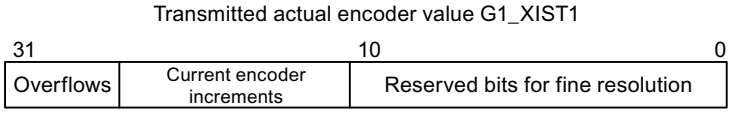
The "Additional substitute bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2.

G1_XIST1

Presently, the number of additional substitute bits for fine resolution must be set permanently to "11". This corresponds to a pulse multiplication of: $2^{11} = 2048$.

Note

Presently, the "Reserved bits for fine resolution" parameter setting must always be set to "11".



G1_XIST2

The number of additional substitute bits for fine resolution is set permanently to "1" and cannot be changed.

Absolute position values supplied in G1_XIST2 are never provided with a number of reserved bits for fine resolution. They are always supplied as they were read from the SSI encoder.

611U conformant mode

Possible settings

In 611U conformant mode, the signal source for homing of axes is no longer specified using the PROFIDrive standard message frame (STD3, encoder control word G1_STW), but rather using the additional digital output word in the PROFIBUS message frame of the ADI4 (see table "Message frame structure" in Section "Message frame type (Page 8356)").

611U conformant mode:

- Not selected
  The signal source for homing is specified via the encoder control word Gx_STW in the PROFIDrive standard message frame.
- Selected
  The signal source for homing is specified via the additional digital output word in the PROFIBUS message frame.

Digital output word

The signal sources for the homing are selected on an axis-specific basis via the following bits of the output word (see also output word in the table "Message frame structure" in Section "Message frame type"):

Table 12-56 Output word: signal sources for homing

Bit	Value	Signal source for homing
8	0	Axis 1: Zero mark of Encoder 1 (X4-1)
	1	Axis 1: **Rising** edge of External zero mark 1 (X6-2, Pin 2)
9	0	Axis 2: Zero mark of Encoder 2 (X4-2)
	1	Axis 2: **Rising** edge of External zero mark 2 (X6-2, Pin 3)

Bit	Value	Signal source for homing
10	0	Axis 3: Zero mark of Encoder 3 (X4-3)
	1	Axis 3: **Rising** edge of External zero mark 3 (X6-2, Pin 4)
11	0	Axis 4: Zero mark of Encoder 4 (X4-4)
	1	Axis 4: **Rising** edge of External zero mark 4 (X6-2, Pin 5)

If the 611U conformant mode has been assigned for an axis to be homed, the axis-specific signal for selection of the signal source must be set in the digital output word of the ADI4 from the PLC user program. This must take place prior to the request of the "Find reference mark" function in the control word.

Note

With 611U conformant mode

- For homing of an axis using an encoder zero mark and an external zero mark, the appropriate axis-specific bit must be set to 0 (encoder zero mark) in the digital output word by the PLC user program.
- The signal source for homing can be switched during operation.

Without 611U conformant mode

- Homing is always performed in relation to the zero mark of the axis.

The following sections show the basic system structure and the respective boundary conditions of the individual homing methods.

Exiting the dialog box

If the "DP Slave Properties" dialog box is exited with "OK", the data are accepted and the dialog box is closed.

Step 1: End

Step 1 of the ADI4 DP slave parameter assignment is now complete.

Homing using encoder zero mark

System structure

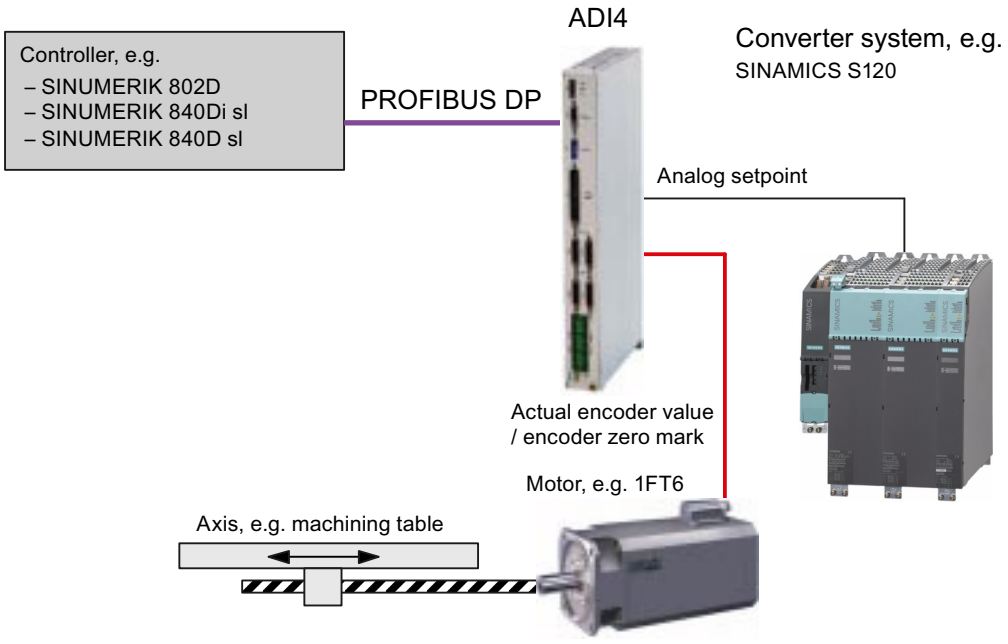


Figure 12-110 Basic system structure: Homing using encoder zero mark

Function

Once the controller requests homing, the ADI4 transmits the actual encoder value to the controller as the home position the next time it detects an encoder zero mark.

Without 611U conformant mode

No further measures are required.

With 611U conformant mode

The relevant signal for the axis to be homed (e.g. Axis 1) must be set in the digital output word:

- Digital output word:
  Bit 0: = 0 ⇒ "Axis 1: Zero mark of Encoder 1 (X4-1)"

Homing using external zero mark

System structure

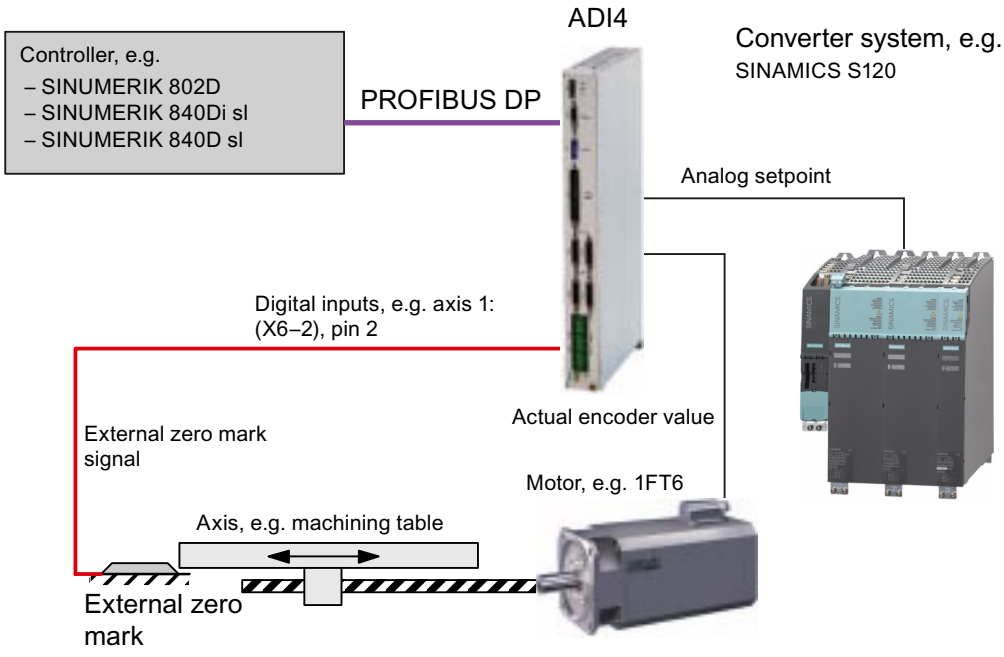


Figure 12-111 Basic system structure: Homing using external zero mark

Function

Once the controller requests homing, the ADI4 transmits the actual encoder value to the controller as the home position the next time it detects an external zero mark signal.

Without 611U conformant mode

The controller must define the relevant function via encoder control word G1_STW:

- Function 2 (Homing via rising edge of external zero mark)
- Function 3 (Homing via falling edge of external zero mark)

With 611U conformant mode

The relevant signal for the axis to be homed (e.g. Axis 1) must be set in the digital output word:

- Digital output word:
  Bit 0: = 1 ⇒ "Axis 1: Rising edge of external zero mark 1 (X6-2, Pin 2)"

Note

Homing using an external zero mark requires 611U conformant mode to be selected.

Homing using encoder zero mark and homing output cam

System structure

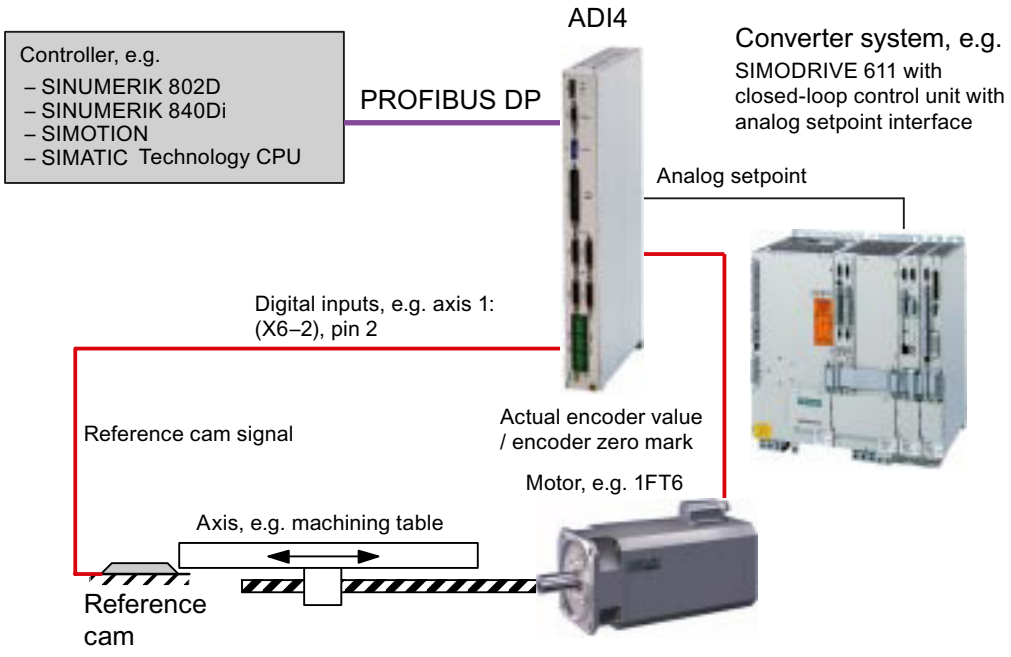


Figure 12-112 Basic system structure: Homing using encoder zero mark and external zero mark

Function

The homing output cam signal must be connected to a digital input on the ADI4 (X6-2, Pins 2 to 5). The homing output cam signal is processed in the controller as part of the homing operation.

Once the homing output cam signal is detected, the controller reduces the axis velocity to the homing approach velocity and requests the ADI4 to home to the next encoder zero mark. Once the request is detected, the ADI4 transmits the actual encoder value to the controller as the home position the next time it detects an encoder zero mark.

Without 611U conformant mode

No further measures are required.

With 611U conformant mode

The relevant signal for the axis to be homed (e.g. Axis 1) must be set in the digital output word:

- Digital output word:
  Bit 0: = 0 ⇒ "Axis 1: Zero mark of Encoder 1 (X4-1)"

Boundary conditions

Measuring input or on-the-fly measurement

ADI4 supports only measurement using a rising **or** falling edge of the measuring input. It is not possible to parameterize simultaneous measurement on a rising edge and a negative edge.

Actual speed

The actual speed value (PZD2/3: NIST_B) contained in Standard message frame 3 (see table "Message frame structure" in Section "Message frame type (Page 8356)") is not supported by the ADI4. The ADI4 always sends the value "0" as the actual speed value.

External encoder interface (encoders without an axis)

If encoders are connected to the ADI4 without at least one axis being assigned, i.e. ADI4 is used exclusively as an external encoder interface, a "Ready" signal (interface X6-1, Pin 10/11) will not be output. For information on the "Ready" signal, refer to Section "Interface (X6-1): Digital outputs (Page 8341)".

Error 20005

In conjunction with an ADI4 DP slave, the following message is displayed when the SIMOTION CPU switches from RUN to STOP mode:

- Error 20005: Device type: 1/2, log. address: x faulted. (Bit: 0, reason: 0x...)

The message can be ignored.

This message is automatically deleted by the system the next time there is a transition from STOP to RUN mode.

Homing using external zero mark

Homing always occurs at a rising edge, irrespective of which external zero mark edge (rising or falling) was selected for homing in SIMOTION.

12.5.4.6 Parameter assignment of the DP communication

Parameter assignment of the equidistant cyclic DP communication

Action steps

Once all the DP slaves have been inserted in the configuration and their function parameters have been assigned as described (Step 1), the parameters for the equidistant cyclic DP communication must then be assigned (Step 2).

Parameters are assigned to the equidistant cyclic DP communication in two steps, as well:

Step 1

- Activation of the equidistant DP cycle
- Equidistant master cyclic component T_{DX}

Step 2

- Equidistant DP cycle T_{DP}
- Master application cycle T_{MAPC}
- Actual value acquisition T_i
- Setpoint acceptance T_o

Note

When assigning parameters for DP communication, you must observe the boundary conditions applicable to the individual parameters (see the chapter "Boundary conditions (Page 8384)").

Activation of the equidistant DP cycle

Procedure

Double-click an ADI4 DP slave. In the station window of HW Config, the dialog box: "DP Slave Properties" opens.

Note

It is recommended that the equidistant DP cycle be enabled for all ADI4 DP slaves by enabling the equidistant DP cycle within the selected ADI4 DP slave, and then performing an alignment:

During an alignment, all values displayed in the "DP Slave Properties" dialog box, "Isochronous mode" tab are transferred to all DP slaves of the same type, ADI4 DP slave here, of the configuration.

Dialog: Start

Dialog: DP slave properties
Tab: Isochronous operation
Radio button: Synchronize drive to equidistant DP cycle ☑
Button: Alignment

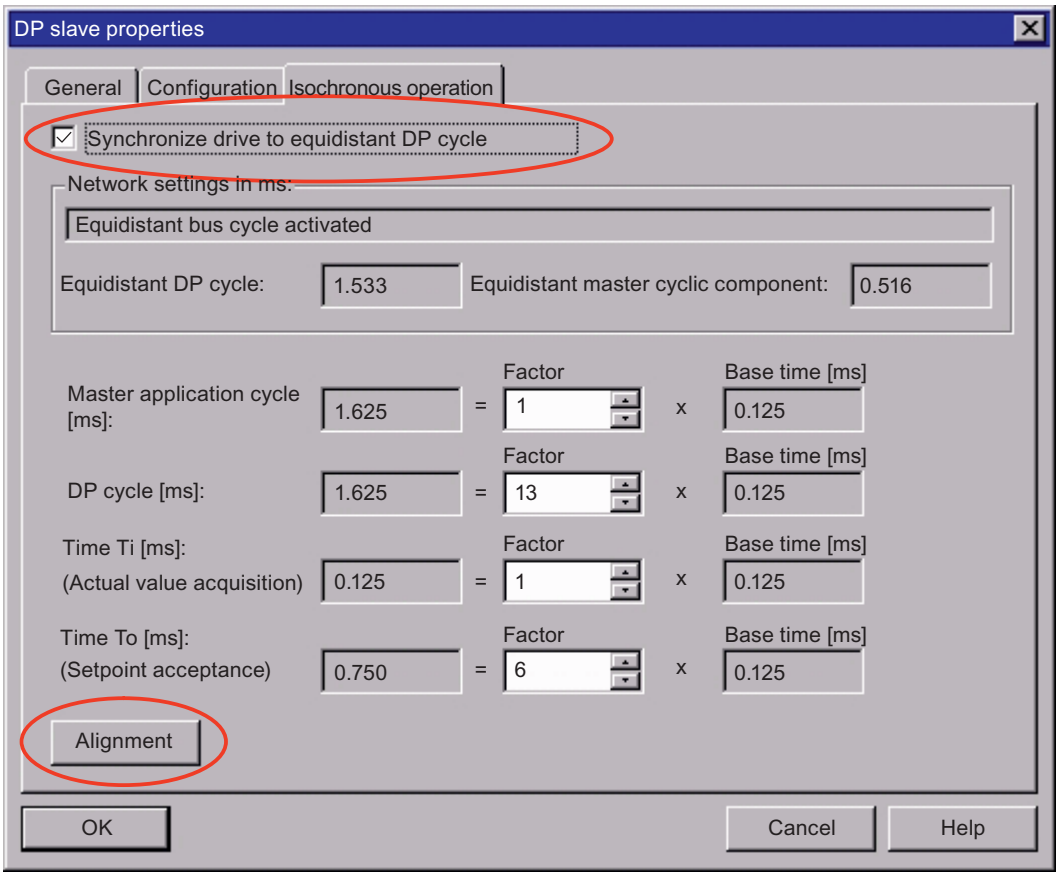


Figure 12-113 "DP Slave Properties" dialog box

Equidistant master cyclic component TDX

Procedure

Once synchronization to the equidistant DP cycle has been enabled for all DP slaves, the time required for the cyclic component of the DP communication must be recalculated.

The calculation is performed automatically by the DP master each time the equidistant bus cycle is enabled. This is performed in the following dialog box by selecting/clearing the "Activate equidistant bus cycle" check box.

Dialog: Continuation

Dialog box: DP slave properties
Tab: General
Group:: Node/Master System
Button: PROFIBUS...
Dialog box: Properties - PROFIBUS interface ADI4 ...
Tab: Parameters

Button: Properties...
Dialog box: PROFIBUS properties
Tab: Network settings
Button: Options...
Dialog box: Options
Tab: Equidistance
1. Radio button: Activate equidistant bus cycle ☐ (deselect)
2. Radio button: Activate equidistant bus cycle ☑ (select)

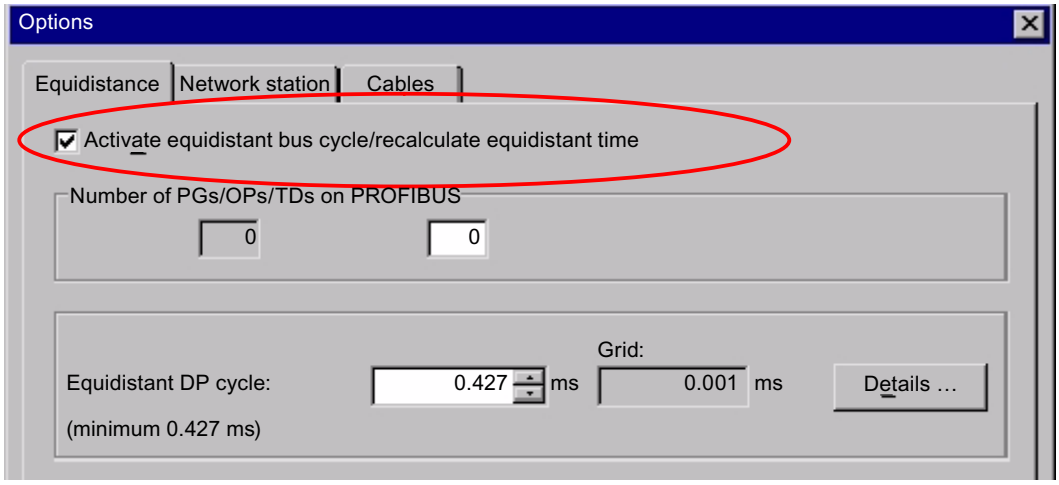


Figure 12-114 "Options" dialog box (excerpt)

Note

If there are different equidistant DP slave types (for example, SINAMICS drives, ADI4, SINAMICS I/O modules) in an S7 project, you must first perform the following two steps for each DP slave type:

1. Synchronize drive to equidistant DP cycle
2. Perform alignment

You can then continue to set the other parameters.

See also

Function parameters (Page 8361)

Equidistant DP cycle TDP

Procedure

When the cyclic component of the DP communication is calculated, the DP master automatically changes the value for the equidistant DP cycle to the minimum required time. This change must be undone by re-entering the intended value for the equidistant DP cycle.

(SINUMERIK 840D sl)

Equidistant DP cycle

For SINUMERIK 840D sl, an ADI4 module is connected via an external PROFIBUS (connection: DP1 (X126) or DP2/MPI (X136)). The DP cycle - with constant bus cycle time - of the external PROFIBUS must then be set the same as the position controller clock cycle parameterized in the NC or the equidistant DP cycle of the internal PROFIBUS.

Note

The DP cycle of the external PROFIBUS must be the same as that of the internal PROFIBUS:

Equidistant DP cycle (external PROFIBUS) = position controller cycle = equidistant DP cycle (internal PROFIBUS)

Connection to "NCU7x0.2 PN"

If an ADI4 module is connected to NCU module "NCU7x0.2 PN" then a module with the following version and higher must be used: **Article No. ...-0AA2**

Dialog: Continuation

Dialog box: Options
Tab: Equidistance
Entry field: Equidistant DP cycle = 2,000 ms (example value)
OK (close dialog box: Options)
OK (close dialog box: PROFIBUS properties)
OK (close dialog box: Properties - PROFIBUS interface ADI4 ...)

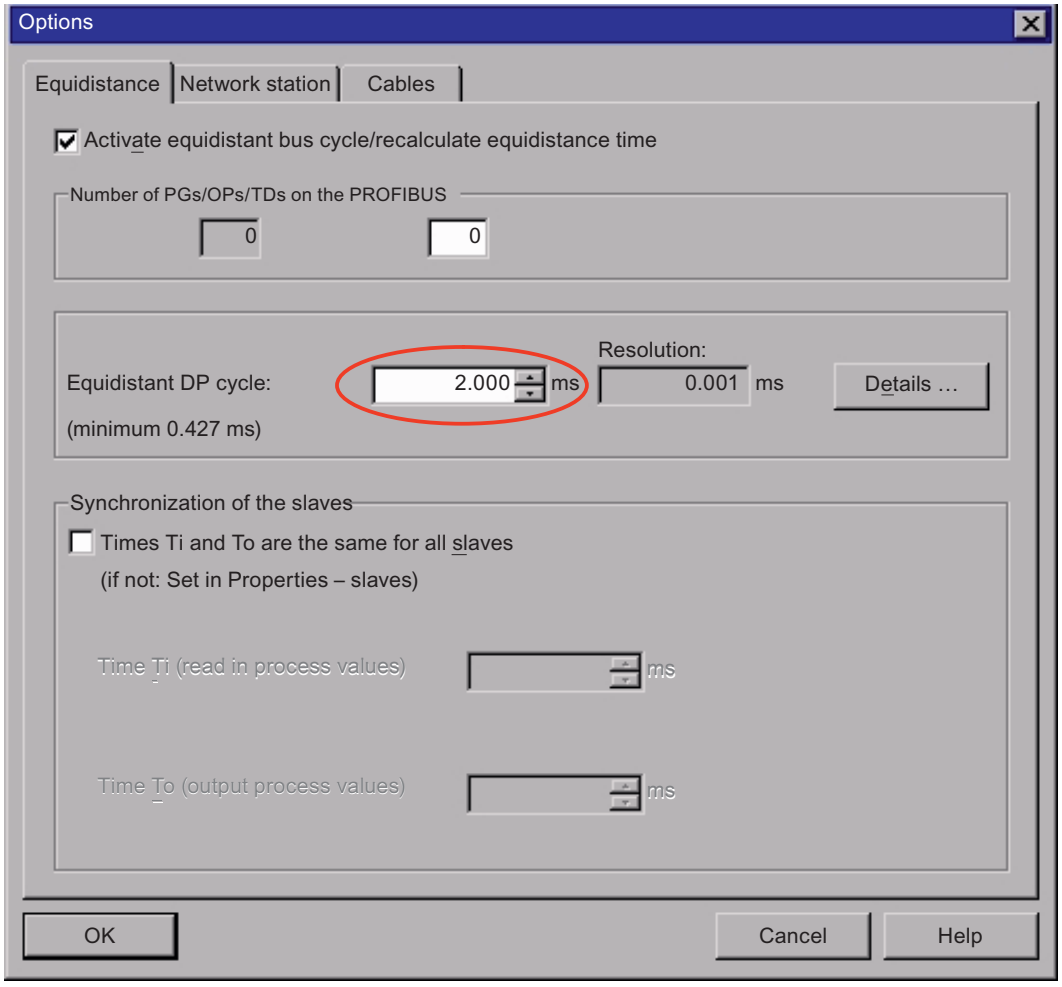


Figure 12-115 "Options" dialog box

DP cycle TDP

Procedure

In the "Factor" entry field of the "DP cycle (ms)", enter a value such that the resulting DP cycle is equal to the equidistant DP cycle.

Dialog: Start

Dialog: DP slave properties
Tab: Isochronous operation
Entry field: Factor = 16 (example value)

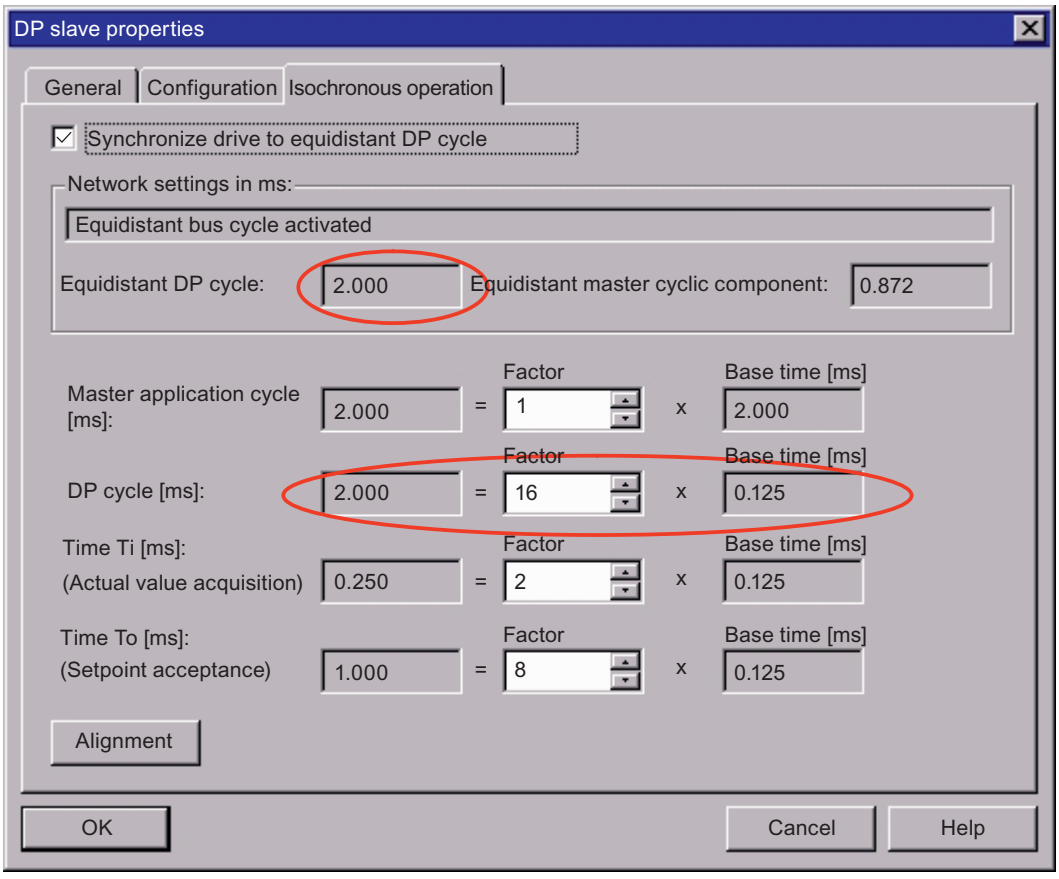


Figure 12-116 "DP Slave Properties" dialog box

Note

The DP cycle time ("DP cycle" parameter) of the ADI4 DP slave must be set to the same value as the DP cycle time setting for the DP master ("Equidistant DP cycle" parameter):

DP cycle = equidistant DP cycle

Master application cycle TMAPC

Introduction

The "Master application cycle T_{MAPC}" parameter specifies the integer ratio between the cycle time of the master application (position controller) and the equidistant DP cycle.

Note

The ratio between master application cycle T_{MAPC} and DP cycle time T_{DP} **must** be 1:1.

Procedure

In the entry field for the factor of the "Master application cycle [ms], enter a value such that the required time ratio is achieved.

Dialog: Continuation

Dialog: DP slave properties
Tab: Isochronous operation
Entry field: Factor = 1

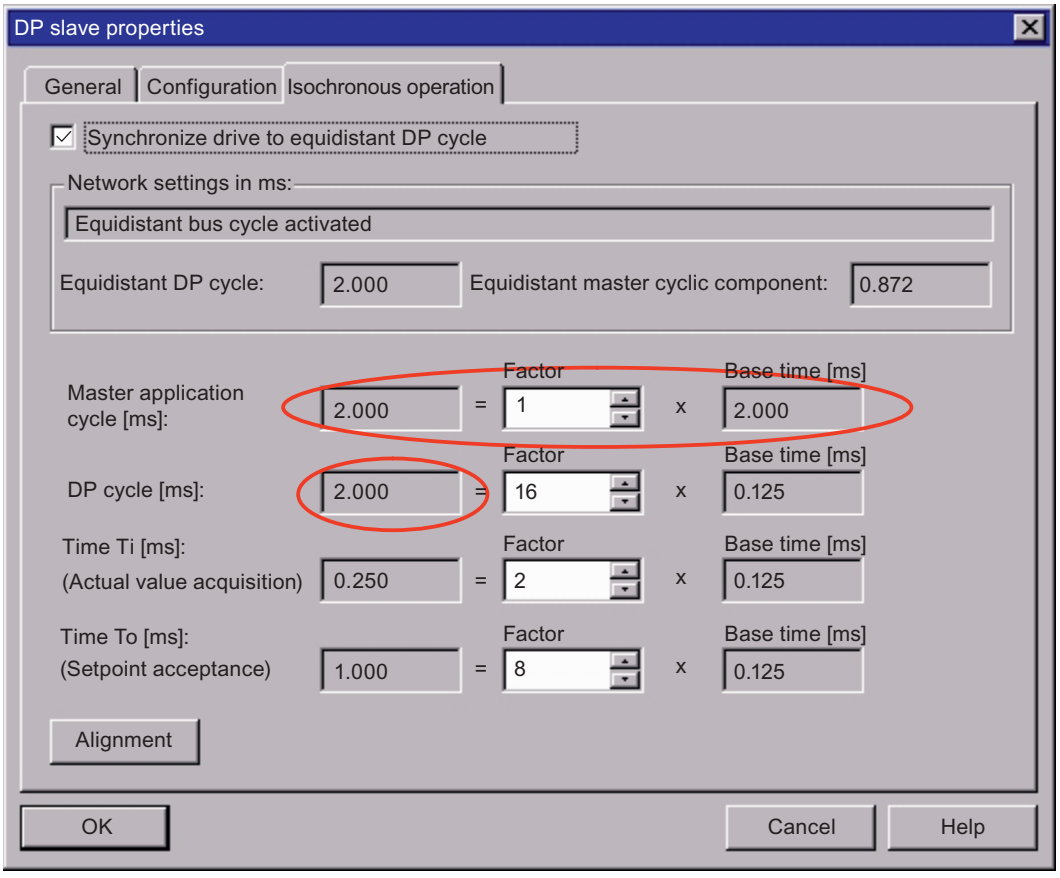


Figure 12-117 "DP Slave Properties" dialog box

Execution scheme T_{MAPC} : T_{DP} = 1 : 1

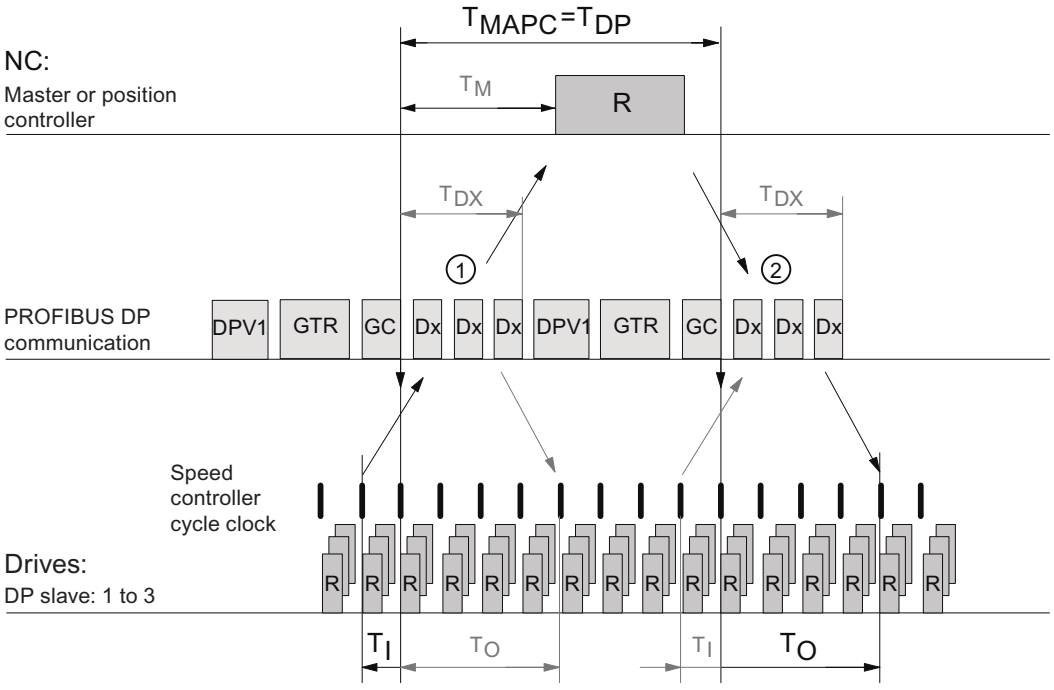


Figure 12-118 Example: Optimized DP cycle with T_{MAPC} : T_{DP} = 1 : 1

T_{MAPC}	Master application cycle: Position control cycle clock
T_{DP}	DP cycle time: DP cycle time
T_{DX}	Data exchange time: Total transfer time for all DP slaves
T_M	Master time: Offset of the start time for NC position control
T_I	Input time: Time of the actual value acquisition. The actual values are transferred to the DP master in the **next** DP cycle.
T_O	Output time: Time of the setpoint acceptance. The setpoints were generated by the DP master application in the **previous** DP cycle.
GC	Global control message frame (broadcast message frame) for cyclic synchronization of the equidistance between the DP master and DP slaves
R	Processing time for speed or position controller
Dx	User data exchange between the DP master and DP slaves
DPV1	After cyclic communication, an acyclic service is sent, if the token holding time T_{TH} has not yet been exceeded. T_{TH} is calculated by the engineering system.
GTR	GAP, TOKEN, RESERVE: GAP: An attempt is made during GAP to accept new active stations. TOKEN: The token passing is either to itself or other masters. RESERVE: The reserve is used as an "Active break" for the station to send the token to itself until the equidistant cycle expires.

① The actual values for the current DP-Cycle/position control cycle clock are transferred from the DP slave drives to the NC position controller.

② The setpoints computed by the NC position controller are transferred to the DP slave drives.

Actual value acquisition Ti

Introduction

The "Actual value acquisition T_i" parameter specifies the time when an ADI4 DP slave reads in the actual values (actual position value).

It is recommended to specify the same time for the actual value acquisition T_i for all ADI4 DP slaves. Special attention must be paid to this if axes of different ADI4 DP slaves move according to interpolation on a common path.

Procedure

Enter the required value in the entry field for the factor of the actual value acquisition.

Dialog box: Start

Dialog box: DP slave properties
Tab: Isochronous mode
Entry field: Factor = 2 (example value)

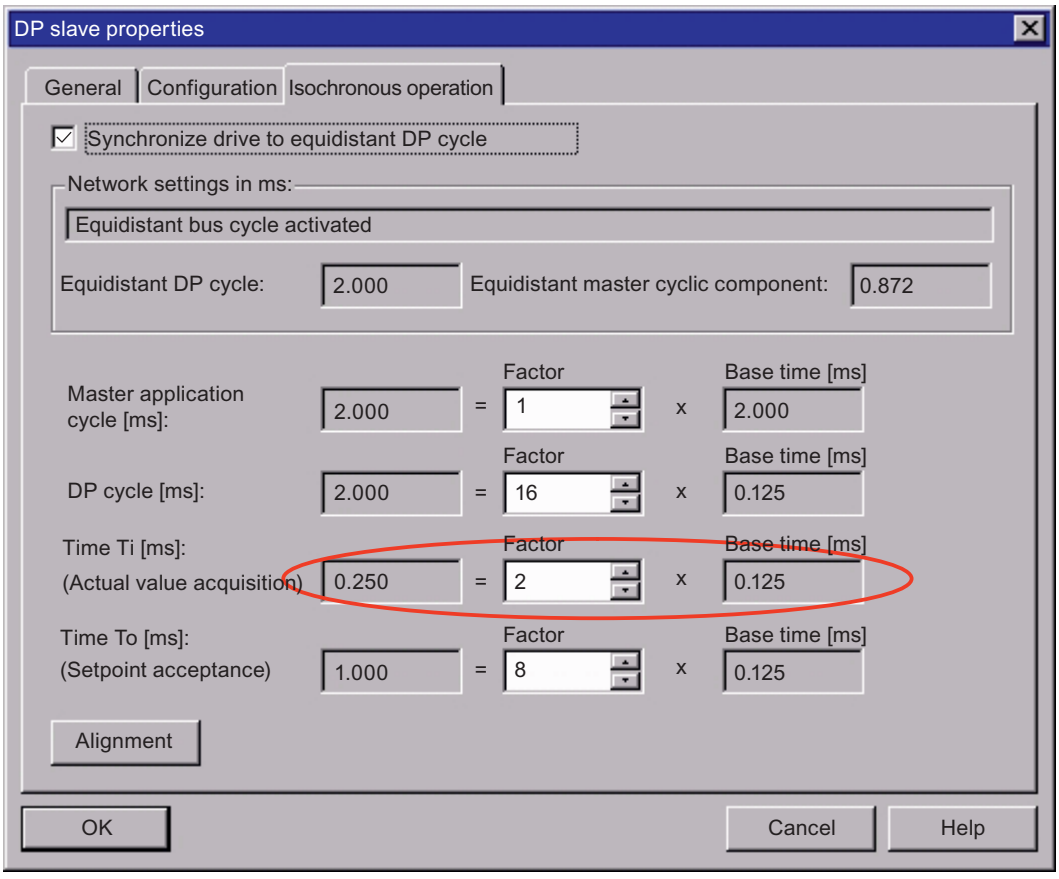


Figure 12-119 "DP Slave Properties" dialog box

Note

The following condition must be observed for the time of actual value acquisition T_i:

Base time ≤ actual value acquisition ≤ DP cycle

Possible times for Ti are: 250 µs ≤ Ti ≤ 625 µs

Setpoint acceptance To

Introduction

The "Setpoint acceptance T_o" parameter specifies the time when the ADI4 DP slave receives the speed setpoint from the position controller.

It is recommended that setpoint acceptance time T_o be the same for all ADI4 DP slaves, particularly if axes are interpolated together.

Procedure

Enter the required value in the entry field for the factor of the setpoint acceptance.

Dialog: Start

Dialog: DP slave properties
Tab: Isochronous operation
Entry field: Factor = 8 (example value)

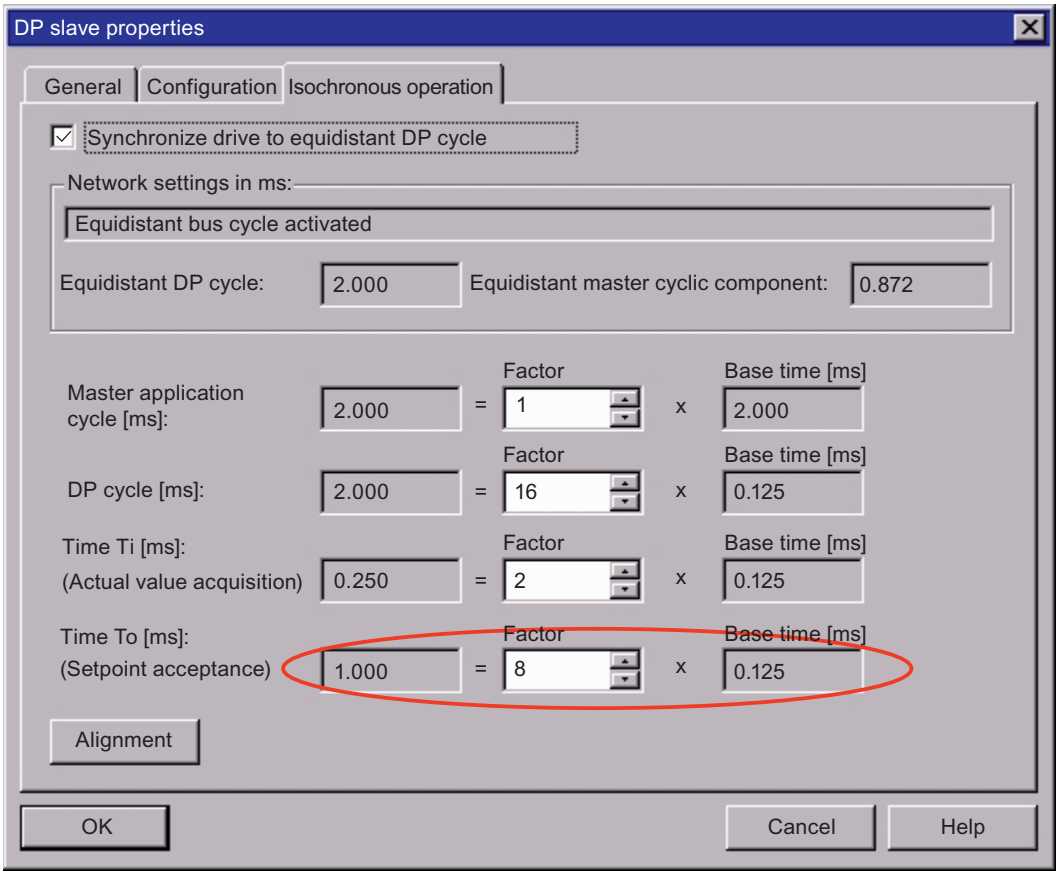


Figure 12-120 "DP Slave Properties" dialog box

Note

The following condition must be observed for the time of setpoint acceptance T_o:

Equidistant master cyclic component + base time ≤ setpoint acceptance ≤ DP cycle

Alignment

Procedure

The alignment is used to transfer the values of the current ADI4 DP slave displayed in the "Isochronous operation" tab to all the other ADI4 DP slaves of the configuration.

Dialog: End

Dialog: DP slave properties
Tab: Isochronous operation
Button: Alignment
OK

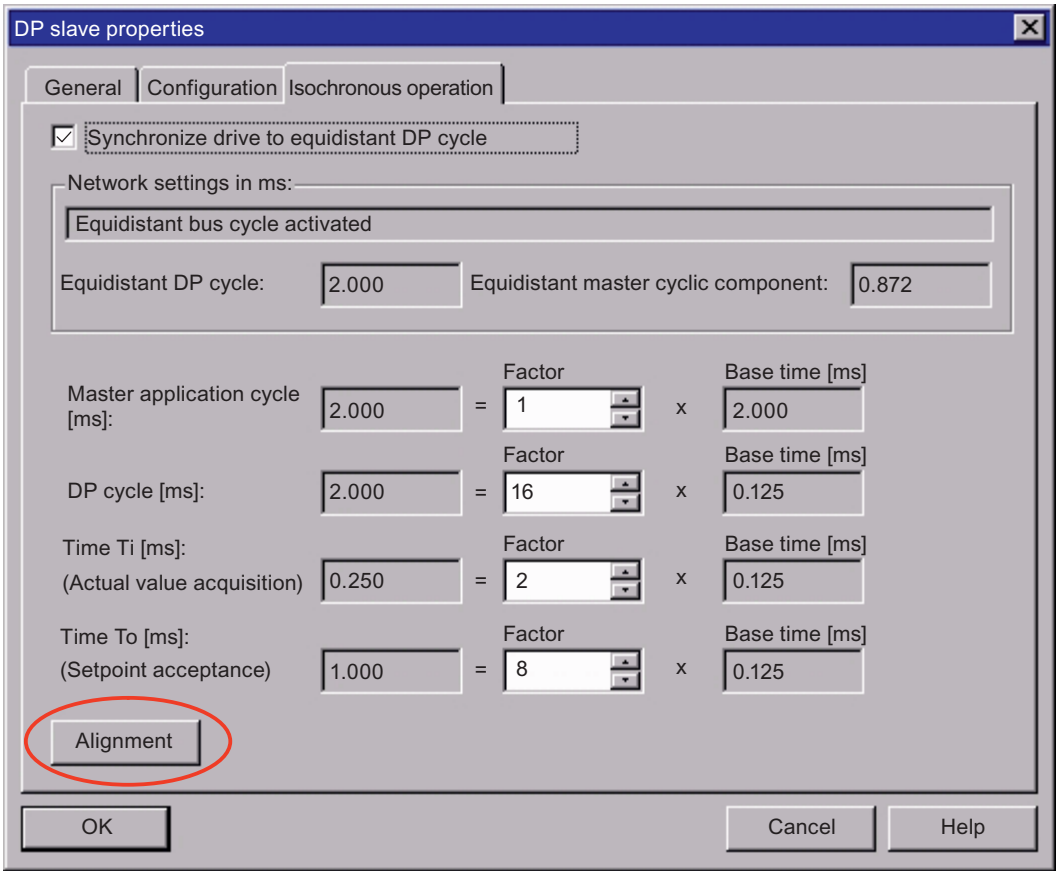


Figure 12-121 "DP Slave Properties" dialog box

Note

If an S7 project includes different equidistant DP slave types (for example, SINAMICS drives, ADI4, SINAMICS I/O modules), the following parameter settings must be made for each DP slave type as described above and an alignment must be performed:

- Equidistant DP cycle T_{DP}
- Master application cycle T_{MAPC}
- Actual value acquisition T_i
- Setpoint acceptance T_o

The alignment only transfers the values displayed in the "Isochronous operation" tab to the DP slaves **of the same** type.

The alignment concludes the parameter assignment of all ADI4 DP slaves with respect to the DP communication.

Boundary conditions

ADI4 with Article No. 6FC5 211-0BA01-0AA1 or higher

The following boundary conditions must be observed during the final parameter assignment of the isochronous DP cycle in conjunction with ADI4 with Article No. 6FC5 211-0BA01-0AA1 and higher:

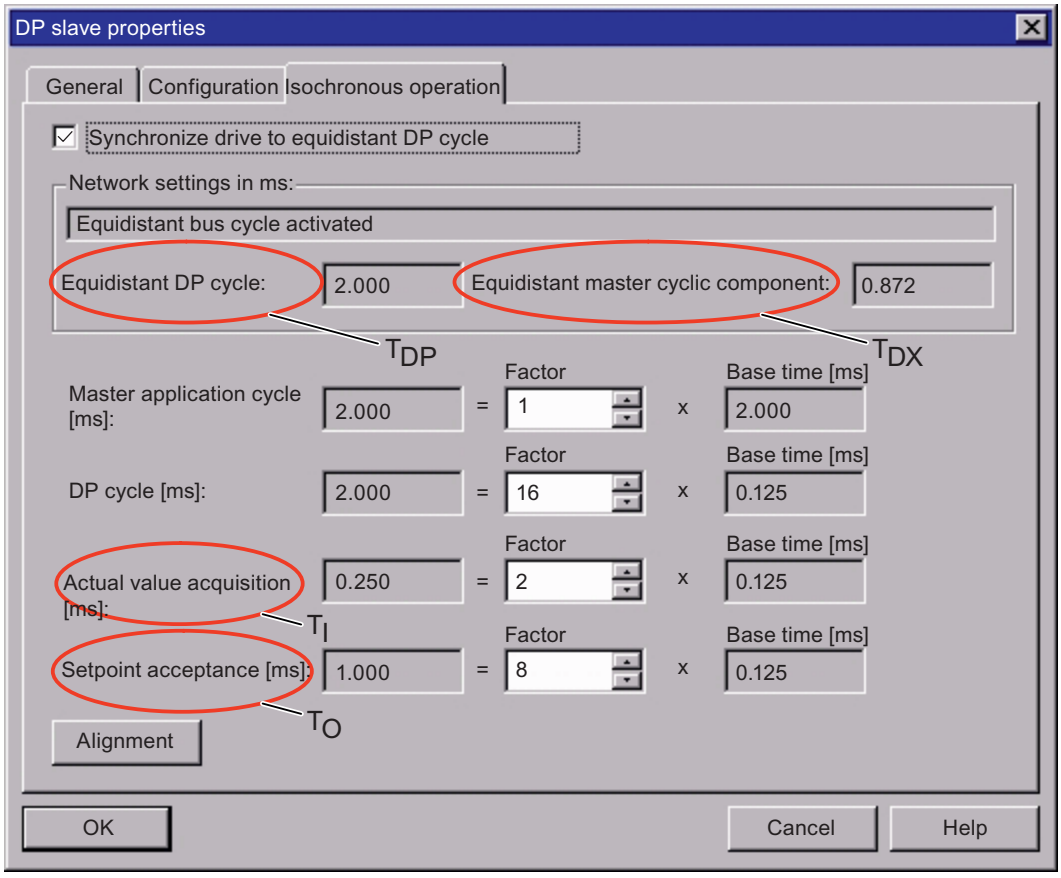


Figure 12-122 Excerpt of "DP Slave Properties" dialog box

1. Equidistant DP cycle (T_{DP})
   $\mathbf{T}_{DP} = 2 * n * 125\ \mu s$;
   with $n \geq 4$ (⇒ Minimum $T_{DP} = 1$ ms)
2. Setpoint acceptance (T_o)
   $(TDX + 125\ \mu s) \leq \mathbf{T}_o < T_{DP}$;
   with rounded variable $TDX = T_{DX}$, rounded to an integer multiple of 125 µs
3. Actual value acquisition (T_i)
   $(250\ \mu s \leq \mathbf{T}_i \leq 625\ \mu s)$ and $(\mathbf{T}_i \neq 500\ \mu s)$
4. T_i and T_o cannot be in the same 125 µs cycle clock
   $\Delta T \neq 0$; with $\Delta T = T_{DP} - \mathbf{T}_i - \mathbf{T}_o$

5. If $\mathbf{T}_o = (T_{DP} - 125\ \mu s)$
   Then for T_i, the following must apply: $\mathbf{T}_i > 3 * 125\ \mu s$
6. If $\mathbf{T}_o = (TDX + 125\ \mu s)$
   Then for $(T_i + T_o)$, the following must apply: $(\mathbf{T}_i + \mathbf{T}_o) \neq (T_{DP} + 125\ \mu s)$

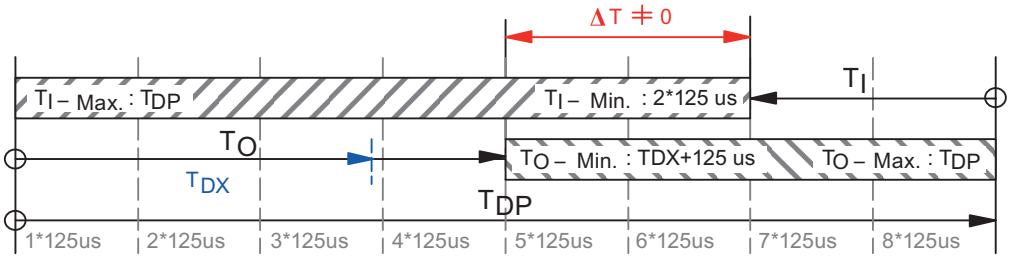


Figure 12-123 Graphic illustration of the boundary conditions

Typical parameter values

Equidistant DP cycle (T_{DP})	2.000 ms	3.000 ms
Actual value acquisition (T_i)	0.250 ms	0.250 ms
Setpoint acceptance (T_o)	1.250 ms	1.000 ms

Note

ADI4 DP slaves

- Article number: 6FC5 211-0BA01-0AA**0**
- Article number: 6FC5 211-0BA01-0AA**1** or ...-0AA**2**

exhibit a different behavior for a parameter assignment for actual value acquisition (T_i) and setpoint acceptance (T_o) deviating from the boundary conditions indicated above.

- ADI4 DP slave with Article No. ...-0AA**0**
  If parameters are assigned that deviate from the boundary conditions stated above, they are ignored by this ADI4 DP slave as the parameters have fixed internal values. The ADI4 DP slave establishes cyclic communication with the DP master using the preset values (and not the assigned parameters) without an error message.
- ADI4 DP slave with Article No. ...-0AA**1** or ...-0AA**2**
  If a parameter assignment that deviates from the boundary conditions stated above is downloaded to this ADI4 DP slave, the ADI4 DP slave does **not** establish cyclic communication with the DP master.

12.5.5 Commissioning

12.5.5.1 Wiring of drive ready signals

In order to use the S7 function block FB 401 (MC_POWER) to switch on a drive connected to the ADI4, the drive must signal its readiness. For this purpose, the ready signal of the drive must be wired with one of the ready signal inputs "Drive Ready" Axis x (DRVx_RDY), Interface (X6-2) of the ADI4.

The ready signal must continue to exist at the ADI4 input even after the drive is switched on. If the signal is cleared, the drive will stop.

Note

Drives without ready signal

If a ready signal is not available for a drive, the corresponding digital ready signal input "Drive Ready" Axis x (DRVx_RDY) of the ADI4 can be assigned statically with 24 V. The disadvantage of this is that the S7 function block FB 401 (MC_POWER) can no longer detect a drive failure. FB 401 returns the status "TRUE" at its output even after a drive failure.

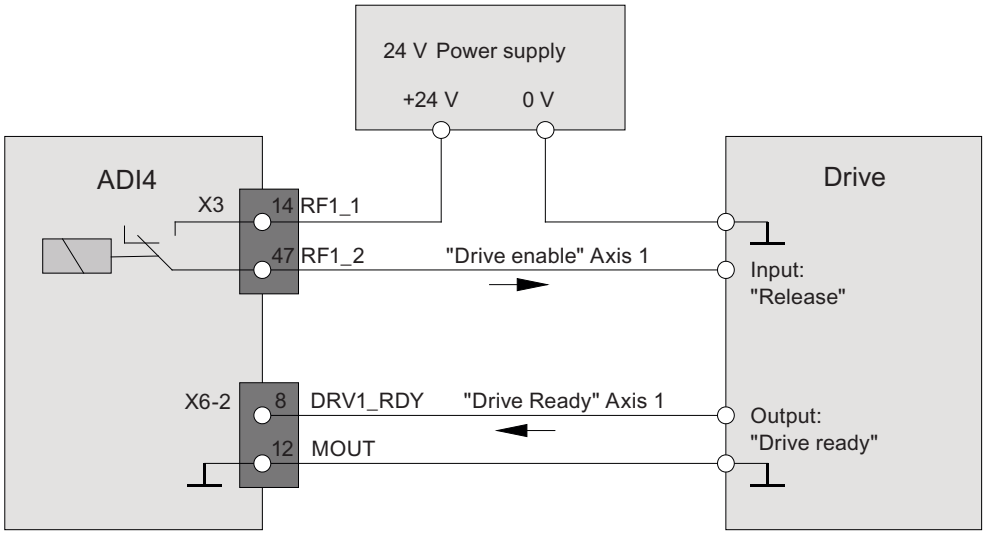


Figure 12-124 Drive enable for Axis 1 (principle)

12.5.5.2 Absolute encoder (SSI), single-turn

General

The parameter assignment of a single-turn absolute encoder (SSI) illustrated in the following example is performed using the "External encoder" technology object.

If the encoder belongs permanently to an axis, the "Axis" technology object is used to assign parameters.

Note

In conjunction with ADI4, SIMOTION only supports encoders with 13-bit and 25-bit message frames.

Encoder data

The encoder used in this example is a Siemens encoder, Article No. 6FX2 001-5HS12 with the following data:

Parameters	Value
Encoder type	Rotary
Encoder type	Cyclic absolute encoder (SSI)
Increments/revolution	4096
User data length	12
Message length	13
Message frame format	PINETREE
Actual value protocol format	GRAY

Settings in STEP 7, HW Config

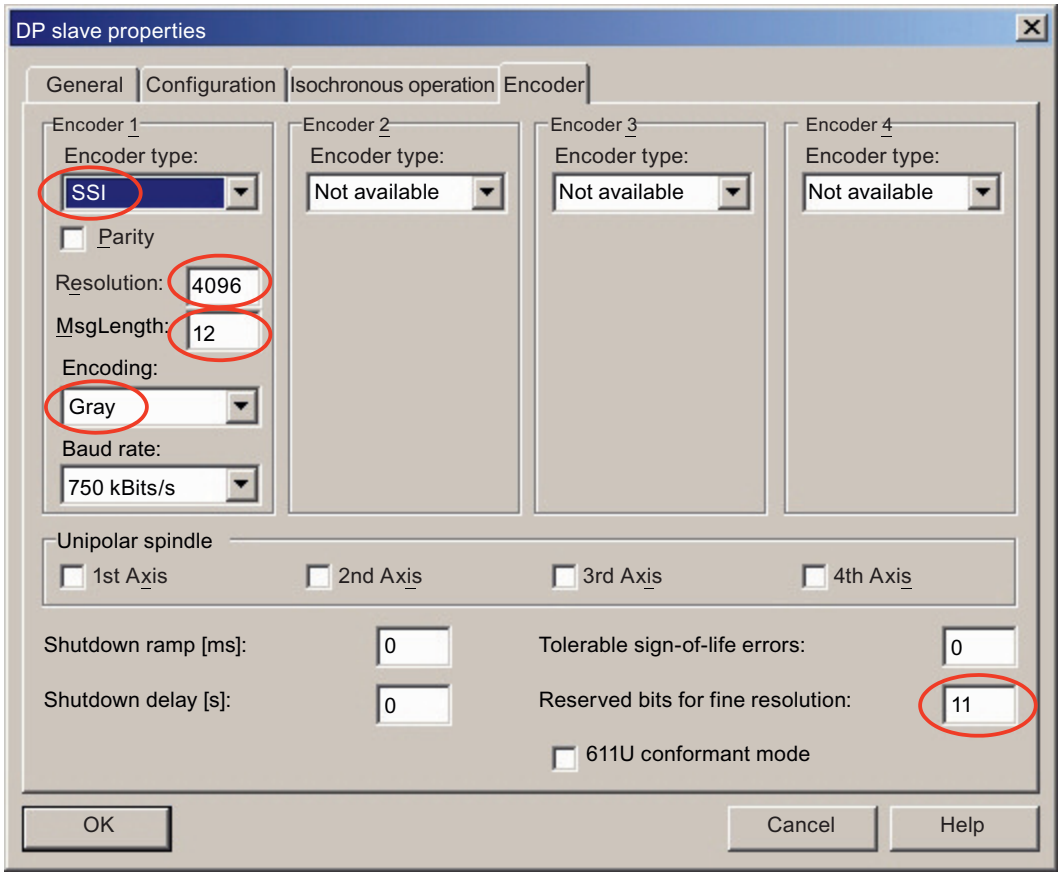


Figure 12-125 Encoder settings in STEP 7, HW Config

Settings	Value
Resolution	Encoder parameters: "Increments/revolution (resolution)"
MsgLength	Encoder parameters: "User data length"
Encoding	Encoder parameters: "Actual value protocol format"
Reserved bits for fine resolution	**Note**: The value is permanently set by ADI4.

SIMOTION/TCPU settings

After you have created a new encoder in the project navigator of SIMOTION SCOUT under "EXTERNAL ENCODERS" and have assigned the technology object parameters in the displayed dialog boxes, e.g. "Axis Type" and "Units", the encoder data must be entered in the "Encoder Assignment" and "Encoder - Data" dialog boxes.

1. Settings in the "Encoder Assignment" dialog box

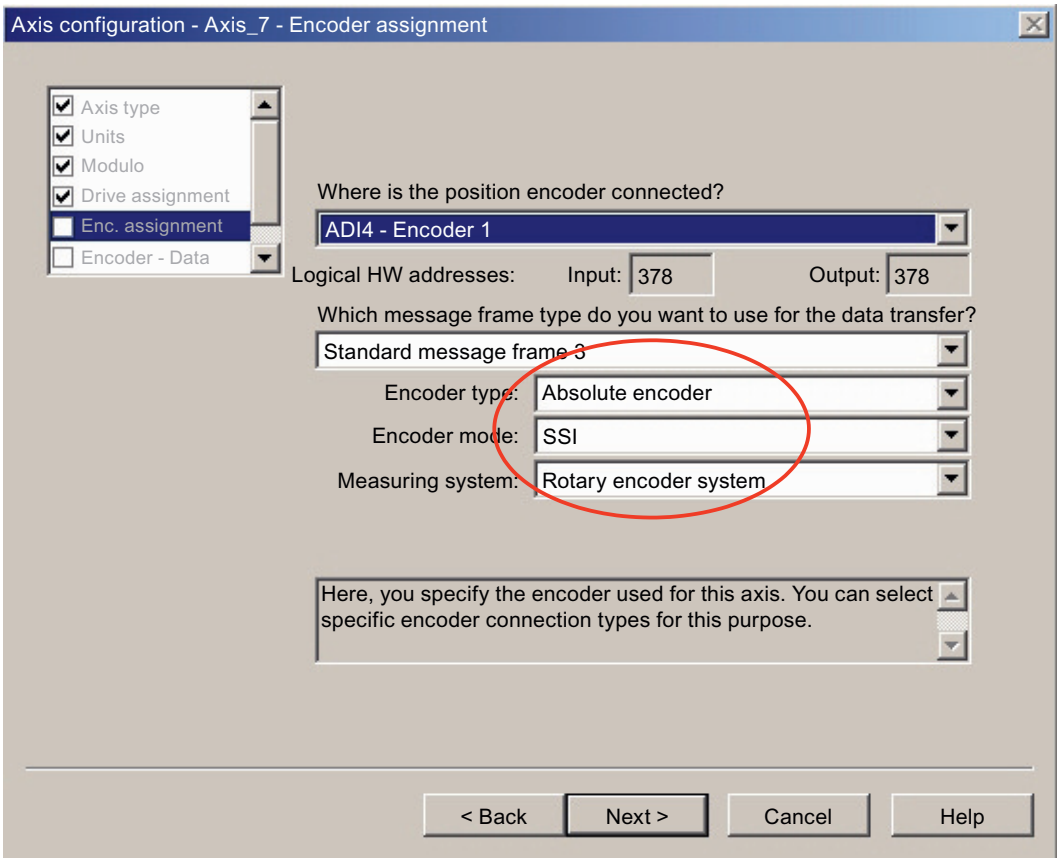


Figure 12-126 Encoder assignment settings

2. Settings in the "Encoder - Data" dialog box

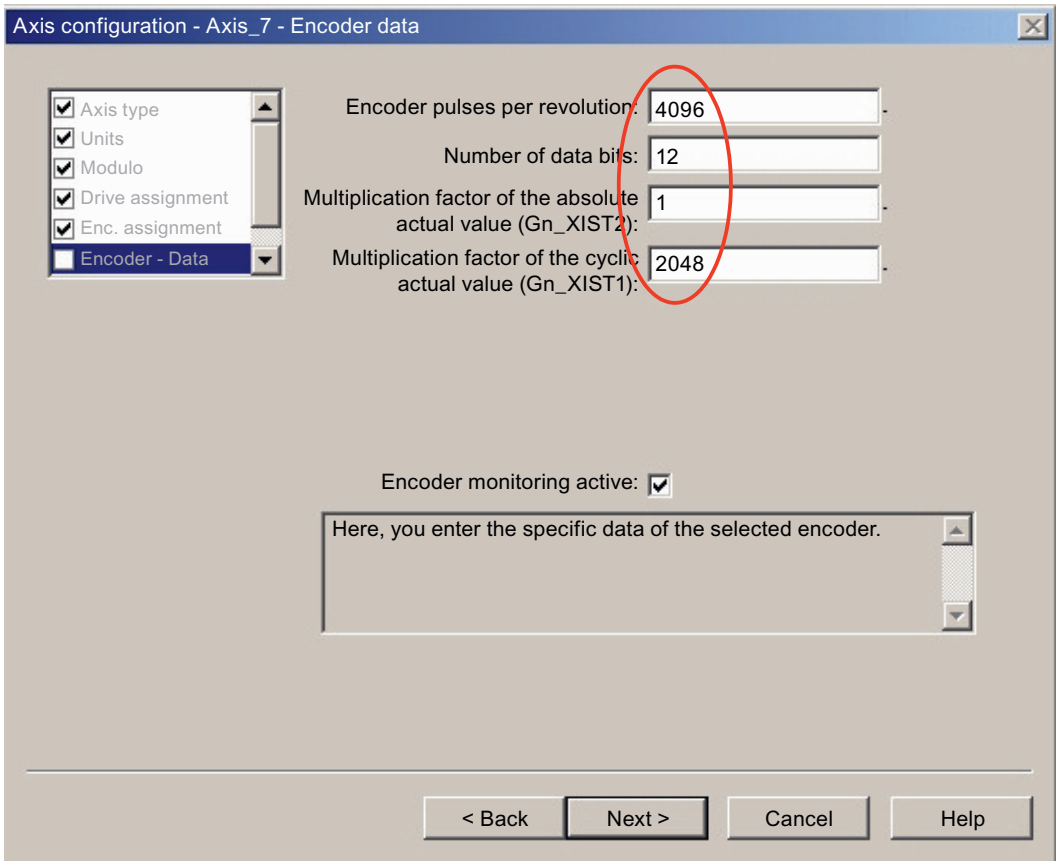


Figure 12-127 Encoder data settings

Settings:

Encoder data settings	Value
Encoder pulses per revolution	Encoder parameters: "Increments/revolution (resolution)"
Number of data bits	Encoder parameters: "User data length of the encoder"
Multiplication factor of the absolute actual value (Gn_XIST2)	1 (The absolute actual value (Gn_XIST2) is taken over directly)
Multiplication factor of the cyclic actual value (Gn_XIST1)	2^(STEP 7, HW Config: "Reserved bits for fine resolution") = 2^{11} = 2048

Overview: SIMOTION encoder parameters

Parameters: TypeOfAxis > Encoder_1 >	Value
encoderTyp	SENSOR_CYCLIC_ABSOLUT
encoderMode	SSI_MODE
encoderSystem	ROTATORY_SYSTEM
AbsEncoder > absResolution	4096
AbsEncoder > absDataLength	12
AbsEncoder > absResolutionMutiplierAbsolute	1 [1)]

Parameters: TypeOfAxis > Encoder_1 >	Value
AbsEncoder > absResolutionMutiplierCyclic	2048 [2]
[1] After the ramp-up of the controller, the absolute actual value of the encoder is read out once.	
[2] 2048 = 2^{11}; corresponds to STEP 7, HW Config: "Reserved bits for fine resolution" = 11	

12.5.5.3 Absolute encoder (SSI), multiturn

General

The parameter assignment of a multiturn absolute encoder (SSI) illustrated in the following example is performed using the "External encoder" technology object.

If the encoder belongs permanently to an axis, the "Axis" technology object is used to assign parameters.

Note

In conjunction with ADI4, SIMOTION only supports encoders with 13-bit and 25-bit message frames.

Encoder data

The encoder used in this example is a Siemens encoder, Article No. 6FX2 001-5HS24 with the following data:

Encoder parameters	Value
Encoder type	Rotary
Encoder type	Cyclic absolute encoder (SSI)
Increments/revolution	8192
User data length	25
Message length	25
Message frame format	PINETREE
Actual value protocol format	GRAY

Settings in STEP 7, HW Config

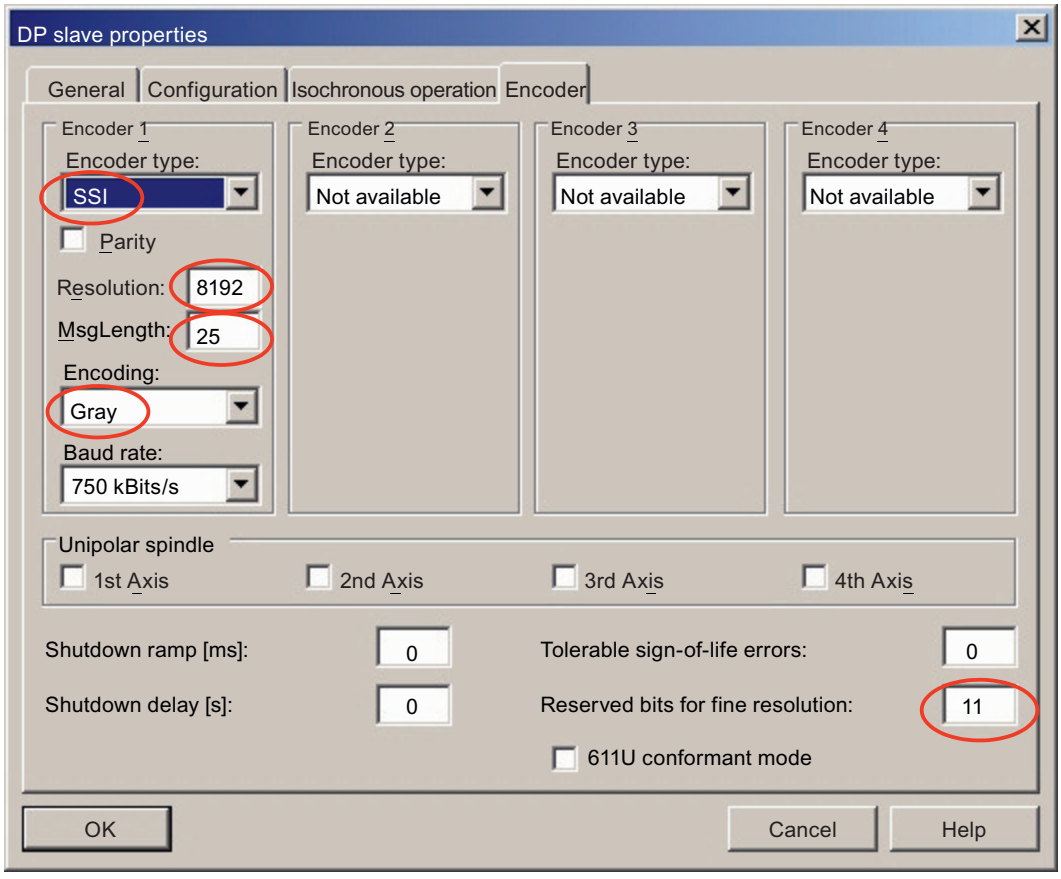


Figure 12-128 Encoder settings in STEP 7, HW Config

STEP 7 settings, HW Config	Corresponding encoder parameters
Encoder type	Encoder parameters: "Encoder type"
Resolution	Encoder parameters: "Increments/revolution (resolution)"
MsgLength	Encoder parameters: "User data length"
Encoding	Encoder parameters: "Actual value protocol format"
Reserved bits for fine resolution	**Note**: The value is permanently set by ADI4.

SIMOTION/TCPU settings

After you have created a new encoder in the project navigator of SIMOTION SCOUT under "EXTERNAL ENCODERS" and have assigned the technology object parameters in the displayed dialog boxes, e.g. "Axis Type" and "Units", the encoder data must be entered in the "Encoder Assignment" and "Encoder - Data" dialog boxes.

1. Settings in the "Encoder Assignment" dialog box

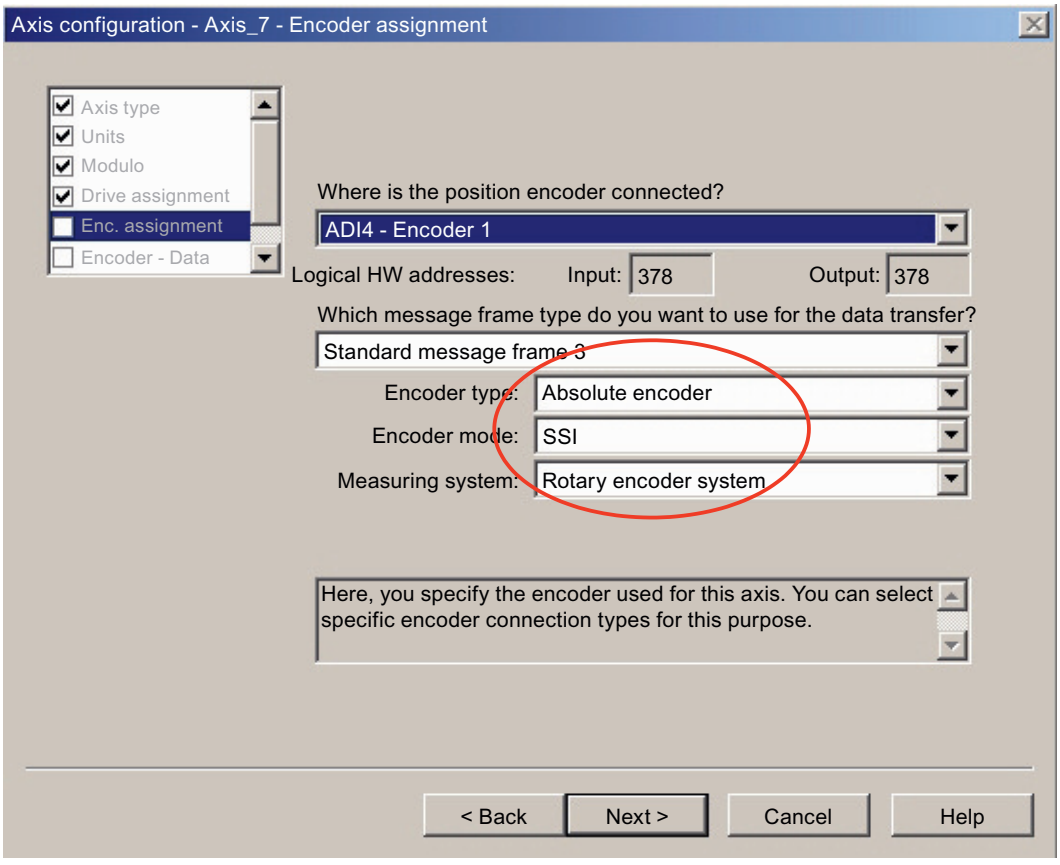


Figure 12-129 Encoder assignment settings

2. Settings in the "Encoder - Data" dialog box

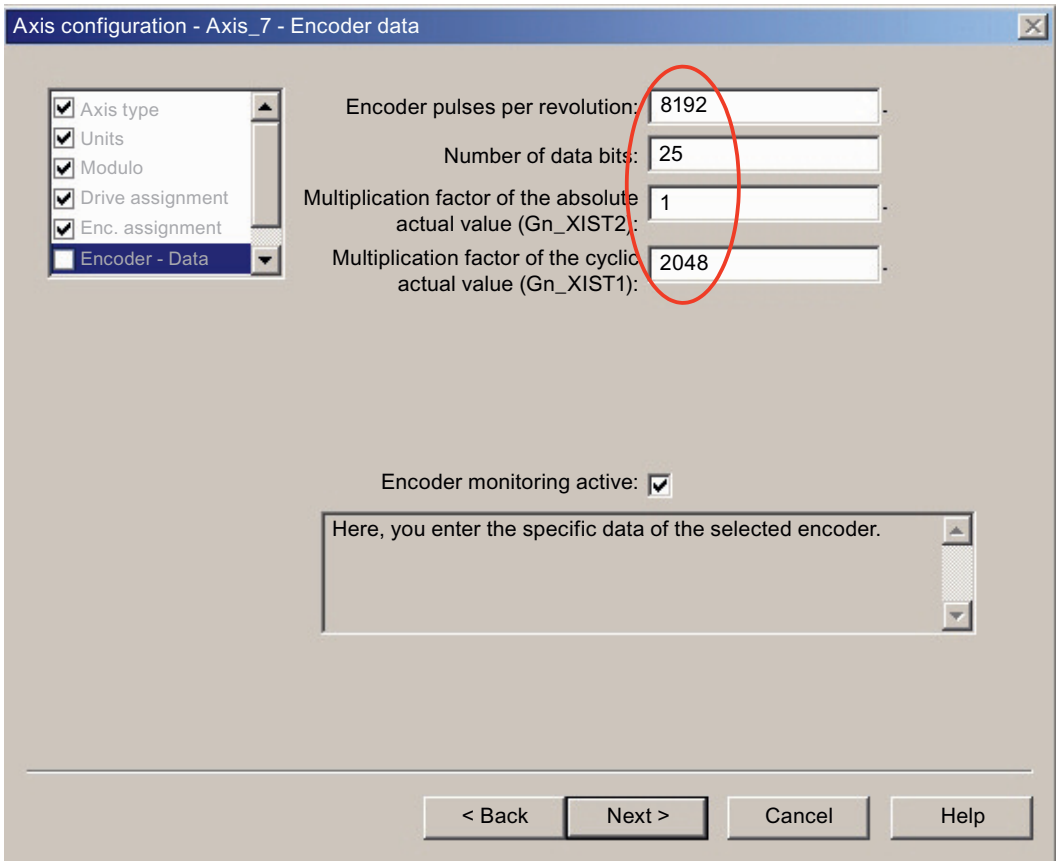


Figure 12-130 Encoder data settings

Settings:

Encoder data settings	Value
Encoder pulses per revolution	Encoder parameters: "Increments/revolution (resolution)"
Number of data bits	Encoder parameters: "User data length of the encoder"
Multiplication factor of the absolute actual value (Gn_XIST2)	1 (The absolute actual value (Gn_XIST2) is taken over directly)
Multiplication factor of the cyclic actual value (Gn_XIST1)	2^(STEP 7, HW Config: "Reserved bits for fine resolution") = 2^{11} = 2048

Overview: SIMOTION encoder parameters

Parameters: TypeOfAxis > Encoder_1 >	Value
encoderTyp	SENSOR_CYCLIC_ABSOLUT
encoderMode	SSI_MODE
encoderSystem	ROTATORY_SYSTEM
AbsEncoder > absResolution	8192
AbsEncoder > absDataLength	25
AbsEncoder > absResolutionMutiplierAbsolute	1 [1)]

Parameters: TypeOfAxis > Encoder_1 >	Value
AbsEncoder > absResolutionMutiplierCyclic	2048 [2]
[1] After the ramp-up of the controller, the absolute actual value of the encoder is read out once.	
[2] 2048 = 2^{11}; corresponds to STEP 7, HW Config: "Reserved bits for fine resolution" = 11	

12.5.5.4 Incremental encoder (TTL)

General

The parameter assignment of an incremental encoder (TTL) illustrated in the following example is performed using the "External Encoder" technology object.

If the encoder belongs permanently to an axis, the "Axis" technology object is used to assign parameters.

Note

In conjunction with ADI4, SIMOTION only supports encoders with 13-bit and 25-bit message frames.

Encoder data

The encoder used in this example is a Siemens encoder, Article No. 6FX2 001-2GB02 with the following data:

Parameters	Value
Encoder type	Rotary
Encoder type	Incremental encoder
Increments/revolution (resolution)	1024
Encoder mode	RECTANGLE_TTL

Settings in STEP 7, HW Config

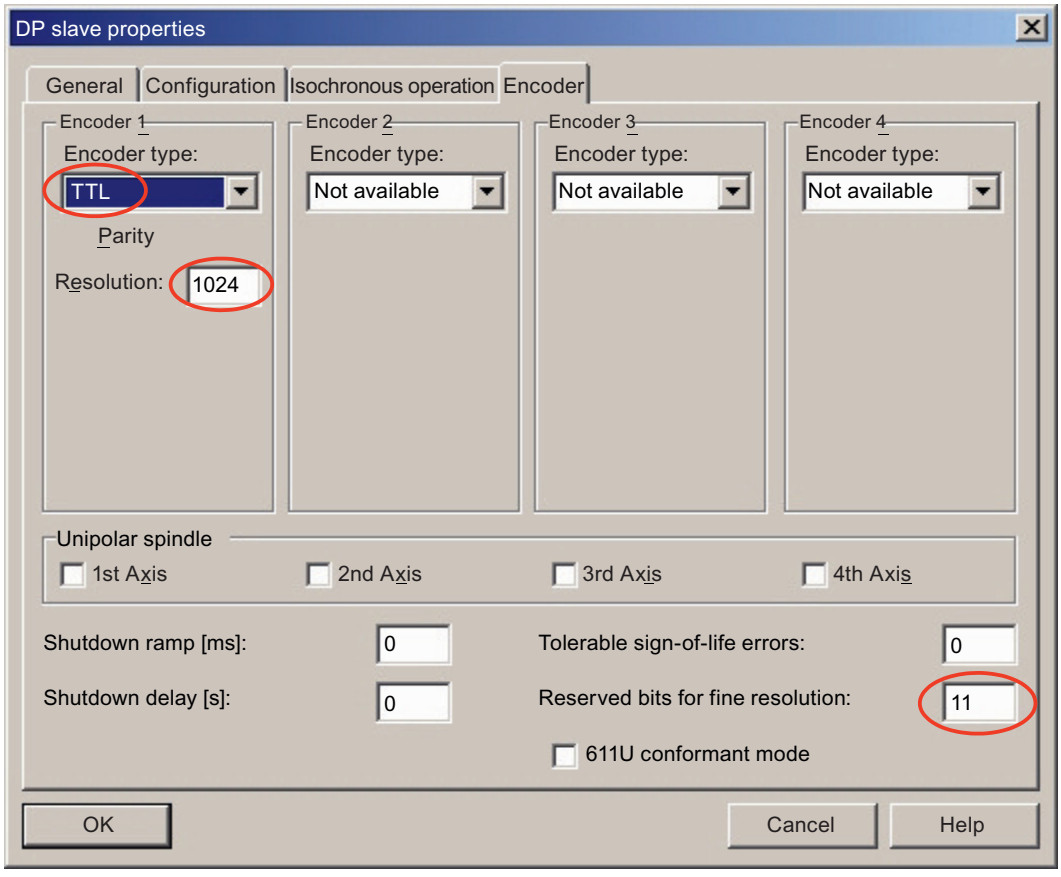


Figure 12-131 Encoder settings in STEP 7, HW Config

Settings:

Settings	
Resolution	Encoder parameters: "Increments/revolution (resolution)"
Reserved bits for fine resolution	The value is permanently set by ADI4.

Settings: SIMOTION/TCPU

After you have created a new encoder in the project navigator of SIMOTION SCOUT under "EXTERNAL ENCODERS" and have assigned the technology object parameters in the displayed dialog boxes, e.g. "Axis Type" and "Units", the encoder data must be entered in the "Encoder Assignment" and "Encoder - Data" dialog boxes:

1. Settings in the "Encoder Assignment" dialog box

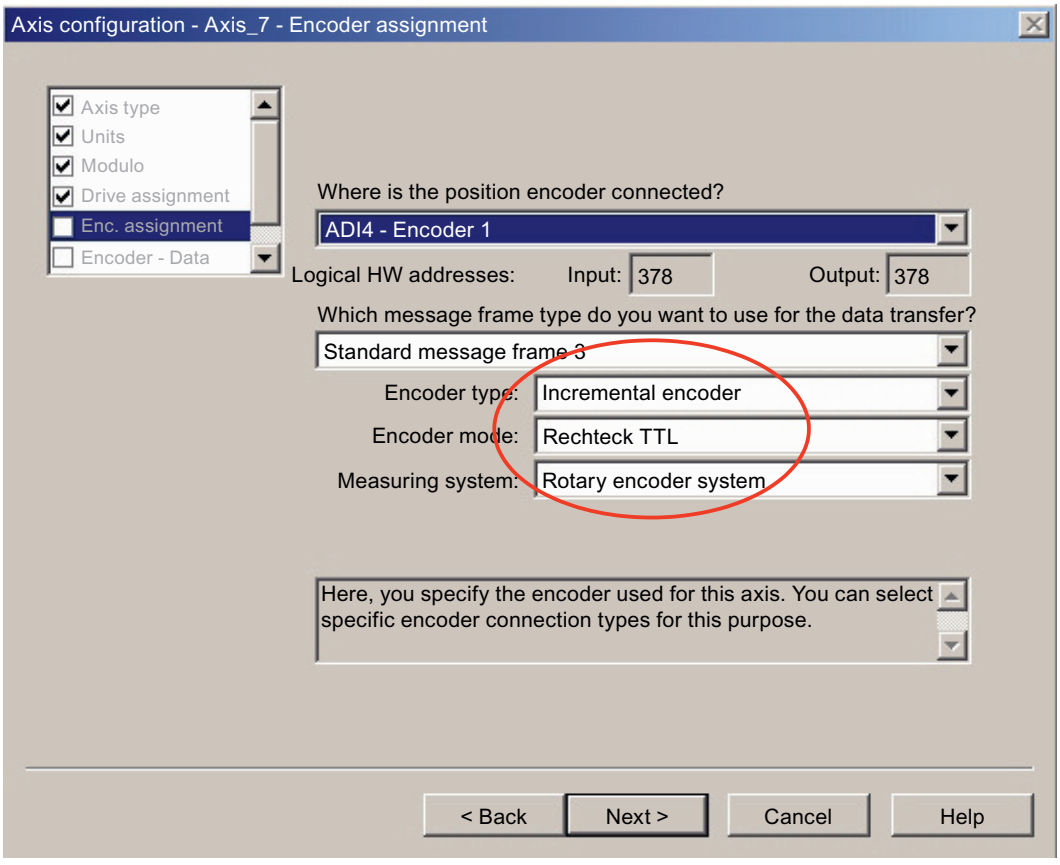


Figure 12-132 Encoder assignment settings

2. Settings in the "Encoder - Data" dialog box

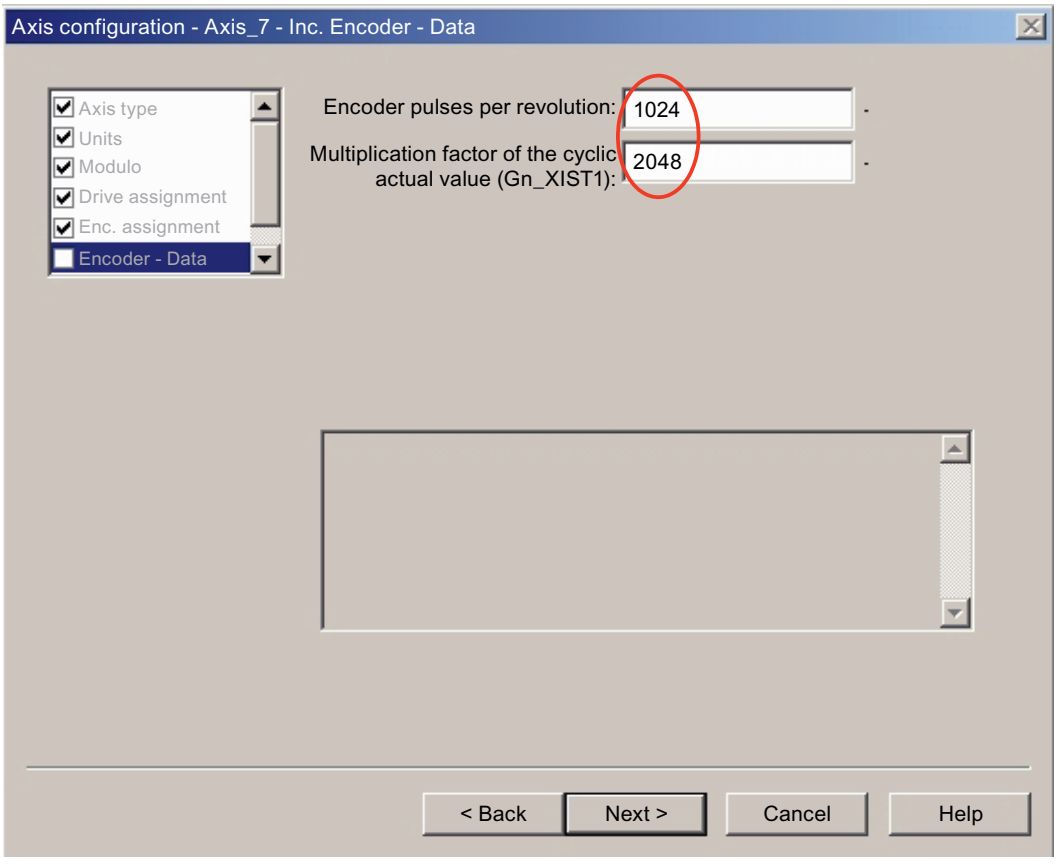


Figure 12-133 Encoder data settings

Settings:

Settings	
Encoder pulses per revolution	Encoder parameters: "Increments/revolution (resolution)"
Multiplication factor of the cyclic actual value (Gn_XIST1)	2^(STEP 7, HW Config: "Reserved bits for fine resolution") = 2^{11} = 2048

Overview: SIMOTION encoder parameters

Parameters: TypeOfAxis > Encoder_1	Value
encoderTyp	SENSOR_INCREMENTAL
encoderMode	RECTANGLE_TTL
encoderSystem	ROTATORY_SYSTEM
IncEncoder > incResolution	1024
IncEncoder > incResolutionMutiplierCyclic	2048 [1]
[1] 2048 = 2^{11}; corresponds to STEP 7, HW Config: "Reserved bits for fine resolution" = 11	

12.5.6 Standards and approvals

12.5.6.1 General rules

CE marking

CE	Our products satisfy the requirements and protection objectives of the EC Directives and comply with the harmonized European standards (EN).

Electromagnetic compatibility

Standards for EMC are satisfied if the EMC Installation Guideline is observed.

SIMOTION products are designed for industrial use in accordance with product standard DIN EN 61800-3, Category C2.

cULus Approval

c UL us	Listed component mark for United States and the Canada Underwriters Laboratories (UL) according to Standard UL 508, File E164110, File E115352, File E85972.

You can find more information on the respective device on the Internet at http://database.ul.com/cgi-bin/XYV/template/LISEXT/1FRAME/index.htm. Enter the first 7 characters of the article number under **Keyword**. Then click **Search**.

EMC

KOREA	
이 기기는 업무용(A급) 전자파적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정외의 지역에서 사용하는 것을 목적으로 합니다.	
For sellers or other users, please keep in mind that this device is an A-grade electromagnetic wave device. This device is intended to be used in areas other than home.	
The EMC limit values to be observed for Korea correspond to the limit values of the EMC product standard for variable-speed electric drives EN 61800-3 of Category C2 or the limit value class A, Group 1 to CISPR11. By implementing appropriate additional measures, the limit values according to category C2 or limit value class A, Group 1, are observed. For this purpose, additional measures, such as e.g., the use of an additional RFI suppression filter (EMC filter) may be necessary. In addition, measures for EMC-compliant configuration of the plant are described in this Manual and/or the "EMC Installation Guideline" Configuration Manual. Please note that ultimately it is always the label on the device that provides the decisive information on the compliance with standards.	

Declaration of conformity

The current Declaration of conformity is available on the Internet at Declaration of conformity.

12.5.6.2 Residual risks of power drive systems

Residual risks of power drive systems

The control and drive components of a drive system are approved for industrial and commercial use in industrial line supplies. Their use in public grids requires a different configuration and/or additional measures.

These components may only be operated in closed housings or in higher-level control cabinets with protective covers that are closed, and when all of the protective devices are used.

These components may only be handled by qualified and trained technical personnel who are knowledgeable and observe all of the safety instructions on the components and in the associated technical user documentation.

When assessing the machine's risk in accordance with the respective local regulations (e.g. EC Machinery Directive), the machine manufacturer must take into account the following residual risks emanating from the controller and drive components of a drive system:

1. Unintentional movements of driven machine components during commissioning, operation, maintenance, and repairs caused by, for example:
   - Hardware faults and/or software errors in sensors, controllers, actuators, and connection systems
   - Response times of the controller and drive
   - Operating and/or ambient conditions not within the scope of the specification
   - Condensation / conductive contamination
   - Parameterization, programming, cabling, and installation errors
   - Use of radio devices / cellular phones in the immediate vicinity of the controller
   - External influences/damage
2. In the event of a fault, exceptionally high temperatures, including open fire, as well as emissions of light, noise, particles, gases, etc. can occur inside and outside the converter, e.g.:
   - Component malfunctions
   - Software errors
   - Operating and/or ambient conditions not within the scope of the specification
   - External influences/damage

   Inverters of the Open Type/IP20 degree of protection must be installed in a metal control cabinet (or protected by another equivalent measure) such that the contact with fire inside and outside the inverter is not possible.

3. Hazardous shock voltages caused by, for example:
   - Component malfunctions
   - Influence of electrostatic charging
   - Induction of voltages in moving motors
   - Operating and/or ambient conditions not within the scope of the specification
   - Condensation / conductive contamination
   - External influences/damage
4. Electrical, magnetic and electromagnetic fields generated in operation that can pose a risk to people with a pacemaker, implants or metal replacement joints, etc. if they are too close
5. Release of environmental pollutants or emissions as a result of improper operation of the system and/or failure to dispose of components safely and correctly

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

Note

The components must be protected against conductive contamination (e.g. by installing them in a control cabinet with degree of protection IP54 according to IEC 60529 or NEMA 12).

Assuming that conductive contamination at the installation site can definitely be excluded, a lower degree of cabinet protection may be permitted.

For more information about residual risks of the components in a drive system, see the relevant chapters in the technical user documentation.

12.5.7 ESD guidelines

12.5.7.1 ESD definition

What does ESD mean?

Electrostatic sensitive devices (ESDs) are individual components, integrated circuits, modules or devices that may be damaged by either electrostatic fields or electrostatic discharge.

NOTICE

Damage caused by electric fields or electrostatic discharge

Electric fields or electrostatic discharge can result in malfunctions as a result of damaged individual parts, integrated circuits, modules or devices.

- Only pack, store, transport and send electronic components, modules or devices in their original packaging or in other suitable materials, e.g conductive foam rubber or aluminum foil.
- Only touch components, modules and devices if you are first grounded by applying one of the following measures:
  - Wearing an ESD wrist strap
  - Wearing ESD shoes or ESD grounding straps in ESD areas with conductive flooring
- Only place electronic components, modules or devices on conductive surfaces (table with ESD surface, conductive ESD foam, ESD packaging, ESD transport container).

12.5.7.2 Electrostatic charging of individuals

Any person who is not conductively connected to the electrical potential of the environment can accumulate an electrostatic charge.

This figure indicates the maximum electrostatic charges that can accumulate on an operator when he comes into contact with the indicated materials. These values comply with the specifications in IEC 801-2.

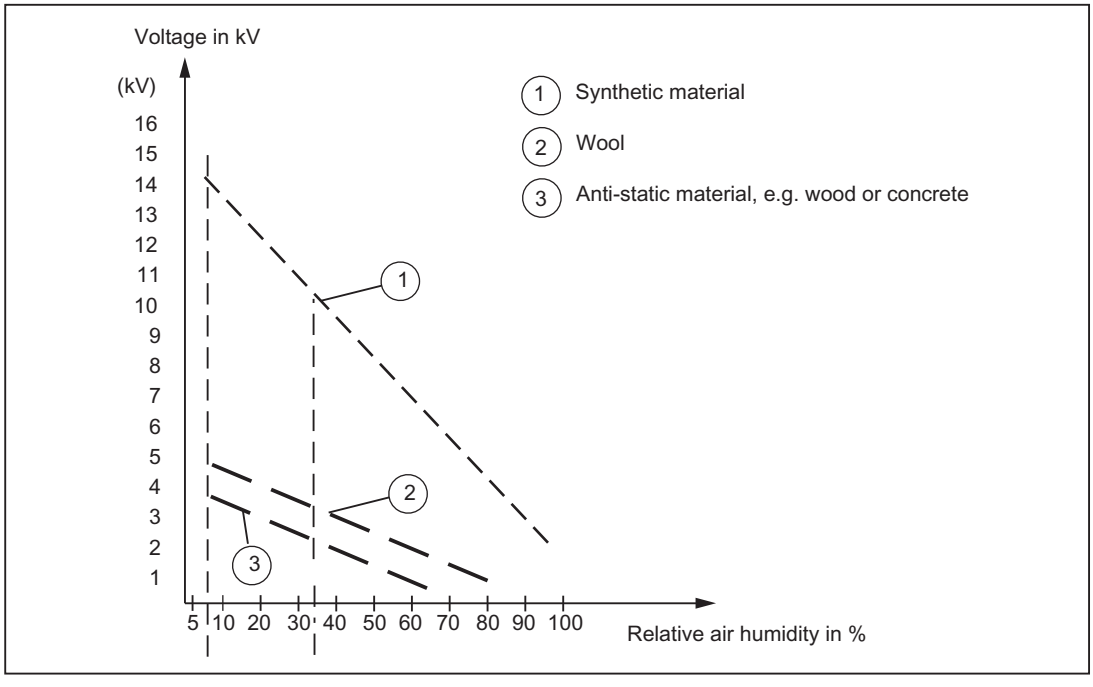


Figure 12-134 Electrostatic voltage that can accumulate on operating personnel

12.5.7.3 Basic measures for protection against discharge of static electricity

Ensure sufficient grounding

When working with electrostatic sensitive devices, make sure that the you, your workstation, and the packaging are properly grounded. This prevents the accumulation of static electricity.

Avoid direct contact

You should only touch ESD components if unavoidable (for example, during maintenance work). When you touch modules, make sure that you do not touch either the pins on the modules or the printed conductors. If you follow these instructions, electrostatic discharge cannot reach or damage sensitive components.

If you have to take measurements on a module, make sure that you first discharge any static that may have accumulated in your body. To do this, touch a grounded metal object. Only use grounded measuring instruments.

12.6 SIMATIC NET (Win7/Win10) for SIMOTION

12.6.1 Introduction

12.6.1.1 Overview

What is SIMOTION?

SIMOTION is an extensive system for the automation of production machines focusing on motion control.

SIMOTION comprises:

- SIMOTION SCOUT,
  An engineering system for creating a project by:
  - Configuring, programming, parameter assignment
  - Graphical or text-based programming
  - Project download to SIMOTION P, C, D

  The project also contains the hardware configuration and user data.
- SIMOTION Kernel
  A kernel for various HW platforms.

What is SIMATIC NET?

SIMATIC NET provides an OPC server as a standard component that enables access to the most diverse communication partners via the open OPC interface.

For a detailed description of SIMATIC NET, please refer to the current documentation CD "SIMATIC NET Manual Collection".

Which functionality is provided by SIMATIC NET for SIMOTION?

SIMATIC NET for SIMOTION provides:

- Access via OPC Data Access, enabling the user to read and write variables of a SIMOTION device.
- OPC Alarms and Events which can be used to receive alarms and events of a SIMOTION device.
- Gateway configuration.
- A SIMOTION OPC File Manager for the conditioning of network information relating to several SIMOTION SCOUT projects so that they can be simultaneously processed by the SIMATIC NET OPC server.

12.6.1.2 Schematic diagram at the design stage

Overview of design stage

The figure below exemplifies an arrangement of the relevant software for the creation of an OPC Client application on an engineering PC/PG.

During the design stage, there is no communication between the PG/PC and the SIMOTION device (OFFLINE mode). A connection is not required.

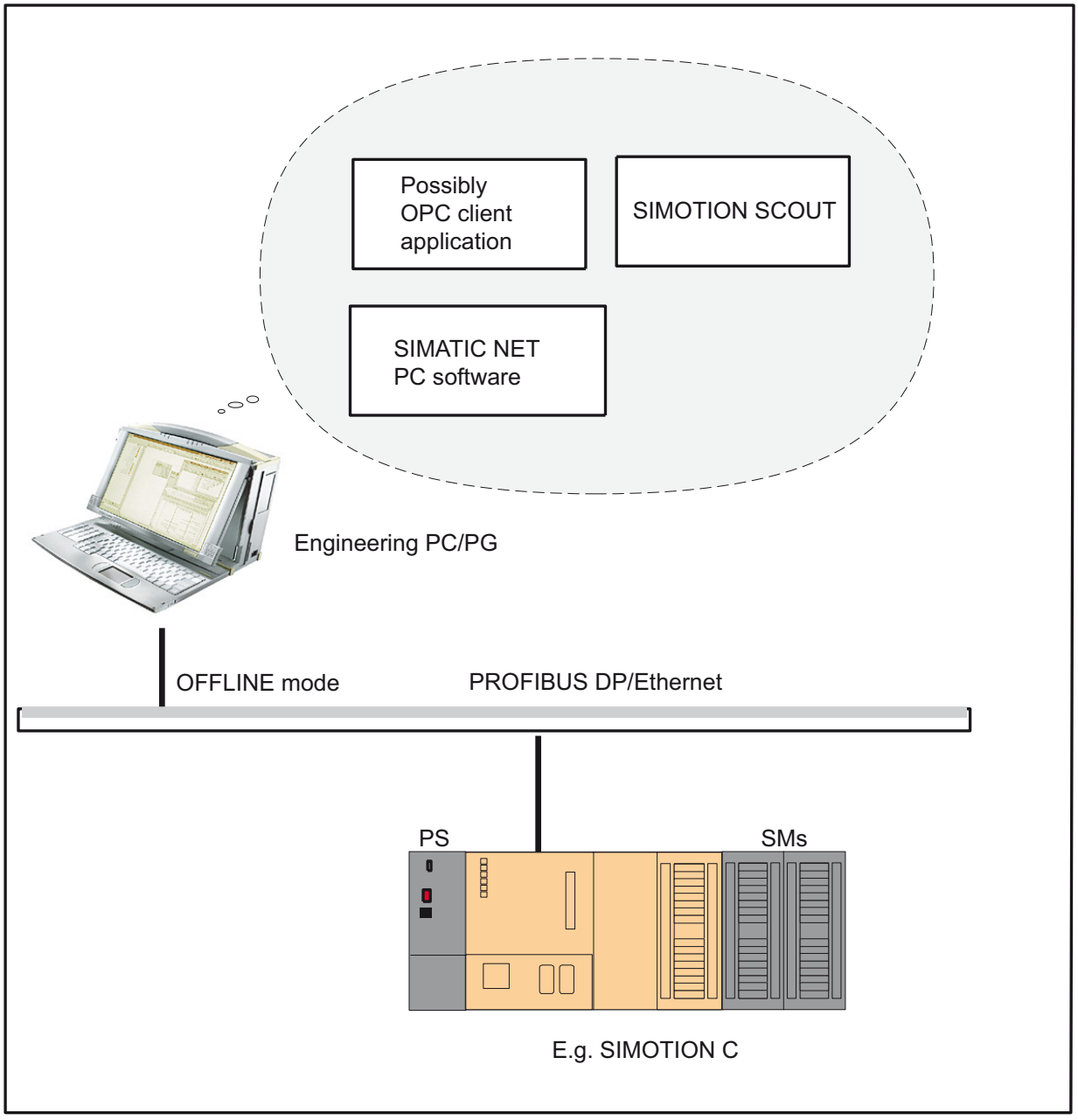


Figure 12-135 Design stage overview (example)

12.6.1.3 Schematic diagram at runtime

Runtime overview

The following figure illustrates the arrangement of an OPC Client with the relevant software on an HMI PC, the kernel and the user data on the SIMOTION device during runtime.

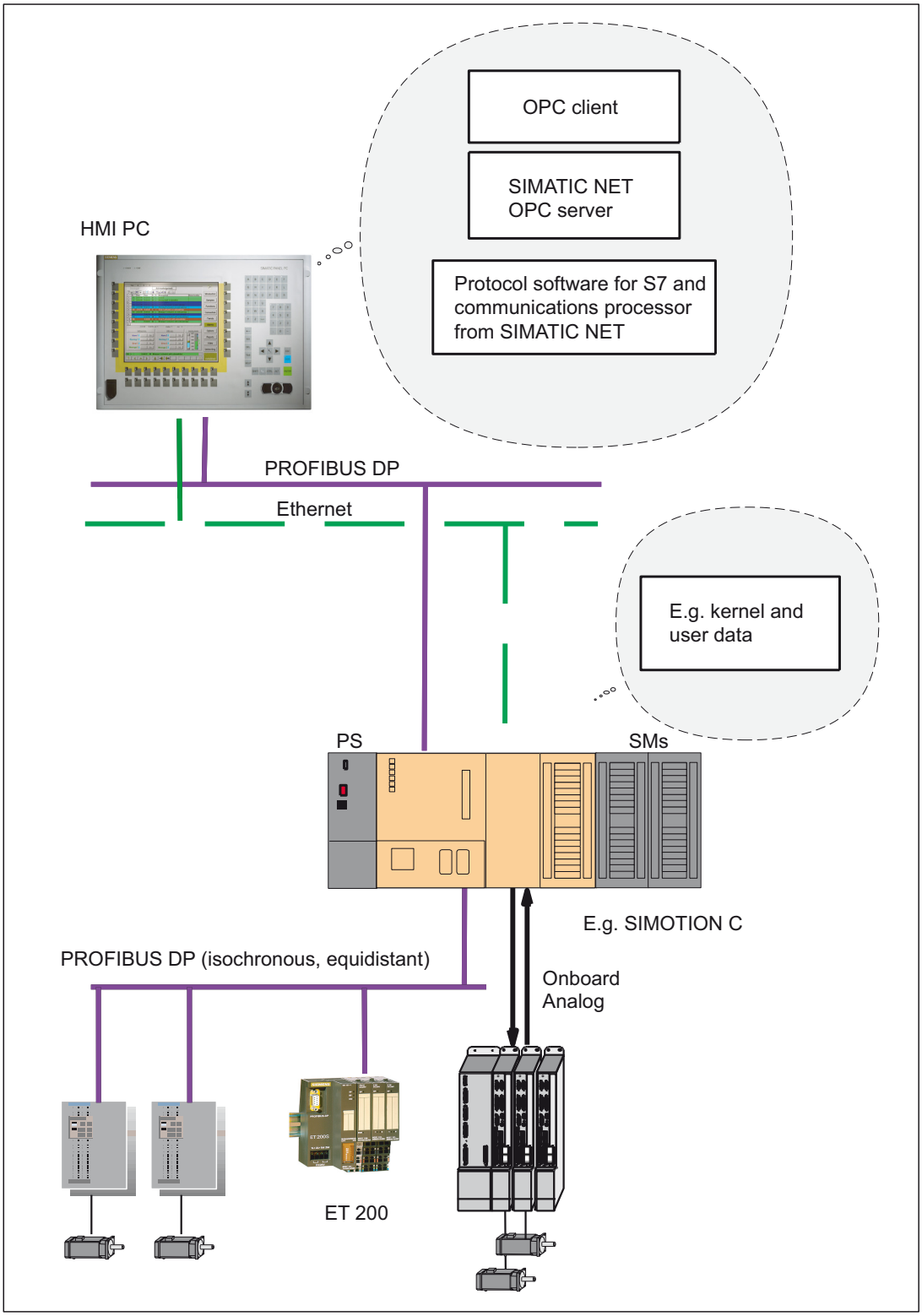


Figure 12-136 Overview at runtime (example)

12.6.2 Installation Guide

Note

This "SIMATIC NET (Win7/Win10) for SIMOTION" Product Information, Edition 08/2018, is valid for Windows® 7 64-bit and Windows® 10 64-bit operating systems.

The "SIMATIC NET (Win7/Win8.1) for SIMOTION" Edition 12/2014 is valid for Windows® 7 32-bit and 64-bit and Windows® 8.1 32-bit and 64-bit operating systems.

The "SIMATIC NET (XP/Win7) for SIMOTION" Edition 05/2013 is valid for Windows® XP and Windows® 7 32-bit and 64-bit operating systems.

The "SIMATIC NET for SIMOTION" Product Information, Edition 05/2009, is valid for Windows® 2000 and Windows® XP operating systems.

12.6.2.1 Hardware and software requirements at the design stage

Hardware requirements at the design stage

Configuration	Minimum requirement
Processor	Intel Pentium III or compatible, 1 GHz
Main memory	512 MB RAM
Screen resolution	1024 x 768 pixels

Software requirements at the design stage

- as of SIMOTION SCOUT V4.2, Windows® 7 32-bit
- as of SIMOTION SCOUT V4.3, Windows® 7 64-bit
- as of SIMOTION SCOUT V5.2, Windows® 10 64-bit

The installation requirements for the product are described in the SIMOTION and SIMATIC NET documentation (SIMATIC NET Manual Collection).

12.6.2.2 Hardware and software requirements at runtime

Hardware requirements at runtime

The hardware requirements of SIMATIC NET as of Version 13 apply.

Software requirements at runtime

- Client/Server
  - as of Windows® 7 64-bit or Windows® 10 64-bit according to the compatibility list **http://support.automation.siemens.com/WW/view/de/18857317**
  - SIMATIC NET PC software as of Version 12 and higher

  SIMOTION C
  - SIMOTION Kernel (kernel included with product shipment)
- SIMOTION P
  - SIMOTION Kernel (kernel included with product shipment)
- SIMOTION D4xx
  - SIMOTION Kernel (kernel included with product shipment)

12.6.2.3 Required licenses

Licenses dependent on communications path

Depending on the communication path used for OPC communication with SIMOTION RT, you will require one of the following SIMATIC NET licenses:

For communication with SIMOTION RT via Ethernet:

- SOFTNET-IE S7 V15 software for S7-, S5-comp. communication, OPC, PG/OP communication (DVD): 6GK1704-1CW15-0AA0
- SOFTNET-IE S7 V15 software for S7-, S5-comp. communication, OPC, PG/OP communication (download): 6GK1704-1CW15-0AK0
- SIMATIC NET SOFTNET-IE S7 LEAN V15; SW for S7-, S5-comp. communication, OPC, PG/OP communication (DVD): 6GK1704-1LW15-0AA0
- SIMATIC NET SOFTNET-IE S7 LEAN V15; SW for S7-, S5-comp. communication, OPC, PG/OP communication (download): 6GK1704-1LW15-0AK0

For communication with SIMOTION RT via PROFIBUS:

- SOFTNET-PB S7 V15 software for S7 communication incl. FDL with OPC server: 6GK1704-5CW15-0AA0
- CP 5711 communications processor USB adapter (USB V2.0): 6GK1571-1AA00

12.6.3 Communication and Handling

12.6.3.1 Fundamental procedures

Overview

Some preparation is necessary for establishing a communication between the OPC Server for SIMATIC NET and a SIMOTION device.

Basically, you must:

Table 12-57 Fundamental procedures

When?	Step	Procedure	Comment
Design stage/ runtime	1	Configure the SIMATIC NET PC software and communications processor.	See the section titled "Configuring the OPC Server/SIMOTION device interface at runtime" (Page 8411)
Design stage	2	Create a project with SIMOTION SCOUT for the SIMOTION device.	See SIMOTION SCOUT operating guide

When?	Step	Procedure	Comment
Design stage	3	Export OPC data.	See the sections titled "Exporting OPC data during the design stage" (Page 8415) and "How is a new OPC configuration enabled when an OPC client is operating?" (Page 8432)
Design stage	4	If you have configured more than one network, you have to configure routing between these networks after file export in SIMOTION SCOUT.	See the section titled "Configuring routing with SIMOTION SCOUT" (Page 8418)
Design stage	5	If there are several projects, you must configure and integrate them with the SIMOTION OPC File Manager.	See the section titled "SIMOTION OPC File Manager" (Page 8422)
Design stage	6	Transfer/copy the exported data to the HMI PC.	See the section titled "Data transfer to the OPC client" (Page 8422)
Design stage	7	Create a Client application based on OLE/COM-DCOM.	For example in Visual Basic
Design stage	8	Download the project to the SIMOTION device.	See SIMOTION SCOUT operating guide

12.6.3.2 Configure the OPC Server/SIMOTION device interface at runtime

Storage paths for export files

Introduction

The runtime behavior of SIMATIC NET is controlled with two parameter files (see "OPC data export during the design stage" (Page 8415)).

Storage paths

The runtime environment looks for the parameter files on the following paths:

- OPC_AE.xml:
  Prior to SIMATIC NET 2007 the file could be found in the following directory:
  "<SIMATIC NET installation directory>\SIMATIC.net\opc2\bins7\simotion\xml\".
  As of SIMATIC NET 2007, SIMATIC NET makes a distinction between the following two installation directories:
  - Installation directory for programs
  - Installation directory for data
    The installation directory for data is identified from the registry using the key "HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\SIMATIC_NET\General\Paths", value "SINEC_DataPath".
    OPC_AE.xml must be stored in this installation directory inside the subdirectory \opc2\bins7\simotion\XML\.
    Accordingly, the full path looks like this:
    "<Installation directory for data>\opc2\bins7\simotion\XML\OPC_AE.xml"
- Symbol file OPC_DATA
  We recommend storing OPC_DATA in the same directory.

File name extension

Note

File name extension

The file name extension of the "OPC_DATA" file (SSD, STI, ATI) depends on the version of SIMOTION SCOUT.

Additional note

Note

The "User Data" folder has the "System Folder" property and may be hidden.

The folder can be shown by selecting the menu command "Tools > Folder Options..." in the Windows Explorer.

Configuring the interface for SIMOTION C/SIMOTION D4xx

Procedure

In order to establish a connection between an HMI PC and SIMOTION C/SIMOTION D4xx, you must carry out the following steps to configure the interface:

Table 12-58 Configuring the interface

Step	Procedure
1	Install a PROFIBUS or Ethernet communication module (e.g. CP5611) in the PC. Install the SIMATIC NET PC software.
2	Call the configuration tool for the PC station by clicking the menu item *Start->Siemens Automation->SIMATIC->SIMATIC NET->Communication settings*. Set the PC properties as indicated on the following pages. 1. Navigate to *SIMATIC NET configuration->OPC settings->Symbols*. 2. In the "Active symbol files" section, click the down arrow. – If you want to use "OPC Data" and "OPC Alarms and Events" or just "OPC Data," enter the file name of the "OPC_DATA" symbol file (see the note at "File name extension" (Page 8411)) in the "File name" field and click the "Browse" button to select the directory *"<Installation directory for data>\opc2\bins7\simotion\xml\"*. (The directory that you used last is always offered for selection by default.) Select the following file: – OPC_DATA (see the note at "File name extension" (Page 8411)) The folder has the "System Folder" property and may be hidden. The OPC server must be exited during the selection of the file. 3. In the "Additional settings for SIMOTION" section, select the communication module (e.g. "CP5611 (PROFIBUS)"). 4. Click the "Apply" button to close the dialog box. 5. In the "OPC protocol selection" dialog box, select the "S7" and "OPC UA" protocols and deselect all protocols that are not required. 6. Complete the settings for the SIMATIC NET OPC server with "Apply".

Note

If project data is changed in SIMOTION SCOUT after the export of the symbol file, you must re-export the data (consistency).

See also

OPC alarms and events for SIMOTION (Page 8430)

Configuring an interface on SIMOTION P

Procedure

In order to establish a connection between an HMI PC and SIMOTION P, you must carry out the following steps to configure the interface:

Table 12-59 Configuring the interface

Step	Procedure
1	The PROFIBUS card and the SIMATIC NET PC software are included in the product shipment.
2	Call the configuration tool for the PC station by clicking the menu item *Start->SIMATIC->SIMATIC NET->Settings->Configuration console*. Set the PC properties as indicated on the following pages. 1. Navigate to *Applications->OPC Settings->Symbols*. 2. Click the "Edit List" button in the "Symbols" dialog. The dialog window "Manage Symbol Files" is displayed. If you want to use "OPC Data" and "OPC Alarms and Events" or just "OPC Data", enter the file name of the symbol file "OPC_DATA" (see the note under "File name extension" (Page 8411)) in the field "File name" and click "Browse" to select the directory *"<Installation directory for data> \opc2\bins7\simotion\xml"* (The directory that you used last is always offered for selection by default.) Select the following files: – OPC_AE.XML – OPC_Data.idl – OPC_DATA (see the note under "File name extension" (Page 8411)) The folder has the "System Folder" property and may be hidden. 3. Exit the dialog "Manage Symbol Files" by clicking "OK" and save the files with "Apply". 4. In the "Select OPC Protocol" dialog select the S7 protocol and deselect all of the other protocols which you don't need. 5. Click "Finish" to exit the installation setup for the SIMATIC NET OPC Server.

Note

During the installation on the SIMOTION P, the access point **CP_SM_1**: must be connected to the **PC internal (local)** at "Configuration Console" after the configuration of the interface!

This access point can be set in the menu item

Start->SIMATIC->SIMATIC NET->Settings->Configuration Console.

Note

With SIMOTION V3.1 and higher, communication via **PC internal (local)** is set as default.

See also

OPC alarms and events for SIMOTION (Page 8430)

12.6.3.3 OPC data export during the design stage

OPC data export at the design stage

Exporting configured data

To declare the data configured in SIMOTION SCOUT to SIMATIC NET, you have to export all usable data by selecting the menu command *Options->OPC Data Export...* (i.e. during the design stage).

Parameters are queried during the data export (see "Parameterizing the data export" (Page 8416)).

Directory for files to be exported

The following directory is set by default as the export directory in SIMOTION SCOUT:
"<LW>:\Siemens\Step7\S7proj\<project name>\U7\Tagfiles\"

If you are using the HMI PC both during the design stage and at runtime, select the following directory:
"<Installation directory for data>\opc2\bins7\simotion\xml"

Which files are exported?

You can export the following files:

- Symbol file "OPC_DATA" (see the note under "File name extension" (Page 8411)), i.e.:
  - System variables of the device and the technology objects
  - Global device user variables
  - Symbolic I/O variables
  - Interface variables from user programs
    (for data types see "System variables")
- "OPC alarm/event" (OPC_AE.xml), i.e.:
  - Technology object alarms
  - Diagnostics buffer alarms
  - Alarm_S/Q
  - Connection information

Note

The online help in SIMOTION SCOUT describes the operator sequences required for the OPC data export in detail.

Router configuration

Once the data export is complete, you will be asked if you wish to configure gateways (for routing). "Configuring gateways with SIMOTION SCOUT" describes how to do this.

The export process

The export process is logged in SIMOTION SCOUT in the detail view on the *Symbol File Export Status Display* tab.

Notes

Note

If project data are changed in SIMOTION SCOUT after export of the symbol file, you must re-export the data (consistency).

Note

During commissioning, you must make sure that OPC data transmission is disabled on the active OPC client.

Note

After the export, the files have to be transferred/copied to the OPC client (see "Data transfer to the OPC client" (Page 8422)).

Note

If you are working with multiple projects, please see the information under "SIMOTION OPC File Manager" (Page 8422).

See also

OPC alarms and events for SIMOTION (Page 8430)

Parameterizing the data export

Overview

If the device has several interfaces, the data export requires the following parameters:

- Device
- Protocol
- Interface

Device

The device for which you must select the bus interface is displayed in the "Device" selection field. You must specify the interface settings for each device in the project. You must specify these settings in this window for each device to be connected.

Protocol

In the "Protocol" selection field, you can select between:

- PROFIBUS
- TCP/IP
- PC-INTERNAL (SIMOTION P)

Interface

Note

The selection "Interface" appears if there are several interfaces on the SIMOTION device and they have been configured using different bus addresses.

In the "Interface" selection field, you must specify the interface of the OPC server on the HMI PC which will be used to connect to the respective SIMOTION device during operation.

Each symbol name must be uniquely assignable to a hardware address (bus address).

This assignment is established via the "Interface" selection. SIMOTION SCOUT detects the communication interface and provides the following selection (e.g. for SIMOTION C):

- Select X8, for example, if the interface X8 is used on the device.
- Select X9, for example, if the interface X9 is used on the device.

⚠ WARNING

Cyclic data transfer not guaranteed

If the interface is parameterized for an **equidistant bus cycle**, this interface **may not** be used for an **OPC client**! Cyclic data transfer is not guaranteed when it is used.

Exporting "OPC_DATA" (symbol file)

Notes

The export of "OPC Data" may take some time.

By deselecting the field "Arrays with single elements" it is possible to shorten the data export. When exporting in .ATI format, deselecting always results in only the first address of the array being exported. The server implements the resolution of the single elements during runtime.

Note

OPC data export will only be possible if

- SIMATIC NET software has been installed for the engineering PC/PG (only applicable to versions prior to SIMOTION SCOUT Version V3.2)
- A project is open

Note

With an array of data type "String", the array is always exported with single elements.

Deselection of the "Arrays with individual elements" field is only possible for this data type.

As a string variable is itself already an array, only the individual elements can be exported for addressing reasons.

Note

Warning in the Symbol Editor

The following warning is displayed when you open a SIMOTION .ATI file in the Symbol Editor: "Symboleditor Warning: The following incompatible symbol names were found in the file"

You can ignore this warning: The SIMOTION .ATI file can be used without restrictions.

Exporting "OPC Alarm/Event"

Notes

Note

No user-defined diagnostics buffer contents are exported with "OPC Alarm/Event", but only the specific SIMOTION diagnostics buffer contents. OPC can access these exported text items.

See also

OPC alarms and events for SIMOTION (Page 8430)

Routing configuration with SIMOTION SCOUT

Introduction

If you have configured several networks with NetPro and wish to access a SIMOTION device via a router to a SIMATIC OPC Server, you have to reconfigure this router **once more** in a dialog after file export in SIMOTION SCOUT.

Configuring a router

The "Configure Router" dialog displays the following:

- All configured networks
- All SIMOTION devices contained in the project

First select a location of the OPC Server and then the first router for each node to be addressed via the OPC Server.

The following diagram illustrates a routing configuration.

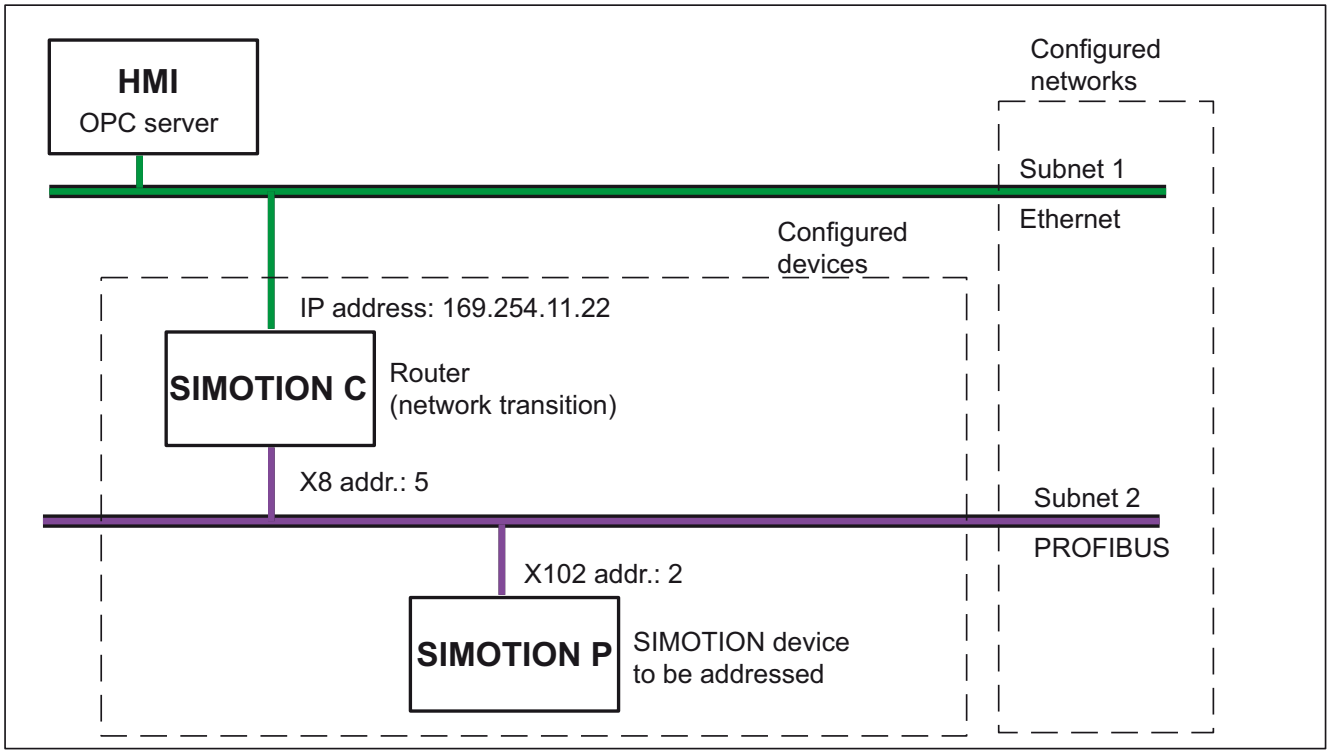


Figure 12-137 Sample configuration for routing

Network configuration for SIMOTION P

With SIMOTION P, you already require the network configuration if you want the OPC Server on the P device to access both the runtime in the SIMOTION P device itself and another SIMOTION device which has been networked via PROFIBUS.

(The reason is that there is an internal "router" between the OPC Server under Windows and the SIMOTION P Runtime. Runtime SIMOTION P has access to the PROFIBUS interface.)

In this case, you need the SIMOTION P device as a router; see the figure below.

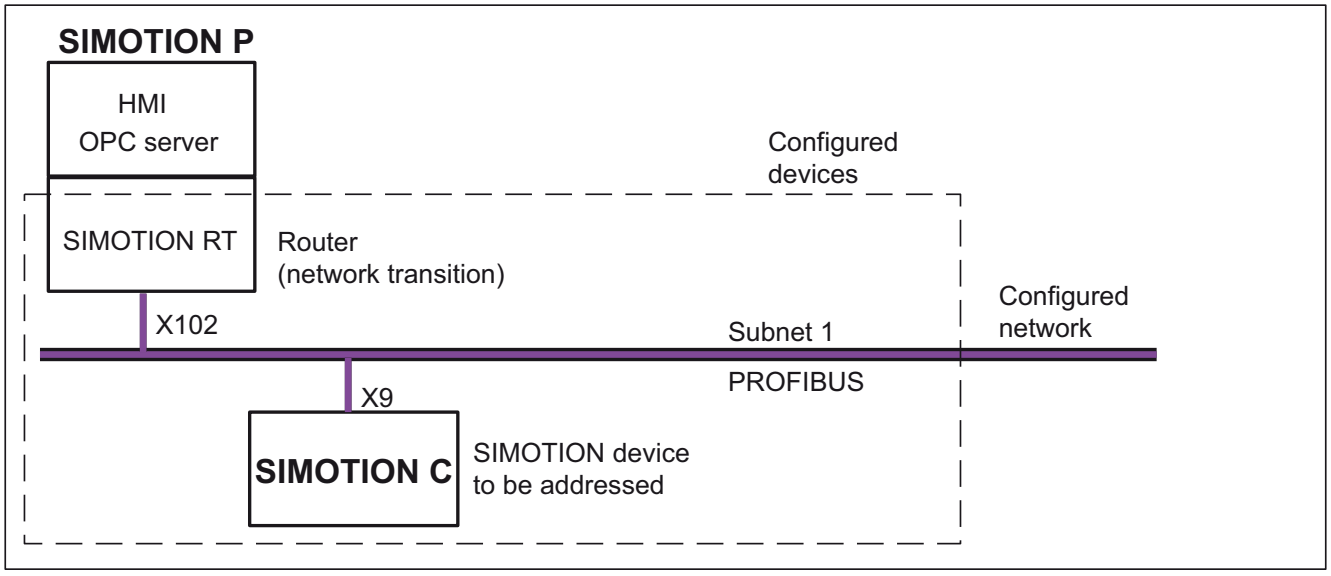


Figure 12-138 Example: SIMOTION P

Network configuration for SIMOTION D4xx

With SIMOTION D4xx the integrated drive is defined as a separate device.

Create a routed connection for this integrated drive as part of the network configuration.

Several projects internetworked

Note

If there are several internetworked projects, please note the following description.

To communicate with the SIMOTION devices in a network interconnection through several projects, the SIMATIC NET OPC Server must be forwarded the following files and items of information:

- "OPC Alarm/Event" files, which may originate from several SIMOTION SCOUT projects
- Time zones
- Router

An auxiliary program, "SIMOTION OPC File Manager", enables you to configure these data for the SIMATIC NET OPC Server. This user program is described under "SIMOTION OPC file manager."

Variables for consistency check

Introduction

When exporting the OPC data from SIMOTION SCOUT, a consistency value is also transferred.

Using this value, the OPC client can check whether this value agrees with the consistency value saved in the SIMOTION device.

Variables for determining the consistency status

To determine the consistency status in relation to a SIMOTION device, the server provides the following variables:

- &stateconsistence()
- &stateconsistenceval()

Table 12-60 Variables for checking the consistency

&stateconsistence()	&stateconsistenceval()	Meaning
"NOTCONFIGURED"	0	No consistency value was saved in the XML file.
"NOTAVAILABLE"	1	Consistency check (still) pending, because the connection was interrupted, for example.
"CONSISTENCE"	2	After the check: The configuration is consistent.
"INCONSISTENT"	3	After the check: The configuration is inconsistent.
"IGNORE"	4	After the check: The controller does not have a consistency value. In SIMOTION SCOUT, the "Activate HMI consistency check" checkbox was deselected during the transfer of the XML project data in the menu *Tools ->Settings->Download*.

Note

The OPC server can only access the consistency status if a connection is established to the respective device. A connection is established if at least one variable is read from the device.

If no connection is established, the status will be "NOTAVAILABLE".

Example for programming the variables in Visual Basic

The syntaxes for variable names for an OPC client with Visual Basic are:

- S7:[<devicename>]&stateconsistence()
- S7:[<devicename>]&stateconsistenceval()

Device name (<device_name>) corresponds to the name for the SIMOTION device as defined in the SIMOTION SCOUT configuration:

For example: "S7:[C230_2]&stateconsistence()"

The syntax for the variable names for the test software OPC SCOUT (included in the SIMATIC NET delivery kit; detailed information can be found in the SIMATIC NET documentation) is described in the following (as example).

In OPC SCOUT, both variables must be entered in one group via *Add Item* (e.g.: "S7: [C230_2]&stateconsistence()).

Then they can be monitored.

You can find OPC SCOUT (SIMATIC NET) in the SIMATIC NET start menu.

12.6.3.4 Data transfer to the OPC Server

Notes

Once you have exported the data, you must transfer/copy it to the HMI PC (OPC client).

The file in the HMI PC has to be located in the following directory:

"<Installation directory for data>\opc2\bins7\simotion\xml"

Note

During commissioning, you must make sure that OPC data transmission is disabled on the active OPC client.

12.6.3.5 SIMOTION OPC File Manager

Introduction

The SIMATIC NET OPC server can be used to monitor SIMOTION devices from multiple SIMOTION projects.

During an export, a symbol file and a file named OPC_AE.xml is generated for each SIMOTION project.

The OPC server can handle several symbol files generated for the files exported from SIMOTION projects, but only one OPC_AE.xml file.

The SIMOTION OPC File Manager closes this gap by combining the exported OPC_AE.xml files from multiple SIMOTION projects in a single file.

SIMOTION OPC File Manager

The SIMOTION OPC File Manager is an auxiliary program which offers the following functionality:

- File selection (selection of the *.xml files to be merged)
- Time zone definition
- Routing definition

The SIMOTION OPC File Manager prepares the network information related to multiple SIMOTION projects in a way that they can be processed together by the SIMATIC NET OPC server.

Delivery and program call

SIMOTION OPC File Manager is delivered as a component of SIMATIC NET.

This is called up in the SIMATIC NET start menu via *Simotion OPC File Manager*.

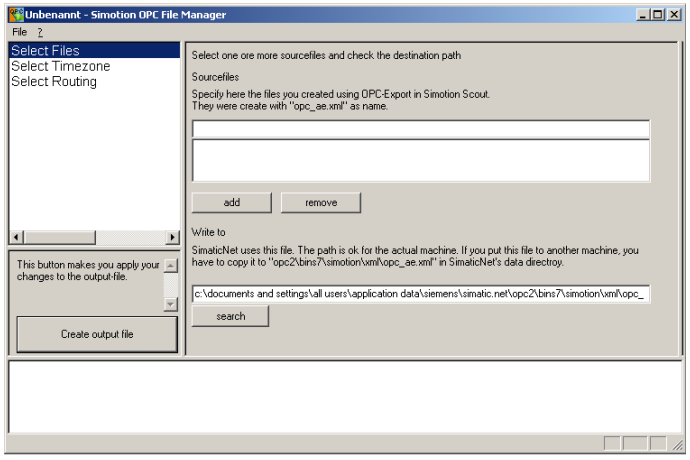


Figure 12-139 SIMOTION OPC File Manager

Note

The software contains open source code with the following copyright "Portions copyright (c) Chris Maunder, 1998".

File selection

SIMOTION OPC File Manager generates from n number of OPC_AE*.xml source files (created during OPC export in SIMOTION SCOUT from several SIMOTION SCOUT projects) a common parameter file and supplements it in places where ambiguities might occur (the respective projects contain the same alarm numbers which wouldn't be doubled in the database) with the information from which project the data originates.

If the project contains data which exclude each other, an error message is displayed in the errors area. Switchover to different screens and a generation of the target file will then be inhibited.

The following screen form represents the file selection in the SIMOTION OPC File Manager:

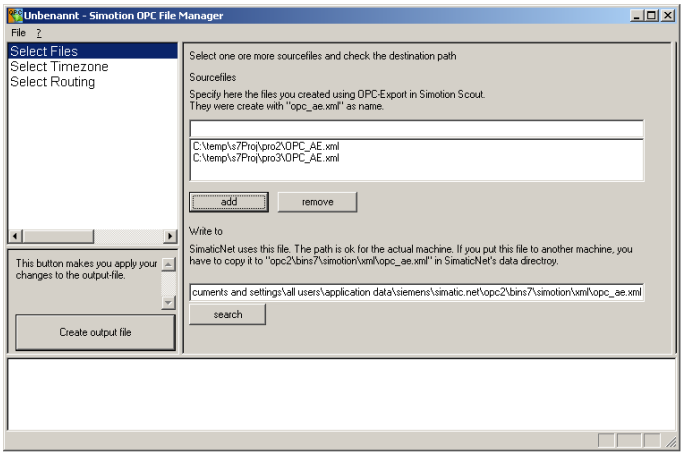


Figure 12-140 File selection

See also

Storage paths for export files (Page 8411)

Fundamental description of an application

Example

Multiple SIMOTION projects have been created for one modular machine. A single PC is to be used for monitoring purposes.

Procedure

Carry out the following steps:

1. Run an OPC export for each SIMOTION project involved and export the data to various directories.
2. Start the SIMOTION OPC File Manager.
3. On the "Select Files" screen, mark all generated OPC_AE*.xml files as source files ("add" button).
4. If necessary, enter time zones and routing information.
5. Select "Create output file" to generate the combined OPC_AE.xml.

Step 1 is completed on the engineering PC/PG. Steps 2 to 5 can be completed either on the engineering PC/PG or on the HMI PC.

Note

If you are working solely on the engineering PC/PG, the XML file generated for the OPC client must be copied to the correct directory, i.e.:

"<Installation directory for data>\opc2\bins7\simotion\xml".

Time zone definition

The time when an alarm is issued will be transmitted without a reference system in SIMOTION RT. The OPC Standard however defines that times will always be transmitted as UTC times. Besides the time, they also contain a time zone indication. The OPC server sets GMT as standard.

Time zone definition

In the "Time zone definition" dialog, a deviation from GMT can be defined for each CPU.

Figure 12-141 Time zones

Routing definition

An OPC client can communicate with several SIMOTION devices over different networks via the SIMATIC NET OPC server. The routers of the respective projects are configured accordingly with the SIMOTION OPC File Manager.

Prerequisite

Prerequisite for introducing the respective projects is:

- Communication of the SIMOTION devices with each other must be configured with *NetPro* in SIMOTION SCOUT.

Note

The router can also be configured within SIMOTION SCOUT. This process is described in "Configuring Routing with SIMOTION SCOUT".

Note

For every additional project, the router must be reconfigured with the SIMOTION OPC File Manager.

Optimum access times

In order to ensure optimum access times, data of the SIMOTION device can only pass every bus segment once. The routing strategy of SIMATIC NET (one standard router per HMI PC), on the other hand, can lead to this standard router referring back to the same segment from which the request comes.

In order to define the access to all devices even in unfavorable configurations, you can, after defining which bus segment you are operating the OPC server on, set any other SIMOTION device of this segment as the first router per SIMOTION device.

Networking example

The following example shows a networking of SIMOTION devices which communicate by an OPC server:

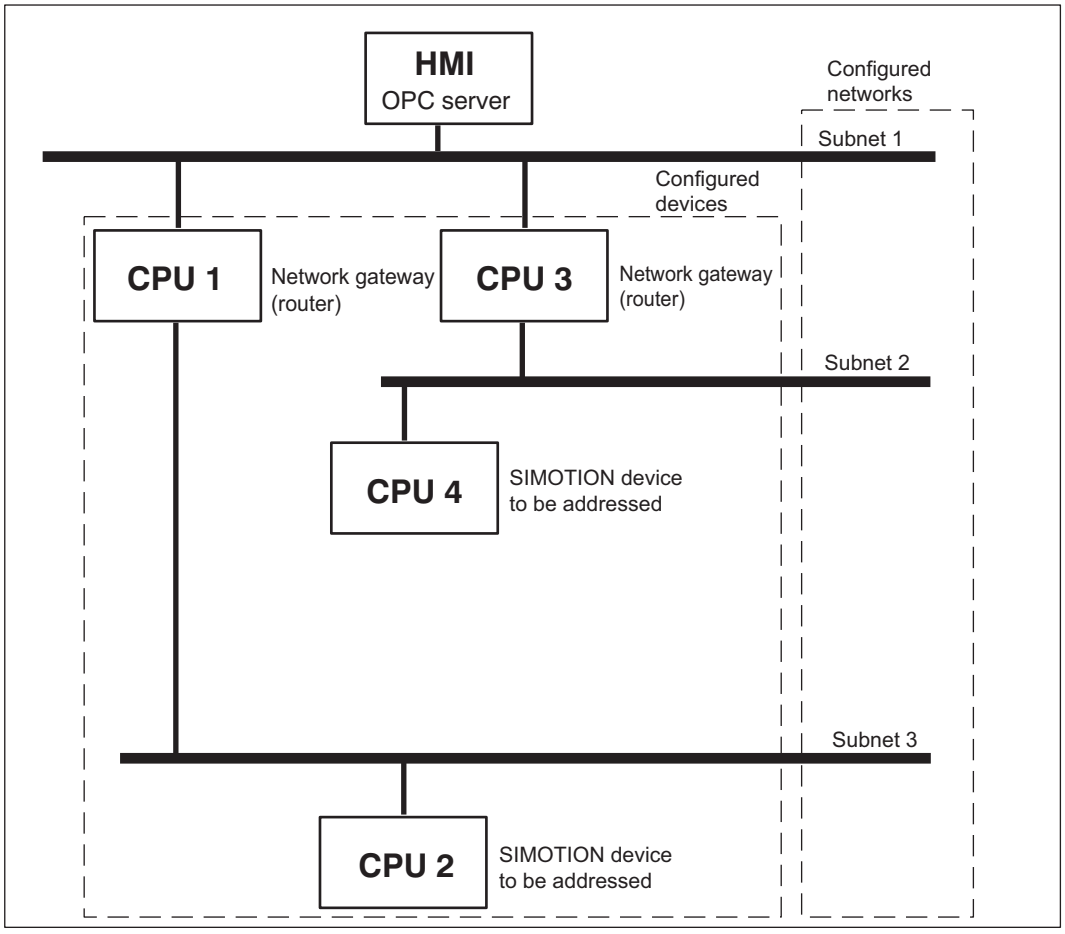


Figure 12-142 Example configuration

The OPC server requires routers for communication. The table below shows the routers of the example configuration:

Table 12-61 Routers of the example configuration

Subnet OPC server	Network node/target device	Router
Subnet 1	CPU 1	is in the same subnet
	CPU 2	CPU 1
	CPU 3	is in the same subnet
	CPU 4	CPU 3

"Routing definition" dialog

In the "Routing definition" dialog you determine:

- the subnet in which the OPC server is,
- the target devices to be addressed,
- the respective routers via which the target device can be addressed by the OPC server.

Figure 12-143 Routing definition

Note

Please observe the following general conditions:

- If you use the "Open" function in the OPC File Manager to read in an existing configuration, you must recheck the router to every target device and re-enter it if necessary under "Routing definition". With the "Open" function, you open a parameter file in which the OPC File Manager notes which exports it is processing and what appropriate information has already been entered. This parameter file is not identical with the "project export" and the merged "group export".
- When you enter **routed connections** with the "routing definition" function, you must select the **subnet** explicitly in the "Subnets" dropdown list **before** you process the routers.

12.6.4 System Features

12.6.4.1 System variables

Introduction

To obtain more precise information on the exported data, an XML export may be initiated in SIMOTION SCOUT using the menu item *Project->Save and Export*.

This XML file can then be viewed using an XML capable Web browser (e.g. MS Internet Explorer 5 or Netscape 6.1).

The variables contained in this file are, amongst others, limit values and file types, and for enums (enumerator data types) the defined values.

Note

To use these enums within an OPC Client environment, a descriptive file (idl file) is generated during OPC data export via the menu item *Options->OPC Data Export...* in addition to the symbols file.

A description of the limit values for SIMOTION system variables can be found in the reference lists, named "SIMOTION Technology Package CAM System Variables".

Data types

The following data types are available in SIMOTION:

Table 12-62 Data types

SIMOTION	Bit width	Sign
BOOL	1 bit	-
BYTE/USINT	8 bits	-
SINT	8 bits	Yes
WORD/UINT	16 bits	-
INT	16 bits	Yes
DWORD/UDINT	32 bits	-
DINT	32 bits	Yes
REAL	32 bits	Yes
LREAL	64 bits	Yes
DATE	64 bits	-
TOD (Time of Day)	32 bits	no
DT (Date and Time)	64 bits	-
TIME	32 bits	Yes
STRING	1 byte/character, max. length -> 254 characters	
Array	Note: Only the data types listed here are valid for an "array".	

Notes

Note

If ProTool/Pro CS is used as an OPC Client, this application will only accept variables in INTEGER format as range pointers.

Note

To determine the consistency status in relation to a SIMOTION device, the server provides the following variables:

- &stateconsistence()
- &stateconsistenceval()

OPC data for SIMOTION

Note

Please note the following general conditions and function restrictions:

- When asynchronously writing a ReadOnly variable, no error message is returned although the value cannot be and is not written. (When synchronously writing a ReadOnly variable, on the other hand, an error message is returned.)
- The valid value range for variables of the type "Date" and "DT" is not maintained via OPC. Values can be written to the variables that are outside of the valid value range. The valid value range for Simotion and OPC is from 1992-01-01 to 2089-12-31.

See also

Variables for consistency check (Page 8421)

12.6.4.2 OPC alarms and events for SIMOTION

Notes

Note

Please note the following general conditions and function restrictions:

- User-defined diagnostic buffers are not exported with "Export OPC data". Therefore no message text is supplied for these events.
- The OPC client that starts the OPC server also receives diagnostic buffer entries that were entered before this time.
- Only **one** language setting for **all** clients:
  Alarm texts are sent to all clients in the same language, namely the language that was set last. The process is independent of the previously set different "LocalIDs".
- A maximum of 40 "Alarm_S" are permitted for OPC alarms and events for SIMOTION:
  If more than 40 different alarms of the "Alarm_S" category are triggered in rapid succession, only the first 40 alarms are transmitted.
- Restriction when using the Automation Wrapper, e.g. with VB OPC clients:
  If a client instances several subscriptions by the Automation Wrapper "sopcdaauto.dll", the subscription activated **first** always gets the events for both subscriptions, i.e. the subscriptions activated later receive no alarms.
- When you disconnect the OPC client from the OPC server, you must wait at least 15 seconds before you re-establish the connection.

Note

Alarm status at "Alarm_SQ"

If an "Alarm_SQ" comes and goes in rapid succession and this alarm is then acknowledged, the status of this alarm may be at "Acknowledged incoming" instead of "Acknowledged outgoing" (only relevant if the time between "Incoming" and "Outgoing" is less than 10 ms).

Note

"ALARM_SQ" alarms with "Outgoing" status show sporadically false time stamp

When connecting an OPC_AE client ("OPC Alarm/Event" client) to an operating SIMOTION controller, the existing alarms are queried (refresh functionality). With "ALARM_SQ" alarms with "Outgoing" status (only acknowledgement required) a false time stamp (date and time) may be displayed sporadically.

12.6.4.3 Consistent data access

General

Access to a data item of a simple SIMOTION device data type is always consistent.

If you want to access several individual data items or arrays in the SIMOTION device (e.g. positions of several axes), user support is required to ensure consistency.

The SIMOTION ST Programming Guide describes in a programming example, how the client application (e.g. an HMI terminal) ensures consistent data access with the SIMOTION device by means of mutual queries.

12.6.5 Tips

12.6.5.1 Programming tips

If you are using an OPC client for creating, then consider the following:

Arrays

If arrays are greater than a PDU (Protocol Data Unit) (currently always 480 bytes on all SIMOTION HW platforms, of which 22 bytes are assigned to the header), we recommend the following:

- Message frames with a user data length greater than 458 bytes need to be split into two or more message frames (e.g. in the case of 540 bytes, into 240 bytes and 300 bytes).
- Use "group.syncRead()" once instead of "item.Read()" several times.

If the data changes within the space of two read access attempts to the PDUs in the array, the array will be inconsistent.

Application solution:

See the SIMOTION ST Programming Manual, "Consistent data access with HMI devices".

Time for data transmission

If you are running a SIMOTION device with multiple OPC clients (user interfaces) on one HMI PC, the data flow will be faster than with a single OPC client.

If, having completed one job, you are waiting for the next job to start based on a timer, you will need to take account of a Windows-related delay of at least 10 to 15 ms.

Recommendation:

You should use "group.syncRead()" to read several arrays rather than sending multiple individual read calls with "item.Read()".

Consistent data access

Access to a data item of an elementary data type in the SIMOTION device is always consistent.

However, if you wish to access several single data items or arrays in the SIMOTION device (e.g. positions of multiple axes), you will need user support to ensure consistency.

The SIMOTION ST Programming Manual uses a programming example to describe how the client application (e.g. HMI device) ensures consistent data access by means of mutual data querying with the SIMOTION device.

Protocol

Activate only the "S7 protocol" on the OPC server and deactivate all other protocols (see also "Configuring the OPC server/SIMOTION device interface at runtime" (Page 8411)). In other words, deactivate protocol multiplexers.

12.6.5.2 How can a new OPC configuration (OPC Data, OPC Alarm/Event) be initialized when an OPC Client is running?

Requirement

No fundamental changes must have been made to the configuration.

Example

An example application for **no** fundamental changes would be:

Within the SIMOTION SCOUT project you delete a variable which has not been declared to the OPC client. Deleting the variable causes the address range to shift in the project.

Procedure

To activate the new OPC configuration by transferring/copying the new project data to the OPC client (see also "Data transfer to the OPC client" (Page 8422)), proceed as follows:

1. Start the file export (see also "Exporting OPC data during the design stage" (Page 8411)).

   Note

   For this export, the target directory must not be the SIMATIC NET installation directory on the HMI PC. (This is because these files are opened exclusively by the OPC server when it is running.)

   Note

   If you have a number of SIMOTION SCOUT configurations for various machines/installations, and these function independently with the task of accessing all devices from a single user interface (their remit might also include routing), you should combine these exported files subsequently with the OPC File Manager (see "SIMOTION OPC File Manager" (Page 8422)).

2. Make the following entry in the "Registry" section of the "sopcsrvr.ini" file (<Installation directory for SIMATIC NET>\SIMATIC.net\opc2\bin):

   ```
   [Registry]
   AllowServerStart=0
   ```

3. Stop the OPC server. You can do this in several ways:

   – You can quit all clients so that the OPC server stops and you copy the new project data to the OPC client.
   Comment: The in-process DP OPC server cannot be stopped in the "Configuration Console" under "Applications->OPC Settings->Quit OPC Server->Stop".
   (This server is activated the first time an OPC client accesses data.)

   To stop the OPC server even if the OPC client is running, proceed as follows:

   – Set the "ForceShutdown" entry in the Windows registry on the HMI PC to the value "1". The entry is located in the following directory:
   MyComputer\HKEY_LOCAL_Machine\SOFTWARE\Siemens\SIMATIC\OPC \SimaticNET.
   The OPC client is informed of the quitting of the OPC server via the shutdown mechanism and will have to subsequently (see Step 6) reconnect and reinitialize.

   Note

   See the "Application example" in the Visual Basic program.

4. Transfer/copy the files exported under Step 1 to the OPC client (target hardware), to the following directory:
   "<Installation folder for SIMATIC NET>\SIMATIC.net\opc2\bins7\simotion\xml".

5. Reset the "AllowServerStart" entry in the "sopcsrvr.ini" file (<Installation directory for data> \opc2\bin) to "1" or delete the line with "AllowServerStart".

```
[Registry]
AllowServerStart=1
```

6. The OPC server will restart with the updated symbols as soon as the OPC client reconnects to the OPC server.

Note

It will not be possible to start the OPC server if the "AllowServerStart" entry is set to "0". This value could be set, for example, due to a process for activating a new OPC configuration failing or the Windows operating system being shut down in the meantime. Therefore, we recommend setting this value to "1" when Windows ramps up (see Step 2) or deleting the line in the "sopcsrvr.ini" file on ramp-up.

See also

OPC alarms and events for SIMOTION (Page 8430)

12.6.5.3 OPC communication to SIMOTION and SIMATIC S7 controller via PROFIBUS

Requirements

If OPC communication to SIMOTION and SIMATIC S7 controllers via PROFIBUS is to be performed, the following are required:

- Requirements and settings for the OPC communication with SIMOTION:
  - The controllers are on the same PROFIBUS line.
  - The configuration in SIMOTION SCOUT has been loaded to the controller via PROFIBUS.
  - An OPC export has been performed for the configuration.
  - The symbol file OPC_DATA (see the note under "File name extension" (Page 8411) has been entered in SIMATIC NET "Set PC station" and the access point CP_SM_1 has been set to PROFIBUS/CP5611.
- Additional requirements and settings for the OPC communication with S7:
  - The S7 configuration has been loaded to the controller via PROFIBUS.
  - In SIMATIC NET "Set PC Station", the module operating mode must be set to "Configured mode" under Modules – CP5611 - General (see following figure).

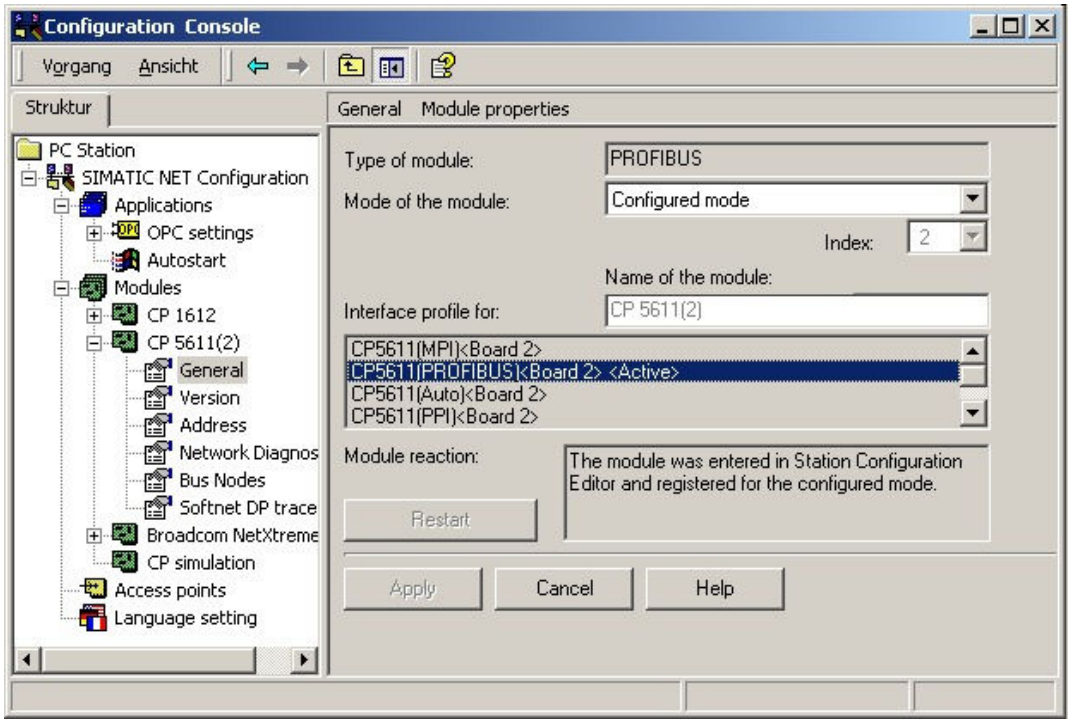


Figure 12-144 Configured mode

  - A PC station from the HW catalog must be inserted in NetPro.

– In the HW Config, an OPC server must be inserted in the PC station at Index 1 and a CP5611 PROFIBUS card must be inserted at Index 2 from the HW catalog (see following figure).

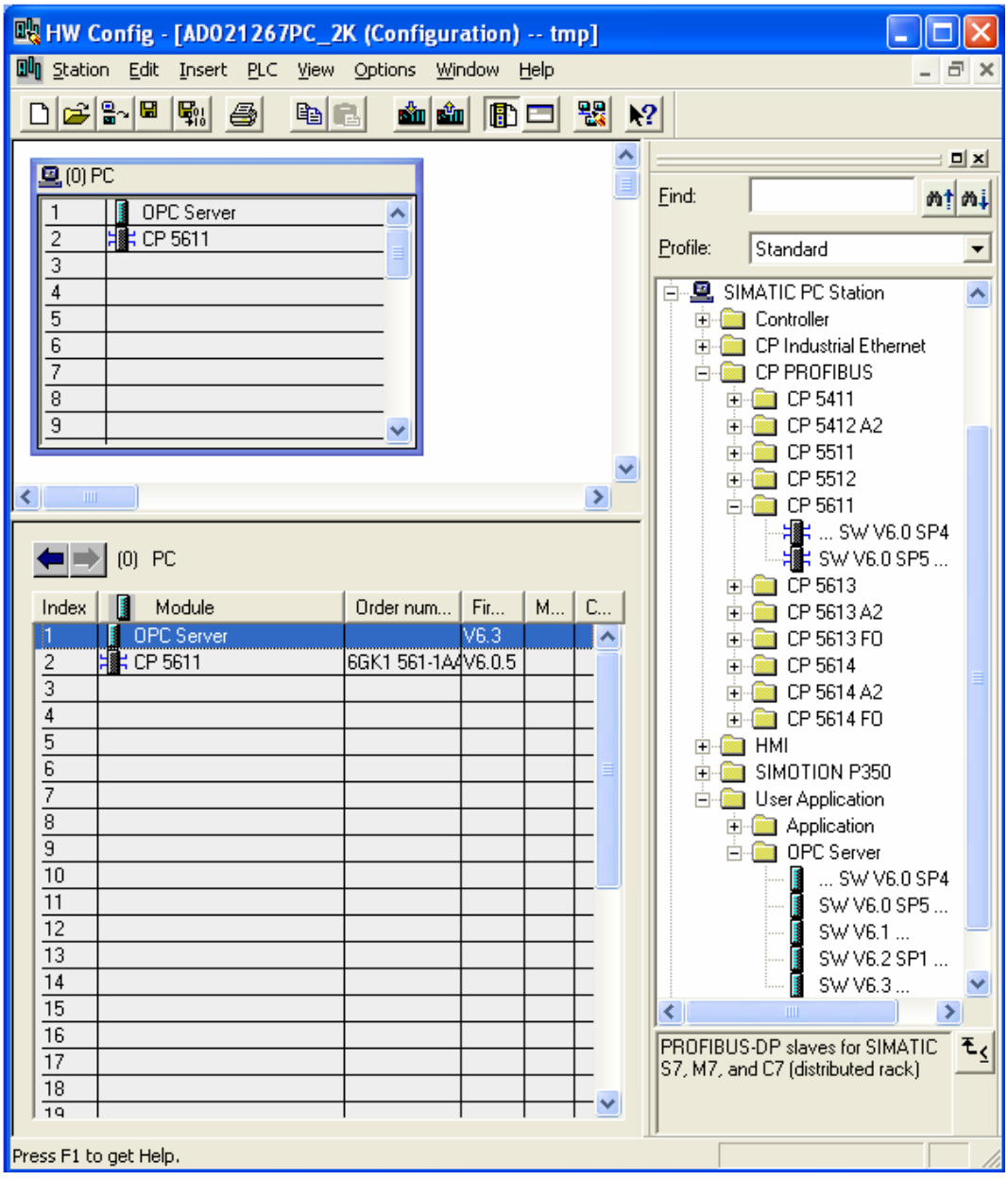


Figure 12-145 HW catalog

– After Save and compile, the PC station must be loaded to the Station Configuration Editor via the PG/PC interface "PC internal" (see following figure).

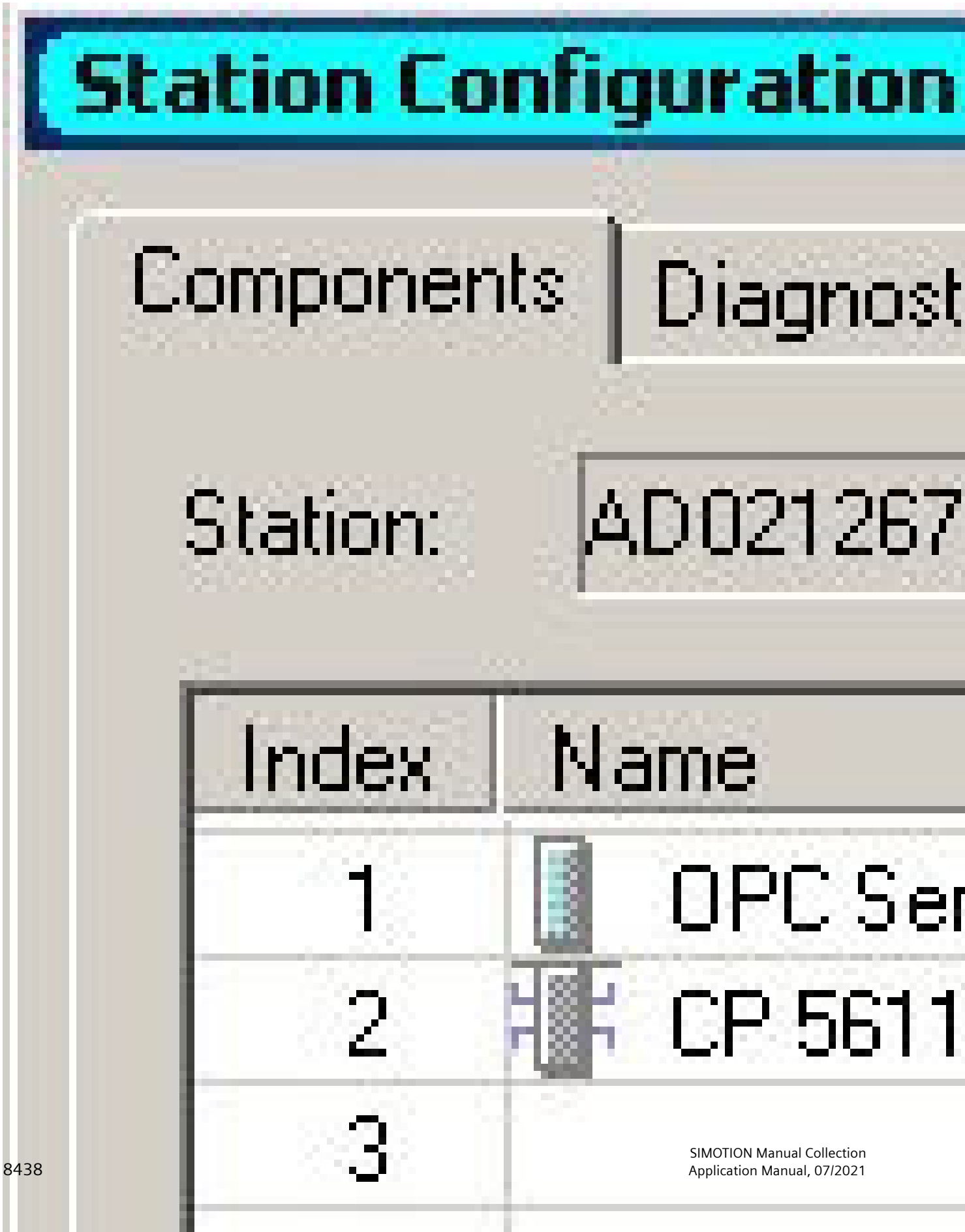

Station Configuration
Components
Diagnost
Station:
AD021267
Index
Name
1
OPC Ser
2
CP 5611
3

Figure 12-146 Station Configuration Editor

- In SIMATIC NET "Set PC Station", the access point CP_L2_1 must be set to "PC internal".
- In NetPro, an S7 connection must be created for the OPC server in the PC station (see following figure).

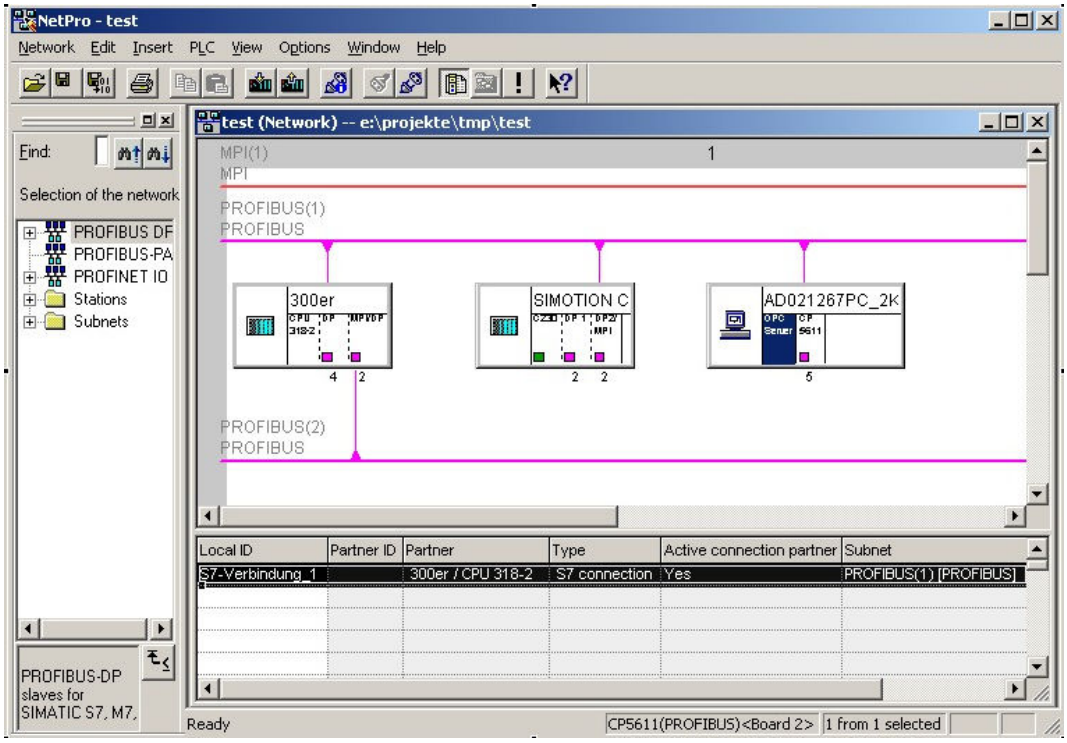


Figure 12-147 NetPro

Note

Further information on the configuration of an OPC S7 connection, can be found under SIMATIC NET "Set PC Station" - SIMATIC NET Configuration: Example: "OPC S7 PROFIBUS"

12.6.5.4 OPC via PROFINET

Introduction

This section describes the fundamentals for OPC communication via PROFINET. The corresponding SIMOTION C/SIMOTION D/SIMOTION P manuals describe what must be observed during a PROFINET configuration.

Fundamentals

This connection is used for the OPC communication via PROFINET in order to establish an OPC connection via TCP/IP. The same protocols are used as for OPC via Ethernet.

This does not effect the other communication via PROFINET between SIMOTION controllers and drives.

The OPC export with the routing and the access points is equivalent to Ethernet / TCP/IP.

It is possible to directly access a SIMOTION P with MCI-PN board or a D4x with CBE30 card from a PG/PC with Ethernet connection via a PROFINET cable.

The OPC routing between PROFIBUS and PROFINET functions from one network to the other.

The following figure represents a possible connection of the HMI PC via PROFINET.

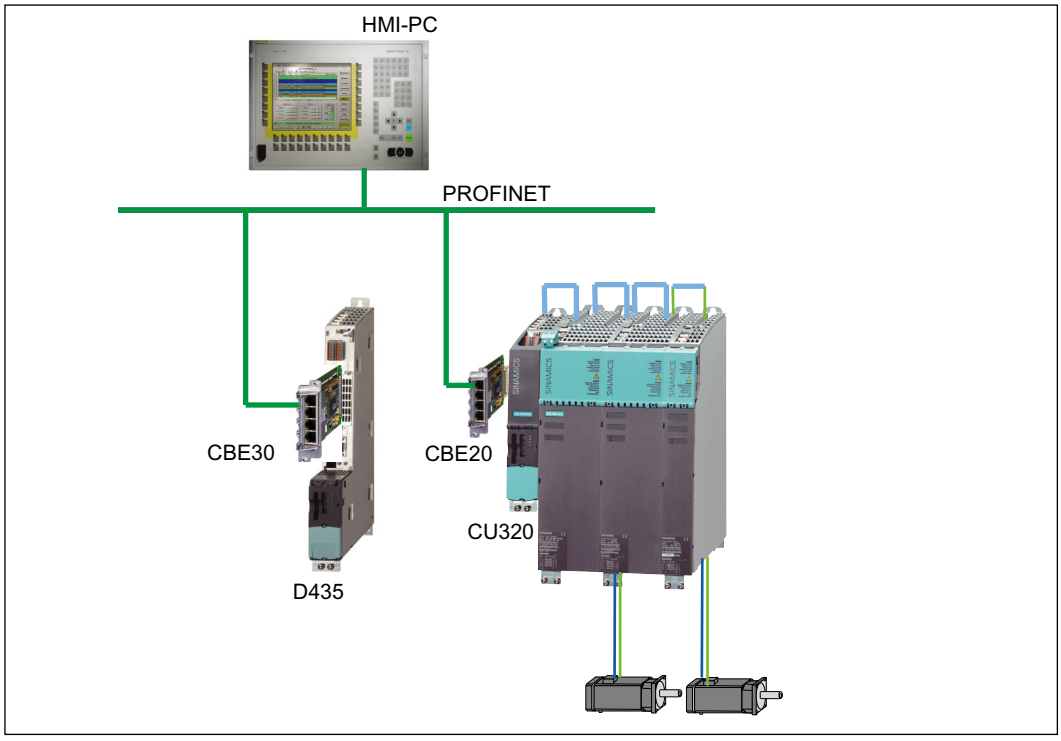


Figure 12-148 PROFINET

12.6.5.5 More tips

How do I create an OPC server in HW Config?

How to create an OPC Server in HW Config is described in the SIMATIC NET documentation on the "SIMATIC NET PC / Software CD, Electronic documentation".

If you use an OPC Server for SIMOTION P, please follow the descriptions in the SIMOTION P manual.

How do I operate the OPC SCOUT?

Among other things, the OPC SCOUT supports connection tests for the OPC client->OPC server->SIMOTION RT connection and programming of an OPC client.

You can create one or more groups from the OPC client via the OPC server among other things. You may choose any name.

In these groups, you can create one or more items. These items directly correspond to the variable names in the SIMOTION SCOUT project.

A group can be switched active or inactive.

- "Active" means that the OPC server scans the variables in the SIMOTION SCOUT in a time base and places them in the cache.
- "Inactive" means that the OPC client reads the variables directly from the SIMOTION device.

Note

The reason for bad quality in an item can be:

- HMI consistency flag set in SIMOTION RT
- Wrong project / wrong OPC configuration loaded
- Item name misspelling
- No connection to SIMOTION RT available

OPC communication with SIMOTION modules

Note

A SIMATIC PC station must not be used for OPC communication with SIMOTION modules only. The reason is that the configuration for OPC communication with SIMOTION is significantly different than that with SIMATIC modules. The procedure for mixed operation is described in "OPC communication with SIMOTION and SIMATIC S7 controller via PROFIBUS" (Page 8435).

Download in running OPC SCOUT

Note

A project download or a download in HW Config can be performed with the OPC SCOUT running. If an OPC server is configured in the component configurator in this case, the connection to the controller is broken and the OPC SCOUT must be restarted. Before restarting the OPC SCOUT it must be ensured that the configured OPC server is in the "RUN" operating state.

Going online with SIMOTION SCOUT with running Alarm client

Note

You may sometimes not be able to go online with SIMOTION SCOUT when an Alarm client is running.

In this case you must first end the alarm client or interrupt the connection to the "OPC Alarms and Events" server.

Maximum permissible number of PG/PC connections via Ethernet/PROFINET

It is possible to go ONLINE to 10 PROFINET nodes simultaneously with SIMOTION SCOUT.

The STEP 7 standard package supports a maximum of 10 simultaneous ONLINE connections via Ethernet.

As SIMOTION SCOUT uses the STEP 7 standard package drivers to go ONLINE, this number also applies here.

If SIMATIC NET is installed in addition to SCOUT, it is possible to establish connections to more than 10 nodes, as SIMATIC NET uses its own drivers that do not have this restriction.

12.6.5.6 Comparison of SIMOTION IT OPC XML-DA/SIMATIC NET for SIMOTION

Overview

In addition to SIMATIC NET for SIMOTION, there is another product,

SIMOTION IT OPC XML-DA. This package also enables you to access data and monitor operating states of the SIMOTION device via Ethernet.

The following table compares the two packages and describes the basic procedures:

Table 12-63 Basic procedure for accessing data

SIMOTION IT OPC XML–DA	SIMATIC NET for SIMOTION
No configuration (OPC export) with SCOUT required. Program variable can be activated via a switch.	OPC export with SIMOTION SCOUT required, which has to be repeated for every project change.
Symbols are interpreted in the SIMOTION device; communication in text format (XML).	Symbols are interpreted during OPC export and stored in the OPC server on the Windows system in a binary form; binary communication -> higher data throughput.
At present only SIMOTION with OPC XML DA. Access to S7 devices not possible at present.	Simultaneous access to SIMOTION and S7 devices.
Client can run on any operating system.	Based on Windows COM/DCOM technology; client and server can only run on Windows operating systems.
Communication with standard protocols (TCP/IP, XML, SOAP), no manufacturer-specific (SIEMENS) tools, drivers required on the client system.	S7 protocol used for communication, appropriate manufacturer-specific drivers required on the client.
Communication only possible via Ethernet.	Communication via PROFIBUS/MPI and Ethernet is possible.
Direct addressing via firewalls is possible.	Generally, DCOM communication not released for firewalls.

12.6.5.7 Example of an application

Example

Note

This section describes a short example of an OPC client application.

The examples for SIMATIC NET can be found in the "SIMATIC.net\opc2\samples" directory.

An OPC client connected to an OPC server is to execute the following tasks:

1. Respond to a OPC server quitting
2. Instantiate the OPC server
3. Restore the connection to the OPC server (see "How is a new OPC configuration enabled when an OPC client is operating?" (Page 8432))

Visual Basic program

The following Visual Basic program contains these steps:

```
Option Expicit
Option Base 0

Const NUM_OF_ITEMS As Long = 3

Dim WithEvents myOPCserver As OPCServer
Attribute myOPCserver.VB_VarHelpID = -1
Dim WithEvents myOPCgroup As OPCGroup
Attribute myOPCgroup.VB_VarHelpID = -1
Dim myOPCitem1 As OPCItem
Dim myOPCitem2 As OPCItem
Dim myOPCitem3 As OPCItem

Dim readData As Variant
Dim writeData(0 To 255) As Integer
Dim writeDataDINT As Long

Dim ItemIDs(NUM_OF_ITEMS) As String
Dim ClientHandles(NUM_OF_ITEMS) As Long
Dim ServerHandles(NUM_OF_ITEMS) As Long
Dim Values() As Variant
Dim Errors() As Long
Dim ErrorString As String
```

```
'called at program start
Private Sub Form_Load()
    Call Init
End Sub

'called at program end
Private Sub Form_Unload(Cancel As Integer)
    Call Destroy
End Sub

'called after connection loss or access fault
Private Sub TimerInit_Timer()
    TimerInit.Enabled = False 'disable timer
    Call Destroy
    Call Init
End Sub

'called at OPC server shutdown
Private Sub myOPCserver_ServerShutDown(ByVal Reason As String)
    Debug.Print Now & " server shutdown"
    TimerInit.Interval = 10000 'try to reconnect after 10 seconds
    TimerInit.Enabled = True
End Sub

'initialize OPC connection
Private Sub Init()

    On Error GoTo InitError

    Set myOPCserver = New OPCServer ' server
    myOPCserver.Connect ("OPC.SimaticNET")
    Set myOPCgroup = myOPCserver.OPCGroups.Add("Test") 'group
    With myOPCgroup
        .IsActive = False 'read synchronous from device
        .IsSubscribed = False 'read synchronous from
        device 'synchronous
        With myOPCgroup.OPCItems
            Set myOPCitem1 = .AddItem("P350.VARIABLE.db1", 1)
            Set myOPCitem2 = .AddItem("P350.ST_1.array1", 1)
            Set myOPCitem3 = .AddItem("P350.ST_1.dint1", 1)
        End With
```

```
    End With
    ServerHandles(0) = myOPCitem1.ServerHandle
    ServerHandles(1) = myOPCitem2.ServerHandle
    ServerHandles(2) = myOPCitem3.ServerHandle
    Debug.Print Now & " server connected"
    Timer1.Interval = 500 'poll data every 500ms
    Timer1.Enabled = True 'poll data

    Exit Sub
InitError:
    MsgBox "Error connecting OPC server: " & Err.Number & " " &
    Err.Description

End Sub
```

12.6.6 Notes on the Online Help and Documentation

Overview

Table 12-64 Online help and documentation

What to do?	Names	Content
SIMOTION SCOUT Online Help	Is part of the SIMOTION SCOUT software	Provides help for the user interface
Entire documentation on the SIMOTION software installation DVD	SIMOTION System Overview	Overview of the product family
	SIMOTION SCOUT, Operator's Guide	Describes the SIMOTION SCOUT engineering system, i.e. details on the installation and Workbench
	SIMOTION MCC, Programming Guide	Describes the graphics-based SIMOTION programming language Motion Control Chart
	SIMOTION ST, Programming Guide	Describes the text-based SIMOTION programming language Structured Text
	SIMOTION LAD/FBD, Programming Manual	Describes the programming language LAD/FBD
	SIMOTION C, Manual	Describes the hardware and commissioning
	SIMOTION Technology Functions of Motion Control (divided in several manuals and lists of references)	Describes the effects of technology objects for Motion Control and contains reference lists for system variables and functions
	SIMOTION Function Blocks, Manual	Supplement for the CP and FM modules
	SIMOTION P, Manual	Describes hardware, software and installation
	SIMOTION D4xx, Manual	Describes hardware, software and installation
All documents on documentation CD "SIMATIC NET Manual Collection"	"SIMATIC NET Manual Collection"	Amongst other subject items, details of the installation, OPC interface, OPC Server

Note

A detailed breakdown of the documentation for SIMOTION can be found in the "SIMOTION documentation overview".

12.6.7 Service and support

12.6.7.1 Service and support

Additional information

Sources of additional information about the products described in the manual are listed at "Contact" in the following table:

Requirement	Contact
Representatives and sales offices	http://www.siemens.com/automation/partner
Additional technical documentation	http://www.siemens.com/mdm
Training center	http://www.siemens.com/sitrain
Technical support	https://support.industry.siemens.com/cs/ww/en/sc
Online support request form	
Service	

Index

B

C

D

E

F

G

H

I

L

M

N

P

Q

R

S

T

U

V

W

X

Y

Z